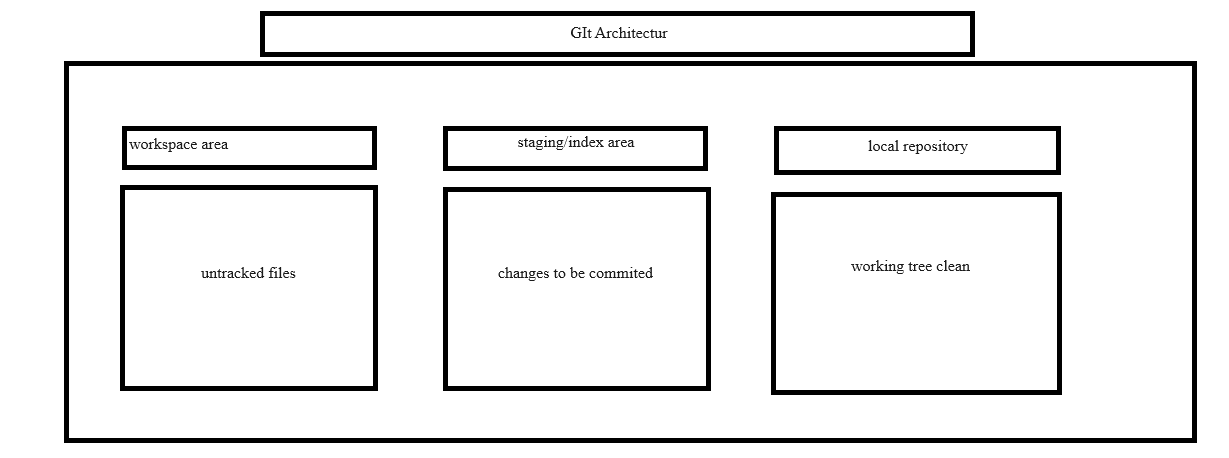
**Linux Command:**

* Create Aws account.
* Launch Linux server.
* Change the path, there are two types of path
  + 1. / path
    2. ~ tilt path
* sudo -i🡪 command is used to the path to (root).
* **ls -la🡪** used to display a list of content of a directory
* **cd🡪**used to change path, from one to another path.
* **mkdir🡪** is used to create new directory
* **rmdir🡪** is used to delete a directory
* **rm -rf🡪** is used to delete a single file
* **rm – rf \* 🡪** is used to delete all the file
* **mkdir d1/d2/d3 🡪** is used to create multiple directory
* **touch🡪** used to create empty file
* **touch <filename> {1 ...50} 🡪** used to create multiple files.
* **vi <filename> or touch 🡪**is used to create a file.
* **esc:i 🡪**is used to insertthe data in to file.
* **esc:wq! 🡪** is used to save the content of the file.
* **cat <filename>🡪**is used to display the content of file
* **esc: se nu 🡪** is used create number is files.
* **esc: 5yy 🡪** is used to select Five lines at a time.
* **esc: p 🡪**is used to paste content
* **esc : dd 🡪** is used delete content
* **esc : u 🡪** is used to undo content
* **esc : / 🡪** is used to search.
* **cp : 🡪**is used to copy the file from one location to another location.
* **mv : 🡪**is used to cut the file from one location to another location.
* var/log ---In Linux, log files are stored in the /var/log directory, which contains subdirectories and files for different system components, services, and applications. For example, application log files are stored in subdirectories of /var/log (user applications server) /etc. directory in Unix-like operating systems. The /etc. directory contains system-wide configuration files and shell scripts that are used to control system operation. Examples of files found in /etc. include /etc./passwd (user account information), (administrator).

**Git Commands:**

**Diagram:**

****

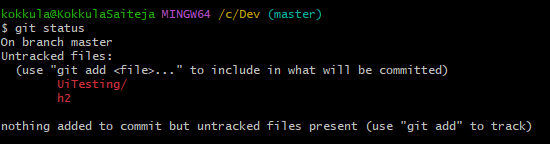
* **git init :** create a new Git repository.(Git now knows that is should watch the folder you initiated it on.

We get (.git files created).

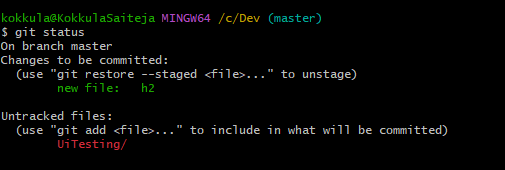
* **Touch <file name> :** used to create a empty file.



* **Git status:** to get a status

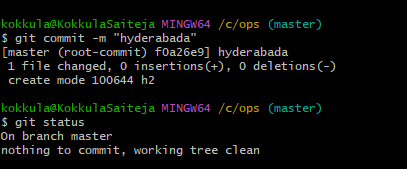


* **Git add <file name>:** staging area

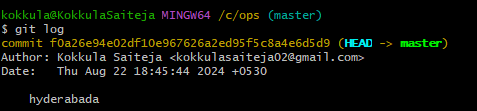
****

* **Git commit – m “normal text”:** local repository

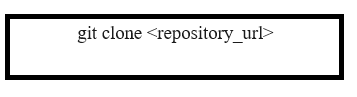
We get working tree clean.

