Creating AWS Volumes

In this lesson, we will explore the options we can opt to persist the state of an application and create AWS Volumes.

WE'LL COVER THE FOLLOWING

Exploring the Options to Persist State

If we want to persist state that will survive even server failures, we have two options we can choose.

Local Storage

We could, for example, store data locally and replicate it to multiple servers. That way, a container could use local storage knowing that the files are available on all the servers. Such a setup would be too complicated if we'd like to implement the process ourselves. Truth be told, we could use one of the volume drivers for that. However, we'll opt for a more commonly used method to persist the state across failures. We'll use external storage.

External Storage

Since we are running our cluster in AWS, we can choose between S3, Elastic File System (EFS), and Elastic Block Store.

S3 is meant to be accessed through its API and is not suitable as a local disk replacement. That leaves us with EFS and EBS.

Elastic File System (EFS)

Elastic File System (EFS), has a distinct advantage that it can be mounted to multiple EC2 instances spread across multiple availability zones. It is the closest we can get to fault-tolerant storage. Even if a whole zone (datacenter) fails, we'll still be able to use EFS in the rest of the zones used by our cluster. However, that comes at a cost. EFS introduces a performance penalty. It is, after all, a network file system (NFS), and that entails higher latency.

Elastic Block Store (EBS)

Elastic Block Store (EBS) is the fastest storage we can use in AWS. Its data access latency is very low thus making it the best choice when performance is the primary concern. The downside is availability. It doesn't work in multiple availability zones. Failure of one will mean downtime, at least until the zone is restored to its operational state.

Our Choice

We'll choose EBS for our storage needs. Jenkins depends heavily on IO, and we need data access to be as fast as possible. However, there is another reason for such a choice. EBS is fully supported by Kubernetes. EFS will come but, at the time of this writing, it is still in the experimental stage. As a bonus advantage, EBS is much cheaper than EFS.

Given the requirements and what Kubernetes offers, the choice is obvious. We'll use EBS, even though we might run into trouble if the availability zone where our Jenkins will run goes down. In such a case, we'd need to migrate EBS volume to a healthy zone. There's no such thing as a perfect solution.

We are jumping ahead of ourselves. We'll leave Kubernetes aside for a while and concentrate on creating an EBS volume.

Creating an EBS Volume

Each EBS volume is tied to an availability zone. Unlike EFS, Elastic Block Store cannot span multiple zones. So, the first thing we need to do is to find out which are the zones worker nodes are running in.

Looking into the Description

We can get the above information by describing the EC2 instances belonging to the security group `nodes.devops23.k8s.local`.

```
aws ec2 describe-instances
```

The **output**, limited to the relevant parts, is as follows.

```
{
  "Reservations": [
    {
      "Instances": [
        {
          ...
          "SecurityGroups": [
            {
              "GroupName": "nodes.devops23.k8s.local",
              "GroupId": "sg-33fd8c58"
            }
          ],
          ...
          "Placement": {
            "Tenancy": "default",
            "GroupName": "",
            "AvailabilityZone": "us-east-2a"
          },
          ...
```

We can see that the information is inside the `Reservations.Instances` array. To get the zone, we need to filter the output by the `SecurityGroups.GroupName` field. Zone name is located in the `Placement.AvailabilityZone` field.

Retrieving the Availability Zones

The command that does the filtering and retrieves the availability zones of the worker nodes is as follows.

```
aws ec2 describe-instances \
    | jq -r \
    ".Reservations[].Instances[] \
```

```
    | select(.SecurityGroups[]\
    .GroupName==\"nodes.$NAME\")\
    .Placement.AvailabilityZone"
```

The **output** is as follows.

```
us-east-2a
us-east-2c
```

We can see that the two worker nodes are located in the zones `us-east-2a` and `us-east-2c`.

Setting Up the Environment Varibales

The commands that retrieve the zones of the two worker nodes and store them in environment variables is as follows.

```
aws ec2 describe-instances \
    | jq -r \
    ".Reservations[].Instances[] \
    | select(.SecurityGroups[]\
    .GroupName==\"nodes.$NAME\")\
    .Placement.AvailabilityZone" \
    | tee zones

AZ_1=$(cat zones | head -n 1)

AZ_2=$(cat zones | tail -n 1)
```

We retrieved the zones and stored the output into the `zones` file. Further on, we retrieved the first row with the `head` command and stored it in the environment variable `AZ_1`. Similarly, we stored the last (the second) row in the variable `AZ_2`.

Creation of Volumes

Now we have all the information we need to create a few volumes.

> ℹ The command that follows requires a relatively newer version of `aws`. If it fails, please update your AWS CLI binary to the latest version.

```
VOLUME_ID_1=$(aws ec2 create-volume \
    --availability-zone $AZ_1 \
    --size 10 \
```

```
    --volume-type gp2 \
    --tag-specifications "ResourceType=volume,Tags=[{Key=KubernetesCluster,Value=$NAME}]" \
    | jq -r '.VolumeId')

VOLUME_ID_2=$(aws ec2 create-volume \
    --availability-zone $AZ_1 \
    --size 10 \
    --volume-type gp2 \
    --tag-specifications "ResourceType=volume,Tags=[{Key=KubernetesCluster,Value=$NAME}]" \
    | jq -r '.VolumeId')

VOLUME_ID_3=$(aws ec2 create-volume \
    --availability-zone $AZ_2 \
    --size 10 \
    --volume-type gp2 \
    --tag-specifications "ResourceType=volume,Tags=[{Key=KubernetesCluster,Value=$NAME}]" \
    | jq -r '.VolumeId')
```

We executed `aws ec2 create-volume` command three times. As a result, we created three EBS volumes. Two of them are in one zone, while the third is in another. They all have `10` GB of space. We chose `gp2` as the type of the volumes. The other types either require bigger sizes or are more expensive. When in doubt, `gp2` is usually the best choice for EBS volumes.

