Cloud Architecture Demystified

Understand how to design sustainable architectures in the world of Agile, DevOps, and Cloud

Keshri Asthana
Ankur Mittal

Foreword by Parminder Bansil and Ravi Pratap Singh

bpb

Cloud Architecture Demystified

Understand how to design sustainable architectures in the world of Agile, DevOps, and Cloud

Keshri Asthana

Ankur Mittal

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-262

www.bpbonline.com

Forewards

What you leave behind is not what is engraved in stone monuments, but what is woven into the lives of others.

— Pericles

It's my pleasure to know Keshri and Ankur in a relationship that has grown over the decades from being colleagues, who were fortunate to work on projects from Tokyo to New York, to thought partners, benefiting me (most of the times!) with unlimited access to their deep knowledge and context in all things architecture.

A common theme across many counters we shared is the dizzying effect of the trichotomy - disorientation brought on by the evolving technology landscape, of growing awareness that systems design being potent means of differentiation and the increasing need for certitude in an otherwise uncertain world requires practitioners (across enterprise consumers and producers of technology) be constantly catching up and keeping up for want of staying ahead. By itself, this is unsustainable. From the experience, one would default to the binary options to solve this conundrum – to reduce the amount of perceived risk (by trying to play catch-up) or by increasing our tolerance for change. However, rarely do edge scenarios help solve the complexity enterprises are challenged with today.

Cloud Architecture Demystified lays out the principles across the ends by providing a continuum of options. You will find the reasoning and rationale of interest and intrigue. Hopefully, it will help you incorporate some or several of these principles in your systems journey.

— Parminder Bansil
(Our Lifelong Mentor & Guide)

Information Technology is well known for its buzzwords and hyped trends. Using Gartner's famous Hype Cycle model, I can say many new trends don't go beyond the Hype Phase. More importantly, only few adopters make the Realization Phase in reality. Among these buzzwords is The Cloud - A panacea for all timeless CTO woes. To many, moving their datacenter to an external cloud is a job well done and

worth a celebration.

Real realization of value from Cloud implementations will need very intense strategization.

Architecture, design, code, the way we build, the way we test, the way we deploy, the way we troubleshoot/monitor may need to undergo an overhaul.

Cloud Architecture Demystified is a must read for all CTOs and Technology Heads involved in taking Cloud Strategy decisions related or executing Cloud adoption projects. It comprehensively covers all the elements that must be considered to extract the real value that Cloud offers. It has been written in simple and few words with loads of examples.

I was fortunate to have worked with Keshri on some exciting and yet challenging IT projects. He was an architect with utter clarity of how he wanted to build the software that delivers business value, is scalable, can survive the hard times software is subjected to, is debuggable and is designed to evolve forever. In my rather long career in IT, there are very few professionals who could do justice to this book and Keshri is one of them. I am glad that he took the initiative to capture all his learnings from real life situations that he experienced in his career into this book.

— Ravi Pratap Singh
(Our Lifelong Mentor & Guide)

About the Authors

- **Keshri Asthana** has been working in software development for more than 22 years, playing central roles in numerous projects as an architect, technical leader, and software engineer, delivering projects using Open source and Microsoft Technologies for big companies, including well succeeded projects in India, Japan, United States, United Kingdom and Singapore. Currently, he is Principal Program Manager with Microsoft. He is also an accomplished postgraduate completing a Masters in Business Administration and has many Microsoft certifications in Azure and Web Development technologies. Furthermore, the author participates as a speaker in Conferences and has successfully contributed to large projects run by multiple enterprises and the Government of India. He is one of the most respected architects in Microsoft and partner Ecosystem.

- **Ankur Mittal** has been working in software development for more than 21 years, playing central roles in numerous projects as an architect, technical leader, and software engineer, delivering projects using Microsoft Technologies for big companies, including well succeeded projects in India, Japan, and the United States. Currently, he is a Senior Cloud Solution Architect with Microsoft. He is also an accomplished postgraduate completing a degree in computer applications and has many Microsoft certifications in Azure and Web Development technologies. Furthermore, the author participates as a speaker in IT Conferences and has successfully contributed to large projects run by multiple startups and the Government of India. He is one of the most respected architects in Microsoft and partner organizations.

Acknowledgement

We would like to express our deepest gratitude to the following individuals who have supported and inspired us throughout the process of writing this book:

First and foremost, we would like to thank our families for their unwavering support and encouragement. Without them, this book would not have been possible.

We would also like to thank Parminder Iqbal Bansil and Ravi Pratap Singh for agreeing to spend some time in writing the forward for us. They have long been mentors and high impact business problem solvers. They have solved and architected various businesses and technical problems that their experience can be a topic for another book. Thank you, Sir's'!!

We would like to acknowledge our editor, BPB Publications, for their guidance, feedback, and patience throughout the editing process. Their expertise and insight have been invaluable.

We are grateful to our mentors. They have ensured we stay on our course and learn to contribute to the world using the best assets we have at our disposal. The lateral thought process that they have built in us has enabled us to think in multiple dimensions.

We would like to extend my thanks to the reviewers of this book, whose insightful comments and suggestions helped us to improve the manuscript.

Finally, we would like to thank my readers, whose interest and enthusiasm for this project has been a constant source of motivation.

Thank you all for your support and encouragement.

Preface

Ankur and I joined the Microsoft at the same time. We joined the Cloud Bandwagon that Microsoft was starting. Both of us came from very similar coding backgrounds, we both worked in project and product companies. We both worked in a wide spectrum of process environments; where we had, on one side CMMI Level 5 and on other side cooking and eating one day at a time. Although we had similar backgrounds, our approach varied with different sets of customers. Ankur worked with various teams of startups having altogether a different problem of scale and I worked mostly in enterprise where process and procedure is key and we both exchanged ideas how business priority impacts architecture and this is something we tried to jot down in this book!

When we joined Microsoft, we got an opportunity to work with lots of companies and a huge number of people. We got a chance to work in different combinations of people, process, and technology. Every interaction was a learning opportunity for us.

One harsh reality that we came across was diminishing architectural practices. We saw a lack of technology vision, lack of enterprise architecture, lack of software architecture and no one in an organization having end-to-end vision. There were several excuses, agile methodology, changing technology, Software Vendor Affinity etc.; none of them really standing test of time.

We saw technology decisions were more cost driven rather than other more important aspects. Technology and architecture decisions were being taken by people who did not have architectural vision. Technology leaders of today discount the importance of architecture and architects in their process and organization. We did see some organizations investing in architecture and reaping benefits. You will read some examples in the chapters.

The knowledge base that we gathered talking with people, understanding their process, their limitations, practical approaches etc. motivated us to write this book. It seems appropriate, however, to offer this little book, reflecting essentially our personal views.

Although we both originally grew up in the programming side of software engineering, we had some great mentors that we would like to thank them for as we feel they have contributed to this book too.

Our own conclusions are embodied in the essays that follow, which are intended for professional architects, technology leaders, and especially the agile architects.

Although written as separable essays, there is a central argument contained. We strongly believe that, irrespective of development method followed, architecture and end-to-end vision is a super critical and mandatory part of software development. We believe the critical need to be architecturally perseverant for conceptual integrity of an IT organization. Some of our chapters challenge some traditional thoughts and are sure to trigger out-of-box thinking.

For example, when humankind starts colonizing Mars, will address databases need to change? Will we need to add one more field in our database, "Planet"? Will it mean that all databases and all UI will need to change? Will it need another Y2K like effort?

Few of our chapters also explore other aspects of software engineering management.

The literature in architecture is abundant, but it is neither widely believed nor followed. Hence, we have tried to give references that will both illuminate points and guide the interested reader to other useful works.

Because this is a book of essays and not a text, all the references and notes have been referenced in line with the text, and the reader is urged to follow the links for detailed study on the topic.

Chapter 1: Ambivalence of Multi-Cloud – Deploying a multi-cloud platform in an enterprise can have both advantages and challenges, and it ultimately depends on the specific needs and goals of the organization. In the chapter we will try to lay out details how deploying a multi-cloud platform in an enterprise can offer many benefits, but it also poses some challenges. Organizations should carefully evaluate their specific needs and goals before deciding whether to adopt a multi-cloud approach and should ensure that they have the necessary expertise and resources to manage multiple cloud environments effectively.

Chapter 2: Cloud Deployment Costs – The cost of cloud deployment can vary depending on several factors, such as the size of the infrastructure, the specific cloud service provider, the duration of usage, and the type of workload being deployed. The chapter analyzes are some of the main cost components that organizations should consider when deploying in the cloud. It is important to note that cloud deployment costs can be highly variable and are heavily dependent on the specific use case and requirements of each organization. To optimize cloud deployment

costs, organizations should consider carefully evaluating their infrastructure needs, choosing the right cloud service provider, optimizing resource utilization, and implementing cost management strategies such as monitoring, automation, and right-sizing.

Chapter 3: Security Sense of Cloud – The security of cloud computing is a topic of great concern for many organizations, particularly as more critical workloads and sensitive data are being migrated to the cloud. The chapter provides architectural views on some of the main security considerations for cloud computing. Overall, cloud computing can provide a high level of security when properly configured and managed. Organizations should work closely with their cloud providers to ensure that their applications and data are properly secured and that all necessary security controls are in place. Additionally, organizations should regularly review their cloud security posture and adapt to any changes in their risk profile.

Chapter 4: Availability and Disaster Recovery – Availability and disaster recovery are two related but distinct concepts in the field of IT infrastructure and operations. In this chapter we try to uncover these concepts in a practical approach. Availability is focused on preventing and minimizing downtime and disruptions, while disaster recovery is focused on recovering from major disruptions and restoring services as quickly as possible. Both are important considerations for ensuring the continuity and reliability of IT services, and organizations should have strategies in place for both availability and disaster recovery to ensure that they can continue to operate in the face of unexpected events.

Chapter 5: Cloud, Agile and Software Development Life Cycle – Cloud computing can be a valuable tool for Agile and Software Development Life Cycle (SDLC) processes, as it offers a range of benefits and capabilities that can improve development efficiency and flexibility. In the chapter we take through how cloud computing can be used in Agile and SDLC processes to make them even more efficient. Cloud computing can be a powerful enabler for Agile and SDLC processes, providing the flexibility, scalability, and automation capabilities required to support rapid and iterative development cycles. Organizations that adopt cloud computing as part of their development processes can gain a competitive advantage by accelerating the delivery of new features and functionality to their customers.

Chapter 6: Retrofitting Cloud Service Accurately – Retrofitting cloud services accurately involves a number of considerations and steps to ensure a successful

migration of existing applications or infrastructure to a cloud environment. In the chapter we talk about are some key considerations when architecting cloud services. Retrofitting cloud services accurately requires careful planning, thorough assessment, and meticulous attention to detail. By following proper steps, organizations can ensure a successful migration to the cloud, while minimizing disruption and maximizing the benefits of cloud computing.

Chapter 7: Design First then Code – Design first then code, is a software development approach that emphasizes the importance of planning and designing a project before starting the coding phase. It involves creating a detailed plan or blueprint of the project's architecture, functionality, and user interface, which can help to ensure that the final product is well-organized, efficient, and user-friendly. Once the design phase is complete, the coding phase can begin. By taking the time to plan and design the project before starting to code, developers can avoid common pitfalls such as scope creep, inefficient code, and user interface issues. This approach can also help to save time and resources in the long run, as it can help to identify potential problems early in the development process.

Chapter 8: Infra Team and App Team Becomes DevOps Team–The merging of an infrastructure team and an application team into a DevOps team can bring numerous benefits to an organization's software development process. We have taken and experienced this journey and through the chapter share the key advantages of creating a DevOps team. Creating a DevOps team by merging infrastructure and application teams can help to streamline the software development process, improve collaboration, and increase efficiency and flexibility, ultimately leading to better software products and a more competitive business

Chapter 9: Traits of Being a Good Cloud Solution Architect – A cloud solution architect is responsible for designing and implementing cloud-based solutions for businesses and organizations. Based on our experience with the best in the industry, we outline the traits that are important for a cloud solution architect to possess. A cloud solution architect can help organizations to achieve their goals by designing and implementing effective cloud-based solutions and work as technology leader for the cloud journey.

Chapter 10: Treat Data and Database Separately – It is important to treat data and database separately in software development. While the terms are often used interchangeably, they represent distinct concepts that require different approaches and considerations. We discuss the details along with advantages of this approach

in this chapter. Treating data and database separately can help to improve the quality and performance of software applications, while also making it easier to manage and maintain the codebase.

Chapter 11: Frozen Architecture is Obsolete Architecture – The concept of frozen architecture refers to an approach in software development where the architecture is designed to be static and unchanging over time. This approach is often contrasted with agile development methodologies, which prioritize flexibility and adaptability. While frozen architecture may have been a viable approach in the past, it is becoming increasingly obsolete in today's rapidly changing technology landscape. We detail out some reasons in the chapter. The concept of frozen architecture is becoming increasingly obsolete in today's fast-paced technology landscape. Instead, software development teams should focus on agile development methodologies that prioritize flexibility, adaptability, and responsiveness to changing business needs and emerging technologies.

Chapter 12: What Exactly is Software Architecture? – Software architecture refers to the high-level design and organization of software systems, which includes the structures of software components, their relationships, and the principles and guidelines governing their design and evolution over time. It is concerned with making strategic decisions about software systems that enable them to meet the functional and non-functional requirements of the system, such as scalability, maintainability, reliability, and security, among others. In this chapter we take you through the forgotten concept and emphasize why it is still very important to have Software architecture in today's agile world.

Coloured Images

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/i9z0uhx

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Ambivalence of Multi-Cloud

Introduction

Let us start with a short introduction of Cloud computing. Cloud computing can be defined as the availability of compute, storage, and networking on-demand. Users do not get involved with direct management of the data center. The data center is shared among many users over the internet. Large cloud platforms have made their presence over multiple locations and countries. Cloud computing has helped consumers with low **Total Cost of Ownership** (**TCO**) due to economies of scale.

Structure

In this chapter, we will cover the following topics:

- Multi-Cloud computing the ambivalence
- Multi-Cloud computing reference architecture
- Multi-Cloud governance and management
- Cloud usability and Multi-Cloud computing

- Domain-driven design and Multi-Cloud computing
- Hub and spoke architecture – Multi-Cloud computing

Objectives

Multi-cloud is a puzzle the companies are trying to deploy and solve. In this chapter, we'll take you through the aspects to be kept in mind when architecting a multi cloud environment and deploying it. We will also be talking about the things to be kept in mind when it comes to security and governance when deploying a multi cloud environment for an organization.

Multi-cloud computing

IT infrastructure costs are normally seen as cost centers for companies, and public cloud computing allows them to minimize upfront capital expenditure and helps improve their cash flows. Due to its capability to support Infrastructure-as-Code and its partially readymade environment, Public Cloud allows enterprises to get their applications up and running faster. Infrastructure has better manageability and needs less maintenance. The resources are more agile, and changes can be made quickly. Public Cloud also enables capability to handle fluctuations in workloads.

The recent uptake of cloud technologies like Microsoft Azure, Microsoft Office 365, Amazon Web Services, Google GSuite and Google Cloud platform shows the next trend of the industry.

Information, data, and information technology are meant to get complex as they grow and progress by the day, and the last 40-50 years are testaments of this. Technologies like deployment platforms, programing languages, databases, security, infrastructure operations etc. have become more complex over the years. At a given time, companies use the best of available technologies to solve their problems and thus, keep building multiple layers. A new deployment betters and covers the gaps of previous deployments, and this goes on. Due to its agile nature, cloud has made these changes even faster. Apart from deployment, companies are now able to select technologies available in the marketplace and deploy, un-deploy or upgrade at a super faster pace.

Another emerging trend is the use of multiple cloud platform and multiple cloud products. For example, companies are using Microsoft Office 365 for productivity, Salesforce for CRM, Adobe Cloud for Creativity, etc. They are migrating applications and data to Azure and Amazon Web Services. They are migrating machine learning-based systems to Google Cloud Platform and Azure. They are adopting IoT systems, edge computing and building Hybrid environment for their business

edge computing and building Hybrid environment for their business.

This has made companies multi-cloud computing environments. Seldom do we see an organization not using multi-cloud technology today.

The Ambivalence

While deploying multi-cloud environments and multi-cloud products **Chief Information Officers** (**CIOs**), **Chief Technology Officers** (**CTOs**) and **Chief Information Security Officers** (**CISOs**) face challenges and have to make tough decisions about how/what/when to use which cloud product. They face equal challenges from users and the support staff on running and maintaining the multi-cloud environment. Multiple factors govern and determine whether the decisions and the deployments serve their purpose. We have seen multiple failures and success stories of multi-cloud deployments. In this chapter, we are going to take you through some of the important factors/decisions/actions that are recommended when deploying or implementing multi-cloud environment.

Multi-Cloud computing reference architecture

Cloud selection must be a methodical exercise with a proper comprehensive enterprise-wide architecture. Random and non-roadmap-based cloud deployments will not help achieve anticipated business value. They will increase **Total Cost of Ownership** (**TCO**), including management and operational costs, and they will also make security operations a major headache.

We would, thus, recommend starting with an enterprise-wide multi-cloud architecture. To have an effective enterprise-wide architecture for cloud deployment, we would start with reusing learnings from global standards and extend it for multi-cloud environment.

Cloud Computing Reference Architecture from **National Institute of Standards and Technology** (**NIST**) is widely referred by CIOs of the organization, and it identifies the major actors, their activities, and functions in cloud computing. The following diagram depicts a generic high-level architecture and is intended to facilitate the understanding of the requirements, uses, characteristics and standards of cloud computing.

Figure 1.1: *Cloud Computing Reference Architecture*

Reference: *NIST Cloud Computing Reference Architecture, Special Publication 500-292*

The image details out:

Cloud Consumer: A person or organization that maintains a business relationship with and uses service from cloud providers.

Cloud Provider: A person, organization, or entity responsible for making a service available to interested parties.

Cloud Auditor: A party that can conduct independent assessment of cloud services, information system operations, performance, and security of the cloud implementation.

Cloud Broker: An entity that manages the use, performance, and delivery of cloud services, and negotiates relationships between cloud providers and cloud consumers.

Cloud Carrier: An intermediary that provides connectivity and transport of cloud services from cloud providers to cloud consumers.

Let us extend this reference architecture for multi-cloud deployment:

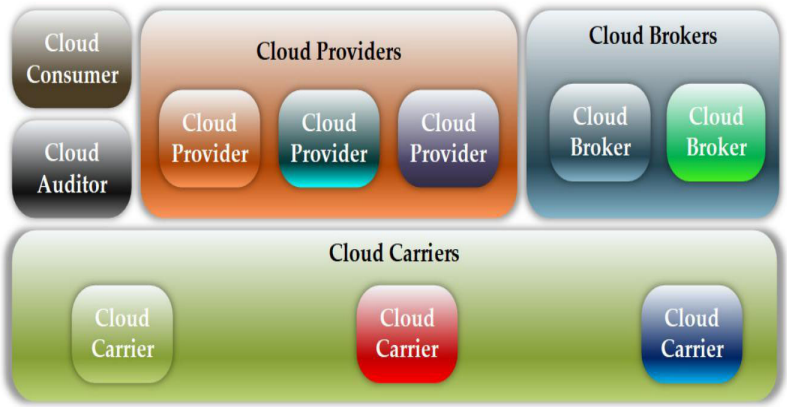


Figure 1.2: *Multi-Cloud Computing Reference Architecture*

Cloud Provider: Multiple cloud providers will be providing services to the organization. The cloud providers will be providing applications, services or platform based on their capabilities and the selection by the technology and the functional team.

Cloud Broker: Multiple cloud brokers provide services and cloud servicing capabilities to the organizations. The brokers will be selected based on their specialization and area of expertise.

Cloud Carrier: One or many cloud carriers would be providing connectivity to the cloud data centers from the user locations.

The cloud brokers and the cloud carriers can be common across multiple cloud providers.

Simplifying Multi-Cloud computing architecture

It is obvious from the above-mentioned multi-cloud architecture that the more cloud providers we have, the more complex it becomes. How do you simplify this architecture to make it more efficient and usable?

First thing that I would want you to consider is rationalizing the cloud providers based on the capabilities and reusing them for multiple use cases. Here's an example of reusing the same cloud provider to provide multiple services:

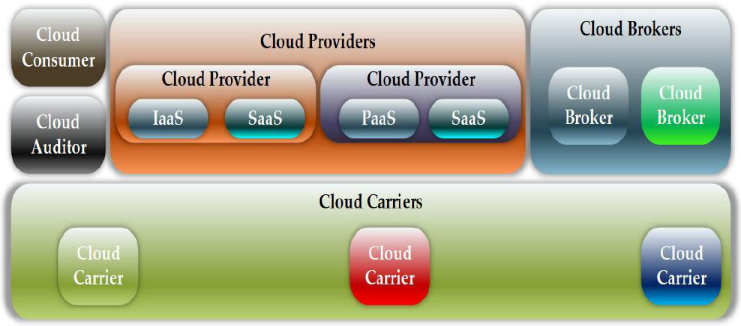


***Figure 1.3**: Simplifying Multi-Cloud Computing Architecture Option 1*

For example, if you select Office 365 from Microsoft, you can very well use Azure for IaaS and PaaS workloads. You can also reuse Dynamics 365 for CRM. If you are using Google Cloud Services for IaaS and PaaS, you can reuse the GSuite. If you are Amazon Web Services for IaaS and PaaS, you can reuse Alexa for business.

Another option that you could consider is using a cloud broker with capabilities stretching across multiple cloud providers and multiple cloud services. Cloud brokers will help stitch together multiple services from multiple cloud providers, abstracting the cloud environment to the consumers.

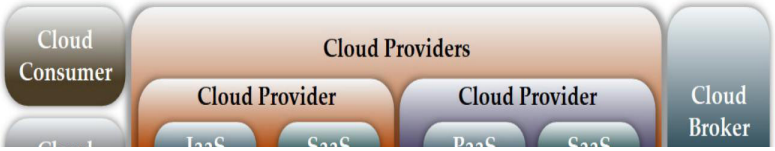


Figure 1.4: Simplifying Multi-Cloud Computing Architecture Option 2

We should also think on the lines of reducing the cloud carriers and reuse a carrier which can connect with multiple clouds and services.

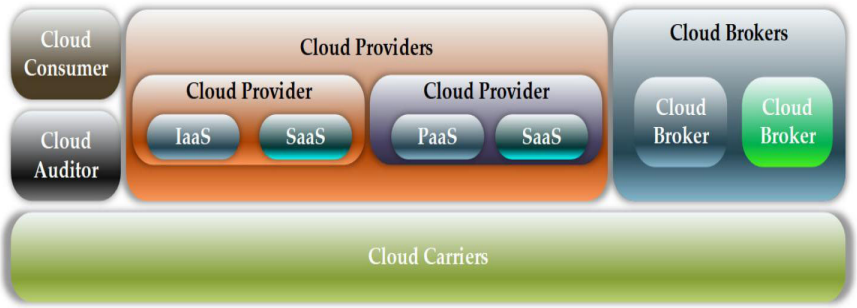


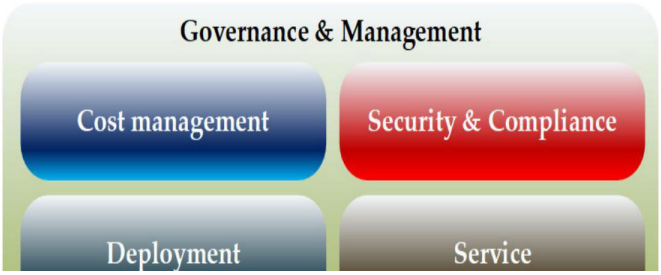
Figure 1.5: Simplifying Multi-Cloud Computing Architecture Option 3

The success of multi-cloud deployment in the organization will completely depend on the simplicity and practicality of the reference architecture. This will be the first step to removing the ambivalence around adopting multi-cloud architecture in an organization.

Multi-Cloud governance and management

Organizations and architects are deploying and adopting multiple cloud products and building multiple cloud layers in the organization. This brings another aspect to the fore: management and governance of multi-cloud.

Cloud management and governance comprises of the following areas:



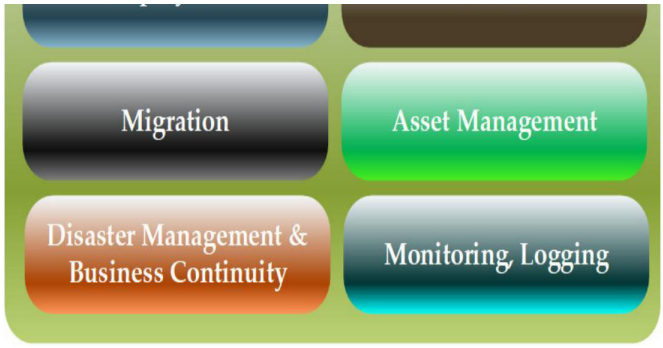


Figure 1.6: Governance and Management in Multi-Cloud Computing Environment

Cost management

Governing, monitoring, and managing costs are major concerns when adopting cloud technologies. Cost management becomes very relevant when cloud migration is part of major business transformations. Balancing performance demands, modifying adoption pace, modernization, process changes, and value realization of investment are some of the traits of cost management.

Cost management tends to become more complicated in a multi-cloud environment because each cloud technology has its own pricing model. An optimization practice in one cloud technology may not be valid for the second cloud technology. Cost in one cloud technology may be unoptimized due to a factor that may not be true in the second cloud. Security in one cloud platform may be part of the offering, and the other cloud would need to be subscribed. With multiple cloud and multiple dimensions, even a cost management tool may find difficult to match up.

When deploying a multi-cloud environment, consider cost management as one of your basic tenets, boundaries and policies that must be adhered to. Cost management cannot be a post-facto decision or a post-facto activity. The process and technology must be built in to your cloud architecture, and you must outline very clearly expectations. A failure in cost management can have lasting impact and very well jeopardize cloud adoption in long run.

Some of the common tools that you can use for multi-Cloud cost management are Azure Cost Management (erstwhile Cloudyn), Apptio, CloudCheckr, and Flexera.

Deployment

Deployment management establishes policies and procedures to govern asset configuration or deployment, which includes deployment, configuration alignment, orchestration, automation, etc. Deployment management also includes establishing DevOps operations in the organization. The deployment strategies and scripts/tools become reusable assets in the long run as the deployment process matures.

Organizations get into a trap of delaying the decision on deployment tools and technologies when building a multi-cloud strategy. The best way to define a multi-cloud strategy without proper deployment management guidelines is equal to having a King without clothes. The very success of multi-cloud strategy and multi-cloud use in an organization will completely depend on how deployments are done, how DevOps works, how SecOps works, and how Data and AIOps works. You will read more about DevOps in *Chapter 8, Infra Team and Apps Team Become DevOps Team*.

Multi-cloud deployment can be based on distributing business applications and domains across the best computing environment. While a sales department can

be on Dynamics 365, the marketing can be on AWS environment, the technology department can be on Azure.

Multi-cloud deployment can also be based on the need to increase redundancy, but this is a very complicated architecture, and it is recommended to stay away from this.

Migration

When adopting a cloud environment, organizations must normally migrate their existing estate to the cloud. Migration is a very complicated exercise, and if you don't have a sound and tested strategy in place, it could lead to a massive disaster. Migration management and governance needs to be done in accordance with the existing management practices while ensuring security and compliance. Failed migration can lead to serious repercussions, including organizations losing their reputation and money.

Consider a scenario when you must migrate from one cloud to another cloud or from on-premises to cloud; an immature migration process or an untested migration process will never enable you to adopt multi-cloud environment. Cloud native migration technologies do not really give you the freedom that you would need to adopt multi-cloud environment. Methodologies for migration must be designed in a wider way and must be compatible with multi-cloud environments.

Good tools that can be part of your migration strategy are Micro Focus PlateSpin Migration Factory, Carbonite Migrate, ScienceLogic, etc.

When selecting the tools, ensure that they support multiple scenarios like Generation 1 and Generation 2 Virtual Machines, **Master Boot Record** (**MBR**), **GUID Partition Table** (**GPT**), **Unified Extensible Firmware Interface** (**UEFI**), etc.

Disaster Recovery Management and Business Continuity Planning

Organizations have **Business Continuity Planning** (**BCP**) and **Disaster Recovery Management** (**DRM**) built into their people, processes, and technology, and cloud is no different. Companies will have to build DRM and BCP even while using cloud. It is a myth that cloud does not fail, or that it does not face disasters. Management and governance must be built to ensure that no or minimal loss when disaster strikes.

DRM is a combination of art and science, while BCP is science. While there are no prescriptive documented disaster guidelines from Standards bodies, you must decide on DRM and BCP based on business requirements.

BCP primarily focuses on preserving a department's ability to function in the event of a disruptive incident by ensuring that the most critical business functions continue to operate – even if at reduced capacity – while attending to the disruption. As such, BCP is particularly essential for departments that rely on rapid throughput or offer products and/or services at scale.

Meanwhile, DRM refers to the specific steps taken to recover to full IT functionality after a disruptive incident has occurred. While it shares much ground with BCP, DRM is a corrective measure that aims to achieve 'business as usual' as soon as possible across all affected IT and related systems.

As such, DRM should cater to a wide range of potential disruptive events and, for each, offer a path to full IT functionality, even if achieving that functionality will take some time. BCP should consider the same wide range of possible events, but it should focus on ensuring (or preserving) at least a minimum level of functionality across the services and systems the department needs to continue to deliver and operate.

When using cloud, we see a lot of confusion between **High Availability** (**HA**)and DRM. Both concepts are used in system architecture that mitigate against failure. HA eliminates single points of failure, and DRM is the process of getting a system back to an operational state when a system is rendered inoperative. More importantly, it is useful to note that DRM picks up when HA fails.

Multi-cloud environment can prove a boon or bane for DRM and BCP: boon because you have an environment to fall back to in case a cloud provider faces disasters, and bane because managing multiple cloud technologies and multiple deployment environments, each having different disaster recovery steps, can prove to be a disaster.

When designing a multi-cloud architecture, ensure that you have proper DRM and BCP in place. Your architecture should ensure that it shows distinctive attributes for

BCP in place. Your architecture should ensure that it shows distinctive attributes for DRM and BCP. We normally recommend organizations to follow or implement ISO 22301 for DRM and BCP.

ISO 22301

The **International Organization for Standardization** (**ISO**) is an independent non-governmental organization and the world's largest developer of voluntary international standards. The ISO formed the TC 223 Societal Security technical committee to develop standards for protecting society, including organizations, in the event of catastrophe, such as a natural disaster, major terrorist attack, or shutdown of power grids.

Published in 2012 by the technical committee, ISO 22301:2012 is the first international standard for management systems that helps ensure business continuity. ISO 22301

is the premium standard for business continuity, and certification demonstrates conformance to rigorous practices to prevent, mitigate, respond to, and recover from disruptive incidents.

Security and compliance

Multi-cloud deployment will always increase the surface area for attacks. Cloud adoption can never be complete without considering security, regulatory requirements, data protection, and compliance. Identifying potential security threats to your cloud environment and establishing processes and procedures for addressing these threats is a priority for any IT security or cybersecurity team. Ensuring that the organization is always compliant to regulations is a task of all cloud management teams.

When deploying multi-cloud environment, the cybersecurity requirements should be adhered to, should be designed, and should be built in each cloud provider. Under no circumstances can you allow one cloud provider to not follow all the security and compliance requirements. Failure to implement compliance in all the cloud providers will make that cloud the weak link in your organization.

The cloud selection process itself should have qualification criteria with all the compliance requirements. Normally, it is seen that large cloud providers like Microsoft, Amazon, and Google already have industry standard compliance and security practices in place. If you are going for a cloud provider other than these large players, you should ensure that they are properly audited and have valid authentic certificates in place. If you're going for a cloud-based SaaS product, you should ensure that the product is also certified with all the compliance requirements of your organization.

Continuous auditing and compliance

It is mandatory that your audit process is remodeled to "Continuous Auditing and compliance". Continuous auditing and compliance checks ensure that all the cloud providers that you are using in multi-cloud environment are always compliant to your needs. Since cloud environments now encourage infrastructure-as-code, it is always a great practice to ensure that the continuous auditing and compliance process keeps monitoring and auditing the deployed infrastructure under all circumstances and in real time. You should also ensure that compliance and security standards for each cloud are evangelized with the team. Security operations and incident response should be uniform across all the cloud environments and should entail a proper **Root Cause Analysis** (**RCA**) involving both your organization and the cloud providers.

Service management

ITIL defines service management as the implementation and management of quality IT services that meet the needs of the business. IT service management is performed by IT service providers through an appropriate mix of people, process, and information technology. ITIL V3 organizes the ITIL processes around the five service lifecycle stages: *Service Strategy*, *Service Design*, *Service Transition*, *Service Operation*, and *Continual Service Improvement*. Service management is a critical part of cloud management as it decides whether the cloud adoption by an organization can be supported from within and whether the organization can sustain the transformation.

When deploying multi-cloud environment, the service management should be modified to handle multiple technology, multiple environments, multiple operational process, multiple issue resolvers, multiple SLA, multiple service SLA, multiple cloud marketplace vendors, multiple stakeholders, and multiple location-based support resources.

Asset management

Asset management focuses exclusively on the management of cloud environment, products, or services that the organization uses. It keeps track of cloud farm, its maintenance, compliance, upgrades, and secure disposal. Asset management is an important pillar of security and cost management.

Asset management has become a different ball game in multi-cloud environment. Cloud resources can be created at will and can be deleted/modified at will. If the asset management is not tuned to ensure that each infrastructure event is captured and actioned, it will just be a matter of days before your asset management tool goes

and actioned, it will just be a matter of days before your asset management tool goes out of sync with what is deployed on the ground. Once asset management is out of sync, the operation process, management procedures, monitoring and alerting, logging will all follow suite. Asset management needs to ensure that IaaS-, PaaS-, or SaaS-based resources are properly accounted for. Remember, asset management is the most critical piece of data when deploying a multi-cloud architecture.

Monitoring and logging

This focuses on monitoring applications, workload, and asset performance. It also entails common tasks like making changes to meet scale demands, remediate performance **service-level agreement** (**SLA**) violations, and build proactive automated remediation to avoid performance SLA violations. Organizations are now collecting more logs and analyzing them in multiple ways to make their decisions and actions more intelligent and using them in spreading Cloud computing footprint. Intelligent log analysis has become an input to infrastructure decisions.

While building monitoring and alerting practices, you will need to ensure that all cloud environments are in scope of the monitoring tool. While cloud providers have their native monitoring capability, some of these are not compatible with other cloud providers. So, the monitoring tool selection needs to ensure that it is compliant and can integrate with cloud environments of choice. Any blind spots in monitoring spectrum would mean inevitable incidents.

When deploying multi-cloud architecture, you will need to ensure that log collection and the correlation tool are integrated with all the Cloud environments. Cloud native logs play an important role in log monitoring. It is always good to check whether Cloud provider's native logs follow one of the standard log formats like Syslog, **JavaScript Object Notation** (**JSON**), Windows Event Log, **Common Event Format** (**CEF**), or **Graylog Extended Log Format** (**GELF**).The services deployed, IaaS, PaaS or SaaS, should also follow the common log formats. Any application that is built on the cloud environment should also follow these log formats.

This will ensure that your log analyzer is integrated with all workloads in the organization and can analyze/correlate logs across multiple clouds and multiple workloads. The more the data, the more effective the monitoring and analysis, and the more effective the decisions!!

Summarizing Multi-Cloud governance and management

Management and governance must be made an integral part of the multi-cloud

Management and governance must be made an integral part of the multi-cloud architecture, else organizations will hit a point where it will become very difficult to operate the systems and manage them. Cost savings and agility will become a mirage. Many enterprises are experiencing these daily. They find themselves defending and navigating through negative return on investments, and the decision of cloud adoption is being questioned. When the same occurs in multi-cloud environments, the problems and issues increase multiple times.

To prevent this, you will need to adhere to governance and management principles as a bible. Cloud and environments change, but principles and practices do not. The basics need to be right; the architecture will need to ensure that it has all the elements built-in and the problems are solved in their own domain. Also, ensure minimal data transfer between the clouds.

Try to stay away from the oversold concept of "vendor lock-in", and do not let that impact the decisions. If your architecture is sound, there can be no lock-ins.

Cloud usability and Multi-Cloud computing

Multi-cloud management is complex, but if the multi-cloud architecture divides these into business domains, abstracted into processes managed by optimal tools and solution, this will remove the complexity from your cloud-based systems. This brings to fore an important aspect of usability and whether the multi-cloud environment is helping the users in their business. The next focus area for us would be designing and determining the usability of the multi-cloud architecture. We would ideally like starting with the NIST cloud usability framework.

NIST, through its special publication 500-316, proposed a framework for cloud usability. ISO defines usability as the extent to which a system, product, or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use, as demonstrated in figure 1.7.

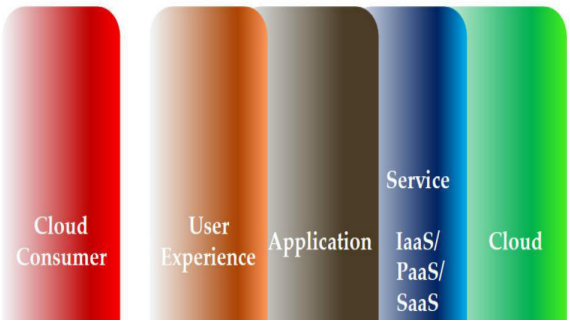


Figure 1.7: Cloud Usability

Based on this definition, NIST came up with a user-centric approach to build usability framework. The center of the world, as demonstrated in Figure 1.8, is user needs and their goals. Taking a bottom-up approach for architecture often leaves out the users when considering the cloud.

