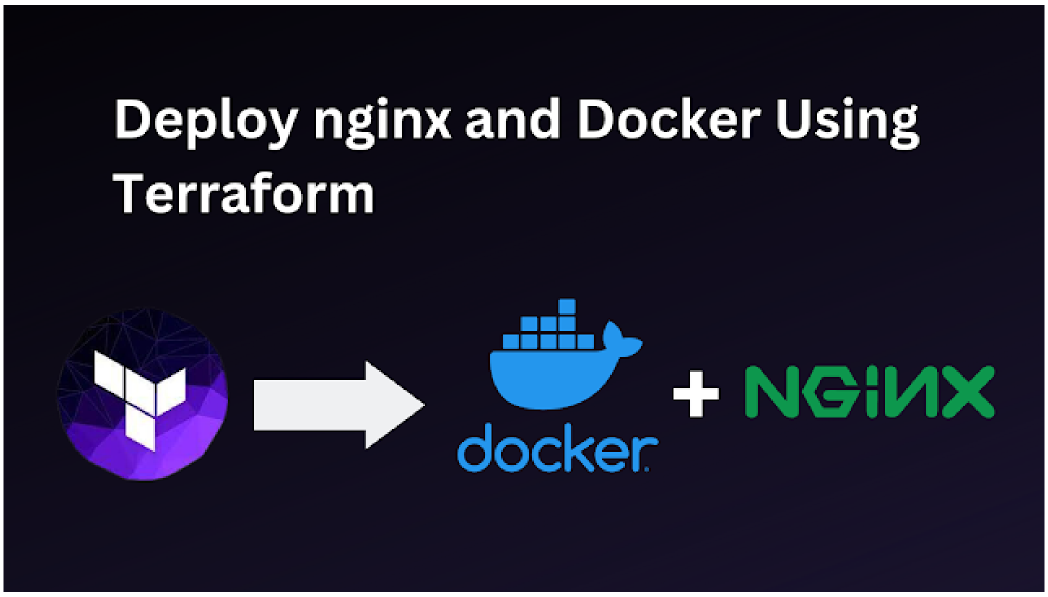
Project on Deploy Nginx and Docker Using Terraform

April 11, 2023



Introduction

Terraform is a powerful Infrastructure-as-Code (IaC) tool that allows you to define and manage your infrastructure in a declarative way. Docker is a popular containerization technology that provides a platform for developers to package, distribute, and run their applications. In this blog post, we will explore how to define the Docker provider in a Terraform configuration and deploy a simple Nginx Docker container.

Prerequisites

Before we get started, you'll need the following:

- A Docker registry to store your Nginx Docker image
- Access to a server or cloud provider to deploy your infrastructure

Step 1: Define your Docker provider

The first step is to define the Docker provider in your Terraform configuration. Create a new directory for your Terraform configuration files:

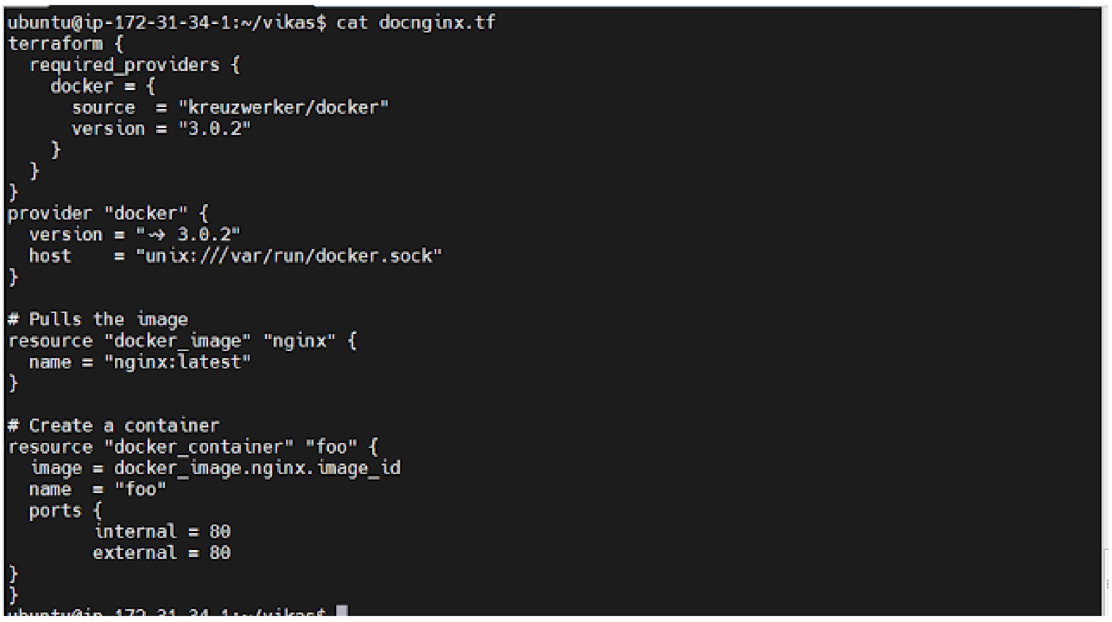
shell

$ mkdir nginx-terraform

$ cd nginx-terraform

Next, create a new file called main.tf and add the following code to define your Docker provider:

Terraform file defines a Docker provider with the host parameter set to the address of your Docker host, which could be a remote Docker host or a local Docker daemon. If you're using a remote Docker host, make sure you have SSH access to the server and that the Docker daemon is running.



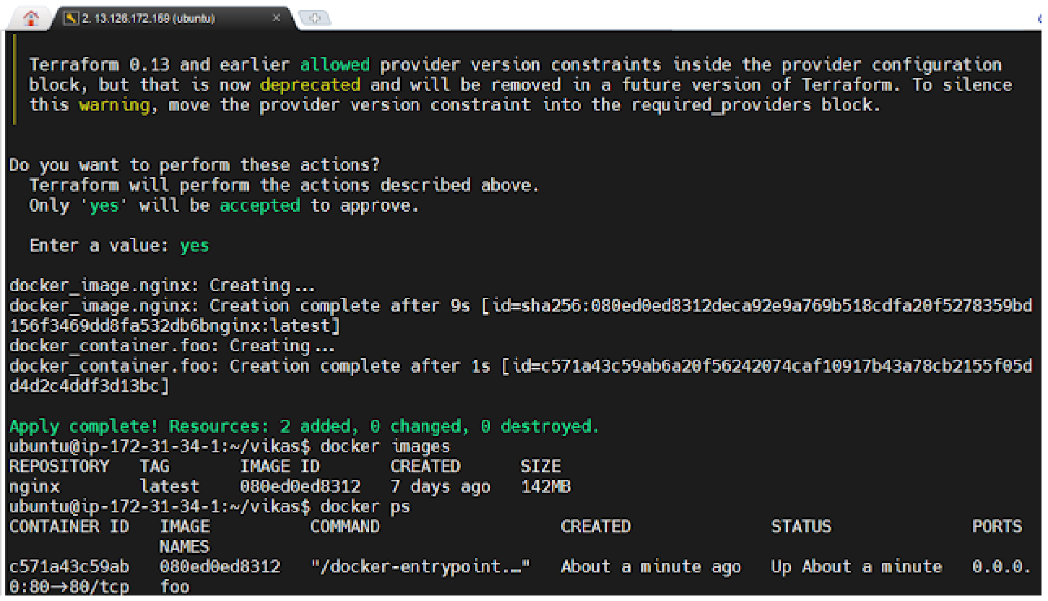
```
ubuntu@ip-172-31-34-1:~/vikas$ cat docnginx.tf
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "3.0.2"
    }
  }
}
provider "docker" {
  version = "~> 3.0.2"
  host    = "unix:///var/run/docker.sock"
}

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

# Create a container
resource "docker_container" "foo" {
  image = docker_image.nginx.image_id
  name  = "foo"
  ports {
        internal = 80
        external = 80
}
}
ubuntu@ip-172-31-34-1:~/vikas$
```

Now that you have your Docker provider defined, it's time to create your Nginx Docker container. Add the following code to your main.tf file:

This Terraform configuration file defines a docker_container resource with the name parameter set to "nginx" and the image parameter set to the official Nginx Docker image. The ports block maps the container's internal port 80 to the host's external port 8080.



```
2. 13.126.172.169 (ubuntu)

  Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration
  block, but that is now deprecated and will be removed in a future version of Terraform. To silence
  this warning, move the provider version constraint into the required_providers block.


Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

docker_image.nginx: Creating...
docker_image.nginx: Creation complete after 9s [id=sha256:080ed0ed8312deca92e9a769b518cdfa20f5278359bd
156f3469dd8fa532db6bnginx:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=c571a43c59ab6a20f56242074caf10917b43a78cb2155f05d
d4d2c4ddf3d13bc]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-34-1:~/vikas$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED      SIZE
nginx         latest    080ed0ed8312   7 days ago   142MB
ubuntu@ip-172-31-34-1:~/vikas$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS
               NAMES
c571a43c59ab   080ed0ed8312   "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.
0:80->80/tcp   foo
```

Step 3: Deploy your infrastructure

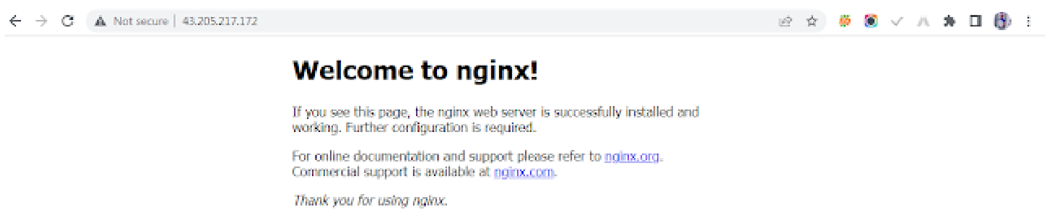
Now that your Terraform configuration is defined, it's time to deploy your infrastructure. Run the following commands in your terminal:

shell

$ terraform init

$ terraform apply

or cloud provider.



Not secure | 43.205.217.172

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

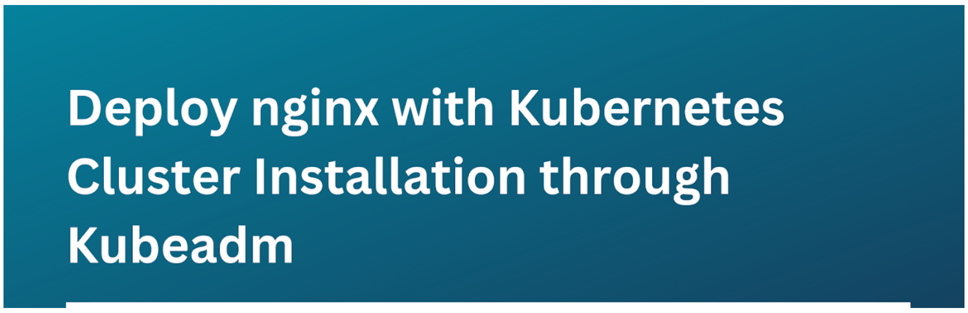
Thank you for using nginx.

Conclusion

In this blog post, we explored how to define the Docker provider in a Terraform configuration and deploy a simple Nginx Docker container. By using Terraform to manage your Docker containers, you can easily version, test, and deploy your infrastructure as code.

DEVOPS

Enter comment



Deploy nginx with Kubernetes Cluster Installation through Kubeadm

First we need to create 2 EC2 instances i.e t2 medium for master node and t2 micro for worker node. And allow ports for their connection. Run all below command on both master and worker nodes sudo apt update -y sudo apt install docker.io -y sudo systemctl start docker sud ...

Deploy wordpress using IAM role, RDS, Docker and push it to ECR and run on ECS

Guide: Deploying WordPress using IAM Role, RDS, Docker, ECS, and pushing it to ECR

Before we get started, it's important to understand the role of each component in this deployment: IAM role: Allows EC2 instances to securely access AWS services such as RDS without needing to store AWS credentials on the instance itself. Docker: A containerization platform tl ...

Nginx web Deployed on Kubernetes Cluster Using Deploy, Service, and Ingress Yaml file

Nginx Web Deployed on Kubernetes Cluster Using Deploy, Service, and Ingress Yaml Files

In this blog, we are going to explore how to deploy nginx to Kubernetes using a Deployment file, Service file, and Ingress file. These files are