# Implementing Terraform to Deploy Your Virtual Machines

**Andrew Mallett**
LINUX AUTHOR AND TRAINER

@theurbanpenguin www.theurbanpenguin.com

# Objectives

Understanding Terraform

- Part of the Hashicorp product range
- Deploy and manage virtual machines and containers

# Terraform

Using one interface you can deploy public and private cloud virtual machines and containers.

```
$ sudo apt-get update
$ sudo apt-get install -y gnupg software-properties-common curl
$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com
$(lsb_release -cs) main"
$ sudo apt install terraform
$ terraform version
$ terraform -install-autocomplete
$ source .bashrc
```

# Installing Terraform on Ubuntu 20.04

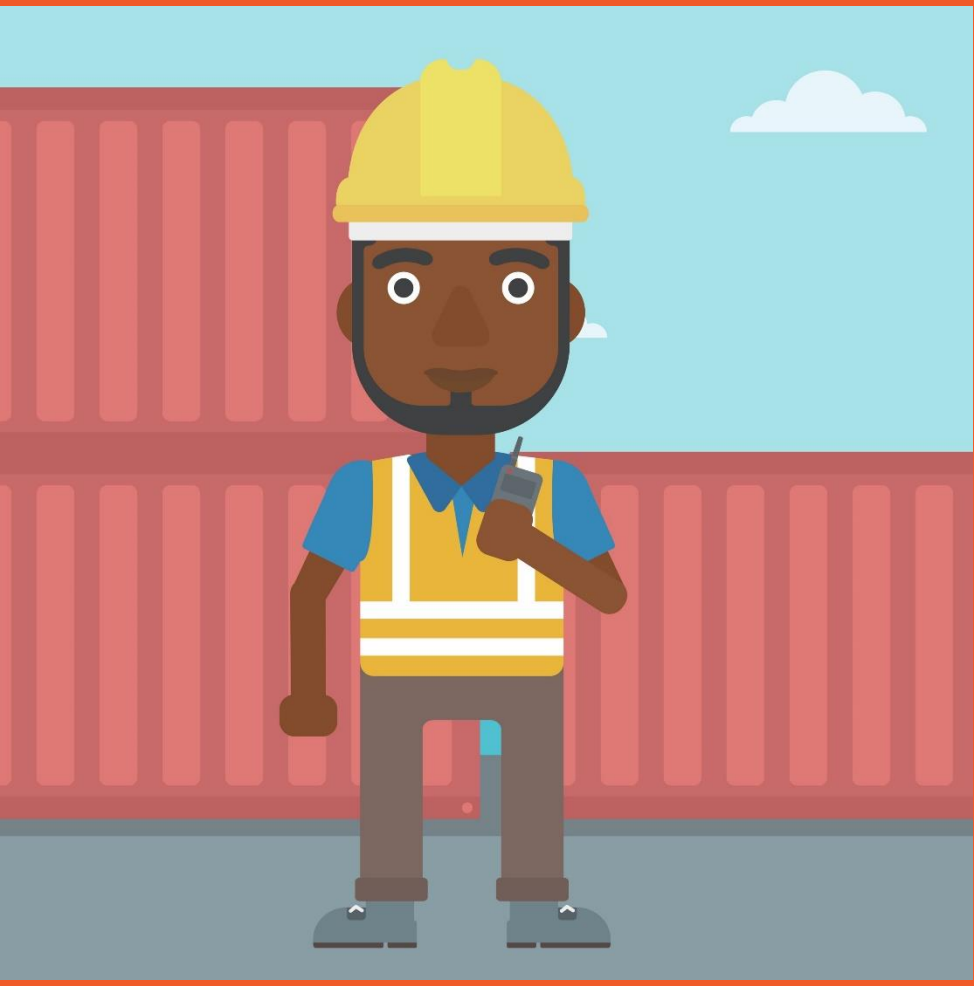
https://learn.hashicorp.com/tutorials/terraform/install-cli

We stick with Ubuntu and add in the Terraform repositories before installing.

# Demo

**Working on the Ubuntu System:**

- Add repositories for Terraform
- Install Terraform



## Terraform Providers:

- Interface to the container or cloud provider
- https://registry.terraform.io/browse/providers
- We will use the docker provider

```
$ sudo apt install -y docker.io
$ sudo gpasswd -a $USER docker
$ exit
$ mkdir terraform ; cd terraform
$ vim main.tf
$ terraform init
```

# Working with Providers

We can make sure that we have Docker installed, we can also use this for the next module. We can then create a working directory for Terraform and create a main.tf file. This will define the provider we are using. Using the init subcommand we can install the driver

main.tf

```
terraform {

  required_providers {

    docker = {

      source = "kreuzwerker/docker"

      version = "2.16.0"
    }
  }
}

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

# Demo

**Terraform and Docker**

- Install Docker
- Add account to Docker group
- Create main.tf
- Download provider

```
resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
```

# Docker Image

When working with Docker we need to define the master image that we want to use with containers

partial main.tf

```
resource "docker_container" "nginx" {
  image = docker_image.nginx.latest
  name  = "webserver"
  ports {
    internal = 80
    external = 8000
  }
  volumes {
    container_path = "/usr/share/nginx/html/"
    host_path = "/home/vagrant/terraform/web/"
  }
}
```

# Demo

**Working with Containers:**

- Image resource
- Container resource
- Map network ports
- Connect host volumes

# Demo

**Automation:**

- Delete vagrant vm
- Recreate
- Install ansible
- Run playbook to recreate all

# Summary

Understanding Hashicorp's Terraform

- Deploy VMs and containers
- Add repository for your OS
- Install Terraform
- Add bash completion
- Create project directory
- Define main.tf
  - Setup provider plugins
  - terraform init
  - Define resources
  - terraform apply

# Implementing Containers Using Docker