What is Kops?

Kops is an official Kubernetes project for managing production-grade Kubernetes clusters. Kops is currently the best tool to deploy Kubernetes clusters to Amazon Web Services. The project describes itself as kubectl for clusters.
If you're familiar with kubectl, then you'll feel at home with Kops. It has commands for creating clusters, updating their settings, and applying changes. Kops uses declarative configuration, so it's smart enough to know how to apply infrastructure changes to existing clusters. It also has support for cluster operational tasks like scaling up nodes or horizontally scaling the cluster. Kops automates a large part of operating Kubernetes on AWS.
Before moving on to examples, let's look at its key features:

- Deploy clusters to existing virtual private clouds (VPC) or create a new VPC from scratch
- Supports public & private topologies
- Provisions single or multiple master clusters
- Configurable bastion machines for SSH access to individual cluster nodes
- Built on a state-sync model for dry-runs and automatic idempotency
- Direct infrastructure manipulation, or works with CloudFormation and Terraform
- Rolling cluster updates
- Supports heterogeneous clusters by creating multiple instance groups

Check out this short ASCII cast demo for more info.
Now, we'll tackle a common scenario: Create a cluster and configure it for your use case.

Creating Your First Kubernetes Cluster on AWS

You'll need to configure IAM permissions and an S3 bucket for the `KOPS_STATE_STORE`. The KOPS_STATE_STORE is the source of truth for all clusters managed by Kops. You'll need appropriate IAM permissions so that Kops can make API calls on your behalf. I won't cover that in this post, but you can follow the instructions here.
You'll also need to configure DNS. Kops supports a variety of configurations. Each has its own setup instructions. AWS Route53 with an existing HostedZone is the easiest. We'll assume that there is an existing AWS Route53 HostedZone for `slashdeploy.com` in these examples.
Kops clusters must be valid DNS names. Let's create the `demo.slashdeploy.com` cluster. Kops will also create the

DNS record for the Kubernetes API sever at `api.demo.slashdeploy.com`, and `bastion.demo.slashdeploy.com`. Keep in mind that DNS names may only be so long, so don't use base cluster names that are too long. Everything starts with `kops create`. You can pass options directly to the command or write a cluster spec file. We'll use the command line options for this exercise. Using a dedicate file is great for source control and other forms of configuration management. `kops create` accepts many options. We'll start with the simplest case by only supplying the required options.

```
$ kops create cluster \
      --yes \
      --zones=eu-west-1a,eu-west-1b,eu-west-1c \
      demo.slashdeploy.com
```

There are two required values. `--zones` states the GCP zones / AWS regions where to create the infrastructure. Here, eu-west-1a, eu-west-1b, eu-west-1c are specified. This instructs Kops to create infrastructure in each eu-west-1 availability zone. This is important because Kops aims to create high availability production clusters. Multiple availability zones make the cluster more reliable by protecting against failures in one availability zone.
You must also specify the cluster name. `--yes` confirms operations that normally prompt for confirmation. `kops create` adds a `kubectl` configuration entry for the new cluster so you're ready to use it right away. The command is async. It will trigger infrastructure creation, but will not block it completely. Luckily, Kops includes a command to validate a cluster. You can *rerun* this command until it succeeds.

```
$ kops validate demo.slashdeploy.com
```

When everything is complete, you should see something similar to the following:

```
$ kops validate cluster demo.slashdeploy.com
Validating cluster demo.slashdeploy.com
INSTANCE GROUPS
NAME                    ROLE    MACHINETYPE     MIN     MAX     SUBNETS
master-eu-west-1a       Master  m3.medium       1       1       eu-west-1a
master-eu-west-1b       Master  m3.medium       1       1       eu-west-1b
master-eu-west-1c       Master  m3.medium       1       1       eu-west-1c
nodes                   Node    t2.medium       2       2       eu-west-1a,eu-west-1b,eu-
west-1c
NODE STATUS
```

```
NAME                                            ROLE    READY
ip-172-20-120-240.eu-west-1.compute.internal    master  True
ip-172-20-50-132.eu-west-1.compute.internal     master  True
ip-172-20-66-106.eu-west-1.compute.internal     master  True
ip-172-20-75-89.eu-west-1.compute.internal      node    True
Your cluster demo.slashdeploy.com is ready
```

Now, you're ready to run any `kubectl` command such as `kubectl get pods -n kube-system`. The cluster is a bit strange because it has three masters and only a single worker. Let's update the node instance group.

