الأكاديمية السعودية الرقمية
SAUDI DIGITAL ACADEMY

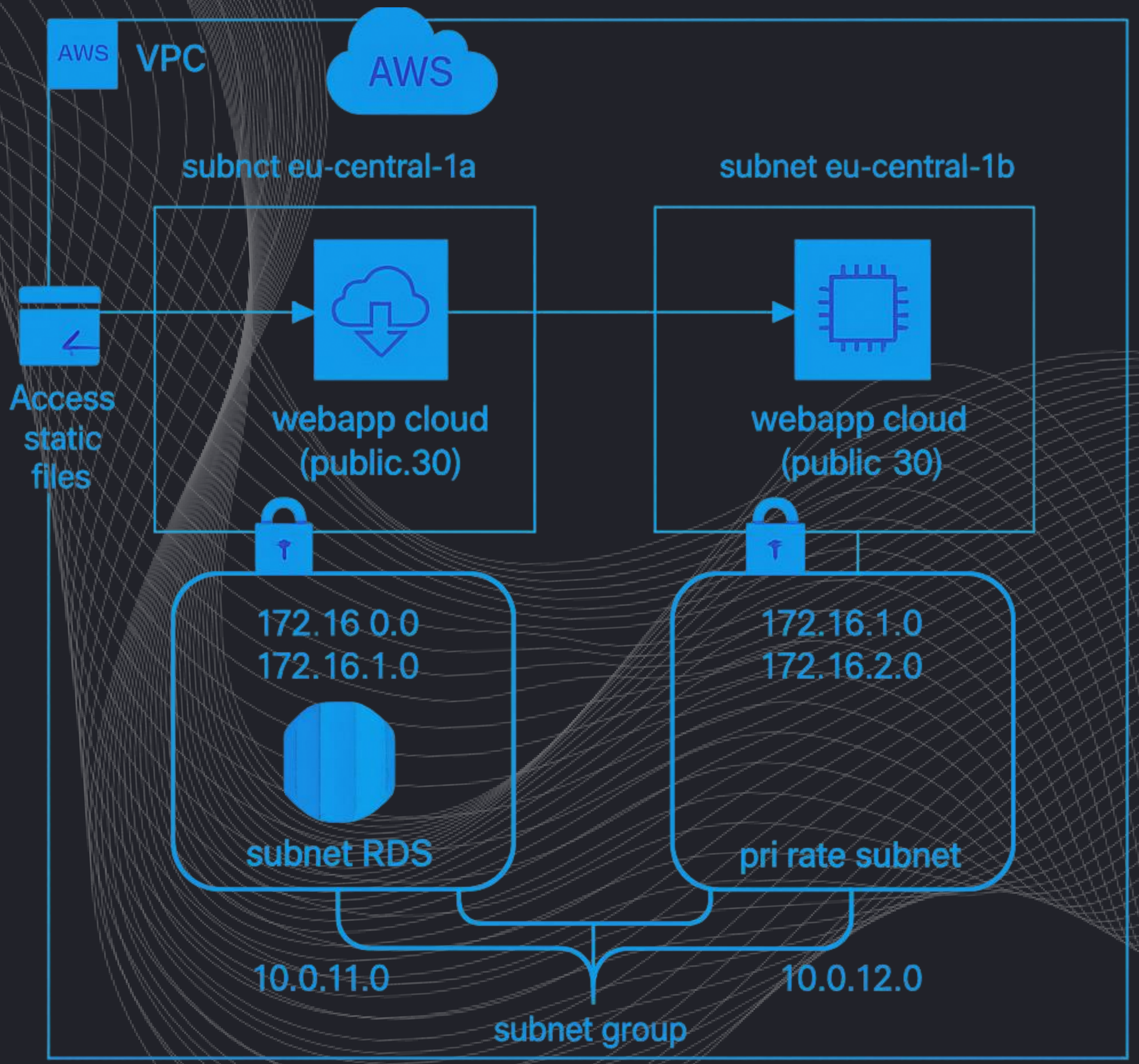
SDA-CAPSTONE – Blog App Deployment

Team 1

Ibrahim-SDA2002 Manal-SDA2020 Shadan-SDA2005 Manal-SDA2006 Bushra-SDA2015 Rawan-SDA2022

Renad-SDA2038 Khalid-SDA2024 Lina-SDA2027 Mafaz-SDA2029 Nawal-SDA2039 Huda-SDA2001

