Cloud Native Architectures

LEARNING VOYAGE

Lesson 1. Introducing Cloud Native Architecture

Lesson 1. Introducing Cloud Native Architecture

- The advent of the cloud has led to a new paradigm in designing, implementing, and ongoing maintenance of computer systems.
- While there are many different names for this new paradigm, the one most commonly used is **cloud native architectures.**

What are cloud native architectures?

- Let's start with a generally accepted definition of what cloud computing is according to AWS:

"Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing."

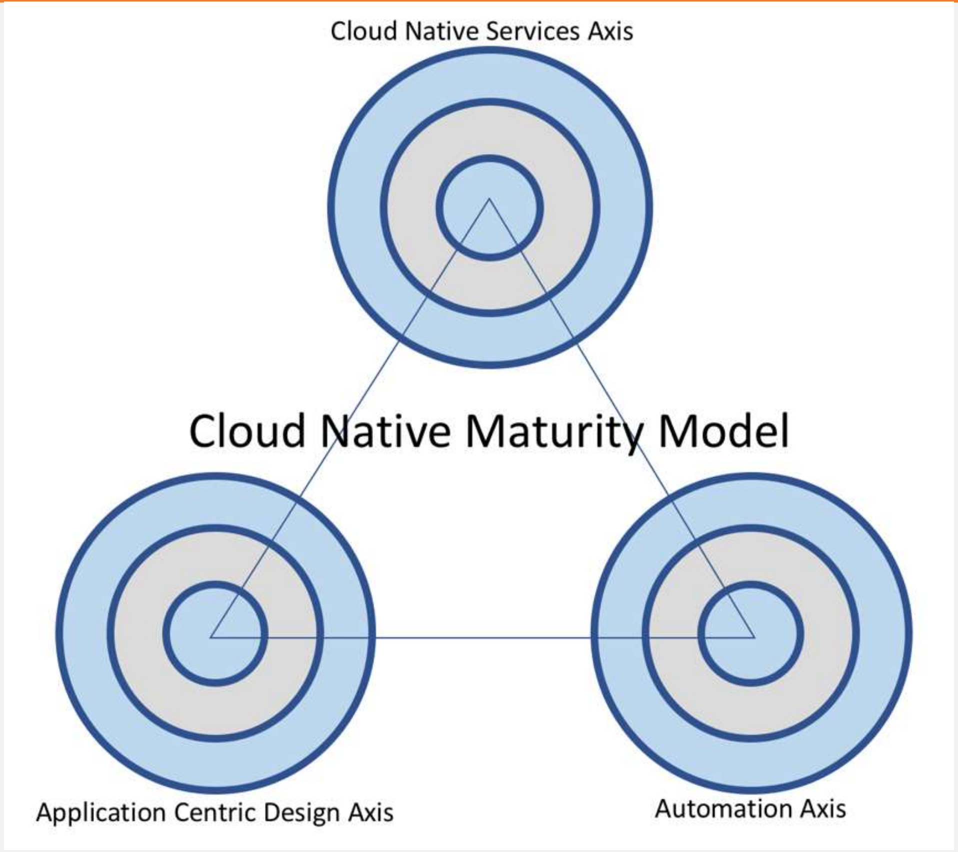
What are cloud native architectures?

- Automation and application design play significant roles in this process as well.
- The cloud, with its API-driven design, allows for extreme automation at scale to not only create instances or specific systems, but to also completely roll out an entire corporate landscape with no human interaction.
- Finally, a critical component in creating a cloud native architecture is the approach used to design a specific application.

Defining the cloud native maturity model

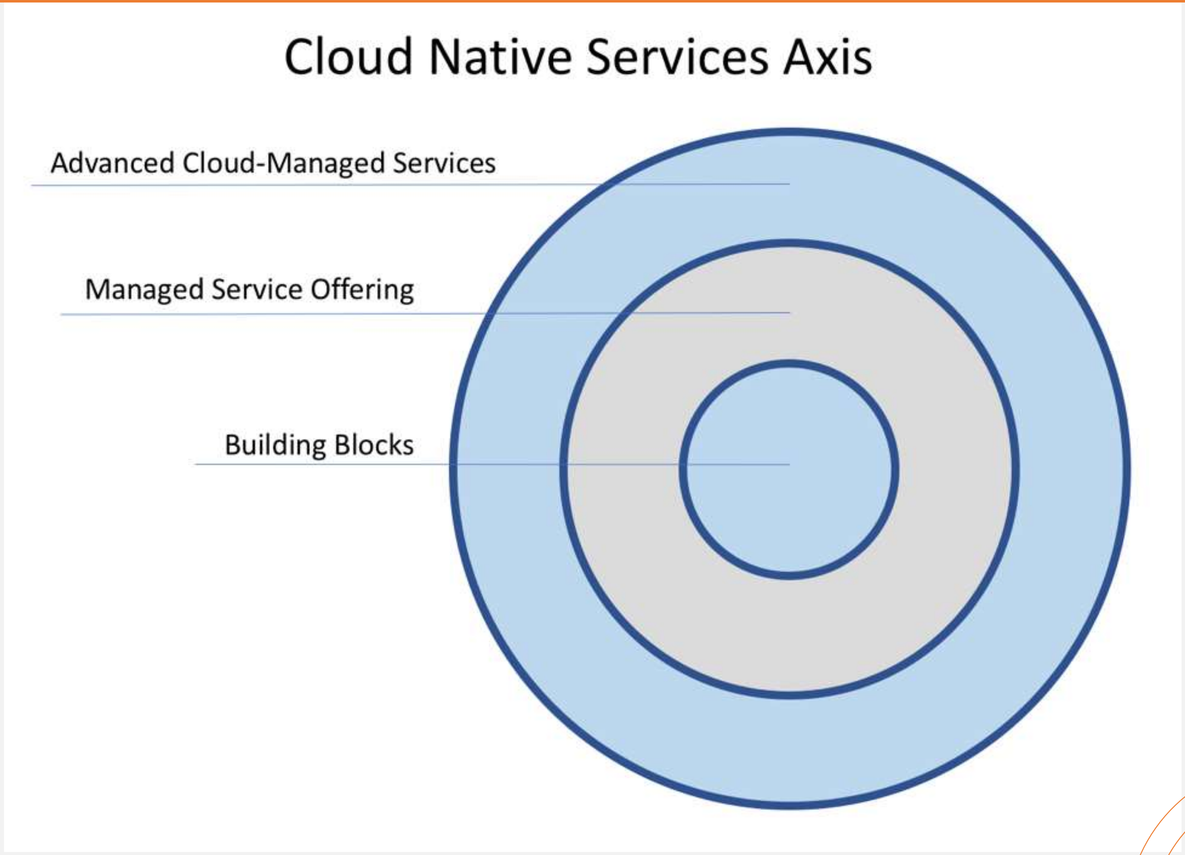
- For the remainder of this course, cloud native architectures will be described using the Cloud Native Maturity Model (CNMM), following(next slide) the design principles outlined, so that architecture patterns can be mapped to their point of evolution:

Defining the cloud native maturity model


Cloud Native Services Axis
Cloud Native Maturity Model
Application Centric Design Axis
Automation Axis

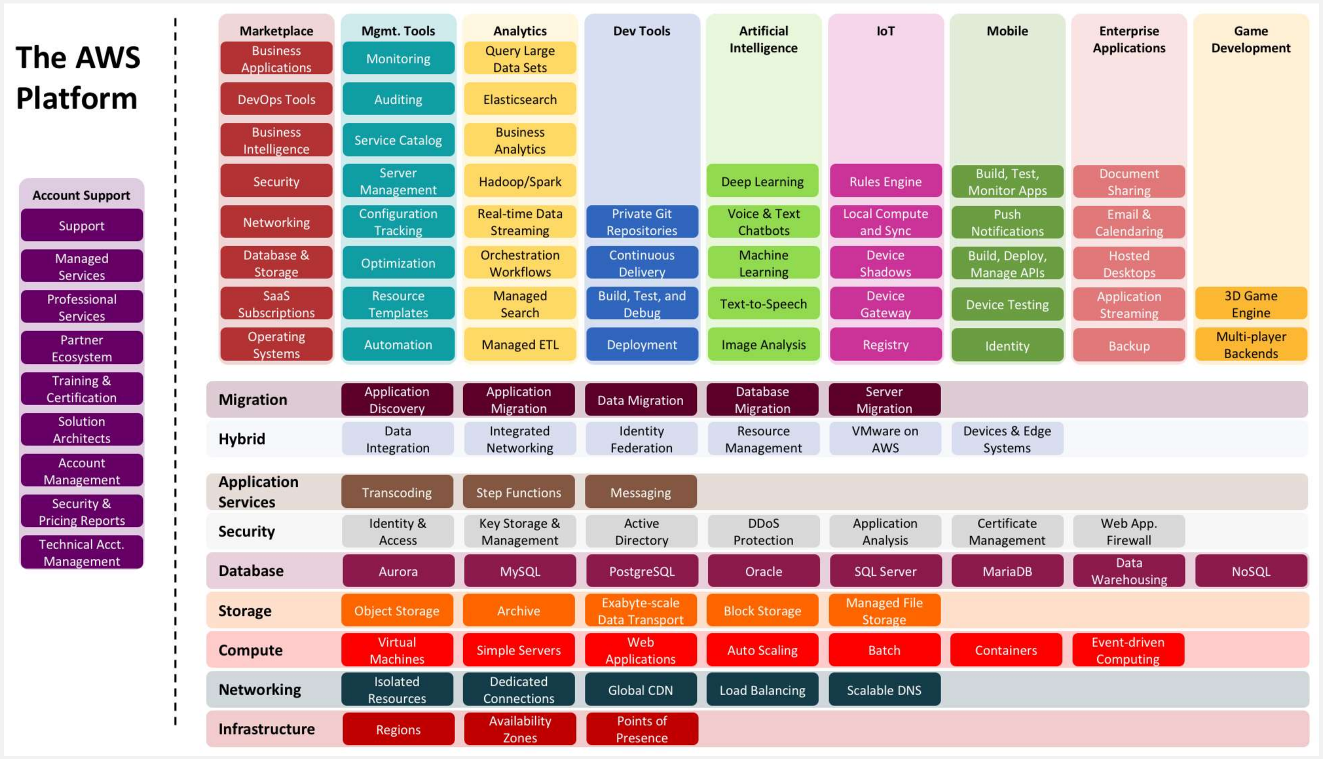
Defining the cloud native maturity model

Axis 1 – Cloud native services



Defining the cloud native maturity model

A mature cloud vendor's services



The AWS Platform

Account Support

- Support
- Managed Services
- Professional Services
- Partner Ecosystem
- Training & Certification
- Solution Architects
- Account Management
- Security & Pricing Reports
- Technical Acct. Management

Marketplace	Mgmt. Tools	Analytics	Dev Tools	Artificial Intelligence	IoT	Mobile	Enterprise Applications	Game Development
Business Applications	Monitoring	Query Large Data Sets						
DevOps Tools	Auditing	Elasticsearch						
Business Intelligence	Service Catalog	Business Analytics						
Security	Server Management	Hadoop/Spark		Deep Learning	Rules Engine	Build, Test, Monitor Apps	Document Sharing	
Networking	Configuration Tracking	Real-time Data Streaming	Private Git Repositories	Voice & Text Chatbots	Local Compute and Sync	Push Notifications	Email & Calendaring	
Database & Storage	Optimization	Orchestration Workflows	Continuous Delivery	Machine Learning	Device Shadows	Build, Deploy, Manage APIs	Hosted Desktops	
SaaS Subscriptions	Resource Templates	Managed Search	Build, Test, and Debug	Text-to-Speech	Device Gateway	Device Testing	Application Streaming	3D Game Engine
Operating Systems	Automation	Managed ETL	Deployment	Image Analysis	Registry	Identity	Backup	Multi-player Backends

Migration	Application Discovery	Application Migration	Data Migration	Database Migration	Server Migration			
Hybrid	Data Integration	Integrated Networking	Identity Federation	Resource Management	VMware on AWS	Devices & Edge Systems		
Application Services	Transcoding	Step Functions	Messaging					
Security	Identity & Access	Key Storage & Management	Active Directory	DDoS Protection	Application Analysis	Certificate Management	Web App. Firewall	
Database	Aurora	MySQL	PostgreSQL	Oracle	SQL Server	MariaDB	Data Warehousing	NoSQL
Storage	Object Storage	Archive	Exabyte-scale Data Transport	Block Storage	Managed File Storage			
Compute	Virtual Machines	Simple Servers	Web Applications	Auto Scaling	Batch	Containers	Event-driven Computing	
Networking	Isolated Resources	Dedicated Connections	Global CDN	Load Balancing	Scalable DNS			
Infrastructure	Regions	Availability Zones	Points of Presence					

Defining the cloud native maturity model

Cloud native services building blocks

- Regardless of the level of maturity of the cloud vendor, they will have the building blocks of infrastructure, which include compute, storage, networking, and monitoring.
- Depending on the cloud maturity level of an organization and the team designing a system, it might be common to begin the cloud native journey by leveraging these baseline infrastructure building blocks.

Defining the cloud native maturity model

- Often, a company will choose to migrate an existing application to the cloud and perform the migration in a lift-and-shift model.
- This approach would literally move the application stack and surrounding components to the cloud with no changes to the design, technology, or component architecture.
- Therefore, these migrations only use the basic building blocks that the cloud offers, since that is what is also in place at the customer-on-premises locations.

Defining the cloud native maturity model

- One of the key outcomes that a company will gain from this maturity stage is the basic premise of the cloud and how that impacts their design patterns:

- for example, horizontal scaling versus vertical scaling, and the price implications of these designs and how to implement them efficiently.

Defining the cloud native maturity model

Cloud vendor managed service offerings

- Undifferentiated heavy lifting is often used to describe when time, effort, resources, or money are deployed to perform tasks that will not add to the bottom line of a company.
- Undifferentiated simply means that there is nothing to distinguish the activity from the way others do it.
- Heavy lifting refers to the difficult task of technology innovation and operations which, if done correctly, nobody ever recognizes, and if done wrong, can cause catastrophic consequences to business operations.

Defining the cloud native maturity model

- Unfortunately, this describes a large majority of the IT operations for enterprise companies, and is a critical selling point to using the cloud.

- Cloud vendors do have a core competency in technology innovation and operations at a scale that most companies could never dream of.