****

* **Git log:** The git log command in Git displays the commit history for a project, starting with the most recent commit

****

* **Git clone :** The git clone command is used to create a copy of an existing Git repository from a remote server to your local machine. The basic syntax is:

****

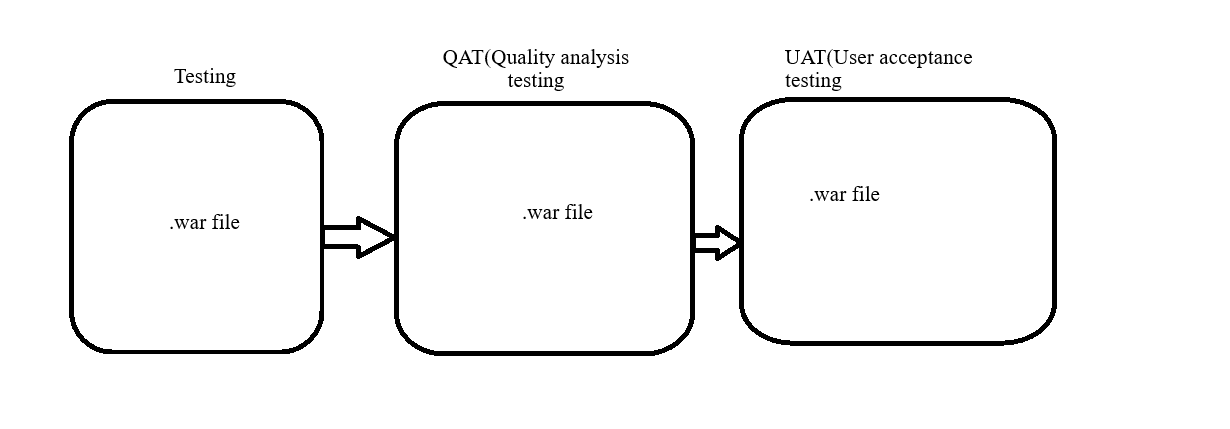
* create some files ,files will be moved local repository, then git push command is move to git to github.
* branch strategy
* Git branch branchname(In git, a branch is a new/separate version of the main repository.
* Branches allow you to work on different parts of a project without impacting the main project. when the work is complete, a branch can be merged with the main project.
* Git checkout branch name
* Ls -la
* main develop -copy to main
* updates/data
* features/data
* switch to updates/data to switch to enter any file then text content
* the file text show will be git any branches. but main doesn't see text content in GitHub
* git push --all origin ---branches will be move to git.
* in GitHub click on pull request select an main to updates/data. Now click merge
* git pull--in git terminal will be GitHub to git

**Maveen**

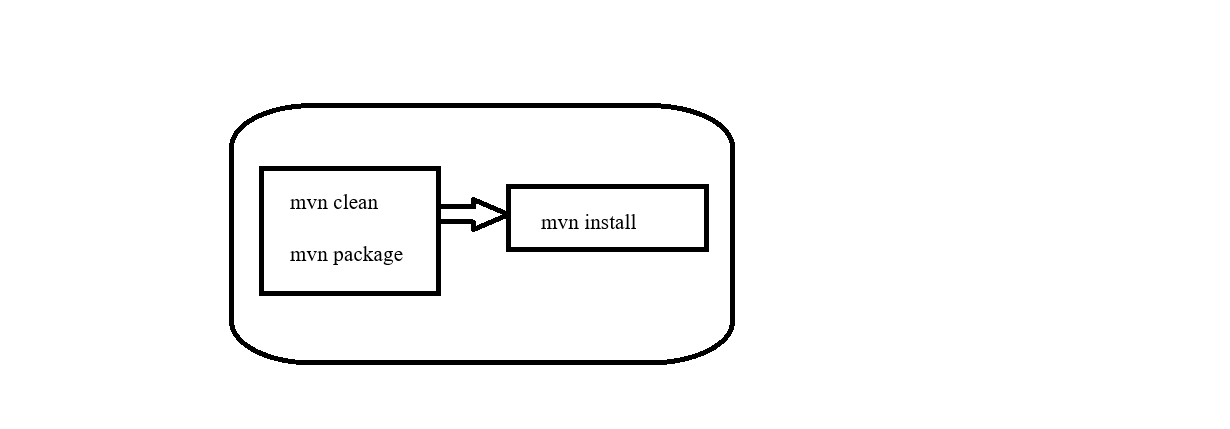
What Is Maven?

Maven is mostly used for the java projects to build web application packages. Maven provides different features which make it easy to build the web-application packages we manage complex projects easily.

