TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

In the modern context, the demand for online communication is increasing, especially after the COVID-19 pandemic. Organizations and businesses need effective, secure, and scalable online meeting solutions to meet the growing needs of users. Jitsi Meet, an open-source online meeting platform, stands out for its ability to enable real-time communication via video and audio without the need for proprietary software solutions. However, to ensure optimal performance and availability when there are fluctuations in the number of users, it is necessary to deploy Jitsi Meet in the direction of automatic scaling on cloud infrastructure.

1.2 Reasons for Choosing the Topic

The choice of the topic "Deploying Jitsi Meet with automatic scaling on cloud infrastructure" comes from the urgent need of organizations to maintain continuous and effective communication activities. In particular, in the digital age, the use of cloud technology to automatically adjust resources according to demand is an inevitable trend. This topic not only helps to optimize the use of system resources but also contributes to reducing operating costs and enhancing user experience.

1.3 Significance of the Topic

This topic has high practical significance when applying cloud technology to deploy a powerful, secure and flexible online meeting solution. The successful research and implementation of Jitsi Meet with the ability to automatically scale not only helps businesses improve remote working performance but also provides a deployment model that can be widely applied in educational institutions, healthcare and many other fields. At the same time, applying security measures and optimizing infrastructure will help increase reliability and the ability to protect user data.

CHAPTER 2. THEORETICAL BASIS

2.1 Overview of Jitsi Meet

2.1.1 What is Jitsi-meet?

Jitsi Meet is a collection of **Open-Source** projects which provide state-of-the-art video conferencing capabilities that are secure, easy to use, and easy to self-host.

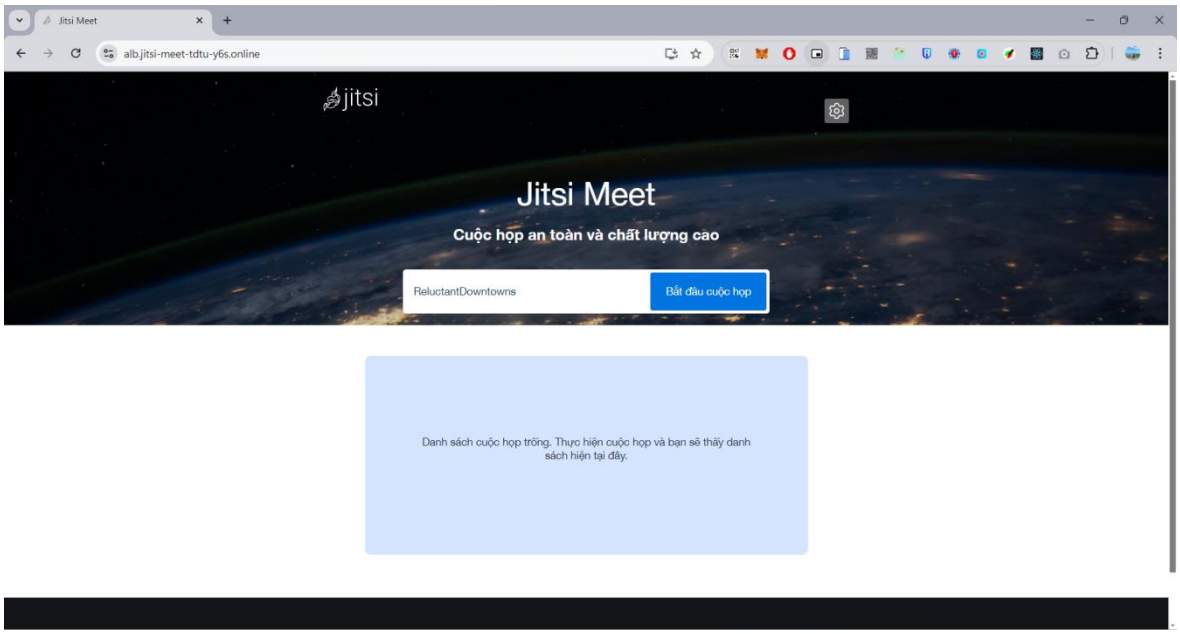

Figure 1. Jitsi Meet website

Jitsi comprises a collection of projects:

- **Jitsi Meet:** WebRTC compatible JavaScript application that uses Jitsi Videobridge to provide high-quality, scalable video conferences. Build upon React and React Native.
- **Jitsi Videobridge (JVB):** WebRTC compatible server designed to route video streams amongst participants in a conference.
- **Jitsi Conference Focus (jicofo):** server-side focus component used in Jitsi Meet conferences that manages media sessions and acts as a load balancer between each of the participants and the videobridge.

- **Jitsi Gateway to SIP (jigasi):** server-side application that allows regular SIP clients to join Jitsi Meet conferences
- **Jitsi Broadcasting Infrastructure (jibri):** set of tools for recording and/or streaming a Jitsi Meet conference that works by launching a Chrome instance rendered in a virtual framebuffer and capturing and encoding the output with ffmpeg.

2.1.2 Requirements Analysis

Based on available numbers (One Core CPU EC2):

- **Two Participants:**
  - CPU Usage: 5%
  - Server Bandwidth: Upload: 30 Kbps, Download: 100 Kbps
- **Three Participants:**
  - CPU Usage: 15%
  - Server Bandwidth: Upload: 5 Mbps, Download: 3 Mbps
- **For 5 Participants:**
  - CPU Usage: around 25%
  - Server Bandwidth: Upload: 10 Mbps, Download: 5 Mbps
- **For 10 Participants:**
  - CPU Usage: likely 50%
  - Server Bandwidth: Upload: more 20 Mbps, Download: aroud 10 Mbps

2.2 Auto-Scalable

2.2.1 Concept of Auto Scaling

Auto Scaling is an important feature in cloud system management, allowing automatic adjustment of system resources based on actual needs. When the number of users increases, Auto Scaling can automatically add resources such as servers, CPUs, or memory to ensure that performance is not reduced. Conversely, when

demand decreases, Auto Scaling will automatically reduce the number of resources to save costs.

2.2.2 The Role of Auto Scaling in Jitsi Meet Deployment

In the context of deploying Jitsi Meet to serve online learning needs, Auto Scaling plays a key role in maintaining stable performance and ensuring a smooth user experience. Using Auto Scaling helps the system to flexibly expand or shrink resources, quickly responding to sudden situations in the number of participants in online meetings, especially during peak periods such as exam season or special events.

2.3 Hostinger

Information on the Custom Domain **alb.jitsi-meet-tdtu-y6s.online** from Hostinger.

To provide a professional and easily accessible interface for users, the custom domain **jitsi-meet-tdtu-y6s.online** was registered and configured through Hostinger.

This domain is used to access the Jitsi Meet platform, and the following steps detail its configuration:

- **Domain Registration:**
    - The domain jitsi-meet-tdtu-y6s.online was registered with Hostinger, a reputable domain registrar known for its user-friendly interface and reliable service.
- **DNS Configuration:**
    - The domain's **DNS** settings were configured to point to the AWS infrastructure.
    - This involved setting up the appropriate DNS records (A records, **CNAME** records) to direct traffic to the AWS Elastic Load Balancer (ELB).
- **Example DNS configuration:**

  - **A Record**: Points the domain to the IP address of the ELB.
  - **CNAME Record**: Used for subdomains to point to the main domain or ELB DNS name.
- **SSL/TLS Certificate:**
  - An SSL/TLS certificate was obtained and installed to ensure secure communication between users and the Jitsi Meet platform.
  - AWS Certificate Manager (ACM) was used to manage and deploy the certificate.
  - This ensures that all data transmitted between the users' browsers and the Jitsi Meet servers is encrypted, providing a secure and trustworthy user experience.
- **Advantages of Using a Custom Domain:**
  - **Professionalism**: A custom domain provides a professional appearance and brand identity, which is essential for an online learning platform.
  - **Ease of Access**: Users can easily remember and access the platform through a simple and recognizable domain name.
- **Security:**
  - With **SSL/TLS** certificates, the custom domain ensures secure communication, which is crucial for maintaining user trust and protecting sensitive data.

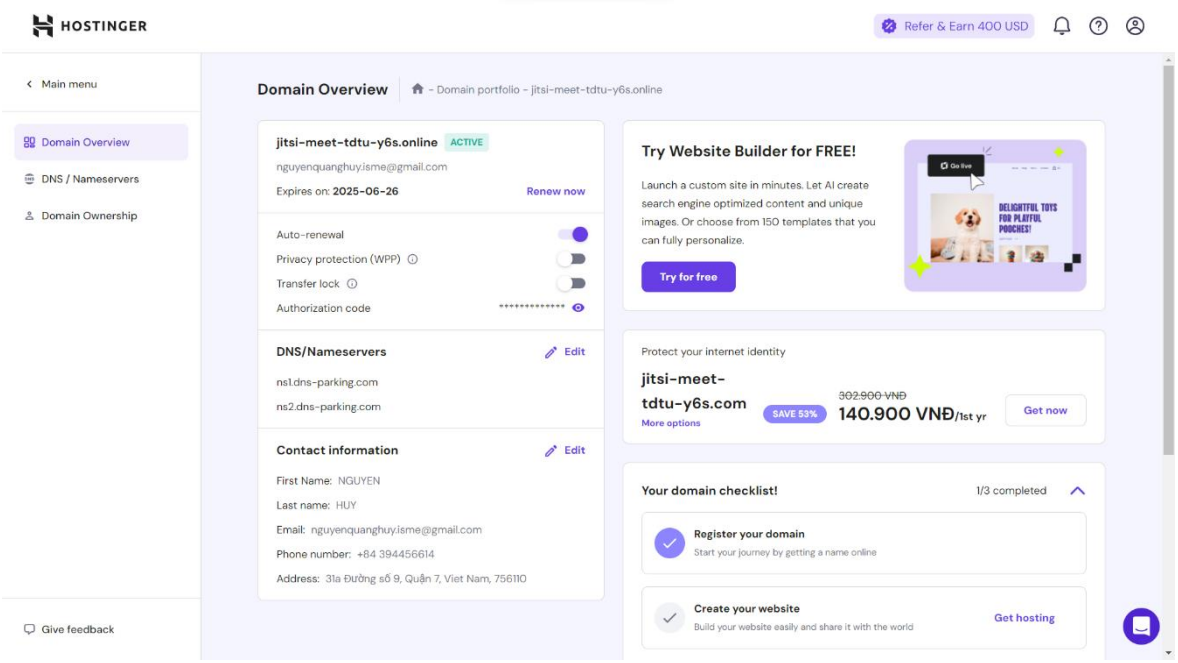

Figure 2. Hostinger domain.

2.4 Amazon Web Services

Amazon Web Services (AWS) was chosen as the cloud provider for deploying the Jitsi Meet solution due to the following reasons:

- **Scalability:**
  - AWS provides robust auto-scaling capabilities that allow the infrastructure to dynamically adjust resources based on demand.
  - This ensures that the Jitsi Meet solution can handle varying loads without manual intervention.
  - Services like EC2 Auto Scaling, Elastic Load Balancing (ELB) or ALB, and AWS Lambda enable the system to scale horizontally and vertically as needed.
- **Global Availability:**
  - AWS has a global network of data centers spread across multiple regions and availability zones.
  - This ensures high availability and low latency for users regardless of their geographic location.

  - Deploying the Jitsi Meet solution across multiple Availability Zones within a region enhances fault tolerance and resilience.
- **Security:**
  - AWS offers a comprehensive suite of security features and compliance certifications, ensuring that the Jitsi Meet deployment is secure.
  - Features such as Virtual Private Cloud (VPC), Identity and Access Management (IAM), Security Groups, and AWS Key Management Service (KMS) help in securing the infrastructure and data.
- **Cost Efficiency:**
  - AWS provides a pay-as-you-go pricing model, which allows for cost optimization by paying only for the resources used.
  - Various pricing models, including On-Demand Instances, Reserved Instances, and Spot Instances, offer flexibility in managing costs based on usage patterns.
- **Extensive Ecosystem:**
  - AWS has a vast ecosystem of services and tools that integrate seamlessly with the Jitsi Meet deployment.
  - Services like AWS CloudWatch for monitoring, AWS CloudFormation for infrastructure as code, and AWS Elastic Beanstalk for application deployment simplify the management and scaling of the solution.
- **Support and Documentation:**
  - AWS provides extensive documentation, tutorials, and a supportive community that helps in resolving issues and optimizing the deployment.
  - Access to AWS Support plans ensures that any technical challenges can be promptly addressed by AWS experts.
- **Domain Configuration**:
  - Information on the Custom Domain alb.jitsi-meet-tdtu-y6s.online from Hostinger

- To provide a professional and easily accessible interface for users, the custom domain **jitsi-meet-tdtu-y6s.online** was registered and configured through Hostinger.

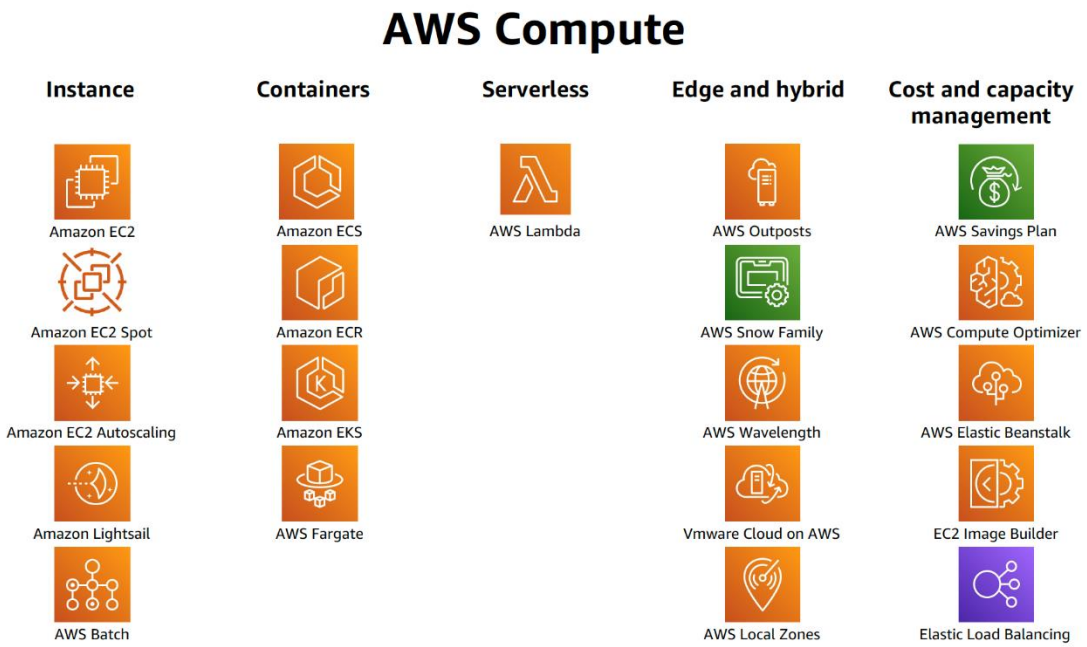

Figure 3. AWS Compute service.

The architecture of a web application using Amazon Web Services (AWS) services:

- **Internet Gateway**: Internet gateway, allowing traffic from the internet to enter VPC (Virtual Private Cloud).
- **Application Load Balancer (ALB):** ALB receives traffic from the internet and delivers to the EC2 instances below based on routing rules.
- **Virtual Private Cloud (VPC):** A separate virtual network in AWS where AWS resources can be deployed.
- **Availability Zones:** AWS divides data centers into different Availability Zones to ensure high availability and fault tolerance. In the picture there are three availability zones (a, b, and c).
- **Subnets**: Subnets in a VPC, each ready to have its own subnet.

- **EC2 Instances:** Virtual servers run applications. These instances are in subnets of different availability zones.
- **Auto Scaling Group (ASG):** A fleet of automatically managed EC2 instances. ASG can automatically adjust the number of instances based on load demand, ensuring that there are enough instances to handle traffic but not too many unnecessary instances.
- **EC2 Security Group:** A set of security rules that control the incoming and outgoing traffic of EC2 instances.
- **Target Group:** A group of destinations (instances) to which the ALB will distribute traffic. These goals are usually within an Auto Scaling Group.

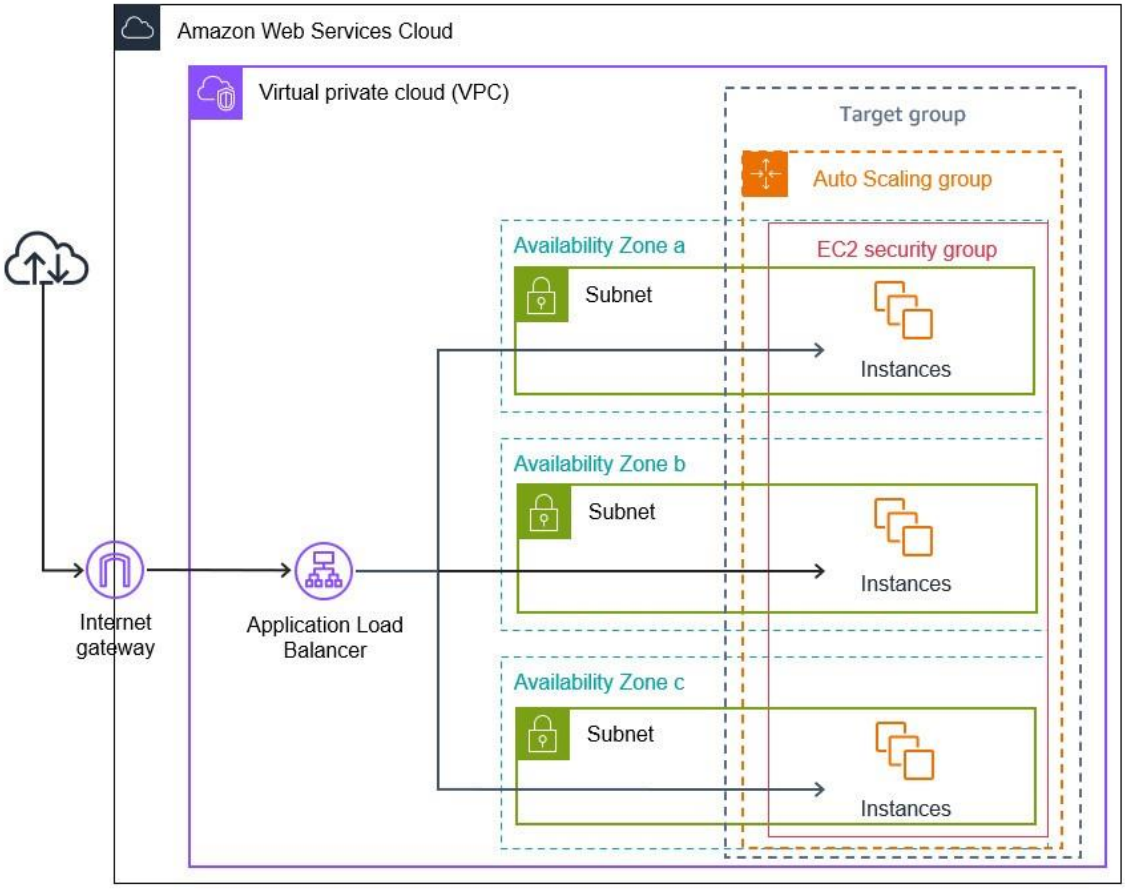

Figure 4. Architecture of a web application using Amazon Web Services

Choosing AWS as the cloud provider for the Jitsi Meet solution offers several advantages, including **scalability**, **global** availability, **security**, **cost** efficiency, and **ease** of access.

These factors contribute to a **robust** and **reliable** online learning platform that can effectively meet user demands and provide a secure and seamless user experience.

CHAPTER 3. SYSTEM DESIGN AND ANALYSIS

3.1 Deploy Jitsi Meet on The Cloud (AWS)

3.1.1 Identity and Access Management (IAM)

Using **Identity and Access Management** (**IAM**) on AWS is extremely important for many reasons related to the **security**, **management**, and **efficiency** of cloud-based resources.

