The Ruby on Rails Web Application with Terraform & Jenkins

Contents

1. Introduction:

This documentation will guide through the deployment of a Ruby on Rails web application on AWS, leveraging Terraform for infrastructure as code. It is used for defining, managing, and provisioning infrastructure resources in a declarative manner. Terraform allows you to describe your infrastructure using code, specifying the desired state of your infrastructure in configuration files. Here we have to fork the repository, build docker images, and create AWS resources for a scalable and secure application. With a focus on best practices, and integrate RDS and S3, ensuring IAM role authentication and database security. This project showcases the synergy between modern web development and infrastructure automation, offering a comprehensive solution for deploying web applications.

2. Prerequisites

Before begin with the deployment of "The Ruby on Rails Web Application with Terraform Scripts," ensure that we have the following prerequisites:

- AWS Account.
- EC2 Ubuntu 22.04LTS Instance.
- Create new IAM User.
- Create IAM Role.
- Linux (Ubuntu) Package Manager
- Docker
- Jenkins

3. What Will We Do

To get started with this project, here's an overview of the essential steps will follow:

3.1 Login to AWS:

If you don't have an AWS account, you will begin by creating one and logging into the AWS Management Console.

3.2 Create EC2 Instance:

You will create an EC2 instance that runs Ubuntu 22.04 LTS (or Linux) to serve as the environment for running Terraform scripts and deploying the Ruby on Rails web application. Ensure that this instance meets the following system requirements:

- Operating System: Ubuntu 22.04 LTS
- RAM: At least 1GB
- vCPU: 1
- Family: t2.micro

3.3 Create New IAM User:

Within the AWS Console, you will create a new IAM user specifically for Terraform. This user will have the necessary permissions to manage AWS resources as code.

3.4 Create IAM Role:

Simultaneously, you will create an IAM role for authentication and to enhance security in the management of AWS resources.

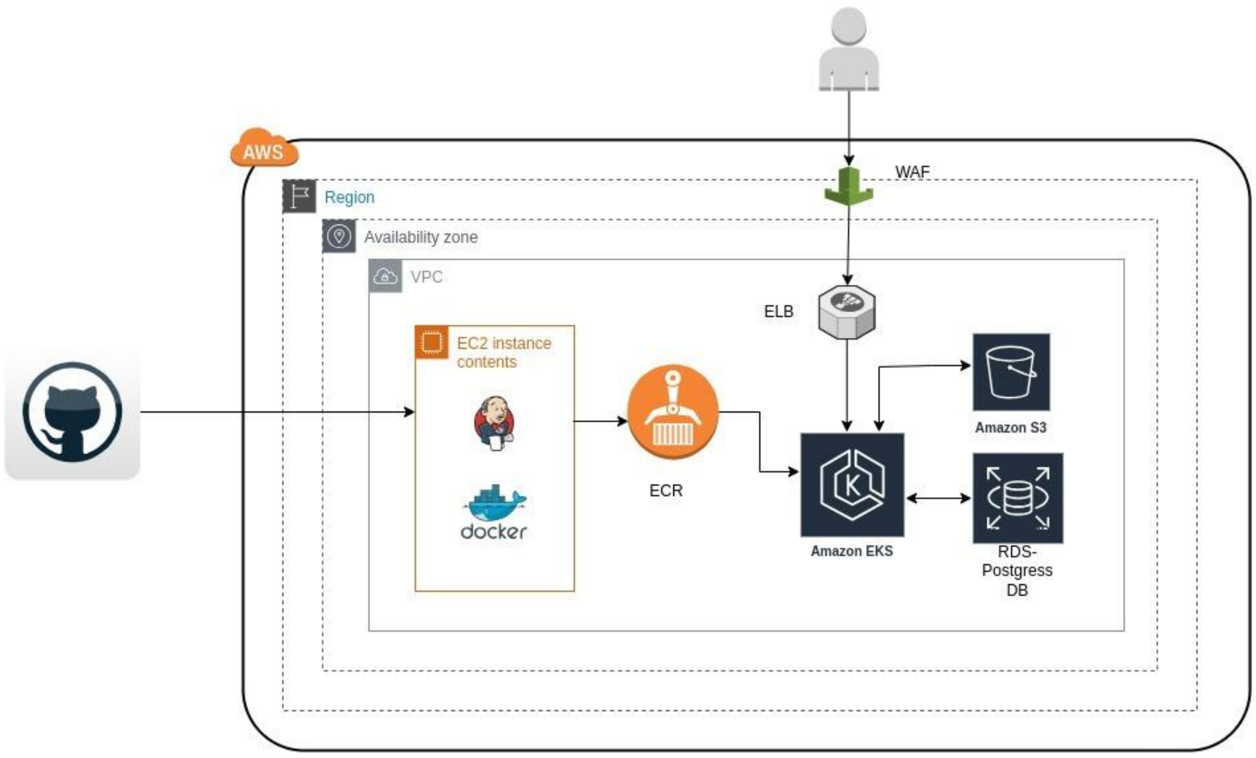
3.5 Create EKS Cluster:

Create an Amazon EKS cluster to serve as the orchestration platform for deploying and managing your application containers.

3.6 Terraform Script:

You will use Terraform scripts to provision the infrastructure for the web application. These scripts will define the desired state of the infrastructure, enabling you to automate the provisioning process effectively.

4. Architecture



Architecture for Ruby on Rails Web Application

5. Terraform Script Execution

Follow these steps to execute your Terraform scripts:

5.1 Set-Up EC2 Instance

- ✓ Launch an EC2 instance running Ubuntu 22.04 LTS.
- ✓ Create an IAM user in AWS with programmatic access and AdministratorAccess permissions.
- ✓ Configure AWS CLI on the EC2 instance with the IAM user's credentials.

5.2 Install Terraform

- Download the Terraform binary from the official website.

  I'm using Ubuntu so here are the commands to get Terraform

  1. Download Terraform stable release here.

     a. Go to browser search (Install | Terraform | HashiCorp Developer)

     b. Click download options in the top of the page

     c. Choose operating system ex. (Linux or Ubuntu)

     d. Select Package manager ex. (Linux or Ubuntu)

     e. Executing the following commands one by one for Amazon Linux

```
$ sudo yum install -y yum-utils shadow-utils
$ sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.reo
```

- Install Terraform on your EC2 instance using the package manager (apt).

```
$ sudo yum -y install terraform
```

- Confirm the Terraform installation by running terraform --version.

```
$ terraform --version
```

5.3 Initialize Terraform

- Find and Clone the code from git repository by given link below:

  https://github.com/Vignesh0519/Dev-Terra.git

- Navigate to the project directory.
- Copy and paste your Terraform scripts into the project directory.
- Initialize the Terraform project with **terraform init**.

```
$ terraform init
```

5.4 Validate Terraform Configuration

- Run **terraform validate** to check the syntax and validity of your Terraform configuration.

```
$ terraform validate
```

5.5 Plan Terraform Deployment

- Generate an execution plan with **terraform plan**.

```
$ terraform plan
```

- Review the plan to ensure it aligns with your desired infrastructure changes.

5.6 Apply Terraform Configuration

- Apply your Terraform configuration with **terraform apply**.

```
$ terraform apply
```

- Confirm the changes when prompted.

6. Jenkins Pipeline Execution Steps

Follow these steps to create and execute a Jenkins pipeline for deploying infrastructure:

6.1 Jenkins Configuration

- Set up Jenkins on your EC2 instance.
- Install the necessary plugins, including the AWS and Docker plugins.

6.2 Create a Jenkins Pipeline

- Create a new Jenkins pipeline job.
- Configure the pipeline job to connect to your Git repository.
- Use the provided Jenkins pipeline script for building Docker images, pushing to AWS ECR, and deploying to AWS EKS.
- Define environment variables for AWS credentials, region, ECR repository name, and Dockerfile path.

6.3 Execute the Jenkins Pipeline

- Trigger the Jenkins pipeline manually or set up a webhook for automatic execution.
- Monitor the Jenkins pipeline execution and view the build and deployment logs.

7. Conclusion

In this article, we explain all the steps to create an infrastructure using terraform and we explain how easy it is to create and destroy a service using terraform.