Kubernetes kOps: Step-By-Step Example & Alternatives (https://blog.kubecost.com/blog/kubernetes-kops/)

12 minute read

Creating, deploying, and managing highly available Kubernetes clusters for production environments can be complex and time-consuming. Things get even more complicated when you consider infrastructure tasks like provisioning AWS resources. Kubernetes kOps helps teams solve this problem by providing a simple and scalable method for configuring and maintaining production-grade clusters and the associated cloud infrastructure.

To help you get started with kOps, we'll describe kOps and its key features, compare kOps with popular alternatives, and walk through an example of how to use kOps on AWS so you can get hands-on with this helpful Kubernetes tool.

What is kOps?

Kubernetes kOps is a free and open-source command-line tool for configuring and maintaining Kubernetes clusters and provisioning the cloud infrastructure needed to run them.

With kOps, teams can automate the management of Kubernetes clusters. For example, kOps can create, apply, and update cluster configurations. It can also provision the cloud infrastructure that clusters require, such as (for AWS) virtual private clouds (VPCs), autoscaling groups, security groups, and IAM roles.

Supported platforms

Kubernetes kOps is officially supported and widely used on AWS, and is expanding to support multiple additional cloud platforms. The table below shows kOps cloud platform support at the start of 2022.

Officially supported	Amazon Web Services (AWS)
Beta support	DigitalOcean OpenStack Google Cloud Engine (GCE)
Alpha support	Azure AliCloud

Key Kubernetes kOps features

Kubernetes kOps offers a variety of features to streamline cluster management, including:

- **Full automation—**kOps automates everything from Kubernetes bootstrap to provisioning of required infrastructure.
- **Cluster identification using DNS—**kOps uses DNS to identify and connect cluster resources. Using DNS has the advantage of uniquely identifying any service or resource from inside or outside of the cluster.
- **Broad operating system (OS) support**—kOps supports** **multiple OSes such as Ubuntu, CentOS, and Amazon Linux. For the latest list of all supported OSes, check the official distro support matrix (https://github.com/kubernetes/kops/blob/master/docs/operations/images.md#distros-support-matrix).
- **Flexible provisioning**—kOps can provision the Kubernetes clusters and related infrastructure or generate Terraform manifests that users can separately apply using Terraform.
- **Leverages**—Auto-Scaling Groups (ASG) for self-healing**—if you're running kOps in AWS, instances used for the Kubernetes cluster are part of an AWS ASG, allowing you to replace cluster nodes when they fail.

You can check this short ASCII cast (https://asciinema.org/a/97298) to see kOps in action.

Alternatives to Kubernetes kOps

kOps is a good choice for teams looking for a flexible and straightforward cluster management tool, particularly if they use the AWS cloud. However, it's not the only tool available for this purpose, and there are several alternatives to kOps. The table below details the pros and cons of some of the most popular Kubernetes kOps alternatives.

Tool	Pros	Cons
Kubespray	Broad platform support—Supported platforms include: AWS GCE Azure OpenStack vSphere Equinix Metal (bare metal) Oracle Cloud Infrastructure (Experimental) Bare metal Ansible-based—Kubespray is based on Ansible and can easily integrate into environments already using Ansible for configuration management. Feature-rich—Kubespray performs generic configuration management, Kubernetes cluster setup, and control plane bootstrapping. Platform agnostic—Kubespray's design is generic, and it can be used to manage the Kubernetes clusters across various cloud platforms.	No infrastructure provisioning—Kubespray does not support the provisioning of infrastructure.
EKSCTL	Excellent AWS Elastic Kubernetes Service (EKS) support—EKSCTL is designed for configuration and management of Kubernetes in AWS EKS and tightly integrates with the platform. CloudFormation support—EKSCTL uses AWS CloudFormation templates which makes automation in the AWS ecosystem easier and more scalable. Infrastructure management—EKSCTL can manage and configure all the infrastructure needed for Kubernetes clusters.	Only supports AWS—EKSCTL doesn't support any other cloud providers.
kubeadm	Bootstrapping with minimal configuration—With kubeadm teams can bootstrap a Kubernetes cluster with very little configuration overhead. Quickly spin up basic clusters—One of kubeadm's primary use cases is quickly setting up a minimum viable cluster. Simple upgrades—Like with cluster bootstrapping, cluster upgrades with kubeadm are straightforward.	No infrastructure provisioning—kubeadm does not support the provisioning of infrastructure.