- **Resource Security**:
  - **AWS IAM** helps protect resources on AWS by providing granular access control mechanisms.
  - Root user (Admin) can assign specific access rights to each user or group of users, minimizing the risk of unauthorized access and enhancing security.
- **Centralized identity management**:
  - IAM provides a centralized identity management system that makes it easy to manage and track all users and services accessing AWS resources.
  - This simplifies management and increases control.
- **Granular access control**:
  - IAM allows for the creation of granular access control policies that clearly define what a user or group of users can and cannot do with AWS resources.
  - This helps ensure that access is granted only when necessary and for the right purpose.
- **Compliance and audits**:
  - Using IAM helps organizations comply with security regulations and standards by providing auditing and monitoring tools.
  - All access activities can be tracked and recorded, making it easy to check and analyze security events.

- **Management by role and policy**:
    - IAM allows you to create and assign roles and policies to users or services.
    - This helps manage access more flexibly and efficiently, especially in complex and large environments.
- **Integration with other AWS services**:
    - IAM is tightly integrated with other AWS services, enabling consistent access management across its entire AWS system.
    - For example, IAM can be used to control access to S3, EC2, RDS, and many other services.
- **Supports MFA (Multi-Factor Authentication)**:
    - IAM supports multi-factor authentication, which provides an additional layer of security by requiring users to provide an additional authentication factor in addition to the password.
- **Access management and automation**:
    - IAM enables automation of access granting and revocation based on predefined policies and rules.
    - Helps manage access more efficiently and minimize human error.

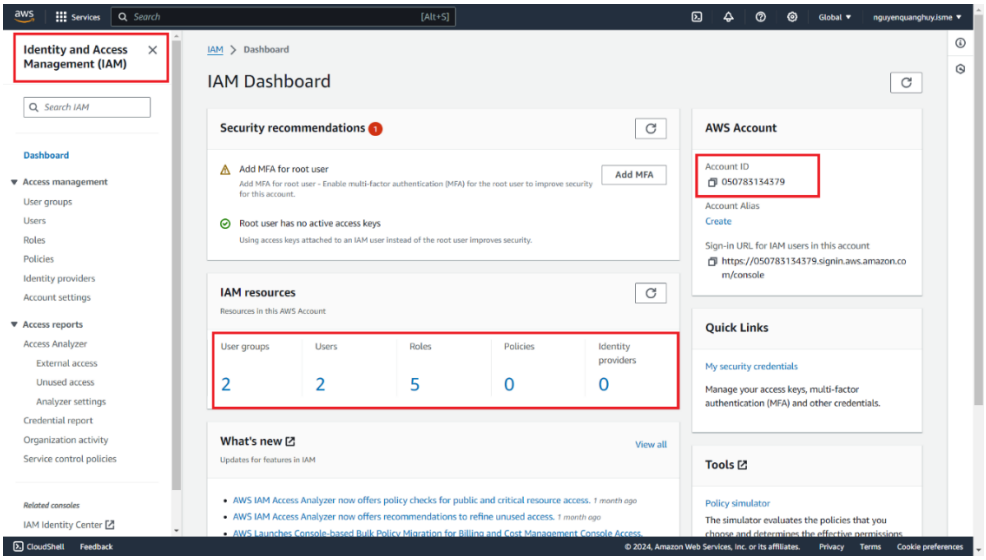

Figure 5. IAM Dashboard

3.1.2 Create Security Groups on AWS

Security Group: The security group acts as a virtual firewall, controlling traffic in and out of EC2 instances, as well as the best security measure to protect Jitsi Meet and user data.

To function, Jitsi Meet needs to communicate with other applications and services over the network, including:

- **WebRTC:** Real-time communication protocol for audio and video transmission.
- **XMPP:** Instant messaging protocol for meeting control.
- **External services:** External services such as STUN, TURN can be used to support the connection.

Why do you need to open specific ports?

Communication: Each port that opens allows for a specific type of communication:

- **UDP 10000, 20000:** Used for WebRTC protocol for audio and video transmission.
- **TCP 443:** Used for HTTPS protocol to access the web interface of Jitsi Meet.
- **TCP 3478, 5349:** Used for STUN and TURN services to support cross-firewall connectivity.
- **Other ports:** Can be used for other add-on services such as SSH for server management.

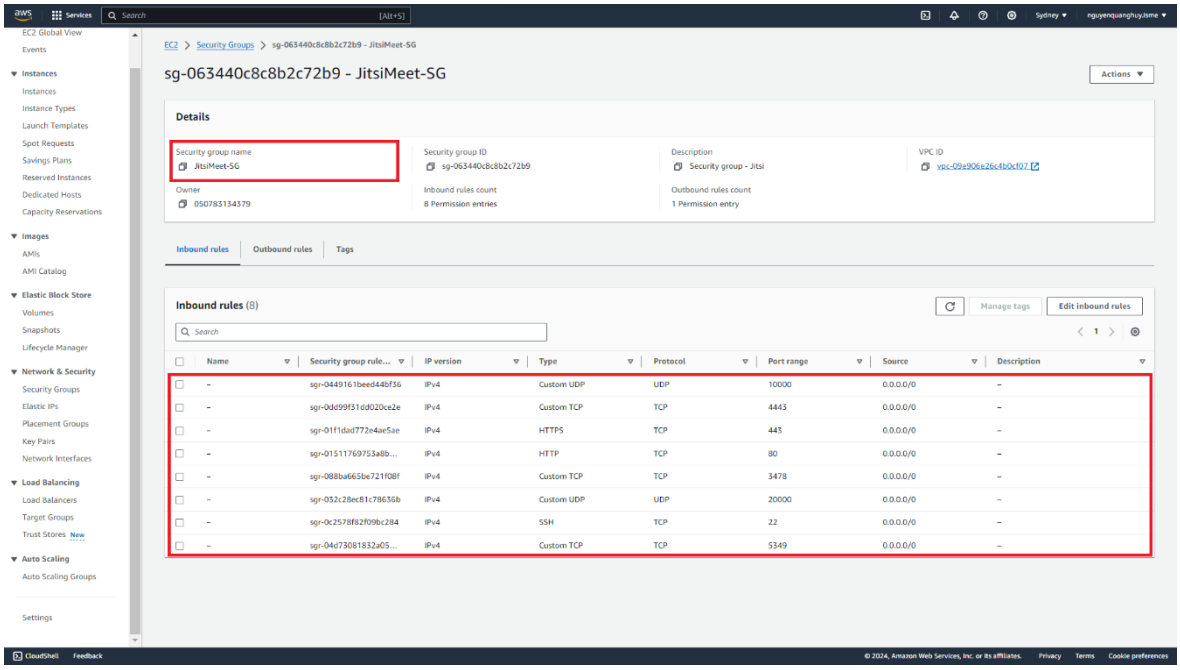

Figure 6. Jitsi Meet Security Groups

3.1.3 *Instance EC2 on AWS*

Create instance EC2 on AWS:

- Instance type: t2.micro.
- CPU: 1 vCPU
- RAM: 1 Gib Memory
- Storage: 1 x 8 Gib gp3
- Application and OS Images: Ubuntu Server 24.04 LTS (HVM)

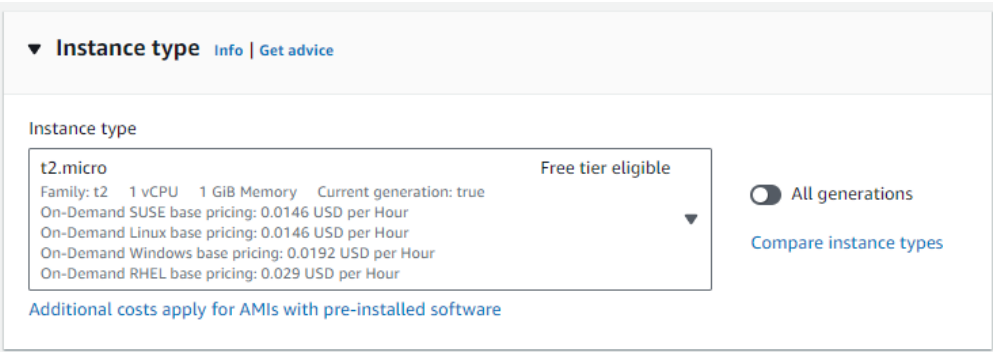

Figure 7. Instance type on AWS.

After selecting a configuration for an EC2 server, you must select Security Groups for that instance to ensure that Jitsi-Meet can be installed and used on the server.

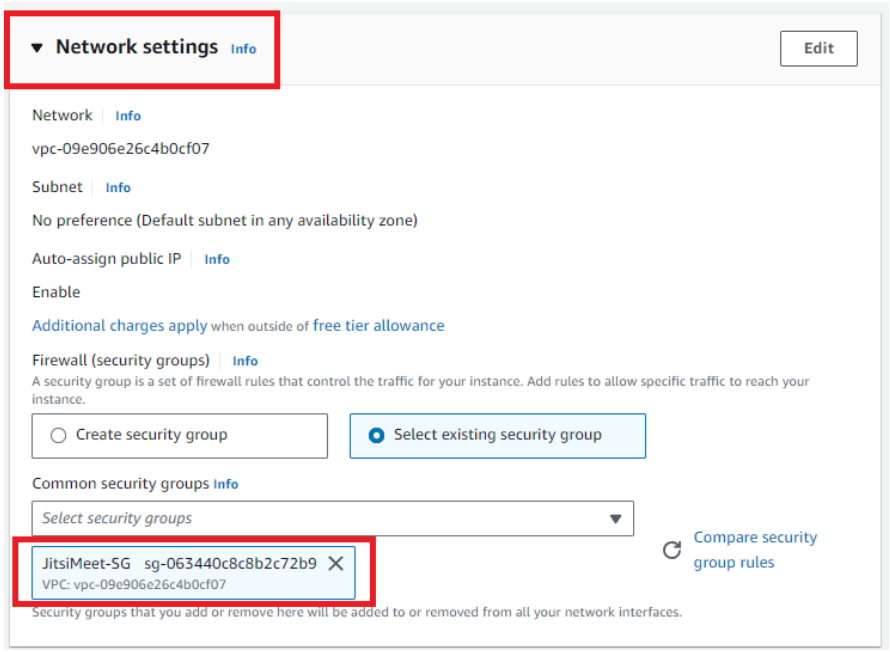

Figure 8. Network settings.

After successfully creating an EC2 server, we need to SSH to the server to install the environment as well as install Jitsi Meet.

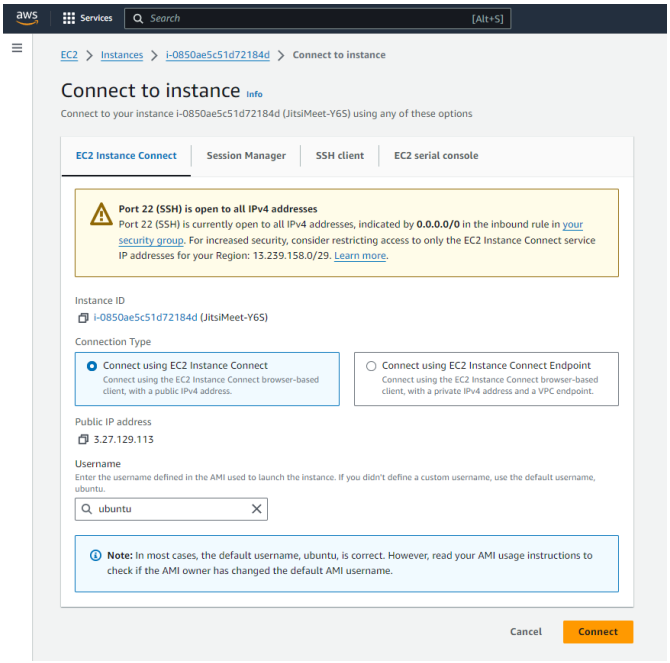

Figure 9. Connect to SSH instance ec2.

3.1.4 Install Jitsi Meet on EC2 instance

Add the Jitsi and the Prosody APT repositories to server. Prosody is an open-source XMPP chat server that Jitsi uses for messaging and admin authentication. Then install the Jitsi Meet package from its repository, which will ensure that always running the latest stable Jitsi Meet package.

First, download the Jitsi GPG key with the curl downloading utility:

```
1  curl https://download.jitsi.org/jitsi-key.gpg.key -o jitsi-key.gpg.key
```

Figure 10. Download the jitsi GPG

Next, add the GPG key:

Figure 11. Add GPG key.

Add the Jitsi repository to server by creating a new APT sources file that contains the Jitsi repository:

Figure 12. Add Jitsi Repository.

Add this line to the newly opened file:

Figure 13. Add newly opened file.

Save and exit the editor.

Next, Add the Prosody package. Download the Prosody GPG key:

Figure 14. Add prosody package.

Then, add the key to server's keyring

```
1  sudo gpg --output /usr/share/keyrings/prosody-keyring.gpg --dearmor prosody-debian-packages.key
```

Figure 15. Add prosody key.

Next, open a new sources file for Prosody

```
1  sudo nano /etc/apt/sources.list.d/prosody.list
```

Figure 16. Add prosody list.

Add the following line to the currently empty Prosody sources file:

```
1  deb [signed-by=/usr/share/keyrings/prosody-keyring.gpg] http://packages.prosody.im/debian jammy main
```

Figure 17. Add Prosody sources file.

Save and exit the editor.

Delete the GPG keys since they are no longer needed:

```
1  rm jitsi-key.gpg.key prosody-debian-packages.key
```

Figure 18. Delete GPG keys.

Finally, perform a system update to collect the package list from the new repositories and then install the jitsi-meet package:

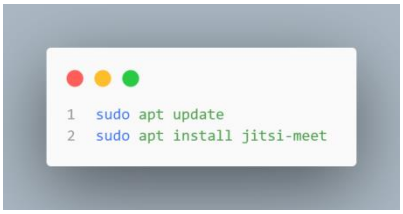

Figure 19. Install Jitsi Meet.

Press Y to confirm the installation of the packages when prompted.

During the jitsi-meet installation, enter the domain name (e.g. jitsi-meet-tdtu-y6s.online) that use for Jitsi Meet instance:

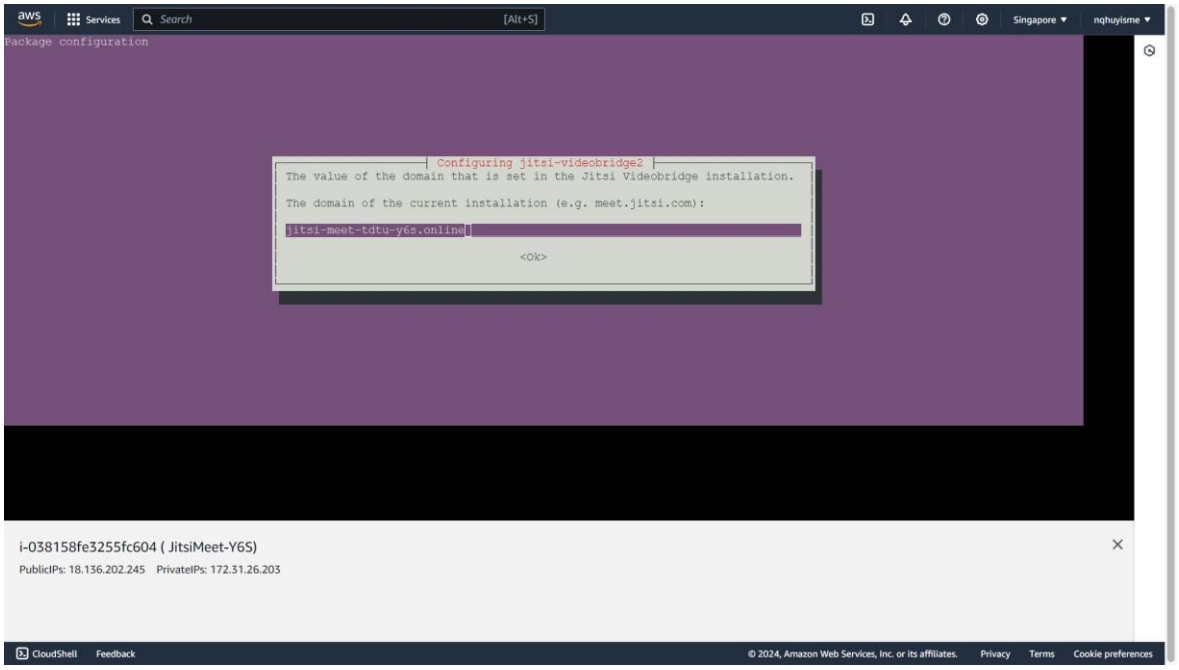

Figure 20. Input domain of Jitsi Meet.

- Use the TAB key to move the cursor from the hostname field to highlight the **<OK>** button. Press ENTER when **<OK>** is highlighted to submit the hostname.
- Select the **Generate a new self-signed certificate** option.

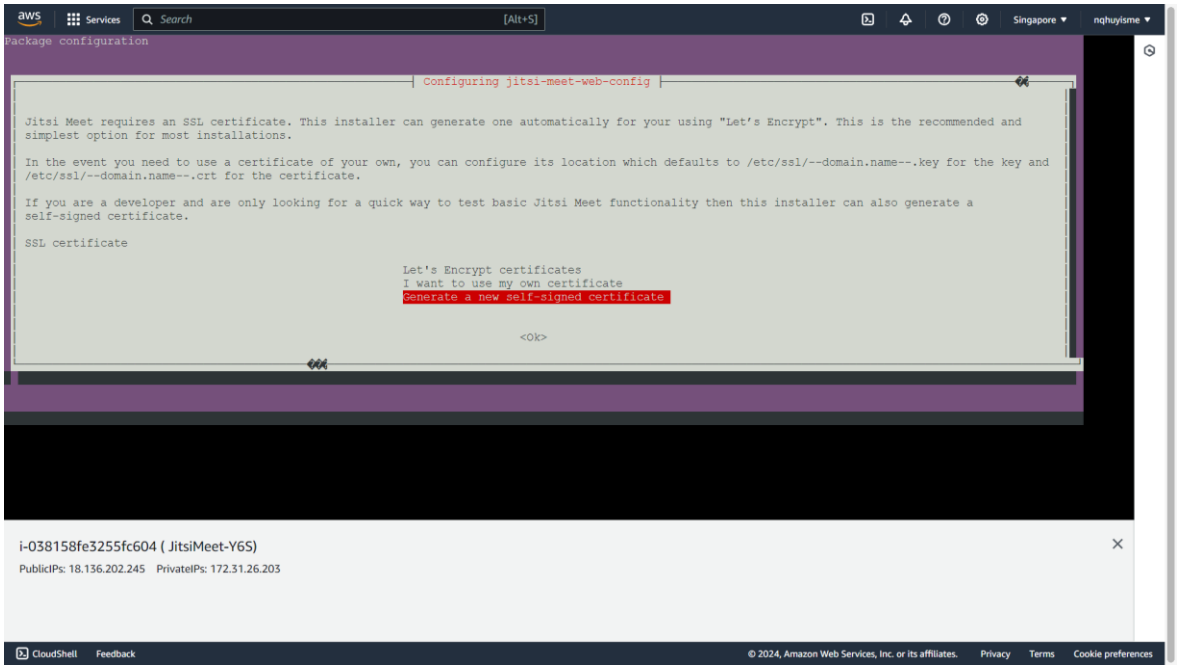

Figure 21. Create SSL Certificate.

3.1.5 Obtaining a Signed TLS Certificate

Jitsi Meet uses TLS certificates to encrypt the call traffic so that no one can eavesdrop on call as it travels over the internet. TLS certificates are the same mechanism used by websites to enable HTTPS-protected pages.

