Lab: Bootstrap Kubernetes Cluster Using Ansible

Introduction:

Kubernetes is one of the most popular **open-source** and **enterprise-ready** container orchestration systems. It's used to automate the deployment, scaling, and management of containerized applications. Manual **Kubernetes installation** is a laborious and error-prone process. However, it can be dramatically simplified by using configuration management tools such as Ansible. In this Lab let's Learn how to deploy a full-function Kubernetes cluster using Ansible with our installation package

Cluster Designing

Our Kubernetes cluster consists of three servers. One of them will be working as Kubernetes **Controller**. The other two are **worker** nodes. All servers are in the same internal network, 192.168.100.0/24. Software components the cluster depends on are **Kubernetes**, **Etcd**, **Docker**, **WeaveNet**

The following is our server inventory

Hostname	IP Address	Roles
1. eoc-controller	192.168.100.150	Controller
2. eoc-node1	192.168.100.151	Worker node
3. eoc-node2	192.168.100.152	Worker node

Objectives

- **Verifying Communication with Kubernetes**
- **Pre-requisites to install the Kubernetes Cluster**
- **Required Images to work**
- **Verifying controller status using Ad-Hoc command**
- **Join token capture**
- **Adding controller and worker nodes using Ad-Hoc command**
- **Verifying cluster status using Ad-Hoc command**
- **Node role**

Note: Login to **eoc-controller** as **admin** user with password as **linux**

1. Verifying Communication with Kubernetes

1.1 Let's create Ansible inventory file **kube-infra** to tell Ansible how to communicate with the Kubernetes controller and worker nodes.

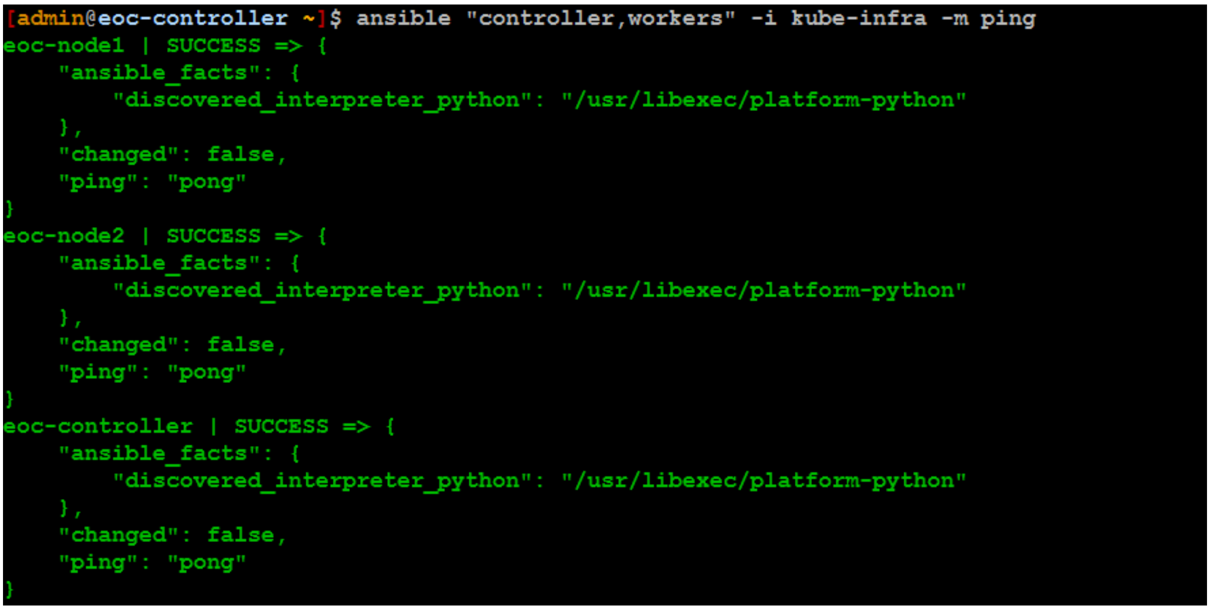
```
# cat > kube-infra  << EOF
[controller]
eoc-controller
[workers]
eoc-node1
eoc-node2
EOF
```

Note: Listing the controller node and the worker nodes in different sections in the hosts file will allow us to target the playbooks at the specific node type later on.

1.2 We can test it's working by doing a Ansible ping.

```
# ansible "controller,workers" -i kube-infra -m ping
```

Output:

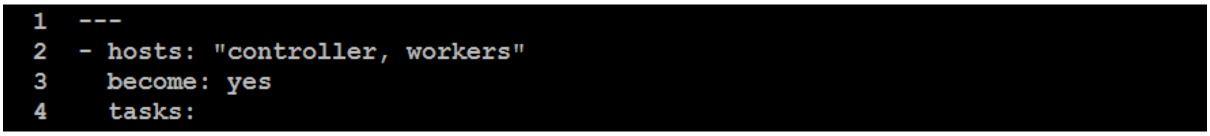


```
[admin@eoc-controller ~]$ ansible "controller,workers" -i kube-infra -m ping
eoc-node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
eoc-node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
eoc-controller | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

2. Pre-requisites to install the Kubernetes Cluster

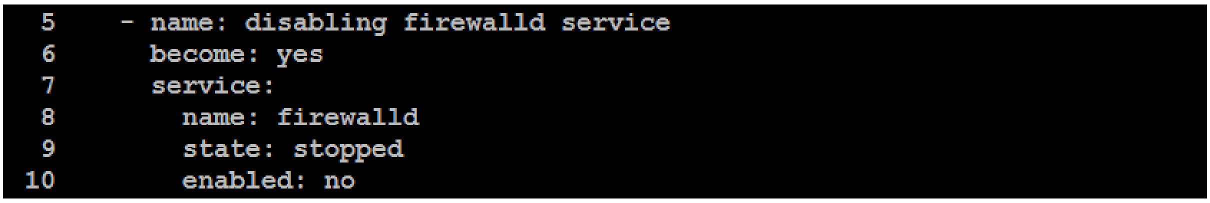
2.1 Let's deploy the pre-requisites to install the kubernetes cluster by creating a yaml named **kube-pre-requisites.yml**

Let's target the host and declaring variables



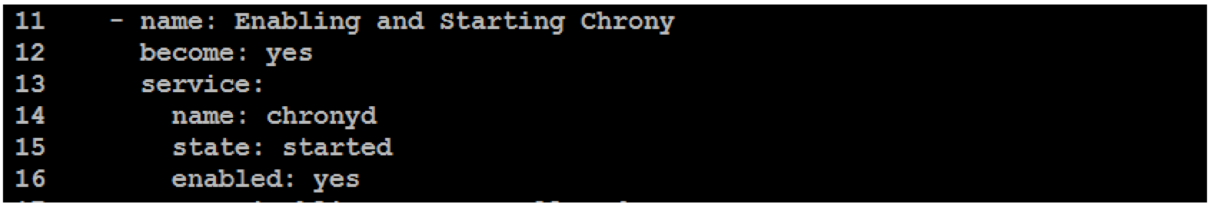
```
---
- hosts: "controller, workers"
  become: yes
  tasks:
```

2.2 Let’s disable the firewalld service.



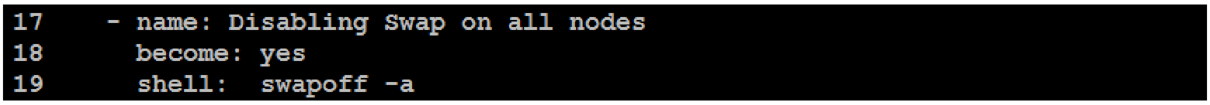
```
  - name: disabling firewalld service
    become: yes
    service:
      name: firewalld
      state: stopped
      enabled: no
```

2.3 Enable and start the chronyd service.



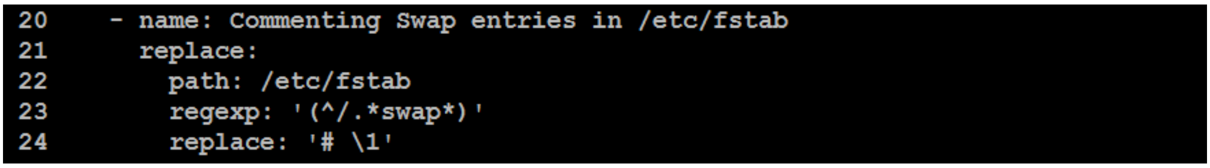
```
  - name: Enabling and Starting Chrony
    become: yes
    service:
      name: chronyd
      state: started
      enabled: yes
```

2.4 Disabling swap on all nodes.



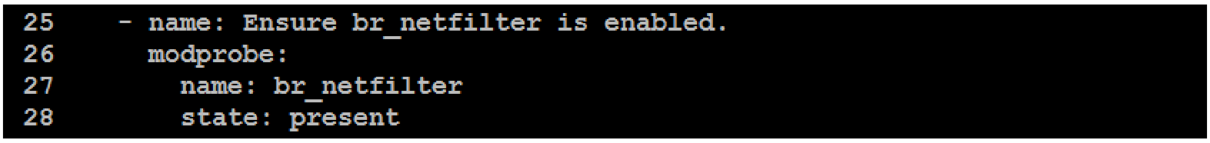
```
  - name: Disabling Swap on all nodes
    become: yes
    shell:  swapoff -a
```

2.5 Commenting out swap functionality completely.



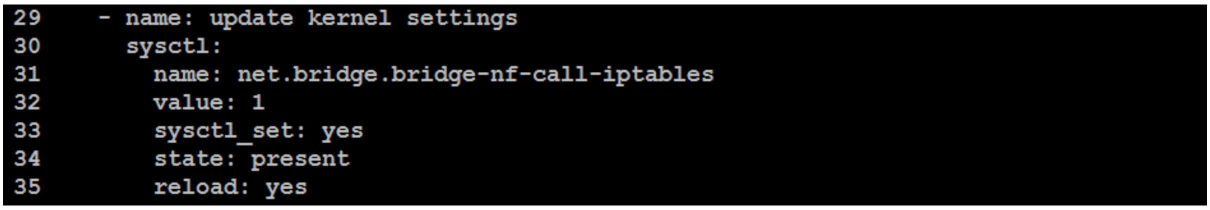
```
  - name: Commenting Swap entries in /etc/fstab
    replace:
      path: /etc/fstab
      regexp: '(^/.*swap*)'
      replace: '# \1'
```

2.6 Enabling br_netfilter.



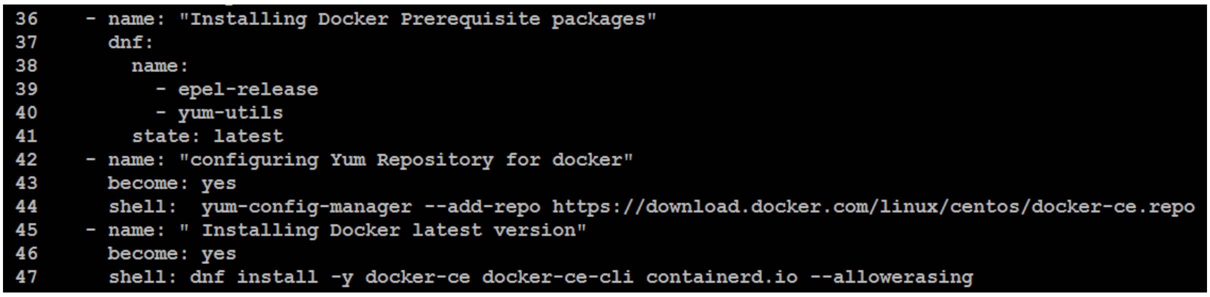
```
  - name: Ensure br_netfilter is enabled.
    modprobe:
      name: br_netfilter
