**Docker Web Server Deployment Guide Using AWS**

Table of Contents  
  
 [Prerequisites](#prerequisites)

 [AWS EC2 Instance Setup](#aws-ec2-instance-setup)

 [Security Configuration](#security-configuration)

 [EC2 Instance Connection](#ec2-instance-connection)

 [System Preparation](#system-preparation)

 [Docker Installation](#docker-installation)

 [Project Deployment](#project-deployment)

 [Monitoring and Logging](#monitoring-and-logging)

**Prerequisites**

**Required Tools**

* AWS Account with appropriate permissions
* SSH client (Terminal on Mac/Linux, PuTTY on Windows)
* Text editor (VS Code recommended)
* Docker Engine (Docker and Docker-Compose)

**Required AWS Permissions**

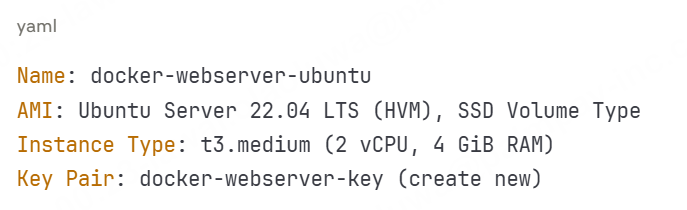
* EC2 Full Access
* VPC management permissions
* Security Group management
* Elastic IP allocation (optional)

**AWS EC2 Instance Setup**

**Step 1: Launch EC2 Instance**

1. **Navigate to EC2 Console**
   * Log into AWS Management Console
   * Navigate to EC2 service
   * Click "Launch Instance"

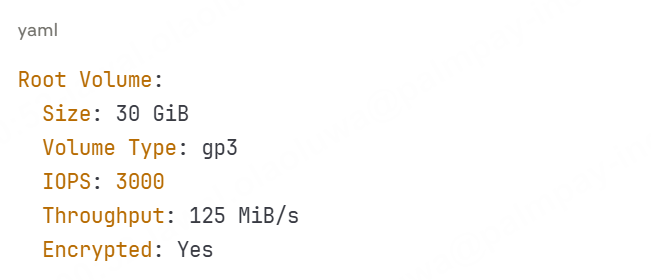
**Instance Configuration**

****

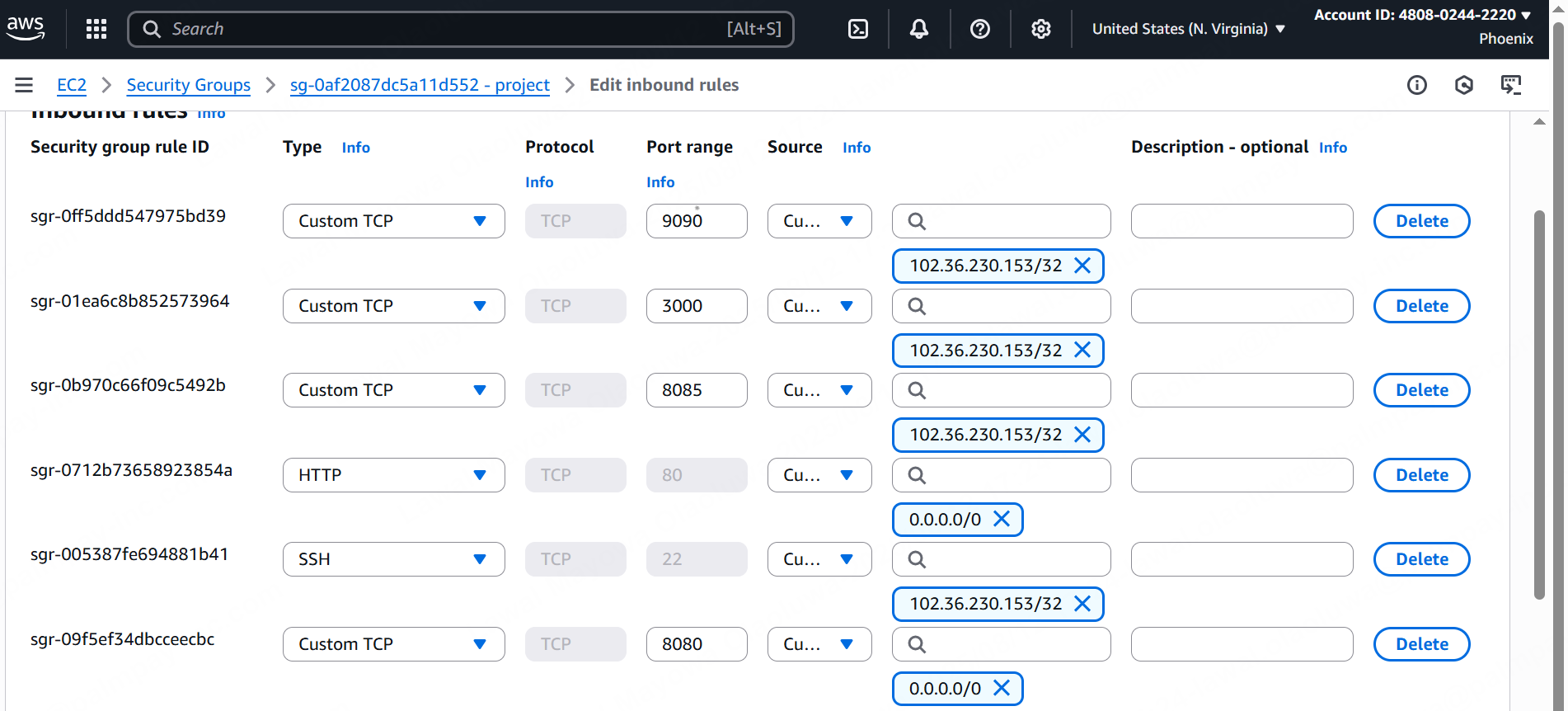
**Network settings**



**Storage Configuration**

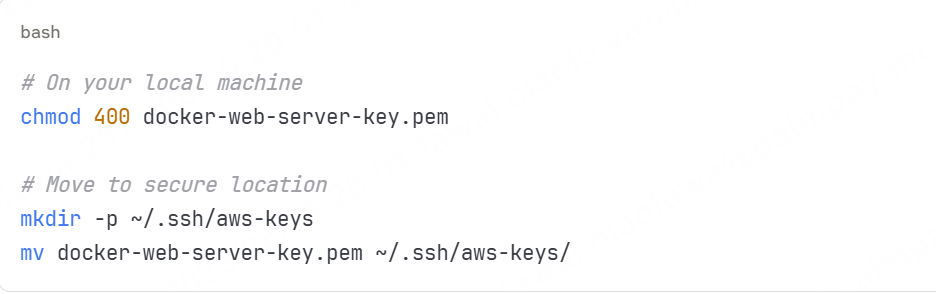
****

**Security Group Rules**

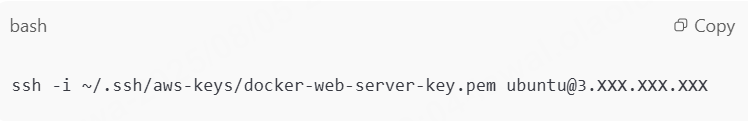
****

**Security Configuration**

Key Pair Security:

****

Connect to Instance:



