K8S Run Book

It is an Orchestrator. Means this

Orchestration means managing the container in proper way. Just like how the above pic shows, he controls the entire musicians. Just like that our K8S will control entire container systems.

So the Technical Definition is :

An Orchestrator is a back-end system that deploys and manages applications.

This means :

It helps you

1. Deploy your application
2. Perform updates
3. Rollbacks and more…. More .. more

If your Orchestrator good, means it does everything automatically. Isn’t it ?

Now.. little bit about history of Kubernetes.

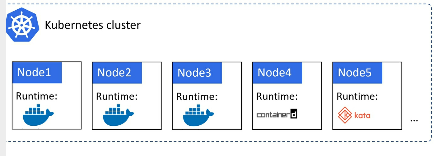
Where did Kubernetes come from ..

1. It came out of Google
2. Later it was open-sources in and handed over to CNCF – Cloud Native Computing Foundation
3. It’s written in Go Language.

Kubernetes and Docker

Both goes hand in hand.. Means Develop your applications with Docker and use K8S to orchestrate.

Take a look at this diagram.



All the nodes are running some sort of Docker container, but orchestration is taken care by K8S

K8S vs Docker Swarm

In 2016, there was a big war between orchestrators. Finally K8S won and became super famous. It’s true that Docker swarm exist but their market share is very less.

Kubernetes – What’s in the name?

Koo-ber-net-eez - it came from Greek Word.. Meaning is “HelmsMan”

Helmsman look like this



Hope you got some idea now what I am trying to say.

The Person who steers the ship.. who rides the ship, who drives the ship etc.,

The theme is reflected in the logo.

Kubernetes is shortened to K8S

KUBERNETES

K8S

8 is replaced with those 8 characters in between K & S

K8S Important Terminology

K8S does two things:

1. Clustering
2. Orchestrating

Clustering means – group of nodes, Control plane, Scheduler for assigning work to nodes, state is recorded in a persistent store. Nodes are where application services run…

Ok if you don’t understand what I said above, don’t worry. This is just a high level overview. We will talk about it soon.

Let’s talk about Football now.

Football – group of individuals. Each player has different roles.

Defend

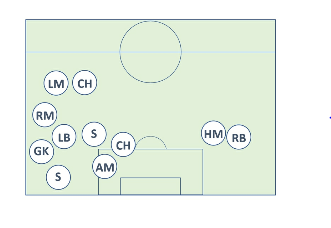
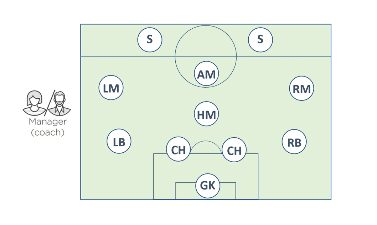
Attack

Passing

Tackle

Shoot

Along with them we will have Coach. Let’s compare the two figures



The 1st figure doesn’t have a Coach and 2nd have a Coach.

1st figure is not organized because no coach is available, but 2nd is properly organized as we have a coach

So what I want to say is . You need a Coach.. means you need an Orchestrator. Means you need K8S

So Let’s see the high-level plan what exactly the K8S does :

1. Start with any application
2. Package it up
3. Give it to the K8S Cluster
4. Now Cluster is made up of one or more masters and a group of nodes

Master – sometimes we call it as heads or head nodes. This is in charge of cluster

This means, master make scheduling decisions, perform mornitoring etc., sometimes we call

It as Control Plane.

Nodes – Here your application will run. We sometimes call it as Data Plane.

They have to report back the status to master and also keep checking for new work given by

Master

Ok.. Enough.. To Run applications on K8S Cluster, you should follow this high level plan

1. Write the application in your favourite language
2. Package in its own container
3. Wrap each container in its own pod
4. Deploy pods to the cluster via higher-level controllers such as Deployments

As you know Deployments offer Scalability and rolling Updates

Now K8S likes to manage applications DECLARATIVELY

This is written in the format of YAML files. POST these files to K8S , and that’s it.. it will deploy

Master and Nodes

A K8S cluster is made up of MASTERS & NODES

MASTERS : (CONTROL PLANE)

Generally 3 or 5 masters should be running in your Production environment. Why ? Because if one master fails, other masters can take over the cluster

What are the components making up these MASTERS?

1. API Server

It is like Our Majestic Railway Station. Every train should go via Majestic. Likewise, all communication between all components goes through API server.

We post YAML files or sometimes we call these YAML files as MANIFESTS, which we will POST it to the API server.

You can think of API server as Brain of the Cluster.

1. Cluster Store

This is the Heart of the Cluster. This stores the state of your cluster configuration. This is very important component. If no cluster store means, no cluster

1. The Controller Manager

This is Controller of Controllers. This implement multiple Control Loops which watch the cluster and respond to events

Sometimes we call it as Background watch-loop that is constantly watching the API server for changes.

Aim of the Controller Manager is : To ensure the Current State of the Cluster matches the Desired state

Logic :

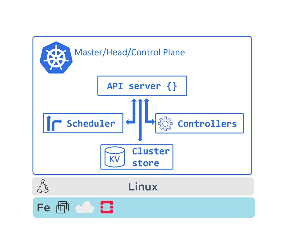
1. Obtain Desired State
2. Observe Current State
3. Determine Differences
4. Reconcile Differences

Control Loop = Watch Loop = Reconcilation Loop … all are same

4.The Scheduler

Scheduler watches for new work tasks and assigns them to appropriate healthy nodes

Control Plane Summary:



The above diagram will tell you about what we discussed earlier

Nodes:

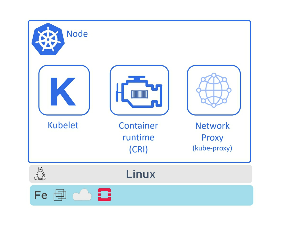
Nodes are the workers of K8S Cluster.

At a High Level, they do 3 things

1. Keep Watching the API Server for new tasks
2. Execute new tasks
3. Report back to Control Plane

Sometimes people call these nodes as “Minions”.

Let’s see the 3 Components of Nodes:



Kubelet:

* It is the star of the show on every node. It’s like Hero of the Film
* It’s the main K8S Agent and it runs on every node in the Cluster
* Kubelet responsibility is “Node Registration Process”
* Major job of Kubelet is to keep watching the API Server for new work assignments
* Any time it sees one, it will execute the task and report it back to Control Plane.

Container Runtime:

* To perform container related activities, you need a Container Runtime. We are using here Docker as ContainerRuntime Interface
* Things like Pulling Images, Starting and Stopping Containers etc.,

Kube-proxy:

* The last piece we are talking about is “kube-proxy”. This runs on every node in the cluster.
* This is responsible for Local Networking.
* For Ex: it will make sure each node gets its own unique IP address

Packaging Apps

Rule : In order to run an Application inside a Kubernetes Cluster, it needs to follow below steps

1. Packaged as a Container
2. Wrapped in a Pod
3. Deployed via a Declarative manifest file

Ok. It goes like this…

1. You Write an application in your favourite language
2. After that build it into an image
3. Store it in a Registry

Till the above 3 steps, you guys are already familiar as we says “Dockerizing” or “Containerizing”

Next,

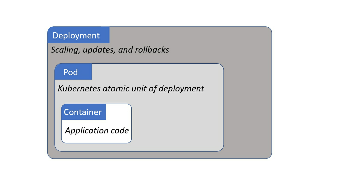
1. We define the K8S Pod to run the Containerized service in
2. Pod is just a wrapper that allows container to run on a K8S cluster

K8S offers several ways of deploying and managing Pods.

The most common is “Deployment” which offers scalability, self-healing and rolling updates.

We define them in YAML file that specifies things like which image to use, how many replicas to deploy.

Look the below figure, shows application code packaged as a container, running inside a Pod, managed by a Deployment



Once everything is defined in Deployment YAML file, we POST it to the cluster as the desired state of the application and K8S will implement it

The Declarative Model and Desired State

This concept should be very clear and you need to good in understanding this.

Declarative model works like this:

1. Declare the Desired state of the application in a MANIFEST or YAML file
2. POST it to K8S API Server
3. K8S stores this in CLUSTER STORE as Application’s DESIRED STATE
4. K8S implements the desired state of the cluster
5. K8S implements watch loops to make sure the current state of the application doesn’t vary from the DESIRED STATE

Tips:

Manifest files are written in SIMPLE YAML file

This YAML file will inform K8S how our application should look like

We call this as Desired State.

This YAML file will have details like

1. Which image to use ?
2. How many replicas ?
3. Which network port to listen ?
4. How to perform updates?

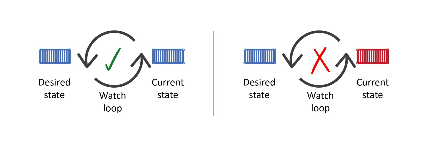
Once we have created the manifest, we POST it to API Server.

K8S implements the manifest file into the cluster.

Records the desired state in the CLUSTER STORE

Finally, K8S utilizes background reconciliation loops that constantly monitor state of the cluster.

If the Current state of the cluster varies from the Desired State, then K8S will jump in and resolve the issue.



Hope you remember the above diagram. ☺

Let’s consider one example:

Assume we have an app with the DESIRED STATE of 10 replicas of a web front-end Pod.

10 replicas are running on around 5 Nodes. It means each node has 2 replicas.

Consider one Node has failed, now you have only 8 replicas are running.

So the current state will be reduced to 8 replicas, but the DESIRED STATE is 10.

This will be monitored by our Looping Team. i.e, Background Loop. This will be identified and K8S will launch 2 more replicas to match with the DESIRED state.

Pods

In VMware world, it is Virtual Machine

In Docker world, it is Containers

Well.. In K8S world, it is Pods

Always remember the below points:

YOU CANNOT RUN A CONTAINER DIRECTLY ON K8S CLUSTER.

CONTAINERS MUST ALWAYS RUN INSIDE OF PODS

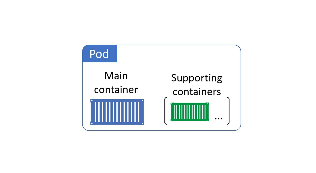
Pods and Containers

Ok. The Term Pod comes from a “Pod of Whales”

In English language, we call “Group of Whales” as “Pod of Whales”

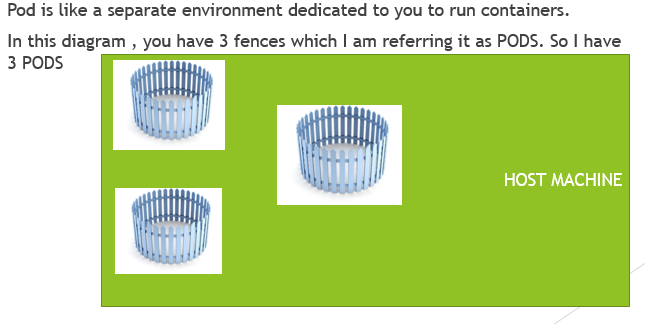
If you see the Docker Logo – it is a whale. It makes sense that we call a group of containers a Pod

One Pod can be running a single container or multiple containers.



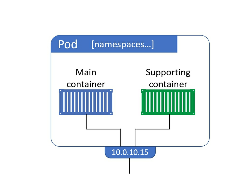
Pod Anatomy:

1. Pod is like a ring-fenced environment.
2. Haha. You didn’t understand?
3. Well. You will be getting a separate section for One Pod inside the Host machine.
4. Once you get the separate section, you can run multiple containers inside



All containers in the Same Pod will share the same IP address

See the below diagram

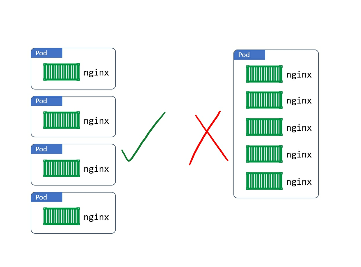


Scaling your Pods

If you want to scale your application, means you need to add or removing the PODS.

