**AKS User manual**

**Step 1: Login into Ubuntu machine using Putty**

**IP address**: 40.68.46.109

**Authenticate using Private Key**

**Step 2:** After Login into Ubuntu machine it will ask username and password

**Username** – \*\*\*\*\*

**Password** – \*\*\*\*\*

**Step 3:** Go to root directory type command – **sudo su**

**Step 4:** If you want to install Azure CLI, Docker and Kubectl use below stepscre

**Install Azure CLI**

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt?view=azure-cli-latest>

**Step 1:** Get packages needed for the install process:

sudo apt-get update

sudo apt-get install curl apt-transport-https lsb-release gpg

**Step 2:** Download and install the Microsoft signing key:

curl -sL https://packages.microsoft.com/keys/microsoft.asc | \

gpg --dearmor | \

sudo tee /etc/apt/trusted.gpg.d/microsoft.asc.gpg > /dev/null

**Step 3:** Add the Azure CLI software repository:

AZ\_REPO=$(lsb\_release -cs)

echo "deb [arch=amd64] https://packages.microsoft.com/repos/azure-cli/ $AZ\_REPO main" | \

sudo tee /etc/apt/sources.list.d/azure-cli.list

Step 4: Update repository information and install the azure-cli package:

sudo apt-get update

sudo apt-get install azure-cli

Run the Azure CLI with the az command. To sign in, use the [az login](https://docs.microsoft.com/en-us/cli/azure/reference-index#az-login) command.

1. Run the login command.

az login

If the CLI can open your default browser, it will do so and load a sign-in page.

Otherwise, you need to open a browser page and follow the instructions on the command line to enter an authorization code after navigating to <https://aka.ms/devicelogin> in your browser.

1. Sign in with your account credentials in the browser.

**Install docker**

<https://askubuntu.com/questions/938700/how-do-i-install-docker-on-ubuntu-16-04-lts>

**Step 1:** Set up the Docker repository

sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

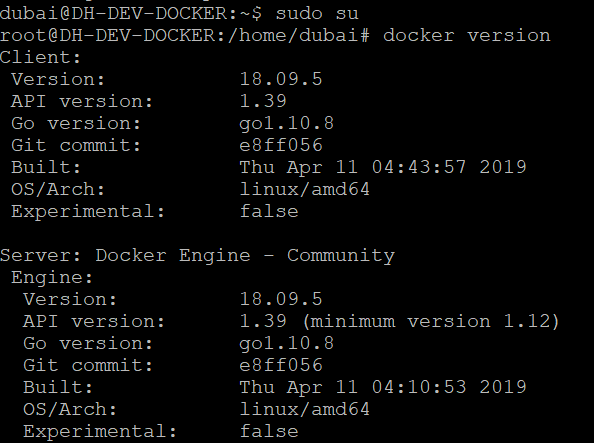
**Step 2:** Install Docker CE

sudo apt-get update

sudo apt-get install docker-ce

**Step 3:** Verify the installation

Docker version



**Install Kubectl**

**sudo apt-get update && sudo apt-get install -y apt-transport-https**

**curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -**

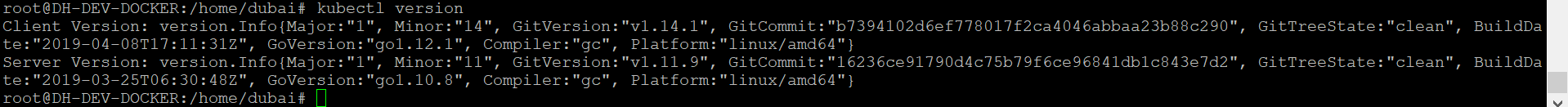
**echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list**

**sudo apt-get update**

**sudo apt-get install -y kubectl**

Test to ensure kubectl install using below command:

**kubectl version**



**Install Helm**

$ curl -LO <https://git.io/get_helm.sh>

$ chmod 700 get\_helm.sh

$ ./get\_helm.sh

**Docker build and run**

<https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app>

Go to root directory of our application where Docker file resides to build Docker image. Building the Docker Image use following commands.

