QATIPv3 AWS Lab2
Create an EC2 instance with Terraform

Contents

Lab Objectives

In this lab, you will:

- Review the Terraform registry documentation on the AWS Provider
- Build a simple main.tf configuration file based on example code
- Deploy an EC2 instance on AWS
- Test and modify the deployment
- Destroy the deployment

Teaching Points

This lab takes you through the writing of code to create an EC2 instance in AWS. A virtual machine cannot exist in isolation however, it must have a vpc with appropriate firewall rules for access. In this lab you will use the default VPC, in future labs you will create your own.

Before you begin

1. Ensure you have completed Lab0 before attempting this lab.

2. In the IDE terminal pane, enter the following commands…

```
cd ~/environment/awslabs/02
```

3. This shifts your current working directory to awslabs/labs/02. ***Ensure all commands are executed in this directory***

4. Close any open files and use the Explorer pane to navigate to and open the empty main.tf file in awslabs/02.

Solution

The solution to this lab can be found in awslabs/solutions/02. Try to use this only as a last resort if you are struggling to complete the step-by-step processes

Task 1: Review the documentation and create a configuration file to deploy an EC2 instance.

1. Review Terraform AWS Provider documentation: https://registry.terraform.io/providers/hashicorp/aws/4.53.0

2. Click `Use Provider`

3. Copy the code block into the empty main.tf file in the labs/02 folder. For convenience, the code is listed below:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.53.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

4. From within the 'Example Usage' section of the documentation, find the sections relating to '**Configure the AWS Provider**'. We see that information regarding the AWS region is needed, us-east-1 being used in the sample code.

5. We will be creating our EC2 instance in us-west-2. Modify your code to include the us-west-2 region. For convenience, the completed code for configuring the AWS provider is listed below:

```
provider "aws" {
  region = "us-west-2"
}
```

6. Within the Terraform AWS Provider documentation, navigate to **EC2 (Elastic Cloud Compute)** and select the resource type `**aws_instance**` Within the **Example Usage Section**, find the sample code for creating a basic aws instance using AMI lookup...

   https://registry.terraform.io/providers/hashicorp/aws/4.53.0/docs/resources/instance

7. Paste this code at the end of main.tf

8. Examine the code you have just added. There are two workflow blocks, **data "aws_ami" "ubuntu " {}** and **resource "aws_instance" "web" {}**. The first block obtains the latest ami for the image specified, in this case ubuntu. The second block creates an EC2 instance using this ami, as referenced by the line "**ami = data.aws_ami.ubuntu.id**"

Task 2: Run Terraform & Test

1. Save changes made to main.tf (Ctrl+s) and then run `**terraform init**` ensuring you are in the correct working directory

2. If there are any errors, correct them before continuing. (Use the solution guide if needed - but try first!)

3. Run `**terraform plan**` -- review what will be created

4. Run `**terraform apply**` typing `**yes**` when prompted. Review output in the CLI. Notice that a persistent terraform.tfstate file now exists in the root directory.

5. Take note of the id of the newly created **aws_instance.web** resource.

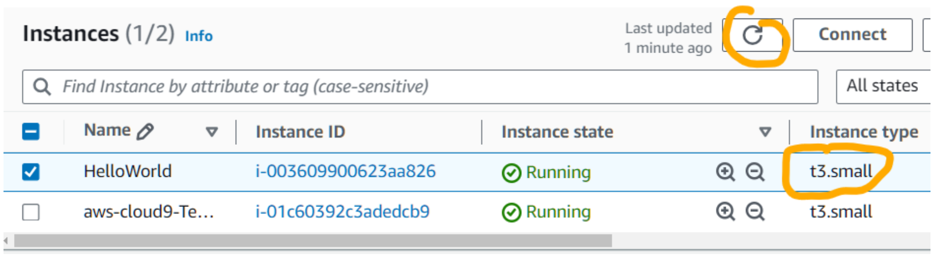
6. Switch to the Console and search for EC2 instances in the Oregon (us-west-2) region. Your newly created instance should be present along with the instance running your IDE. Note the Instance type as being t3.micro

7. Switch back to the IDE and within the resource block for the aws_instance, modify the instance size from **t3.micro** to **t3.small** to ascertain if this performs a modification or a deletion/re-creation. Save your changes.

8. Run `terraform plan` Review the output and note that this is an 'update in-place'

9. Run `terraform apply` typing `yes` when prompted. As the modification is made, note that the id of the instance is unchanged.

10. Switch back to the console and monitor the stopping of the existing instance before it is resized and restarted. (use the refresh button as needed)..


Instances (1/2) Info

Last updated 1 minute ago

	Name	Instance ID	Instance state	Instance type
☑	HelloWorld	i-003609900623aa826	Running	t3.small
☐	aws-cloud9-Te...	i-01c60392c3adedcb9	Running	t3.small

11. Switch back to the IDE and within the **aws_ami.ubuntu** data block, change the name of the Ubuntu version from its current value to `**hvm-ssd/ubuntu-jammy-20.04**`The modified block should now be...

```
filter {
  name   = "name"
values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]

}
```

12. Save your changes and then run `terraform plan` Review the output and note that this is a replacement. A new instance must be created using the new amazon machine image that has ubuntu-jammy pre-installed.

13. Run `terraform apply` typing `yes` when prompted.

14. Switch to the console and watch as the old instance is first destroyed, and a new one is created; its id being displayed once the creation completes. You

can also click into the running instance of “HelloWorld” to verify that the operating system has been updated.

Task 3: Destroy your deployment

1. Swich back to the IDE and run `**terraform destroy**` review the output and type `**yes**`

***** Congratulations, you have completed this lab *****