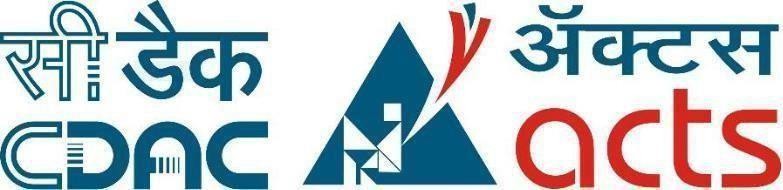
## Project Report On

**Simulating centralized identity and access management in a distributed network with high server availability.**



Submitted in partial fulfillment for the award of **Post Graduate Diploma in IT Infrastructure System & Security (PG-DITISS)** from**C-DAC ACTS(Pune)**

###### Guided By:

Ms.TejaswiniApate

###### Presented by:

|  |  |  |
| --- | --- | --- |
| Abhishek Raj | PRN: | 230940123002 |
| Sahil Sunil Tadas | PRN: | 230940123036 |
| Sayyad Nayda Anwar  Madhavi Mallesh Jangam | PRN:  PRN: | 230940123043  230940123059 |

Page|1

# CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that:

|  |
| --- |
| Abhishek Raj |
| Sahil Sunil Tadas |
| Sayyad Nayda Anwar  Madhavi Mallesh Jangam |

Have successfully completed their project on

**“ Simulating centralizedidentity and access management in a distributed network with high server availability ”**

Under the guidance of

Ms.Tejaswini Apate

Project Guide Project Supervisor

# ACKNOWLEDGEMENT

This project “**Simulating centralized identity and access management in a distributed network with high server availability**” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We allare very glad to mention the name of **Ms.TejaswiniApate** for her valuable guidance to work on this project. Her guidance and support helped us to overcome various obstacles and intricacies during project work.

Our most heartfelt thanks go to Ms.Swati Salunkhe (Course Coordinator, **PG- DITISS**) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| From: | |  | | |
| Abhishek Raj (230940123002) | |  |  | |
| Sahil Sunil Tadas (230940123036) | | : |  | |
| Sayyad Nayda Anwar (230940123043)  Madhavi Mallesh Jangam (230940123059) | | : |  | |

# ABSTRACT

This project focuses on Simulating centralized identity and access management in a distributed network with high server availability. Integrating LDAP protocol, Jenkins Pipeline, Docker,Amazon elastic compute cloud,Amazon Elastic Container Service provide a comprehensive solution for centralized identity management,automating the deployment of containerized applications in AWS. Jenkins can be used to define CI/CD pipelines that build Docker images, run tests, and deploy applications to ECS. Docker ensures consistency between development and production environments by packaging applications into containers, while ECS manages the underlying infrastructure and orchestrates the deployment of containers across a cluster of EC2 instances . This combination enables teams to deliver software faster, with greater reliability and scalability, in the AWS cloud.

.

# TABLE OF CONTENTS

1. Introduction 6
2. Tools and Operating system used 9
3. Flowchart 20

4. Prerequisites 21

1. Workflow 22
2. Implementation 23
3. Output 29

8. References 31

**INTRODUCTION**



Simulating centralized(means bringing things to one central place or under unified control) identity and access management in a distributed network with high server availability refers to creating a system where user identities and their access permissions are managed centrally, even though the network spans multiple locations or servers. This system ensures that users can securely access resources regardless of their physical location and that the network remains operational even if some servers experience downtime.

Problem statement:- In today's distributed computing environments, maintaining centralized control over identity and access management while ensuring high server availability poses a significant challenge. As organizations expand their network infrastructure across multiple locations and employ a diverse array of servers, ensuring consistent access controls and minimizing downtime becomes increasingly complex. The task at hand is to develop a solution that can effectively simulate centralized identity and access management in such a distributed network environment, all while guaranteeing high server availability.

Key Objectives:

1. Centralized Identity Management: Develop a system that centrally manages user identities and access permissions across the distributed network, ensuring consistency and security.

2. Access Control: Implement robust access control mechanisms to regulate user access to resources, regardless of their physical location within the network.

3. High Server Availability: Design the system with redundancy and failover capabilities to maintain network operation even in the event of server failures or maintenance activities.

4. Scalability: Ensure that the solution can scale seamlessly to accommodate the growing demands of the network, supporting an increasing number of users and resources.

5. Performance: Optimize the system's performance to provide fast and reliable access to resources, minimizing latency and ensuring a seamless user experience.

Objective:- Integrating LDAP (Lightweight Directory Access Protocol), Jenkins, Docker, Amazon Elastic Compute Cloud (EC2), and Amazon Elastic Container Service (ECS) collectively supports a range of goals related to modern software development, deployment, and management in the cloud. Here are several key objectives they help achieve:

1. **Centralized Identity and Access Management**: LDAP serves as a centralized directory service for managing user identities and access permissions. Integrating LDAP with Jenkins, Docker, ECS, and EC2 ensures consistent authentication and access control across all stages of the development and deployment lifecycle. This centralization simplifies user management and enhances security.

2**. Automated CI/CD Pipelines**: Jenkins orchestrates automated CI/CD pipelines for building, testing, and deploying applications. By integrating with LDAP, Jenkins can authenticate users and enforce access controls on pipeline execution. Docker facilitates consistent packaging and deployment of applications, while ECS manages the orchestration of containerized workloads, enabling seamless deployment of applications across environments.

3. **Containerization and Microservices Architecture**: Docker enables containerization of applications, promoting consistency and portability. ECS provides a managed environment for deploying and scaling containerized applications, ensuring high availability and resource efficiency. Leveraging Docker and ECS together allows for the adoption of microservices architecture, enhancing scalability and agility.

4. **Scalable Infrastructure Management:** EC2 offers resizable compute capacity in the cloud, while ECS simplifies the management of containerized workloads. Together, they provide scalable infrastructure resources for hosting applications. With auto-scaling capabilities, EC2

and ECS dynamically adjust resource allocation based on workload demands, ensuring optimal performance and cost efficiency.

5. **High Availability and Fault Tolerance**: ECS and EC2 support high availability and fault tolerance by distributing workloads across multiple availability zones and automatically replacing unhealthy instances. Docker's container orchestration features, combined with ECS's service auto-recovery mechanisms, enhance resilience and minimize downtime during failures.

6**. Efficient Resource Utilization**: Integrating Docker with ECS optimizes resource utilization by allowing multiple containers to run on a single EC2 instance. This maximizes resource efficiency and reduces operational costs associated with managing infrastructure resources.

7. **Standardized Development and Deployment Processes**: Using LDAP, Jenkins, Docker, ECS, and EC2 together promotes standardized development and deployment processes. Developers can leverage automated CI/CD pipelines to build, test, and deploy applications consistently across different environments, leading to improved efficiency and reliability.

**Tools and Operating Systems used:**

**JENKINS**

****

Jenkins is an open-source automation server widely used in software development. It automates tasks like building, testing, and deploying code, promoting CI/CD practices. With its rich plugin ecosystem, Jenkins integrates seamlessly with various tools and platforms.

It offers flexible pipeline scripting for custom workflows. Jenkins enables developers to catch integration bugs early, speeding up development cycles and improving code quality.

It supports distributed builds, can trigger actions based on code changes, and provides a dashboard for monitoring builds. Jenkins' extensibility and community support make it a popular choice for automating software delivery pipelines, fostering collaboration, and enhancing DevOps practices.

Key features and concepts of Jenkins include Continuous Integration (CI), Continuous Deployment (CD) and Continuous Delivery, Plugins, Jobs and Pipelines. Automated Testing. Scheduled Builds, Version Control Integration ,etc.

Key points about Jenkins:

1. Open-source automation server for building, testing, and deploying software.

2. Supports Continuous Integration and Continuous Deployment (CI/CD) practices.

3. Integrates with diverse tools and services through its plugin system.

4. Offers flexible pipeline scripting for defining complex workflows.

5. Aids in catching integration bugs early and improving code quality.

6. Supports distributed builds for scalability.

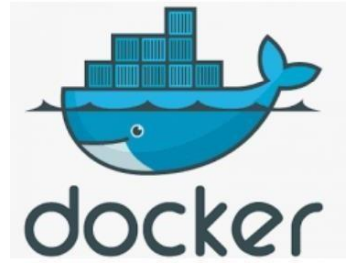
7. Triggers actions based on code changes.

8. Provides a web-based dashboard for tracking build statuses.

9. Enhances collaboration between development and operations teams.

10.Widely used in DevOps to automate software delivery pipelines.

**Docker:**

****

Docker is an open-source platform that automates the deployment, packaging, and management of applications within lightweight, isolated software containers. Containers provide a consistent and portable environment for running applications, ensuring that they run the same way across different environments, from development to production. While containers have been used in Linux and Unix systems for some time, Docker, an open source project launched in 2013, helped popularize the technology by making it easier than ever for developers to package their software to “build once and run anywhere. Founded as DotCloud in 2008 by Solomon Hykes in Paris, what we now know as Docker started out as a platform as a service (PaaS) before pivoting in 2013 to focus on democratizing the underlying software containers its platform was running on.

Key concepts and features of Docker include:

1. Containerization: Docker allows you to package an application and its dependencies into a container image. This image includes everything needed to run the application, such as code, runtime, libraries, and system tools. Containers ensure consistency and eliminate "it works on my machine" issues.

