

Agile Vs Waterfall

1. What is waterfall model?

Waterfall methodology, also known as the linear sequential lifecycle model, is defined by its linear, structured approach to project management. It is made up of a series of steps that are completed in sequential order within the software development life cycle (SDLC).

It is made up of a series of steps that are completed in sequential order within the software development life cycle (SDLC).

Since its publication, variations of waterfall have emerged, but there is general consensus around the following steps within the process:

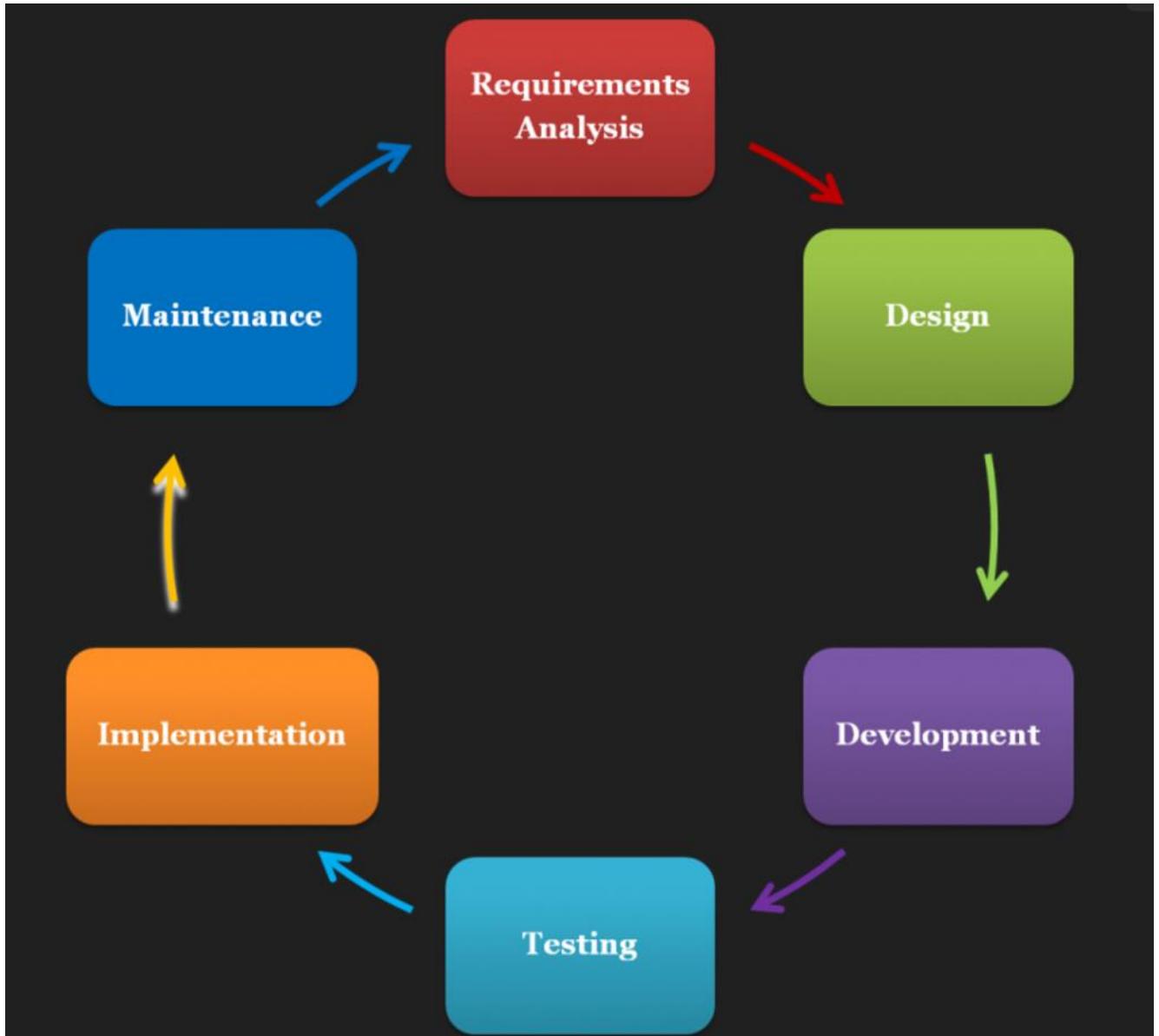
1. **Gathering of requirements:** This stage demands upfront documentation between the development team and the client or end user. During this phase, the product features within the project plan are documented in great detail, enabling the team to determine a clear cost and timeline. After both parties align on the requirements, there is limited to no correspondence between the development team and client until the project is completed.
2. **Design:** The design phase is comprised of two steps: logical design and physical design. In logical design, the team brainstorms possible ways to tackle the client problem. When the development team agrees on a solution, these ideas are translated into specific technical tasks, which are then distributed across the team to construct the physical design.
3. **Implementation:** In next phase, developers start coding based on specifications that were developed in the prior steps.
4. **Verification:** This stage tests ensures that the code functions as intended and that the requirements in the scoping document have been met. The development team checks for bugs in the code and a final validation is conducted by the client to ensure that functionality met expectations.
5. **Maintenance:** As users onboard and use the end product, there will be a need for ongoing support as new issues arise.

Key benefits of the waterfall method

1. Detailed product requirements and documentation enable new programmers to onboard quickly and easily.
2. Documentation provides a clear scope to the project, enabling project managers to communicate budgets, timelines, and key milestones to interested parties.

Key challenges of the waterfall method

1. Clients can find it difficult to outline all of their requirements at the beginning of the project, leading to gaps in documentation.
2. Minimal customer collaboration during the development process can lead to costly changes if the product does not meet expectations.
3. Testers report issues and bugs later in the process, which could have informed an alternative program architecture.



2. What is Agile method?

In contrast to waterfall development, agile is defined by its iterative approach to project management. Instead of drafting lengthy project requirements at the onset, an agile team breaks out the product into specific features, and they tackle each one under a specific time constraint, known as a sprint.

Agile project management requires a cross-functional, self-organizing team that typically consists of five to nine members. Together, they develop a workable piece of software during each sprint, which combines with other functional code from previous iterations. By the end of the sprint timebox, the team demos their work to stakeholders for feedback, allowing them to be flexible in their approach to software development. Since the team has access to frequent feedback, they can adapt the product roadmap during the development lifecycle to ensure that functionality truly meets user expectations. In a waterfall approach, customer involvement typically coincides with the delivery of the final product, which can be costly when requirements are misinterpreted or documented incorrectly.

Agile scrum framework

Inspired by the game of rugby, agile scrum emphasizes teamwork to meet deliverables, similar to the way that forwards need to work together in a scrum to gain possession of a rugby ball. The skillset of the agile scrum team varies, but they usually include the following roles:

Inspired by the game of rugby, agile scrum emphasizes teamwork to meet deliverables, similar to the way that forwards need to work together in a scrum to gain possession of a rugby ball. The skillset of the agile scrum team varies, but they usually include the following roles:

- **Product owner:** This team member represents the needs of both the customer and the business. By crafting user stories, the team can understand how a feature request can help resolve a specific problem, and these stories formulate the backlog of tasks for the team to tackle. This person also prioritizes the stories by their value to the customer, which should, in theory, translate into value for the business. While the product owner leads the team in this way, they do not set deadlines or instruct the team on how work should be delivered.
- **Scrum master:** This team member facilitates the overall agile development process. Similar to a project manager, this person keeps the team on task, ensuring that the team remains focused during the project. They can also act as a neutral party to mediate disagreements among team members. For example, team members may disagree on how much to take on in a given sprint. Product owners, in particular, may pressure teams to commit to more than they can deliver within a given timeframe. In these cases, scrum masters can remind team members the scope of their role on the team.

Other team members of an agile team can vary, but they typically include users from a variety of disciplines, such as design, analytics, QA, and development. These individuals collaborate together to decide on how much work to take on and how they will complete it.

Agile methodologies are also defined by the ways in which the team comes together. There are specific meetings which help facilitate the workflow across the team. Some of them include the following:

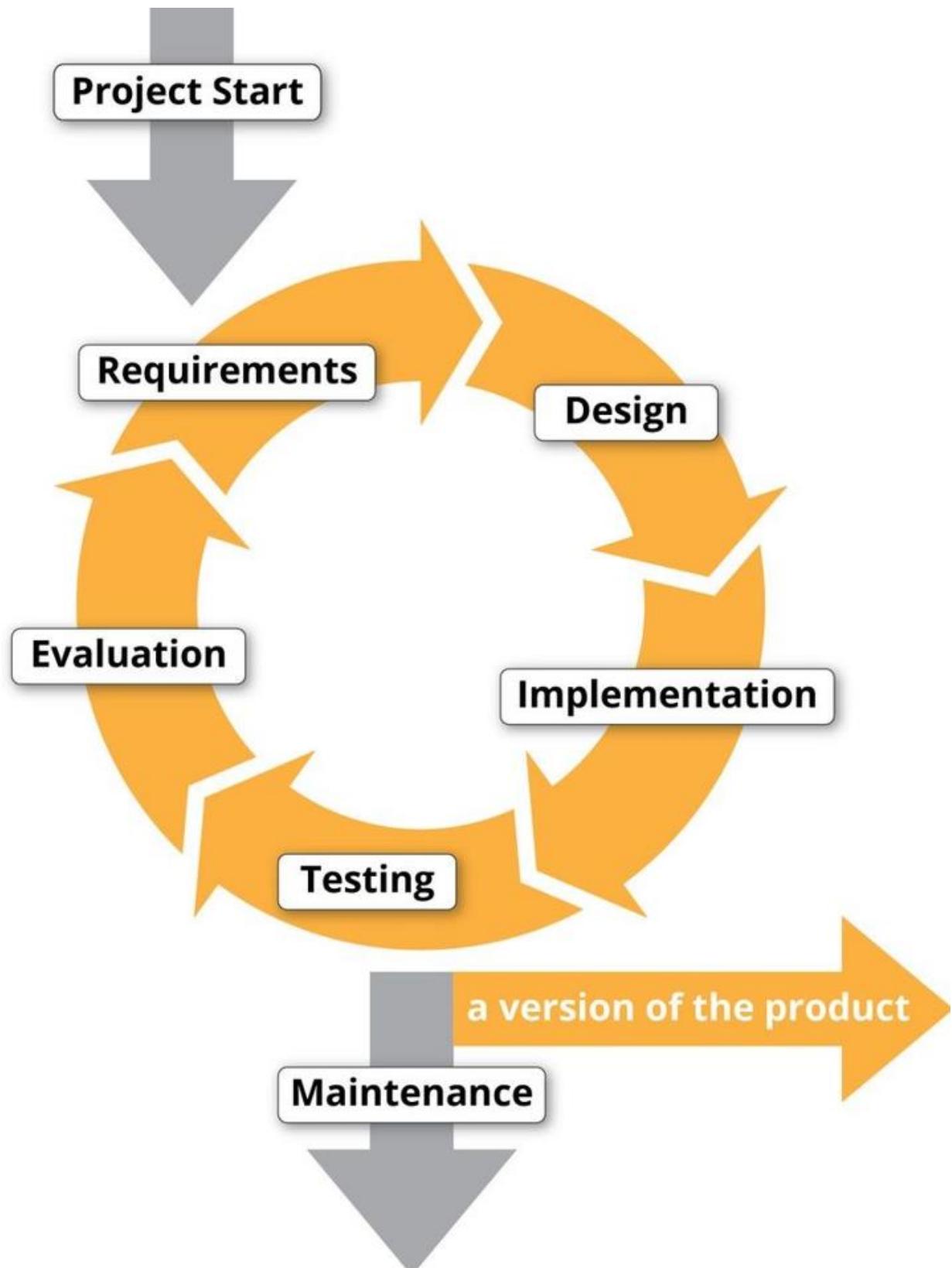
- **Sprint planning:** During this meeting, the team comes together to determine which stories will be part of the current sprint. The product owner will prioritize the user stories, but the rest of the team will need to agree on how many and which user stories they can complete during that set time period.
- **Daily standup:** These brief meetings are also known as daily scrums. During these check-ins, each team member communicates their individual progress, such as completed tasks, upcoming ones, and any blockers or dependencies which may result in delays.
- **Demo:** This meeting showcases the working software that the team completed over the course of the sprint, which can range between two- to four-week increments. The product owner will determine if a user story has met the definition of “done.” If not, the product backlog may be groomed to account for anything missing. This is also an opportunity for the team to present to stakeholders for feedback.
- **Retrospective:** This time is reserved for team introspection, where the team identifies how they could improve upon their workflow to achieve better results in the future.

Key benefits of the agile method-

- Team design facilitates more collaboration.
- Product development takes an adaptive design approach.
- Since code is tested with each iteration in the development phase, code defects can inform future design of the software.
- Tends to yield higher customer satisfaction since frequent feedback leads to increased prioritization of customer needs.
- Enables [continuous integration](#) as each feature is its own workable piece of software.
- This lean type of software development can lead to lower costs as there is less risk of customer and product misalignment.

Key challenges of the agile method

- An agile approach can lack comprehensive documentation. This makes it difficult to onboard new developers, project timelines to stakeholders, and provide accurate cost estimates.
- Can be difficult to scale.



3. Difference between Agile and Waterfall Model:

Below is a difference between Agile and Waterfall methodologies:

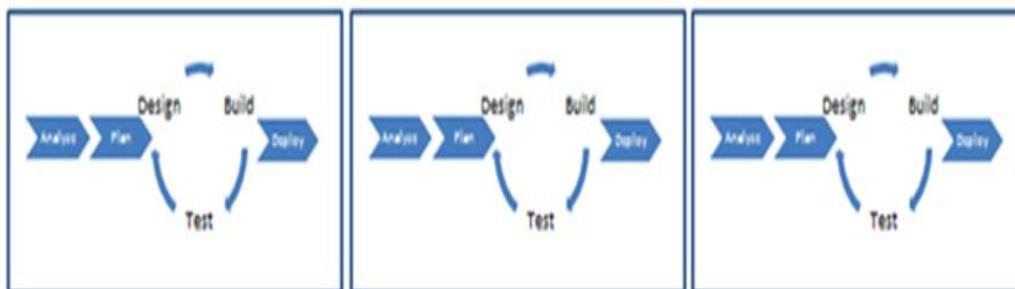
<u>Agile</u>	<u>Waterfall</u>
It separates the project development lifecycle into sprints.	Software development process is divided into distinct phases.
It follows an incremental approach	Waterfall methodology is a sequential design process.
Agile methodology is known for its flexibility.	Waterfall is a structured software development methodology so most times it can be quite rigid.
Agile can be considered as a collection of many different projects.	Software development will be completed as one single project.
Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed.	There is no scope of changing the requirements once the project development starts.
Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once.	All the project development phases like designing, development, testing, etc. are completed once in the Waterfall model.
Test plan is reviewed after each sprint	The test plan is rarely discussed during the test phase.
Agile development is a process in which the requirements are expected to change and evolve.	The method is ideal for projects which have definite requirements and changes not at all expected.
In Agile methodology, testing is performed concurrently with software development.	In this methodology, the “Testing” phase comes after the “Build” phase

Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer's demands.	This model shows a project mindset and places its focus completely on accomplishing the project.
Agile methodology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios.	Reduces risk in the firm fixed price contracts by getting risk agreement at the beginning of the process.
Prefers small but dedicated teams with a high degree of coordination and synchronization.	Team coordination/synchronization is very limited.
Products owner with team prepares requirements just about every day during a project.	Business analysis prepares requirements before the beginning of the project.
Test team can take part in the requirements change without problems.	It is difficult for the test to initiate any change in requirements.
Description of project details can be altered anytime during the SDLC process.	Detail description needs to implement waterfall software development approach.

Waterfall



Agile



4. Supporting Tools

Service Now (Snow)

Ticketing software allows organizations to resolve their internal IT issues by streamlining the resolution process. The elements they handle, called tickets, provide context about the issues, including details, categories, and any relevant tags.

What is a ticket?

Within an employee IT ticketing system, a ticket is a special document or record that represents an incident, alert, request, or event that requires action from the IT department. It often contains additional contextual details and may also include relevant contact information of the individual who created the ticket.

Tickets are usually employee-generated, but automated tickets may also be created when specific incidents occur and are flagged. Once a ticket is created, it is assigned to an IT agent to be resolved. Effective ticketing systems allow tickets to be submitted via a variety of methods. These include submissions through virtual agents, phone, email, service portals, live agents, walk-up experience, etc.

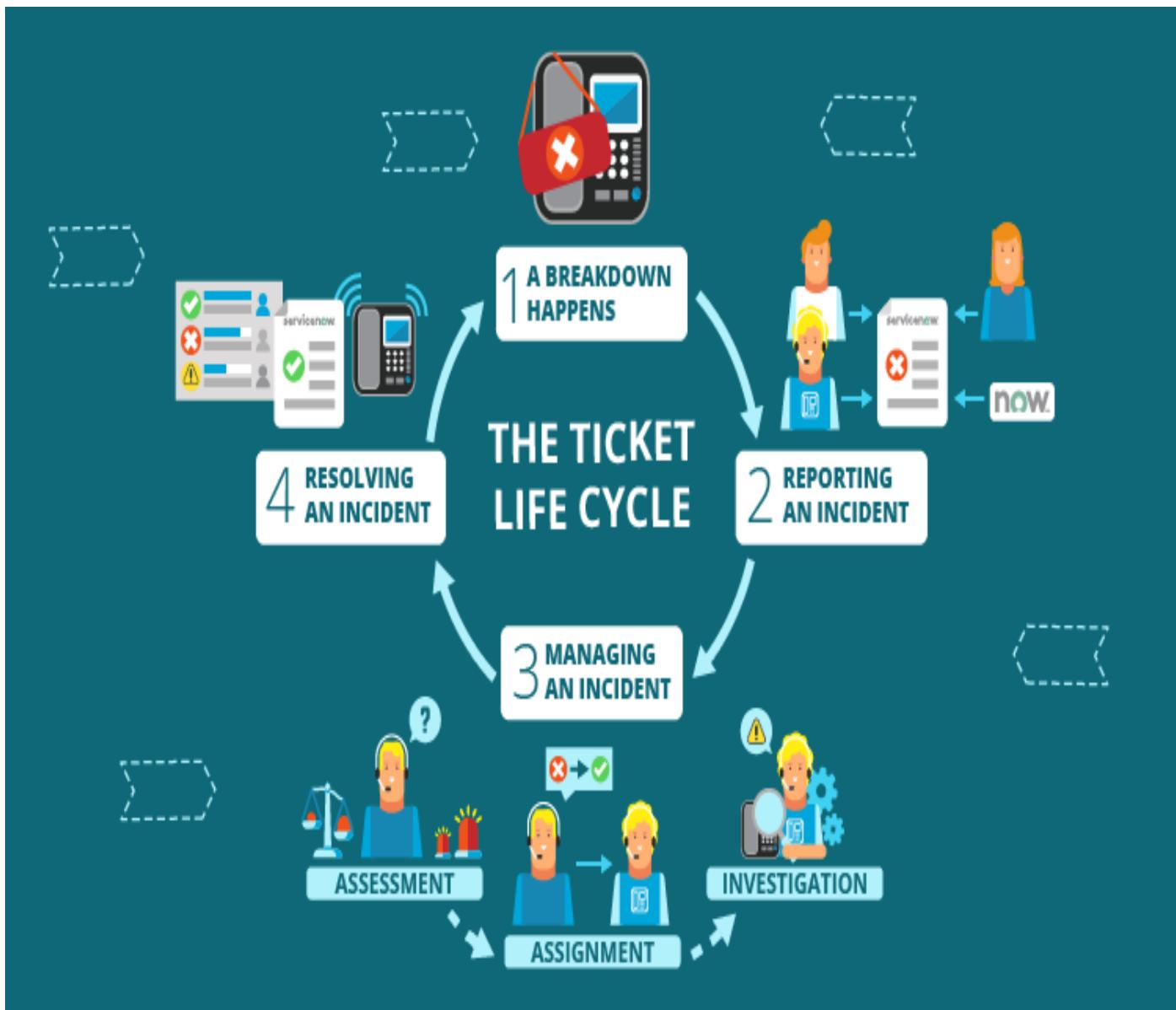
There are a number of work types used for IT Service Management. When assigned to the correct type, work gets the handling appropriate to it.

The types are:

- Service request
- Incident
- Change

Incident - INC0011211

Number	INC0011211	Opened	2015-07-07 12:02:19
Caller	Enterprise Manager Connector	Opened by	System Administrator
Location	Grand Rapids	Contact type	Phone
Category	EM Incident	State	Active
Subcategory	-- None --	Assignment group	EMSampleGroup
Configuration item		Assigned to	
Impact	1 - High		
Urgency	2 - Medium		
Priority	2 - High		
Short description	CPU Utilization for 1 is 19.409%, crossed warning () or critical (0) threshold.		
Related Search Results >			





Service request

A service request is a request from a user for information, advice, or access to an IT service, such as:

- Associate asks for a access of particular server.
- A telecom coordinator requests a new desk phone on behalf of someone they support.

Incident

Let's suppose you work in an organization, and you are using outlook for connectivity, then you face issue while opening the outlook you are not able to connect to the outlook.

Then you will raise this concern to the related team (who is responsible for outlook).

Problem

The corresponding team will look into it, and try to fix it. If they don't find any root cause for the incident and if they get multiple incidents for the same underlying cause then the incident will be considered as a problem. They will give you some workaround until the root cause is determined.

- If multiple users are facing the same issue with the outlook.

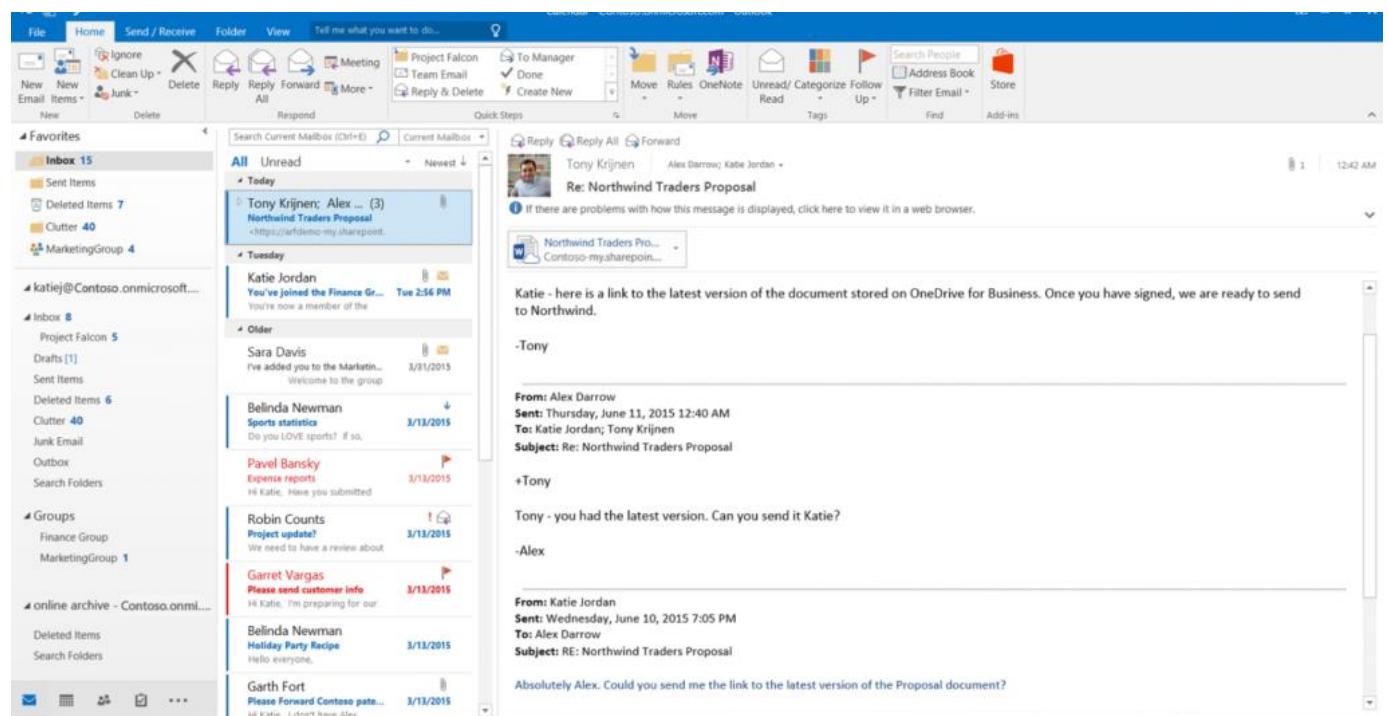
Change

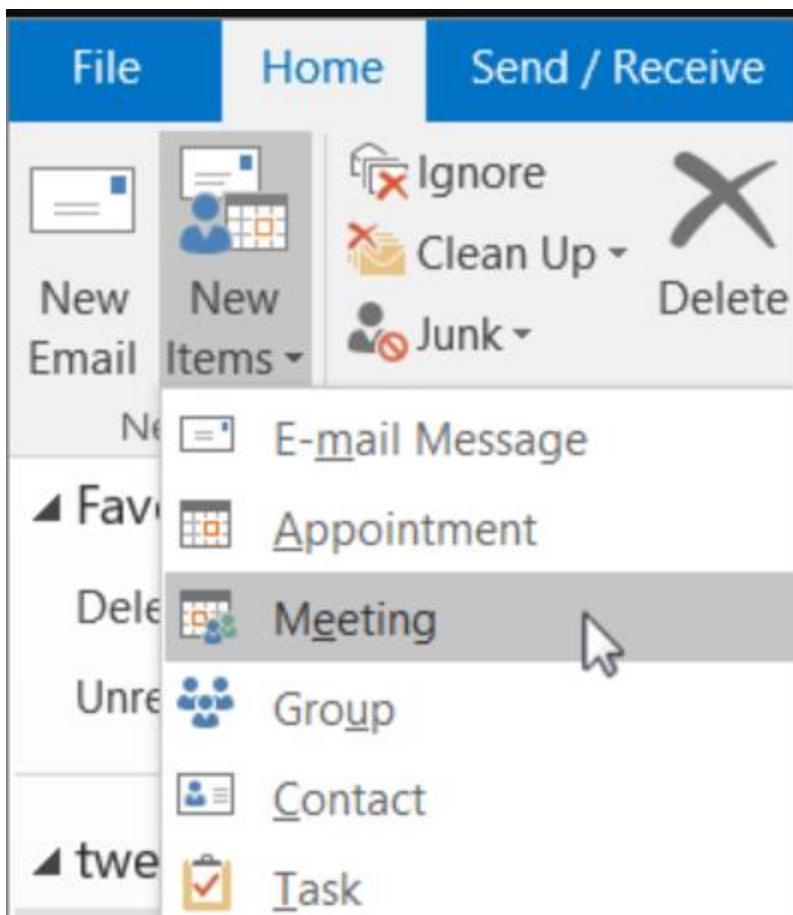
If after investigation, the technical team found that the outlook server requires patching to resolve this issue. the respective team will raise the change request to apply OS patch on the server.

A change is an addition, modification, or removal of anything that could affect Production. This may include IT services, configuration items, processes, documentation, and other related elements.

Outlook:

Outlook allows you to send and receive email messages, manage your calendar, store names and numbers of your contacts, and track your tasks. However, even if you use Outlook every day, you might not know some of the cool things it can do to help you be more productive.





Screenshot of Microsoft Teams meeting creation interface:

Title: Untitled - Meeting

Send:

Required:

Optional:

Start time: Sun 02-10-2022 17:30 Time zones

End time: Sun 02-10-2022 18:00

Location: Microsoft Teams Meeting

Microsoft Teams meeting

Join on your computer, mobile app or room device
[Click here to join the meeting](#)

Meeting ID: 290 369 452 096
Passcode: U7SeCn
[Download Teams](#) | [Join on the web](#)

Join with a video conferencing device

Teams:

Screenshot of Microsoft Teams messaging interface showing a conversation between Jennifer Jin and Tracy Johnson:

q4-planning Today

Jennifer Jin EXTERNAL 11:40 AM Hey Tracy, can you send me the vendor's security white paper again?

You 11:40 AM Security White Paper U... 151 KB

You 11:40 AM Here it is

Jennifer Jin EXTERNAL 11:41 AM Awesome thank you. This is much faster than email or switching apps all the time

Tracy Johnson 11:40 AM Hey Tracy, can you send me the vendor's security white paper again?
[Reply](#)

Tracy Johnson 11:40 AM Security White Paper UPDATE.pdf
[Reply](#)

Tracy Johnson 11:40 AM Here it is
[Reply](#)

Jennifer Jin 11:41 AM Awesome thank you. This is much faster than email or switching apps all the time
[Reply](#)

Start a new conversation. Type @ to mention someone.

1. What is Cloud computing?

Cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing.

On-demand delivery indicates that AWS has the resources you need, when you need them. You don't need to tell us in advance that you're going to need them. Suddenly you find yourself needing 300 virtual servers. Well, just a few clicks and launch them. Or you need 2000 TB of storage. You don't have to tell us in advance, just start using the storage you need, when you need it. Don't need them anymore, just as quickly, you can return them and stop paying immediately. That kind of flexibility is just not possible when you're managing your own datacenters.

The idea of IT resources is actually a big part of the AWS philosophy. We often get asked why AWS has so many products and the answer is really simple: Because businesses need them. If there are IT elements that are common across a number of businesses, then this is not a differentiator.

What are public, private and hybrid clouds?

1. Public Cloud-

Public clouds are the most common type of cloud computing deployment. The cloud resources (like servers and storage) are owned and operated by a third-party cloud service provider and delivered over the internet. With a public cloud, all hardware, software and other supporting infrastructure are owned and managed by the cloud provider. In a public cloud, you share the same hardware, storage and network devices with other organisations

Ex. Microsoft Azure, Amazon Web Services, Google Cloud Platform

Advantages of public clouds:

Lower costs- No need to purchase hardware or software and you pay only for the service you use.

No maintenance- Your service provider provides the maintenance.

Near-unlimited scalability- On-demand resources are available to meet your business needs.

High reliability- a vast network of servers ensures against failure.

2. Private Cloud-

A private cloud consists of cloud computing resources used exclusively by one business or organisation. The private cloud can be physically located at your organisation's on-site

datacenter or it can be hosted by a third-party service provider. But in a private cloud, the services and infrastructure are always maintained on a private network and the hardware and software are dedicated solely to your organisation.

In this way, a private cloud can make it easier for an organisation to customise its resources to meet specific IT requirements. Private clouds are often used by government agencies, financial institutions, any other mid- to large-size organisations with business-critical operations seeking enhanced control over their environment.

Advantages of a private cloud:

More flexibility – your organisation can customise its cloud environment to meet specific business needs.

More control – resources are not shared with others, so higher levels of control and privacy are possible.

More scalability – private clouds often offer more scalability compared to on-premises infrastructure.

3. Hybrid Cloud-

A hybrid cloud platform gives organisations many advantages—such as greater flexibility, more deployment options, security, compliance and getting more value from their existing infrastructure. When computing and processing demand fluctuates, hybrid cloud computing gives businesses the ability to seamlessly scale up their on-premises infrastructure to the public cloud to handle any overflow—without giving third-party datacenters access to the entirety of their data. Organisations gain the flexibility and innovation the public cloud provides by running certain workloads in the cloud while keeping highly sensitive data in their own datacenter to meet client needs or regulatory requirements.

Advantages of the hybrid cloud:

Control—your organisation can maintain a private infrastructure for sensitive assets or workloads that require low latency.

Flexibility—you can take advantage of additional resources in the public cloud when you need them.

Cost-effectiveness—with the ability to scale to the public cloud, you pay for extra computing power only when needed.

Ease—transitioning to the cloud does not have to be overwhelming because you can migrate gradually—phasing in workloads over time.

What is cloud computing service model?

IaaS, PaaS and SaaS are the three most popular types of cloud service offerings. (They are sometimes referred to as cloud service models or cloud computing service models.)

1. IAAS–

Infrastructure As A Service (IAAS) is means of delivering computing infrastructure as on-demand services. It is one of the three fundamental cloud service model servers storage network operating system. In the user purchasing servers, software datacenter space, or network equipment and rent those resources as a fully outsourced service can demand model. It allows dynamic scaling and the resources are distributed as a service. It generally includes multiple-user on a single piece of hardware.

Advantages of IaaS –

- The resources can be deployed by the provider to a customer's environment at any given time.
- Its ability to offer the users to scale the business based on their requirements.
- The provider has various options when deploying resources including virtual machines , applications , storage , networks.
- Its potential to handle a immense number of users.
- It is easy to expand and saves a lot of money. Companies can afford the huge costs associated with implementation of advanced technologies.

2. PAAS-

Platform As A Service (PAAS) is a cloud delivery model for applications composed of services managed by a third party. It provides elastic scaling of your application which allows developers to build applications and services over the internet and the deployment models include public, private and hybrid.

Advantages of PaaS –

- Programmers need not worry about what specific database or language the application has been programmed in.
- It offers developers to build applications without overhead of the underlying operating system or infrastructure.
- Provides freedom to developers to focus on application's design while the platform takes care of the language and the database.

3. SAAS-

Software As A Service (SAAS) allows users to run existing online applications and it is a model software that is deployed as a hosting service and is accessed over Output Rephrased/Re-written Text the internet or software delivery model during which software and its associated data are

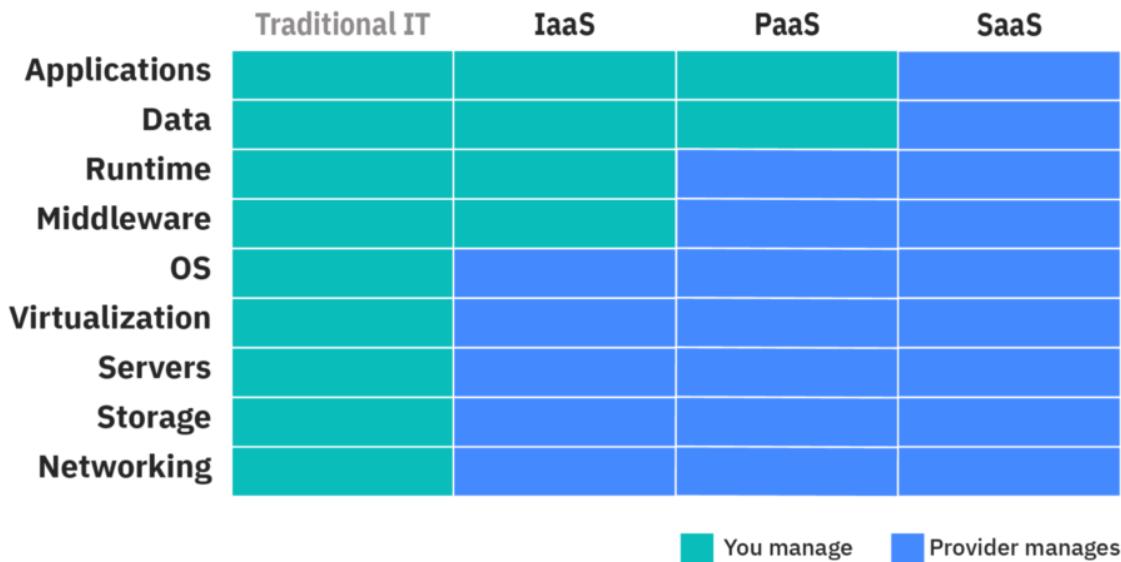
hosted centrally and accessed using their client, usually an online browser over the web. SaaS services are used for the development and deployment of modern applications.

Advantages of SaaS –

- It is a cloud computing service category providing a wide range of hosted capabilities and services. These can be used to build and deploy web based software applications.
- It provides a lower cost of ownership than on premises software. The reason is it does not require the purchase or installation of hardware or licenses.
- It can be easily accessed through a browser along a thin client.



Cloud Computing Services: Who Manages What?



2. Why AWS?

This course is gonna cover all the essential information that you need to understand, to be comfortable discussing AWS, to know why it's beneficial to your business.

AWS offers a massive range of services for every business, starting with basic elements, like compute, storage, and network security tools, through complex solutions like blockchain, machine learning, or artificial intelligence, and robot development platforms, all the way through

very specialized tool sets, like video production management systems, and orbital satellites you can rent by the minute.

All that, however, is way more than we have time to cover in a foundational class like this one. So let's simplify the conversation by starting with the fundamental cloud compute model.

I know it can be complicated, so let's take a look at our coffee shop.

This coffee shop is going to give us some real world examples to help you understand why AWS can change the way your IT operates.

Let's make Morgan the server, the barista. And I am the client, the customer. I make a request. In this case, it is for coffee. Now in the computing world, the request could be anything. It could be rain pattern analysis in South Africa, or the latest x-rays of your knee, or videos of kittens. Whatever is the business, basically a customer makes a request, and with permissions, the server responds to that request. All I want is a caffeinated beverage.

Morgan represents the server part of the client-server model. **In AWS, she would be called an Amazon Elastic Compute Cloud, or EC2, an EC2 instance, a virtual server.** So from an architectural point of view, the transaction we did is really simple to explain. I, the user, made a request to Morgan, the server. Morgan validated that the request was legitimate, in this case, did I give her money? Then she returned a response, which in this case, is a berry blaster with extra caramel shots.

Now in the real world, applications can get more complicated than just a single transaction with a single server. In a business solution that is more mature, it can get beautifully complex.

To avoid this complexity, we're going to start simple. We will build this discussion out so that it is easy for anyone to understand how these concepts build on each other. So, by the end, those complex concepts, they'll be easy to understand. Let's start with a key concept to AWS, and that is, you only pay for what you use.

This principle makes sense when you run a coffee shop. Employees are only paid when they're in the store working. If Rudy and Morgan are off the clock, well then they don't get paid. The store owner simply decides how many baristas are needed and then just pays for the hours they work. For example, the coffee shop is about to release a new drink, the Pumpkin Monster Spice. In anticipation of this launch, you could always staff your shop with a dozen baristas all day long, just in case you suddenly get an unexpected rush at some point in the day. Only, let's be honest. For most of your day, you don't have near enough customers to justify paying for all those employees.

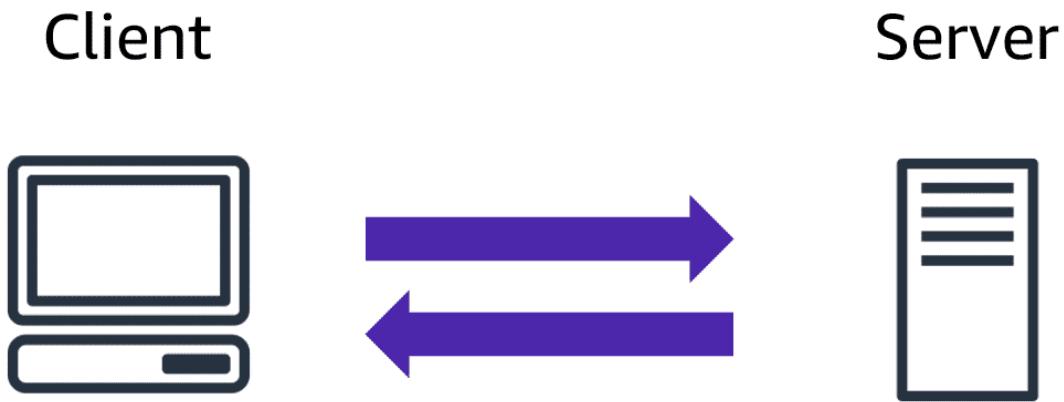
And yet, this is exactly what happens in an on-premises data center. You can't just snap your fingers and triple your capacity. At AWS, you don't pre-pay for anything. And you don't have to worry about capacity constraints.

When you need instances, or baristas, you just click a button, and you have them. And when you don't need them, another click, and they go away, and you stop paying for them. The same way you don't pay for employees for hours that they're not working.

So, pay for what you need, becomes the first key value of many for running your business on AWS. And that is really why we're here, to help you understand how AWS is built to help you run your business better.

What is a client-server model?

You just learned more about AWS and how almost all of modern computing uses a basic client-server model. Let's recap what a client-server model is.



In computing, a client can be a web browser or desktop application that a person interacts with to make requests to computer servers. A server can be services such as Amazon Elastic Compute Cloud (Amazon EC2), a type of virtual server. For example, suppose that a client makes a request for a news article, the score in an online game, or a funny video. The server evaluates the details of this request and fulfills it by returning the information to the client.

AWS Market Share- IaaS/PaaS market (Source: Jefferies)

Cloud vendor	Annualized revenue	% of market	Year-over-year growth
Amazon Web Services	\$18.34 billion	51%	42%
Microsoft Azure	\$6.17 billion	17%	89%
IBM Cloud	\$4.03 billion	11%	22%
Google Cloud Platform	\$2.05 billion	6%	125%
Alibaba Cloud	\$1.79 billion	5%	92%
Salesforce	\$1.78 billion	5%	31%
Oracle Cloud	\$1.59 billion	4%	20%
Subtotal	\$35.75 billion	86%	54%
Total Gartner estimate	\$41.79 billion	100%	33%

3. Elastic Compute Cloud

EC2 stands for Amazon **Elastic Compute Cloud**.

Your business, whether it be in healthcare, manufacturing, insurance, or delivering video content to millions of users all around the world, are also using this model to deliver products, resources, or data to your end users. And you're going to need servers to power your business and your applications. You need raw compute capacity to host your applications and provide the compute power that your business needs. When you're working with AWS, those servers are virtual. And the service you use to gain access to virtual servers is called EC2.

Using EC2 for compute is **highly flexible, cost effective, and quick when you compare it to running your own servers on premises** in a datacenter that you own. The time and money it takes to get up and running with on-premises resources is fairly high. When you own your own fleet of physical servers, you first have to do a bunch of research to see what type of servers you want to buy and how many you'll need. Then you purchase that hardware up front. You'll wait for multiple weeks or months for a vendor to deliver those servers to you. You then take them to a datacenter that you own or rent to install them, rack and stack them, and wire them all up. Then you make sure that they are secure and powered up and then they're ready to be used. Only then can you begin to host your applications on top of these servers. The worst part is, once you buy these servers you are stuck with them whether you use them or not.

With EC2, it's much easier to get started. AWS took care of the hard part for you already. AWS already built and secured the datacenters. AWS has already bought the servers, racked and stacked them, and they are already online ready to be used. AWS is constantly operating a massive amount of compute capacity. And you can use whatever portion of that capacity when

you need it. All you have to do is request the EC2 instances you want and they will launch and boot up, ready to be used within a few minutes. Once you're done, you can easily stop or terminate the EC2 instances. You're not locked in or stuck with servers that you don't need or want. Your usage of EC2 instances can vary greatly over time. And you only pay for what you use. Because **with EC2, you only pay for running instances, not stopped or terminated instances.**

EC2 runs on top of physical host machines managed by AWS using virtualization technology. When you spin up an EC2 instance, you aren't necessarily taking an entire host to yourself. Instead, you are sharing the host with multiple other instances, otherwise known as virtual machines. And a hypervisor running on the host machine is responsible for sharing the underlying physical resources between the virtual machines. This idea of sharing underlying hardware is called multitenancy. The hypervisor is responsible for coordinating this multitenancy and it is managed by AWS. The hypervisor is responsible for isolating the virtual machines from each other as they share resources from the host. This means EC2 instances are secure. Even though they may be sharing resources, one EC2 instance is not aware of any other EC2 instances also on that host. They are secure and separate from each other.

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) provides secure, resizable compute capacity in the cloud as Amazon EC2 instances.

Imagine you are responsible for the architecture of your company's resources and need to support new websites. With traditional on-premises resources, you have to do the following:

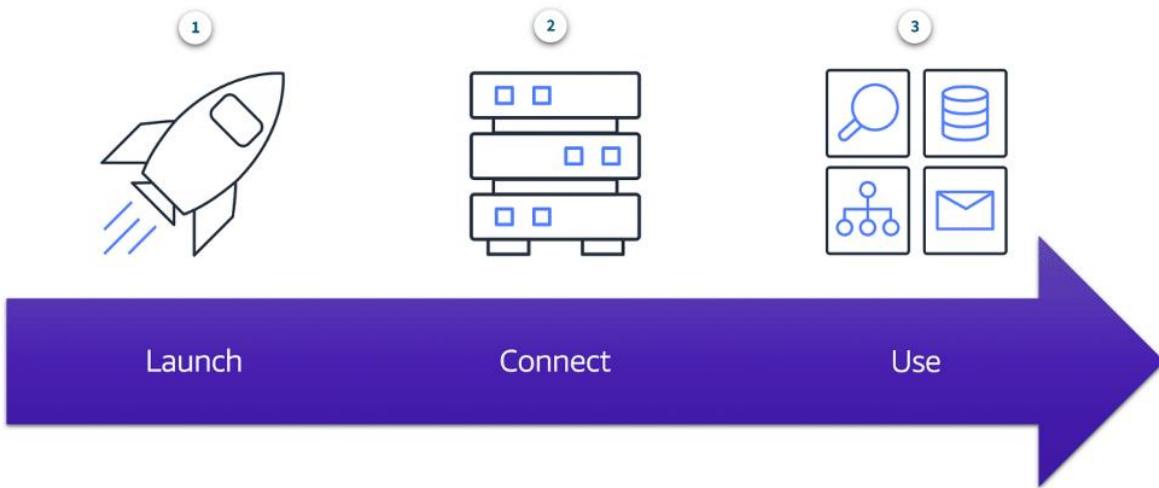
- Spend money upfront to purchase hardware.
- Wait for the servers to be delivered to you.
- Install the servers in your physical datacenter.
- Make all the necessary configurations.

By comparison, with an Amazon EC2 instance you can use a virtual server to run applications in the AWS Cloud.

- You can provision and launch an Amazon EC2 instance within minutes.
- You can stop using it when you have finished running a workload.
- You pay only for the compute time you use when an instance is running, not when it is stopped or terminated.
- You can save costs by paying only for server capacity that you need or want.

How Amazon EC2 works

To learn more, select each marker.



Amazon EC2 is a service that provides reusable compute capacity in the cloud.

Free tier details:-

- 12 MONTHS FREE
- 750 hours per month of Linux, RHEL, or SLES t2.micro or t3.micro instance dependent on region.
- 750 hours per month of Windows t2.micro or t3.micro instance dependent on region.
- 30 GB volume per month free.

url – <https://aws.amazon.com/ec2/pricing/>

Benefits of amazon ec2-

1. Elastic web-scale computing, i.e. its scalable.
2. Completely controlled by the user.
3. Can be easily used for cloud hosting.
4. Can be configured with other AWS services.
5. Other benefits such as its Reliable, Secure, Inexpensive, easy to start.

Features of EC2-

1. Virtual computing service.
2. Preconfigured templates are available or else we are create our own templates.

3. Various config of CPU, Memory, storage etc. can be done.
4. Secure login using keypairs
5. Persistent volumes/storages are available for your data.
6. Multiple physical locations for your resources.
7. Firewall that enables you to specify the ports, protocols, IP etc.
8. Static IP addresses available.
9. Isolated Virtual networks possible.

Types/categories of instances-

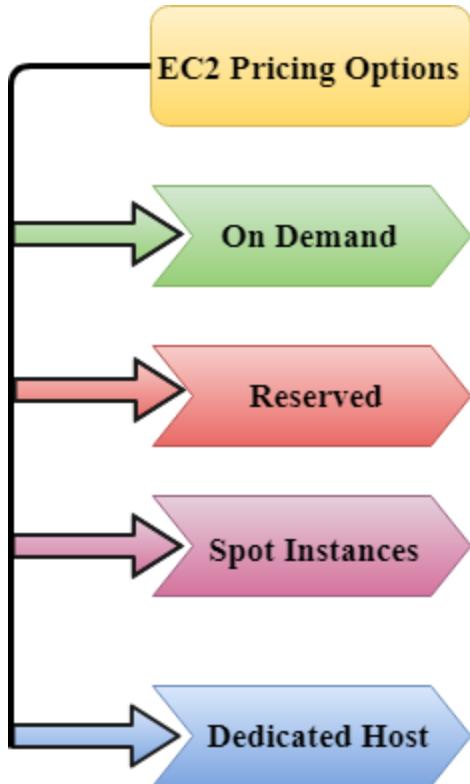
1. General Purpose Instances.
2. Compute Optimized Instances.
3. GPU Graphic instances/ Accelerated computing
4. Memory Optimized
5. Storage Optimized instances

url – <https://aws.amazon.com/ec2/instance-types/>

What all OS can we launch?

1. Windows
2. RHEL
3. Fedora
4. CentOS
5. Ubuntu
6. Suse Linux and many many more.

EC2 Pricing Options –



On Demand –

- It allows you to pay a fixed rate by the hour or even by the second with no commitment.
- Linux instance is by the second and windows instance is by the hour.
- On Demand is perfect for the users who want low cost and flexibility of Amazon EC2 without any up-front investment or long-term commitment.
- It is suitable for the applications with short term, spiky or unpredictable workloads that cannot be interrupted.
- It is useful for the applications that have been developed or tested on Amazon EC2 for the first time.
- On Demand instance is recommended when you are not sure which instance type is required for your performance needs.

Reserved –

- It is a way of making a reservation with Amazon or we can say that we make a contract with Amazon. The contract can be for 1 or 3 years in length.
- In a Reserved instance, you are making a contract means you are paying some upfront, so it gives you a significant discount on the hourly charge for an instance.
- It is useful for applications with steady state or predictable usage.
- It is used for those applications that require reserved capacity.

- Users can make up-front payments to reduce their total computing costs. For example, if you pay all your upfronts and you do 3 years contract, then only you can get a maximum discount, and if you do not pay all upfronts and do one year contract then you will not be able to get as much discount as you can get. If you do 3 year contract and pay all the upfronts.

Spot Instances –

- It allows you to bid for a price whatever price that you want for instance capacity, and providing better savings if your applications have flexible start and end times.
- Spot Instances are useful for those applications that have flexible start and end times.
- It is useful for those applications that are feasible at very low compute prices.
- It is useful for those users who have an urgent need for large amounts of additional computing capacity.
- EC2 Spot Instances provide less discounts as compared to On Demand prices.
- Spot Instances are used to optimize your costs on the AWS cloud and scale your application's throughput up to 10X.
- EC2 Spot Instances will continue to exist until you terminate these instances.

Dedicated Hosts –

- A dedicated host is a physical server with EC2 instance capacity which is fully dedicated to your use.
- The physical EC2 server is the dedicated host that can help you to reduce costs by allowing you to use your existing server-bound software licenses. For example, VMware, Oracle, SQL Server depending on the licenses that you can bring over to AWS and then they can use the Dedicated host.
- Dedicated hosts are used to address compliance requirements and reduces host by allowing to use your existing server-bound server licenses.
- It can be purchased as a Reservation for up to 70% off On-Demand price.

EC2 Instance Families-

1. General Purpose Instances (e.g., t2, t3, m5) –

These instances provide a balance of compute, memory, and network resources. They are suitable for a wide range of workloads, including web servers, small databases, and development environments.

2. Compute Optimized Instances (e.g., c5, c6g) –

These instances are designed for computationally intensive workloads that require high performance processors. They are suitable for applications such as scientific modeling, gaming servers, and high-performance web servers.

3. Memory Optimized Instances (e.g., r5, x1e) –

These instances are optimized for memory-intensive workloads that require high memory capacity. They are suitable for applications like in-memory databases, real-time big data analytics, and high-performance computing.

4. Accelerated Computing Instances (e.g., p3, g4) –

These instances are equipped with specialized hardware accelerators such as GPUs (Graphics Processing Units) or FPGAs (Field-Programmable Gate Arrays). They are suitable for tasks like machine learning, data processing, and video encoding.

5. Storage Optimized Instances (e.g., i3, d2) –

These instances are designed for applications that require high-speed, low-latency storage. They are suitable for data warehousing, distributed file systems, and large-scale analytics.

4. EC2 Backup

let alone protect it from unexpected data loss as a result of hardware failure, software corruption, accidental deletion, malicious attack, or an unpredictable disaster. More issues may arise still when it comes to managing AWS EC2 environments and protecting data stored in the cloud.

In short, AWS EC2 backup instances, you should choose one of the following options:

Take an EBS snapshot;

Create a new AMI;

Design an AWS EC2 Backup plan;

Automate AWS EC2 backup with a third party solution

AWS Backup is a rather new addition to the rich set of AWS services and tools, and is definitely worth your attention. AWS Backup is a valuable tool which can help you automatically back up and protect your data and applications in the AWS cloud as well as on-premises IT environments.

How to Backup AWS EC2 Instances

AWS is a high-performance, constantly evolving cloud computing platform that allows you to store data and applications in the cloud environment. AWS can provide you with the tools you need to create EC2 instances which act as virtual servers with varying CPU, memory, storage, and networking capacity.

Currently, there are three ways to back up AWS EC2 instances: taking EBS snapshots, creating AMIs, or designing an AWS Backup plan. Let's take a closer look at each of these approaches and see how they differ.

Taking EBS Snapshots

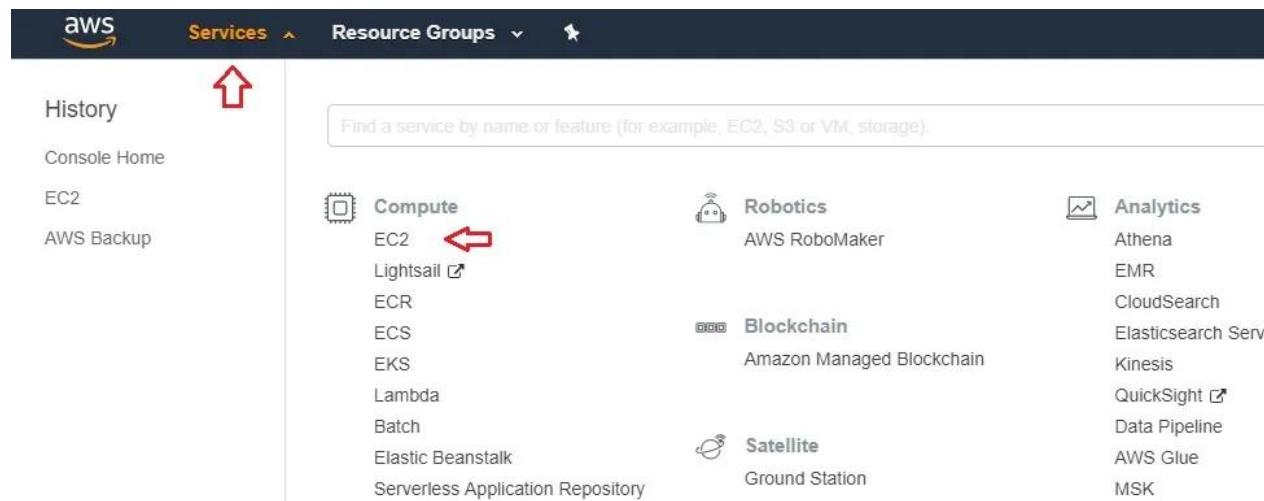
If you want to back up an AWS EC2 instance, you should create snapshots of EBS volumes, which are stored with the help of Amazon Simple Storage Service (S3). Snapshots can capture all data within EBS volumes and create their exact copies. Moreover, these EBS snapshots can then be copied and transferred to another AWS region to ensure safe and reliable storage of critical data. Thus, in case of a disaster or accidental data loss, you can be sure that you have a backup copy securely stored in a remote location which you can use for restoring critical data.

Prior to running AWS EC2 backup, it is recommended that you stop the instance or at least detach an EBS volume which is about to be backed up. This way, you can prevent failure or errors from occurring and affecting the newly created snapshots.

Please note that, for security purposes, some sensitive information has been removed.

To back up AWS EC2 instance, you need to take the following steps:

1. Sign in to your AWS account to open the AWS console.
2. Select Services in the top bar and click EC2 to launch the EC2 Management Console



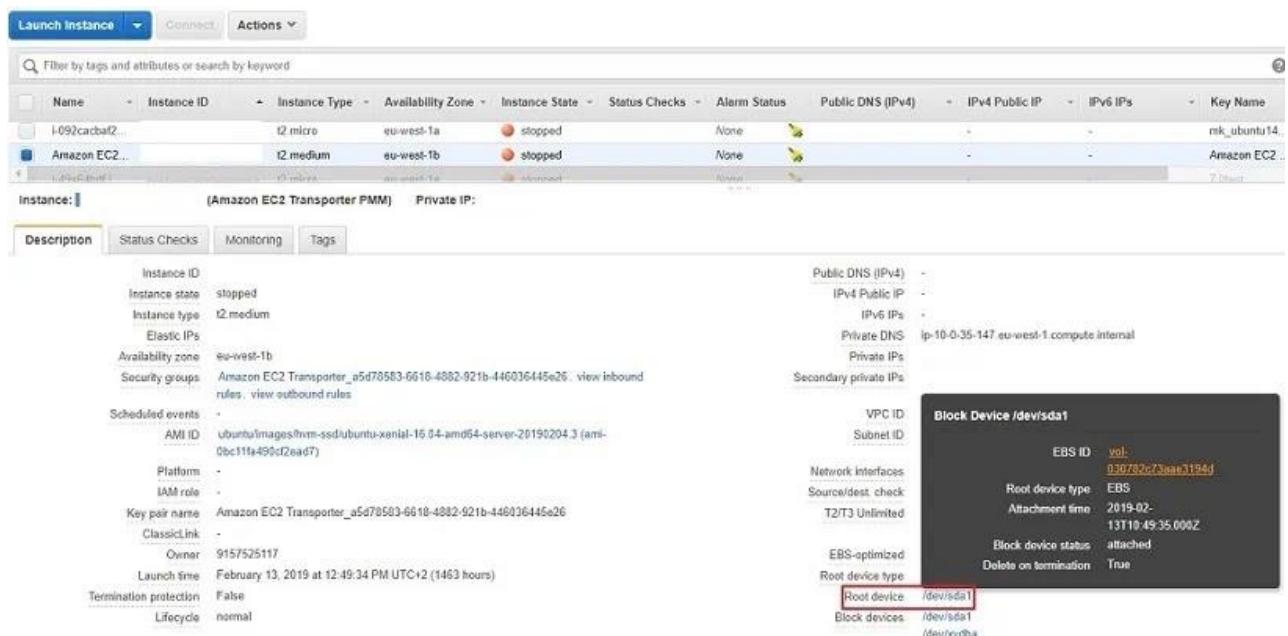
3. select running instances and choose the instance you would like to backup.

Resources

You are using the following Amazon EC2 resources in the EU West (Ireland) region:

3 Running Instances		2 Elastic IPs
0 Dedicated Hosts		25 Snapshots
32 Volumes		0 Load Balancers
330 Key Pairs		314 Security Groups
0 Placement Groups		

- In the bottom pane, you can view the central technical information about the instance. In the Description tab, find the Root device section and select the /dev/sda1 link



The screenshot shows the AWS EC2 console with the following details:

Instances Overview:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name
i-092cardaf2...	i2.micro	eu-west-1a	stopped	None	None	None	-	-	-	rsk_ubuntu14...
Amazon EC2...	i2.medium	eu-west-1b	stopped	None	None	None	-	-	-	Amazon EC2...
ami-092...	ami-092...	eu-west-1a	terminated	None	None	None	-	-	-	-

Description Tab (selected):

- Instance ID: i-092cardaf2...
- Instance state: stopped
- Instance type: i2.medium
- Elastic IPs: -
- Availability zone: eu-west-1b
- Security groups: Amazon EC2 Transporter_a5d70583-6618-4882-921b-440036445e26, view inbound rules, view outbound rules
- Scheduled events: -
- AMI ID: ubuntu/images/hvm-ssd/ubuntu-xenial-16-04-amd64-server-20190204.3 (ami-0bc11fa490cd2ead7)
- Platform: -
- IAM role: -
- Key pair name: Amazon EC2 Transporter_a5d70583-6618-4882-921b-440036445e26
- ClassicLink: -
- Owner: 9157525117
- Launch time: February 13, 2019 at 12:49:34 PM UTC+2 (1463 hours)
- Termination protection: False
- Lifecycle: normal

Block Device /dev/sda1 (Detailed View):

EBS ID: vol-030732c73aae3184f	Root device type: EBS
Subnet ID: -	Attachment time: 2019-02-13T10:49:35.000Z
Network interfaces: -	Block device status: attached
Source/dest. check: T2/T3 Unlimited	Delete on termination: True
EBS-optimized: -	Root device: /dev/sda1
Root device type: EBS	Block devices: /dev/sda1, /dev/xvdb1

- The Create Snapshot box should open, where you can add a description for the snapshot to make it distinct from other snapshots, as well as assign tags to easily monitor this snapshot. Click Create Snapshot.

Create Snapshot

Volume vol-030782c73aae3194d ⓘ

Description Snapshot 15/04/2019 ⓘ

Encrypted Not Encrypted ⓘ

Key	(127 characters maximum)	Value	(255 characters maximum)
This resource currently has no tags			
Choose the Add tag button or click to add a Name tag			
Add Tag	50 remaining	(Up to 50 tags maximum)	

* Required

[Cancel](#) [Create Snapshot](#)

The snapshot creation should start and be completed in a minimal amount of time. The main factor here is the size of data in your Amazon EBS volume.

After the snapshot creation is complete, you can find your new snapshot by selecting the Snapshots section in the left pane. As you can see, we have successfully created a point-in-time copy of the EBS volume, which can later be used to restore your EC2 instance.

The screenshot shows the AWS Management Console with the EBS service selected. The left sidebar shows 'ELASTIC BLOCK STORE' with 'Volumes' and 'Snapshots' selected. The main area displays a table of snapshots, each with a checkbox, Name, Snapshot ID, Size, Description, and Status (all marked as 'completed'). A 'Create Snapshot' button is visible at the top left of the main content area.

Name	Snapshot ID	Size	Description	Status
snap-0017f4af710d...	8 GiB	Created by CreateImage(i-047f91a2a2fd1fda8) for a...	completed	
Recovery poi...	127 GiB	Created by NAKIVO Backup & Replication on Thu, ...	completed	
Recovery poi...	10 GiB	Created by NAKIVO Backup & Replication on Wed, ...	completed	
Recovery poi...	10 GiB	Created by NAKIVO Backup & Replication on Wed, ...	completed	
Recovery poi...	8 GiB	Created by NAKIVO Backup & Replication on Wed, ...	completed	
	20 GiB	Created by CreateImage(i-01f04e6b4e81213de) for ...	completed	
Recovery poi...	8 GiB	Created by NAKIVO Backup & Replication on Fri, O...	completed	
Recovery poi...	8 GiB	Created by NAKIVO Backup & Replication on Mon, ...	completed	

For this purpose, you need to select the snapshot of the backed up volume, press the Actions button above, and click Create Volume. Following the prompts, configure the volume details (volume type, size, IOPS, availability zone, tags). Then, click Create Volume for the new volume to be created, which can later be added to the AWS EC2 instance of your choice.

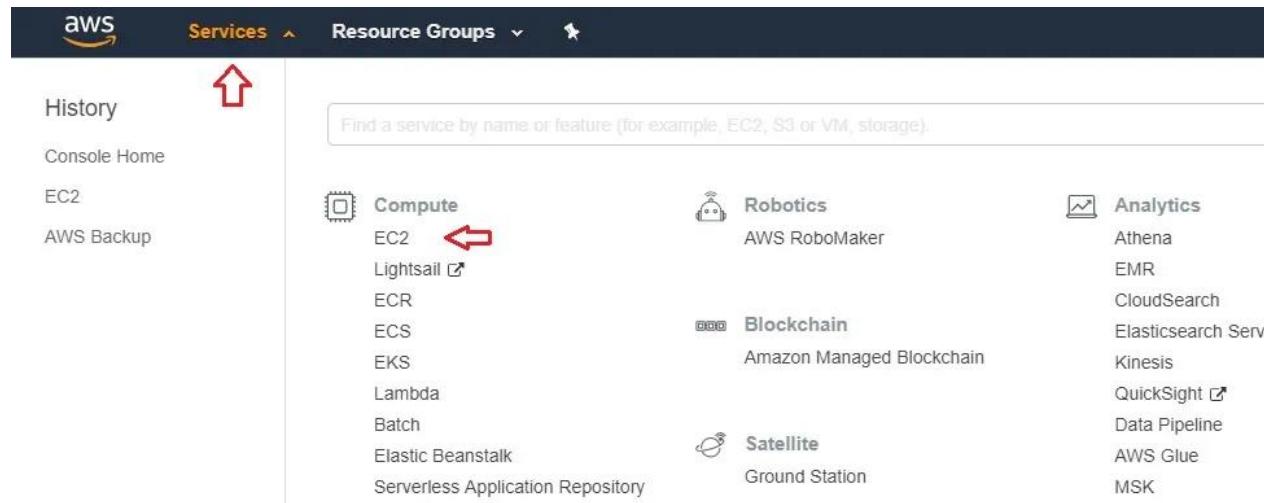
Creating a new AMI

The next approach to performing AWS EC2 backups is creating an Amazon Machine Image (AMI) of your AWS EC2 instances. An AMI contains all the information required for creating an EC2 instance in the AWS environment, including configuration settings, the root volume

template, launch permissions, and block device mapping. Basically, the AMI can act as a template for launching a new AWS EC2 instance and replacing the corrupted one. Note that, prior to creating the new AMI, it is recommended that you stop the AWS EC2 instance which you want to back up.

To create a new AMI and ensure AWS EC2 backup, you should do the following:

- Sign in to your AWS account to open the AWS console.
- Select Services in the top bar and click EC2 to launch the EC2 Management Console.



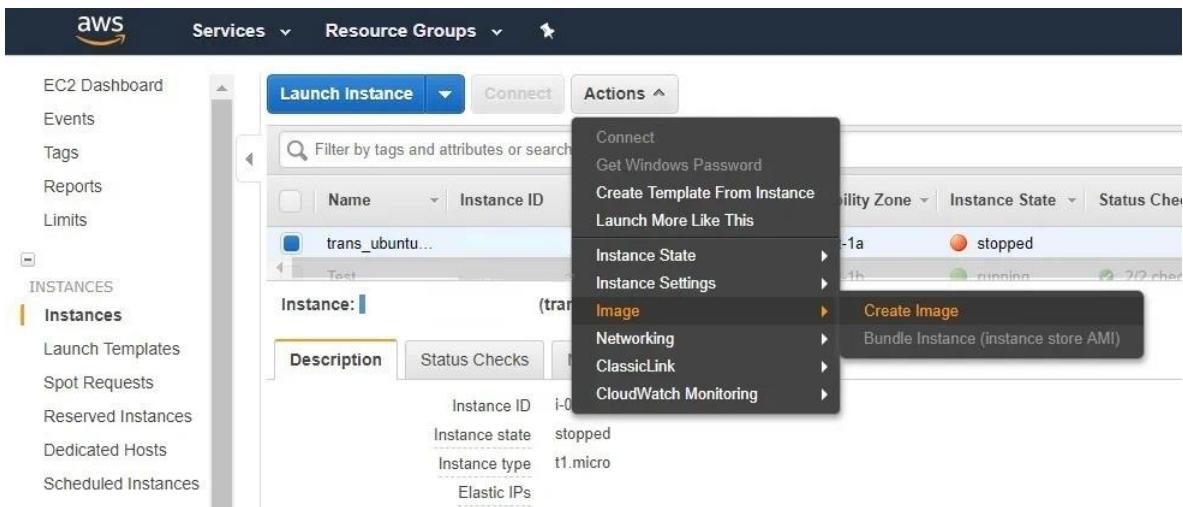
- Select Running Instances and choose the instance you want to back up.

Resources

You are using the following Amazon EC2 resources in the EU West (Ireland) region:

3 Running Instances	2 Elastic IPs
0 Dedicated Hosts	25 Snapshots
32 Volumes	0 Load Balancers
330 Key Pairs	314 Security Groups
0 Placement Groups	

- Click Actions > Image > Create Image.



- The Create Image menu should open. Here, you can specify the image name, add the image description, enable/disable reboot after the AMI creation, and configure instance volumes.

Do note that when you create an EBS image, an EBS snapshot should also be created for each of the above volumes. You can access these snapshots by going to the Snapshots section.

Create Image

Instance ID	Ubuntu_ubuntu...
Image name	Ubuntu_backup
Image description	Created 15/04/2019
No reboot	<input checked="" type="checkbox"/>

Instance Volumes

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-03dd3803ebdfc92	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume 

Total size of EBS Volumes: 8 GiB
When you create an EBS image, an EBS snapshot will also be created for each of the above volumes.

[Cancel](#) [Create Image](#)

- Click Create Image.
- The image creation process should now start. Click the link to view the pending AMI.
- It should take some time for the new AMI to be created. You can start using the AMI when its status switches from pending to available.

After the AMI has been successfully created, it can then be used to create a new AWS EC2 instance, which will be an exact copy of the original instance. For this purpose, simply go to the Instances section, click Launch Instance, select the AMI you have created in the My AMIs section, and follow the prompts to finish the instance creation.

Creating AMIs is arguably a more effective backup strategy than taking EBS snapshots. This is due to the fact that AMIs often contain EBS snapshots as well as a software configuration which allows you to simply and easily launch the new AWS EC2 instance in just a few clicks, created free of charge (you only pay for snapshot storage).

Automating AWS EC2 backup

Previously, the only way to automate AWS EC2 backup was by running scripts or using API calls, which was a very challenging and resource-intensive process. The person responsible for backup automation had to be highly proficient in scripting in order to avoid any issues and inconsistencies. However, there was still a high risk that you would waste your time, effort, and money on a backup job configuration and still be left with failed or corrupted AWS EC2 backups.

To create an AWS backup plan, take the following steps:

1. Sign in to your AWS account to open the AWS Management Console.
2. Select Services in the top bar and then type AWS Backup in the search bar. Click Backup plans in the left pane.
3. Press the Create Backup plan button.

- Here, you have three start options: Start from an existing plan, Build a new plan, and Define a plan using JSON. Click Info if you want to learn more about available options to help you make the right decision.

As we don't have any existing backup plans, let's build a new plan from scratch. Enter the new backup plan name and proceed further.

The screenshot shows the 'Create Backup plan' page in the AWS Backup console. The navigation path is: AWS Backup > Backup plans > Create Backup plan. The main section is titled 'Start options' with the sub-instruction: 'Choose how you want to begin.' Below this, there are three options:

- Start from an existing plan: Describes creating a new Backup plan based on an existing one.
- Build a new plan: Describes entering configuration details to create a new Backup plan. This option is highlighted with a green border.
- Define a plan using JSON: Describes defining a plan using JSON.

Below the options, there is a 'Backup plan name' field containing 'AWSBackup-NAKIVO'. A red arrow points to this field, indicating it is the current focus. A note below the field states: 'Backup plan name is case sensitive. Must contain from 1 to 63 alphanumeric characters or hyphens.'

- The next step is Backup rule configuration. Here, you should specify the backup rule name.
- After that, you can set up a backup schedule. You should determine the backup frequency (Every 12 hours, Daily, Weekly, Monthly, Custom cron expression); backup window (Use backup window defaults or Customize backup window); backup lifecycle (Transition to cold storage and Expiration of the backup).

Backup rule configuration [Info](#)

Add a Backup rule by defining a backup schedule, backup window, and lifecycle rules. You can add additional Backup rules to this Backup plan later.

General

Rule name



NewBackupRule-NAKIVO

Backup rule name is case sensitive. Must contain from 1 to 63 alphanumeric characters or hyphens.

Schedule

Add a Backup rule by defining a backup schedule, backup window, and lifecycle rules.

Frequency



Daily ▾

Backup window



Use backup window defaults - *recommended* [Info](#)

Customize backup window

Lifecycle [Info](#)

Schedule transition to cold storage and expiration of the backup.

Transition to cold storage



Never ▾

Expire



Never ▾

- At this step, you should select the backup vault for storing your recovery points (the ones created by this Backup rule). You can click Create new Backup vault if you want to have a new customizable vault. You can also use the existing Backup vault if you have one. Alternatively, you can choose the default AWS Backup vault.

Backup vault Info

Specify the Backup vault that recovery points created by this Backup rule are organized in.

Default

[Create new Backup vault](#)

- Next, you must add tags to recovery points and your backup plan in order to organize them and easily monitor their current status.

