PS No. 10844721 **M Assessment** Rahul Prasad Sahu

Q1). Using configuration management tool Ansible create multiple users and confirm on your manage host.

Solution =>

Ansible is a configuration management tool used to configure on OS on an existing infrastructure.

I am going to use Ansible loop for this purpose

To achieve this I am going to launch an Redhat instance(Best works for ansible) and given a hostname ansible-controller

And two remote hosts named it ans-one and ans-two

```
root@ip-172-31-19-212:~
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\10844721> cd downloads
PS C:\Users\10844721\downloads> ssh -i "my-kp.pem" ec2-user@ec2-52-91-100-8.compute-1.amazonaws.com
The authenticity of host 'ec2-52-91-100-8.compute-1.amazonaws.com (52.91.100.8)' can't be established.
ED25519 key fingerprint is SHA256:14KKwCStFmzIZMw99UXR0OjHajKlENSgIeAkARmm/fM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-91-100-8.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Register this system with Red Hat Insights: rhc connect

Example:
# rhc connect --activation-key <key> --organization <org>

The rhc client and Red Hat Insights will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
[ec2-user@ip-172-31-19-212 ~]$ sudo su -
[root@ip-172-31-19-212 ~]# hostnamectl set-hostname ansible-controller.example.com
[root@ip-172-31-19-212 ~]# bash
[root@ansible-controller ~]# hostname
ansible-controller.example.com
[root@ansible-controller ~]#
```

This is for instance ans-one

```
[ec2-user@ip-172-31-29-83 ~]$ sudo su -
[root@ip-172-31-29-83 ~]# hostnamectl set-hostname ans-one.example.com
[root@ip-172-31-29-83 ~]# bash
[root@ans-one ~]# hostname
ans-one.example.com
[root@ans-one ~]#
```

And this is for instance ans-two

```
   part of the FQDN) in the /etc/hosts file.
[ec2-user@ip-172-31-27-31 ~]$ sudo su -
[root@ip-172-31-27-31 ~]#  hostnamectl set-hostname ans-two.example.com
[root@ip-172-31-27-31 ~]# bash
[root@ans-two ~]# hostanme
bash: hostanme: command not found
[root@ans-two ~]# hostname
ans-two.example.com
[root@ans-two ~]#
```

Now to make a communication between ansible controller and remote hosts ssh must be configure so generated a ssh key and pasted into authorized_keys into remote hosts

Since ansible is an agentless so works with ssh and python should be there .

```
[root@ansible-controller ~]# cd .ssh
[root@ansible-controller .ssh]# ll
total 12
-rw-------. 1 root root 553 Oct 30 08:46 authorized_keys
-rw-------. 1 root root 432 Oct 30 08:52 id_ed25519
-rw-r--r--. 1 root root 117 Oct 30 08:52 id_ed25519.pub
[root@ansible-controller .ssh]# cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPXQJOs1W1a9XOSfeOUZOD/TeiPcghWBzWPSd9p/bHJQ root@ansible-controller.example.com
[root@ansible-controller .ssh]#
```

Once we are done with ssh we will make a ansible.cfg (configuration file) for ansible

```
# paramiko on older platforms rather than removing it
#ssh_args = -o ControlMaster=auto -o ControlPersist=60s

# The path to use for the ControlPath sockets. This defaults to
# "%(directory)s/ansible-ssh-%%h-%%p-%%r", however on some systems with
# very long hostnames or very long path names (caused by long user names or
# deeply nested home directories) this can exceed the character limit on
# file socket names (108 characters for most platforms). In that case, you
# may wish to shorten the string below.
#
# Example:
# control_path = %(directory)s/%%h-%%r
#control_path = %(directory)s/ansible-ssh-%%h-%%p-%%r

# Enabling pipelining reduces the number of SSH operations required to
# execute a module on the remote server. This can result in a significant
# performance improvement when enabled, however when using "sudo:" you must
# first disable 'requiretty' in /etc/sudoers
#
# By default, this option is disabled to preserve compatibility with
# sudoers configurations that have requiretty (the default on many distros).
#
#pipelining = False

# if True, make ansible use scp if the connection type is ssh
# (default is sftp)
#scp_if_ssh = True

[accelerate]
accelerate_port = 5099
accelerate_timeout = 30
accelerate_connect_timeout = 5.0
-- INSERT --                                                    178,33        Bot
```