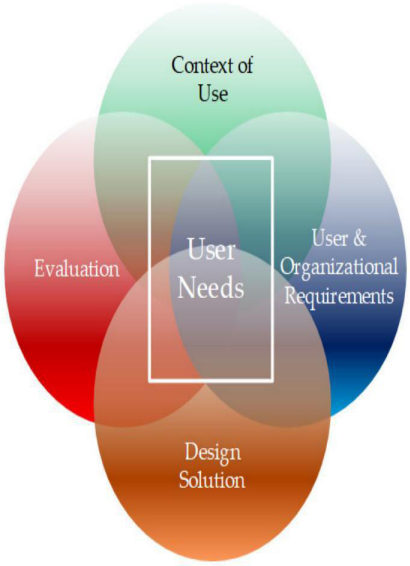


Figure 1.8: User Centric Cloud Usability

Reference: NIST Framework for Cloud Usability, Special Publication 500-316

User needs are central to developing a usable system, product, or service. It is these user needs that form the basis of the NIST framework:

***Figure 1.9**: User Needs from Cloud Environment*

***Reference**: NIST Framework for Cloud Usability, Special Publication 500-316*

Capable

Multi-cloud consumers will expect their cloud services to be *Current* (based on the latest technology), be *Device Independent* (accessible using any device form factor), be *Platform Independent* (should not be tightly coupled with cloud platform) and have *Cloud Functionality* (like elasticity, scalability; rapid provisioning, etc.).

Each cloud service, product, platform, application, and its integration capabilities should be evaluated, and only the cloud providers who fit the bill should be selected. If the multi-cloud providers do not provide the capability as per the framework, it will become a usability issue, and that might end up making the environments not user-friendly. Since the whole premise of building a multi-cloud environment is to make applications and systems more user-friendly and agile, capability evaluations are important.

Personal

Multi-cloud services should allow consumers *Accessibility* (accessible to consumers with a variety of needs), *Customization* (allow consumers to change their user interface to suit their own needs, *Data Ownership* (consumers should have complete ownership over the data they store in the cloud), *Identity Access Management* (integrated authentication/authorization) and *Control* (should have a sense of

control over the functionality of the cloud service).

Personalization is an important aspect of usability, be it an application or a cloud product. An architect or technical team can deploy a state-of-the-art cloud or cloud application, but if it is not customizable, accessible, or user-friendly, it will never get used. Users will be resistant to adopting the new application/environment, and this will become a major challenge in a multi-cloud environment.

Reliable

The ability of a system or component to perform its required functions under stated conditions for a specified period should be *Available* (high availability and SLA-based uptime), *Responsive* (high degree of performance), *Consistent* (exhibit the same functionality under every situation), *Transparency* (technology should be transparent) and *Error-Resistant* (should resist the adverse effects of errors).

Determine the uptime SLA requirements from the business users and ensure that the cloud reliability conforms to the requirements.

Secure

Multi-cloud consumers want their cloud services and data to have *Security* (resistant to attacks from unauthorized users, other cloud services, malicious software, and

attacks on cloud hardware and the internet), *Privacy* (data leakage prevention), *Identity Authorization Management* (not allow unauthorized access user data or process execution) and *Trust* (user should trust the cloud).

We see security as a basic principle of architecture, even for multi-cloud deployment. Building security into multi-cloud architecture is necessity to ensure usability.

Valuable

Cloud consumers will expect cloud services to provide value to them and their organization in the form of *Savings* (save on costs), *User satisfaction* (should be high for the user to continue consuming the service) and *Utility* (provide new features that are not possible with any other IT setup).

Brain supplements

The following are a couple of ideas that we are seeding and would really like to hear feedback on. We are not going to delve deep into these at this moment in time.

Domain-driven design and Multi-Cloud computing

Eric Evans, in his book *Domain-Driven Design: Tackling Complexity in the Heart of Software*, introduced Domain-Driven Design for building applications. *Domain-driven design* is the expansion upon and application of the *domain* for software development. Evans' *Domain-Driven Design* further defines *Context* (the setting in which a word or statement appears that determines its meaning), *Model* (a system of abstractions that describes selected aspects of a domain and can be used to solve problems), *Ubiquitous Language* (a language structured around the domain model and used by all team members to connect all the activities of the team with the software) and *Bounded Context* (a description of a boundary (typically a subsystem or the work of a specific team) within which a particular model is defined and applicable).

On first sight, domain-driven design looks out of place in the context of multi-cloud computing, as domain-driven design has been widely used in application development and microservices development. But our view is that we can use this principle in multi-cloud architecture as well. Multi-cloud architecture will be driven by business domains; they will need input from domain experts, will need domain models, and have context. Below is an example of domain driven design when using multiple cloud products:

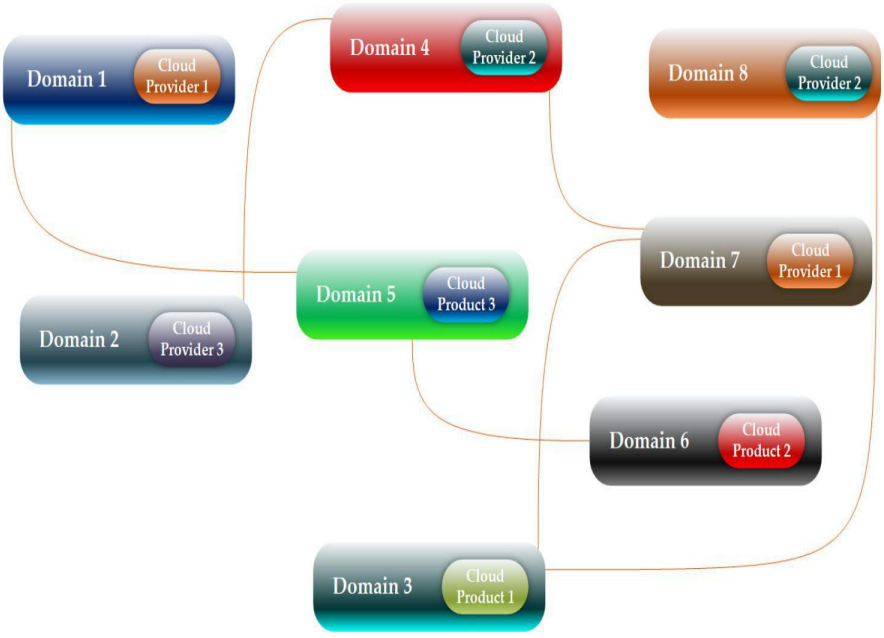


***Figure 1.10**: Domain Driven Design for Multi-Cloud*

Based on the domain of the business and the specific cloud requirements, the enterprise-wide cloud architecture can deploy multiple cloud services and products. The cloud domains remain atomic, data remains limited to the domains and integration platform is built for inter-domain data flow.

Hub and spoke architecture – multi-Cloud computing

The second idea that we would like to ponder is deploying multi-cloud architecture as hub and spoke.

A hub-spoke model is where a single hub system serves all spokes connected to it. The data flowing from one spoke to another always traverses via the hub. The hub transforms and abstracts the data and identity of one spoke to another. The hub also

provides a **common interface model** (**CIM**) for the data, process, procedure, and technology.

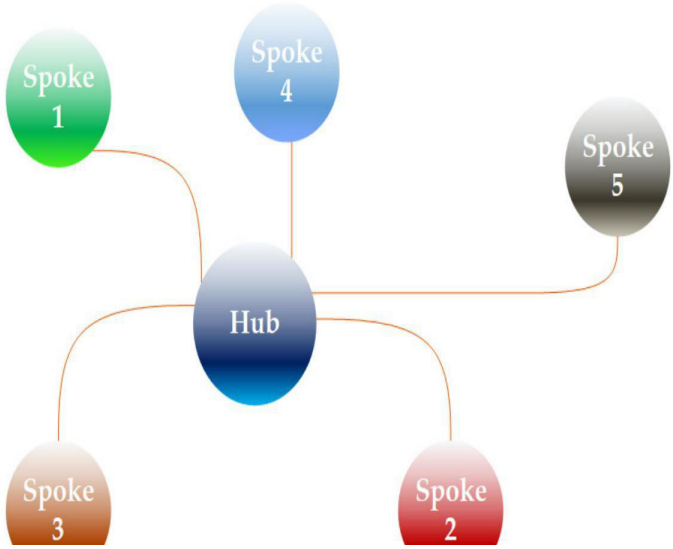


Figure 1.11. Hub and Spoke Design for Multi-Cloud – Option 1

Applying CIM to multi-cloud architecture, we can deploy a central security cloud for managing security, compliance, and policies. All the cloud environments (spokes) will use the central security cloud (hub) to connect with customer office and deploy security controls.

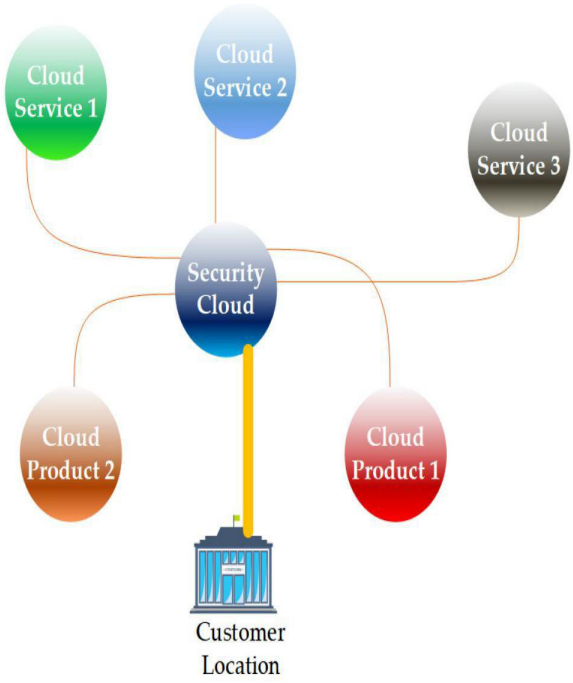
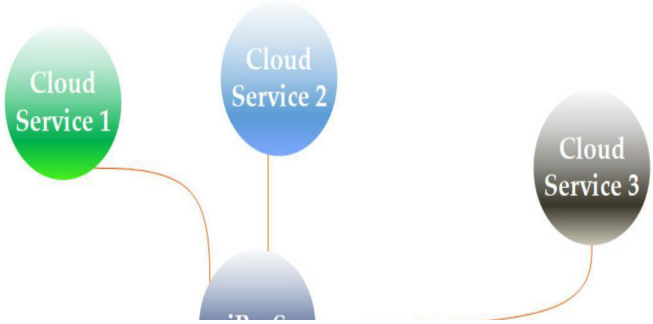


Figure 1.12: Hub and Spoke Design for Multi-Cloud – Option 2

There is a lot of data flow among the cloud, so does this defeat the purpose of multi-cloud deployments? On the one hand, it simplifies the CIO and CISO lives, but on the other, it can become a single point of failure. But this can be a useful model for some use cases.

The second model would be deploying a messaging platform/integration platform/enterprise service bus as hub cloud and integrating all spokes. Popularly, these services are called iPaaS.



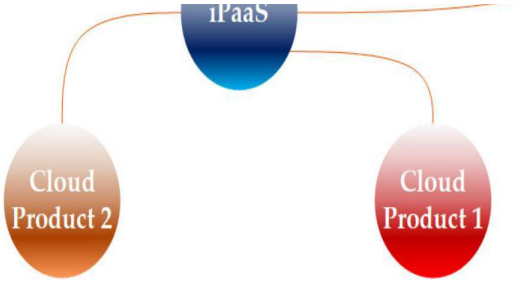


***Figure 1.13**: Hub and Spoke Design for Multi-Cloud – Option 3*

This looks to be a good model and defines data boundaries.

Alternatively, can we have a combination of domain-driven multi-cloud architecture and hub-spoke-driven multi-cloud architecture?

Conclusion

In this chapter, we discussed the elements of multi-cloud adoption and organizational changes to people, processes and technology that should be done for it. Multi-cloud is an aspiration that cannot become reality if not planned and implemented with end-to-end vision with optimal management and governance.

CHAPTER 2

Cloud Deployment Costs

Costs

Introduction

Gartner's had forecasted Worldwide Cloud revenue to grow to $600 billion by 2023. This is more than the GDP of multiple countries. They also forecasted that almost all enterprises will adopt cloud in the coming years, and that nearly all legacy applications would have migrated to public cloud infrastructure by 2024.

Total cost of ownership and return on investment analysis will lead to optimization in cloud spend and would drive more cost-effective cloud adoption. Global cloud providers have made cost optimization one of the KPIs for their sales teams and are also strengthening their native optimization capabilities. This is helping organizations select the most cost-effective architecture that can deliver the required performance.

Structure

In this chapter, we will do the following:

- Introduce Total Cost of Ownership (TCO) for Data Centers
- Discuss how to model TCO for Cloud and discuss qualitative and quantitative aspects of TCO

- Discuss Outage impacts on TCO
- Look at mechanisms to optimize cost

Objectives

In this chapter, we will discuss different cost factors for cloud adoption and look at how to best use the tools to reduce costs during the cloud adoption journey. There are several differences between traditional data center TCO and Cloud, and we will look at them in detail in this chapter.

Total Cost of Ownership

Total Cost of Ownership (**TCO**) can be defined as the total cost of acquiring/purchasing, using, managing/operating, and withdrawing/decommissioning an asset over its life cycle. In effect, it also encompasses direct costs and indirect costs.

TCO is influenced by three factors:

- **Capital Expenditure**
  - **Supporting devices cost**: Costs related to HVAC, networking, power, power backup, and security devices.
  - **Hardware cost**
  - **Design & engineering cost**: Cost paid to the architect/engineer to build the data center.
  - **Installation cost**: Cost incurred during installation, including but not limited to mason work, racking and stacking work, and cabling work.
  - **Software license cost**: All the software deployed in the data center, including but not limited to operating systems, business software, application software, security software, operational software, and management software.
  - **Subsidies & grants**: Government, at different times and using different schemes, gives subsidies and grants to companies to set up their office. This is one of the investments that helps in building the data center.

- **Operational expenditure**
  - **Maintenance cost**: Costs like dispose-off cost, annual maintenance fees, cleaning and disinfecting cost, air purification costs, pest control costs.
  - **Virtual infrastructure cost**: When using a cloud environment, the servers are deployed in form of Virtual Machines (Infrastructure-as-a-Service) and services (Platform-as-a-Service). These are normally charged operationally, i.e., you pay as and when and for how much you use.
  - **Repairs cost**: Any cost incurred for fixing physical damage or damage to software, building repairs, extensions, etc.
  - **Software license cost**: Software subscription cost, software support

- **Software license cost**: Software subscription cost, software assurance costs, etc.
- **Human resource cost**
- **Power consumption cost**
- **Depreciation cost**
- **SLA & compliances cost**: Investment done to ensure SLA monitoring, SLA adherence, compliance monitoring, compliance auditing and certification, etc.
- **Building security cost**: Physical security, building resiliency cost, access control practices, etc.
- **Bandwidth/Egress/Ingress Cost**: If we go for an on premises data center, bandwidth needs to be purchased from different ISP's; if it is a cloud data center, then it is the cost related with egress (outgoing data cost from cloud), and ingress (incoming data cost to the cloud).
- **Integration cost**: As the business grows and companies integrate/merge/are acquired, data centers have to keep changing and integrating.
- **Disaster recovery & Business continuity cost**: Based on the business of the company, they must invest in disaster recovery and business continuity practices.

- **Qualitative costs**

  Qualitative costs cannot easily be translated into financial numbers but represent significant advantages or disadvantages of a cloud-based or on-premises solution that should be considered when comparing alternatives. Since these factors do not have monetary values, a relative cost calculation must be used. For example, identify the relative importance (high, medium, low), followed by the "cost" of each solution relative to the others (-1, 0, +1).

  - **Regulatory & policy cost**: There might be scenarios where companies

need to follow certain practices, and failure to follow these practices will invite penalties, fine, and reputation loss.

- **Loss due to SLA breaches**: Every organization is bound by SLA that they guarantee to their end customers. Losses due to SLA breaches can be unthought of.
- **Loss due to security breaches**: Material damages due to security breaches can be accounted for financially. What can't be accounted for is the loss of reputation, and the loss of customers' confidence and trust, the loss of potential market, etc.
- **Performance (scalability/elasticity)**: There are scenarios where organizations need to be scalable or elastic in providing their infrastructure, and a majority of these lead to potential business. If organizations are not ready to provide scalability and elasticity, there might be potential losses to business, trust, and reputation.
- **Loss due to outage**: **Mean-Time-To-Failure** (**MTTF**) and **Mean-Time-To-Recovery** (**MTTR**) are critical metrics for an organization to track and ensure that the losses due to outages are minimal. Outages are natural behaviors of data centers, so the architecture and design must take care of them. To design for outages and failures, you must spend extra, but if you don't cater to failures and outages in design, you lose even more.
- **Agility**: Agile is the preferred method across technology and businesses. If your data center doesn't have agility, you lose business and customers.

There can be additional factors based on business or a particular organizational local practice that we have not taken into consideration here. In summary, TCO can be represented as:

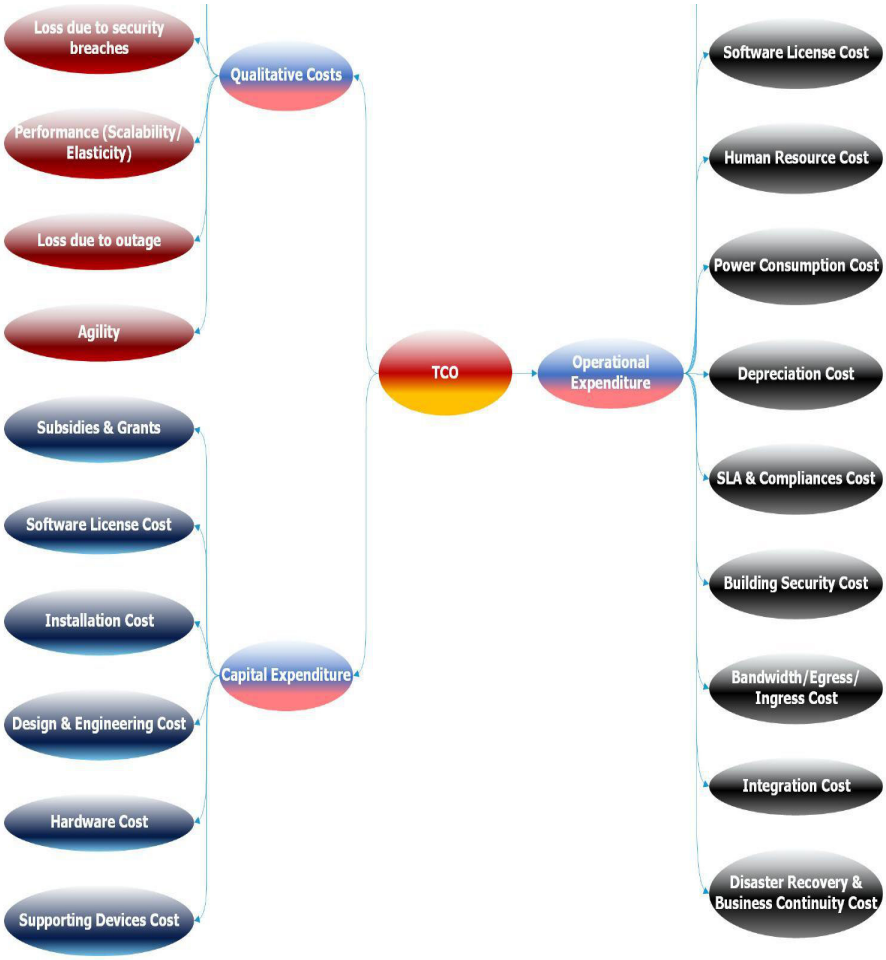


Figure 2.1: Total Cost of Ownership

Modeling Total Cost of Ownership (TCO) for Cloud

When deploying a cloud environment, upgrading infrastructure, or upgrading data center CIOs and CTOs generally get into total cost of ownership discussions and

evaluate the **Return on Investment** (**ROI**). Since cloud is a lever to reduce the cost of IT investment, it is a general practice to go through TCO and analyze the benefits before moving on to adopting cloud.

Generally, it has been seen that when the CIOs and CTOs get into this exercise, they take a traditional approach that's very similar to the one taken when deploying infrastructure on premises.

Our experience, more importantly our cloud deployment failures, have taught us that cloud TCO is not the same as the classical or traditional on premises TCO.

In this chapter, we will share a few thoughts on TCO for cloud deployment and try to differentiate between that and on premises TCO.

Cloud TCO

There are multiple types of cloud deployment that companies prefer nowadays:

- On-Premises Cloud: Deployed on Hyper Converged Infrastructure or Private Cloud OS
- Public Cloud: Deployed on Public Cloud provided by organizations like Amazon, Microsoft, Google etc.
- Hybrid Cloud Deployment: Hybrid Cloud deployment where part of the infrastructure is deployed on-premises and part on the Public Cloud; the Public cloud is also used as scalability/DR data center for the on premises
- Multi Cloud Deployment: Deployment on multiple public cloud

For each of these deployment models, TCO will have different cost components:

Capital Expenditure	On-Premises Cloud	Hybrid Cloud Deployment	Public Cloud	Multi Cloud Deployment
Supporting Devices Cost	✓	✓	●	●
Hardware/Virtual Machines Cost	✓	●	●	●
Design & Engineering Cost	✓	✓	●	✓
Installation Cost	✓	●	×	×
Software License Cost	✓	●	●	●
Subsidies & Grants	✓	✓	×	×

Table 2.1: Total cost of ownership for Cloud

✓ → Cost will incur.

× → Cost will not be incurred.

● → Partial or lower costs will incur.

Capital expenditure

There are multiple capital expenditures that will be incurred when building data center:

- **Supporting devices cost**: Supporting devices like network devices, storage devices, etc. will have to be purchased when deploying on premises cloud or hybrid cloud. When deploying public cloud or multi public cloud, these investments will be limited to bare minimum network devices.
- **Hardware/Virtual machines cost**: When deploying on premises cloud, you have capital expenditure, i.e., investments in purchasing hardware to deploy the cloud. When deploying hybrid cloud, the investment is lesser than for on premises cloud. When deploying public cloud and multiple public cloud, the investment is purely in virtual machines that are purchased and paid for upfront; Amazon and Microsoft call them "*Reserved Instances*".
- **Design & engineering cost**: On premises cloud and hybrid cloud will need a physical data center, so investment is needed for designing and engineering the on-premises data center. Public cloud and multi public cloud will only need initial architecture to be done for building the landing zones for the workload. These are minimal investments for a single cloud, but the investment is substantial if it must be done for multi-public cloud deployment.
- **Installation cost**: Installation would be done in on-premises data center for deploying cloud and hybrid cloud environment. There would not be any installation cost involved when deploying public cloud or multiple public clouds.
- **Software license cost**: License cost is a capital expenditure incurred when deploying the software on premises. When deploying in hybrid cloud environment or public cloud or multi public cloud, license cost gets converted into operational expenditure. There are few scenarios where, in hybrid cloud or public cloud, you want to purchase licenses upfront; in such cases, it becomes a capital expenditure. But license costs are less in hybrid cloud or public cloud scenarios.
- **Subsidies & grants**: These grants only exist when deploying physical data centers. Sometimes, global cloud providers also tend to provision one-time marketing credits, which are less in value.

Operational Expenditure	On-Premises Cloud	Hybrid Cloud Deployment	Public Cloud	Multi Cloud Deployment
Maintenance Cost	✓	✓	×	×
Virtual Infrastructure Cost	×	●	✓	✓
Repairs Cost	✓	✓	×	×
Software License Cost	●	✓	✓	✓
Human Resource Cost	✓	●	●	✓
Power Consumption Cost	✓	✓	×	×
Depreciation Cost	✓	✓	×	×
SLA & Compliances Cost	✓	✓	●	✓
Building Security Cost	✓	✓	×	×
Bandwidth/Egress/Ingress Cost	✓	✓	●	●
Integration Cost	✓	✓	✓	✓
Disaster Recovery & Business Continuity Cost	✓	●	●	✓

Table 2.2: Capital expenditure in data center

✓ → Cost will incur.

× → Cost will not be incurred.

● → Partial or lower costs will incur.

Operational expenditure

Multiple operational expenditures will be incurred when building data center:

- **Virtual infrastructure cost**: When on premises, you do not pay for the virtual infrastructure, but you pay for it when using public cloud or hybrid cloud. Hybrid cloud will have lower virtual infrastructure cost because the hardware is owned by you.
- **Software license cost**: License cost is a capital expenditure when deploying the software on premises. When deploying in hybrid cloud environment or public cloud or multi public cloud, the license cost is converted into operational expenditure. For on-premises costs, this amount is only as much as you pay to keep support or software assurances going on.
- **Human resource cost**: Due to Infrastructure-as-Code capability, human resource investment will be less than on-premises and hybrid cloud in public cloud

cloud.

- **SLA & compliances cost**: When running on premises cloud and hybrid cloud, you own the SLA and are liable to pay penalties in case of breaches. In the case of public cloud, public cloud providers today have a financially backed SLA, which will allow you to forward commit to your customers.
- **Bandwidth/Egress/Ingress cost**: Public cloud charges egress cost for your deployment, while the bandwidth and ingress are free. When deploying on premises or hybrid, bandwidth is provided by ISPs, and we pay for it.
- **Disaster recovery & Business continuity cost**: Public cloud is designed and invested in for handling disaster recovery and business continuity better. Published in 2012 by the technical committee, ISO 22301:2012 is the first international standard for management systems that helps ensure business continuity. ISO 22301 is the premium standard for business continuity, and certification demonstrates conformance to rigorous practices to prevent, mitigate, respond to, and recover from disruptive incidents. Public cloud with these certifications is more adept to provide DR and BCP capabilities. By nature, cloud DR is provided in shutdown state (unbilled state), and the cost will always be less than on-premises.

Qualitative Costs	On-Premises Cloud	Hybrid Cloud Deployment	Public Cloud	Multi Cloud Deployment
Regulatory & Policy Cost	✓	✓	●	●
Loss due to SLA breaches	✓	✓	●	●
Loss due to security breaches	✓	✓	●	●
Performance (Scalability/ Elasticity)	✓	●	×	×
Loss due to outage	✓	✓	●	●
Agility	✓	●	×	×

***Table 2.3**: Operational cost for Cloud*

✓ → Cost will incur.

× → Cost will not be incurred.

● → Partial or lower costs will incur.

Qualitative costs

Though qualitative cost cannot be measured, the following are the heads that we normally incur:

- **Regulatory & policy cost**: On premises and hybrid cloud deployment would

mean that you are responsible for deploying all the regulatory practices and

ensuring that all the audits are done anytime. In case regulatory requirements are not met, you may run into trouble. When using public cloud, the onus lies on the public cloud provider, while you own part of the requirements.

- **Loss due to SLA breaches**: While SLA breaches in public cloud are owned by public cloud providers, SLA breaches on premises would mean direct and indirect costs for you.
- **Loss due to security breaches**: Public clouds have huge security investments made with global experts working on it, ensuring that security is the best. As a result, security breaches are not as common in public clouds as they are on premises data centers. So, the costs associated with losses due to security breaches will be higher on premises and in hybrid cloud environment. The loss of reputation cost would be much higher for security breaches on-premises.
- **Performance (Scalability/Elasticity)**: Hybrid cloud and public cloud provide scalability and elasticity, which on-premises cloud cannot. This means the cost would be higher for the on-premises cloud.
- **Loss due to outage**: On-premises outages would cost more than public cloud outages, and an outage would also make the company lose reputation, trust, and confidence of the customer.
- **Agility**: One of my seniors used to say that there are three aspects that you should always consider in an IT plan: known-knows, known-unknowns and unknown-unknowns. While on-premises cloud has agility to handle known-knows, Public cloud provides the agility to handle known-unknowns and unknown-unknowns more effectively.

Outage analysis and cost of outage

As per the latest outage analysis report of Uptime institute, despite an increase in robustness of IT infrastructure, major failures are still common, and the consequences of such failures are undoubtedly higher than in the past. It was also seen that one failure can cascade between data centers and across networks, triggering secondary failures.

The growing move to cloud services and the extensive use of co-location makes the need for a good understanding of outages and their consequences more important at the executive level. While it is possible to outsource the work, it is not possible to outsource the responsibility. Similarly, the use of outsourcing for all maintenance and management can create problems

and management can create problems.

The number of publicly reported outage frequency as per uptime institute were 27 in 2016, 57 in 2017, 71 in 2018 and 163 in 2019. During this period, the severity of

outages increased, and the most severe outages increased from 11% of the total in 2016 to 18% of the total in 2019 (9% in 2017 and 4% in 2018).

For 2016–2018, the contribution of Network, power and IT Infrastructure failure in outage has been 90% and 76%, respectively. If we look deeper, you will see that while power failures have decreased from 2016 to 2019, network and IT infrastructure failures have increased. Companies have also reported an increase in "*unknown reason*" for outages in these years.

In the window of 2018 and 2019, the number of Public cloud provider outages have been 1/4th the number of on-premises outages.

60% of companies reported less than $100,000 loss due to outage, 28% reported losses between $100,000 and $ 1 million, and around 10% reported losses worth more than $ 1 million.

During this period, MTTR has also increased, and outages are now taking longer. Uptime also mentions in the report that the disruption and remediation run into billions of dollars globally.

A clear takeaway from this report is that resiliency requires ever greater investment, attention and discipline, and cloud is something that you cannot turn away from.

You can read more at **https://uptimeinstitute.com/annual-outage-analysis-2020**.

Cloud cost estimation and management

As cloud adoption increases, there are challenges in budgeting and management of cloud costs. CFO organizations who have traditionally been modeling their budgets based on capital expenditure for on-premises investment now are up against challenges to budget Operational Expense for cloud purchase and management. Considering that IT infrastructure is the cost center for many organizations, innovative solutions need to be applied to aid CFO's work on budgeting. Customers come to us asking questions about how to budget their annual cost and how to ensure that the cost never exceeds the allocated budget. Some organizations find challenges in adopting cloud if they are unable to model this cost. The following are some of our recommendations from what we have seen and practiced with our customers.

Using predictive analytics for Cloud cost estimation

One of the newer and more efficient ways to budget for cloud cost is to use predictive analytics for cloud cost, where you can employ various statistical techniques, from modeling to machine learning and data mining, to analyze current and historical

data and facts to make predictions about future costs. The data can be derived from open-source data sources and should be as neutral as possible. The vendor neutrality will remove the bias in the data. Cloud providers today have data scientists who can help you in making this prediction.

Managing and monitoring Cloud deployment cost for cloud

One of the advantages of using public cloud is cost management. All major cloud providers today provide in-built tools to manage, monitor and forecast cost for cloud. By using the cost management tools, companies can reduce costs and the overhead required to manage organizational assets. Cost management and billing tools provided by cloud providers help analyze, manage, and optimize the costs of workloads. Using the product helps ensure that an organization is taking advantage of the benefits provided by the cloud.

With cloud products and services, you only pay for what you use. As you create and use cloud resources, you get charged for them. Because of the deployment of ease for new resources, the costs of workloads significantly increase without proper analysis and monitoring. So, it is recommended to use cost management software to do the following:

- Perform administrative billing tasks, i.e., paying bills, managing subscription, and the like:
- Download itemized billing data.
- Proactively analyze costs.
- Set thresholds for spending.
- Identify workload optimization and spending.

Cloud management ensures that companies would never get into a situation where actual costs cross the budget.

Reference for Data Center TCO

For Data Center TCO, there are multiple models that are available and can be used. One of the effective models is FETCO, an estimation and exploration tool that can be

One of the effective models is EETCO, an estimation and exploration tool that can be used to assess data center design decisions on **Total-Cost-of-Ownership** (**TCO**) and environmental impact: **https://ieeexplore.ieee.org/document/6557146**.

Conclusion

Based on our experience, we have recognized that optimization is an integral part of cloud migration projects. Processes and skills need to be developed and implemented very early in the adoption process. The more delayed the deployment, the more costly cloud adoption will become.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

CHAPTER 3
Security Sense of Cloud

Introduction

Modern businesses look toward technology to increase their reach and improve their revenue and profits. Remaining competitive and aggressive needs companies to be innovative, and digital transformation is center to that. Data center plays a pivotal role in enabling many digital transformation initiatives. These data centers can reside in the customer premises or in a Co-Located Data Center or Cloud.

Structure

We would be going through these topics in the chapter:

- Data Center Security and ISO 27001 standards and references for Data Center Security

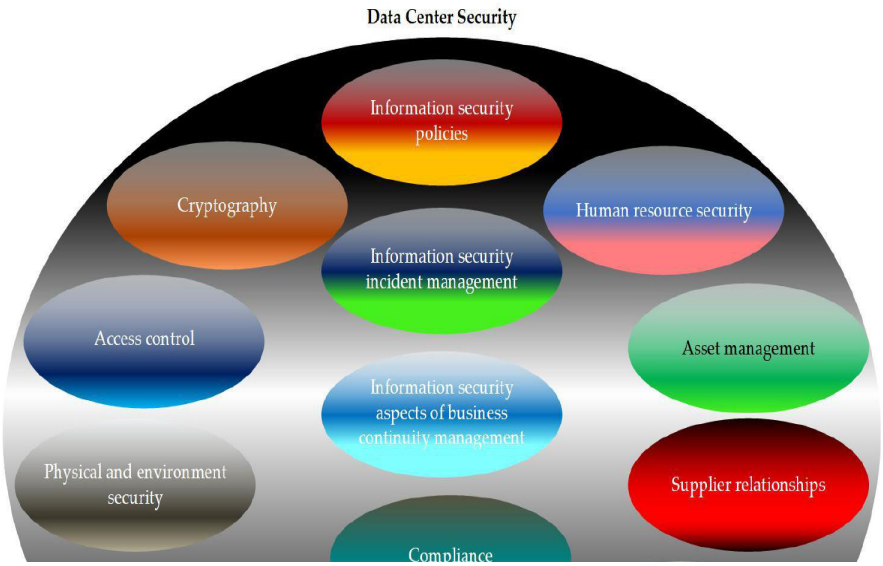
- Cloud Security and how it differs from on-premises security
- Cloud Native Tools Versus Specialized Tools; which tool to go with
- Security Reference Architecture for deploying security in cloud
- Zero Trust Security (ZTS) as a reference architecture

Objectives

Securing the data center deployment is of paramount importance, irrespective of where it is located. In this chapter, we will go through the various aspects of security in Cloud and look at how to ensure business innovation does not get impacted due to security but can provide optimal security.

Data center security

ISO 27001 is an Information Security standard that provides a risk-based approach to managing people, processes, and technical controls. The standard's modular approach to people and technical dependencies ensures that numerous operational benchmarks can be measured, compared, and improved if security gaps are discovered. ISO 27001 is a global standard followed by Data Centers for their security, and large Cloud organizations like Microsoft, Amazon, Google, etc. use this for certifying their data centers for Information Security. Following are the focus areas of ISO 27001:



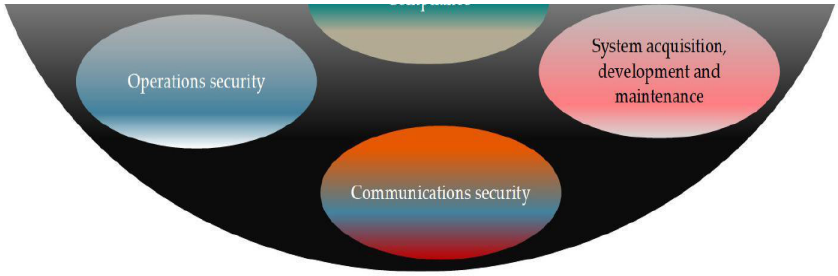


Figure 3.1: Pillars of Data Center Security – ISO 27001

For details of these, read the ISO Standards at **https://www.iso.org/isoiec-27001-information-security.html**.

Cloud security

In this chapter, we will discuss what security standards should be deployed in view of ISO 27001, when using cloud. Let us go through each of these ISO 27001 controls and understand the changes required:

ISO 27001 Control	On-Premises	Cloud
Information security policies	✓	✓
Human resource security	✓	✓
Asset management	✓	●
Access control	✓	●
Cryptography	✓	●
Physical and environment security	✓	✘
Operations security	✓	●
Communications security	✓	●
System acquisition, development, and maintenance	✓	●
Supplier relationships	✓	●
Information security incident management	✓	✓
Information security aspects of business continuity management	✓	✓
Compliance	✓	✓

Table 3.1: ISO 27001; its relevance for cloud and on-premises

✓ → Policy deployment is the responsibility of the Data Center Owner.

● → Only need to follow partial policy.

✗ → Does not need to follow the policy.

Asset management

When using a cloud environment, you will only need to worry about the virtual assets and not need to manage the physical assets. So, it reduces the asset management responsibility by more than 80%.

Access control

Only application access control and server login access control need to be ensured. You no longer need to worry about building access or access into physical locations or facilities.

Cryptography

All the global cloud providers already have encryption enabled for their cloud platform, and the new RDBMS and the new databases that are deployed in the cloud are already encrypted. You only need to manage the keys and ensure that your applications are using the encryption in the correct way. The cloud platform providers already take care of performance and ensure that these encryptions have minimal impact on performance.

Physical and environmental security

Naturally, when deploying a cloud platform, you no longer need to worry about the physical security of the data center as the cloud providers ensure that. Also, based on their global experience and global practices, they provide the best-in-class physical security. Usually, cloud providers like Amazon or Microsoft follow strict ISO and NIST standards for physical security.

Operations security

Since it is a virtual environment, security operations will now only need to manage cloud operations; all other operations like physical and cyber security operations for physical devices are no longer required.

Communications security

Communications security

Communication security will only encompass securing connections with the cloud data center and any work from home or remote connectivity from outside office to the cloud data center.

System acquisition, development, and maintenance

Cloud providers like Amazon, Microsoft, or Google today provide assistance in application development and migrations. Once you have a cloud contract, it becomes easier for you to manage system acquisition and development in a much easier way. Also, the agility in the cloud has enabled better control over sizing and forecasting of cloud usage, enabling application designers to be more creative and innovative.

Supplier relationships

Now you just have one supplier to manage, one neck to catch, and one interface to the supplier world. Marketplace is on the cloud today has enabled infinite possibilities and numerous offerings, which did not exist earlier. Supplier relationships have similarly become simpler, as most of the transactions can be done in cloud marketplaces. Simpler supplier relationships would mean better supplier background security checks and more trusted relationships.

