AWS and Terraform Cloud

Overview
This lab introduces Terraform Cloud and will demonstrate moving state storage and command execution off the local machine and into Terraform Cloud.

Setup

1. Ensure you have run Lab0

steps 2 to 9 removed

10. Switch to the Console Throughout this lab, use this screen to search for and visualize your code deployments.

11. Switch back to the browser tab containing the Cloud9 Lab IDE. Dismiss the Welcome tab if displayed and arrange the panes as shown below. This will referred to as the IDE throughout this lab.

12. Using the terminal window, create a new file tree for this lab..

```
cd ~
mkdir ./environment/lab/06a
cd ./environment/lab/06a
touch main.tf
touch variable.tf
```

13. Paste the following into main.tf and save the changes

```
terraform {
 required_providers {
  aws = {
   source  = "hashicorp/aws"
   version = "4.50"
  }
 }

 required_version = ">= 1.2.0"
}

provider "aws" {
 region = var.region
}

resource "aws_vpc" "lab_vpc" {
resource "aws_vpc" "labvpc" {
  cidr_block = "10.1.0.0/16"
}
```

```
resource "aws_internet_gateway" "labigw" {
  vpc_id = aws_vpc.labvpc.id

  tags = {
    Name = "labig"
  }
}

resource "aws_subnet" "publicsubnet" {
    vpc_id     = aws_vpc.labvpc.id
    cidr_block = "10.1.10.0/24"

    tags = {
    Name = "publicsubnet"
    }

}

data "aws_availability_zones" "azlist" {
  state = "available"
}

resource "aws_subnet" "privatesubnets" {
  count             = var.az_count
  cidr_block        = cidrsubnet(aws_vpc.labvpc.cidr_block, 8, count.index)
  availability_zone = data.aws_availability_zones.azlist.names[count.index]
  vpc_id            = aws_vpc.labvpc.id

  tags = {
    Name = "privatesubnet"
  }

}

resource "aws_route_table" "public_rt" {
    vpc_id = aws_vpc.labvpc.id

    route {
        cidr_block = "0.0.0.0/0"
        gateway_id=aws_internet_gateway.labigw.id
  }
  tags = {
  Name = "publicroute"
  }

}

resource "aws_eip" "static_ip" {
  vpc      = true
```

```
}

resource "aws_nat_gateway" "labnatgw" {
 allocation_id = aws_eip.static_ip.id
 subnet_id     = aws_subnet.publicsubnet.id

 tags = {
  Name = "lab_nat_gw"
 }

  Ensuring IGW is created
  Ensuring EIP is created
 depends_on = [aws_internet_gateway.labigw, aws_eip.static_ip]
}

resource "aws_route_table" "private_rt" {
   vpc_id = aws_vpc.labvpc.id

   route {
      cidr_block = "0.0.0.0/0"
      gateway_id=aws_nat_gateway.labnatgw.id
 }
 tags = {
 Name = "privateroute"
 }
}

resource "aws_route_table_association" "public_rta" {
   subnet_id      = aws_subnet.publicsubnet.id
   route_table_id = aws_route_table.public_rt.id
}

resource "aws_route_table_association" "private_rta" {
 count = var.az_count
 subnet_id      = "${element(aws_subnet.privatesubnets..id, count.index)}"
   route_table_id = aws_route_table.private_rt.id
}
```

14. Paste the following into variable.tf and save the changes

```
variable "region" {
 description = "location to build resources"
 type        = string
 default     = "useast2"
}

variable "az_count" {
   default     = 3
```

}

15. Run

terraform init
terraform plan

16. Verify there are no errors and then run

terraform apply

17. Enter yes to deploy

18. Switch the console to review the deployment.

This build deploys a VPC into the useast2 region. There is a single public subnet and a private subnet on each of 3 availabilityzones. There is a public routing table which uses an internet gateway and a private routing table which uses a NAT Gateway. The subnets have been associated with these routing tables.

Review the code and note any questions you may have for later discussion.

Qustion. What is the single point of failure here and how might it be addressed ? Discuss this with the instructor later.

