**Movie App CICD**

**Current CI/CD Flow (implemented till monitoring)**

1. **Code Commit (Git → GitHub)**
   * Developer pushes code to GitHub.
   * This acts as the **source of truth** and triggers the pipeline.
2. **Pipeline Trigger (GitHub → Jenkins)**
   * A webhook or polling triggers Jenkins pipeline.
   * Jenkins acts as the CI/CD orchestrator.
3. **Code Quality Analysis (Jenkins → SonarQube → Quality Gate)**
   * Jenkins runs SonarQube scanner.
   * SonarQube checks code quality, security vulnerabilities, coverage, etc.
   * Quality Gate ensures build continues **only if code meets quality standards**.
4. **Dependency Installation (npm install)**
   * Jenkins installs project dependencies.
   * Ensures application can build successfully.
5. **Security Checks**
   * **OWASP Dependency Check** → scans project dependencies for known vulnerabilities.
   * **Trivy File Scan** → scans config files, IaC (Infrastructure as Code), and manifests.
6. **Build & Package (Docker)**
   * Jenkins builds a **Docker image** of the application.
   * Ensures consistency across environments.
7. **Image Security Scan (Trivy Image Scan)**
   * Scans the built Docker image for vulnerabilities.
   * If critical vulnerabilities are found, pipeline can stop here.
8. **Push to Docker Hub**
   * Safe image is pushed to Docker Hub (or any container registry).
   * Makes it available for deployment.
9. **Monitoring Setup (Prometheus + Grafana)**
   * Prometheus scrapes application + infrastructure metrics.
   * Grafana visualizes metrics and helps with alerting/dashboards.

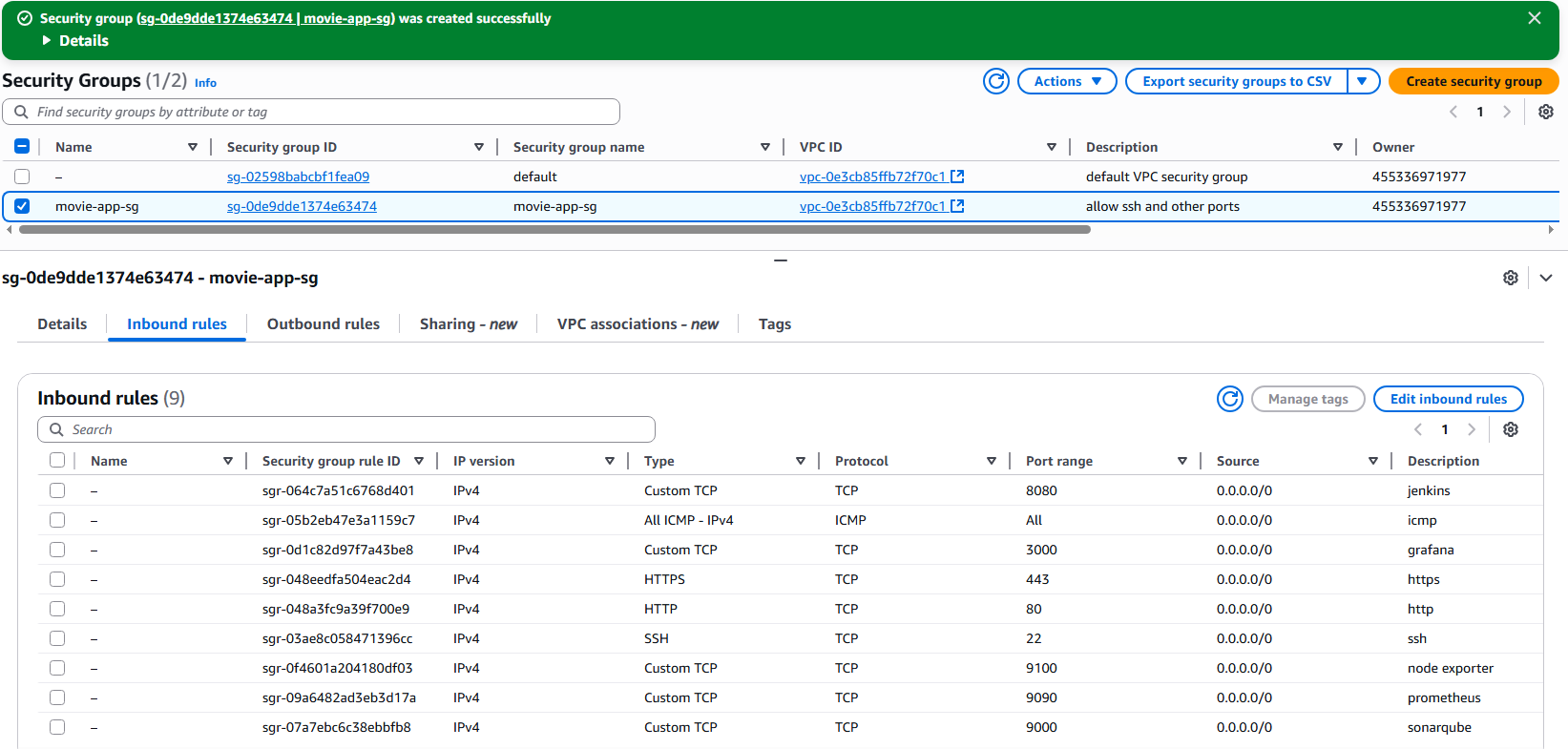
**Next Steps (Yet to be done)**

* **Helm** → Package your Kubernetes manifests as Helm charts for easy versioning & deployment.
* **Kubernetes Deployment** → Jenkins (or ArgoCD if going GitOps) deploys containerized app to a K8s cluster.
* **GitOps (ArgoCD/FluxCD)** → Automates syncing of Git state (Helm charts/manifests) with Kubernetes cluster.
* **Route 53 (DNS)** → Manage domain names, map to your Kubernetes ingress (Nginx/ALB), and expose services securely.

