EC2 Auto Scaling with Load Balancer and EFS

**Project Overview**

The objective of this project is to build a scalable and resilient web application infrastructure on AWS that leverages Amazon EC2 instances, Auto Scaling, an Application Load Balancer, and Amazon Elastic File System (EFS). This architecture aims to deliver high availability and consistency for web content across multiple instances.

**Purpose**

**Centralized Content Management**: By using Amazon EFS, the project enables a shared file system where the index.html file is stored. This allows all EC2 instances to access the same content, ensuring that any updates made to the file are reflected across all instances in real time.

**Scalability**: The Auto Scaling feature allows the infrastructure to automatically adjust the number of EC2 instances based on traffic demand, ensuring optimal performance during peak loads while minimizing costs during low traffic periods.

**High Availability**: With an Application Load Balancer distributing incoming traffic across multiple EC2 instances, the setup enhances fault tolerance and maintains uptime, even if one or more instances fail.

**Cost Efficiency**: The combination of Auto Scaling and EFS allows for efficient resource utilization, ensuring that you only pay for what you need while maintaining a responsive web application.

This project provides a robust and flexible environment for hosting web applications, leveraging the capabilities of AWS to manage content effectively and respond dynamically to user demands.

**Components**

**Amazon EC2 (Elastic Compute Cloud):** Virtual servers in the cloud that run applications. EC2 instances host the web application, allowing for scalable compute capacity based on traffic demands.

**Auto Scaling: A service that automatically adjusts the number of EC2 instances in response to current demand. It ensures that the application can handle varying levels of traffic efficiently while maintaining performance.**

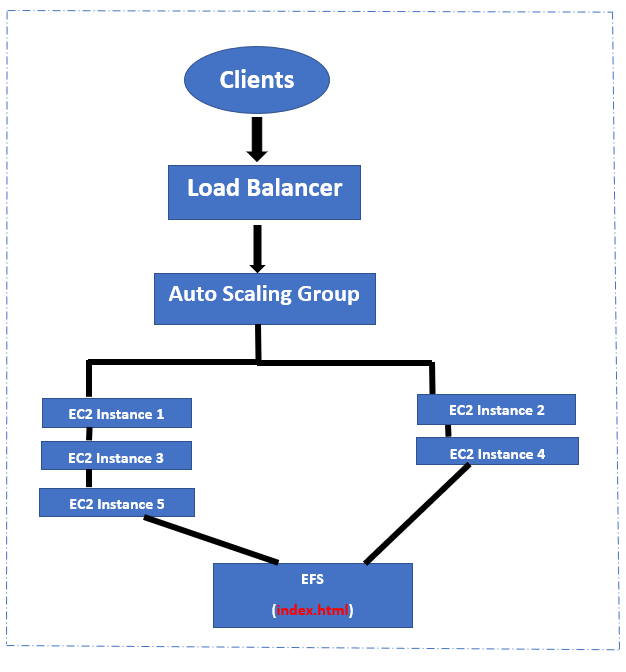
**Application Load Balancer (ALB):** A type of load balancer that distributes incoming traffic across multiple EC2 instances. The ALB enhances fault tolerance and ensures high availability by routing user requests to the healthiest instances.

**Amazon EFS (Elastic File System):** A fully managed, scalable file storage service that allows multiple EC2 instances to share a common file system. EFS stores the index.html file, ensuring that any updates are accessible to all instances, providing consistency in content delivery.

**VPC (Virtual Private Cloud):** A logically isolated section of the AWS cloud where the resources are launched. The VPC provides enhanced security and control over the network settings for the instances and other components.

**Security Groups:** Virtual firewalls that control inbound and outbound traffic for the EC2 instances and load balancer. They ensure that only authorized traffic can reach the application.

**Architecture Diagram**



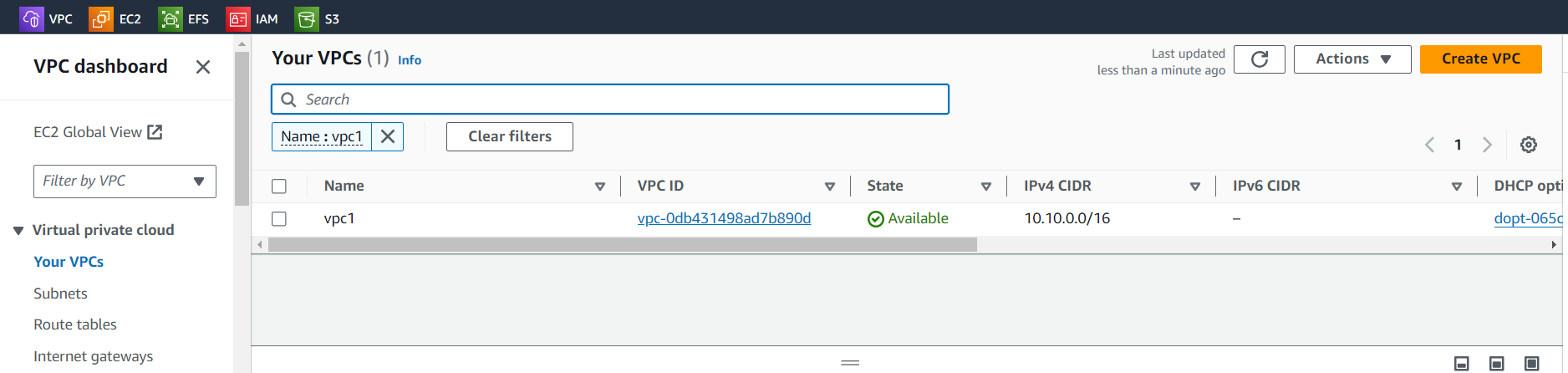
**Prerequisites**

**AWS Account**: Ensure you have access to an AWS account.

**IAM Roles/Permissions**: create IAM role and set permission as Amazon Elastic File System Client Read Write Access.

**Setup Instructions**

**VPC Creation**: After login in to the AWS management console click create VPC. Then provide a name and set IPv4 CIDR block: 10.10.0.0/16, set tendency as default and then click create.