Maven is build Tool, used to convert all files into single file that may be war or jar file

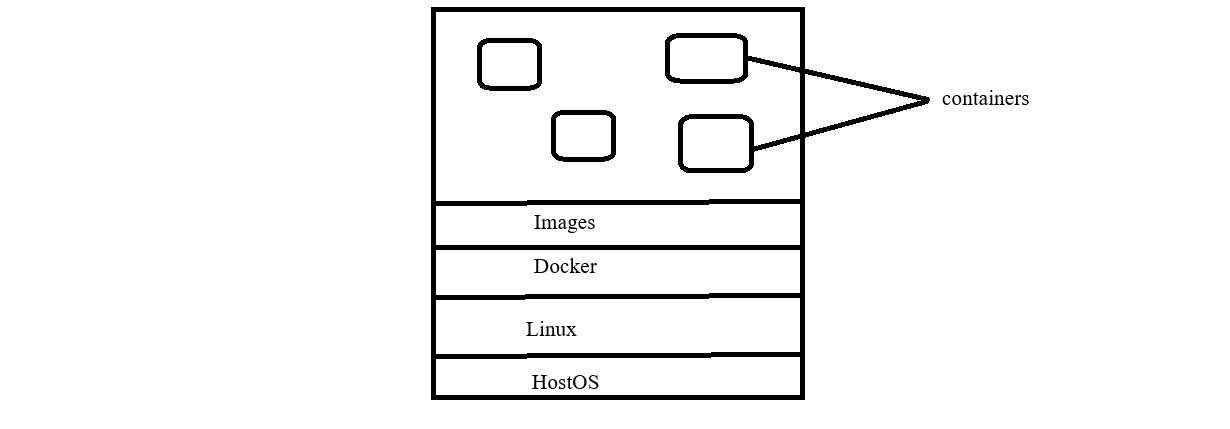


**Maven command or goals:**



**Docker**

Docker is a containerization tool, used to deploy application in the server.



Types of Dockers:

Technology based sever

OS based sever

Database based sever

Graphics based sever

Gamming based sever

**Docker Commands;**

1. login in to aws account
2. Launch ubuntu server
3. 🡪(docker search ubuntu)
4. 🡪(docker pull ubuntu) into local System means (Linux).
5. To check images created or not command is 🡪(docker image).
6. 🡪docker run -itd id of container
7. To check container is created or not command is🡪(docker ps -a).
8. 🡪Docker attach (container id)
9. To come out from container to root(ubuntu) command is (exit).
10. After enter to container, we have to update the Linux command is

🡪(apt-get update)

1. After update in the Linux, we have to install the git and maven.
2. Apt-get install git
3. Apt-get install maven
4. After install the git and maven to check the version is🡪(git –version) 🡪(mvn -version).
5. Exit form the container 🡪 (ctrl p+q), container well be continuous it well be run at background.
6. To Create our own container, form the image. 🡪 (docker commit container id, docker hub image).
7. After create our own container, and push the image in to docker hub.
8. To search our own image the command is 🡪 (docker search image name or sample name of image).

TO DEPLOY A APPLICATION:

First, launch instances of ubuntu.

Push application files🡪(.war files) into git hub

Next, enter in to ubuntu terminal and set path.

Update the ubuntu and install the docker and git, maven

🡪Apt-get install docker.io

🡪Apt-get install git

🡪Apt-get install maven

After install of all the tool in docker

Git clone git hub url of respority

Next, enter into respority and check the list of file by (ls -la).

Create a Dockerfile

Docker run -t “--name”

Docker images

Docker run -itd images name

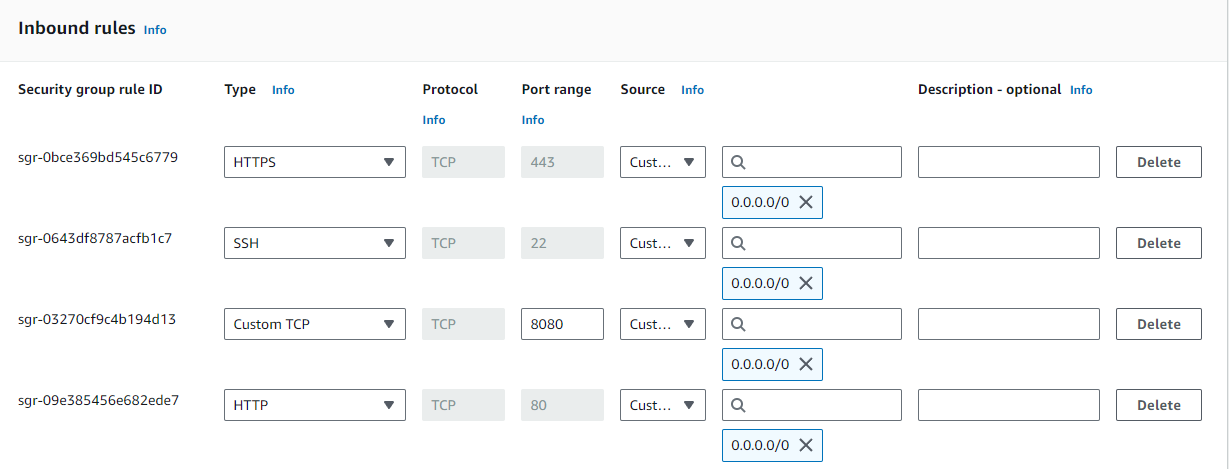
Docker ps -a

docker run -d -p 8080:8080 --name slokamcontainer deptest

Copy (Public IPv4 address) open google paste ip address and (:8080) and paste war file.

