12/6/2025

CLOCKLY

AWS-Based Employee Attendance System

NGUYEN Auguste Duc-Liem Kim
DURAES VALADARES David

Contents

CLOCKLY – AWS-Based Employee Attendance System

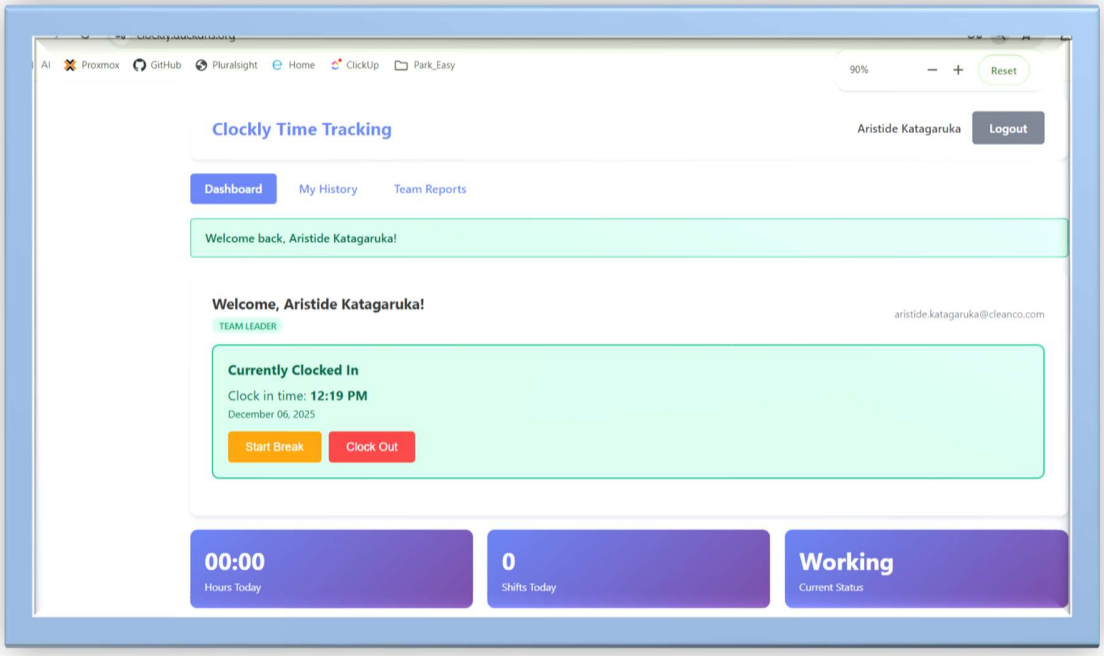
1. Project Scenario Description

1.1 Project Name

CLOCKLY – AWS-Based Employee Attendance System

1.2 Scenario Summary

Clockly is a cloud-based employee attendance tracking system designed for small and medium-sized companies. It allows employees to clock in/out via a web application hosted on AWS. The project demonstrates real-world cloud deployment, automated reporting, authentication, monitoring, and secure data storage.



