Auto Scaling Group Web Application Deployment

This presentation documents the step-by-step process of setting up an EC2 instance, configuring a launch template, Auto Scaling Group (ASG), and ensuring every instance launched by the ASG runs the same web application. We'll cover the entire deployment process from initial network setup to monitoring and troubleshooting.

Members

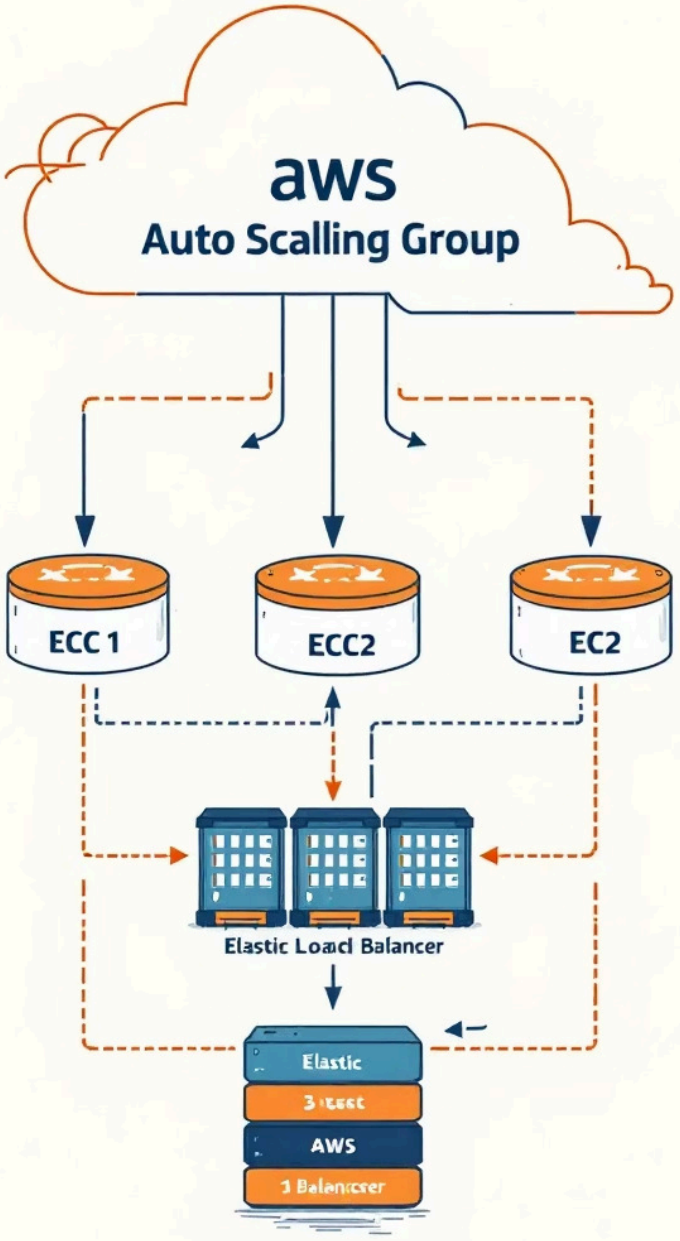
Fred

Mary

Richard

Priscilla

Emmanuel



Initial Network Infrastructure

VPC Creation

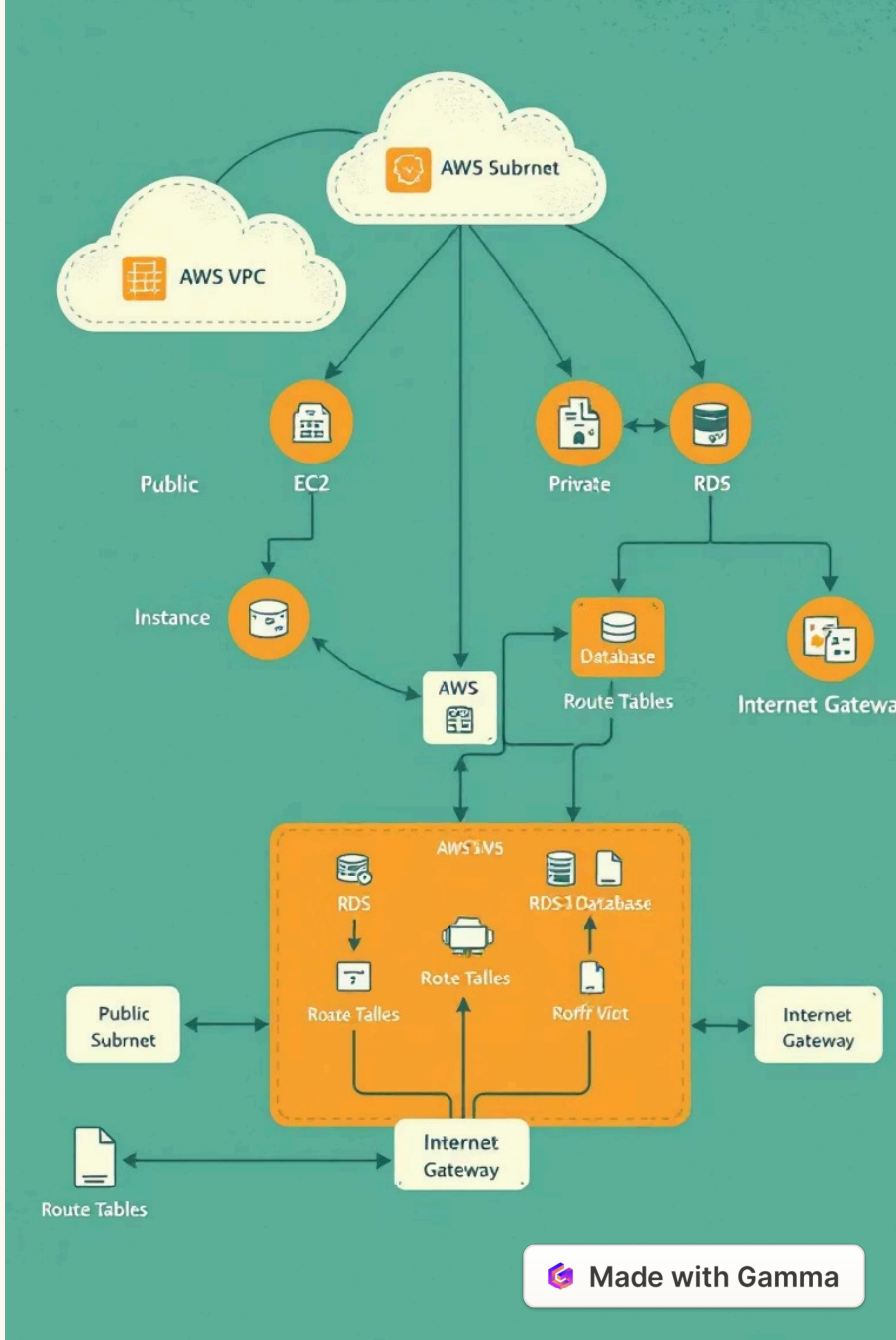
A Virtual Private Cloud (VPC) was created to logically isolate the network for the application infrastructure.

