Why Move to the Cloud Now ?

This section of the course will go into the fundamentals of why there is a need to move to the cloud for software development manager (SDM), Technical Program Manager (TPM), IT-Manager, directors.

We are at an interesting point in time where the way we use and think about infrastructure is changing. The move to the cloud has been talked about for the last 10 years or so but I believe we are at an interesting tipping point. The reason for this is that there has been a rapid growth in the maturity of cloud infrastructure available and of course the cost of being on the cloud.

When we talk here about infrastructure it is important to understand that IaaS is made up of hardware like switches and servers but also there is a layer of software that helps users to deploy, manage and maintain the infrastructure that is built but the cloud provider. Though the primary capabilities are similar, advanced capabilities differ from provider to provider i.e AWS vs Azure.

Over the last couple of years the core components of infrastructure have advanced in terms of capabilities and from an operational perspective teams have learnt how to operate in an Agile way. This has in turn made leaders within the IT and technology space to take a second look at moving to the cloud.

If you take a step back and look at what organizations have been doing; they have been procuring hardware, racking servers, managing and maintaining their hardware infrastructure. There is a tremendous amount of manpower required and manpower translates to cost. Also the cost of the acquired hardware translates into hardware depreciating over the predetermined life of the hardware.

The cost of procurement and maintenance aspect can be minimized when you move from an on-premises datacenter setup to using the cloud. This also helps companies focus more on their core business rather than spending time and resources working on managing their infrastructure that does not add a lot of

business value to its customers. On the other hand moving to the cloud helps organizations build reliable, scalable, and durable applications.

One of the fundamental benefits of the cloud is the opportunity to swap upfront capital infrastructure cost with low variable costs that scale with growth. With the cloud, you no longer need to plan and procure servers and other IT infrastructure equipment weeks or months in advance. Instead, infrastructure can be instantly spun up in minutes. This is something that can no longer be overlooked. Think about the retail industry that has certain days in the year like black friday or boxing day where traffic to websites go up by 5x. If you are a retail company like Target or Best Buy you would need to buy 5x more hardware just to accommodate the spike after which that hardware is no longer necessary.

Moving to the cloud is not all rosy, it comes with it own issues like the initial cost of moving to the cloud, having the right type of expertise to run your applications in the cloud and your security posture in general. Sometimes customers who move to the cloud feel that a lot of work has been put into moving to the cloud but the outcome of moving to the cloud is not very tangible from a end user perspective.

In this course you will learn about the fundamentals of why there is the need now more than ever to move to the cloud. The reason why moving to the cloud makes sense, how do legacy applications move to the cloud and the various types of approaches to move to the cloud. You will learn terminologies that you would need to fundamentally understand when we talk about the cloud. We will move on to talking about what cloud native designs i.e building applications for the cloud. As we learn about Cloud native architecture we will get you a quick primer on microservices and their importance and also touch a little on Kubernetes and understand the part it plays.

You will get an overview of the various types of cloud deployment strategies keeping in mind Operations, Security, Redundancy and Scaling.

We will be using the AWS cloud of most of the examples in this course. The reason for choosing Amazon AWS for this course is that AWS is probably the more mature than most of its competition. It also has a wider range of

adoption in the industry and is easier to access for you as an individual.

Who is this course for ?

This course is primarily designed for managers and leaders in IT or Tech organizations. You could be a development manager (SDM), Technical Program Manager (TPM), IT-Manager, director or you could be someone in tech who just wants to learn about the cloud.

The course aims to give you a good primer on cloud design patterns, cloud terminology, cloud components cloud best practices practices and operating on the cloud. This is to ensure that you get to know all you need to know when your organization moves to the cloud or when you start interacting with a cloud team. It will also help people who are looking to work for cloud providers like AWS or Azure or any other SaaS service ask the right type of questions.

If you are a leader in the tech industry we are sure you are seeing the need to know of the fundamentals of the cloud now more than ever. More than 60% jobs require some sort of cloud exposure. Some places don't even call this out anymore as they believe that most of the talent already know of the fundamentals. The job market is evolving driven by the evolution of organizations.

Why Cloud ?

If you think about every successful tech company - Facebook, Amazon, Netflix, Google have been able to grow and scale the way they have only because of the scalability and agility that cloud offers. By choosing to run your infrastructure on the cloud you are able to provide value to your customer at a quicker pace. Your teams do not need to wait for provisioning i.e ordering, waiting and racking servers which in most places takes a couple of weeks at the very least. An idea today could be deployed as soon as possible with less time spent waiting for infrastructure.

Also consider the rapid growth of startups. The startup space has exploded over the last 8 years because there is no longer a large cost to acquire infrastructure to build and launch your applications. And supposing an app have viral growth then the fact that you are on the cloud automatically takes care of the infrastructure aspects of scale.

Cloud Computing Models & Their Evolution

Understand the differences between on-premises vs IaaS vs PaaS and SaaS offerings.

On-Premises

In the On-Premises model you as a customer own and manage your own Datacenter. Hard to manage and maintain. There are two models here -

1. The customer has the datacenter at the location where the dev team is located. Its normally in a generic office building. Though this is only done when the hardware footprint is small the disadvantages of this setup is that when you grow its very hard to grow the hardware as there are physical space constraints. Also if your building loses power for some reasons your infrastructure is affected.

2. You are co-located with several other tenants in a datacenter owned and operated by a third party. You pay a fee for the space, electricity you as a customer use. This is a very common mode of operation. However in an event of a hardware/server failure you need to send a technician to fix it.

Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) offers the basic as well as advanced building blocks for cloud and provides access to three things -

1. Compute
2. Storage
3. Networking.

IaaS provides you flexibility and management control over your IT resources that are similar to existing IT controls that most IT departments and developers are familiar with today.

IaaS supports services such as identity and access management (IAM), provisioning and inventory systems. IaaS allows organizations to get rid of all

of their in-house hardware and to rent VMs or physical servers from someone else. This frees up a lot of people resources and gets rid of processes that are needed for purchasing, maintenance, and in some cases eliminates the need for capacity planning. The magnitude of time and money organizations spend on managing their infrastructure on their own cannot be overstated.

Think about it you can provision hardware infrastructure anywhere in the world by simply clicking on a couple of buttons or you can mirror your entire datacenter deploy hundreds of machines and set up your VCN just with a few clicks or with a quick deployment script using cloud formation or terraform.

Platform as a Service (PaaS)

Platform as a Service (PaaS) removes the need for your organization to manage the underlying infrastructure (usually hardware and operating systems) and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Think about the team of 100s of database engineers in a enterprise doing patching, tuning, etc now with a managed Database service your team would not need to worry about doing this. Platform As A Service basically takes the managing of a certain piece of software away form the user. The way you would pay for this is depending on your usage. There are advantages as a well as disadvantages of using PaaS today.

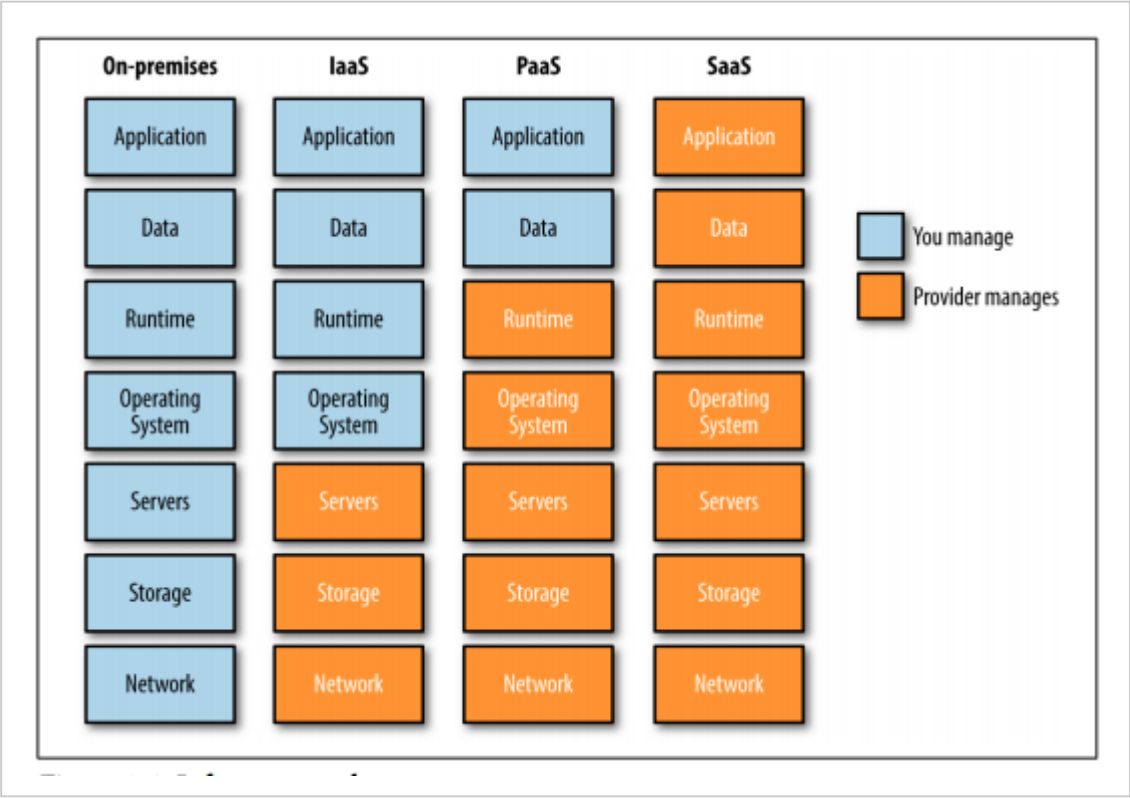
Some of the key advantages would be that you do not need to own and operate something like your database. You do not necessarily need to get into contract negotiations with companies like Oracle. You don't necessarily need to worry about managing the application. Your cloud provider gives you an SLA with money back guarantee.

The disadvantage would be that as a customer you would lose out on granular controls on when patches are done or what version of a patch you would want to use etc. Though cloud providers understand these pain points and are working to mitigate these.

Software as a Service (SaaS)

Software as a Service (SaaS) provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software.

A common example of a SaaS application is web-based email which you can use to send and receive email without having to manage feature additions to the email product or maintain the servers and operating systems that the email application is running on.



Let's Start With Some Basics

Understand the fundamentals of what is a Server, Virtualization, Hypervisor. What are the fundamental components that encompass a Data Center ? This section will then go into the reasons why organizations move to the cloud and the technical benefits of moving to the cloud.

The below is a glossary of terms that you need know. The reason for brining this right in the start of the course is to help you to understand the fundamental terminologies before proceeding through the course.

Servers

Think of them as computers that are responsible for the storage of data, computation (where your application lives and provides some value), perform some networking functions. Inside a server you will find a processor, motherboard, RAM, a networking card, a hard disk and a power supply.

Servers are expensive and they require a lot of power and people to keep them operational. They are normally procured by the IT team and are kept running as long as possible so that the organization gets the maximum return on their investment. The hardware on the servers fail more often than you think and requires maintenance.

Physical servers are great because they can be configured however you want. But, physical servers lead to waste because it is difficult to run multiple applications on the same server. Software conflicts, network routing, and user access all become more complicated when a server is maximally utilized with multiple applications. Hardware virtualization solve some of these problems.

Virtualization

Virtualization emulates a physical server’s hardware in software. A Virtual Machine (VM) can be created on demand, is entirely programmable in software. Hypervisors increases these benefits because you can run multiple virtual machines (VMs) on a physical server.

Hypervisors

Hypervisors allows applications to be portable because you can move a VM from one physical server to another. One problem with running your own virtualization platform, is that VMs still require hardware to run. Companies still need to have all the people and approvals required to run physical servers, but now capacity planning becomes harder because they have to account for VM overhead too.

What is a Data Center ?

A data center is a physical facility that is used to house all the hardware that is needed for applications to function.

Data Center Components

Data centers are often referred to as a singular thing, but in actuality they are composed of a number of technical elements such as routers, switches, security devices, storage systems, servers, application delivery controllers and more. These are the components that are needed to store, run and manage applications. Due to the needs of the application and the decreasing life of the hardware managing a data center is a time consuming task.

Data Center Infrastructure

In addition to technical equipment a data center also requires a significant amount of facilities infrastructure to keep the hardware and software up and running. This includes power subsystems, uninterruptible power supplies (UPS), ventilation and cooling systems, backup generators and cabling to connect to external network operators i.e a connection to the internet where applicable.

Data Center Architecture

Any company of significant size will likely have multiple data centers possibly in multiple regions. This gives the organization flexibility in how it backs up its information and protects against natural and manmade disasters like floods, storms, earthquakes etc. How the data center is architected can be some of the most difficult decisions because there are almost unlimited options.

Reasons To Move To The Cloud

Capital Expense vs Variable Expense !

Instead of having to invest heavily in building data centers and acquiring servers before you know how you are going to use them, you pay only when you consume resources.

Benefit from massive economies of scale:

By using cloud computing, you can achieve a lower variable cost than you can get on your own. Because usage from hundreds of thousands of customers are aggregated in the cloud, providers like AWS, Azure, Oracle can achieve higher economies of scale, which translates into lower pay as-you-go prices for the end IaaS customers - you !

Stop guessing about capacity:

Eliminate guessing on your infrastructure capacity needs. When you make a capacity decision prior to deploying an application, you often end up either sitting on expensive idle resources or dealing with limited capacity. With cloud computing, these problems are negated. You can access as much or as little capacity as you need, and scale up and down as required.

Increase speed and agility:

In a cloud computing environment, new IT resources are only a click away, which means that you reduce the time to make those resources available to your developers from weeks to just minutes. This results in a dramatic increase in agility for the organization, since the cost and time it takes to experiment and develop is significantly lower.

Stop spending money on running and maintaining data centers:

Focus on projects that differentiate your business, not the infrastructure. Cloud computing lets you focus on your own customers, rather than on the heavy lifting of racking, stacking, powering and maintaining your servers.

Go global in minutes

Easily deploy your application in multiple regions around the world with just a few clicks. This means you can provide lower latency and a better experience for your customers at minimal cost.

Almost zero upfront infrastructure investment:

If you have to build a large-scale system it may cost a fortune to invest in real estate, physical security, hardware (racks, servers, routers, backup power supplies), hardware management (power management, cooling), and operations personnel. Because of the high upfront costs, the project would typically require several rounds of approvals before the project could even get started. Now, with utility-style cloud computing, there are no fixed or startup cost.

Just-in-time Infrastructure:

In the past, if your application became popular and your systems or your infrastructure did not scale you became a victim of your own success. Conversely, if you invested heavily and the app did not get popular, your investment in infrastructure is lost. By deploying applications in-the-cloud with just-in-time self-provisioning, you do not have to worry about pre-procuring capacity for large-scale systems. This increases agility, lowers risk and lowers operational cost because you scale only as you grow and only pay for what you use.

More efficient resource utilization:

System administrators usually worry about procuring hardware (when they run out of capacity) and higher infrastructure utilization (when they have excess and idle capacity). With the cloud, they can manage resources more effectively and efficiently by having the applications request and relinquish resources ondemand.

Usage-based costing:

With utility-style pricing, you are billed only for the infrastructure that is being used. You are not paying for allocated or unused infrastructure. This adds a new dimension to cost savings. You can see immediate cost savings (sometimes as early as your next month's bill) when you deploy an optimization patch to update your cloud application.

Moreover, if you are building and selling a platform on the top of the cloud, you can pass on the same flexible, variable usage-based cost structure to your own customers.

Reduced time to market:

Parallelization is the one of the great ways to speed up processing. If one

compute-intensive or data-intensive job that can be run in parallel takes 500 hours to process on one machine, with cloud architectures, it would be

possible to spawn and launch 500 instances and process the same job in 1 hour. Having available an elastic infrastructure provides the application with the ability to exploit parallelization in a cost-effective manner reducing time to market.

Technical Benefits of Cloud Computing

Some of the technical benefits of cloud computing include:

Automation“Scriptable infrastructure”:

You can create repeatable build and deployment systems by leveraging programmable (API-driven) infrastructure. This comes back to the point of deploying in a new geographical region in less than 10 minutes. You are not hunting for data centers or buying hardware. You simply run your cloud formation script to create your entire fleet of servers along with all your network configurations.

Auto-scaling:

You can scale your applications up and down to match your unexpected demand without any human intervention. Auto-scaling encourages automation and drives more efficiency.

Proactive Scaling:

Scale your application up and down to meet your anticipated demand with proper planning understanding of your traffic patterns so that you keep your costs low while scaling.

Efficient Development lifecycle:

Production systems may be easily cloned for use as development and test environments. Staging environments may be easily promoted to production.

Improved Testability:

Never run out of hardware for testing. Spin-up and spin down testing environments as and when you need them. Whether you want to run automation tests, load test, longevity tests you can use the infrastructure and return it when you are done with them.

Disaster Recovery and Business Continuity:

Disaster Recovery and Business Continuity:

The cloud provides a lower cost option for maintaining a fleet of DR servers and data storage. With the cloud, you can take advantage of geo-distribution and replicate the environment in other location within minutes.

“Overflow” the traffic to the cloud:

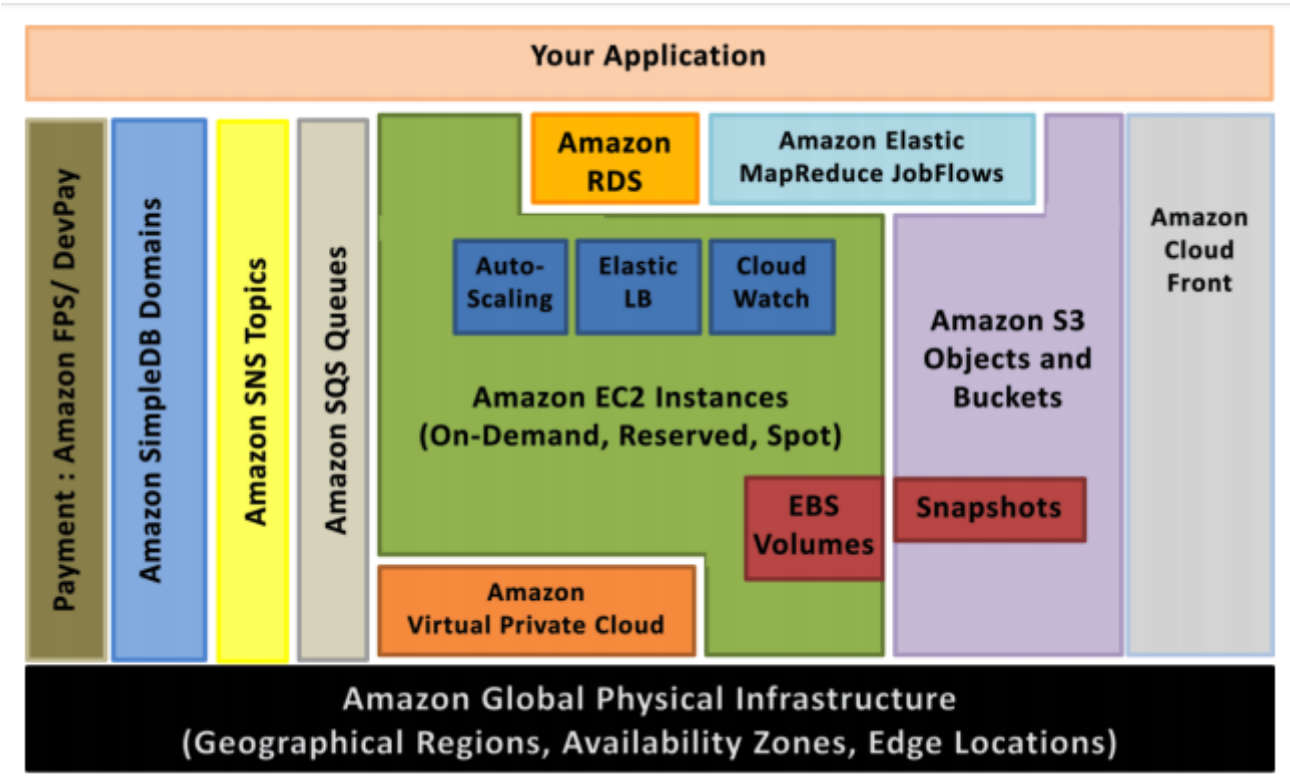
With a few clicks and effective load balancing tactics, you can create a complete overflow-proof application by routing excess traffic to the cloud from your existing on-prem infrastructure.

Understanding Amazon Web Services (AWS)

Review the fundamental building blocks of Amazon Web Services (AWS) and how they all fit together. Look at basic compute, Load Balancers, Identity & Access Management, Storage, etc.

The Amazon Web Services (AWS) cloud provides a highly reliable and scalable infrastructure for deploying web-scale solutions, with minimal support and administration costs, and more flexibility than you've come to expect from your own infrastructure, either on-premise or at a datacenter facility.

AWS offers variety of infrastructure services today. The diagram below will introduce you the AWS terminology and help you understand how your application can interact with different Amazon Web Services and how different services interact with each other.



Compute / VMs

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides re-sizable compute capacity in the cloud. You can bundle the operating system, application software and associated configuration settings into an

Amazon Machine Image (AMI).

Identity & Access Management

You can then use these AMIs to provision multiple virtualized instances as well as decommission them using simple web service calls to scale capacity up and down quickly, as your capacity requirement changes. You can purchase On-Demand Instances in which you pay for the instances by the hour or Reserved Instances in which you pay a low, one-time payment and receive a lower usage rate to run the instance than with an On-Demand Instance or Spot Instances where you can bid for unused capacity and further reduce your cost. Instances can be launched in one or more geographical regions.

Reliability / Fault Tolerance

Each region has multiple Availability Zones. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.

Public & Private DNS

Elastic IP addresses allow you to allocate a static IP address and programmatically assign it to an instance. You can enable monitoring on an Amazon EC2 instance using Amazon CloudWatch in order to gain visibility into resource utilization, operational performance, and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic). You can create an auto-scaling group using the Auto-scaling features to automatically scale your capacity on certain conditions based on metric that Amazon CloudWatch collects.

Load Balancing

You can also distribute incoming traffic by creating an elastic load balancer using the Elastic Load Balancing service.

Storage

Amazon Elastic Block Storage (EBS) volumes provide network-attached persistent storage to Amazon EC2 instances. Point-in-time consistent snapshots of EBS volumes can be created and stored on Amazon Simple Storage Service (Amazon S3).

Amazon S3 is highly durable and distributed data store. With a simple web

services interface, you can store and retrieve large amounts of data as objects

in buckets (containers) at any time, from anywhere on the web using standard HTTP verbs.

