Automated deployment pipeline for Mono-repo system

SESAPZG628T PROJECT WORK

by

Veena Sri Varadharajulu
2017HS70012

Project Work carried out at
SAP Labs Pvt Ltd , Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) India

June 2021

SESAPZG628T DISSERTATION

Automated deployment pipeline for Mono-repo system

Submitted in partial fulfillment of the requirements of the

M. Tech. Software Engineering Degree programme

By

Veena Sri Varadharajulu
2017HS70012

Under the supervision of

Sudhanva Laxminarayan
Senior Developer

Project work carried out at

SAP, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

(June 2021)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Project Work entitled **Automated Deployment pipeline for Mono-repo system** and submitted by **Veena Sri Varadharajulu** ID No. **2017HS70012** in partial fulfillment of the requirements of SESAP ZG628T Project Work, embodies the work done by him/her under my supervision.

Date: 09-06-2021

Signature of the Supervisor

Name: Sudhanva Laxminarayan

Designation: Senior Developer

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I would like to profoundly thank the **Management** of **BITS Pilani**, for providing such a healthy environment for the successful completion of the project work.

I would like to express my sincere thanks to my Bits Coordinator **Ms. Preeti NG**, for her constant guidance, support and constructive suggestions for the betterment of the project.

I wish to express my gratitude to my supervisor **Mr. Sudhanva Laxminarayan** and Examiner **Ms. Vidya Shetty,** for their support and endurance in visualizing and completing this project.

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
SESAPZG628T DISSERTATION

EIGHTH SEMESTER 2020-21

Project Work Title : Automated deployment pipeline for Mono-repo system

Name of Supervisor : Sudhanva Laxminarayan

Name of Student : Veena Sri Varadharajulu

BITS ID No. of Student : 2017HS70012

Abstract

A Jenkins job will trigger the Automated deployment pipeline 4 times a day, this deployment pipeline will take the responsibility to deploy all the component to the Production SCP environment. These deployment details will be stores the RDS database in AWS, data stored in this AWS region will be used for monitoring purpose.
We will have the deployment monitoring dashboards providing information about everyday deployment along with the regions to which the applications are deployed, the deployment status, etc.,

Keywords: AWS RDS, deployment, Grafana, Jenkins

Sudhanva Laxminarayan (Supervisor)

Veena Sri Varadharajulu (Student)

TABLE OF CONTENTS

List of Symbols & Abbreviations used

AWS: Amazon Web Service
SCP: SAP Cloud Platform
RDS: Relational Database System
IST: Indian Standard Time
SLA: Service Level Agreement

List of Figures

INTRODUCTION

AUTOMATED DEPLOYMENT PIPELINE FOR MONO-REPO SYSTEM

Project Objective

Automated deployment trigger pipeline will let a deployment be trigger 4 times a day starting from morning 9.30 AM IST – 9.30 PM IST with an interval of 4 hours between each deployment. This deployment pipeline will take the responsibility to deploy the new version of the components and bind required service instances , switch routes from the old version to the newly deployed version , removing service instance binding from the old components and delete these old version application.

After a successful deployment, the pipeline will run tests as a sanity check to validate if the newly deployed services are performing as expected , if not then appropriate alerting will be made.

Benefits of the project

- No manual effort will be needed to perform deployment
- When new production environment is added, this pipeline will be stable enough to make deployment to this new region
- Latest version of the components will be automatically detected and get deployed
- This pipeline will run sanity checks after each deployment to validate if the new versions of the application is functioning as performed

CHAPTER 1

PROBLEM STATEMENT

Manual deployment can be a complicated process, especially when there are lot of applications to deploy. In a growing organization where, new components get added very frequently, manual deployment is complex. The Mono-repo system for which we are building the automatic deployment will have nearly 54 components and each component will be need set of service instances to be bounded with them.

Manually binding services and then deploying the applications, along with the route switching from old version to new version, deleting the old deployed version , running sanity check after deploying all the applications from our Mono-repo system just to validate if the newly deployed version is functioning as expected is becoming a tedious task. This manual process if very time consuming and lot of resources will be needed.

Automated deployment pipeline will now help us get rid of all these complexions. We will maintain a Jenkins seed job which will act as trigger pipeline to deploy applications to our production environment 4 time a day. Also, we will integrate this pipeline with AWS RDS and store the deployment related data which will help us monitor about the deployments to different production regions and their deployment status.

So, this project will help us eliminate the manual deployment process and the gap in monitoring of the deployment becomes way easier.

