

# Laboratory Practice Report

## Practice 9

November 1, 2023

Computer Systems Engineering

*Cloud Architecture*

Prof. M.S. Rodolfo Luthe Ríos

Marco Ricardo Cordero Hernández

is727272@iteso.mx



**ITESO**

Universidad Jesuita  
de Guadalajara

## Abstract

This document has the goal of providing a general view on a new take of code integration inside automated deploy processes. This it's made to contribute to the CI/CD implantation inside software development.

Also, use of preexisting technologies and tools will be demonstrated to show how relatively new services and cloud can be seamlessly integrated within the workflow of a software product, giving developer teams the ability of using the preset processes combined with new artifacts to make further progress towards remote architecture.

The added benefit of what's about to be implemented it's that previously seen cloud services will be used to show how several families of technologies and core services can be interlaced to provide a more complete and not so complex infrastructure coexisting inside the same cloud provider environment. This, in real production environments, can lead to faster problem solving and issue fixing, as common errors can arise, and because of the similar background in which these are found, they can be eradicated in record times.

## State of the Art

Back in the first half of the past century, the arrival of computer sciences as a new scientific field brought with it multiple challenges, improvement areas, questions and a deep desire of technology advancement. Back then, even a simple megabyte of information was beyond the most brilliant minds of that time, but, perhaps not for the father of computers: Alan Turing [1]. Turing envisioned a machine, called "a-machine" (now known as Turing machine), a computational model which operates through an infinite *memory* tape, modifying its contents to realize an action based on a finite set of actions. This simple premise lead to modern computers, but, setting aside the fascinating fact, Turing already implemented some sort of theoretical storage (although infinite).

Eventually, more feasible storage means and devices came to the light, provisioning computers with several methods of preserving programs or general data. Thinking about punch cards, one primitive way of storing computer data through physical paper cards with holes punched unto them [2] (even predating computers as a mere concept), this format couldn't be more restrictive, as a single mistake in their creation could result in a whole different meaning for its contents, perhaps an incorrect one. But even when correct punching was made, surely new upgrades needed to be introduced; this could've led to distinct scenarios, such as taking the previous punch card and punching new holes for a new iteration of the previously stored data, or, altogether copying into a new card the previous data and then adding new information. Cards are such an inconvenient way of storage device, but even so, the first modern version control tool was introduced for them: *IEBUPDTE* [3]. Introduced back in the 1960's, this technology was capable of using punch cards to create and update code libraries, similarly to contemporary patching systems.

A concept that was mentioned in the last paragraph stands out: *version control*. Also known as source control, this concept references the practice of tracking and managing changes to software code [4], however, the reality it's that any file can be tracked through version control and version control systems. In the 1970's, vcs's gained a lot of popularity caused by the ability of creating,