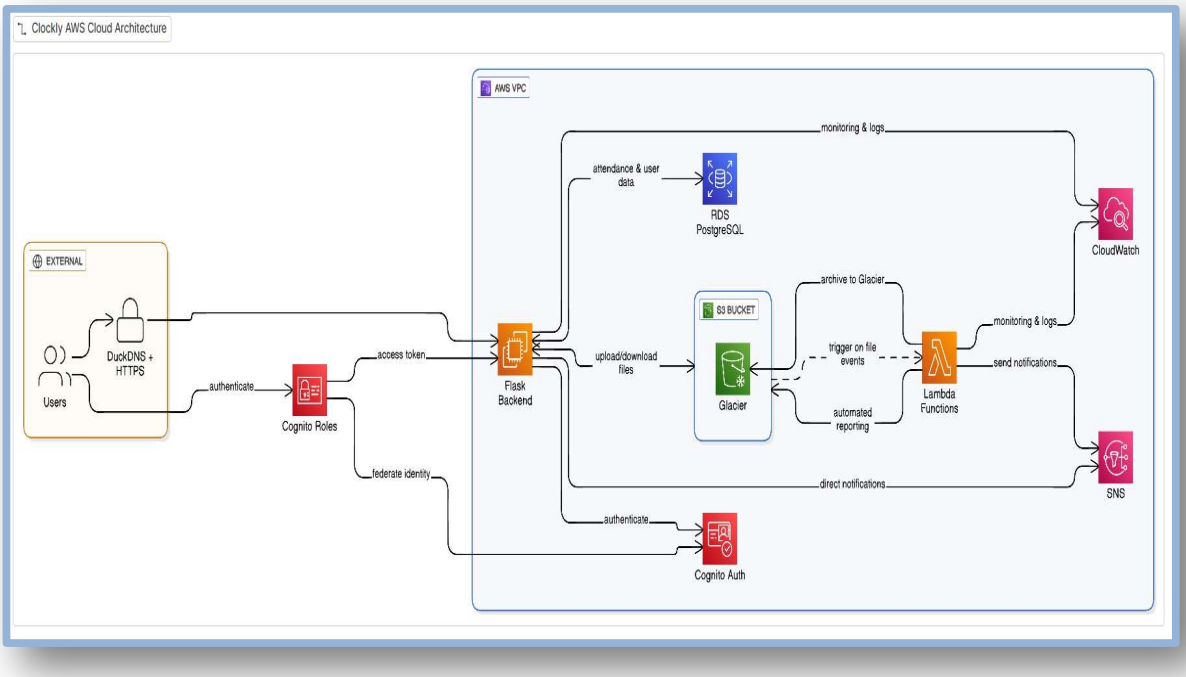
2. System Architecture

2.1 Architecture Overview

Clockly consists of an EC2-hosted Flask backend, Cognito authentication, RDS PostgreSQL database, Lambda automation functions, S3 storage, and SNS notifications, all operating inside a secured AWS VPC.

The application uses:

- AWS Cognito for user authentication
- AWS RDS (PostgreSQL) for storing attendance and user data
- AWS S3 for storing employee attendance files and images
- AWS Glacier for long-term archival
- AWS SNS for notifications
- AWS Lambda for automated reporting and S3 archival
- AWS CloudWatch for monitoring and alerting
- AWS IAM for secure roles & permissions
- EC2 to host the Flask backend
- DuckDNS + HTTPS for secure external access via custom domain



