[suchintannit/EPAM-COE-2022-DEVOPS (github.com)](https://github.com/suchintannit/EPAM-COE-2022-DEVOPS)

<https://www.youtube.com/watch?v=vHzOq7n6gjo>

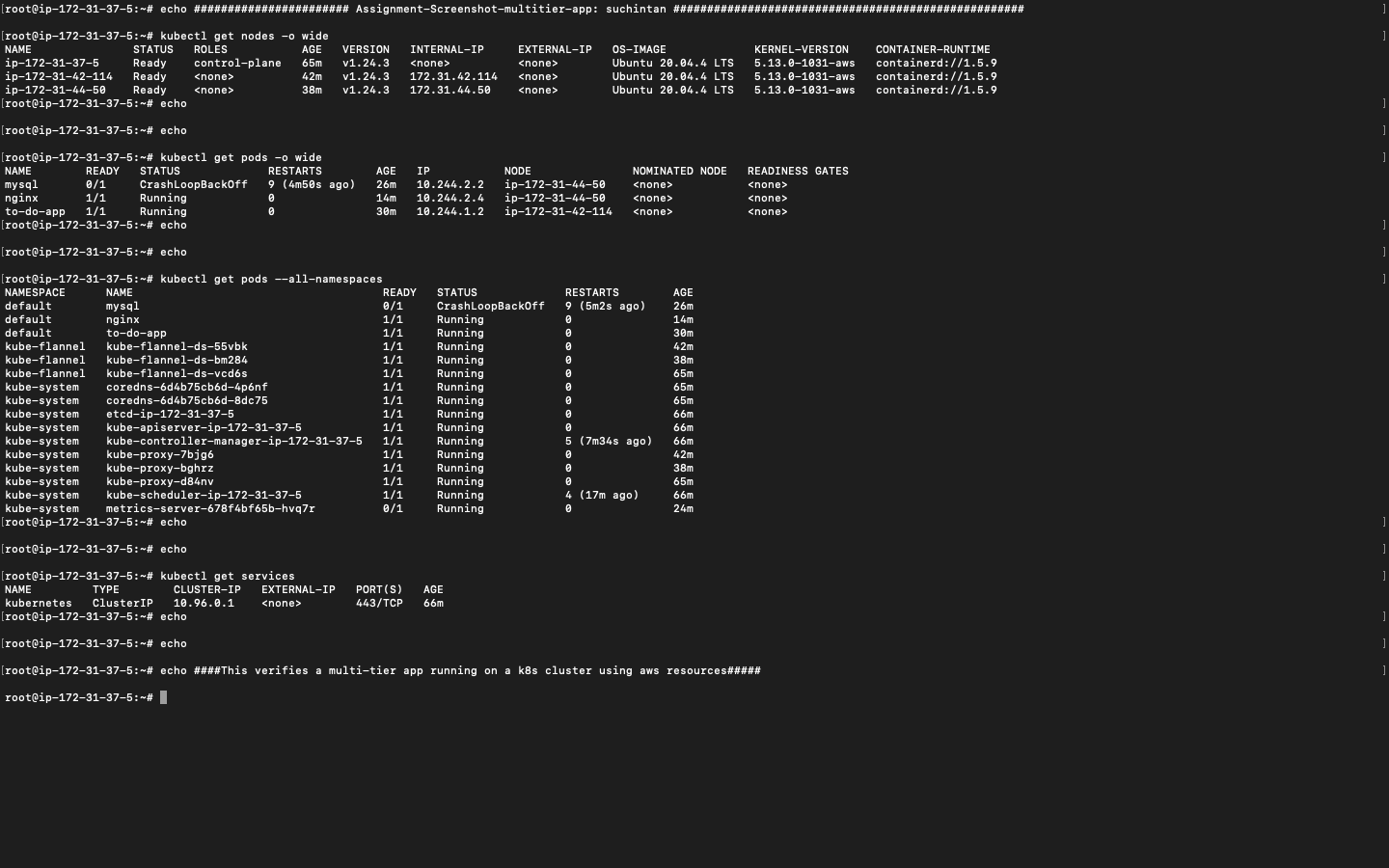
guys create 3 ubuntu (not amazon linux) vms on aws. strictly follow these steps to create a kubernetes cluster and deploy 3-tier app anyone who reports success ping