CHAPTER 2

OBJECTIVE OF THE PROJECT

Automated deployment pipeline will trigger deployment to all the production environments 4 times a day starting from 9.30 AM – 9.30 PM IST, with an interval of 4 hours between each deployment. We will check if the new versions are performing as expected after each deployment, on failure we trigger appropriate notification.

This project aims to deploy new versions to the production environment and record deployment details for monitoring purpose. This will eliminate burdens in tracking the versions deployed, regions, etc.,

CHAPTER 3

UNIQUENESS OF THE PROJECT

In my current organization, there is no powerful pipeline which can perform the task of multiple deployments a day, tracking the deployed versions, etc.,
Also, the pipeline which we are building now will be dynamic enough that we define the list of apps to be deployed in each production environment.
Example: We can mention which application should be deployed in each production region.

- Production region 1 can have
  - App 1
  - App 2
  - App 3
- Production region 2 can have
  - App 1
  - App 2
  - App 4

CHAPTER 4

BENEFIT TO THE ORGANIZATION

- Delivery latest feature to the customer
- Easy roll back to previous version if needed
- New features get deployed every day, so fast delivery of features to customers
- Lesser deployment time (in comparison with the efforts need for manual deployment)

CHAPTER 5

SCOPE OF WORK

The purpose of the CICD tool is to address the need of continuous feature delivery for the cloud application, without the pressure of timelines/deadlines to the development teams. This project involves setting-up an end-to-end deployment pipeline which will make sure the services required for applications are created in the live regions .After deploying latest version of application , old components should be deleted and all the bindings made to those component must also be unbonded along with routes switching .We run tests in prod as a sanity check after each deployment.

These deployment data should be recorded in the our RDS in AWS region which will later be consumed by out Grafana monitoring dashboards.

