Laboratory Practice Report

Version control (CodeCommit) and continuous deployment (CI/CD)

April 10th, 2024

 Departamento de Electrónica, Sistemas e Informática (DESI)

Cloud Architecture *(Arquitectura en la Nube)*

Mtro. Rodolfo Luthe Ríos

Erick Roman Ramos Rocha

746797

Computer System

# Introduction

Code versioning is a method that has been implemented in almost all organizations due to it is a simple way to revert changes in several processes. Talking about applications, it is a method to have excellent control in the code developed by dev teams; it helps to have a good control about the activities made by each team member. Also, in case of any failure or code released accidently, it can be reverted easily and no impacting other areas in the application. The same thing applies to any document or process that is documented, it can be saved as version, in case, something is lost or modified by mistake, it can be recovered faster.

Deployments are complicated when they are handled manually because it can come with human mistakes. For that reason, it is so important to automate them, it can be scheduled on specific date and time, or it can schedule each time that any code is added to a specific folder. In this laboratory, deployment is going to be scheduled now it is merged to GitHub on master branch, Code Pipeline is going to be triggered to deploy our application in elastic bean.

# Theoretical Framework

Version control (VC) [1] is the practice to of tracking and managing changes to software code, documents, processes, etc. It is regular used by third party tools that help the companies to have more control under the important assets. In addition, it helps to the teams to manage and achieve the main goal that is to successfully complete an activity that everyone is participating and avoiding mistakes providing a tool to go back on specific point or version.

Git bash [2] is a free and open-source distributed version control system. It can be used by anyone without additional cost.

GitHub [3] is a free version control system. It is one of the most popular tools implemented by users and organizations, it can have public and private repositories to be created. It has an option to have an enterprise account to add more options to do a version control easily.

CodeCommit [4] is a private version control system provided by AWS. As part of AWS, it provides security and highly scalable. It is based on git open-source version.

Repository [5] is a place where you can store your code, files, and each file revision. It helps to have everything in one place providing versions of each modification made on each file.

# Architectural diagram

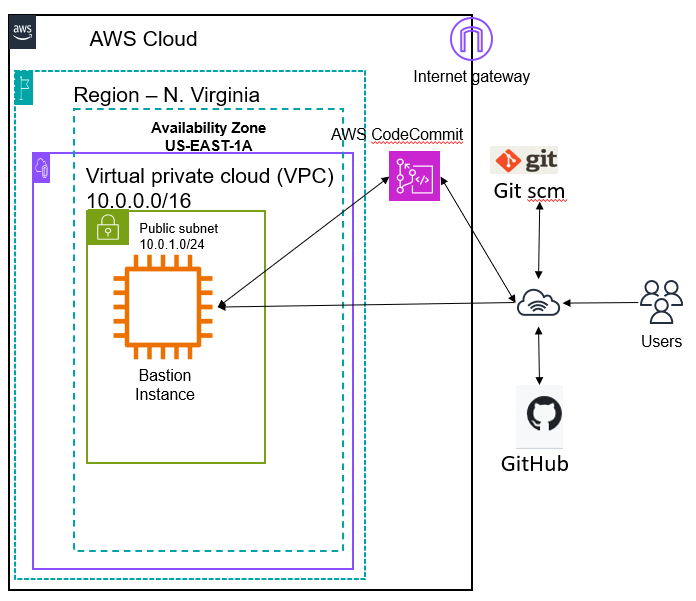


Figure 1 Diagram Control Versioning.

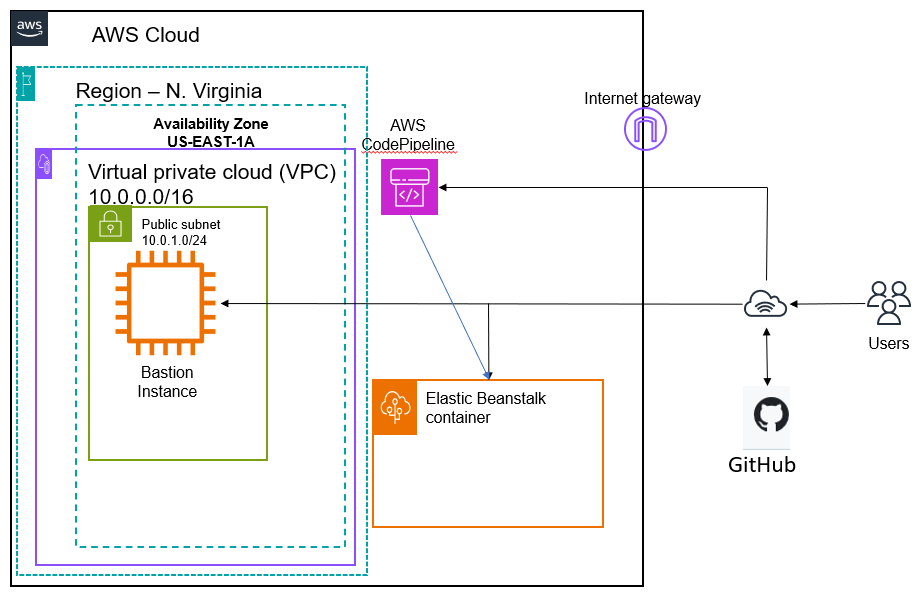


Figure 2 Diagram Continuous deployments.

# Practice Development

AWS Academy Account is going to be used to do source control file version implementing CodeCommit.

## Version control

### Set up local repository on git bash.

Following the guideline in the notebook, git must be installed, and account must be created to be able to do the file control version.

* Bastion windows is used to complete these steps (Figure 3).
* Iteso account is set in git (Figure 4).
* Setting up repository (Figure 5).
* Creating new file and pushing commit to git (Figure 6).
* Editing existing file and committing it again (Figure 7).
* Creating third version of the file (Figure 8).
* Reviewing log history to validate all changes are registered (Figure 9).

### GitHub repository.

* Creating GitHub account using iteso email and repository.
* Pushing to master current changes in the version.txt file (Figure 10).
* **What change do you see?** Last file version was pushed to GitHub (Figure 11); In addition, credentials were requested when push command was executed.

### CodeCommit repository.

* Repository is created.
* Push to CodeCommit is not working. Also, personal account was used but, same error message is thrown (Figure 12).

### Control version of a document.

## Continuous deployment environment.

* Go to Elastic Beanstalk service.
* Create new application. It is the same as previous laboratory where green/blue was implemented (Figure 13).
* Uploading Node.js application to GitHub (Figure 14 and Figure 15).
* Go to CodePipeline service.
* Create new pipeline following the instructions in the notebook.
* GitHub V1 must be used, as V2 is not authorized to AWS Academy.
* New repository must be selected (AN-cicd-lab) along master branch, as there is no other branch created.
* CodePipeline created successfully in personal account (Figure 16 and Figure 17).
* After editing index.html file and committed it to git, new deployment ran automatically (Figure 19).

# Problems and Solutions

List all the problems encountered during the development of the practice and how they were resolved.

# Experiments and Results

This section should explain each of the experiments conducted and their results. The results should be properly interpreted to understand their significance and why they are considered good or bad. Diagrams, photographs, or images can be included in this section to help represent the results.

# Cost analysis

Explain the cost of the implemented solution, justifying the chosen solution based on costs. Should detail monthly and annual costs.

# Conclusions

The conclusions should be a reflective work presenting the knowledge gained from the experiments, results, and the theoretical framework presented. Invalid conclusions include: "I learned a lot!", "I really liked the practice", "everything worked correctly." (1 or 2 paragraphs)

# Bibliography

In IEEE format.