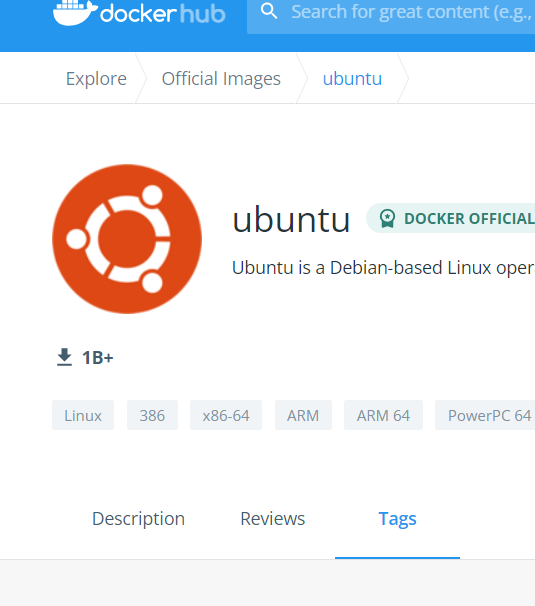
[**https://hub.docker.com/**](https://hub.docker.com/)

All images in docker hub can be pulled/ run by just finding the right tags.



**Commands**

docker help

docker images –help

docker image help

ls

docker containers ls

docker containers --ls

docker containers ls --help

docker run --help

docker run hello-world

docker image ls

docker container ls

docker container ls -a

docker run python (will run and execute python)

docker pull ubuntu:18.04

docker run ubuntu:18.04

docker container ls -a

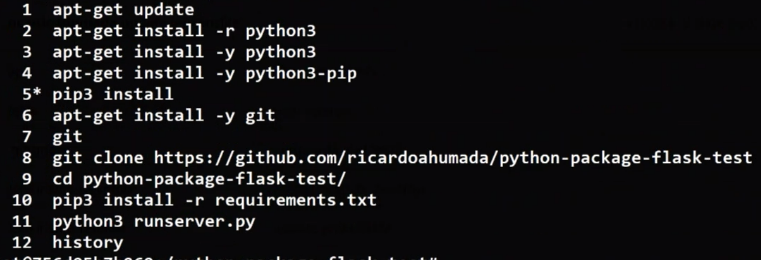
docker run ubuntu:18.04 ls -lah

docker run -i -t ubuntu:18.04 (inside this command we can make multiple linux commands in the next line that pops up pip install, ls -lah etc.)

Any commands followed by - - help will provide more info about the command

Everytime a new container is created when a ‘run’ command is executed

**To clone the application inside the container**



docker container ls -a

docker start -it 505ffa399fc4 (to start a container)

***create an image inside the container and deploy it in the repo***

docker images

docker run -it ubuntu

apt-get install -y git python3 python3-pip

git clone <https://github.com/ricardoahumada/python-package-flask-test.git>

docker ps -a

docker commit 899f4c493b2f sreeapp:v1

docker run -it sreeapp:v1 ( to test)

docker container prune (to delete all containers)

docker image prune (to delete all images)

docker image rm “image id”

docker image rm “container id”

so far with **container** to generate **images**

Publish in Docker HUB

**create image to publish in registry like docker hub, nexus etc.**

**-**create the command inside nano dockerfile and then run the command

**samples**

FROM ubuntu

RUN mkdir temp

COPY . temp

WORKDIR temp

To install python dependencies

FROM ubuntu

RUN apt-get update && apt-get install -y python3-pip

COPY . /

RUN cd python-package-flask-test && pip3 install -r requirements.txt

WORKDIR python-package-flask-test

Create images with dockerfile

FROM ubuntuRUN apt-get update && apt-get install -y python3 python3-pipCOPY . /RUN cd python-package-flask-test && pip3 install -r requirements.txtWORKDIR python-package-flask-testCMD ["python3", "runserver.py"]EXPOSE 5000