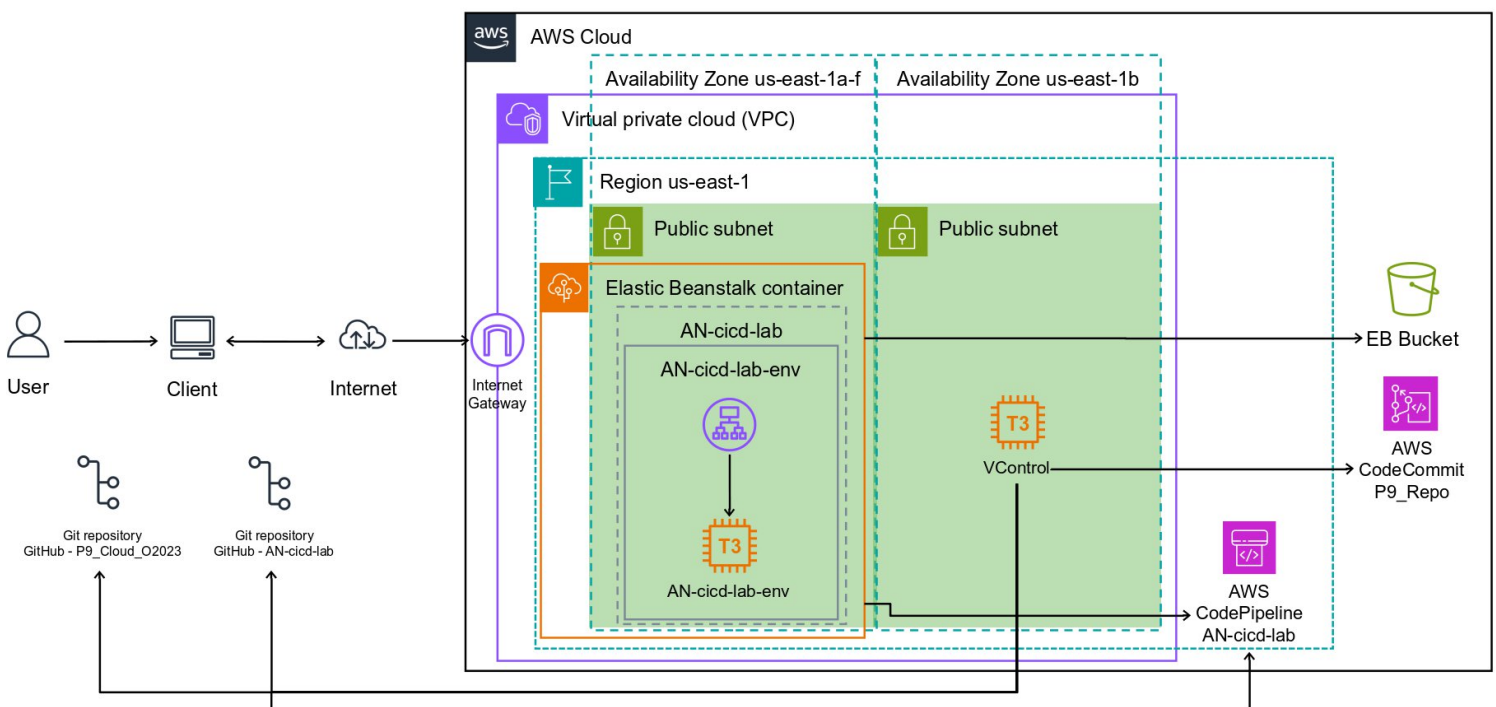
editing and tracking changes to files made inside computers. Although these tools can now also be considered primitive, the principles that constituted them are now still valid for modern systems.

With remote stored resources and the first rudimentary implementations of cloud services, new platforms for code collaboration and contribution began to rise as the default mean for non-physical team developing. A few years back, in 1991, Grady Booch would be introducing the “Continuous integration” concept [5], in which multiple working codebases were merged into the main branch of a software. Then, “Continuous deployment” term would come along, in which new iterations of the same software would be rolled out frequently in order to try to provide the newest version of it. Nowadays, CI/CD it’s becoming the new standard for production environments inside enterprises with well defined automated pipelines; this last term is defined as series of steps that must be performed in order to deliver a new version of software [6].

In this practice, the use of version control systems, along with CI/CD pipelines, will be demonstrated in order to provide a general sight of how modern applications can be delivered and upgraded to provide users with recently upgraded, remote stored source code applications.

## Diagram

The following architecture it’s proposed as a graphic solution for the stated goals.



## Practice Development

To start the demonstration, an AWS EC2 [7] instance will be created with the following specifications (this step can be omitted and a local machine can be used):

- Name: VControl
- OS: Amazon Linux 2023 or newest
- Architecture: 64-bit x86
- Instance type: t3.small
- Key pair: vockey
- Network settings
  - VPC: AN-VPC
  - Subnet: an-subnet-public1-us-east-1b (or any public subnet)
  - Auto-assign public IP
  - Default security group or any other that accepts all connections
- Storage: 8GB (or minimum) gp2

The distribution that Amazon Linux uses comes with the `dnf` [8] package manager, and it'll be used to install *git*, a distributed version control system [9]. The following commands will install and configure the aforementioned utility with personal configurations:

```
sudo dnf install git
git --version
git config --global user.name "Marco Cordero"
git config --global user.email "is727272@iteso.mx"
```

After installation and configuring it's done, git presence can also be checked.

```
[ec2-user@ip-10-0-2-251 ~]$ git --version
git version 2.40.1
```

### Local repository creation and usage

Inside the default path (root; `~`), a new directory has to be created. In this case, the name of it doesn't matter but for this demonstration, it'll be called "git". After that, a new repository has to be initialized. For all mentioned, these are the commands:

```
mkdir ~/git && cd ~/git
git init
```

```
[ec2-user@ip-10-0-2-251 ~]$ mkdir ~/git && cd ~/git
git init
Initialized empty Git repository in /home/ec2-user/git/.git/
```

