IGP Project Report - Sameer Sabale

Project Overview:

In this project, I have set up a DevOps pipeline using multiple tools and technologies to automate the deployment and monitoring of applications. The infrastructure involves 4 servers:

- It is a using for Jump server For Docker, Jenkins, Git, and Maven.
- **Master and Worker Servers (3 servers):** For setting up a Kubernetes cluster.

The goal was to integrate version control, continuous integration, continuous deployment, and monitoring tools to provide a complete DevOps pipeline.

Task 1: Git Setup and Code Version Control

1. **Install Git:**
   - Installed **Git** on the jump server to manage version control for the source code of **ABC Technologies**.
2. **Download the Source Code:**
   - The project source code was downloaded from the relevant repository to initialize the Git environment.

```
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /source code/ABC Technologies/.git/
```

   -
3. **Initialize Git:**
   - Initialized Git in the project directory using the **git init** command.
4. **Move Code to Staging Area:**
   - The necessary changes were staged using **git add .** to move the code into the staging area.
5. **Check Git Status:**
   - To verify the changes, ran the **git status** command to check for any uncommitted changes.

```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .classpath
        new file:   .project
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   .settings/org.eclipse.m2e.core.prefs
        new file:   README.md
        new file:   pom.xml
        new file:   pom.xml.bak
        new file:   src/main/java/com/abc/RetailModule.java
        new file:   src/main/java/com/abc/dataAccessObject/RetailAccessObject.java
        new file:   src/main/java/com/abc/dataAccessObject/RetailDataImp.java
        new file:   src/main/webapp/WEB-INF/web.xml
        new file:   src/main/webapp/index.jsp
        new file:   src/test/java/com/abc/dataAccessObject/ProductImpTest.java
```

6. **Commit and Push Code:**
    - After committing the changes using **git commit -m "Initial commit"**, pushed the code to the **master branch** on GitHub.
    - The repository URL is: https://github.com/SameerSabale/IGP-Project-Submission-Name-Sameer-Sabale

---

Task 2: Jenkins Pipeline Setup

1. **Install Jenkins:**
    - Installed Jenkins on the jump server to manage continuous integration.

```
Last metadata expiration check: 0:08:37 ago on Wed Nov 13 06:16:39 2024.
Dependencies resolved.
================================================================================
 Package                  Architecture  Version                    Repository  Size
================================================================================
Installing:
 java-17-amazon-corretto  x86_64        1:17.0.13+11-1.amzn2023.1  amazonlinux 213 k
Installing dependencies:
 alsa-lib                 x86_64        1.2.7.2-1.amzn2023.0.2     amazonlinux 504 k
 cairo                    x86_64        1.17.6-2.amzn2023.0.1      amazonlinux 684 k
 dejavu-sans-fonts        noarch        2.37-16.amzn2023.0.2       amazonlinux 1.3 M
 dejavu-sans-mono-fonts   noarch        2.37-16.amzn2023.0.2       amazonlinux 467 k
 dejavu-serif-fonts       noarch        2.37-16.amzn2023.0.2       amazonlinux 1.0 M
 fontconfig               x86_64        2.13.94-2.amzn2023.0.2     amazonlinux 273 k
 fonts-filesystem         noarch        1:2.0.5-12.amzn2023.0.2    amazonlinux 9.5 k
 freetype                 x86_64        2.13.2-5.amzn2023.0.1      amazonlinux 423 k
 giflib                   x86_64        5.2.1-9.amzn2023.0.1       amazonlinux  49 k
 google-noto-fonts-common noarch        20201206-2.amzn2023.0.2    amazonlinux  15 k
 google-noto-sans-vf-fonts noarch       20201206-2.amzn2023.0.2    amazonlinux 492 k
 graphite2                x86_64        1.3.14-7.amzn2023.0.2      amazonlinux  97 k
 harfbuzz                 x86_64        7.0.0-2.amzn2023.0.1       amazonlinux 868 k
 java-17-amazon-corretto-headless x86_64 1:17.0.13+11-1.amzn2023.1 amazonlinux  91 M
 javapackages-filesystem  noarch        6.0.0-7.amzn2023.0.6       amazonlinux  12 k
 langpacks-core-font-en   noarch        3.0-21.amzn2023.0.4        amazonlinux  10 k
 libICE                   x86_64        1.1.1-3.amzn2023.0.1       amazonlinux  76 k
 libSM                    x86_64        1.2.4-3.amzn2023.0.1       amazonlinux  45 k
 libX11                   x86_64        1.8.10-2.amzn2023.0.1      amazonlinux 659 k
 libX11-common            noarch        1.8.10-2.amzn2023.0.1      amazonlinux 147 k
 libXau                   x86_64        1.0.11-6.amzn2023.0.1      amazonlinux  33 k
```