▼ Tags added to recovery points

Tags specified here are added to recovery points when they are created.

Key



RecoveryPoint1

Value - *optional*

PrimaryBackup

[Remove tag](#)

[Add tag](#)

Tags added to Backup plan

Tags specified here help organize and track your Backup plan

Key



BackupPlan1

Value - *optional*

NAKIVO-Backup

[Remove tag](#)

[Add tag](#)

You can assign resources to this Backup plan after the plan has been created.

You can add more rules to this Backup plan after the plan has been created.

[Cancel](#)

[Create plan](#)

▼ **Tags added to recovery points**
Tags specified here are added to recovery points when they are created.

Key	Value - <i>optional</i>	
RecoveryPoint1	PrimaryBackup	Remove tag

Add tag

Tags added to Backup plan
Tags specified here help organize and track your Backup plan

Key	Value - <i>optional</i>	
BackupPlan1	NAKIVO-Backup	Remove tag

Add tag

i You can assign resources to this Backup plan after the plan has been created.

i You can add more rules to this Backup plan after the plan has been created.

Cancel **Create plan**

After that, you can click Create plan to proceed to the next stage, the backup rule creation.

- Your backup plan has been successfully created. However, before you can run this plan and deploy it in your environment, you should also assign resources which need to be backed up. Click the Assign resources button, which can be found in the top bar.

Success
Backup plan "AWSBackup-NAKIVO" creation successful. You can now add additional schedule rules and assign resources to the Backup plan by selecting the Backup plan.

AWS Backup > Backup plans > AWSBackup-NAKIVO

AWSBackup-NAKIVO

[Delete](#) [View JSON](#)

Summary			
Backup plan name AWSBackup-NAKIVO	Version ID NmJhOTA1MmMtZWRjZC00NDM2LWE0N2EtNTFjZW NINDFmOGNi	Last modified Apr 15, 2019 @ 11:36:53 AM UTC+03:00	Last runtime -

Backup rules
Backup rules specify the backup schedule, backup window, and lifecycle rules.

Name	Backup vault
NewBackupRule-NAKIVO	Default

Resource assignments
Resource assignments specify which resources will be backed up by this Backup plan.

Name	IAM role ARN
Empty resources You don't have any resource assignments.	

[Assign resources](#)

- In the next menu, you can specify the resource assignment name and define the IAM (Identity and Access Management) role.

By selecting the IAM role, you specify what a user can or cannot do in AWS and determine which users are granted permission to manage selected AWS resources and services.

Additionally, you can assign resources to this Backup plan using tags or resource IDs, meaning that any AWS resources matching these key-pair values should be automatically backed up by this Backup plan.

Assign resources

General

Resource assignment name

 DailyBackups

Resource assignment name is case sensitive. Must contain from 1 to 63 alphanumeric characters or hyphens.

IAM role Info

AWS Backup will assume this IAM role when creating and managing recovery points on your behalf.


 Default role

If the AWS Backup default role is not present, one will be created for you with the correct permissions

 Choose an IAM role


Assign resources

Assign resources to this Backup plan using tags and resource IDs.

Assign by

 Tags

Key

 BackupPlan1

Value

 NAKIVO-Backup

 Remove

Assign by

 Tags

Key

 Daily

Value

 Backups

 Remove

Assign by

 Resource ID

Resource type

 EBS

Volume ID

 vol-9bc7342b

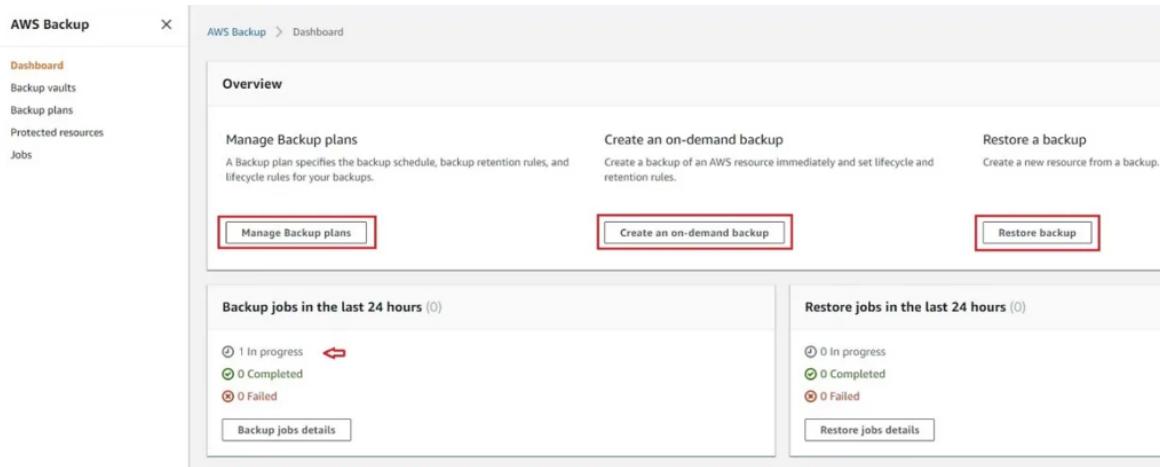
 Remove

 Add assignment

 Cancel

 Assign resources

- Click Assign resources to complete the configuration process. After that, the backup job should run automatically. You can go to the AWS Backup dashboard to see the current status of your backup jobs and verify that they are working as planned.



As you can see, our backup job is already in progress. In this menu, you can also Manage Backup plans, Create an on-demand backup, or Restore backup. Choose the required option and set up another data protection job in AWS environment following the prompts.

5.Elastic Block Store

When you're using Amazon EC2 to run your business applications, those applications need access to CPU, memory, network, and storage. EC2 instances give you access to all those different components, and right now, let's focus on the storage access. As applications run, they will oftentimes need access to block-level storage.

You can think of block-level storage as a place to store files. A file being a series of bytes that are stored in blocks on disc. When a file is updated, the whole series of blocks aren't all overwritten. Instead, it updates just the pieces that change. This makes it an efficient storage type when working with applications like databases, enterprise software, or file systems.

When you use your laptop or personal computer, you are accessing block-level storage. All block-level storage is in this case is your hard drive. EC2 instances have hard drives as well. And there are a few different types.

When you launch an EC2 instance, depending on the type of the EC2 instance you launched, it might provide you with local storage called instance store volumes. These volumes are physically attached to the host, your EC2 instances running on top of. And you can write to it just like a normal hard drive. The catch here is that since this volume is attached to the underlying physical host, if you stop or terminate your EC2 instance, all data written to the instance store volume will be deleted. The reason for this, is that if you start your instance from a stop state, it's likely that EC2 instance will start up on another host. A host where that volume does not exist. Remember EC2 instances are virtual machines, and therefore the underlying host can change between stopping and starting an instance.

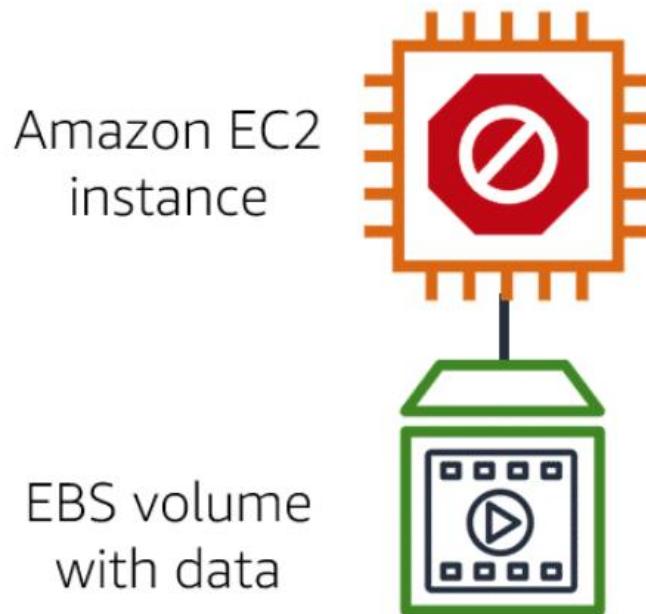
Because of this ephemeral or temporary nature of instance store volumes, they are useful in situations where you can lose the data being written to the drive. Such as temporary files, scratch data, and data that can be easily recreated without consequence.

All right, I'm telling you not to write important data to the drives that come with EC2 instances. I'm sure that sounds a bit scary because obviously you'll need a place to write data that persists outside of the life cycle of an EC2 instance. You don't want your entire database getting deleted every time you stop an EC2 instance. Don't worry, this is where a service called Amazon Elastic Block Store, or EBS, comes into play.

With EBS, you can create virtual hard drives, that we call EBS volumes, that you can attach to your EC2 instances. These are separate drives from the local instance store volumes, and they aren't tied directly to the host that your EC2 is running on. This means, that the data that you write to an EBS volume can persist between stops and starts of an EC2 instance.

EBS volumes come in all different sizes and types. How this works, is you define the size, type and configurations of the volume you need. Provision the volume, and then attach it to your EC2 instance. From there, you can configure your application to write to the volume and you're good to go. If you stop and then start the EC2 instance, the data in the volume remains.

Since the use case for EBS volumes is to have a hard drive that is persistent, that your applications can write to, it's probably important that you back that data up. EBS allows you to take incremental backups of your data called snapshots. It's very important that you take regular snapshots of your EBS volumes. This way, if a drive ever becomes corrupted, you haven't lost your data. And you can restore that data from a snapshot.



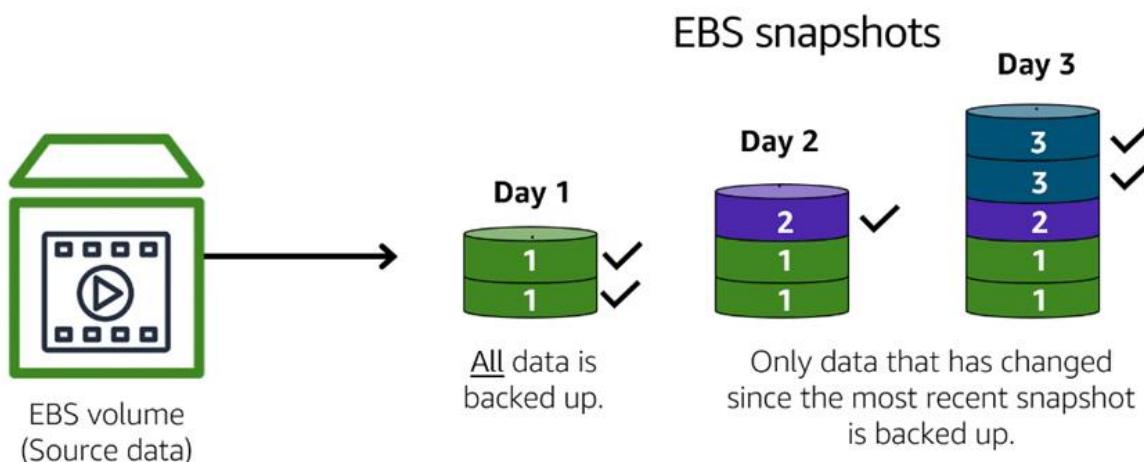
EBS is a service that provides block-level storage volumes that you can use with Amazon EC2 instances. If you stop or terminate an Amazon EC2 instance, all the data on the attached EBS volume remains available.

To create an EBS volume, you define the configuration (such as volume size and type) and provision it. After you create an EBS volume, it can attach to an Amazon EC2 instance.

Because EBS volumes are for data that needs to persist, it's important to back up the data. You can take incremental backups of EBS volumes by creating Amazon EBS snapshots.

- We cannot launch any instance without any disk, by default there is a disk attached for the root or primary partition.
- There are various types of volumes available.
- A IMP thing regarding volume is, volume can only be attached to a machine which is in the same AZ.

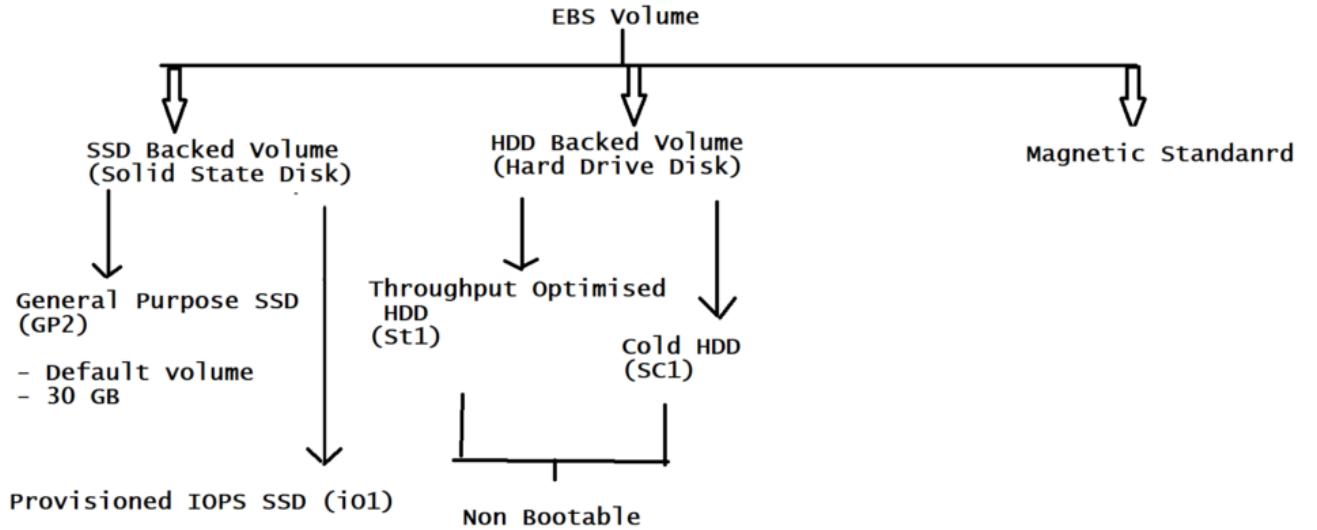
EBS Snapshots:



- An EBS snapshot is an incremental backup. This means that the first backup taken of a volume copies all the data. For subsequent backups, only the blocks of data that have changed since the most recent snapshot are saved.
- These snapshots are also stored in Amazon S3.
- Incremental backups are different from full backups, in which all the data in a storage volume copies each time a backup occurs. The full backup includes data that has not changed since the most recent backup.
- These snapshots are the exact replica of the EBS Volume.
- Take up less space as stored incrementally.
- Low cost as it is stored in S3 which is cheaper storage in AWS.
- Full Data recovery guaranteed.

- The 1st snapshot will have the written blocks and the next snapshots will have the change blocks, hence we don't need to have the change blocks. Hence we don't need to have multiple full copies of the snapshots.
- Deleting the snapshot will only remove the data which is exclusive to the snapshot.

Types of volumes-



- **General purpose SSD (gp2) :-**
 1. Basic speed of disk.
 2. GP2 is default EBS Volume type for Amazon EC2 instance.
 3. Backed by SSDs.
 4. General Purpose balances both, Price and performance.
 5. Ratio of 3IOPS/GB with upto 10,000 IOPS.
 6. Boot Volume having low latency.
 7. Volume size – 1GB to 16 TB
 8. Price : \$0.10/GB/month
- **Provisioned IOPS (IO1) :-**
 1. The IOPS offered is more. Here IOPS means input output per second. This means the input output transaction speed of the disk is very high
 2. These volumes are ideal for both Iops intensive and throughput intensive workloads that requires extremely low Latency, or for mission critical applications.
 3. Designed for I/p-o/p intensive applications such as large relational or NoSQL Database.
 4. Use if you need more than 10,000 IOPS.
 5. Can provision up to 32000 IOPS per volume.
 6. Volume Size 4 GB to 16 TB
 7. Price: \$0.125 / Gb/ Month

Practical Eg:- Database Disk.

- **Throughput Optimized (St1):-**
 1. Number of Bytes/bits written in a particular second is called as throughput, so the disks are more optimized for throughput. (40MB/second written on disk)
 2. It is backed by HDD and ideal for frequently accessed throughput intensive large datasets.
 3. These volumes deliver performance in terms of throughput, measured in Mbps.
 4. Big Data, Data Warehouse, log processing
 5. It is Non Bootable.
 6. Can be provisioned up to 500 IOPS per volume
 7. Volume Size 500 GB to 16 TB
 8. Price: \$0.045 / Gb/ Month
- **Cold HDD–**
 1. Fixed Speed offered. Practically used for backup purpose.
 2. This volume is also backed by HDD, and provides lowest cost per GB, of all EBS Volume types.
 3. Lowest Cost storage for infrequent access Workloads.
 4. Used in file servers.
 5. It is Non Bootable.
 6. Can be provisioned upto 250 IOPS per volume.
 7. Volume Size 500 GB to 16 TB
 8. Price: \$0.025 / Gb/ Month
- **Magnetic Standard –**
 1. This is very old generation, used for back up purpose.
 2. Lowest cost per GB of all Bootable EBS Volume types.
 3. Ideal for workloads where data is accessed infrequently.
 4. Volume Size: 1 GB to 1 TB
 5. Price: \$0.05 / Gb/ Month
 6. Can be provisioned upto 40 – 200 IOPS per volume.

Advantages-

- Low Latency
- High Performance
- It's a bootable drive
- Highly redundant (data recovery is fast)

6. Elastic Load Balancing

Elastic Load Balancing is the AWS service that automatically distributes incoming application traffic across multiple resources, such as Amazon EC2 instances.

A load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group. This means that as you add or remove Amazon EC2 instances in response to the amount of incoming traffic, these requests route to the load balancer first. Then, the requests spread across multiple resources that will handle them. For example, if you have multiple Amazon EC2 instances, Elastic Load Balancing distributes the workload across the multiple instances so that no single instance has to carry the bulk of it.

Although Elastic Load Balancing and Amazon EC2 Auto Scaling are separate services, they work together to help ensure that applications running in Amazon EC2 can provide high performance and availability.

- A load balancer is a device that distributes network or application traffic across a number of servers.
- ELB distributes incoming traffic amongst various EC2 instances or available servers.
- ELB can distribute the load in between single AZ or multiple AZ
- ELB helps increase fault tolerance and makes sure there is high availability of the application 24*7.

How does it work?

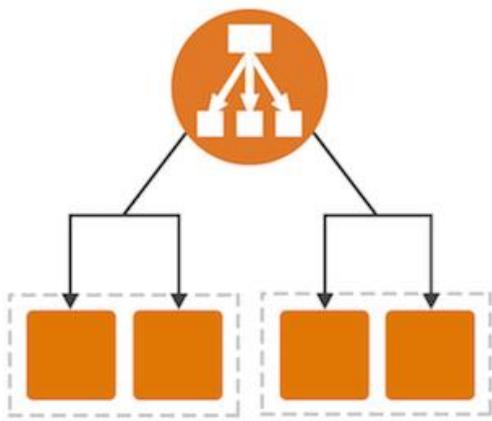
- Load balancer checks on which server the traffic is less and accordingly forwards the request to the corresponding server.
- Incase any of the server is down it will move the request to any of the healthy server which is configured.
- Here we have to note that the load balancer will also do health checks on a regular time intervals and forward requests only to the healthy server.
- Usually load balancing is done together with auto-scaling, if any server is faulty then it gets replaced by a healthy server.

Types of load balancers

- Application load balancer
- Network load balancer
- Classic load balancer
- Gateway load balancer

Application load balancer-

- Application load balancer is used for mobile applications or web applications.
- ALB operates at the application layer of the OSI model, i.e. layer 7
- ALB is able to inspect application level content and route traffic based on HTTP and HTTPS protocol.
- Also ALB can route based on HTTP headers. For eg we want to access application which has a header /foo or /bar its possible



Application Load Balancer components-

- Load Balancers
- Listeners
- Target Groups

- **Listener –**

A listener is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections, and a protocol and a port for back-end (load balancer to back-end instance) connections.

It listens to the incoming requests and forward requests accordingly.

- **Target Group –**

This is nothing but a cluster of EC2 instances. The ELB will only forward the traffic to EC2 instances which are part of target group.

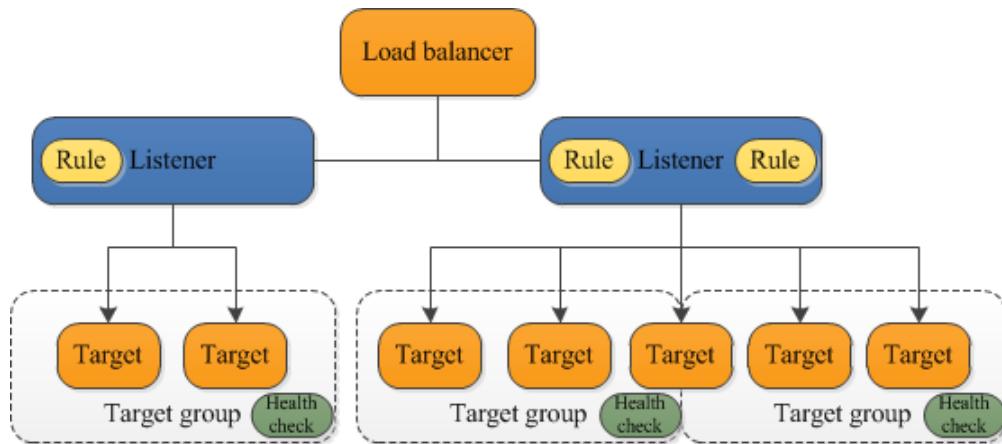
- **Target –**

This is nothing but our individual EC2 instance, there is where we are going to target our traffic.

- **Health checks –**

This is done to check whether instance is healthy or not. ELB does some prior health checks before registering targets and forwarding traffic to it.

The following diagram illustrates the basic components. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



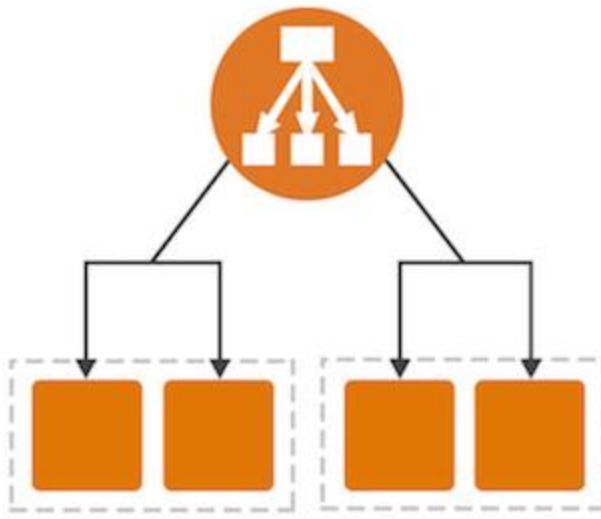
- A listener checks for connection requests from clients, using the protocol and port that you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.
- Each target group routes requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify. You can register a target with multiple target groups. You can configure health checks on a per target group basis.

Benefits of migrating from a Classic Load Balancer-

- Support for path condition. You can configure rules for your listener that forward requests based on the URL in the request.
- Support for Host conditions. You can configure rules for your listener that forward requests based on the host field in the HTTP header.
- Support for registering targets by IP address, including targets outside the VPC for the load balancer.
- Support for redirecting requests from one URL to another.
- Support for registering Lambda functions as targets.
- Improved load balancer performance.

Network load balancer

- Network load balancer works on Layer 4 i.e is transport layer of OSI model.
- Basically network load balancer provides low latency (response time) and can manage heavy loads at a time.
- Network LB works on TCP/UDP/TLS protocols.



Difference between ALB & NLB-

- NLB just forward requests whereas ALB examines the contents of the HTTP request header to determine where to route the request.
- When you need to seamlessly support spiky or high-volume inbound TCP requests we use the network load balancer.
- ALBs are typically used for web applications. If you have a microservices architecture, etc

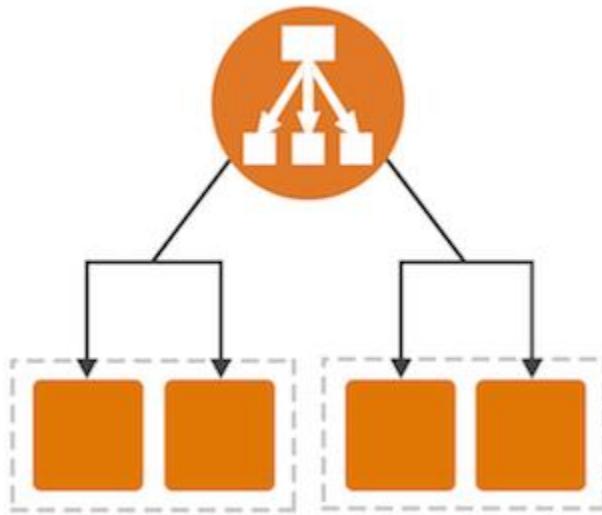
Difference between HTTP & TCP-

- HTTP typically uses port 80 – this is the port that the server “listens to” or expects to receive from a Web client. TCP doesn’t require a port to do its job.
- HTTP is faster in comparison to TCP as it operates at a higher speed and performs the process immediately. TCP is relatively slower.
- TCP tells the destination computer which application should receive data and ensures the proper delivery of said data, whereas HTTP is used to search and find the desired documents on the Internet.
- TCP contains information about what data has or has not been received yet, while HTTP contains specific instructions on how to read and process the data once it’s received.
- TCP manages the data stream, whereas HTTP describes what the data in the stream contains.
- TCP operates as a three-way communication protocol, while HTTP is a single-way protocol.

Classic Load Balancer –

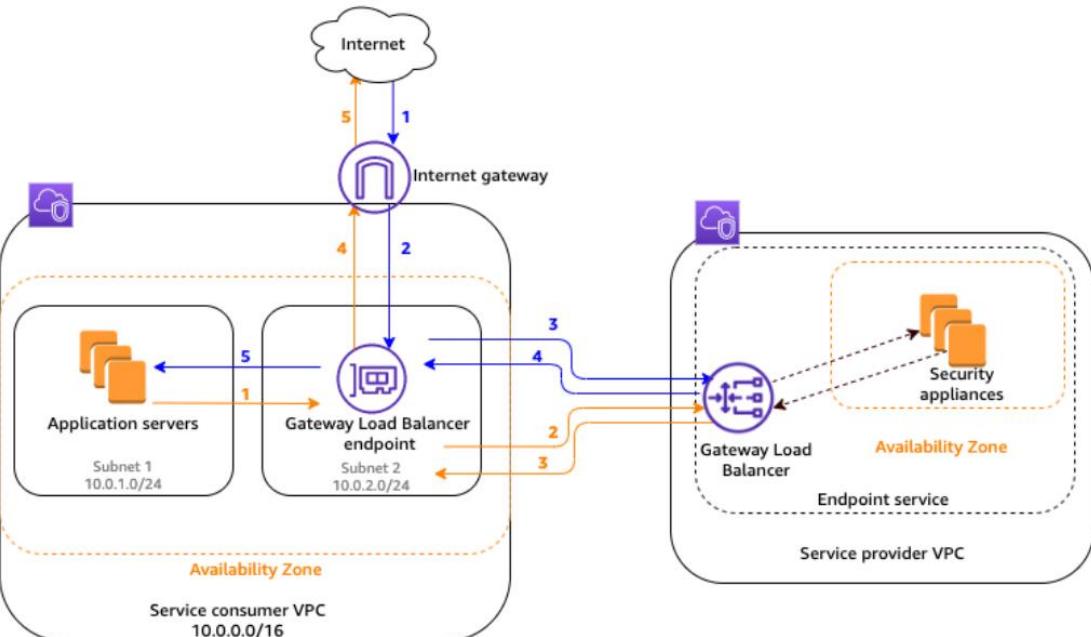
- A Classic Load Balancer makes routing decisions at either the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS).

- Classic Load Balancers currently require a fixed relationship between the load balancer port and the container instance.
- Classic load balancer is going to be discontinued by AWS.



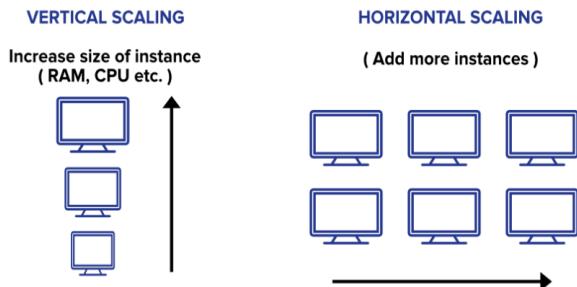
Gateway Load Balancers –

- Gateway Load Balancers allow you to deploy, scale, and manage virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems.
- It combines a transparent network gateway (that is, a single entry and exit point for all traffic) and distributes traffic while scaling your virtual appliances with the demand.
- A Gateway Load Balancer operates at the third layer of the Open Systems Interconnection (OSI) model, the network layer.
- It listens for all IP packets across all ports and forwards traffic to the target group that's specified in the listener rule.
- It maintains stickiness of flows to a specific target appliance using 5-tuple (for TCP/UDP flows) or 3-tuple (for non-TCP/UDP flows).
- A Gateway Load Balancer endpoint is a VPC endpoint that provides private connectivity between virtual appliances in the service provider VPC and application servers in the service consumer VPC.



7. Autoscaling

What is Scaling?



- In cloud computing, scaling is the process of adding or removing compute, storage, and network services to meet the demands a workload makes for resources in order to maintain availability and performance as utilization increases.
- In simple words, as per our requirement increasing the capacity of anything is called scalability.
- Auto-scaling is nothing but automatic scaling of our capacity as per the requirement.
- Here basically the concept is whenever the load on the server increases/decreases auto-scaling functionality helps to launch as well as terminate the EC2 instances.
- So here if you see scaling is done in both ways, we are increasing as well as decreasing the horizontal capacity.
- Scale OUT/UP means increasing the number of ec2 instances
- Scale IN/DOWN means decreasing the number of ec2 instances
- Auto-scaling is region specific, we cannot do auto-scaling between 2 regions.
- Auto Scaling (AS) happens in Availability zones and we can distribute the instances evenly in Availability zones.
- Here we create group of EC2 instances which can scale up and down as per the conditions we set.
- Auto scaling ensures that we have right number of EC2 instances to suffice our needs all the time, also autoscaling helps us reduce the cost by cutting down the number of instances when not needed.
- No extra cost is needed for Auto-Scaling, only cost of EC2 instances.
- Auto scaling makes sure that all the launches instances are balanced in between the set availability zones.
- Just incase these are not balanced or we later add the instances, auto scaling tries to balance it out automatically
- Here the criteria is we can only add a additional EC2 instance to the ASG group only if ec2 instance is in running state, it's in the same AZ which is configured with ASG, and if the ASG has reached its max limit then we cannot add as the request fails.

1. Vertical Scaling

- In simple terms upgrading the capacity of a single machine or moving to a new machine with more power is called vertical scaling. You can add more powers to your machine by adding better processors, increasing RAM, or other power increasing adjustments. Vertical scaling can be easily achieved by switching from small to bigger machines but remember that this involves downtime. You can enhance the capability of your server without manipulating your code.
- This approach is also referred to as the ‘scale-up’ approach.
- It doesn’t require any partitioning of data and all the traffic resides on a single node with more capacity.
- Easy implementation.
- Less administrative efforts as you need to manage just one system.
- Application compatibility is maintained.
- Mostly used in small and mid-sized companies.
- MySQL and Amazon RDS is a good example of vertical scaling.

Drawbacks-

- Limited Scaling.
- Limited potential for improving network I/O or disk I/O.
- Replacing the server will require downtime in this approach.
- Greater risk of outages and hardware failures.
- Finite scope of upgradeability in the future.
- Implementation cost is expensive.

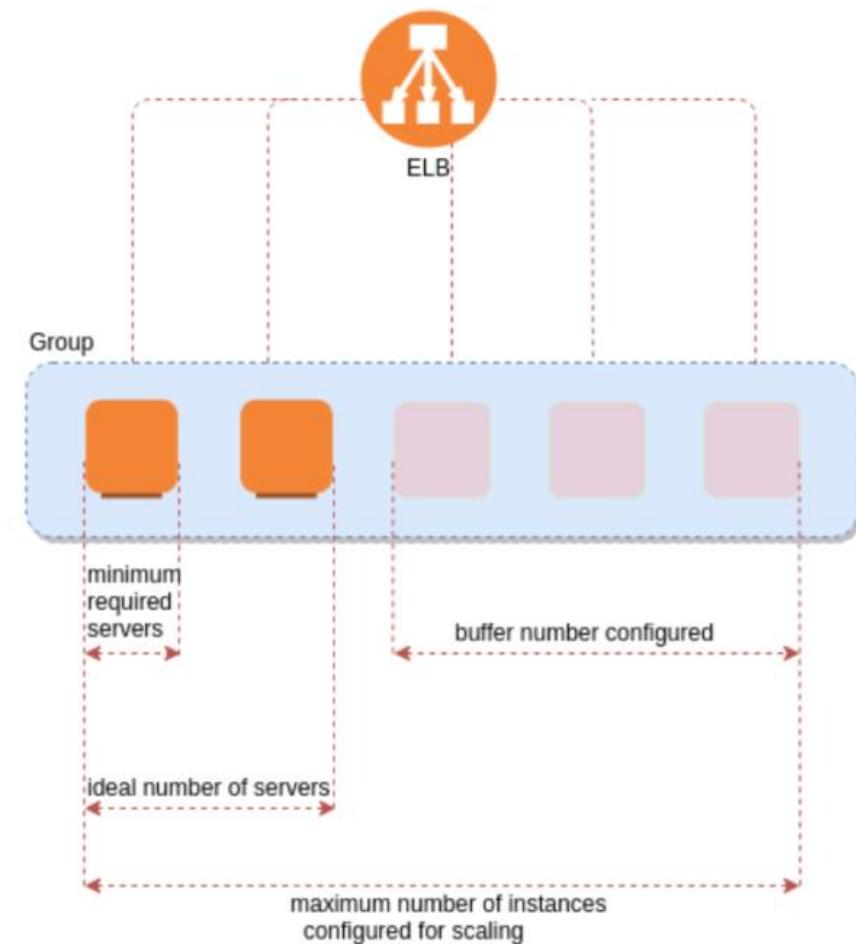
2. Horizontal Scaling-

- This approach is the best solution for projects which have requirements for high availability or failover. In horizontal scaling, we enhance the performance of the server by adding more machines to the network, sharing the processing and memory workload across multiple devices. We simply add more instances of the server to the existing pool of servers and distribute the load among these servers. In this approach, there is no need to change the capacity of the server or replace the server. Also, like vertical scaling, there is no downtime while adding more servers to the network. Most organizations choose this approach because it includes increasing I/O concurrency, reducing the load on existing nodes, and increasing disk capacity.
- This approach is also referred to as the ‘scale-out’ approach.
- Horizontal scalability can be achieved with the help of a distributed file system, clustering, and load-balancing.
- Traffic can be managed effectively.
- Easier to run fault-tolerance.
- Easy to upgrade
- Instant and continuous availability.
- Easy to size and resize properly to your needs.
- Implementation cost is less expensive compared to scaling-up
- Google with its Gmail and YouTube, Yahoo, Facebook, eBay, Amazon, etc. are heavily utilizing horizontal scaling.
- Cassandra and MongoDB is a good example of horizontal scaling.

Drawbacks-

- Complicated architectural design
- High licensing fees
- High utility costs such (cooling and electricity)
- The requirement of extra networking equipment such as routers and switches.

ELB with Autoscaling



- We can attach one or more ELB to the ASG.
- The ELB must be in the same region.
- Once we configure this, any EC2 instance existing or added by the ASG will be automatically registered with the ASG defined ELB.
- While enabling ELB with ASG we always need to enable ELB health check as well, by default ASG uses EC2 health check option.

Healthcheck-

- Auto scaling by default classifies its EC2 instances status healthy or unhealthy with the help of EC2 status check.
- By default the health check grace period is 300 sec, this means once the instance is launched after that till 300 seconds there will be no health check, it's not recommended to keep this value as 0 as, if this is kept zero once the instance is launched immediately it will start health check and we all know that it takes some time for the instance to get into “initialized state i.e 2/2 status”
- Until the grace period timer expires any unhealthy status reported won't be acted upon by the ASG.

- Unlike rebalancing, here the termination of instances will happen 1st in case if its unhealthy, then the ASG attempts to launch new instances to replace the terminated one.
- The Elastic IP and EBS volume (additional) gets detached from the terminated instances, then we may need to manually attach them to new instances.
- Basic monitoring is enabled by default and is free of cost (300 sec).
- We can also make instances in stand by state manually in case we want to do any patch activity, the ASG will not do any health check on these machines, and there won't be any load forwarded on this machine for that time period.

Auto Scaling Components –

- **Launch Configuration:-** We have to configure which type of instance we want, we can select the AMI, key pair, security group, instance type etc. Here once the Launch config is created we cannot edit the same. We can only delete or copy it.
- **Auto-Scaling Group:-** We can select the Group Name, Group Size (Min instances, Max instances, desired instances), also the health check period which is 300 sec by default.
- Scaling policies

Warm up time of an instance:-

Instance warm-up defines the number of seconds it takes for a newly launched instance to warm up.

This prevents the ASG from adding more instances than needed. Before this launch, the instance warm-up time was set to a default value of 300 seconds.

Warm-up value for Instances allows you to control the time until a newly launched instance can contribute to the Cloud-Watch metrics, so when warm-up time has expired, an instance is considered to be a part Auto Scaling group and will receive traffic.

Cool down period:-

A cooldown period is a period of time after each scaling action is complete. During the cooldown period, scaling actions triggered by alarms will be denied.

The cooldown period is a configurable setting for your Auto Scaling group that helps to ensure that it doesn't launch or terminate additional instances before the previous scaling activity takes effect

The amount of time to wait for a previous scaling activity to take effect is called the cooldown period.

Auto Scaling Policies

- Manual :- keep scaling policies same (as it is)

Here the values of min, max and desired will be the same and If I have to change this value then I need to do it manually.

- Dynamic:-
 1. Target tracking policy
 2. Simple scaling policy
 3. Step scaling policy
 4. Scheduled scaling policy
 5. Predictive scaling policy

Target Tracking Policy

- It helps to auto scale based on the metrics like Average CPU Utilization, Load balancer request per target, and so on. Simply stated it scales up and down the resources to keep the metric at a fixed value.
- For example, if the configured metric is Average CPU Utilization and the value is 60%, the Target Tracking Policy will launch more instances if the Average CPU Utilization goes beyond 60%. It will automatically scale down when the usage decreases. Target Tracking Policy works using a set of CloudWatch alarms which are automatically set when the policy is configured.
- One metric value set i.e 60 %
- AWS recommends to use Target Tracking policy for a metric like Average CPU utilization hence mostly this is used.

Simple Scaling

- Simple scaling relies on a metric as a basis for scaling.
- For example, you can set a Cloud Watch alarm to have a CPU Utilization threshold of 80%, and then set the scaling policy to add 20% more capacity to your Auto Scaling group by launching new instances.
- Accordingly, you can also set a Cloud Watch alarm to have a CPU utilization threshold of 30%. When the threshold is met, the Auto Scaling group will remove 20% of its capacity by terminating EC2 instances.

Step Scaling

- Step Scaling further improves the features of simple scaling.
- Step scaling applies “step adjustments” which means you can set multiple actions to vary the scaling depending on the size of the alarm breach.

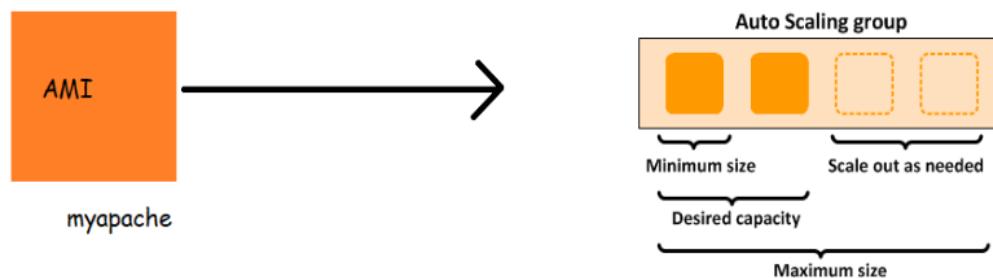
More Policies-

- Scheduled scaling – Set a schedule for eg weekend load is high scale up etc.
- Predictive scaling – based on history and uses AI

AWS EC2 AutoScaling –

- Now we would want to increase/decrease number of ec2 instances in auto scaling group depending on some factors.
- So let's try to increase ec2 instances when the load on CPU increases for this
 - When CPU utilization > 80 % for 5 minutes lets increase one ec2 instance
 - When CPU utilization < 40 % for 5 minutes lets decrease one ec2 instance
 - Minimum ec2 instances => 1
 - Max ec2 instances => 5

Fixed Scaling => 2



EC2 Auto Scaling:

- When AWS introduced the EC2 Auto Scaling service in 2009, it pioneered configurable scaling. As its name indicates, it focuses on the Amazon Elastic Compute Cloud (EC2) service, and it enables users to automatically launch and terminate EC2 instances based on configurable parameters.
- The most common use case in EC2 Auto Scaling is to configure CloudWatch alarms to launch new EC2 instances when a specific metric exceeds a threshold. For example, a developer could configure Auto Scaling to launch two EC2 instances when CPU utilization is greater than 50% for five consecutive minutes. Users also configure CloudWatch alarms to decrease the number of EC2 instances, for example, when CPU utilization falls to a value considered low usage.
- For certain applications, developers can also configure EC2 Auto Scaling to launch and terminate instances based on schedules. This is useful for known periods of low utilization such as nights or weekends.

AWS Auto Scaling

- AWS Auto Scaling, meanwhile, offers a centralized place to manage configurations for a wider range of scalable resources, such as EC2 instances, Amazon Elastic Container Service (ECS), Amazon DynamoDB tables or Amazon Relational Database Aurora read replicas.
- With AWS Auto Scaling, users can keep EC2 Auto Scaling groups within a configurable range of metrics. Developers can configure dynamic DynamoDB read/write capacity units for a specific table, also based on utilization. ECS services can be configured to launch or terminate ECS tasks based on CloudWatch metrics. The same applies to RDS read replicas; AWS Auto Scaling can add or terminate RDS read replicas based on utilization.
- AWS Auto Scaling introduced the concept of scaling plans, which use scaling strategies in order to manage resource utilization. Application owners can select a target utilization, such as CPU utilization at 50%, and AWS Auto Scaling will add or remove capacity to achieve that target.

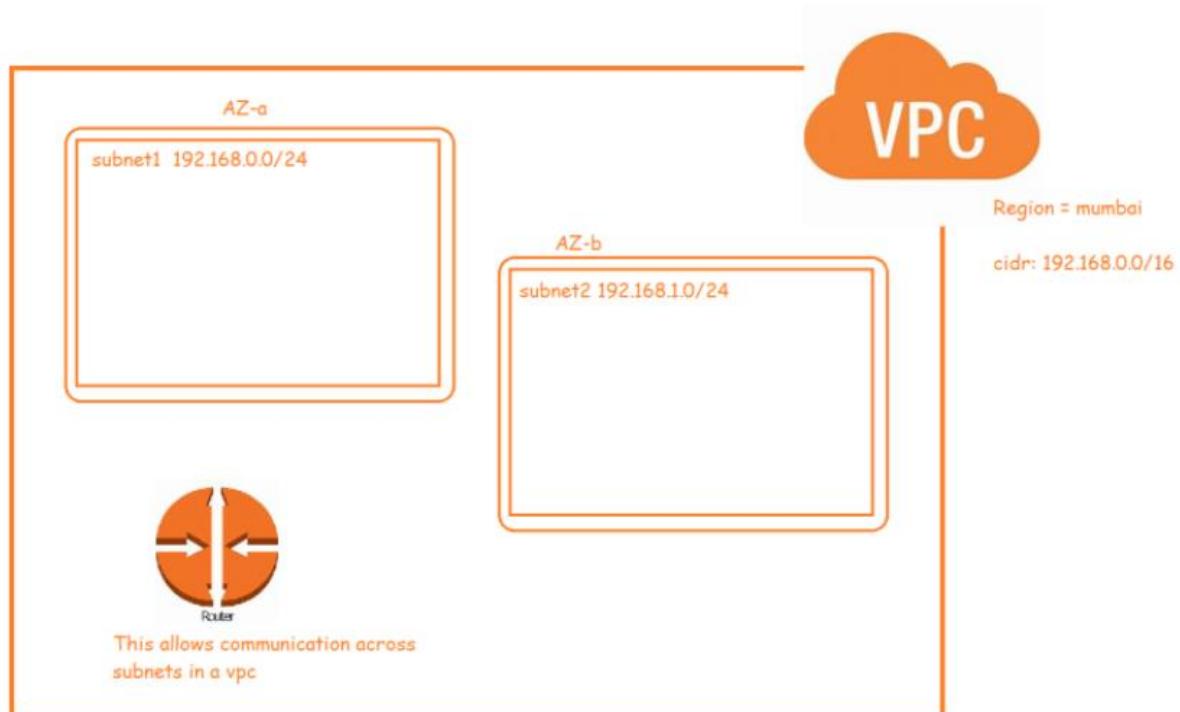
Key differences in Amazon EC2 Auto Scaling vs. AWS Auto Scaling

- Overall, AWS Auto Scaling is a simplified option to scale multiple Amazon cloud services based on utilization targets. Amazon EC2 Auto Scaling focuses strictly on EC2 instances to enable developers to configure more detailed scaling behaviours.
- Another important distinction is that AWS Auto Scaling focuses on target utilization — for example, “Add a number of EC2 instances when a particular metric exceeds a threshold” — rather than let developers configure specific actions. Meanwhile, EC2 Auto Scaling relies on predictive scaling, which uses machine learning to determine the right amount of resource capacity necessary to maintain a target utilization for EC2 instances.
- While EC2 Auto Scaling provides more flexibility, AWS Auto Scaling delivers simplicity. The choice will come down to which features and capabilities are most relevant to the IT team and developers planning to scale the cloud environment.

8. Virtual Private Cloud

- VPC (Virtual Private Cloud) is a logical data center or virtual data center in Cloud. Its provide an isolated section to host your machine.
- VPC is a collection of the region, IG, route table, ACL, security group, subnet, instances. You can add all the security matrix-like security groups, etc.
- VPC provide us a completely separate environment where we can place our machine in our own way.
- It helps you to create a virtual isolated environment in the same cloud.
- Multiple virtual isolated Environments are also possible.

- Amazon Virtual Private Cloud is a commercial cloud computing service that provides users a virtual private cloud, “Provisioning a logically isolated section of Amazon Web Services Cloud”.
- AWS Virtual Private Cloud (VPC) gives you complete control over your virtual networking environment, including resource placement, connectivity, and security.
- We cannot establish the connection between two EC2 instances in two different VPC’s. It is possible only if they have public IP address
- In many case we would want connectivity b/w EC2 instances in different VPC’s but privately.
- AWS supports peering connection.
- When we create VPC in AWS , the allowed block size is between /16 and /28
- In the case of AWS VPC we cannot use 5 IP addresses in every subnet
 - All 0’s is network IP (x.x.x.0)
 - All 1’s is broad cast IP (x.x.x.255)
 - x.x.x.1 (Reserved by AWS for VPC Router)
 - x.x.x.2 Reserved by AWS for IP address of the DNS Server
 - x.x.x.3 Reserved for future usage.
- To create a network in AWS, we use a service called as VPC (Virtual Private Cloud)
 - Network is created at a region level
 - Subnets are created at AZ level



Why do we need VPC ?

- You need a VPC: a virtual private network that keeps your servers safe from the ravages of the public internet.
- Multiple Connectivity Options:-
- Your AWS VPC can be connected to a variety of resources, such as the internet, your on-premise data center, other VPCs in your AWS account, or VPCs in other AWS accounts; once connected, you can make your resources accessible or inaccessible in your VPC from outside of your VPC based on your requirement.
- Secure
- Simple
- All the scalability and reliability of AWS is available.

Use Cases –

- Host a public facing website.
- Host a multi-tier application, such as it will have web-layer, LB layer, DB layer etc.
- Hosting of scalable web applications in your cloud.
- Manage multiple projects, create isolated networks for those projects.

Some IMP terms –

- VPC:- this is nothing but a full isolated network.
- Default VPC:- in every region there is a small default VPC created by amazon, this default VPC has all the standard routing rules set by amazon.
- Non Default VPC
- Subnets
- Internet gateway
- Network ACLs (Network Access Control List)
- Security Groups
- NAT Gateway/Instance.
- Routing Tables.
- IPs, DNS etc
- VPC peering

Default limits –

- We can have only up to 5 non-default AWS VPC's per region
- You can create up to 200 subnets per VPC
- We can create up to 200 network ACL per amazon VPC
- We can have up to 5 Elastic IP addresses per AWS account per region.
- Count of IPv4 CIDR blocks / VPC – 5
- Count of IPv6 CIDR blocks / VPC – 1
- Count of Internet gateways / Region – 5
- Count of NAT gateways / Availability Zone – 5

Subnets –

- A subnet, or subnetwork, is a network inside a network.
- When we break down a network into smaller networks is called as subnet or subnetwork.
- This breaking down of network is based on IP. (CIDR)
- Basically there are 2 types of subnets
 1. Public Subnet
 2. Private Subnet

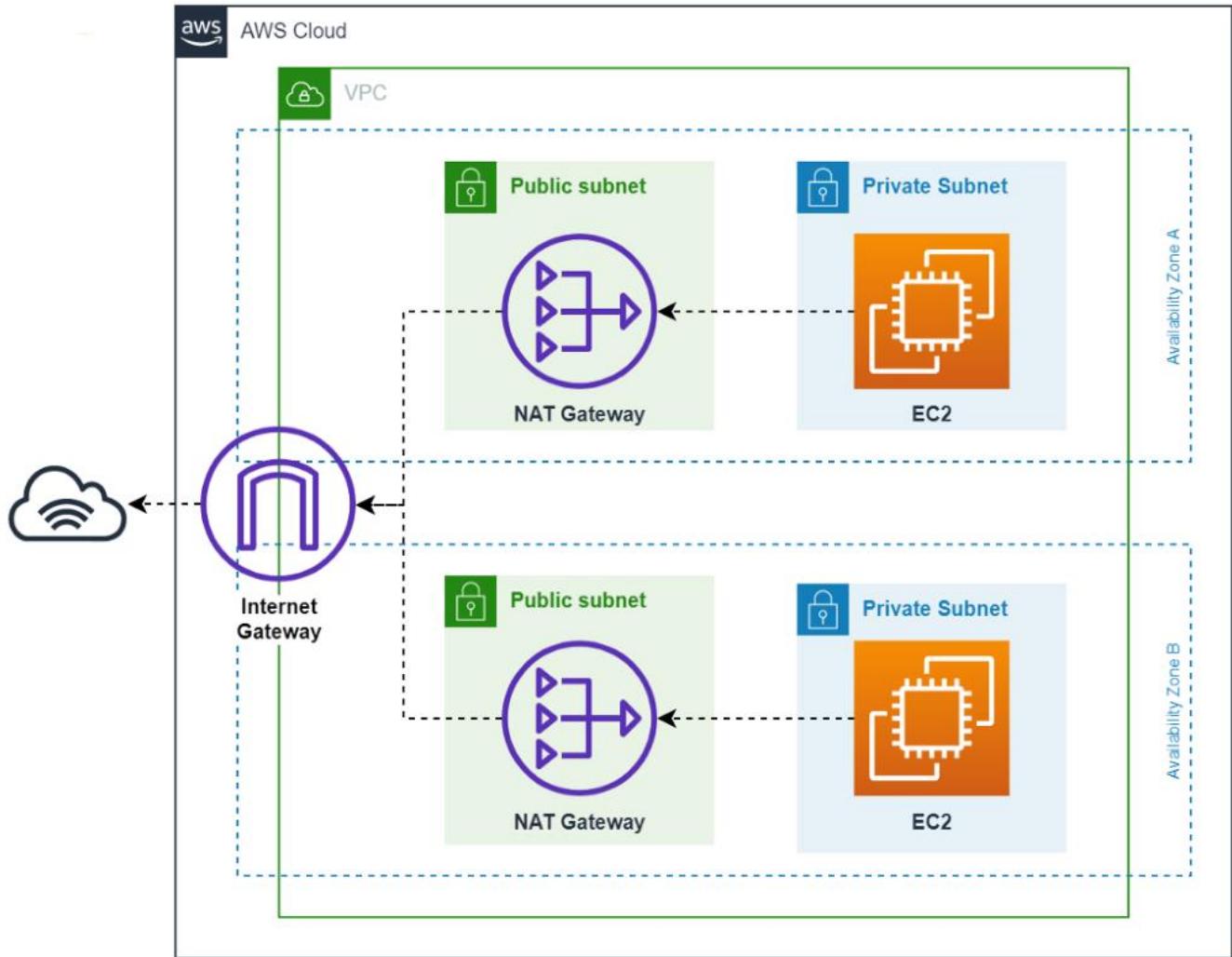
Route table –

- A route table contains a set of rules, called routes, that are used to determine where network traffic from your subnet or gateway is directed.
- To put it simply, a route table tells network packets which way they need to go to get to their destination.
- There are 2 types of routing Public routing and Private routing
- Public routing is where we can route to the internet with the help of internet gateway.
- Private routing means we can route between private subnets or internal network only.

Internet Gateway –

- We created VPC because we want to run our resources in it.
- When we run our resources in VPC, we might need to access our resources (VM's) from internet
- AWS VPC created by us is private in nature by default and cannot be accessed from internet.
- Internet Gateway (IGW) allows instances with public IPs to access the internet.
- To enable access to our VPC from/to internet, we need to Create an internet gateway.
- An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet.
- You can associate exactly one Internet Gateway with a VPC.
- If a VPC does not have an Internet Gateway, then the resources in the VPC cannot be accessed from the Internet.
- A public subnet is a subnet that's associated with a route table that has a route to an internet gateway.
- In simple terms this means, when a subnet is connected to internet its called as public subnet
- An internet gateway has a Public IP attached to it.
- There's no additional charge for having an internet gateway in your account.

Nat Gateway vs Internet Gateway



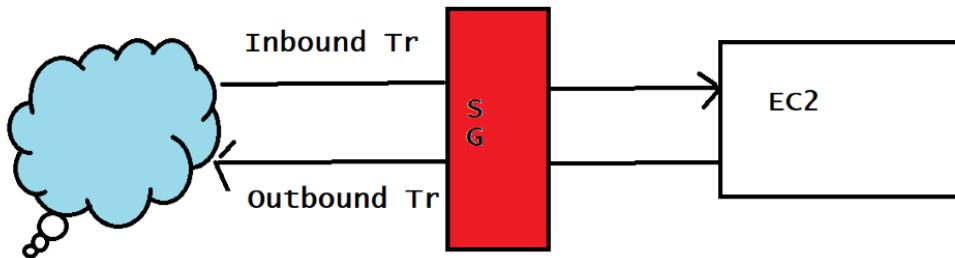
Nat Gateway –

- NAT is nothing but network address translation.
- NAT Gateway (NGW) allows instances with no public IPs to access the internet.
- It only works one way. Through NAT we can go from Private to Public but a public entity cannot access the private n/w.
- One way communication
- If we want two way NAT then we have to setup reverse NATing
- In cloud computing we do not go for reverse NATing which is a standard
- NAT gateways are supported for IPv4 or IPv6 traffic.
- NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone.

- You can associate exactly one Elastic IP address with a public NAT gateway.
- You are charged for each hour that your NAT gateway is available and each Gigabyte of data that it processes.

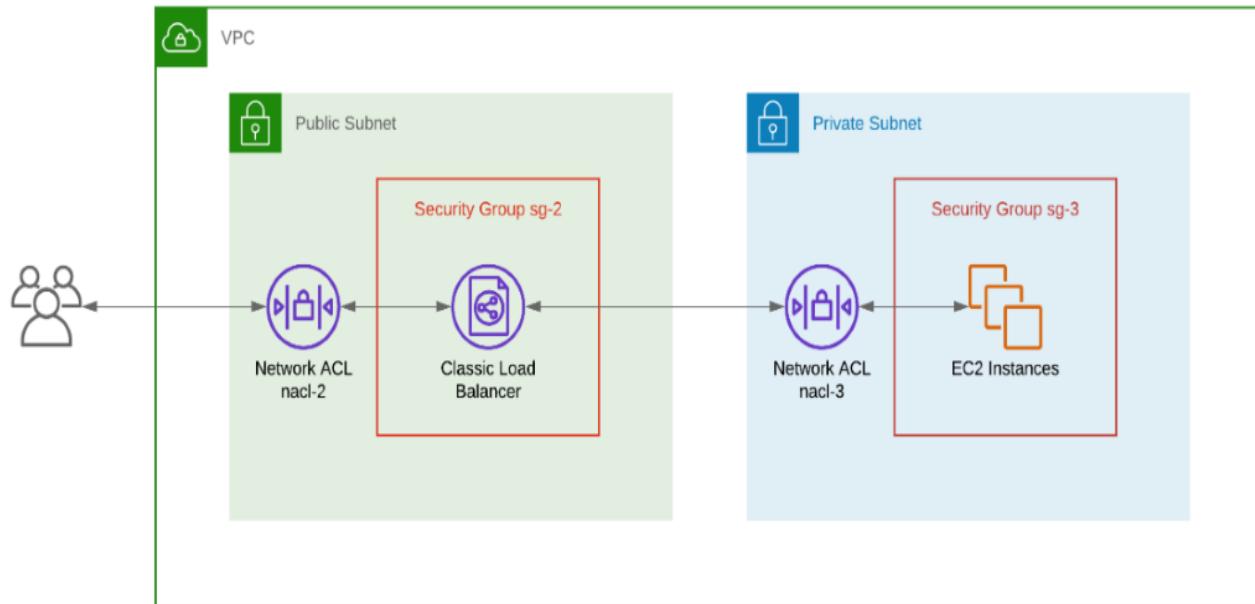
Security groups –

- An AWS security group acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic. Both inbound and outbound rules control the flow of traffic to and from your instance, respectively.
- AWS Security Groups help you secure your cloud environment by controlling how traffic will be allowed into your EC2 machines. With Security Groups, you can ensure that all the traffic that flows at the instance level is only through your established ports and protocols.



Network ACL (access control list) –

- This acts as a firewall for associated subnets.
- It controls both incoming and outgoing traffic at the subnet level.
- It's the 1st firewall at the network level.
- A network access control list (ACL) is a layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets.



- A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets.

The following are the parts of a network ACL rule:

- **Rule number** – Rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it's applied regardless of any higher-numbered rule that might contradict it.
- **Type** – The type of traffic; for example, SSH. You can also specify all traffic or a custom range.
- **Protocol** – You can specify any protocol that has a standard protocol number.
- **Port range** – The listening port or port range for the traffic. For example, 80 for HTTP traffic.
- **Source** – [Inbound rules only] The source of the traffic (CIDR range).
- **Destination** – [Outbound rules only] The destination for the traffic (CIDR range).
- **Allow/Deny** – Whether to allow or deny the specified traffic.

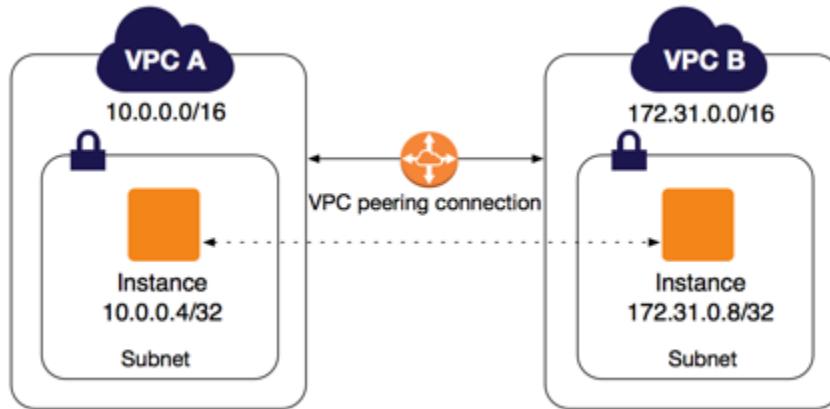
Security Groups VS Network ACL's-

Security Group	Network Access Control List
In security group, we operate at instance level.	In network ACL, we operate sub net level.
It support only allow rules.	It support allow rules and deny rules.
It is stateful, Return traffic is automatically allowed regardless of any rules.	It is stateless, it return traffic must be allowed explicitly.

We cannot block specific IP address using SGs.	We can block specific IP Address using NACL.
All rules are evaluated before deciding to permit traffic.	Rules are processed in number order when deciding whether allow traffic.
It starts with instance launch configuration.	In which we assigned to subnet for all instances.
It applies when someone specifies security group when launching the instance and it associates with security group.	They do not depend on user it automatically applies to all instances with subnet.

VPC Peering-

- A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.
- VPC peering can be created b/w any two vpcs where cidr's donot collide.
- Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.
- The VPCs can be in different regions (also known as an inter-region VPC peering connection).

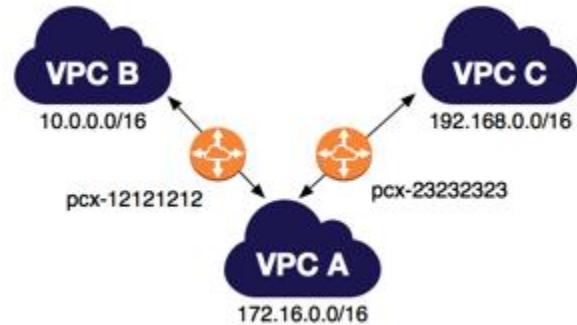


- AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck.

Multiple VPC peering connections-

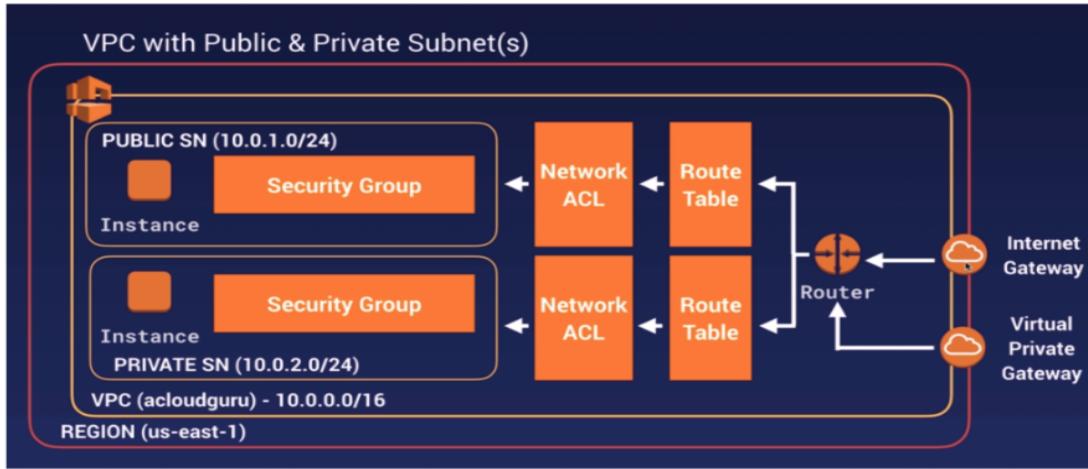
- A VPC peering connection is a one to one relationship between two VPCs.

- You can create multiple VPC peering connections for each VPC that you own, but transitive peering relationships are not supported.
- You do not have any peering relationship with VPCs that your VPC is not directly peered with.



VPC connection-

- The owner of the requester VPC sends a request to the owner of the accepter VPC to create the VPC peering connection.
- The accepter VPC can be owned by you, or another AWS account, and cannot have a CIDR block that overlaps with the requester VPC's CIDR block's.
- The owner of the accepter VPC accepts the VPC peering connection request to activate the VPC peering connection.
- To enable the flow of traffic between the VPCs using private IP addresses, the owner of each VPC in the VPC peering connection must manually add a route to one or more of their VPC route tables that points to the IP address range of the other VPC (the peer VPC).



9. Elastic File System

- EFS provides simple, elastic file system for use with Cloud Services and on-premises solutions
- EFS supports NFS(Network File system) version 4 protocol.
- With EFS you pay only for the storage used by your file system
- Create an EFS in a region with selected zones.
- EFS is a network file system, so while creating EFS, We need to provide the subnet ids and security group
- Amazon Elastic File System (Amazon EFS) is a simple, server-less, set-and-forget, elastic file system.
- EFS is a user-based encryption control technique that enables users to control who can read the files on their system. The typical method of using EFS is to perform encryption at the folder level. This ensures that all files added to the encrypted folder are automatically encrypted.
- There is no minimum fee or setup charge.
- You pay only for the storage you use, for read and write access to data stored in Infrequent Access storage classes, and for any provisioned throughput.
- EBS is a high-performance per-instance block storage system designed to act as storage for a single EC2 instance (most of the time) EFS is a highly scalable file storage system designed to provide flexible storage for multiple EC2 instances.
- You can use an EFS file system as a common data source for workloads and applications running on multiple instances.
- EFS provides a file system interface and file system semantics to AWS EC2 instances. When EFS is attached to EC2 instances, it acts just like a local file system but EFS is a shared file system which means you can mount the same file system across multiple EC2 instances with low latencies.

Attributes of EFS:

- Fully managed: We don't need to maintain any hardware or software
- File System access semantics: We get what we expect from a regular file system including read-after-write consistency locking.
- File System Interface: It exposes file system interface that works with standard os APIs. EFS appears to be like any other file system to your OS
- Shared Storage:
- Elastic and scalable: EFS elastically grows to petabyte scale. We don't have to specify a provisioned size up front, We just create a file system, and it grows and shrinks automatically as we add/remove data
- Performance: It is built for performance across wide variety of workloads. It provides consistent, low latencies, high throughput and high IOPS
- High Available & Durable: The data in EFS is automatically replicated across AZs with in a region.
- EFS also works with NFS protocol, Using Direct connect and VPC, we can also mount EFS on your on-premises server via the NFS 4.1 protocol. This a great use case for transferring a large number of data from the servers running on-premise to AWS cloud
- Performance Mode of EFS
- General Purpose Mode:
- This is default mode for EFS
- This is optimized for latency-sensitive applications & general purpose file-based workloads
- This mode is best option for majority of the use cases
- Max I/O mode
- This is optimized for large-scale and data-heavy applications, where tens, hundreds or thousands of EC2 instances are accessing the file system
- This scales to higher level aggregate throughput and operations per second with a trade-off of slightly higher latencies for file operations

Home > Storage accounts > qtstorageaccessreplica >

imagereplica Container

Container

Search (Ctrl+ /) Upload Change access level Refresh Delete Change tier Acquire lease Break lease View snapshots Create snapshot

Overview Diagnose and solve problems Access Control (IAM)

Authentication method: Access key (Switch to Azure AD User Account)
Locations: imagereplica

Search blobs by prefix (case-sensitive) Show deleted blobs

Add filter

Name	Modified	Access tier	Blob type	Size	Lease state
quote5.jpg	11/3/2021, 7:42:08 AM	Hot (Inferred)	Block blob	1.28 MB	Available
quote6.jpg	11/3/2021, 7:42:08 AM	Hot (Inferred)	Block blob	4.2 MB	Available

- EFS mounts don't work on Windows Servers so AWS has a service called FSx

Understand General Purpose Volumes-

- Here lets focus on GP2 which are SSD-backed, low cost and default in all regions.
- Lets look at the documented performance characteristics of a GP2 volume
- Max IOPS/Volume => 16000
- Max Throughput/Volume => 250 MB/s
- Max Burstable IOPS => 3000
- The actual max IOPS for any specific volume is based on size 3 IOPS/GB i.e only volume or size 5.3 TB or larger can hit a max IOPS
- Baseline IOPS and Burst Limits
- Lets consider a smaller volume 100 GB GP2 Volume, at this size the volume has baseline performance of 300 IOPS, but it can burst upto 3000 IOPS using a bucket credit system

Throughput:

- IOPS is a tricky metric and you typically care more about throughput-speed at which data comes on and off the volume
- GP2 Volumes have a throughput limit based on size
|Volume Size| Throughput Limit | < 170 GB| 128 MB/s | |170-334 GB| 250 MB/s (burst) | >334 GB| 250 MB/s |
- So take, use GP3 volumes with higher throughputs to save costs rather than provisioned IOPS

Advantages-

- EFS can be used when our application needs a shared file system, which can be accessed by multiple instances at the same time.
- Reliable
- Pay as you go (cost effective)
- Performance that scales to support any workload.
- Dynamic elasticity
- Fully managed by AWS.
- Security and compliance.
-

Disadvantages-

- No Windows instances. Amazon EFSs are not supported on AWS Windows EC2 instances. EFS volumes can only be used with non-Windows instances, such as Linux, that support NFS volumes.
- No system boot volumes. Amazon EFS volumes also cannot be used for system boot volumes.

Use Cases-

- Web serving and content management.
- Application development and testing.
- Media and entertainment.

- Database backups.

NFS:

- Network File System is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed.

10. Simple Storage Service (S3)

- Object storage is a computer data storage architecture that manages data as objects, as opposed to other storage architectures like file systems which manages data as a file hierarchy, and block storage which manages data as blocks

What is S3 in AWS?

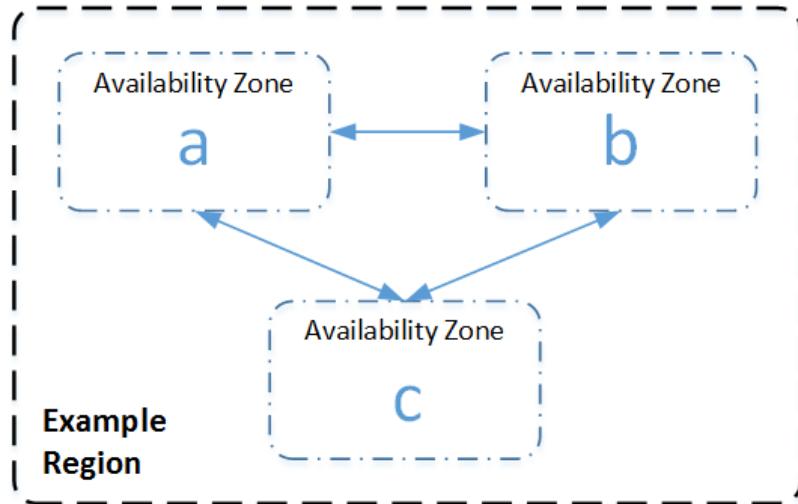
- Amazon S3 or Amazon Simple Storage Service is a service offered by AWS that provides object storage through a web service interface.
- This is similar to google drive but more advanced for technical people.
- This is not like block storage we cannot mount anywhere.
- It's a only drive we do not need to create any file system.
- It is designed to make web-scale computing easier for IT people.
- Basically, s3 is storage for the internet which has a simple webservices interface for simple storing and retrieving data anytime or from anywhere on the internet.
- S3 has a distributed data-size architecture where objects are redundantly stored in multiple locations (min 3 zones).

Advantages-

- Reliable Security
- All-time Availability
- Very Low cost
- Ease of Migration
- The Simplicity of Management

Distributed architecture-

- An AWS Availability Zone is a physically isolated location within an AWS Region. Within each AWS Region, S3 operates in a minimum of three AZs, each separated by miles to protect against local events like fires, power down, etc.