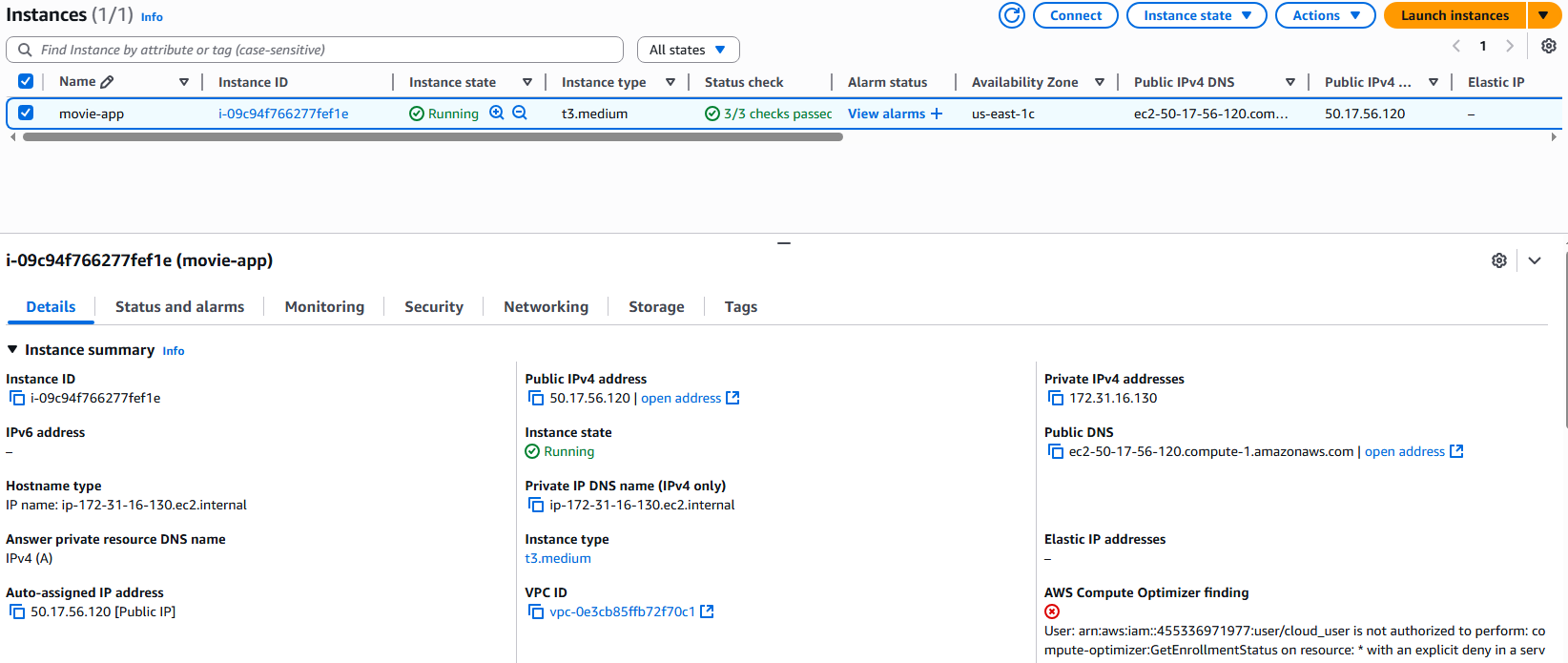
**Step1:- Create Security Groups on AWS**