We also defined a tag that will help us distinguish the volumes dedicated to this cluster from those we might have in our AWS account for other purposes.

Finally, jq filtered the output so that only the volume ID is retrieved. The results are stored in the environment variables `VOLUME_ID_1`, `VOLUME_ID_2`, and `VOLUME_ID_3`.

Verification

Let's take a quick look at one of the IDs we stored as an environment variable.

```
echo $VOLUME_ID_1
```

The **output** is as follows.

```
vol-092b8980b1964574a
```

Finally, to be on the safe side, we'll list the volume that matches the ID and thus confirm, without doubt, that the EBS was indeed created.

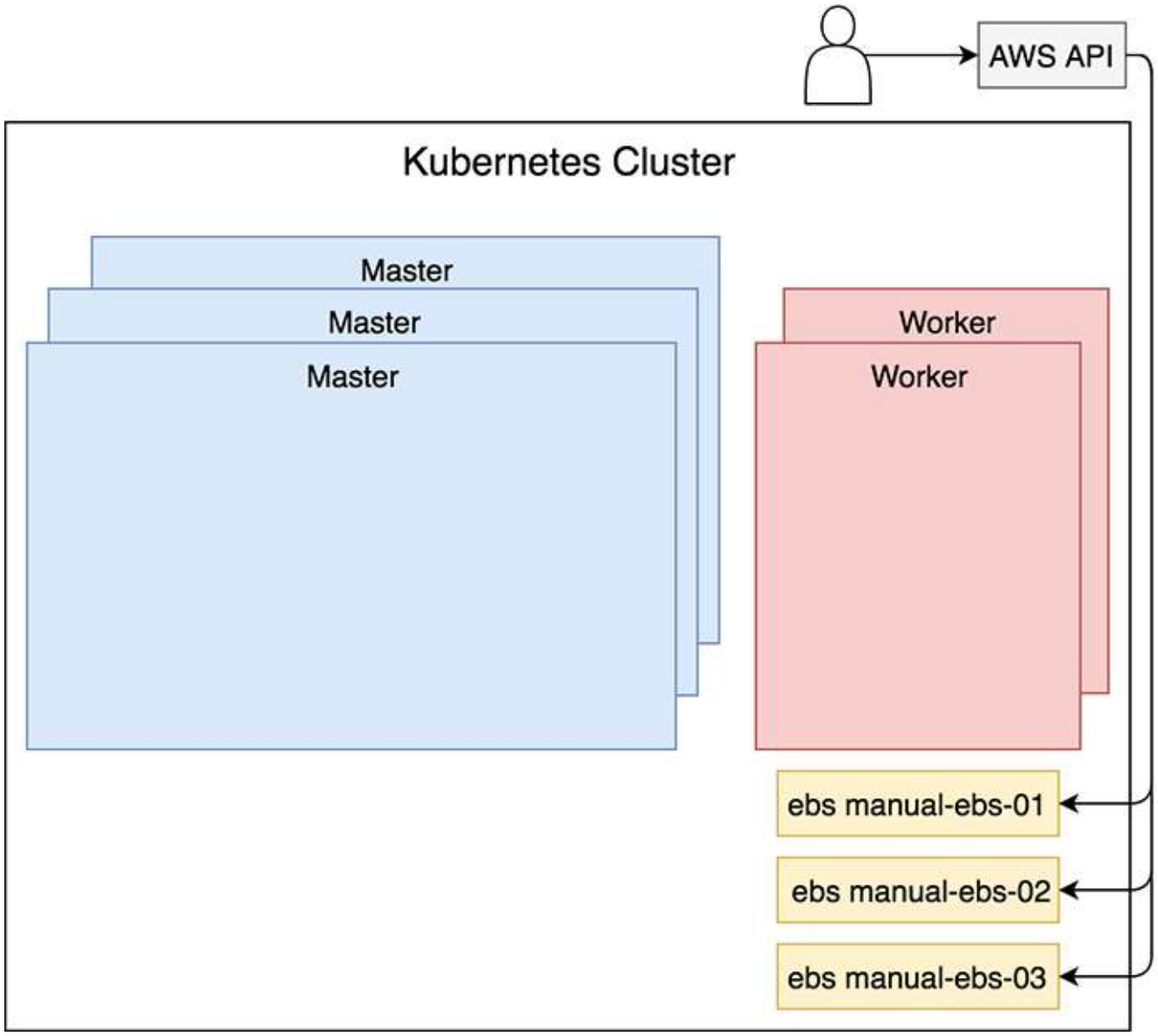
```
aws ec2 describe-volumes \
```

```
    --volume-ids $VOLUME_ID_1
```

The **output** is as follows.

```
{
  "Volumes": [
    {
      "AvailabilityZone": "us-east-2c",
      "Attachments": [],
      "Tags": [
        {
          "Value": "devops23.k8s.local",
          "Key": "KubernetesCluster"
        }
      ],
      "Encrypted": false,
      "VolumeType": "gp2",
      "VolumeId": "vol-092b8980b1964574a",
      "State": "available",
      "Iops": 100,
      "SnapshotId": "",
      "CreateTime": "2018-03-14T21:47:13.242Z",
      "Size": 10
    }
  ]
}
```

Now that the EBS volumes are indeed `available` and in the same zones as the worker nodes, we can proceed and create Kubernetes persistent volumes.



EBS volumes created in the same zones as the worker nodes

In the next lesson, we will create Kubernetes persistent Volumes.