LOG8415 Final Project

Automated website deployment using S3 static pages

Polytechnique de Montréal

Semester: Fall 2021

Estefan Vega-Calcada (1934346)

Advanced Concepts in Cloud Computing

December 20, 2021

Introduction

For my final project of the “Advanced Concepts of Cloud Computing” class, I chose to automate the deployment of a simple HTML web page using the various services offered on AWS. In this report, I will explain how I successfully used an S3 bucket, a pipeline to do so and other services to do so.

The following is divided into seven sections: the issue (2) discusses the reasons why this project exists. What is its purpose, what are we trying to solve? Section 3, literature review describes a short paper and reviews its relevance to the project. Methodology (section 4) gives a general idea of the solution implemented to solve the issue. Architecture (section 5) describes the general architecture of the project. AWS Services (section 6) describes each cloud service used in the project. The “Data Flow and Methodology” section presents a more visual view of the project with a small graph and explains how every component of the project is related together. Section 8, Demonstration, contains the necessary information to view the code and the video demonstration of the project. Finally, Future Directions (section 9), talks about some improvements that could be implemented in the future for this project

2- The Issue

The issue we are trying to solve in this project is the repetitiveness of certain tasks when trying to host a website online. The goal is to allow automatic deployment of new source code added or modified in a git repository.

3- Literature Review

The paper IOS CI/CD Build and Test Pipelineon AWS [1] is a simple chart that presents the different steps involved in the pipeline used to build and test iOS applications. The literature focuses on a more visual approach to explain the general flow of the pipeline. There are seven steps to the deployment of the code: first the AWS CodePipeline is called when new code is detected on AWS CodeCommit (kind of the equivalent of a Git Repository). Then when the changes are detected, the build process is initiated. Lambda Functions are used to load the required build scripts from an S3 Bucket and trigger the AWS Systems Manager to run the build scripts on an EC2 instance. The logs are then sent to Amazon CloudWatch for monitoring. Finally, AWS DEvice Farm is used for testing the app on real devices and generates the test results, logs and recordings in the console.
This article has a lot of common points with the current projects which was very interesting for me to see the usefulness of the AWS CodePipeline service

4- Proposed Solution

The approach taken in this project to solve the issue was to use a pipeline. It is a service offered on the AWS platform that allows users to generate code and deploy it to the world in a very short amount of time. It is quite easy to set up once you get an understanding of the platform itself and can automate all the boring tasks we discussed earlier. The service “CodePipeline” was used with an S3 Bucket that stored all the source code. Each time a user commits new code to his repository, the pipeline automatically runs again, which reuploads the code on the web. This is the magic of it all. Let’s see how it works in the following section

5- Architecture

This project requires it’s users to have AWS CLI installed on their machine. It is used to interact with the AWS platform via the command line.

The main components of this project are :

- Two S3 Buckets (S3Buckets)
- Pipeline (CodePipeline)
- GitHub Repository

The first bucket contains all the source code that was collected from the Github Repository and the second bucket is used to collect the artifacts generated by the different pipeline stages. The Pipeline is responsible for the updates of the website code.

The pipeline itself is a two stage pipeline:

1. The “Source” stage. This stage is responsible for linking the pipeline with the GitHub code repository that contains all the web application code. It also triggers an update and

is responsible for notifying the CodePipeline service whenever a change is detected on the repository. Whenever a change is detected the whole pipeline runs again. This is what allows the website to constantly be updated. Once this stage is completed, the artifacts generated as output are used as input in the following pipeline stage.

2. The “Deploy” stage. This stage is the second and last stage of the pipeline. It takes the source code and uploads it on S3 by overriding the previous successful build. The input artifacts for this stage are the output artifacts from the “Source” stage.

The architecture of this pipeline isn’t ideal. It is pretty fragile because there is no building stage used to verify the code.

