Exploring Terraform and Terragrunt on Windows with Docker and WSL2

This tutorial assumes you are familiar with basic command-line operations and have Docker and WSL2 already set up.

This tutorial will walk you through: 1. Setting up your environment 1. Install Terraform and Terragrunt 1. Explore their basic functionalities using a local provider

Introduction

In the modern era of cloud computing, managing infrastructure manually is no longer a viable or efficient option. Infrastructure as Code (IaC) has emerged as a key practice in the DevOps culture, enabling developers and operations teams to define, provision, and manage the infrastructure required for applications to run, using code.

Terraform and Terragrunt are two powerful tools that embody this concept. Terraform, an open-source tool created by HashiCorp, allows you to define and provide data center infrastructure using a declarative configuration language. This means you describe your desired state of infrastructure, and Terraform will figure out how to achieve that state. It supports a multitude of cloud providers, making it a versatile tool in the creation and management of any infrastructure.

Terragrunt, on the other hand, acts as a thin wrapper for Terraform, providing extra tools for keeping your Terraform configurations DRY, maintainable, and modular. It simplifies the management of remote state and provides a way to orchestrate dependencies between your resources.

By using Terraform and Terragrunt, you can automate the process of setting up your infrastructure, making it faster, more efficient, and less error-prone. This tutorial will guide you through the process of setting up and exploring these two powerful tools.

What is Terraform?

Terraform is an open-source Infrastructure as Code (IaC) software tool created by HashiCorp. It allows you to define and provide data center infrastructure using a declarative (desired state) configuration language.

Terraform can manage local, remote, on-premises and hybrid cloud infrastructures. With Terraform, you can manage a wide variety of systems and services. The current terraform providers can be found here.

One of the key features of Terraform is its plan command, which allows you to see what changes Terraform will apply before it does so, providing a sort of "preview" of the changes.

Terraform is widely used in DevOps practices for automating infrastructure setup and consistently replicating environments, making it a crucial tool in the modern cloud-focused landscapes.

Terraform uses HashiCorp Configuration Language (HCL) to define the infrastructure. HCL is a declarative language that allows you to define the desired state of your infrastructure. Hashicorp Terraform Style Guide

What is Terragrunt?

Terragrunt is a thin wrapper for Terraform that provides extra tools for working with multiple Terraform modules. It's an open-source tool created by Gruntwork.

Terragrunt helps to keep your Terraform code DRY (Don't Repeat Yourself), maintainable, and modular, while also providing a way to compose and combine configurations, manage remote state, and apply configurations in a specific order.

One of the key features of Terragrunt is its ability to manage dependencies between Terraform modules. This means you can ensure resources that depend on each other are created in the correct order.

Terragrunt also provides locking and error handling for concurrent remote operations, which is useful when multiple team members are working on the same infrastructure.

In essence, while Terraform is used for creating, modifying, and versioning infrastructure safely and efficiently, Terragrunt is used to manage and orchestrate those resources in a more controlled and structured manner.

Step 1: Set Up Your WSL2 Environment

- **Install and Configure Ubuntu:**
  - Open the Microsoft Store and search for "Ubuntu."
  - Install Ubuntu or your preferred Linux distribution.
  - Open the Ubuntu terminal and complete the initial setup.
- **Configure WSL2 as the Default:**
  - Open PowerShell as an administrator.
  - Run `wsl --set-default-version 2` to set WSL2 as the default version.

Step 2: Install Terraform and Terragrunt

- **Download Terraform:**
  - Get the latest version of the Terraform binary from the official website: https://www.terraform.io/downloads.html or use the following commands.
  - In the Ubuntu terminal, run the following command to download and install Terraform:

```
sudo apt-get install -y unzip jq

# Get terraform
terraform_latest_version=$(curl -s
https://api.github.com/repos/hashicorp/terraform/releases/latest | jq -r
'.tag_name')
TERRAFORM_VERSION="${terraform_latest_version#v}"
curl -o terraform.zip
"https://releases.hashicorp.com/terraform/$TERRAFORM_VERSION/terraform_${TERRAFORM
_VERSION}_linux_amd64.zip"

# Get terragrunt
# https://github.com/gruntwork-
io/terragrunt/releases/download/v0.57.1/terragrunt_linux_amd64
TERRAGRUNT_VERSION=$(curl -s https://api.github.com/repos/gruntwork-
io/terragrunt/releases/latest | jq -r '.tag_name')
curl -L -o terragrunt "https://github.com/gruntwork-
```

```
io/terragrunt/releases/download/$TERRAGRUNT_VERSION/terragrunt_linux_amd64"

# Move terraform and terragrunt to /usr/local/bin
chmod +x terragrunt
sudo mv terragrunt /usr/local/bin/
unzip terraform.zip
sudo mv terraform /usr/local/bin/

# Cleanup
rm terraform.zip
rm terragrunt
```

- **Verify Terraform and Terragrunt Installation:**
  - Run `terraform version` in the terminal to verify the installation.
  - Run `terragrunt --version` in the terminal to verify the installation.

Step 4: Explore Terraform and Terragrunt

If using VSCode, I recommend installing the following extensions:

- hashicorp.hcl
- hashicorp.terraform

Create a New Project Directory in the terminal and Navigate into It:

```
mkdir my-terraform-project
cd my-terraform-project
```

- **Create a Terraform Configuration File:**
  - Create a file named `main.tf` in your project directory:
  - Add a basic Terraform configuration:

```
terraform {
    required_providers {
        local = {
        source = "hashicorp/local()"
        version = "2.5.1"
        }
    }
}

resource "local_file" "foo" {
            content  = "foo!"
            filename = "${path.module}/bar/foo.bar"
        }
```

- **Initialize Terraform:**
  - Run the following command to initialize your Terraform project:

```
terraform init
```

- **Plan and Apply Configuration:**
  - Plan your changes and view the execution plan:

```
terraform plan
```

  - Apply your changes:

```
terraform apply
```

- **Explore Terragrunt:**
  - Add a configuration that includes the path to your Terraform configuration:

```
terraform {
    source = "./"
}
```

  - Run Terragrunt commands in place of Terraform commands, such as:

```
terragrunt plan
terragrunt apply
terragrunt destroy
```

Conclusion

With this setup, you now have Terraform and Terragrunt installed and configured on your Windows machine with Docker and WSL2. You can begin exploring infrastructure as code (IaC) by writing and managing Terraform configurations with the assistance of Terragrunt for managing dependencies and environments. Make sure to explore additional resources and examples to expand your knowledge and experience.

Start by searching for modules and providers. For local development, you can use the local provider to create resources on your local machine. For example, you can create a local file using the local_file resource type. The resource foo is created with the content "foo!" and is saved in the path specified by ${path.module}/bar/foo.bar.