Installing TLS (Transport Layer Security) is very important for the server where Jitsi Meet is installed for the following reasons:

- Data security: TLS encrypts data between the client and the server, preventing outsiders from being able to read or modify the communication between them.
- Server authentication: TLS ensures that the server that the user is connecting to is authentic and not tampered with by third parties.
- Trust and trust: When connecting to a TLS-protected server, users can trust that their information won't be exposed to the network.
- Security regulatory compliance: For many applications, the use of TLS is a mandatory requirement to comply with security regulations and information security policies.

Jitsi uses the certbot utility to obtain and manage free but industry standard and secure TLS certificates. EC instance need install this utility before Jitsi can get the certificate it needs.

- Install certbot with the following command:

Figure 22. Install Cerbot library.

- Press y to confirm the installation of the certbot package when prompted.
- Jitsi Meet supplies a script to download a TLS certificate for **jitsi-meet-tdtu-y6s.online** domain automatically.
- Run this certificate installation script provided by Jitsi Meet at /usr/share/jitsi-meet/scripts/install-letsencrypt-cert.sh with the following command:

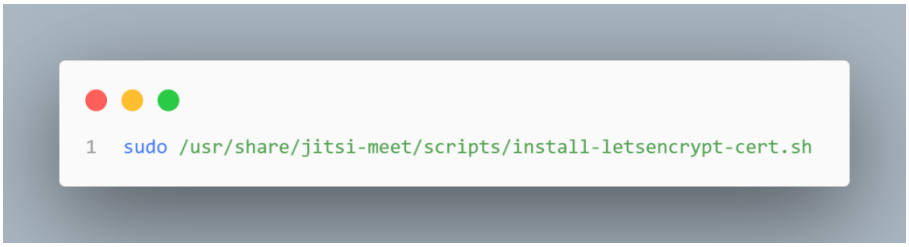

Figure 23. Create SSL certificate.

After install Jitsi Meet on EC2 instance we can click on domain jitsi-meet-tdtu-y6s.online or Public IPv4 address for access Jitsi Meet website.

When accessing with a public IP:

- The browser sends HTTP/HTTPS requests directly to the server's IP address.
- The Jitsi Meet server will process the request and send it back to the website.

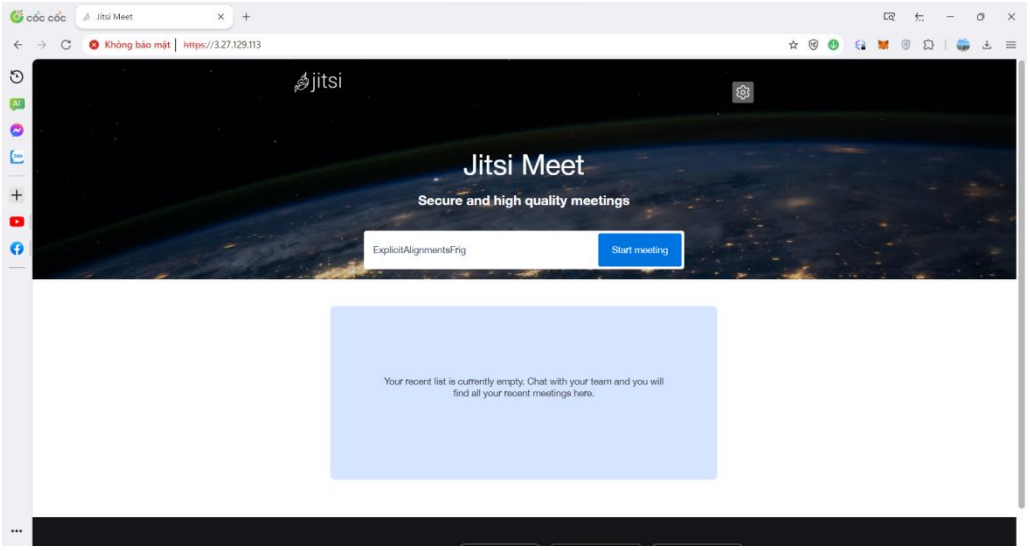

Figure 24. Jitsi Meet with IP address.

When accessing with a domain:

- The browser will send a DNS request to find out the IP address that corresponds to that domain.
- The DNS server will return the IP address of the server running Jitsi Meet.
- The browser will send an HTTP/HTTPS request to this IP address.
- The Jitsi Meet server will process the request and send it back to the website.

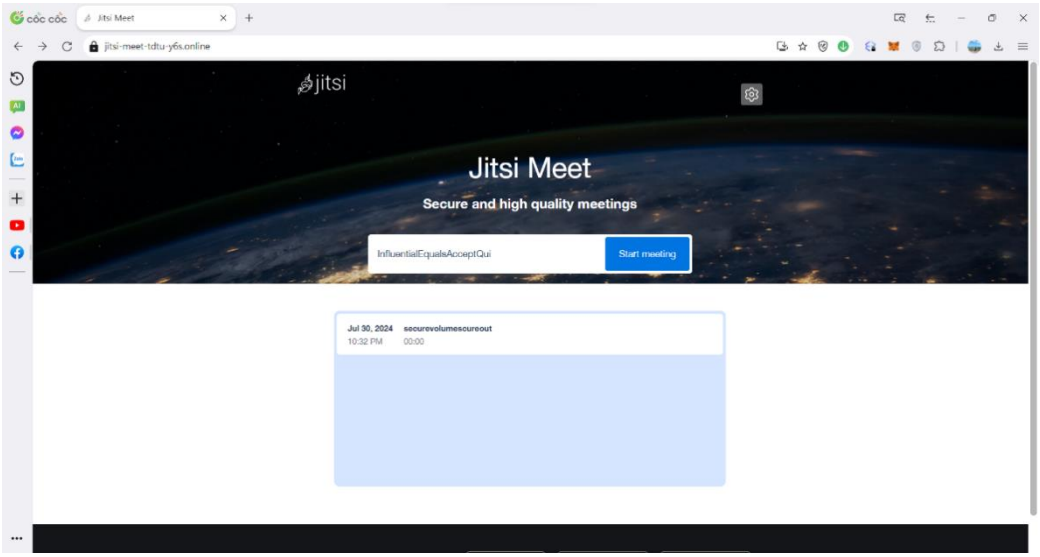

Figure 25. Jitsi Meet with Domain.

It is recommended to use Domain because:

- Easy to remember: Domain addresses are usually easier to remember than IP addresses.
- Professional: Use a domain to make your website look more professional.
- Flexible: If changing the IP address, simply update the A record in DNS again without changing the user's access address.
- Security: Using a domain can help increase security by using features like HTTPS and SSL certificates.

3.2 Application Load Balancers

3.2.1 Application Load Balancer (ALB)

AWS Application Load Balancers (ALBs) are a component of Amazon Web Services' Elastic Load Balancing (ELB) service. ALBs help distribute traffic to your application across multiple destinations such as EC2 instances, containers, and IP addresses, to optimize application availability and load capacity. Here are some key points about ALBs:

- 7 Layer Load Balancing:
    - ALBs operate at layer 7 of the OSI (Application Layer) model, allowing them to handle HTTP and HTTPS requests.
    - ALBs can route traffic based on the content of the request (e.g., URL path, HTTP header information).
- Smart Routing:
    - ALBs provide routing features based on rules.
    - Can route traffic based on a variety of criteria such as URL paths, host headers, HTTP headers, HTTP methods, and more.
- Integration with AWS services:
    - ALBs are deeply integrated with other AWS services such as EC2, ECS, EKS, Lambda, and many others.
    - Easily deploy distributed applications and microservices.
- Security features:

  - ALBs support SSL/TLS termination, which encrypts traffic to and from the load balancer.
  - ALBs also integrates with AWS Certificate Manager (ACM) to easily manage SSL/TLS certificates.
- Health Checks:
  - ALBs perform health checks on the targets behind it, ensuring that traffic is sent only to healthy targets and is ready to process the request.
- Autoscaling:
  - ALBs are capable of automatically scaling to accommodate elevated traffic without user intervention.
- WebSocket và HTTP/2:
  - ALBs support WebSockets and HTTP/2.
  - Helps optimize performance and scalability for modern web applications.

Use Application Load Balancer (ALB) to distribute the load between EC2 instances in a Target Group. Users interact with the system through a browser or application, sending requests to the system through a domain name managed by DNS. The DNS system resolves this domain name to the IP address of the ALB, which helps the ALB receive requests from users and distribute them to EC2 instances in the Target Group. These instances run applications, such as Jitsi Meet, and are responsible for handling requests from ALBs.

This architecture is deployed in a VPC (Virtual Private Cloud) on AWS, providing a secure and isolated environment for resources such as ALB and EC2 instances. ALB helps balance the load between instances, ensuring that the system operates efficiently and has good fault tolerance. Moreover, if configured with Auto Scaling, the system can automatically adjust the number of instances based on actual demand, helping to maintain performance and optimize costs. It is an ideal architecture for systems that require high scalability and availability, like Jitsi Meet.

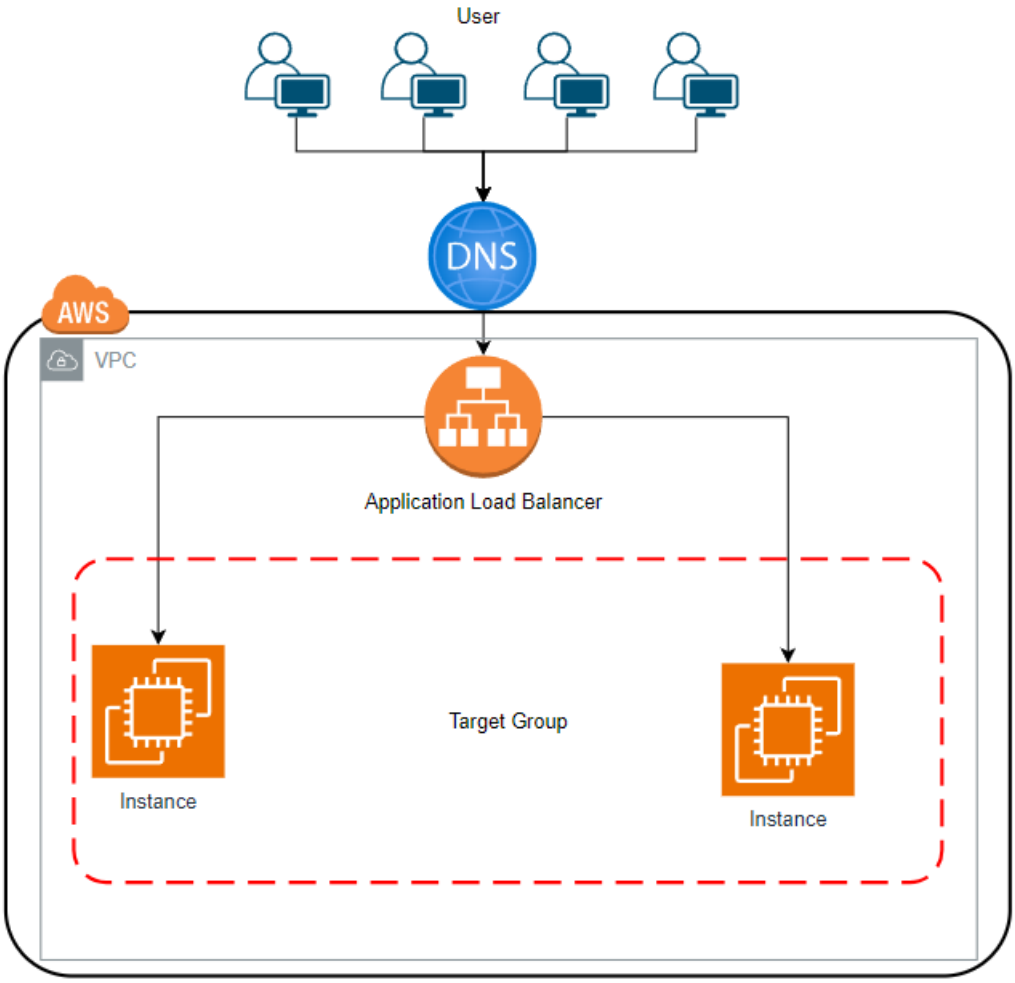

Figure 26. Application AWS architecture.

3.2.2 Target Groups

Target Groups are a critical element of AWS's architecture, especially when working with services such as Application Load Balancer (ALB) and Network Load Balancer (NLB), which play a critical role in distributing traffic to the system's EC2 instances.

Target Group: A Target Group is a group of targets that a load balancer can send traffic to.

Possible objectives:

- **Instance EC2**: these are EC2 instances running the Jitsi Meet application.
- **IP**: Static IP addresses can be added to the target group.

- **Lambda function**: In some cases, Lambda functions can be added to **the target group** to perform short-term processing tasks.

Relationship between Target Group and EC2 Instance (Jitsi Meet instance):

- Target Group contains EC2 instances: The basic relationship is that the Target Group contains the EC2 instances that the load balancer distributes traffic to.
- Traffic distribution: When a request reaches a load balancer, the load balancer selects a random instance from the corresponding target group to process that request.
- Load balancing: Target Groups help distribute traffic evenly among instances in the group, helping to increase the fault tolerance and performance of the application.
- Flexible management: Easily add, remove, or change instances in a target group without changing the load balancer's configuration.

Example:

Target Group Name: Jitsi-Meet-Y6S-443

- The app is Jitsi Meet and it listens on port 443 (HTTPS).
- Protocol và Port:
    - Protocol: HTTP
    - Port: 443
    - This means that this target group only receives HTTP requests on port 443 (HTTPS).
    - It is the standard gateway for encrypted data transmission over the internet.
- Target type: Instances The targets in this target group are EC2 instances.
- IP address type: Load balancer
    - Instances in the target group will be accessed through the load balancer's IP address.
- Total targets: 2

- Currently, there are 2 EC2 instances registered in this target group.

- Healthy: 2
  - Both instances are currently in a healthy state, which means they are ready to receive traffic.
- Unhealthy: 0
  - No instances are in an unhealthy state.
- Draining: 0
  - No instances are in the process of escaping the target group.
- Instance ID, Name, Port, Zone, Health status:
  - Lists the details of each instance in the target group, including ID, name, port, availability zone, and health status.

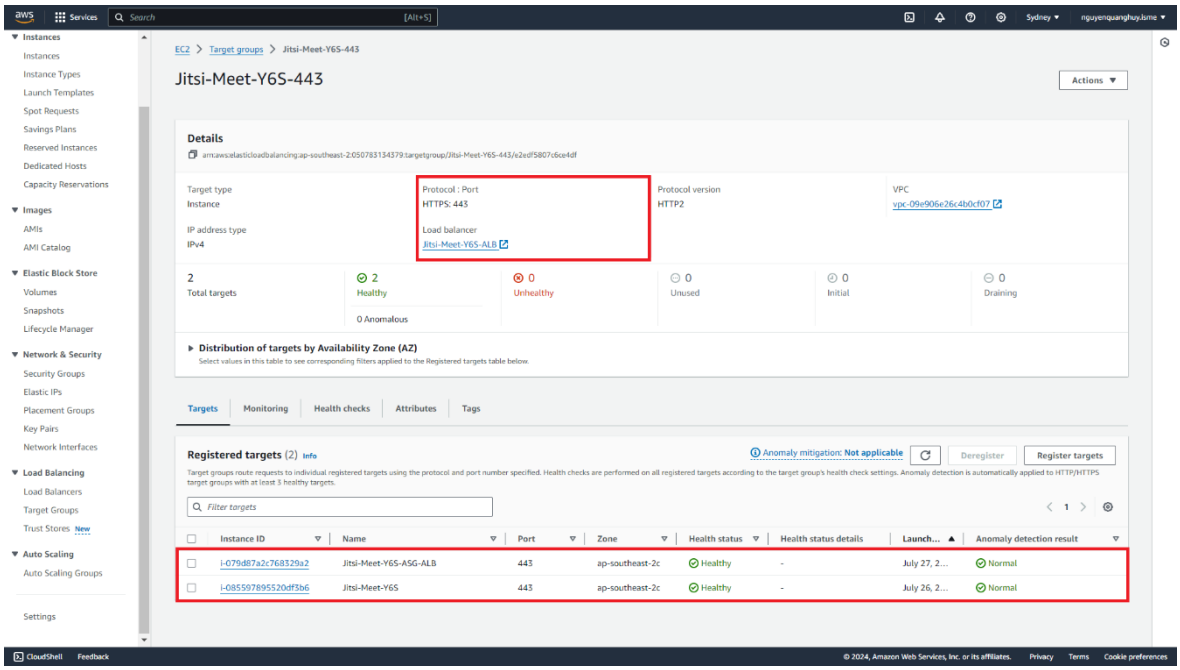

Figure 27. Jitsi Meet Target Groups.

3.2.3 ALB and Target Group

- Configuring Routing:
  - When configuring ALB, define the routing rules in the Listener.

- These rules define how the ALB handles incoming requests and routes them to specific Target Groups based on criteria such as URL paths, host headers, HTTP headers, and HTTP methods.
- An ALB Listener can have multiple routing rules, and each rule can point to a different Target Group.

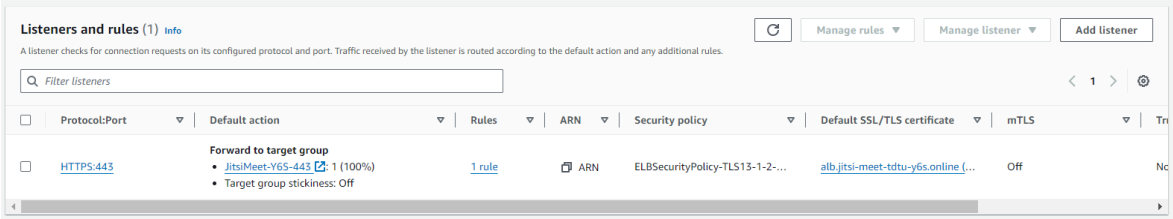

Figure 28. Listeners and rules of Target groups.

- Load Distribution:
  - ALB receives requests from users, and based on routing rules, it forwards these requests to the respective Target Group.
  - In a Target Group, requests will be evenly distributed to targets in the group based on the Round Robin algorithm or other algorithms if configured.

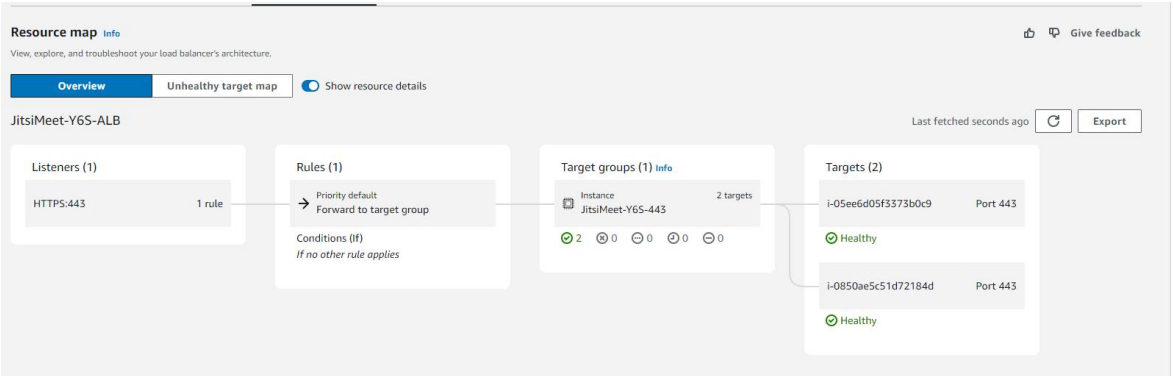

Figure 29. Resource map

- Health Checks:
  - Target Groups have health checks to ensure that the targets in the group are active and ready to handle requests. If a target fails the health check, the ALB will not send traffic to that target until it recovers.

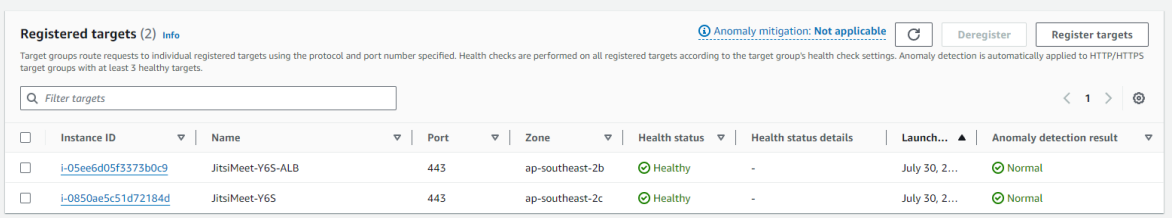

Figure 30. Registered target.

- Automatic expansion and load balancing:
    - As targets in the Target Group change (for example, when EC2 instances are added or removed), the ALB automatically updates to reflect these changes and continues to distribute traffic efficiently.
    - ALB is capable of auto-scaling to handle elevated traffic without user intervention.
- Flexibility:
    - The same Target Group can be used for multiple ALBs or multiple routing rules within the same ALB.
    - This provides high flexibility in configuring and managing the application.

Port 443 (HTTPS):

The use of port 443 ensures that the communication between the client and the server is encrypted using HTTPS, ensuring the security of the transmitted data.

This will help the Jitsi Meet system on the EC2 server:

- **Load Balancing**: Distribute traffic evenly to increase performance and fault tolerance.
- **Flexible management**: Easily add, delete, or change instances in a target group.
- **Integration with other services**: Target Groups can be used with many other AWS services such as Auto Scaling Groups, Application Load Balancing.

3.2.4 Application Load Balancer (ALB) and record CNAME of Domain

CNAME and its role in DNS

- CNAME (Canonical Name):
  - CNAME is a type of record in DNS that allows you to create an alias for another domain.
  - This means that when a user visits a domain with a CNAME, they will be redirected to the destination domain that the CNAME is pointing to.
- DNS (Domain Name System):
  - It is a system that translates domain names into IP addresses.
  - Make it easier to access websites by using domain names instead of IP addresses.

Why use **a CNAME record of DNS Domain** for **Application Load Balancers?**:

- Domain separation:
  - The use of CNAME separates Jitsi Meet's domain from ALB's constantly changing IP address.
  - This makes it easier to manage and maintain the system.
- Load balancing:
  - When creating a CNAME pointing to the ALB, all requests to the domain are redirected to the ALB.
  - ALB will automatically distribute traffic to EC2 instances in the target group, ensuring that no instances are overloaded.
- Flexibility:
  - If want to change the IP address of your ALB, you only need to update your CNAME record once.
  - All clients will be automatically redirected to the new address without having to change any other configuration.
- Confidentiality:

- The use of CNAME helps mask the actual IP addresses of EC2 instances, enhancing system security.

Example:

A Jitsi Meet application runs on multiple EC2 instances, and we want users to access it through the **alb.jitsi-meet-tdtu-y6s.online domain**.

We will take the following steps:

- Create an Application Load Balancer: Create an ALB and configure it to distribute traffic to EC2 instances running Jitsi Meet.

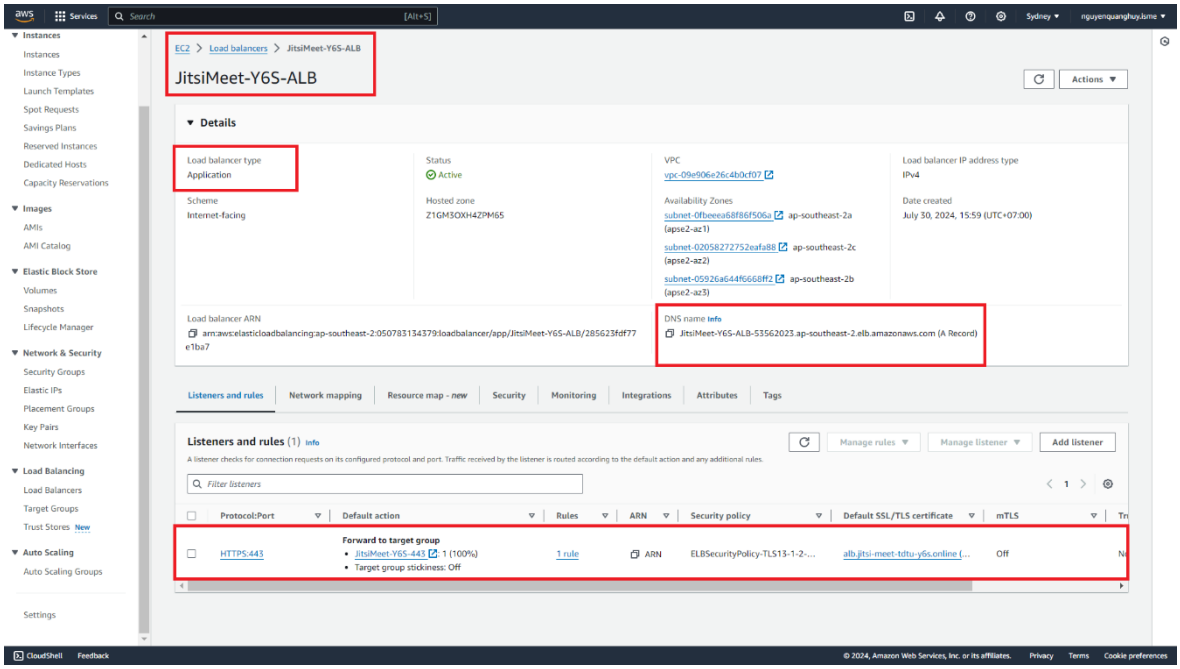

Figure 31. JitsiMeet-Y6S-ALB Load balancers.

- Create a Target Group: Add EC2 instances to this target group.

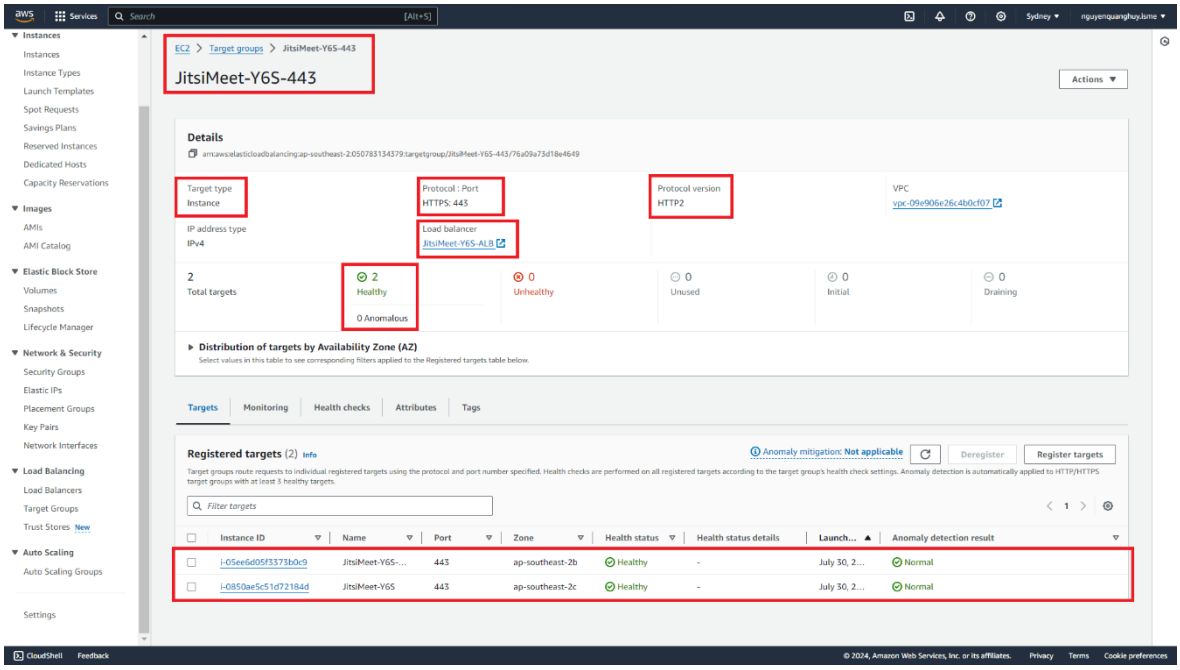

Figure 32. JitsiMeet-Y6S-443 Target groups.

- Create a CNAME record: Create a CNAME record for **alb.jitsi-meet-tdtu-y6s.online** pointing to the ALB's DNS address.

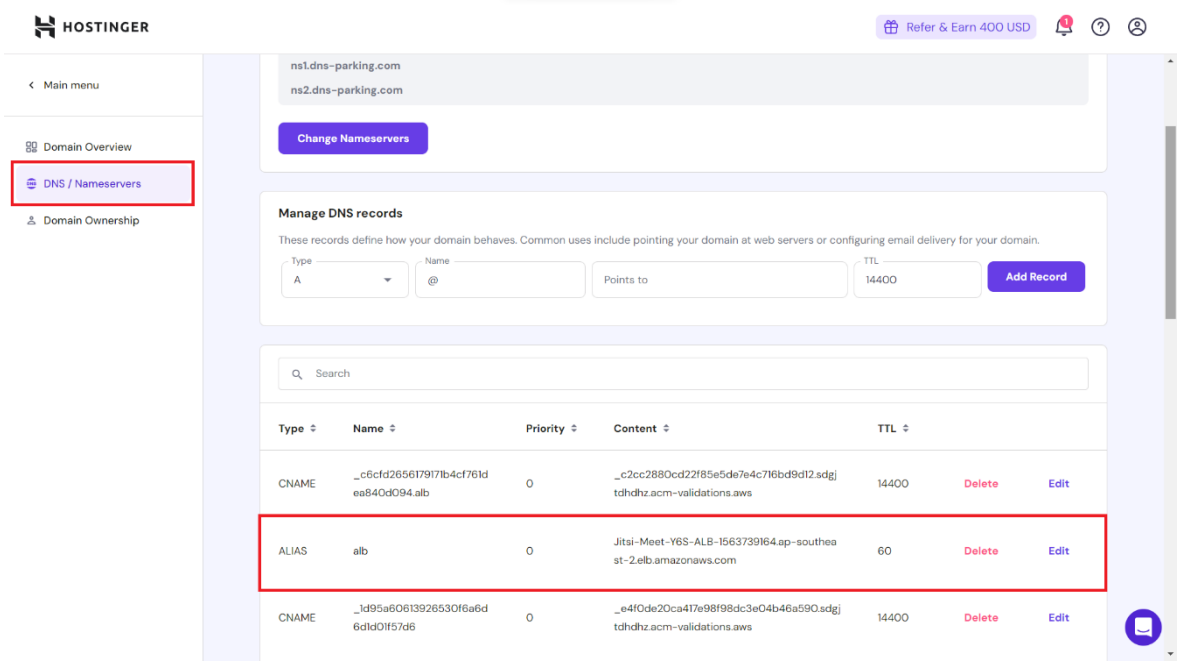

Figure 33. Domain for DNS configration.

- When the user enters **alb.jitsi-meet-tdtu-y6s.online** into the browser, the request is sent to the DNS server, which finds the CNAME record and redirects the request to the ALB.
- ALB will select a suitable EC2 instance from the target group to handle this request.

3.2.5 Resource Map – ALB

Resource map is an AWS feature used to visualize and manage AWS resources related to a specific Application Load Balancer (in this case, JitsiMeet-Y6S-ALB).

Detailed analysis of each component of ALB in the form of a Resource Map:

- **Listeners**: List the Listeners and their respective protocols and ports.
- **Rules**: Describe the routing rules and conditions for matching a Rule.
- **Target Groups**: Lists the Target Groups and the instances or Lambda functions that belong to each group.
- **Targets**: List the Targets and their health status (in this case, the EC2 servers that have Jitsi Meet installed).

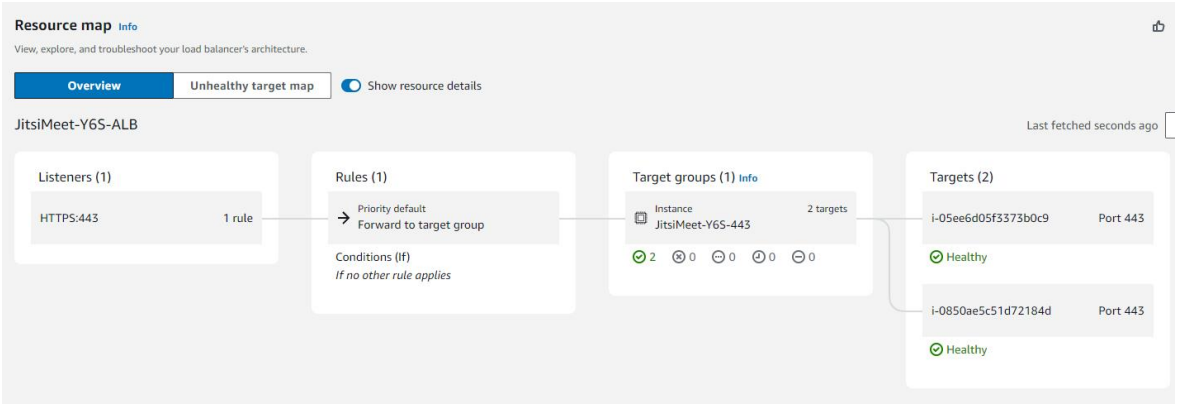

Figure 34. Resource map of Load balancers

3.2.6 Using AWS Certificate Manager (ACM) Certificates with Jitsi Meet on EC2 and ALB

AWS Certificate Manager (ACM): This certificate encrypts data in transit between users and servers, securing connections.

Certificate Details

- **Certificate status**:
    - **Identifier**: Certificate Unique Identifier (bf5041c2-605d-4a84-8ecc-bae45ddb1f7d).
    - **ARN**: The Amazon Resource Name of the certificate, uniquely identifies the certificate in the AWS system.
    - **Type**: Certificate type, here "Amazon Issued" i.e. issued by Amazon.
    - **Status**: The status of the current certificate is "Issued".
- **Domains**:
    - **Domain**: The domain name associated with the certificate, here alb.jitsi-meet-tdtu-y6s.online.
    - **Status**: The status associated with the domain name, here "Success" (success).
    - **Type**: The type of DNS record used for authentication, here is "CNAME".
    - **CNAME name**: The CNAME name used to validate the domain name.
    - **CNAME value**: The CNAME value must be generated in the DNS configuration to validate the certificate.
- **Details**:
    - **In use**: This certificate is currently in use.
    - **Serial number**: Serial number of certificate.
    - **Domain name**: Domain name associated with certificate.
    - **Public key info**: Public Key Information (RSA 2048).
    - **Signature algorithm**: Signing algorithm (SHA-256 with RSA).
    - **Requested at**: When to request a certificate.
    - **Issued at**: When the certificate is issued.
    - **Not before**: The start time of the certificate is valid.

- **Not after**: Certificate expiration time.

- **Renewal eligibility**: gCertificate expiration timeia eligibility (Eligible).

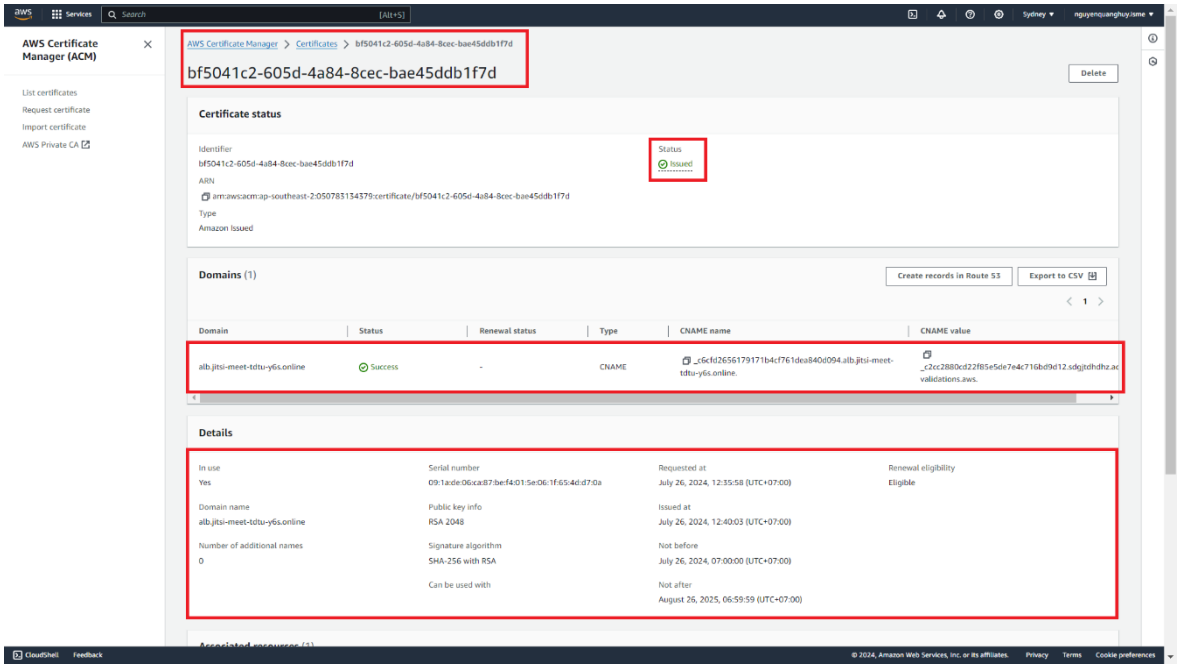

Figure 35. AWS Certificate Manager.

SSL/TLS certificates from ACM ensure that the connection between users and the Jitsi Meet service on AWS is encrypted and secure.

An ALB configuration is required to use this certificate, and make sure the proper DNS configuration is required to validate **alb.jitsi-meet-tdtu-y6s.online domain name.**

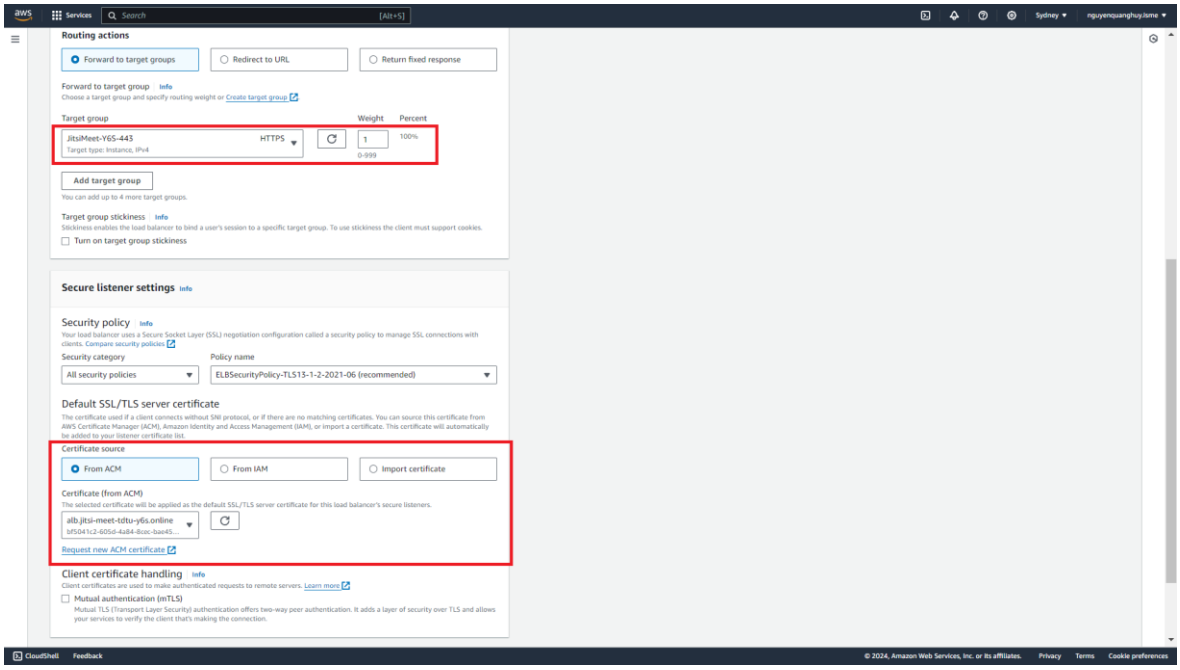

Figure 36. How to use AWS certificate.

3.2.7 Configuring Nginx for DNS Application Load Balancers.

Configure Nginx to Forward Traffic from ALB to Jitsi Meet Servers in Auto Scaling Groups.

Figure 37. Access Nginx.

Install Jitsi Meet and Nginx in Launch Configuration/Template (before creating AMIs, you must configure these information so that when you initialize the Launch Template, the config will be available).

- **"listen 80":** This configuration for Nginx listens for HTTP requests on port 80 (the default port for HTTP).

- **“listen [::]:80”:** This is the configuration for Nginx to listen for HTTP requests on port 80 for IPv6. The :: is the address for all IPv6 network interfaces.
- **“server_name alb.jitsi-meet-tdtu-y6s.online”:** This configuration assigns a domain name to this server. In this case, the domain name is alb.jitsi-meet-tdtu-y6s.online. Nginx compares the domain name of the incoming request to this value to determine whether the request should be processed.
- **“listen 443 ssl http2”**: This configuration allows Nginx to listen for HTTPS requests on port 443 (the default port for HTTPS) and use the HTTP/2 protocol.
- **“listen [::]:443 ssl http2”**: This is the configuration for Nginx to listen for HTTPS requests on port 443 for IPv6 and use the HTTP/2 protocol.
- **“server_name alb.jitsi-meet-tdtu-y6s.online”**: Like the previous server block, this configuration specifies the domain name for this server as alb.jitsi-meet-tdtu-y6s.online.

Figure 38. Configuration Nginx file.

3.3 Auto Scaling Group

3.3.1 Auto Scaling Groups: Automated Scale Management.

- **Auto Scaling Groups:**
  - ASG allows you to automatically adjust the number of EC2 instances running based on the policies that you set.
  - For example, based on CPU utilization, number of connections, or on a schedule.
- **Using AMIs:** ASGs will use the created AMIs to initialize new instances as needed.
- **Ensure Availability:** ASG helps ensure that there are always enough resources to serve users, even during load spikes.

Benefits of Auto Scaling Groups:

- Cost Optimization: Automatically adjust the number of instances to run only the number needed, helping to minimize costs..
- Increase Reliability: Automatically replace instances that are not performing well, ensuring that the application is always available.
- Increase Load Capacity: Automatically scale the number of instances up or down based on the load of the system, ensuring good performance even when traffic spikes.
- Flexible Integration: Easily integrate with other AWS services such as ELB, CloudWatch, SNS, and more.

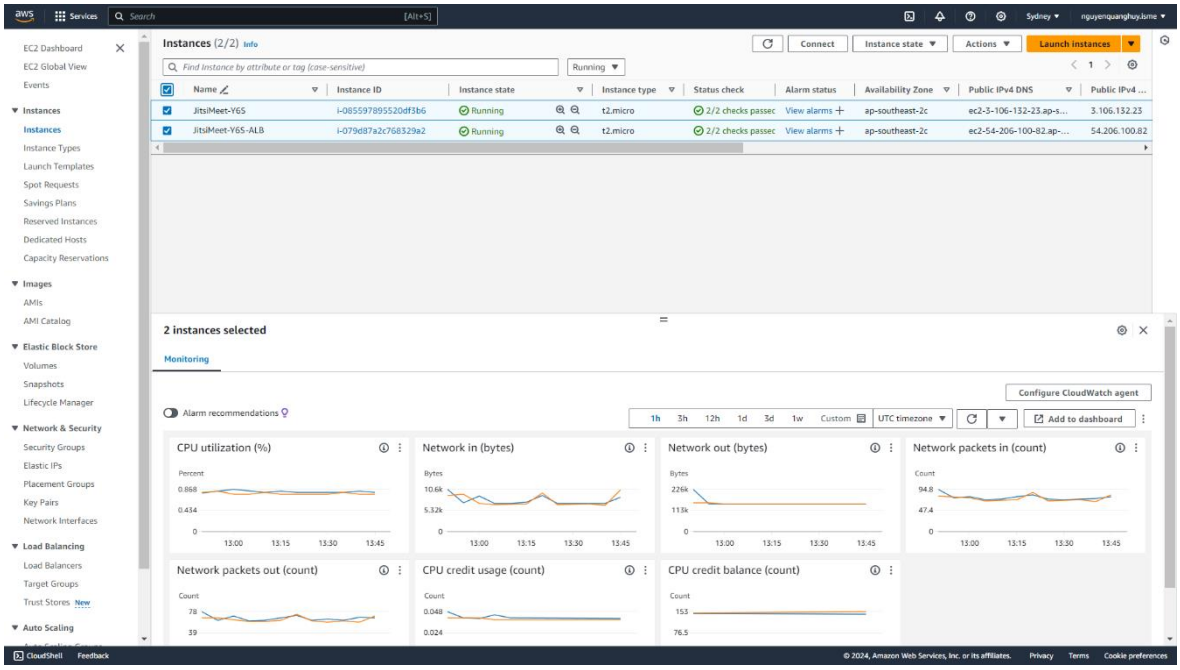

Figure 39. Instance EC2.

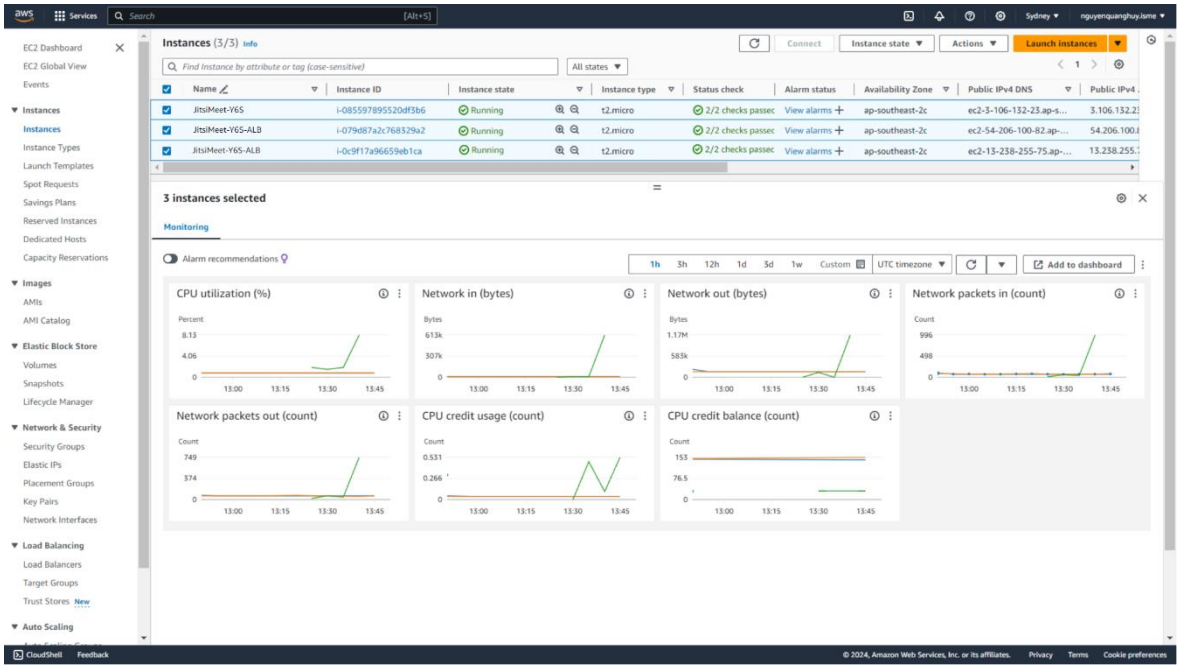

Figure 40. Instance EC2 with Auto Scaling Groups.

3.3.2 Key Components of Auto Scaling Group.

- Launch Configuration/Launch Template:

    - Contains instance configuration information such as instance type, AMI, key pair, security groups, user data, etc..
- Auto Scaling Group:
    - Group Size: Defines the min, max, and desired capacity of instances.
    - VPC and Subnets: Defines the VPC and subnets where the instances will be initialized.
- Load Balancer:
    - ASG can be integrated with Elastic Load Balancing (ELB) to automatically distribute traffic between instances in a fleet.
- Health Check:
    - Configure ASG to check the health of instances, which can be based on EC2 status checks or ELB health checks.
- Scaling Policies:
    - Dynamic scaling: Based on monitoring metrics from CloudWatch, policies can be defined to automatically increase or decrease the number of instances.
    - Scheduled Scaling: Define scaling actions at fixed times.
- Notifications:
    - ASG can be configured to send notifications via SNS (Simple Notification Service) when an event occurs, such as when a new instance is initiated or terminated.

3.3.3 Create AMI from Instance Jitsi Meet was config ALB.

Creating an Amazon Machine Image (AMI) from an EC2 instance running Jitsi Meet and **Auto Scaling Groups (ASGs)** is a closely related component of managing and scaling Jitsi Meet applications on AWS.

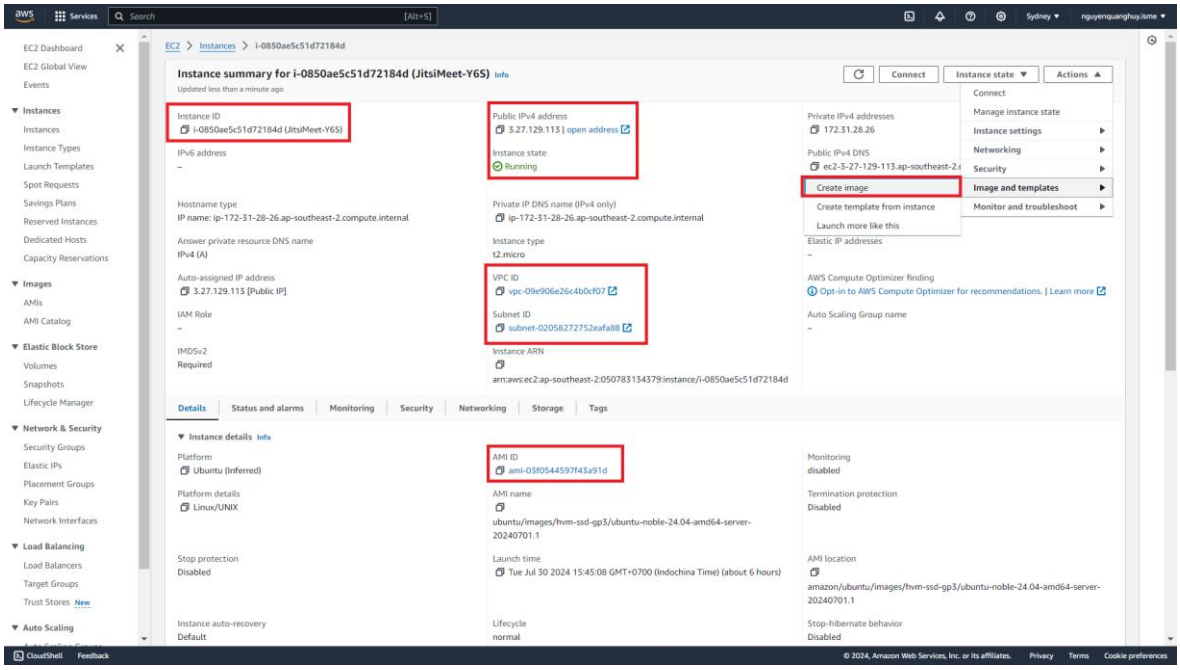

Figure 41. Create Image from EC2.

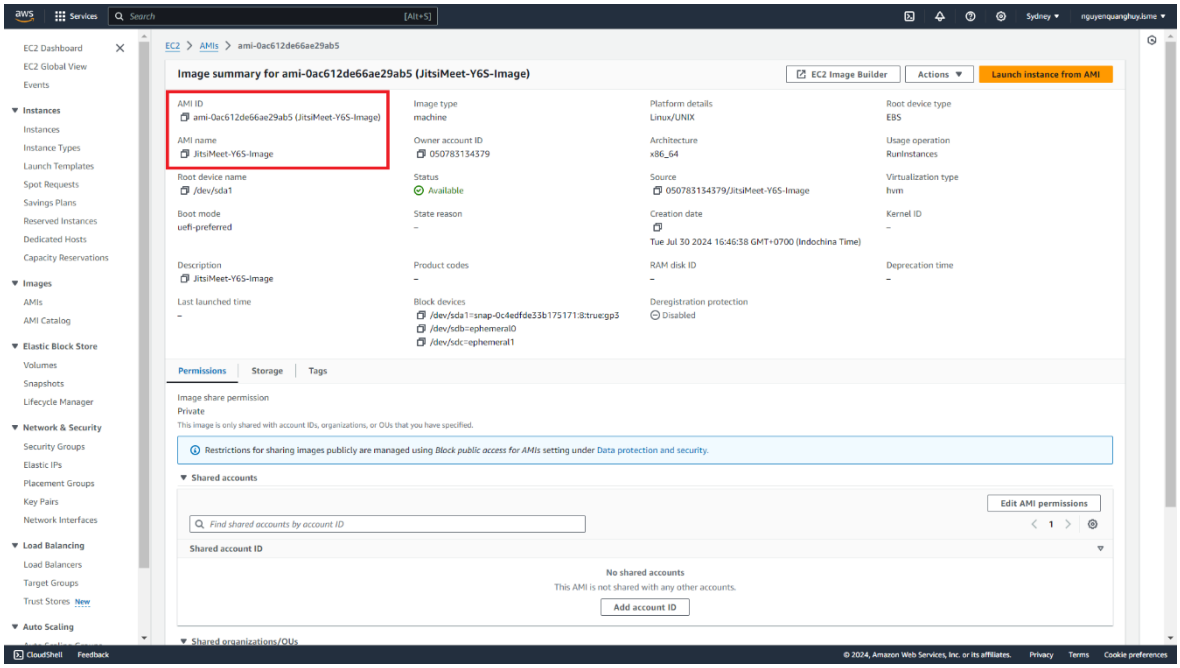

Figure 42. Detail of AIM EC2.

3.3.4 Create AMI: Grounds for expansion

- **Configuration Copy:**

- An AMI is a snapshot of the entire configuration of an EC2 instance at a given time.
- Includes operating system, application (Jitsi Meet), data, and other settings including Application Load Balancers.

- **Basis for Auto Scaling Groups:**
  - The AMI serves as a template for automatically creating multiple new EC2 instances.
  - As demand increases, ASG will automatically initialize new instances from this AMI to meet the load through Application Load Balancers.
- **Ensure uniformity:**
  - All instances created from the same AMI will have the same configuration, helping to ensure the consistency and stability of the system.

3.3.5 Launch Template

Launch Templates and **Auto Scaling Groups (ASGs)** are two important factors in managing and scaling EC2 instances on AWS.

They work together to provide a flexible and efficient way to adjust the number of instances based on the needs of the application.

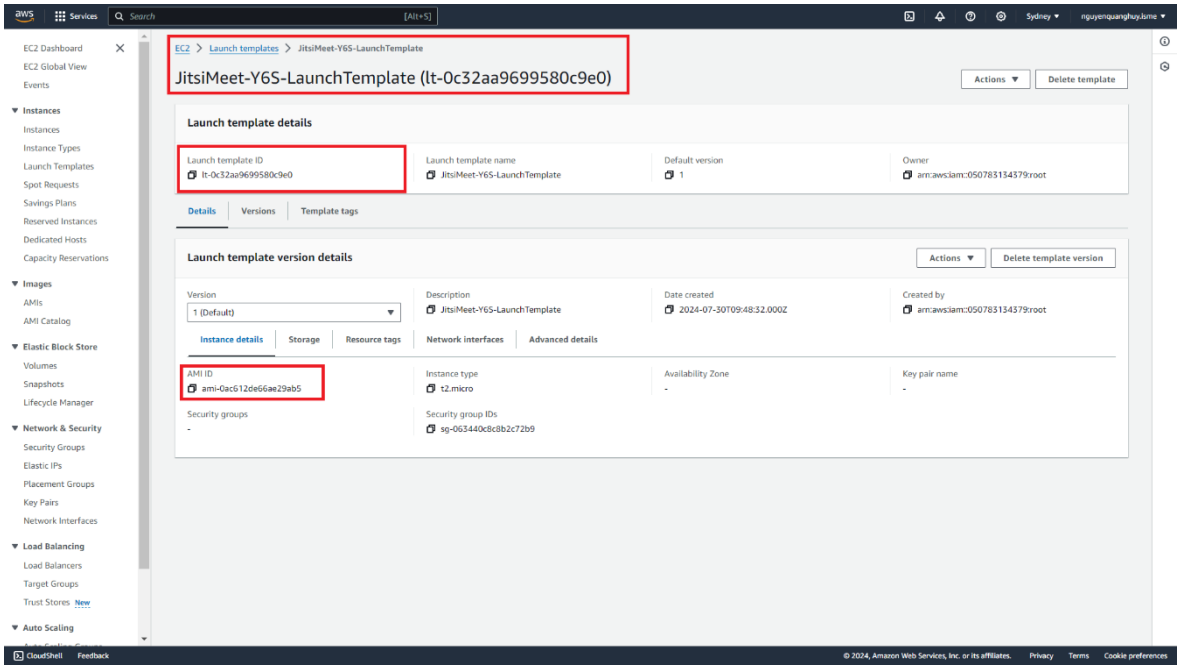

Figure 43. Launch Template.

Launch Template: The Launch Template is a detailed configuration template for an EC2 instance.

It includes all the information needed to start an instance, such as:

- AMI Image: The AMI Image (Amazon Machine Image) used to create the instance.
- Instance Type: Instance Type (t2.micro, m5.large,...)
- Volume size: The size of the EBS volumes attached to the instance.
- Security group: The security group that allows access to the instance.
- Network configuration: Network configuration, subnet, vpc,...
- IAM Configuration: IAM Access Permissions for Instances.
- UserData: The script runs when the instance starts.

Reusability: The Launch Template can be used multiple times to initialize new instances, helping to ensure consistency and ease of management.

Relationship between Launch Template and ASG:

- Launch Template: The Launch Template provides a detailed template for instances in the ASG.
- ASG uses Launch Template to create instances: When ASG needs to initialize more instances, it will use Launch Template to create new instances.
- Ensure Consistency: The use of the Launch Template helps ensure that all instances in the ASG have the same configuration, making it easy to manage and maintain.

Purpose of using Launch Template with ASG:

- Centralized configuration management: Instead of configuring one instance at a time, we only need to manage a single Launch Template.
- Scale easily: As demand increases, ASG will automatically initialize additional instances based on Launch Template.
- Ensure stability: All instances in the ASG have the same configuration, which minimizes the risk of deploying changes.
- Reusability: Launch Template can be used to create a variety of ASGs, saving time and effort.

Example:

Currently on AWS EC2 there is a web application called Jitsi Meet. To ensure that your app always meets the needs of your users, you can:

- **Create a Launch Template:** Detailed configuration for the instance running the Jitsi Meet web application.
- **Create an ASG:** Link the ASG to the Launch Template you just created.
- **Set up scaling policies**: Define scaling policies so that ASG automatically adjusts the number of instances based on CPU utilization or number of connections.

3.3.6 Auto Scaling Groups details

Detailed Configuration:

- **Desired capacity**: 1 (ASG Ongoing 1 instance).
- **Minimum capacity**: 1 (The minimum number of instances that ASG will maintain, whether or not there is a workload).
- **Maximum capacity**: 2 (The maximum number of instances that an ASG can scale up as the load increases).
- **Launch template**: Jitsi-Meet-Y6S-LaunchTemplate-ALB
    - The configuration template is used to create new instances for the ASG.
    - This template defines the instance type, AMI, security group, and other configuration parameters of the Jitsi Meet instance.
- **Instance type**: t2.micro (Instance types used in ASG).
- **AMI**: ami-0440215 (This AMI contains the operating system and applications required to run Jitsi Meet).
- Security groups: sg-02274632052138 (The security group defines the rules that allow access to the instance from the outside).
- Availability Zones: ap-southeast-2a, ap-southeast-2b, ap-southeast-2c (ASG will distribute instances across 3 Availability Zones to increase application resiliency)
- Subnet: subnet-0592064466502 (Instances in the ASG will be placed in this subnet).

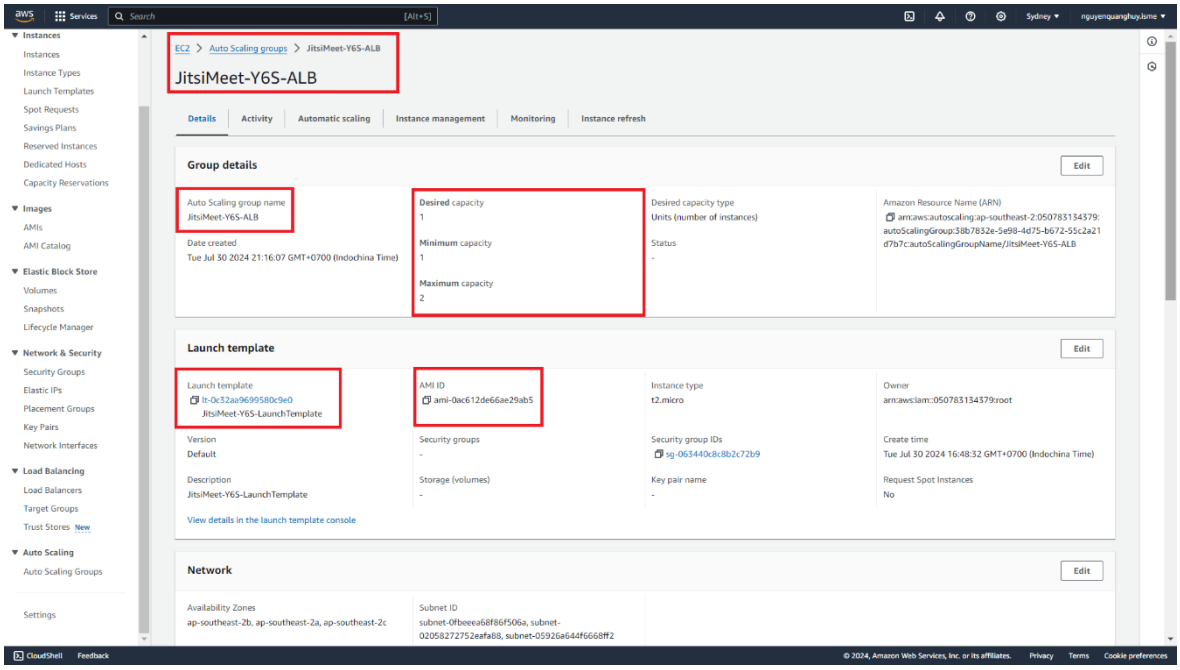

Figure 44. Auto Scaling Groups.

Target Tracking Policy

- As required to maintain **Average CPU** utilization at **50%:**
  - Objective: The goal of the policy is to maintain the average CPU usage of the instances in the group at 50%.
  - What it means: This means that the Auto Scaling Group will automatically adjust the number of instances to ensure that on average each instance uses only 50% of the CPU resources.
- Add or remove capacity units, in increments of 1:
  - Adjustment step: When you need to adjust the number of instances, the system will increase or decrease the number of instances by each unit (in this case, 1 instance).
- 300 seconds to warm up before including in metric:
  - Warm-up time: Each newly added instance will need a period of 300 seconds to boot up and be ready to serve before being counted towards CPU utilization.

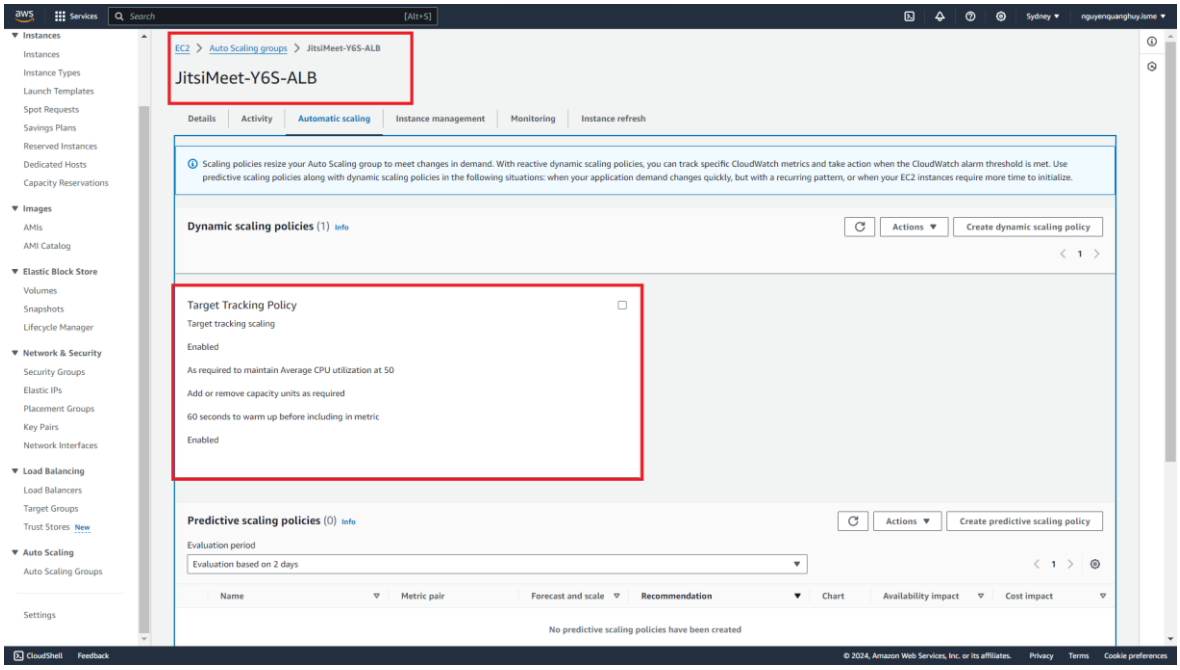

Figure 45. Target tracking policy of ASG.

3.3.7 Application Load Balancers (ALB) và Auto Scaling Groups (ASG)

Application Load Balancers (ALB) and **Auto Scaling Groups (ASGs)** are two **AWS** services that work in tandem to provide a load balancing and auto-scaling solution for web applications.

Both of these services play a critical role in ensuring the performance, availability, and scalability of applications.

The detailed relationship between **ALB** and **ASG**:

- **ALB** Traffic Distribution:
    - **ALB** is a layer 7 load balancing service, which means it can distribute traffic based on application-layer information such as **HTTP** headers, paths, or other attributes.
    - **ALB** will distribute traffic to the **instances** in **the ASG** evenly, helping to ensure that no instances are overloaded.
- **ASG** provides instances:

- An ASG is a group of EC2 instances that are managed automatically. ASG will automatically adjust the number of instances in the pool based on the scaling policies that have been set.
- Instances in the ASG will be registered with ALB to receive traffic.

- **Closely linked**:
  - **ALB** and **ASG** work closely together to form a complete system.
  - As the demand of the application increases, **ASG** will automatically initialize new instances.
  - These new instances will automatically be registered with **ALB** and start receiving traffic.
  - Conversely, when demand drops, **ASG** automatically shuts down unnecessary instances.

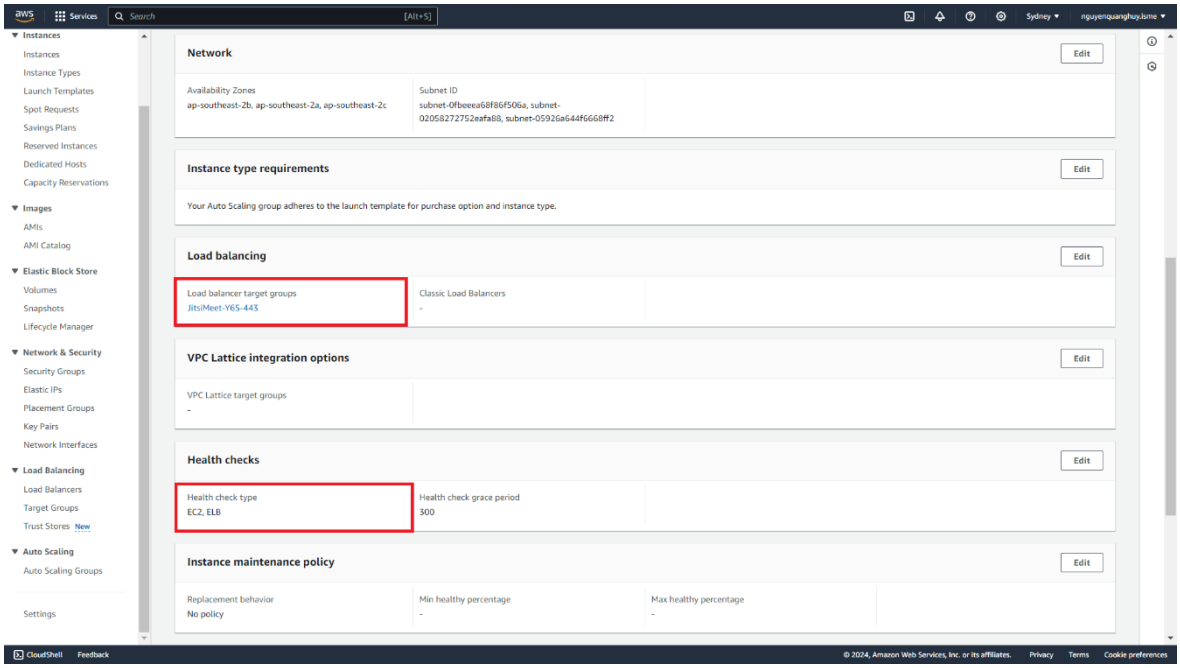

Figure 46. ALB to ASG.

CHAPTER 4. EXPERIMENT

4.1 Auto Scaling Groups and Application Load Balancer

Suppose the sudden increase in the number of requests may be due to a large number of users accessing the Jitsi Meet system at the same time, which could be due to a large online meeting or a specific event. As shown in the image below, we can see that there are "212 Requests" sent to ALB at the same time:

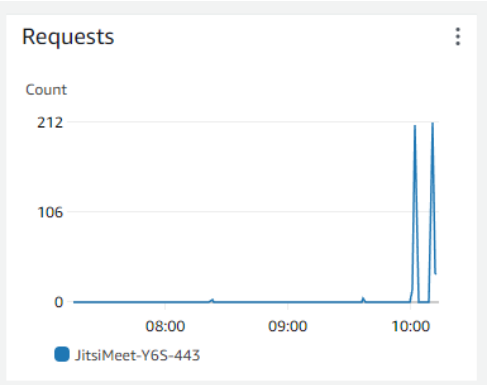

Figure 47. Requests matrix on ALB

For example, ALB currently has only 2 Targets registered with ALB.

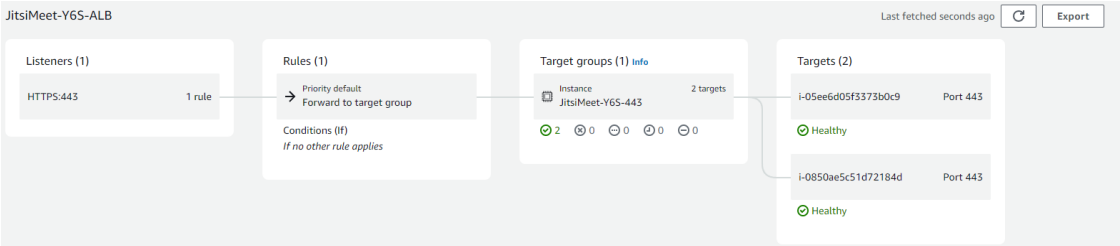

Figure 48. Targets of ALB

ALB will divide these requests evenly among registered Targets, with 212 requests that ALB receives, it will divide for each Target of ALB about 106 requests as shown below:

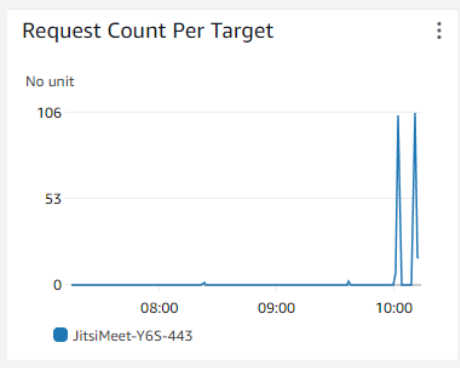

Figure 49. Requests count on per target.

Next, we'll reconfigure the Auto Scaling Group a little bit to make it easier for AWS to automatically deploy Auto Scaling. We'll use the Application Load Balancer request count per target instead of using Average CPU utilization, which will make Auto Scaling easier to deploy based on the Application Load Balancer's metrics:

- **Metric Type:** Application Load Balancer request count per target: This metric measures the number of requests that ALB processes for each target (instance) in a group. When the number of requests exceeds or falls below the set target, Auto Scaling will increase or decrease the number of instances to maintain performance.
- **Target Group**:
    - JitsiMeet-Y6S-443: This is the specific target group selected, associated with the instances running Jitsi Meet. ALB will distribute traffic to instances in this pool.
- **Target Value**:
    - 10: This is the number of requests that each instance in the target group is expected to handle. If the average number of requests per instance exceeds or falls below this value, the system will automatically adjust the number of instances to maintain this level.

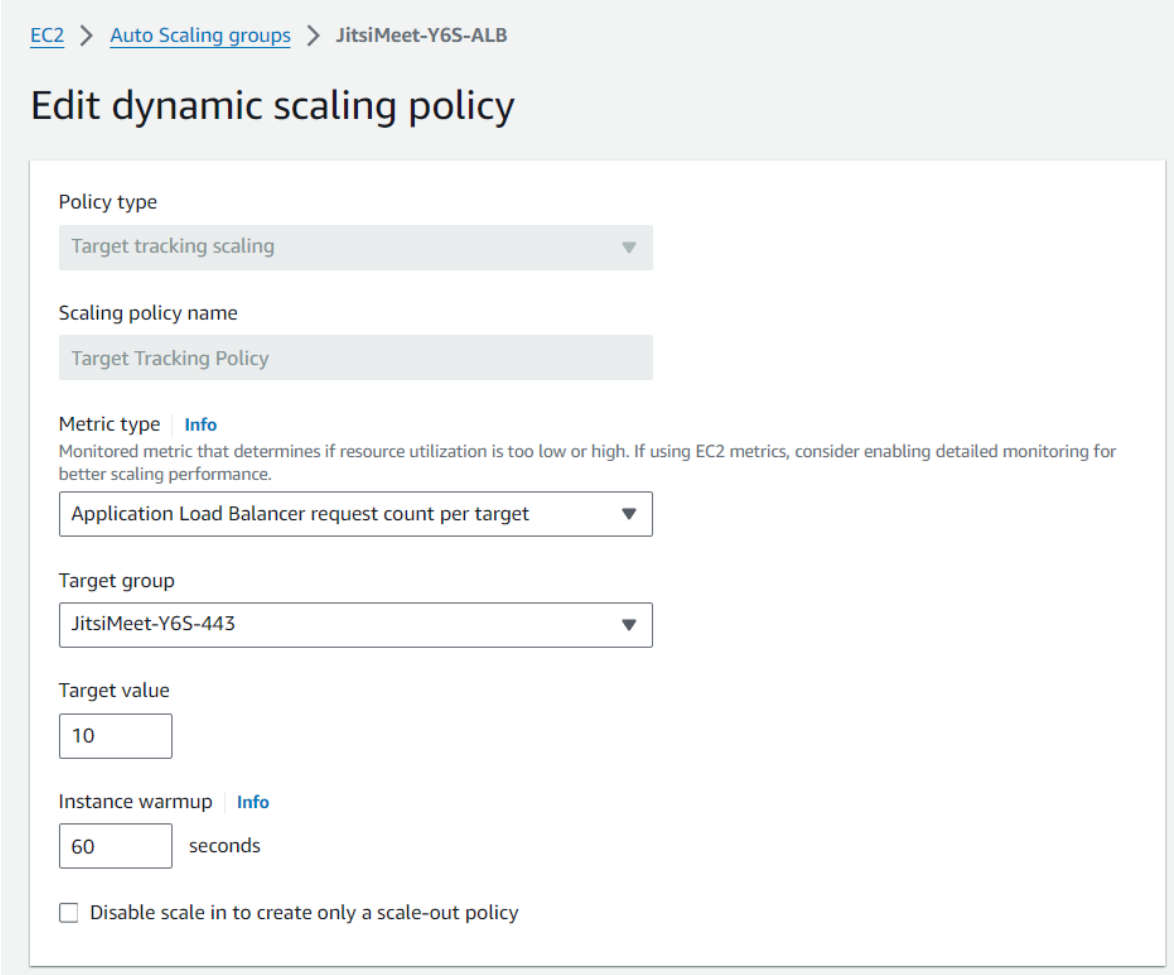

Figure 50. Scaling Policy

As we mentioned, the number of requests to 1 target is about 106 requests, which will exceed **the target value** of **10** of the Auto Scaling Groups that we have configured, so ASG will automatically expand EC2 Instances to ensure that the system is always stable. When ASG expands EC2 Instances, there will be Activity History logs to check as well as look up.

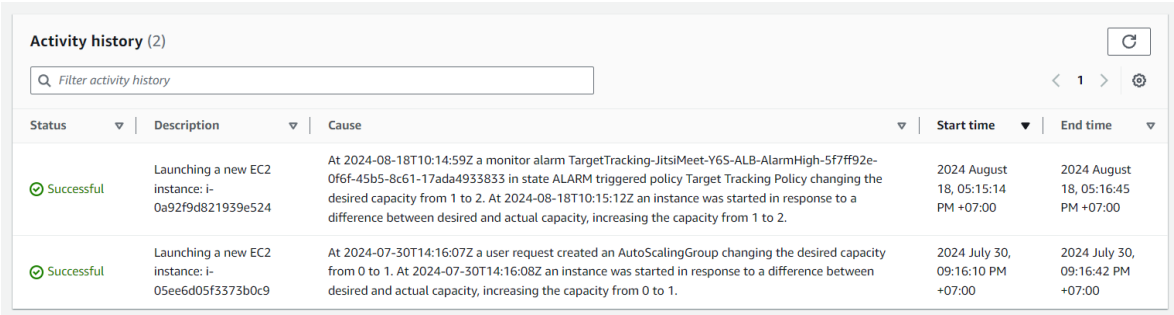

Figure 51. ASG activity history.

When ASG expands EC2 Instances, it will automatically add the newly created Instance to the Application Load Balancer groups to receive requests as well as ensure the ability to reduce the load on the system.

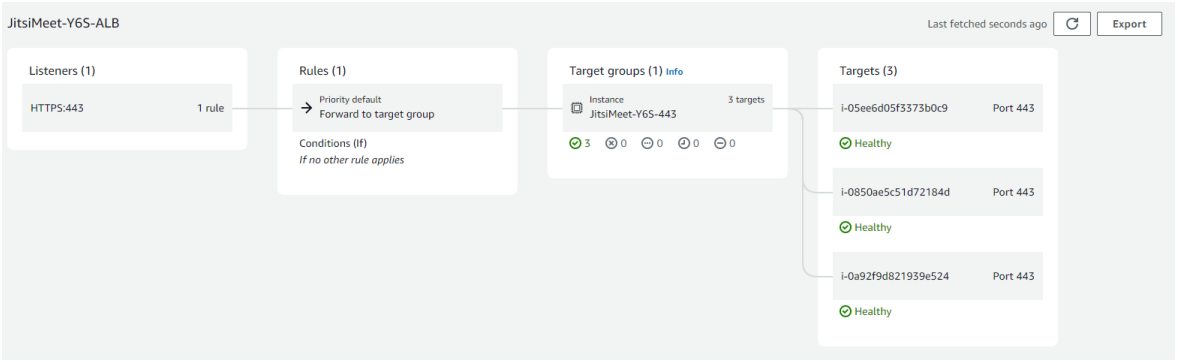

Figure 52. Targets of ALB.

