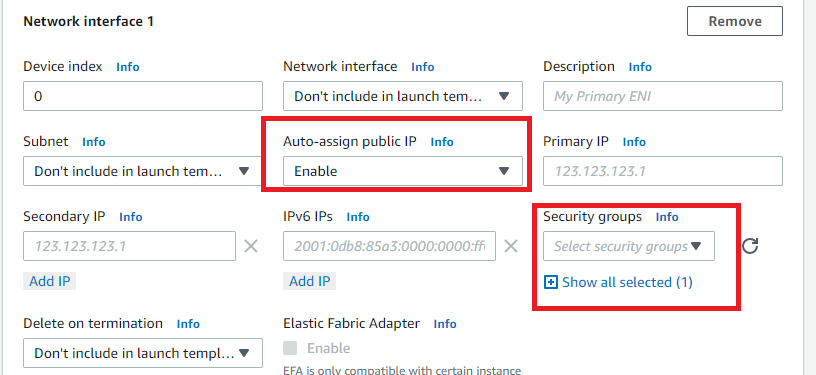
**AWS KEY POINTS**

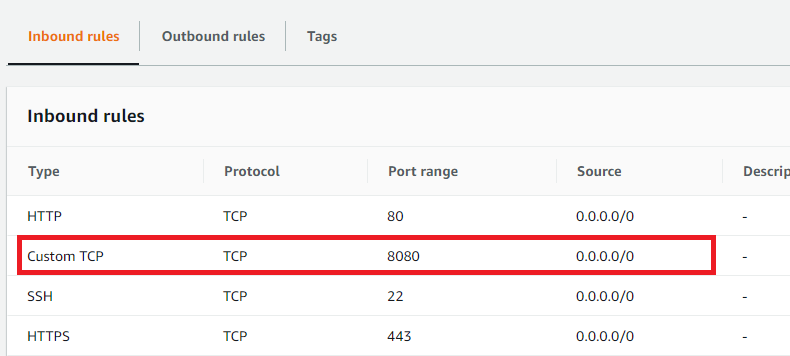
If you want to get ELB and Load balancing to work in auto scaling and avoid the “502 gateway error” you need to allow the “Auto assign IP” on your Template and also assign the security.

Make sure you use the same security group as the Load balancer security group and you use the load balancer DNS addresss



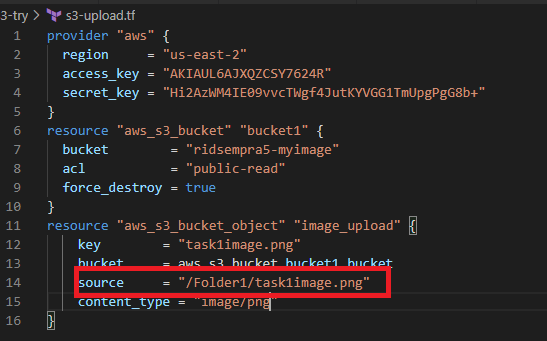
IF you have 2 servers installed on a server (example and ubuntu instance)

1. You have installed ngnix for AWS
2. Now you want to install Jenkins, since you would have already install ngnix on port 80, so now we have to install Jenkins on anpther port can be any number but we using port 8080 , so in this case we log onto the web with “<http://3.129.83.111:8080/>”
3. We must also activate the security group of the Instance to allow port 8080



**TERRAFORM AWS S3 UPLOAD**

To upload you have specify a source path



**GITHUB STEPS TO MOVE SCRIPT FROM TERRAFORM TO GITHUB**

**git clone https://github.com/rid432/terraformcode.git**

**echo "# devops2" >> README.md**

**git init**

**git add README.md**

**git commit -m "devops2"**

**git push -u origin master / git push origin <branchname>**

1. Create a folder where you want your repository to store
2. Create a repository
3. Now we copy the clone of the repository and run it in terraform 🡺 we get the address by click code then copy the text in the drop down
4. git clone <repository address> example git clone <https://github.com/rid432/terraformcode.git>
5. Now you “cd” into the directory
6. we run the step by step that shows under the repository
7. git init 🡺 this command is run every time you create a repository
8. git add README.md 🡺 this is to add a read write
9. git commit -m "adding Readme file” 🡺 this is a comment/description always good practice to comment on every action
10. Now you can copy your “\*.tf” files from your terraform folder to newly created github repository
11. git status 🡺 check the status of the file you want to add on git
12. git add . 🡺 adds all the file showed in red with “git status” and this can also be done if you use “git add <filename>”
13. Example git add iam.tf 🡺 git commit -m “terraform code to add iam\_user”
14. make sure you always add a comment after your every command for this we use 🡺 git commit -m “comment”
15. git push -u origin master / git push origin <branchname>

**Now Moving script into a Branch and not master like the above**

1. Create a folder where you want your repository to store
2. Create a repository
3. Now we copy the clone of the repository and run it in terraform 🡺 we get the address by click code then copy the text in the drop down
4. git clone <repository address> example git clone <https://github.com/rid432/terraformcode.git>
5. we run the step by step that shows under the repository
6. git init 🡺 this command is run every time you create a repository
7. git add README.md 🡺 this is to add a read write
8. git commit -m "adding Readme file” 🡺 this is a comment always good practice to comment on every action
9. git checkout -b <branchname> 🡺 here we are creating a branch and also telling it to checkout of master
10. git checkout <branchname> or git checkout <master> 🡺 here you can switch between master and branch
11. now we switch to a branch called “testbranch” as an example
12. git checkout testbranch 🡺 now whatever we run is under branch “testbranch”
13. Now you can copy your “\*.tf” files from your terraform folder to newly created github repository
14. git status 🡺 check the status of the file you want to add on git
15. git add . 🡺 adds all the file showed in red with “git status” and this can also be done if you use “git add <filename>”
16. Example git add iam.tf 🡺 git commit -m “terraform code to add iam\_user”
17. make sure you always add a comment after your every command for this we use 🡺 git commit -m “comment”
18. git push origin <branchname> 🡺 in this case we use git push origin testbranch

To pull the modification code to master to be seen on master

1. We go to our terraform and add “git pull” 🡺 to pull the latest modification onto the master

**See activity performed in your git, we check the log :**

1. git log 🡺 this shows all the activity performed in that repository
2. git log –oneline 🡺 this shows all the activity performed within that repository in oneline

**git restore 🡺 we use this command to undo “uncommitted changes”**

Say you modify your code and you haven’t committed it or ran it, you can easily undo it with the command

git restore <codename> 🡺 for one file

git restore . 🡺 if you want to undo all the file in that repository branch or master where you making the change

**git revert 🡺 we use this command to undo “committed changes”**

1. We run git log –oneline 🡺 so we can get the commit Id of the file we want to revert

Here we want to revert the “new EC2” back to just the “add EC2” we added earlier since we decided we don’t want to move forward the that code , all we need to is revert the new latest committed file

talkt@RID MINGW64 ~/OneDrive/Desktop/AWS/terraform/git\_terra\_project/devops (rid)

$ git log –oneline

f7884c4 (HEAD -> rid) new EC2

229b623 (origin/rid) adding iam

11ed620 to add EC2

10aa6d5 (master) first commit

1. Now we run git revert f7884c4🡺 we revert this commit change back to how it was 🡺 a screen will show “ just type (:q!)”
2. This leaves us with only “add EC2” and delete the newly added change i.e “new EC2”