**Ports to be enabled in Security Group**

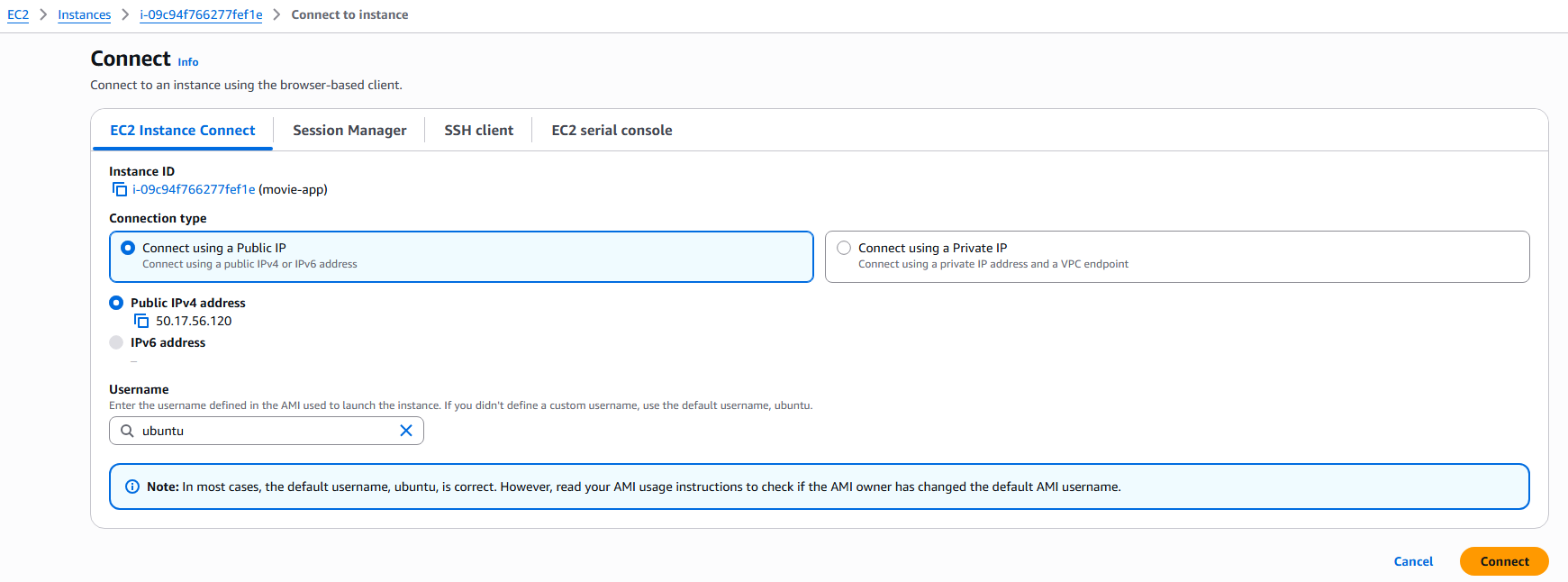
| **Service** | **Port** |
| --- | --- |
| HTTP | 80 |
| HTTPS | 443 |
| SSH | 22 |
| Jenkins | 8080 |
| SonarQube | 9000 |
| Prometheus | 9090 |
| Node Exporter | 9100 |
| Grafana | 3000 |

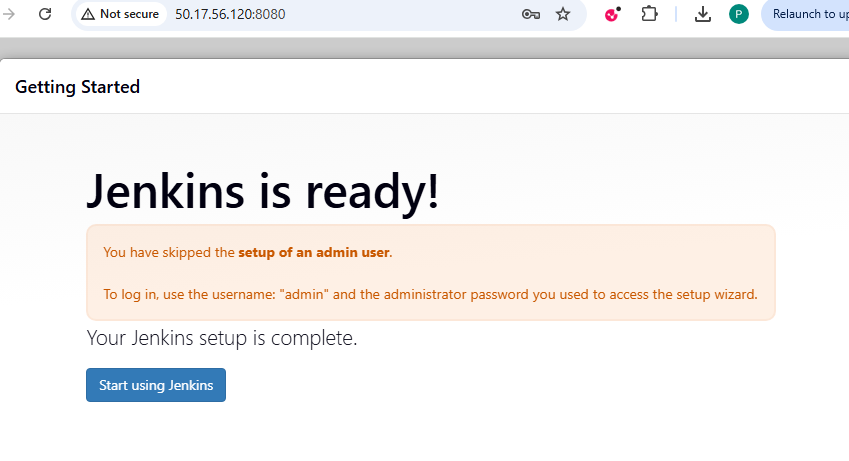


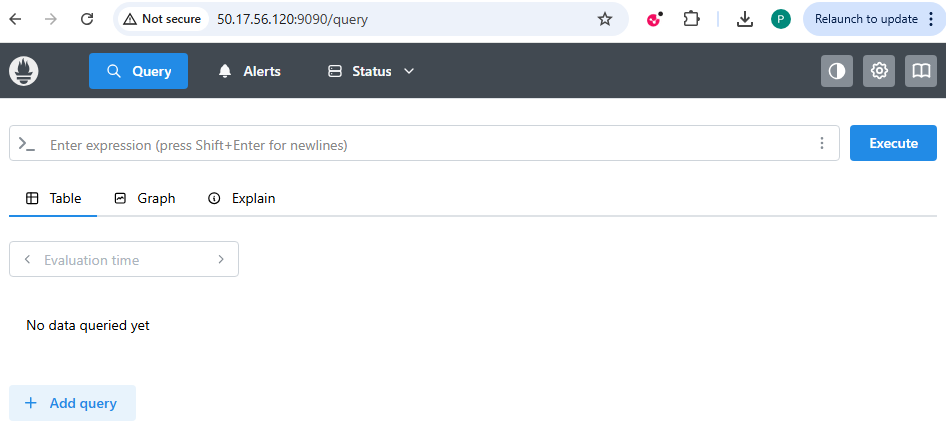
**Step2:- Create EC2 instance using t3.medium and storage with min 16 GiB.**

