

**IaC runbook for ReactJS and RoRAPI application**

**Date: 18/11/2022**

**Document Control**

**Document Information**

|  |  |
| --- | --- |
|  | **Information** |
| Document ID | IaC runbook for ReactJS and RoRAPI application |
| Document Owner | Chakravarthi Thangavelu |
| Issue Date | 18/11/2022 |
| File Name | Project Iac ReactJS and RoRAPI |

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Owners** | **Version** | **Issue Date** | **Remarks** |
| Chakravarthi Thangavelu | Draft | 18/11/2022 | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Introduction**

Cloud Foundry is an [open source](https://en.wikipedia.org/wiki/Open-source_software), [multi-cloud](https://en.wikipedia.org/wiki/Cloud_computing) application [platform as a service](https://en.wikipedia.org/wiki/Platform_as_a_service) (PaaS) governed by the Cloud Foundry Foundation, a [501(c)(6)](https://en.wikipedia.org/wiki/501(c)_organization) organization

The software was originally developed by [VMware](https://en.wikipedia.org/wiki/VMware), transferred to [Pivotal Software](https://en.wikipedia.org/wiki/Pivotal_Software) (a joint venture by [EMC](https://en.wikipedia.org/wiki/EMC_Corporation), [VMware](https://en.wikipedia.org/wiki/VMware) and [General Electric](https://en.wikipedia.org/wiki/General_Electric)), who then transferred the software to the Cloud Foundry Foundation upon its inception in 2015.

**Document Objective**

Objective of this document is to explain the IaC project code deployment of the application backend written in RoRAPI and frontend in ReactJS. Part of this document, candidate need to submit the artifacts of the deployment and configuration

**Scope**

The Scope includes deploy IaC code to host provided application written RoRAPI and ReactJS.

The scope expansion.  Tool/cloud platform of our own choice

**Design Considerations**

As part of this assessment, I have decided to use this methodology to implement as pee the ask

1. Terraform – Deployment tool
2. AWS – Cloud platform choice
3. Ansible – Configuration tool
4. Docker – a choice of platform to have above tools

**Pre-requisites**

These are prerequisites needed to deploy in this solution

1. Laptop with Windows 11
2. Oracle Virtual box
3. Centos 18 is installed Virtual box
4. Docker software is installed on Centos OS
5. Created a container using Dockerfile

**Assumptions**

These are assumptions made for this solution to create IaC and Ansible configuration

1. Solution is public application architecture
2. This application need a high availability solution
3. Codes should be hosted on repository

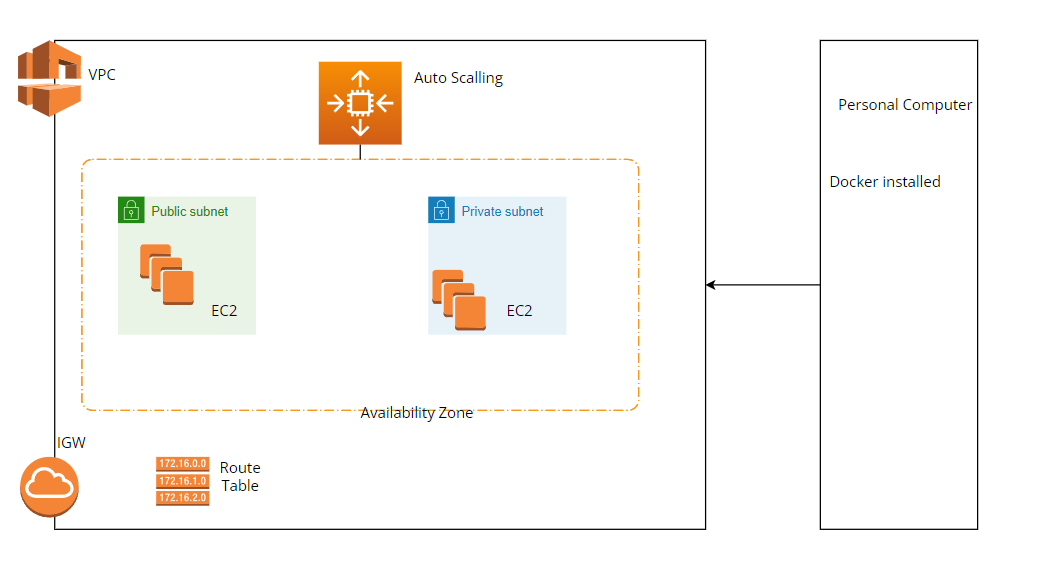
**Security**

These are the security are in place for this deployment

1. AWS security groups allows only port 80,443 and 22
2. There are two security groups in place, 1 for ec2 instances and another one for load balancer
3. We can further use NACL for subnets but I haven’t used in this solution
4. We can deploy postgres in private subnet and allow access only via security group. I was not sure how RoRAPI works with postgress and hence I haven’t considered this solution
5. Ansible host holds private key of ec2 instance and hence I have encrypted the hosts file using ansible-vault
6. All EC2 instance’s disks are encrypted as part of security

**Deployment Architecture**

Using Terraform IaC code and ansible configuration tool, and to have high available architecture, its decided to use Auto scalling behind load balancer as a multi AZ deployment. Here is the architecture diagram



**Implementation**

**Step1: Create Dockerfile with all DevOps tools**

**Step2: Execute terraform IaC on docker container created in step1**

**Step3: Execute Ansible code on docker container to install required**

**package and configuration**