Putting my two remote servers into hosts file

```
[dev-server]
172.31.29.83

[prod-server]
172.31.27.31
-- INSERT --                                                    58,13         Bot
```

As I can see that controller is all set with the remote hosts

```
[root@ansible-controller ansible]# ansible all --list-hosts
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
  hosts (2):
    172.31.29.83
    172.31.27.31
```

Creating a playbook named user.yaml which will create multiple users into remote hosts

```
---
- name: creating users
  hosts: all
  tasks:
   - name: Creating user
     user:
      name: "{{ item }}"
      state: present
     with_items:
      - soni
      - vikash
      - rahul
~
~
~
```

To check the syntax used this command since yaml follow indentation

```
[root@ansible-controller ansible]# ansible-playbook user.yaml --syntax-check
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
[DEPRECATION WARNING]: The 'smart' option for connections is deprecated. Set the connection plugin directly instead. This feature
will be removed in version 2.20. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

playbook: user.yaml
```

After checking syntax executed this playbook and can see the yellow coloured text that represent that it worked.

```
[root@ans-two ~]# exit
logout
Connection to 172.31.27.31 closed.
[root@ansible-controller ansible]# ansible-playbook user.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
[DEPRECATION WARNING]: The 'smart' option for connections is deprecated. Set the connection plugin directly instead. This feature
will be removed in version 2.20. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

PLAY [creating users] ***********************************************************************************************

TASK [Gathering Facts] **********************************************************************************************
[WARNING]: Platform linux on host 172.31.29.83 is using the discovered Python interpreter at /usr/bin/python3.9, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.29.83]
[WARNING]: Platform linux on host 172.31.27.31 is using the discovered Python interpreter at /usr/bin/python3.9, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.27.31]

TASK [Creating user] ************************************************************************************************
changed: [172.31.29.83] => (item=soni)
changed: [172.31.27.31] => (item=soni)
changed: [172.31.29.83] => (item=vikash)
changed: [172.31.27.31] => (item=vikash)
changed: [172.31.29.83] => (item=rahul)
changed: [172.31.27.31] => (item=rahul)

PLAY RECAP **********************************************************************************************************
172.31.27.31               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.29.83               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[root@ansible-controller ansible]#
```

Now checking those users into the remote hosts

First checking into ans-one and all the users are there

```
[root@ans-one ~]# tail /etc/passwd
systemd-timesync:x:995:995:systemd Time Synchronization:/:/usr/sbin/nologin
chrony:x:994:994:chrony system user:/var/lib/chrony:/sbin/nologin
ec2-instance-connect:x:993:993::/home/ec2-instance-connect:/sbin/nologin
stapunpriv:x:159:159:systemtap unprivileged user:/var/lib/stapunpriv:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
soni:x:1001:1001::/home/soni:/bin/bash
vikash:x:1002:1002::/home/vikash:/bin/bash
rahul:x:1003:1003::/home/rahul:/bin/bash
[root@ans-one ~]#
```

