K8 commads

How do you restart worker node..?

kubectl uncordon worker node 1

To get nodes

**$kubectl get nodes**

In details

**$kubectl get nodes -o wide**

To get pods

**$kubectl get pods**

In details

**$kubectl get pods -o wide**

To create lable

**$kubectl label <nodename> <label name=value>**

To remove

**$kubectl label nodes <nodename> <labelname>-**

To list labels

**$kubectl get nodes --show-labels**

To create pod

**$kubectl create -f pod1.yml**

To create taint node

**$kubectl taint nodes <node name> taint key(user defined)=taint value(user defined):<effect(Noexcute)/Noschedule/preferNoschedule>** /# we have three types of effect in taint and tolerance

To remove taint

**$kubectl taint nodes <node name> taint key(user defined)=taint value(user defined):<effect(Noexcute)/Noschedule/preferNoschedule>-**

kubectl taint nodes node1 key1=value1:NoSchedule

NoSchedule:places a taint on node node1. The taint has key key1, value value1, and taint effect NoSchedule. This means that no pod will be able to schedule onto node1 unless it has a matching toleration.

3.Step-Need to check the volume is attached or not by running below command on master

**$Lsblk**

**Step-Need create file system on same volume by running below command.**

**sudo mkfs -t xfs /dev/xvds**

**To verify that your volume and claim are associated,**

**$kubectl get pvc**

**Output:**

**NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE**

**aws-pvc Bound aws-pv 3Gi RWO 3s**

**Perform a local exec into the pod, using $kubectl exec -it nameOfPod -- /bin/bash**

**To Create dashboard K8.**

1. **Need to apply dashboard yaml file**

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.2.0/aio/deploy/recommended.yaml>

1. need edit kubernates dashboard service from clusterIP to Node port using below Commands.

**$kubectl get svc -n kubernetes-dashboard**

root@ip-172-31-37-68:~# kubectl get svc -n kubernetes-dashboard

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

dashboard-metrics-scraper ClusterIP 10.96.72.66 <none> 8000/TCP 17m

kubernetes-dashboard NodePort 10.99.48.93 <none> 443:30006/TCP 17m

**$kubectl edit svc kubernetes-dashboard**

1. Creating sample user:

Need to Creating a Service Account using below yaml fle

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

1. Need to open dashboard page using below url in browser

<https://publicip> of master instance: Nodeport

<https://3.6.37.62:30006>

Then it will ask token

1. Need to generate Token using below commands.

**$kubectl -n kubernetes-dashboard get secret $(kubectl -n kubernetes-dashboard get sa/admin-user -o jsonpath="{.secrets[0].name}") -o go-template="{{.data.token | base64decode}}"**

1. **Need paste generated token in dashboard then dashboard will open**

**TO create service for pod/deployemnt objecte**

kubectl expose pod hello-pod --name=hello-svc --port=80 --type=NodePort/clusterIp/loadbalancer

kubectl expose Deployment <Deployment name> --name=hello-svc --port=80 --type=NodePort/clusterIp/loadbalancer

Setting the namespace for a request

To set the namespace for a current request, use the --namespace flag.

For example:

**kubectl run nginx --image=nginx --namespace=<insert-namespace-name-here>**

**kubectl get pods --namespace=<insert-namespace-name-here>**

Kubernetes starts with four initial namespaces:

**Default**The default namespace for objects with no other namespace

**kube-system** The namespace for objects created by the Kubernetes system

**kube-public** This namespace is created automatically and is readable by all users (including those not authenticated).

**Setting the namespace preference**

**You can permanently save the namespace for all subsequent kubectl commands in that context**.

**kubectl config set-context --current --namespace=<insert-namespace-name-here># Validate it**

**kubectl config view --minify | grep namespace:**

**Steps to create statefulset application**

1. **We need three components**

* **Headless service**
* **Statefulset**
* **Persistant Volume**

1. **need to create headless service**

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

clusterIP: None

selector:

app: nginx

1. **Need to create stateful set**

root@ip-172-31-37-68:~# cat stateful\_set.yaml

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

selector:

matchLabels:

app: nginx

serviceName: "nginx"

replicas: 3

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

name: web

volumeMounts:

- name: example1

mountPath: /home/ubuntu

volumes:

- name: example1

persistentVolumeClaim:

claimName: example2

1. **Need to create persistence volume and claim**

root@ip-172-31-37-68:~# cat pv\_local.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: example1

spec:

capacity:

storage: 10Gi

volumeMode: Filesystem

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Delete

storageClassName: local-storage

local:

path: /home/ubuntu/

nodeAffinity: ##local volume require nodeAffinity

required:

nodeSelectorTerms:

- matchExpressions:

- key: disktype

operator: In

values:

- ssd

root@ip-172-31-37-68:~# cat pvc\_local.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: example2

labels:

app: nginx

spec:

accessModes:

- ReadWriteOnce

storageClassName: local-storage

resources:

requests:

storage: 1Gi

volumeName: example1

root@ip-172-31-37-68:~#

**Scale-up a stateful set:**

**kubectl scale statefulset web --replicas=5**

**To verify**

**kubectl get pods -l app=nginx**

**Scale down a stateful set:**

**kubectl scale statefulset web --replicas=2**

**kubectl get pods -w -l app=nginx**

**# Create different namespaces**

**kubectl create namespace blue**

**kubectl create namespace red**

### Create Resource Quota

**To limit the resource to our container**

versions

Git version 2.35

Jenkins:2.3.

Docker 19.03

K8 1.8

**Plugins:sonar scanar**

**K8 continues deployment**

**Nexus**

**Sonarcube condition –waitfor qualitygate**

**To create EKS cluster**

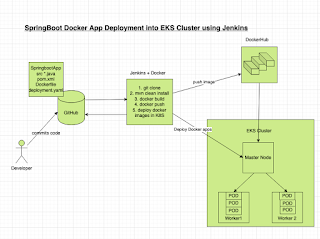
1. **need to insatll AWS cli**
2. **Need to install eksctl**
3. **Need to install kubectl.**
4. **Need to create IAM user attach admin, eks policies and cloud formation policy and need add user access key and secret key in credential file /home/ubuntu/.aws/credentials**
5. **create clutser and nodes using**

**eksctl create cluster --name demo-eks --region us-east-2 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2**

aws eks delete-nodegroup --nodegroup-name [my-nodes](https://ap-south-1.console.aws.amazon.com/cloudformation/home?region=ap-south-1#/stacks/stackinfo?filteringStatus=active&filteringText=&viewNested=true&hideStacks=false&stackId=arn%3Aaws%3Acloudformation%3Aap-south-1%3A597964732828%3Astack%2Feksctl-demo-eks1-nodegroup-my-nodes%2F651ed3d0-b7dc-11ec-834f-06d342e7c8dc) --cluster-name <my-cluster>

1. **Delete EKS Cluster using eksctl**

**eksctl delete cluster --name demo-eks --region us-east-2**



1. Need to create user using this command in master
2. sudo useradd -m jenkins

sudo -u jenkins mkdir /home/jenkins/.ssh

sudo cp ~/.ssh/known\_hosts  /var/lib/jenkins/.ssh

In salve

9.Now go to Slave node and execute the below command  
sudo -u jenkins vi /home/jenkins/.ssh/authorized\_key

1. Need enable ssh between jenkins master and slave by copying id.rsa.pub key into authorized file of slave machine under ssh folder
2. Changing file permission of id.rsa.pub to 700 in master and authorized file to 600 in slave.

**In jenkins**

**1, need add docker hub credential to push the image,in manage credential.**

**Need to define docker ID in pipe line script which we have created while adding credentials**

**stage ('Docker Build') {**

**// Build and push image with Jenkins' docker-plugin**

**withDockerRegistry([credentialsId: "dockerhub", url: "https://index.docker.io/v1/"]) {**

**image = docker.build("harish1009/mywebapp", "MyAwesomeApp")**

**image.push()**

**} }yellow is docker ID,Blue is username docker**

1. **To deploy in k8 we need add k8 kube config file in jenkins credentials**
2. **execute the below command to get kubeconfig info, copy the entire content of the file:**

**sudo cat ~/.kube/config**

**Need to add kebe config file in jeinks credentials**

**Imp plugins.**

**Docker,**

**Docker pipeline and Kubernetes Continuous Deploy plug-ins are installed in Jenkins**

**Error Kubernetes Continuous Deploy plug was 2.3 then we downgraded in 1.0 now application is working**

**node {**

**def mvnHome = tool 'Maven3'**

**stage ('checkout'){**

**checkout([$class: 'GitSCM', branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://bitbucket.org/ananthkannan/myawesomeangularapprepo']]])**