Defining the cloud native maturity model

- Managed service offerings from cloud vendors would include the following:

- ❑ Databases
- ❑ Hadoop
- ❑ Directory services
- ❑ Load balancers
- ❑ Caching systems
- ❑ Data warehouses
- ❑ Code repositories
- ❑ Automation tools
- ❑ Elastic searching tools

Defining the cloud native maturity model

- Another area of importance for these services is the agility they bring to a solution.

- If a system were designed to use these tools but managed by the company operations team, often the process to provision the virtual instance, configure, tune, and secure the package will significantly slow the progress being made by the design team.

Defining the cloud native maturity model

- Using managed service offerings from a cloud vendor doesn't necessarily lead to more advanced architecture patterns; however, it does lead to the ability to think bigger and not be constrained by undifferentiated heavy lifting.
- This concept of not being constrained by limitations that are normally found on-premises, like finite physical resources, is a critical design attribute when creating cloud native architectures, which will enable systems to reach a scale hard to achieve elsewhere.

Defining the cloud native maturity model

Advanced cloud native managed services

According to AWS:
"Serverless computing allows you to build and run applications and services without thinking about servers. Serverless applications don't require you to provision, scale, and manage any servers. You can build them for virtually any type of application or backend service, and everything required to run and scale your application with high availability is handled for you."

Defining the cloud native maturity model

- There are many types of serverless services, including compute, API proxies, storage, databases, message processing, orchestration, analytics, and developer tools.
- One key attribute to define whether a cloud service is serverless or only a managed offering is the way that licensing and usage are priced.
- Serverless leaves behind the old-world model of core-based pricing, which would imply it is tied directly to a server instance, and relies more on consumption-based pricing.

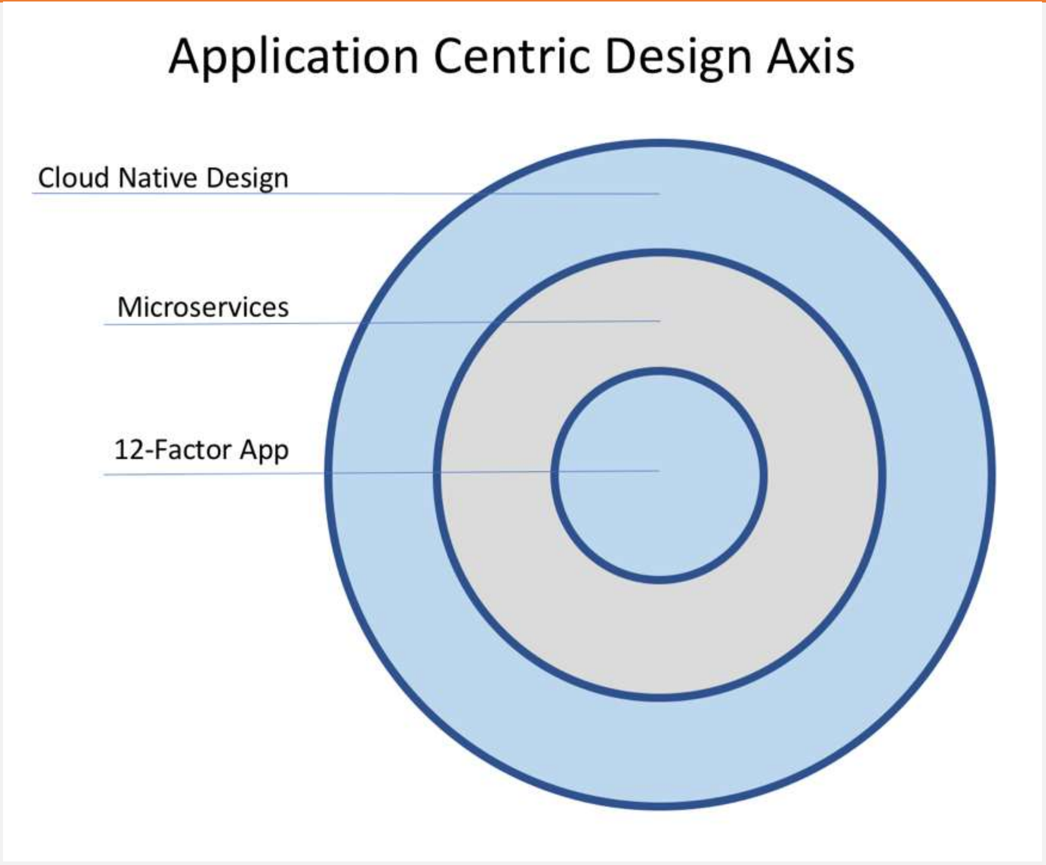
Defining the cloud native maturity model

Cloud native services axis recap

- The Cloud native services axis section described the components that could make up a cloud native architecture, and showed ever more mature approaches to creating applications using them.
- As with all of the design principles on the CNMM, the journey will begin with a baseline understanding of the principle and mature as the design team becomes more and more knowledgeable of how to implement at scale.

Defining the cloud native maturity model

Axis 2 – Application centric design



Defining the cloud native maturity model

Twelve-factor app design principles

- The twelve-factor app is a methodology for building software-as-a-service applications (https://12factor.net/).

- This methodology was written in late 2011, and is often cited as the base building blocks for designing scalable and robust cloud native applications.

Defining the cloud native maturity model

- The twelve factors (https://12factor.net/) are as follows:

Factor No.	Factors	Description
1	Code base	One code base tracked in revision control, many deploys.
2	Dependencies	Explicitly declare and isolate dependencies.
3	Config	Store config in the environment.
4	Backing services	Treat backing services as attached resources.
5	Build, release, run	Strictly separate build and run stages.
6	Processes	Execute the app as one (or more) stateless process(es).
7	Port binding	Export services through port binding.
8	Concurrency	Scale-out through the process model.
9	Disposability	Maximize robustness with fast startup and graceful shutdown.
10	Dev/prod parity	Keep development, staging, and production as similar as possible.
11	Logs	Treat logs as event streams.
12	Admin processes	Run admin/management tasks as one-off processes.

Defining the cloud native maturity model

Monolithic, SOA, and microservices architectures

- Architecture design patterns are always evolving to take advantage of the newest technology innovations.
- For a long time, monolithic architectures were popular, often due to the cost of physical resources and the slow velocity in which applications were developed and deployed.
- These patterns fit well with the workhorse of computing and mainframes, and even today there are plenty of legacy applications running as a monolithic architecture.

Defining the cloud native maturity model

- Service-oriented architectures consist of two or more components which provide their services to other services through specific communication protocols.

- The communication protocols are often referred to as web services, and consist of a few different common ones: WSDL, SOAP, and RESTful HTTP, in addition to messaging protocols like JMS

Defining the cloud native maturity model

- The microservices architecture style takes the distributed nature of SOA and breaks those services up into even more discrete and loosely coupled application functions.

- Microservices not only reduce blast radius by even further isolating functions, but they also dramatically increase the velocity of application deployments by treating each microservice function as its own component

Defining the cloud native maturity model

Cloud native design considerations

- Regardless of the methodology used or the final cloud native design pattern implemented, there are specific design considerations that all cloud native architectures should attempt to implement.
- While not all of these are required to be considered a cloud native architecture, as these considerations are implemented, the maturity of the system increases, and will fall on a higher level of the CNMM.

Defining the cloud native maturity model

These considerations include instrumentation, security, parallelization, resiliency, event-driven, and future-proofed:

- Instrumentation
- Security
- Parallelization
- Resiliency
- Event-driven
- Future-proofed

Defining the cloud native maturity model

Application centric design axis recap

- There are many different methodologies that can be employed to create a cloud native application, including microservices, twelve-factor app design patterns, and cloud native design considerations.

- There is no one correct path for designing cloud native applications, and as with all parts of the CNMM, maturity will increase as more robust considerations are applied.

Defining the cloud native maturity model

Axis 3 – Automation

- The third and final cloud native principle is about automation.
- Throughout this lesson, the other CNMM principles have been discussed in detail and explained, particularly why using cloud native services and application centric design enable cloud native architectures to achieve scale.

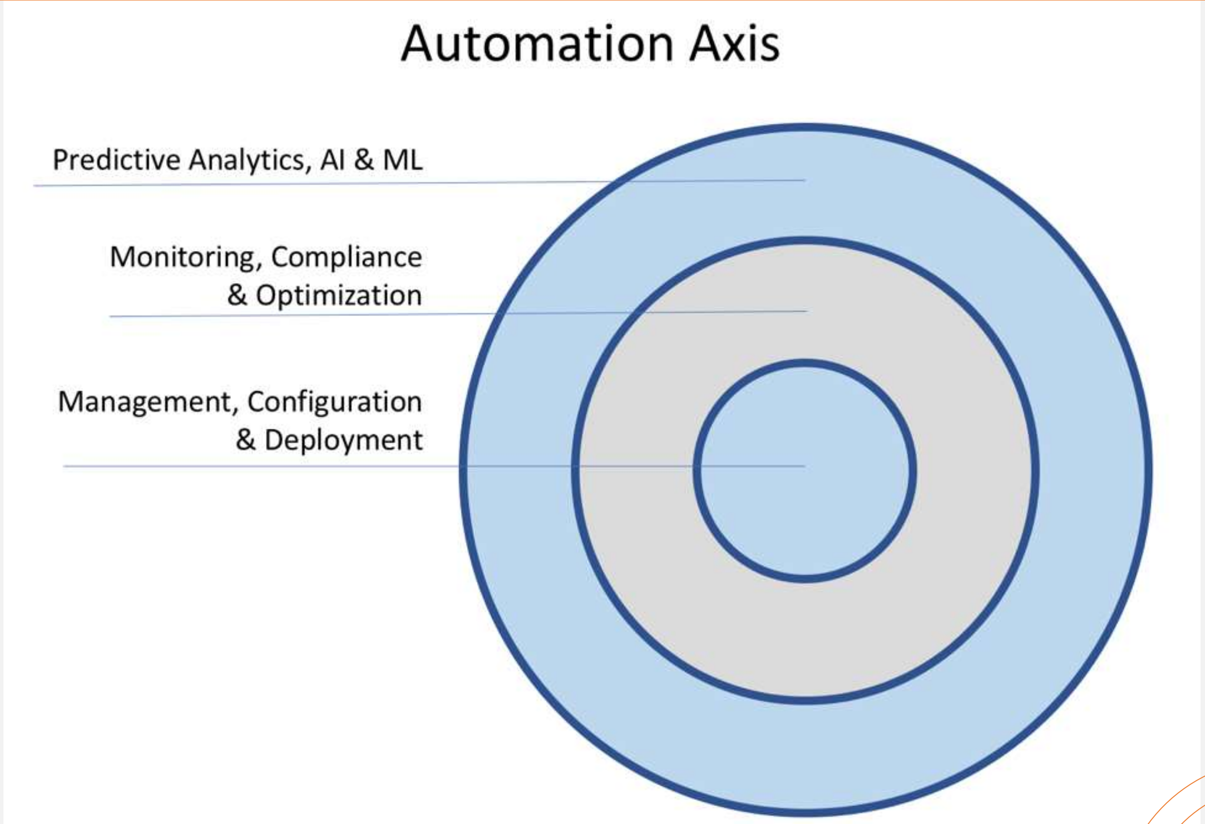
Defining the cloud native maturity model

- With Infrastructure as Code automation, operations teams can now focus on the application-specific design and rely on the cloud vendor to handle the undifferentiated heavy lifting of resource deployment.

- This Infrastructure as Code is then treated like any other deployment artifact for the application, and is stored in source code repositories, versioned and maintained for long-term consistency of environment buildouts.

Defining the cloud native maturity model

- Automation is the key to achieving the scale and security required by cloud native architectures:


Automation Axis
Predictive Analytics, AI & ML
Monitoring, Compliance & Optimization
Management, Configuration & Deployment

Defining the cloud native maturity model

Environment management, configuration, and deployment

- Designing, deploying, and managing an application in the cloud is complicated, but all systems need to be set up and configured.
- In the cloud, this process can become more streamlined and consistent by developing the environment and configuration process with code.
- There are potentially lots of cloud services and resources involved that go well beyond traditional servers, subnets, and physical equipment being managed on-premises.

Defining the cloud native maturity model

- There are multiple schools of thought on how to handle the operations of a system using Infrastructure as Code.

- In some cases, every time an environment change occurs, the full suite of Infrastructure as Code automation is executed to replace the existing environment.

- This is referred to as immutable infrastructure, since the system components are never updated, but replaced with the new version or configuration every time.

Defining the cloud native maturity model

- Environment management and configuration is not the only way automation is required at this baseline level.
- Code deployments and elasticity are also very important components to ensuring a fully automated cloud native architecture.
- There are numerous tools on the market that allow for the full deployment pipeline being automated, often referred to as continuous integration, continuous deployment (CICD).

Defining the cloud native maturity model

Monitoring, compliance, and optimization through automation

- Cloud native architectures that use complex services and span across multiple geographic locations require the ability to change often based on usage patterns and other factors.
- Using automation to monitor the full span of the solution, ensure compliance with company or regulatory standards, and continuously optimize resource usage shows an increasing level of maturity

Defining the cloud native maturity model

- Automated compliance of environment and system configurations is becoming increasingly critical for large enterprise customers.
- Incorporating automation to perform constant compliance audit checks across system components shows a high level of maturity on the automation axis.
- These configuration snapshot services allow a complete picture in time of the full environmental makeup, and are stored as text for long-term analysis.

Defining the cloud native maturity model

- With the cloud, those challenges all but disappear; however, system designers still run into situations where they don't know what capacity is needed.

- To resolve this, automated optimization can be used to constantly check all system components and, using historical trends, understand if those resources are over-or under-utilized.

Defining the cloud native maturity model

Predictive analytics, artificial intelligence, machine learning, and beyond

- As a system evolves and moves further up the automation maturity model, it will rely more and more on the data it generates to analyze and act upon.
- Similar to the monitoring, compliance, and optimization design from the previous part of the axis, a mature cloud native architecture will constantly be analyzing log event streams to detect anomalies and inefficiencies

Defining the cloud native maturity model

- Using the automation building blocks already discussed from this axis in combination with the AI and ML, the system has many options to deal with a potential business impacting event.
- Data is king when it comes to predictive analytics and machine learning. The never-ending process of teaching a system how to categorize events takes time, data, and automation.

Defining the cloud native maturity model

- There are many examples of how using ML on datasets will show correlation that could not be seen by a human reviewing the same datasets—like how often a failed user login resulted in a lockout versus a retry over millions of different attempts, and if those lockouts were the result of a harmless user forgetting a password, or a brute-force attack to gain system entry.
- Because the algorithm can search all required datasets and correlate the results, it will be able to identify patterns of when an event is harmless or malicious.

