Creating a Cluster: Running and Verification

In this lesson, we will run and verify the cluster having the previously discussed specifications.

WE'LL COVER THE FOLLOWING

Creating the Cluster

The command that creates a cluster using the specifications we discussed is as follows.

```
kops create cluster \
    --name $NAME \
    --master-count 3 \
    --node-count 1 \
    --node-size t2.small \
    --master-size t2.small \
    --zones $ZONES \
    --master-zones $ZONES \
    --ssh-public-key devops23.pub \
    --networking kubenet \
    --kubernetes-version v1.14.8 \
    --yes
```

- **Line 3:** We specified that the cluster should have three masters and one worker node. Remember, we can always increase the number of workers, so there's no need to start with more than what we need at the moment.
- **Line 5-8:** The sizes of both worker nodes and masters are set to `t2.small`. Both types of nodes will be spread across the three availability zones we specified through the environment variable `ZONES`.
- **Line 9-10:** We defined the public key and the type of networking as discussed earlier.

- **Line 11:** We used `--kubernetes-version` to specify that we prefer to run version `v1.9.1`. Otherwise, we'd get a cluster with the latest version considered stable by kops. Even though running latest stable version is probably a good idea, we'll need to be a few versions behind to demonstrate some of the features kops has to offer.
- **Line 12:** The `--yes` argument specifies that the cluster should be created right away. Without it, `kops` would only update the state in the S3 bucket, and we'd need to execute `kops apply` to create the cluster. Such a two-step approach is preferable, but we would like to see the cluster in all its glory as soon as possible.
- **Authorization:** By default, kops sets `authorization` to `AlwaysAllow`. Since this is a simulation of a production-ready cluster, we changed it to `RBAC`, which we already explored in one of the previous chapters.

The **output** of the command is as follows.

```
...
kops has set your kubectl context to devops23.k8s.local

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

Suggestions:
 * validate cluster: kops validate cluster
 * list nodes: kubectl get nodes --show-labels
 * ssh to the master: ssh -i ~/.ssh/id_rsa admin@api.devops23.k8s.local
The admin user is specific to Debian. If not using Debian please use the appropriate user bas
 * read about installing addons: https://github.com/kubernetes/kops/blob/master/docs/addons.m
```

We can see that the `kubectl` context was changed to point to the new cluster which is starting, and will be ready soon. Further down are a few suggestions of the next actions. We'll skip them, for now.

Verification

We'll use kops to retrieve the information about the newly created cluster.

```
kops get cluster
```

The **output** is as follows.

```
NAME                CLOUD ZONES
```

```
devops23.k8s.local aws   us-east-2a,us-east-2b,us-east-2c
```

This information does not tell us anything new. We already knew the name of the cluster and the zones it runs in.

How about `kubectl cluster-info`?

```
kubectl cluster-info
```

The **output** is as follows.

```
Kubernetes master is running at https://api-devops23-k8s-local-ivnbim-609446190.us-east-2.elb
KubeDNS is running at https://api-devops23-k8s-local-ivnbim-609446190.us-east-2.elb.amazonaws

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

We can see that the master is running as well as KubeDNS. The cluster is probably ready. If in your case KubeDNS did not appear in the output, you might need to wait for a few more minutes.

We can get more reliable information about the readiness of our new cluster through the `kops validate` command.

```
kops validate cluster
```

The **output** is as follows.

