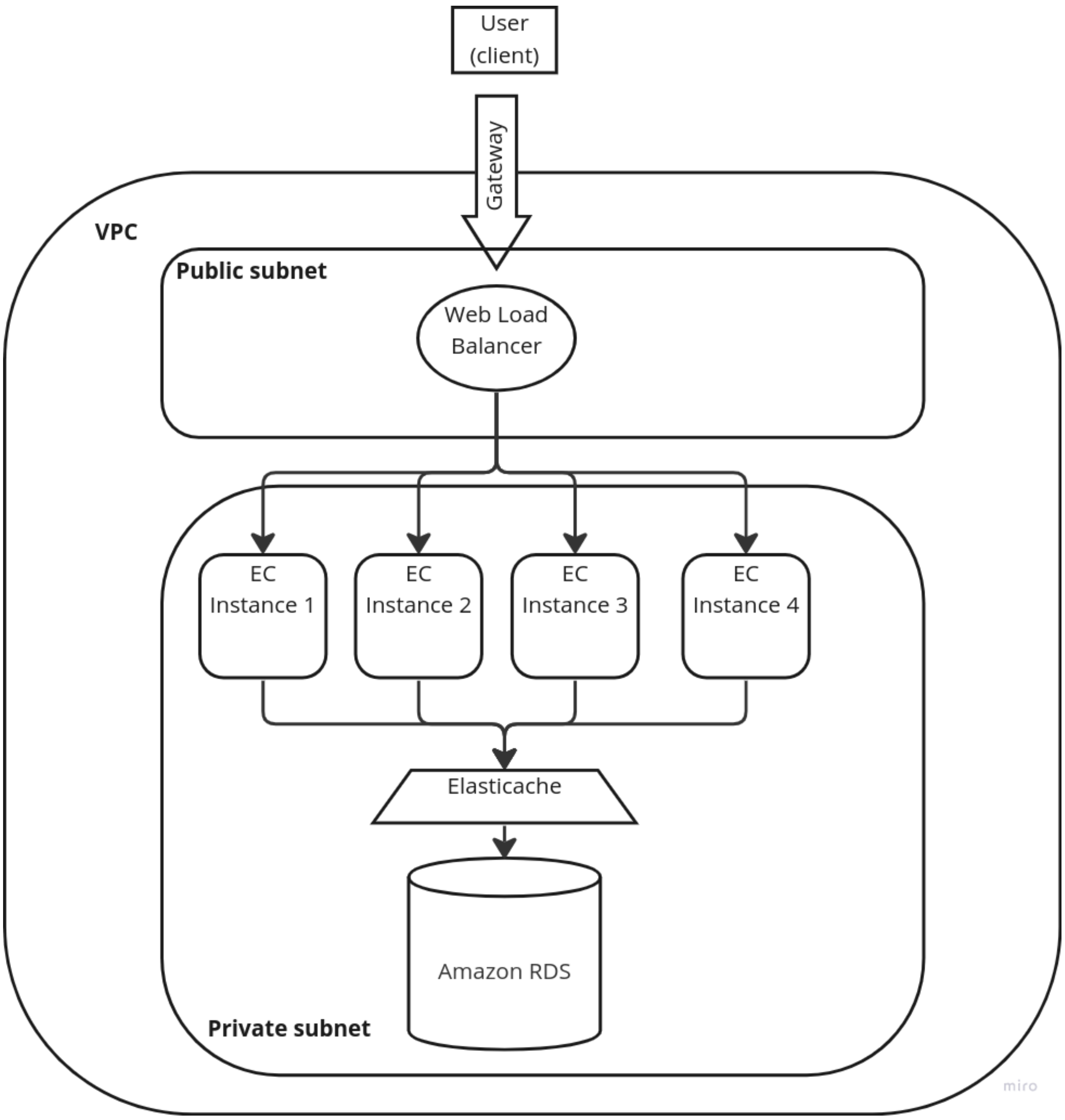
Proposed Infrastructure



Objective

This infrastructure is designed to support 100 requests per second, to be secure but, at the same time, cheap.

Network Segmentation and Isolation

We will create a virtual private cloud (VPC) within AWS. The VPC will contain a private subnet that will host the API instances, while the public subnet will host a load balancer. This will allow us to control inbound and outbound network traffic and restrict access to the API instances to only authorized IP addresses.

Secure Storage of Sensitive Data

We can use AWS Secrets Manager to store and manage keys. The passwords will be encrypted using AWS Key Management Service (KMS) and can be accessed only by authorized users.

HTTPS/TLS Encryption

This can be achieved by obtaining a valid SSL certificate and configuring it on the load balancer. The load balancer will then terminate the SSL connection and pass encrypted traffic to the API instances.

Load Balancer

We can use Amazon Elastic Load Balancer (ELB). ELB is a scalable load balancing service that automatically distributes incoming application traffic across multiple EC2 instances.

Firewall Rules

We can use security groups. Security groups act as a firewall for EC2 instances and allow us to define inbound and outbound traffic rules. We will create a security group for the API instances and allow traffic only from the IP addresses that need access to the API.

Bottlenecks

To achieve the objective of 100 queries per second we will also set up an Amazon ElastiCache between the EC2 instances and the database. This will reduce the load over the database for any repetitive query.

EC2 Instances

Since we can scale horizontally, we can try to use t2.micro instances, but **this is not recommended for production**. t3.micro or t3a.micro are good choices for production.

Database

We can use AWS RDS, a managed relational database service for PostgreSQL, among others.

Conclusion

In conclusion, the implementation of the above components will result in a secure and scalable AWS infrastructure capable of handling 100 queries per second. To test this we can use Apache JMeter to perform a load test and, if needed, we can iterate depending on the results.