And all the users are also int ans-two

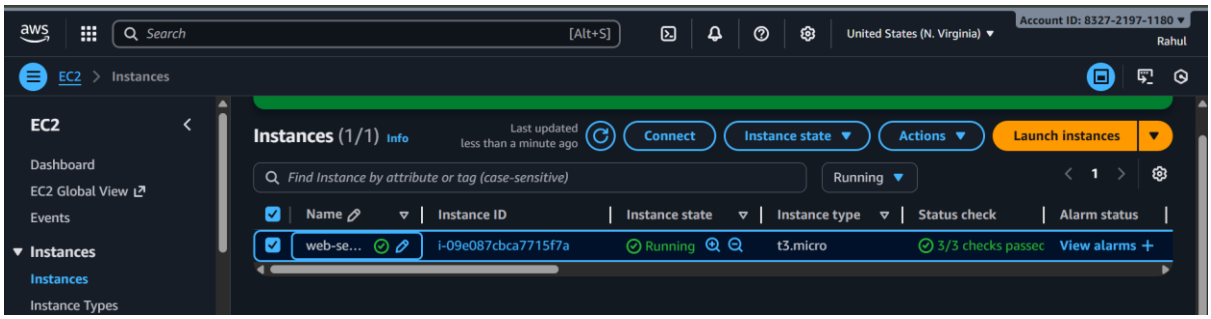
```
[root@ans-two .ssh]# tail /etc/passwd
systemd-timesync:x:995:995:systemd Time Synchronization:/:/usr/sbin/nologin
chrony:x:994:994:chrony system user:/var/lib/chrony:/sbin/nologin
ec2-instance-connect:x:993:993::/home/ec2-instance-connect:/sbin/nologin
stapunpriv:x:159:159:systemtap unprivileged user:/var/lib/stapunpriv:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
soni:x:1001:1001::/home/soni:/bin/bash
vikash:x:1002:1002::/home/vikash:/bin/bash
rahul:x:1003:1003::/home/rahul:/bin/bash
[root@ans-two .ssh]#
```

****

Q2). We have an EC2 instance in N. Virginia Region where a web server is running. create custom image of this server and launch new server in Ohio region using of this image

Solution =>

Launching an instance into N.virigia Region



Setting hostname for this instance

```
root@ip-172-31-27-31:~
[root@ans-two ~]# hostnamectl set-hostname web-server-nv.example.com
[root@ans-two ~]# bash
[root@web-server-nv ~]# hostname
web-server-nv.example.com
```

Installed httpd using => yum install httpd -y

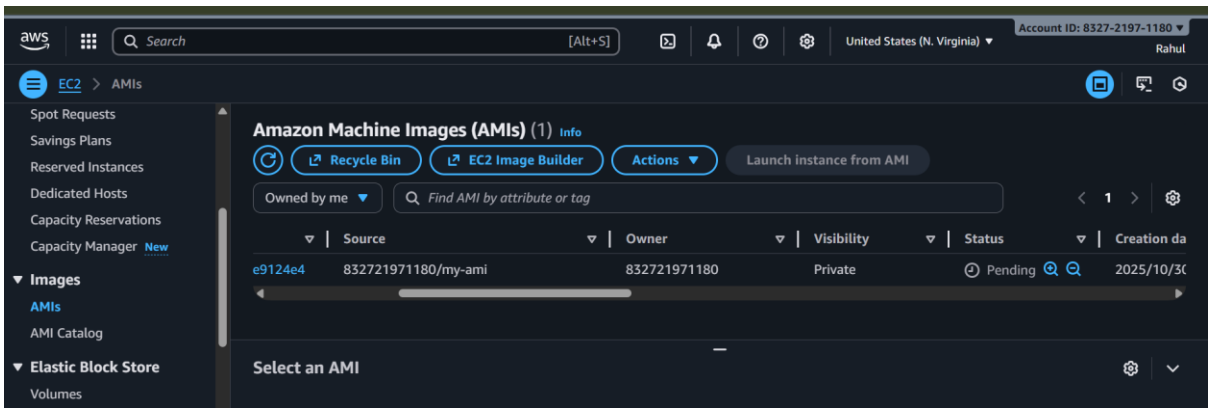
And after installation created an index.html file which will act as an application

So started and enabled this service and can see that my application is reachable with the ip

