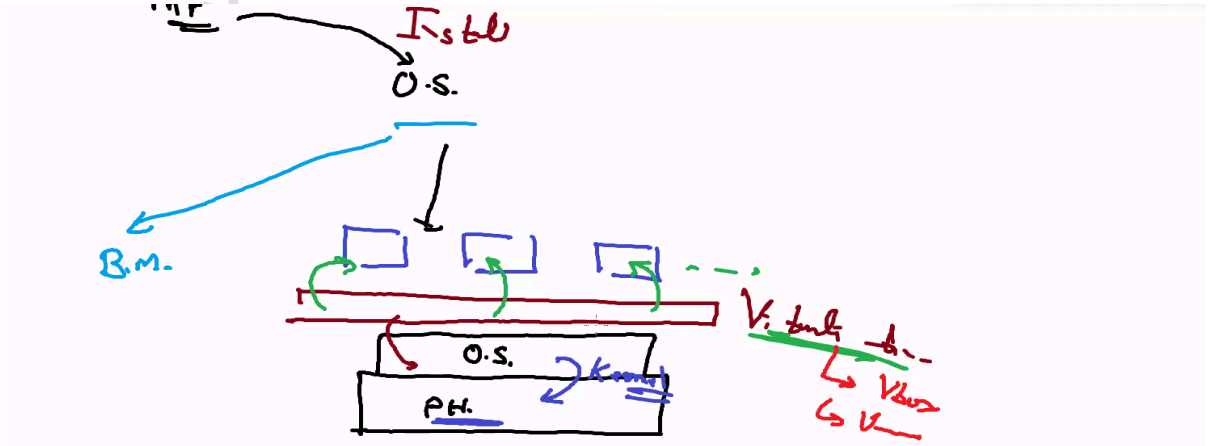
7 Days DevOps Deloitte Session 2

Summary 03-01-2025

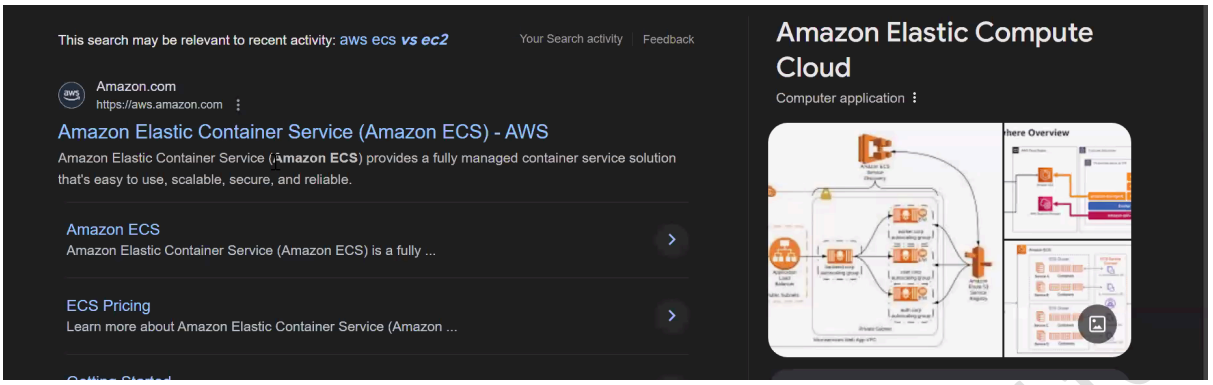
- In the world of technology, everything starts with an idea. Afterward, we select the appropriate technology and create an app or a web app.
  To run any program, a minimum requirement is an operating system (OS), which needs to be installed first.
- There are three primary ways to install an OS:

1. **Bare Metal**: This involves directly installing the operating system on physical hardware.
2. **Virtualization**: In this method, we use base hardware to install an operating system. On top of this OS, we install virtualization software like VMware or Oracle VirtualBox. This allows us to install and use multiple operating systems on a single piece of hardware simultaneously.
3. **Containerization**: In this approach, container software like Docker is installed on the operating system. Unlike virtualization, Docker can deploy and run isolated environments (containers) in just a few seconds. Each container behaves like a lightweight operating system but shares the base OS kernel.

- In the case of bare metal or containerization, only one operating system can run at any given time. However, with virtualization, multiple operating systems (referred to as virtual OS or guest OS) can run on the same hardware simultaneously.



- The only challenge with virtualization is that it doesn't provide high performance compared to bare-metal installations, and the installation process can take around 30 minutes.
- The fastest virtualization software is **Nitro**, created by AWS Cloud. Virtualization technology is implemented using software called **hypervisors**, such as KVM, Oracle VirtualBox, and many more.
- Companies needing servers can choose between virtualization servers and bare-metal servers, depending on their requirements. They can either purchase the hardware themselves or use cloud services.
- In the case of AWS Cloud, if you request a virtual machine, they provide it through their **EC2 Service**. AWS manages everything in the cloud, which is why these are referred to as **managed services**. If you request a bare-metal server, AWS also offers that option under EC2 as **Dedicated Hosts**.
- By default, AWS provides virtual machines, and their older hypervisor was **XEN**. However, they have now switched to the **Nitro hypervisor**, which offers improved performance and efficiency. This means that cloud providers like AWS don't invent their own underlying technologies; instead, they leverage hypervisors to launch instances and wrap them in a web application (such as the AWS Management Console) to make them easier to manage.
- **Container technology**, on the other hand, can deploy an operating system in just 1 second. Any operating system in the world can be fully installed within this short time frame. To implement container technology, specialized software is required, such as **Docker**, **Podman**, **Rocket**, and others.
- You also have a choice when it comes to managing container technology:
- **Self-hosted**: You manage the containers yourself.
- **Cloud-managed**: The cloud provider manages the containers for you.
- For AWS Cloud, if you need a service to manage containers, they offer **ECS Service**. When the cloud provider handles the management, it is referred to as a **cloud-managed service**.



