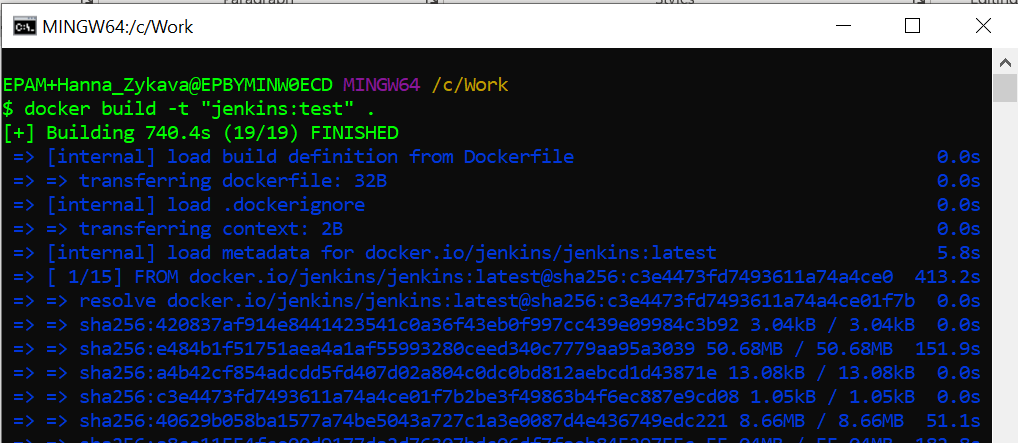
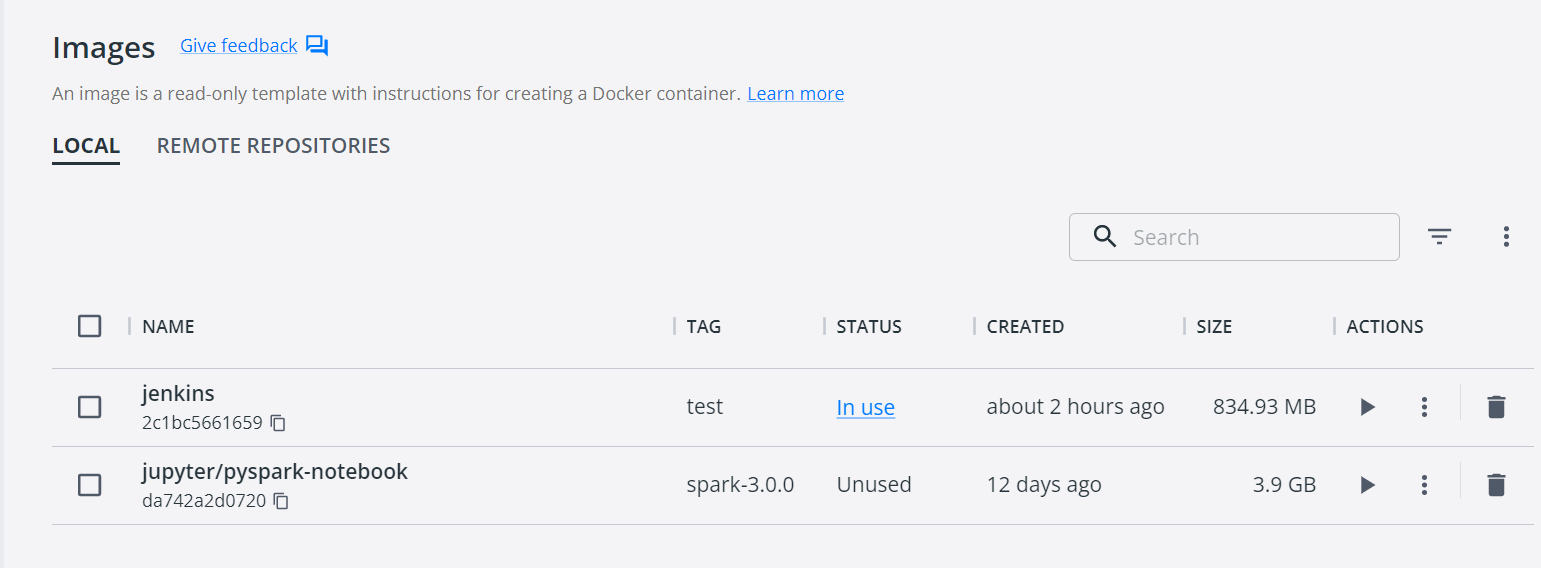
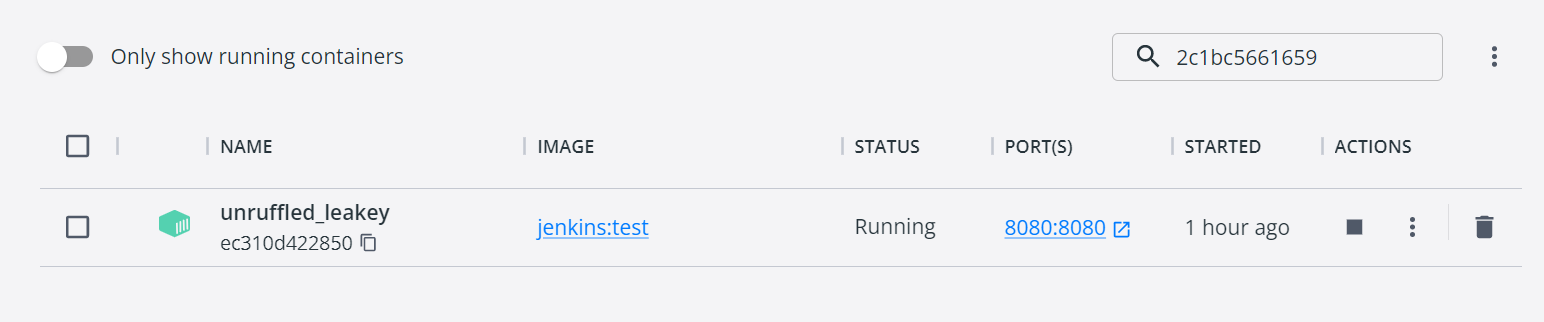
**Report for Module 5 CI-CD basics for DQE**