**Git Reset**

If you have made so many changes and you want to reset your changes to one of the previous codes we use git reset <commit id>

git log –oneline 🡺 to list the set of commit available

now we choose the commit id we want to use , see example below

talkt@RID MINGW64 ~/OneDrive/Desktop/AWS/terraform/git\_terra\_project/terraform (master)

$ git log –oneline 🡺 **Here we view the list of commits**

699417f (HEAD -> master) Merge branch 'master' of https://github.com/rid432/terraform

be5f51a (origin/master) Merge pull request #3 from rid432/ridtest

7ada401 (origin/ridtest, ridtest) Revert "we trying to go now"

b1ef7fa Merge branch 'master' of https://github.com/rid432/terraform

bb0894c Merge pull request #2 from rid432/ridtest

fc56c37 we trying to go now

eb449bc Merge branch 'master' of https://github.com/rid432/terraform

298ce6a reversing

0da97e2 Merge pull request #1 from rid432/ridtest

c1086ff adding comments

5bbbdf4 (origin/ridbranchterra, ridbranchterra) pushing terraform file

talkt@RID MINGW64 ~/OneDrive/Desktop/AWS/terraform/git\_terra\_project/terraform (master)

$ **git reset --hard** fc56c37 🡺 we are Reseting all our changes back to a old date and that is “we trying to go now”

HEAD is now at fc56c37 we trying to go now

talkt@RID MINGW64 ~/OneDrive/Desktop/AWS/terraform/git\_terra\_project/terraform (master)

$ git log –oneline 🡺 if you check this log we only have the logs we had before this change all, the logs after have been removed

fc56c37 (HEAD -> master) we trying to go now

c1086ff adding comments

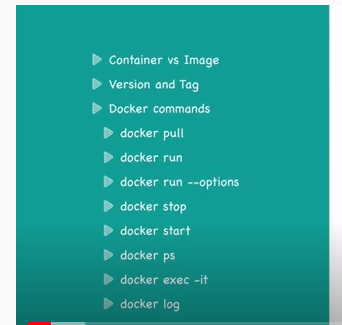
5bbbdf4 (origin/ridbranchterra, ridbranchterra) pushing terraform file

talkt@RID MINGW64 ~/OneDrive/Desktop/AWS/terraform/git\_terra\_project/terraform (master)

$

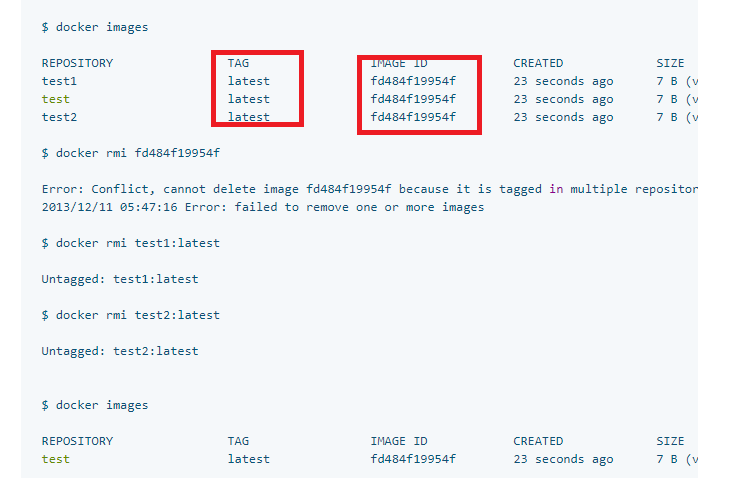
git reset –hard

****



**Container** 🡺 It’s a running env of an Image

**How to delete an Image in multiple Repository**

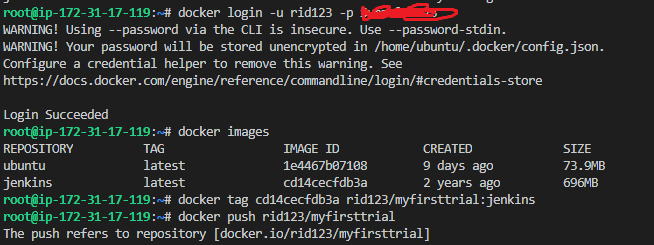


docker rmi -f <imageid> 🡺 forcefully removing an image

**TO PUSH AN IMAGE TO YOUR DOCKERHUB REPOSITORY**

1. We log into our repository with our dockerhub repository username and password = docker login -u <username> -p <password>
2. If we see the login succeded then we login
3. docker tag <image\_id> yourhubusername/<repositorynameyoucreatedindockerhub>:<tagnameforyourimage>
4. docker tag 1e4467b07108 rid123/myfirsttrial:ubuntu 🡺 so I created a repository with the name “myfirsttrial” and I try to move the ubuntu image to it == then you now push it into the repository
5. We can also use🡺 docker tag <repositoryname:tag> rid123/myfirsttrial:ubuntu 🡺

example : docker tag Jenkins:latest rid123/myfirsttrial:jenkins



**Note: always run docker with “-d” so the container will run on the background**

root@ip-172-31-22-254:~# docker run -d --name jenk -p 8080:8080 -p 50000:50000 jenkins

6b1e549893e1918d66ed38f855bec5c75dc01916dcd91792f1c2768476ad3685

To Run a docker ubuntu image 🡺 we use the command

“root@ip-172-31-22-254:~# docker run -it ubuntu

root@0d5775928e46:/# 🡺 it automatically logs you int the ubuntu sever

**Runing a docker application with an assigned port**

docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts

**To execute any command within a running container**

If you already have ur container running and u want to execute any command within that container we use the syntax:

**docker** exec -it <**container** name> <**command**>  🡺**to execute** whatever **command you** specify **in** the **container**.

docker exec -it <container\_id\_or\_name> echo "I'm inside the container!"

docker exec -it d15de3d365c1 cat /var/jenkins\_home/secrets/initialAdminPassword = 9e33bbe0cce54fba8280fa3a4b0ea35b

**Run a docker image for mysql**

docker run –name <nameofcontainer> -e MYSQL\_ROOT\_PASSWORD=<password> -d <image>

docker run --name my-db -e MYSQL\_ROOT\_PASSWORD=db-password -d mysql

docker run --name my-wordpress -p 8080:80 --link my-db:mysql -d wordpress

docker run --name my-db -e MYSQL\_ROOT\_PASSWORD=db-password -d rid123/wordpressweb:mysql

docker run --name my-wordpress -p 9090:80 --link my-db:mysql -d rid123/wordpressweb:wordpress

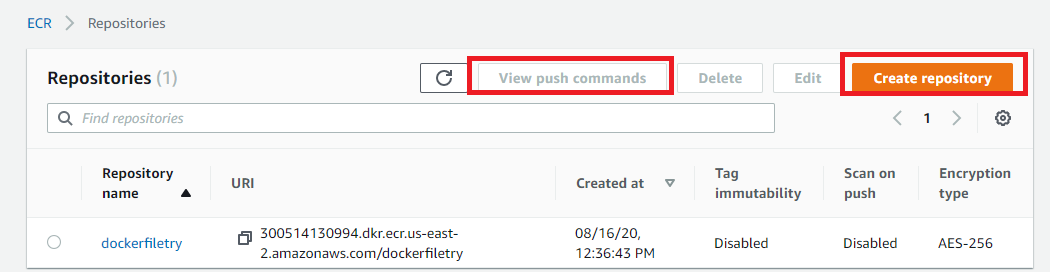
**Docker Troubleshooting commands**

1. docker logs <runningcontainerID> or docker logs <nameofcontainer>

**TO PUSH AN IMAGE TO YOUR ECR (EC2 CONTAINER REGISTER)**

This is the same like you pushing to your docker hub/ repository , the only different is you pushing into your EC2 ECR in the cloud.