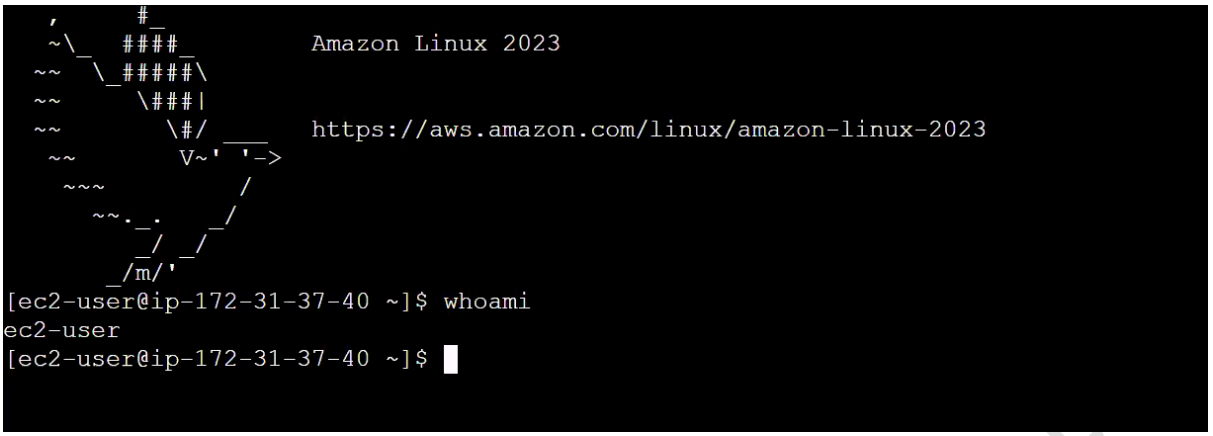
- Cloud is essentially a product that provides access to multiple technologies.
- One of the most popular container products is **Docker**, which can deploy an operating system in just one second. This is the core strength of Docker.
- In today's fast-paced world, the primary focus is on speed and agility. Once you launch an idea, new features are added regularly—sometimes even daily—to enhance the product. Developing and adding features involves constant iteration.
- The environment where applications are deployed is known as the **production system**, also referred to as a **release** or **rollout**. However, we cannot directly deploy our applications to the production system without thorough testing to ensure there are no bugs or vulnerabilities.
- To identify bugs and vulnerabilities, tools like **SonarQube** can be used. The process of changing code, testing it, and adding new features is continuous and forms part of the **Software Development Life Cycle (SDLC)**.
- In modern SDLCs, changes are made frequently—sometimes up to 100 times a day. After every code change, testing is performed, and only after successful testing does the application move to the production system. This repetitive cycle can be managed either manually or through automation.

- For automation, tools like **Jenkins** are widely used. However, there is a common challenge: most companies follow a rule where a new operating system is required after every testing cycle, regardless of whether the testing is successful or not. Installing a new OS typically takes 20–30 minutes, which results in delays and increased costs due to extended time-to-market and unnecessary employee hours.

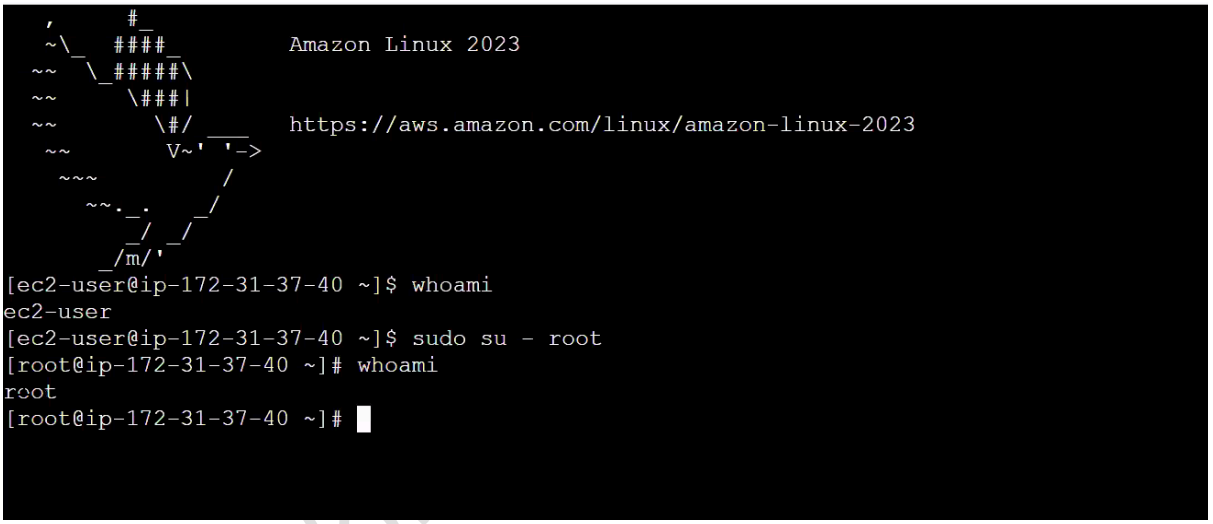
- To overcome this issue, **Docker** can be used to launch operating systems quickly. Docker can be installed on Linux, Windows, or macOS. Using **AWS EC2 Service**, we can launch Docker to create operating systems (containers) on demand.

- The OS on which Docker is installed is called the **Docker host**, while the OS launched within Docker is referred to as a **container**. Production systems can also be set up using Docker, making it easy to deploy operating systems as needed. After logging into your instance, Docker allows you to efficiently manage and deploy operating systems.

- In Linux we have the command whoami it will tell you from which account you login.



- But this user has limited power so you have to login with root account so you can use sudo su - root to login with root account.