Amazon SimpleDB is a web service that provides the core functionality of a database- real-time lookup and simple querying of structured data - without the operational complexity. You can organize the dataset into domains and can run queries across all of the data stored in a particular domain. Domains are collections of items that are described by attribute-value pairs.

Content Deliver Network (CDN)

Copies of objects can be distributed and cached at 14 edge locations around the world by creating a distribution using Amazon CloudFront service – a web service for content delivery (static or streaming content).

Simple Notification Service

Amazon SNS is a fully managed pub/sub messaging service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications. With SNS, you can use topics to decouple message publishers from subscribers, fan-out messages to multiple recipients at once, and eliminate polling in your applications.

Amazon Simple Queue Service (SQS)

SQS is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS eliminates the complexity and overhead associated with managing and operating message oriented middleware.

Introduction to Cloud Native Principles

A cloud-native application is engineered to run on a platform and is designed for resiliency, agility, operability, and observability.

Resiliency, Agility, and Operability are the core pillars of the cloud. We then go on to learn a little bit of Microservices, Containers, VM and Kubernetes.

Now that you have an overview of a) what data center looks like b) have an understanding of the various pieces of AWS c) understand the reason on why your company needs to move to the cloud lets go into some of the Cloud Native Principles.

The reason that you need to deeply understand these principles is that ideally when an application is running on the cloud it needs to maximize the benefits of running on the cloud. When applications are not built keeping these principles in mind there are more chances of things failing.

Cloud Native Principles

A cloud native application is engineered to run on a platform and is designed for resiliency, agility, operability, and observability.

1. Resiliency embraces failures instead of trying to prevent them; it takes advantage of the dynamic nature of running on a platform.
2. Agility allows for fast deployments and quick iterations.
3. Operability adds control of application life cycles from inside the application instead of relying on external processes and monitors. Observability provides information to answer questions about application state. Think monitoring.

The Cloud Native Computing Foundation (CNCF) has defined cloud native as:

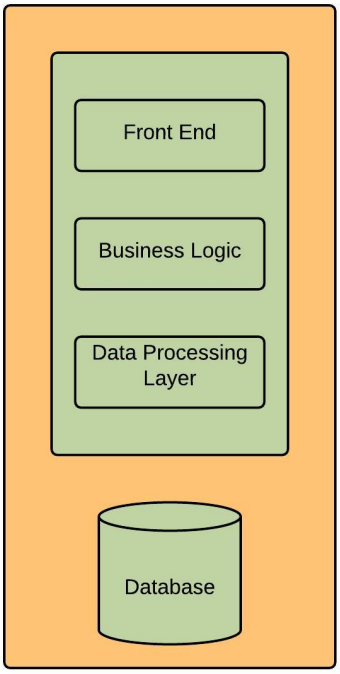
1. Micro-services based Architecture
2. Containerized
3. Distributed Management and Orchestration

Lets dive a little deeper and understand what this means.

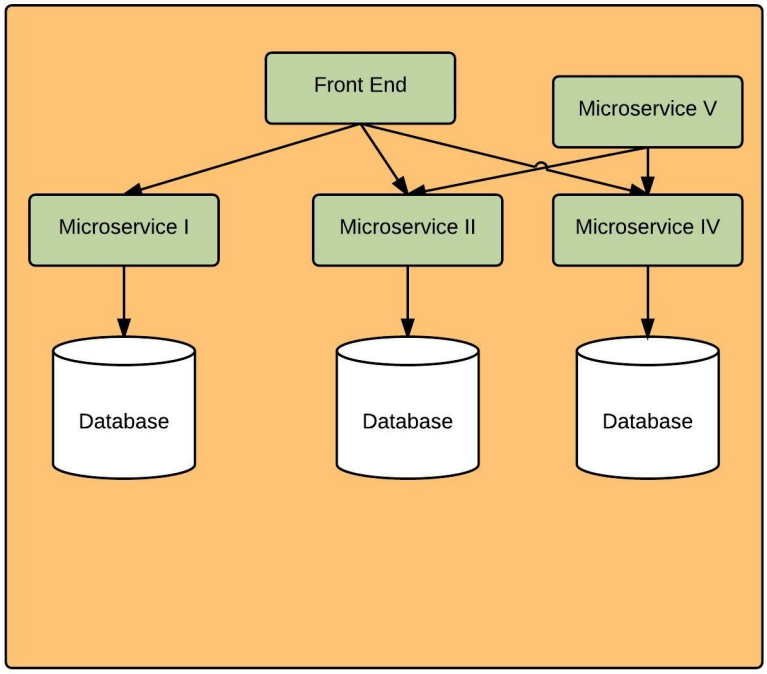
Micro-services Architecture

A micro-service architecture is a software architecture style where complex applications are composed of several small, independent processes communicating with each other using language-agnostic APIs. These application services are small, highly decoupled and focus on doing a small task. Microservices refers to smaller and more manageable services that serves a specific use case.

Microservices are generally deployed into containers. We will learn what is meant by containers in general and look into the most popular types of containers - docker containers.



Enterprise / Monolithic Architecture



Microservice Architecture

What is a container?

Container in short is a lightweight Virtual Machine. Containers are a way to package all that is needed for your application to work, be it operating system, libraries, config files or other applications all into one bundle.

There are 2 concepts in containers, Images & Running Containers.

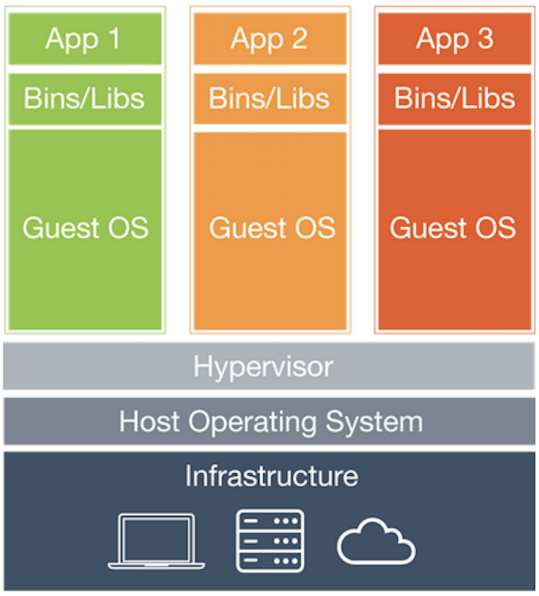
1. Image containers have images of the OS or the application itself, they are

static.

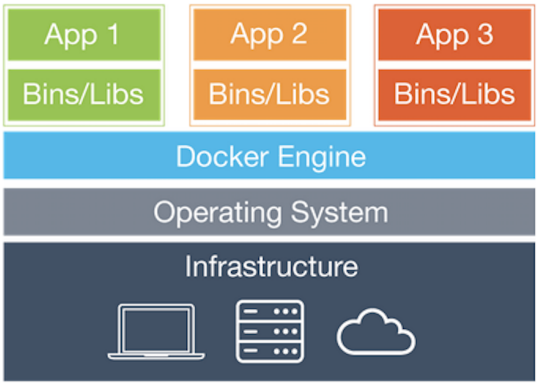
2. Running containers on the other hand are containers on a single host without visibility into each others' processes, files, network

Now, a container image includes an application & all its dependencies, using which you can create instances of Containers which can be run on any environment, be it developer, QA, support or in data-centers.

To understand the difference between VMs & containers, we should briefly look at VM architecture vs the Container Architecture.



Virtual Machines



Containers

Virtual Machines

VMs virtualize hardware aspect of a computer whereas containers virtualize at the operating system level. VMs need a Guest OS on top of Hypervisor, a virtualization software.

Hypervisors

Hypervisor manages the allocation of Hardware's processor, memory & resources to the VMs. On the flip side, Containers share the OS's compute, memory & resources and at the same time achieve the isolation provided by the VM architecture. Simply put, by eliminating the guest OS from a VM would give us Containers, which in itself reduces lot of overhead that comes along with VMs.

Containers

What are we trying to achieve using containers? Isolation, Co-existence of applications, virtualization of software, replication in diff environments.

Containerization is an OS feature where Kernel allows the existence of multiple, isolated user spaces known as Containers. These containers may appear as real computers to programs running in them as they have access to all the resources a real computer provides, like connected devices including network devices, compute power, memory, hard disk. However, a container is isolated from other containers & resources allocated to them.

If you need to replicate your application in multiple environments, you can package the application and all of its dependencies into a container image and all your developer, test, production & support teams will have the same image to start with. And once you have the container image, you can bring up the container in a matter of minutes or less.

Imagine a scenario in which a customer has an issue on the field and support folks try to replicate it in their own environments. How many times have they struggled with setting up the environments and not being able to reproduce it? Similar case repeats itself with developer & QA environments.

Now, just imagine a situation where the support folks are able to whip up a container and have exactly same environment as is in the field with the customer. Similarly, developers and test/QA personnel. I am sure you would have come across such inconsistencies in a lot of scenarios starting from developing a product to testing. This is just one of the issues that containers tackle.

What are docker containers ?

Docker is a platform for developing, shipping, running applications using container virtualization technology. Docker aids in separating your application from your infrastructure and helps in treating your Infrastructure like the way you would treat any managed application. Docker is one of the most commonly used Containerization tools.

Docker containers have demonstrated that containerization can drive the scalability and portability of applications. Developers and IT operations are

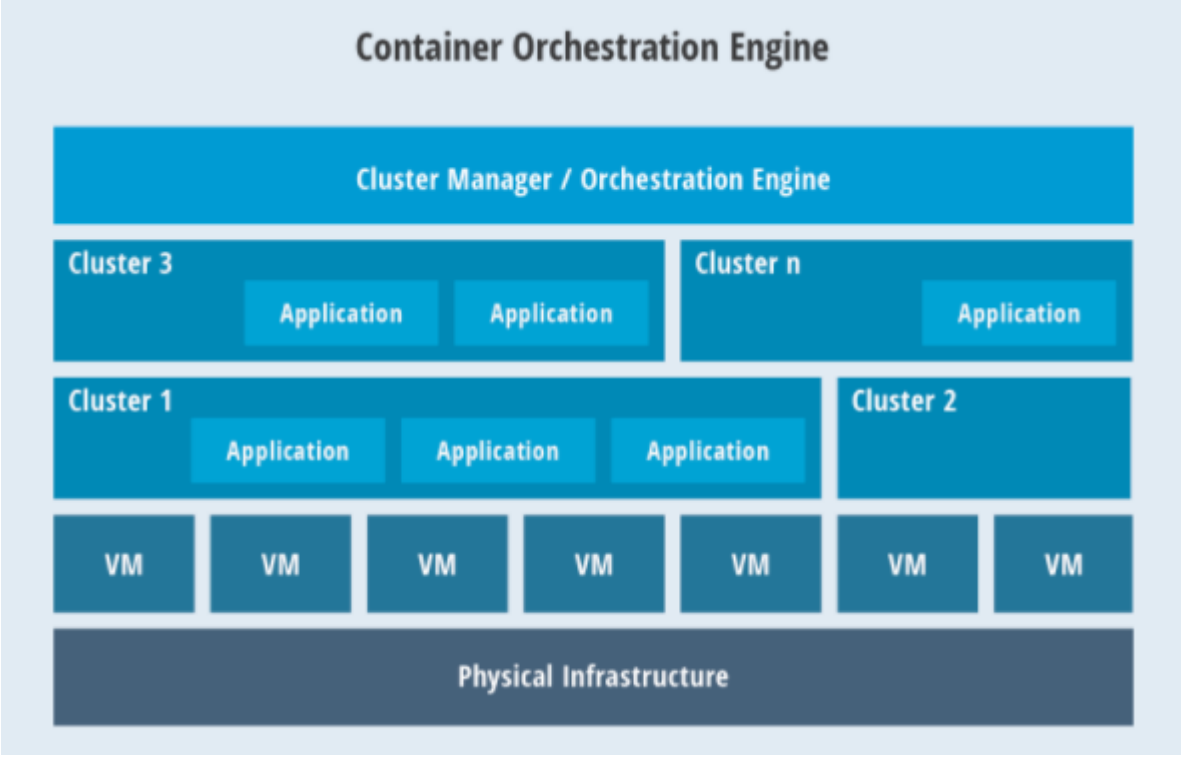
turning to containers for packaging code and dependencies written in a variety of languages. Containers are also playing a crucial role in DevOps processes. They have become an integral part of build automation and continuous integration and continuous deployment (CI/CD) pipelines.

Docker aims to provide a lightweight way to create containers to manage and deploy your applications with isolation and security where you can get more out of your hardware.

Kubernetes

Kubernetes is an open source container orchestration tool designed to automate deploying, scaling, and operating containerized applications. Why is this here ? It is here because “Distributed Management and Orchestration” is an integral part of making your application Cloud Native.

Kubernetes was born from Google’s 15-year experience running production workloads. It is designed to grow from tens, thousands, or even millions of containers. Kubernetes is container runtime agnostic.



Kubernetes’ features provide everything you need to deploy containerized applications. Here are the highlights:

Container Deployments & Rollout Control

Describe your containers and how many you want with a “Deployment.” Kubernetes will keep those containers running and handle deploying changes

Kubernetes will keep those containers running and handle deploying changes

(such as updating the image or changing environment variables) with a "rollout." You can pause, resume, and rollback changes as you like.

Resource Bin Packing

You can declare minimum and maximum compute resources (CPU & Memory) for your containers. Kubernetes will slot your containers into where ever they fit. This increases your compute efficiency and ultimately lowers costs.

Built-in Service Discovery & Autoscaling

Kubernetes can automatically expose your containers to the internet or other containers in the cluster. It automatically load-balances traffic across matching containers. Kubernetes supports service discovery via environment variables and DNS, out of the box. You can also configure CPU-based auto scaling for containers for increased resource utilization.

Heterogeneous Clusters

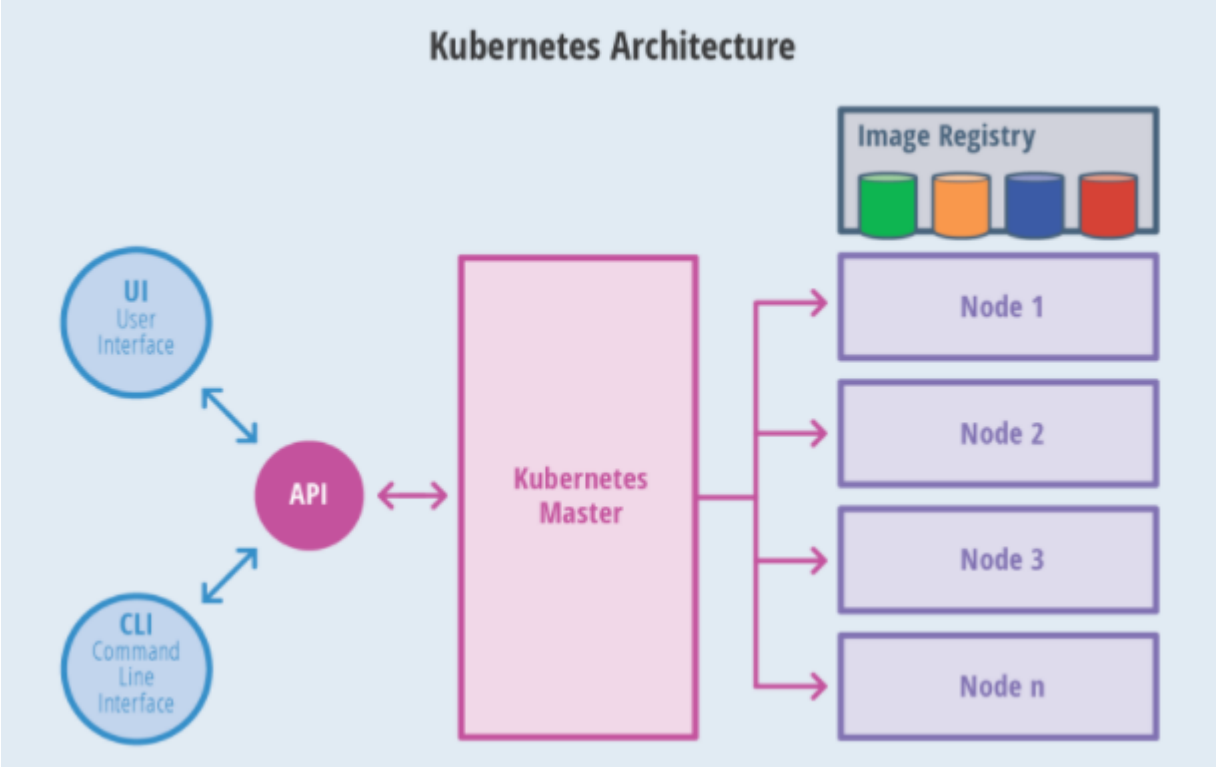
Kubernetes runs anywhere. You can build your Kubernetes cluster for a mix of virtual machines (VMs) running the cloud, on-prem, or bare metal in your datacenter. Simply choose the composition according to your requirements.

Persistent Storage

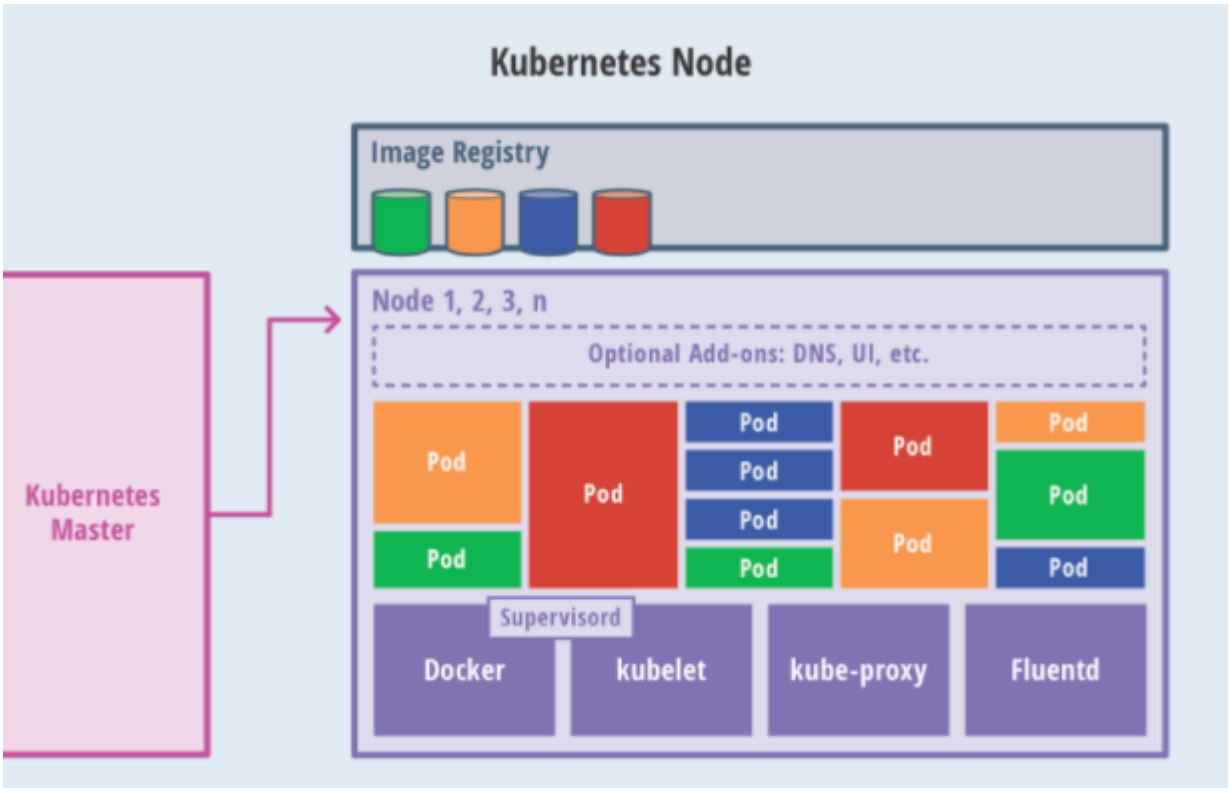
Kubernetes includes support for persistent storage connected to stateless application containers. There is support for Amazon Web Services EBS, Google Cloud Platform persistent disks, and many, many more.

High Availability Features.

Kubernetes is planet scale. This requires special attention to high availability features such as multi-master or cluster federation. Cluster federation allows linking clusters together so that if one cluster goes down containers can automatically move to another cluster.



These key features make Kubernetes well suited for running different application architectures from monolithic web applications, to highly distributed microservice applications, and even batch driven applications.



Kubernetes “clusters” are composed of “nodes.” The term “cluster” refers to “nodes” in the aggregate. “Cluster” refers to the entire running system. A “node” is a worker machine Kubernetes. A “node” may be a VM or physical machine. Each node has software configured to run containers managed by

machine. Each node has software configured to run containers managed by Kubernetes' control plane.

The control plane is the set of APIs and software that Kubernetes users interact with. The control plane services run on master nodes. Clusters may have multiple masters for high availability scenarios.

Horizontal Scaling vs Vertical Scaling & Multi-Tenancy

In this section we will learn about various Cloud Concepts:- a) Scaling on the cloud b) Degradation of Services c) Availability Vs Durability on the Cloud d) Single & Multi-tenancy Applications e) Types of Cloud Deployments

In this section we will learn various Cloud Concepts.

1. Scaling on the cloud
2. Degradation of Services
3. Availability Vs Durability on the Cloud
4. Single & Multi-tenancy Applications
5. Types of Cloud Deployments

Scaling Out / Horizontal Scaling vs Scaling up/ Vertical Scaling

Horizontal scaling means that you scale by adding more machines into your pool of resources. Eg. As traffic goes up you add more web servers to take on the traffic.

Vertical scaling means that you scale by adding more power (CPU, RAM) to an existing machine.

On the Cloud you Scale horizontally !

Graceful Degradation of Services

Applications need to have a way to handle excessive load and handle failure gracefully no matter if it's the application or a dependent service under load.

The Site Reliability Engineering handbook describes graceful degradation in applications as offering "responses that are not as accurate as or that contain less data than normal responses, but that are easier to compute" when under excessive load.

Modern applications are build with breakers that do not overload other microservices if they malfunction. They also need to have sufficient telemetry that helps in troubleshooting and help in identifying what is causing a

problem.

If you take a moment to think of all the apps on your phone that you use when was the last time it had an outage ? The probability of an outage is so less likely because the system design and architecture of modern applications take into account every single point of failure from an application, database, Load balancer and even the infrastructure.

