

Create CI/CD pipeline using AWS CodePipeline to automate source code repository, code build & code deployment

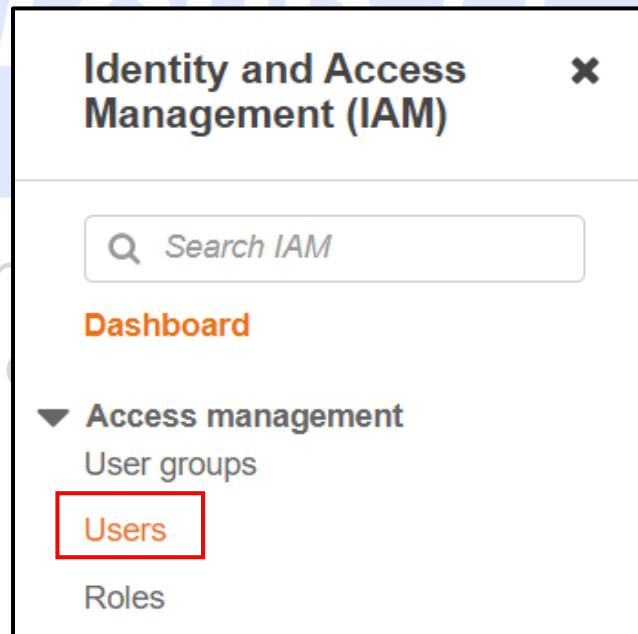
Tutorial Objectives:

1. Learn to create CodeCommit repository for the application source code.
2. Learn to configure CodeBuild to build the application source code.
3. Learn to automate software release process by continuous delivery pipeline with AWS CodePipeline.

Step 1: Log into your AWS management console and navigate to Identity and Access Management (IAM)

From the left navigation pane click on **Users**.

Click on **Add users**



Cloud Plus Plus Services



Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Find users by username or access key

| <input type="checkbox"/> | User name | Groups | Last activity | MFA | Password age | Active key age |
|--------------------------|-----------|----------------|---------------|------|--------------|----------------|
| <input type="checkbox"/> | admin | administrators | 3 minutes ago | None | 127 days ago | 113 days ago |

Name this user as **Developer1**

Tick the checkbox for both
Access Key – Programmatic access and
Password – AWS Management Console access

Give this account a secure password and click on **Next: Permissions**

User name* Developer1

+ Add another user

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Select AWS credential type* ☒ Access key - Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☒ Password - AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ☐ Autogenerated password
☒ Custom password
.....
☐ Show password


Require password reset ☐ User must create a new password at next sign-in
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.


* Required


Cancel Next: Permissions

Click on the radio button for **Attach existing policies directly**.

▼ Set permissions

 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

Create group Refresh

Attach the following policies to your user

- **IAMFullAccess**
- **AWSCodeCommitFullAccess**
- **AWSCodeBuildAdminAccess**
- **AWSCodePipeline_FullAccess**
- **AdministratorAccess-AWSElasticBeanstalk**
- **AWSCloud9Administrator**

Click on **Next: Tags**

There's no need for any tags for now, Click on **Next: Review**

Verify User Details and click on **Create User**

User details

| | |
|------------------------|---|
| User name | Developer1 |
| AWS access type | Programmatic access and AWS Management Console access |
| Console password type | Custom |
| Require password reset | No |
| Permissions boundary | Permissions boundary is not set |

Permissions summary

The following policies will be attached to the user shown above.

| Type | Name |
|----------------|---|
| Managed policy | IAMFullAccess |
| Managed policy | AWSCodeCommitFullAccess |
| Managed policy | AWSCodeBuildAdminAccess |
| Managed policy | AWSCodePipeline_FullAccess |
| Managed policy | AdministratorAccess-AWSElasticBeanstalk |

Tags

No tags were added.

Cancel Previous **Create user**

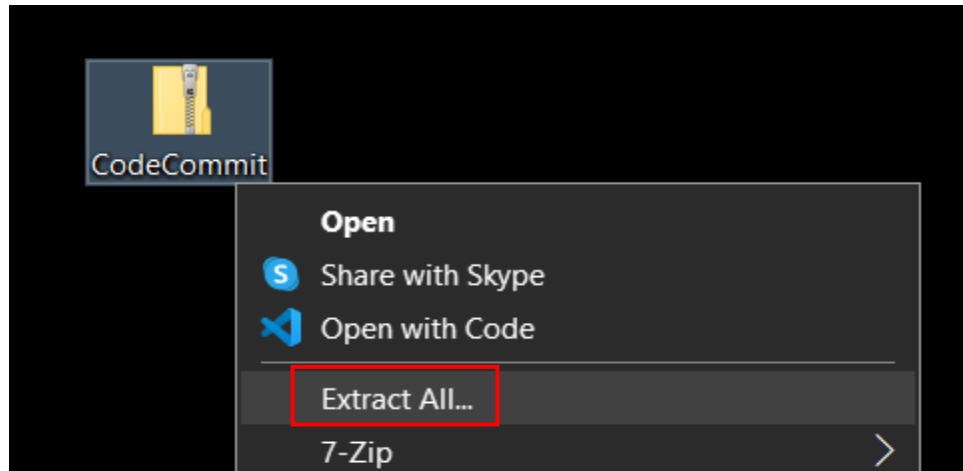
Cloud Plus Plus Services



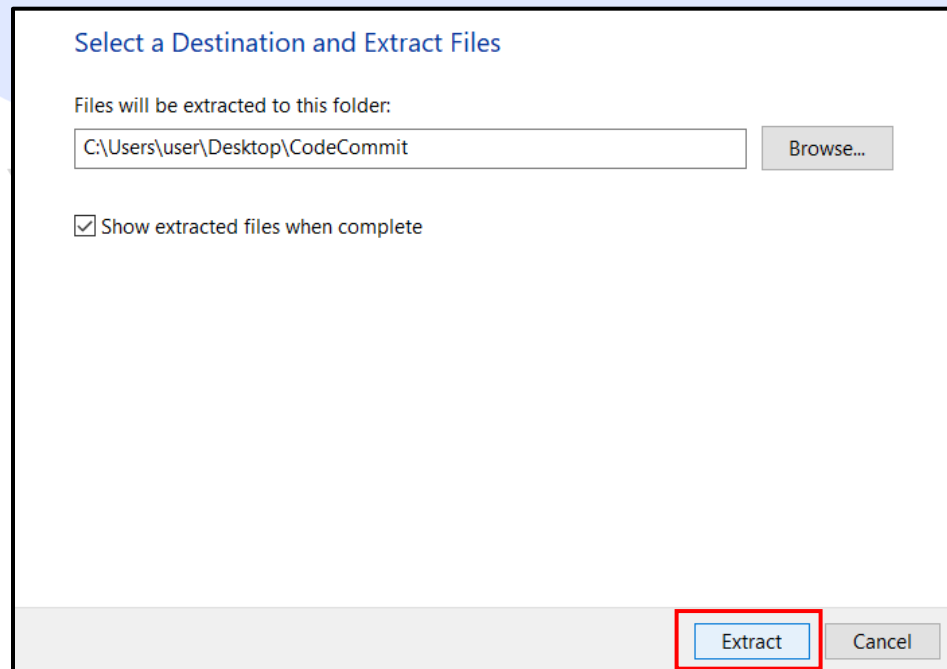
Log out of our AWS Management console from your current user and log back in as Developer1.

Step 2: Download the CodeCommit.zip file from [here](#)

Right click on this zip file and click on **Extract All...**



Click on **Extract**



Cloud Plus Plus Services



Step 3: Open AWS Elastic Beanstalk

Click on **Create Application**

The image shows the Amazon Elastic Beanstalk landing page. It features a dark blue header with the text 'Compute' and 'Amazon Elastic Beanstalk End-to-end web application management.' Below this, a description states: 'Amazon Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.' On the right side, there is a white box titled 'Get started' with the text 'Easily deploy your web application in minutes.' and a red-bordered button labeled 'Create Application'.