Architecture Overview



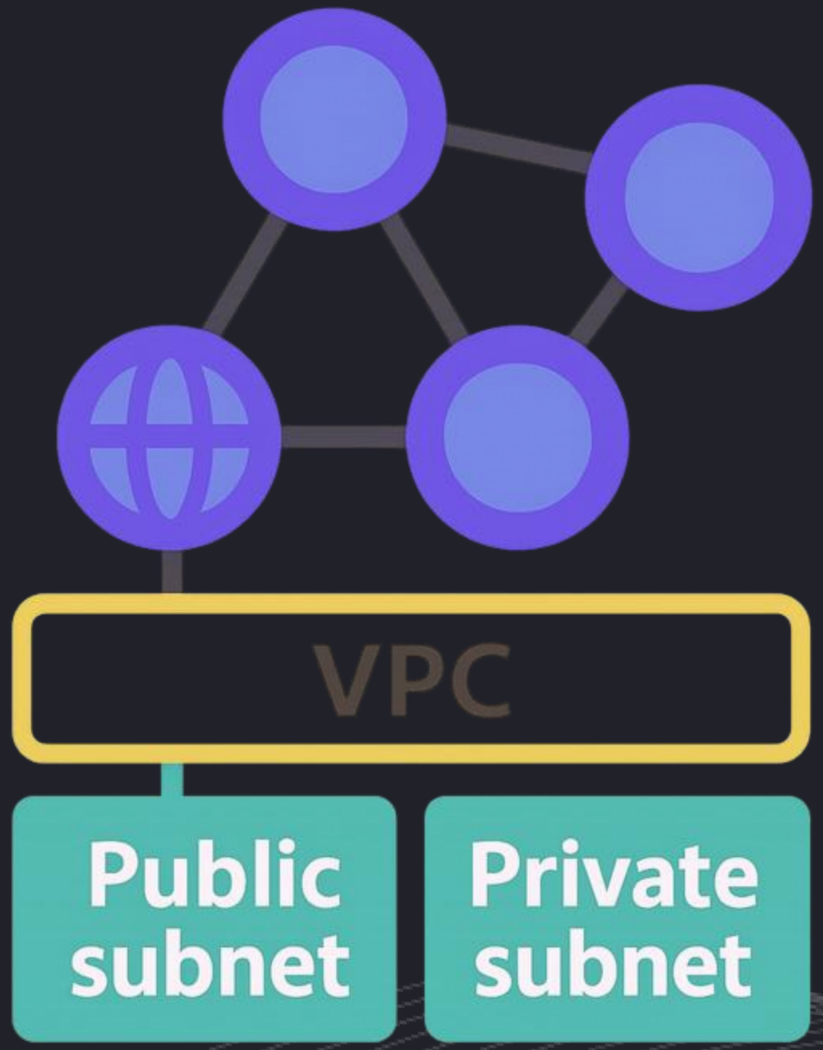
🧱 Overview of Architecture Components

Custom VPC with public and private subnets distributed across two Availability Zones (eu-central-1a & eu-central-1b).

Networking Architecture

What to do :

- Create VPC: 10.90.0.0/16
- 2 Public & 2 Private subnets across 2 AZs
- Enable DNS hostnames



Internet & NAT Gateway Setup

SDA

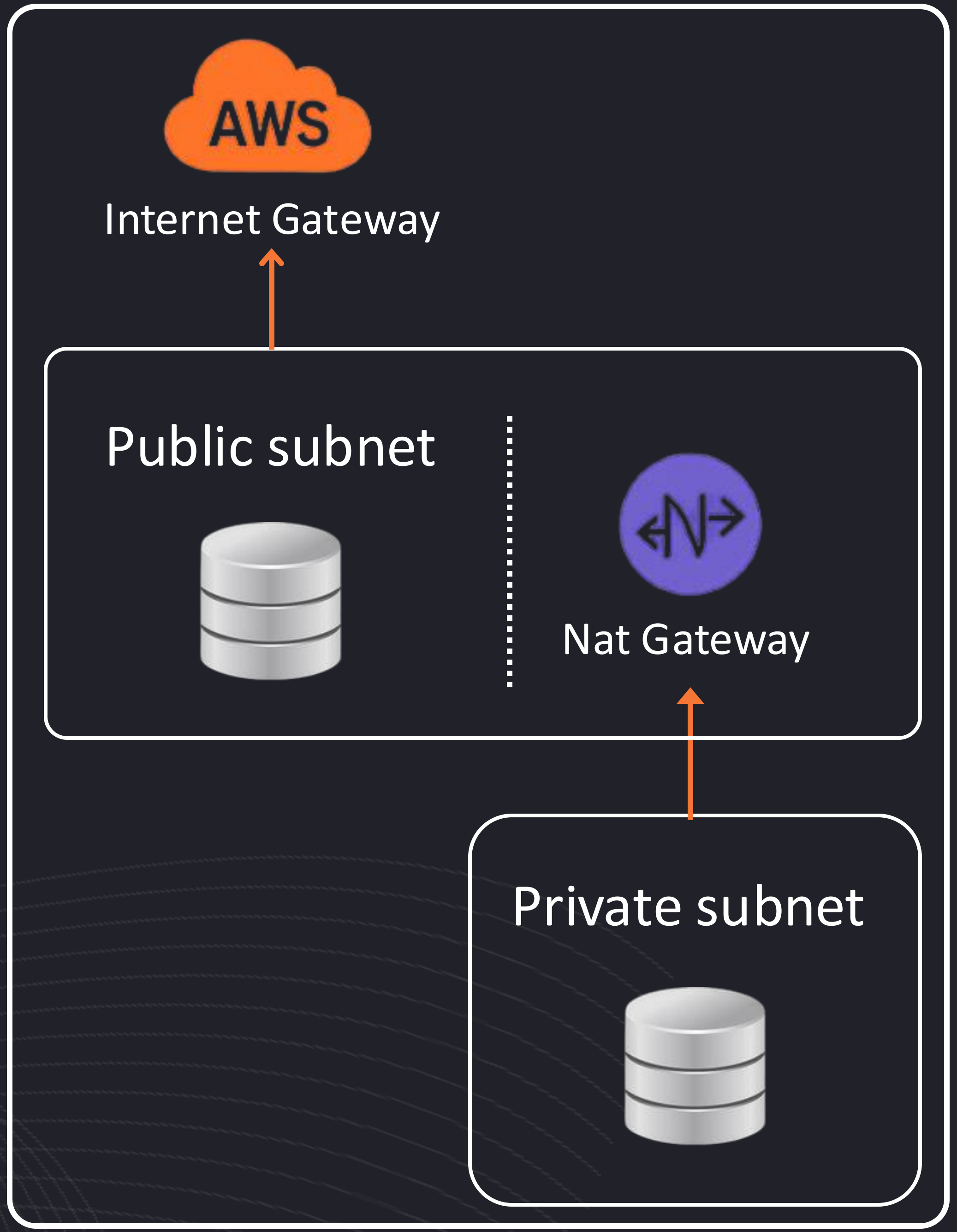
- **Public subnets** are connected to the Internet Gateway (IGW) to allow direct internet access for resources like web servers.