Modifying Cluster Infrastructure

Remember that `kops` behaves like `kubectl`. This means that you can `kops edit` to edit the configuration files in your editor. The next step is to run `kops update`. This applies configuration changes, but does not modify running infrastructure. `kops rolling-update` manages updating or recreating infrastructure.
This process applies to all sorts of configuration changes. First `edit`, then `update`, and finally `rolling-update`. Let's take this for a spin by editing the node instance group to increase the number of worker nodes.

```
$ kops edit instancegroup nodes
```

That will open a YAML file in your editor. You'll see something similar to the following:

```
apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
  creationTimestamp: "2017-04-05T15:33:52Z"
  labels:
    kops.k8s.io/cluster: demo.slashdeploy.com
  name: nodes
spec:
  image: kope.io/k8s-1.5-debian-jessie-amd64-hvm-ebs-2017-01-09
  machineType: t2.medium
  maxSize: 1
  minSize: 1
  role: Node
  subnets:
  - eu-west-1a
  - eu-west-1b
  - eu-west-1c
```

All we need to do is replace `minSize` and `maxSize` with appropriate values. I'll set both values to `3` and save the file. This writes the updated file back to the `KOPS_STATE_STORE`. Now, we need to `update` the cluster. Again, we'll supply `--yes` to confirm the changes.

```
$ kops update cluster --yes
Using cluster from kubectl context: demo.slashdeploy.com
I0422 07:34:58.458492   26834 executor.go:91] Tasks: 0 done / 114 total; 35 can run
I0422 07:34:59.990241   26834 executor.go:91] Tasks: 35 done / 114 total; 26 can run
I0422 07:35:01.211466   26834 executor.go:91] Tasks: 61 done / 114 total; 36 can run
I0422 07:35:04.215344   26834 executor.go:91] Tasks: 97 done / 114 total; 10 can run
I0422 07:35:04.845173   26834 dnsname.go:107] AliasTarget for "api.demo.slashdeploy.com."
is "api-demo-1201911436.eu-west-1.elb.amazonaws.com."
I0422 07:35:05.045363   26834 executor.go:91] Tasks: 107 done / 114 total; 7 can run
I0422 07:35:05.438759   26834 executor.go:91] Tasks: 114 done / 114 total; 0 can run
I0422 07:35:05.438811   26834 dns.go:140] Pre-creating DNS records
I0422 07:35:06.707548   26834 update_cluster.go:204] Exporting kubecfg for cluster
Wrote config for demo.slashdeploy.com to "/home/ubuntu/.kube/config"
Kops has set your kubectl context to demo.slashdeploy.com
Cluster changes have been applied to the cloud.
Changes may require instances to restart: kops rolling-update cluster
```

Finally, apply the `rolling-update`.

```
$ kops rolling-update cluster --yes
Using cluster from kubectl context: demo.slashdeploy.com
NAME                    STATUS  NEEDUPDATE     READY   MIN     MAX     NODES
bastions                Ready   0              1       1       1       0
master-eu-west-1a       Ready   0              1       1       1       1
master-eu-west-1b       Ready   0              1       1       1       1
master-eu-west-1c       Ready   0              1       1       1       1
```

```
nodes                   Ready   0               3       3       3       3
No rolling-update required
```

That’s a bit strange. Kops says that there is no rolling-update required. This is true because we only changed the minimum and maximum number of instances in the `nodes` auto scaling group. This does not require any changes to the existing infrastructure. AWS simply triggers creation of two instances.

Let’s make another change that requires changing infrastructure. Imagine that the existing t2`.medium` instances are not cutting it. We need to scale up to meet workload requirements. To do that, we need to change the instance type. The same `edit`, `update`, and `rolling-update` process applies here. Let’s upgrade to `m4.large`. Repeat the exercise and replace t2`.medium` with `m4.large` then apply the `rolling-update`. Now, Kops kills each node in order to trigger creation of an up-to-date node.

```
$ kops rolling-update cluster --yes
Using cluster from kubectl context: demo.slashdeploy.com
NAME                    STATUS          NEEDUPDATE      READY   MIN     MAX     NODES
bastions                Ready           0               1       1       1       0
master-eu-west-1a       Ready           0               1       1       1       1
master-eu-west-1b       Ready           0               1       1       1       1
master-eu-west-1c       Ready           0               1       1       1       1
nodes                   NeedsUpdate     3               0       3       3       3
I0422 07:42:31.615734     659 rollingupdate_cluster.go:281] Stopping instance "i-
038cbac0aeaca24d4" in AWS ASG "nodes.demo.slashdeploy.com"
I0422 07:44:31.920426     659 rollingupdate_cluster.go:281] Stopping instance "i-
046fe9866a3b51fe6" in AWS ASG "nodes.demo.slashdeploy.com"
I0422 07:46:33.539412     659 rollingupdate_cluster.go:281] Stopping instance "i-
07f924becaa46d2ab" in AWS ASG "nodes.demo.slashdeploy.com"
```

Caution though! Current versions (<= 1.6) do not yet perform a real rolling update It just shuts down machines in sequence with a delay; there will be downtime Issue #37 We have implemented a new feature that does drain and validate nodes. This feature is experimental, and you can use the new feature by setting `export KOPS_FEATURE_FLAGS="+DrainAndValidateRollingUpdate"`. This should be fixed in a future release.