2. Docker Images: Docker images are the templates that define the contents and configurations of containers. Images can be versioned and shared, making it easy to distribute applications and replicate environments.

3. Docker Containers: Containers are instances of Docker images. Each container is isolated from the host system and other containers, providing a secure and consistent runtime environment. Containers are lightweight and start up quickly.

4. Docker Hub: Docker Hub is a cloud-based repository where you can find and share Docker images. It's a central hub for the Docker community to share and collaborate on container images.

5. Dockerfile: A Dockerfile is a text file that contains instructions for building a Docker image. It defines the base image, application code, dependencies, environment variables, and other configurations.

6. Docker Compose: Docker Compose is a tool for defining and running multi-container applications using a simple YAML file. It's useful for setting up complex applications that consist of multiple interconnected services.

7. Container Orchestration: Docker supports container orchestration platforms like Kubernetes and Docker Swarm. These platforms help manage and scale containerized applications across clusters of machines.

8. Microservices: Docker is commonly used in microservices architectures, where applications are broken down into smaller, independent services that can be developed, deployed, and scaled separately.

9. Isolation and Security: Containers offer process-level isolation, meaning each container has its own filesystem, network, and process space. This enhances security and avoids conflicts between applications.

10. Resource Efficiency: Docker containers share the host operating system's kernel, which results in better resource utilization compared to traditional virtualization.

11. Cross-Platform: Docker containers can run on various operating systems, including Linux, Windows, and macOS.

12. Development and Testing: Docker provides a consistent environment for development and testing, ensuring that the application behaves the same way in various stages of the software development lifecycle.

13. Portability: Docker containers can be easily moved between environments, making it easier to switch between development, testing, and production systems.

Docker has revolutionized how applications are developed, deployed, and managed by simplifying the process of packaging applications and their dependencies. It has become a cornerstone technology in modern application development and deployment workflows.

GITHUB:

****

GitHub is a web-based platform for version control and collaborative software development. It provides a space for developers to store, manage, and collaborate on code repositories. Using the Git version control system, GitHub allows multiple contributors to work on the same codebase simultaneously. Developers can clone repositories, make changes, and propose these changes to the main codebase through pull requests. It offers tools for issue tracking, project management, and documentation, streamlining the development lifecycle.

GitHub fosters collaboration through features like code reviews and discussions. It's widely used in open-source and private projects, enabling teams to work together on code, share knowledge, and contribute to a global developer community.

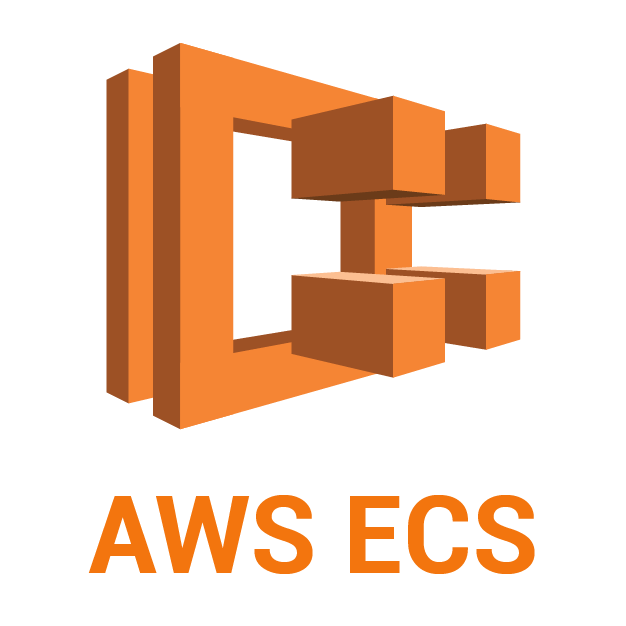
**EC2:**



EC2 stands for Elastic Compute Cloud, which is a web service provided by Amazon Web Services (AWS) that allows users to rent virtual computers (known as instances) on which they can run their own applications.

EC2 instances provide scalable computing capacity in the cloud and are commonly used for a wide range of purposes including hosting websites, running applications, processing data, and more. Users can choose from various instance types with different combinations of CPU, memory, storage, and networking capacity to meet their specific requirements. EC2 offers flexibility, scalability, and cost-effectiveness for businesses and developersneeding computing resources in thecloud.

**ECS:**



ECS stands for Amazon Elastic Container Service. It is a fully managed container orchestration service provided by Amazon Web Services (AWS). ECS allows users to easily run, manage, and scale containerized applications using Docker containers and the familiar tools of the Docker ecosystem.

With ECS, users can define their applications as collections of Docker containers that run across a cluster of Amazon EC2 instances. ECS takes care of scheduling containers across the cluster, managing the underlying infrastructure, and monitoring the health of containers.

ECS supports various deployment options, including running tasks as standalone containers or as part of a service that ensures a specified number of tasks are always running and automatically handles scaling, load balancing, and rolling updates.

ECS also integrates with other AWS services such as Elastic Load Balancing, Amazon VPC, AWS IAM, AWS CloudWatch, and AWS CloudFormation, providing a comprehensive platform for deploying and managing containerized applications in the cloud.

Top

**LDAP:**

LDAP stands for Lightweight Directory Access Protocol. It is an open, vendor-neutral protocol used for accessing and maintaining directory information services over an IP network. LDAP is often used for centralized authentication, authorization, and directory services in networked environments. LDAP directories typically store information such as user identities, group memberships, access control policies, and other attributes in a hierarchical manner. These directories can be accessed and queried by LDAP clients to retrieve or modify information.

LDAP is commonly used in various applications and services for tasks such as user authentication (e.g., logging into a system or application using LDAP credentials), storing and retrieving address book or contact information, and centralized management of user accounts across multiple systems or services.Popular LDAP directory servers include Open LDAP, Microsoft Active Directory, and Apache Directory Server. LDAP is widely used in enterprise environments for managing user accounts, groups, and other directory-related information in a centralized and standardized manner.

Advantages:

1. Centralized Directory Service: LDAP provides a centralized repository for storing and managing directory information, such as user accounts, groups, access control policies, and other attributes. This centralization simplifies administration and ensures consistency across multiple systems and services.

2. Scalability: LDAP directories can scale to accommodate large numbers of entries, making them suitable for enterprise environments with thousands or even millions of users. LDAP servers can handle high volumes of queries efficiently, ensuring fast response times even in large-scale deployments.

Disadvantages:

1. Complexity: Setting up and configuring an LDAP directory can be complex, especially

for organizations without dedicated IT expertise in directory services. LDAP requires careful planning and design to ensure proper schema definition, directory structure, and access control policies.

1. Resource Intensive: LDAP directories can consume significant computing resources, especially in large-scale deployments with high query volumes. Administrators need to carefully manage and optimize LDAP server resources to ensure optimal performance and scalability.

**SSO:**

SSO stands for Single Sign-On. It is an authentication process that allows a user to access multiple applications or services with a single set of login credentials (such as username and password). Instead of requiring users to log in separately to each application or service, SSO enables users to authenticate once and then access all authorized resources without being prompted to log in against works by using a trusted identity provider (IdP) to authenticate users and generate security tokens that can be used to access various applications or services within a specific domain or ecosystem. Once a user logs in to one application or service using SSO, subsequent requests to access other applications or services within the same SSO domain are automatically authenticated using the security token, eliminating the need for additional logins.

SSO improves user experience by reducing the number of times users need to enter their credentials, simplifying the authentication process, and enhancing security by centralizing authentication and access control. It is widely used in enterprise environments and web applications to streamline access to multiple resources while maintaining security and user privacy. Popular SSO protocols and standards include SAML (Security Assertion Markup Language), OAuth, and OpenID Connect.

Advantages:

1. Improved User Experience: SSO simplifies the login process for users by allowing them to access multiple applications or services with a single set of credentials. This reduces the need for users to remember and manage multiple passwords, leading to a more seamless and convenient user experience.

2. Increased Productivity: With SSO, users can quickly access various applications and services without the need to repeatedly log in, which saves time and improves productivity. Users can focus on their tasks without interruptions caused by frequent authentication prompts.

