Lab: Ansible Role to Bootstrap Kubernetes Cluster

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

Objective:

- **Writing role with ansible-galaxy**
- **Verifying Communication with Kubernetes**
- **Creating 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**

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

1. Writing role with ansible-galaxy

1.1 The first step in creating a role is creating its directory structure. In order to create the base directory structure, we're going to use a tool called **ansible-galaxy**:

```
# ansible-galaxy init roles/pre-requisites --offline
```

Output:

```
[admin@eoc-controller ~]$ ansible-galaxy init roles/pre-requisites --offline
- Role roles/pre-requisites was created successfully
```

1.2 This command will create an pre-requisites directory with the following structure let's use the tree command to view.

```
# tree roles/pre-requisites
```

Output:

2. Verifying Communication with Kubernetes

2.1 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.

2.2 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/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
eoc-node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
eoc-controller | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

3. Creating Pre-requisites to Install the Kubernetes Cluster

3.1 Let's create a role to disable firewalld service.

Path → roles/pre-requisites/tasks/c_swap.yml

```
[admin@eoc-controller ~]$ cat -n roles/pre-requisites/tasks/c_swap.yml
     1  ---
     2  - name: Comment out swap entries in /etc/fstab
     3    ansible.builtin.replace:
     4      path: /etc/fstab
     5      regexp: '^(.* swap .*)$'
     6      replace: '#\1'
```

3.2 Let's create a role to turn-off swap.

Path → roles/pre-requisites/tasks/swapoff.yml

```
[admin@eoc-controller ~]$ cat -n roles/pre-requisites/tasks/swapoff.yml
     1  ---
     2  - name: Disabling Swap on all nodes
     3    become: yes
     4    shell:  swapoff -a
```

3.3 Loading br_netfilter.

Path → roles/pre-requisites/tasks/net_filter.yml

```
[admin@eoc-controller ~]$cat roles/pre-requisites/tasks/net_filter.yml
---
- name: Load br_netfilter module
  ansible.builtin.modprobe:
    name: br_netfilter
```

3.4 Setting bridge-nf-call-iptables to 1.

Path → roles/pre-requisites/tasks/val_1.yml

```
[admin@eoc-controller ~]$ cat -n roles/pre-requisites/tasks/val_1.yml
     1  ---
     2  - name: Set bridge-nf-call-iptables to 1
     3    ansible.builtin.sysctl:
     4      name: net.bridge.bridge-nf-call-iptables
     5      value: 1
     6      state: present
```

3.5 Let's import the tasks into the main file.

Path → roles/pre-requisites/tasks/main.yml

```
[admin@eoc-controller ~]$cat roles/pre-requisites/tasks/main.yml
---
- import_tasks: c_swap.yml
- import_tasks: swapoff.yml
- import_tasks: net_filter.yml
- import_tasks: val_1.yml
```

3.6 Let's edit the meta **main.yaml** to add the information about the roles like author, description, license, platform supported.

```
# vim roles/pre-requisites/meta/main.yml
```

Output:

```
galaxy_info:
  author: your name
  description: your role description
  company: your company (optional)

  # If the issue tracker for your role is not on github, uncomment the
  # next line and provide a value
  # issue_tracker_url: http://example.com/issue/tracker

  # Choose a valid license ID from https://spdx.org - some suggested licenses:
  # - BSD-3-Clause (default)
  # - MIT
  # - GPL-2.0-or-later
  # - GPL-3.0-only
  # - Apache-2.0
  # - CC-BY-4.0
  license: license (GPL-2.0-or-later, MIT, etc)

  min_ansible_version: 2.9
```

Change the above reuired lines in the file.

```
galaxy_info:
  author: john
  description: pre-requisites to install kubernetes
  company: your company (optional)

  # If the issue tracker for your role is not on github, uncomment the
  # next line and provide a value
  # issue_tracker_url: http://example.com/issue/tracker

  # Choose a valid license ID from https://spdx.org - some suggested licenses:
  # - BSD-3-Clause (default)
  # - MIT
  # - GPL-2.0-or-later
  # - GPL-3.0-only
  # - Apache-2.0
  # - CC-BY-4.0
  license: license (GPL-2.0-or-later, MIT, etc)
```

