

Project Report on

**Automated Deployment and Monitoring of TodoApp with DevOps Practices**

**Submitted by**

**Komal Vede (230944223049)**

Under the guidance of

**Mr. Gajanan Taur**

**In partial fulfillment of the award of Industrial Training and Internship in DevOps and Cyber Security**



**Sunbeam Institute of Information Technology,**

**Pune (Maharashtra)**

**PG-DITISS -2023**

# DECLARATION

I hereby declare that this written submission represents my own ideas in my own words. Wherever I have included ideas or words from others, I have appropriately cited and referenced the original sources. I further declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated or falsified any idea, data, fact or source in this submission.

I understand that any violation of the above may result in disciplinary action by the Institute and may also lead to legal or penal action from the original sources if proper permission has not been obtained or proper citation has not been made.

**Komal Vede (230944223049)**

Place: Pune

Date: 7th October 2025

# CERTIFICATE

This is to certify that the project report entitled **“Automated Deployment and Monitoring of TodoApp with DevOps Practices”**, submitted by **Komal Vede (230944223049)** is the bonafide work completed under our supervision and guidance in partial fulfillment of the requirements for the award of **Industrial Training and Internship in DevOps and Cyber Security** at Sunbeam Institute of Information Technology, Pune.

Place: Pune

Date: 7th October 2025

**Mr. Gajanan Taur Mr. Vishal Salunkhe**

Guide Course Coordinator

**Mr. Nitin Kudale**

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

# APPROVAL CERTIFICATE

This is to certify that the Project-II report entitled **“Automated Deployment and Monitoring of TodoApp with DevOps Practices”** by **Komal Vede (230944223049)** is approved for **Industrial Training and Internship in DevOps and Cyber Security** at Sunbeam Institute of Information Technology, Pune.

Place: Pune

Date:

Examiner:

**(Signature)**

**(Name)**

# CONTENTS

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO** |
| **Declaration** |  |
| **Certificate** |  |
| **Approval Certificate** |  |
| **Abstract** | i |
| **1.INTRODUCION** | 1 |
| 1.1 Applications | 1 |
| 1.2 Organization and Project Plan | 3 |
| **2. LITERATURE SURVEY** | 4 |
| Paper 1 | 4 |
| Paper 2 | 4 |
| Paper 3 | 5 |
| **3. SYSTEM DEVELOPMENT AND DESIGN** | 6 |
| 3.1 Proposed System | 6 |
| 3.2 Flow Chart | 7 |
| 3.3 Technology used | 8 |
| 3.3.1 AWS EC2 | 8 |
| 3.3.2 Git | 9 |
| 3.3.3 Docker | 10 |
| 3.3.4 Jenkins | 11 |
| 3.3.5 Snort | 12 |
| 3.3.6 Nagios | 13 |
| **4. PROJECT OUTPUT** | 14 |
| **5. CONCLUSION** | 18 |
| 5.1 Conclusion | 18 |
| 5.2 Future Scope | 18 |
| **REFERENCES** | 19 |

# ABSTRACT

This project implements a complete DevOps lifecycle for a Python Flask-based Todo application with a MySQL backend. Using Docker for containerization, Kubernetes for orchestration, and GitHub Actions with ArgoCD for CI/CD, the system achieves fully automated deployment and scalability. Persistent storage ensures data reliability, while Prometheus and Grafana provide real-time monitoring and alerts. Security is integrated at every stage, including container hardening, RBAC, network policies, and CI/CD secret management.

The project demonstrates how automation, observability, and security work together to create a production-ready, resilient, and scalable application, reflecting modern industry DevOps practices.

# INTRODUCTION

In the modern software development landscape, delivering applications rapidly, reliably, and securely is crucial. DevOps, which bridges the gap between development and operations, provides methodologies and tools for **continuous integration (CI), continuous delivery (CD), automated deployment, and monitoring**. Integrating security into DevOps—often referred to as **DevSecOps**—ensures that applications are resilient against cyber threats while maintaining high availability and performance.

This project demonstrates a **full DevOps lifecycle** applied to a Python Flask-based Todo application with a MySQL backend. The goal is to automate the deployment, orchestration, monitoring, and security of the application using industry-standard tools and cloud-native technologies.

Key objectives -

* **Containerization:** Packaging the application and database into portable Docker containers.
* **Orchestration:** Managing application deployment, scaling, and health using Kubernetes.
* **Persistent Storage:** Ensuring database reliability through Persistent Volumes (PV) and Persistent Volume Claims (PVC).
* **CI/CD Automation:** Automating builds, tests, and deployment using GitHub Actions and ArgoCD.
* **Monitoring & Observability:** Implementing real-time metrics collection, visualization, and alerting with Prometheus and Grafana.
* **Security Integration:** Applying best practices such as RBAC, encrypted secrets, network policies, and container vulnerability scanning.

The successful implementation of this project showcases how **automation, observability and security** work together to create a production-ready system.

