LOG8415
Advanced Concepts of Cloud Computing
Individual Project

Chun-An Bau
2165883
chun-an.bau@polymtl.ca

Dec 17th, 2021

1 Introduction

1.1 Problem

Most of the services we use every day are deployed on the cloud. These services usually need to update their version periodically to remove bugs or release new features. Therefore, how to complete the task without failures in the shortest amount of downtime becomes a problem. If we temporarily suspended the service, attentively deployed and tested the new version, it indeed could eliminate most of the potential risks. However, the downtime might disappoint the customers. On the other hand, if we deployed the new version directly, we could minimize the downtime but might come up with a high risk of failure.

1.2 Literature Review

In this section, I would introduce two well-known deployment strategies, Blue-Green and Canary.

Blue-Green Deployment [7] runs two identical production environments called Blue and Green, and only one of the environments is live at any time. While releasing a new version, deployment and the final stage of testing will take place in the idle environment. Once the idle environment has been fully tested, the developers switch the router so all incoming requests now go to the new environment and the previous operating environment is idle now.

Canary Deployment [5] is a pattern for rolling out releases to partial users or servers. It first deploys the change to a small subset of servers and rolls the change out to the remaining servers if the tests succeed. This strategy can minimize the risk and impact of deployment failures.

1.3 Proposed Solution

In this project, I proposed a solution by using the Blue-Green Deployment Strategy. The service was deployed on two identical environments, named blue and green instances. The customers interacted with a proxy-liked server, that would control which instance should the customers connect to. Since only one instance would provide service at anytime, we could deploy the new version on the other instance, and switched the operating instance if there was no issue for the new version.

2 Architecture

The architecture diagram was illustrated in Fig 1. The users connected to the proxy server, which would redirect to the blue instance initially.

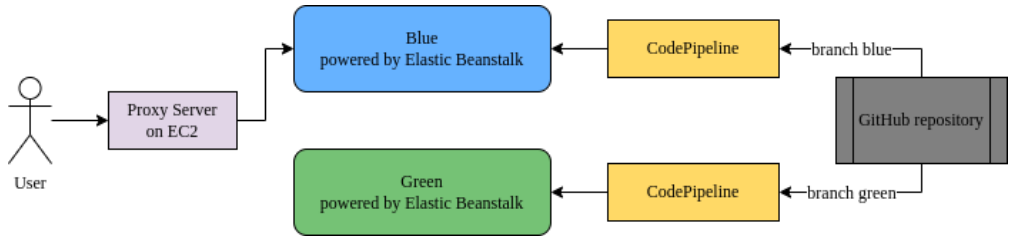

Figure 1: Architecture Diagram

If the developers would like to release a new version, they pushed the commit to branch *green* and the *CodePipeline* would automatically deploy the commit on the green instance. If there's no problem after testing, the developers could modify the proxy server configuration to redirect the traffic to the green instance. The result was shown in Fig 2.

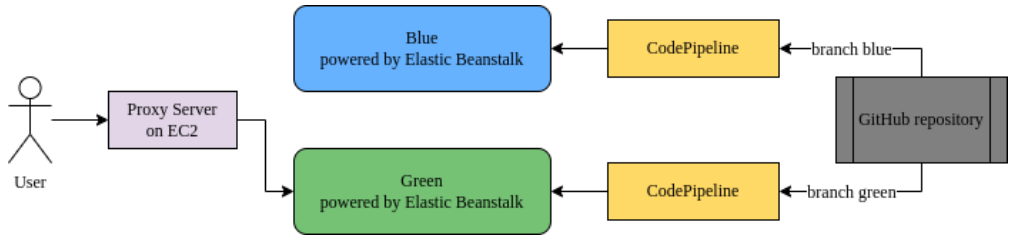

Figure 2: Architecture Diagram after New Deployment

3 AWS Services

Several AWS services were involved in the project.

- Elastic Beanstalk [4]: Build an execution environment for particular platforms, such as nodejs in my project.
- CodePipeline [1]: Connect with GitHub repository, deploy new version if there's commit submitted.
- EC2 [3]: Run the proxy server.

4 Data Flow and Methodology

Following the AWS CodePipeline tutorial [2], the building procedure of Elastic Beanstalk and CodePipeline was quite smooth and straightforward. However, I tried several approaches to complete the proxy server part.

- Route 53: I've heard of this functionality for a long time. It provided a DNS and could redirect the traffic to a specific EC2 instance or load balancer. Yet, the student account had no permission to access it.
- Load balancer rules: Each Elastic Beanstalk would create its own load balancer and target group. Per the experience in TP1, I knew that the traffic could be directed to a specific target group by modifying the load balancer configuration. However, I couldn't add the target group from green into the load balancer of blue.
- Switch environment url: Elastic Beanstalk provided functionality to swap the url of two environments. But when I attempted to do the operation, it showed that there weren't two or more environments in the ready state.
- Proxy server on EC2: Finally, I decided to write a simple Flask program to redirect the traffic to the current operating Elastic Beanstalk.

5 Instructions for Demo

I found an interesting open source project, musician-app [6], which was a simple project written in nodejs. I created two git branches, blue and green, and deployed them on two Elastic Beanstalk instances. After that, I executed a simple Flask script on an EC2 instance to redirect the input request to the currently operating instance.

```
app = Flask(__name__)

@app.route('/')
def rule():
    return redirect(sys.argv[1], code = 302)

app.run(host='0.0.0.0', port = 5000)
```

To demonstrate the system ability, I intentionally made a typo in the current operating version (blue instance). Next, I fixed the typo on git branch green, and it would automatically deploy on green instance. If there wasn't any failure on the green instance, the proxy server started to direct input traffic to the green instance.

6 Summary

In this project, I made a simple practice of automatic deployment on AWS, which used the blue-green deployment strategy. Some AWS functionalities were leveraged to complete the task, including Elastic Beanstalk, CodePipeline, and EC2 instances. The practice helped me to understand the concept of automatic deployment and some basic operations. For future works, the automatically testing, such as unit testing, could also be involved in the procedure.

References

[1] *AWS CodePipeline*. URL: https://aws.amazon.com/codepipeline/.

[2] *AWS CodePipeline tutorial — Build a CI/CD Pipeline on AWS*. URL: https://www.youtube.com/watch?v=NwzJCSPSPZs.

[3] *AWS EC2*. URL: https://aws.amazon.com/ec2/.

[4] *AWS Elastic Beanstalk*. URL: https://aws.amazon.com/elasticbeanstalk/.

[5] *Canary deployments*. URL: https://octopus.com/docs/deployments/patterns/canary-deployments.

[6] *GitHub Repo of misician-app*. URL: https://github.com/jspruance/musician-app.

[7] *Using Blue-Green Deployment to Reduce Downtime and Risk*. URL: https://docs.cloudfoundry.org/devguide/deploy-apps/blue-green.html.