Name your application as **DevOpsGettingStarted**

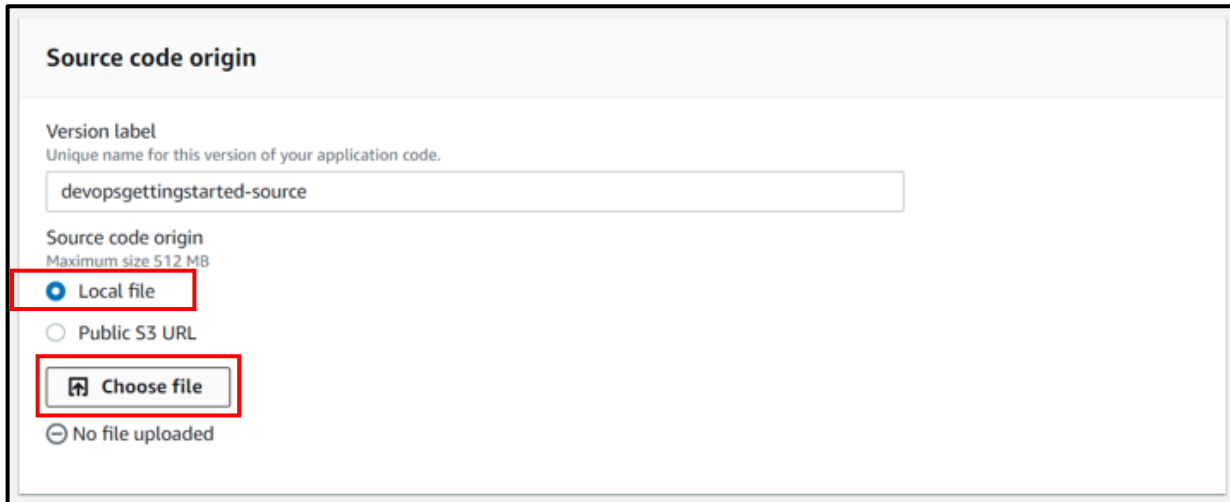
In Platform open the dropdown and select **NodeJs** as your platform

The image shows a 'Platform' configuration form. It has three dropdown menus. The first dropdown, labeled 'Platform', has 'Node.js' selected and is highlighted with a red box. The second dropdown, labeled 'Platform branch', has 'Node.js 14 running on 64bit Amazon Linux 2' selected. The third dropdown, labeled 'Platform version', has '5.4.7 (Recommended)' selected.

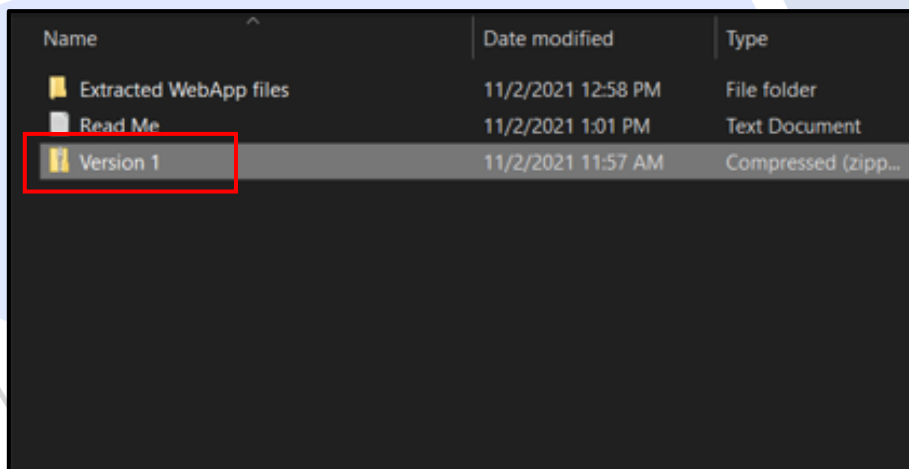
Inside the **Application code** click on **Upload your code**

The image shows an 'Application code' selection form. It has two radio buttons. The first radio button is labeled 'Sample application' with the subtext 'Get started right away with sample code'. The second radio button is labeled 'Upload your code' with the subtext 'Upload a source bundle from your computer or copy one from Amazon S3.' and is highlighted with a red box.

Under **Source Code Origin** make sure **local file** is selected and then click on **Choose File**

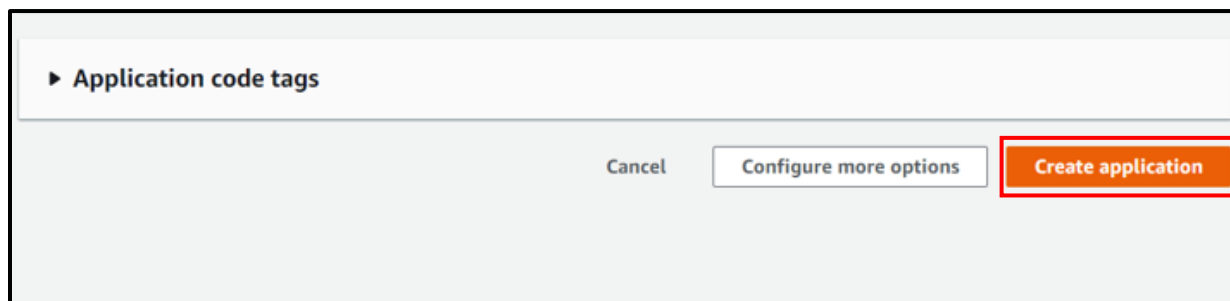


Select the **Version 1.zip** file from your previously extracted CodeCommit folder



| Name | Date modified | Type |
|------------------------|--------------------|---------------------|
| Extracted WebApp files | 11/2/2021 12:58 PM | File folder |
| Read Me | 11/2/2021 1:01 PM | Text Document |
| Version 1 | 11/2/2021 11:57 AM | Compressed (zipp... |

Finally click on **Create Application** and Version 1 of your NodeJs web application will be hosted

