

# Modern ToDoList App

DevOps Project Documentation

A containerized task management application with full CI/CD pipeline,  
Kubernetes orchestration, and comprehensive monitoring

React

Node.js

Docker

Kubernetes

GitHub Actions

Jenkins

Prometheus

Grafana

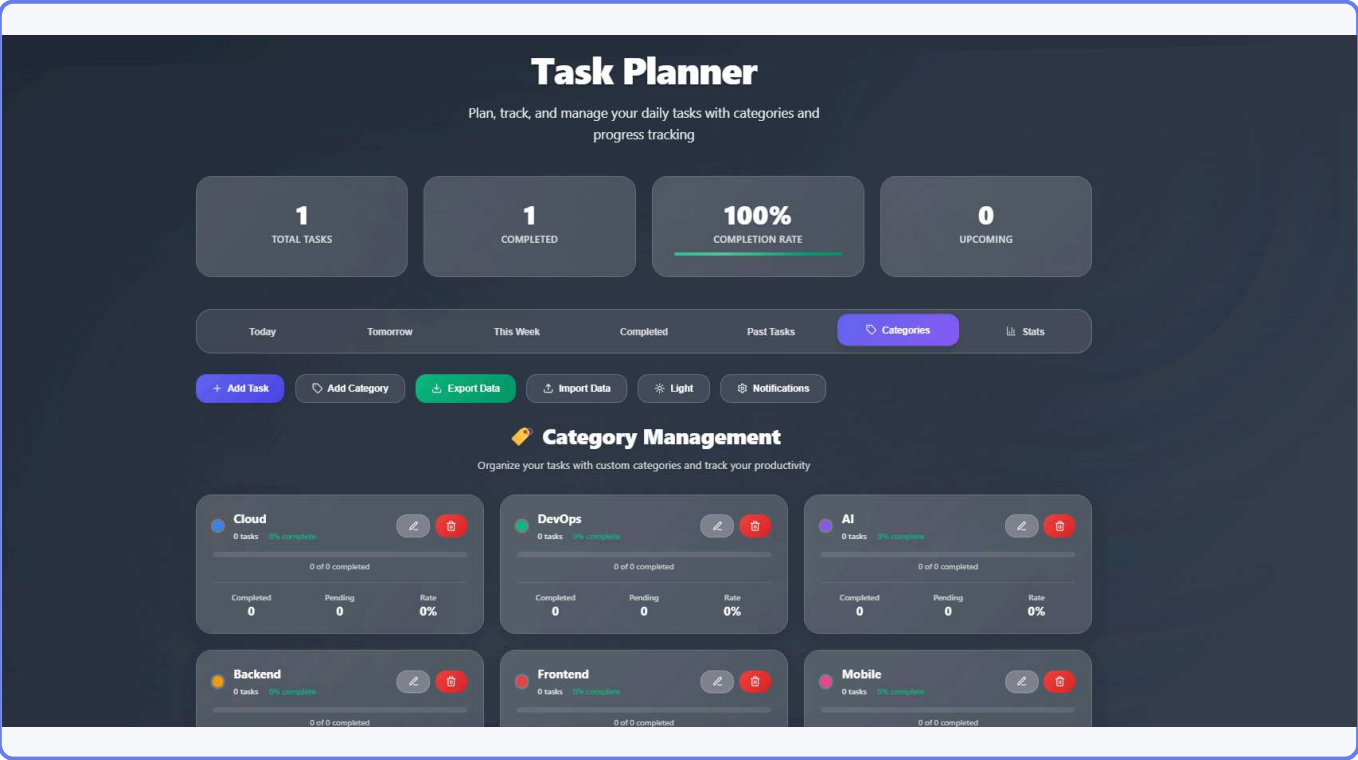
Terraform

**Author:** Mezni Ahmed Habib (didaa16)

GitHub: @didaa16 | Docker Hub: dida1609

