



# Grafana Dashboard - Quick Guide

---



## Access Information

- **Grafana URL:** `http://localhost:3000`
  - **Username:** `admin`
  - **Password:** `admin` (you'll be prompted to change it on first login)
- 



## Getting Started

### Step 1: Login to Grafana

1. Open `http://localhost:3000` in your browser
2. Login with username `admin` and password `admin`
3. (Optional) Change the password when prompted, or click "Skip"

### Step 2: Import the Dashboard

#### Option A: Automatic Import (Recommended)

The dashboard has been automatically configured! Just navigate to:

1. Click on the "≡" **menu** (top left)
2. Select "**Dashboards**"
3. Click on "**Heart Disease ML API Monitoring**"

#### Option B: Manual Import

If the dashboard doesn't appear:

1. Click "≡" → "**Dashboards**" → "**Import**"
  2. Click "**Upload JSON file**"
  3. Select: `deployment/grafana/dashboards/heart_disease_dashboard.json`
  4. Click "**Load**" then "**Import**"
- 



## Dashboard Panels Explained

### 1. Total Predictions (Top Left)

- Shows total number of predictions made since server started
- Real-time counter that increases with each API call

### 2. Prediction Rate (Top Center)

- Predictions per minute
- Helps identify usage patterns and peak times

### 3. Average Prediction Time (Top Right)

- Average latency for predictions in seconds
- Should typically be < 100ms for good performance

### 4. Predictions Over Time (Row 2, Left)

- Line graph showing prediction requests over time
- Helps visualize traffic patterns
- Useful for capacity planning

### 5. Prediction Latency Distribution (Row 2, Right)

- Histogram showing response time distribution
- Most predictions should be in the fastest buckets (left side)
- Outliers on the right indicate slow requests

### 6. Prediction Results Distribution (Row 3, Left)

- Pie chart showing Disease (1) vs No Disease (0) predictions
- Helps understand the prediction distribution
- Useful for detecting data drift

### 7. Risk Level Distribution (Row 3, Center)

- Distribution of Low/Medium/High risk predictions
- Based on prediction confidence levels

### 8. API Health Status (Row 3, Right)

- Real-time health check status
- Shows if the API is up (1) or down (0)

### 9. System Metrics (Bottom Row)

- **CPU Time:** Shows server resource usage
- **Memory Usage:** Tracks Python process memory
- **Active Requests:** Number of concurrent requests

---

## Troubleshooting

### No Data Showing in Dashboard?

#### 1. Check if API is running:

```
curl http://localhost:8000/health
```

#### 2. Check if Prometheus is scraping:

```
curl http://localhost:9090/api/v1/targets
```

Look for status: "up"

### 3. Generate some traffic:

```
cd
/Users/v0s01jh/Documents/BITS/ML0psExperimentalLearning_Assignment_1_Group_81/heart-disease-mlops
./generate_traffic.sh
```

### 4. Verify Prometheus is collecting metrics:

```
curl "http://localhost:9090/api/v1/query?
query=prediction_requests_total"
```

Grafana shows "No data"

- **Wait 10-20 seconds** for Prometheus to scrape new metrics
- **Adjust time range:** Click the time picker (top right) and select "Last 5 minutes"
- **Refresh dashboard:** Click the refresh icon (top right) or enable auto-refresh

Data source connection error

1. Go to "≡" → "**Connections**" → "**Data sources**"
2. Click on "**Prometheus**"
3. Verify URL is: `http://localhost:9090`
4. Scroll down and click "**Save & test**"

---

## Customizing the Dashboard

### Change Time Range

- Click the **time picker** (top right corner)
- Select predefined ranges or create custom ones
- Enable **auto-refresh** (5s, 10s, 30s, etc.)

### Add New Panels

1. Click "**Add**" button (top right) → "**Visualization**"
2. Select "**Prometheus**" as data source
3. Enter your PromQL query
4. Choose visualization type (Graph, Gauge, Stat, etc.)
5. Click "**Apply**"

## Useful PromQL Queries

```
# Total predictions
prediction_requests_total

# Prediction rate (per minute)
rate(prediction_requests_total[1m]) * 60

# Average latency (seconds)
rate(prediction_latency_seconds_sum[5m]) /
rate(prediction_latency_seconds_count[5m])

# 95th percentile latency
histogram_quantile(0.95, rate(prediction_latency_seconds_bucket[5m]))

# Predictions by result
sum by (prediction) (prediction_results_total)

# Error rate
rate(http_requests_total{status="500"}[5m])
```

---

## Taking Screenshots for Assignment

### Recommended Screenshots

1. **Overview Dashboard** (Full page)
  - Shows all panels with live data
  - Set time range to "Last 5 minutes"
2. **Prediction Trends** (Zoom into top graphs)
  - Shows API usage patterns
  - Demonstrates monitoring capability
3. **System Metrics** (Bottom panels)
  - CPU and Memory usage
  - Proves resource monitoring
4. **Grafana Sidebar** (Menu with dashboard list)
  - Shows dashboard organization
  - Professional setup

### Tips for Better Screenshots

- Enable dark theme: **User icon → Preferences → UI Theme → Dark**
- Set time range to show data: **Last 5 minutes** or **Last 15 minutes**
- Enable auto-refresh: **5s** or **10s**

- Run `./generate_traffic.sh` before taking screenshots for live data
  - Use full-screen mode: Click panel title → **View**
- 

## Maintaining the Setup

### Start All Services

```
# 1. Start FastAPI
cd
/Users/v0s01jh/Documents/BITS/ML0psExperimentalLearning_Assignment_1_Group
_81/heart-disease-mlops
source venv/bin/activate
PYTHONPATH=. python src/app.py &

# 2. Start Prometheus
prometheus --config.file=deployment/prometheus/prometheus.yml \
  --storage.tsdb.path=./prometheus-data \
  --web.enable-lifecycle &

# 3. Start Grafana
brew services start grafana

# 4. Generate traffic (optional)
./generate_traffic.sh &
```

### Stop All Services

```
# Stop FastAPI
kill -f "python src/app.py"

# Stop Prometheus
kill -f prometheus

# Stop Grafana
brew services stop grafana

# Stop traffic generator
kill -f generate_traffic.sh
```

### Check Service Status

```
# Check API
curl http://localhost:8000/health

# Check Prometheus
curl http://localhost:9090/-/healthy
```

```
# Check Grafana
curl http://localhost:3000/api/health
```

---

## Learning Resources

- **Grafana Docs:** <https://grafana.com/docs/grafana/latest/>
- **Prometheus Query Language:** <https://prometheus.io/docs/prometheus/latest/querying/basics/>
- **Dashboard Best Practices:** <https://grafana.com/docs/grafana/latest/dashboards/build-dashboards/best-practices/>

---

## Success Checklist

- ☐ Grafana accessible at <http://localhost:3000>
- ☐ Dashboard imported and visible
- ☐ All panels showing data
- ☐ Time range set to "Last 5 minutes"
- ☐ Auto-refresh enabled
- ☐ Traffic generator running
- ☐ Screenshots taken for assignment
- ☐ Understanding of each panel's purpose

---

## Your Grafana monitoring setup is complete!

You now have professional-grade monitoring for your ML API, just like production systems!