**Study Material: Creating and Managing a Local Kubernetes Cluster with Minikube**

**Introduction**

In this section, we’ll walk through the steps to create a local Kubernetes cluster using **Minikube**, connect to it, and explore its components. This is a hands-on guide to help you understand how Kubernetes works in a local environment.

**Step 1: Start the Minikube Cluster**

Minikube allows you to create a single-node Kubernetes cluster locally. Let’s start the cluster and verify its status.

**Start Minikube:**

1. Open your terminal.
2. Start Minikube with your preferred VM driver (e.g., VirtualBox, Hyper-V, KVM, etc.):
3. minikube start --driver=<driver-name>

Replace <driver-name> with your VM manager:

* + **VirtualBox**: virtualbox
  + **Hyper-V**: hyperv
  + **KVM**: kvm2
  + **Docker**: docker

Example for VirtualBox:

minikube start --driver=virtualbox

1. Wait for Minikube to set up the cluster. You’ll see messages like:
   * Creating VirtualBox VM.
   * Preparing Kubernetes.
   * Booting up the control plane.
2. Once done, you’ll see a message: **"Done! kubectl is now configured to use the Minikube cluster."**

**Verify Minikube Status:**

1. Check the status of the Minikube cluster:
2. minikube status

You should see:

* + **Host**: Running
  + **Kubernetes**: Running
  + **API Server**: Running
  + **kubeconfig**: Configured

1. Check the IP address of the Minikube cluster:
2. minikube ip

This will display the IP address of the Minikube node (e.g., 192.168.99.100).

**Step 2: Connect to the Minikube Node**

The Minikube node is a virtual machine running Kubernetes. You can connect to it using SSH to explore its internals.

**SSH into the Minikube Node:**

1. Get the IP address of the Minikube node:
2. minikube ip

Example output: 192.168.99.100.

1. Connect to the node using SSH:
2. ssh docker@<minikube-ip>

Replace <minikube-ip> with the IP address from the previous step.

Example:

ssh docker@192.168.99.100

1. When prompted for a password, enter: **tcuser**.
2. You’re now inside the Minikube node. Run the following command to list all running Docker containers:
3. docker ps

You’ll see containers like:

* + kube-apiserver
  + kube-scheduler
  + kube-proxy
  + coredns

These are the system components of Kubernetes running as containers.

1. Exit the SSH session:
2. exit

**Step 3: Explore the Kubernetes Cluster**

Now that the cluster is running, let’s explore it using **kubectl**, the Kubernetes command-line tool.

**Check Cluster Information:**

1. Get cluster information:
2. kubectl cluster-info

You’ll see details like:

* + Kubernetes control plane is running at https://<ip>:<port>.
  + CoreDNS is running.

**List Nodes in the Cluster:**

1. List all nodes in the cluster:
2. kubectl get nodes

You’ll see a single node (since Minikube creates a single-node cluster):

* + **NAME**: minikube
  + **STATUS**: Ready
  + **ROLES**: control-plane, master

**List Pods in the Cluster:**

1. List all pods in the **default** namespace:
2. kubectl get pods

Initially, you’ll see: **No resources found in the default namespace.**

1. List all namespaces:
2. kubectl get namespaces

You’ll see namespaces like:

* + default
  + kube-system
  + kube-public
  + kube-node-lease

1. List pods in the **kube-system** namespace (where system components run):
2. kubectl get pods -n kube-system

You’ll see system pods like:

* + coredns
  + kube-apiserver
  + kube-proxy
  + kube-scheduler

**Step 4: Recap and Key Takeaways**

* **Minikube** creates a single-node Kubernetes cluster locally.
* The Minikube node runs system components (e.g., kube-apiserver, kube-scheduler) as Docker containers.
* You can connect to the Minikube node using SSH and explore its internals.
* Use **kubectl** to interact with the Kubernetes cluster:
  + kubectl get nodes: List nodes.
  + kubectl get pods: List pods.
  + kubectl get namespaces: List namespaces.

**Next Steps**

Now that your local Kubernetes cluster is up and running, you can:

* Create deployments.
* Expose services.
* Scale applications.
* Explore advanced Kubernetes features.

**Troubleshooting**

* **Minikube fails to start**: Ensure your VM manager (e.g., VirtualBox, Hyper-V) is installed and running.
* **kubectl not working**: Verify that kubectl is installed correctly and can connect to the cluster using kubectl get nodes.

If you encounter issues, refer to the [Minikube documentation](https://minikube.sigs.k8s.io/docs/) or the [Kubernetes documentation](https://kubernetes.io/docs/).

### Study Material: Creating and Managing a Local Kubernetes Cluster with Minikube

---

#### \*\*Introduction\*\*

In this section, we’ll walk through the steps to create a local Kubernetes cluster using \*\*Minikube\*\*, connect to it, and explore its components. This is a hands-on guide to help you understand how Kubernetes works in a local environment.

---

### \*\*Step 1: Start the Minikube Cluster\*\*

Minikube allows you to create a single-node Kubernetes cluster locally. Let’s start the cluster and verify its status.

