**Cloud Engineer Challenge: Hybrid Infrastructure & Automation**

**Project Overview**

This solution implements a \*\*hybrid cloud migration\*\* of a legacy Java web application from on-premise to AWS using \*\*Terraform IaC\*\*. It includes:

- Simulated on-premise Ubuntu server with Apache, Java, MySQL

- AWS VPC with public/private subnets, NAT Gateway, ALB, EC2 (Tomcat), RDS MySQL

- Secure network segmentation, IAM roles, and backup/DR strategy

- Hybrid connectivity via \*\*IPsec VPN\*\* (conceptual)

Designed for \*\*resilience, security, and cost-efficiency\*\* using AWS best practices.

Github Link : https://github.com/UthayakumarRakav/cloud-engineer-challenge

**Architecture Diagram (Mermaid)**

```mermaid

flowchart LR

subgraph On-Premise

A[Ubuntu Server 192.168.10.10] -->|IPsec VPN| B(VPC: 10.0.0.0/16)

end

subgraph AWS

B --> C[Public Subnet 1 ALB + Web EC2]

B --> D[Public Subnet 2 Web EC2]

B --> E[Private Subnet 1 RDS MySQL]

B --> F[Private Subnet 2 - DR-ready]

C -->|HTTP| D

C -->|3306| E

D -->|3306| E

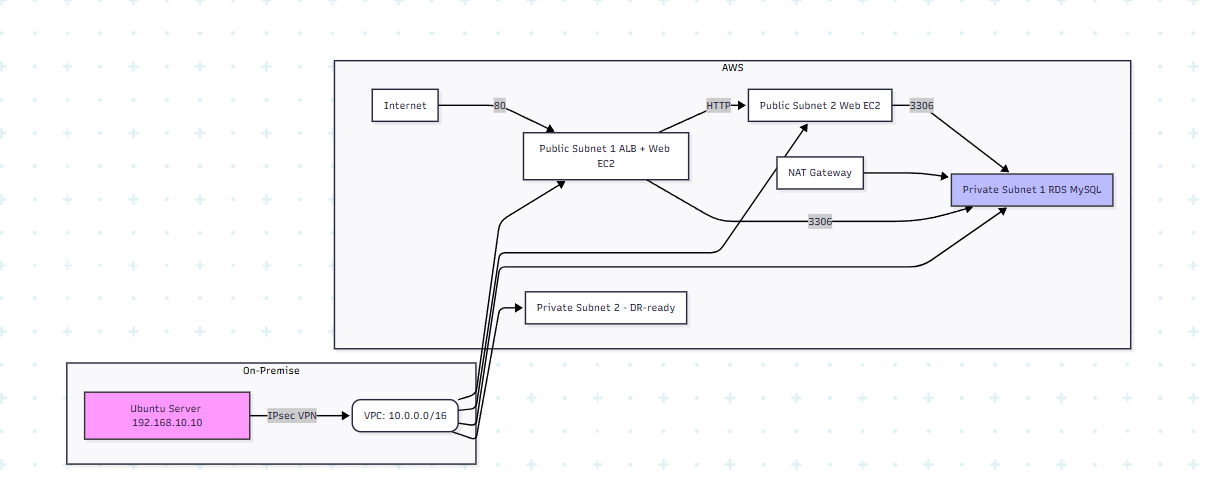
G[NAT Gateway] --> E

H[Internet] -->|80| C

end

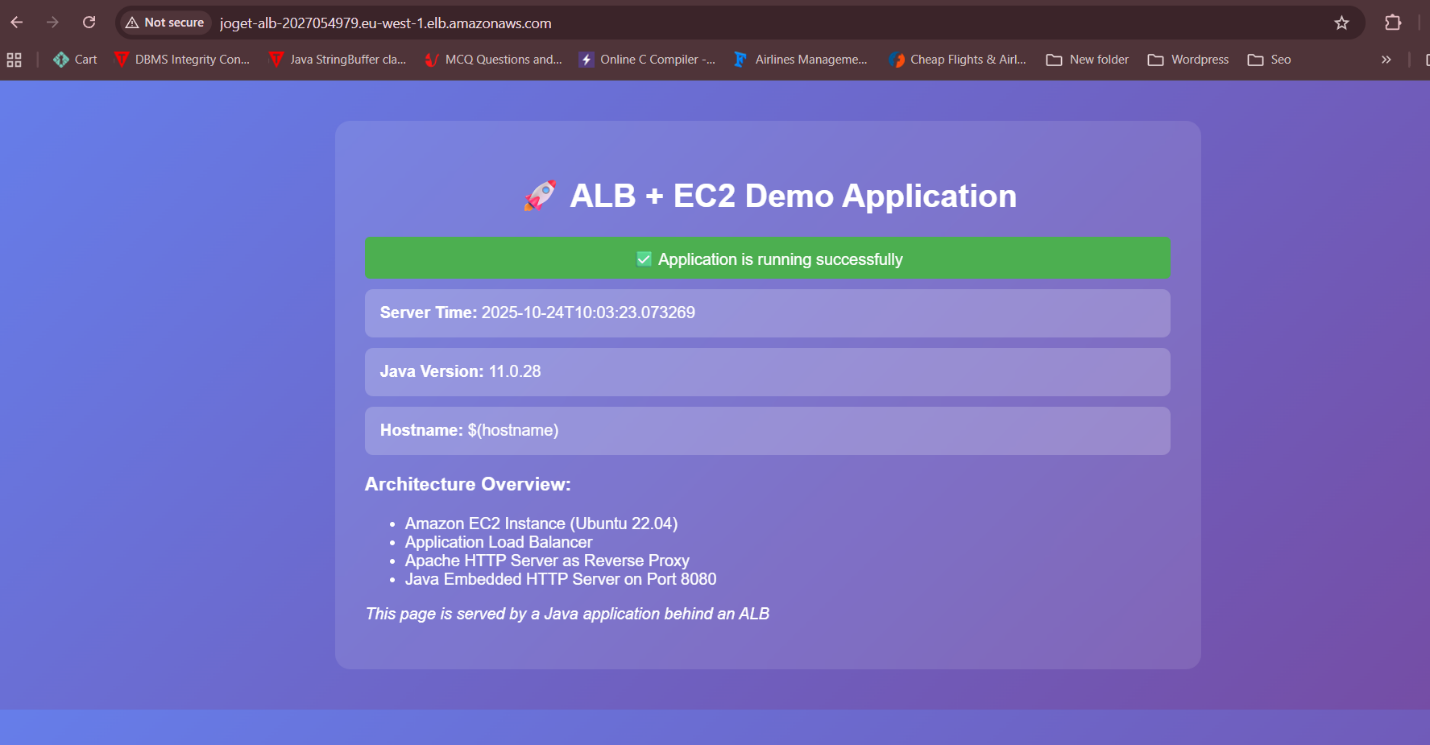
style A fill:#f9f,stroke:#333

style E fill:#bbf,stroke:#333