1. Go to your ECR service and create a repository
2. In this example we created a repository called “dockerfiletry”
3. We click on the view push command to see the commands needed to push the repositorys
4. apt install awscli 🡺 we first need to install awscli on our servers
5. aws configure 🡺 to login with our accesskey and secretkey
6. Build the image we intend to build with a dockerfile 🡺 docker build -t sixtthdockerfile -f Dockerfile6 .
7. Tag the file 🡺 docker tag sixtthdockerfile:latest 300514130994.dkr.ecr.us-east-2.amazonaws.com/dockerfiletry:sixtthdockerfile
8. docker push 300514130994.dkr.ecr.us-east-2.amazonaws.com/dockerfiletry 🡺 we push the file



**Steps for running a container:**

1. Create a ECR and push your container into it
2. Create a TASK DEFINITION 🡺 Just fill the name 🡺 select “ADD CONTAINER” 🡺 copy the Image URI from the ECR
3. Containers parameters : Hard Limit = set it has default , Port mapping 80:80, CPU = 1024 🡺 click “ADD” 🡺 Click create
4. Create a cluster under “Amazon ECS”
5. Click on View cluster
6. Now we create a service

**Docker file**

Here are all the commands that we can use in the Dockerfile.  
Comments  
FROM

MAINTAINER 🡺 person who manages the container (must come immediately after the FROM command)

RUN 🡺 Help build the docker file  
CMD 🡺 used after a container has been built  
ENTRYPOINT 🡺 can override a CMD command, it’s the first command to be run after a container has been built  
WORKDIR 🡺 This is used to set where the command defined with CMD will be executed e.g WORKDIR /path WORJDR ~}  
EXPOSE🡺 Help specify which port to run a container 🡺 e.g EXPOSE 8080  
ADD 🡺 copy a fine from a source to a container destination e.g ADD {Source directory or URL} {destination directory}  
.dockerignore  
ARG  
  
USER 🡺 specifying a user to run a command in the container with a userid e.g USER 751  
VOLUME🡺 where we want the container to store all the files e.g VOLUME [\*/my\_files”]

Example of a dockerfile

# from base image node

FROM ubuntu

RUN apt-get update

RUN apt-get install -y openjdk-8-jre

Example2

# from base image node

FROM ubuntu:18.04

MAINTAINER ridwan

RUN apt-get update && apt-get install openjdk-8-jre && update-alternatives --config java && apt-get install jenkins

EXPOSE 8080

Note== If you are installing anything in Dockerfile you need the “-y” to enable it continue as you wont be able to type in the “yes” option , so syntax to install is “install -y”

**wget in Dockerfile**

You need to install it first. Create a new Dockerfile, and install wget in it and also gnup2 before you can use the wget commonda

FROM ubuntu:14.04

RUN apt-get update \

RUN apt-get install -y gnupg2

&& apt-get install -y wget \

&& rm -rf /var/lib/apt/lists/\*

**Dockerflie for Java**

RUN apt-get install -y --no-install-recommends software-properties-common

RUN add-apt-repository -y ppa:openjdk-r/ppa

RUN apt-get update

RUN apt-get install -y openjdk-8-jdk

RUN apt-get install -y openjdk-8-jre

RUN update-alternatives --config java

RUN update-alternatives --config javac

**Dockerfile for Jenkins**

# from base image node

FROM ubuntu:18.04

RUN apt-get update && apt-get install -y openjdk-8-jre

RUN update-alternatives --config java

RUN apt-get install -y gnupg2

RUN apt-get install -y wget

RUN wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | apt-key add - 🡺 to run “wget” it must be installed first

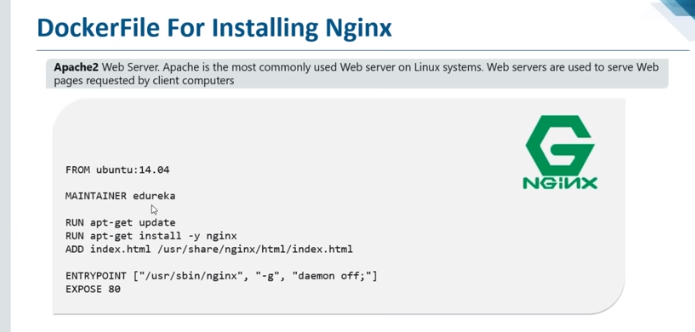
RUN apt-get install -y software-properties-common

RUN apt-add-repository "deb [https://pkg.jenkins.io/debian-stable binary/](https://pkg.jenkins.io/debian-stable%20binary/)" 🡺 to run “app-add-repository” we need to run “apt-get install -y software- properties-common” first

RUN apt-get install -y jenkins

EXPOSE 8080

Jenkins 🡺 is a CICD that helps create a pipeline which create an Artifact, used to deploy an application



Comments  
Comments in the dockerfile start with # and you can put anywhere those comments.  
# from base image node  
FROM  
This is the first command in the Dockerfile. Without this, we can’t build an image. We can build the image just with this command. when we build just with FROM, we are actually taking the base image CMD whenever the image is instantiated.

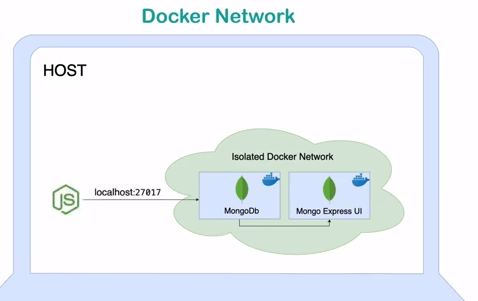
<https://gist.githubusercontent.com/bbachi/32ccfc3ebb3cdb7f66cce2e21a3377c3/raw/fb4213b9742225cfaf417662e8b9d130b3f8f8c1/Dockerfile>

// build the image  
docker build -t myfirstimage -f Dockerfile1 . 🡺 the build doesn’t override but only duplicate   
// list image  
docker images  
// run the image  
docker run -it -d first-dockerfile  
// use exec for interaction  
docker exec -it f1edbfca3eac bash

docker build -t myfirstimage:2.0 -f Dockerfile1 . 🡺 using tags to help differentiate what image container been run

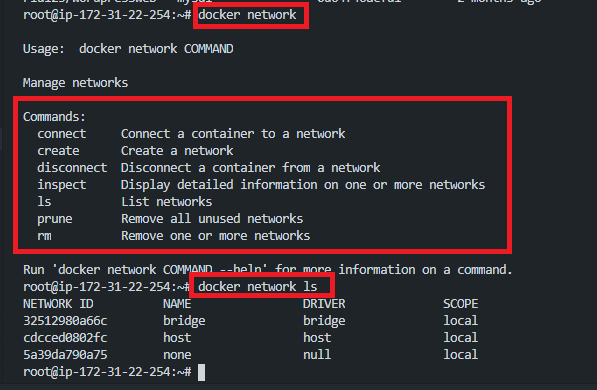
**Docker Network**

Docker deploy “Isolated Docker Network” where the container is running in 🡺 i.e when we deploy 2 different docker container within the same Docker Network, they will be able to communicate with each other. Example : Mongo and Mongo express can talk to each other just by using the Container Name