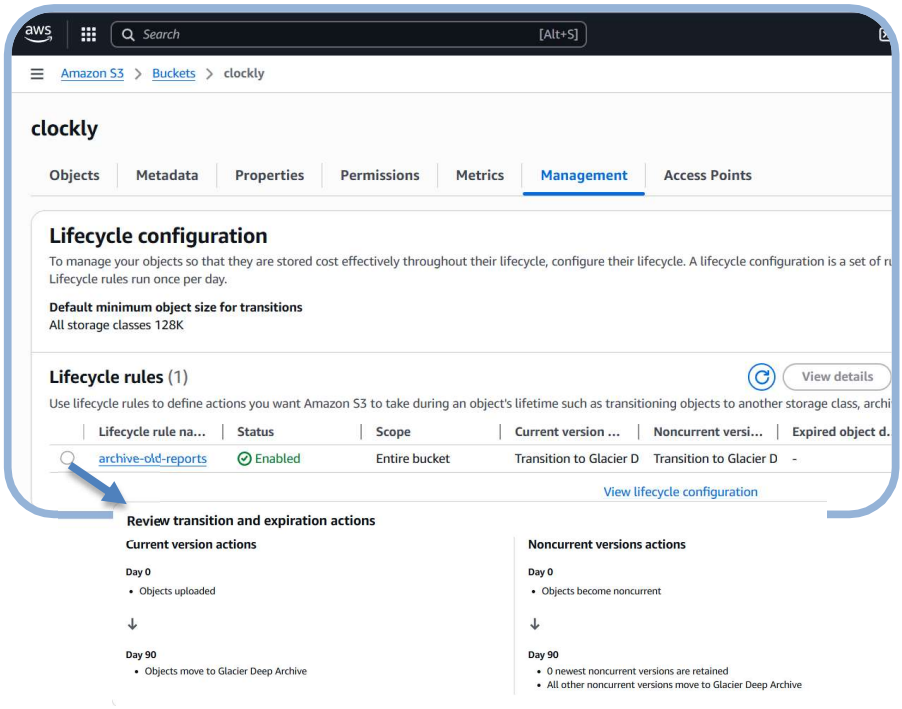
2.2 AWS Services Implementation

2.2.1 EC2 Instance

- Ubuntu Server (t2.micro)
- Hosts Flask backend via Gunicorn + Nginx
- HTTPS enabled using DuckDNS + Let's Encrypt
- IAM Role: AmazonS3FullAccess, CloudWatchAgentServerPolicy
- Security Groups: HTTP/HTTPS allowed, restricted PostgreSQL, IP-restricted SSH
- Managed with systemd for auto-restarts/logging

2.2.2 S3 Bucket & Glacier Storage

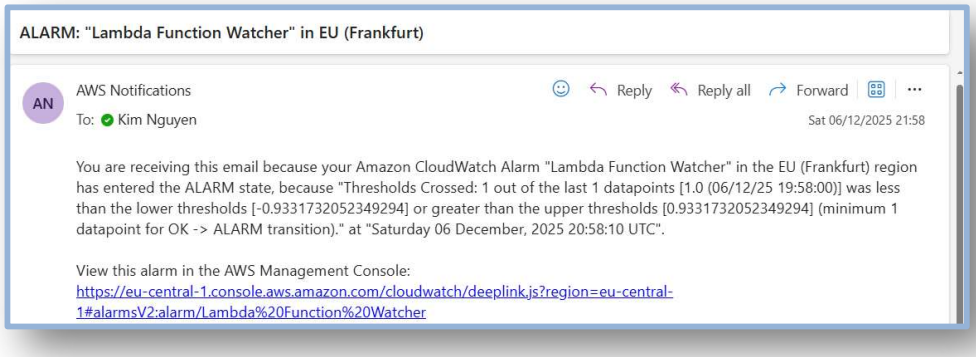
Clockly uses an S3 bucket to store daily team attendance reports generated by AWS Lambda. The bucket includes a lifecycle rule that automatically transitions files older than 90 days into Glacier for cost-effective long-term archival.



S3 Glacier is used to archive historical attendance reports older than 90 days. This reduces storage cost while ensuring auditability and compliance.