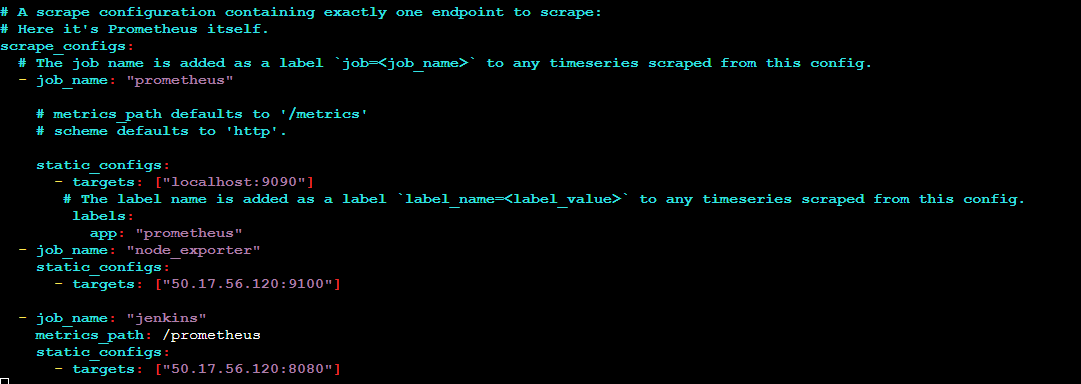
****

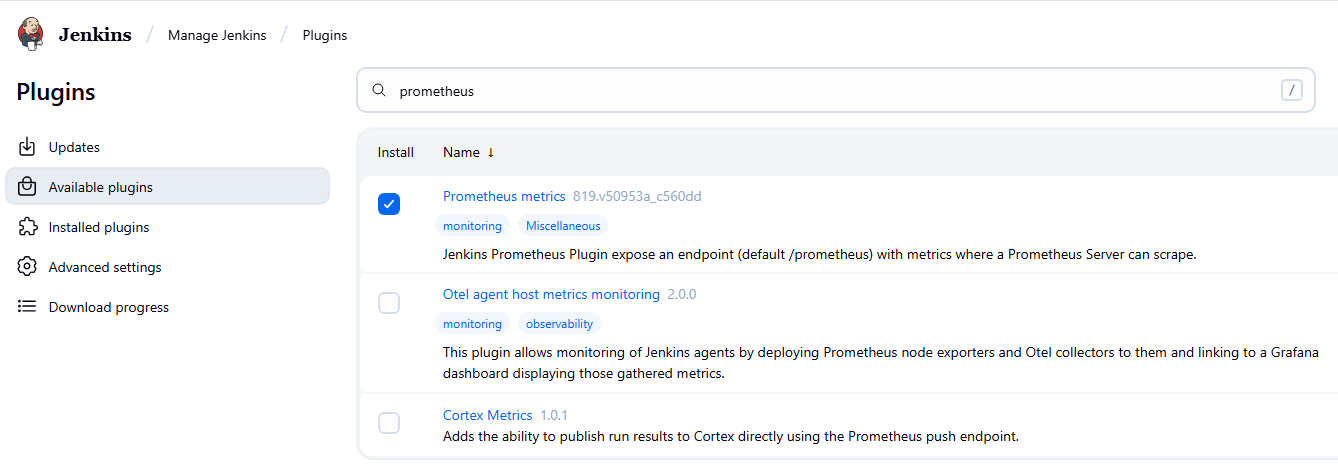
**Step3:- Connect using EC2 Instance Connect**

