KUBERNETES

Kubernetes, often abbreviated as K8S, derives its name from the eight letters between the 'K' and the 'S'. It is a powerful orchestration tool used to manage Docker containers, which can include front-end, back-end, and database services. Kubernetes handles tasks such as scaling containers, identifying and healing failed containers, and managing the interactions between them.

**Monolithic vs. Microservices**

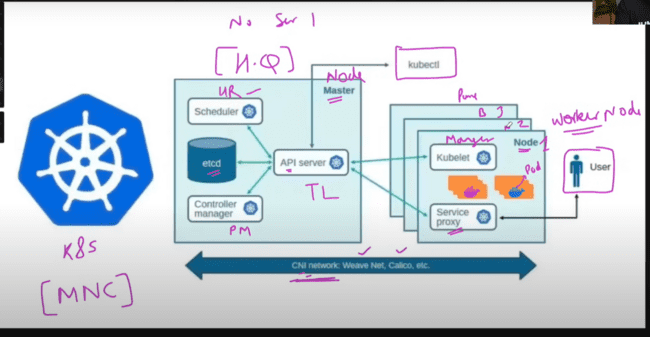
Consider a large store like DMART, which houses a variety of products such as clothing and confectionery in a single repository. This monolithic approach can be challenging to manage and maintain. In contrast, a microservices architecture is akin to having separate stores for clothing, confectionery, and kids' care, each operating independently. Similarly, microservices, often encapsulated in Docker containers, are managed by Kubernetes.

In Kubernetes terminology:

* **1 node** represents **1 server**.
* **Multi-node** refers to multiple servers, collectively known as a **cluster**.

Docker logo is like ship which is handled by sailor whose handle logo has been mentioned on the Kubernetes, so that’s why its logo is like ship handle.

It manages the cluster.



K8S master node is like Manager, which assigns the work to the worker nodes.

Docker containers are there in the worker nodes and always work there and none of the containers work in the master nodes.

API server: this is like the TL and a communication medium between the multiple serves.

PODs are the unit where containers are running, and SCHEDULER is like the HR which places the containers on the PODs and SCHEDULERS are responsible to run the PODs.

ETCD: In this data has been stored in the key value pair. This is the system where we can check when the PODs have been created, when the schedulers have been placed and the time when it has been placed.

Controller Manager: It is like a PROJECT MANAGER which checks whether all the PODs, nodes and other things are working, running or not.

WORKER NODES:

KUBELET: This is the manager on the worker nodes or WORKER NODE MANAGER to check whether all the nodes, servers are running, working or not.

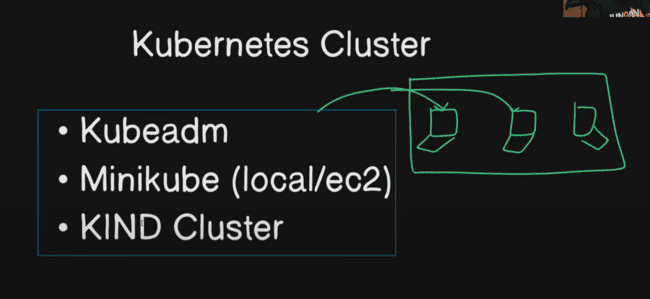
KUBELET tells this information to the API server and then API server updates this information to the scheduler, ETCD and controller manager.

PODs inside the worker nodes are accessed from the outside user using the SERVICE PROXY by asking the API SERVER or same using the KUBLET via API SERVER.

Master and worker nodes interact with each other using the CONTAINER NETWORK INTERFACE(CNI) network, example of which is Weave Net, Calico.

KUBECTL is the CEO or the director through which we can check how many nodes are active, how many PODs are running, create a POD, or create some service, this is done by KUBECTL via API SERVER.

Three ways to create Kubernetes cluster:



In the kubeadm, three EC2 instances needs to be created of t2.medium or t2.large and then combined to make a cluster, but this one is very costly.

In the MiniKube, in the local we can create the cluster or on the EC2 instance.

KIND CLUSTER: Kubernetes in Docker, Kubernetes would be running inside DOCKER container.

Fourth one is EKS of amazon, AKS of Microsoft, GKE of google. These are managed services by the cloud providers, and we don’t need to manager ourselves.

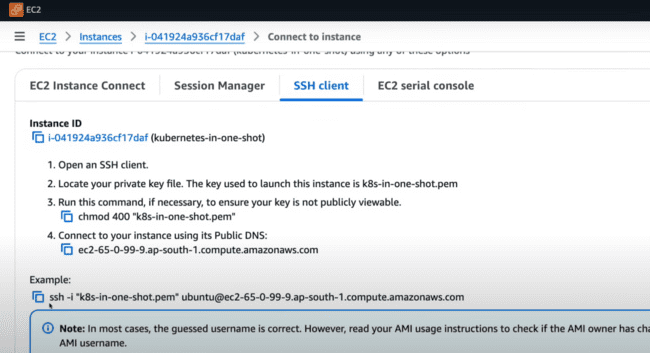
CIVO, RANCHER are other enterprise level Kubernetes.

We would be following the KIND cluster:

Creating a EC2 instance on the AWS for the Kubernetes and instance type should be t2.medium or t2.large, as it is very large infrastructure.

And storage should also be of 30GB which is necessary for this one.

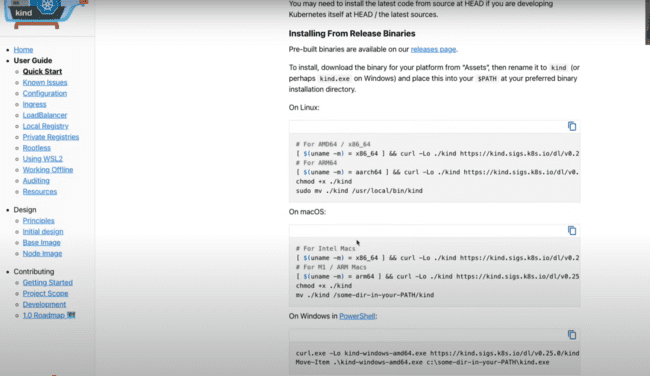
Under SSH, we need to run the command on the LINUX to connect the server:



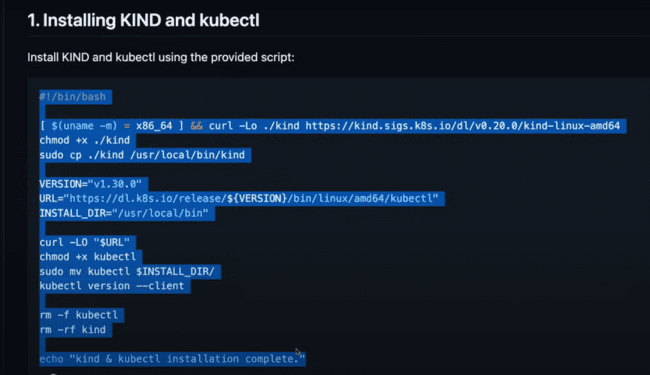
Change the permissions of the .pem file using the command -> chmod 400 <filename>

And then run the above command.

For proceeding further, we need to use the KIND cluster for LINUX or MAC, or we need to use the Minikube for the windows:



We need to execute the shell script to install the kind and kubectl:



After installing kind we need to install docker, but before that we need to run

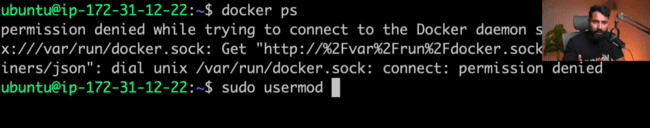
-> sudo apt-get update.

After that install the docker:

A black background with white text

AI-generated content may be incorrect.

Check if docker is installed or not:

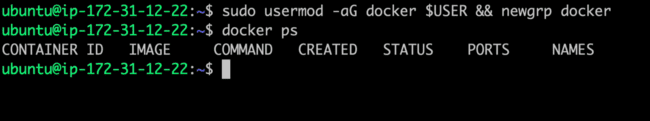


Check using whoami:

A black background with white text

AI-generated content may be incorrect.

Using command, append the user to the docker group accessing the user using the $USER and refresh the docker group using the command after the && as:



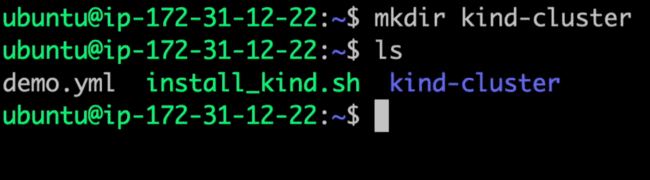
Check the docker version using the command:

A screenshot of a computer

AI-generated content may be incorrect.

We need to do the configuration of the cluster using the .YAML file.

Make a folder and go to that folder:



*The MASTER NODE in the Kubernetes is also known as CONTROL PLANE.*

There would be one file which would having the cluster information called config.yml.

In this file we need to make the kind cluster with some version and multiple nodes first is master node which is designated by control plane and other worker planes as:

A screen shot of a computer

AI-generated content may be incorrect.

Now, the cluster would be running inside the DOCKER container which would be having a different port number from the EC2 instance which would be having different port number, so we need to map them using the following key values in the same file.

PORT number 80 is HTTP and 443 is HTTPS.