## Applications

1. **Web Applications:** Continuous deployment and scaling of web apps like e-commerce sites or social media platforms.
2. **Enterprise Software:** Secure and reliable deployment of business-critical applications.
3. **Cloud Services:** Automation of cloud-native applications in AWS, GCP, or Azure.
4. **Microservices Architecture:** Efficient orchestration and monitoring of microservices-based systems.
5. **IoT Platforms:** Continuous updates and monitoring of IoT applications with minimal downtime.
6. **Educational Platforms:** Automated deployment of learning management systems or collaborative apps.

The successful implementation of this project showcases how **automation, observability, and security** work together to create a production-ready system.

## 1.2 Project Plan

**Table: Activities Details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **ACTIVITY** | **WEEK** | | | |
| **1** | **2** | **3** | **4** |
| 1 | Requirement Analysis & Planning |  |  |  |  |
| 2 | Environment Setup (Docker, Kubernetes, GitHub Actions, ArgoCD, Prometheus, Grafana) |  |  |  |  |
| 3 | Application Development (Flask Todo App + MySQL) |  |  |  |  |
| 4 | |  | | --- | | Containerization and Kubernetes Orchestration | |  |  |  |  |
| 5 | CI/CD Pipeline Implementation (GitHub Actions + ArgoCD + Helm) |  |  |  |  |
| 6 | Monitoring & Observability (Prometheus + Grafana Dashboards) |  |  |  |  |
| 7 | Security Integration (RBAC, Secrets, Network Policies, Trivy Scans) |  |  |  |  |
| 8 | Testing and Validation (CI/CD & Application) |  |  |  |  |
| 9 | Documentation & Report Preparation |  |  |  |  |

# 2. LITERATURE SURVEY

The literature survey focuses on existing tools, methodologies, and previous research in DevOps, CI/CD, monitoring, and security practices.

### ****2.1 DevOps and CI/CD Practices****

* DevOps bridges development and operations for faster software delivery.
* CI/CD pipelines automate building, testing, and deployment, reducing human error and increasing efficiency.
* Popular CI/CD tools include **Jenkins, GitHub Actions, GitLab CI, ArgoCD, CircleCI**.

### ****2.2 Containerization and Orchestration****

* **Docker** enables packaging applications into isolated, portable containers.
* **Kubernetes** provides orchestration, scaling, and self-healing for containerized applications.
* Persistent storage is managed using **PV/PVC**, ensuring that database data survives pod restarts.

### ****2.3 Monitoring and Observability****

* **Prometheus** collects metrics from nodes, pods, and applications.
* **Grafana** visualizes these metrics and enables alerts on system thresholds.
* Effective monitoring improves reliability and helps in proactive problem resolution.

### ****2.4 Security in DevOps****

* Integrating security in DevOps involves **container hardening, RBAC, secret management, vulnerability scanning**, and secure CI/CD pipelines.
* Tools like **Trivy, Snyk, HashiCorp Vault** help maintain secure deployments.

# SYSTEM DEVELOPMENT AND DESIGN

### ****3.1 System Architecture****

The high-level architecture demonstrates the flow from code development to deployment and monitoring:

**Workflow:**

Developer → GitHub Repository → GitHub Actions (CI) → Docker Hub

↓

ArgoCD (CD) → Kubernetes Cluster

↓

(Pods + PV/PVC + Services)

↓

Prometheus → Grafana → Alerts & Dashboards

**Components:**

* **Application Layer:** Flask Todo App (Python)
* **Database Layer:** MySQL with persistent storage
* **Orchestration Layer:** Kubernetes cluster managing pods and services
* **Automation Layer:** CI/CD pipeline with GitHub Actions and ArgoCD
* **Monitoring Layer:** Prometheus metrics collection and Grafana dashboards

### ****3.2 Docker Setup****

**Objective:** Containerize the application and database.

**Steps:**

1. Install Docker on Ubuntu/Debian.
2. Create a Dockerfile for Flask app:

FROM python:3.9

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]

1. Build and run containers:

docker build -t todo-app:latest .

docker run -d --name todo-db -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=mytododb -e MYSQL\_USER=todo-user -e MYSQL\_PASSWORD=password mysql:8.0

docker run -d --name todo-app -p 5001:5001 --link todo-db:db todo-app:latest

**Observations:**

* Containers communicate successfully via Docker network.
* Flask app connects to MySQL and performs CRUD operations.

**3.3: Kubernetes Cluster Setup**

**Objective:** Deploy the application in a scalable cluster.

**Steps:**

* Create Master and Worker nodes.
* Install kubeadm, kubelet, kubectl.
* Initialize cluster: kubeadm init --pod-network-cidr=10.244.0.0/16
* Apply networking (Flannel) and join Worker nodes.

**Observations:**

* Cluster running with nodes in “Ready” state.

**3.4 : Application Deployment on Kubernetes**

**Objective:** Orchestrate and expose the app using Kubernetes.

**Files:**

* todo-deployment.yaml (Deployment with 2 replicas)
* todo-service.yaml (NodePort service to expose app)

**Execution:**

kubectl apply -f todo-deployment.yaml

kubectl apply -f todo-service.yaml

**Observations:**

* App accessible via NodePort, scalable with multiple replicas.

**3.5 : Persistent Storage and CI/CD**

**Persistent Storage:** PV/PVC for MySQL ensures data retention.

**CI/CD Automation:**

* GitHub Actions automatically builds and pushes Docker images.
* ArgoCD updates Kubernetes deployment on code changes.
* Helm charts simplify deployments.