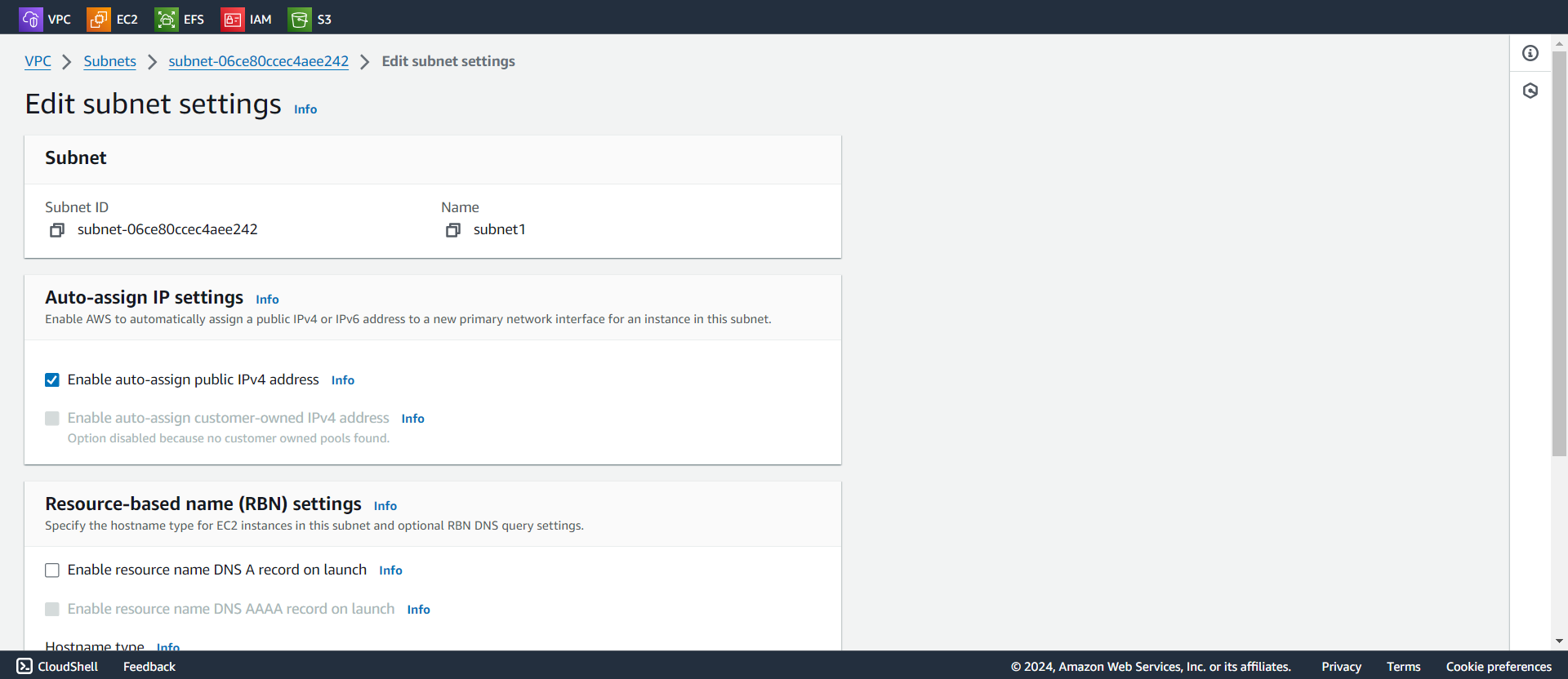


**Create Subnets:** Create at least two subnets (one for public and one for private)

**Public Subnet**: For public subnet set a name and choose the created VPC. Then set IPv4 CIDR block: 10.10.1.0/24 and availability zone

**Private Subnet**: For public subnet set a name and choose the created VPC. Then set IPv4 CIDR block: 10.10.2.0/24 and availability zone

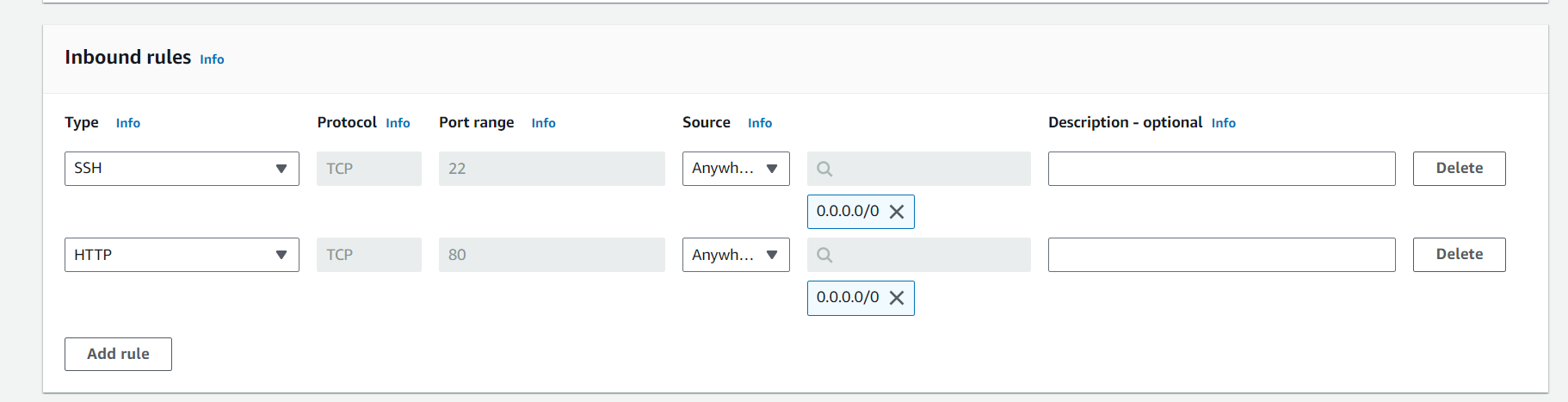
While creating subnets enable the auto-assign public IPv4 address