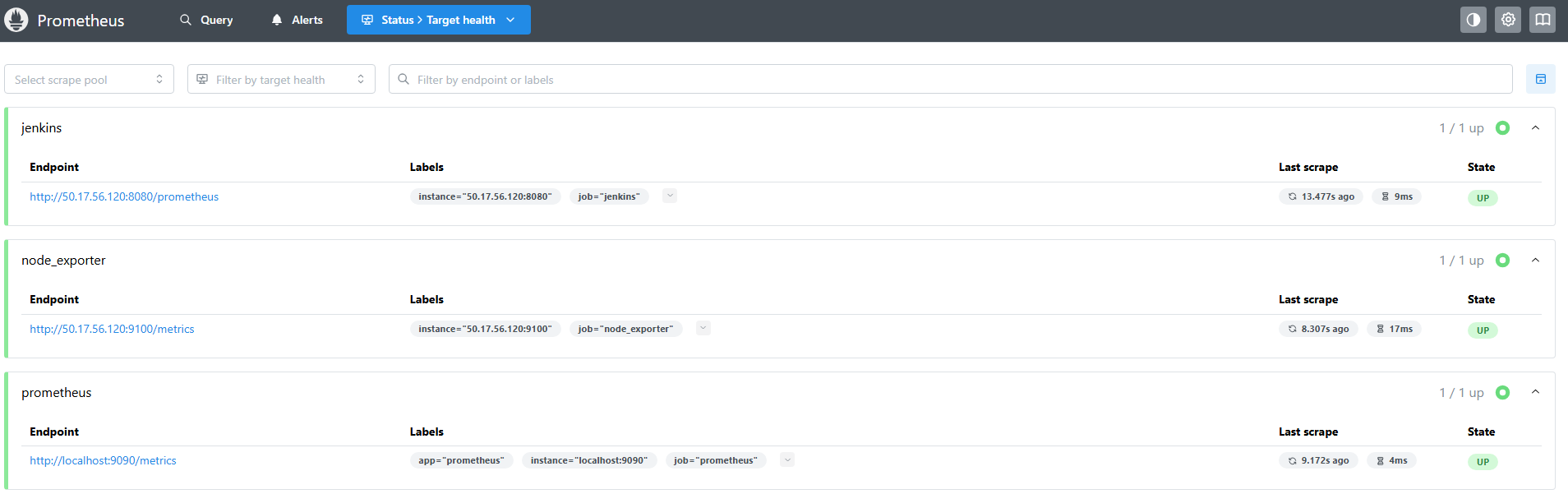
****

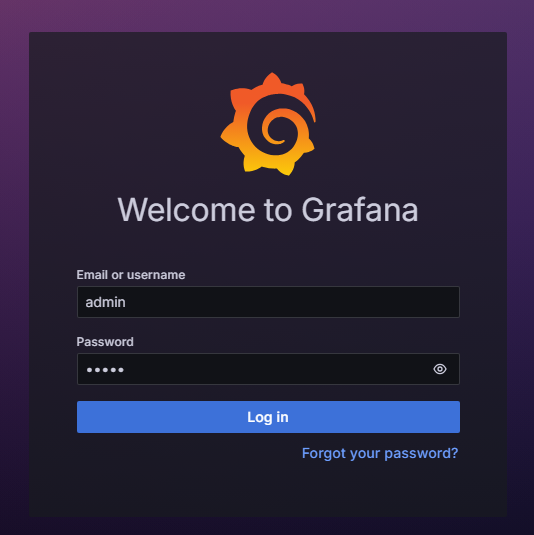
****

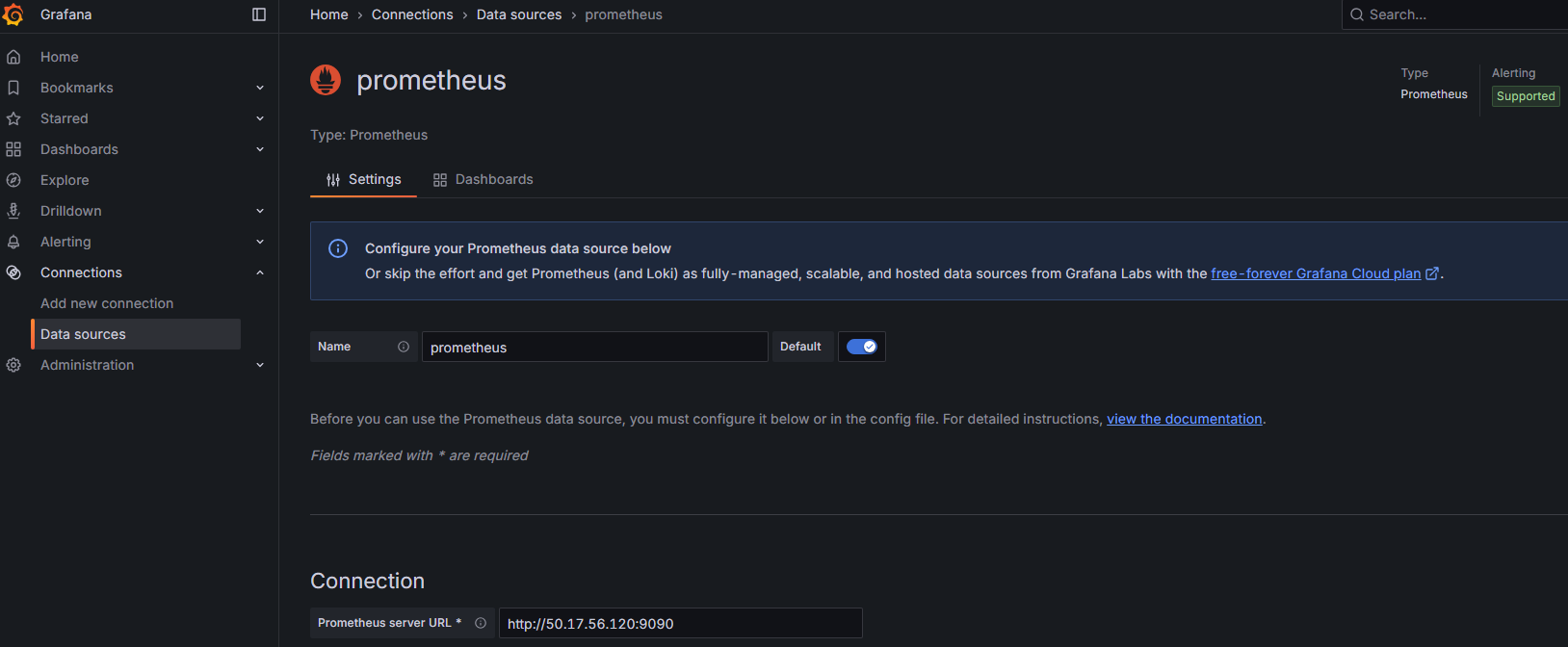
****

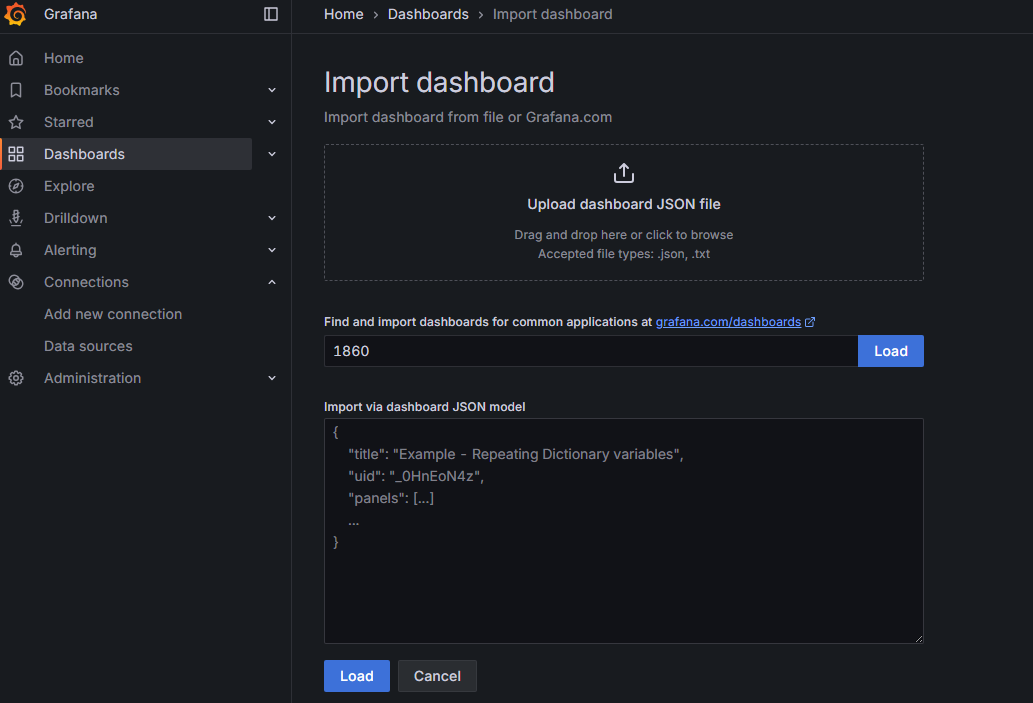
****

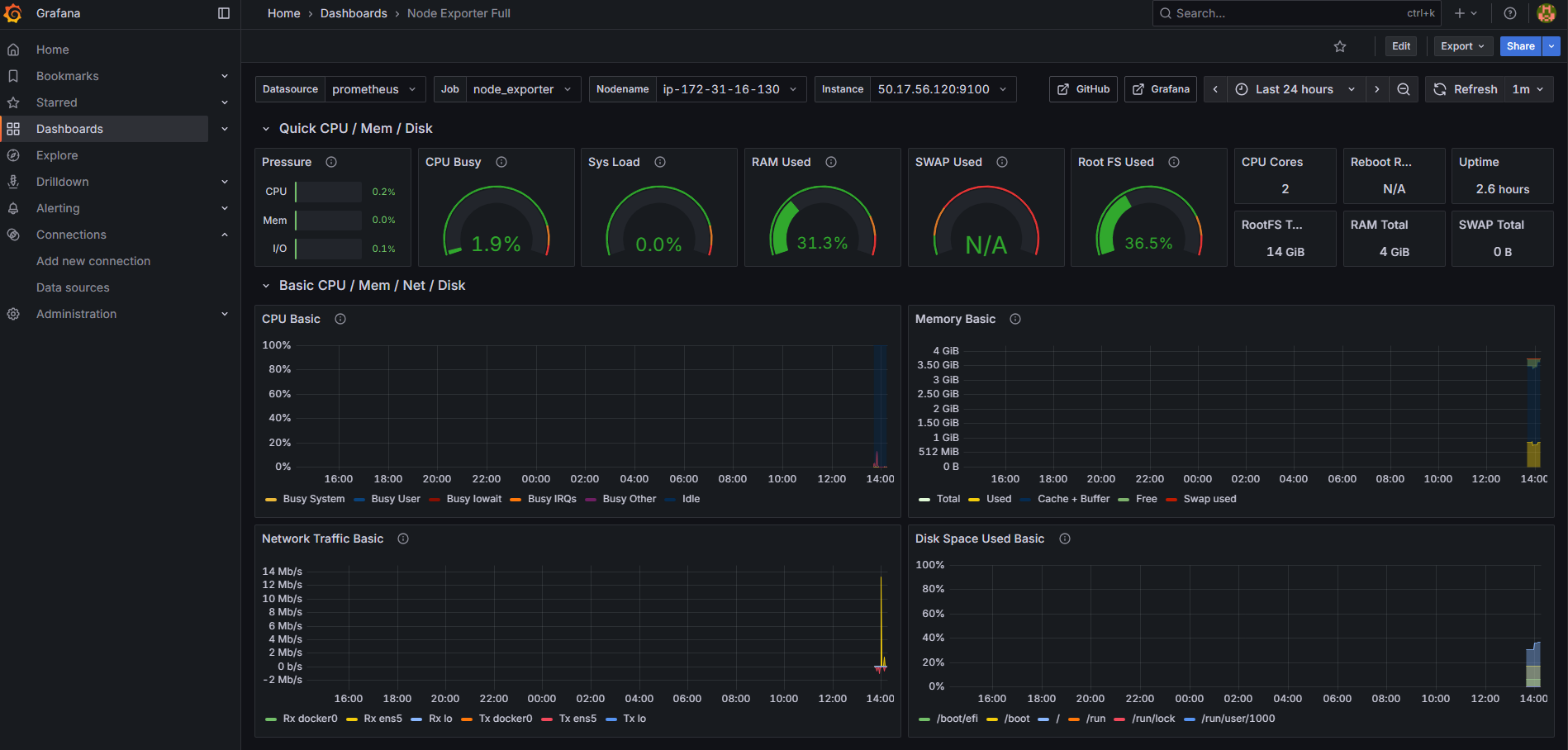
