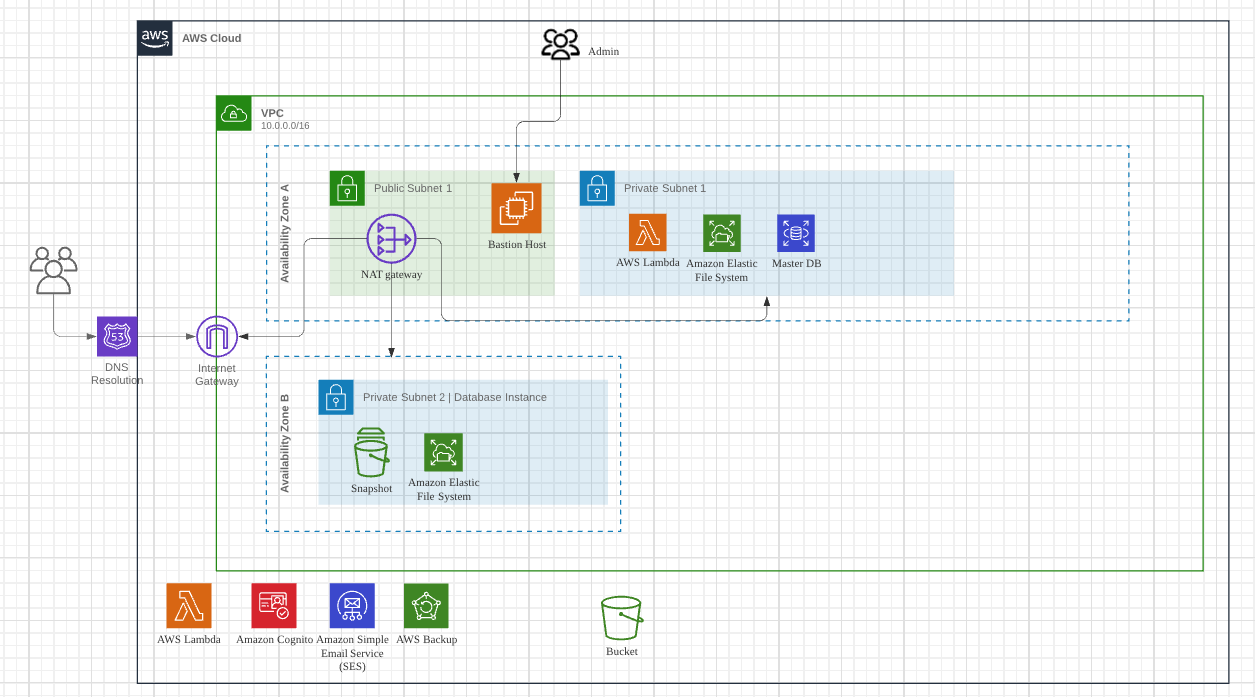
# AWS Architecture Overview

# Reference Architecture

## Summary

Below is the reference architecture of the server in AWS. Note that in the existing implementation, only two availablity zone is implemented for cost savings reasons.



# Networking

## Virtual Private Cloud

#### Definition

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

#### Implementation

The entire platform is hosted in a Virtual Private Cloud in AWS Singapore region. Details as follows:

|  |  |
| --- | --- |
| VPC ID: |  |



### Subnets

The VPC is split into three main subnets across two main availability zones A and B:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Subnet ID** | **Availability Zone** | **Description** |
| Public Subnet 1 |  | ap-southeast-1a | Public subnet for availability zone A |
| Private Subnet 1 | App tier |  | ap-southeast-1a | Private subnet for availability zone A |
| Private Subnet 2 | App tier |  | ap-southeast-1b | Private subnet for availability zone B |

#### A note on public vs private subnets…

*A public subnet* routes 0.0.0.0/0 through an Internet Gateway (igw). Instances in a public subnet require public IPs to talk to the internet.

*A private subnet* sets that route to a NAT instance. Private subnet instances only need a private ip and internet traffic is routed through the NAT in the public subnet. You could also have no route to 0.0.0.0/0 to make it a truly private subnet with no internet access in or out.

### Network Addressable Translation Gateways

The private subnets (private-subnet-1 and private-subnet-2) are routed to a single NAT Gateway instance that sits within public-subnet-01. NAT instances are expensive resources, so we only deployed one. Details as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **NAT Gateway ID** | **Subnet** | **Description** |
| Nat Gateway Public Subnet 1 |  | Public Subnet 1 | Gateway to allow private subnets to reach public networks |

For resources within the private subnets to reach the NAT gateway, the respective route tables have to be configured by point 0.0.0.0/0 to the NAT gateway as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Route Table ID** | **Subnet** | **Description** |
| Private Route Table 1 |  | Public Subnet 1 | Route table that is assigned to Private Subnet 1 & 2 to enable resources to reach internet |

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Platform Implementation Overview

The main components of the platform are implemented in the private subnet (private-subnet-1):

1. Database Server
2. Network File System

## Database Server

The database server uses AWS RDS service which provides a managed PostgreSQL platform, note that the OS of the DB server is not accessible and managed by AWS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **DB Endpoint** | **Availability Zone** | **Description** |
| db |  | ap-southeast-1a  ap-southeast-1b | DB for the platform |

The database is configured to be not accessible to the public internet. Only specific resources such as the bastion host are allowed to access the database – this is controlled by specifying the Security Groups settings. The Security Group for the DB and its settings are as follows:

**Name:** Database Security Group| **Security Group ID:**

Graphical user interface, text, application, email

Description automatically generated

At present, only the Web Server and the bastion host are allowed to connect to the database.