How to set up a Kubernetes Cluster in AWS with kOps

Now that we understand what Kubernetes kOps is, let's move on to a hands-on tutorial. In this section, we'll follow a step-by-step process to set up, modify, and delete a Kubernetes cluster in AWS using kOps.

Prerequisites

To follow along, you'll need to have the following:

1. **An AWS account and the AWS CLI configured with admin credentials—**If you're not familiar with the AWS CLI, refer to these pages for installation (https://aws.amazon.com/cli/) and configuration (https://docs.aws.amazon.com/sdk-for-go/v1/developer-guide/configuring-sdk.html#specifying-credentials).

2. **kubectl**—Ensure you have installed kubectl locally. You can refer to this guide (https://kubernetes.io/docs/tasks/tools/install-kubectl/) for detailed installation steps on Linux, macOS, or Windows.

3. **jq—**You can download and install this simple text manipulation tool here (https://stedolan.github.io/jq/download/).

4. **An active domain with a dedicated "kops" subdomain**—kOps depends on DNS records for internal and external communication. For this guide, we assume you have an active domain and that you dedicate a subdomain for this kOps setup, e.g., kops.yourdomain.com. We'll use "kops.yourdomain.com" in our examples. When executing commands, replace kops.yourdomain.com with your "kops" subdomain.

Step 1: Create the "kops" AWS IAM user

Create an IAM user called "kops" with required permissions by running the following commands in the AWS CLI:

```
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 attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonSQSFullAccess --group-name kops;
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess --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;
```

Record the SecretAccessKey and AccessKeyID values in the output of the previous command, and then use them in the commands below:

```
# configure the AWS CLI to use 'kops' user
aws configure       # use the new access and secret key
aws iam list-users  # you should see a list of all your IAM users here

#Export the following variables for a session:
export AWS_ACCESS_KEY_ID=$(aws configure get aws_access_key_id)
export AWS_SECRET_ACCESS_KEY=$(aws configure get aws_secret_access_key)
```

Step 2: Configure DNS setup

Next, create a hosted zone in AWS for your kops subdomain:

```
#install jq locally before running the below command
aws route53 create-hosted-zone --name kops.yourdomain.com --caller-reference $(uuidgen) | jq .DelegationSet.NameServers
```

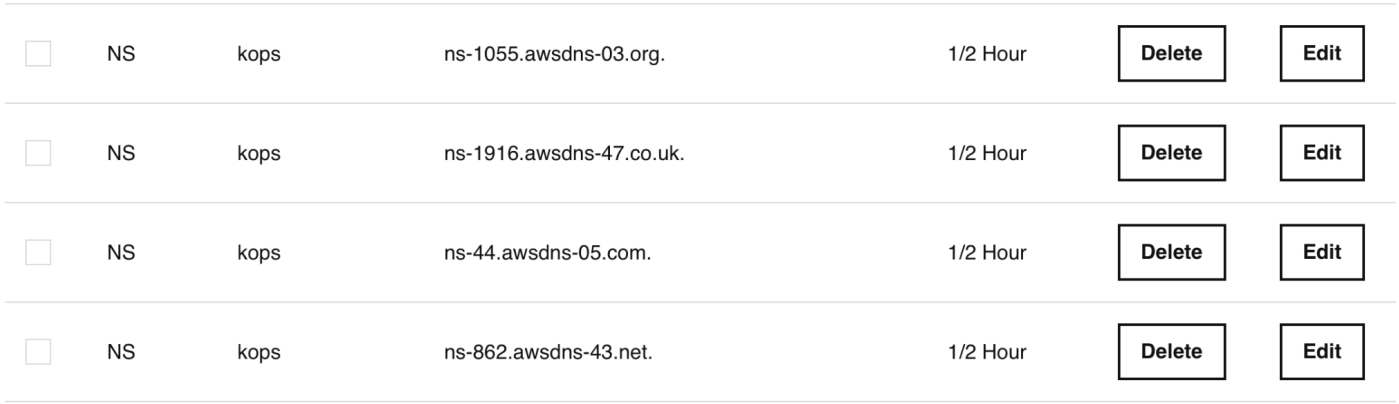
You should see output similar to this:

```
aws route53 create-hosted-zone --name kops.yourdomain.com --caller-reference $(uuidgen) | jq
.DelegationSet.NameServers

[
  "ns-1055.awsdns-03.org",
  "ns-862.awsdns-43.net",
  "ns-44.awsdns-05.com",
  "ns-1916.awsdns-47.co.uk"
]
```