      state: present
```

2.7 Updating kernal settings.



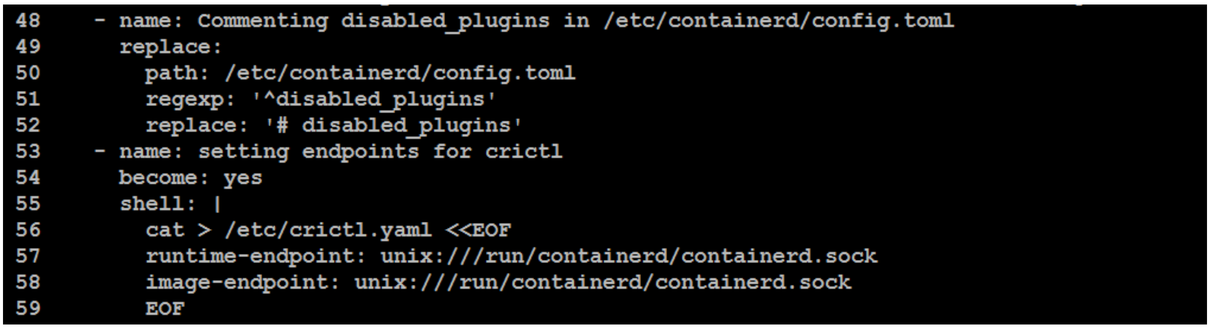
```
  - name: update kernel settings
    sysctl:
      name: net.bridge.bridge-nf-call-iptables
      value: 1
      sysctl_set: yes
      state: present
      reload: yes
```

2.8 Installing and configuring docker.



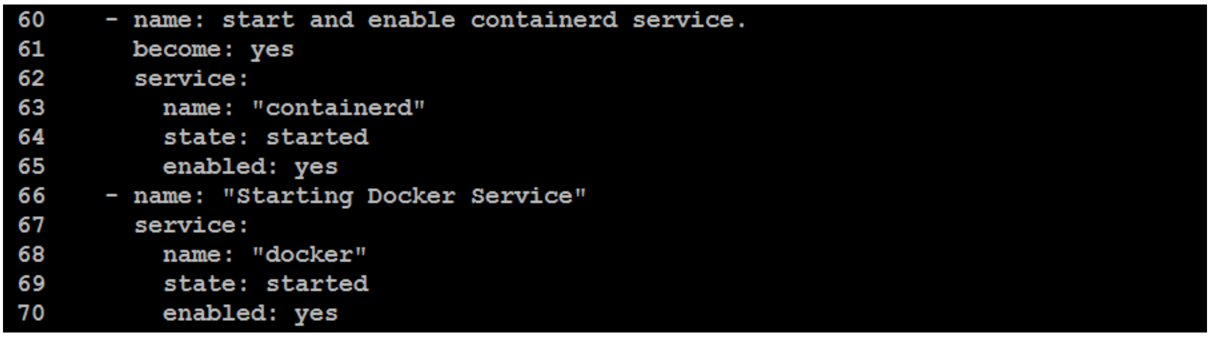
```
  - name: "Installing Docker Prerequisite packages"
    dnf:
      name:
        - epel-release
        - yum-utils
      state: latest
  - name: "configuring Yum Repository for docker"
    become: yes
    shell:  yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
  - name: " Installing Docker latest version"
    become: yes
    shell: dnf install -y docker-ce docker-ce-cli containerd.io --allowerasing
```

2.9 Configuring crictl.



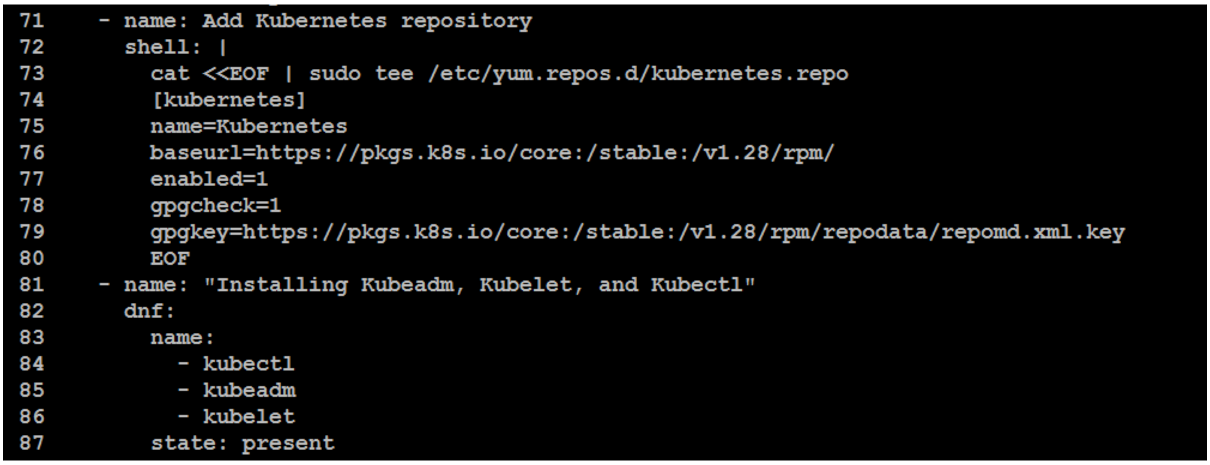
```
  - name: Commenting disabled_plugins in /etc/containerd/config.toml
    replace:
      path: /etc/containerd/config.toml
      regexp: '^disabled_plugins'
      replace: '# disabled_plugins'
  - name: setting endpoints for crictl
    become: yes
    shell: |
      cat > /etc/crictl.yaml <<EOF
      runtime-endpoint: unix:///run/containerd/containerd.sock
      image-endpoint: unix:///run/containerd/containerd.sock
      EOF
```

2.10 Enabling docker and containerd.



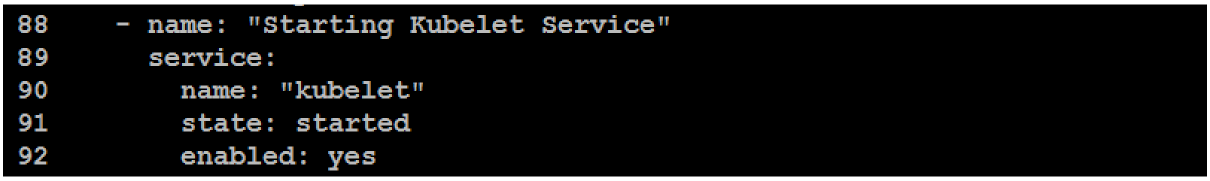
```
  - name: start and enable containerd service.
    become: yes
    service:
      name: "containerd"
      state: started
      enabled: yes
  - name: "Starting Docker Service"
    service:
      name: "docker"
      state: started
      enabled: yes
```

2.11 Adding Kubernetes repo and installing.