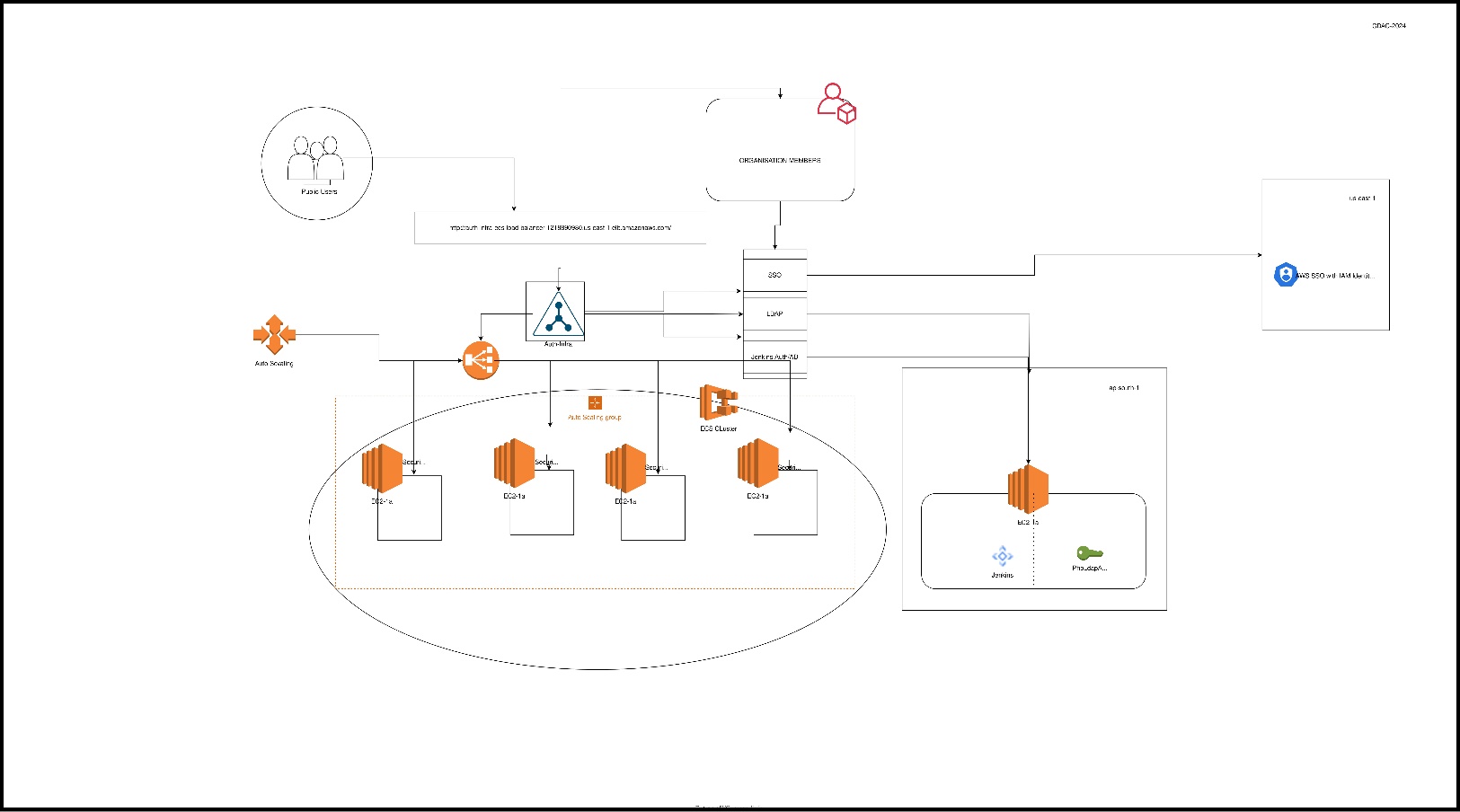
Disadvantages:

1. Single Point of Failure: SSO introduces a single point of failure, as authentication relies on a centralized identity provider (IdP). If the IdP experiences downtime or issues, users may be unable to access any connected applications or services, leading to disruptions in productivity.

2. Dependency on External Providers: Organizations that rely on third-party IdPs for SSO may face dependency issues and potential service disruptions if the IdP experiences outages or service degradation. Organizations need to carefully evaluate the reliability and availability of chosen IdPs to mitigate risks.

Top of Form

**FLOWCHART**

****

**PREREQUISITES**

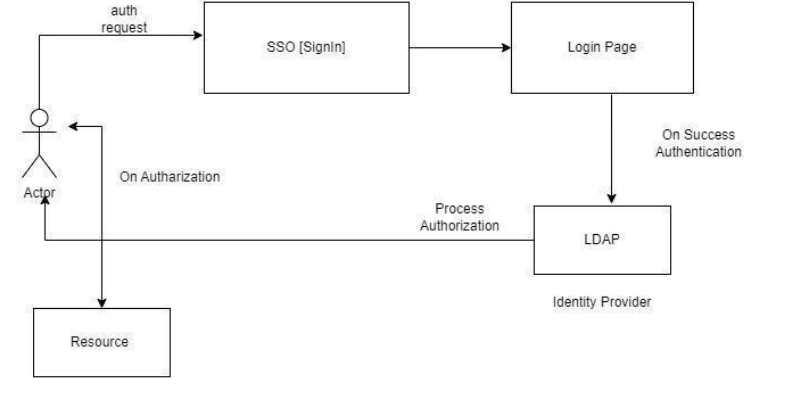
**System Configurations-**

* Hardware Requirements –
* Minimum 4 GB RAM
* Core- i5, AMD Ryzen 7 Processor
* 40 GB Free Space
* Software Requirements-
* Jenkins
* Docker
* Github
* Platform:-
* Amazon Web Services (AWS)
* Services: -

AWS:-

* EC2 Instances:- To host an application that will act as a service provider.
* ECS:- Plan to containerize and orchestrate the application.
* VPC:- To get a private cloud where we can securely manage all of our infrastructural resources.
* Auto Scaling:- The ability of a system to readily add extra processing resources to handle the handled loads.
* Load Balancing:- It involves distributing incoming network traffic across multiple servers.
* LDAP Server:- Act as an identity provider.