You shouldn’t add or remove the containers in the PODS.

Please see the below diagram, if you want to scaleup, you have to increase the number of Pods.



Pod LifeCycle

Pods are like Humans. It will live and die

If it died unexpectedly, we will not bring it back

Instead, K8S starts a new Pod with a new IP address

Services

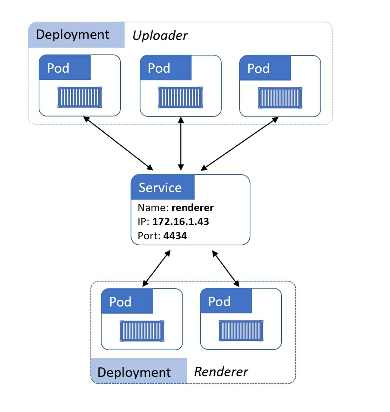
We have learnt Pods are mortal. i.e, it will die. Then it will launch a new Pod with a new IP address.

Now I want to say that Pods are unreliable. i.e, it will re-launch with a new IP address and it’s not stable.

For ex : One apache webserver Pod is running with an ip 192.168.45.33. Suddenly it crashed and re-launched a different Pod with a new IP 191.154.33.48. Now I have already hard-coded the earlier ip into my database. Now that connection would have gone and my entire application will not be working.

In order to avoid this, we are introducing Services.

Service Provides reliable networking for a set of Pods



See the above diagram:

UPloader micro-services is talking to Renderer micro-services through K8S Service

The K8S Service is providing a reliable name and IP, load-balancing requests to the two renderer pods behind it.

Services contains:

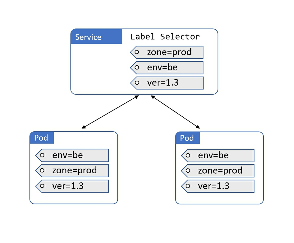
1. Stable DNS name
2. IP address
3. Port

So Pods can come and Go. But Services remain stable.

So I have put this service IP address hard-coded in my DB ( earlier example ), I would have never faced any issue.

Connecting Pods to Services

1. Services use LABELS & LABEL SELECTORS
2. Let me show you the diagram and you will understand without any further explanation



Now see the Label Selector is inside the Service and it has mentioned:

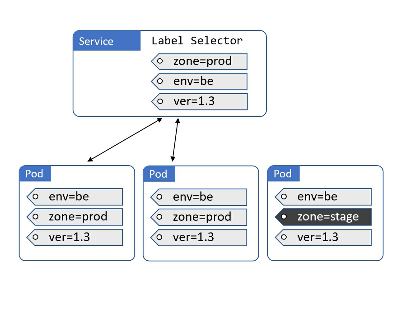
Zone-prod

Env-be

Ver-1.3

Now under pod, if the above labels are matched, then the traffic will be flowed into those pods

Now see the next diagram :



The 1st and 2nd Pod matches the label selector of the service, hence traffic is routed to those 2 pods. But the 3rd Pod doesn’t match the label selector “zone=stage”

Hope you are getting the idea …

Services only send traffic to HEALTHY NODES

INSTALLING K8S

Here, we will look at few ways of installing K8S.

I would like to tell you about Hosted K8S services as well which is provided by cloud providers like Azure Kubernetes Service ( AKS ), Google Kubernetes Engine ( GKE )

More companies are using Hosted Kubernetes services as they don’t want the head-ache to manage the clusters.

We have various ways of installing K8S.

1. Play with Kubernetes ( PWK )

This is the learning website where you can do for the testing purpose, to learn kubernetes commands.

URL : <https://labs.play-with-k8s.com/>

You can login using github login or dockerhub login

1. Docker Desktop

You can download Docker Desktop if you are on windows OR mac and can use for testing purpose

URL : <https://www.docker.com/products/docker-desktop>

1. Minikube

You can use MiniKube application to learn K8S locally, for development purposes

URL : <https://kubernetes.io/docs/tasks/tools/install-minikube/>

1. Google Kubernetes Engine

For Production, better to utilize this hosted service

URL : cloud.google.com

1. Kops

For Production, We can use this tool called “Kops” and install cluster using this tool on AWS

1. Kubeadm

Manual installation

1. Kubectl

This is K8S Command Line Tool

Ok. If you google various ways of installing kubernetes. You will see lot of other ways. I have mentioned only few 7 ways of installing K8S which I feel very important.

And remember, hosted K8S like AKS, EKS, GKE where management of control plane is taken care by themselves.

AKS – Azure K8S Service

EKS – AWS Elastic Kubernetes Service

GKE – Google Kubernetes Engine

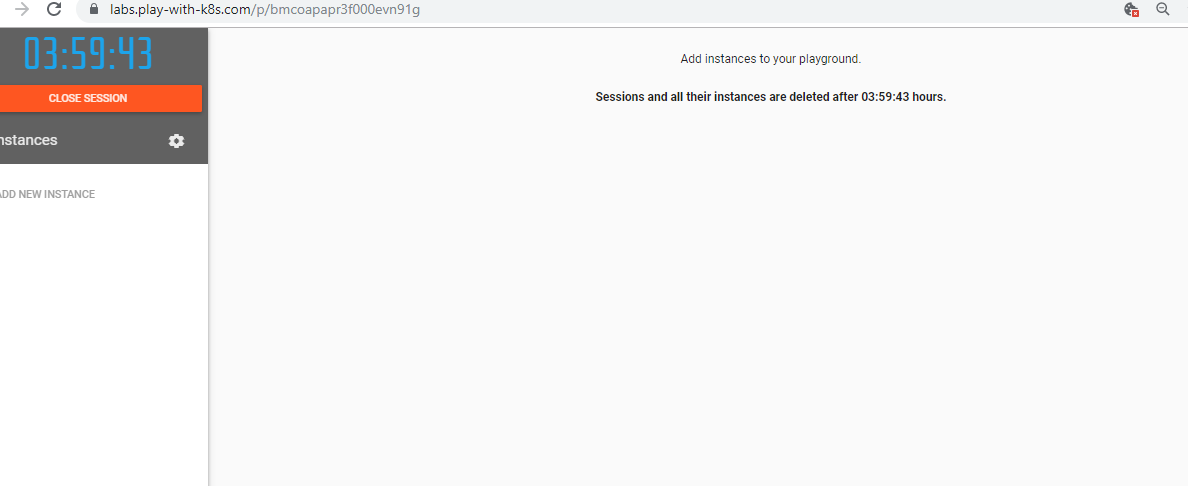
Here I am not going to explain you all the 7 ways of installing K8S. I will show you with “Play with K8S” , GKE, Kops, Kubeadm

1. Play with Kubernetes

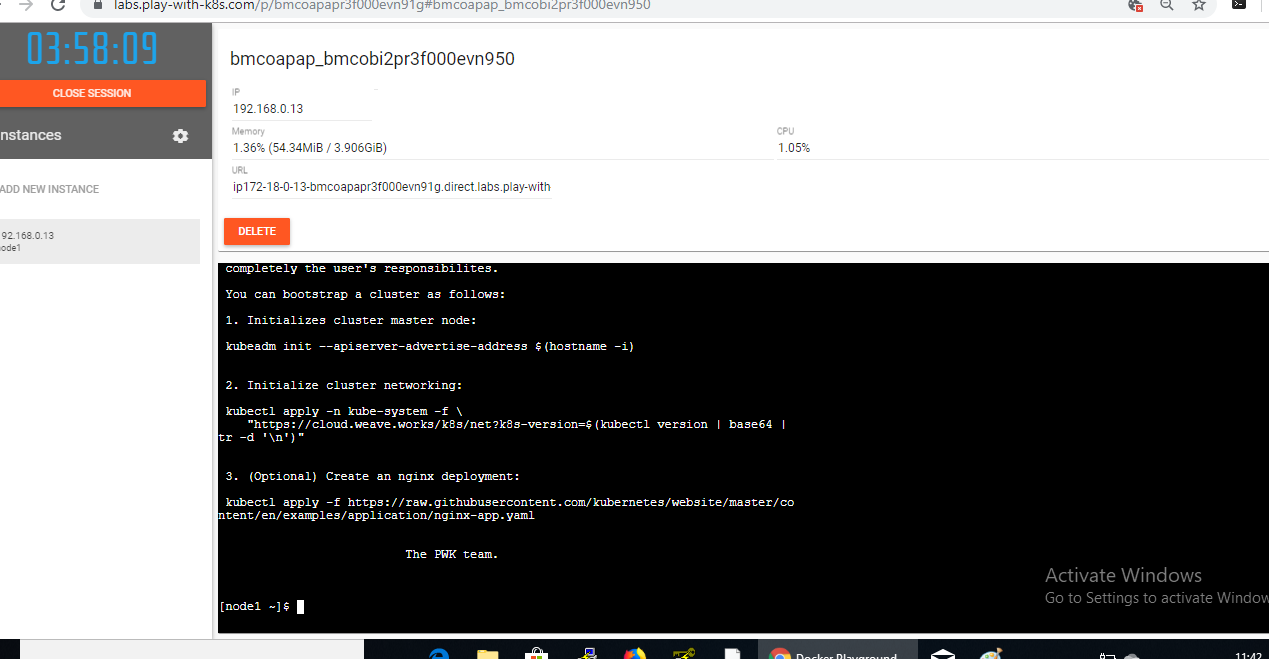
Ok. This is free and you don’t need to install anything. It’s ready-made. If you want to learn the commands quickly and for testing purpose, you can use this.

Go to <https://labs.play-with-k8s.com/>

You will get the below screen



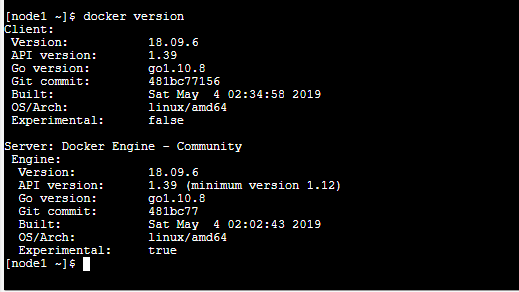
Click on Add New Instance on the left side, you will get the below screen



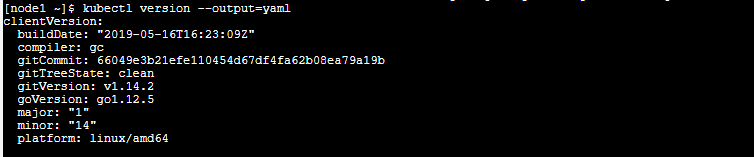
Just analyse what is the above screen says. You will get some 3 steps.

Try some commands like

docker version



Kubectl version –output=yaml



As the output shows, node already has Docker and kubectl (K8S client) pre-installed. Other tools like kubeadm are also pre-installed.

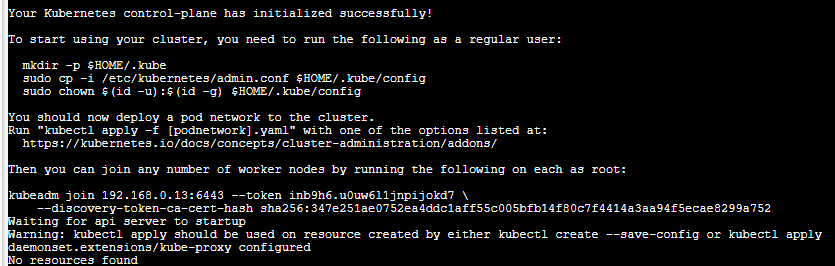
Ok.. Now let’s go deep.

Use the kubeadm command to initialize a new cluster:

1. Run the below command in node1 now



You will get the output that K8S Control-plane has initialized successfully.



If you see the above screenshot, it’s asking you to run some extra commands as regular user. Well since we are using PWK, these things are already done.

