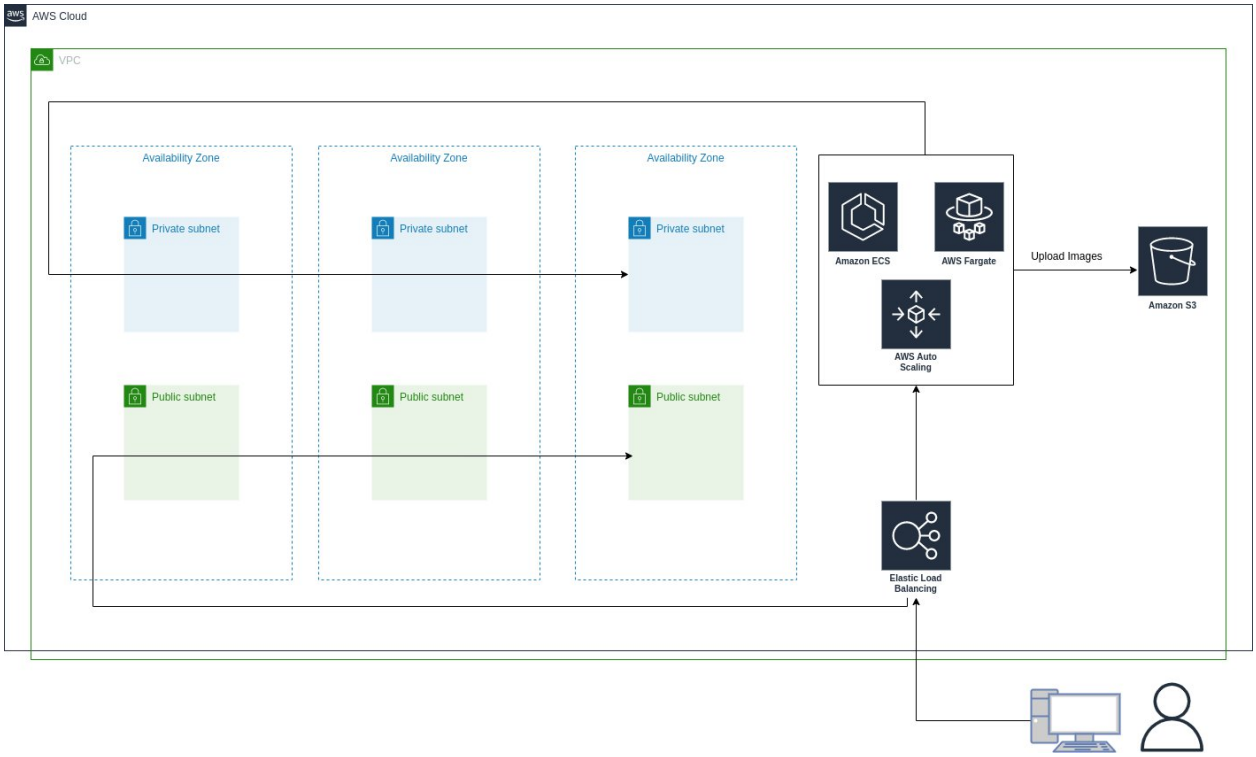
Report Block 2 Zurich Hackathon

Diagram:



I have decided to set up an ECS (Elastic Container Service) cluster with a service running a container that hosts the web application. My main goal is to ensure maximum availability, so I have chosen to deploy a minimum of three replicas across three availability zones (AZs). I have configured the services to be exposed privately within my VPC and made them accessible through a public load balancer.

The ECS service utilizes an IAM role to establish a connection with S3. This role is configured to have the necessary permissions to access the specified S3 bucket. Additionally, the ECS task definition specifies a container image that has been uploaded to Docker Hub.

The ECS service pulls this image from Docker Hub during task startup to ensure the container is running the desired version of the application.

Docker Image: https://hub.docker.com/r/ajotaops/zurich-hackathon (Dockerfile are on docker folder on block2)

Explanation and decisions:

ECS Cluster: I have created an ECS cluster to manage the deployment and scaling of my containerized web application. This cluster provides a logical grouping for my ECS tasks.