- **Private subnets** route outbound traffic through a NAT Gateway, which enables instances in private subnets to access the internet securely without exposing them to inbound traffic.

- **Route Tables** are configured to direct traffic properly:
  - Public subnets → route via IGW for inbound and outbound internet traffic.
  - Private subnets → route via NAT Gateway for outbound internet access only.

This setup improves security by isolating private resources from direct internet exposure while maintaining their ability to update and communicate externally.

VPC (Virtual Private Cloud)



Security Groups

SDA

ALB SG: HTTP/HTTPS from anywhere

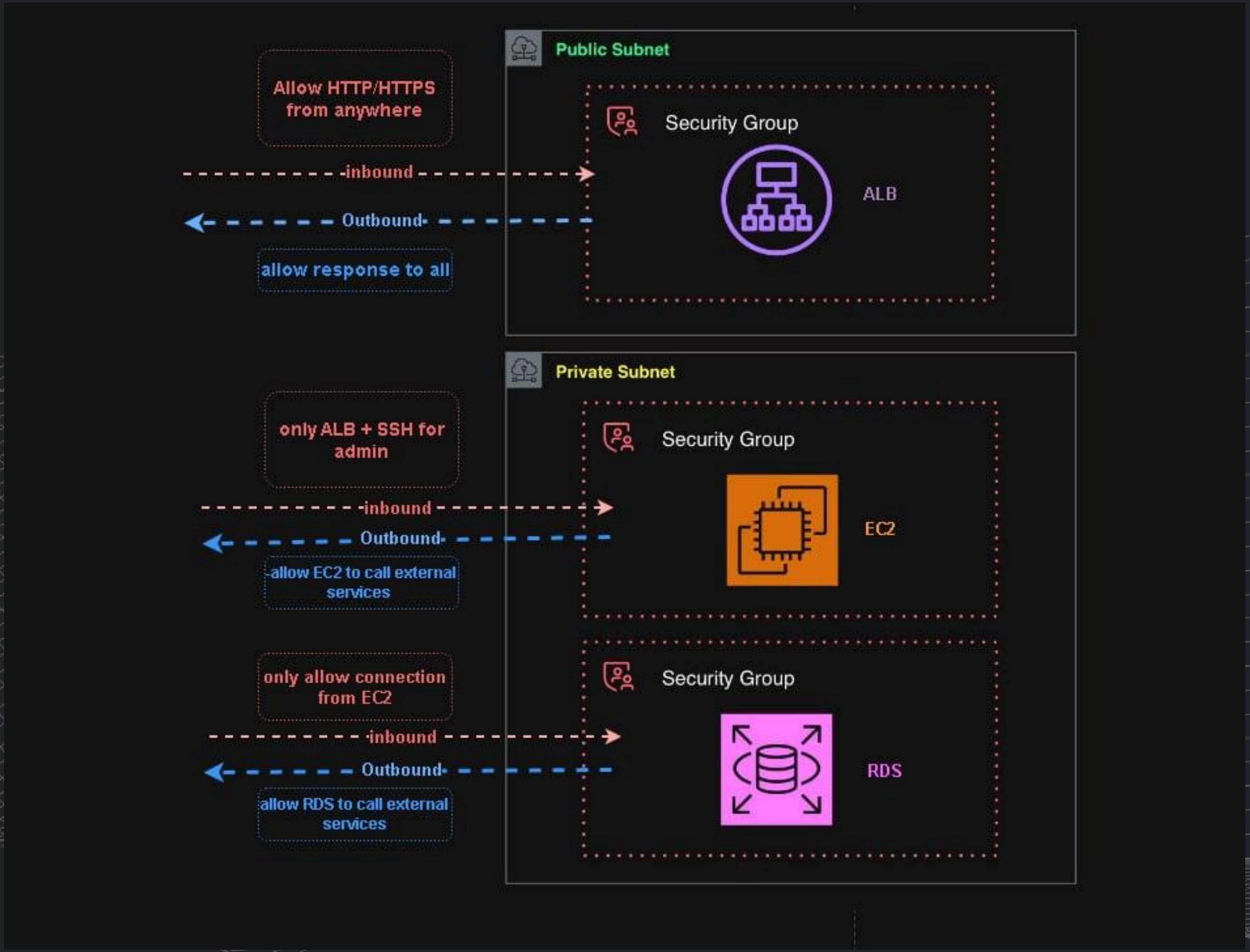
Allows HTTP/HTTPS from anywhere for p public access