So, we need to map host and container port of http and https to each other as:

A screenshot of a computer

AI-generated content may be incorrect.

Asking kind to create the cluster from the config file as:

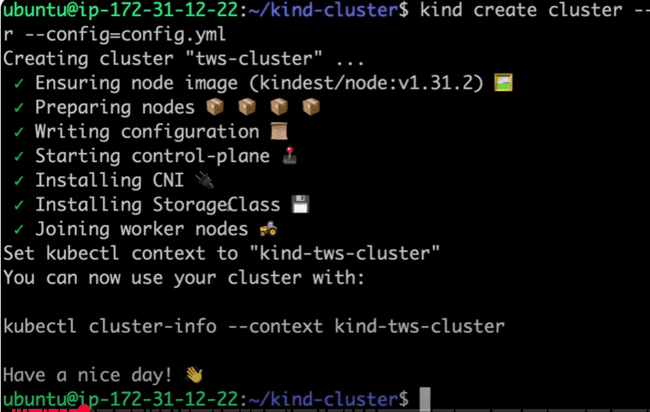
A black screen with white text

AI-generated content may be incorrect.

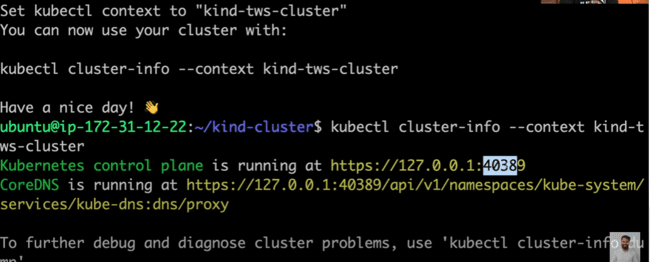
A black screen with white text

AI-generated content may be incorrect.

Things it is doing:



Information regarding the running cluster:



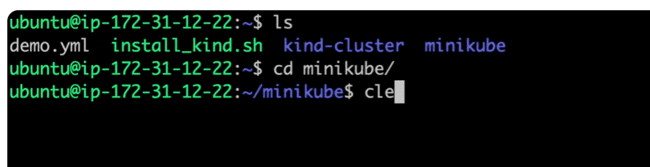
To check the running nodes:

A screenshot of a computer screen

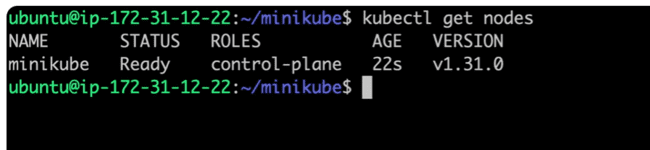
AI-generated content may be incorrect.

Creating the same using the MINIKUBE:

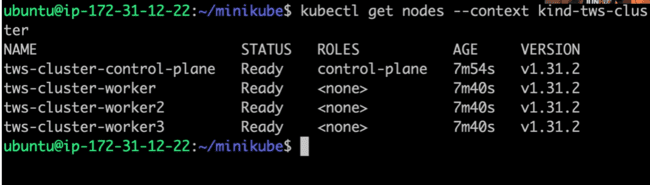
Creating the new folder as “minikube”:



Follow the steps mentioned in the document and after installation, check that to whom kubectl is pointing:



After the minikube installation, default is minikube one and it is pointed, but to get the KIND CLUSTER, we need to type the name with context as:



Delete the MINIKUBE:

A screen shot of a computer

AI-generated content may be incorrect.

Now when you type the command -> kubectl get nodes, following would be the output:

A computer screen shot of a black screen

AI-generated content may be incorrect.

So, we need to change the default context to KUBE CLUSTER:

A black background with white text

AI-generated content may be incorrect.

Now it is pointing to KUBE CLUSTER again:

A screen shot of a computer

AI-generated content may be incorrect.

CREATING A KUBEADM:

We would be creating one master EC2 instance and other worker instances and install the KUBEADM on the master.

A screenshot of a computer

AI-generated content may be incorrect.

The worker nodes are joined on 6443, which also needs to be opened.

Created two EC2 instances with t2. medium and 10 GB of RAM.

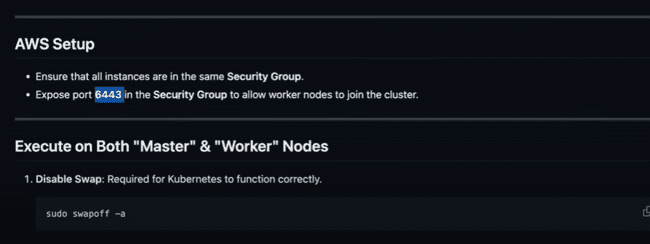
A screenshot of a computer

AI-generated content may be incorrect.

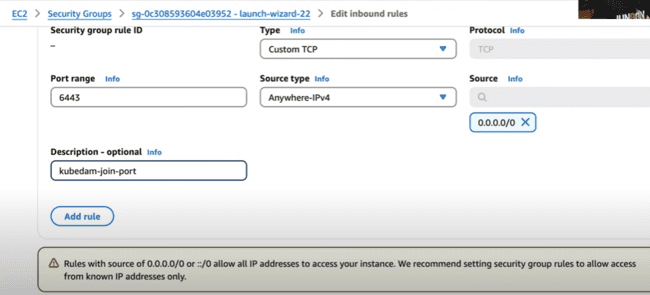
Connect both in the command prompt.

This connects Kubernetes cluster using the Contained.

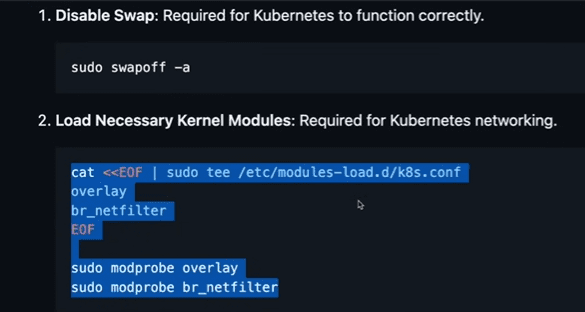
Do the following on the instances:



For the 6443 ports, do the following in the security group for the worker node:



Follow the commands for both master and worker nodes:



A screen shot of a computer

AI-generated content may be incorrect.

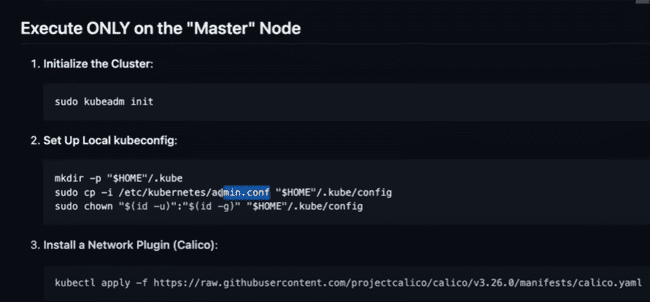
A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

In the instance where we have enabled the kubeadm init, there master would be installed.



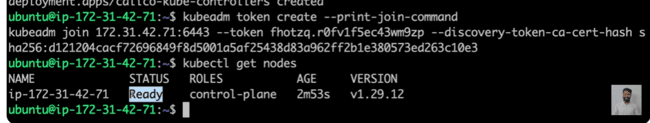
Installing the CNI (Calico) on the master node:

A computer screen with a blue and white text

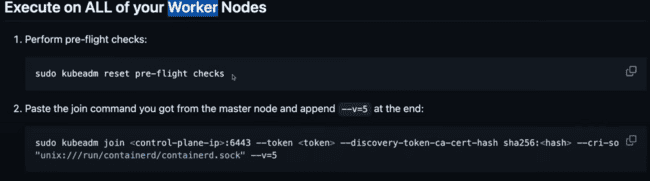
AI-generated content may be incorrect.

When we execute the generate command, then a token would be generated, and anyone can join this master network:

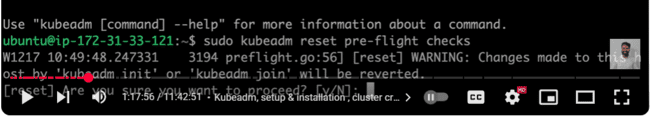
And control-plane is ready, and we would be adding the worker nodes to this one.



Execute this on worker node:



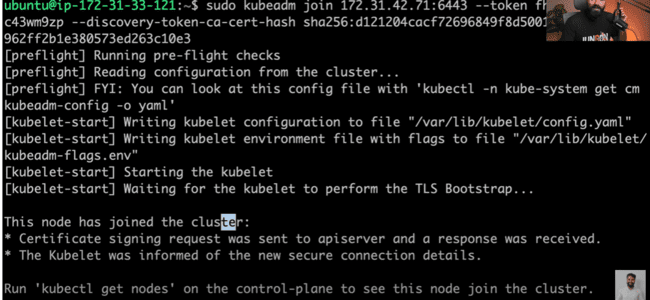
We’ll be reverting the kubeadm things from this worker node:



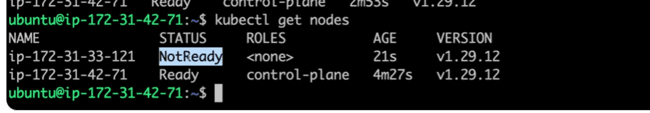
Copy the join command from master and paste in the worker node proceeding with “sudo” command:



Now this worker node has joined the cluster:



Now if you type same command, then you’ll get that node in the prompt:



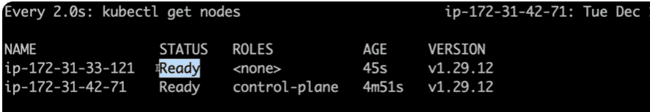
Type command “watch kubectl get nodes”, the output would be refresh every two seconds:

A black background with white text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.