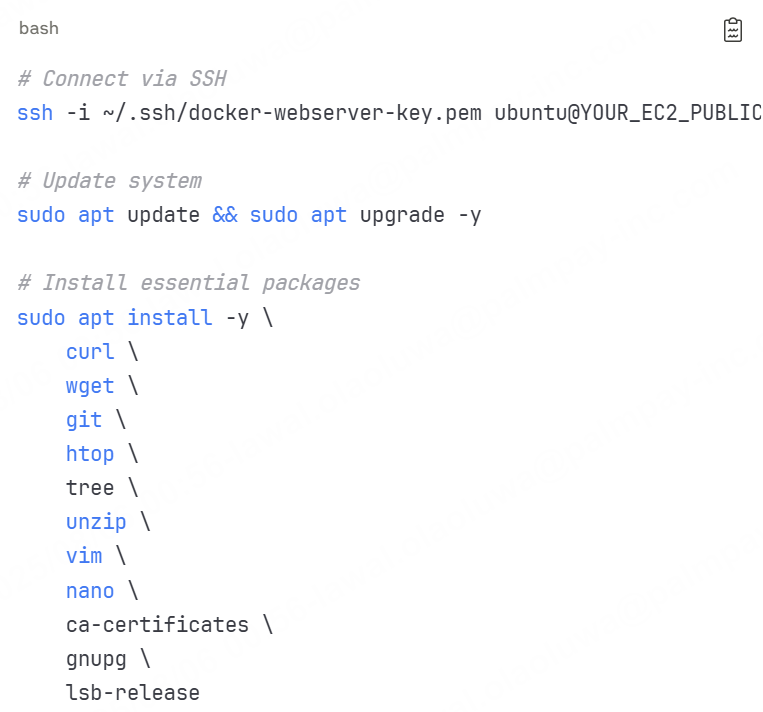
Alternative with DNS Name

ssh -i ~/.ssh/aws-keys/docker-web-server-key.pem ubuntu@ec2-3-XXX-XXX-XXX.region.compute.amazonaws.com

Initial Connection Verification:

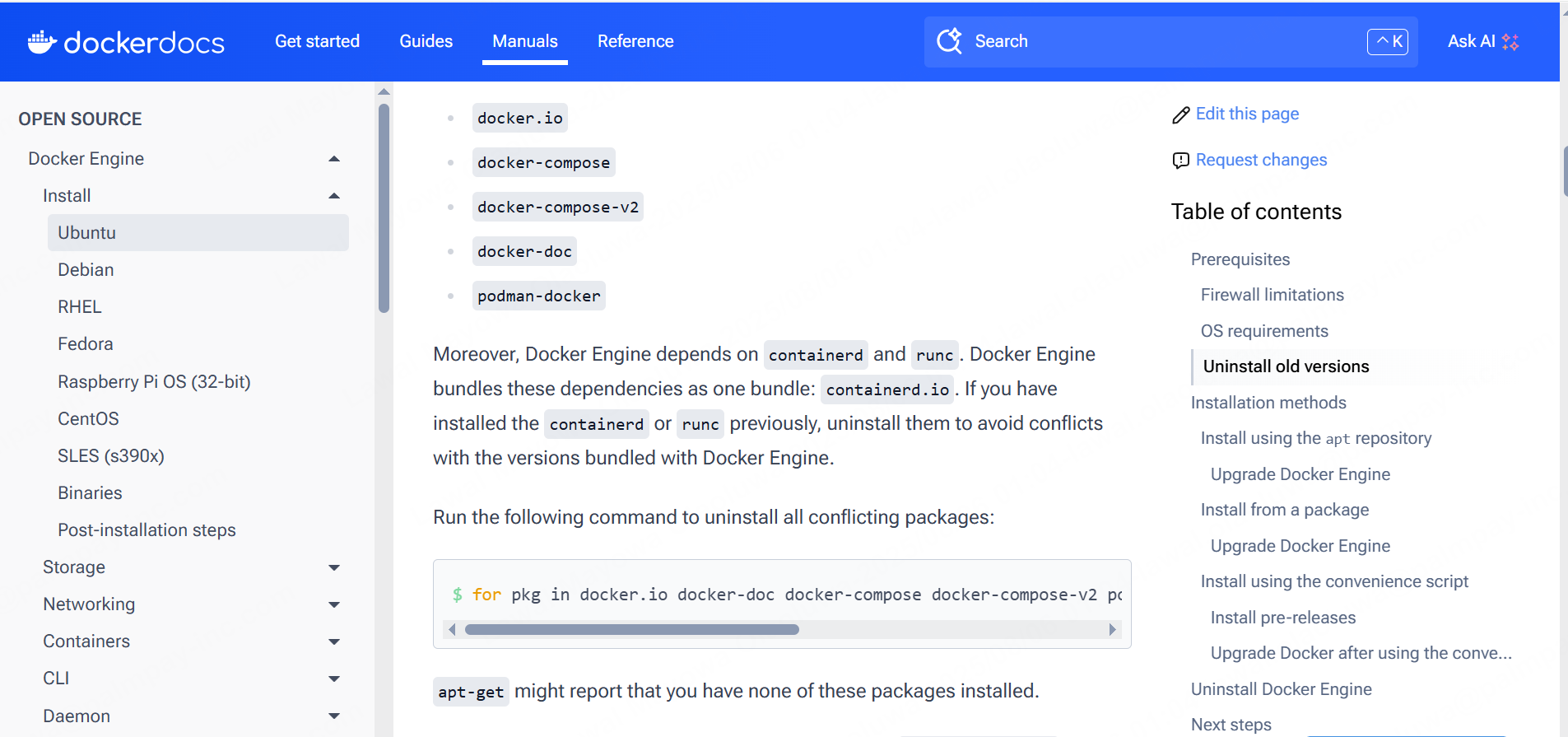


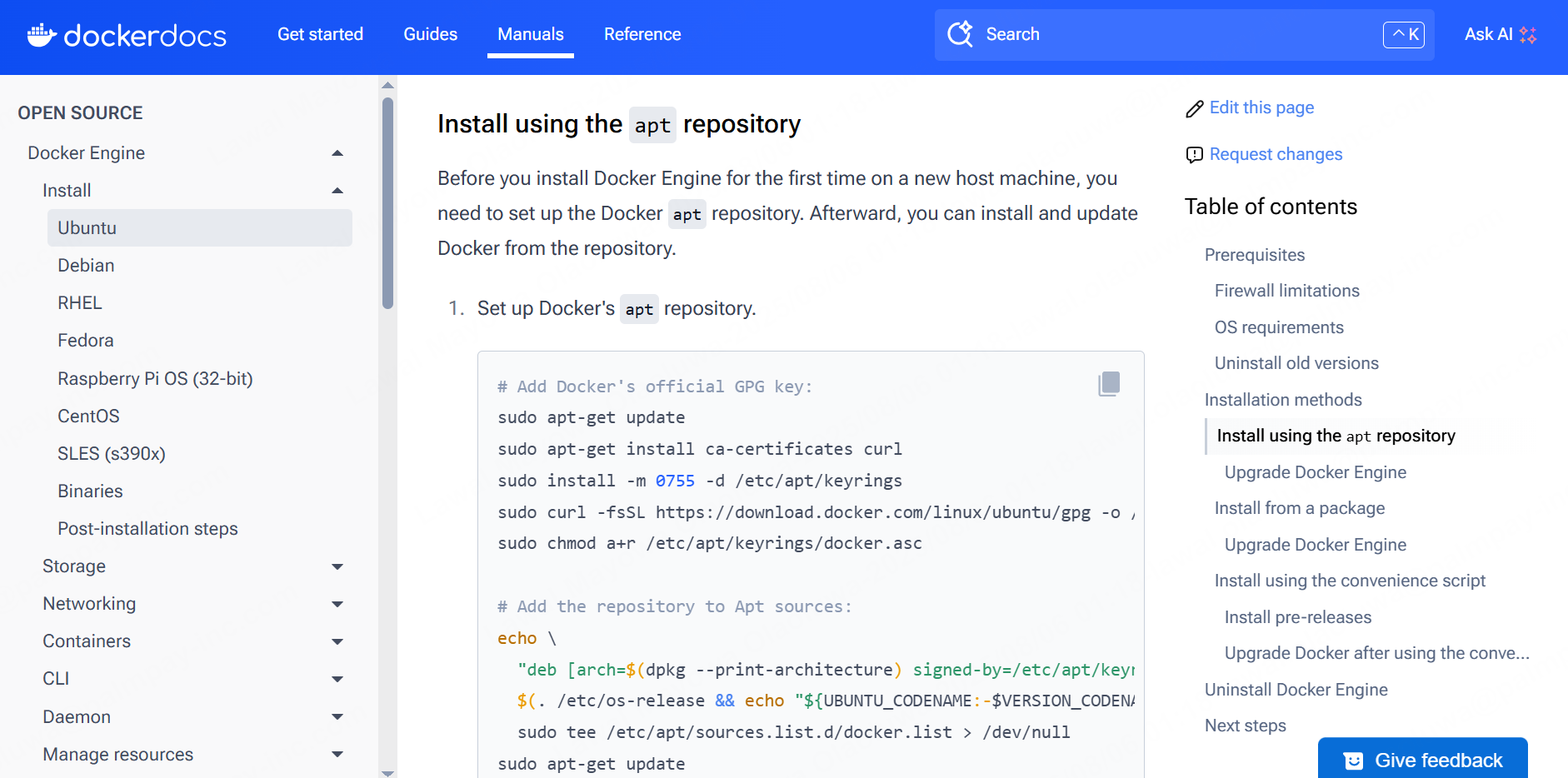
**Connect to Ubuntu Instance**



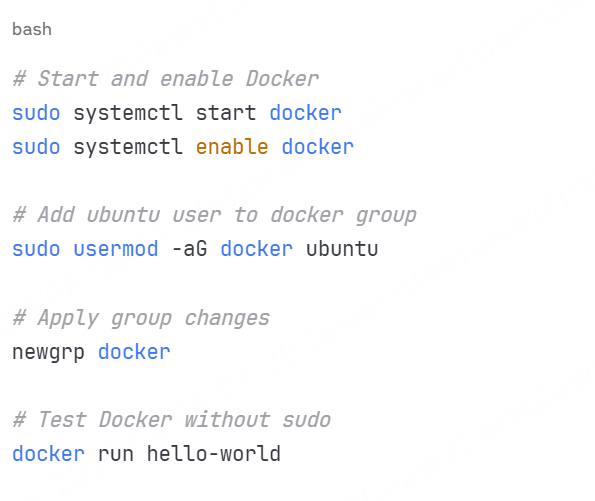
**Docker Installation & Configuration**

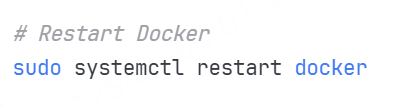
**Install Docker Engine**

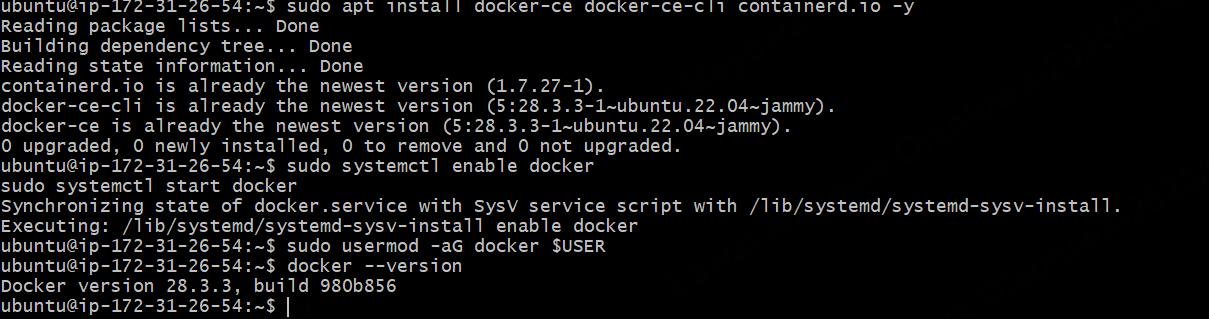
****

****

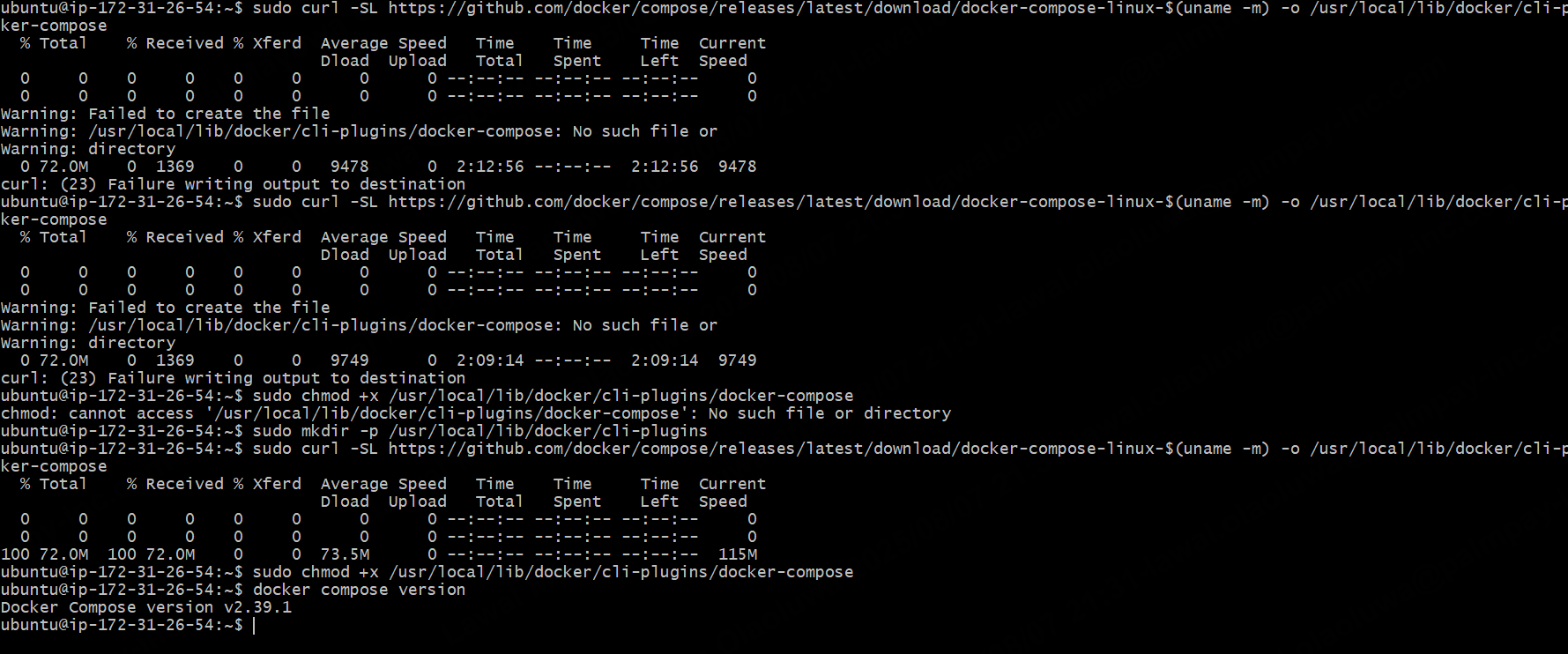
**Configure Docker**

****

****

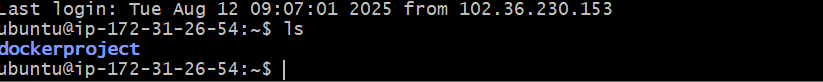
****

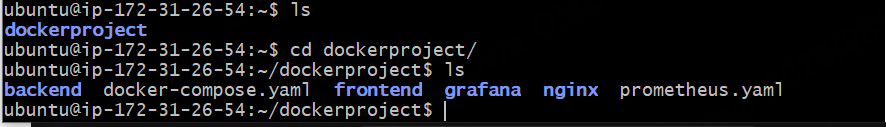
**Download docker-compose.**



**Project Deployment**

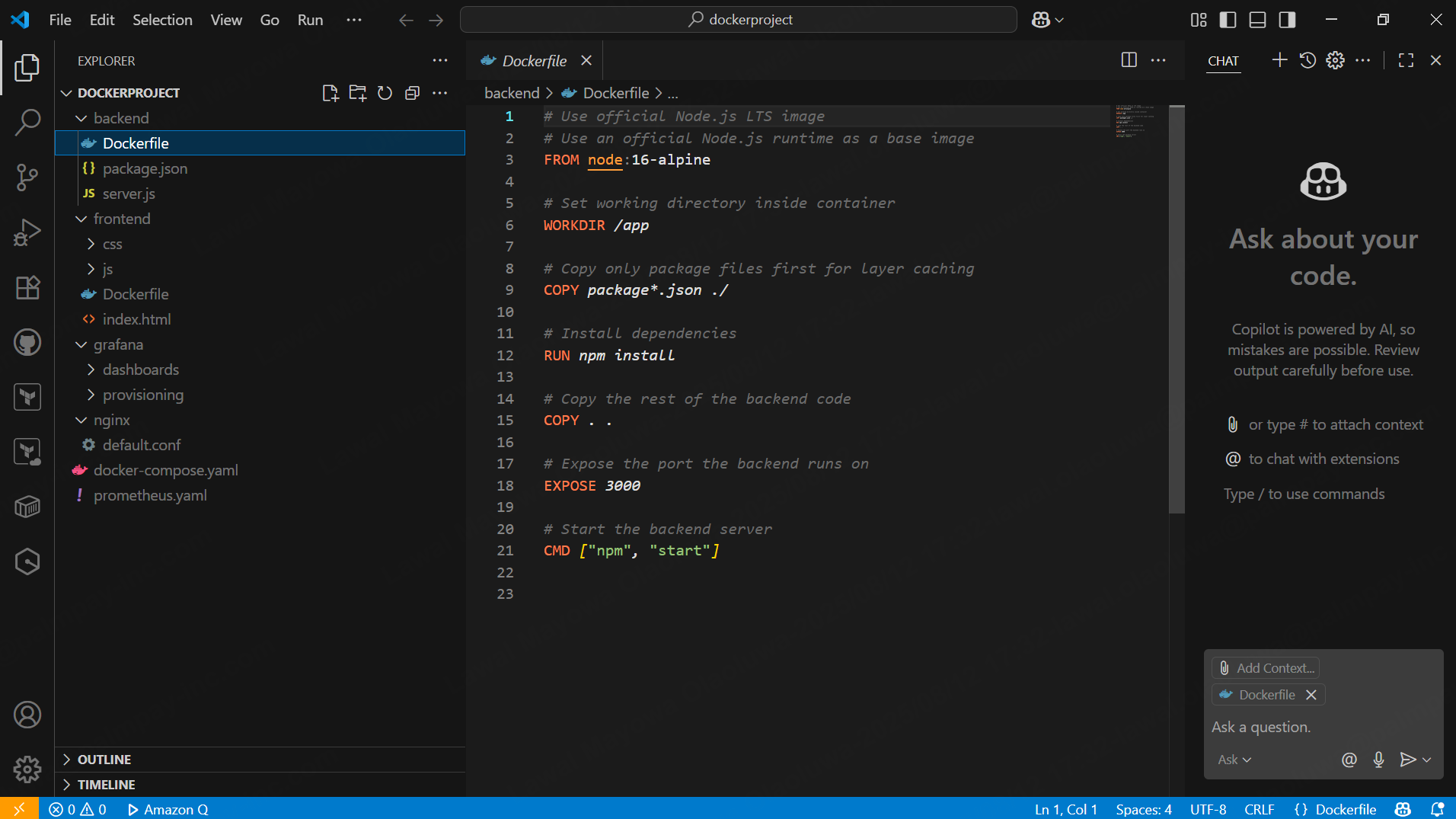
**Create a folder “dockerproject”**

****

****

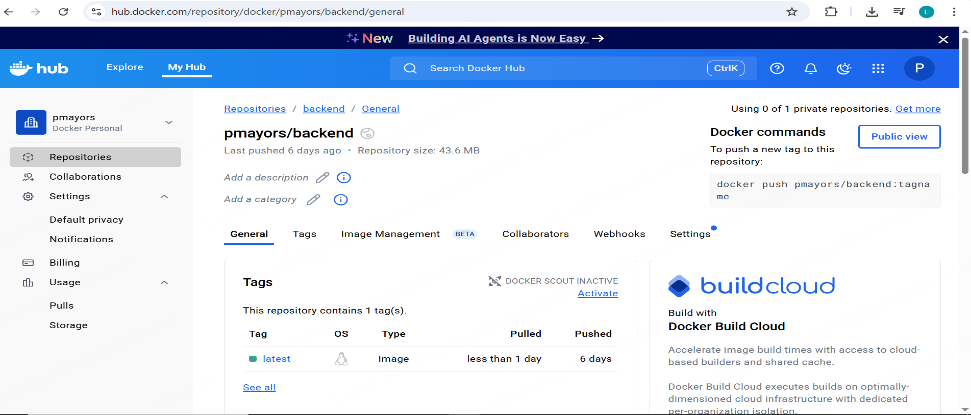