Cloud native tools versus specialized tools

Based on our experience of interacting with multiple CISOs, we normally get into a discussion where a question is always asked, 'Should we deploy cloud native solutions or should we deploy specialized solutions for security?'.

Understandably, there won't be a black and white answer to this question; it will completely depend on the organizational security policies and compliance policies. Let us take an example; if an organization is looking for multi-cloud deployment as a policy, using cloud native security solutions will limit their vision. So even if there are overheads in multi-cloud security deployment, using non-native solution will ensure compliance to policies and vision. Similarly, if there were requirements of the same cloud but multi-region, native firewall would be good enough.

One more aspect that governs the decision is your overall security architecture, i.e., the different security controls you want to put in to ensure that cloud is considered safe to be used in the organization. There are multiple pillars as detailed in ISO 27001 standards, and the security architecture needs to take care of all these aspects. You will have to ensure that each of the controls have proper architecture, security tools, controls, audit and compliance.

There are no great or weak tools for security. It is the architecture and implementation

There are no great or weak tools for security. It is the architecture and implementation that makes security weak or strong. Remember, cloud is adopted by organizations to make them agile, and your security architecture should not be going against that principle. Security by obscurity shouldn't be a thought process in cloud; remember that cloud organizations invest billions annually to ensure their security. When you decide to use them, you are just going to build supplemental security to what is already there in the data center and provided by default by the cloud providers.

There cannot be a scenario where you use a cloud provider and not trust their security. Cloud providers have more at stake than us, and any breach will mean huge amount of loss, both financially and reputational. Your architecture should just be a bridge between the cloud service providers security and your security policies.

We have seen organizations having firewalls on each floor of their building to make them unbreachable. Yet, many have been attacked, vulnerable, and have suffered from security breaches.

Considering this, let's look at reference security architecture that we can recommend for cloud deployment.

Security reference architecture

Security architecture is as important in cloud as in on-premises. There are several reference architectures for deploying security; let us discuss one of the most popular and effective one based on our experience.

Figure 3.2: Cloud Security Reference Architecture

There are multiple sections of security and multiple tools and practices. Let us understand each of these in detail.

Security Operations Center

Security of an organization needs to be managed in **Security Operations Center** (**SOC**), and the following tools are mandatory when running SOC on Cloud:

- Security Incident and Event Management (SIEM)
- Security Orchestration, Automation and Response (SOAR)
- DevSecOps → Development of Security Operations
- Vulnerability management

- Advanced threat protection

Network

Protection of virtual network in the cloud is mandatory for deployment. The cloud provider protects their network, but it is your responsibility to protect the deployment on top of the cloud. The following are the protection that you may consider when deploying in cloud:

- Next-Generation Firewall (NGFW)
- Edge Data Leak Prevention (DLP)
- Security Socket Layer (SSL) Proxy
- Network Intrusion Prevention System / Network Intrusion Detection System (NIPS / NIDS)
- Web Application Firewall
- Network Access Control Lists (ACLs)
- Distributed Denial of Service (DDoS) Protection

Servers

Servers, ports, services, libraries, binaries, files, processors, etc. must be protected to ensure the security of your data. The following are suggested security components:

- Anti-malware
- Host Intrusion Prevention System / Host Intrusion Detection System (HIPS /

- Host Intrusion Prevention System / Host Intrusion Detection System (HIPS/HIDS)
- Update management
- DevOps

Threat detection and protection

It's mandatory to ensure that your system is consistently monitored and any threat is proactively detected. The visibility must be across the platform and all the resources deployed in the cloud. The following are controls that you could put in:

- Cross-platform visibility
- Configuration hygiene
- Access management to protected resources
- Data protection

Policy management

Security policy and compliance can be implemented in cloud in a much easier and effective way using the inbuilt security policy tools of the cloud providers. On-premises security policy management and compliance is always a challenge, and costly tools have to be implemented to achieve that. Cloud providers through their investments have created products that allow you to bring in your own policies and use standard industry policies and deploy policy management with the click of a button. The following are the policy deployments that you should consider when in cloud:

- Policy compliance
- Policy implementation
- Policy audit

Information protection

Cloud providers take effective steps to ensure information protection, and they ensure that customer information is protected at any cost. Yet, when you deploy in cloud, you should ensure that you have the following protections:

- Conditional access
- Discover, classify, detect, monitor (Sensitive Data)
- Information integrity monitoring

Confidential computing

With the advent of sophisticated data centers and sophisticated infrastructure environments, security attacks have also become sophisticated. Encryption of data-in-transit and data-at-rest used to be standard procedures for ensuring data confidentiality. Data theft from processors has now made data encryption mandatory, irrespective whether it is in transit, at rest, or in use. The following are the different mechanisms that you should deploy for data confidentiality:

- Data masking
- Encryption of data in transit
- Encryption of data at rest
- Encryption of data in use
- Encryption key management

Identity and access management

IAM is defining and managing the roles and privileges of users (both network and services) and the policy for granting or denying those privileges. Cloud is no different, and the following should be considered for IAM in Cloud:

- Identity management
- Privilege identity management
- Privilege access management
- External identity management
- Secret management (password management, authentication key management, password-less authentication, etc.)

Zero Trust Security (ZTS)

Cloud has changed the way information technology is used today. All practices were based on assumptions, and all of them have now changed:

- Application users are no longer only internal users; they're also external and partners/vendors.
- Devices are no longer managed by the organization; systems are accessed from miscellaneous devices using external and personal devices.
- Every application today users Cloud in some or the other form.
- Applications have changed shape and form and have lost their monolithic simplicity. They have become more distributed and independent.
- Data is now traveling, getting used and stored in multiple places outside an organization's premises and controls.
- We used to get limited number of security signals, but now it is unlimited, and each signal must be read, understood and used.

A Zero Trust approach is an integrated security philosophy and end-to-end strategy that extends in entire digital estate. Following are important aspects to be considered:

- **Identities**: Identities, whether they represent people, services, or IOT devices, define the Zero Trust control plane. When an identity attempts to access a resource, we need to verify that identity with strong authentication ensures that access is compliant and typical for that identity, and it follows least privilege access principles.

- **Devices**: Once an identity has been granted access to a resource, data can flow to different devices, from IoT devices to smartphones, BYOD to partner managed devices, and on-premises workloads to cloud-hosted servers. This diversity creates a massive attack surface area, requiring us to monitor and enforce device health and compliance for secure access.
- **Applications**: Applications and APIs provide the interface by which data is consumed. They may be legacy on premises, lift and shifted to cloud workloads, or modern SaaS applications. Controls and technologies should be applied to discover Shadow IT, ensure appropriate in-app permissions, give access based on real-time analytics, monitor for abnormal behavior, control user actions, and validate secure configuration options.
- **Data**: Ultimately, security teams are focused on protecting data. Wherever possible, data should remain safe even if it leaves the devices, apps, infrastructure, and networks controlled by the organization. Data should be classified, labeled, and encrypted, and access should be restricted based on those attributes.
- **Infrastructure**: Infrastructure (whether on-premises servers, cloud-based VMs, containers, or micro services) represents a critical threat vector. Assess for version, configuration, and JIT access to harden defense, use telemetry to detect attacks and anomalies, and automatically block and flag risky behavior and take protective actions.
- **Networks**: All data is ultimately accessed over network infrastructure. Networking controls can provide critical "in pipe" controls to enhance visibility and help prevent attackers from moving laterally across the network. Networks should be segmented (including deeper in-network micro segmentation) and real-time threat protection, end-to-end encryption, monitoring, and analytics should be employed.

Each of these six foundational elements serves as a source of the signal, a control plane for enforcement, and a critical resource to defend. You should appropriately spread your investments across each of these elements for maximum protection.

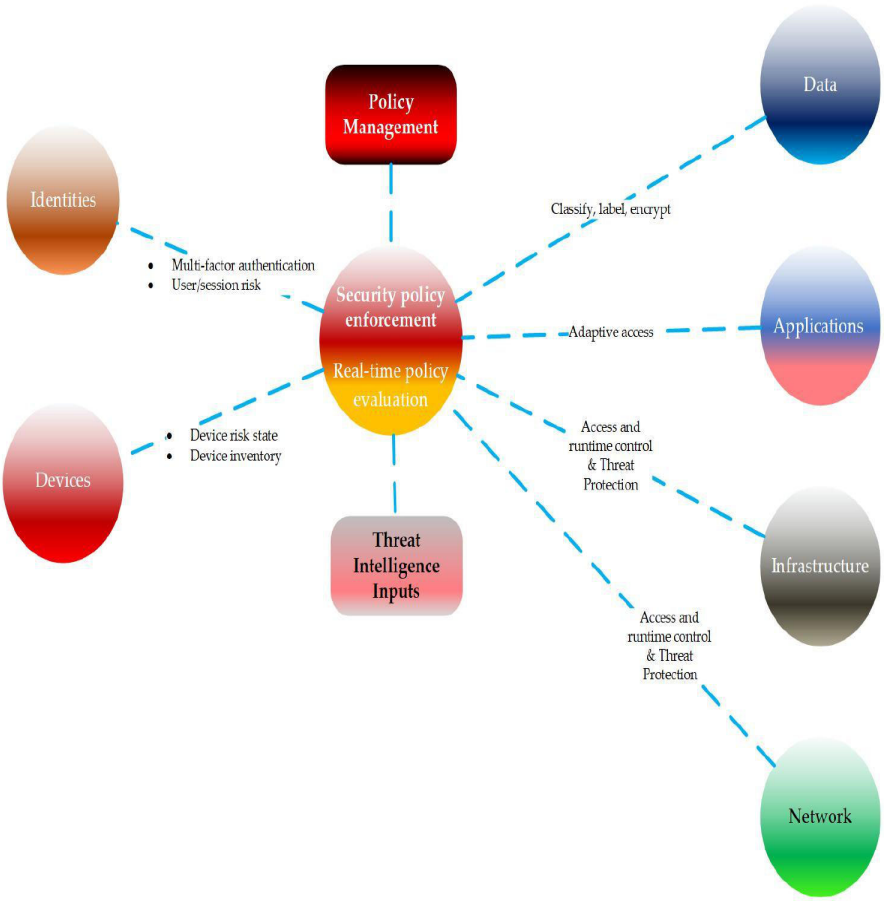


Figure 3.3: Zero Trust Security – Reference Architecture

Zero Trust controls and technologies across your six foundational elements need to do the following:

- **Verify identity**: Knowing who is requesting access is essential, and that identity must be validated explicitly, not inferred from the environment. Ensure that you are secure at the point of access by bringing users into a common identity system and using strong auth and threat intelligence to validate authentication.
- **Verify devices**: All data access requests result in the transfer of that data to a browser or app on a device. Knowing the state of that device is critical in a world where devices can be infected, lost, or stolen. **Mobile Device Management** (**MDM**) and Mobile Application Management are critical to protecting data once it is accessed.
- **Protect data**: Wherever possible, data should be protected from unauthorized transfer by autoclassification and encryption. This protects against intentional or accidental misrouting of downloaded data.

- **Harden applications**: Application access and configuration must be secure to mitigate intrinsic application risks and to ensure that access is governed by policy. Application behavior, including shadow IT, should be understood, and it should be monitored for and protected from anomalies.
- **Protect infrastructure**: Where you are using cloud workloads (IaaS or PaaS), ensure that you are utilizing your cloud fabric according to best security principles, utilizing the intelligence and protection provided.
- **Govern networks**: Mitigate lateral movement by using an intelligent, adaptive segmentation strategy for workloads, monitoring traffic and protecting from anomalous traffic patterns.

Then, the key tools to tie the security aspects together are:

- **Policy driven access**: Modern micro-segmentation means more than networks. It requires us to also gate access based on their role, location, behavior patterns, data sensitivity, client application, and device security. We must ensure that all policy is automatically enforced at the time of access and continuously throughout the session where possible.
- **Automated threat detection and response**: Telemetry from the mentioned systems must be processed and acted on automatically. Attacks happen at cloud speed, so your defense systems must act at cloud speed as well, and humans just can't react quickly enough. Integrate intelligence with policy-based response for real-time protection.

Another aspect of ZTS is ZTNA.

Zero Trust Network Access (ZTNA)

Zero trust network access (**ZTNA**) is a **software-defined perimeter** (**SDP**) of security technologies that operates on an adaptive trust model, where trust is never implicit and access is granted on a "need-to-know" basis. The access is least-privileged basis defined by granular policies. ZTNA gives users seamless and secure connectivity to private network, without exposing apps to the internet.

Conclusion

So, we come to a question that we are always asked, "Is Cloud more secure?".

The global cloud business is $257.9 billion in 2020. This has been growing for several years now, and the growth is evidence that the CISO community has more

several years now, and the growth is evidence that the CISO community has more confidence on Cloud security today and is more confident to adopt cloud. According to security survey by Nominet, 61% of security professionals believe that the risk

of a security breach is the same or lower in cloud environments compared to on-premises. Although the security concerns have not gone, the nature of cloud, the agility, and the spectrum of solution available has ensured that security is no longer the walking elephant in the organization; it is now an agile running horse. Cloud service providers today are investing huge in their cloud security; e.g., Microsoft invests over $1 billion just in security and has a huge team working on it. This is making them technically more advanced than what we could build on-premises. Access to network is restricted, and you now only have access to a few layers in the network, which also reduces the attack areas. Competition in the Cloud space has ensured that the cloud providers keep themselves ahead of curve, and thus, always have newer technologies built into their platform.

That said, we would also like to emphasise that moving into cloud does not mean you become secure by default. You will still need to ensure that you have proper architecture deployed for security and all best practices are followed, some of which we discussed in this chapter.

Brain supplements

Which is the correct word?

- Data Center
- Datacenter
- Data Centre

All the 3 are correct; Data Center is the US English spelling, while Data Centre is the British English one. And Datacenter is just another way to spell it. The following international standards and publications use "Data Center":

- TIA/EIA 942
- ANSI/BICSI 002
- ANSI/BICSI TDDM
- Data Center Dynamics
- Uptime Institute (creators of the Tier systems)
- The Green Grid (creators of the PUE efficiency rating system)

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 4
Availability and Disaster Recovery

Introduction

Organizations today face a myriad of risks, both internal and external, including cyberattacks, natural disasters, power failures, political disruptions, industrial action, and human error. These risks are often unknown, unthought, and unpredictable, and they have the potential to severely interrupt business operations. No organization would want its critical functions disrupted or want to be prevented from accessing essential information. No organization would want its critical data to be lost.

It is, therefore, imperative to have a guideline and standards in place that provide organizations view of how to evaluate risk and how to mitigate it. It will give organizations a perspective on how to maintain the functionality of business operations when an incident occurs.

The guidelines need to be clear on what is disaster recovery and business continuity, and what measures should be taken to overcome disasters. Implementing a **business continuity management system (BCMS)** is the most comprehensive way to ensure

continuity management system (BCMS) is the most comprehensive way to ensure that your organization returns to 'business as usual' as quickly as possible in the event of a disruption.

This chapter will provide best practices, practical recommendation and standards for Disaster Recovery and Business Continuity in line with standard international practices and procedures, when you deploy your workload in cloud.

Structure

We will be covering the following in this chapter:

- Differences between High Availability and Disaster Recovery
- Business Continuity Planning
- Principles of Disaster Recovery and Business Continuity Planning
- ISO Business Continuity Planning and Disaster Recovery Standard ISO 22301
- Recommended guidelines for designing Business Continuity Planning and Disaster Recovery Standard for your organization
- Testing and reviewing Business Continuity Planning and Disaster Recovery Standard

Objectives

Business continuity primarily focuses on preserving an organization's ability to function in the event of a disruptive incident, by ensuring that the most critical business functions continue to operate – even if at reduced capacity – while attending to the disruption. As such, business continuity management is particularly essential for organizations that rely on rapid throughput or offer products and/or services at scale.

Meanwhile, disaster recovery refers to the specific steps taken to recover to full IT functionality after a disruptive incident has occurred. While it shares much ground with business continuity, disaster recovery is a corrective measure that aims to achieve 'business as usual' as soon as possible across all affected IT and related systems.

As such, disaster recovery plans should be for a wide range of potential disruptive events, and they should offer a path to full IT functionality for each event, even if achieving that functionality will take some time. Business continuity planning should consider the same wide range of possible events, but it should focus on ensuring (or preserving) at least a minimum level of functionality across the services

and systems that the organization needs to continue to deliver and operate.

High Availability (HA) vs Disaster Recovery (DR)

These are related but confused concepts. Both concepts are used in system architecture that mitigate against failure. High availability eliminates single points of failure,

and disaster recovery is the process of getting a system back to an operational state after it is rendered inoperative. More importantly, it is useful to note that Disaster recovery picks up when high availability fails.

High Availability

A **single point of failure (SPOF)** is a potential risk posed by a flaw in the architecture, design, implementation, or configuration of a system in which one fault or malfunction causes an entire system to stop operating. SPOFs are undesirable in any system with a goal of high availability or reliability, be it a business practice, software application, or other industrial system.

High availability is about eliminating single points of failure by introducing redundancy. The redundancy can be hardware or software based.

Hardware redundancy removes single points of failure in hardware be building servers which have:

- Redundant storage with RAID or similar technology, which ensures that data is written to and read from multiple physical disks. This prevents data loss and downtime.
- Redundant power, typically in the form of multiple power supplies, enables admins to connect servers to independent power sources so that servers can remain powered on if there was a power loss from one source.
- Error correction that enables data to be healed in the event of data corruption in storage.
- Redundant networking, such as multiple NIC's connected to independent networks, to ensure that a server remains online in the event of network failures.

Software redundancy, wherein software is built to ensure that they could tolerate failures in a system, be it hardware failure, configuration errors, or any number of other reasons that could take down a part of the software. Some of the popularly known methods are listed here:

- Clustering technologies, such as server clusters, spread workloads across multiple servers.
- Encouragement and deployment of stateless architecture in applications provides redundancy and scalability together.
- Use of Load balancing technologies along with application monitoring by way of health probes enables requests to be routed to healthy application nodes and raise events to proactively handle failure.

- With virtualization taking over, self-healing systems have also come into the picture; they move workloads around or allocate additional capacity when failures are detected.

With the rise of cloud computing, cloud providers provide even more advanced redundancy with the following:

- Hardware redundancy on a server rack within data centers to include discrete networking, power, and storage for hardware that allows users to spread workloads to mitigate single points of failure; Azure calls these "fault domains".
- Data center redundancy within a geographic region allows users to run applications in separate data centers that are located geographically close to one another. Amazon Web Services and Azure refer to this as an "availability zone".

All these domains (hardware, software, and environmental) seek to solve the same basic problem by making efforts to eliminate single points of failure.

High availability is measured by a term called **uptime**. The concept of "Uptime" was pioneered by the Uptime Institute, which was founded in 1993 and introduced its well-defined Tier Classification system: I, II, III and IV, of which Tier IV represents the highest level of projected availability. Today, its Tier Certification system is globally recognized.

Uptime is a measure of system reliability, expressed as the percentage of time that a machine, typically a computer, has been working and available. Uptime is the opposite of downtime. It is the guaranteed availability of a data center/application measured over a definite period, e.g., month, quarter, year, etc. Uptime is often used as a measure of computer operating system reliability or stability, in that this time represents the time a computer can be left unattended without crashing or needing to be rebooted for administrative or maintenance purposes.

Disaster recovery

Disaster recovery

Disaster recovery picks up where high availability fails. It can be as simple as restoring data from a backup, but it can also be very complex. DR depends on two very important factors: **Recovery Time Objective** (**RTO**) and **Recovery Point Objective** (**RPO**).

A **Recovery Time Objective** is the maximum amount of time that a system can be down for before it is recovered to an operational state. For some systems, this recovery time objective can be measured in hours or even days, but for more mission-critical systems, the recovery time objectives are typically measured in seconds.

A **Recovery Point Objective** is the amount of data loss, measured in time, that is tolerable in a disaster. For some systems, losing a day's worth of data might be acceptable, while for other systems, this might be mere minutes or seconds.

RTOs and RPOs have a huge impact on how disaster recovery plans are implemented and on the cost of implementation.

Smaller RTOs and RPOs require a system to implement active data replication between primary and recovery systems (such as database log shipping, storage replication, etc.) and maintain failover systems in a ready (expressed as "hot-hot") or near ready ("hot-warm") state to take over in the event of a disaster.

For longer RTOs and RPOs, restoring systems from daily backups might be enough. The process for restoring these may be manual, automated, or both. Whenever backups are used to restore systems to an operational state though, this is typically referred to as a "hot-cold" configuration. In any case, the process of recovering a hot-cold configuration is significantly longer than hot-warm or hot-hot.

One of the biggest factors that prevents organizations from implementing high availability and short RTOs and RPOs is cost. Where HA is concerned, more redundancy requires more resources, which translates into higher costs. Similarly, short RTOs and RPOs require capacity to be available to handle a failover, which also translates into higher costs. There is always a balancing act between costs and system downtime, and sometimes the costs of HA, short RTOs, and short RPOs is not worth it for some apps, while for others it is necessary no matter what the costs may be.

Fundamentally, high availability and disaster recovery are aimed at the same problem: keeping systems up and running in an operational state, with the main difference being that **HA is intended to handle problems while a system is running, and DR is intended to handle problems after a system fails**. Regardless of how highly available a system is, any production application, no matter how trivial, needs to have some sort of disaster recovery plan in place.

Benefits of Business Continuity and Disaster Recovery

Digital India and the rapid digitization of India is increasingly interconnecting organizations and citizens, and being able to respond to an incident quickly is imperative. Even a small glitch can be enough to cause irreparable loss of lives or revenue or have long-term reputational effects. Being able to optimally recover from a potentially damaging event, however, is likely to benefit the organization's reputation. Making business continuity arrangements can minimize any costs incurred by a disruptive incident and improve overall insurance rates.

Being able to demonstrate the ability to recover from and survive such incidents will provide the organization more confidence to adopt IT technology rapidly and ensure that data becomes a treasure for the country.

Finally, adoption of IT by organization will need enhancement of resilience by mitigating risks and successfully recovering from an incident, should one occur. An example of such legislation is the UK's **Network and Information Systems** (**NIS**) Regulations 2018, which aims to ensure the availability of essential services in all but the most adverse circumstances. Another example is the Scottish Public Sector Cyber Resilience Framework, which has dedicated one of its four key domains to 'responding and recovering' from cybersecurity incidents.

Principles of Business Continuity (BC) and Disaster Recovery (DR)

Business Continuity and Disaster Recovery are founded based on the following core principles and activities:

- Securing regulatory body and organization leadership support
- Risk assessment of the organization systems
- Business impact analysis (BIA)
- One or more business continuity plans (BCPs)

Securing regulatory body and organization leadership support

As with any major project, a BC & DR practice must be clearly guided and directed

by the governing body for IT in India. It should also be supported by the organization senior management and leadership for it to have any chance of success.

Guideline / direction from leadership will help ensure the following:

- Necessary resources will be available.
- The BC & DR will be consistent with the overall strategic direction of the organization.
- Continual improvement is promoted.
- The project and management system will be supported throughout the organization.

If organization senior management and leadership provide support throughout the project due to clear directions and guidelines from regulators, the staff is more likely to comply with the BC & DR requirements, making it more effective overall. When planning to implement any system, it is important to remember that the organization is unlikely to commit to a plan that has not been clearly defined. One of the first considerations should be the guidelines, scope, and plan.

Risk assessment

Naturally, to make informed decisions about how and when to continue key business functions, organizations should first consider what scenarios might disrupt them. Risk assessment needs to determine all these scenarios and document them.. This assessment also establishes how likely these disruptive scenarios are and how severe their impact might be.

Ultimately, risk exposure is about the combination of impact – how serious an incident would be if it occurred – and how likely that occurrence is. This combination gives a 'risk score', which can then be compared to the organization's risk acceptance criteria or 'risk appetite', which identifies the level of risk it is willing to accept. This will be heavily influenced by the nature, size, and sensitivity of the organization.

If the risk score is low, you may choose to not do anything about it, thus accepting its existence. If the risk falls outside the risk acceptance criteria, the organization should act. This could be an extreme response, such as suspending an activity altogether, or it could be something more moderate, such as getting insured or providing backups.

Business impact analysis

The BIA, alongside the risk assessment, is the most critical process involved in a BC & DR. It is used to identify an organization's critical activities and resources,

and how severe the organization function's impact would be if those activities were disrupted or those resources were rendered unavailable.

This information is then used to determine the priorities for recovery after a disruption. The BIA will help you work out how quickly each activity needs to be resumed following an incident; it does not concern itself with what those incidents might be (which is a risk assessment output).

A critical outcome of the BIA is a **Recovery Time Objective** (**RTO**) and **Recovery Point Objective** (**RPO**) for each activity or resource. A key consideration is the fact that the impact of an incident usually increases with time; the BIA will determine when that impact becomes unacceptable. This information will then directly inform the RTOs, which, in turn, will form the basis of the **Business Continuity Plan** (**BCP**).

Developing a BIA normally includes the following:

- Identify the scope
- Identify key functional areas
- Identify critical functions
- Identify dependencies between organizations and functions
- Determine the RTO and RPO for each critical function
- Create a BCP

Business continuity plans

The content of the BCP(s) is primarily developed based on the risk assessment and BIA. This ensures that it accurately reflects an organization's needs and specific circumstances.

BCPs often include the following:

- Contact details for authorities, suppliers, and other interested parties.
- Call trees featuring key staff to ensure availability of the right competence.
- Checklists or steps to be taken in the case of specific events.

Ultimately, the goal is to stabilize the situation, allowing the organization to continue operating despite the incident.

ISO 22301

The **International Organization for Standardization** (**ISO**) is an independent nongovernmental organization and the world's largest developer of voluntary international standards. The ISO formed the TC 223 Societal Security technical committee to develop standards for protecting society, including organizations, in the event of catastrophe, such as a natural disaster, a major terrorist attack, or the shutdown of power grids.

Published in 2012 by the technical committee, ISO 22301:2012 is the first international standard for management systems that helps ensure business continuity. ISO 22301 is the premium standard for business continuity, and certification demonstrates conformance to rigorous practices to prevent, mitigate, respond to, and recover from disruptive incidents.

Making ISO 22301 effective

- Organization's leadership and bureaucracy's commitment is key to making this a success.
- All organization staff needs to be informed of what's going on, create a team or assign a champion, as this will increase motivation. This could include a well communicated plan of activities and timescales.
- Think about how different organizations work together to avoid silos.
- Review systems, policies, procedures and processes you have in place; you may already do much of what's in the standard, and make it work for your business.
- Speak to your citizens and suppliers. They may be able to suggest improvements and give feedback on your service.
- Organizations should train their employees to carry our internal audits of the system. This can help with their understanding, and it could also provide valuable feedback on potential problems or opportunities for achievement.

Guidelines for Disaster Recovery and Business Continuity

Following are the guidelines and Framework for BC & DR:

- Every organization having an IT system should have BIA and BCP in place to maintain data protection.
- Based on BIA and BCP, organizations should have a **Primary Data Center** (**PDC**) and a **Disaster Recovery Site/Data Center** (**DRS**).
- Data protection is done by either backing up data in a geographically different location (DRS) or replicating it at a geographically different location (DRS).
- The distance between the two locations needs to be decided based on the following factors:
    - **Earthquakes**: If an organization's PDC location is in a seismic-sensitive area; earthquakes become an important risk to be considered.
    - **Floods**: Organizations should position an alternative site out of the same flood plain.
    - **Tsunamis**: Organizations shouldn't place both PDC and DRS locations on the coast of an ocean.

- **Other natural disasters**: Forest fires, tornados/hurricanes, volcanos, etc.; if PDC is close to such areas, the DRS should be further away.
- **Large industrial facilities, nuclear power plants, military installations or enemy country's prime target locations**: Again, at least one of locations should be at a safe distance.
- **Dependence on the same source of electrical power**: The organization should look for locations on different power grids.
- Even if your risk assessment proves that none of the mentioned points are applicable to you, consider risks like **pandemic diseases, political unrest**; in such cases, authorities will likely close the whole metropolitan area.
- Based on BIA, it is critical to ensure that the DRS is at a safe distance. The distance cannot be so less that it lies in the same city. Wide-range disasters in India in the past are good references for this study.
- Seismic zones cannot be the parameter for DRS. They describe an area with a level of hazard due to earthquakes, but it does not mean that two places classified as the same Seismic zone will have earthquakes occurring at the same time.

- The ideal distance would be more than 800 Km. Some of the references are as follows:
  - Bank of Japan standards: **https://www.boj.or.jp/en/announcements/release_2003/data/fsk0307a.pdf**
    - Page 5, Section (3), states that DC and DR should avoid Geographical Concentration
    - Page 14, Section 5 (2), States that location of DR should be chosen so that it is far enough from the DC and single threat should not impact both DC and DR locations.
    - Page 14, Section 5 (2), also talks about selecting the location so that IT staffing requirements are there.
      - Staffing requirements cannot be met if the same cyclone is hitting the city of flood or pandemic.
      - Federal Reserve – USA Guidelines: **https://ithandbook.ffiec.gov/media/296178/ffiec_itbooklet_businesscontinuitymanagement_v2.pdf**
        - Talks about distance between DC and DR

  - SEBI guidelines: **https://www.sebi.gov.in/legal/circulars/mar-2019/guidelines-for-business-continuity-plan-bcp-and-disaster-recovery-dr-of-market-infrastructure-institutions-miis-_42482.html**
    - Distance of 500 Km between DC and DR
  - AWS DR best practices:
    - **https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/**
    - **https://aws.amazon.com/blogs/startups/large-scale-disaster-recovery-using-aws-regions/**
    - **https://aws.amazon.com/blogs/database/cross-region-automatic-disaster-recovery-on-amazon-rds-for-oracle-database-using-db-snapshots-and-aws-lambda/**

- Both PDC and DRS should have the same set of services. Whatever services the application uses in PDC should be available in DRS and should be configured for failover when disaster occurs.
- The manpower deployed at DRS should have the same expertise as available at PDC in terms of knowledge/awareness of various technological and procedural systems and processes relating to all operations such that DRS can function at short notice, independently. Organizations should have enough trained staff at their DRS to have the capability of running live operations from DRS without involving the staff of the primary site.
- Solution architecture of PDC and DRS should ensure high availability, fault tolerance, no single point of failure, zero data loss, and data and transaction integrity.
- An organization needs to ensure that the uptime SLA that is required in PDC is also deployed in DRS. The uptime needs to be guaranteed from the application and architecture, not only at data center or cloud level.
- Any updates made at the PDC should be reflected at DRS based on RPO without compromising any of the performance metrics.
- Replication architecture, bandwidth and load consideration between the DRS and PDC should be within stipulated RTO and ensure high availability, right sizing, and no single point of failure.
- Replication between PDC and DRS may be asynchronous.
- When organizations need to decide a Data Center for Disaster Recovery, they

should ensure that the Data center is certified with the following:

- ISO 22301 Premium standard for business continuity
- ISO 27002 (17799) - Deals with Information Security; if an organization is using Co-located data center or ISO/IEC 27017:2015 certification, an international standard that aligns with and complements the ISO/IEC 27002:2013 with an emphasis on cloud-specific threats and risks when using a Cloud Service provider.
- ISO 20000 - Information Technology Service Management
- ISO 9001, Quality Management - Record Retention and Data Availability
- ISO 14001, Environmental Mgt - Emergency Preparedness and Response
- TIA-942 standard based Tier 3 Data Center

Testing of BCP and DR drills

Testing a plan is the only way to truly know whether it will work. The BCP must be rigorously tested to know if it's complete and will fulfill its intended purpose. Testing should not go for an easy scenario; always make it credible but challenging. This is the only way to improve. Organizations should also ensure that the objectives are measurable and stretching. Doing the minimum and 'getting away with it' just leads to a weak plan and no confidence in a real incident.

The schedule and frequency of the BCP should depends on the type of organization, the turnover of key personnel and the number of business processes and IT changes that have occurred since the last round of testing.

Common tests include table-top exercises, structured walkthroughs and simulations. Test teams are usually composed of the recovery coordinator and members from each functional unit.

- A **table-top exercise** usually occurs in a conference room with the team poring over the plan, looking for gaps and ensuring that all business units are represented therein.
- In a **structured walk-through**, each team member walks through their components of the plan in detail to identify any weaknesses. Often, the team works through the test with a specific disaster in mind. Some organizations incorporate drills and disaster role-playing into the structured walkthrough. Any weaknesses should be corrected, and an updated plan should be distributed to all pertinent staff.

- It's also a good idea to conduct a **full emergency evacuation drill** at least once a year. This type of test lets you determine whether you need to make special arrangements to evacuate staff members who have physical limitations.

- Lastly, **disaster simulation testing** can be quite involved and should be performed annually. For this test, create an environment that simulates an actual disaster, with all the equipment, supplies and personnel (including business partners and vendors) who would be needed. The purpose of a simulation is to determine whether you can carry out critical business functions during the event.

During each phase of business continuity plan testing, include some new employees on the test team. "Fresh eyes" might detect gaps or lapses of information that experienced team members could overlook.

Review and improve your business continuity plan

Much effort goes into creating and initially testing a BCP. Once that's done, organizations let the plan sit while other, more critical tasks get attention. When this happens, plans will go stale and are of no use when needed.

Technology evolves, and people come and go, so the plan needs to be updated as well. Organizations needs to bring key personnel together at least annually to review the plan and discuss any areas that must be modified.

Prior to the review, feedback should be solicited from ground staff to incorporate into the plan. Ask all organizations or business units to review the plan, including remote locations and units. If an organization has had the misfortune of facing a disaster, it needs to incorporate the lessons learned. It should also conduct a review in tandem with other organizations in a table-top exercise or structured walk-through.

Conclusion

One way to ensure that an organization's plan is not successful is to adopt a casual attitude toward its importance. Every business continuity plan must be supported from the top down. That means the organization's senior leadership and management must be represented when creating and updating the plan; no one can delegate that responsibility to subordinates. In addition, every effort should be made to keep the plan fresh and viable by ensuring that senior management and leader makes it a priority by dedicating time for adequate review and testing.

References

IT Standards

- National Institute of Standards and Technology (NIST)

- Contingency Planning Guide for Information Technology Systems
- IT Governance Institute Standards COBIT
- Control Objectives for Information and related Technology
- ISO 27002:2013 (17799)
- ISO/IEC 27017:2015 certification, an international standard that aligns with and complements the ISO/IEC 27002:2013 with an emphasis on cloud-specific threats and risks
- ISO 22301 Premium standard for business continuity

ISO Standards and Business Continuity

- ISO/TS 16949 - Applicable to any supplier to automotive original equipment manufacturer
- ISO 27002 (17799) - Deals with Information Security
- ISO 9001, Quality Management - Record Retention and Data Availability
- ISO 14001, Environmental Mgt - Emergency Preparedness and Response
- ISO 22301 Premium standard for business continuity

USA BCP Standards for Financial Institutions

- Federal Financial Institutions Examination Council (FFIEC) BCP Handbook 2003
- SEC Rule 17a Record Retention Requirements
- NASD Rule 3510
- NYSE Rule 446
- National Association of Insurance Commissioners (NAIC)
- National Futures Association Compliance Rule 2-38
- Uniform Commercial Code
- Electronic Funds Transfer Act

- Electronic Funds Transfer Act
- Basel Committee's Capital Accords and Sound Practices for the Management and Supervision of Operational Risk

USA BCP Standards for Non-Financial institutions

- Liability of Corporations
- Liability of Corporate Executives
- Liability to Outside Parties
- Standard of Negligence
- Informed Business Judgement v. Gross Negligence

USA Cross-Industry BCP Standards

- Sarbanes-Oxley Act of 2002
- IRS Procedure 86-19

USA BCP Standards for the Healthcare/Life Science Industries

- Health Insurance Portability and Accountability Act of 1996 (HIPAA), Final Security Rule
- FDA's GxP
- 21 CFR Part 11
- FDA Guidance on Computerized Systems in Clinical Trials
- USA BCP Standards for the Energy Industry
- Federal Electric Reliability Council's (FERC) Security Standards for Electric Market Participants, July 2002
- North American Electric Reliability Council's (NERC) Security Guidelines for the Electricity Sector, June 2002

Other National Standards

- Australian Standards BCP Guidelines
- Monetary Authority of Singapore BCP Guidelines
- UK: Turnbull Report
- PAS 56

- UK: Financial Services Authority (FSA) Handbook, Ch. 3 Systems & Control

India

- SEBI's Guidelines for Business Continuity Plan (BCP) and Disaster Recovery (DR) of Market Infrastructure Institutions (MIIs)

CHAPTER 5
Cloud, Agile and Software Development Life Cycle

Introduction

Agile development is iterative and continuous by nature, but it faces several challenges due to the lack of infrastructure and software. There are several instances where we see that teams skip steps and phases in development due to a lack of infrastructure and software.

Cloud plays an important role in helping agile development and ensuring that the delivery timelines are met while due process is also followed. Cloud computing provides flexibility, elasticity and agility to ensure that teams get access to infrastructure and software, and the development process runs smoothly till production.

We will discuss the impact and use of Cloud computing in agile development

methodologies in this chapter and see how to best use them to make the development truly agile.

Structure

In this chapter, we will discuss the following:

- Introducing Cloud computing role in agile development
- Using Cloud computing for efficient agile development

- Using Cloud computing for efficient agile testing
- Using Cloud computing for efficient agile configuration management

Objectives

Agile is one of the most popular methodologies of development currently. It was created to ensure response to changes in an uncertain and turbulent environment. The whole idea of Agile is to be more adaptive, flexible, and responsive to changes. Agile developments emphasize on self-dependent groups and parallel groups. Agile development involves collaboration among various groups working in different geographies and time zones. Community and collaboration are the basic principles of agile development. Ultimately, Agile is a mindset powered by parallelism and independence. Agile software development ultimately achieves business agility.

There are various challenges that you face in Agile development. While we're not going to talk about all of them, we will be talking about some of the challenges that agile faces in terms of continuous development, continuous testing, test-driven development, development methodology, and aesthetics of the development environment.