```
  - name: Add Kubernetes repository
    shell: |
      cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
      [kubernetes]
      name=Kubernetes
      baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
      enabled=1
      gpgcheck=1
      gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
      EOF
  - name: "Installing Kubeadm, Kubelet, and Kubectl"
    dnf:
      name:
        - kubectl
        - kubeadm
        - kubelet
      state: present
```

2.12 Starting the kubelet service.

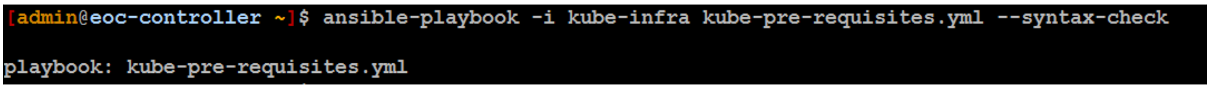


```
  - name: "Starting Kubelet Service"
    service:
      name: "kubelet"
      state: started
      enabled: yes
```

2.13 Let's verify the syntax of the yaml **kube-pre-requisites.yml.**

```
# ansible-playbook -i kube-infra kube-pre-requisites.yml --syntax-check
```

Output:

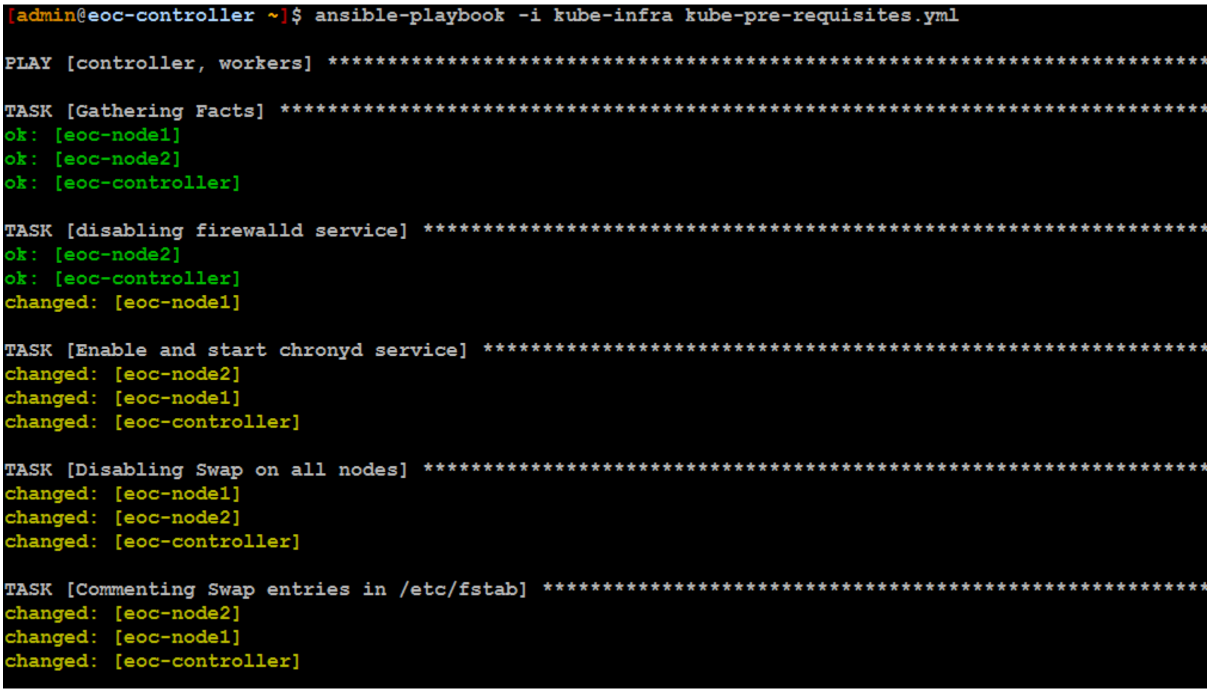


```
[admin@eoc-controller ~]$ ansible-playbook -i kube-infra kube-pre-requisites.yml --syntax-check

playbook: kube-pre-requisites.yml
```

2.14 Let's run the Playbook **kube-pre-requisites.yml.**

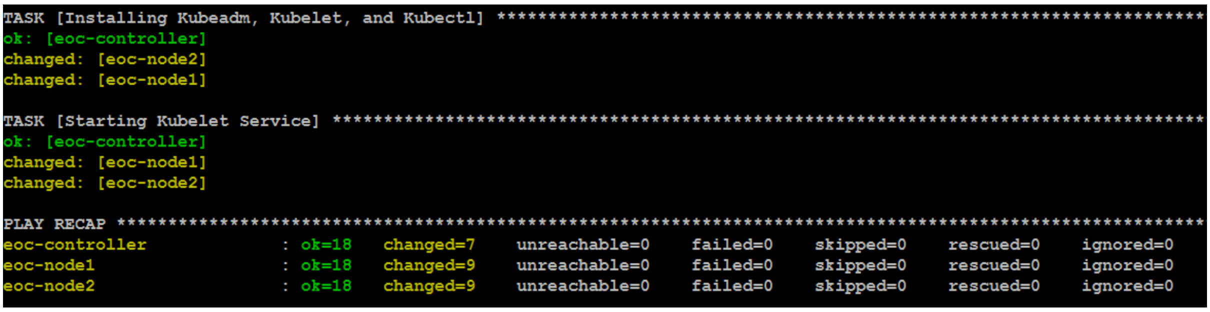
```
# ansible-playbook -i kube-infra kube-pre-requisites.yml
```

Output:



```
[admin@eoc-controller ~]$ ansible-playbook -i kube-infra kube-pre-requisites.yml

PLAY [controller, workers] *************************************************************************

TASK [Gathering Facts] *****************************************************************************
ok: [eoc-node1]
ok: [eoc-node2]
ok: [eoc-controller]

TASK [disabling firewalld service] *****************************************************************
ok: [eoc-node2]
ok: [eoc-controller]
changed: [eoc-node1]

TASK [Enable and start chronyd service] ************************************************************
changed: [eoc-node2]
changed: [eoc-node1]
changed: [eoc-controller]

TASK [Disabling Swap on all nodes] *****************************************************************
changed: [eoc-node1]
changed: [eoc-node2]
changed: [eoc-controller]

TASK [Commenting Swap entries in /etc/fstab] *******************************************************
changed: [eoc-node2]
changed: [eoc-node1]
changed: [eoc-controller]
```

……..



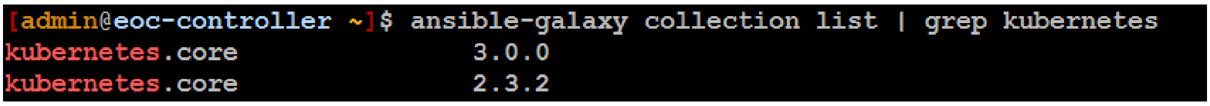
```
TASK [Installing Kubeadm, Kubelet, and Kubectl] ****************************************************************
ok: [eoc-controller]
changed: [eoc-node2]
changed: [eoc-node1]

TASK [Starting Kubelet Service] ********************************************************************************
ok: [eoc-controller]
changed: [eoc-node1]
changed: [eoc-node2]

PLAY RECAP *****************************************************************************************************
eoc-controller             : ok=18   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
eoc-node1                  : ok=18   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
eoc-node2                  : ok=18   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

3. Required Images to work

3.1 Add the ansible-galaxy collection of Kubernetes module if not present.

```
# ansible-galaxy collection list | grep kubernetes
```

Output:

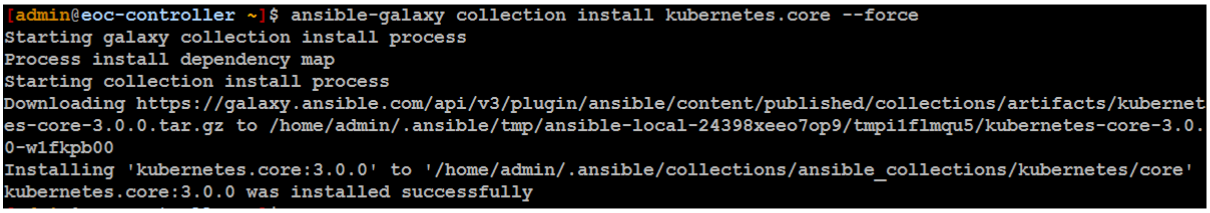


```
[admin@eoc-controller ~]$ ansible-galaxy collection list | grep kubernetes
kubernetes.core                    3.0.0
kubernetes.core                    2.3.2
```

Info: If not present use this command to install or download the latest one.

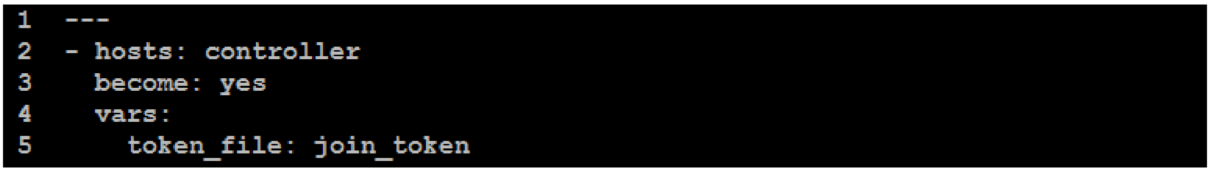
```
# ansible-galaxy collection install kubernetes.core --force
```

Output:



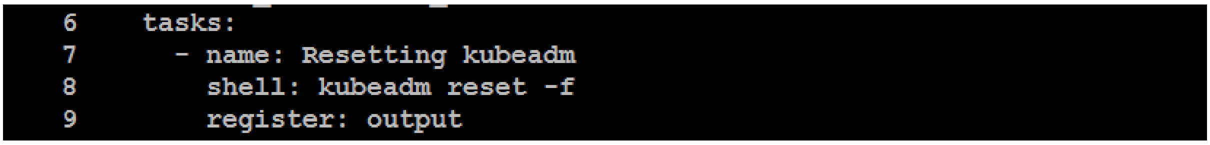
```
[admin@eoc-controller ~]$ ansible-galaxy collection install kubernetes.core --force
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/api/v3/plugin/ansible/content/published/collections/artifacts/kubernet
es-core-3.0.0.tar.gz to /home/admin/.ansible/tmp/ansible-local-24398xeeo7op9/tmpi1flmqu5/kubernetes-core-3.0.
0-w1fkpb00
Installing 'kubernetes.core:3.0.0' to '/home/admin/.ansible/collections/ansible_collections/kubernetes/core'
kubernetes.core:3.0.0 was installed successfully
```