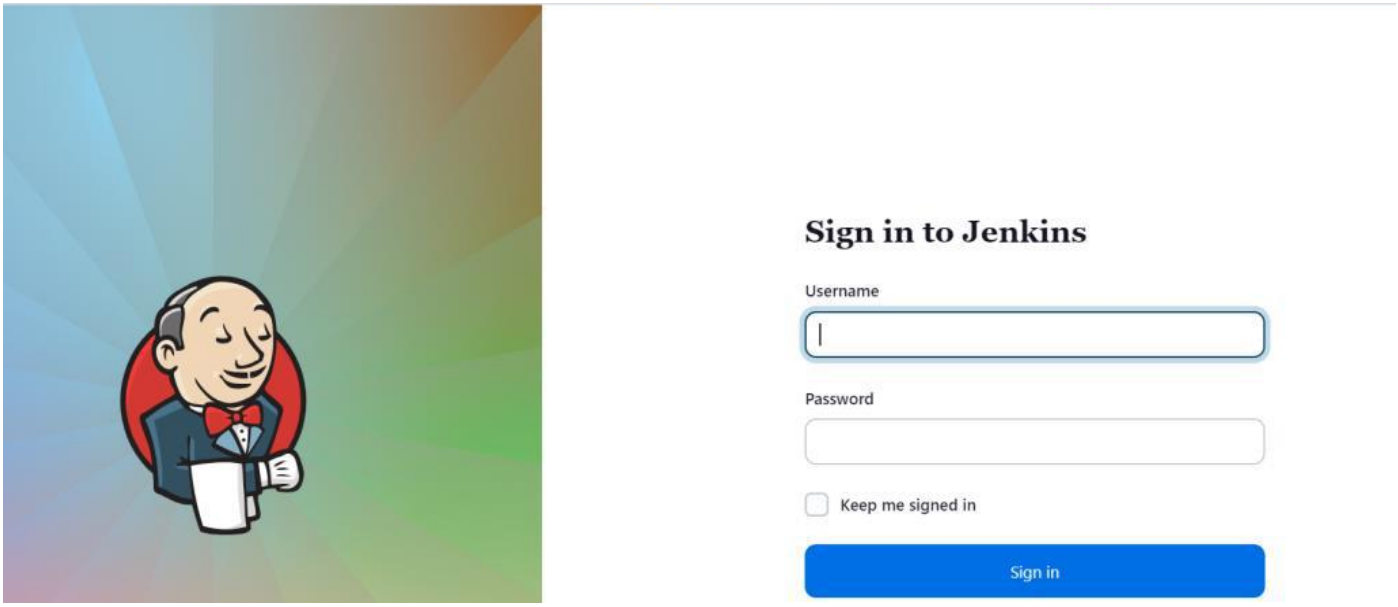
2. **Install Maven:**

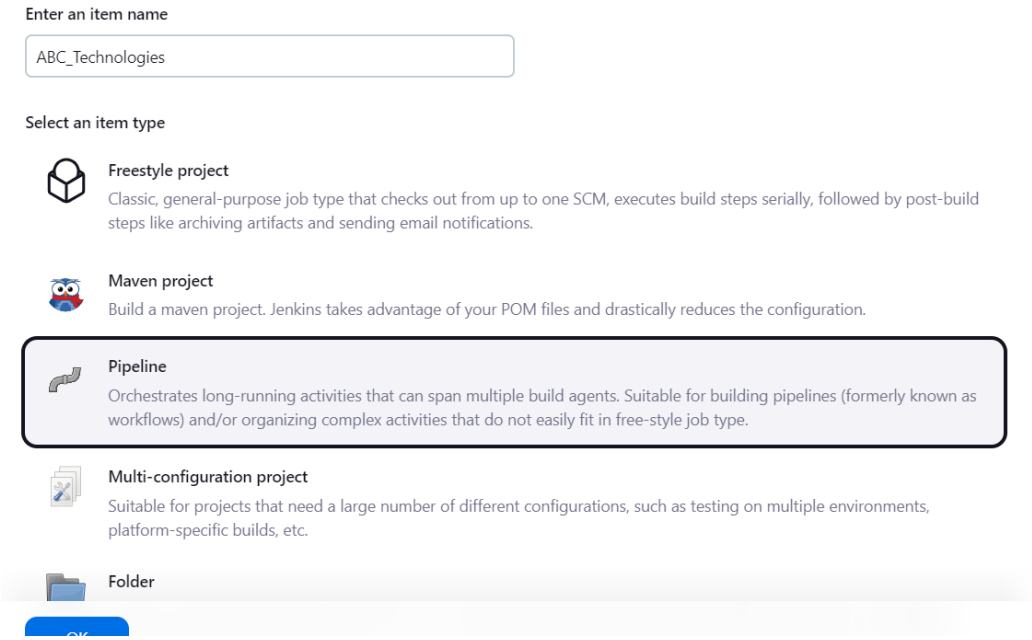
- **Maven** was installed to manage dependencies and build the application.

```
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: /opt/maven
Java version: 11.0.24, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.24.0.8-3.el8.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.18.0-553.16.1.el8_10.x86_64", arch: "amd64", family: "unix"
```

3. **Configure Jenkins:**
   - After logging into Jenkins, created a pipeline folder to organize the project.

