**Migrate an Existing Terraform Project into Terraform Cloud**

**Introduction**

In this hands-on lab, we will take an existing Terraform configuration and convert it to run the code using Terraform Cloud. We will use the CLI-driven method to deploy our resources to AWS with Terraform Cloud doing the bulk of the work.

**Solution**

1. Open a terminal session, and log in to the lab server using the credentials provided:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

1. Log in to your GitHub account in the browser.
2. Log in to your Terraform Cloud account in the browser: https://app.terraform.io/
3. In an incognito or private browser window, log in to the live AWS environment using the credentials provided. Make sure you're in the N. Virginia (us-east-1) region throughout the lab.

**Prepare the Environment**

1. In the terminal, clone the lab repo:

git clone https://github.com/ACloudGuru-Resources/content-introduction-terraform-cloud-lab.git

1. Make sure you see the content-introduction-terraform-cloud-lab directory:

ls

1. Change directory:

cd content-introduction-terraform-cloud-lab

1. List the files in the directory:

ls

You should see main.tf and README.md.

1. Open main.tf:

vim main.tf

1. In the Terraform Cloud console, click your avatar and then **User Settings**.
2. In the left-hand menu, click **Organizations**.
3. Click **Create new organization**.
4. For **Organization name**, enter *terraform-guru* (followed by some characters to make it globally unique).
5. Click **Create organization**.
6. On the **Create a new Workspace** screen, select **CLI-driven workflow**.
7. For **Workspace Name**, enter *terraform-cloud-guru*.
8. Click **Create workspace**.
9. In the left-hand menu, click **Workspaces**.
10. In the left-hand menu, click **Settings**.
11. In the left-hand menu, click **Variable sets**.
12. Click **Create variable set**, and set the following values:
    * **Name:** Enter *AWS Credentials*.
    * **Workspaces:** Select **Apply to specific workspaces**.
    * **Search for workspaces:** Search for and select **terraform-cloud-guru**.
13. Click **Add variable**, and set the following values:
    * **Select variable category:** Select *Environment variable*.
    * **Key:** Enter *AWS\_ACCESS\_KEY\_ID*.
    * **Value:**
      1. In the AWS Management Console, navigate to **IAM** > **Users**.
      2. Click **cloud\_user**.
      3. Click **Security credentials**.
      4. Click **Create access key**.
      5. Once it's created, copy the access key ID and secret access key, and paste them into a text file.
      6. Back on the variable creation page in Terraform Cloud, paste the access key ID into the **Value** field.
      7. Check **Sensitive**.
14. Click **Add variable**.
15. Click **Add variable** again, and set the following values:
    * **Select variable category:** Select *Environment variable*.
    * **Key:** Enter *AWS\_SECRET\_ACCESS\_KEY*.
    * **Value:** Paste the secret access key ID you copied.
    * **Sensitive:** Check the box.
16. Click **Add variable**.
17. Click **Create variable set**.
18. In the AWS Management Console, navigate to **VPC** > **Subnets**.
19. Copy the listed subnet ID, and paste it into a text file.
20. Back in the terminal, still in the main.tf file, replace <SUBNET\_ID> with the one you just copied.
21. Save and exit the file by pressing **Escape** followed by :wq!.
22. In the Terraform Cloud console, click your user profile and select **User Settings**.
23. In the left-hand menu, click **Tokens**.
24. Click **Create an API token**.
25. For **Description**, enter *terraform-cloud*.
26. Click **Create API token**.
27. Copy the API token, and paste it into a text file.
28. In the terminal, enter:

terraform login

1. At the prompt, enter yes.
2. At the prompt asking for the API token, the API token you just copied.
   * You should then see a welcome message.

**Migrate the Existing Terraform Project into Terraform Cloud**

1. Reopen main.tf:

vim main.tf

1. In the Terraform Cloud console, click **Done** in the API token dialog.
2. Click the Terraform logo in the upper left.
3. Click the **terraform-guru** organization.
4. Click the **terraform-cloud-guru** workspace.
5. Under **CLI-driven runs**, copy everything in the **Example code** box except the first and last lines, which would be:

cloud { organization = "terraform-guru" workspaces { name = "terraform-cloud-guru" } }

1. In the terminal, beneath terraform {, indent two spaces and paste in that code.
2. Copy the AMI ID in the variable "ami" block.
3. In the Terraform Cloud console, in the left-hand menu, click **Variables**.
4. Click **Add variable**, and set the following values:
   * **Select variable category:** Select *Terraform variable*.
   * **Key:** Enter *instance\_count*.
   * **Value:** Enter *2*.
5. Click **Save variable**.
6. Click **Add variable**, and set the following values:
   * **Select variable category:** Select *Terraform variable*.
   * **Key:** Enter *ami*.
   * **Value:** Paste in the AMI ID you copied.
7. Click **Save variable**.
8. Click **Add variable**, and set the following values:
   * **Select variable category:** Select *Terraform variable*.
   * **Key:** Enter *instance\_type*.
   * **Value:** Enter *t3.micro*.
9. Click **Save variable**.
10. Click **Add variable**, and set the following values:
    * **Select variable category:** Select *Terraform variable*.
    * **Key:** Enter *subnet*.
    * **Value:** Paste in the subnet ID you copied earlier.
11. Click **Save variable**.
12. In the terminal, delete the type and default lines from each variable block, so they all look like:

variable "instance type" {}

1. Save and exit the file by pressing **Escape** followed by :wq!.

**Deploy the EC2 Instances with Terraform Cloud**

1. Initialize the working directory:

terraform init

You should then see a success message.