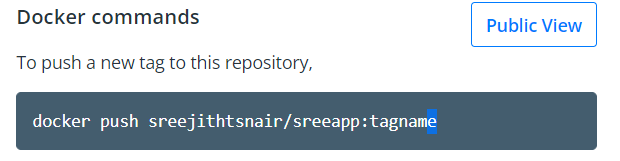
docker container stop 42a3bd00f42c (to stop)

docker run -d -p 500XX:5000 sreeapp:v2 (to enable port to run – first port number is the new port / 2nd original port)

Docker ps to check if the container is running in the background .. add -a to find all the containers those were running

Docker login

User name & password



docker tag sreeapp:v2 sreejithtsnair/sreeapp:v2 (first part is local docker repo & second one is the name tagged for docker hub with username/ app name)

docker push sreejithtsnair/sreeapp:v2

docker image rm cd25bb0d4331 cd25bb0d4331 180f411e3e0c –force (to remove) use force only if it is not removing

# Publish in Azure Container Registry

Create **Container registry** in AZURE

Take the username and pwd from the container registry

docker login sreeappregistry.azurecr.io

docker images

docker tag sreejithtsnair/sreeapp:v2 sreeappregistry.azurecr.io/sreeapp:v2

docker images

docker push sreeappregistry.azurecr.io/sreeapp:v2

Publish new image in Jenkins & push to GitHub

Jenkins can do a *docker build* & *docker push*

sudo service jenkins status (if doesn’t start – do a sudo service jenkins start)

log in to Jenkins for docker

install docker pipeline if it is not installed

sudo usermod -a -G docker Jenkins (enabling docker in Jenkins)

restart Jenkins (by adding restart after the port number to ensure that docker is added to Jenikns)

KUBERNETES

To start minikube ; type ‘minikube start’ in terminal

Same for stop

Kubectl ..

kubectl --help | more

kubectl create --help | more

kubectl expose --help | more

kubectl run --help | more

kubectl explain --help | more

kubectl get --help | more

kubectl edit --help | more

kubectl delete --help | more

kubectl get services --help | more

kubectl create deployment --help | more

kubectl expose deployment --help | more

kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4 (for single executions)

kubectl expose deployment hello-minikube --type=NodePort --port=8080

kubectl get pods

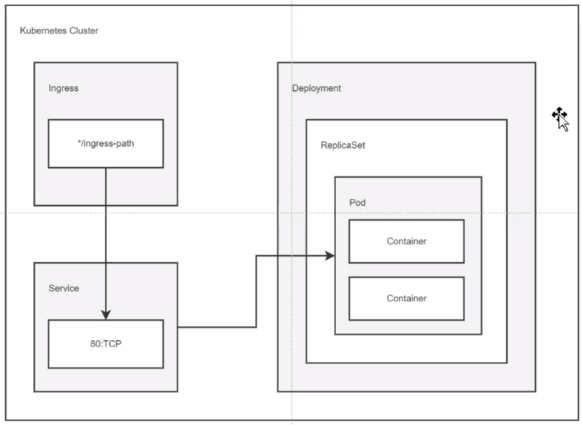
kubectl get deployments

kubectl get services

kubectl delete service hello-minikube

kubectl delete deployment hello-minikube

**NB** : get and delete are the important commands for beginners



Use nginx deployment for executing Kubernetes changes

By creating a yaml file and updating it periodically.