3.7 Let's view all the files using tree command.

```
# tree roles/pre-requisites/
```

Output:

```
[admin@eoc-controller ~]$ tree roles/pre-requisites/
roles/pre-requisites/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   ├── c_swap.yml
│   ├── main.yml
│   ├── net_filter.yml
│   ├── swapoff.yml
│   └── val_1.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

8 directories, 12 files
```

3.8 We have got the required files for pre-requisites roles. Let's apply this role into the ansible playbook "**run-prerequisites.yml**" as below to deploy it on the client nodes.

```
[admin@eoc-controller ~]$ cat -n run-prerequisites.yml
     1  ---
     2  - hosts: "controller, workers"
     3    roles:
     4      - pre-requisites
```

3.9 Let's verify if there are any syntax errors.

```
# ansible-playbook -i kube-infra --syntax-check run-
prerequisites.yml
```

Output:

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

playbook: run-prerequisites.yml
```

3.10 Let's deploy the roles by executing below commnad.

```
# ansible-playbook -i kube-infra run-prerequisites.yml
```

Output:

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

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

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

TASK [pre-requisites : Comment out swap entries in /etc/fstab] *****************************
changed: [eoc-node2]
changed: [eoc-node1]
changed: [eoc-controller]

TASK [pre-requisites : Disabling swap on all nodes] ****************************************
changed: [eoc-node1]
changed: [eoc-node2]
changed: [eoc-controller]

TASK [pre-requisites : Load br_netfilter module] *******************************************
changed: [eoc-node2]
changed: [eoc-controller]
changed: [eoc-node1]

TASK [pre-requisites : Set bridge-nf-call-iptables to 1] ***********************************
changed: [eoc-node2]
changed: [eoc-node1]
changed: [eoc-controller]

PLAY RECAP *********************************************************************************
eoc-controller             : ok=5    changed=4    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
eoc-node1                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
eoc-node2                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
```

4. Creating role to install and configure docker.

4.1 Creating another directory sturcture for this role.

```
# ansible-galaxy init roles/installing --offline
```

Output:

```
[admin@eoc-controller ~]$ ansible-galaxy init roles/installing --offline
- Role roles/installing was created successfully
```

4.2 Role to remove the buildah package.

Path → roles/installing/tasks/buildah.yml

```
[admin@eoc-controller ~]$ cat -n  roles/installing/tasks/buildah.yml
     1  ---
     2  - name: Remove buildah package
     3    ansible.builtin.dnf:
     4      name: buildah
     5      state: absent
```

4.3 Installing the pre-requisites for the docker.

Path → roles/installing/tasks/pre-req.yml

```
[admin@eoc-controller ~]$ cat -n  roles/pre-requisites/tasks/pre-req.yml
     1  ---
     2  - name: Install yum-utils package
     3    ansible.builtin.dnf:
     4      name: yum-utils
     5      state: present
```

4.4 Adding docker repo.

Path → roles/installing/tasks/add_repo.yml

```
[admin@eoc-controller ~]$ cat -n  roles/pre-requisites/tasks/add_repo.yml
     1  ---
     2  - name: adding repo
     3    become: yes
     4    shell: yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-
ce.repo
```

4.5 Installing docker.

Path → roles/installing/tasks/install.yml

```
[admin@eoc-controller ~]$ cat -n  roles/pre-requisites/tasks/install.yml
     1  ---
     2  - name: Install Docker packages
     3    ansible.builtin.yum:
     4      name:
     5        - docker-ce
     6        - docker-ce-cli
     7        - containerd.io
     8        - docker-compose-plugin
     9      state: present
```