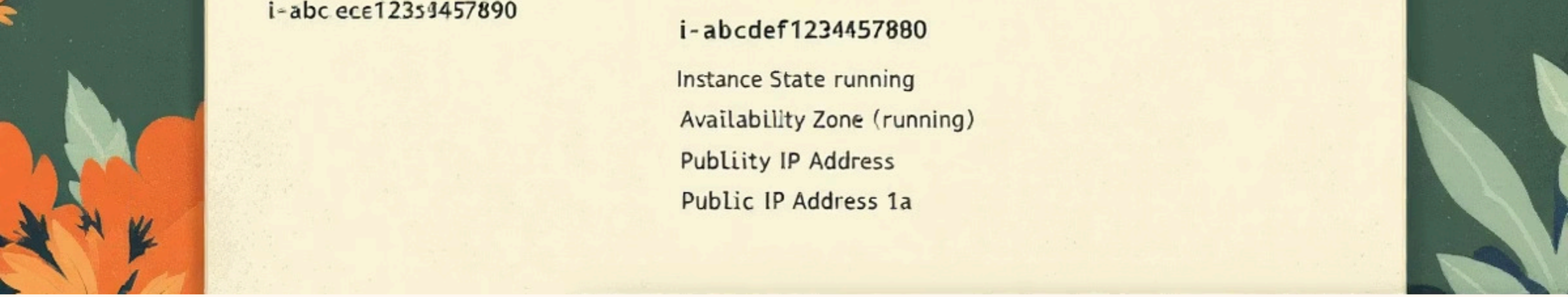
Subnet Configuration

Public subnets for Load Balancer and ASG, and private subnets for EC2 instances and database.

Security Setup

Security Groups configured to allow inbound HTTP (80), HTTPS (443), and SSH (22) access. NACLs set up with matching permissions.





EC2 Instance Launch & Configuration

Launch EC2 Instance

Successfully launched the first EC2 instance where the web application was deployed.

Launch Template Creation

Created a launch template based on the original EC2 instance, including instance type, AMI, and security groups.

Web Application Upload

The application was uploaded manually via AWS CloudShell or EC2 terminal to the appropriate directory.