EC2 SG: Only ALB + SSH for admin

Allows traffic only from the ALB, plus SSH for admin access.

RDS SG: Only EC2

Allows only EC2 instances to connect to the database.



GitHub Integration

Private GitHub repo for source code

Token-based clone using HTTPS

Automate file changes with github _repository_file in Terraform

Secrets with AWS SSM

Store sensitive values:

/capstone/username

/capstone/password

/capstone/token

These are stored securely in AWS Systems Manager Parameter Store as SecureStrings.

Used by EC2 & Terraform

EC2 Instances:

Use aws ssm get-parameters to fetch secrets at runtime for environment variables or config files.

Terraform:

Fetches SSM parameters using the aws_ssm_parameter data source to inject secrets into infrastructure as code (e.g., for user data or app setup).

RDS Configuration

Private subnet group (2 A2s)

MySQL &.x. db.t4g.micro

Not publicly accessible

Use SSM for DB credentials

S3 Bucket for Media

Bucket : name it as you like

ACL enabled, public access allowed

Stores uploaded images/videos from the app

Up and ready

EC2 Test Instance

EC2 Test Instance :

This EC2 instance was created just for testing purposes.

We launched it inside a public subnet so we could access it easily during setup.

We attached an IAM role that gives it access to:

- S3(to store and retrieve media files),
- SSM(to connect securely without SSH),
- and RDS(our database).