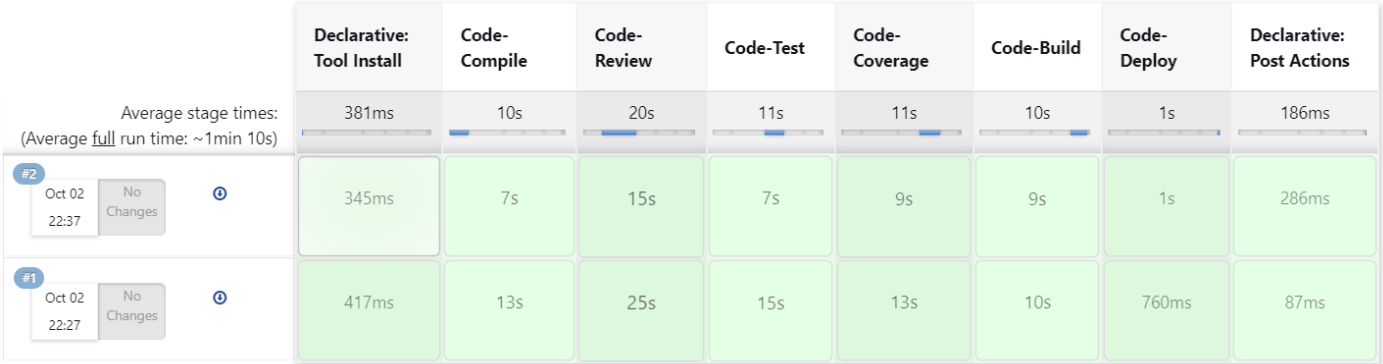
```
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-11-13 06:30:47 UTC; 8s ago
   Main PID: 26743 (java)
      Tasks: 50 (limit: 4659)
     Memory: 552.4M
        CPU: 17.027s
     CGroup: /system.slice/jenkins.service
             └─26743 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort>

Nov 13 06:30:44 ip-172-31-84-18.ec2.internal jenkins[26743]: 6008e5ded3344c788543cb052b0b2ed2
Nov 13 06:30:44 ip-172-31-84-18.ec2.internal jenkins[26743]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Nov 13 06:30:44 ip-172-31-84-18.ec2.internal jenkins[26743]: ************************************************************
Nov 13 06:30:44 ip-172-31-84-18.ec2.internal jenkins[26743]: ************************************************************
Nov 13 06:30:44 ip-172-31-84-18.ec2.internal jenkins[26743]: ************************************************************
Nov 13 06:30:47 ip-172-31-84-18.ec2.internal jenkins[26743]: 2024-11-13 06:30:47.662+0000 [id=32]       INFO       jenkins.InitReactor>
Nov 13 06:30:47 ip-172-31-84-18.ec2.internal jenkins[26743]: 2024-11-13 06:30:47.703+0000 [id=23]       INFO       hudson.lifecycle.Li>
Nov 13 06:30:47 ip-172-31-84-18.ec2.internal systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Nov 13 06:30:48 ip-172-31-84-18.ec2.internal jenkins[26743]: 2024-11-13 06:30:48.014+0000 [id=48]       INFO       h.m.DownloadService>
Nov 13 06:30:48 ip-172-31-84-18.ec2.internal jenkins[26743]: 2024-11-13 06:30:48.015+0000 [id=48]       INFO       hudson.util.Retrier>
```





4. **Pipeline Script:**
    - The pipeline script, which automates the build and deployment process, is available in the GitHub repository.
    - This script runs the pipeline and deploys the application on a **Tomcat Web Server**.

5. **Pipeline Execution:**
    - Ran the pipeline to test its functionality. The pipeline worked successfully, and the output was verified through Jenkins logs.



Task 3: Docker Image Build and Deployment

1. **Install Docker:**
    - Installed Docker on the jump server for containerizing the application.

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2024-10-02 13:40:16 EDT; 20s ago
     Docs: https://docs.docker.com
 Main PID: 33547 (dockerd)
    Tasks: 7
   Memory: 33.4M
   CGroup: /system.slice/docker.service
           └─33547 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Oct 02 13:40:15 jamp systemd[1]: Starting Docker Application Container Engine...
Oct 02 13:40:15 jamp dockerd[33547]: time="2024-10-02T13:40:15.338889410-04:00" level=info msg="Starting up"
Oct 02 13:40:15 jamp dockerd[33547]: time="2024-10-02T13:40:15.402625192-04:00" level=info msg="Loading containers: start."
Oct 02 13:40:16 jamp dockerd[33547]: time="2024-10-02T13:40:16.187713313-04:00" level=info msg="Loading containers: done."
```