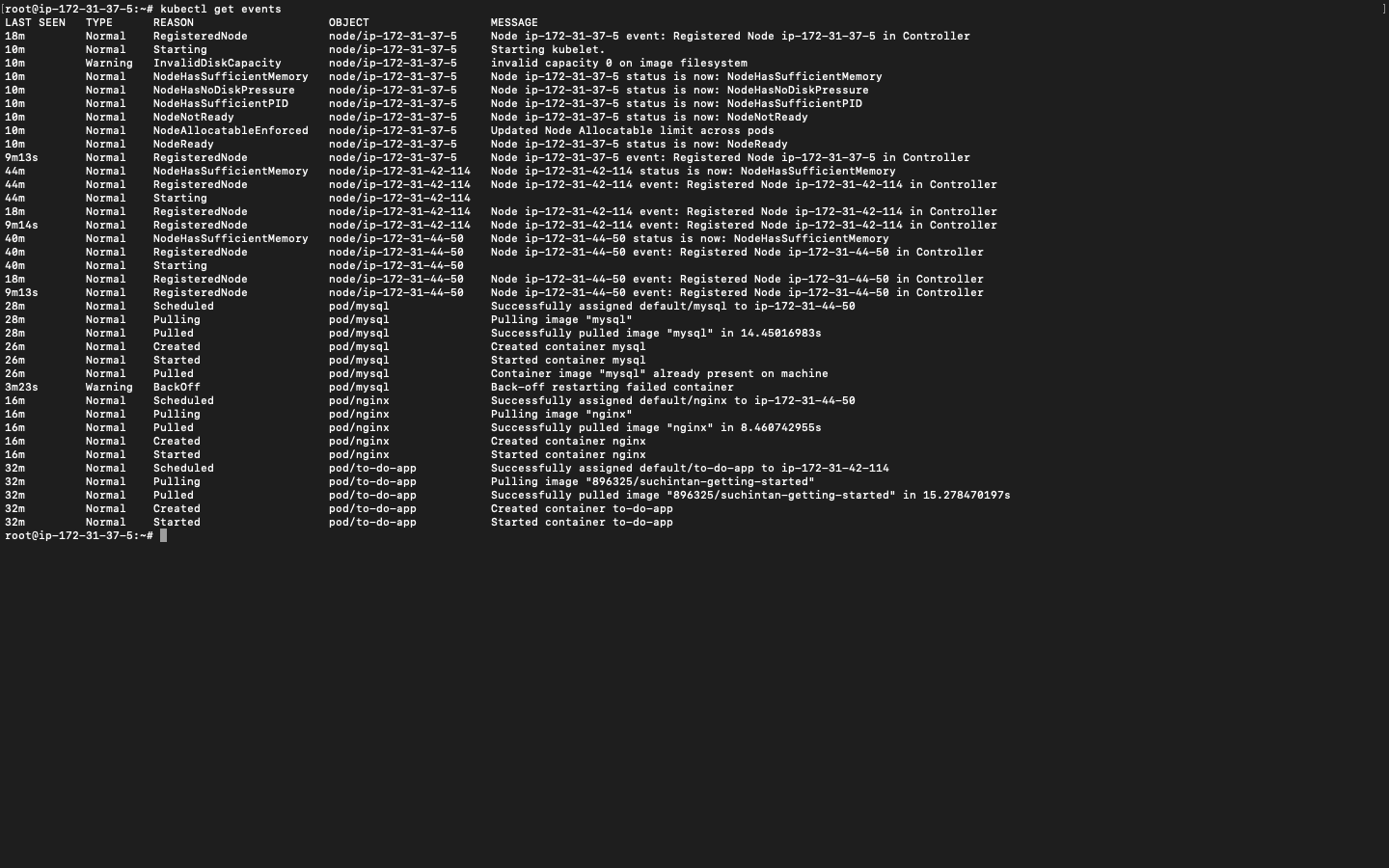
Step 4: On the master execute the following (Shell Script version)  
    sudo snap install microk8s --classic  
        sudo snap install docker  
    sudo usermod -a -G microk8s ubuntu  
       sudo chown -f -R ubuntu ~/.kube  
    newgrp microk8s  
        microk8s.enable dns dashboard registry  
        echo "alias kubectl='microk8s.kubectl'" > /home/ubuntu.bash\_aliases  
        chown ubuntu:ubuntu /home/ubuntu/.bash\_aliases  
        sudo -i echo "alias kubectl='microk8s.kubectl'" > /root/.bash\_aliases  
                or alternatively (ANSIBLE version)  
    Execute the Following in master and worker  
    $sudo apt-get install ansible  
    Create a yml file in master and workers give it any name with the following content:  
---  
- name: Install Kubernetes and Docker in Master and Worker  
  hosts: localhost  
 become: yes  
  tasks:  
  - name: Update the System  
    Command: sudo apt-get update -y  
  - name: Install Microk8s from snap  
    Command: sudo snap install microk8s --classic  
  - name: Install Docker from Snap  
    Command: sudo snap install docker  
  - name: Check Microk8s status  
  - name:Change Ownership  
    Command: sudo usermod -a -G microk8s ubuntu  
    Command: sudo chown -f -R ubuntu ~/.kube  
    Command: microk8s.status --wait-ready  
  - name: Install DNS DASHBOARD AND REGISTRY  
    Command: microk8s.enable dns dashboard registry  
  - name: Make ubuntu User the Owner  
    Command: usermod -a -G microk8s ubuntu  
  - name: Copy the Aliases  
    Command: echo "alias kubectl='microk8s.kubectl'" > /home/ubuntu/.bash\_aliases  
  - name: Make vagrant the owner of the aliases  
    command: chown ubuntu:ubuntu /home/eC2 USER NAME/.bash\_aliases  
  