Introducing Cloud computing role in agile development

In an agile change environment, there are multiple areas where cloud computing would be able to contribute and help handle the challenges, as follows:

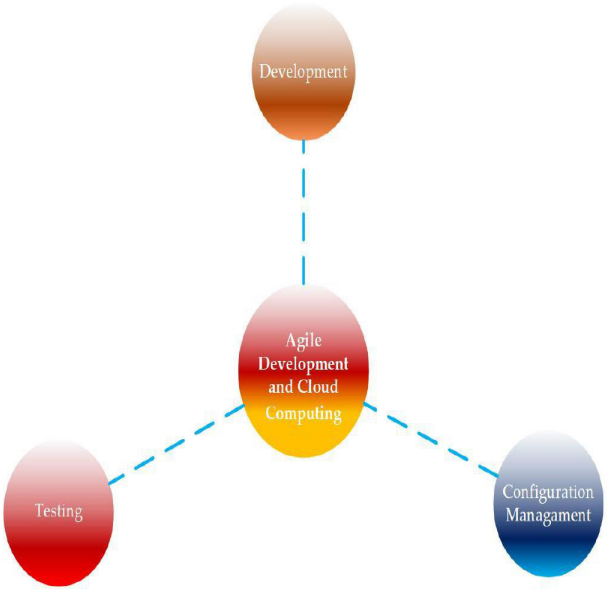


***Figure 5.1**: Solving challenges of Agile Development*

Development

There are multiple areas in which cloud computing can help in Agile development. Introducing cloud computing in agile development ensures that the development process is flexible and quick to adapt changes. There are various challenges in agile development where cloud can play an important role. Following are some of the challenges:

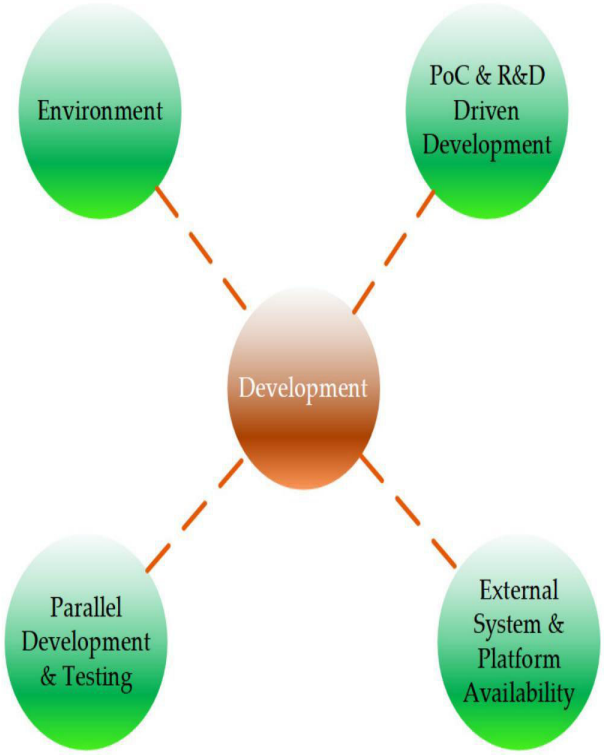


Figure 5.2: Agile Development and Cloud

Development environment

Many times, we have seen development teams struggling to get a proper development environment. This has led to teams finding temporary fixes and silly workarounds to get a development environment. A badly designed development environment or a development environment that doesn't give a production- or testing-like environment ends up having a huge development cycle that is error-prone. Developers and designers struggle to understand the application they are developing, with implications and bugs leaking into production.

As an example, we have seen teams developing on open-source versions of a software when the testing and production would be running on proprietary versions. We have also seen teams who were developing a service-oriented application using simulations of web services.

Cloud enables a development team to set up various types of development environment and gives developers a real environment for development. It also allows developers to create and destroy an environment quickly, bringing in the real agility and change discipline in the process.

Cloud also enables developers to see and visualize how the actual production environment would be. They no longer need to guess the production environment and will very well know what will work and what will not work in the production environment.

Just by eliminating guess work from the development process, you reduce the number of bugs.

Development teams no longer need to wait for days to get a development environment, thereby reducing the change time to minimal. Moreover, the teams can build multiple development environments simultaneously, without worrying about impacting other teams' work.

Cloud also enables development teams to have independent development environments for geographically distributed teams working on different modules/services/microservices. This ensures Agile development and change at rapid pace and with confidence.

Cloud enables development teams to have their own environment similar to the production environment, enabling truly parallel development, and ensuring that independent teams develop their own modules without dependency on a single point of failure.

Parallel development and testing

One of the challenges that development teams regularly face is sequential development and testing processes. The reason for this is either a lack of testing environment or a clash between the versions of code running in the two environments. Several times, we see testers ending up testing on development environment or developers changing the test environment. We also see challenges where the initial planning of projects needs a heavy capital expenditure to build two separate environments for development and testing. Even then, teams are unable to test or develop in parallel.

Hotfixes and emergency changes make the situation even more complicated. Everything stops to ensure that hotfixes are tested and released in production. Hotfixes, hence, have a negative impact on the project, impacting the timelines and the confidence of the development team.

Introducing cloud would mean you have freedom to create as many environments in parallel as you wish, without impacting the others.

Using Cloud, development teams can create separate environments for development and testing. Multiple testing environments would also enable parallelism in the process and increase the speed of development as well as fixes.

Even hotfixes could be dealt with separately and would never impact the normal development and testing sprints. The probability of bugs would decrease with an increase in efficiency. Development would become parallel to testing and as any thing in the world, parallelism would improve efficiency.

Commercially, clouds would also make sense as resources can be deleted after work is done. Since it would reduce the time taken for the development cycle, it will also decrease the development cost. As such, the development of Agile quick turnarounds to changes would enable the true essence of Agile methodology and philosphy.

Proof of Concept (PoC) and Research and Development (R&D) driven development

The ever-changing world of business and competition enforces changes in requirements. The rate of change of requirements is mind boggling and very hard to catch up with. Requirements that worked a quarter ago might no longer be required or might now be modified.

It is the same with technology. It is changing rapidly and becomes redundant within a matter of months. Problem-solving mechanisms change even faster and are replaced by much more efficient and much better mechanisms. Any new development changes will need lots of research before they are implemented. When research is done, product conceptualization cannot be showcased until a pilot or a PoC is done and showcased.

Let's talk about research-based development; every team that is looking to develop new software wants to quickly develop business requirements in the most efficient way. Teams would really like to do more research on a particular concept and develop based on the outcome of the research. Thus, they need an environment that can be quickly created and/or be destroyed, and where research can be done without impacting the actual development. Cloud environment encourages independent environments to be created and new concepts to be tried without impacting the other sprints. It also gives enough sense of security to the research team, that there wrong or destructive experiments don't impact the real environment. It also gives them a good playground that can be created and destroyed as many times as they want, without consuming much time.

Also, cloud environment gives freedom to "taste the pie", and it allows decision makers to see how and what the actual product will be. PoC showcases the feasibility of an idea and also gives freedom to the team to view the usefulness of the idea.

Proof of Concept (PoC), Prototype, Pilot, and Minimal Viable Product (MVP)

One of the important things that we keep discussing is the difference between PoC, Prototype, Pilot, and MVP. The following tables tries to outline the difference between them:

	PoC	Prototype	Pilot	MVP
Environment	Test environment	Test environment	Real environment	Real environment
Method	Minimum investment to test an idea or assumption	Minimum investment to test an idea and how it will work	Substantial investment to test if the system will work in a real environment	Substantial investment to test if the core of the system will work in a real environment
Ideal time	Early stage of projects	Early stage of projects	Production rollout	Live production testing
What is being tested?	Idea or assumption	Idea	Solution	Core of the solution
Success criteria	Feasibility	Success would mean idea works; failure would give learnings	Solution feasibility	Market pulse and solution feasibility
Audience	Internal stakeholders	Sponsors, users	Sponsors, real users	Real users
Time that should be invested	Few days	Few weeks	Few months	Continuous
Best environment to do this	Temporary – Cloud based	Temporary – Cloud based	Portable and temporary – Cloud based	Actual production environment

***Table 5.1**: Agile Development and Cloud*

As you can see, apart from MVP, all the other process would always get benefited using Cloud environments. Using Cloud will enable teams to quickly do R&D and build one of the mechanisms to move ahead.

External system and platform availability

One of the challenges that teams keep facing is the availability of external systems and platforms during the development and testing phases. For example, the development team needs to integrate with an external API on an external server or an external service; if the development is being done on premises, it becomes difficult to create or access such an environment. When using cloud environment, such external systems and platforms can be made available with minimal effort.

Let's take an example: suppose an application needs to have external APIs and should be hosted with API gateway and an API management platform for security and management. When using any of the major cloud platforms providers, they give API management platform on the click of a button. When not using a cloud platform, the team would have to build and/or deploy an API management platform, which will take a substantial amount of time and energy.

Similarly, if an application must be deployed behind a web application firewall with OWASP protection, it is always good to ensure that the development and testing are being done behind a web application firewall. It is often seen that if an application is not deployed/tested behind a firewall, you end up having a huge number of bugs in a real environment.

Cloud platforms provide actual platforms and systems to enable production-like environment, ensuring minimal bugs and quick time to market. Developers will never be in the blind about how the system will integrate with an external system and what would be the mechanism/ precautions to be taken during integration.

Cloud has ensured that the environment that developer's finally use for deploying the production application is made available even in the development environment, helping better agile changes.

Testing

Testing is another challenging area in an agile development environment. Testing environment creation and ensuring that the testing environment is as close to the actual environment is a challenge for all teams. Some of the areas are as follows:

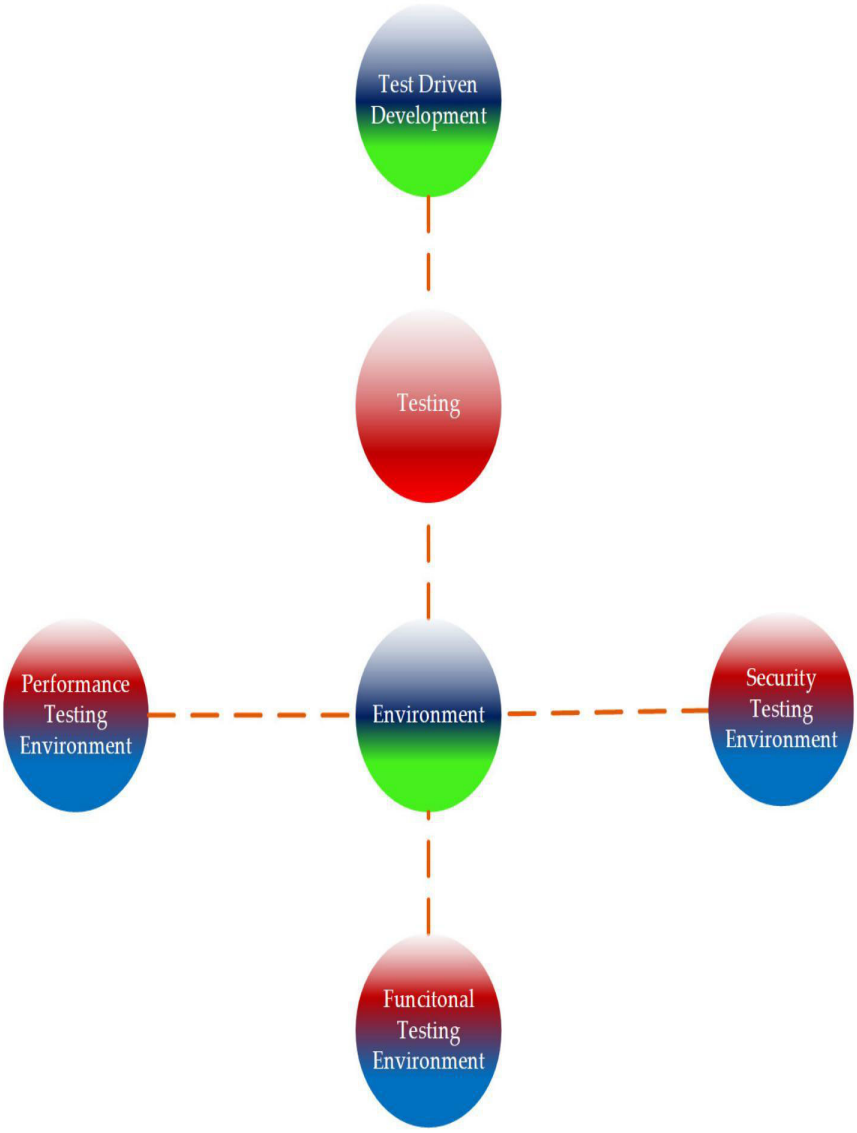


Figure 5.3: Agile Testing and Cloud

Test Driven Development

We will not be talking about **Test Driven Development** (**TDD**) in detail, as it can be a book in itself. On a high-level, Test Driven Development is a progressive approach to application development, which combines testing and development. Developers

write a test before they write production code to fulfill that test. The outcome of test drives the development, and changes in the code are done based on the outcome of the development.

Agile coupled with TDD is one of the best combinations for rapid development. Agile development needs quick time to market, ensuring that it gets driven by testing, makes the process very effective and less error prone. It ensures that on the one hand, the code gets developed quickly and on the other hand, it has minimal bugs.

Cloud has a crucial role in this process. It provides the environment that will enable the development team to write the code and test it more effectively. Cloud has the capability of providing the environment per developer or per module. It is up to the development team to adopt the best method. Development and test environments on Cloud ensure that the testing is done at great pace and closest to the actual environment.

Cloud will provide non-overlapping development and testing environments for the team. They will be able to run automated and manual tests in different environments and will be able to collate test results for better analysis.

Test environment

Apart from this, Cloud also ensures that Agile test teams have greater freedom and greater access to test environments. Two of the common types of non-functional testing that is commonly done is performance testing and security testing.

Performance testing

Usually, performance tests are either not done or are only done when there are performance issues in production. Most times, the reason for this is unavailability of resources (for both tester and testing environment), and these tests are deprioritized.

Cloud is the best option for performance testing. Due to the elastic and agile nature of resource availability, cloud can readily provide the environment for performance testing. It also ensures that the tester has enough client resources to run tests. Additionally, the performance tests can be run unattended and be easily integrated into the CI/CD pipelines.

Using cloud, Agile methodology gets the freedom to ensure that the product can perform on day one in terms of load.

Security testing

Security testing

Very similar to performance test, security tests and checks are deprioritized due to a lack of environment. Teams are under the impression that without security and

performance tests, they can move their agile development ahead quickly. Agile somewhat gives them reason for skipping these important steps.

Cloud enables security test steps by providing a real production environment and ensuring that all security controls are put in place so that real tests are done. All security tools are deployed and all **Access Control Lists** (**ACLs**) are deployed with proper zones in the environment so that application and developers have the "real environment" to test against.

An agile developed application that has been tested for performance and security before release will really decrease time to market and the bug leakage significantly. The change will improve the confidence of the team and will also ensure that the customer is happy with the team.

Functional testing

Functional testing during normal release and during hot fixes are very intensive activities and require both manual and automatic testing. A lack of environment for doing functional testing leads to bug leaks and bad user experience. Cloud can enable functional testing environment availability separately for functional testing in each phase. It also enables quick creation and destruction of environment.

Configuration management

Another area where the agile method faces challenges due to lack of environment is configuration management.

***Figure 5.4**: Agile Configuration Management and Cloud*

Code repository branching

Some of the most critical requirements in an agile development life cycle are

code repository and its branching strategy. The way teams are located at distant geographic location and doing different modules in an agile environment can only be possible with a proper code repository branching strategy.

We're not going to discuss branching strategy here, but we will be discussing the importance of cloud in code repository branching.

There are situations where the branches must be developed and tested in parallel, but a lack of resources and infrastructure prevents teams to perform these tests. Cloud enables infrastructure for parallel development and branching of code repository.

Continuous Integration and Continuous Deployment

CI and CD has a very critical role in Agile development. CI CD ensures that the code is continuously developed, deployed, tested, and stress tested so that it has minimal bugs when it reaches production.

Cloud environment is a facilitator for efficient CI and CD. Using cloud CI and CD applications and platforms can easily create multiple environments and run the pipelines for enabling the teams in their development lifecycles. Cloud environments also enable infrastructure-as-code, thereby enabling CI and CD pipelines to create infrastructure for their own use and ensuring that the infrastructure becomes more agile and elastic in nature. The fun part of cloud infrastructure in CI and CD is that the pipeline itself can create the testers, the testing environment, the performance testers, the security testers, and enable the pipeline that can produce efficient code at the end of the day.

Cloud ensures that CI and CD provide full value in agile methodology and help teams do quicker sprints and efficient scrums.

Conclusion

Cloud provides great flexibility around provisioning of infrastructure and software services. It enables teams to start developing and testing in an environment very similar to production. Cloud computing eliminates dependencies of test and development on physical servers. Cloud environment becomes very useful for agile teams that practice continuous integration and deployment. Cloud enables agile development to become more parallel, rather than being sequential, by eliminating the delays in provisioning. Enterprises can better align innovation with business goals.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 6

Retrofitting Cloud Services Accurately

Introduction

Retrofitting cloud services refers to the process of integrating existing on-premises systems and applications with cloud-based services. This allows organizations to leverage the benefits of cloud computing, such as scalability, cost-effectiveness, and agility, without having to completely replace their existing systems.

There are different approaches to retrofitting cloud services, depending on the specific needs and constraints of the organization. Retrofitting cloud services requires careful planning and consideration of various factors, such as security, compliance, data privacy, and performance. Organizations need to ensure that the integration between on-premises systems and cloud services is seamless, reliable, and scalable, and that any potential risks or issues are addressed proactively.

In this chapter, we will take you through various approaches and share with you our experience, both good and bad. We will also look at how best to select and use the available services.

Structure

We will be covering the following in this chapter:

- Decision to move to cloud

- Requirements and constraints to move to cloud
- Principles of Cloud migration
- Approaches to Cloud migration
- Tools to use during Cloud migration
- Decision tree

Objectives

The objective of retrofitting cloud services is to enable organizations to modernize their IT infrastructure and applications by integrating them with cloud-based services. This can provide several benefits:

- **Scalability**: Cloud services can provide on-demand resources that can be scaled up or down as needed, allowing organizations to respond quickly to changes in demand.
- **Cost-effectiveness**: Cloud services can help reduce IT costs by eliminating the need for upfront investments in hardware, software, and maintenance.
- **Agility**: Cloud services can enable organizations to quickly develop and deploy new applications and services, allowing them to stay ahead of the competition.
- **Innovation**: Cloud services can provide access to cutting-edge technologies and capabilities that may not be available on-premises.
- **Improved user experience**: Cloud services can help improve user experience by providing access to services and applications from anywhere, at any time, on any device.

By retrofitting cloud services, organizations can leverage these benefits while retaining their existing IT investments and systems. This can help organizations achieve a more efficient and effective IT infrastructure that can better support their

achieve a more efficient and effective IT infrastructure that can better support their business objectives.

The objective of this chapter is to explain the various approaches to and learnings that we got while executing the cloud migration and adoption projects.

Decision to move to cloud

Moving to the cloud can provide significant benefits for organizations, but it's essential to consider several factors before making the transition. The organization needs to have a clear understanding of their business goals and objectives; ensure

security and compliance; choose the right cloud service provider; develop a migration strategy; consider the costs; and ensure seamless integration, performance and scalability. By considering these factors, the organization can make an informed decision about moving to the cloud and select the right cloud services to meet their business needs. Let's look at some of the critical factors to consider when moving to the cloud.

Business goals and objectives

The first factor to consider is the organization's business goals and objectives. It's essential to understand what the organization wants to achieve by moving to the cloud. Is it to reduce costs, improve scalability, increase agility, or enable innovation? Having a clear understanding of the business goals and objectives will help determine which cloud services will best meet those needs.

Security and compliance

Security and compliance are critical factors when moving to the cloud. The organization needs to ensure that their data and applications are secure and comply with all relevant regulations and industry standards. The cloud service provider should have robust security measures in place, and the organization should understand their responsibilities regarding security and compliance.

Cloud service provider

Choosing the right cloud service provider is crucial. The organization should evaluate the provider's reputation, experience, reliability, scalability, and support. It's also essential to consider the provider's pricing structure, **service level agreements** (**SLAs**), and the range of services they offer.

Migration strategy

The organization needs to develop a migration strategy that outlines how they will move their data and applications to the cloud. The migration strategy should consider factors like the complexity of the existing infrastructure, the volume of data to be migrated, the required downtime, and the cost of migration.

Cost

Cost is a critical factor when moving to the cloud. The organization needs to consider the cost of the cloud services, the cost of migration, and the ongoing maintenance and support costs. It's important to ensure that the organization can afford to move to the cloud and that the cost savings outweigh the costs.

Integration

The organization needs to consider how the cloud services will integrate with the existing systems and applications. Integration can be complex and can require significant changes to the existing infrastructure. The organization needs to ensure that the cloud services can be seamlessly integrated with the existing systems to avoid disruption to the business.

Performance and scalability

Performance and scalability are critical factors when moving to the cloud. The organization needs to ensure that the cloud services can provide the required performance and scalability to meet the business needs. It should consider factors like the expected workload, peak usage times, and the ability to scale up or down as required.

Requirements and constraints for cloud migration

Moving to the cloud is a complex process that requires careful consideration of various requirements and constraints. Let's look at some of the critical requirements and constraints to consider when moving to the cloud.

Security and compliance requirements

Security and compliance requirements are critical when moving to the cloud. Organizations must ensure that their data and applications are secure and comply with relevant regulations and industry standards. The cloud service provider must have robust security measures in place, and the organization must understand its

responsibilities regarding security and compliance.

Performance requirements

Performance requirements are essential when moving to the cloud. Organizations must ensure that the cloud services can provide the required performance to meet their business needs. Factors like expected workload, peak usage times, and the ability to scale up or down as required must be considered.

Scalability requirements

Scalability requirements are also critical when moving to the cloud. Organizations must ensure that the cloud services can scale up or down as required to meet their business needs, and the cloud service provider must be able to provide the necessary resources to meet the organization's changing demands.

Availability requirements

Availability requirements are also essential when moving to the cloud. Organizations must ensure that the cloud services are available when needed. The cloud service provider must have robust redundancy and failover mechanisms in place to ensure that the organization's data and applications are always available.

Data location requirements

Data location requirements are critical for organizations that need to comply with data sovereignty laws. The organization must ensure that its data is stored in a location that complies with relevant data protection laws and regulations.

Integration requirements

Integration requirements are also critical when moving to the cloud. Organizations must ensure that the cloud services can be seamlessly integrated with their existing systems and applications. Integration can be complex and may require significant changes to the existing infrastructure.

Cost constraints

Cost constraints are an essential consideration when moving to the cloud. Organizations must ensure that they can afford to move to the cloud and that the cost savings outweigh the costs. They must consider the cost of the cloud services, the cost of migration, and the ongoing maintenance and support costs.

Legacy system constraints

Legacy system constraints are also a critical consideration when moving to the cloud. Organizations with legacy systems may face challenges when moving to the cloud and may need to consider options like re-platforming or refactoring their

applications.

Principles for cloud migration

Before moving to cloud, it is important to follow a few principles. Here are some of the best principles to consider when moving to the cloud.

Define a clear cloud strategy

The first and most crucial step when moving to the cloud is to define a clear cloud strategy. This strategy should outline the objectives, benefits, and goals of the cloud migration. It should also define the scope of the migration, the expected outcomes, and the risks involved. A clear cloud strategy will help ensure that the migration is aligned with the organization's business goals.

Assess your current IT infrastructure

Before moving to the cloud, it is essential to assess your current IT infrastructure. This assessment will help identify the systems and applications that can be migrated to the cloud, and any legacy systems that may need to be replaced. It will also help identify any potential security or compliance risks.

Choose the right cloud service provider

Choosing the right cloud service provider is critical to the success of your cloud migration. It is essential to evaluate different cloud service providers and choose one that can meet your organization's needs in terms of security, scalability, performance, and cost.

Develop a cloud migration plan

Developing a cloud migration plan is essential to ensure that the migration process is well-structured and executed smoothly. The plan should outline the steps involved in the migration process, including the timeline, resource allocation, and communication plan. It should also include contingency plans to address any unexpected challenges or issues that may arise during the migration process.

Optimize your cloud infrastructure

Once you have migrated to the cloud, it is essential to optimize your cloud infrastructure continually. This includes monitoring the performance of your systems and applications, scaling your infrastructure as needed, and continuously evaluating your cloud service provider's performance.

Foster a culture of collaboration

A culture of collaboration is essential to the success of your cloud migration. It is essential to involve all stakeholders in the process, including IT, operations, and business teams. This will help ensure that everyone is aligned with the cloud strategy and that the migration process is executed smoothly.

Ensure data security and compliance

Ensuring data security and compliance is critical when moving to the cloud. It is essential to understand the security and compliance requirements of your organization and ensure that the cloud service provider can meet those requirements. This includes implementing appropriate security measures, such as encryption and access controls, and complying with relevant regulations and industry standards.

Hybrid cloud

A hybrid cloud approach involves using a combination of on-premises and cloud-based services. This principle can be useful for organizations that have legacy systems that cannot be moved to cloud but still want to take advantage of cloud services. In a hybrid cloud model, the organization can use on-premises services for some applications and cloud-based services for others. Deciding on this principle can be complex, but it can provide significant benefits in terms of flexibility and cost savings.

Approaches to cloud migration

There are various approaches to cloud migration, each with its own benefits and challenges. Let us explore the different approaches and then discuss retrofitting services in each approach.

Rehost or lift & shift

The rehost approach is the simplest way to retrofit a legacy system with cloud services. This approach involves moving the existing application to the cloud without making any changes to the application. The application runs in the cloud environment without any modification. The advantage of the lift and shift approach is that it can be done quickly and with minimal disruption to the business. However, this approach does not take advantage of all the capabilities of the cloud, and it may not be cost-effective in the long run.

Refactor

Refactoring involves making significant changes to the application's code to take

advantage of cloud services fully. This approach involves breaking down the application into smaller components that can be deployed separately and taking advantage of cloud services like serverless computing and microservices architecture. Refactoring can be time-consuming and costly, but it can provide significant benefits in terms of scalability, agility, and cost savings.

Revise or rearchitect

The revise or rearchitect approach for cloud migration involves making significant changes to the existing applications, systems, or architecture to make them compatible with the cloud environment. This approach is typically taken when the existing applications or systems cannot be migrated to the cloud without significant modifications. The revise or rearchitect approach involves analyzing the existing applications, systems, or architecture to determine which parts can be migrated to the cloud and which parts need to be revised or rearchitected. This approach involves a detailed review of the existing infrastructure, applications, and data to identify any issues or constraints that may affect the migration process.

Rebuild

Rebuilding is the most comprehensive approach to cloud migration. This approach involves rebuilding the application from scratch using cloud-native technologies like containers and Kubernetes. Rebuilding allows the application to take full advantage of the cloud's capabilities and can provide significant benefits in terms of scalability, resilience, and agility. However, rebuilding can be costly and time-consuming, and it may require significant changes to the application's architecture and design.

Replace

The Replace approach for cloud migration involves completely replacing the existing applications or systems with new cloud-native applications or systems. This approach is typically taken when the existing applications or systems are outdated, inefficient, or difficult to migrate to the cloud environment. It involves building new cloud native applications or systems that can replace the existing applications or systems. The new applications or systems are designed and developed specifically for the cloud environment, taking advantage of the cloud's scalability, flexibility, and cost-effectiveness.

Tools to use during cloud migration

Cloud migration can be a complex and challenging process that requires careful planning and decision-making. Fortunately, there are several decision tools available to help organizations evaluate their options and make informed decisions about

cloud migration. Let's discuss some popular decision tools for cloud migration.

Cloud readiness assessment

This tool evaluates an organization's existing infrastructure and applications to determine their readiness for migration to the cloud. It can help identify potential challenges and provide recommendations for optimizing the migration process.

Total Cost of Ownership (TCO) calculator

This tool helps organizations compare the costs of running their applications and infrastructure on-premises versus in the cloud. It considers factors like hardware, software, maintenance, and personnel costs to provide a comprehensive cost analysis.

Cloud migration assessment

This tool evaluates an organization's existing environment and provides a detailed plan for migrating to the cloud. It can help identify the most suitable cloud deployment model (public, private, or hybrid), migration strategy, and necessary resources.

Cloud vendor comparison tool

This tool allows organizations to compare different cloud vendors and their offerings, including pricing, features, security, and support. It can help organizations make an informed decision when choosing a cloud provider.

Cloud security assessment

This tool evaluates an organization's security posture and provides recommendations for securing their cloud environment. It can help identify potential vulnerabilities and provide guidance on how to mitigate them.

Overall, these decision tools can be helpful in evaluating different options and making informed decisions about cloud migration. However, it's important to note

making informed decisions about cloud migration. However, it's important to note that no single tool can provide a complete solution, and organizations may need to use a combination of tools to achieve their goals.

Selecting and fitting best cloud services

Now that we have all the approaches and tools, let's start discussing cloud service selection and the aspects we should consider when selecting services in each approach. We normally follow this decision tree when selecting a service for architecture:

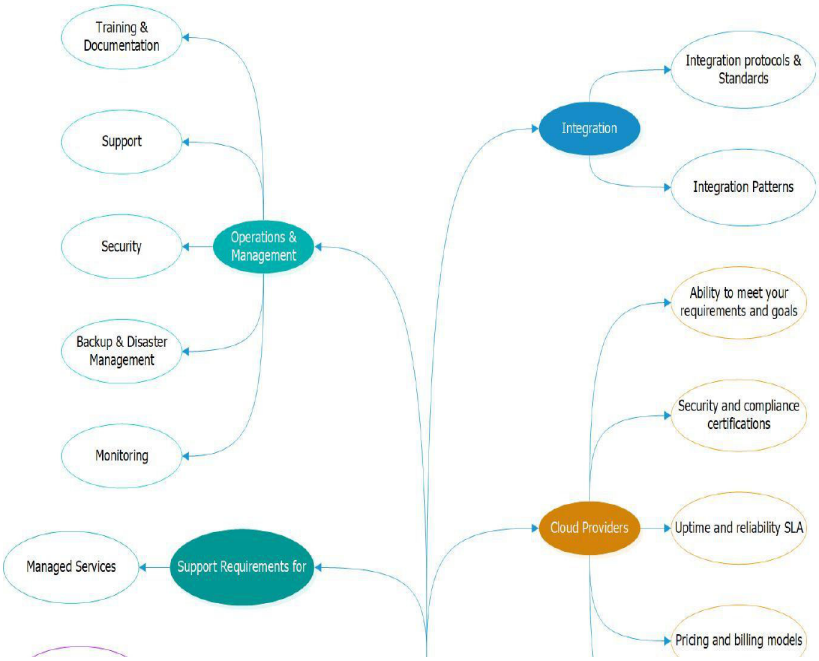


Figure 6.1: Decision Tree to select a cloud service

Workload sensitivity

We must classify the sensitivity of workload because that will govern policies of management and non-functional requirements.

Critical workload: A critical workload refers to a specific type of workload that is deemed essential for the functioning of an organization or system. The failure of or any disruptions in this workload can have significant impact on the organization or system's performance, safety, or security. For example, critical workloads can include systems that are responsible for financial transactions, medical equipment that supports life-saving procedures, or communication networks that support emergency services. These workloads often require high levels of reliability, availability, and performance to ensure that they function effectively and without interruption.

Non-critical workload: Non-critical workloads are not essential for the functioning of an organization or system. They are typically less important and have lower priority compared to critical workloads. These workloads can include tasks like routine maintenance, administrative tasks, or non-essential services. While they may still be important for the overall functioning of the organization or system, their failure or disruption would not have the same level of impact as that of a critical workload. Because non-critical workloads are less important, they may not receive the same level of resources, attention, or prioritization as critical workloads

receive the same level of resources, attention, or prioritization as critical workloads. However, they still play an important role in supporting the overall functioning of the organization or system.

Non-functional aspects

Reliability, availability, scalability, and performance are four key characteristics that are essential to the design, implementation, and operation of modern systems and services. These characteristics are interdependent and often need to be balanced against one another to achieve the optimal system design.

Reliability: Reliability refers to the ability of a system or service to operate continuously without interruption or failure. It is the degree to which a system performs its intended function accurately and consistently under a specified set of conditions. A reliable system is one that can withstand unforeseen events, such as hardware or software failures, without compromising its performance or availability. To ensure reliability, a system may be designed with redundancy, which means multiple instances of a particular component or system are deployed to ensure that if one fails, the other can take over. Additionally, a system may be designed with fault tolerance, which means it can continue to operate in the presence of failures.

Availability: Availability refers to the degree to which a system or service is operational and accessible when it is needed. It is the measure of how often a

system is able to perform its intended function during a specified time period. A highly available system is one that can provide its services continuously, without interruption or downtime. To ensure availability, a system may be designed with redundancy, load balancing, or failover mechanisms. These mechanisms ensure that the system is able to provide its services even if one or more components fail.

Scalability: Scalability refers to the ability of a system or service to handle increasing amounts of work without compromising its performance or availability. It is the measure of a system's ability to handle growth or increased demand over time. To ensure scalability, a system may be designed with horizontal or vertical scaling. Horizontal scaling involves adding more instances of a component or system to distribute the workload, while vertical scaling involves increasing the resources of a single instance of a component or system. Additionally, a system may be designed with autoscaling mechanisms that can automatically adjust the number of instances based on the workload.

Performance: Performance refers to the speed and efficiency with which a system or service performs its intended function. It is the measure of how well a system can meet its performance requirements under a specified set of conditions. To ensure performance, a system may be designed with optimized code, caching mechanisms, load balancing, or database indexing. These mechanisms ensure that the system can handle a high volume of requests while maintaining a fast response time.

Technical workload type

Nowadays, we have several types of applications that require different strategies to get hosted in cloud:

Monolithic application: A monolithic application is a software application that is designed and built as a single, self-contained unit with all its components and modules tightly coupled together. In a monolithic architecture, all the features and functions of the application are packaged together in one codebase and deployed as a single executable file. These applications typically have a single point of failure and can be challenging to scale and maintain over time. Any updates or changes made to one component of the application require the entire application to be recompiled and redeployed. This can make it difficult to iterate quickly and deploy updates frequently. Despite these limitations, monolithic applications have been popular for many years because they are relatively easy to develop and deploy. They can also be highly performant because all the components are tightly integrated and optimized for the specific application. However, as software applications have become more complex and the demand for faster and more frequent updates has increased, many organizations have started to move towards microservices and other distributed architectures. Microservices architectures, in particular, offer greater scalability, fault tolerance, and flexibility by breaking down an application into smaller, independent services that can be developed, tested, and deployed separately.

Microservice application: A microservice application is a software application architecture that is designed as a collection of small, independently deployable services that work together to provide the overall functionality of the application. Each microservice is responsible for a specific business capability and communicates with other microservices through lightweight protocols, typically HTTP or messaging. Microservices architectures are known for their flexibility, scalability, and fault tolerance. They enable continuous delivery, faster deployment times, and easier maintenance of individual services. Microservices also enable organizations to adopt new technologies and platforms more easily, as each service can be developed and deployed independently of the others.

Service-oriented application: A **Service-Oriented Architecture** (**SOA**) is a software design approach that structures applications as a set of loosely coupled, interoperable services that communicate with each other over a network. Each service is designed to provide a specific business functionality and can be accessed by other services or applications in a standardized way. SOA offers flexible, scalable, and modular architecture that allows for easy integration of the existing systems, reuse of services, and rapid development of new applications. SOA enables organizations to achieve agility and reduce development costs by abstracting and encapsulating business functionality as services. In summary, a service-oriented application is one that follows the principles of SOA, where services are the primary building blocks of the application and are designed to be modular, reusable, and interoperable.

Big Data/Data Warehouse application: A Big Data/Data Warehouse application is a software application that is designed to manage, store, and analyze large volumes of structured, semi-structured, and unstructured data. This type of application typically uses distributed computing technologies to process and analyze large data sets in parallel. Data warehouses, on the other hand, are a type of data storage and management system that is optimized for querying and analyzing large data sets. They are designed to support complex queries and provide a single source of truth for the organization's data. Big Data/Data Warehouse applications are essential for organizations that need to process and analyze large volumes of data to gain insights and make data-driven decisions. They enable organizations to store and analyze data from multiple sources and provide a holistic view of the organization's data. These applications can be used in various industries, including finance, healthcare, and e-commerce.

Technical capability requirement of workload

There are several protocols that can be used to access an application. Here are a few examples:

- **Hypertext Transfer Protocol (HTTP)**: This is a widely used protocol for accessing web applications over the internet. It is used to request and

transmit web pages and other resources, such as images and videos, from web servers to web browsers.

- **HTTP Secure (HTTPS)**: This is a secure version of HTTP that uses encryption to protect the data transmitted between a web server and a web browser. It is often used for online transactions and other sensitive data.
- **File Transfer Protocol (FTP)**: This is a protocol used to transfer files between computers over a network. It is often used for large files, such as software updates and multimedia files.
- **Secure Shell (SSH)**: This is a protocol used to access and control remote servers over a network. It provides a secure way to log in and execute commands on a remote machine.
- **Remote Desktop Protocol (RDP)**: This is a protocol used to access a remote desktop or computer over a network. It allows users to control a remote machine as if they were sitting in front of it.
- **Telnet**: This is a protocol used to access a remote computer or server over a network. It allows users to log in and execute commands on a remote machine.
- **Simple Mail Transfer Protocol (SMTP)**: This is a protocol used to send email messages between email servers. It is used to transfer email messages from the sender's email server to the recipient's.

These are just a few examples of the many protocols used to access applications over a network. The choice of protocol depends on the specific application being accessed and the requirements for security, speed, and reliability.

Several protocols can be used for data transfers; some of the most common ones are listed here:

- **Hypertext Transfer Protocol (HTTP)**: This is the protocol used for transferring data over the internet. It is used for web browsing, file transfers, and other types of data transfer.
- **File Transfer Protocol (FTP):** This protocol is used for transferring files between computers. It is commonly used for uploading and downloading files to and from a server.
- **Simple Mail Transfer Protocol (SMTP):** This protocol is used for transferring email messages between servers. It is used to send email messages from one email client to another.
- **Transmission Control Protocol/Internet Protocol (TCP/IP):** This is a set of protocols used for transferring data over the internet. It provides reliable and

error-free transmission of data.