- Part of problem:: The ECS cluster solves the problem of efficiently managing and orchestrating the deployment and scaling of containerized applications. It provides a centralized platform for managing tasks, scheduling, and resource allocation.
- Efficiency: Compared to manually managing containers or using other container orchestration platforms, ECS offers a highly efficient solution. It simplifies the management of containers and automates many tasks, such as load balancing, scaling, and health monitoring.
- Cost Optimization: ECS optimizes costs by allowing efficient resource utilization. It dynamically scales the number of task instances based on demand, ensuring that resources are allocated only when needed. This helps minimize idle resources and reduces unnecessary costs.

ECS Service: Within the ECS cluster, I have set up an ECS service. This service allows me to manage the desired number of task instances running my containerized web application. To ensure high availability, I have specified a minimum of three replicas for the service, distributing them across the three availability zones.

- Part of problem: The ECS service addresses the need for managing the lifecycle of a specific task or set of tasks, ensuring their availability and scalability.
- Efficiency: Compared to managing individual containers manually, using an ECS service provides automation and ease of management. It simplifies tasks such as scaling, load balancing, resulting in increased operational efficiency.
- Cost Optimization: The ECS service optimizes costs by automatically managing the number of task instances based on demand. It scales up or down to meet the desired capacity, ensuring efficient resource utilization. This prevents overprovisioning and reduces unnecessary costs.

Private Service: I have configured the ECS service to run within private subnets of my VPC. This means that the service instances are not directly accessible from the public internet, providing an extra layer of security.

- Problem: Running services privately within a VPC addresses the need for secure communication and limited access to services.
- Efficiency: By running services privately, you benefit from enhanced security and isolation. It reduces exposure to potential threats and limits access to authorized entities within the VPC.
- Cost Optimization: If you need a NAT Gateway for your services to be able to connect to the internet, if you don't need a NAT Gateway, there is no additional cost.

Load Balancer: To make my web application accessible to the public, I have set up an Application Load Balancer (ALB). The ALB acts as the entry point for incoming HTTP traffic, distributing it across the running instances of my ECS service. In case it is an internal tool and from the work network we have access to the aws private vpc, it could be in that network.

- Part of problem: The load balancer solves the challenge of distributing incoming traffic evenly across multiple instances of a service, ensuring scalability and high availability.

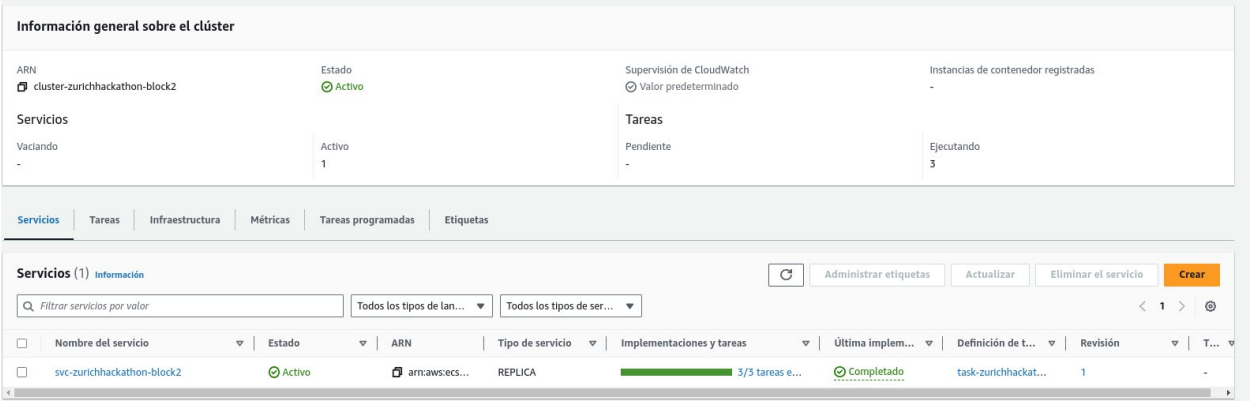
- Efficiency: Load balancers, such as the ALB used in this setup, efficiently distribute traffic based on various algorithms and health checks. They optimize resource utilization by evenly distributing requests and routing them to healthy instances, improving overall performance and availability.
- Cost Optimization: Load balancers optimize costs by efficiently distributing traffic and preventing any single instance from being overwhelmed. This ensures that resources are used effectively, preventing the need for excessive capacity provisioning and reducing costs associated with downtime or performance issues.