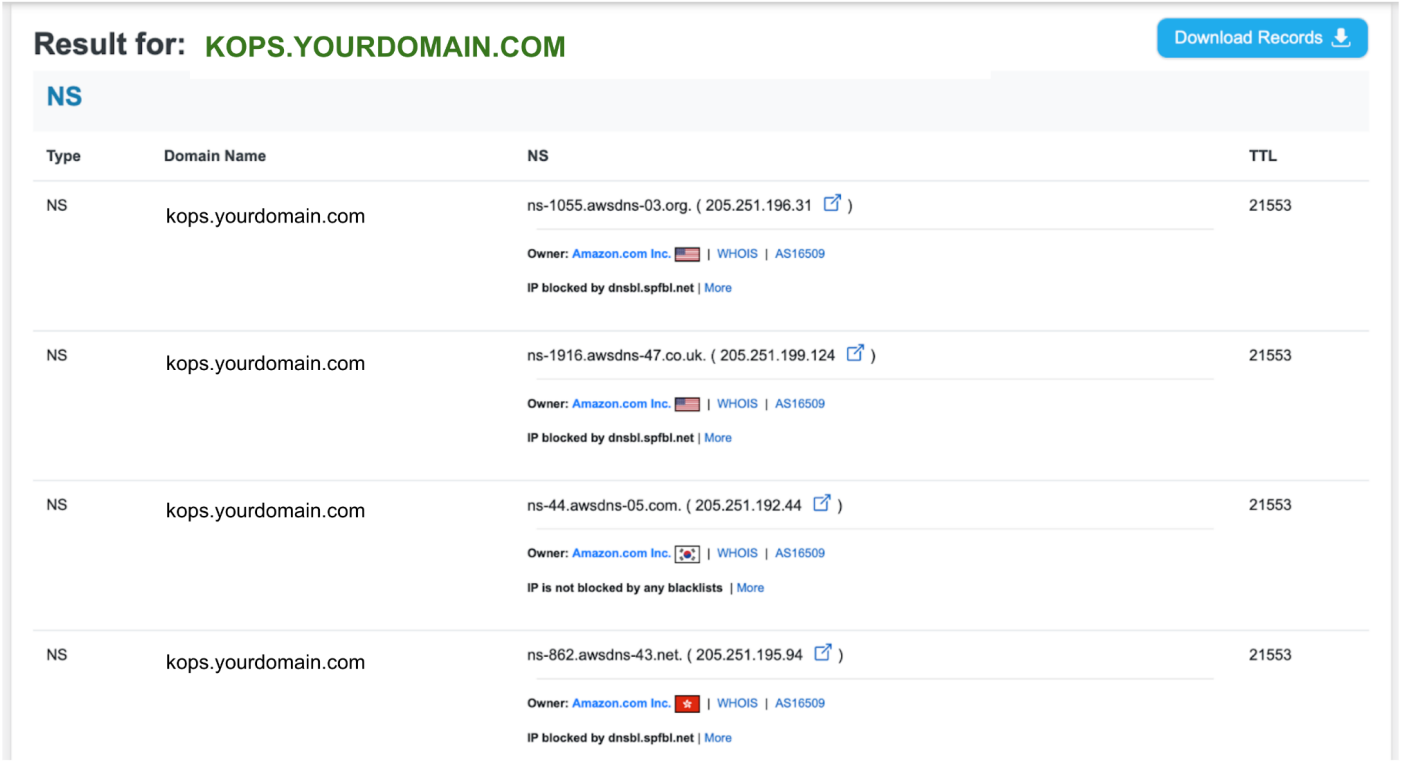
Record the nameservers (NS entries) from the above command output.

Next, you need to add these NS records for the subdomain in the DNS setting of the main domain. Exactly how you need to add the subdomain will vary depending on your domain registrar. For example, with GoDaddy, you would need to create the following NS records:



Example of NS records in Godaddy portal.

Wait for a few minutes and verify the subdomain's NS records. This step is crucial to ensure the NS records are correctly set. You can use https://dnschecker.org/ns-lookup.php (https://dnschecker.org/ns-lookup.php) for NS record lookup.



The output of NS Lookup in dnschecker.org

Another way of verifying the NS records is by using the dig tool shown below:

```
dig ns kops.yourdomain.com +short
```

Example output:

```
dig ns kops.yourdomain.com +short
ns-1055.awsdns-03.org.
ns-1916.awsdns-47.co.uk.
ns-44.awsdns-05.com.
ns-862.awsdns-43.net.
```

The NS records must be correct for the rest of the steps to work. Don't proceed unless your output is similar to the example output above.

