

CASE STUDY - INTRODUCTION TO KUBERNETES

You have just joined a startup Ventura Software as a Devops Lead Engineer. The company relies on a Monolithic Architecture for its product. Recently, the senior management was hired. The new CTO insists on having a Microservice Architecture. The Development Team, is working on breaking the Monolith. Meanwhile, you have been asked to host a Test Application on Kubernetes, to understand how it works.

Following things have to be implemented:

1. Deploy an Apache2 deployment of 2 replicas
2. Sample code has been checked-in at the following Git-Hub repo;

<https://github.com/hshar/website.git>.

You have to containerize this code, and push it to Docker Hub. Once done, deploy it on Kubernetes with 2 replicas

3. Deploy Ingress with the following rules:

- i) `*/apache*` should point to the apache pods
 - ii) `*/custom*` should point to the GitHub application
-

Solution:

Sample code has been checked-in at the following GitHub repo:

<https://github.com/hshar/website.git>

You have to containerize this code, and push it to Docker Hub. Once done, deploy it on Kubernetes with 2 replicas

Launch an instance for kubernetes

[EC2](#) > [Instances](#) > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

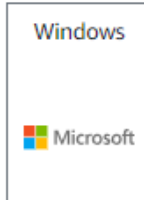
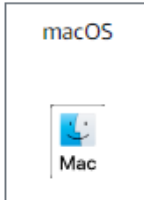
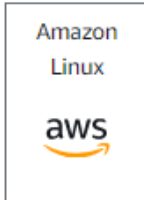
k8s

[Add additional tags](#)

Search our full catalog including 1000s of application and OS images

Recents

Quick Start



Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-01811d4912b4ccb26 (64-bit (x86)) / ami-0f0a689dea27d9dc8 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Architecture

AMI ID

Instance type

t2.medium

Family: t2 2 vCPU 4 GiB Memory Current generation: true

On-Demand RHEL base pricing: 0.0872 USD per Hour

On-Demand Windows base pricing: 0.0764 USD per Hour

On-Demand SUSE base pricing: 0.1584 USD per Hour

On-Demand Linux base pricing: 0.0584 USD per Hour

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Ruchii-kp

Create new key pair

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☐ Create security group

☒ Select existing security group

Common security groups [Info](#)

Select security groups

default sg-05f30ea3543a0ed7a ✕
VPC: vpc-05adb8e0b5c08af51

[Compare security group rules](#)

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Configure storage [Info](#)

[Advanced](#)

Successfully launched an instance

<input type="checkbox"/>	Name ✎ ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status
<input type="checkbox"/>	k8s1	i-04ed21340b555c1c7	✔ Running 🔍 🔍	t2.medium	🕒 Initializing	View alarms +

Connect to instance [Info](#)

Connect to your instance i-Off2969f268692ae4 (k8s) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console



All ports are open to all IPv4 addresses in your security group

All ports are currently open to all IPv4 addresses, indicated by **All** and **0.0.0.0/0** in the inbound [security group](#). For increased security, consider restricting access to only the EC2 Instance Connect IP addresses for your Region: 3.0.5.32/29. [Learn more](#).

Instance ID

i-Off2969f268692ae4 (k8s)

Connection Type

☒ Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ Connect using EC2 Instance Connect client

Connect using the EC2 Instance Connect client, with a private IPv4 address and a

Public IP address

13.213.65.28

Cancel

Connect

Update the machine

```
ubuntu@ip-172-31-25-166:~$ sudo apt-get update
```

i-Off2969f268692ae4 (k8s)

PublicIPs: 13.213.65.28 PrivateIPs: 172.31.25.166

create a file a.sh

```
ubuntu@ip-172-31-25-166:~$ sudo nano a.sh
ubuntu@ip-172-31-25-166:~$
```

i-Off2969f268692ae4 (k8s)

Install kubernetes, docker and minikube

```
sudo apt-get update
sudo apt-get install docker.io -y
sudo systemctl enable --now docker
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
sudo chmod 777 /var/run/docker.sock
minikube start --force --driver=docker
sudo snap install kubectl --classic
minikube status
minikube addons enable ingress
```

Execute the script

```
ubuntu@ip-172-31-25-166:~$ bash a.sh | |
```

```
i-0ff2969f268692ae4 (k8s)
```