NAMESPACES:

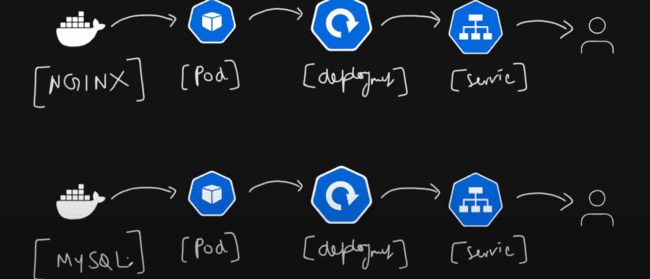
For the Kubernetes, we need to make the docker containers, and we need to run this container in the PODs and in the PODs, multiple containers can run inside that.

And the main feature of the Kubernetes is auto-scaling, auto-healing, so the next step is the DEPLOYMENT.

And now this deployment needs to be accessed from the user outside, so we need to create the SERVICE PROXY for this one, through which user can access the deployment.

For example: for NGNIX and MYSQL we need to do the same steps as creating the PODs, doing the deployment, making the service and then that service would be accessible by the user, through which it can access the deployment.

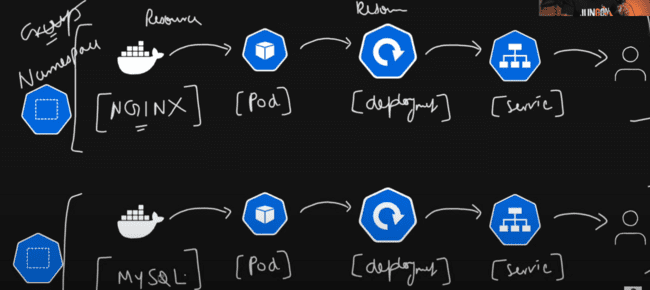
So the basic is to create the PODs, deployment and service, which remains almost same, just we need to change the containers inside that.



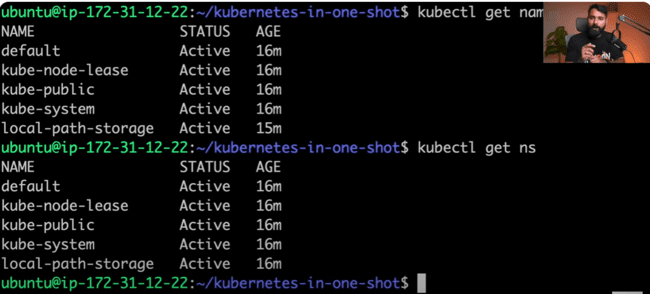
If the two containers are running in the same cluster, so we need to identify them on the basis of groups also known as namespaces.

Like for MYSQL, there is one group which contains its resources as PODs, deployment and service.

Like, two namespaces for the two containers mentioned below:



To get the namespace, we need to follow the following commands:



Now if we have not created any namespace, and created resources, then it would go to the default namespace.

Kube-node-lease: contains the information of Kubernetes cluster nodes.

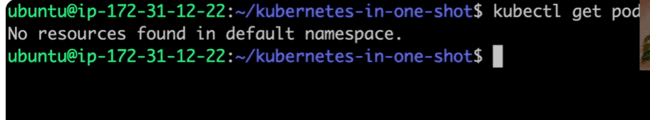
Kube-public: all the services which are accessible from public are inside this one.

Kube-system: services running on the system level run here in the kube-system.

Local-path-storage: there is one POD for the local path-storage as well which can store the data.

To get the information regarding the running pods:

“kubectl get pods”



Checking the pods in the namespace, kube-system: “-n” means in the namespace.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Creating the ngnix namespace:

A black background with blue text

AI-generated content may be incorrect.

Ngnix would be created:

A screen shot of a black screen

AI-generated content may be incorrect.

Creating the container, PODs, deployment and service in ngnix namespace:



Like creating container in docker:

A black background with blue text

AI-generated content may be incorrect.

POD has been created in the default namespace, not in the ngnix namespace, so we need to delete that:

A black screen with white text

AI-generated content may be incorrect.

A black background with blue text

AI-generated content may be incorrect.

Creating the POD again but with the namespace in the ngnix with appending the command as: “-n ngnix”

A screen shot of a computer

AI-generated content may be incorrect.

Deleting the POD in the ngnix namespace and ngnix namespace also:

A black background with blue text

AI-generated content may be incorrect.

Now, we’ll be creating all the PODs, deployment, service using the yml file which is also know as MANIFEST file.

Creating a new folder ngnix and creating the yml file inside that, which is manifest file.

Creating a new file as “namespace.yml” file:

Metadata is an object under which we need to create the re2

A screen shot of a computer

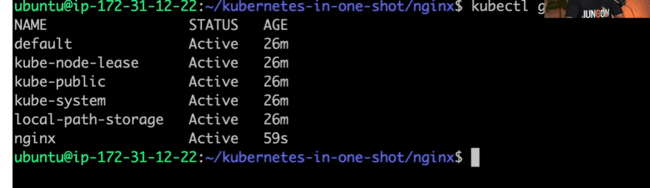
AI-generated content may be incorrect.

“create” command is only creation.

But “apply” is creation as well as update.

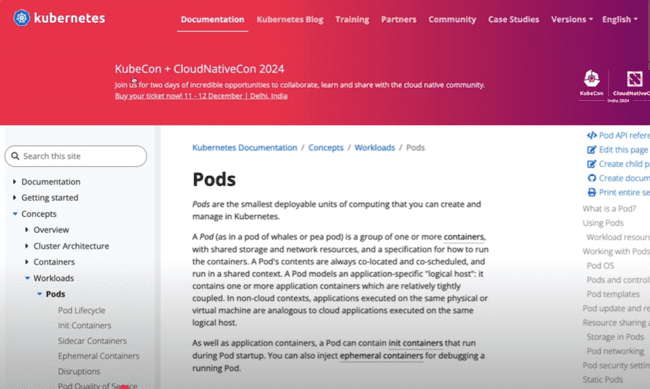
So, applying the namespace.yml to the API server and apply to the cluster.



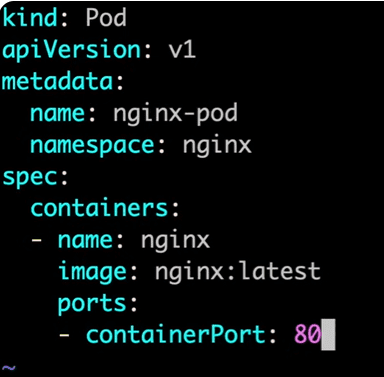
Ngnix object would be created by checking the command “kubectl get ns

Creating pod.yaml file and creating the POD inside the “ngnix” namespace.

Search for the Kubernetes pod on the internet and check the following site:



In the spec of the POD in yml file, we need to create the containers, which can be either one or many:



Same “apply” command for the POD.

A black background with blue text

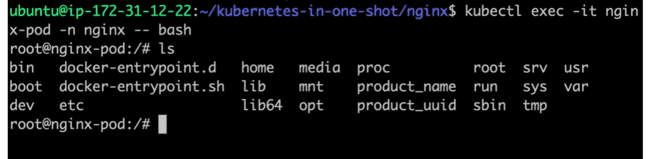
AI-generated content may be incorrect.

No Pod in the default namespace, but there in the ngnix namespace:

A screen shot of a computer

AI-generated content may be incorrect.

Entering inside the ngnix POD, in the same way, we are entering in the docker container:



Using curl command, welcome message would be printed on the screen:

A screenshot of a computer

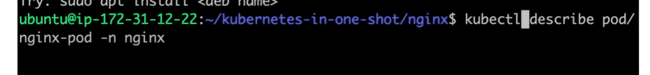
AI-generated content may be incorrect.

A computer screen with white text

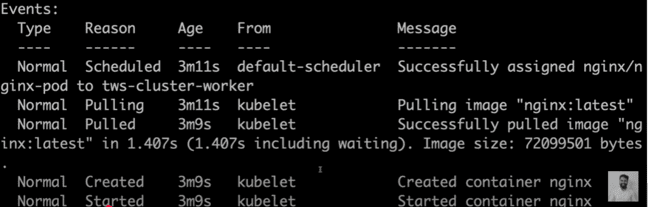
AI-generated content may be incorrect.

We can see the PODs in the ngnix, but does that POD is running fine or not, we need to debug that one, which we would be checking later:

Describing the ngnix POD: This command logs everything, which happens to your POD, all that from creation to who has done what thing to that.



Following are events which are fired when POD is getting created:



Assigning the ngnix pod to cluster which has been done by default-scheduler.

Then KUBELET pulls the image and then created and started the container.

KUBECTL asks API server to create POD ngnix, then API server goes to SCHEDULER, the SCHEDULER decides to start the POD on the first node, then KUBELET pulls the image and then runs the image, then finally POD started.