The application page is not loaded in google.

Go security🡪 security group and add rules



Example url : http://3.111.32.67:8080/slokam-0.0.1-SNAPSHOT/index.html

🡪docker stop <container id>

🡪docker rm <container id>

* Docker rmi <images name>
* Docker tag images name DockerHud

**Kubernetes**

Kubernetes is also known as 'k8s'. This word comes from the Greek language, which means a pilot or helmsman

Kubernetes is a orchestration tool, It is open source platform designed by google in 2014.

There are four advantages of Kubernetes

1.Auto scaling

2.Security layers are high

3.Memory capacity

4.Time consumption

🡪 **Auto-scaling**: Kubernetes automatically adjusts the number of running containers based on the load, ensuring that applications can handle traffic surges without manual intervention.

Example: Netflix

🡪**Security layers**: Kubernetes offers strong security features, such as role-based access control (RBAC), network policies, and secret management, to keep applications and data secure.

Example:Airbnb

🡪**Memory capacity**: Kubernetes efficiently manages memory and computing resources, allowing applications to scale smoothly without overwhelming the system.

Example: Spotify

🡪**Time-saving**: Kubernetes automates many deployment, scaling, and management tasks, which reduces the time required to manage applications compared to manual processes.

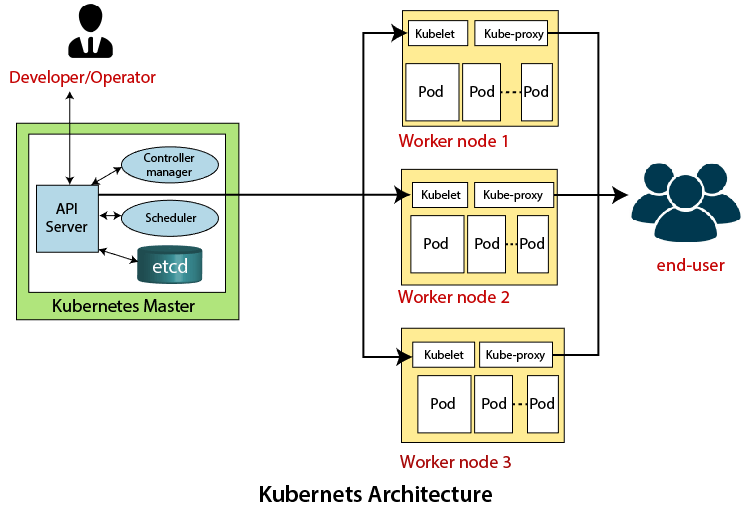
Example: amazon

Checking URL within fraction of the milli second the Application page will be loaded. It can reduce the time.

🡪Docker will take one or two minutes of time to load the page, but Kubernetes is latest version compare to docker.

Example: IRTC is not deployed in the Kubernetes, but it will takes one to load the page and payment issues are occur.

**Kubernetes Architecture:**



**Master Node or Kubernetes Control Plane:**

The master node in a Kubernetes architecture is used to manage the states of a cluster. It is actually an entry point for all types of administrative tasks. In the Kubernetes cluster, more than one master node is present for checking the fault tolerance.

Following are the four different components which exist in the Master node or Kubernetes Control plane:

1. API Server
2. Scheduler
3. Controller Manager
4. ETCD

API Server:

The Kubernetes API server receives the REST commands which are sent by the user. After receiving, it validates the REST requests, process, and then executes them. After the execution of REST commands, the resulting state of a cluster is saved in 'etcd' as a distributed key-value store.

Scheduler:

The scheduler in a master node schedules the tasks to the worker nodes. And, for every worker node, it is used to store the resource usage information.  
In other words, it is a process that is responsible for assigning pods to the available worker nodes.

Controller Manager:

The Controller manager is also known as a controller. It is a daemon that executes in the non-terminating control loops. The controllers in a master node perform a task and manage the state of the cluster. In the Kubernetes, the controller manager executes the various types of controllers for handling the nodes, endpoints, etc.

ETCD:

It is an open-source, simple, distributed key-value storage which is used to store the cluster data. It is a part of a master node which is written in a GO programming language.

Now, we have learned about the functioning and components of a master node; let's see what is the function of a slave/worker node and what are its components.

**Worker/Slave node:**

The Worker node in a Kubernetes is also known as minions. A worker node is a physical machine that executes the applications using pods. It contains all the essential services which allow a user to assign the resources to the scheduled containers.

Following are the different components which are presents in the Worker or slave node.

Kubelet:

This component is an agent service that executes on each worker node in a cluster. It ensures that the pods and their containers are running smoothly. Every kubelet in each worker node communicates with the master node. It also starts, stops, and maintains the containers which are organized into pods directly by the master node.

Kube-proxy:

It is a proxy service of Kubernetes, which is executed simply on each worker node in the cluster. The main aim of this component is request forwarding. Each node interacts with the Kubernetes services through Kube-proxy.

Pods:

A pod is a combination of one or more containers which logically execute together on nodes. One worker node can easily execute multiple pods.

(or)

**Master Node:**

The master node is the brain of the Kubernetes system. It controls and manages the entire cluster, including all the worker nodes.