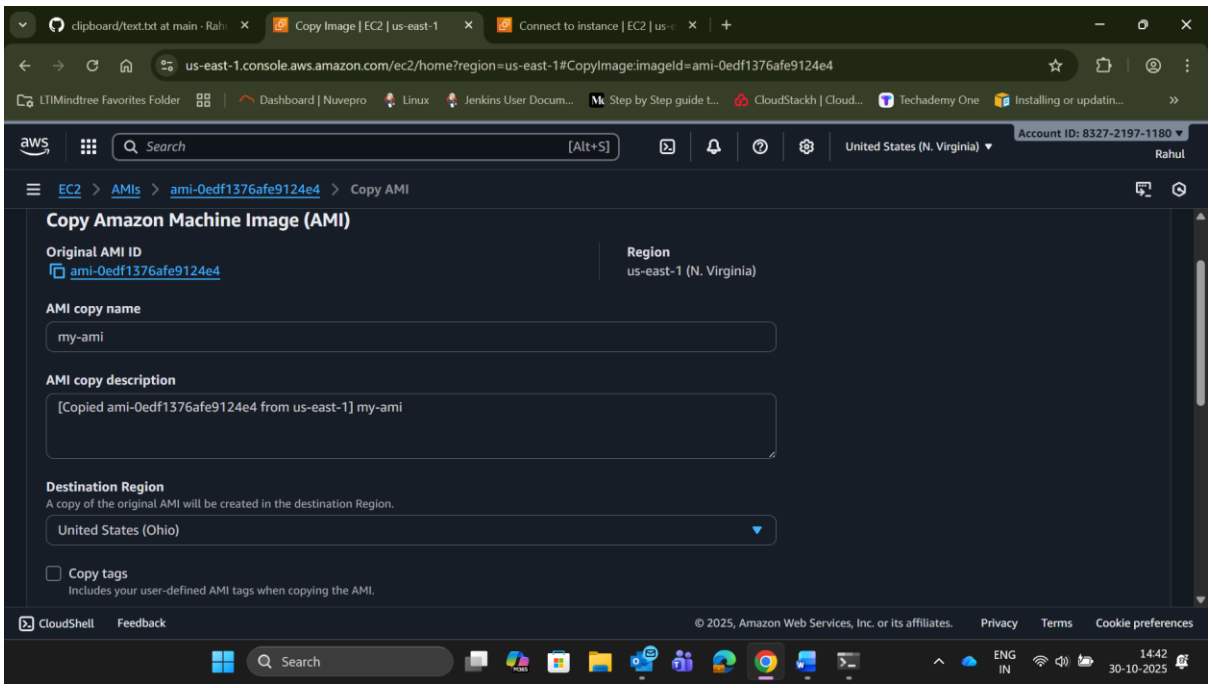
```
[root@web-server-nv ~]# echo "hi this is my web-app" > /var/www/html/index.html
[root@web-server-nv ~]# systemctl start httpd
[root@web-server-nv ~]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@web-server-nv ~]# curl http://localhost
hi this is my web-app
[root@web-server-nv ~]#
```

Since it is running into n.virginia and I want to use this into ohio so for that purpose I am creating an image from this instance.

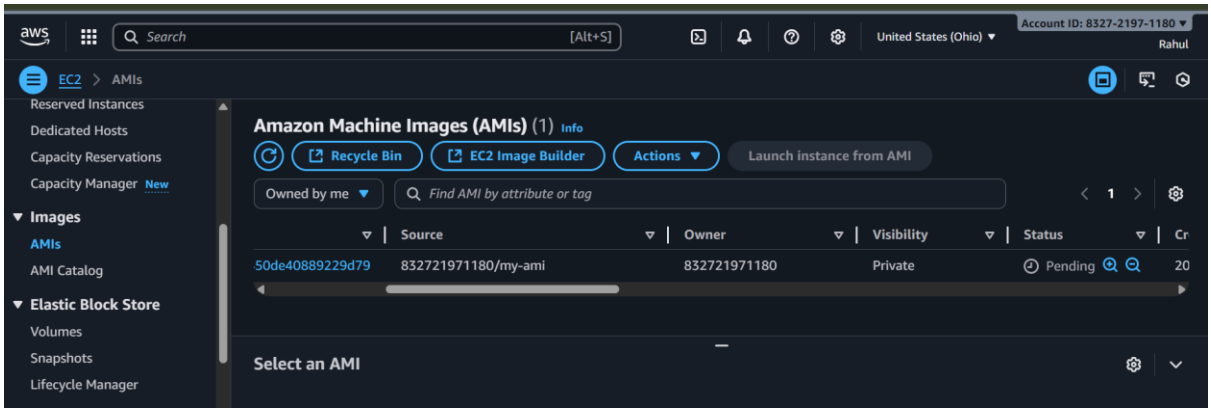
So before creating image it is recommended to stop before image creation