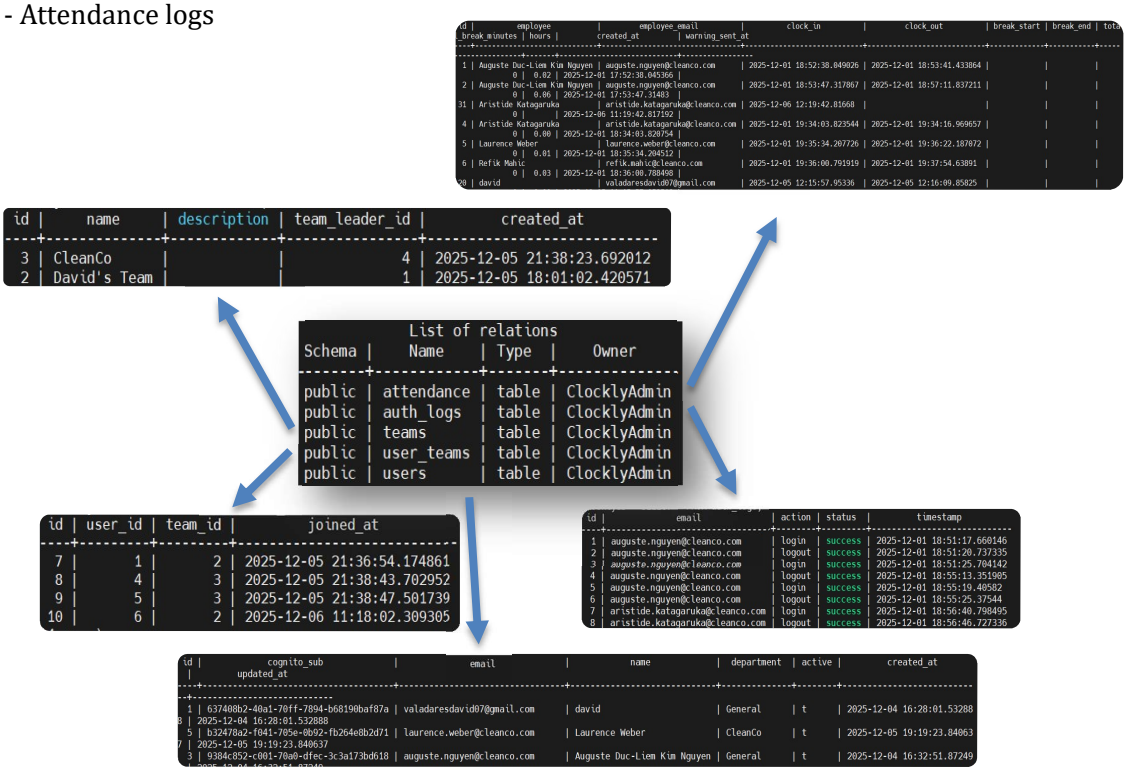
2.2.3 SNS Notifications

SNS is used to send:
- Shift warnings (8h and 10h auto clock-out)
- Daily team attendance reports
- System alerts for administrators

