# Steps to create a k8s cluster on AWS using a bootstrap instance

## 1. Install KOPS :

curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag\_name | cut -d '"' -f 4)/kops-darwin-amd64

chmod +x ./kops

sudo mv ./kops /usr/local/bin/

## 2. Install kubectl :

curl -Lo kubectl https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

## 3. Create User kops and assign proper permissions to the user.

In order to build clusters within AWS we'll create a dedicated IAM user for kops. This user requires API credentials in order to use kops.

aws iam create-group --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonRoute53FullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/IAMFullAccess --group-name kops

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonVPCFullAccess --group-name kops

aws iam create-user --user-name kops

aws iam add-user-to-group --user-name kops --group-name kops

aws iam create-access-key --user-name kops

## 4. You should record the SecretAccessKey and AccessKeyID in the returned JSON output, and then use them below:

# configure the aws client to use your new IAM user

aws configure # Use your new access and secret key here

aws iam list-users # you should see a list of all your IAM users here

# Because "aws configure" doesn't export these vars for kops to use, we export them now

export AWS\_ACCESS\_KEY\_ID=$(aws configure get aws\_access\_key\_id)

export AWS\_SECRET\_ACCESS\_KEY=$(aws configure get aws\_secret\_access\_key)

## 5. Cluster state storage

In order to store the state of your cluster, and the representation of your cluster, we need to create a dedicated S3 bucket for kops to use. This bucket will become the source of truth for our cluster configuration. In this guide we'll call this bucket example-com-state-store, but you should add a custom prefix as bucket names need to be unique.

aws s3api create-bucket \

--bucket prefix-example-com-state-store \

--region us-east-1

Note: It is **STRONGLY** recommended versioning your S3 bucket in case you ever need to revert or recover a previous state store.

aws s3api put-bucket-versioning --bucket prefix-example-com-state-store --versioning-configuration Status=Enabled

## 6. Creating your first cluster

### a. Prepare local environment

For a gossip-based cluster, make sure the name ends with k8s.local. For example:

export NAME=myfirstcluster.k8s.local

export KOPS\_STATE\_STORE=s3://prefix-example-com-state-store

Note: You don’t have to use environmental variables here. You can always define the values using the –name and –state flags later.

### b. Create cluster configuration

Below is a create cluster command. We'll use the most basic example possible, with more verbose examples in [high availability](https://github.com/kubernetes/kops/blob/master/docs/operations/high_availability.md#advanced-example). The below command will generate a cluster configuration, but not start building it. Make sure that you have generated SSH key pair before creating the cluster.

ssh-keygen -b 2048 -t rsa -f ~/.ssh/id\_rsa

kops create cluster --zones us-west-2a,us-west-2b,us-west-2c ${NAME}

kops create secret --name ${NAME} sshpublickey admin -i ~/.ssh/id\_rsa.pub

### c. Customize Cluster Configuration

Now we have a cluster configuration, we can look at every aspect that defines our cluster by editing the description.

kops edit cluster ${NAME}

This opens your editor (as defined by $EDITOR) and allows you to edit the configuration. The configuration is loaded from the S3 bucket we created earlier, and automatically updated when we save and exit the editor.

We'll leave everything set to the defaults for now, but the rest of the kops documentation covers additional settings and configuration you can enable.

For instance group configuration change :

kops edit ig nodes --name ${NAME}

kops get ig --name ${NAME}

### d. Build the cluster

Now we take the final step of actually building the cluster. This'll take a while. Once it finishes you'll have to wait longer while the booted instances finish downloading Kubernetes components and reach a "ready" state.

kops update cluster ${NAME} --yes

To check status :

kops validate cluster

### e. use the cluster

Remember when you installed kubectl earlier? The configuration for your cluster was automatically generated and written to ~/.kube/config for you!

A simple Kubernetes API call can be used to check if the API is online and listening. Let's use kubectl to check the nodes.

kubectl get nodes

You will see a list of nodes that should match the --zones flag defined earlier. This is a great sign that your Kubernetes cluster is online and working.

Also kops ships with a handy validation tool that can be ran to ensure your cluster is working as expected.

kops validate cluster

You can look at all the system components with the following command.

kubectl -n kube-system get po

### f. delete the cluster

Running a Kubernetes cluster within AWS obviously costs money, and so you may want to delete your cluster if you are finished running experiments.

You can preview all of the AWS resources that will be destroyed when the cluster is deleted by issuing the following command.

kops delete cluster --name ${NAME}

When you are sure you want to delete your cluster, issue the delete command with the --yes flag. Note that this command is very destructive, and will delete your cluster and everything contained within it!

kops delete cluster --name ${NAME} --yes