4.6 Configuring containerd dedfault package.

Path → roles/installing/tasks/config.yml

```
[admin@eoc-controller ~]$ cat -n  roles/pre-requisites/tasks/config.yml
     1  ---
     2  - name: config
     3    become: yes
     4    shell: containerd config default | sudo tee /etc/containerd/config.toml
```

4.7 Set SystemdCgroup to true in /etc/containerd/config.toml

Path → roles/installing/tasks/c_config.yml

```
[admin@eoc-controller ~]$ cat -n  roles/pre-requisites/tasks/c_config.yml
     1  ---
     2  - name: Set SystemdCgroup to true in /etc/containerd/config.toml
     3    ansible.builtin.replace:
     4      path: /etc/containerd/config.toml
     5      regexp: 'SystemdCgroup = false'
     6      replace: 'SystemdCgroup = true'
```

4.8 Enable and start the docker service.

Path → roles/installing/tasks/svc.yml

```
[admin@eoc-controller ~]$ cat -n  roles/pre-requisites/tasks/svc.yml
     1  ---
     2  - name: Enable and start Docker service
     3    ansible.builtin.systemd:
     4      name: docker
     5      enabled: yes
     6      state: started
```

4.9 Edit the **main.yml** file and import the tasks.

Path → roles/installing/tasks/main.yml

```
[admin@eoc-controller ~]$ cat -n  roles/installing/tasks/main.yml
     1  ---
     2  - import_tasks: buildah.yml
     3  - import_tasks: pre-req.yml
     4  - import_tasks: add_repo.yml
     5  - import_tasks: install.yml
     6  - import_tasks: config.yml
     7  - import_tasks: c_config.yml
     8  - import_tasks: svc.yml
```

4.10 We have got the required files for installing roles. Let's apply this role into the ansible playbook "**run-installing.yml**" as below to deploy it on the client nodes.

```
[admin@eoc-controller ~]$ cat -n  run-installing.yml
     1  ---
     2  - hosts: "controller, workers"
     3    roles:
     4      - installing
```

4.11 Let's verify the syntax by executing command.

```
# ansible-playbook -i kube-infra --syntax-check run-installing.yml
```

Output:

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

playbook: run-installing.yml
```

4.12 Let's deploy the role by executing belwo command.

```
# ansible-playbook -i kube-infra run-installing.yml
```

Output:

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

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

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

TASK [installing : Remove buildah package] ************************************************
changed: [eoc-node1]
changed: [eoc-controller]
changed: [eoc-node2]

TASK [installing : Install yum-utils package] *********************************************
changed: [eoc-controller]
changed: [eoc-node1]
changed: [eoc-node2]
```

```
TASK [installing : Set SystemdCgroup to true in /etc/containerd/config.toml] *****************
changed: [eoc-node1]
changed: [eoc-node2]
changed: [eoc-controller]

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

PLAY RECAP ********************************************************************************
eoc-controller             : ok=8    changed=7    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
eoc-node1                  : ok=8    changed=7    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
eoc-node2                  : ok=8    changed=7    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
```

5. Creating role to install and configure kubernetes.

5.1 Creating another directory sturcture for this role.

```
# ansible-galaxy init roles/k8s --offline
```

Output:

```
[admin@eoc-controller ~]$ ansible-galaxy init roles/k8s --offline
- Role roles/k8s was created successfully
```

5.2 Add the latest kuberenetes repo.

Path → roles/k8s/tasks/add_repo.yml

```
[admin@eoc-controller ~]$ cat -n roles/k8s/tasks/add_repo.yml
     1  ---
     2  - name: Create Kubernetes repo file
     3    ansible.builtin.copy:
     4      content: |
     5        [kubernetes]
     6        name=Kubernetes
     7        baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
     8        enabled=1
     9        gpgcheck=1
    10        gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
    11      dest: /etc/yum.repos.d/kubernetes.repo
    12      owner: root
    13      group: root
    14      mode: '0644'
```