```
Docker version 26.1.3, build b72abbb
```

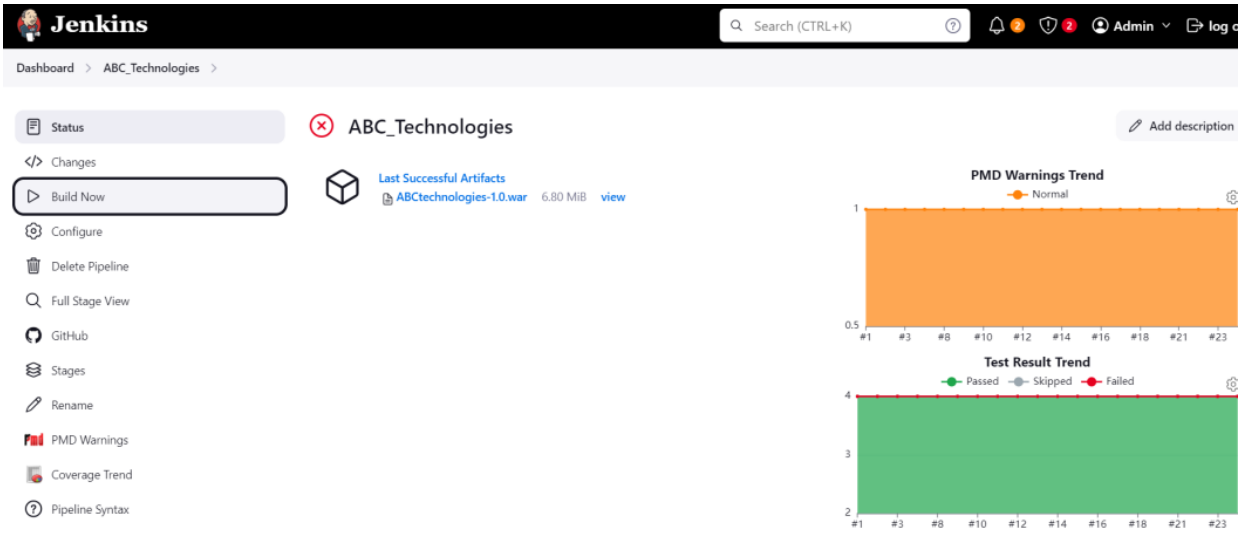
2. **Login to Docker:**
   - Logged into Docker with the necessary credentials using the **docker login** command.
3. **Build Docker Image:**
   - Created a **Docker image** from the application's source code using **docker build -t <image_name> ..**

```
[+] Building 2.5s (8/8) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                        0.0s
 => => transferring dockerfile: 393B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/tomcat:9                                                                 2.4s
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [internal] load build context                                                                                           0.0s
 => => transferring context: 47B                                                                                            0.0s
 => [1/3] FROM docker.io/library/tomcat:9@sha256:67f6b5f0d4061aea0cbd1e005d5987b13ccab5505aeecc75383203cf0650dfe3           0.0s
 => CACHED [2/3] COPY ABCtechnologies-1.0.war /usr/local/tomcat/webapps/                                                    0.0s
 => CACHED [3/3] WORKDIR /usr/local/tomcat/bin                                                                              0.0s
 => exporting to image                                                                                                      0.0s
 => => exporting layers                                                                                                     0.0s
 => => writing image sha256:2f0b7c70e483c6644bb72dd3701591385c31c8d6c8f6b882839deb38a502a43c                                0.0s
 => => naming to docker.io/library/abctechnologies                                                                          0.0s
```

4. **Deploy to Docker Container:**
   - Deployed the Docker image to a container for testing, and the application was accessed via the container's IP address with the bound port using the **docker run -d -p <port>:<container_port> <image_name>** command.

```
CONTAINER ID   IMAGE     COMMAND             CREATED          STATUS          PORTS                                       NAMES
7c79f84d9478   app       "catalina.sh run"   5 minutes ago    Up 5 minutes    0.0.0.0:8082→8080/tcp, :::8082→8080/tcp     my-tomcat-container
```

5. **Push Docker Image to Docker Hub:**
   - After confirming the container was working correctly, I pushed the Docker image to **Docker Hub** using **docker push <dockerhub_username>/<image_name>.**



Stage View

	Declarative: Tool Install	Code-Compile	Code-Review	Code-Test	Code-Coverage	Code-Build	Build Docker Image	Run Docker Container	Push to Docker Hub	Declarative: Post Actions
Average stage times: (Average full run time: ~2min 47s)	577ms	19s	38s	25s	31s	31s	4s	1s	6s	203ms
#26 Oct 03 20:40 No Changes	598ms	16s	37s	24s	30s	32s	4s	3s	13s	276ms

---

Task 4: Kubernetes Cluster Deployment

1. **Set Up Kubernetes Cluster:**
   -
   - Set up a **Kubernetes cluster** using 1 **master** and 2 **worker** nodes for the deployment of ABC Technologies' application.

```
NAME     STATUS   ROLES           AGE   VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE                        KERNEL-VERSION                 CONTAINER-RU
NTIME
master   Ready    control-plane   56d   v1.29.7   192.168.242.112   <none>        Rocky Linux 8.10 (Green Obsidian)   4.18.0-553.el8_10.x86_64   containerd:/
/1.6.32
worker1  Ready    <none>          56d   v1.29.7   192.168.242.113   <none>        Rocky Linux 8.10 (Green Obsidian)   4.18.0-553.el8_10.x86_64   containerd:/
/1.6.32
worker2  Ready    <none>          56d   v1.29.7   192.168.242.114   <none>        Rocky Linux 8.10 (Green Obsidian)   4.18.0-553.el8_10.x86_64   containerd:/
/1.6.32
```

2. **Deployment and Service Files:**
   - Wrote the necessary **deployment** and **service** files for Kubernetes, creating the **pods** using the **kubectl apply -f <file_name>** command.