API Server:

This is like the receptionist of Kubernetes. It receives commands from the user, checks if they are valid, and then executes them.

After execution, it saves the current state of the system in a database called etcd.

Scheduler:

Think of the scheduler as the manager who decides which worker gets which task.

It assigns tasks (pods) to the available worker nodes based on resource availability.

Controller Manager:

This is like a supervisor. It constantly checks if everything is running as expected.

If something is wrong (like a failed pod), it makes sure to take the necessary action to fix it.

ETCD:

This is the memory of the Kubernetes cluster.

It stores all the important information about the state of the system in a key-value format.

**Worker (Slave) Node:**

The worker nodes are like the employees of the system. They run the actual applications.

Kubelet:

This is like a foreman on each worker node.

It ensures that the applications (pods) are running properly, and it communicates with the master node to receive instructions.

Kube-proxy:

This is the network helper.

It forwards requests and helps in the communication between different parts of the system.

Pods:

A pod is a group of one or more containers (small virtualized environments) that run applications.

A worker node can run multiple pods at once.

Summary:

Master Node: Think of it as the brain, controlling everything.

Worker Node: These are the muscles that do the actual work of running applications.

Command:

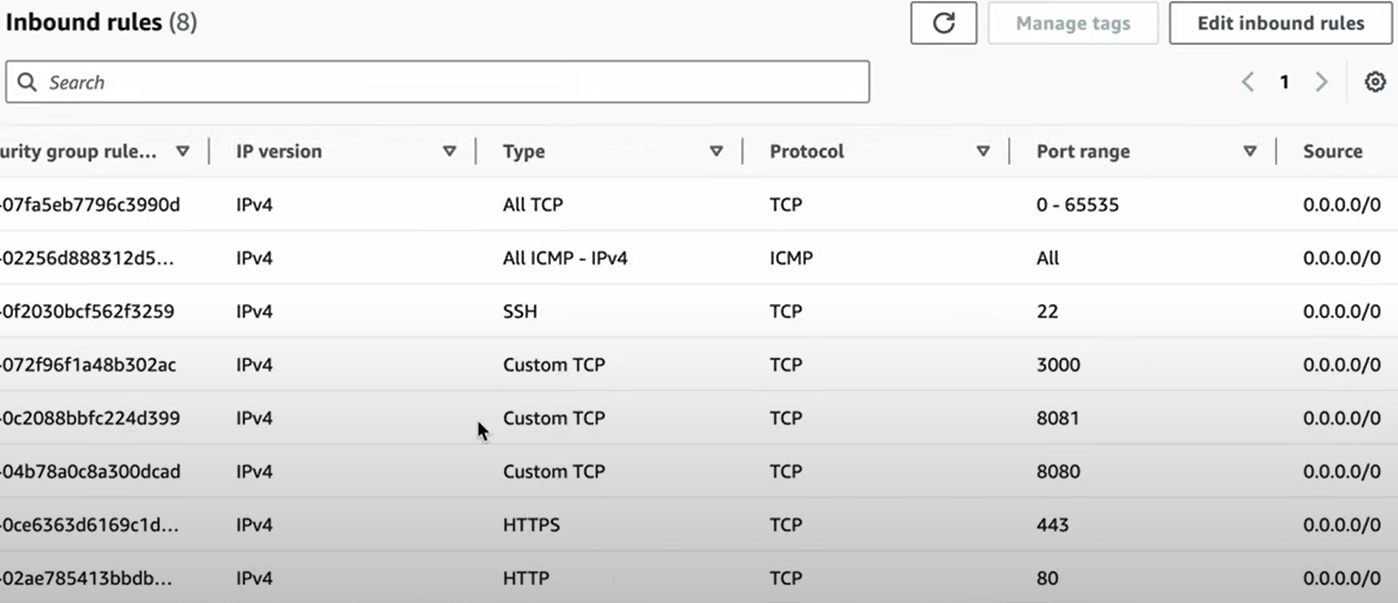
🡪For installation and setup of Kubernetes:

🡪 <https://github.com/yeshwanthlm/Kubeadm-Installation-Guide>

🡪Launch two ec2 instance

* **0One is master node**
* **One is worker node**

**🡪**First, select the master node and go to security group and edit inbounded rules.



**🡪**After launch instanceexecute following command proved in above link.

🡪After installation of Kubernetes

* To check any pods and nodes, service execute or not.

**🡪 Command**

**🡪kubectl get nodes**

**🡪kubectl get pods**

**🡪kubect1 get svc**

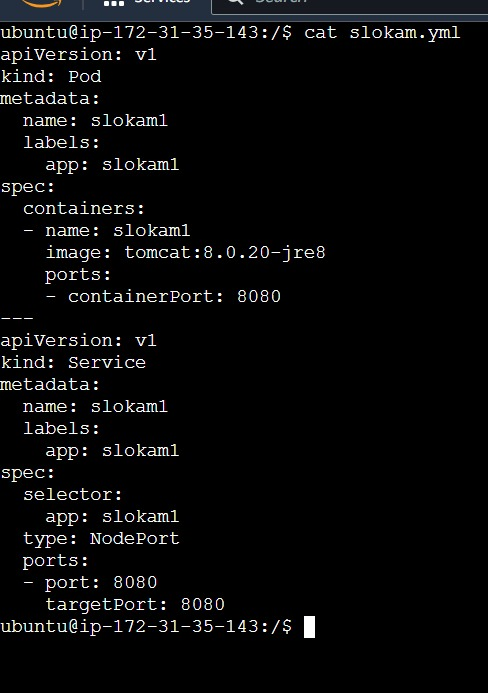
**🡪**After completion checking

🡪check any (.yml) file created or not with command (ls -la)

🡪To create file with extension (. yml)

* Create in master node.