CHAPTER 6

SOLUTION ARCHITECTURE

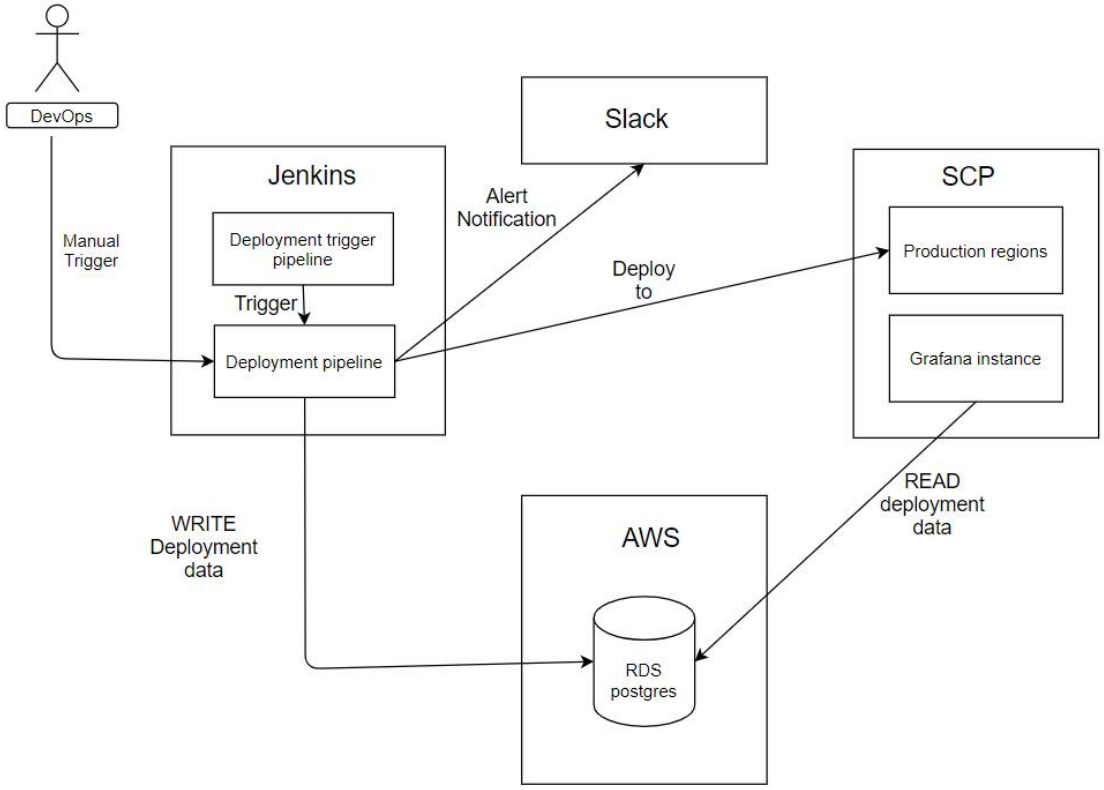


Figure 1: Business Process Flow

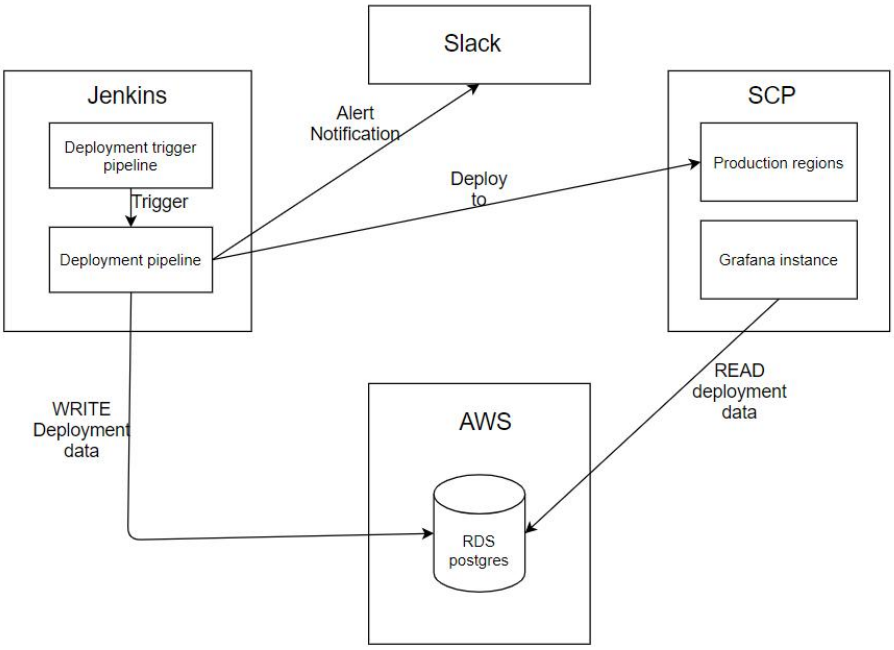


Figure 2: Jenkins integration

The above diagrams show how Jenkins is integrated with different components. Jenkins uses CF-CLI to login to the cf regions and performs the deployment actions. We use Slack webhooks to make notification in slack channel.

A webhook is a user-defined callback over HTTP. Slack webhooks can be used to notify about the deployment start, success/failure. A webhook in Jenkins is defined by the following information that you need to provide when creating a slack notification:

- Slack info message
- Production environment details
- Deployment Status
- Deployment commit version

Monitoring Dashboards will be created in the Grafana instance running in the SCP. The deployment data will be written to AWS RDS postgres and will be consumed by the dashboards in Grafana.

CHAPTER 7

WORK ACCOMPLISHED SO FAR

Jenkins Deployment trigger job

The seed job to trigger the deployment pipeline. The Bootstrap code to trigger the Deployment pipeline will be configured in the Jenkins job. This seed job is supposed to trigger the Deployment pipeline 4 times a day, this interval is set in the Jenkins job.

The deployment pipeline needs following parameters:

- **Latest deployable version**: will be picked from the GitHub
- **Deployment region**: The prod region to deploy
- **Action**: deploy/switch/test/clean
- **Alerting**: Notify about the deployment in Slack channel

Development Environment setup

CF deployment needs .yml file to deploy applications. This project supports dynamic deployment where multiple CF regions can have different applications deployed. So we maintain .yml files for different regions along with the routes, services to be bounded with the application. Service instance details and the plans will be maintained in the GitHub branch , we have tools written in NodeJS which will create the services mentioned .yml file in CF space.

Jenkins Deployment pipeline

Input: Will be supplied from the Deployment trigger job
The script for deployment is maintained in the GitHub master branch. The Jenkins job is configured to execute this script for the regions defined, the pipeline will execute the following steps:

- Create required service instances in the CF Space
- Deploy the applications
- Bind the service instance with the application
- Switch the routes for the applications
- Start the newly deployed application
- Stop the old running applications
- Sanity test to check if the newly deployed version is working as expected
- Delete the old versions of the applications from CF space

Alert Notification Service

When a deployment is triggered, completed successfully, deployment failed, deployment skipped

the information about this is send to a slack channel. In the Slack channel we will send the following information about a deployment,

- Message: The kind of activity initiated by the pipeline (deployment started/Deployment Failed, etc.,)
- Region: CF region for which the deployment is initiated
- Commit: The version for which the deployment activity is happening
- Error Message[optional]: In case of failure, the exception from pipeline is thrown

Store deployment metrics

We upload the activities of deployment in the postgres database located in AWS. We have a tool written in nodeJS for persisting this deployment data in the database.