On successful execution of the above the master node will have docker and kubernetes installed. check its status by running  
    $microk8s  
    $sudo docker ps  
Step 5: Execute the above steps on each worker node.  
Step 6: In the master node create the kubernetes control plane using the following commands  
    $ micrk8s add-node  
Step 7: Step 6 will give an output like the following:  
From the node you wish to join to this cluster, run the following:  
microk8s join 172.21.70.84:25000/9589cfd5c28392d2592f295bbb90341e/0689700fab2e  
Use the '--worker' flag to join a node as a worker not running the control plane, eg:  
microk8s join 172.21.70.84:25000/9589cfd5c28392d2592f295bbb90341e/0689700fab2e  --worker  
If the node you are adding is not reachable through the default interface you can use one of the following:  
microk8s join 172.21.70.84:25000/9589cfd5c28392d2592f295bbb90341e/0689700fab2e  
microk8s join 172.17.0.1:25000/9589cfd5c28392d2592f295bbb90341e/0689700fab2e  
Step 8: Try the first join command. Copy and paste in the ssh window of the workers. The workers should join the master. If not try the alternative commands.  
Step 9: Once workers have joined master, in the master node run the following commands  
    $kubectl get nodes  
    $kubectl get all  
Step 10: At this point the kubernetes cluster is created.  
Step 11: To deploy pods in the worker nodes execute the following commands in the master node (nothing is done in the workers except join ) from the website. You can use any docker image to define your container (even download form your docker hub. Deploy 1 containers in each worker by assigning them different labels and using nodeselector to deploy them on specified nodes detailed instruction is present in:  
<https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/>  
Step 12: once the pods are deployed run the following command to get the External IP address of the pods. Curl to the IP to see if the container is running.  
Step 13: Run  
    kubectl get all  
to get the status of your entire cluster.

