Task-Management

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Task Management Application Documentation\*\*\*\*\*\*\*\*\*

Overview:

The Task Management Application is a web-based tool developed using the Django framework, designed to help users manage their tasks efficiently. The application allows users to perform key actions such as creating, updating, deleting, and viewing tasks. The backend is powered by Django, and the frontend can be built using HTML, CSS, and JavaScript.

--This documentation provides an overview of the application features, architecture, and guides for setting up and using the application.

Key Features

1.User Authentication:

-Users can register, log in, and log out. -Password management and secure authentication are handled by Django’s authentication system.

2.Task Creation & Management:

Users can create new tasks, specifying details such as title, description, due date, and priority. Tasks can be marked as completed or in progress.

3.Task Editing:

Users can edit the details of existing tasks.

4.Task Deletion:

Users can delete tasks they no longer need.

5.Responsive Design:

The application is designed to be responsive, ensuring a smooth user experience on both mobile and desktop devices.

6. Monitoring:

The system implements monitoring. Metrics for API response times and error rates can be collected and visualized.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SystemArchitecture \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Frontend: -HTML/CSS/JavaScript used -The frontend communicates with the backend using API endpoints.
2. Backend: -The backend is built using Django, a high-level Python web framework. -MySQL is used as the database to store task information, user data, and other related entities.
3. Database: MySQL Database: Stores all the tasks, user details, and related entities. Here I am used AWS RDS service. Endpoint of database, included in database configuration. RDS database must be up and running

-Tables:

-users: Stores user authentication information. -tasks: Stores task-related details such as title, description, due date, created\_at,updated\_at.

1. Deployment: -The application containerized using Docker for simplified deployment. -Jenkins CI/CD tools used for continuous integration and deployment.
2. Monitoring with Prometheus and Grafana is implemented for real-time performance tracking.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Setup Instructions\*\*\*\*\*\*\*\*\*\*\*\*

**Phase 1: Initial Setup and Deployment**

**Step 1: Launch 2 EC2 (Ubuntu 22.04):**

* Provision an EC2 instance on AWS with Ubuntu 22.04.
* Connect to the instance using SSH.
* One instance for Jenkins master, one for slave.

