Documentation Kubernetes with Docker + Cilium + OpenFaaS + Prometheus + Grafana + ChaosMesh

1. Prerequisites

1.1. Activate IPv4 Packet-Forwarding (control-plane only)

```
# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

1.2. Install Helm (control-plane only)

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee
/usr/share/keyrings/helm.gpg > /dev/null

sudo apt-get install apt-transport-https --yes

echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all
main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

sudo apt-get update
sudo apt-get install helm
```

1.3. Install Arkade (control-plane only)

```
curl -sSL https://get.arkade.dev | sudo -E sh
```

1.4. Install DockerEngine (all nodes)

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
```

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update


sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin


sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

1.5. Install cri-dockerd (all nodes)

```
sudo apt-get install -y golang-go

wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.3.17/cri-
dockerd_0.3.17.3-0.ubuntu-jammy_amd64.deb

sudo apt-get install ./cri-dockerd_0.3.17.3-0.ubuntu-jammy_amd64.deb
```

2. Kubernetes

2.1. Installation (all nodes)

```
sudo apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that package
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

[optional] If the directory `/etc/apt/keyrings` does not exist, it should be created before the curl command:

```
sudo mkdir -p -m 755 /etc/apt/keyrings
```

[Continue here]:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key | sudo gpg --
dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

# This overwrites any existing configuration in
/etc/apt/sources.list.d/kubernetes.list
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.32/deb/ /' | sudo tee
```

```
/etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

2.2. Cluster initialization

2.2.1 Create kubeadm-config (control-plane only)

Create a file called `kubeadm-config.yaml` and add the following configuration:

```
# kubeadm-config.yaml
apiVersion: kubeadm.k8s.io/v1beta4
kind: InitConfiguration
nodeRegistration:
    criSocket: unix:///run/cri-dockerd.sock
    kubelet:
        extraArgs:
          - name: "cgroup-driver"
            value: "systemd"

---
apiVersion: kubeadm.k8s.io/v1beta4
kind: ClusterConfiguration
kubernetesVersion: v1.32.4
networking:
    podSubnet: "10.0.0.0/16"
```

2.2.2 Create the cluster (control-plane only)

```
sudo kubeadm init --config kubeadm-config.yaml
```

→ Save join-command (`sudo` and `--cri-socket unix:///run/cri-dockerd.sock` need to be added, as DockerEngine is used as CRI and to work properly):

```
sudo kubeadm join 172.16.44.202:6443 --token dsdsr5.w17065zywxcbntb1 --discovery-
token-ca-cert-hash
sha256:07ec5a076b23f6373bc52ff3019db333946741c62e77268ed5b96f847c5ab04d --cri-
socket unix:///run/cri-dockerd.sock
```

→ Follow instructions (control-plane only):

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

2.2.3 Install CNI: Cilium (control-plane only)

- Cilium-CLI

```
CILIUM_CLI_VERSION=$(curl -s
https://raw.githubusercontent.com/cilium/cilium-cli/main/stable.txt)
CLI_ARCH=amd64

if [ "$(uname -m)" = "aarch64" ]; then CLI_ARCH=arm64; fi
curl -L --fail --remote-name-all https://github.com/cilium/cilium-
cli/releases/download/${CILIUM_CLI_VERSION}/cilium-
linux-${CLI_ARCH}.tar.gz{,.sha256sum}
sha256sum --check cilium-linux-${CLI_ARCH}.tar.gz.sha256sum

sudo tar xzvfC cilium-linux-${CLI_ARCH}.tar.gz /usr/local/bin
rm cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}
```

- Create a configuration file called `cilium-values.yaml` and add the following configuration:

```
# cilium-values.yaml
kubeProxyReplacement: true
bandwidthManager:
  enabled: true
enableK8sEndpointSlice: true
ipam:
  mode: kubernetes
k8sServiceHost: 172.16.44.202
k8sServicePort: 6443
```

- Apply configuration:

```
cilium install --version 1.17.2 --values cilium-values.yaml
```

2.2.4. [OPTIONAL] Deploy Kubernetes Dashboard (requires `helm`, control-plane only)

```
# Add kubernetes-dashboard repository
helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
# Deploy a Helm Release named "kubernetes-dashboard" using the kubernetes-
dashboard chart
helm upgrade --install kubernetes-dashboard kubernetes-dashboard/kubernetes-
dashboard --create-namespace --namespace kubernetes-dashboard
# Apply manifest
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommend
ed.yaml
```

- Create new configuration file `dashboard-adminuser.yaml` and add the following configuration:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
subjects:
  - kind: ServiceAccount
    name: admin-user
    namespace: kubernetes-dashboard
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

→ Apply configuration:

```
kubectl apply -f dashboard-adminuser.yaml
```

→ Generate token for accessing dashboard:

```
kubectl create serviceaccount admin-user -n kubernetes-dashboard
kubectl create clusterrolebinding admin-user-binding --clusterrole=cluster-admin -
-serviceaccount=kubernetes-dashboard:admin-user
kubectl -n kubernetes-dashboard create token admin-user
```

→ Save token:

```
HERE COMES A VERY LONG TOKEN!
```

- Make dashboard accessible (`https://172.16.44.202:8443`)

```
kubectl port-forward --address 172.16.44.202 -n kubernetes-dashboard
svc/kubernetes-dashboard 8443:443 &
```

2.2.5. Install OpenFaaS (requires `arkade`, control-plane only)

- Install FaaS-CLI:

```
arkade get faas-cli
```

- Install OpenFaaS:

```
arkade install openfaas
```

- Check rollout status:

```
kubectl rollout status -n openfaas deploy/gateway
```

- Make OpenFaaS dashboard accessible (`http://172.16.44.202:8081`):

```
kubectl port-forward --address 172.16.44.202 -n openfaas svc/gateway-
external 8081:8080 &
```

❗❗ **[optional]** To avoid an error about missing `socat`, it has to be installed on all nodes of the cluster:

```
sudo apt-get install socat
```

- **[optional]** Create credentials:

```
kubectl create secret generic basic-auth \
-n openfaas \
--from-literal=basic-auth-user=admin \
--from-literal=basic-auth-password="<NEWPASSWORD>"
```

- Retrieve credentials:

```
echo $(kubectl -n openfaas get secret basic-auth -o jsonpath="{.data.basic-
auth-password}" | base64 --decode)
```

⇒ **[CREDENTIALS]:**
username: admin
password: <generated_password_from_above>

- Restart gateway:

```
kubectl delete pod -n openfaas -l app=gateway
```

2.2.6. Connect workers (worker nodes only)

```
sudo kubeadm join 172.16.44.202:6443 --token dsdsr5.w17065zywxcbntb1 --discovery-token-ca-cert-hash sha256:07ec5a076b23f6373bc52ff3019db333946741c62e77268ed5b96f847c5ab04d --cri-socket unix:///run/cri-dockerd.sock
```

[optional] If the VM designated as `control-plane` also needs to be a worker node use (control-plane only):

```
kubectl taint nodes control-plane node-role.kubernetes.io/control-plane-
```

3. Monitoring + Evaluation

3.1. Prometheus (requires `helm`, control-plane only)

3.1.1. Create a namespace in the cluster

```
kubectl create namespace monitoring
```

3.1.2. Installation

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update

helm install prometheus prometheus-community/prometheus \
  --namespace monitoring \
  --set alertmanager.persistentVolume.enabled=false \
  --set server.persistentVolume.enabled=false
```

3.1.3. Access Prometheus (`http://172.16.44.202:9090`)

```
kubectl port-forward --address 172.16.44.202 -n monitoring svc/prometheus-server 9090:80 &
```

3.2. Grafana (requires `helm`, control-plane only)

3.2.1. Installation

```
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update

helm install grafana grafana/grafana -n monitoring --create-namespace
```

3.2.2. Retrieve Credentials

```
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-
password}" | base64 --decode ; echo
```

⇒ **[CREDENTIALS]** username: admin password: xFpNlRwJ7C0qLGVCWpr3xCWyeWESFRHBN5T73hbR

3.2.3. Access Grafana (`http://172.16.44.202:3000`)

```
kubectl port-forward --address 172.16.44.202 -n monitoring svc/grafana 3000:80 &
```

4. Network Resource Constraints

4.1. Chaos Mesh

4.1.1. Create a namespace in the cluster

```
kubectl create namespace chaos-mesh
```

4.1.2. Installation

```
helm repo add chaos-mesh https://charts.chaos-mesh.org
helm repo update


helm install chaos-mesh chaos-mesh/chaos-mesh \
  --namespace=chaos-mesh \
  --set chaosDaemon.runtime=docker \
  --set chaosDaemon.socketPath=/var/run/docker.sock
```

[optional] Verify Installation:

```
kubectl get pods -n chaos-mesh
```

⇒ Every pod's `STATUS` should be `Running`

4.1.3. Access ChaosMesh (`http://172.16.44.202:2333`)

```
kubectl port-forward --address 172.16.44.202 -n chaos-mesh svc/chaos-dashboard
2333:2333 &
```

4.1.4. Generate RBAC-Token

Use the `Click here to generate`-link on the `Enter the token (RBAC Authorization) to continue`-site.

- Activate `cluster scoped`
- Role: Manager
- Copy the configuration as indicated
- Create a new directory `chaos-mesh`
- Create a new file `rbac.yaml` in `chaos-mesh`
- Paste the copied configuration into `rbac.yaml`

```
kind: ServiceAccount
apiVersion: v1
metadata:
  namespace: default
  name: account-cluster-manager-iimqm

---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: role-cluster-manager-iimqm
rules:
- apiGroups: [""]
  resources: ["pods", "namespaces"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["chaos-mesh.org"]
  resources: [ "*" ]
  verbs: ["get", "list", "watch", "create", "delete", "patch", "update"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: bind-cluster-manager-iimqm
subjects:
- kind: ServiceAccount
  name: account-cluster-manager-iimqm
  namespace: default
roleRef:
  kind: ClusterRole
  name: role-cluster-manager-iimqm
  apiGroup: rbac.authorization.k8s.io
```

- Apply configuration:

```
kubectl apply -f chaos-mesh/rbac.yaml
```

- Create the actual token:

```
kubectl create token account-cluster-manager-iimqm
```

⇒ **[Token]**:

```
HERE COMES A VERY LONG TOKEN!
```

- Login to the dashboard with an arbitrary name (e.g. `admin`) and the token from above