**Introduction**

Elastic Kubernetes Service (EKS) is a fully managed Kubernetes service from AWS. In this lab, you will work with the AWS command line interface and console, using command line utilities like eksctl and kubectl to launch an EKS cluster, provision a Kubernetes deployment and pod running instances of nginx, and create a LoadBalancer service to expose your application over the internet.

**Solution**

Log in to the live AWS environment using the credentials. Make sure you're in the N. Virginia (us-east-1) region throughout the lab.

**Create an IAM User with Admin Permissions**

1. Navigate to **IAM** > **Users**.
2. Click **Add user**.
3. Set the following values:
   * *User name*: **k8-admin**
   * *Access type*: **Programmatic access**
4. Click **Next: Permissions**.
5. Select **Attach existing policies directly**.
6. Select **AdministratorAccess**.
7. Click **Next: Tags** > **Next: Review**.
8. Click **Create user**.
9. Copy the access key ID and secret access key, and paste them into a text file, as we'll need them in the next step.

**Launch an EC2 Instance and Configure the Command Line Tools**

1. Navigate to **EC2** > **Instances**.
2. Click **Launch Instance**.
3. On the AMI page, select the Amazon Linux 2 AMI.
4. Leave *t2.micro* selected, and click **Next: Configure Instance Details**.
5. On the *Configure Instance Details* page:
   * *Network*: Leave default
   * *Subnet*: Leave default
   * *Auto-assign Public IP*: **Enable**
6. Click **Next: Add Storage** > **Next: Add Tags** > **Next: Configure Security Group**.
7. Click **Review and Launch**, and then **Launch**.
8. In the key pair dialog, select **Create a new key pair**.
9. Give it a *Key pair name* of "mynvkp".
10. Click **Download Key Pair**, and then **Launch Instances**.
11. Click **View Instances**, and give it a few minutes to enter the *running* state.
12. Once the instance is fully created, check the checkbox next to it and click **Connect** at the top of the window.
13. In the *Connect to your instance* dialog, select **EC2 Instance Connect (browser-based SSH connection)**.
14. Click **Connect**.
15. In the command line window, check the AWS CLI version:

aws --version

It should be an older version.

1. Download v2:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

1. Unzip the file:

unzip awscliv2.zip

1. See where the current AWS CLI is installed:

which aws

It should be /usr/bin/aws.

1. Update it:

sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update

1. Check the version of AWS CLI:

aws --version

It should now be updated.

1. Configure the CLI:

aws configure

1. For AWS Access Key ID, paste in the access key ID you copied earlier.
2. For AWS Secret Access Key, paste in the secret access key you copied earlier.
3. For Default region name, enter us-east-1.
4. For Default output format, enter json.
5. Download kubectl:

curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.8/2020-04-16/bin/linux/amd64/kubectl

1. Apply execute permissions to the binary:

chmod +x ./kubectl

1. Copy the binary to a directory in your path:

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin

1. Ensure kubectl is installed:

kubectl version --short --client

1. Download eksctl:

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

1. Move the extracted binary to /usr/bin:

sudo mv /tmp/eksctl /usr/bin

1. Get the version of eksctl:

eksctl version

1. See the options with eksctl:

eksctl help

**Provision an EKS Cluster**

1. Provision an EKS cluster with three worker nodes in us-east-1:

eksctl create cluster --name dev --version 1.16 --region us-east-1 --nodegroup-name standard-workers --node-type t3.micro --nodes 3 --nodes-min 1 --nodes-max 4 --managed

It will take 10–15 minutes since it's provisioning the control plane and worker nodes, attaching the worker nodes to the control plane, and creating the VPC, security group, and Auto Scaling group.

1. In the AWS Management Console, navigate to CloudFormation , stacks and take a look at what’s going on there.
2. Select the eksctl-dev-cluster stack (this is our control plane).
3. Click **Events**, so you can see all the resources that are being created.
4. We should then see another new stack being created — this one is our node group.
5. Once both stacks are complete, navigate to **Elastic Kubernetes Service** > **Clusters**.
6. Click the listed cluster.
7. Click the **Compute** tab, and then click the listed node group. There, we'll see the Kubernetes version, instance type, status, etc.
8. Click **dev** in the breadcrumb navigation link at the top of the screen.
9. Click the **Networking** tab, where we'll see the VPC, subnets, etc.
10. Click the **Logging** tab, where we'll see the control plane logging info.
    * The control plane is abstracted — we can only interact with it using the command line utilities or the console. It’s not an EC2 instance we can log into and start running Linux commands on.
11. Navigate to **EC2** > **Instances**, where you should see the instances have been launched.
12. Close out of the existing CLI window, if you still have it open.
13. Select the original t2.micro instance, and click **Connect** at the top of the window.
14. In the *Connect to your instance* dialog, select **EC2 Instance Connect (browser-based SSH connection)**.
15. Click **Connect**.
16. In the CLI, check the cluster:

eksctl get cluster

1. Enable it to connect to our cluster:

aws eks update-kubeconfig --name dev --region us-east-1

**Create a Deployment on Your EKS Cluster**

**Test the High Availability Features of Your EKS Cluster**

1. In the AWS console, on the EC2 instances page, select the three t3.micro instances.
2. Click **Actions** > **Instance State** > **Stop**.
3. In the dialog, click **Yes, Stop**.
4. After a few minutes, we should see EKS launching new instances to keep our service running.
5. In the CLI, check the status of our nodes:

kubectl get node

All the nodes should be down (i.e., display a NotReady status).

1. Check the pods:

kubectl get pod

We'll see a few different statuses — Terminating, Running, and Pending — because, as the instances shut down, EKS is trying to restart the pods.

1. Check the nodes again:

kubectl get node

We should see a new node, which we can identify by its age.

1. Wait a few minutes, and then check the nodes again:

kubectl get node

We should have one in a Ready state.

1. Check the pods again:

kubectl get pod

We should see a couple pods are now running as well.

1. Check the service status:

kubectl get service

1. Copy the external IP listed in the output.
2. Access the application using the load balancer, replacing <LOAD\_BALANCER\_EXTERNAL\_IP> with the IP you just copied:

curl "<LOAD\_BALANCER\_EXTERNAL\_IP>"

We should see the Nginx web page HTML again. (If you don't, wait a few more minutes.)

1. In a new browser tab, navigate to the same IP, where we should again see the Nginx web page.
2. **In the CLI, delete everything:**

eksctl delete cluster dev