**Phase 2: CI/CD Setup**

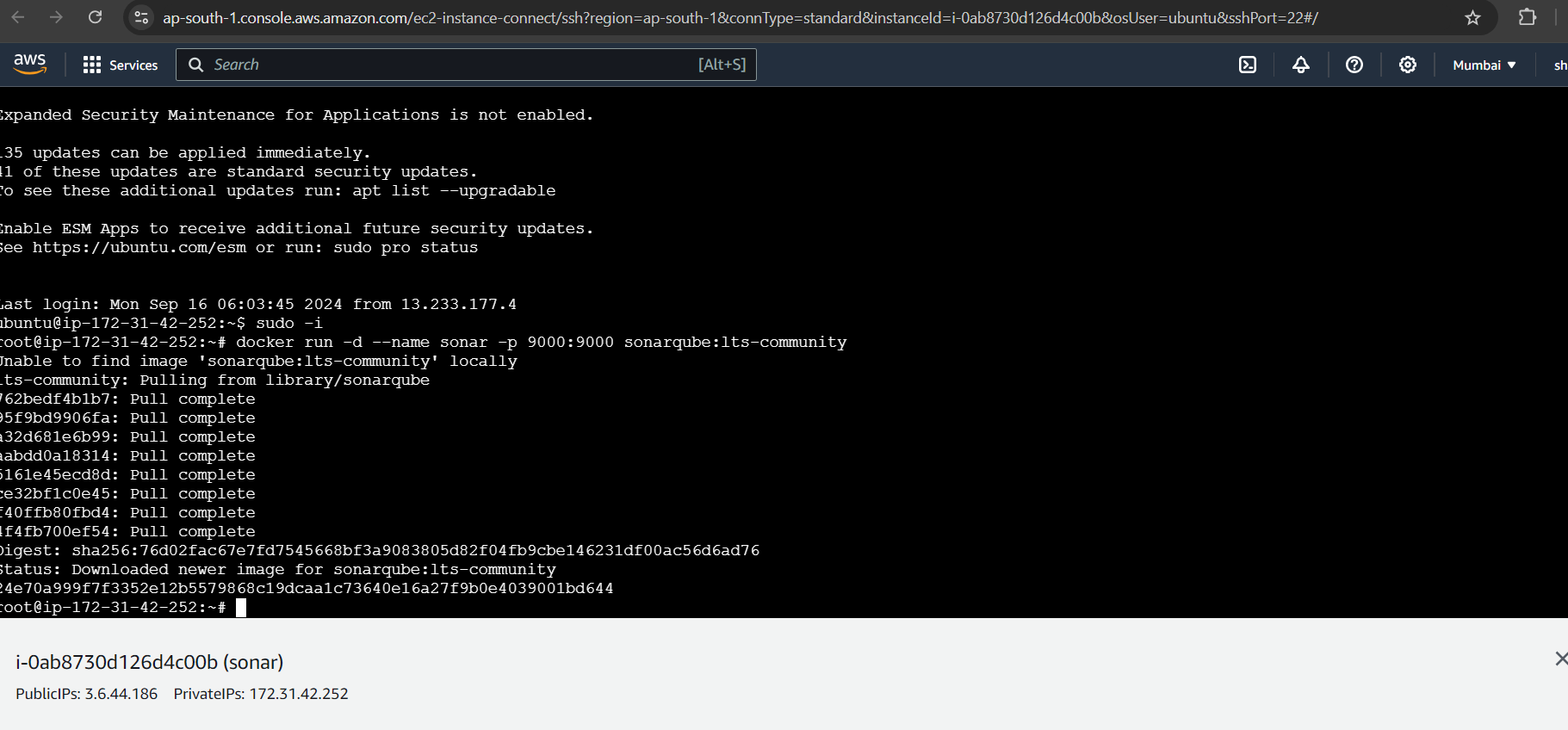
Install Jenkins, java, docker, docker compose on master node.

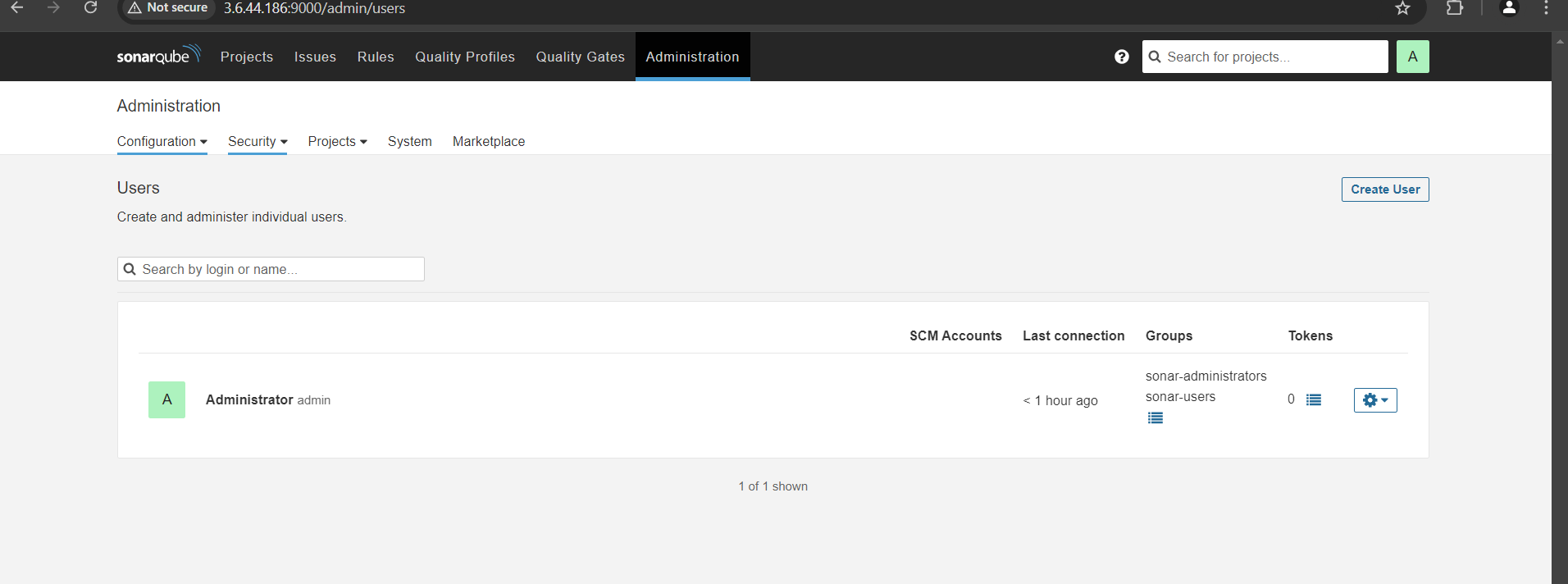
* On slave node there is no need to install Jenkins.
* Jenkins node = sign in to Jenkins. Install required plugins like SonarQube Scanner, Docker Commons, Docker Pipeline, Docker API, docker-build-step.
* **Integrate SonarQube and Configure(slave-node):**

--Install SonarQube run on port 9000 by default. Create project and generate token.

Integrate SonarQube with your CI/CD pipeline.

Configure SonarQube to analyze code for quality and security issues.





**Phase 3: Add slave node to master**

Jenkins dashboard -> manage Jenkins -> nodes -> add node -> using username and private key -> add credential of slave -> private key -> add

**Phase 4: Jenkins configuration**

-Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this

After adding sonar token. Click on Apply and Save.

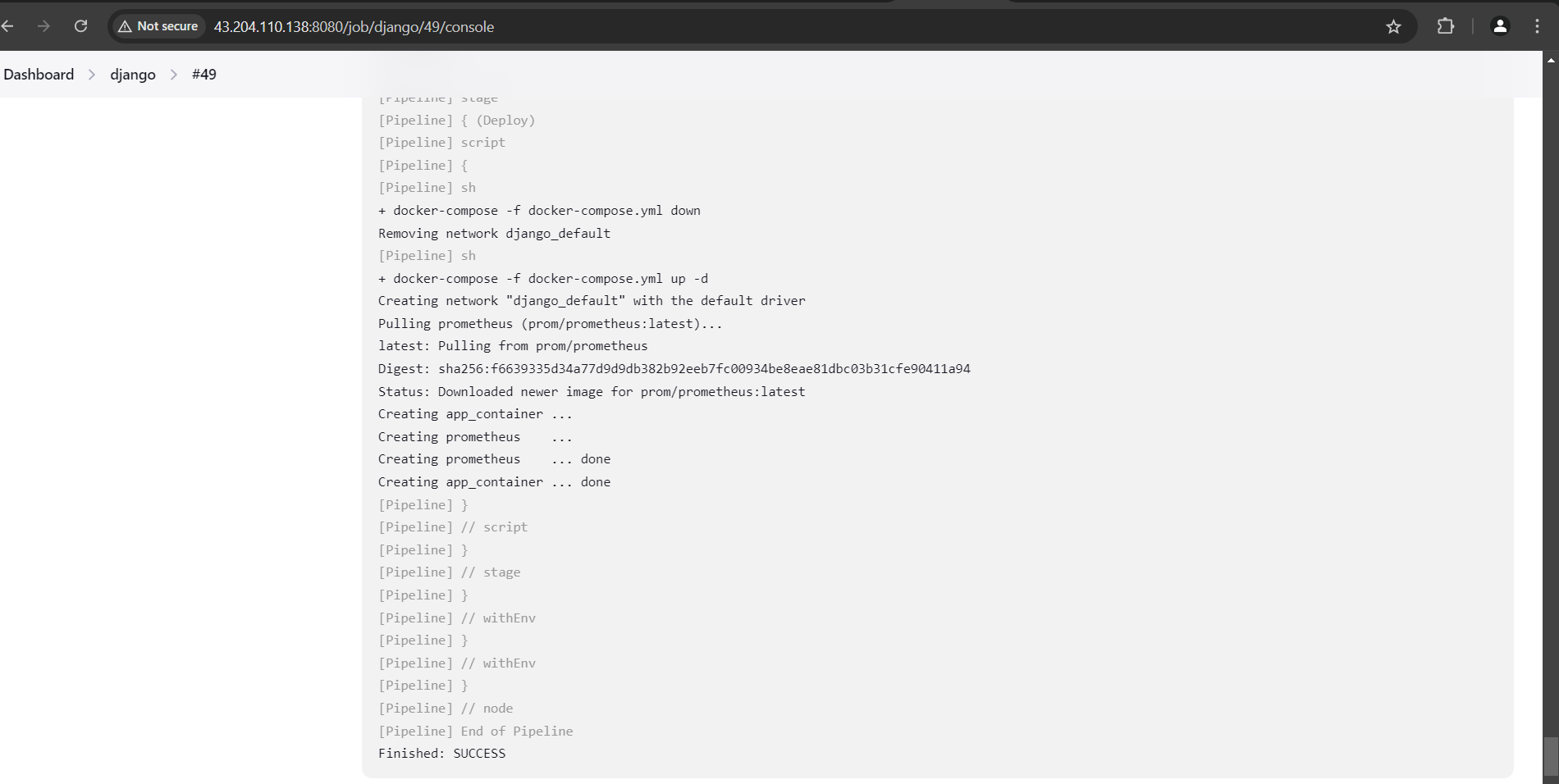
**-Global Tool Configuration** is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

**Add DockerHub Credentials:**

* To securely handle DockerHub credentials in your Jenkins pipeline, follow these steps:
  + Go to "Dashboard" → "Manage Jenkins" → "Manage Credentials."
  + Click on "System" and then "Global credentials (unrestricted)."
  + Click on "Add Credentials" on the left side.
  + Choose "Secret text" as the kind of credentials.
  + Enter your DockerHub credentials (Username and Password) and give the credentials an ID (e.g., "docker").
  + Click "OK" to save your DockerHub credentials.

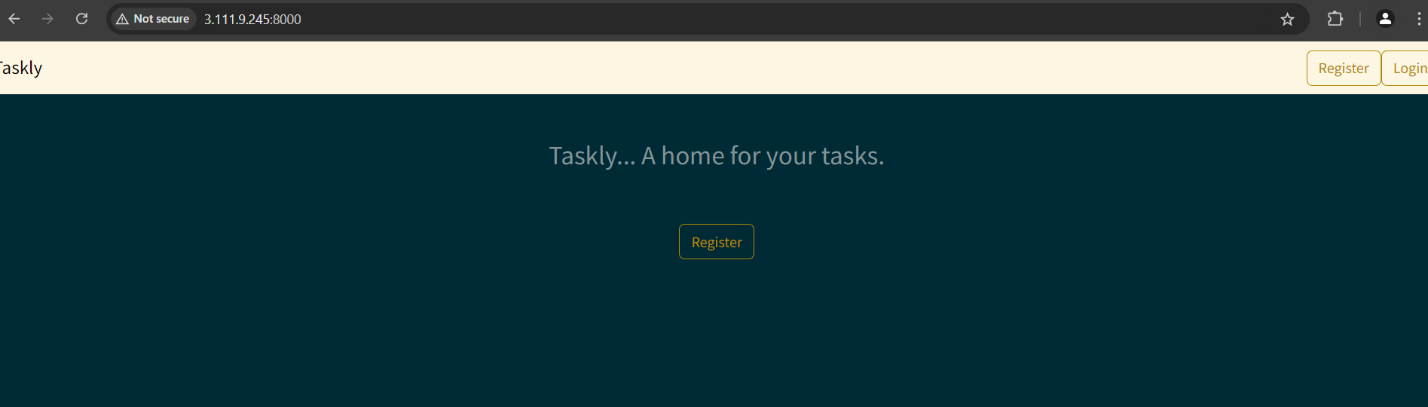
Run CI-CD pipeline.



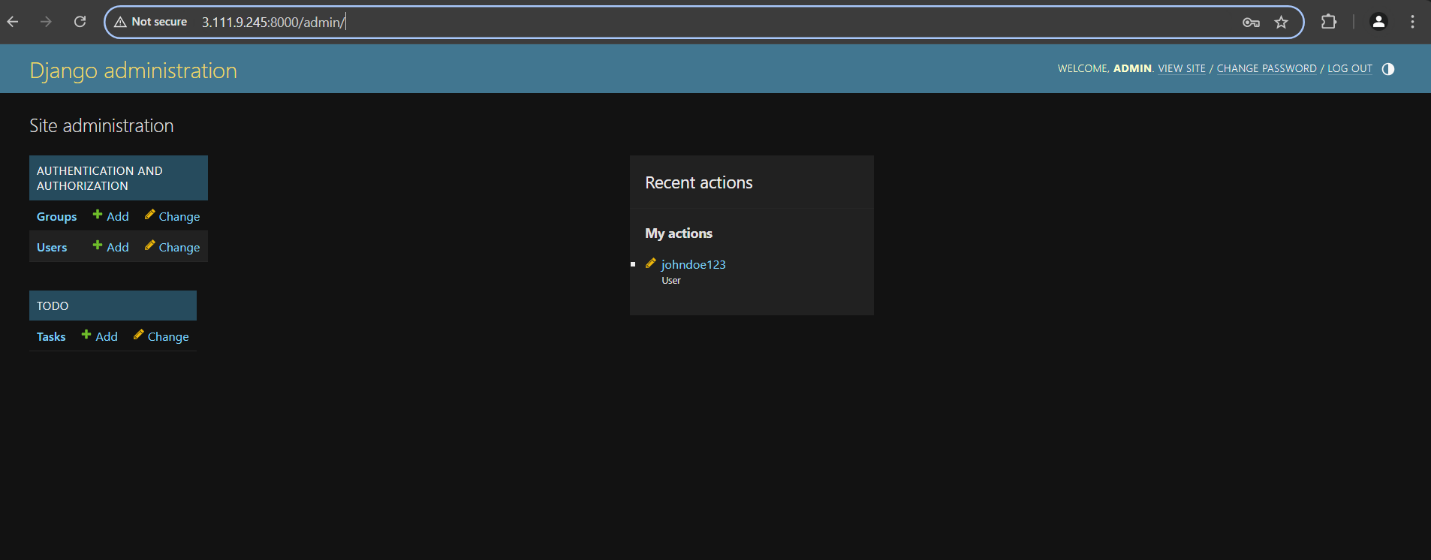
On slave node install Grafana for monitoring visualizations. I used Promethues in-built image in docker compose. So there is no need to install Prometheus. Grafana runs on port 3000 by default