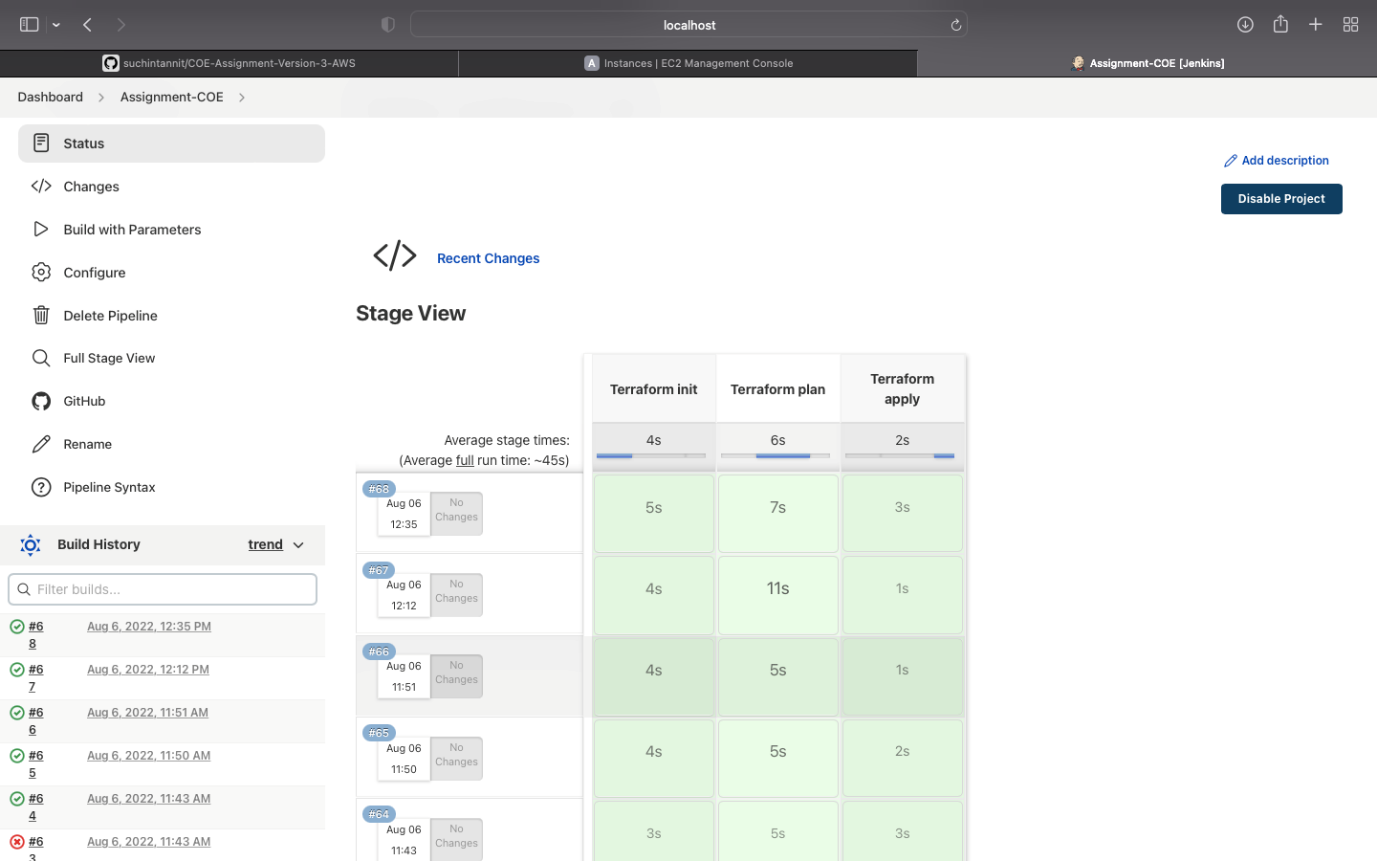
Assign Pods to Nodes

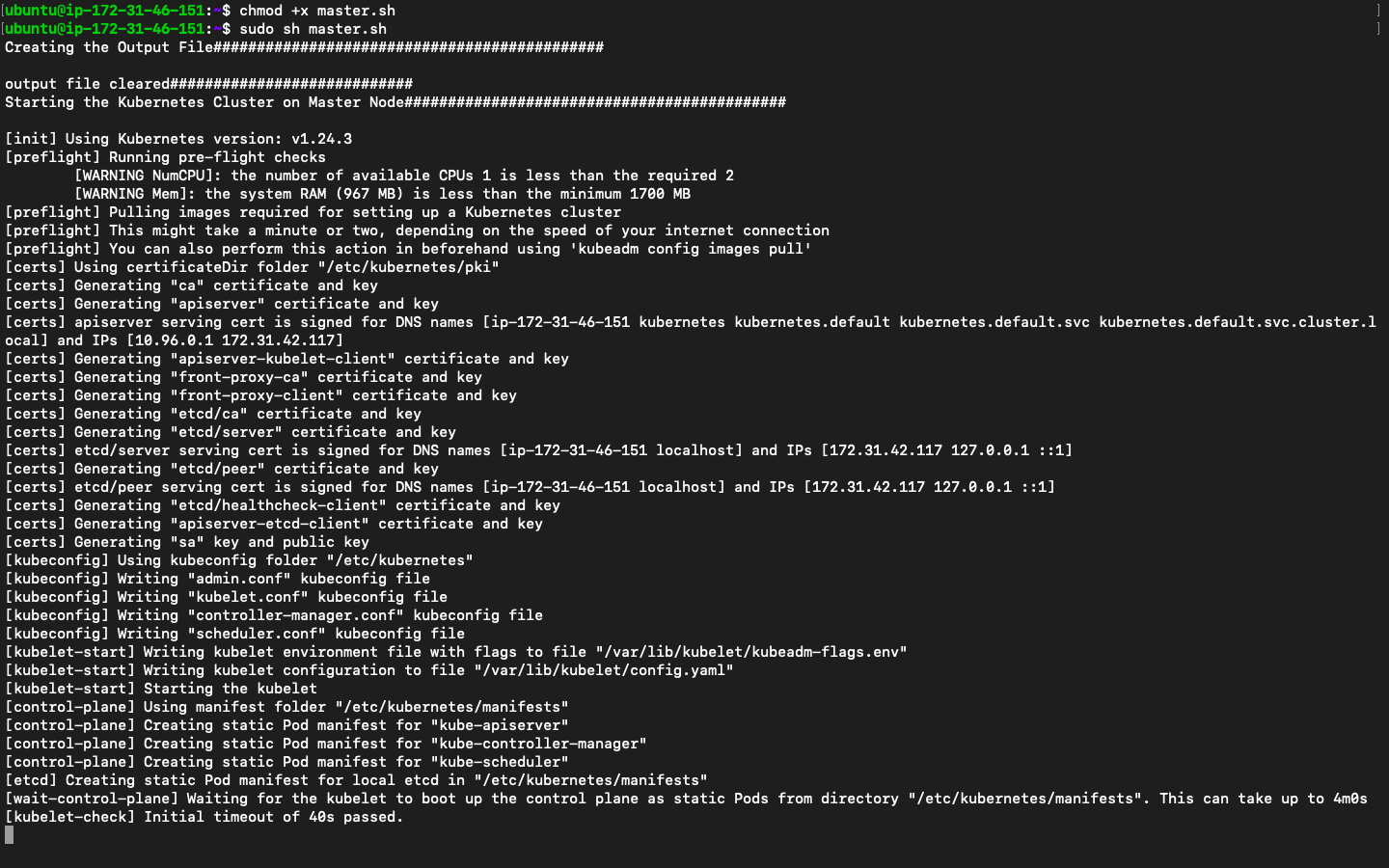
This page shows how to assign a Kubernetes Pod to a particular node in a Kubernetes cluster. Before you begin You need to have a Kubernetes cluster, and the kubectl command-line tool must be config...

[Assign Pods to Nodes | Kubernetes](https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/)