**Ex: vi sample.yml**

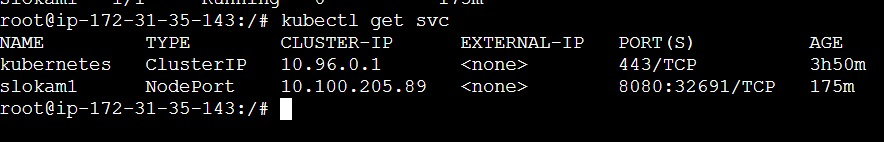
****

**🡪** To execute file the command is **🡪(kubectl apply -f file name with extension)**

**🡪**To check pods and service are create the command is:

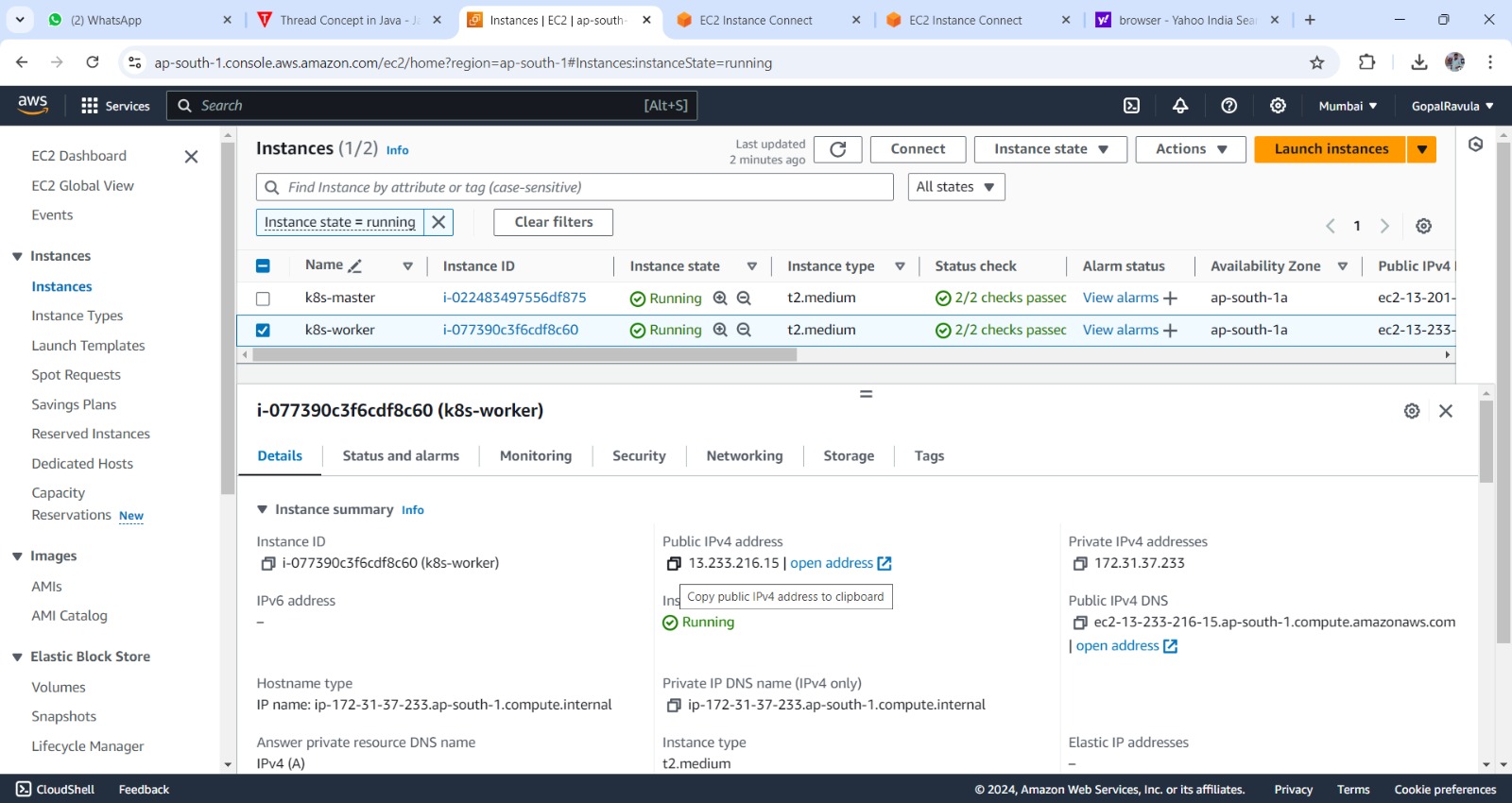
🡪**kubectl get pods**

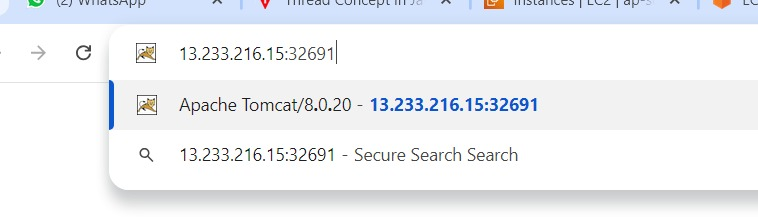
