CONFIGURATION OF KUBERNETES CLUSTER WITH EXTERNAL LOAD BALANCER

Cognitive Assistant for Networks (CAN) Release 5.5

JULY 6, 2021

AVANSEUS TECHNOLOGY PVT. LTD.

REVISION HISTORY

Version	Date	Change description	Created by	Updated by	Reviewed by
V 1.0	July, 2021	Initial Release	Hemanth/Umesh	Sandeep Singh	Chiranjib

Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

avanseus

Table of Contents

Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

1. Introduction

This document has to be referred only if there is a need of deploying external load balancer and if you wish to use EBS for persistent volumes. It is important to note that a load balancer provisioning and EBS provisioning is billable from AWS. It is advised to refer "CONFIGURATION OF KUBERNETES CLUSTER" document which uses Software load balancer.

2. Prerequisites

- We assume that the Docker and Kubernetes have been already installed successfully on the master and all the worker nodes. Helm should be installed only in the master node.
- Root access to the servers is mandatory.

3. Kubernetes Cluster Setup

3.1. Steps to set up Cluster for Kubernetes Master Node

Follow the below steps only in the master node:

Step 1. Configuring Cloud Provider as AWS.

Edit the file and configure the cloud provided as AWS. Login to the master node and open the **10-kubeadm.conf** using the below command:

```
vi /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Append only "**--cloud-provider=aws**" at the end of the file as shown below:

Add the above configuration as mentioned in the screenshot and save the file.

Step 2. Initializing Cluster from Master Node.

Start Kubernetes cluster from the master machine. You will find all the files required to configure the Cluster in the folder **kubernetes_resources/AWS_NOTES** of the git repository. You will find a file named **aws_kubeadm.conf**, edit this file as per server configurations before setting up the cluster.

Change the below parameters in the **aws_kubeadm.conf**:

- **<advertise-address>**: Write internal ip address of the master node here.
- **<kubernetesVersion>**: Provide installed version of kubernetes here.
- **<podSubnet>**: Change the configuration based on your choice of CNI. There are two choices of CNI that can be used:

  i. **Calico**(Preferred): For Calico, provide the <podSubnet> value as 192.168.0.0/16.

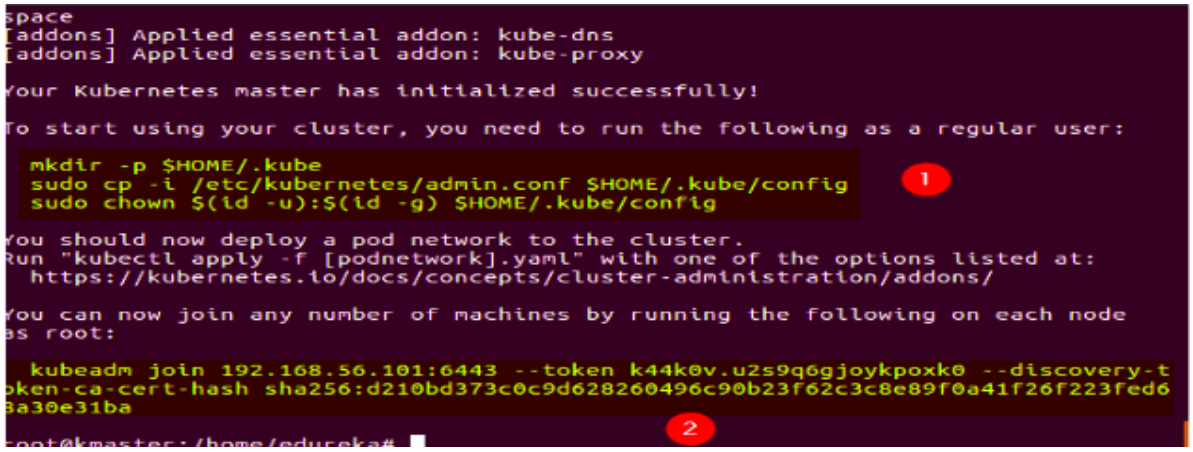
Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

ii. **Flannel**: For Flannel, provide the <podSubnet> value as 10.244.0.0/16.

After modifying all the above parameters, execute the following command to set up the cluster from the master node.