5.3 Install kubeadm, kubelet, and kubectl.

Path → roles/k8s/tasks/install.yml

```
[admin@eoc-controller ~]$ cat -n roles/k8s/tasks/install.yml
     1  - name: Install Kubernetes packages
     2    ansible.builtin.dnf:
     3      name:
     4        - kubelet-1.28.0
     5        - kubeadm-1.28.0
     6        - kubectl-1.28.0
     7      state: present
```

5.4 Let's start the kubelet service.

Path → roles/k8s/tasks/svc.yml

```
[admin@eoc-controller ~]$ cat -n roles/k8s/tasks/svc.yml
     1  ---
     2  - name: Enable and start kubelet service
     3    ansible.builtin.systemd:
     4      name: kubelet
     5      enabled: yes
     6      state: started
```

5.5 Let's import the roles in the main file.

Path → roles/k8s/tasks/main.yml

```
[admin@eoc-controller ~]$ cat -n roles/k8s/tasks/main.yml
     1  ---
     2  - import_tasks: add_repo.yml
     3  - import_tasks: install.yml
     4  - import_tasks: svc.yml
```

5.6 We have got the required files for k8s roles. Let's apply this role into the ansible playbook "**run-k8s.yml**" as below to deploy it on the client nodes.

```
[admin@eoc-controller ~]$ cat -n run-k8s.yml
     1  ---
     2  - hosts: "controller, workers"
     3    roles:
     4      - k8s
```

5.7 Let's verify the syntax so that we don't encounter any problem.

```
# ansible-playbook -i kube-infra --syntax-check run-k8s.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook -i kube-infra --syntax-check run-k8s.yml

playbook: run-k8s.yml
```

5.8 Let's deploy our roles by executing below command.

```
# ansible-playbook -i kube-infra run-k8s.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook -i kube-infra run-k8s.yml

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

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

TASK [k8s : Create Kubernetes repo file] ***************************************************
ok: [eoc-node2]
ok: [eoc-node1]
ok: [eoc-controller]

TASK [k8s : Install Kubernetes packages] ***************************************************
ok: [eoc-node2]
ok: [eoc-node1]
ok: [eoc-controller]

TASK [k8s : Enable and start kubelet service] **********************************************
ok: [eoc-node2]
ok: [eoc-node1]
ok: [eoc-controller]

PLAY RECAP *********************************************************************************
eoc-controller             : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0
eoc-node1                  : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0
eoc-node2                  : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0
```

6. Initialising the cluster.

6.1 Creating another directory sturcture for this role.

```
# ansible-galaxy init roles/init --offline
```

Output:

```
[admin@eoc-controller ~]$ ansible-galaxy init roles/init --offline
- Role roles/init was created successfully
```

6.2 Let's create a role to initialise the cluster.

Path → roles/init/tasks/init.yml

```
[admin@eoc-controller ~]$ cat -n roles/init/tasks/init.yml
     1  - name: Run kubeadm init and save output
     2    ansible.builtin.command:
     3      cmd: kubeadm init
     4    register: kubeadm_init_output
```

6.3 Let's save the token in the bootstrap.txt file.

Path → roles/init/tasks/copy.yml

```
[admin@eoc-controller ~]$ cat -n roles/init/tasks/copy.yml
     1  - name: Save kubeadm init output to a file
     2    ansible.builtin.copy:
     3      content: "{{ kubeadm_init_output.stdout }}"
     4      dest: bootstrap.txt
```

6.4 Let's create the config file directory.

Path → roles/init/tasks/config_files.yml

```
[admin@eoc-controller ~]$ cat -n roles/init/tasks/config_files.yml
     1  ---
     2  - name: "Configuration Files Setup"
     3    file:
     4      path: "$HOME/.kube"
     5      state: directory
```

6.5 Let's copy the configuration files.

Path → roles/init/tasks/copy_cfg.yml

```
[admin@eoc-controller ~]$ cat -n roles/init/tasks/copy_cfg.yml
     1  ---
     2  - name: "Copying Configuration File"
     3    copy:
     4      src: /etc/kubernetes/admin.conf
     5      dest: $HOME/.kube/config
     6      remote_src: yes
```