Defining the cloud native maturity model

Automation axis recap

- Automation unlocks significant value when implemented for a cloud native architecture.

- The maturity level of automation will evolve from simply setting up environments and configuring components to performing advanced monitoring, compliance, and optimization across the solution.

The cloud native journey

- Companies large and small, new and mature, are seeing the benefits of cloud computing.
- There are many paths to get to the cloud, and that often depends on the maturity of the organization and the willingness of senior management to enact the change required.
- Regardless of the type of organization, the shift to cloud computing is a journey that will take time, effort, and persistence to be successful.

The cloud native journey

The decision to be cloud-first

- Cloud computing is here to stay. Years ago there were many discussions on whether a company should declare a cloud-first model, or not chase the latest and greatest technologies;

- however, at this point in time, just about every company has taken the first step towards cloud computing, and many have made the decision to be a cloud-first organization.

The cloud native journey

People and process changes in the cloud

- Organizations with large IT departments or long-term outsourced contracts will inherently have a workforce that is skilled at the technologies that have been running in the company up until that point.
- Making the shift to any new technology, especially cloud computing, will require a significant amount of retooling, personnel shifting, and a change in the thought pattern of the workforce

The cloud native journey

- One specific area that can often be difficult for experienced IT professionals to overcome, especially if they have gained their experience with data center deployments and lots of large legacy workloads, is the concept of unlimited resources.
- Since most cloud vendors have effectively unlimited resources to be consumed, removing that constraint on application design will open up a lot of unique and innovative ways to solve problems that were impossible when designing applications before.

The cloud native journey

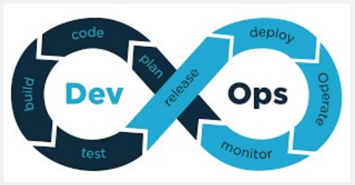
- Processes are also a big hurdle for being a cloud-first organization. Lots of companies that are transitioning to the cloud phase are also in transitioning from the SOA to microservices phase.
- Therefore, it would be common for the processes in place to be supportive of SOA architectures and deployments, which are most likely there to slow things down and ensure that the big bang deployments to the composite application are done correctly and with significant testing

The cloud native journey

Agile and DevOps

- The cloud is not a magic place where problems go away. It is a place where some of the traditional challenges go away; however, new challenges will come up.
- Legacy enterprise companies have been making the switch from waterfall project management to agile for a while now.
- That is good news for a company intending to be cloud native, since iteration, failing fast, and innovation are critical to long-term success, and agile projects allow for that type of project delivery.

The cloud native journey



- DevOps (or the merging of development teams and operations teams) is a new IT operating model that helps bridge the gap between how code is developed and how it is operated once deployed to production.

- Making a single team accountable for the code logic, testing, deployment artifacts, and the operations of the system will ensure that nothing is lost in the life cycle of a code development process

The cloud native journey

Cloud operating environment

- The journey to the cloud will take time and lots of trial and error. Typically, a company will have identified a primary cloud vendor for their requirements and, in some cases, they will have a second cloud vendor for specific requirements.
- In addition, almost all companies begin with a hybrid architecture approach, which allows them to leverage their existing investments and applications while moving workloads into their chosen cloud.

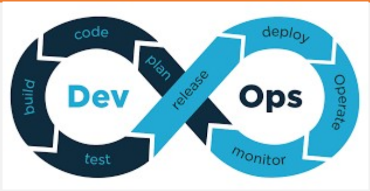
The cloud native journey

Cloud operating foundation

- The cloud is a vast set of resources that can be used to solve all kinds of business problems; however, it is also a complex set of technologies that requires not only skillful people to use it, but also a strict operating foundation to ensure it is being done securely and with cost and scale in mind.
- Even before a single workload is deployed to the cloud, it is important for a company to fully identify their expected foundational design.

The cloud native journey

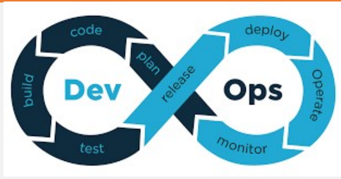
Hybrid cloud



- In addition to the foundation of the cloud platform, a company must decide how to leverage its existing assets. While the value proposition of cloud computing is not debated much anymore, the pace of migration and how fast to deprecate existing assets is.
- Using a hybrid cloud approach for the beginning of the cloud native journey is very common, and lets the company easily operate with its two existing groups (the legacy group and the cloud-first group).

The cloud native journey

Typical patterns for a hybrid cloud architecture are:



- Legacy workloads on-premises and new projects in the cloud
- Production workloads on-premises and non-production in the cloud
- Disaster recovery environment in the cloud
- Storage or archival in the cloud
- On-premises workloads bursting into the cloud for additional capacity

The cloud native journey

Multi-cloud

- Enterprise companies need to ensure their risk is spread out so that they reduce the blast radius in the event of an issue, whether this be a natural disaster, security event, or just making sure that they are covering their customers in all of the locations they operate in.
- Therefore, the allure of a multi-cloud environment is strong, and some larger organizations are starting to go down this path for their cloud journey.

The cloud native journey

- Another common approach to multi-cloud is the use of containers for moving workloads between clouds.
- In theory, this approach works and solves a lot of the challenges that multi-cloud poses.
- There is currently a lot of innovation going on with this approach and the ability to be successful with moving containers between clouds is still in its infancy.

The cloud native journey

Application migration at scale

- A cloud native company will have the goal of reducing their self-managed data centers and workloads and shifting those as much as possible to the cloud, There are three main paths this can present:

1. Lift-and-shift migration of legacy workloads to the cloud
2. Re-engineering of legacy workloads to optimize in the cloud
3. Greenfield cloud native development

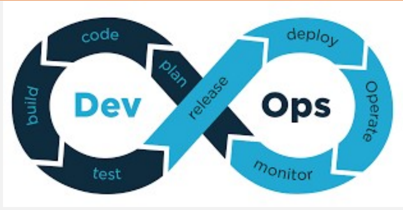
The cloud native journey

Lift-and-shift migration

- Lift-and-shift migration is the act of moving existing workloads, as is, to the target cloud-operating foundation already implemented.
- This type of exercise is usually done against a grouping of applications, by business unit, technology stack, or complexity level of some other type of metric.
- A lift-and-shift migration in its purest form is literally bit-by-bit copies of existing instances, databases, storage, and so on, and is actually rare, since the cost benefits of doing this to the cloud would be negligible

The cloud native journey

Re-engineer migration



- Companies that are truly moving to be a cloud native organization will most likely choose to re-engineer most of their legacy workloads so that they can take advantage of the scale and innovation that the cloud has to offer.
- Workloads that are chosen to be migrated to the cloud but re-engineered in the process might take longer to move, but once completed they will fall on some part of the CNMM and be considered cloud native

The cloud native journey

Cloud native companies

- While technically not a migration, cloud native companies that are creating new applications will choose to go through the entire development cycle with a cloud native architecture in mind.

- Even workloads that are re-engineered might not be able to fully change their underlying technologies for whatever reason.

Cloud native architecture case study – Netflix

The journey

The following quote has been extracted from the press release that both companies published at that time

"Amazon Web Services today announced that Netflix, Inc., has chosen AWS to run a variety of mission-critical, customer-facing and backend applications. While letting Amazon Web Services do the worrying about the technology infrastructure, Netflix continues to improve its members' overall experience of instantly watching TV episodes and movies on the TV and computer, and receiving DVDs by mail."

Cloud native architecture case study – Netflix

- 2009: Migrating video master content system logs into AWS S3
- 2010: DRM, CDN Routing, Web Signup, Search, Moving Choosing, Metadata, Device Management, and more were migrated into AWS
- 2011: Customer Service, International Lookup, Call Logs, and Customer Service analytics
- 2012: Search Pages, E-C, and Your Account
- 2013: Big Data and Analytics
- 2016: Billing and Payments

DEVOPS

Cloud native architecture case study – Netflix

The benefits

- The journey to becoming a cloud native company at Netflix was impressive, and continues to yield benefits for Netflix and its customers.

- The growth Netflix was enjoying around 2010 and beyond made it difficult for them to logistically keep up with the demand for additional hardware and the capacity to run and scale their systems.

Cloud native architecture case study – Netflix

CNMM

- Now that we understand what the Netflix journey was about and how they benefited from that journey, this section will use the CNMM to evaluate how that journey unfolded and where they stand on the maturity model.

- Since they have been most vocal about the work they did to migrate their billing and payment system to AWS, that is the workload that will be used for this evaluation.

Cloud native architecture case study – Netflix

Cloud native services axis

- The focus of the cloud native services adoption spectrum is to demonstrate the amount of cloud vendor services that are in use for the architecture.

- While the full extent of the services that Netflix uses is unknown, they have publicly disclosed numerous AWS services that help them achieve their architecture.

Cloud native architecture case study – Netflix

"Essentially everything before you hit "play" happens in AWS, including all of the logic of the application interface, the content discovery and selection experience, recommendation algorithms, transcoding, etc.; we use AWS for these applications because the need for this type of computing is not unique to Netflix and we can take advantage of the ease of use and growing commoditization of the "cloud" market. Everything after you hit "play" is unique to Netflix, and our growing need for scale in this area presented the opportunity to create greater efficiency for our content delivery and for the internet in general."

Cloud native architecture case study – Netflix

Application centric design axis

- Designing an application for the cloud is arguably the most complicated part of the journey, and having a high level of maturity on the application centric design axis will require specific approaches.

- Netflix faced some big challenges during its billing and payment system migration to the cloud; specifically, they wanted near-zero downtime, massive scalability, SOX compliance, and global rollout

Cloud native architecture case study – Netflix

To quote their blog on this topic:

"We started chipping away existing code into smaller, efficient modules and first moved some critical dependencies to run from the Cloud. We moved our tax solution to the Cloud first. Next, we retired serving member billing history from giant tables that were part of many different code paths. We built a new application to capture billing events, migrated only necessary data into our new Cassandra data store and started serving billing history, globally, from the Cloud. We spent a good amount of time writing a data migration tool that would transform member billing attributes spread across many tables in Oracle into a much simpler Cassandra data structure. We worked with our DVD engineering counterparts to further simplify our integration and got rid of obsolete code."

Cloud native architecture case study – Netflix

Automation axis

"...our philosophy when we built Chaos Monkey, a tool that randomly disables our production instances to make sure we can survive this common type of failure without any customer impact. The name comes from the idea of unleashing a wild monkey with a weapon in your data center (or cloud region) to randomly shoot down instances and chew through cables — all the while we continue serving our customers without interruption. By running Chaos Monkey in the middle of a business day, in a carefully monitored environment with engineers standing by to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them. So next time an instance fails at 3 am on a Sunday, we won't even notice."

Cloud native architecture case study – Netflix

The full suite of the Simian Army is:

- Latency Monkey: Induces artificial delays into RESTful calls to similar service degradation
- Conformity Monkey: Finds instances that do not adhere to predefined best practices and shuts them down
- Doctor Monkey: Taps into health checks that run on an instance to monitor external signs of health
- Janitor Monkey: Searches for unused resources and disposes of them
- Security Monkey: Finds security violations and vulnerabilities and terminates offending instances
- 10-18 Monkey: Detects configuration and runtime problems in specific geographic regions
- Chaos Gorilla: Similar to Chaos Monkey, but simulates an entire outage of AWS available zones

Summary

- In this lesson, we defined exactly what a cloud native is and what areas of focus are required to develop a mature cloud native architecture.
- Using a CNMM, we identified that all architectures will have three design principles: cloud services adoption, degree of automation, and application-centric design.
- These principles are used to gauge the maturity of the components of the architecture, as it relates to them and where they fall on their own spectrum.

Lesson 2. The Cloud Adoption Journey

LEARNING VOYAGE

Lesson 2. The Cloud Adoption Journey

- Becoming a cloud native company is undeniably a journey, and one that is not only focused on technology.

- As demonstrated by the Netflix case study, this journey can take a very long time to do correctly and will often be full of tough decisions, including those relating to technology and business trade-offs. In addition, this journey is never-ending.

Cloud adoption drivers

- Adopting the cloud is not an accident, but a decision that has to be made.

- That decision, however, is the starting point of a long journey that will involve lots of manpower, processes, and technology changes.

- There are many reasons why an organization will make this decision, with agility and cost factors often being very high on the list.

Cloud adoption drivers

Moving fast and constraining costs

- Before the advent of cloud computing, when a system was being designed, the team would have to estimate the performance requirements and then procure hardware resources to match those estimations, an expensive and slow process often undertaken with limited data.
- Not only is this a bad pricing decision, but it also means that unused capacity is sitting idle in data centers.

Cloud adoption drivers

Agility

- Often cited as the top driver for cloud adoption is agility.
- A common issue facing companies for decades has been long lead times in deploying hardware to their data centers, causing all kinds of issues, including the expansion of project timelines.
- Removing this constraint by having virtually limitless capacity and providing the ability to use that capacity within minutes is critical for an organization with high business velocity requirements

Cloud adoption drivers

Cost

- Failing fast is important, and so is the ability to constrain costs in so doing.
- If the cost of experimentation is high, then failure becomes something that will have an impact on the company.
- As soon as a project or system has a meaningful impact on the company, checks and balances will be implemented that are designed to slow it down to prevent an expensive decision from being made with little chance of a good return on investment, such as procuring expensive hardware.
- It's not as if this concept of business agility is new, owing to the cloud; it has been around for decades

Cloud adoption drivers

- Cost drivers don't only stop with using minimal deployable resources for a new system.
- Most cloud vendors have innovated in the way the resources are billed to the customer.
- It's common for the largest vendors to have very fine-grained billing mechanisms, sometimes charging by the second or even faster, with aggregated monthly bills.
- Gone are the days of a vendor requiring the company to buy the resources and then charging a substantial fee for ongoing maintenance.

Cloud adoption drivers

Being secure and maintaining proper governance

- For a long time, organizations of all sizes had to run their own data centers to be able to control the security and governance of their data and workloads.
- Cloud computing, however, has finally delivered an option that provides higher security and governance than most organizations can achieve on their own.

Cloud adoption drivers

Security

- Every organization is different, with different security requirements, different governance needs, and different ways they believe their company assets should be managed and secured.

- Cloud vendors have staked their business and reputation on the fact that they provide their services in the most secure way possible.