# 1. Application Interface

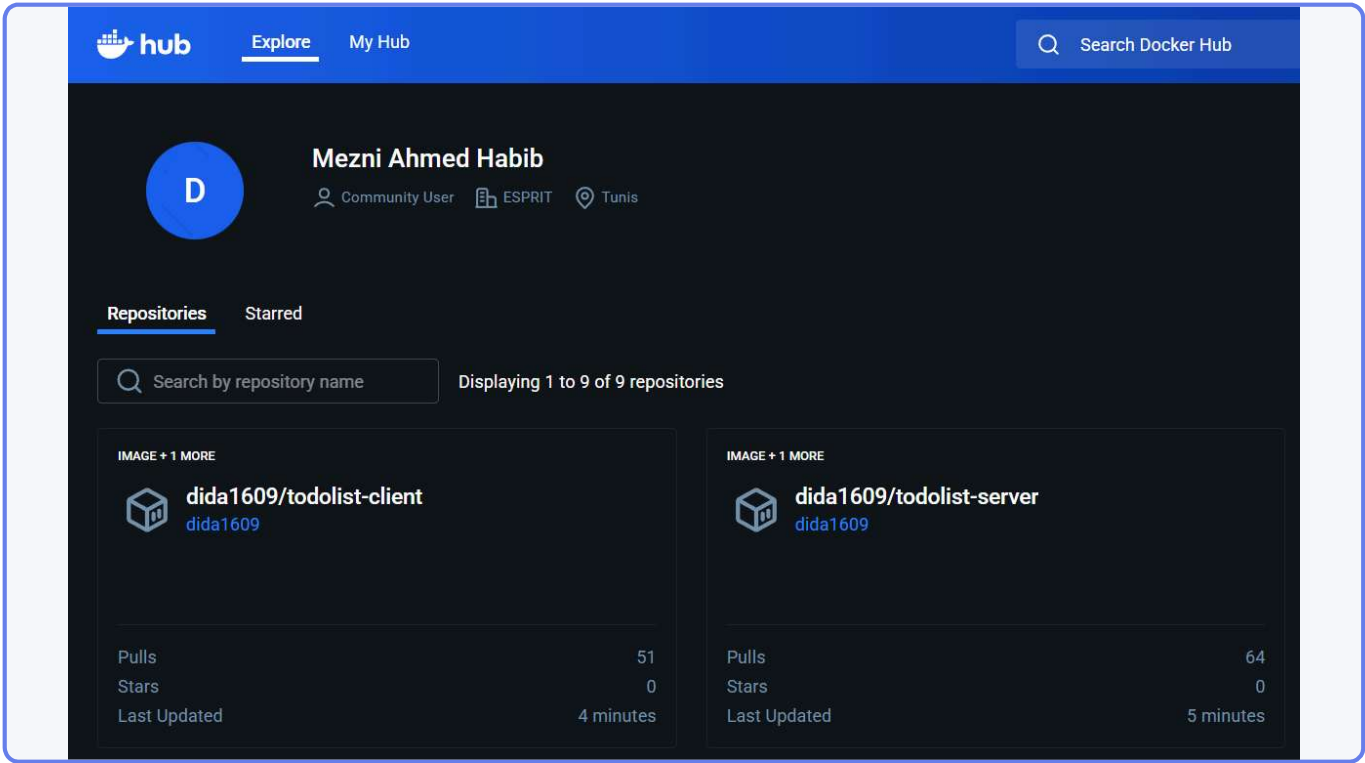
## Task Planner Dashboard



The main application interface showcasing the **Task Planner** with a modern dark theme featuring category management, real-time statistics, and comprehensive action buttons for task management, data import/export, and theme customization.