**WORKFLOW**

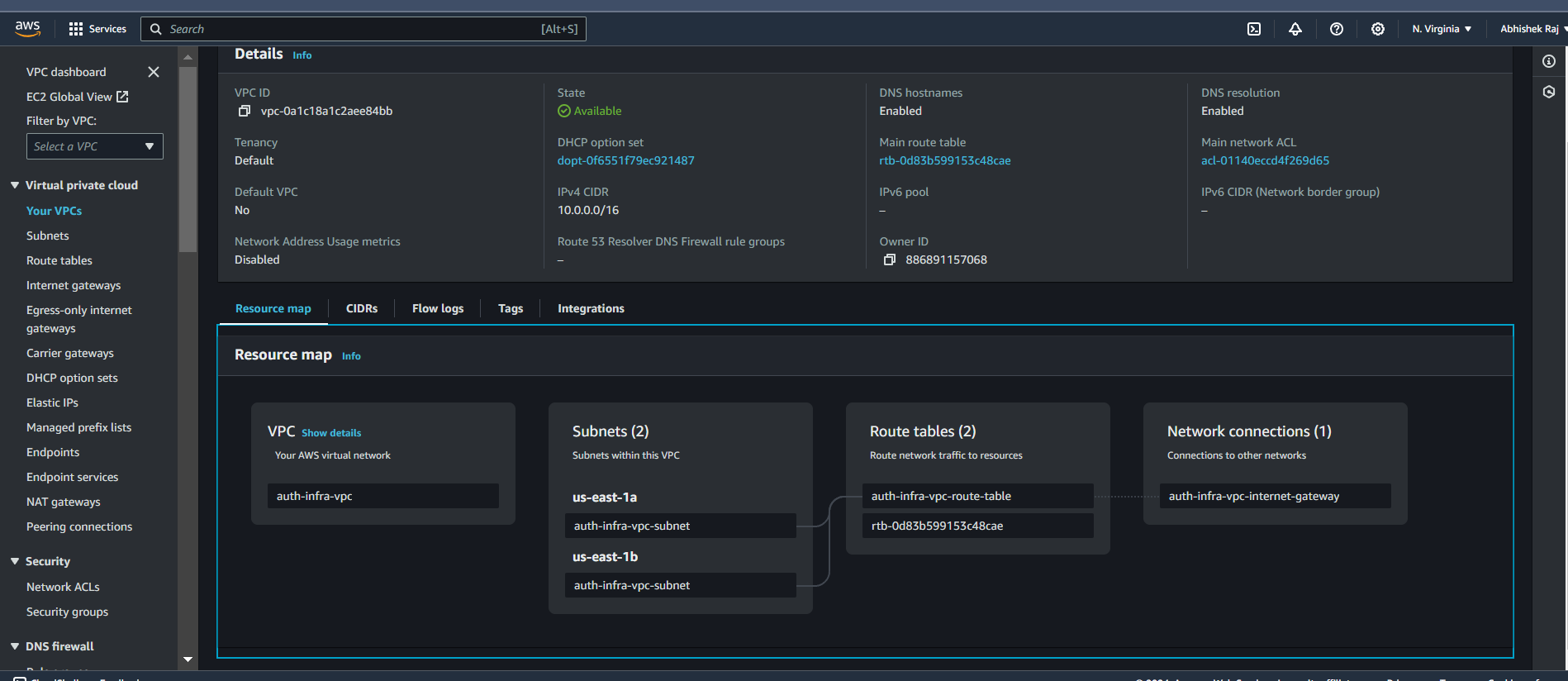
****

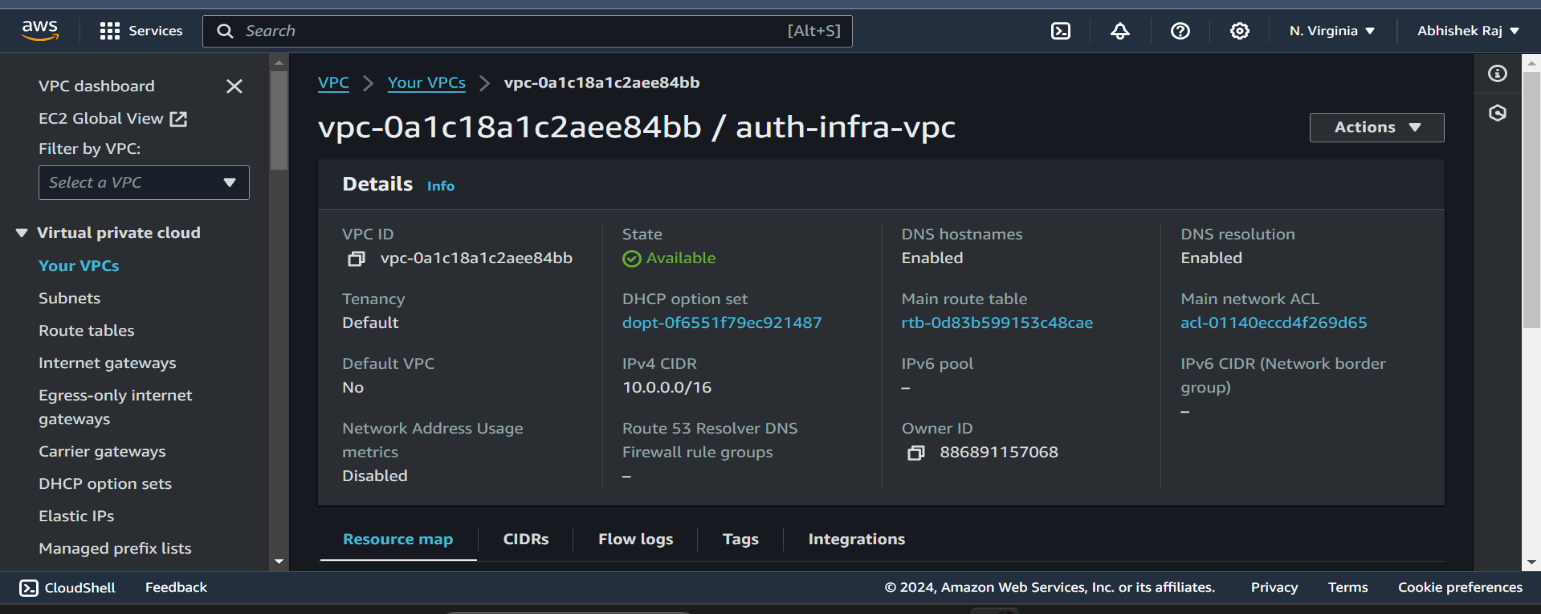
1. First client/user accesses the SSO login page where the user will provide his/her credentials. 2. After Successful Authentication on the SSO login page it sends a request to the identity provider in this case (LDAP).

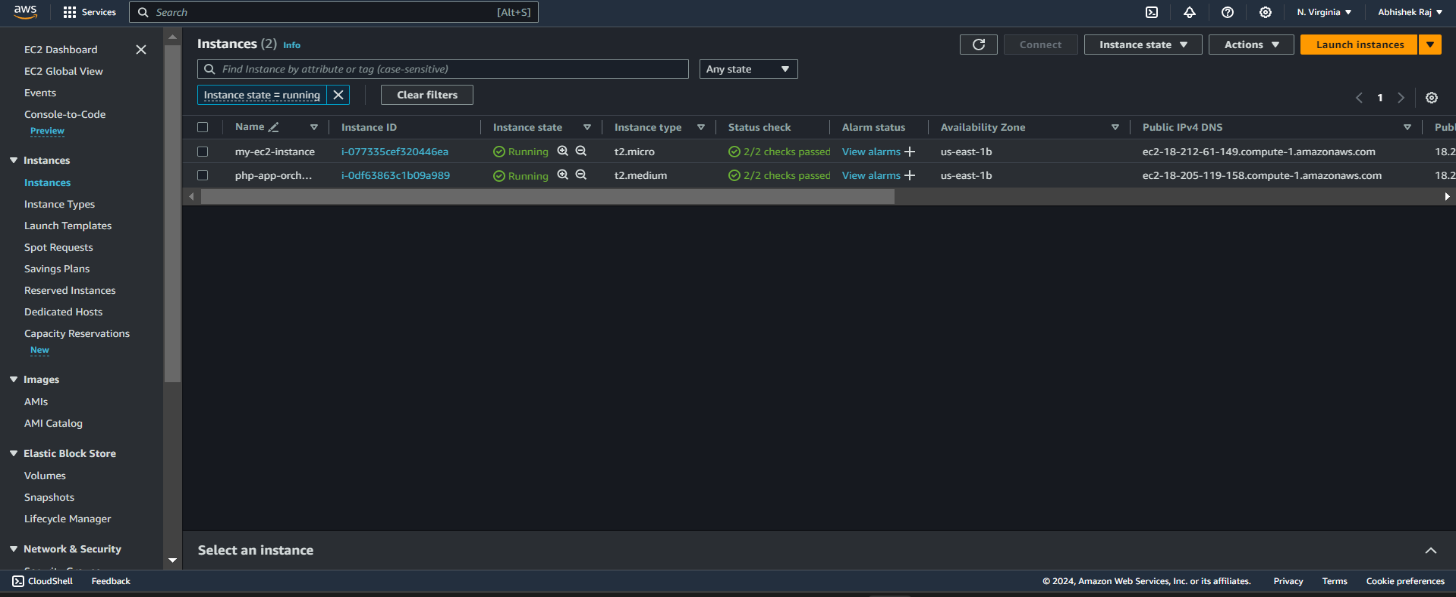
3. When a request is received by LDAP it checks the USER entry in its active directory.

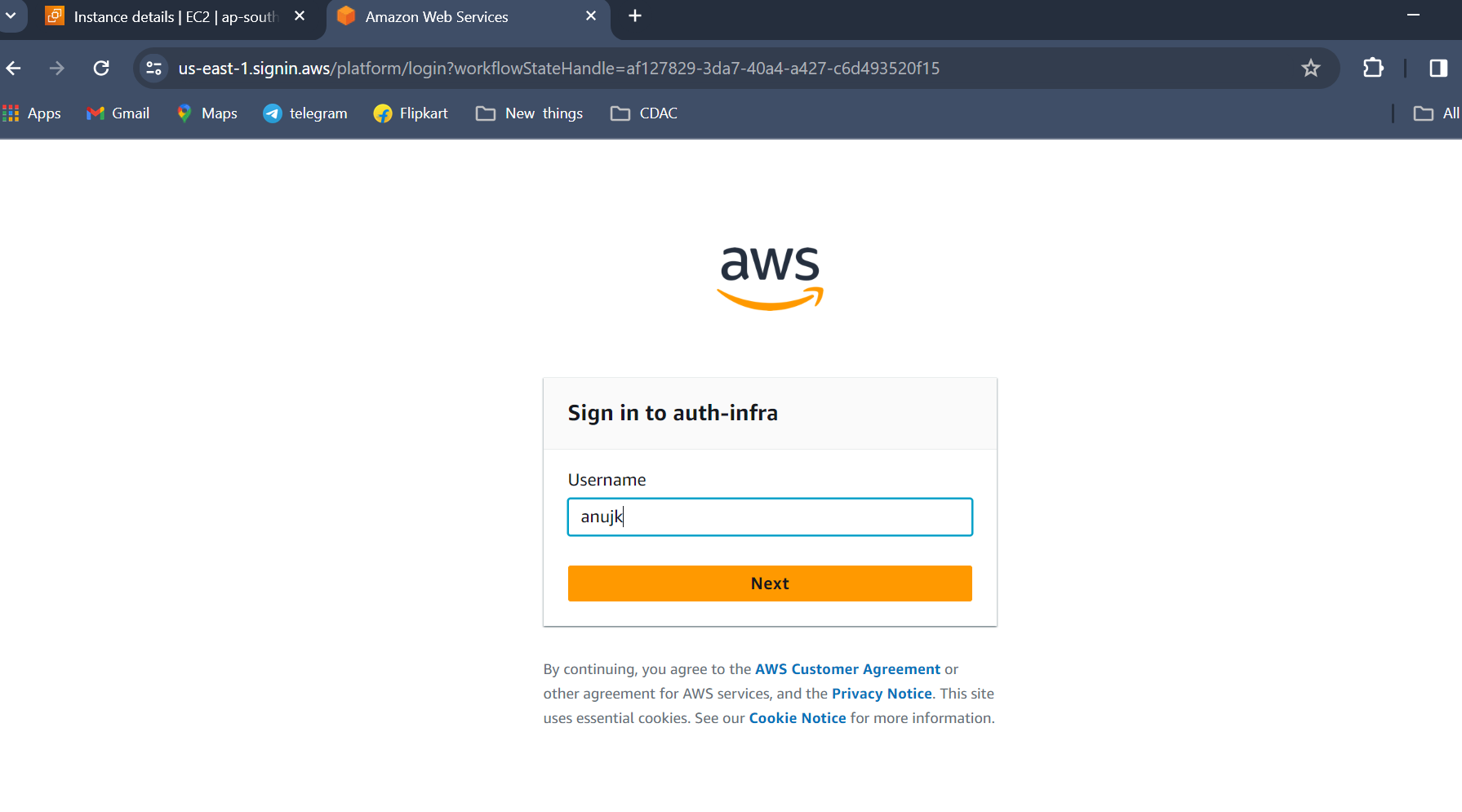
4. If this user entry is present there then the authorization process is completed for that user and the user gets access to those services which is being allowed to use.