S3 concepts –

- Buckets
- Keys
- Objects
- S3 bucket resources
- S3 storage classes

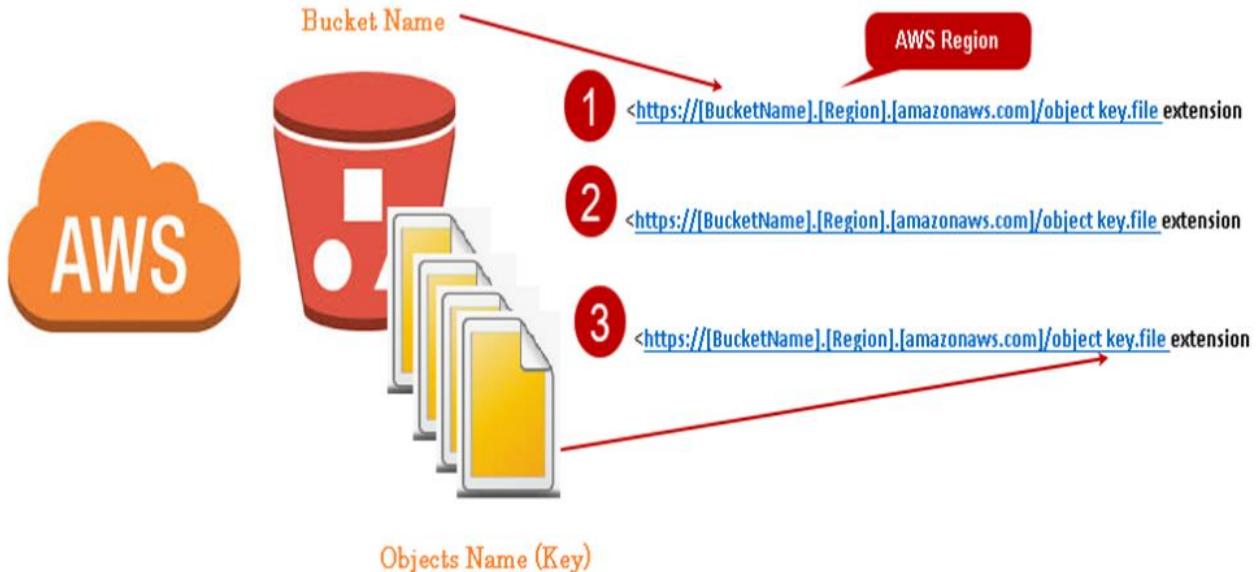
Buckets/Keys/Objects –

- An Amazon S3 bucket is a public cloud storage resource available in Amazon Web Services' (AWS) Simple Storage Service (S3), an object storage offering.
- Amazon S3 buckets, which are similar to file folders, store objects, which consist of data.
- Data is stored inside a bucket.
- Bucket is nothing but a flat container of objects
- Individual Amazon S3 objects can range in **size** from 1 byte to 5 terabytes.
- The largest object that can be uploaded in a single PUT is 5 gigabytes. For objects larger than 100 megabytes, customers should consider using the Multipart Upload capability.
- There is no limit on objects per **bucket**.
- You can create or upload multiple folders in one bucket but you cannot create a bucket inside a bucket (Nested bucket not possible)
- S3 bucket is region specific.
- You can have 100 buckets per account, this number can be increased.
- By default buckets and its objects are private, thus by default only the owner can access the buckets.
- There are some naming rules while creating a s3 bucket.
- A S3 bucket name should be globally unique across all the regions and accounts.
- Bucket names must be between 3 and 63 characters long.
- Bucket names can consist only of lowercase letters, numbers, dots (.), and hyphens (-).
- Bucket names must begin and end with a letter or number.

- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).

url –

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html>



S3 sub resources-

- Lifecycle
- Website
- Versioning
- Access control lists

S3 object lifecycle-

- A lifecycle configuration is a set of rules that are applied to the objects in specific S3 buckets.
- Each rule specifies which objects are affected and when those objects will expire (on a specific date or after some number of days).

Website-

- You can use Amazon S3 to host a static website. On a static website, individual webpages include static content.
- When you configure a bucket as a static website, you must enable static website hosting, configure an index document, and set permissions.
- You can enable static website hosting using the Amazon S3 console, REST API, the AWS SDKs, the AWS CLI, or AWS CloudFormation.
- Configuring a static website using a custom domain registered with Route 53

Versioning-

- Versioning in Amazon S3 is a means of keeping multiple variants of an object in the same bucket.
- You can use the S3 Versioning feature to preserve, retrieve, and restore every version of every object stored in your buckets.
- With versioning you can recover more easily from both unintended user actions and application failures.
- Versioning-enabled buckets can help you recover objects from accidental deletion or overwrite
- For example, if you delete an object, Amazon S3 inserts a delete marker instead of removing the object permanently.
- The delete marker becomes the current object version.
- You can still recover the object by deleting the delete marker.
- This versioning is incremental versioning.
- You can use versioning with S3 lifecycle policy to delete older versions to move them to cheaper s3 versions.
- Versioning can be applied to all objects in a bucket, not partially applied

Access Control Lists-

- Amazon S3 access control lists (ACLs) enable you to manage access to buckets and objects.
- Each bucket and object has an ACL attached to it as a subresource.
- It defines which AWS accounts or groups are granted access and the type of access.
- When a request is received against a resource, Amazon S3 checks the corresponding ACL to verify that the requester has the necessary access permissions.
- <https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html>

MFA delete in S3 bucket-

- Multifactor authentication delete is a versioning capacity that adds another level of security in case your account is compromised.
- This adds another layer of security for the following
- Changing your bucket versioning state or permanently deleting an object version



AWS S3 Storage Classes

S3 offers a variety of storage class designs to cater to different usecases. You can choose the appropriate storage class & also move objects from one storage class to other.

- Amazon s3 standard
- Amazon glacier deep archive
- Amazon glacier
- Amazon s3 infrequent access (standard IA)
- Amazon s3 one-zone IA
- Amazon S3 intelligent tiering
- Theory Link:- <https://aws.amazon.com/s3/storage-classes/>
- Pricing link:- <https://aws.amazon.com/s3/pricing/>

Storage Classes

1. S3 Standard:

- This is default storage classes.
- It offers HA (high availability), durability and performance
- In this storage class, you pay more for storage and less for accessing
- When we use S3 the files in S3 standard are synchronously copied across three facilities and designed to sustain the loss of data in two facilities
- Durability is 99.99999999%
- Designed for 99.9% availability over a given year.
- Largest object that can be uploaded in a single PUT is 5TB.

1. S3 Standard Infrequent access:

- Designed for the objects that are accessed less frequently i.e. paying more for access and less for storage

- Resilient against events that impact an entire AZ.
- Availability is 99.9% in year.
- Support SSL for data in transit and encryption of data at rest.
- Data that is deleted from S3 IA within 30 days will be charged for a full 30 days.

1. S3 One Zone infrequent access:

- In this data is stored only in one AZ when compared to S3 Standard/S3 standard infrequent access where the data is copied in 3 AZ's
- Storage cost is less and access cost is high.
- Ideal for those who want lower cost option of IA data.
- It is good choice for storing secondary backup copies of one premise data or easily re-creatable data.
- You can use S3 lifecycle policies.
- Availability is 99.5%
- Durability is 99.99999999%
- Because S3 zone-IA stores data in a single is it data stored in this storage class will be lost in the event of AZ destructions.

1. S3 Intelligent-Tiering:

- Designed to optimize costs by automatically moving the data from one storage class to another based on access patterns.
- It works by storing object into access tiers.
- If an object in the infrequent access tire is it automatically moved back to frequent access tier.
- There are no retrieval fees when using S3 intelligent tiering storage class and no additional tiering fee when objects are moved between access tiers.
- Same low latency and high performance of S3 standard.
- Object less than 128 KB cannot move to IA.
- Durability is 99.99999999%
- Availability is 99.9%

1. S3 Glacier:

- S3 Glacier is secure durable low cost storage class for data archiving.
- To keep cost low yet suitable for wearing needs S3 Glacier provides 3 retrieval options that range from a few minute to hours.
- You can upload object directly to Glacier or use lifecycle policies.
- Durability is 99.99999999%
- Data is resilient in the event of one entire AZ destructions.
- Support SSL for data in transit and encryption data at rest.

1. S3 Glacier Deep Archive:

- This is used for long-term archival storage (compliance reasons)

- You may be accessing this data hardly once or twice a year.
- Glacier Deep archive is the ideal solution for replacement of tape storage
- With S3 Glacier Deep Archive, Data can be restore within 12 hours
- Design to retain data for long period eg..10 years.
- All object store in S3 Glacier deep achieve are replicated and told across at least at three geographically depressed AZ.
- Durability is 99.99999999%
- Storage cost is up to 75% less than for the existing S3 Glacier storage class.
- Availability is 99.9%

Characteristic	S3 Standard	S3 Intelligent Tiering	S3 IA	S3 One-Zone IA	S3 Glacier	S3 Glacier Deep Archive
Designed for Durability	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Designed for Availability	99.99%	99.90%	99.90%	99.50%	99.99%	99.99%
Availability SLA	99.90%	99%	99%	99%	99.90%	99.90%
AZs	>= 3	>=3	>=3	1	>=3	>=3
Min charge/object	N/A	N/A	128 KB	128 KB	40 KB	40 KB
Min storage duration cache	N/A	30 days	30 days	30 days	90 days	90 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milli seconds	milli seconds	milli seconds	milli seconds	minutes or Hours	hours

Versioning:

- By default aws will try to show the latest object, if we need to maintain multiple versions of the object then we need to enable versioning
- After enabling the versioning, each object gets the version id and default url will point to latest version

S3 supports enable versioning and suspend versioning.

Bucket Deletion

- S3 bucket deletion requires the bucket to be empty before deleting.

Policies and Permissions in Amazon S3:

In its most basic sense, a policy contains the following elements:

- **Resources**— Buckets, objects, access points, and jobs are the Amazon S3 resources for which you can allow or deny permissions. In a policy, you use the Amazon Resource Name (ARN) to identify the resource. For more information, see [Amazon S3 resources](#).
- **Actions**— For each resource, Amazon S3 supports a set of operations. You identify resource operations that you will allow (or deny) by using action keywords. For example, the s3>ListBucket permission allows the user to use the Amazon S3 [GET Bucket \(List Objects\)](#) operation. For more information about using Amazon S3 actions, see [Amazon S3 actions](#). For a complete list of Amazon S3 actions, see [Actions](#).
- **Effect**— What the effect will be when the user requests the specific action—this can be either allow or deny.
If you do not explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user can't access the resource, even if a different policy grants access. For more information, see [IAM JSON Policy Elements: Effect](#).
- **Principal**— The account or user who is allowed access to the actions and resources in the statement. In a bucket policy, the principal is the user, account, service, or other entity that is the recipient of this permission. For more information, see [Principals](#).
- **Condition**— Conditions for when a policy is in effect. You can use AWS-wide keys and Amazon S3-specific keys to specify conditions in an Amazon S3 access policy. For more information, see [Amazon S3 condition key examples](#).

S3 Naming –

The following example bucket names are valid and follow the recommended naming guidelines:

- docexamplebucket1
- log-delivery-march-2020
- my-hosted-content

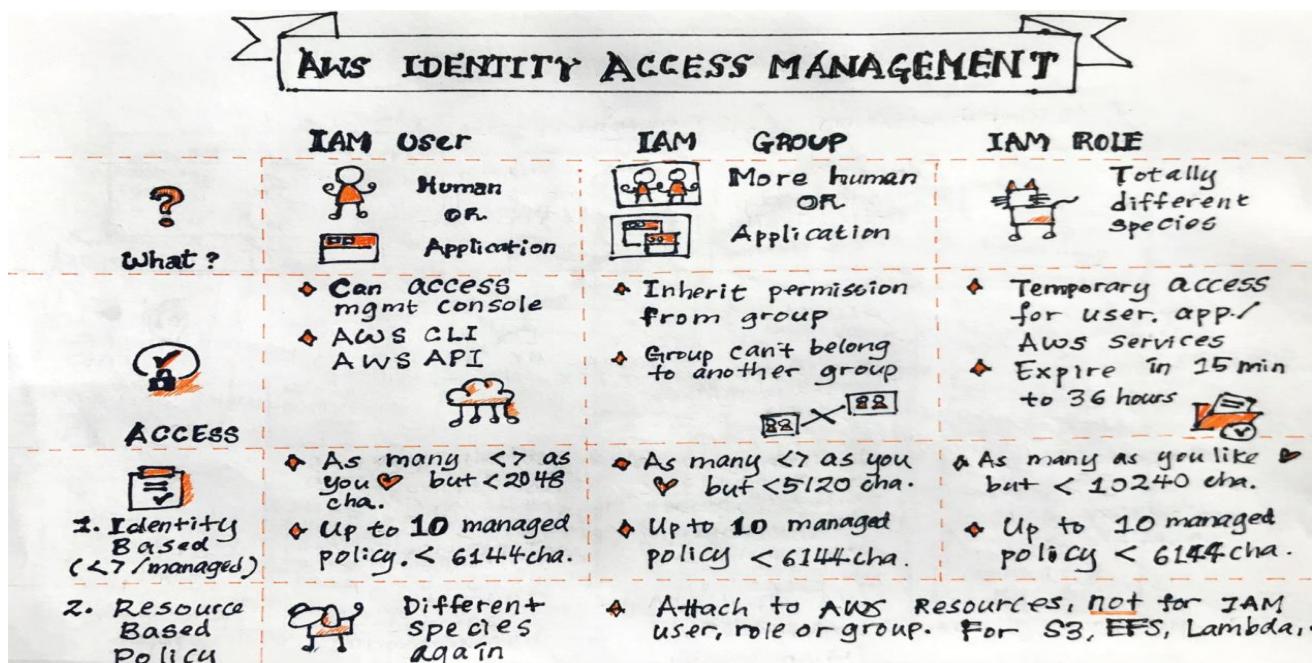
The following example bucket names are valid but not recommended for uses other than static website hosting:

- docexamplewebsite.com
- www.docexamplewebsite.com
- my.example.s3.bucket

The following example bucket names are *not* valid:

- doc_example_bucket (contains underscores)
- DocExampleBucket (contains uppercase letters)
- doc-example-bucket- (ends with a hyphen)

11. IAM-Identity and Access Management



- AWS identity and access management service (IAM) is a web service that helps you securely control access to AWS resources for your users.
- AWS Identity and Access Management (IAM) provides fine-grained access control across all of AWS.
- With IAM, you can specify who can access which services and resources, and under which conditions.
- With IAM policies, you manage permissions to your workforce and systems to ensure least-privilege permissions.
- IAM is a feature of your AWS account and is offered at no additional charge.
- IAM Provides –
- Shared access to your AWS account.
- Granular Permissions.
- Secure access to AWS resources.
- Integrated with many of the AWS resources.
- Free to use.
- Global service
- An AWS Identity and Access Management (IAM) user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.
- A user in AWS consists of a name and credentials.
- We can also create a user as a service account.
- An IAM user with administrator permissions is not the same thing as the AWS account root user.
- As a best practice, do not use the AWS account root user for any task where it's not required. Instead, create a new IAM user for each person that requires administrator access.
- Then make those users administrators by placing the users into an “Administrators” user group to which you attach the Administrator-Access managed policy.
- IAM refers to a framework or policies and technologies for ensuring the proper people in an organization have the appropriate access to technology Resources.

OR

- AWS Identity and access Management is web service that helps you securely control access to AWS Resources you use IAM to control who is Authenticated (signed-in) and authorized (has permission) to use Resources.
- When you create an AWS account, you begin with a single sign-in identity that has completely access to all AWS services and resources in the account.
- This identity is called the AWS account “Root-User” and is accessed by signing-in with the email address and password that you used to create the account.
- AWS Strongly recommends that you do not use the root user for your everyday task, even the administrative ones.
- Use other IAM user accounts to manage the administrative task of your account and securely lock away the root user credentials and use them to perform only a few account and service Management Task.
- IAM user limit is 5000 per AWS Account you can add up to 10 users at one time.
- You are also limited to 300 Groups per AWS account.
- You are limited to 1000 IAM Roles under AWS account.
- Default limits of Managed Policies attached to an IAM role and IAM user is 10.

- IAM users can be member of 10 groups (max).
- We can assign two access keys (max) to an IAM user.
- Shared access to your AWS account.

You can grant other people permission to administer and use Resources in your AWS account without having to share your access Credentials (password or access key).

GRANULAR PERMISSIONS-

- You can Grant different permission to different people for different resources.
- For Instance, you can allow Some users complete access to EC2, S3, dynamo DB, Red shift while for others, you can allow read only access to just some S3 brackets or permission to administer just some EC2 instances or to access your billing Information but nothing else.
- Secure access to AWS resources for applications that run on amazon EC2.
- You can use IAM features to Securely give application that run on EC2 instances the credentials that they need in order to access other AWS Resources Example include S3 Buckets and RDS or DynamoD3 database.

Multifactor Authentication (MFA)-

- You can add two factor authentication to your account and to individual users for extra security you can use physical hardware or virtual MFA (for example. Google Authenticator).

Identity federation-

- You can allow users who already have password elsewhere for example. In your Corporate Network or with an Internet Identity provider to get temporary access to your AWS account.

Identity Information for assurance-

- If you use AWS cloud trail, you receive log Records that include Information about those who made request for resources in your account that information is based on IAM Identities.

PCI-DSS Compliance-

- IAM support the processing storage and transmission of credit card by merchant or service provider, and has been validated as being compliant with payment card Industries (PCI) Data Security Standard (DSS).

Eventually Consistent-

- If a request to change some data is successful the change is committed & Safely stored However the change must be Replicated across IAM which can take some time.
- IAM achieves high availability by Replicating data across multiple servers within AWS data center around the world.
- Free to Use.
- AWS IAM is feature of your AWS account offered at no additional charges.
- You will be charged only for use of others AWS products by your IAM users.

What are IAM policies?

Policies provide authorization to AWS services and resources

Two parts:

Specification: *Defining* access policies

Enforcement: *Evaluating* policies

```
[{"Statement": [ { "Effect": "Allow", "Action": ["s3:Get*", "s3>List*"], "Resource": "*" } ]}
```

When you *define* access policies. You specify which IAM *principals* are allowed to perform which *actions* on specific AWS *resources* and under which *conditions*.

IAM enforces this access by *evaluating* the AWS request and the policies you defined and returns either yes or no answer.

AWS re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.



IAM Terms-

Principal-

- A principal is a person or application that can make a request for an action or operation on an AWS Resources.
- Your administrative IAM user your first principal.
- You can allow users and services to assume a Role.
- IAM users, Roles, Federated users and application are all AWS Principals.
- You can support federated user or Programmatic access to allow an application to access yours AWS Account.

Request-

When a principle tries to use the AWS management Console, the AWS API, or the AWS CLI, that principal sends a Request to AWS the Request includes the following information.

- Actions
- Resources
- Principal
- Environment data
- Resources Data
- Actions- That principal wants to perform.
- Resources- Upon which the actions are performed.
- Principal- Information including the environment from which the Request was made.

Authentication-

- A principal sending a request must be authenticated (signed in to AWS) to send a Request to AWS.
- Some AWS services, Like AWS S3, allow request from anonymous users, they are exception to the rule.
- To authenticate from the console as a root user, you must sign in with your user name and password.
- To authenticate from the API to CLI, you must provide your access key and secret key.
- You might also require to provide additional Security Information like MFA (example Google Authentication).

Authorization-

- To authorize Request, IAM users value from the Request content to check for matching policies and determine whether to allow or deny the request.
- IAM policies are stored in IAM as JSON documents and specify the permission that are allowed or denied.
- User (Identity) based policies specify permission allowed Denied for principal.

Note – By default, only the AWS root user has access to all the resources in that account.

- Resource base policies- specify permission allowed/ denied for Resources popular for granting Gross account permissions.
 - IAM checks each policy that matches the context of your Request.
 - If a single policy includes a denied action, IAM Denies the entire Request and stop evaluating this is called Explicit deny.
 - The evaluation logic follows these Rules.
 - By default all request are denied.
 - An explicit allow overrides this default.
 - An explicit deny overrides any allows.
-
- You can create a new IAM policy in the AWS management console using one of the following ways-

- JSON- you can create your own JSON syntax.
- Visual- you can construct a new policy from scratch in the visual editor if you use the visual editors, you can do not have to understand JSON syntax.
- Import- you can Import a managed policy within your account and then edit the policy to customize it to your specific Requirement.

Actions

- Actions are defined by a service, and are the things that you can do to a resources Such as Viewing, Creating, Editing and Deleting that Resources.
- IAM supports approx. 40 actions for a user Resource including create user, Deleting user etc.
- Any action or resources that are not explicitly allowed are denied by default.
- After your request has been authenticated and authorized, AWS approves the actions in your Request.

Resource

- A resource is an entity that exist with is a service.
- Examples are EC2 instances, S3 bucket, IAM user, Dynamo DB table.
- Each WAS service defines a set of actions that can be performed on each resource.
- After AWS approves the action in your request those actions can be performed on the Related Resource within your account.
- If you create a request to perform an unrelated action on a Resource, that Request is denied.
- When you provide permission using an identity based policy in IAM than you provide permissions to access resources only within the same account.

Identity Federation

- If your account user already have a way to be authenticated such as authentication through your corporate network.
- You can federate those user Identities in to AWS.
- A user who has already logged to the corporate using their corporate Identity.
- The corporate can replace their existing identity in your AWS account.
- This user can work in the AWS management console.
- Similarly, an application that the user is working with can make programmatic request using permissions that you define.

Federation is particularly use in these cases

- If your corporate directory is compatible with security Assertion Markup Language (2.0)
- You can configure your corporate directory to provide sign-on (SSO) access to the AWS Management for your users.
- If your corporate directory is not compatible with SAML (2.0)

- You can create Identity Broker application to provide single sign-on (SSO) access to the AWS management console for your user.
- If your corporate directory is Microsoft active Directory, you can use AWS directory services to establish trust between your corporate directory and yours AWS account.
- If you are creating a mobile app or web based app that can let uses identity themselves through an internet identity provider like login with Amazon, Facebook, Google or any open id connect (OIDC) compatible identity provider the app can use web federation to access AWS.
- AWS recommends to use AWS cognito for identity federation.

IAM Users and SSO

- IAM users in your account have access only to the AWS Resources that you specify in the policy that is attached to the users or to an IAM Group that the user belongs to.
- To work in the console user must have permission to perform the action that the console performs such as listing and creating AWS resources.



- IAM Identities-
- Users
- Groups.
- Role.
- IAM identities is that you create under your AWS account to provide authentication for people application and process in your AWS account.
- Identities represent the user and can be authenticated and the authorized to perform action in AWS.

- Each of this can be associated with one or more policies to determine what actions are user role or member of a group can do with which resources and under what conditions.
- IAM group is a collection of IAM users.
- IAM roll is very similar to IAM user.

IAM Users

- IAM user is an entity that you create in AWS it represent the person or service who uses the IAM user to interact with AWS.
- You can create 10 users at a time.
- IAM user can represent an actual person or an application that requires AWS access to perform actions on AWS resources.
- A primary use for IAM users is to give people the ability to sign in to the AWS management console for interactive tasks and to make programmatic requests to AWS services using the API or CLI.

For any Users you can assign them-

- A username and password to access the AWS console.
- An access key ID and secret key that they can use for programmatic access.
- The newly created IAM users have no password and no excess key you need to create the user password.
- Each IAM user is associated with one and only one AWS account.
- Users can define within your account so users do not have to pay by the parent account.

IAM Groups

- An IAM group is a collection of IAM users.
- It is a way to assign permission or policies to multiple users at a time.
- Use groups to specify permissions for a collection of users which can make those permissions easier to manage for those users for example you could have a group called HR and that group the types of permissions that HR department typically needs.
- Any user that joins the group automatically has the permission that are assigned to the group.
- If a new user joins your organization and should have HR privileges you can assign the appropriate permissions by adding the user to that group.
- If a person changes a job in your organization instead of editing that user's permissions you can remove him or her from the old group and add him or her to the appropriate new group.

IAM Groups

Why (Benefits)

- Reduces the complexity of access management as number of users grow
- Easy way to reassign permissions based on change in responsibility
- Easy way to update permissions for multiple users
- Reduces the opportunity for a user to accidentally get excessive access

Do

- Create groups that relate to job functions
- Attach policies to groups
- Use **managed policies** to logically manage permissions
- Manage group membership to assign permissions



IAM Groups Limitations

- A group is not truly an identity and it is because it cannot be identified as a principal in a permission policy.
- Groups can't be nested.
- You have a limit of 300 groups in an AWS account.
- A user can be member of up to 10 IAM groups.
- IAM role is very similar to user in that it is an identity with permission policies that determine what the Identity can and cannot do in AWS.
- An IAM role does not have any credentials password or access key associated with it.
- Instead of being uniquely associated with one person a role is intended to be assumable by anyone who needs it.
- An IAM user can assume a role to temporarily take on different permissions for a specific task.
- An IAM role can be assigned to a federated user who signs in by using an external identity provider instead of IAM.

IAM Roles

IAM Roles

What

- Another identity with permission policies that determine what the identity can and cannot do in AWS
- Can be assumed by anyone who needs it; not uniquely associated with one person or application
- Does not have credentials; access keys are created and provided dynamically

When

- Give cross-account access
- Give access within an account
 - E.g. access for application running on Amazon EC2
- [Federation] Give access to identities defined outside AWS
 - E.g. access for identities maintained in your corporate IdP



- IAM role is very similar to user in that it is an identity with permission policies that determine What the Identity can and cannot do in AWS.
- An IAM role does not have any credentials password or access key associated with it.
- Instead of being unequally associated with one person a role is intended to be assumable by anyone who needs it.
- An IAM user can assume a roll to temporary take an different permissions for a specific task.
- An IAM role can be assigned to a federated user who signs in by using an external identity provider instead IAM.

IAM Temporary Credentials-

- Temporary credentials are primarily used with I am Rose but there are also other uses.
- You can request temporary credentials that have a more restricted set of permissions than you are standard IAM users.
- This present you from accidentally performing task that are not permitted by the more restricted credentials.
- A benefit of temporary credentials is that they expire automatically after a set of time.

Permissions and Policies

The access management portion of AWS identity and access management IAM helps you get define what are you this are there in the di is allowed to do in and around dolphin the referred to as authorization.

Permission our granted through policies that are created on then attach to users groups or roles.

Policies and Users.

IAM Policies

Two types of identity-based policies in IAM

- Managed policies (newer way)
 - Can be attached to multiple users, groups, and roles
 - AWS managed policies (created and managed by AWS)
 - Customer managed policies (created and managed by you)
 - Up to 5K per policy
 - Up to 5 versions
 - You can limit who can attach managed policies
- Inline policies (the older way)
 - You create and embed directly in a single user, group, or role
 - Variable policy size (2K per user, 5K per group, 10K per role)



- By default I am uses can't access anything in your account.
- You grant permissions to a user by creating a policy which is document that defines the effect actions resource and optional conditions.
- Any actions are resources that are not explicitly allowed by default.

Policy types and core use cases

AWS Organizations Service control policies (SCPs)	<i>Guardrails to disable service access on the principals in the account</i>
AWS Identity and Access Management (IAM) As Permission Policies and Permission Boundaries	<i>Grant granular permissions on IAM principals (users and roles) and control the maximum permissions they can set</i>
AWS Security Token Service (AWS STS) Scoped-down policies	<i>Reduce general shared permissions further</i>
Specific AWS services Resource-based policies	<i>Cross-account access and to control access from the resource</i>
VPC Endpoints Endpoint Policies	<i>Controls access to the service with a VPC endpoint</i>

All use the same policy language



© 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.



IAM multiple policies

- User or groups can have multiple policies attached to them that grant different permission.
- In case of multiple policies attach to user or a group the users permission are calculated based on the combination of policies.

Federated users and Roles

- Federated user don't have permanent identities in your AWS account the way that IAM users do.
- To assign permission to federated uses you can create an integrity referred to as a role and define permission for the role.
- When a federated user sign in to AWS the user is associated with the role and is granted the permission that are defined in the role.

Resource Based Policies-

- In some cases like S3 bucket you can attach a policy to resource in addition to attaching into a group or user this is called a resource based policy.

- A resource based policy contains slightly different information than a user-based policy in a resource-based policy you specify what actions are permitted and what resources are affected.
- You also explicitly list who is allowed access to the resource a principal.
- Resource-based policies include a principal element that specifies who is granted the permission.

IAM User-

- When you first create an AWS account you create account or root user identity which you use to sign in to AWS account.
- The account root user credentials are the email address used to create the account and a password which can be used to sign in to the AWS management console as the root user.
- When you sign in as the root user you have complete unrestricted access to all resources in your AWS account including access to your billing information and the ability to change your password.
- The level of access is necessary when you initially set up the account.
- It is not possible to restrict the permissions that are granted to the AWS account.
- An IAM user is an entity that you create in AWS it represents the person or service who accesses the IAM user to interact with AWS.
- An IAM user can represent an actual person or an application that requires AWS access to perform actions on AWS resources.
- IAM users are global entities like an AWS account so today no reason is required to be specified when you define users' permissions users can use AWS services in any geographically region.

For any user you can assign them–

- A Username and password to access the AWS console.
- An access key (access key ID and secret key) that they use for programmatic access (issue in requests) to your AWS services using API and CLI.
- You assign either or both based on the user's activities and needs.
- You can view and download your secret access key only when you create the access key.
- You cannot view or recover a secret access key later.
- If you lose your secret access key you can create a new access key.
- Each IAM user is associated with one AWS account.

AWS recommends

- AWS recommends that you don't use root user credentials everyday access.
- Also AWS recommends that you do not share your root user credentials with anyone because doing so gives them unrestricted access to your account.

- Create an IAM user for yourself and then assign yourself administrative permission for your account.
- You can then sign in as that user to add more users as needed.
- An IAM user with administrator permission is not the same thing as the AWS account root user.

By default a new IAM user

- A new IAM user has no permission to do anything.
- Has no password and no access key neither and access key ID nor a secret access key it means no credentials of any kind.
- You must create the type of potentials for and IAM uses best on what the user will be doing.
- You can grant user permissions by attaching I am policies to them directly or making them members of I am group where they inherit the group policies or permissions.
- You can have up to 5000 users per AWS account.

IAM Role

- An IAM Role Is a set of permission that grant excess to actions and resources in AWS.
- This permissions are attached to the role not to an I am user or group instead of being unequally associated with one person or role is intended to be assumable by anyone who needs it.
- A Role does not have standard long term credentials password or access keys associated with it.
- If a user assumes a role temporary security credentials are created dynamically and provided to the user.
- An IAM user in the same AWS account.
- An IAM used in the different AWS account.
- A web service offered by AWS such as Amazon elastic complete cloud.

There are two things to use role-

- Interactively in the IAM console.
- IAM uses in your account using I am console can switch to a roll to temporary use that permissions of the role in the console.
- The user give up there original permission and take on the permission assigned to the role.
- When they use exits the role their original permission are restored.
- Programmatically with the AWS CLI tools for windows PowerShell or API.

- An application or service offered by AWS (like Amazon EC2) can assume a role by requesting temporary security credentials for a role with which to make programmatic request to AWS.
- You use a role this way so that you didn't have to share or maintain long term security credentials for each entity that require access to a resource.

Difference between IAM role and resource based policy

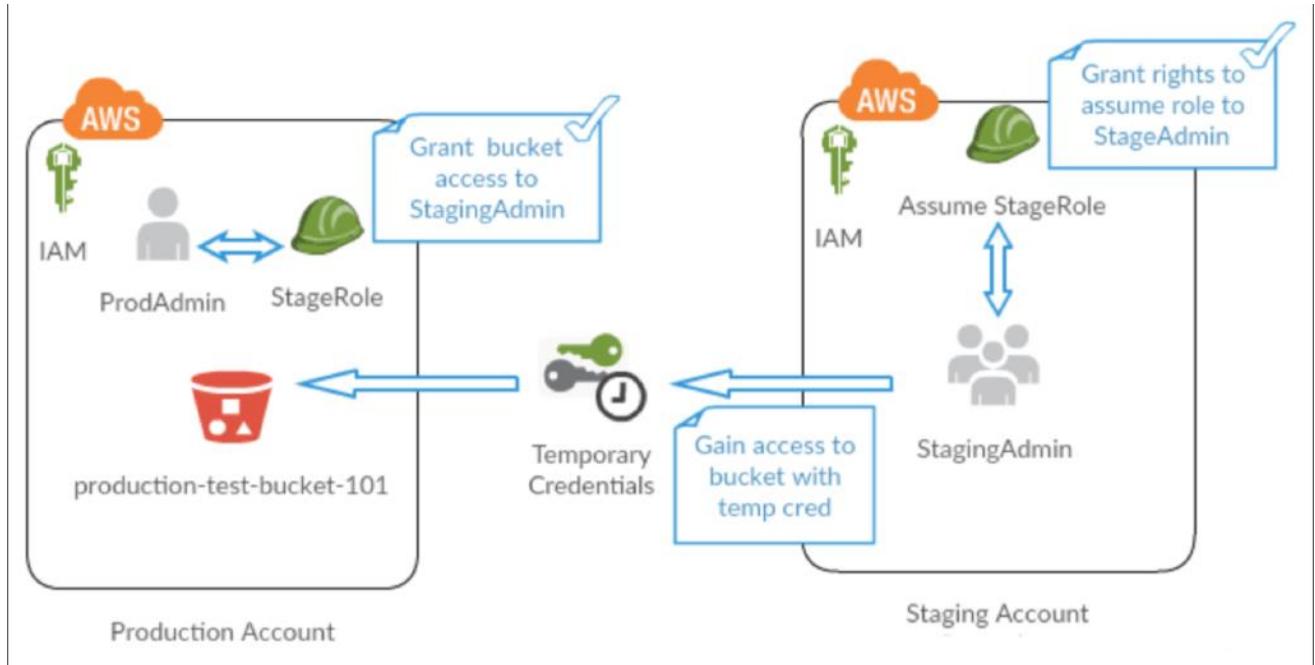
- Unlike a user based policy a resource best policy specifies who can access that resource.
- Cross account access with a resource based policy has an advantage over a role with a resource that is excess through the resource best policy the user still work in the trusted account and does not have to give up his or her user permission in place of the role permission.
- In other words the user continues to have access to resource in the trusted account at the same time as he or she has excess to the resource in the trusting account.
- This is useful for task such as copying information to or from the shared resource in the other account.
- Note that not all services support resource based policy.

IAM role delegation

- Delegation is the granting of permission to someone to allow to resources that you control.
- Delegation involves setting up a trust between the account that owns the resource the trusting account and the account that contains the users that need to access the resource the trusted account.
- The trusted and trusting account can be any of the following.
 - The same account.
 - To accounts that are both under your organizations control.
 - To account owned by different organization.
- To delegate permission to access resource you create an IAM role that has two policies attached.
 1. The trust policy
 2. The permission policy
- The trusted entity is included in the policy as the principle element in the documents.
- When you create a trust policy you cannot specify a wild card as a principal.

Cross account permissions

- You might need to allow users from another AWS account to access resources in your AWS account if so don't share security credentials such as access keys between accounts in state use IAM roles.
- You can define a role in the trusting account that specifies what permissions the IAM user in the other account are allowed.
- You can also designate which AWS account have the IAM users that are allowed to assume the role we do not define users here rather AWS account.



Role for Cross account

- Granting access to resource in one account to a trusted principle in different account.
- Roles are the primary way to grant cross account excess
- However with some of the web services offered by AWS you can attach a policy directly to a resource there are called resource best policy you can use them to grant principles in another AWS account access to the resource.
- The following services support resource best policy
- Amazon S3
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Glacier Vault

12. CloudFront

Amazon Cloudfront is a Web service that gives businesses and Web application developers an easy and Cost effective way to distribute content with low latency and high data transfer Speed.

- Cloudfront is a global service.
- Amazon cloudfront is a web service that speeds up distribution of your Static and dynamic web content such as HTML, CSS, JS and image files to your users.
- Amazon cloud front is a global content delivery network service that accelerates delivery of your websites, API, video content or other web assets.
- The major issues now a days with file or site access over the internet is latency, hence there is another layer of service added over your normal web app infra that is called CDN (connect delivery network)
- CDN provider has multiple storage or you can say cache locations distributed all over the world, which help data/web-app access easy and fast with reduced latency.
- Such CDN service is provided by Amazon with the help of Cloud Front.
- Cloud front dealers you are content through a worldwide network of data centers called edge locations.
- When a user request content that you are serving with cloudfront the user is routed (via DNS resolution) to edge location that provides the lowest latency so that content is delivered with the best possible performance.
- If the content is already in the edge location with the lowest latency cloudfront delivers to immediately.
- This dramatically reduces the numbers of networks that you are users request must pass through which improves performance.
- If not, cloud font retrievers it from an Amazon S3 bucket or an HTTP or web server that you have identified as the source of the definitive version of content (original server).
- Cloudfront also keeps persistent connection with origin server so files are fetched from the Origins as quickly as possible.

You can access Amazon cloud front in the following ways-

1. AWS management console
2. AWS SDKs
3. Cloudfront API
4. AWS Command Line Interface

Cloudfront Edge Locations-

- Edge locations are not tied to availability zones or regions.

- CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.
- Amazon CloudFront uses a global network of 310+ Points of Presence (300+ Edge locations and 13 regional mid-tier caches) in 90+ cities across 47 countries.
- <https://aws.amazon.com/cloudfront/features/?whats-new-cloudfront.sort-by=item.additionalFields.postDate&whats-new-cloudfront.sort-order=desc>

Cloudfront – Regional Edge Cache

- Amazon cloud front has added several regional edge cache locations globally at close proximity to your viewers.
- They are located between your origin web server and the Global edge locations that serve content directly to you are viewer.
- As objects become less popular individual edge locations may remove those objects to make room for more popular content.
- Regional edge cache working as a alternative of origin to reduce the burden of origin.
- Regional edge cache have a large catch with than any individual edge location so object remain in the catch a longer at the nearest regional edge caches.
- Regional Edge Caches have larger cache-width than any individual edge location, so your objects remain in cache longer at these locations.
- This helps keep more of your content closer to your viewers, reducing the need for CloudFront to go back to your origin webserver, and improving overall performance for viewers

CloudFront Regional Edge Cache- Working

- When a viewer makes a request on your website or through your application, DNS routes the request to the cloud front edge location that can based serve the users request.
- This location is typically the nearest cloudfront edge location in terms of latency.
- In the edge location, cloudfront checks its cache for the requested files.
- If the files are in the catch a cloudfront returns them to the user.
- If the files are not in the cache, the edge servers go to the nearest regional edge cache to fetch the object.
- Regional edge catches have feature parity with edge locations for example _ a catch a invalidation request remove an object from both edge catch catches and regional edge catches before it expires.
- The next time a viewer request the object cloudfront returns to the origin to fetch the latest version of the Object.
- Proxy method PUT/ POST/ PATCH/OPTIONS/DELETE go directly to the origin from the edge locations and do not proxy through the original edge catches.
- Dynamic content as determined at request time, does not flow through regional edge cache, but goes directly to the Origin.

Features of CF

- Can support both static and dynamic content
- Supports PUT/POST and other http methods.
- Can add custom SSL certificate.
- Invalidation enabled.
- We can add custom error Responses.
- Low TTL (time to live)
- Geo targeting
- Wildcard/zone apex certificate support

New Concepts:

- TTL (Time to Live) :-

1. Time to live (TTL) is the time that an object is stored in a caching system before it's deleted or refreshed.
2. Minimum TTL Specify the minimum amount of time, in seconds, that you want objects to stay in CloudFront caches before CloudFront forwards another request to your origin to determine whether the object has been updated.
3. The default value for Minimum TTL is 0 seconds. Maximum is 1 year and default value is 1 day. These values are specified in seconds.

- Invalidation:-

1. Amazon Cloudfront's invalidation feature, which allows you to remove an object from the CloudFront cache before it expires.
2. To invalidate files, you can specify either the path for individual files or a path that ends with the * wildcard, which might apply to one file or to many, as shown in the following example:
 - /images/image1.jpg
 - /images/image*
 - /images/*
 - Pricing

You incur CloudFront charges when CloudFront responds to requests for your objects.

<https://aws.amazon.com/cloudfront/pricing/>

How much load can CF cache server handle?

Amazon Cloud front can handle 250,000 requests per second per distribution. The data transfer rate per distribution is 150 Gbps, the maximum length of the request including headers, query strings is 20,480 bytes

The maximum size of a single file that can be delivered through Amazon CloudFront is 20 GB. This limit applies to all Amazon CloudFront distributions.

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/cloudfront-limits.html>

- Use Cases:-
 1. Accelerate static website content delivery. (S3 combined)

2. CloudFront can speed up the delivery of your static content to viewers across the globe.

<https://aws.amazon.com/blogs/networking-and-content-delivery/amazon-s3-amazon-cloudfront-a-match-made-in-the-cloud/>

- 2. Cloud front can be attached over the load balancer to reduce some load and latency.

13. CloudTrail

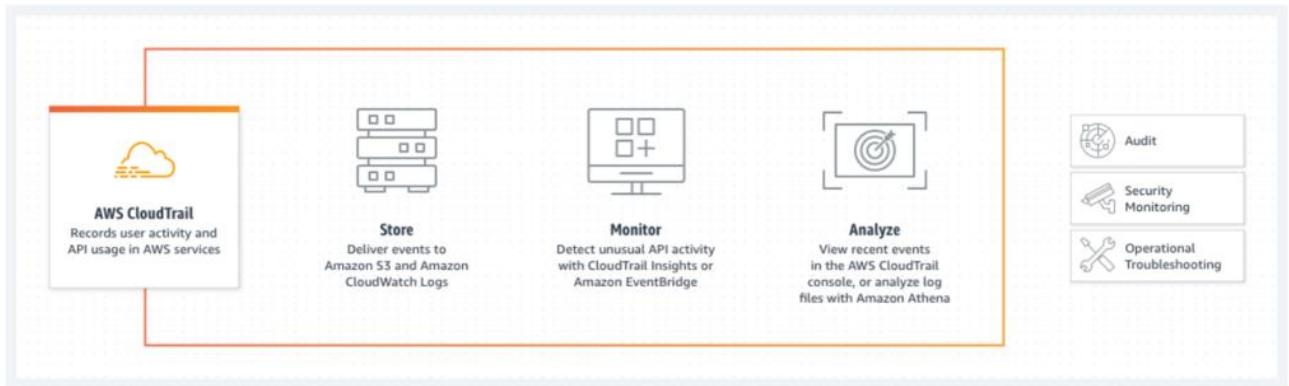
What is AWS CloudTrail?

AWS CloudTrail enables auditing, security monitoring, and operational troubleshooting by tracking user activity and API usage. CloudTrail logs, continuously monitors, and retains account activity related to actions across your AWS infrastructure, giving you control over storage, analysis, and remediation actions.

You should use CloudTrail if you need to audit activity, monitor security, or troubleshoot operational issues.

- AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account.
- Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
- Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.
- CloudTrail is enabled on your AWS account when you create it.
- When activity occurs in your AWS account, that activity is recorded in a CloudTrail event.
- You can easily view recent events in the CloudTrail console by going to Event history. For an ongoing record of activity and events in your AWS account, create a trail.

AWS CloudTrail monitors and records account activity across your AWS infrastructure, giving you control over storage, analysis, and remediation actions.



Benefits of CloudTrail:

CloudTrail helps you prove compliance, improve security posture, and consolidate activity records across regions and accounts. CloudTrail provides visibility into user activity by recording actions taken on your account. CloudTrail records important information about each action, including who made the request, the services used, the actions performed, parameters for the actions, and the response elements returned by the AWS service. This information helps you track changes made to your AWS resources and troubleshoot operational issues. CloudTrail makes it easier to ensure compliance with internal policies and regulatory standards.

Types of Trails:

- **A trail that applies to all regions —**

When you create a trail that applies to all regions, CloudTrail records events in each region and delivers the CloudTrail event log files to an S3 bucket that you specify.

If a region is added after you create a trail that applies to all regions, that new region is automatically included, and events in that region are logged.

Because creating a trail in all regions is a recommended best practice, so you capture activity in all regions in your account, an all-regions trail is the default option when you create a trail in the CloudTrail console.

- **A trail that applies to one region**

When you create a trail that applies to one region, CloudTrail records the events in that region only. It then delivers the CloudTrail event log files to an Amazon S3 bucket that you specify.

Pillars of CloudTrail:

- **Capture** :- capture the activity recorded in the AWS CT events
- **Store** :- store in S3 and CW logs

- **Act:-** Set CW alarms using CW log metrics, send notifications using SNS
- **Review :-** Analyze the logs using AWS CW logs, CT console or CW insights query or any 3rd party compatible software.

CloudTrail records two types of events:

1. Management events that capture control plane actions on resources, such as creating or deleting Amazon Simple Storage Service (Amazon S3) buckets.
2. Data events that capture data plane actions within a resource, such as reading or writing an Amazon S3 object.

1. Management Events:

- Management events provide information about management operations that are performed on resources in your AWS account.
- These are also known as control plane operations.
- Example management events include:
- Configuring security (for example, AWS Identity and Access Management AttachRolePolicy API operations).
- Registering devices (for example, Amazon EC2 CreateDefaultVpc API operations).
- Configuring rules for routing data (for example, Amazon EC2 CreateSubnet API operations).
- Setting up logging (for example, AWS CloudTrail CreateTrail API operations).

1. Data Events:

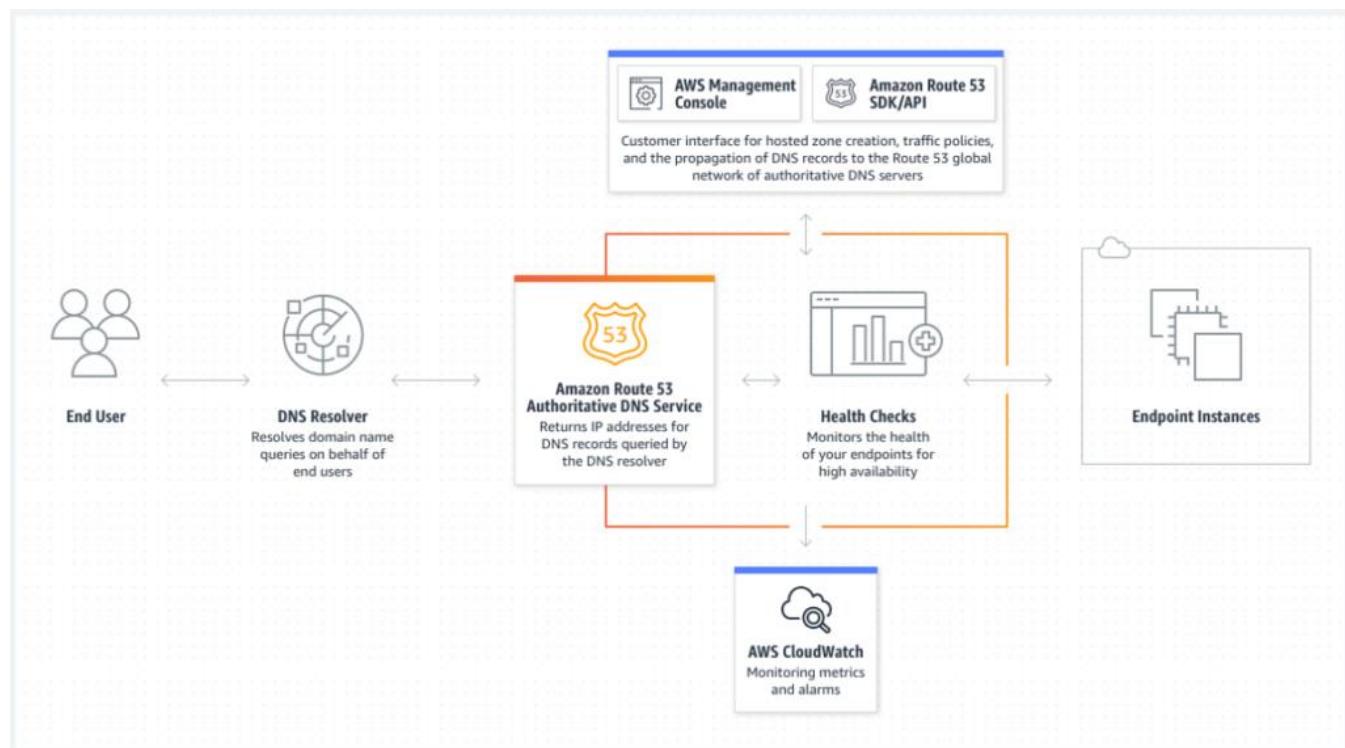
- Data events provide information about the resource operations performed on or in a resource.
- These are also known as data plane operations.
- Data events are often high-volume activities.
- The following data types are recorded:
- Amazon S3 object-level API activity (for example, GetObject, DeleteObject, and PutObject API operations) on buckets and objects in buckets
- AWS Lambda function execution activity (the Invoke API)
- Amazon DynamoDB object-level API activity on tables (for example, PutItem, DeleteItem, and UpdateItem API operations)

CloudTrail uses these events in three features:

- Event history provides a 90-day history of control plane actions at no additional cost. As part of its core audit capabilities, CloudTrail provides customer managed keys for encryption and log file validation to enable immutability. You pay only for what you use of the paid features. Some of the following features are provided at no additional charge. No minimum fees or upfront commitments are required.

- CloudTrail Lake is a managed data lake for capturing, storing, accessing, and analyzing user and API activity on AWS for audit and security purposes. You can aggregate, immutably store your activity logs (control plane and data plane) for up to seven years, and query logs within seconds for search and analysis. IT auditors can use CloudTrail Lake as an immutable record of all activities to meet audit requirements. Security administrators can ensure that user activity is in accordance with internal policies, and DevOps engineers can troubleshoot operational issues such as an unresponsive Amazon Elastic Compute Cloud (EC2) instance or a resource being denied access.
- Trails capture a record of AWS account activities, delivering and storing these events in Amazon S3, with optional delivery to Amazon CloudWatch Logs and Amazon Event Bridge. These events can be fed into your security monitoring solutions. You can use your own third-party solutions or solutions such as Amazon Athena for searching and analyzing logs captured by CloudTrail. You can create trails for a single AWS account or for multiple AWS accounts by using AWS Organizations. AWS CloudTrail Insights analyzes control plane events for anomalous behavior in API call volumes, and can detect unusual activity such as spikes in resource provisioning or gaps in periodic activity.

14. Route 53



Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. Route 53 connects user requests to internet applications running on AWS or on-premises.

- DNS is Domain name system.
- It helps resolve Domain name to IP address.
- This works on port number 53, hence this DNS related service which is provided by Amazon is called as Route 53.
- The Domain Name System is the hierarchical and decentralized naming system used to identify computers, services, and other resources reachable through the internet or other internet protocol networks.
- The resource records contained in the DNS associate domain names with other forms of information

Route 53 serves main 4 functions:-

1. DNS management
 2. Traffic Management
 3. Availability monitoring
 4. Domain registration
- Here we can create and register our own domains.
 - This is not free and the price of the domains depends upon what type of domain we are taking, .com, .NET, .org etc.
 - Here the domain which we are trying to create or register should be available.
 - Route 53 also acts as a registrar.
 - A registrar is where website are registered. Other registrar examples are godaddy.com etc.
 - A registrar also tells us that whether the domain is available or not, and also gives us other options.
 - One of the major use of route 53 is DNS management.
 - It helps route traffic to the resources for your domain, also the domain can be inside our AWS account or outside with any other registrar.
 - Route 53 sends automated request to our resources over the internet to a resource to verify that the server is reachable or not.
 - You can also choose to receive notifications when a resource becomes unavailable and choose to route traffic away from unhealthy resources.

Some important concepts of Route 53

1. **Hosted zones –**
- A hosted zone is an Amazon Route 53 concept.
 - A hosted zone is traditional DNS zone file; it represents a collection of records that can be managed together, belonging to a single parent domain name.
 - Basically a hosted zone is a container that holds information about how we want to route traffic for a domain and a subdomain.
 - When we buy a domain from AWS we don't need to create a hosted zone and name servers everything is created automatically.

- There are 2 types of hosted zones, public and private.
- A Route 53 hosted zone is a collection of records for a specified domain.
- You create hosted zone for a domain and then you create records to tell the domain name system how you want traffic to be routed for the domain.
- Basically hosted zone is a container that holds information about how you want to route traffic for a domain and its subdomains.
- You can create public internet hosted zone or private internal DNS hosted zones.
- For each public hosted zone that you create Amazon route 53 automatically creates a name server NS record and a start of authority SOA record don't change these records.
- Route 53 automatically creates a name server NS record with the same name as your hosted zone.
- At least four name servers that are the authoritative name servers for your hosted zone.
- Do not add, change, or delete name servers in this record.

1. Name servers –

- Nameservers help connect URLs with the IP address of web servers. Nameservers are an important part of the Domain Name System (DNS), which many people call the “phone book of the Internet”.
- Usually we have 4 name servers which will have all the server info, for each hosted zone and they will be unique to the account.
- When someone uses the browser to access our website these name servers inform the browser where to find our resources.

1. Records and types of record sets –

- DNS records are instructions that live in authoritative DNS servers and provide information about a domain including what IP address is associated with that domain and how to handle requests for that domain.
- Name server records identify the 4 name servers that you give to your registrar on your DNS service so that DNS is routed to route 53 name server.
- SOA – every single hosted zone has one and only one SOA (start of authority record) at the beginning of the zone.
- This is not an actual record it just holds the info of who the owner is and its email.

Route 53 performs 3 main functions

- Register a domain
- As a DNS it routes internet traffic
- Check the health of your resources

In case, you choose to use Route 53 for all three functions, perform the steps in this order:

1. Register domain names:

Your website needs a name, such as example.com. Route 53 allows you to register a name for your website or web application. This is a domain name.

2. Route internet traffic to the resources for your domain:

When a user opens a web browser and inscribes your domain name (example.com) or subdomain name (acme.example.com) in the address bar, Route 53 assists connect the browser with your website or web application.

3. Check the health of your resources:

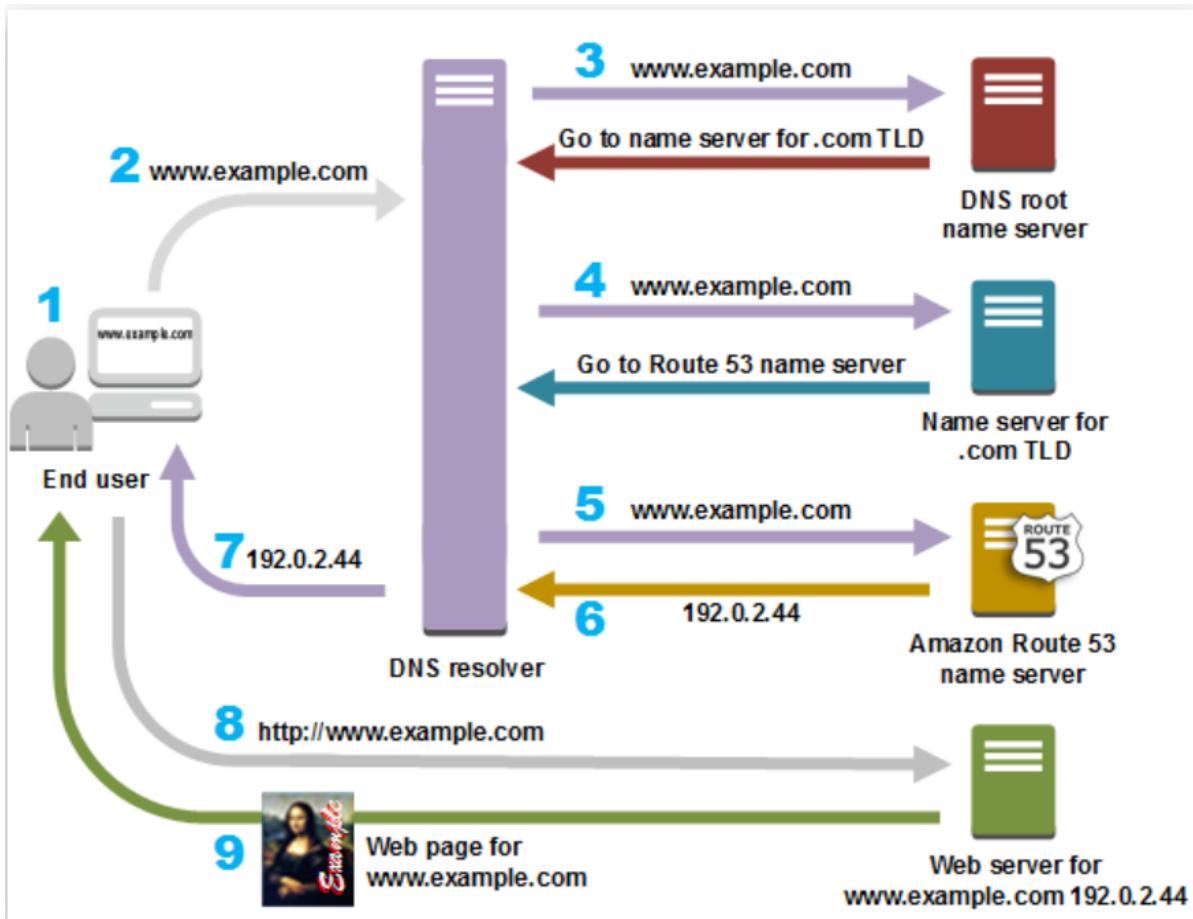
Route 53 makes automatic calls to a resource, such as a web server, across the internet to ensure that it is accessible, available, and functioning. You may also opt to be notified when a resource becomes inaccessible, as well as to redirect internet traffic away from harmful resources.

- Route 53 sends automatic request over the internet to a resource can be web server to verify that the server is reachable functional or available.
- Also you can choose receive notifications when a resource become unavailable and choose to route internet traffic away from unhealthy resources.
- You can use route 53 for any combination of these functions.
- For example you can use route 53 port to register your domain name and to route internet traffic for the domain.
- Or you can use route 53 to route internet traffic for a domain that you register with another domain register.

How Does Route 53 Work?

- A user opens a web browser and sends a request for example.com
- The request from example.com is routed to a DNS resolver, which is usually managed by the Internet Service Provider (ISP).
- The ISP DNS resolver forwards the request from example.com to a DNS root name server.
- The DNS resolver forwards the request from example.com again, this time to one of the top-level domain (TLD) name servers of .com domains. The .com domain name server responds with the names of the four Route 53 name servers associated with the example.com domain. The DNS resolver caches the four Route 53 name servers for future use.
- The DNS resolver chooses a Route 53 name server and forwards the request from example.com to that Route 53 name server.
- The Route 53 name server looks for the record example.com in the hosted zone site.com, gets its value, such as the alias of Amazon CloudFront distribution in the case of simple routing.
- The DNS resolver finally has the right route (CloudFront IP) the user needs and returns the value for the user's web browser.
- The web browser sends a request from example.com to the IP address of the CloudFront distribution.

- The example CloudFront distribution returns the web page from cache or origin server for example.com to the web browser.



When you register a domain with route 53

- The service automatically makes itself the DNS service for the domain by doing the following-
- It creates that has the same name as your domain.
- It assign a set of four name servers to the hosted zone unique to the account.
- When someone use a browser to access you are website this name servers inform the browser when to find your resources such as a web server or an Amazon S3 bucket.
- It gets the name servers from the hosted zone and adds them to the domain.

AWS supports

- Generic top level domains
- Geographic top level domains

Registering a domain with route 53

- You can register a domain with route 53 if the TLD is included on the supported TLD list.
- If the TLD is not included you can't register the domain with Route-53.

Using route 53 as your service

- You can use route 53 as the DNS service for any domain even if they TLD for the domains is not included on the supported TLD list.

Note:- each Amazon route 53 account is limited to maximum of 500 hosted zones and 10000 resource record sets per hosted zone you can increase this limit by requesting to AWS.

Steps to configure Route-53

- This domain can be Route 53, or another DNS register but then you connect you are domain name in that register to Route 53.
- Create hosted zone on Route 53, this is clone automatically if you registered your domain using Route 53.
- Inside the hosted zone you need to create record sets.
- Delegate to Route 53
- This step connects everything and make it works.
- Connect the domain name to the route 53 hosted zone this is called delegation.
- Update you are domain register with the correct name servers for your route 53 hosted zone.
- No other customer hosted zone will share the delegation set with you.
- Doing this means route 53 DNS service will be serving DNS traffic for the domain of the hosted zone.
- If you register you are domain with different registrar, you need to configure the route 53 NS service list in your registrar DNS database for your domain
- If you are using another domain provider and you did all the changes
- When you migrate from one DNS provider to another for an existing domain this change can take up to 48 hours to be effective.
- This is because name server DNS records are typically catches across the DNS system globally on the internet for up to 48 hours TTL periods.

Transferring domain to route 53

- You can transfer a domain to route 53 if the TLD is included on the following list
- If the TLD is not included you can't transfer the domain to route 53.
- For most TLD you need to get an authorization code from the current registrar to transfer a domain.

- When you create hosted zone Amazon Route 53 automatically creates a name server (NS) records and start of authority records (SOA) for the zone.
- The NS records identifies the four names servers that you give your register or you are DNS service so that DNS Queries are routed to route 53 name servers.
- By default route 53 assign a unique set of four name servers (known collectively as a delegation set) to each hosted zone that you create....
- E.g. : ns-1337 awsdns-39.com

Route 53 as Your Authoritative DNS

- Once you update the route 53 NS setting with you are domain register to include the route 53 name servers route 53 will be responsible to respond to DNS queries for the hosted zone.
- This is true whether you do have a functioning website or not.
- Route 53 will respond with information about the hosted zone whenever someone types the associated domain name in a web browser.
- You can create more than one hosted zone with the same name and add different records to each hosted zone.
- Route 53 assigns four name servers to every hosted zone.
- The name servers are different for each of them.
- When you update you are registers name server records be careful to use the route 53 name servers for the correct hosted zone the one that contains the records that you want route 53 to use when responding to queries for your domains.
- Route 53 never returns value for records in other hosted zones that have the same zone.

Route 53 Hosted zone default Entries–

- Inside the hosted zone by default you have two entries.
- NS entry:- Contains the unique Sets of name servers for this Hosted zone.
- SOA entry:- Contains information about the hosted zone.
- If you are currently using another DNS service and you want to migrate to Amazon route 53.
 - Start by creating a hosted zone a doubt 53 automatically a sign the delegation sets, the four name servers.
 - To ensure that the DNS routes queries for you are domain to the route 53 name servers.
 - Update your registers on you are DNS service NS records for the domain to replace the current name servers with the names of the four route 53 name servers for your hosted zone.
- The method that you use to update the NS records depends on which register or DNS service you are using.
- Some register only allow you to specify name servers using IP address they don't allow you to specify fully qualified domain names
- If you are register requires using IP address you can get the IP address for your name servers using the DIG utility for Mac Linux and NS look up for Windows.

Transferring a domain between accounts within AWS-

1. Transferring a domain to different AWS account

- If you registered a domain using one AWS account and you want to transfer the domain to another AWS account you can do so by contacting the AWS support centre and requesting the transfer.
- Migrating a hosted zone to different account.
- If you are using route 53 as the day and S service for the domain route 53 does not transfer the hosted zone when you transfer a domain to a different AWS account
- If domain registration is associated with one account and the correspondence hosted zone is associated with another account neither domain registration nor DNS functionality is affected
- The only effect is that you will need to sign into the route 53 console using one account to see the domain and signed in using the other account to see the hosted zone.

2. Support DNS record types by route 53-

- A Record- Address Record – Maps domain name to IP address www.ygminds.com IN A 5:5:5:5
- AAAA Record- IPv6 address record Maps domain name to an IPv6 address www.ygminds.com IN AAAA 2002:b768::1
- CNAME Record- Maps an alias to a host name Web IN CNAME www.ygminds.com
- NS Records- Name server record used for delegating zone to a name server ygminds.com IN ns1 ygminds.com
- SOA Records- Start of Authority Record
- MX Records- Mail exchange – defines where to deliver mail for user @ a domain name.
- NS records defines which name server is authoritative to a particular zone or domain name and point you to other DNS servers-
- A/AAAA are called host records, like business cards.
- CNAME is an alternative record or an alias for another record
- Helpful in direction or if you want to hide details about you are actual servers from the users

Start of Authority Records (SOA)

- Every single zone has won and only one so a resource record at the beginning of the zone.
- It is not an actual record it includes the following information.
- Who the owner is email for the domain.
- The authoritative server
- The serial number which is incremental with changes to the zone data.
- The refreshing time/cycle into and the TTL.

C NAME Record types–

- CNAME value element is the same format as a domain name.
- The DNS protocol does not allow you to create a CNAME record for the top not of a DNS namespace also known as the zone Apex or (root domain).
- You cannot create a CNAME Record for ygminds.com
- However you can create CNAME records for www.ygminds.com support ygminds.com and so on
- In addition if you create a CNAME record for a subdomain you cannot create any other records for that sub domain.

Routing Policies:

- Simple routing policy
- Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website. You can use simple routing to create records in a private hosted zone.

Failover routing policy:

- Use when you want to configure active-passive failover. You can use failover routing to create records in a private hosted zone.
- Fail over routing lets you route traffic to a resource when the resource is healthy if the main resource is not healthy then route traffic to a different resource.
- The primary and secondary records can row traffic to anything from an Amazon S3 bucket that is configured as a website to a complex tree of records.
- Failover Routing policy is applicable for public hosted zone only.

Geolocation routing policy :

- Use when you want to route traffic based on the location of your users. You can use geolocation routing to create records in a private hosted zone.
- Geo location routing lets you choose the resources that servers you are traffic based on the geographic location of you are users i.e. the location that DNS queries originate from.
- For e.g. You may have person in Europe and Asia now you want users in the Asia to be served in the Asia and those in Europe to be served by servers in Europe.

Geo-proximity routing policy :

- Use when you want to route traffic best on the location of your resources and optionally shift traffic from resources in one location to resources in another.
- You can also optionally choose to route more traffic or less to a given resource by specifying a value known as ABI as BIA s expand or shrinks the size of Geographic region from which traffic is routed to a resource
- **Benefits:**

- You can localize your content and present some or all of your website in the language of your users.
- You can also use Geo location routing to restrict distribution of content to only the locations in which you have distribution rights.
- You can specify Geographic locations by continent by country or by state in the United States.
- If you create separate records for overlapping Geographic regions for example one record for North America and one for Canada priority goes to the smallest Geographic region.
- Geolocation Works By mapping IP address to locations however some IP addresses are not mapped to geographic location

Latency routing policy

- Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency. You can use latency routing to create records in a private hosted zone.
- If your application is hosted in multiple Amazon regions you can improve performance for your users by serving their request from the Amazon EC2 region that provides the lowest latency.
- To use latency based routing you create latency records for your resources in multiple EC2 regions.
- When Amazon Route 53 receives a DNS query for your domain or subdomain.
- It determines which Amazon region you have created latency records for.
- Determine which regions give the lowest latency to users.
- Then select a latency record for that region, For example - suppose you have ELB in US East and in Asia Pacific Mumbai region.
- You created a latency record for each Load Balancer
- Here's what happens when a user in London enters the name of your domain in a browser.
- DNS routes the request to a Route 53 name server
- Route 53 refers to its data on latency between London and the Mumbai region and between London and the North Virginia.
- Latency is lower between London and North Virginia, Route 53 responds to the query with the IP address for the North Virginia LB.

IP-based routing policy

- Use when you want to route traffic based on the location of your users, and have the IP addresses that the traffic originates from.

Multivalue answer routing policy

- Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random. You can use multivalue answer routing to create records in a private hosted zone.
- Multi value answer routing lets you configure Amazon route 53 to return multiple values such as IP addresses for your web servers in response to DNS queries you can specify multiple values for almost any record but multi value answer routing also lets you check the health of each resource so route 53 returns only values for healthy resources it is substitute for A Load Balancer.
- But the ability to return multiple health check cable IP addresses is a way to use DNS to improve availability and load balancing.

Weighted routing policy

- Use to route traffic to multiple resources in proportions that you specify. You can use weighted routing to create records in a private hosted zone.
- Weighted routing policy lets you associate multiple resources with a single domain name or subdomain name and choose how much traffic is routed to each resource.
- This can be useful for a variety of purposes including load balancing and testing new versions of software.
- Weighted can be assigned any number from 1 to 255.
- Weighted routing policy can be applied when there are multiple resource that perform the same function for example web server serving the same website.
- To configure weighted routing you create records that have the same name and type for each of your resource.
For example suppose for www.tz.com has three resource record sets with weight of 1 (20%) and 3 (60%)(sum=5)
- On average route 53 select each of the list two resource records set on 5th of the line and returns the third resource record set three-fifth of the time.

- Benefits of Route 53



15. RDS

What is Data?

In simple words data can be facts related to any object for example – your age, job, house number, contact number, name, place or some data related to you.

What is Database?

Database is systematic Collection of data database support storage and manipulation of data.

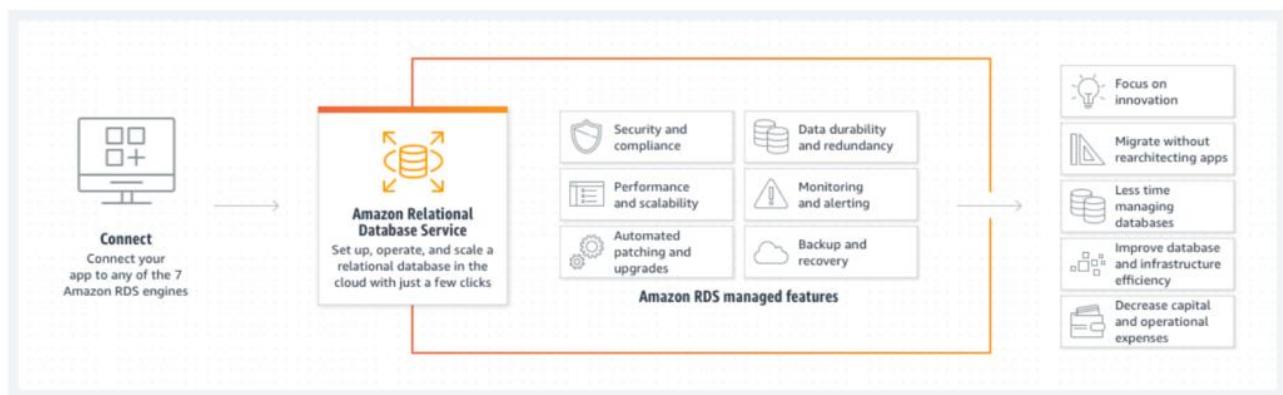
For example Facebook telecom companies amazon.com

What is DBMS?

DBMS is a collection of programs which enable its user to access database manipulate data reporting or representation of data.

Types of DBMS

1. Hierarchical
2. Network
3. Relational
4. Object oriented



Relational Database

1. A relational database is a data structure that allows you to link information from different tables in different types of data buckets.
2. Tables are related to each other.
3. All fields must be filled.
4. Best Suited for OLTP (online transaction processing).

5. Relational DB – MY SQL, Oracle DBMS, IBM DB2.

- A role of a table is also called record it contents the specific information of each individual entry in the table.
 - Each table has its own primary key
 - A schema is used to strictly defines tables column index and relation between tables.
1. Relational DB are usually used in Enterprises application or scenario.
 - Exception in MY SQL which is used for web application

Common application for MY SQL include PHP and Java based web applications that requires a database storage backend for example JOOMLA.

1. Cannot Scale – Out Horizontally.
2. Virtually all Relational DB uses SQL.

In a **relational database**, data is stored in a way that relates it to other pieces of data.

An example of a relational database might be the coffee shop's inventory management system. Each record in the database would include data for a single item, such as product name, size, price, and so on.

Relational databases uses **structured query language (SQL)**to store and query data. This approach allows data to be stored in an easily understandable, consistent, and scalable way. For example, the coffee shop owners can write a SQL query to identify all the customers whose most frequently purchased drink is a medium latte.

Example of data in a relational database:

ID	Product name	Size	Price
1	Medium roast ground coffee	12 oz.	\$5.30
2	Dark roast ground coffee	20 oz.	\$9.27

Amazon Relational Database Service

[**Amazon Relational Database Service \(Amazon RDS\)**](#)is a service that enables you to run relational databases in the AWS Cloud.

Amazon RDS is a managed service that automates tasks such as hardware provisioning, database setup, patching, and backups. With these capabilities, you can spend less time completing administrative tasks and more time using data to innovate your applications. You can integrate Amazon RDS with other services to fulfil your business and operational needs, such as using AWS Lambda to query your database from a serverless application.

Amazon RDS provides a number of different security options. Many Amazon RDS database engines offer encryption at rest (protecting data while it is stored) and encryption in transit (protecting data while it is being sent and received).