Ok. So what we have done so far. We have initialized the cluster. Run the below command.

kubectl get nodes



Now see the above diagram, it shows One Node cluster which acts as master.

But the STATUS shows NotReady. Why ?

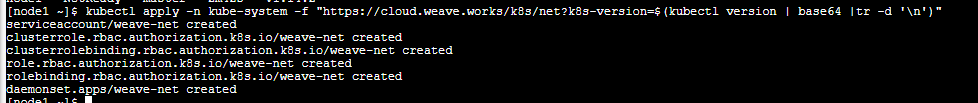
This is because, we haven’t configured the Pod Network yet.

Go back to Page 16 and see the screenshot, we had completed the 1st step using kubeadm and successfully created the cluster.

Now if you see the page 16 screenshot, it asks us to create Pod Networking. Since we haven’t created Pod network yet, that’s why Status shows Not Ready.

1. Let’s try to create the Pod network.

kubectl apply -n kube-system -f [https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 |tr -d '\n')](https://cloud.weave.works/k8s/net?k8s-version=$(kubectl%20version%20|%20base64%20|tr%20-d%20'\n'))



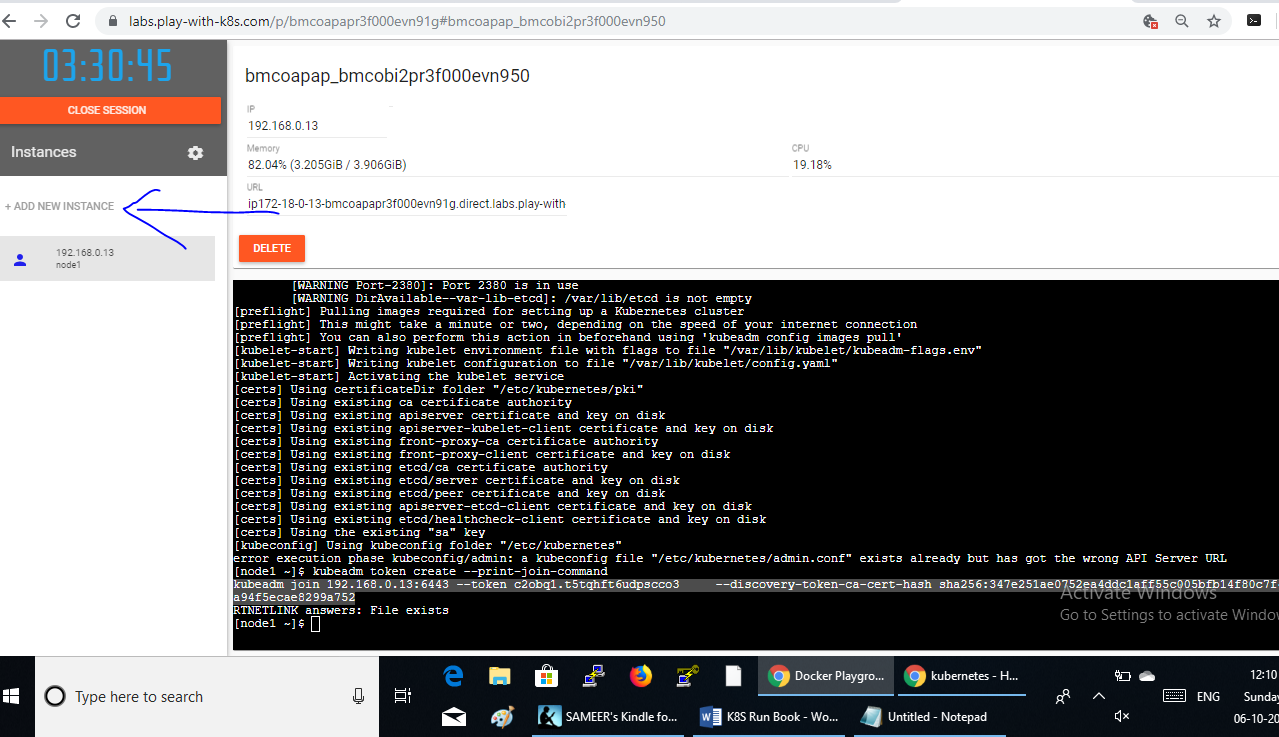
Now Pod network has been created.

1. Now verify the nodes



Now we have made the control plane by initializing the cluster and created Pod network. Now time to add some worker nodes.

Well, in order to join the worker node, click on Add New Instance



So you will be in Node 2 now. What are we going to do in node 2 ?

We are going to run some join command which will make node 2 as worker and node1 will be our master.

Go to Node1 and findout your join command syntax. Run the below command in Node1.

kubeadm token create --print-join-command

the above command will give you the join command.



Now if you see the above diagram, you will get the join command.

Copy this join command and execute it in Node 2.

In Node2: Once you execute the join command, you will get the message like below



Now Switch back to Node1 and run the below command

Kubectl get nodes



So you get 2 nodes and both in READY status.

Congratulations, You have made a fully working K8S cluster that can use as a test lab.

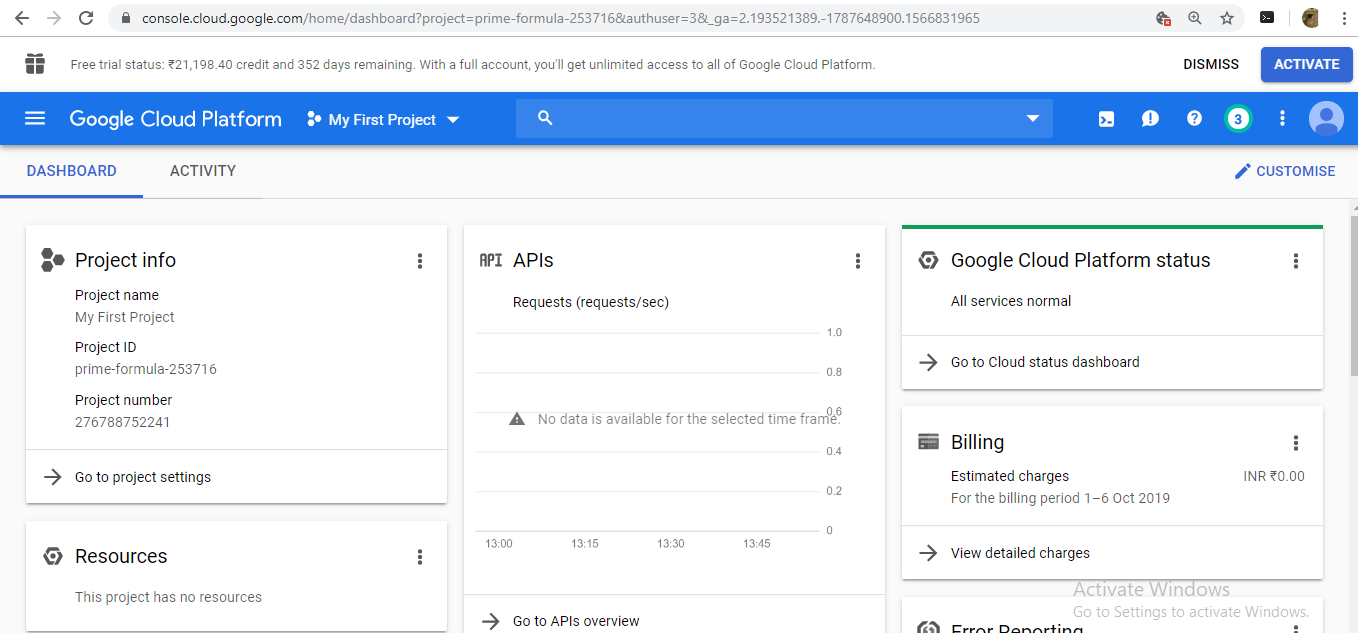
Next… let’s try to build the cluster in Google Cloud.

Google Kubernetes Engine – GKE

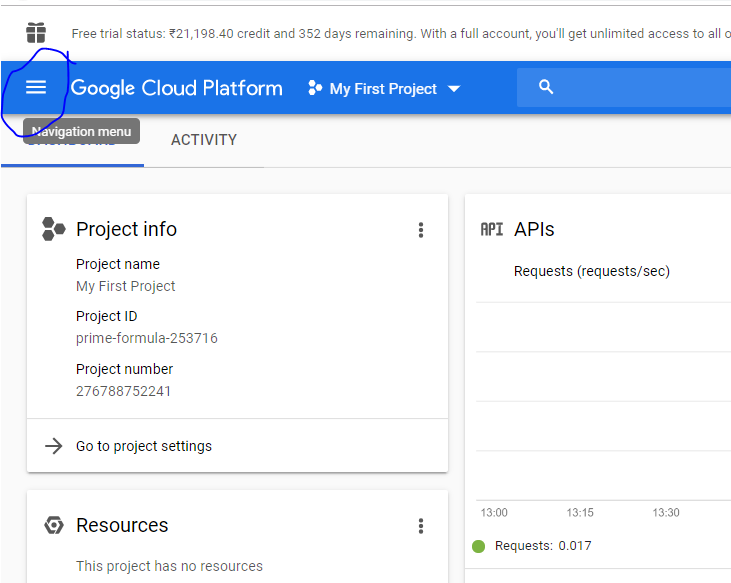
This is one of the Hosted K8S where we don’t need to manage the masters/control plane

Most of the companies would go for Hosted K8S

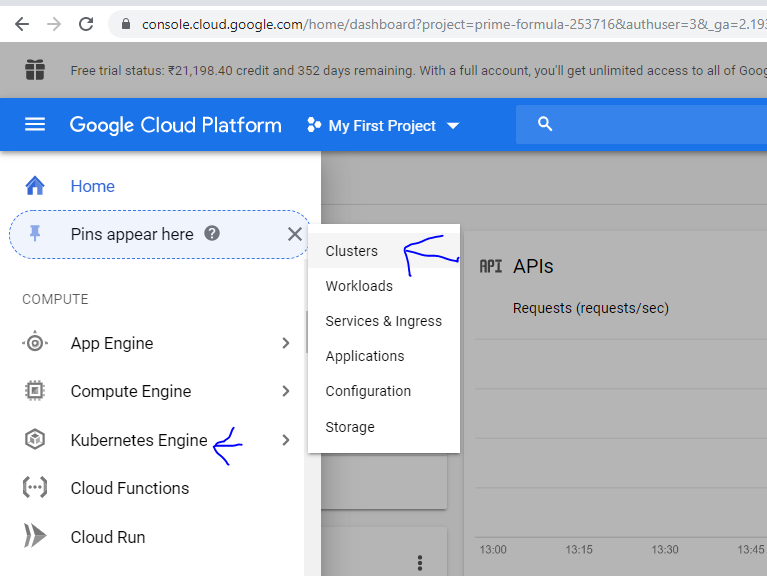
1. You should create a Google Cloud account in <https://console.cloud.google.com/>
2. You will get the below Dashboard screen