mkdir Kubernetes

nano nginx\_deployment.yaml

more nginx\_deployment.yaml

kubectl apply -f nginx\_deployment.yaml

kubectl describe service nginx-deployment

curl localhost:5050

416 curl localhost:5050/api/v1/restricted

Do a minikube status or start if it doesn’t work

413 cd workspace/kubernetes/

414 more nginx\_deployment.yaml

415 nano nginx\_deployment.yaml

416 kubectl get pods

417 kubectl get services

418 kubectl get replica sets

419 kubectl get volumes

420 kubectl apply nginx\_deployment.yaml

421 kubectl apply -f nginx\_deployment.yaml

422 more nginx\_deployment.yaml

423 kubectl get pods

424 nano nginx\_deployment.yaml

425 kubectl apply -f nginx\_deployment.yaml

426 kubectl get replica sets

427 kubectl get pods

428 minikube status

429 kubectl apply -f nginx\_deployment.yaml

430 kubectl get pods

431 kubectl get all

432 kubectl get all |grep nginx

434 kubectl expose deployment nginx\_deployment --type=NodePort

435 kubectl expose deployment nginx-deployment --type=NodePort

442 kubectl port-forward service/nginx-deployment 8090:80

443 ls -lah

444 git clone https://github.com/ricardoahumada/kubernetes

445 ls -lah

446 ls

447 ls -lah

448 git clone https://github.com/ricardoahumada/kubernetes

455 rm -rr kubernetes/

457 git clone https://github.com/ricardoahumada/kubernetes

458 ls -lah

459 cd kubernetes/

460 cd python-package-flask-test/

461 kubecl get all

462 kubectl get all

463 kubectl delete service/nginx-deployment

464 kubectl delete deployment.apps/nginx-deployment

465 kubectl get all

466 kubectl apply -f 1-python-package-flask-test\_deployment.yaml

467 kubectl get all

468 nano 1-python-package-flask-test\_deployment.yaml

469 kubectl get all

470 nano 1-python-package-flask-test\_deployment.yaml

471 kubectl get all

472 kubectl apply -f 1-python-package-flask-test\_deployment.yaml

473 kubectl get all

474 kubectl apply -f 2-python-package-flask-test\_deployment.yaml

475 kubectl apply -f 2-python-package-flask-test\_service.yaml

476 kubectl get all

477 kubectl describe service/python-package-flask-test-service

478 kubectl logs

479 kubectl logs service/python-package-flask-test-service

480 kubectl port-forward service/python-package-flask-test-service 5050:5000

481 kubectl delete service/python-package-flask-test-service

482 kubectl delete deployment.apps/python-package-flask-test-deployment

483 docker login ricardobootcampregistry.azurecr.io

484 more $HOME/,docker/config.json

485 more $HOME/.docker/config.json

486 kubectl create secret generic acr-secret --from-file=.dockerconfigjson=$HOME/.docker/config.json --type=kubernetes.io/dockerconfigjson

487 kubectl get secrets

488 kubectl apply -f 3-python-package-flask-test\_deployment\_az.yaml

489 kubectl get all

490\*

491 kubectl apply -f 4-python-package-flask-test\_service\_az.yaml

492 kubectl get all

493 kubectl port-forward service/python-package-flask-test-service-az 5050:5000

494 kubectl logs service/python-package-flask-test-service-az

495 git pull

496 kubectl apply -f 4-python-package-flask-test\_service\_az.yaml

497 kubectl port-forward service/python-package-flask-test-service-az 5050:5000

498 kubectl delete service/python-package-flask-test-service-az

499 kubectl delete deployment.apps/python-package-flask-test-deployment-az

⚠Needs AZ cluster:

<https://docs.microsoft.com/en-us/azure/aks/learn/quick-kubernetes-deploy-portal?tabs=azure-cli>

⚠Needs AZ installation:

curl -sL <https://aka.ms/InstallAzureCLIDeb> | sudo bash

ref: <https://docs.microsoft.com/en-us/cli/azure/get-started-with-azure-cli>

az login



az aks get-credentials --resource-group [myResourceGroup] --name [myAKSCluster] --admin

az aks get-credentials --resource-group [myResourceGroup] --name [myAKSCluster] --admin

