A black and white logo

Description automatically generated with medium confidence

**DevOps document**

OFS Platform

By Mohammed Al Harbi

Contents

[Introduction 3](#_Toc121279351)

[What, Why and How DevOps? 3](#_Toc121279352)

[DevOps with serverless 3](#_Toc121279353)

[Current pipeline vs Potential pipeline 4](#_Toc121279354)

[1. Current pipeline: 4](#_Toc121279355)

[2. Potential pipeline: 5](#_Toc121279356)

# Introduction

In this document we will investigate the terms DevOps, why do we need it and how to perform DevOps. Then we will review DevOps in terms serverless and eventually show what the expected DevOps look like versus what we currently have.

# What, Why and How DevOps?

The DevOps terms comes from “Development operations" which aims to automates developing processes of core business logic services or functionalities. Automating development processes is commonly essential as businesses need to grow, more and more tools need to be integrated and thus more development processes need to be checked in a repetitive manner. Where repetitive manner is present is chances of automation raise and thus the automation of such manners is called DevOps. Typing we would need some servers where we can run those repetitive manners on some kind of schedule or triggers and this can typically be found in versioning applications such as GitHub or GitLab, this is also possible to do on vendors such as AWS. It is commonly known for developers to use versioning applications for such task and in my case OFS does DevOps in GitHub on a trigger where any update from my end gets committed.

# DevOps with serverless

DevOps as mentioned above are some repetitive actions but if we took a deeper look into those manners, they differ from a technology to another, from a programming language to another. By utilizing GitHub Actions, we can use a universal language called Yaml to configure them, but the configuration differs based on technology. When it comes to serverless the main part where developers need to automate a repetitive manner of is basically the continuous checking of vulnerabilities in serverless dependencies as well as deploying the infrastructure as a code configurations snippet along with all function as service codes to the cloud. Unlike typical non-serverless application, developers **do not** need to build an artifact and dockerizing that as cloud providers already take care of the dependencies, patching, installing OS of servers.

# Current pipeline vs Potential pipeline

# Current pipeline:

The following is the screenshot demonstrate Oman freelancing services platform in Github Actions:

Graphical user interface, application

Description automatically generated

The first step in the pipe is vulnerability check, in this part of the all the dependencies of microservices are scanned through static code analysis took named Snyk.

Text

Description automatically generated

The second step is a parallelization step where each microservice get to test its own unit tests, build the serverless stack artifact based on given infrastructure as service snippet and deploy it to AWS and the results are ready to use API gateway links associated with HTTPS configured endpoints. I deploy all microservices along with Cognito configurations for strong security measurements.

The third step in the pipeline is to test services integrated and the last step is to access the AWS CLI using the AWS secrets I setup using GitHub Secrets to deploy the complied frontend solution into an S3 Bucket. The Bucket is then configured with a Cloudfront instance where the content of that bucket is distributed globally over HTTPS network for security purposes and minimizing latency.

# Potential pipeline:

* The pipeline can add E2E testing prior to deploying to AWS.
* The pipeline can include 2-3 more microservices.
* The pipeline can include AI pricing estimation testing.
* The pipeline can include automated performance testing.

# Final pipeline:

The pipeline has been updated and it now supports E2E testin prior to deploying to AWS.

It also supports a paid version of sonarcloud for static code analysis. Pricing estimator service has been finalized, monitoring tool datadog has been added.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated Graphical user interface, application

Description automatically generated