**}**

**stage('build'){**

**sh 'mvn -f MyAwesomeApp/pom.xml clean install'**

**//mvn -f MyAwesomeApp/pom.xml clean install**

**}**

**stage ('Code Quality scan') {**

**withSonarQubeEnv('sonarcube') {**

**sh 'mvn -f MyAwesomeApp/pom.xml sonar:sonar'**

**}**

**}**

**stage ('Docker Build') {**

**// Build and push image with Jenkins' docker-plugin**

**withDockerRegistry([credentialsId: "dockerhub", url: "https://index.docker.io/v1/"]) {**

**image = docker.build("harish1009/mywebapp", "MyAwesomeApp")**

**image.push()**

**}**

**}**

**stage ('K8S Deploy') {**

**kubernetesDeploy(**

**configs: 'MyAwesomeApp/springboot-lb.yaml',**

**kubeconfigId: 'k8',**

**enableConfigSubstitution: true**

**)**

**}**

**}**

**Sonar cube**

[**https://www.coachdevops.com/2019/04/install-sonarqube-on-ubuntu-how-to.html**](https://www.coachdevops.com/2019/04/install-sonarqube-on-ubuntu-how-to.html)**-install sonar cube**

[**https://www.coachdevops.com/2020/04/how-to-integrate-sonarqube-with-jenkins.html-integrate**](https://www.coachdevops.com/2020/04/how-to-integrate-sonarqube-with-jenkins.html-integrate)**- integrate sonar cube with Jenkins**

**If we get error while build and pushing docker images use below commands**

[How to fix "dial unix /var/run/docker.sock: connect: permission denied" when group permissions seem correct?](https://stackoverflow.com/questions/51342810/how-to-fix-dial-unix-var-run-docker-sock-connect-permission-denied-when-gro)

sudo usermod -aG docker $USER

sudo reboot

**Dockerfile example:**

from docker.io/centos  
maintainer devops engineer  
run yum update && yum -y install httpd  
run mkdir -p /data/myscript  
workdir /data/myscript  
cmd python app.py

We have listed all dockerfile instructions and dockerfile explained with examples below.

## ****Dockerfile Explained with Examples****

Lets get started.

**FROM**  
FROM instruction used to specify the valid docker image name. So specified Docker Image will be downloaded from docker hub registry if it is not exists locally.

Examples:

from docker.io/centos:latest  
from docker.io/centos:6

If tag “6” is not specfied, FROM instruction will use the latest tag (version).  
This is a mandatory instruction in dockerfile, rest all are optional and those can be used based on the requirement.

**MAINTAINER**  
MAINTAINER instruction is used to specify about the author who creates this new docker image for the support.

Examples:

maintainer administrator  
maintainer admin @ learnitguide.net  
maintainer devops engineer(admin @ learnitguide.net)

**LABEL**  
LABEL instruction is used to specify metadata information’s to an image. A LABEL is a key-value pair.

Examples:

label “application\_environment”=”development”  
label “application\_support”=”learnitguide.net group”

**EXPOSE**  
EXPOSE instruction is used to inform about the network ports that the container listens on runtime. Docker uses this information to interconnect containers using links and to set up port redirection on docker host system.

Examples:

expose 80 443  
expose 80/tcp 8080/udp

**ADD**  
ADD instruction is used to copy files, directories and remote URL files to the destination (docker container) within the filesystem of the Docker Images. Add instruction also has two forms – Shell Form and Executable Form.

Examples:  
Shell Form – ADD src dest

add /root/testfile /data/

Executable Form – ADD [“src”,”dest”]

add /root/testfile /data/

If the “src” argument is a compressed file (tar, gzip, bzip2, etc) then it will extract at the specified “dest” in the container’s filesystem.

**COPY**  
COPY instruction is used to copy files, directories and remote URL files to the destination within the filesystem of the Docker Images. COPY instruction also has two forms – Shell Form and Executable Form.

Examples:  
Shell Form

copy src dest  
copy /root/testfile /data/

Executable Form

copy [“src”,”dest”]  
copy /root/testfile /data/

If the “src” argument is a compressed file (tar, gzip, bzip2, etc), then it will copy exactly as a compressed file and will not extract.

**RUN**  
RUN instruction is used to execute any commands on top of the current image and use to this will create a new layer. RUN instruction has two forms – Shell Form and Executable Form.