19. Having verified its validity, run terraform destroy followed by yes to tear down the deployment.

The Challenge

The terraform state file in currently held on a single computer from which all terraform commands are executed. Whilst possibly suitable for small/test/poc, production environments should have a more robust and resilient Terraform configuration. This lab will introduce Terraform Cloud and the challenge is to move state storage and command execution off the local machine and into Terraform Cloud.

This is a fully guided lab, but feel free to eximine the Terraform Cloud enviroment you will be creating.

Exercise 1. Signing up for a free Terraform Cloud account.

To successfully complete this lab you will need an active email account. Consider creating a new burner email account that you can use purely for this lab.

1. Browse to https://app.terraform.io/

2. Complete the enrolment form to create a new free account..

3. A confirmation email will be sent to you...

4. Acknowledge the email sent...

5. A new browser session will open. You can safely close the first session tab. There are 3 setup workflow options. We will begin with Start from scratch...

6. As this is a new account you are required to create a new organization. Choose an organization name and complete the form...

7. Before creating our first workspace we must consider the credentials that Terraform Cloud will use as its authority to perform tasks in AWS on our behalf. We will create environmental variables with our AWS account keys. In the Projects and workspaces view, select Settings...

8. This will take us to the Organization menu. Select Variable Sets and then Create variable set...

9. Name the set **Cloud Credentials**, apply the set globally across all workspaces, and click to add your first variable...

10. Select Environment variable, and mark as sensitive Enter **AWS_ACCESS_KEY_ID** as the first key. On the left of the qwiklabs instructions screen, find the access_key generated for this lab. Copy this and paste it into the value field. Click on Add variable. Click on image below to enlarge if necesary...

11. Create another varaiable called **AWS_SECRET_ACCESS_KEY**, again copying the appropriate information from the qwiklabs instructions screen. Click on Add variable Once both variables have been created, click on Create variable set...

12. We can now create our first workspace. Select Workspaces...

13. Select Create a workspace...

14. Our workspace will be used to generate CLIdriven workflow...

15. Name the new workspace myfirstworkspace...

16. The workspace now exists but has no configuration files associated with it. To associate our local terraform files with Terraform Cloud we need to add the command block shown onscreen, prepopulated with your organization and workspace name, to our configuration...

17. Copy the block,paste it into the bottom of main.tf and save the changes..

18. Run terraform init ....

19. You need credentials on the IDE to identify yourself to Terraform Cloud. This is a token generated during the terraform login process. Run terraform login and enter yes...

20. Follow the link displayed. Resize the terminal window if you do not see the URL link..

21. A new browser window opens to TFC and 'Create API token' appears. Click on Create API token...

22. A new token is generated. Copy it and close the browser tab ....

23. In the IDE click into the terminal session, rightclick and choose paste. The token values will not appear on screen. Press Enter...

24. If successful, the 'Welcome to Terraform Cloud' banner will show...

25. Run terraform init..

26. You will be notified that the Terraform Cloud initialization completes successfully...

27. Run terraform apply...

28. Switch to TFC page. Planning should begin...

29. Switch back to IDE and wait for prompt to appear. DO NOT TYPE 'yes' yet...

30. Switch to TFC page. Planning is now complete...

31. Switch back to IDE and enter yes..

32. Switch back to TFC page. The plan is being applied..

33. Click on See details to watch deployment progressing...

34. Switch back to IDE and wait for deployment to complete...

35. Switch back to TFC. Select Overview to see a summary of the applied job and the resources created...

Question. The IDE shows 14 resources added whilst TFC shows 15. Can you identify the extra resource ?

36. In the IDE enter terraform destroy Do not confirm yet.

37. Switch to TFC and note the planning of the destroy is in progress.

38. Wait until the destroy run shows as Planned Click on See details

39. Scroll down and click on Confirm and Apply

40. Click on Confirm Plan

41. Switch back to IDE and, if timely, you will see a message indicating the apply has been

approved on TFC. The destroy will then progress.

Exercise 2. Creating Terraform Cloud variables

1. Switch back to TLC and create a Terraform variable region with a value of uswest2 without quotes.

2. Create a second Terraform variable az_count with a value of 2

3. Switch back to the IDE and run terraform apply and yes

4. Observe in the IDE and TLC as the deployment now occurs in uswest2 over 2 availability zones.

5. Tear down the deployment using teraform destroy

Congratulations, you have completed this lab