Graceful failure is about making sure that at the event of failure at any level that there is a backup that takes the responsibility of the component that failed and at the end there is no perceived customer impact. Note the word 'perceived', Lets take for example that a large ecommerce / retail website calculates the tax for each shipment before you place the order. If the tax microservice goes down then maybe you do not charge the customer tax for that one transaction the cost will be covered by the organization.

Availability Vs Durability on the Cloud

Availability

Availability and durability are two very different aspects of data accessibility. Most cloud services offer a 99.9999% availability. This varies by provider and by each service that is offered by the provider.

Availability refers to system uptime, i.e. the storage system is operational and can deliver data upon request. Historically, this has been achieved through hardware redundancy so that if any component fails, access to data will prevail.

Durability

Durability, on the other hand, refers to long-term data protection, i.e. the stored data does not suffer from bit rot, degradation or other corruption. Rather than focusing on hardware redundancy, it is concerned with data redundancy so that data is never lost or compromised.

Availability and durability serve different objectives. For data centers, availability/uptime is a key metric for operations as any minute of downtime is costly. The measurement focuses on storage system availability. But what happens when a component, system or even the data center goes down? Will your data be intact when the fault is corrected?

This illustrates the equal importance of data durability. When an availability fault is corrected, it is essential that access to uncorrupted data is restored. With the explosion of data created, the potential of mining, and growing needs for longer retention rates (for everything) you can imagine how this is paramount for business success.

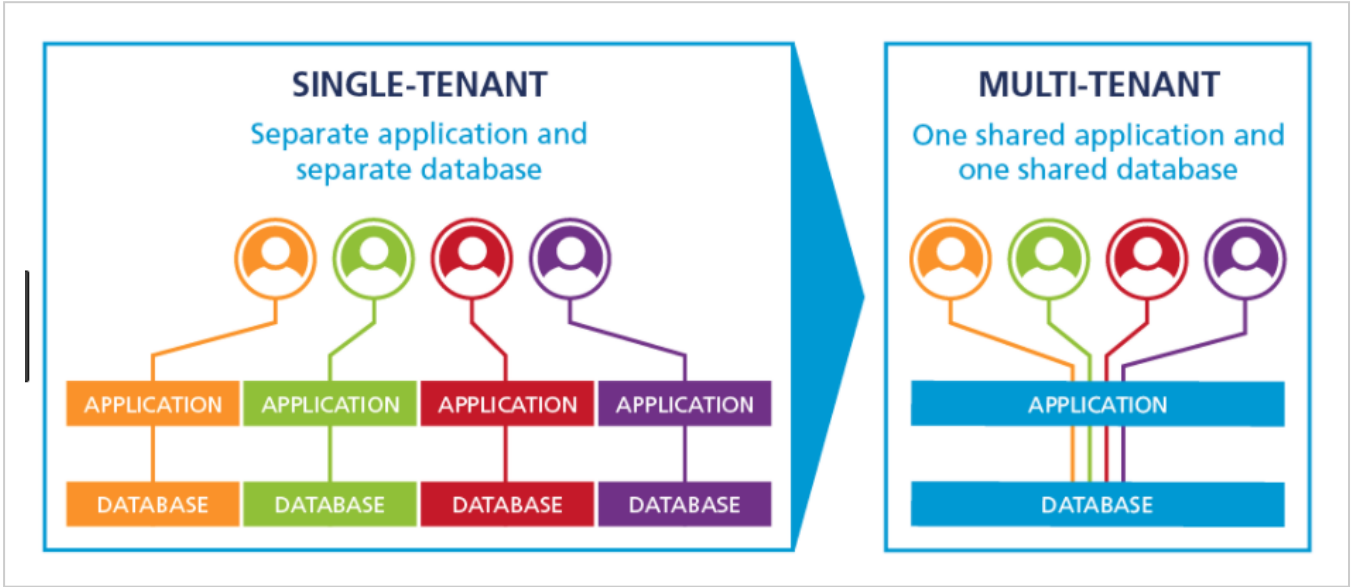
Consider the potential competitive, financial or even legal impact of not being able to retrieve the archived master/reference copy of data. Hence, both data availability and data durability are essential for short- and long-term business success.

Single vs Multi Tenancy Cloud Deployment

Single tenancy or multi-tenancy is in context with SaaS applications. A SaaS application can either choose to be one to the other. Most modern applications choose to be multi-tenant applications.

Multi-tenancy

Multi-tenancy means that a single instance of the software and all of the supporting infrastructure serve multiple end-customers. Each customer shares the software application and also shares a single database. The data is tagged in the database as belonging to one customer or another, and the software is smart enough to know who the data belongs to.



Single tenancy

In a Single tenancy approach a single instance of the software and all of the

supporting infrastructure serves a single customer. With single tenancy, each

customer has their own independent database and instance of the application software as well. With this option, there's essentially no sharing going on. Everyone has their own, separate from everyone else.

Think about a SaaS application that is serving customers like coke and pepsi, sometimes these companies would care that they are not sharing the database tier in particular so that there is no possibility for information to leak. If this was a concern then they would go for a Single tenancy rather than a multi tenancy approach.

With single-tenant architectures, the marginal cost of adding new customers never goes down. Maintenance With single-tenant, maintenance is incremental. If you botch it, you typically only take down a single team. Instead of deploying a single app update, you're deploying N app updates. Instead of migrating one database, you're migrating N databases. On the surface level it sounds like this would create more work for you, but since the systems are isolated and identical most of that work can be automated. All you need is a bit of tooling to orchestrate the updates.

Resilience

Single-tenant creates resilience at the system level. Outside of DDOS attacks at the DNS level there are very few ways to take down the entire system. A team may experience problems, but it's unlikely those problems will extend beyond that single instance.

Types of Cloud Deployments

Hybrid cloud is an environment that uses a mix of on-premises, private cloud and /or third-party, public cloud services with orchestration between the two platforms.

Multi-cloud

A multi-cloud strategy is the use of two or more cloud services. While a multi-cloud deployment can refer to any implementation of multiple software as a service (SaaS) or platform as a service (PaaS) cloud offerings, today, it generally refers to a mix of public infrastructure as a service (IaaS) environments, such as Amazon Web Services, Microsoft Azure, Oracle Cloud

Infrastructure.

Multi-cloud was, and still is, seen as a way to prevent data loss or downtime due to a localized component failure in the cloud. The ability to avoid vendor lock-in was also an early driver of multi-cloud adoption. A multi-cloud strategy also offers the ability to select different cloud services or features from different providers.

The pros and cons of a multi-cloud strategy

There are several commonly cited advantages to multi-cloud computing, such as the ability to avoid vendor lock-in, the ability to find the optimal cloud service for a particular business or technical need and increased redundancy.

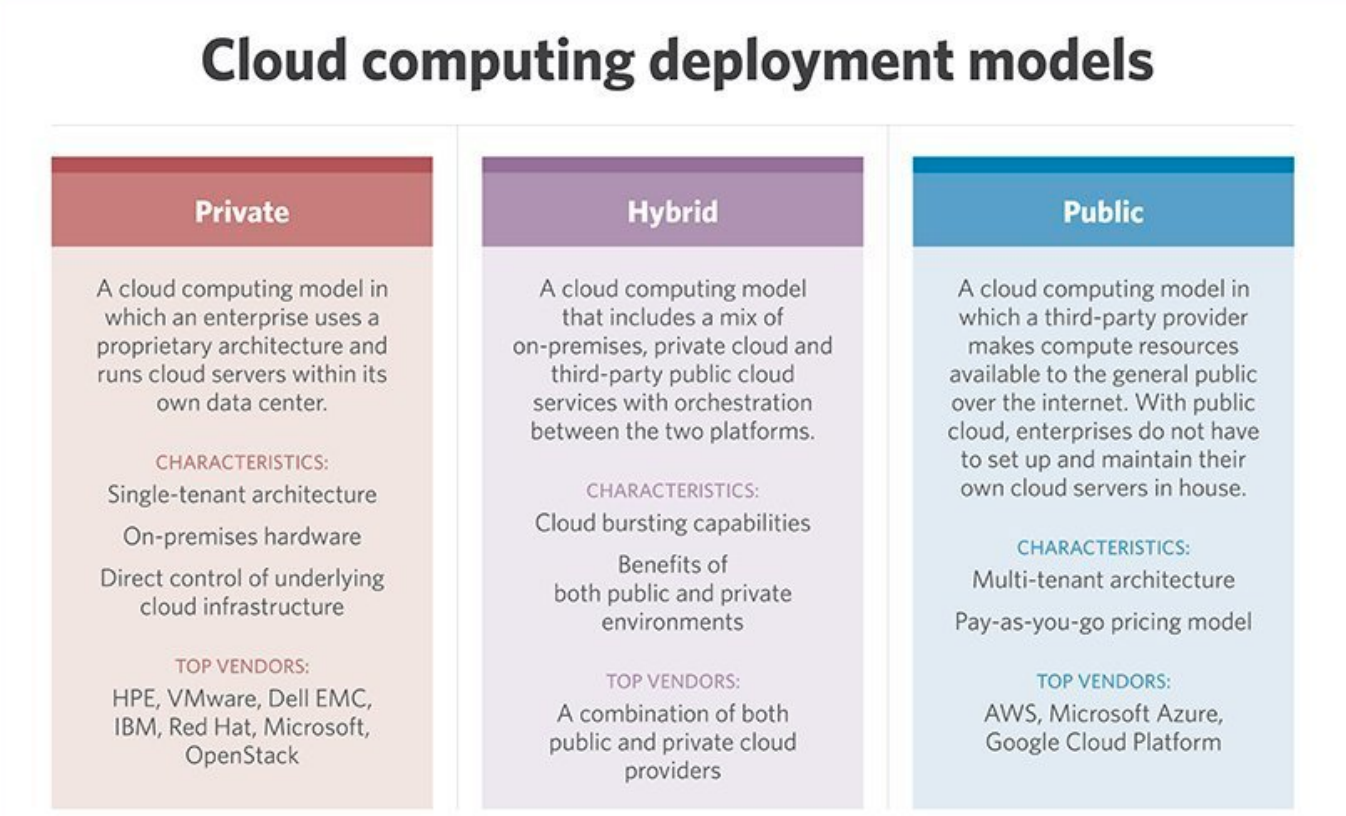
However, there are some potential drawbacks. For example, most public cloud providers offer volume discounts, where prices are reduced as customers buy more of a particular service. It becomes more difficult for an organization to qualify for those discounts when it doesn't concentrate its business with a single cloud provider.

In addition, a multi-cloud deployment requires an IT staff to have multiple kinds of cloud platform or provider expertise. Workload or application management in multi-cloud computing can also be a challenge, as information moves from one cloud platform to another.

Multi-cloud computing vs. Hybrid cloud computing

Multi-cloud and hybrid cloud computing are similar, but different IT infrastructure models. In general, hybrid cloud refers to a cloud computing environment that uses a mix of an on-premises, private cloud and a third-party, public cloud, with orchestration between the two. An enterprise often adopts hybrid cloud to achieve a specific task, such as the ability to run workloads in house, and then burst into the public cloud when compute demands spike.

However, multi-cloud doesn't preclude hybrid cloud, and a hybrid cloud could be part of a multi-cloud deployment. The two models are not an either/or situation; it simply depends on what a business hopes to achieve.



The basic benefits of multi-tenant are:

Resilience

Both architectures offer their own form of resilience. Multi-tenant creates resilience at the team level. Each team is serviced by multiple instances, spread across multiple regions/zones, and hosted behind load balancers. A team is unlikely to experience issues unless there is a system-wide outage.

Maintenance

With multi-tenant, deploys are typically all or nothing. Maintenance on multi-tenant systems can be scary. You push out a single update, and every customer is immediately on the new system. If you botch it you take down the entire system.

Easy Upgrades

Instead of a vendor needing to update every instance of their software across a large number of servers, they are able to update a single, central application or codebase and have the changes instantly available to all users. With a multi-tenant application the process for spinning up a new cloud and application on behalf of a new customer is incredibly easy and can be done very quickly.

Easy Customizations

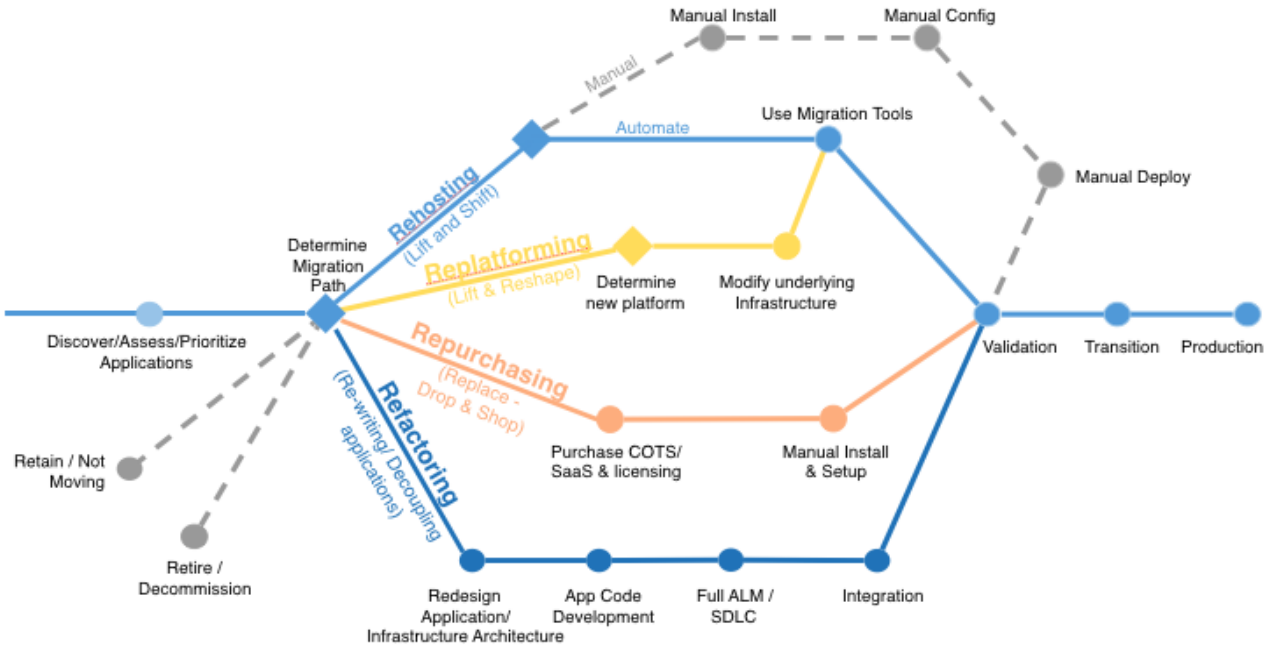
With multi-tenancy based applications you can provide an additional layer to allow for customizations while still maintaining an underlying codebase which remains constant for all users, including of course, all new customers.

Ongoing Cost Savings

Multi-tenancy speeds up upgrades, saves time (and also cost) but in addition the server / cloud requirements for a multi-tenancy application cost much less. The opportunity to save money takes many forms and becomes greater as the application scales up. This reduces the cost of doing business for the vendor and savings can be passed onto customers.

Application Migration Strategies

Learn the common cloud migration strategies.



There six different migration strategies organizations use to migrate applications to the cloud. These strategies build upon the 5 R's that Gartner outlined here in 2011.

The most common application migration strategies we see are:

Re-Hosting

We find that many early net-new development initiatives like startups build cloud-native applications, but in a large legacy migration scenario where the organization is looking to scale its migration quickly to meet a business case, we find that the majority of applications are rehosted. GE Oil & Gas, found that, even without implementing any cloud optimizations, it could save roughly 30 percent of its costs by rehosting with a cloud provider. White paper here.

Most rehosting can be automated with tools available from cloud providers like AWS, Azure, Oracle etc although some customers prefer to do this

manually as they learn how to apply their legacy systems to the new cloud platform.

Applications are easier to optimize/re-architect once they're already running in the cloud. Partly because organizations will have developed better skills to do so, and partly because the hard part designing the network topology, migrating the application, data, and traffic—has already been done.

Replatforming

This is what is called "Lift-and-Shift". Here you might make a few cloud (or other) optimizations in order to achieve some tangible benefit, but you aren't otherwise changing the core architecture of the application. You may be looking to reduce the amount of time you spend managing database instances by migrating to a database-as-a-service platform like Amazon Relational Database Service (Amazon RDS), or migrating your application to a fully managed platform like Amazon Elastic Beanstalk.

A large media company migrated hundreds of web servers it ran on-premises to AWS, and, in the process, it moved from WebLogic (a Java application container that requires an expensive license) to Apache Tomcat, an open-source equivalent. This media company saved millions in licensing costs on top of the savings and agility it gained by migrating to AWS.

Repurchasing

Moving to a different product. Moving a CRM to Salesforce.com, an HR system to Workday, a CMS to Drupal, and so on.

Refactoring / Re-Architecting

Re-imagining how the application is architected and developed, typically using cloud-native features. This is typically driven by a strong business need to add features, scale, or performance that would otherwise be difficult to achieve in the application's existing environment.

Customers using this approach are looking to migrate from a monolithic architecture to a service-oriented (or server-less) architecture to boost agility or improve business continuity. This pattern tends to be the most expensive, but, if you have a good product-market fit, it can also be the most beneficial.

Retire

Once there is an audit done to discovered everything in your environment,

you might ask each functional area who owns each application. It has been found that as much as 10% to 20% of an enterprise IT portfolio is no longer useful, and can simply be turned off. These savings can boost the business case, direct your team's scarce attention to the things that people use and lessen the surface area you have to secure.

Retain

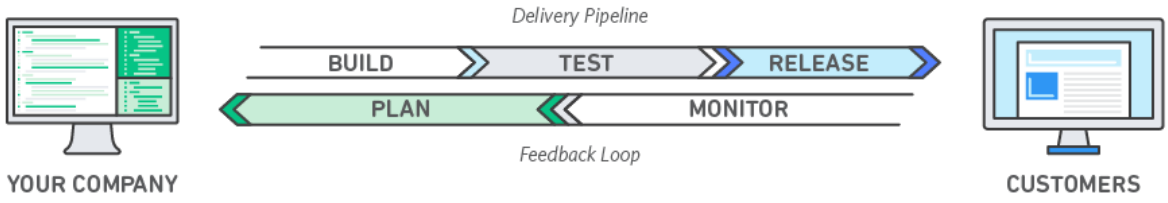
Usually this means "revisit" or do nothing (for now). Maybe you're still riding out some depreciation, aren't ready to prioritize an application that was recently upgraded, or are otherwise not inclined to migrate some applications. You should only migrate what makes sense for the business; and, as the gravity of your portfolio changes from on-premises to the cloud, you'll probably have fewer reasons to retain.

What is DevOps ? And why is it important on the Cloud ?

Fundamentals of DevOps. Why is DevOps important for any cloud move?

What is DevOps? DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

The reason why we talk about DevOps here is that every successful cloud deployment needs to have the team move to a DevOps model to fully utilize the native cloud capabilities. Under a DevOps model, development and operations teams are no longer "siloed." Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function. Quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle.



Believe it or not, most if not all tech companies carry out the entity of the above diagram literally 100s of time in just everyday. And that by itself is one of the fundamental reason of their success.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them

operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team's velocity.

Benefits of DevOps

Speed

Move at higher velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker. Since speed and quick iteration is one of the key benefits of moving to the cloud DevOps is important.

Rapid Delivery

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage. Continuous integration & continuous delivery are practices that automate the software release process, from build to deploy.

Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

Improved Collaboration

Build more effective teams under a DevOps cultural model, which emphasizes

values such as ownership and accountability. Developers and operations

teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

Security

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

Why DevOps Matters

Software and the Internet have transformed the world and its industries, from shopping to entertainment to banking. Software no longer merely supports a business; rather it becomes an integral component of every part of a business.

Companies interact with their customers through software delivered as online services or applications and on all sorts of devices. They also use software to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations. In a similar way that physical goods companies transformed how they design, build, and deliver products using industrial automation throughout the 20th century, companies in today's world must transform how they build and deliver software.

How to Adopt a DevOps Model

DevOps Cultural Philosophy

Transitioning to DevOps requires a change in culture and mindset. At its simplest, DevOps is about removing the barriers between two traditionally siloed teams, development and operations. In some organizations, there may not even be separate development and operations teams; engineers may do both. With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations.

They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers. They take full ownership for

their services, often beyond where their stated roles or titles have traditionally been scoped by thinking about the end customer's needs and how they can contribute to solving those needs.

Quality assurance and security teams may also become tightly integrated with these teams. Organizations using a DevOps model, regardless of their organizational structure, have teams that view the entire development and infrastructure lifecycle as part of their responsibilities.

DevOps Practices

There are a few key practices that help organizations innovate faster through automating and streamlining the software development and infrastructure management processes. Most of these practices are accomplished with proper tooling.

One fundamental practice is to perform very frequent but small updates. This is how organizations innovate faster for their customers. These updates are usually more incremental in nature than the occasional updates performed under traditional release practices. Frequent but small updates make each deployment less risky. They help teams address bugs faster because teams can identify the last deployment that caused the error. Although the cadence and size of updates will vary, organizations using a DevOps model deploy updates much more often than organizations using traditional software development practices.

Organizations might also use a microservices architecture to make their applications more flexible and enable quicker innovation. The microservices architecture decouples large, complex systems into simple, independent projects. Applications are broken into many individual components (services) with each service scoped to a single purpose or function and operated independently of its peer services and the application as a whole. This architecture reduces the coordination overhead of updating applications, and when each service is paired with small, agile teams who take ownership of each service, organizations can move more quickly.

However, the combination of microservices and increased release frequency leads to significantly more deployments which can present operational challenges. Thus, DevOps practices like continuous integration and continuous

delivery solve these issues and let organizations deliver rapidly in a safe and reliable manner. Infrastructure automation practices, like infrastructure as code and configuration management, help to keep computing resources elastic and responsive to frequent changes. In addition, the use of monitoring and logging helps engineers track the performance of applications and infrastructure so they can react quickly to problems.

Together, these practices help organizations deliver faster, more reliable updates to their customers.

Operational Excellence on the Cloud

Operational Excellence on the Cloud. Learn the fundamental design principles of operating on the Cloud.

Operations teams need to understand their business and customer needs so they can effectively and efficiently support business outcomes. They create and use procedures to respond to operational events and validate their effectiveness to support business needs. They also collect metrics that are used to measure the achievement of desired business outcomes.

Operational Excellence

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.

The operational excellence pillar provides an overview of design principles, best practices, and questions

Design Principles

There are six design principles for operational excellence in the cloud:

Perform operations as code:

In the cloud, you can apply the same engineering discipline that you use for application code to your entire environment. You can define your entire workload (applications, infrastructure, etc.) as code and update it with code. You can script your operations procedures and automate their execution by triggering them in response to events. By performing operations as code, you limit human error and enable consistent responses to events.