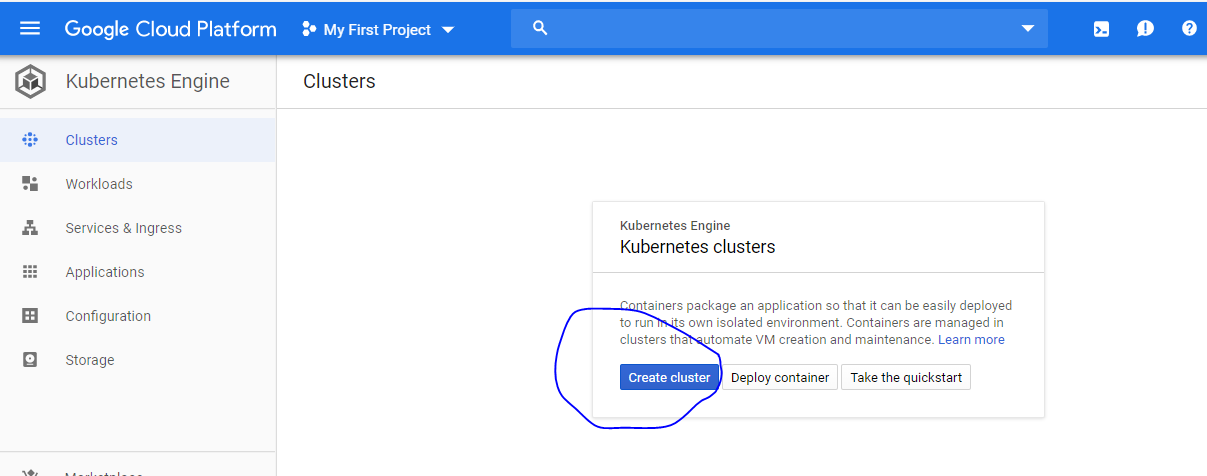
1. Click on the burger menu



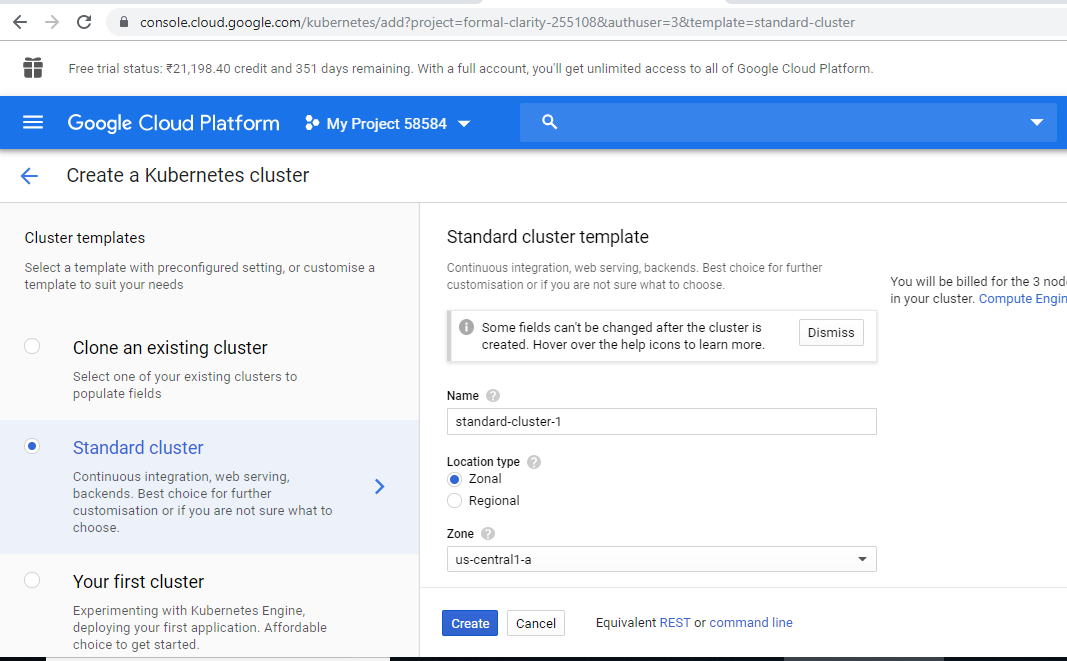
1. And choose, Kubernetes Enginer -> Cluster



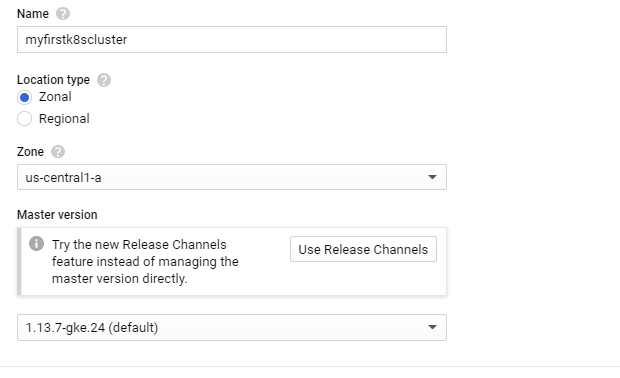
Now click on “Create Cluster”



Once you clicked “Create Cluster”, you will get the below page, ON the left side, you will have lot of options. For this course, you can choose “Standard Cluster”. Other options are for different purposes



Ok. Now you have to fill up few fields in this. Let’s see what are those..



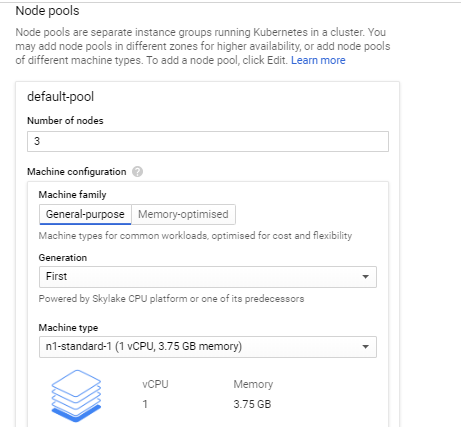
I have updated my cluster name as “myfirstk8scluster”. You can put whatever name you want.

Location Type: I am not going to teach you about Google Cloud here. If you already have basic knowledge on AWS or Availability Zone or Regions then you can easily understand what is this. Try choosing zonal and see what the drop down changes. Try choosing Regional and see what happens. In this example, am choosing Zonal as default and Zone as “us-central1-a” as default.

If you are launching your application from Mumbai or anywhere in Europe, you many have to choose the Zone something else from the drop down.

Master Version: K8S Version default will be installed on master.

If you scroll down further, you will get the below page

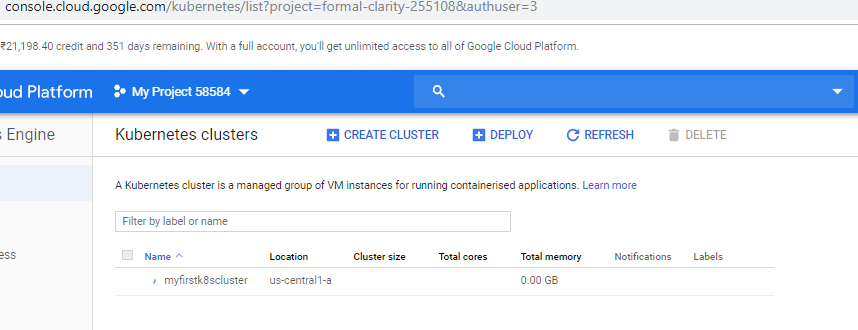


On the above screen, it has mentioned 3 nodes, means.. Total 3 machines ( 1 is master & 2 are nodes ).

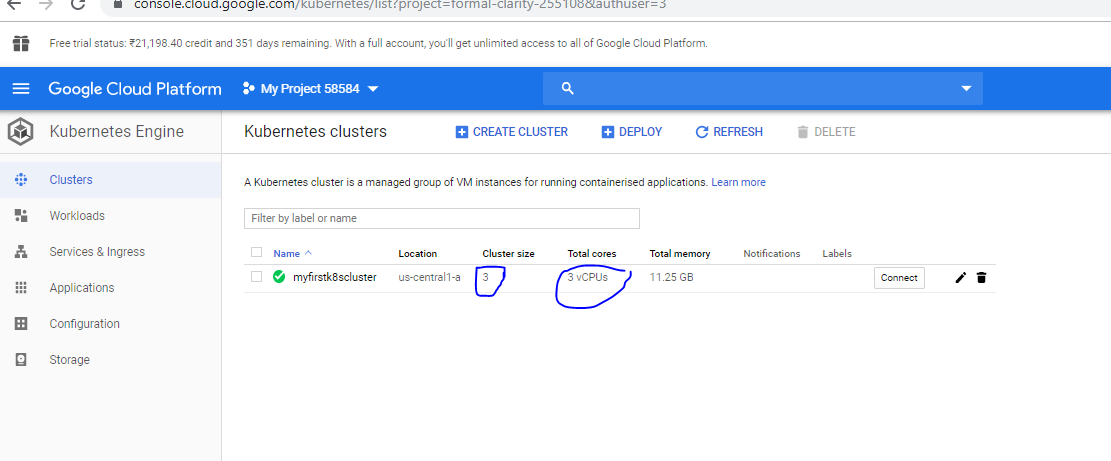
Machine Family, Machine Type are used depends on the applications. Please click on the drop-down and go through it. For this example, just use the default one.

After all this verification, Click on Create button.

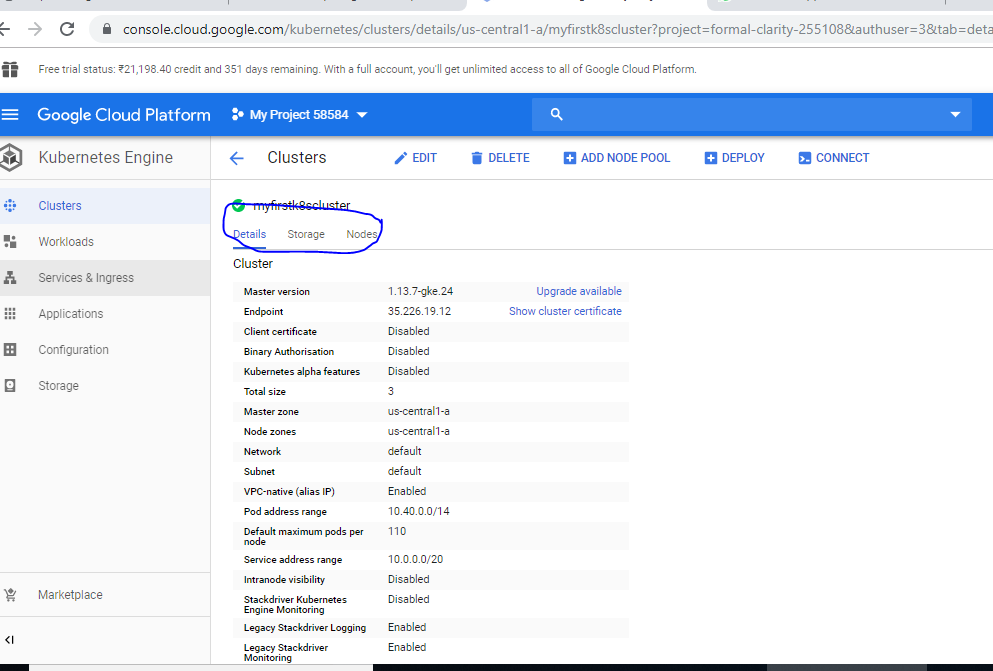
It will keep creating and you may have to wait for max., 5 mins., depends on your internet connection.

So what exactly its doing is, it’s created 3 VM instances, installing kubernetes, docker in all the instances and making 1 instance as master and other 2 nodes joining as worker nodes.

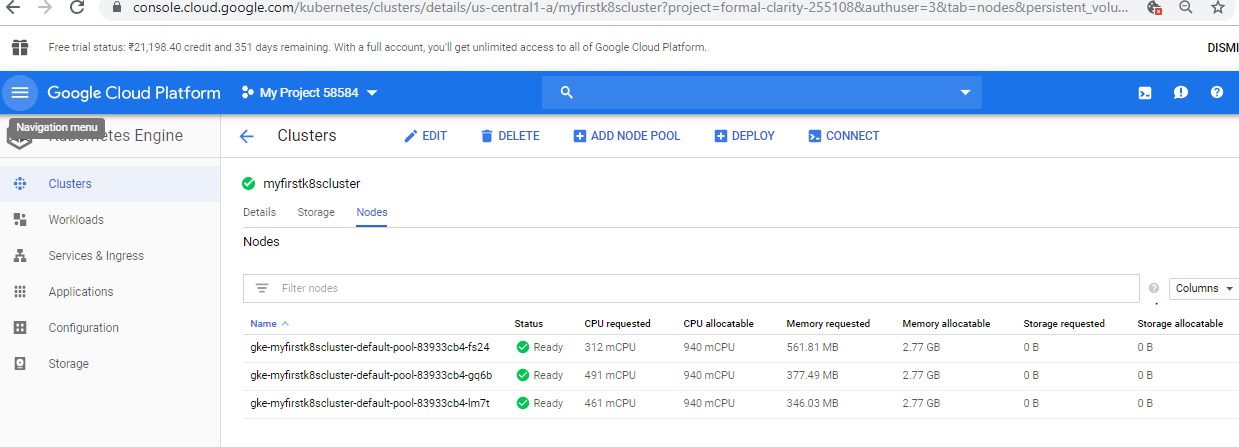
Ok. I have waited for 3 mins., and I got the below page.