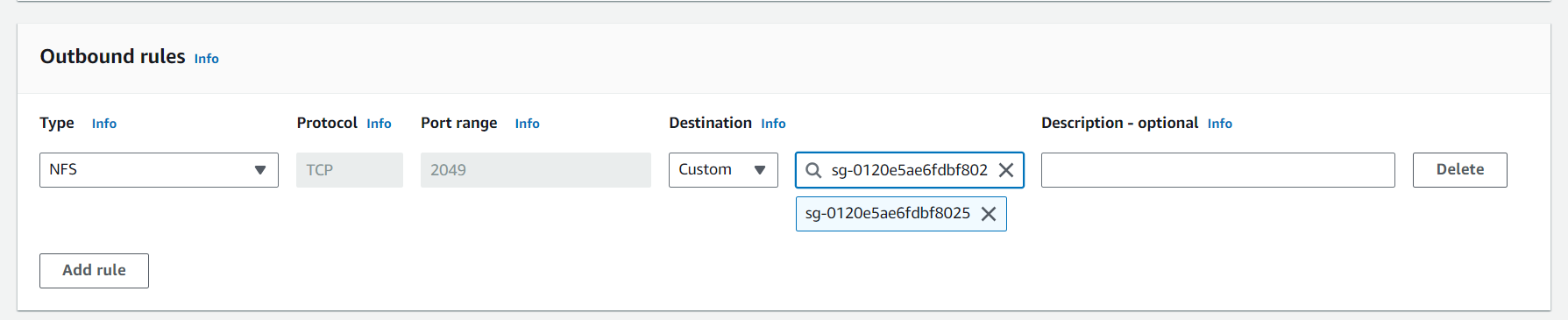
**Create an Internet Gateway: create an internet gateway then Attach it to VPC and enter the route table entries and associate it with the created subnets**

**Create Security Groups: create a two security groups one for webserver and one for EFS**

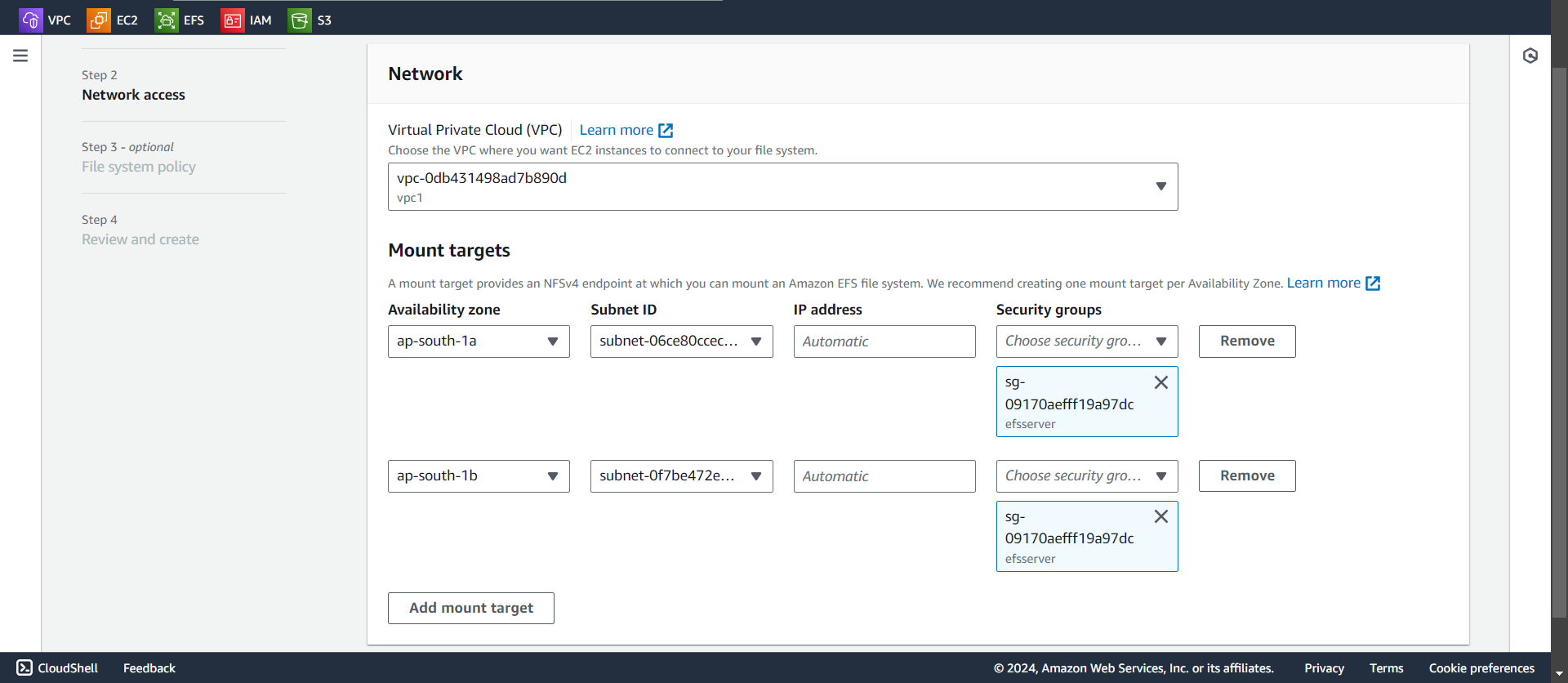
**Webserver:** In webserver choose the created VPC and add two inbound rules SSH and HTTP then set source as anywhere from IPv4