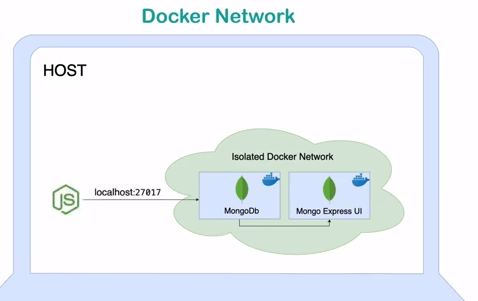


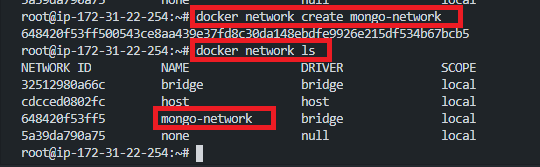
The Node.JS will talk to the 2 apps running in the Isolated Network with an IP associated with a port number.

Docker by default already has Networks 🡺 to view the default networks running in docker we do “docker network ls”



**To create a Docker Network : For Instance we want to create a Mongo db Docker Network :**





FYI: When you run a docker container with example 🡺 **docker run -p 27017:27017 -d mongo** 🡺 this means run the docker image on the host with port 27017 and also in the container with 27017

**To enable the Mongodb and Mongo-Express communicate with each other via Mongo-Network we perform the steps below**

Default port for Mongo DB = 27017

**Mango-db:**

root@ip-172-31-22-254:~# docker run -d \ #run docker image in background#

> --name mongodb -p 27017:27107 \ #provide a name for the container (e.g mongodb) and run it on port 27017:27017#

> -e MONGO\_INITDB\_ROOT\_USERNAME=admin \ #always create a username and password for a db#

> -e MONGO\_INITDB\_ROOT\_PASSWORD=password \ #always create a username and password for a db#

> --net mongo-network \ #now run the container within the already created docker network (called mongo-network)#

> mongo #image#

**Mango-express:**

root@ip-172-31-22-254:~# docker run -d \ #run docker image in background#

> --name mongoexpress -p 8081:8081 \ #create a name for the container (e.g mongoexpress) run it on port 8081:8081#

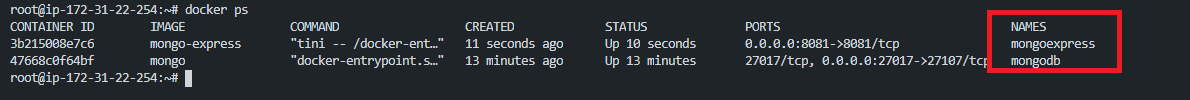
> -e ME\_CONFIG\_MONGODB\_ADMINUSERNAME=admin \ #always create a username for mongo-express must be same with mongodb userna#

> -e ME\_CONFIG\_MONGODB\_ADMINPASSWORD=password \ #always create a password for a mongo-express must be same with mongodb#

> --net mongo-network \ #now run container within already created docker network (called mongo-network), to enable mongodb&&mongoexpress talk#

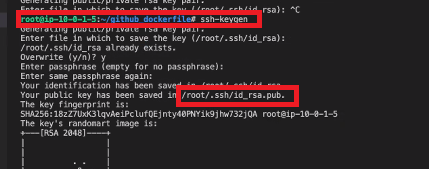
> -e ME\_CONFIG\_MONGODB\_SERVER=mongodb \ #run the mongoexpress within the already running mongodb container called “mongodb” as above#

> mongo-express #image#

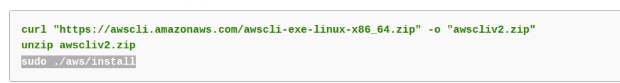


**IF we run git push login command and it returns an error:**

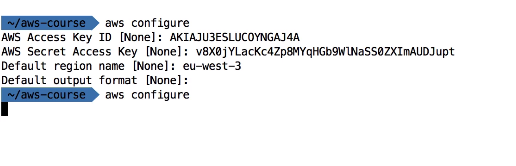
1. We have to generate on our server by running “ssh-keygen” this generate a private and public key, then we **cat /root/.ssh/id\_rsa.pub** and we copy the key and paste it in our github i.e 🡺 setting 🡺 ssh

cat

**TO INSTALL “aws cli” on Linux**

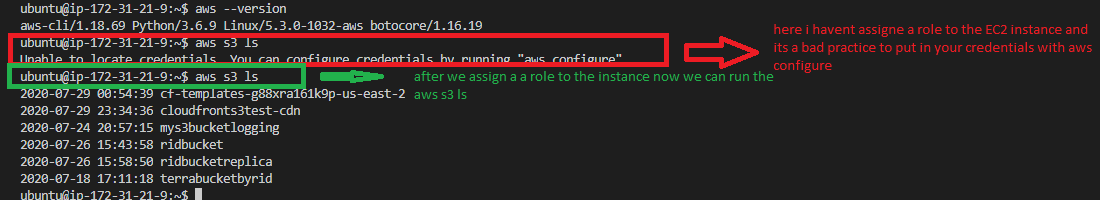


**Aws configure**



AWS S3 CLI 🡺 TO KNOW MORE AWS S3 COMMANDS, WE TYPE ON GOOGLE WITH “**aws s3 cli”**

Never use “aws configure” directly on your instance, its always advisable to use roles



aws s3 ls 🡪 list the files in the s3 bucket

ubuntu@ip-172-31-21-9:~$ aws s3 ls s3://ridbucket 🡺 Listing the content in the specific bucket “s3://ridbucket”

2020-07-26 16:05:40 24769442 AWS Certified Cloud Practitioner Slides v1.3.pdf

2020-07-26 15:46:09 1209033 Auto-scaling.pdf

2020-07-26 16:05:41 171 index.html

2020-07-26 16:05:41 171 index2.html

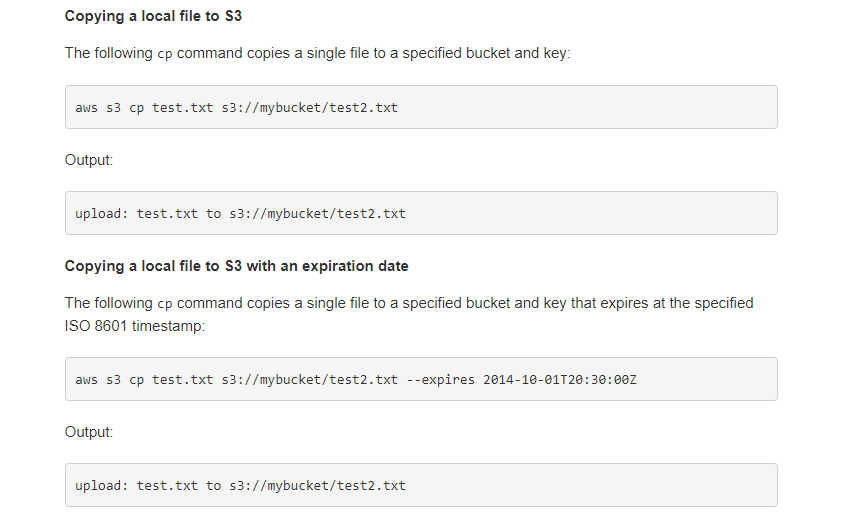
2020-07-26 16:05:41 199289 man\_world\_mart-1533225685-9306.jpg

aws s3 mb s3://<bucket\_name>> 🡺 making a bucket

aws s3 rb s3://<bucket\_name>> 🡺 Remove bucket

**to download a file from s3 bucket using the CLI**

aws s3 cp aws s3 ls s3://ridbucket/ Auto-scaling.pdf Auto.pdf 🡺 this is actually downloading the file or copying a file from the s3 bucket named “ridbucket” to your local machine/instance, now we renamed the file to “Auto.pdf”



**AWS S3 PRESIGN URL**

If we have an encryption then we have to run this command first : aws configure set default.s3.signature\_version s3v4

aws s3 presign s3://ridbucket/man\_world\_mart-1533225685-9306.jpg --expires-in 300 --region us-east-2

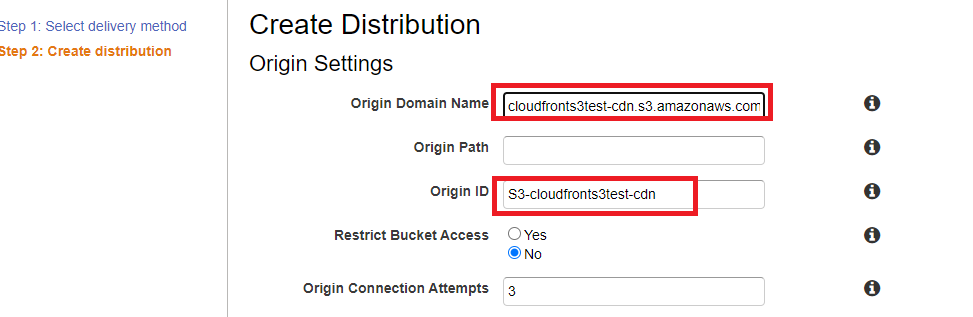
**AWS METADATA**

This is used to get information about your roles or some settings , that’s the general IP below

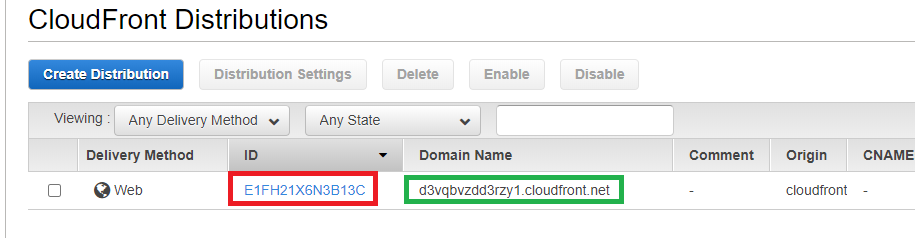


***AWS CLOUD FRONT***

1. You create an S3 bucket and make it public
2. Create a cloud front
3. Select the bucket you initially created

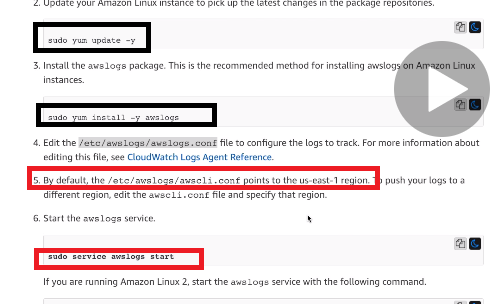


1. This takes a while to create now we copy the name in domain name to our web browers
2. d3vqbvzdd3rzy1.cloudfront.net\<name of the object uploaded> example 🡺 d3vqbvzdd3rzy1.cloudfront.net\coffe.jpg
3. If it did not open or access deny then we make the object public because normally it should work but we have to wait for 3hrs for DNS to help or propagate resolve this issue but since we cant wait hence we made it public
4. You notice we are now checking the content of our s3 bucket through cloudfront and not the s3 directly



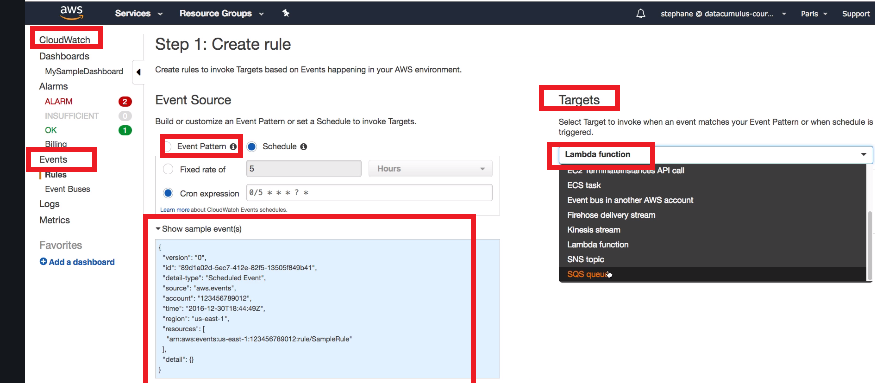
**TO ENABLE CLOUD WATCH LOGS**

1. Create an EC2 instance
2. Use and IAM role to connect EC2 to a Cloudwatch service
3. Connect to the instance VIA SSH
4. Then we run this command below on the EC2 instance

****

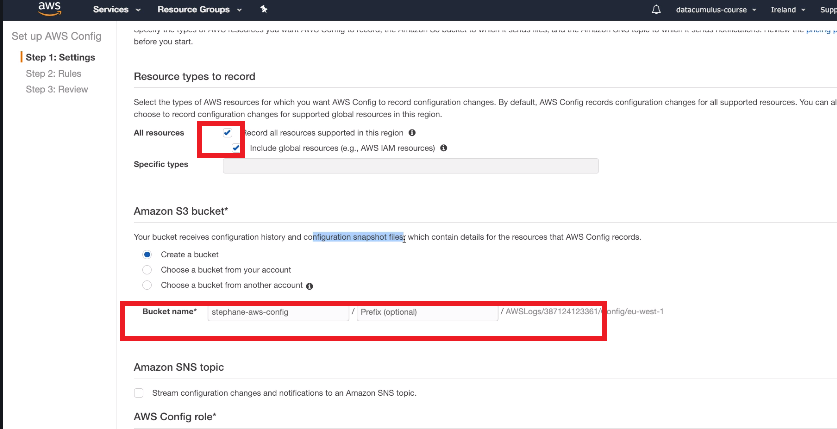
**CLOUD WATCH EVENT**

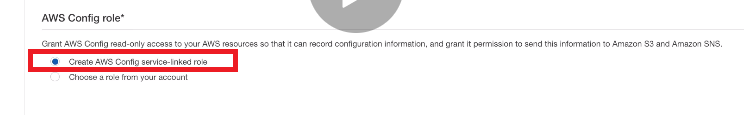
1. Cloudwatch 🡺 Click Event
2. We select event rate also select the Target to invoke when the event is triggered