## Network File System

The network file system hosts the web application of the MISP platform and is fulfilled by the Elastic File Storage (EFS) service in AWS. By separating the web application files from the compute server, we are able to spin up and down the compute instances without running into data loss issues.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **EFS ID** | **Subnet** | **Description** |
| EFS |  | Private Subnet 1 Private Subnet 2 | Network file system for the platform |

The EFS mounting points on both availability zones A and B and is a shared resource across all servers.

Graphical user interface, text, application, chat or text message

Description automatically generated

## Security Groups

The following security groups are set up:

Name:

Description:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Ingress | Egress |
| ssh-security-group | Enable SSH access on Port 22 | from\_port = 22  to\_port = 22  protocol = "tcp"  cidr\_blocks = ["${var.ssh-location}"]  //limit to your own IP | from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = [“0.0.0.0/0”] |
| webserver-security-group | Enable HTTP/HTTPS access on Port 22 via SSH SG | from\_port = 22  to\_port = 22  protocol = "tcp"  security\_groups = [ ssh-security-group.id] | from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = [“0.0.0.0/0”] |
| database-security-group | Enable MySQL/Aurora access on Port 3306 | from\_port = 5432  to\_port = 5432  protocol = "tcp"  cidr\_blocks = ["10.0.0.11/32”]  //Private IPv4 address from bastion host | from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = [“0.0.0.0/0”] |
| lambda-security-group | Enable lambda access |  | from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = [“0.0.0.0/0”] |

-END-

## Compute Server

The compute server is fulfilled by the AWS service EC2 – Elastic Computing. Additional background reading materials on EC2 instances can be found here: <https://docs.aws.amazon.com/ec2/index.html>

Details of the EC2 instance used is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Instance Type** | **Subnet** | **Description** |
| misp-ec2-autoscale | t2.medium | misp-subnet-03 | EC2 instance for the MISP server compute |

The MISP EC2 instance belongs to an autoscaling group that allows the compute resources to scale on demand. The autoscaling group is configured to shut down the server every day at 10pm and start it at 7.30am. This is to save on running costs. Details of the autoscaling group are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Launch Template** | **Subnet** | **Description** |
| misp-autoscale-ec2 | misp-ec2-template  lt-0baf5d43026efb55e | misp-subnet-03 | EC2 instance for the MISP server compute |

Scaling Schedule:

Graphical user interface, text, application, Teams

Description automatically generated

Autoscaling groups uses launch templates to launch the image (AMI) of the server. The launch template was created based on the latest working version of the server. Details are as follows (note that changes to the launch templates might break the server, so handle with care):

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Launch Template ID** | **AMI ID** | **Description** |
| misp-ec2-template | lt-0baf5d43026efb55e | ami-0c1f1bb1ca0d80e82 | EC2 instance for the MISP server compute |

## Public Facing Resources

We outlined the key components of the MISP platform in the earlier section. This section will cover the public facing resources that are used expose the MISP platform over the internet. There are 3 key resources:

1. Application Load Balancer
2. Web Application Firewall

### Application Load Balancer

Application Load Balancer operates at the request level (layer 7), routing traffic to targets, in this case the EC2 compute instances based on the content of the request. ALB can handle both HTTP and HTTPS traffic (HTTP is configured to redirect to HTTPS for security reasons). The ALB is configured to operate across both availability zones for redundancy reasons.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Target Group** | **Subnet** | **Description** |
| misp-alb | misp-target-group-https | misp-subnet-03, misp-subnet-04 | Used to handle incoming https request and routes to MISP platform |

The application load balancer contains target groups that conducts health checks on the MISP server over port 443 with status code 200 or 302 indicating the server is healthy.

Graphical user interface, application

Description automatically generated

### Web Application Firewall

The web application firewall (WAF) is deployed in front of the Application Load Balancer (ALB) and filters our malicious traffic based on managed rules provisioned by AWS. At present, these rules are not engaged as there needs to be some level of testing to ensure legitimate traffic does not get blocked. You can find the WAF in Web ACLs Asia Pacific (Singapore) region.

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

To add new rules, got to Rules -> Add rules -> Add managed rule groups

Graphical user interface, text, application

Description automatically generated

Choose AWS managed rules -> Core set rules

You can adjust individual rules to switch on or off accordingly.

Graphical user interface, text

Description automatically generated