Introduction to Terraform – Creating a simple local resource

Contents

This lab will create a Docker Container using the Cloud Shell in Google Cloud.

Lab Objectives

In this lab, you will:

1. Review the Terraform registry and Docker documentation
2. Build a simple main.tf configuration file based on example code
3. Deploy a Docker image and container
4. Test the container
5. Modify the deployment
6. Test the modified deployment
7. Destroy the deployment

There is a Solution section at the very end of these instructions. Try to use this only as a last resort if you are struggling to complete the step-by-step processes.

Lab Setup

For each lab, you are provided with a new Google Cloud project and a set of resources for a fixed time at no cost.

1. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time. There is no pause feature. You can restart if needed, but you have to start at the beginning.

2. When ready, click **Start lab.**

3. Note your lab credentials (Username and Password). You will use them to sign in to the Google Cloud Console.

4. Click **Open Google Console**.

5. Click **Use another account** and copy/paste credentials for this lab into the prompts. If you use other credentials, you'll receive errors or incur charges.

6. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Task 1. Set up Cloud Shell as your IDE

1. In the Cloud Console, click Activate Cloud Shell

2. If prompted, click Continue.

3. Select the **Open in a new window** icon

4. Select the **Open Editor** icon

5. Select **Open Home Workspace** if explorer pane does not display.

6. Create a directory for your Terraform configuration and initial files by running the following commands in the terminal:

```
cd ~
mkdir ./lab
cd ./lab
touch main.tf
```

7. In the Explorer pane, navigate to and click on main.tf to open the empty file in the File Pane.

8. You now have 2 browser tabs open. This tab will be referred to as the `IDE` throughout these instructions.

9. Switch to the Google console tab and close the cloud shell. This tab will be referred to as the `Console` throughout these instructions.

10. Switch back to the `IDE`

11. Execute all terraform commands from within the lab directory

Task 2: Review the documentation and create a configuration file

1. Review docker/TF documentation: https://registry.terraform.io/providers/kreuzwerker/docker/3.0.1/docs

2. Note: At time of writing, version 3.0.1 was the latest version of this Provider. This may have changed. This lab was created and tested using version 3.0.1 so please ensure you use this version.

3. Click `**Use Provider**`

4. Copy the code block into **main.tf** For convenience the code is listed below:

```
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version = "3.0.1"
    }
  }
}

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

5. From within the 'Example Usage' section of the documentation, copy the sections relating to '**Pulls the image'** and 'Create a container' and add these lines below the current content of main.tf For convenience the code is listed below:

```
# Pulls the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
}
```

Task 3: Update the configuration

1. Modify the docker_image resource block to deploy the latest httpd image rather than Ubuntu. Rename the resource to ‘apache_web’ and reference the latest httpd image.

   ```
   resource "docker_image" "apache_web" {
    name  = "httpd:latest"
   }
   ```

2. Modify the docker_container resource block to create a container called **‘web_server’** using the docker_image resource **‘apache_web’**

   ```
   resource "docker_container" "web_server" {
    image = docker_image.apache_web.image_id
    name  = "web_server"
   }
   ```

3. Add ports for internal 80 external 8080. The docker_container resource block should now be

   ```
   resource "docker_container" "web_server" {
    image = docker_image.apache_web.image_id
    name  = "web_server"
    ports {
      internal = 80
      external = 8080
    }
   }
   ```

4. Save main.tf

Task 4: Run Terraform & Test

1. 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
5. Run `**docker images**` in CLI. The httpd image has been downloaded

6. Run `**docker ps**` to list your containers running

We will now modify the deployment by changing the external port number from 8080 to 88. Some resource changes can be applied to the existing instance of the resource, whilst others require the destruction of the original resource and the creation of a replacement. Let’s see whether a port change modifies or recreates this resource.

7. Note the alphanumeric value of the container ID. For example, something similar to: 7671010f97a8
8. Type curl http://127.0.0.1:8080 View the "it Works" response that is standard with an Apache web server (index.html)
9. Return to the main.tf and change the external port to 88.
10. Save the file
11. Run `terraform plan` and review the changes
12. Run `terraform apply`, typing `yes` when prompted
13. Once the new deployment completes, type `curl http://127.0.0.1:8080` This should fail as we no longer have a container running on port 8008
14. Type `curl http://127.0.0.1:88`We should return a success "it works"
15. Run `docker ps`

Note that the name of the container is the same, but the ID has changed - Terraform destroyed the original container and deployed a replacement. This is an important point to be aware of as we progress through the course. Whether we ‘update’ or ‘replace’ a resource depends upon the resource itself and the changes we make.

Task 5: Destroy your deployment

1. Run `terraform destroy` review the output and type `yes`
2. Run `docker ps` to confirm your container has been deleted
3. Run `docker images` and confirm your image has been deleted

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

Solution

The completed main.tf for this lab is...

```
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version = "3.0.1"
    }
  }
}

provider "docker" {
  # Configuration options
}

# Pulls the image
resource "docker_image" "apache_web" {
  name  = "httpd:latest"
}

# Create a container
resource "docker_container" "web_server" {
  image = docker_image.apache_web.image_id
  name  = "web_server"
  ports {
    internal = 80
    #external =8080
    external = 88
  }
}
```