- **User Datagram Protocol (UDP):** This protocol is used for transmitting data that is not critical, such as video and audio streams. It provides fast, efficient transmission of data but does not guarantee delivery.
- **Secure Shell Protocol (SSH):** This protocol is used for securely transferring data over an insecure network. It is commonly used for remote login and file transfer.
- **Domain Name System (DNS):** This protocol is used for translating domain names into IP addresses. It is used by web browsers to locate web servers on the internet.

These were a few examples of the many protocols used for data transfer. The choice of protocol depends on the type of data being transferred, the network being used, and other factors. These are as follows:

- **Volumetrics**: There are several parameters used to define volumetrics of an application. Here are some of the most important ones:
    - **Number of users**: The number of users who will be accessing the application helps determine the amount of processing power, memory, and network bandwidth required to handle user requests.
    - **User activity**: The frequency and complexity of user actions within the application helps determine the amount of processing power and network bandwidth needed to support user interactions.
    - **Data volume**: The amount of data stored and processed by the application, including user data, application data, and any associated files, helps determine the amount of storage and memory required.
    - **Transaction volume**: The number of transactions processed by the application, including data input/output, data processing, and any automated processes, helps determine the amount of processing power and network bandwidth needed to support transaction processing.
    - **Traffic patterns**: The amount and timing of incoming and outgoing network traffic generated by the application helps determine the amount of network bandwidth required to support the application.
    - **Performance requirements:** The desired level of application performance, including response time, availability, and reliability helps determine the resources required to meet performance requirements.

    By considering these parameters, application developers and administrators can estimate the volumetrics of the application and design an infrastructure

can estimate the volumetrics of the application and design an infrastructure

that meets the application's requirements while ensuring optimal performance and scalability.

- **Storage requirements**: There are several parameters that can be used to determine storage requirements for an application. Here are some of the most important ones:
    - **Data type and size**: This is the type of data being stored and the amount of space required for each data type. Different data types may require different storage configurations, such as relational databases or NoSQL databases.
    - **Data growth rate**: The rate at which data is expected to grow over time helps determine the amount of storage needed to accommodate future data growth.
    - **Data access patterns**: The frequency and type of access to data helps determine the appropriate storage medium, such as **solid-state drives** (**SSDs**) or **hard disk drives** (**HDDs**), and the data access speed required.
    - **Data retention policies**: The duration that data needs to be stored for and any regulatory requirements for data retention help determine the amount of storage space needed and the appropriate storage configuration.
    - **Backup and recovery requirements**: The frequency and type of backup and recovery needed to protect data help determine the amount of storage needed for backup and recovery purposes.
    - **Disaster recovery requirements**: This is the amount of storage needed for disaster recovery purposes, including backup and replication of data across multiple locations.
    - **Archiving requirements**: This is the amount of storage needed for long-term archiving of data that is not frequently accessed but must be retained for regulatory or business purposes.

By considering these parameters, application developers and administrators can estimate the storage requirements for the application and design a storage infrastructure that meets the application's needs while ensuring optimal performance and scalability.

Integration

Integration

There are various integration protocols and standards in computer networking; here are some of the most common ones:

- **RESTful APIs**: REST stands for Representational State Transfer, and it is a set of architectural principles for building web services. RESTful APIs are stateless, meaning each request contains all the necessary information to complete the transaction.
- **SOAP**: **Simple Object Access Protocol (SOAP**) is an XML-based messaging protocol that is used to exchange structured data between web services. It provides a standardized way of communicating between different systems and platforms.
- **GraphQL**: It is a query language and runtime for APIs that was developed by Facebook. It allows clients to specify the structure of the data they need and only returns that data, making it more efficient than traditional RESTful APIs.
- **Message Queuing Telemetry Transport (MQTT):** MQTT is a lightweight messaging protocol that is used for IoT (Internet of Things) devices. It is designed to be low power and low bandwidth, making it ideal for devices with limited processing power.
- **Remote Procedure Call (RPC):** RPC is a protocol that allows a program on one computer to execute a program on another. It is used for distributed computing and can be used to build complex systems with multiple components.
- **Representational State Transfer (REST):** REST is an architectural style that uses HTTP requests to access and manipulate data. It is often used for building web services and APIs.

These were just a few examples of integration protocols; there are many others out there. The choice of protocol depends on the specific needs and requirements of the system being built.

Integration patterns refer to the design patterns that are used to solve common integration problems in software development. Here are some of the most common integration patterns:

- **Message-oriented middleware:** This pattern involves using a middleware layer to manage communication between different components in a distributed system. Messages are sent between components, and the middleware handles the routing and transformation of the messages.
- **Publish/Subscribe**: This pattern involves publishers sending messages to a message broker, which then distributes the messages to subscribers. This

pattern is useful when there are multiple consumers of the same data.

- **Request/Reply**: This pattern involves one component sending a request to another component and waiting for a response. It is useful when a component needs to get data from another component.
- **Service-Oriented Architecture (SOA)**: SOA is a pattern that involves breaking down a system into a set of independent services that can communicate with each other using well-defined interfaces. This pattern makes it easier to scale and maintain complex systems.
- **Batch Processing**: This pattern involves processing large amounts of data in batches rather than processing each record individually. This pattern is useful when there are large amounts of data to be processed.
- **Remote Procedure Invocation (RPI):** This pattern involves calling a remote procedure as if it were a local procedure. It is useful when there are multiple components in a system that need to communicate with each other.
- **Data Integration**: This pattern involves integrating data from multiple sources into a single system. It is useful when there are multiple sources of data that need to be combined and processed.

These were just a few examples of integration patterns; there are many others out there as well. The choice of pattern depends on the specific needs and requirements of the system being built.

Operations and management

Operations and management are integral parts of any deployment and play major roles post deployment. Operations ensures smooth running of workloads, and we need different processes to achieve that:

Monitoring

Monitoring refers to the process of observing and measuring the performance and status of a system or application over time. It involves collecting data on various metrics, such as system resources, application performance, network traffic, and user activity, and analyzing that data to identify patterns, trends, and anomalies. The goal of monitoring is to ensure that a system or application is functioning as expected and to detect any issues or problems as quickly as possible. Monitoring can also help identify areas for optimization and improvement, and track changes over time to evaluate the effectiveness of those improvements. Monitoring can be done using various tools and techniques, such as performance monitoring tools, log analysis tools, network monitoring tools, and user activity monitoring tools. Some of the common metrics that are monitored are CPU usage, memory usage, disk space, network latency, response time, error rate, and availability. In addition to traditional system and application monitoring, there are specialized types of monitoring, such

as security monitoring, compliance monitoring, and environmental monitoring (for example, temperature and humidity monitoring in a data center).

Backup and disaster management

Backup and disaster management are related but distinct concepts in the field of IT management. Backup refers to the process of creating copies of data and files to protect against data loss due to accidental deletion, hardware failure, or other unforeseen events. Backup can be done using various methods, such as local backups to external drives, network backups to remote storage, and cloud backups to offsite data centers. The frequency and scope of backups depend on the organization's needs and the criticality of the data being backed up. Disaster management, on the other hand, is the process of preparing for and responding to catastrophic events, such as natural disasters, cyberattacks, or other disruptions that could impact the organization's ability to operate. Disaster management involves creating and testing disaster recovery plans, which outline the steps to be taken to restore critical systems and data in the event of an outage or disaster. Disaster recovery plans typically involve a combination of backup and recovery strategies, such as redundant data centers, backup power systems, and failover systems that can take over in the event of a primary system failure. Disaster recovery plans also typically include procedures for communication and collaboration, such as contact lists, crisis management teams, and communication protocols.

Security

Security refers to the implementation of measures and controls to protect the underlying infrastructure used to deploy applications and services. This infrastructure includes hardware, software, networks, and data centers, among others. The goal of security in infrastructure deployment is to ensure the confidentiality, integrity, and availability of the infrastructure and the data and applications that are deployed on it. This involves a range of security measures, such as access controls, encryption, monitoring, and vulnerability management. Access controls involve limiting access to the infrastructure to authorized users and implementing measures to prevent unauthorized access, such as strong authentication and password policies, role-based access controls, and network segmentation. Encryption is used to protect data in transit and at rest by converting it into a form that is unreadable without the proper decryption key. Encryption is commonly used to secure sensitive data, such as payment information and personal identification. Monitoring involves the collection and analysis of data to identify and respond to potential security threats. This includes log analysis, intrusion detection and prevention systems, and real-time threat intelligence. Vulnerability management involves identifying and remediating security vulnerabilities in the infrastructure and ensuring that all software and firmware are up to date with the latest security patches.

Support

The support function plays a critical role in maintaining the availability, performance, and security of IT infrastructure, and it is often the first line of defense in resolving technical issues and incidents. The key responsibilities of the support function in IT operations may include the following:

- **Troubleshooting and issue resolution**: The support function is responsible for identifying, diagnosing, and resolving technical issues related to hardware, software, and applications. This involves working closely with end users to understand their needs and provide timely and effective solutions to their problems.
- **Incident management**: The support function is responsible for managing incidents and ensuring that all incidents are promptly and accurately recorded, escalated, and resolved. This involves adhering to established incident management processes and procedures, and communicating effectively with stakeholders and other support teams.
- **Service requests**: The support function is responsible for managing service requests and ensuring that they are processed in a timely and efficient manner. This involves triaging service requests, assigning tasks to appropriate teams or individuals, and tracking progress to ensure that service levels are met.
- **System monitoring and maintenance**: The support function is responsible for monitoring the performance and availability of IT systems and applications, and for performing routine maintenance tasks, such as software updates and system backups.
- **Knowledge management**: The support function is responsible for creating, maintaining, and sharing knowledge related to IT systems and applications, and for providing training and support to end users.

Training and documentation

The role of the training and development in IT operations is to equip employees with the necessary knowledge, skills, and competencies to effectively perform their job duties and support the organization's IT infrastructure and applications. The training and development function plays a critical role in ensuring that employees are properly trained and prepared to meet the evolving needs of the organization and to keep up with emerging technologies and best practices. The key responsibilities of the training and development function in IT operations may include the following:

- **Conducting needs assessments**: The training and development function is responsible for conducting needs assessments to identify skill gaps and training needs across the organization. This involves working closely with

managers and employees to understand their needs and identify areas where additional training and development are required.

- **Developing training programs**: The training and development function is responsible for designing, developing, and delivering training programs to address the identified skill gaps and training needs. This may include classroom training, e-learning modules, on-the-job training, and other types of training and development programs.
- **Evaluating training effectiveness**: The training and development function is responsible for evaluating the effectiveness of training programs and making the necessary adjustments to ensure that they are meeting the needs of the organization and its employees.
- **Supporting new hire onboarding**: The training and development function is responsible for supporting new hire onboarding and ensuring that new employees are properly trained and prepared to support the organization's IT infrastructure and applications.
- **Promoting continuous learning**: The training and development function is responsible for promoting a culture of continuous learning and development across the organization. This may involve providing access to resources like online training courses, certification programs, and other professional development opportunities.

Support requirements for managed services

A managed service is a type of service delivery model where a third-party provider assumes responsibility for the maintenance, monitoring, and management of a customer's IT infrastructure or applications. This includes tasks like software updates, security patching, backups, and disaster recovery. Cloud providers can deliver managed services to their customers by offering a range of managed services that are designed to meet the specific needs of their customers. Some of the managed services offered by cloud providers are as follows:

- **Managed compute**: Cloud providers offer managed compute services, such as **virtual machines** (**VMs**) and containers, which are fully managed by the provider. This includes tasks like patching, updating, and scaling the compute resources to meet the needs of the customer.
- **Managed storage**: Cloud providers offer managed storage services, such as block, file, and object storage, which are fully managed by the provider. This includes tasks like backups, disaster recovery, and data replication.
- **Managed database**: Cloud providers offer managed database services, such as relational and NoSQL databases, which are fully managed by the provider.

This includes tasks like backups, disaster recovery, and automatic scaling.

- **Managed security**: Cloud providers offer managed security services, such as threat detection and response, identity and access management, and encryption, which are fully managed by the provider. This includes tasks like security monitoring, vulnerability scanning, and patch management.
- **Managed networking**: Cloud providers offer managed networking services, such as **virtual private cloud** (**VPC**) and load balancing, which are fully managed by the provider. This includes tasks like network security, monitoring, and troubleshooting.
- **Managed monitoring**: Cloud providers offer managed monitoring services, such as logging and monitoring, which are fully managed by the provider. This includes tasks like log analysis, performance monitoring, and alerting.

Cloud providers

Cloud providers like Azure, AWS, GCP, and Alibaba provide services like virtual machines, network, region availability, and the like. It is important that you check various aspects before placing your workload in a Cloud provider; some of them are listed here:

Reputation and track record

It is important to check the reputation and track record of a cloud provider because the cloud provider is responsible for hosting critical data and applications for the organization. The cloud provider's reputation and track record can provide insights into their ability to provide reliable, secure, and high-quality cloud services, and into their ability to respond effectively to issues and incidents. Here are a few reasons why checking the reputation and track record of a cloud provider is important:

- **Reliability**: A cloud provider's reputation and track record can provide insights into their reliability in terms of uptime, availability, and performance. A provider with a strong reputation and track record is more likely to provide reliable services that meet or exceed the organization's needs.
- **Security**: A cloud provider's reputation and track record can provide insights into their ability to provide secure cloud services. A provider with a strong reputation and track record is more likely to have robust security measures in place to protect data and applications from unauthorized access and other security threats.
- **Compliance**: A cloud provider's reputation and track record can provide insights into their ability to comply with industry standards and regulations. A provider with a strong reputation and track record is more likely to have a track record of compliance with relevant standards and regulations, which

can provide assurance to the organization that their data and applications are being hosted in a compliant manner.

- **Responsiveness**: A cloud provider's reputation and track record can provide insights into their ability to respond effectively to issues and incidents. A provider with a strong reputation and track record is more likely to have well-established incident management processes and procedures, and also a track record of responding promptly and effectively to issues and incidents.

Pricing and billing models

It is important to check the pricing and billing models of a cloud provider because it can have a significant impact on the **total cost of ownership** (**TCO**) of the cloud services being used. The pricing and billing models of a cloud provider can vary widely, and understanding them is important to ensure that the organization is getting the best value for their investment in cloud services. Here are a few reasons why checking the pricing and billing models of a cloud provider is important:

- **Cost**: The pricing and billing models of a cloud provider can have a significant impact on the overall cost of the cloud services being used. It is important to understand the pricing structure of the cloud provider to ensure that the organization is getting the best value for their investment in cloud services.
- **Flexibility**: The pricing and billing models of a cloud provider can impact the flexibility of the cloud services being used. Some providers offer pay-as-you-go models, while others require long-term commitments. Understanding the billing model can help organizations make informed decisions about how to best use cloud services based on their specific needs.
- **Transparency**: The pricing and billing models of a cloud provider can impact the transparency of the cost of cloud services. It is important to understand the billing model to ensure that the organization is not surprised by unexpected charges or fees.
- **Budgeting**: Understanding the pricing and billing models of a cloud provider is important for budgeting purposes. It allows organizations to forecast costs and plan for future expenses related to cloud services.

Uptime and reliability Service Level Agreement (SLAs)

It is important to check the uptime and reliability SLAs of a cloud provider because it can have a significant impact on the availability and performance of the organization's data and applications. The uptime and reliability SLAs of a cloud provider can assure the organization that the cloud services being used will be available and will perform at a level that meets their needs. Here are a few reasons why checking the uptime and reliability SLAs of a cloud provider is important:

- **Availability**: The uptime SLA of a cloud provider can assure the organization that the cloud services being used will be available when needed. It ensures that the provider is contractually obligated to meet a certain level of availability for the cloud services being used.
- **Performance**: The reliability SLA of a cloud provider can assure the organization that the cloud services being used will perform at a level that meets their needs. It ensures that the provider is contractually obligated to meet a certain level of performance for the cloud services being used.
- **Downtime**: The uptime SLA of a cloud provider can assure the organization that any downtime will be minimized and managed effectively. It ensures that the provider is contractually obligated to minimize downtime and provide prompt and effective incident management if downtime occurs.
- **SLA Credits**: The uptime and reliability SLAs of a cloud provider often include SLA credits, which can provide financial compensation to the organization if the provider fails to meet the SLA. This can help offset any costs associated with downtime or performance issues.

Security and compliance certifications

It is important to check the security and compliance certifications of a cloud provider because they can have significant impact on the security and compliance posture of the organization's data and applications. The security and compliance certifications of a cloud provider can assure the organization that the cloud services being used are secure and comply with applicable regulatory requirements and industry standards. Here are a few reasons why checking the security and compliance certifications of a cloud provider is important:

- **Security**: The security certifications of a cloud provider provide assure the organization that the cloud services being used are secure. These certifications often require the provider to adhere to strict security controls and best practices, ensuring that the organization's data and applications are protected against security threats.
- **Compliance**: The compliance certifications of a cloud provider can assure the organization that the cloud services being used comply with applicable regulatory requirements and industry standards. These certifications often require the provider to adhere to specific compliance frameworks, ensuring that the organization's data and applications are compliant with relevant laws and regulations.
- **Risk mitigation:** The security and compliance certifications of a cloud provider can help mitigate the risk of data breaches or other security incidents. By partnering with a provider having a strong security and

compliance posture, the organization can reduce the risk of data breaches, legal and regulatory violations, and other security incidents.

- **Customer trust**: Checking the security and compliance certifications of a cloud provider can help build trust between the organization and its customers. Customers are increasingly concerned about the security and privacy of their data, and partnering with a provider having strong security and compliance certifications can help build trust and confidence in the organization's ability to protect customer data.

Ability to meet your requirements and goals

It is important to ensure that a cloud provider can meet your requirements and goals because it can have a significant impact on the success of your cloud adoption strategy. The ability of a cloud provider to meet your requirements and goals will determine whether the cloud services being used are a good fit for your organization and whether they can help you achieve your business objectives. Here are a few reasons why it is important to ensure that a cloud provider can meet your requirements and goals:

- **Fit for purpose**: The cloud services being used should be fit for purpose and should meet the specific needs of your organization. This includes factors like scalability, performance, security, compliance, and cost-effectiveness. By partnering with a provider that can meet your specific requirements, you can ensure that the cloud services being used are a good fit for your organization.
- **Business objectives**: The cloud services being used should help you achieve your business objectives. This includes factors like improving operational efficiency, reducing costs, increasing agility, and enhancing customer experience. By partnering with a provider that can help you achieve your business objectives, you can ensure that the cloud services being used are aligned with your strategic goals.
- **Vendor lock-in**: The cloud services being used should provide flexibility to switch to another provider if needed. This includes factors like data portability, interoperability, and API compatibility. By partnering with a provider that can meet your requirements, you can reduce the risk of vendor lock-in and ensure that you have the flexibility to switch providers if needed.
- **ROI**: The cloud services being used should provide a positive **return on investment** (**ROI**) for your organization. This includes factors like cost-effectiveness, revenue generation, and risk mitigation. By partnering with a provider that can meet your requirements and goals, you can ensure that the cloud services being used provide a positive ROI for your organization.

Learnings

Here are a few learnings that we can share from our experience of doing cloud migration:

- Cloud migration is not a one-size-fits-all solution; it requires careful consideration of the specific needs and goals of the organization.
- Cloud migration requires a deep understanding of the existing infrastructure, applications, and data, and also their dependencies and relationships.
- Proper planning and coordination among different teams and stakeholders are critical for smooth and successful cloud migration.
- Testing and validating the migrated applications and data are essential to ensure that they function correctly in the new environment.
- Ensuring security and compliance is a critical factor in cloud migration, and organizations need to have appropriate measures in place to protect sensitive data.
- Refactoring or redesigning applications may be necessary to fully take advantage of the benefits of cloud computing.
- Proper monitoring and optimization of cloud resources are necessary to ensure that the cloud environment is cost-effective and performs optimally.
- Change management processes must be put in place to manage and mitigate risks associated with cloud migration.
- Communication and collaboration among different teams and stakeholders are essential for successful cloud migration.
- Cloud migration can be a lengthy process that requires significant resources, both in terms of time and budget.
- Cloud migration can lead to unexpected challenges, such as data loss or application downtime, that need to be managed proactively.
- The cloud environment is constantly evolving, and cloud migration requires ongoing monitoring and adaptation to ensure that it remains aligned with business objectives.
- Cloud migration can offer significant benefits, including scalability, flexibility, and cost savings, but it requires a commitment to continuous improvement and optimization.

- Moving to the cloud requires a shift in mindset from traditional infrastructure management to more agile and iterative approaches.

- Proper training and education are essential to ensure that teams are equipped with the necessary skills and knowledge to manage the cloud environment effectively.
- Cloud migration should be seen as a strategic business decision that requires careful consideration and planning, rather than just a technical solution to a problem.
- Cloud providers offer a range of services and tools that can help organizations improve their productivity, collaboration, and operational efficiency.
- Scalability is a key advantage of cloud computing, allowing organizations to rapidly expand or contract their computing resources as needed.
- Cloud computing can help organizations improve their disaster recovery and business continuity capabilities.
- Cloud providers offer a range of security features and capabilities, including encryption, multi-factor authentication, and access controls.
- Moving to the cloud can help organizations improve their data analytics and business intelligence capabilities.
- Cloud computing can help organizations improve their customer experience by providing faster and more responsive services.
- Cloud providers offer a range of backup and recovery options, including backup to the cloud and disaster recovery services.
- Cloud computing can help organizations improve their agility and speed to market by enabling rapid application development and deployment.
- Cloud providers offer a range of tools and services for data migration, making it easier to move data between different systems and environments.
- Cloud computing can help organizations improve their collaboration and communication capabilities by providing a centralized platform for sharing and collaborating on documents and data.
- Cloud providers offer a range of tools and services for automating common tasks and workflows, reducing the need for manual intervention.

Cloud computing can help organizations improve their cost management by providing greater transparency and control over IT spending.

Conclusion

Conclusion

Overall, retrofitting cloud services can help organizations leverage the benefits of cloud computing and improve the performance, scalability, and cost-effectiveness of

their systems. However, it requires careful planning and execution to ensure that the integration is successful, and the system continues to function reliably. The process of retrofitting cloud services can vary depending on the specific requirements of the system and the cloud services being integrated. In general, it involves identifying the areas of the system that can benefit from cloud services, such as storage, processing, and communication, and then selecting the appropriate cloud services that can address those needs.

In the next chapter, we will focus more on the designing aspect of solutioning and its relevance over coding.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

CHAPTER 7
Design First then Code

Introduction

In this chapter, we are going to look at how design thinking is very important these days. Though nothing new we always were taught as basics of software engineering that design constitute majority of effort in any project but sometimes it gets overvalued when we move from waterfall model into agile modeling where it is said that we should focus on test driven development to optimize cost and bring products quicker in market.

You will learn how to consume managed services in your design.

We'll see how to balance it, what are the right questions to ask, how important nonfunctional requirement is.

Structure

This chapter is divided into multiple parts starting from:

- Coding history
- Design thinking
- Approach to a practical problem
- Design approach

Objectives

The objective of this chapter is to help understand the relevance of design thinking over coding, we would be reading about coding history when it all started, followed by various aspects of design thinking using architecture principles and a practical design approach for a given problem.

Coding history

While reading this topic my foremost submission is not to discourage coding in any sense. But at the same time, we would encourage you to use word effective coding and not only coding.

Let's go a few decades back to when people used to code in mainframe systems, writing assembly languages, then COBOL and then finally came object-oriented programming. Our engineers write code to achieve business functionality.

But with years passing by we matured in programming languages. From assembly we moved to structured languages like COBOL, FORTRAN etc. and then from structured to 3rd generation languages like C++, C, C#, and so on.

What has reduced in all these years - Lines of Code, people contributed to make frameworks like .NET, Java JDK, that further speed up development time and time to market of business solutions developed in these programming languages.

Now, how cloud impacted our approach to coding business solutions, how managed services can be leveraged, how we should proceed in this cloud offered services world.

I would like to divert your attention as to how many services we have like Virtual Machine service, network service, storage service, Web App service, API service, media service, messaging service, database service, data service, caching service, search service.

If you look carefully these are complete programming blocks in themselves and you can directly consume in your development. For example, around a decade back people used to implement messaging by either implementing MSMQ or any database with a status field. They keep track of those records in the table and ultimately struggle with load and common issues like locking, dirty reads etc.

While implementing API Management we used to write Facade class or embed functionality in our API code itself.

How to start thinking when developing applications on cloud?

We would encourage you to read cloud design patterns for respective cloud service providers Azure, AWS, GCP etc. Generally, these three cloud providers have ample

amounts of managed services in areas of caching, identity, security, data, CDN and hosting.

Focus areas should be:

[Ref.**https://docs.microsoft.com/en-us/azure/architecture/patterns/**]

- Availability
- Data Management
- Design and implementation
- Management and monitoring
- Messaging
- Performance and scalability
- Resiliency
- Security

Let's start by taking a business problem and see how we should proceed with the problem-solving aspect, then designing and then coding.

In today's world you need more art in assembling the services rather than coding the services.

Design thinking

For effective design thinking let's move to our childhood and we would say become a curious child – this will help you first in overcoming your own experience bias, opens learning and most important good listening. Now, the first question that you will ask is "WHY" is doing this even if you know why someone is doing it.

By asking "Why" you always get either a strong technical requirement which is impacting their business directly or business problem directly. If you are not convinced with "Why" then you cannot do justice to the requirement. In my opinion you or your customer must get convinced on "Why" and whether it is really required before approaching further.

We have seen people have thought of complete project how they will develop, test, and implement whereas a complete SaaS based solution is available. We must ask what the need of re-invention is it addressing something innovative, or you are intentionally moving knowing a similar product exist in market, but you would like to proceed because that cannot be implemented due to certain restrictions which cannot be worked out.

Sometimes, problems mentioned below require different aspects to be reviewed:

A web application around 15 years back might not be doing well today hence think twice before investing further in that application and rather explore other options like Mobile app, tab Apps etc.

Server is choked - if server is choked should we step back, review our application architecture rather increase RAM and CPU of server and fix it temporarily. See whether scaling can be implemented (prefer horizontal scaling over vertical).

Infra is aging up: Should we make capex investments in procuring infrastructure or move towards Infrastructure as a service.

Too many security attacks are getting difficult to manage - Should we move to more managed runtime services like Azure Web Apps, Serverless technology where infrastructure is completely managed by cloud providers including infrastructure security, so that you can focus on application security.

Database is getting slow: Should we invest in increasing the size of the server, review partitioning strategy, how are we storing information, do we really need to store logs in the database.

Identity management: This is getting more critical these days. Please evaluate if your application can be modernized to adopt B2C or AD, or 3rd Party Identity Management solutions like Amazon, Google, Microsoft, Facebook, LinkedIn etc.

If you observe the above aspects none of them deals with coding but the environment itself and many things have changed over time.

What can be fixed by design cannot be fixed by coding. In broader terms we have seen the problem from following perspectives:

- Latest technology
- Better economics
- Adoption of best practices
- Review of existing design (most important)

Let's answer some of the basic questions that generally people are confused with:

Question: Are better designs complex to change, simple to change or do not require change?

If you review design and ask the above question you should pick up the middle one, there is no design which can remain good every time, what works best today may become obsolete or even out of market in coming years, it should not be complex at all. Remember the **keep it simple and stupid** (**KISS**) principle.

Why we called simple and stupid because simple to change and stupid means do not resist to change. The best architecture is the one which is simple to understand and simple to change.

Question: Am I genius or great architect if I design something simple and stupid?

A good architect is one who believes in change, does more experiments and is visionary, he must focus on basics and common sense, there is no use of complex design which cannot be maintained. So, a good architect is one who resonates common thinking, user behavior, easy adoption without much interaction with the system. Hence, your design should not have too many controls to manage, it should be really open to a change (read SOLID principles) and top of all it should not be complex.

If you would like to verify complexity, discuss it with your developers, take their feedback, it should not be difficult to implement because if it is difficult to implement remember it will be difficult to maintain also, and expertise is required to make a change.

Question: Should I leverage managed services, or consumption of managed service will make me less technical because I cannot play with thin controls of that service?

You must first ask do you really need to play with thin controls and if yes why you want to work on those controls:

Something not provided by a cloud service provider and your application needs those tweaks to work.

I am comfortable with it - because maybe it is not required but I feel it is good to have.

Unless you have a convincing need, and it is not provided by a cloud provider and it is impacting a nonfunctional requirement you must adopt managed services because it works on major principle **Do not repeat yourself** (**DRY**). Some of the managed services components are listed below:

Azure App Service, Azure functions, AWS Lambda, Azure SQL, Azure MySQL etc.

Question: If I use managed service can I change it just in case something better comes in future by another cloud provider?

This is a most important question and generally asked by many people that it locks with the platform, so, my answer to that is if you have created your own interface to interact with these services you need not worry and absolutely if your architecture is open for change you should be able to change managed service at any given point of time.

You should create interface in your application when dealing with managed service for example AWS S3, Azure Storage and GCP Storage - all provide storage but what your application needs for example create, update, and delete documents - then you should create interfaces and consume them in separate class - some design patterns are useful like Factory (avoid programming to concrete classes).

In the past we have created products that can work with multiple databases like SQL and Oracle, so that same way managed service is to be treated (which will avoid locking).

Question: What is the right way to think and as you said it must be simple how should we think simple?

We'll explain it to you with the following example:

If you are having musical instrument around 20 years back it used to have many musical controls like bass, frequency setting etc. how many times you change them (unless you are music director yourself) and let's see around 10 years back when we saw IPOD or musical pods coming in which do not have these controls at all and person can download and simply listen to music. Now see the time today, we have Alexa, google assistant, Cortana without these settings and we are doing good, even mobile apps like wynk, gaana, saavn do not have any settings plus it does not have any requirement for you to even download because we have better network bands like 3G, 4G, 5G (coming up) and better internet connectivity.

So, what we learned, for only 1 or 2 % requirement a complete design was made, marketed but never used. What musical device modernization did - it simplified because lesser controls mean lesser management and more adoption.

We should be explorer, experimental and our thinking ability should start from the user rather than a solution provider. We must think as a user what do I need and very important thing we should not think about corner cases (means features or changes that influence only 5-10% of your users) because that is where complexity finds its door into your thinking.

We must address common sense - this is not so simple because sometimes our sophistication introduces complexity which reduces feature adoption and impacts returns on investment. For example, which one would you prefer:

I have made the most complex routine in the system which does so many things.

Or

I have tried to develop a routine simple to understand and easy to maintain and update and it provides features easy to use by business users

it provides features easy to use by business users.

The second approach is called a simple approach whereas the first one may give you quick wins but in the long run it will bring additional complexity and will be

challenged. Whereas in the second approach you will maintain low cost and even if it is not required you can choose to remove some of the features or introduce new ones.

In my experience I overcame this habit to make everything perfect on day 1 and aiming to fulfill most of the complex requirements as well, rather I thought about how much impact I can make by building this solution and how will I grow this impact based on market conditions, customer feedback. If you wish you can read more about Test Driven Development (one of the key aspects of Agile framework).

Approach to a practical problem

Now let's hit a real time problem and see how we bring our design thinking hat and then develop, we will also, ensure how to ask questions which are not answered because feedback influences design.

Problem: A company in India owns over 400 advertisement boards across the city and currently running this system from a home-grown client server application where a person must reach or call the company for booking of advertisement, then submit the clip or advertisement which is reviewed, edited as per specifications of board, and then displayed on boards. This system failed 4 times in the last 5 years for a brief period, also, customer satisfaction is going down because of lack of transparency, sometimes boards become non-functional and customers despite spending on subscription remain unsatisfied.

Classification into business goals:

When we read business problems, we should see what users want. If you expert in that domain, it becomes more useful because you must have observe similar problems from similar types of customer and that is where consulting helps if you have business consultants of that domain better include them in your discussion because they can guide on similar ask or even help in classifying the ask in terms of priority and may even provide a strategic vision on returns on investments.

We did the workshop with customers to understand his existing business problems and discussed briefly how similar companies have done modernization of their systems and how upgrading their process can bring them up in the market and increase the reach of the system.

Requirements:

- Develop a mobile app where users can buy subscription basis on location of display, prime time. They should be able to upload the video to see the

preview of any specific board of any location.

- The company is looking for zero downtime for such applications (SLA - 99% availability).

- Companies want to send display reports to improve transparency and in case of any failures they want to give back credits to retain customer satisfaction.
- The company wants to improve the uptime of display boards by upgrading infrastructure, predictive maintenance of boards.
- Companies would like to sell slots at economical rates to reach small size subscribers.

The above are very high-level requirements some are business, and some are technical which comes out of design thinking and after due diligence that investments were thought of - their results were testified by some other reference who have taken this approach.

Design approach

What do the above requirements show - It only represents functional requirements? Now if we straight away jump in to designing at this stage and engage at coding level, we will miss a very significant gap which every architect should ask or evaluate.

We should never miss Non-Functional Requirement as it is the most important pillar of designing.

If we see above, we must ask following questions:

- How many users will be accessing this application?
- What is the average latency acceptable by business?
- What are availability requirements - need to know **Recovery Point Objective** (**RPO**) and **Recovery Time Objective** (**RTO**) for this workload?
- Is there any geographical constraint for this application?
- How is identity management done for these applications?
- Is there any specific compliance or regulation to be followed?

You may receive the following replies for the above requirements?

- We will launch this service for approximately 1000 users in the first 6 months growing up to 50000 in the next 2 years.
- Our subscribers should get responses in maximum 3-4 seconds and page load should happen within 3-4 seconds. If it crosses 10 seconds, we must address it immediately.
- The RPO for this application is 30 mins - which means in case of any disaster we must get the copy of logs and database maximum 30 mins back. Similarly,

in case of disaster we must get the system back in 4 hrs. However, during downtime, the customer should be able to see static content and wherever the transaction is happening it should display a maintenance page.

- Since it is advertisement and Indian customer data, it must remain within India only.
- The identity management is to be done using Facebook, Google, Microsoft, Mobile OTP.
- The payment gateway integrated with this application should be PII compliant and the rest of the entire system should be ISO 27001 compliant.

With the above requirements coming in we got a good grasp as to what is required on Day 1 and what is required from my system in the next 2 years - means we captured nonfunctional requirements for next 2 years.

So, what will happen after 2 years - the nonfunctional requirements and functional requirements both changes with time, in this case for example he is forecasting 50000 users, but it can go very well above 1 million or may even close or reduce to 5000 also.

How will it impact design then and how should it be handled?

We must design a system which is scalable in nature, with higher availability. So, let me explain that part in detail on how to address Scalability and availability in any solution:

Scaling is of 2 types - Vertical where we increase RAM, CPU, and other resources of existing systems, it may bring downtime whenever we try to upgrade the resources and then Horizontal scaling where a similar instance joins the existing cluster and shares the load.

Vertical scaling is also known as Scale Up/Down whereas Horizontal scaling is known as Scaling Out / In.

We must opt for horizontal scaling because of its higher availability. I am listing down some of the famous performance and scalability patterns:

Static Content Hosting: Before cloud services came into picture we used to deploy static content like scripts, images etc. in web servers which results in computer consumption and impacting network egress from these servers. Then came **content delivery network** (**CDN**) or caching which kept these resources near to users and offloaded web servers for those requests. Now, with cloud services we should directly host static content in Cloud storage like Azure Storage, AWS S3 etc. You can enable CDN on them and configure only transactional components in your web interfaces.

Cache - Aside - Uses cache to keep data instead of data store, it uses the pattern where read is made from cache first and if not, available data is fetched from data store, cache is updated and returned. Similarly, while updating any information which is cached is removed first from cache and then updated in the data store.

Sharding: Keeping data in partitions, a very useful pattern which must be used keeping scale in mind. One should always see the data store from Query first approach and design partition key. For example, in this problem, we can have Month and City as partitions for tables containing advertisement content, because we know advertisement get expired after a certain time, if we keep data month wise, we can remove partitions without impacting performance, similarly if it is partitioned by City the data can be fetched faster and localized.

Queue Based Load Leveling: Use Queuing solution where a huge number of tasks that are created are pushed as messages inside Queue and scale workers to consume those messages. In this problem it can be used when accepting requests from customers about their clips of advertisement, the request can be queued and then event hubs patterns can be used for transcoding these clips in various formats. Similarly, when they will be pushed to devices for display, we can implement Queuing solution and display boards can read queues whenever they are empty. This way you can take any board offline very easily and requests can be buffered.

Now, let's see how we address our first question: "Approximately we will launch this service for 1000 users in the first 6 months growing up to 50000 in the next 2 years."

In our design we must have scaling implemented (preferably horizontally) so that when user numbers increases from 1000 to 50,000 or even beyond we can scale the system by putting more web or app servers.

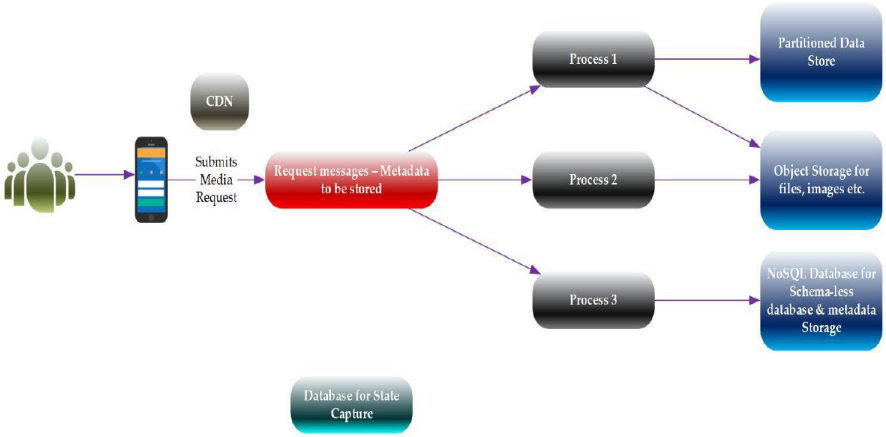


***Figure 7.1**: Solution Design using managed services*

In the figure above (*Figure 7.1*) we have shown how mobile apps can leverage CDN for better content management and remain responsive.