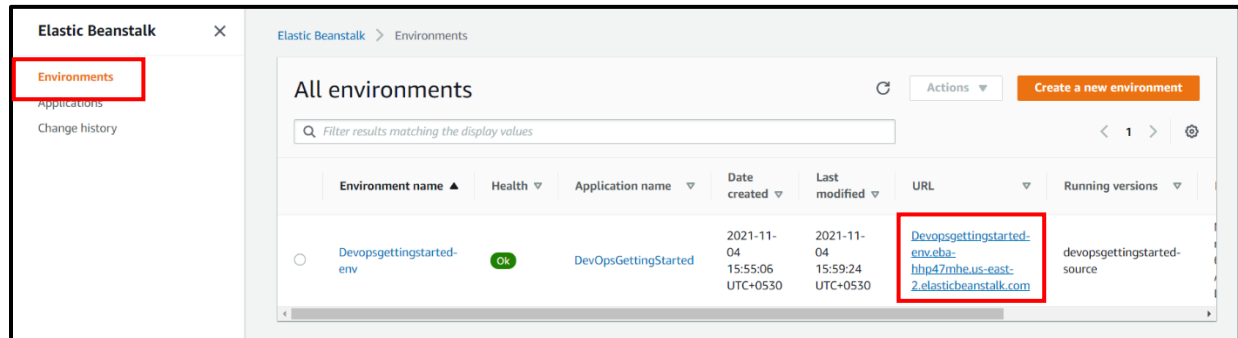


Cloud Plus Plus Services

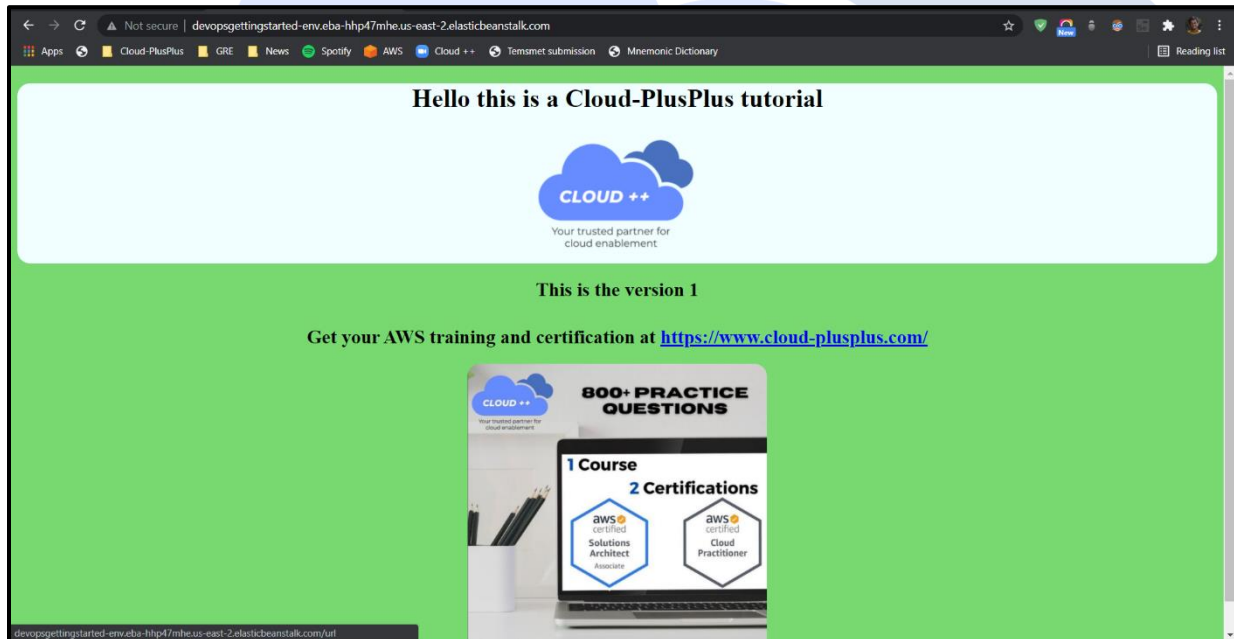


Wait for a few minutes while Elastic Beanstalk is hosting your web application.

Once the application is up and running from the left navigation pane click on **Environments** and click on the URL given for your Web application environment.



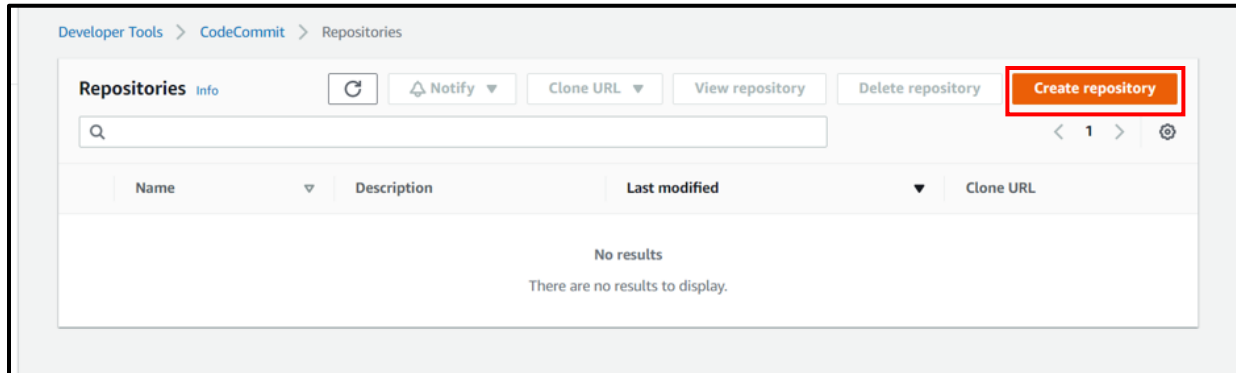
And as you can see our Web Application Version 1 is now hosted on the internet.



Cloud Plus Plus Services

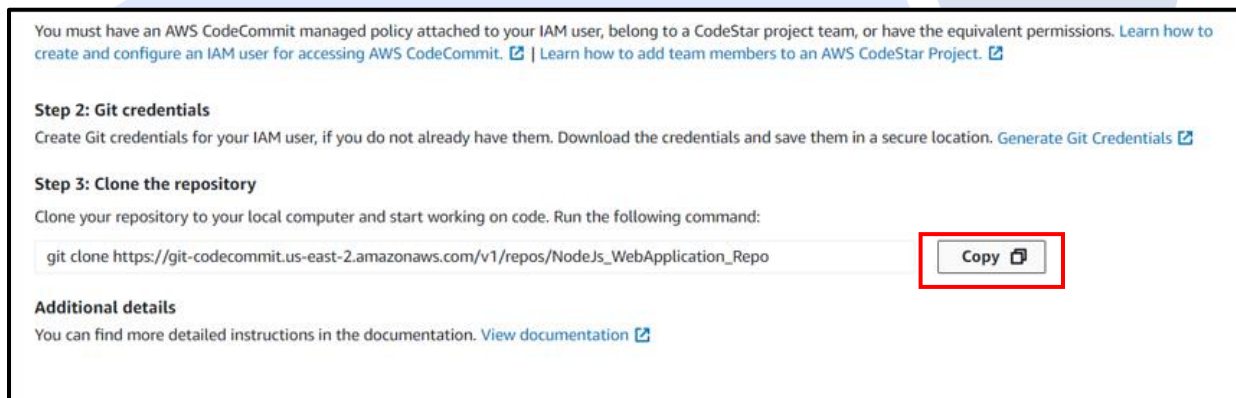


Step 4: Open your AWS management console in another tab and navigate to AWS CodeCommit. Click on **Create repository**

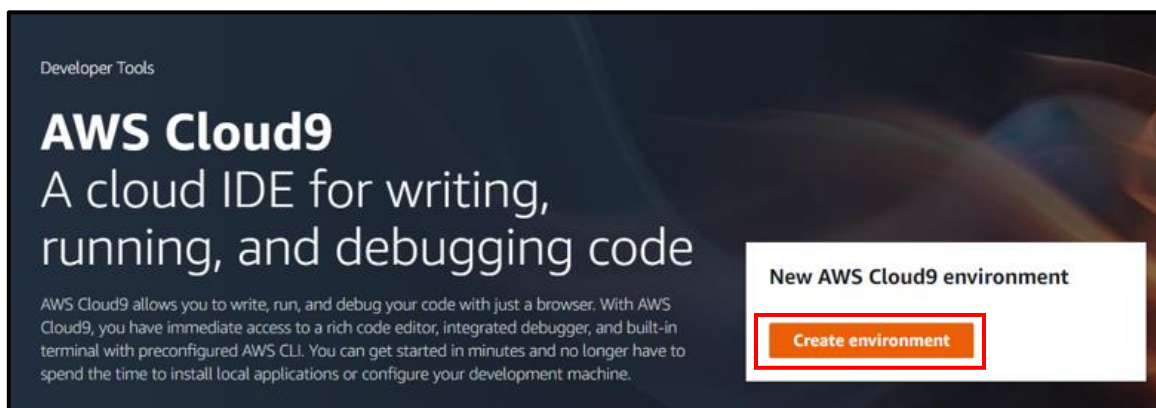


Give an appropriate name to your repository. For now, we'll call it **NodeJs_WebApplication_Repo**

Once our repository is created scroll down and click on the **Copy** button to copy the git command to clone our repository.



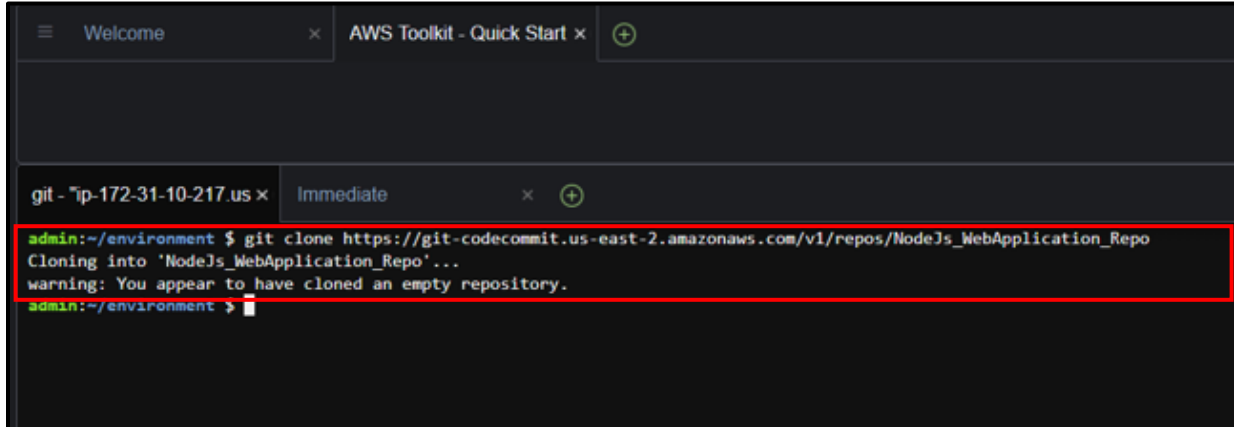
Step 5: Now, open your AWS management console in another new tab and navigate to Cloud9.