kubectl config get-contexts

kubectl config set current-context AZ-CONTEXT

kubectl get nodes

512 sudo apt update

513 sudo apt install -y redis-server

514 sudo service redis status

515 export REDIS\_HOSTNAME=localhost

516 echo REDIS\_HOSTNAME

517 echo $REDIS\_HOSTNAME

518 pip install -r

519 cd workspace/kubernetes/python-package-flask-test/

520 pip install -r

521 sudo service redis status

522 echo REDIS\_HOSTNAME

523 export REDIS\_HOSTNAME=localhost

524 echo $REDIS\_HOSTNAME

525 pip install -r

526 python3 flask\_app.py

527 pip install -r

528 python3 flask\_app.py

529 pipn install -r re

530 sudo apt update

531 sudo apt install -y redis-server

532 sudo service redis status

533 export REDIS\_HOSTNAME=localhost

534 echo $REDIS\_HOSTNAME

535 pip install -r

536 pip install -r re

537 pip install -r requirements.txt

538 cd ..

539 cd sample\_flask\_with\_redis/

540 pip install -r

541 kubectl get all

542 kubectl get contexts

543 kubectl config get -contect

544 kubectl config get -context

545 kubectl config get-context

546 kubectl config get-contexts

547 ps -qax

548 ps -ax

549 ps -ax|grep python

550 kill -9 46266

551 kill -9 46584

552 docker ps

553 docker container prune

554 docker ps

555 docker container prune e7c4380ffc6b c85935083505 794dbf3cb7c4 8b8c0f6e00e3

556 docker container prune

557 pip install -r requirements.txt

558 python3 flask\_app.py

559 kubectl get all

560 docker ps

561 docker container stop c85935083505 e7c4380ffc6b 794dbf3cb7c4 8b8c0f6e00e3

562 docker container prune

563 python3 flask\_app.py

564 pas -ax |grep python

565 ps -ax |grep python

566 python3 flask\_app.py

567 sudo service redis stop

568 unset REDIS\_HOSTNAME

569 sudo service redis status

570 docker build --tag=sreejithtsnair/redis -f redis/Dockerfile .

571 docker build --tag=sreejithtsnair/flaskappforredis -f app/Dockerfile .

572 kubectl apply -f flaskappwithredis/app-ns.yaml

573 sudo service redis status

574 kubectl apply -f flaskappwithredis/app-ns.yaml

575 kubectl get all

576 kubectl get namespaces

577 sudo service redis status

578 minikube status

579 minikube start

580 minikube status

581 kubectl apply -f flaskappwithredis/app-ns.yaml

582 kubectl apply -f flaskappwithredis/redis-rs.yaml

583 kubectl apply -f flaskappwithredis/redis-svc.yaml

584 kubectl apply -f flaskappwithredis/flaskapp-rs.yaml

585 kubectl apply -f flaskappwithredis/flaskapp-svc.yaml

586 kubectl get namespaces

587 kubectl get all

588 kubectl get all -n flaskapp-dev

589 kubectl port-forward service/flaskapp-service 5000:5000 -n flaskapp-dev

590 curl localhost:5000

591 curl localhost:5000/set\_price/34

592 curl localhost:5000/get\_price

## Jenkins to launch Kubectl commands

Jenkins needs credentials for that;

602 cd

603 ls -lah

604 more .kube/config

605 export KUBECONFIG=~/.kube/config

606 sudo apt install -y acl

607 setfacl -R -m u:jenkins:rwx /home/ubuntu

pipeline {

agent any

stages {

stage('Exexuting kubectl') {

steps {

echo 'Exexuting kubectl'

withKubeConfig([credentialsId: 'k8-credentials']) {

sh 'kubectl get all'

}

}

}

}

}

# TERRAFORM

First create a folder (directory)

Create a nano main.tf

Az login

Terraform init

Terraform apply …… nano file…

To create a plan - terraform apply out.plan ( a plan doesn’t reach directly the cloud, instead it will create a plan file in azure or cloud. This plan doesn’t change the file.