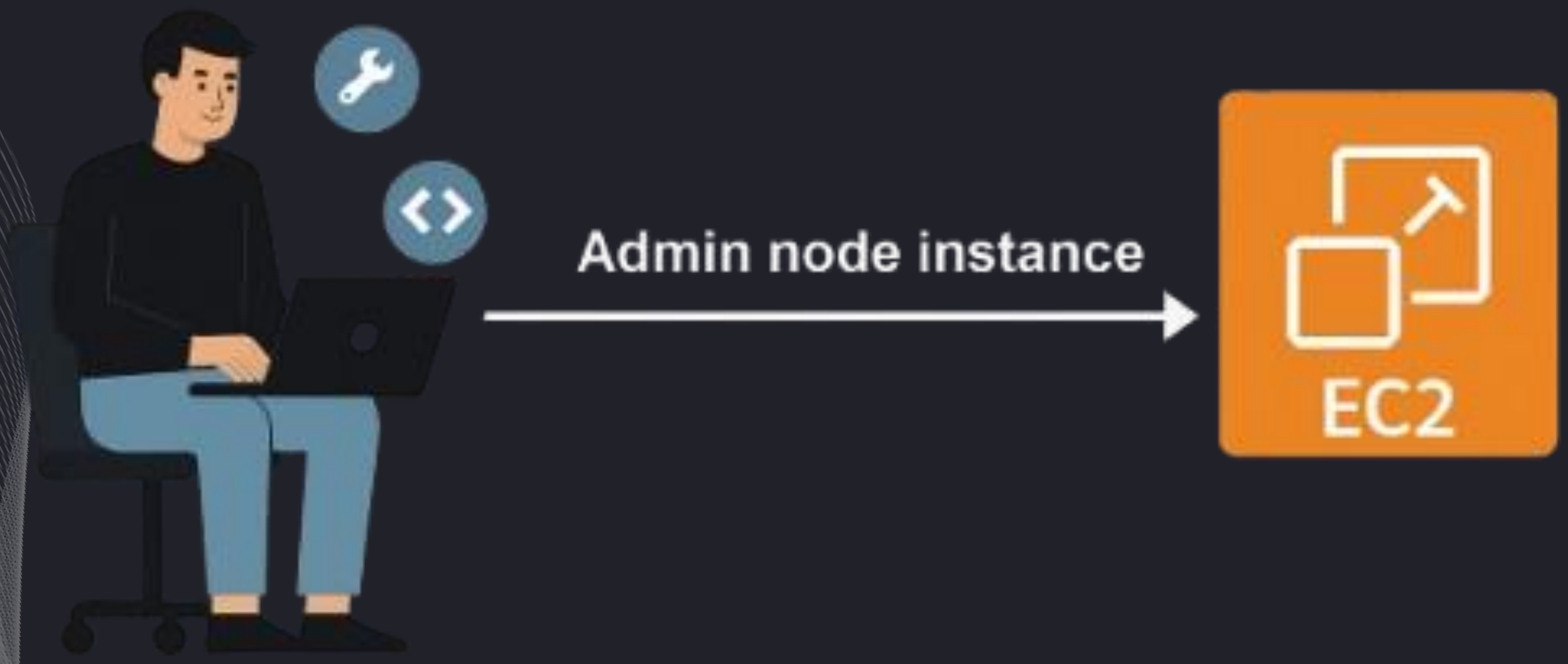
Amazon EC2

After launching the instance, we manually ran the userdata.sh script to:

- test the Django installation,
- check if environment variables work properly,
- and make sure our server setup is complete.

This helped us confirm that everything – like the app, database, and storage – is connected and working before moving to productio

Admin node setup



🔓 Open Ports

22 – SSH

3000 – Grafana

9090 – Prometheus

9100 – Node Export

🔷 **Central EC2 Instance (Amazon Linux)**

Used for:

- **JumpBox – secure entry point for SSH access**
- **Ansible Control Node – configuration management and automation**
- **Monitoring Stack – runs Prometheus and Grafana for system metrics and dashboards**