**backend folder:**

* **Dockerfile** - Container build instructions specifically for the backend service, **package.json and** **server.js**, dependencies installation, and server startup commands

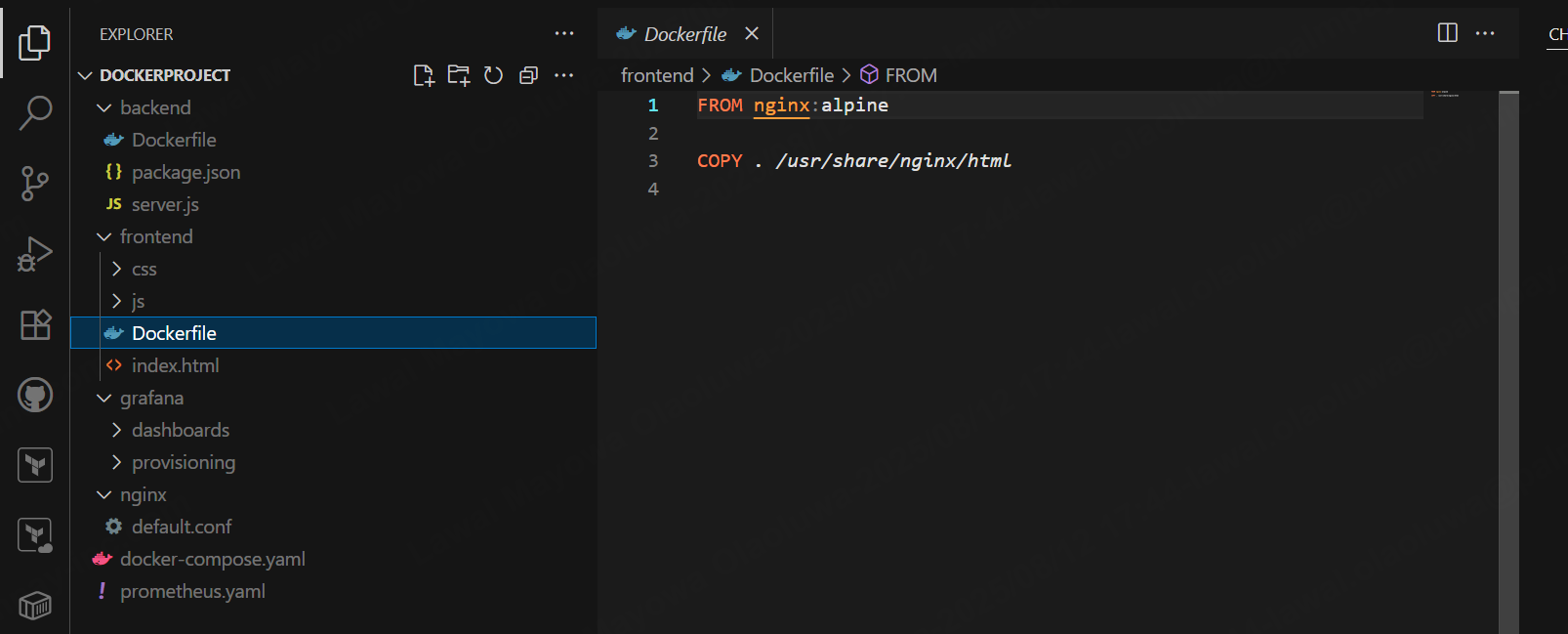
****

* **package.json** - Node.js project configuration containing backend dependencies, scripts, and metadata for the server-side application
* **server.js** - Main backend application entry point, an Express.js server handling API routes, database connections, and business logic

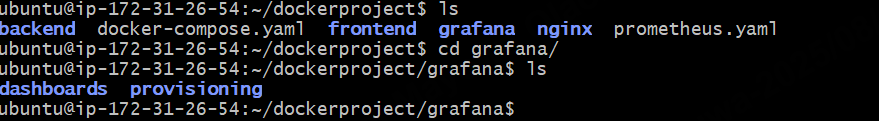
**Note: Backend was pushed to docker hub, and later mention in the docker compose file.**