****

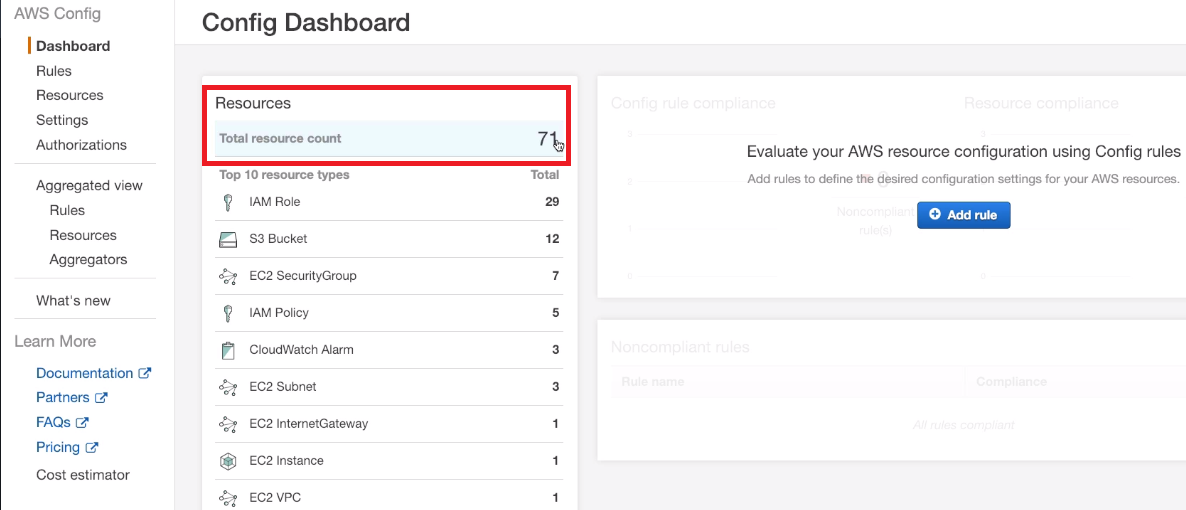
**AWS CONFIG**

1. Here we create AWS Config , we check the 2 boxes
2. And also attach it to a bucket which we create or already created
3. Then we click create

****

****

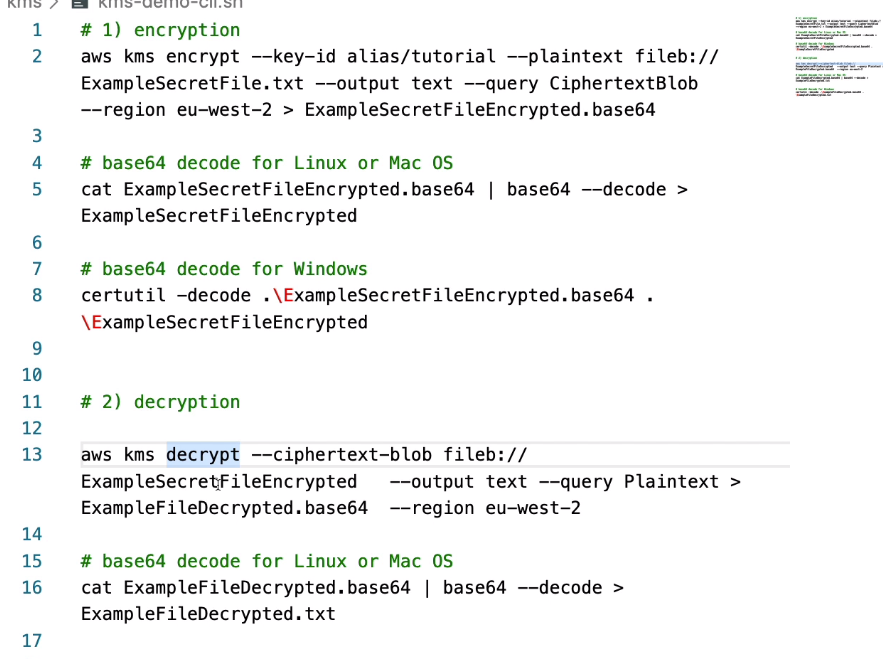
1. After creation we can see the number of resources we have, here we have 71 resources

****

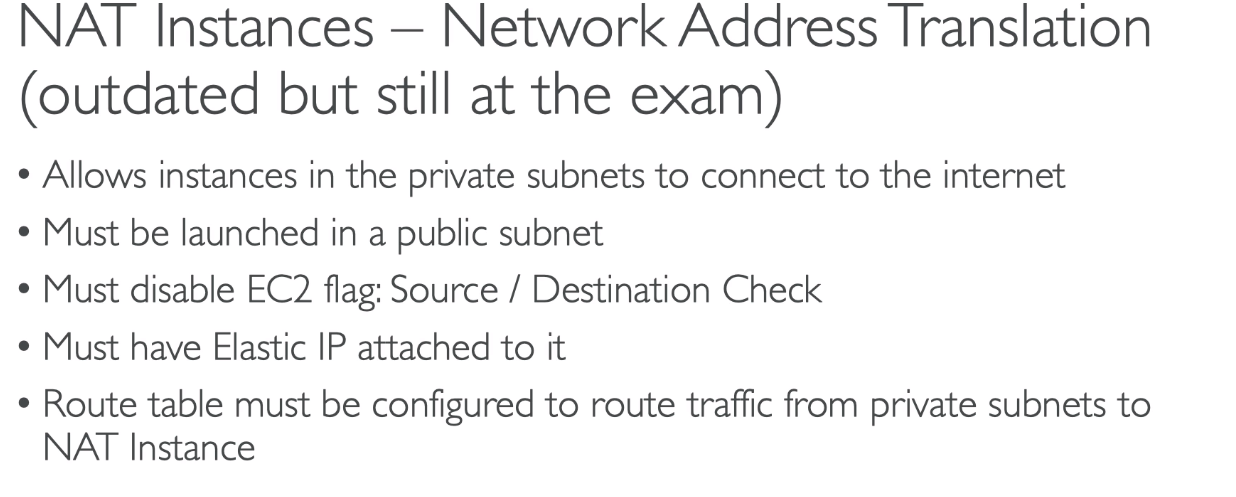
* We can check the changes made for each Rule by checking the Configuration Timeline and Compliance Timeline

**After YOU CREATE YOUR AWS KMS CONFIGURATION ON THE GUI,**

1. Below is a set of command you use on AWS CLI to encrypt and decrypt a test file
2. For this we use an example text file called “ExampleSecreatFile.txt” here we have our secret word in it

****

**VPC🡺 NAT INSTANCE**

****

**CREATE A GROUP WITH AWS CLI**

**Iam commands:**

aws iam create-group --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonRoute53FullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/IAMFullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonVPCFullAccess --group-name kops

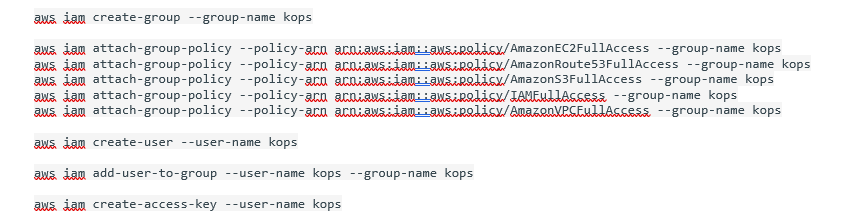
aws iam create-user --user-name kops

aws iam add-user-to-group --user-name kops --group-name kops

aws iam create-access-key --user-name kops

**KUBENETES INSTALL**

1. Update your server = sudo apt-get update
2. Install aws cli =  apt-get install awscli
3. Run aws configure 🡺 create a user with a key to log into your aws console
4. Run the commands below :