Once the repository has been successfully created, a placeholder file inside it will be placed with arbitrary contents. Then, the file has to be added to the repository tracked files; consequently, a commit action to save the file's version will be made:

```
echo Version 1 > versions.txt
git add -A # Add all created and modified files
git commit -m "First version" -m "Version 1"
```

```
[ec2-user@ip-10-0-2-251 git]$ echo Version 1 > versions.txt
[ec2-user@ip-10-0-2-251 git]$ git add -A
[ec2-user@ip-10-0-2-251 git]$ git commit -m "First version" -m "Version 1"
[main (root-commit) 9316732] First version
1 file changed, 1 insertion(+)
create mode 100644 versions.txt
```

Modify and commit the same file with two new versions:

```
[ec2-user@ip-10-0-2-251 git]$ echo Version 2 > versions.txt
[ec2-user@ip-10-0-2-251 git]$ git commit -a -m "Second version" -m "Version 2"
[main c316d98] Second version
1 file changed, 1 insertion(+), 1 deletion(-)
[ec2-user@ip-10-0-2-251 git]$ echo Version 3 > versions.txt
[ec2-user@ip-10-0-2-251 git]$ git commit -a -m "Third version" -m "Version 3"
[main 060650d] Third version
1 file changed, 1 insertion(+), 1 deletion(-)
```

The following command shows the version history:

```
git log
```

```
commit 060650da6b4f82b0ea215fe76437e68831e93fde (HEAD -> main)
Author: Marco Cordero <is727272@iteso.mx>
Date: Sun Oct 29 23:41:01 2023 +0000

    Third version

    Version 3

commit c316d984111489044f070df65e74349953bc759b
Author: Marco Cordero <is727272@iteso.mx>
Date: Sun Oct 29 23:40:26 2023 +0000

    Second version

    Version 2

commit 9316732502b9ad6475e2931099cb22e5bfd634a9
Author: Marco Cordero <is727272@iteso.mx>
Date: Sun Oct 29 23:38:13 2023 +0000

    First version

    Version 1
```

## Remote repository

One of the main benefits of repositories is its remote collaboration when a repository hosting service (can be seen as cloud storage) is present. For this development, GitHub [10] will be used; although it's the more popular remote repository manager, it's not the only option available.

This platform requires an account in order to access most of its features, fundamentally, repository related actions. The process of account creation is skipped in this demonstration, however, the mail used for local repository should also be used for the newly created GitHub account; this is not necessary for offered services usage, but it keeps consistence.

Once the account has been properly setup, a new repository will be created:

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \* Marco385 / Repository name \* P9\_Cloud\_O2023

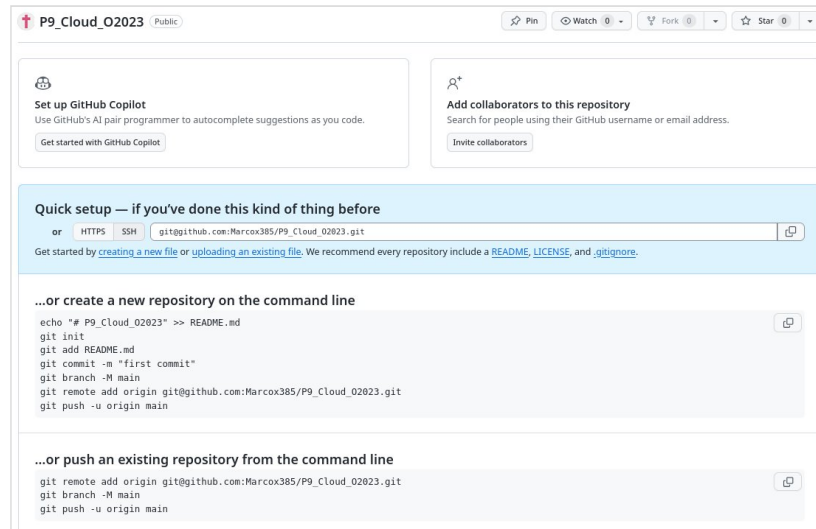
P9\_Cloud\_O2023 is available.

Great repository names are short and memorable. Need inspiration? How about [redesigned-giggle](#)?