As we can see, when the ASG creates a new EC2 Instance, we can see a new Instance added.

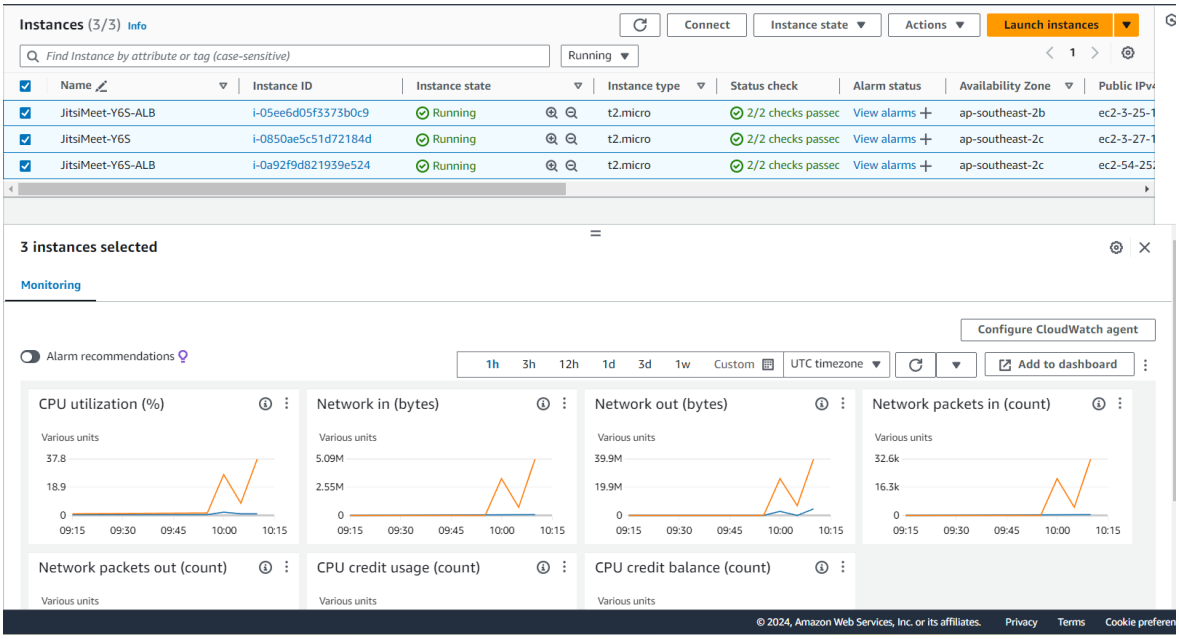

Figure 53. List instance ec2.

After the system is no longer overloaded, the newly created Instance will be canceled to ensure cost reduction.

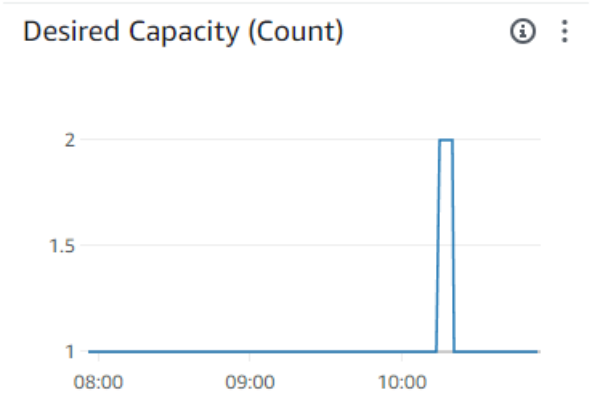

Figure 54. Desired Capacity.

4.2 Cloudwatch

Amazon CloudWatch is a monitoring and management service from AWS. It provides capabilities to collect and track metrics, collect and monitor log files, set up alerts, and react automatically to changes in AWS resources.

4.2.1 CloudWatch can monitor resources and applications in real-time.

Key components of CloudWatch:

- Metrics: Collect metrics on the performance and performance of AWS resources, such as CPU utilization, disk I/O, network traffic, etc. v.v.
- Alarms: Set up alerts based on metrics to take actions such as sending notifications or triggering Auto Scaling.
- Logs: Collect and monitor log files from AWS resources and applications, helping to analyze and resolve issues.
- Events: Monitor events from AWS services and applications, triggering automated actions based on event patterns.
- Dashboard: User interface for creating custom dashboards, displaying metrics and logs in a visual way.

4.2.2 Use CloudWatch with other services.

4.2.3 EC2 Instances

- Metrics:
    - CloudWatch automatically collects some default metrics from EC2 such as CPU utilization, disk I/O, network traffic.
    - Additional custom metrics can be configured using the CloudWatch Agent.

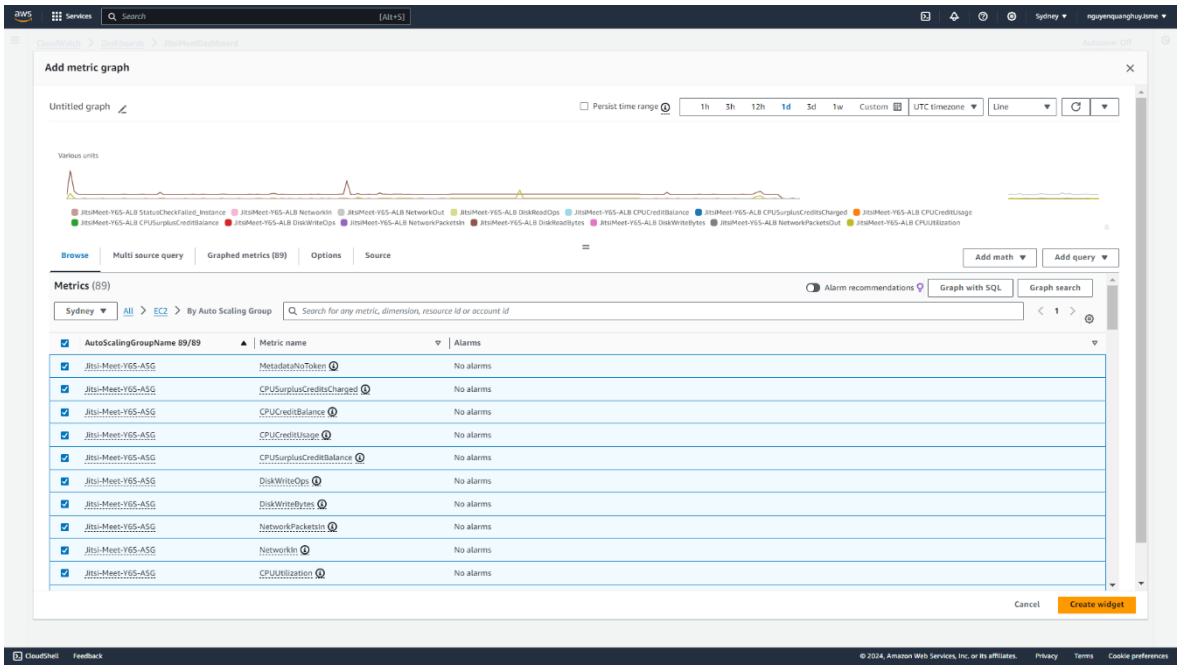

Figure 55. Metric graph.

- Alarms:
    - Create alerts based on EC2 metrics.
    - Example: Warning when CPU utilization is exceeded 80%.
    - These alerts can be configured to send notifications via SNS or trigger Auto Scaling actions.
- Logs:
    - Use CloudWatch Logs to collect logs from applications running on EC2. Install the CloudWatch Agent to send logs from an EC2 instance to CloudWatch.
- Dashboard:

- Create custom dashboards to monitor metrics and logs from EC2 instances.

4.2.4 Application Load Balancer (ALB)

- Metrics:
  - CloudWatch collects metrics from ALB such as request count, error count, latency, target health, v.v.
  - These metrics help track the performance and health of ALB.

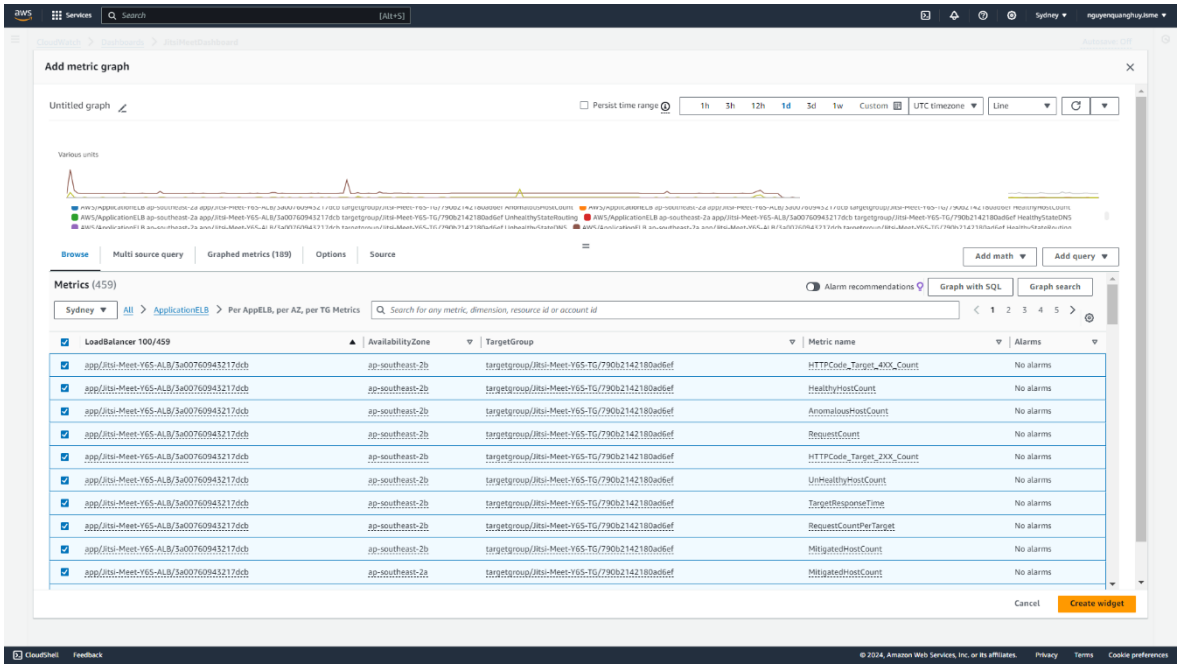

Figure 56. Metric graph.

- Alarms:
  - Create alerts based on ALB indicators. Example: Warning when the error rate of 5xx exceeds a defined threshold.
  - Alerts can be configured to send notifications via SNS or trigger Auto Scaling actions.

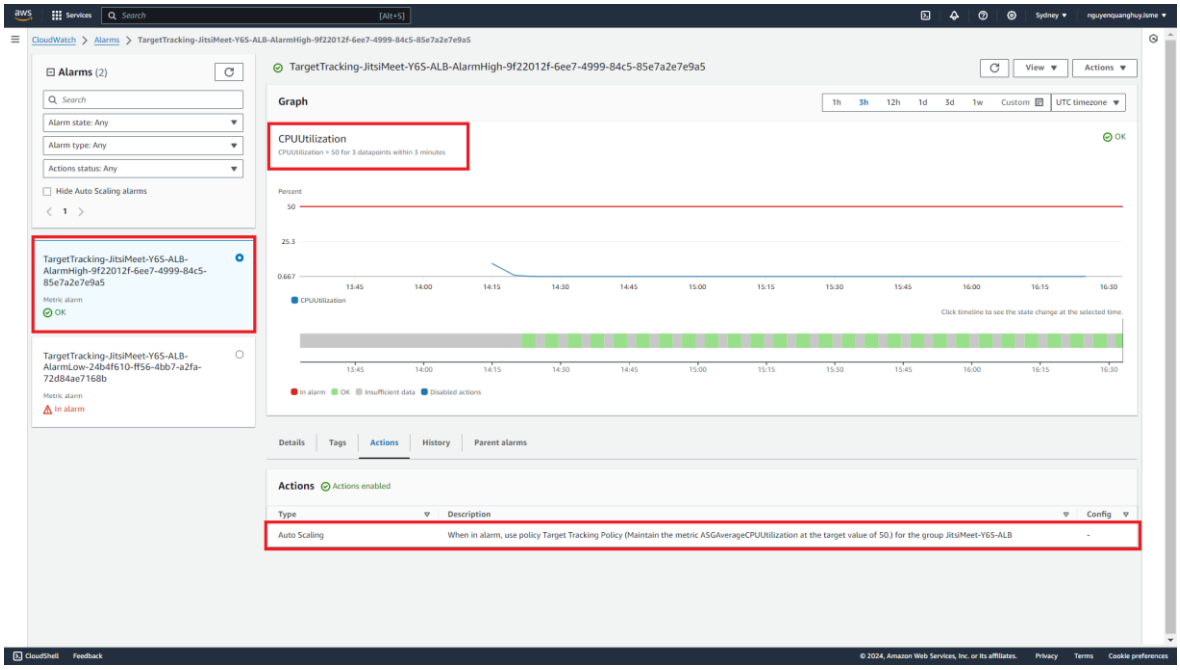

Figure 57. Alarms of CloudWatch.

- Logs:
    - Use CloudWatch Logs to collect logs from ALB. Configure ALB to send logs to CloudWatch Logs, which provides detailed analysis of HTTP/S requests.
- Dashboard:
    - Create custom dashboards to monitor metrics and logs from ALB.

4.2.5 Auto Scaling Groups (ASG)

- Metrics:
    - CloudWatch collects metrics from ASGs such as number of instances, desired capacity, in-service instances, pending instances, v.v.
    - These metrics help track and manage the performance and scale of the ASG.

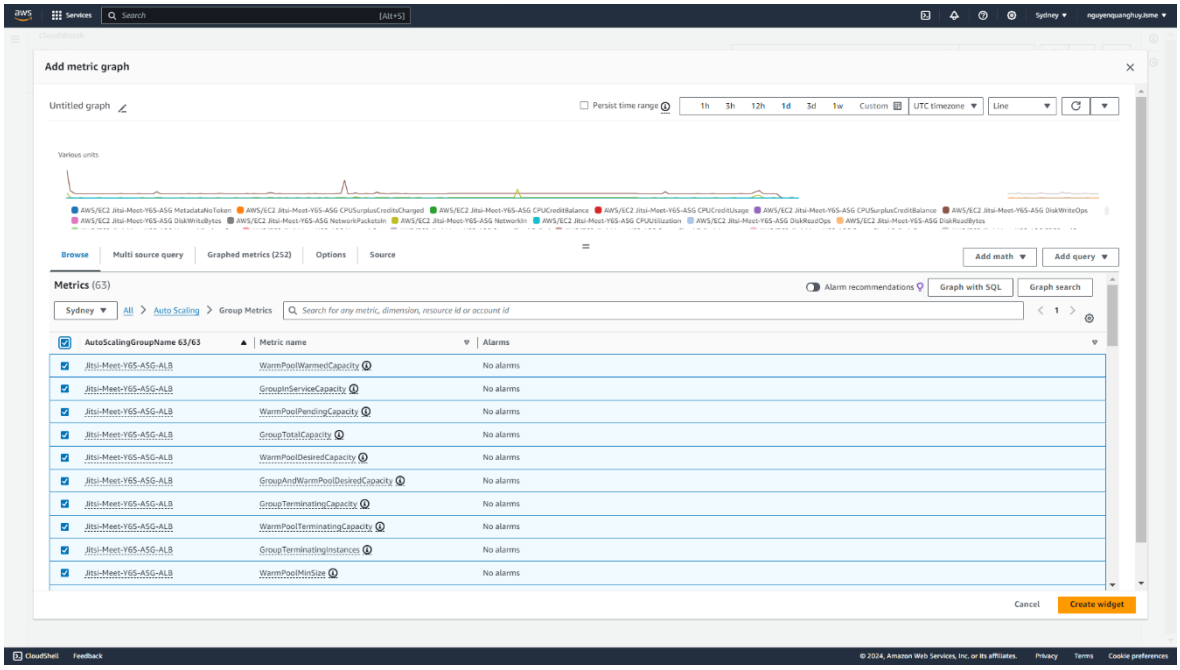

Figure 58. Metric graph of ASG.

- Alarms:
    - Create alerts based on ASG metrics.
    - Example: Alerts when the number of in-service instances drops below a defined threshold.
    - Alerts can trigger Auto Scaling actions such as increasing or decreasing the number of instances.
- Logs:
    - Collect and monitor logs from instances in the ASG using CloudWatch Logs.
- Dashboard:
    - Create custom dashboards to monitor metrics and logs from ASG.

4.2.6 CloudWatch monitoring details – Auto Scaling

This is where important metrics related to ASG activity can be tracked for Jitsi Meet on ALB. Details about each section in this interface:

Auto Scaling Group (ASG) Key Indicators:

- Minimum Group Size (Count):

    - Minimum number of EC2 instances that ASG will keep at all times.
    - In this example, the value is 1, which means that the ASG will always make sure at least 1 instance runs.
- Maximum Group Size (Count):
    - Maximum number of EC2 instances that ASG can scale to.
    - The value is 2, which means that the ASG will not create more than 2 more instances.
- Desired Capacity (Count):
    - The number of instances that ASG wishes to run to meet the current requirement.
    - This is a value that ASG will strive to uphold. If the current number of instances is less or more than this value, ASG will automatically create or remove more instances.
- In Service Instances (Count):
    - The number of instances that are currently active and ready to receive traffic from users.
    - This metric helps to know how many instances are running and ready to serve.
- Pending Instances (Count):
    - Number of instances in the process of initialization but not yet ready to operate.
    - This metric helps to know how many instances are being created to respond to load changes.
- Standby Instances (Count):
    - The number of instances that are on standby, i.e., have been initialized but are not participating in the processing of current traffic.
    - Usually rarely used, this metric can be used when it is necessary to prepare an instance in advance for unexpected operations.
- Terminating Instances (Count):

- Number of instances in the process of shutting down and removing from ASG.
- This metric helps track instances that are being removed by ASGs because they are no longer needed or because they don't meet health requirements.

- Total Instances (Count):
  - The total number of instances available in the ASG, including active, initializing, and shutting down instances.

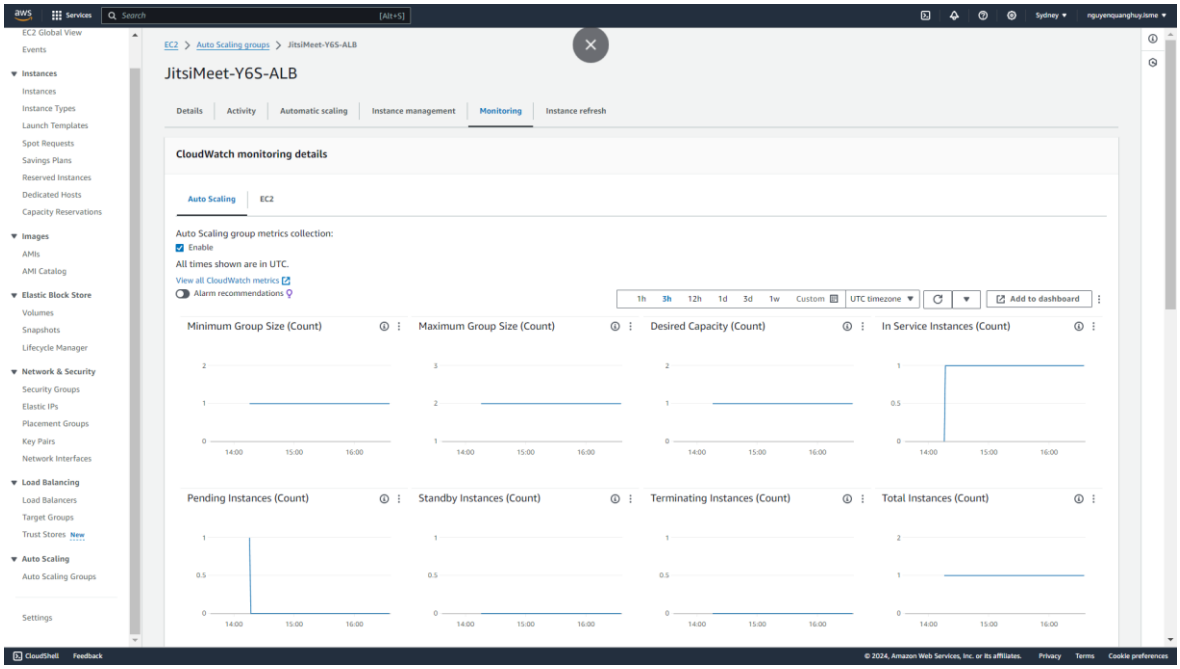

Figure 59. ASG of Jitsi Meet.

4.3 High Availability and Testing Optimization

4.3.1 Setting up instance maintenance policies for Auto Scaling Groups (ASGs) in AWS.

Instance Maintenance Policy

Instance maintenance policies allow you to control how ASG manages instance replacement events. This includes health checks, instance refreshes, instance life-related events, and events that occur automatically to keep the group balanced, called rebalancing events.

Instance Replacement Options:

- Mixed Behavior
    - No Policy: For rebalancing events, new instances will be launched before removing other instances. For other events, the instance will be removed and launched at the same time.
- Prioritize Availability
    - Launch Before Terminating: Launch new instances and wait for them to be ready before removing the old ones. This allows for exceeding the desired capacity at a certain rate and can temporarily increase the cost.
- Control Costs
    - Terminate and Launch: Remove and launch instances at the same time. This allows for a drop below the desired capacity by a certain percentage and can temporarily reduce availability.
    - Set Healthy Percentage.
    - Min: Set the minimum percentage of desired capacity that must be healthy and ready for EC2 Auto Scaling to replace the instances.
    - Max: Set the maximum percentage of desired capacity that must be healthy and ready for EC2 Auto Scaling to replace the instances.
    - Example:
        - Min: 90%: This means that at least 90% of the desired number of instances must be healthy to proceed with the replacement.
        - Max: 100%: This means that up to 100% of the desired number of instances must be healthy to proceed with the replacement.
    - Scaling Limits: When replacing an instance, the group's scaling limits may be temporarily exceeded based on current calculations.
- Flexible
    - Custom Behavior: Set custom values for the minimum and maximum amount of available power.

- This provides more flexibility in setting EC2 Auto Scaling's desired capacity overruns and shortages when replacing instances.

Purpose and Benefits:

- Instance maintenance policies help control instance replacement in the ASG to optimize between availability and cost.
- Availability: Ensure that the system is always available even when replacing instances.
- Cost: Manage costs by limiting the number of instances launched at a time.
- Flexibility: Provides customization options to suit the specific needs of the application and business.

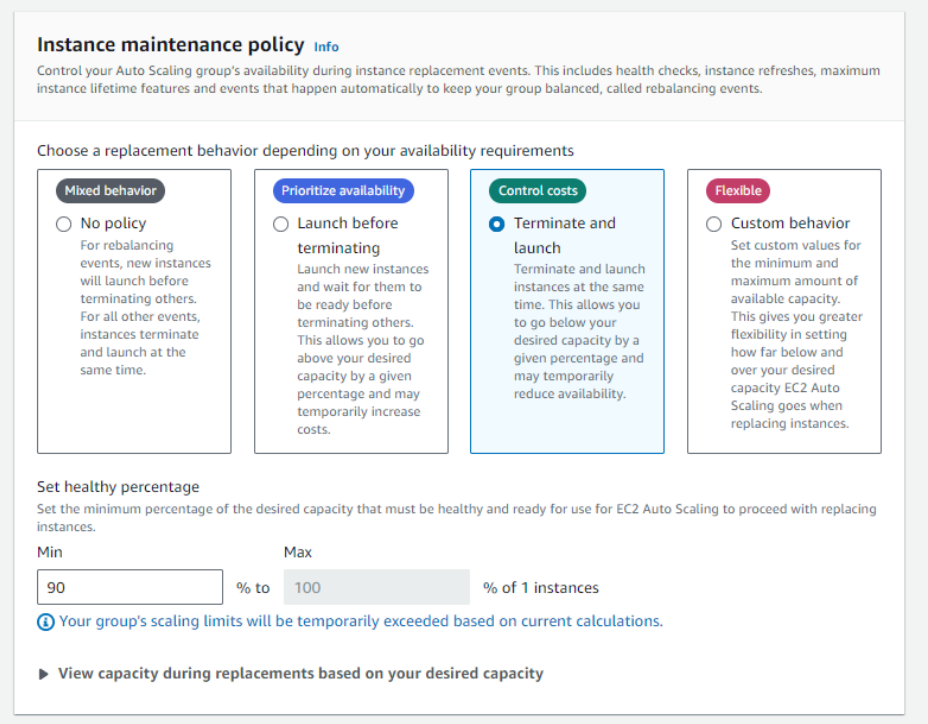

Figure 60. Instance maintenance policy for ASG.

4.3.2 Testing and Optimization

Using Jitsi Meet Stress Test Tool to simulate a group of users participating in a Jitsi Meet conference. You can customize the number of sessions and the duration of the session to your liking. Choose your media options (video and audio, video only, audio only, or none). Track progress as you test.

Requirements:

- Python version 3.12.5

```
C:\Users\hhiep>python --version
Python 3.12.5
```

Figure 61. Check python version.

- Node version 20.16.0, npm version 10.8.1

```
C:\Users\hhiep>node --version
v20.16.0
```

```
C:\Users\hhiep>npm --version
10.8.1
```

Figure 62. Check node and npm version.

- Python Package Installer:

```
C:\Users\hhiep>pip --version
pip 24.2 from D:\Lib\site-packages\pip (python 3.12)
```

Figure 63. Check pip version.

- Automated open source testing tool: Selenium
- Google chrome browser version 127.0.6533.100
- ChromeDriver version 127.0.6533.100

Testing:

- Fire up a command prompt or terminal in the directory containing this repository

```
C:\Users\hhiep\Downloads\jitsi-stress-test-main>
```

Figure 64. Fire up a command prompt in the directory.

- Run the script using Python:

```
C:\Users\hhiep\Downloads\jitsi-stress-test-main>python jitsi_stress_test.py
Please enter the server address (http(s)//jitsi.meet):
```

Figure 65. Run the python script.

- Follow the prompts to tailor the stress test:

- Enter the Jitsi Meet server address: https://alb.jitsi-meet-tdtu-y6s.online/
- Enter the number of sessions to simulate: 25
- Enter the runtime in seconds for the stress test: 1800
- Choose the media option (1 - Video and Audio [default], 2 - Video only, 3 - Audio only, 4 - None): 1

```
C:\Users\hhiep\Downloads\jitsi-stress-test-main>python jitsi_stress_test.py
Please enter the server address (http(s)//jitsi.meet): https://alb.jitsi-meet-tdtu-y6s.online/
Please enter the number of sessions: 25
Please enter the runtime in seconds: 1800
Choose the media option (1 - Video and Audio [default], 2 - Video only, 3 - Audio only, 4 - None): 1
```

Figure 66. Enter the information required for testing.

- Jitsi meet conversation:

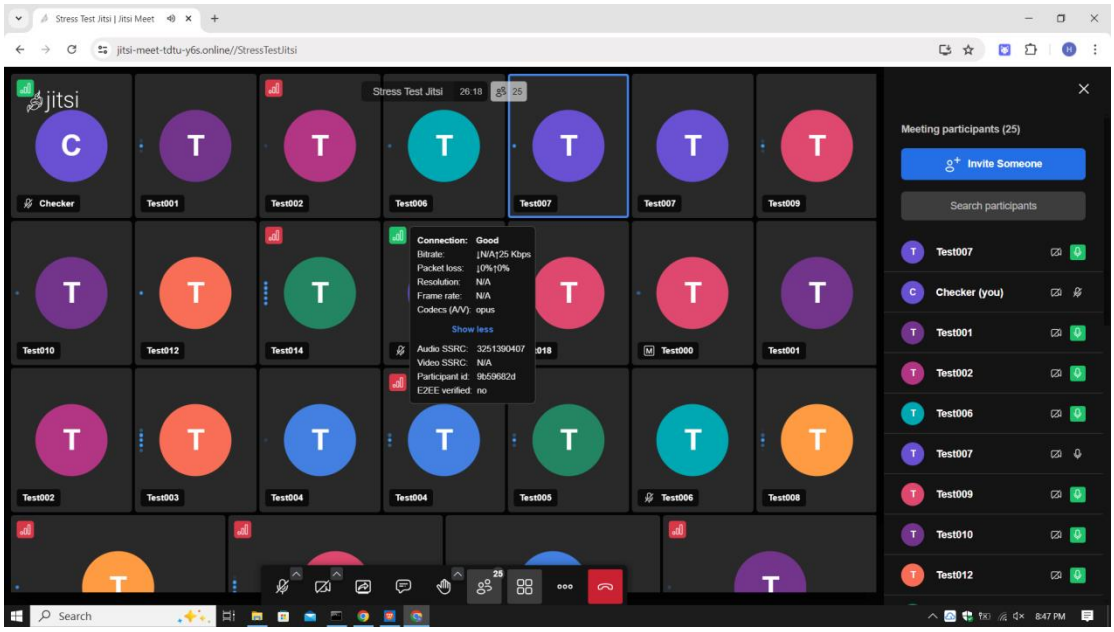

Figure 67. Jitsi meet conversation room.

Detailed information about the hardware of the first ec2 server:

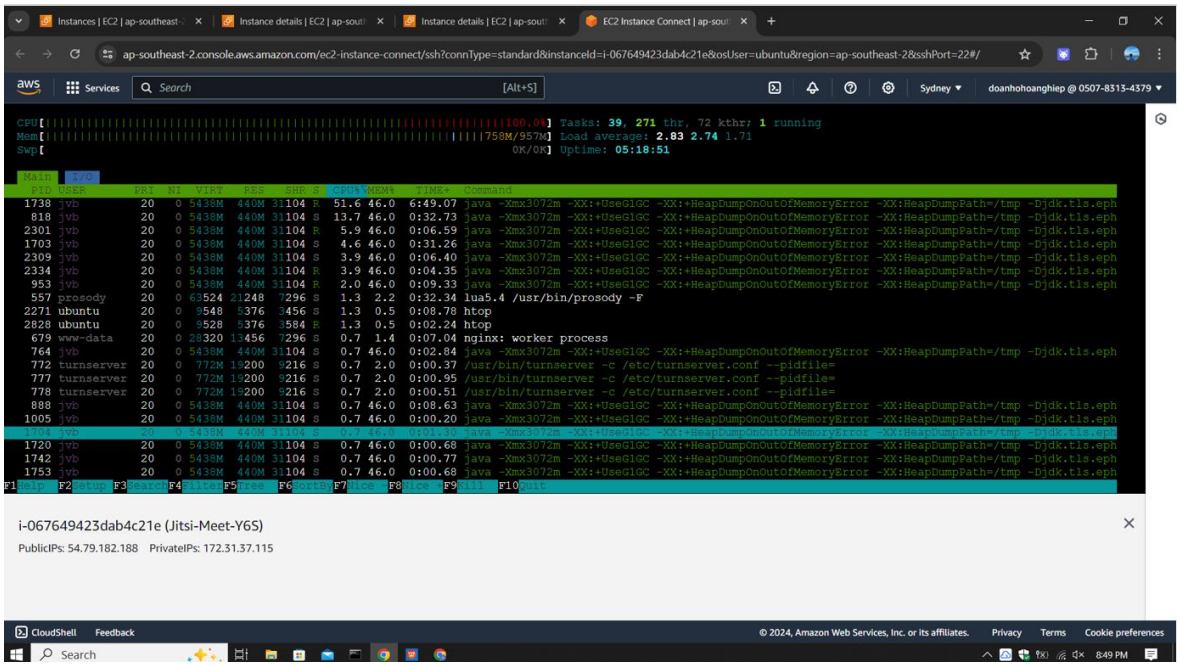

Figure 68. Hardware details of first ec2.

Hardware details of the second EC2 server:

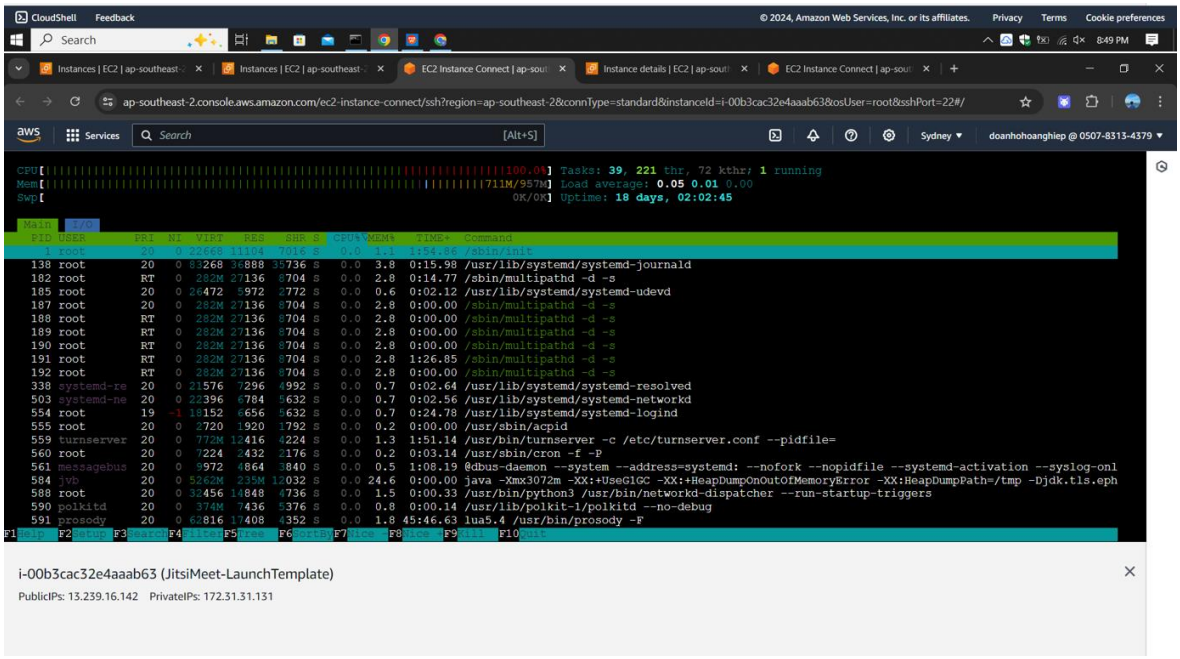

Figure 69. Hardware of second ec2

When there are not too many users using the Jitsi Meet system, the system will still divide the requests into active ec2 instances based on ALB to help the system run smoothly without lag or server overload.

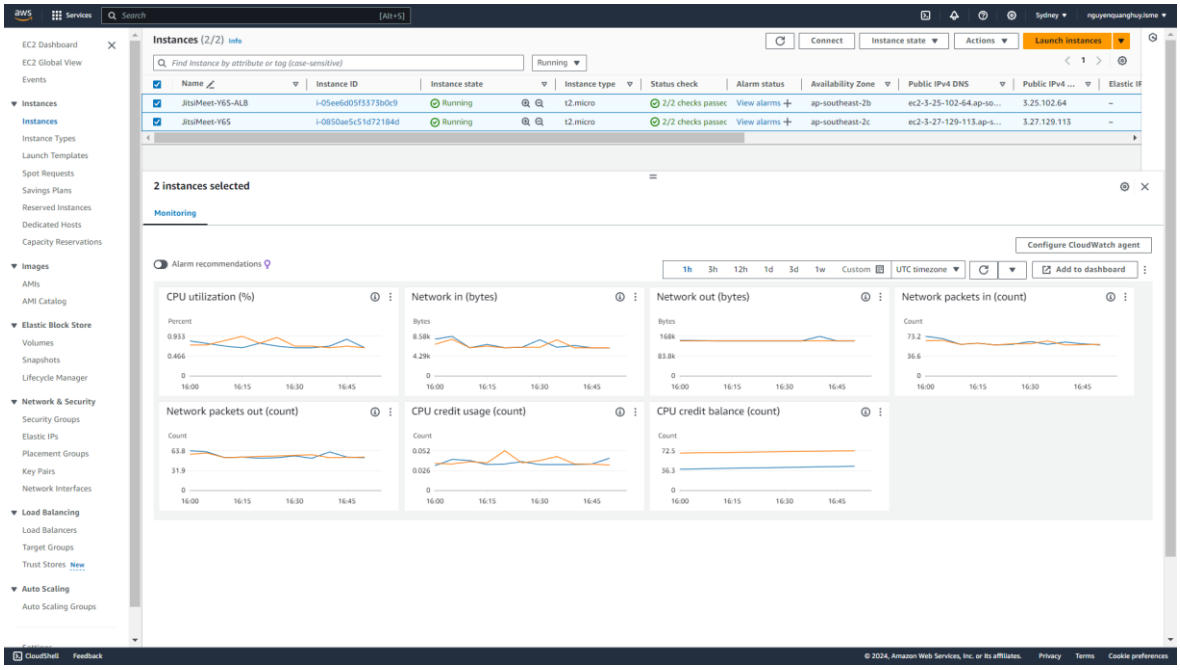

Figure 70. EC2 monitoring.

CHAPTER 5. CONCLUSION

5.1 Results Achieved

Through the process of deploying and testing the Jitsi Meet system with the ability to automatically scale on the AWS platform, the topic has achieved important results. First of all, the Jitsi Meet system has been successfully installed and configured on the cloud infrastructure, with the integration of technologies such as Application Load Balancer (ALB) and Auto Scaling Groups (ASG). This ensures that the system can automatically adjust resources (such as EC2 instances, CPU, and memory) to better meet actual usage needs, especially when the number of users changes. In addition, the application of security measures such as access management (IAM), setting up security groups (Security Groups), and using TLS certificates has helped increase the security of the system.

5.2 Pros and Cons

Pros:

Automatic Scaling: The system is capable of automatically scaling resources based on actual usage, helping to optimize performance and save costs.

High Security: The implementation of strict security measures, such as the use of TLS certificates and access control, ensures the safety and protection of user data.

High Availability: The system is designed to operate continuously, even when incidents occur, thanks to multi-region deployment and the use of AWS services.

Cons:

Operating Costs: While auto-scaling saves costs when demand is low, operating costs can increase significantly when the number of users increases.

Configuration Complexity: The process of deploying and configuring the system requires in-depth knowledge of both Jitsi Meet and AWS cloud services, which can be difficult for beginners.

5.3 Future Development Directions

In the future, the system can be further developed in the following directions:

Cost Optimization: Find and apply more optimal solutions to reduce operating costs, such as using lower-cost instance types in AWS or optimizing auto-scaling policies.

Enhanced Security: Although there are existing security measures, it is necessary to continue to improve and update the latest security mechanisms to deal with increasingly sophisticated threats.

Develop New Features: Research and integrate new features of Jitsi Meet to enhance the user experience, such as improving video quality, expanding integration with other systems, and providing more meeting management tools.

Cross-Platform Scalability: Expand the deployment of the Jitsi Meet system on cloud platforms other than AWS such as Google Cloud or Microsoft Azure, to increase flexibility and compatibility.

These development directions not only help maintain the stability and efficiency of the system, but also better meet the needs of users in different contexts.