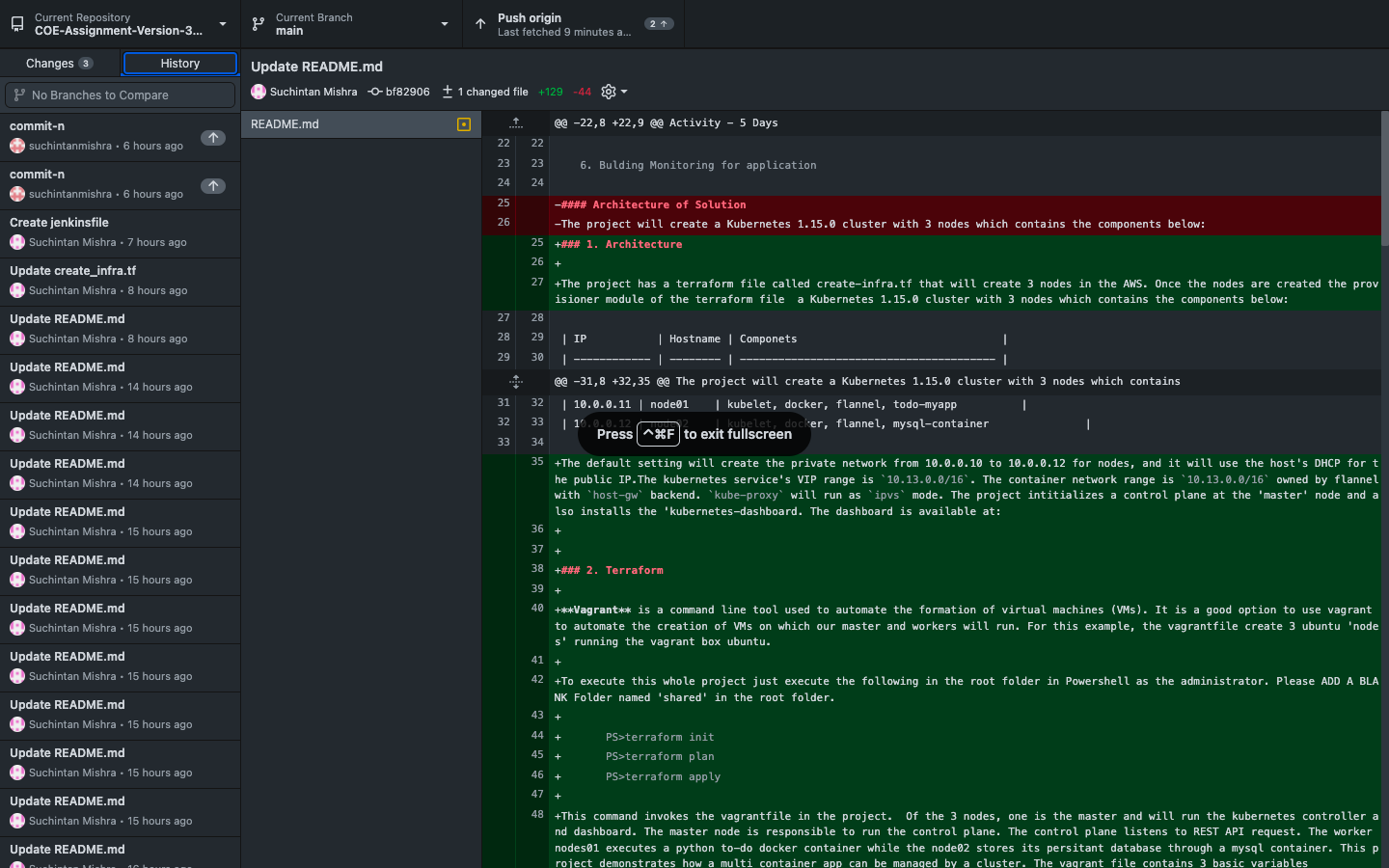






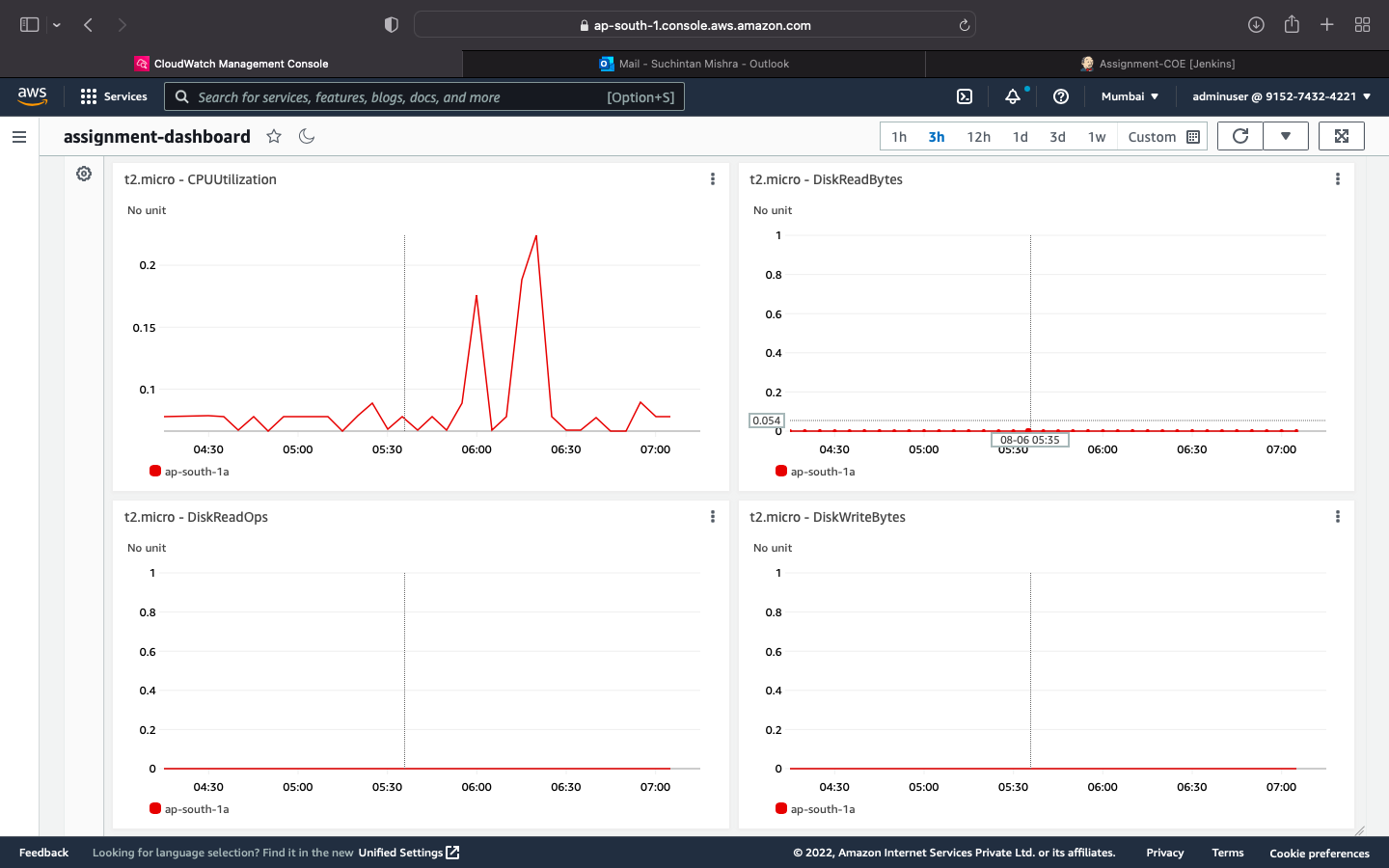
Text

Description automatically generated



Text

Description automatically generated with medium confidence



Text

Description automatically generated

<https://github.com/ValaxyTech/DevOpsDemos/blob/master/Kubernetes/k8s-setup.md>

[03-08 18:52] Sohail Mohammad

Dockerfile

FROM centos:7.9.2009

MAINTAINER mohammadsohail009@gmail.com

RUN yum install -y httpd \

 zip \

 unzip

ADD <https://www.free-css.com/assets/files/free-css-templates/download/page247/kindle.zip> /var/www/html/

WORKDIR /var/www/html

RUN unzip kindle.zip

RUN cp -rvf markups-kindle/\* .

RUN rm -rf \_MACOSX markups-kindle kindle.zip

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

EXPOSE 80

Git integration

Integrating Jenkins using GIT via webhook:  
1. got to githubrepo > repo settings > webhooks > add webhook>  
    i. payload URL > give your <jenkins url>/github-webhook/  
    ii. content type - application/json  
    iii. add token  
    now goto jenkins and generate token  
    click on username->configure-> API token-> add new >copy token  
    11e04b104903e25243af0c0e07ad5d1492  
    iv. paste token in field  
    v. add webhook  
done.

now go to jenkins create one simple free style job.  
    in scm paste github repo URL  
    NOTE: if we get error such as failed to connect to git repo  
then it means on jenkins server git is not installed.