# 2. Container Registry

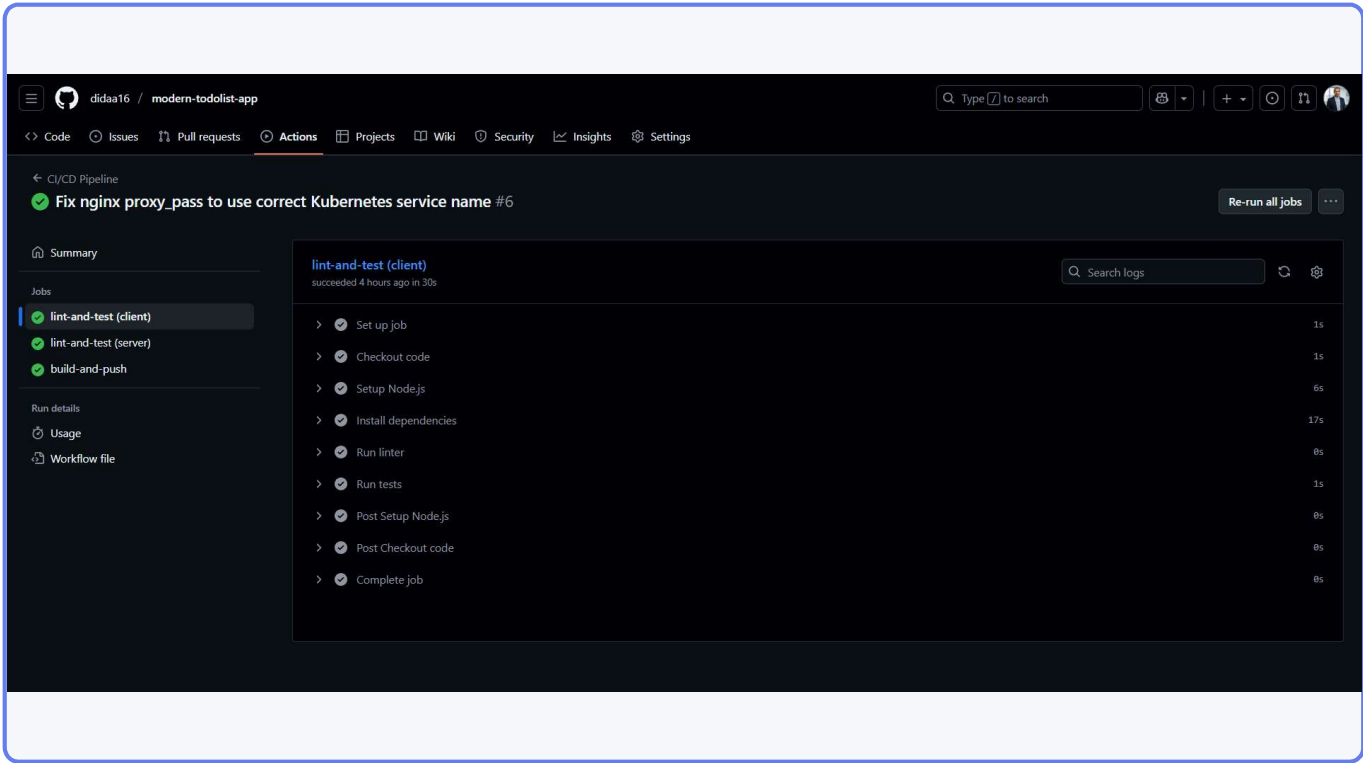
## Docker Hub Repositories



Docker Hub repositories showing published container images with **multi-stage builds** achieving 60% size reduction, improved security, and automated CI/CD deployment workflow.