A computer screen shot of a diagram

AI-generated content may be incorrect.

Command to enter inside the POD:

A black background with blue text

AI-generated content may be incorrect.

*DEPOLYMENT: making the POD scaleable is the major feature of the Kubernetes.*

Now you have created multiple PODs, but if there is a lot of traffic, so if there are 3 PODs and millions of requests, then these PODs won’t work and crash, then we need to create group of PODs or REPLICA of PODs.

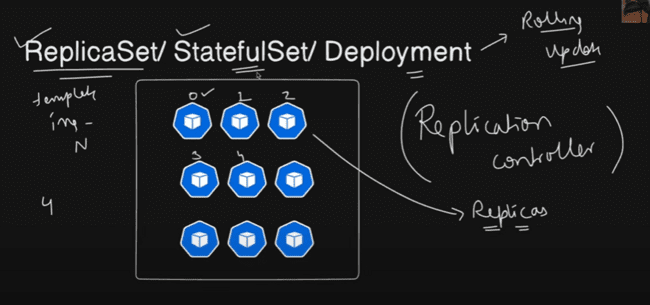
And there are three things that are used in creating the REPLICAS:

Replica Controller: manages the replicas of PODs.

Replica Set: In this you need to use the “template” in the yml file and ask to create the 4 replicas of ngnix container, then 4 PODs would be created.

Stateful Set: It designates the number to each of the PODs, and the state would be maintained on the basis of number of each set.

Deployment: This is almost similar to Replica Set, but there is one difference, it provide the rolling update which is like, if the image would be update inside the PODs, so it won’t update images in all the replicas of the POD immediately, which would create the downtime, due to update. It updates one by one. Like, one POD is running, while update is going on other one, and so-on and no down time would be there.

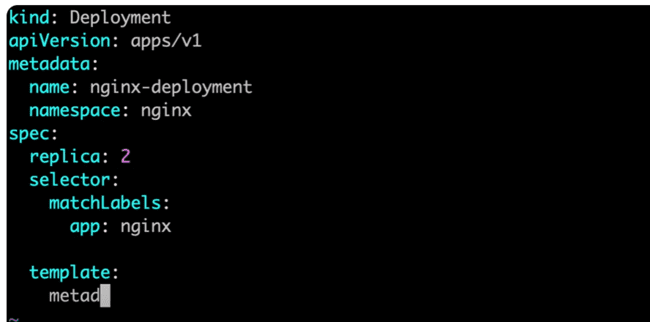


Like the POD and other yml files, we need to create the deployment.yml file.

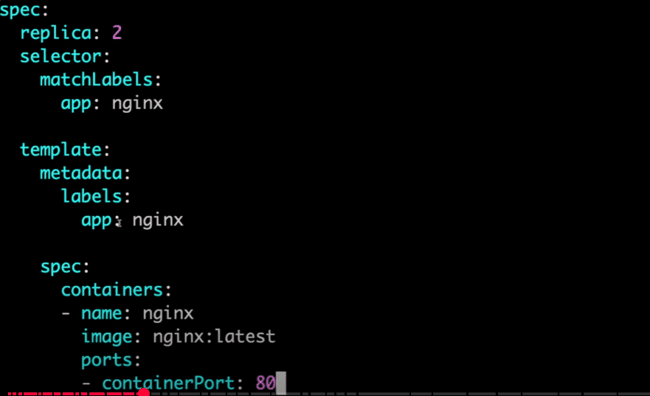
POD is identified using the labels and to select that POD, we need to use the selector, which is then understands by the deployment unit.

So, we give each POD a label and selector and asks the same to the deployment unit to match with and create the replica of them.

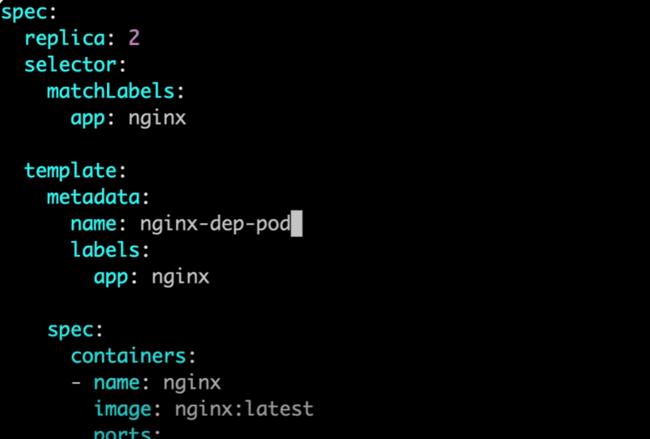
We need to match with the label of the app as ngnix:



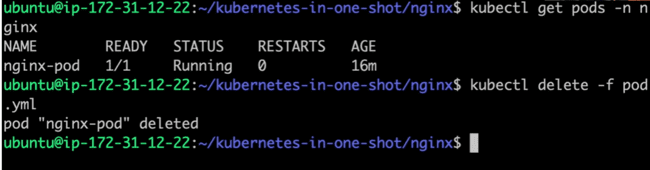
Then in the template we need to provide the container name which is same in the pod.yml.



If we don’t need to write the POD.yml, so we can provide the name under the template:



As POD will be created by above yml file, so we can delete the created POD using:



Executing deployment.yml:

A black screen with white text

AI-generated content may be incorrect.

Issue: changing the name from replica to replicas in yml file.

Two PODs get ready after the command:

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a black screen

AI-generated content may be incorrect.

If we want to scale that to 5 PODs:

A screenshot of a computer program

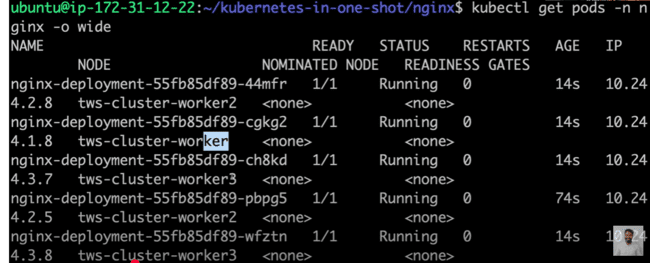
AI-generated content may be incorrect.

It can be scaled to 1, 10 or any number:

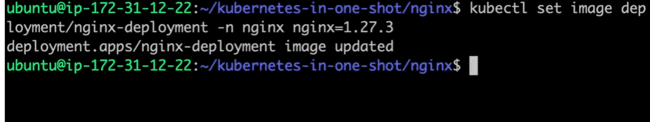
A screen shot of a computer screen

AI-generated content may be incorrect.

Scaled the POD to 5 and get the more information related to PODs using the “-o wide” command:



These PODs are using the latest ngnix version and if we want to update to a older version using KUBECTL rollout command:



Now if you try to get the POD information, following would be displayed:

A screenshot of a computer program

AI-generated content may be incorrect.

There is issue while pulling the image, so we need to update “ngnix=1.27.3” to ngnix=ngnix:1.27.3:



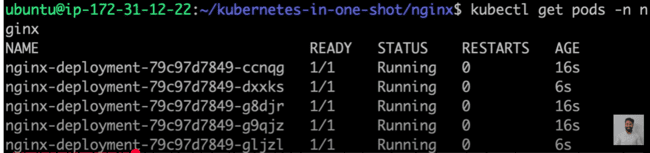
Now containers started creating:

A screenshot of a computer

AI-generated content may be incorrect.

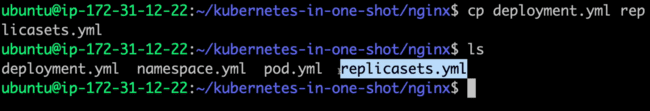
**This ContainerCreating and ErrorImagePull is the example of rolling update, like all the PODs are not down, some are running and some are creating the issues.**

Again after getting the POD information:



REPLICA SET:

Copying the deployment.yml to the replicaset.yml file:



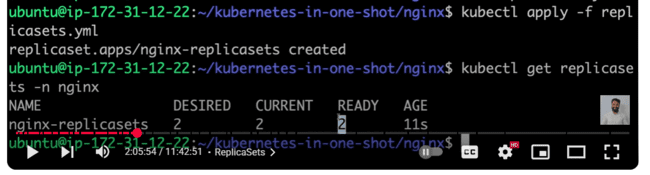
The file would be same as the deployment, only some configurations need to be changed:

Kind, metadata.name and template.metadata.name need to be updated:



And the only difference is the rolling out update between the replica set and deployment set.

Applying the replicaset.yml files and getting the replicasets:



DAEMON SETS:

Now we have three worker nodes and the PODs have been assigned only to the two worker nodes as shown in the following command:

A screenshot of a computer screen

AI-generated content may be incorrect.

