Lab Report - Lab 11

Name: Tyler Burleson Date: 4/8/2024

Summary

The purpose of this lab was to practice configuring the AWS-CLI on our personal machines and create EC2 instances from the CLI. First, we started by downloading the CLI installation packages from the AWS website. Once installed we opened PowerShell and ran “aws –version” to confirm the installation. Once the installation was confirmed we navigated to the academy page and selected “AWS Details.” This then allowed us to copy our access key, secret access key, and our session token. Once we had all the details, we saved them into Notepad++ to be used for later. Finally, we downloaded our “labsuser.pem.”

Now that we had all our prerequisite information, we were ready to configure our device. We started by changing into our user directory instead of windows32. Next, we ran “aws configure” and followed the steps and pasted our access key when prompted. Next we decided to use our default region as “us-east-1” and set our default output format as JSON. Once this was finished we had to open the “.aws/credentials” and pasted our “aws_session_token.” Once this was all complete our device was successfully configured to have access to our AWS account through the CLI.

To confirm our device was correctly configured we ran the command “aws ec2 describe-regions –output text.” This command would list all the available AWS regions available to us. If this command didn’t show this output, it would instead throw an error that our device doesn’t have access or was missing certain information. Now that our device is confirmed to be configured, we can start attempting to run an EC2 instance from our machine.

To start creating instances we need some more required info to save in our Notepad++. We started by navigating to the EC2 Security Groups tab. Inside here we selected our default security group and copied the SG ID into our document. Next, we clicked inbound rules and added a rule to allow IPv4 traffic from anywhere and SSH traffic from anywhere. Once these rules were saved, we then migrated to the “Launch Instances” tab and documented the current image-id of the AWS Linux 2 AMI. Finally, we saved the subnet ID we would be using. By using the following information: “VPC,” “AMI,” “Subnet,” and “Security Group” we are ready to run the creation command “aws ec2 run-instances –image-id ami-0c02fb55956c7d316 –security-group-ids sg-0be20c7be8d9aebee –instance-type t2.micro –key-name vockey –subnet subnet-03ba78155531aa5eb” If all the information in this command is correct our output would show a page of JSON. Next, we took the public IP address from the output and attempted to SSH into the machine using our PEM file. Once we confirmed to connect to the machine, we were inside our newly created instance.

Now that we have seen how to manually create an instance, we needed a way to automate this process by using a PowerShell script. This script would start by setting a variable using the user input and ask for the server’s name. Next, the script made a new variable called “tagSpecs” which created a new Key-Value to give the instance a name. Finally, we ran the previous creation command and appended the “–tag-specifications $tagSpecs” to the end. Once we saved the ps1 file we tested it and were able to see our new instance on AWS. Once this was complete, we were finished with the lab and we were able to close our AWS Learner Lab.