See we designed from a very high level that solution requires scaling at various levels, API layer, then backend. If we see the above picture we have solved for the backend where we have put conceptually as queuing solution, then various processes are shown that implement functional requirements but as users increase the system will respond. Some of the key things we should know:

Avoid update of data - Sounds strange but let me tell you what happens when we do update in conventional RDBMS databases, lock is acquired (to avoid dirty read) on the table, update is performed and then commit happens. In today's world when we are reaching millions of users, updating the table is not a good idea, so what needs to be done. We must implement a write ahead pattern that means for every operation add a new record and use low-cost storage because we want state capture and not a transaction. So, in the above case when request is queued, we should do following:

Enter the record in Azure table with flag as I –Inserted.

Add another record in Azure table with flag as P - In process when Process 1 or Process 2 picks it up for processing.

Add another record in Azure table if there is a timeout or exception with Flag as E - Exception.

Add another record in Azure table with flag as C - Completed.

By doing the above implementation we can implement notification based on flags, monitoring based on these entries. We should not worry about cost because this data can be archived once the message is marked completed.

For ask "Our subscribers should get responses in maximum 3-4 seconds and page load should happen within 3-4 seconds. If it crosses 10 seconds, we must address it immediately."

We must host static content in the storage account of cloud service and, see we must enable geo redundancy along with enablement of **content delivery network** (**CDN**).

The RPO for this application is 30 mins - which means in case of any disaster we must get the copy of logs and database maximum 30 mins back. Similarly, in case of disaster we must get the system back in 4 hrs. However, during downtime, the customer should be able to see static content and wherever the transaction is happening it should display a maintenance page.

Above is a very important design requirement and influences design to a great extent, it states the requirement for business availability, and we should always ask this clearly and business must be aware of it. Here it states RPO of 30 mins that means at any given point of time we must be able to restore back to 30mins old copy of information whether databases, logs, messages etc.

Also, it states RTO of 4 hrs means we should be able to restore back a complete running system in 4 hrs. Also, we must deal with availability and disaster recovery separately in any kind of solution, many requirements say RTO and RPO are equal to zero when we have availability closer to 100%. This is partially correct and even all SLAs calculations in this world do not account for natural calamities, planned maintenance time which contributes to downtime. Hence the world disaster came up and addresses both natural and planned maintenance scenarios.

Some of the incidents like floods, storms, earthquakes can impact operations in data centers and may bring prolonged outages, so we need to address disaster recovery. Also, the alternate location of the data center should be far enough so that we can mitigate natural disaster scenarios.

For example, in Azure you can choose Central India (Near Pune, Maharashtra) as primary location and South India as disaster region (Near Chennai). This way you will improve the availability of your solution for your business.

One good thing I like about Azure cloud is it is designed from complete enterprise scenarios and addresses most of the best practices of enterprise architecture.

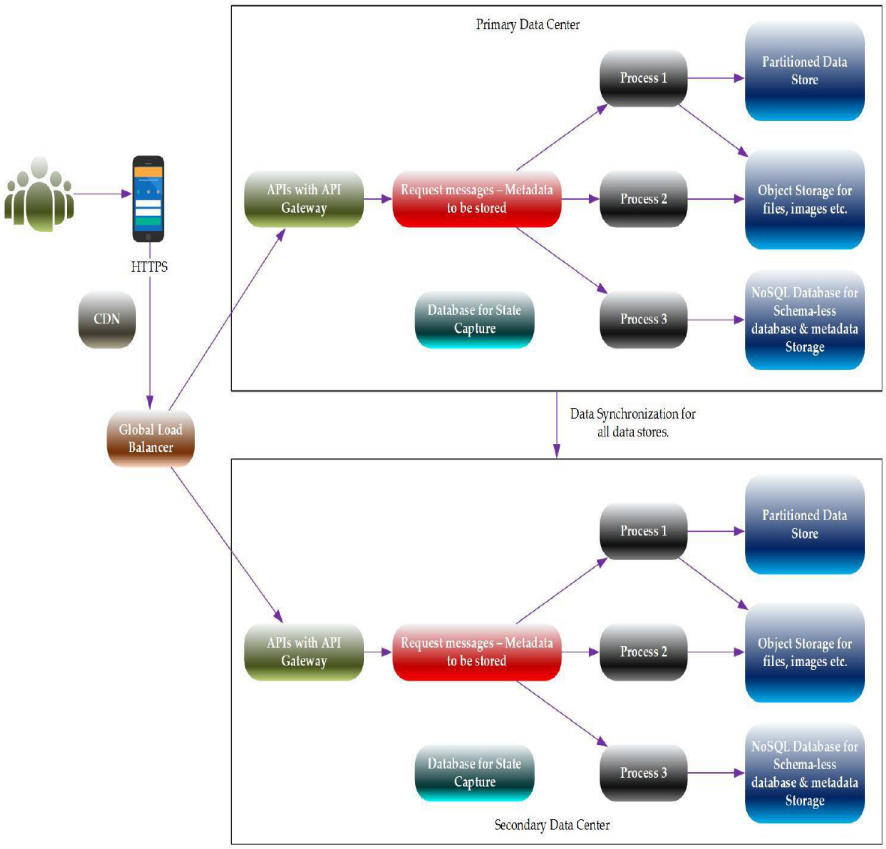


Figure 7.2: Disaster Recovery Scenario

In the above design (*Figure* 7.2) we have shown low level architecture of the mobile app or web app that should work in Disaster recovery scenario, and we have also used most of the managed service provided by cloud providers.

Here we have used Azure Front Door which acts both as a web application firewall and global DNS router. So, in case the primary region goes down the request will be transmitted to the secondary region.

If you wish to read more about these implementations read Stamp Pattern or Geodes pattern.

Now, we can design a system which can be scaled at various levels, highly available, implement queuing solution in case load increases, suggested to implement write ahead pattern.

Since it is advertisement and Indian customer data, it must remain within India only.

When such requirements are asked, we must be careful that all components used in your system should take care of data and ensure that storage remains in that geography. Also, not every data has requirements for storage in a particular geography, so we can step forward and classify information that needs to be inside a geographic location.

In above design we should pick up data center that are in same geography, must be good distance apart (as per MEITY - Ministry of Electronics and Information and Technology the guidelines given is 100 kms apart) but it is advisable to be minimum 500kms apart just to mitigate any natural disaster scenario like earthquake, storm etc.

In Azure for India region one can choose data centers as Central India (Pune) and South India (Chennai) as primary and secondary data centers.

The identity management is to be done using Facebook, Google, Microsoft, Mobile OTP.

For this we should implement Oauth identity management in our solution rather than developing complete identity management ourselves because it is one of the most focused areas in any system. Having dependencies on famous identity management helps prevent most of the attacks which is very difficult if you implement your own identity system.

Some guidelines if you want to develop your own identity management:

Identity Management systems should have separate data stores and should not be kept on business servers. This way business servers can be protected from malicious attacks.

We should have proper authentication and authorization developed in our application. Authentication deals only with successful login whereas authorization deals with the kind of access that user or process needs to be given.

Azure Active Directory B2C service gives many features in this regard where you can implement global identity management for your application, with existing identity management plugins available.

The above identity management providers check for various types of checks, they have business intelligence built in which can detect and invoke 2 step authentication (must these days) or prevent geographical attacks (it checks if someone is not login from different countries, devices etc.)

The payment gateway integrated with this application should be PII compliant and the rest of the entire system should be ISO 27001 compliant.

Most of the managed services given by cloud providers are ISO 27001 certified, but as an architect we should check SOC compliance reports and specifically check for that service. If it is not present or has that certification, we must re-visit our architecture and move to Infrastructure as a service and implement the solution on Cloud IaaS for that component.

The only trade off that will happen is you need to manage those IaaS components for security and patches.

Security

The most important part security for a given solution must be thought off at the design level and not in the last. Something I have observed is that people think about security when they deploy solutions in production. This is a wrong practice and must be avoided. So, how to keep security in mind when working in various environments.

Development: You should consider endpoint security even in development to safeguard your services from external attacks. Like you should configure network security rules in a storage account, databases that can only be accessed from a designated development environment. We must have Role based access control on all cloud services where only contributor or read access is given.

QA: This must simulate your production environment in real sense; besides all development security, this environment must have all identity security, infra/service security implemented.

Production: All required security.

Now let's understand why we are giving so much stretch on security while designing a solution and what benefits it gives at design level and how to think about it.

Question: How should I think about security in the cloud because everything is public?

In the cloud we do not say security, but it is better to say endpoint security and endpoint security of all endpoints that our services are exposing. We need to see if those endpoints are accessible to whom it is required and should not have anonymous access. Once you have this understanding you will automatically prepare a baseline document for all components used and what security is designed for each environment - Dev, QA, UAT and Production.

Question: What happens if we ignore the Dev and QA environment and only consider UAT and Production?

Azure or any other cloud does not understand your environment and for what purpose you are using it, to prevent abuse in your dev and QA environment we must have basic security controls implemented. Some of the mandatory controls are the following:

Role based access controls: It means a specific user can only do a specific operation in that environment, for example developer is only authorized to deploy the code but he is not allowed to change the configuration or delete the service itself.

Endpoint Security: We must secure all public endpoints, always prefer network security because it prevents from network point of view and then desired access or identity driven access. For example, if you are using Azure SQL database it has identity level access control leverage that feature of that service.

Conclusion

You must have observed that as an architect what is required from you when you are assigned a business problem, what is the role of designing at an early stage and how nonfunctional requirements influence design.

Remember, what is developed and tested can work in production and not assumptions. So, if something is not tested thoroughly in earlier stages it is a risky proposition to test those features in production.

How we learned from business problems, functional and nonfunctional requirements are extracted, what is the relevance of non-functional requirements and how it can impact design.

The vision of non-functional requirements is also mandatory, so that you can design accordingly rather changing a complete implementation later.

Make use of cloud design patterns in designing service level components.

In my next chapter we shall be reading how the new DevOps team should be created – a mix of Infra and application team, useful for organization and tech leads how they should form new team.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 8

Infra Team and Apps Team Becomes DevOps Team

"With cloud technologies how are we collaborating with different teams like Infra, security and development, in the new world when we can code infrastructure it is a time to change our way of working and adapt a new normal of including Infra and security teams in your agile teams."

Introduction

In this chapter, I am going to tell you what DevOps is, what is the significance of DevOps, building blocks of DevOps team, role of Internal IT team and Application Team in DevOps and how should we collaborate in new world.

Agile process needs to include the efforts required Internal IT Team because of quicker availability of resources and changes required.

Structure

This chapter will start with an overview of DevOps, and how the existing process in most organizations with broken Devops work, challenges were discussed and then a complete strategy for reform of various teams is explained.

By the end of this chapter, you will be able to visualize the changes needed in your process to implement a successful Devops.

Objectives

The objective of this chapter to challenge our thoughts to unlearn traditional structures of IT Teams where IT was shared service, then application teams and Infosec teams. In this chapter we are going to read why there is a need to collaboration between these teams, role of new IT Teams in an organization, increased responsibility of application development team, inclusion of security team upfront in development cycle.

DevOps

By now many of us know the straight definition DevOps is unification of people, process and technology and they all work together to achieve business goals faster and with enhanced quality.

Let's deep dive a bit into this and understand why the entire industry is talking about it – is it a new thing and what is it importance.

IT Team – A conventional look

A typical IT Department of any big enterprise looks something shown below that typically managed either an OnPrem data center or Co-located data center.

Figure 8.1: IT Organization

System operations: Team responsible for maintaining and provisioning of systems.

Security & Monitoring Team: Team responsible for observing security, patching, and securing systems.

Asset Management Team: Team managing organization assets and compliance of

Asset Management Team: Team managing organization assets and compliance of resources like physical devices, laptops, servers, racks etc.

Identity Management Team: Team managing Identity of computers, servers and people working inside organization.

Network Team: This team is responsible for managing and updating Network resources.

In big organizations where cloud adoption is already started, they have opened a new operations team called "Cloud Team" which looks after cloud related operations or where entire data center is migrated, they called this IT Team as Cloud Operations Team because everything is in cloud.

Now the following flowchart shows typical interaction with IT Team:

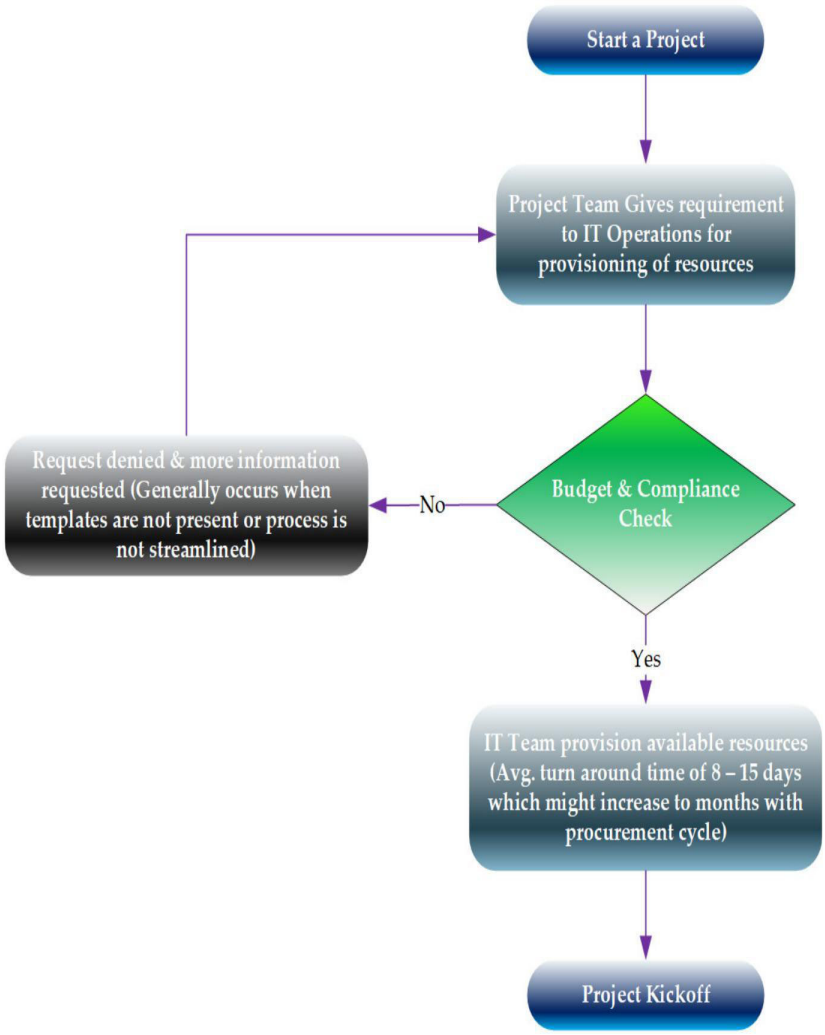


Figure 8.2: IT and Project Team interaction

Figure 8.2: IT with Project Team Interaction

Challenges with approach shown in *Figure 8.2* above:

- Clearly both departments Project and IT are working behind very closed boundaries and priorities are not known to each other.
- Many times, projects were derailed because teams failed to justify architecture, security requirements, and in some cases budgets as well.

- Matured companies have drafted processes by creating templates to avoid to and fro of information and work under internal SLAs.
- Change request in such a model will impact project timelines, and IT department also, won't be able to prioritize hence its performance and reliability may be questioned.

In summary unless both departments unify their operations and approach, we cannot mitigate above challenges hence if you see a department which can smoothly work together under a single orchestrated process with defined roles all the way from IT Operations to Project Management will be called DevOps.

DevOps – How to reform?

In this section I am going to discuss practical approaches on how to overcome above challenges and classifying them in to five categories and will detail them so, that if you are also, trying to mature your DevOps in your organization you can start working on this approach and remember results won't be overnight it may take few quarters to year to reach a maturity level. But the journey is important.

1. Classification of Cloud Adoption – Hybrid or Complete
2. Skilling of resources – Certifications for specific departments (Agile and Cloud for all)
3. Architecture board – Responsibility in DevOps process
4. Cloud operations team – New Roles and Responsibilities
5. Program management – Updated responsibility

Classification of Cloud Adoption – Hybrid or Complete

Depending on the size of your organization, priorities, regulation we have observed either organization moves completely to Cloud (Azure, AWS, Google etc.) or partially move operations to cloud after due diligence of workloads.

As of today, almost 100% of Fortune 500 organizations have invested in Cloud Technology and some organizations in various domains like healthcare, manufacturing, oil, retail, ecommerce has moved their entire operations to cloud.

Startups and their investors are now preferring direct adoption to cloud to avoid Capex investments.

Companies that have moved entire operations to cloud or booting on cloud have advantage to adopt DevOps faster than Hybrid organization.

Hybrid Organization must start their journey by creating a Cloud Team or Cloud Operations Team because existing OnPrem departments will be part of DevOps team as well.

See below *Figure 8.3* Hybrid IT Operations Team - the new structure of Hybrid Organization with IT Operations Team:

***Figure 8.3**: Hybrid IT Operations Team*

To begin on transformation journey we cannot expect unification of IT Operations and Business teams, but a proper delegation of policies can sort this to a great extent. In *figure 8.3* the cloud operations team would be responsible for setting up of Landing zones (a set of policies, network design, security benchmarking) – just like when a new township or society is launched it is supplied with water, electricity etc. and individual is responsible to use it under threshold values learned over time.

The modernized organization should look something like below – *Figure 8.4* who is running majority of cloud operations from cloud:

***Figure 8.4:** Hybrid IT Operations Team*

We'll see more details around transformation in next sections.

Skilling of resources –Agile and Cloud for all

Learning should never stop even if someone ages, I like one very good statement "I really wish not to know it all but learn it all" and it goes with everything we do in our life. I am also a firm believer that as an organization we must "Train to retain" and grow rather fearing attrition if someone learns.

In the new world each individual must learn 2 important things: Agile and Cloud (could be Azure, AWS, GCP) – naming them because they are very matured and widely accepted in any organization.

I am listing below some most important certifications you should aim for agile:

- Certified Scrum Developer – Must for all developer and admin roles
- Certified Scrum Professional
- Safe4 Agilist (Advance)

If everyone understands Agile, they can better understand their participation in any program and project and how the orchestration of tasks works. They can participate in Scrum calls, measure their productivity, and will collaborate better with business teams.

The following are some important certifications from Cloud perspective:

Azure

- Azure Fundamentals (AZ 900) – For everyone
- Azure Developer Associate – All developers and admins
- Azure Administrator Associate – Must for Admin roles
- Azure Solution Architect Expert – Must for existing Lead roles and architects.

AWS

- AWS Cloud Practitioner – For everyone
- AWS Associate – All Developers and Admins
- AWS Solution Architect / AWS Devops Engineer – Must for developers and existing Lead and architect roles

GCP (Google Cloud)

- Cloud Engineer – Must for everyone
- Cloud Developer/Cloud Engineer – Devops/Security/Network – Must for developers and admins
- Cloud Architect – Must for Lead roles and existing architects

You must drive above as a managed learning program in your organization (any size), also, encourage to learn more than one cloud so, that you can design as per business requirements.

Once the team is trained as per the above skill matrix it will be convenient for you to form a Cloud Operations team and I shall be explaining inception of Devops in my next section.

Architecture Board – Responsibility in DevOps process

This is the team of all existing leads/architects who are responsible for defining policies, design, blueprints keeping organization goals in mind and, how cloud will be adopted in an organization.

To help them every cloud has documented Cloud Adoption Framework which should be completely known to architecture board members and, re-define their roles and responsibility as per framework for smooth adoption.

The following are the key objectives which should be set by the Architecture board:

- **Landing Zone Design**: A design which demonstrates network and security posture of cloud in an organization and how different teams will construct and perform in that landing zone.
- **Baseline policies**: Here different leads should work together to put down organization policies in form of controls like Virtual Machine Image management policies, patching policies, network policies, etc. here they must refer their existing policies – To help setting up fresh there are many templates like Azure Blueprints, AWS Guardrail etc.
- **Sharing and update artifacts**: This is most important where these architects regularly update baseline based on new services, updates policies and share documentation with project teams. It will mature their existing templates for capturing requirement of provisioning on Cloud, where each team will have to adhere with organization policies and rules.
- **Roles for Business Team**: In cloud we must give additional controls to business teams because they will use Infra as a service (not a physical asset anymore) or managed services hence clear roles should be defined for each service like Virtual Machine Operator (only able to start/stop VM from portal), Storage Admin.
- **Project Architecture Review**: They should be responsible for reviewing project architecture when it comes to provisioning or design, review gaps, help imbibing best practices. This may lead to more time in reviews in beginning but with time it will reduce because business teams will learn controls like Role base access controls, endpoint security, adherence with benchmarking documents. The team can develop automation to validate controls. For example, if the environment is production and storage account is having control as open to all – should raise warning etc.

- **Governance Model Setup**: This team should help in setting up governance model, for example how will system operations team, monitoring team will raise appropriate incidents and how the gaps will be addressed. This model should always ensure adherence to organizational policies. Regular reviews of policies and **root cause analysis** (**RCAs**) of incidents should be carried out and post that benchmarking, policies need to be updated and communicated.

Cloud Operations Team – New Roles and Responsibilities

As we have our existing roles in IT Operations like Architects, System Operations, Network, Security, monitoring, all these roles will have similar roles in cloud as well so, let's understand what the change is and how they should transform.

Security and Compliance Team must work on creating or redefining benchmarking documents like "Accidental Deletion Policy" for cloud this will enforce disaster recovery design for business and mission critical workloads, ensure DR drills, and own organization security benchmark for all cloud services apart from their existing OnPrem controls.

The Network Team must study overall requirements in terms of the data center location that Cloud provider owns in a particular region. Azure and AWS have more than 100 data centers in almost all major regions. In India specifically Azure has 3 regions West India (Mumbai), Central India (Pune) and South India (Chennai), whereas both Google and AWS has data center in Mumbai.

They should start provisioning the network resources in cloud like network connectivity (Site to Site, MPLS or P2S) basis on requirement, design network in Hub and spoke topology where Hub contains Transit connectivity gateways, firewalls, along with Management Virtual Network (or Shared Services) which contains all resources like Active Directory, Patching servers etc. They must own routing in the same way as On-Premises. Also, they must know that cloud gives option of Disaster recovery, and this should never be ignored when you are setting up landing zone in a particular region. Let me explain, all cloud provides data center in a region and implicitly contains controls for local data center failure like keeping 3 copies of storage, and so on. but each Cloud provider also, have controls to mitigate data center failure see the details below:

Azure among other cloud providers has best disaster management strategy in their design i.e., you will observe Azure always create multiple data center in a particular Geography – For example in India they have it in Central India (Pune), West India

(Mumbai) and South India (Chennai) where Central India DC and South India DC are paired for data center redundancy via Azure backbone and one can use Azure Site recovery to design DR solutions. AWS on other hand provides High availability

by 2 data centers nearby locations.

I would also like to stress one point in this article High availability and disaster recovery can be seen as if my RTO and RPO reaches zero it automatically becomes Highly available. But if we see this practically one could challenge the above assumption. Almost all data centers (Azure, AWS, GCP) have been certified for multiple network connections, multiple power input but what happens in case of earthquake, zonal failures like riots in that city where data center exists. We need another data center at least a few hundred kilometers apart so, that natural disaster, riots should not impact both data centers simultaneously.

With this even AWS related workloads (in absence of distant data center) can provision resources along with Azure (as Azure provides Azure site recovery which can be integrated with AWS) data center. For example, AWS Mumbai can have disaster recovery configured in Azure South India.

Monitoring Team: They must work on configuring Alerts for organization policies, threshold controls, authentication controls, authorization controls, raise appropriate alerts take actions, automate towards integration with their existing ITSM tool (like service now etc.) using Rest API or out of box integration provided by cloud provider.

Extend their existing monitoring to include logs from cloud resources. Each Cloud provider has their own monitoring tools and that can even be integrated in Hybrid implementations. For example, Azure Sentinel is a good SIEM tool covering almost all types of log collection i.e., from security, virtual machines, SaaS based services like Office 365 etc., on AWS side we have CloudTrail that can be implemented for monitoring of resources.

I must encourage this decision should be taken in a long-term view, for example if you feel most workloads are going to learn on cloud it is better to implement monitoring tools provided by that cloud provider or you can do vice versa as well for example you're existing SIEM solution may be Qradar or ArcSight can also, fetch events from cloud resources like virtual machine, storage, network etc.

Automation Team: This team should work on automation, PowerShell scripting, integration of resources, different systems in an organization to create a unified view and operations set up. They must be writing templates which can be executed using Azure DevOps pipeline to push things in Azure, AWS, GCP etc. Many times, people call this team as DevOps team because they are coding, but it should not be the case the entire team should be called DevOps team.

Some of the key components that need to be automated:

- **Virtual machine image management**: Your organization must prevent usage

- **Virtual machine image management**: Your organization must prevent usage of marketplace images in environment because it may not be hardened as per your organization policies. Hence creation of images and storing it in private marketplace should be prioritized.

- Create templates for capturing requirements, terraform should be preferred because they can achieve cloud agnostics approach of execution.
- Create automated workflows for provisioning most common workloads like policy changes, creation of new virtual networks, provisioning of shared services, hardening of resources to be automated.
- Creation of Role based access control templates, customized roles, their mapping with their existing Identity management system should be done. There are many customized roles each cloud provider already has which can be leveraged and reviewed. It should be done for each resource so that adoption eases by business teams.
- Cost Management/ Budget Management policies should be provisioned for each project team using Tags so that consumption happens within budgets and avoid over expense.

Post automation or even before automation customized roles should be leveraged by project teams to control resources access like virtual machine contributor, database operators etc.

Program management - Updated responsibility

Program management should extend their responsibility not only to cater functional and nonfunctional requirements but also, visualize how different teams we discussed above participate in project development and execution.

It must create tasks for each one we discussed above like Architecture Board, Cloud Operations Team, Automation Team, and then their own project core development team. The shared team like Architecture Board, Cloud Operations Team and Automation Team responsibility is to provide you a landscape or plot to work which may include automation of services for that specific project. Let me explain this below by a real time example:

"A project team came up with requirement to provision e-commerce site for their customer and would like to do development in an organization"

Though it is just a statement I am assuming the requirements are present to the team in terms of functional and nonfunctional aspects. The project team must first create

in terms of functional and nonfunctional aspects. The project team must first create an architecture defining all layers, services, endpoint security.

Also, keeping in mind what is required in development, UAT and production. Once architecture is created it must be reviewed by Architecture Board and then tasks

need to be created for Cloud Operations team for creation of relevant controls so, that project team can work and they should also, be able to do scripting of services like virtual machines, managed services etc. We must work in a way that gives flexibility for project team to do complete automation from project scope perspective and deliver while at the same time Cloud Operations and Monitoring ensures project team should work under defined controls, guardrails or policies defined for them.

Conclusion

In this chapter we have learned how departments like IT and Business Units must change the way of their working where IT teams focus more on organization related policies, guidelines and preparing good execution environment for different project teams.

Also, DevOps is not only limited to your project team but also, entire organization and 2 teams specifically Architecture Board and Program Management will play an important role in setting up a successful Devops.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 9

Traits of Being a Good Cloud Solution Architect

Introduction

In this chapter, I am going to tell how to become a good (not great) solution architect. I intentionally chose to eliminate "great" because this role is a learning role and do not want to subdue it under the hood of greatness but regularly improving it. You must have seen on various job descriptions what organization wants in their adoption of cloud. What a typical day of Cloud solution architect looks like? What all you can do to improve your skills and become a good solution architect.

Structure

This chapter is divided into multiple parts starting from:

- Role definition of Cloud Solution Architect

- Role definition of Cloud Solution Architect
- Understanding business problems
- Solution design
- Improve design with newer services
- Define process and facilitate adoption

Objectives

The objective of this chapter is to help application architects, infrastructure architects or any individual in apprising about Cloud solutions architect role. The growth mindset, problem solving techniques. After going through this chapter you can easily craft your path to become a successful cloud solution architect.

Role definition of Cloud Solution Architect (CSA)

A cloud solution architect is a person who helps in designing solutions on cloud and regularly improves them on a continuum basis. He must have the ability to understand business problems and should be able to guide how the problem can be solved.

This role has huge responsibility in modernization of business solutions, systems, and overall organization so, the role demands good communication skills (both verbal and email) to interact with various stakeholders. It also emphasizes good listening skills.

A typical day of Cloud solution architect starts with customer/colleague/partner reaching out to you for discussion, some knowledge session, or even production issues where some service/solution is not working up to benchmark, and so on. You may even be working on a few assignments together. This is more of a consulting role and my suggestion would be to keep sharpening your toolkit by being open to learning from different sources or from your own mistakes and learning.

Evangelism plays a key role in adoption and sometimes cloud solution architects are expected to do knowledge-based sessions for some cloud services, specific use case discussion. My only guidance would be preparing well before delivery and making it an ever-lasting impact. Because the more you evangelize, the more it would be easier for other people to implement and understand cloud use cases and services. If possible, please create a small demo to showcase. The topmost I got to learn from

if possible, please create a small demo to showcase. The topmost I got to learn from one of the community sessions – The architect tools are 2 markers (Black & Red) and duster, and he must always carry them like a doctor carries Statoscope. It is always good to do whiteboarding or even use OneNote, Microsoft Whiteboard, Jam board etc. for online sessions.

In case of production issue when some service is not working certainly falls under the scope of Support engineer and less of **Cloud Solution Architect** (**CSA**) – But CSA has lot to contribute, he can provide alternatives, changes in architecture or rethink and redesign to solve the problem in long term. Remember Cloud Design patterns will help you in solving most of them from design perspective.

Area of specialization plays an important role, so, a basic knowledge of the following areas is very important:

a) **Cloud Design Patterns**: Availability, Resiliency, Messaging, Performance and Security related – Read Azure Cloud Design Patterns.

b) **Infrastructure Background**: Virtual Networks, Virtual Machines, CPU, RAM, Storage – Disks

c) **Application Background**: Mobile Apps, Java/Nodejs/.NET/Angular, IIS, Apache, Serverless, Queues, SOLID Principles, MVC, Agile, Devops.

d) **Data Engineering**: Database, Datawarehouse, ETL, NoSQL Databases, Reporting Tools like Tableau, PowerBI, Data Governance.

e) **Data Analysis**: Machine Learning, Data Lineage, Data Lake Design.

f) **IoT**: IoT Hub/EventHub, Device Management, AMQP, MQTT Protocols

g) **Security**: Endpoint Security, Firewalls, Web Application Firewall, Compliances – PCI DSS, CIS, HIPPA, ISO Standards, Antimalware

h) Workloads (Good to have) – ERP Packages – Oracle EAP, SAP, CRM Dynamics

Please do not get scared with the list above, if you do not have knowledge in any area above, please start with basics only for example: What is IoT, different protocols and most important in which business use cases this is used (most important).

Also, in Database you must have observed I segregated Data Engineering with Data Analysis because both are way different, Data Engineering deals with movement and management of Data, governance whereas Data Analysis deals with fetching meaningful information from existing set of information.

So, as cloud solution architect, you must know the basics of each area and the use cases they can be used for example, you must be able to identify the use case and most appropriate solution around it.

In case you are not able to solve that specific area, there is no harm in admitting

In case you are not able to solve that specific area, there is no harm in admitting that with any stake holder and help them connect with other **Subject Matter Expert** (**SME**) for deep dive discussions, or Architect who has created solution in that area.

Understanding business problems

In these changing times architecture and business cannot work in isolation and it is never the case earlier as well. For an architect it is very important to understand business needs and relevance of technology in that solution. Couple of instances I state below where relevance of right technology or approach can be challenged:

"A customer asked that he wants to read facial expression of people standing in queues by studying the video recordings and basis on their details registered or available on public information they would like to send targeted offers"

If you address this purely from technology point of view it's really a good use case, you will use cognitive technologies to retrieve information and then attach it to some metadata and then send, but if you hold yourself for 2 mins think following questions:

a) What could be the facial expression of person standing in queue? Put yourself in that situation and imaging your smile itself is not real.

b) Root cause of business problem to be solved – they want to retain a frustrated or annoyed customer.

c) Is it the right way to solve this problem? – A famous story US spends millions in a pen whereas Russians use pencil in space missions.

d) Knowledge of privacy laws specially in the geography you are working or going to implement this solution – are you allowed to retrieve or link customer information with other information, does it require explicit approval of customer.

If we review above scenarios we do not foresee or not with much confidence can say we should go ahead with this, there are other ways to detect and when it is obvious no one likes to stand in long queues find alternative ways for customer satisfaction.

In cloud world we have seen people jump to technical solution immediately and then fail later numerous grounds like business feasibility, cheaper solution available, technical approach, security.

Many people who are looking to shift their career options from Application Architect / technical Architect to Cloud Solution Architect – ideation of solution is one of the

major requirements and this makes him a better solution architect.

So, when you are invited to provide solution to business problem review from following aspects (it is also, mentioned in software engineering aspects):

- **Economic Feasibility**: Whether the solution is economical or can there be other approaches to make it economical – even that may or may not require technical solution. That's why hold yourself and think whether you would like to spend this much amount of money on this type of use case. If you are not aware, it would be great to refer if someone else has done it and their feedback. Hence, have open discussion with enterprise architects, business stakeholders etc.
- **Technical Feasibility**: If you are coming from application architecture background you may easily be able to check feasibility but as a solution

architect you need to cover all non-functional aspects, like accessibility, security, efficiency etc. Running for few hundreds than running for millions of users may influence the approach. This is the area which may require deeper analysis of understanding business flows, integration points etc.

- **Operational Feasibility**: This is one of the major and most important factors that influences the success of solution, we all know that facial expressions, biometric readers were present long time back, even smart homes or to some extent IoT as well, but the adoption only happened at scale when other factors like Vendor support, more OEMs, managed support system, knowledge of systems also evolved. All these things influence the success of solution. For example, you are designing a chip that does bit amount of edge compute to take some decision, but there is no much manufacturing available that may provide that chip at scale, or you really designed something way complex and that is increasing training costs or scaling support system, the chances of success of such solutions decreases.

By above statements I never meant that one should not try new things, but we need to carefully draw a line between R&D and providing solution that will address the business problems.

R&D is an alternative approach to business problem and does not own success of adoption. However, when we are expected to provide solution, the expectation of business is to make it success and one must provide entire approach like analysis, solution, and post deployment support.

Solution designing

You may have already read basics of solution design in this book or may read it in detail in other chapters, here I will emphasize more on what is expected from cloud solution architect in terms of design decisions or solution design

solution architect in terms of design decisions of solution design.

A cloud solution architect post vetting and carefully probing the feasibility aspects must capture business data points specially in following areas:

- User information: Business Users, System Users and Admin Users
- User Authorization and Authentication
- Number of users
- Scaling requirements
- Fair Idea of functional aspects specially around information that will flow both internally and externally
- Data classification, governance, storage etc.

- Security and Privacy needs

It can happen one person may not know it all, but a cloud solution architect is expected at least to draw a solution blueprint and then he may pair up with specific subject matter experts like data architects, security architects who can provide guidance and best practices along with approach.

If you are coming with Infra background you may have good knowledge to contribute in non-functional aspects like cores, RAM required but you must invest time in learning application architectures and design like **Do not repeat yourself** (**DRY**), KISS principles (Keep it simple and stupid), cloud design patterns, and programming skills (at least basic one) – it would be good if you read basics of most of the languages like C#, Java, Nodejs, Python – how do they compile, configurations of IIS, Apache, tomcat etc.

On database side you must know how to create database, partitioning strategy, scaling support etc. along with replication support.

If you are coming from Application architecture perspective and already doing cloud native development you must invest time in learning Infra aspects like automation of Infra and PaaS components, updated knowledge of new VM types, networking basics, Web application firewalls, proxy solutions, endpoint security etc.

Post designing the flow of application couple of things worth considering and I learned from experience:

a) As the scaling will increase the synchronous activity in your system will start becoming a challenge

b) Technology is not a solution to every business problem but sometime a

change in business process can bring optimal or best use of technology

Let's understand above points when I say synchronous activity start becoming a challenge, lets understand it from an example we are carrying out some business process in which we need to insert or update a record and post successful submission of record we need to send a notification via SMS or email basis on caller configuration.

Above system was working fine let's say till 500 users but suddenly the spike came due to any reason and user base increased from 500 to 5 lacs – now many challenges started coming like system started going slow during insert or update there were locking issues, SMS service itself experienced throttling and breaking quite often.

If we study above problem carefully, we were synchronous earlier that means things are getting executed in serial order and if next block in computing is experiencing throttling the entire process will raise exceptions.

Now if we change the business process in following way:

a) Take the request to either update or insert the record in a queue type solution.

b) Create a subscriber that will act on that queue and insert or update the record.

c) One above operation is done it will queue another notification message.

d) A notification subscriber will act on that and deliver SMS message.

Above example is completely event driven and new languages and framework pushes for event driven applications because of their easiness in implementing resiliency, evolution, and extension.

Observe in above problem if Database also, becomes slow, or unresponsive we can still take the request, let's say SMS api or provider is slow we can implement retry or can bring another SMS provider if load is high.

So, reduce complexity, make the process looks easy, event based so, that it can scale easily, and we can modify it whenever required.

Improve design with latest services

Innovation is happening day by day, and with failures many design patterns are getting documented and making way for better architecture. As a good solution architect, it is your duty to review your existing architectures see how you can improve following:

1. Availability
    - Time for which system will be available to business considering all

transient and non-transient failures.

- If one goes down another instance should resume business flows – generally in cloud it is defined as Availability Sets (2 or more compute resources are provisioned in different fault domains) or Availability zones (2 or more resources are provisioned in different nearby data centers).

2. Resiliency – Ability of system to recover from transient, hardware failure.
    - In Cloud terminology this is generally defined as **Service Level Agreement** (**SLA**) like 99.9% which means uptime of respective resources during a month for example 99.9% => 99.9% of 730hrs which comes as 729 hrs 16 mins and 12 seconds.
    - A system that can retry in case of transient failure.