Cloud adoption drivers

- While there are probably companies or organizations that can claim a similar level of focus on this important topic, the vast majority cannot deploy the resources required to continuously innovate their security like cloud vendors can.

- Furthermore, the same argument for using the cloud applies to the security sub-topic as well, and that is, even if you could match the innovations of a cloud vendor, is that really the best use of an organizations resources, or should they be focused on business results?

Cloud adoption drivers

Governance

- Operating on the premise that the cloud is secure is one thing, but being able to actually take advantage of security to ensure a robust posture is another.
- Cloud vendors have lots of services designed to meet just about every security requirement.
- However, it is up to the individual systems to implement those services according to their design specifications.

Cloud adoption drivers

Company expansion

- Company growth is commonly a top priority for most organizations, in addition to serving existing customers, gaining market share, and exploring new business channels.
- This expansion can quickly become a major expense to the company if the need to deploy more data centers arises.
- Setting up a data center is not something that a company will do unless they have a strong business case for their return on that investment, and, even then, a significant capital outlay will be required.

Cloud adoption drivers

Attracting and retaining talent

- Companies that are moving to the cloud require talented people who understand the technology and the new processes that accompany that change.
- In a lot of cases, a company will seek to retain their existing workforce and focus on re-tooling them to be successful on their cloud journey.
- This process is advantageous for many reasons, the main one of which is the fact that all of the institutional knowledge that these individuals possess stays within the company.

Cloud adoption drivers

Cloud innovation and economies of scale

- Allowing major cloud vendors to innovate as fast as possible and then taking advantage of that innovation while focusing resources on the core competencies of a company is the best path to success for a cloud journey.
- There are very few companies that can claim managing data centers as their core competency and, therefore, any decision made to go cloud native must consider this as a driving principle.

The cloud operating model

- There are many organizational change theories that can be applied to how the cloud journey is approached, and, similar to any large transformational undertaking, it will take stakeholders across the whole organization to participate.
- One organizational change pattern that aligns well to the cloud journey is Dr. Kotter's theory (https://www.kotterinternational.com/8-steps-process-for-leading-change/).

The cloud operating model

Kotter has identified eight steps that lead to successful, top-down change management across enterprises. These steps are as follows:

1. Create a sense of urgency
2. Build a guiding coalition
3. Form a strategic vision and initiatives
4. Enlist a volunteer army
5. Enable action by removing barriers
6. Generate short-term wins
7. Sustain acceleration
8. Institute change

The cloud operating model

Stakeholders

- The journey to the cloud is not just about what new technology will be implemented; it's about business change agility and all of the other drivers outlined in the beginning of this lesson.
- Therefore, the list of stakeholders that will be impacted, and who should therefore be included in all aspects of the journey, can be long.
- In almost all cases, the full C-suite will be involved in some way, since this change is transformative to the entire company

The cloud operating model

- In many organizations, the IT department is made up of a few important functions, most notably, the CIO or VP of corporate systems, the CTO or VP of business applications, the VP of end user computing or support, the VP of infrastructure, and a CISO, whose remit encompasses all existing functions.
- The cloud doesn't remove these roles, although, in some cases, it does change their functions.

The cloud operating model

- The business is frequently talked about as a key participant in this journey, but what exactly does that mean? In this case, the business is the part of the organization that drives revenue and owns the product or services on behalf of the company.

- For large multinational corporations, the business may actually be separate subsidiaries with many sub-business groups that work toward their specific requirements

The cloud operating model

Change and project management

- As an organization undertakes its journey to the cloud, it must consider the processes that will be impacted, specifically change management and project management.
- These two important aspects to an IT operating model must be considered with the journey, and changes are usually required to existing policies on how they will be achieved in the cloud.

The cloud operating model

- The cloud doesn't remove the need to have strict change management processes.
- What it does, however, is change how these are implemented in ways that can actually increase their effectiveness, while removing a lot of the bureaucracy that caused the slowdowns to begin with.
- Cloud native architectures, by their very nature, are less tightly coupled and more services-based, thus removing large big-bang deployments and speeding up the delivery process.
- Moving to the cloud can remove not only technical debt in the form of obsolete code, but also in removing obsolete processes.

The cloud operating model

Change management

- Cloud native application design patterns, specifically those made up of many smaller services, are not the only reason why change management is happening faster; automation and containerization are also key contributors.

- This is not something that is unique to the cloud.

The cloud operating model

Project management

- In addition to the change management processes, the approach to project management will need to evolve with the shift to the cloud.
- The waterfall methodology, where the full requirements are identified in advance, and these requirements are then developed in sequence, with testing cycles happening at the end, is too slow for the business agility required.

The cloud operating model

- When changing the project management processes to account for cloud first design principles, one important impact would be to include best practices for cloud native development.

- Including quality assurance checks in the design processes that identify best practices for designing scalable, secure, and cloud native architectures will ensure that workloads are deployed cloud-scale ready, and will need less remediation once they are in production.

The cloud operating model

Risk, compliance, and quality assurance

- The risk profile of a company is higher as it moves to the cloud, at least initially. This is not because the cloud is a risky place, quite the opposite; it's because there are new ways for problems to occur that a company must understand.

- Once company assets (data, and code, for example) are migrated to compute resources located outside of its direct control, there must be additional considerations regarding how to keep those resources secure.

The cloud operating model

- All of these technical aspects are often very different when comparing on-premises with the cloud and it is critical for companies to not only adjust their approach in the cloud, but also to adjust their on-premises approach to better align to the cloud.

- As with any journey, the cloud native goal is to transform the way the company consumes IT resources and remove on-premises limitations as to how applications are designed

The cloud operating model

Risk and compliance

- Risk is inevitable, whenever a company stores data, processes transactions, or interacts with their customers.
- They are dealing with risky situations since there will always be bad actors that want to steal or manipulate the data, transactions, or interactions.
- But bad actors are not the only risk that a company faces; compute resource faults or software bugs are also a very real possibility that must be managed appropriately by any company.

The cloud operating model

- Compliance is not only an external concern; often, a company will have a strict internal governance model that requires specific controls to be in place for certain types of data and workloads.
- In these cases, the cloud provider will not be able to implement those requirements since they are specific to that organization.
- However, there are often ways to map these internal controls to other certifications with similar requirements to show compliance.

The cloud operating model

Quality assurance and auditing

- Once the company's compliance and risk mitigation activities have been solidified, the guardrails are in place to allow for cloud deployments.
- However, these guardrails and governance methodologies are only as good as when they were developed and implemented.
- Over time, as workloads are built and deployed to the cloud, a company has to re-evaluate the risk posture they originally implemented and ensure that it keeps up with the current climate

The cloud operating model

- Deploying new business functionality doesn't mean that the security posture has to change, so performing continuous auditing of the guardrails will ensure that the drift from the original posture isn't excessive.
- Cloud vendors often have configuration services that take periodic views of the overall cloud landscape and store them in a digital format, perfect for auditing and comparison testing.
- These configuration services can usually be extended to not only get a view of the cloud landscape, but also to perform custom system-level verifications and store output alongside the cloud configuration.

The cloud operating model

Foundational cloud operating frameworks and landing zones

- Not knowing where to start to ensure a secure and operational base environment is a common issue that companies have when beginning the cloud native journey.
- Even if all of the people, processes, and technology considerations are aligned, this lack of a landing zone can cause slowdowns or even failure during the early stages of cloud native activities.
- There are many different aspects that go into creating the base landing zone for future cloud workloads, and this section will cover some of these and how to consider the right approach for individual organizations.

The cloud operating model

Cloud landing zone

- A cloud landing zone is all of the technical and operational aspects that need to be considered and designed before any workloads are actually deployed.
- There are a range of areas that fall into this category and not all are required for a successful cloud journey.
- However, as a best practice, it is advisable to take all of them into consideration.

The cloud operating model

Account structure design

- The account structure design strategy is the start of it all, and each company will have different requirements for how they will want to set up their accounts.

- For smaller companies that will have just a few workloads running in the cloud, a single account may be sufficient.

The cloud operating model

Network design

- Once the accounts are designed and ready, network design is the next most critical component.
- The accounts are mostly administrative constructs, but the ability to deploy workloads and secure data is done by the network design.
- Most cloud vendors will have a virtual networking concept that allows for a custom design of the network space needed for various requirements.

The cloud operating model

Central shared services

- Once the accounts and virtual networks are identified and mapped out, the next major focus area concerns what will run as a central shared service.
- Some services are going to be standard across the entire cloud estate, and trying to replicate them in different accounts or virtual networks is counterproductive or even risky.

The cloud operating model

Security and audit requirements

- Finally, security and audit requirements should be pervasive across the cloud estate.
- The entire account, network, and shared services concepts will be underpinned by a holistic security and audit ability design strategy.
- As discussed in the central shared services section, directory and authentication services are important for securing access to workloads, data, and accounts.

The cloud operating model

External governance guidelines

- There are many different governance guidelines developed or evolved to support secure and compliant operating environments in the cloud.

- Depending on the industry vertical, critical data requirements, or external compliance demands, companies have options on how to enhance their base cloud landing zone.

Cloud migration versus greenfield development

Migration patterns

- Migrations are often broken down into patterns using The 6Rs (Rehost, Replatform, Repurchase, Refactor, Retire, and Retain).
- Each of these 6Rs has a place in the cloud native discussion. However, retire, retain, and repurchase are not as applicable for the cloud native discussion.
- Often, the decision to retire or retain an application workload is specific to the complexity or the expected life remaining until a new replacement can be implemented.

Cloud migration versus greenfield development

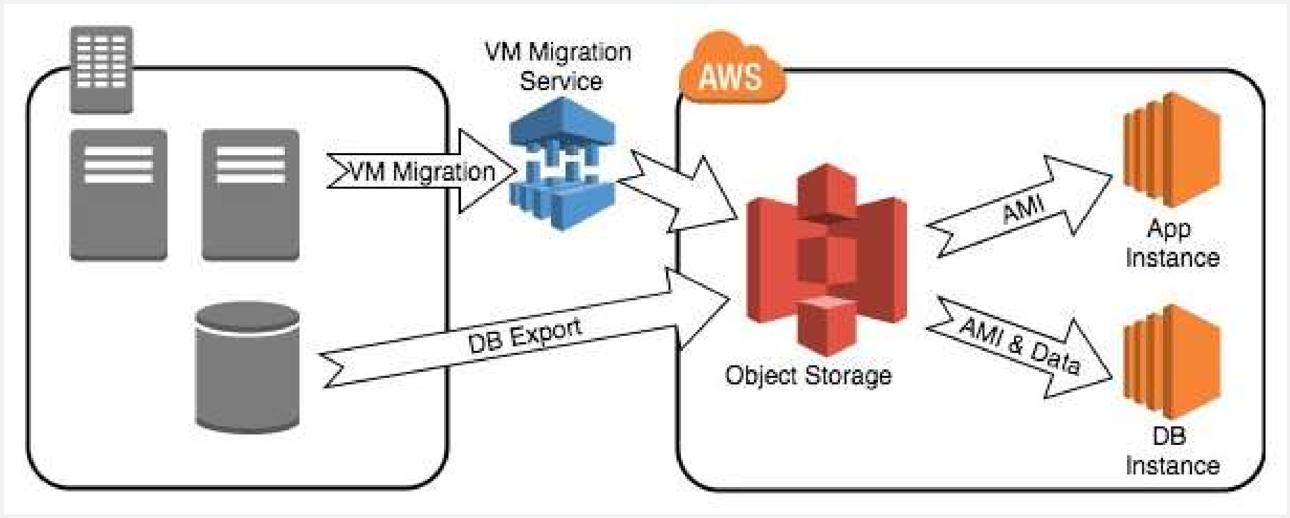
Rehost

- Rehost, often referred to as lift-and-shift, is the quickest path to the cloud and, in its purest form, literally means moving a workload exactly as is from its existing location (usually on-premises) to the cloud.

- This migration pattern is often chosen because it allows for relatively small amounts of upfront analysis work and provides a quick return on investment.

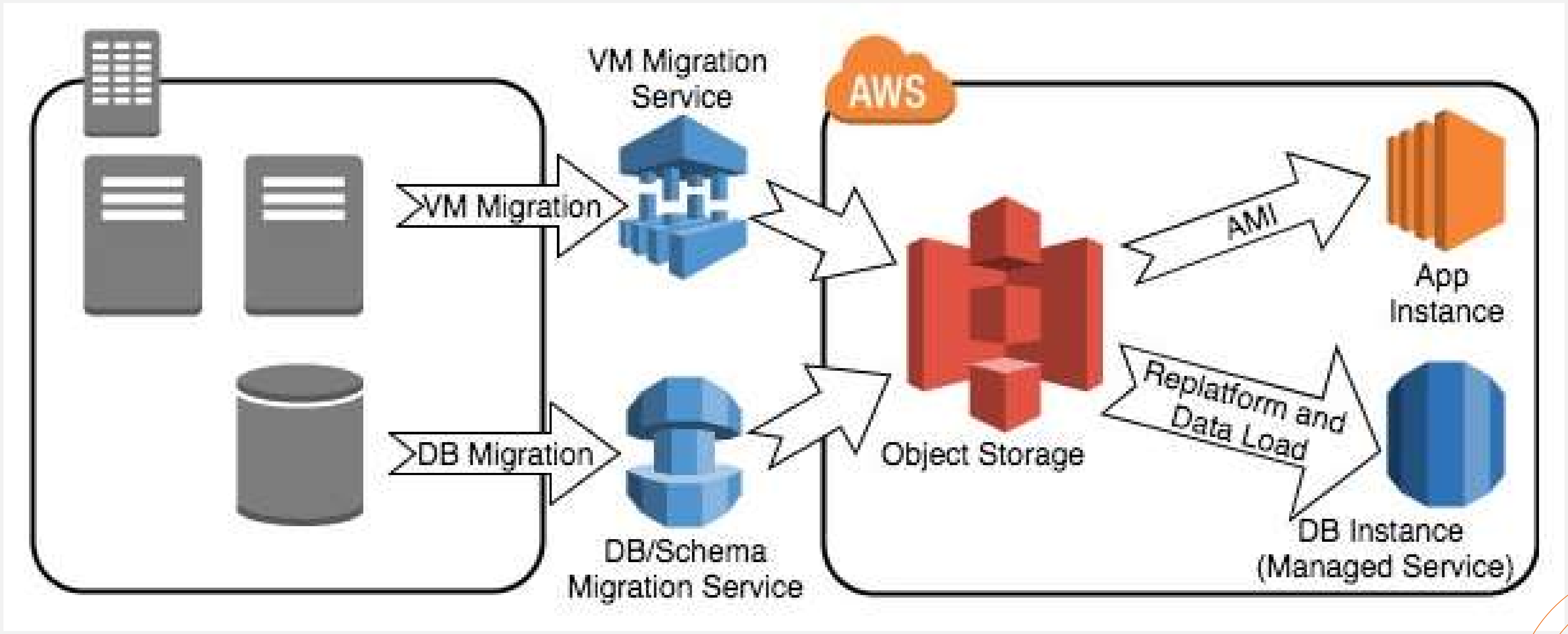
Cloud migration versus greenfield development