**1. ALB Serving Tomcat App**

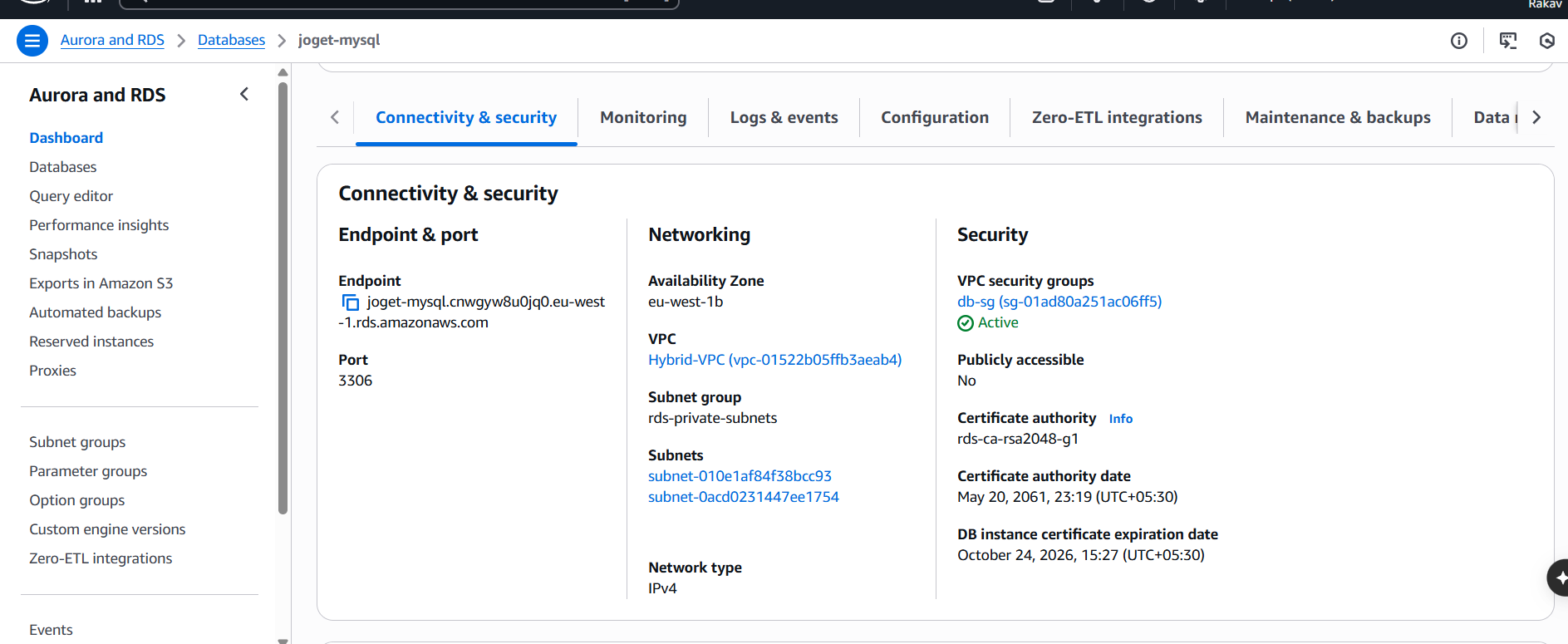
ALB successfully routes traffic to Apache Tomcat running on EC2 instances. Page loads after ~2 minutes.



**2. RDS Configuration (Private & Secure)**

Confirmed:

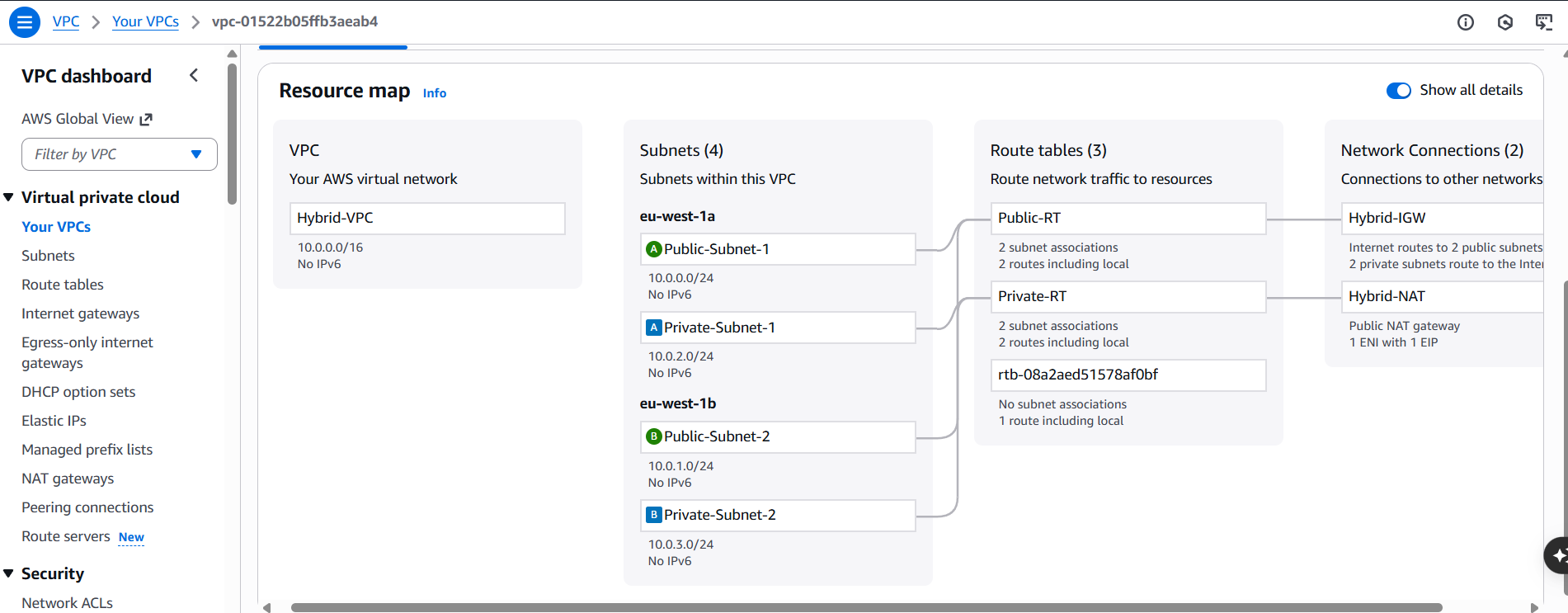
* Publicly accessible: No
* Subnet group: Private subnets only
* Security group: Only allows 3306 from web-sg



**3. AWS VPC Resource Map**

Validated:

* 2 public subnets (with ALB & EC2)
* 2 private subnets (with RDS)
* NAT Gateway in public subnet
* Internet Gateway attached



**Deployment Instructions**

**Prerequisites**

* AWS CLI configured (aws configure)
* Terraform v1.5+
* GitHub repo cloned

cd terraform

# Create terraform.tfvars (do NOT commit!)

echo 'db\_password = "YourSecureRDSPassword123!"' > terraform.tfvars

terraform init

terraform apply -auto-approve

Destroy Resources

cd terraform

terraform destroy -auto-approve

**On-Premise Simulation**

* Simulated via onprem/setup\_onprem.sh
* IP: 192.168.10.10/24
* Services: Apache (80), MySQL (3306), SSH
* Firewall: UFW allows 80, 3306 (temp), SSH
* Hybrid connectivity: Uses IPsec VPN (strongSwan) to AW

**Security Design**

**Resource Inbound Rules**

|  |  |