Auto Scaling Group Setup

Link Template

Connect ASG to launch template

Configure Network

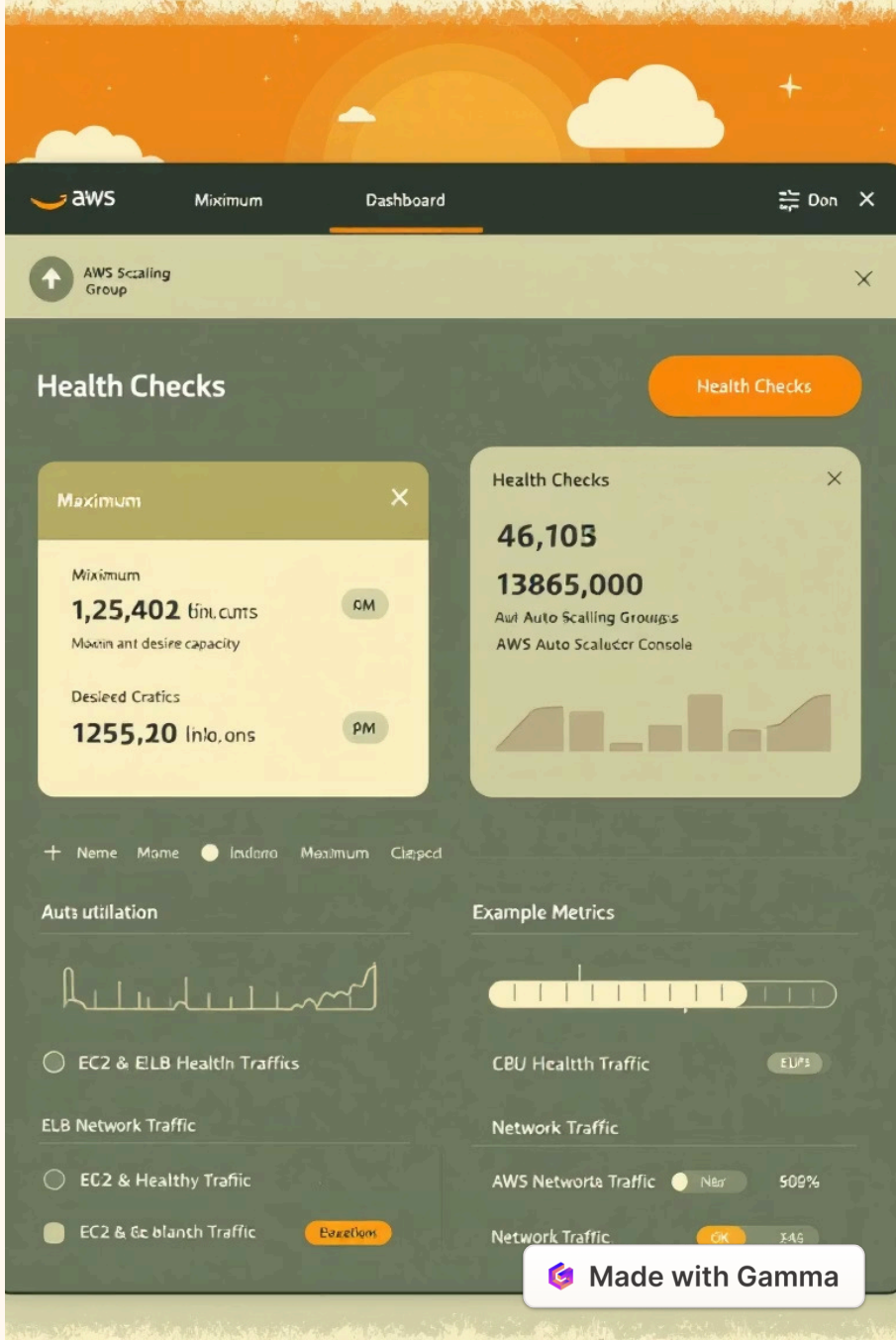
Set availability zones and subnets

Define Capacity

Set minimum, maximum, and desired capacity

Health Checks

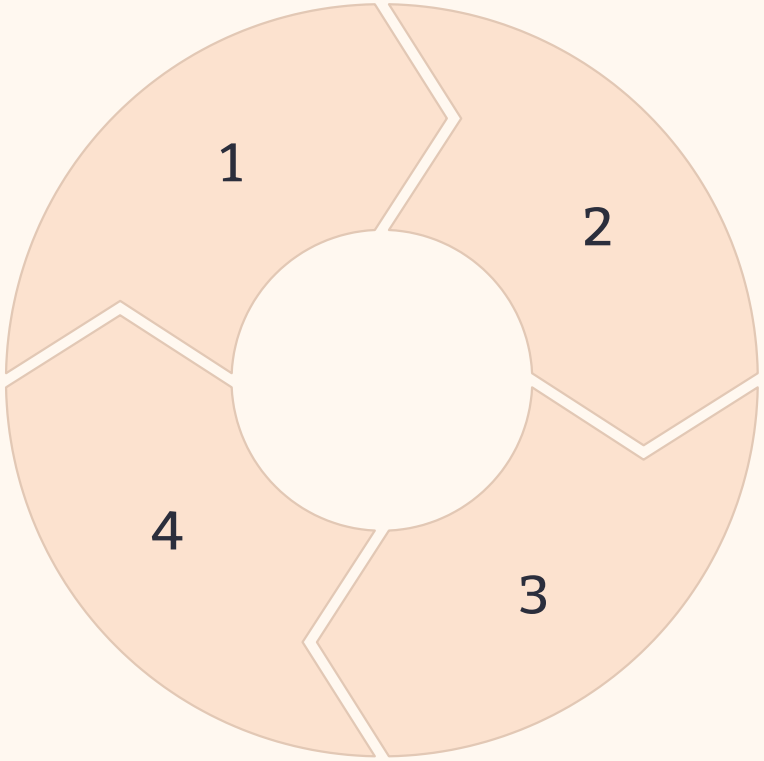
Configure EC2 and ELB health checks



Load Balancer Integration

Create Load Balancer

Application Load Balancer created with internet-facing configuration



Configure Listeners

HTTP listener on port 80 forwarding to target group

Health Checks

Configure path and thresholds for instance health verification

Target Group Setup

Target group linked to Auto Scaling Group

CPU Stress Testing & Scaling

Stress Test Procedure

Load was generated using stress-ng tool with the command:

```
stress-ng --cpu 8 --cpu-method all --verify --timeout 10m
```

This simulated high CPU utilization across multiple cores to trigger the auto-scaling policy.

Scaling Response

When CPU utilization exceeded the threshold (approximately 72-90%), the Auto Scaling Group automatically launched additional instances to handle the increased load.