- There is also a belief that rehosting in the cloud will accelerate the ability to replatform or refactor applications once in the cloud, thereby reducing the risk of making the changes during migration, as seen in the following diagram:



Cloud migration versus greenfield development

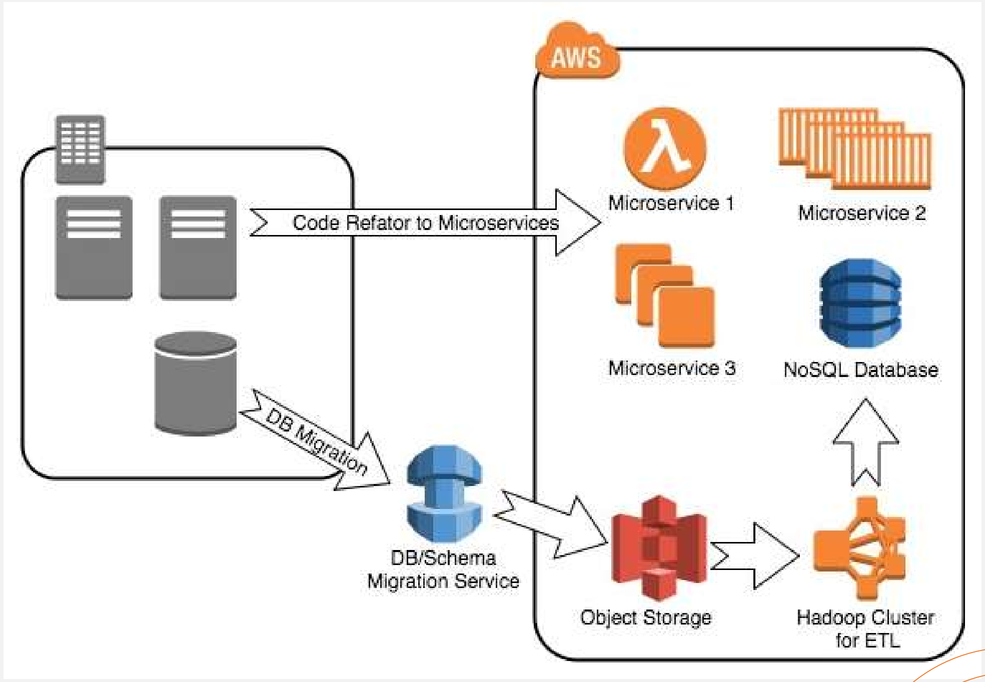
Replatform



Cloud migration versus greenfield development

Refactor

- Refactoring still applies to migrations since, in most cases, the data is being transformed to a newer database engine (in other words, NoSQL), the application is being redesigned to decouple the service components (in other words, microservices) and, most importantly, the business logic stays the same (or is enhanced) during this process, as seen in the following diagram:



Cloud migration versus greenfield development

Migrate or greenfield development?

- Throughout this lesson, cloud drivers and considerations were discussed, and we dived deeper into how an enterprise would migrate existing workloads into the cloud.

- While some of these patterns contain development activities to refactor or replatform the application, they are still migration patterns.

Summary

- In this lesson, we identified what exactly is reason why companies make the decision to move to the cloud.
- Understanding these key drivers – agility, cost, security, governance, expansion, talent, and innovation – and their level of maturity within the company, will drive the final decision to make the journey.
- Once the decision is made, it is critical to define and implement the operating model, which will assure stakeholders, as well as change and project management, that risk and compliance requirements are managed and implemented to ensure success in the cloud.

Lesson 3. Cloud Native Application Design

Lesson 3. Cloud Native Application Design

- In this lesson, we will dive deep into the development of cloud native architectures, using microservices and serverless computing as a mature CNMM state.
- Starting with the evolution of computer systems from monolithic and moving through much more mature architectures, this lesson will cover containers, orchestration, and serverless, how they fit together, and why they are considered mature on the application-centric design axis.

From monolithic to microservices and everything in between

- Client-server applications have always been popular. However, as networking technology and design patterns have evolved, the need to have less tightly coupled applications intercommunicating has given way to service-oriented architectures (SOAs).

- An SOA is the concept of breaking down the components that make up a monolith or server into more discrete business services.

From monolithic to microservices and everything in between

- Cloud computing continues to evolve these patterns because of increasingly lower costs for these services, as well as the rising availability of resources to create services.

- Microservices, which are similar to SOAs, are an evolution that allows for a specific service to be broken down into even more discrete components. A service in an SOA is a black box that performs a full business function.

From monolithic to microservices and everything in between

System design patterns

- The evolution of systems design is one where various pressures impacted the way systems were built and deployed to solve ever more complex business problems.
- This section will cover these patterns in more detail and provide additional insights into how they work and the challenges they faced, forcing evolution.
- To understand how solution architecture has progressed, it is important to define a few key concepts, namely primitives, subsystems, and systems.

From monolithic to microservices and everything in between

Monolithic

- From the early days of information technology and the advent of computing power to speed up computational tasks, the preferred design pattern for an application was monolithic.

- Simply put, a monolithic design pattern is one in which all aspects of the system are self-contained and independent from other systems or processes.

From monolithic to microservices and everything in between

Client server

- Eventually, as technology costs lowered and design patterns continued to advance to more sophisticated levels, a new style of architecture became popular: client server applications.
- The monolith still existed to provide the backend processing power and data storage required. However, because of advances in networking and database concepts, a frontend application, or client, was used to provide a user experience and transmit data to the server.

From monolithic to microservices and everything in between

Services

- Client server applications worked very well, but the coupling of components caused slow and risky deployments for mission-critical applications.
- System designers continued to find ways to separate components from each other to reduce the blast radius of failure, increase the speed of deployments, and reduce risk during the process.
- A services design pattern is where business functions are broken down into specific objectives to complete in a self-contained manner that doesn't require knowledge of or dependencies on the state of another service.

From monolithic to microservices and everything in between

Service-oriented architectures (SOAs)

- A services design pattern that has been around since before the cloud is the SOA. This is the first major evolution of breaking down a monolithic application into separate parts to reduce the blast radius and help make application development and deployment processes faster to support business velocity.
- SOA design patterns are often still a large application split into subsystems that interact with each other just as they would in a monolithic system.

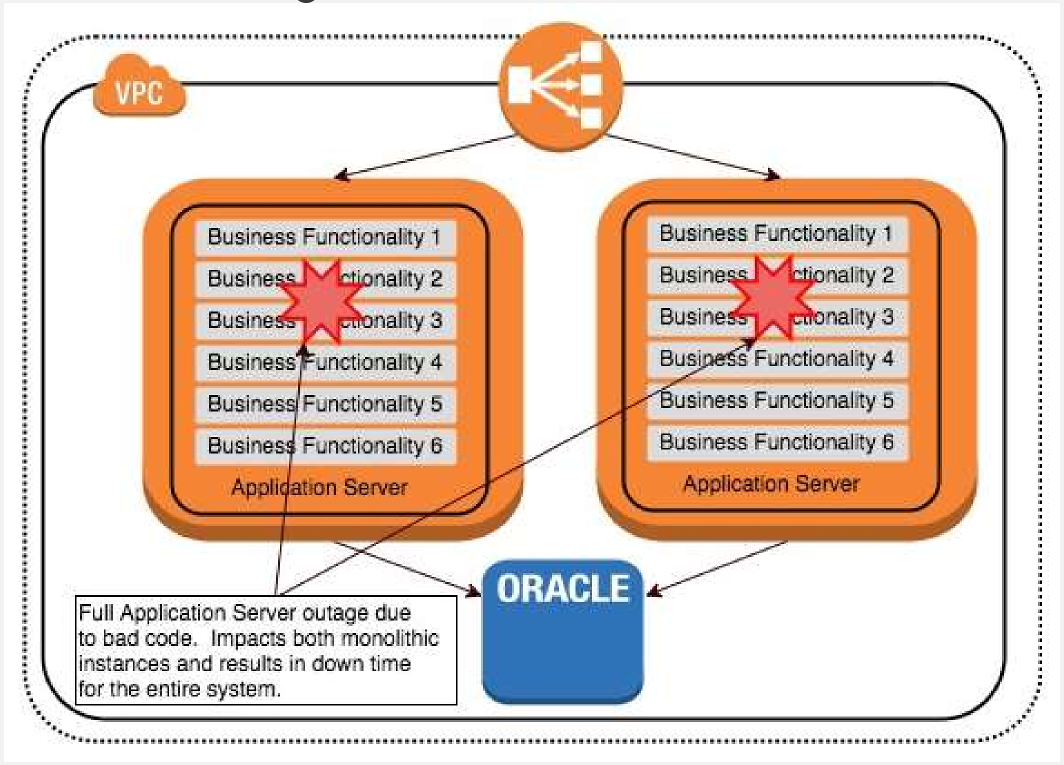
From monolithic to microservices and everything in between

Microservices

- The evolution of SOAs has led to continued breakdown of business functionality into even smaller components, often referred to as microservices.
- These are still services, as defined earlier. However, the scope of the services and the way they intercommunicate has evolved to take advantage of technology and pricing.
- Microservices are often used to help reduce some of the challenges faced with SOAs, while continuing to mature the services model.

From monolithic to microservices and everything in between

Why services matter

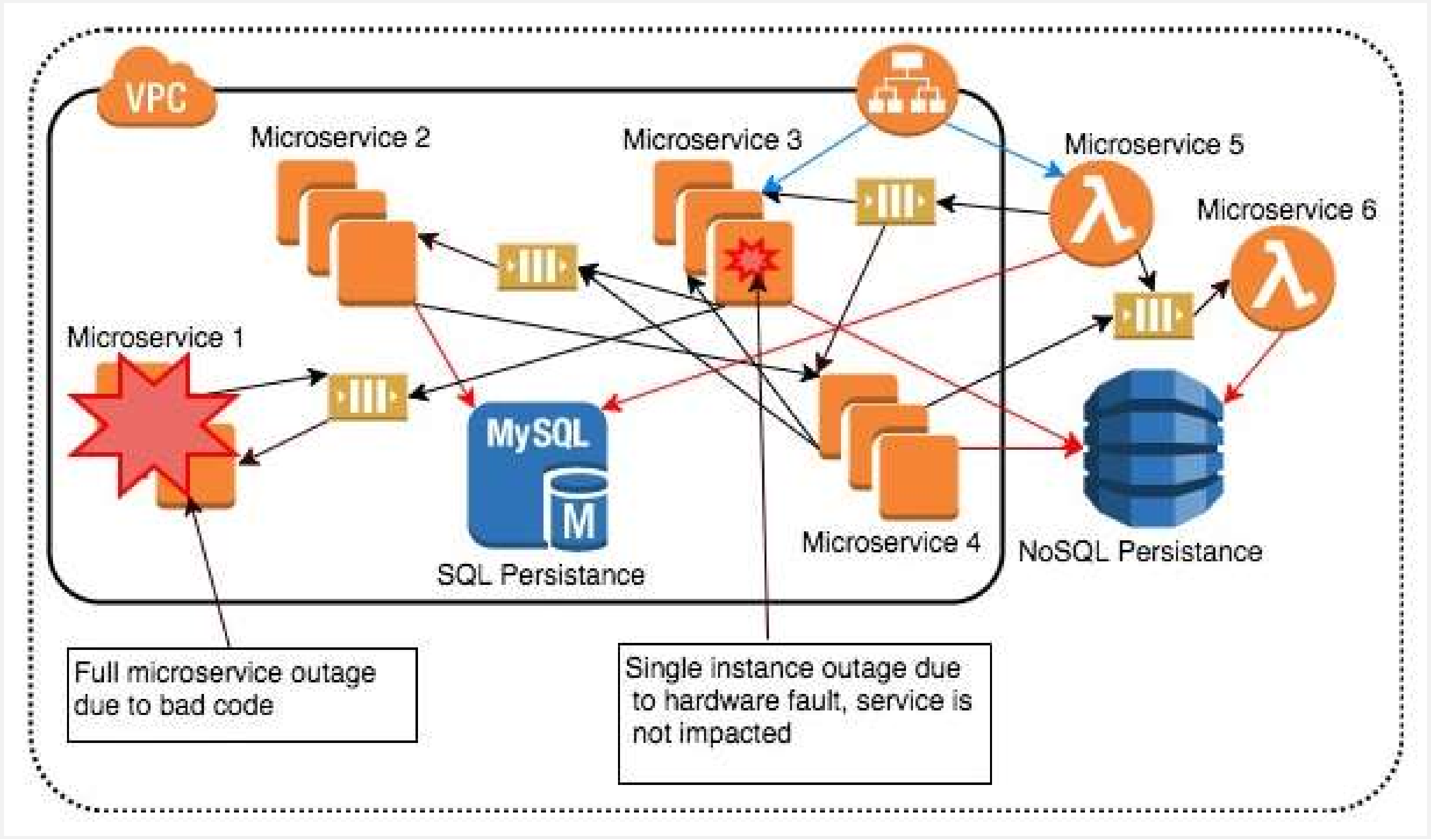


Full Application Server outage due to bad code. Impacts both monolithic instances and results in down time for the entire system.

From monolithic to microservices and everything in between

- The preceding diagram shows a monolithic system, deployed to multiple instances for high availability, but using a design pattern to deploy the full application functionality into an application server on each instance.
- In this case, the business functionality boxes are an example of a subsystem that would have many different functions, or primitives that are included to complete the required functionally.

From monolithic to microservices and everything in between



From monolithic to microservices and everything in between

- In the preceding diagram, the original monolithic application was redesigned and deployed into separate subsystems and primitives, using a mix of instances, containers, functions, and cloud vendor services.
- Not all services interact with each other or persist data, so if there are problems in the environment most are still usable. Subsystem 1 shows as having an outage due to bad code being deployed, which will be discovered and resolved by the appropriate services team.