Grafana Instance and Monitoring dashboard

Grafana instance is deployed to the CF space and we create deployment monitoring dashboards here. We have the data source created to read data from the AWS postgres database , then we chose the type of graphical display we want and then write queries to give us visual representation of the data for acquiring insights .

CHAPTER 8

KEY CHALLENGES

The current project depends on few pre-requisites which is something to be done manually and cannot be automated, following are few if the issue with/in capabilities in current pipeline and required manual intervention,

- The ideal approach should have retry mechanisms, on certain kind of failures, automatic retry should get enabled.

- After route switching if the application fails to start, then service will become un-operable, so if we don't revert quickly to the stable version then downtime will be long. This will affect the SLA's made.

- Deploying applications in CF will need few services to be bounded, our pipeline is capable for creating service instances in the CF given that appropriate entitlements are added to the particular CF Space. This entitlement is a manual process and cannot be automated as this needs Admin rights to increase are decrease the entitlements for a particular service.

CONCLUSION

The tech stack of this development includes JaaS, Groovy, nodeJS, Grafana, AWS RDS, SAP BTP, etc., The entire setup is made in way that triggering deployment should be simple , we can do manual trigger of deployment and also we have job for automatic deployment . Minimal Knowledge will be required for triggering a deployment.

This project for sure reduces the workload and time duration in deployment, in compare with the manual deployment this automated deployment have saved lot of resources. Understanding the architecture proposed for this solution needs little bit of background in different area like, SAP BTP, AWS, etc.,

This project touched on different areas like Cloud (SAP BTP, AWS RDS), open-source application (Grafana) ,etc.,

The project entitled “Automated Deployment pipeline for Mono-Repo system” had been successfully completed and satisfies all the requirements specified. The system provides live data to organization and help in decision making without wastage of time.

DIRECTION FOR FUTURE WORK

The system is flexible enough to accommodate any further functionality. In future we must focus on enabling a retry mechanism based on the failure. Currently this is not implemented, so when certain failures occur the deployment is aborted, and then manual interference will be needed.

Instead we can improvise the current structure and automate to re-trigger deployment based on failure cases. But this will need some good amount of time for feature to be live , then track the cases manually and alter of pipeline to handle those cases and perform retry.

