Demo

The script you've provided outlines the process of installing Kubernetes on CentOS or RHEL nodes, both for manager (master) and worker nodes. Below is a breakdown and overview of each step in the script, which covers tasks like kernel module loading, container runtime installation, and Kubernetes tool installation.

### 1. \*\*Basic Overview of Kubernetes Installation\*\*

Kubernetes is a container orchestration platform that automates the deployment, scaling, and management of containerized applications. The script installs the required components for Kubernetes, including:

- \*\*Kubelet\*\*: An agent that runs on each node in the cluster.

- \*\*Kubeadm\*\*: A tool for initializing and managing Kubernetes clusters.

- \*\*Kubectl\*\*: A command-line tool to interact with Kubernetes clusters.

- \*\*Containerd\*\*: The container runtime that is used to manage containers (alternative to Docker).

The script takes the following steps to install these components and prepare the system for Kubernetes cluster setup.

---

### 2. \*\*Step-by-Step Breakdown\*\*

#### 2.1 \*\*Enable Kernel Modules (`overlay`, `br\_netfilter`)\*\*

- \*\*Why?\*\*

Kubernetes requires certain kernel modules to ensure proper networking and container isolation. The `overlay` module allows for efficient networking between containers, and `br\_netfilter` is used for bridge networking in Kubernetes.

- \*\*What happens?\*\*

- The script adds `overlay` and `br\_netfilter` to the `/etc/modules-load.d/containerd.conf` to ensure these modules are loaded on boot.

- Then, it manually loads them using `modprobe`.

```bash

sudo tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

```

#### 2.2 \*\*Configure Sysctl Settings\*\*

- \*\*Why?\*\*

These settings configure the system's networking to enable Kubernetes features such as IP forwarding and iptables filtering across bridges (for pod communication).

- \*\*What happens?\*\*

The script adds kernel parameters related to networking into `/etc/sysctl.d/k8s.conf` to ensure Kubernetes works correctly. It then reloads the sysctl settings.

```bash

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF

sudo sysctl --system

```

#### 2.3 \*\*Disable Swap (Important for Kubernetes)\*\*

- \*\*Why?\*\*

Kubernetes requires swap to be disabled. If swap is enabled, kubelet will fail to run, as it can cause issues with memory management in a cluster.

- \*\*What happens?\*\*

- The script disables swap with `swapoff -a`.

- It also removes the swap entry from `/etc/fstab` to prevent it from being enabled on reboot.

```bash

sudo swapoff -a

sudo sed -i '/swap/d' /etc/fstab

```

#### 2.4 \*\*Install Containerd Runtime\*\*

- \*\*Why?\*\*

Kubernetes requires a container runtime to run containers, and `containerd` is a lightweight, industry-standard container runtime. This step installs it using `dnf` (for CentOS/RHEL).

- \*\*What happens?\*\*

- Adds Docker's repository to ensure availability of containerd packages.

- Installs the `containerd.io` package.

- Modifies the `containerd` configuration to enable systemd-based cgroups, which is recommended for Kubernetes.

- Finally, it starts, enables, and checks the status of the `containerd` service.

```bash

sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

sudo yum install containerd.io -y

sudo containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl enable containerd

sudo systemctl status containerd &> /dev/null

```

#### 2.5 \*\*Add Kubernetes Repository\*\*

- \*\*Why?\*\*

The script adds the Kubernetes repository to the system so that the `kubelet`, `kubeadm`, and `kubectl` packages can be installed from a stable source.

- \*\*What happens?\*\*

- It creates a repository configuration file for Kubernetes in `/etc/yum.repos.d/kubernetes.repo`.

- It ensures the repository points to the stable Kubernetes v1.29 release, with GPG checks enabled.

```bash

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.29/rpm/

enabled=1

gpgcheck=1

gpgkey=https://pkgs.k8s.io/core:/stable:/v1.29/rpm/repodata/repomd.xml.key

exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

EOF

```

#### 2.6 \*\*Install Kubernetes Components (`kubelet`, `kubeadm`, `kubectl`)\*\*

- \*\*Why?\*\*

These are the main tools for running and managing Kubernetes clusters.

- `kubelet`: Runs on every node in the cluster and ensures containers are running.

- `kubeadm`: A tool to set up and manage the cluster.

- `kubectl`: A command-line tool to interact with the cluster.

- \*\*What happens?\*\*

The script installs the Kubernetes components (`kubelet`, `kubeadm`, `kubectl`) using `yum` and disables excluding them from the repositories.

```bash

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

```

#### 2.7 \*\*Start and Enable Kubelet Service\*\*

- \*\*Why?\*\*

The `kubelet` service needs to be enabled and running on each node in the cluster to ensure it is managed by the system.

