

Hiren
Rathod

Software Industrialization

EPITA 2020

Documentation

CentOS Installation

We will make a use of CentOS 8 DVD. iOS for the process of software industrialization by installing it using virtual box environment.

You can download DVD iOS for virtual box by clicking [here...](#)

Before that you need to download and setup VirtualBox environment for running the CentOS operating system as a server or UIX



You can follow the installation instruction by using the following [Link](#)

A screenshot of a terminal window titled "CentOS-Hiren [Running] - Oracle VM VirtualBox". The window has a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". The terminal output shows the CentOS Linux 8 (Core) boot process, including the kernel version (4.18.0-193.6.3.el8_2.x86_64) and the architecture (x86_64). It prompts the user to activate the web console with the command "systemctl enable --now cockpit.socket". The user logs in as "hiren-si" and runs the command "whoami", which returns "hiren-si".

```
CentOS Linux 8 (Core)
Kernel 4.18.0-193.6.3.el8_2.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

localhost login: hiren-si
Password:
Last login: Mon Jun 29 20:31:24 from 192.168.0.103
[hiren-si@localhost ~]$ whoami
hiren-si
[hiren-si@localhost ~]$
```

Install Docker Engine on CentOS

We will make a use o to install Docker Engine, you need a maintained version of CentOS 7. Archived versions aren't supported or tested.

If you already have an old docker version you need to remove it from the version and start it by fresh.

You can remove old version by using following command

```
$ sudo yum remove docker \
    docker-client \
    docker-client-latest \
    docker-common \
    docker-latest \
    docker-latest-logrotate \
    docker-logrotate \
    docker-engine
```



Installation Process:

Step 1:

You first need to set up the docker repository

Run the following commands into your machine

```
$ sudo yum install -y yum-utils

$ sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

Step 2: Install Docker Engine

Run the command in your machine:

```
$ sudo yum install docker-ce docker-ce-cli containerd.io
```

Step 3: Install Docker Compose

Run this command to download the current stable release of Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Apply executable permission to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

To test the docker-compose is installed or not run the following command:

- `docker-compose --version`

Step 4: Configuration and Testing for Docker And Docker-Compose.

1. Start a Docker service:
 - `sudo systemctl start docker`
2. Verify the docker engine by using following command:
 - `sudo docker run hello-world`



Step 5: Jenkins Setup

We will setup Jenkins using docker-compose commands



To pull the image from the repo configure the ports and bindings the volumes from host machine to our machine:

Step 1:

Create a directory using:

`"mkdir software-industrialization"`

And change the directory by using

`"cd software-industrialization"`

Step 2:

Create another directory to save Jenkins data into it :

`"mkdir Jenkins"`

And give that directory owner permission recursively

`"chown 1000:1000 Jenkins -R"`

then

Create a docker-compose.yml file

`"vi docker-compose.yml"`

And



Add the following lines in file

```
version: '3.5'  
services:  
  jenkins:  
    image: "jenkins/jenkins:lts"  
    ports:  
      - "10380:8080"  
    restart: "always"  
    volumes:  
      - "./jenkins:/var/jenkins_home"
```

- save the file by using following commands
press ESC key
SHIFT + ; key
wq and hit ENTER



Step 2:

Run the following command to execute Jenkins:

- docker-compose up -d jenkins

Step 3:

Go to your browser and in URL box type your IP address and following port that you have added in docker-compose.yml file

Eg: 192.168.0.102:10380

Jenkins will start.....

Step 4:

It will ask for admin passcode:

Use: "docker -f logs jenkins"

It will show you passcode in a format

.....
aksdj234513jasjh322421
.....

copy paste the passcode and click on next button

- register your-self by adding username
passcode
- Click on setup Jenkins and here you go.. it
will start Jenkins and push you to Jenkins
dashboard

Step 6: Sonatype/Nexus Setup

Step 1: by using the same procedure as we did before for Jenkins

Create another directory for binding nexus data into it:

"mkdir nexus"

And give that directory owner permission recursively

"chown 999 nexus -R"

Add the following lines to your docker-compose.yml file:



nexus:

image: sonatype/nexus3

ports:

- "10680:8081"

volumes:

- "./nexus:/nexus-data"

- Save the file go to the directory where docker-compose.yml is and run the following command:
"docker-compose up -d nexus"
- Visit browser and add you ipaddress add port to run nexus live
"192.168.0.102: 10680"
- After the setup is up it will ask for username and password used by default credentials
Username: admin
Password: goto nexus folder there you'll find admin.password file copy that password and paste it there to log in.
- Your nexus is setup and running...

Step 7: SonarQube Setup

Step 1: by using the same procedure as we did before for Jenkins

Create another directory to save SonarQube data into it :

"mkdir sonarqube"

And give that directory owner permission recursively

"chown 999 sonarqube -R"

then

Add the following lines to your docker-compose.yml file:

```
sonarqube:
  image: "sonarqube:lts"
  ports:
    - "10580:9000"
  volumes:
    - ./sonarqube/data:/opt/sonarqube/data
```

- Save the file go to the directory where docker-compose.yml is and run the following command:
"docker-compose up -d sonarqube"
- Visit browser and add you ipaddress add port to run nexus live
"192.168.0.102: 10580"
- After the setup is up it will ask for username and password used by default credentials
Username: admin
Password: admin
- Your sonarqube it setup and running...



Step 8: Gitea Setup

Step 1: by using the same procedure

Create another directory for binding gitea data into it :

"mkdir gitea"

then

Add the following lines to your docker-compose.yml file:

gitea:

image: "gitea/gitea:latest"

ports:

- "10122:22"

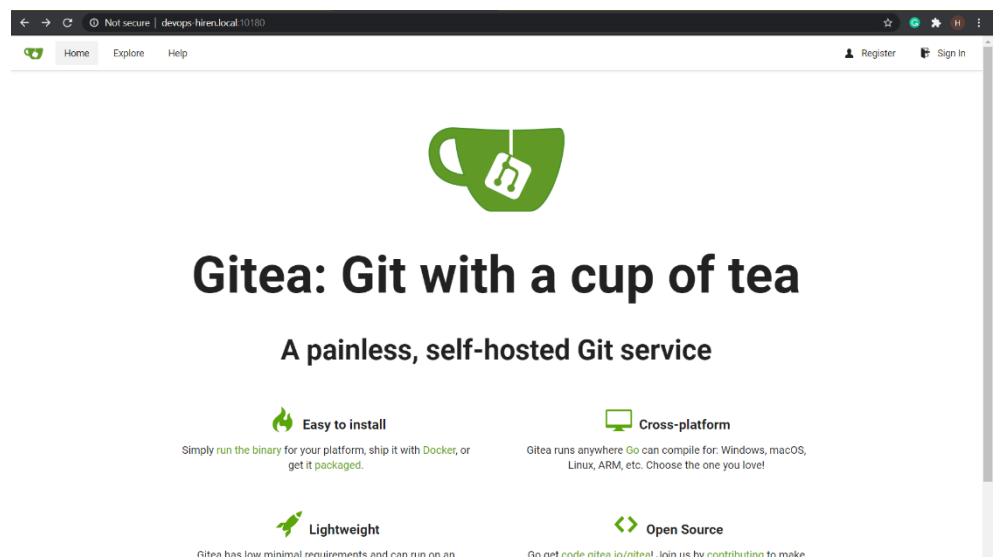
- "10180:3000"

volumes:

- ./gitea:/data



- Save the file go to the directory where docker-compose.yml is and run the following command:
"docker-compose up -d gitea"
- Visit browser and add you ipaddress add port to run nexus live
"192.168.0.102: 10180"
- Your gitea is setup and running...



⇒ Building, Testing and deploying java maven project with Jenkins and Nexus3

Step 1:

- Open nexus and Jenkins in your browser on the following ports which we binded in our docker-compose.yml file and login with the following credentials as shown above.
- To deploy your maven project and push it to nexus repository manager you need to do the following procedure first:

- **Create your role in nexus**



The screenshot shows the Sonatype Nexus Repository Manager (OSS 3.24.0-02) interface. The left sidebar has a menu with 'Administration' (Repository, Blob Stores, Content Selectors, Cleanup Policies, Routing Rules), 'Security' (Privileges), 'Roles' (highlighted), 'Users', 'Anonymous Access', 'LDAP', 'Realms', 'SSL Certificates', 'IQ Server', 'Support', and 'System'. The main content area is titled 'Roles / deployment'. It includes a 'Delete role' button and a 'Settings' tab. The configuration fields are: 'Role ID' (deployment), 'Role name' (deployment), 'Role description' (deployment), and 'Privileges'. Under 'Privileges', there are two sections: 'Available' and 'Given'. The 'Available' section has a filter and a list of roles including 'nx-all', 'nx-analytics-all', 'nx-apikey-all', 'nx-atlas-all', 'nx-blobstores-all', 'nx-blobstores-create', 'nx-blobstores-delete', 