****

****

****

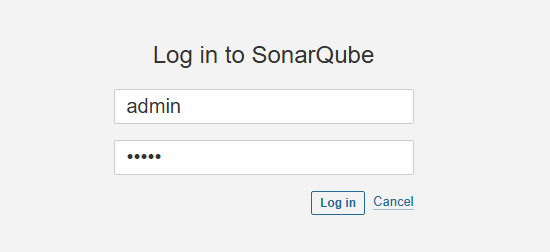
****

****

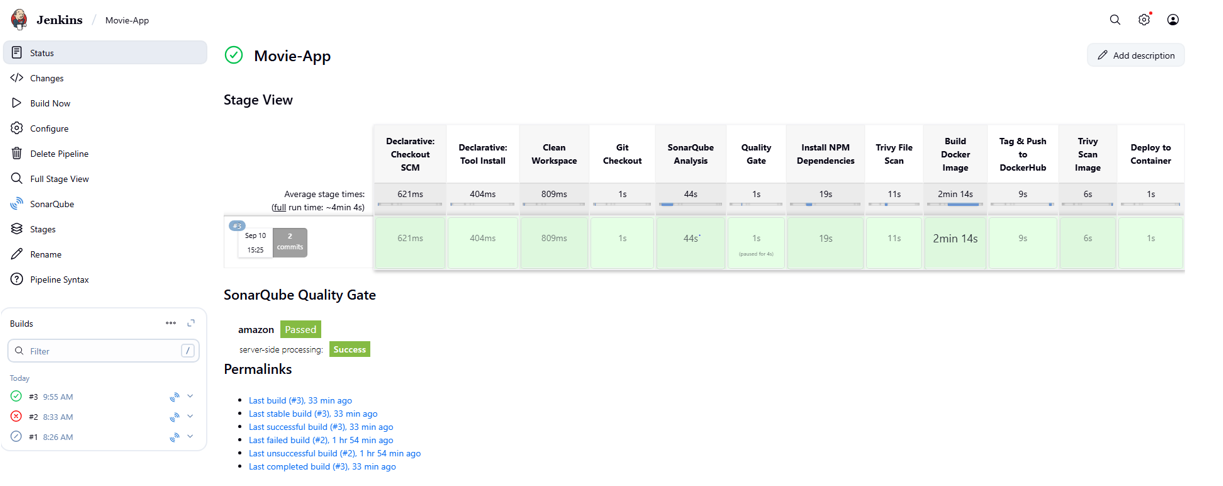
****

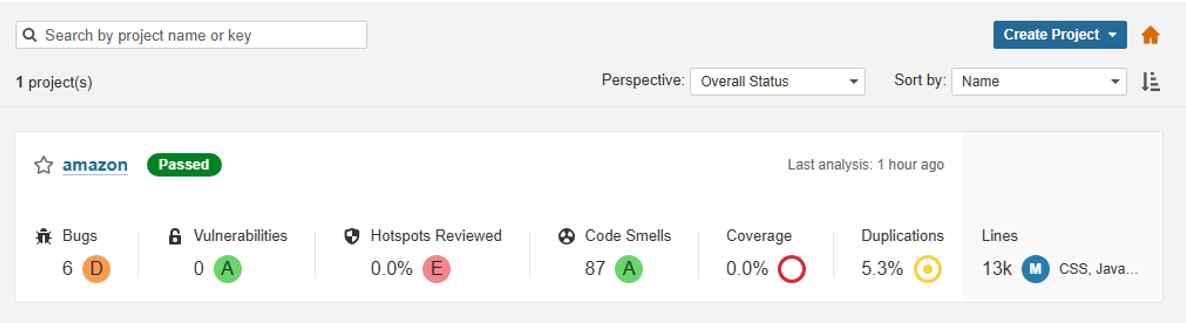
**Jenkins Plugins to Install**

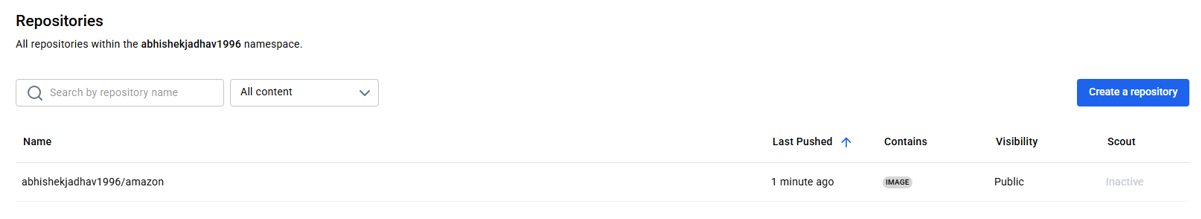
* Eclipse Temurin installer Plugin
* NodeJS
* Email Extension Plugin
* OWASP Dependency-Check Plugin
* Pipeline: Stage View Plugin
* SonarQube Scanner for Jenkins
* Prometheus metrics plugin
* Docker API Plugin
* Docker Commons Plugin
* Docker Pipeline
* Docker plugin
* docker-build-step