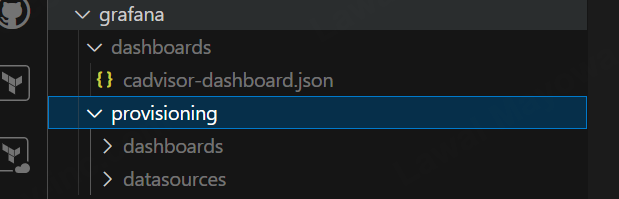
**frontend folder:**

* **css** - Contains stylesheet files for frontend styling, layout, and visual presentation of the web interface
* **js** - JavaScript files handling frontend functionality, user interactions, API calls, and dynamic content manipulation
* **Dockerfile** - Container build instructions for frontend service, setting up a web server (nginx/apache) to serve static files  
  
* **index.html** - Main HTML entry point for the web application, containing the base structure and references to CSS/JS files

**grafana folder:**

****

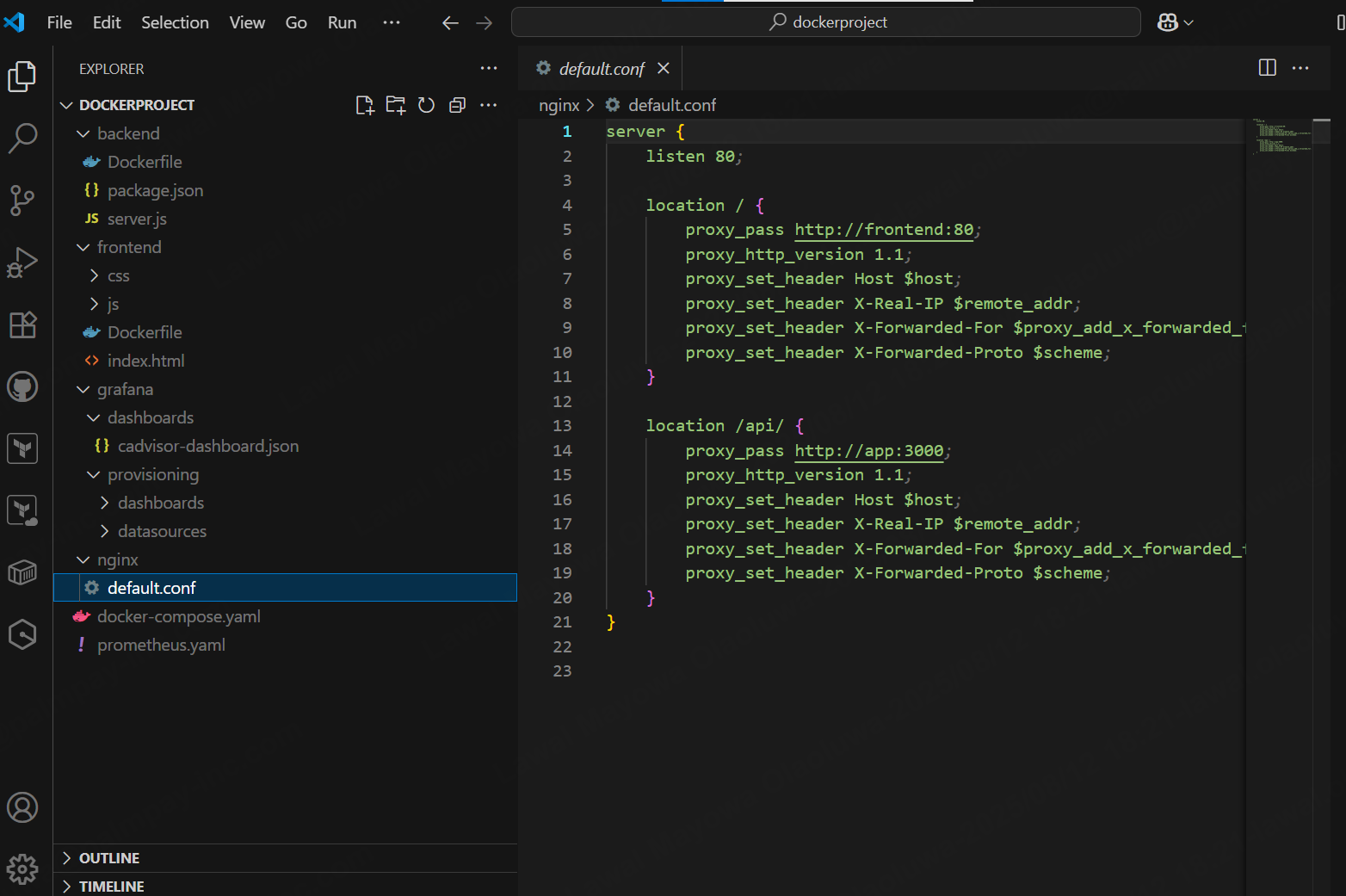
* **dashboards** - Pre-configured JSON dashboard templates for monitoring metrics, visualizing data, and creating custom monitoring views
* **provisioning** - Configuration files for automatically setting up datasources, dashboards, and other Grafana components on container startup

****

**Web Server**

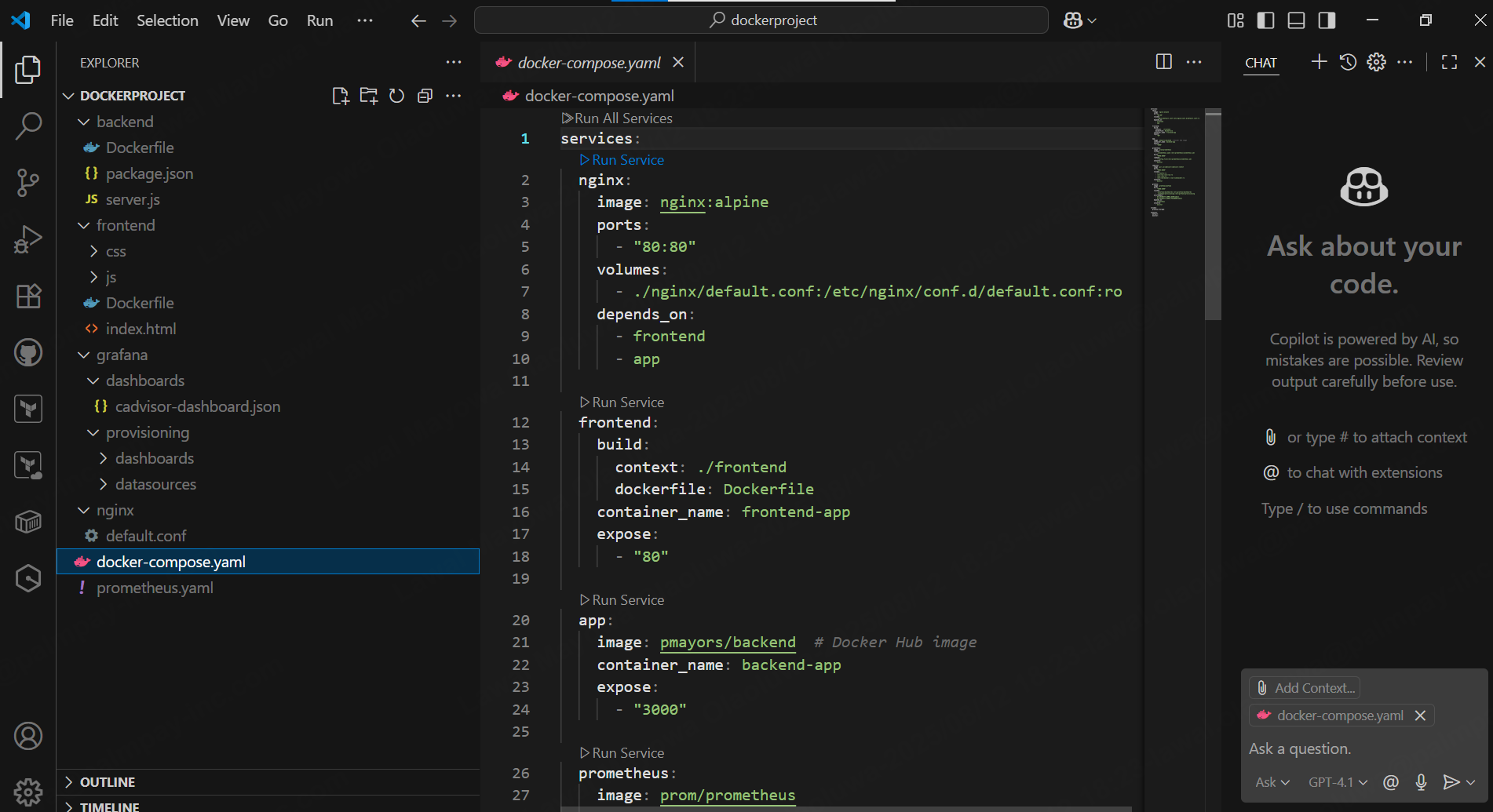
**nginx folder:**

* **default.conf** - Reverse proxy configuration defining routing rules, upstream services, load balancing, and request forwarding between frontend/backend