3.2 Let's pull the images requires to work as a control-plane on eoc-node1 by creating a yaml named **bootstraping-controller.yml.**



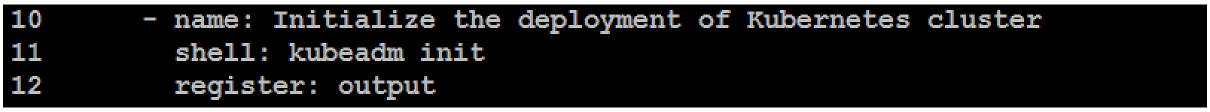
```
---
- hosts: controller
  become: yes
  vars:
    token_file: join_token
```

3.3 Let set the task if something goes wrong it will reset the kubeadm.



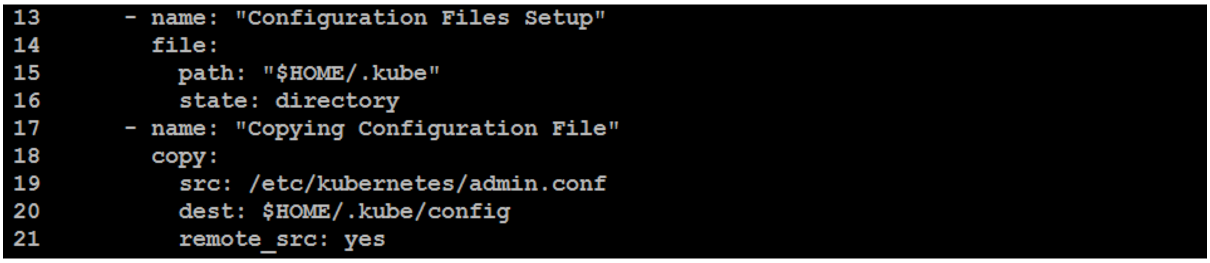
```
  tasks:
    - name: Resetting kubeadm
      shell: kubeadm reset -f
      register: output
```

3.4 Let initialize the kubernetes cluster.



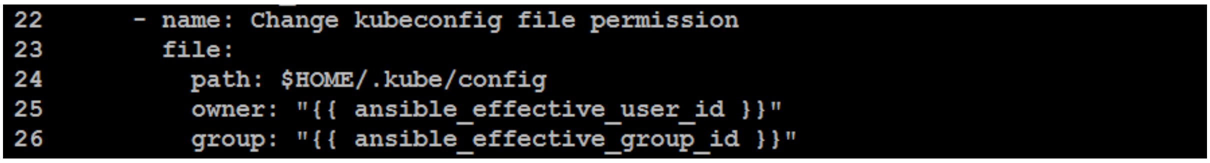
```
    - name: Initialize the deployment of Kubernetes cluster
      shell: kubeadm init
      register: output
```

3.5 Setting up the configuration files.



```
    - name: "Configuration Files Setup"
      file:
        path: "$HOME/.kube"
        state: directory
    - name: "Copying Configuration File"
      copy:
        src: /etc/kubernetes/admin.conf
        dest: $HOME/.kube/config
        remote_src: yes
```

3.6 Changing kubeconfig file permissions.

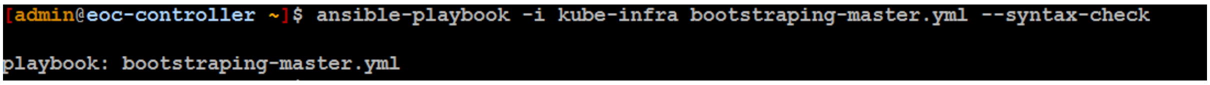


```
    - name: Change kubeconfig file permission
      file:
        path: $HOME/.kube/config
        owner: "{{ ansible_effective_user_id }}"
        group: "{{ ansible_effective_group_id }}"
```

3.7 Let's verify the syntax of the yaml **bootstraping-controller.yml**

```
# ansible-playbook -i kube-infra bootstraping-controller.yml
--syntax-check
```

Output:

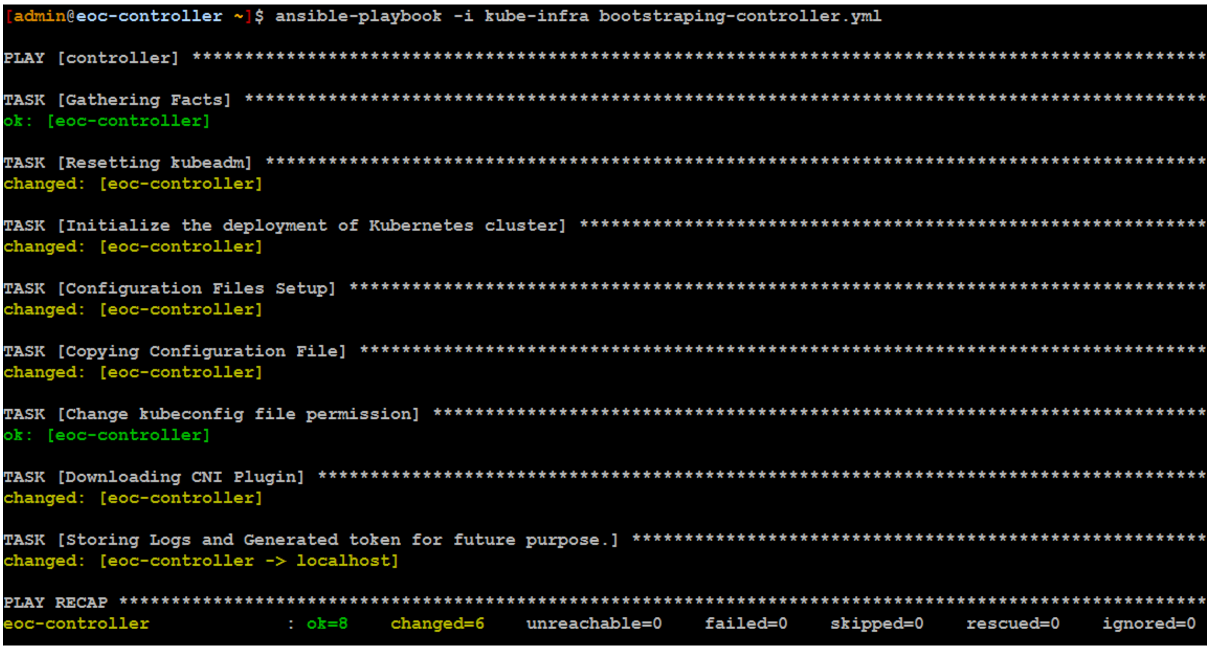


```
[admin@eoc-controller ~]$ ansible-playbook -i kube-infra bootstraping-master.yml --syntax-check

playbook: bootstraping-master.yml
```