Containers and serverless

- Cloud native architectures have matured over the years as the patterns evolve to take advantage of the newest advances in cloud technologies.

- Microservices are currently the hot topic for architecture trends since they allow for massive decoupling of components while utilizing cloud native services in ways that would be impossible with on-premises software. However, microservices are just a pattern.

Containers and serverless

Containers and orchestration

- Containers are a natural progression of additional isolation and virtualization of hardware.
- Where a virtual instance allows for a bare metal server to have multiple hosts, hence gaining more efficiencies and usage of that specific server's resources, containers behave similarly but are much more lightweight, portable, and scalable.
- Typically, containers run a Linux, or, in some cases, Windows operating system

Containers and serverless

Registries

- A container registry is simply a public or private area for the storage, and serving, of pre-built containers.

- There are many options for a registry that can be used, and all the major cloud vendors have a hosted version of registries that can be made private for a specific organization.

Containers and serverless

Orchestration

- The orchestration of containers is where the hard part comes in.
- If an application was designed to use containers as a large monolithic stack that performs all the tasks required for the subsystem to execute, then deploying those containers becomes a pretty easy task.
- However, this also defeats the purpose of using containers and microservices, which is to decouple the components into small services that interact with each other or other services.

Containers and serverless

- Kubernetes master: This is responsible for maintaining the desired state for your cluster.
- Kubernetes node: Nodes are the machines (VMs, physical servers, and so on) that run your applications and cloud workflows. The Kubernetes Master controls each node; you'll rarely interact with nodes directly.
- Pod: This is the basic building block of Kubernetes; the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod represents a running process on your cluster.
- Service: An abstraction that defines a logical set of pods and a policy by which to access them, sometimes called a microservice.

Containers and serverless

- Replica controllers and Replica Sets: These ensure that a specified number of pod replicas are running at any one time. In other words, they make sure that a pod or a homogeneous set of pods is always up and available.
- Deployment: Provides declarative updates for pods and Replica Sets.
- Daemon Set: Ensures that all (or some) nodes run a copy of a pod. As nodes are added to the cluster, pods are added to them. As nodes are removed from the cluster, those pods are garbage collected. Deleting a Daemon Set will clean up the pods it created.

Containers and serverless

Container usage patterns

- Containers are a great tool for designing and implementing cloud native architectures.
- Containers fit well in the cloud native maturity model that was outlined in Lesson 1, Introducing Cloud Native Architecture.
- Specifically, they are a native cloud service from the vendor, they allow for extreme automation through CICD, and they are great for microservices due to their lightweight and scalable attributes.

Containers and serverless

Microservices with containers

- Containers are often synonymous with microservices. This is due to their lightweight properties, containing only a minimal operating system and the exact libraries and components needed to execute a specific piece of business functionality.

- Therefore, applications, whether designed as new or a broken down monolithically, will be designed to contain exactly what is needed, with no additional overhead to slow down the processing.

Containers and serverless

Hybrid and migration of application deployment

- Containers are a great way to run a hybrid architecture or to support the migration of a workload to the cloud faster and easier than with most other methods.
- Since a container is a discrete unit, whether it's deployed on-premises (such as a private cloud) or in the public cloud (for example, AWS), it will be the same.

Containers and serverless

Container anti patterns

- In the world of cloud computing, containers are not only a popular way to achieve cloud native architectures, but they are quickly becoming the standard with which to design microservices.
- However, there are some use cases where containers are not always suited, specifically using containers for multiple concerns.
- The concept of concerns, or focus areas of a component, is that each module or class should have responsibility over a single part of the functionality.

Containers and serverless

Serverless

- Serverless does not mean the absence of servers; however, it does mean that resources can be used without having to consider server capacity.
- Serverless applications do not require the design or operations team to provision, scale, or maintain servers. All of that is handled by the cloud provider.
- This approach is probably the first time ever where the developers can focus on exactly what they are best at: writing code.

Containers and serverless

Scaling

- Serverless applications are able to scale by design, without the requirement of having to understanding individual server capacity units.

- Instead, consumption units are the important factor for these services, typically throughput or memory, which can be automatically updated to have more or less capability as the application requires.

Containers and serverless

Serverless usage patterns

- The ways to utilize serverless technology is limited to only the imagination of the design team involved in the system creation.
- Every day new techniques are developed that further expand the designs of what can be achieved with serverless, and due to the fast pace of innovation by the cloud providers, the new features and services in this area are increasing all the time.

Containers and serverless

Web and backend application processing

- The easiest way to describe this design pattern is to compare it with a traditional three-tier application architecture.
- For many years, the dominant approach to a scalable web app was to set up a load balancer, web servers, app servers, and databases.
- There are many types of third-party tools and techniques to support this design and, if done properly, a fault tolerant architecture can be achieved that will scale up and down to support the spike in traffic that the application will experience.

Containers and serverless

Data and batch processing

- One of the biggest advantages of using the cloud is the ability to process large amounts of data quickly and efficiently.
- The two most common high-level approaches are real-time and batch data processing.
- Similar to the previous example, each of these patterns has been around for a while and there are ways to achieve them using more traditional tools and approaches.
- However, adding serverless will increase velocity and reduce pricing.

Containers and serverless

System automation

- Another popular serverless design pattern is to perform common environment maintenance and operations.
- Following the CNMM requirement to have full automation, functions are a great way to execute the tasks of the system demands.
- As cloud landscapes grow, automation will ensure that the critical tasks to support that growth stay consistent and do not require the organization to linearly scale up the operations team supporting the environment.

Containers and serverless

Serverless anti patterns and concerns

- Serverless is a growing and critical way to achieve success with cloud native architectures. Innovation is happening all the time in this space, and will continue for many years.
- However, there are some patterns that do not directly apply to serverless architectures. For example, long-running requests are often not valid given the short-lived time for a function (less than 5 minutes).

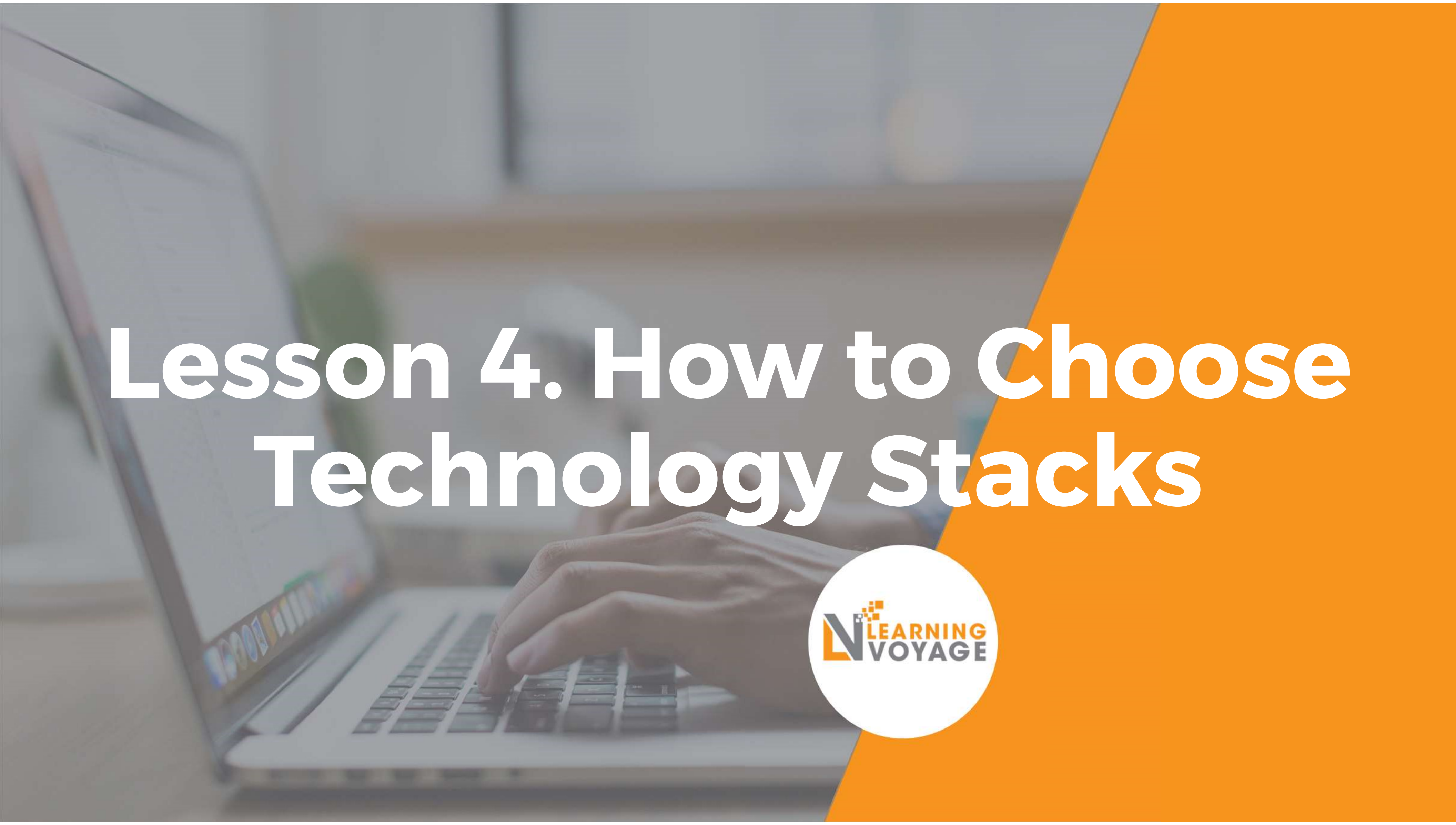
Development frameworks and approaches

- There are many frameworks and approaches taken to design horizontal scalable applications in the cloud.
- Due to the nature of each of the cloud providers, each has their own specific services and frameworks that work best.
- In Lesson 9, Amazon Web Services, a comprehensive outline of doing this with AWS can be found. In Lesson 10, Microsoft Azure, Microsoft Azure's approach is detailed. Lesson 11, Google Cloud Platform, breaks down the patterns and frameworks that work best for that cloud.
- Each of these are similar, yet different, and will be implemented differently by each customer design teams.

Summary

- In this lesson, we dived deep into the development of cloud native architectures, using microservices and serverless computing as a design principle.
- We were able to learn the difference between SOAs and microservices, what serverless computing is and how it fits into a cloud native architecture.
- We learned that each of the mature cloud vendors has a unique set of frameworks and approaches that can be used to design scalable architectures on those platforms.

Lesson 4. How to Choose Technology Stacks



Lesson 4. How to Choose Technology Stacks

- The world of cloud computing is vast, and while there are a few dominant players in the cloud vendor space, there are other parts of the ecosystem that are critical to success.

- How to decide which cloud vendor to use? What types of partners should be considered and what do they bring to the table? How will procurement change or stay the same in the cloud? How much do you want to rely on the cloud vendor to manage services? These are all important and valid questions that will be answered in this Lesson.

Cloud technology ecosystems

Public cloud providers

- This course is focused on public cloud providers, which, in the fullness of time, will most likely be the dominant way companies consume IT resources.

- The cloud, as it is today, began in 2006 when Amazon Web Services (AWS) launched its first public services (Amazon Simple Queueing Service and Amazon Simple Storage Service).

Cloud technology ecosystems

- While there are other clouds out there, with different niche focus areas and approaches to addressing customer requirements, these three comprise the dominant global market share for cloud services.

- Making the decision on a cloud vendor can appear complex when originally researching the vendors, mainly because less dominant players in the space try to position themselves as better situated than they are by showing cloud revenues or market share on services that are really not cloud at all.

Cloud technology ecosystems

- Scale: The cloud vendor business is one of scale and ability to deliver anywhere at any time. Even if a customer doesn't currently need cloud resources on a global scale, or to do a massive scale out in a single geography, using a cloud vendor that can do this is critical to ensure they have the experience, funding, and desire to continue to grow and innovate.

- Security/compliance: The priority of all cloud vendors should be security. If a cloud vendor doesn't focus on this and already has the majority of compliance certifications, it is best to stay clear and choose a vendor that makes this a priority.

Cloud technology ecosystems

- Feature richness: The pace of innovation is ever increasing and cloud vendors are moving into areas that were not even considered a few years ago. Whether it be machine learning, blockchain, serverless, or some other new technology, the cloud vendors that are constantly innovating are the ones to focus on.

- Price: While never the sole a reason to choose a vendor, price should always be considered. Contrary to popular belief, the cloud vendor market is not a race to the bottom in terms of pricing. Scale and innovation allow vendors to lower prices by passing along savings to customers (another reason why scale matters).

Cloud technology ecosystems

Independent software vendor (ISV) and technology partners

- Cloud vendors are the base of any cloud native strategy, while ISV and technology partners will form a core component in this journey.
- In a lot of cases, the cloud vendors have tools that are native to their platforms and will fill the exact needs of customers.
- However, there are many reasons why they cannot meet all the requirements, and that is where ISV and technology partners come in.

Cloud technology ecosystems

Customer managed products

- In the case of customer managed products, the ISV has chosen to either create or migrate their existing product to a cloud-based model.

- Often, this means the vendor has taken significant steps to adopt cloud native architectures themselves and then sell their product to the end customer.

Cloud technology ecosystems

Software as a Service

- ISVs and technology partners have often chosen to completely redesign their offerings to be service usage-based, making consumption easier for the customer.
- To achieve this, the vendors will almost always choose a specific cloud vendor and develop their offering on top of the cloud vendor, making their software available to customers but not giving access to the underlying cloud vendor infrastructure or services.

Cloud technology ecosystems

Consulting partners

- Consulting partners, or system integrators (SIs), have been around since almost the beginning of IT itself.
- The concept is that companies don't have the time, resources, or desire to skill up on new technology or scale out their workforce to handle the demands of their business requirements.
- Therefore, these SI partners are used to fill gaps and help the company move faster with people who can quickly make an impact and deliver projects and business results.

Cloud technology ecosystems

Niche SI partners

- Niche partners are those companies that offer very specific services on a type of technology, cloud, or area of focus.
- Often, these partners will be relatively small in size, but have some of the most experienced experts in their respective areas.
- They are used as subject matter experts on large projects or to own the delivery of a specific project in their area of expertise.

Cloud technology ecosystems

Regional SI partners

- Regional SI partners are popular with companies that operate in a specific geography or region, since the partner will usually have all the resources located in that area.
- Often considered strategic partners with senior executive relationships, these partners will help the customer define their strategy and execute specific projects with customer resources.