Click on **Create environment**

Give a name to your Environment, We'll call it **CodeCommit_Playground**

Once the environment is up and running paste the previously copied git clone command in the terminal.



```
admin:~/environment $ git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/NodeJs_WebApplication_Repo
Cloning into 'NodeJs_WebApplication_Repo'...
warning: You appear to have cloned an empty repository.
admin:~/environment $
```

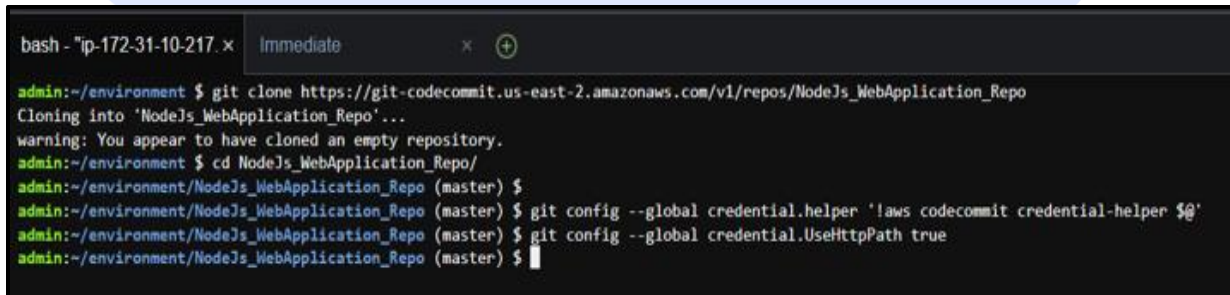
For remote accessing the CodeCommit bucket we need to provide our user's http credentials to the AWS via the terminal. For that, run the following commands.

```
cd NodeJs_WebApplication_Repo/
```

```
git config --global credential.helper '!aws codecommit credential-helper $@'
```

```
git config --global credential.UseHttpPath true
```

These commands will provide your http credentials to access the bucket.



```
admin:~/environment $ git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/NodeJs_WebApplication_Repo
Cloning into 'NodeJs_WebApplication_Repo'...
warning: You appear to have cloned an empty repository.
admin:~/environment $ cd NodeJs_WebApplication_Repo/
admin:~/environment/NodeJs_WebApplication_Repo (master) $
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git config --global credential.helper '!aws codecommit credential-helper $@'
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git config --global credential.UseHttpPath true
admin:~/environment/NodeJs_WebApplication_Repo (master) $
```

Now, we need to specify the user who is going to commit the changes in our CodeCommit repository.

For that, we need to execute the following commands to specify the user.

```
git config --global user.name "<Your Name>"
```

```
git config --global user.email <your.email@email.com>
```

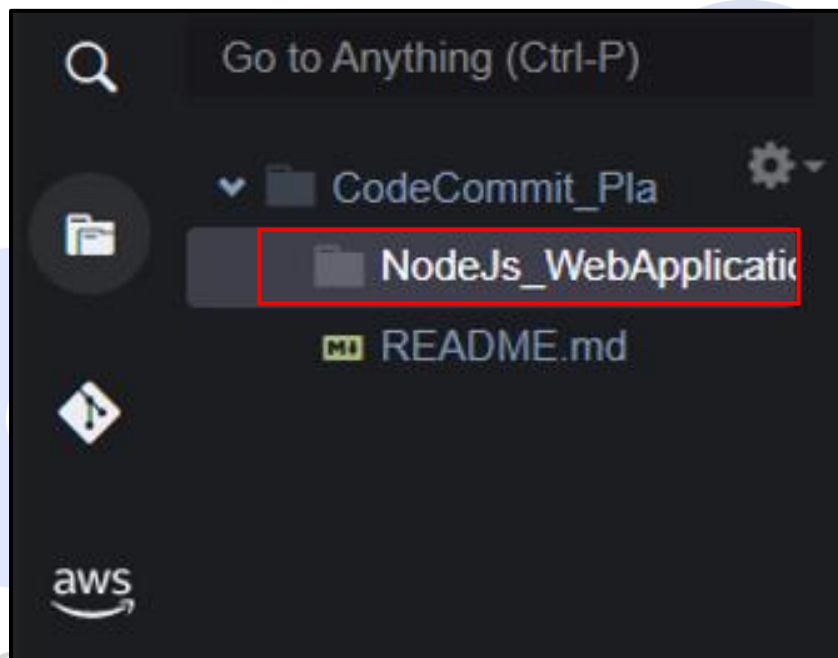
Cloud Plus Plus Services



```
bash - "ip-172-31-10-217. x Immediate x +
admin:~/environment/NodeJs_WebApplication_Repo (master) > git config --global credential.helper :aws codecommit credential-helper >
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git config --global credential.UseHttpPath true
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git config --global user.name "Developer1"
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git config --global user.email soham.pingat@gmail.com
admin:~/environment/NodeJs_WebApplication_Repo (master) $ |
```

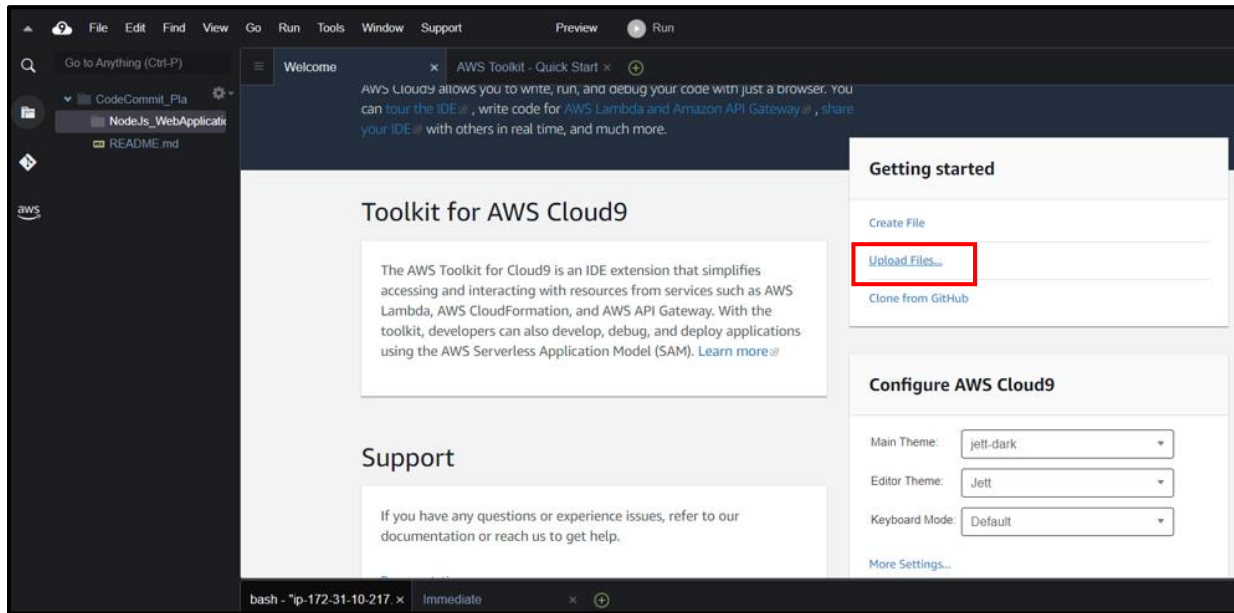
Now, In the left navigation pane, where you can see the folder structure of your environment click on your repository name. For now, it is **NodeJs_WebApplication_Repo**.

This will ensure that when we upload our Version 2 files, they will be uploaded in the folder of the repository.