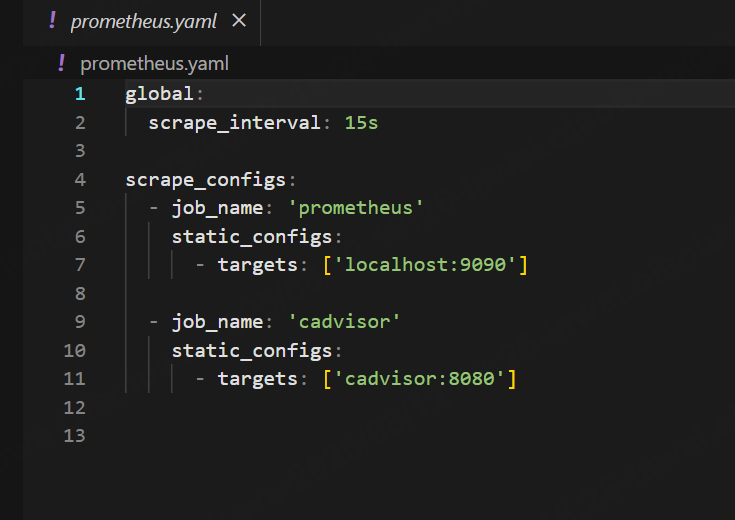


**Configuration files:**

* **docker-compose.yaml** - Multi-container orchestration file defining services, networks, volumes, and dependencies for the entire application stack

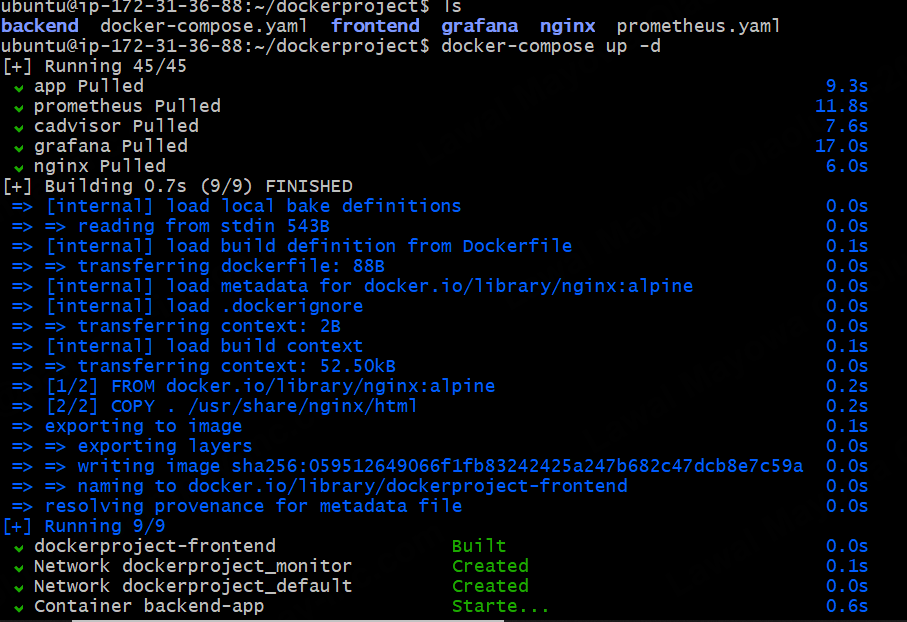


* **prometheus.yml** - Monitoring configuration specifying scrape targets, intervals, and rules for collecting metrics from various services



This is a comprehensive microservices architecture with separate frontend/backend services, monitoring stack, and reverse proxy setup.