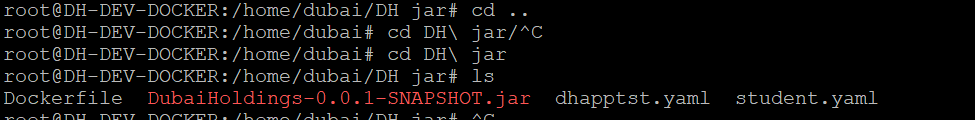
**Step 1: Go to application directory**

root@DH-DEV-DOCKER:/home/dubai# **cd DH\ jar**

**Step2:** List the file available in the folder

root@DH-DEV-DOCKER:/home/dubai/DH jar# **ls**

**Dockerfile DubaiHoldings-0.0.1-SNAPSHOT.jar dhapptst.yaml student.yaml**

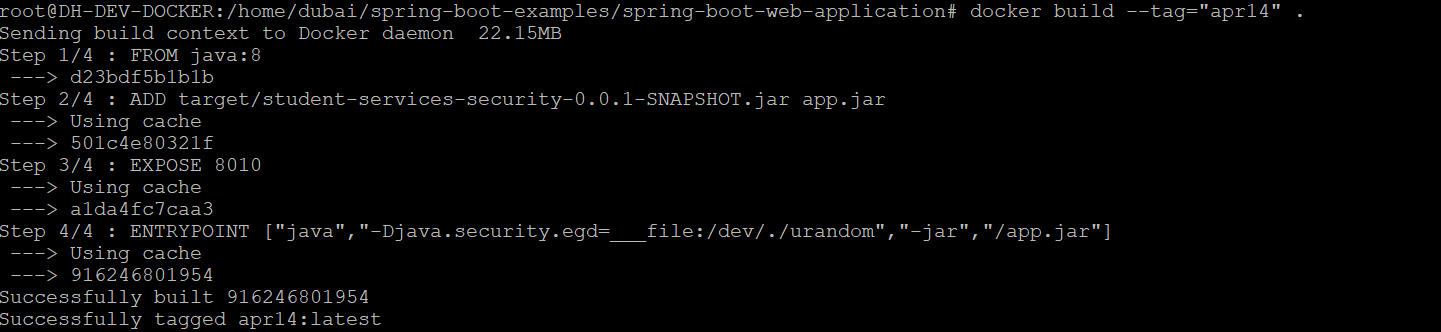


**Step 3: Build the Docker image using Docker file**

****

**Docker build syntax usage:** docker build [OPTIONS] PATH | URL | -

root@DH-DEV-DOCKER:/home/dubai/DH jar# **docker build tag="apr14" .**



**Step 4: Log in to the Azure Account and Container registry**

az login

Username: [jispnsvc@dubaiholding.com](mailto:jispnsvc@dubaiholding.com)

Password: Povu0151

**AZ login with service principle**

az login --service-principal --username "95e2f236-4732-424b-98f5-3a4855451c38" --password "}:GrB)A3rZs?kdFlJiN?6#QG^]i" --tenant "eee3385e-742f-4e2e-b130-e496ed7d6a49"

**ACR Login**

az acr login --name dhdevregistrypoc

Username: dhdevregistrypoc

Password: 30O1ngYE3woXTa5iwYEZp1Nk1q/1C6C=

**Step 5: Now, tag your local *apr14* image with the *acrloginServer (*dhdevregistrypoc.azurecr.io*)* address of the container registry. To indicate the image version, add: v1 to the end of the image name:**

docker tag apr14 dhdevregistrypoc.azurecr.io/apr14:v1

docker tag <imageid> <Login server>/<Dockerimage> 🡪 create a copy in proper name

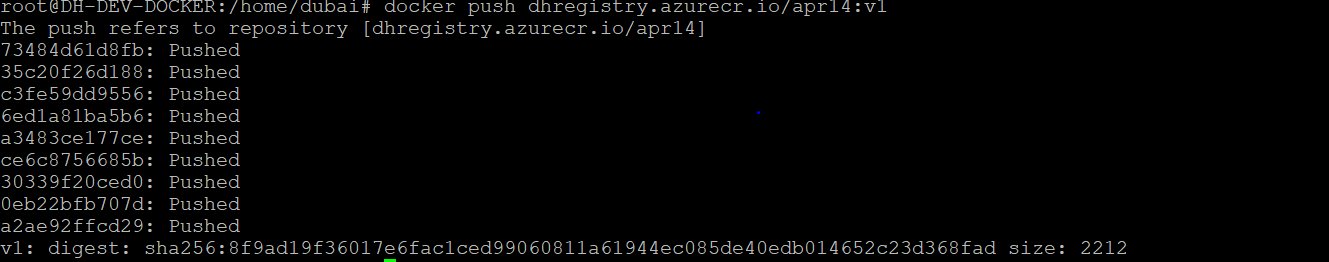
docker images 🡪Will give u the details abt images in docker