- \*\*What happens?\*\*

- The `systemctl` command ensures the `kubelet` service starts automatically on boot and is running immediately.

```bash

sudo systemctl enable --now kubelet

sudo systemctl enable kubelet

```

#### 2.8 \*\*Completion Message\*\*

- \*\*Why?\*\*

After the installation is complete, it prints a success message to inform the user that the setup has been finished.

```bash

echo "#######Installation completed successfully #####"

```

---

### 3. \*\*Mind Map of the Installation Process\*\*

Here’s a simplified mind map of the installation steps:

```

+----------------------+

| Install Kubernetes |

+----------------------+

|

+---------------------+---------------------+

| |

Enable Kernel Modules Install Containerd

(overlay, br\_netfilter) (containerd runtime)

| |

+------------------------+ +----------------------+

| | | |

Configure Sysctl Disable Swap Add Kubernetes Repo Install K8s (kubelet, kubeadm, kubectl)

| | | |

Reload Sysctl Remove swap from fstab Install from repo Enable and Start kubelet

| |

+--------------------+ Display Completion

| |

Restart containerd Enable containerd

```

---

### Conclusion

This script is designed to set up Kubernetes on CentOS/RHEL systems by:

1. Configuring the kernel and system networking for containerization.

2. Disabling swap (a Kubernetes requirement).

3. Installing a container runtime (containerd).

4. Setting up Kubernetes components (`kubelet`, `kubeadm`, `kubectl`).

5. Enabling the necessary services (like `kubelet` and `containerd`).

By running this script on both manager (master) and worker nodes, you can prepare the system for the Kubernetes cluster setup. The next steps would involve using `kubeadm` to initialize the cluster and join nodes.

Cordon vs Drain in Kubernetes

In Kubernetes, cordon and drain are two related but distinct commands used for node management. Understanding the differences between them is crucial for effective cluster maintenance and optimization.

Cordon: kubectl cordon marks a node as “unschedulable,” preventing new pods from being scheduled on that node. Existing pods on the node continue to run, but no new pods will be deployed or rescheduled on that node. This is useful for:

Temporarily preventing new workloads from running on a node for maintenance or troubleshooting.

Isolating a node for debugging or testing purposes.

Reducing resource contention by limiting the number of pods on a node.

Drain: kubectl drain evicts all running pods from a node, including those managed by ReplicationControllers, Jobs, and DaemonSets. This process ensures that:

All pods are terminated and removed from the node, freeing up resources.

The node is prepared for maintenance, upgrades, or removal from the cluster.

Key differences:

Purpose: Cordon prevents new pods from being scheduled, while drain removes existing pods from a node.

Scope: Cordon affects only new pod scheduling, while drain affects all running pods on the node.

Node availability: After cording, the node remains available for existing pods, but after draining, the node is empty and ready for maintenance or removal.

When to use each:

Use cordon when you need to temporarily prevent new workloads from running on a node for maintenance or troubleshooting.

Use drain when you need to prepare a node for maintenance, upgrades, or removal from the cluster, and all pods on the node need to be terminated and removed.

Remember to use --ignore-daemonsets with kubectl drain if you have DaemonSets running on the node, as they will automatically recreate pods on the node if it’s drained.

Troubleshooting if ip address of cni0 is conflicted

Container-creating state

This error message suggests that there is a network configuration issue with Flannel, which is preventing Kubernetes from setting up the pod sandbox. Specifically, the `cni0` bridge interface is being assigned an IP address that conflicts with the expected IP range for Flannel (`10.244.3.1/24`), causing the network setup to fail.

Here’s a step-by-step approach to troubleshoot this issue:

### 1. \*\*Verify Flannel Configuration\*\*

- Ensure that Flannel's IP address configuration is correct.

- Check the Flannel configuration file (typically located in `/etc/cni/net.d/` on the nodes) for the correct subnet and network settings.

- You can also check the Flannel configuration in Kubernetes as a `ConfigMap` or `DaemonSet`:

```bash

kubectl get cm -n kube-system

kubectl describe cm kube-flannel-cfg -n kube-system

```

### 2. \*\*Check Existing `cni0` Interface\*\*

- Check the IP address of the `cni0` interface to see if it matches what is expected by Flannel (in this case, `10.244.3.1/24`):

```bash

ip addr show cni0

```

- If `cni0` already has an IP address that conflicts with the Flannel IP range, that could explain the error.

### 3. \*\*Clear Conflicting IP Address\*\*

- If you find that `cni0` has an IP address that conflicts with the Flannel subnet, you may need to manually remove or change the conflicting IP address:

```bash

ip addr del <conflicting-ip> dev cni0

```

Replace `<conflicting-ip>` with the actual IP address that is causing the conflict.