1. aws s3api create-bucket --bucket <kopsridwan> --region us-east-2 --create-bucket-configuration LocationConstraint=us-east-2
2. Run this command below:

export NAME=myfirstcluster.example.com

export KOPS\_STATE\_STORE=s3://prefix-example-com-state-store

or

fa

export NAME=rid-kops.k8s.local

export KOPS\_STATE\_STORE=s3://kopsridwan

1. Now you install kubenetess with Kops, see the command below :
2. curl -LO <https://github.com/kubernetes/kops/releases/download/$(curl> -s <https://api.github.com/repos/kubernetes/kops/releases/latest> | grep tag\_name | cut -d '"' -f 4)/kops-linux-amd64
3. chmod +x kops-linux-amd64
4. sudo mv kops-linux-amd64 /usr/local/bin/kops
5. kops version

aws configure list 🡺

cat **~/.aws/credentials 🡺 only when you are in root**

**Work on Global Accelerators**

Kubenetes installation

**# Kubernetes on AWS using Kops### 1. Launch Linux EC2 instance in AWS (Kubernetes Client)**  
**### 2. Create and attach IAM role to EC2 Instance.**  
    Kops need permissions to access  
        S3  
        EC2  
        VPC  
        Route53  
        Autoscaling  
        etc..  
**### 3. Install Kops on EC2**

curl -LO <https://github.com/kubernetes/kops/releases/download/$(curl> -s <https://api.github.com/repos/kubernetes/kops/releases/latest> | grep tag\_name | cut -d '"' -f 4)/kops-linux-amd64

chmod +x kops-linux-amd64  
sudo mv kops-linux-amd64 /usr/local/bin/kops

**### 4. Install kubectl**

curl -LO <https://storage.googleapis.com/kubernetes-release/release/$(curl> -s <https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl>  
chmod +x ./kubectl  
sudo mv ./kubectl /usr/local/bin/kubectl

**### 5. Create S3 bucket in AWS**  
S3 bucket is used by kubernetes to persist cluster state, lets create s3 bucket using aws cli  
**\*Note:\***  Make sure you choose bucket name that is uniqe accross all aws accounts

aws s3 mb s3:// kopsridwan --region us-west-2

OR

aws s3api create-bucket --bucket <kopsridwan> --region us-east-2 --create-bucket-configuration LocationConstraint=us-east-2

**### 6. Create private hosted zone in AWS Route53**  
 1. Head over to aws Route53 and create hostedzone  
 2. Choose name for example ([dafesmith.com](http://dafesmith.com/))  
 3. Choose type as privated hosted zone for VPC  
 4. Select default vpc in the region you are setting up your cluster  
 5. Hit create**### 7 Configure environment variables.**  
Open .bashrc file

    vi ~/.bashrc

Add following content into .bashrc, you can choose any arbitary name for cluster and make sure buck name matches the one you created in previous step.

export KOPS\_CLUSTER\_NAME=ridkopsdomain.com

export KOPS\_STATE\_STORE=s3://kopsridwan

Then running command to reflect variables added to .bashrc

    source ~/.bashrc

**### 8. Create ssh key pair**  
This keypair is used for ssh into kubernetes cluster

ssh-keygen

**### 9. Create a Kubernetes cluster definition.**

kops create cluster \  
--state=${KOPS\_STATE\_STORE} \  
--node-count=2 \  
--master-size=t2.medium \  
--node-size=t2.micro \  
--zones=us-east-2a,us-east-2b \  
--name=${KOPS\_CLUSTER\_NAME} \  
--dns private \  
--master-count 1

**### 10. Create kubernetes cluster**

kops update cluster --yes

Above command may take some time to create the required infrastructure resources on AWS. Execute the validate command to check its status and wait until the cluster becomes ready

kops validate cluster

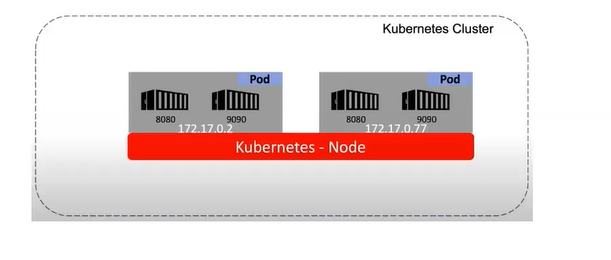
For the above above command, you might see validation failed error initially when you create cluster and it is expected behaviour, you have to wait for some more time and check again.**### 11. To connect to the master**