Amazon RDS database engines

Amazon RDS is available on six database engines, which optimize for memory, performance, or input/output (I/O). Supported database engines include:

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle Database
- Microsoft SQL Server

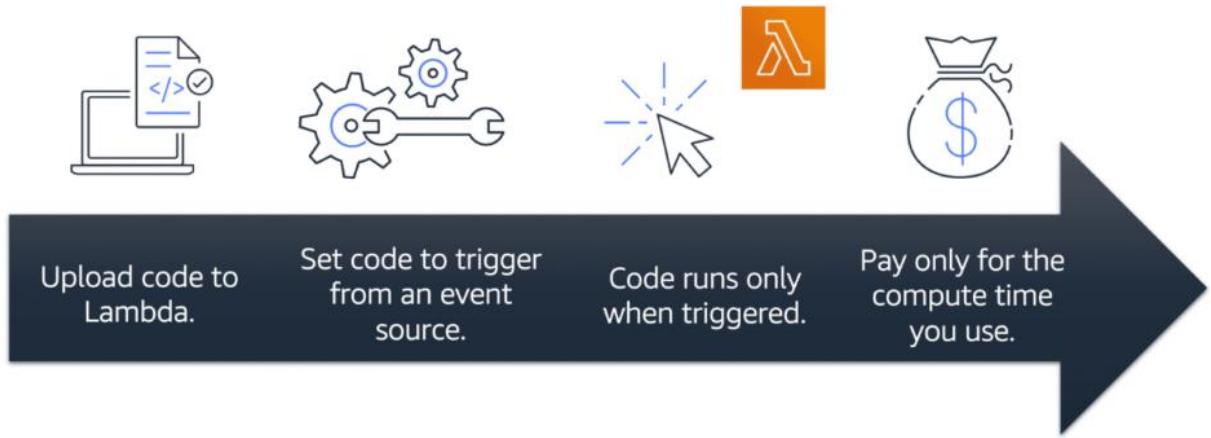
16. Lambda

[**AWS Lambda**](#) is a service that lets you run code without needing to provision or manage servers.

1. AWS Lambda is compute service that let's you run code without provisioning or managing servers.
 2. With AWS lambda, you can run code for virtually any type of application or back end service all with zero administration.
- AWS Lambda manages all the administration it manages
 - Provisioning and capacity of the computer split that offers a balance of memory.
 - Server and O S maintenance.
 - High availability and automatic scaling.
 - Monitoring fleet health
 - Applying Security patches
 - Deploying your Code.
 - Monitoring and logging your Lambda Functions
 - AWS Lambda Runs your Code on a high-availability Compute Infrastructure.

While using AWS Lambda, you pay only for the compute time that you consume. Charges apply only when your code is running. You can also run code for virtually any type of application or backend service, all with zero administration.

For example, a simple Lambda function might involve automatically resizing uploaded images to the AWS Cloud. In this case, the function triggers when uploading a new image.



- AWS Lambda runs your code on a high availability compute infrastructure.
- AWS Lambda executes your code only when needed and scales automatically from a few requests per day to thousands per second.
- You pay only for the compute time you consume – No charge when your code is not running.
- All you need to do is supply your code in the form of one or more Lambda functions to AWS Lambda in one of the languages that AWS Lambda supports (currently Node.js, Java, PowerShell, Ruby, C#, Python, and Go) and the service can run the code on your behalf.
- Typically the life cycle for an AWS Lambda based application includes authoring code, deploying code to AWS Lambda and then monitoring and troubleshooting..
- This is in exchange for flexibility which means you cannot log into computer instances or customize the operating system or language run time.
- If you do want to manage your own compute, you can use EC2 or elastic Beanstalk.
- AWS Lambda runs your code on a high availability compute infrastructure.
- AWS Lambda executes your code only when needed and scales automatically from a few requests per day to thousands per second.
- You pay only for the compute time you consume – No charge when your code is not running.
- All you need to do is supply your code in the form of one or more Lambda functions to AWS Lambda in one of the languages that AWS Lambda supports (currently Node.js, Java, PowerShell, Ruby, C#, Python, and Go) and the service can run the code on your behalf.
- Typically the life cycle for an AWS Lambda based application includes authoring code, deploying code to AWS Lambda and then monitoring and troubleshooting..

- This is in exchange for flexibility which means you cannot log into computer instances or customized the operating system or language run time.
- If you do want to manage your own compute, you can use EC2 or elastic Beanstalk.

How AWS Lambda works

1. You upload your code to Lambda.
2. You set your code to trigger from an event source, such as AWS services, mobile applications, or HTTP endpoints.
3. Lambda runs your code only when triggered.
4. You pay only for the compute time that you use. In the previous example of resizing images, you would pay only for the compute time that you use when uploading new images. Uploading the images triggers Lambda to run code for the image resizing function.

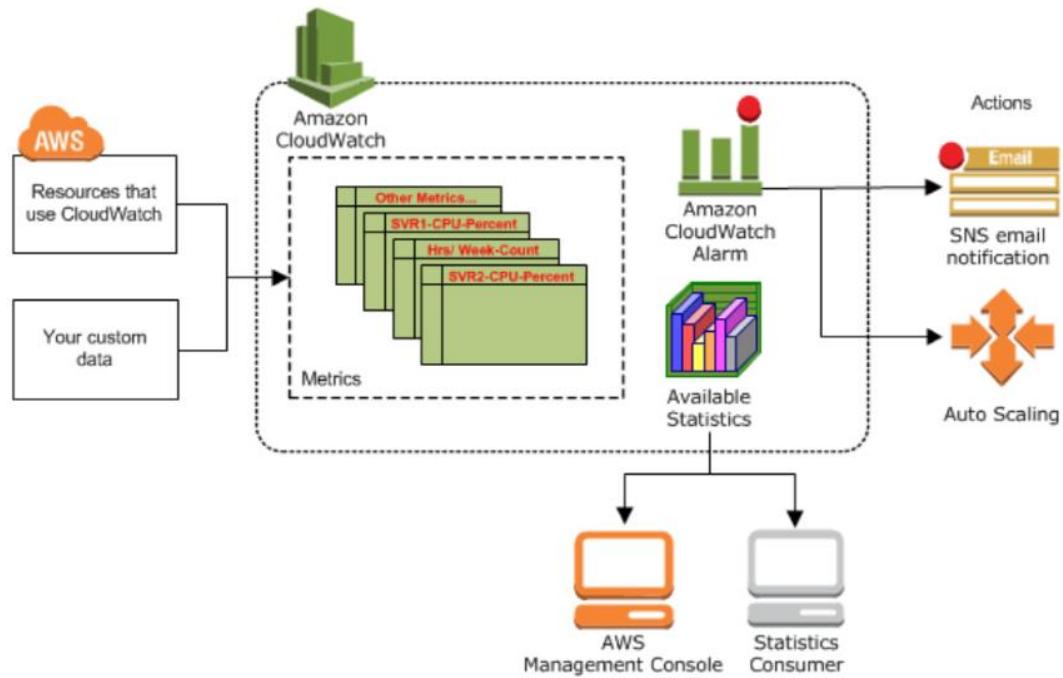
When Lambda Triggers-

- You can use AWS to run your Code in response to-
- Events such as changes to data in an Amazon S3 bucket or and Amazon dynamo DB table.
- To run your code in response to http request using Amazon API gateway.
- With this capabilities you can use lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon dynamo DB, process streaming data stored in Kinesis or create your own bacon that separates at AWS scale performance and security.

Important Terms used in Lambda-

1. **Function-** A function is a resource that you can invoke to run your code in a AWS Lambda. A function has code that processes events and a run time that passes request and responses between Lambda and the function code.
2. **Runtime-** Lambda allows functions in different languages to run in the same execution environment. The runtime sits in between the Lambda service and your function code, relaying context information and responses between the two.
3. **Event-** Is a JSON-formatted document that contains data for a function to process.
4. **Event Source/ Trigger-** An AWS service such as Amazon SNS or a custom service that triggers your functions and executes its logic.
5. **Downstream Resource-** An AWS service, such as dynamo DB tables or S3 buckets, that your Lambda function calls once it is triggered.
6. **Concurrency-** Number of requests your function is serving in any given time.

17. CloudWatch



Amazon CloudWatch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources.

CloudWatch provides you with data and actionable insights to monitor your applications, respond to system-wide performance changes, and optimize resource utilization. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. You get a unified view of operational health and gain complete visibility of your AWS resources, applications, and services running on AWS and on-premises. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.

Benefits of CloudWatch:

1. Use a single platform for observability –
 - Modern applications, such as those running on microservices architectures, generate large volumes of data in the form of metrics, logs, and events.
 - Amazon CloudWatch allows you to collect, access, and correlate this data on a single platform from across all your AWS resources, applications, and services running on

AWS and on-premises, helping you break down data silos to gain system-wide visibility and quickly resolve issues.

2. Collect metrics on AWS and on premises–

- Monitoring your AWS resources and applications is easy with CloudWatch. It natively integrates with more than 70 AWS services, such as Amazon EC2, Amazon DynamoDB, Amazon S3, Amazon ECS, Amazon EKS, and AWS Lambda.
- It automatically publishes detailed one-minute metrics and custom metrics with up to one-second granularity so you can dive deep into your logs for additional context. You can also use CloudWatch in hybrid environments by using the CloudWatch Agent or API to monitor your on-premises resources.

3. Collect metrics on AWS and on premises–

- Set alarms and automate actions based on predefined thresholds or on machine learning (ML) algorithms that identify anomalous behavior in your metrics.
- For example, you can start Amazon EC2 Auto Scaling automatically or stop an instance to reduce billing overages. You can also use CloudWatch Events for serverless to trigger workflows with services like AWS Lambda, Amazon SNS, and AWS CloudFormation.

4. Get operational visibility and insight–

- To optimize performance and resource utilization, you need a unified operational view, real-time granular data, and historical reference.
- CloudWatch provides automatic dashboards, data with one-second granularity, and up to 15 months of metrics storage and retention.
- You can also perform metric math on your data to derive operational and utilization insights; for example, you can aggregate usage across an entire fleet of EC2 instances.

5. Derive actionable insights from logs–

- Explore, analyze, and visualize your logs so you can troubleshoot operational problems with ease. With CloudWatch Logs Insights, you pay only for the queries you run.
- It scales with your log volume and query complexity, giving you answers in seconds.
- In addition, you can publish log-based metrics, create alarms, and correlate logs and metrics together in CloudWatch Dashboards for complete operational visibility.

How it works

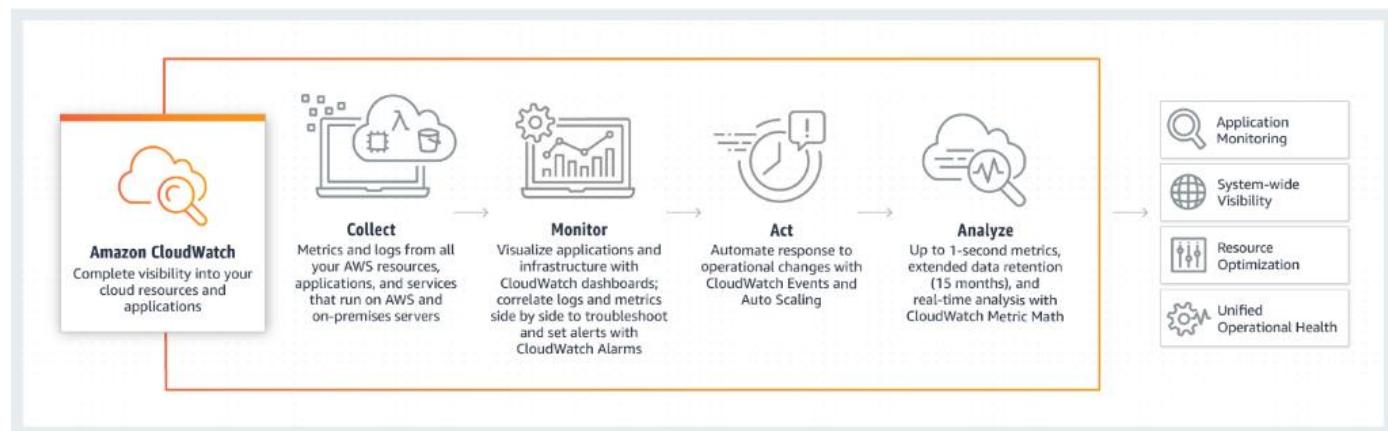
CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, and visualizes it using automated dashboards so you can get a unified view of your AWS resources, applications, and services that run on AWS and on premises. You can visualize the experience of your application end users and validate design choices through experimentation. Correlate your metrics and logs to better understand the health and performance of your resources. Create alarms based on metric value thresholds you specify, or alarms that can watch

for anomalous metric behavior based on ML algorithms. For example, set up automated actions to notify you if an alarm is triggered and automatically start auto scaling to help reduce mean time to resolution (MTTR).

You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications, and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money.



Use cases-

1. Monitor and troubleshoot infrastructure

- Monitor key metrics and logs, visualize your application and infrastructure stack, create alarms, and correlate data to understand and resolve the root cause of performance issues in your AWS resources.
- This includes monitoring your container ecosystem across Amazon ECS, AWS Fargate, Amazon EKS, and Kubernetes.

2. Improve mean time to resolution

- Correlate, visualize, and analyze metrics and logs so you can resolve issues quickly, and combine them with trace data from AWS X-Ray for full observability.
- You can also analyze user requests to speed up troubleshooting and debugging, and reduce overall MTTR.

3. Optimize resources proactively

- CloudWatch alarms watch your metric values against thresholds that you specify or that it creates using ML models to detect anomalous behavior.
- If an alarm is triggered, CloudWatch can act automatically to enable Amazon EC2 Auto Scaling or stop an instance, so you can automate capacity and resource planning.

4. Monitor applications

- Monitor your end user's digital experience and your applications that run on AWS (on Amazon EC2, containers, and serverless) and on-premises.
- CloudWatch collects data at every layer of the performance stack, from your frontend to your infrastructure.
- You can use ServiceLens to identify performance bottlenecks in your applications and isolate them using the correlated metrics, logs, and traces. Add canaries for SLA/SLO monitoring of endpoints and UI workflows.
- Collect client-side data on application performance in near real time to identify and debug issues that impact end users. Experiment with features across the full application stack, measure against performance and business metrics, and launch features safely.

5. Use observability analytics

- Analyze millions of operational logs and metrics in near real time to identify trends and patterns in your application performance, and use these insights to reduce MTTR.
- Use fast and interactive operational queries to create powerful visualizations, helping you monitor and pinpoint issues quickly.

The following terminology and concepts are central to your understanding and use of Amazon CloudWatch:

- **Namespaces** – A *namespace* is a container for CloudWatch metrics. Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.
- **Metrics** – *Metrics* are the fundamental concept in CloudWatch. A metric represents a time-ordered set of data points that are published to CloudWatch.

- Metrics exist only in the Region in which they are created. Metrics cannot be deleted, but they automatically expire after 15 months if no new data is published to them.
- **Dimensions** – A *dimension* is a name/value pair that is part of the identity of a metric. You can assign up to 30 dimensions to a metric.
 - Every metric has specific characteristics that describe it, and you can think of dimensions as categories for those characteristics. Dimensions help you design a structure for your statistics plan.
- **Resolution** – Each metric is one of the following:
 - **Standard resolution**, with data having a one-minute granularity
 - **High resolution**, with data at a granularity of one second
 - Metrics produced by AWS services are standard resolution by default. When you publish a custom metric, you can define it as either standard resolution or high resolution.
- **Statistics** – *Statistics* are metric data aggregations over specified periods of time. CloudWatch provides statistics based on the metric data points provided by your custom data or provided by other AWS services to CloudWatch.
- **Percentiles** – A *percentile* indicates the relative standing of a value in a dataset. For example, the 95th percentile means that 95 percent of the data is lower than this value and 5 percent of the data is higher than this value. Percentiles help you get a better understanding of the distribution of your metric data.
- **Alarms** – You can use an *alarm* to automatically initiate actions on your behalf. An alarm watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time. The action is a notification sent to an Amazon SNS topic or an Auto Scaling policy. You can also add alarms to dashboards.

18. Simple Notification Service (SNS)

Example:

Messages are sent into the queue by Application A and they are processed by Application B. If Application B fails, Application A doesn't experience any disruption. Messages being sent can still be sent to the queue and will remain there until they are eventually processed.

This is loosely coupled. This is what we strive to achieve with architectures on AWS. And this brings me to two AWS services that can assist in this regard. Amazon Simple Queue Service or SQS and Amazon Simple Notification Service or SNS. But before I dive into those two services, let me just order a to-go coffee on our cafe website. Done. All right, well, that's great. I should get a message when that order is ready.

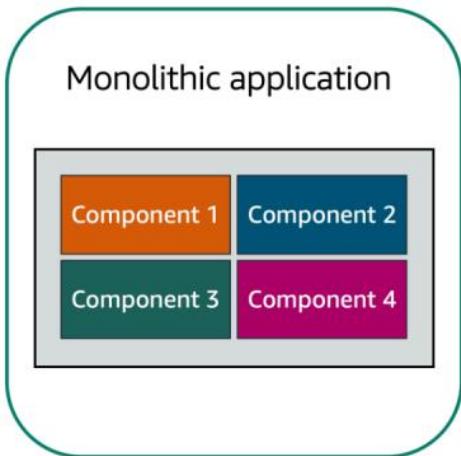
First up, let's discuss Amazon SQS. SQS allows you to send, store, and receive messages between software components at any volume. This is without losing messages or requiring other services to be available. Think of messages as our coffee orders and the order board as an SQS queue. Messages have the person's name, coffee order, and time they ordered. The data contained within a message is called a payload, and it's protected until delivery. SQS queues are where messages are placed until they are processed. And AWS manages the underlying infrastructure for you to host those queues. These scale automatically, are reliable, and are easy to configure and use.

Now, Amazon SNS is similar in that it is used to send out messages to services, but it can also send out notifications to end users. It does this in a different way called a publish/subscribe or pub/sub model. This means that you can create something called an SNS topic which is just a channel for messages to be delivered. You then configure subscribers to that topic and finally publish messages for those subscribers. In practice, that means you can send one message to a topic which will then fan out to all the subscribers in a single go. These subscribers can also be endpoints such as SQS queues, AWS Lambda functions, and HTTPS or HTTP web hooks.

Additionally, SNS can be used to fan out notifications to end users using mobile push, SMS, and email. Taking this back to our coffee shop, we could send out a notification when a customer's order is ready. This could be a simple SMS to let them know to pick it up or even a mobile push.

In fact, it looks like my phone just received a message. Looks like my order is ready. See you soon.

Monolithic applications and microservices

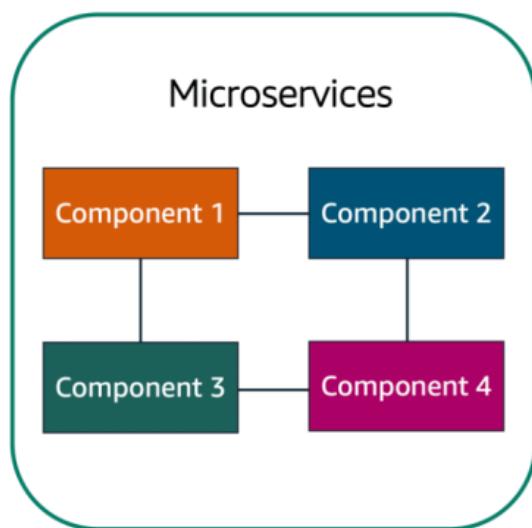


Applications are made of multiple components. The components communicate with each other to transmit data, fulfill requests, and keep the application running.

Suppose that you have an application with tightly coupled components. These components might include databases, servers, the user interface, business logic, and so on. This type of architecture can be considered a **monolithic application**.

In this approach to application architecture, if a single component fails, other components fail, and possibly the entire application fails.

To help maintain application availability when a single component fails, you can design your application through a **microservices** approach.



In a microservices approach, application components are loosely coupled. In this case, if a single component fails, the other components continue to work because they are communicating with each other. The loose coupling prevents the entire application from failing.

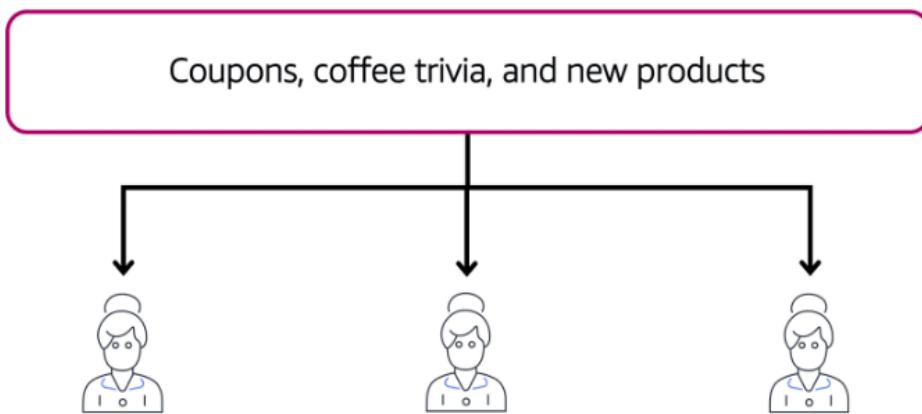
When designing applications on AWS, you can take a microservices approach with services and components that fulfill different functions. Two services facilitate application integration: Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).

Amazon Simple Notification Service (Amazon SNS)

Amazon Simple Notification Service (Amazon SNS) is a publish/subscribe service. Using Amazon SNS topics, a publisher publishes messages to subscribers. This is similar to the coffee shop; the cashier provides coffee orders to the barista who makes the drinks.

Step 1

Publishing updates from a single topic

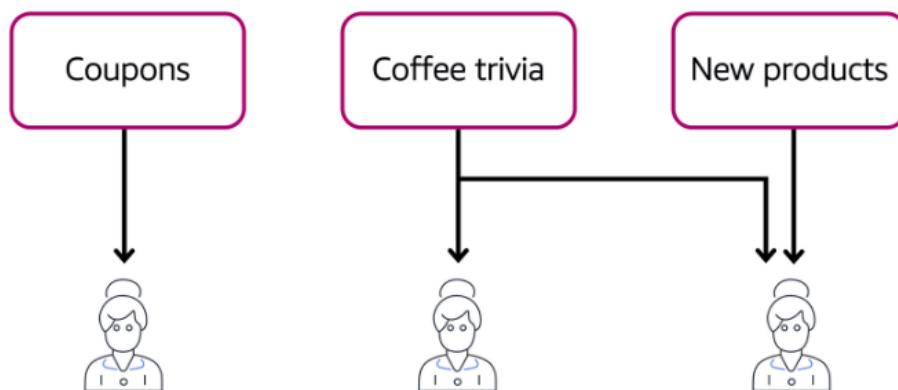


Suppose that the coffee shop has a single newsletter that includes updates from all areas of its business. It includes topics such as coupons, coffee trivia, and new products. All of these topics are grouped because this is a single newsletter. All customers who subscribe to the newsletter receive updates about coupons, coffee trivia, and new products.

After a while, some customers express that they would prefer to receive separate newsletters for only the specific topics that interest them. The coffee shop owners decide to try this approach.

Step 2

Publishing updates from multiple topics



Now, instead of having a single newsletter for all topics, the coffee shop has broken it up into three separate newsletters. Each newsletter is devoted to a specific topic: coupons, coffee trivia, and new products.

Subscribers will now receive updates immediately for only the specific topics to which they have subscribed.

It is possible for subscribers to subscribe to a single topic or to multiple topics. For example, the first customer subscribes to only the coupons topic, and the second subscriber subscribes to only the coffee trivia topic. The third customer subscribes to both the coffee trivia and new products topics.

DevOps (9)

1. Source code management

- Source control refers to tracking and managing changes to code.
- This ensures that developers are always working on the right version of source code.

- Every development team needs a good way to manage changes and version code in their codebases

That's why they use source control tools.

- Source control management (SCM) refers to tools that help you keep track of your code with a complete history of changes.
- Source code management tool (SCM) tracks changes to a source code repository.
- SCM also maintains a history of changes.
- This is used to resolve conflicts when merging updates from multiple developers.

Why Source Control Is Important?

- When multiple developers are working within a shared codebase it is a common occurrence to make edits to a shared piece of code.
- Separate developers may be working on a seemingly isolated feature, however this feature may use a shared code module.
- Therefore developer 1 working on Feature 1 could make some edits and find out later that Developer 2 working on Feature 2 has conflicting edits.
- Before the adoption of SCM this was a nightmare scenario.
- Developers would edit text files directly and move them around to remote locations using FTP or other protocols. Developer 1 would make edits and Developer 2 would unknowingly save over Developer 1's work and wipe out the changes.

The benefits of source code management

- Collaborative code development a more user friendly experience.
- Historical record can then be used to 'undo' changes to the codebase.
- A clean and maintained SCM history log can be used interchangeably as release notes. This offers insight and transparency into the progress of a project that can be shared with end users or non-development teams.
- SCM will reduce a team's communication overhead and increase release velocity.
- With SCM developers can work independently on separate branches of feature development, eventually merging them together.
- The most IMP benefit Is Version control.
- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.
- Using a VCS also generally means that if you screw things up or lose files, you can easily recover.
- Here version control is also called as source control its one and the same thing. SCM or VCM is one and the same.

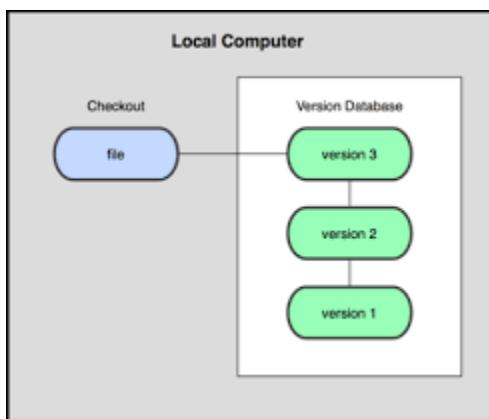
Version control-

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.
- Using a VCS also generally means that if you screw things up or lose files, you can easily recover.
- Here version control is also called as source control its one and the same thing. SCM or VCM is one and the same.

Types of Version control system (VCS)-

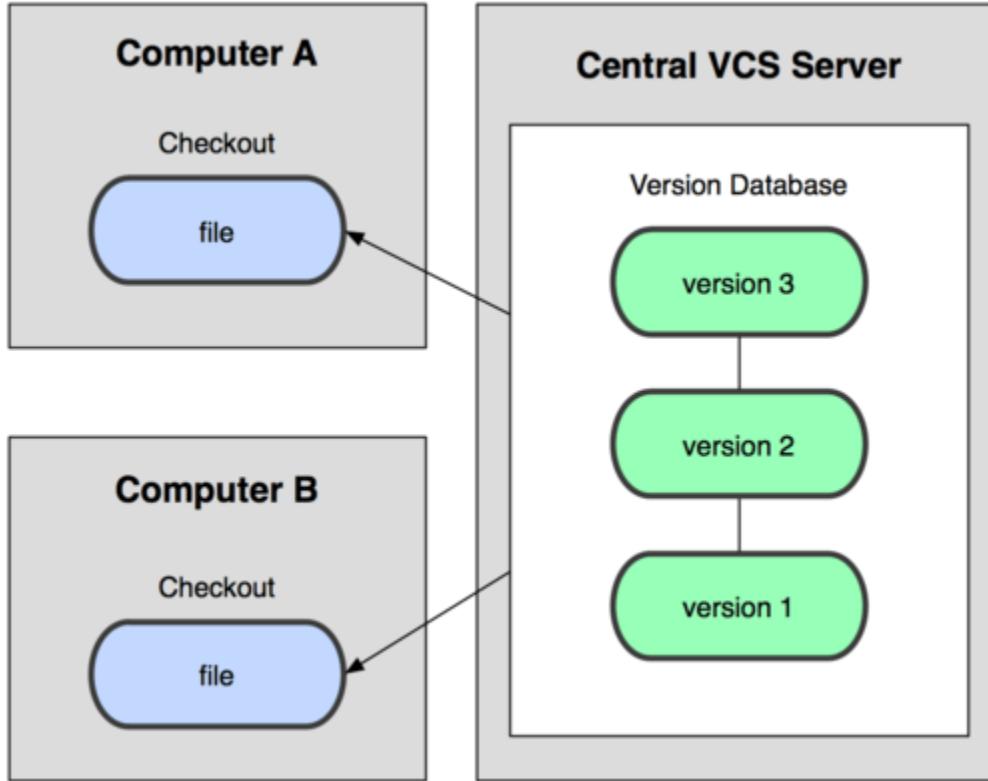
- Local VCS
- Centralized VCS
- Distributed VCS

Local VCS-



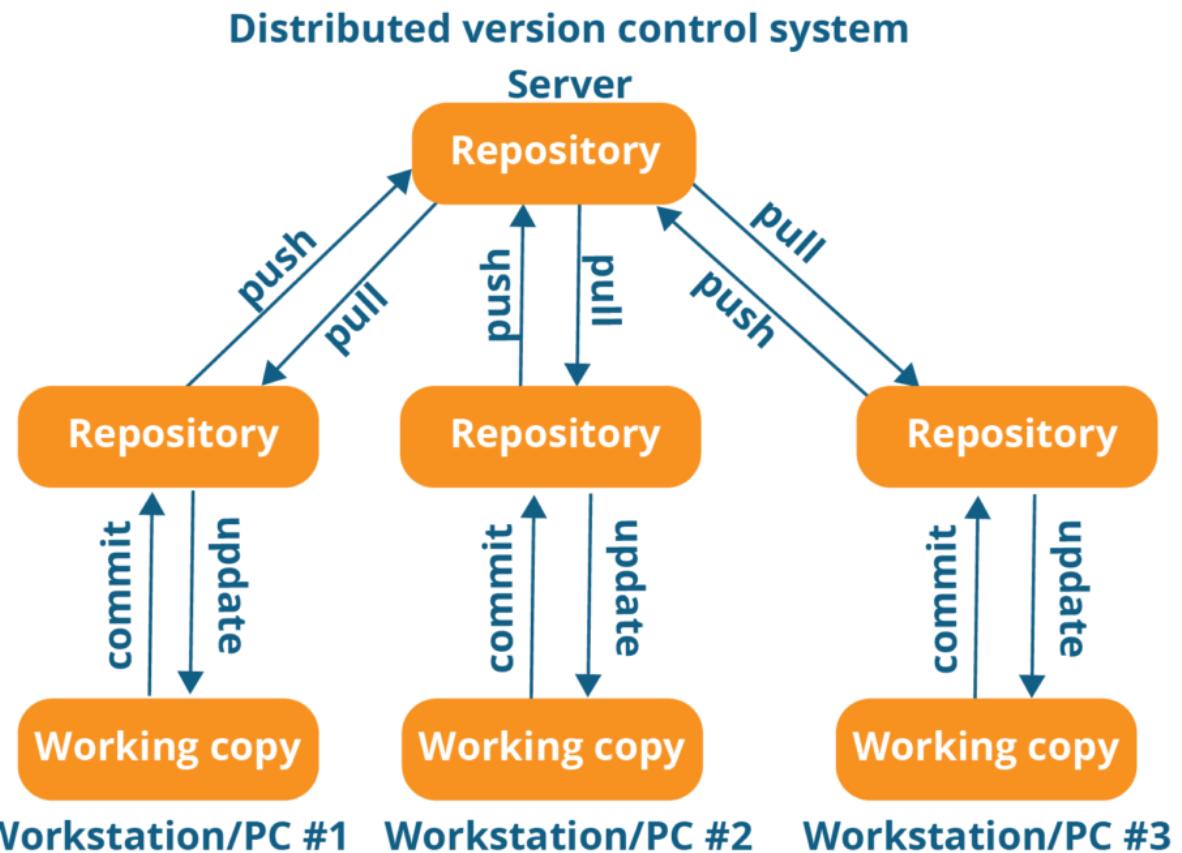
- Many people's version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if they're clever).
- This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.
- To deal with this issue, programmers long ago developed local VCSs that had a simple database that kept all the changes to files under revision control.
- One of the most popular VCS tools was a system called RCS (Revision control system).(1982).

Centralized VCS-



- The next major issue that people encounter is that they need to collaborate with developers on other systems.
- To deal with this problem, Centralized Version Control Systems (CVCSs) were developed.
- These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place.
- For many years, this has been the standard for version control.
- This setup offers many advantages, especially over local VCSs. For example, everyone knows to a certain degree what everyone else on the project is doing.
- However, this setup also has some serious downsides. The most obvious is the single point of failure that the centralized server represents.
- If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on.
- If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines.

Distributed Version control System-



- This is where Distributed Version Control Systems (DVCSs) step in. In a DVCS clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.
- Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it.
- Every clone is really a full backup of all the data.
- DVCS system are working fine on offline mode as a client copies the entire repository on their local machine
- If DVCS server is down developer can work using their local copies
- Furthermore, many of these systems deal pretty well with having several remote repositories they can work with, so you can collaborate with different groups of people in different ways simultaneously within the same project.

GIT –

Stages of Git/ Workflow

1. Workspaces/ Working directory
2. Staging area
3. Local Repo

Repository

- Repository is a place where you have all your code or kind of folder on server
- It is a kind of folder related to one product
- Changes are personal to that particular repository

Server

- It stores all repositories
- It contains metadata also

Working directory

- Where you see files physically and do modification
- At a time you can work on particular branch
- In another CVCS, developers generally makes modification and commit they are changes directly to the repository but Git uses a different strategy. Git does not track each and every modified file whenever you do come it and operation. Git Looks for the files present in the Staging area only those files present in the Staging area are considered for commit and not all the modified files.
- Working directory
- Staging area
- Local Repository

Commit-ID / Version-ID/ Version

- Reference to identify each change
- To identify Who changed the file

Tags

- Tax assign a meaningful name with a specific version in the repository once a tag is created for a particular save even if you create a new commit it will not be updated.

Snapshots

- Represent some data of particular time
- It is always incremental I.e. it stores the changes (appended data) only not entire copy

Commit

- Store changes in repository you will get one commit ID
- It is 40 alphanumeric characters
- It uses SHA-1 check sum concept
- Even if you change one dot comet ID will get change
- It actually helps you to track the changes
- Commit is also named as SHA1 hash

Push

- Push operations copies changes from a local repository instances to a remote or central Repo this is used to store the changes permanently into the GIT repository

Pull

- Full operation copies the changes from a remote repository to a local machine the pull operation is used for synchronisation between two Repo.

Branch

- Product is same so one repository but different track
- Each task has one separate branch
- Finally merge code all branches
- Useful when you want to work parallel
- Can create one branch on the basis of another branch
- Changes are personal to that particular branch
- Default branch is Master
- File created in work space will be visible in any of the branch workspace until you commit once you come it then that file belongs to the particular branch.

Advantages of Git

- Free and open source
- Fast and small as most of the operations are performed locally therefor it is fast
- Security git uses of common Cryptography has function called secure has function SHA1 to name add identify objects within its database
- No need of powerful hardware
- Easier branching if we create a new branch it will copy all the codes to the new branch

Types of Repositories

- Bare Repositories (Central Repo)
- Store and share only
- All Central repositories are Bare Repo
- Non-Bare Repositories (local repo)

- Where you can modify the files
- All local repositories are non-bare repositories

How to install git on EC2

```
# sudo su  
  
# yum update-y  
  
# yum install git -y  
  
# git --version  
  
# git config --global user.name  
  
# git config --global user.email
```

How to Commit, push & pull from github

- Create one directory and go inside it

```
#git init  
  
-Touch my file (put some data)  
  
#git status  
  
#git add .  
  
#git commit -m "My first commit"  
  
#git status  
  
#git log  
  
#git show <commit-id>  
  
#git remote add origin <central git url>  
  
#git push -u origin master (enter username & password)
```

To ignore some files while committing

- Create one hidden file .gitignore and enter file format which you want to ignore

For eg. #vim .gitignore

Enter *.css

```
#git add .gitignore
```

```
#git commit -m "latest update exclude css"
```

```
#git status
```

- Create some text, Java and CSS files and add them by running “git add”

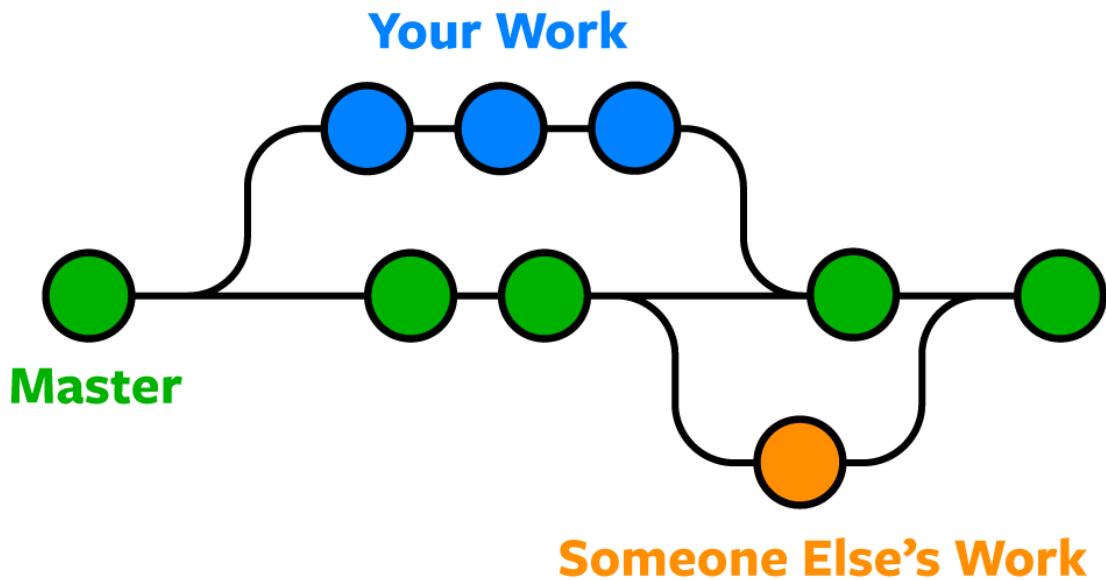
```
# ls
```

```
# git status
```

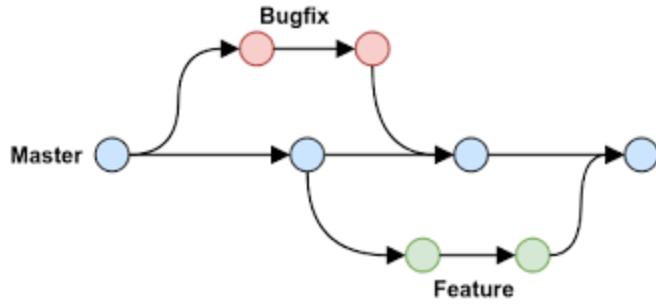
```
# git add .
```

```
# git status
```

```
# git commit -m "my text files only"
```



The diagram about visualize a repository with two isolated lines and development one for a little features and one for a longer running features by developing them is branches it's not only possible to work on both of them in parallel but it also keeps the main master branch free from error.



- Each task has one separate branch
- After done with code merge other branches with master
- This concept is useful for parallel development
- You can create any number of branches
- Changes are personal to that particular branch
- Default branch is Master
- Files created in workspace will be visible in any of the branch workspace until you come to it once you come to it then that file belongs to that particular branch
- When creating new branch data of existing branch is copied to new branch
- To see list of available branches git branch

#git branch

- Create a new branch

#git branch <branch name>

- To switch branch

#git checkout <name of branch you want to change>

Merge

You can't merge branches of different repositories

We use pulling mechanism to merge branches

#git merge <branch name>

- To verify the merge

#git log

- To push to central repo like GitHub

#git push origin master

Git Conflict

When some file having different content in different branches if you do merge conflict occurs
(resolve conflict then add and commit)Conflict occurs when you merge branches

Merge Strategies in Git-

Merge in Git allows you to join two or more development work created using git branch into a single branch

1. Recursive-

```
$git merge -s recursive branch1 branch2
```

Git will select recursive as a default strategy when pulling or merging branches. The recursive strategy can detect and manage merges that involve renames, but it cannot use detected copies.

2. Resolve-

```
$git merge -s resolve branch1 branch2
```

The resolve strategy uses a 3-way merge for resolving branches and can resolve only two HEADs using a 3-way merge algorithm. It is safe and fast and detects criss-cross merge ambiguities in detail.

3. Octopus-

```
$git merge -s octopus branch1 branch2 branch3 branchN
```

When two or more branches are passed, the octopus strategy is engaged, by default. Octopus refuses if the merge has conflicts that need manual resolution. The basic use of Octopus is to bundle feature branch HEADs that have similarities.

4. Ours-

```
$git merge -s ours branch1 branch2 branchN
```

Our strategy resolves multiple branches, but the result is always that of the current branch HEAD. It ignores all changes from all other branches very effectively. It is intended to be used to replace an old history of side branches.

5. Subtree-

```
$git merge -s subtree branchA branchB
```

The subtree strategy is the modified version of the recursive strategy. For example, we merge A and B trees. When corresponding to a subtree of A, B is first modified to reflect the tree structure of A. The modification can be done to the shared ancestor tree of A and B.

Git Stashing

Suppose you are implementing a new feature for your product your code is in progress and suddenly a customer escalation comes because of this you have to keep a side you are a new feature work for new hour you cannot commit you are partial code and also cannot throw away you are changes so you need some temporary storage when you can store you are partial changes and later on commit it

To stash an item (only applies to modified files not new files)

- To stash an Item

```
#git stash
```

- To see stashed items list

```
#git stash list
```

- To apply stashed items

```
#git stash apply stash@{0}
```

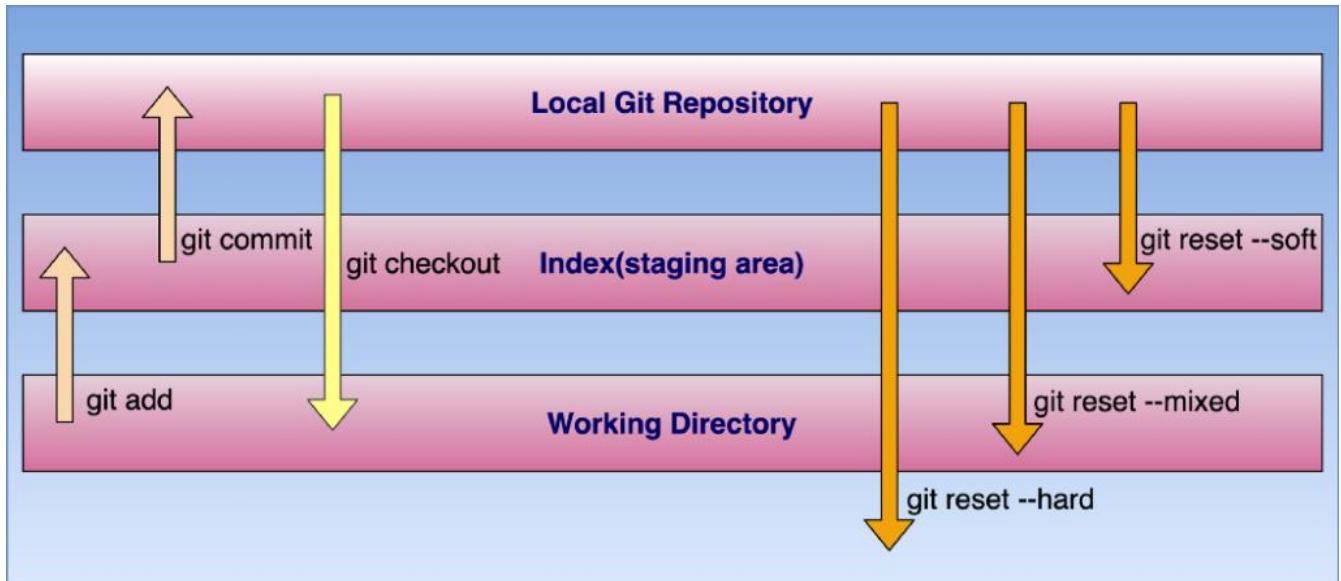
Then you can add and commit

- To clear the stash items

```
#git stash clear
```

Git Reset

There are three different kinds of resets in Git, and they all differ based on how they handle the commits that get left hanging. They all rewrite Git history, and they all move the HEAD back, but they deal with the changes differently:



git reset –soft which will keep your files, and stage all changes back automatically.

```
#git reset –soft HEAD
```

git reset –hard which will completely destroy any changes and remove them from the local directory. Only use this if you know what you're doing.

```
# git reset –hard HEAD
```

git reset –mixed which is the default, and keeps all files the same but unstages the changes. This is the most flexible option, but despite the name, it doesn't modify files.

```
# git reset –mixed commit_id
```

The difference between soft and mixed is whether or not the changes are staged. Staged is basically an in-between zone between the local directory and Git history. `git add` stages files, and `git commit` writes them to the history. In either case, the local directory is unaffected, it just changes the state of Git's tracking of those changes.

Git fork –

A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive use of forking is to propose changes for bug fixing.

Git clone vs. fork –

Any public Git repository can be forked or cloned. A fork creates a completely independent copy of Git repository. In contrast to a fork, a Git clone creates a linked copy that will continue to synchronize with the target repository.

Git fetch –

The git fetch command downloads commits, files, and refs from a remote repository into your local repo. Fetching is what you do when you want to see what everybody else has been working on, but it doesn't force you to actually merge the changes into your repository.

```
# git fetch <remote>
```

Fetch all of the branches from the repository. This also downloads all of the required commits and files from the other repository.

```
# git fetch <remote> <branch>
```

Same as the above command, but only fetch the specified branch.

```
# git fetch --all
```

Git pull –

The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content.

```
# git pull <remote>
```

Git Fetch Vs Git pull

Git Fetch is the command that tells the local repository that there are changes available in the remote repository without bringing the changes into the local repository. Git Pull on the other hand brings the copy of the remote directory changes into the local repository.

Git Tag –

Git tag command, which allows you to label your commits (and other Git objects) by assigning them readable names that can be easily referenced when traversing the history of a Git repository.

```
$ git tag v1.0.0
```

```
$ git tag --list
```

Git Revert-

The git revert command can be considered an ‘undo’ type command, however, it is not a traditional undo operation. Instead of removing the commit from the project history, it figures out how to invert the changes introduced by the commit and appends a new commit with the

resulting inverse content. This prevents Git from losing history, which is important for the integrity of your revision history and for reliable collaboration.

```
$ git revert HEAD
```

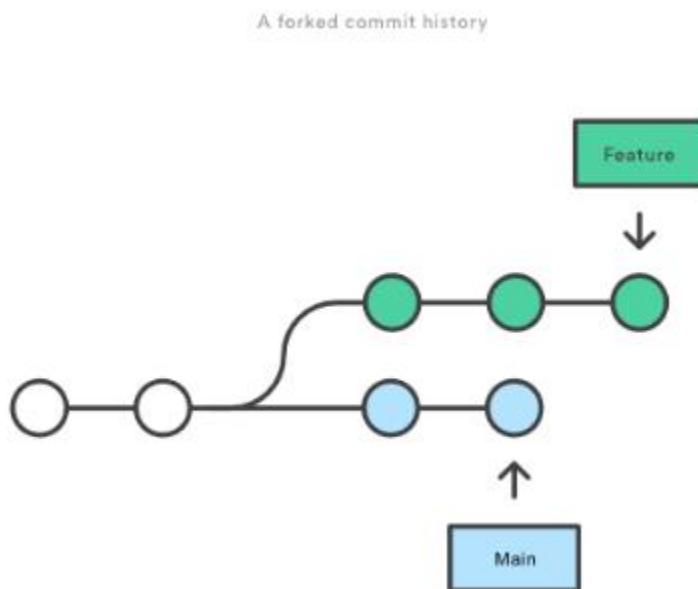
Git revert expects a commit ref was passed in and will not execute without one. Here we have passed in the HEAD ref. This will revert the latest commit. This is the same behaviour as if we reverted to commit.

Git Rebase –

Rebase is one of two Git utilities designed to integrate changes from one branch onto another. Rebasing is the process of combining or moving a sequence of commits on top of a new base commit. Git rebase is the linear process of merging.

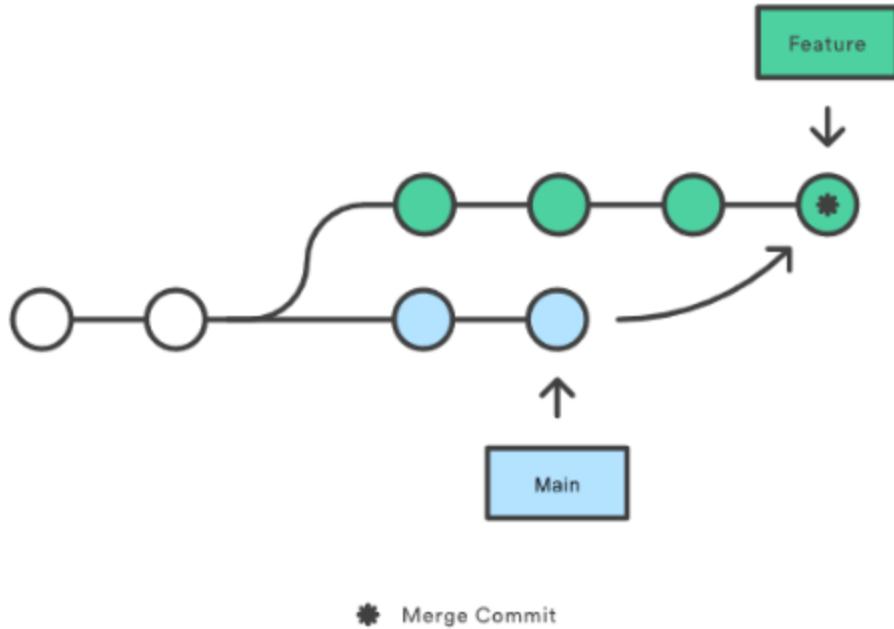
A Git rebase changes the base of the developer's branch from one commit to another, so it looks like they have created their branch from a different commit. Internally, Git creates a new commit and applies it to the specified base. However, it's essential for everyone involved to understand that although the branch appears the same, it's made up of entirely new commits. When you perform a Git rebase, you are, in effect, rewriting history.

Here's how Git rebasing compares to Git merging. Let's say you're a developer who is working on a new feature on a dedicated branch. Then, another development team member updates the main branch with some new commits. The situation looks like this:



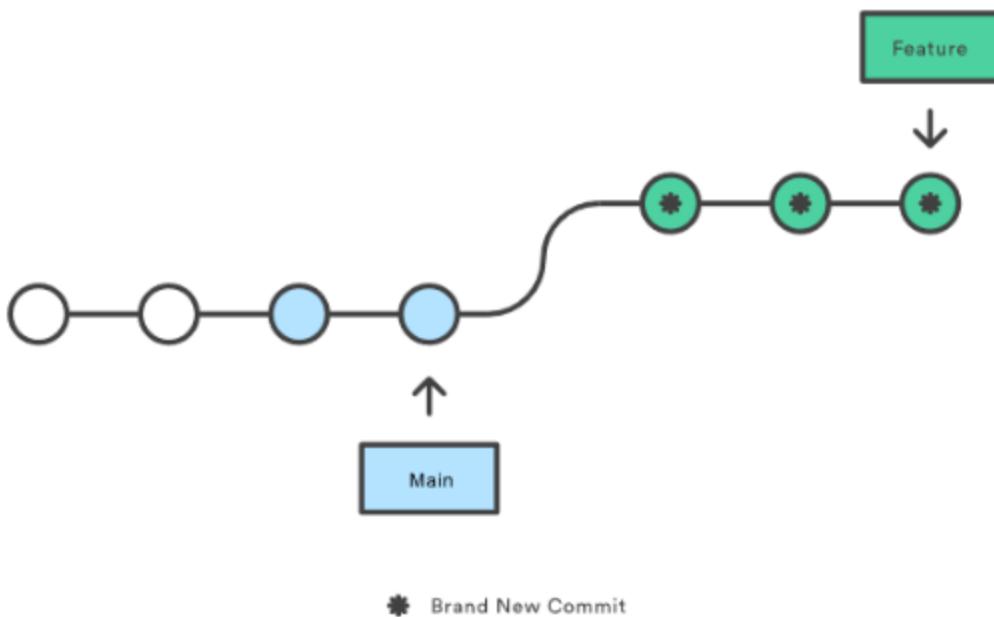
Eventually, however, the team concludes that the main's new commits are relevant to the feature you are working on. So then, if you want to incorporate the new commits onto your branch, you can either do a merge or a rebase. If you decide to use Git merging, you tack on the new commits to your new branch like this:

Merging main into the feature branch



However, if you use Git rebase, you move your whole feature branch, starting it on the tip of the main branch so that all the new commits are now part of the whole. This action rewrites the project history by making new commits for each of the original branch's commits. So, this is how the new branch looks:

Rebasing the feature branch onto main



First, you need to switch to the branch in question:

```
$ git checkout <branch name>
```

Then, just rebase to the master.

```
$ git rebase master
```

Git cherry-pick –

It is necessary in case of bug fixing because bugs are fixed and tested in the development branch with their commits. It is mostly used in undoing changes and restoring lost commits. You can avoid useless conflicts by using git cherry-pick instead of other options.

git cherry-pick is a powerful command that enables arbitrary Git commits to be picked by reference and appended to the current working HEAD. Cherry picking is the act of picking a commit from a branch and applying it to another. git cherry-pick can be useful for undoing changes.

In its most basic form, you only need to provide the SHA identifier of the commit you want to integrate into your current HEAD branch:

```
# git cherry-pick af02e0b
```

This way, the specified revision will directly be committed to your currently checked-out branch. If you would like to make some further modifications, you can also instruct Git to only add the commit's changes to your Working Copy – without directly committing them:

```
# git cherry-pick af02e0b --no-commit
```

Git Hooks

Git hooks are **scripts that run automatically every time a particular event occurs in a Git repository**. They let you customize Git's internal behavior and trigger customizable actions at key points in the development life cycle.

Major benefits of using Git hooks include encouraging a commit policy, automating development workflow, and implementing continuous integration.

There are two groups of Git hooks:

1. client-side / local hooks, which are prompted by events on the local repository, such as when a developer commits or merges code.
2. server-side / remote hooks, which are run on the network hosting the repository, and they are prompted by events such as receiving pushes.

Of the client-side hooks, the most commonly used ones are:

Hook Name	Event	Description
pre-commit	git commit	This hook is called before obtaining the proposed commit message. Exiting with anything other than zero will abort the commit. It is used to check the commit itself (rather than the message).
prepare-commit-msg	git commit	Called after receiving the default commit message, just prior to firing up the commit message editor. A non-zero exit aborts the commit. This is used to edit the message in a way that cannot be suppressed.
commit-msg	git commit	Can be used to adjust the message after it has been edited in order to ensure conformity to a standard or to reject based on any criteria. It can abort the commit if it exits with a non-zero value.
post-commit	git commit	Called after the actual commit is made. Because of this, it cannot disrupt the commit. It is mainly used to allow notifications.
pre-rebase	git rebase	Called when rebasing a branch. Mainly used to halt the rebase if it is not desirable.
post-checkout	git checkout git clone	Run when a checkout is called after updating the worktree or after git clone. It is mainly used to verify conditions, display differences, and configure the environment if necessary.

Some scripts take in one to three arguments, while others take none. Common use cases include checking the commit message for spelling errors, notifying (via email or text) team members of new commits, and implementing continuous integration workflows.

Sharing Git Hooks:

As mentioned above, client-side hooks are local to any given Git repository. This means they are not cloned with the rest of the project and are not version-controlled. As such, maintaining hooks for a team can be tricky.

One workaround is to store the hooks in the project directory, above the .git folder, which would allow the files to be included in the version-control flow. Then, each developer would just copy and paste the files into their local .git/hooks directory. Alternatively, you could create server-side hooks to reject commits that do not conform to certain standards.

Cheat Sheet

Start a project

\$git init <directory>	Create a local repo to initialise the current directory as git repo
git clone <url>	Download a git repo

Make changes

\$git add <file> Add a file to staging

\$git add . Stage all files

\$git commit -m "commit message" Commit all tagged files to git with message

\$git commit -am "commit message" Add all changes made to tracked files & commit

Branches

\$git branch List all local branches.

\$git branch -a List all branches

\$git branch -r List all remote branches

\$git branch new-branch Create new branch

\$git checkout branch Switch to a branch and update working directory

`$git checkout -b new-branch` Create a new branch and switch to it

`$git branch -d branch-name` Delete a merged branch

`$git branch -d branch-name` Delete a branch whether merged or not

`$git tag tag-name` Add a tag to current commit

Merging

`$git merge --squash` Merge & squash all commits into one new commit

Rebasing

`$git rebase -i main` Interactively clean up a branch commit before rebasing into main

`$git rebase -I Head-3` Interactively rebase the last 3 commits on current branch

Undo things

`$git mv existing-path new-path` Move/Rename a file & stage move

`$git rm file` Remove a file from working directory, staging area then stage the removal

`$git rm --cached file-name` Remove from staging area only

`$git checkout commit-id` View a previous commit (Read Only)

`$git revert commit-id` Create a new commit, reverting the changes from a specified commit

`$git reset commit-id` Go back to a previous commit & delete all commits & delete all commits ahead of it

Stashing

`$git stash` Store modified and staged changes

`$git stash save "comment"` Store modified and staged changes with comment

`$git stash -p` Partial stash. Stash just a single file, a collection of files or individual changes from within files

\$git stash list	List all stashes
\$git stash apply	Re-apply stash without deleting it

Synchronising

\$git remote add alias url	Add a remote repo
\$git remote	View all remote connections
\$git remote rename old new	Rename a connection
\$git fetch alias	Fetch all branches from remote repo
\$git pull	Fetch the remote repo's copy and then merge
\$git push alias	Upload local content to remote repo

DevOps

| Tags: [branch](#), [commit](#), [CVCS](#), [DVCS](#), [git](#), [github](#), [hotfix](#), [merge](#), [pull](#), [push](#), [repo](#), [scm](#), [stash](#), [tags](#)

2. Maven

What is Maven: Features

Maven is loaded with many valuable and useful features, which goes a long way towards explaining its popularity. Here are some of Maven's more

noteworthy features:

1. A huge, continuously growing repository of user libraries
2. The ability to set up projects easily, using best practices
3. Dependency management, featuring automatic updating
4. Backwards compatible with previous versions
5. Strong error and integrity reporting
6. Automatic parent versioning
7. Ensures consistent usage across all projects
8. It's extensible, and you can easily write plug-ins using scripting languages or java.

The Need for Maven

Maven is chiefly used for Java-based projects, helping to download dependencies, which refers to the libraries or JAR files. The tool helps get the right JAR files for each project as there may be different versions of separate packages.

After Maven, downloading dependencies doesn't require visiting the official websites of different software. You can visit <https://mvnrepository.com/> to find libraries in different languages. The tool also helps to create the right project structure in struts, servlets, etc., which is essential for execution.

Build Tools Build tools are the tools or programs that help create an executable application from the source code. As the name suggests, it's essential for building or scripting a wide variety of tasks.

The build tool is needed for the following processes:

1. Generating source code
2. Generating documentation from the source code
3. Compiling source code
4. Packaging the compiled codes into JAR files
5. Installing the packaged code in the local repository, server, or central repository

Project Object Model (POM)

Maven is so useful thanks to the Project Object Model (POM), which is an XML file that has all the information regarding project and configuration details. The POM has the description of the project, details regarding the versioning, and configuration

management of the project. The XML file is located in the project home directory. When you execute a task,

Maven searches for the POM in the current directory.

Advantages of Maven:

1. Helps manage all the processes, such as building, documentation, releasing, and distribution in project management.
2. Simplifies the process of project building
3. Increases the performance of the project and the building process
4. The task of downloading Jar files and other dependencies is done automatically
5. Provides easy access to all the required information.
6. Makes it easy for the developer to build a project in different environments without worrying about the dependencies, processes, etc.
7. In Maven, it's easy to add new dependencies by writing the dependency code in the pom file

Disadvantages:

1. Maven requires installation in the working system and the Maven plug-in for the IDE
2. If the Maven code for an existing dependency is unavailable, you cannot add that dependency using Maven itself.

Maven Lifecycle or Goals

1. **Validate** – validate the project is correct and all necessary information is available
2. **Compile** – compile the source code of the project
3. **Test** – test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
4. **package** – take the compiled code and package it in its distributable format, such as a JAR.
5. **Verify** – run any checks on results of integration tests to ensure quality criteria are met
6. **install** – install the package into the local repository, for use as a dependency in other projects locally
7. **deploy** – done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

3. Jenkins

Question: What are Jenkins used for?

Answer: Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on.

What is CI-CD?

- CI/CD is an important DevOps practice and an Agile methodology best practice.
- This practice allows development teams to frequently deliver and deploy applications and accelerate the application development process.
- Introducing a CI/CD pipeline into our software development lifecycle allows us to efficiently implement automation and monitor code changes, new features, potential bug fixes, and more.
- CI/CD typically refers to continuous integration and continuous delivery, but the “CD” can also stand for continuous deployment.
- Continuous delivery and continuous deployment both refer to automating stages of the CI/CD pipeline, but continuous deployment goes a step further.
- The purpose of continuous delivery is to make it easy to deploy new code.

- The purpose of continuous deployment is to allow teams to be “hands-off” in the process by automating the deployment stage.

Benefits Of CI –

- Increased Speed of delivery
- Real time feedback
- Easy maintenance
- Reliable
- Improved collaboration
- High quality code

CI

- Continuous integration refers to the build / integration stage of the software release process.
- It's a stage where developers consistently merge their changes into the main repository of a version control system (like Git).
- After these changes are merged into the main repository, automated builds and tests are run.
- If you don't run the tests yourself, the CI service performs automated tests and builds on any new code changes.
- Continuous integration is a coding philosophy and set of practices that drive development teams to implement small changes and check in code to version control repositories frequently.
- Continuous Integration is a software development method where team members integrate their work at least once a day. In this method, every integration is checked by an automated build to detect errors. This concept was first introduced over two decades ago to avoid “integration hell,” which happens when integration is put off till the end of a project.
- In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If the deployment is a success, the code is pushed to Production. This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

Development without CI	Development with CI
Lots of Bugs	Fewer bugs
Infrequent commits	Regular commits
Infrequent and slow releases	Regular working releases
Difficult integration	Easy and Effective Integration
Testing happens late	Continuous Integration testing happens early and often.
Issue raised are harder to fix	Find and fix problems faster and more efficiently.
Poor project visibility	Better project visibility

Best practices of using CI Systems

Here, are some important best practices while implementing

- Commit Early and Commit Often never Commit Broken Code
- Fix build failures immediately
- Act on metrics
- Build-in every target environment Create artifacts from every build
- The build of the software need to be carried out in a manner so that it can be automated
- Do not depend on an IDE
- Build and test everything when it changes
- The database schema counts as everything
- Helps you to find out key metrics and track them visually
- Check-in often and early
- Stronger source code control
- Continuous integration is running unit tests whenever you commit code
- Automate the build and test everyone
- Keep the build fast with automated deployment

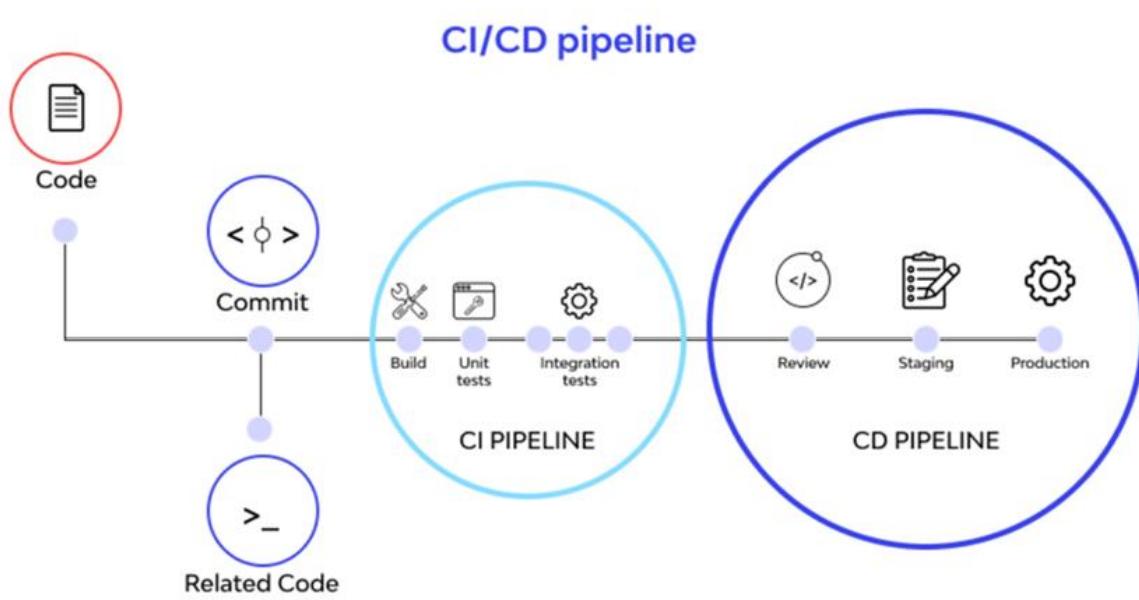
Disadvantages of CI

Here, are cons/drawbacks of Continuous Integration process:

- Initial setup time and training is required to get acquainted with CI server
- Development of suitable test procedures is essential
- Well-developed test-suite required many resources for CI server
- Conversion of familiar processes
- Requires additional servers and environments
- Waiting times may occur when multiple developers want to integrate their code around the same time.

CD

- Continuous delivery picks up where continuous integration ends.
- CD automates the delivery of applications to selected infrastructure environments.
- Most teams work with multiple environments other than the production, such as development and testing environments, and CD ensures there is an automated way to push code changes to them.



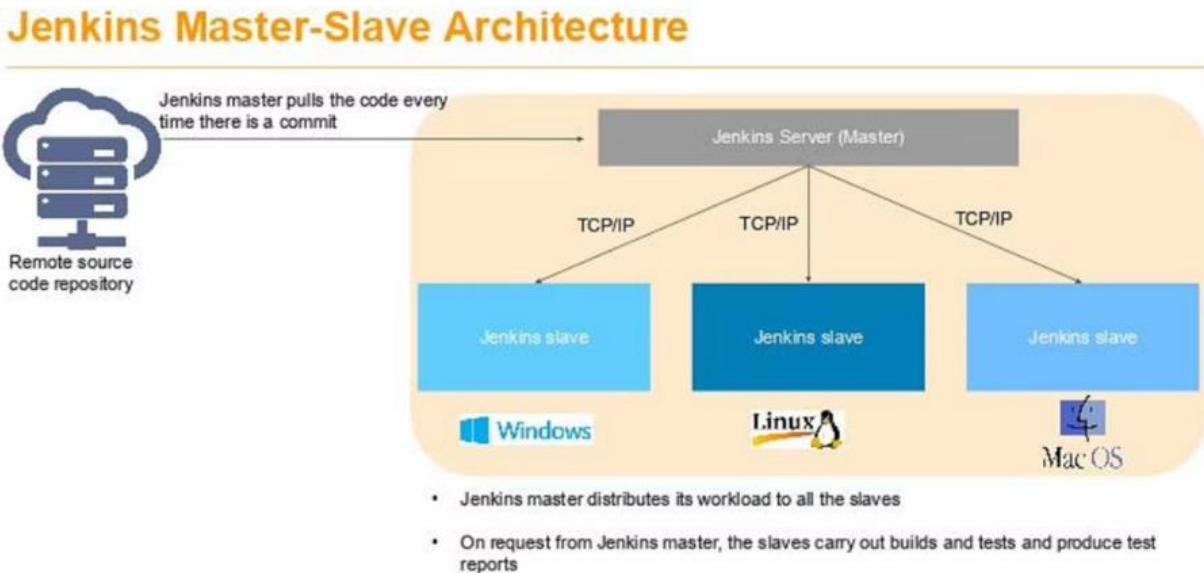
Jenkins is an open-source Continuous Integration server written in Java for orchestrating a chain of actions to achieve the Continuous Integration process in an automated fashion. Jenkins supports the complete development life cycle of software from building, testing, documenting the software, deploying, and other stages of the software development life cycle.

Jenkins is a widely used application around the world that has around 300k installations and growing day by day. By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate build and test at a rapid rate.

As a Continuous Integration tool, Jenkins allows seamless, ongoing development, testing, and deployment of newly created code. Continuous Integration is a process wherein developers commit changes to source code from a shared repository, and all the changes to the source code are built continuously. This can occur multiple times daily. Each commit is continuously monitored by the CI Server, increasing the efficiency of code builds and verification. This removes the testers' burdens, permitting quicker integration and fewer wasted resources.

- Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
- Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.
- Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, **regardless of the platform you are working on**.

Jenkins Architecture:



- Jenkins manages the builds with the help of master-slave architecture. Master and slave units communicate with each other using IP/TCP protocol. Here is a little download on how it all works.

Jenkins master

- This is the primary server of Jenkins.
- It handles a number of tasks that include but are not limited to scheduling build jobs, recording and presenting build results, dispatching builds to slaves for execution, monitoring all the slaves offline as well as online, and others.
- Master Jenkins is capable of directly executing build jobs.

Jenkins slave

- It runs on the remote server.
- The Jenkins server follows the requests of the Jenkins master and is compatible with all operating systems.
- Building jobs dispatched by the master are executed by the slave.
- The project can be suitably configured to choose a specific slave machine.

Jenkins -Master Connectivity

- Using the SSH method: Uses the ssh protocol to connect to the agent. The connection gets initiated from the Jenkins master. There should be connectivity over port 22 between master and agent.

- Using the JNLP method: Uses java JNLP protocol (Java Network Launch Protocol).

Features of Jenkins

- Easy installation and configuration
- Available plugins :- There are hundreds of plugins available in the Update Center, integrating with every tool in the CI and CD toolchain.
- Extensible :- Jenkins can be extended by means of its plugin architecture, providing nearly endless possibilities for what it can do.
- Easy distribution :- Jenkins can easily distribute work across multiple machines for faster builds, tests, and deployments across multiple platforms.
- Free Open Source :- Jenkins is an open-source resource backed by heavy community support.

Types of Jenkins build jobs:

<i>Options</i>	<i>Description</i>
<i>Freestyle Project</i>	Freestyle Project in Jenkins is an improvised or unrestricted build job or task with multiple operations. Operations can be a build, pipeline run, or any script run.
<i>Maven Project</i>	Maven project is selected for managing as well as building the projects which contain POM files. Jenkins will automatically pick the POM files, make configurations, and run our build.
<i>Pipeline</i>	Pipeline demonstrates long-running activities that contain multiple build agents. It is suitable for running pipelines that cannot run through normal freestyle type jobs.
<i>Multi-configuration Project</i>	This option is suitable in those conditions where different configurations like testing on multiple environments, platform-specific builds are required.
<i>GitHub Organization</i>	This option scans the User's GitHub account for all repositories matching some defined markers.

Jenkins Pipeline

- In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.
- The above diagram represents a pipeline in Jenkins. It contains a collection of states such as build, deploy, test and release.
- These jobs or events are interlinked with each other. Every state has its jobs, which work in a sequence called a continuous delivery pipeline.

Declarative pipeline vs Scripted Pipeline:

Declarative	Scripted
<pre>pipeline { agent any stages { stage('Build') { steps { ... } } stage('Test') { steps { ... } } stage('Deploy') { steps { ... } } } }</pre>	<pre>node { stage('Build') { } stage('Test') { } stage('Deploy') { } }</pre>

Declarative Pipeline:

Declarative pipelines are a more recent approach to the “pipeline-as-code” principle. They are quite easy to write and understand. The structure may seem to be a bit complex, but overall, it contains only a couple of basic sections. The “pipeline” block is the main block that contains the entire declaration of a pipeline. In this example, we’ll consider only “agent”, “stages”, and “steps” sections:

- **pipeline** – contains the whole pipeline
 - **agent** – defines the machine that will handle this pipeline
 - **stages** – declares the stages of the pipeline
 - **steps** – small operations inside a particular stage
- **Another important part of declarative pipelines is directives.** In fact, directives are a convenient way to include additional logic. They can help to include tools into a pipeline and can also help with setting triggers, environment variables, and parameters. Some directives can prompt users to input additional information. **There's an easy-to-use generator that can help with creating these directives.**
- In the previous example, “steps” is a directive that contains the logic that will be executed in the stage. There's a dedicated snippet generator that provides a convenient way of creating steps.
- **The power of declarative pipelines comes mostly from directives.** Declarative pipelines can leverage the power of scripted pipelines by using the “script” directive. This directive will execute the lines inside as a scripted pipeline.

