A
Project Report
On

CI / CD Pipeline using AWS

BTech-IT, Sem VIII

Prepared By:
Akshar Lathiya (IT067)

Internal Guided By:
Prof. (Dr.) Harshad Prajapati
Dept. of Information Technology

External Guided By:
Suraj Kalgude
Toshal Infotech.

Department of Information Technology
Faculty of technology,
Dharmsinh Desai University
College road, Nadiad- 387001
April 2024

CANDIDATE'S DECLARATION

I declare that 8th semester report entitled "**CI / CD Pipeline Using AWS**" is my own work conducted under the supervision of the internal guide **Prof. (Dr.) Harshad Prajapati** and External guide **Suraj Kalgude**.

I/We further declare that to the best of my/our knowledge the report for B.Tech. VIII semester does not contain part of the work which has been submitted either in this or any other university without proper citation.

Candidate's Signature
Akshar Lathiya
Student ID: 20ITUOS102

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project carried out in the subject of System Design Practice, entitled " **CI / CD Pipeline Using AWS** " and recorded in this report is a bonafide report of work of

AKSHAR LATHIYA ALPESHKUMAR IT067 20ITUOS102

of Department of Information Technology, semester VIII. He/She/They was/were involved in Project work during academic year 2023 - 2024.

Prof. (Dr.) Harshad Prajapati
(Project Guide),
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. (Dr.) V. K. Dabhi
Head, Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

ACKNOWLEDGEMENT

I would like to extend my heartfelt gratitude to all those who contributed to the successful completion of project on the CI/CD pipeline.

Firstly, I express my deepest appreciation to our internal guide, **Prof. (Dr.) Harshad Prajapati**, for their invaluable support, guidance, and mentorship throughout this project. Their expertise and encouragement have been instrumental in shaping the direction of our work and ensuring its quality.

I would also like to extend my sincere thanks to our external guide, **Suraj Kalgude**, for their insightful feedback, constructive criticism, and expert advice. Their perspective from outside our organization has provided us with fresh insights and helped us refine our approach.

Additionally, I am grateful to the entire team for their dedication, collaboration, and hard work, without which this project would not have been possible.

Finally, I acknowledge the support of all individuals who directly or indirectly contributed to this endeavour. Your assistance and encouragement have been invaluable.

Thank you all for your contributions and commitment.

Sincerely,

Lathiya Akshar A.

Contents

ABSTRACT

The Continuous Integration/Continuous Deployment (CI/CD) pipeline has emerged as a pivotal component in modern software development, facilitating the seamless integration, testing, and deployment of code changes. This abstract explores the fundamental concepts, principles, and benefits of CI/CD pipelines in software engineering. It delves into the various stages of the pipeline, including code integration, automated testing, and continuous deployment, highlighting their significance in ensuring software quality, reducing time-to-market, and enhancing collaboration among development teams. Moreover, the abstract discusses the adoption challenges, best practices, and emerging trends in CI/CD pipeline implementation, providing insights into how organizations can leverage this methodology to streamline their software delivery processes and achieve greater agility and efficiency in today's dynamic digital landscape.

LIST OF FIGURES

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Traditional software development and deployment processes often suffer from inefficiencies, delays, and errors due to manual intervention and lack of automation. Developers face challenges in integrating their code changes smoothly into a shared repository, ensuring consistent builds, running tests reliably, and deploying updates quickly and consistently. The need for a streamlined and automated approach to software delivery has become paramount to meet the demands of modern development practices and agile methodologies. Hence, the problem at hand is to design and implement a CI/CD pipeline that automates the integration, testing, and delivery processes, enabling continuous integration of code changes, rigorous testing, and seamless deployment to production environments, thereby reducing errors, accelerating time-to-market, and enhancing overall software quality.

1.2 SCOPE

The scope of the CI/CD pipeline is comprehensive, covering every aspect of the software delivery lifecycle. It involves automating key processes such as code integration, testing, deployment, and monitoring. Continuous integration (CI) entails automatically building and testing code changes as they are committed to the version control system, ensuring early detection of integration issues and maintaining code quality. Continuous deployment (CD) focuses on automating the deployment process to various environments, including development, testing, staging, and production. Automated testing is a critical component within this scope, encompassing unit tests, integration tests, and end-to-end tests to validate the functionality, performance, and security of the application. Infrastructure as code (IaC) is integrated into the pipeline to provision and configure necessary resources consistently across environments, ensuring reliability and scalability. Additionally, monitoring and feedback mechanisms are implemented to provide real-time insights into the health and performance of deployed applications, enabling rapid detection and resolution of issues.

1.3 OBJECTIVE

The objective of a CI/CD pipeline is to revolutionize software delivery processes, aligning with modern agile development methodologies. By automating integration, testing, and deployment tasks, the pipeline aims to enhance efficiency, improve product quality, and accelerate time-to-market. Automated testing within the pipeline ensures that code changes meet quality standards before deployment, reducing the likelihood of errors and regressions.Consistency across development, testing, and production environments is maintained through infrastructure as code principles, minimizing deployment failures due to environmental discrepancies.

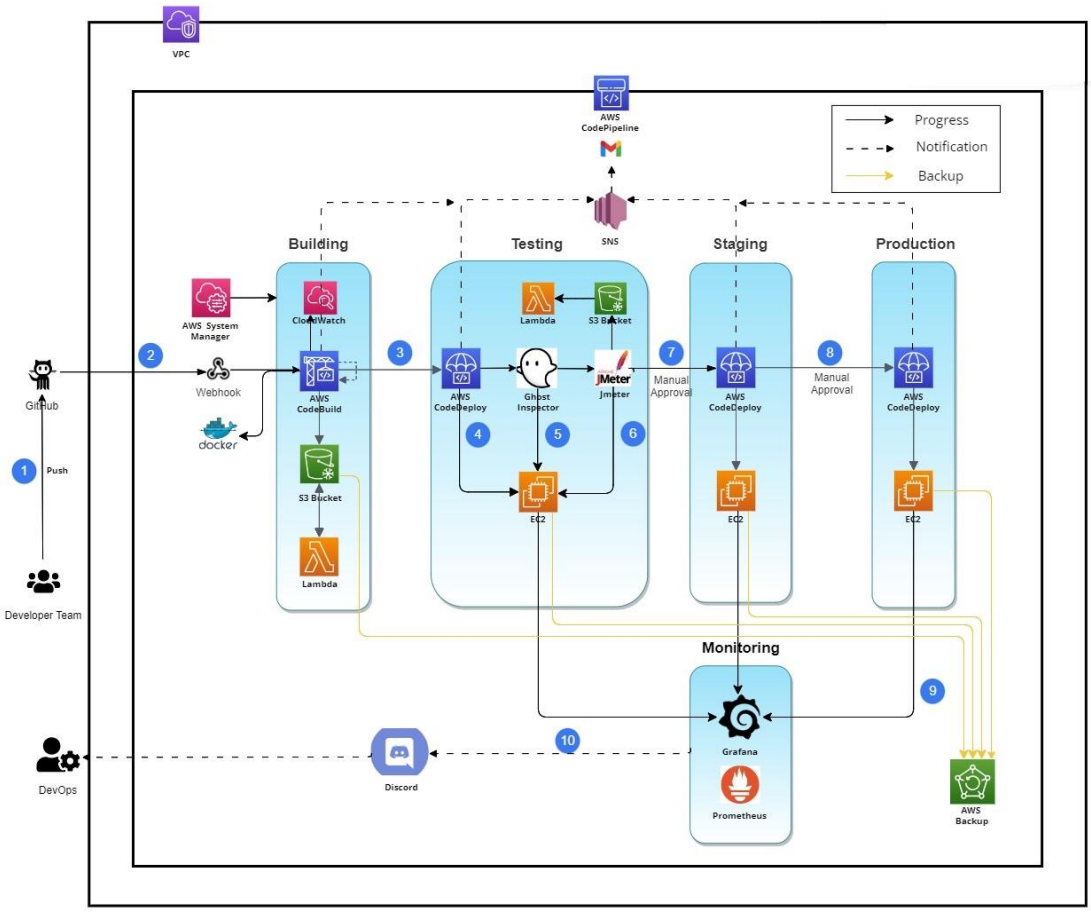


Fig. 1 Code Pipeline Flow Chart

2. BACKGROUND THEORY

2.1 AWS SERVICES

2.1.1 Code Pipeline

AWS CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service provided by Amazon Web Services (AWS). It automates the build, test, and deployment phases of your release process every time there is a code change, based on the release workflow defined.

The main features and components of AWS CodePipeline include:

1. Pipeline: A pipeline represents your CI/CD workflow and consists of a series of stages. Each stage can contain one or more actions, such as source, build, test, and deploy.

2. Source: This stage retrieves the source code from a version control repository such as AWS CodeCommit, GitHub, Bitbucket, or Amazon S3. It triggers the pipeline whenever there is a change in the source code repository.

3. Build: The build stage compiles the source code, runs unit tests, and packages the artifacts for deployment. AWS CodePipeline integrates with AWS CodeBuild or other third-party build services like Jenkins.

4. Test: The test stage executes automated tests, including unit tests, integration tests, and any other tests needed to validate the code changes. You can use AWS services like AWS CodeBuild, AWS Device Farm, or third-party testing tools.

5. Deploy: The deploy stage deploys the artifacts generated in the build stage to the target environment, such as AWS Elastic Beanstalk, Amazon ECS, AWS Lambda, or an on-premises server. It can also update AWS CloudFormation stacks or invoke AWS Step Functions.

6. Approval: Optionally, it can include manual approval actions between stages to require human intervention before proceeding with the deployment process.

7. Integration: AWS CodePipeline seamlessly integrates with other AWS services, enabling you to build end-to-end CI/CD pipelines. It provides features like artifact storage, versioning, and notifications through Amazon SNS or Amazon CloudWatch Events.

Overall, AWS CodePipeline simplifies and automates the software release process, enabling teams to deliver changes quickly and reliably while adhering to best practices in CI/CD.

2.1.2 Code Build

AWS CodeBuild is a fully managed continuous integration service provided by Amazon Web Services (AWS). It compiles source code, runs tests, and produces deployable artifacts without the need for provisioning or managing build servers. CodeBuild seamlessly integrates with other AWS services and popular source control repositories, allowing developers to focus on writing code while CodeBuild handles the build and test processes. It scales automatically to accommodate the size and complexity of builds, ensuring fast and reliable build execution.

2.1.3 Code Deploy

AWS CodeDeploy is a fully managed deployment service provided by Amazon Web Services (AWS) that automates software deployments to a variety of compute services such as Amazon EC2 instances, AWS Lambda functions, and on-premises servers. It simplifies the process of releasing new features, updates, and bug fixes to production environments, enabling developers to deploy applications quickly and reliably.

With AWS CodeDeploy, you can deploy code from various sources, including Amazon S3 buckets, GitHub repositories, or Bitbucket repositories. It allows to define deployment configurations that specify how the deployment should proceed, including deployment targets, deployment groups, and deployment settings.

CodeDeploy supports both in-place deployments, where the application is updated directly on the target instances, and blue/green deployments, where a new set of instances is provisioned with the updated application before swapping traffic from the old instances to the new ones.

2.1.4 S3 Bucket

An Amazon S3 bucket is a cloud storage resource provided by Amazon Web Services (AWS) for storing and retrieving data. It serves as a central repository for storing a wide variety of data types, including documents, images, videos, and application data. S3 buckets are highly scalable, durable, and secure, making them suitable for a wide range of use cases, from simple file storage to complex data analytics pipelines.

2.1.5 EC2 Instance

Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon Web Services (AWS) that enables you to launch and manage virtual servers, known as instances, in the cloud. EC2 instances provide scalable compute capacity and allow you to run a wide variety of applications, from simple web servers to complex data processing workloads.

2.1.6 SNS (Simple Notification Service)

Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service provided by Amazon Web Services (AWS) that enables you to send messages or notifications to a variety of endpoints, including email, SMS, mobile push, HTTP/HTTPS, and Amazon SQS queues. SNS simplifies the process of sending messages and allows you to decouple the sender from the receiver, making it ideal for building event-driven and distributed systems.

2.1.7 AWS Backup

AWS Backup is a comprehensive solution offered by Amazon Web Services (AWS) that simplifies data protection and management across various AWS services and on-premises environments. With AWS Backup, users can centrally manage backup policies, schedules, and retention settings through a unified console and API, streamlining the backup process. The service seamlessly integrates with popular AWS offerings such as Amazon EBS, RDS, DynamoDB, EFS, and Storage Gateway, allowing users to protect critical data stored in these services effortlessly. Automated backup and restore functionalities enable users to schedule recurring backups and easily recover data in the event of accidental deletion or corruption. Additionally, AWS Backup supports cross-region and cross-account backups, providing data redundancy and disaster recovery capabilities. With built-in encryption, monitoring, and compliance features, AWS Backup ensures data security and regulatory compliance, making it a reliable and cost-effective solution for organizations seeking to safeguard their data assets.

2.1 STAGES IN DEVELOPMENT & OPERATIONS

2.2.1 Backup of RDS in S3 bucket

RDS (Microsoft SQL) backup is taken in s3 bucket through local machine using **Microsoft SQL management Studio.**

Steps to configure RDS:

- Launch a New RDS Instance
- Click on "Create database".
- Select the "Standard create" method.
- Choose the Microsoft SQL Server engine.
- Select the edition of SQL Server you want to use (e.g., Express, Standard, Enterprise).
- Choose the version of SQL Server (e.g., SQL Server 2019, SQL Server 2017, SQL Server 2016).
- Specify other configuration details such as instance size, storage type, storage size, and database identifier.
- Provide a unique DB instance identifier.
- Set the master username and password for the SQL Server instance.
- Choose the DB instance class and allocate the desired amount of compute and memory resources.
- Specify the storage type (e.g., General Purpose SSD, Provisioned IOPS SSD) and storage size.
- Configure the VPC and subnet settings for the RDS instance.
- Define security groups and network settings to control inbound and outbound traffic to the RDS instance.
- Configure IAM roles and policies to manage permissions for accessing the RDS resources.
- Review the configuration settings to ensure they are accurate.
- Click on "Create database" to launch the RDS instance.

Create new bucket in S3 and give permission to IAM role of S3 full access.

Download Microsoft SQL Server Management Studio latest version and login to RDS Microsoft SQL Server using arn. Write the command to export the backup in S3 bucket.

All backup will be saved in rdbackup.bak in bucket.

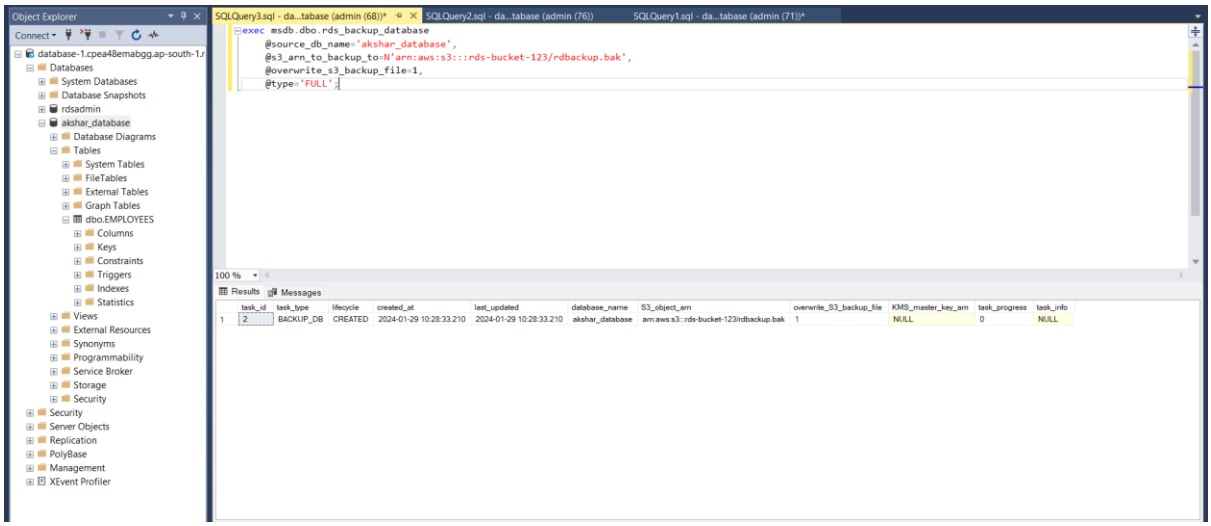


Fig. 2.2.1.1 Command to upload backup in S3 bucket

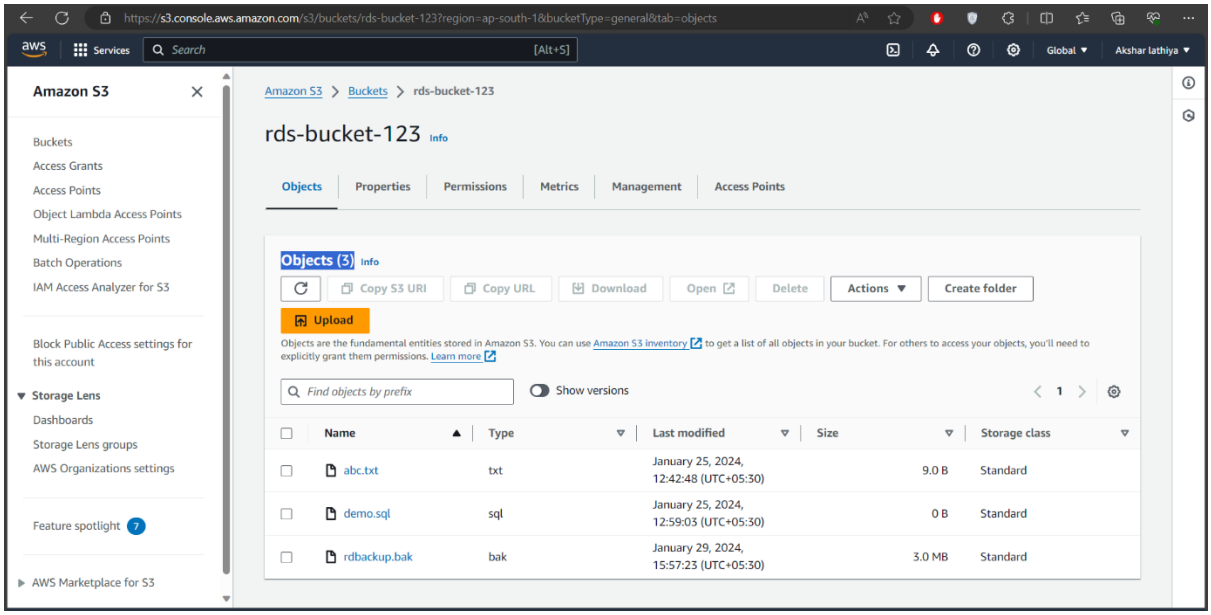


Fig. 2.2.1.2 Backup Uploaded in S3 bucket

2.2.2 Hosting proxy site on EC2 instance

The react static website was hosted on 3000 port of EC2 instance. Firstly, it was pushed on github and then it is downloaded on instance. Nginx server is being installed on instance. The configuration of site was stored in sites-enabled folder in nginx server where the particular domain was given for website and the changes were also made in host file of local machine.