**Observations:**

* Full automation achieved; zero manual deployment required.

**3.6 : Monitoring & Security**

**Monitoring:**

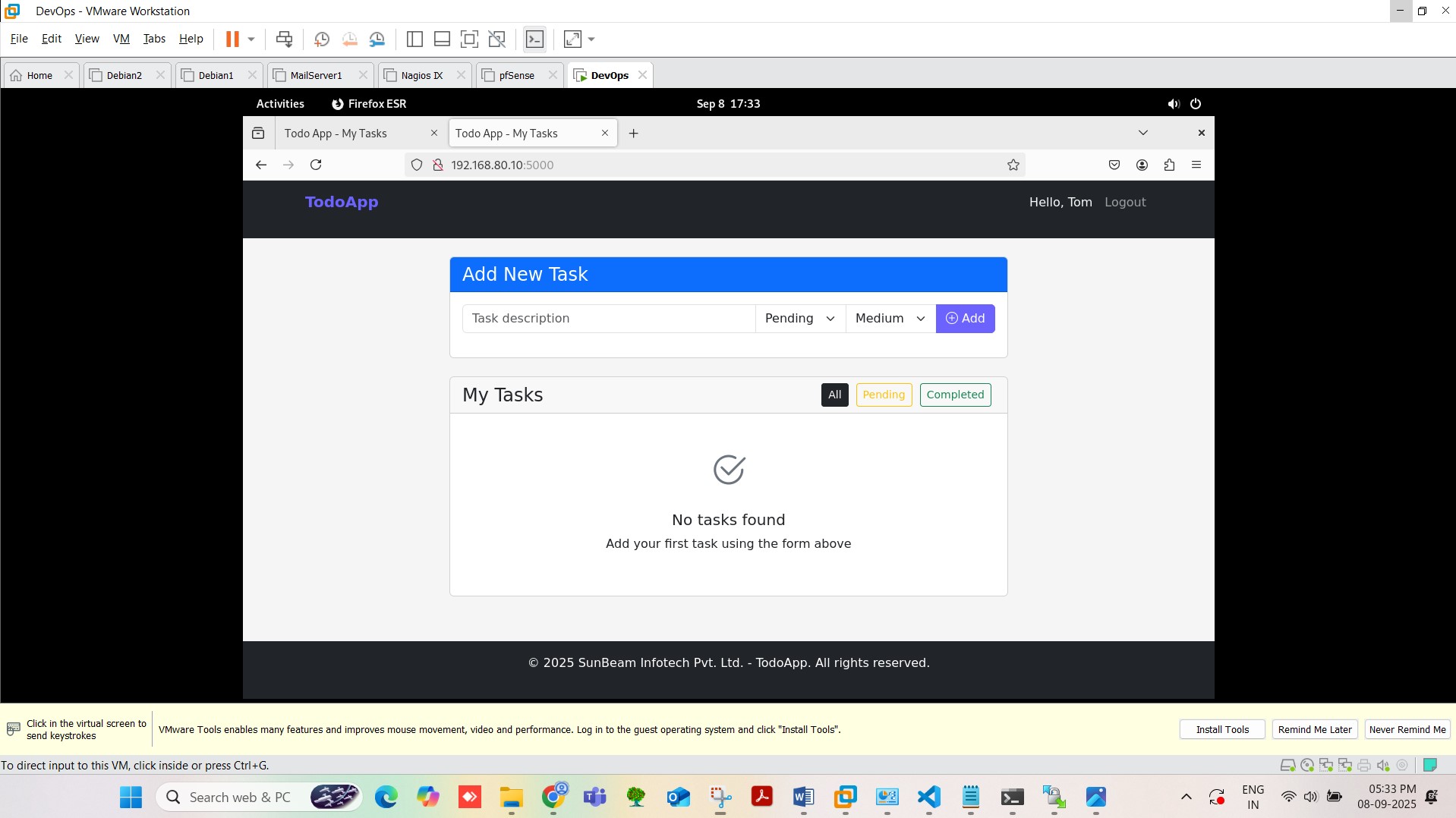
* Prometheus collects metrics from nodes, pods, and app.
* Grafana visualizes metrics with dashboards and alerts.

**Security Measures:**

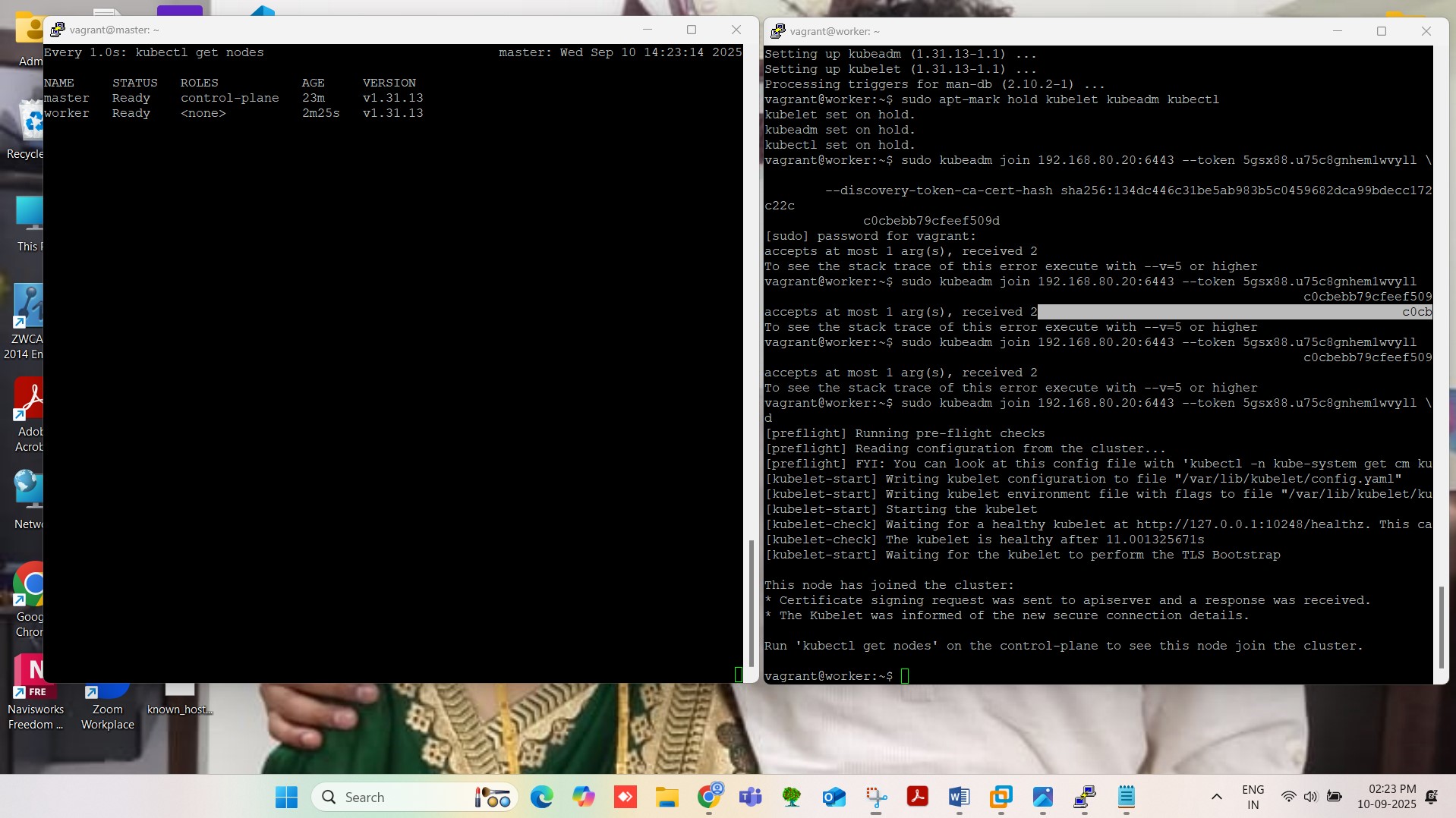
| **Layer** | **Security Measures** |
| --- | --- |
| Docker | Non-root users, minimal base images |
| Kubernetes | RBAC, service accounts |
| Networking | Network policies restricting inter-pod traffic |
| Secrets | Encrypted DB credentials |
| CI/CD | GitHub secrets for tokens and keys |
| Image Security | Trivy vulnerability scanning |
| Monitoring | Password-protected Grafana, internal Prometheus access |