```
deployment.apps/abctechnologies created
[vivek@master ~]$
[vivek@master ~]$
[vivek@master ~]$ kubectl apply -f service.yml
service/service-abctechnologies created
```

```
NAME                                  READY   STATUS    RESTARTS   AGE    IP              NODE      NOMINATED NODE   READINESS GATES
abctechnologies-5f747f7bfb-gqrtw      1/1     Running   0          9m5s   10.10.189.77    worker2   <none>           <none>
abctechnologies-5f747f7bfb-qw85p      1/1     Running   0          9m6s   10.10.235.141   worker1   <none>           <none>
abctechnologies-5f747f7bfb-v9sk5      1/1     Running   0          9m5s   10.10.189.78    worker2   <none>           <none>
```

```
NAME                TYPE                             DATA   AGE
dockerhub-secret    kubernetes.io/dockerconfigjson   1      14m
```

3. **Create Docker Secret for Authentication:**
    - Created a secret file to securely pull Docker images from **Docker Hub** into the Kubernetes cluster using the **kubectl create secret docker-registry <secret_name> --docker-server=<dockerhub_url> --docker-username=<username> --docker-password=<password> --docker-email=<email>** command.
4. **Pod Verification:**
    - The pods were successfully created, and I verified the deployment using Kubernetes CLI commands like **kubectl get pods**.
5. **NodePort for Access:**
    - The application was made accessible through a **NodePort**, providing external access to the services running inside the Kubernetes pods using **kubectl expose pod <pod_name> --type=NodePort --name=<service_name>**.

Task 5: Kubernetes Monitoring Setup

1. **Install Monitoring Tools:**
    - For monitoring the cluster, I installed the following Helm charts:
        - **helm repo add stable https://charts.helm.sh/stable**