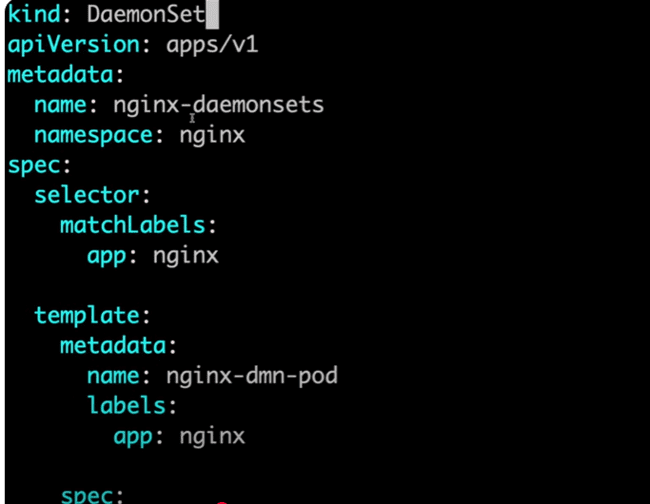
Now Daemon sets make sure that for all the worker nodes, there should be at least one replica or POD assigned.

Deleting the replicaset.yml and creating the new daemonset.yml file:



And copying the replicaset.yml to the daemonset.yml file and then making the changes:

Same configuration needs to be updated with the “daemon” as that of replica and need to remove the “replicas” tag, as it assigned every worker one POD.



Daemon set created:

A screen shot of a computer program

AI-generated content may be incorrect.

There are three nodes and three PODs gets assigned:

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

JOBS AND CRON JOBS

There should be one task associated with some container and after the task is done, then that job is closed.

Like taking a backup, doing the update once, or the system update and that’s it, the job is done.

The job can be single handled or that can be multi-threaded. Like taking the update of system by one POD or by three PODs at once.

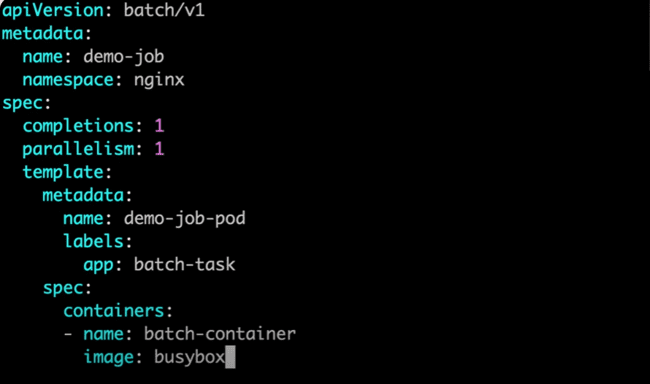
Creating a file job.yml:

“Completions:1” is how many times you want this job to execute.

“parallelism”:1, means one POD would be working on this job and if that is 2, then two PODs would be parallelly working on this job.

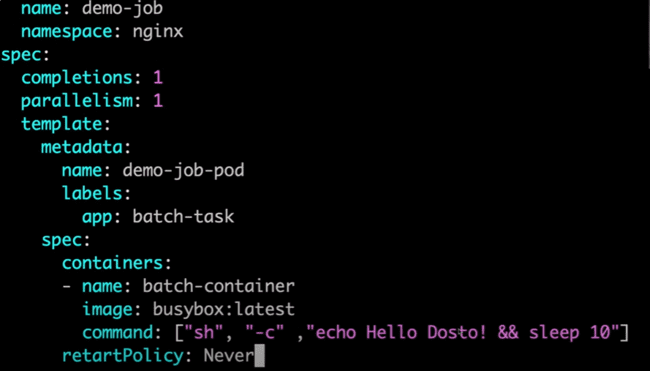
The “template” tag means making the PODs.

“image:busybox” is used to run the commands, like in the similar way ngnix work is to surf the websites, and mysql work is for the database.



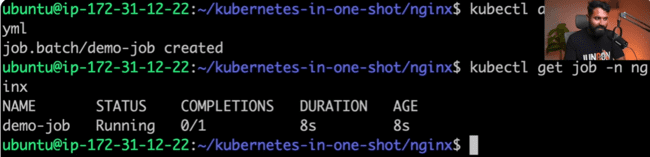
“command: [“sh”, “-c”, “echo HELLO && sleep 10”” means running the shell from the sh command which is “-c”, which would print “HELLO” and then sleep for 10 seconds.

“restartPolicy” is for restarting when the process is completed or failed.



Need to update the retart to restart and then run using the apply command:

Job start running:



It will take 10 seconds, as there is sleep for 10 seconds. After 10 seconds.

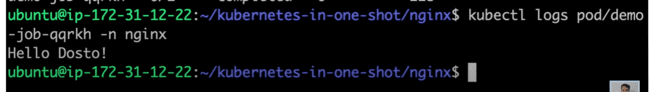
A black screen with white text

AI-generated content may be incorrect.

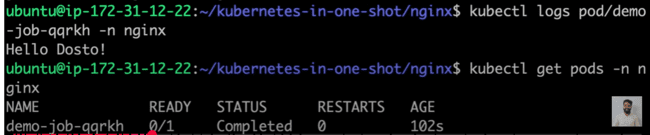
POD lifecycle:

CONTAINER CREATING -> RUNNING -> TERMINATING -> COMPLETED.

To check the log of the POD:



We can again run that:



And if you want to check the states of the PODs again, we need to delete that and run again as:

A screenshot of a computer screen

AI-generated content may be incorrect.

Now if you want to schedule this job at some time that is called CRON JOB.

Creating one cronjob.yml file.

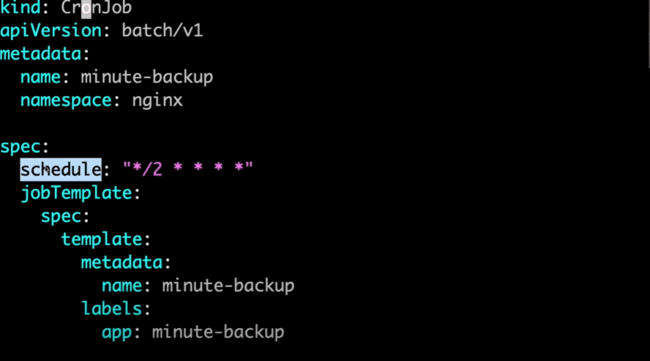
CRON pattern:

A black background with green writing

AI-generated content may be incorrect.

First star is minute, then hour , day, day of week, day of month.

“schedule” follows the cron pattern which needs to be get from the CRON job site.



A screenshot of a computer program

AI-generated content may be incorrect.

If you want to provide the command in the single line, then we need to use the “command: []” or we can use the following if we want to give the command in multiple lines as:

After the “>” symbol, then we can write any type of commands line by line as shown below:

There is need of volume of give or take the data from the container, so we need to use the volumes. Volume is a unit which allows to make the folder and temporarily stores the data.

If we want to restart on failure, then we need to use the “restartPolicy: OnFailure”

WHAT IS VOLUME:

Example: if there is one container running inside the POD, there is one HOST device and now you want to persist the data between the HOST and the container, then we need to use the VOLUMES, so that data won’t be lost and mount the same volume to the container.

A screen shot of a computer

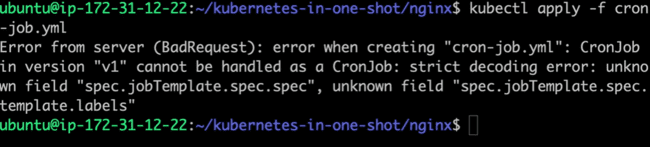
AI-generated content may be incorrect.

“type: DirectoryOrCreate”, if the directory is not found then it would create one.

And we need to mount volume for both ‘demo-data' and 'backups’

Now the job is to take the backup from one folder to another.

Issue while running apply command:



Need to use labels inside the metadata as:

A screen shot of a computer

AI-generated content may be incorrect.

Again, issue of unknown field:

A computer screen with text

AI-generated content may be incorrect.

Need to use the spec inside the jobTemplate and need to spec inside the template of that jobTemplate spec.

Again there is one issue:

A computer screen with white text

AI-generated content may be incorrect.

“volumes” inside containers should not be under “volumeMounts”.

A screen shot of a computer

AI-generated content may be incorrect.

Again issue:

A screen shot of a computer program

AI-generated content may be incorrect.

“restartPolicy” should be for the containers:

A screen shot of a computer

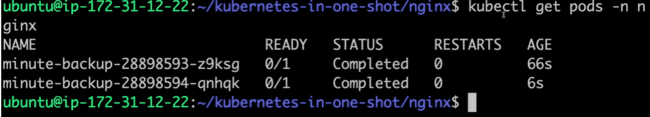
AI-generated content may be incorrect.

Apply and run the .yml file:

A screenshot of a computer program

AI-generated content may be incorrect.

Backups have been completed for one minute:



To check the logs of the POD, then we need to give the name and check the logs:

A black background with blue text

AI-generated content may be incorrect.

Again, in the next minute, another task completed:

A screen shot of a computer

AI-generated content may be incorrect.

STORAGE AND VOLUMES:

What are voulmes and volumeMounts in the yml file above created.

PODs are running inside the nodes and in the future, if the data inside the PODs gets deleted.

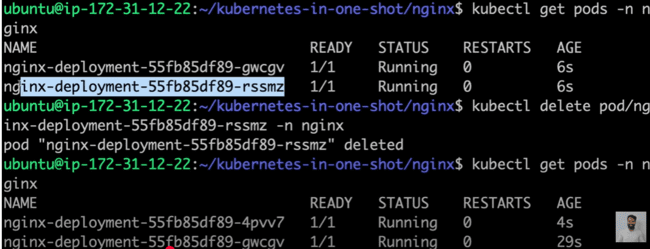
Example:

We have created two pods using deployment.yml file as:

A screen shot of a computer

AI-generated content may be incorrect.