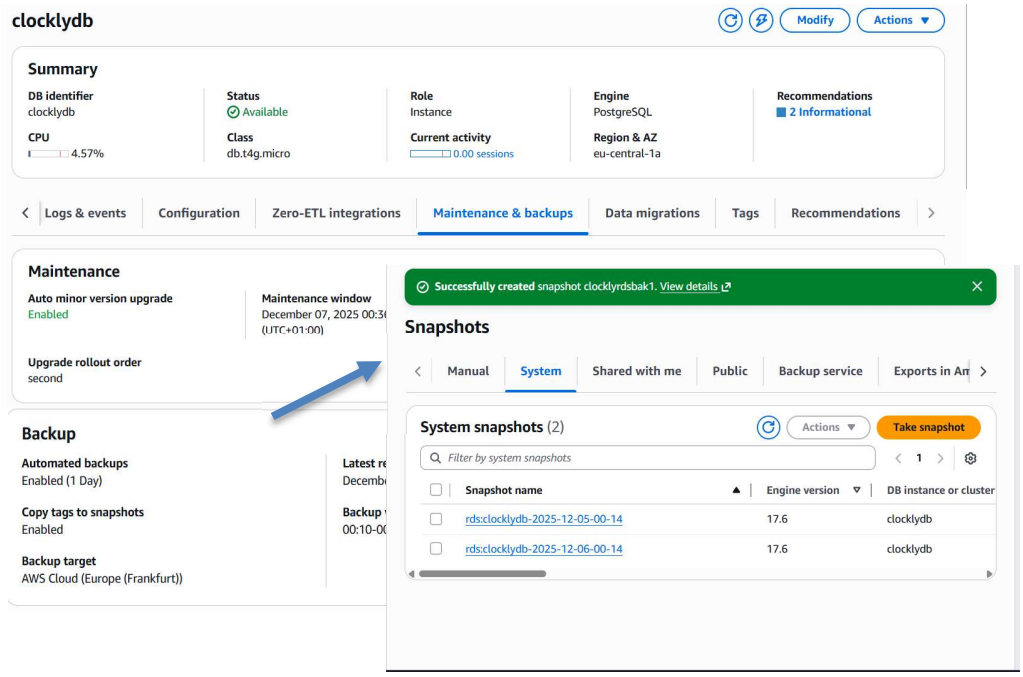


2.2.4 AWS RDS (PostgreSQL)

Clockly uses AWS RDS PostgreSQL for relational data storage. The database stores:
- Users
- Teams
- Attendance logs



RDS runs inside a private subnet, accessible only from EC2 and Lambda.
It is also backup daily.

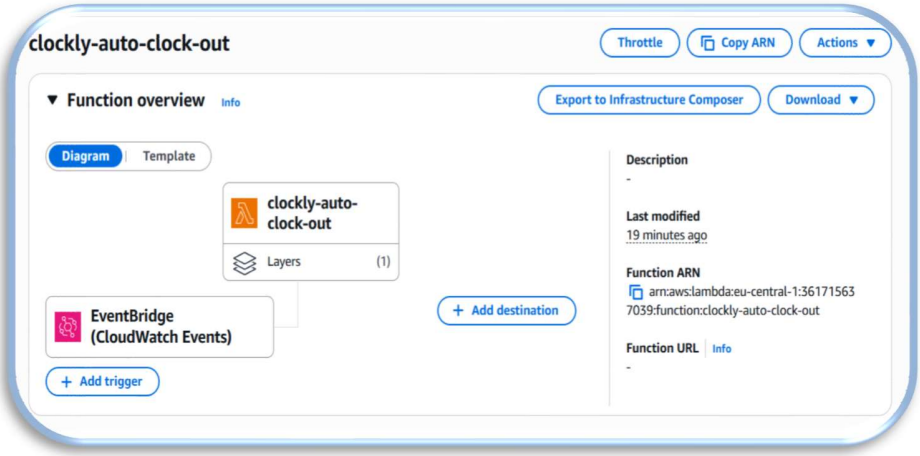


3. Supplementary AWS Services

3.1 AWS Lambda

Two Lambda functions automate Clockly operations. Both run inside a VPC and connect securely to RDS:

1. Auto Clock-Out (hourly)



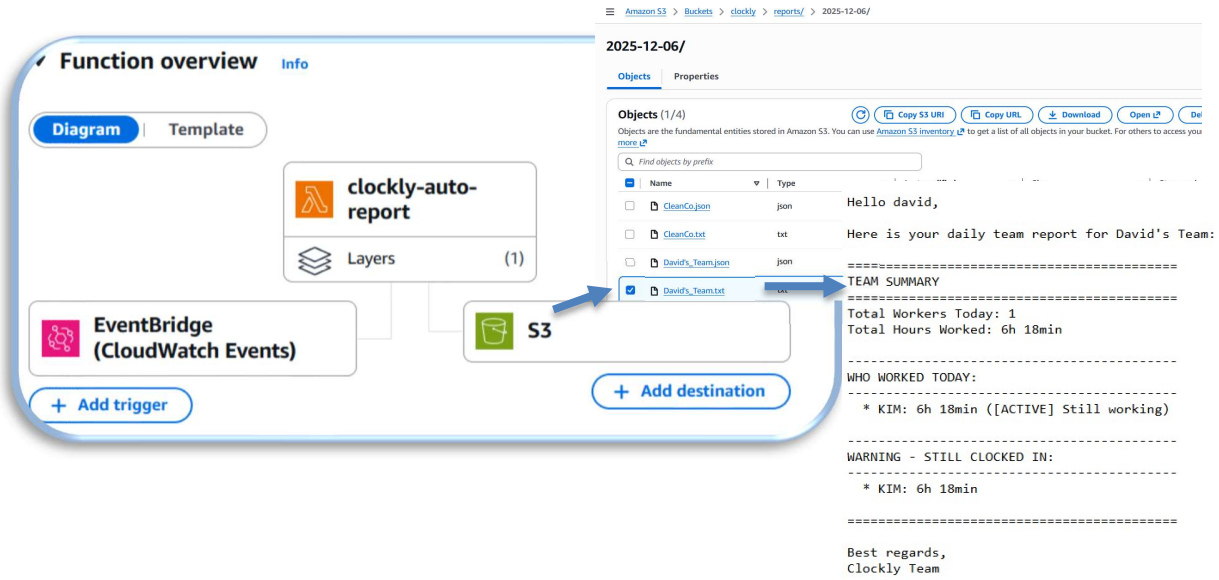
Purpose: Monitors active shifts and automatically manages overtime