Step 3: Create cluster state storage

kOps stores its configurations, keys, and related items, in an S3 bucket to manage Kubernetes clusters. You need to create a dedicated S3 bucket for this purpose. Run the below command to create an S3 bucket named "kops-state-storage-cluster":

```
aws s3api create-bucket \
    --bucket kops-state-storage-cluster \
    --region ap-south-1 \
    --create-bucket-configuration LocationConstraint=ap-south-1
```

Step 4: Install kOps

Next, install kOps version v1.22.2 (the latest version at the time of writing). The installation process varies depending on your local operating system.

For macOS:

```
curl -Lo kops https://github.com/kubernetes/kops/releases/download/v1.22.2/kops-darwin-amd64
chmod +x kops
sudo mv kops /usr/local/bin/kops
```

For Linux:

```
curl -Lo kops https://github.com/kubernetes/kops/releases/download/v1.22.2/kops-linux-amd64
chmod +x kops
sudo mv kops /usr/local/bin/kops
```

For Windows:

1. Download kops-windows-amd64 from the releases page (https://github.com/kubernetes/kops/releases/tag/v1.22.2)
2. Rename kops-windows-amd64 to kops.exe
3. Update the Path environment variable to point to the location of the folder containing kops.exe

Step 5: Create the Kubernetes cluster

Now that you have taken care of all the prerequisites, the next step is to create the Kubernetes cluster. First, you need to set up the name and point to the S3 bucket so that kOps can be aware of it.

```
export NAME=kops.yourdomain.com
export KOPS_STATE_STORE=s3://kops-state-storage-cluster
```

Then, create the Kubernetes cluster in the “ap-south-1” region and under the “ap-south-1a” zone. Read this (https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html) for more info on AWS regions and zones. While we’re using “ap-south-1a” in our examples, you can select a preferred region and zone.

```
kops create cluster \
    --zones=ap-south-1a \
    --name ${NAME}
```

Verify you see your cluster listed with this command:

```
kops get cluster
NAME                    CLOUD    ZONES
kops.yourdomain.com     aws     ap-south-1a
```

Now, create the Kubernetes cluster with this command:

```
kops update cluster --name kops.yourdomain.com --yes --admin
```

Output should look similar to this:

```
kops update cluster --name kops.yourdomain.com --yes --admin

I1212 10:54:01.269309   29532 executor.go:111] Tasks: 0 done / 89 total; 44 can run
W1212 10:54:01.473903   29532 vfs_castore.go:377] CA private key was not found
I1212 10:54:01.503298   29532 keypair.go:213] Issuing new certificate: "etcd-clients-ca"
I1212 10:54:01.511183   29532 keypair.go:213] Issuing new certificate: "etcd-manager-ca-
events"
I1212 10:54:01.511206   29532 keypair.go:213] Issuing new certificate: "etcd-manager-ca-main"
I1212 10:54:01.522435   29532 keypair.go:213] Issuing new certificate: "etcd-peers-ca-main"
I1212 10:54:01.523116   29532 keypair.go:213] Issuing new certificate: "etcd-peers-ca-events"
I1212 10:54:01.524579   29532 keypair.go:213] Issuing new certificate: "apiserver-aggregator-
ca"
W1212 10:54:01.580788   29532 vfs_castore.go:377] CA private key was not found
I1212 10:54:01.626277   29532 keypair.go:213] Issuing new certificate: "kubernetes-ca"
I1212 10:54:01.626549   29532 keypair.go:213] Issuing new certificate: "service-account"
I1212 10:54:02.720787   29532 executor.go:111] Tasks: 44 done / 89 total; 19 can run
I1212 10:54:04.482780   29532 executor.go:111] Tasks: 63 done / 89 total; 24 can run
I1212 10:54:05.158356   29532 executor.go:111] Tasks: 87 done / 89 total; 2 can run
I1212 10:54:06.222246   29532 executor.go:155] No progress made, sleeping before retrying 2
task(s)
I1212 10:54:16.226912   29532 executor.go:111] Tasks: 87 done / 89 total; 2 can run
I1212 10:54:17.882117   29532 executor.go:111] Tasks: 89 done / 89 total; 0 can run
I1212 10:54:19.626484   29532 dns.go:234] Pre-creating DNS records
I1212 10:54:20.840771   29532 update_cluster.go:326] Exporting kubeconfig for cluster
Kops has set your kubectl context to kops.yourdomain.com

Cluster is starting.  It should be ready in a few minutes.

Suggestions:
* [ ] validate cluster: kops validate cluster --wait 10m
* [ ] list nodes: kubectl get nodes --show-labels
* [ ] ssh to the master: ssh -i ~/.ssh/id_rsa ubuntu@api.kops.yourdomain.com
* [ ] the ubuntu user is specific to Ubuntu. If not using Ubuntu please use the appropriate
user based on your OS.
* read about installing addons at: https://kops.sigs.k8s.io/operations/addons.↙td>
```