# 3. CI/CD Pipeline - Client Build

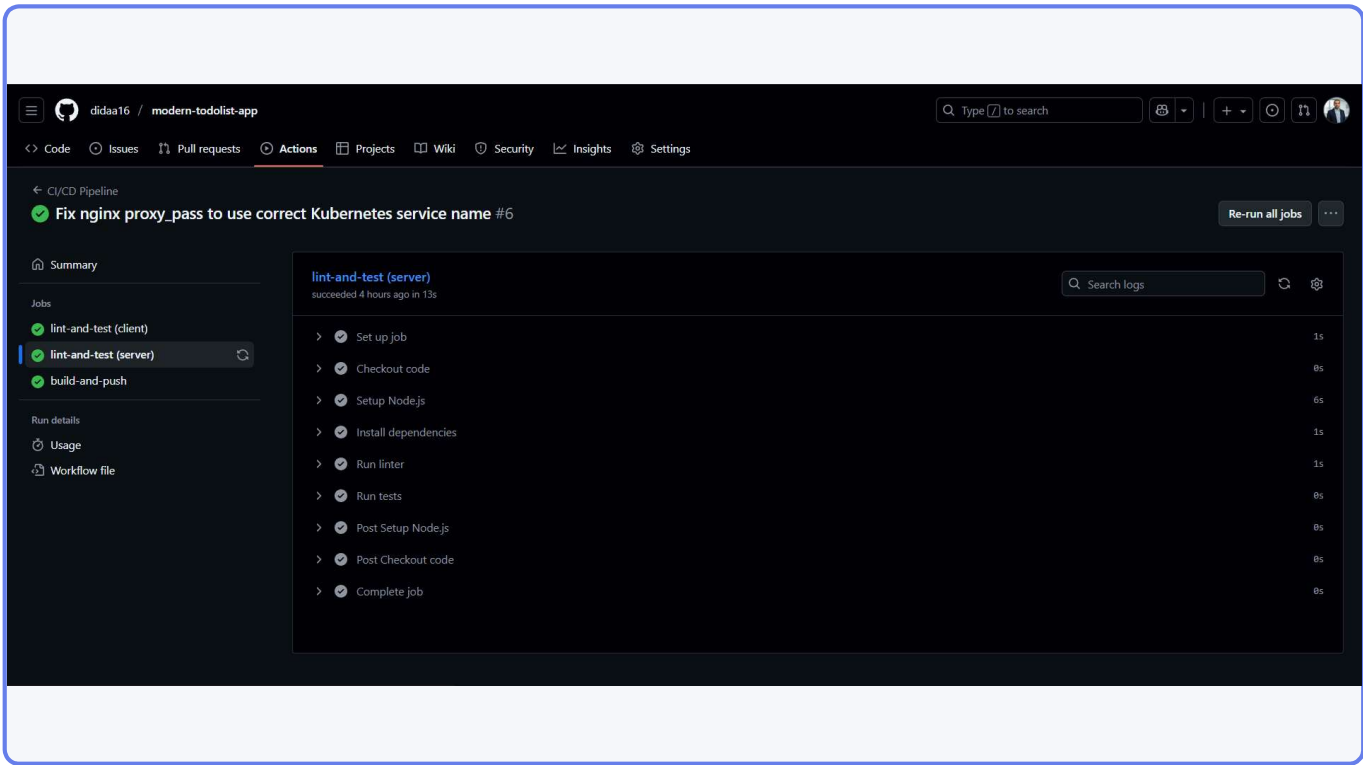
## GitHub Actions Workflow - Frontend



**GitHub Actions workflow** for frontend showing 9 automated stages including code checkout, Node.js setup, dependency installation, linting, and testing - completing in 30 seconds with full quality gate validation.

# 4. CI/CD Pipeline - Server Build

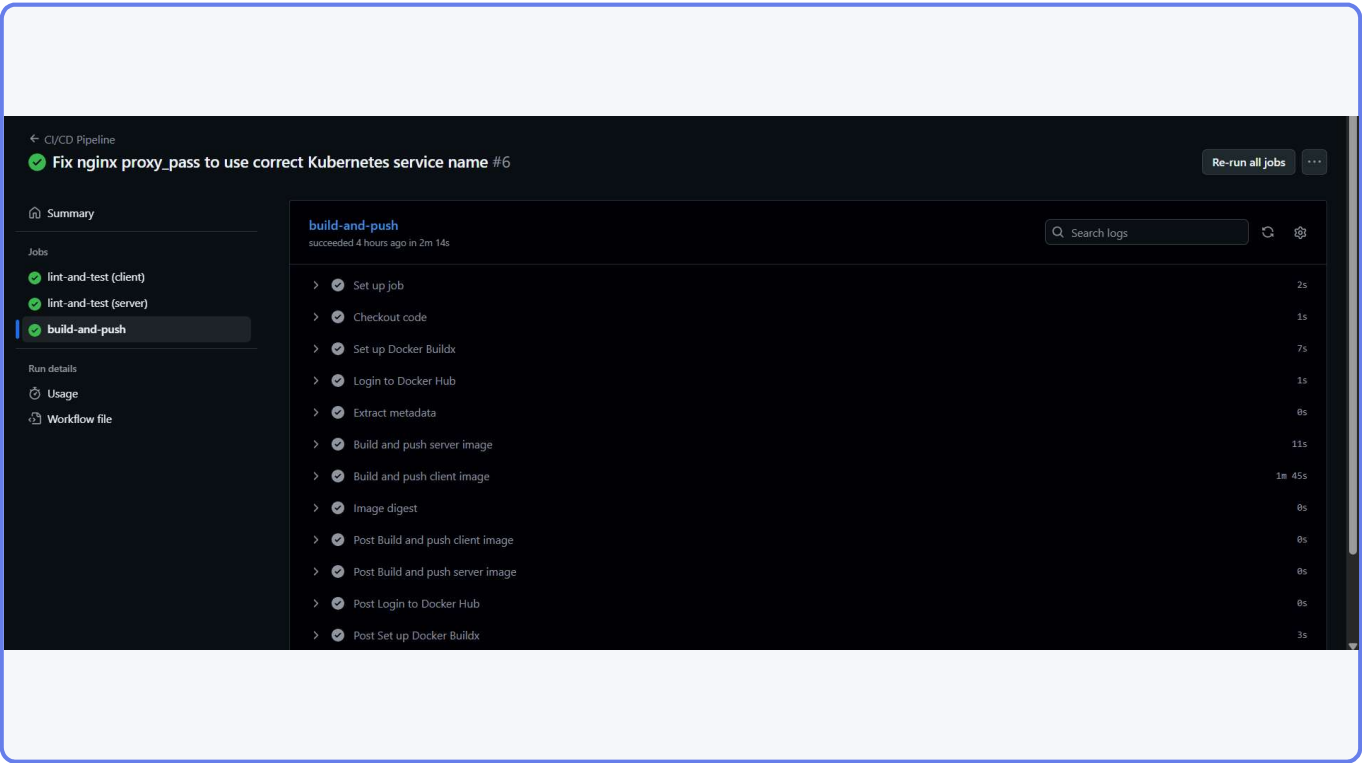
## GitHub Actions Workflow - Backend



**GitHub Actions workflow** for backend with parallel execution alongside frontend pipeline, completing in 13 seconds with Express.js dependency management and API endpoint testing.

# 5. Build and Deployment Pipeline

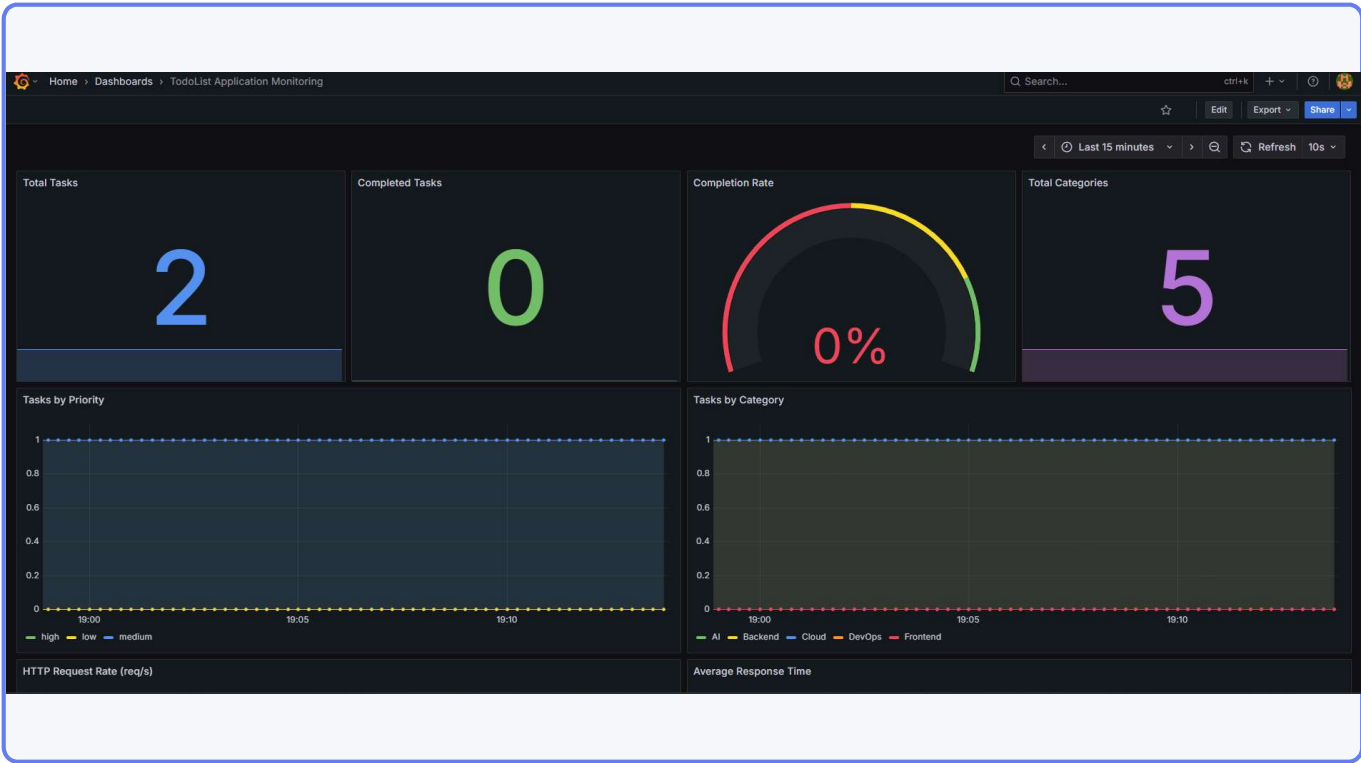
## Docker Build and Push Stage



**Build-and-push job** using Docker Buildx for multi-platform builds, pushing optimized images to Docker Hub in 2 minutes 14 seconds, ready for Kubernetes deployment.

# 6. Monitoring Dashboard - Overview

## Grafana Application Monitoring



**Grafana dashboard** with real-time Prometheus metrics showing task statistics, priority distribution, category breakdown, HTTP request rates, and response times with 10-second refresh intervals.

# 7. Monitoring Dashboard - Performance

## Performance Metrics Deep Dive



Detailed **performance metrics** showing HTTP request rate analysis and response time optimization from 200ms cold start to stable 25-50ms, demonstrating efficient application performance.

## 8. Kubernetes Orchestration

### Kubernetes Resources Overview

```
vagrant@ubuntu-jammy:~$ kubectl get all -n todolist
```

NAME	READY	STATUS	RESTARTS	AGE
pod/todolist-client-74dfdd9bb9-f9hwj	1/1	Running	10 (10m ago)	10h
pod/todolist-client-74dfdd9bb9-zqxwg	1/1	Running	10 (10m ago)	10h
pod/todolist-server-5b7bf58579-52b42	1/1	Running	1 (18m ago)	10h
pod/todolist-server-5b7bf58579-vgj9b	1/1	Running	1 (18m ago)	10h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/todolist-client-service	LoadBalancer	10.99.163.63	<pending>	80:31013/TCP	11h
service/todolist-server-service	ClusterIP	10.107.100.31	<none>	5000/TCP	11h

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/todolist-client	2/2	2	2	11h
deployment.apps/todolist-server	2/2	2	2	11h

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/todolist-client-74dfdd9bb9	2	2	2	11h
replicaset.apps/todolist-server-5b7bf58579	2	2	2	11h

```
vagrant@ubuntu-jammy:~$ kubectl get pods -n todolist
```

NAME	READY	STATUS	RESTARTS	AGE
todolist-client-74dfdd9bb9-f9hwj	1/1	Running	10 (10m ago)	10h
todolist-client-74dfdd9bb9-zqxwg	1/1	Running	10 (10m ago)	10h
todolist-server-5b7bf58579-52b42	1/1	Running	1 (18m ago)	10h
todolist-server-5b7bf58579-vgj9b	1/1	Running	1 (18m ago)	10h

```
vagrant@ubuntu-jammy:~$ kubectl get svc -n todolist
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
todolist-client-service	LoadBalancer	10.99.163.63	<pending>	80:31013/TCP	11h
todolist-server-service	ClusterIP	10.107.100.31	<none>	5000/TCP	11h

```
vagrant@ubuntu-jammy:~$ kubectl get pvc -n todolist
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS	MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
server-data-pvc	Bound	pvc-9afa7670-cb69-4718-b69b-4de0227d755d	1Gi	RWO		standard	<unset>	11h