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

If the repository it's successfully created, the following screen will be prompted:

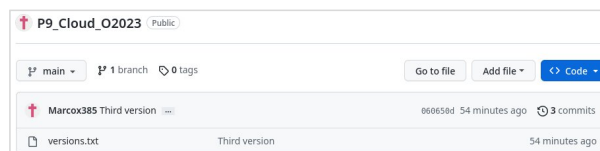


GitHub provides auxiliary commands for either creating a new local repository and pushing (uploading) to this remote repository, or, pushing existing local changes, like the ones made before. The second set of commands (to be executed inside the instance) can be modified to accommodate this demonstration:

```
git remote add Hub git@github.com:Marc Cox385/P9_Cloud_O2023.git
git push Hub main
```

```
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 699 bytes | 699.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Marc Cox385/P9_Cloud_O2023.git
* [new branch]      main -> main
```

After this, GitHub can be refreshed to verify upload:




## AWS CodeCommit

To demonstrate that GitHub isn't the only remote repository manager, AWS CodeCommit [11] will be used as source control platform. As any other service, CodeCommit has its own section inside the console. Once again, this development it's made with an institutional account, before repository creation, assume LabRole and then create it.

Name	Description	Last modified	Clone URL
P9_Repo	Practice 9 repository	Just now	<a href="#">HTTPS</a> <a href="#">SSH</a> <a href="#">HTTPS (GRC)</a>

Then the HTTPS or SSH URL can be obtained to add a new remote repository and push the same changes made before, as the previously shown process it's the same, just replace the "Hub" for "AWS" and the GitHub URL for the AWS one. The CodeCommit now can be seen with a new file inside:

P9_Repo <a href="#">Info</a>	
Name	
	<a href="#">versions.txt</a>

### Control versions of a document

One of the many advantages that a source control manager brings it's the deletion of multiple file creation in order to preserve potential versions of said files. In other words, source control managers makes it easy for users to create new files and track their changes throughout time, making possible to restore a previous version. Any kind of file can be tracked, as this exact same report. With every advancement made inside this development section or any other one, a commit was issued, resulting in the following changes (result of `git log --graph` and `git log --oneline` commands):

```
* commit 1709e5090c3a522120c5c071018192c5580 (HEAD -> main)
Author: Marco Cordero <ls727272@iteso.mx>
Date: Mon Oct 30 08:28:36 2023 +0000

    Report - Diagram done

    Diagram fully constructed and inserted inside the report

* commit 806c570b31a4f3235f3a3f69ba10e50608040
Author: Marco Cordero <ls727272@iteso.mx>
Date: Mon Oct 30 07:45:08 2023 +0000

    Report - State of the art done

    State of the part second introductory part finished

* commit ea274822946aa8a826c3275f8f8a30805b0dda
Author: Marco Cordero <ls727272@iteso.mx>
Date: Mon Oct 30 05:54:17 2023 +0000

    Report - Abstract done

    Abstract introductory part completed

* commit c839455c080b0e3015cc0e47f5a45b0b38074
Author: Marco Cordero <ls727272@iteso.mx>
Date: Mon Oct 30 05:51:09 2023 +0000

    Report - Experiments done

    Experiments section completed

* commit 1a8f8c0c096128146c39254a40d01574d3425
Author: Marco Cordero <ls727272@iteso.mx>
Date: Mon Oct 30 05:41:04 2023 +0000

    Report - Development and Problems done

    Development and Problems sections done

* commit 8ef780958c0f9080522a173089c9c9a05f6
Author: Marco Cordero <ls727272@iteso.mx>
Date: Mon Oct 30 01:03:09 2023 +0000

    Report - Development advance

    Development further advancement it's made (Control versions of a document portion of Practice Notebook)
```

```
9316732 First version
c316d98 Second version
060650d Third version
e83f09a Report - Development advance
1df8cbc Report - Development and Problems done
c639455 Report - Experiments done
ee27402 Report - Abstract done
006c57d Report - State of the art done
f19bee5 (HEAD -> main) Report - Diagram done
```

A common report creation would follow every section in order of appearance, however, in this particular case, this development section it's the very last one added to the report in order to have all the sections completed before submission. In fact, a new commit would still be needed to add these last changes.

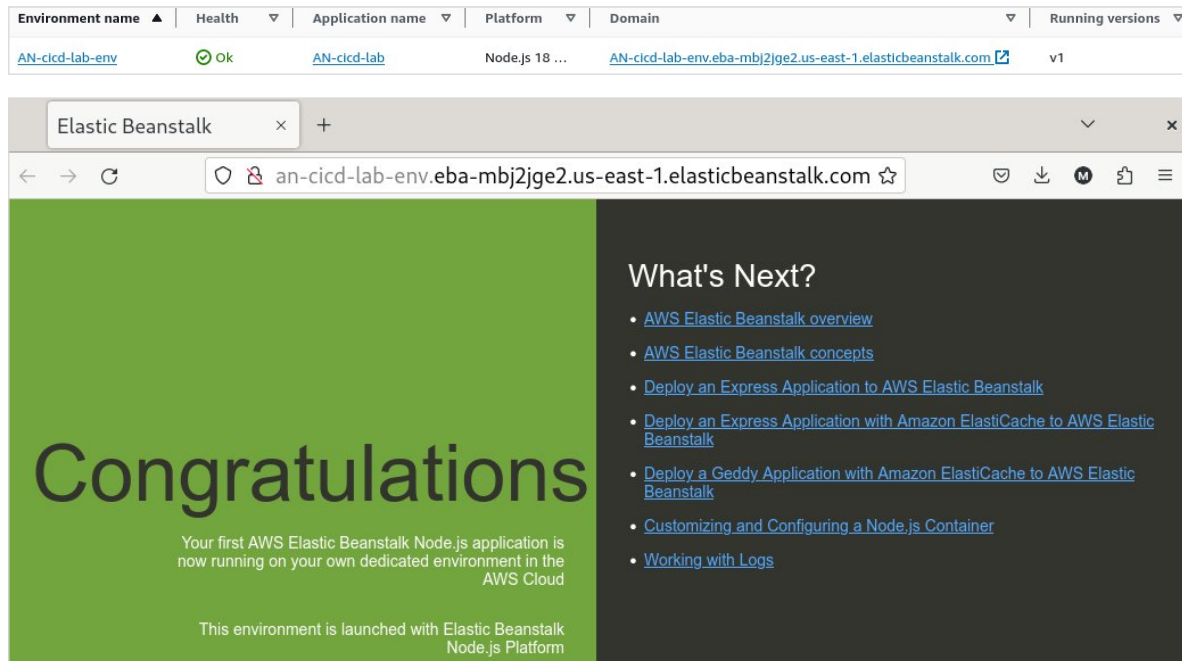
### CodePipeline and Elastic Beanstalk

As seen before, the AWS Elastic Beanstalk [12] service can be used as PaaS to deploy applications through managed environments (platforms) and applications containing the code. Previously a blue/green application environment [13] was implemented, however, new versions were uploaded manually. To eradicate this issue, AWS CodePipeline [14] service can be used. Before formally introducing this tool, an Elastic Beanstalk environment has to be created with the following characteristics:

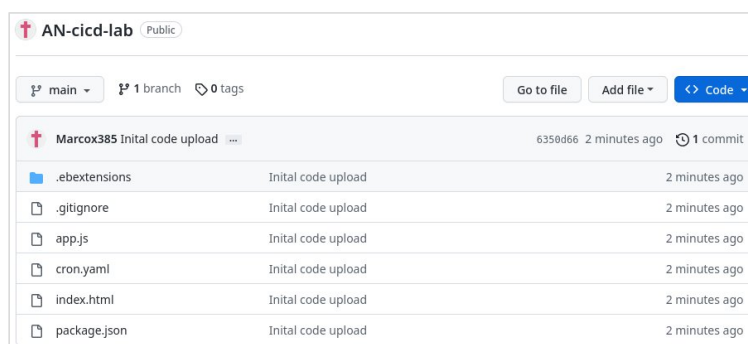
- Web server environment
- Application name: AN-cicd-lab (it will automatically create an application)
- Default environment values
- Platform
  - Managed platform



- Node.js platform
- Application code
  - Version label: v1
  - Upload your code: Node.js application (found [here](#))
- Single instance (free tier)
- Existing service role: LabRole
- EC2 key pair: vockey
- EC2 instance profile: LabInstanceProfile
- Leave default values on everything else



With this step correctly done, a new repository inside GitHub with the same name as the previous application will be created (this repository will contain future code versions). Having downloaded the same Node.js application source code, it has to be uploaded to this same repository.



