Terraform

Install Terraform in CentOS

- sudo yum install wget
- sudo yum install unzip
- wget https://releases.hashicorp.com/terraform/0.12.24/terraform_0.12.24_linux_amd64.zip
- unzip terraform_0.12.24_linux_amd64.zip
- sudo mv terraform /usr/local/bin
- terraform --version

Terraform HCL

- It is human-readable as well as machine friendly.
- Terraform can also read 'JSON' configurations.
- However, it is preferable to use HCL Configuration syntax.

Comments

There are two types of comments in Terraform:

Single line comments # and multi-line comments are wrapped in /* and */

Terraform Workflow

- **Terraform init** - Initializes the working directory having Terraform configuration files.
- **Terraform plan** - Creates an execution plan and it mentions the changes it is going to make.
- **Terraform apply** - When you are not amazed by seeing the results of the terraform plan, you can go ahead and run terraform apply to apply the changes.

The main difference between plan and apply is - apply asks for a prompt like Do you want to perform these actions? and then implement the changes, whereas, plan shows the possible end state of your infrastructure without implementing them.

- Terraform destroy - Destroys the created infrastructure.

Terraform configuration

- A set of files that describe the infrastructure in Terraform is called as Terraform Configuration.
- The entire configuration file is saved with .tf extension. Make sure to have all the configurations maintained in a single .tf file instead of many. Terraform loads all the .tf files present in the working directory.
- The scope of terraform is to a directory

Creating Virtual Network

- Create a file with .tf extension and copy the below code into it.

```
resource "azurerm_virtual_network""myterraformnetwork"
 {
   name = "myvnet"
   address_space = ["10.0.0.0/16"]
   location="East US"
   resource_group_name="er-tyjy"
  }
```

Explaining .tf file

- Resource blocks define the infrastructure (resource "azurerm_virtual_network" "myterraformnetwork"). It contains the two strings - type of the resource(azurerm_virtualnetwork) and name of the resource(myterraformnetwork).
- resource "azurerm_virtual_network""myterraformnetwork" (Correct format)
- resource "azurerm_virtual_network" "myterraformnetwork" (Correct format)
- Space between type and name of the resource is valid
- Within the resource block, the configuration is mentioned. This varies depending on the service provider.

Running .tf File

DEMO

Terraform Validate

- There is a command called terraform validate that validates the information you provided.
- You can run this command whenever you like to check whether the code is correct. Once you run this command, if there is no output, it means that there are no errors in the code.

Signs in Terraform

- After running terraform plan and terraform apply, you can see the symbols like +,- ,-/+ or ~.
- In Terraform, each symbol has a particular meaning as git.
- If a resource is added, it indicates + sign. If a resource is deleted, it is indicated with - and if a resource is altered it is indicated by -/+ and ~ indicates resources are updated in place.

Destroying Infrastructure

- Now, you have an idea about how to create Terraform infrastructure successfully. It is time to learn how to destroy the infrastructure.
- You can destroy the infrastructure by using Terraform destroy command.
- When you run the command, the resources that are going to be destroyed is shown by using -
- After this, Terraform shows the execution plan and waits for approval before making any changes.
- As like apply, the terraform shows the plan in the order it is going to destroy.

State File

- terraform.tfstate
- It is a JSON file that maps the real world resources to the configuration files. In terraform, there are three states
- Desired state - Represents how the infrastructure to be present.
- Actual state - Represents the current state of infrastructure.
- Known state - To bridge desired state and actual state, there is a known state. You can get the details of known state from a tfstate file.
- When you run terrfarom plan command, terraform performs a quick refresh and lists the actions needed to achieve the desired state.
- When terraform apply command is run, it applies the changes and updates the actual state.

Manually Changing the Infrastructure

- It is advised not to change any of the configurations manually, i.e. from the web console (Azure Portal)

Mapping to Real World

- When you have the resource in your configuration like resource "aws_instance" "foo". Terraform uses this as a map to know whether aws_instance is represented by the corresponding resource.
- Early prototypes of terraform use tags because there is no concept of state file.
- This leads to problems because not all the resources and cloud providers support tags.
- Therefore, in order to avoid these problems terraform has its own state structure.

Performance

- To perform the basic mapping, Terraform stores the cache of attribute values of available resources in the state.
- By doing so, the performance is improved.
- When you run terraform plan command, it knows what changes should be applied to reach the desired configuration.
- For small infrastructures, terraform queries the providers and sync is performed with the latest attributes from all the resources.
- For larger infrastructures, to query a resource it takes time. Many cloud providers do not have APIs to perform a query on the multiple resources at a time.
- There is an API rate limitation due to which you can request only a certain number of resources.
- Proceed further to know the solution to the problem.

refresh=false

- You can avoid API calling multiple times by using -refresh =false or -target flag. The cached state is considered as the record of truth.
- terraform plan -refresh=false

State File

Demo

Syncing

- If multiple people are working on the same infrastructure and making changes at the same time, it causes problems.
- There is a solution to this problem. Remote state, fully featured backend so that terraform can use the remote locking to avoid running terraform at the same time and it ensures that terraform is running with the most recent update.