Deployment of scalable simulation

Kubernetes and Docker allow you to containerize the Simulation web application, enabling you to run multiple simulations at one time. This document describes the steps for installing a Kubernetes cluster using the kubeadm command and deploying a simulation web application to it. The following commands must be executed on all nodes, unless specifically directed to act differently.

Prerequisites

Before initiating the process of installation, do the following:

- Check the OS, Hardware Configurations and Network connectivity. To check if your environment meets the listed requirements, see https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#before-you-begin.
- Turn off the swap and firewall by executing the following command:

```
swapoff -a
systemctl stop firewalld
systemctl disable firewalld
```

If you do not want to disable the firewall, find out which ports should be allowed.

The instructions below have been tested with CentOS 7. Some steps might differ if you are using a different Linux distribution.

To install the Kubernetes cluster and deploy the Simulation web application

---

1. If you already have the Kubernetes cluster setup and running (or you are on a cloud which provides it), you can skip to step 10. Steps 2-9 are required if you are working in an environment (such as a bare-metal server or raw VM) where Kubernetes is not yet installed.
2. Configure the local IP tables to see the bridged traffic:
   a. Enable the bridged traffic by executing the following command:

```
lsmod | grep br_netfilter
modprobe br_netfilter
```

   b. Copy the contents below into the */etc/modules-load.d/k8s.conf* file. If the file does not exist, create the file.

```
br_netfilter
```

   c. Copy the contents below into the */etc/sysctl.d/k8s.conf* file. If the file does not exist, create the file.

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

   d. Reload settings by executing the following command:

```
sysctl --system
```

3. Install Docker as a container runtime.
   a. Uninstall any older versions by executing the following command:

```
yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine
```

   b. Install yum utilities by executing the following command:

```
yum install -y yum-utils
```

   c. Set up the Docker repository by executing the following command:

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

d. Install the Docker Engine, Docker CLI, and Docker runtime by executing the following command:

```
yum install -y docker-ce docker-ce-cli containerd.io
```


If you are having problems installing some packages, try adding the following content at the beginning of the */etc/yum.repos.d/docker-ce.repo* file. If the file does not exist, create the file.

```
[centos-extras]
name=Centos extras - $basearch
baseurl=http://mirror.centos.org/centos/7/extras/x86_64
enabled=1
gpgcheck=1
gpgkey=http://centos.org/keys/RPM-GPG-KEY-CentOS-7
```

4. Configure the Docker Daemon for cgroups management and Start Docker:
   a. Create a directory by executing the following command:

```
mkdir /etc/docker
```

   b. Copy the contents below into the */etc/docker/daemon.json* file. If the file does not exist, create the file.

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

   c. Reload settings and start Docker by executing the following command:

```
systemctl daemon-reload
systemctl restart docker
systemctl enable docker
systemctl status docker
```

5. Install kubeadm, kubectl, and kubelet:
   a. Copy the contents below into the */etc/yum.repos.d/kubernetes.repo* file. If the file does not exist, create the file.

```
kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum
/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
```


If you are getting a “repomd.xml signature could not be verified” error, try disabling the repository GPG key checking in the yum repo configuration by setting repo_gpgcheck=0. For more Information, see https://cloud.google.com/compute/docs/troubleshooting/known-issues#keyexpired.

b. Set SELinux in permissive mode (effectively disabling it) by executing the following command:

```
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
systemctl enable --now kubelet
```

> ⚠ If you do not want to disable security (SELinux), find out what should be configured to work with Kubernetes.

6. Deploy the Kubernetes cluster using kubeadm:
   a. Initialize a cluster in the master node as a root user, by executing the following command . Replace <master_nodeIP> with the IP address of the master node:

```
kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=<master_nodeIP>
```

   b. After seeing a successful cluster initialization message, you should see a “kubeadm join” command with a token next to it. Copy the command and save it. It will be used to join other nodes to the cluster.
   c. Create a Kubernetes configuration directory by executing the following command:

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

7. Install CNI for POD Networking in the master node as a root user by executing the following command :

```
kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/v0.17.0/Documentation/kube-flannel.
yml
```

8. Join any number of worker nodes to the cluster as a root user by running the kubeadm command (copied in step 6a) on each node. The command below is just an example.

```
kubeadm join 192.168.74.10:6443 --token s2ld5q.pm8vflxp90mtm416 \
                                       --discovery-token-ca-cert-hash sha256:
967ebd0b2c6c4c18c2833ec1a61b224696d87f5ee78bc31014830d0dcb3e0a88
```

> ⚠ You cannot deploy pods on the master node by default. If you want to do so, you have to execute the following command:
>
> ```
> kubectl taint node master node-role.kubernetes.io/master:NoSchedule- master node-role.kubernetes.
> io/master:NoSchedule-
> ```

9. Confirm the cluster status by executing the following command:

```
kubectl get nodes
```

If everything is correct, you should see a “Ready” status on all cluster nodes.

10. Build a Docker image:
    a. Create a Docker image. For more information, see Creating a Docker image.
    b. Push the image to the image registry, by executing the following command:

```
docker push <image_name:version>
```

> ⚠ At this point, the Docker image registry should be up and running. For more information, see https://docs.docker.com/registry/.

11. Apply Kubernetes configurations:

a. Prepare configuration files according to your environment.

> ⚠
> - You may need to change the “spec.template.spec.containers.image” property in the *simulation.deployment.yml* file to point to the Docker image you created in step 10.
> - You may need to change the “spec. jobTargetRef.template.spec.containers.image” property in the *simulation.job.yml* file to point to the Docker image you created in step 10.
> - You need to create a Docker image for the Apache ActiveMQ Artemis broker. For more information, see https://github.com/apache/activemq-artemis/tree/main/artemis-docker.

b. Apply configurations in the master node by executing the following commands:

```
kubectl apply -f https://github.com/kedacore/keda/releases/download/v2.6.0/keda-2.6.0.yaml
kubectl apply -f simulation.deployment.yml
kubectl apply -f simulation.service.yml
kubectl apply -f simulation.job.yml
kubectl apply -f artemis.deployment.yml
kubectl apply -f artemis.service.yml
```

c. Set up load balancer configurations by doing one of the following:

- If you are working in an environment such as a bare-metal server or raw VM, where load balancers are not available on demand, complete the following steps:

  i. Apply the configuration by executing the following command:

  ```
  kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx
  /controller-v1.1.1/deploy/static/provider/baremetal/deploy.yaml
  kubectl apply -f ingress.yml
  ```

  ii. To expose the load balancer on port 80, execute the following command:

  ```
  kubectl patch deployment ingress-nginx-controller -n ingress-nginx -p '{"spec":
  {"template":{"spec":{"hostNetwork":true}}}}'
  ```

  Now the load balancer will be deployed on one of your cluster nodes.

  iii. If you want to deploy the load balancer on a specific node, execute the following command:

  ```
  kubectl patch deployment ingress-nginx-controller -n ingress-nginx -p '{"spec":
  {"template":{"spec":{"nodeSelector":{"kubernetes.io/hostname":
  "<node_hostname>"}}}}}'
  ```

  > ⚠ Replace <node_hostname> with the hostname of the desired cluster node.

- If you are on a cloud environment that supports services of the LoadBalancer type, execute the following command:

  ```
  kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.
  1.1/deploy/static/provider/cloud/deploy.yaml
  ```

  > ⚠ In most cases, this should be enough to install the load balancer. If you need to apply more specific configurations, see https://kubernetes.github.io/ingress-nginx/deploy/.

12. After the load balancer configuration is set up, simulation service is available on default port (80) on the node, on which the load balancer is deployed. To test if simulation service is available, query one of the endpoints. Request and response example:

```
http://<node_hostname>/simulation/api/running
{"running":[]}
```