        - **helm repo add prometheus-community https://prometheus-community.github.io/helm-charts**
        - **helm search repo prometheus-community**
2. **Create Prometheus Namespace:**
    - Created a **Prometheus** namespace for the monitoring tools to be deployed using the **kubectl create namespace prometheus** command.

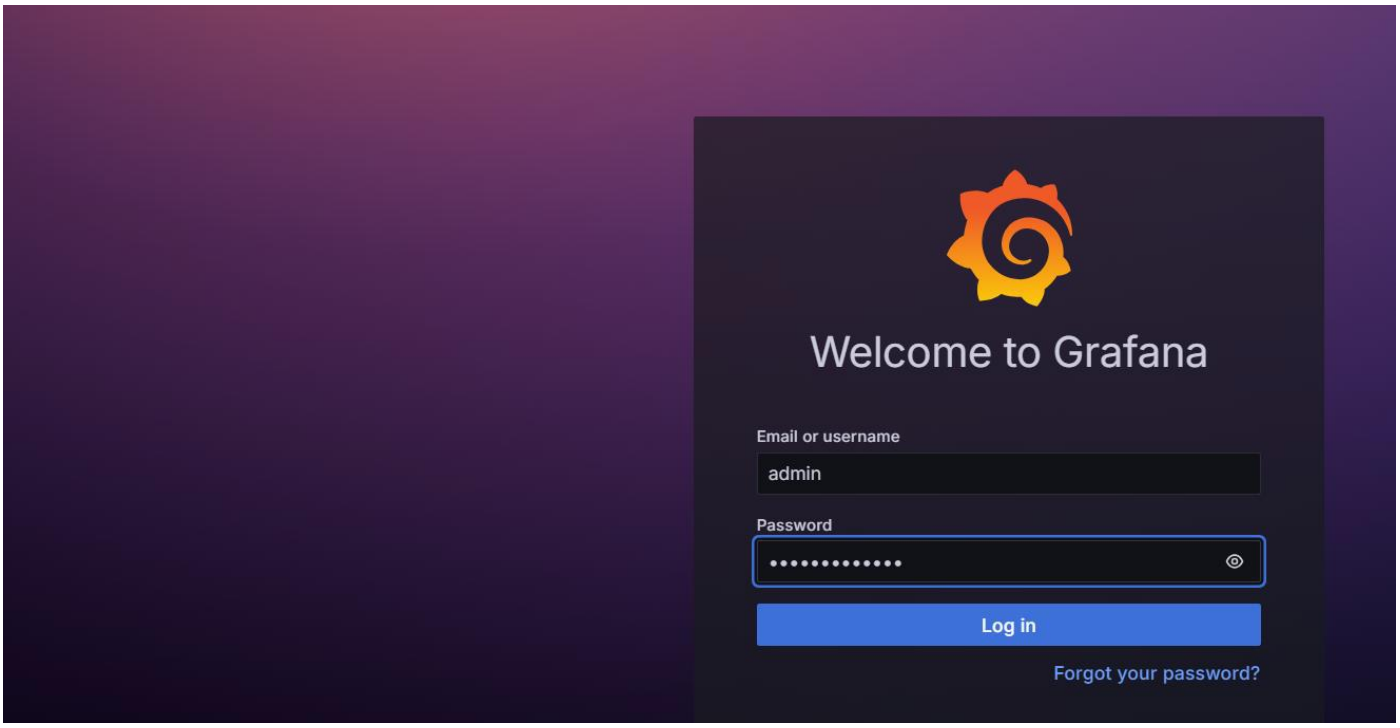
```
NAME                                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                       AGE
alertmanager-operated                       ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP    6m59s
prometheus-operated                         ClusterIP   None             <none>        9090/TCP                      6m56s
stable-grafana                              NodePort    10.109.90.183    <none>        80:30261/TCP                  7m8s
stable-kube-prometheus-sta-alertmanager     ClusterIP   10.105.236.205   <none>        9093/TCP,8080/TCP             7m8s
stable-kube-prometheus-sta-operator         ClusterIP   10.99.137.131    <none>        443/TCP                       7m8s
stable-kube-prometheus-sta-prometheus       NodePort    10.96.40.152     <none>        9090:32074/TCP,8080:30217/TCP 7m8s
stable-kube-state-metrics                   ClusterIP   10.108.2.167     <none>        8080/TCP                      7m8s
stable-prometheus-node-exporter             ClusterIP   10.109.36.141    <none>        9100/TCP                      7m8s
```

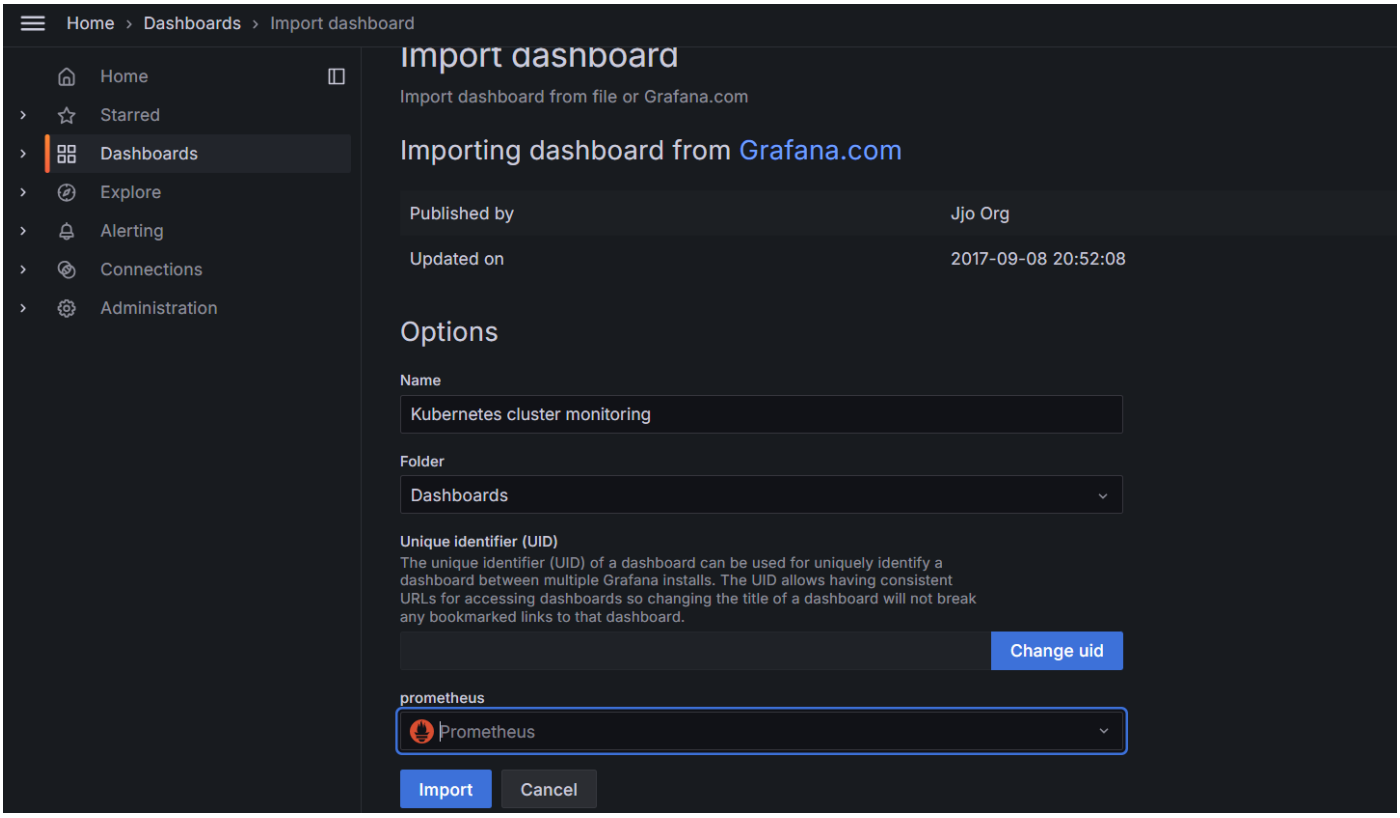
3. **Verify Monitoring Pods:**

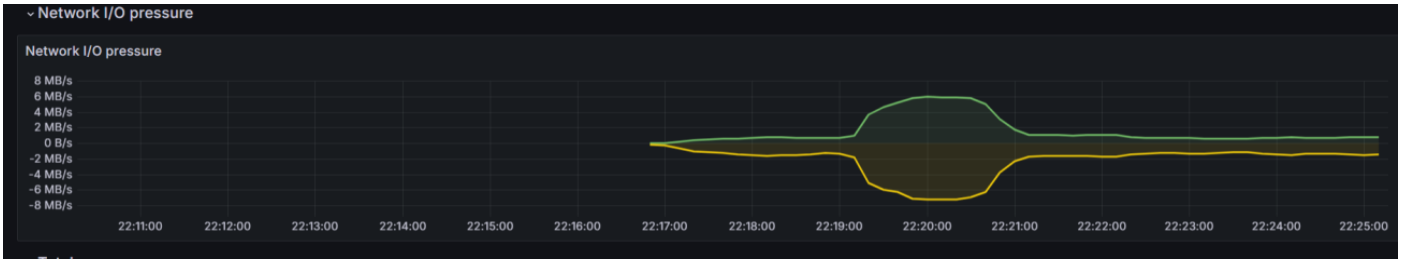
- Confirmed the Prometheus and monitoring pods were running properly using **kubectl get pods -n prometheus**.

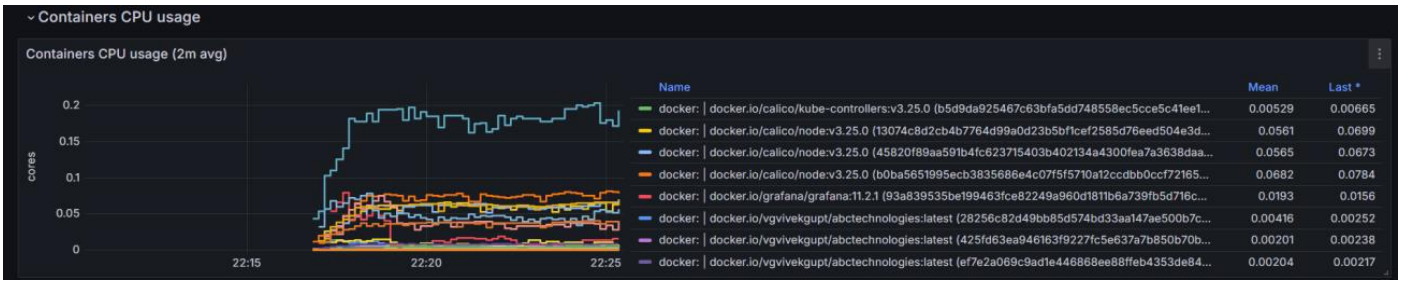
4. **Grafana Dashboard Setup:**
   - Logged into **Grafana** using **NodePort** and set up dashboards to monitor the application performance.

