# **DevOps Assessment: Yii2 PHP Application Deployment Using Docker, Ansible, and CI/CD**

## **Overview**

This document describes the complete setup and deployment of a Yii2 PHP application using Docker Swarm, NGINX, Ansible automation, and CI/CD with GitHub Actions. It also covers monitoring setup using Prometheus and Node Exporter. The infrastructure is hosted on AWS EC2.

## **Architecture Summary**

* **Cloud Provider:** AWS EC2 (Ubuntu 22.04)
* **Application:** Yii2 PHP containerized application
* **Container Orchestration:** Docker Swarm
* **Reverse Proxy:** NGINX
* **Automation:** Ansible for provisioning and deployment
* **CI/CD:** GitHub Actions for automated deployments
* **Monitoring:** Prometheus and Node Exporter running in Docker containers

## **Step-by-Step Setup**

### **1. Provision AWS EC2 Instance**

* Launched an Ubuntu 22.04 EC2 instance (t2.micro)
* Configured security groups to allow ports: 22 (SSH), 80 (HTTP), 443 (HTTPS), 9090 (Prometheus), 9100 (Node Exporter)

### **2. Docker and Docker Swarm Installation via Ansible**

* Used Ansible playbook to install Docker engine and Docker Compose on the EC2 instance
* Initialized Docker Swarm mode on the instance
* Deployed the Yii2 PHP application and NGINX reverse proxy as Docker services
* Ansible playbook manages idempotent installation and service setup

### **3. Application Deployment Automation**

* Cloned Yii2 PHP application repository onto the EC2 instance using Ansible
* Used Docker Compose or Docker Swarm stack file to define services:  
  + yii2app service running PHP app container
  + nginx service acting as reverse proxy
* Configured NGINX with site config files under /etc/nginx/sites-available/yii2

### **4. CI/CD Pipeline with GitHub Actions**

* Configured GitHub Actions workflow .github/workflows/deploy.yml
* On every push to the main branch:  
  + Connects via SSH to EC2 instance
  + Pulls latest code from Git repository
  + Runs Ansible playbook or Docker stack update commands to redeploy services
* Enables fully automated deployment on code changes

### **5. Monitoring Setup**

* Installed Prometheus server as a Docker container on port 9090
* Installed Node Exporter as a Docker container on port 9100 to expose host metrics
* Configured Prometheus to scrape metrics from Node Exporter endpoint
* Verified monitoring targets and metrics via Prometheus UI: http://<EC2-IP>:9090

## **Important File Locations**

* **Ansible Playbook:** playbook.yml
* **Inventory File:** inventory.ini
* **Nginx Site Config:** /etc/nginx/sites-available/yii2
* **Prometheus Config:** /opt/prometheus/prometheus.yml
* **GitHub Actions Workflow:** .github/workflows/deploy.yml

## **Verification & Testing**

* Confirmed Yii2 PHP app is accessible at http://<EC2-IP>
* Verified NGINX reverse proxy is routing requests correctly
* Accessed Prometheus dashboard on http://<EC2-IP>:9090 and confirmed targets are UP
* Confirmed Node Exporter metrics available and scrape status healthy
* Triggered code push and validated automated deployment with GitHub Actions pipeline

## **Conclusion**

This project demonstrates a professional-grade DevOps deployment pipeline for a PHP web application using modern container orchestration and infrastructure automation tools. The setup includes:

* Automated infrastructure provisioning with Ansible
* Containerized app deployment using Docker Swarm
* Automated CI/CD pipeline with GitHub Actions
* Robust monitoring of system and application metrics via Prometheus and Node Exporter

This architecture ensures scalable, reliable, and maintainable application deployment with real-time observability.