

Production Environment: Infrastructure & Deployment Overview

Project: Wisling: V2

1. Overview

The Wisling production environment is hosted on Amazon Web Services and is designed to support a stable, secure, and automated production workload. The infrastructure focuses on minimizing manual operations, protecting data, and ensuring the client is notified proactively in case of any performance or availability issues.

All resources are deployed in a single AWS region to reduce complexity, improve performance, and simplify maintenance.

2. AWS Region and Availability

The entire infrastructure is deployed in the **eu-north-1** region. The database is placed specifically in **eu-north-1a**, ensuring predictable availability and consistent performance by keeping compute and database resources geographically close.

3. Compute Layer (EC2)

A single EC2 instance named **wisling-prod** is used as the main application server. It runs on a **t3.medium** instance type with **4 GB RAM** and a **40 GB EBS volume**, which is suitable for running containerized frontend and backend services in production.

The server runs **Ubuntu** and is accessed securely via **SSH on port 2222** using key-based authentication. Docker and Docker Compose are installed on the instance, allowing the application to run in isolated containers.

To protect data, the EC2 volume is configured to be **retained if the instance is accidentally deleted**, ensuring that application data and configuration can be recovered easily.

4. Networking, Domain, and SSL

The EC2 instance is deployed in a **public subnet**, allowing it to receive internet traffic. An **Nginx reverse proxy** is configured on the server to manage incoming requests and route them internally to the correct services.

The frontend application runs on **port 5177**, while the backend API runs on **port 8082**. Nginx exposes these services through a single domain, hiding internal ports from the end users.

A custom domain is attached to the server, and **SSL/TLS is configured using Certbot (Let's Encrypt)**. This ensures all traffic is served over HTTPS, protecting user data and improving security and trust.

5. Database Layer (RDS MySQL)

The application uses a **private Amazon RDS MySQL database** named **wisling-prod**, running on a **db.t3.medium** instance. The database is not publicly accessible and is isolated within AWS networking.

Access to the database is strictly controlled. Only the EC2 instance's security group is allowed to connect to the database on **port 3306**, which prevents any external or unauthorized access.

To avoid accidental data loss, **deletion protection is enabled** on the RDS instance.

6. Backups and Data Protection

Automated backups are configured for the RDS database. A snapshot is created **every day at 7:45 (UTC +05:00)**, and these backups are retained for **15 days**.

This backup strategy allows the database to be restored in case of data corruption, accidental deletion, or application issues, while keeping storage usage under control.

7. Monitoring and Alerts

Monitoring is implemented using **Amazon CloudWatch** for both the EC2 instance and the RDS database. CloudWatch alarms are configured to trigger when resource usage exceeds **80 percent**. When an alarm is triggered, a notification is sent through **Amazon SNS** directly to the client's email address. This allows the client to be informed immediately about potential performance or capacity issues.

CloudWatch Monitoring

Monitoring is enabled for both **EC2** and **RDS**.

- Metrics tracked:
 - CPU utilization
 - Memory usage
 - Disk usage

Alerts

- Alarms trigger when usage exceeds **80%**.
- Alerts are sent via **SNS email** directly to the client.

This provides early warnings before performance issues affect users.

8. Application Deployment and Runtime

The application runs entirely using Docker containers. All deployment-related files, including `docker-compose.yml` and the environment configuration, are stored in the **/app directory inside the home directory** on the EC2 server.

Docker Compose is used to start and manage both frontend and backend services together. This approach ensures consistent deployments and makes it easier to manage updates and restarts.

9. CI/CD Pipeline (GitHub Actions)

A fully automated CI/CD pipeline is implemented using **GitHub Actions**. The pipeline is triggered from the **main branch**, which represents the production environment.

Authentication between GitHub and AWS is handled using **OIDC**, meaning no long-term AWS credentials are stored in GitHub. Instead, short-lived, secure credentials are generated during each pipeline run.

During the pipeline execution, Docker images for both the frontend and backend are built and pushed to private Amazon ECR repositories named **wisling-frontend** and **wisling-backend**. The EC2 instance then pulls the latest images from ECR and deploys them using Docker Compose.

As part of every deployment, **Laravel Artisan database migrations** are executed against the RDS database. This ensures the database schema is always aligned with the application code.

10. Secrets and Configuration Management

All sensitive configuration values are stored securely in **GitHub Secrets**. These secrets are not committed to the codebase.

During deployment, a **.env file is generated dynamically** on the EC2 server using these secrets. This ensures sensitive data remains protected while still being available to the application at runtime.

11. Supervisor Configuration for Wisling Queue Worker

To manage long-running background tasks, I've configured **Supervisor** on the server. Supervisor is a process control system that ensures critical background services like queue workers continue to run even if they crash.

1. Queue Worker Script:

- The script to manage the queue worker is located at:
`/usr/local/bin/wisling-queue-worker.sh`
This script runs the necessary commands to start the queue worker process that handles tasks such as job processing.

2. Supervisor Configuration:

- The configuration file for Supervisor is located at:
`/etc/supervisor/conf.d/wisling-queue-worker.conf`
This file specifies how the queue worker process should be managed by Supervisor. It ensures that if the worker crashes or stops, Supervisor will automatically restart it, maintaining the reliability of the application.

3. Automated Cron Job:

- Additionally, a cron job is set up to run Laravel's **scheduled tasks** using the command:
`php artisan schedule:run >> /dev/null 2>&1`
This ensures that scheduled tasks defined in the Laravel application (like sending emails, clearing cache, etc.) run automatically at their set intervals.

4. Automated Database Migrations in CI/CD Pipeline:

- As part of the CI/CD pipeline, **database migrations** are automatically triggered during every deployment. This ensures that the production database is always in sync with the application's schema without requiring manual intervention.

These configurations work together to ensure that the backend tasks run smoothly in production without manual interference, and any issues with long-running processes are handled automatically.

12. Reliability & Production Readiness

- Private database with strict access control
- Automated backups with retention
- Deletion protection for critical resources
- Persistent EC2 storage
- Real-time monitoring and alerts
- Secure CI/CD pipeline

Together, these measures make the infrastructure stable, secure, and production-ready.

13. Summary

The Wisling production infrastructure is designed with real-world production needs in mind. It includes secure networking, private databases, automated backups, monitoring and alerts, and a fully automated deployment pipeline.

This setup reduces operational risk, improves security, and ensures the application can be maintained and scaled reliably over time.