```
NAME                                                READY   STATUS    RESTARTS   AGE
alertmanager-stable-kube-prometheus-sta-alertmanager-0   2/2     Running   0          6m40s
prometheus-stable-kube-prometheus-sta-prometheus-0       2/2     Running   0          6m37s
stable-grafana-5fff8dc495-b9kp6                     3/3     Running   0          6m48s
stable-kube-prometheus-sta-operator-6cf7d5cf64-kkwvm     1/1     Running   0          6m48s
stable-kube-state-metrics-784c9bff7d-b9ps7          1/1     Running   0          6m48s
stable-prometheus-node-exporter-7jnh5               1/1     Running   0          6m48s
stable-prometheus-node-exporter-n9xfx               1/1     Running   0          6m48s
stable-prometheus-node-exporter-qpxgs               1/1     Running   0          6m48s
```

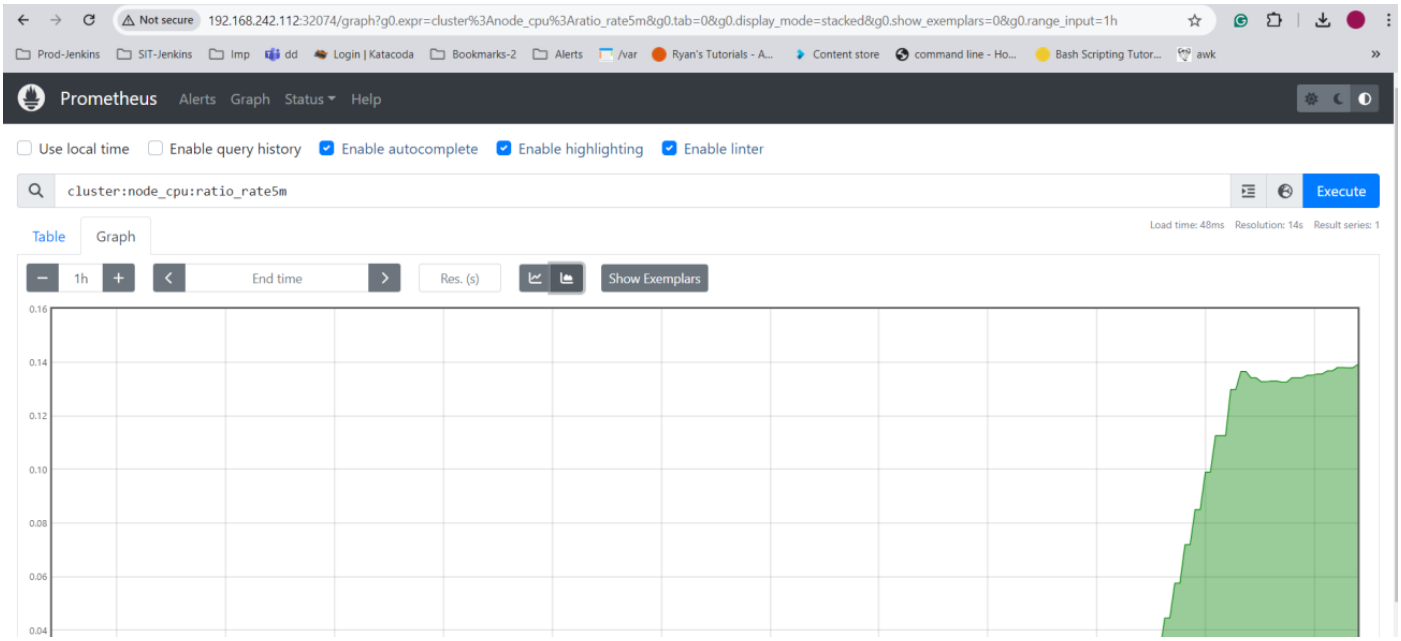



Home › Dashboards › Import dashboard
Home
Starred
Dashboards
Explore
Alerting
Connections
Administration
Import dashboard
Import dashboard from file or Grafana.com
Importing dashboard from Grafana.com
Published by
Jjo Org
Updated on
2017-09-08 20:52:08
Options
Name
Kubernetes cluster monitoring
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
Change uid
prometheus
Prometheus
Import
Cancel


Network I/O pressure
Network I/O pressure
8 MB/s
6 MB/s
4 MB/s
2 MB/s
0 B/s
-2 MB/s
-4 MB/s
-6 MB/s
-8 MB/s
22:11:00
22:12:00
22:13:00
22:14:00
22:15:00
22:16:00
22:17:00
22:18:00
22:19:00
22:20:00
22:21:00
22:22:00
22:23:00
22:24:00
22:25:00


Containers CPU usage
Containers CPU usage (2m avg)
cores
0.2
0.15
0.1
0.05
0
22:15
22:20
22:25
Name
Mean
Last *
docker: | docker.io/calico/kube-controllers:v3.25.0 (b5d9da925467c63bfa5dd748558ec5cce5c41ee1... 0.00529 0.00665
docker: | docker.io/calico/node:v3.25.0 (13074c8d2cb4b7764d99a0d23b5bf1cef2585d76eed504e3d... 0.0561 0.0699
docker: | docker.io/calico/node:v3.25.0 (45820f89aa591b4fc623715403b402134a4300fea7a3638daa... 0.0565 0.0673
docker: | docker.io/calico/node:v3.25.0 (b0ba5651995ecb3835686e4c07f5f5710a12ccdbb0ccf72165... 0.0682 0.0784
docker: | docker.io/grafana/grafana:11.2.1 (93a839535be199463fce82249a960d1811b6a739fb5d716c... 0.0193 0.0156