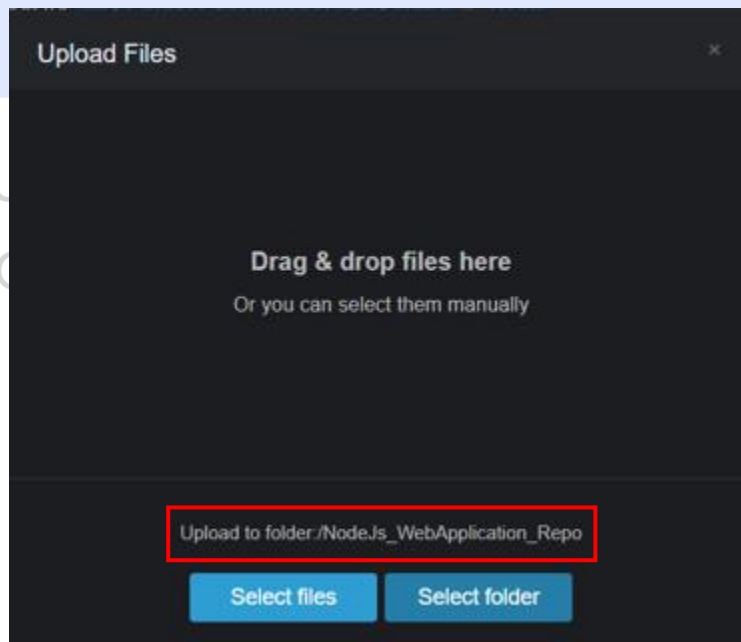
From the **Welcome** tab click on **Upload Files...**

Cloud Plus Plus Services



When the **Upload Files** pop-up appears make sure the path specifies Upload to folder: `/NodeJs_WebApplication_Repo` and click on **Select Files** to upload the version 2 files of our Web Application

CLOUD ++



Cloud Plus Plus Services

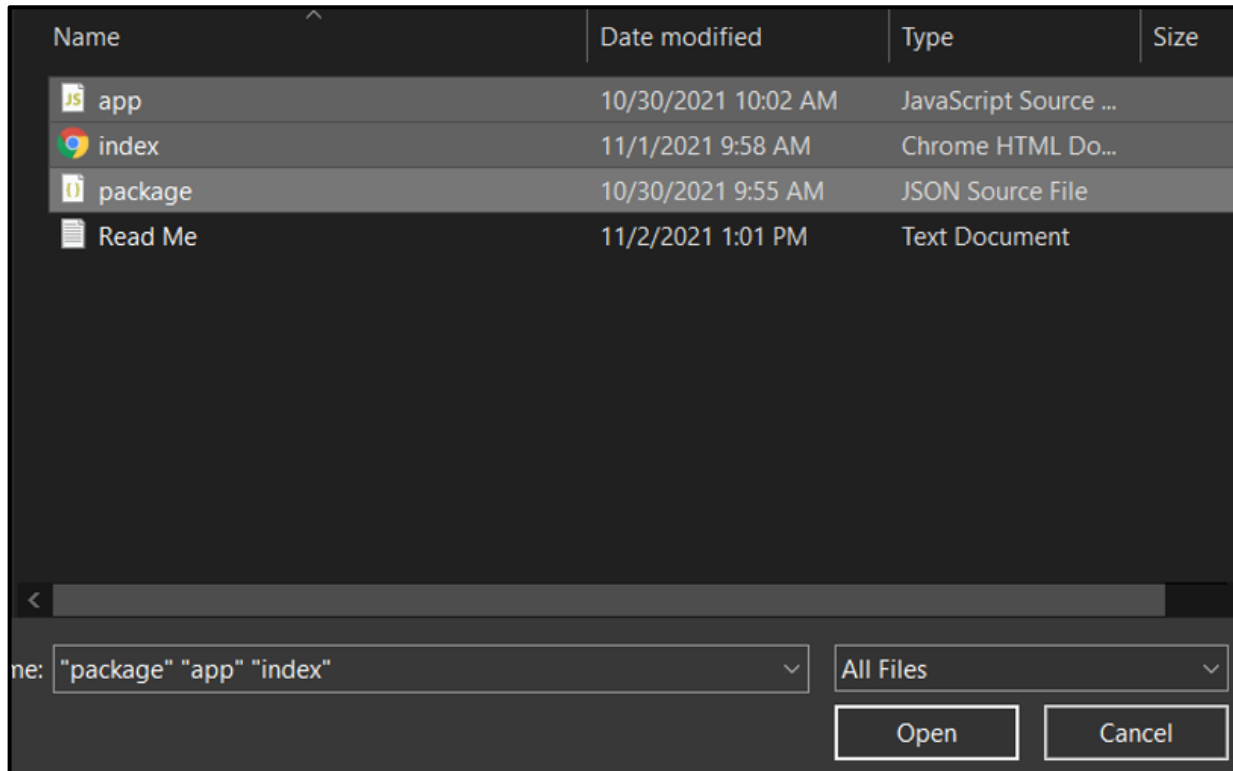


From the extracted CodeCommit folder, Go into the **Version 2** folder and select the following files to upload:

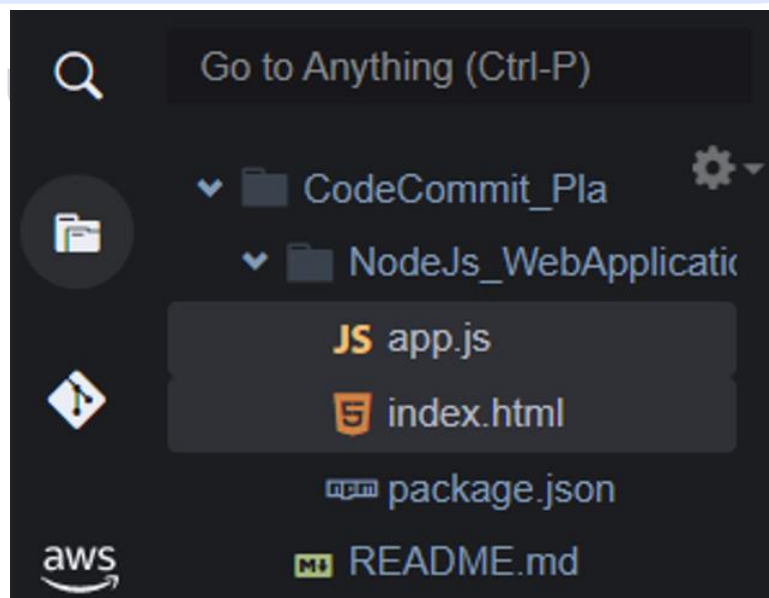
app.js

index.html

package.json



Click on Open and these files will be uploaded in our repository folder.



As you can see in the left folder structure pane, all our files are uploaded inside our NodeJs_WebApplication_Repo folder.

Run the following commands to push these files to our CodeCommit repository.

git add .

git commit -m "Version 2 of the Web Application"

git push

git add . : This command will add all the uploaded files to the staging area to commit the change.

git commit -m "Version 2 of the Web Application" : This command will commit our new changes with the message "Version 2 of the Web Application".

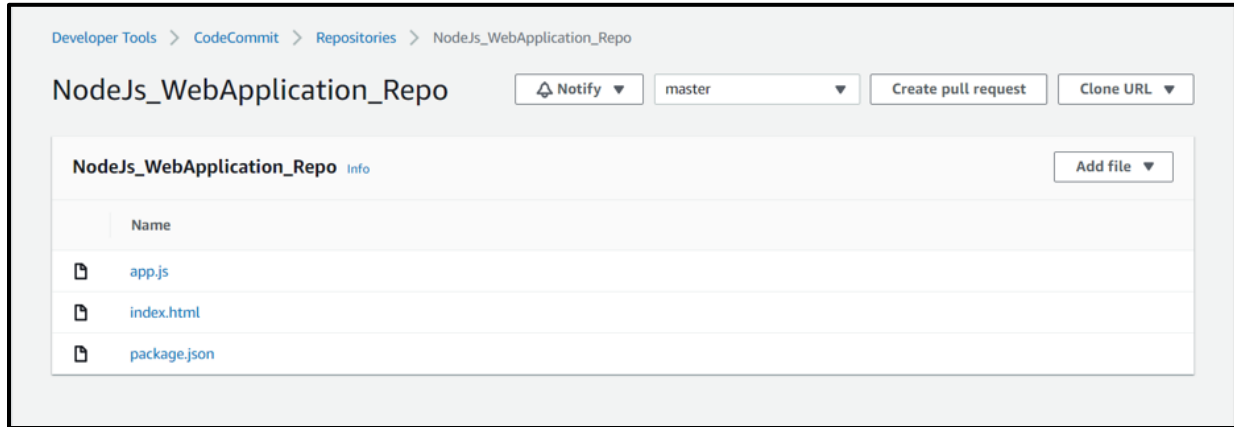
git push : This command will push all these changes to the master branch of our CodeCommit repository.

```
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git add .
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git commit -m "Version 2 of the Web Application"
[master (root-commit) fd27a63] Version 2 of the Web Application
3 files changed, 75 insertions(+)
create mode 100644 app.js
create mode 100644 index.html
create mode 100644 package.json
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.29 KiB | 661.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/NodeJs_WebApplication_Repo
* [new branch]      master -> master
admin:~/environment/NodeJs_WebApplication_Repo (master) $
```