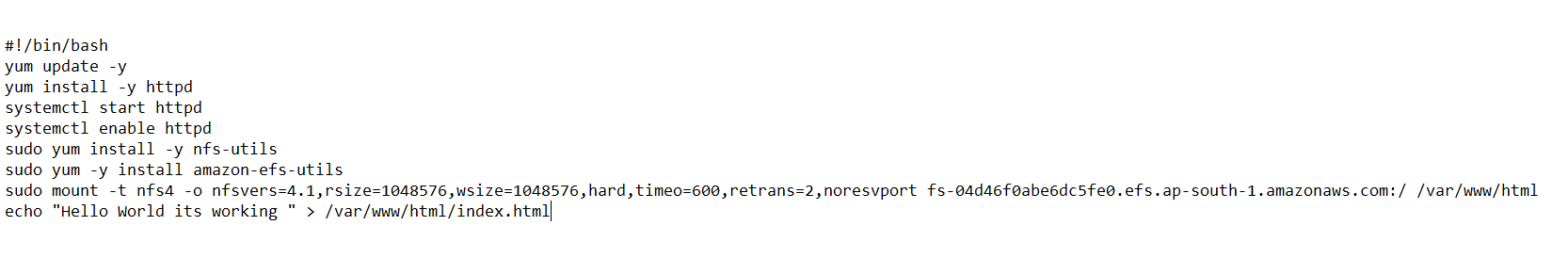
**EFS:** For EFS also choose the created VPC and add only one inbound rule NFS then set the source as custom and choose the created webserver security group



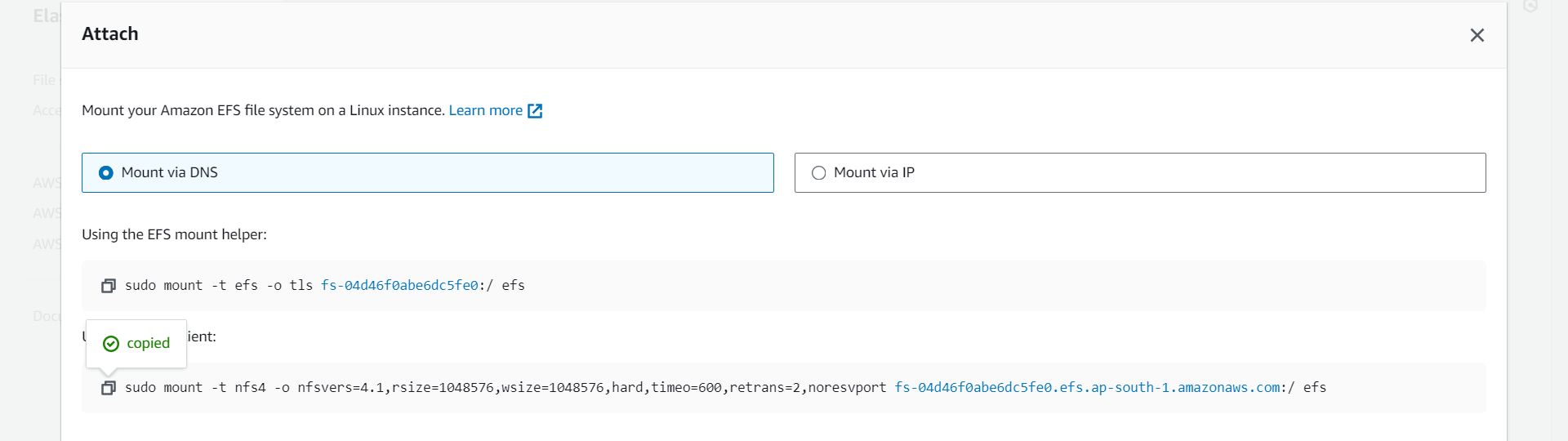
**Create an EFS File System: Create a file system with created VPC and in networks choose the mount targets in the created availability zones and choose the EFS security group**



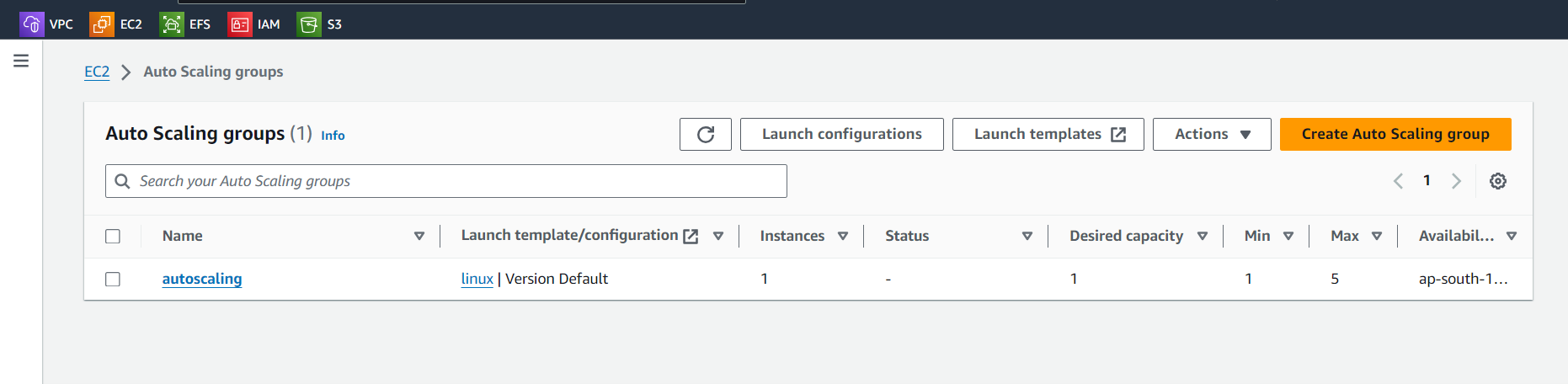
**Launch EC2 Instances with a Launch Template:** Navigate to EC2 and then click launch template provide a name, choose created VPC, instance type, security group (webserver security group), IAM role and add user data