1. Build docker image using Dockerfile:  
     
   
2. Check that image appeared:

Graphical user interface, text, application

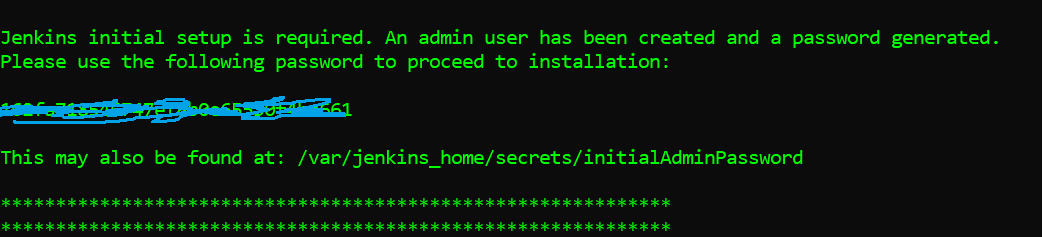
Description automatically generated



1. Run docker using the port 8080:8080 and image ‘jenkins:test’  
     
   

Graphical user interface, text, application

Description automatically generated

1. Copy password in cash:  
     
   
2. In browser page write localhost:8080 and enter the password, then install all proposed plugins:  
     
   Graphical user interface, text

   Description automatically generated  
     
   Graphical user interface, application, table

   Description automatically generated
3. Enter credentials:

Graphical user interface, text, application, email

Description automatically generated

1. Install the following plugins in Plugin manager:

[Python Plugin](https://plugins.jenkins.io/python);

[Robot FrameworkVersion](https://plugins.jenkins.io/robot);

[Build Pipeline](https://plugins.jenkins.io/build-pipeline-plugin);

[Docker PipelineVersion](https://plugins.jenkins.io/docker-workflow);