Annotate documentation:

In an on-premises environment, documentation is created by hand, used by people, and hard to keep in sync with the pace of change. In the cloud, you can automate the creation of documentation after every build (or automatically annotate hand-crafted documentation). Annotated documentation can be used by people and systems. Use annotations as an

input to your operations code.

Make frequent, small, reversible changes: Design workloads to allow components to be updated regularly. Make changes in small increments that can be reversed if they fail (without affecting customers when possible).

Refine operations procedures frequently:

As you use operations procedures, look for opportunities to improve them. As you evolve your workload, evolve your procedures appropriately. Set up regular game days to review and validate that all procedures are effective and that teams are familiar with them.

Anticipate failure:

Perform “pre-mortem” exercises to identify potential sources of failure so that they can be removed or mitigated.

Test your failure scenarios and validate your understanding of their impact. Test your response procedures to ensure that they are effective and that teams are familiar with their execution. Set up regular game days to test workloads and team responses to simulated events.

Learn from all operational failures:

Drive improvement through lessons learned from all operational events and failures. Share what is learned across teams and through the entire organization.

Definition

There are three best practice areas for operational excellence in the cloud:

1. Prepare
2. Operate
3. Evolve

Operations teams need to understand their business and customer needs so they can effectively and efficiently support business outcomes. Operations creates and uses procedures to respond to operational events and validates their effectiveness to support business needs. Operations collects metrics that are used to measure the achievement of desired business outcomes.

Everything continues to change—your business context, business priorities, customer needs, etc. It's important to design operations to support evolution over time in response to change and to incorporate lessons learned through their performance.

Best Practices

Prepare

Effective preparation is required to drive operational excellence. Business success is enabled by shared goals and understanding across the business, development, and operations. Common standards simplify workload design and management, enabling operational success. Design workloads with mechanisms to monitor and gain insight into application, platform, and infrastructure components, as well as customer experience and behavior.

Create mechanisms to validate that workloads, or changes, are ready to be moved into production and supported by operations. Operational readiness is validated through checklists to ensure a workload meets defined standards and that required procedures are adequately captured in runbooks and playbooks. Validate that there are sufficient trained personnel to effectively support the workload. Prior to transition, test responses to operational events and failures. Practice responses in supported environments through failure injection and game day events.

AWS enables operations as code in the cloud and the ability to safely experiment, develop operations procedures, and practice failure. Using AWS CloudFormation enables you to have consistent, templated, sandbox development, test, and production environments with increasing levels of operations control.

AWS enables visibility into your workloads at all layers through various log collection and monitoring features. Data on use of resources, application programming interfaces (APIs), and network flow logs can be collected using Amazon CloudWatch, AWS CloudTrail, and VPC Flow Logs.

You can use the CloudWatch Logs agent, or the collectd plugin, to aggregate information about the operating system into CloudWatch. The following questions focus on preparing considerations for operational excellence :-

OPS 1: What factors drive your operational priorities ?

OPS 2: How do you design your workload to enable operability ?

OPS 3: How do you know that you are ready to support a workload ?

Implement the minimum number of architecture standards for your workloads. Balance the cost to implement a standard against the benefit to the workload and the burden upon operations. Reduce the number of supported standards to reduce the chance that lower-than-acceptable standards will be applied by error.

Operations personnel are often constrained resources. Invest in scripting operations activities to maximize the productivity of operations personnel, minimize error rates, and enable automated responses. Adopt deployment practices that take advantage of the elasticity of the cloud to facilitate pre-deployment of systems for faster implementations.

Operate

Successful operation of a workload is measured by the achievement of business and customer outcomes. Define expected outcomes, determine how success will be measured and identify the workload and operations metrics that will be used in those calculations to determine if operations are successful. Consider that operational health includes both the health of the workload and the health and success of the operations acting upon the workload (for example, deployment and incident response). Establish baselines from which improvement or degradation of operations will be identified, collect and analyze your metrics, and then validate your understanding of operations success and how it changes over time.

Use collected metrics to determine if you are satisfy customer and business needs, and identify areas for improvement. Efficient and effective management of operational events is required to achieve operational excellence. This applies to both planned and unplanned operational events. Use established runbooks for well-understood events, and use playbooks to aid in the resolution of other events. Prioritize responses to events based on their business and customer impact. Ensure that if an alert is raised in response to an event, there is an associated process to be executed, with a specifically identified owner.

Define in advance the personnel required to resolve an event and include

escalation triggers to engage additional personnel, as it becomes necessary, based on impact (that is, duration, scale, and scope). Identify and engage individuals with the authority to decide on courses of action where there will be a business impact from an event response not previously addressed. Communicate the operational status of workloads through dashboards and notifications that are tailored to the target audience (for example, customer, business, developers, operations) so that they may take appropriate action, so that their expectations are managed, and so that they are informed when normal operations resume.

Determine the root cause of unplanned events and unexpected impacts from planned events. This information will be used to update your procedures to mitigate future occurrence of events. Communicate root cause with affected communities as appropriate. In AWS, you can generate dashboard views of your metrics collected from workloads and natively from AWS. You can leverage CloudWatch or third-party applications to aggregate and present business, workload, and operations level views of operations activities. AWS provides workload insights through logging capabilities including AWS X-Ray, CloudWatch, CloudTrail, and VPC Flow Logs enabling the identification of workload issues in support of root cause analysis and remediation.

The following questions focus on operate considerations for operational excellence.

OPS 4: What factors drive your understanding of operational health? OPS 5: How do you manage operational events?

Routine operations, as well as responses to unplanned events, should be automated. Manual processes for deployments, release management, changes, and rollbacks should be avoided. Releases should not be large batches that are done infrequently. Rollbacks are more difficult in large changes. Failing to have a rollback plan, or the ability to mitigate failure impacts, will prevent continuity of operations. Align metrics to business needs so that responses are effective at maintaining business continuity. One-time, decentralized metrics with manual responses will result in greater disruption to operations during unplanned events.

Evolve

Evolution of operations is required to sustain operational excellence. Dedicate

work cycles to making continuous incremental improvements. Regularly evaluate and prioritize opportunities for improvement (for example, feature requests, issue remediation, and compliance requirements), including both the workload and operations procedures.

Include feedback loops within your procedures to rapidly identify areas for improvement and capture learnings from the execution of operations. Share lessons learned across teams to share the benefits of those lessons. Analyze trends within lessons learned and perform cross-team retrospective analysis of operations metrics to identify opportunities and methods for improvement.

Implement changes intended to bring about improvement and evaluate the results to determine success. With AWS Developer Tools you can implement continuous delivery build, test, and deployment activities that work with a variety of source code, build, testing, and deployment tools from AWS and third parties. The results of deployment activities can be used to identify opportunities for improvement for both deployment and development. You can perform analytics on your metrics data integrating data from your operations and deployment activities, to enable analysis of the impact of those activities against business and customer outcomes. This data can be leveraged in cross-team retrospective analysis to identify opportunities and methods for improvement. The following question focuses on evolve considerations for operational excellence.

OPS 6: How do you evolve operations?

Successful evolution of operations is founded in: frequent small improvements; providing safe environments and time to experiment, develop, and test improvements; and environments in which learning from failures is encouraged. Operations support for sandbox, development, test, and production environments, with increasing level of operational controls, facilitates development and increases the predictability of successful results from changes deployed into production.

Key AWS Services

The AWS service that is essential to operational excellence is AWS CloudFormation, which you can use to create templates based on best practices. This enables you to provision resources in an orderly and consistent fashion from your development through production environments.

The following services and features support the three areas of operational excellence:

Prepare:

AWS Config and AWS Config rules can be used to create standards for workloads and to determine if environments are compliant with those standards before being put into production.

Operate:

Amazon CloudWatch allows you to monitor the operational health of a workload.

Evolve:

Amazon Elasticsearch Service (Amazon ES) allows you to analyze your log data to gain actionable insights quickly and securely

Continuous Integration & Continuous Delivery

DevOps has come the new methods of Continuous Integration, Continuous Delivery, (CI/CD) and Continuous Deployment. Understand How CI/CD helps in automating on the Cloud.

With the rise of DevOps has come the new methods of Continuous Integration, Continuous Delivery, (CI/CD) and Continuous Deployment.

As discussed in the previous topic conventional software development and delivery methods are rapidly becoming obsolete. Historically, in the agile age, most companies would deploy/ship software in monthly, quarterly, bi-annual, or even annual releases. Now however, in the DevOps era, weekly, daily, and even multiple times a day is the norm.

With the DevOps paradigm shift most teams have automated processes to check in code and deploy to new environments. This has been coupled with a focus on automating the process along the way.

Continuous Integration (CI)

With continuous integration, developers frequently integrate their code into a main branch of a common repository. Rather than building features in isolation and submitting each of them at the end of the cycle, a developer will strive to contribute software work products to the repository several times on any given day.

The big idea here is to reduce integration costs by having developers do it sooner and more frequently. In practice, a developer will often discover boundary conflicts between new and existing code at the time of integration. If it's done early and often, the expectation is that such conflict resolutions will be easier and less costly to perform.

Of course, there are trade-offs. This process change does not provide any additional quality assurances. Indeed, many organizations find that such integration becomes more costly since they rely on manual procedures to ensure that new code doesn't introduce new bugs, and doesn't break existing

code. To reduce friction during integration tasks, continuous integration relies on test suites and an automated test execution. It's important, however, to realize that automated testing is quite different from continuous testing.

The goal of CI is to refine integration into a simple, easily-repeatable everyday development task that will serve to reduce overall build costs and reveal defects early in the cycle. Success in CI will depend on changes to the culture of the development team so that there is incentive for readiness, frequent and iterative builds, and eagerness to deal with bugs when they are found much earlier.

Continuous Delivery (CD)

Continuous delivery is actually an extension of CI, in which the software delivery process is automated further to enable easily and confident deployments into production at any time. A mature continuous delivery process exhibits a codebase that is always deployable on the spot. With CD, software release becomes a routine event without emotion or urgency. Teams proceed with daily development tasks in the confidence that they can build a production-grade release any old time they please without elaborate orchestration .

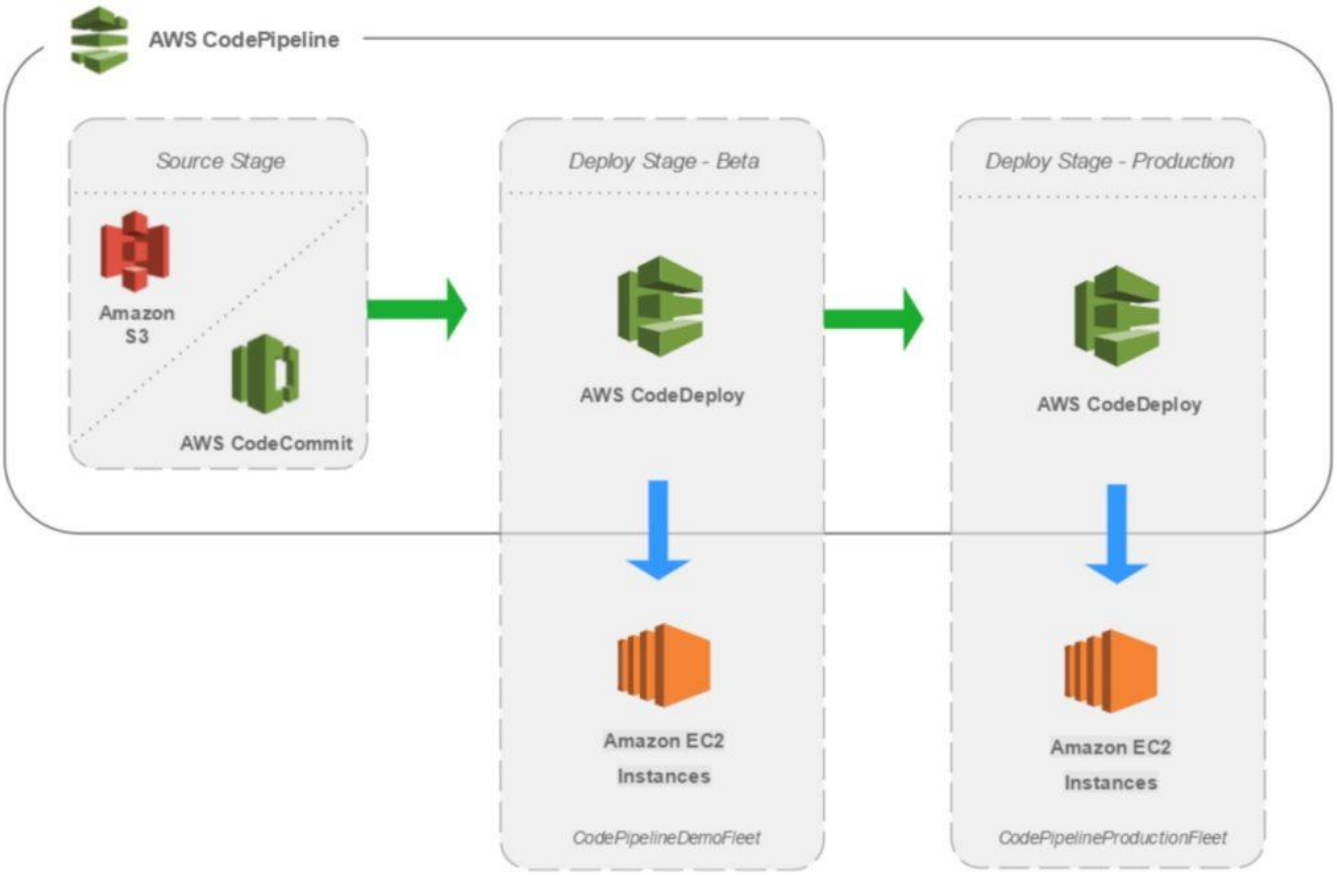
CD depends centrally on a deployment pipeline by which the team automates the testing and deployment processes. This pipeline is an automated system that executes a progressive set of test suites against the build. CD is highly automatable and in some cloud-computing environments easily configurable.

In each segment in the pipeline, the build may fail a critical test and alert the team. Otherwise, it continues on to the next test suite, and successive test passes will result in automatic promotion to the next segment in the pipeline. The last segment in the pipeline will deploy the build to a production-equivalent environment. This is a comprehensive activity, since the build, the deployment, and the environment are all exercised and tested together. The result is a build that is deployable and verifiable in an actual production environment.

A solid exhibit of a modern CI/CD pipeline is available on on the cloud with AWS. Amazon is one of the cloud-computing providers that offers an impressive CI/CD pipeline environment, and provides a walk-through

procedure in which you can choose from among its many development

resources and link them together in a pipeline that is readily configurable and easily monitored.



Continuous deployment extends continuous delivery so that the software build will automatically deploy if it passes all tests. In such a process, there is no need for a person to decide when and what goes into production.

The last step in a CI/CD system will automatically deploy whatever build components/packages successfully exit the delivery pipeline. Such automatic deployments can be configured to quickly distribute components, features, and fixes to customers, and provide clarity on precisely what has is presently in production.

Organizations that employ continuous deployment will likely benefit from very quick user feedback on new deployments. Features are quickly delivered to users, and any defects that become evident can be handled promptly. Quick user response on unhelpful or misunderstood features will help the team refocus and avoid devoting more effort into to functional area that is unlikely to produce a good return return on that investment.

Security In a Multi-tenant Environment

Learn about protecting your data on the cloud - a) data in transit b) data at rest c) protecting your credentials d) securing your Application.

Security Best Practices for the Cloud In a multi-tenant environment.

We as cloud architects often express concerns about security. Security should be implemented in every layer of the cloud application architecture. Physical security is typically handled by your service provider, which is an additional benefit of using the cloud. Network and application-level security is your responsibility and you should implement the best practices as applicable to your business. It is recommended to take advantage of these tools and features mentioned to implement basic security and then implement additional security best practices using standard methods as appropriate or as they see fit.

Protect your data in transit

If you need to exchange sensitive or confidential information between a browser and a web server, configure SSL on your server instance. You'll need a certificate from an external certification authority like VeriSign or Entrust. The public key included in the certificate authenticates your server to the browser and serves as the basis for creating the shared session key used to encrypt the data in both directions.

Creating a Virtual Private Cloud by making a few command line calls (using VPC). This will enable you to use your own logically isolated resources within the AWS cloud, and then connect those resources directly to your own datacenter using industry-standard encrypted IPSec VPN connections. You can also setup an OpenVPN server on an Amazon EC2 instance and install the OpenVPN client on all user PCs.

Protect your data at rest

If you are concerned about storing sensitive and confidential data in the cloud, you should encrypt the data (individual files) before uploading it to the

cloud. For example, encrypt the data using any open source or commercial

PGP based tools before storing it as Amazon S3 objects and decrypt it after download.

This is often a good practice when building HIPAA-Compliant applications that need to store Protected Health Information (PHI). On Amazon EC2, file encryption depends on the operating system. Amazon EC2 instances running Windows can use the built-in Encrypting File System (EFS) feature.

This feature will handle the encryption and decryption of files and folders automatically and make the process transparent to the users . However, despite its name, EFS doesn't encrypt the entire file system; instead, it encrypts individual files. If you need a full encrypted volume, consider using the open-source TrueCrypt product; this will integrate very well with NTFS-formatted EBS volumes. Amazon EC2 instances running Linux can mount EBS volumes using encrypted file systems using variety of approaches (EncFS, Loop-AES , dm-crypt, TrueCrypt).

Likewise, Amazon EC2 instances running OpenSolaris can take advantage of ZFS Encryption Support. Regardless of which approach you choose, encrypting files and volumes in Amazon EC2 helps protect files and log data so that only the users and processes on the server can see the data in clear text, but anything or anyone outside the server see only encrypted data. No matter which operating system or technology you choose, encrypting data at rest presents a challenge: managing the keys used to encrypt the data. If you lose the keys, you will lose your data forever and if your keys become compromised, the data may be at risk.

Therefore, be sure to study the key management capabilities of any products you choose and establish a procedure that minimizes the risk of losing keys. Besides protecting your data from eavesdropping, also consider how to protect it from disaster. Take periodic snapshots of Amazon EBS volumes to ensure it is highly durable and available. Snapshots are incremental in nature and stored on Amazon S3 (separate geo-location) and can be restored back with a few clicks or command line calls.

Protect your AWS credentials

AWS supplies two types of security credentials: AWS access keys and X.509

certificates. Your AWS access key has two parts: your access key ID and your secret access key. When using the REST or Query API, you have to use your secret access key to calculate a signature to include in your request for authentication. To prevent in-flight tampering, all requests should be sent over HTTPS. If your Amazon Machine Image (AMI) is running processes that need to communicate with other AWS web services, one common design mistake is embedding the AWS credentials in the AMI.

Instead of embedding the credentials, they should be passed in as arguments during launch and encrypted before being sent over the wire. If your secret access key becomes compromised, you should obtain a new one by rotating to a new access key ID. As a good practice, it is recommended that you incorporate a key rotation mechanism into your application architecture so that you can use it on a regular basis or occasionally (when disgruntled employee leaves the company) to ensure compromised keys can't last forever. Alternately, you can use X.509 certificates for authentication to certain AWS services.

The certificate file contains your public key in a base64-encoded DER certificate body. A separate file contains the corresponding base64-encoded PKCS private key. AWS supports multi-factor authentication as an additional protector for working with your account information on AWS Management Console.

Manage multiple Users and their permissions with IAM AWS Identity and Access Management (IAM) enables you to create multiple Users and manage the permissions for each of these Users within your AWS Account. A User is an identity (within your AWS Account) with unique security credentials that can be used to access AWS Services.

IAM eliminates the need to share passwords or access keys, and makes it easy to enable or disable a User's access as appropriate. IAM enables you to implement security best practices, such as least privilege, by granting unique credentials to every User within your AWS account and only grant permission to access the AWS Services and resources required for the Users to perform their job.

IAM is secure by default; new Users have no access to AWS until permissions are explicitly granted. IAM is natively integrated into most AWS Services. No

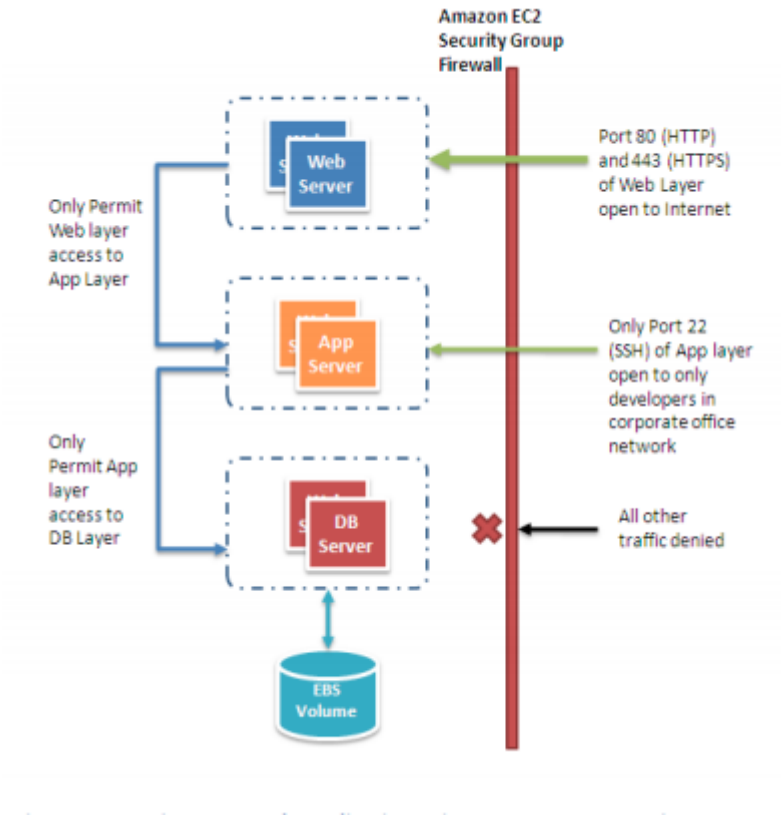
service APIs have changed to support IAM, and applications and tools built on top of the AWS service APIs will continue to work when using IAM.

Applications only need to begin using the access keys generated for a new User.

You should minimize the use of your AWS Account credentials as much as possible when interacting with your AWS Services and take advantage of IAM User credentials to access AWS Services and resources.

Secure your Application

Every Amazon EC2 instance is protected by one or more security groups 43, named sets of rules that specify which ingress (i.e., incoming) network traffic should be delivered to your instance. You can specify TCP and UDP ports, ICMP types and codes, and source addresses. Security groups give you basic firewall-like protection for running instances. For example, instances that belong to a web application can have the following security group settings:



Another way to restrict incoming traffic is to configure software-based firewalls on your instances. Windows instances can use the built-in firewall. Linux instances can use netfilter and iptables. Over time, errors in software are discovered and require patches to fix.

You should ensure the following basic guidelines to maximize security of your

application:

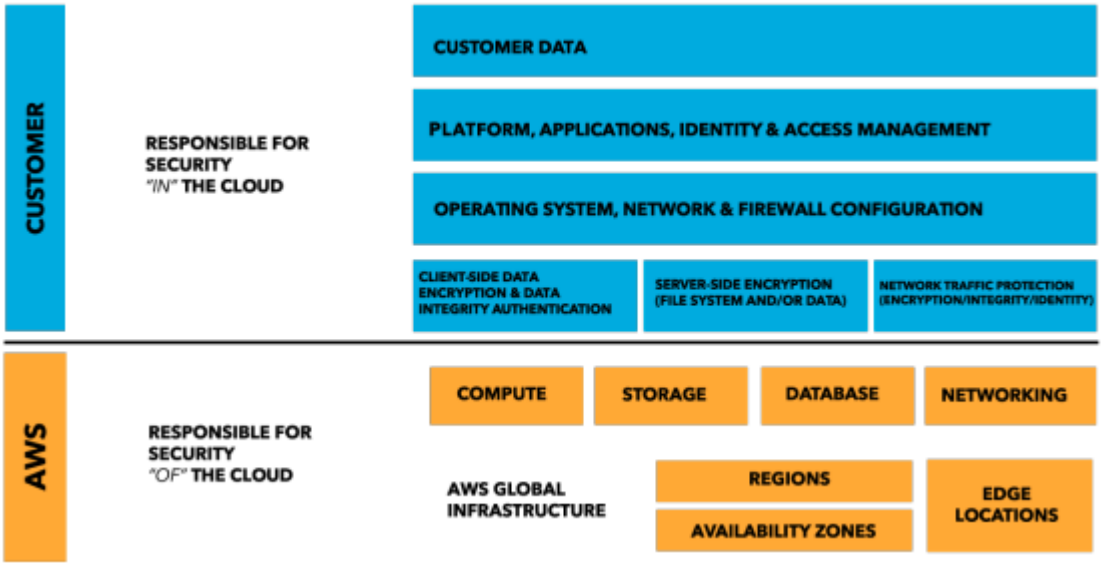
1. Regularly download patches from the vendor's web site and update your AMIs
   Redeploy instances from the new AMIs and test your applications to ensure the patches don't break anything. Ensure that the latest AMI is deployed across all instances
2. Invest in test scripts so that you can run security checks periodically and automate the process
3. Ensure that the third-party software is configured to the most secure settings
4. Never run your processes as root or Administrator login unless absolutely necessary

All the standard security practices pre-cloud era like adopting good coding practices, isolating sensitive data are still applicable and should be implemented. In retrospect, the cloud abstracts the complexity of the physical security from you and gives you the control through tools and features so that you can secure your application.

Cloud Security - Shared Security Responsibility Model

Before we go into the details of how AWS secures its resources, we should talk about how security in the cloud is slightly different than security in your on premises data centers. When you move computer systems and data to the cloud, security responsibilities become shared between you and your cloud service provider.

In this case, AWS is responsible for securing the underlying infrastructure that supports the cloud, and you're responsible for anything you put on the cloud or connect to the cloud. This shared security responsibility model can reduce your operational burden in many ways, and in some cases may even improve your default security posture without additional action on your part.



The amount of security configuration work you have to do varies depending on which services you select and how sensitive your data is. However, there are certain security features—such as individual user accounts and credentials, SSL/TLS for data transmissions, and user activity logging—that you should configure no matter which AWS service you use.

AWS Security Responsibilities

Amazon Web Services is responsible for protecting the global infrastructure that runs all of the services offered in the AWS cloud. This infrastructure is comprised of the hardware, software, networking, and facilities that run AWS services.

Protecting this infrastructure is AWS's number one priority, and while you can't visit our data centers or offices to see this protection firsthand, we provide several reports from third-party auditors who have verified our compliance with a variety of computer security standards and regulations. Note that in addition to protecting this global infrastructure, AWS is responsible for the security configuration of its products that are considered managed services. Examples of these types of services include Amazon DynamoDB, Amazon RDS, Amazon Redshift, Amazon Elastic MapReduce, Amazon WorkSpaces, and several other services.

These services provide the scalability and flexibility of cloud-based resources with the additional benefit of being managed. For these services, AWS will handle basic security tasks like guest operating system (OS) and database

patching, firewall configuration, and disaster recovery. For most of these managed services, all you have to do is configure logical access controls for

the resources and protect your account credentials. A few of them may require additional tasks, such as setting up database user accounts, but overall the security configuration work is performed by the service.

Customer Security Responsibilities

With the AWS cloud, you can provision virtual servers, storage, databases, and desktops in minutes instead of weeks. You can also use cloud-based analytics and workflow tools to process your data as you need it, and then store it in your own data centers or in the cloud. Which AWS services you use will determine how much configuration work you have to perform as part of your security responsibilities. AWS products that fall into the well-understood category of Infrastructure as a Service (IaaS)—such as Amazon EC2, Amazon VPC, and Amazon S3 are completely under your control and require you to perform all of the necessary security configuration and management tasks.

For example, for EC2 instances, you're responsible for management of the guest OS (including updates and security patches), any application software or utilities you install on the instances, and the configuration of the AWS-provided firewall (called a security group) on each instance.

These are basically the same security tasks that you're used to performing no matter where your servers are located. AWS managed services like Amazon RDS or Amazon Redshift provide all of the resources you need in order to perform a specific task—but without the configuration work that can come with them.

With managed services, you don't have to worry about launching and maintaining instances, patching the guest OS or database, or replicating databases AWS handles that for you. But as with all services, you should protect your AWS Account credentials and set up individual user accounts with Amazon Identity and Access Management (IAM) so that each of your users has their own credentials and you can implement segregation of duties.

We also recommend using multi-factor authentication (MFA) with each account, requiring the use of SSL/TLS to communicate with your AWS resources, and setting up API/user activity logging with AWS CloudTrail. AWS Global Infrastructure Security AWS operates the global cloud infrastructure

that you use to provision a variety of basic computing resources such as processing and storage. The AWS global infrastructure includes the facilities, network, hardware, and operational software (e.g., host OS, virtualization software, etc.) that support the provisioning and use of these resources. The AWS global infrastructure is designed and managed according to security best practices as well as a variety of security compliance standards. As an AWS customer, you can be assured that you're building web architectures on top of some of the most secure computing infrastructure in the world.

AWS as well as Azure have Compliance Programs to customers understand the robust controls in place to maintain security and data protection in the cloud.

As systems are built on top of any infrastructure, compliance responsibilities will be shared.

By tying together governance-focused, audit friendly service features with applicable compliance or audit standards, AWS Compliance enablers build on traditional programs; helping customers to establish and operate in an AWS security control environment.

The IT infrastructure that is provided to customers is designed and managed in alignment with security best practices and a variety of IT security standards, including:

• SOC 1/SSAE 16/ISAE 3402 (formerly SAS 70) • SOC 2 • SOC 3 • FISMA, DIACAP, and FedRAMP • DOD CSM Levels 1-5 • PCI DSS Level 1 • ISO 9001 / ISO 27001 • ITAR • FIPS 140-2 • MTCS Level 3

Security On the Cloud - Design Principles

Learn about the five best practice areas for security in the cloud: a) Identity and Access Management b) Detective Controls c) Infrastructure Protection d) Data Protection e) Incident Response

The security pillar includes the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies. The security pillar provides an overview of design principles, best practices, and questions

Design Principles

There are six design principles for security in the cloud:

Implement a strong identity foundation:

Implement the principle of least privilege and enforce separation of duties with appropriate authorization for each interaction with your AWS resources. Centralize privilege management and reduce or even eliminate reliance on long term credentials.

Enable traceability:

Monitor, alert, and audit actions and changes to your environment in real time. Integrate logs and metrics with systems to automatically respond and take action.

Apply security at all layers:

Rather than just focusing on protecting a single outer layer, apply a defense-in-depth approach with other security controls. Apply to all layers, for example, edge network, virtual private cloud (VPC), subnet, load balancer, every instance, operating system, and application.

Automate security best practices:

Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost effectively. Create secure architectures, including the implementation of controls that are defined and managed as

code in version-controlled templates.

Protect data in transit and at rest:

Classify your data into sensitivity levels and use mechanisms, such as encryption and tokenization where appropriate. Reduce or eliminate direct human access to data to reduce risk of loss or modification.

Prepare for security events:

Prepare for an incident by having an incident management process that aligns to your organizational requirements. Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery.

Definition There are five best practice areas for security in the cloud:

1. Identity and Access Management
2. Detective Controls
3. Infrastructure Protection
4. Data Protection
5. Incident Response

Before you architect any system, you need to put in place practices that influence security. You will want to control who can do what. In addition, you want to be able to identify security incidents, protect your systems and services, and maintain the confidentiality and integrity of data through data protection.

You should have a well-defined and practiced process for responding to security incidents. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

The AWS Shared Responsibility Model enables organizations that adopt the cloud to achieve their security and compliance goals. Because AWS physically secures the infrastructure that supports our cloud services, as an AWS customer you can focus on using services to accomplish your goals.

Best Practices Identity and Access Management

Identity and access management are key parts of an information security

program, ensuring that only authorized and authenticated users are able to access your resources, and only in a manner that is intended. For example,

you should define principals (users, groups, services, and roles that take action in your account), build out policies aligned with these principals, and implement strong credential management.

These privilege-management elements form the core concepts of authentication and authorization. In AWS, privilege management is primarily supported by the AWS Identity and Access Management (IAM) service, which allows you to control user access to AWS services and resources. You should apply granular policies, which assign permissions to a user, group, role, or resource. You also have the ability to require strong password practices, such as complexity level, avoiding re-use, and using multi-factor authentication (MFA).

You can use federation with your existing directory service. For workloads that require systems to have access to AWS, IAM enables secure access through instance profiles, identity federation, and temporary credentials. The following questions focus on identity and access management considerations for security

SEC 1: How are you protecting access to and use of the AWS account root user credentials?

SEC 2: How are you defining roles and responsibilities of system users to control human access to the AWS Management Console and API?

SEC 3: How are you limiting automated access to AWS resources (for example, applications, scripts, and/or third-party tools or services)?

It is critical to keep root user credentials protected, and to this end AWS recommends attaching MFA to the root user and locking the credentials with the MFA in a physically secured location. IAM allows you to create and manage other non-root user permissions, as well as establish access levels to resources.

Detective Controls:

You can use detective controls to identify a potential security incident. These controls are an essential part of governance frameworks and can be used to support a quality process, a legal or compliance obligation, and for threat

identification and response efforts. There are different types of detective controls.

For example, conducting an inventory of assets and their detailed attributes promotes more effective decision making (and lifecycle controls) to help establish operational baselines. Or you can use internal auditing, an examination of controls related to information systems, to ensure that practices meet policies and requirements and that you have set the correct automated alerting notifications based on defined conditions. These controls are important reactive factors that can help your organization identify and understand the scope of anomalous activity.

In AWS, you can implement detective controls by processing logs, events, and monitoring that allows for auditing, automated analysis, and alarming. CloudTrail logs, AWS API calls, and CloudWatch provide monitoring of metrics with alarming, and AWS Config provides configuration history. Service-level logs are also available, for example, you can use Amazon Simple Storage Service (Amazon S3) to log access requests. Finally, Amazon Glacier provides a vault lock feature to preserve mission-critical data with compliance controls designed to support auditable long-term retention.

The following question focuses on detective controls considerations for security.

SEC 4: How are you capturing and analyzing logs? Log management is important to a well-architected design for reasons ranging from security or forensics to regulatory or legal requirements.

It is critical that you analyze logs and respond to them so that you can identify potential security incidents. AWS provides functionality that makes log management easier to implement by giving you the ability to define a data-retention lifecycle or define where data will be preserved, archived, or eventually deleted. This makes predictable and reliable data handling simpler and more cost effective.

Infrastructure Protection

Infrastructure protection includes control methodologies, such as defense in depth and MFA, which are necessary to meet best practices and industry or

depth and MFA, which are necessary to meet best practices and industry or regulatory obligations. Use of these methodologies is critical for successful

ongoing operations either in the cloud or on-premises. In AWS, you can implement stateful and stateless packet inspection, either by using AWS-native technologies or by using partner products and services available through the AWS Marketplace.

You should use Amazon Virtual Private Cloud (Amazon VPC) to create a private, secured, and scalable environment in which you can define your topology—including gateways, routing tables, and public and private subnets. The following questions focus on infrastructure protection considerations for security.

Infrastructure protection includes control methodologies, such as defense in depth and MFA, which are necessary to meet best practices and industry or regulatory obligations. Use of these methodologies is critical for successful ongoing operations either in the cloud or on-premises. In AWS, you can implement stateful and stateless packet inspection, either by using AWS-native technologies or by using partner products and services available through the AWS Marketplace.

You should use Amazon Virtual Private Cloud (Amazon VPC) to create a private, secured, and scalable environment in which you can define your topology—including gateways, routing tables, and public and private subnets. The following questions focus on infrastructure protection considerations for security.

SEC 5: How are you enforcing network and host-level boundary protection?

SEC 6: How are you leveraging AWS service-level security features?

SEC 7: How are you protecting the integrity of the operating system?

Multiple layers of defense are advisable in any type of environment. In the case of infrastructure protection, many of the concepts and methods are valid across cloud and on-premises models. Enforcing boundary protection, monitoring points of ingress and egress, and comprehensive logging, monitoring, and alerting are all essential to an effective information security plan.

AWS customers are able to tailor, or harden, the configuration of an Amazon

AWS customers are able to tailor, or harden, the configuration of an Amazon Elastic Compute Cloud (Amazon EC2), Amazon EC2 Container Service (Amazon

ECS) container, or AWS Elastic Beanstalk instance, and persist this configuration to an immutable Amazon Machine Image (AMI). Then, whether triggered by Auto Scaling or launched manually, all new virtual servers (instances) launched with this AMI receive the hardened configuration.

Data Protection

Before architecting any system, foundational practices that influence security should be in place. For example, data classification provides a way to categorize organizational data based on levels of sensitivity, and encryption protects data by rendering it unintelligible to unauthorized access. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

In AWS, the following practices facilitate protection of data:

As an AWS customer you maintain full control over your data.

AWS makes it easier for you to encrypt your data and manage keys, including regular key rotation, which can be easily automated by AWS or maintained by you. Detailed logging that contains important content, such as file access and changes, is available. AWS has designed storage systems for exceptional resiliency. For example, Amazon S3 is designed for 11 nines of durability. (For example, if you store 10,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000,000 years.)

Versioning, which can be part of a larger data lifecycle management process, can protect against accidental overwrites, deletes, and similar harm. AWS never initiates the movement of data between Regions. Content placed in a Region will remain in that Region unless you explicitly enable a feature or leverage a service that provides that functionality. The following questions focus on data protection considerations for security.

SEC 8: How are you classifying your data?

SEC 9: How are you encrypting and protecting your data at rest?

SEC 10: How are you managing keys?

SEC 11: How are you encrypting and protecting your data in transit?

AWS provides multiple means for encrypting data at rest and in transit. AWS build features into our services that make it easier to encrypt your data. For example, AWS has implemented server-side encryption (SSE) for Amazon S3 to make it easier for you to store your data in an encrypted form. You can also arrange for the entire HTTPS encryption and decryption process (generally known as SSL termination) to be handled by Elastic Load Balancing (ELB).

Incident Response

Even with extremely mature preventive and detective controls, your organization should still put processes in place to respond to and mitigate the potential impact of security incidents. The architecture of your workload will strongly affect the ability of your teams to operate effectively during an incident to isolate or contain systems and to restore operations to a known-good state.

Putting in place the tools and access ahead of a security incident, then routinely practicing incident response, will make sure the architecture is updated to accommodate timely investigation and recovery. In AWS, the following practices facilitate effective incident response:

Detailed logging is available that contains important content, such as file access and changes. Events can be automatically processed and trigger scripts that automate runbooks through the use of AWS APIs.

You can pre-provision tooling and a “clean room” using AWS CloudFormation. This allows you to carry out forensics in a safe, isolated environment. The following question focuses on incident response considerations for security

SEC 12: How do you ensure that you have the appropriate incident response?

Ensure that you have a way to quickly grant access for your InfoSec team, and automate the isolation of instances as well at the capturing of data and state for forensics.

Key AWS Services

The AWS service that is essential to security is IAM, which allows you to securely control access to AWS services and resources for your users. The following services and features support the five areas in security:

following services and features support the five areas in security:

Identity and Access Management:

IAM enables you to securely control access to AWS services and resources. MFA adds an extra layer of protection on top of your user name and password.

Detective Controls:

AWS CloudTrail records AWS API calls, AWS Config provides a detailed inventory of your AWS resources and configuration, and Amazon CloudWatch is a monitoring service for AWS resources.

Infrastructure Protection:

Amazon VPC lets you provision a private, isolated section of the AWS Cloud where you can launch AWS resources in a virtual network.

Data Protection:

Services such as ELB, Amazon Elastic Block Store (Amazon EBS), Amazon S3, and Amazon Relational Database Service (Amazon RDS) include encryption capabilities to protect your data in transit and at rest. Amazon Macie automatically discovers, classifies, and protects sensitive data, while AWS Key Management Service (AWS KMS) makes it easy for you to create and control keys used for encryption.

Incident Response: IAM should be used to grant appropriate authorization to incident response teams. AWS CloudFormation can be used to create a trusted environment for conducting investigations.

Basics of Securing Your Cloud

Security on the cloud consists of - a) Infrastructure Structure Security b) DDoS Mitigation c) Data Encryption d) Inventory & Configuration e) Monitoring and Logging

Security on the cloud consists of -

1. Infrastructure Structure Security
2. DDoS Mitigation
3. Data Encryption
4. Inventory & Configuration
5. Monitoring and Logging

Infrastructure Structure Security

Several security capabilities and services to increase privacy and control network access. These include: Network firewalls built into the VPC and web application firewall capabilities using WAF will let you create private networks, and control access to your instances and applications Encryption in transit with TLS across all services Connectivity options that enable private, or dedicated, connections from your office or on-premises environment

DDoS Mitigation

A distributed denial-of-service (DDoS) attack occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers. Such an attack is often the result of multiple compromised systems (for example, a botnet) flooding the targeted system with traffic. Availability is of paramount importance in the cloud. Customers benefit from DDoS protection services and technologies built from the ground up to provide resilience in the face of DDoS attacks. Some services like auto scaling also are designed with an automatic response to DDoS help minimize time to mitigate and reduce impact.

Data Encryption

AWS offers you the ability to add an additional layer of security to your data

at rest in the cloud, providing scalable and efficient encryption features. This includes:

Data encryption capabilities available in AWS storage and database services, such as EBS, S3, Glacier, Oracle RDS, SQL Server RDS, and Redshift.

Flexible key management options, including AWS Key Management Service, allowing you to choose whether to have AWS manage the encryption keys or enable you to keep complete control over your keys.

Encrypted message queues for the transmission of sensitive data using server-side encryption (SSE) for Amazon SQS.

Dedicated, hardware-based cryptographic key storage using AWS CloudHSM, allowing you to satisfy compliance requirements.

Inventory & Configuration

The Cloud range of tools allow you to move fast while still ensuring that your cloud resources comply with organizational standards and best practices. This includes:

A security assessment service like Amazon Inspector, that automatically assesses applications for vulnerabilities or deviations from best practices, including impacted networks, OS, and attached storage.

Deployment tools to manage the creation and decommissioning of AWS resources according to organization standards.

Inventory and configuration management tools, including AWS Config, that identify AWS resources and then track and manage changes to those resources over time.

Template definition and management tools, including CloudFormation to create standard, preconfigured environments.

Monitoring and Logging

The cloud provides tools and features that enable you to see exactly what's happening in your environment. This includes:

Deep visibility into API calls , including who, what, who, and from where calls were made.

Log aggregation options, streamlining investigations and compliance reporting.

Alert notifications through CloudWatch when specific events occur or thresholds are exceeded.

These tools and features give you the visibility you need to spot issues before they impact the business and allow you to improve security posture, and reduce the risk profile, of your environment.

Identity and Access Control

Cloud Providers offer you capabilities to define, enforce, and manage user access policies across services. This includes services like: AWS Identity and Access Management (IAM) lets you define individual user accounts with permissions across AWS resources.

AWS Multi-Factor Authentication for privileged accounts, including options for hardware-based authenticators AWS Directory Service allows you to integrate and federate with corporate directories to reduce administrative overhead and improve end-user experience.

AWS provides native identity and access management integration across many of its services plus API integration with any of your own applications or services.

Most cloud providers like Azure and AWS are ISO1, ISO2, ISO3, PCI, HIPPA, SOC, FedRAMP compliant.

The Well Architected Framework

The Well Architected Cloud Framework is based on five pillars - operational excellence, security, reliability, performance efficiency, and cost optimization.

The Well Architected Framework helps you understand the pros and cons of decisions you make while building systems on AWS or Azure. As with most of this course we will be using AWS as the service provider.

By using the Framework you will learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. It provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. We believe that having well-architected systems greatly increases the likelihood of business success.

As Solutions Architects my team and I have years of experience architecting solutions across a wide variety of business verticals and use cases. We have helped design and architect 70+ of customers' architectures on AWS, Azure and a few on Oracle. From this experience, we have identified best practices and core strategies for architecting systems in the cloud.

The AWS Well-Architected Framework documents a set of foundational questions that allow you to understand if a specific architecture aligns well with cloud best practices.

The framework provides a consistent approach to evaluating systems against the qualities you expect from modern cloud-based systems, and the remediation that would be required to achieve those qualities.

As AWS continues to evolve, and we continue to learn more from working with our customers, we will continue to refine the definition of well-architected. This section of the course is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, TPMs, SDMs and operations team members.

Every day we assist customers in architecting systems to take advantage of best practices in the cloud. We work with you on making architectural trade-offs as your designs evolve. As you deploy these systems into live environments, we learn how well these systems perform and the consequences of those trade-offs.

The Well Architected Framework, provides a consistent set of best practices for customers and partners to evaluate architectures, and provides a set of questions you can use to evaluate how well an architecture is aligned to AWS best practices. The Well Architected Framework is based on five pillars operational excellence, security, reliability, performance efficiency, and cost optimization.