Terraform Infrastructure

Resources

3

GitHub automation
for settings.py

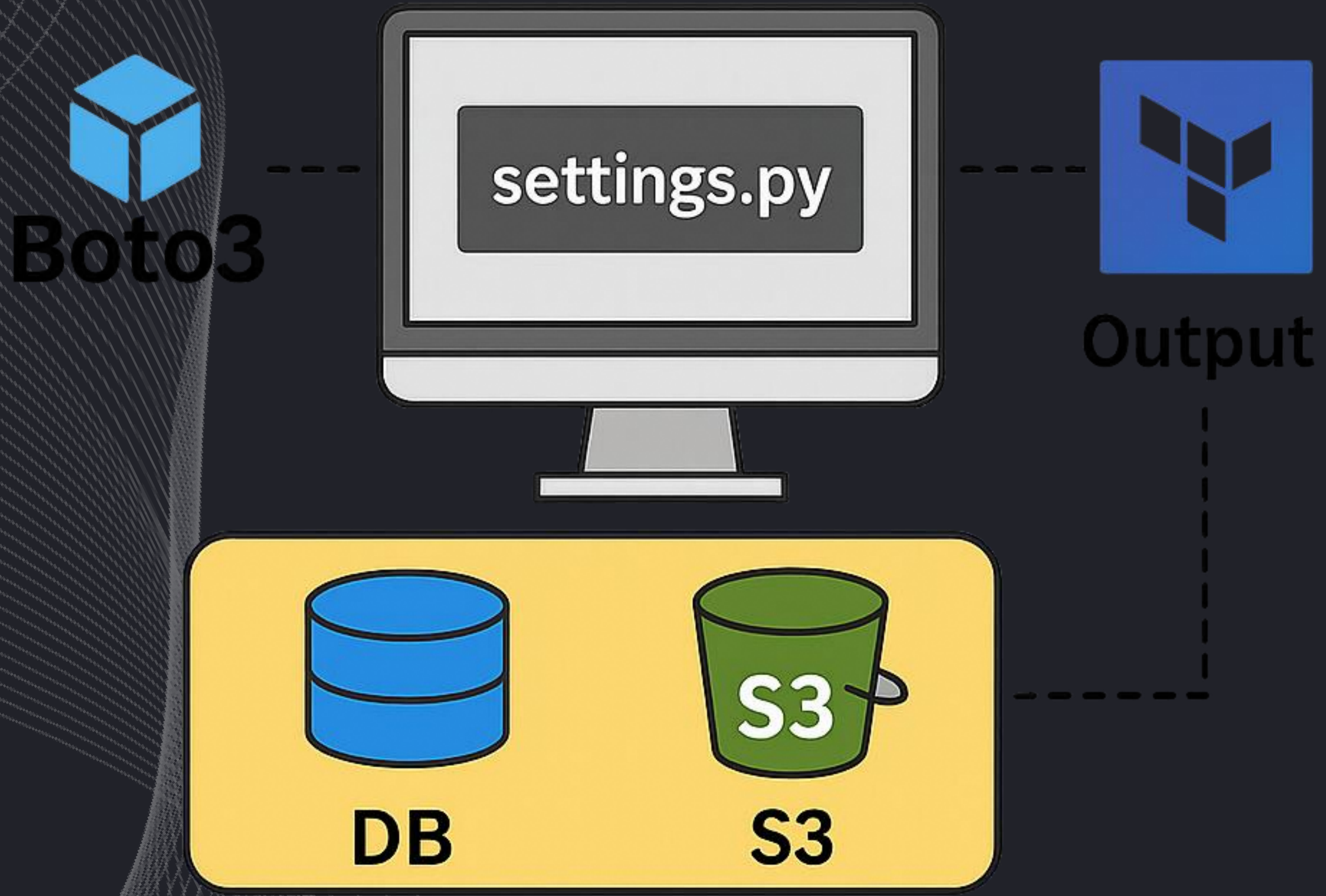
2

IAM Roles, Security
Groups, RDS

1

ALB, Target Group, Launch
Template, ASG

Django settings.py Configuration



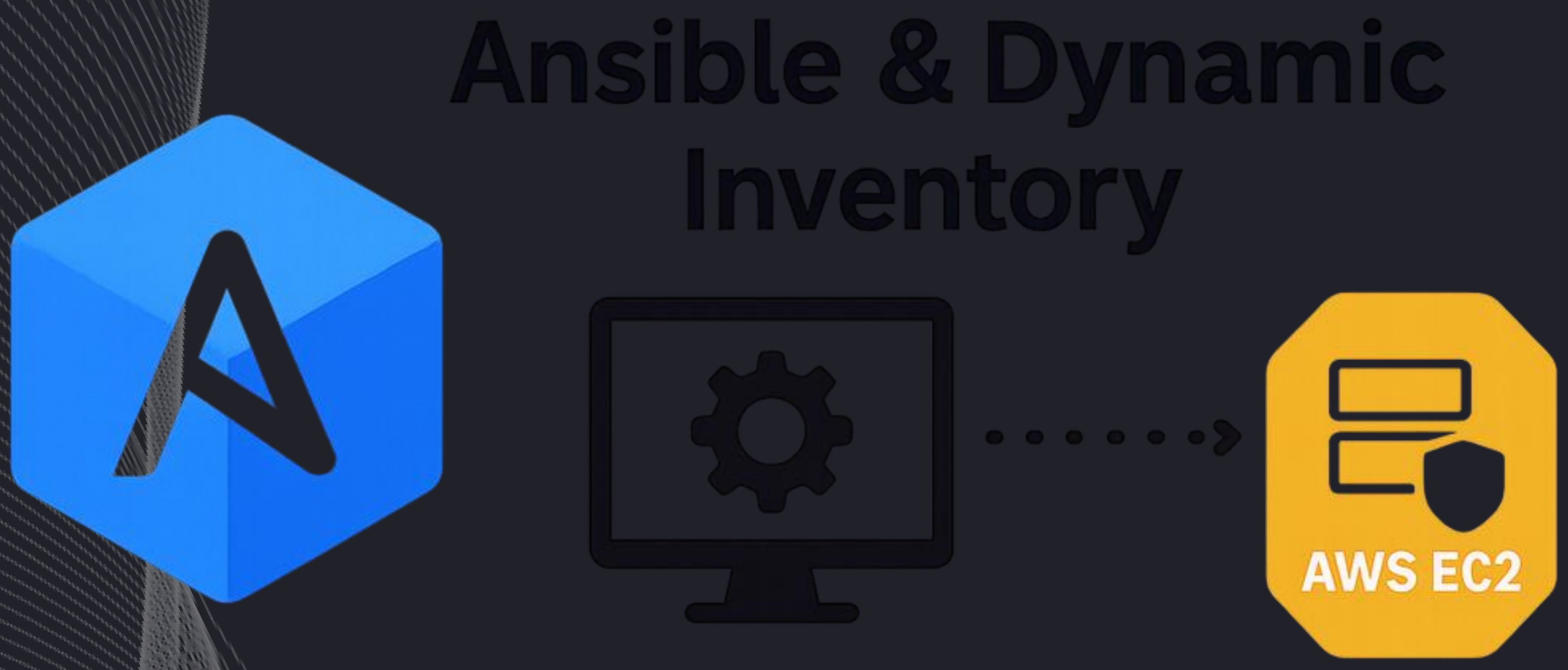
🔧 Key Points for settings.py Configuration

Boto3 is used to fetch secrets (DB credentials, tokens) securely from AWS SSM Parameter Store

Terraform output dynamically injects the DB endpoint into the Django config

S3 and DB credentials are loaded at runtime via environment variables, keeping settings.py clean and secure