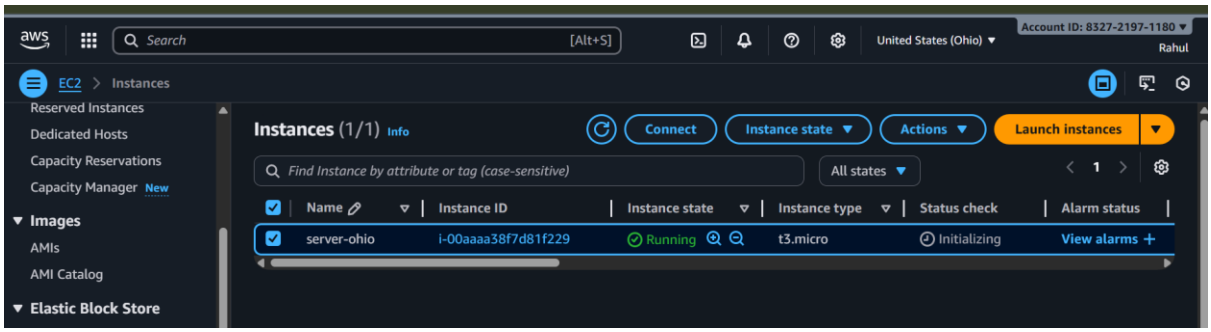
Copying this image into ohio region



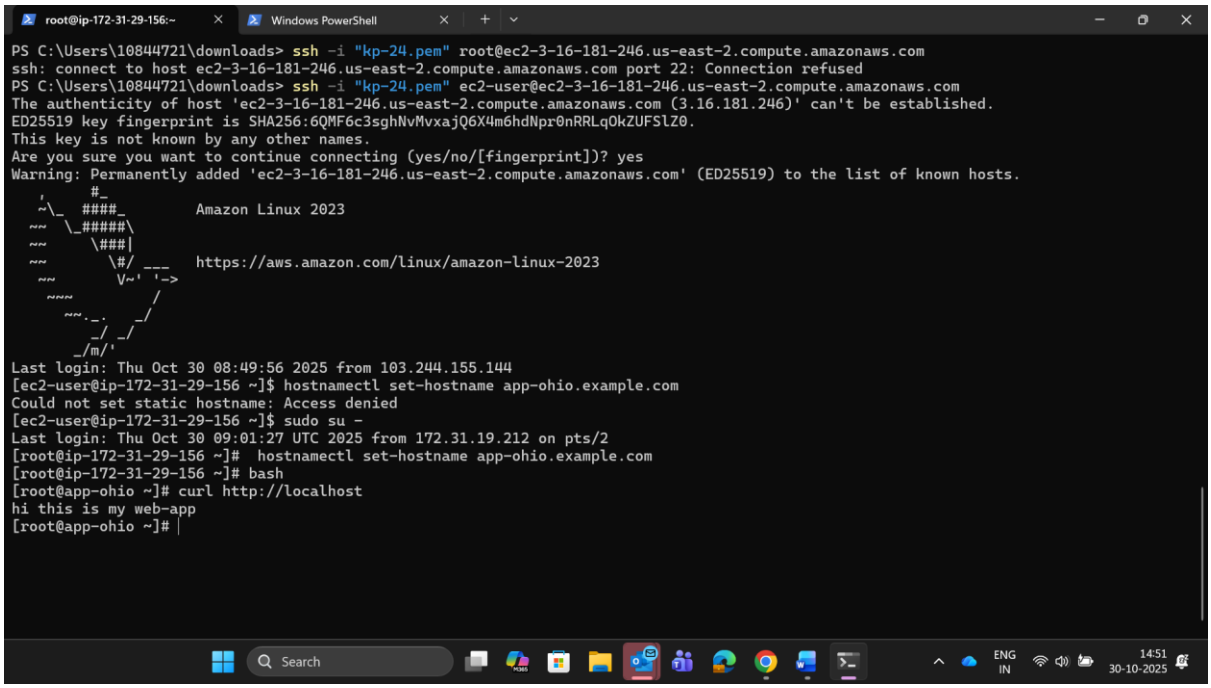
Now I am able to that image into ohio region



So now launched an instance with that image and named it server-ohio



After giving name to the host we are able to access the same application without download or starting and enabling it .