Schedule: Every hour via EventBridge (CloudWatch Events)

Functionality:

- Queries all active shifts (not clocked out)
- Calculates hours worked (clock_in to now, minus breaks)
- **8-hour threshold:** Sends warning email via SNS, marks warning_sent_at
- **10-hour threshold:** Automatically clocks out user, sends notification

2. Daily Team Reports (22 PM)



Purpose: Generates and distributes daily attendance reports to team leaders

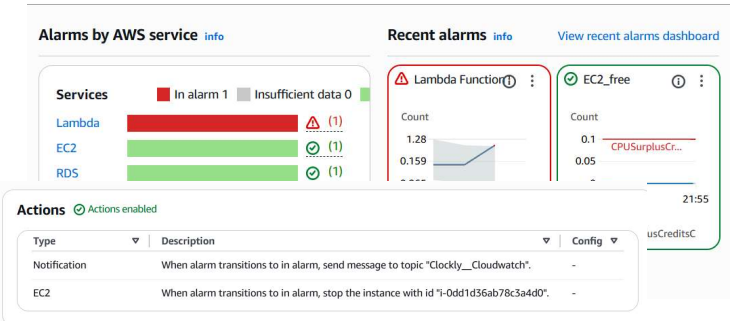
Schedule: Daily at 6 PM (18:00 UTC) via EventBridge

Functionality: - Queries all teams and their leaders
- For each team, collects today's attendance data
- Generates comprehensive report:
- Total workers and hours
- Who worked today (with clock-out status)
- Still clocked in warnings
- Overtime warnings (>8 hours)
- Long break warnings (>1 hour)
- Saves reports to S3 (text + JSON formats)
- Sends email to team leader via SNS

3.2 AWS CloudWatch

CloudWatch monitors EC2 and Lambda logs, tracks errors, and provides insights on performance and system health.

You can even add triggers to your alarms. We added a trigger action to an alarm to shut off the EC2, in case of overcharge.



4. Additional AWS Features

4.1 AWS IAM

IAM roles and policies protect access to AWS services. Separate roles exist for EC2, Lambda, and administrative tasks.

4.2 HTTPS (DuckDNS + Let's Encrypt)

Clockly uses DuckDNS as a free dynamic DNS domain and Let's Encrypt for automated TLS certificate generation, ensuring secure HTTPS access.