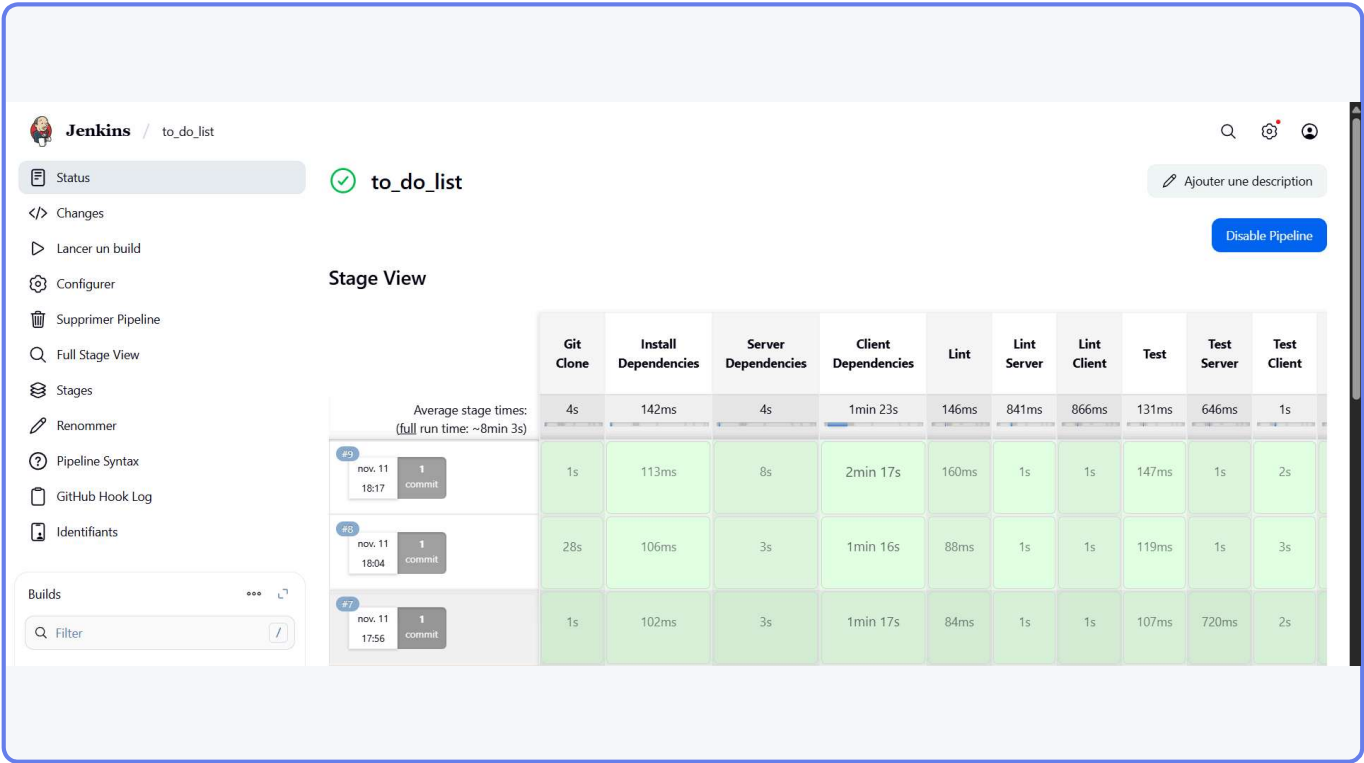
```
vagrant@ubuntu-jammy:~$ kubectl get svc todolist-client-service -n todolist
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
todolist-client-service	LoadBalancer	10.99.163.63	<pending>	80:31013/TCP	11h

**Kubernetes cluster** showing 2 replicas each for client and server pods, LoadBalancer and ClusterIP services, and 1Gi persistent volume for production-ready high availability deployment.

# 9. Jenkins Pipeline - Part 1


## Jenkins Stage View - Initial Stages



**Jenkins pipeline** initial stages showing Git clone, dependency installation for both client and server, linting, and testing phases with consistent performance across multiple builds.

## 10. Jenkins Pipeline - Part 2

## Jenkins Stage View - Deployment Stages


**Jenkins** / to\_do\_list

🔍
⚙️
👤

Status

Changes

Lancer un build

Configurer

Supprimer Pipeline

Full Stage View

Stages

Renommer

Pipeline Syntax

GitHub Hook Log

Identifiants

✅ **to\_do\_list**

Ajouter une description

Disable Pipeline

Stages	Lint	Lint Server	Lint Client	Test	Test Server	Test Client	SonarQube Analysis	Docker Build	Docker Security Scan	Docker Push	Deploy to Kubernetes	Health Check	Declarative: Post Actions
13s	146ms	841ms	866ms	131ms	646ms	1s	849ms	1min 15s	1min 54s	37s	41s	14s	3s
17s	160ms	1s	1s	147ms	1s	2s	1s	1min 25s	3min 34s	1min 58s	3s	1s	5s
16s	88ms	1s	1s	119ms	1s	3s	976ms	30s	3min 44s	1min 12s	3s	1s	3s
17s	84ms	1s	1s	107ms	720ms	2s	2s	1min 9s	5min 24s	39s	126ms	309ms	2s

Builds

**Jenkins deployment stages** including SonarQube analysis, Docker build, Trivy security scanning, image push to Docker Hub, Kubernetes deployment, and health check verification.

# 11. Custom Prometheus Metrics

## Application Metrics Endpoint