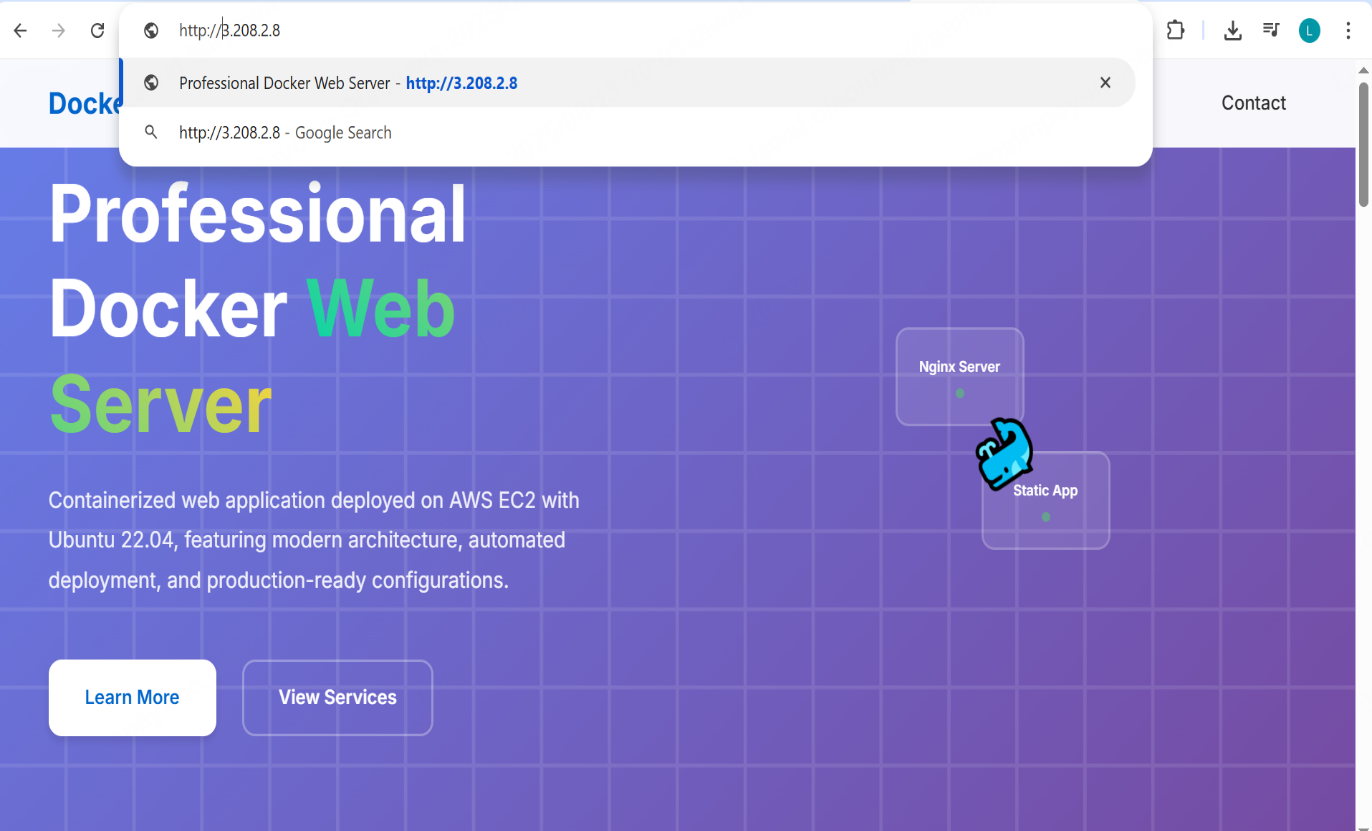
**Launching**

****

Docker compose up -d

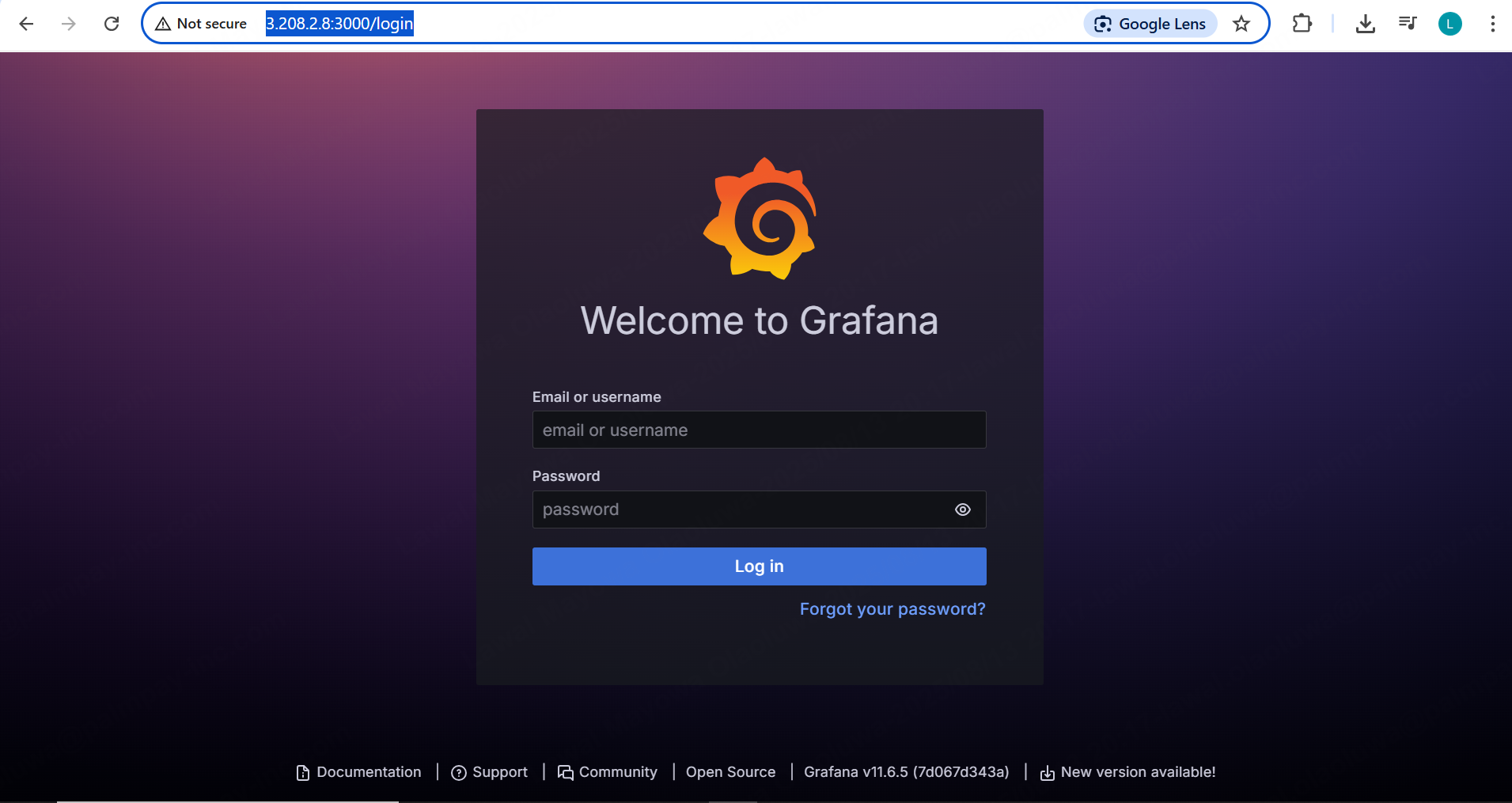
for a Docker multi-service project with **backend, frontend, Grafana, Nginx, Prometheus**, etc.

* **Images** for services (prometheus, app, grafana, nginx) were pulled or built locally.
* The **frontend** image was built from a Dockerfile using nginx:alpine.
* Docker created two networks (dockerproject\_monitor and dockerproject\_default) for service communication.
* The backend-app container was started successfully in detached mode.



Check with the IP of your instance.

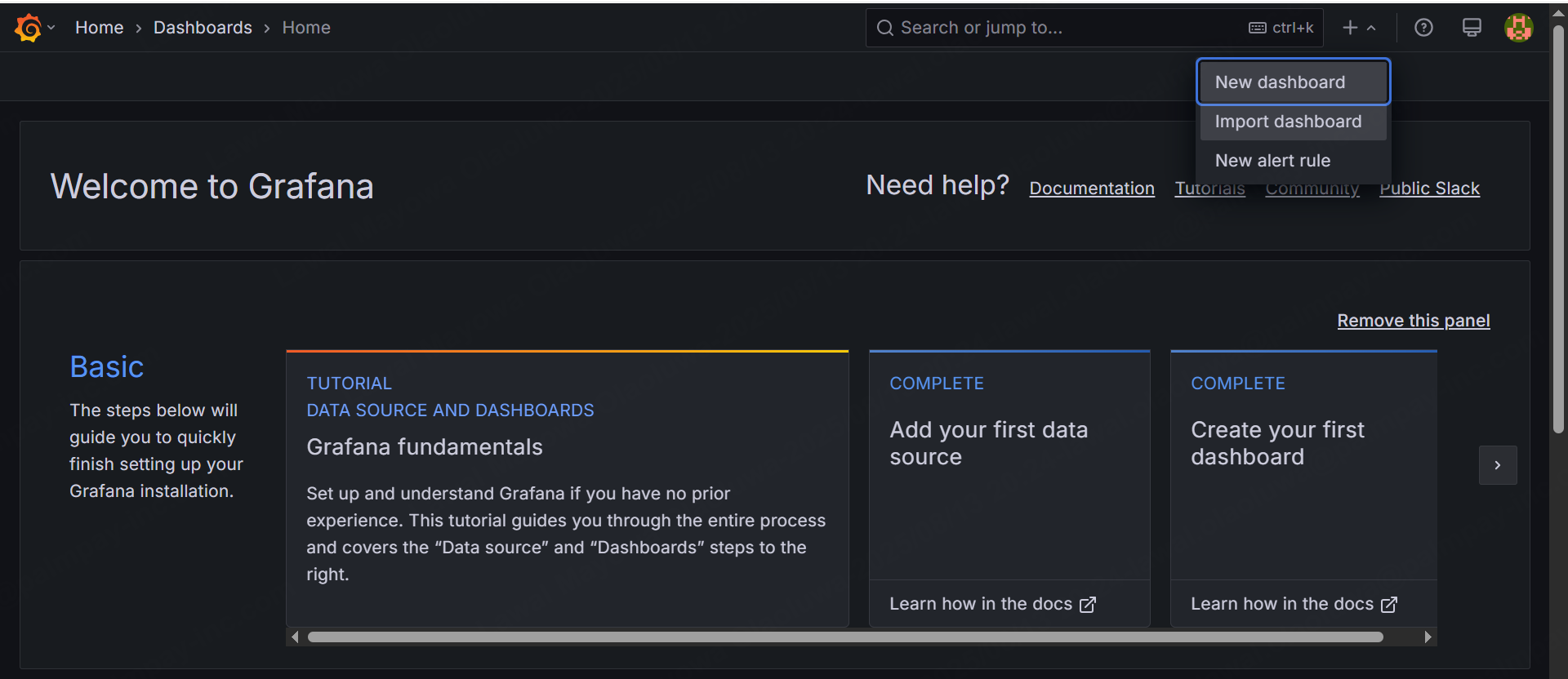
**Steps for monitoring and Visualization steps**.