3.8 Let's run the Playbook **bootstraping-controller.yml.**

```
# ansible-playbook -i kube-infra bootstraping-controller.yml
```

Output:



```
[admin@eoc-controller ~]$ ansible-playbook -i kube-infra bootstraping-controller.yml

PLAY [controller] *******************************************************************************

TASK [Gathering Facts] **************************************************************************
ok: [eoc-controller]

TASK [Resetting kubeadm] ************************************************************************
changed: [eoc-controller]

TASK [Initialize the deployment of Kubernetes cluster] ******************************************
changed: [eoc-controller]

TASK [Configuration Files Setup] ****************************************************************
changed: [eoc-controller]

TASK [Copying Configuration File] ***************************************************************
changed: [eoc-controller]

TASK [Change kubeconfig file permission] ********************************************************
ok: [eoc-controller]

TASK [Downloading CNI Plugin] *******************************************************************
changed: [eoc-controller]

TASK [Storing Logs and Generated token for future purpose.] *************************************
changed: [eoc-controller -> localhost]

PLAY RECAP **************************************************************************************
eoc-controller             : ok=8    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

3.9 Installing the cni plugin.

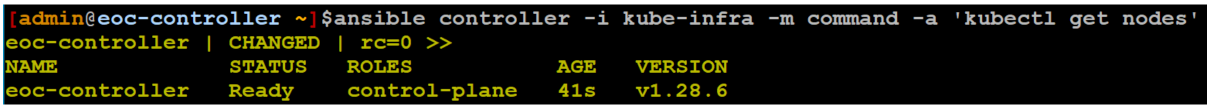
```
# ansible controller -i kube-infra -m command -a 'kubectl
apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1
/weave-daemonset-k8s-1.11.yaml'
```

4. Verifying the cluster.

4.1 Let's verify the controller status using ad-hoc command.

```
# ansible controller -i kube-infra -m command -a 'kubectl
get nodes'
```

Output:

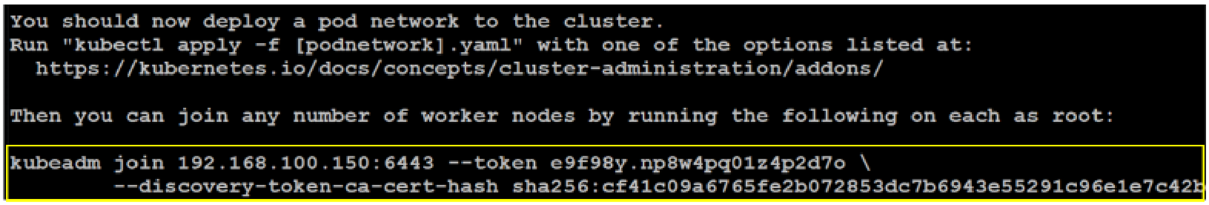


```
[admin@eoc-controller ~]$ansible controller -i kube-infra -m command -a 'kubectl get nodes'
eoc-controller | CHANGED | rc=0 >>
NAME             STATUS   ROLES           AGE   VERSION
eoc-controller   Ready    control-plane   41s   v1.28.6
```

4.2 Get the join token from the output saved from the above playbook.

