written by Garry Wilson on 27 October 2016 in devops, kubernetes

getting storage where you want it

To make our Kubernetes cluster as reliable as possible, we want our CoreOS instances running it to be both very simple but very smart. The goal is a single launch configuration being used to create three Kubernetes controller nodes, across the three availability zones.

There are a few things making that troublesome: EC2 instances come and go, we shouldn't assign static IPs, and endpoints and DNS may differ depending on which zone an instance starts in.

In my previous post on how to get etcd2 to cope well, I covered how to have our CoreOS Kubernetes controllers adapt their configuration to handle the first two of those problems; dynamic IPs and controllers starting up or being terminated.

smarter storage

I've talked a bit about why Elastic File System storage is exciting, especially as we move to the Kubernetes world. When it's set up, Amazon provide you DNS endpoints to access EFS — one for each of the zones in your region.

This is the third problem: when you want to have your Kubernetes instances in many zones, but also to maintain consistent, zone-agnostic configuration to maintain those instances, to simplify future changes.

Fortunately, with a bit of playing around with CoreOS's NFS settings, it's possible to do just that. Here's the code:

The file being written, `/etc/conf.d/nfs`, and the rpc-statd service need to be in place before we start mounting EFS. We then define the mount itself - this has three main components:

1. `AZ_ZONE` - which is where you'd normally put the zone name, but which will be replaced automatically later,
2. `[EFS-ID]` - which should be replaced with your specific EFS ID, as generated by AWS (you may also have to change eu-west-1 if deploying in a different region),
3. and `Where=/mnt/efs`; defining where on the CoreOS instance we'd like to mount the EFS volume.

getting into the zone

The final block of the cloud-config shown, efs-az.service, is how we update the EFS mount to bind within the instance's zone. We replace the AZ_ZONE string defined in the previous block with the value obtained by querying AWS's zone endpoint, *http://169.254.169.254/latest/meta-data/placement/availability-zone*

.

Once updated, we reload the config and restart the service. If everything is correct, the EFS volume should then be available at `/mnt/efs`, or whatever was specified as `Where=`.

Though simple, this block of code allows the instances to be smart enough to make the correct DNS endpoint for themselves, and allows us to have just one cloud-config, launch configuration and autoscaling group to maintain and replace should we need to. Less [config] is definitely more [win].

If you enjoyed the read, drop us a comment below or share the article, follow us on Twitter or subscribe to our #MetaBeers newsletter. Before you go, grab a PDF of the article, and let us know if it's time we worked together.