How To deploy Kubernetes cluster using Ingress Controller on tomcat web service

* [Top DevOps Interview Questions You Must Prepare In 2019](https://www.edureka.co/blog/interview-questions/top-devops-interview-questions-2016/)
* [Top Git Interview Questions – All You Need To Know About Git](https://www.edureka.co/blog/interview-questions/git-interview-questions/)
* [Top Jenkins Interview Questions – All You Need To Know About Jenkins](https://www.edureka.co/blog/interview-questions/jenkins-interview-questions/)
* [Top Docker Interview Questions – All You Need To Know About Docker](https://www.edureka.co/blog/interview-questions/docker-interview-questions/)
* [Top 50 Kubernetes Interview Questions You Must Prepare In 2019](https://www.edureka.co/blog/interview-questions/kubernetes-interview-questions/)
* [Top Puppet Interview Questions – All You Need To Know About Puppet](https://www.edureka.co/blog/interview-questions/puppet-interview-questions/)
* [Top Chef Interview Questions – All You Need To Know About Chef](https://www.edureka.co/blog/interview-questions/chef-interview-questions/)
* [Top Nagios Interview Questions – All You Need To Know About Nagios](https://www.edureka.co/blog/interview-questions/nagios-interview-questions/)
* **Create a Kubernetes cluster on AWS**

In order to deploy any kind of service/application on Kubernetes, is a cluster. Every cluster consists of one master and single or multiple nodes depending on the requirement. In this demo, I’m going to show you how to create a Kubernetes cluster on AWS.

**Step 1:** Create an instance, name it kubectl. We’re going to deploy the cluster using this instance. This instance only has the kubectl tool installed that will interact with the master, it does not have Kubernetes installed on it.

Note: I am using three services here – s3 bucket which stores all the files, EC2 which is used to create an instance and deploy the service and IAM which is used to configure permissions and create roles

**Step 2:** Create a Role in the IAM section

Attach the appropriate policy to your Role (for this example admin access is given)

Next, it’ll ask you to add tags which are optional. In my case, I haven’t attached any tags.

Give your Role a name and review the policies assigned to it and then press **Create role**.

**Step 3:** Attach the role to the instance. Go to **instance settings -> Attach/Replace IAM role** -> attach the role created and then click on **Apply**.

**Step 4:** Once created the instance and attached the role, open the command emulator i.e. cmder or putty and connect to the AWS instance. I used cmder. Once connected to the instance, update the repository and install aws-cli using the following commands:

$ sudo apt-get install

$ sudo apt-get install awscli

**Step 5:** Install and set up kubectl using the following commands:

$ sudo apt-get update && sudo apt-get install -y apt-transport-https

$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

$ echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

$ sudo apt-get update

$ sudo apt-get install -y kubectl

**Step 6:** Install Kops on the system using the following commands:

$ wget https://github.com/kubernetes/kops/releases/download/1.10.0/kops-linux-amd64

$ chmod +x kops-linux-amd64

$ mv kops-linux-amd64 /usr/local/bin/kops

**Step 7:** With Kops installed, configure a domain for cluster to access it from outside. Create a hosted zone for it

**Services-> Route53-> Hosted zones-> Create Hosted Zone**

Add a domain name for your cluster, change the type from **Public Hosted Zone** to **Private Hosted Zone for Amazon VPC** and copy your instance **VPC ID** from the instance page to the VPC ID column and add the region you want to create your hosted zone in and then Copy the VPC ID and need to add Domain name and VPC ID

After which Hosted Zone is created.

**Step 8:** Create a bucket as the same name as domain name using the following command:

$ aws s3 mb s3://kube-demo.com

Once after creating the bucket, execute the following command:

$ export KOPS\_STATE\_STORE=s3://kube-demo.com

**Step 9:** Before creating the cluster, We have to create SSH public key.

$ ssh-keygen

Enter file where you want your key pair to be saved and create a password to access the ssh public key. In this case, I’ve chosen the default location and used no password.

**Step 10:** Now that after creating the SSH key, create the cluster using the following command:

$ kops create cluster –cloud=aws –zones=us-east-1a –name=useast1.kube-demo.com –dns-zone=kube-demo.com –-dns private

And then update the cluster using the below command

$ kops update cluster useast1.kube-demo.com

This will create the resources needed for your cluster to run. It will create a master and two node instances.

Now when we check the instances, we would see three new instances that would have got created. One of them will be master node and the two other nodes. Cluster has been created.

Your s3 bucket will now have some folder in it, which is basically your cluster configuration file.

**Step 11:** Now if **ssh** into master node and do a kubectl get nodes using below commands. We will see that node is in ready state.

$ ssh  -i .ssh/id\_rsa admin@ipv4-public-ip-of-master

$ kubectl get nodes

* **Create an Ingress(nginx) loadbalancer controller**

**Step 1:** The following command is mandatory for all configurations

$ kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/mandatory.yaml>

**Step 2:** Now the next command depends upon the environment you’re using your cluster in.

I used AWS L4 configuration:

$ kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/service-l4.yaml>

$ kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/patch-configmap-l4.yaml>

**Step 3:** Check the pods to see all the Ingress pods are up and running

$ kubectl get pods --all-namespaces

**Step 4:** Check the services to verify Ingress service is working

$ kubectl get svc --all-namspaces

**Step5:** Now create a HTML deployment

**Step 6:** Create tomcat web application server with a HTML page using below command

$ kubectl create tomcat web application html --tcp=80:80

**Step 7:** Curl the service IP to make sure it is attached to the pods

$ curl <Cluster IP address>

**Step 8:** Now, create an ingress rule for the service so we can access the service add /test to route the external traffic.

$ vi ingress.yaml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

 name: test-ing

 annotations:

   nginx.ingress.kubernetes.io/rewrite-target: /

spec:

 rules:

 - http:

     paths:

     - path: /test

       backend:

         serviceName: html

         servicePort: 80

**Step 9:** Deploy the Ingress rule using below command

$ kubectl apply -f ingress.yaml

**Step 10:** Now copy the Ingress service external IP and add/test to it in browser to verify.