Scripted Pipeline–

Scripted pipelines were the first version of the “pipeline-as-code” principle. They were designed as a DSL build with Groovy and provide an outstanding level of power and flexibility. However, this also requires some basic knowledge of Groovy, which sometimes isn't desirable.

- These kinds of pipelines have fewer restrictions on the structure. Also, they have only two basic blocks: “node” and “stage”. A “node” block specifies the machine that executes a particular pipeline, whereas the “stage” blocks are used to group steps that, when taken together, represent a separate operation. **The lack of additional rules and blocks makes these pipelines quite simple to understand:**
- **Think about scripted pipelines as declarative pipelines but only with stages.** The “node” block in this case plays the role of both the “pipeline” block and the “agent” directive from declarative pipelines.
- **As mentioned above, steps for the scripted pipelines can be generated with the same snippet generator.** Because this type of pipeline doesn't contain directives, steps contain all the logic. For very simple pipelines, this can reduce the overall code.
- However, it may require additional code for some boilerplate setups, which can be resolved with directives. **More complex logic in such pipelines is usually implemented in Groovy.**

Jenkins Plugins:

- Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit organization- or user-specific needs. There are over a thousand different plugins which can be installed on a Jenkins controller and to integrate various build tools, cloud providers, analysis tools, and much more.
- Plugins can be automatically downloaded, with their dependencies, from the Update Center. The Update Center is a service operated by the Jenkins project which provides an inventory of open source plugins which have been developed and maintained by various members of the Jenkins community.

Build Triggers:

The screenshot shows the 'Build Triggers' tab selected in a software interface. The 'Build after other projects are built' checkbox is checked. The 'Projects to watch' field is empty and displays a red error message 'No project specified'. Below the field are three trigger options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. A question mark icon is located in the top right corner of the tab.

- **Build after other projects are built –**

If your project depends on another project build then you should select **Build after other projects are built** option from the build triggers.

In this, you must specify the project(Job) names in the **Projects to watch** field section and select one of the following options:

1. **Trigger only if the build is stable** Note: A **build** is stable if it was **built** successfully and no publisher reports it as **unstable**
2. **Trigger even if the build is unstable** Note: A **build** is **unstable** if it was **built** successfully and one or more publishers report it **unstable**
3. **Trigger even if the build fails**

After that, It starts watching the specified projects in the **Projects to watch** section.

Whenever the build of the specified project completes (either is stable, unstable or failed according to your selected option) then this project build invokes.

- **Build Periodically –**

If you want to schedule your project build periodically then you should select the Build periodically option from the build triggers.

You must specify the periodical duration of the project build in the scheduler field section.

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace: * * * * *

For example –

@hourly -> Build every hour at the beginning of the hour -> 0 * * * *

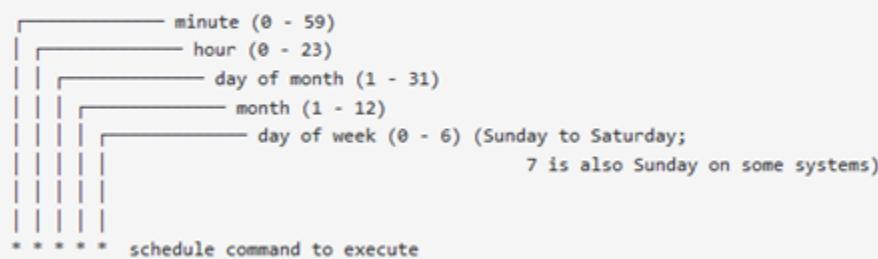
@daily, @midnight -> Build every day at midnight -> 0 0 * * *

@weekly -> Build every week at midnight on Sunday morning -> 0 0 * * 0

@monthly -> Build every month at midnight of the first day of the month -> 0 0 1 * *

Jenkins schedule format

Jenkins schedule format is nothing but a cron schedule expression. It contains 5 fields



- **Poll SCM:**

Poll SCM periodically polls the SCM to check whether changes were made (i.e. new commits) and builds the project if new commits were pushed since the last build.

You must schedule the polling duration in the scheduler field. Like we explained above in the Build periodically section. You can see the Build periodically section to know how to schedule.

After successfully scheduled, the scheduler polls the SCM according to your specified duration in scheduler field and builds the project if new commits were pushed since the last build.

Poll SCM

Schedule

No schedules so will only run due to SCM changes if triggered by a post-commit hook

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)
HOUR The hour of the day (0–23)
DOM The day of the month (1–31)
MONTH The month (1–12)
DOW The day of the week (0–7) where 0 and 7 are Sunday.

- **GitHub webhook trigger for GIT SCM polling:**

A webhook is an HTTP call back, an HTTP POST that occurs when something happens through a simple event-notification via HTTP POST.

GitHub webhooks in Jenkins are used to trigger the build whenever a developer commits something to the branch.

Let's see how to add build a webhook in GitHub and then add this webhook in Jenkins.

- a. Go to your project repository.
- b. Go to "settings" in the right corner.
- c. Click on "webhooks."
- d. Click "Add webhooks."
- e. Write the Payload URL as

- **Github Pull request –**

Trigger that integrates with GitHub Pull Requests and Issues activities and launches runs in response.

- **Trigger Build remotely:**

This option is used when we want to trigger new builds by accessing a special predefined URL.

4. Ansible

Need Of Configuration Management

- DevOps includes coding, building, testing, packaging, release, configuration, and monitoring.
- Configuration management is important in DevOps because it helps you automate otherwise tedious tasks and allows an organization to increase agility.

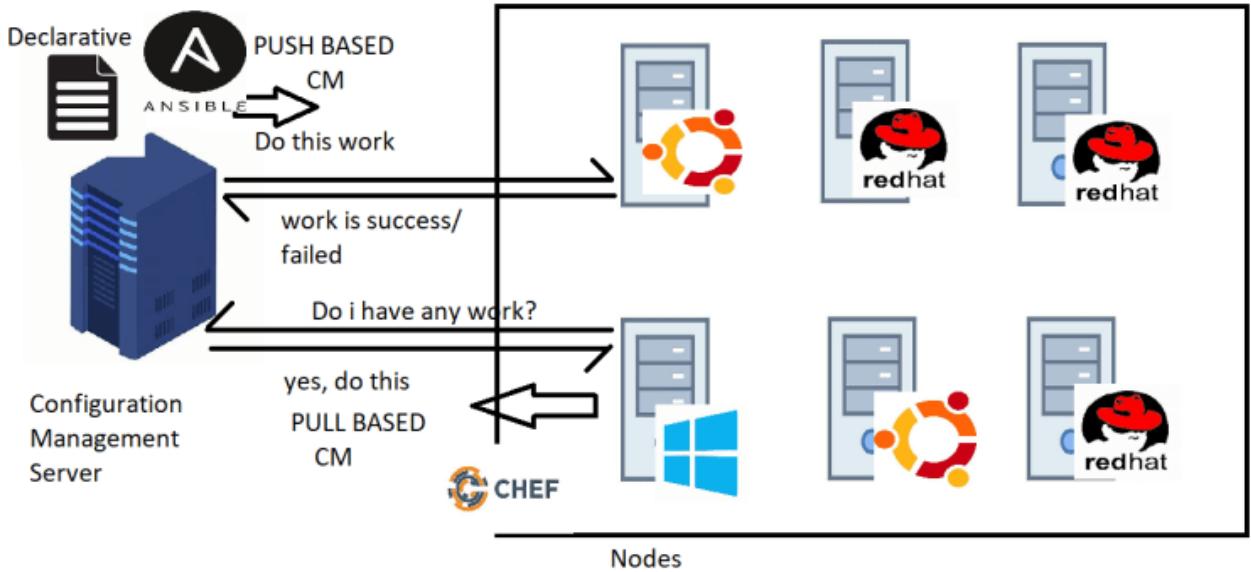
Configuration Management

- In software development circles, configuration management refers to the process by which all environments hosting software are configured and maintained.
- Every development pipeline requires multiple environments for multiple purposes – unit testing, integration testing, acceptance testing, load testing, system testing, end-user testing, etc.
- These environments become increasingly complex as the testing moves towards pre-prod and prod environments.
- Configuration management is an automated process that ensures the configuration of these environments is optimal.

Advantages of Configuration Management Tool

- Increased efficiency with a defined configuration process that provides control and improves visibility with tracking.
- Cost reduction by having detailed knowledge of all the elements of the configuration which allows for unnecessary duplication to be avoided.
- Your business will have greater agility and faster problem resolution, giving a better quality of service for your customers.
- Enhanced system and process reliability through more rapid detection and correction of improper configurations that could negatively impact performance.
- The ability to define and enforce formal policies and procedures that can help with process status monitoring and auditing.
- More efficient change management that reduces the risk of product incompatibility or problems.
- Faster restoration of your service if a process failure occurs. If you know the required state of the configuration then recovering the working configuration will be much quicker and easier.

There are two methods of Configuration Management (CM):



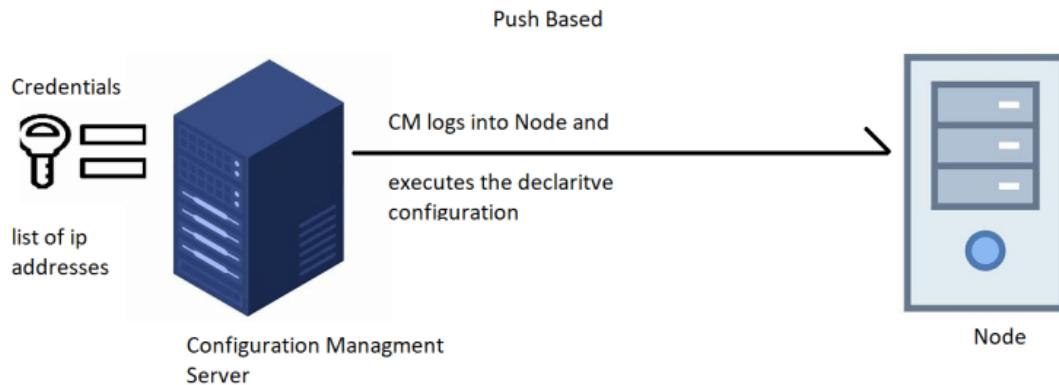
- **PULL BASED CM:** Nodes initiate the communication to the CM Server
- **PUSH BASED CM:** CM server will initiate the communication to the nodes.

PULL BASED CM

- In PULL BASED CM, An agent is installed on every node which is responsible for initiating the communication and following instructions from CM Server

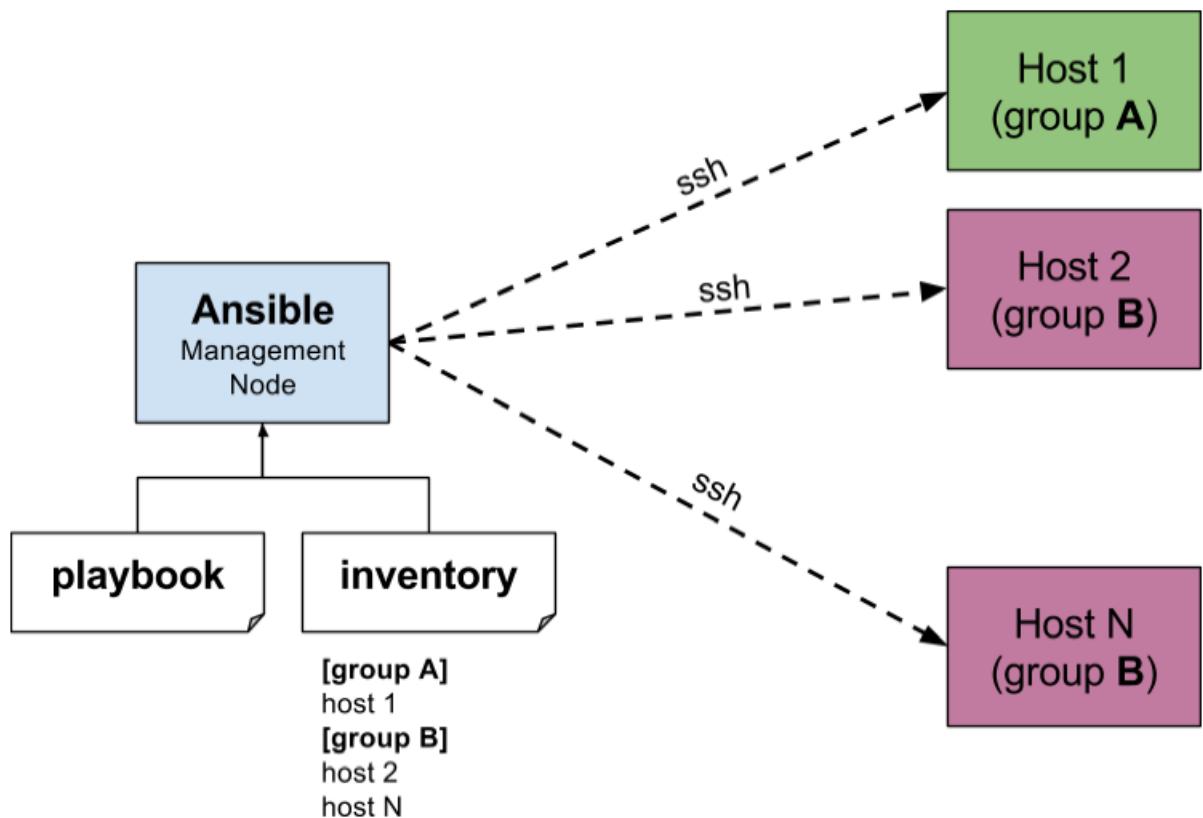
PUSH BASED CM

- In PUSH BASED CM, CM Server has admin credentials of the node and the details like ip address/hostname to login and execute the declarative configuration



Ansible

- Ansible is an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes.
- Ansible is a push based CM.
- Ansible maintains the list of nodes to be communicated and is referred as inventory.
- To write the declarative configuration, Ansible used YAML and calls it as Playbook.
- Ansible is a software tool that provides simple but powerful automation for cross-platform computer support.
- Ansible requirements:
 - Configuration Management Server in the case of ansible can be very light weight machine.
 - Ansible logs in to the nodes and executes the declarative configuration & for that it requires python to be installed on the node.



Inventory files

- Ansible has a default inventory file created when the ansible is installed.

- When ansible is installed in /etc/ansible/ansible.cfg a default configuration for ansible is created.
- Inventory file – /etc/ansible/hosts
- Inventory file can have groups. An entry can be duplicate in many groups.
- The ideal way of dealing with inventory files is to create a inventory file with the name of the environment and ensure the groups names are consistent across different inventories
 - systemtest_hosts
 - performancetests_hosts
 - pre_prod_hosts
 - prod_hosts

Ansible Master & Node setup:

- Refer – <https://www.decodingdevops.com/how-to-install-ansible-on-aws-ec2-instances/>

Ad-Hoc commands, Modules and Playbooks

Ansible Modules

- Ansible with a number of modules (called module library) that can be executed directly on remote hosts or through playbooks
- Your library of modules can reside on any machine and there are no service Daemons or data bases required

Ad-Hoc commands

- Ad -Hoc commands are commands which can be Run individually to perform quick functions.
- Ad-hoc commands can-
 - Can execute any ansible module
 - Good for rare activities
 - history cannot be maintained
- These ad hoc commands are not used for configuration management and deployment because these commands are of one time usage.
- The ansible ad hoc commands uses the user /bin /ansible command line tool to automate a single task.
- Adhoc command:
 - ansible -m <module-name> -a <parameters/arguments> -i <inventory> all
 - arguments:
 - state argument: touch
 - path: /tmp/1.txt
 - ansible -m ansible.builtin.file -a ‘path=/tmp/1.txt state=touch’ all

To run reboot for all your company servers in a group, ‘abc’, in 12 parallel forks

```
$ Ansible abc -a “/sbin/reboot” -f 12
```

Transferring file to many servers/machines

```
$ Ansible abc -m copy -a "src = /etc/yum.conf dest = /tmp/yum.conf"
```

Creating new directory

```
$ Ansible abc -m file -a "dest = /path/user1/new mode = 777 owner = user1 group = user1 state = directory"
```

Deleting whole directory and files

```
$ Ansible abc -m file -a "dest = /path/user1/new state = absent"
```

Managing Packages-

```
$ Ansible abc -m yum -a "name = demo-tomcat-1 state = present"
```

```
$ Ansible abc -m yum -a "name = demo-tomcat-1 state = absent"
```

```
$ Ansible abc -m yum -a "name = demo-tomcat-1 state = latest"
```

Ansible Playbook, Variables, Handlers & Loops

YAML (Yet another Markup Language)

- For ansible nearly every YAML files starts with a list
- Each item in the list is list of key values pairs commonly called a dictionary
- All YAML files have to begins “—“with and ends with”—“
- All members of a list lines must begins with same indentation level starting with”—“

For eg :

```
### dictionary
mysqldatabase:
  hostname: localhost
  port: 3012
  username: root
  password: root
```

```
Imaro:
```

```
author: Charles R. Saunders
language: English
publication-year: 1981
pages: 224
```

- Playbook in ansible are written in YAML format
- It is human readable data serialization language it is commonly used for configuration files
- Playbook is like a file where you write quotes consists of vars, tasks, handlers, files, templates and Roles
- Each Playbook is composed of one or more modules in a list module is a collection of configuration files
- Playbook are divided in to many sections like

* Target section defines the host against which playbooks task has to be executed

* Variable section define variables

* Task section- list of all modules that we need to run in an order

Example Playbook:

```
- hosts: webservers
  become: true
  tasks:
    - name: Create a file
      file: path=/home/batch-13.txt state=touch
```

Variables

- Ansible uses variables which are define previously to enable more flexibility in playbooks and Roles then can be used to loop through a set of given values excess various information like the host name of a system and replace certain strings in templates with specific values.

- Put variable section about task so that we define it first and use it later.
- Now go to ansible server and create one Playbook

```
- hosts: all
vars:
  name: batch-13
tasks:
  - name: Ansible Basic Variable Example
    debug:
      msg: "{{ name }}"
```

```
---
hosts: demo
user: ansible
become: yes
connection: ssh
vars:
  pkgname: https
task:
  - name: install https server
    action: yum name='{{pkgname}}' state=installed
```

Handlers Section

- A handler is exactly the same as a task but it will run when called by another task.
 - OR
- Handlers are just like regular task is an ansible Playbook but our only Run if the task contains a notify directive and also indicates that it changed something.

```
---
- name: Handlers Example
hosts: webservers
gather_facts: false
tasks:
- name: Install httpd latest version
  yum:
    name: httpd
    state: latest
    become: true
    notify: restart_httpd
  handlers:
    - name: restart_httpd
      become: true
      service:
        name: httpd
        state: started
```

Loops

- Sometimes you want to repeat a task multiple Times in computer programming this is called loops common & loops include changing ownership on several files and directories with the file module Creating multiple user with the user module and repeating a bowling step until certain result is reached.

Now go to ansible server

```
[server]# vi loops.yml
```

```
---  
- hosts: webservers  
  become: true  
  tasks:  
    - name: Create multiple directories  
      file: path={{item}} state=directory  
      with_items:  
        - '/home/vn1'  
        - '/home/vn2'  
        - '/home/vn3'
```

Conditions, Vaults and Roles in Ansible

Conditions

Whenever we have different scenarios we put conditions according to the scenario.

“When” statement?

Sometime you want to skip a particular command on a particular node

```
---
- hosts: all
  become: yes

  tasks:
    - name: Install Apache on Ubuntu
      apt: name=apache2 state=present
      when: ansible_os_family == "Debian"
      ignore_errors: True

    - name: Install httpd on Red Hat
      yum: name=httpd state=present
      when: ansible_os_family == "RedHat"
      ignore_errors: True
```

Vault

- Ansible allows keeping sensitive data such as password or keys in encrypted files, rather than a plaintext in your playbooks.

Creating a new encrypted Playbook

```
# ansible-vault create vault.yml // Enter the password
```

Edit the encrypted Playbook

```
# ansible-vault edit vault.yml
```

To change the password

```
# ansible-vault rekey vault.yml
```

To encrypt an existing playbook

```
# ansible-vault encrypt target.yml
```

To Decrypt an encrypt playbook

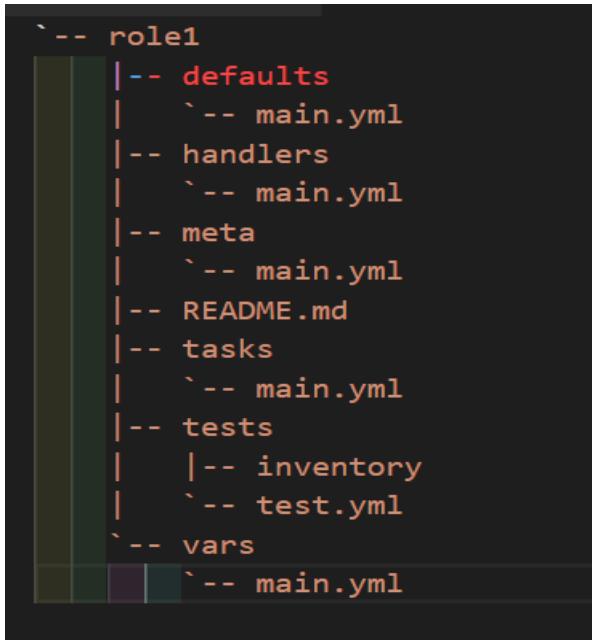
```
# ansible-vault decrypt target.yml
```

Ansible Roles-

- We can use to techniques for reusing a set of tasks includes and Roles.
- Roles provide a framework for fully independent or interdependent collections of files, tasks, templates, variables, and modules.
- The role is the primary mechanism for breaking a playbook into multiple files.
- This simplifies writing **complex playbooks** and makes them easier to reuse.
- Roles are good for organizing task and encapsulating data needed and accomplish those task.
- We can organized play books into directory structure called roles
- Adding more and more functionality to the playbooks will make it difficult to maintain in the single file

Roles

- **Default**— it stores the data about role/ application default variables example if you want to run to port 80 or 8080 then variables needs to define in this path.
- **Files**— it contains files made to transferred to the remote VM (static files)
- **Handlers**— They are triggers or task we can segregate all the handlers required in Playbook.
- **Meta**— this directory content files that establish rolls dependencies example- author name, supported platform, dependencies if any.
- **Task**— it contains all the task that is normally in the Playbook for example- installing packages and copies files etc.
- **Vars**— Variables for the role can be specified in this directory and used in your configuration files both wars and Default stores variables.



Ansible Galaxy –

Ansible Galaxy is essentially **a large public repository of Ansible roles**. Roles ship with READMEs detailing the role's use and available variables. Galaxy contains a large number of roles that are constantly evolving and increasing. Galaxy can use git to add other role sources, such as GitHub.

Ansible is an open-source configuration management tool while Ansible Galaxy is a repository for Ansible roles. Ansible Galaxy for Ansible is what PyPI is for Python, or what Maven is for Java.

Here are some helpful ansible-galaxy commands you might use from time to time:

ansible-galaxy list // displays a list of installed roles, with version numbers.

ansible-galaxy remove <role> // removes an installed role.

ansible-galaxy info // provides a variety of information about Ansible Galaxy.

ansible-galaxy init //can be used to create a role template suitable for submission to Ansible Galaxy.

To create an Ansible role using Ansible Galaxy, we need to use the `ansible-galaxy` command and its templates. Roles must be downloaded before they can be used in playbooks, and they are placed into the default directory `/etc/ansible/roles`. You can find role examples at <https://galaxy.ansible.com/geerlingguy>:

5. Docker

What are Microservices?

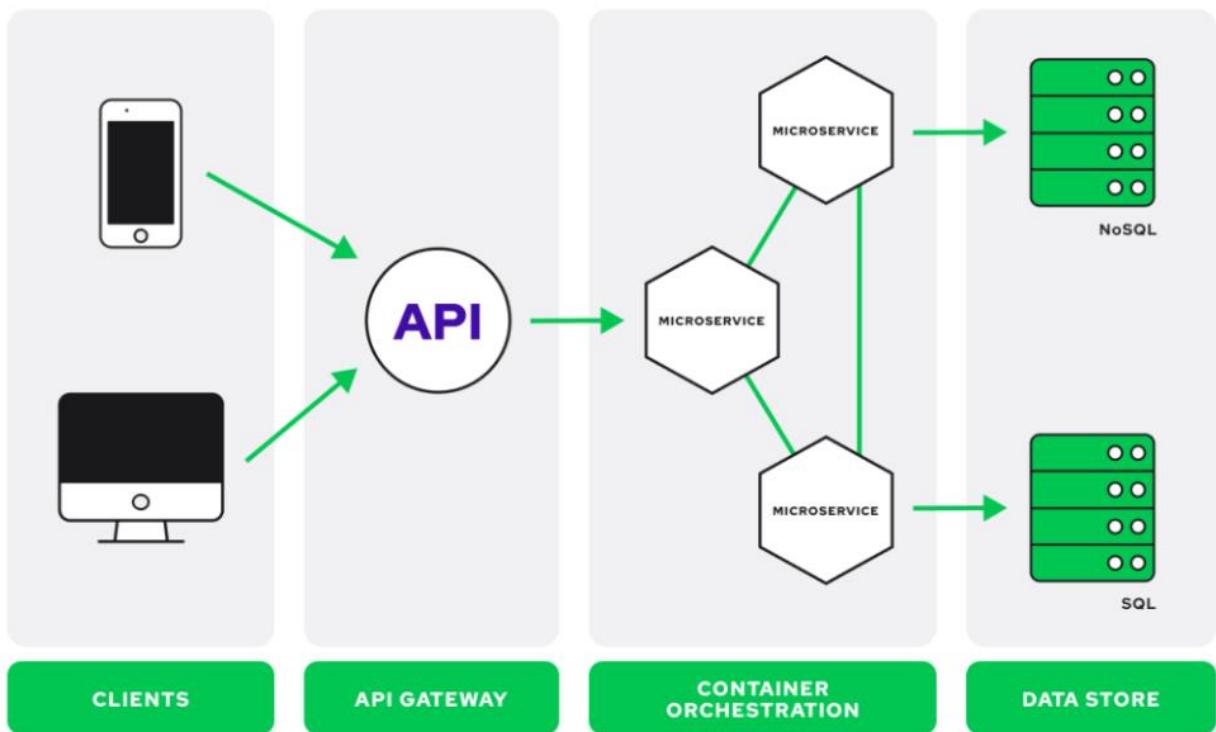
Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams.

Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features. Microservices are:-

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

Benefits of Microservices-

1. **Agility** – Microservices foster an organization of small, independent teams that take ownership of their services.
2. **Flexible Scaling** – Microservices allow each service to be independently scaled to meet demand for the application feature it supports.
3. **Easy Deployment** – Microservices enable continuous integration and continuous delivery, making it easy to try out new ideas and to roll back if something doesn't work.
4. **Technological Freedom** – Microservices architectures don't follow a "one size fits all" approach. Teams have the freedom to choose the best tool to solve their specific problems.
5. **Reusable code** – Dividing software into small, well-defined modules enables teams to use functions for multiple purposes. A service written for a certain function can be used as a building block for another feature.
6. **Resilience** – Service independence increases an application's resistance to failure. In a monolithic architecture, if a single component fails, it can cause the entire application to fail.



What is Virtualisation?

- Virtualization is technology that lets you create useful IT services using resources that are traditionally bound to hardware.
- It allows you to use a physical machine's full capacity by distributing its capabilities among many users or environments.
- Virtualization is a technique of importing a Guest OS on the top of the host OS.
- This technique was a revelation at the beginning as it allowed developers to run multiple OS in different VMs on the same physical host.

Benefits of Virtualisation

1. More flexible and efficient allocation of resources.
2. Enhance development productivity.
3. It lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.
5. High availability and disaster recovery.
6. Pay peruse of the IT infrastructure on demand.
7. Enables running multiple operating systems.

Disadvantages of Hypervisor:

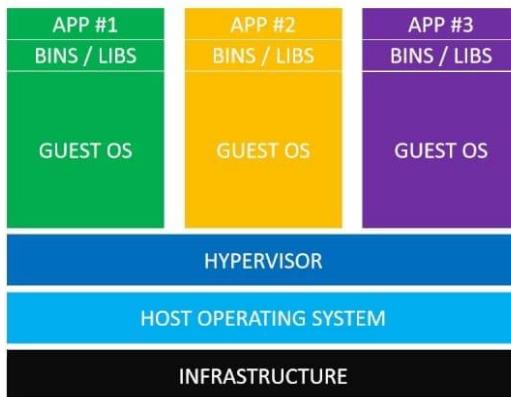
- Running multiple VMs leads to unstable performance.
- Hypervisors are not as efficient as the host OS.
- Boot up process is long and takes time.

Containerization:

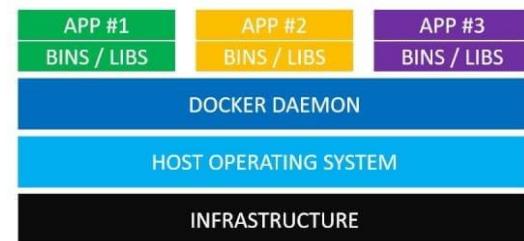
- Containerization is a form of virtualization where applications run in isolated user spaces, called containers, while using the same shared operating system (OS).
- A container is essentially a fully packaged and portable computing environment.
- Everything an application needs to run—its binaries, libraries, configuration files, and dependencies—is encapsulated and isolated in its container.
- The container itself is abstracted away from the host OS, with only limited access to underlying resources—much like a lightweight virtual machine (VM).

What is Docker?

- Docker is an open source centralized platform designed to create deploy and run actions.
- Docker uses container on the host OS to run applications. It allows applications to use the same Linux Kernel as a system on the host computer rather than creating a whole virtual OS.
- We can install docker on any OS but docker engine runs natively on Linux distribution.
- Docker is a tool that performs OS level virtualization also known as containerization.
- Before docker many users faces the problem that a particular code is running in developers system but not in the uses system.
- Docker is a set of platform as a service that uses OS level virtualization where VMware uses hardware level virtualization.



Virtual Machines



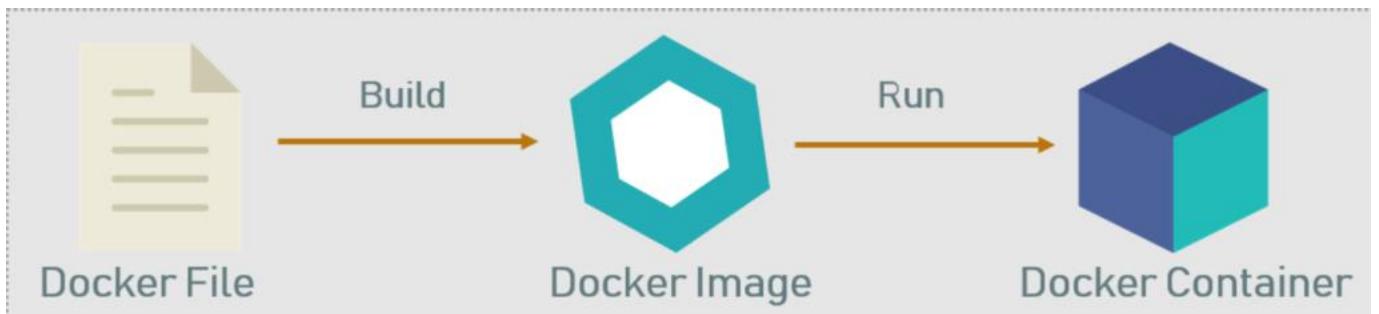
Docker Containers

Advantages of Docker

- No pre allocation of RAM.
- CI efficiency- docker enables you to build a container image and use that same image across every step of the deployment process.
- Less cost.
- It is light weight.
- It can run on physical H/W virtual H/W or on cloud.
- You can reuse the image.
- It took very less time to create container.

Disadvantages of Docker

- Docker is not a good solution for application that requires rich GUI.
- Difficult to manage large amount of containers
- Docker does not provide cross platform compatibility means if an application is designed to run in docker container on Windows than it can't run on Linux or vice versa.
- Docker is suitable when the developer OS and testing OS are same if the OS is different we should use VM.
- No solution for data recovery and backup.



Docker Architecture:

Docker Daemon

- The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
- A daemon can also communicate with other daemons to manage Docker services.
- Docker daemon runs on the host OS.
- It is responsible for running containers to manage docker services.
- Docker Daemon can communicate with other daemons.

Docker client

- Docker users can interact with docker daemon through a client (CLI).

- Docker client uses commands and rest API to communicate with the docker daemon
- When a client runs any server command on the docker client terminal, the client terminal sends this docker commands to the docker daemon.
- It is possible for docker client to communicate with more than one day one daemon.
- The Docker client (docker) is the primary way that many Docker users interact with Docker.
- When you use commands such as docker run, the client sends these commands to dockerd, which carries them out.
- The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker host

- Docker host is used to provide an environmental execute and run applications it contains the docker daemon images, containers, networks and storages.

Docker Hub/Registry

- Docker registry manages and stores the docker images.
- There are two types of registries in the docker.

1. Public registry- public registry is also called as docker hub.

2. Private registry- it is used to share images within the Enterprise.

- A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.
- When you use the docker pull or docker run commands, the required images are pulled from your configured registry.
- When you use the docker push command, your image is pushed to your configured registry.

Docker images

- Docker images are the read only binary templates used to create docker containers.

Or

Single file with all dependency and configuration required to run a program.

- Ways to create an images-

1. Take image from docker hub
2. Create image from docker file
3. Create image from existing docker containers.

Docker Objects

- When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects.
- These we will discuss one by one.

Objects

1. Images

- Images are nothing but a read-only binary template that can build containers.
- They also contain metadata that describe the container's capabilities and needs.
- Images are used to store and ship applications.
- An image can be used on its own to build a container or customized to add additional elements to extend the current configuration.
- You might create your own images or you might only use those created by others and published in a registry.
- To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it.
- Each instruction in a Dockerfile creates a layer in the image.
- When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.
- This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

Docker Container

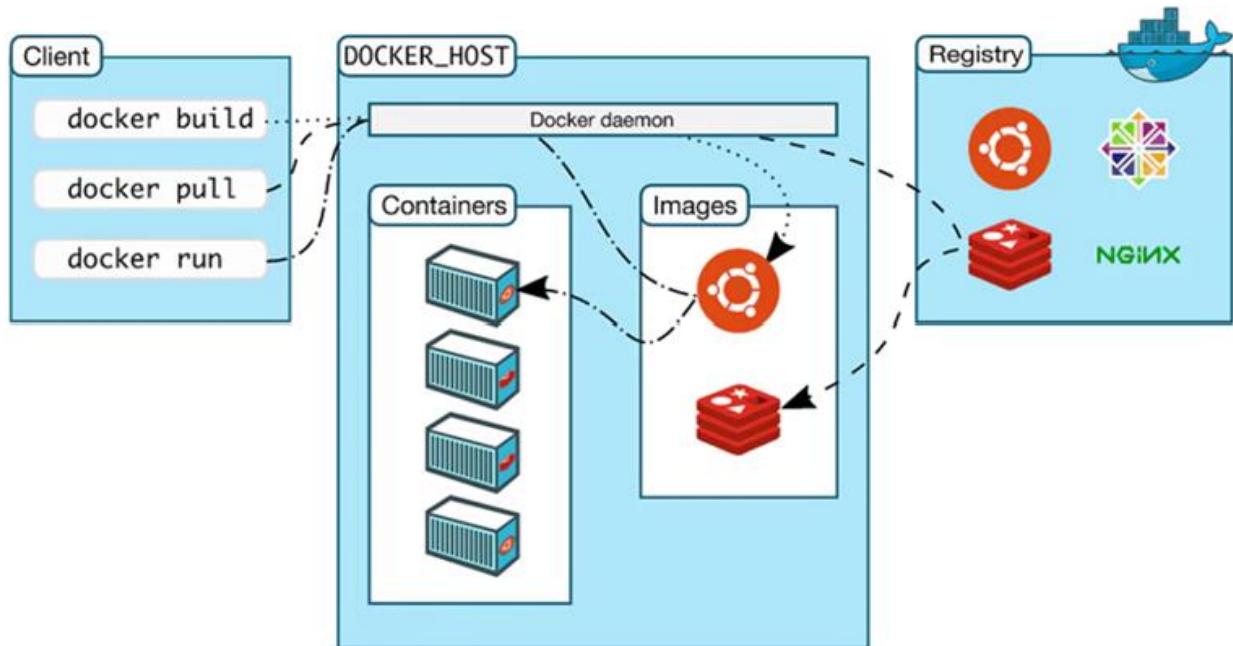
- Container hold the entire packages that is needed to run the application.

Or

In other words we can say that the image is template and the container is a copy of that template.

- Container is like a virtual machine.
- Images becomes container when they run on docker engine.
- To see all images present in your local machine
`# docker images`
- To find out images in docker hub
`# docker search Jenkins`
- To download image from docker hub to local machine.
`# docker pull Jenkins`
- To give name to container
`#docker run -it --name container_name ubuntu bin/ bash`

- To check service is start or not
#service docker status
- To start container
#docker start container_name
- To go inside container
#docker attach container_name
- To see all containers
#docker ps -a
- To see only running containers
#docker ps
- To stop container
#docker stop container_name
- To delete container
#docker rm container_name



Dockerfile Components & Commands

Therefore, create one container first.

```
#docker run -it --name container_name ubuntu /bin/bash
```

Dockerfile

- Docker file is basically a text file it contains set of instruction.
- Automation of docker image creation

Docker Components

- **FROM**- Every dockerfile starts with from command for Base image this command must be on top of the docker file. ex. FROM Ubuntu
- **RUN**- The run command is used to run any shell commands these commands will run on top of the base image layer. run commands executed during the build time. you can use this command any number of times in dockerfile. ex. RUN apt-get update && apt-get install -y nginx
- **MAINTAINER**- Author/ Owner /description ex. MAINTAINER "ygminds73@gmail.com"
- **COPY**- It will copy the files and directories from the host machine to the container. if the destination does not exist. it will create the directory. ex. COPY . /usr/src/app
- **ADD**- it will copy the files and directories from the host machine to the container and its support copy the files from remote URLs. ex. ADD <SRC> <DEST>
- **EXPOSE**- to expose points such as port 8080 for tomcat port 80 nginx etc, ex. EXPOSE 80
- **WORKDIR**- It is just like Linux cd command. If you mention any path after workdir the shell will be changed into this directory. The next mentioned commands like RUN, CMD, ENTRYPOINT commands will be executed in this directory.

ex. WORKDIR /devops

RUN npm install

- **CMD**- The CMD command doesn't execute during the build time it will execute after the creation of the container. there can be only one CMD command in Dockerfile. if you add more than one CMD commands the last one will be executed and remaining all will be skipped. whatever you are mentioning commands with CMD command in Dockefile can be overwritten with docker run command. if there is ENTRYPOINT command in the Dockerfile the CMD command will be executed after entry point command. with CMD command you can pass arguments to ENTRYPOINT command. CMD command you can use in two types one is as executable for and another one is parameters form
 - ex. **Parameters form:**

CMD [“param1”, “param2”] this param1 and param2 will be passed to ENTRYPOINT command as arguments.

ENTRYPOINT [“/db.sh”]

CMD [“postgress”]

Executable form:

CMD [“python”, “app.py”]

python app.py will be run in the container after the creation of the container.

- **ENTRYPOINT**–

The first command that will execute in the container is ENTRYPOINT command.

ENTRYPOINT command will be executed before the CMD command. If you mentioned more than one, only the last one will be executed and remaining all will be skipped. CMD command parameters are arguments to ENTRYPOINT command.

ex.

ENTRYPOINT [“/db.sh”]

CMD [“postgress”]

above commands will run as /db.sh postgress

here postgress is the first argument to script db.sh

- **ENV**– This command is used to set environment variables in the container.

DockerFile

1. Create a file name docker file
2. Add instruction in docker file
3. Build docker file to create image
4. Run image to create container

#vim dockerfile

```

# Dockerfile Maintainer
MAINTAINER "ygminds73@gmail.com"

# Install nginx and adjust nginx config to stay in foreground
RUN apt-get update && apt-get install -y nginx; \
echo "daemon off;" >> /etc/nginx/nginx.conf

# Expose HTTP
EXPOSE 80

# Start nginx
CMD ["/usr/sbin/nginx"]

```

To create image out of docker file

```
#docker build -t myimage.
```

```
#docker ps -a
```

```
#docker images
```

- Now create container from the above image

```
#docker run -it --name mycontainer myimage /bin/bash
```

```
#cat /tmp/testfile
```

Docker Volume & how to share it

- Volume is simply a directory inside our containers
- Firstly we have to declare this directory as a volume and then share volume
- Even if we stop container still we can access volume
- Volume will be created in Once container
- You will declare a directory as a volume only while creating container
- You can't create volume from existing container
- You can share one volume across any number of containers
- Volume will not be included when you update an image

You can map volume in the two ways

1. Container – container

2. Host – container

Benefits of volume

- Decoupling container from storage
- Share volume among different containers
- Attach volume to containers
- On deleting container volume does not delete

Creating volume from Dockerfile

- Create a docker file and write

Then create image from this docker file

```
#docker build -t myimage
```

- Now create a container from this image and run
 - #docker run -it --name cont1 myimage/bin/bash
 - Create files under volume
 - #docker run -it --name cont2 --privileged=true --volumes-from cont1 ubuntu /bin/bash
- Now try to create volume by using command
 - #docker run -it --name cont3 -v /volume2 ubuntu /bin/bash
 - Create files in the containers
 - Now create one more container and share volume 2
 - #docker run -it --name cont4 --privileged=true --volumes-from cont3 ubuntu /bin/bash
- Volumes (Host-container)

Create/decide a directory which we want to be treat as a volume **/home/ec2 user**

```
# docker run -it --name hostcont -v /home/ec2-user:/volume --privileged=true ubuntu /bin/bash
```

```
cd volume/
```

check and add files

•

Docker Attach Vs Docker Exec

Docker exec creates a new process in the containers environment while docker attach just connect this standard input or output of the main process inside the container to corresponding

standard input or output error of current terminal. Docker exec is specifically for running new things in a already started container, be it a shell or some other process

```
#docker attach jenkins
```

```
#docker exec -i -t jenkins /bin/bash
```

Docker exec is specifically for running new things in a already started container, be it a shell or some other process

Docker Port Forwarding–

Port forwarding or port mapping **redirects a communication request from one IP address and port number combination to another**. Through port forwarding, services are exposed to the applications residing outside of the host's internal network.

```
#docker run -d --name jenkins -p 8080:8080 jenkins/jenkins
```

Difference between expose and publish

Basically you have three options

1. Neither specify expose nor -P

If you specify neither expose nor -p the service in the container will only be accessible from inside the container itself.

2. Only specify expose

If you expose a port, the service in the container is not accessible from outside docker but from inside other docker containers so this is good for inter container communication.

3. Specify expose and -P

If you expose and -P a port, the service in the container is accessible from anywhere even outside docker

If you do -p but do not expose docker does an implicit expose this is because if a port is open to the public, it is automatically also open to the other docker containers.

hence '-P' includes expose

Docker Networks:

- Docker networking is a passage through which all the isolated container communicate. There are mainly five network drivers in docker:

- **Bridge:** It is the default network driver for a container. You use this network when your application is running on standalone containers, i.e. multiple containers communicating with the same docker host.
- **Host:** This driver removes the network isolation between docker containers and docker host. You can use it when you don't need any network isolation between host and container.
- **Overlay:** This network enables swarm services to communicate with each other. You use it when you want the containers to run on different Docker hosts or when you want to form swarm services by multiple applications.
- **None:** This driver disables all the networking.
- **macvlan:** This driver assigns mac address to containers to make them look like physical devices. It routes the traffic between containers through their mac addresses. You use this network when you want the containers to look like a physical device, for example, while migrating a VM setup.

Docker Compose:

- Docker Compose is a tool that was developed to help define and share multi-container applications.
- Docker compose is a tool for defining and running multi-container docker applications
- With compose, you can use a compose file to configure your application services. Then you use a single command, you create and start all the services from your configuration.
- Compose have multiple commands so as to start, stop, rebuild, view the status of running services.
- To install docker compose – <https://www.techgeekbuzz.com/tutorial/docker/how-to-install-docker-compose-in-linux/>

Docker Compose File:

- Docker compose uses a yaml file where you define all the steps.
- Default file Is docker-compose.yml or docker-compose.yaml

YAML:

- YAML stands for YAML Ain't Markup Language.
- Previously, it stood for Yet Another Markup Language.
- YAML is a very simple, text-based, human-readable language used to exchange data between people and computers.
- YAML is not a programming language. It is mostly used for storing configuration information.
- YAML files store data, and they do not include commands and instructions.
- YAML is a data serialization language similar to XML or JSON.

Basic Syntax of YAML:

- YAML is used to store key-value pairs, or mappings, and lists, or sequences of elements.
- <key>: <value>

- where <key> represents name and <value> represents data separated by : (the space is mandatory).
- In yaml file whatever data we will give is line by line.
- In yaml any key can also have a child key.
- Sometime we may also need to group parent keys, where we will have something called as sections.

Docker Compose File:

- Service:- A service definition contains configuration that is applied to each container started for that service, much like passing command-line parameters to docker run.
- In simple words collection of everything that I want to tell docker-compose to create a container or multiple containers i.e what is the image you want how many containers etc.
- Here in compose we are not going to refer as a container but as a service.
- Docker compose will not create container it will create service, service will include image, containers, ports etc.
- Compose handles everything as a service.

```
docker-compose.yaml
1  version: "3.9"
2  services:
3    # Service "web"
4    web:
5      # Build Dockerfile in current dir
6      build: .
7      # Expose app port 5000 on machine as 8000
8      ports:
9        - "8000:5000"
10     # mount the project directory (current directory) on the host to /code inside the container
11     # modify the code on the fly, without having to rebuild the image.
12     volumes:
13       - ./code
14     # tell flask run to run in development mode and reload the code on change
15     environment:
16       FLASK_ENV: development
17     # Service "redis" from image redis:alpine
18     redis:
19       image: "redis:alpine"
```

Commands:

- To run compose-file:- docker-compose –f <compose-file> options servicename
- To create/start container:- docker-compose up

- To start in detach mode:- docker-compose up -d
- To start specific service:- docker-compose up -f <servicename>
- To list container:- docker-compose ps <service>
- To stop-remove container:- docker-compose down
- List images:- docker-compose images
- Build services:- docker-compose build servicename
- View output of container:- docker-compose logs -f service
- Stop services :- docker-compose stop sevicename
- Remove:- docker-compose rm servicename
- Execute commands in a running container:- docker-compose exec service command
- Execute commands in a running container detach mode:- docker-compose exec -d
- Scaling container:- docker-compose up -d --scale service=num

Docker Commands

DOCKER VOLUME

docker volume ls	List all Volumes
docker volume create <volume name>	Create a Volume
docker volume rm <volume name>	Remove Volume
docker volume prune	It removed all unused docker volume
docker volume inspect <volume name>	Full info about volume
DOCKER INFORMATION	
docker ps	Running container
docker ps -a	Show all container
docker ps --format "{{.Names}}"	Get docker container names
docker inspect container	Return low-level information on Docker objects
docker rename	Rename a container
docker pause	Pause all processes within one or more containers

<code>docker logs</code>	Fetch the logs of a container
<code>docker logs --until 10m <container_id></code>	logs until 10min
<code>docker logs --tail 100 <container_id></code>	latest 100 lines logs
<code>journalctl CONTAINER_NAME=mycontainer</code>	In order to inspect logs sent to journal
<code>docker container inspect <container name></code>	Full info about container
<code>docker exec -it <container> /bin/bash</code>	execute commands on running containers
<code>dcoker events <container name></code>	List real time events from container
<code>docker port <container name></code>	Show port mapping from container
<code>docker top <container name></code>	Show running process in container
<code>dcoker stats <container name></code>	Show live resources usage
<code>docker rm <container name></code>	Remove specific container
<code>docker rm \$(docker ps -a -q)</code>	Remove all containers
<code>docker commit</code>	Create a new image from a container's changes
<code>docker cp</code>	Copy files/folders between a container & host
<code>docker kill <container name></code>	Kill one or more running containers
<code>docker diff</code>	Show changes to file and file system
<code>dcoker update <container name></code>	To update the configuration of container
<code>docker import {url/file}</code>	Create a image from tarball
<code>docker save [image] > [Tar file]</code>	Save an image to tar file
DOCKER IMAGES	

docker images ls	show all images
docker rmi <image>	Remove image
docker rmi \$(sudo docker images -q)	remove all images
DOCKER ACTIONS	
docker start	Start container
docker stop	Stop container
docker restart	Restart container
dcoker pause	Pause container
docker unpause	Unpause container
docker wait	Wait container
docker kill	Kill container
docker attach	Attach container /Access of container
DOCKER HUB & SYSTEM	
docker login	Log in to a Docker registry
docker pull	Pull an image or a repository from a registry
docker push	Push an image or a repository to a registry
docker system df	Show docker disk usage.
docker system events	Get real-time events from the server
docker system info	Display system-wide information.

docker system prune	Remove unused data.
DOCKER NETWORKING	
docker network ls	List Network
docker netowrk rm <network>	Remove one or more network
docker netowrk inspect <netowrk>	Network information
docker netowrk connect <network><container>	Connect container to a network
docker netowrk disconnect <network><container>	Disconnect container to a network

6. Kubernetes

What is Kubernetes?

Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Google originally designed Kubernetes, but the Cloud Native Computing Foundation now maintains the project.

Cloud based K8S Services-

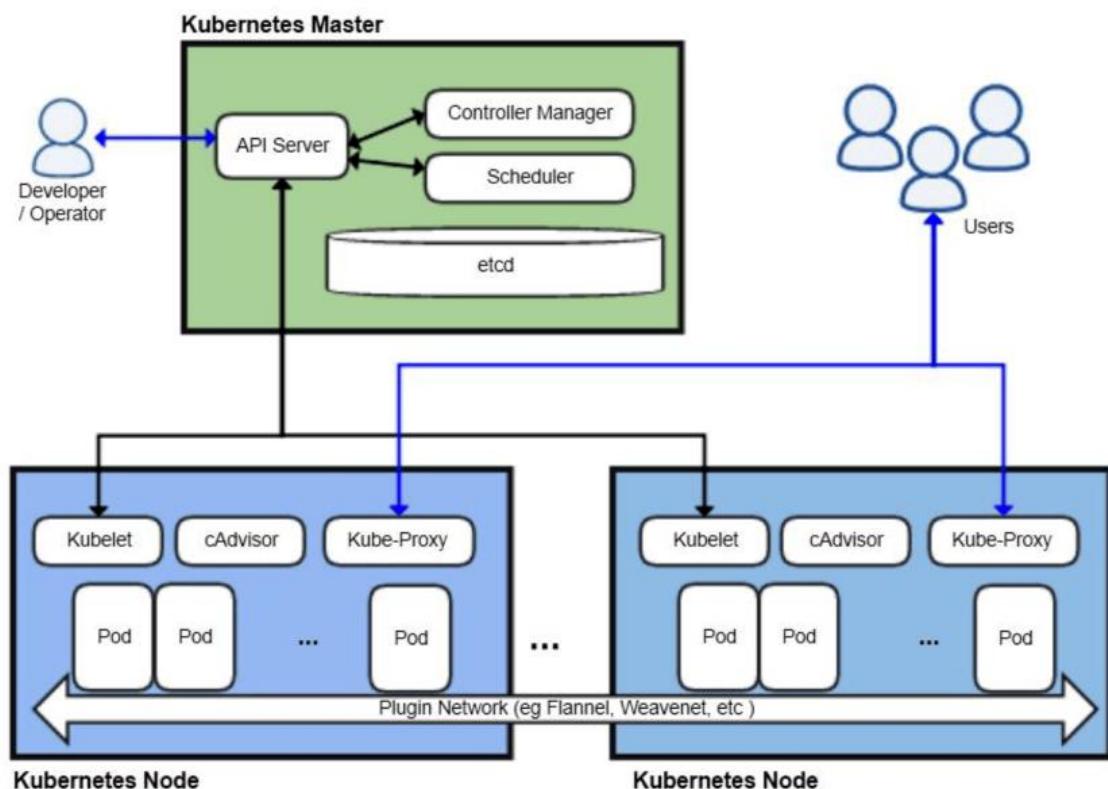
- GKE- Google kubernetes services
- AKS- Azure kubernetes services
- Amazon EKS(Elastic kubernetes services)

Kubernets Vs Docker Swarm

FEATURES	KUBERNETES	DOCKER SWARM
Installation and cluster configuration	Complicated & time Consuming	Fast & Easy
Support	K8s Can work with almost all container	Work with Docker only

	types like Rocket, Docker, ContainerD	
GUI	GUI Available	GUI not available
Data Volumes	Only shared with containers in same pod	Can be shared with any other Container
Update & Rollback	Process Scheduling to maintain services while updating	Progressive updates of services health monitoring throughout the update
Autoscaling	Support vertical & Horizontal Autoscalling	Not support Autoscalling
Logging & Monitoring	Inbuilt tool present for monitoring	Used 3 rd party tools like Splunk

Architecture:



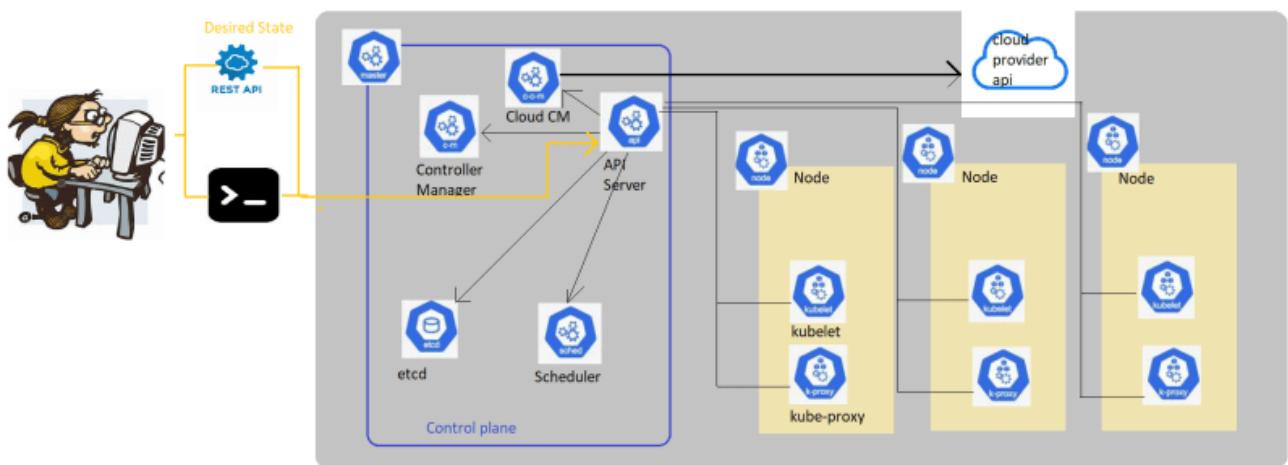
- Components of Kubernetes (k8s) cluster

- **Control Plane Components**
- The control plane components make decisions about the cluster
- Control plane can be run on any machine in the cluster
- We can create a highly available cluster by using multiple machines for control plane components
- **The components are:**
- kube-apiserver
- etcd
- kube-scheduler
- kube-controller-manager
- kube-cloud-controller-manager
- **Node Components:**
- They run on every node, maintaining running pods and providing k8s runtime environment.
- Our applications will be running on nodes
- The Node Components are
- kubelet
- kube-proxy
- Container runtime
- **Kube-Apiserver:**
- The API server is a component of k8s control plane that exposes k8s API (Front-end of k8s)
- All the communication between control plane and nodes is also handled by api server
- To make k8s HA (highly Available), we can horizontal scale api-server
- As a user of k8s cluster we can interact with kube-api server using API with json or a tool called a kubectl which is a command line tool
- **etcd:**

Is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines. It gracefully handles leader elections during network partitions and can tolerate machine failure, even in the leader node.

- This is distribute key-value store.
- k8s uses etc to store all the cluster data
- **kube-scheduler:**
- Control plane component that creates Pods on the nodes by selecting them
- **kube-controller-manager:**
- Control plane component runs controller processes. Each controller is a separate process, but to reduce complexity they run in single process
- **Some major types of controller are**
- Node Controller: Responsible for noticing and responding when node goes down
- Job Controller
- Endpoints controller

- **Cloud-controller-manager:**
- This component embeds cloud-specific logic
- **Kubelet:**
- This is an agent that runs on each node in the cluster.
- Kubelet receives requests/orders to create new Pods
- **kube-proxy:**
- This is a network proxy that runs on each node in k8s
- This maintains network rules on the nodes
- Kube proxy is responsible for routing network traffic in the k8s cluster. To do this job, the proxy should be present on all the nodes in the cluster
- **Container runtime:**
- Kubernetes supports container run times such as contained, CRI-O and any implementation of Kubernetes CRI (Container Runtime instance)
- **Basic Workflow**



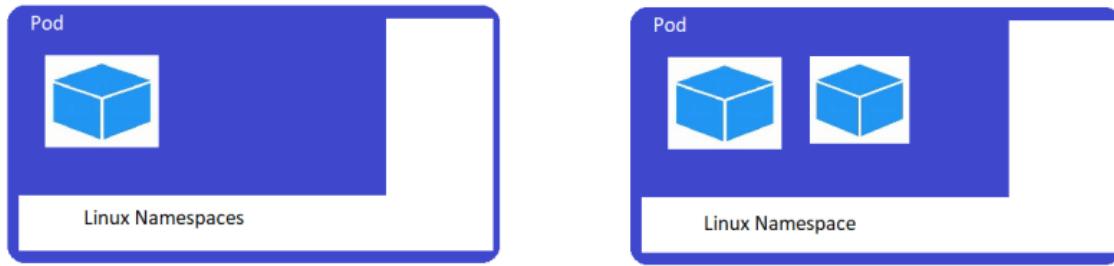
To interact with k8s cluster we have two major options

- programmatically by using REST API with json payloads
- kubectl command line by using YAML manifests
- When we interact with kubectl we create yaml manifest which has minimum details required where we express what we want rather than how it is done.
- when we work with clusters especially container clusters we embrace cattle mindset (pet vs cattle)

Pods in k8s

- A Pod is smallest unit of creation in k8s.
- Container will exist inside Pod.

- A Pod is collection of application containers and volumes running inside the same execution environment
- Each container in a pod runs with in its own cgroup, but they share a number of Linux namespaces
- Each Pod gets a unique IP address in k8s cluster. The containers running inside the Pod share the same IP Address and port space, have the same host name



- A Pod can have any number of containers, but ideally it's not a good idea to run multiple containers in a Pod.
- A Pod should represent a microservice/application so running one container is considered as best idea.

Kubectl cheatsheet

- [Refer Here](#)

kubectl has two primary commands to obtain information

- get
- describe

Pod:

- Smallest unit in kubernetes.
- Pod contains a container
- Pods can contain more than one container and the extra containers are referred as side-car containers
- Scaling the application in k8s is scaling pods not containers
- To create pods (anything) in k8s we have two approaches
- **Imperative:** we create objects using command line
- **Declarative:** We create manifests i.e. yaml files where we express what we want
- Imperative way of creating pods

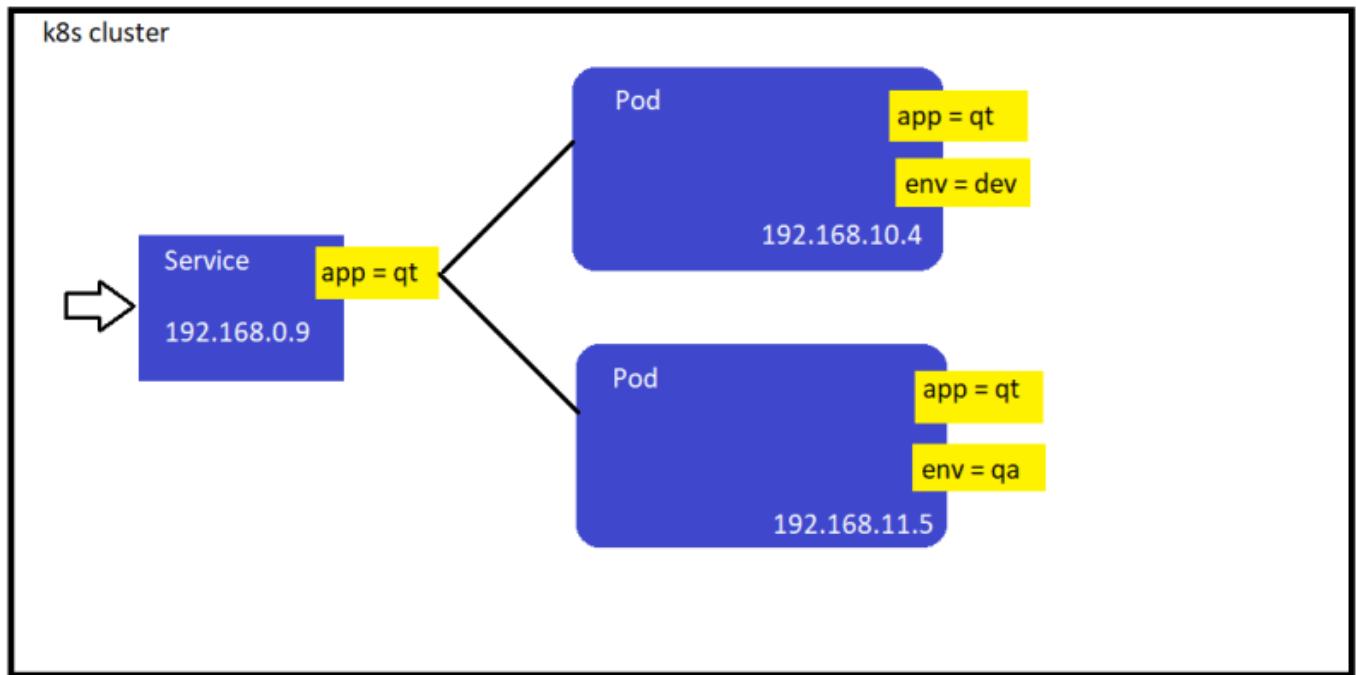
```
#kubectl run --help
```

```
#kubectl run hello-pod --image jenkins/jenkins:lts-jdk11
```

```
#kubectl delete pods hello-pod
```

K8s Service

- K8s service when created gets a cluster ip which is virtual in nature, when any other resource tries to access the service using cluster ip it forwards to request to one of the pod matching labels



- An easier way to create the service is by using `kubectl expose`
 - K8s service uses a label selectors which will find all the pods with matching labels and will load-balance across all the pods
 - since the cluster ip is virtual, its stable and it is appropriate to give it a DNS address.
 - K8s provides a DNS service exposed to Pods running in cluster.
 - Lets create a K8s service [Refer Here](#) for the changes.
 - Service should not forward the request to faulty pods, as this might impact application access, so lets see what can be done over here.
-
- **Readiness Checks/Probes:**
 - This is to check whether the application in container running in Pod is ready to serve requests or not
 - If this check fails the k8s removes the Ip address of Pods from all endpoints in Services
-
- **Liveness Checks/Probes:**

- This is to check whether or not application in container is running or not.
- K8s restarts containers if this check fails based on restart policy

- [Refer Here](#) for writing checks or probes
- [Refer Here](#) for the sample probes

- Accessing the service from outside cluster: For this in k8s service we have 3 options
- **Node Port:** Where you expose service on some port of the node
- **Load Balancer Integration:** Generally in all the managed clusters like AKS, EKS, GKE cluster is configured to integrate with external load balancers, so this can be used to expose the service
- **External Name:** Will be a DNS record which you can add to existing DNS servers

Kubernetes Deployment

- We know that Replica set manage pods.
- Deployment manages replica set.
- K8s is a self-healing system. The top level deployment object manages replica set, when you adjust number of replicas it will not match desired state so it will scale up or down
- Deployment allows us to deploy the newer versions of the applications by ensuring it supports all the necessary options to minimize/make zero down time deployments and rollout to a new version or roll back to the older version.
- [Refer Here](#) for the changeset & [Refer Here](#) for the fix for wrong indentation.

K8s Storage Solutions

- **Volumes:** k8s Volume has a lifecycle equal to Pod. Once the Pod is deleted, the data will be lost
- **Persistent Volumes:** These volumes have lifecycle independent of Pod, So data will not be lost
- To create Persistent Volumes, We have two options
- Manual Provisioning: In this case we need to manually create the storage to be used by the k8s cluster
- Dynamic Provisioning: In this K8s will try to automatically create the storage based on the details provided. We prefer this approach on clouds
- K8s has the following types of persistent volumes [Refer Here](#)
- In K8s Storage Class provides a way for administrators to describe the classes of storage.
- If you are using managed k8s, it will already have some storage classes defined
- aks
- eks
- In K8s, we need to understand the relation between storage classes, Persistent Volumes and Persistent Volume Claims

What is Persistent Volume Claim (PVC)

- PVC is request for platform to create a persistent volume (PV) and attach it to your pods.
- Storage classes define the details (hardware details(ssd/hdd/block/file))

Pod -> PVC -> PV -> Target Machine (Type of this is defined by SC)

- When we are using the volume to mounted on multiple pods [Refer Here](#)
- [Refer Here](#) for the changeset containing changes to create a PVC which create a PV of size 1 GB of type managed-premium
- **Stateful Sets:** For storage solutions, where each instance of the application in Pod requires its own private volume then, we go for stateful sets. Stateful sets use PVC to claim PV.

Helm

- Helm is a package-manager for k8s.
- When we want to create a package for our application, we create helm-chart.

7. Terraform

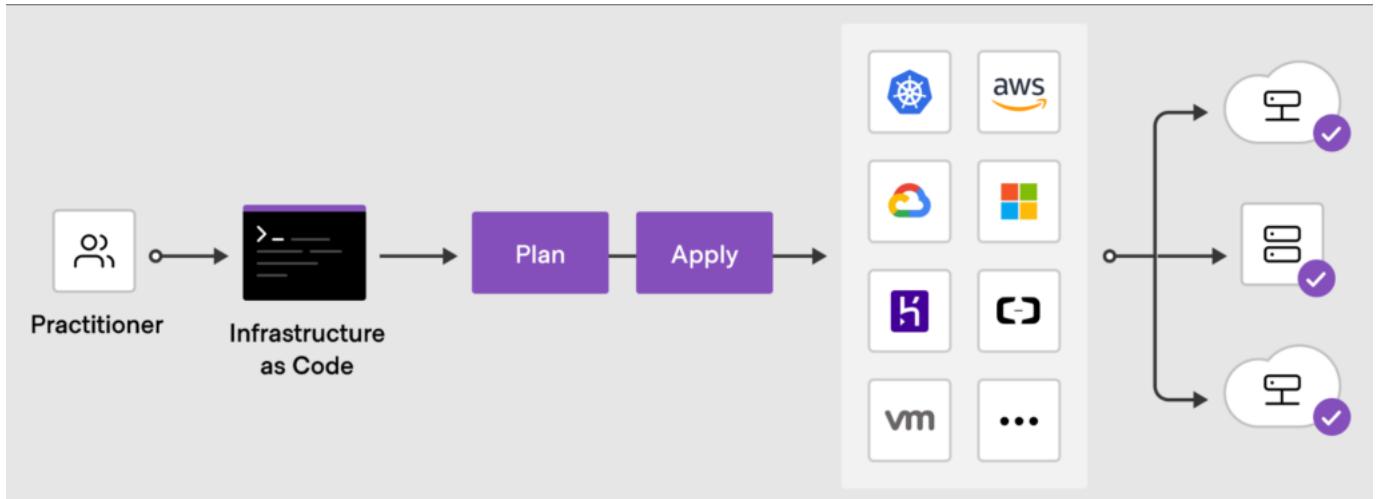
Infrastructure as a code

Infrastructure as Code it is the process of managing infrastructure in a file or files rather than manually configuring resources in a user interface. A resource in this instance is any piece of infrastructure in a given environment, such as a virtual machine, security group, network interface, etc.

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

Configuration files describe to Terraform the components needed to run a single application or your entire datacenter. Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

The infrastructure Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.



LOCAL VALUES

A local value assigns a name to an [expression](#), allowing it to be used multiple times within a module without repeating it.

Comparing modules to functions in a traditional programming language: if [input variables](#) are analogous to function arguments and [outputs values](#) are analogous to function return values, then *local values* are comparable to a function's local temporary symbols.

Note: For brevity, local values are often referred to as just “locals” when the meaning is clear from context.

Declaring a local value:

A set of related local values can be declared together in a single locals block

```
locals {
  service_name = "forum"
  owner      = "Community Team"
}
```

The expressions assigned to local value names can either be simple constants like the above, allowing these values to be defined only once but used many times, or they can be more complex expressions that transform or combine values from elsewhere in the module:

When to use local values:

Local values can be helpful to avoid repeating the same values or expressions multiple times in a configuration, but if overused they can also make a configuration hard to read by future maintainers by hiding the actual values used.

Use local values only in moderation, in situations where a single value or result is used in many places *and* that value is likely to be changed in future. The ability to easily change the value in a central place is the key advantage of local values.

ENVIRONMENT VARIABLES

We can create an export with our variable before execute terraform plan, and overwrite the value on the .tf files, for example export TF_VAR_vpcname=envvpc. This is useful for pass secrets or sensitive information in a secure form.

CLI VARIABLES

Another way to set variables is by using the command-line, for example terraform plan - var="vpcname=cliname"

TFVARS FILES

Passing variables inside a file, this is possible create a file called terraform.tfvars this file can be in a yaml or json notation, and is very simple, and also we can add maps, for example:

```
vpcname = "tfvarsname"  
port = 22  
policy = {  
    test = 1  
    debug = "true"  
}
```

Note: The terraform.tfvars file is used to define variables and the .tf file declare that the variable exists.

MASTER THE WORKFLOW

3 types of users, the workflow change according to the user

- Individual
 - Write: Create the Terraform files
 - Plan: Run Terraform plan and check
 - Create: Create the infrastructure
- Team
 - Write: Create the Terraform files and Checkout the latest code
 - Plan: Run Terraform Plan and raise a Pull-Request
 - Create: Merge and create
- Terraform Cloud
 - Write: Use Terraform Cloud as your development environment (statefiles, variables and secrets on Terraform Cloud)
 - Plan: When a PR is raised, Terraform Plan is run
 - Create: Before merging a second plan is run before approval to create

TERRAFORM INIT

The terraform init command is used to initialize a working directory containing Terraform configuration files. It is safe to run this command multiple times, , this command will never delete your existing configuration or state. During init, the root configuration directory is consulted for [backend configuration](#) and the chosen backend is initialized using the given configuration settings.

Link: <https://www.terraform.io/docs/commands/init.html>

TERRAFORM VALIDATE

The terraform validate command validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc.

Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state.

It is thus primarily useful for general verification of reusable modules, including correctness of attribute names and value types.

It is safe to run this command automatically, for example as a post-save check in a text editor or as a test step for a re-usable module in a CI system.

Note: Validation requires an initialized working directory with any referenced plugins and modules installed.

TERRAFORM PLAN

The terraform plan command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

TERRAFORM APPLY

The terraform apply command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.

TERRAFORM DESTROY

The terraform destroy command is used to destroy the Terraform-managed infrastructure.

Subcommands:

FMT

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. The canonical format may change in minor ways between Terraform versions, so after upgrading Terraform it is recommended to proactively run fmt

TAINT

The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply. Taint force the recreation. This command *will not* modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next [plan](#) will show that the resource will be destroyed and recreated and the next [apply](#) will implement this change.

Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. For example: re-running provisioners will cause the node to be different or rebooting the machine from a base image will cause new startup scripts to run.