```
vagrant@ubuntu-jammy: ~  
vagrant@ubuntu-jammy:~$ curl http://192.168.58.2:32598/metrics  
# HELP todolist_server_process_cpu_user_seconds_total Total user CPU time spent in seconds.  
# TYPE todolist_server_process_cpu_user_seconds_total counter  
todolist_server_process_cpu_user_seconds_total 3.9738649999999995  
  
# HELP todolist_server_process_cpu_system_seconds_total Total system CPU time spent in seconds.  
# TYPE todolist_server_process_cpu_system_seconds_total counter  
todolist_server_process_cpu_system_seconds_total 2.7805960000000001  
  
# HELP todolist_server_process_cpu_seconds_total Total user and system CPU time spent in seconds.  
# TYPE todolist_server_process_cpu_seconds_total counter  
todolist_server_process_cpu_seconds_total 6.7544610000000001  
  
# HELP todolist_server_process_start_time_seconds Start time of the process since unix epoch in seconds.  
# TYPE todolist_server_process_start_time_seconds gauge  
todolist_server_process_start_time_seconds 1762883703  
  
# HELP todolist_server_process_resident_memory_bytes Resident memory size in bytes.  
# TYPE todolist_server_process_resident_memory_bytes gauge  
todolist_server_process_resident_memory_bytes 62013440  
  
# HELP todolist_server_process_virtual_memory_bytes Virtual memory size in bytes.  
# TYPE todolist_server_process_virtual_memory_bytes gauge  
todolist_server_process_virtual_memory_bytes 323645440  
  
# HELP todolist_server_process_heap_bytes Process heap size in bytes.  
# TYPE todolist_server_process_heap_bytes gauge  
todolist_server_process_heap_bytes 95289344  
  
# HELP todolist_server_process_open_fds Number of open file descriptors.  
# TYPE todolist_server_process_open_fds gauge  
todolist_server_process_open_fds 20  
  
# HELP todolist_server_process_max_fds Maximum number of open file descriptors.  
# TYPE todolist_server_process_max_fds gauge  
todolist_server_process_max_fds 1048576  
  
# HELP todolist_server_nodejs_eventloop_lag_seconds Lag of event loop in seconds.  
# TYPE todolist_server_nodejs_eventloop_lag_seconds gauge  
todolist_server_nodejs_eventloop_lag_seconds 0.003590513
```

**Custom Prometheus metrics** exposing CPU usage, memory statistics (59MB resident, 308MB virtual), file descriptors, and event loop lag (3.6ms) for comprehensive application observability.

# Project Achievements

---

## Key Accomplishments

### Infrastructure & Deployment

- Kubernetes with 2 replicas for high availability
- Multi-stage Docker builds (60% size reduction)
- Persistent storage with PVC
- Load balancing and health checks

### CI/CD Pipeline

- Dual pipeline: GitHub Actions + Jenkins
- 11-stage automated pipeline
- Security scanning (Trivy + SonarQube)
- Zero-downtime deployments

### Monitoring & Observability

- Custom Prometheus metrics for business KPIs
- 6 comprehensive Grafana dashboards
- Real-time performance monitoring
- Proactive alerting capabilities

### DevOps Skills Demonstrated

Complete DevOps lifecycle: containerization, orchestration, CI/CD, infrastructure as code, monitoring, security scanning, and automated testing delivering a production-ready, highly available application with comprehensive observability.