3. Operational Efficiency
    - In cloud have you configured all monitoring solutions like log analytics, monitoring solutions etc.
    - Help setting up L1, L2, L3 support.
    - Integrate with your SIEM solutions – like Qradar, CyberArk, Sentinel etc.
    - Configured alert rules both from Application and Cloud perspective.
4. Cost Management
    - Defining cost budgets, setting up alerts on percentage consumption.
    - Review consumption every week or fortnight in your daily scrums.
    - Seek advisory and encourage reservations to optimize costs.
    - The system should not break the boundaries of consumption as defined for system under various circumstances, optimized.

[Note: It is important to optimize cloud cost to encourage new projects, PoCs and pilots]

5. Security
    - In the era of zero trust the system should inspect each network

packet, authenticate and authorize every request, if possible, achieve confidential computing, encryption of information at rest or in transit.

- Adopt MITRE security automation framework.

If we observe architectures around 10 years back, they were mostly monolithic due to user behaviour and acceptance, constrained network boundaries and security requirements. But when we review the same architectures today such architectures have lots of design constraints which is either resulting them paying higher cost, scaling constraints, not so, good user experience.

Let's understand how availability can be improved, review architecture from SLA perspective and see whether each component failure aspect is covered. For example, if we are using web servers running Tomcat, how many instances are we running, you must have seen various cloud providers are giving virtual machine specific SLA so, see how we can increase SLA of our systems, like replicating one instances to multiple, which improves availability – for example if one instance goes down the other ones caters to load. The services are innovated to extent that you can use scale sets with autoscaling rules, fault domains etc.

Another way to improve availability and operational efficiency is using more managed services for example cloud providers are giving managed services like web environments, databases like SQL, oracle, PostgreSQL, MySQL etc. with even geo replication hence it makes absolute sense to update or upgrade your services towards more managed services. You will get rid of operational expenses like periodical patches, security updates, server monitoring and maintaining high availability.

In the new world review block wise, and if we move towards more managed services we will improve. For example, if you are not using any queue solution or doing tracking by keeping flags in database – its high time to shift those process to queuing or event-based systems.

Let me explain by example – I have reviewed a SQL database in which a column is kept for tracking operation status like SMS queued, then when service send SMS it updates the same record with send status like that – such systems are expensive when you do them in database, because event based services are available now with defined SLAs, if you use them it will resolve locking issues, some other datastores are available now like Name key value pairs and use them to store that kind of information in those datastores.

On Security aspect if you use managed services, it will remove overhead of patching runtimes, and you only need to work on protecting end point but inside runtime you need not to worry.

Above are few aspects where managed services or even more concrete functional

service not only resolves bottleneck in system but helps scaling it and improve architecture in every aspect of well architected framework.

Define process and facilitate adoption

As a solution architect you need to think how you will define process of adoption and execution both. So, when you architect a solution, you job does not end there but you should also, define a path for adoption.

Now, let's understand what process is, what is the relevance of process and why adoption.

The process is very important to execute and operate your architecture, hence you need to define all actors interacting with that architecture for example, development team, project management team, architecture board, release management teams and business development teams. It also, depends on scale but roles will be there and no matter a person may be playing more than one role.

In cloud related architecture, it would be helpful if CSA invest some time understanding the current way of working in that system and come up with skill gap that is required for them to manage cloud design architecture. There are still many companies who still works in waterfall model and really struggle with ad hoc requests. But if those people are trained on Agile, they can not only absorb those ad hoc requests but better plan the releases.

You must be observing a lot has been spoken around DevOps and companies are looking for good devops engineers this is because they want to streamline people, process and technology which are core pillars for any digital innovation. Some of the things that all teams must understand is various methodologies like Scrum, Agile, Kanban so, that entire tracking can take place.

As an architect help creating these Devops teams by guiding to prepare complete roadmap at least for next 3-6 months. In initial sprints there must be regular checkpoints to evaluate if everything is going by architecture and in case of challenges proper design considerations can be changed. For sure, if we want to keep our development costs low we must imbibe test driven development so, that we can ensure our approach with proper data points. I will give you an example where my tech lead has developed a small module that is pushing messages to queue and another subscriber that is acting, we just written a small test case and guess later we used the same test case for doing stress testing, fetching metrics etc. It has given us an edge to take decision basis on data. In cloud design there may be different ways

to achieve same results hence you need to bring availability, resiliency, security, and cost as your data points to take decision.

Once you establish entire devops process, second important aspect is adoption, cloud technology is more of self service, hence it is very important to evangelize your team on various services including making them understand on how to operate cloud services. It is important for you to design and help in designing complete Role based polices and authorization for various teams interacting with your architecture.

There may be people whose responsibility is to do small testing of various services before putting them in core solution, such people should be given access on Sandbox environment, where they can freely do experiments, create, and test services in isolation.

Another can be support teams whose responsibility is more of monitoring and invoking relevant action such teams should be given read only access to various services supporting that system so, that they can operate with full ease. Remember if CSA helps in creating good governance and monitoring policies you can extend services to more people for them to test, develop and adopt.

Conclusion

In this chapter, we have seen how CSA as a role is quite responsible and a great technical leadership role, the person besides good communication should be a good listener and then decisive. He should be looked upon as role model who helps everyone to change and we have seen how various design patterns, infra, database, web and security knowledge can help make a good CSA.

In our next chapter we shall be reading around to deal with data, understand different types of data and how that needs to be stored, various services where NoSQL, unstructured storage, RDBMS to be used.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 10

Treat Data and Database Separately

Introduction

In this chapter, I am going to tell you how we should treat data and database differently, will study what are different types of data we find today, what that data contains and what is the effective way to store data. Data will always have a database as a destination for storage and processing but with lots of the latest technology in place we can today take decisions that in a system how different type of data will be kept in different kind of storage.

Structure

This chapter is divided into multiple parts starting from:

- Type of Data
- Data Storage Options
- Processing of Data

Type of data

Every object in this universe is defined in two major categories – one is state, and another is behavior. While state defines the object the behavior which is dynamic

represents the action of that object. During execution of this behavior the data gets generated. Let's understand in a very simple example:

"**Mohan** drives a **SUV** from home to office daily and encourages carpooling."

In above line – **Mohan, SUV** are some of the objects while "**drives**" is the behavior by which a person and a vehicle interacts.

Now let's understand each of them what if we need to record the above sentence digitally or in modern world what all data gets generated, its volume, different formats and finally storage.

***Figure 10.1**: A person*

Mohan: This is a noun and represents a person as shown in Figure10.1, and if we define its state there will be information like Name (first, middle, last), Age, DOB, weight, height and many more like health, academics, picture, videos, social content etc. When you read the attributes for a person you must have started thinking that all this information is not stored at a single place but with different systems, different formats for example – person details like Name, age, DOB may be stored in some database like DB2, NOSQL, RDBMS like SQL / Oracle / MYSQL, similarly health, academics details in some other databases in tabular or jSON format, but pictures and videos are generally kept on unstructured storage like Flash drive, shared drives, google drives, one drives etc.

***Figure 10.2**: A SUV*

SUV: Again this is a noun and to represent SUV as shown in *Figure 10.2* there are details like Model, make, number of seats, engine details, color, pictures, videos, tags, manuals etc. Now again here the state can be stored in different format and pictures/videos in different formats, same goes with manuals (in pdf formats).

Figure 10.3: Driving

Driving: This is a verb shown as steering in Figure 10.3 and represents the association between a person and a car, the data points captured like speed, curve, GPS coordinates, route etc.

Broadly speaking we have the following categories:

Unstructured data: This is data which can be of any format and generally requires some tool to read, for example PDF documents (require PDF reader), pictures in jpg, gif, png formats – viewed by any picture display tool. This data is hard to classify unless you know its association hence this type of data must be kept in native storage like blob storage, hard drives.

So, if you design a system and see you are capturing this unstructured information by File Uploads – it is always good to keep content in Blob or Page blob storage. In cloud world these storages are provided as Azure Storage, AWS S3 which supports both Blob and Page Blob storage.

Previously this information was kept in RDBMS as BLOB fields which is not an efficient way of storing this information, rather you store it in Azure Blob/AWS S3 and store the path in RDBMS or even in jSON.

If you store this information in RDBMS imagine the cost you bear to read this information, storage cost which includes license costs, database processing. Whereas if you put this on Blob storage it can read when required and using simple https protocol with SAS key tokens.

Semi – Structured data: This data is native object state or behavior data like person object, vehicle as object etc and design it with various attributes for example I can represent person object with a unique key, name, age, DOB etc., similarly vehicle as vehicle id, model, make, mileage etc. One thing you notice is that it is next to impossible to capture all the information of these objects if you are planning to design a system for global usage. The reason is object representation changes with demographics like caste information, health analysis records etc. also, with time the representation changes for example vehicle may not have some feature today but it can have 6 months down the line. Hence, the way you store this information and uniquely define its representation should be a "key" "value" pair, where key represents uniqueness and value which can be dynamic, change as per requirement. This data is generally stored in NoSql databases like Mongodb, DynamoDB etc. This data is independent of a fixed schema / structure and above databases support storing them in "key-value" pair.

Structured data: This is the data which we have known for a period now, where we design with a fixed schema, generally these systems are designed for a specific requirement where we fix the state and behavior of an object. Like 100 fields to be captured for a person and same some 200 fields for a vehicle, then normalization is applied to reduce redundancy in data and make processing smoother. This

is applied to reduce redundancy in data and make processing smoother. This

is something we have been doing over the past 2 decades now and most of the systems are designed with this methodology. The challenge that these systems have is upgrading them whenever the business requirement changes, for example you want to add a few more attributes which leads to changes in various layers of the system.

Summarizing, when you design a system, please categorize your data inside that system which is structured (mostly transactional), semi structured (object state representation) and unstructured (any format) and conclude its storage.

Data storage options

In the last section we read about the type of data and classified it into 3 major categories unstructured, semi structured and structured also, seen where they should be stored. In this section we shall be reading the options we have for storing data, which type of data and how it can be processed.

For unstructured data the following are the different storage options available:

- File shares supporting NFS, SMB protocols
- Azure Block Blobs
- AWS S3
- GCP Blob Storage
- Azure Files

In the traditional world people also use File shares (NFS mounts) to store unstructured data, then various applications used to process that data by mounting those file shares.

If you want to reduce the costs of storage solutions prefer managed Block blob storage provided by cloud providers like Azure Block Blog storage, AWS s3 etc. You can also implement life cycle policy on this data like keep in Hot tier for 90 days after that move it to cool storage and after 180 days move it to Archive.

While designing your system, if you encounter unstructured data like log files, images, temp files, JSON moves them to Block Blob storage (optimized managed storage) implement life cycle policy. This will reduce lots of overhead purging in your system.

Extending the use cases further you can even use Azure Block Blob storage or AWS S3 as web servers for static content hosting powered by CDN. So, when you are hosting your next web application make sure you keep all `*.js` files, images, static

content (in form of jSON) in the storage account. There are several benefits – your network load on web servers will decrease significantly, with the CDN your content availability will get enhanced.

Semi structured data

This is mostly a json format data in the form of key, value pair like Azure CosmosDB, AWS Dynamo DB, MongoDB etc. let's understand why they are created specifically in last 10 years they gained lots of popularity with following example:

We wanted to create an ecommerce system which consists of majorly 3 components – Product catalog, Cart Management (for storing products) and payment. All this information if we have to design, we'll observe that it has data requirements from all perspectives like unstructured data (product images), their technical specifications (in form of pdfs), then linear attributes like ProductID, Name, Manufacturer etc., and lastly structured data to store transactions like which user did payment and what needs to be shipped.

In current scenario when above systems created with monolithic design face load you may experience conventional notations, like query being slow, database not scaling etc. However, if we bifurcate this information and start storing it appropriately, we can not only scale but make the system more efficient as well. Let's understand how:

In product catalog leverage Block blob storage for storing images (they can be hosted as web endpoint as well), any pdfs – as explained in previous section. Now for storing Product information we can make use of NoSql databases like CosmosDB, DynamoDB, Azure Tables etc. and path of pics can be stored in Jsons.

If we see above json we shall observe its denormalized information which will be frequently read in any ecommerce site. Imagine you store this information in RDBMS in product master table, publisher master table, product_publisher mapping, address table, publisher_address mapping. To fetch information frequently we would have created a view ProductView fetching all relevant information. If pics are stored in DB imaging the volume of data you would be picking (few MBs per record), this will flow via your DB server, App server and web server before reaching to browser.

You must read about pattern **Command and Query responsibility segregation** (**CQRS**) which tells us that in highly read intensive system we must split our read and write loads hence NoSql databases came into picture, they took load off RDBMS and comes with a flexibility of schema less as well. Even in runtime we can store denormalized information in these data stores like Azure CosmosDB or AWS DynamoDB (may define the Time limit as well) and a copy of it in any alternate data store.

NoSql Databases are designed to keep denormalized information, that does not have high consistency requirement and can replicate at a higher frequency.

Structured data:

Structured data is the data that we used to see for quite long now like fixed schema, the relevance of this structured data is not going to go away so, quickly because

we would be having business transactions (OLTP based systems) and applications will generate transactional data as well for example payments, reservations etc. In Microservices architecture most of the transactional handling will be done in middle layer rather than in RDBMS and will use various patterns like Compensating transactions (it deals with failure of transaction by implementing a reverse logic to compensate it), SAGA pattern (where one process will publish event and another child process will subscribe to it).

In any transactional system the requirement of RDBMS should not be ignored by introduction of classification of data but needs to be carefully segregated so, that we can make system more efficient and cost optimized.

For advance use cases that deal with geographic distribution of data we have methodologies in RDBMS as well like Sharding, partitioning etc. Hence when we design RDBMS going forward, we must keep partitioning in place this will help carefully purging of data without need to run expensive Delete queries and overheads of indexing.

We have known famous RDBMS like Microsoft Sql Server, Oracle, MySql etc. All these RDBMS systems are designed to store and handle structured information.

Processing of data

By now we have learned different types of data in a system and what are the different types and their storage available and now when you are going to design a new system you have better clarity to choose relevant tools and technology to store that information.

In this section we shall be reading about processing data and will understand different techniques to process it. As seen in systems today, a lot of analytical processing is required to improve business decision-making, lots of machine learning models being developed and transactional processing. For transactional processing most of the RDBMS provides querying capabilities to process data and most of them are designed to handle large volumes as well. We now have Azure SQL Hyperscale databases, Azure PostgreSQL Hyperscale databases which can store hundred terabytes of data and provides query capability on top of that, these can be used in high volumetric systems.

For fast processing in NoSQL databases ensure proper partitioning is done and one must think from Query first approach. This I would explain by following example

must think from Query first approach. This I would explain by following example which is a business problem for most users who are using CosmosDB, DynamoDB

Problem: If we start using CosmosDB or DynamoDB after some time the cost of processing increases and becomes unmanageable.

Some observations that one must look and go back to CQRS principles that states segregation of read from write hence read becomes priority and when read becomes priority one must think from Query first approach find that element or attribute that will uniformly divide the data and prevent hotbeds while processing the information.

If you are capturing events in CosmosDB for frequent read remember it should have a life attached to it, the best way to implement is send this event data (in json format) to CosmosDB and alternatively to any Blob storage or Key Value pair data and implement time to live for event stored in CosmosDB (say 15 days, 30 days etc). The reason for this life is practically when an event is captured it can be queried only for a specific period after that its read probability reduces and increases for analytical processing. Think of any system and then think of transactional information you can practically assume that frequency of read for a particular transactional data point reduces with time (max 6 months). After that this information is required for analytical processing like sampling for designing of models etc.

Another important aspect is indexing, one must enable indexing of that key attribute which is frequently used to fetch information from NoSql databases and also, prevent over indexing of data.

For catalog data or schema less information if CosmosDb is used for longer persistence remember to have proper partitioning and prefer using caching on top it to prevent frequent read on CosmosDB.

By above approaches you can reduce lots of read overheads in NoSql databases.

For RDBMS I would only say in new systems partitioning is a must and it should become part of regular admin activity to purge this information and send it for a data lake kind of store.

For unstructured data like PDFs, images if processing is required prefer SSD or ultra-SSDs based stores, use HPC like clusters to process at a scale.

Let's learn about processing of data (any format) for OLAP (analytical processing), we all must have heard about ETL jobs (Extract, transform and load) for mining information or analyze it or aggregate it for showcasing meaningful information.

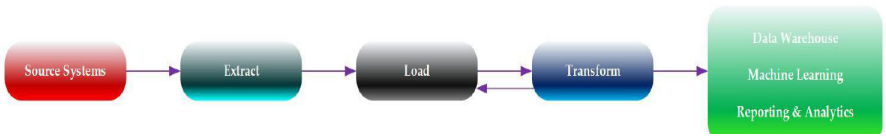


Figure 10.4: Workflow process

In *Figure 10.4* you must have observed a change in pattern where we have swapped load and transform, in new world of analytics and overall enterprise data strategy organizations are going for Enterprise data lake solutions and good reads are

Azure Data Lake landing zones, AWS Data landing zones if you wish to design complete strategy. The load part in above figure (*Figure 10.1*) defines enterprise data lake and after that Data engineering team can write various transformation routines like aggregation, sampling, filter and store it again in data lakes for further consumption. By this recursion, you can create various stages of data and each stage can have its own processing and use cases. One can observe transformation for Data warehouse which comes up by aggregation of data, Reports coming from querying and correlation, and Machine learning from sampling and modeling of data.

For processing of data in **Extract, load and transform** (**ELT**) one can use various tools like Azure Data Factory (Extract part), Air flow pipelines, loading can be done in Azure Data lake store that supports for structured and unstructured storage with hierarchical namespaces – making it super-efficient for faster reads. For transform one is flexible enough to use any kind of compute service like HPC, Databricks, Hadoop, HDInsight etc. and for final consumption there are many use cases like reporting, machine learning modeling, Data warehouse etc.

Conclusion

In this chapter we have read about types of data, how to deal with various aspects of data like unstructured, semi structured, and structured. How to store this data like unstructured data in file systems, blob storages, semi structured in NoSQL and structured in DBMS based systems. Next, we read about how to process this data, why it is important to think about enterprise data storage and stress of swapping ETL to ELT because you should not challenge source of information and capture it completely whether that is useful in today's perspective or not and then implement various transformations to fetching and study meaningful information.

Join our book's Discord space

CHAPTER 11

Frozen Architecture is Obsolete

Introduction

In a typical development cycle, teams aim to freeze their software architecture before they start the development cycle, and frozen architecture is the most desired artefact. Considering the rapid technological changes, changing customer needs and business processes, competition, regulatory changes, and global aspirations of organizations, won't a frozen architecture become obsolete from the day it was frozen? At the same time, if we do not freeze the architecture, development might get impacted. Let us discuss this in detail and try to view this problem from both perspectives.

Structure

We will be covering the following in this chapter:

- Architecture and design differences
- What freezing of architecture means
- Our views about freezing enterprise architecture

Objectives

In this chapter, we will discuss the pros and cons of freezing architecture in a development life cycle and look at what would be the best way forward of balancing

the act of freezing the architecture and, at the same time, balancing it so that it does not become obsolete. We will also discuss some differences between architecture and design and study how to handle stability and fix these artifacts.

Architecture versus design

We see these two terms being used interchangeably. This topic needs a whole book, so we can only discuss a few aspects of the relationship between architecture and design and how they are different yet related to each other.

Software architecture can be defined as an abstract plan for the structure of software, while design is a concrete plan to build something specific. Each Software architecture will have multiple concrete designs. Architecture is an overarching principle, and multiple designs are built aligning to the architecture. Architecture defines the fundamental high-level structure and properties of the software of a system. It is a plan that helps build an organization's business and technology strategy and limits software design to avoid known errors.

In concrete terms, design defines the detailed properties of a software. It creates specifications of software artefacts that will help developers implement the software. Software design gives developers a detailed idea for implementing consistent practices across the software.

At a high level, we can term architecture as a mechanism that describes software and operations at a broad and lateral level, and design as a mechanism that describes software in depth.

We also strongly believe that an architecture artefact should be seen as a "user manual" for the entire software development and operations life cycle. The manual should be a bible for the manager, developers, testers, deployers, and the operations and support staff. The manual should give them an abstract view on what, why and when of their function. The better the manual is, the better the software will be.

We also strongly believe that the need for architects' presence in the software and

operations life cycle should define the success and strength of their architecture. The architect's presence should keep fading as teams progress in the development cycle. Their presence and the need for them to keep modifying or explaining the architecture would imply that the quality of the architecture was not so great. This can be represented as follows:

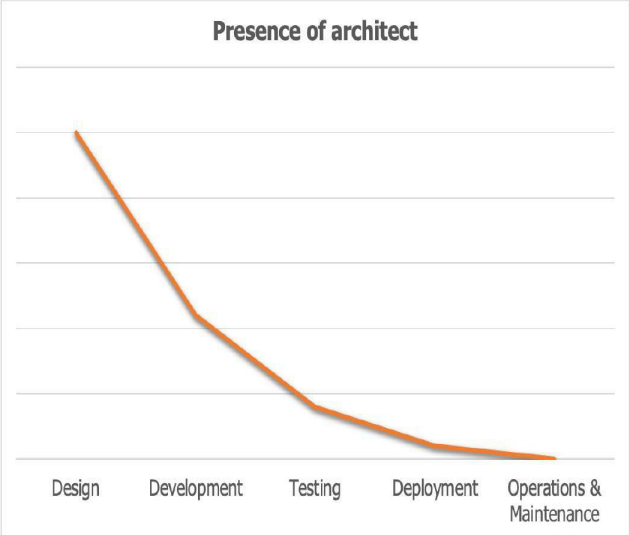


***Figure 11.1**: Architecture Effectiveness*

But would that mean that architecture needs to be frozen? Let us discuss this further.

Freezing an architecture

Let us delve into this discussion with the context of TOGAF. TOGAF standards define the following architecture phases:

- Business Architecture
- Information System Architecture
  - Data Architecture
  - Application Architecture
- Technology Architecture

[NB] Refer to the TOGAF framework for the details of these phases and their purpose.

Each of these architectures delivers multiple artefacts. There are categorised as Catalogue, Matrices and Diagram. Let's go through each of the artefacts and mention whether it can be frozen and what would be the pros and cons of freezing them.

Columns Phase, Artefact and Purpose have been referenced from **https://pubs.opengroup.org/togaf-standard/architecture-content/chap03.html**.

Business architecture

During this stage, the enterprise creates the target business architecture, which outlines how it should function to accomplish its business objectives and adapt to the strategic factors outlined in the architecture vision while addressing the concerns of stakeholders and the statement of architecture work. The process also involves pinpointing potential components for the architecture roadmap by assessing the discrepancies between the baseline and target business architectures.

Artefact	Purpose	Freezing of the artefact
Organizational / Actor Catalogue	The Organization/Actor Catalogue aims to compile a comprehensive list of all individuals and entities that engage with IT, including those who use and manage IT systems and applications.	Freezing Organizational / Actor Catalogue will ensure that the use cases that is built in design phase would have more definitive details and would be more concrete. This would also mean that any further change in user base would impact the design and may even impact bottoms-up. But if frozen, any change in user base or interactions would not be considered in the design, which would increase the software development recursion.
Driver / Goal / Objective Catalogue	The Driver/Goal/Objective Catalogue serves as a cross-organizational guide that outlines how an organization achieves its drivers and objectives, by providing reference of its goals and objectives, and (if necessary) through the use of measures.	Driver / Goal / Objective Catalogue won't change very frequently. Freezing this would not have much negative impact on downstream design and development activity.
Role Catalogue	The Role Catalogue's purpose is to create a comprehensive list of all authorization levels or access zones within an enterprise. This is to avoid complexity and unexpected consequences when locally understood concepts of authorization are combined on the user desktop, often caused by defining application security or behavior.	Considering ever-evolving security threats and realms, this should never be frozen. The Role Catalogue should be an ever-changing document with regular audit and reviews.

Artefact	Purpose	Freezing of the artefact
Business Service / Function Catalogue	The Business Service / Function Catalogue aims to provide a functional breakdown of the organization in a format that can be easily filtered, reported on, and queried, as a supplement to visual Functional Decomposition diagrams.	Business Service / Function Catalogue won't change very frequently due to less frequent changes in business functions. So, freezing this would not have a negative impact. But if a business function changes due to factors like regulations/ mergers/acquisitions, a change would mean a major impact on all the downstream activities and outputs.
Location Catalogue	The Location Catalogue is a list of all places where an enterprise conducts business operations or holds architecturally significant assets, such as data centers or end user computing devices.	Location Catalogue won't change very frequently due to less frequent changes in business functions. So, freezing this would not have a negative impact.
Process / Event / Control / Product Catalogue	The Process/Event/Control/ Product Catalogue offers a hierarchy of processes, events that initiate processes, outcomes of processes, and controls implemented during the execution of processes. This Catalogue serves as a supplement to any Process Flow diagrams created, enabling the enterprise to filter, report on, and query across organizations and processes to identify scope, commonality, or impact.	Process / Event / Control / Product Catalogue will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.
Contract / Measure Catalogue	The Contract/Measure Catalogue is a list of all service contracts and the measures associated with those contracts. It serves as the main list of service levels agreed across the enterprise.	Contract / Measure Catalogue would be linked to multiple catalogues and would always be dependent on them. It would not be advisable to freeze this artefact as this document would not be of much use when frozen.

Artefact	Purpose	Freezing of the artefact
Business Capabilities Catalogue	This is a catalog of capabilities that a business possesses to accomplish specific goals.	The Business Capabilities Catalogue won't change very frequently. So, freezing this would not have a high negative impact.
Value Stream Catalogue	This is a catalog of complete sets of value-adding activities that produce an overall outcome for a customer, stakeholder, or end user.	Value Stream Catalogue won't change very frequently. There might be new additions or changes in terms and conditions, but these would mostly not be critical. So, freezing this would not have a negative impact.
Value Stream Stages Catalogue	This is a catalog of complete sets of stages for the value-adding activities that create an overall outcome for a customer, stakeholder, or end user.	Value Stream Stages Catalogue won't change very frequently. There might be new additions or changes in terms and conditions, but these would mostly not be critical. So, freezing this would not have a negative impact.
Business Glossary Catalogue	This Catalogue includes the name and clear description of all elements present or interacting within the Enterprise Architecture.	The Business Glossary Catalogue won't change very frequently. There might be new additions or changes in terms, but these would not be critical maximum number of times. So, freezing this would not have a negative impact.
Business Interaction Matrix	The aim of this matrix is to show the interaction and relationships between organizations and business functions throughout the enterprise.	The Business Interaction Matrix will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.

Artefact	Purpose	Freezing of the artefact
Actor / Role Matrix	The aim of this matrix is to illustrate the actors who carry out which roles, which helps define security and skill requirements.	Considering ever-evolving security threats and realms, Actor / Role Matrix should never be frozen. The Actor / Role Matrix should be an ever-changing document with regular audit and reviews.
Value Stream / Capability Matrix	The aim of this matrix is to illustrate the capabilities needed to support each stage of a value stream.	Value Stream / Capability Matrix won't change very frequently. There might be new additions or changes in terms and conditions, but these would mostly not be critical. So, freezing this would not have a negative impact.
Strategy / Capability Matrix	The aim of this matrix is to illustrate the capabilities needed to support specific strategic statements.	The Strategy / Capability Matrix won't change very frequently. There might be new additions or amendments. The amendment might have a larger impact, but additions would lead to new projects. So, freezing this would have a moderate impact.
Capability / Organization Matrix	The aim of this matrix is to illustrate the organizational elements that execute each capability.	Capability / Organization Matrix won't change very frequently. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Business Footprint Diagram	A Business Footprint diagram illustrates the connections between business goals, organizational units, business functions, and services, and it shows how these functions are related to the technical components that provide the necessary capability.	The Business Footprint diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.

Artefact	Purpose	Freezing of the artefact
Business Service / Information Diagram	The Business Service/Information diagram illustrates the information required to support one or more business services. It illustrates what data is consumed or produced by a business service and may also show the origin of the information.	Business Service / Information Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.
Functional Decomposition Diagram	The aim of the Functional Decomposition diagram is to provide a clear visual representation of an organization's capabilities relevant to the architecture analysis on a single page. By focusing on the capabilities of an organization from a functional perspective, it allows for the development of models of the organization's functions without getting bogged down in discussions on the methods and processes used.	The Functional Decomposition Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.
Product Lifecycle Diagram	This diagram illustrates the potential state changes of a business product, from its initial creation or acquisition to its sale, disposal, or destruction.	Freezing Product Lifecycle Diagram would mean that we have assumed that the product life cycle has become fully matured and would not change. But the fact is that business innovations will continue unhindered, and it would really be non-practical to freeze this document. At the same time, freezing this would be a one-sided decision not helping in the development life cycle.

Artefact	Purpose	Freezing of the artefact
Goal / Objective / Business Service Diagram	The purpose of a Goal/Objective/Business Service diagram is to identify the relationship between a business service and the business objectives and strategies it supports in order to achieve the overall business vision.	The Goal / Objective / Business Service Diagram won't change very frequently. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Business Use-Case Diagram	A Business Use-Case diagram illustrates the interactions between the entities that use and provide business services. It shows the connections between the actors or other business services that consume a service and the business service itself, providing a more detailed understanding of the business capabilities and how they are utilized.	The Business Use-Case Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.
Organization Decomposition Diagram	The purpose of an Organization Decomposition diagram is to show the hierarchical structure of an organization and the relationships between actors, roles, and locations within the organization. It provides a visual representation of the different levels of an organization and how they interact with each other.	Considering the ever-evolving security threats and realms, the Organization Decomposition Diagram should never be frozen. The Organization Decomposition Diagram should be an ever-changing document with regular audit and reviews.

Artefact	Purpose	Freezing of the artefact
Process Flow Diagram	The purpose of the Process Flow diagram is to graphically depict the steps and flow of a process, including inputs, outputs, and any decision points or branches in the process. It provides a visual representation of the process and helps understand and analyze its components and interactions.	Process Flow Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.
Business Event Diagram	The purpose of the Business Event diagram is to show how different events trigger specific processes within a business and how these processes are connected to one another.	Business Event Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.
Business Capability Map	The purpose of the Business Capability diagram is to depict the abilities and resources required for an enterprise to achieve its goals and objectives.	The Business Capability Map won't change very frequently. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Value Stream Map	This is a diagram that illustrates the series of steps or stages involved in creating value for a customer, stakeholder, or end user from start to finish.	The Value Stream Map won't change very frequently. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.

Artefact	Purpose	Freezing of the artefact
Organization Map	This is a diagram depicting the connections and associations between key components of the enterprise, external partners, and individuals or groups with an interest or stake in the enterprise's operations.	The Organization Map won't change very frequently. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Information Map	A diagram that shows the connections between the key concepts that make up the enterprise, its partners, and stakeholders, represented in terms of business vocabulary, such as client, account, or product. This collection of information concepts and their relationships is used to map out the business architecture and identify the most important elements.	The Information Map won't change very frequently. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.

***Table 11.1**: Freezing of Business Architecture*

Data architecture

During this phase, the enterprise creates the target data architecture that supports the business architecture and the architecture vision while addressing the concerns of stakeholders and the statement of architecture work. Additionally, the process involves identifying potential components for the architecture roadmap by evaluating the differences between the baseline and target data architectures.

Artefact	Purpose	Freezing of the artefact
Data Entity / Data Component Catalogue	This Catalogue keeps track of all the data utilized within the enterprise, connecting data entities to the data components and illustrating the structure of the data entities.	The Data Entity / Data Component Catalogue will be an ever growing and changing document. Any software change in any department, maximum no of times, will change this. It is highly unlikely that this document would be of much use when frozen.

Artefact	Purpose	Freezing of the artefact
Data Entity/ Business Function Matrix	The Data Entity/Business Function matrix shows the connection between data entities and the specific business functions they support within the enterprise, highlighting how data entities are used to carry out business processes and services.	The Data Entity/Business Function Matrix will be an ever growing and changing document. Any software change in any department, maximum no of times, will change this. It is highly unlikely that this document would be of much use when frozen.
Application / Data Matrix	The purpose of the Application/ Data matrix is to show the connection between the software applications and the data that they access and modify within the enterprise.	The Application / Data Matrix will be an ever growing and changing document. Any software change in any department, maximum no of times, will change this. It is highly unlikely that this document would be of much use when frozen.
Conceptual Data Diagram	The main goal of the Conceptual Data diagram is to show the connections between crucial data elements within the organization, as seen from the perspective of business stakeholders.	Conceptual Data Diagram depicts the as-is stage and will need constant updates. Any gap in this document would mean a major bug in downstream applications. It is highly unlikely that this document would be of much use when frozen.
Logical Data Diagram	The purpose of the Logical Data diagram is to depict the logical connections and dependencies between important data entities within the organization, providing a clear understanding of how the data is structured and used.	Logical Data Diagram depicts the as-is stage and will need constant updates. Any gap in this document would mean a major bug in downstream applications. It is highly unlikely that this document would be of much use when frozen.

Artefact	Purpose	Freezing of the artefact
Data Dissemination Diagram	The main objective of the Data Dissemination diagram is to depict the connections between data entities, business services, and application components, and how the logical entities are implemented by the application components. This diagram helps in determining the IT resources required and refining the IT infrastructure, as well as evaluating the business importance of application components based on the value assigned to data.	Change in the data, application or business process will change the Data Dissemination Diagram. By default, the nature of data, application or business process keeps data dissemination changing, so it is highly unlikely that this document would be of much use when frozen.
Data Security Diagram	This diagram illustrates the connections between data entities, the roles, organizational units, and applications that utilize them.	Considering ever-evolving security threats and realms, the Data Security Diagram should never be frozen. It should be an ever-changing document with regular audit and reviews.
Data Migration Diagram	This diagram illustrates the process of moving data from its original source to its final destination, including any transformations or cleaning that may occur during the transfer.	The Data Migration Diagram won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Data Lifecycle Diagram	The purpose of the Data Lifecycle diagram is to depict the stages and transitions of business data throughout its existence, from creation or receipt to disposal or destruction, within the context of the business processes that govern it.	Changes in the data, application or business process will change the Data Lifecycle Diagram. By default, the nature of data, application or business process keeps data lifecycle changing, so it is highly unlikely that this document would be of much use when frozen.

***Table 11.2**: Freezing of Data Architecture*

Application architecture

During this phase, the enterprise establishes the target application architecture, which supports the business architecture and the architecture vision while addressing stakeholder concerns and the statement of architecture work. Moreover, the process entails identifying possible components for the architecture roadmap by analyzing the gaps between the baseline and target application architectures.

Artefact	Purpose	Freezing of the artefact
Application Portfolio Catalogue	The purpose of this Catalogue is to create and maintain a comprehensive list of all the applications used in the enterprise. This list helps establish the scope of change initiatives that could potentially impact different types of applications. By agreeing on an Application Portfolio, it is possible to standardize and govern a set of applications.	The Application Portfolio Catalogue won't change frequently and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Interface Catalogue	The Interface Catalogue serves as a guide for understanding the connections and dependencies between different applications in the enterprise. It documents the interfaces between the applications, allowing early identification and management of potential impacts on change initiatives.	Interface Catalogue won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes. The changes sometime would have large impacts and would also mean massive testing effort. The impact will also depend on how teams have developed the consumption of the interface. So, freezing this would not be a great approach, but at the same time, the as-is state would need to be documented with great clarity and concrete evidence/examples.
Application / Organization Matrix	The goal of the Application/ Organizational Unit matrix is to show the connections between the various applications used in the enterprise and the specific organizational units that use or support them. This allows for a clear understanding of which applications are important to different parts of the organization and how they are being utilized.	The Application / Organization Matrix won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.

Artefact	Purpose	Freezing of the artefact
Role / Application Matrix	The main objective of the Role/ Application matrix is to show the connection between the applications and the specific business roles that utilize them within the organization. It illustrates how different applications support the functions and tasks of different roles.	The Role / Application Matrix won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Application / Function Matrix	The main goal of the Application/ Function matrix is to show the relationship between various applications and the specific business functions they support within the enterprise. It helps understand how different applications contribute to the overall functioning of the business and how they are linked to different business activities.	The Application / Function Matrix won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Application Interaction Matrix	The purpose of the Application Interaction matrix is to show how applications interact with one another by depicting the communication relationships between them.	The Application Interaction Matrix will depend on the Interface Catalogue and will follow changes as per the latter. The changes would have large impacts sometimes and would also mean massive testing effort. The impact will also depend on how teams have developed interactions. So, freezing this would not be a great approach, but at the same time, the as-is state would need to be documented with great clarity and concrete evidence/examples.

Artefact	Purpose	Freezing of the artefact
Application	The main goal of the Application Communication diagram is to show the interactions and connections between different applications within the enterprise, as outlined in the metamodel entity.	The Application Communication Diagram will depend on the Interface Catalogue and will follow changes as per the latter. The changes would have large impacts sometimes and would also mean massive testing effort. The impact will also depend on how teams have developed interactions. So, freezing this would not be a great approach, but at the same time, the as-is state would need to be documented with great clarity and concrete evidence/examples.
Application and User Location Diagram	The purpose of the Application and User Location diagram is to depict the geographical distribution of applications and how they are used, hosted, developed, tested, and released by the end user, organization units, and locations within the enterprise.	The Application and User Location Diagram won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Application Use-Case Diagram	The purpose of an Application Use-Case diagram is to depict the relationships and interactions between the users or systems that utilize an application's services and the application itself, providing a visual representation of how and when the application's functionality is utilized.	The Application Use-Case Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen.