If we have deleted one POD, then another would be automatically created which is known as replication and which is auto in deployment.



But the question here is what happens with the data of that deleted POD…...?

So, we need to persist that data to the HOST which can also be said as binding, so that the data won’t get lost.

For that we need to use PV (Persistent Volume) and PVC(Persistent Volume Claim).

If we have the host machine of 30 GB and we want to provide the 1 GB to provide for the persistence.

Creating the persistentvolume.yml file.

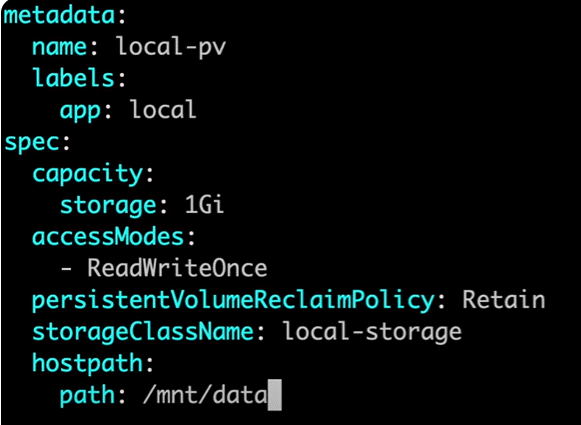
If we wanto retain the persistentvolume, even if gets deleted, the we use the command “persistentVolumeReclaimPolicy: Retain”

“storageClassName” is used to provide the type of storage which would be local storage of host machine or cloud storage.

A computer screen shot of a black background

AI-generated content may be incorrect.

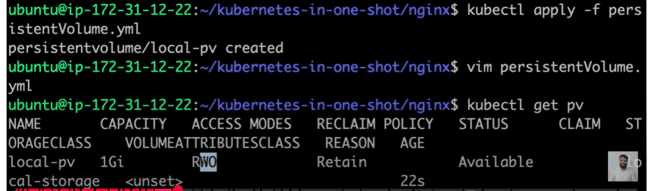
Also need to provide the path of the storage in the host machine using:



A black screen with white text

AI-generated content may be incorrect.

Error was there in which “hostpath” needs to be updated as “hostPath”



Now, from the above command, we can check that volume is available, means PV is available, so we need to claim that using the PVC.

Now we need to make a new yml file persistentvolumeclaim.yml.

“accessModes” is same.

And to claim the storage, we need to use the “resources.requests.storage: 1Gi”, like how much we need to claim we can define that, it can be less that 1GB or equal to 1GB.

What is basically “local-storage”…..?

For the command “kubectl get ns”:

A screen shot of a computer

AI-generated content may be incorrect.

Which has been described at the starting, so this local-path-storage creates the storage binding with the docker container.

A screen shot of a computer

AI-generated content may be incorrect.

Error: spelling of resources was incorrect.

A computer screen shot of a error

AI-generated content may be incorrect.

A computer screen shot of a error

AI-generated content may be incorrect.

Now of you check that 1 Gb got bounded:

A black screen with white text

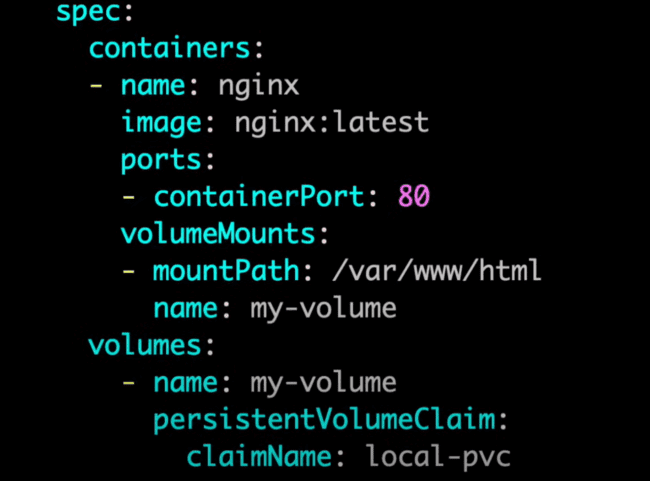
AI-generated content may be incorrect.

A black background with white text

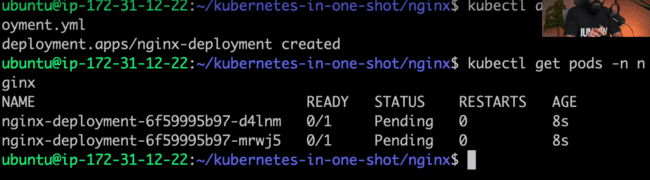
AI-generated content may be incorrect.

Now that volume which we have claimed needs to be assigned to that POD.

Now we need to add the volumes in the deployment.yml file. We need to map the local container path which is of ngnix “/var/www/html” to the HOST machine using the persistenetVolumeClaim: local-pvc

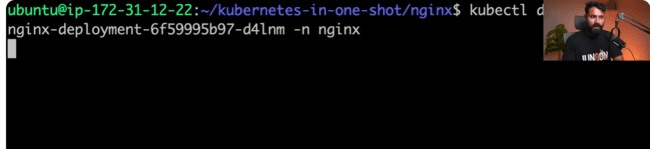


PODs created for the deployment.yml file:



If these are pending from a long time, so there might be some issue and we can describe them to check the issue.

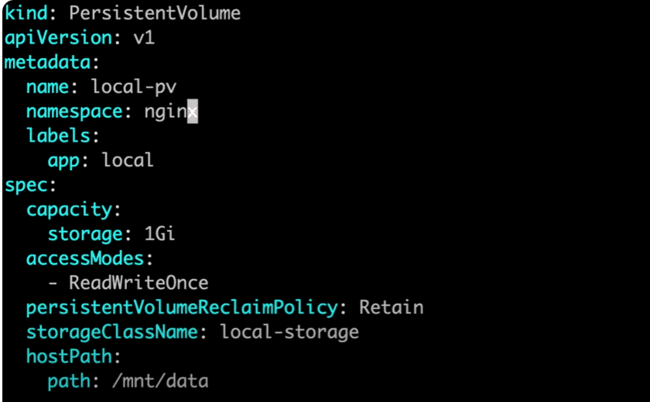
“kubectl describe <pod name> -n ngnix”



A screenshot of a computer screen

AI-generated content may be incorrect.

As the local-pvc and local-pv was in the local namespace, so we need to move that to the ngnix namespace:



A screenshot of a computer

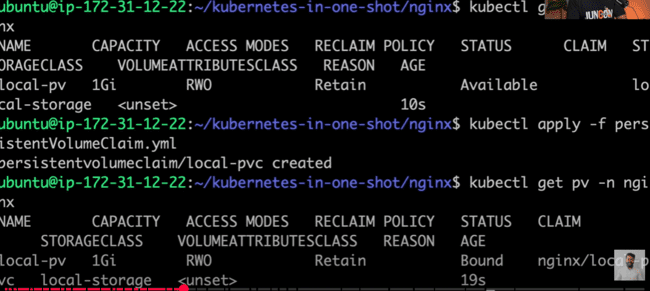
AI-generated content may be incorrect.

Deleting from default namespace:

A screen shot of a computer

AI-generated content may be incorrect.

Creating for the ngnix namespace:



Started running:

A screenshot of a computer program

AI-generated content may be incorrect.

Now, Describing the POD and checking the logs in that:

A screenshot of a computer screen

AI-generated content may be incorrect.

Now, we need to map the file from the POD folder to the host folder.

Checking using the wide command:

A screenshot of a computer program

AI-generated content may be incorrect.

The first POD is running on the “tws-cluster-worker” node.

Now this worker is not a node, but is a docker container.

Using the command “kubectl get nodes”:

A black screen with white text

AI-generated content may be incorrect.

Need to use the command “docker ps” and check the id and name of the container:

A screenshot of a computer

AI-generated content may be incorrect.

Need to go inside the container using the id of the container.



Using “ls” we’ll get the file structure:

A screenshot of a computer

AI-generated content may be incorrect.

Inside that /mnt/data:

A black screen with white text

AI-generated content may be incorrect.

Folder is created and data is bounder, but as we have not created any data, so that is not visible.

These PODs are created, but we are unable to see them:

A black screen with white text

AI-generated content may be incorrect.

So, we need to create the SERVICE PROXY for the deployment to access for the outside world:

There are two many PODs in one deployment and too many deployments available to service, so service redirect to the required deployment.

So, creating a new service.yml file:

A screen shot of a computer

AI-generated content may be incorrect.

Search for the Kubernetes service on the internet:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

“selector” is for the part for which service needs to be exposed under “spec.”

“port” is the port which needs to be accessed from the browser.

“targetPort” is the port which is the port of the container, or on which container is running.

“type” is the type of service which needs to be used in this part.

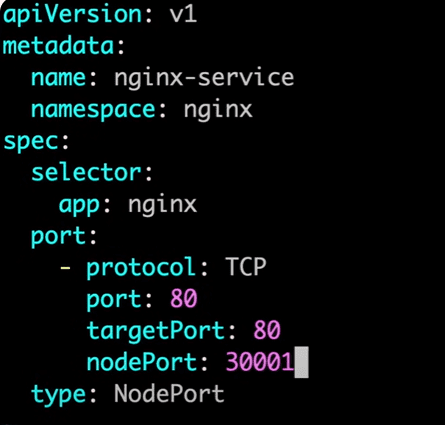
NodePort, ClusterIP, by default is this one, if none is defined, LoadBalancer, ExternalIP, HeadleessService.

