Preparing for the Cluster Setup: IAM Group and User

In this lesson, we will create an Identity and Access Management group and a user.

WE'LL COVER THE FOLLOWING

In this lesson, we'll create a few Identity and Access Management (IAM) resources. Even though we could create a cluster with the user you used to register to AWS, it is a good practice to create a separate account that contains only the privileges we'll need for the exercises that follow.

Creating the IAM Group

First, we'll create an IAM group called `kops`.

```
aws iam create-group \
    --group-name kops
```

The **output** is as follows.

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2018-02-21T12:58:47.853Z",
        "GroupId": "AGPAIF2Y6HJF7YFYQBQK2",
        "Arn": "arn:aws:iam::036548781187:group/kops",
        "GroupName": "kops"
    }
}
```

We don't care much for any of the information from the output except that it does not contain an error message thus confirming that the group was created successfully.

Next, we'll assign a few policies to the group thus providing the future users of the group with sufficient permissions to create the objects we'll need.

Since our cluster will consist of EC2 instances, the group will need to have the permissions to create and manage them. We'll need a place to store the state of the cluster so we'll need access to S3. Furthermore, we need to add VPCs to the mix so that our cluster is isolated from prying eyes. Finally, we'll need to be able to create additional IAMs.

In AWS, user permissions are granted by creating policies. We'll need `AmazonEC2FullAccess`, `AmazonS3FullAccess`, `AmazonVPCFullAccess`, and `IAMFullAccess`.

The commands that attach the required policies to the `kops` group are as follows.

```
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/AmazonS3FullAccess \
    --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/IAMFullAccess \
    --group-name kops
```

Creating a User

Now that we have a group with the sufficient permissions, we should create a user as well.

```
aws iam create-user \
    --user-name kops
```

The **output** is as follows.

```
{
    "User": {
        "UserName": "kops",

        "Path": "/",
        "CreateDate": "2018-02-21T12:59:28.836Z",
        "UserId": "AIDAJ22UOS7JVYQIAVMWA",
        "Arn": "arn:aws:iam::036548781187:user/kops"
    }
}
```

Just as when we created the group, the contents of the output are not important, except as a confirmation that the command was executed successfully.

The user we created does not yet belong to the `kops` group. We'll fix that next.

```
aws iam add-user-to-group \
    --user-name kops \
    --group-name kops
```

Finally, we'll need access keys for the newly created user. Without them, we would not be able to act on its behalf.

```
aws iam create-access-key \
    --user-name kops >kops-creds
```

We created access keys and stored the output in the `kops-creds` file. Let's take a quick look at its content.

```
cat kops-creds
```

The **output** is as follows.

```
{
    "AccessKey": {
        "UserName": "kops",
        "Status": "Active",
        "CreateDate": "2018-02-21T13:00:24.733Z",
        "SecretAccessKey": "...",
        "AccessKeyId": "..."
    }
}
```

> 📝 Please note that we removed the values of the keys as it is not

> recommended to show them publicly.

Installing jq to Parse the Credentials

We need the `SecretAccessKey` and `AccessKeyId` entries. So, the next step is to parse the content of the `kops-creds` file and store those two values as the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.

In the spirit of full automation, we'll use jq to parse the contents of the `kops-creds` file. Please download and install the distribution suited for your OS.

📝 A note to Windows users

> Using Chocolatey, install `jq` from an Administrator Command Prompt via `choco install jq`.

```
export AWS_ACCESS_KEY_ID=$(\
    cat kops-creds | jq -r \
    '.AccessKey.AccessKeyId')

export AWS_SECRET_ACCESS_KEY=$(
    cat kops-creds | jq -r \
    '.AccessKey.SecretAccessKey')
```

We used `cat` to output contents of the file and combined it with `jq` to filter the input so that only the field we need is retrieved.

From now on, all the AWS CLI commands will not be executed by the administrative user you used to register to AWS, but as `kops`.

> It is imperative that the `kops-creds` file is secured and not accessible to anyone but people you trust. The best method to secure it varies from one organization to another. No matter what you do, do not write it on a post-it and stick it to your monitor. Storing it in one of your GitHub repositories is even worse.

In the next lesson, we will continue to prepare for the setup of our production-ready cluster by setting up the availability zones and SSH keys.