Application deployed successfully

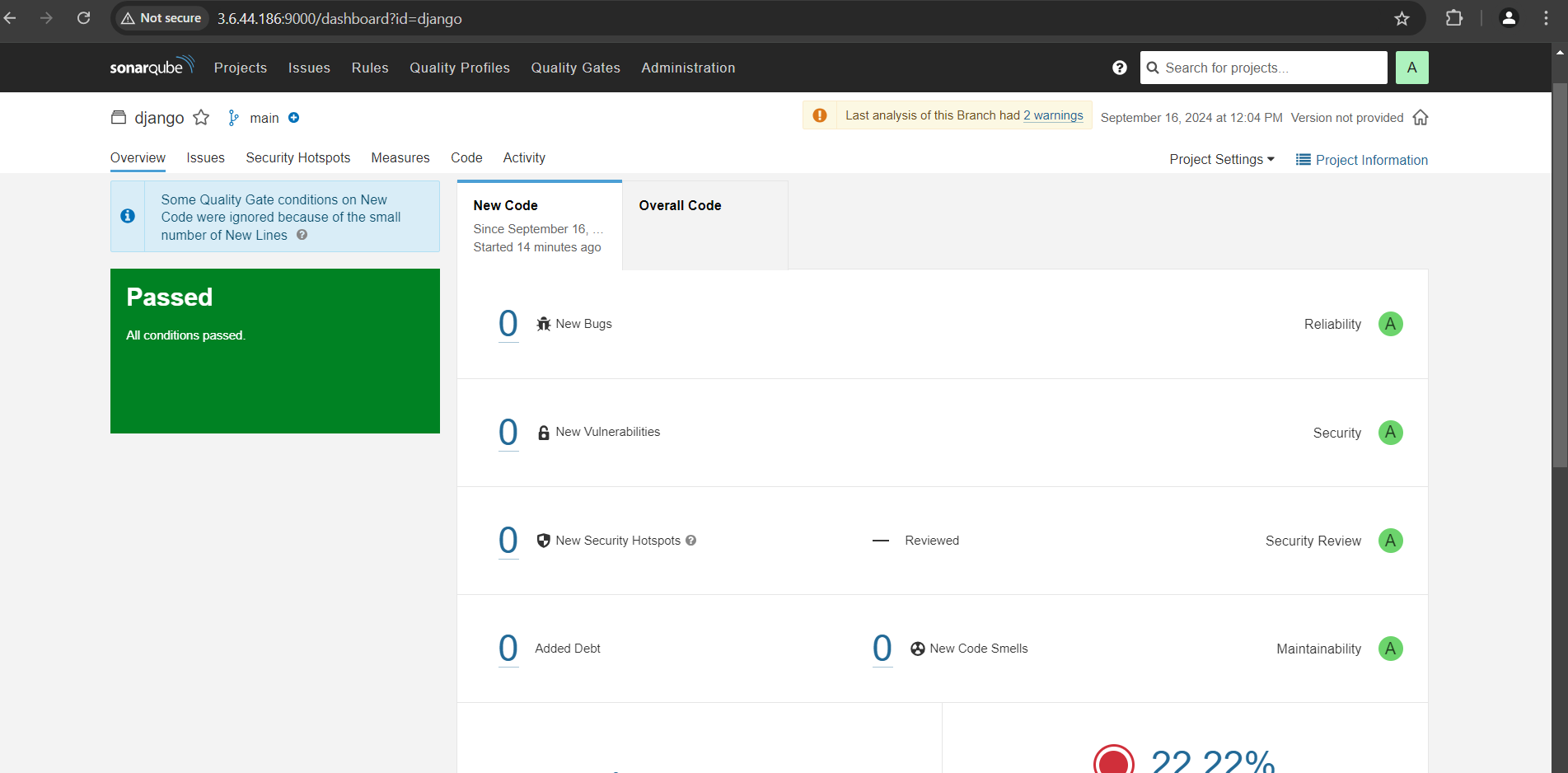
Application UI=



Admin Page=



Code-test passed



**Phase 4: Grafana setup**

Open a web browser and navigate to Grafana using your server's IP address. The default port for Grafana is 3000. For example:

http://<your-server-ip>:3000

You'll be prompted to log in to Grafana. The default username is "admin," and the default password is also "admin."

**Change the Default Password:**

When you log in for the first time, Grafana will prompt you to change the default password for security reasons. Follow the prompts to set a new password.

**Step 9: Add Prometheus Data Source:**

To visualize metrics, you need to add a data source. Follow these steps:

* Click on the gear icon (⚙️) in the left sidebar to open the "Configuration" menu.
* Select "Data Sources."
* Click on the "Add data source" button.
* Choose "Prometheus" as the data source type.
* In the "HTTP" section:
  + Set the "URL" to http://<ip address on which port Prometheus is running>:9090
  + Click the "Save & Test" button to ensure the data source is working.

**Step 10: Import a Dashboard:**

To make it easier to view metrics, you can import a pre-configured dashboard. Follow these steps:

* Click on the "+" (plus) icon in the left sidebar to open the "Create" menu.
* Select "Dashboard."
* Click on the "Import" dashboard option.
* Enter the dashboard code you want to import (e.g., code 7996).
* Click the "Load" button.
* Select the data source you added (Prometheus) from the dropdown.
* Click on the "Import" button. application matric on Grafana.