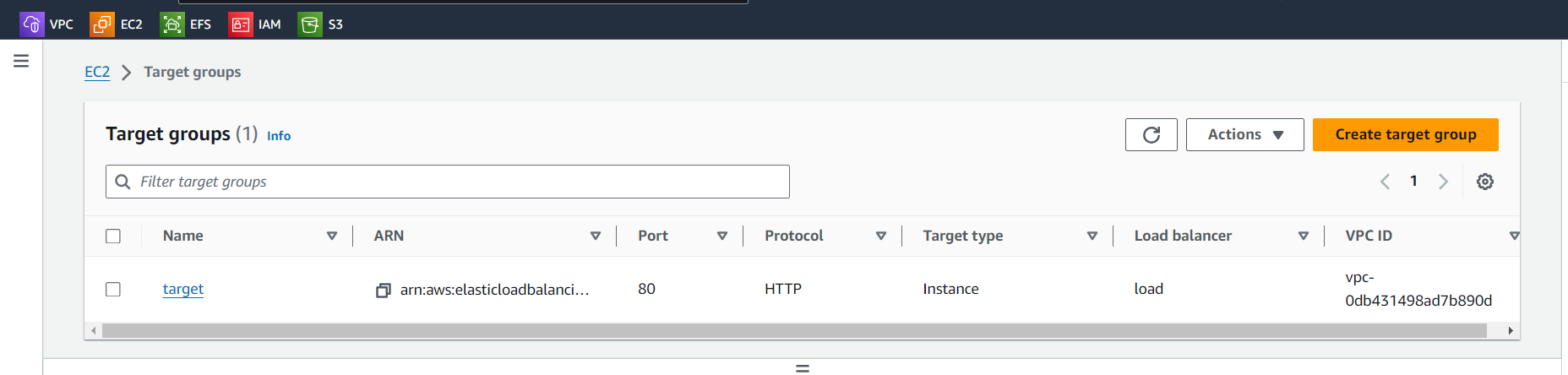
The EFS mount command can be extracted from the EFS file system



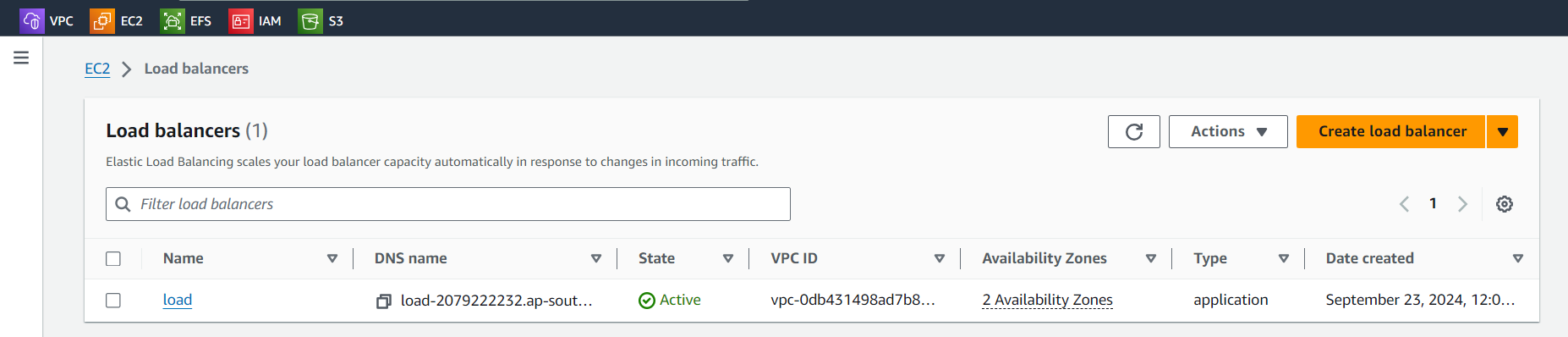
**Create Auto Scaling Group:** create auto scaling group by providing name, launch template, VPC and subnets and set the desired capacity to five and click create.



**Create Target Group:** create target group with providing name, target type instance, protocol as HTTP 80 and health check as /index.html and then after created attach the target group with the already created auto scaling group.