6- AWS Services

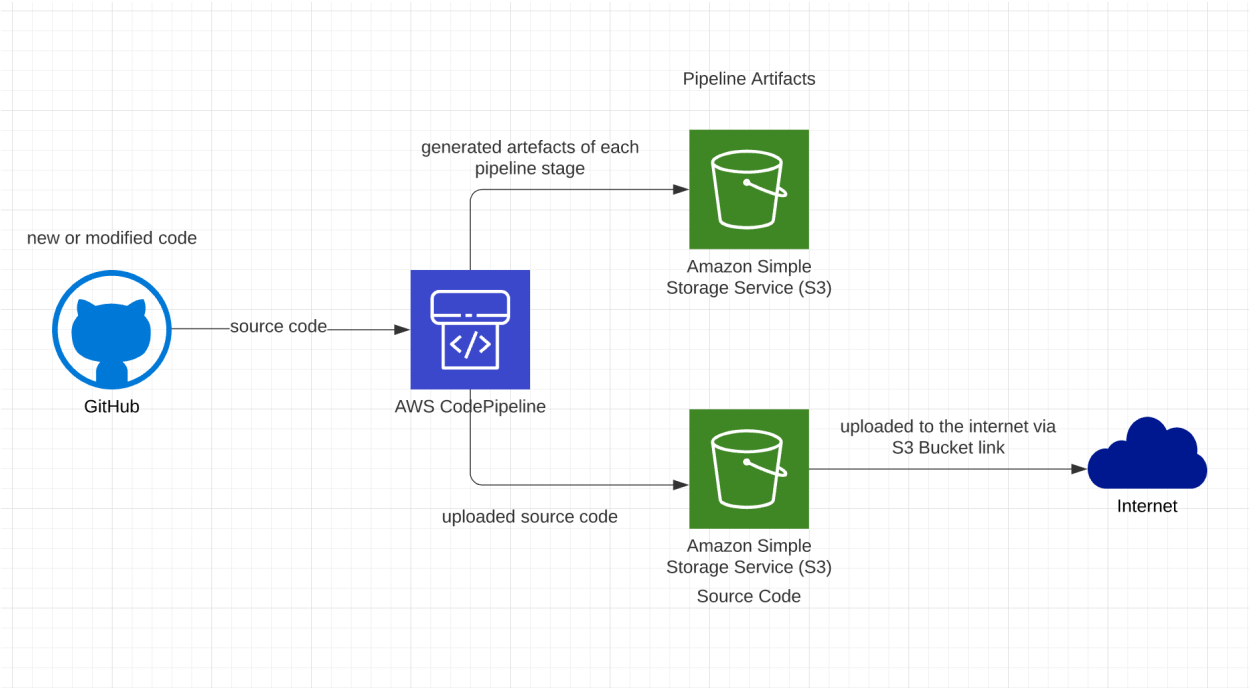
The main services used in this project were:

- Amazon S3 Bucket
- Amazon CodePipeline

Amazon S3: Service used to store objects. I used S3 Buckets to store
the source code from Github, deploy the source code and store the artifacts of the different pipeline stages.
Amazon CodePipeline: This is the main service used to create a pipeline that automates the deployment of my source code. I created a two stage pipeline “Source” & “Deploy” that automatically runs when it detects a new change on the Github Repository.

7- Data Flow and Methodology

The figure down below shows a general look of the architecture implemented in this project. we can see that it all starts from Github. Github contains all the source code. When the developer pushes new code to his branch, the AWS CodePipeline is configured to keep track of the “main” branch. When it detects some changes, the Pipeline runs and redeploys the code to the S3 bucket. The S3 Bucket is in charge of containing the code deployed to the website or to the internet and that is how the website is made accessible to everyone automatically



To implement this architecture, I used a bash script. The bash script named “launch.sh” is in charge of using the AWS API to create and use the multiple services offered by AWS. The first services created are two S3 buckets. The first bucket will be the one containing the source code. The second bucket contains the artefacts created by the execution of the pipeline. I then proceeded to create a some po

More details on the script will be available in the video demonstration

8- Demonstration

- The link for the youtube demonstration is: PROVIDED ON MOODLE
- The link for the git repo that contains all the code is: https://github.com/faynz/finalproject-log8415
- The requirements for this projects are : AWS CLI

To run the code, you simply have to run the bash script named: "launch.sh"

9- Future Directions

Unfortunately, this assignment wasn't fully completed. I was missing on the EC2 automation part. It would be interesting to add a deployment on an EC2 Instance. It could also be interesting to add a building stage to the pipeline to verify that the source code is correct. Finally, the website itself could be used for something more interesting than a simple html page, but for the purposes of this demonstration I kept it simple.

Conclusion

In conclusion, I am pretty disappointed in the outcome of this final project. I wish I had more time to complete the full project and get to learn how to automate website deployment on EC2 instances as well. I will probably look into it for myself. The project was very beneficial, I learned a lot about the available services offered by AWS and the platform itself. It was quite interesting to see how quickly and efficient it is. It allows for very quick updates.

References

https://medium.com/@kyle.galbraith/how-to-host-a-website-on-s3-without-getting-lost-in-the-sea-e2b82aa6cd83

https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-s3deploy.html

https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/automate-static-website-deployment-to-amazon-s3.html

https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html

https://gist.github.com/shortjared/4c1e3fe52bdfa47522cfe5b41e5d6f22

https://docs.aws.amazon.com/cli/latest/reference/sts/get-caller-identity.html

https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html

https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html#reference-action-artifacts

https://docs.aws.amazon.com/codepipeline/latest/userguide/appendix-github-oauth.html#action-reference-GitHub

https://docs.aws.amazon.com/cli/latest/reference/codepipeline/create-pipeline.html

Annex

[1] Amazon Web Services. (2021). IOS CI/CD Build and Test Pipelineon AWS.
[Online] Available:
https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/ios-cicd-build-test-pipeline-ra.pdf?did=wp_card&trk=wp_card