Example:

```
$ terraform taint aws_vpc.myvpc
The resource aws_vpc.myvpc in the module root has been marked as tainted.
```

Another example if we want taint the resource “aws_instance” “baz” resource that lives in module bar which lives in module foo.

```
terraform taint module.foo.module.bar.aws_instance.baz
```

Link: <https://www.terraform.io/docs/internals/resource-addressing.html>

UNTAINT

The terraform untaint command manually unmark a Terraform-managed resource as tainted, restoring it as the primary instance in the state.

Note: This command *will not* modify infrastructure, but does modify the state file in order to unmark a resource as tainted.

```
$ terraform untaint aws_vpc.myvpc
Resource aws_vpc.myvpc2 has been successfully untainted.
```

IMPORT

The terraform import command is used to [import existing resources](#) into Terraform.

Import will find the existing resource from ID and import it into your Terraform state at the given ADDRESS.

ADDRESS must be a valid [resource address](#). Because any resource address is valid, the import command can import resources into modules as well directly into the root of your state.

ID is dependent on the resource type being imported. For example, for AWS instances it is the instance ID (i-abcd1234) but for AWS Route53 zones it is the zone ID (Z12ABC4UGMOZ2N).

Usage: `terraform import [options] ADDRESS ID`

Example:

```
$ terraform import aws_vpc.vpcimport vpc-06f0e46d612
```

Note: `terraform import` command can import resources directly into modules

Link: <https://www.terraform.io/docs/commands/import.html>

SHOW

The `terraform show` command is used to provide human-readable output from a state or plan file. This can be used to inspect a plan to ensure that the planned operations are expected, or to inspect the current state as Terraform sees it.

Usage: `terraform show [options] [path]`

You may use `show` with a path to either a Terraform state file or plan file. If no path is specified, the current state will be shown.

WORKSPACE INTRODUCTION

The `terraform workspace` command is used to manage [workspaces](#). It is useful to split and separate the state files.

This command is a container for further subcommands such as `list`, `select`, `new`, `delete` and `show`

Usage: `terraform workspace <subcommand> [options] [args]`

If we never create a workspace we use the default workspace

```
# terraform workspace list
* default
```

Note: For local state, Terraform stores the workspace states in a directory called `terraform.tfstate.d`. This directory should be treated similarly to local-only `terraform.tfstate`

Note: Terraform Cloud and Terraform CLI both have features called “workspaces,” but they’re slightly different. CLI workspaces are alternate state files in the same working directory; they’re

a convenience feature for using one configuration to manage multiple similar groups of resources.

CREATION OF A WORKSPACE

The terraform workspace new command is used to create a new workspace.

```
$ terraform workspace new dev  
Created and switched to workspace "dev"!
```

You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.

Now, if we execute a terraform plan in our environment, the plan will try to show many resources to create, because is a different state file.

SHOW WORKSPACE

The terraform workspace show command is used to output the current workspace.

```
$ terraform workspace show  
dev
```

SELECT WORKSPACE

The terraform workspace select command is used to choose a different workspace to use for further operations. First we need to execute a terraform workspace list to know the workspaces names.

```
$ terraform workspace select default  
Switched to workspace "default".
```

DELETE THE WORKSPACE

The terraform workspace delete command is used to delete an existing workspace.

```
$ terraform workspace delete dev  
Deleted workspace "dev"!
```

STATE LIST

The terraform state list command is used to list resources within a [Terraform state](#)

The command will list all resources in the state file matching the given addresses (if any). If no addresses are given, all resources are listed.

Example list all resources:

```
$ terraform state list
aws_instance.foo
aws_instance.bar[0]
aws_instance.bar[1]
module.elb.aws_elb.main
```

Example Filtering by Resource

```
$ terraform state list aws_instance.bar
aws_instance.bar[0]
aws_instance.bar[1]
```

Link: <https://www.terraform.io/docs/commands/state/list.html>

STATE PULL

The terraform state pull command is used to manually download and output the state from [remote state](#). This command also works with local state (but is not very useful because we can see the local file)

This command will download the state from its current location and output the raw format to stdout.

This is useful for reading values out of state (potentially pairing this command with something like [jq](#)). It is also useful if you need to make manual modifications to state.

STATE MV

The terraform state mv command is used to move items in a [Terraform state](#). This command can move single resources, single instances of a resource, entire modules, and more. This command can also move items to a completely different state file, enabling efficient refactoring.

Usage: terraform state mv [options] Source Destination

This can be used for simple resource renaming, moving items to and from a module, moving entire modules, and more. And because this command can also move data to a completely new state, it can also be used for refactoring one configuration into multiple separately managed Terraform configurations.

Example:

```
terraform state mv 'packet_device.worker' 'packet_device.helper'
```

Link: <https://www.terraform.io/docs/commands/state/mv.html>

STATE RM

The terraform state rm command is used to remove items from the [Terraform state](#). This command can remove single resources, single instances of a resource, entire modules, and more.

Usage: `terraform state rm [options] ADDRESS...`

Items removed from the Terraform state are *not physically destroyed*. Items removed from the Terraform state are only no longer managed by Terraform. For example, if you remove an AWS instance from the state, the AWS instance will continue running, but `terraform plan` will no longer see that instance.

There are various use cases for removing items from a Terraform state file. The most common is refactoring a configuration to no longer manage that resource (perhaps moving it to another Terraform configuration/state).

Example remove a resource:

```
$ terraform state rm 'packet_device.worker'
```

Example remove a module:

```
$ terraform state rm 'module.foo'
```

TERRFORM MODULES

A module is a simple directory that contains other .tf files. Using modules we can make the code reusable. Modules are local or remote.

TERRAFORM REGISTRY

Is the place to find modules, theses modules are verified by Hashicorp

Link: <https://registry.terraform.io/>

MODULES INPUTS/OUTPUTS

For make modules inputs we use inputs variables. Example module code:

```
variable "dbname" {
    type = string
}
resource "aws_instance" "myec2db" {
    ami = "ami-01a6e"
    tags = {
        Name = var.dbname
    }
}
```

```
    }
}
```

Call to the module example:

```
module "dbserver" {
    source = "./db"
    dbname = "mydbserver"
}
```

Module outputs are very similar to module inputs, an example in a module output:

```
output "privateip" {
    value = aws_instance.myec2db.private_ip
}
```

When we use a module with an output, to use the output we need to specified in the call to our module, for example:

```
module "dbserver" {
    source = "./db"
    dbname = "mydbserver"
}

output "dbprivateip" {
    value = module.dbserver.privateip
}
```

CHILD MODULES

Terraform allow having child modules, modules within modules. Basically is a directory with tf files, with others directories with others sub modules. After add a subdirectory, remember to execute again terraform init

DEBUG IN TERRAFORM

Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value. This will cause detailed logs to appear on stderr.

Is not about to Terraform Client Debugging, panic errors, go errors, etc. Is useful to provide the logs to Hashicorp.

```
export TF_LOG=TRACE
```

You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs. TRACE is the most verbose and it is the default if TF_LOG is set to something other than a log level name.

Note: When TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled, and TF_LOG_PATH point to a specific file (not directory), for example TF_LOG_PATH=./terraform.log

8. Webserver & Application Server

Web server is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

An application server works with a web server to handle requests for dynamic content, such as servlets, from web applications. A web server uses a web server plug-in to establish and maintain persistent HTTP and HTTPS connections with an application server.

Ex. Apache, Microsoft's Internet Information Services (IIS) and Nginx

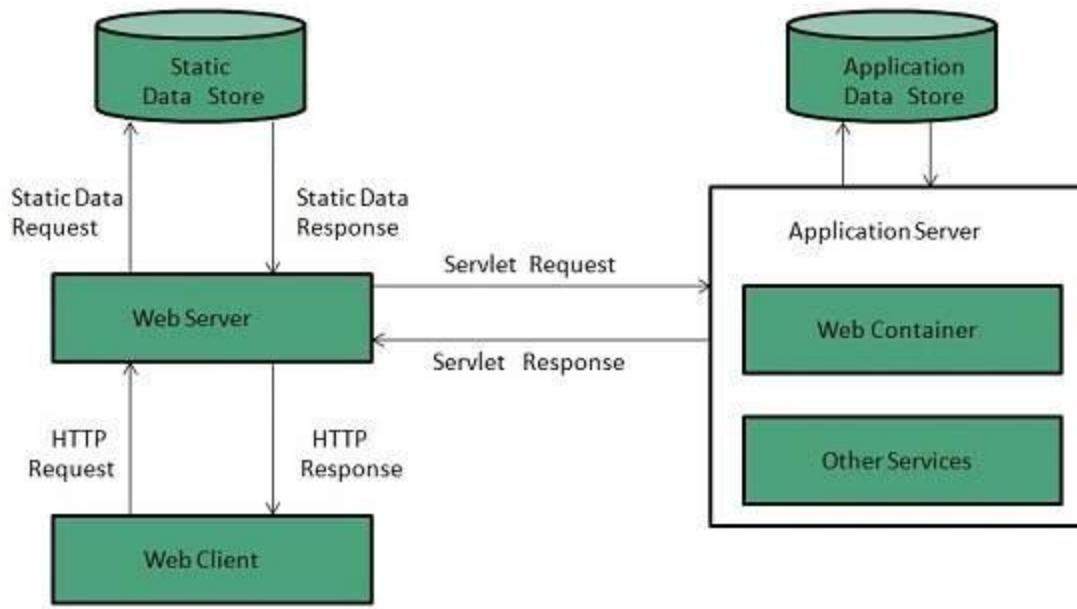
Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database

Key Points-

- When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.



- If the requested web page is not found, web server will send an **HTTP response:Error 404 Not found.**
- if client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

Application Server-

An application server is a Java™ Virtual Machine (JVM) that runs user applications. The application server collaborates with the web server to return a dynamic, customized response to a client request. The client request can consist of servlets, JavaServer Pages (JSP) files, and enterprise beans, and their supporting classes.

An application server executes and provides user and/or other app access when utilizing the installed application's business/functional logic.

Key required features of an application server include data redundancy, high availability, load balancing, user management, data/application security and a centralized management interface. Moreover, an application server may be connected by enterprise systems, networks or intranet and remotely accessed via the Internet.

Ex. Jboss, Weblogic, Websphere, Glassfish, Tcat Server, Apache Geronimo

	Web Server	Application Server
General	Limited to handling HTTP requests.	Executes programs, routines, or scripts that support application construction.
Content	Limited to sent static HTML content for display in a Web Browser.	Provides access to server-side logic (Server Applications)
Visual Display	Adding content extensions are technically possible, but time-consuming, difficult to use, and maintain.	Includes web server within a complete integrated application server framework.
Scope	May be used alone in certain limited circumstances, or as a component in an application server.	May be utilized as a GUI engine for remote displays where server applications dynamically create the UI in a browser.
Limitation	Content creation is time consuming and lack-luster for server-side applications.	Supports any web application, including Dynamic Content and a Modern UI.

8.Deployment Strategies

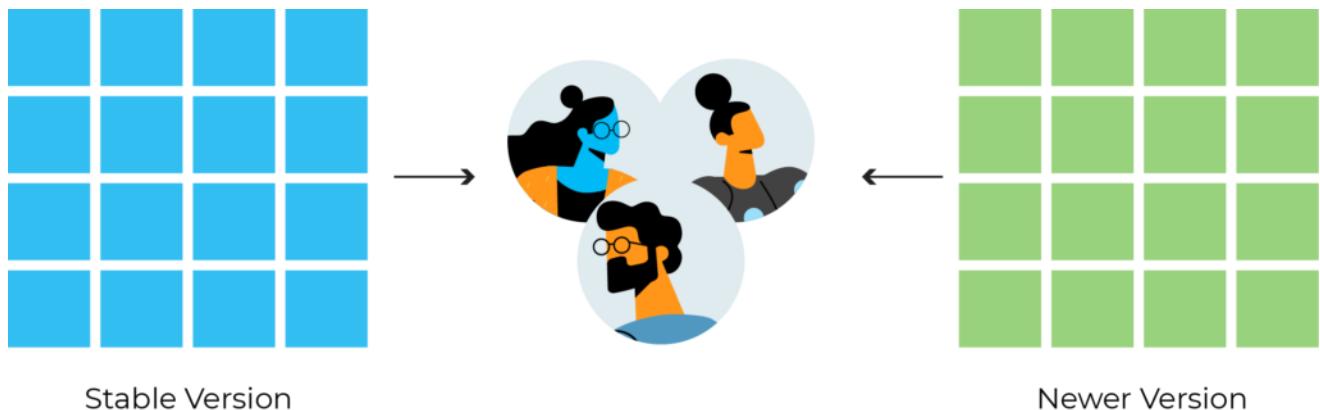
Types of Deployment Strategies –

1. Blue/Green Deployment

In this type of deployment strategy, the new version of the software runs alongside the old version. Note that you can also refer to this as red/black deployment strategy in some cases.

Here, the stable or the older version of the application is always blue or red, while the newer version is green or black.

After the new version has been tested and certified to meet all the requirements, the load balancer automatically switches the traffic from the older version to the newer version.

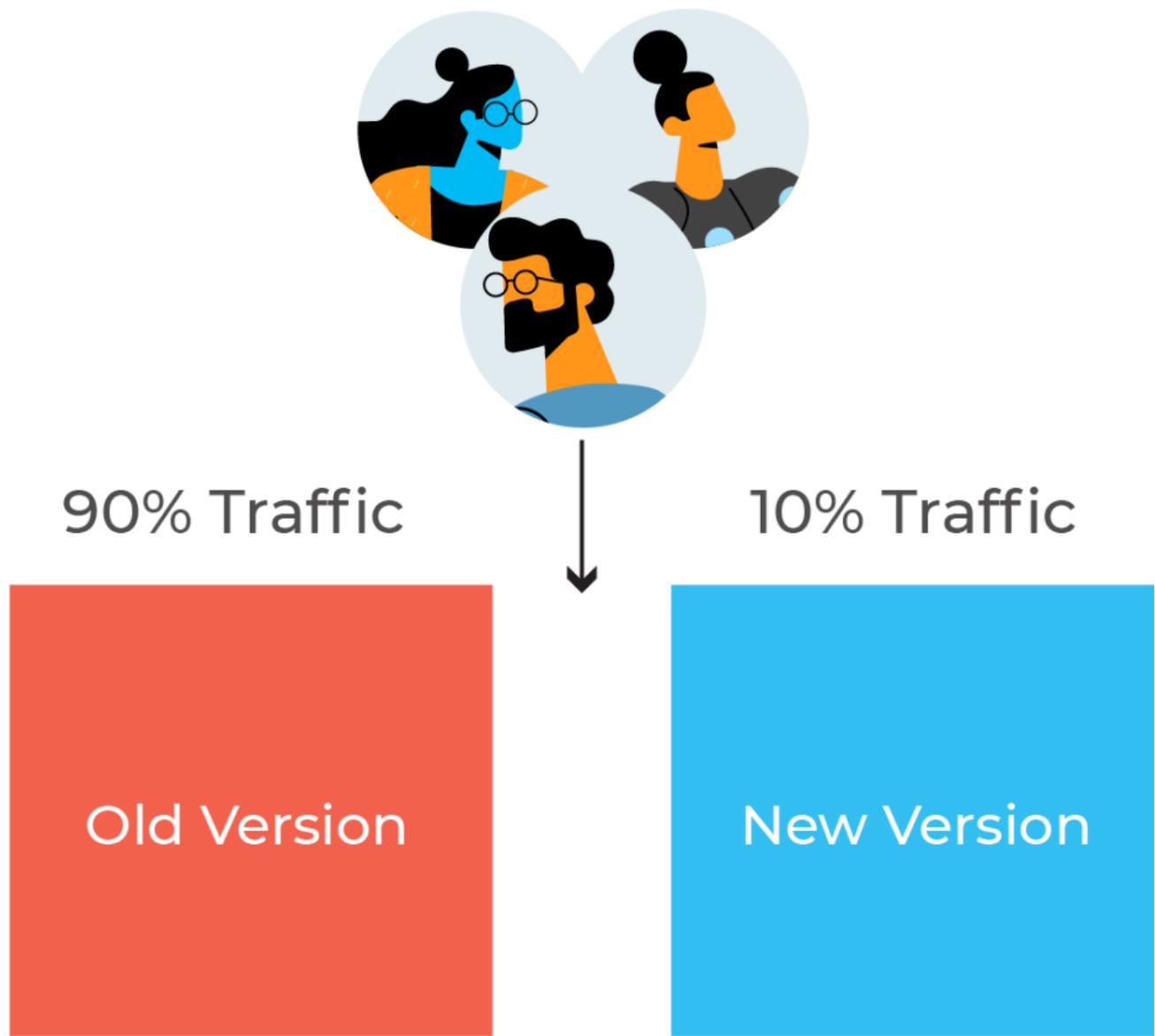


The major advantage of this strategy is that it avails a quick update or rollout of a new application version. However, its main disadvantage is that it is costly because you must run both the new and old versions concurrently. Engineers mostly use this method in mobile app development and deployment.

2. Canary Deployment

In canary deployment, the deployment team sets up the new version and then gradually shifts the production traffic from the older version to the newer version. For example, at a point in time during the deployment process, the older version might retain 90% of all traffic for the software while the newer version hosts 10% of the traffic.

This deployment technique helps the DevOps engineers test the stability of the new version. It uses live traffic from a subset of the end-users at different levels that varies as production occurs.

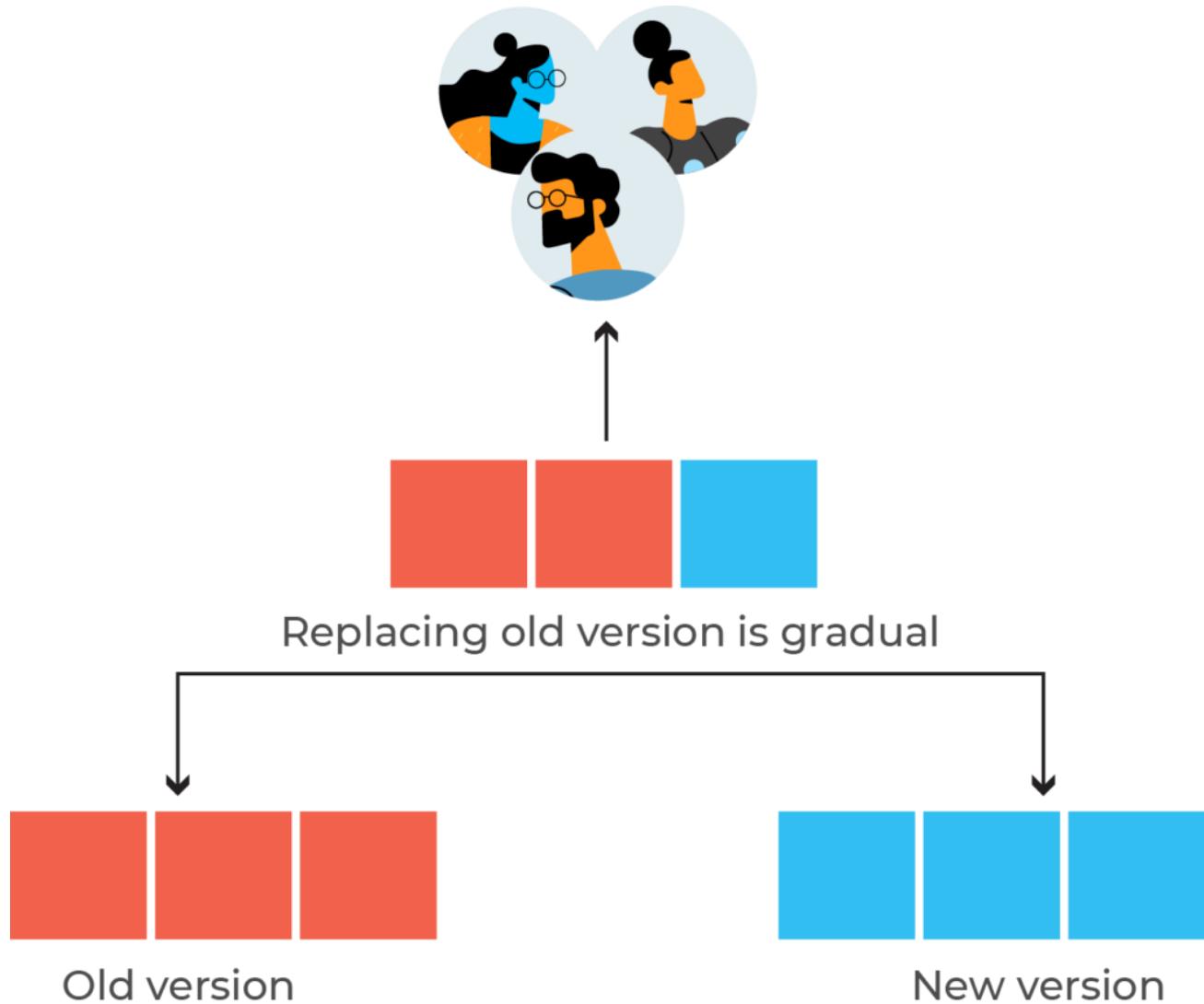


Canary deployment enables better performance monitoring. It also aids in a faster and better rollback of the software if the new version fails. However, it has a slow nature and a more time-consuming deployment cycle.

3. Ramped Deployment

The ramped deployment strategy gradually changes the older version to the new version. Unlike canary deployment, the ramped deployment strategy makes its switch by replacing instances of the old application version with the instances from the new application version one instance at a time. You can also call this method the rolling upgrade deployment strategy.

When developers replace all instances of the older version, they shut down the older version. The new version then controls the whole production traffic.



This strategy gives zero downtime and also enables performance monitoring. Nevertheless, the rollback duration is long in case there is an unexpected event. This is because the downgrading process to the initial version follows the same cycle, one instance at a time.

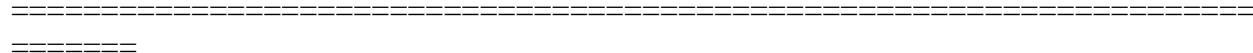
Basic Networking

A **computer network** is a system in which multiple computers are connected to each other to share information and resources.



Characteristics of a Computer Network-

1. Share resources from one computer to another.
2. Create files and store them in one computer, access those files from the other computer(s) connected over the network.
3. Connect a printer, scanner, or a fax machine to one computer within the network and let other computers of the network use the machines available over the network.



OSI Model:

The OSI 7 Layers-

7	Application Layer	Human-computer interaction layer, where applications can access the network services
6	Presentation Layer	Ensures that data is in a usable format and is where data encryption occurs
5	Session Layer	Maintains connections and is responsible for controlling ports and sessions
4	Transport Layer	Transmits data using transmission protocols including TCP and UDP
3	Network Layer	Decides which physical path the data will take
2	Data Link Layer	Defines the format of data on the network
1	Physical Layer	Transmits raw bit stream over the physical medium

How to remind OSI Layers-

Please Do Not Throw Sausage Pizza Away

Advantages of OSI Model

The OSI model helps users and operators of computer networks:

1. Determine the required hardware and software to build their network.

2. Understand and communicate the process followed by components communicating across a network.
3. Perform troubleshooting, by identifying which network layer is causing an issue and focusing efforts on that layer.

The OSI model helps network device manufacturers and networking software vendors:

1. Create devices and software that can communicate with products from any other vendor, allowing open interoperability
 2. Define which parts of the network their products should work with.
 3. Communicate to users at which network layers their product operates – for example, only at the application layer, or across the stack.
-
-

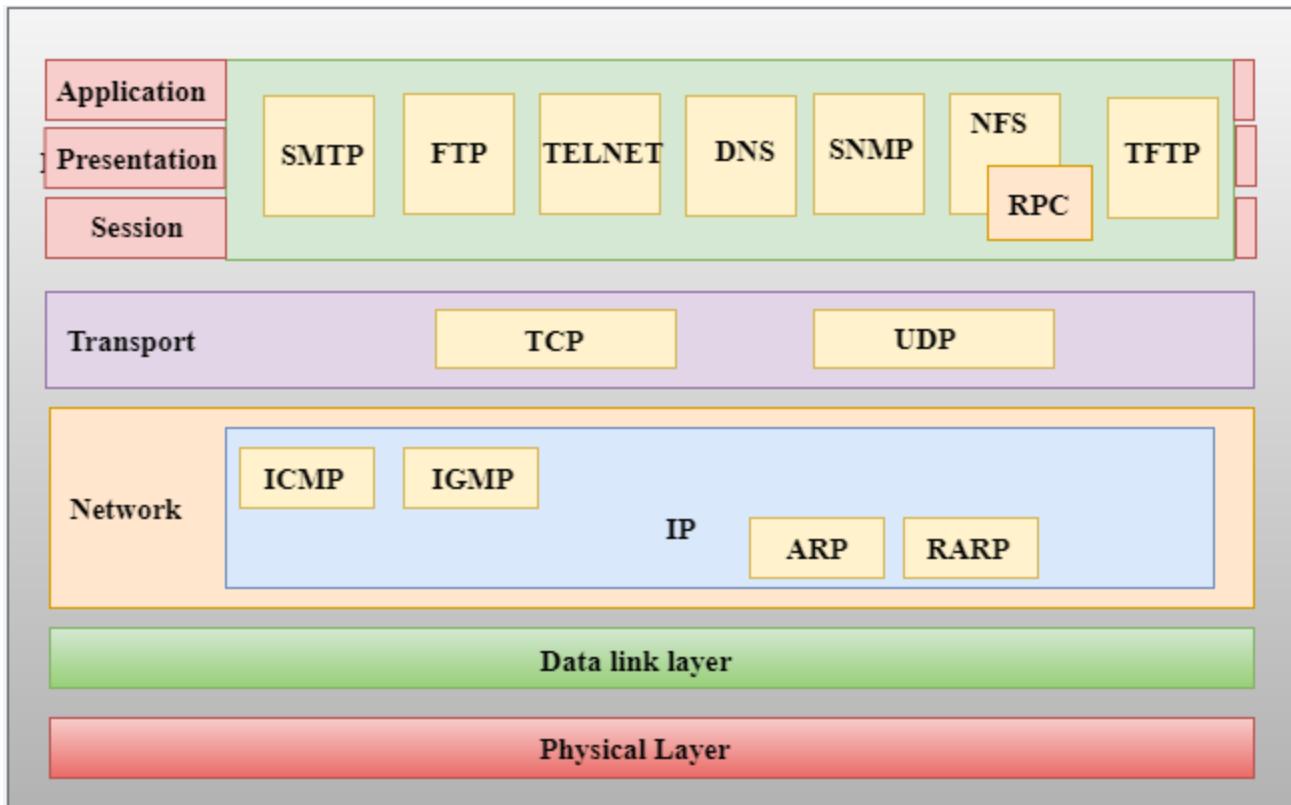
TCP/IP Model-

The purpose of TCP/IP model is to allow communication over large distances. TCP/IP stands for Transmission Control Protocol/ Internet Protocol. TCP/IP Stack is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.

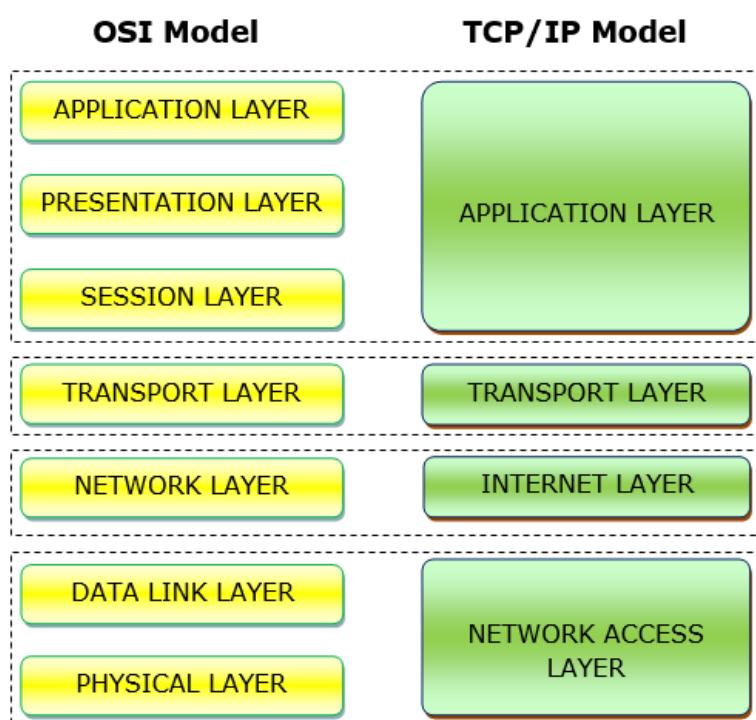
The TCP/IP model was developed prior to the OSI model.

The TCP/IP model consists of five layers: the application layer, transport layer, network layer, data link layer and physical layer.

TCP/IP is a hierarchical protocol made up of interactive modules, and each of them provides specific functionality.



OSI vs. TCP/IP Model-



The Transfer Control Protocol/Internet Protocol (TCP/IP) is older than the OSI model and was created by the US Department of Defense (DoD). A key difference between the models is that TCP/IP is simpler, collapsing several OSI layers into one:

- OSI layers 5, 6, 7 are combined into one Application Layer in TCP/IP
- OSI layers 1, 2 are combined into one Network Access Layer in TCP/IP – however TCP/IP does not take responsibility for sequencing and acknowledgement functions, leaving these to the underlying transport layer.

Other important differences:

- TCP/IP is a functional model designed to solve specific communication problems, and which is based on specific, standard protocols. OSI is a generic, protocol-independent model intended to describe all forms of network communication.
- In TCP/IP, most applications use all the layers, while in OSI simple applications do not use all seven layers. Only layers 1, 2 and 3 are mandatory to enable any data communication.

=====

Protocols-

FTP (File Transfer Protocol) Port-21

It is used for file transmission between internetwork nodes.

SMTP (Simple Mail Transfer Protocol) Port-25

It can be used for exchanging email.

TELNET Port-23

TELNET represents Terminal Network. It allows the client to create host-based software by initiating one of the host terminals. It also supports connectivity between the diverse operating framework.

DNS (Domain Name Systems) Port-53

The DNS can change the domain name into IP addresses. The TCP/IP protocol needs the IP address that recognizes linking a host to the computer network.

HTTP (Hypertext Transfer Protocol) Port-80

HTTP is an internet protocol created for a particular software, the World Wide Web (WWW).

HTTPS (Hypertext Transfer Protocol Secure) Port-443

HTTPS is Hypertext Transfer Protocol Secure. The HTTP protocol does not provide the security of the data, while HTTPS ensures the security of the data. Therefore, we can say that HTTPS is a secure version of the HTTP protocol.

This protocol allows transferring the data in an encrypted form.

RDP Remote Desktop Protocol (3389)

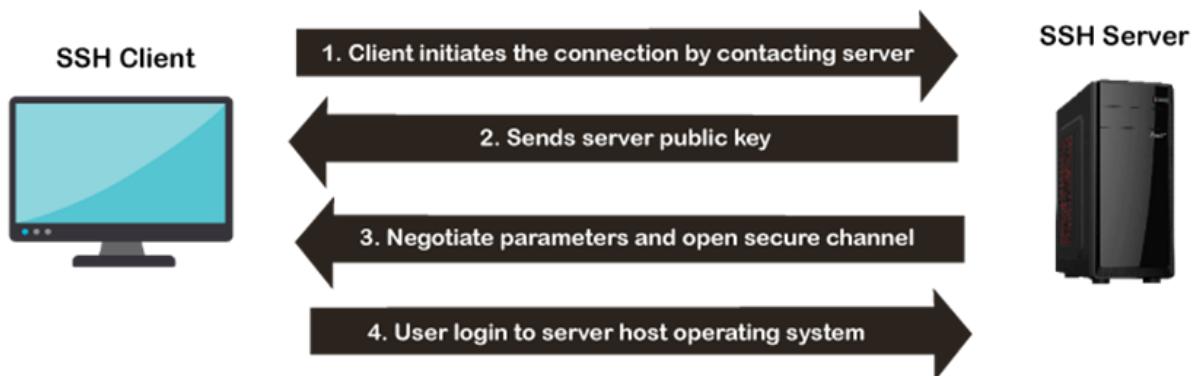
Remote Desktop Protocol is a proprietary protocol developed by Microsoft which provides a user with a graphical interface to connect to another computer over a network connection.

SSH (Secure Shell) Port- 22

It provides secure access to users and automated processes.

It is an easy and secure way to transfer files from one system to another over an insecure network.

ssh UserName@SSHserver.test.com



What is an IP Address?

All the computers of the world in the Internet network communicate with each other with underground or underwater cables or wirelessly. If I want to download a file from the internet or load a web page or literally do anything related to the internet, my computer must have an address so that other computers can find and locate mine in order to deliver that particular file or webpage that I am requesting. In technical terms, that address is called IP Address or Internet Protocol Address.

IP Address is of two types:

1. IPv4:

Internet Protocol version 4. It consists of 4 numbers separated by the dots. Each number can be from 0-255 in decimal numbers. But computers do not understand decimal numbers, they instead change them to binary numbers which are only 0 and 1. Therefore, in binary, this (0-255) range can be written as (00000000 – 11111111). Since each number N can be represented by a group of 8 digit binary digits. So, a whole IPv4 binary address can be represented by 32-bits of binary digits. In IPv4, a unique sequence of bits is assigned to a computer, so a total of (2^{32}) devices approximately = 4,294,967,296 can be assigned with IPv4.

IPv4 can be written as:

189.123.123.90

Address Classes	RANGE	Bit Pattern of 1 st byte	Decimal Range	Default Subnet Mask	Reserved for
A	1.0.0.0 to 126.255.255.255	0xxxxxxxx	1 to 127	255.0.0.0	Governments
B	128.0.0.0 to 191.255.255.255	10xxxxxx	128-191	255.255.0.0	Medium Companies
C	192.0.0.0 to 223.255.255.255	110xxxxx	192-223	255.255.255.0	Small Companies
D	224.0.0.0 to 239.255.255.255	1110xxxx	224-239	Not Applicable	Reserved for Multicasting
E	240.0.0.0 to 255.255.255.255	11110xxx	240-255	Not Applicable	Experimental or future use

2. **IPv6**: But, there is a problem with the IPv4 address. With IPv4, we can connect only the above number of 4 billion devices uniquely, and apparently, there are much more devices in the world to be connected to the internet. So, gradually we are making our way to **IPv6 Address** which is a 128-bit IP address. In human-friendly form, IPv6 is written as a group of 8 hexadecimal numbers separated with colons(:). But in the computer-friendly form, it can be written as 128 bits of 0s and 1s. Since, a unique sequence of binary digits is given to computers, smartphones, and other devices to be connected to the internet. So, via IPv6 a total of (2^{128}) devices can be assigned with unique addresses which are actually more than enough for upcoming future generations.

IPv6 can be written as:

2011:0bd9:75c5:0000:0000:6b3e:0170:8394

Classification of IP Address

An IP address is classified into the following types:

1. Public IP Address: This address is available publicly and it is assigned by your network provider to your router, which further divides it to your devices.

2. Private IP Address: This is an internal address of your device which are not routed to the internet and no exchange of data can take place between a private address and the internet.

Windows Server

Windows Server

Large organizations depend on Windows **Active Directory (AD)** to maintain order in the chaos that is managing users, computers, permissions, and file servers.

What is Active Directory?

Active Directory is a Microsoft product that operates on Windows Server. It is a database and set of services developed to help you with access, management, and permissions for your network resources. The organizational data is stored as an object in the Active Directory, and it can be in the form of devices, files, users, applications, groups, or shared folders. In addition, these objects can be categorized by their name or attribute.

The directory or database stores critical information related to your IT environment, including essential details about users, user permissions, and computers. In short, it helps you control various activities going on in your IT environment. Most importantly, AD also ensures user authentication, generally via user ID and passwords, and allows them to access data they're authorized to use.

How does Active Directory work?

In the AD, the domain is the primary unit in a logical structure. The objects named under the same directory database, trust relationships, and security policies with other domains are called Domains. Each domain will store data about objects belonging to that domain only.

Settings and security policies, for example, Access Control Lists (ACLs), admin rights, etc., do not pass from one domain to another. In short, the admin can set policies only for the domain they belong to. Domains allow admins to set boundaries for objects and handle security policies for shared network resources.

One of the primary Active Directory services is the AD DS (Active Directory Domain Services), a crucial part of the Windows Server OS. The AD DS runs on servers known as Domain Controllers (DCs). An enterprise usually has multiple DCs, and each of these controllers has a copy of the main directory for the domain. Any changes made to the directory on one DC- for example, deleting a user account or changing a password are all applied to the other DCs in a domain to keep them up-to-date.

Why is Active Directory So Important?

To simplify and understand the concept of AD better, consider Active Directory as the “Contacts” application on your mobile phone. The Contacts app itself acts as an Active Directory, while individual contacts in the app would be its “objects”. The values stored in each object, such as phone number, address, email, etc., would be your Active Directory. The only difference is that objects like in the mobile app aren’t just limited to people, but AD may also contain group objects such as printers, computers, devices, etc.

Active Directory is vital for organizations as it helps you efficiently manage company users, computers, devices, and applications. For example, IT managers can leverage Active Directory to systematically organize company data in a hierarchy structure, which states which users or computers belong to which network, or which users have access to which network resources, and so forth.

How to Setup Active Directory Domain Controller?

A domain controller contains many computers on the network and allows the system administrators to manage them from the central place. It is a server or computer used to authenticate other computers throughout the network. It stores the login credentials of all other computers and printers in the network.

This section will show you how to install Active Directory Domain Services and set up a domain controller on Windows server 2019.

Follow the below steps to install Active Directory Domain Services:

Step 1 – Login to Windows server 2019 as an administrator and open the Server Manager as shown below:

Dashboard

Local Server

All Servers

WELCOME TO SERVER MANAGER

QUICK START

WHAT'S NEW

LEARN MORE

1 Configure this local server

2 Add roles and features

3 Add other servers to manage

4 Create a server group

5 Connect this server to cloud services

ROLES AND SERVER GROUPS

Roles: 0 | Server groups: 1 | Servers total: 1

Local Server	1
Manageability	1
Events	
Services	1
Performance	

All Servers	1
Manageability	1
Events	
Services	1
Performance	

Step 2 – Click on the Add Roles and Features. This will open the Add Roles and Features Wizard as shown below:

Add Roles and Features Wizard

Before you begin

DESTINATION SERVER
CLOUD-7H24HNKQS

Before You Begin

Installation Type

Server Selection

Server Roles

Features

Confirmation

Results

This wizard helps you install roles, role services, or features. You determine which roles, role services, or features to install based on the computing needs of your organization, such as sharing documents, or hosting a website.

To remove roles, role services, or features:
[Start the Remove Roles and Features Wizard](#)

Before you continue, verify that the following tasks have been completed:

- The Administrator account has a strong password
- Network settings, such as static IP addresses, are configured
- The most current security updates from Windows Update are installed

If you must verify that any of the preceding prerequisites have been completed, close the wizard, complete the steps, and then run the wizard again.

To continue, click Next.

Skip this page by default

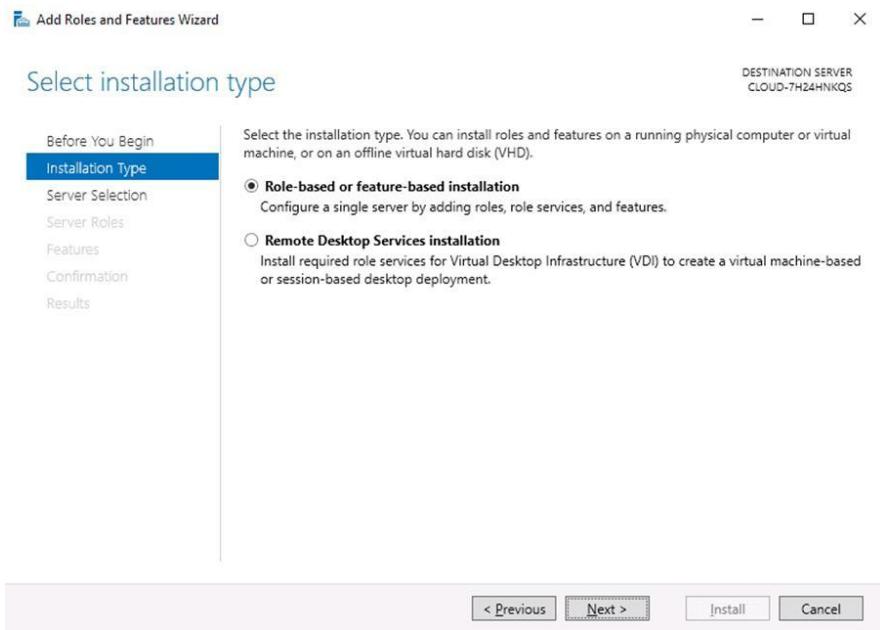
< Previous

Next >

Install

Cancel

Step 3 – Click on the Next button. You will be asked to select the installation type as shown below:



Step 4 – Select Add Roles and Features Wizard and click on the Next button. Next, you will be asked to select a destination server as shown below:

DESTINATION SERVER
CLOUD-7H24HNKQS

Select destination server

Before You Begin

Installation Type

Server Selection

Server Roles

Features

Confirmation

Results

Select a server or a virtual hard disk on which to install roles and features.

 Select a server from the server pool Select a virtual hard disk

Server Pool

Filter:

Name	IP Address	Operating System
CLOUD-7H24HNKQS	45.58.46.58	Microsoft Windows Server 2019 Datacenter

1 Computer(s) found

This page shows servers that are running Windows Server 2012 or a newer release of Windows Server, and that have been added by using the Add Servers command in Server Manager. Offline servers and newly-added servers from which data collection is still incomplete are not shown.

< Previous

Next >

Install

Cancel

Step 5 – Select “Select a server from the server pool” and click on the Next button. Next, you will be asked to select server roles as shown below

Select server roles

DESTINATION SERVER
CLOUD-7H24HNKQS

Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results

Select one or more roles to install on the selected server.

Roles

- Active Directory Certificate Services
- Active Directory Domain Services**
- Active Directory Federation Services
- Active Directory Lightweight Directory Services
- Active Directory Rights Management Services
- Device Health Attestation
- DHCP Server
- DNS Server
- Fax Server
- File and Storage Services (1 of 12 installed)
- Host Guardian Service
- Hyper-V
- Network Controller
- Network Policy and Access Services
- Print and Document Services
- Remote Access
- Remote Desktop Services
- Volume Activation Services
- Web Server (IIS) (3 of 43 installed)
- Windows Deployment Services

Description

Active Directory Domain Services (AD DS) stores information about objects on the network and makes this information available to users and network administrators. AD DS uses domain controllers to give network users access to permitted resources anywhere on the network through a single logon process.

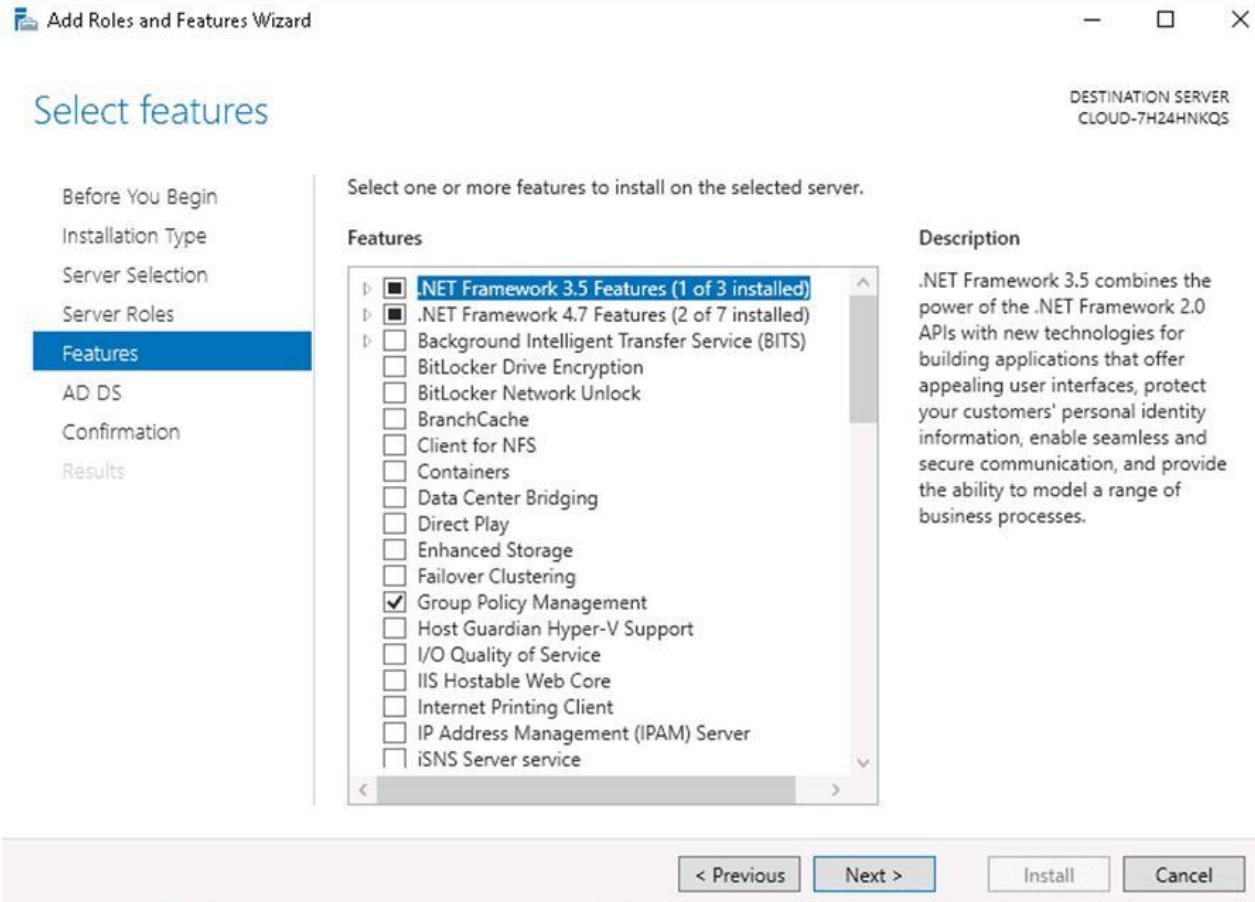
< Previous

Next >

Install

Cancel

Step 6 – Select Active Directory Domain Services and click on the Next button. You will be asked to select features as shown below



Step 7 – Leave all default settings and click on the Next button. Next, you should see the confirm installations selections page.

Confirm installation selections

DESTINATION SERVER
CLOUD-7H24HNKQS

Before You Begin

Installation Type

Server Selection

Server Roles

Features

AD DS

Confirmation

Results

To install the following roles, role services, or features on selected server, click Install.

 Restart the destination server automatically if required

Optional features (such as administration tools) might be displayed on this page because they have been selected automatically. If you do not want to install these optional features, click Previous to clear their check boxes.

Active Directory Domain Services

Group Policy Management

[Export configuration settings](#)[Specify an alternate source path](#)[< Previous](#)[Next >](#)[Install](#)[Cancel](#)

Step 8 – Click on the Install button to start the installation. Once the installation has been finished. You should see the following page.

Installation progress

DESTINATION SERVER
CLOUD-7H24HNKQS

Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD DS
Confirmation
Results

View installation progress

 Feature installation

Configuration required. Installation succeeded on CLOUD-7H24HNKQS.

Active Directory Domain Services

Additional steps are required to make this machine a domain controller.

[Promote this server to a domain controller](#)**Group Policy Management** You can close this wizard without interrupting running tasks. View task progress or open this page again by clicking Notifications in the command bar, and then Task Details.[Export configuration settings](#)[< Previous](#)[Next >](#)[Close](#)[Cancel](#)**Step 9 – Click on the Close button. You should see the following page.**

The screenshot shows the Windows Server Manager Dashboard. On the left, there's a navigation bar with links like 'Dashboard', 'Local Server', 'All Servers', 'AD DS', 'File and Storage Services', and 'IIS'. The main area has a 'WELCOME TO SERVER MANAGER' header. Below it, a large orange box contains the text '1 Configure this local server' and a numbered list from 2 to 5: 'Add roles and features', 'Add other servers to manage', 'Create a server group', and 'Connect this server to cloud services'. To the right of this are four cards representing different roles: AD DS, File and Storage Services, IIS, and Local Server. Each card lists 'Manageability', 'Events', 'Services', and 'Performance' under its respective role name.

Step 10– Click on the yellow notification icon. You should see the following page:

This screenshot is similar to the previous one, showing the Server Manager Dashboard. However, a yellow warning icon in the top right corner has been clicked, which has triggered a modal dialog. The dialog is titled 'Post-deployment Configuration' and contains the message 'Configuration required for Active Directory Domain Services at CLOUD-7H24HNKQ5. Promote this server to a domain controller'. It also shows a progress bar for 'Feature installation' with the status 'Configuration required. Installation succeeded on CLOUD-7H24HNKQ5.' and a link to 'Add Roles and Features'. The rest of the interface looks identical to the first screenshot.

Step 11 – Click on Promote this server to a domain controller. You should see the deployment configuration page:

Additional Options

TARGET SERVER
CLOUD-7H24HNKQS

Deployment Configuration

Domain Controller Options

DNS Options

Additional Options

Paths

Review Options

Prerequisites Check

Installation

Results

Verify the NetBIOS name assigned to the domain and change it if necessary

The NetBIOS domain name:

MYDOMAIN

[More about additional options](#)

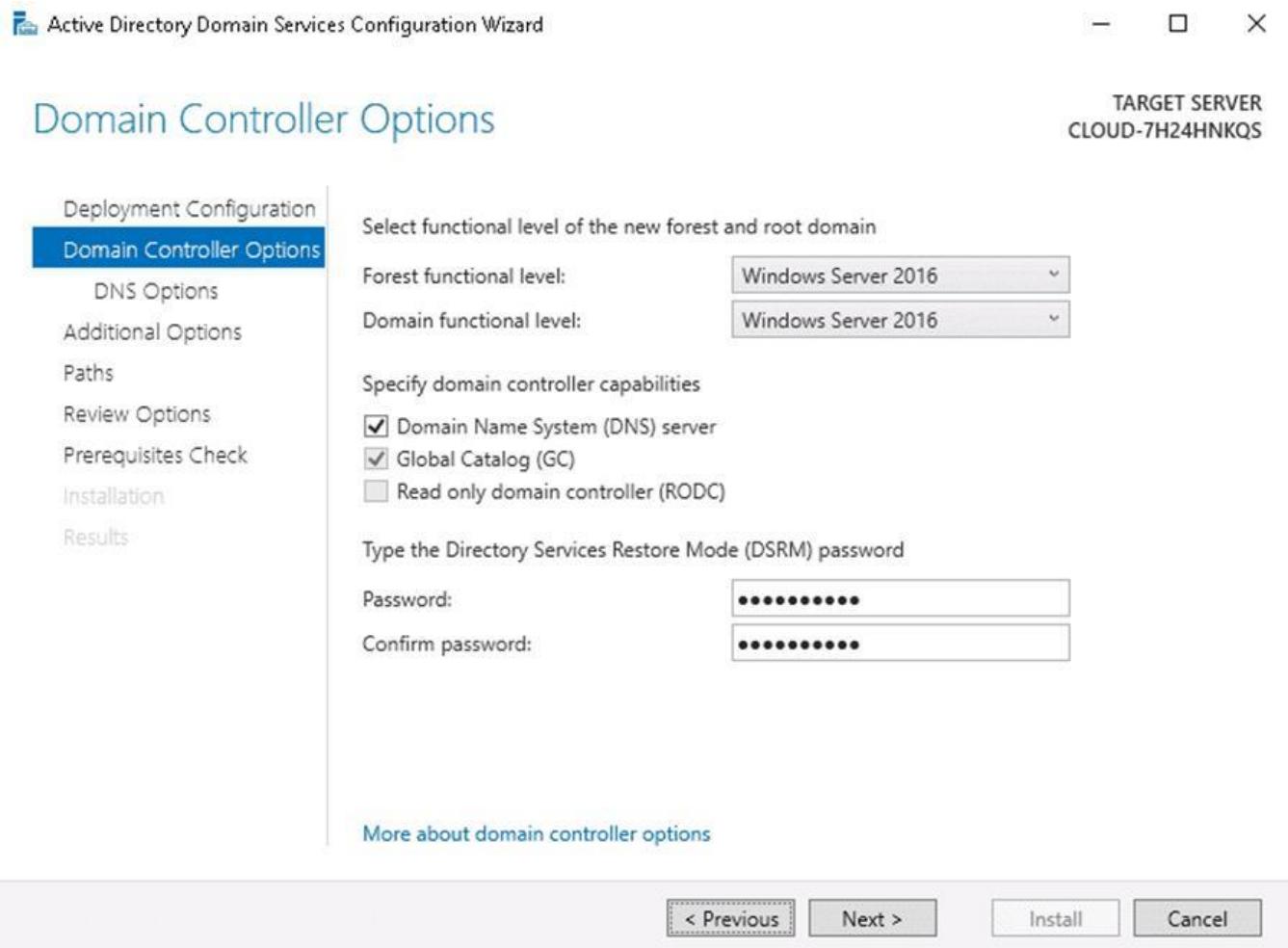
< Previous

Next >

Install

Cancel

**Step 12 – Select add a new forest, define your domain name and click on the Next button.
You should see the domain controller options page:**



Step 13 – Define your directory service restore mode password and click on the Next button. You should see the DNS options page:

DNS Options

TARGET SERVER
CLOUD-7H24HNKQS

! A delegation for this DNS server cannot be created because the authoritative parent zone cannot be found... [Show more](#) X

Deployment Configuration

Domain Controller Options

DNS Options

Additional Options

Paths

Review Options

Prerequisites Check

Installation

Results

Specify DNS delegation options

 Create DNS delegation[More about DNS delegation](#)

< Previous

Next >

Install

Cancel

Step 14 – Leave the default configuration and click on the Next button. You will be asked to set a NetBIOS name as shown below:

TARGET SERVER
CLOUD-7H24HNKQS

Additional Options

Deployment Configuration

Domain Controller Options

DNS Options

Additional Options

Paths

Review Options

Prerequisites Check

Installation

Results

Verify the NetBIOS name assigned to the domain and change it if necessary

The NetBIOS domain name:

MYDOMAIN

[More about additional options](#)

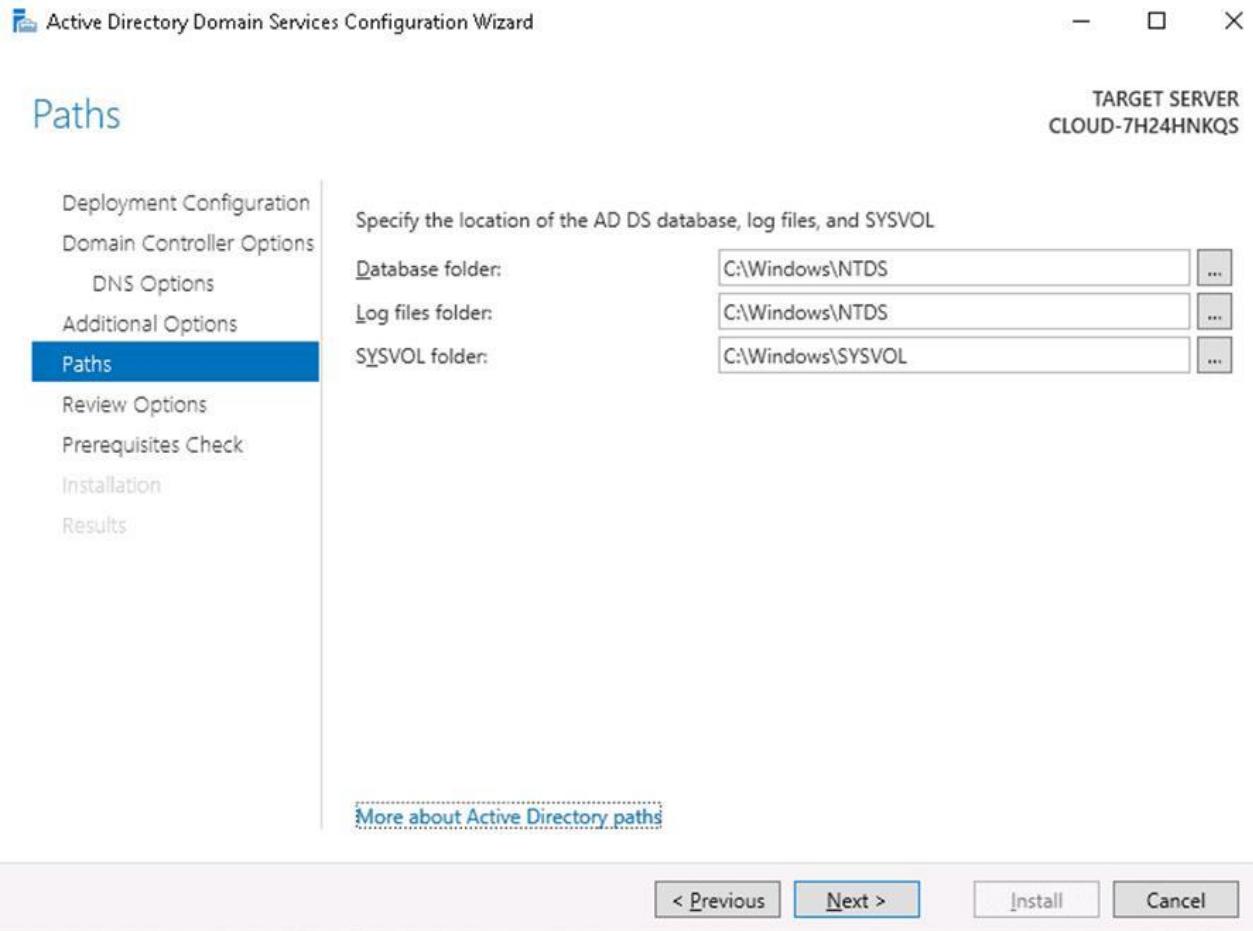
< Previous

Next >

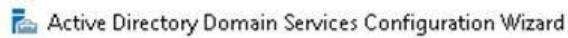
Install

Cancel

Step 15 – Set your NetBIOS name and click on the Next button. You will be asked to define AD DS database path location:



Step 16 – Leave the default path as it is and click on the Next button. You should see the review all options page:



- □ ×

Review Options

TARGET SERVER
CLOUD-7H24HNKQS

Deployment Configuration
Domain Controller Options
DNS Options
Additional Options
Paths
Review Options
Prerequisites Check
Installation
Results

Review your selections:

Configure this server as the first Active Directory domain controller in a new forest.

The new domain name is "mydomain.com". This is also the name of the new forest.

The NetBIOS name of the domain: MYDOMAIN

Forest Functional Level: Windows Server 2016

Domain Functional Level: Windows Server 2016

Additional Options:

Global catalog: Yes

DNS Server: Yes

Create DNS Delegation: No

These settings can be exported to a Windows PowerShell script to automate additional installations

[View script](#)

[More about installation options](#)

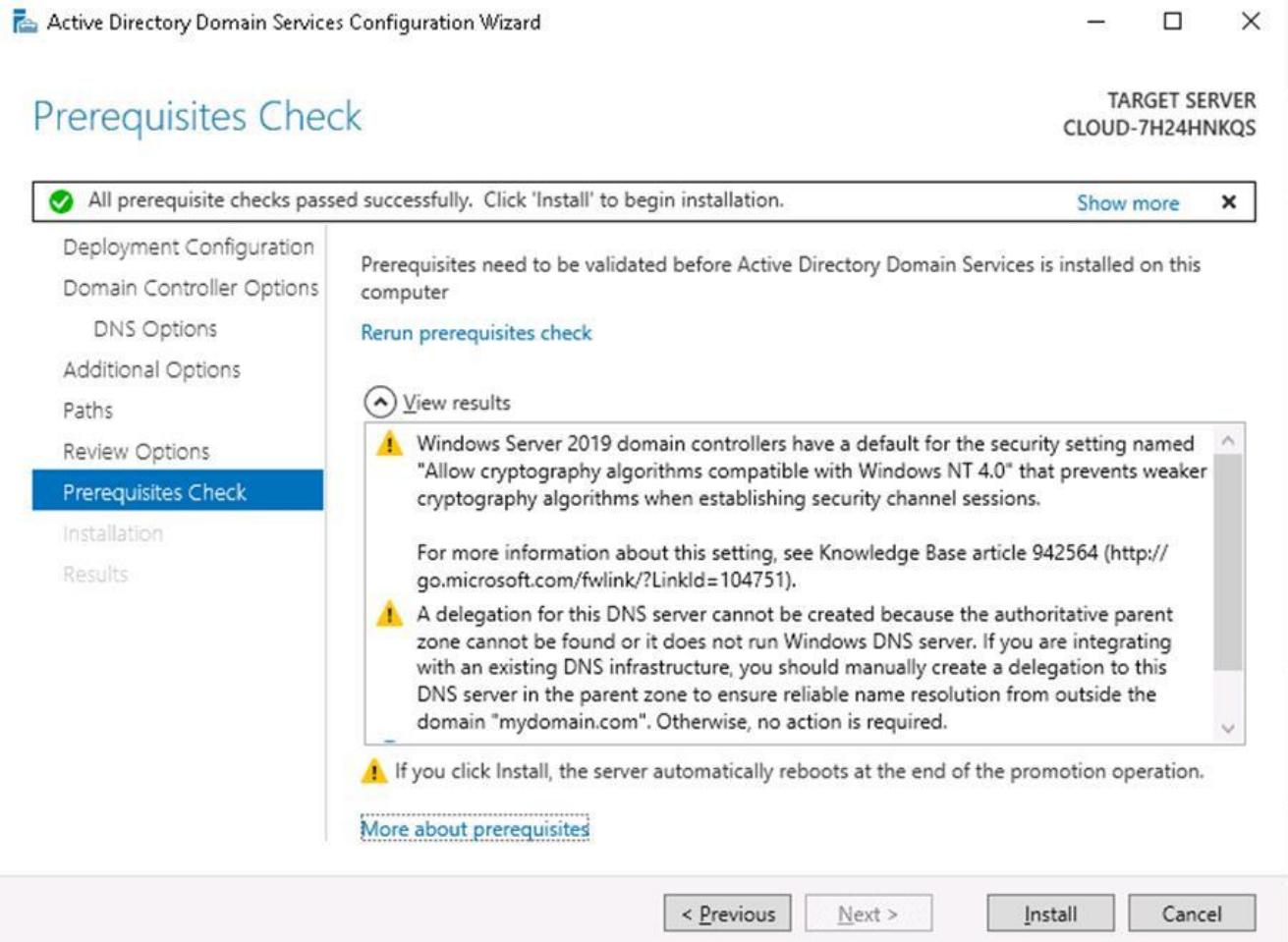
< Previous

Next >

Install

Cancel

Step 17 – Review all the configurations and click on the Next button. You should see the prerequisites check page:



Verify Domain Controller –

To confirm the successful installation of the services, run the following command on Windows PowerShell.

```
Get-Service adws,kdc,netlogon,dns
```

You should see the status of all services on the following screen:

```

Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-Service adws,kdc,netlogon,dns
status   Name           DisplayName
----   --
Running  adws          Active Directory Web Services
Running  dns            DNS Server
Running  kdc            Kerberos Key Distribution Center
Running  Netlogon       netlogon

PS C:\Users\Administrator>

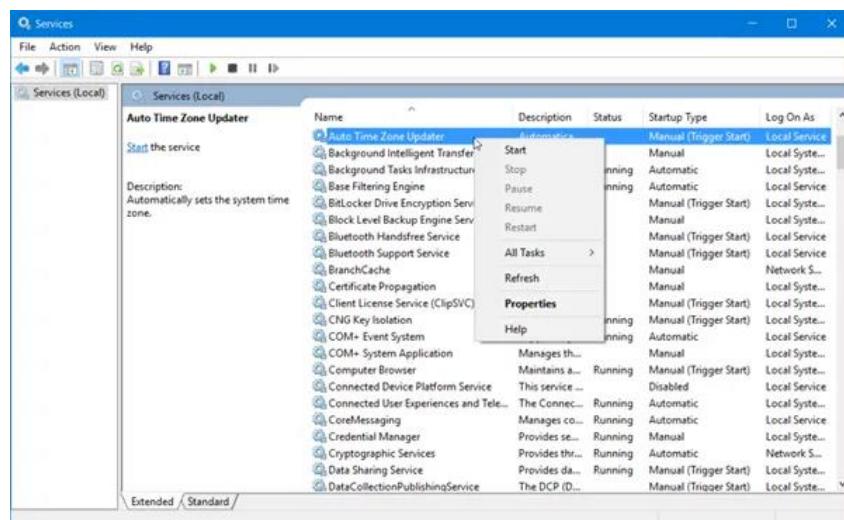
```

Get-ADDomain mydomain.com

Windows Services-

To open the Windows Services Manager on your Windows 11 or Windows 10 computer, do the following:

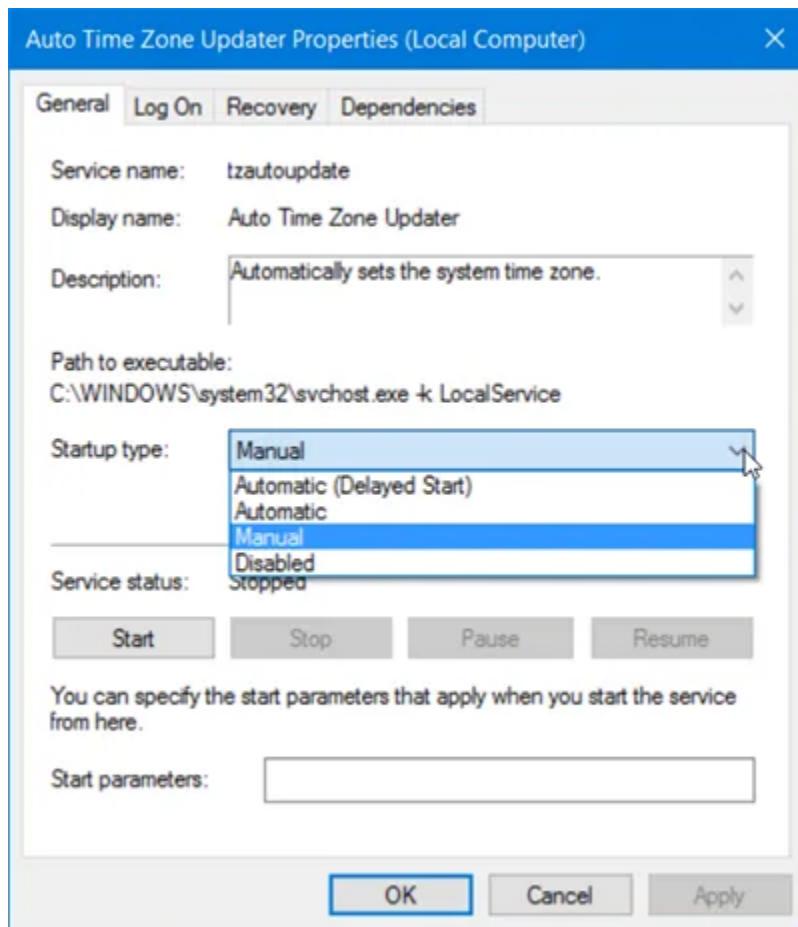
1. Right-click on the Start button to open the WinX Menu
2. Select Run
3. Type services.msc in the Run box which opens
4. Windows Services Manager will open.



Start, Stop, Disable Windows Services-

To start, stop, pause, resume or restart any Windows Service, select the Service and right-click on it. You will be offered these options.

If you wish to manage more options, double-click on the Service to open its Properties box.



Manage Windows Services using Command Line-

You can also use the Command Prompt to start, stop, pause, resume service. To use it, from the WinX Menu, open Command Prompt (Admin) and execute one of the following commands:

To start a service:

net startservice

To stop a service:

net stopservice

To pause a service:

net pauseservice

To resume a service:

net continueservice

To disable a service:

sc config "Name Of Service" start= disabled

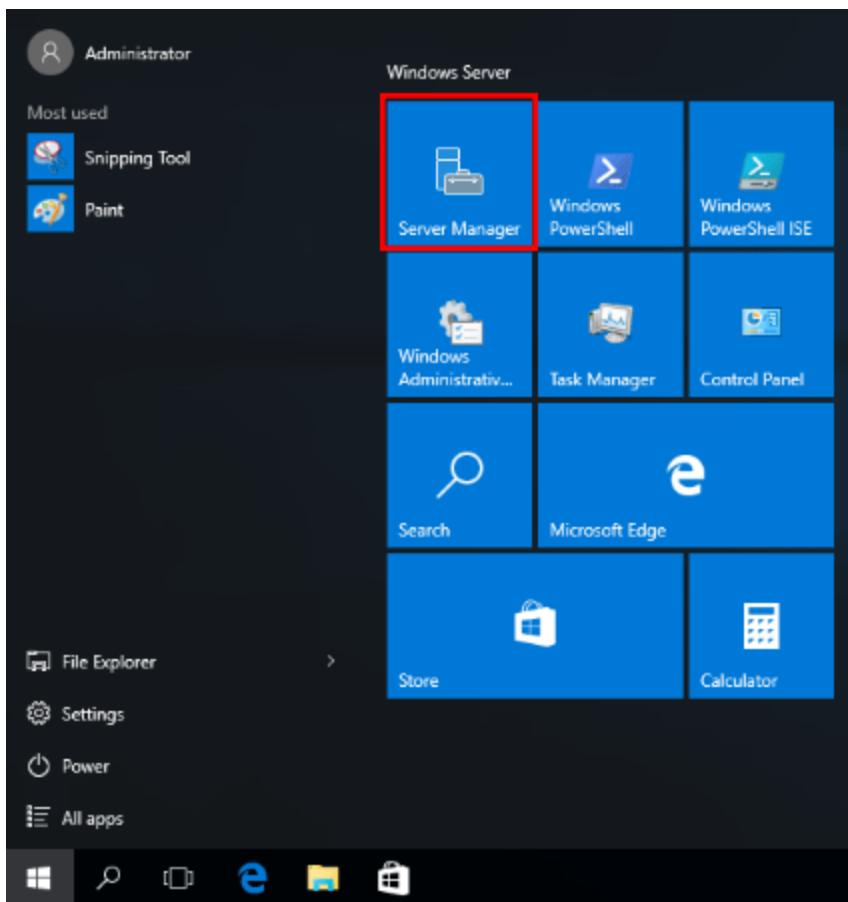
=====

IIS-

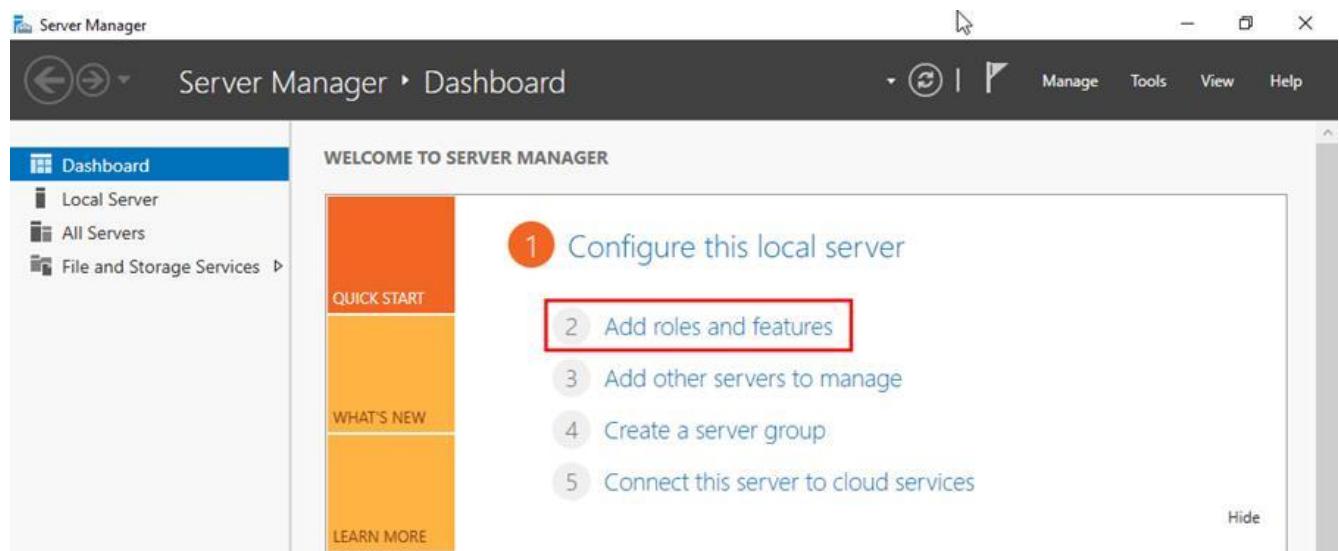
The term “IIS” stands for Internet Information Services, which is a general-purpose webserver that runs on the Windows operating system.