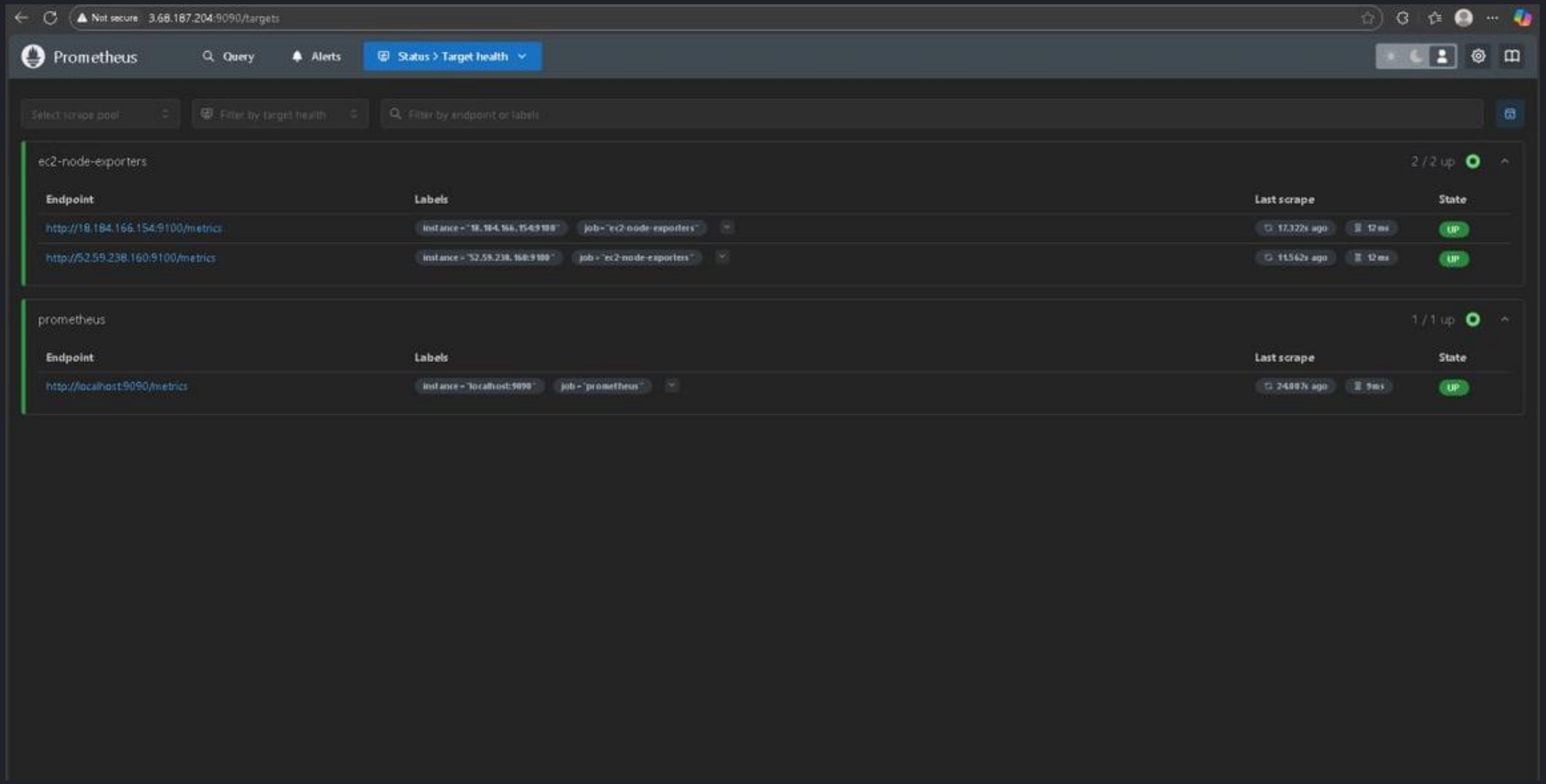
Ansible & Dynamic Inventory



- Use AWS EC2 dynamic inventory plugin for real-time host discovery
- Filter instances by tags:
- Project=Capstone, App=BlogPage
- Set up ansible.cfg and provide the .pem key for secure access

Monitoring with Prometheus

- Deploy Node Exporter on EC2 instances using Ansible
- Enable EC2 service discovery in Prometheus (ec2_sd_configs)
- Monitor key metrics: CPU usage, memory, disk space, and uptime



ec2-node-exporters 2 / 2 up

Endpoint	Labels	Last scrape	State
http://18.184.166.154:9100/metrics	instance="18.184.166.154:9100" job="ec2-node-exporters"	17.322s ago 12ms	UP
http://52.59.238.160:9100/metrics	instance="52.59.238.160:9100" job="ec2-node-exporters"	11.562s ago 12ms	UP

prometheus 1 / 1 up

Endpoint	Labels	Last scrape	State
http://localhost:9090/metrics	instance="localhost:9090" job="prometheus"	24.887s ago 9ms	UP

Grafana dashboard setup:

Log in to Grafana with default credentials: admin / admin

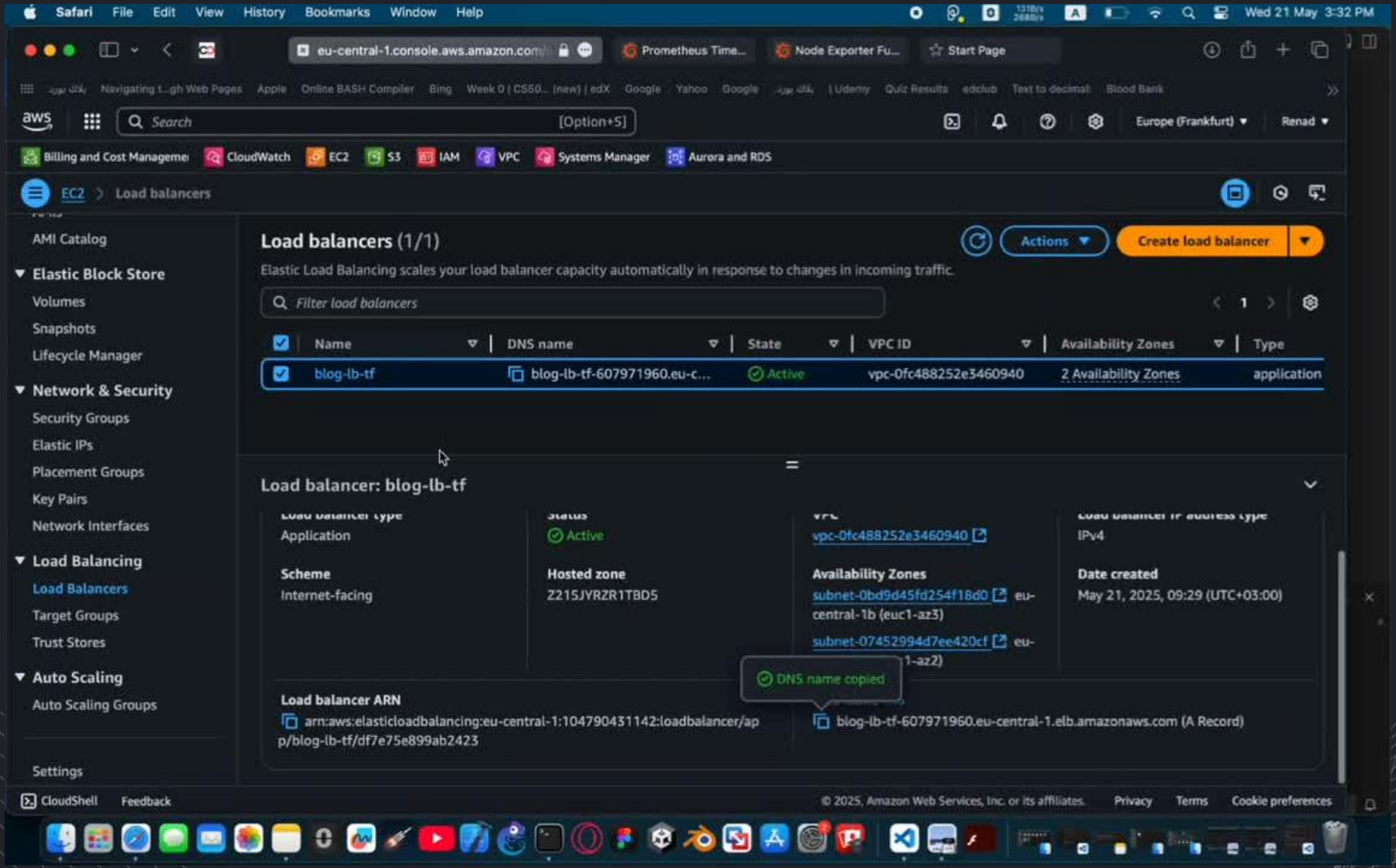
Add Prometheus as a data source via server URL

http://localhost:9090

Import Dashboard ID 1860 – Node Exporter Full for system metrics

Live Application Demo

Time for a live demonstration !!!



Load balancers (1/1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

☑	Name	DNS name	State	VPC ID	Availability Zones	Type
☑	blog-lb-tf	blog-lb-tf-607971960.eu-c...	Active	vpc-0fc488252e3460940	2 Availability Zones	application

Load balancer: blog-lb-tf

Load balancer type
Application

Status
Active

VPC
vpc-0fc488252e3460940

Load balancer IP address type
IPv4

Scheme
Internet-facing

Hosted zone
Z215JYRZR1TBD5

Availability Zones
subnet-0bd9d45fd254f18d0 eu-central-1b (euc1-az3)
subnet-07452994d7ee420cf eu-[illegible]1-az2)

Date created
May 21, 2025, 09:29 (UTC+03:00)

DNS name copied

Load balancer ARN
arn:aws:elasticloadbalancing:eu-central-1:104790431142:loadbalancer/app/blog-lb-tf/df7e75e899ab2423

blog-lb-tf-607971960.eu-central-1.elb.amazonaws.com (A Record)

challenges

- The initial AMI image was incompatible with our region, so we had to search for and configure a suitable one manually.
- Encountered several IAM permission errors that required time-consuming debugging and policy adjustments.
- Some Terraform modules were outdated or misconfigured, leading to deployment failures and resource rollbacks.
- Managing the SSH access for multiple teammates required careful coordination and key sharing.
- File permission issues on the EC2 instance caused problems during Ansible deployment.
- Environment variables were not being picked up properly by the backend service initially, delaying testing.

Thank You

GitHub Repositor:

https://github.com/NawalSuf/infrastructure-capstone-team1/blob/main/README.md