Cloud technology ecosystems

Global SI partners

- Global SI partners are popular for their scale and ability to support the largest, longest, and most complex projects customers have.
- As the name implies, they are global in nature, operating in most countries in the world, with a significant presence in all geographies.
- From the beginning, global SI partners have played a role in helping outsource IT operations, design, strategy, and delivery for big and small companies alike, allowing them to focus on their core business.

Procurement in the cloud

- One of the reasons that the cloud is so popular, and that cloud native architectures are becoming the normal method of designing workloads, is due to the procurement and consumption model used.

- The easy way to think about this is the ability to procure using operational expenditure (OpEx) instead of the traditional capital expenditure (CapEx), which can have a large impact on the way a company handles revenues, taxes, and long-term depreciation of assets.

Procurement in the cloud

Cloud marketplaces

- Using a cloud vendor marketplace as a location to find and procure software required for their business to operate in the cloud is one way companies are able to have the tools needed at any moment.
- Fundamentally, customers are looking to either develop new workloads or move existing workloads to the cloud, modernize the workload if it was migrated, and then manage their landscape at scale with a streamlined workforce.

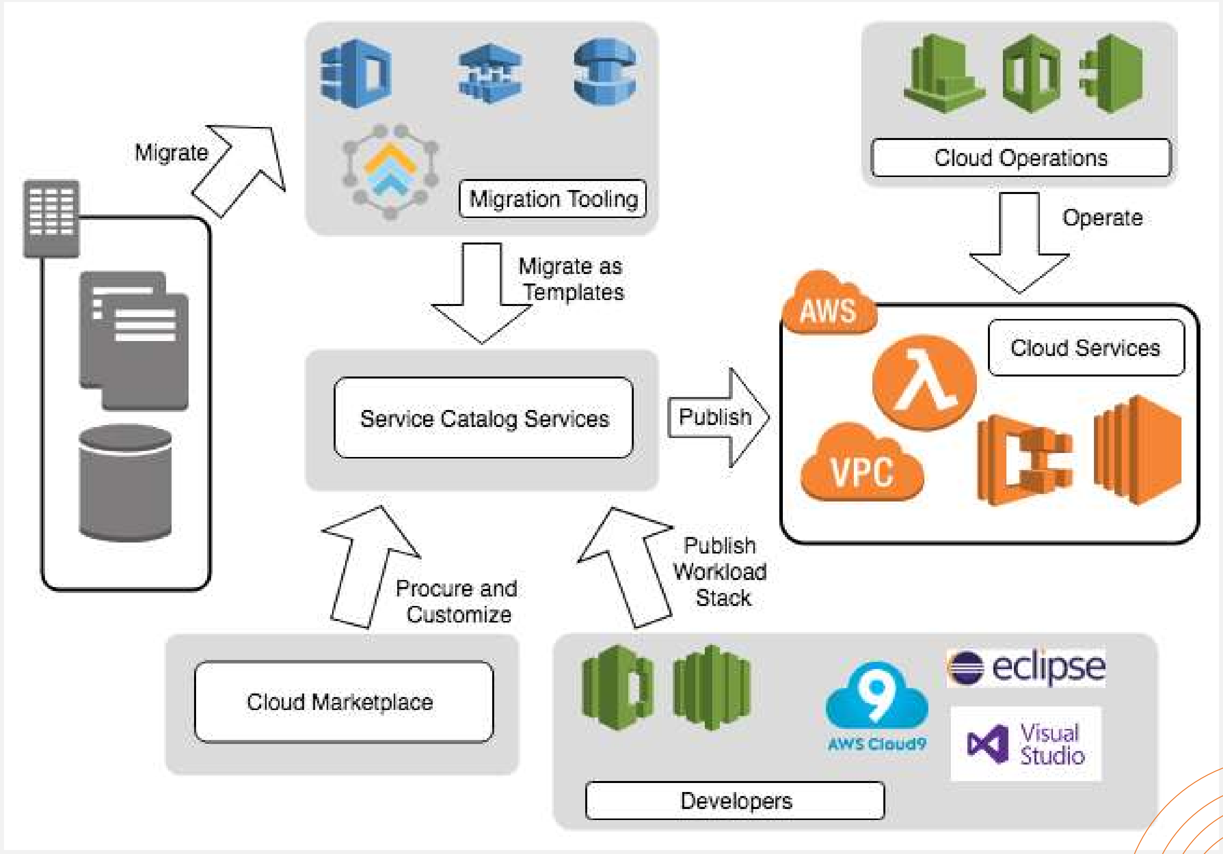
Procurement in the cloud

Marketplace and service catalogs

- Cloud vendors often have a service catalog feature that allows for pre-created application stacks or complex solutions to be stored and deployed when required by users with the right permissions.
- This approach, when combined with the marketplace, offers a powerful way to allow the service design teams to choose the right software, stage it in the catalog location, and either deploy it or have the operations team deploy it following company guidelines.

Procurement in the cloud

- The following diagram shows where the marketplace and a service catalog offering might fit into an organization's procurement strategy:



Procurement in the cloud

Cloud marketplace anti-patterns

- The cloud marketplace can be a powerful tool to allow companies to acquire software with little or no long-term negotiating.

- It also allows companies to test new software to determine whether it will solve a business problem with ease and little in the way of upfront costs. However, there are cases where using the marketplace is not a fit.

Procurement in the cloud

Licensing considerations

- The previous section identified some areas of consideration with regards to procurement and discussed the common pay-as-you-go model of consumption.
- Typically, the cost of cloud vendor services would not be considered licenses, but consumption; however, there are often additional license costs associated with it, depending on what software is being used.

Procurement in the cloud

- Network throughput, storage amount, or other physical component. They meter and bill by how much network traffic is used, how many Gigabytes (GB) of storage are used, or other hardware metrics that are not CPU-based.

- Per host. Server instances are still very popular in the cloud, and will be for a long time. Depending on the software, charging a per host fee will be enough to get an accurate accounting of how much of the software is used. This would usually be charged on a unit of time basis (for example, hours) to account for elasticity and would not be tied to the CPUs.

Procurement in the cloud

- Percentage of cloud vendor spend. For software that is used across the entire cloud landscape, such as monitoring or endpoint security, vendors might charge a small percentage of the overall customer cloud spend. This would allow for elastic growth and shrinkage to be accounted for in the payment to the ISV.

- By transaction. Some ISVs have very high transactional rates, but very low sizes of transaction. By metering the amount of transactions and charging a small amount per transaction, the ISV can account for elasticity and still give a pay-per-use model to the customer.

Procurement in the cloud

Cloud vendor pricing models

- It's important for customers to fully understand the pricing metric for each piece of technology being purchased.

- This is true especially with cloud vendors, who can have complex pricing models for services that are very new and otherwise hard to price.

Procurement in the cloud

Example - AWS Lambda pricing

- Lambda counts, as a request, each time it starts executing in response to an event notification or invocations call, including test invocations from the console, You are charged for the total number of requests across all your functions.

- Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 100 ms, The price depends on the amount of memory you allocate to your function.

Procurement in the cloud

- The following table shows the free tier seconds and the approximate price per 100 ms associated for different memory sizes:

Memory (MB)	Free tier seconds per month	Price per 100 ms ($)
128	3,200,000	000000208
192	2,133,333	000000313
256	1,600,000	000000417
320	1,280,000	000000521
2,816	145,455	000004584
2,880	142,222	000004688
2,944	139,130	000004793
3,008	136,170	000004897

Procurement in the cloud

Example - Amazon DynamoDB pricing

- This example of Amazon DynamoDB pricing is taken directly from the AWS pricing page, and additional details can be found on that page.
- Unlike traditional NoSQL deployments that ask you to think about memory, CPU, and other system resources that could affect your throughput, DynamoDB simply asks you to specify the target utilization rate and minimum to maximum capacity that you want for your table.

Procurement in the cloud

- The following table summarizes key DynamoDB pricing concepts:

Resource type	Details	Monthly price
Provisioned throughput (write)	One **write capacity unit** (WCU) provides up to one write per second, enough for 2.5 million writes per month	As low as $0.47 per WCU
Provisioned throughput (read)	One **read capacity unit** (RCU) provides up to two reads per second, enough for 5.2 million reads per month	As low as $0.09 per RCU
Indexed data storage	DynamoDB charges an hourly rate per GB of disk space that your table consumes	As low as $0.25 per GB

Procurement in the cloud

- Manual provisioning example: Assume that your application running in the US East (N. Virginia) region needs to perform 5 million writes and 5 million eventually consistent reads per day on a DynamoDB table, while storing 8 GB of data.

- For simplicity, assume that your workload is relatively constant throughout the day and your table items are no larger than 1 KB in size.

Procurement in the cloud

Open source

- Similar to traditional on-premises environments, customers will have a large range of software options to choose from when starting on their cloud native journey.
- As pointed out already, licensing considerations are critical to understand before making any decision.
- Open source software can be very effective in the cloud, and often the preferred method for companies to fill important gaps in their technology landscape.

Cloud services

- Cloud providers have a lot of services, and innovation is accelerating; therefore, understanding how to choose the right service to solve business problems can be a tough prospect.
- The large cloud providers have designed their services to be like building blocks that can be used together to solve problems that may never have been considered for that specific service.

Cloud services

Cloud services – vendor versus self-managed

- In the early days of the cloud, the foundational infrastructure services were reason enough to develop new workloads there.
- However, as those services matured and the cloud vendors quickened their pace of innovation, they began to develop other services that were managed versions of existing software customers had been using.

Cloud services

Self-managed approach

- Setting up databases, sometimes with clusters or other high availability approaches, and including the correct disk type and networking requirements, can be challenging.

- For years, this skill set, often referred to as a database administrator (DBA), was one that required very skilled and experienced professionals to do correctly.

Cloud services

Managed cloud services

- Vendor-managed cloud services are a way for customers to use the technology they need without having to deal with the undifferentiated lifting of managing the operations of the technology.

- Cloud vendors spend a lot of time and resources to design services that mimic or exceed the way that companies manage similar technology on-premises.

Cloud services

Vender lock-in

- Throughout the history of IT, there has been a common theme of vendors making their products sticky and hard to migrate away from.
- While this trend is seemingly done to support customer requirements, if often makes it difficult, if not impossible, to switch to a newer technology, leaving customers locked into the vendor they originally chose.

Cloud services

- Have an exit plan. Every decision that is mission-critical requires a plan of how to exit or migrate if the need arises. Know which applications are capable of moving quickly, what data goes with them, and how it's done.

- Design loosely coupled and containerized applications. Following the 12-factor app and other cloud native design principles, designing loosely coupled applications will enable easier movement with less risk of blast radius issues.

Cloud services

- Organize and manage critical data closely. Know exactly what data you have, where it resides, and its classification and encryption requirements. This will ensure that if the decision is made to move to a different cloud, the order of operations for the data is already known.

- Automate everything. Using Infrastructure as Code and other DevOps philosophies will allow for minimal changes to the automation to support another cloud. Automation is key to a cloud native company.

Cloud services

Operating systems

- Foundational infrastructure services typically include cloud virtual instances.
- These instances require an operating system to run, and for the cloud, that often comes down to choosing between a flavor of Linux or a version of Windows.
- In addition to virtual instances, containers also require a flavor of operating system to run, albeit a slimmed down version, to deploy code and the frameworks that make them execute.

Cloud services

Windows versus Linux

- The old question of Windows versus Linux is one that most customers have to agree on at some point.
- However, for any cloud deployment of size, most likely there will be a mixture of both Windows and Linux, with different versions and flavors of each.
- Therefore, it's important to know which versions are needed and which are offered by the cloud vendor.

Cloud services

Do operating systems really matter any longer?

- The real question is do operating systems really matter any longer? The answer is yes and no.
- Yes, because containers are still a common technology for developing microservices and therefore an operating system is required.
- No, because as the architecture matures, more and more managed cloud services will be leveraged with serverless as a core component of cloud native architecture.

Summary

- In this Lesson we learned the answers to the questions you may ask while deciding on which technology stack to choose.
- The next Lesson talks about scalability and availability in cloud architectures.

Lesson 5. Scalable and Available

Lesson 5. Scalable and Available

- In the previous lessons of this course, we defined what a cloud-native architecture is and the impact these architectures have on people, processes, and technology.
- The journey to building cloud-native systems is neither straightforward nor short.
- It can take years to fully realize the potential of going cloud-native as your organization and culture matures.
- Skill sets and problem solving approaches need to change over time.

Lesson 5. Scalable and Available

- In the next four lessons of this course, we seek to define the core pillars of cloud native design.
- These design pillars are core features that make a system cloud-native.
- These features on their own are not exclusive to cloud applications or systems.
- However, when taken together, they are a set of features that no other platform but cloud can bring to the table.

Lesson 5. Scalable and Available

- In this lesson, the reader will first gain an understanding of the scale at which modern cloud providers operate.
- There has been a clear departure from ordinary scale (the scale of data centers with which most seasoned IT practitioners are familiar with) to hyper-scale.
- This hyper-scale cloud infrastructure inherently forms many of the constructs and approaches discussed in this lesson.

Lesson 5. Scalable and Available

The following topics will be covered in this Lesson:

- Global Cloud infrastructure and common terminology
- Cloud infrastructure concepts (regions and AZs)
- Autoscaling groups and load balancers
- VM sizing strategies
- Always-on architectures
- Designing network redundancy and core services
- Monitoring

Lesson 5. Scalable and Available

The following topics will be covered in this Lesson:

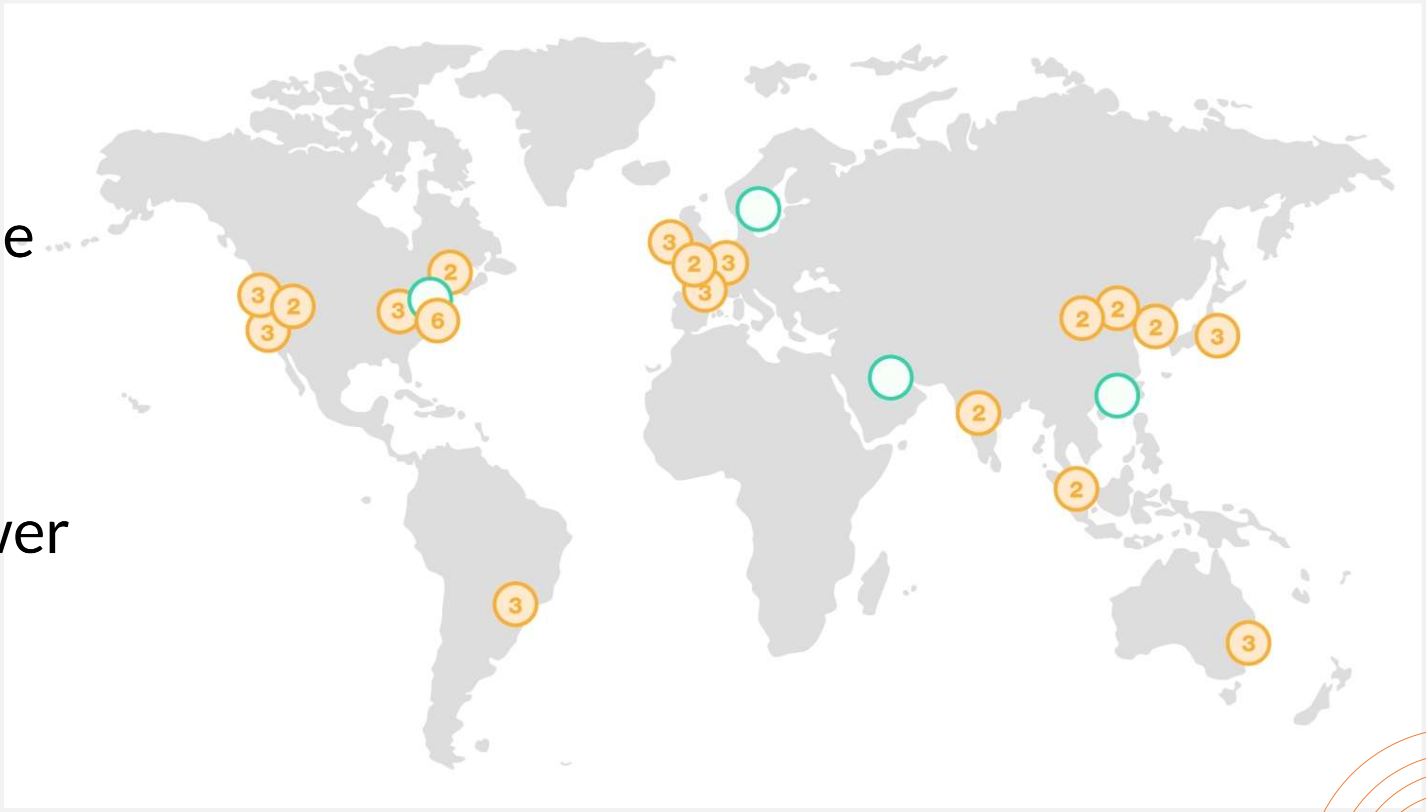
- Infrastructure as Code
- Immutable deployments
- Self-healing infrastructure
- Core tenets of scalable and available architectures
- Service oriented architectures
- Cloud Native Toolkit for scalable and available architectures

Introduction to the hyper-scale cloud infrastructure

- When deploying systems or stacks to the cloud, it is important to understand the scale at which leading cloud providers operate.
- The three largest cloud providers have created a footprint of data centers, spanning almost every geography.
- They have circled the globe with large bandwidth fiber network trunks to provide low latency, high throughput connectivity to systems running across their global data center deployment.

Introduction to the hyper-scale cloud infrastructure

- The following diagram depicts the global footprint of AWS, the largest cloud provider by total compute power (estimated by Gartner):



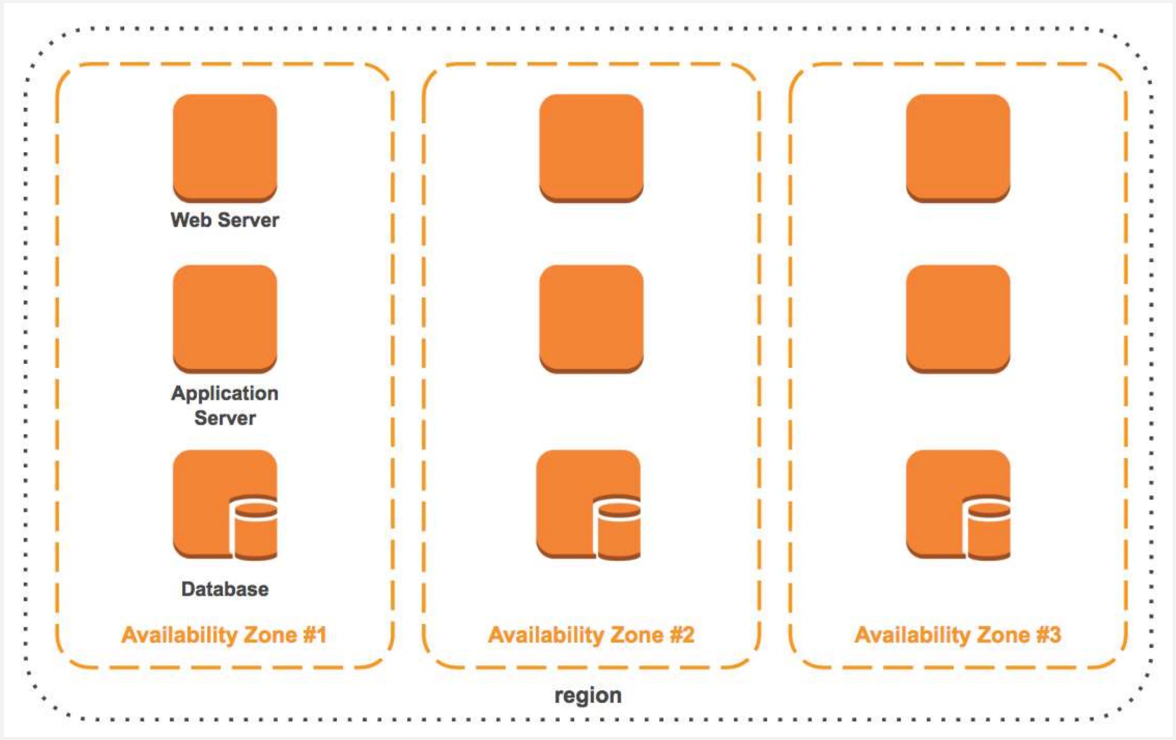
Introduction to the hyper-scale cloud infrastructure

Introduction to the hyper-scale cloud infrastructure

- Google Cloud Platform (GCP) and Microsoft
  Azure provide similar geographic coverage. All three hyper cloud platforms provide core cloud services from an infrastructure abstraction called a region.
- A region is a collection of data centers that operate together under strict latency and performance requirements.
- This in turn allows consumers of the cloud platform to disperse their workloads across the multiple data centers comprising the region. AWS calls this availability zones (AZs) and GCP calls it zones.

Introduction to the hyper-scale cloud infrastructure

- The following diagram shows three different AZs within one region:



Introduction to the hyper-scale cloud infrastructure

- Load balancers are not a unique development to cloud, but the major platforms all have native services that provide this functionality.
- These native services provide much higher levels of availability, as they are running on many machines instead of a virtual load balancer running on a VM operated by the cloud consumer.

Introduction to the hyper-scale cloud infrastructure

- The concept of load balancers should be familiar to all who work in the IT industry.
- Flowing from and adding to the concept of LBs are cloud services, which allow users to provide DNS services coupled with load balancing.
- This combination allows users to build globally available applications that extend seamlessly to geographies of your choosing.

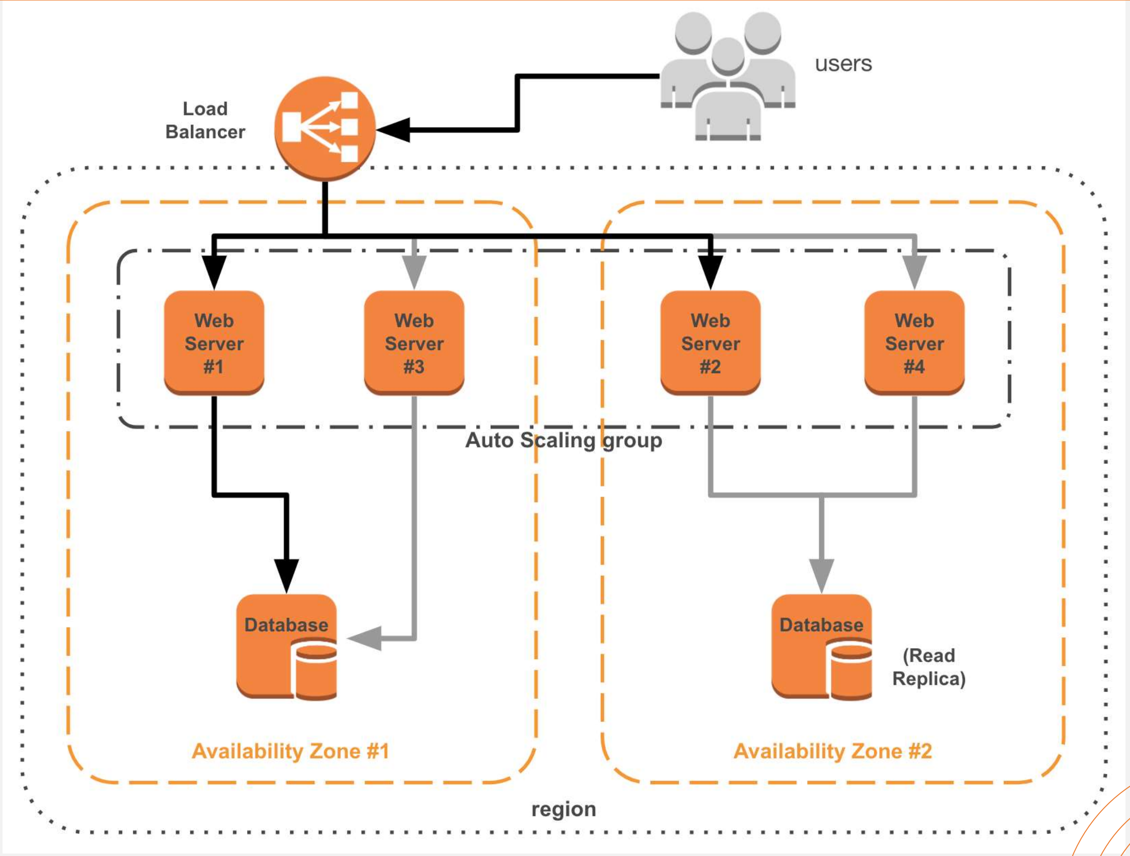
Introduction to the hyper-scale cloud infrastructure

- Another important tool in the cloud-native toolbox is the deployment of auto scaling groups (ASG) – a novel service abstraction that allows users to replicate and expand application VMs dynamically based on various alarms or flags.

- In order to deploy an ASG, a standardized image of the application must first be pre-configured and stored.

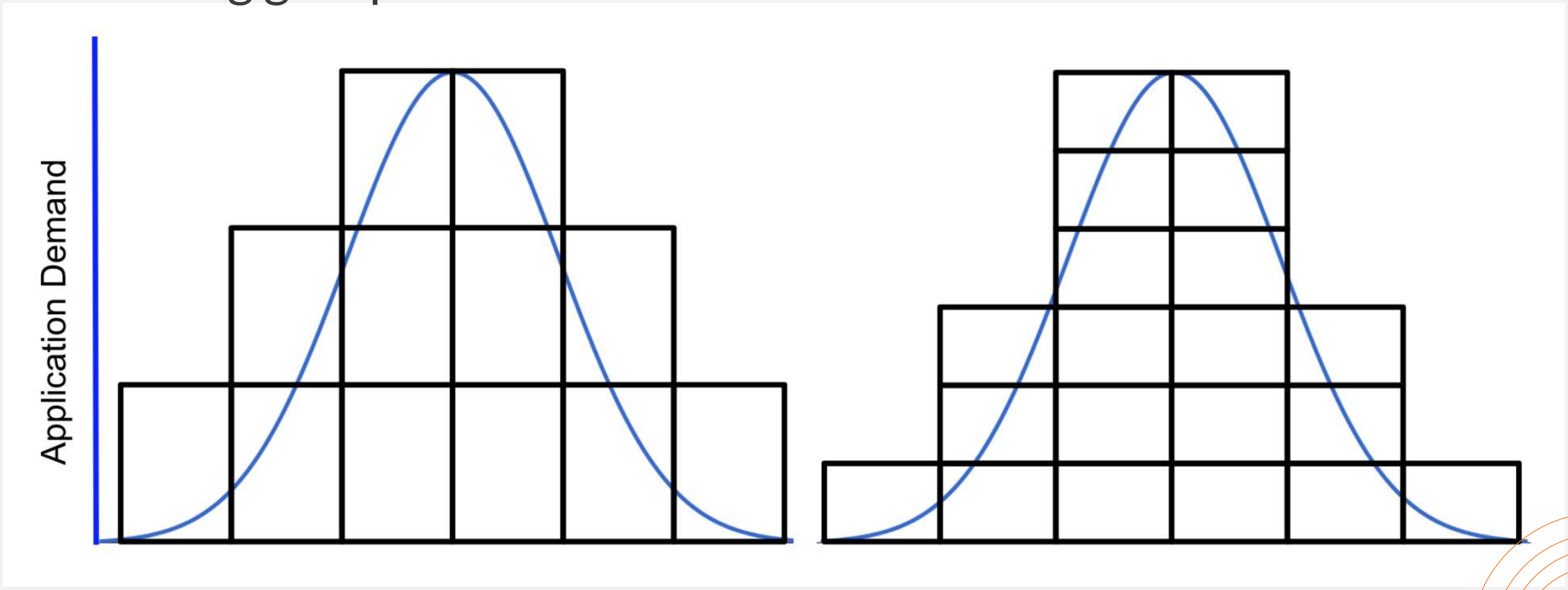
Introduction to the hyper-scale cloud infrastructure

- A basic auto-scaling configuration across two AZs is shown in the following diagram:



Introduction to the hyper-scale cloud infrastructure

- Let's have a look at the comparison between auto-scaling groups with different VM sizes:



Introduction to the hyper-scale cloud infrastructure

- In the preceding graph, we have demonstrated the advantage of using smaller compute nodes in an ASG fleet.
- Any spaces within a block that appear above the blue line are wasted resources that are paid for.
- This means that the performance of the system is not ideally tuned to the application's demand, leading to idle resources.
- By minimizing the VM specs, significant cost savings can be achieved.

THE END

LEARNING VOYAGE