You can see ingress is enabled

```
* ingress is an addon maintained by Kubernetes. For any concerns
You can view the list of minikube maintainers at: https://github
- Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen
* Verifying ingress addon...
* The 'ingress' addon is enabled
ubuntu@ip-172-31-44-183:~$
```

```
i-04ed21340b555c1c7 (k8s1)
```

Check if kubernetes properly installed, if not re-enter the command

```
ubuntu@ip-172-31-44-183:~$ sudo snap install kubectl --classic
kubectl 1.30.4 from Canonical ✓ installed
ubuntu@ip-172-31-44-183:~$
```

You can see node is installed

```
ubuntu@ip-172-31-44-183:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	control-plane	5m31s	v1.30.0

```
ubuntu@ip-172-31-44-183:~$
```

Clone your repo

```
ubuntu@ip-172-31-44-183:~$ git clone https://github.com/hshar/website.git
Cloning into 'website'...
remote: Enumerating objects: 8, done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 8 (from 1)
Receiving objects: 100% (8/8), 82.69 KiB | 20.67 MiB/s, done.
Resolving deltas: 100% (1/1), done.
ubuntu@ip-172-31-44-183:~$
```

Successfully cloned the project, go inside the website and create a Dockerfile

```
ubuntu@ip-172-31-44-183:~$ ls
a.sh  snap  website
ubuntu@ip-172-31-44-183:~$ cd website/
ubuntu@ip-172-31-44-183:~/website$ ls
images  index.html
ubuntu@ip-172-31-44-183:~/website$
```

Sudo nano Dockerfile

```
GNU nano 7.2
FROM ubuntu
RUN apt-get update
RUN apt-get install apache2 -y
RUN apt-get install apache2-utils -y
RUN apt-get clean
ENTRYPOINT apachectl -D FOREGROUND
ADD . /var/www/html/
```

Run the below-given command to build an image from the Dockerfile:

```
ubuntu@ip-172-31-44-183:~/website$ sudo docker build -t img .
```

The image will be successfully created. Now run the below command to view the images in the website directory

```
ubuntu@ip-172-31-44-183:~/website$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
img                  latest       948795715c47  58 seconds ago 231MB
ubuntu               latest       edbfe74c41f8  4 weeks ago   78.1MB
gcr.io/k8s-minikube/kicbase v0.0.44     5a6e59a9bdc0  3 months ago  1.26GB
ubuntu@ip-172-31-44-183:~/website$
```

i-04ed21340b555c1c7 (k8s1)

this command will rename the image and create new out of it and run docker images to view the images

```
ubuntu@ip-172-31-44-183:~/website$ sudo docker tag img ruchithabtgowda/k8s-casestudy
ubuntu@ip-172-31-44-183:~/website$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
img                  latest       948795715c47  3 minutes ago 231MB
ruchithabtgowda/k8s-casestudy latest       948795715c47  3 minutes ago 231MB
ubuntu               latest       edbfe74c41f8  4 weeks ago   78.1MB
gcr.io/k8s-minikube/kicbase v0.0.44     5a6e59a9bdc0  3 months ago  1.26GB
ubuntu@ip-172-31-44-183:~/website$
```

i-04ed21340b555c1c7 (k8s1)

To push the image to docker hub first login to docker using below command

```
ubuntu@ip-172-31-44-183:~/website$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you
one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope P
Learn more at https://docs.docker.com/go/access-tokens/