Conclusions:

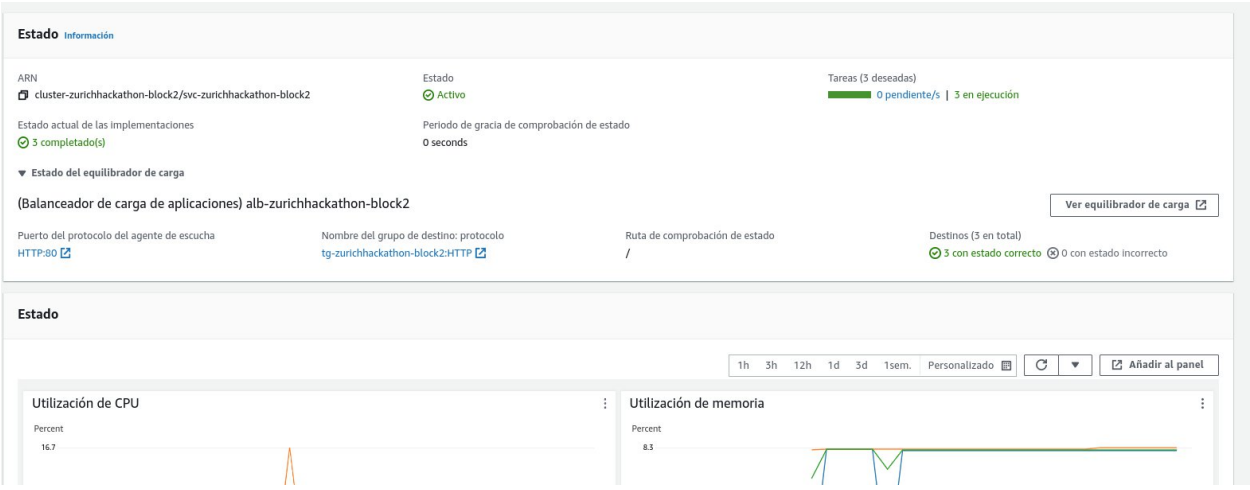
In summary, the ECS cluster, ECS service with private service configuration, and load balancer all solve specific parts of the problem related to managing and scaling this app. They offer efficient solutions compared to other solutions, optimize costs through resource allocation and efficient traffic distribution, and enhance security and availability.

Some images from the process:

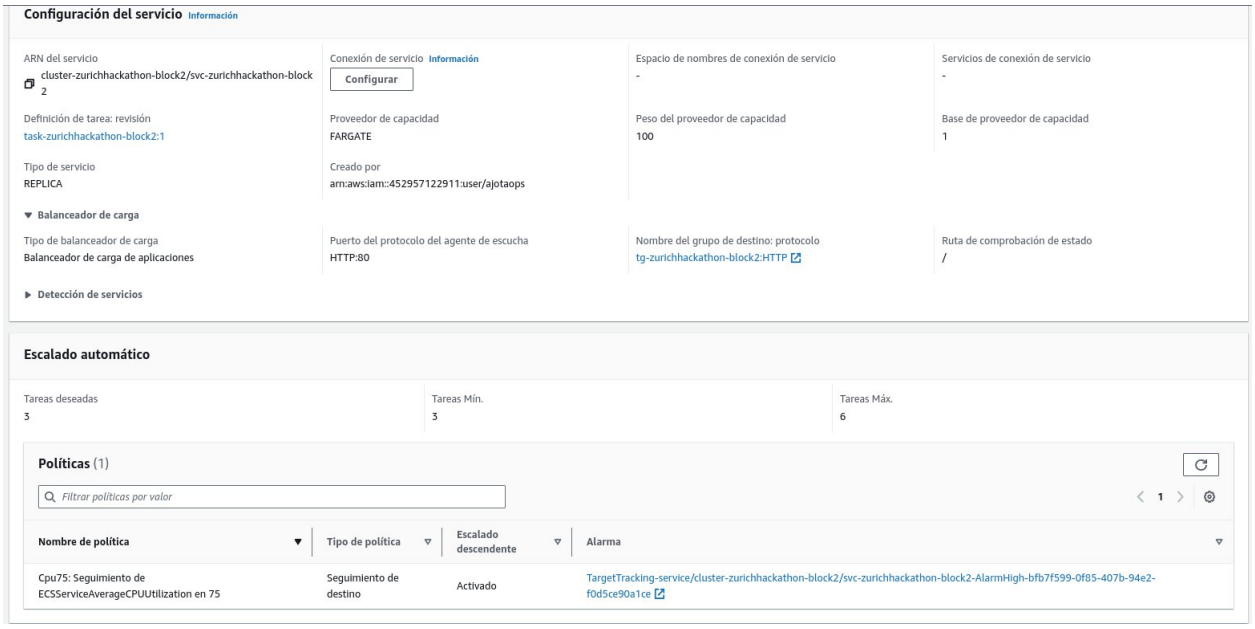
Cluster Info:



Service:

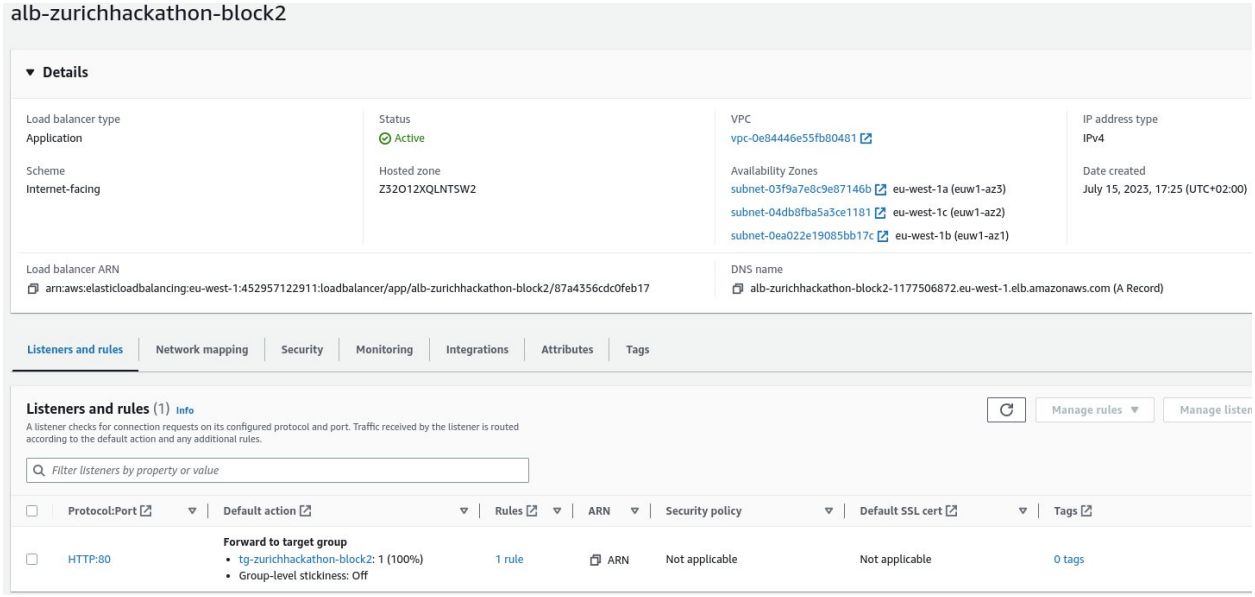


Config of Service:



Target Group:

ALB:



App working on ALB:

Upload file: