

# DevOps CA 2 Report

---

## Group Members:

Siddharth Sandeep - 22070122210

Shreyas Tambe - 22070122221

Rohit Raj - 22070122248

## 1. Project Overview

**Project Name:** PathoScanAI

**GitHub Repo:** <https://github.com/shreyastambe103/PathoScanAI>

**Deployed at:** <https://pathoscanai-50rd.onrender.com/>

### Objective:

To implement a full DevOps pipeline for PathoScanAI, including:

- Automated deployment using CI/CD tools
- Configuration management and environment setup using Ansible
- Containerization and orchestration with Docker & Kubernetes
- Monitoring and logging with Prometheus (or Nagios)
- Reflection on challenges and lessons learned

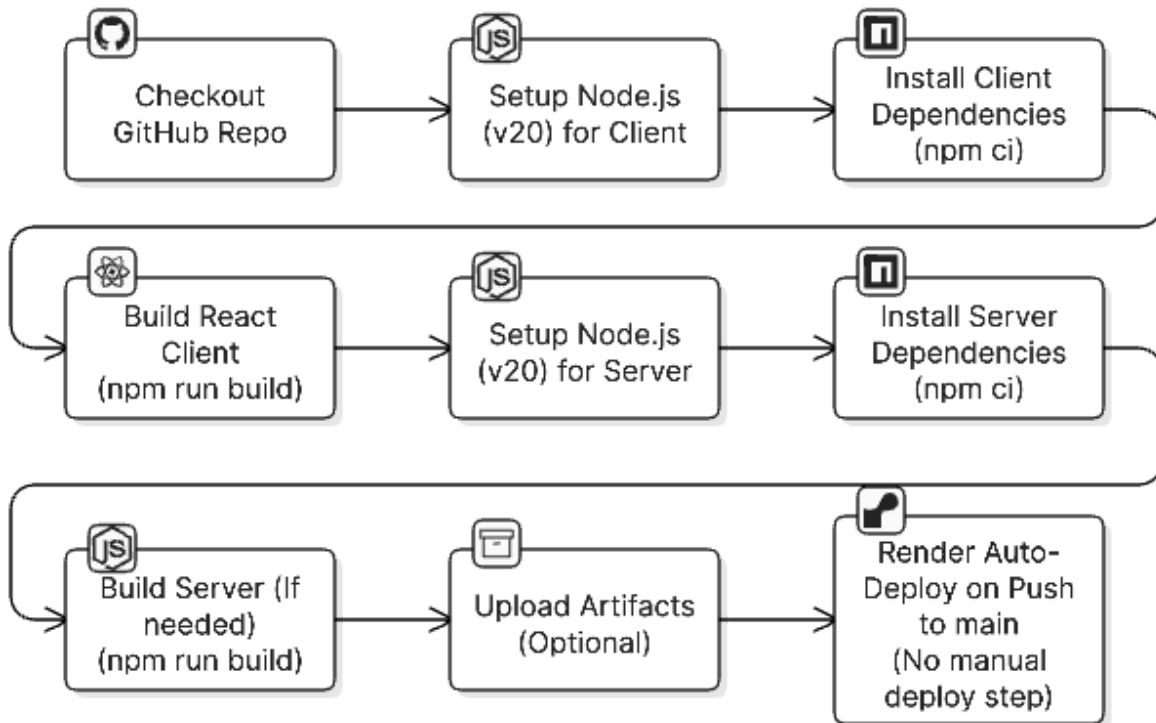
### Scope:

- All steps were implemented locally using WSL2 / Docker / Windows binaries where necessary.
- The pipeline ensures reproducibility, automation, and observability of the PathoScanAI application.

## Step 1 – Deployment Strategy

**Tool Chosen:** GitHub Actions

## Pipeline Overview:



## Pipeline Steps:

1. Trigger: Push or Pull Request to main branch
2. Checkout repository
3. Install dependencies (Node.js, npm)
4. Run tests (if any)
5. Deploy application to local environment / Docker container

## Files Submitted:

- `deploy.yml` – GitHub Actions workflow file

## Step 2 – Configuration Management (Ansible)

**Objective:** Automate environment setup for PathoScanAI

**Flow:**

## WSL2 Ubuntu (Ansible Host) → Ansible Playbook → Localhost Environment

### Tasks Performed:

- Installed Node.js, npm, git, build tools
- Cloned PathoScanAI repository
- Ran npm install and npm run build

### Files Submitted:

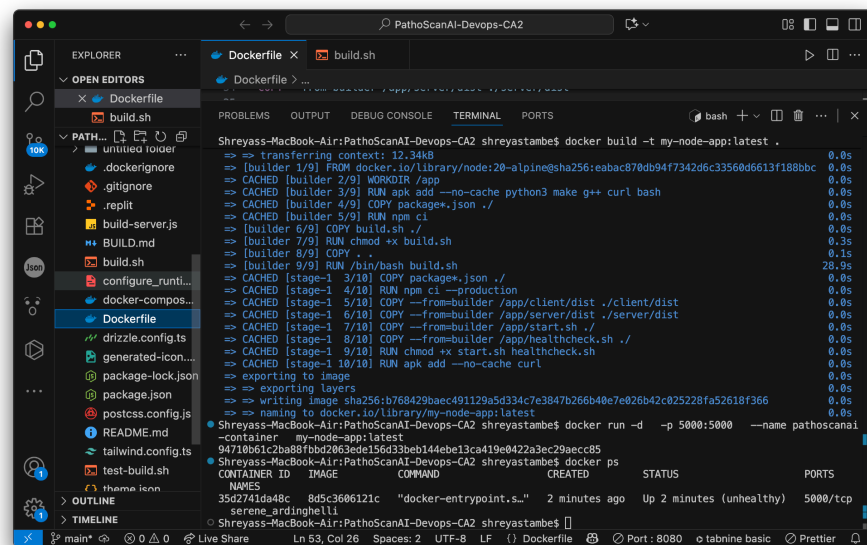
- `playbook.yml` – main playbook
- `inventory.ini` – localhost target