```
# cat join_token
```

Output:

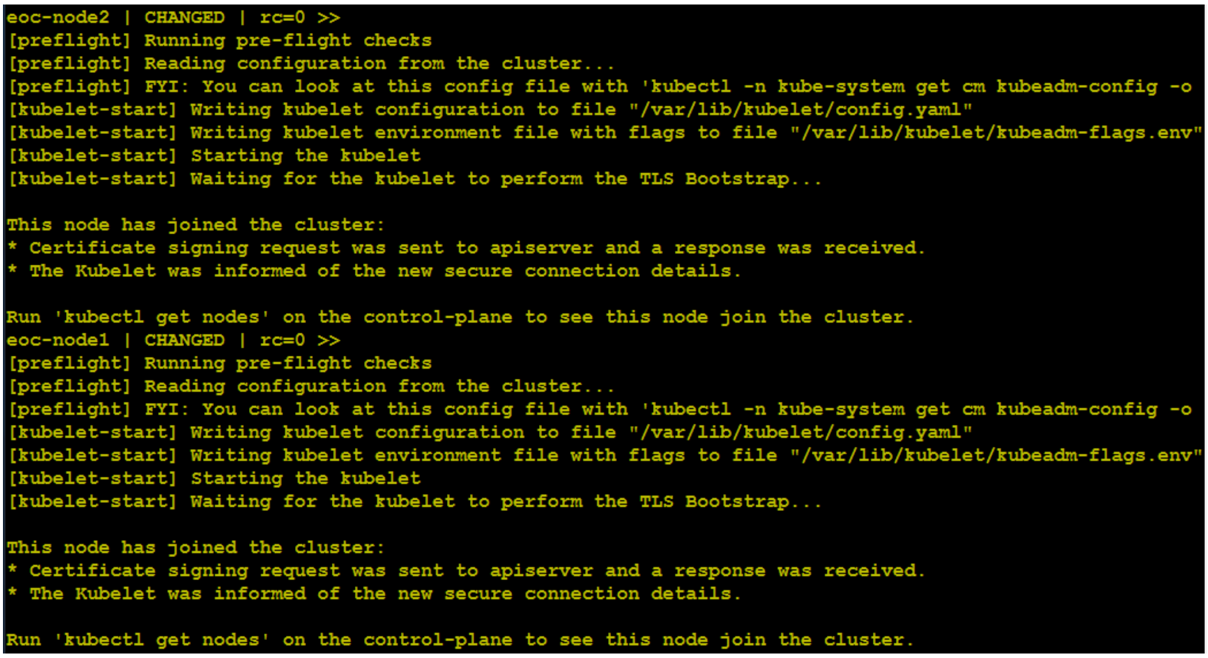


```
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

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

kubeadm join 192.168.100.150:6443 --token e9f98y.np8w4pq01z4p2d7o \
        --discovery-token-ca-cert-hash sha256:cf41c09a6765fe2b072853dc7b6943e55291c96e1e7c42b
```

4.3 Let’s join the worker nodes to the controller using ad-hoc command.

```
# ansible workers -i kube-infra -m command -a 'kubeadm join
192.168.100.150:6443 --token 6ms418.ytixb4sn41v7d6b8 --
discovery-token-ca-cert-hash
sha256:b44451fd3211c0b2615a87a1738c5e23c1b498a9bea697208138f
ca2a5c8728e'
```

Output:



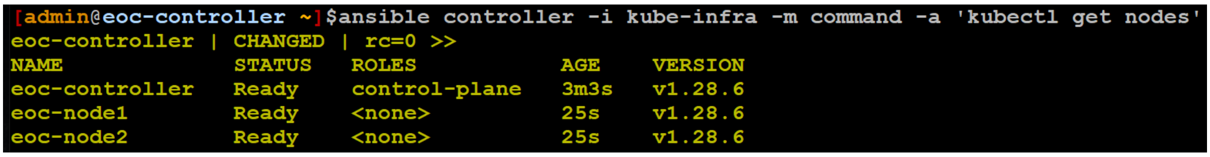
```
eoc-node2 | CHANGED | rc=0 >>
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
eoc-node1 | CHANGED | rc=0 >>
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

5. Verifying cluster status using Ad-Hoc command

5.1 Let’s verify the Cluster status using ad-hoc command.

```
# ansible controller -i kube-infra -m command -a 'kubectl
get nodes'
```

Output:



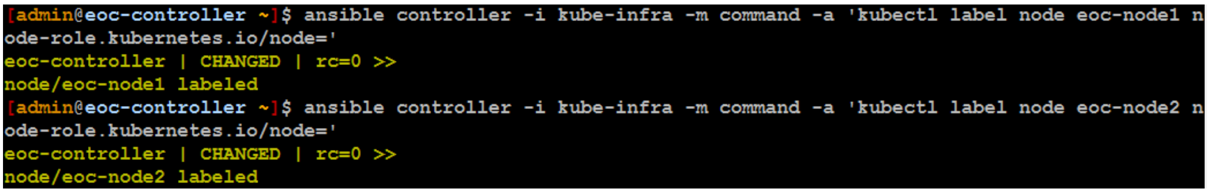
```
[admin@eoc-controller ~]$ansible controller -i kube-infra -m command -a 'kubectl get nodes'
eoc-controller | CHANGED | rc=0 >>
NAME             STATUS   ROLES           AGE    VERSION
eoc-controller   Ready    control-plane   3m3s   v1.28.6
eoc-node1        Ready    <none>          25s    v1.28.6
eoc-node2        Ready    <none>          25s    v1.28.6
```

5.2 Let’s label the nodes by executing below command.

```
# ansible controller -i kube-infra -m command -a 'kubectl
label node eoc-node1 node-role.kubernetes.io/node='
```

```
# ansible controller -i kube-infra -m command -a 'kubectl
label node eoc-node2 node-role.kubernetes.io/node='
```

Output:

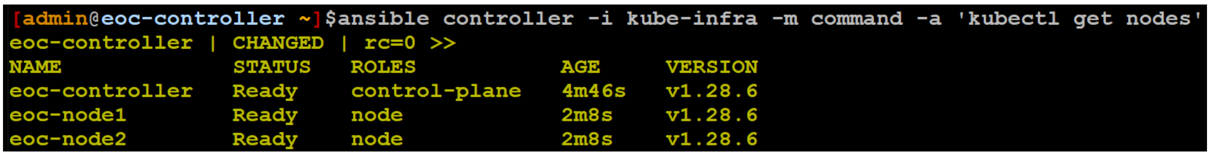


```
[admin@eoc-controller ~]$ ansible controller -i kube-infra -m command -a 'kubectl label node eoc-node1 n
ode-role.kubernetes.io/node='
eoc-controller | CHANGED | rc=0 >>
node/eoc-node1 labeled
[admin@eoc-controller ~]$ ansible controller -i kube-infra -m command -a 'kubectl label node eoc-node2 n
ode-role.kubernetes.io/node='
eoc-controller | CHANGED | rc=0 >>
node/eoc-node2 labeled
```

5.3 Let’s verify the Cluster status using ad-hoc command

```
# ansible controller -i kube-infra -m command -a 'kubectl
get nodes'
```

Output:



```
[admin@eoc-controller ~]$ansible controller -i kube-infra -m command -a 'kubectl get nodes'
eoc-controller | CHANGED | rc=0 >>
NAME             STATUS   ROLES           AGE     VERSION
eoc-controller   Ready    control-plane   4m46s   v1.28.6
eoc-node1        Ready    node            2m8s    v1.28.6
eoc-node2        Ready    node            2m8s    v1.28.6
```