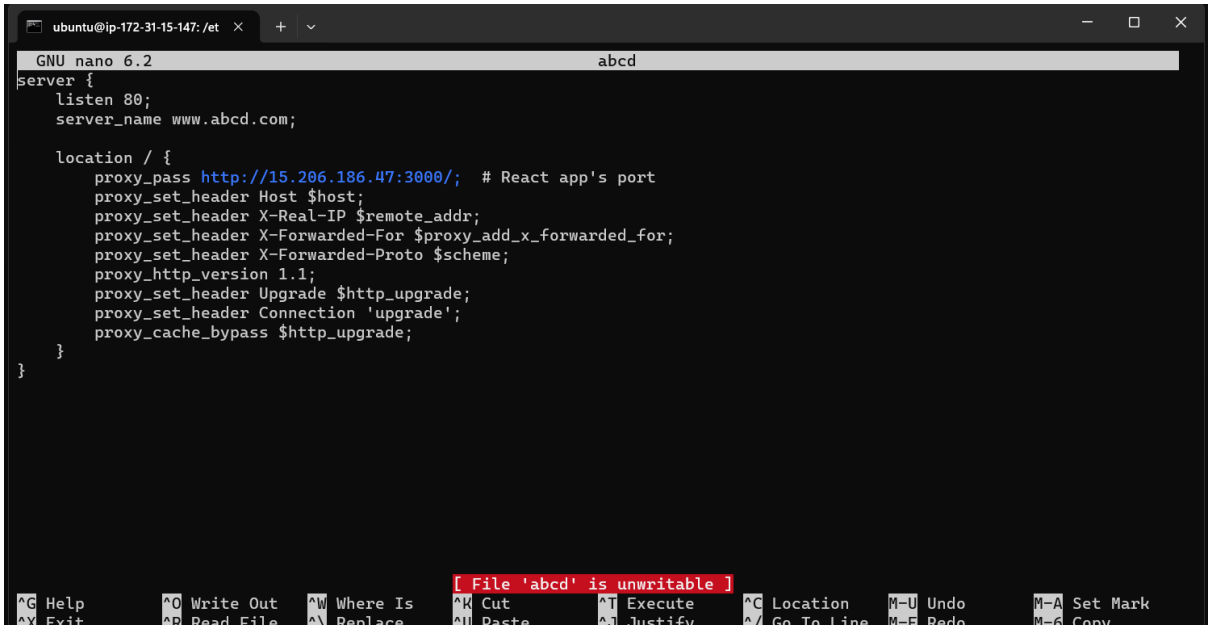


Fig. 2.2.2.1 Config File In Nginx Server

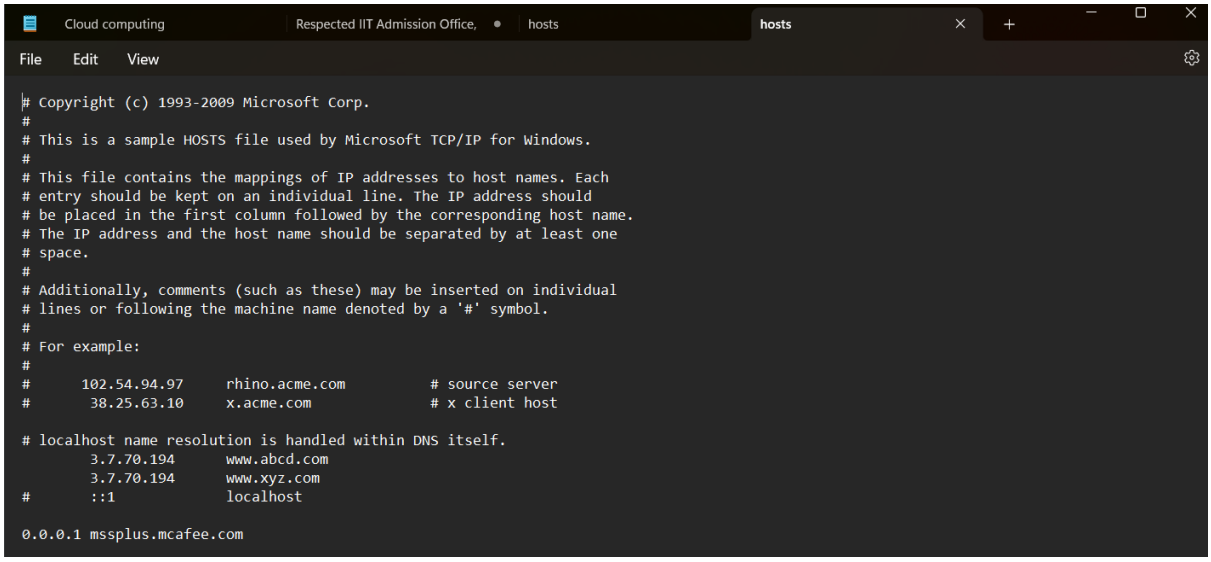


Fig. 2.2.2.2 Host File In Local Machine

2.2.3 Downloading SSL certificate of hosted React Site

SSL stands for Secure Sockets Layer. It's a standard security protocol used to establish an encrypted link between a web server and a browser. This link ensures that all data passed between the web server and browser remains private and integral.

The SSLforfree website provides free ssl certificate for website. The HTTP file should be uploaded in react project of public folder so they can verify the website security.

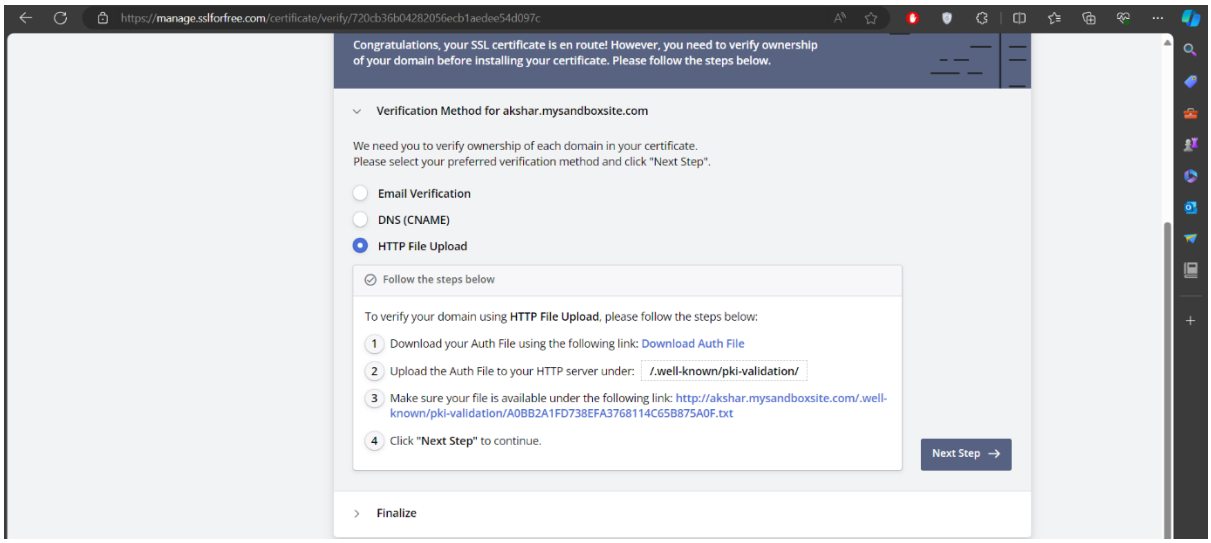


Fig. 2.2.3.1 SSLforfree Authentication For Certificate

```
README.md  node_modules  package-lock.json  package.json  public  src
root@ip-172-31-15-147:/home/ubuntu/react-app# cd public
root@ip-172-31-15-147:/home/ubuntu/react-app/public# mkdir .well-known
root@ip-172-31-15-147:/home/ubuntu/react-app/public# cd .well-known/
root@ip-172-31-15-147:/home/ubuntu/react-app/public/.well-known# mkdir pki-validation
root@ip-172-31-15-147:/home/ubuntu/react-app/public/.well-known# cd pki-validation/
root@ip-172-31-15-147:/home/ubuntu/react-app/public/.well-known/pki-validation# nano A0BB2A1FD738EFA3768114C65B875A0F.tx
t
root@ip-172-31-15-147:/home/ubuntu/react-app/public/.well-known/pki-validation# ls
A0BB2A1FD738EFA3768114C65B875A0F.txt
root@ip-172-31-15-147:/home/ubuntu/react-app/public/.well-known/pki-validation#
```

Fig. 2.2.3.2 Uploading HTTP File In Public Folder

```
C:\Users\91701\Downloads>scp -i "linux-aws.pem" C:\Users\91701\Downloads\ssl\ca_bundle.crt ubuntu@52.66.245.16:/home/ubu
ntu
ca_bundle.crt                                                 100% 2431    51.9KB/s   00:00

C:\Users\91701\Downloads>
```

Fig. 2.2.3.3 Uploading Certificate to EC2 instance

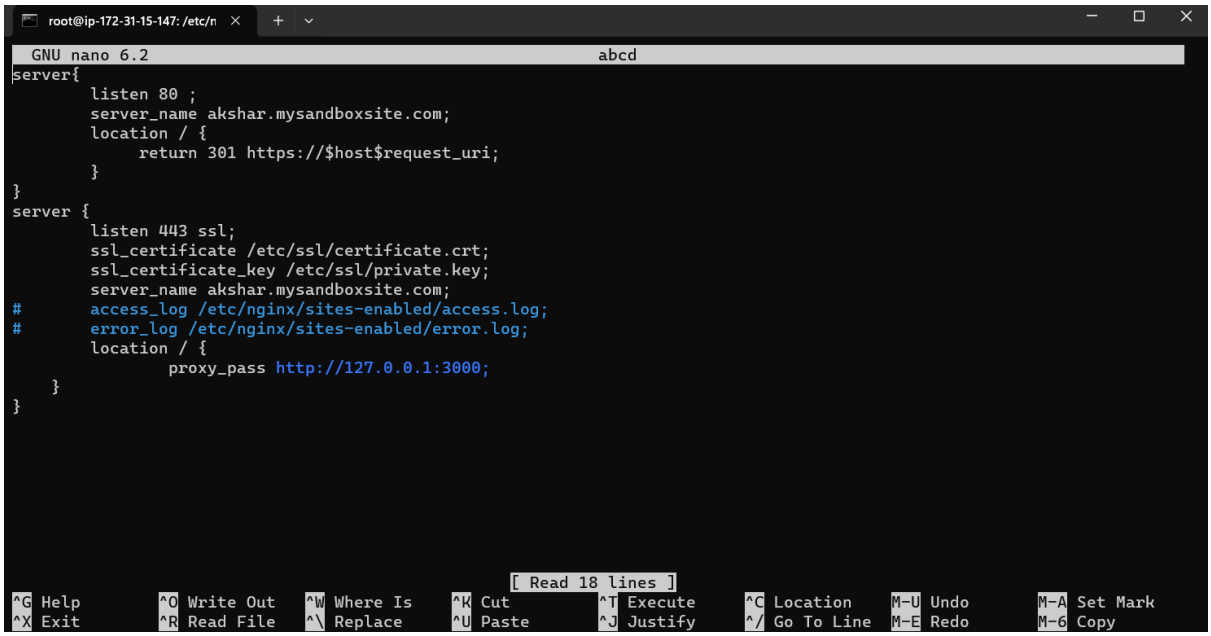


Fig. 2.2.3.4 Nginx Config File

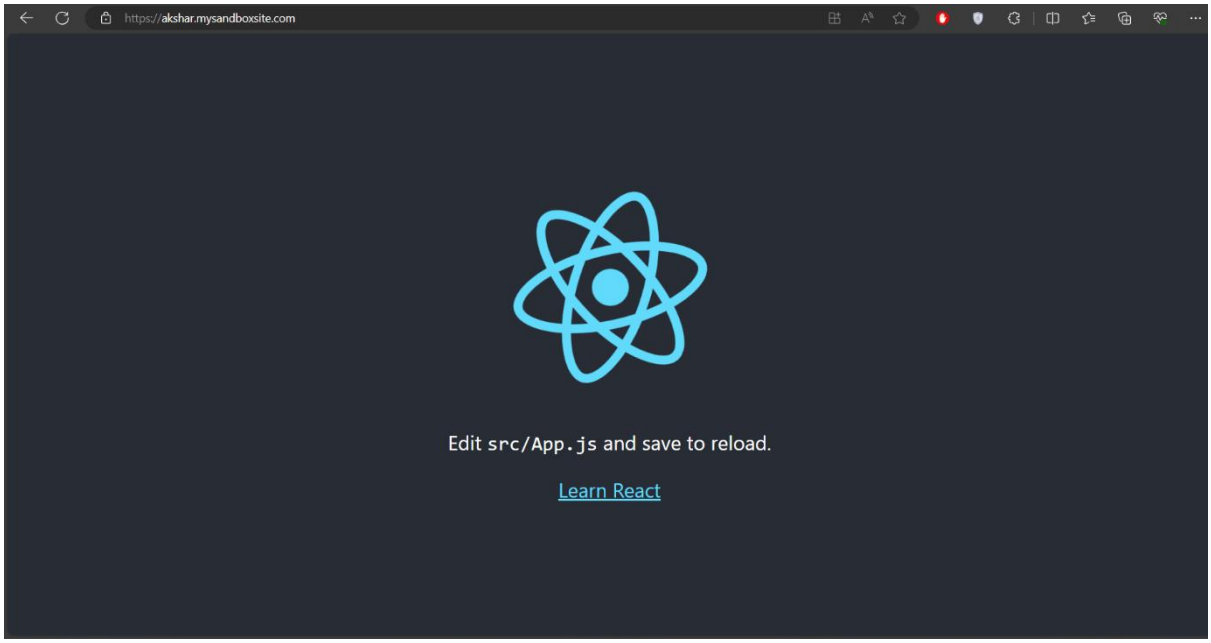


Fig. 2.2.3.5 Site With SSL Certificate

3. PROJECT PLANNING & SCHEDULING

3.1 AGILE

Agile methodologies are commonly applied in software development projects, including the development and management of CI/CD pipelines. Here's how you can adapt Agile principles and practices to the CI/CD pipeline development process:

- **Iterative Development:**
  - Break down the development of the CI/CD pipeline into smaller, manageable iterations called sprints. Each sprint focuses on delivering a specific set of features, improvements, or fixes to the pipeline.
- **User Stories and Backlog:**
  - Define user stories that capture the requirements and functionalities of the CI/CD pipeline from the perspective of its users (developers, testers, operations teams, etc.). Prioritize these user stories and maintain a backlog of items to be implemented in future sprints.
- **Sprint Planning:**
  - Conduct sprint planning meetings at the beginning of each sprint to select user stories from the backlog and define the tasks required to implement them. Estimate the effort required for each task and assign them to team members.
- **Daily Stand-ups:**
  - Hold daily stand-up meetings to discuss progress, obstacles, and plans for the day. Team members share updates on their tasks, raise any issues or blockers, and collaborate to resolve them.
- **Continuous Integration:**
  - Embrace the Agile principle of continuous integration by integrating code changes into the mainline repository frequently, preferably multiple times a day. Automate the process of building and testing code changes to ensure that they are integrated smoothly and without errors.
- **Continuous Deployment:**
  - Apply the concept of continuous deployment to automate the deployment of code changes to production-like environments as soon as they pass the required tests. Automate the provisioning and configuration of infrastructure to support rapid and reliable deployments.
- **Feedback and Adaptation:**
  - Solicit feedback from developers, testers, and end-users, throughout the development process. Use feedback to iterate on the design and implementation of the CI/CD pipeline, making adjustments as needed to meet evolving requirements and expectations.