Examples:  
Shell form:

run yum update  
run systemctl start httpd

Executable form:

run [“yum”,”update”]  
run [“systemctl”,”start”,”httpd”]

**CMD**  
CMD instruction is used to set a command to be executed when running a container. There must be only one CMD in a Dockerfile. If more than one CMD is listed, only the last CMD takes effect.  
CMD instruction has two forms – Shell Form and Executable Form.

Example :  
Shell form:

cmd ping google.com  
cmd python myapplication.py

Executable form:

cmd [“ping”,”google.com”]  
cmd [“python”,”myapplication.py”]

**ENTRYPOINT**  
ENTRYPOINT instruction is used to configure and run a container as an executable. ENTRYPOINT instruction also has two forms – Shell Form and Executable Form.

Examples:  
Shell form:

entrypoint ping google.com  
entrypoint python myapplication.py

Executable form:

entrypoint [“ping”,”google.com”]  
entrypoint [“python”,”myapplication.py”]

If user specifies any arguments (commands) at the end of “docker run” command, the specified commands override the default in CMD instruction, But ENTRYPOINT instruction are not overwritten by the docker run command and ENTRYPOINT instruction will run as it is.

So Docker CMD and ENTRYPOINT commands are used for same purpose, but both has some different functionality, refer this link to understand the [differences between Docker CMD and ENTRYPOINT Command with examples](https://www.learnitguide.net/2018/06/dockerfile-cmd-entrypoint-differences.html).

**VOLUME**  
VOLUME instruction is used to create or mount a volume to the Docker container from the Docker host filesystem.

Examples:

volume /data  
volume /appdata:/appdata

**USER**  
USER instruction is used to set the username, group name, UID and GID for running subsequent commands. Else root user will be used.

Examples:

user webadmin  
user webadmin:webgroup  
user 1008  
user 1008:1200

**WORKDIR**  
WORKDIR instruction is used to set the working directory.

Examples:

workdir /app/  
workdir /java\_dst/

**ENV**  
ENV instruction is used to set environment variables with key and value. Let’s say, we want to set variables APP\_DIR and app\_version with the values /data and 2.0 respectively. These variables will be set during the image build also available after the container launched.

Examples:

env app\_dir /data/  
env app\_version 2.0

**ARG**  
ARG instruction is also used to set environment variables with key and value, but this variables will set only during the image build not on the container.

Examples:

arg tmp\_name mycustom\_image  
arg tmp\_ver 2.0

So both Docker ENV and ARG commands are used to set environment variables, but there are some differences in functionality, refer this link to understand the [differences between Docker ENV and ARG Command with examples](https://www.learnitguide.net/2018/06/docker-arg-vs-env-command-differences.html).

**ONBUILD**  
ONBUILD instruction is used to specify a command that runs when the image in the Docker file is used as a base image for another image.

Examples:

onbuild add . /app/data  
onbuild run yum install httpd

**Types of storage in AWS**

AWS s3

AWS EFS

AWS Fsx

AWS EBS

VPC Endpoints Used to connect private instance with AWS services.

Services like s3

S3 used to store the data in terms of objects S3 storage classes are

1. Standard

2. Amazon S3 Standard-Infrequent Access

3. Amazon S3 One Zone-Infrequent Access

4. S3 on Outposts Data can be encrypted using KMS service

**How can we enable static website**

To enable static website hosting

Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

In the Buckets list, choose the name of the bucket that you want to enable static website hosting for.

Choose Properties.

Under Static website hosting, choose Edit.

Choose Use this bucket to host a website.

Under Static website hosting, choose Enable.

In Index document, enter the file name of the index document, typically index.html.

**10 how many bucket can we created in s3**

ans max 100

**3. How do share AMI to another account**

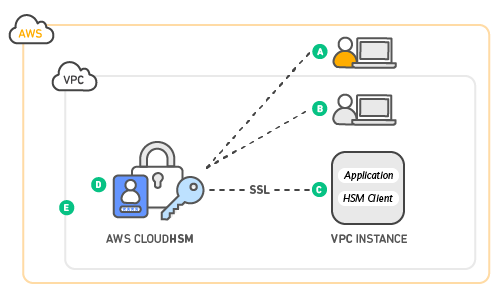
Go to ami and select ami

Go to action and select edit permission

Select private and need add ID

**CloudSHM:**

Hardware security module that enable us to generate and use your own encryption key in aws

****

**Declartive pipeline**

pipeline {

agent { docker 'maven:3.8.1-adoptopenjdk-11' }

stages {

stage('Example Build') {

steps {

sh 'mvn -B clean verify'

}

}

}

}