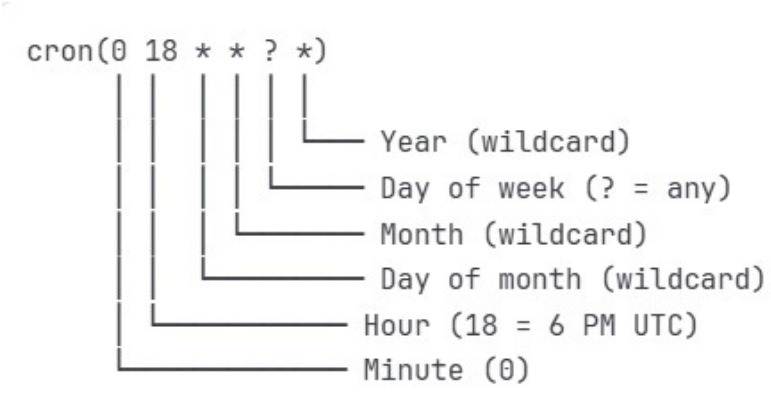
4.3 Systemctl Control Panel

Clockly backend is managed through systemctl:
- Start/stop the Flask server
- View logs
- Auto-restart on failures

4.4. AWS EventBridge

To schedule AWS Lambda functions, we used EventBridge.
- Auto Clock-Out (hourly)
- Daily Team Reports (18PM)

Here is an explanation of the cron code used



5. Application Features

5.1 Folder Structure

```
backend-python/
├── app.py                          # Main application setup
├── config.py                       # AWS, DB, Cognito configuration
├── cognito_auth.py                 # Authentication + role helpers
├── init_database.py                # Database initialization
│
├── models/                         # Data access layer
│   ├── user.py
│   ├── team.py
│   └── attendance.py
│
├── routes/                         # Flask Blueprints
│   ├── auth.py
│   ├── dashboard.py
│   ├── attendance.py
│   ├── admin_users.py
│   ├── admin_teams.py
│   └── reports.py
│
├── utils/                          # Utility modules
│   ├── helpers.py                  # Time, formatting utilities
│   └── s3_helpers.py               # S3 operations
│
└── templates/                      # Jinja2 HTML Templates
    ├── base.html
    ├── index.html
    ├── history.html
    ├── reports.html
    ├── admin_dashboard.html
    ├── admin_users.html
    ├── admin_user_form.html
    ├── admin_teams.html
    ├── admin_team_form.html
    ├── error.html
    └── login.html
```

5.2 Role-Based Access Control (RBAC)

Clockly uses Cognito custom attributes (custom:role) to implement three access levels:

Role	Permissions
Worker	Clock in/out, manage breaks, view own attendance
Team Leader	All worker actions + team-level reports
Admin	Manage users, teams, system-wide reports

5.3 Break Tracking System

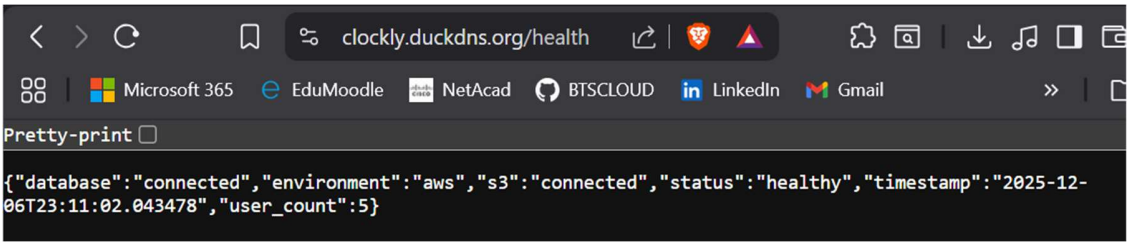
Clockly supports multiple breaks per shift with accurate time deduction:

- break_start & break_end track the current break
- total_break_minutes accumulates all breaks
- Users cannot clock out while on a break
- Break durations automatically added to total break time

Work hours are calculated as:

(clock_out – clock_in – total_break_time)

5.4 Health Check API



5.5 Timezone (Luxembourg)

All timestamps run in Europe/Luxembourg timezone using a “naive datetime” approach to avoid PostgreSQL timezone conversions. This ensures:

- Consistent local timestamps
- Correct hour and duration calculations
- Predictable behavior across EC2 regions

5.6 Authentication Flow

Clockly uses the Cognito Hosted UI and Authorization Code flow:

Steps:

1. User redirected to Cognito
2. Upon login, Cognito sends a one-time authorization code
3. Backend exchanges code for tokens (server-to-server)
4. JWT signature verified against Cognito JWKS
5. Minimal user info stored in Flask session

Best practices used:

- Tokens not stored in session (avoids 4 KB cookie limit)
- Client secret never exposed
- Signed JWT verification (RS256)
- Local session cleared + Cognito logout

5.7 Report Access Control

Two access levels:

Admin

- Full visibility of all teams and employees
- Can view entire system reports over any range

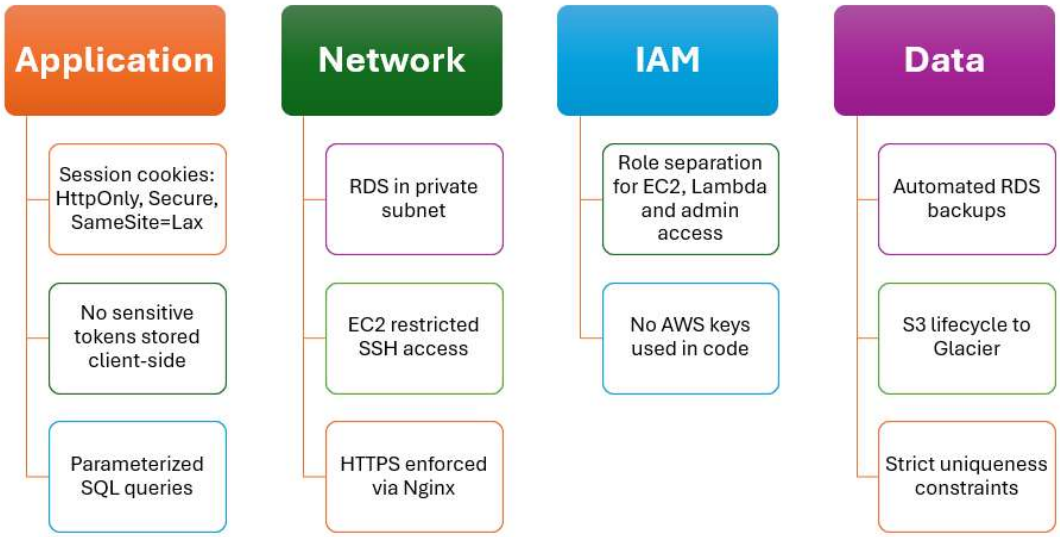
Team Leader

- Sees only their team members
- Access controlled using employee emails matched to team membership
- Can view personal history plus team statistics

This ensures clean separation of team data.

5.8 Security Best Practices

Clockly employs several layered security measures:



5.9 DuckDNS Integration

Clockly uses DuckDNS as a free dynamic DNS service to map a stable domain name to the EC2 instance's IP.

- DuckDNS updates automatically via cron or system
- Let's Encrypt issues certificates for the DuckDNS domain
- Nginx uses these certificates to enable full HTTPS

This allows secure access without needing a paid domain.

6. User Guide

6.1 Logging In

Users authenticate through AWS Cognito using OAuth2. Cognito handles sessions, MFA (optional), and user identity.

6.2 Clocking In

Workers clock in using the main dashboard. The system prevents duplicate sessions and logs timestamps in PostgreSQL.

6.3 Viewing Attendance

Users can view their last 30 days of attendance or filter by date range.

Team leaders can view their attendance and from their respective team.

6.4 Admin Functions

Admins can manage users, teams, roles, and generate manual reports.

7. Challenges Faced

- Configuring VPC access for Lambda to reach SNS and S3
- Correctly setting security groups
- Ensuring SSL certificates and redirecting
- IAM permission troubleshooting

8. Conclusion

Clockly demonstrates a complete AWS-based cloud system including authentication, automation, monitoring, secure deployment, and scalable architecture. It simulates a real enterprise-grade attendance tracking solution.