Once the application has been correctly deployed and the new repository has been created, CodePipeline can be configured. This service, as every other, has its own section inside the console, in which continuous integration and delivery can be integrated with the already created Beanstalk



environment; this is specially useful for new version deployment automation. The pipeline which will be created should have these features:

- Pipeline settings
  - Name: AN-cicd-lab
  - Type: leave default
  - Service role: New service role (leave default generated name)
- Source
  - Source provider: GitHub (newest version available or version 1)
  - Connect to GitHub (an authorization windows will be prompted to allow access)
  - Repository: [Username]/AN-cicd-lab (or chosen name)
  - Branch: main
  - GitHub webhooks
- Skip build stage
- Deploy
  - Provider: AWS Elastic Beanstalk
  - Application name: AN-cicd-lab (or chosen name)
  - Environment name: AN-cicd-lab-env (or chosen name)

Similarly to Beanstalk, individual pipeline creation should take about 5 minutes.

[[ Pipeline couldn't be created ]]

Once everything has been setup, the same URL provided can be used to access the application. It should return the same view as before, but this time, the source code comes from GitHub through CodePipeline.

[[ Application backed up by CodePipeline couldn't be deployed ]]

To corroborate the last mentioned, the application can be locally modified to display a blue background (mimicking the blue/green environment) and then pushed to GitHub. This process should trigger a new deploy inside CodePipeline's pipeline which would take 5 to 10 minutes to complete. After that, the same URL from before can be used to instantly see these changes, without manual intervention.

```
[marcordero@fedora nodejs]$ git diff
diff --git a/index.html b/index.html
index f11a506..f475453 100644
--- a/index.html
+++ b/index.html
@@ -35,7 +35,7 @@
 
     text-align: right;
     padding-top: 11em;
     background-color: #73A53E;
-    background-color: #062551;
+
 }
 
 .textColumn p {
[marcordero@fedora nodejs]$ git commit -a -m "New background color" -m "Index
page background color changed from green to blue" && git push origin main
[maint cc2ac6f] New background color
1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
to github.com:Marco385/AN-cicd-lab.git
6150d66..c2ac6ff --> AN-cicd-lab:main
```

```
[[ CodePipeline new deploy demonstration couldn't be made ]]
```

[[ Beanstalk new view couldn't be demonstrated ]]

After this verification it's made, both CodePipeline and Elastic Beans created resources can be terminated to avoid unexpected charges. As its name suggests, GitHub Free, the default plan for personal accounts (like the one that should've been used for this demonstration), provides users with free service usage; because of this, both repositories created can be left as they are and no additional fees should be charged, however, they can be deleted, as they won't be used anymore (at least for the scope of this practice).

## Problems and Solutions

### Default branch name

When git is used for the first time, some global configurations will be missing such as user information and such. One of these is the default branch name, having a preset value of "master". This term was deemed derogatory back in 2020 [15], so the new widely adopted term became "main" as default branch name.

```
git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ec2-user/git/.git/
```

The previous image prompts the user with default branch name setting suggestion for further repositories. The next image shows this configuration setting and further hint suppression.

```
[ec2-user@ip-10-0-2-251 git]$ git config --global init.defaultBranch main
[ec2-user@ip-10-0-2-251 git]$ git init
Initialized empty Git repository in /home/ec2-user/git/.git/
```

### Can't upload changes to remote repository

Since August 13, 2021, GitHub doesn't accept account passwords when authenticating remote Git operations such as push or fetch [16], so, when trying to push directly with or without user credentials, the process will fail.

```
[ec2-user@ip-10-0-2-251 git]$ git push Hub main
The authenticity of host 'github.com (140.82.112.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.
```

```
Please make sure you have the correct access rights
and the repository exists.
```

To solve this, SSH key pairs need to be set. Inside the GitHub profile configurations, search for “SSH and GPG keys” section. Inside, the profile configured keys can be found, along with guides for SSH/GPG keys generation and setup. Inside the instance, this commands need to be executed:

```
ssh-keygen -t ed25519 -C "is727272@iteso.com" # Generate SSH key
```

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_ed25519
Your public key has been saved in /home/ec2-user/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:zZ2pE/N/AJwTc34w8DHZb8R45aMpKjL9YEsAx3auLU is727272@iteso.com
The key's randomart image is:
+--[ED25519 256]--+
|      .o=0|
|      o.*+=|
|    . + . . *.*o|
|    + o o .o+ =|
|    . . S =.+ = o|
|  o o + . =. .|
| . E + * .o . .|
|    = = . . .|
|    . . . . .|
+-----[SHA256]-----+
```