### Outcome:

Environment fully configured automatically, no manual steps required

## Step 3 – Containerization & Orchestration

Tools Used: Docker & Kubernetes



The screenshot shows a VS Code editor window with a file explorer on the left and a terminal on the right. The file explorer shows a project named 'PathoScanAI-Devops-CA2' with files like 'Dockerfile', 'build.sh', 'package.json', and 'README.md'. The terminal displays the output of a Docker build command, showing the progress of building the image, including copying files, running npm commands, and exporting the image. The build is successful, and the image is named 'my-node-app:latest'.

```
Shreyass-MacBook-Air:PathoScanAI-Devops-CA2 shreyastambe$ docker build -t my-node-app:latest .
=> transferring context: 12.34kB
=> [builder 1/9] FROM docker.io/library/node:20-alpine@sha256:eabac870db94f7342d6c33560d6613f188bbc 0.0s
=> CACHED [builder 2/9] WORKDIR /app 0.0s
=> CACHED [builder 3/9] RUN apk add --no-cache python3 make g++ curl bash 0.0s
=> CACHED [builder 4/9] COPY package*.json ./ 0.0s
=> CACHED [builder 5/9] RUN npm ci 0.0s
=> [builder 6/9] COPY build.sh ./ 0.0s
=> [builder 7/9] RUN chmod +x build.sh 0.3s
=> [builder 8/9] COPY . . 0.1s
=> [builder 9/9] RUN /bin/bash build.sh 28.9s
=> CACHED [stage-1 3/10] COPY package*.json ./ 0.0s
=> CACHED [stage-1 4/10] RUN npm ci --production 0.0s
=> CACHED [stage-1 5/10] COPY --from=builder /app/client/dist ./client/dist 0.0s
=> CACHED [stage-1 6/10] COPY --from=builder /app/server/dist ./server/dist 0.0s
=> CACHED [stage-1 7/10] COPY --from=builder /app/start.sh ./ 0.0s
=> CACHED [stage-1 8/10] COPY --from=builder /app/healthcheck.sh ./ 0.0s
=> CACHED [stage-1 9/10] RUN chmod +x start.sh healthcheck.sh 0.0s
=> CACHED [stage-1 10/10] RUN apk add --no-cache curl 0.0s
=> exporting to image 0.0s
=> exporting layers 0.0s
=> writing image sha256:b768429baec491129a5d334c7e3847b266b40e7e026b42c025228fa52618f366 0.0s
=> naming to docker.io/library/my-node-app:latest 0.0s
Shreyass-MacBook-Air:PathoScanAI-Devops-CA2 shreyastambe$ docker run -d -p 5000:5000 --name pathoscanai
-container my-node-app:latest
947106612ba88f0bd28c3ede156d33eb144eb13ca419e0422a3ec29aacc85
Shreyass-MacBook-Air:PathoScanAI-Devops-CA2 shreyastambe$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
35d2741db48c   8d5c3686121c   "docker-entrypoint.s..." 2 minutes ago   Up 2 minutes (unhealthy)   5000/tcp
serene_ardinghell
```

**Tasks Performed:**

- Dockerized PathoScanAI application
- Created Kubernetes manifests:
  - deployment.yaml
  - service.yaml

**Outcome:**

Application containerized and orchestrated with Kubernetes successfully

---

**Step 4 – Monitoring & Logging**

**Tools Used:** Prometheus + Node Exporter (or Nagios)

**Architecture Diagram:****Tasks Performed:**

- Installed Prometheus (Docker / Windows binaries)
- Installed Node Exporter for system metrics
- Configured Prometheus to scrape application and system metrics
- Created basic dashboard showing:
  - Uptime
  - Latency
  - Error rate

**Files Submitted:**

- prometheus.yml
- Node Exporter setup scripts

**Outcome:**

Application and system metrics visible in real-time, ensuring observability

---

**Reflection****Challenges Faced:**

- Learning curve for Ansible syntax & YAML
- Running Ansible on Windows localhost (not cloud VM)
- Docker Desktop issues / Prometheus setup
- Kubernetes manifests & rolling update commands
- Mapping metrics to meaningful dashboards

**Lessons Learned:**

- Automation ensures consistent and reproducible environment setup
- Containerization & orchestration simplify deployment and scaling
- Monitoring is essential for understanding app and system health
- Documentation and screenshots are crucial for reproducibility

**Tools Mastered:**

- GitHub Actions
  - Ansible
  - Docker & Kubernetes
  - Prometheus + Node Exporter
-

# Final Pipeline Architecture