- **Retrospectives:**
  - Hold sprint retrospectives at the end of each sprint to reflect on what went well, what could be improved, and what lessons were learned. Identify opportunities for process improvements and take actions to address any issues or concerns raised by the team.
- **Continuous Improvement:**
  - Foster a culture of continuous improvement within the team by encouraging experimentation, innovation, and learning. Regularly review and refine the CI/CD pipeline to incorporate new technologies, best practices, and lessons learned from previous sprints.
- **Collaboration and Communication:**
  - Promote collaboration and communication among team members, stakeholders, and other relevant parties involved in the development and operation of the CI/CD pipeline. Foster transparency, trust, and open dialogue to facilitate effective teamwork and decision-making.

3.1.1 Advantages of Agile model:

- **Flexibility and Adaptability:** Agile methodologies allow teams to respond quickly to changing requirements and market conditions. This flexibility is particularly beneficial in CI/CD pipelines, where rapid iteration and deployment are essential.
- **Incremental Delivery:** Agile promotes incremental delivery of features and improvements, allowing teams to release valuable updates to the CI/CD pipeline regularly. This approach enables faster time-to-market and continuous improvement of the pipeline.
- **Stakeholder Collaboration:** Agile encourages collaboration between development teams, operations teams, and other stakeholders involved in the CI/CD pipeline. By involving stakeholders throughout the development process, Agile ensures that the pipeline meets their needs and expectations.
- **Continuous Feedback:** Agile emphasizes the importance of continuous feedback from stakeholders, which helps identify issues early and guide the direction of development. In the context of CI/CD pipelines, feedback loops enable teams to iterate on the pipeline and improve its performance over time.
- **Reduced Risk:** Agile methodologies promote early and frequent testing, which helps identify and mitigate risks early in the development process. By delivering smaller, incremental changes, Agile reduces the likelihood of major failures and disruptions in the CI/CD pipeline.
- **Improved Quality:** Agile promotes continuous testing and integration, leading to higher software quality. By identifying and fixing issues early in the development process, Agile helps prevent defects from propagating through the CI/CD pipeline, resulting in more reliable releases.

3.1.2 Disadvantages of Agile model :

- **Complexity:** Agile methodologies can introduce complexity, especially in large-scale CI/CD pipelines with multiple teams and dependencies. Managing dependencies, coordinating releases, and ensuring consistency across teams can become challenging in Agile environments.
- **Resource Intensive:** Agile requires active participation from team members, stakeholders, and other parties involved in the CI/CD pipeline. This can be resource-intensive, particularly for teams with limited time, budget, or expertise.
- **Cultural Shift:** Adopting Agile methodologies often requires a cultural shift within organizations, including changes in mindset, practices, and workflows. Resistance to change and lack of buy-in from stakeholders can hinder the successful implementation of Agile in CI/CD pipelines.
- **Scope Creep:** Agile's focus on flexibility and adaptability can sometimes lead to scope creep, where the scope of the CI/CD pipeline expands beyond the original plan. Without proper controls and prioritization, this can result in delays, budget overruns, and reduced efficiency.
- **Documentation and Governance:** Agile places less emphasis on documentation and formal governance processes compared to traditional development methodologies. While this can improve agility and responsiveness, it may also lead to inconsistencies, lack of traceability, and compliance issues in CI/CD pipelines

4. PROJECT DEVELOPMENT & TESTING

4.1 CONTINUOUS INTEGRATION & DEVELOPMENT

Continuous Integration (CI) is a software development practice where members of a team integrate their work frequently, usually multiple times a day. Each integration is verified by automated builds and tests, allowing teams to detect problems early. Continuous Integration aims to improve the quality of software, reduce integration issues, and increase the speed of development.

4.1.1 Visual Studio code

Visual Studio, Microsoft's integrated development environment (IDE), plays a significant role across various stages of the CI/CD pipeline. In the development stage, Visual Studio serves as a robust environment for software creation, offering features for code editing, debugging, and version control. It has been used for developing react website and I have added linting tool named **ESlint** is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs.

Fig. 4.1.1 ESlint code file

4.1.2 GitHub

GitHub, a pivotal platform for version control and collaborative development, plays a pivotal role across the spectrum of the CI/CD pipeline. In the development phase, GitHub serves as a centralized repository where developers store and manage their codebase using Git.

Its version control features enable developers to track changes, manage branches, and facilitate seamless collaboration among team members. It also have branch protection so developers can not directly push to main branch rather they have to first, take approval from administrator that changes made are valid.

Only administrator , is allowed to make changes directly to main branch. All other developers have to first make new branch and add their respective changes in that branch which is later verified by administrator.

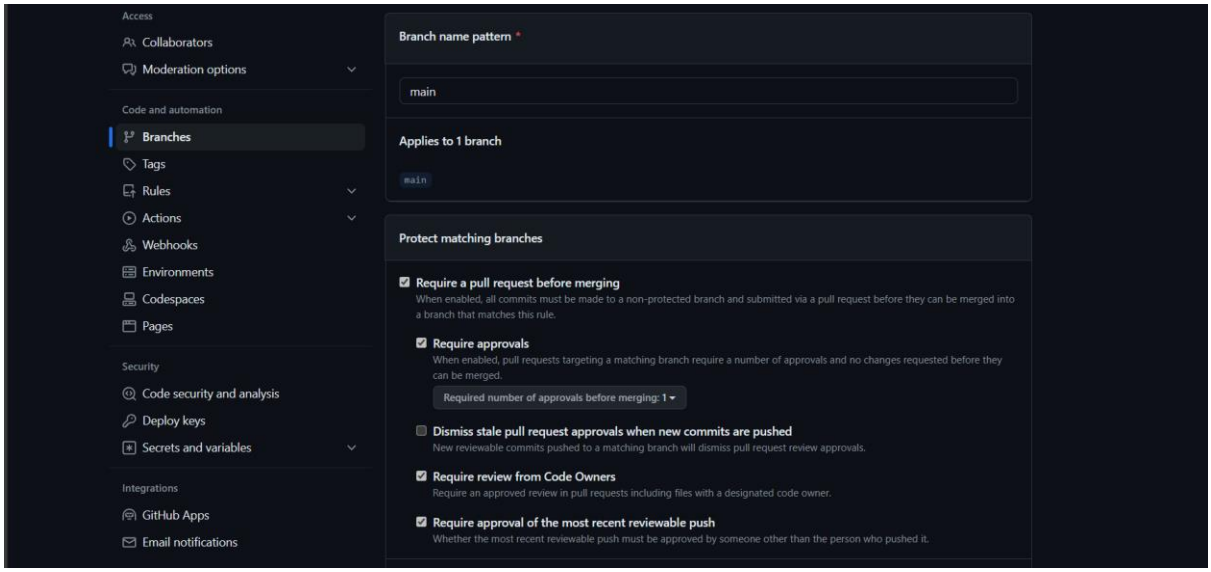


Fig. 4.1.2 Branch protection

4.1.3 Code build

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. It eliminates the need to provision, manage, and scale your own build servers.

Steps to create build project:

- Go to the AWS Management Console and navigate to the CodeBuild service.
- Click on "Create build project".
- Enter a name for your project and optionally provide a description.
- Choose the source provider (e.g., AWS CodeCommit, GitHub, Bitbucket).
- Configure the source settings (e.g., repository URL, branch).
- Choose the environment for your build (e.g., operating system, runtime, compute type).
- Configure build specifications: You can either provide a buildspec file in your source repository or define build commands directly in the console.
- Choose the service role for CodeBuild: Either create a new service role or select an existing one. This role should have the necessary permissions to access AWS resources required during the build process.
- Review your configuration and click "Create build project".

The code build will be triggered in pipeline after running successfully source in pipeline . It will build a EC2 instance according to predefined configuration to test the project. The buildspec file will be run and it will firstly install npm on machine , the react app will be tested by command npm run build.

After successfully testing , the docker is installed and the docker image is created and pushed to docker repository. The necessary information for login and pushing image is stored in AWS system manager , it is accessed using parameter store in buildspec file . Then after successful pushing image in repository , artifacts are stored in S3 bucket. Lambda function will delete old artifacts file when new is added.

Buildspec Edit

```yaml
version: 0.2

env:
  #variables:
     # key: "value"
     # key: "value"
  parameter-store:
       DOCKER_USERNAME: /react-app/username
       DOCKER_PASSWORD: /react-app/password
       DOCKER_URL: /react-app/url

  #secrets-manager:
     # key: secret-id:json-key:version-stage:version-id
     # key: secret-id:json-key:version-stage:version-id
  #exported-variables:
     # - variable
     # - variable
  #git-credential-helper: yes
#batch:
  #fast-fail: true
  #build-list:
  #build-matrix:
  #build-graph:
phases:
  install:
    runtime-versions:
       nodejs: 20
      # name: version
    commands:
       - npm install
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
       - npm run build
       - echo "Building Docker Image"
       - echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_USERNAME" --password-stdin "$DOCKER_URL"
       - docker build -t "$DOCKER_URL/$DOCKER_USERNAME/react-app:latest" .
       - docker push "$DOCKER_URL/$DOCKER_USERNAME/react-app:latest"

  post_build:
    commands:
       - echo "Build successfull"
      # - command
#reports:
  #report-name-or-arn:
    #files:
      # - location
      # - location
    #base-directory: location
    #discard-paths: yes
    #file-format: JunitXml | CucumberJson
artifacts:
  files:
     - '**/*'


  #name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: build
#cache:
  #paths:
    # - paths
```