Q3). Pull the Ubuntu image from Docker hub and Launch a web application in the container on port no. 8080 and this application should be reachable globally

Solution =>

Launched an instance and set the hostname to docker.example.com.

This instance will act as a docker host

```
[root@ip-172-31-19-174 ~]# hostnamectl set-hostname docker.example.com
[root@ip-172-31-19-174 ~]# bash
[root@docker ~]# hostname
docker.example.com
[root@docker ~]#
```

Installed docker into this machine and started and enabled it.

After that pulled ubuntu image from docker hub

```
[root@docker ~]# systemctl start docker
syst[root@docker ~]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@docker ~]# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
4b3ffd8ccb52: Pull complete
Digest: sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[root@docker ~]#
```

I used this command to see all the available images into this docker host

And using that image launched a container named my-app and also enabled port forwarding so that it will also be accessible globally using 8080 port (I also enabled this port into inbound security group). And assigning it a shell where we will run commands.

```
[root@docker ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    97bed23a3497   4 weeks ago   78.1MB
[root@docker ~]# docker run -it --name my-app -p 8080:80 ubuntu:latest /bin/bash
root@9e763ed4b730:/#
```

After updating ubuntu installed apache2 and creating an index.html file and given some content.

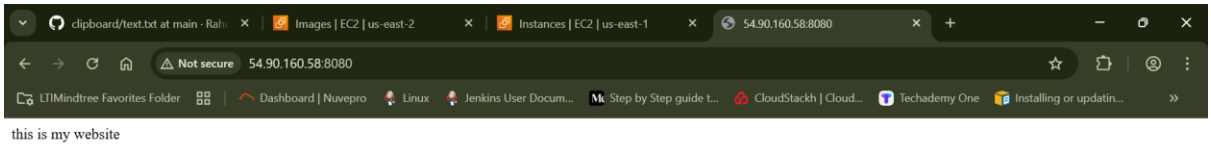
```
root@9e763ed4b730:/# echo "this is my website" > /var/www/html/index.html
```

After that started and enabled apache2 and try to access inside docker host with the ip of the container 172.17.0.2

And can see that web page is accessible so now will try to access it globally

```
root@9e763ed4b730:/# service apache2 start
 * Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' direc
tive globally to suppress this message
 *
root@9e763ed4b730:/# read escape sequence
[root@docker ~]# curl http://172.17.0.2
this is my website
[root@docker ~]#
```

Now accessing with the help of public ip of that machine with port no 8080 and can see that page is accessible globally.



****

Q4). Deploy a web application in the kubernetes pod. And create a replica set. In any case load is going to increase on your replica set. increase the number of replica of the pods

Solution =>

To deploy a web application (will use nginx) in Kubernetes first I will create a cluster

One Control plane and one worker node.

```
ubuntu@ip-172-31-23-130:~$ sudo su -
root@ip-172-31-23-130:~# hostnamectl set-hostname control-plane.example.com
root@ip-172-31-23-130:~# bash
root@control-plane:~# vim k8s.sh
root@control-plane:~#
```

Created a script file(k8s.sh) for creating the control plane

```
root@control-plane: ~

echo "CRI runtime installed successfully"

# Add Kubernetes APT repository and install required packages
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.g
pg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.32/deb/ /' | sudo tee /etc/ap
t/sources.list.d/kubernetes.list

sudo apt-get update -y
sudo apt-get install -y kubelet="1.32.0-*" kubectl="1.32.0-*" kubeadm="1.32.0-*"
sudo apt-get update -y
sudo apt-get install -y jq

sudo systemctl enable --now kubelet
sudo systemctl start kubelet

## Execute ONLY on "Master Node"

sudo kubeadm config images pull

sudo kubeadm init

mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config


# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml


:wq!
```