If ClusterIP is used, then there would be one IP of the cluster which would be appear and there is also one port, through both of this, we can do the mappings.

A screenshot of a computer

AI-generated content may be incorrect.

If the NodePort is used, then we need use the port of the node which would be in between 30000 to 32000. Which needs to be used like this:

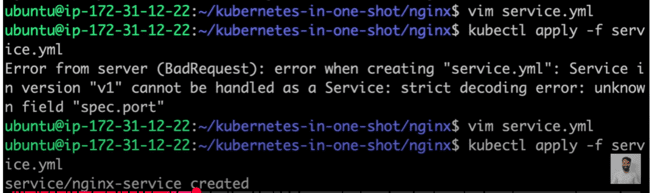


For ExternalIp, we can use the external IP and use that.

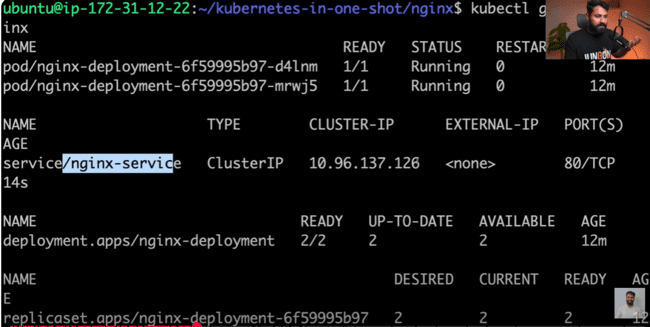
And Headless service is used with stateful sets.

LoadBalancer, is used for the cloud services.

“port” needs to be changed to “ports”, and then again need to apply the file:



Now if you want to see all, POD, service, deployment, then we need to use the command “kubectl get all -n ngnix”



Access the app on this IP:

A screenshot of a computer

AI-generated content may be incorrect.

Not working:

A screenshot of a computer

AI-generated content may be incorrect.

Cluster is a docker container and docker container port needs to be forward and IP needs to be exposed which is done in the following command using 80:80 for port forward and exposing the ip using the command –address=0.0.0.0:

A computer screen with white text

AI-generated content may be incorrect.

As 80 is already in use, we can use the 81:

A screenshot of a computer

AI-generated content may be incorrect.

Port 81 needs to exposed using the EC2:

A screenshot of a computer

AI-generated content may be incorrect.

Using the IP with port 81, ngnix would be loaded on the screen.

A screenshot of a computer

AI-generated content may be incorrect.

SMALL PROJECT:

Clone the app from github:

A screenshot of a computer screen

AI-generated content may be incorrect.

Checkout in the dev branch:

A screen shot of a computer program

AI-generated content may be incorrect.

Creating an image of the notes app using the command:

A screenshot of a computer

AI-generated content may be incorrect.

“docker images” command:

A screenshot of a computer

AI-generated content may be incorrect.

This image won’t run locally, so we need to send that to the Docker Hub.

Creating a token on docker hub.

A screenshot of a computer

AI-generated content may be incorrect.

Login using these commands:

A screenshot of a video

AI-generated content may be incorrect.

Login succeeded.

A computer screen with white text

AI-generated content may be incorrect.

Docker image needs to be tagged to the Docker hub using the command, tag as follows in which “docker image tag <OLD IMAGE> <NEW IMAGE>”:

A black background with white text

AI-generated content may be incorrect.

IMPORTANT: Kubernetes pulls the image when that is publicly available, but can also be pull private image when given access. So that’s why we push the image to the Docker Hub.

New image would be created:

A screenshot of a computer

AI-generated content may be incorrect.

Pushing the latest image to the docker hub as:

A screenshot of a computer

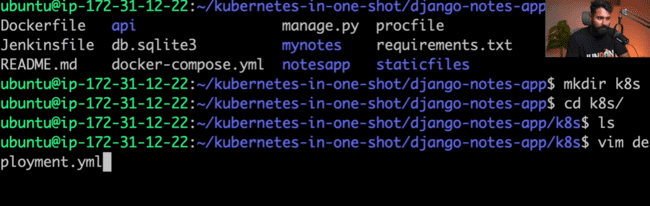
AI-generated content may be incorrect.

So, firstly, image is pushed from the local container to the Docker Hub, after that it needs to added to the POD as:

A black background with a blue hexagon and white text

AI-generated content may be incorrect.

Creating a new folder and creating a new deployment.yml file:



A screen shot of a computer

AI-generated content may be incorrect.

Image name would be that which has been pushed to the doker hub:

A screenshot of a computer

AI-generated content may be incorrect.

And this application runs on port 8000:

A screenshot of a computer

AI-generated content may be incorrect.

Creating a namespace.yml file:

A black background with a black square

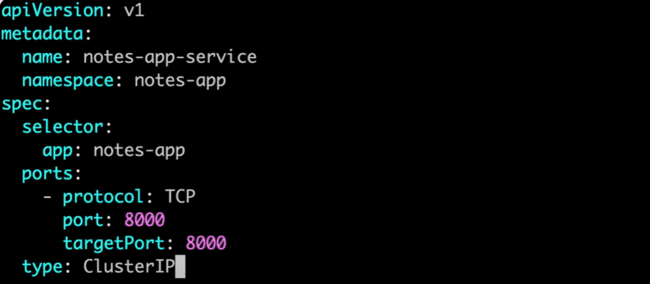
AI-generated content may be incorrect.

And adding the same namespace in deployment.yml file:

A black background with white text

AI-generated content may be incorrect.

Creating a service.yml file:

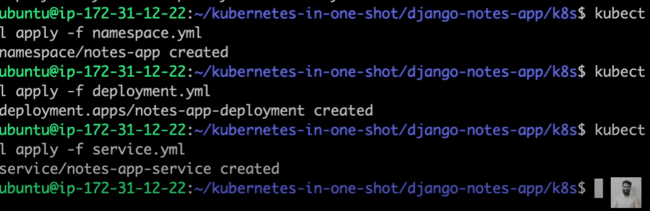


Now deployment, namespace and service are created:

A black screen with white text

AI-generated content may be incorrect.

And applying all of them:

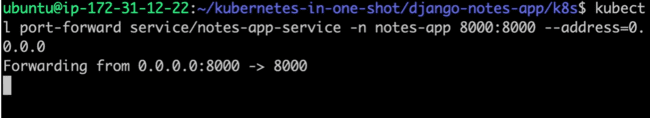


Getting nodes:

A black screen with white text

AI-generated content may be incorrect.

Forwarding the port and exposing the IP:



Adding the port with IP:8000 in the EC2 instance:

**A screenshot of a computer

AI-generated content may be incorrect.**

App is accessible:

A screenshot of a computer

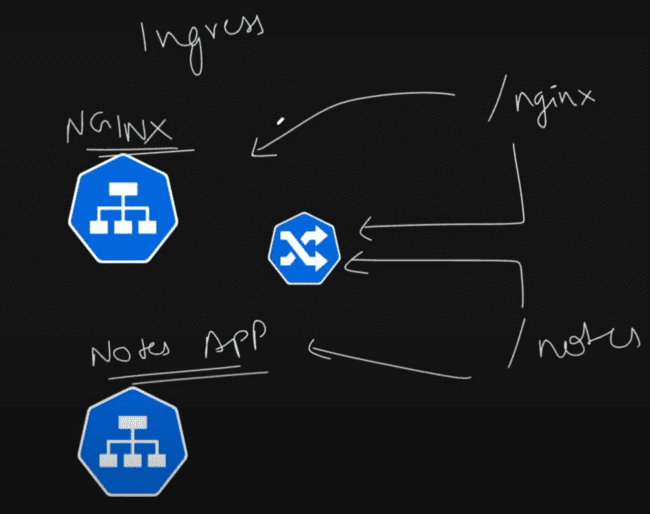
AI-generated content may be incorrect.

INGRESS:

Now if we want, that IP:8000/ngnix would be redirected to ngnix and IP:8000/app would be redirected to app, then we need to use the NGNIX.

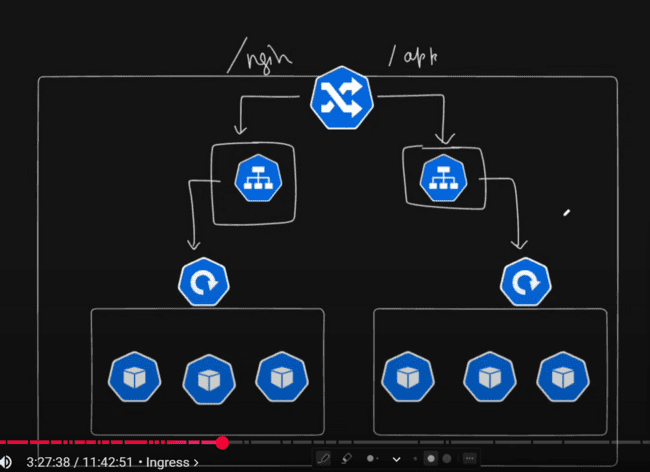
This is called redirection of the services.