Pillar Name	Description
Operational Excellence	The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.
Security	The ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.
Reliability	The ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.
Performance Efficiency	The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
Cost Optimization	The ability to avoid or eliminate unneeded cost or suboptimal resources.

When architecting solutions you make trade-offs between pillars based upon your business context. These business decisions can drive your engineering priorities. You might optimize to reduce cost at the expense of reliability in development environments, or, for mission-critical solutions, you might optimize reliability with increased costs. In ecommerce solutions, performance can affect revenue and customer propensity to buy. Security and operational excellence are generally not traded-off against the other pillars.

On Architecture

In on-premises environments customers often have a central team for technology architecture that acts as an overlay to other product or feature

teams to ensure they are following best practice. Technology architecture

teams are often composed of a set of roles such as Technical Architect (infrastructure), Solutions Architect (software), Data Architect, Networking Architect, and Security Architect. Often these teams use TOGAF or the Zachman Framework as part of an enterprise architecture capability. It is preferable to distribute capabilities into teams rather than having a centralized team with that capability.

There are risks when you choose to distribute decision making authority, for example, ensuring that teams are meeting internal standards. We mitigate these risks in two ways.

First, it is encouraged to have practices that focus on enabling each team to have that capability, to access to experts who ensure that teams raise the bar on the standards they need to meet.

Second, is to put in place mechanisms that carry out automated checks to ensure standards are being meet.

Customer obsessed teams build products in response to a customer need. For architecture this means that we expect every team to have the capability to create architectures and to follow best practices.

General Design Principles

The Well Architected Framework identifies a set of general design principles to facilitate good design in the cloud:

Stop guessing your capacity needs:

Eliminate guessing about your infrastructure capacity needs. When you make a capacity decision before you deploy a system, you might end up sitting on expensive idle resources or dealing with the performance implications of limited capacity. With cloud computing, these problems can go away. You can use as much or as little capacity as you need, and scale up and down automatically.

Test systems at production scale:

In the cloud, you can create a production-scale test environment on demand, complete your testing, and then decommission the resources. Because you

only pay for the test environment when it's running, you can simulate your live environment for a fraction of the cost of testing on premises.

Automate to make architectural experimentation easier:

Automation allows you to create and replicate your systems at low cost and avoid the expense of manual effort. You can track changes to your automation, audit the impact, and revert to previous parameters when necessary

Allow for evolutionary architectures:

Allow for evolutionary architectures. In a traditional environment, architectural decisions are often implemented as static, one-time events, with a few major versions of a system during its lifetime.

As a business and its context continue to change, these initial decisions might hinder the system's ability to deliver changing business requirements. In the cloud, the capability to automate and test on demand lowers the risk of impact from design changes. This allows systems to evolve over time so that businesses can take advantage of innovations as a standard practice.

Drive architectures using data:

In the cloud you can collect data on how your architectural choices affect the behavior of your workload. This lets you make fact-based decisions on how to improve your workload. Your cloud infrastructure is code, so you can use that data to inform your architecture choices and improvements over time.

Improve through game days:

Test how your architecture and processes perform by regularly scheduling game days to simulate events in production. This will help you understand where improvements can be made.

The Five Pillars of the Well Architected Framework Creating a software system is a lot like constructing a building. If the foundation is not solid structural problems can undermine the integrity and function of the building. When architecting technology solutions, if you neglect the five pillars of operational excellence, security, reliability, performance efficiency, and cost optimization it can become challenging to build a system that delivers on your expectations and requirements.

Incorporating these pillars into your architecture will help you produce stable

and efficient systems. This will allow you to focus on the other aspects of

design, such as functional requirements.

Best Practices for the Cloud

Design For Failure & Nothing Will Fail, Decouple your components, Implementing elasticity, Automating Your Infrastructure, Thinking parallel.

In this section, you will learn about designing the best practices that will help you build an application in the cloud.

Design For Failure & Nothing Will Fail

Rule of thumb: Be a pessimist when designing architectures in the cloud; assume things will fail. In other words, always design, implement and deploy for automated recovery from failure.

In particular, assume that your hardware will fail. Assume that outages will occur. Assume that some disaster will strike your application. Assume that you will be slammed with more than the expected number of requests per second some day.

If you realize that things will fail over time and incorporate that thinking into your architecture, build mechanisms to handle that failure before disaster strikes to deal with a scalable infrastructure, you will end up creating a fault-tolerant architecture that is optimized for the cloud.

Questions that you need to ask yourself:

What happens if a node in your system fails? How do you recognize that failure? How do I replace that node? What kind of scenarios do I have to plan for? What are my single points of failure? If a load balancer is sitting in front of an array of application servers, what if that load balancer fails? If there are master and slaves in your architecture, what if the master node fails? How does the failover occur and how is a new slave instantiated and brought into sync with the master? Just like designing for hardware failure, you have to also design for software failure.

Questions that you need to ask:

What happens to my application if the dependent services changes its interface? What if downstream service times out or returns an exception? What if the cache keys grow beyond memory limit of an instance? Build mechanisms to handle that failure. For example, the following strategies can help in event of failure:

1. Have a coherent backup and restore strategy for your data and automate it.
2. Build process threads that resume on reboot.
3. Allow the state of the system to re-sync by reloading messages from queues.
4. Keep pre-configured and pre-optimized virtual images to support on launch/boot.
5. Avoid in-memory sessions or stateful user context, move that to data stores. Good cloud architectures should be impervious to reboots and re-launches. You can do this using a combination of Amazon SQS and Amazon SimpleDB, the overall controller architecture is very resilient to the types of failures listed in this section.

For instance, if the instance on which controller thread was running dies, it can be brought up and resume the previous state as if nothing had happened. This was accomplished by creating a pre-configured Amazon Machine Image, which when launched dequeues all the messages from the Amazon SQS queue and reads their states from an Amazon SimpleDB domain on reboot.

Designing with an assumption that underlying hardware will fail, will prepare you for the future when it actually fails. This design principle will help you design operations-friendly applications.

If you can extend this principle to proactively measure and balance load dynamically, you might be able to deal with variance in network and disk performance that exists due to multi-tenant nature of the cloud.

Tactics for implementing the above best practice:

1. Failover Gracefully Using Elastic IPs: Elastic IP is a static IP that is dynamically re-mappable. You can quickly remap and failover to another

set of servers so that your traffic is routed to the new servers. It works great when you want to upgrade from old to new versions or in case of hardware failures.

2. Utilize Multiple Availability Zones: Availability Zones / Availability Domains are conceptually like logical data centers. By deploying your architecture to multiple availability zones, you can ensure highly availability. Utilize Amazon RDS Multi-AZ deployment functionality to automatically replicate database updates across multiple Availability Zones.
3. Maintain a Machine Image so that you can restore and clone environments very easily in a different Availability Zone; Maintain multiple Database slaves across Availability Zones and setup hot replication.
4. Utilize CloudWatch to get more visibility and take appropriate actions in case of hardware failure or performance degradation. Setup an Auto scaling group to maintain a fixed fleet size so that it replaces unhealthy EC2 instances by new ones.
5. Utilize EBS and set up cron jobs so that incremental snapshots are automatically uploaded to Amazon S3 and data is persisted independent of your instances.
6. Utilize RDS and set the retention period for backups, so that it can perform automated backups.

Decouple your components

The cloud reinforces the SOA design principle that the more loosely coupled the components of the system, the bigger and better it scales. The key is to build components that do not have tight dependencies on each other, so that if one component were to fail, not respond or slow to respond for some reason, the other components in the system are built to continue to work as if no failure is happening. In essence, loose coupling isolates the various layers and components of your application so that each component interacts asynchronously with the others and treats them as a “black box”.

For example, in the case of web application architecture, you can isolate the

app server from the web server and from the database. The app server does

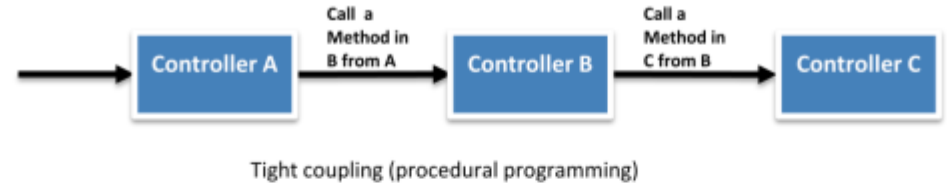
not know about your web server and vice versa, this gives decoupling between these layers and there are no dependencies code-wise or functional perspectives. In the case of batch processing architecture, you can create asynchronous components that are independent of each other.

Questions you need to ask:

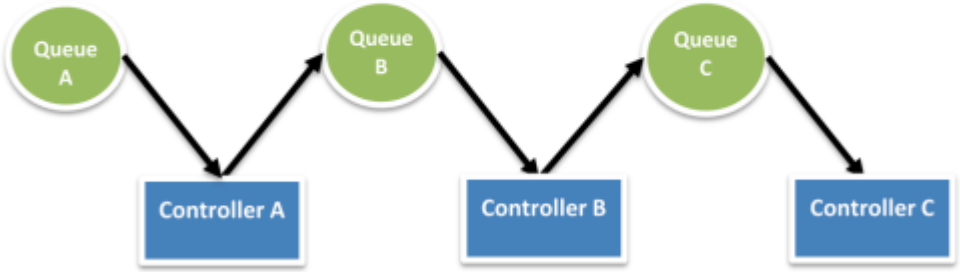
Which business component or feature could be isolated from current monolithic application and can run standalone separately? And then how can I add more instances of that component without breaking my current system and at the same time serve more users? How much effort will it take to encapsulate the component so that it can interact with other components asynchronously? Decoupling your components, building asynchronous systems and scaling horizontally become very important in the context of the cloud.

It will not only allow you to scale out by adding more instances of same component but also allow you to design innovative hybrid models in which a few components continue to run in on-premise while other components can take advantage of the cloud-scale and use the cloud for additional compute-power and bandwidth. That way with minimal effort, you can “overflow” excess traffic to the cloud by implementing smart load balancing tactics.

One can build a loosely coupled system using messaging queues. If a queue/buffer is used to connect any two components together, it can support concurrency, high availability and load spikes. As a result, the overall system continues to perform even if parts of components are momentarily unavailable. If one component dies or becomes temporarily unavailable, the system will buffer the messages and get them processed when the component comes back up.



Tight coupling (procedural programming)



Loose coupling (independent phases using queues)

You will see heavy use of queues in in the Cloud Architectures in general. If lots of requests suddenly reach the server (an Internet-induced overload situation) or the processing of regular expressions takes a longer time than the median (slow response rate of a component), the Amazon SQS queues buffer the requests in a durable fashion so that those delays do not affect other components. AWS specific tactics for implementing this best practice:

1. Use SQS to isolate components
2. Use SQS as buffers between components
3. Design every component such that it expose a service interface and is responsible for its own scalability in all appropriate dimensions and interacts with other components asynchronously
4. Bundle the logical construct of a component into a Machine Image so that it can be deployed more often
5. Make your applications as stateless as possible. Store session state outside of component.

Implement elasticity

The cloud brings a new concept of elasticity in your applications. Elasticity can be implemented in three ways:

1. Proactive Cyclic Scaling: Periodic scaling that occurs at fixed interval (daily, weekly, monthly, quarterly)
2. Proactive Event-based Scaling: Scaling just when you are expecting a big

surge of traffic requests due to a scheduled business event (new product launch, marketing campaigns)

3. Auto-scaling based on demand. By using a monitoring service, your system can send triggers to take appropriate actions so that it scales up or down based on metrics (utilization of the servers or network i/o, for instance) To implement “Elasticity”, one has to first automate the deployment process and streamline the configuration and build process. This will ensure that the system can scale without any human intervention.

This will result in immediate cost benefits as the overall utilization is increased by ensuring your resources are closely aligned with demand rather than potentially running servers that are under-utilized. Automate your infrastructure One of the most important benefits of using a cloud environment is the ability to use the cloud’s APIs to automate your deployment process. It is recommended that you take the time to create an automated deployment process early on during the migration process and not wait till the end. Creating an automated and repeatable deployment process will help reduce errors and facilitate an efficient and scalable update process.

To automate the deployment process:

1. Create a library of “recipes” – small frequently-used scripts (for installation and configuration)
   Manage the configuration and deployment process using agents bundled inside an AMI
2. Bootstrap your instances
3. Bootstrap your instances Let your instances ask you a question at boot “who am I and what is my role?” Every Instance should have a role (“DB server”, “app server”, “slave server” in the case of a Web application) to play in the environment.

This role may be passed in as an argument during launch that instructs the AMI when instantiated the steps to take after it has booted. On boot, instances should grab the necessary resources (code, scripts, configuration) based on the role and “attach” itself to a cluster to serve its function.

Benefits of bootstrapping your instances:

1. Recreate the (Dev, staging, Production) environment with few clicks and minimal effort

minimal effort

2. More control over your abstract cloud-based resources
3. Reduce human-induced deployment errors
4. Create a Self Healing and Self-discoverable environment which is more resilient to hardware failure

AWS Specific Tactics To Automate Your Infrastructure

1. Define Auto-scaling groups for different clusters using the Amazon Auto-scaling feature in EC2.
2. Monitor your system metrics (CPU, Memory, Disk I/O, Network I/O) using Amazon CloudWatch and take appropriate actions (launching new AMIs dynamically using the Auto-scaling service) or send notifications.
3. Store and retrieve machine configuration information dynamically: Utilize Amazon SimpleDB to fetch config data during boot-time of an instance (eg. database connection strings). SimpleDB may also be used to store information about an instance such as its IP address, machine name and role.
4. Design a build process such that it dumps the latest builds to a bucket in Amazon S3; download the latest version of an application from during system startup.
5. Invest in building resource management tools (Automated scripts, pre-configured images) or Use smart open source configuration management tools like Chef, Puppet, CFEngine or Genome.
6. Bundle Just Enough Operating System (JeOS) and your software dependencies into an Amazon Machine Image so that it is easier to manage and maintain. Pass configuration files or parameters at launch time and retrieve user data and instance metadata after launch.
7. Reduce bundling and launch time by booting from Amazon EBS volumes and attaching multiple Amazon EBS volumes to an instance. Create snapshots of common volumes and share snapshots among accounts wherever appropriate.
8. Application components should not assume health or location of hardware it is running on. For example, dynamically attach the IP address of a new node to the cluster. Automatically failover and start a new clone in case of a failure.

Think parallel

The cloud makes parallelization effortless. Whether it is requesting data from the cloud, storing data to the cloud, processing data (or executing jobs) in the cloud, you need to internalize the concept of parallelization when designing architectures in the cloud. It is advisable to not only implement parallelization wherever possible but also automate it because the cloud allows you to create a repeatable process every easily.

When it comes to accessing (retrieving and storing) data, the cloud is designed to handle massively parallel operations. In order to achieve maximum performance and throughput, you should leverage request parallelization.

Multi-threading your requests by using multiple concurrent threads will store or fetch the data faster than requesting it sequentially. Hence, wherever possible, the processes of a cloud application should be made thread-safe through a share-nothing philosophy and leverage multi-threading. When it comes to processing or executing requests in the cloud, it becomes even more important to leverage parallelization.

A general best practice, in the case of a web application, is to distribute the incoming requests across multiple asynchronous web servers using load balancer. In the case of batch processing application, you can master node can spawn up multiple slave worker nodes that processes task in parallel (as in distributed processing frameworks like Hadoop)

The beauty of the cloud shines when you combine elasticity and parallelization. Your cloud application can bring up a cluster of compute instances which are provisioned within minutes with just a few API calls, perform a job by executing tasks in parallel, store the results and terminate all the instances.

Tactics for parallelization:

1. Multi-thread your Amazon S3 requests
2. Multi-thread your Amazon SimpleDB GET and BATCHPUT requests
3. Create a JobFlow using the Amazon Elastic MapReduce Service for each of your daily batch processes (indexing, log analysis etc.) which will compute the job in parallel and save time.
4. Use the Elastic Load Balancing service and spread your load across multiple web app servers dynamically Keep dynamic data closer to the compute and static data closer to the end-user.

In general it's a good practice to keep your data as close as possible to your compute cluster or processing elements to reduce latency. In the cloud, this best practice is even more relevant and important because you often have to deal with Internet latencies.

Moreover, in the cloud, you are paying for bandwidth in and out of the cloud by the gigabyte of data transfer and the cost can add up very quickly. If a large quantity of data that needs to be processed resides outside of the cloud, it might be cheaper and faster to "ship" and transfer the data to the cloud first and then perform the computation.

For example, in the case of a data warehousing application, it is advisable to move the dataset to the cloud and then perform parallel queries against the dataset. In the case of web applications that store and retrieve data from relational databases, it is advisable to move the database as well as the app server into the cloud all at once. If the data is generated in the cloud, then the applications that consume the data should also be deployed in the cloud so that they can take advantage of in-cloud free data transfer and lower latencies.

For example, in the case of an ecommerce web application that generates logs and clickstream data, it is advisable to run the log analyzer and reporting engines in the cloud. Conversely, if the data is static and not going to change often (for example, images, video, audio, PDFs, JS, CSS files), it is advisable to take advantage of a content delivery service so that the static data is cached at an edge location closer to the end-user (requester) thereby lowering the access latency. Due to the caching, a content delivery service provides faster access to popular objects.

Tactics for implementing this best practice:

1. Ship your data drives to Amazon using the Import/Export service. It may be cheaper and faster to move large amounts of data using the sneakernet than to upload using the Internet.
2. Utilize the same Availability Zone to launch a cluster of machines.
3. Create a distribution of your Amazon S3 bucket and let Amazon CloudFront caches content in that bucket across all the several edge locations around the world

locations around the world.

Scaling on the cloud

Learn about scaling on the cloud by using a) Redundancy b) Monitoring c) Failover.

Highly Available (HA) Architecture

Introduction

This reference architecture provides the best practices needed for planning, designing, and deploying High Availability (HA) architectures that can be followed for any Cloud Infrastructure Service. A high availability service or application is one designed for maximum potential uptime and accessibility. To design a high availability architecture, three key elements should be considered

1. Redundancy
2. Monitoring
3. Failover

Redundancy

Redundancy means that multiple components can perform the same task. The problem of a single point of failure is eliminated because redundant components can take over a task performed by a component that has failed.

Monitoring

Monitoring checks whether a component is working properly.

Failover

Failover is the process by which a secondary component becomes primary when a primary component failsflev. Although high availability can be achieved at many \ levels, including the application level and the cloud infrastructure level.

High Availability Building Blocks is by region is a localized geographic area composed of several availability domains. An availability domain is one or

more data centers located within a region.

Availability zones / domains are isolated from each other, fault tolerant, and very unlikely to fail simultaneously. Because availability domains do not share physical infrastructure, such as power or cooling, or the internal availability zone network, a failure that impacts one availability zone is unlikely to impact the availability of the others. All the availability zone/domain in a region are connected to each other by a low-latency, high bandwidth network.

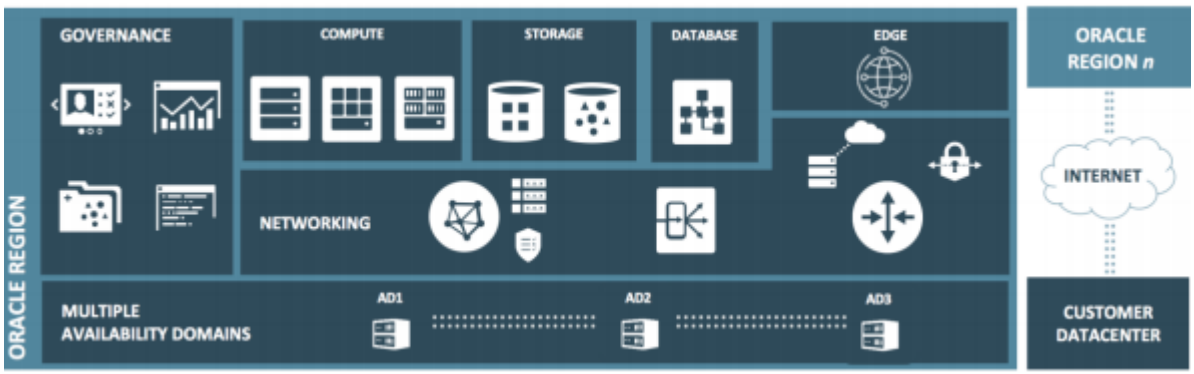
This predictable, encrypted interconnection between availability domains provides the building blocks for both high availability and disaster recovery. Note that cloud Infrastructure resources are either specific to a region, such as a virtual cloud network, or specific to an availability domain, such as a Compute instance.

When you configure your cloud services, if the services are specific to an availability domain, it is important to leverage with multiple availability domains to ensure high availability and to protect against resource failure.

For example, when you deploy a Compute instance, it resides in one particular availability domain. If there is no redundant deployment of this instance to another availability domain, the instance will be impacted when its availability domain encounters any issues.

Architecting High Availability Solutions

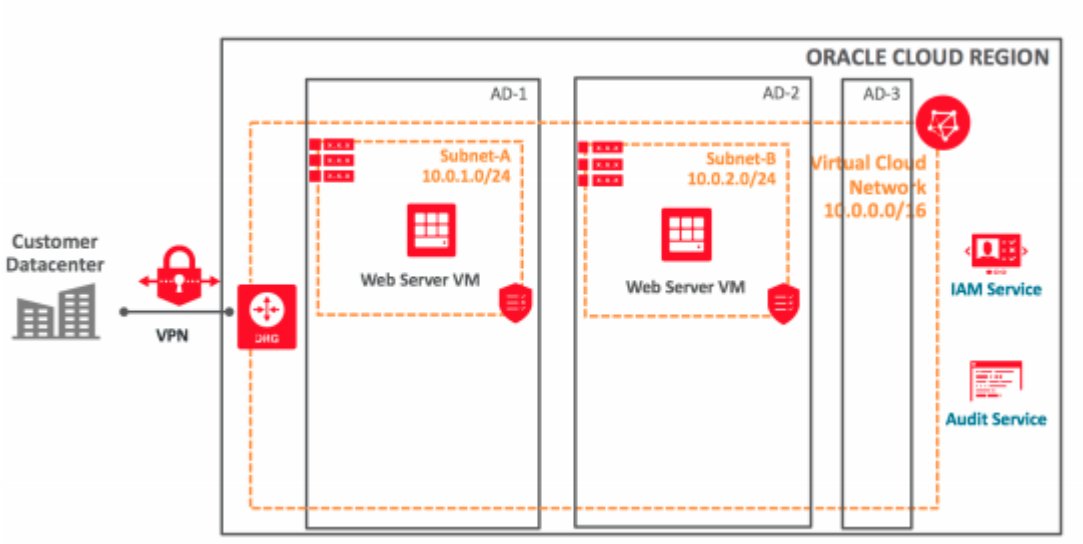
This describes each Cloud Infrastructure layer and provides detailed best practices and design guidelines for architecting high availability solutions.