- Docker is software and if you want to install any software we can use the yum command.
- You can use yum install docker command so it will install the docker.

```
      5/10
  Verifying        : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
      6/10
  Verifying        : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
      7/10
  Verifying        : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
      8/10
  Verifying        : pigz-2.5-1.amzn2023.0.3.x86_64
      9/10
  Verifying        : runc-1.1.14-1.amzn2023.0.1.x86_64
     10/10

Installed:
  containerd-1.7.23-1.amzn2023.0.1.x86_64     docker-25.0.6-1.amzn2023.0.2.x86_64     ipta
4
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64    libcgroup-3.0-1.amzn2023.0.1.x86_64     libn
0.2.x86_64
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64   libnftnl-1.2.2-2.amzn2023.0.2.x86_64    pigz

  runc-1.1.14-1.amzn2023.0.1.x86_64

Complete!
```

- To start the Docker service, we can use the `systemctl` command.

```
[root@ip-172-31-37-40 ~]# history
    1  whoami
    2  yum install docker
    3  systemctl  start docker
    4  history
[root@ip-172-31-37-40 ~]#
```

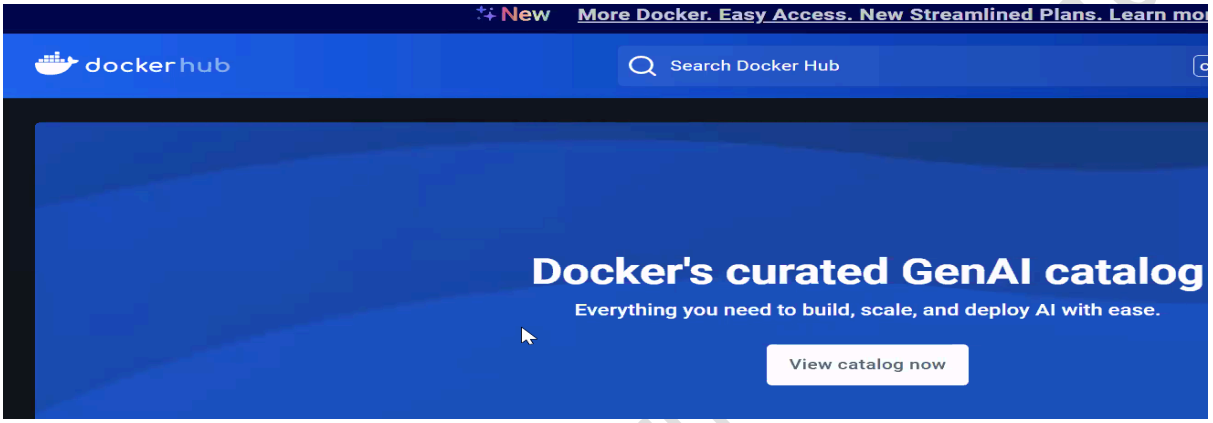
- This system now becomes our Docker host. Once you have a Docker host, you can launch as many operating systems as you want. To check how many containers are running, you can use the `docker ps` command.

```
[root@ip-172-31-37-40 ~]#
[root@ip-172-31-37-40 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-37-40 ~]#
```

- To install an operating system, we always need a setup file, such as a bootable DVD file, which is also known as an OS image. It is impossible to install an OS without an OS image. In AWS, these are referred to as **AMI images**, while in containers, they are called **Docker images** or **container images**. To check how many images are available, you can use the `docker images` command.

```
[root@ip-172-31-37-40 ~]#
[root@ip-172-31-37-40 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-37-40 ~]# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[root@ip-172-31-37-40 ~]#
```

- To download images, you can use the website **hub.docker.com**, which is a public repository where you can find all container images.



- For storing sensitive information, we primarily use private registries. In AWS, the **ECR service** is used for private registries.
- On Docker Hub, if you search for 'Ubuntu,' it will display Ubuntu images along with the available versions. To download an image, you can use the `docker pull` command.

```
[root@ip-172-31-37-40 ~]# docker pull  ubuntu:14.04
```

- The `docker pull` command, by default, always pulls images from Docker Hub. However, this can be changed if needed. You can check the downloaded images using the `docker images` command

```
[root@ip-172-31-37-40 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       14.04     13b66b487594   3 years ago   197MB
[root@ip-172-31-37-40 ~]#
```

- And for running the container using this image we can use docker run command and give the image name but to get terminal we have to give -t option and for interactive use -i options.

```
[root@ip-172-31-37-40 ~]# docker  ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-37-40 ~]# docker run  -t -i ubuntu:14.04
root@7966bed75693:/#
```

- Within the second your operating system install and you get the shell of ubuntu.

```
       6/10
[root@ip-172-31-37-40 ~]# docker  ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-37-40 ~]# docker run  -t -i ubuntu:14.04
root@7966bed75693:/# yum
bash: yum: command not found
root@7966bed75693:/# apt-get
apt 1.0.1ubuntu2.24 for amd64 compiled on Oct 10 2019 22:48:21
Usage: apt-get [options] command
       apt-get [options] install|remove pkg1 [pkg2 ...]
       apt-get [options] source pkg1 [pkg2 ...]

apt-get is a simple command line interface for downloading and
installing packages. The most frequently used commands are update
and install.
```

- You can run all the command which you use in the normal operating system.

```
root@7966bed75693:/#
root@7966bed75693:/#
root@7966bed75693:/# python
bash: python: command not found
root@7966bed75693:/# python3
Python 3.4.3 (default, Nov 12 2018, 22:25:49)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Now its your choice what you want to do with this os.
- If you want to shutdown the container then you can use the exit command.

```
[root@ip-172-31-37-40 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS         PORTS     NAMES
7966bed75693   ubuntu:14.04   "/bin/bash"   4 minutes ago   Up 4 minutes             angry_jennings
[root@ip-172-31-37-40 ~]#
```

- If you use the `docker ps` command, it will display the containers that are currently running. Each container has a unique identifier known as the **Container ID**.
- Docker has a couple of challenges. The primary purpose of Docker is to provide an OS within an OS, and the second is to launch the OS with minimal resource usage. However, when you launch a Docker container, it runs in the foreground by default, which prevents you from executing commands on the base OS.
- If you want to launch a container in the background (detached mode), you can use the `-d` option with the `docker run` command.