docker: | docker.io/vgvivekgupt/abctechnologies:latest (28256c82d49bb85d574bd33aa147ae500b7c... 0.00416 0.00252
docker: | docker.io/vgvivekgupt/abctechnologies:latest (425fd63ea946163f9227fc5e637a7b850b70b... 0.00201 0.00238
docker: | docker.io/vgvivekgupt/abctechnologies:latest (ef7e2a069c9ad1e446868ee88ffeb4353de84... 0.00204 0.00217


Containers memory usage
Containers memory usage
1.25 GiB
1 GiB
768 MiB
512 MiB
256 MiB
0 B
22:11:00
2024-10-03 22:11:20
docker: | registry.k8s.io/kube-apiserver:v1.29.7 (64a7e3d1d30334fad062fabd7c3e69656e189245ff80732b7f0f9827a77e7c3e) 667 MiB
docker: | docker.io/calico/node:v3.25.0 (45820f89aa591b4fc623715403b402134a4300fea7a3638daa1be925c76d58b0) 341 MiB
docker: | docker.io/vgvivekgupt/abctechnologies:latest (ef7e2a069c9ad1e446868ee88ffeb4353de8478a62ef83b676daa6556343aa36) 254 MiB
docker: | docker.io/vgvivekgupt/abctechnologies:latest (28256c82d49bb85d574bd33aa147ae500b7cf4a81c3e134d5a18c512650764b8) 199 MiB
docker: | registry.k8s.io/etcd:3.5.12-0 (0da361153597c510a0645e14d214ea362a9267541ebb63ff8e7f3b712f467db1) 198 MiB
Name
All processe



5. **Monitoring Data:**
    - Monitored various metrics, including:
        - **Network input/output** data.
        - **CPU utilization** of containers.
        - **Memory utilization** of containers.
6. **Prometheus Queries:**
    - Configured **Prometheus** queries for detailed insights and performance data of the Kubernetes cluster and containers.

Conclusion:

This project demonstrates the full cycle of DevOps automation for **ABC Technologies**, from **source code management** to **continuous integration** and **deployment** with Docker and Kubernetes. The integration of **monitoring tools** like Prometheus and Grafana further enhances the observability of the system, allowing for better management and troubleshooting.

By completing this project, I have gained hands-on experience with key DevOps tools, which will help in automating deployments, scaling applications, and ensuring their stability in production environments for **ABC Technologies**.