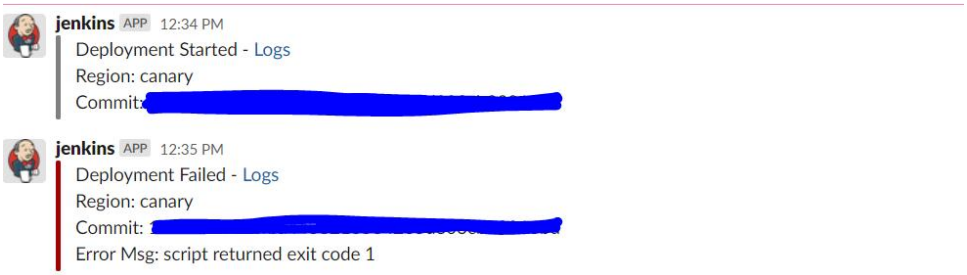
APPENDIX

A. INPUT

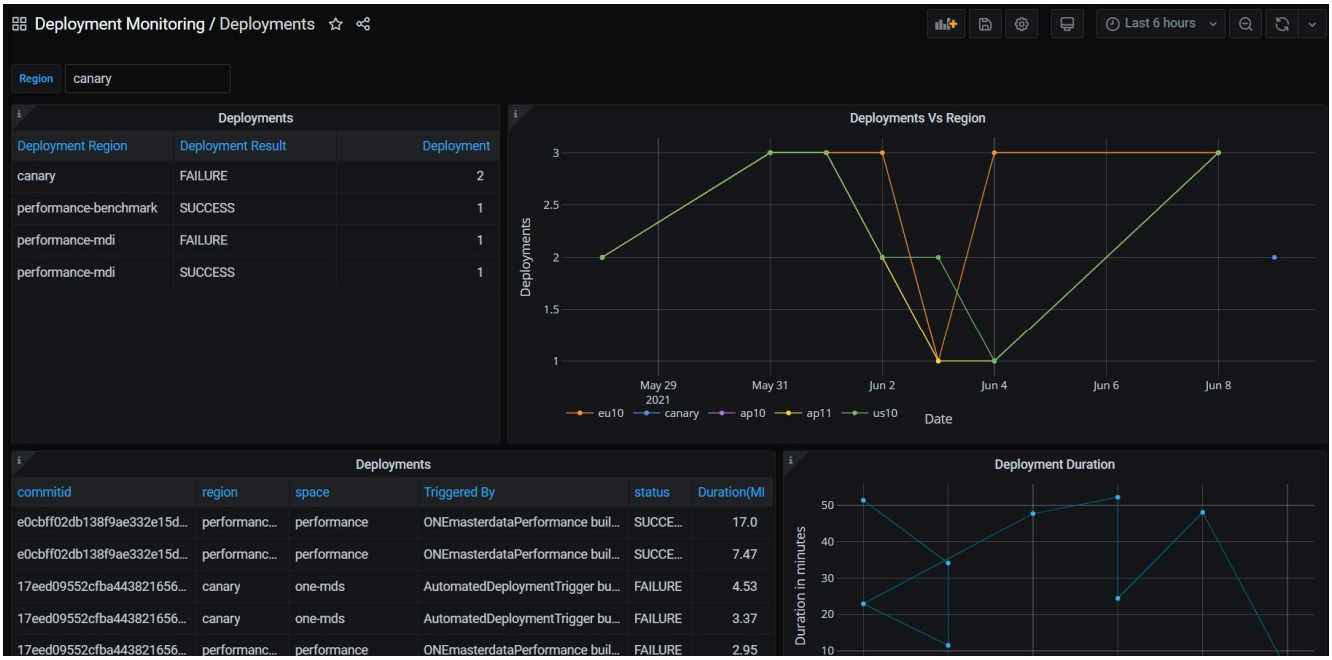
MANUALLY TRIGGERING DEPLOYMENT

B. OUTPUT

SLACK NOTIFICATION ABOUT DEPLOYMENTS



GRAFANA MONITORING DASHBOARD



REFERENCES

About Jenkins: https://www.jenkins.io/

Slack webhook for sending notifications: https://api.slack.com/messaging/webhooks

Connecting to AWS RDS postgres: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToPostgreSQLInstance.html

About Grafana: https://grafana.com/docs/grafana/latest/installation/

CF-CLI: https://docs.cloudfoundry.org/cf-cli/

CHEKLIST

Detailed Plan of Work (for 16 weeks)

Serial Number of Task	**Tasks or subtasks to be done**	**Planned duration in weeks**	**Deliverable**	**Break-down of task**	**Status**
1.	Jenkins job to trigger the deployment pipeline in mentioned interval of time	3	Jenkins job which can trigger another pipeline for a defined interval		Completed
2.	Set-up deployment environment	3	Files with services and components required for different production environment	- Setup yaml files for deployment 1.Production regions (**1 week)** 2.Development/test regions (**1 week)** - Dynamic service creation yaml files (**1 week)**	Completed
3.	Deployment pipeline	4	Pipeline which can deploy applications and services to production regions	1.Category of Actions to performed (push/deploy/switch/clean/test) (**2 weeks)** 2.Scripts to perform the actions defined during deployment (**2 weeks)**	Completed
4	Alert Notification	1	Send deployment notifications to Slack channel		Completed
5	Store deployment metrics	2	Writing data to the postgres database		Completed

6	Grafana – for monitoring dashboards	1	A running Grafana instance		Completed
7	Monitoring dashboards	2	Dashboards on Grafana showing deployment related data’s		Completed

a) Is the Cover page in proper format? Y / N
b) Is the Title page in proper format? Y / N
c) Is the Certificate from the Supervisor in proper format? Has it been signed? Y / N
d) Is Abstract included in the Report? Is it properly written? Y / N
e) Does the Table of Contents' page include chapter page numbers? Y / N
f) Is Introduction included in the report? Is it properly written? Y/N
g) Are the Pages numbered properly? Y / N
h) Are the Figures numbered properly? Y / N
i) Are the Tables numbered properly? Y / N
j) Are the Captions for the Figures and Tables proper? Y / N
k) Are the Appendices numbered? Y / N
l) Does the Report have Conclusions/ Recommendations of the work? Y / N
m) Are References/ Bibliography given in the Report? Y / N
n) Have the References been cited in the Report? Y / N
o) Is the citation of References/ Bibliography in proper format? Y / N