### 4. \*\*Restart Flannel DaemonSet\*\*

- Restart the Flannel DaemonSet to reapply the configuration and reset the network state:

```bash

kubectl rollout restart daemonset kube-flannel-ds -n kube-system

```

### 5. \*\*Check for Overlapping Subnets\*\*

- Ensure that no other network plugin or manual configurations (e.g., on the host or in Docker) are causing overlapping subnets. You can inspect the Docker network settings:

```bash

docker network inspect bridge

```

### 6. \*\*Check for Other Network Plugins\*\*

- If you're running multiple CNI plugins (e.g., Flannel, Calico, Weave), there could be a conflict between them. You can inspect the CNI configuration directory to ensure only Flannel is being used:

```bash

ls /etc/cni/net.d/

```

### 7. \*\*Inspect Kubelet Logs\*\*

- Look at the kubelet logs for more detailed information about the error:

```bash

journalctl -u kubelet -f

```

- This may provide additional context about what’s failing.

### 8. \*\*Verify Node Network Settings\*\*

- Ensure that there is no IP address conflict on the host machine itself, as the `cni0` bridge is typically created as a virtual interface. You can check for IP conflicts using:

```bash

ip addr show

```

### 9. \*\*Verify Pod Network Configuration\*\*

- Finally, verify the pod network configuration and ensure that it aligns with Flannel's expected behavior:

```bash

kubectl get pods -n kube-system -o wide

kubectl describe pod <flannel-pod-name> -n kube-system

```

### 10. \*\*Reboot Node (if necessary)\*\*

- If you still can't resolve the issue, a node reboot may help clear up any stale network configurations or IP conflicts.

### 11. \*\*Check for Flannel Logs\*\*

- Look for Flannel-specific logs to see if there are issues with Flannel itself:

```bash

kubectl logs -n kube-system -l app=flannel

```

### 12. \*\*Network Plugin Reinstallation (last resort)\*\*

- As a last resort, if you suspect that Flannel or the CNI plugin is misconfigured or corrupted, you may need to reinstall or reapply the network plugin.

---

If the above steps don’t resolve the issue, let me know if you need more help debugging or any additional context on the environment.

kubectl get pods -o wide

299 cat /etc/hosts

300 kubectl get pods -o wide

301 kubectl delete deployment app1

302 kubectl create deployment app1 --image=dhpcsa/kube-app1:v1 --port=80 --replicas=3

303 kubectl get pods -o wide

304 kubectl get nodes

305 kubectl describe app1-6c4644978f-58q7g

306 kubectl describe pod app1-6c4644978f-58q7g

307 clear

308 kubectl get nodes

309 kubectl delete deployment app1

310 kubectl create deployment app1 --image=dhpcsa/kube-app1:v1 -port=80 --replicas=3

311 kubectl create deployment app1 --image=dhpcsa/kube-app1:v1 --port=80 --replicas=3

312 kubectl get pods -o wide

313 kubectl get svc

314 kubectl delete svc my-k8s-app-service

315 kubectl get svc

316 clear

317 kubectl expose deployment app1 --type=NodePort

318 kubectl get svc

319 curl localhost:31035

320 clear

321 kubectl get pods -o wide

322 kubectl get svc

323 curl node1:31035

324 curl node2:31035

325 curl node3:31035

326 curl node1:31035

327 curl node2:31035

328 curl node3:31035

329 curl node1:31035

330 curl node2:31035

331 curl node3:31035

332 kubectl expose

333 clear

334 kubectl get svc

335 curl node1:31035

336 curl node2:31035

337 curl node3:31035

338 curl node1:31035

339 curl node2:31035

340 curl node3:31035

341 kubectl get pods -o wide

342 curl node3:31035

343 curl node2:31035

344 curl node1:31035

345 kubectl exec -ti app1-6c4644978f-5v8rk --ping 10.244.3.169

346 kubectl exec -ti app1-6c4644978f-5v8rk -- ping 10.244.3.169

347 apt-get iputils-ping

348 apt-get install iputils-ping

349 ping

350 clear

351 kubectl get pods -o wide

352 kubectl exec -ti app1-6c4644978f-5v8rk

353 kubectl get svc

354 curl node1:31035

355 curl node2:31035

356 curl node3:31035

357 curl node1:31035

358 curl node2:31035

359 curl node3:31035

360 curl node1:31035

361 curl node2:31035

362 curl node3:31035

363 kubectl get nodes

364 ssh node3

365 ping node3

366 curl node3:31035

367 ls

368 curl node1:31035

369 curl node2:31035

370 curl node3:31035

371 kubectl describe deployment app1

372 curl node3:31035

373 curl node2:31035

374 curl node1:31035

375 cler

376 clear

377 kubectl get pods

378 kubectl get svc

379 curl node1:31035

380 curl node2:31035

381 curl node3:31035

382 curl node1:31035

383 kubectl get nodes