The number of instances increased from 1 to 3 during peak load periods, demonstrating successful horizontal scaling.

Database Integration

Verify RDS Details

Confirmed RDS endpoint (database-1.cu5emkwymw55.us-east-1.rds.amazonaws.com), port, credentials, and VPC configuration.

Configure Security Groups

Added inbound rule to RDS security group allowing access from EC2 security group on the appropriate database port.

Install Database Client

Installed MariaDB client on EC2 instances to connect to the MySQL-compatible RDS instance.

Test Database Connection

Successfully connected to RDS, created a sample database with users table, and verified data insertion and retrieval.

Monitoring & Logging Setup

CloudWatch Metrics

Enabled detailed monitoring for EC2 instances to track CPU, memory, and network utilization with 1-minute granularity.

CloudWatch Alarms

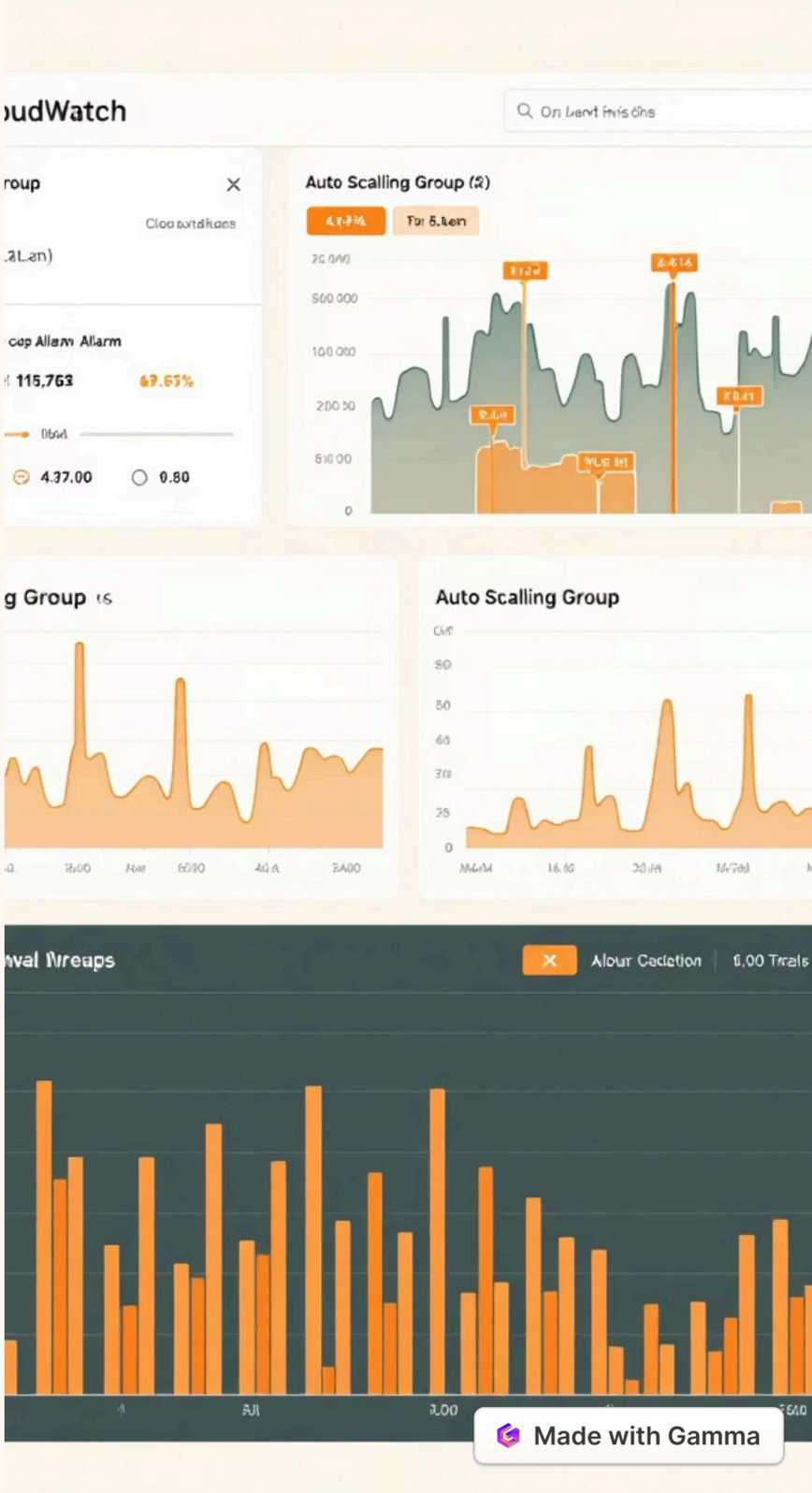
Created alarms for high CPU usage (>50%) and low CPU usage (<35%) to trigger scaling actions and notifications.