Fig. 4.1.3.1 buildspec.yml file

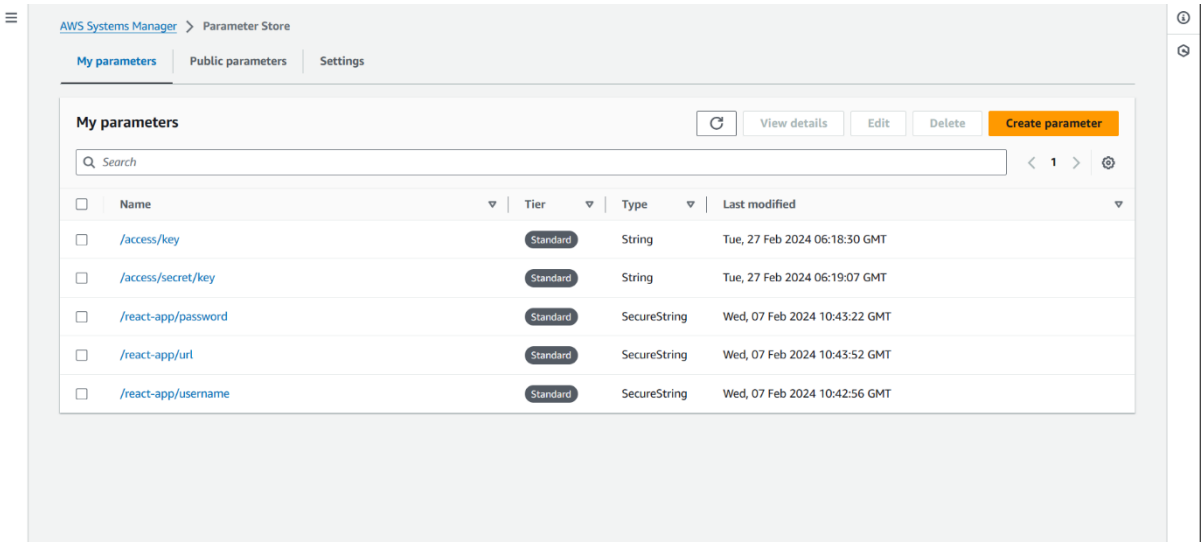


Fig. 4.1.3.2 AWS System Manager

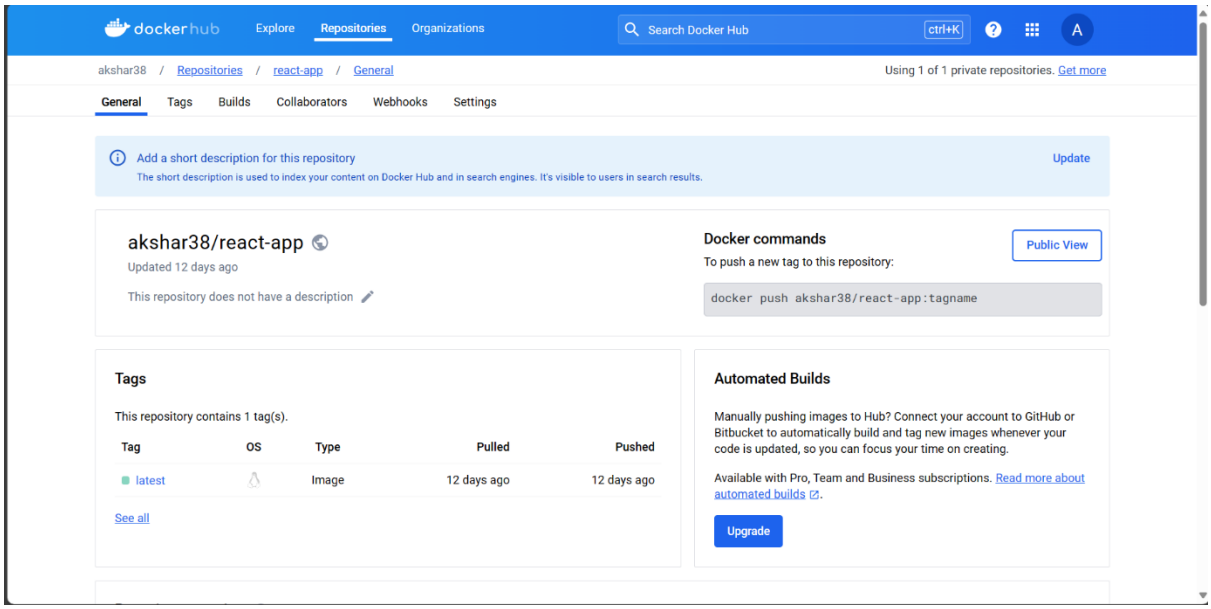


Fig. 4.1.3.3 Docker Repository

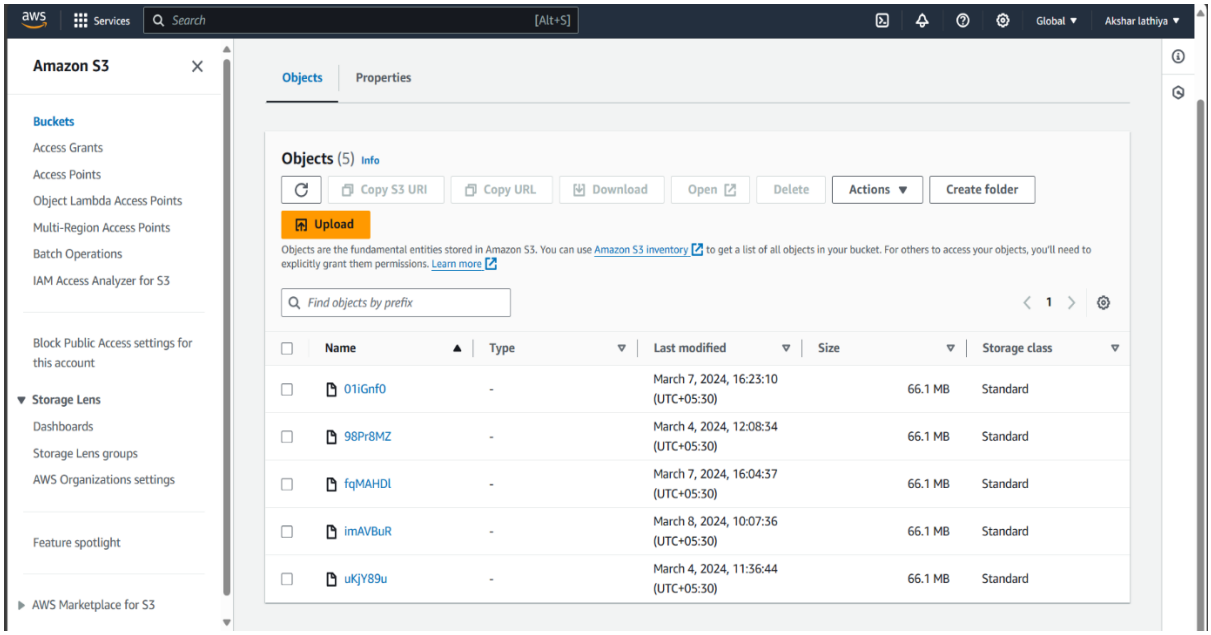


Fig. 4.1.3.4 Build Artifacts

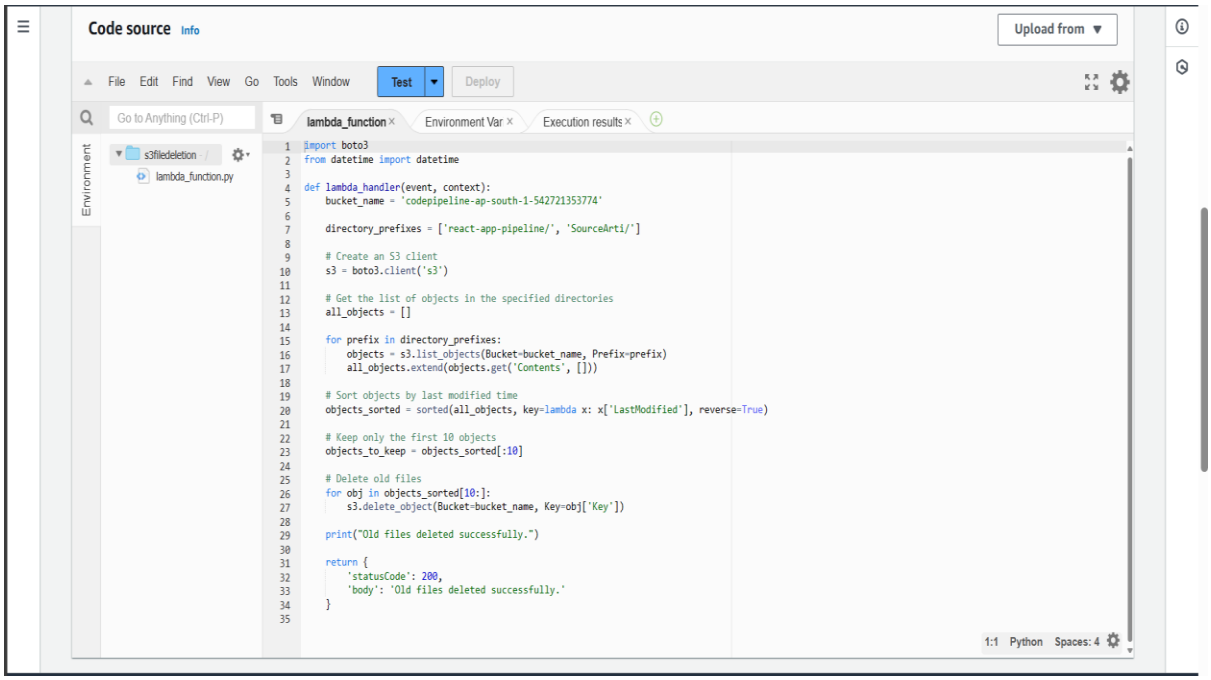


Fig. 4.1.3.5 Lambda Function Code

4.2 CONTINUOUS TESTING

4.2.1 UI-Testing

UI testing, also known as User Interface testing or GUI testing, is a type of software testing that focuses on verifying the graphical user interface (GUI) of an application. The purpose of UI testing is to ensure that the application's interface behaves as expected and meets the specified requirements, providing a smooth and user-friendly experience for the end-users.

4.2.1.1 Ghost Inspector

Ghost Inspector is a cloud-based automated testing tool designed specifically for website and web application testing. It provides users with a comprehensive platform to create, manage, and execute automated tests to validate the functionality, performance, and appearance of web pages and applications. With Ghost Inspector, users can record their interactions with web pages using a browser extension, capturing actions such as clicks, form submissions, and navigation. These interactions are then converted into test steps that can be replayed as automated tests. The tool supports various types of tests, including functional tests, regression tests, and visual regression tests, allowing users to ensure the integrity of their web applications across different browsers and environments. Additionally, Ghost Inspector offers features such as scheduled testing, integration with CI/CD pipelines, and detailed reporting and analysis, enabling teams to incorporate automated testing seamlessly into their development workflows. Overall, Ghost Inspector streamlines the process of automated testing for web applications, empowering teams to deliver high-quality software with confidence and efficiency.

Steps to configure Ghost Inspector:

- **Sign Up for an Account:**
  - Visit the Ghost Inspector website and sign up for an account. You can choose from different pricing plans based on your testing needs. Once signed up, log in to your account.
- **Create a New Test Suite:**
  - After logging in, navigate to the dashboard and click on "New Suite" to create a new test suite. A test suite is a collection of related tests that you want to run together, such as tests for a specific web application or feature.
- **Create a New Test:**
  - Within the test suite, click on "New Test" to create a new test. Give your test a descriptive name and specify the URL of the web page or application you want to test.
- **Record Test Steps:**
  - Use the Ghost Inspector browser extension (available for Chrome and Firefox) to record your interactions with the web page. Perform the actions you want to test, such as clicking buttons, filling out forms, and navigating between pages. The extension will capture your actions as test steps.

- **Edit Test Steps :**
  - After recording your test steps, you can edit them as needed. You can rearrange steps, add assertions to verify expected outcomes, and customize settings such as timeouts and delays.
- **Save and Run Test:**
  - Once you're satisfied with your test, save it and run it to verify that it executes successfully. Ghost Inspector will open a browser window and perform the recorded actions on the specified URL. You can monitor the test execution in real-time and view the results afterward.
- **Review Test Results:**
  - After the test completes, review the results to see if any steps failed. Ghost Inspector provides detailed logs, screenshots, and error messages to help you diagnose issues and troubleshoot failures.
- **Integrate with CI/CD Pipeline:**
  - To incorporate Ghost Inspector into your CI/CD pipeline, you can use the Ghost Inspector API or integrations with code pipeline service. Set up automated tests to run as part of your build and deployment process for continuous validation.
- **Manage and Maintain Tests:**
  - As your application evolves, regularly update and maintain your tests to ensure they remain accurate and effective. Modify tests as needed to accommodate changes in the application's functionality or user interface.

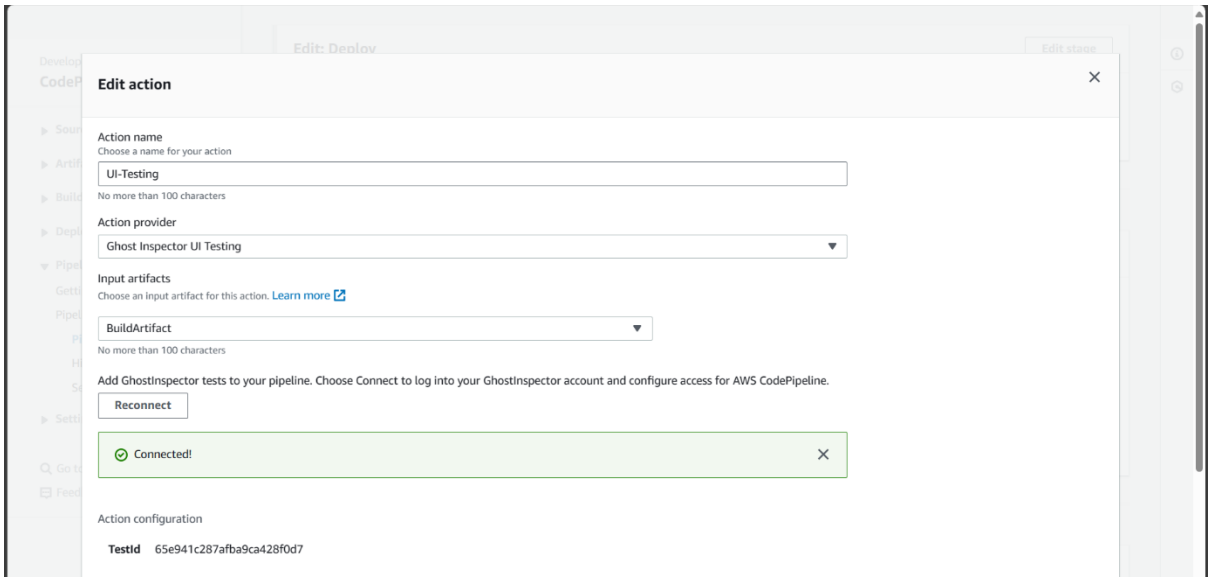


Fig. 4.2.1.1.1 Pipeline Connection with Ghost Inspector

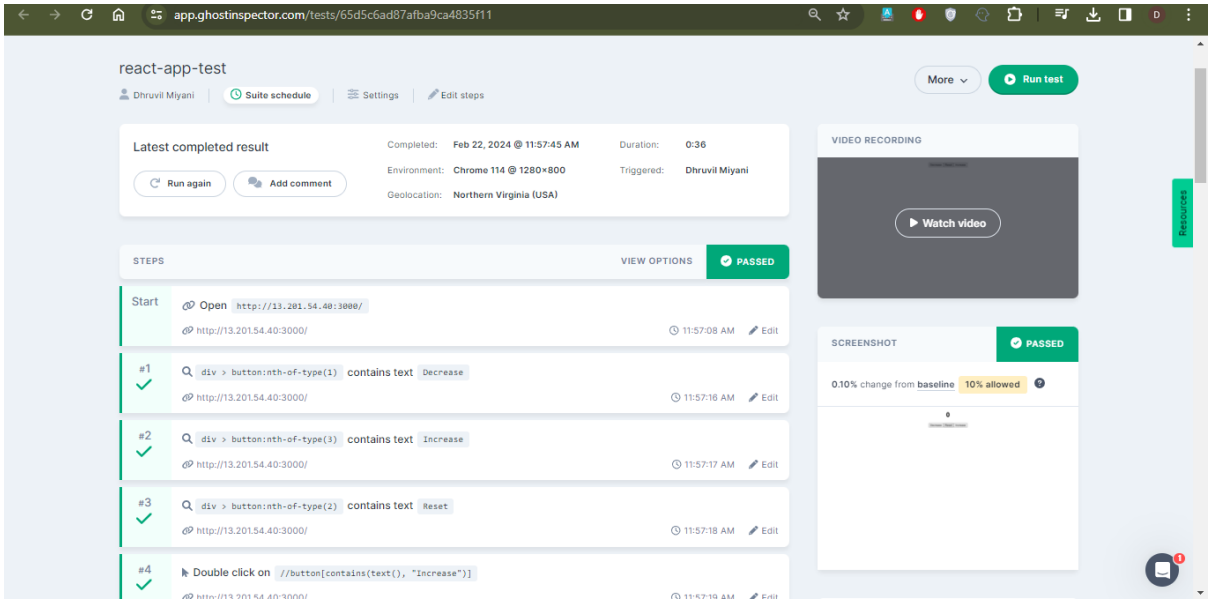


Fig. 4.2.1.1.2 Ghost Inspector Test Suite

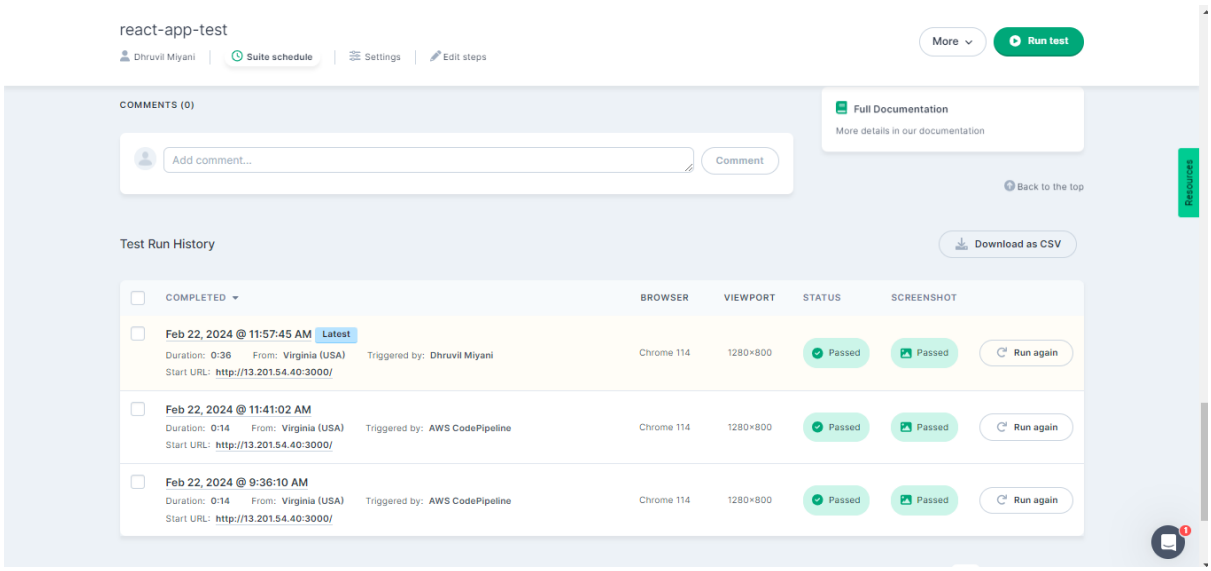


Fig. 4.2.1.1.3 Ghost Inspector Test Result

4.2.2 Load Testing

A load test of a website is a type of performance testing that evaluates how well a web application or website performs under a specific load or traffic volume. The purpose of load testing is to determine the application's capacity to handle concurrent users and transactions without experiencing performance degradation or downtime.

During a load test, simulated users (often referred to as virtual users or VUs) interact with the website by performing various actions, such as browsing pages, submitting forms, and downloading files. The load testing tool generates a predefined number of concurrent VUs and distributes them across the website to simulate realistic user behaviour.

4.1.1.1 JMeter

Apache JMeter is an open-source load testing tool developed by the Apache Software Foundation. It is widely used for performance testing of web applications, APIs, and other services. JMeter is Java-based and provides a user-friendly graphical interface for creating and running load tests.

Steps to configure JMeter:

- **Download and Install JMeter:**
  - Go to the Apache JMeter website (https://jmeter.apache.org/) and download the latest version of JMeter.
  - Extract the downloaded archive to a directory on your computer.
  - JMeter does not require installation; you can run it directly from the extracted folder.
- **Start JMeter:**
  - Navigate to the bin directory within the extracted JMeter folder.
  - Run the JMeter executable file (jmeter.bat for Windows or jmeter.sh for Unix/Linux) to start the JMeter GUI.
- **Create a Test Plan:**
  - In the JMeter GUI, create a new test plan by selecting "File" > "New".
  - Add elements to your test plan such as Thread Group, HTTP Request Sampler, and Listeners.
  - The Thread Group defines the number of virtual users, ramp-up period, and test duration.
  - The HTTP Request Sampler represents the web request to be sent to the server.
- **Configure Thread Group:**
  - Right-click on the Test Plan and select "Add" > "Threads" > "Thread Group".
  - Configure the Thread Group properties such as the number of threads (virtual users), ramp-up period (time to start all threads), and loop count (number of iterations).

- **Add HTTP Request Sampler:**
  - Right-click on the Thread Group and select "Add" > "Sampler" > "HTTP Request".
  - Configure the HTTP Request Sampler properties such as the server name or IP address, port number, protocol (HTTP or HTTPS), and path.
- **JMX File:**
  - Save the file having .jmx extension and copy it into project.

Load Test is carried out in code build service where the JMeter is firstly installed on EC2 Instance and run the .jmx file using JMeter command . The result are generated in csv file and dashboard which are uploaded in S3 bucket.