Cloud Plus Plus Services

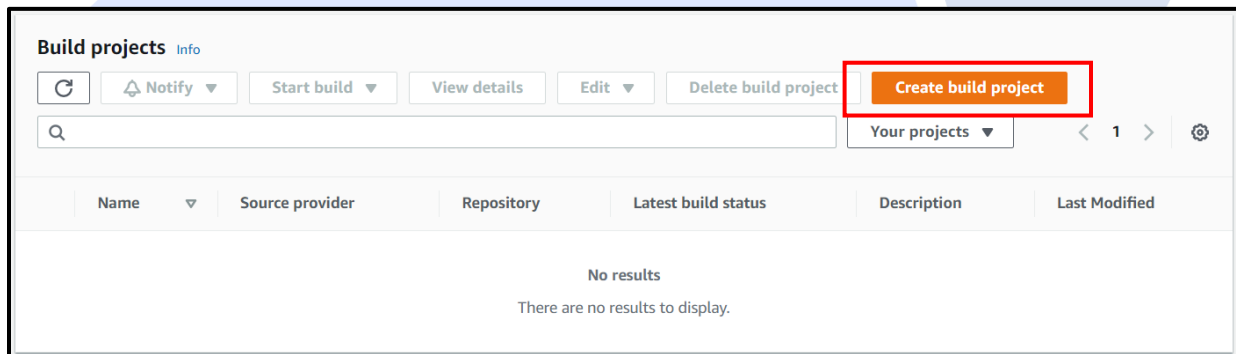


Now, go back to your CodeCommit tab and refresh the page. And you'll be able to see the new Version 2 files are uploaded inside the CodeCommit repository



Step 6: Open AWS management console in a new tab and navigate to AWS CodeBuild.

Click on **Create Build Project**



Project Name: **Build-DevOpsGettingStarted**

Under **Source**:

- Open the Dropdown and select **AWS CodeCommit** as a source.
- For Repository open the dropdown and select NodeJs_WebApplication_Repo.
- Let the reference tag be Branch

Cloud Plus Plus Services



- Select **Master** Branch as your branch.

Source

Add source

Source 1 - Primary

Source provider

AWS CodeCommit

Repository

NodeJs_WebApplication_Repo

Reference type

Choose the source version reference type that contains your source code.

☒ Branch

☐ Git tag

☐ Commit ID

Branch

Choose a branch that contains the code to build.

master

Commit ID - optional

Choose a commit ID. This can shorten the duration of your build.

Source version Info

refs/heads/master

fd27a637 Version 2 of the Web Application

Under **Environment**:

- Let it be **Managed Image**
- For operating system select **Amazon Linux 2**
- Runtimes : **Standard**
- Image : **aws/codebuild/amazonlinux2-x86_64-standard:3.0**
- Visually confirm that **Always use the latest image for this runtime version" is selected for "Image version.**
- Visually confirm that **Linux** is selected for **Environment type**
- Visually confirm that **New service role** is selected.

Cloud Plus Plus Services



Environment image

☒ **Managed image**
Use an image managed by AWS CodeBuild

☐ **Custom image**
Specify a Docker image

Operating system

Amazon Linux 2

The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See [Docker Images Provided by CodeBuild](#) for details.

Runtime(s)

Standard

Image

aws/codebuild/amazonlinux2-x86_64-standard:3.0

Image version

Always use the latest image for this runtime version

Environment type

Linux

For Buildspec:

Click on the Radio button for **Insert build Commands** and click on **Switch to editor**

Buildspec

Build specifications

☐ **Use a buildspec file**
Store build commands in a YAML-formatted buildspec file

☒ **Insert build commands**
Store build commands as build project configuration

Build commands

Enter commands you want to run during the build phase. Separate each build command with "&&." For example, "mvn test && mvn package." Use a buildspec file to run commands in other phases or if you have a long list of commands.

Switch to editor

Cloud Plus Plus Services



Replace the Buildspec in the editor with the code below

```
version: 0.2
phases:
  build:
    commands:
      - npm i --save
artifacts:
  files:
    - '**/*'
```

This code is to install the required runtime libraries from Package.json which we uploaded previously to host and run our web application.

Buildspec

Build specifications

☐ Use a buildspec file
Store build commands in a YAML-formatted buildspec file

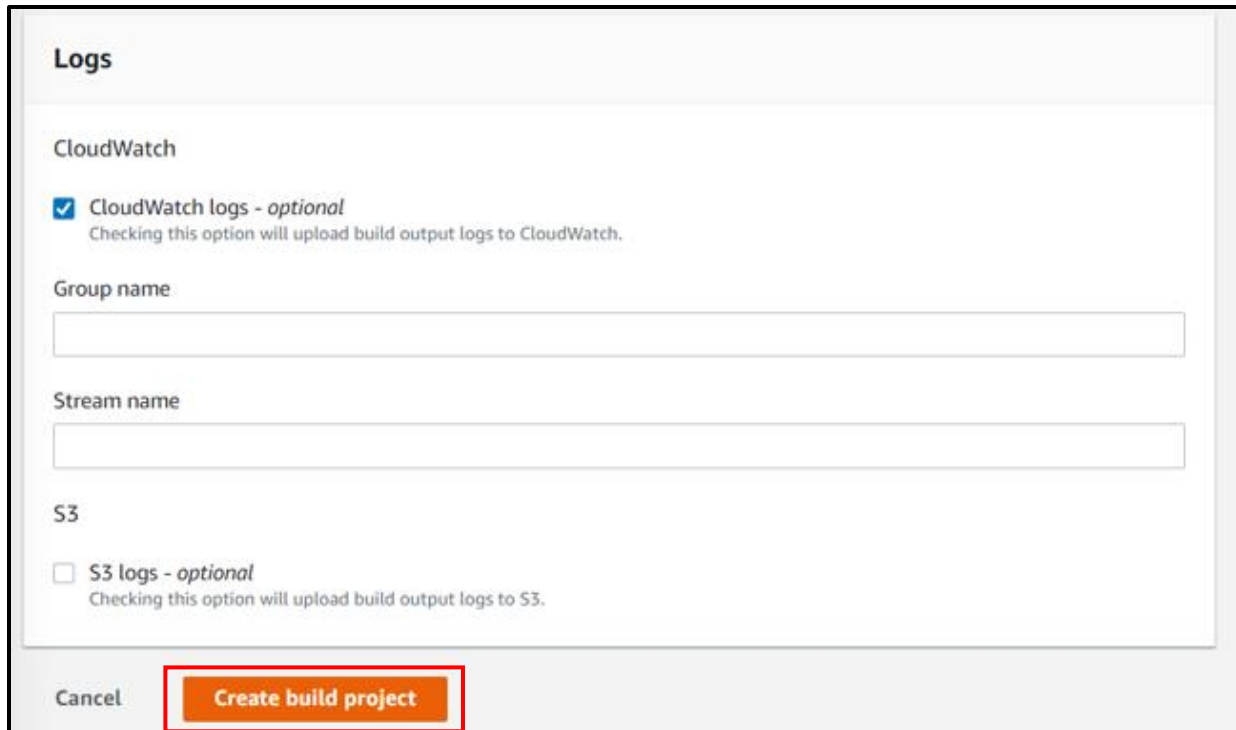
☒ Insert build commands
Store build commands as build project configuration

Build commands

```
1 version: 0.2
2 phases:
3   build:
4     commands:
5       - npm i --save
6 artifacts:
7   files:
8     - '**/*'
```

Switch to single line

Leave the rest of the settings as it is and scroll down to click on **Create Build Project**.



Step 7: Open AWS Management console in a new tab and open AWS CodePipeline.