Compute High Availability Design Cloud Infrastructure

Compute provides a virtual machine (VM) instances to give the flexibility to

deploy any size server that you need. This gives you the performance, flexibility, and control to run your most demanding applications and workloads in the cloud. Elimination of a Single Point of Failure One of the key principles of designing high availability solutions is to avoid single point of failure. It is recommended designing your architecture to deploy Compute instances that perform the same tasks in multiple availability domains. This design removes a single point of failure by introducing redundancy.



Deploy Web Server VMs in Two Availability Domains Depending on your system requirements, you can implement this architecture redundancy in either standby or active mode: In standby mode, a secondary or standby component runs side-by-side with the primary component. When the primary component fails, the standby component takes over. Standby mode is typically used for applications that need to maintain their states.

In active mode, no components are designated as primary or standby; all components are actively participating in performing the same tasks. When one of the components fails, the related tasks are simply distributed to another component. Active mode is typically used for stateless applications.

Network High Availability Design

One of the first steps in working with any Cloud Infrastructure service is to set up a Virtual Cloud Network (VCN) for your cloud resources. A VCN is a software-defined network that you set up in the Cloud data center. A subnet is a subdivision of a cloud network. Ensuring the high availability of your network is one of the most important items in your architecture design.

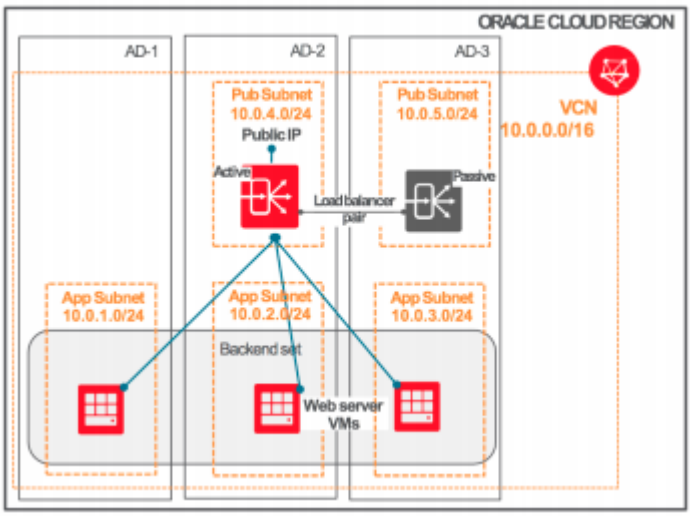
Load Balancing High Availability Design

Cloud Infrastructure services have Load Balancers provides automated traffic distribution from one entry point to multiple servers reachable from your virtual cloud network (VCN). These service offers a load balancer with your choice of a public or private IP address and provisioned bandwidth.

The Load Balancing service improves resource utilization, facilitates scaling, and helps ensure high availability. It supports routing incoming requests to various back-end sets based on virtual hostname, path route rules, or combination of both.

Public Load Balancer accept traffic from the internet. When you create a public load balancer the service assigns it a public IP address that serves as the entry point for incoming traffic. You can associate the public IP address with a friendly DNS name through any DNS vendor. A public load balancer is regional in scope and requires two subnets, each in a separate availability domain. As a result, a public load balancer is inherently highly available across availability domains.

To achieve high availability for your systems, you can put your web server VMs as back-end server sets behind a public load balancer.



Private Load Balancers help isolate your load balancer from the internet and simplify your security posture. The Load Balancing service assigns it a private IP address that serves as the entry point for incoming traffic. When you create a private load balancer, the service requires only one subnet to host both the primary and standby load balancers. In this case, private load balancer

service is bounded within an availability domain.

To provide high availability across availability domains, customers can configure multiple private load balancers on Cloud Infrastructure service and use on-premises or private DNS servers to set up a round-robin DNS configuration with the IP addresses of the private load balancers.

FastConnect and VPN High Availability Design

Highly available, fault-tolerant network connections are key to a well-architected system. This section gives guidelines on how to design your network for redundancy so that it meets the requirements for the Cloud Infrastructure IPSec VPNs and FastConnect / DirectConnect service level agreement (SLA). It discusses high availability options for redundant VPN connections, redundant FastConnect connections, and a FastConnect connection with a backup VPN connection. An organization's business-availability and application requirements help determine the most appropriate configuration when designing remote connections.

Generally, however, you should consider using redundant hardware and network service providers between your location and Service Provider's data centers.

Network High Availability Design with FastConnect or Direct Connect

Cloud Infrastructure FastConnect/DirectConnect provides an easy way to create a dedicated, private connection between your data center and the Cloud Infrastructure. FastConnect provides higher-bandwidth options and a more reliable and consistent networking experience compared to internet-based connections. With FastConnect, you can choose to use private peering, public peering, or both. Use private peering to extend your existing infrastructure into a virtual cloud network (VCN) in the Cloud for example, to implement a hybrid cloud, or in a lift and-shift scenario.

FastConnect / Direct Connect - Redundancy

To avoid a single point of failure with redundancy use -

1. Multiple FastConnect locations within each metro area
2. Multiple routers in each FastConnect location
3. Multiple physical circuits in each FastConnect location

4. Cloud Providers normally handle the redundancy of the routers and physical circuits in the FastConnect locations.

In your network design with FastConnect / DirectConnect, it is recommended considering the following redundancy configurations for your high availability requirements:

Availability domain redundancy:

Connect to any FastConnect location and access services located in any availability domain within a region. This now provides availability domain resiliency via multiple POPs per region. Peering connections terminate on routers in the POP.

Data center location redundancy:

Connect at two different FastConnect locations per region.
Router redundancy: Connect to two different routers per FastConnect location.

Circuit redundancy:

Have multiple physical connections at any of the FastConnect locations. Each of these circuits can have multiple physical links in an aggregated interface/LAG, which adds another level of redundancy.

Partner/provider redundancy:

Connect to the FastConnect locations by using single or multiple partners.

Continuous Testing of Redundant Paths

During normal operation, it is recommended using all available paths between your on-premises network and the Cloud. Doing so ensures that if a failure occurs, your redundant path is already working. Alternatively, using an active/backup design means that you trust that your backup path will work during a failure Storage High Availability Design Cloud Infrastructure provides the following storage services:

1. Block Volume
2. Object Storage
3. File Storage

Cloud Infrastructure Block Volume

Cloud Infrastructure Block Volume enables you to dynamically provision and

manage block storage volumes. You can create, attach, connect, and move volumes as needed to meet your storage and application requirements. When a volume is attached and connected to an instance, you can use it like a regular hard drive. Volumes can also be disconnected and attached to another Compute instance while the data on the volume is maintained.

Cloud Infrastructure Object Storage

Cloud Infrastructure Object Storage are generally built an internet-scale, high-performance storage platforms that offers reliable and cost-efficient data durability. The Object Storage service can store an unlimited amount of unstructured data of any content type, including analytic data and rich content, like images and videos. Object Storage is a regional service and is available across all the availability domains within a region. Data is stored redundantly across multiple storage servers and across multiple availability domains.

Cloud Infrastructure File Storage

Cloud Infrastructure File Storage provides a durable, scalable, distributed, enterprise-grade network file system. You can connect to a File Storage file system from any virtual machine, or container instance in your Virtual Cloud Network (VCN).

You can also access a file system from outside the VCN by using Cloud Infrastructure FastConnect and IPSec VPNs. Large Compute clusters of thousands of instances can use File Storage for high-performance shared storage, and it provides redundant storage for resilient data protection. To achieve high availability and durability, we recommended the following best practices for the storage layer:

Use Object Storage to back-up application data. Data is stored redundantly across multiple storage servers across multiple availability domains. Data integrity is actively monitored by using checksums, and corrupt data is detected and automatically repaired. Any loss in data redundancy is automatically detected and corrected, without any customer impact.

Use Block Volume policy-based backups to perform automatic, scheduled backups and retain them based on a backup policy. Consistently backing up your data allows you to adhere to your data compliance and regulatory requirements. If you need an immediate, point-in-time, direct disk-to-disk

copy of your block volume, use the Block Volume cloning feature. Volume

cloning is different from snapshots because there is no copy-on-write or dependency to the source volume. No backup is involved. The clone operation is immediate, and the cloned volume becomes available for use right after the clone operation is initiated. You can attach and use the cloned volume as a regular volume as soon as its state changes to available.

If you need to safeguard data against accidental or malicious modifications by an untested or untrusted application, use a block volume with a read-only attachment. A read-only attachment marks a volume as read-only, so the data in the volume is not mutable. You can also use read-only attachments when you have multiple Compute instances that access the same volume for read-only purposes.

For example, the instances might be running a web front end that serves static product catalog information to clients.

When your workload requires highly available shared storage with file semantics, and you need built-in encryption and snapshots for data protection, use File Storage. File Storage uses the industry-standard Network File System (NFS) file access protocol and can be accessed concurrently by thousands of Compute instances. File Storage can provide high performance and resilient data protection for your applications. The File Storage service runs locally within one availability domain. Within an availability domain, File Storage uses synchronous replication and high availability failover to keep your data safe and available.

If your application needs high availability across multiple availability domains, use GlusterFS on top of the Block Volume service. Plan and size your storage capacity by considering future growth needs.

Cloud Database High Availability Design

The Cloud Infrastructure Database service enable you launch a Database System (DB System) and create one or more databases on it. The Database service supports several types of DB Systems, ranging in size, price, and performance.

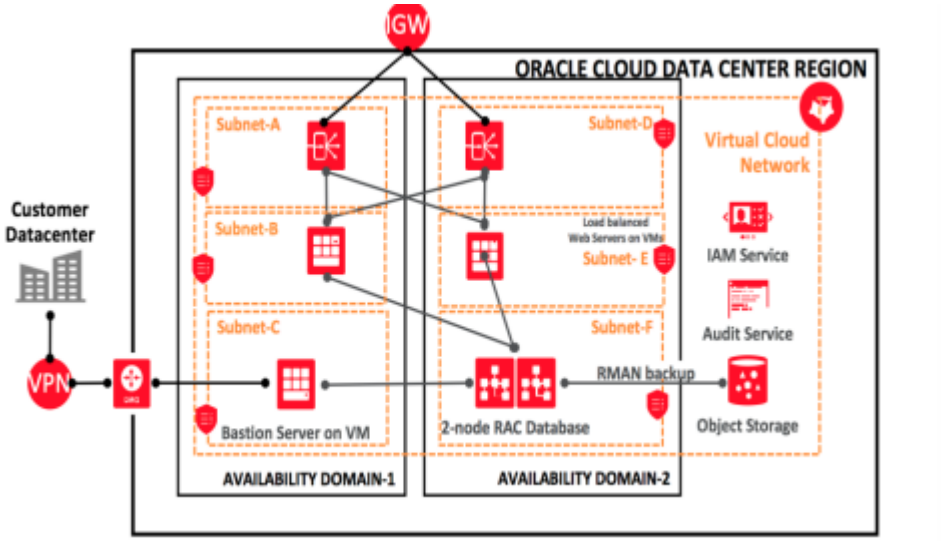
You can configure automatic backups, optimize for different workloads, and

scale up the system to meet increased demands.

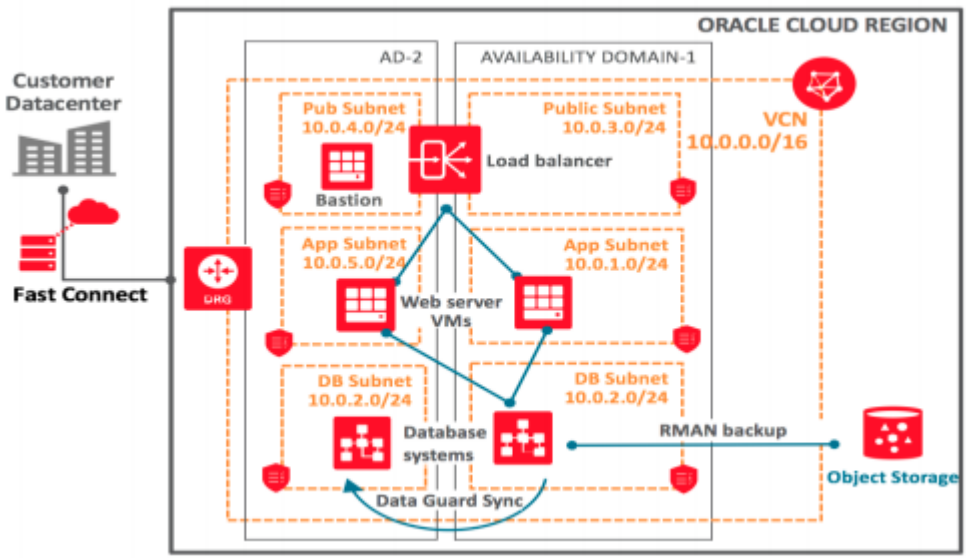
Using 2-node RAC DB Systems - This is exclusively offered by Oracle Cloud

Oracle Infrastructure offers 2-node RAC DB Systems on virtual machine Compute instances. 2-node RAC DB systems provide built-in high availability capabilities, it is recommended using 2-node RAC DB Systems for your solutions that require high availability.

You can configure the Database service to automatically backup to the Object Storage. The following diagram shows the deployment of a 2-node RAC DB System to support the high availability of a two-tier web application:



Working with Data Guard For solutions with a single-node DB system, we recommended using Oracle Data Guard to achieve high availability. Data Guard ensures high availability, data protection, and disaster recovery for enterprise data. Implementation of Data Guard service requires two databases, one in a primary role and one in a standby role. The two databases compose a Data Guard association. Most of your applications access the primary database. The standby database is a transactional consistent copy of the primary database. To improve availability and disaster recovery, it is recommended placing the DB System of the standby database in a different availability domain from the DB System of the primary database.



Data Guard maintains the standby database by transmitting and applying redo data from the primary database. If the primary database becomes unavailable, you can use Data Guard to switch the standby database to the primary role.

You can perform following actions with Data Guard configuration to support high availability:

Switchover:

Reverses the primary and standby database roles. Each database continues to participate in the Data Guard association in its new role. A switchover ensures no data loss. You can use a switchover before you perform planned maintenance on the primary database.

Failover:

Transitions the standby database into the primary role after the existing primary database fails or becomes unreachable. A failover might result in some data loss when you use Maximum Performance protection mode.

Reinstate:

Reinstates a database into the standby role in a Data Guard association. You can use the reinstate command to return a failed database to service after correcting the cause of the failure.

Automated CPU and Storage Scaling

To achieve high availability for your solutions, you must ensure that your DB

Systems have sufficient capacity. Database services on Cloud Infrastructure

can dynamically scale CPU cores or database storage based on the different shapes of your Database service.

For DB Systems based on bare metal Compute instances, we recommend that you start with minimum CPU cores and dynamically increase the number of CPU cores as needed. For DB Systems based on virtual machine Compute instance, you can dynamically increase the storage size.

Conclusion

When planning any deployment, planning for availability is a key concern. This reference architecture provides guidance to help design high availability (HA) solutions on any Cloud Infrastructure Service, include the compute, network, storage, and database layers, and provides several best practices to help guide your planning:

1. Eliminate single points of failure with redundancy
2. Deploy your application or solution components across multiple availability domains
3. Design with future growth in mind and ensure that you have sufficient resource capacity
4. Leverage Database Data Guard and dynamic scaling capabilities
5. Replicate your application data across availability domains
6. Automate problem resolution and failover processes

Scale Your Web Application — One Step at a Time

Scale Your Web Application section consists of - a) Easing your server load, b) Reduce write requests c) Reduce read load by adding more read replicas and a general overview of Building Scalable Cloud Architectures.

Scale Your Web Application — One Step at a Time

Often teams experiencing frustration as they attempt to scale their application particularly around the cost and complexity related to scaling. Customers think scaling, it often means horizontal scaling and microservices, but usually, people aren't sure about how to dive in and effectively scale their sites.

Now let's talk about different scaling options. For instance, if your current workload is in a traditional data center, you can leverage the cloud for your on-premises solution. This way you can scale to achieve greater efficiency with less cost. It's not necessary to set up a whole powerhouse to light a few bulbs. If your workload is already in the cloud, you can use one of the available out-of-the-box options. Designing your API in microservices and adding horizontal scaling might seem like the best choice unless your web application is already running in an on-premises environment and you'll need to quickly scale it because of unexpected large spikes in web traffic.

So how to handle this situation? Take things one step at a time when scaling and you may find horizontal scaling isn't the right choice, after all. For example, assume you have a tech news website where you did an early look review of an upcoming and highly anticipated smartphone launch, which went viral. The review, a blog post on your website, includes both video and pictures. Comments are enabled for the post and readers can also rate it. For example, if your website is hosted on a traditional Linux with a LAMP stack, you may find yourself with immediate scaling problems.

Let's get more details on the current scenario and dig out more:

Where are images and videos stored? How many read/write requests are received per second? Per minute? What is the level of security required? Are

these synchronous or asynchronous requests? We'll also want to consider the following if your website has a transactional load like e-commerce or banking:

How is the website handling sessions? Do you have any compliance requests—like the Payment Card Industry Data Security Standard (PCI DSS compliance)—if your website is using its own payment gateway? How are you recording customer behavior data and fulfilling your analytics needs? What are your loading balancing considerations (scaling, caching, session maintenance, etc.)?

So, if we take this one step at a time:

Step 1: Ease server load

We need to quickly handle spikes in traffic, generated by activity on the blog post, so let's reduce server load by moving image and video to some third party content delivery network (CDN). A CDN solution, which is highly scalable with built-in security to verify origin access identity and handle any DDoS attacks. CloudFront can direct traffic to your on-premises or cloud-hosted server with its 113 Points of Presence (102 Edge Locations and 11 Regional Edge Caches) in 56 cities across 24 countries, which provides efficient caching.

Step 2: Reduce read load by adding more read replicas

MySQL provides a nice mirror replication for databases. Oracle has its own Oracle plug for replication and AWS RDS provide up to five read replicas, which can span across the region and even the Amazon database Amazon Aurora can have 15 read replicas with Amazon Aurora auto-scaling support.

If a workload is highly variable, you should consider Amazon Aurora Serverless database along with amazon Lamda to achieve high efficiency and reduced cost. While most mirror technologies do asynchronous replication, AWS RDS can provide synchronous multi-AZ replication, which is good for disaster recovery but not for scalability. Asynchronous replication to mirror instance means replication data can sometimes be stale if network bandwidth is low, so you need to plan and design your application accordingly.

It is recommended that you always use a read replica for any reporting needs and try to move non-critical GET services to read replica and reduce the load on the master database. In this case, loading comments associated with a blog can be fetched from a read replica as it can handle some delay in case there is an issue with asynchronous reflection.

This can be achieved by introducing queues to process the asynchronous message. Amazon Simple Queue Service (Amazon SQS) is a highly scalable queue, which can handle any kind of work message load. You can process data, like rating and review; or calculate Deal Quality Score (DQS) using batch processing via an SQS queue. If your workload is in AWS, It is recommended using a job observer pattern by setting up Auto Scaling to automatically increase or decrease the number of batch servers, using the number of SQS messages, with Amazon CloudWatch, as the trigger. For on-premises workloads, you can use SQS SDK to create an Amazon SQS queue that holds messages until they're processed by your stack. Or you can use Amazon SNS to fan out your message processing in parallel for different purposes like adding a watermark in an image, generating a thumbnail, etc.

Step 4: Introduce a more robust caching engine

You can use something like Amazon Elasticache for Memcache or Redis to reduce write requests. Memcached and Redis have different use cases so if you can afford to lose and recover your cache from your database, use Memcache. If you are looking for more robust data persistence and complex data structure, use Redis. In AWS, these are managed services, which means AWS takes care of the workload for you and you can also deploy them in your on-premises instances or use a hybrid approach.

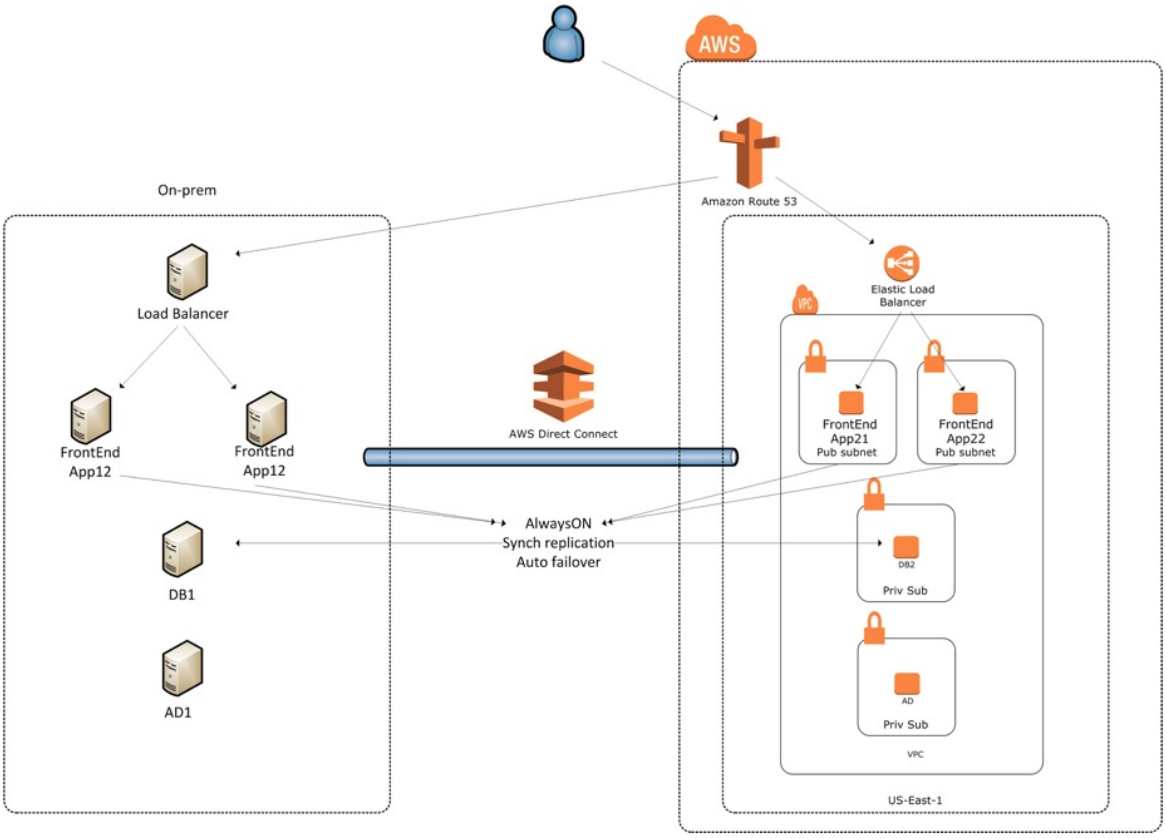
Step 5: Scale your server

If there are still issues, it's time to scale your server. For the greatest cost-effectiveness and unlimited scalability, It is always suggested using horizontal scaling.