Install IIS-

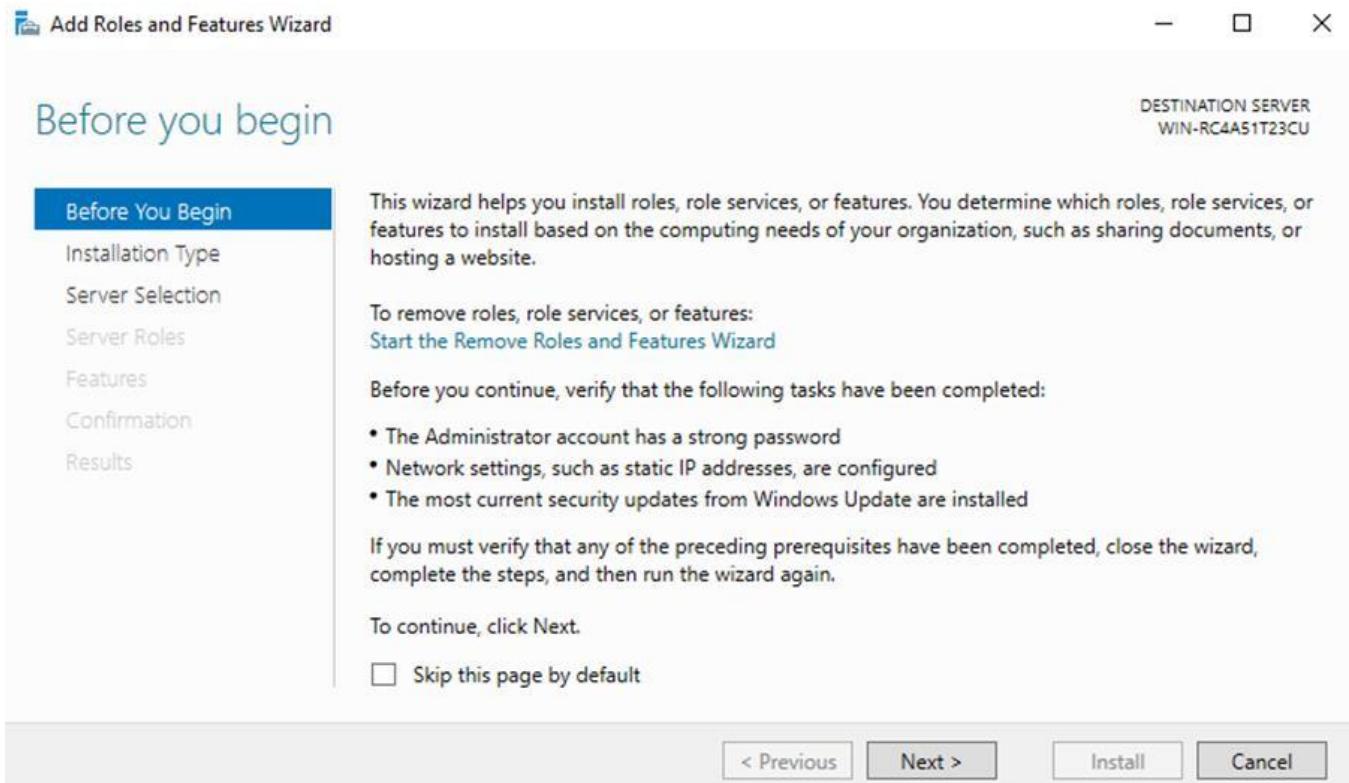
1. Open Server Manager, this can be found in the start menu. If it's not there simply type “Server Manager” with the start menu open and it should be found in the search.



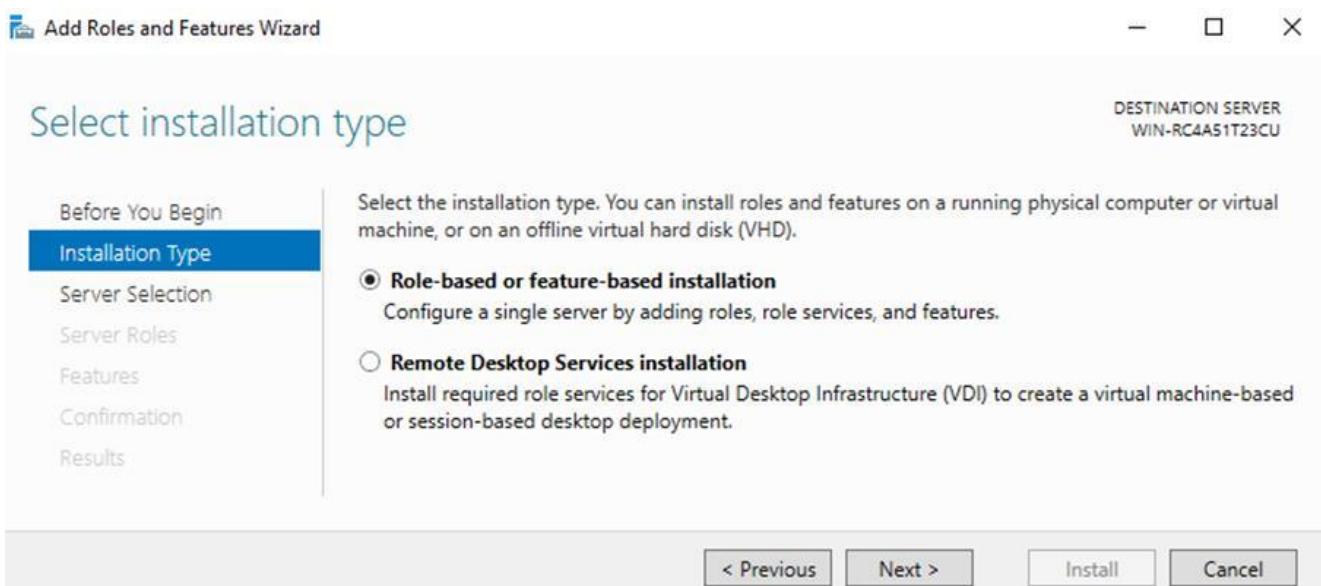
2. Click the “Add roles and features” text.



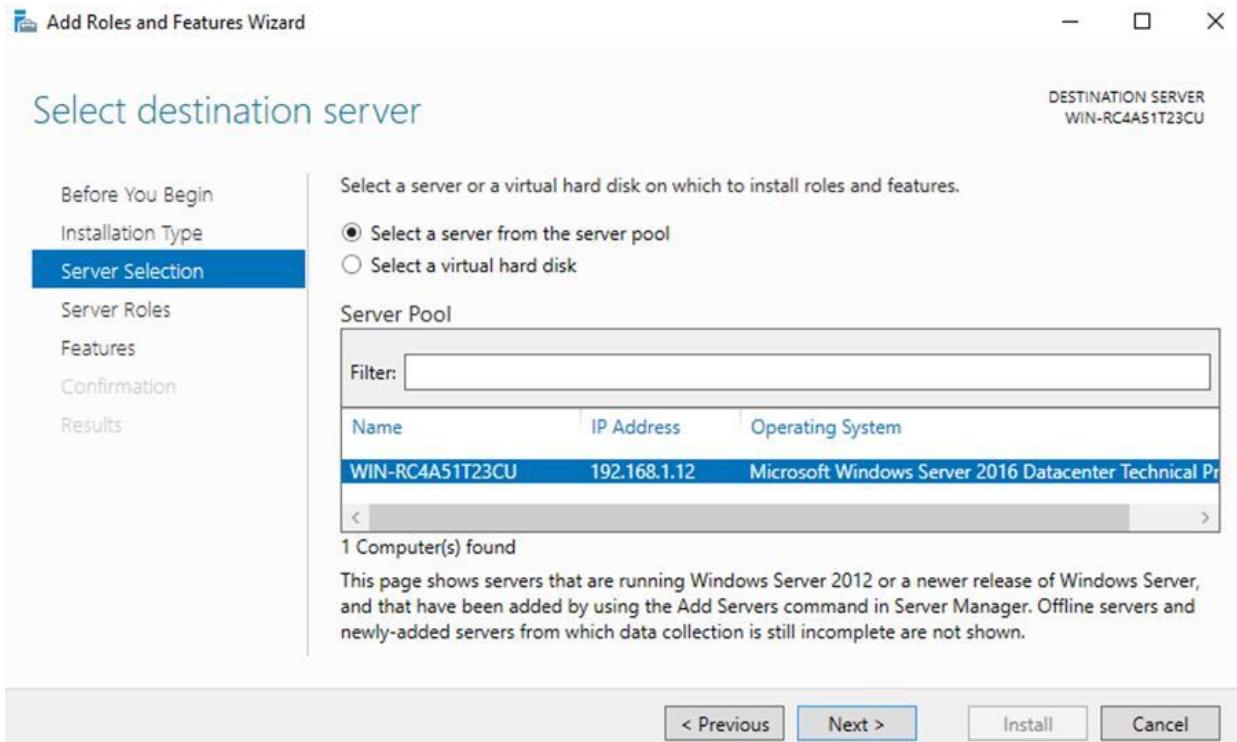
3. On the “Before you begin” window, simply click the Next button.



4. On the “Select installation type” window, leave “Role-based or feature-based installation” selected and click Next.



5. As we're installing to our local machine, leave "Select a server from the server pool" with the current machine selected and click Next. Alternatively you can select another server that you are managing from here, or a VHD.



6. From the "Select server roles" window, check the box next to "Web Server (IIS)". Doing this may open up a new window advising that additional features are required, simply click the "Add Features" button to install these as well. Click Next back on the Select server roles menu once this is complete.

Add Roles and Features Wizard

- □ ×

Select server roles

DESTINATION SERVER
WIN-RC4A51T23CU

Before You Begin

Installation Type

Server Selection

Server Roles

Features

Confirmation

Results

Select one or more roles to install on the selected server.

Roles

- Active Directory Lightweight Directory Services
- Active Directory Rights Management Services
- Device Health Attestation
- DHCP Server
- DNS Server
- Fax Server
- File and Storage Services (1 of 12 installed)
- Host Guardian Service
- Hyper-V
- MultiPoint Services
- Network Controller
- Network Policy and Access Services
- Print and Document Services
- Remote Access
- Remote Desktop Services
- Volume Activation Services
- Web Server (IIS)**
- Windows Deployment Services
- Windows Server Essentials Experience
- Windows Server Update Services

Description

Web Server (IIS) provides a reliable, manageable, and scalable Web application infrastructure.

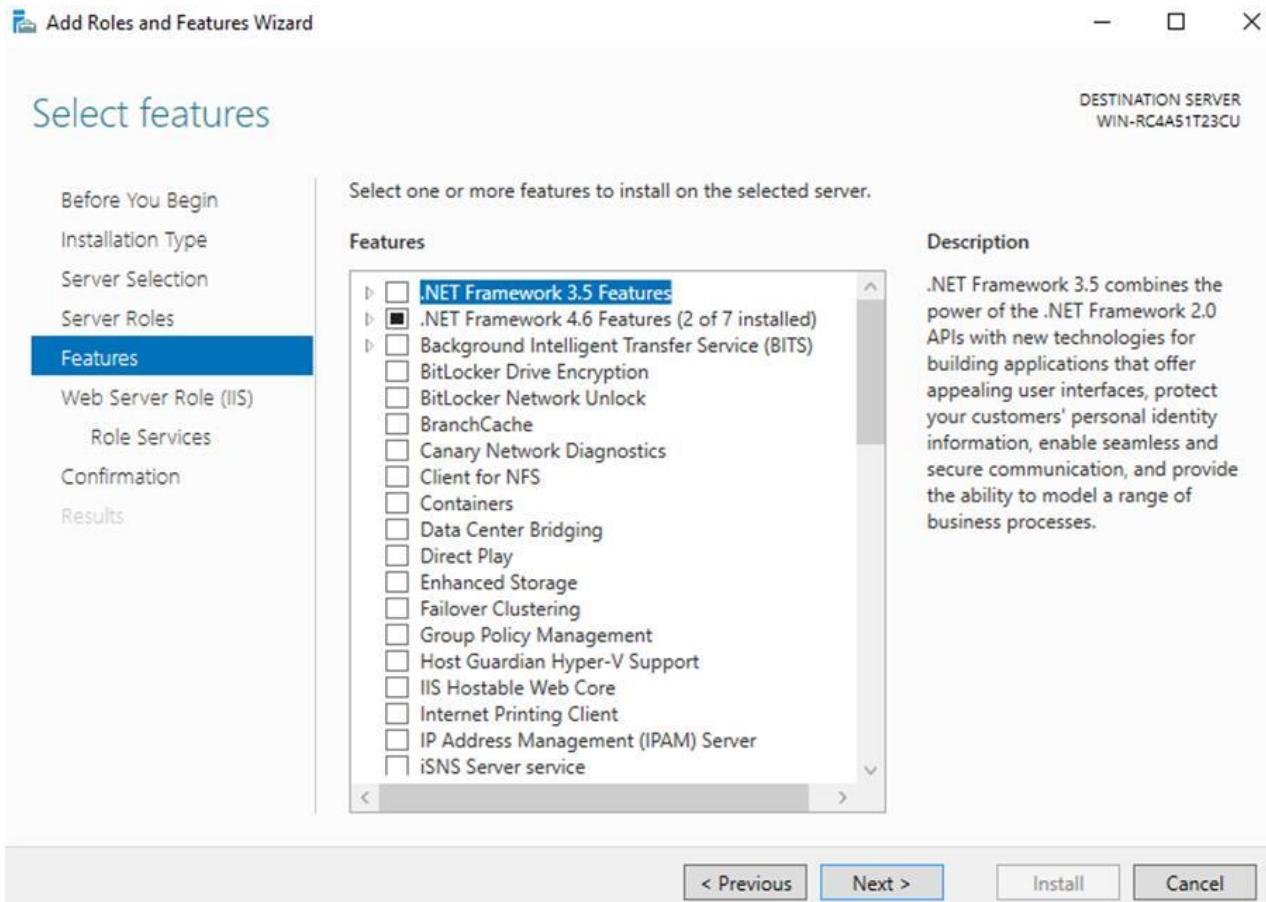
< Previous

Next >

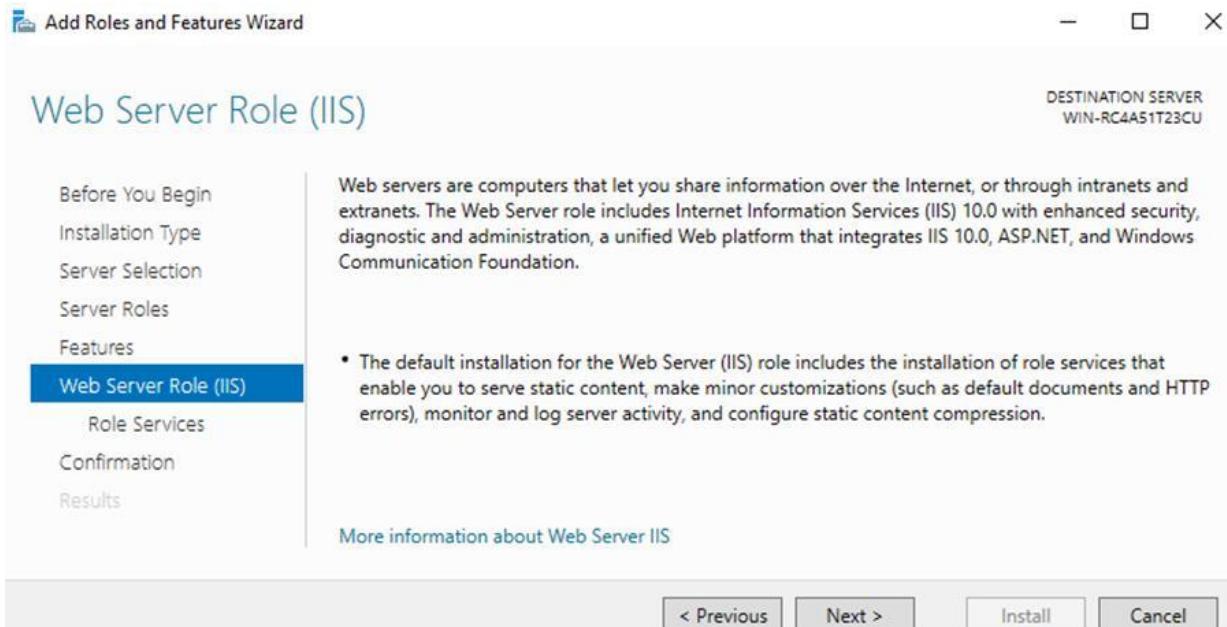
Install

Cancel

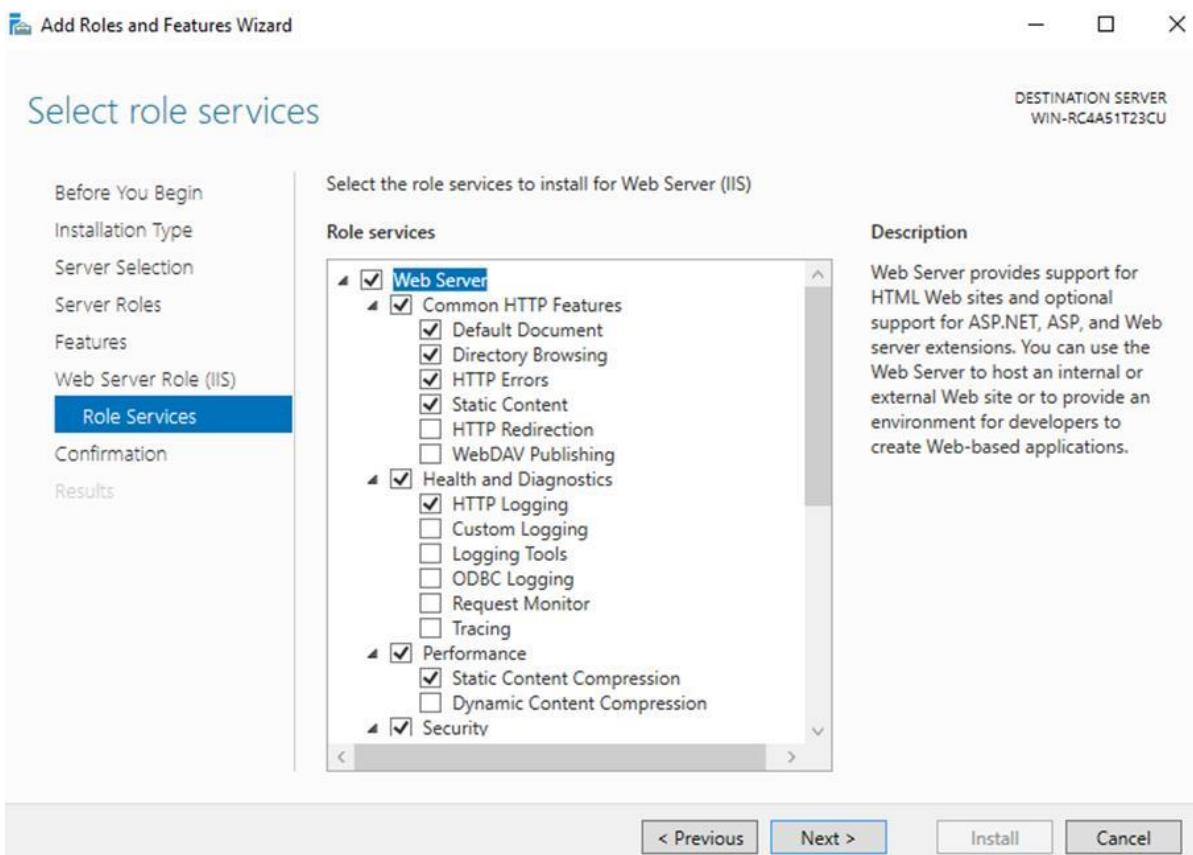
7. We will not be installing any additional features at this stage, so simply click Next on the “Select features” window.



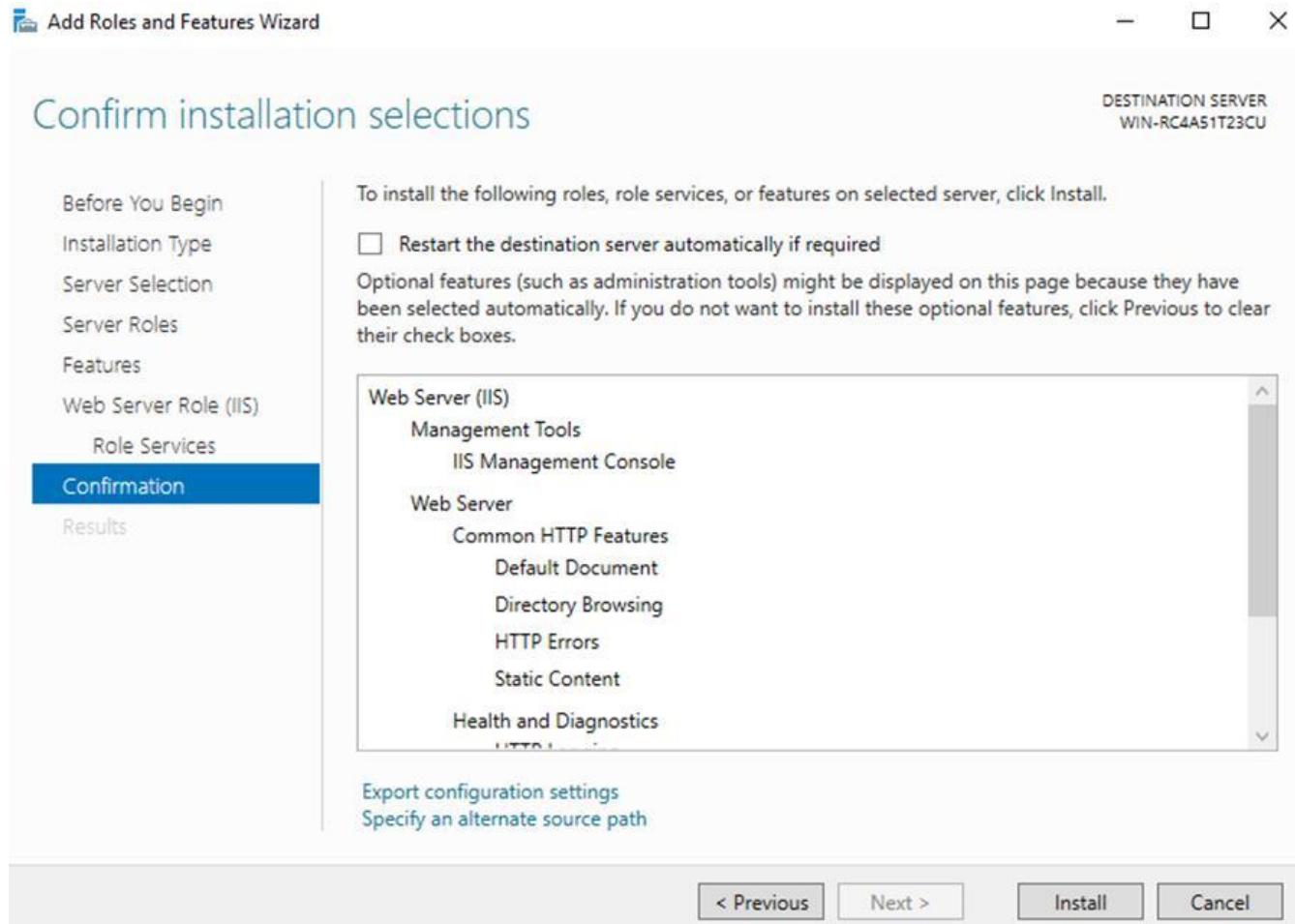
8. Click Next on the “Web Server Role (IIS)” window after reading the information provided.



9. At this point on the “Select role services” window you can install additional services for IIS if required. You don’t have to worry about this now as you can always come back and add more later, so just click Next for now to install the defaults.



10. Finally on the “Confirm installation selections” window , review the items that are to be installed and click Install when you’re ready to proceed with installing the IIS web server.



11. Once the installation has succeeded, click the close button. At this point IIS should be running on port 80 by default with the firewall rule “World Wide Web Services (HTTP Traffic-In)” enabled in Windows firewall automatically.

DESTINATION SERVER
WIN-RC4A51T23CU

Installation progress

[Before You Begin](#)[Installation Type](#)[Server Selection](#)[Server Roles](#)[Features](#)[Web Server Role \(IIS\)](#)[Role Services](#)[Confirmation](#)[Results](#)

View installation progress

i Feature installation

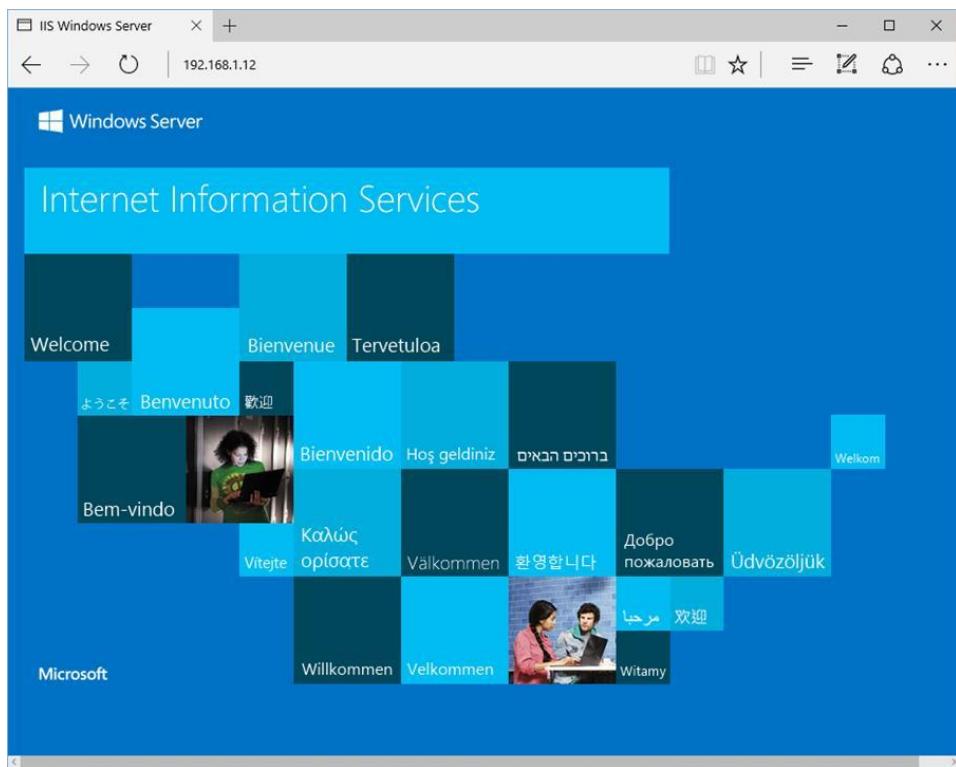
Installation succeeded on WIN-RC4A51T23CU.

Web Server (IIS)**Management Tools**[IIS Management Console](#)**Web Server****Common HTTP Features**[Default Document](#)[Directory Browsing](#)[HTTP Errors](#)[Static Content](#)**Health and Diagnostics**[HTTP Logging](#)

 You can close this wizard without interrupting running tasks. View task progress or open this page again by clicking Notifications in the command bar, and then Task Details.

[Export configuration settings](#)[< Previous](#)[Next >](#)[Close](#)[Cancel](#)

12. We can perform a simple test by opening up a web browser and browsing to the server that we have installed IIS on. You should see the default IIS page.



[Windows](#)

| Tags: [AD](#), [Domain](#), [iis](#), [services](#), [web server](#), [windows](#), [winodws services](#)

1. Linux introduction

What is Linux?

Linux is an open-source operating system like other operating systems such as Microsoft Windows, Apple Mac OS, iOS, Google android, etc.

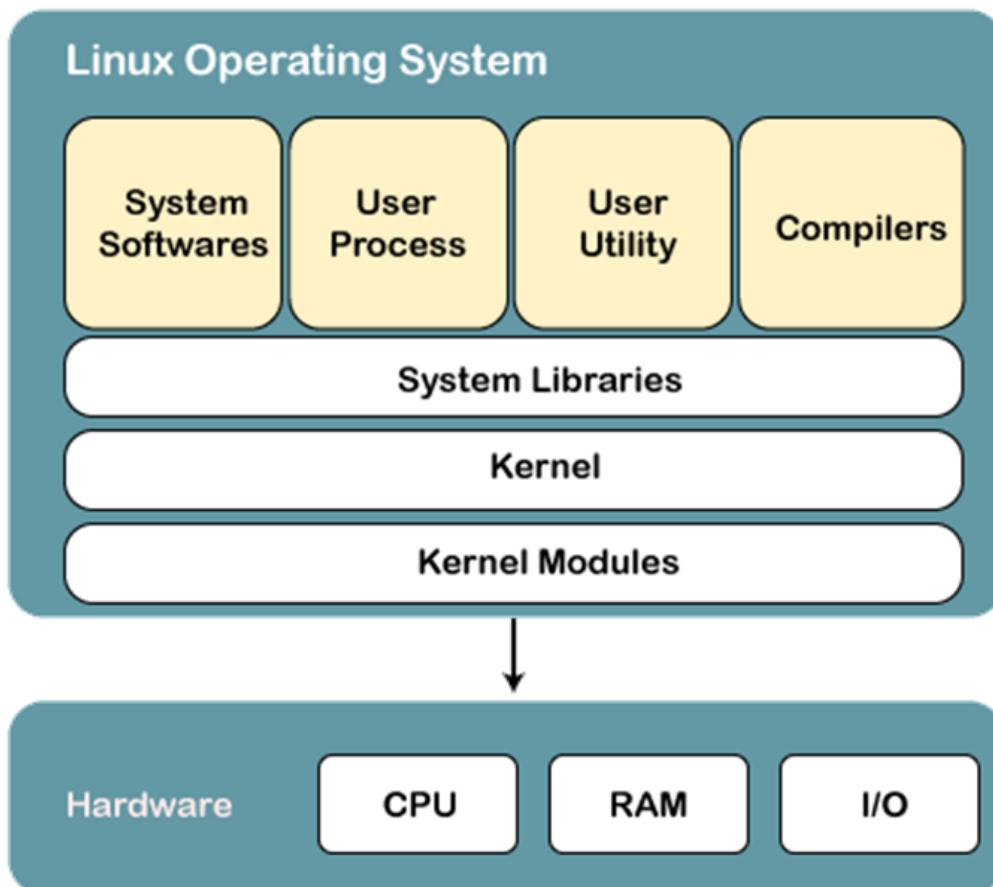
An operating system is a software that enables the communication between computer hardware and software. It conveys input to get processed by the processor and brings output to the hardware to display it

The Linux OS was developed by Linus Torvalds in 1991

Structure Of Linux Operating System-

An operating system is a collection of software, each designed for a specific function.

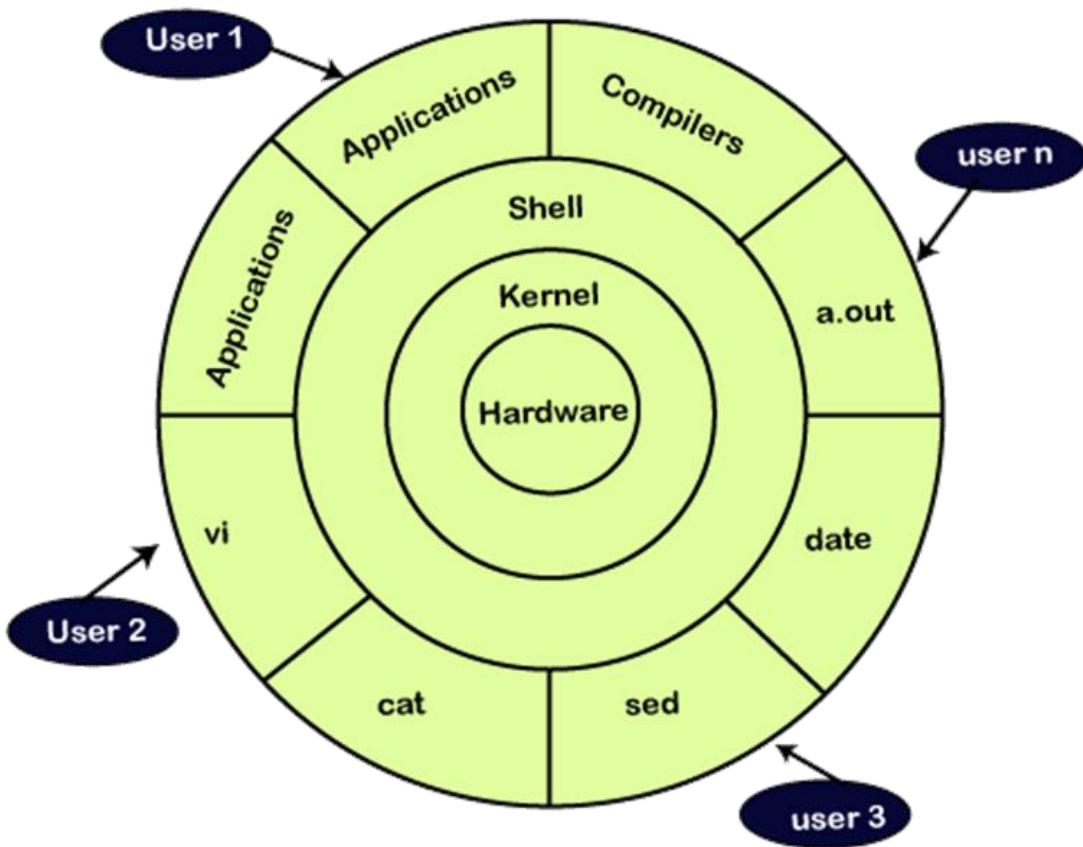
Linux OS has following components:



1) Kernel

Linux kernel is the core part of the operating system. It establishes communication between devices and software. Moreover, it manages system resources. It has four responsibilities:

1. Device management
2. Memory management
3. Process management
4. Handling system calls



Why use Linux?

Linux may be a perfect operating system if you want to get rid of viruses, malware, slowdowns, crashes, costly repairs, and many more. Further, it provides various advantages over other operating systems, and we don't have to pay for it.

In short, Linux is an operating system that is “for the people, by the people.”

Why is Linux better than other operating systems?

- Open Source
- Security
- Free
- Lightweight
- Stability
- Performance
- Flexibility
- Software Updates
- Graphical User Interface
- Suitable for programmers
- Community Support
- Privacy
- Networking
- Compatibility
- Installation
- Multiple Desktop Support
- Multitasking

Windows Vs Linux –

Sr No.	Linux	Windows
1	Free source code i.e open source	Source code is not free or open to all.
2	Secure	Not secure
3	Light weight	Not light weight
4	Linux provides full control to its users on updates.	Windows updates are annoying. The updates will come at any time and take too much time to install. Sometimes, you power on your machine, and updates are automatically getting started. Unfortunately, the user does not have much control over updates.
5	Multiuser and multitasking is very easy	Limitations to multitasking and multiuser management
6	Linux is written in assembly language and C.	Windows is written in C++ and assembly language.
7	Linux has a good support as it has a huge community of user	Windows also provide good support to its user. It provides free as

forums and online search. well as paid support. It has an easily accessible online forum.

Linux Set Environment Variable–

Some standard environment variables are as follows:

- **PATH-**

This variable contains a list of directories in which our system looks for files. It separates directories by a (:) colon.

- **USER-**

This variable holds the username.

- **HOME-**

This variable holds the default path to the user's home directory.

- **EDITOR-**

This variable contains the path to the specified editor.

- **UID**

This variable contains the path to the user's unique id.

- **TERM**

This variable contains the path to the default terminal emulator.

- **SHELL**

This variable contains the path to the default shell that is being used by the user.

- **ENV**

This variable displays all the environment variable.

How to set Environment Variable in Linux?

```
export NAME=VALUE
```

```
export new_variable=10
```

```
echo $new_variable
```

2. Linux Directory Commands

Directory Command	Description
pwd	The pwd command stands for (print working directory). It displays the current working location or directory of the user. It displays the whole working path starting with /. It is a built-in command.
ls	The ls command is used to show the list of a folder. It will list out all the files in the directed folder.
cd	The cd command stands for (change directory). It is used to change to the directory you want to work from the present directory.
mkdir	With mkdir command you can create your own directory.
rmdir	The rmdir command is used to remove a directory from your system.

Examples-

1. Pwd

```
ubuntu@ip-172-31-80-168:~$ pwd  
/home/ubuntu  
ubuntu@ip-172-31-80-168:~$ |
```

1. Ls

```
ubuntu@ip-172-31-80-168:/$ ls  
bin dev home lib32 libx32 media opt root  
boot etc lib lib64 lost+found mnt proc run  
ubuntu@ip-172-31-80-168:/$
```

ls -a list all files including hidden file starting with ‘.’

```
ubuntu@ip-172-31-80-168:/$ ls  
bin dev home lib32 libx32 media opt root  
boot etc lib lib64 lost+found mnt proc run  
ubuntu@ip-172-31-80-168:/$
```

ls -lrt -l list with long format – show permissions

-r list in reverse order

-t sort by time & date

```
ubuntu@ip-172-31-80-168:/$ ls -lrt
total 64
lrwxrwxrwx  1 root root      8 Apr 21 14:07 sbin -> usr/sbin
lrwxrwxrwx  1 root root      9 Apr 21 14:07 lib64 -> usr/lib64
lrwxrwxrwx  1 root root      9 Apr 21 14:07 lib32 -> usr/lib32
lrwxrwxrwx  1 root root      7 Apr 21 14:07 lib -> usr/lib
lrwxrwxrwx  1 root root      7 Apr 21 14:07 bin -> usr/bin
lrwxrwxrwx  1 root root     10 Apr 21 14:07 libx32 -> usr/libx32
drwxr-xr-x  2 root root   4096 Apr 21 14:07 srv
drwxr-xr-x  2 root root   4096 Apr 21 14:07 opt
drwxr-xr-x  2 root root   4096 Apr 21 14:07 mnt
drwxr-xr-x  2 root root   4096 Apr 21 14:07 media
drwxr-xr-x 14 root root   4096 Apr 21 14:07 usr
drwxr-xr-x 13 root root   4096 Apr 21 14:08 var
drwx----- 2 root root 16384 Apr 21 14:09 lost+found
drwxr-xr-x  8 root root   4096 Apr 21 14:11 snap
drwxr-xr-x  4 root root   4096 Apr 21 14:11 boot
dr-xr-xr-x 13 root root      0 Jun  7 13:14 sys
dr-xr-xr-x 158 root root     0 Jun  7 13:14 proc
drwxr-xr-x  3 root root   4096 Jun  7 13:14 home
drwxr-xr-x 16 root root  3200 Jun  7 13:14 dev
drwx-----  4 root root   4096 Jun  7 13:14 root
drwxr-xr-x  98 root root  4096 Jun  7 13:14 etc
drwxrwxrwt 12 root root  4096 Jun  7 13:15 tmp
```

cd ~ Change to home directory

```
ubuntu@ip-172-31-80-168:/$ cd ~
ubuntu@ip-172-31-80-168:~$
```

cd / Change to root directory

```
ubuntu@ip-172-31-80-168:~$ cd /
ubuntu@ip-172-31-80-168:/$
```

cd .. Change to parent directory

```
ubuntu@ip-172-31-80-168:/etc$ cd ..
ubuntu@ip-172-31-80-168:$ ..
```

cd folder Change to subdirectory var

```
root@ip-172-31-83-229:~# pwd
/root
root@ip-172-31-83-229:~# cd /var/log/
root@ip-172-31-83-229:~/var/log# ..
```

```
root@ip-172-31-83-229:~# pwd
/root
root@ip-172-31-83-229:~# cd /var/log/
root@ip-172-31-83-229:~/var/log# ..
```

mkdir – Stands for ‘make directory’. With the help of mkdir command, you can create a new directory wherever you want in your system.

rmdir remove files

3. Linux File Commands

Linux Files

In Linux system, everything is a file and if it is not a file, it is a process. A file doesn't include only text files, images and compiled programs but also include partitions, hardware device drivers and directories. Linux consider everything as as file.

Files are always case sensitive.

Linux File Commands

Command Description

File Determines file type.

Touch Used to create a file.

Rm To remove a file.

Cp To copy a file.

Mv To rename or to move a file.

Rename To rename file.

When we install the Linux operating system, Linux offers many file systems such as **Ext**, **Ext2**, **Ext3**, **Ext4**, **JFS**, **ReiserFS**, **XFS**, **btrfs**, and **swap**.

The file system Ext stands for **Extended File System**.

Examples-

1. `touch file1`

```
root@ip-172-31-22-230:~# ls
snap
root@ip-172-31-22-230:~# touch file1
root@ip-172-31-22-230:~# ls
file1  snap
root@ip-172-31-22-230:~# .....
```

1. `rm file1`

```
root@ip-172-31-22-230:~# ls
snap
root@ip-172-31-22-230:~# touch file1
root@ip-172-31-22-230:~# ls
file1 snap
root@ip-172-31-22-230:~# rm file1
root@ip-172-31-22-230:~# ls
snap
root@ip-172-31-22-230:~#
```

1. cp <existing file name> <new file name>

```
root@ip-172-31-22-230:~# ls
file1 snap
root@ip-172-31-22-230:~# cp file1 file21
root@ip-172-31-22-230:~# ls
file1 file21 snap
root@ip-172-31-22-230:~#
```

1. mv <existing file name> <new file name>

```
root@ip-172-31-22-230:~# ls
file1 file21 snap
root@ip-172-31-22-230:~# mv file1 file9
root@ip-172-31-22-230:~# ls
file21 file9 snap
root@ip-172-31-22-230:~#
```

=====

=

Linux File Ownership–

Every Linux system have three types of owner:

1. **User:** A user is the one who created the file. By default, whosoever, creates the file becomes the owner of the file. A user can create, delete, or modify the file.
2. **Group:** A group can contain multiple users. All the users belonging to a group have same access permission for a file.
3. **Other:** Any one who has access to the file other than **user** and **group** comes in the category of **other**. Other has neither created the file nor is a group member.

File Permissions-

All the three owners (user owner, group, others) in the Linux system have three types of permissions defined. Nine characters denotes the three types of permissions.

1. **Read (r) :** The read permission allows you to open and read the content of a file. But you can't do any editing or modification in the file.
2. **Write (w) :** The write permission allows you to edit, remove or rename a file. For instance, if a file is present in a directory, and write permission is set on the file but not on the directory, then you can edit the content of the file but can't remove, or rename it.
3. **Execute (x):** In Unix type system, you can't run or execute a program unless execute permission is set.But in Windows, there is no such permission available.

Permissions are listed below-

permission on a file	on a directory
r (read)	read file content (cat) read directory content (ls)
w (write)	change file content (vi) create file in directory (touch)
x (execute)	execute the file enter the directory (cd)

Lets see output of below cmd-

ls -l file_name

Output

```
-rw-r--r-- 12 linuxize users 12.0K Apr 28 10:10 file_name
|[-][-][-]- [-----] [---]
| | | | | | | |
| | | | | | +-----> 7. Group
| | | | | +-----> 6. Owner
| | | | +-----> 5. Alternate Access Method
| | | +-----> 4. Others Permissions
| | +-----> 3. Group Permissions
| +-----> 2. Owner Permissions
+-----> 1. File Type
```

Changing File permissions-

The File permissions can be changed using the chmod command. Only root, the file owner, or user with sudo privileges can change the permissions of a file. Be extra careful when using chmod, especially when recursively changing the permissions. The command can accept one or more files and/or directories separated by space as arguments.

Numeric Method-

- r (read) = 4
- w (write) = 2
- x (execute) = 1
- no permissions = 0

The permissions number of a specific user class is represented by the sum of the values of the permissions for that group.

To find out the file's permissions in numeric mode, simply calculate the totals for all users' classes. For example, to give read, write and execute permission to the file's owner, read and execute permissions to the file's group and only read permissions to all other users, you would do the following:

- Owner: $rwx=4+2+1=7$
- Group: $r-x=4+0+1=5$
- Others: $r-x=4+0+1=5$

Ex. `chmod 644 dirname`

Or `chmod 755 filename`

```
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rw-r--r-- 1 root root   42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~# chmod 755 ygminds
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rwxr-xr-x 1 root root   42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~#
```

chown owner_name file_name

chown Sana ygminds

```
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rwxr-xr-x 1 root root   42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~# chown Sana ygminds
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rwxr-xr-x 1 Sana root   42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~#
```

Linux File Links –

A Linux filesystem has many hard links and symbolic links. A link is a connectivity between the filename and the actual data byte in the disk space. More than one filename can **link** to the same data.

There are two types of links in Linux OS:

1. Hard Links
2. Soft Links

1) Hard Links

They are the low-level links. It links more than one filename with the same Inode and it represents the physical location of a file.

When hard link is created for a file, it directly points to the Inode of the original file in the disk space, which means no new Inode is created. Directories are not created using hard links and they can not cross filesystem boundaries. When the source file is removed or moved, then hard links are not affected.

2) Soft Links (Symbolic Links)

Soft links are very common. It represents a virtual or abstract location of the file. It is just like the shortcuts created in Windows. A soft link doesn't contain any information or content of the linked file, instead it has a pointer to the location of the linked file. In other words, a new file is created with new Inode, having a pointer to the Inode location of the original file.

It is used to create link between directories and can cross filesystem boundaries. When the source file is removed or moved, then soft links are not updated.

Linux Inodes –

An Inode number is a uniquely existing number for all the files in Linux and all Unix type systems.

When a file is created on a system, a file name and Inode number is assigned to it.

Generally, to access a file, a user uses the file name but internally file name is first mapped with respective Inode number stored in a table.

Note: Inode doesn't contain the file name. Reason for this is to maintain hard-links for the files. When all the other information is separated from the file name then only we can have various file names pointing to the same Inode.

Inode Contents –

An Inode is a data structure containing metadata about the files.

Following contents are stored in the Inode from a file:

- User ID of file

- Group ID of file
- Device ID
- File size
- Date of creation
- Permission
- Owner of the file
- File protection flag
- Link counter to determine number of hard links

Inode Table-

The Inode table contains all the Inodes and is created when file system is created. The **df -i** command can be used to check how many inodes are free and left unused in the filesystem.

```
root@ip-172-31-8-13:~# df -i
Filesystem      INodes   IUsed   IFree  IUse% Mounted on
/dev/root        1032192 102448  929744   10% /
tmpfs            123891      2  123889    1% /dev/shm
tmpfs            819200     560  818640    1% /run
tmpfs            123891      3  123888    1% /run/lock
/dev/xvda15        0        0       0      - /boot/efi
tmpfs            24778      25  24753    1% /run/user/1000
root@ip-172-31-8-13:~# |
```

Hard Links –

Creating Hard Links

Hard links for any file can be created with command **ln**. One extra hard link file will be created in the respective directory.

```
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rw xr-xr-x 1 Sana root    42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~# touch xyz
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rw xr-xr-x 1 Sana root    42 Jul  7 01:59 ygminds
-rw r--r-- 1 root root     0 Jul  7 13:51 xyz
root@ip-172-31-8-13:~# ln xyz hardlink_to_xyz
root@ip-172-31-8-13:~# ls -lrt
total 8
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rw xr-xr-x 1 Sana root    42 Jul  7 01:59 ygminds
-rw r--r-- 2 root root     0 Jul  7 13:51 xyz
-rw r--r-- 2 root root     0 Jul  7 13:51 hardlink_to_xyz
root@ip-172-31-8-13:~# |
```

Look at the above snapshot, we have created a hard link for the file **xyz** in the directory **new1**.

The original file and hard linked file both contain the same Inode number and hence, they have the same permissions and same owners. Content will also be the same for both the files. In short, both the files are equal now, but if original file will be removed then hard link file will not be affected.

Symbolic Links-

Symbolic links are also called **soft links**. Command **ln -s** is used to create soft link. It doesn't link to Inodes but create a name to mapping. It create its own Inode number.

ln -s xyz symlink_to_xyz

Look at the above snapshot, we have created a symbolic link for file **xyz** with command "**ln -s xyz symlink_to_xyz**". Symbolic link Inode is different from the original file Inode number. Target permissions are applied on the symlink file. Hard links are limited to their own partition, but symbolic links can be linked anywhere.

```

root@ip-172-31-8-13:~# ls -lrt
total 16
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rwxr-xr-x 1 Sana root   42 Jul  7 01:59 ygminds
-rw-r--r-- 2 root root   0 Jul  7 13:51 xyz
-rw-r--r-- 2 root root   0 Jul  7 13:51 hardlink_to_xyz
-rw-r--r-- 1 root root  32 Jul  7 17:51 file1.txt
-rw-r--r-- 1 root root  27 Jul  7 17:51 file2.txt
root@ip-172-31-8-13:~# ln -s xyz symlink_to_xyz
root@ip-172-31-8-13:~# ls -lrt
total 16
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rwxr-xr-x 1 Sana root   42 Jul  7 01:59 ygminds
-rw-r--r-- 2 root root   0 Jul  7 13:51 xyz
-rw-r--r-- 2 root root   0 Jul  7 13:51 hardlink_to_xyz
-rw-r--r-- 1 root root  32 Jul  7 17:51 file1.txt
-rw-r--r-- 1 root root  27 Jul  7 17:51 file2.txt
lrwxrwxrwx 1 root root   3 Jul  7 17:59 symlink_to_xyz -> xyz
root@ip-172-31-8-13:~#

```

Removing Links-

With **rm** command links can be removed.

```

root@ip-172-31-8-13:~# ls -li
total 16
16369 -rw-r--r-- 1 root root  32 Jul  7 17:51 file1.txt
16359 -rw-r--r-- 1 root root  27 Jul  7 17:51 file2.txt
1892 -rw-r--r-- 2 root root   0 Jul  7 13:51 hardlink_to_xyz
258199 drwx----- 4 root root 4096 Jul  5 02:12 snap
1044 lrwxrwxrwx 1 root root   3 Jul  7 17:59 symlink_to_xyz -> xyz
1892 -rw-r--r-- 2 root root   0 Jul  7 13:51 xyz
67676 -rwxr-xr-x 1 Sana root   42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~# rm symlink_to_xyz
root@ip-172-31-8-13:~# rm hardlink_to_xyz
root@ip-172-31-8-13:~# ls -li
total 16
16369 -rw-r--r-- 1 root root  32 Jul  7 17:51 file1.txt
16359 -rw-r--r-- 1 root root  27 Jul  7 17:51 file2.txt
258199 drwx----- 4 root root 4096 Jul  5 02:12 snap
1892 -rw-r--r-- 1 root root   0 Jul  7 13:51 xyz
67676 -rwxr-xr-x 1 Sana root   42 Jul  7 01:59 ygminds
root@ip-172-31-8-13:~#

```

Look at the above snapshot, directory **link** contains both hard link and soft link. With the command **rm** we have removed both the links.

Tar command

tar command in Linux with examples –

The Linux ‘tar’ stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important command which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Cmd-

tar [options] [archive-file] [file or directory to be archived]

Options:

-c : Creates Archive

-x : Extract the archive

-f : creates archive with given filename

-t : displays or lists files in archived file

-u : archives and adds to an existing archive file

-v : Displays Verbose Information

-A : Concatenates the archive files

-z : zip, tells tar command that creates tar file using gzip

-j : filter archive tar file using tbzip

-W : Verify a archive file

-r : update or add file or directory in already existed .tar file

What is an Archive file?

An Archive file is a file that is composed of one or more files along with metadata. Archive files are used to collect multiple data files together into a single file for easier portability and storage, or simply to compress files to use less storage space.

Ex.

```
tar -czvf file1.tar.gz file1.txt
```

```
root@ip-172-31-8-13:~# ls -lrt
total 12
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rw-r--r-- 1 root root   32 Jul  7 17:51 file1.txt
-rw-r--r-- 1 root root   27 Jul  7 17:51 file2.txt
root@ip-172-31-8-13:~# tar -czvf file1.tar.gz file1.txt
file1.txt
root@ip-172-31-8-13:~# ls -lrt
total 16
drwx----- 4 root root 4096 Jul  5 02:12 snap
-rw-r--r-- 1 root root   32 Jul  7 17:51 file1.txt
-rw-r--r-- 1 root root   27 Jul  7 17:51 file2.txt
-rw-r--r-- 1 root root  153 Jul  8 11:41 file1.tar.gz
root@ip-172-31-8-13:~#
```

Compress directory using the tar command:

```
tar -czvf ygminds.tar.gz youngminds/
```

```
root@ip-172-31-8-13:/lost+found# mkdir youngminds
root@ip-172-31-8-13:/lost+found# ls -lrt
total 4
drwxr-xr-x 2 root root 4096 Jul  8 11:56 youngminds
root@ip-172-31-8-13:/lost+found# tar -czvf ygminds.tar.gz youngminds/
youngminds/
root@ip-172-31-8-13:/lost+found# ls -lrt
total 8
drwxr-xr-x 2 root root 4096 Jul  8 11:56 youngminds
-rw-r--r-- 1 root root  117 Jul  8 11:57 ygminds.tar.gz
root@ip-172-31-8-13:/lost+found#
```

```
tar -czvf ygminds.tar.gz youngminds/
```

Show the archive content:

```
tar -tf archive.tar
```

4. Linux File Contents Command

There are many commands which help to look at the contents of a file. Now we'll look at some of the commands like head, tac, cat, less & more and strings.

Commands Function

head	It displays the beginning of a file.
tail	It displays the last part of a file.
cat	This command is versatile and multi worker.
tac	Opposite of cat.
more	Command line displays contents in pager form that is either in more format.
less	Command line displays contents in pager form that is either in less format.

1. To display the file content, execute the cat command as follows:

```
cat <file name>
```

```
root@ip-172-31-22-230:~# ls
file1 snap
root@ip-172-31-22-230:~# cat file1
welcome to the Young Minds.
root@ip-172-31-22-230:~# |
```

1. Let's create a file to understand how to open a file. Execute the below command:

```
cat > Test.txt
```

This is a Test file.

```
root@ip-172-31-22-230:~# cat >test.txt
Welcome to the Young Minds
^C
root@ip-172-31-22-230:~# ls
snap test.txt
root@ip-172-31-22-230:~# cat test.txt
Welcome to the Young Minds
root@ip-172-31-22-230:~# |
```

1. To display the file content by the more command, execute it as follows:

```
more test.txt
```

```
root@ip-172-31-22-230:~# ls
snap test.txt
root@ip-172-31-22-230:~# more test.txt
Welcome to the Young Minds
root@ip-172-31-22-230:~# |
```

1. We can display the file content by using the head command, but it is slightly different than others. It displays the first part of files via standard input. By default, it displays the first ten lines of the files. It starts reading the file from the head (first line).

```
head <file name>
```

```
head -15 <file name>
```

```
root@ip-172-31-22-230:/var/log# head auth.log
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: new group: name=ubuntu, GID=1000
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: new user: name=ubuntu, UID=1000,
GID=1000, home=/home/ubuntu, shell=/bin/bash, from=none
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'adm'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'dialout'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'cdrom'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'floppy'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'sudo'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'audio'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'dip'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'video'
root@ip-172-31-22-230:/var/log# |
```

1. The tail command is similar to the head command. The difference between both commands is that it starts reading the file from the tail (last line). Similar to head command, it also displays the output of the last ten lines by default.

```
tail <file name>
```

```
tail -15 <file name>
```

```
root@ip-172-31-22-230:/var/log# head auth.log
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: new group: name=ubuntu, GID=1000
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: new user: name=ubuntu, UID=1000,
GID=1000, home=/home/ubuntu, shell=/bin/bash, from=none
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'adm'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'dialout'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'cdrom'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'floppy'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'sudo'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'audio'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'dip'
Jun 24 07:20:16 ip-172-31-22-230 useradd[430]: add 'ubuntu' to group 'video'
root@ip-172-31-22-230:/var/log# |
```

Linux Edit file

Linux file system allows us to operate various operations on files like create, edit, rename, remove. We can edit files by different Linux editors like vim, nano, Emacs, Gedit, Gvim, and more.

How to edit files in Linux

Let's understand how to edit files on a Linux server over different text editors.

Edit files with VI editor

The VI editor is the most widely used text editor in Linux based systems. The Vi editor has various modes like **normal mode**, **insert mode**, **command mode**, **line mode**, and **more**. Each mode allows us to operate its specific operations.

Before editing files, let's understand how to switch a mode in Vi editor:

- Press the **ESC key** for **normal mode**.
- Press **i Key** for **insert mode**.
- Press **:q!** keys to exit from the editor without saving a file.
- Press **:wq!** Keys to save the updated file and exit from the editor.

- Press :w **test.txt** to save the file as test.txt

5. Linux Filesystem Hierarchy Standard (FHS)

Filesystem hierarchy standard describes directory structure and its content in Unix and Unix like operating system. It explains where files and directories should be located and what it should contain.

The Root Directory

All the directories in the Linux system comes under the root directory which is represented by a **forward** slash (/). Everything in your system can be found under this root directory even if they are stored in different virtual or physical devices.

```
root@ip-172-31-4-17:~# ls /
bin  dev  home  lib32  libx32      media   opt   root  sbin  srv  tmp  var
boot  etc  lib  lib64  lost+found  mnt    proc  run  snap  sys  usr
root@ip-172-31-4-17:~# |
```

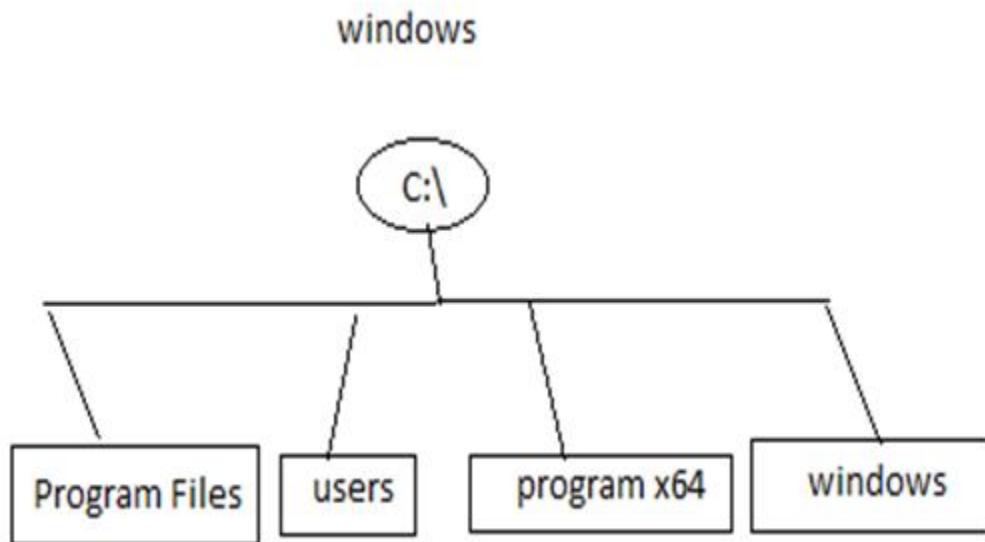
Linux Directories

We have categorize the directories according to the type of file as given below:

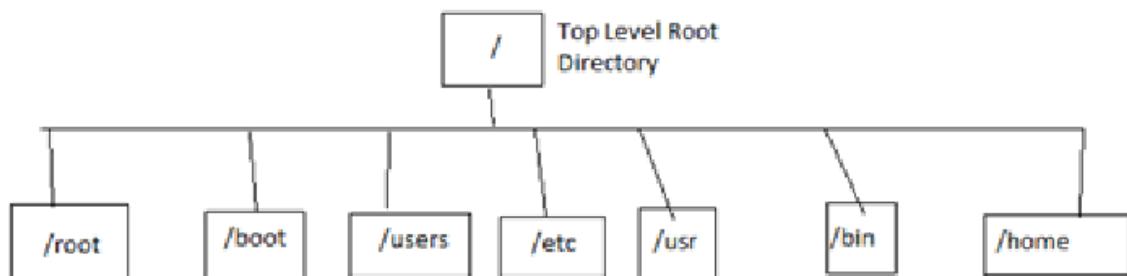
Directory type	Types of files stored
Binary directories	Contains binary or compiled source code files, eg, /bin, /sbin, etc.
Configuration directories	Contains configuration files of the system, eg, /etc, /boot.
Data directories	Stores data files, eg, /home, /root, etc.
Memory directories	Stores device files which doesn't take up actual hard disk space, eg, /dev, /proc, /sys.
Unix System Resources	Contains sharable, read only data, eg, /usr/bin, /usr/lib, etc.

Variable directories	Contains larger size data, eg, /var/log, /var/cache, etc.
Non-standard directories	Directories which do not come under standard FHS, eg, lost+found, /run, etc.

WINDOWS –



Linux –



6. Linux Filters

Linux cut Command-

```
cut OPTION... [FILE]...
```

-b, --bytes=LIST: It is used to cut a specific section by bytes.

-c, --characters=LIST: It is used to select the specified characters.

-d, --delimiter=DELIM: It is used to cut a specific section by a delimiter.

-f, --fields=LIST: It is used to select the specific fields. It also prints any line that does not contain any delimiter character, unless the -s option is specified.

--output-delimiter=STRING: This option is specified to use a STRING as an output delimiter; The default is to use “input delimiter”.

-z, --zero-terminated: It is used if line delimiter is NUL, not newline.

Using Space As Delimiter

```
cut -d ' ' -f2 ygminds
```

```
root@ip-172-31-4-17:/# cat ygminds
welcome to ygminds
we have to class at 7.30
Good bye
root@ip-172-31-4-17:/# cut -d ' ' -f2 ygminds
to
have
bye
root@ip-172-31-4-17:/# |
```

Cut by byte

```
cut -b <byte number> <file name>
```

```
cut -b 2 ygminds
```

```
root@ip-172-31-4-17:/# cat ygminds
welcome to ygminds
we have to class at 7.30
Good bye
root@ip-172-31-4-17:/# cut -b 2 ygminds
e
e
o
root@ip-172-31-4-17:/#
```

Cut by Character

```
cut -c < characters > <file name>
```

```
cut -c 1,6 ygminds
```

```
cut -c 1-3 ygminds
```

```
root@ip-172-31-4-17:/# cat ygminds
welcome to ygminds
we have to class at 7.30
Good bye
root@ip-172-31-4-17:/# cut -c 1,6 ygminds
wm
wv
Gb
root@ip-172-31-4-17:/# cut -c 1-3 ygminds
wel
we
Goo
root@ip-172-31-4-17:/#
```

Linux grep

The ‘grep’ command stands for “**global regular expression print**”. grep command filters the content of a file which makes our search easy.

grep with pipe

The ‘grep’ command is generally used with pipe ()).

```
cat ygminds | grep welcome
```

```
root@ip-172-31-4-17:/# cat ygminds
welcome to ygminds
we have to class at 7.30
Good bye
root@ip-172-31-4-17:/# cat ygminds | grep welcome
welcome to ygminds
root@ip-172-31-4-17:/# |
```

grep without pipe

```
grep <searchWord> <file name>
```

```
grep welcome ygminds
```

```
root@ip-172-31-4-17:/# cat ygminds
welcome to ygminds
we have to class at 7.30
Good bye
root@ip-172-31-4-17:/# grep welcome ygminds
welcome to ygminds
root@ip-172-31-4-17:/# |
```

- **grep -i:** The ‘grep -i’ command filters output in a case-insensitive way.

```
grep -i <searchWord> <fileName>
```

```
root@ip-172-31-4-17:/# cat ygminds
welcome to ygminds
we have to class at 7.30
Good bye
Happy Birthday To You!!!
root@ip-172-31-4-17:/# grep -i to ygminds
welcome to ygminds
we have to class at 7.30
Happy Birthday To You!!!
root@ip-172-31-4-17:/#
```

Linux comm-

The ‘comm’ command compares two files or streams. By default, ‘comm’ will always display three columns. First column indicates non-matching items of first file, second column indicates non-matching items of second file, and third column indicates matching items of both the files. Both the files has to be in sorted order for ‘comm’ command to be executed.

```
comm <file1> <file2>
```

Ex – comm file1.txt file2.txt

```
root@ip-172-31-8-13:~# cat file1.txt
Dhoni
Dravid
Sachin
Sehwag
Yuv
root@ip-172-31-8-13:~# cat file2.txt
Dhoni
Dravid
Sachin
Zadeja
root@ip-172-31-8-13:~# comm file1.txt file2.txt
          Dhoni
          Dravid
          Sachin
Sehwag
Yuv
          Zadeja
root@ip-172-31-8-13:~# |
```

Linux sed Command | Linux Stream Editor-

Linux ‘sed’ command stands for stream editor. It is used to edit streams (files) using regular expressions. But this editing is not permanent. It remains only in display, but in actual, file content remains the same.

```
sed [OPTION]... {script-only-if-no-other-script} [input-file]...
```

Global Replacement

In the earlier example, all ‘learn’ words were not edited into ‘study’. To edit every word, we have to use a global replacement ‘g’. It will edit all the specified words in a file or string.

Syntax

```
command | sed 's/<oldWord>/<newWord>/g'
```

Consider the below examples:

- echo class7 class9 | sed 's/class/jtp/g'
- cat msg.txt | sed 's/learn/study/g'

The above command will delete the lines having the word ‘jtp’. Consider the below output:

```
root@ip-172-31-8-13:~# cat file1.txt
Dhoni
Dravid
Sachin
Sehwag
Yuvi
root@ip-172-31-8-13:~# cat file2.txt
Dhoni
Dravid
Sachin
Zadeja
root@ip-172-31-8-13:~# comm file1.txt file2.txt
                  Dhoni
                  Dravid
                  Sachin
Sehwag
Yuvi
      Zadeja
root@ip-172-31-8-13:~# |
```

Removing a Line –

The ‘d’ option will let us remove a complete line from a file. We only need to specify a word from that line with ‘d’ option, and that line will be deleted. But, note that all the lines having that same word will be deleted. It will be executed as:

```
cat <fileName> | sed '/<Word>/d'
```

Ex. cat msg.txt | sed '/jtp/d'

```
root@ip-172-31-42-137:~# cat msg.txt
this is jtp
welcome to jtp
learn linux
linux is very easy
its interesting
root@ip-172-31-42-137:~# cat msg.txt | sed '/jtp/d'
learn linux
linux is very easy
its interesting
root@ip-172-31-42-137:~# |
```

Replacing Characters –

```
sed 's!/bin/bash!/bin/csh!' /etc/passwd
```

We can use the exclamation mark (!) as a string delimiter. For example, we want to replace bash shell and replace it with csh shell in the “/etc/passwd”. To do so, execute the below command:

```
sed 's!/bin/bash!/bin/csh!' /etc/passwd
```

Limiting the sed

The basic use of the sed command process the entire file. But, we can limit the sed command and specify any line. There are two ways to limit the sed command:

- A range of lines.
- A pattern that matches a specific line.

```
sed '3s/Red/Blue/' exm.txt
```

The above command will apply the specified operation on the third line:

```
root@ip-172-31-42-137:~# cat exm.txt
Apple is red
mango is yellow
your dress color is Red
Red color suits on all
root@ip-172-31-42-137:~# sed '3s/Red/Blue/' exm.txt
Apple is red
mango is yellow
your dress color is Blue
Red color suits on all
root@ip-172-31-42-137:~# |
```

From the above output, only the line three is modified.

We can also specify a range of lines. To specify a range of lines, execute the command as follows:

```
sed '1,3s/Red/Blue/' exm.txt
```

The above command will update the specified text in lines 1 and 3. Consider the below output:

```
root@ip-172-31-42-137:~# cat exm.txt
Apple is Red
mango is yellow
your dress color is Red
Red color suits on all
root@ip-172-31-42-137:~# sed '1,3s/Red/Blue/' exm.txt
Apple is Blue
mango is yellow
your dress color is Blue
Red color suits on all
root@ip-172-31-42-137:~# |
```

Modifying Lines –

The ‘c’ flag is used to modify a specific line. To modify a line, execute the command as follows:

```
sed '3c\\This is a modified line.' exm.txt
```

The above command will update the line three. Consider the below output:

```
root@ip-172-31-42-137:~# sed '3c\\This is a modified line.' exm.txt
Apple is Red
mango is yellow
This is a modified line.
Red color suits on all
root@ip-172-31-42-137:~# |
```

Linux wc Command-

Linux wc command helps in counting the lines, words, and characters in a file. It displays the number of lines, number of characters, and the number of words in a file. Mostly, it is used with pipes for counting operation.

```
wc <file name>
```

The above command will display the number of lines, number of words, number of bytes, and file name from the file ‘exm.txt’. Consider the below output:

```
root@ip-172-31-42-137:~# wc exm.txt
4 16 76 exm.txt
root@ip-172-31-42-137:~# |
```

Display count information of multiple files-

To display the complete count information of multiple files at once, specify the file names after space (' '). It is executed as follows:

```
wc <file1> <file2>
```

```
wc exm.txt marks.txt
```

The above command will display the number of words, the number of characters, and the number of the bytes from the files ‘exm.txt’ and ‘msg.txt’. Consider the below output:

```
root@ip-172-31-42-137:~# wc exm.txt msg.txt
 4   16   76 exm.txt
 5   14   74 msg.txt
 9   30  150 total
root@ip-172-31-42-137:~# |
```

Display the number of lines in a file-

The ‘-l’ option is used to display the number of lines in a file. It is executed as follows:

```
wc -l <file name>
```

```
wc -l exm.txt
```

```
root@ip-172-31-42-137:~# wc -w exm.txt
16 exm.txt
root@ip-172-31-42-137:~# |
```

Display the number of characters in a file-

The ‘-m’ option is used to display the number of characters in a file. It is executed as follows:

```
wc -m <file name>
```

Display the number of words in a file –

The ‘-w’ option is used to display the total number of words from a file. It is executed as follows:

```
wc -w <file name>
```

```
wc -w exm.txt
```

The above command will display the total number of words from the file ‘exm.txt’. Consider the below output:

```
root@ip-172-31-42-137:~# wc -w exm.txt
16 exm.txt
root@ip-172-31-42-137:~# |
```

Count the number files in a directory –

To count the number of files and folders in a directory, combine the wc command with the ls cmd. Execute it as follows:

```
ls | wc -l
```

The above command will display the count of the files from the current working directory. Consider the below output:

```
root@ip-172-31-42-137:~# ls | wc -l
3
root@ip-172-31-42-137:~# |
```

What is the awk command?

awk is a scripting language, and it is helpful when working in the command line. It's also a widely used command for text processing.

When using awk, you are able to select data – one or more pieces of individual text – based on a pattern you provide.

For example, some of the operations you can do with awk are searching for a specific word or pattern in a piece of text given, or even select a certain line or a certain column in a file you provide.

It looks something like this:

```
awk '{action}' your_file_name.txt
```

When you want to search for text that has a specific pattern or you're looking for a specific word in the text, the command would look something like this:

```
awk '/regex pattern/{action}' your_file_name.txt
```

To print all the contents of a file, the action you specify inside the curly braces is print \$0.