Now, Click on your Cluster name.



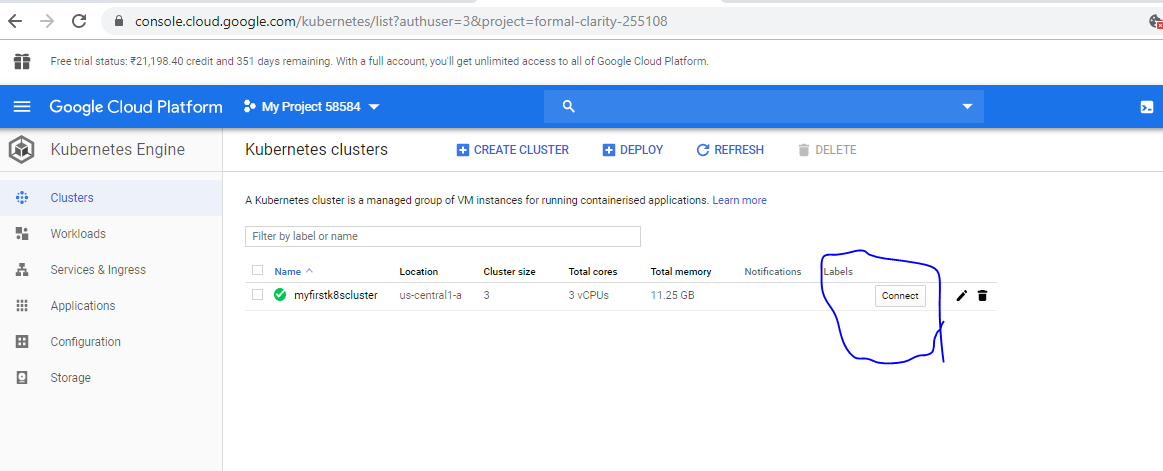
You can see the Details, Storage & nodes. Click on Nodes.



As I told you earlier, you see 3 instances/nodes are running.

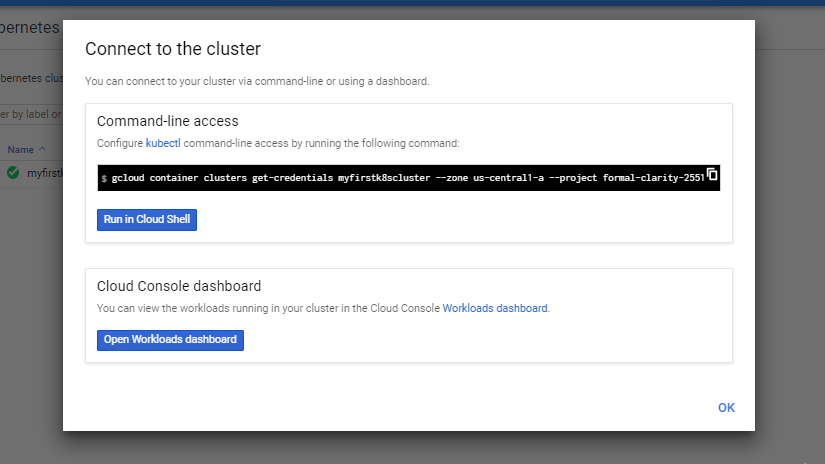
Ok. Now. Let’s see how to connect to the cluster.

You will be on this page now.



Click on Connect, and you will get the below page

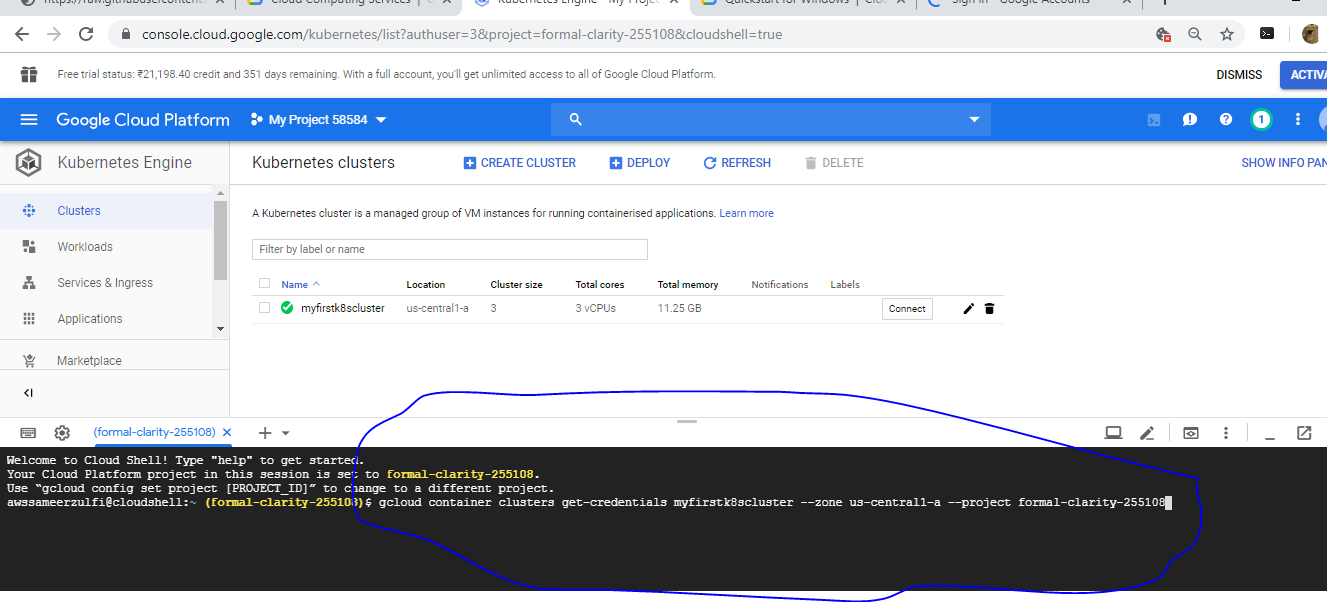
You will get the below pop-up



OK. Stop. I want to tell you something. We have various ways of connecting to google cluster. If you Google, you will see lot of steps. Here As I have already mentioned, this is not Google Cloud Course.

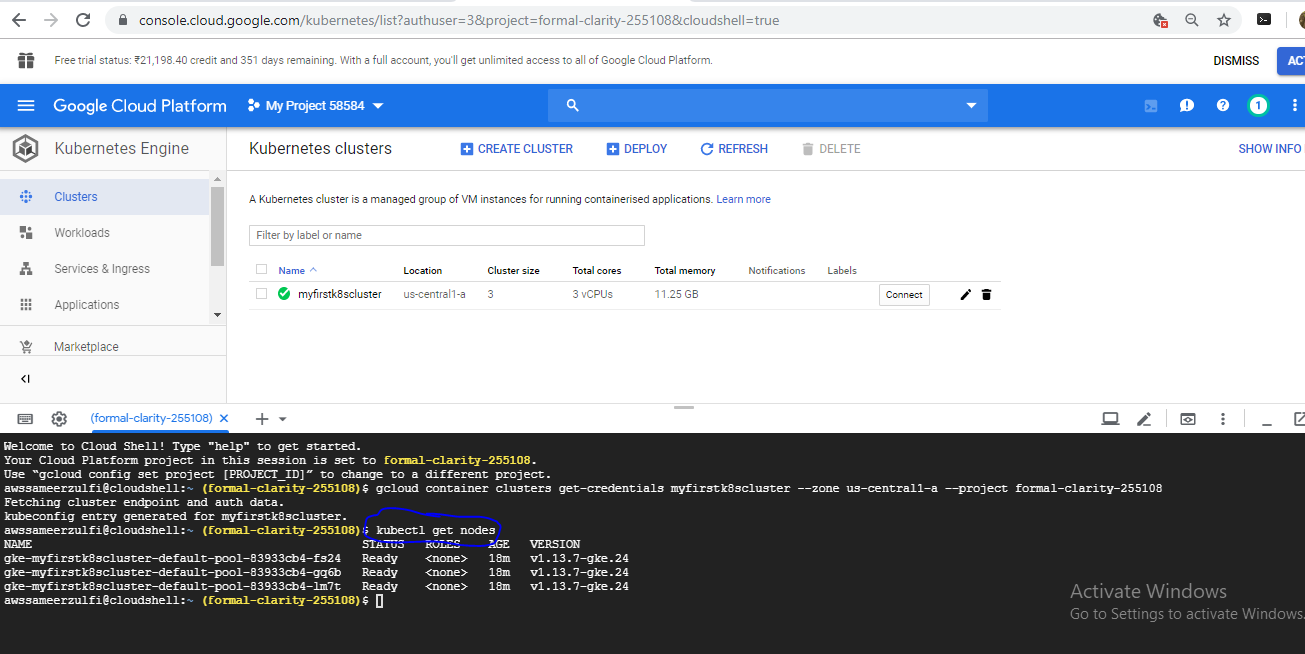
So, I am going to go with default option. i.e, “Run in Cloud Shell”

You will get the below page



Already in the black screen, there will be a command waiting for you. You need to his enter.

Try this command now

You will see 3 nodes as we expected.

A Big congratulations. You know how to create a production K8S cluster using GKE.

Let’s see the next way of installation

Installing Kubernetes on AWS with kops

Kops… what the heck is this ?

Well. Kops is short for Kubernetes Operations. This tool will make your K8S installation very easy on AWS.

Kops command line utility is only available on Mac and Linux.

What do you require for this kops:

1. Aws account and you should have decent understanding of fundamentals of AWS
2. Kubectl
3. Latest version of kops
4. AWS CLI tool
5. AWS account with following permissions:

Amazonec2fullaccess

Amazonroute53fullaccess

Amazons3fullaccess

Iamfullaccess

Amazonvpcfullaccess

1. You need a Domain of yourself or you don’t require at all

Download and Install Kubectl:

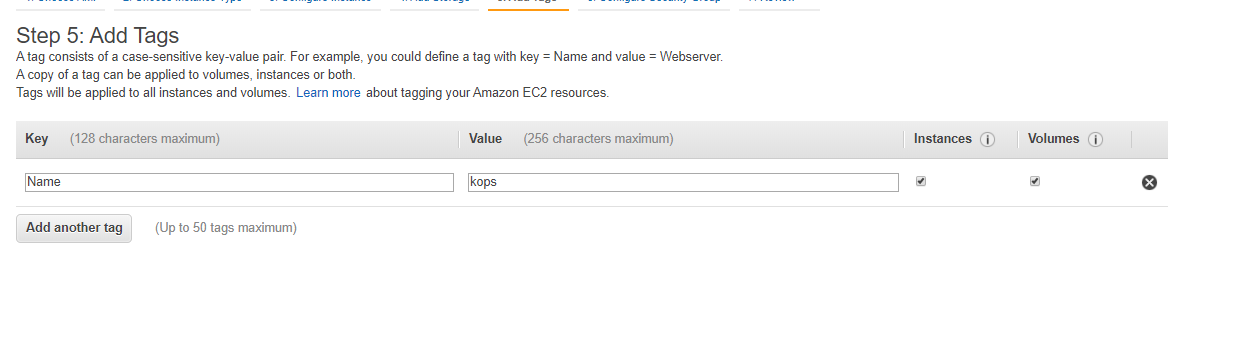
Now, before proceed on any installation, you have to spin up a new ec2 instance.