**Step 6: Push the Docker Image to ACR**

docker push dhdevregistrypoc.azurecr.io/apr14:v1

**Step 7: List all the ACR images**

az acr repository list --name dhdevregistrypoc



**Step 8: To access images stored in ACR, you must grant the AKS service principal the correct rights to pull images from ACR. Assign “ACRPULL” Role.**

****

**Step 8: Create AKS in azure in portal with one node and select Advance Networking and also configure existing service principle**

Display Name:hospitalitydxpdevaks

Client ID: 95e2f236-4732-424b-98f5-3a4855451c38

tenant ID: eee3385e-742f-4e2e-b130-e496ed7d6a49

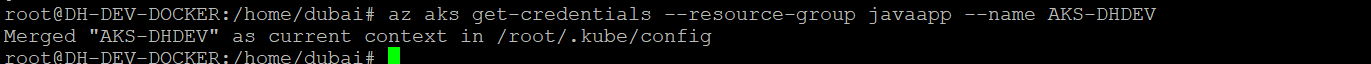
Object ID:be04a37d-36a7-494d-8c51-0da22f84bcf7

key: }:GrB)A3rZs?kdFlJiN?6#QG^]i

## Step 9 : Connect to cluster using kubectl

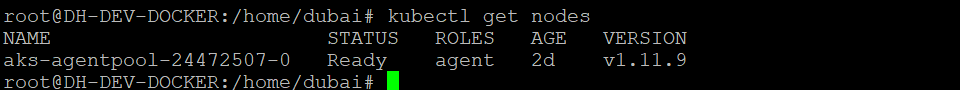
az aks get-credentials --resource-group **DXP-Hospitality-DEV-RG-Compute** --name DXP-DEV-AKS

output



**Step 9: To verify the AKS Cluster nodes**

Kubectl get nodes



**Step 10: To deploy a basic Tiller into an AKS cluster, use the**[**helm init**](https://docs.helm.sh/helm/#helm-init)**command.**

helm init

helm repo update

**Step 11 : create service Account using below kubectl apply command**

Before you can deploy Helm in an RBAC-enabled AKS cluster, you need a service account and role binding for the Tiller service. For more information on securing Helm / Tiller in an RBAC enabled cluster, see [Tiller, Namespaces, and RBAC](https://docs.helm.sh/using_helm/#tiller-namespaces-and-rbac). If your AKS cluster is not RBAC enabled, skip this step.

kubectl apply -f helm-rbac.yaml



**Step 12: Create Namespace for your ingress controller**

kubectl create namespace ingress

**Step 13: Generate TLS Certificate**

For this article, let's generate a self-signed certificate with openssl. For production use, you should request a trusted, signed certificate through a provider or your own certificate authority (CA). In the next step, you generate a Kubernetes Secret using the TLS certificate and private key generated by OpenSSL.

The following example generates a 2048-bit RSA X509 certificate valid for 365 days named ingress-tls.crt. The private key file is named ingress-tls.key. A Kubernetes TLS secret requires both of these files.

This article works with the demo.azure.com subject common name and doesn't need to be changed. For production use, specify your own organizational values for the -subjparameter:

openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-out **ingress-tls**.crt \

-keyout ingress-tls.key \

-subj "/CN=aks.dev.com/O=ingress"

**Step 14: Use Helm to deploy an NGINX ingress controller with Https and TLS certificate**

helm install stable/nginx-ingress --namespace ingress --name dhmay17 --set controller.service.enableHttp=false --set default-ssl-certificate=default/**ingress-tls**

**Step 15: To deploy your application, service and ingress configuration, use the**[kubectl apply](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#apply)**command using attached yaml.**



kubectl apply –f Deployment&Service.yaml –-namespace=ingress

**Step 16: To configure your application in ingress controller, use the**[**kubectl apply**](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#apply)**command using attached yaml.**



Kubectl apply –f ingress.yaml –-namespace=ingress

**Step 17: some of the useful command**

* Kubectl get pods –-namespace=ingress
* Kubectl get service –-namespace=ingress
* Kubectl delete deployment deploymentname
* Kubectl delete services servicesname
* Kubectl logs podname –-namespace=ingress
* Kubectl get events
* Kubectl describe podname –-namespace=ingress
* **kubectl get pods --all-namespaces**
* --overwrite-existing