Example: we have two services, one of which is ngnix and other is nodes app. And /ngnix would redirect to ngnix service and /app would redirect to nodes service. So we need to use the INGRESS, which is used for the traffic rerouting for the cluster.



The high level diagram would be like: there is one big cluster and it has multiple PODs and these PODs have been managed using the deployment. And to access that deployment, we need to use one service. And in the same way, there is another cluster with multiple PODs, which have a deployment and a service to access them.

Now to access these services using the name, we need to use the INGRESS which is basically used to managed the traffic and routes.



And all the above would be in the same namespace.

For this one, we need to use the same namespace as “ngnix” in the deployment.yml and service.yml file as shown:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Deleting the notes-app deployment and service which has been created previously:

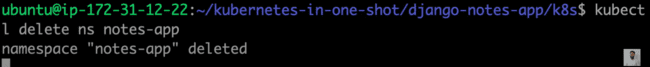
A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

And deleting the notes app namespace as well:

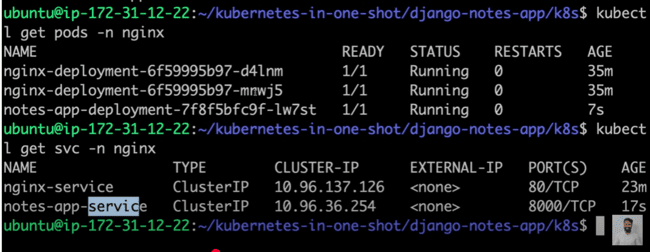


Creating the PODs and services using same apply command for the multiple .yml files:

A screen shot of a computer

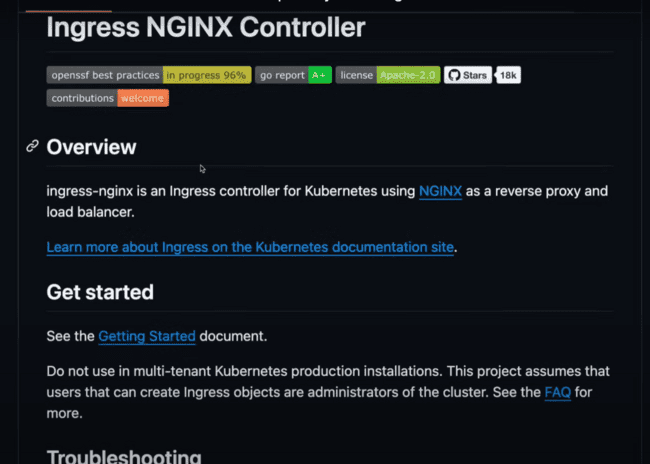
AI-generated content may be incorrect.

Now we have ngnix and notes app service and deployment:



We need to create the INGRESS controller for the ingress.

There is one available on the github, the ingress controller which is available, that helps in the reverse proxy and load balancer.

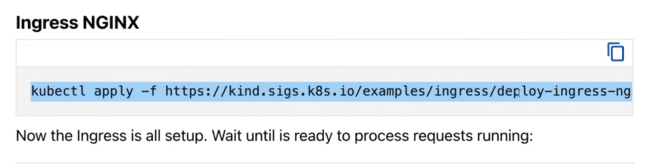


We need to create the same for the kind cluster:

A screenshot of a computer

AI-generated content may be incorrect.

Applying the file of the ingress in the home directory:



A black screen with blue text

AI-generated content may be incorrect.

It starts getting created:

A screen shot of a computer program

AI-generated content may be incorrect.

In the namespace, if you check ingress-ngnix would be created:

A screenshot of a computer

AI-generated content may be incorrect.

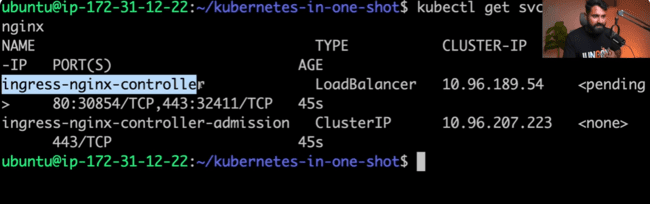
PODs in the ingress-ngnix namespace:

A screenshot of a computer

AI-generated content may be incorrect.

The “Completed” status is the job which gets completed and POD is in the “Running” state.

Services in the ingress-ngnix namespace:



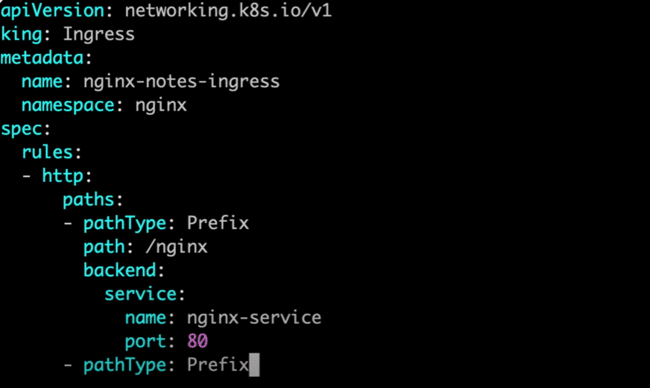
Now we need to create a new file in the ngnix directory as ingress.yml:

“apiVersion” is the version which would be there in the website:

A white background with black and red text

AI-generated content may be incorrect.

Under “rules” we need to define the paths for which we need to redirect to under the http or https. Then we need to redirect to which service would be annotated by “backend” tag.



A screenshot of a computer program

AI-generated content may be incorrect.

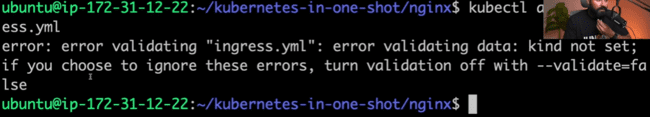
Applying the ingress.yml file:

A black screen with white text

AI-generated content may be incorrect.

We need to change the “king” to “kind” in the .yml file.

After that ingress created and getting the ingress service:



Getting all the things which are running in the namespace ngnix:

A black screen with blue text

AI-generated content may be incorrect.

A screenshot of a computer

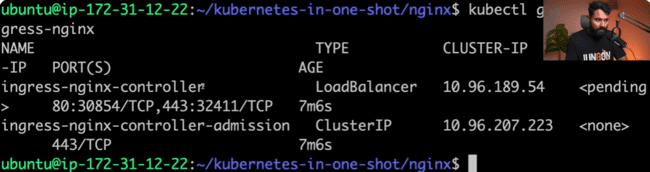
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

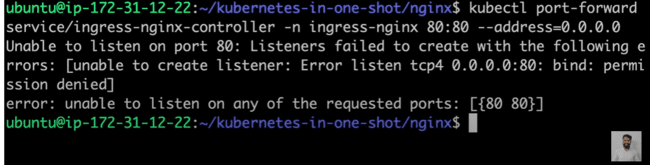
Now, exposing the ingress-ngnix controller.

Getting the service in the ngnix namespace:



This ingree-ngnix-controller needs to be exposed.

So we used the following command for this one:



So, for that we use the command as “sudo E”, but port is already bind:

A screen shot of a computer error

AI-generated content may be incorrect.

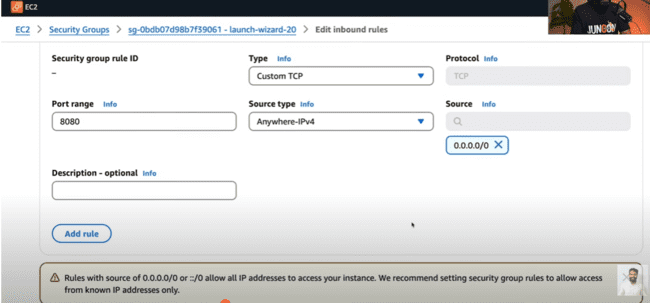
We use the port 8080:80 instead of 80:80 in the same command:

A black screen with white text

AI-generated content may be incorrect.

Now in the port 8080, your ingress controller is running.

Adding the same port 8080 in the EC2 instance:



65.0.99.9:8080/app, the app is running, but page is not displaying:

A screenshot of a computer

AI-generated content may be incorrect.

65.0.99.9:8080/ngnix, page not found is there:

A screenshot of a computer

AI-generated content may be incorrect.

So, we need to change in the ingress.yml file as:

In place of “path: /app” we’ll change to “path: /” and then test the same:

A screen shot of a computer

AI-generated content may be incorrect.

Again, applying the ingress.yml and run again:

A screen shot of a computer program

AI-generated content may be incorrect.

In the port 8080:/, app is loading as:

A screenshot of a computer

AI-generated content may be incorrect.

But ngnix is not opening:

A screenshot of a computer

AI-generated content may be incorrect.

This is due to most of the applications run on the slash as http:IP:80/,

As, mostly port 80 is considered as the invisible port, as in the following IP address, 80 port would be replaced by slash “/” as shown below:

A blackboard with green writing on it

AI-generated content may be incorrect.

So, now for the slash, notes app is opening, but for the “/ngnix”, ngnix app needs to be opened also, for which we need to use the annotations.

Like we need to rewrite the slash “/” in the Kubernetes.io as whenever / is there, as :80/ would be considered as /, so it would redirect to port 80 in ngnix.