rsync -avh /var/lib/jenkins/workspace/kubernetesproject/Dockerfile [root@172.31.85.105:/opt](mailto:root@172.31.85.105:/opt)

cd /opt

docker build -t $JOB\_NAME:$BUILD\_ID .

docker tag $JOB\_NAME:$BUILD\_ID mohammadsohail558/$JOB\_NAME:$BUILD\_ID

docker tag $JOB\_NAME:$BUILD\_ID mohammadsohail558/$JOB\_NAME:latest

docker push mohammadsohail558/$JOB\_NAME:$BUILD\_ID

docker push mohammadsohail558/$JOB\_NAME:latest

docker image rm $JOB\_NAME:$BUILD\_ID

docker image rm mohammadsohail558/$JOB\_NAME:$BUILD\_ID

docker image rm mohammadsohail558/$JOB\_NAME:latest

ansible-playbook /opt/ansible.yml

vim ansible.yml

- hosts: all

  become: true

  tasks:

  #- name: delete old deployment

  #  shell: kubectl delete -f /opt/deployment.yml

  #- name: delete service

  #  shell: kubectl delete -f /opt/service.yml

   - name: create new deployment

     shell: kubectl apply -f /opt/deployment.yml

   - name: create service

     shell: kubectl apply -f /opt/service.yml

controller-vm

vim deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

 name: happyuser

spec:

 selector:

   matchLabels:

     app: happyuser

 replicas: 2

 strategy:

  type: RollingUpdate

  rollingUpdate:

    maxSurge: 1

    maxUnavailable: 1

 template:

  metadata:

    labels:

      app: happyuser

  spec:

    containers:

     - name: happyuser

       image: mohammadsohail558/kubernetesproject

       imagePullPolicy: Always

       ports:

        - containerPort: 80

Vim service.yml

apiVersion: v1

kind: Service

metadata:

 name: happyuser

spec:

 selector:

   app: happyuser

 type: LoadBalancer

 ports:

   - port: 8080

     targetPort: 80

     nodePort: 31290

<https://github.com/suchintanmishra/Kuberenetes-vagrant-ubuntu>

<https://github.com/kubernetes/examples.git>

<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>

[kubectl Cheat Sheet | Kubernetes](https://kubernetes.io/docs/reference/kubectl/cheatsheet/)

<https://github.com/derailed/k9s>

<https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-interactive/>

<https://kubernetes.io/docs/tutorials/kubernetes-basics/>

<https://github.com/luksa/kubernetes-in-action-2nd-edition>

<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#expose>

Steps to Create an Image and rename it to your user-defined name and push it to docker hub.

Step 1: Create an image with user defined name from the nginx official image

>docker pull nginx

Step 2: Create an account in hub.docker.com remember the username.

Step 3: In CMD type:

>docker login

Login using credentials created in step 2

Step 4: This image needs to be pushed to the docker hub so create a tag for the container. Use the docker tag command to give the nginx image a new name. Be sure to swap out YOUR-USER-NAME with your Docker ID.

>docker tag nginx 896325/suchintandocker-nginx (896325 is my username)

Run docker images to verify you should have 896325/suchintan-nginx just a renamed image of nginx.

Step 5: docker push 896325/suchintandocker-nginx

>docker push 896325/suchintandocker-nginx

Open hub.docker.com to verify if your image has been uploaded.

Step 6: This image can be pulled by anyone from the hub using the command:

>docker pull 896325/suchintandocker-nginx

Similarly when we create any image from a docker file that can also be pushed by creating a tag for it.

<https://www.digitalocean.com/community/tutorials/how-to-share-data-between-docker-containers>

<https://catonmat.net/ftp/bash-redirections-cheat-sheet.pdf>

[Rasul Osmanov / DevOps\_School · GitLab (epam.com)](https://git.epam.com/rasul_osmanov/devops_school)

<https://git.epam.com/rasul_osmanov/devops_school.git>

<https://kapeli.com/cheat_sheets/Unix_File_Permissions.docset/Contents/Resources/Documents/index>

<https://www.docker.com/resources/what-container/>

<https://www.nginx.com/resources/wiki/start/topics/tutorials/install/>