6.6 Lets give the permissions to the file.

Path → roles/init/tasks/permission.yml

```
[admin@eoc-controller ~]$ cat -n roles/init/tasks/permission.yml
     1  ---
     2  - name: Change kubeconfig file permission
     3    file:
     4      path: $HOME/.kube/config
     5      owner: "{{ ansible_effective_user_id }}"
     6      group: "{{ ansible_effective_group_id }}"
```

6.7 Let's add the CNI plugin.

Path → roles/init/tasks/cni.yml

```
[admin@eoc-controller ~]$ cat -n  roles/init/tasks/cni.yml
     1  ---
     2  - name: Apply Calico manifest using kubectl
     3    become: yes
     4    shell: kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

6.8 Let's Import the tasks into the **main.yml** file.

```
[admin@eoc-controller ~]$ cat -n roles/init/tasks/main.yml
     1  ---
     2  - import_tasks: init.yml
     3  - import_tasks: copy.yml
     4  - import_tasks: config_files.yml
     5  - import_tasks: copy_cfg.yml
     6  - import_tasks: permission.yml
     7  - import_tasks: cni.yml
```

6.9 We have got the required files for init roles. Let's apply this role into the ansible playbook "**run-init.yml**" as below to deploy it on the client nodes.

```
[admin@eoc-controller ~]$ cat -n run-init.yml
     1  ---
     2  - hosts: "controller"
     3    roles:
     4      - init
```

6.10 Let's verify the syntax of the roles in the roles.

```
# ansible-playbook -i kube-infra --syntax-check run-init.yml
```

Output:

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

playbook: run-init.yml
```

6.11 Let's deploy the roles by executing below command.

```
# ansible-playbook -i kube-infra run-init.yml
```

Output:

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

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

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

TASK [init : Run kubeadm init and save output] **********************************************
changed: [eoc-controller]

TASK [init : Save kubeadm init output to a file] ********************************************
changed: [eoc-controller]

TASK [init : Configuration Files Setup] *****************************************************
ok: [eoc-controller]

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

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

TASK [init : Apply Calico manifest using kubectl] *******************************************
changed: [eoc-controller]

PLAY RECAP **********************************************************************************
eoc-controller             : ok=7    changed=4    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0
```

7. Verifying cluster status using Ad-Hoc command

7.1 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   2m8s   v1.28.0
```

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

```
# cat bootstrap.txt
```

Output:

```
Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 1muzbc.v1ui93xj3pt8n1dg \
        --discovery-token-ca-cert-hash sha256:72b599fb4d3e9c55af8151ddb1716400898cda7aec73aea97b1
a8ba611027e2b [admin@eoc-controller ~]$
```

7.3 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 1muzbc.v1ui93xj3pt8n1dg --
discovery-token-ca-cert-hash
sha256:72b599fb4d3e9c55af8151ddb1716400898cda7aec73aea97b1a8
ba611027e2b'
```

Output:

```
[admin@eoc-controller ~]$ ansible workers -i kube-infra -m command -a 'kubeadm join 192.168.100.1
50:6443 --token 1muzbc.v1ui93xj3pt8n1dg --discovery-token-ca-cert-hash sha256:72b599fb4d3e9c55af8
151ddb1716400898cda7aec73aea97b1a8ba611027e2b'
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-con
fig -o yaml'
[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-fla
gs.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-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-con
fig -o yaml'
[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-fla
gs.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.
```

7.4 Let's verify the Cluster status.

```
# 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   4m47s   v1.28.0
eoc-node1        NotReady   <none>          42s     v1.28.0
eoc-node2        NotReady   <none>          42s     v1.28.0
```

7.5 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 node-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 node-role.kubernetes.io/node='
eoc-controller | CHANGED | rc=0 >>
node/eoc-node2 labeled
```

7.6 Let's verify the Cluster status now.

```
# 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   6m50s   v1.28.0
eoc-node1        Ready    node            2m45s   v1.28.0
eoc-node2        Ready    node            2m45s   v1.28.0
```