```
Using cluster from kubectl context: devops23.k8s.local

Validating cluster devops23.k8s.local

INSTANCE GROUPS
NAME              ROLE   MACHINETYPE MIN MAX SUBNETS
master-us-east-2a Master t2.small    1   1   us-east-2a
master-us-east-2b Master t2.small    1   1   us-east-2b
master-us-east-2c Master t2.small    1   1   us-east-2c
nodes             Node   t2.small    1   1   us-east-2a,us-east-2b,us-east-2c

NODE STATUS
NAME                 ROLE   READY
ip-172-20-120-133... master True
ip-172-20-34-249...  master True
ip-172-20-65-28...   master True
ip-172-20-95-101...  node   True

Your cluster devops23.k8s.local is ready
```

> It may take some time for the validation to succeed.

That is useful. We can see that the cluster uses four instance groups or, to use AWS terms, four auto-scaling groups (ASGs). There's one for each master, and there's one for all the (worker) nodes.

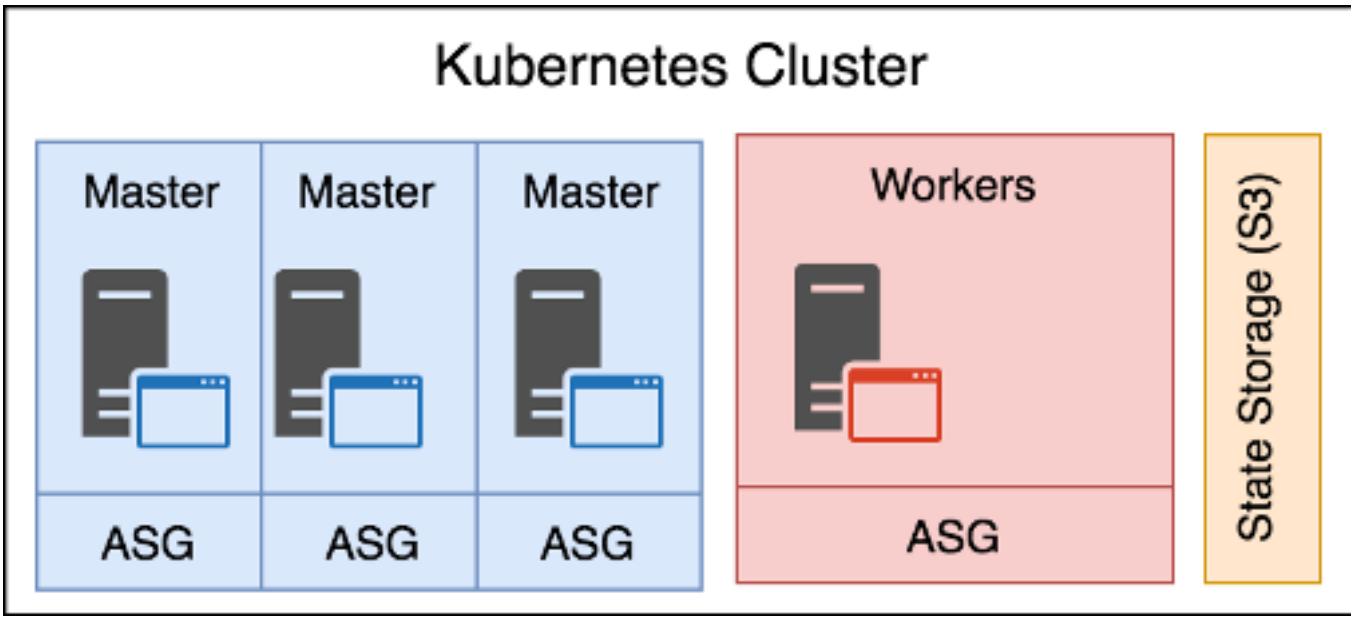
The reason each master has a separate ASG lies in the need to ensure that each is running in its own availability zone (AZ). That way we can guarantee that failure of the whole AZ will affect only one master. Nodes (workers), on the other hand, are not restricted to any specific AZ. AWS is free to schedule nodes in any AZ that is available.

We'll discuss ASGs in more detail later on.

Further down the output, we can see that there are four servers, three with masters, and one with worker node. All are ready.

Finally, we got the confirmation that our `cluster devops23.k8s.local is ready`.

Using the information we got so far, we can describe the cluster through the following illustration.



The servers that form the Kubernetes cluster

There's apparently much more to the cluster than what is depicted in the

illustration above.

In the next lesson, we'll try to discover the components that kops created for us.