Os: Ubuntu

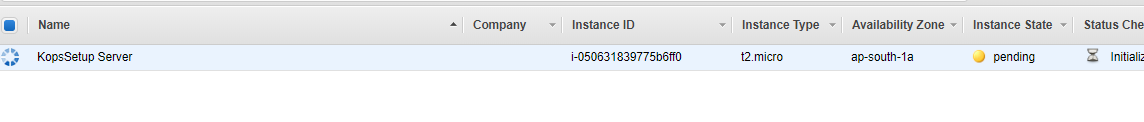
Instance type: t2.micro which is having 1 vCPU



Use any tags



Launch the Instance.



Now instance started running mode



Ok. So we have launched the instance. Now 1st installation is kubectl component

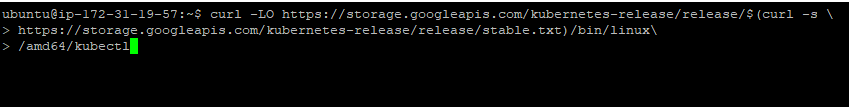
1. Download and install kubectl
2. Run the below command in the ec2 machine which you have just launched. Before running the command, make sure you update the repo by running “sudo apt-get update –y”

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s \

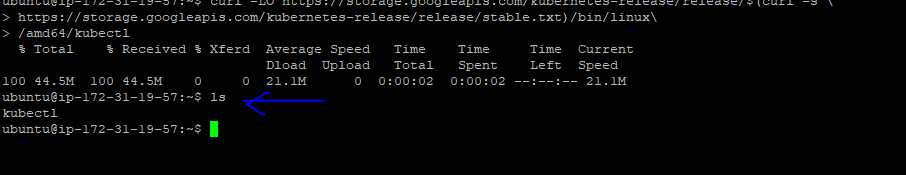
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux\

/amd64/kubectl

Make sure you just copy the above command and paste it on the instance. This command just downloads the kubectl binary in your machine.

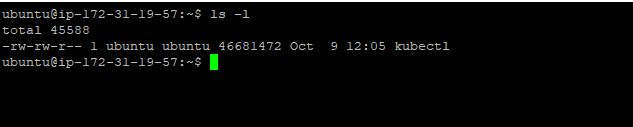


1. Just type “ls” and see the file would have already been downloaded.



1. Next is to change the permission of the file.

If you see the permission now after downloading,



There is no “X” – executable permission for this kubectl file.

We are going to update the executive permission.

1. Run the below command

chmod +x ./install



1. If you type “ls -l” now , you will see the change of the color of the file and also the “x” will be added up.
2. And now move that executable file to the below location

sudo mv ./kubectl /usr/local/bin/kubectl



Ok. We have installed kubectl. Next we are going to install kops

Download and Install kops:

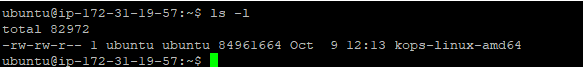
1. Copy and paste the below command

curl -LO https://github.com/kubernetes/kops/releases/download/1.11.1/kops-lin\

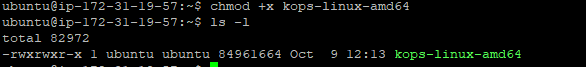
ux-amd64



1. Try “ls –l” you will get the downloaded file which doesn’t have “x” permission



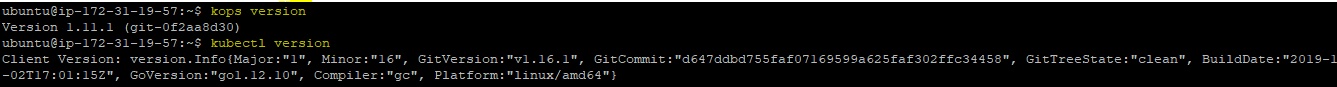
1. Execute the below command as per the screenshot



1. Execute the below command as per the screenshot



1. That’s it. You have installed kops as well.
2. If you want to see whether “kubectl” & “kops” are really installed. Please try the below command as per the screen.



If you get the above screen, you are good so far.

If you go to Page 30, we have completed 1,2,3 points. Now we will jump onto 4th point.

4th Point is “Install and Configured AWS CLI”. If you ask me what is this ? You should have some basic understanding about AWS. Call me if you want to know more on this.

For now, I could say that you can access your aws services using command line. For that you need this AWS CLI tool. That’s it.

Install and Configure AWS CLI:

1. Run the following command to install AWS CLI

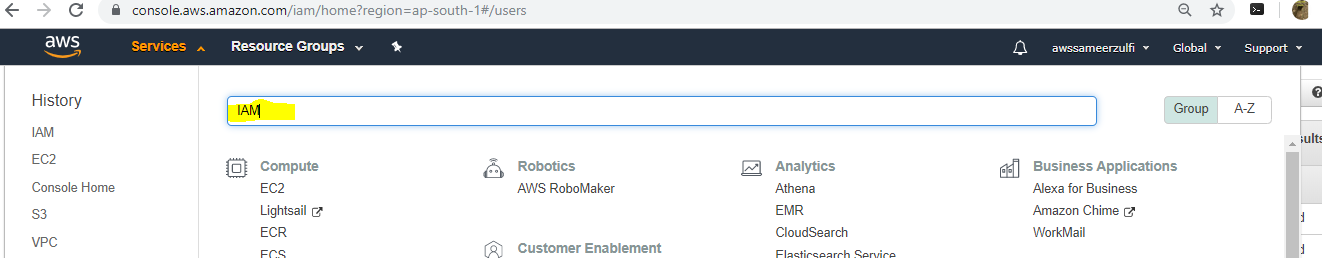


1. Before going next, if you had done AWS course before, you know what I am going to talk about.

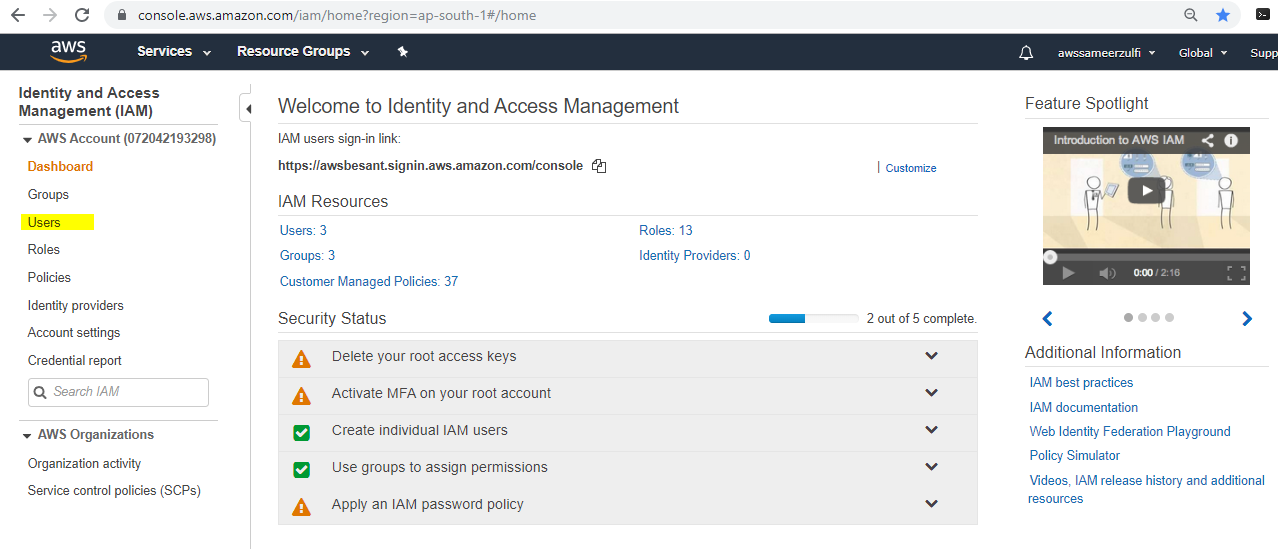
We are going to configure your aws user account in this CLI. And that user account should have couple of permissions added from AWS console.

Plan is to create a user account in AWS under IAM services. Please follow the below screenshot.

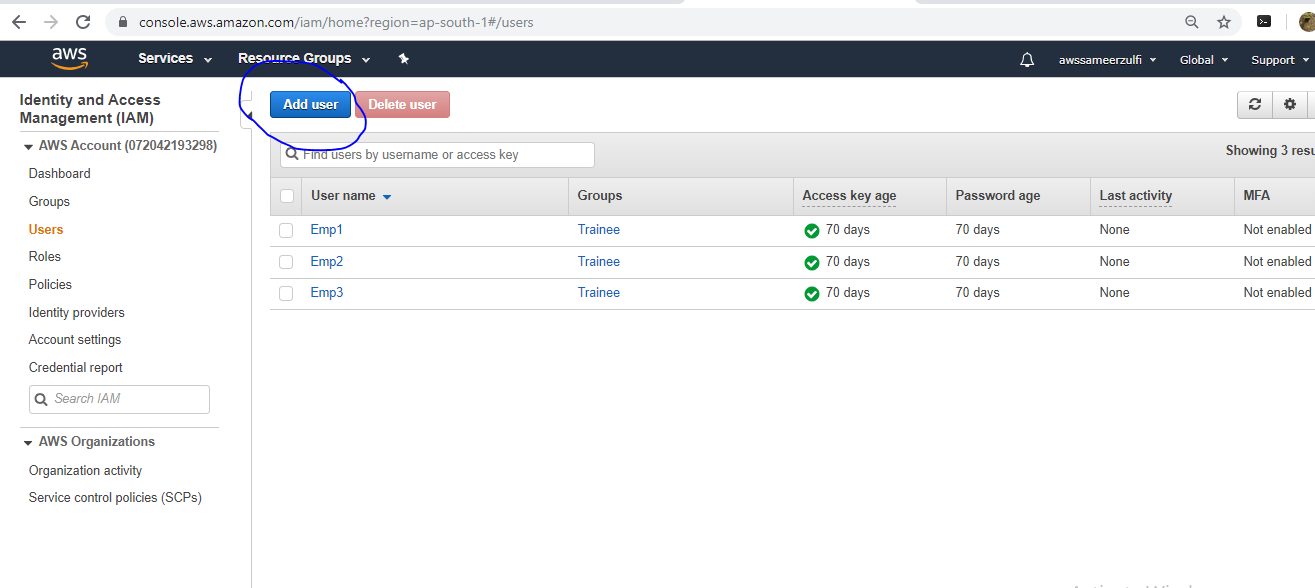
1. Search for IAM



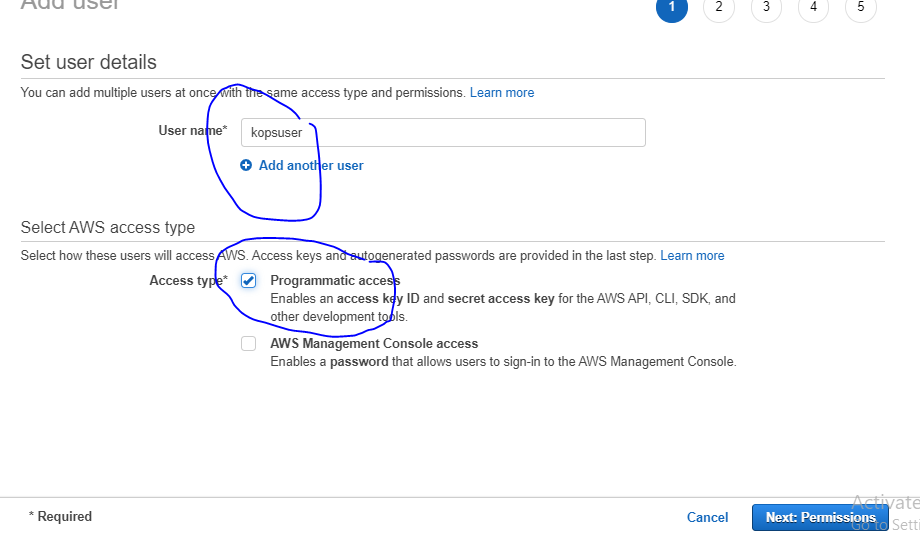
1. Click on Users on the left



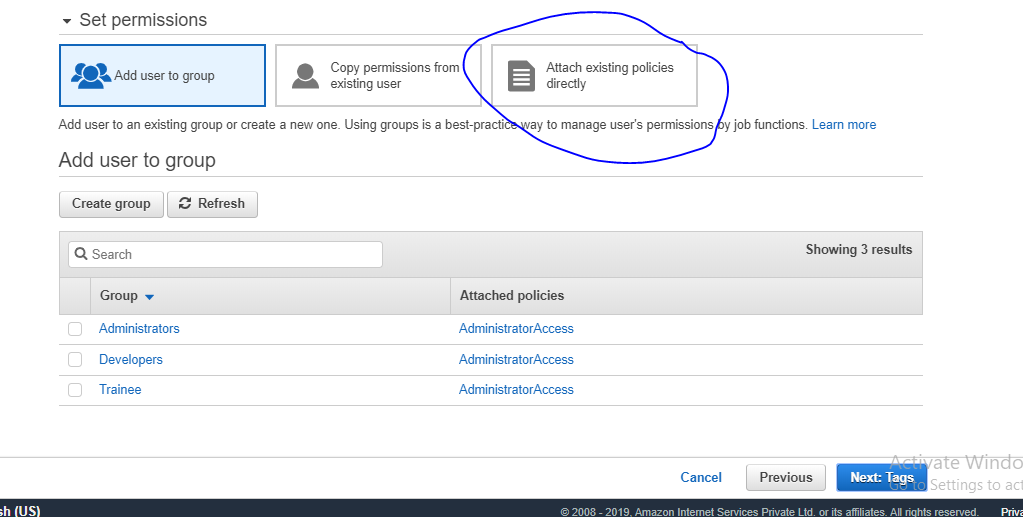
1. Click “Add User”



1. Type any name for the user and make sure you select “Programmatic Access”



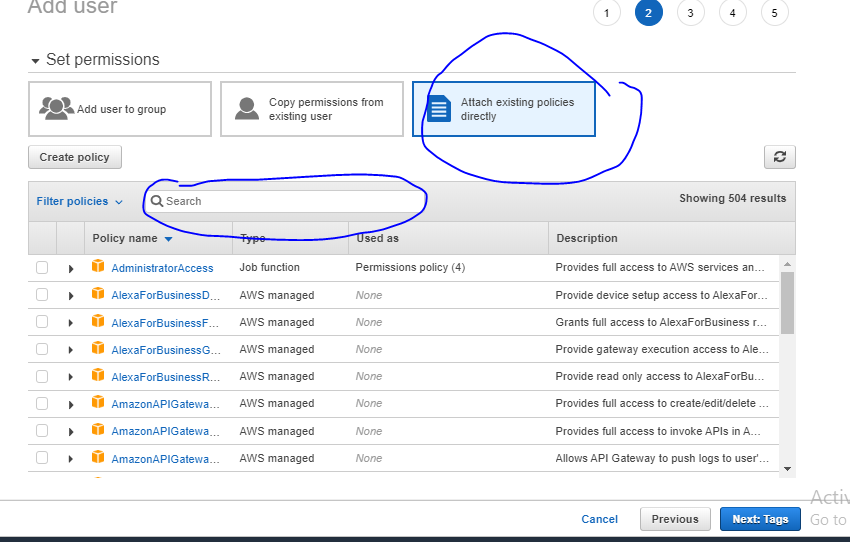
1. After hitting, Next: Permissions, Select “Attach existing Policies directly”



1. There are 5 policies you need to attach to this user.

Below are the policies you need to search in the search box

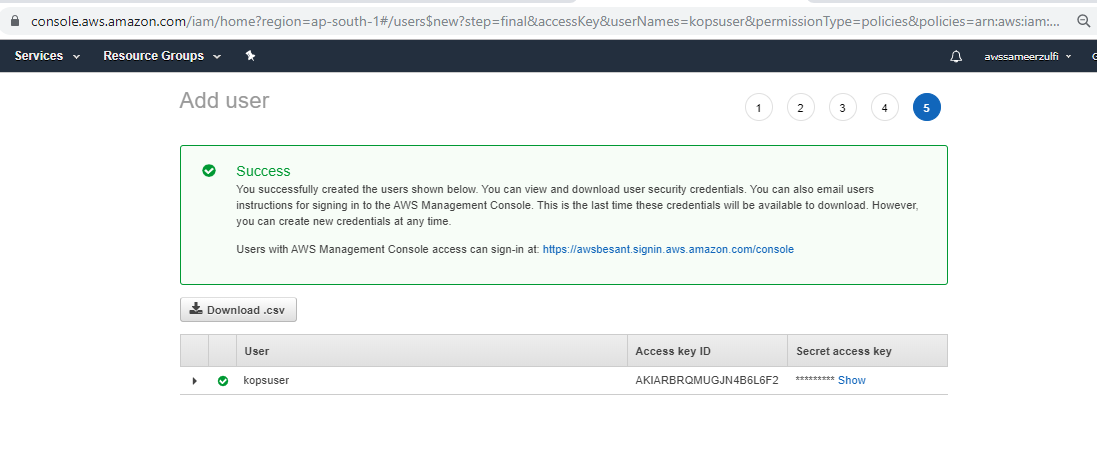
1. Amazonec2fullaccess
2. Amazonroute53fullaccess
3. Amazons3fullaccess
4. Iamfullaccess
5. Amazonvpcfullaccess



Once you have done, click Next

If you want to give a tag name, go ahead, else click next, and then Create User.

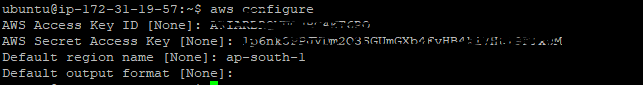
Now you will be getting the below page



The above screen shows you the Access Key ID and Secret Access Key. Please click on the Download.csv button to save it on your machine.

1. Go back to your EC2 instance and run the below command

“aws configure”



It will ask you Access Key ID , Secret Access Key.

Hope you remember, you had downloaded a file in the last step. Inside the file, you will have the keys. Please copy and paste those keys in here.

Default region name : I have chosen ap-south-1 because my region in the aws console and instance I have launched in Mumbai region.

The last one Default output format : Don’t type anything. Just Hit Enter.

1. Ok. So what exactly we have done so far.

We have installed kubectl

We have installed kops

We have created a new user in AWS and also gave 5 different service permissions

We have configured that user’s secret & access key in aws cli

Let’s see what is next now

1. Create a new S3 bucket for kops to store configuration and state information. You remember the concept of cluster store/etcd database etc., this is what it is..
2. Create an S3 bucket by running the below command



Mb means make bucket

My bucket name is cluster1prod.k8s.local

You just shuffle the name before k8s and give some other names like newclusterprod.k8s.local or primaryclusterprod.k8s.local

You can try giving anything.

1. If you type the below command, you can confirm that it has been created and available

aws s3 ls | grep k8s



1. Next is we have to tell the kops tool to find the config/state . this will be the S3 bucket which we created previously.



Remember, your bucket name would be different. You need to put that accordingly

1. We are on the final stage now. Let’s create a cluster by executing the following command… First Let’s examine each item here.

kops create cluster \

--cloud=aws \

--zones=ap-south-1a \

--name=mycluster.k8s.local \

--ssh-public-key /home/ubuntu/kops.pub \

--yes

Kops is telling to create the cluster, and the name of the cluster is “mycluster.k8s.local” and it is hosted in cloud “aws”, in the availability zone “ap-south-1a” the we are supplying the public key.

Now do you have public key for your private key file which you used to connect to your ec2 instance.

Let’s create a public key using your private key file.

I assume you guys are connected to the EC2 instance from your windows machine using putty.

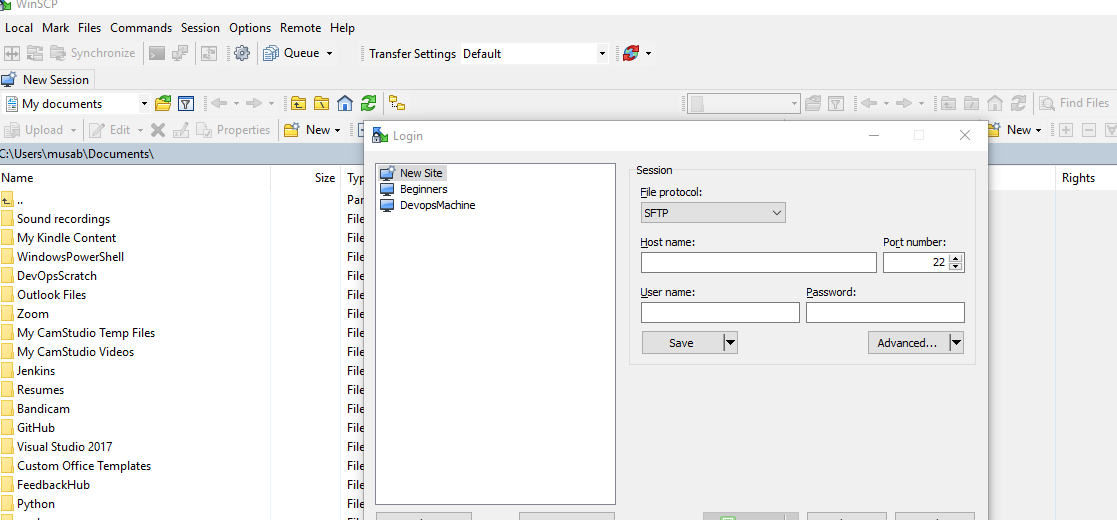
If you remember you would have converted the .pem file to .ppk file.

We want that .pem file to be moved to the ec2 instance.

How?

I use a software called “WinScp” . You can download it from google.

Once you have downloaded, you will get the below screen.