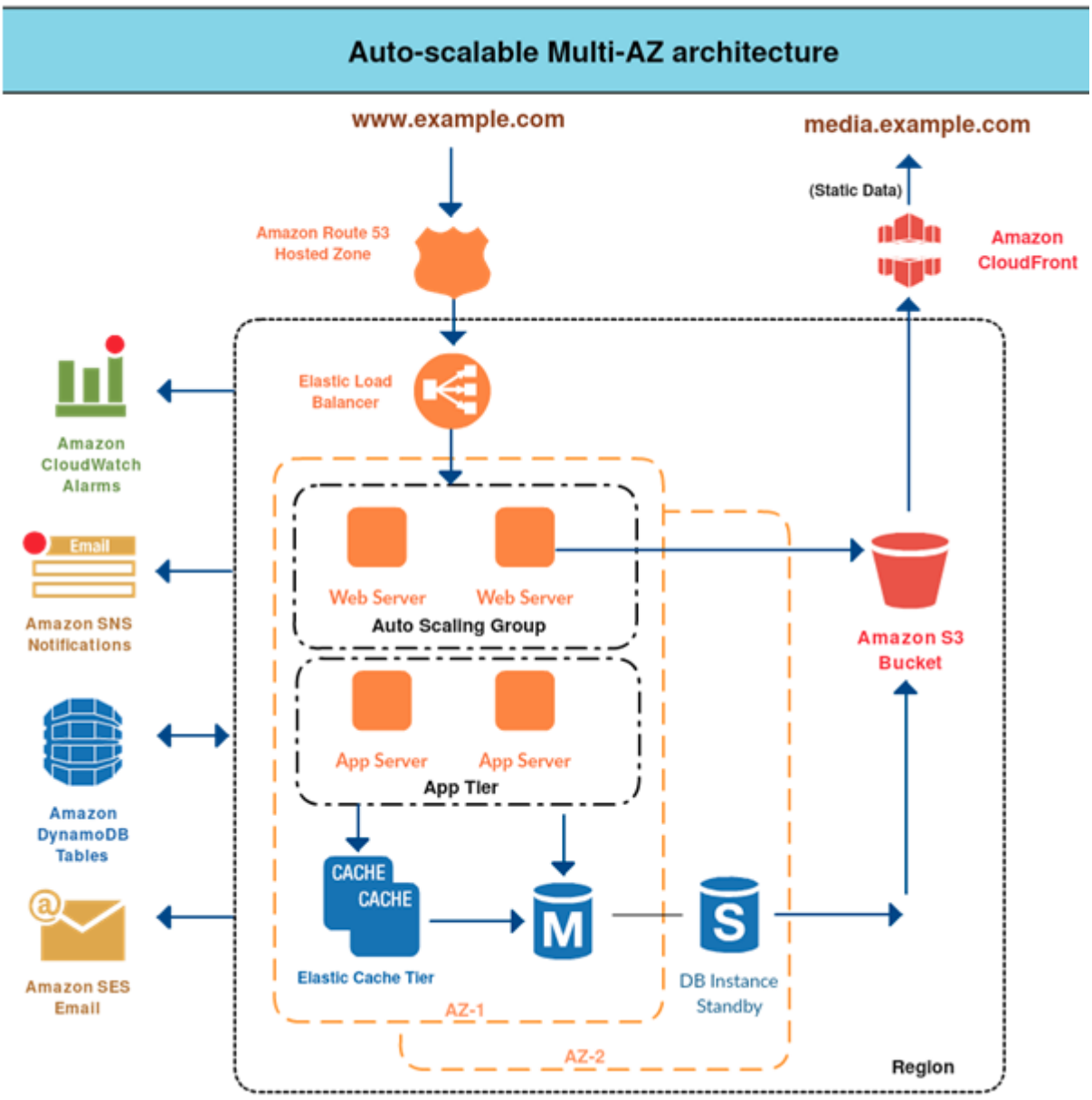
However, use cases like database vertical scaling may be a better choice until you are good with sharding; or use Amazon Aurora Serverless for variable workloads. It will be wise to use Auto Scaling to manage your workload effectively for horizontal scaling. Also, to achieve that, you need to persist the session. Amazon DynamoDB can handle session persistence across instances.

If your server is on premises, consider creating a multisite architecture, which will help you achieve quick scalability as required and provide a good disaster recovery solution. You can pick and choose individual services like Amazon Route 53, AWS CloudFormation, Amazon SQS, Amazon SNS, Amazon RDS, etc.

depending on your needs.



In this architecture, you can run your regular workload on premises, and use your AWS workload as required for scalability and disaster recovery. Using Route 53, you can direct a precise percentage of users to an AWS workload. If you decide to move all of your workloads to AWS, the recommended multi-AZ architecture would look like the following:



In this architecture, you are using a multi-AZ distributed workload for high availability. You can have a multi-region setup and use Route53 to distribute your workload between AWS Regions.

CloudFront helps you to scale and distribute static content via an S3 bucket and DynamoDB, maintaining your application state so that Auto Scaling can apply horizontal scaling without loss of session data. At the database layer, RDS with multi-AZ standby provides high availability and read replica helps achieve scalability.

This is a high-level strategy to help you think through the scalability of your workload by using AWS even if your workload in on premises and not in the cloud yet.

It is highly recommended creating a hybrid, multisite model by placing your

It is highly recommended creating a hybrid, multisite model by placing your on-premises environment replica in the public cloud like AWS Cloud, and

using Amazon Route53 DNS Service and Elastic Load Balancing to route traffic between on-premises and cloud environments. AWS now supports load balancing between AWS and on-premises environments to help you scale your cloud environment quickly, whenever required, and reduce it further by applying Amazon auto-scaling and placing a threshold on your on-premises traffic using Route 53.

Building Scalable Architectures

It is critical to build a scalable architecture in order to take advantage of a scalable infrastructure. The cloud is designed to provide conceptually infinite scalability. However, you cannot leverage all that scalability in infrastructure if your architecture is not scalable. Both have to work together.

You will have to identify the monolithic components and bottlenecks in your architecture, identify the areas where you cannot leverage the on-demand provisioning capabilities in your architecture and work to refactor your application in order to leverage the scalable infrastructure and take advantage of the cloud.

Characteristics of a truly scalable application:

1. Increasing resources results in a proportional increase in performance
2. A scalable service is capable of handling heterogeneity
3. A scalable service is operationally efficient
4. A scalable service is resilient
5. A scalable service should become more cost effective when it grows. Cost per unit reduces as the number of units increases.

These are things that should become an inherent part of your application and if you design your architecture with the above characteristics in mind, then both your architecture and infrastructure will work together to give you the scalability you are looking for.

Reliability on The Cloud

The reliability pillar includes the ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.

The reliability pillar includes the ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.

Design Principles: The five design principles for reliability on the cloud:-

Test recovery procedures:

In an on-premises environment, testing is often conducted to prove the system works in a particular scenario. Testing is not typically used to validate recovery strategies. In the cloud, you can test how your system fails, and you can validate your recovery procedures. You can use automation to simulate different failures or to recreate scenarios that led to failures before. This exposes failure pathways that you can test and rectify before a real failure scenario, reducing the risk of components failing that have not been tested before.

Automatically recover from failure:

By monitoring a system for key performance indicators (KPIs), you can trigger automation when a threshold is breached. This allows for automatic notification and tracking of failures, and for automated recovery processes that work around or repair the failure. With more sophisticated automation, it’s possible to anticipate and remediate failures before they occur.

Scale horizontally to increase aggregate system availability:

Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall system. Distribute requests across multiple, smaller resources to ensure that they don’t share a common point of failure.

Stop guessing capacity:

A common cause of failure in on-premises systems is resource saturation, when the demands placed on a system exceed the capacity of that system (this is often the objective of denial of service attacks). In the cloud, you can monitor demand and system utilization, and automate the addition or removal of resources to maintain the optimal level to satisfy demand without over or under-provisioning.

Manage change in automation:

Changes to your infrastructure should be done using automation. The changes that need to be managed are changes to the automation.

Definition

There are three best practice areas for reliability in the cloud:

1. Foundations
2. Change Management
3. Failure Management

To achieve reliability, a system must have a well-planned foundation and monitoring in place, with mechanisms for handling changes in demand or requirements. The system should be designed to detect failure and automatically heal itself.

Best Practices Foundations

Before architecting any system, foundational requirements that influence reliability should be in place. For example, you must have sufficient network bandwidth to your data center. These requirements are sometimes neglected (because they are beyond a single project's scope).

This neglect can have a significant impact on the ability to deliver a reliable system. In an on-premises environment, these requirements can cause long lead times due to dependencies and therefore must be incorporated during initial planning.

The cloud is designed to be essentially limitless, so it is the responsibility of the cloud provider to satisfy the requirement for sufficient networking and compute capacity, while you are free to change resource size and allocation,

such as the size of storage devices, on demand. The following questions focus on foundations considerations for reliability

REL 1: How are you managing service limits for your accounts?

REL 2: How are you planning your network topology ?

Service limits (an upper limit on the number of each resource your team can request) to protect you from accidentally over-provisioning resources. You will need to have governance and processes in place to monitor and change these limits to meet your business needs. As you adopt the cloud, you may need to plan integration with existing on-premises resources (a hybrid approach). A hybrid model enables the gradual transition to an all-in cloud approach over time. Therefore, it's important to have a design for how your cloud and on-premises resources will interact as a network topology.

Change Management

Being aware of how change affects a system allows you to plan proactively, and monitoring allows you to quickly identify trends that could lead to capacity issues or SLA breaches. In traditional environments, change-control processes are often manual and must be carefully coordinated with auditing to effectively control who makes changes and when they are made. You can monitor the behavior of a system and automate the response to KPIs, for example, by adding additional servers as a system gains more users. You can control who has permission to make system changes and audit the history of these changes. The following questions focus on change management considerations for reliability

REL 3: How does your system adapt to changes in demand?

REL 4: How are you monitoring AWS resources?

REL 5: How are you executing change?

When you architect a system to automatically add and remove resources in response to changes in demand, this not only increases reliability but also ensures that business success doesn't become a burden. With monitoring in place, your team will be automatically alerted when KPIs deviate from expected norms.

Automatic logging of changes to your environment allows you to audit and

quickly identify actions that might have impacted reliability. Controls on

change management ensure that you can enforce the rules that deliver the reliability you need.

Failure Management

In any system of reasonable complexity it is expected that failures will occur. It is generally of interest to know how to become aware of these failures, respond to them, and prevent them from happening again. With all providers, you can take advantage of automation to react to monitoring data. For example, when a particular metric crosses a threshold, you can trigger an automated action to remedy the problem.

Also, rather than trying to diagnose and fix a failed resource that is part of your production environment, you can replace it with a new one and carry out the analysis on the failed resource out of band. Since the cloud enables you to stand up temporary versions of a whole system at low cost, you can use automated testing to verify full recovery processes.

The following questions focus on failure management considerations for reliability

REL 6: How are you backing up your data?

REL 7: How does your system withstand component failures?

REL 8: How are you testing your resiliency?

REL 9: How are you planning for disaster recovery?

Regularly back up your data and test your backup files to ensure you can recover from both logical and physical errors. A key to managing failure is the frequent and automated testing of systems to cause failure, and then observe how they recover. Do this on a regular schedule and ensure that such testing is also triggered after significant system changes. Actively track KPIs, such as the recovery time objective (RTO) and recovery point objective (RPO), to assess a system's resiliency (especially under failure-testing scenarios).

Tracking KPIs will help you identify and mitigate single points of failure. The objective is to thoroughly test your system-recovery processes so that you are confident that you can recover all your data and continue to serve your

confident that you can recover all your data and continue to serve your customers, even in the face of sustained problems. Your recovery processes should be as well exercised as your normal production processes.

Key Services

You would essentially use CloudWatch, which monitors runtime metrics.

The following services and features support the three areas in reliability:

Foundations:

IAM enables you to securely control access to AWS services and resources. Amazon VPC lets you provision a private, isolated section of the AWS Cloud where you can launch AWS resources in a virtual network. AWS Trusted Advisor provides visibility into service limits. AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that safeguards web applications running on AWS.

Change Management:

AWS CloudTrail records AWS API calls for your account and delivers log files to you for auditing. AWS Config provides a detailed inventory of your AWS resources and configuration, and continuously records configuration changes. Auto Scaling is a service that will provide an automated demand management for a deployed workload. CloudWatch provides the ability to alert on metrics, including custom metrics.

Failure Management:

AWS CloudFormation provides templates for the creation of AWS resources and provisions them in an orderly and predictable fashion. Amazon S3 provides a highly durable service to keep backups. Amazon Glacier provides highly durable archives. AWS KMS provides a reliable key management system that integrates with many AWS services.

Understanding Elasticity on the Cloud

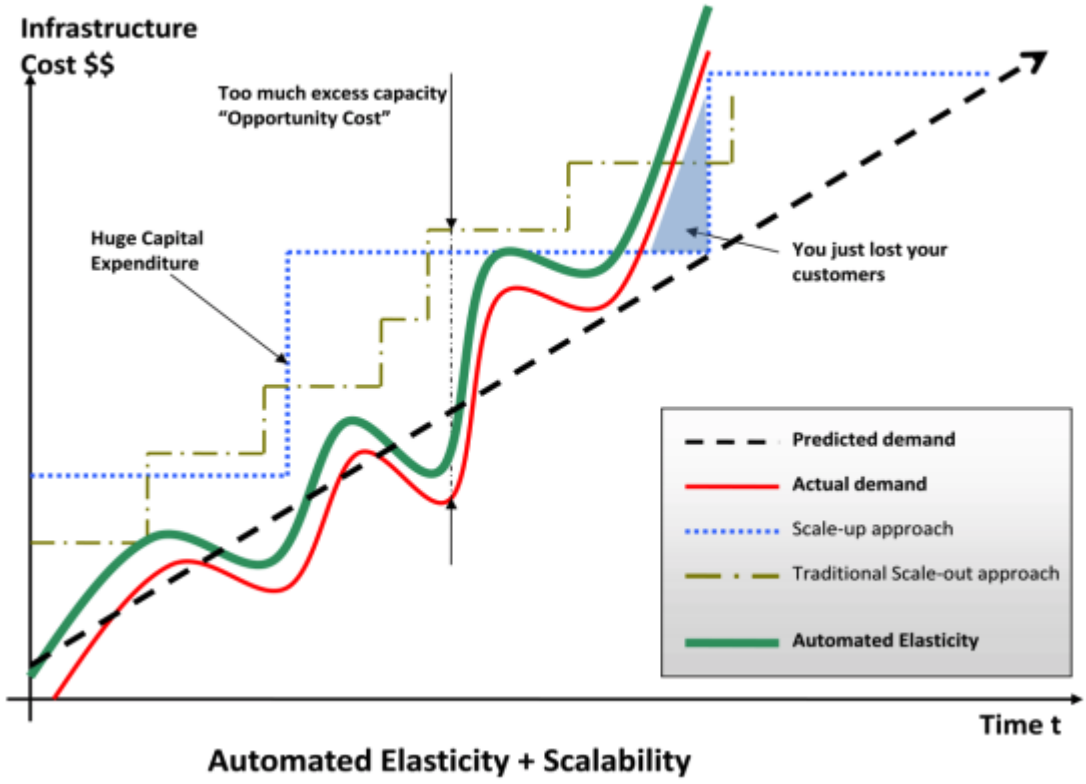
When you decide to move your applications to the cloud and try to map your system specifications to those available in the cloud, you will notice that cloud might not have the exact specification of the resource that you have on-premise.

The graph below illustrates the different approaches a cloud architect can take to scale their applications to meet the demand. Scale-up approach: not worrying about the scalable application architecture and investing heavily in larger and more powerful computers (vertical scaling) to accommodate the demand. This approach usually works to a point, but could either cost a fortune or the demand could outgrow capacity.

The traditional scale-out approach: creating an architecture that scales horizontally and investing in infrastructure in small chunks. Most of the businesses and large-scale web applications follow this pattern by distributing their application components, federating their datasets and employing a service-oriented design.

This approach is often more effective than a scale up approach. However, this still requires predicting the demand at regular intervals and then deploying infrastructure in chunks to meet the demand. This often leads to excess capacity (“burning cash”) and constant manual monitoring (“burning human cycles”). Moreover, it usually does not work if the application is a victim of a viral fire.

Note: Both approaches have initial start-up costs and both approaches are reactive in nature.



Traditional infrastructure generally necessitates predicting the amount of computing resources your application will use over a period of several years. If you underestimate, your applications will not have the horsepower to handle unexpected traffic, potentially resulting in customer dissatisfaction. If you over-estimate, you're wasting money with superfluous resources. The on-demand and elastic nature of the cloud approach (Automated Elasticity), however, enables the infrastructure to be closely aligned (as it expands and contracts) with the actual demand, thereby increasing overall utilization and reducing cost. Elasticity is one of the fundamental properties of the cloud.

Elasticity is the power to scale computing resources up and down easily and with minimal friction. It is important to understand that elasticity will ultimately drive most of the benefits of the cloud. As a cloud architect, you need to internalize this concept and work it into your application architecture in order to take maximum benefit of the cloud.

Traditionally, applications have been built for fixed, rigid and pre-provisioned infrastructure. Companies never had the need to provision and install servers on daily basis. As a result, most software architectures do not address the rapid deployment or reduction of hardware. Since the provisioning time and upfront investment for acquiring new resources was too high, software

architects never invested time and resources in optimizing for hardware utilization. It was acceptable if the hardware on which the application is running was under-utilized. The notion of “elasticity” within an architecture was overlooked because the idea of having new resources in minutes was not possible.

With the cloud, this mindset is changing. Cloud computing streamlines the process of acquiring the necessary resources; there is no longer any need to place orders ahead of time and to hold unused hardware captive. Instead, cloud architects can request what they need mere minutes before they need it or automate the procurement process, taking advantage of the vast scale and rapid response time of the cloud. The same is applicable to releasing the unneeded or under-utilized resources when you don’t need them. Or in with most modern cloud provider now have auto scaling.

If you cannot embrace the change and implement elasticity in your application architecture, you might not be able to take the full advantage of the cloud. As a cloud architect, you should think creatively and think about ways you can implement elasticity in your application. For example, infrastructure that used to run daily nightly builds and perform regression and unit tests every night at 2:00 AM for two hours (often termed as the “QA/Build box”) was sitting idle for rest of the day.

Now, with elastic infrastructure, one can run nightly builds on boxes that are “alive” and being paid for only for 2 hours in the night. Likewise, an internal trouble ticketing web application that always used to run on peak capacity (5 servers 24x7x365) to meet the demand during the day can now be provisioned to run on-demand (5 servers from 9AM to 5 PM and 2 servers for 5 PM to 9 AM) based on the traffic pattern. Designing intelligent elastic cloud architectures, so that infrastructure runs only when you need it, is an art in itself. Elasticity should be one of the architectural design requirements or a system property.

Question that you need to ask:

What components or layers in my application architecture can become elastic? What will it take to make that component elastic? What will be the impact of implementing elasticity to my overall system architecture?

Not Fearing Constraints

When you decide to move your applications to the cloud and try to map your system specifications to those available in the cloud, you will notice that cloud might not have the exact specification of the resource that you have on-premise. For example, “Cloud does not provide X amount of RAM in a server” or “My Database needs to have more IOPS than what I can get in a single instance”. You should understand that cloud provides abstract resources and they become powerful when you combine them with the on-demand provisioning model.

You should not be afraid and constrained when using cloud resources because it is important to understand that even if you might not get an exact replica of your hardware in the cloud environment, you have the ability to get more of those resources in the cloud to compensate that need. For example, if the cloud does not provide you with exact or greater amount of RAM in a server, try using a distributed cache like memcached or partitioning your data across multiple servers.

If your databases need more IOPS and it does not directly map to that of the cloud, there are several recommendations that you can choose from depending on your type of data and use case. If it is a read-heavy application, you can distribute the read load across a fleet of synchronized slaves.

Alternatively, you can use a sharding algorithm that routes the data where it needs to be or you can use various database clustering solutions. In retrospect, when you combine the on-demand provisioning capabilities with the flexibility, you will realize that apparent constraints can actually be broken in ways that will actually improve the scalability and overall performance of the system.

Performance Efficiency on the Cloud

Use computing resources efficiently to meet system requirements and to maintain that efficiency as demand changes and technologies evolve.

Performance Efficiency on the Cloud

The performance efficiency pillar includes the ability to use computing resources efficiently to meet system requirements and to maintain that efficiency as demand changes and technologies evolve. Design Principles There are five design principles for performance efficiency in the cloud:

Democratize advanced technologies:

Technologies that are difficult to implement can become easier to consume by pushing that knowledge and complexity into the cloud vendor's domain. Rather than having your IT team learn how to host and run a new technology, they can simply consume it as a service. For example, NoSQL databases, media transcoding, and machine learning are all technologies that require expertise that is not evenly dispersed across the technical community. In the cloud, these technologies become services that your team can consume while focusing on product development rather than resource provisioning and management.

Go global in minutes:

Easily deploy your system in multiple Regions around the world with just a few clicks. This allows you to provide lower latency and a better experience for your customers at minimal cost.

Use serverless architectures:

In the cloud, serverless architectures remove the need for you to run and maintain servers to carry out traditional compute activities. For example, storage services can act as static websites, removing the need for web servers, and event services can host your code for you. This not only removes the operational burden of managing these servers, but also can lower

transactional costs because these managed services operate at cloud scale.

Experiment more often:
With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations.

Mechanical sympathy:

Use the technology approach that aligns best to what you are trying to achieve. For example, consider data access patterns when selecting database or storage approaches. Take a data-driven approach to selecting a high-performance architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types. By reviewing your choices on a cyclical basis, you will ensure that you are taking advantage of the continually evolving Cloud providers. Monitoring will ensure that you are aware of any deviance from expected performance and can take action on it.

Finally, your architecture can make tradeoffs to improve performance, such as using compression or caching, or relaxing consistency requirements.

Feedback

For feedback, comments, and suggestions, please feel free to reach out to us - feedback@cloudexpertsacademy.com