1. Validate the code:

terraform validate

You should then see a success message.

1. Apply your configuration:

terraform apply

1. At the prompt, enter yes.
2. In the Terraform Cloud console, in the left-hand menu, click **Runs**.
   * You should see a run has been triggered.
3. Click the **Triggered via CLI** box to see the **Apply running** output.
4. In the terminal, note that you see the same output listed.
5. In the AWS Management Console, navigate to **EC2** > **Instances**.
   * You should see three instances listed, including the two you just created.

**Create a Terraform Module from Existing Code**

1. In the GitHub console, create a new repo, and set the following values:
   * **Owner:** Select yourself.
   * **Repository name:** Enter *terraform-as-ec2-psacg*.
   * **Private:** Select.
   * **Add .gitignore:** Search for and select **Terraform**.
2. Click **Create repository**.
3. Click **Add file** > **Create new file**.
4. For the file name, enter *main.tf*.
5. In the terminal, reopen main.tf:

vim main.tf

1. Copy everything from the resource block down, which should be:

resource "aws\_instance" "inst" { count = var.instance\_count ami = var.ami instance\_type = var.instance\_type subnet\_id = var.subnet tags = { Name = "TERRAFORM-GURU-${count.index}" } } variable "instance\_count" {} variable "ami" {} variable "instance\_type" {} variable "subnet" {} output "aws\_instances" { value = aws\_instance.inst.\*.tags.Name }

1. Paste it into the **Edit new file** window.
2. Click **Commit new file**.
3. Click **0 tags**.
4. Click **Create a new release**.
5. Click **Choose a tag**.
6. Enter *1.0.0*.
7. Click **Create new tag: 1.0.0 on publish**.
8. For **Release title**, enter *Module v1.0.0*.
9. Click **Publish release**.
10. Click **Code**.
11. In the upper right, click your avatar, and select **Settings**.
12. In the left-hand menu, click **Applications**.
    * You shouldn't see any apps listed.
13. In the Terraform Cloud console, in the left-hand menu, click **Overview**.
14. Click **Actions** > **Lock workspace**.
15. In the left-hand menu, click **Settings** > **Version Control**.
16. Click **Connect to version control**.
17. Click **Version control workflow**.
18. Click **GitHub**.
19. In the **Install Terraform Cloud** dialog, select your GitHub user account.
20. Select **Only select repositories**.
21. Click **Select repositories**.
22. Click the **terraform-as-ec2-psacg** repository.
23. Click **Install**.
24. In the Terraform Cloud console, under **Choose a repository**, click **terraform-as-ec2-psacg**.
25. Leave all the default settings, and click **Update VCS settings**.
26. In the left-hand menu, click **terraform-cloud-guru**.
27. In the left-hand menu, click **Workspaces**.
28. In the left-hand menu, click **Registry**.
29. With **Modules** selected, click **Publish a module**.
30. Click **GitHub**.
31. Click the **terraform-as-ec2-psacg** repository.
32. Click **Publish module**.

**Modify the Terraform Configuration to use the Module and Redeploy**

1. On the right-hand side, copy what's provided under **Copy configuration details**.
2. In the terminal, still in the main.tf file, delete the entire resource block.
3. Paste in the configuration details you copied.
4. Replace # insert required variables here with the following lines:

instance\_count = var.instance\_count ami = var.ami instance\_type = var.instance\_type subnet = var.subnet

1. In the output block, change the value to module.ec2-psacg.\*.aws\_instances:

output "aws\_instances" { value = module.ec2-psacg.\*.aws\_instances

1. Save and exit the file by pressing **Escape** followed by :wq!.
2. In the Terraform Cloud console, in the left-hand menu, click **Workspaces**.
3. Click **terraform-cloud-guru**.
4. Click **Settings** > **Version Control**.
5. Under **Connected to VCS**, click **Change source**.
6. Click **CLI-driven workflow**.
7. Click **Update VCS settings**.
8. In the left-hand menu, click **terraform-cloud-guru**.
9. Under **Latest Run**, click **See details**.
10. Click **Cancel Run**.
11. Click **Cancel Run** again.
12. Click **Actions** > **Unlock workspace**.
13. Click **Yes, unlock workspace**.
14. In the terminal, initialize the working directory:

terraform init

You should then see a success message.

1. Validate the code:

terraform validate

You should then see a success message.

1. Apply your configuration:

terraform apply

1. At the prompt, enter yes.
2. In the Terraform Cloud console, in the left-hand menu, click **Runs**.
   * You should see a run has been triggered.
3. Click the **Triggered via CLI** box to see the **Apply running** output.
4. In the terminal, note that you see the same output listed.
5. In the AWS Management Console, navigate to **EC2** > **Instances**.
   * You should see the new instances you originally created have been terminated, and two new instances have been created.

**Conclusion**