VPC Flow Logs

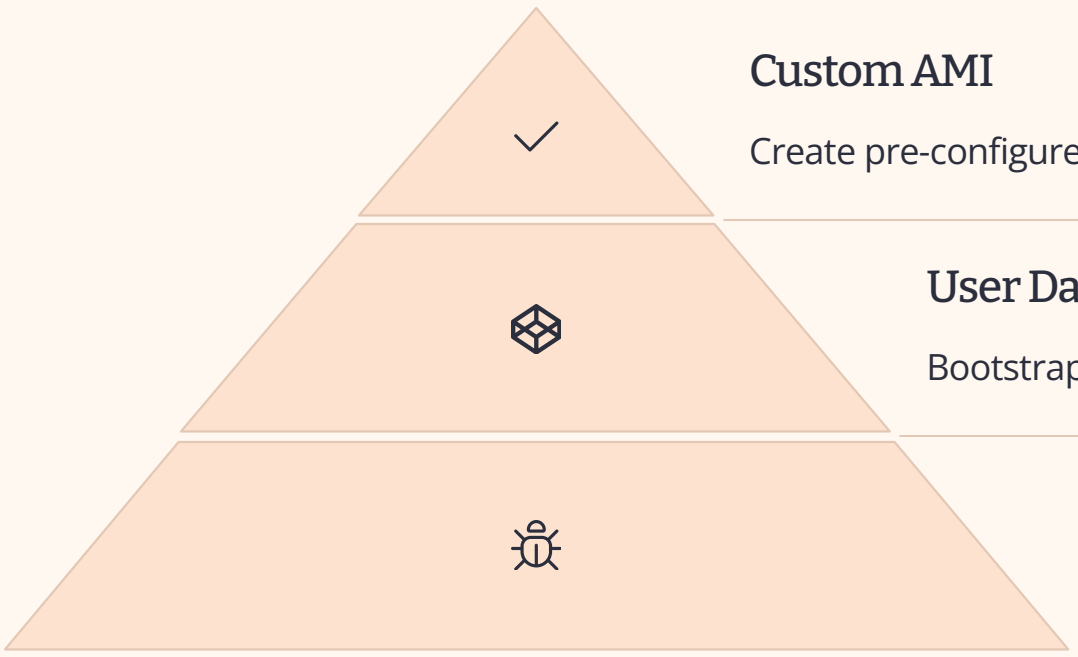
Enabled flow logs to capture information about IP traffic going to and from network interfaces in the VPC.

Log Groups

Created EC2group-Logs log group to centralize application and system logs for troubleshooting and analysis.



Troubleshooting & Solutions



Custom AMI

Create pre-configured image with application installed

User Data Script

Bootstrap instances with installation scripts

Problem Identified

Instances launched by ASG didn't run the web application

We identified that instances launched by the Auto Scaling Group did not automatically run the web application because the launch template lacked application deployment instructions. Two solutions were proposed: creating a custom AMI with the application pre-installed, or adding a user data script to the launch template that installs and configures the application during instance startup.

The architecture successfully demonstrated horizontal scaling in response to load, but requires one of these solutions to ensure consistent application deployment across all instances.