**🡪 kubectl get svc**

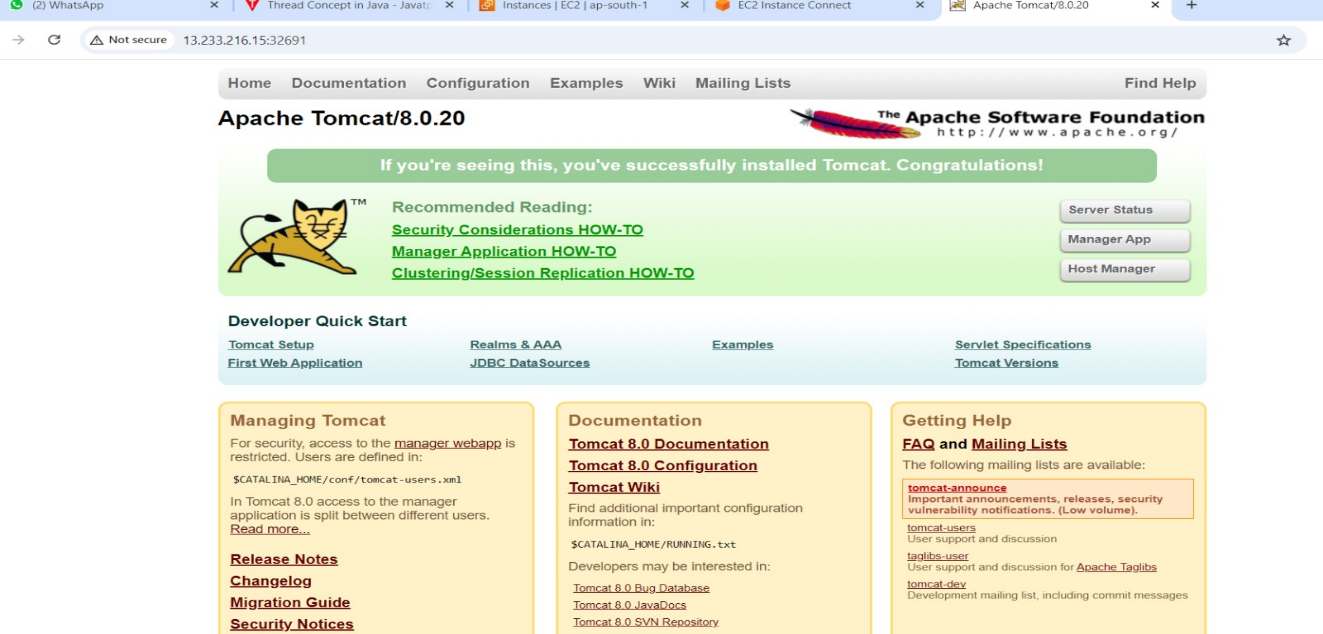
****

**🡪**To check server is run or not

**🡪Go to instance and select the worker node Ip public address and paste in the google chrome browser**



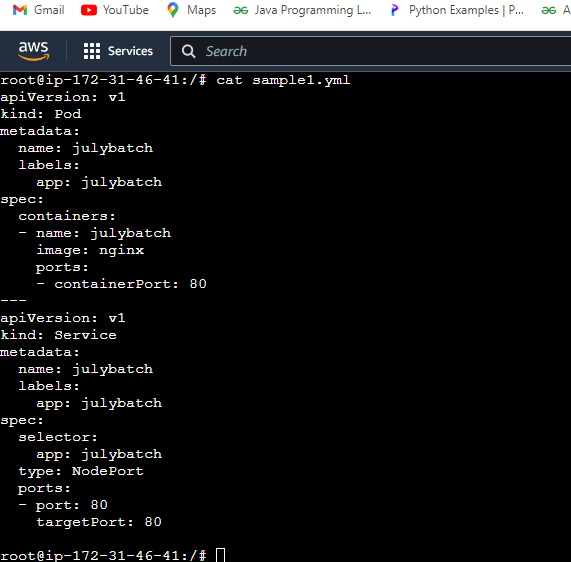
****

****

🡪To create file with extension (. yml)

* Create in master node (for nginx application (or) server).