Click on create **Create Pipeline**

- Give an appropriate name to your pipeline. For now, We'll call it **Pipeline-DevOpsGettingStarted**
- Visually confirm that "New service role" is selected

Under **Source**

- Source Provider : **AWS CodeCommit**
- Repository Name : **NodeJs_WebApplication_Repo**
- Branch Name : **Master**
- Visually Confirm **Amazon CloudWatch Events** is selected for Change Detection option.
- For Output artifact format, select **Full Clone**

Click on **Next**

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

NodeJs_WebApplication_Repo

Branch name
Choose a branch of the repository

master

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

☒ **Amazon CloudWatch Events (recommended)**
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

☐ **AWS CodePipeline**
Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.

☐ **CodePipeline default**
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.

☒ **Full clone**
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

Under **Build:**

- Build Provider: **AWS CodeBuild**
- Region: Your default region. For now, it is **Ohio**
- Project Name: **Build-DevOpsGettingStarted**
- Visually confirm **Single Build** is selected for Build Type

Click on **Next** to continue to Deployment stage.

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild ▼

Region

US East (Ohio) ▼

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

X or [Create project](#)

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Add environment variable

Build type

☒ **Single build**
Triggers a single build.

☐ **Batch build**
Triggers multiple builds as a single execution.

Cancel

Previous

Skip build stage

Next

Under Deployment:

- Deploy Provider: **AWS Elastic Beanstalk**
- Region: Your default region. For now, it is **Ohio**
- Application Name: **DevOpsGettingStarted**
- Environment Name: **DevOpsGettingStarted-env**

Click on **Next** and Create the Pipeline.

Once the Pipeline is created you can see 3 Modules

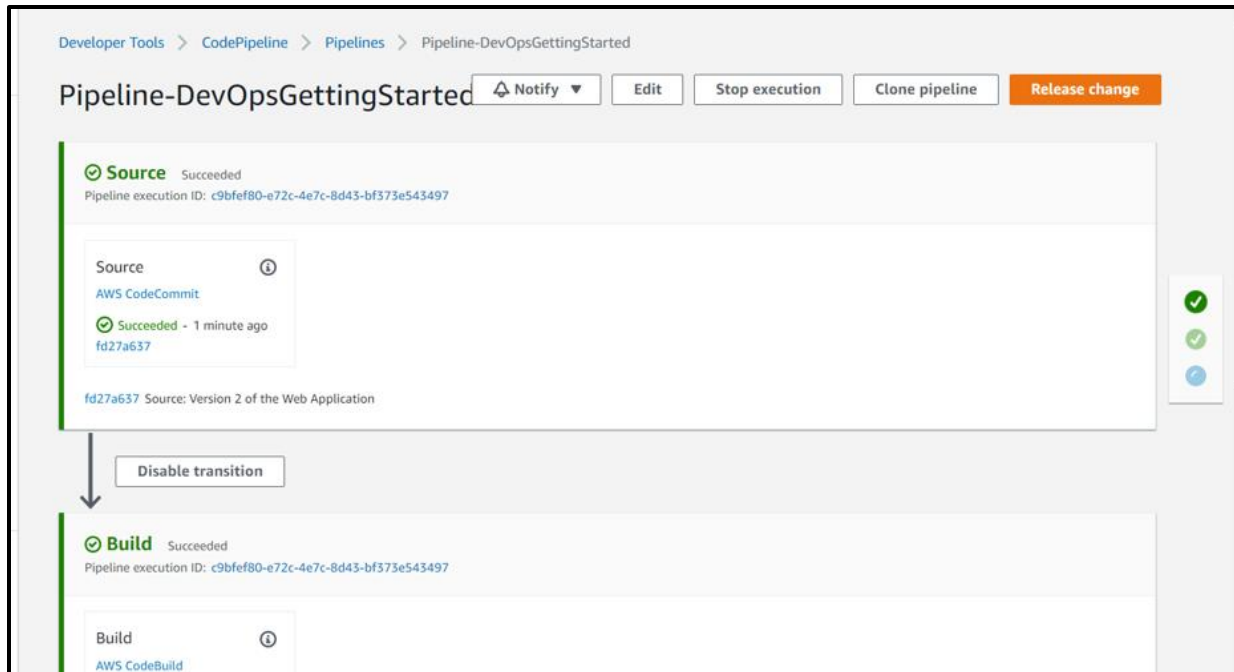
Source

Build

Deploy

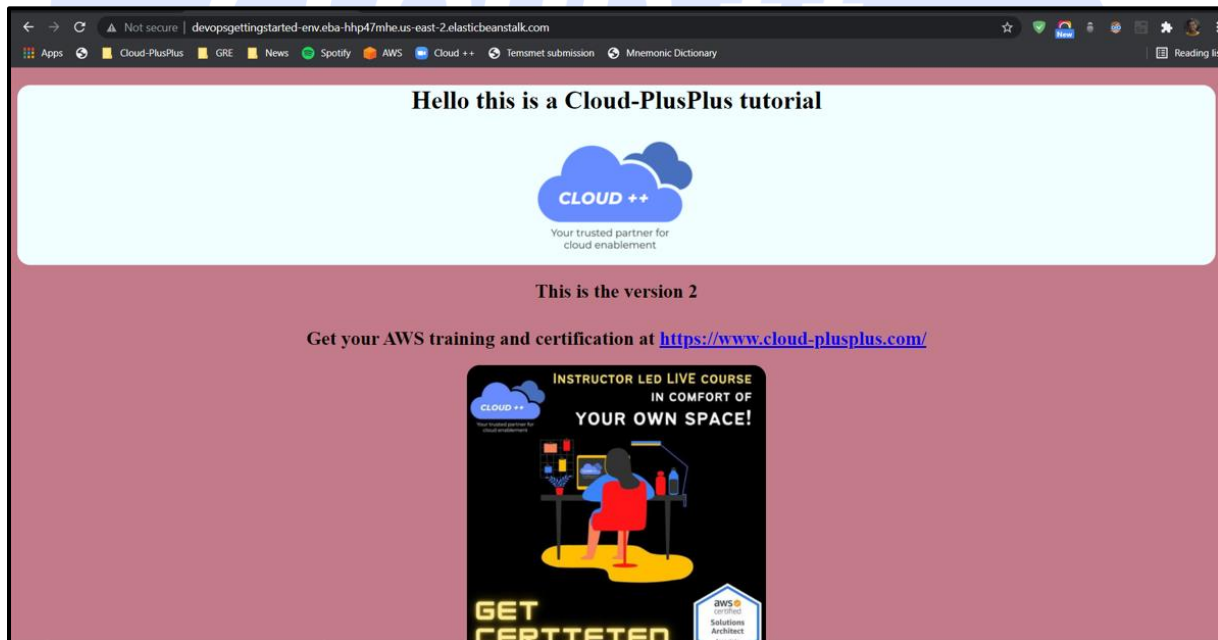
Are under execution.

Cloud Plus Plus Services



Once the **Deploy** phase displays “**Succeeded**”, Go to Elastic Beanstalk and again click on the hosting URL.

And you can see the Version 2 of our Web Application is now hosted.

