AWS Auto Scaling and EFS Project Documentation

# Project Overview

This project demonstrates how to create a highly available and scalable web infrastructure on AWS, leveraging Auto Scaling Groups, Elastic Load Balancers (ALB), and Elastic File System (EFS) for shared file storage across multiple EC2 instances. The goal is to ensure that multiple instances can access and update shared files, and these changes reflect across all instances automatically.

# Components Used

- Amazon EC2: Virtual servers to run the web application.  
- Amazon Elastic File System (EFS): Provides shared storage for the web servers.  
- Amazon Elastic Load Balancer (ALB): Distributes incoming traffic across multiple EC2 instances.  
- Auto Scaling Group: Automatically adjusts the number of EC2 instances based on traffic.  
- Amazon VPC: Provides networking for the resources, including subnets, route tables, and security groups.

# Project Architecture

The architecture includes:  
1. An Auto Scaling Group with EC2 instances behind an Application Load Balancer.  
2. Amazon EFS mounted to each EC2 instance to provide shared file storage.  
3. User data scripts ensure EFS is mounted on instance launch and web server configurations are automatically applied.

# Steps to Implement

## 1. Set Up VPC and Subnets

- Ensure you have a VPC with appropriate subnets across multiple Availability Zones for high availability.  
- Enable DNS hostnames and DNS resolution in your VPC.

## 2. Create the Elastic File System (EFS)

- Go to the EFS Console and create an EFS file system.  
- Create mount targets in the subnets corresponding to your EC2 instances.  
- Attach a security group that allows NFS (port 2049) access.

## 3. Create Security Groups

- EFS Security Group:  
 - Inbound: Allow NFS (port 2049) from the EC2 instance security group.  
- EC2 Security Group:  
 - Inbound: Allow HTTP (port 80), SSH (port 22) from anywhere or the Load Balancer's security group.  
 - Allow outbound traffic on NFS port 2049 to the EFS security group.

## 4. Launch Template and Auto Scaling Group

Create a Launch Template with the following configurations:  
- Instance type, AMI (Amazon Linux 2), and other configurations.  
- User Data Script to mount EFS and set up the web server:

#!/bin/bash

yum update -y

yum install -y httpd

systemctl start httpd

systemctl enable httpd

sudo yum -y install nfs-utils

sudo yum install -y amazon-efs-utils

sudo mount -t nfs -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-00becd30c90852c5d.efs.ap-south-1.amazonaws.com:/ /var/www/html