Wait for about 10 minutes for the cluster to come up. You can run this command to validate the cluster's health:

```
kops validate cluster --wait 10m
```

When the cluster is ready, you will see output similar to this:

```
INSTANCE GROUPS
NAME                    ROLE    MACHINETYPE     MIN     MAX     SUBNETS
master-ap-south-1a      Master  t3.medium       1       1       ap-south-1a
nodes-ap-south-1a       Node    t3.medium       1       1       ap-south-1a

NODE STATUS
NAME                                            ROLE    READY
ip-172-20-37-204.ap-south-1.compute.internal    node    True
ip-172-20-52-224.ap-south-1.compute.internal    master  True

Your cluster kops.yourdomain.com is ready
```

Now you can interact with the Kubernetes cluster using kubectl. For example:

```
kubectl get nodes
NAME                                            STATUS   ROLES                  AGE    VERSION
ip-172-20-37-204.ap-south-1.compute.internal    Ready    node                   100m   v1.22.4
ip-172-20-52-224.ap-south-1.compute.internal    Ready    control-plane,master   101m   v1.22.4
```

You can see that **two** instances are running—one master and one node. Let us deploy a simple Nginx workload and see if you can load the website in a browser.

```
kubectl create deployment my-nginx --image=nginx --replicas=1 --port=80;
kubectl expose deployment my-nginx --port=80 --type=LoadBalancer;
```

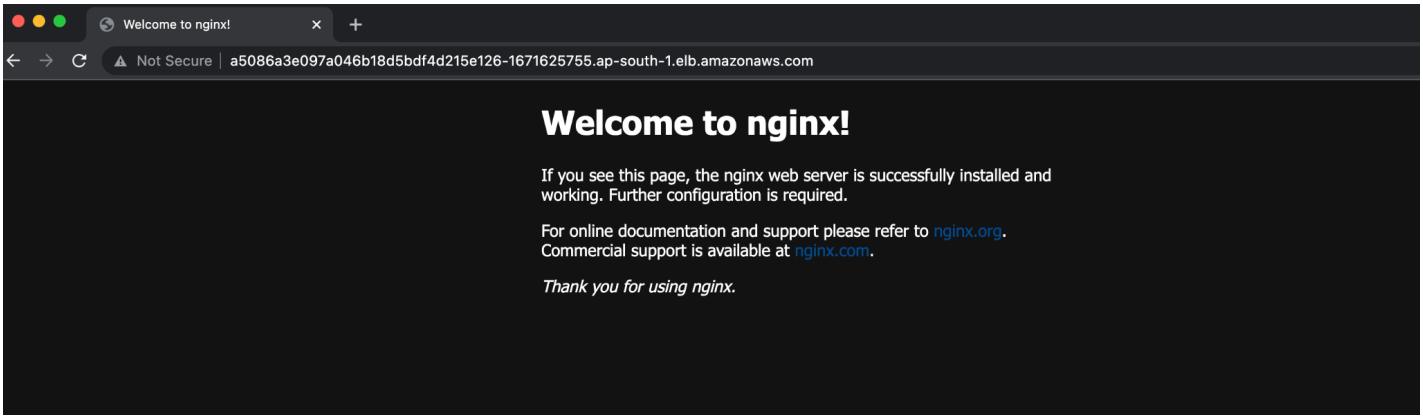
Verify if the Nginx pods are running:

```
kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
my-nginx-7576957b7b-w4pr6   1/1     Running   0          3m54s
```

Get the Load Balancer (LB) address:

```
kubectl get svc my-nginx
NAME       TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE
my-nginx   LoadBalancer   100.67.50.233   a5086a3e097a046b18d5bdf4d215e126-1671625755.ap-
south-1.elb.amazonaws.com   80:30258/TCP   4m31s
```

Here, `5086a3e097a046b18d5bdf4d215e126-1671625755.ap-south-1.elb.amazonaws.com` is the DNS name (endpoint) of the LB. Copy and paste it into a browser. You should see an Nginx default page:



The Kubernetes cluster is working as expected. In the next section, we'll modify the Kubernetes cluster using kOps.