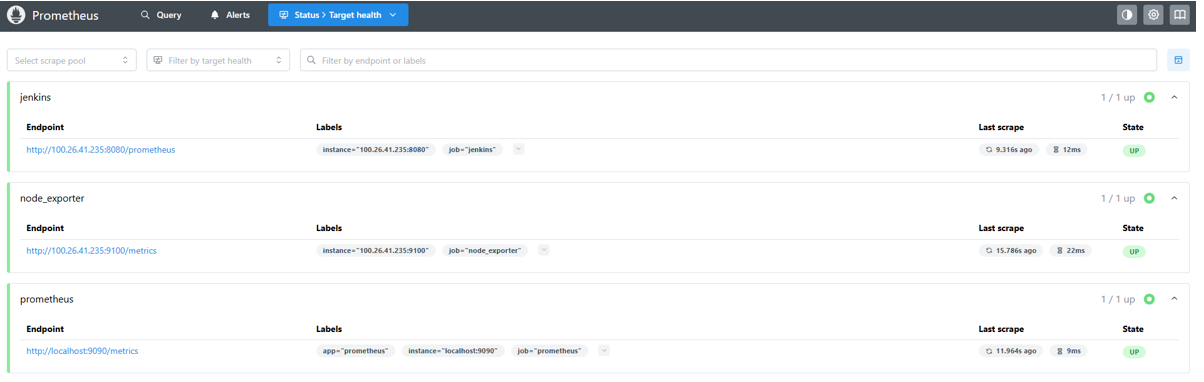


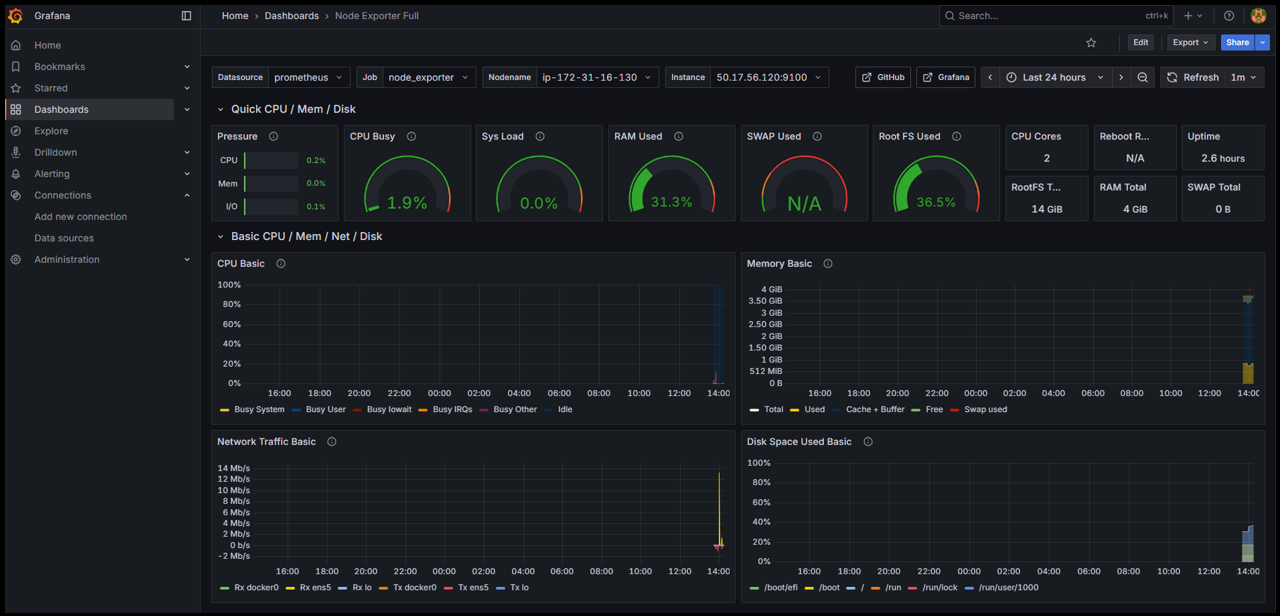
**Outputs**

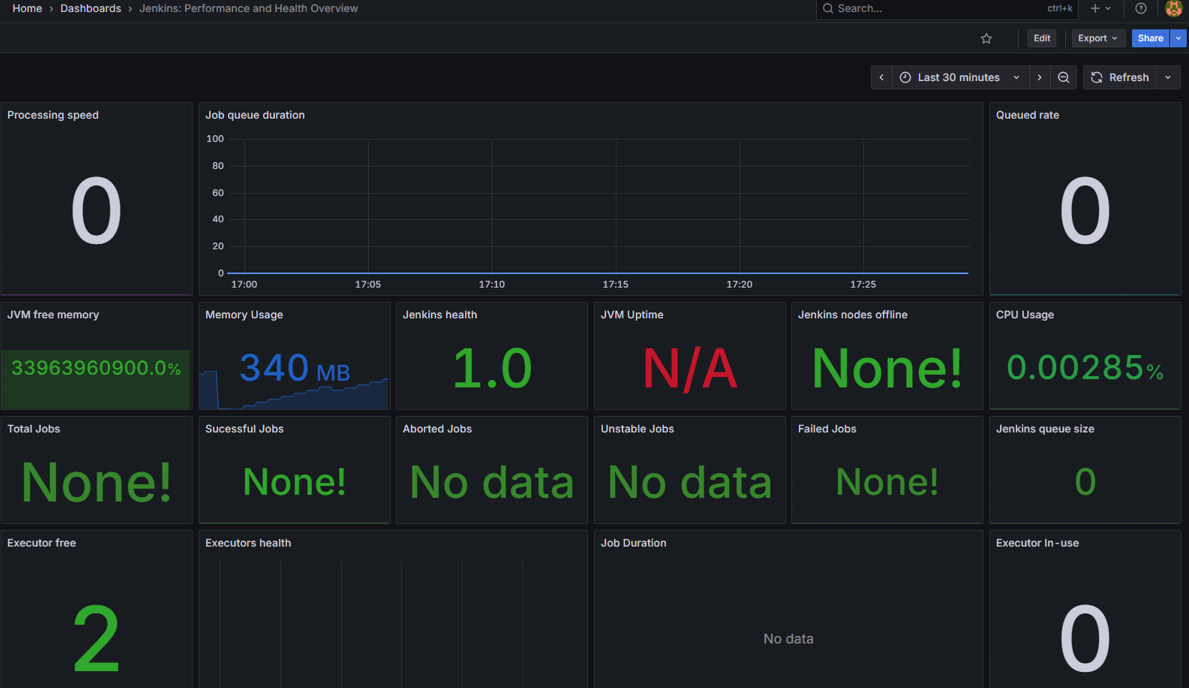
****

****

****

****

****

****