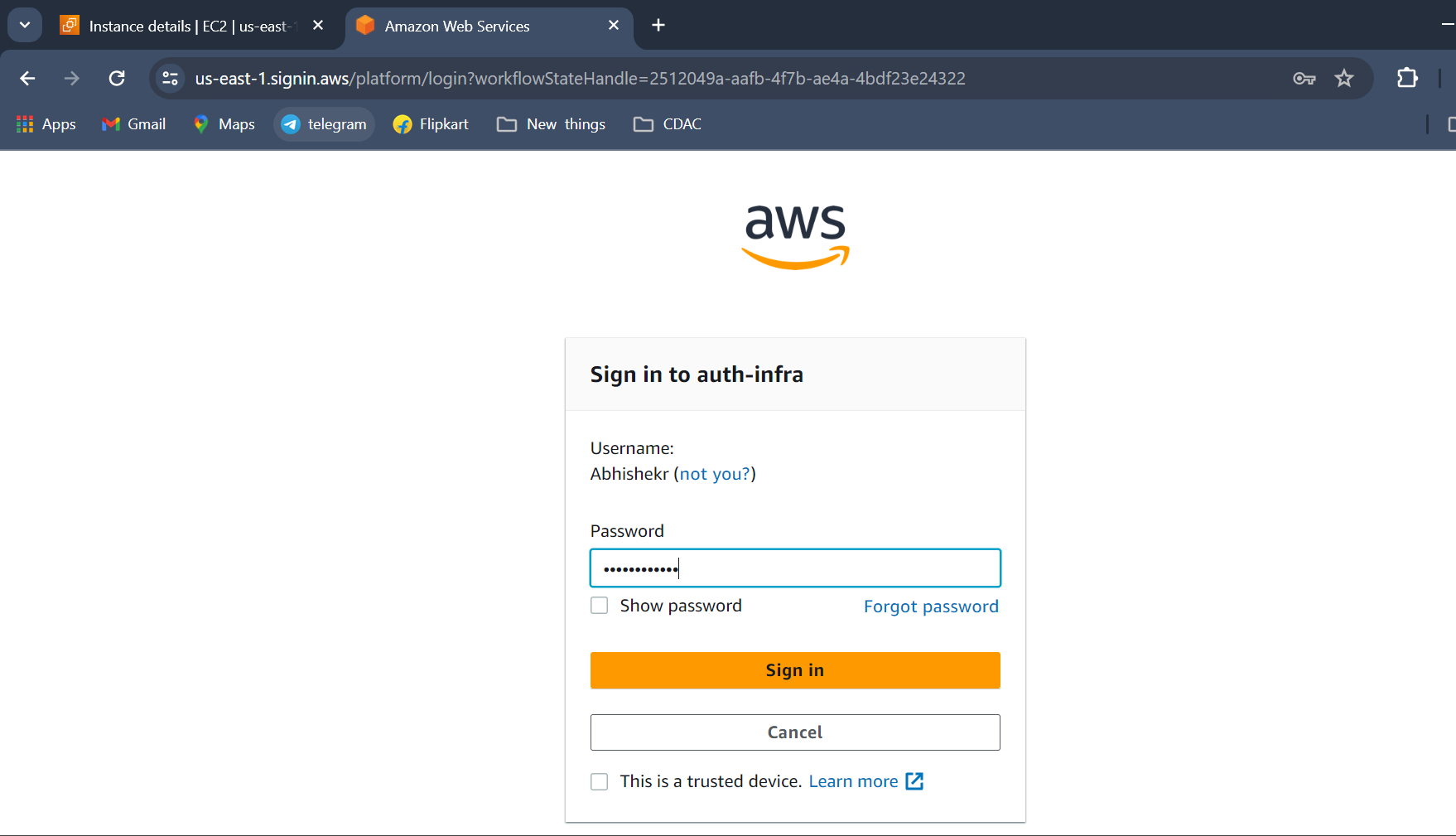
**IMPELEMENTAION**

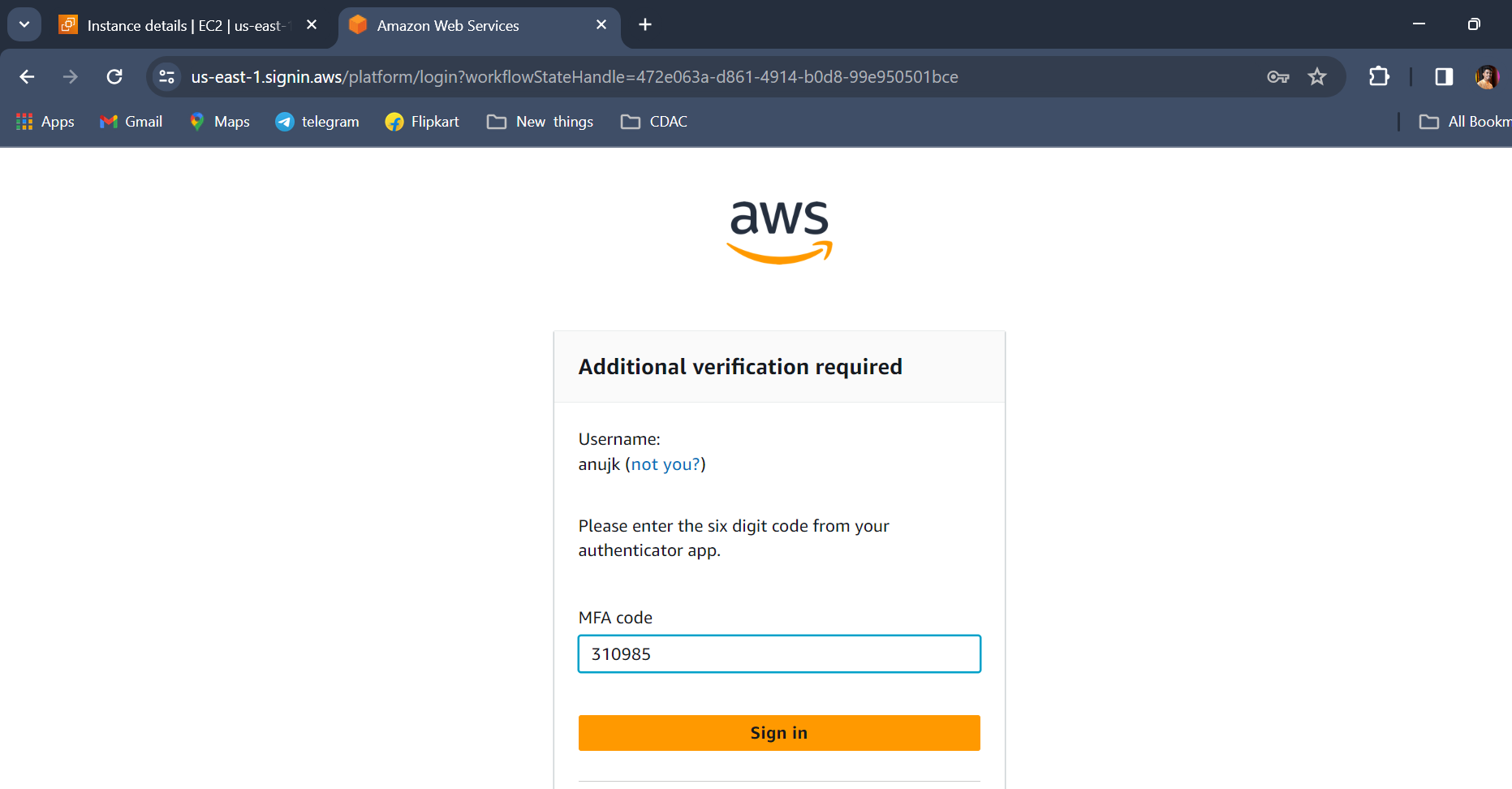
****

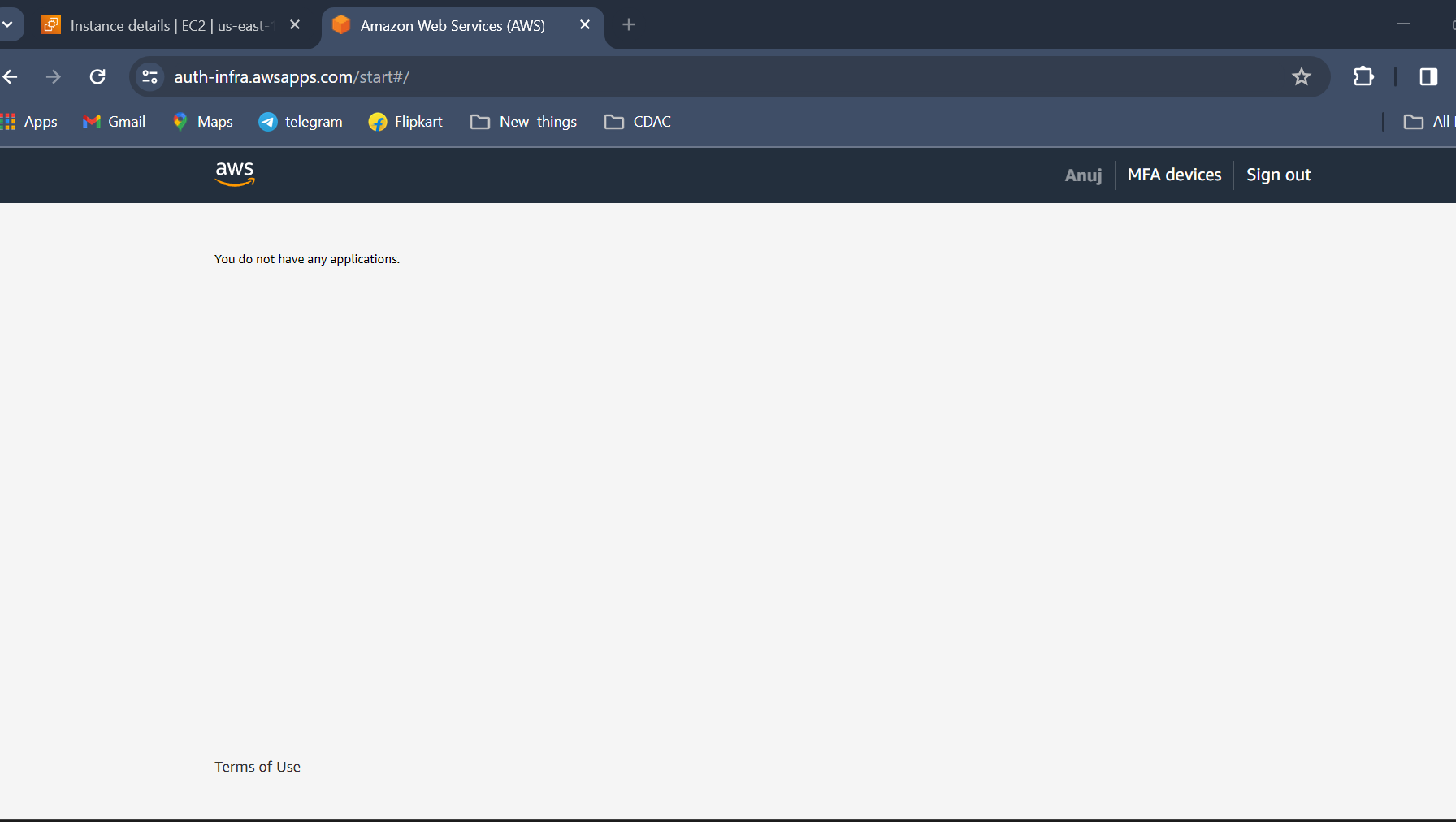
****

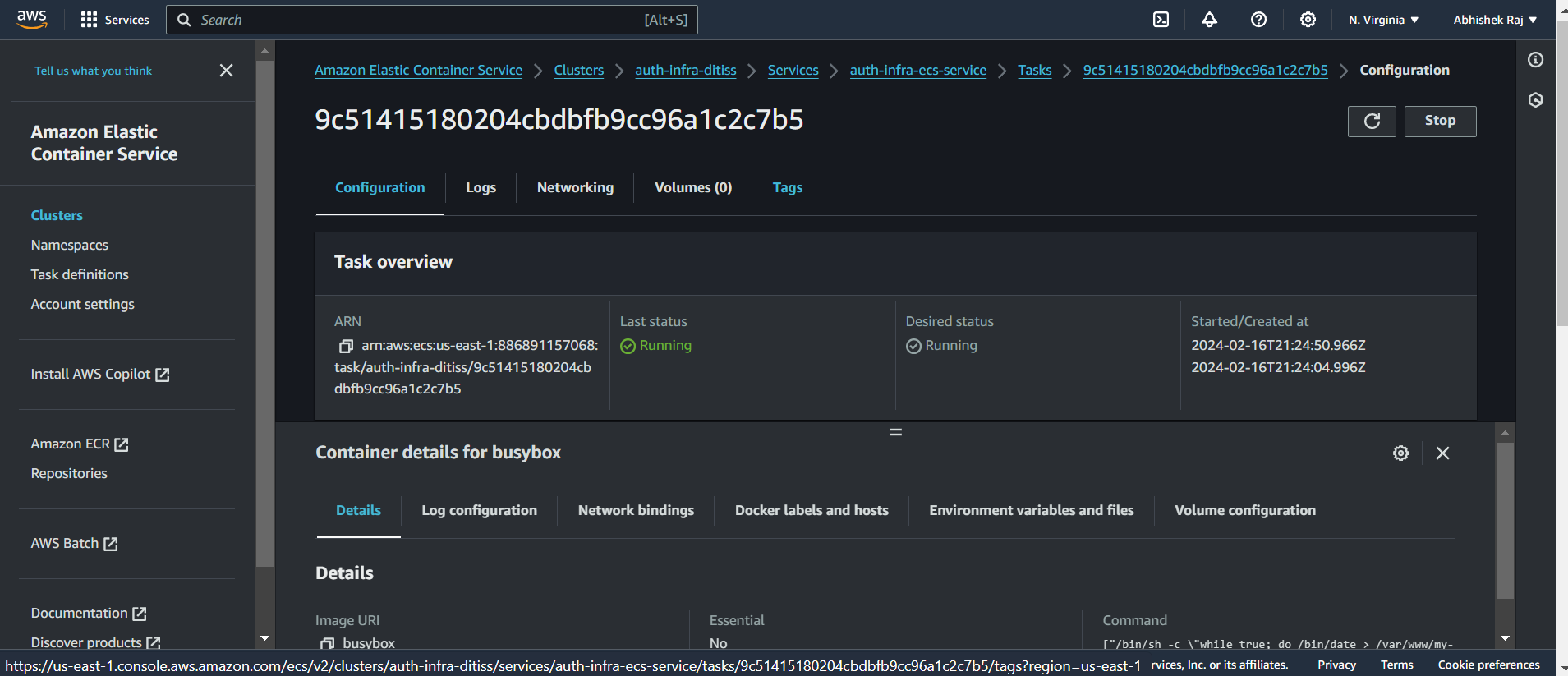
****

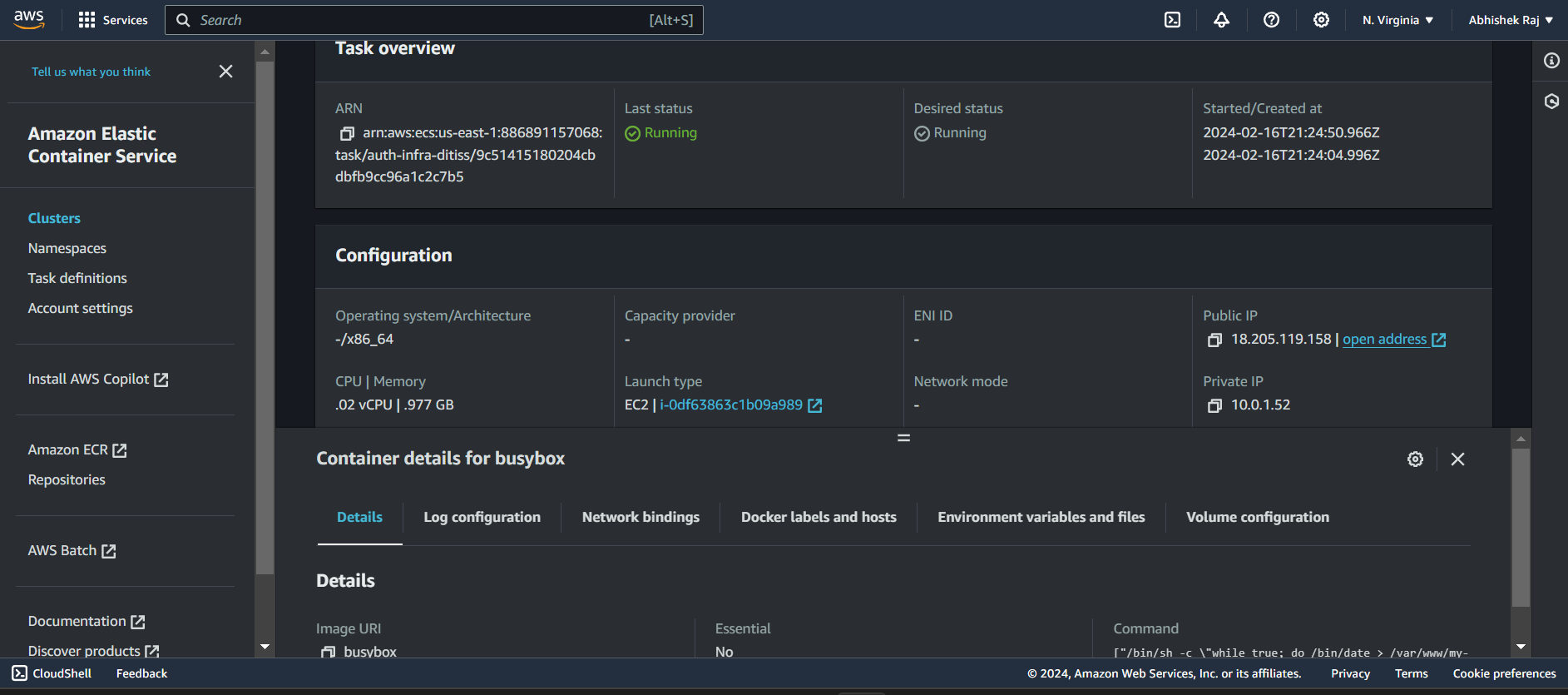


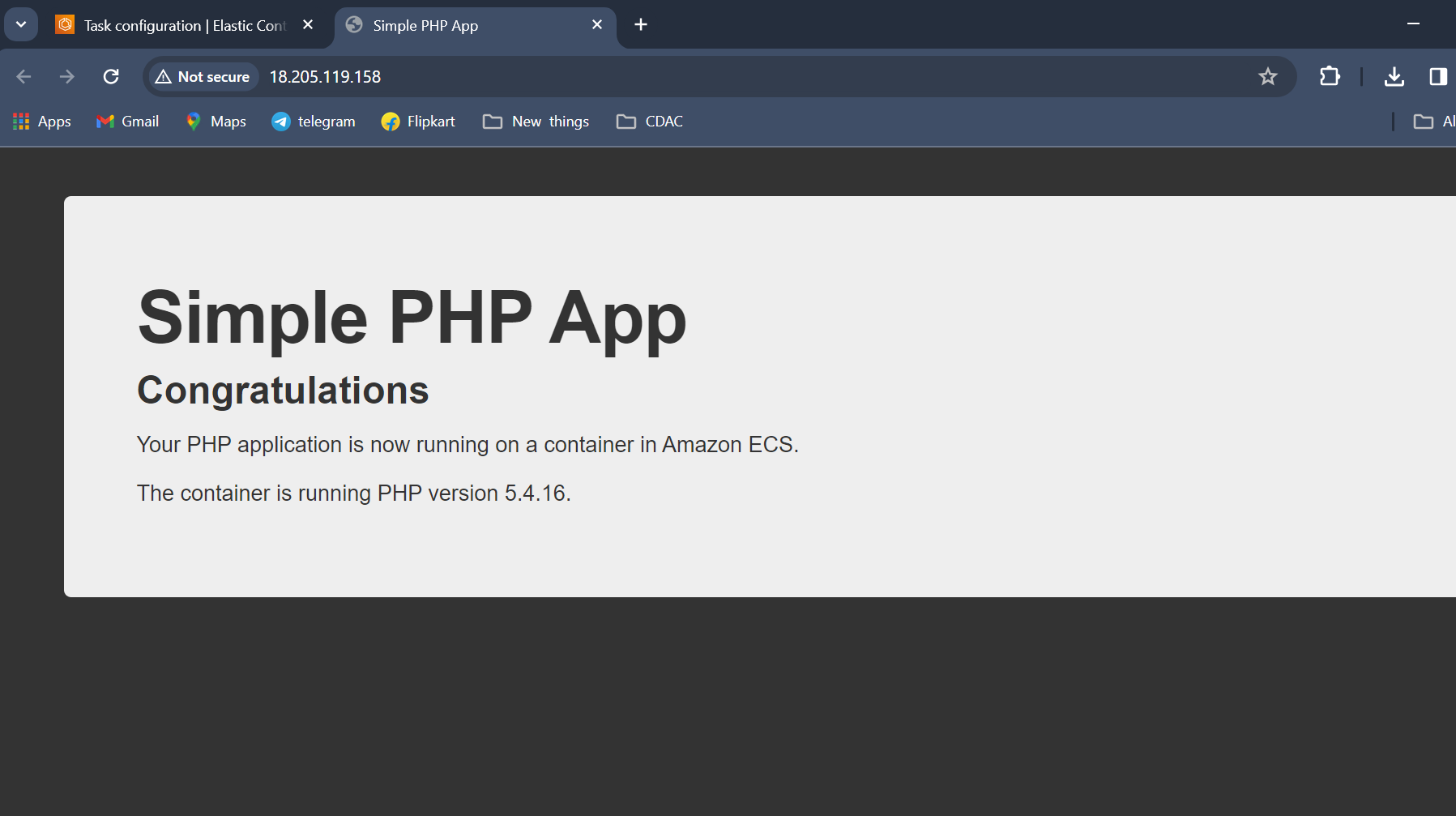


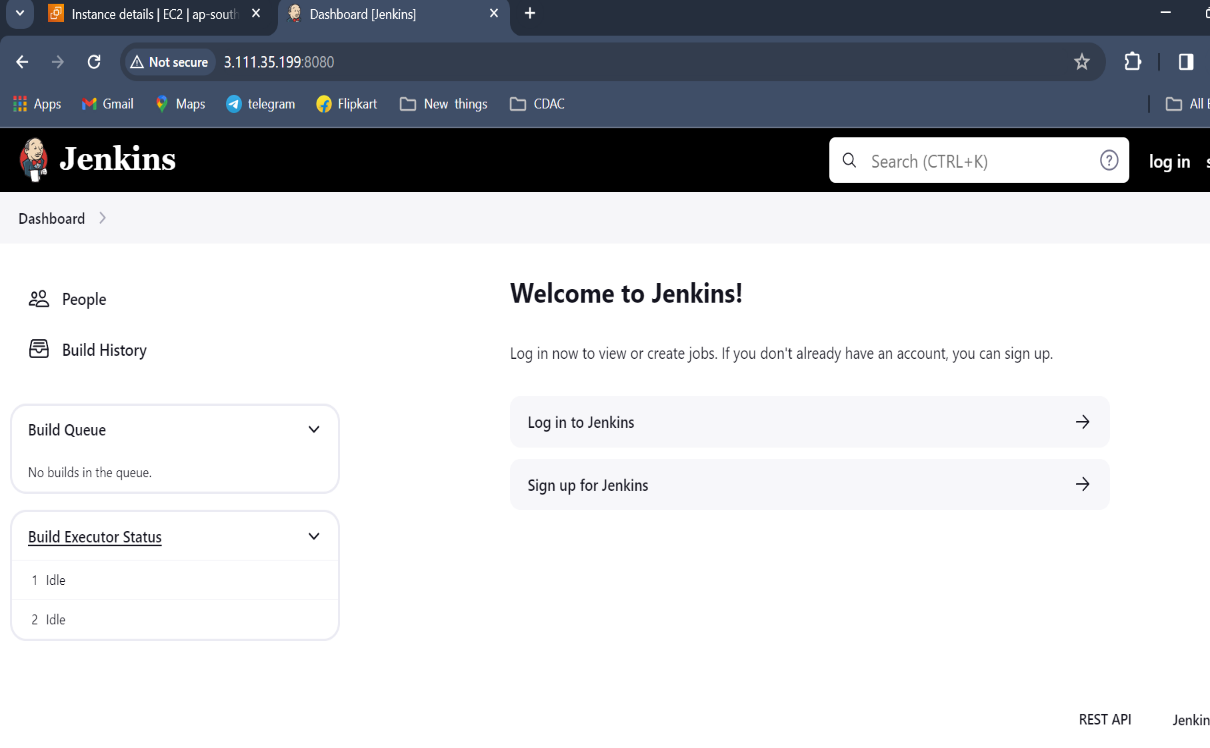


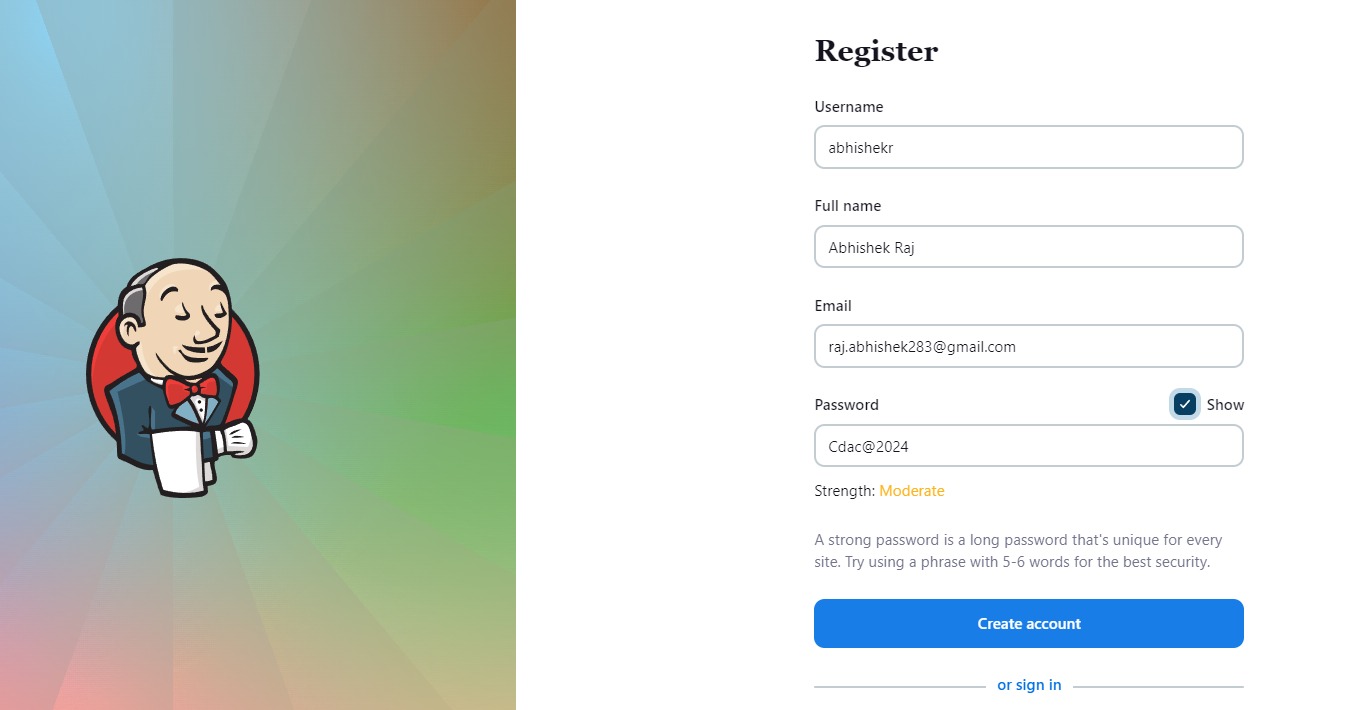




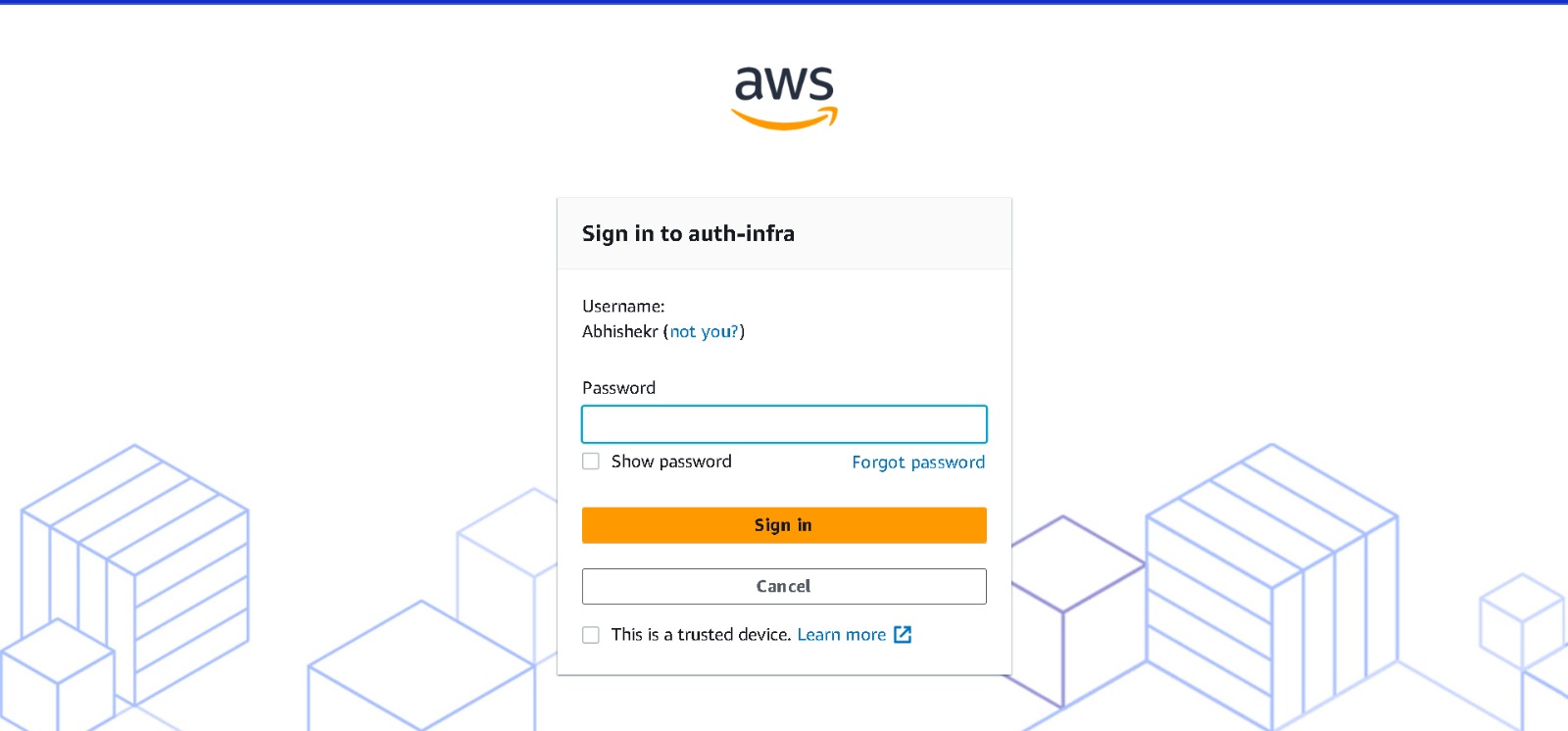