Step 6: Modify the Kubernetes cluster

Earlier, we saw that kOps created one master and one node by default.

```
kubectl get nodes
NAME                                            STATUS   ROLES                  AGE    VERSION
ip-172-20-37-204.ap-south-1.compute.internal    Ready    node                   138m   v1.22.4
ip-172-20-52-224.ap-south-1.compute.internal    Ready    control-plane,master   140m   v1.22.4
```

Now, let's increase the node count of the cluster from 1 node to 3 nodes. For this, we need to learn the instance group name by running the command:

```
kops get instance groups
Using cluster from kubectl context: kops.yourdomain.com

NAME                    ROLE    MACHINETYPE     MIN     MAX     ZONES
master-ap-south-1a      Master  t3.medium       1       1       ap-south-1a
nodes-ap-south-1a        Node   t3.medium       1       1       ap-south-1a
```

Here, the instance group name 'nodes-ap-south-1a' is for the node role. You can edit it and update the 'maxSize' and 'minSize' to values 3. First, open the editor with this command:

```
kops edit instancegroups nodes-ap-south-1a
```

Then, find and edit the ‘maxSize’ and ‘minSize’ to match the example below.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: kops.k8s.io/v1alpha2
kind: InstanceGroup
metadata:
  creationTimestamp: "2021-12-12T05:16:35Z"
  labels:
    kops.k8s.io/cluster: kops.yourdomain.com
  name: nodes-ap-south-1a
spec:
  image: 099720109477/ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-20211118
  instanceMetadata:
    httpPutResponseHopLimit: 1
    httpTokens: required
  machineType: t3.medium
  maxSize: 3
  minSize: 3
  nodeLabels:
    kops.k8s.io/instancegroup: nodes-ap-south-1a
  role: Node
  subnets:
 —ap-south-1a
```

Save and quit the editor. Apply the changes by running:

```
kops update cluster --name kops.yourdomain.com --yes --admin
```

After a few minutes, you can verify that the node count is 3.

```
kubectl get nodes
NAME                                          STATUS   ROLES                  AGE
VERSION
ip-172-20-32-164.ap-south-1.compute.internal   Ready    node                   17s
v1.22.4
ip-172-20-37-204.ap-south-1.compute.internal   Ready    node                   3h34m
v1.22.4
ip-172-20-52-224.ap-south-1.compute.internal   Ready    control-plane,master   3h36m
v1.22.4
ip-172-20-57-28.ap-south-1.compute.internal    Ready    node                   12s
v1.22.4↙td>
```

```
kops get instancegroups nodes-ap-south-1a
Using cluster from kubectl context: kops.yourdomain.com

NAME                    ROLE    MACHINETYPE     MIN     MAX     ZONES
nodes-ap-south-1a       Node    t3.medium       3       3       ap-south-1a
```

Step 7: Delete the demo cluster and resources

Since you are running a Kubernetes cluster in AWS, the underlying infrastructure such as EC2 instances and LoadBalancers come with a cost.—so remember to delete the cluster when you're done with the demo.

Run these following commands to delete the resources as well as the cluster:

```
kubectl delete svc my-nginx;
kubectl delete deploy my-nginx;

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

Now that you know the basics, here are a few ideas for additional exercises that can help you master kOps:

- Configure a cluster for high availability
- Transform cluster configuration into Terraform manifests
- Make the cluster private
- Read more on kops CLI usage—https://kops.sigs.k8s.io/cli/kops/ (https://kops.sigs.k8s.io/cli/kops/)

Conclusion

kOps is a versatile tool for Kubernetes cluster management. It provides an automated way to provision your cluster's underlying resources such as instances, load balancers, security groups, and volumes. If you need a good balance of control and simplicity, using kOps can help you handle your Kubernetes cluster management requirements, especially when using AWS.

Kubernetes clusters and cloud infrastructure (including those you manage with kOps) come with a cost. Kubecost simplifies the cost allocation process by measuring resource usage directly at the Kubernetes concept level, marrying consumption data with detailed billing files from your cloud provider, and allowing you to allocate costs to teams or projects. Kubecost can be installed within minutes using a Helm chart and is free to use forever on a single cluster. You can get started here (https://www.kubecost.com/install.html).

Tags: Kubernetes

Categories: blog

Updated: January 27, 2022