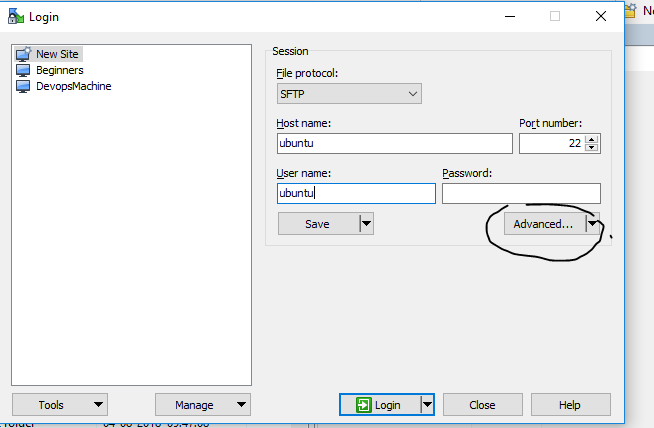


You need to type the ec2 instance public ip under Host name

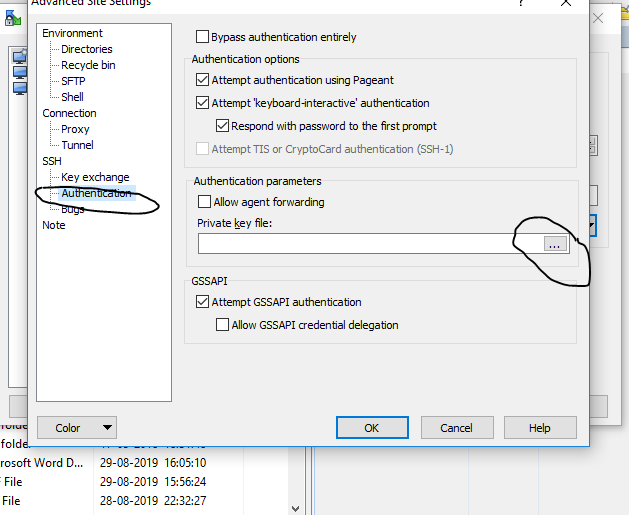
Username : Ubuntu

Don’t type anything in the password

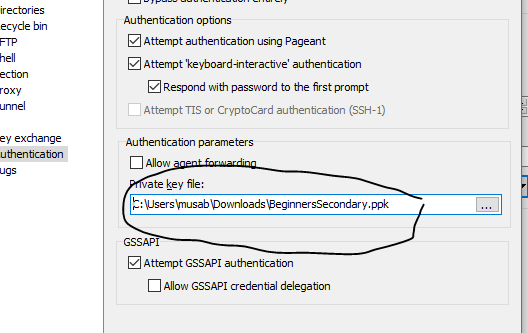
Click on Advanced button



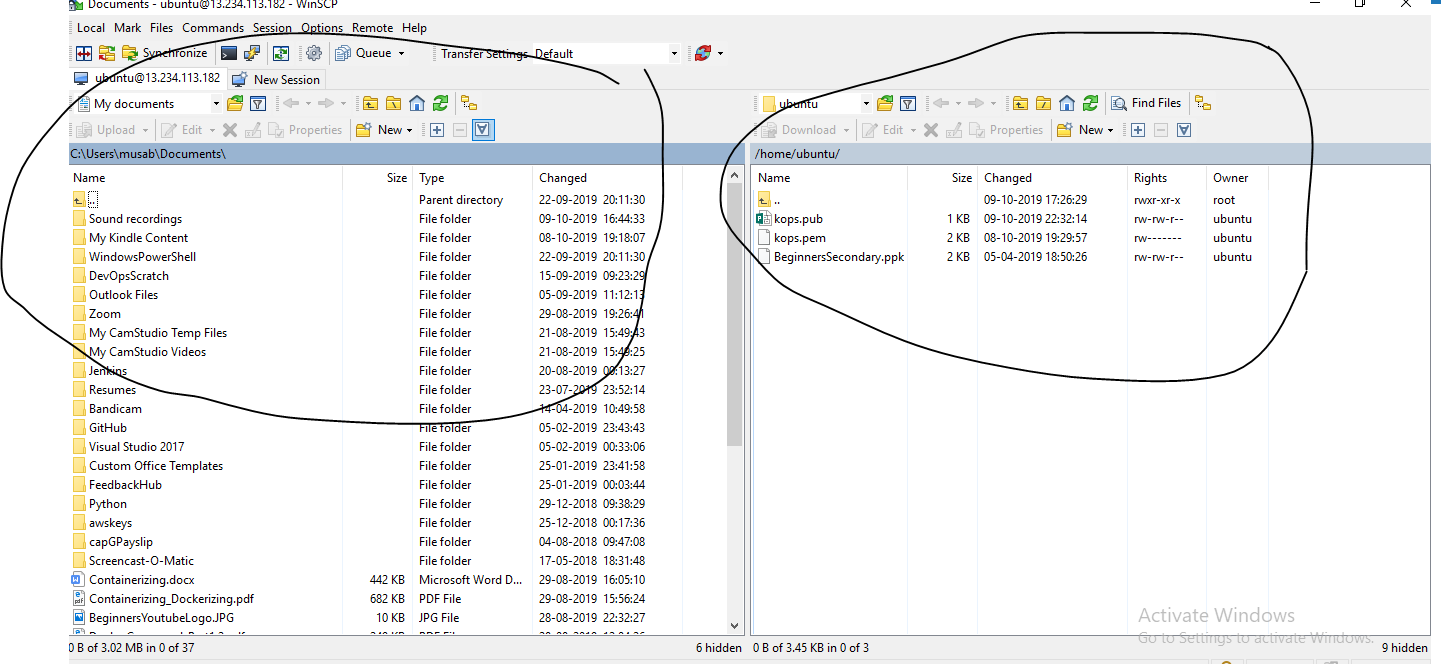
You will get the below screen :



Search the ppk file



Hit OK & Login



You will get the screen above like that.

ON left side is your laptop screen files and on the right is your ec2 instance files

From the left, you need to move your .pem file to the righ. Just click , drag and drop it.

That’s all

After you moved the .pem file to your EC2 instance. If you run “ls” , you just verify whether you have it or not

We have to convert this .pem to .pub

Execute the below command:

ssh-keygen -y -f /home/ubuntu/kops.pem > /home/ubuntu/kops.pub

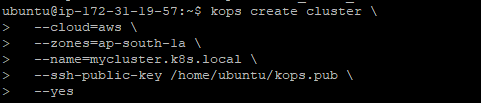
kops.pem is the one which I moved from my laptop to ec2 instance

>

Am giving kops.pub as the new for my public file

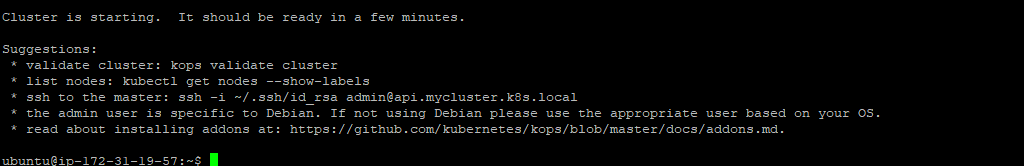
That’s it. Our public key got generated.

Let’s try to execute the cluster command again



That’s it. You will get the message like this.

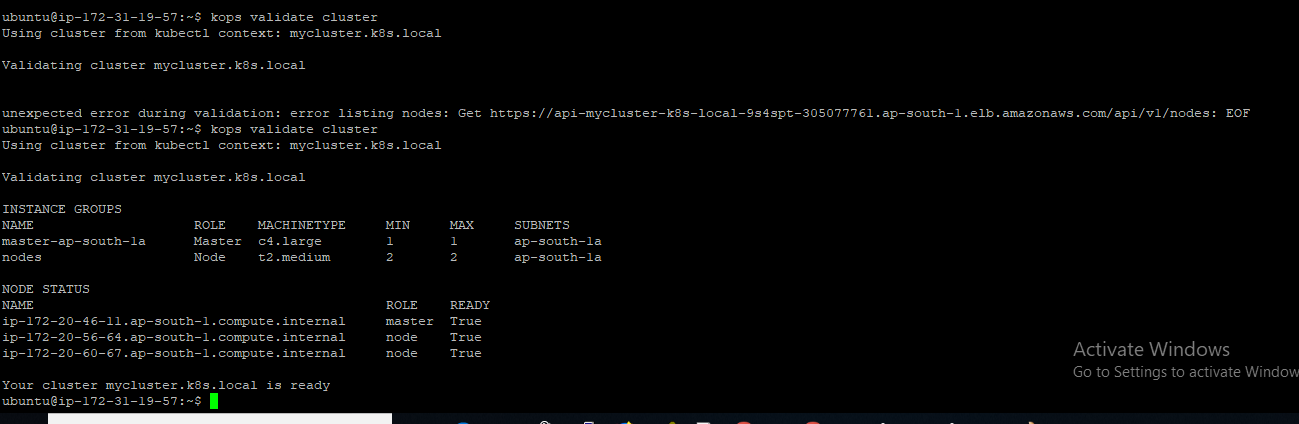
Cluster is Starting.



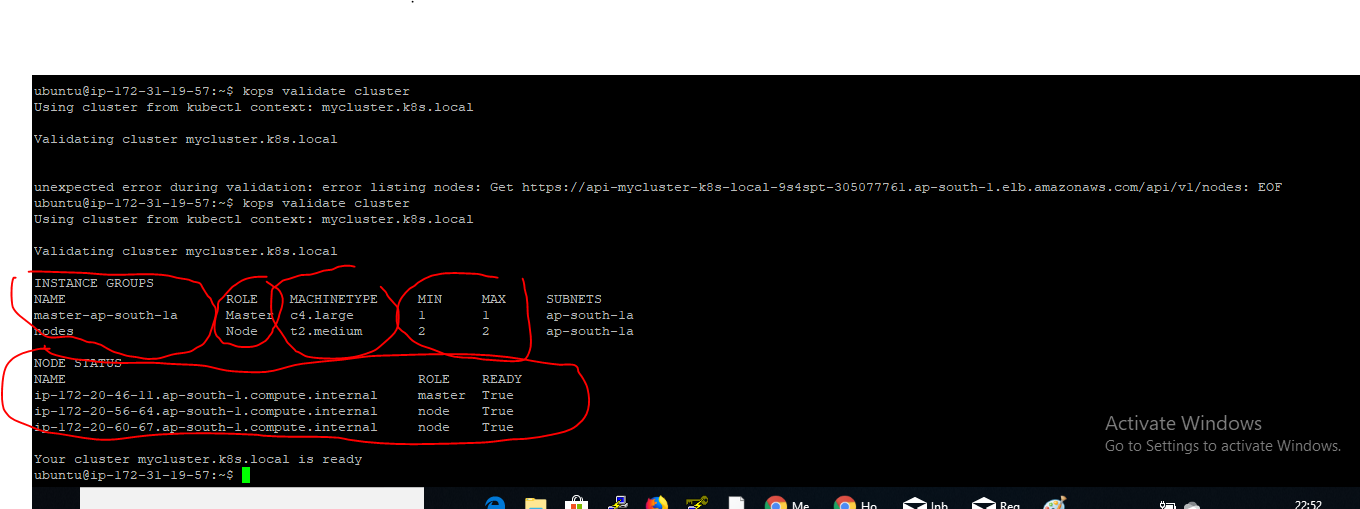
We need to wait 3 – 5 mins to get all the nodes up and running.

Now execute the below command :

Kops validate cluster



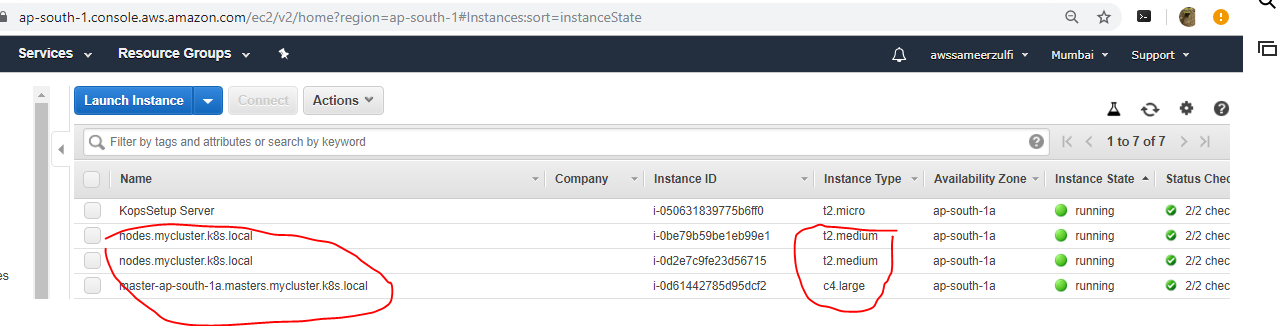
You have to analyse lot of items on this screen.



This shows by default, you have launched 1 master & 2 nodes

And below it shows the node status.

That’s it. If you go to your EC2 INSTANCE Page, automatically these 3 instances would have been created and tagged already.



If you want more nodes and more masters, then you have to update the command when you create the cluster.

Congratulations. You have done a great job now.

You can deploy apps and playaround if you want.

If you don’t want to do anything now, please delete the cluster as it will occur charges if it runs for long time.

Execute the below command:

kops delete cluster --name=mycluster.k8s.local –yes

that’s all.