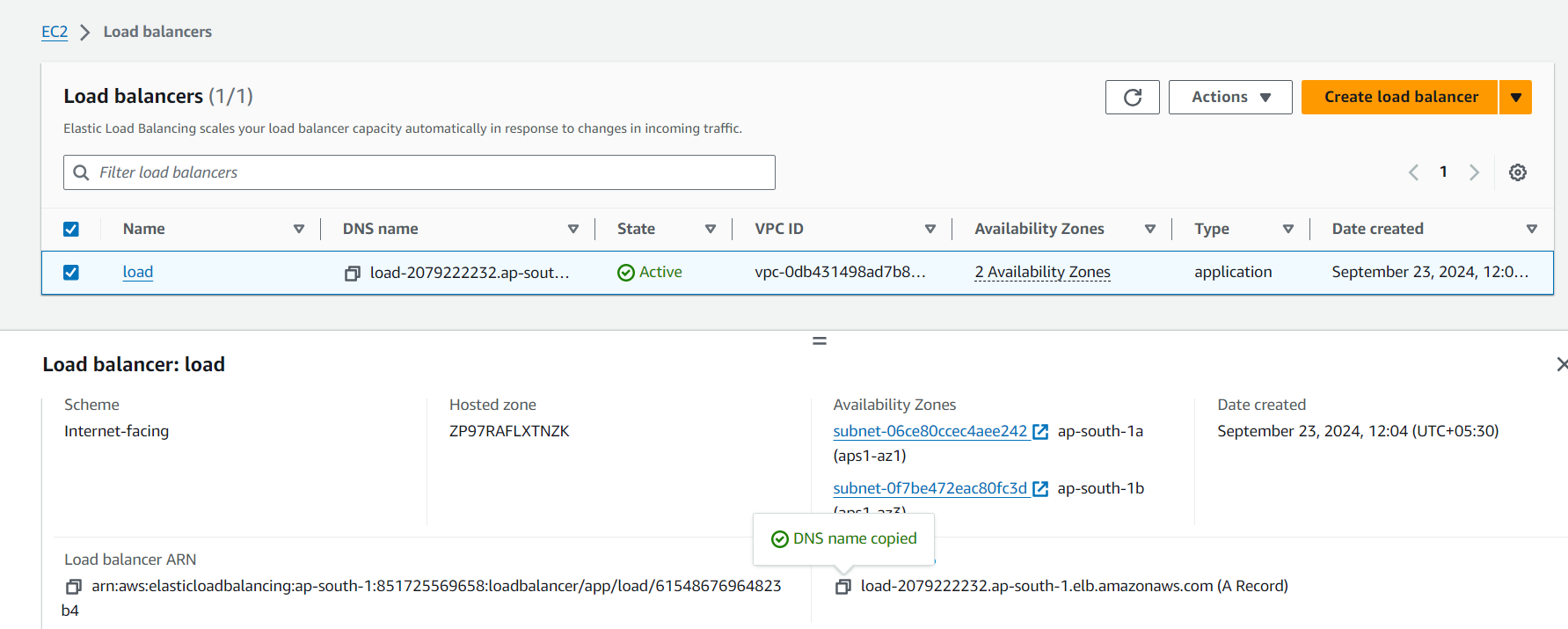


**Create Load Balancer:** create load balancer by providing name, VPC, listeners as HTTP 80, select the availability zone and attach the target group.

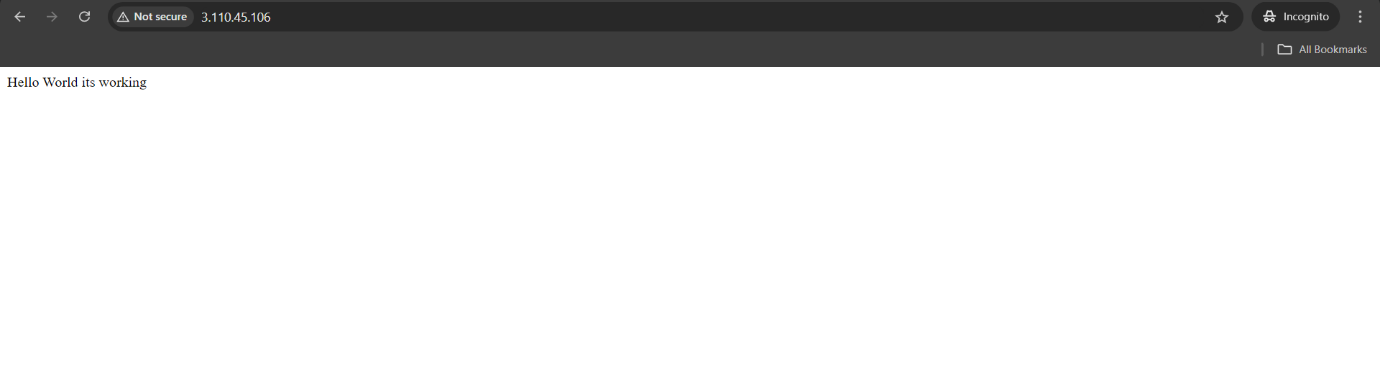
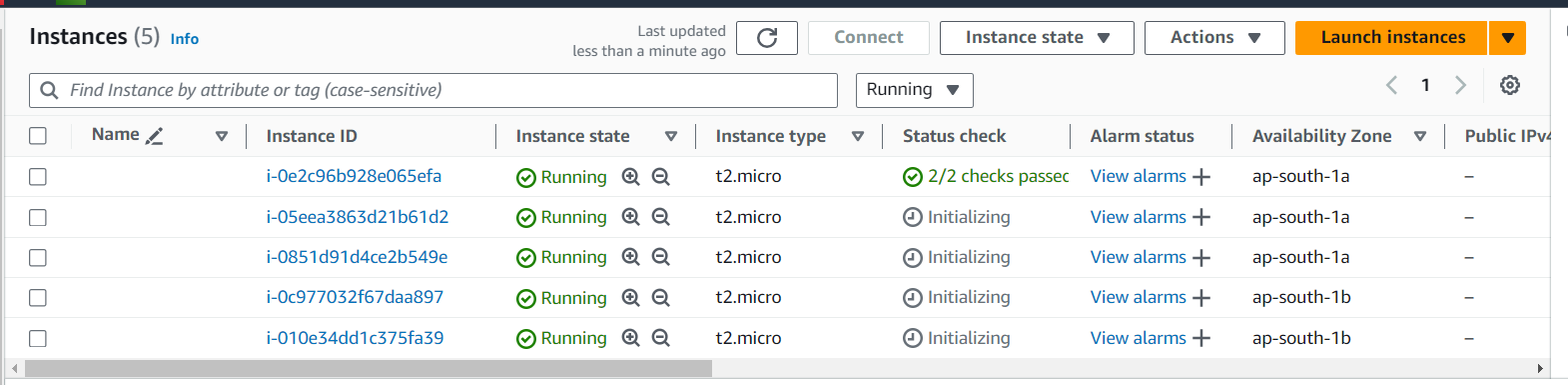