ssh admin@api.javahome.in

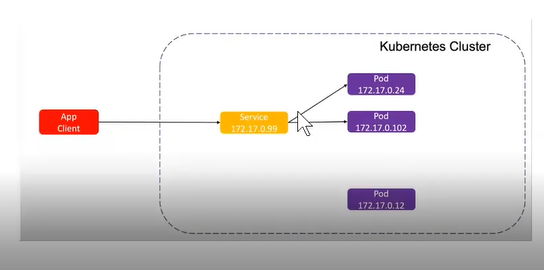
**# Destroy the kubernetes cluster**

kops delete cluster  --yes

**## Update Nodes and Master in the cluster**  
We can change numner of nodes and number of masters using following commands  
```  
   kops edit ig nodes change minSize and maxSize to 0  
   kops get ig- to get master node name  
   kops edit ig - change min and max size to 0  
   kops update cluster --yes

****

**Services 🡺 Allow multiple pods communicate with each other**

****

**Kubernetes Tutorial**

**[ec2-user@ip-172-31-41-222 ~]$ kubectl get node**

**NAME STATUS ROLES AGE VERSION**

**ip-172-20-34-28.us-east-2.compute.internal Ready master 95m v1.18.9**

**ip-172-20-61-133.us-east-2.compute.internal Ready node 94m v1.18.9**

**ip-172-20-85-183.us-east-2.compute.internal Ready node 94m v1.18.9**

**Workers and Master Node :**

**Worker node🡺 is where the actual servers and run**

**Master Node 🡺 also run a process called controller manager , controls all that happens in the worker environment, say a cluster dies and need to be restarted we use the master node**

**Master node also run a process called Scheduler managers 🡺 it helps schedule activity on the node, it decide which worker node the next available node the resource should be scheduled on.**

**We can restore/recover the whole cluster from “ectd” snapshot**

**Virtual network : helps the cluster communicate with each other**

**Pods 🡺 We can only have one pod per application example (1 pod = 1 App)**

**API server is the main entry point to the Master cluster and we communicate to the master through the API server**

**Kind: Deployment 🡺 This is a template / blueprint for creating PODS**

**Kubernetes 🡺 High availability (no downtime) , Highly scalable , disaster recovery**

**Kubelet 🡺 enable kubernetes cluster to communicate with each other.**

**PODS syntax**

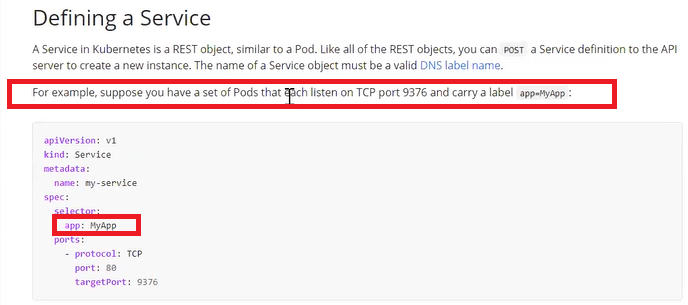
See below a Pod creation



**The “label” section in the above POD, reference the selector session in the “Service” syntax below ,**

**Services 🡺** [**https://kubernetes.io/docs/concepts/services-networking/service/**](https://kubernetes.io/docs/concepts/services-networking/service/)

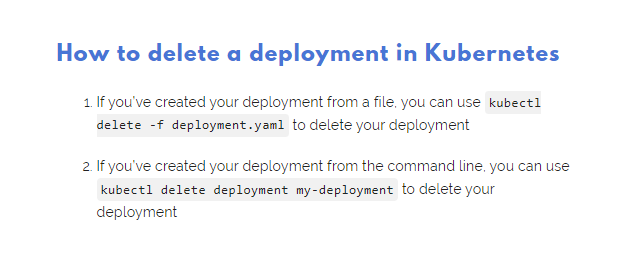
**Note : Service IP address are also called cluster IP**

****

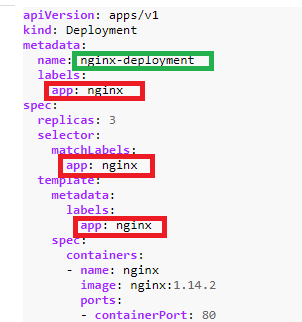
**To delete a Kubernetes service 🡺 kubectl delete services *hello-world***

**Note 🡺** If you created a replicationcontroller or a deployment through a file, you can always stop that rc or deployment with the command

Kubectl delete -f <rc\_filename> or kubectl delete -f <deployment\_filename>

****

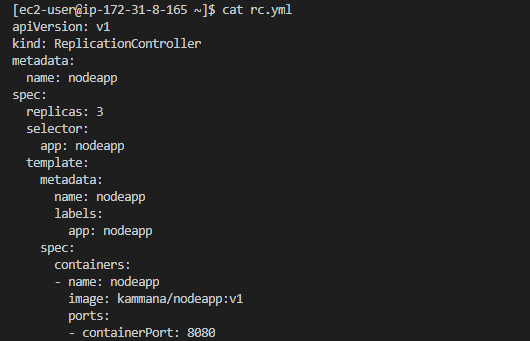
**Updating A Running Deployment**



Let's update the nginx Pods to use the nginx:1.16.1 image instead of the nginx:1.14.2 image.

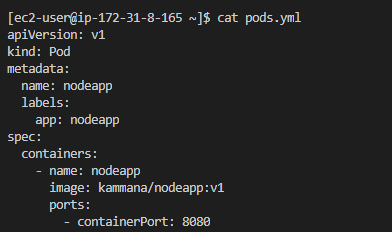
1. kubectl set image deployment/nginx-deployment nginx=nginx:1.16.1 –record
2. kubectl rollout status deployment.v1.apps/nginx-deployment
3. kubectl get rs 🡺 show replication set
4. kubectl describe deployments

**Replication controller 🡺 https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/**



**Kubenetes Pod creation**

1. wget <https://raw.githubusercontent.com/dafesmith/kubernetes-1/master/pods/pods.yml>



kubectl create -f <https://raw.githubusercontent.com/javahometech/kubernetes/master/pods/pods.yml>

or

kubectl create -f pods.yml

kubectl get pods

**TO see the content of a POD :**  kubectl describe pods/nodeapp

**To delete the pod :** kubectl delete pod nodeapp

Replication controller 🡺

wget <https://raw.githubusercontent.com/dafesmith/kubernetes-1/master/ReplicationController/rc.yml>

kubectl create -f  <https://raw.githubusercontent.com/dafesmith/kubernetes-1/master/ReplicationController/rc.yml>

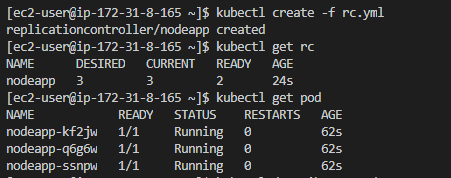
or

kubectl create -f  rc.yml

kubect get rc 🡺 list the number of replication controller

kubectl describe rc nodeapp

kubectl get pods



To kill one of the pods : kubectl delete pod <nameofpod>

Example : kubectl delete pod nodeapp-kf2jw

kops delete cluster  --yes 🡺 you remove all KOPS from the source if you create a replication controller if not a replica will reproducing

TO RUN DASHBOARD

Kubectl get namespace

kubectl get services --all-namespaces

kubectl proxy --address 0.0.0.0 --accept-hosts '.\*'

And in your browswer

http://<IP>:<PORT>/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/login

kubectl -n kubernetes-dashboard delete serviceaccount admin-user

kubectl -n kubernetes-dashboard delete clusterrolebinding admin-user

kubectl delete -f dashboard.yaml

**To roll back an export command**

export KUBECONFIG=/etc/kubernetes/admin.conf

unset KUBECONFIG 🡺 undo the above command

**Format to create a yaml file**

cat > eks-admin-service-account.yaml << EOF

apiVersion: v1

kind: ServiceAccount

metadata:

name: eks-admin

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: eks-admin

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: eks-admin

namespace: kube-system

EOF

**Importing and Exporting Database backup using SSM**

1. create a single DB: <https://docs.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?tabs=azure-portal>
2. connect and query your single DB: <https://docs.microsoft.com/en-us/azure/azure-sql/database/connect-query-ssms>
3. Export Backup the DB created 🡺 <https://help.fasthosts.co.uk/app/answers/detail/a_id/517/~/backing-up-a-mssql-database-with-sql-server-management-studio-18>
4. Restore/Import database 🡺 <https://help.fasthosts.co.uk/app/answers/detail/a_id/513/related/1/session/L2F2LzEvdGltZS8xNjAwNzQ0OTQzL2dlbi8xNjAwNzQ0OTQzL3NpZC9mVU5QZUh2V0JGNndhVEc3blRhbm9qc3pya1RhSjd3TldoVWgwbm1WUTlqODlEQVZUeGY0S1RoQlJtdCU3RWo1TjFhekJ0ZWRhMmdYbElNWEIzcGUlN0VPTmZYcjU0RGRacFZuZHd5YTQlN0UwV1RtcVdQZjEzTnVycmMlN0UyQSUyMSUyMQ%3D%3D>

**Youtube for SSMS**

<https://www.youtube.com/watch?v=Ggf4f3pqwgw&ab_channel=DevNami>

[**MS SQL Server Management Studio - Insert Update Delete Query Tutorial**](https://www.youtube.com/watch?v=Ggf4f3pqwgw&amp;ab_channel=DevNami)

<https://www.youtube.com/watch?v=QdKOqlD_3jw&ab_channel=YassirTaqatqa>

[**How to export and import database in SQL Server 2012**](https://www.youtube.com/watch?v=QdKOqlD_3jw&amp;ab_channel=YassirTaqatqa)

9’ 9

Run any code that start with csproj 🡺 \*\*/\*.csproj

To ignore a part of a code with the name “SeleniumUiTests.csproj 🡺 !\*\*/\*SeleniumUiTests.csproj

Note: for azure terraform as a code you need to install Azure cli using the link below, verify your cli with “az --version”

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest&tabs=azure-powershell>

Command for listening to a port == sudo netstat -tunlp