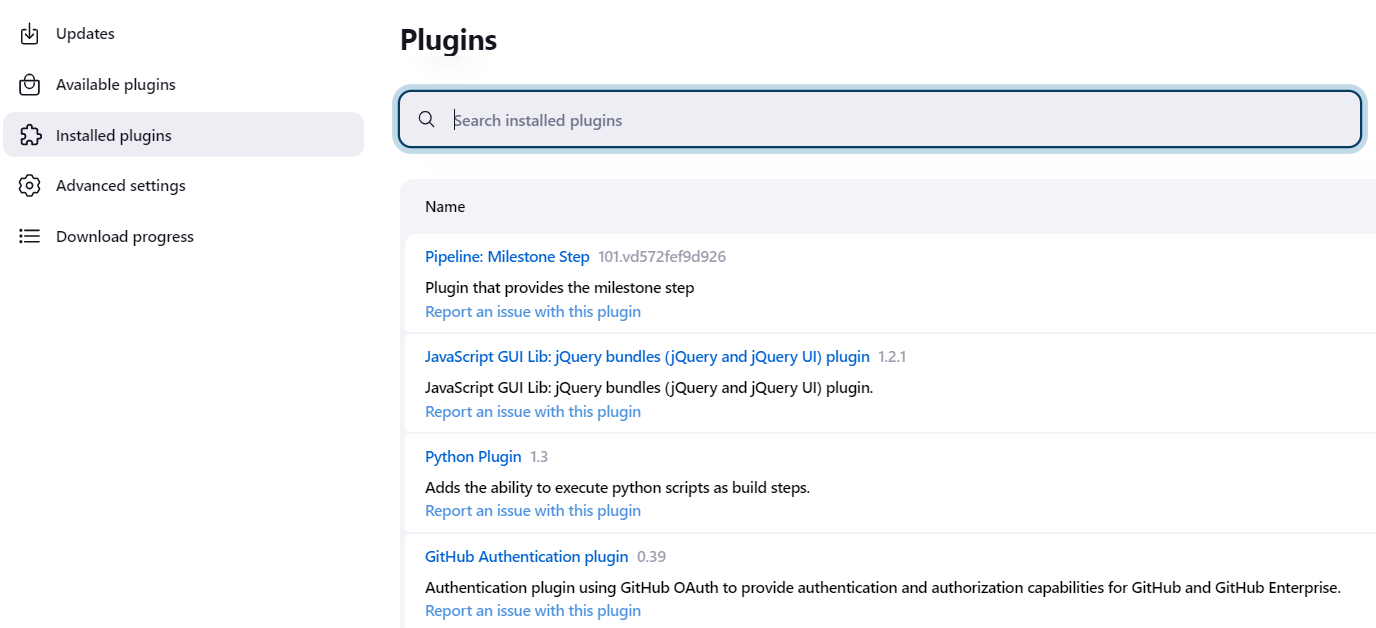
[Microsoft SQL Server Database](https://plugins.jenkins.io/database-sqlserver);

[GitHub Integration](https://plugins.jenkins.io/github-pullrequest);

[GitHub Authentication](https://plugins.jenkins.io/github-oauth);

[GitHub Checks](https://plugins.jenkins.io/github-checks);

[GitHub Issues](https://plugins.jenkins.io/github-issues)



1. Create new item “Pipeline”:  
     
    Graphical user interface, text, application, email

   Description automatically generated

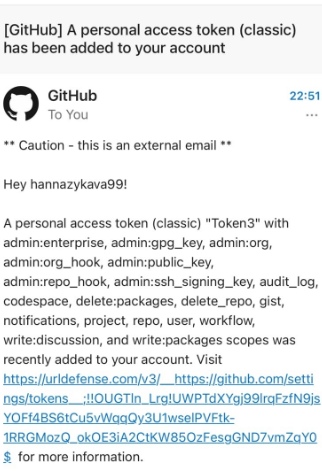
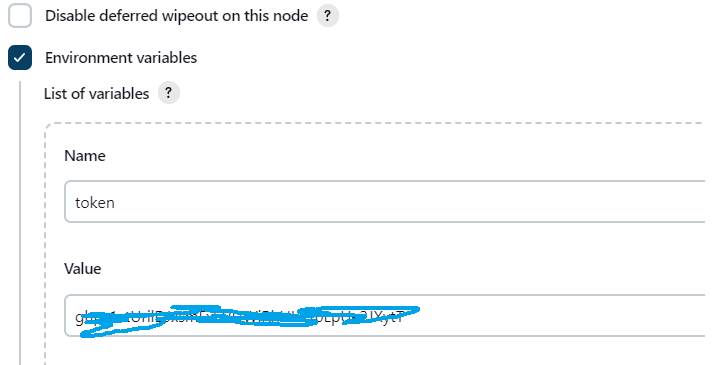
1. In configurations description choose SCM -> Git and add the GitHub and Jenkins file:

Graphical user interface, text, application, chat or text message

Description automatically generated  
Graphical user interface, text

Description automatically generated

1. Create token in GitHub and add it to Jenkins variables:

1. In Dashboard section run the created pipeline to check that it works:

Graphical user interface, application

Description automatically generated  
Table, calendar

Description automatically generated

Check logs:  
  
Table

Description automatically generated  
  
If I run pipelines many times, it works well without merge conflicts:

Table

Description automatically generated with medium confidence

Example how it works:

Let’s go to main branch and add 8th test case, commit changes:

Graphical user interface, text, application, email

Description automatically generated

Then run this pipeline.

When it is finished, let’s check how changes were implemented to pre\_prod branch:

Graphical user interface, text, application

Description automatically generated

The 8th test case was added to pre\_prod branch:  
  
Graphical user interface, text, application, email

Description automatically generated

The detailed console output is attached in GitHub in *Console Output.txt.*

1. Schedule:

|  |  |  |  |
| --- | --- | --- | --- |
| Release | Description | Code | Status |
| Pre\_prod | every month at midnight of the first and 15th days of the month | H 0 1,15 \* \* | Done |
| Prod | every month at midnight of 16th day of the month | H 0 16 \* \* | TBD |

Graphical user interface, text, application, email, Teams

Description automatically generated