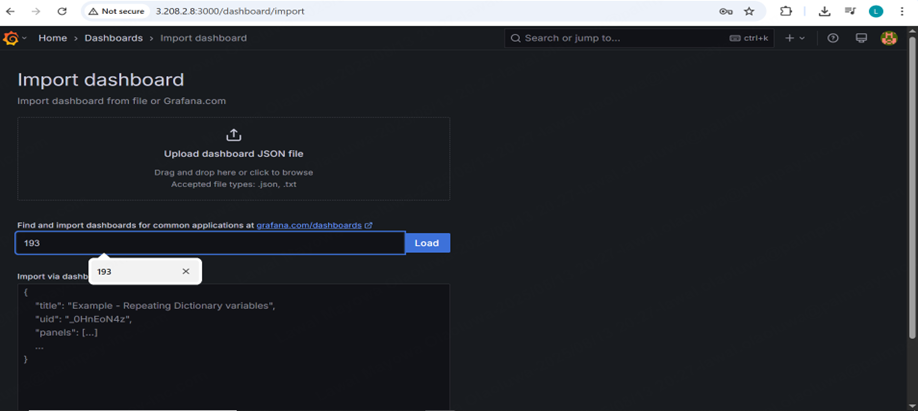
Login to Grafana with <http://yourIpaddress:3000>

Username: admin Password: admin.

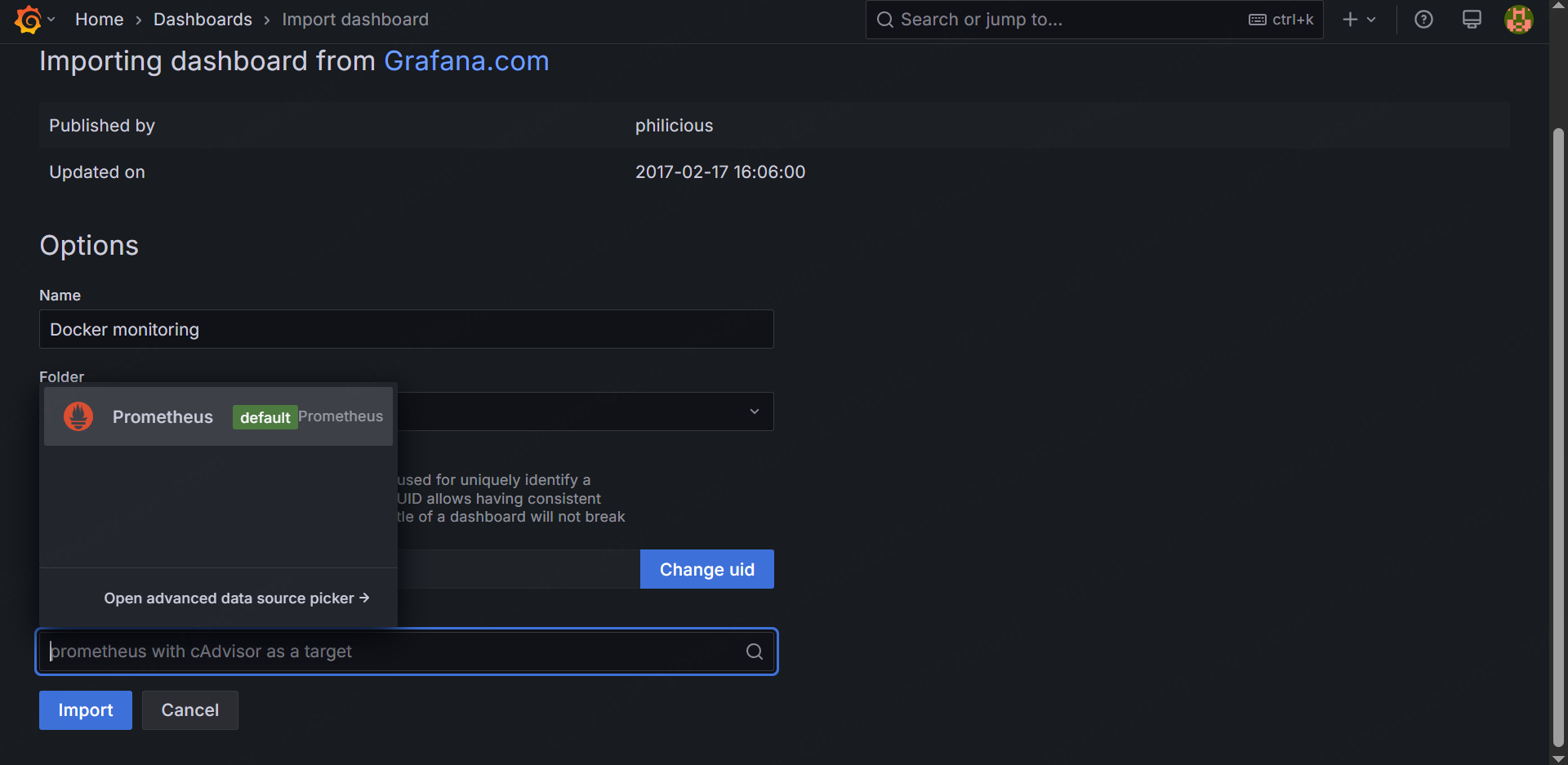
Click on **Import Dashboards**



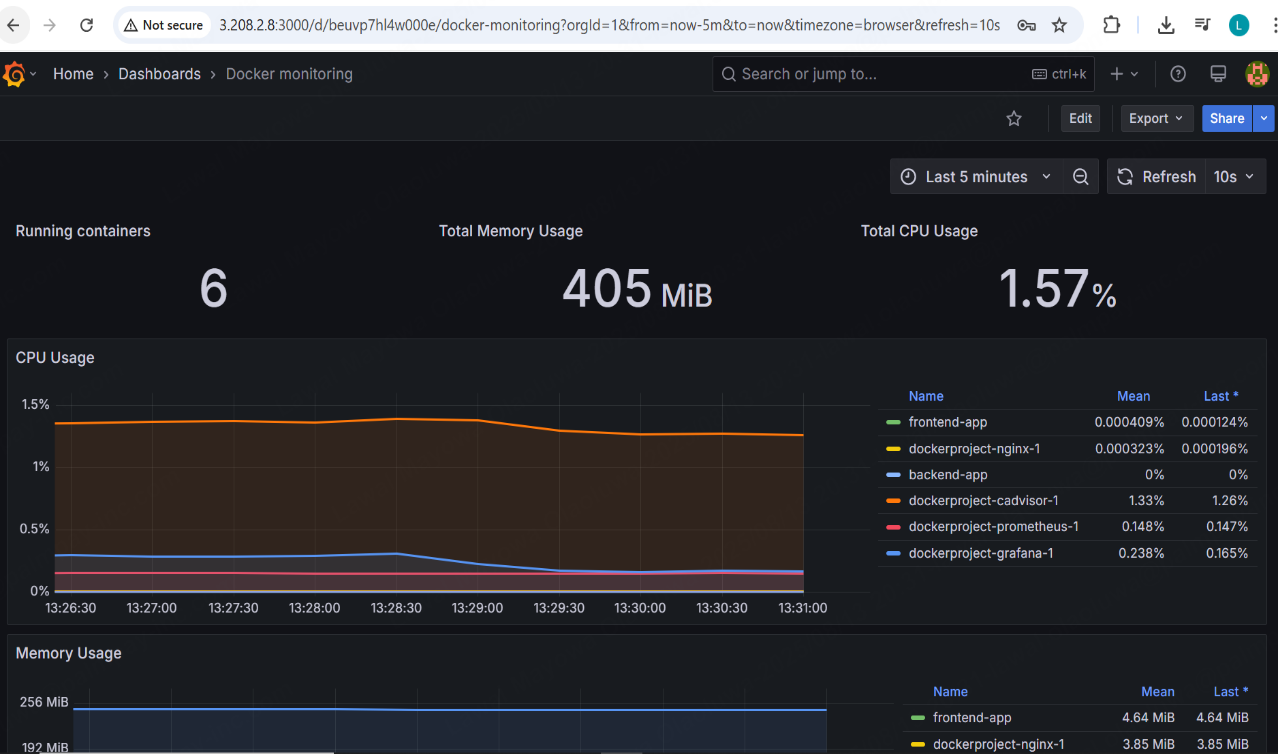
**Imput 193 and click load**



**Click Prometheus and import to see the dashboard**



**Grafana Dashboard**

****