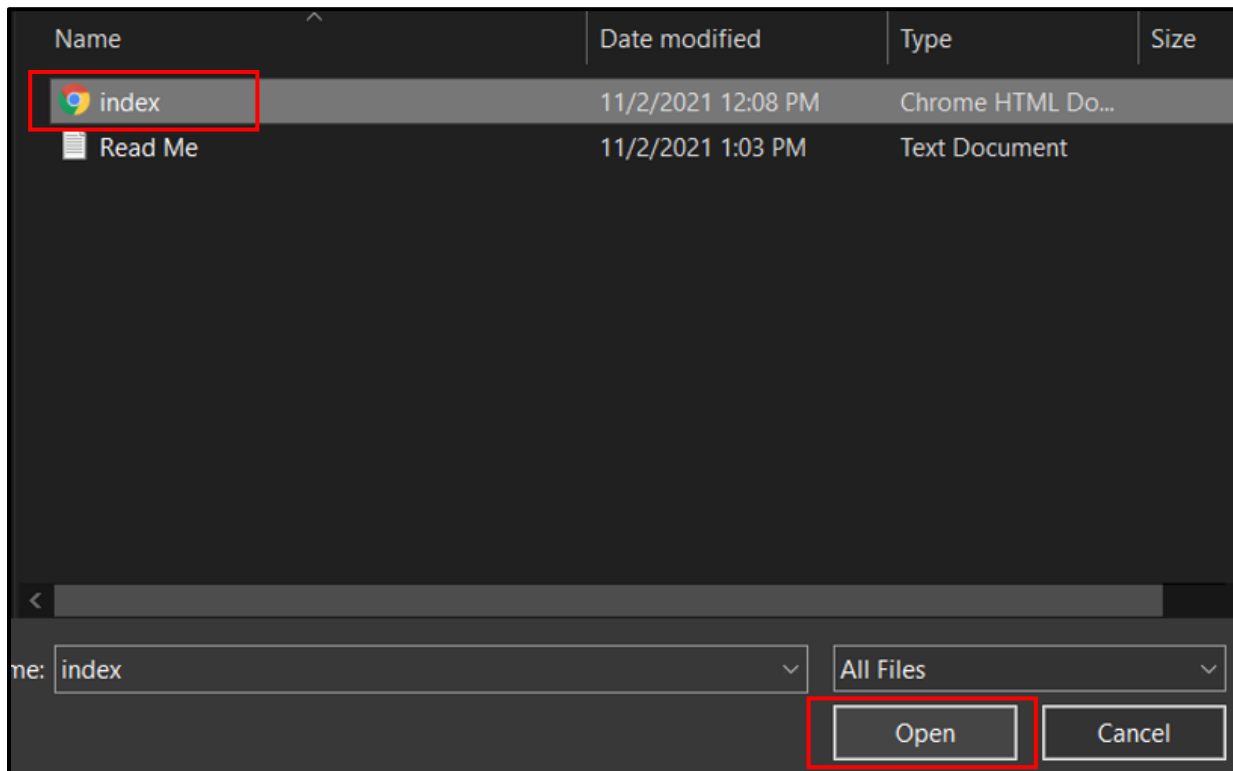


Step 8: Now to test the continuous delivery of our tutorial we're going to commit a Version 3 of our Web Application to the CodeCommit repository.

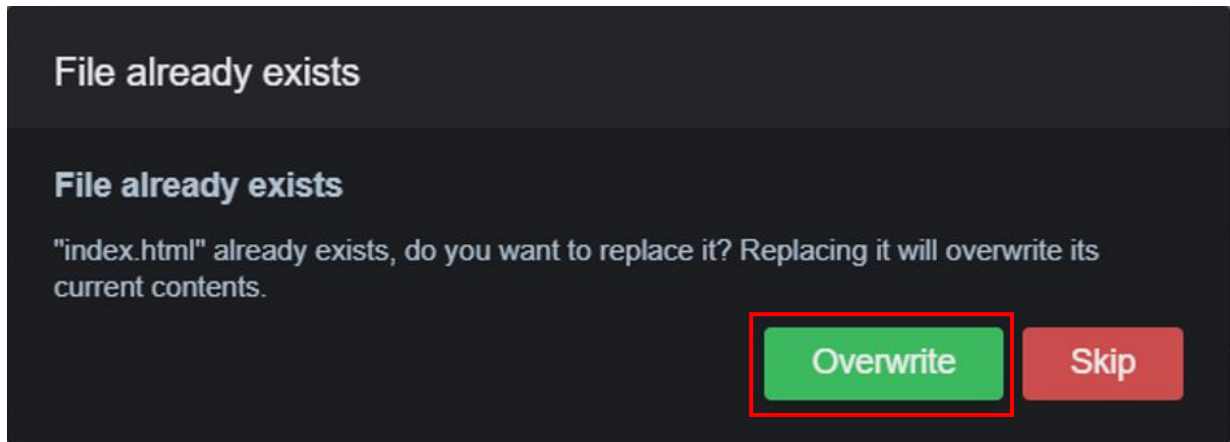
Go back to Cloud9 IDE tab and click on your Repository Folder from the Left Folder Structure pane (as instructed in Step 5)

Click on Upload Files. Make sure the Upload to folder path displays /NodeJs_WebApplication_Repo and click on Select files.

Open the **Version 3** folder from the extracted CodeCommit folder and select the **index.html** file. Click on Open



The Cloud9 will give a warning about a file with similar name is already existing and will confirm if you want to overwrite the existing index.html file. Click on **Overwrite**



Again run the git commands to push this change to the CodeCommit repository

`git add .`

`git commit -m "Version 3 of the Web Application"`

`git push`

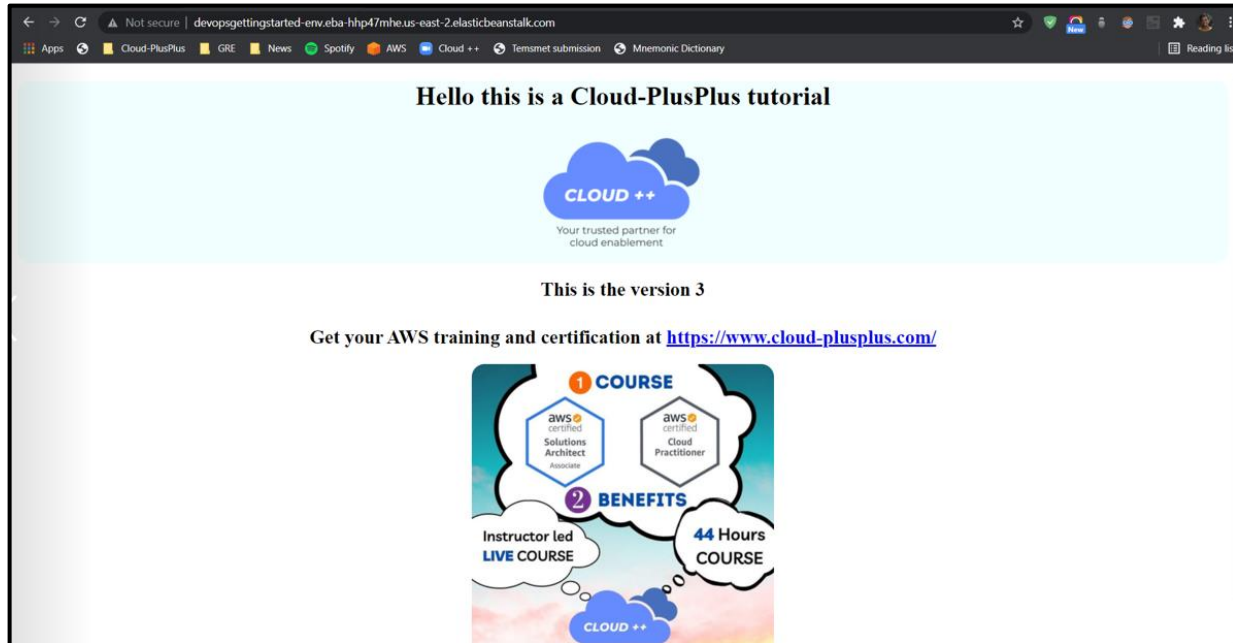
```
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git add .
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git commit -m "Version 3 of the Web Application"
[master c3bf1d3] Version 3 of the Web Application
1 file changed, 3 insertions(+), 3 deletions(-)
admin:~/environment/NodeJs_WebApplication_Repo (master) $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 411 bytes | 411.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/NodeJs_WebApplication_Repo
fd27a63..c3bf1d3 master -> master
admin:~/environment/NodeJs_WebApplication_Repo (master) $
```

Now, Navigate to CodePipeline and observe all 3 stages Source, Build, and Deploy in progress and succeed for the new change we made in our repository.

Once the **Deploy** stage displays **Succeeded** go to Elastic Beanstalk and click on the hosting URL.



Cloud Plus Plus Services



This is how, we have successfully created a continuous delivery pipeline using AWS CodeCommit.

Note: If you no longer need the resources, Delete the following resources:

- CodeCommit Bucket
- Cloud9 Environment
- CodeBuild Build
- CodePipeline Pipeline
- Developer1 IAM user (you'll need to login back again as admin to delete this user)

Was this document helpful? YES / NO

| Document Created by | Version |
|---------------------|------------------|
| Soham Pingat | 4-November-2021 |
| Bavyaa R | 24-November-2021 |

Your trusted partner for
cloud enablement