Assigned permission to execute this script

```
root@control-plane:~# chmod +x k8s.sh
root@control-plane:~# ./k8s.sh
```

After setting of control plane hosted one more server and named it worker

```
ubuntu@ip-172-31-20-171:~$ sudo su -
root@ip-172-31-20-171:~# hostnamectl set-hostname worker.example.com
root@ip-172-31-20-171:~# bash
root@worker:~#
```

To connect worker node with the control node we will require this kubeadm token so will save it for further usecase

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.23.130:6443 --token 7bs6f1.r6nu75y3fks2jfjh \
        --discovery-token-ca-cert-hash sha256:c49a8db8529c8ba6a5b106e52235b4350e035c34ddeed16a9fe4b31db26f56e9
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
```

Also created a script(worker.sh) and assigned permission to execute

```
ubuntu@ip-172-31-20-171:~$ sudo su -
root@ip-172-31-20-171:~# hostnamectl set-hostname worker.example.com
root@ip-172-31-20-171:~# bash
root@worker:~# vim worker.sh
root@worker:~# chmod +x worker.sh
root@worker:~# ./worker.sh
```

Paste that copied token into this script

```
root@control-plane: ~   root@worker: ~

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-a
pt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee
/etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"

# Add Kubernetes APT repository and install required packages
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.g
pg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.32/deb/ /' | sudo tee /etc/ap
t/sources.list.d/kubernetes.list

sudo apt-get update -y
sudo apt-get install -y kubelet="1.32.0-*" kubectl="1.32.0-*" kubeadm="1.32.0-*"
sudo apt-get update -y
sudo apt-get install -y jq

sudo systemctl enable --now kubelet
sudo systemctl start kubelet
##Note Plz replace this token from your cluster Token
kubeadm join 172.31.23.130:6443 --token 7bs6f1.r6nu75y3fks2jfjh \
        --discovery-token-ca-cert-hash sha256:c49a8db8529c8ba6a5b106e52235b4350e035c34ddeed16a9fe4b31db26f56e9
~
-- INSERT --                                                                    52,111        Bot
```

Now can see our control and worker machine is ready

```
root@control-plane:~# kubectl get nodes
NAME                    STATUS   ROLES           AGE     VERSION
control-plane.example.com   Ready    control-plane   3m53s   v1.32.0
worker.example.com          Ready    <none>          14s     v1.32.0
root@control-plane:~#
```

So Now creating a replicaset with the help of manifest file which is a yaml (yet another markup language).

```
root@control-plane:~# vim replicaset.yaml
root@control-plane:~#
```

Can see apiversion and kind for the replicaset with the help of command

Kubectl api-resources

From these information I have puted the kind and version

And the name of the replicaset is nginx-replicaset and also given it a label so that in future it can be used to group the resources that have the same label like app:web

And replica is one which means it will create only one replica of a pod and total will be one

Selector label is used for the purpose on which pod it will going to apply replication controller.

And finally template is for pod. I am using nginx image for my web application .

It will be accessible with the port 80.

```
root@control-plane: ~    root@worker: ~
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: nginx-replicaset
 labels:
   app: web
spec:
 replicas: 1
 selector:
  matchLabels:
    app: web
 template:
   metadata:
     labels:
       app: web
   spec:
    containers:
    - name: web
      image: nginx:1.14
      ports:
      - containerPort: 80

~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    10,15          All
```

After applying with the command line tool kubectl for accessing k8s cluster

Can see the pod where which will manage the container and container will manage the application

```
root@control-plane:~# kubectl apply -f replicaset.yaml
replicaset.apps/nginx-replicaset created
root@control-plane:~# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
nginx-replicaset-5c2v4  1/1     Running   0          6s
root@control-plane:~#
```

Can see more details about this pod where it is deployed and on which port and which image is being used and controlled by which controller.