Artefact	Purpose	Freezing of the artefact
Enterprise Manageability Diagram	The purpose of the Enterprise Manageability diagram is to illustrate the connections and interactions between applications and the various components that are responsible for maintaining and managing the overall operation of the solution. This includes monitoring, troubleshooting, and adjusting to ensure smooth and efficient performance.	The Enterprise Manageability Diagram won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Process / Application Realization Diagram	The goal of the Process/ Application Realization diagram is to visually demonstrate the step-by-step interactions and dependencies between various applications that are involved in carrying out a specific business process.	The Process / Application Realization Diagram won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Software Engineering Diagram	The purpose of the Software Engineering diagram is to depict the internal structure and organization of an application from a development perspective by breaking it down into smaller components, such as packages, modules, services, and operations.	The Software Engineering Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable and accepted by business teams in the current ever-changing and highly competitive world. It is highly unlikely that this document would be of much use when frozen. At the same time, this document will need a hybrid approach, where a frozen version is provided to development team and new versions are getting modified.

Artefact	Purpose	Freezing of the artefact
Application Migration Diagram	The purpose of the Application Migration diagram is to depict the movement of applications from their current state to a new target state, including any interfaces and dependencies that need to be considered during the migration process. This diagram helps accurately estimate the cost and effort required for the migration by identifying the specific applications and interfaces that need to be transitioned.	The Application Migration Diagram won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Software Distribution Diagram	The purpose of the Software Distribution diagram is to depict the organization and distribution of application software throughout the enterprise, providing insights into how the software is structured and deployed. It can be used as a tool in planning systems upgrades or application consolidation projects.	The Software Distribution Diagram won't change frequently, and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.

***Table 11.3**: Freezing of Application Architecture*

Technology architecture

During this phase, the enterprise creates the target technology architecture that enables the delivery of the target business, data, and application building blocks through technology components and services while considering stakeholder concerns and the statement of architecture work. Furthermore, the process involves identifying potential components for the architecture roadmap by examining the disparities between the baseline and target technology architectures.

Artefact	Purpose	Freezing of the artefact
Technology Standards Catalogue	The purpose of the Technology Standards Catalogue is to document the agreed upon standards for technology across the enterprise, including the specific technologies, versions and lifecycle management for those technologies, as well as the refresh cycles for them.	Technology Standards Catalogue won't change very frequently and possibly, this will be a one-time artefact. There might be new additions or changes. But these would not be critical majority of times. So, freezing this would not have a negative impact.
Technology Portfolio Catalogue	The Technology Inventory Catalogue keeps track of all the technology used in the enterprise, including hardware, infrastructure software, and application software. It helps manage the life cycle of technology products and versions and forms the basis for establishing technology standards.	The Technology Portfolio Catalogue won't change very frequently and possibly, this will be a one-time artefact. There might be new additions or changes, but these would not be critical majority of times. So, freezing this would not have a negative impact.
Application / Technology Matrix	The purpose of the Application / Technology matrix is to depict the relationship between applications and the technology platforms they are built or run on within the enterprise. It helps identify the applications that are dependent on specific technology and can assist in technology upgrade or consolidation projects.	Application / Technology Matrix won't change very frequently and possibly, this will be a one-time artefact. There might be new additions or changes, but these would not be critical majority of times. So, freezing this would not have a negative impact.
Environments and Locations Diagram	The purpose of the Environments and Locations diagram is to show the relationship between the physical locations where applications are hosted, the technologies and applications used at those locations, and the locations from where business users typically access the applications. It provides a	The Environments and Locations diagram won't change very frequently and possibly, this will be a onetime artefact. There might be new additions or changes, but these would not be critical majority of times. So, freezing this would not have a negative impact.

	visual representation of the deployment of applications and technologies across the enterprise.	

Artefact	Purpose	Freezing of the artefact
Platform Decomposition Diagram	The Platform Decomposition diagram illustrates the breakdown of the technology infrastructure that supports the Information Systems Architecture. It includes all elements of the infrastructure platform and provides a comprehensive view of the enterprise's technology landscape. The diagram can be further expanded to show the mapping of the technology platform to specific application components within a particular business function or process. It may include detailed specifications, such as product versions and number of CPUs, or it may simply provide a high-level visual representation of the technical environment.	Platform Decomposition Diagram will always be under immense pressure to be frozen because changes in this document will make all downstream artefacts obsolete. At the same time, freezing this document would not be advisable due to the ever-changing world of technology. We have seen projects, when being delivered to customers, become obsolete on day 1 due to old technology stack. Microservices pattern and Polyglot give us the capability to ensure that we keep changing this document and keep making downstream applications closer to new technology. It is highly unlikely that this document would be of much use when frozen.
Processing Diagram	The Processing diagram illustrates the units of code and configuration that would be deployed and also explains how these will be deployed.	Processing Diagram won't change very frequently and possibly, this will be a one-time artefact. There might be new additions or changes, but these would mostly not be critical. So, freezing this would not have a negative impact.
Networked Computing / Hardware Diagram	This document illustrates deployment architecture of an application and includes details about landing zone, network, compute, storage, protection and security. It is common practice for applications to be deployed and hosted in a shared and common infrastruc-	Networked Computing / Hardware Diagram normally won't change very frequently and possibly, this will be a one-time artefact. But changes in technology stack and development methodology can impact this artefact. Non-functional requirements are the

	ture environment.	most important inputs into this artefact. Other inputs still would not impact this much. Freezing this would still be fine and would not have a negative impact.

Artefact	Purpose	Freezing of the artefact
Network and Communications Diagram	The Network and Communications diagram describes how different technology assets would communicate with each other at the network layer.	Network and Communications Diagram normally won't change very frequently, and possibly, this will be a one-time artefact. But changes in technology stack and development methodology can impact this artefact. Non-functional requirements are the most important inputs into this artefact. Other inputs still would not impact this much. Freezing this would still be fine and would not have a negative impact.

***Table 11.4**: Freezing of Technology Architecture*

Conclusion

Let us discuss if freezing the architecture is really a good practice or something that should be avoided.

Is frozen architecture obsolete architecture?

If we refer to table 11.4 above, there are multiple artefacts in each phase of architecture that need to change or where change cannot be prevented. It ranges across areas, like business, information system, technology, operations, and management. It would impact a software development life cycle negatively when freezing these artefacts.

On the other hand, we also saw that there are artefacts that can be frozen easily and can be frozen across projects and software development life cycle. Also, some artefacts should be frozen.

Each software development life cycle has tailored processes, and the artefacts to be generated are pretty much pre-decided. The list is usually a subset or the same as listed in multiple sections above. Our view is that your artefact list and the factors we have mentioned should be the bases on which you take a judicious call of whether the architecture can be frozen. A frozen architecture will become obsolete, and a non-frozen would become lead to extra costs to the software development cycle.

frozen would become lead to extra costs to the software development cycle.

What about enterprise architecture?

This brings up another question, "Should enterprise architecture be frozen?". We will not go into the details of what enterprise architecture is, but we will surely

encourage you to read through the current TOGAF framework and understand enterprise architecture.

In miserly language, we can state that enterprise architecture is a well-defined practice that includes transforming people, process, and technology of an enterprise to upgrade it from one state to another. It is a strategy that is driven by business growth needs of an organization.

For example, how can we transform a bank so that it can transform its business and move from a near bankrupt state to becoming profitable.

According to TOGAF, each enterprise architecture will follow the Architecture Development Method (ADM). One ADM will generate multiple projects and multiple changes.

Can or should Enterprise architecture be frozen? The ideal answer is yes. It is more of a vision of an organization and sets goals for the enterprise. Freezing an architecture would be required to ensure that organizations run the ADM, and the outputs/feedback from the ADM drives the next ADM.

Join our book's Discord space

CHAPTER 12
What Exactly is Software Architecture?

Introduction

Software architecture and its definition have changed multiple times since its inception. There are multiple definitions, and each have their own reasons. In this chapter, we will be discussing this in detail and bring fresh perspective to it.

Structure

We will be covering the following topics in this chapter:

- Discuss architecture and design differences
- What freezing of architecture means
- Views on freezing enterprise architecture

Objectives

Software architecture refers to the fundamental organization of a software system, which includes the overall structure of the system, the components and their relationships, and the principles and guidelines governing their design and evolution. In this chapter, we will go through various views of software architecture and understand the basic principle behind it.

Software architecture

Software architecture refers to the fundamental organization of a software system, which includes the overall structure of the system, the components and their relationships, and the principles and guidelines governing their design and evolution.

It defines how different parts of the software system interact with each other and how they work together to achieve the desired functionality. It also specifies the mechanisms for managing system requirements, design decisions, and implementation details, which enables efficient development, maintenance, and evolution of the software system.

Software architecture is crucial for ensuring the quality and success of a software project, as it establishes a blueprint for the development team to follow and helps them make informed decisions about design and implementation.

Technology vision versus software architecture

Technology vision and software architecture are two different concepts that are crucial for the success of any software development project. While both concepts are related to the overall design and implementation of a software system, they differ in their scope, purpose, and level of detail.

Technology vision is a high-level concept that defines the long-term goals and objectives of an organization in terms of technology. It is a broad, aspirational statement that outlines the desired outcomes and the role that technology will play in achieving them. Technology vision is typically driven by the business strategy of an organization and is meant to guide its technology investments and decision-making in the long term.

For example, a technology vision for a manufacturing company might be to leverage

the latest digital technologies to optimize the production process, improve supply chain management, and enhance customer experience. This vision provides a high-level direction for the organization's technology investments and serves as a guiding principle for the development of software systems that support these goals.

On the other hand, software architecture is a detailed and structured approach to designing and implementing software systems. It is a set of design principles and guidelines that define the structure, behaviour, and functionality of a software system. It considers the technical and functional requirements of a software system and specifies how its different components interact with each other to achieve the desired functionality.

Software architecture provides a detailed plan for the development of software systems and serves as a blueprint for the development team to follow. It includes the identification of key components of the software and their relationships, the selection of appropriate technologies and tools, and the specification of design patterns and best practices.

For example, the software architecture for a manufacturing company might include the identification of key modules like production planning, inventory management, and customer relationship management. It might also specify the technologies and tools that should be used to develop each module, and the design patterns and best practices that should be followed to ensure a robust and scalable system.

Are enterprise architecture and software architecture the same?

Enterprise architecture (**EA**) and software architecture are both essential for the design and implementation of complex IT systems in modern organizations. While there are some similarities between these two disciplines, there are also significant differences in their scope, goals, and approaches.

EA is a strategic planning process that seeks to align the organization's IT capabilities with its business objectives and goals, and with the needs of its stakeholders. EA provides a framework for managing the organization's IT resources, including hardware, software, data, and networks. The goal of EA is to improve the organization's efficiency, effectiveness, and agility by ensuring that its IT infrastructure is aligned with its business objectives and goals.

Software architecture, on the other hand, is a process of designing and developing software systems. It focuses on the detailed structure and behavior of the software, including its components, modules, interfaces, and interactions. Software architecture is concerned with the technical aspects of software development, such as the selection of programming languages, frameworks, and design patterns. Its goal is to ensure that the software system is reliable, efficient, and scalable.

goal is to ensure that the software system is reliable, efficient, and scalable.

One way to understand the difference between EA and software architecture is to compare the levels of abstraction at which they operate. EA operates at a higher level of abstraction, focusing on the overall structure and operation of the organization. It provides a holistic view of the organization's IT infrastructure, including its hardware, software, data, and networks. It seeks to optimize the overall performance of the organization by aligning its IT capabilities with its business objectives and goals.

Software architecture, on the other hand, operates at a lower level of abstraction. It focuses on the detailed design and development of software systems, including the

selection of programming languages, frameworks, and design patterns. Software architecture seeks to optimize the performance of individual software systems by ensuring that they are reliable, efficient, and scalable.

Another key difference between the two is the scope of their respective concerns. EA is concerned with the overall structure and operation of the organization, including its business processes, information systems, and technology infrastructure.

Software architecture, on the other hand, is concerned with the detailed design and development of individual software systems. It focuses on the structure, behavior, and performance of the software system, including its components, modules, and interactions.

Types of architecture

There are different types of software architecture, each with its own approach, strengths, and weaknesses. Here are some of the most common types of software architecture:

Monolithic architecture

In this type of architecture, the entire software application is built as a single, self-contained unit. All the application's components are tightly integrated, and any changes or updates require the entire application to be redeployed. Monolithic architecture is simple and easy to develop and deploy, but it can be challenging to scale and maintain. Some of the key components of a monolithic architecture are mentioned here.

User interface layer

The user interface layer is responsible for rendering the user interface of the

application. It includes all the UI components, such as forms, buttons, menus, and other controls, that allow the user to interact with the application.

Business logic layer

The business logic layer contains the core business rules and processes that govern the behavior of the application. It is responsible for processing the user input and generating the appropriate output.

Data access layer

The data access layer is responsible for accessing the database and retrieving or storing data. It provides a way to interact with the database by creating, reading, updating, and deleting data.

Application infrastructure

The application infrastructure consists of various components and services that support the application's operation, including logging, monitoring, security, and authentication.

Third-party libraries

Monolithic architectures often rely on a range of third-party libraries and frameworks to support the development and operation of the application. These libraries may include database connectors, UI frameworks, or other tools and services.

Deployment and release processes

The deployment and release processes are also an essential component of a monolithic architecture. These processes are responsible for building, packaging, and deploying the application to the production environment.

Overall, a monolithic architecture is a tightly coupled architecture where all the components of the application are combined into a single executable file. While this approach may offer some benefits in terms of simplicity and ease of deployment, it can also make it challenging to scale the application or update individual components without affecting the entire system.

Service-oriented architecture

In **Service-oriented architecture** (**SOA**), the software application is divided into separate, loosely coupled services that communicate with each other through a common protocol. Each service can be developed and deployed independently, which makes it easier to maintain and scale. However, SOA can be more complex

and requires careful planning and management to prevent performance and compatibility issues. The following are the key components of a service-oriented architecture:

Service

The service is the fundamental building block of an SOA. It is a self-contained unit of functionality that can be accessed and used by other services or applications. Services are designed to be reusable, and they expose their functionality through a well-defined interface that can be accessed using standard protocols.

Service interface

The service interface defines how the service can be accessed and used by other services or applications. It includes information like the input and output parameters of the service, the data format used, and the protocols that are supported.

Service registry

The service registry is a central repository that holds information about the services that are available in the SOA. It includes information like the location of the service, the service interface, and any metadata associated with the service.

Service broker

The service broker is responsible for managing the interactions between services in the SOA. It mediates the communication between services, handling tasks like routing requests to the appropriate service, and managing the security and authentication of the service.

Service bus

The service bus is a middleware component that provides a platform for services to communicate with each other. It enables services to communicate with each other regardless of their location or platform, and it provides features like routing, transformation, and protocol conversion.

Service orchestration

Service orchestration is the process of coordinating the execution of multiple services to achieve a particular business function. It involves defining the order in which services are executed, handling any errors or exceptions that occur, and managing the overall flow of the process.

Service composition

Service composition is the process of creating new services by combining the existing services. It involves defining the inputs and outputs of the new service and any business rules or processes that are involved.

Service monitoring

Service monitoring is the process of monitoring the performance and availability of the services in the SOA. It involves measuring metrics like response time, throughput,

and error rates, and using this information to improve the overall performance and availability of the services.

In summary, SOA is a flexible and scalable architecture that is based on the concept of services. Its key components include services, service interfaces, service registry, service broker, service bus, service orchestration, service composition, and service monitoring. These components work together to enable the creation, management, and use of services in an SOA.

Microservices architecture

Like SOA, microservices architecture involves breaking down an application into smaller, independent services that communicate with each other through APIs. However, in microservices architecture, each service is designed to perform a single, specific function. This makes the application more flexible, scalable, and easier to maintain than SOA. However, it can also be more complex and challenging to develop. Remember, containerization is not microservices architecture. The following are the key components of a microservices architecture.

Service

A microservice is a small, independent component that performs a single business function. Each microservice has its own database and is responsible for handling its own business logic.

API gateway

The API gateway is the entry point for all external requests into the microservices architecture. It acts as a reverse proxy, routing requests to the appropriate microservice based on the request URL or other criteria.

Service registry and discovery

The service registry is a central repository that holds information about the microservices that are available in the architecture. The service discovery component allows microservices to find and communicate with other microservices in the architecture.

Load balancer

The load balancer is responsible for distributing incoming requests to the available instances of a microservice. It ensures that each instance of a microservice is used efficiently and that the overall performance of the architecture is optimized.

Containerization

Containerization is the process of packaging an application and its dependencies into a container. Containers are lightweight and portable, and they allow microservices to be deployed and managed independently of the underlying infrastructure.

Configuration management

Configuration management is the process of managing the configuration of the microservices architecture. This includes managing environment variables, database connections, and other configuration settings.

Monitoring and logging

Monitoring and logging are critical components of a microservices architecture that allow developers to monitor the performance and availability of the architecture and troubleshoot any issues that arise.

Database management

In a microservices architecture, each microservice has its own database. The database management component is responsible for managing these databases and ensuring that data is stored and retrieved correctly.

In summary, microservices architecture is a software architecture that structures an

In summary, microservices architecture is a software architecture that structures an application as a collection of loosely coupled, independently deployable services. Its key components are microservices, API gateway, service registry and discovery, load balancer, containerization, configuration management, monitoring and logging, and database management. These components work together to enable the creation, deployment, and management of microservices-based applications.

Event-driven architecture

In this type of architecture, the application's components communicate with each other through events, such as messages or signals. The system responds to these events by triggering actions or processes. Event-driven architecture is scalable, flexible, and responsive to changing conditions, but it can be complex and difficult to implement. The following are the key components of an event-driven architecture:

Event producers

Event producers are components that generate events. They can be sensors, users, applications, or other systems that produce events when certain conditions are met

or actions are performed. Event producers can be internal to a system or external to it.

Event consumers

Event consumers are components that receive and process events. They can be applications, services, or other systems that react to events by performing actions or triggering other events. These can also be internal to a system or external to it.

Event bus

The event bus is a messaging system that facilitates the communication between event producers and event consumers. It is responsible for receiving, routing, and delivering events to their appropriate destinations. The event bus can be implemented using various technologies, such as message brokers, publish-subscribe systems, or event streaming platforms.

Event store

The event store is a database that stores all the events that are produced and consumed in the system. It is a source of truth for the system's state and history, and it can be used to replay events, perform analytics, or support auditing and compliance requirements.

Event processing

Event processing is the core logic of an event-driven architecture. It includes the rules, algorithms, and workflows that determine how events are processed, filtered, transformed, and reacted to. Event processing can be implemented using various techniques, such as event-driven microservices, serverless functions, or stream processing frameworks.

Event sourcing

Event sourcing is a design pattern that emphasizes the use of the event store as the primary source of truth for the system's state. It involves storing all the events that led to the current state of the system instead of storing the current state itself. Event sourcing can improve the system's scalability, reliability, and auditability.

Command Query Responsibility Segregation

Command Query Responsibility Segregation (**CQRS**) is a design pattern that separates the commands that modify the state of the system from the queries that read the state of the system. It involves having separate models and databases for commands and queries, and using events to synchronize them. CQRS can improve the system's performance, scalability, and maintainability.

In summary, event-driven architecture is a software architecture paradigm that focuses on the production, detection, and consumption of events, and the reaction to them. Its key components are event producers, event consumers, event bus, event store, event processing, event sourcing, and CQRS. These components work together to enable the creation, deployment, and management of event-driven applications.

Layered architecture

This type of architecture involves dividing the application into separate layers, each of which performs a specific function, such as presentation, business logic, and data storage. Each layer communicates with the adjacent layers through well-defined interfaces. Layered architecture is simple, and easy to maintain and test, and is

often used in web applications. The following are the key components of a layered architecture.

Presentation layer

The presentation layer is the topmost layer of the architecture, and it is responsible for the user interface and presentation logic. It is responsible for displaying data to the user, receiving user input, and validating it. The presentation layer can be implemented using various technologies, such as HTML, CSS, JavaScript, or a **graphical user interface** (**GUI**) toolkit.

Application layer

The application layer is the layer that contains the business logic of the system. It is responsible for implementing the use cases of the system, coordinating the interactions between the layers, and enforcing the business rules. The application layer can be implemented using various programming languages, frameworks, or libraries.

Domain layer

The domain layer is the layer that contains the domain model of the system. It is responsible for representing the business concepts and entities, and the relationships between them. The domain layer can also contain the business rules and logic that apply to the domain model. It can be implemented using **object-oriented programming** (**OOP**) techniques, such as classes, objects, and inheritance.

Data access layer

The data access layer is the layer that is responsible for accessing and manipulating the data stored in the system. It provides a uniform interface for the application layer to access the data, and it shields the application layer from the details of the data storage and retrieval mechanisms. The data access layer can be implemented using various technologies, such as SQL, **Object-Relational Mapping** (**ORM**) frameworks, or NoSQL databases.

Infrastructure layer

The infrastructure layer contains the supporting infrastructure of the system, such as the network, the security, the logging, and the caching mechanisms. It is responsible for providing the non-functional requirements of the system, such as performance

for providing the non-functional requirements of the system, such as performance, scalability, and reliability. It can be implemented using various technologies, such as middleware, libraries, or frameworks.

Cross-cutting concerns

Cross-cutting concerns are aspects of the system that cut across multiple layers and modules, such as logging, security, and error handling. They are typically implemented as separate modules or components that can be used by multiple layers of the architecture. Cross-cutting concerns can be implemented using various techniques, such as **aspect-oriented programming** (**AOP**), decorators, or interceptors.

In summary, layered architecture is a software architecture pattern that divides the system into multiple layers, each responsible for a specific set of functionalities. Its key components are the presentation layer, the application layer, the domain layer, the data access layer, the infrastructure layer, and cross-cutting concerns. These components work together to enable the creation, deployment, and management of layered applications.

Client-server architecture

In client-server architecture, the application is divided into two components: the client, which runs on the user's device and handles the presentation and user interaction, and the server, which runs on a remote server and handles the business logic and data storage. Client-server architecture is simple, scalable, and secure, but it can be slower than other architectures due to the need for network communication. The following are the key components of client-server architecture.

Client

The client is the user-facing component of the system. It can be a desktop application, a mobile application, a web browser, or any other type of software that interacts with users. The client is responsible for presenting the user interface, accepting user input, and communicating with the server to retrieve and display data. It can be implemented using various programming languages, frameworks, or libraries.

Server

The server is the back-end component of the system that provides data and services to the client. It can be a physical or virtual machine that runs the server software, such as a web server, an application server, or a database server. The server is responsible for processing client requests, retrieving and storing data, executing business logic, and generating responses to send back to the client. The server can be implemented using various technologies, such as Java, .NET, or Node.js.

Network

The network is the communication infrastructure that connects the client and the server. It can be a **local area network** (**LAN**), a **wide area network** (**WAN**), or the internet. The network is responsible for transmitting data between the client and the server, and ensuring the reliability, security, and availability of the communication. The network can be implemented using various technologies, such as TCP/IP, HTTP, or WebSocket.

Protocol

The protocol is the set of rules that governs the communication between the client and the server. It defines the format and structure of the messages exchanged between the client and the server, and the semantics and syntax of the commands and responses. The protocol can be standardized, such as HTTP or TCP, or proprietary, such as the **Remote Procedure Call** (**RPC**) or the **Simple Object Access Protocol** (**SOAP**).

Data storage

Data storage is the component that is responsible for storing and retrieving the data used by the system. It can be a file system, a database, or a caching system. The data storage is responsible for ensuring the consistency, integrity, and security of the data, and for providing the necessary performance and scalability. The data storage can be implemented using various technologies, such as MySQL, PostgreSQL, or MongoDB.

Security

Security is the component that ensures the confidentiality, integrity, and availability of the system. It can be implemented using various mechanisms, such as authentication, authorization, encryption, or firewalls. Security is critical in client-server architecture to prevent unauthorized access, data breaches, and other security threats.

In summary, the client-server architecture is a software architecture pattern that divides the system into two main components: the client and the server. Its key components are the client, the server, the network, the protocol, the data storage, and the security. These components work together to enable the creation, deployment

the security. These components work together to enable the creation, deployment, and management of distributed systems that provide data and services to the users.

Peer-to-peer architecture

In peer-to-peer architecture, the application is divided into equal peers, each of which performs a specific function and communicates with other peers through a common protocol. Peer-to-peer architecture is highly scalable and resilient, but it can be complex and challenging to implement and maintain. The following are the key components of P2P architecture.

Nodes

Nodes are the individual computers or devices that participate in the P2P network. Each node has equal status and can communicate with any other node in the network. Nodes can share files, data, or processing power with other nodes, and they can also request resources from other nodes. They can join or leave the network at any time, and the network can adapt to changes in the number and distribution of nodes.

Connections

Connections are the links between nodes that enable communication and resource sharing. Connections can be established in various ways, such as direct communication over the internet, through a **local area network** (**LAN**), or using

specialized protocols like BitTorrent. Connections can be persistent or transient and can be used to transmit data, messages, or commands between nodes.

Routing

Routing is the process of determining the path that data should take through the network to reach its destination. In P2P architecture, routing is typically decentralized, with each node contributing to the routing decisions based on its knowledge of the network topology and its own resources. Routing algorithms can be based on various factors, such as proximity, load, or reliability.

Indexing

Indexing is the process of organizing the resources available in the network to facilitate their discovery and retrieval. In P2P architecture, indexing can be centralized or decentralized, and it can use various mechanisms, such as search engines, **distributed hash tables** (**DHT**), or **peer-assisted content delivery** (**PACD**)

engines, **distributed hash tables** (**DHT**), or **peer-assisted content delivery** (**PACD**). Indexing is critical in P2P architecture to enable efficient resource discovery and sharing.

Security

Security is the component that ensures the confidentiality, integrity, and availability of the network and its resources. P2P networks can be vulnerable to security threats like malware, phishing, or **denial-of-service** (**DoS**) attacks. Security mechanisms in P2P architecture can include authentication, encryption, firewalls, or reputation-based systems.

Applications

Applications are the software programs that run on top of the P2P architecture and use its resources and services. P2P architecture can support various types of applications, such as file sharing, content delivery, messaging, or gaming. Applications can be developed using various programming languages, platforms, or frameworks, and they can interact with the P2P network using standard protocols or APIs.

In summary, P2P architecture is a decentralized software architecture pattern in which nodes in the network share resources and communicate directly with each other. Its key components are nodes, connections, routing, indexing, security, and applications. These components work together to enable the creation, deployment, and management of P2P networks that can support various types of applications and services.

Components of software architecture

Software architecture consists of various components that work together to achieve the desired functionality and performance of the software. Here are some of the key components of software architecture.

Components

Software architecture is composed of multiple components that make up the system. These components are usually implemented as software modules or objects that represent the functionality and features of the system. Each component has a well-defined interface that defines its inputs, outputs, and behavior, and it can be developed and tested independently of other components. Components can be classified into different types, such as UI components, database components, middleware components, and business logic components.

Relationships

Components are interconnected in a software architecture through various types of relationships. The most common relationships include dependencies, associations, and interfaces. Dependencies represent the relationship where one component relies on another to function properly. Associations represent the relationship where one component is related to another, but they are not dependent on each other. Interfaces represent the communication protocols between components, which define how they interact with each other.

Patterns

Software architecture often involves the use of design patterns, which are reusable solutions to common software design problems. Design patterns can be used to structure the software components and relationships in a way that is consistent with best practices and industry standards. Examples of design patterns include **Model-View-Controller** (**MVC**), layered architecture, and microservices architecture.

Quality attributes

Software architecture defines the quality attributes of the system, such as performance, scalability, reliability, security, and maintainability. These attributes are critical to the success of the system and must be considered throughout the software development life cycle. Quality attributes are often defined in terms of requirements, such as response time, throughput, availability, and fault tolerance.

Constraints

Software architecture must also take into account the constraints imposed by the system's environment, such as hardware, software platforms, network infrastructure, and organizational policies. These constraints can impact the selection of technology, design decisions, and system performance.

Views

Software architecture can be represented through different views or perspectives, such as logical, physical, and process views. Each view provides a different perspective on the system and its components, and it can help stakeholders understand the system's design and behavior. Views can also be used to communicate the system's design to different stakeholders, such as developers, architects, and end users.

Documentation

Software architecture should be documented to provide a clear and concise representation of the system's design and behaviour. Documentation can include diagrams, models, and specifications that describe the system's components, relationships, behaviour, and quality attributes. Documentation can be used to communicate the system's design to different stakeholders, provide guidance to developers, and ensure that the system meets the requirements.

Roles in software architectures

Software architecture requires various roles to be fulfilled to ensure the success of the project. Each role has a unique set of responsibilities, and together, they work to ensure that the software system meets its requirements and is maintainable, scalable, and performant. Here are some of the key roles in software architecture.

Software architect

A software architect is responsible for putting together the overall design of the software system. They work closely with the project stakeholders to understand the requirements of the system and develop a high-level design that meets those requirements. The software architect is also responsible for selecting the appropriate technologies, frameworks, and design patterns to be used in the software system. Additionally, they ensure that the design is scalable, maintainable, and flexible.

Technical lead

A technical lead is responsible for providing technical direction to the project. They work closely with the software architect to ensure that the design is implemented in a way that is consistent with the overall requirements. They are also responsible for mentoring the development team and ensuring that the code is well-organized, maintainable, and performant.

Software developer

A software developer is responsible for implementing the design of the software system. They work closely with the technical lead and the software architect to understand the design and requirements of the system. They are responsible for writing code that is maintainable, scalable, and performant, and they also write unit

tests to ensure that the code is working as expected.

Quality assurance engineer

A quality assurance engineer is responsible for ensuring that the software system meets the desired level of quality. They work closely with the software developers to test the code and ensure that it meets the requirements of the system. They are also responsible for writing test cases and automating the testing process.

DevOps engineer

A DevOps engineer is responsible for the deployment and operation of the software system. They work closely with the software developers to ensure that the code can be deployed in a reliable and automated manner. They are also responsible for monitoring the system and ensuring that it is performing as expected.

Database administrator

A database administrator is responsible for managing the data storage of the software system. They work closely with the software developers to ensure that the data is organized in a way that is consistent with the requirements of the system. They are also responsible for ensuring that the data is secure, available, and backed up.

Technical writer

A technical writer is responsible for creating the documentation for the software system. They work closely with the software developers and the software architect to ensure that the documentation is complete, accurate, and up to date. They are also

responsible for creating user manuals, training materials, and other documentation necessary to support the software system.

Project manager

A project manager is responsible for ensuring that the software development project is completed on time and within budget, and that it meets the desired level of quality. They work closely with the stakeholders to understand the requirements of the system and develop a project plan that meets those requirements. They are also responsible for managing the development team, tracking the progress of a project, and communicating with stakeholders.

Architecture maturity

Architecture maturity is the level of effectiveness and sophistication that an organization has achieved in the practice of software architecture. It is a measure of the organization's ability to produce high-quality software systems that meet the needs of its stakeholders.

There are different models of architecture maturity, but one of the most popular is the **Software Engineering Institute's (SEI) Capability Maturity Model Integration (CMMI)**, which provides a framework for assessing an organization's software development process maturity. This model has five levels of maturity, ranging from an initial ad-hoc level to an optimized level:

Level 1 – Initial

At this level, the organization has an ad-hoc software development process, and there is no formal process in place. The development process is usually reactive and not well understood, and there is a high risk of failure.

Level 2 – Managed

At this level, the organization has established basic project management practices to ensure that software development is under control. There is some level of planning, tracking, and monitoring of the development process. However, the focus is on managing the development process rather than improving it.

Level 3 – Defined

At this level, the organization has a defined software development process that is consistently followed by the development team. There are established procedures and standards in place, and the focus is on continuous improvement of the development process.

Level 4 - Quantitatively managed

At this level, the organization has a well-defined and quantitatively managed software development process. The focus is on process measurement, analysis, and optimization. The organization collects and analyzes data to improve the development process.

Level 5 – Optimizing

At this level, the organization has a continuously improving software development process. The organization focuses on identifying and addressing the root causes of problems, improving the process, and preventing defects from occurring. It also uses

problems, improving the process, and preventing defects from occurring. It also uses process innovation and automation to improve the software development process.

Architecture in the age of agile development

Software architecture plays an essential role in the development of software applications. It provides a blueprint of a system's structure and its components and their interactions. In the context of Agile development, software architecture is a critical component of the development process. Agile development emphasizes iterative and incremental development, continuous delivery, and adaptability to change. To achieve these goals, software architecture must be flexible, adaptable, and responsive to changing requirements.

Agile development is based on the Agile Manifesto, which emphasizes collaboration, flexibility, and rapid response to change. It is an iterative and incremental approach to software development. The Agile development process consists of a series of short iterations or sprints, during which the development team works on a specific set of features or requirements. Each sprint typically lasts for 2-4 weeks, and the team delivers a working software product at the end of each sprint.

In Agile development, the software architecture is not fixed or predetermined. Instead, it evolves as the development process progresses. The Agile development process emphasizes continuous integration and continuous delivery, and the software architecture must support these practices. The software architecture must be designed to be flexible and adaptable to changes in requirements and to support the rapid delivery of new features.

One of the key principles of Agile development is that the development team is responsible for the software architecture. This means that the development team, not just the software architect, has the responsibility of creating and evolving the software architecture. The team should be able to make architectural decisions and have the authority to modify the architecture as needed.

The Agile development process also emphasizes the importance of communication and collaboration among the development team members. The software architecture must be designed to support collaboration and communication among team members. The architecture should be accessible to all team members, and the team should be able to review and discuss it regularly.

In Agile development, the software architecture should be designed to support the development of modular, loosely coupled components. These components should be designed to be independent of each other so that changes in one component do not

affect the others. This design approach helps support the iterative and incremental development process, as changes can be made to individual components without affecting the entire system.

The software architecture in Agile development should also support continuous integration and continuous delivery. Continuous integration is the practice of merging code changes into a shared repository regularly, and continuous delivery is the practice of delivering working software products to customers on a regular basis. The software architecture should be designed to support these practices, with automated testing, continuous build, and deployment processes.

The Agile development process also emphasizes the importance of feedback. Feedback helps the development team identify and correct problems early in the development process. The software architecture should be designed to support feedback by providing a mechanism for monitoring and measuring the performance of the system. The architecture should also support the use of metrics to track progress and identify areas for improvement.

Agile development is a highly collaborative and adaptive approach to software development. The software architecture in Agile development must be designed to support collaboration, flexibility, and rapid response to change. The architecture should be designed to support the development of modular, loosely coupled components; continuous integration; continuous delivery; and feedback. Also, the development team, not just the software architect, should be responsible for creating and evolving the architecture. By following these principles, the software architecture can support the Agile development process and help deliver high-quality software products that meet the needs of customers.

The Open Group has published standards of how Enterprise architecture can be aligned with sprints and scrums; refer to **https://pubs.opengroup.org/togaf-standard/guides/agile-sprints.html.**

Conclusion

It was interesting to know the various aspects of software architecture! Here's a summary of the chapter.

Technology vision and software architecture are two different concepts that are both essential for the success of a software development project. While technology vision provides a high-level direction for an organization's technology investments and decision-making, software architecture provides a detailed plan for the design and implementation of software systems. While technology vision focuses on the long-term goals and outcomes, software architecture focuses on the detailed design and implementation of software systems to achieve those goals.

At the same time, enterprise architecture and software architecture are two different

disciplines that are both essential for the success of modern organizations. While both disciplines are concerned with the design and implementation of complex IT systems, they differ in their scope, goals, and approaches. Enterprise architecture focuses on the overall structure and operation of the organization, while software architecture focuses on the detailed design and development of individual software systems.

Selecting the appropriate software architecture for a given application depends on the specific requirements of the application, including its scalability, flexibility, performance, and maintainability needs. Developers should carefully consider the strengths and weaknesses of each type of architecture before selecting the best one for a project.

Once the software architecture is selected, the various components provide ways to express the architecture, including components, relationships, patterns, quality attributes, constraints, views, and documentation. These components work together to define the structure, behavior, and quality of the software system; they must be carefully considered and documented throughout the software development life cycle.

Software architecture requires a diverse set of roles to be fulfilled to ensure the success of the project. Each role has a unique set of responsibilities, and together, they work to ensure that the software system meets its requirements and is maintainable, scalable, and performant. A good software architecture team will work collaboratively to ensure that the software system meets the needs of the stakeholders and is delivered on time and within budget.

All of software architecture components come together to define the architecture maturity of an organization. An organization with a higher architecture maturity level is likely to have a more effective and efficient software development process. It is also more likely to deliver high-quality software systems that meet the needs of its stakeholders. The organization's ability to adopt new technologies, improve software architecture design, and meet the changing needs of its stakeholders is enhanced as its the maturity level of its architecture improves.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Index

C

D

W

Z

Cloud Architecture Demystified

DESCRIPTION

As more and more businesses move their operations to the cloud, understanding cloud architecture becomes crucial for anyone involved in IT, software development, or data management. If you want to leverage the power of the cloud to deliver efficient and resilient services, then this book is for you.

This book is a comprehensive guide that will help you with the knowledge and insights to successfully navigate the challenges of Agile development and cloud computing. With its practical advice and in-depth analysis, this book offers a deep understanding of key topics such as multi-cloud adoption, cloud deployment costs, security considerations, availability and disaster recovery, and the integration of Agile methodologies with cloud architecture. It also explores the traits of a good cloud solution architect, the importance of treating data and databases separately, and the impact of public cloud on software architecture.

Whether you're a seasoned architect or new to cloud solutions, this book provides valuable guidance for designing robust and effective cloud-based systems.

KEY FEATURES

- Learn how to enable effective architectural decision-making and cloud deployment strategies within the context of Agile DevOps.
- Gain insights into unconventional principles and practices of architecture in the modern era.
- A comprehensive guide for CTOs and technology leaders to navigate the ever-evolving technology landscape.

WHAT YOU WILL LEARN

- Gain insights into assessing various aspects while designing cloud deployments.
- Understand the intersection of Agile methodologies, DevOps practices, and cloud computing.
- Understand the importance of adopting a design-first mindset.
- Understand how Agile principles and practices impact software architecture.
- Discover how architects can effectively drive positive change within organizations.

WHO THIS BOOK IS FOR

The book is for CTOs who are responsible for making strategic decisions regarding cloud adoption and infrastructure. Cloud architects, infrastructure architects, and DevOps architects who are involved in designing and implementing cloud architectures will find this book helpful.

BPB PUBLICATIONS
www.bpbonline.com

ISBN 978-93-5551-826-2