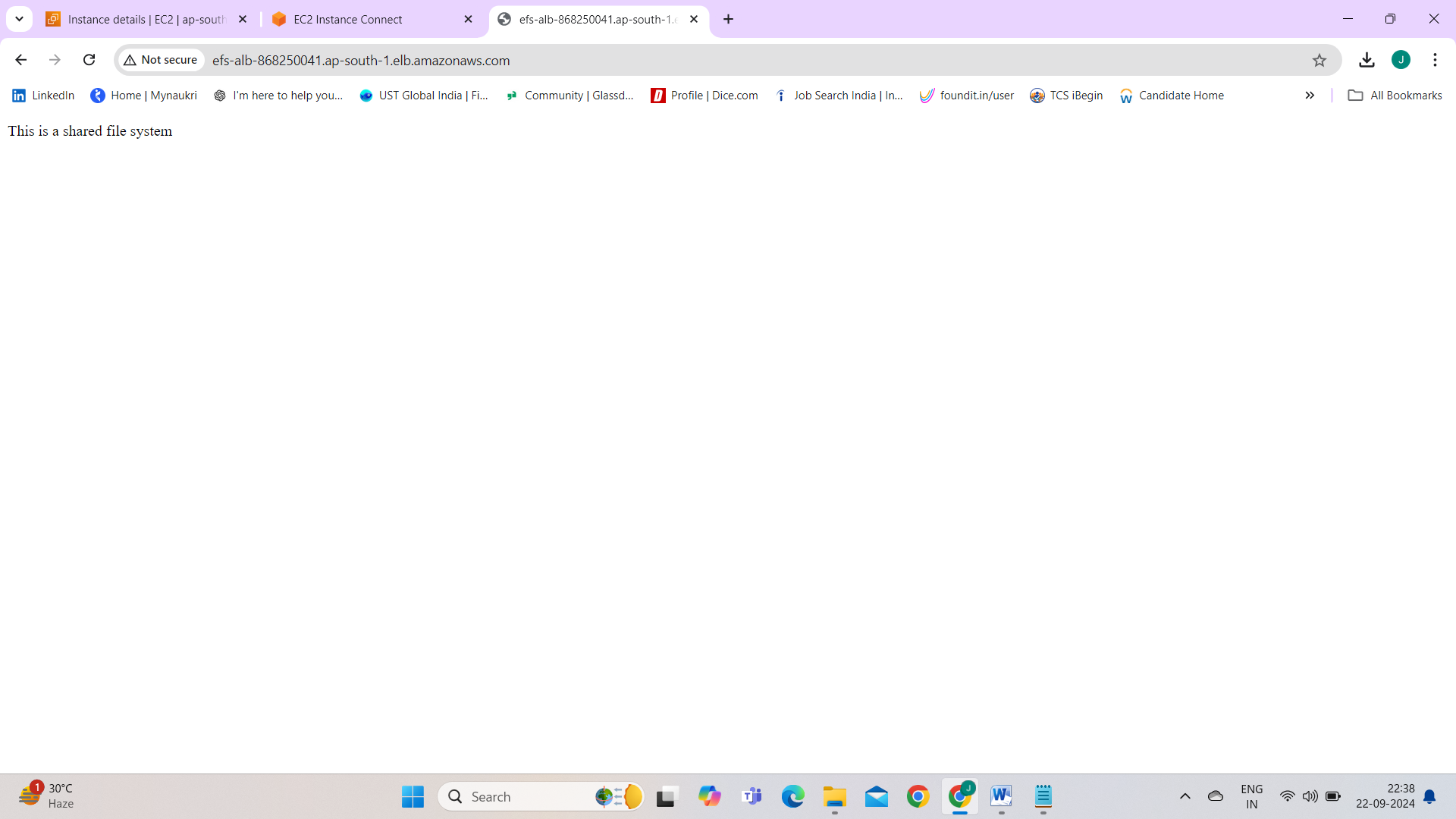
echo "This is a shared file system" > /var/www/html/index.html

## 5. Create an Application Load Balancer (ALB)

- Create an ALB in the EC2 Console.  
- Add at least two Availability Zones for high availability.  
- Associate the ALB with the security group that allows HTTP (port 80) traffic.  
- Create a Target Group with HTTP protocol and add the Auto Scaling Group to it.