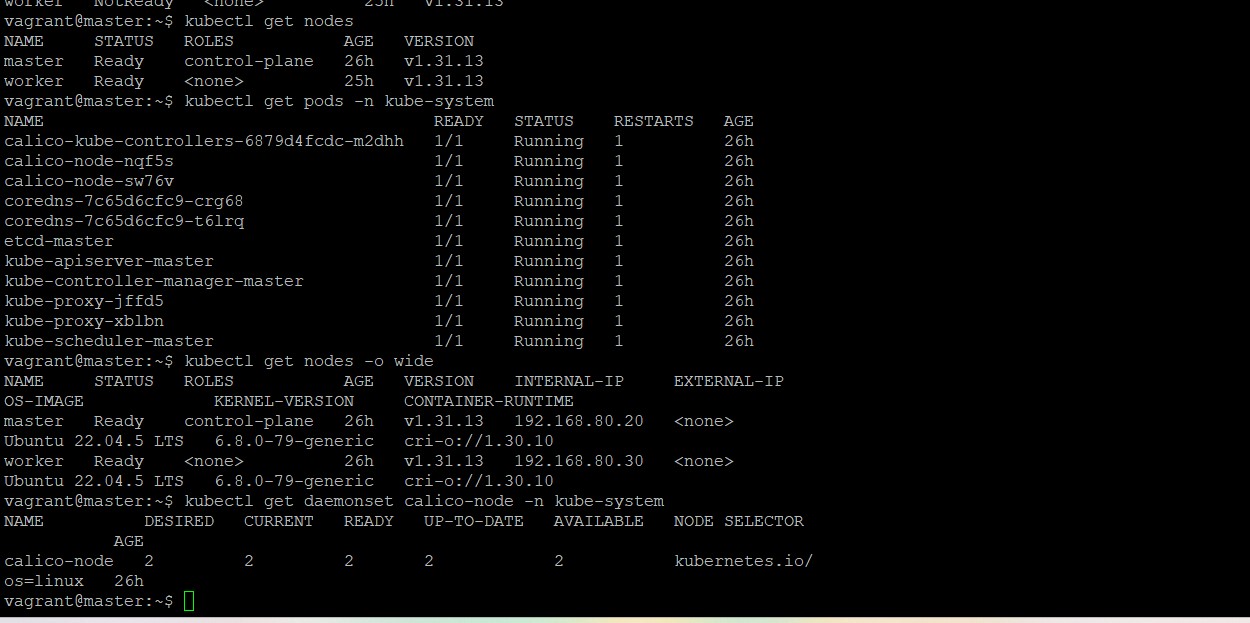
# 4. Project Output

## Todo app

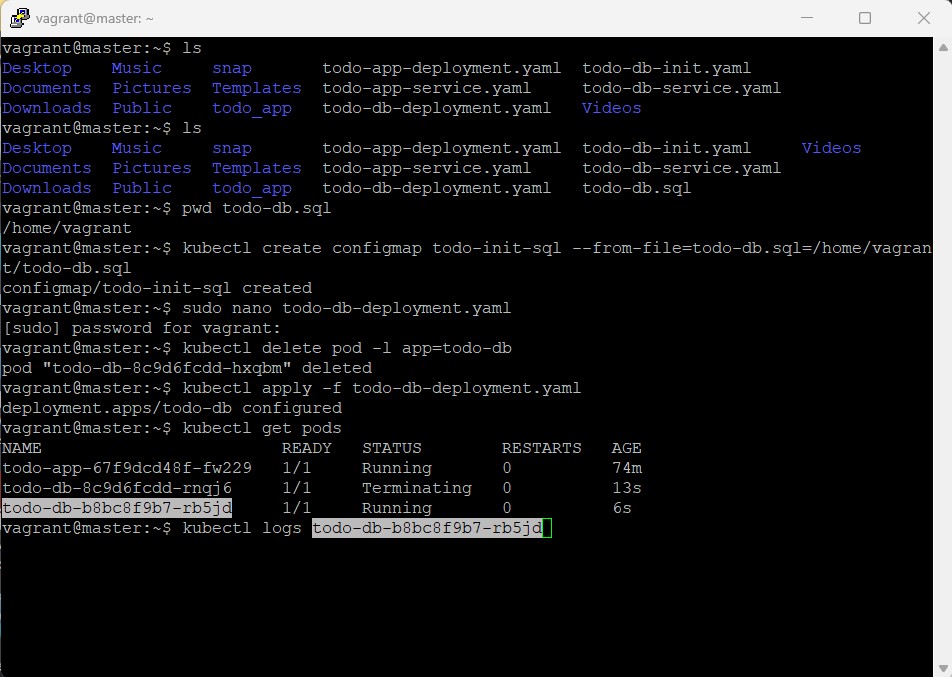


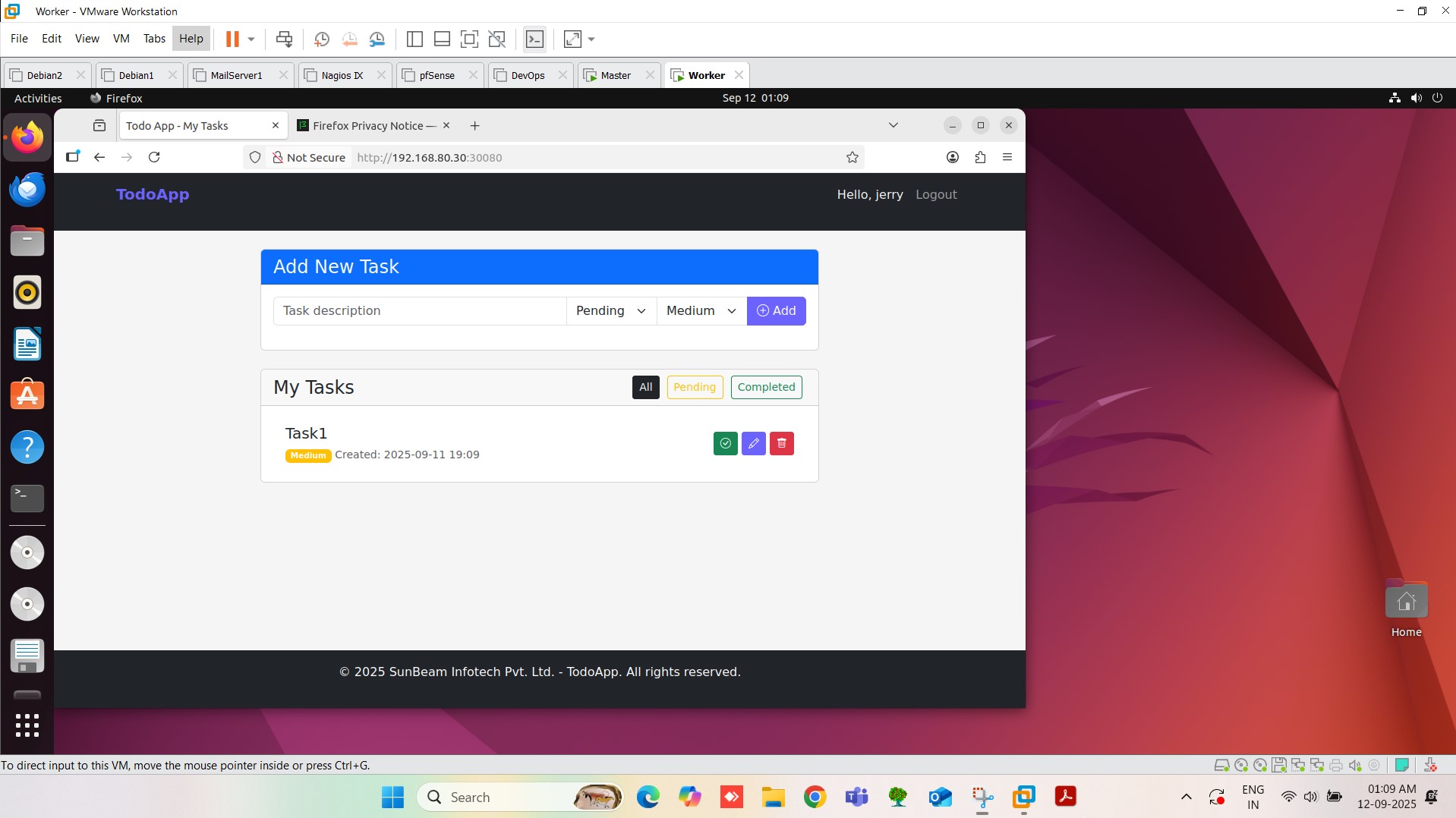
## K8s cluster



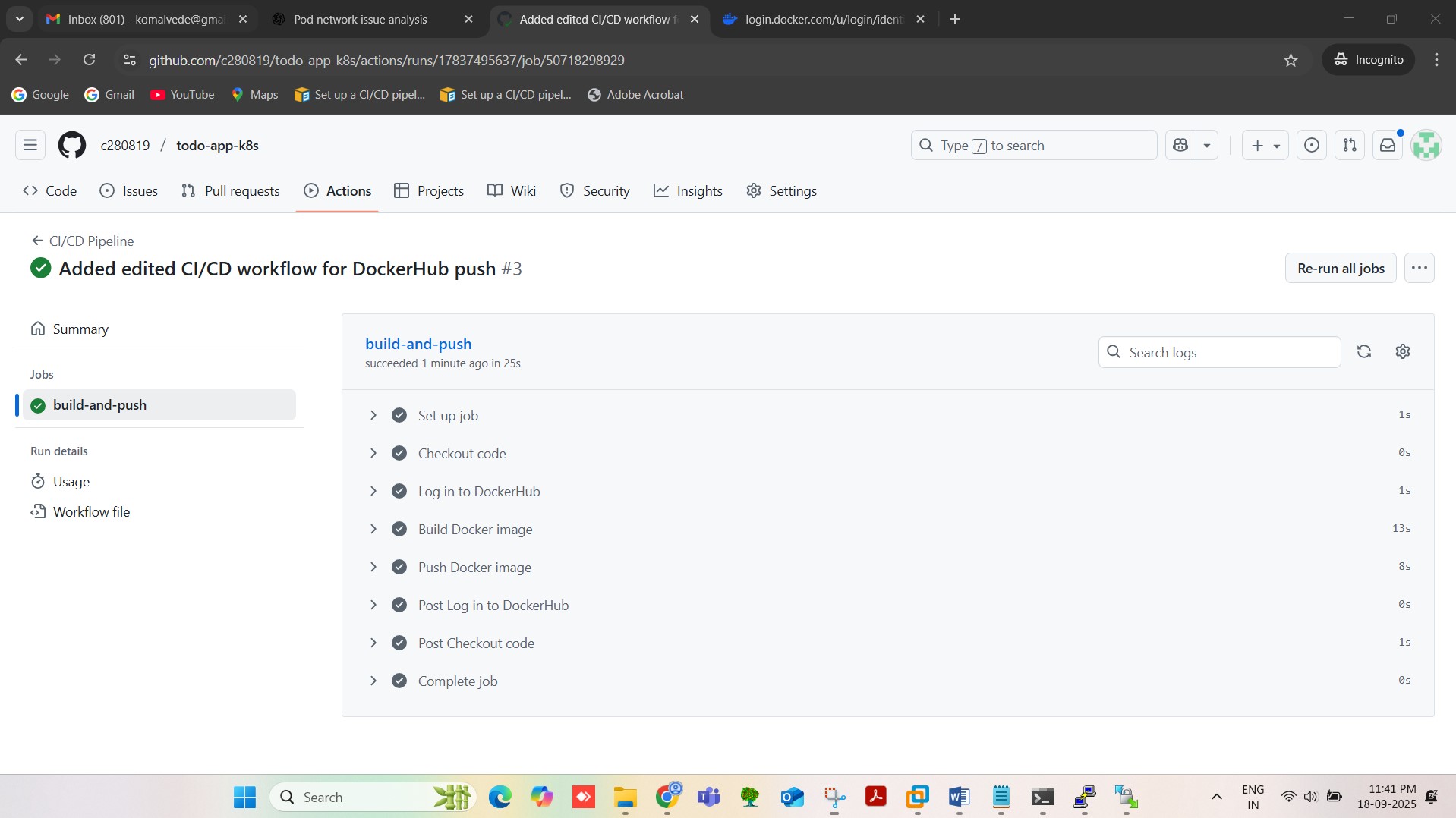


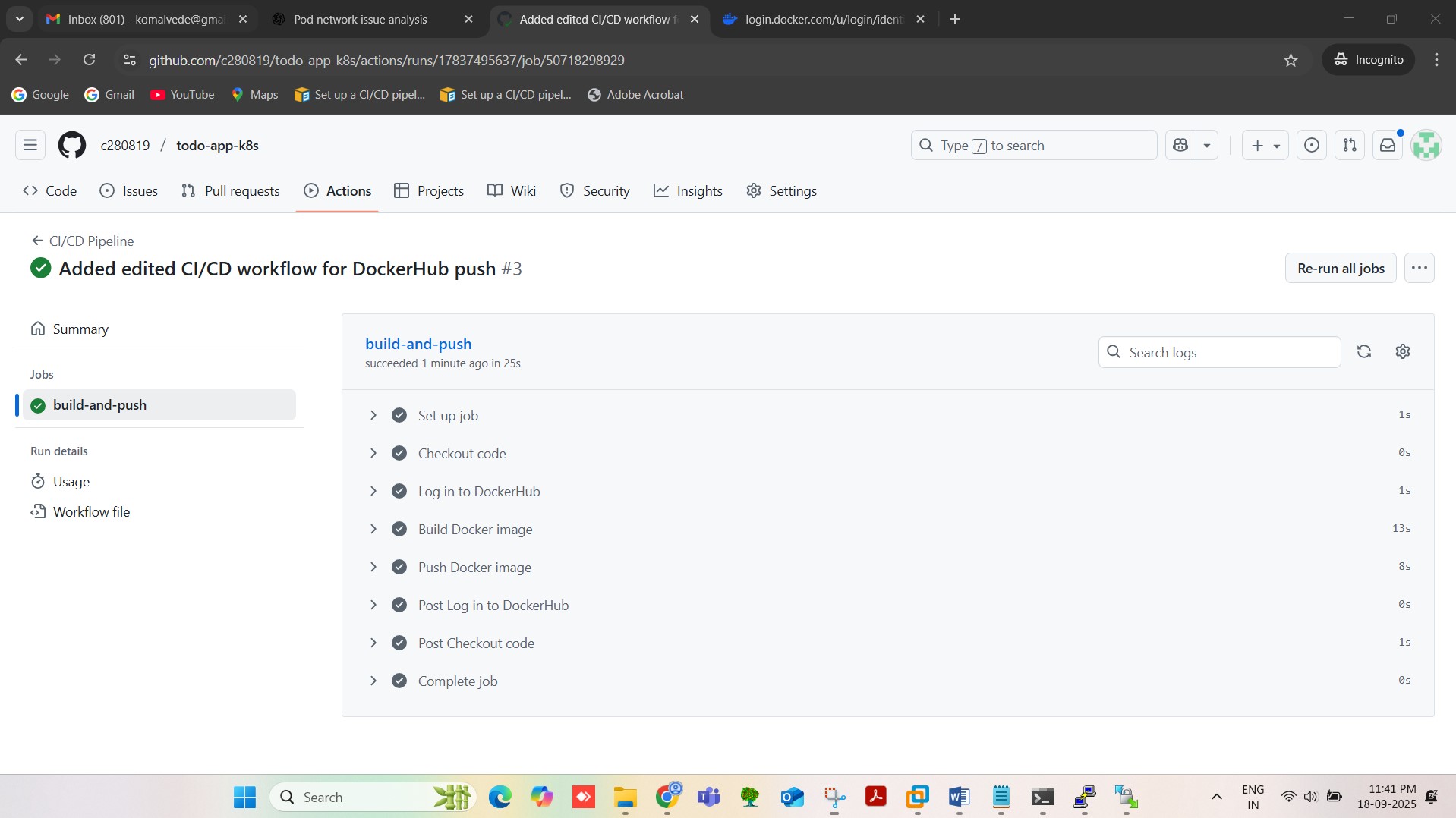
## App Orchestration

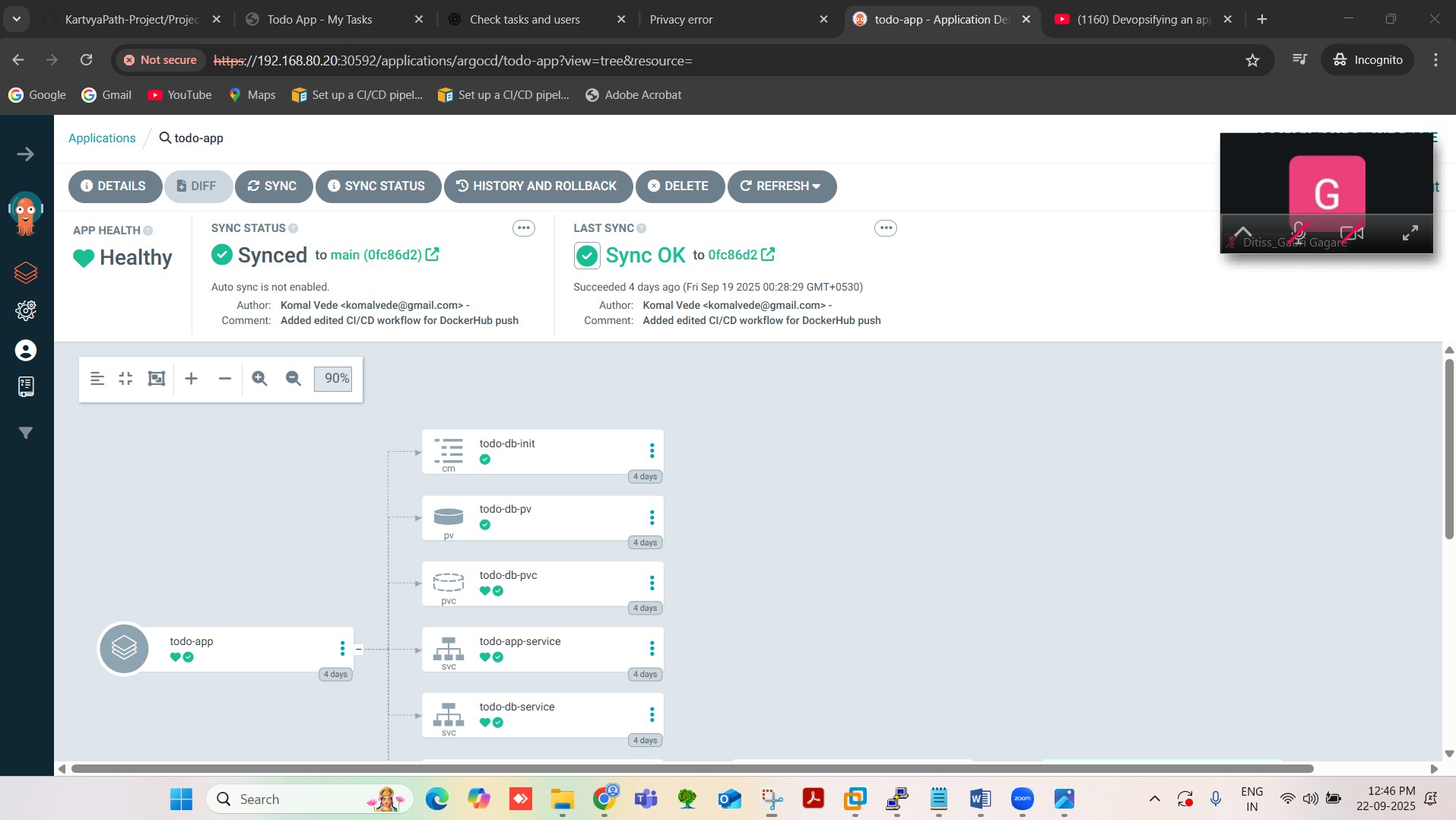




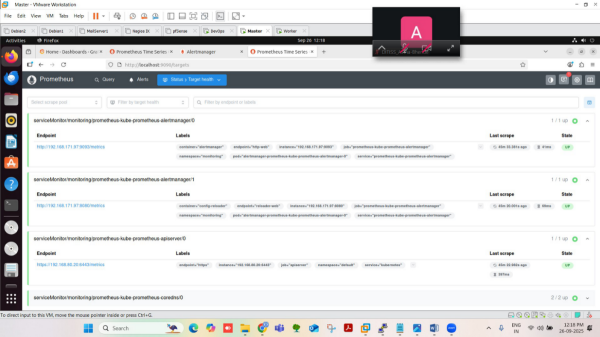
* 1. **GitHub Action and ArgoCD**







* 1. **Prometheus:**

****

# 5. CONCLUSION

## 5.1 Conclusion

This project successfully implements a **complete DevOps lifecycle** integrating automation, monitoring, and security. Key achievements include:

* Full containerization and orchestration using Docker and Kubernetes.
* CI/CD automation with GitHub Actions and ArgoCD.
* Persistent storage, ensuring data reliability.
* Real-time monitoring with Prometheus and Grafana.
* Security integration at all stages, including container, cluster, and pipeline.

## 5.2 Future Scope

* Cloud deployment (AWS/GCP)
* Advanced logging with ELK stack
* Kubernetes Ingress with SSL
* Automated backups with Velero
* Advanced vulnerability scanning with Snyk

# REFERENCES

1. Kubernetes Documentation – <https://kubernetes.io/docs>
2. Docker Official Docs – <https://docs.docker.com>
3. Prometheus & Grafana Docs – <https://prometheus.io/docs>, <https://grafana.com/docs>
4. CI/CD Tutorials – YouTube: ArgoCD + GitHub Actions, Mastering Kubernetes