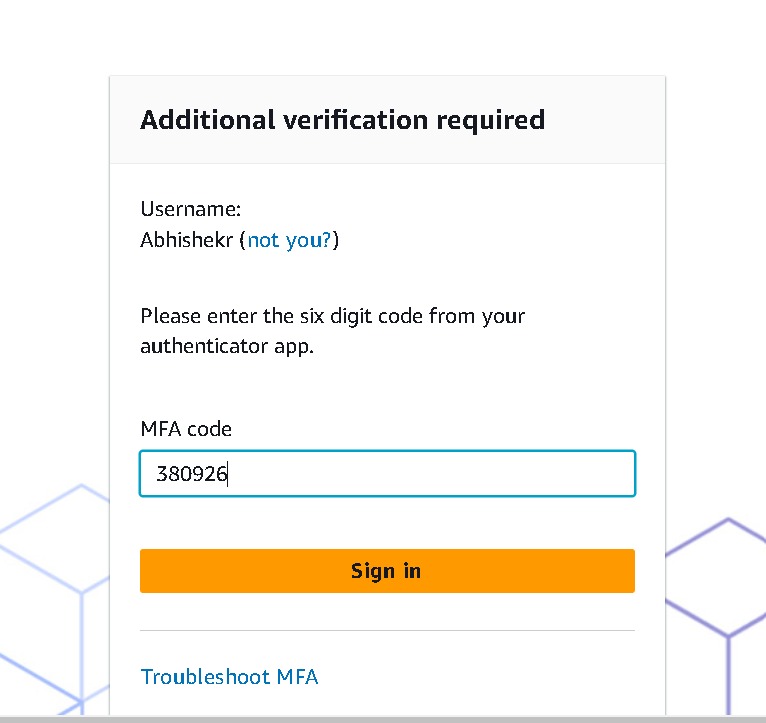


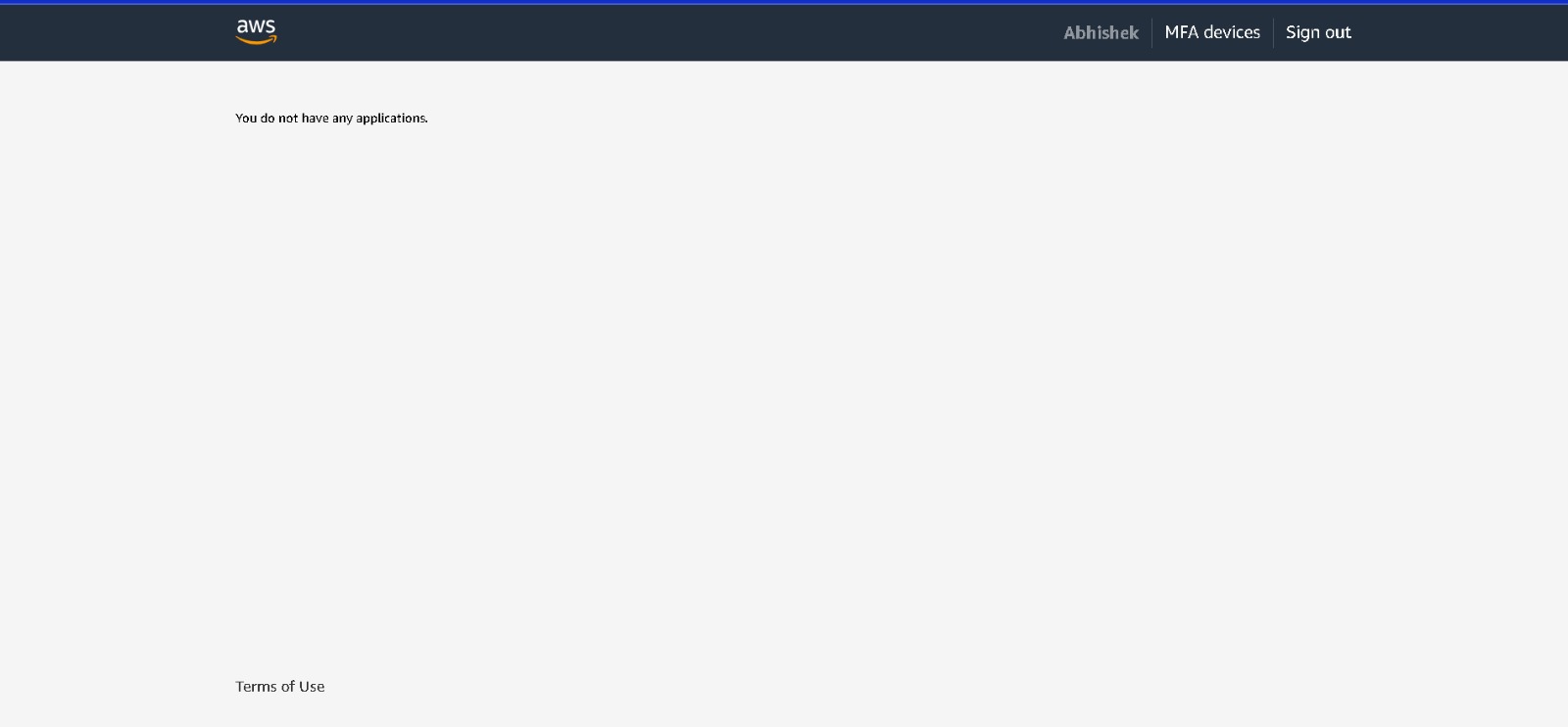


****

**OUTPUT:**

****

****

****

**REFERENCE:**

1. <https://guides.github.com/>
2. <https://docs.docker.com/>
3. <https://www.youtube.com/watch?v=SK8Yw-CiRHk&t=3s>
4. <https://www.youtube.com/watch?v=0FwOcZNjjQA&t=325s>
5. <https://www.youtube.com/watch?v=dBmxNsS3BGE>
6. <https://www.youtube.com/watch?v=C9tdKurLFoc>
7. <https://www.youtube.com/watch?v=23EghFNQdj4>