```
eval "$(ssh-agent -s)" # Start SSH agent (PID returned if successful)
```

```
Agent pid 4844
```

```
ssh-add ~/.ssh/id_ed25519
```

```
Enter passphrase for /home/ec2-user/.ssh/id_ed25519:
Identity added: /home/ec2-user/.ssh/id_ed25519 (is727272@iteso.com)
```

Now that the SSH key it's created and added to the local authentication agent, the generated public key will be added inside the GitHub configurations. To see this public key, run the following command:

```
cat ~/.ssh/id_ed25519.pub
```

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIG0tgPRfgfaX46L2tK8o4sMRwxw+gw1o0Kmol+8tFJ1 is727272@iteso.com
```

Copy this output and paste it to a new GitHub SSH key:

Add new SSH Key

Title

P9 Instance key

Key type

Authentication Key

Key

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIG0tgPRfgfaX46L2tK8o4sMRwxw+gw1o0Kmol+8tFJ1 is727272@iteso.com

Add SSH key

P9 Instance key

SHA256:zZ2pE/N/AJwTc34w8DHZb8R45aMpKjL9YEsAx3auLU

Added on Oct 29, 2023

Never used — Read/write

Finally, the new secure connection can be tested with the following command:

```
ssh -T git@github.com
```

ITESO

11

```
[ec2-user@ip-10-0-2-251 git]$ ssh -T git@github.com  
Hi Marcox385! You've successfully authenticated, but GitHub does not provide shell access.
```

As stated, shell access it's not provided for evident security reasons, however, now that everything it's setup, common git usage operations could now be done normally.

### Can't push to CodeCommit

As mentioned in development section, this demonstration was made with the use of an institutional/federated account, so, some actions are not available. Unfortunately, remote connections for repository change pushing are limited, thus, related actions are restricted.

⚠ You are signed in using federated access or temporary credentials. The only supported connection method for these sign-in types is to use the credential manager included with the AWS CLI, as documented below. To configure a connection using SSH or Git credentials over HTTPS, sign in as an IAM user.

Unfortunately, this can't be solved without a normal account, and console usage for file uploading and modifying was made directly through the AWS console.

### Pipeline can't be created

Similarly to the previous problem, an issue is faced at the moment of a new pipeline creation through CodePipeline.

❌ Could not create role AWSCodePipelineServiceRole-us-east-1-AN-cicd-lab

CodePipeline requires a service role in order to function, however, as it can be seen, an error raises when trying to create the configured pipeline, because the account used for this demonstration can't create new roles whatsoever. AWS provides the option for creating a new service role (root cause of this problem) or choosing an existing one, but, this second option won't solve the problem neither, as no roles can be found.

Service role

☐ New service role  
Create a service role in your account

☒ Existing service role  
Choose an existing service role from your account

Role ARN

🔍 |

Enter a service role ARN

Yet again, unfortunately, this problem can be solved, which it's a real shame because CodePipeline service seemed promising and very useful for automated deployment.

## Experiments and Results

### Would it be suitable using version control for general purpose files?

Most definitely. In every kind of development, versions are such a crucial part of them, being for recovery plans or data retrieval. This can be really useful, however, the ultimate benefit comes with

a auxiliary backup tool, so, in the case of accidental deletion, previously present files can be restored with relative ease.

### Would it be correct labeling GitHub as the same as CodeCommit?

No. Although they provide remote repository capabilities, CodeCommit is behind AWS access, whereas GitHub services are available at no cost.

### Should CI/CD be used for every kind of scenario?

It depends, but in general, no. CI/CD it's a common term used nowadays as a basic concept in software development, and having the presence of it as part of deployment for constant product upgrading it's critical for well-established processes. In a scholar context, it's implementation can be useful as part of exercising it, however, this type of scenarios involves a relatively small codebase that can be managed without automation intervention.

