CI/CD Pipeline Documentation

Table of Contents

CI/CD Pipeline Documentation

Secure Messaging Application - GitHub Actions CI/CD

DevOps Engineer: Naeem Dosh | Fiverr

Work Completed - February 13, 2026

Summary

Implemented a complete CI/CD pipeline using GitHub Actions for automated testing, building, and deployment of the secure messaging application to AWS.

Application: https://taxplanner.app **Repository:** https://github.com/appcropolisdevops/awspoc **CI/CD Dashboard:** https://github.com/appcropolisdevops/awspoc/actions **AWS Region:** us-east-2 (Ohio)

Pipeline Architecture

```
Developer pushes code to GitHub (main branch)
                ↓
    ┌────────────────────────┐
    │   CI - Build & Test    │  (~55 seconds)
    │   ✅ PHP Lint           │
    │   ✅ Security Scan      │
    │   ✅ Docker Build       │
    │   ✅ Health Check       │
    └────────────────────────┘
                │
                ↓
    ┌────────────────────────┐
    │   CI - Deploy to AWS   │  (~90 seconds)
    │   ✅ Pull latest code   │
    │   ✅ Install deps       │
    │   ✅ Rebuild Docker     │
    │   ✅ Health check       │
    └────────────────────────┘
                │
                ↓
        Application Live at
       https://taxplanner.app
```

Total time from push to live: ~2 minutes

Three Pipelines Created

1. CI Pipeline (ci.yml)

File: `.github/workflows/ci.yml` **Triggers:** Every push to main/develop, every PR, manual trigger

Jobs:

Job	Duration	What It Does
PHP Lint	~9s	Checks all PHP files for syntax errors
Security Check	~5s	Scans for hardcoded secrets, verifies .env not committed
Docker Build	~38s	Builds containers, starts app, runs health check

PHP Lint: - Scans all `.php` files in `src/` directory - Catches syntax errors before deployment - Uses PHP 8.2

Security Check: - Scans for hardcoded API keys, OAuth secrets - Verifies `.env` file is not committed to repo - Blocks deployment if secrets found in code

Docker Build: - Builds PHP-FPM and Nginx containers - Creates test `.env` file - Starts containers and waits for startup - Verifies HTTP response from application - Checks container logs for errors

2. CD Pipeline (deploy.yml)

File: `.github/workflows/deploy.yml` **Triggers:** Push to main (ignores docs/ and .md files), manual trigger

Deployment Steps:

Step	Action	Details
1	Pull latest code	`git fetch` + `git reset --hard` on EC2
2	Install dependencies	Composer install via Docker
3	Stop containers	Stops all existing Docker containers
4	Build & start	`docker-compose up -d --build`
5	Wait	10 second startup delay
6	Health check	Verify containers running
7	Verify HTTP	Curl localhost:80 for response

How Deployment Works: - Uses AWS SSM (Systems Manager) to send commands to EC2 - No SSH keys needed - secure IAM-based access - EC2 is in private subnet, only accessible via SSM - Commands execute directly on the server - Output captured and displayed in GitHub Actions

Deployment Flow:

```
# On EC2 via SSM:
cd /app
git fetch origin main
git reset --hard origin/main
docker run composer install --no-dev
docker-compose -f docker-compose.prod.yml down
docker-compose -f docker-compose.prod.yml up -d --build
```

3. Terraform Pipeline (terraform.yml)

File: `.github/workflows/terraform.yml` **Triggers:** Changes to terraform/ files, manual trigger

On Pull Request: - Runs `terraform init` - Runs `terraform validate` - Runs `terraform plan` - Posts plan output as PR comment

On Push to Main: - Runs `terraform apply` automatically

Manual Options: - Plan (preview changes) - Apply (deploy infrastructure) - Destroy (teardown all infrastructure)

GitHub Secrets Configured

All secrets are securely stored in GitHub repository settings:

Secret	Purpose	Status
`AWS_ACCESS_KEY_ID`	AWS authentication	✅ Configured
`AWS_SECRET_ACCESS_KEY`	AWS authentication	✅ Configured
`EC2_INSTANCE_ID`	Target EC2 for deployment	✅ Configured
`GOOGLE_CLIENT_ID`	OAuth authentication	✅ Configured
`GOOGLE_CLIENT_SECRET`	OAuth authentication	✅ Configured

Location: Repository Settings → Secrets and Variables → Actions

Pipeline Test Results

Test 1: CI Pipeline

- **Commit:** "Add CI/CD pipelines with GitHub Actions"
- **Result:** ✅ PASSED
- **Duration:** 1m 2s
- **Jobs:** PHP Lint ✅ | Security Check ✅ | Docker Build ✅

Test 2: CD Pipeline Fix & Deploy

- **Commit:** "Fix CD pipeline - stop existing containers before deploy"
- **Result:** ✅ PASSED
- **Duration:** 1m 28s
- **Jobs:** CI Checks ✅ | Deploy to Production ✅

Test 3: Application Change

- **Commit:** "Update page title"
- **Result:** ✅ PASSED (CI + CD)
- **Duration:** CI 55s + CD 1m 31s
- **Change verified live at:** https://taxplanner.app

Test 4: Content Change

- **Commit:** "Update tagline to Enterprise Secure Messaging"
- **Result:** ✅ PASSED
- **Change visible:** Login page updated automatically

How to Use

Automatic Deployment (Recommended)

```
# Make code changes
git add .
git commit -m "Your change description"
git push origin main

# Pipeline runs automatically:
# 1. CI tests your code (~55s)
# 2. CD deploys to production (~90s)
# 3. Change is live at taxplanner.app
```

Manual Deployment

1. Go to https://github.com/appcropolisdevops/awspoc/actions
2. Select “CD - Deploy to AWS”
3. Click “Run workflow”
4. Select branch: main
5. Click “Run workflow”

View Pipeline Status

1. Go to https://github.com/appcropolisdevops/awspoc/actions
2. See all pipeline runs with status
3. Click any run for detailed logs
4. Each step shows output and timing

Issues Fixed During Setup

Issue	Cause	Fix
CD workflow file error	CI missing `workflow_call` trigger	Added `workflow_call` to ci.yml
Docker buildx error	Buildx plugin not installed on EC2	Used `DOCKER_BUILDKIT=0` flag
Port 80 already allocated	Existing containers not stopped	Added `docker stop` + `docker rm` before deploy

Files Created

File	Purpose
`.github/workflows/ci.yml`	CI pipeline - lint, security, build
`.github/workflows/deploy.yml`	CD pipeline - deploy to AWS via SSM
`.github/workflows/terraform.yml`	Infrastructure pipeline

Security Notes

- AWS credentials stored as GitHub Secrets (never in code)
- Deployment via SSM (no SSH keys exposed)
- EC2 in private subnet (not directly accessible)
- Security scan blocks commits with hardcoded secrets
- `.env` file verified not in repository
- All pipeline logs available in GitHub Actions

Document Version: 1.0 **Date:** February 13, 2026 **DevOps Engineer:** Naeem Dosh **Platform:** Fiverr