Screen Shots

```
PS C:\users\saint\desktop> aws ec2 describe-regions --output text
REGIONS ec2.ap-south-1.amazonaws.com    opt-in-not-required     ap-south-1
REGIONS ec2.eu-north-1.amazonaws.com    opt-in-not-required     eu-north-1
REGIONS ec2.eu-west-3.amazonaws.com     opt-in-not-required     eu-west-3
REGIONS ec2.eu-west-2.amazonaws.com     opt-in-not-required     eu-west-2
REGIONS ec2.eu-west-1.amazonaws.com     opt-in-not-required     eu-west-1
REGIONS ec2.ap-northeast-3.amazonaws.com        opt-in-not-required     ap-northeast-3
REGIONS ec2.ap-northeast-2.amazonaws.com        opt-in-not-required     ap-northeast-2
REGIONS ec2.ap-northeast-1.amazonaws.com        opt-in-not-required     ap-northeast-1
REGIONS ec2.ca-central-1.amazonaws.com  opt-in-not-required     ca-central-1
REGIONS ec2.sa-east-1.amazonaws.com     opt-in-not-required     sa-east-1
REGIONS ec2.ap-southeast-1.amazonaws.com        opt-in-not-required     ap-southeast-1
REGIONS ec2.ap-southeast-2.amazonaws.com        opt-in-not-required     ap-southeast-2
REGIONS ec2.eu-central-1.amazonaws.com  opt-in-not-required     eu-central-1
REGIONS ec2.us-east-1.amazonaws.com     opt-in-not-required     us-east-1
REGIONS ec2.us-east-2.amazonaws.com     opt-in-not-required     us-east-2
REGIONS ec2.us-west-1.amazonaws.com     opt-in-not-required     us-west-1
REGIONS ec2.us-west-2.amazonaws.com     opt-in-not-required     us-west-2

PS C:\users\saint\desktop>
```

```
PS C:\users\saint\desktop> aws ec2 run-instances --image-id ami-0c101f26f147fa7fd --security-group-ids sg-046e91a7fca344
108 --instance-type t2.micro --key-name vockey --subnet subnet-083e998b8513ba068
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0c101f26f147fa7fd",
            "InstanceId": "i-06c720eabe598e50a",
            "InstanceType": "t2.micro",
            "KeyName": "vockey",
            "LaunchTime": "2024-04-01T19:42:31+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1e",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-30-1-110.ec2.internal",
            "PrivateIpAddress": "172.30.1.110",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
```

```
PS C:\users\saint\desktop> .\aws-launch-instances.ps1
Server name: FluxAWSLinux2
Launchin FluxAWSLinux2...
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0c101f26f147fa7fd",
            "InstanceId": "i-0bde2c774139e3830",
            "InstanceType": "t2.micro",
            "KeyName": "vockey",
            "LaunchTime": "2024-04-01T20:07:29+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1e",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-30-1-89.ec2.internal",
            "PrivateIpAddress": "172.30.1.89",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
```

```
PS C:\users\saint\desktop> aws ec2 stop-instances --instance-ids i-06c720eabe598e50a i-0bde2c774139e3830
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-0bde2c774139e3830",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        },
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-06c720eabe598e50a",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

PS C:\users\saint\desktop> aws ec2 terminate-instances --instance-ids i-0bde2c774139e3830 i-06c720eabe598e50a
{
    "TerminatingInstances": [
        {
            "CurrentState": {
```

Research Questions

1. What's the use of the API key? Why do we need one?

   API keys provide a mechanism for developers to securely access and control access to their APIs, monitor usage, and enforce usage policies such as rate limits and access permissions.

2. How might using the command line facilitate maintenance of your organization's infrastructure?

   It provides a powerful set of tools for automating tasks, managing resources, troubleshooting issues, and ensuring compliance with best practices.

3. In this activity, we wound up entering a lot of looooong commands, one at a time. How might we move forward to make using the CLI easier for executing common tasks (this question is kind of moot, since we did it already, this time)?

   Automate the process with a PowerShell script like we did in the lab.

4. How can we get help with a specific command in the AWS CLI (apart from Google, i.e., directly from the command line or PowerShell)?

   You can use the help command "aws help <aws_command>" or append "—help" after an AWS command.

5. How might we make the script above more extensible? Say, for example, we wanted it to be able to launch instances on different VPCs and/or subnets?

We can add prompts in the script to ask for the user to choose from a list of pre-configured VPC's and Subnets, in addition to any other desired settings.

6. All of the commands we used in this lab fall into the 'ec2' subset of AWS CLI commands. What are two other AWS services we can manage with the CLI?

In addition to "EC2" we can also manage S3, RDS, IAM, and several other services AWS offers.

7. Briefly enumerate and describe the steps we have to follow to launch an instance into an already existing VPC (this is the same, whether we're using the CLI or the GUI)

1. Navigate to launch instance either through the GUI or navigate to your CLI.
2. Select an AMI.
3. Choose an Instance type.
4. Select an existing VPC and choose the subnet within.
5. Add any desired storage.
6. Select or create a security group for the instance.

8. Let's say we've already created a VPC with a CIDR block of 10.1.0.0/16 and one subnet whose CIDR block is 10.1.1.0/24. What command would we use to create a second subnet that can host up to 512 IP addresses?

We would use "aws ec2 create-subnet --vpc-id <VPC-ID> --cidr-block 10.1.2.0/23"

9. What command (and options) would launch, say, five servers instead of only one?

We would run "aws ec2 run-instances --image-id <IMAGE-ID> --instance-type <INSTANCE-TYPE> --count <COUNT>"

10. Having launched my five servers, what command would I enter to find out the Public IP address of each?

We would use "aws ec2 describe-instances --instance-ids <INSTANCE-ID-1> <INSTANCE-ID-2> ... <INSTANCE-ID-N> --query 'Reservations[*].Instances[*].PublicIpAddress' --output text"