```
$sudo su
$kubeadm init --config=aws_kubeadm.conf
```

You will get the output as shown in the below screenshot:



Now, the master node has been initialized successfully. Note down few important points before you proceed with the cluster setup. There are a few more steps that need to be followed in the master node.

Important Note:

Please make a note of the join command (Marked as 2 in the above snapshot). This command will be used in 2.2 while we join worker nodes into this newly created cluster.

Step 3. Enabling kubectl command.

To start using the cluster, you need to use the following command as a regular user. Execute the following commands. (This step is marked as 1 in the above screenshot)

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

This will enable to use kubectl from the CLI. To verify kubectl, execute the below command:

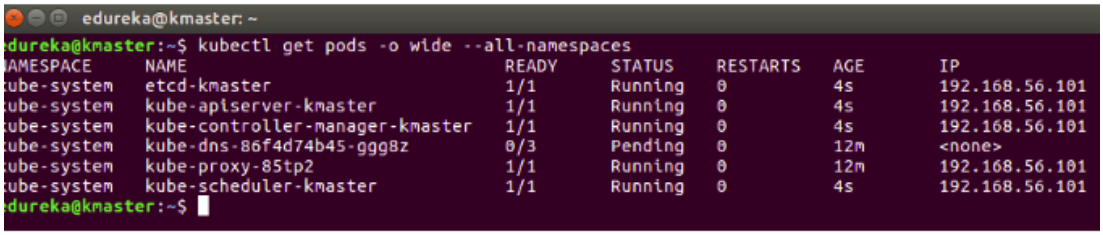
```
$kubectl version
```

Use the below command to check the pods status:

```
$kubectl get pods -o wide --all-namespaces
```


Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

You can see that some of the pod status will be shown as “**Pending**”, This is because CNI for the pod Network is not installed yet.



Step 4. Installation of CNI for the pod network

There are two choices of CNI:

1. Calico (Preferred)
2. Flannel

To use Calico CNI for the pod network, execute the below command:

```
$kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

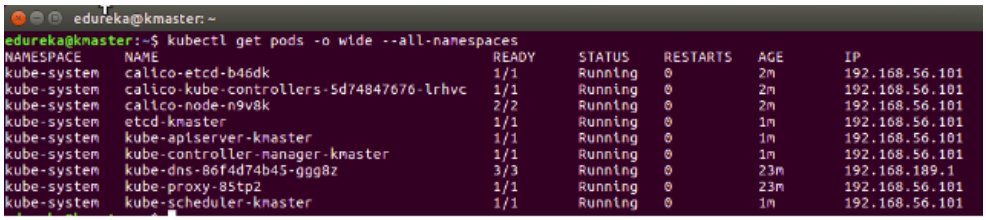
To use Flannel CNI for the pod network, execute the below command:

```
$kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

After applying the CNI pod network, check the status of each pod.

To check the status of each pod, use the below command:

```
$kubectl get pods -o wide --all-namespaces
```



You can clearly see that all the pods are up and have status as “**Running**”.

All the steps to set up cluster for Kubernetes master node are completed.

3.2. Steps to set up Cluster for Kubernetes Worker Nodes

Get your worker node to join the cluster. This is probably the only step that you will be doing on the node, after installing kubernetes on it.

You will find a file named **node.yaml** in the folder **kubernetes_resources/AWS_NOTES**, There are some parameters that have to be changed in the file **node.yaml**, before running the join command to set up the cluster from the worker node.

All the below mentioned parameters have to be updated in the **node.yaml** file based on the output we obtained during cluster creation in master node. Use the same join command output which we noted previously while configuring the cluster in the master node.

- <token>: token obtained from the join command we got as output from the master.

 Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

- <apiServerEndpoint>: endpoint <ip/domainName>:<port_no> obtained from the join command we got as output from the master.
- <caCertHashes>: hash obtained from the join command we got as output from the master.
- <name>: hostname of the worker node in which you are currently working.

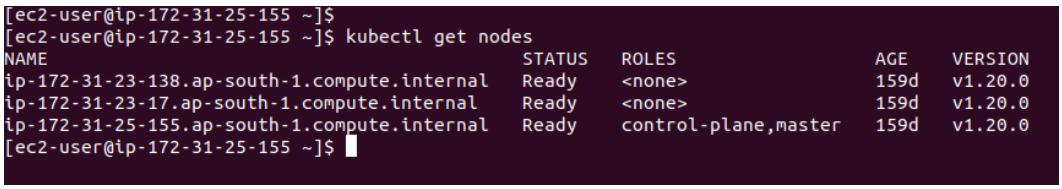
After modifying all the parameters, execute the following commands to join the worker node into our earlier created k8 cluster:

```
$sudo su
$kubeadm join --config=node.yaml
```

After all the setup has been done, verify whether the cluster is ready by executing the below command in the master node:

```
$kubectl get nodes
```

The status of all the nodes should be "**Ready**" as shown in the below screenshot.



The kubernetes cluster is set up. If you face any issues in terms of cluster setup like core-DNS not getting resolved, worker node not able to join the cluster, unhealthy worker node etc., then you skip the next section and try reinstalling the cluster from scratch. Deletion of clusters is discussed in the section 4.

4. Provisioning Elastic load balancer (ELB) and Elastic block

To provision ELB and EBS, IAM policy has to be created for both master and worker nodes from the AWS Console and the same as to be attached to your ec2 instances:

Step 1: IAM Role Master node.

Create a policy with the below configuration for the master node in the AWS console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVolumes",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:ModifyInstanceAttribute",
        "ec2:ModifyVolume",
```


Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

```
            "ec2:AttachVolume",
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:CreateRoute",
            "ec2:DeleteRoute",
            "ec2:DeleteSecurityGroup",
            "ec2:DeleteVolume",
            "ec2:DetachVolume",
            "ec2:RevokeSecurityGroupIngress",
            "ec2:DescribeVpcs",
            "elasticloadbalancing:AddTags",
            "elasticloadbalancing:AttachLoadBalancerToSubnets",
            "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
            "elasticloadbalancing:CreateLoadBalancer",
            "elasticloadbalancing:CreateLoadBalancerPolicy",
            "elasticloadbalancing:CreateLoadBalancerListeners",
            "elasticloadbalancing:ConfigureHealthCheck",
            "elasticloadbalancing:DeleteLoadBalancer",
            "elasticloadbalancing:DeleteLoadBalancerListeners",
            "elasticloadbalancing:DescribeLoadBalancers",
            "elasticloadbalancing:DescribeLoadBalancerAttributes",
            "elasticloadbalancing:DetachLoadBalancerFromSubnets",
            "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
            "elasticloadbalancing:ModifyLoadBalancerAttributes",
            "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
            "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
            "elasticloadbalancing:AddTags",
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:CreateTargetGroup",
            "elasticloadbalancing:DeleteListener",
            "elasticloadbalancing:DeleteTargetGroup",
            "elasticloadbalancing:DescribeListeners",
            "elasticloadbalancing:DescribeLoadBalancerPolicies",
            "elasticloadbalancing:DescribeTargetGroups",
            "elasticloadbalancing:DescribeTargetHealth",
            "elasticloadbalancing:ModifyListener",
            "elasticloadbalancing:ModifyTargetGroup",
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
            "iam:CreateServiceLinkedRole",
            "kms:DescribeKey"
        ],
        "Resource": [
            "*"
        ]
    }
  ]
}
```

Attach the above created policy to Master ec2 instance from the AWS console.

Step 2: IAM Role Worker node.

Create another policy with the below configuration for the worker node in the same way in the AWS console.

```
{
  "Version": "2012-10-17",
```

 Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    }
  ]
}
```

Attach the above created policy to all the Worker node ec2 instances from the AWS console.

5. Troubleshooting the Kubernetes Cluster creation

This section needs to be followed only if you face any issues or unseen challenges in creating the kubernetes cluster or if joining worker nodes to the cluster create issues. This section helps in purging all the kubernetes cluster related dependencies before setting up a new cluster.

5.1. Steps to follow in Kubernetes Master Node

Execute the below commands to reset the complete cluster in master node:

```
$kubeadm reset -f
$rm -rf /etc/cni /etc/kubernetes /var/lib/dockershim/var/lib/etcd /var/lib/kubelet /var/run/kubernetes ~/.kube/*
```

5.2. Steps to follow in Kubernetes Worker Nodes

Execute the below commands to reset the complete cluster in worker nodes:

```
$rm -rf /etc/cni /etc/kubernetes /var/lib/dockershim var/lib/etcd /var/lib/kubelet /var/run/kubernetes ~/.kube/*
$systemctl daemon-reload && systemctl restart kubelet
```

Once the above commands are executed, start setting up the cluster again. To set up the cluster again, refer this document from start.

Copyright © 2021 Avanseus Holdings Pte. Ltd. All rights reserved.