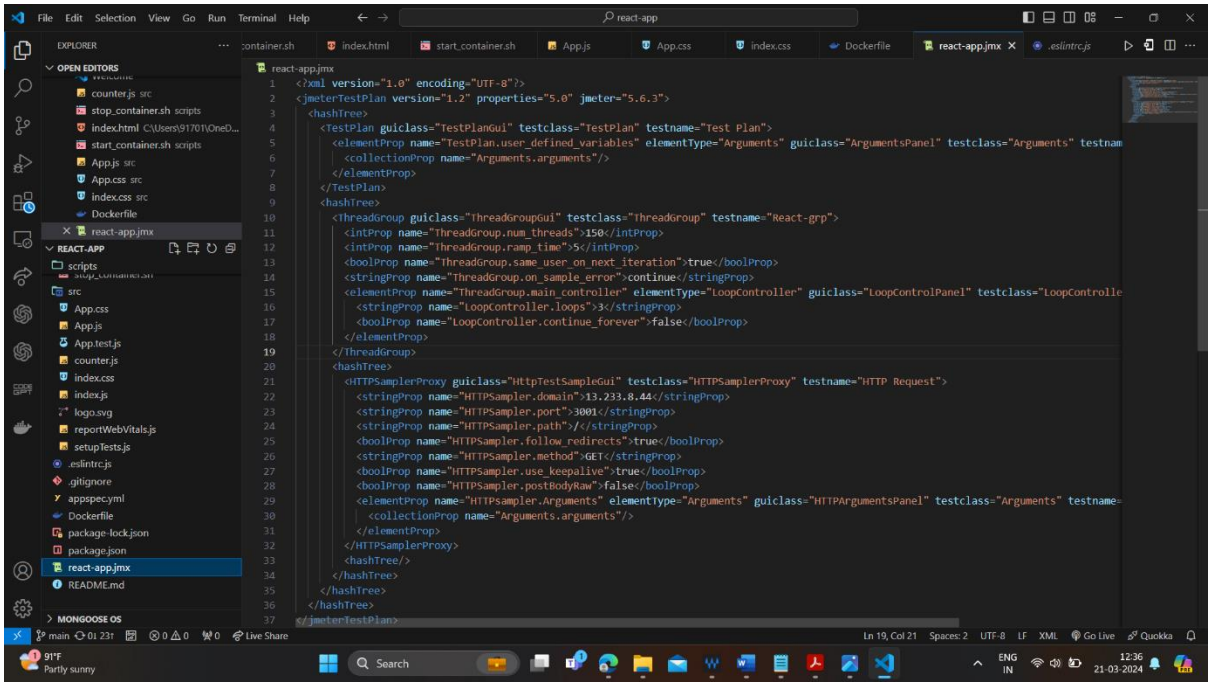


Fig. 4.2.2.1.1 JMeter Config File

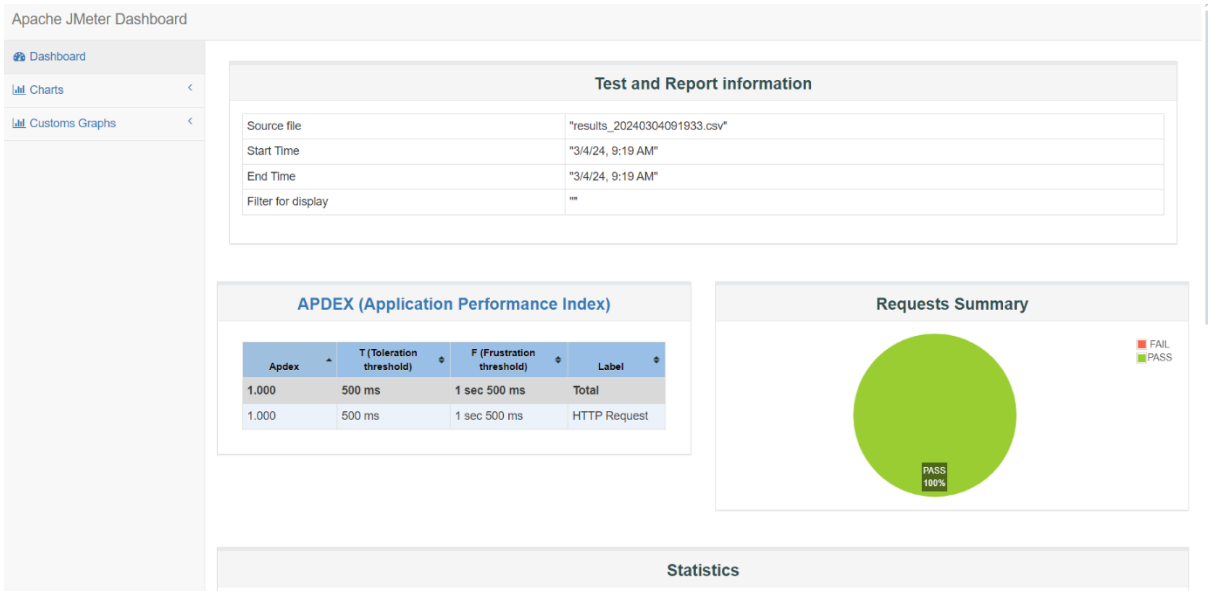
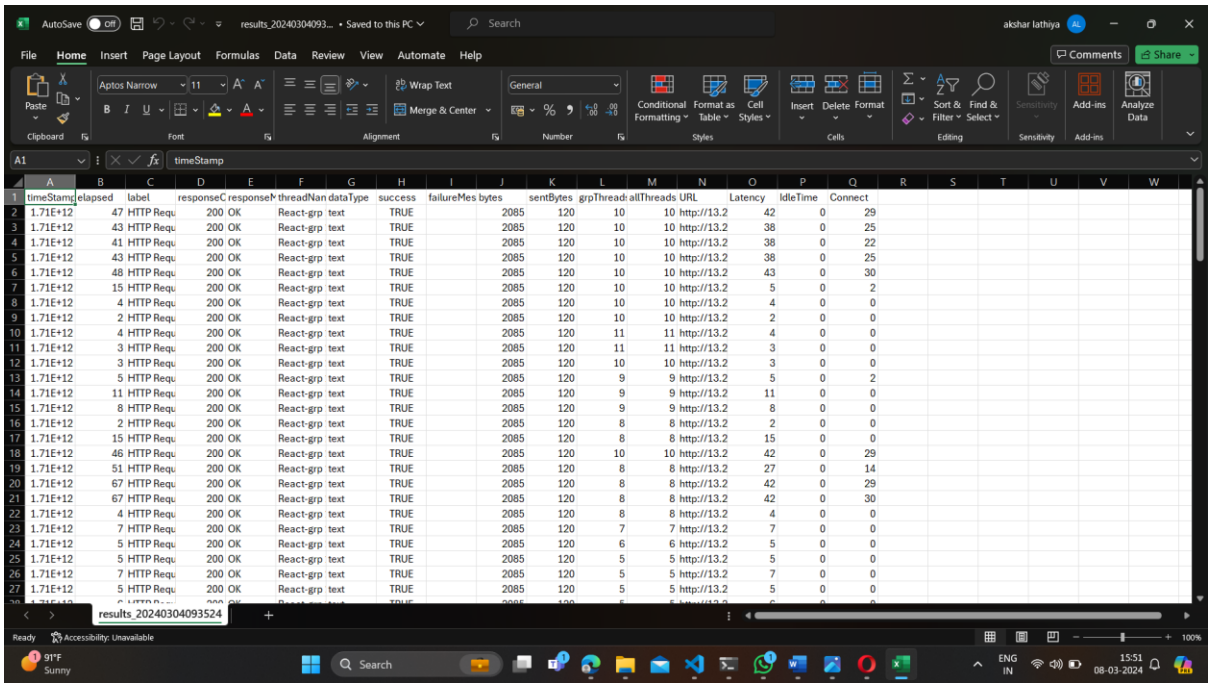
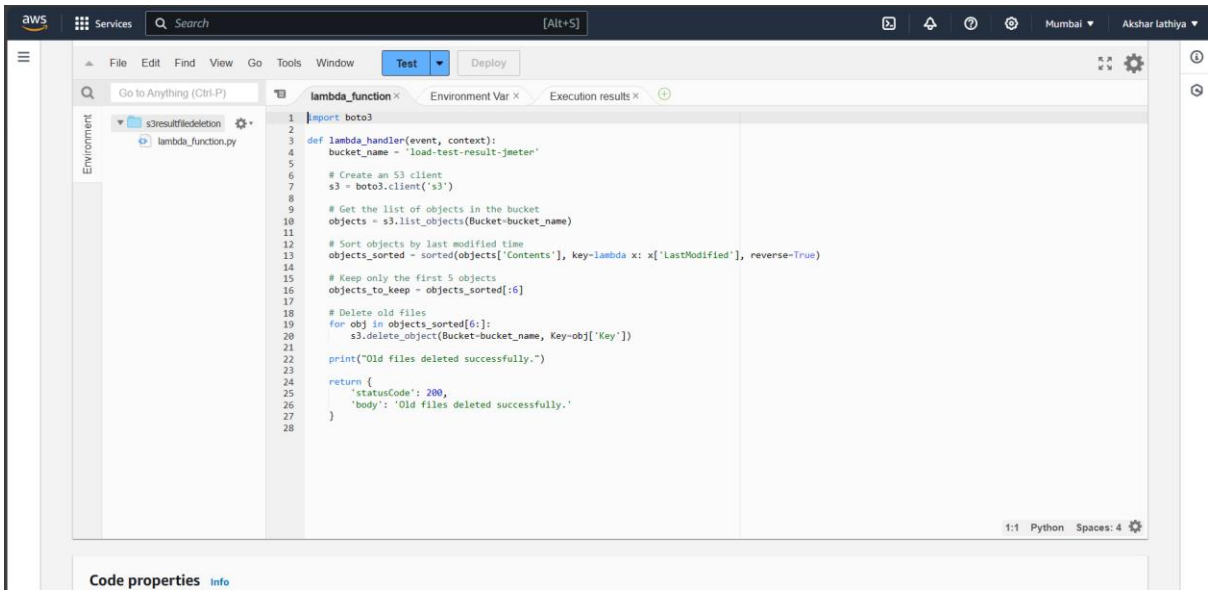


Fig. 4.2.2.1.2 Dashboard of Result Generated from JMeter Test



timeStamp	elapsed	label	responseC	responseM	threadNan	dataType	success	failureMes	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect
1.71E+12	47	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	42	0	29
1.71E+12	43	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	38	0	25
1.71E+12	41	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	38	0	22
1.71E+12	43	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	38	0	25
1.71E+12	48	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	43	0	30
1.71E+12	15	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	5	0	2
1.71E+12	4	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	4	0	0
1.71E+12	2	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	2	0	0
1.71E+12	4	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	11	11	http://13.2	4	0	0
1.71E+12	3	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	11	11	http://13.2	3	0	0
1.71E+12	3	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	3	0	0
1.71E+12	5	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	9	9	http://13.2	5	0	2
1.71E+12	11	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	9	9	http://13.2	11	0	0
1.71E+12	8	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	9	9	http://13.2	8	0	0
1.71E+12	2	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	8	8	http://13.2	2	0	0
1.71E+12	15	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	8	8	http://13.2	15	0	0
1.71E+12	46	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	10	10	http://13.2	42	0	29
1.71E+12	51	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	8	8	http://13.2	27	0	14
1.71E+12	67	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	8	8	http://13.2	42	0	29
1.71E+12	67	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	8	8	http://13.2	42	0	30
1.71E+12	4	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	8	8	http://13.2	4	0	0
1.71E+12	7	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	7	7	http://13.2	7	0	0
1.71E+12	5	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	6	6	http://13.2	5	0	0
1.71E+12	5	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	5	5	http://13.2	5	0	0
1.71E+12	7	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	5	5	http://13.2	7	0	0
1.71E+12	5	HTTP Requ	200	OK	React-grp	text	TRUE		2085	120	5	5	http://13.2	5	0	0

Fig. 4.2.1.1.3 csv File Output



```
import boto3

def lambda_handler(event, context):
    bucket_name = 'load-test-result-jmeter'

    # Create an S3 client
    s3 = boto3.client('s3')

    # Get the list of objects in the bucket
    objects = s3.list_objects(Bucket=bucket_name)

    # Sort objects by last modified time
    objects_sorted = sorted(objects['Contents'], key=lambda x: x['LastModified'], reverse=True)

    # Keep only the first 5 objects
    objects_to_keep = objects_sorted[:6]

    # Delete old files
    for obj in objects_sorted[6:]:
        s3.delete_object(Bucket=bucket_name, Key=obj['Key'])

    print("Old files deleted successfully.")

    return {
        'statusCode': 200,
        'body': 'Old files deleted successfully.'
    }
```

Fig. 4.2.1.1.4 Lambda Function Code For S3 file Deletion

5. PROJECT DEPLOYMENT AND MONITORING

5.1 Continuous Deployment

Continuous Deployment (CD) is a progressive software development approach where code changes are seamlessly and automatically deployed to production environments without the need for manual intervention, provided they pass a battery of automated tests and validation processes. This methodology extends the principles of Continuous Integration (CI), where code changes are integrated frequently into a shared repository and verified through automated builds and tests.

In the Continuous Deployment workflow, developers commit their code changes to a version control system, like Git, where these changes are regularly merged into a primary branch such as 'master' or 'mainline'. Upon each commit or merge, an automated build process is initiated. This build process compiles the code, executes unit tests, and conducts other automated checks to ensure the integrity of the changes and to guard against introducing regressions.

5.1.1 Code Deploy

AWS CodeDeploy is a fully managed deployment service provided by Amazon Web Services (AWS) that automates the process of deploying applications to a variety of compute environments including Amazon EC2 instances, AWS Lambda functions, and instances running on-premises. It facilitates continuous deployment by automating code deployments and enabling you to release new features and updates rapidly and reliably.

Steps to configure code deploy:

- Create an IAM Service Role:
  - Before you begin, ensure you have an IAM service role with the necessary permissions for AWS CodeDeploy. This role allows CodeDeploy to access AWS resources during the deployment process.
  - You can create a new IAM service role specifically for CodeDeploy or use an existing one.
  - Ensure the IAM role has permissions for CodeDeploy actions, access to the S3 bucket where application revisions are stored, and permissions to interact with the EC2 instances or other compute resources targeted for deployment.
- Prepare Your Application:
  - Package your application code and any necessary deployment artifacts into an archive format (e.g., .zip, .tar.gz).
  - Upload the application archive to an S3 bucket. Ensure that the S3 bucket is accessible to AWS CodeDeploy and that the IAM service role has permissions to access the bucket

- Create an Application:
  - Navigate to the AWS CodeDeploy console.
  - Click on "Create application".
  - Enter a name and optionally a description for your application.
- Create Deployment Groups:
  - After creating an application, create one or more deployment groups within the application.
  - Deployment groups define the target environment and specify the instances or Amazon ECS services where the application will be deployed.
  - Define deployment group settings such as deployment type, deployment configuration, load balancer settings (if applicable), and auto-scaling group settings (if deploying to EC2 instances).
- Create a Deployment Configuration (Optional):
  - You can create custom deployment configurations to specify the deployment settings, such as the deployment strategy, minimum healthy hosts, and deployment alarm thresholds.
  - If you don't create a custom deployment configuration, AWS CodeDeploy uses the default deployment configuration.

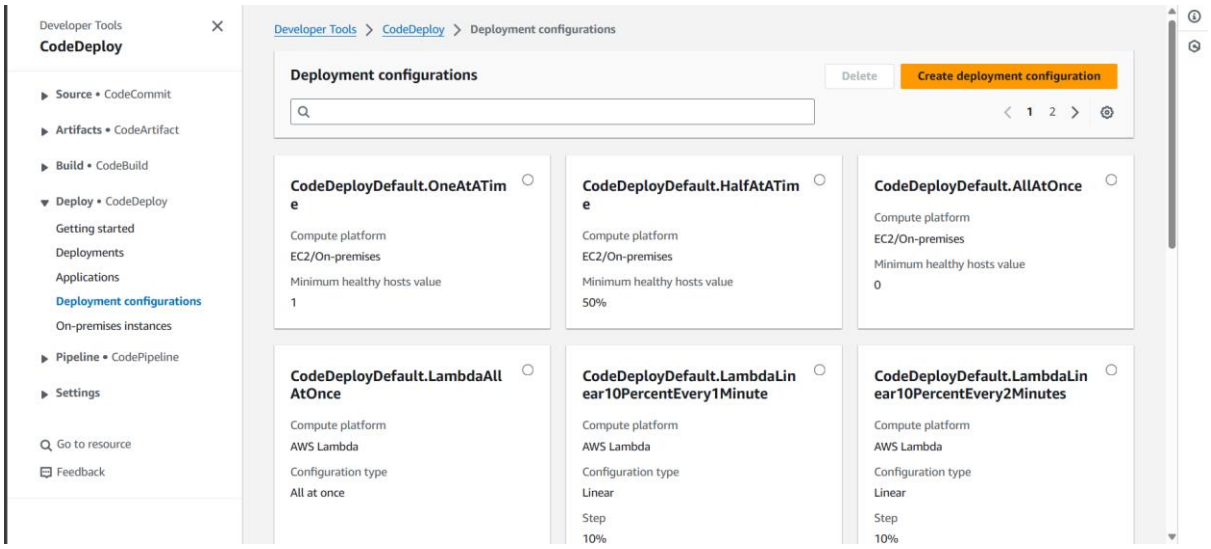


Fig. 5.1.1.1 Deployment Configuration

- Create a Deployment:
  - Once your application, deployment groups, and optionally deployment configuration are set up, you can create a deployment.
  - Select the application and deployment group, choose the revision (application version) stored in the S3 bucket, and specify any deployment configuration settings.
  - Start the deployment process.

- Monitor Deployment Progress:
  - Monitor the deployment progress in the AWS CodeDeploy console.
  - View deployment logs and details to track the status of each deployment step and any errors encountered during deployment.
- Verify and Test Deployment:
  - After the deployment is complete, verify that the application is successfully deployed to the target environment.
  - Perform any necessary testing to ensure that the application functions as expected in the deployed environment.

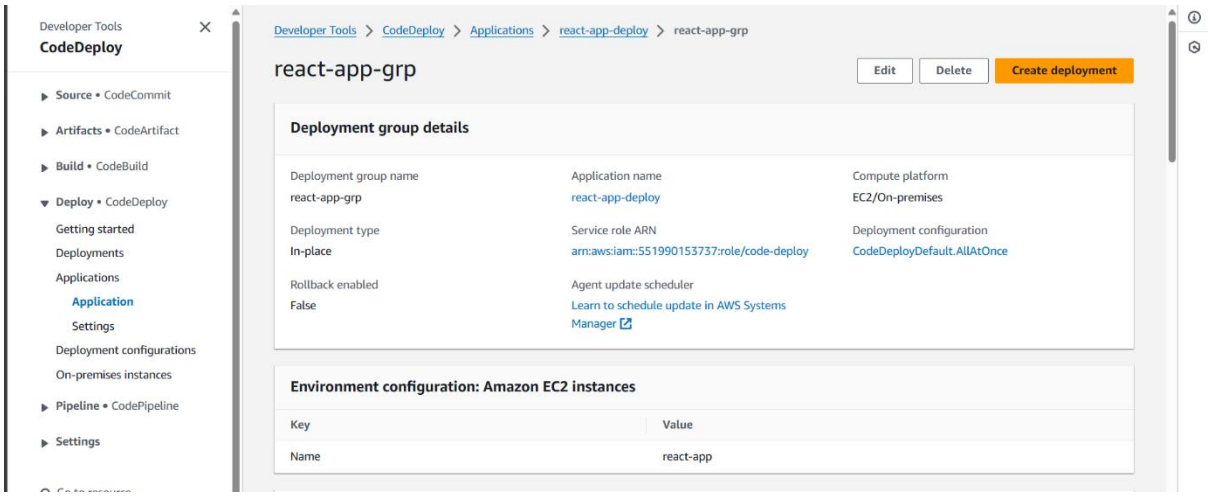


Fig. 5.1.1.2 Deployment Application

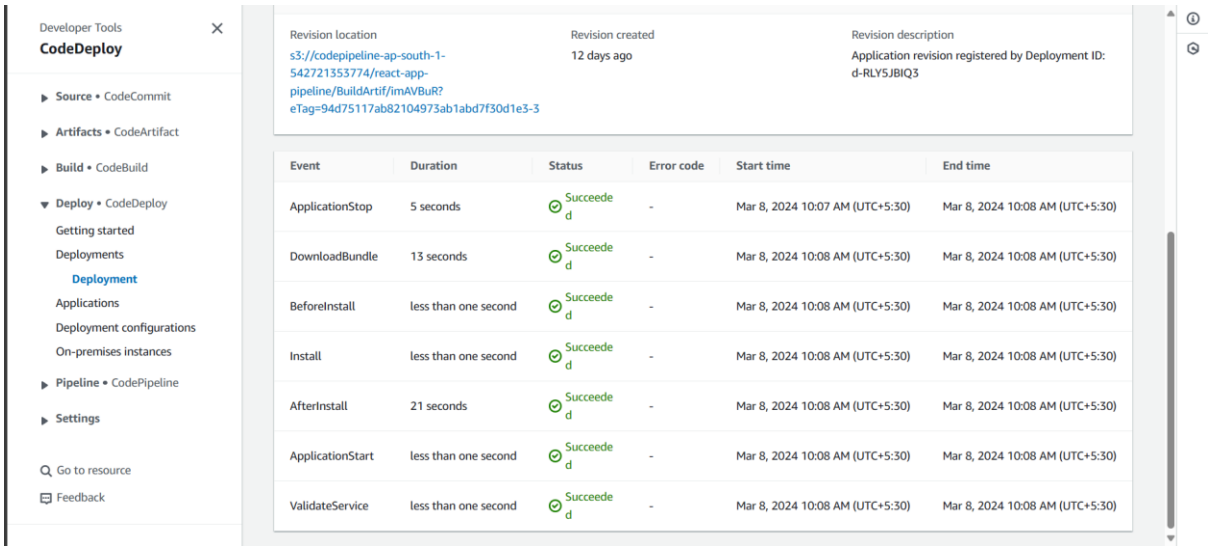


Fig. 5.1.1.3 Lifecycle Events of Code Deployment

5.1.2 Code deployment in EC2 Instance

After configuring code deploy service , the code deploy-agent should be installed in EC2 instance so the deployment can be successfully work.

It involves several steps to install code deploy-agent in ubuntu instance:

- Sign in to the instance.
- Enter the following commands, one after the other:
  - **sudo apt update**
  - **sudo apt install ruby-full**
  - **sudo apt install wget**
- Enter the following command:
  - **wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install**

    bucket-name is the name of the Amazon S3 bucket that contains the CodeDeploy Resource Kit files for your region, and region-identifier is the identifier for your region.

    For example:
    https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install
- Enter the following command:
  - **chmod +x ./install**
- To install the latest version of the CodeDeploy agent on any supported version of Ubuntu Server except 20.04:
  - **sudo ./install auto**
- Commands to check status of code deploy-agent:
  - **sudo systemctl status codedeploy-agent**
- If the status is no codedeploy-agent found then run following command:
  - **sudo systemctl start codedeploy-agent**

```
ubuntu@ip-172-31-15-147:~$ wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
--2024-02-07 07:25:38--  https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
Resolving aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com (aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com).
.. 3.5.208.129, 3.5.210.149, 3.5.211.111, ...
Connecting to aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com (aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.c
om)|3.5.208.129|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17892 (17K) []
Saving to: 'install'

install                     100%[===================================================>]  17.47K  --.-KB/s    in 0.001s

2024-02-07 07:25:38 (25.4 MB/s) - 'install' saved [17892/17892]

ubuntu@ip-172-31-15-147:~$
```

Fig. 5.1.2.1 Command to get S3 Bucket

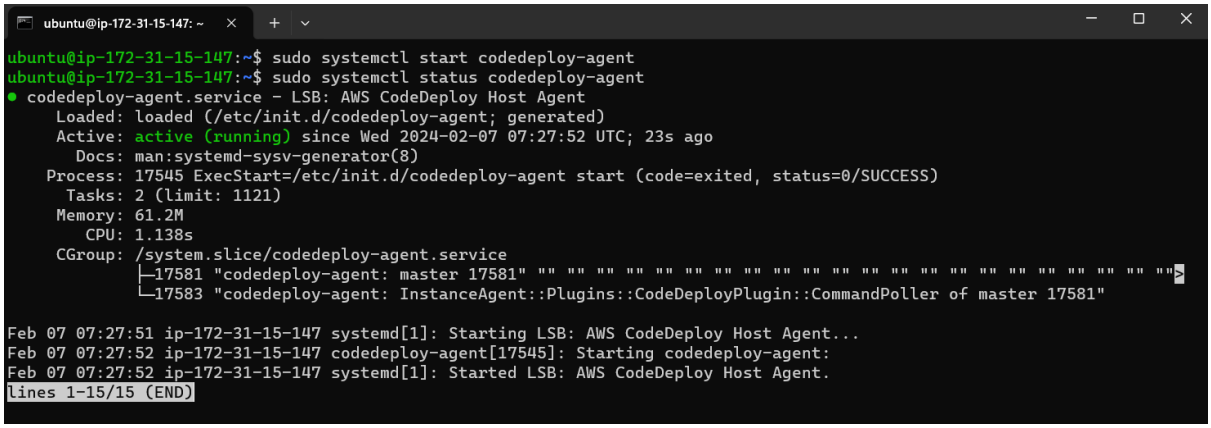


Fig. 5.1.2.2 Status of Code Deploy-agent

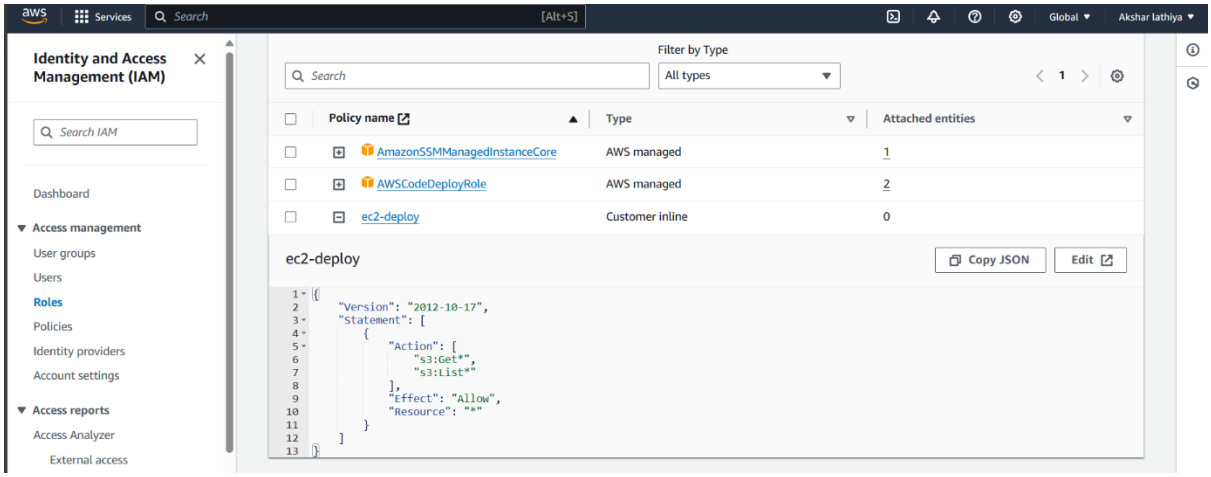


Fig. 5.1.2.3 IAM Role of EC2 Instance

After adding IAM role to EC2 instance , restart the code deploy-agent using **sudo systemctl restart codedeploy-agent.**

5.2 Continuous Feedback

Continuous feedback is an essential aspect of a CI/CD pipeline, providing valuable insights and information at every stage of the software development and deployment process. It involves collecting data, analysing metrics, and providing feedback to developers, testers, and stakeholders to improve the quality, efficiency, and reliability of the pipeline.

5.2.1 SNS (Simple Notification Service)

Amazon Simple Notification Service, is a fully managed messaging service provided by Amazon Web Services. It enables you to send notifications or messages to a large number of subscribers or endpoints, including email, SMS, HTTP/S, AWS Lambda, SQS, mobile devices, and more.

Steps to create SNS topics:

➢ Create an SNS Topic:

- Go to the AWS Management Console and navigate to the SNS service.
- Click on "Create topic".
- Enter a name and an optional display name for your topic.
- Click on "Create topic" to create the SNS topic.

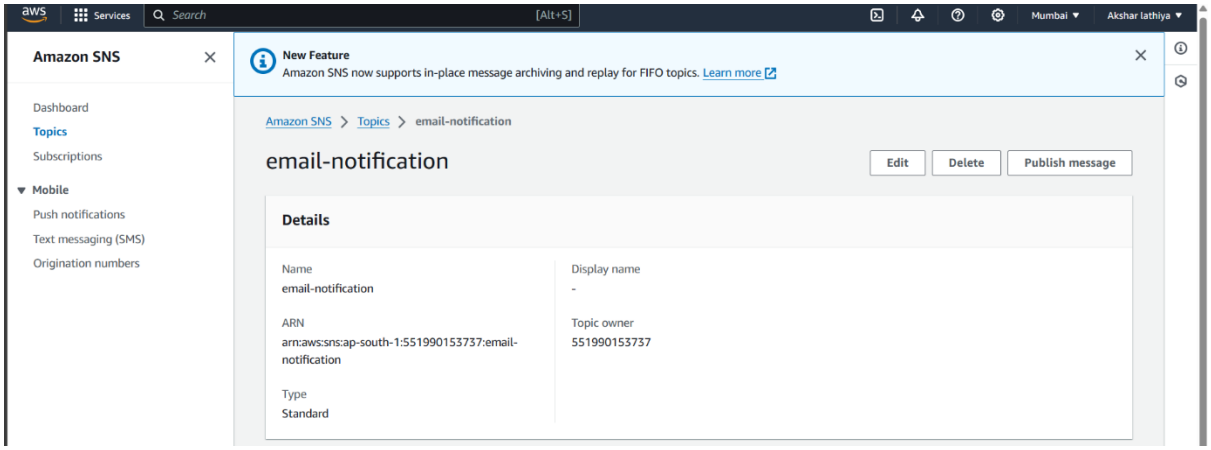


Fig. 5.2.1.1 SNS Topic

➢ Create Subscriptions:

- Once the topic is created, you can create subscriptions to specify where you want to send messages published to the topic.
- Click on the topic you created, then click on "Create subscription".
- Choose the protocol for the subscription (e.g., Email, SMS, HTTP, HTTPS, Lambda, SQS).
- Depending on the chosen protocol, provide the required endpoint information (e.g., email address, phone number, URL, Lambda function ARN, SQS queue ARN).
- Confirm the subscription request (e.g., by clicking on a confirmation link sent to an email address).

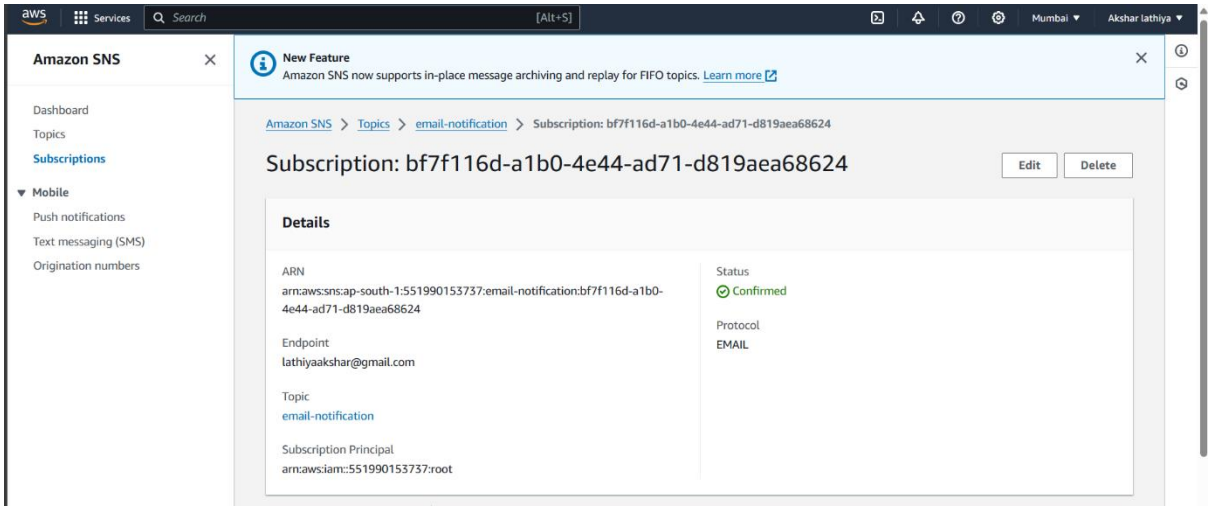


Fig. 5.2.1.2 Subscription of SNS Topic

- Publish Messages to the Topic:
  - After creating the topic and subscriptions, you can publish messages to the topic.
  - Click on the topic, then click on "Publish message".
  - Enter the message subject and message body.
  - Optionally, you can add message attributes to provide additional metadata.
  - Click on "Publish message" to send the message to all subscribed endpoints.
- Monitor and Manage SNS Topics:
  - Monitor the activity and performance of your SNS topics using CloudWatch metrics and alarms.
  - Use the SNS console to manage topics, subscriptions, access policies, and other configurations.

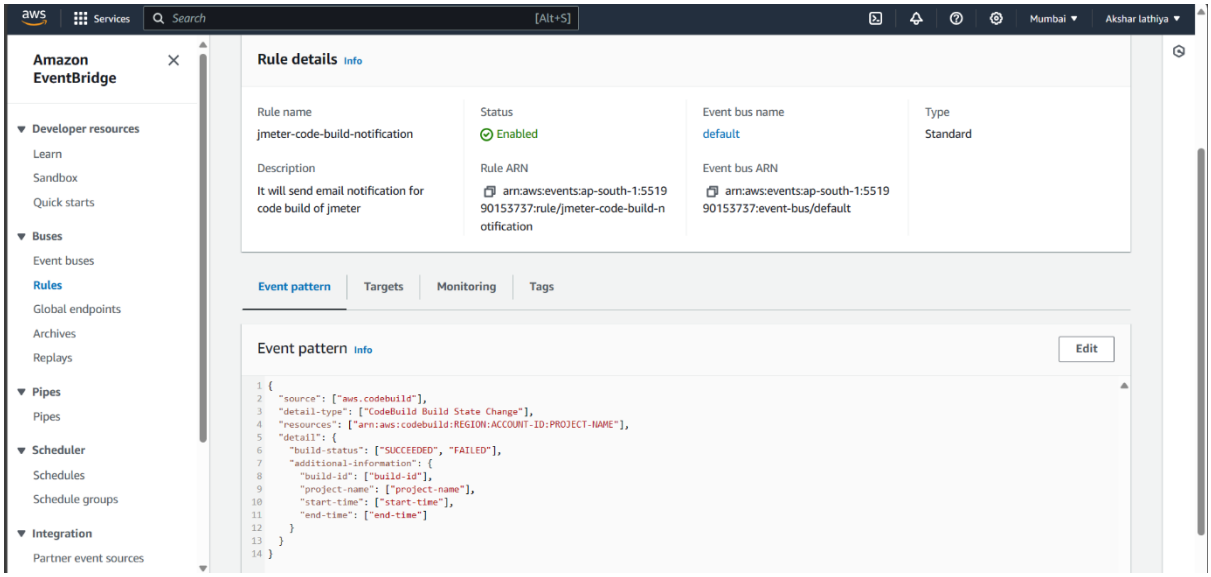


Fig. 5.2.1.3 Code Build Notification rule

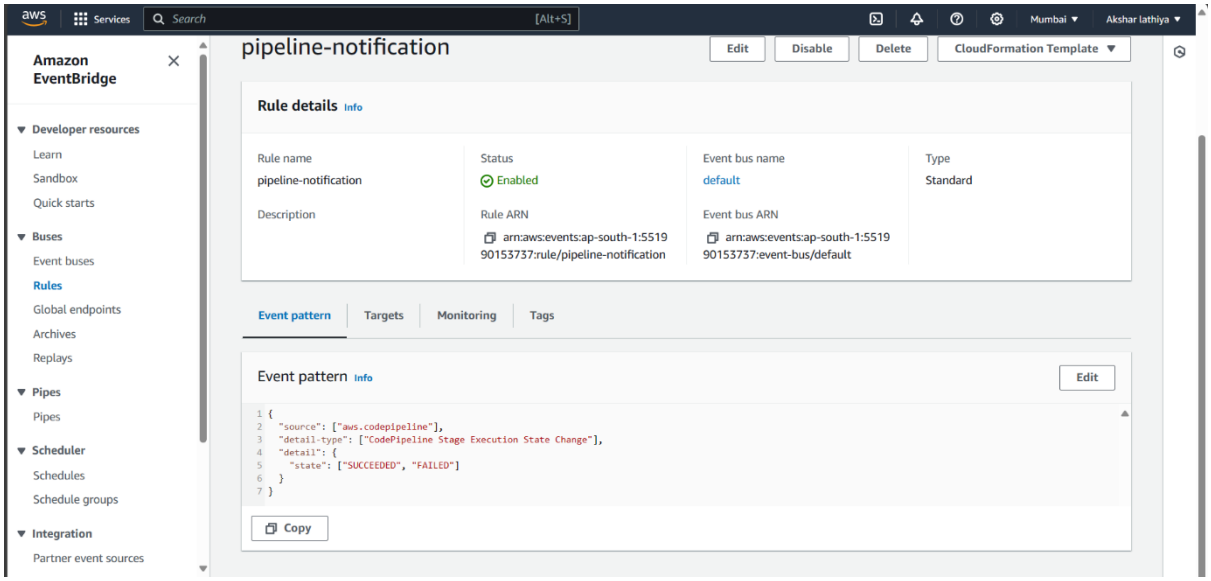


Fig. 5.2.1.4 Code Pipeline Notification Rule

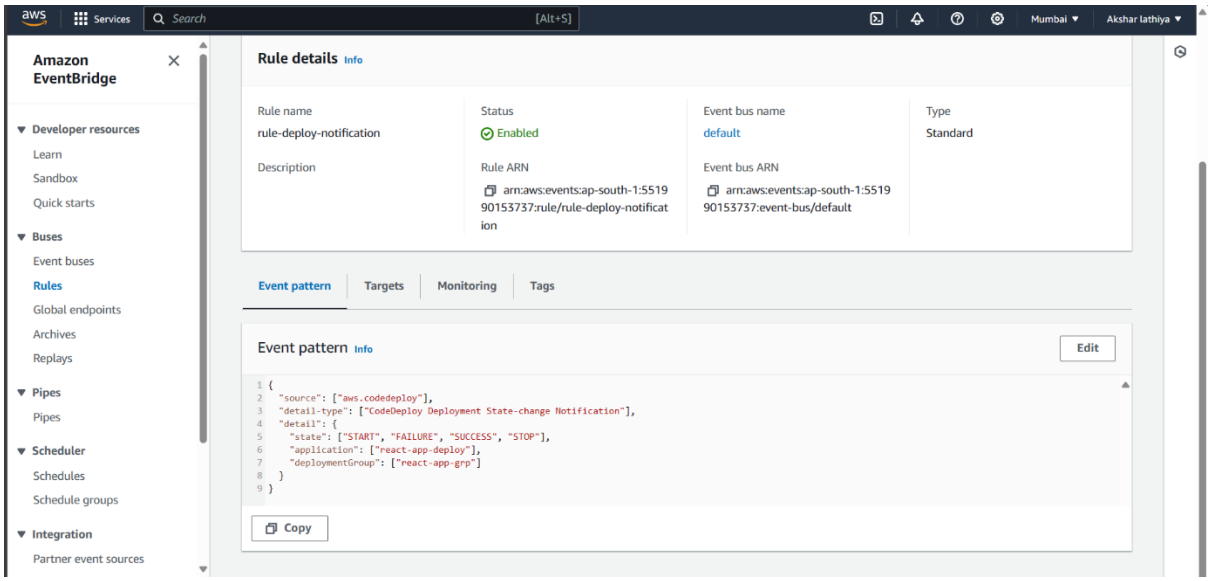


Fig. 5.2.1.5 Code Deploy Notification Rule

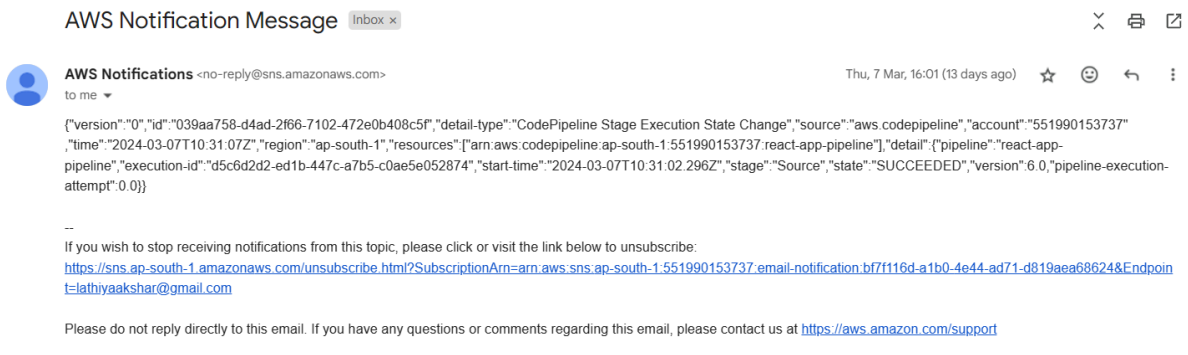


Fig. 5.2.1.6 Email Notification of Code Pipeline

5.2.2 Discord

Discord is a popular communication platform primarily used by group of people but has expanded to encompass various communities and groups beyond chatting. It provides a combination of text, voice, and video chat features, along with robust community-building tools.

Discord is used to send the notification of CPU usage alarm of Grafana, if the threshold value hits the maximum of its value then it will send the notification in discord. Grafana and discord are connected with webhook.

Steps to create webhook:

- Open Discord and navigate to the channel where you want to receive notifications.
- Click on the gear icon next to the channel name and select "Integrations".
- Click on "Create Webhook", then provide a name for your webhook and optionally customize the avatar.
- Copy the webhook URL generated by Discord. This URL will be used to send messages to the channel.

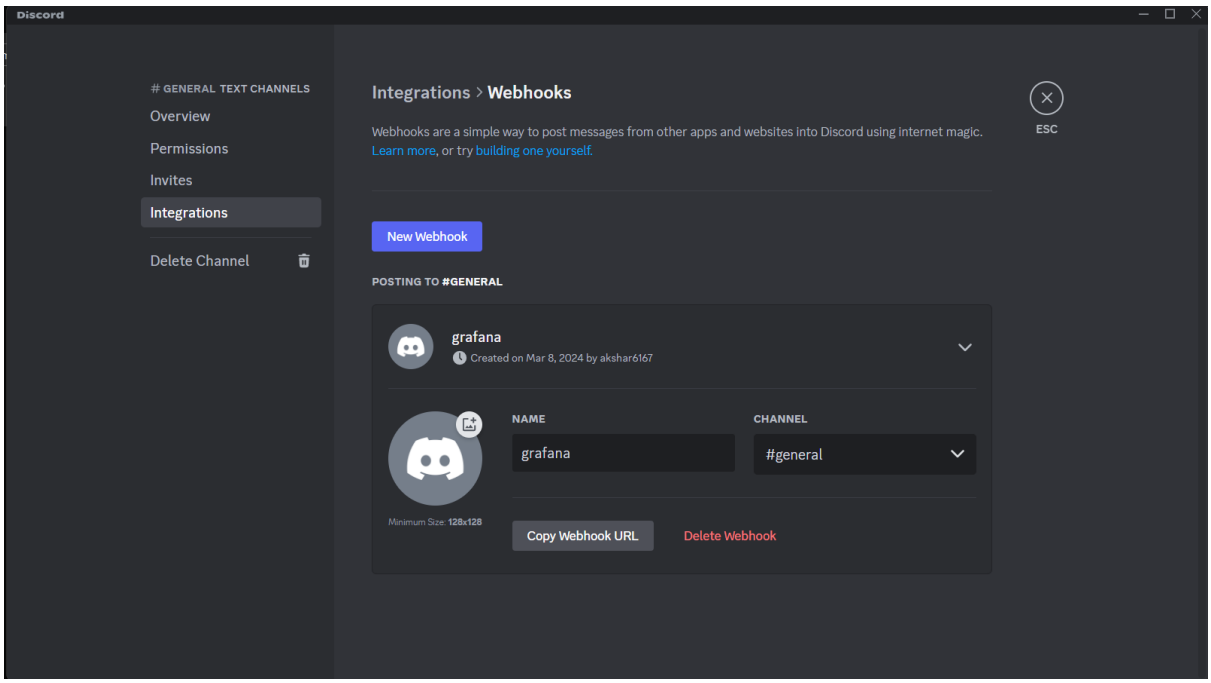


Fig. 5.2.2.1 Discord webhook

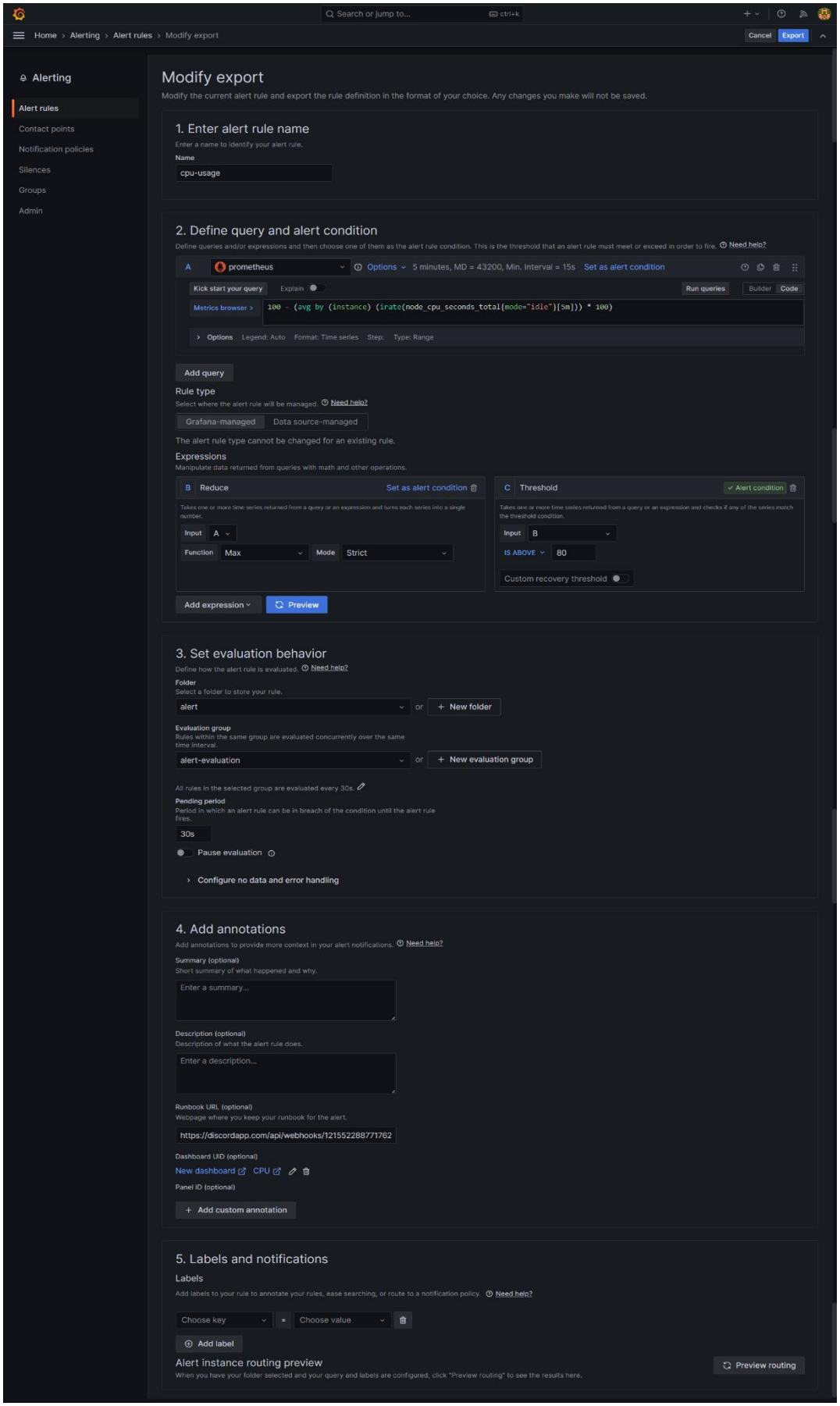


Fig. 5.2.2.2 Grafana webhook configure

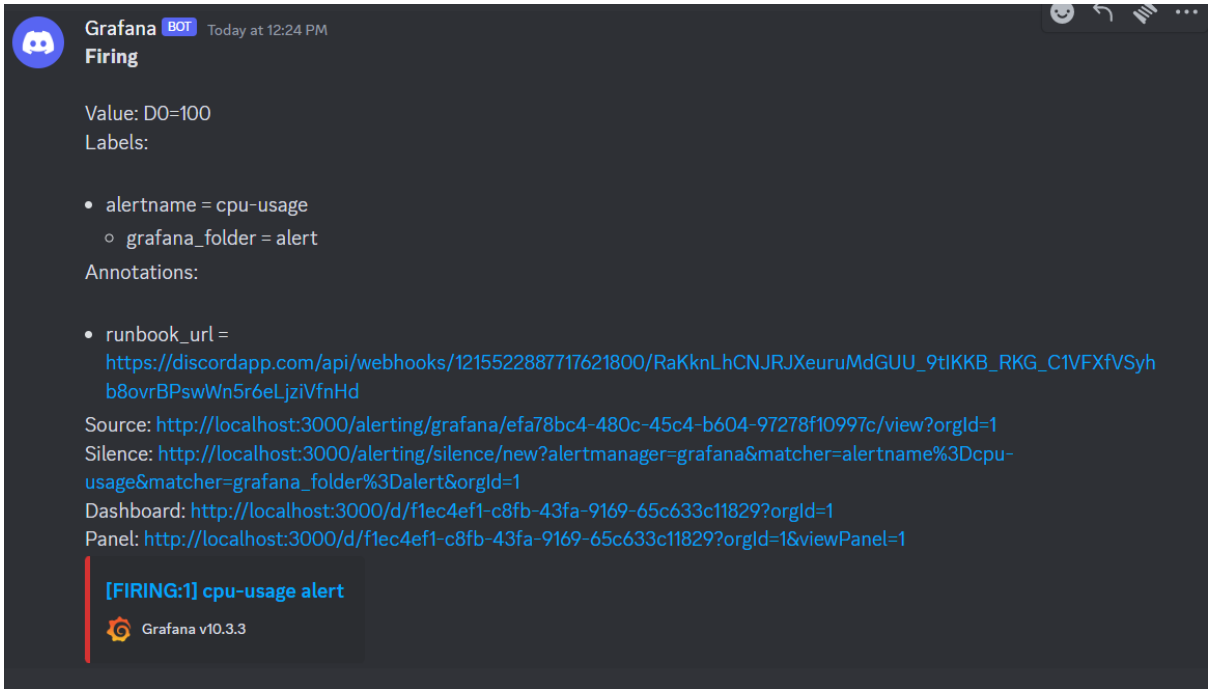


Fig. 5.2.2.3 CPU Usage alert message

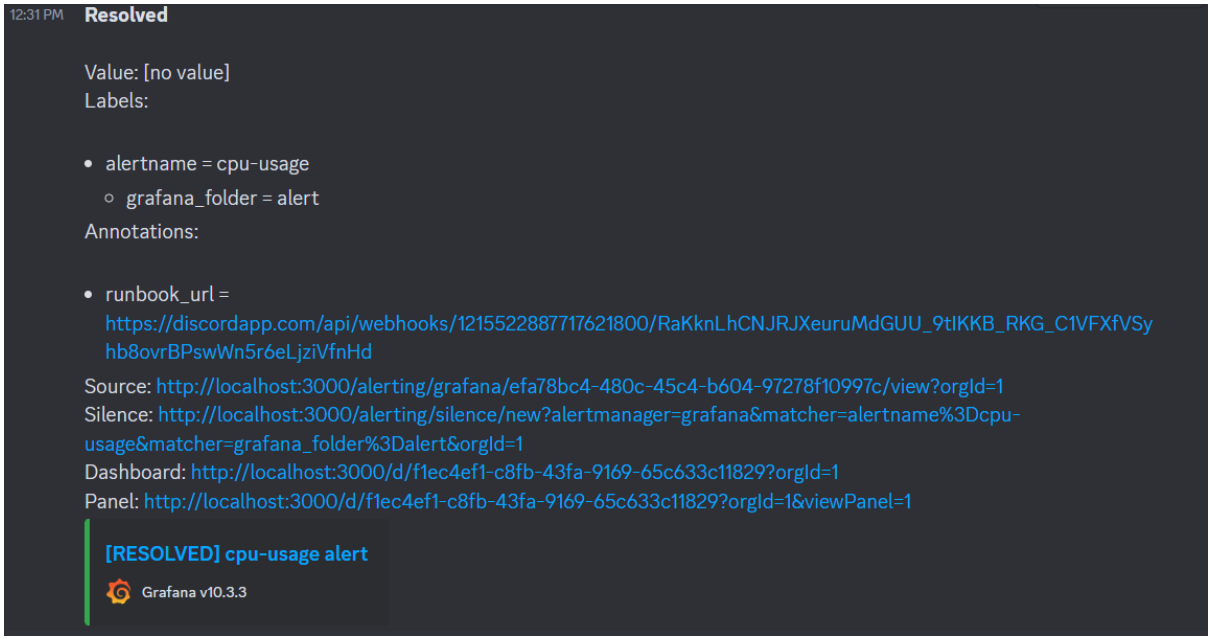


Fig. 5.2.2.4 CPU Usage alert resolved message

5.3 Continuous Monitoring

Monitoring in a pipeline context typically refers to the process of observing and tracking various aspects of data, processes, and performance as they flow through a system. In the context of software development or data processing, a pipeline is a series of connected stages or steps where data is processed sequentially. Monitoring these pipelines is crucial for ensuring their efficiency, reliability, and effectiveness.

5.3.1 Grafana

Grafana is an open-source analytics and monitoring platform primarily focused on visualizing time-series data. It allows users to create, explore, and share dashboards containing graphs, charts, and other visualizations of their data. Grafana is highly flexible and supports a wide range of data sources, making it popular for monitoring and observability use cases in various domains, including IT infrastructure, application performance monitoring, IoT, and more.

Steps to configure Grafana in EC2 instance:

- **Launch an EC2 Instance:**
  - Sign in to the AWS Management Console.
  - Navigate to the EC2 dashboard.
  - Launch a new EC2 instance, choosing an appropriate Amazon Machine Image (AMI) (e.g., Amazon Linux, Ubuntu, etc.).
  - Configure security groups to allow inbound traffic on the Grafana port (default is TCP port 3000).
- **Connect to the EC2 Instance:**
  - Once the instance is launched, connect to it via SSH using a tool like PuTTY (for Windows) or SSH (for Linux/Mac).
  - Use the key pair associated with the EC2 instance for authentication.
- **Install Grafana:**
  - sudo apt install grafana
- **Start Grafana Service:**
  - sudo systemctl start grafana-server
  - sudo systemctl enable grafana-server
- **Access Grafana Web Interface:**
  - Open a web browser and navigate to the public IP address or DNS name of your EC2 instance, appending :3000 to the address (Grafana's default port).
  - You should see the Grafana login page.
- **Log in and Configure Grafana:**
  - Log in using the default credentials (username: admin, password: admin).
  - Follow the prompts to change the default password.
  - Optionally, configure Grafana data sources and dashboards based on your requirements. You can add data sources like Prometheus, InfluxDB, etc., from the Grafana web interface.

- **Monitor and Maintain:**
  - Monitor the EC2 instance and Grafana service to ensure they are running smoothly.
  - Regularly update Grafana and the underlying operating system to patch security vulnerabilities and access new features.

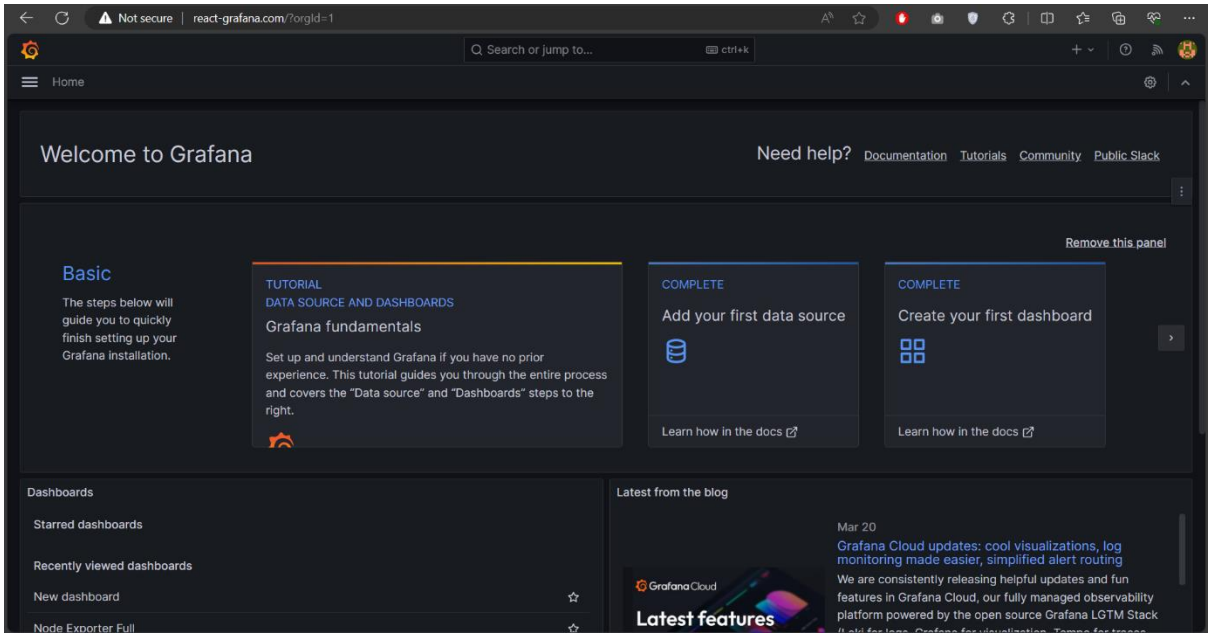


Fig. 5.3.1 Grafana on 3000 Port of Instance

5.3.1.1 Prometheus

Prometheus is an open-source monitoring and alerting toolkit widely utilized in modern IT environments, particularly within cloud-native and containerized applications. Developed originally by SoundCloud, Prometheus is renowned for its robust capabilities in collecting, storing, querying, and visualizing metrics. Its flexible data model employs labels, enabling multidimensional analysis crucial for efficient querying and insightful analysis. Using a pull-based approach, Prometheus scrapes metrics from HTTP endpoints at regular intervals, ensuring flexibility and adaptability to dynamic environments.

Central to its functionality is PromQL, a powerful query language enabling users to manipulate and retrieve metrics with precision. With its integrated time-series database optimized for high write throughput and efficient querying, Prometheus excels in historical trend analysis and anomaly detection. Its support for various service discovery mechanisms, including Kubernetes integration and cloud service providers, simplifies monitoring setup and maintenance. Additionally, Prometheus offers built-in alerting capabilities and seamless integration with Alertmanager for effective alert routing and management. Its scalability, federation support, and extensive integration ecosystem make Prometheus a cornerstone in the toolkit of DevOps

teams and system administrators, facilitating reliable and comprehensive monitoring solutions across diverse infrastructure landscapes.

Prometheus is connected with grafana and it is also run on port 9090 of instance. Grafana will show the graphs on the basis of data collected by Prometheus.

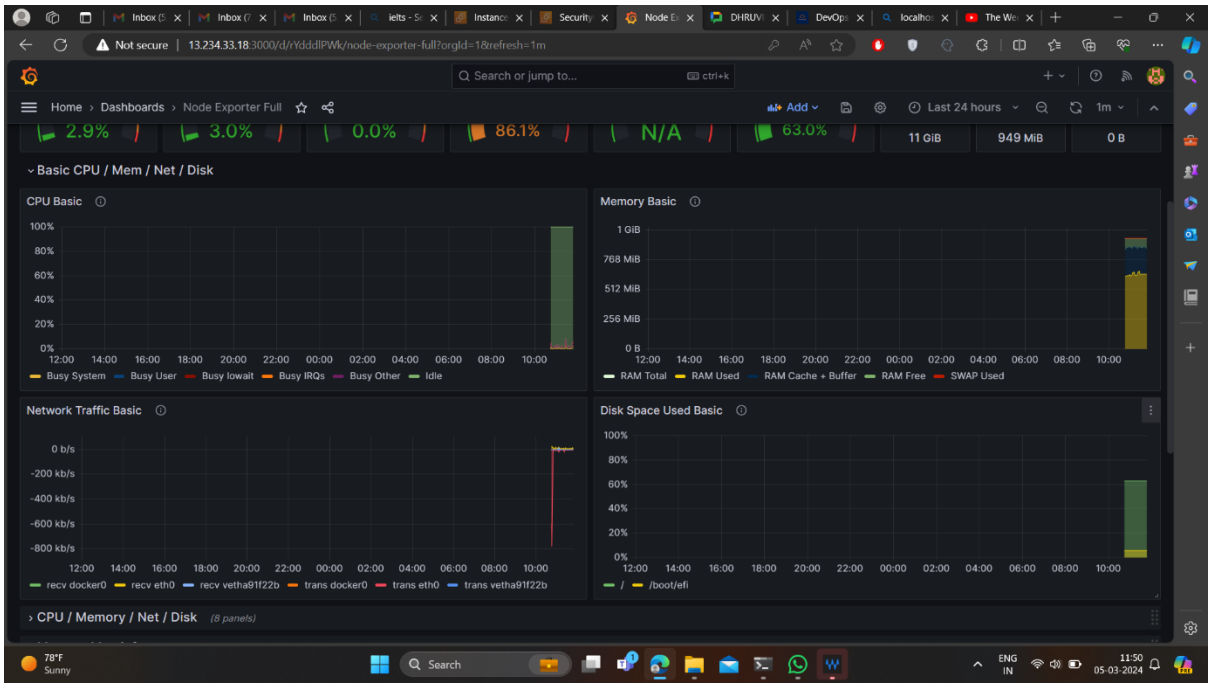


Fig. 5.3.1.1.1 Grafana Dashboard

Fig. 5.3.1.1.2 Grafana Dashboard

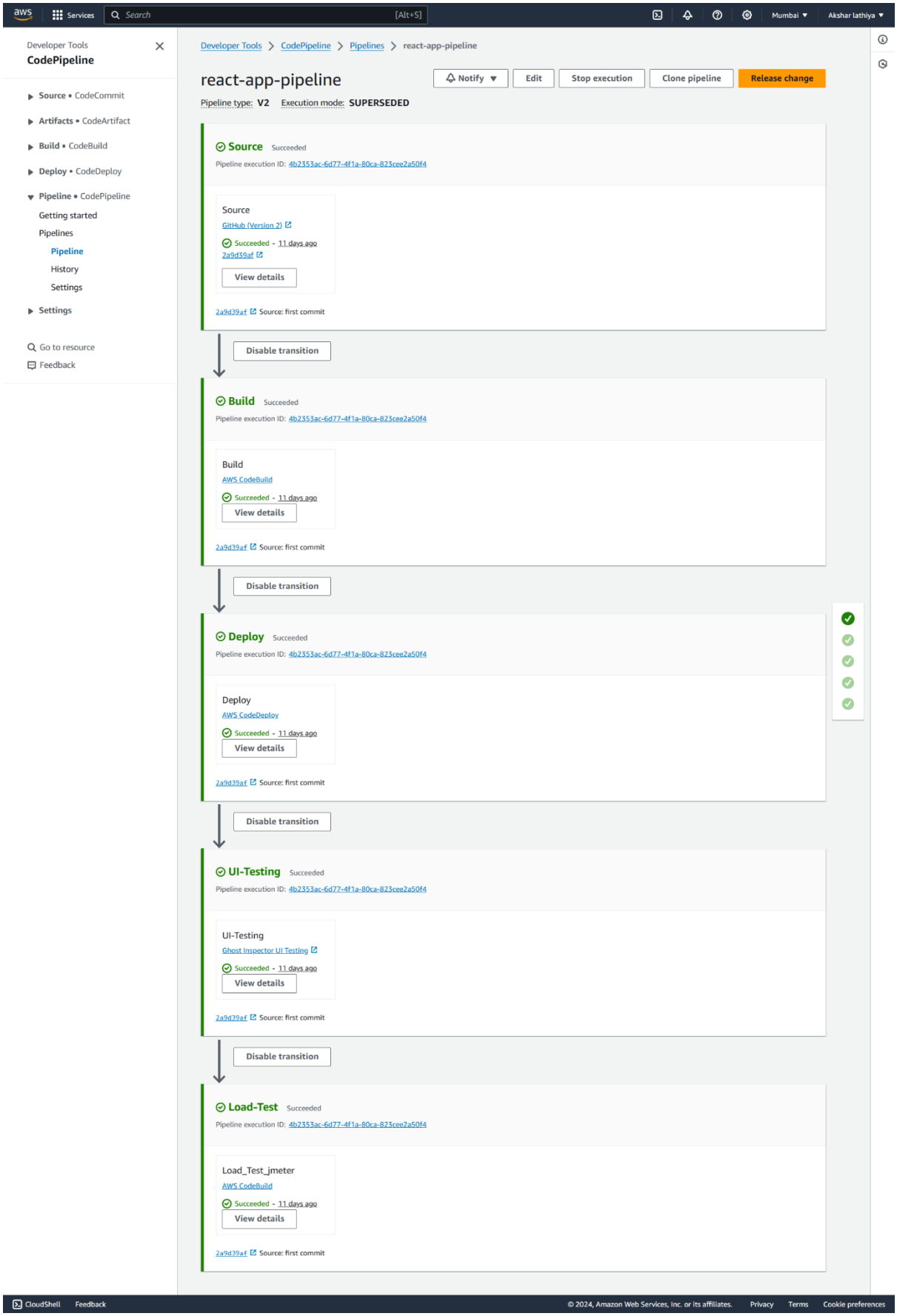


Fig 5.1 CI CD Pipeline

6. CONCLUSION

6.1 CONCLUSION

In conclusion, the CI/CD pipeline project has been a resounding success, revolutionizing our software development and deployment processes. Through the implementation of robust CI/CD pipelines, we have achieved significant improvements in efficiency, reliability, and collaboration within our development teams.

The adoption of CI/CD practices has enabled us to automate key aspects of our software delivery lifecycle, from code integration to deployment, leading to faster feedback cycles and quicker time-to-market for our applications. By automating repetitive tasks such as testing and deployment, we have freed up valuable time for our developers to focus on innovation and delivering value to our customers.

Furthermore, the introduction of CI/CD pipelines has enhanced the overall quality of our software products. Continuous integration ensures that code changes are integrated and tested frequently, reducing the likelihood of integration issues and catching bugs early in the development process. Meanwhile, continuous deployment enables us to deploy code changes to production environments with confidence, knowing that they have been thoroughly tested and validated.

Looking ahead, we recognize that the journey towards CI/CD maturity is ongoing. We must continue to refine and optimize our pipelines, incorporating feedback and lessons learned to drive continuous improvement. Additionally, we must remain adaptable and open to emerging technologies and practices that can further enhance our software delivery capabilities.

In summary, the CI/CD pipeline project has not only transformed our software development practices but also laid the foundation for a culture of continuous improvement and innovation within our organization. By embracing CI/CD principles and practices, we are better positioned to meet the evolving needs of our customers and stay ahead in today's competitive marketplace.

7. REFERENCES

AWS Documentation: https://docs.aws.amazon.com/

YouTube Channel (practical): https://www.youtube.com/@AbhishekVeeramalla

Medium: Building a CI/CD pipeline for a ReactJS app using AWS CodePipeline | by Marcrinemm | Medium

YouTube Channel: https://www.youtube.com/@skillcurb

YouTube Channel (theory): https://www.youtube.com/@TechnicalGuftgu

Stack Overflow (error): https://stackoverflow.com/questions/63667602/unable-to-execute-aws-pipeline-error-an-error-occurred-accessdenied-when-cal

GeeksforGeeks: https://www.geeksforgeeks.org/how-to-build-a-ci-cd-pipeline-with-aws/