| --- | --- |
| **ALB** | **HTTP (80) from 0.0.0.0/0** |
| **Web EC2** | **HTTP (80) from ALB SG only** |
| **RDS** | **MySQL (3306) from Web SG only** |

Implemented:

* Secrets: RDS password via Terraform vars (use AWS Secrets Manager in production)
* Encryption: RDS encrypted at rest (AWS default)
* Least Privilege: EC2 IAM role has minimal permissions
* No public exposure: RDS and private subnets are internet-isolated

**Backup & Disaster Recovery**

**RDS**

* Automated backups: Enabled (backup\_retention\_period = 7)
* Cross-region DR: Use AWS Backup to copy snapshots to secondary region (e.g., eu-west-2)

**On-Premise**

* Daily encrypted mysqldump → uploaded to S3:

mysqldump -u root -p legacy\_app | gzip > backup.sql.gz

aws s3 cp backup.sql.gz s3://onprem-backups-$(date +%F)/  
  
**Monitoring**

* EC2: CPU > 80% for 5 min → SNS alert
* ALB: HTTP 5xx errors > 10/min
* RDS: FreeStorageSpace < 10 GB

**Change & Configuration Management**

* IaC Drift: Enforced via CI pipeline running terraform plan
* Secrets Management: In production, store DB credentials in AWS Secrets Manager and inject via IAM roles
* App Deployment: Future state would use Ansible or SSM for config management

**Justifications & Trade-offs**

* Terraform over CloudFormation: Better modularity, state management, and readability
* Public Subnets for Web Tier: Simplifies POC; production would place app servers in private subnets behind ALB
* Single-AZ RDS: Cost-saving for assessment; real workload would use Multi-AZ
* IPsec VPN: Lower cost than Direct Connect for temporary hybrid phase

**Assumption**

* On-premise has static public IP for VPN tunnel
* No legacy code changes required during migration
* AWS region: eu-west-1

**Challenges Faced & solutions**

* Challenge: Joget installer URLs returned 404, causing deployment failure.  
  Solution: Switched to Apache Tomcat from Ubuntu’s official repository, which includes sample Java web apps and requires no external downloads. This ensures 100% reproducibility and reliability.
* Challenge: ALB health checks initially failed.  
  Solution: Validated Tomcat responds on / and adjusted Apache proxy configuration.

**Future Improvements**

1. Move web tier to private subnets; ALB in public
2. Replace EC2 with ECS/Fargate for containerized deployment
3. Implement AWS Systems Manager for secure, keyless access
4. Add WAF + Shield for DDoS protection
5. Automate DR with Route 53 failover routing