```
[root@ip-172-31-37-40 ~]# docker run  -d -t -i ubuntu:14.04
86eb1ad0398435f86982908a34c87a581e0008d3f3b522cffdc947ff7b998401
[root@ip-172-31-37-40 ~]#
```

```
[root@ip-172-31-37-40 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS          PORTS     NAMES
86eb1ad03984   ubuntu:14.04   "/bin/bash"   7 seconds ago    Up 6 seconds              blissful_cray
[root@ip-172-31-37-40 ~]#
```

- The challenge is, why do you launch a container? It's because you want to run an application. However, when we run a web app, it’s typically running continuously, not just occasionally.
- While this might be an Ubuntu OS, you can also launch applications within this operating system. Generally, we need an OS to run applications.
- To access the application, a client needs the IP address of the OS. If you want to check the IP address of a container, you can use the `docker inspect` command.

```
            "MacAddress": "02:42:ac:11:00:03",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "MacAddress": "02:42:ac:11:00:03",
                    "NetworkID": "6cc4b943157429098a2f79c4b2856f00931601312c2b04cf883a9b1
                    "EndpointID": "cda1d606606c4aa3c781c15da4e7f68f98451a8867415f530db987
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.3",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DriverOpts": null,
                    "DNSNames": null
                }
```

- If you want to access the website there are chrome browser but you want to use command line browser then we have curl command.

```
[root@ip-172-31-37-40 ~]# curl  http://172.17.0.3
```

- After you hit enter you will get the output coming from server.

```
[root@ip-172-31-37-40 ~]# curl  http://172.17.0.3
<body bgcolor='aqua'>
<pre>

welcome to vimal web server for testingeth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 21  bytes 1439 (1.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3  bytes 182 (182.0 B)
        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
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

- There's one challenge: when a site is running—say, Netflix running its application in a container—and clients are accessing the website, what happens if the container turns off or crashes for some reason?

- If the container stops, the application goes down, and clients won't be able to access the site because the OS is down. This is a common issue, as an OS can experience faults at any point in time.

- To solve this challenge, there are two approaches. One is manual intervention: if the container stops, a human can start it again. But humans can't work 24/7, and we are inherently lazy. We want our sites to be 100% up, and this is only possible if we remove human intervention and apply automation. This is where an additional tool can be applied to Docker.

- If the container stops, this tool monitors the container and automatically launches a new one when it goes down. This is known as **fault tolerance**, and the tool used for this is **Kubernetes**—one of its key use cases. Kubernetes is designed to keep monitoring and managing containers.

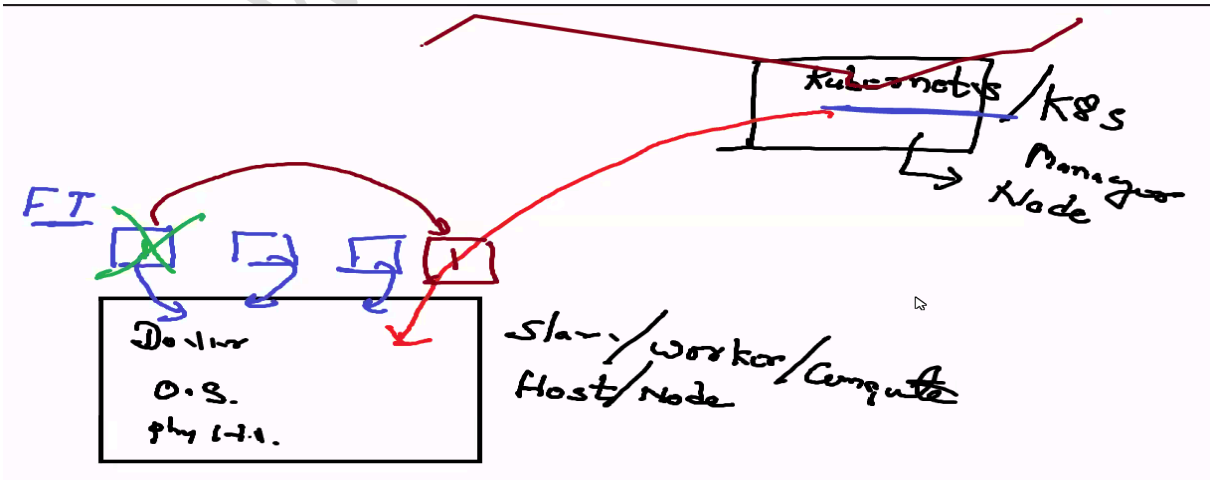
- That's why Kubernetes falls under the category of **DevOps tools**. We don't replace Docker; it's still there. But to manage Docker containers, we use Kubernetes. Kubernetes, behind the scenes, communicates with Docker to relaunch containers as needed.

- A **Docker host** is also known as a **node**. We launch Docker containers on top of this node, but managing these containers manually is not ideal. So, we use Kubernetes for automation.

- To achieve this, we take another computer and set up Kubernetes there, which is often abbreviated as **K8s**. The role of K8s is to continuously manage the nodes, Docker hosts, and containers.

- For example, Google runs over 1 billion containers per week. It would be impossible to manage this scale of containers using humans alone, so Google uses Kubernetes to manage them. Kubernetes was created by Google.

- The computer where Kubernetes is installed is called the **manager node**, while the computer running Docker hosts is referred to as the **worker node**, **slave node**, or **compute node**.



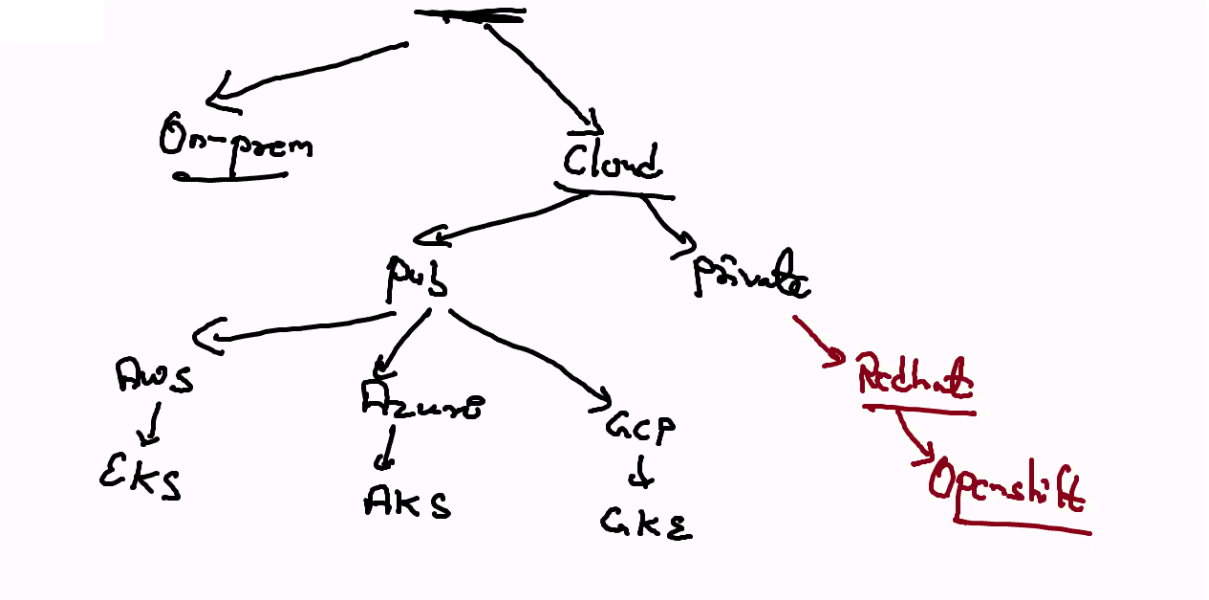
- When many computers work together, this is known as a **cluster**. In the context of containers, this is referred to as a **container cluster**. There are

other products for setting up container clusters, such as **Docker Swarm**, but one of the most popular is **Kubernetes**.

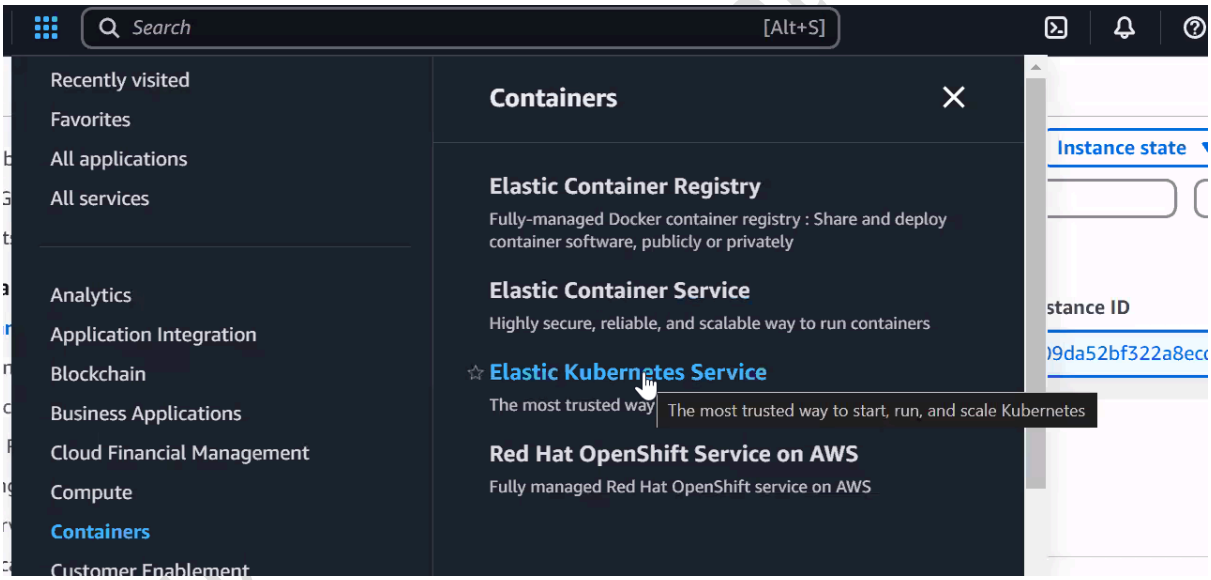
- To install Kubernetes, you have two options: you can either use it on-premises or use cloud services to manage it.
- There are two types of clouds:

1. **Public cloud** – Anyone can access and use the resources.
2. **Private cloud** – A company's personal cloud, accessible only within the company.

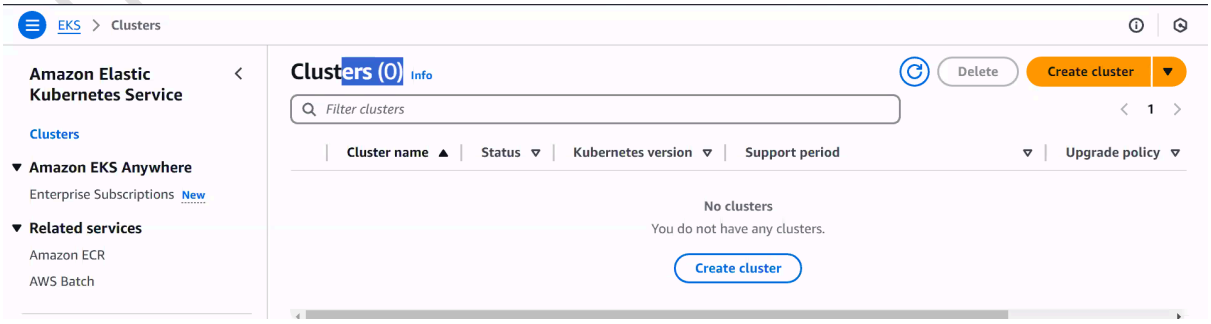
- In AWS, you can use the **EKS service** to set up Kubernetes. In Google Cloud, it's **GKE**, and in Azure, it's **AKS**. The names are different, but the use cases are the same.
- In a private cloud, **Red Hat** offers **OpenShift** as one of the products for managing Kubernetes.
- All major cloud providers support Kubernetes, which is why setting up a multi-cloud environment is very easy.
- For example, when you visit Facebook, it may look like a single application, but the search bar, chat, and other features are actually separate apps, each managed by different teams. These apps may even be created using different programming languages, a concept known as **polyglot programming**.
- A single app divided into smaller, independent services is called a **microservices architecture**. This means that different features, such as search, can run on **Google Cloud**, while chat features might be better suited for **AWS Cloud**.
- This strategy, where a single app is distributed across multiple clouds, is known as a **multi-cloud strategy**. Some data may be stored on the cloud, while other data is kept on the company's own private cloud.
- Currently, many companies are moving towards a **hybrid multi-cloud** approach.



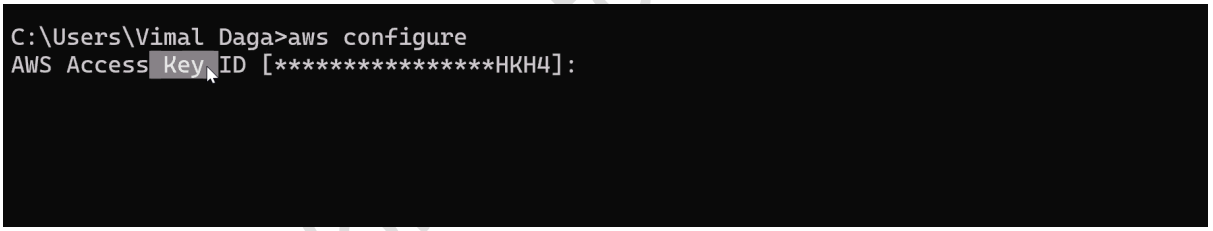
- So we are gone use the EKS.



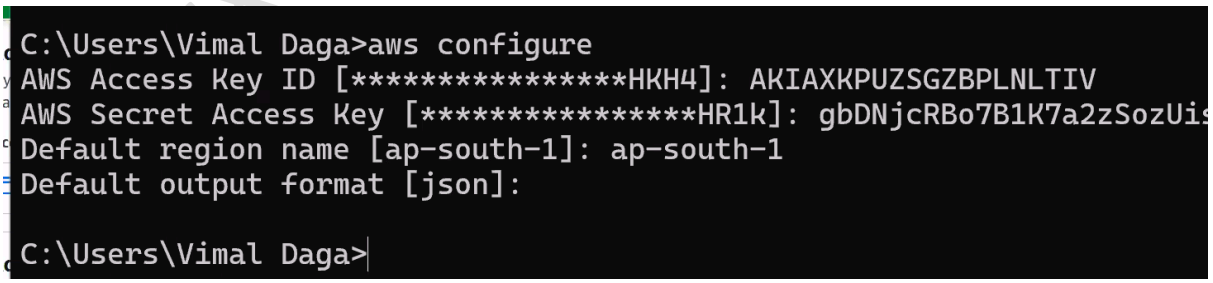
- Right now we don't have a cluster.



- To launch a cluster, you can click on **Create Cluster**, but that's a manual process. What we really want is automation, and there are two ways to achieve this: programming or using the command line.
- For this, we'll use the command line. To launch a cluster using commands in the **EKS service**, you can use the **eksctl** command. This command provides a quick and simple way to set up a cluster.
- However, before you can use the **eksctl** command, you need to authenticate it. Since the command works on your behalf, you need to provide it with your AWS credentials.
- One simple way to set up these credentials is by installing the **AWS CLI** and using the `aws configure` command to pass your keys. Once configured, any new AWS commands you run will automatically use these keys.
- To set up your AWS credentials, open your command prompt and run the `aws configure` command.



- They will ask for an access key and secret key. Create the keys in your AWS account and paste them here.



- From now on, any command that requires AWS keys will automatically retrieve them from here. It's not a good practice to hard-code the keys, which is why we always store the keys securely in our system. Now, we can use the **eksctl** command to create the EKS cluster.

```
C:\Users\Vimal Daga>eksctl create cluster --name lwcluster
```

- This command will go to AWS cloud and connect with eks services and create a cluster with master and slave cluster with multi node cluster.
- To check it created or not then we have aws command throw which we can check the cluster.

```
C:\Users\Vimal Daga>aws  eks  list-clusters
{
    "clusters": [
        "lwcluster"
    ]
}


C:\Users\Vimal Daga>
```

- Through the AWS command line, we can manage every service in the AWS cloud.
- Now that our cluster is running with the EKS service, those who will use the cluster are known as **users**. To connect with the cluster, we use the **kubectl** command.
- To get information about the nodes, you can use the `kubectl get nodes` command.

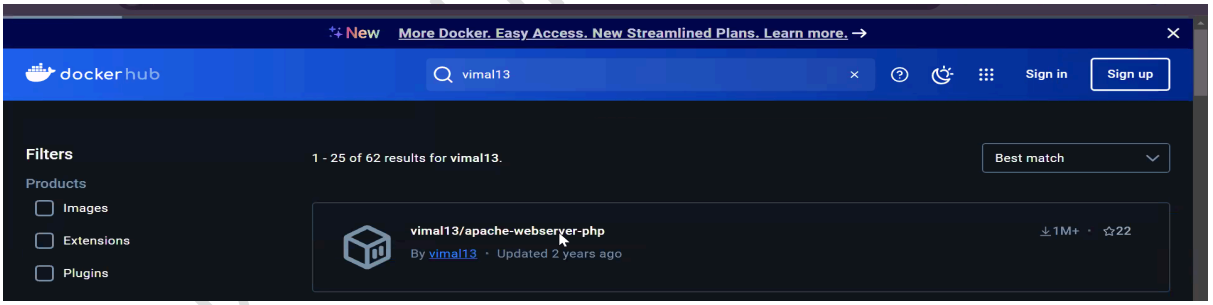
```
C:\Users\Vimal Daga>kubectl get nodes
NAME                                            STATUS   ROLES    AGE    VERSION
ip-192-168-19-184.ap-south-1.compute.internal   Ready    <none>   106m   v1.30.7-eks-59bf375
ip-192-168-66-11.ap-south-1.compute.internal    Ready    <none>   106m   v1.30.7-eks-59bf375

C:\Users\Vimal Daga>
```

- The **eksctl** command by default creates 2 nodes, but if you want to increase the number of nodes, you can modify the options.
- If you manually install Kubernetes, you would need to create and manage the master node manually. However, EKS manages the master node for you, which is why it is considered a **fully managed service**.

- So, the nodes you see when you run the `kubectl get nodes` command are **worker nodes**, not master nodes.
- The reason EKS creates two nodes by default is for fault tolerance: if a container is launched on the first node and that node goes down, Kubernetes can automatically shift the container to the second node. EKS also monitors your nodes, and if Node 1 goes down, Kubernetes will automatically shift the container to Node 2.
- This setup, where multiple nodes are used in a cluster, is known as a **multi-node cluster**. The more worker nodes you have, the lower the chance of failure, improving fault tolerance.
- The main use of Kubernetes is container orchestration. In Kubernetes, containers are called **pods**. To check how many pods you have, you can use the `kubectl get pods` command.

```
C:\Users\Vimal Daga>kubectl get  pods
No resources found in default namespace.
```

- If you want to launch the pods in kubernetes from this image.



- In kubernetes we can use kubectl create deployment command to create a deployment.

```
C:\Users\Vimal Daga>kubectl  create deployment myweb  --image=vimal13/apache-webserver-php
```

- If you launch pod using the deployment then if the container fail or down then it will automatically launch the copy of that container in another node.

```
C:\Users\Vimal Daga>kubectl get  pods
NAME                     READY   STATUS              RESTARTS   AGE
myweb-59f99ddd48-kcdrp   0/1     ContainerCreating   0          5s

C:\Users\Vimal Daga>kubectl get  pods
NAME                     READY   STATUS    RESTARTS   AGE
myweb-59f99ddd48-kcdrp   1/1     Running   0          18s
```

- If you want to delete the pod in Kubernetes, you can use the `kubectl delete pod` command followed by the pod ID.

```
C:\Users\Vimal Daga>kubectl delete pod  myweb-59f99ddd48-kcdrp
pod "myweb-59f99ddd48-kcdrp" deleted
```

- But after delete kubernetes automatically relaunch that pod.

```
C:\Users\Vimal Daga>kubectl delete pod  myweb-59f99ddd48-kcdrp
pod "myweb-59f99ddd48-kcdrp" deleted

C:\Users\Vimal Daga>kubectl get  pods
NAME                     READY   STATUS              RESTARTS   AGE
myweb-59f99ddd48-n7gm8   0/1     ContainerCreating   0          5s
```

- You can also create the copy of this container this copies are known as replicas.
- Deployment is a controller program of a kubernetes which help to launch the contaienere if you want to check the deployment then you can use kubectl get deployment command.

```
C:\Users\Vimal Daga>kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myweb   1/1     1            1           5m19s

C:\Users\Vimal Daga>kubectl get  pods
```

- Deployment is the one who is controlling the pod if deployment is deleted then your pod also deleted.

```
C:\Users\Vimal Daga>kubectl get  pods
NAME                     READY   STATUS    RESTARTS   AGE
myweb-59f99ddd48-n7gm8   1/1     Running   0          4m27s

C:\Users\Vimal Daga>kubectl delete deployment  myweb
deployment.apps "myweb" deleted

C:\Users\Vimal Daga>kubectl get deployment
No resources found in default namespace.

C:\Users\Vimal Daga>kubectl get  pods
No resources found in default namespace.
```

- To make more replicas you can use the scale command to create more replicas.

```
C:\Users\Vimal Daga>kubectl scale deployment myweb  --replicas=5
deployment.apps/myweb scaled

C:\Users\Vimal Daga>kubectl get  pods
NAME                     READY   STATUS    RESTARTS   AGE
myweb-59f99ddd48-cgv7f   1/1     Running   0          5s
myweb-59f99ddd48-ffmn8   1/1     Running   0          37s
myweb-59f99ddd48-h6xzc   1/1     Running   0          5s
myweb-59f99ddd48-pfgjs   1/1     Running   0          5s
myweb-59f99ddd48-vmqnb   1/1     Running   0          5s
```

- If you want to know the ip address of the pod then we have -o wide option in get pods command.

```
C:\Users\Vimal Daga>kubectl get  pods  -o wide
NAME                     READY   STATUS    RESTARTS   AGE     IP              NODE
                                   NOMINATED NODE   READINESS GATES
myweb-59f99ddd48-cgv7f   1/1     Running   0          2m25s   192.168.79.15   ip-192-168-66-
11.ap-south-1.compute.internal    <none>           <none>
myweb-59f99ddd48-ffmn8   1/1     Running   0          2m57s   192.168.16.169  ip-192-168-19-
184.ap-south-1.compute.internal   <none>           <none>
myweb-59f99ddd48-pfgjs   1/1     Running   0          2m25s   192.168.16.53   ip-192-168-19-
184.ap-south-1.compute.internal   <none>           <none>
myweb-59f99ddd48-vmqnb   1/1     Running   0          2m25s   192.168.94.160  ip-192-168-66-
11.ap-south-1.compute.internal    <none>           <none>
```

- If you are running with 4 pods, these are individual operating systems. Whenever the first client comes, I want to send the request to the first OS, then to the second OS, and so on, to distribute the load. This concept is known as load balancing, and when the load is distributed one by one, this algorithm is known as round robin.

- However, to perform load balancing, we need a load balancer program. We normally hit the load balancer program, and then that program routes us to the main computer or target computer. Kubernetes also provides a

facility for load balancing; it has an internal load balancer, and in Kubernetes, these are known as services (svc).

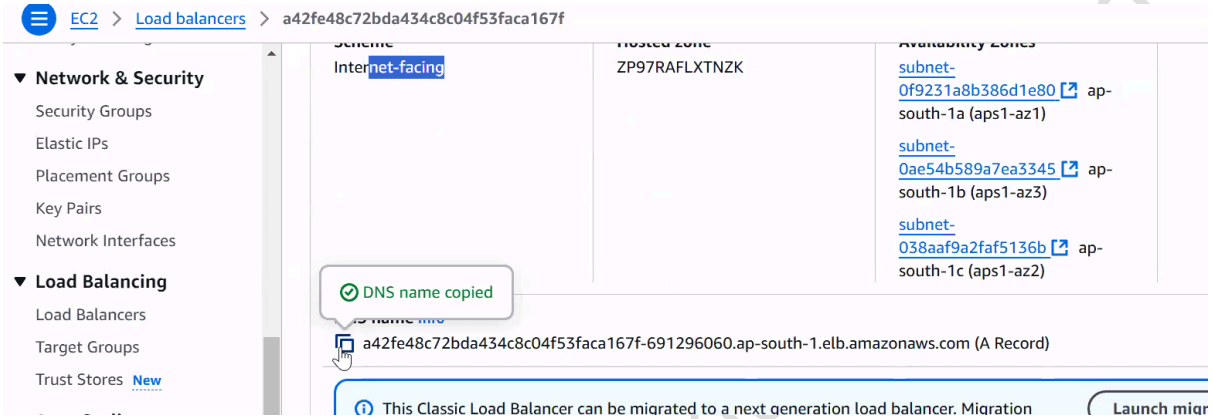
```
C:\Users\Vimal Daga>kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   151m
```

- There is one load balancer running internally, but we are not going to use this. We are going to create our own load balancer through which we can take requests from the public and distribute the load to the pods.
- There are private and public load balancers. Private load balancers don't provide internet access; they work in a LAN, while public load balancers work in a WAN.
- Any load balancer has a type called `ClusterIP`, which is private. You can't connect to the public world using this load balancer. However, if you want to connect to the internet, this is called 'exposing' the service. For that, we have the `expose` command in Kubernetes.
- If you want to check the help for a command, you can add `-h` at the end to view the documentation. If you want a load balancer with public access, then we have `NodePort` and `LoadBalancer`.
- If you use `NodePort`, Kubernetes provides its own load balancer, but if you use `LoadBalancer`, it will use the load balancer provided by the platform it's running on. For example, if Kubernetes is running on EKS, AWS uses the ELB load balancer, and Kubernetes will use AWS's ELB load balancer.
- The Kubernetes internal load balancer doesn't offer many features, which is why we always prefer to use an external load balancer. Every load balancer has its network limits, and the ELB load balancer can handle very large traffic.
- So, we are going to use the `LoadBalancer` type. In the program through which we connect to the internet, there are specific port numbers, such as port 25 for mail servers and port 80 for web servers. In the `expose` command, you have to specify which port your application is running on.

```
C:\Users\Vimal Daga>kubectl expose deployment myweb  --type LoadBalancer --port 80
```

```
C:\Users\Vimal Daga>kubectl expose deployment myweb  --type LoadBalancer --port 80 --target-p
ort 80
```

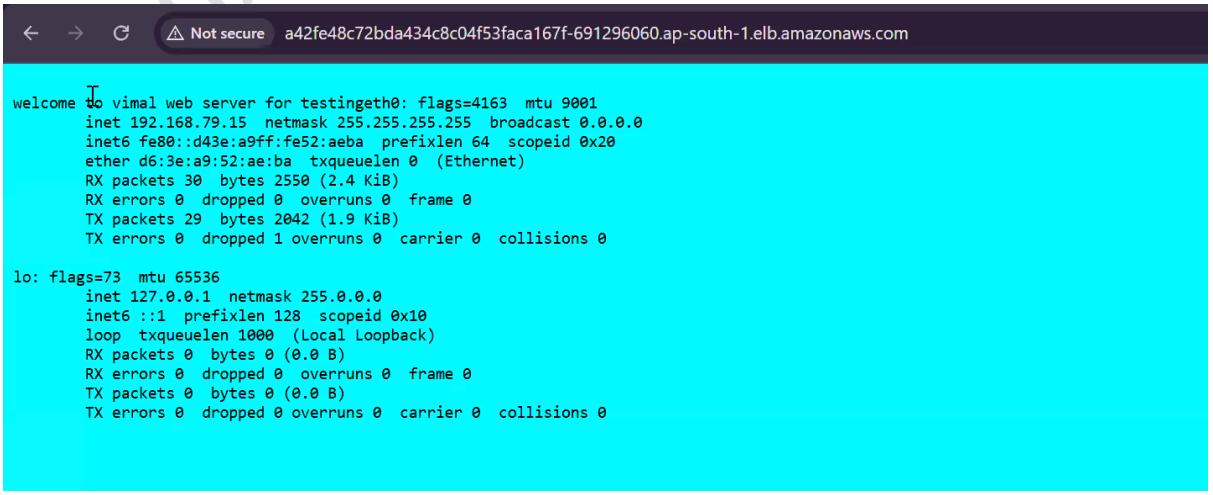
- This command creates the load balancer, and if you check the load balancer in the ELB service, you will see it there.



- You can also see it using the `kubectl get svc` command.

```
C:\Users\Vimal Daga>kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP
                       PORT(S)        AGE
kubernetes   ClusterIP      10.100.0.1       <none>
                       443/TCP        166m
myweb        LoadBalancer   10.100.145.211   a42fe48c72bda434c8c04f53faca167f-691296060.ap-so
uth-1.elb.amazonaws.com   80:32719/TCP   2m31s

C:\Users\Vimal Daga>
```

- If you copy this URL and paste it in the browser, you will see your website.



- And if you refresh the page again, you will be connected to another pod, and the traffic will be distributed automatically. If you think the client traffic is not coming, you can scale in.

```
C:\Users\Vimal Daga>kubectl scale deployment myweb  --replicas=1
```

- The command "eksctl delete cluster" is being executed to remove a Kubernetes cluster.

```
C:\Users\Vimal Daga>eksctl delete cluster --name lwcluster
```