Username: ruchithabtgowda
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-44-183:~/website$
```

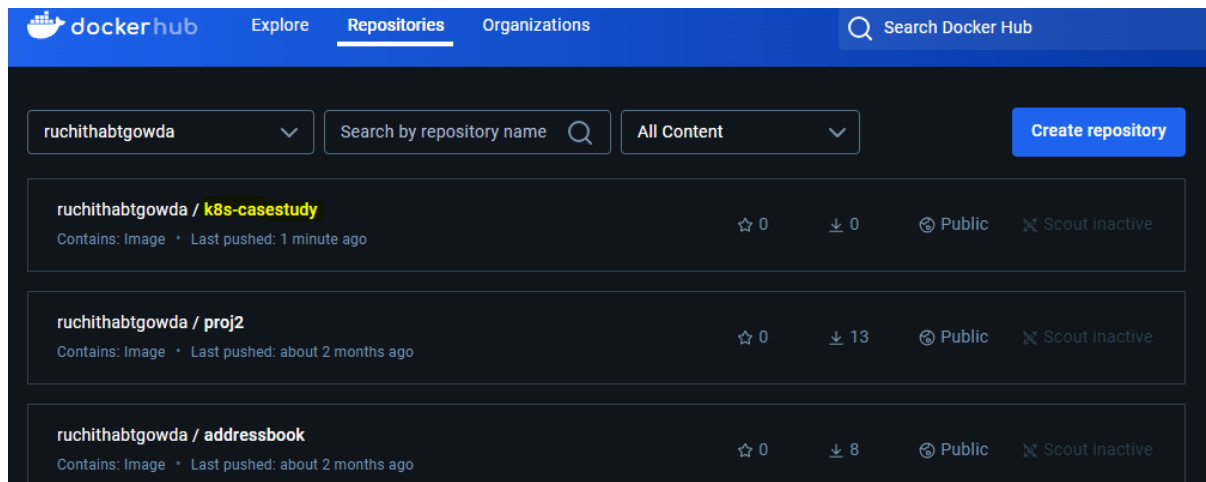
Use the below command to push the image to docker hub

```
ubuntu@ip-172-31-44-183:~/website$ docker push ruchithabtgowda/k8s-casestudy
Using default tag: latest
The push refers to repository [docker.io/ruchithabtgowda/k8s-casestudy]
2b9623bb7156: Pushed
22469224ce21: Pushed
3fd2364fdb57: Pushed
692b4a468249: Pushing [=====>] 56.96MB/113.3MB
4f9aac330f89: Pushing [=====>] 39.76MB
f36fd4bb7334: Waiting
```

Successfully pushed to docker hub

```
ubuntu@ip-172-31-44-183:~/website$ docker push ruchithabtgowda/k8s-casestudy
Using default tag: latest
The push refers to repository [docker.io/ruchithabtgowda/k8s-casestudy]
2b9623bb7156: Pushed
22469224ce21: Pushed
3fd2364fdb57: Pushed
692b4a468249: Pushed
4f9aac330f89: Pushed
f36fd4bb7334: Mounted from library/ubuntu
latest: digest: sha256:f6cc5d5da2ffa18e9435fe9f3e8bd170d6c17b1b46a26f280455862dfe3006c size: 1577
ubuntu@ip-172-31-44-183:~/website$
```

Go to your docker hub and you will find your image



The screenshot shows the Docker Hub interface for the user 'ruchithabtgowda'. The 'Repositories' tab is selected. A list of repositories is displayed, including 'k8s-casestudy', 'proj2', and 'addressbook'. Each repository entry shows its status (e.g., 'Contains: Image'), last pushed time, star count, download count, and visibility (Public). The 'k8s-casestudy' repository is highlighted in yellow.

Repository	Status	Last pushed	Stars	Downloads	Visibility	Scout
ruchithabtgowda / k8s-casestudy	Contains: Image	Last pushed: 1 minute ago	0	0	Public	Scout inactive
ruchithabtgowda / proj2	Contains: Image	Last pushed: about 2 months ago	0	13	Public	Scout inactive
ruchithabtgowda / addressbook	Contains: Image	Last pushed: about 2 months ago	0	8	Public	Scout inactive

Now we need to deploy the Custom Image with 2 Replicas:

First Run the below-given command to create the custom image with 2 replicas

```
kubectl create deployment custom --image=visalntyagi12/k8scasestudy --replicas=2 --port=80
```

```
ubuntu@ip-172-31-44-183:~/website$ kubectl create deployment custom --image=ruchithabtgowda/k8s-casestudy --replicas=2 --port=80
deployment.apps/custom created
ubuntu@ip-172-31-44-183:~/website$
```

i-04ed21340b555c1c7 (k8s1)

A Custom Deployment will be created with Two Pods.

Run the below-given command to check how many deployments are present:

```
ubuntu@ip-172-31-44-183:~/website$ kubectl get deploy -o wide
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
custom	2/2	2	2	65s	k8s-casestudy	ruchithabtgowda/k8s-casestudy	app=custom

```
ubuntu@ip-172-31-44-183:~/website$
```

Check the pods

```
ubuntu@ip-172-31-44-183:~/website$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
custom-b5769bc85-7dth	1/1	Running	0	93s
custom-b5769bc85-njgxn	1/1	Running	0	93s