This will work in exactly the same way as the cat command mentioned previously.

```
awk '{print $0}' information.txt
```

```
root@ip-172-31-42-137:~# vim information.txt
root@ip-172-31-42-137:~# cat information.txt
fristName      lastName      age      city      ID
Thomas        Shelby        30       Rio       400
Omega          Night         45       Ontario   600
Wood           Tinker        54       Lisbon    N/A
Giorgos        Georgiou     35       London    300
Timmy          Turner        32       Berlin    N/A
root@ip-172-31-42-137:~# awk '{print $0}' information.txt
fristName      lastName      age      city      ID
Thomas        Shelby        30       Rio       400
Omega          Night         45       Ontario   600
Wood           Tinker        54       Lisbon    N/A
Giorgos        Georgiou     35       London    300
Timmy          Turner        32       Berlin    N/A
root@ip-172-31-42-137:~# |
```

If you would like each line to have a line-number count, you would use the **NR** built-in variable:

```
awk '{print NR,$0}' information.txt
```

```
root@ip-172-31-42-137:~# awk '{print NR,$0}' information.txt
1 fristName      lastName      age      city      ID
2
3 Thomas        Shelby        30       Rio       400
4 Omega          Night         45       Ontario   600
5 Wood           Tinker        54       Lisbon    N/A
6 Giorgos        Georgiou     35       London    300
7 Timmy          Turner        32       Berlin    N/A
root@ip-172-31-42-137:~# |
```

How to print specific columns using awk?

When using awk, you can specify certain columns you want printed.

To have the first column printed, you use the command:

```
awk '{print $1}' information.txt
```

```
root@ip-172-31-42-137:~# awk '{print $1}' information.txt
fristName
Thomas
Omega
Wood
Giorgos
Timmy
root@ip-172-31-42-137:~# |
```

To print the second column, you would use **\$2**

```
root@ip-172-31-42-137:~# awk '{print $2}' information.txt
lastName
Shelby
Night
Tinker
Georgiou
Turner
root@ip-172-31-42-137:~# |
```

To print more than one column, for example the first and forth columns, you would do:

```
root@ip-172-31-42-137:~# awk '{print $1, $4}' information.txt
fristName city
Thomas Rio
Omega Ontario
Wood Lisbon
Giorgos London
Timmy Berlin
root@ip-172-31-42-137:~# |
```

To print the last field (the last column), you can also use **\$NF** which represents the last field in a record:

```
root@ip-172-31-42-137:~# awk '{print $NF}' information.txt
ID
400
600
N/A
300
N/A
root@ip-172-31-42-137:~# |
```

How to print specific lines of a column?

You can also specify the line you want printed from your chosen column:

awk '{print \$1}' information.txt |head-1

```
root@ip-172-31-42-137:~# awk '{print $1}' information.txt | head -1
fristName
```

How to print out lines with a specific pattern in awk?

You can print a line that **starts** with a specific letter.

awk '/^O/' information.txt

awk '/^W/' information.txt

```
root@ip-172-31-42-137:~# awk '/0$/' information.txt
Thomas      Shelby      30      Rio      400
Omega       Night       45      Ontario   600
Giorgos     Georgiou    35      London    300
root@ip-172-31-42-137:~# |
```

You can also print a line that **ends** in a specific pattern:

awk '/0\$/' information.txt

```
root@ip-172-31-42-137:~# awk '/0$/' information.txt
Thomas          Shelby        30      Rio       400
Omega           Night         45      Ontario   600
Giorgos         Georgiou     35      London    300
root@ip-172-31-42-137:~#
```

7. Linux Networking Commands

Linux ifconfig –

The command ifconfig stands for interface configurator. This command enables us to initialize an interface, assign IP address, enable or disable an interface. It display route and network interface.

ifconfig

Hostname -I

```
root@ip-172-31-8-13:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
      inet 172.31.8.13  netmask 255.255.240.0  broadcast 172.31.15.255
      inet6 fe80::87b:4cff:fecc:da54  prefixlen 64  scopeid 0x20<link>
        ether 0a:7b:4c:cb:da:54  txqueuelen 1000  (Ethernet)
          RX packets 14055  bytes 2127912 (2.1 MB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 12256  bytes 1374101 (1.3 MB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
          RX packets 130  bytes 14273 (14.2 KB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 130  bytes 14273 (14.2 KB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

root@ip-172-31-8-13:~# hostname -i
172.31.8.13
root@ip-172-31-8-13:~# |
```

Get details of specific interface-

ifconfig eth0

```
root@ip-172-31-8-13:~# ifconfig eth0
Command 'ifcofig' not found, did you mean:
  command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntu5)
Try: apt install <deb name>
root@ip-172-31-8-13:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 9001
      inet 172.31.8.13 netmask 255.255.240.0 broadcast 172.31.15.255
      inet6 fe80::87b:4cff:fe:da54 prefixlen 64 scopeid 0x20<link>
        ether 0a:7b:4c:cb:da:54 txqueuelen 1000 (Ethernet)
          RX packets 14186 bytes 2136033 (2.1 MB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 12376 bytes 1386979 (1.3 MB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@ip-172-31-8-13:~# |
```

Linux traceroute command-

Linux traceroute command is a network troubleshooting utility that helps us determine the number of hops and packets traveling path required to reach a destination. It is used to display how the data transmitted from a local machine to a remote machine. Loading a web page is one of the common examples of the traceroute. A web page loading transfers data through a network and routers. The traceroute can display the routes, IP addresses, and hostnames of routers over a network. It can be useful for diagnosing network issues.

apt install inetutils-traceroute

apt install traceroute

The above commands will install the traceroute utility on our system. After the successful installation, the output will look like as follows:

traceroute ygminds.com

```
root@ip-172-31-8-13:~# traceroute ygminds.com
traceroute to ygminds.com (103.171.45.241), 64 hops max
 1  52.66.0.243  4.632ms  16.366ms  1.674ms
 2  100.65.19.64  4.743ms  8.162ms  8.114ms
 3  100.66.8.138  4.165ms  8.205ms  8.186ms
 4  100.66.11.160  6.260ms  8.085ms  8.509ms
 5  100.66.6.103  55.192ms  61.146ms  58.743ms
 6  100.66.4.157  6.818ms  8.171ms  7.938ms
 7  100.65.8.65  0.514ms  0.513ms  0.566ms
 8  99.83.77.25  6.368ms  1.425ms  1.114ms
 9  52.95.67.174  4.011ms  5.965ms  5.902ms
10  52.95.65.224  0.651ms  1.131ms  0.766ms
11  115.114.89.57  0.804ms  0.707ms  0.688ms
12  *  *  *
13  219.65.44.178  22.596ms  22.711ms  22.493ms
14  180.179.197.214  22.594ms  22.699ms  22.628ms
15  180.179.194.124  26.635ms  24.119ms  24.283ms
16  *  *  *
```

Linux tracepath –

It is similar to traceroute command, but it doesn't require root privileges. By default, it is installed in Ubuntu but you may have to download traceroute on Ubuntu. It traces the network path of the specified destination and reports each hop along the path. If you have a slow network then tracepath will show you where your network is weak.

tracepath <destination>

```
ubuntu@ip-172-31-8-13:~$ tracepath ygminds.com
1?: [LOCALHOST]                                pmtu 1500
1:  no reply
2:  no reply
3:  no reply
4:  no reply
5:  no reply
6:  no reply
7:  100.65.10.65                               18.321ms asymm 8
8:  99.83.77.31                                0.654ms
9:  99.83.76.242                               1.495ms asymm 14
10: 52.95.65.230                                0.599ms asymm 12
11: 115.114.89.57.static-Mumbai.vsnl.net.in    0.976ms asymm 13
12:  no reply
13: 219.65.44.178.static-delhi.vsnl.net.in     22.608ms asymm 18
14: 180.179.197.202                            21.442ms asymm 19
15: 180.179.194.124                            27.670ms asymm 20
16:  no reply
17:  no reply
```

Linux ping Command-

Linux ping command stands for (Packet Internet Groper). It checks connectivity between two nodes to see if a server is available. It sends ICMP ECHO_REQUEST packets to network hosts and displays the data on the remote server's response. It checks if a remote host is up, or that network interfaces can be reached. Further, it is used to check if a network connection is available between two devices. It is also handy tool for checking your network connection and verifying network issues.

Ping command keeps executing and sends the packet until you interrupt.

The ping command supports various command-line options. But, the basic syntax for the ping command is as follows:

ping <option> <destination>

```
root@ip-172-31-8-13:~# ping ygminds.com
PING ygminds.com (103.171.45.241) 56(84) bytes of data.
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=1 ttl=44 time=21.7 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=2 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=3 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=4 ttl=44 time=21.6 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=5 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=6 ttl=44 time=21.7 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=7 ttl=44 time=21.7 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=8 ttl=44 time=21.7 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=9 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=10 ttl=44 time=21.7 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=11 ttl=44 time=21.7 ms
```

We can limit the number of sent packets by using the ping command. To limit the packet, specify the 'c' option followed by the number of packets to be sent. It will be executed as:

ping -c <number> <destination>

Ping -c 5 ygminds.com

```
root@ip-172-31-8-13:~# ping -c 5 ygminds.com
PING ygminds.com (103.171.45.241) 56(84) bytes of data.
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=1 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=2 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=3 ttl=44 time=22.3 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=4 ttl=44 time=21.8 ms
64 bytes from server.exabyteserver.com (103.171.45.241): icmp_seq=5 ttl=44 time=21.8 ms

--- ygminds.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 21.822/21.934/22.314/0.190 ms
root@ip-172-31-8-13:~# |
```

Linux netstat Command-

Linux netstat command stands for **Network statistics**. It displays information about different interface statistics, including open sockets, routing tables, and connection information. Further, it can be used to displays all the socket connections (including TCP, UDP). Apart from connected sockets, it also displays the sockets that are pending for connections. It is a handy tool for network and system administrators.

apt install net-tools

netstat - a

```
root@ip-172-31-8-13:~# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:ssh              0.0.0.0:*
tcp      0      0 0.0.0.0:http             0.0.0.0:*
tcp      0      0 localhost:domain        0.0.0.0:*
tcp      0     332 ip-172-31-8-13.ap-s:ssh 150.107.192.87:37514 ESTABLISHED
tcp6     0      0 [::]:ssh                [::]:*
tcp6     0      0 [::]:http               [::]:*
udp      0      0 localhost:domain        0.0.0.0:*
udp      0      0 ip-172-31-8-13.a:bootpc 0.0.0.0:*
udp      0      0 localhost:323           0.0.0.0:*
udp      0      0 ip-172-31-8-13.ap:55740 ip-172-31-0-2.ap:domain ESTABLISHED
udp6     0      0 ip6-localhost:323       [::]:*
raw6     0      0 [::]:ipv6-icmp         [::]:*
Active UNIX domain sockets (servers and established)
7
```

Linux dig Command (DNS Lookup)-

Linux dig command stands for **Domain Information Groper**. This command is used for tasks related to DNS lookup to query DNS name servers. It mainly deals with troubleshooting DNS related problems. It is a flexible utility for examining the DNS (Domain Name Servers). It is used to perform the DNS lookups and returns the queried answers from the name server. Usually, it is used by most DNS administrators to troubleshoot the DNS problems. It is a straightforward tool and provides a clear output. It is more functional than other lookups tools.

The dig command supports plenty of command-line options. Additionally, it facilitates batch mode, which is useful for accessing the lookup requests from a file. If it is not specified to the dig command to query a specific name server, it will access each of the servers from “/etc/resolv.conf.” The dig without any command-line options will perform an NS query for “.” (the root).

Query a Domain name

dig ygminds.com

```
ubuntu@ip-172-31-8-13:~$ dig ygminds.com

; <>> DiG 9.18.1-1ubuntu1.1-Ubuntu <>> ygminds.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24936
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
; QUESTION SECTION:
ygminds.com.          IN      A

; ANSWER SECTION:
ygminds.com.      300     IN      A      103.171.45.241

; Query time: 95 msec
; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
; WHEN: Sat Jul 09 07:29:34 UTC 2022
; MSG SIZE  rcvd: 56
```

Linux nslookup –

This command is also used to find DNS related query.

nslookup <domainName>

```
root@ip-172-31-8-13:~# nslookup ygminds.com
Server:      127.0.0.53
Address:      127.0.0.53#53

Non-authoritative answer:
Name:    ygminds.com
Address: 103.171.45.241

root@ip-172-31-8-13:~#
```

8. Linux Users & Groups Management

Linux Users

User management includes everything from creating a user to deleting a user on your system. User management can be done in two ways on a Linux system.

Graphical tools are easy and suitable for new users, as it makes sure you'll not run into any trouble.

Command line tools includes commands like useradd, userdel, passwd, etc. These are mostly used by the server administrators.

1. useradd username
2. passwd username

```
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
1xd:x:999:100::/var/snap/1xd/common/1xd:/bin/false
```

Accessing a user configuration file-

```
cat /etc/passwd
```

Command to Modify the group ID of a user-

```
usermod -g new_group_id username
```

```
root@ip-172-31-8-13:~# usermod -g 1002 Ram
root@ip-172-31-8-13:~#
```

```
txu.x.999.100.../var/snap/txu/common/txu./bin/false
Ram:x:1001:1002::/home/Ram:/bin/sh
Shyam:x:1002:1002::/home/Shyam:/bin/sh
root@ip-172-31-8-13:~#
```

Delete User-

```
root@ip-172-31-8-13:~/# userdel Shyam
root@ip-172-31-8-13:~/# cat /etc/passwd
```

```
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
1xd:x:999:100:/var/snap/1xd/common/1xd:/bin/false
```

User is deleted hence no reference found for the user Shyam

Linux Groups-

The groupadd command creates or add a group in our system.

```
groupadd <groupName>
```

```
groupadd DevOps
```

```
txu.x.999.100.../var/snap/txu/common/txu./bin/false
Ram:x:1001:1002::/home/Ram:/bin/sh
Shyam:x:1002:1002::/home/Shyam:/bin/sh
root@ip-172-31-8-13:~#
```

Group File-

The /etc/group file defines the group membership. A user can be a member of more than one group.

```
root@ip-172-31-8-13:~# tail -4 /etc/passwd
_chrony:x:113:120:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
lxr:x:999:100::/var/snap/lxr/common/lxr:/bin/false
RAM:x:1001:1003::/home/RAM:/bin/sh
root@ip-172-31-8-13:~#
```

Gruopdel –

The command groupdel will delete a group permanently from the system.

```
groupdel <group>
```

```
root@ip-172-31-8-13:~# tail -6 /etc/group
ubuntu:x:1000:
Ram:x:1001:
DevOps:x:1002:
RAM:x:1003:
suraj:x:1004:
Radha:x:1005:
root@ip-172-31-8-13:~# groupdel Radha
groupdel: cannot remove the primary group of user 'Radha'
root@ip-172-31-8-13:~# groupdel -f Radha
root@ip-172-31-8-13:~# tail -6 /etc/group
_chrony:x:120:
ubuntu:x:1000:
Ram:x:1001:
DevOps:x:1002:
RAM:x:1003:
suraj:x:1004:
root@ip-172-31-8-13:~#
```

File **/etc/gshadow** keeps the information about the group administrators as shown in below snapshot.

```
root@ip-172-31-8-13:~# tail -4 /etc/gshadow
Ram:!::
DevOps:!::
RAM:!::
suraj:!::
root@ip-172-31-8-13:~# |
```

10. Important Linux Commands

find –

The find command is used to find a particular file within a directory. It also supports various options to find a file such as byname, by type, by date, and more.

The following symbols are used after the find command:

(.) : For current directory name

(/) : For root

find . -name “*.txt”

```
root@ip-172-31-8-13:~# ls
file1.tar.bz2  file1.tar.gz  file1.txt  file2.txt  snap
root@ip-172-31-8-13:~# find . -name "*.txt"
./file1.txt
./file2.txt
root@ip-172-31-8-13:~# |
```

locate –

The locate command is used to search a file by file name. It is quite similar to find command; the difference is that it is a background process. It searches the file in the database, whereas the find command searches in the file system. It is faster than the find command. To find the file with the locates command, keep your database updated.

locate <file name>

```
root@ip-172-31-8-13:~# locate file1
/root/file1.tar.bz2
/root/file1.tar.gz
/root/file1.txt
root@ip-172-31-8-13:~#
```

date –

Date

```
root@ip-172-31-8-13:~# date
Fri Jul  8 14:02:47 UTC 2022
root@ip-172-31-8-13:~#
```

df –

The df command is used to display the disk space used in the file system. It displays the output as in the number of used blocks, available blocks, and the mounted directory.

```
df -h
```

```
root@ip-172-31-42-137:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  1.6G  6.1G  20% /
tmpfs           484M    0  484M   0% /dev/shm
tmpfs           194M  828K  193M   1% /run
tmpfs           5.0M    0  5.0M   0% /run/lock
/dev/xvda15     105M  5.3M  100M   5% /boot/efi
tmpfs           97M  4.0K  97M   1% /run/user/1000
root@ip-172-31-42-137:~# |
```

```
=====
```

```
=====
```

```
du -
```

If you want to check how much space a file or a directory takes, the **du** (**Disk Usage**) command is the answer. However, the disk usage summary will show disk block numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the **-h** argument to the command line.

```
root@ip-172-31-42-137:/var# du -h
4.0K  ./mail
4.0K  ./snap/core18/common
4.0K  ./snap/core18/2409
12K   ./snap/core18
4.0K   ./snap/lxd/common/lxd
4.0K   ./snap/lxd/common/lxd-user
16K   ./snap/lxd/common
4.0K   ./snap/lxd/22923
24K   ./snap/lxd
4.0K   ./snap/amazon-ssm-agent/common
4.0K   ./snap/amazon-ssm-agent/5656
12K   ./snap/amazon-ssm-agent
4.0K   ./snap/core20/common
4.0K   ./snap/core20/1518
12K   ./snap/core20
4.0K   ./snap/snapd/common
4.0K   ./snap/snapd/16010
12K   ./snap/snapd
76K   ./snap
```

mount –

The mount command is used to connect an external device file system to the system's file system.

mount -t type <device> <directory>

```
root@ip-172-31-8-13:~# mount
/dev/xvda1 on / type ext4 (rw,relatime,discard,errors=remount-ro)
devtmpfs on /dev type devtmpfs (rw,relatime,size=488900k,nr_inodes=122225,mode=755,inode64)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,size=198228k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursive兮
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
```

mail –

mail -s “Subject” <recipient address>

```
root@ip-172-31-8-13:~# mail -s "welcome to Young Minds" ygminds73@gmail.com
Cc:
Welcome to Young Minds.

You will learn AWS/DevOps

root@ip-172-31-8-13:~# |
```

Man –

man command in Linux is used to display the user manual of any command that we can run on the terminal.

```
root@ip-172-31-42-137:~# man ls
root@ip-172-31-42-137:~# |
```

ps aux-

The **ps** (process status) command is one of the most frequently used commands in Linux. Usually it is used to get the more and detailed information about a specific process or all processes.

“aux” print all the running process in system regardless from where they have been executed.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	1.2	167412	12816	?	Ss	05:36	0:06	/sbin/init
root	2	0.0	0.0	0	0	?	S	05:36	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	05:36	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	05:36	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	05:36	0:00	[netns]
root	7	0.0	0.0	0	0	?	I<	05:36	0:00	[kworker/0:0H]
root	8	0.0	0.0	0	0	?	I	05:36	0:00	[kworker/0:1-]
root	10	0.0	0.0	0	0	?	I<	05:36	0:00	[mm_percpu_wq]
root	11	0.0	0.0	0	0	?	S	05:36	0:00	[rcu_tasks_ru]
root	12	0.0	0.0	0	0	?	S	05:36	0:00	[rcu_tasks_tr]

kill-

If you have an unresponsive program, you can terminate it manually by using the kill command. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.

There is a total of sixty-four signals that you can use, but people usually only use two signals:

SIGTERM (15) — requests a program to stop running and gives it some time to save all of its progress. If you don’t specify the signal when entering the kill command, this signal will be used.

SIGKILL (9) — forces programs to stop immediately. Unsaved progress will be lost.

Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to kill. If you don’t know the PID, simply run the command ps ux.

After knowing what signal you want to use and the PID of the program, enter the following syntax:

kill [signal option] PID

```
root@ip-172-31-42-137:~# ps -aux | grep nginx
root      3032  0.0  0.1  55212  1676 ?        Ss   08:44   0:00 nginx: master process /usr/sbin/nginx -g dae
mon on; master_process on;
www-data   3033  0.0  0.5  55844  5648 ?        S     08:44   0:00 nginx: worker process
root      3035  0.0  0.2  7004  2196 pts/3    S+   08:44   0:00 grep --color=auto nginx
root@ip-172-31-42-137:~# kill -9 3033
root@ip-172-31-42-137:~# ps -aux | grep nginx
root      3032  0.0  0.1  55212  1676 ?        Ss   08:44   0:00 nginx: master process /usr/sbin/nginx -g dae
mon on; master_process on;
www-data   3037  0.0  0.5  55844  5648 ?        S     08:44   0:00 nginx: worker process
root      3040  0.0  0.2  7004  2176 pts/3    S+   08:44   0:00 grep --color=auto nginx
root@ip-172-31-42-137:~# |
```

Wget-

The Linux command line is super useful — you can even download files from the internet with the help of the **wget** command. To do so, simply type **wget** followed by the download link.

```
root@ip-172-31-42-137:~# wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.17.2.tar.xz
--2022-07-10 08:54:21--  https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.17.2.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.153.176, 2a04:4e42:24::432
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.153.176|:443... connected.
HTTP request sent, awaiting response... |
```

Uname –

The **uname** command, short for Unix Name, will print detailed information about your Linux system like the machine name, operating system, kernel, and so on.'

```
root@ip-172-31-42-137:~# uname
Linux
root@ip-172-31-42-137:~# uname -r
5.15.0-1011-aws
root@ip-172-31-42-137:~# |
```

History –

When you've been using Linux for a certain period of time, you'll quickly notice that you can run hundreds of commands every day. As such, running **history** command is particularly useful if you want to review the commands you've entered before.

```
root@ip-172-31-42-137:~# history
 1 clear
 2 apt install nginx
 3 clear
 4 systemctl start nginx
 5 systemctl status nginx
 6 clear
 7 id
 8 clear
 9 ps aux
10 clear
11 systemctl status nginx
12 ps -aux | nginx
13 .
```

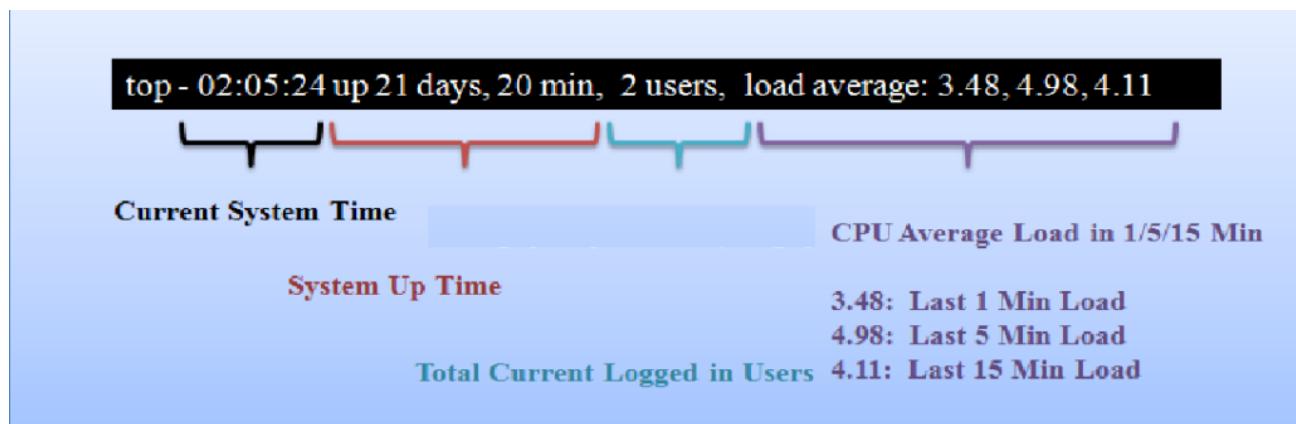
```
=====
=====
```

Top –

As a terminal equivalent to Task Manager in Windows, the **top** command will display a list of running processes and how much CPU each process uses. It's very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3077	root	20	0	10904	3936	3228	R	0.3	0.4	0:00.01	top
1	root	20	0	167412	12824	8192	S	0.0	1.3	0:06.54	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
8	root	20	0	0	0	0	I	0.0	0.0	0:00.04	kworker/0:1-cgroup_destroy
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
13	root	20	0	0	0	0	S	0.0	0.0	0:00.12	ksoftirqd/0
14	root	20	0	0	0	0	I	0.0	0.0	0:00.53	rcu_sched
15	root	st	0	0	0	0	S	0.0	0.0	0:00.00	migration/0

- **PID:** Shows task's unique process id.
- **PR:** The process's priority. The lower the number, the higher the priority.
- **VIRT:** Total virtual memory used by the task.
- **USER:** User name of owner of task.
- **%CPU:** Represents the CPU usage.
- **TIME+:** CPU Time, the same as 'TIME', but reflecting more granularity through hundredths of a second.
- **SHR:** Represents the Shared Memory size (kb) used by a task.
- **NI:** Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
- **%MEM:** Shows the Memory usage of task.
- **RES:** How much physical RAM the process is using, measured in kilobytes.
- **COMMAND:** The name of the command that started the process



Category: Azure DevOps



Azure Boards



Azure Repos



Azure Pipelines



Azure Test Plans



Azure Artifacts

Plan, track, and discuss work across teams, deliver value to your users faster.

Unlimited cloud-hosted private Git repos. Collaborative pull requests, advanced file management, and more.

CI/CD that works with any language, platform, and cloud. Connect to GitHub or any Git provider and deploy continuously to any cloud.

The test management and exploratory testing toolkit that lets you ship with confidence.

Create, host, and share packages. Easily add artifacts to CI/CD pipelines.

The Boards service in Azure DevOps is the management hub of the project. Boards can be used to plan, track, and collaborate between team members.

Azure Boards provides software development teams with the interactive and customizable tools they need to manage their software projects. It provides a rich set of capabilities including native support for Agile, Scrum, and Kanban processes, calendar views, configurable dashboards, and integrated reporting.

- **Why do we use Azure boards?**
- **Visual, Interactive Tool**

Azure Boards is the interface that helps you track the tasks, bugs, and features within your software project. It allows you to track all of your ideas in every development stage to keep your team aligned with all the necessary code changes that are linked to your work operations.

- **Easy to Customize**

Kanban boards, task boards, and delivery plans are easy to configure and customize through the user interface.

For example, with Kanban boards, you can configure columns, swim lanes, card styles, fields shown on cards, and more. You can configure them all through a common configuration dialog.

- Define [area paths and iteration paths](#) to group work items by product or feature. You also can group work items into sprints, milestones, or other time-related periods.

The screenshot shows the Azure DevOps Boards interface for the 'Fabrikam Fiber' project. The top navigation bar includes 'Azure DevOps', 'fabrikamprime / Fabrikam Fiber / Boards / Boards', and a search bar. The main area is titled 'Fabrikam Team' with a star icon and a gear icon. Below this, there are tabs for 'Board' (selected), 'Analytics', 'View as Backlog', 'Stories', and filter icons. The board itself has three columns: 'Backlog' (2/10 items), 'Analyze' (3/6 items), and 'Develop' (3/6 items). Each column contains several work item cards. For example, in the 'Analyze' column, there is a card for '352 Hello World Web Site' assigned to 'Jamal Hartnett' with a state of 'Approved'. In the 'Develop' column, there is a card for '1069 Permissions' assigned to 'Raisa Pokrovskaya' with a state of 'Committed'. The sidebar on the left is titled 'Fabrikam Fiber' and includes links for 'Overview', 'Boards' (selected), 'Work items', 'Backlogs', 'Sprints', 'Queries', and 'Delivery Plans'.

- Monitor status and progress with built-in dashboards and analytics

Backlogs display work items as a list and boards display them as cards.

You use your product backlog to quickly plan and prioritize your work.

You use your sprint backlogs and task boards when you work in Scrum.

You use your Kanban board to update work status and when you employ Kanban methods.

Each backlog is associated with a board, changes to priority order you make in one are reflected in its corresponding board.

Plans allow you to review the deliverables for several teams across sprints and a calendar schedule

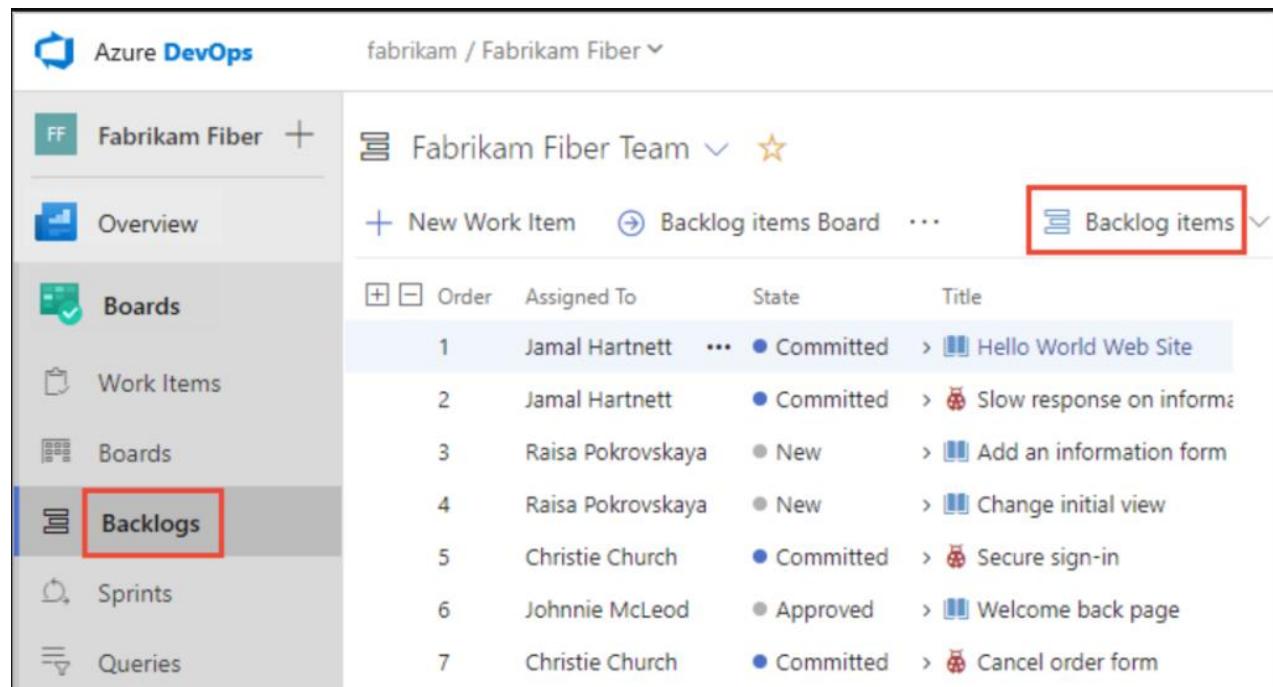
Backlogs, boards, and plans are configurable for each team.

Sprints:

#	Order	Assigned To	Remaining Work	Title
1	Jamal Hartnett	...		> Hello World Web Site
2	Raisa Pokrovskaya	6		> Cancel order form
3	Jamal Hartnett	5		> GSP locator interface
4	Jamal Hartnett	3		> Request support
5	Jamal Hartnett			> Check service status
6	Raisa Pokrovskaya	8		> Cancel order form
7	Raisa Pokrovskaya	14		> Phone sign in

Sprint backlogs and task boards provide a filtered view of work items a team has assigned to a specific iteration path, or sprint. Sprints are defined for a project and then selected by teams. From your backlog, you can map work to an iteration path using drag-and-drop, and then view that work in a separate sprint backlog.

Backlog:



The screenshot shows the Azure DevOps interface for the 'Fabrikam Fiber' project under the 'Fabrikam Fiber Team'. The left sidebar is a navigation menu with the following items: Overview, Boards, Work Items, Boards, Backlogs (which is highlighted with a red box), Sprints, and Queries. The main area displays a table titled 'Backlog items' with the following columns: Order, Assigned To, State, and Title. There are seven items listed:

Order	Assigned To	State	Title
1	Jamal Hartnett	Committed	Hello World Web Site
2	Jamal Hartnett	Committed	Slow response on informa
3	Raisa Pokrovskaya	New	Add an information form
4	Raisa Pokrovskaya	New	Change initial view
5	Christie Church	Committed	Secure sign-in
6	Johnnie McLeod	Approved	Welcome back page
7	Christie Church	Committed	Cancel order form

With Backlogs, you can quickly plan your project by adding user stories or requirements to your product backlog. Once you have your plan in place, you can start driving code development efforts.

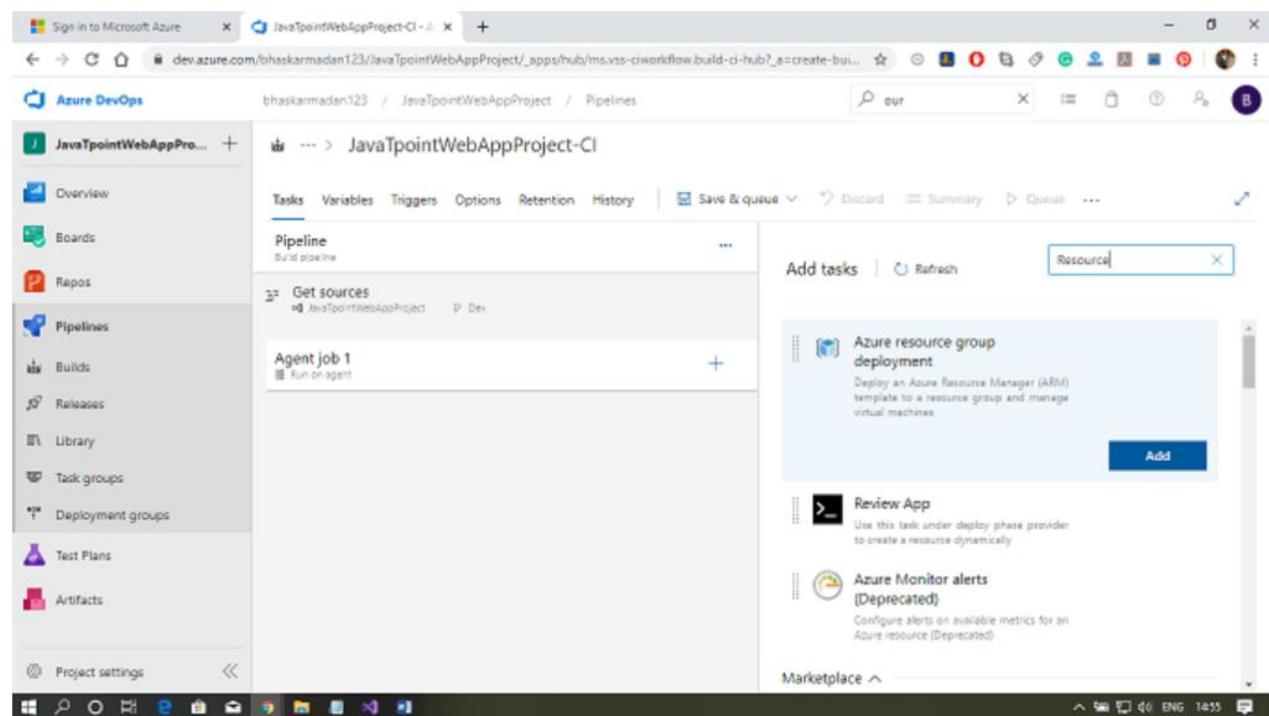
Azure backlog consists of a list of work items. You use work items to share information, assign work to team members, track dependencies, organize work, and more. Because the most important work appears at the top of the list, your team always knows what to work on next.

- To change sprint dates follow the [link](#).

2. Azure Repo

Azure Repos is a set of version control tools that you can use to manage your code. Whether your software project is large or small, using version control as soon as possible is a good idea. Version control systems are software that help you track changes you make in your code over time.

As you can see, in the corporate world, sometimes the development of a project takes an unexpected direction. The Connectix is a great example of such a case. The company released a product called Connectix Virtual PC, which has been later acquired by Microsoft. They changed its name to Microsoft Virtual PC first and then Windows Virtual PC later. There is a popular opinion among many IT professionals that Hyper-V originated from this project exactly. There has been quite a long road that leads from a single machine virtualization tool to a hypervisor that can virtualize and manage multiple virtual machines in a cluster.



Version control systems are software that help you track changes you make in your code over time. As you edit your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it. Use version control to save your work and coordinate code changes across your team.

Even if you're just a single developer, version control helps you stay organized as you fix bugs and develop new features. Version control keeps a history of your development so that you can review and even roll back to any version of your code with ease.

Azure Repos provides two types of version control:

- Git: distributed version control
- Team Foundation Version Control (TFVC): centralized version control

TFVC

Azure Repos also supports Team Foundation Version Control (TFVC). TFVC is a centralized version control system. Typically, team members have only one version of each file on their dev machines. Historical data is maintained only on the server. Branches are path-based and created on the server.

Get started by creating a project, configuring your workspace, and reviewing and sharing your code. You can use any one of these clients or IDEs:

Visual Studio

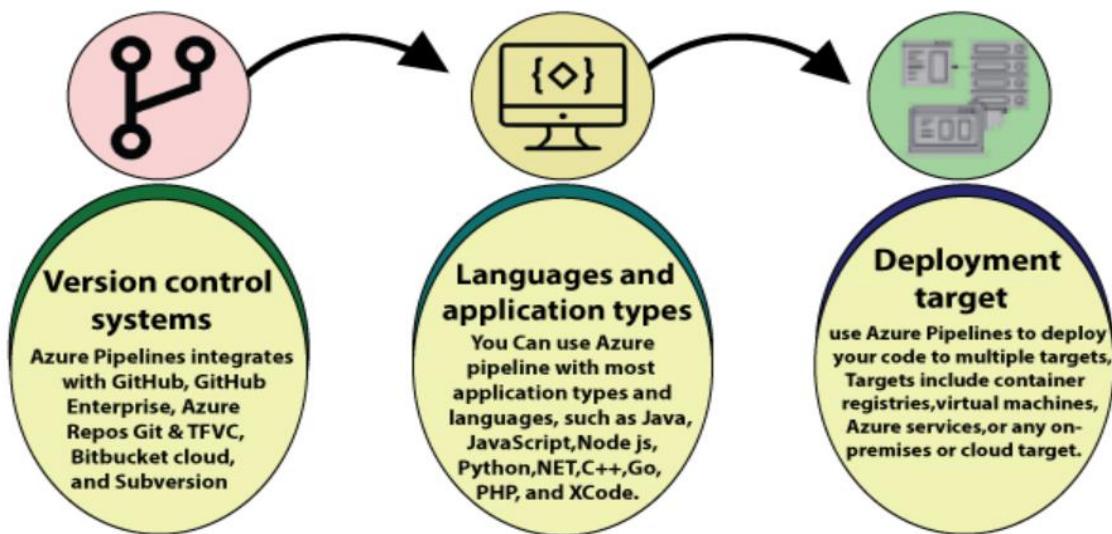
Xcode

Eclipse

3. Azure Pipelines

Pipelines are the [CI/CD](#) tool that facilitates automated building, [testing](#), and deployment. Azure Pipelines supports any programming language or platform which enables users to create pipelines that support Windows, Linux, and macOS using cloud-hosted agents.

Azure Pipelines automatically builds and tests code projects to make them available to others. It works with just about any language or project type. Azure Pipelines combines continuous integration (CI) and continuous delivery (CD) to test and build your code and ship it to any target. The Azure pipeline has a lot of capabilities such as continuous integration and continuous delivery to regularly and consistently test and build our code and ship to any target. There are three key distinct advantages of using Azure DevOps pipelines.



Version control system: Azure Pipelines integrates with GitHub, GitHub Enterprise, Azure Repos Git & TFVC, Bitbucket Cloud, and Subversion.

Language and application types: We can use Azure Pipeline with most application types and languages, such as Java, JavaScript, Node.js, Python, .Net, C++, Go, PHP, and Xcode.

Deployment target: We can use Azure Pipelines to deploy our code to multiple targets. Targets include – container registries, virtual machines, Azure services, or any on-premises or cloud target.

Azure DevOps Pipeline concepts

1. **Pipeline:** It is a workflow that defines how our test, build, and deployment steps are run.
2. **Stage:** It is a logical boundary in the pipeline. It can be used to mark the separation of concerns. Each stage contains one or more jobs.
3. **Job:** A stage can contain one or more jobs. Each job runs on an agent. It represents an execution boundary of a set of steps.
4. **Step:** It is the smallest building block of a pipeline. It can either be a script or a task. A task is simply an already created script offered as a convenience to you.
5. **Agent and Agent pools:** An agent is an installable software that runs one job at a time. Instead of managing each agent individually, you organize agents into agent pools.
6. **Artifact:** It is a collection of files or packages published by a run. The Artifact is made available to subsequent tasks, such as distribution or deployment.
7. **Trigger:** It is something that is set up to tell the pipeline when to run. We can configure a pipeline to run upon a push to the repository, at scheduled times, etc.
8. **Environment:** It is a collection of resources, where you deploy your application. It contains one or more virtual machines, containers, web apps, etc.
9. **Checks:** Checks define a set of validations required before a deployment can be performed.

10. Runs: It represents a single execution of a pipeline and collects the logs associated with running the steps and the results of running tests.

These pipelines are easily extensible through the extensions available in the marketplace. Besides, they support advanced workflows that can be used to facilitate:

- Multi-phase builds
- Test integrations
- Custom reporting functions

On top of that, Azure Pipelines provide native container support, enabling them to push containers to container registries from the pipeline directly. The pipelines offer flexibility to deploy to multiple environments from Kubernetes clusters to serverless functions and even deploy to other cloud providers such as AWS or GCP.

4. Azure Artifacts

Azure Artifacts enable developers to consume and publish different types of packages to Artifacts feeds and public registries such as NuGet.org and npmjs.com. You can use Azure Artifacts in conjunction with Azure Pipelines to deploy packages, publish build artifacts, or integrate files between your pipeline stages to build, test, or deploy your application.

Publish and Download Pipeline Artifacts

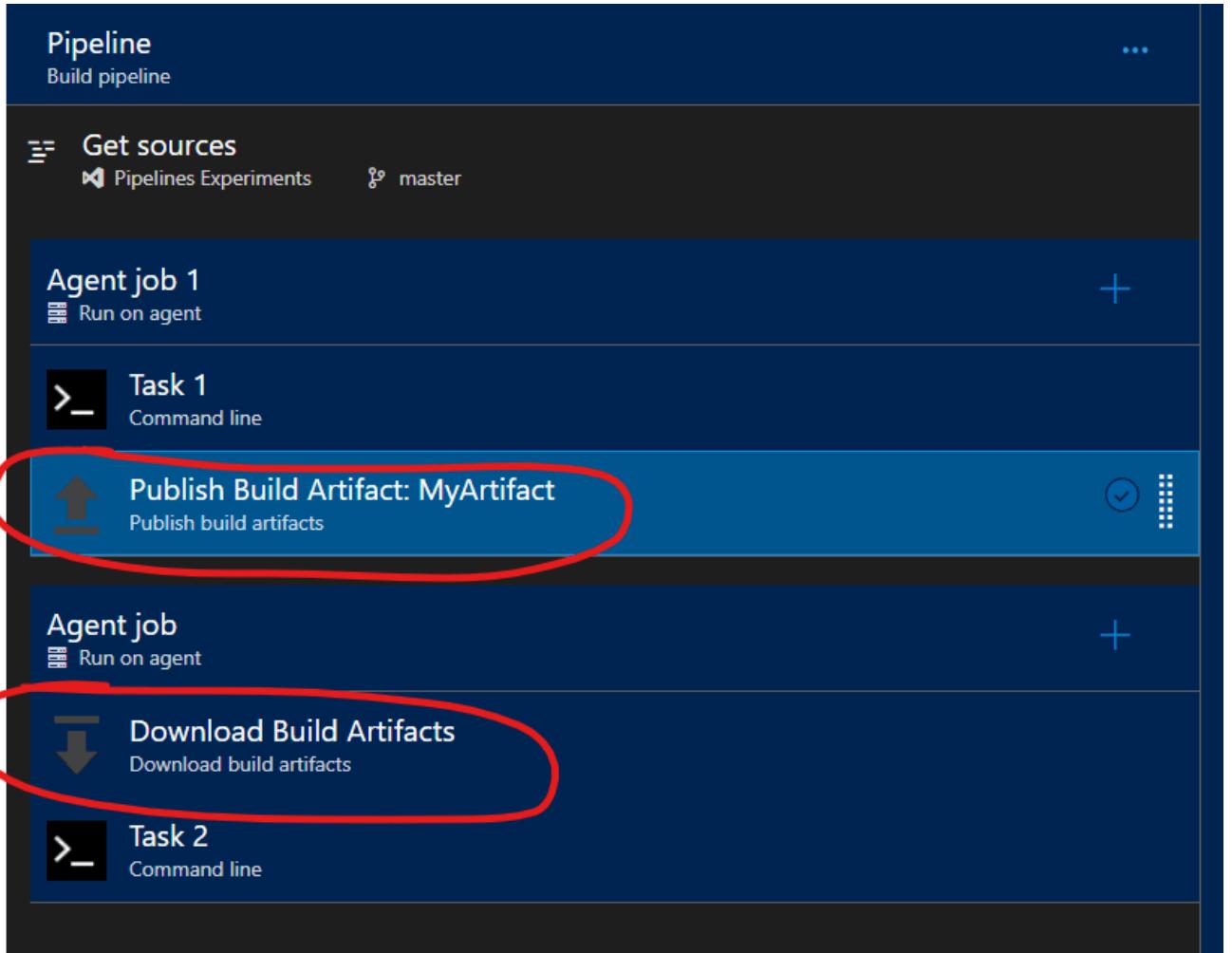
Azure DevOps Services

Using Azure Pipelines, you can download artifacts from earlier stages in your pipeline or from another pipeline. You can also publish your artifact to a file share or make it available as a pipeline artifact.

Build Artifacts

Build Artifacts have been in Azure DevOps for a long time and are the **built-in artifact storage mechanism** for Azure Pipelines.

They can be used in both the Classic Build Pipelines, the one created using the UI, as well as the newer YAML Pipelines.



Build Artifacts are published via the Publish Build Artifacts task and can be downloaded with the Download Build Artifact task. And when you publish them, you can instruct the task to either push the content up to the Azure DevOps cloud or serve, or to copy the files to a local file share instead.

Build Artifacts can be consumed from other jobs in the same Pipeline, and from other Pipelines.

This screenshot shows the Azure DevOps interface for managing artifacts. On the left, a pipeline diagram displays two stages: 'Dev' and 'Prod'. The 'Dev' stage has one job with one task, and the 'Prod' stage has one job with one task. To the right, a detailed view of the 'Source type' section is shown. It includes icons for 'Build' (selected), 'Azure Repos ...', 'GitHub', and 'TFVC'. Below these are dropdown menus for 'Project' (set to 'Pipelines Experiments'), 'Source (build pipeline)' (set to 'ASP.NET-CI-Classic'), 'Default version' (set to 'Latest'), and 'Source alias' (set to 'ASP.NET-CI-Classic').

Additionally, Build Pipelines can be used if you want to consume your artifact from a Release Pipeline triggered by the build completion.

And you can always download your artifacts from the Build run status page.

Name	Size	⋮
▼ MyArtifact	17 B	
myfile.txt	17 B	

And as you can see in the image above, you can explore the content of your artifact directly in the UI.

Pipeline Artifacts

Alright, let's talk about **Pipelines Artifacts** now.

These are the **newer version**, if you will, of Build Artifacts, and as such they can be used only from within the YAML Pipelines.

One of the main benefits of Pipeline Artifacts is that they can **dramatically reduce the time** it takes to upload and download the artifacts because of the way the files are both uploaded and stored. And this is especially true for large artifacts.

Until fairly recently, Pipelines Artifacts couldn't be used in Classic Release Pipelines, or from other Pipelines, but now that limitation is gone so their usage is very similar to Build Artifacts.

```
31     Settings
32     - task: VSTest@2
33     - inputs:
34       - platform: '$(buildPlatform)'
35       - configuration: '$(buildConfiguration)'
36
37     Settings
38     - task: PublishPipelineArtifact@1
39     - inputs:
40       - targetPath: '$(Pipeline.Workspace)'
41       - artifact: 'My Artifact'
42       - publishLocation: 'pipeline'
43
44   - job: Job2
45     displayName: Second Job
46     pool:
47       - vmImage: 'ubuntu-latest'
48     variables:
49       - thisIsLocal: true
50     steps:
51       Settings
52       - task: DownloadPipelineArtifact@2
53       - inputs:
54         - buildType: 'current'
55         - artifactName: 'MyArtifact'
56         - targetPath: '$(Pipeline.Workspace)'
```

To publish the Pipelines Artifacts you can use the Publish Pipeline Artifact task and you can download them using the Download Pipeline Artifact task.

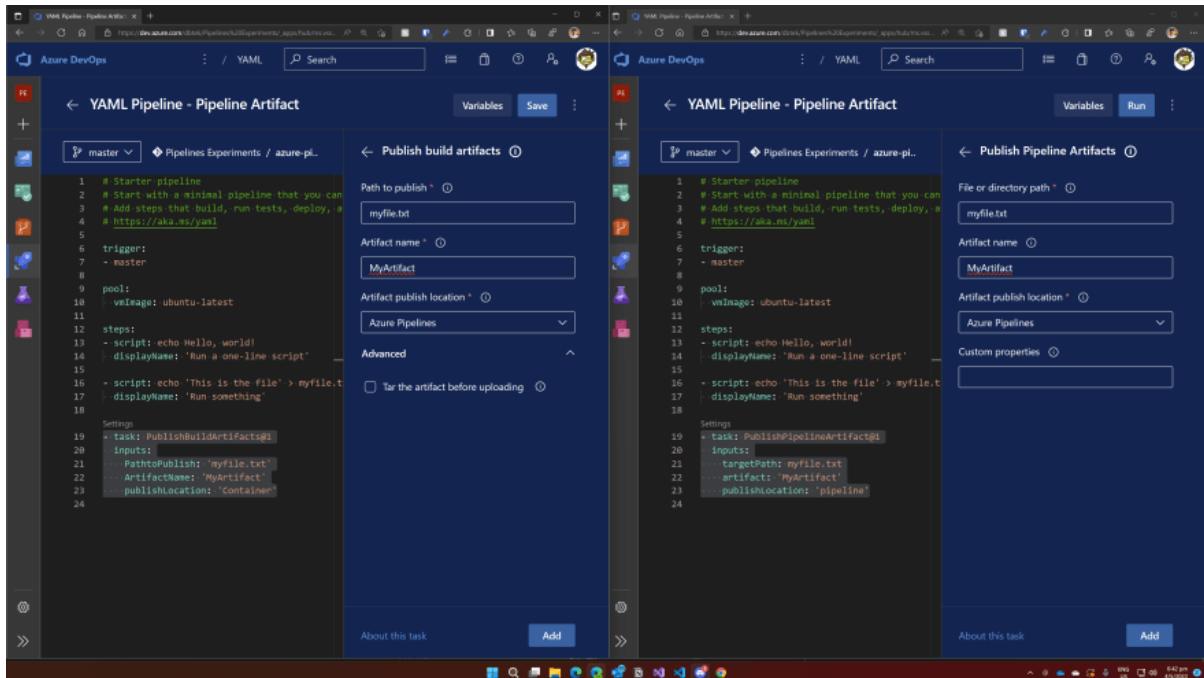
```
32     Settings
33     - task: VSTest@2
34     - inputs:
35       - platform: '$(buildPlatform)'
36       - configuration: '$(buildConfiguration)'
37
38     - publish:
39       - target: '$(Pipeline.Workspace)'
40       - artifact: 'My Artifact'
41
42   - job: Job2
43     displayName: Second Job
44     pool:
45       - vmImage: 'ubuntu-latest'
46     variables:
47       - thisIsLocal: true
48     steps:
49       - download: DownloadPipelineArtifact@2
50       - inputs:
51         - artifact: 'MyArtifact'
52         - target: '$(Pipeline.Workspace)'
53         - prefix: 'Cache Doing something'
```

Alternatively, since this feature is only available in the YAML Pipelines, you can use the publish keyword and the download keyword, which are just the abbreviation for the whole tasks.

And if you publish a Pipeline Artifacts, and you want to use it in a deployment job in the same pipeline, you don't even have to add the download task because Azure Pipelines will download them automatically.

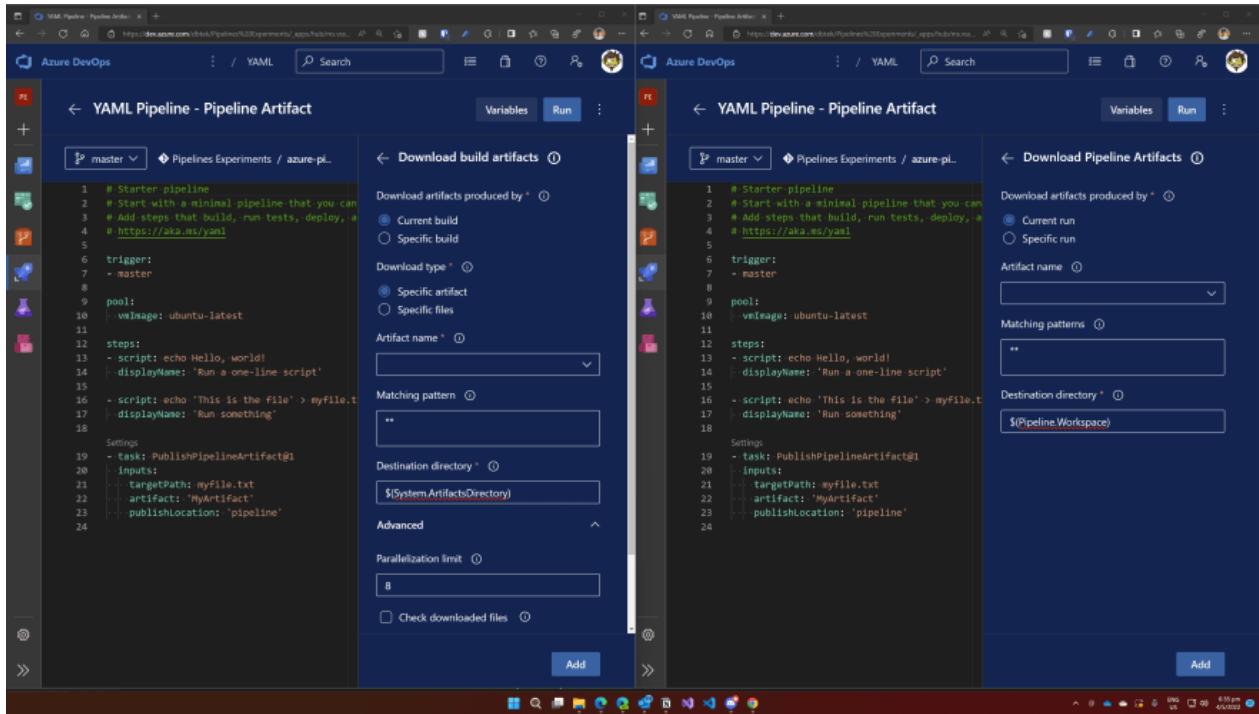
Build vs Pipeline Artifacts

There are few more differences in publishing and downloading the artifacts between Build and Pipeline Artifacts



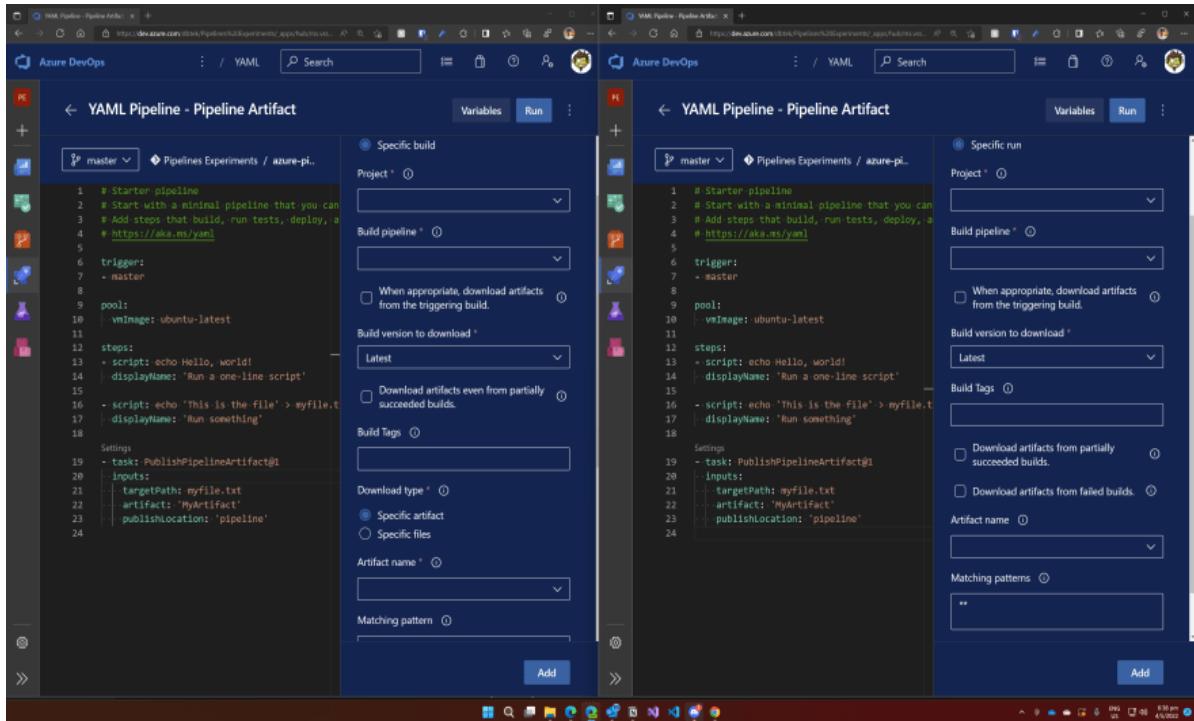
The publish tasks are virtually identical, with the only differences being that in the Publish Build Artifact task here on the left you can optionally choose to further include your artifact in a Tar file, while this is not present on the right on the Publish Pipeline Artifact Task. This one, instead, allows you to add some custom properties to the artifact. They must be in JSON format, all keys having the prefix user-

It gets more interesting if we look at the Download tasks



As you can see in this side by side view, when you download a Build Artifact (on the left) you can choose if you want to download the whole thing, or just some specific file from the artifact. You can also set some parallelization settings and other parameters. When downloading a Pipeline Artifact, instead, you don't have that option, as you can see here on the right hand side of the image.

And if we switch the task to download from a different pipeline or run, instead of from the current one, we have one more difference.



Aside from a different positioning of the fields, you can see that when you download a Pipeline Artifact you can choose to do so even if the pipeline run you are targeting has failed

And that basically covers everything there is to say about Build and Pipeline Artifacts. If you have noticed, I kept comparing the two... but I haven't mentioned Azure Artifacts yet. Why? Well, because it is a **completely different thing**.

What About Azure Artifacts?

So Azure Artifacts... as I was saying it is pretty different from Build and Pipeline Artifacts. Despite the similar name, it's a **different service which serves a different purpose**.

Both Build and Pipeline Artifacts are very generic, you can save whatever you want in them, and what Azure DevOps does is just packaging the files in a zip archive and saving it somewhere. Azure Artifacts, instead, is a **typed package repository**.

Package type	Azure DevOps Services	Azure DevOps Server	TFS
NuGet packages	Yes	Yes	TFS 2017 and TFS 2018
npm packages	Yes	Yes	TFS 2017 and TFS 2018
Maven packages	Yes	Yes	TFS 2018
Python packages	Yes	Server 2019 Update 1 and newer, Server 2020	TFS 2018
Universal Packages	Yes	No	No

It supports multiple package types such as NuGet, npm, Python, Maven, and Universal Packages... you can basically see it as an alternative to Artifactory, Nexus, GitHub Packages, and services like that.

You may have seen that Azure Artifacts also supports Universal Packages, and although that is somewhat similar to the other types of artifact we have seen before, it is conceptually different.

You would use Universal Packages when you want to create an artifact with a lifetime independent of the pipeline that created it. In fact both Build and Pipeline Artifacts are always tied to the Pipelines that created them. As we have seen, you can download Pipeline Artifacts after a pipeline has completed via the artifacts UI – but if you want something that really exists independent of pipeline you would go for Universal Packages.

Another big difference is about the pricing. Whether you use Build Artifacts or Pipeline Artifacts, you will not have to pay a single cent for them, no matter how many files you store or how big they are. Azure Artifacts, on the other hand, is billed by size.

Billing

Billing has not been set up for this organization. Access will be available up to [free tier limits](#).

[Set up billing](#)

Pipelines for private projects	Free	Paid parallel jobs
--------------------------------	------	--------------------

MS Hosted CI/CD ↳	49999 minutes	0
-----------------------------------	---------------	---

Self-Hosted CI/CD ↳	9999	0
-------------------------------------	------	---

Visit [parallel jobs](#) for full details on free pipelines and public concurrency

Boards, Repos and Test Plans	Free
------------------------------	------

Basic users ↳	5
-------------------------------	---

Basic + Test Plans ↳	Start free trial
--------------------------------------	----------------------------------

Settings	Access level
----------	--------------

Default access level for new users ↳	Stakeholder
--	-------------

Resources	Free	Used	Usage limit
-----------	------	------	-------------

Artifacts ↳	2 GiB*	0.36 GiB	Up to 2 GiB free
-----------------------------	--------	----------	------------------

*Artifacts now bills for packages-only. For other updates, please see <https://aka.ms/artbilling>.

You have a free grant of 2 Gb for each organization, but once you reach the maximum storage limit, you can no longer upload new artifacts and will need to either delete some of your existing artifacts, or [set up billing](#) to increase your storage limit.

Shell Scripting

1. Shell Scripting Introduction

How to execute the script?

1. use ./script-name.sh
2. else can do sh script-name.sh for Bourne shell
3. else bash script-name.sh for Bourne again shell execution.

This does not matter which shell you use sh or bash as for normal scripts it will have same output, this is needed when you want specific shell to run your script.

Variables-

Variable is nothing but a character string to which we assign a value, or else we can say it's a pointer to actual data.

The value can be a number, string, filename, device or any other type of data.

Types of variables

1. Local Variable
2. Environmental variable
3. Shell Variable

Local variables –

A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.

Ex.

```
Func () {  
    NAME=Variable name # here the name variable is just limited to this particular function block  
    but not in  
    the program  
}
```

Environmental variables –

An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually, a shell script defines only those environment variables that are needed by the programs that it runs.

Ex.

1. \$PATH
2. \$HOME

We can set env variables using export command.

Shell Variables –

A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

Ex.

A shell variable is created with the following syntax:

“variable_name=variable_value”

=====

=====

The following table shows a number of special variables that you can use in your shell scripts –

Sr.No.	Variable & Description
1	\$0 The filename of the current script.
2	\$n These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument (the first argument is \$1, the second argument is \$2, and so on).
3	\$# The number of arguments supplied to a script.
4	\$* All the arguments are double quoted. If a script receives two arguments, \$* is equivalent to \$1 \$2.
5	\$@ All the arguments are individually double quoted. If a script receives two arguments, \$@ is equivalent to \$1 \$2.
6	\$? The exit status of the last command executed.
7	\$\$ The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
8	\$! The process number of the last background command.

Exit Status(\$?) variable in Linux –

“\$?” is a variable that holds the return value of the last executed command. “echo \$?” displays **0** if the last command has been successfully executed and displays a **non-zero** value if some error has occurred.

```

root@ip-172-31-42-137:~# ls
exm.txt information.txt msg.txt scripts snap ygminds.txt
root@ip-172-31-42-137:~# cat msg.txt
this is jtp
welcome to jtp
learn linux
linux is very easy
its interesting
root@ip-172-31-42-137:~# echo $?
0
root@ip-172-31-42-137:~# |

root@ip-172-31-42-137:~# eecho hello
Command 'eecho' not found, did you mean:
  command 'echo' from deb coreutils (8.32-4.1ubuntu1)
Try: apt install <deb name>
root@ip-172-31-42-137:~# echo $?
127
root@ip-172-31-42-137:~#

```

Basic operators in shell-

- **Arithmetic Operators**

Assume variable **a** holds **10** and variable **b** holds **20** then

Operator	Description	Example
+(Addition)	Adds values on either side of the operator	`expr \$a + \$b` will give 30
-(Subtraction)	Subtracts right hand operand from left hand operand	`expr \$a - \$b` will give -10
*(Multiplication)	Multiplies values on either side of the operator	`expr \$a * \$b` will give 200
/(Division)	Divides left hand operand by right hand operand	`expr \$b / \$a` will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	`expr \$b % \$a` will give 0

= (Assignment)	Assigns right operand in left operand	a = \$b would assign value of b into a
== (Equality)	Compares two numbers, if both are same then returns true.	[\$a == \$b] would return false.
!= (Not Equality)	Compares two numbers, if both are different then returns true.	[\$a != \$b] would return true.

Script-

```
#!/bin/sh
val=`expr 2 + 2`
echo "Total value : $val"
```

- **Relational Operators-**

Bourne Shell supports the following relational operators that are specific to numeric values. These operators do not work for string values unless their value is numeric.

For example, following operators will work to check a relation between 10 and 20 as well as in between “10” and “20” but not in between “ten” and “twenty”.

Assume variable **a** holds **10** and variable **b** holds **20** then

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[\$a -eq \$b] is not true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[\$a -ne \$b] is true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[\$a -gt \$b] is not true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[\$a -lt \$b] is true.

-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[\$a -ge \$b] is not true.
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[\$a -le \$b] is true.

- **Boolean Operators –**

The following Boolean operators are supported by the Bourne Shell.

Assume variable **a** holds 10 and variable **b** holds 20 then –

Operator	Description	Example
!	This is logical negation. This inverts a true condition into false and vice versa.	[! false] is true.
-o	This is logical OR . If one of the operands is true, then the condition becomes true.	[\$a -lt 20 -o \$b -gt 100] is true.
-a	This is logical AND . If both the operands are true, then the condition becomes true otherwise false.	[\$a -lt 20 -a \$b -gt 100] is false.

- **String Operators –**

The following string operators are supported by Bourne Shell.

Assume variable a holds “abc” and variable b holds “efg” then –

Operator	Description	Example
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[\$a = \$b] is not true.
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[\$a != \$b] is true.

-z	Checks if the given string operand size is zero; if it is zero length, then it returns true.	[-z \$a] is not true.
-n	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.	[-n \$a] is not false.
str	Checks if str is not the empty string; if it is empty, then it returns false.	[\$a] is not false.

- **File Test Operators**

We have a few operators that can be used to test various properties associated with a Unix file.

Assume a variable **file** holds an existing file name “test” the size of which is 100 bytes and has **read**, **write** and **execute** permission on –

Operator	Description	Example
-b file	Checks if file is a block special file; if yes, then the condition becomes true.	[-b \$file] is false.
-c file	Checks if file is a character special file; if yes, then the condition becomes true.	[-c \$file] is false.
-d file	Checks if file is a directory; if yes, then the condition becomes true.	[-d \$file] is not true.
-f file	Checks if file is an ordinary file as opposed to a directory or special file; if yes, then the condition becomes true.	[-f \$file] is true.
-g file	Checks if file has its set group ID (SGID) bit set; if yes, then the condition becomes true.	[-g \$file] is false.
-k file	Checks if file has its sticky bit set; if yes, then the condition becomes true.	[-k \$file] is false.

-p file	Checks if file is a named pipe; if yes, then the condition becomes true.	[-p \$file] is false.
-t file	Checks if file descriptor is open and associated with a terminal; if yes, then the condition becomes true.	[-t \$file] is false.
-u file	Checks if file has its Set User ID (SUID) bit set; if yes, then the condition becomes true.	[-u \$file] is false.
-r file	Checks if file is readable; if yes, then the condition becomes true.	[-r \$file] is true.
-w file	Checks if file is writable; if yes, then the condition becomes true.	[-w \$file] is true.
-x file	Checks if file is executable; if yes, then the condition becomes true.	[-x \$file] is true.
-s file	Checks if file has size greater than 0; if yes, then condition becomes true.	[-s \$file] is true.
-e file	Checks if file exists; is true even if file is a directory but exists.	[-e \$file] is true.

Types of shell-

The Bourne Shell (sh) :-

It is the original UNIX shell. It is faster and more preferred. It lacks features for interactive use like the ability to recall previous commands. It also lacks built-in arithmetic and logical expression handling. It is default shell for Solaris OS.

The C shell (csh):-

It includes helpful programming features like built-in arithmetic and C-like expression syntax.

The Korn Shell (ksh):-

Korn Shell or KSH was developed by a person named David Korn, which attempts to integrate the features of other shells like C shell. Is a superset of the Bourne shell. So it supports everything in the Bourne shell. It has interactive features. It includes features like built-in

arithmetic and C-like arrays, functions, and string-manipulation facilities. It is faster than C shell. It is compatible with script written for C shell.

Bourne Again Shell (bash) :-

Bash shell was developed by the Freeware Software and it is written and licensed under the GNU organization. It is compatible to the Bourne shell. It includes features from Korn and Bourne shell. Bash shell is free and open-source to use for computer users.

Comment *

2. Shell Decision Making

we will understand shell decision-making in Unix. While writing a shell script, there may be a situation when you need to adopt one path out of the given two paths. So you need to make use of conditional statements that allow your program to make correct decisions and perform the right actions.

The if...else statements

If else statements are useful decision-making statements which can be used to select an option from a given set of options.

Unix Shell supports following forms of if...else statement –

1. if...fi statement
2. if...else...fi statement
3. if...elif...else...fi statement

Ex. 1

Syntax for If.

```
# If [expression]
```

then

Statements to be executed

Fi

Ex. 2

Syntax for if else.

```
# if
```

```
[expression/condition]
```

then

Statements to be executed when condition true

else

Statements to be executed when condition not true

fi

Ex. 3

if [expression 1]

then

 Statement(s) to be executed if expression 1 is true

elif [expression 2]

then

 Statement(s) to be executed if expression 2 is true

elif [expression 3]

then

 Statement(s) to be executed if expression 3 is true

else

 Statement(s) to be executed if no expression is true

fi

Shell Loop Types –

1. While loop
2. For loop
3. Until loop
4. Nested loop
5. Infinity loop
6. Loop control

While Loop-

The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.

while command

do

 Statement(s) to be executed if command is true

done

Ex.

```
#!/bin/bash
```

a=0

while [\$a -lt 10]

do

```
echo $a  
a=`expr $a + 1`  
done
```

For Loop-

The for loop operate on lists of items. It repeats a set of commands for every item in a list.

```
for ((initialize ; condition ; increment));  
do  
#someting  
done
```

Ex.

```
for i in 1 2 3 4 5  
do  
echo "Looping ... number $i"  
done
```

Until Loop-

- Sometimes we need to execute the set of commands until a condition is true
- In simple words, here if the condition is false, then the given statement is executed

```
until  
do  
    statement (s) to be executed until command is true  
done
```

Eg

```
a=0  
# -gt is greater than operator  
#Iterate the loop until a is greater than 10  
until [ $a -gt 10 ]  
do  
# Print the values  
echo $a
```

```
# increment the value  
a=`expr $a + 1`  
done
```

Nested loop-

Nested for loops means loop within loop. They are useful for when you want to repeat something several times for several things.

```
#!/bin/bash  
  
# A shell script to print each number five times.  
  
for (( i = 1; i <= 5; i++ ))    ### Outer for loop ###  
do  
    for (( j = 1 ; j <= 5; j++ )) ### Inner for loop ###  
    do  
        echo -n "$i "  
    done  
    echo "" ##### print the new line ###  
done
```

Loop Control-

- The Break Statement-**

The break statement is used to terminate the execution of the entire loop, after completing the execution of all of the lines of code up to the break statement. It then steps down to the code following the end of the loop.

- The Continue Statement-**

The continue statement is similar to the break command, except that it causes the current iteration of the loop to exit, rather than the entire loop.

This statement is useful when an error has occurred but you want to try to execute the next iteration of the loop.