**Ex: vi sample1.yml**

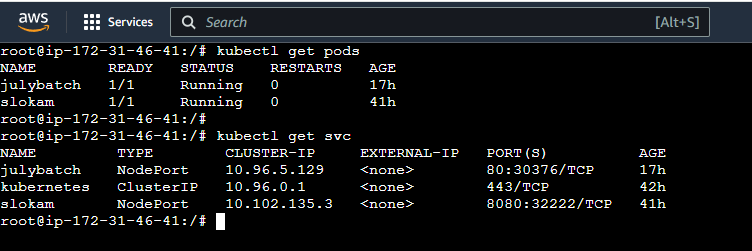
****

🡪 To execute file the command is 🡪(kubectl apply -f file name with extension)

🡪To check pods and service are create the command is:

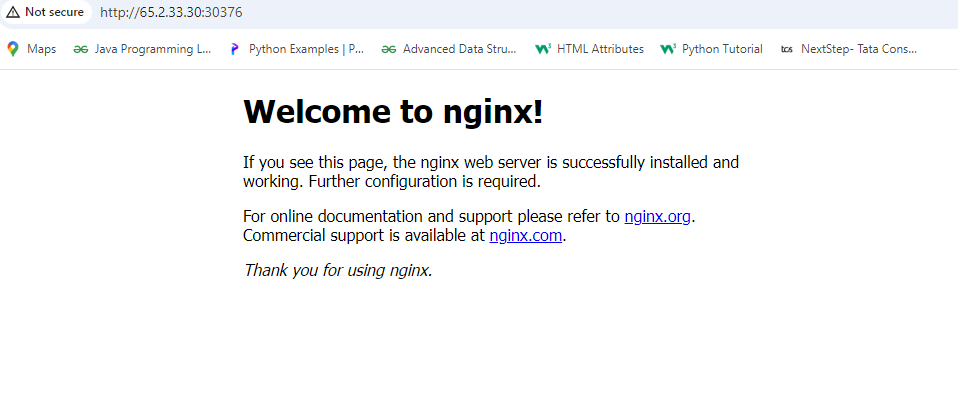
**🡪kubectl get pods**

**🡪 kubectl get svc**

****

**🡪**To check server is run or not

**🡪Go to instance and select the worker node ip public address and paste in the google chrome browser**

****

**🡪To delete pods the command is:**

**🡪kubectl delete pod <pod-name>**

**🡪To delete svc the command is:**

**🡪kubectl delete svc <service-name>**

**Auto scaling:**

Auto-scaling in Kubernetes means automatically adjusting the number of pods in your application based on how much work they are doing. If your app needs more resources because of high traffic or load, Kubernetes will add more pods. If the load decreases, it reduces the number of pods to save resources.

**Here's a simple explanation:**

1. **Deployment** or **Replica Set** manages how many pods you want running. For example, if you set it to 3 pods, Kubernetes will make sure that there are always 3 running.
2. **Pod Deletion**: When you delete a pod (using kubectl delete pod <pod-name>), Kubernetes notices that one pod is missing.
3. **Self-Healing**: Kubernetes immediately starts a new pod to replace the one you deleted. This ensures that the number of running pods matches what you wanted (the **desired state**).
4. **New Pod**: The new pod will have the same configuration (like the app version, environment, etc.), but it will have a **new name** and a **new IP address** because pods in Kubernetes are temporary.

**Key Points:**

🡪**Self-Healing:** Kubernetes will automatically create new pods to replace any that are accidentally or intentionally deleted.

🡪**New Pod:** The new pod will have a different name and IP address, but it will be exactly the same in function and configuration as the deleted one.

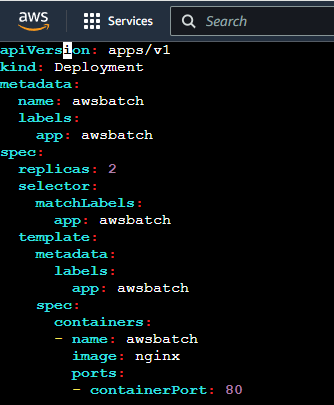
🡪**No Manual Intervention:** You don’t need to do anything after deleting the pod. Kubernetes takes care of it!

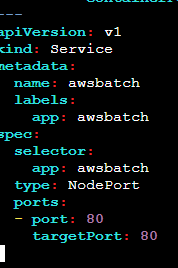
**Create a Deployment**

A Deployment in Kubernetes manages a set of pods. Here’s how to create one

Create a Deployment YAML file:

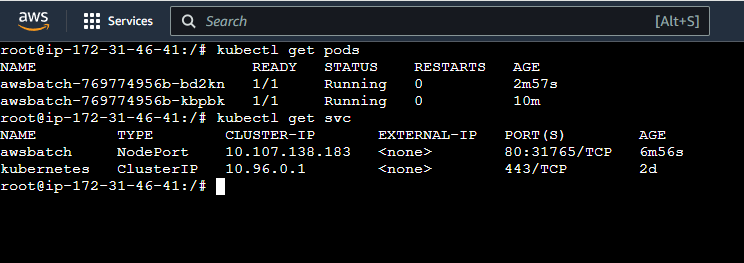
Save the following YAML content in a file called. (samplepod.yml)





**2.Apply the Deployment to your Cluster:**

Use kubectl apply to create the deployment.



**Deployment of Kubernetes**