```
ubuntu@ip-172-31-44-183:~/website$
```

Now we need to deploy the Apache2 with 2 Replicas

Run the below-given command to create the deployment with 2 replicas:

```
kubectl create deployment apache --image=ubuntu/apache2 --replicas=2 --port=80
```

then check the pods available

```
ubuntu@ip-172-31-44-183:~/website$ kubectl create deployment apache --image=ubuntu/apache2 --replicas=2 --port=80
deployment.apps/apache created
ubuntu@ip-172-31-44-183:~/website$ kubectl get deploy -o wide
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
apache	1/2	2	1	10s	apache2	ubuntu/apache2	app=apache
custom	2/2	2	2	2m53s	k8s-casestudy	ruchithabtgowda/k8s-casestudy	app=custom

```
ubuntu@ip-172-31-44-183:~/website$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
apache-5b9b5bff7c-js6pg	1/1	Running	0	19s
apache-5b9b5bff7c-vp42l	1/1	Running	0	19s
custom-b5769bc85-7dth	1/1	Running	0	3m2s
custom-b5769bc85-njgxn	1/1	Running	0	3m2s

```
ubuntu@ip-172-31-44-183:~/website$
```

We need to deploy ingress with the following rules: -

- a. */apache* should point to the Apache pods
- b. */custom* should point to the GitHub application.

Expose Both the Replicas on NodePort for Creating a Service

```
ubuntu@ip-172-31-35-69:~/website$ kubectl expose deploy apache --type=NodePort
service/apache exposed
ubuntu@ip-172-31-35-69:~/website$ kubectl expose deploy custom --type=NodePort
service/custom exposed
ubuntu@ip-172-31-35-69:~/website$ ||
```

i-021f0791e65df49fd (k8s)

PublicIPs: 47.128.223.5 PrivateIPs: 172.31.35.69

This command is used to find the services

```
ubuntu@ip-172-31-35-69:~/website$ kubectl get svc -o wide
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE    SELECTOR
apache        NodePort    10.96.95.7    <none>        80:32755/TCP     75s    app=apache
custom        NodePort    10.108.25.221 <none>        80:30443/TCP     37s    app=custom
kubernetes    ClusterIP   10.96.0.1     <none>        443/TCP          23m    <none>
```

i-021f0791e65df49fd (k8s)

PublicIPs: 47.128.223.5 PrivateIPs: 172.31.35.69

Create ingress.yml file

```
ubuntu@ip-172-31-35-69:~/website$ nano ingress.yml
ubuntu@ip-172-31-35-69:~/website$ ||
```

i-021f0791e65df49fd (k8s)

PublicIPs: 47.128.223.5 PrivateIPs: 172.31.35.69

This is the command for Ingress file

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
    - http:
        paths:
          - path: /apache
            pathType: Prefix
            backend:
              service:
                name: apache
                port:
                  number: 80
          - path: /custom
            pathType: Prefix
            backend:
              service:
                name: custom
                port:
                  number: 80
```

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /apache
        pathType: Prefix
        backend:
          service:
            name: apache
            port:
              number: 80
      - path: /custom
        pathType: Prefix
        backend:
          service:
            name: custom

```

Create an ingress.yml file

```

ubuntu@ip-172-31-44-183:~/website$ sudo nano ingress.yaml
ubuntu@ip-172-31-44-183:~/website$ kubectl create -f ingress.yaml
ingress.networking.k8s.io/ingress created
ubuntu@ip-172-31-44-183:~/website$

```

This command is used to view the Ingress

```

ubuntu@ip-172-31-44-183:~/website$ kubectl get ing

```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress	nginx	*	192.168.49.2	80	31s

```

ubuntu@ip-172-31-44-183:~/website$

```

Run the following command

```
kubectl port-forward service/ingress-nginx-controller -n ingress-nginx --address 0.0.0.0 :443
```

```
ubuntu@ip-172-31-35-69:~/website$ kubectl port-forward service/ingress-nginx-controller -n ingress-nginx --address 0.0.0.0 :443
Forwarding from 0.0.0.0:36387 -> 443
Handling connection for 36387
Handling connection for 36387
Handling connection for 36387
Handling connection for 36387
Handling connection for 36387
Handling connection for 36387
Handling connection for 36387
Handling connection for 36387
```

i-021f0791e65df49fd (k8s)

PublicIPs: 47.128.223.5 PrivateIPs: 172.31.35.69

IP address following with the port number