---

#### \*\*Start Minikube:\*\*

1. Open your terminal.

2. Start Minikube with your preferred VM driver (e.g., VirtualBox, Hyper-V, KVM, etc.):

```bash

minikube start --driver=<driver-name>

```

Replace `<driver-name>` with your VM manager:

- \*\*VirtualBox\*\*: `virtualbox`

- \*\*Hyper-V\*\*: `hyperv`

- \*\*KVM\*\*: `kvm2`

- \*\*Docker\*\*: `docker`

Example for VirtualBox:

```bash

minikube start --driver=virtualbox

```

3. Wait for Minikube to set up the cluster. You’ll see messages like:

- Creating VirtualBox VM.

- Preparing Kubernetes.

- Booting up the control plane.

4. Once done, you’ll see a message: \*\*"Done! kubectl is now configured to use the Minikube cluster."\*\*

---

#### \*\*Verify Minikube Status:\*\*

1. Check the status of the Minikube cluster:

```bash

minikube status

```

You should see:

- \*\*Host\*\*: Running

- \*\*Kubernetes\*\*: Running

- \*\*API Server\*\*: Running

- \*\*kubeconfig\*\*: Configured

2. Check the IP address of the Minikube cluster:

```bash

minikube ip

```

This will display the IP address of the Minikube node (e.g., `192.168.99.100`).

---

### \*\*Step 2: Connect to the Minikube Node\*\*

The Minikube node is a virtual machine running Kubernetes. You can connect to it using SSH to explore its internals.

---

#### \*\*SSH into the Minikube Node:\*\*

1. Get the IP address of the Minikube node:

```bash

minikube ip

```

Example output: `192.168.99.100`.

2. Connect to the node using SSH:

```bash

ssh docker@<minikube-ip>

```

Replace `<minikube-ip>` with the IP address from the previous step.

Example:

```bash

ssh docker@192.168.99.100

```

3. When prompted for a password, enter: \*\*tcuser\*\*.

4. You’re now inside the Minikube node. Run the following command to list all running Docker containers:

```bash

docker ps

```

You’ll see containers like:

- `kube-apiserver`

- `kube-scheduler`

- `kube-proxy`

- `coredns`

These are the system components of Kubernetes running as containers.

5. Exit the SSH session:

```bash

exit

```

---

### \*\*Step 3: Explore the Kubernetes Cluster\*\*

Now that the cluster is running, let’s explore it using \*\*kubectl\*\*, the Kubernetes command-line tool.

---

#### \*\*Check Cluster Information:\*\*

1. Get cluster information:

```bash

kubectl cluster-info

```

You’ll see details like:

- Kubernetes control plane is running at `https://<ip>:<port>`.

- CoreDNS is running.

---

#### \*\*List Nodes in the Cluster:\*\*

1. List all nodes in the cluster:

```bash

kubectl get nodes

```

You’ll see a single node (since Minikube creates a single-node cluster):

- \*\*NAME\*\*: `minikube`

- \*\*STATUS\*\*: `Ready`

- \*\*ROLES\*\*: `control-plane`, `master`

---

#### \*\*List Pods in the Cluster:\*\*

1. List all pods in the \*\*default\*\* namespace:

```bash

kubectl get pods

```

Initially, you’ll see: \*\*No resources found in the default namespace.\*\*

2. List all namespaces:

```bash

kubectl get namespaces

```

You’ll see namespaces like:

- `default`

- `kube-system`

- `kube-public`

- `kube-node-lease`

3. List pods in the \*\*kube-system\*\* namespace (where system components run):

```bash

kubectl get pods -n kube-system

```

You’ll see system pods like:

- `coredns`

- `kube-apiserver`

- `kube-proxy`

- `kube-scheduler`

---

### \*\*Step 4: Recap and Key Takeaways\*\*

- \*\*Minikube\*\* creates a single-node Kubernetes cluster locally.

- The Minikube node runs system components (e.g., `kube-apiserver`, `kube-scheduler`) as Docker containers.

- You can connect to the Minikube node using SSH and explore its internals.

- Use \*\*kubectl\*\* to interact with the Kubernetes cluster:

- `kubectl get nodes`: List nodes.

- `kubectl get pods`: List pods.

- `kubectl get namespaces`: List namespaces.

---

### \*\*Next Steps\*\*

Now that your local Kubernetes cluster is up and running, you can:

- Create deployments.

- Expose services.

- Scale applications.

- Explore advanced Kubernetes features.

---

### \*\*Troubleshooting\*\*

- \*\*Minikube fails to start\*\*: Ensure your VM manager (e.g., VirtualBox, Hyper-V) is installed and running.

- \*\*kubectl not working\*\*: Verify that kubectl is installed correctly and can connect to the cluster using `kubectl get nodes`.

If you encounter issues, refer to the [Minikube documentation](https://minikube.sigs.k8s.io/docs/) or the [Kubernetes documentation](https://kubernetes.io/docs/).