**Deploy the Application**

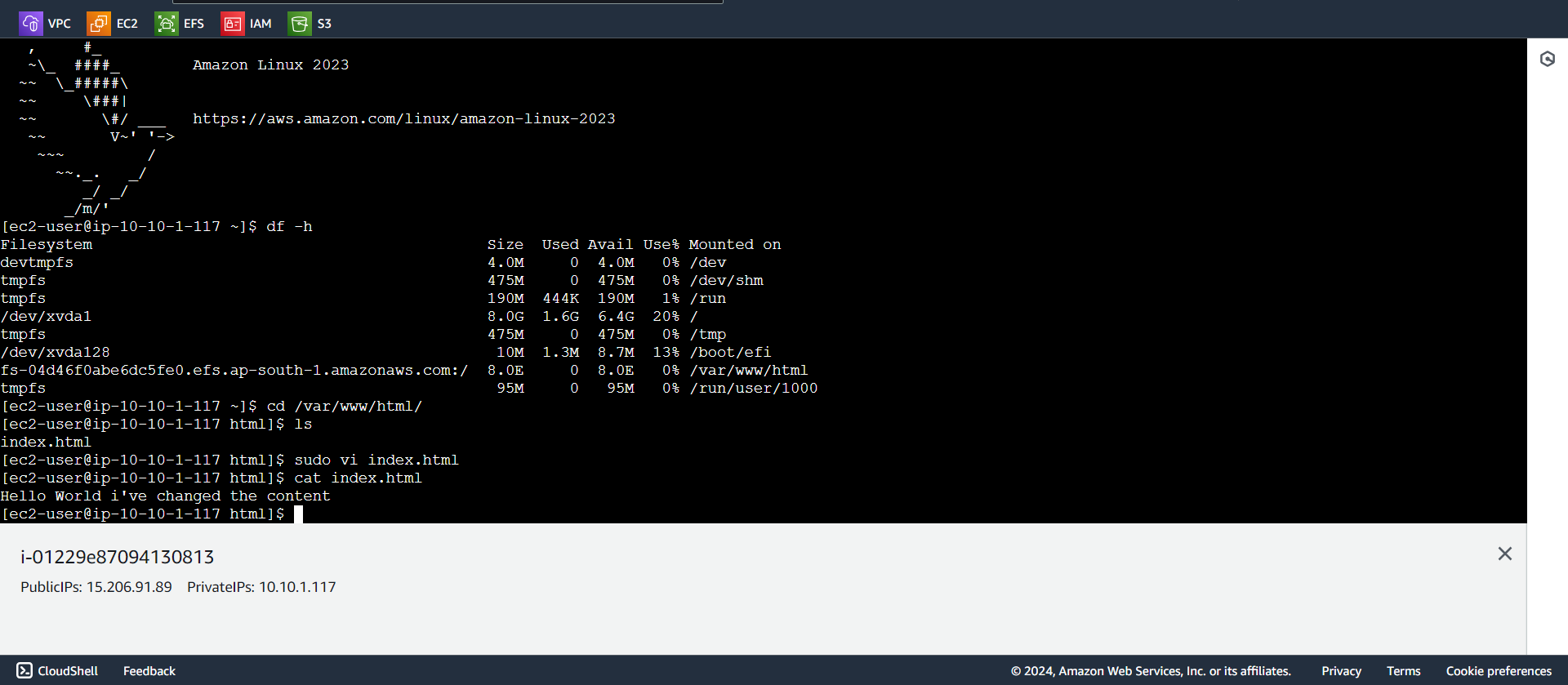
Navigate to the DNS name of your Load Balancer in a web browser.

****

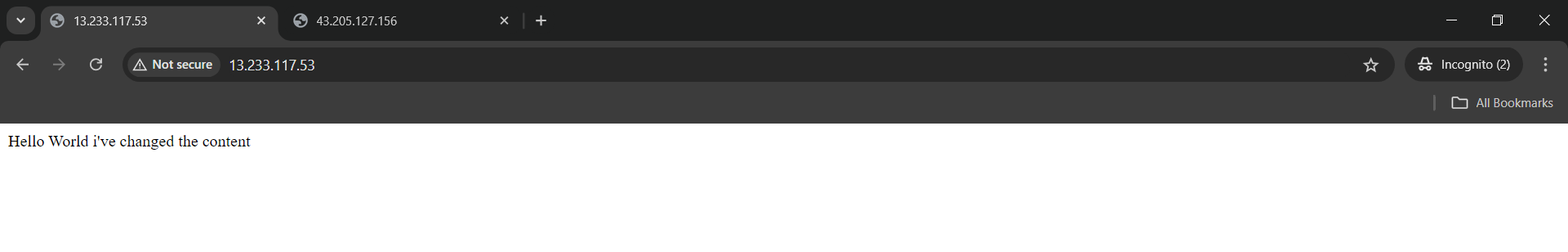
Verify that the page displays the content from index.html.

****Depending upon the increase of traffic the EC2 instances are created based on auto scaling and load balancer.

Then change the content in index.html file located in any one of the EC2 instance



Then hit the different instances and verify the changes made are reflected all across

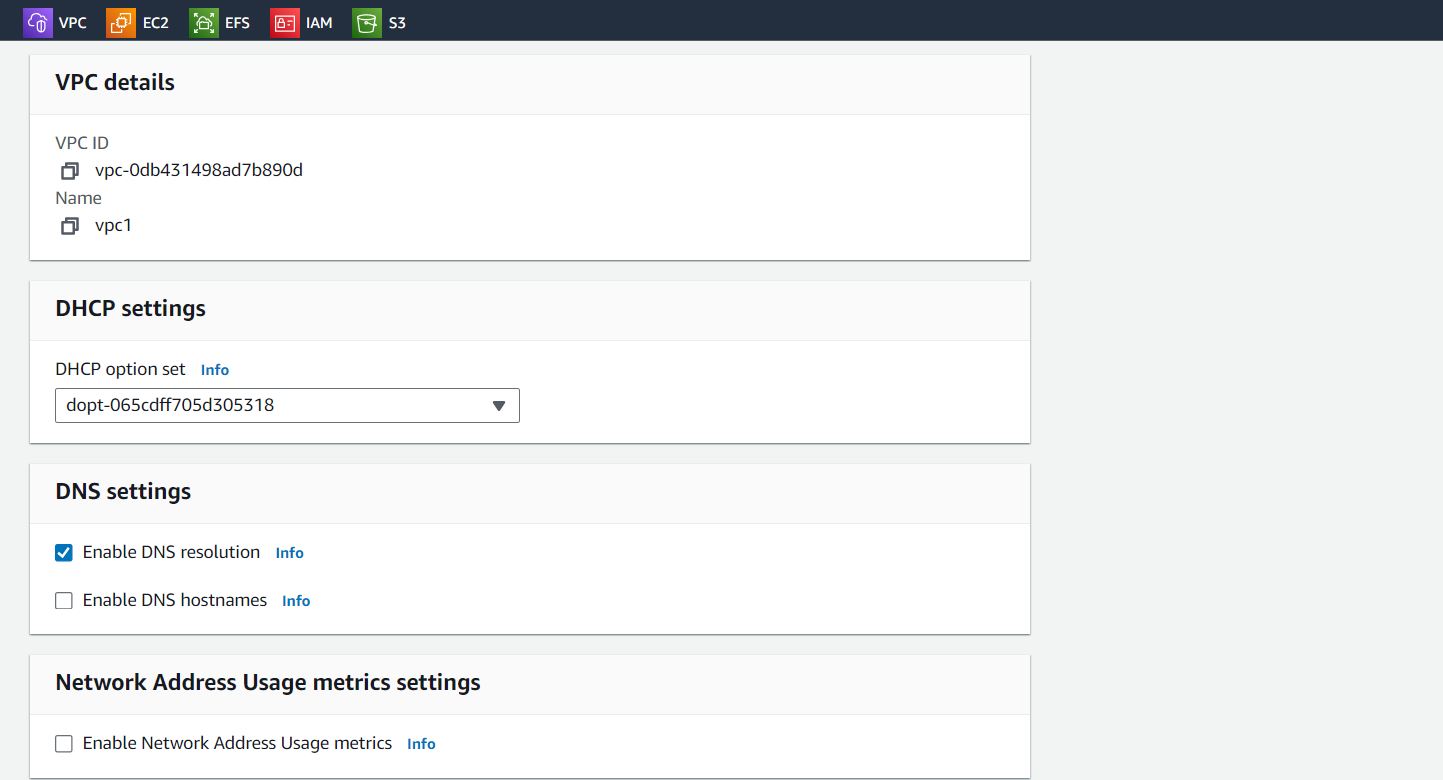


**Troubleshooting**

During the project implementation, a critical oversight occurred after creating the Virtual Private Cloud (VPC) in AWS. I neglected to enable DNS hostnames, which resulted in an inability to mount the Elastic File System (**EFS**) path across the five EC2 instances. This misconfiguration prevented proper communication and access to shared resources.



Upon realizing the error, I promptly enabled the DNS hostname feature within the VPC settings. After this adjustment, I was able to successfully mount the **EFS**, ensuring that all EC2 instances could access the common file system as intended.



**Summary**

This project successfully demonstrates the implementation of a scalable, highly available web application architecture using AWS services, specifically Amazon EC2, Auto Scaling, Application Load Balancer (ALB), and Amazon EFS. Here’s a detailed conclusion summarizing the key aspects and outcomes:

1. **Objective Achieved**

The primary goal of the project was to create an infrastructure that can dynamically handle varying traffic loads while maintaining consistent content delivery across multiple instances. By utilizing Amazon EFS, any changes made to shared files are instantly reflected across all EC2 instances, ensuring a seamless user experience.

2. **Scalability**

The integration of Auto Scaling allows the infrastructure to automatically adjust the number of running EC2 instances based on real-time traffic demands. This capability not only optimizes resource usage but also enhances application performance during peak loads. The architecture can efficiently scale up or down, ensuring cost-effectiveness by only utilizing resources when needed.

3. **High Availability**

With the Application Load Balancer distributing incoming traffic across multiple EC2 instances, the application benefits from improved fault tolerance. If one instance becomes unavailable, the ALB redirects traffic to healthy instances, thereby minimizing downtime and maintaining a reliable service.

4. **Centralized Content Management**

By leveraging Amazon EFS as a centralized file storage solution, the project enables all EC2 instances to access and serve the same content. This eliminates inconsistencies that might arise from individual instance storage and simplifies content updates. Any modification made to files stored in EFS is automatically accessible by all instances, enhancing operational efficiency.

5. **Cost Efficiency**

The use of Auto Scaling and EFS contributes to an overall reduction in operational costs. The project ensures that resources are allocated based on actual demand rather than fixed provisioning, leading to savings on infrastructure costs. Moreover, by using a pay-as-you-go model, the project remains economically viable for varying usage patterns.

**Conclusion**

Overall, this project serves as a solid foundation for deploying robust web applications on AWS. By leveraging the power of cloud services, it illustrates the effectiveness of modern architectures in addressing the needs of scalability, availability, and performance. The project not only fulfils its objectives but also sets the stage for future growth and enhancements in cloud infrastructure management.