But if we use terraform apply, it will change the resources in the cloud/ estate)

Any changes in the nano file will be overwritten in az/ cloud/ estate if we apply

Everything can be managed with a tf file (main.tf example)

How we change inside the tf file is what matters

After applying to log in to the azure VM

terraform output -raw tls\_private\_key > id\_rsa

ls -lah (to find id\_rsa)

Do an ssh command to log in to the vm

ssh -i id\_rsa [azureuser@104.208.100.2](mailto:azureuser@104.208.100.2) (azureuser – user name defined in tf file and the ip address is the public ip established in the end)

chmod 400 id\_rsa (to change the permissions

destroy to save resources

do - Az vm list - to check running vms

# AKS cluster Principal

To create principal account

az ad sp create-for-rbac --skip-assignment

use the value produced (app. Password etc.) to update the terraform.tfvars thru a

nano terraform.tfvars

674 az ad sp create-for-rbac --skip-assignment

675 cd ..

676 cd aks-cluster/

677 nano terraform.tfvars

679 ls -lah

680 nano aks-cluster.tf

681 terraform init

682 history

Terraform output – should show the info in Kubernetes

az aks get-credentials --resource-group $(terraform output -raw resource\_group\_name) --name $(terraform output -raw kubernetes\_cluster\_name) -LINUX ANY THING AFTER $ INSIDE () IS ANOTHER COMMAND.

Terraform output

Az aks list

Az aks list -o table

## Terratest to test Terraform

# Integrating the pipeline – After creating the infrastructure using terraform and deploying the app using Kubernetes

# Jenkins + terraform

## Install the following plugins:

* [Azure Credentials](https://plugins.jenkins.io/azure-credentials/)

## Get the sucription id

* subId=$(az account show --output tsv --query id)

## Create a principal

* az ad sp create-for-rbac --name terraform-jenkins --role Contributor --scopes /subscriptions/${subId}
  + Create an azure credential with the returned data, named: "azure-credentials"
* scp ubuntu@[ubuntu IP]:.kube/config k8-config

in local machine

## Create storage account and a storage container

* resourceGroup="terraform\_group"
* az group create --name $resourceGroup --location eastus
* az storage account create --name jenkinsterraformsa --resource-group $resourceGroup --location eastus
* az storage container create --account-name jenkinsterraformsa --name jenkinsterraformac

### Get the primary key

resourceGroup="terraform\_group"

- az storage account keys list -g $resourceGroup -n #name of storage container in this case jenkinsterraformssree --query [0].value -o tsv

- Create a text credential named: "access-key"

scp ubuntu@[ubuntu IP]:.kube/config k8-config

## Create pipeline as in the example

Create two credentials

* One with (Azure Principal) – appid tenant id and password which was acquired using   
  az ad sp create-for-rbac --name terraform-jenkins --role Contributor --scopes /subscriptions/${subId}
* Second one – access-key (secret Key) – with the output of   
  az storage account keys list -g $resourceGroup -n #name of storage container in this case jenkinsterraformssree --query [0].value -o tsv

[Jenkinsfile-terraform](https://github.com/ricardoahumada/terraform/blob/master/jenkins/Jenkinsfile-terraform)

[terraform-sreejith/Jenkinsfile-terraform at master · sreejithtsnair/terraform-sreejith (github.com)](https://github.com/sreejithtsnair/terraform-sreejith/blob/master/jenkins/Jenkinsfile-terraform)

cp -r hello-world *hello-world-wenv* for copying to a new directory (creating and copying the data)

TF\_VAR\_HELLO\_TEXT sample – TF\_VAR is a constant for defining environment variable

## How to hide username and password\*\*\*\*\*\*\*

export TF\_VAR\_HELLO\_TEXT="hOLA MUNDO!" (“HELLO\_TEXT” is the variable from the tf file for user name)

echo $ TF\_VAR\_HELLO\_TEXT and check

repeat the same for password too

try to do a terraform apply.. to validate

Export appid & password

Do an init and apply

Acquire appid, pwd, sub id..

Initiate service principal

# ANSIBLE – used for making changes in the infrastructure

Gen info --- if you don’t want to write a complete command in one line, then a back slash can be used

[ansible.builtin.apt module – Manages apt-packages — Ansible Documentation](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html)

check the notepad from codeshare

rm -rf to remove