## Budget Justification

Having:

- Two EC2 instances
  - One for git use (optional)
  - One automatically created by Elastic Beanstalk
- One S3 bucket (automatically created by Elastic Beanstalk)
- One CodeCommit repository
- One CodePipeline pipeline

... the costs would be the following:

AWS Pricing Calculator > Version Control / Continuous Deployment		Export	Share
Version Control / Continuous Deployment		Edit	
<b>Estimate summary</b> Info		<b>Getting Started with AWS</b>	
Upfront cost	Monthly cost	Total 12 months cost	
0.00 USD	1.97 USD	23.64 USD	
		Includes upfront cost	
		Get started for free	
		Contact Sales	
<b>Version Control / Continuous Deployment</b>		Duplicate	Delete
Find resources		Move to	Create group
		Add support	Add service
		< 1 >	
<input type="checkbox"/>	Service Name	Status	Upfront cost
<input type="checkbox"/>	Amazon EC2	-	0.00 USD
<input type="checkbox"/>	AWS CodeCommit	-	0.00 USD
<input type="checkbox"/>	Amazon EC2	-	0.00 USD
<input type="checkbox"/>	Amazon Simple Storage Service (S3)	-	0.00 USD
<input type="checkbox"/>	AWS CodePipeline	-	0.00 USD
		Monthly cost	Description
		0.47 USD	Git Instance
		0.00 USD	CodeCommit
		1.50 USD	Elastic Beanstalk Instance
		0.00 USD	Elastic Beanstalk S3 Bucket
		0.00 USD	CodePipeline pipeline
			Region
			US East (N. Virginia)
			US East (N. Virginia)
			US East (N. Virginia)
			US East (N. Virginia)
			US East (N. Virginia)
			Config Summary
			Tenancy (Shared Instances), Operating system (Linux), ...
			Number of active users (1)
			Tenancy (Shared Instances), Operating system (Linux), ...
			Number of active pipelines used per account per mont...

## Conclusions

Similar to the practice in which Elastic Beanstalk was used, this demonstration leaves a great feeling about infrastructure concerns, yet again avoiding the necessity of an on-premise physical server. Both CodeCommit and CodePipeline shows opportunity for cloud implantation for multiple enterprises that are used to antiquate (but still valid) methods of serving an indistinct application, being at an enterprise level or public access.

The joint formed with traditional technologies such as git with new ones such as any AWS service allows the seamless integration with previous defined process to do a quick migration, resulting in better application with resilience and continuous operation backed up by continuous development and thus delivery/deployment.

## Bibliography

- [1] R. Peralta. 'Alan Turing's Everlasting Contributions to Computing, AI and Cryptography'. [Online]. Available: <https://www.nist.gov/blogs/taking-measure/alan-turings-everlasting-contributions-computing-ai-and-cryptography>.
- [2] Pingdom. 'The History of Computer Data Storage, in Pictures'. [Online]. Available: <https://www.pingdom.com/blog/the-history-of-computer-data-storage-in-pictures/>.
- [3] T. McMillan. 'A History of Version Control'. [Online]. Available: <https://tarynwritescode.hashnode.dev/a-history-of-version-control>.
- [4] Atlassian. 'What is version control?'. [Online]. Available: <https://www.atlassian.com/git/tutorials/what-is-version-control>.
- [5] K. Cosgrove. 'A Brief DevOps History: The Road to CI/CD'. [Online]. Available: <https://thenewstack.io/a-brief-devops-history-the-road-to-ci-cd/>.
- [6] Red Hat. 'What is a CI/CD pipeline?'. [Online]. Available: <https://www.redhat.com/en/topics/devops/what-cicd-pipeline>.
- [7] Aws.amazon.com. 'Amazon EC2'. [Online]. Available: <https://aws.amazon.com/es/ec2/>.
- [8] The Fedora Project. 'DNF'. [Online]. Available: <https://docs.fedoraproject.org/en-US/fedora/latest/system-administrators-guide/package-management/DNF/>.
- [9] Git. 'git'. [Online]. Available: <https://www.git-scm.com/>.
- [10] GitHub. 'GitHub'. [Online]. Available: <https://github.com/>.
- [11] Aws.amazon.com. 'AWS CodeCommit'. [Online]. Available: <https://aws.amazon.com/codecommit/>.
- [12] Aws.amazon.com. 'AWS Elastic Beanstalk'. [Online]. Available: <https://aws.amazon.com/elasticbeanstalk/>.

- [13] Red Hat. 'What is blue green deployment?'. [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-blue-green-deployment>.
- [14] Aws.amazon.com. 'AWS CodePipeline'. [Online]. Available: <https://aws.amazon.com/codepipeline/>.
- [15] A. Mullans. 'Renaming the default branch from master'. [Online]. Available: <https://github.com/github/renaming#renaming-the-default-branch-from-master>.
- [16] GitHub. 'Git password authentication is shutting down'. [Online]. Available: <https://github.blog/changelog/2021-08-12-git-password-authentication-is-shutting-down/>.