Gorilla Logic
San Jose, Costa Rica

TimeOff Management

Oscar A. Mora

4th May 2019

OVERVIEW

As a devops engineer I want the given application [Timeoff management] running on a cloud provider and automatically deployed via autonomous methods.

GOALS

The following can be found here

1. Create architecture diagram of the solution
2. Fork on local repository of Version Control.
3. Select Cloud provider.
4. CI/CD triggered via commit within the Source Control.
5. Application must be secured from external access, serving HTTP & HTTPS protocols.
6. Application must be highly available and load balanced.

MILESTONES

Architecture Diagram of Solution

The architecture diagram of the solution consists of the utilization of various technologies such as:

- Amazon Web Services as the cloud provider of choice.
- Terraform to suffice the Infrastructure as code requirement (deployment of infrastructure within AWS).
- Jenkins for the Continuous Integration and continuous deployment.
- DockerHub to allocate our docker image for the application.

The solution diagram can be found here

Terraform

I have chosen to utilize terraform as the Infrastructure as code solution, because if there was a need to migrate to a different cloud provider this could be done easily. Terraform is cloud-agnostic as well as having integration with many other tools such as CI/CD. Also, terraform addresses the issue by utilizing an immutable infrastructure approach where every configuration change leads to a separate configuration snapshot which means deployment of a new server and de-provisioning the old one.

AWS

The cloud provider of choice for this solution was chosen based on the services it provides and how comfortable I am with this solution due to my experience. Nonetheless, this solution could also be replicated with Azure, GCP, etc. I was merely interest in using the Elastic container service for Kubernetes, which gives us the ability to create a cluster and have it be highly available. Also, lets not forget important services that provide ease such as Route 53 and ALBs, which in my point of view are much easier to provision and efficient due to the algorithms used that other cloud providers.

Jenkins

Whenever I can leverage an open source tool I will opt for it. Open source tools bring a lot of advantages, starting off by the community support and the ability to access the code and modify it at will, hence the reason for the use of jenkins in my solution. Jenkins is also platform independent, easily configurable and has a rich ecosystem.

DockerHub

DockerHub uses a very familiar collaboration model as GitHub, therefore being very easy to use, especially for GitHub users. It also has **security** scanning at no cost.

Security

IAM: Created IAM policies attached to roles. It is a common practice and not a secure one, to attach it to users. For the IAM user present (like myself) I have added multi factor authentication to create another layer of security.

For the account access a strong password policy is also enforced.

VPC: For this component I have also tighten the controls, ensuring the inbound and outbound rules are the minimum access we need.

Monitoring: For this solution we are also using grafana & prometheus to not only monitor the resources (health/Performance) but also to detect any unusual activity that could compromise the application. There various policy templates embedded into grafana as well as prometheus to do log parsing and alert based on conditions met.

Infrastructure: The infrastructure services that are not being utilized have been disabled as well as ports and protocols. This will avoid any attacks such as privilege escalation, binary exploitation, spoofing, etc.

It has been a pleasure to work on this project, if you have further questions or doubts feel free to reach me at **ao.mora@gmail.com**