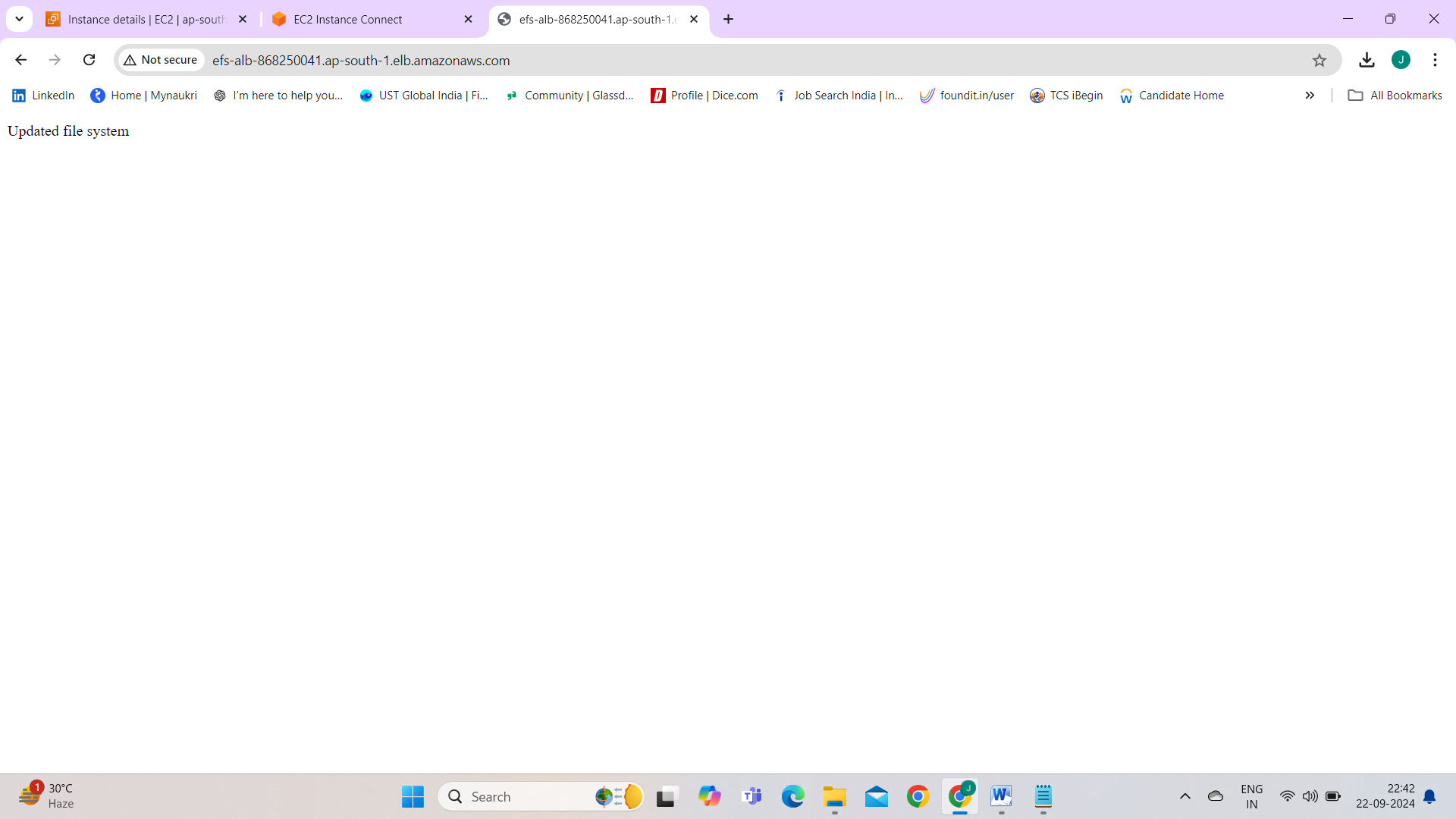
## 6. Test the Setup

- After everything is set up, visit the DNS name of your ALB in the browser:  
 http://<ALB\_DNS\_NAME>  
- Verify the content served is the same across all EC2 instances.

  
- SSH into one of the EC2 instances and update the index.html file in /var/www/html:

echo "Updated file system" > /var/www/html/index.html

- Confirm the changes reflect across all instances via the Load Balancer.



# Troubleshooting

## EFS Not Mounting

- Ensure EFS utilities are installed on all instances:  
 sudo yum install -y amazon-efs-utils  
- Verify security group rules allow NFS (port 2049) from EC2 instances to EFS.  
- Check if mount targets are available in the correct subnets.

## DNS Hostname Issue

- Ensure \*\*DNS hostnames\*\* and \*\*DNS resolution\*\* are enabled in your VPC.  
- If this is not enabled, the instances will fail to resolve the EFS mount point DNS name, resulting in an error such as:  
 mount.nfs: Failed to resolve server fs-xxxxxxxxx.efs.ap-south-1.amazonaws.com: Name or service not known.

## Changes Not Reflected Across Instances

- Ensure all instances have correctly mounted the EFS:  
 df -h | grep efs  
- Check Apache (or Nginx) configuration to ensure the document root points to /var/www/html

# Conclusion

This setup ensures that your application is highly available and scalable with shared storage, making it ideal for scenarios where you need consistent data across multiple instances. You can now automatically scale your EC2 fleet and share files between them using Amazon EFS.