```
root@control-plane:~# kubectl describe pod nginx-replicaset-5c2v4
Name:             nginx-replicaset-5c2v4
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker.example.com/172.31.20.171
Start Time:       Thu, 30 Oct 2025 09:51:31 +0000
Labels:           app=web
Annotations:      cni.projectcalico.org/containerID: 77b441f21db29a0e6cefb601c11aa9e843d6f44a07db7ed7a785cf375b9720e0
                  cni.projectcalico.org/podIP: 192.168.1.69/32
                  cni.projectcalico.org/podIPs: 192.168.1.69/32
Status:           Running
IP:               192.168.1.69
IPs:
  IP:           192.168.1.69
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  web:
    Container ID:   cri-o://a7504c2c712a6c3788c94ea33563ba0bad158dd0bf625a539f6ec06ca57f9af0
    Image:          nginx:1.14
    Image ID:       docker.io/library/nginx@sha256:706446e9c6667c0880d5da3f39c09a6c7d2114f5a5d6b74a2fafd24ae30d2078
    Port:           80/TCP
```

Initially only one pod was running so will try to scale this with the controller hpa (horizontal pod autoscaling).

So for that purpose created one more manifest file and named it hpa.yaml

```
root@control-plane:~# vim hpa.yaml
root@control-plane:~#
```

Can see the api version and Kind of controller and target on which it will going to apply hpa so which is nginx-replicaset .

And will scale pod on the behalf of cpu utilization.

So flow is like our hpa will manage the replicaset and replicaset will manage the pods

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
 name: my-app-hpa
spec:
 scaleTargetRef:
   apiVersion: apps/v1
   kind: ReplicaSet
   name: nginx-replicaset
 minReplicas: 4
 maxReplicas: 10
 metrics:
   - type: Resource
     resource:
       name: cpu
       target:
         type: Utilization
         averageUtilization: 25
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    18,32          All
```

After applying this hpa we can see our replicaset(rs) and hpa.

```
root@control-plane:~# kubectl get rs
NAME                 DESIRED   CURRENT   READY   AGE
nginx-replicaset     4         4         4       3m54s
root@control-plane:~# kubectl get hpa
NAME         REFERENCE                     TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
my-app-hpa   ReplicaSet/nginx-replicaset   cpu: <unknown>/25%   4         10        4          107s
root@control-plane:~#
```

Now initially only one pod was there but after some seconds can see it scaled to 4.

```
root@control-plane:~# kubectl apply -f hpa.yaml
horizontalpodautoscaler.autoscaling/my-app-hpa created
root@control-plane:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-5c2v4   1/1     Running   0          2m23s
root@control-plane:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-5c2v4   1/1     Running   0          2m49s
nginx-replicaset-5vcvd   1/1     Running   0          22s
nginx-replicaset-r6frt   1/1     Running   0          22s
nginx-replicaset-sbsgw   1/1     Running   0          22s
root@control-plane:~#
```

To see the more details about this newly created pods used this command.

```
root@control-plane:~# kubectl describe pod nginx-replicaset-sbsgw
Name:             nginx-replicaset-sbsgw
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker.example.com/172.31.20.171
Start Time:       Thu, 30 Oct 2025 09:53:58 +0000
Labels:           app=web
Annotations:      cni.projectcalico.org/containerID: 658a8573c4b95da4721da37c5d152721b2c59deceaad0d7d22c5dfc005c4bac7
                  cni.projectcalico.org/podIP: 192.168.1.72/32
                  cni.projectcalico.org/podIPs: 192.168.1.72/32
Status:           Running
IP:               192.168.1.72
IPs:
  IP:           192.168.1.72
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  web:
    Container ID:   cri-o://96fe303d2305a6f5328db40a93482192849ebbf3067d708f4ad292d340830653
    Image:          nginx:1.14
    Image ID:       docker.io/library/nginx@sha256:706446e9c6667c0880d5da3f39c09a6c7d2114f5a5d6b74a2fafd24ae30d2078
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 30 Oct 2025 09:53:59 +0000
```

Now upon deletion of pod it will again create the new pod.

But in production it is not recommended to directly control pod with replicaset controller because upon deletion of replicaset no pod will be created. It will be gone forever. So will use Deployment.

THANK YOU

**********End**********