'nx-blobstores-read', and 'nx-blobstores-update'. The 'Given' section contains a list of specific privileges: 'nx-repository-admin-maven2-maven-central-*', 'nx-repository-admin-maven2-maven-central-browse', 'nx-repository-admin-maven2-maven-central-edit', 'nx-repository-admin-maven2-maven-central-read', 'nx-repository-admin-maven2-maven-snapshots-*', 'nx-repository-admin-maven2-maven-snapshots-browse', 'nx-repository-admin-maven2-maven-snapshots-delete', 'nx-repository-admin-maven2-maven-snapshots-edit', and 'nx-repository-admin-maven2-maven-snapshots-read'.

- Create a user and add your role to the user to handle all your work

The screenshot shows the Sonatype Nexus Repository Manager Administration interface. The left sidebar contains a navigation menu with categories: Administration (Repository, Security, Users, Anonymous Access, LDAP, Realms, SSL Certificates, IQ Server, Support, System) and a search bar. The main content area is titled 'Users / HIREN RATHOD' and includes buttons for 'Delete user' and 'Change password'. Below these is a 'Settings' tab. The settings form includes fields for ID (hiren), First name (HIREN), Last name (RATHOD), Email (hr:hirenmeck@gmail.com), and Status (Active). The Roles section shows 'Available' roles (deployment, nx-anonymous) and 'Granted' roles (nx-admin).

- Browsers to Jenkins-data folder in CentOS => goto tools => config => open settings.xml file

The screenshot shows a file explorer window with the address bar displaying the path: `/.../hiren-si/software-l/jenkins/tools/hudson.tasks.Maven_MavenInstallation/Maven-Home/conf/`. The file list contains the following items:

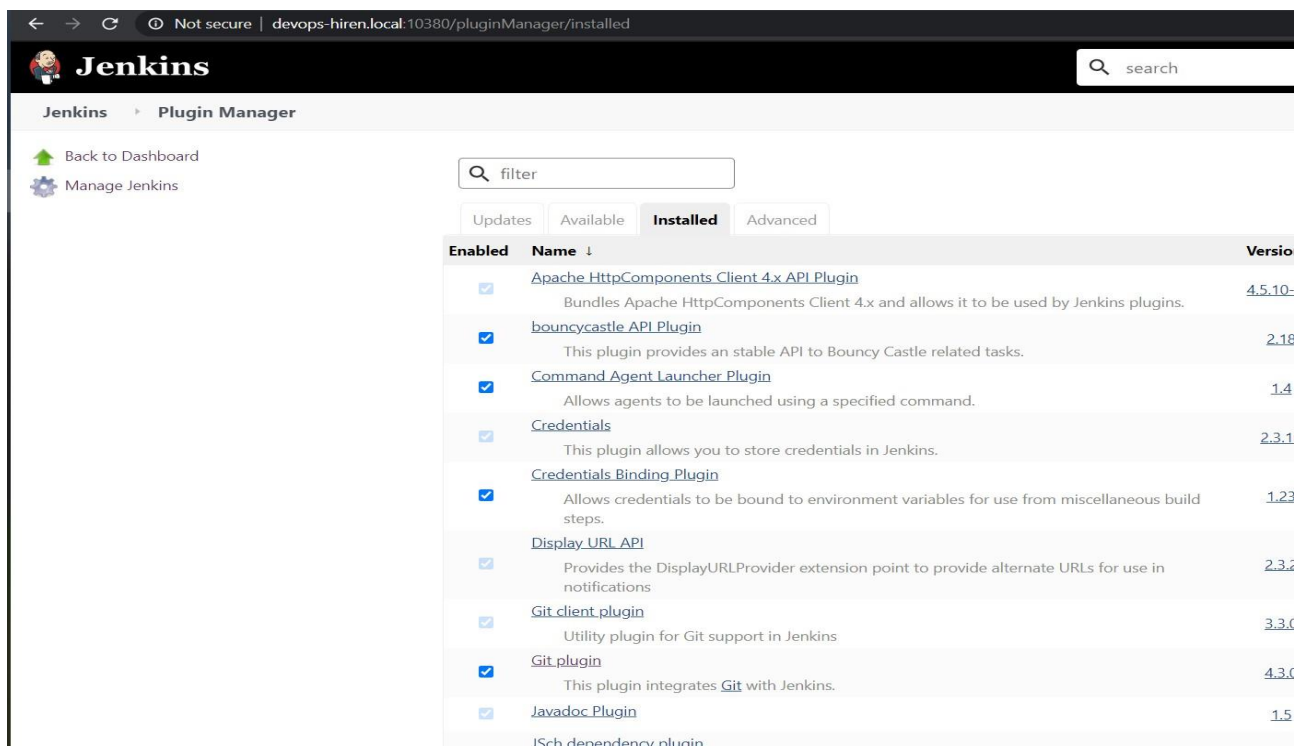
Name	Size	Changed	Rights
logging		6/29/2020 8:43:54 PM	rwxr-xr-x
settings.xml	10 KB	6/30/2020 1:21:56 AM	rwxr-xr-x
toolchains.xml	4 KB	11/7/2019 6:02:18 PM	rwxr-xr-x

- Add user ID and password that you created in nexus into the file as shown below

```
<servers>
  <!-- server
    | Specifies the authentication information to use when
    | a unique name within the system (referred to by the
    |
    | NOTE: You should either specify username/password
    |       used together.
  -->
  <server>
    <id>hirenRepo</id>
    <username>hiren</username>
    <password>Hiren@54321</password>
  </server>
</servers>
```

- Go to Jenkins Add a new job as a maven project.
- Go to manage plugins section in Jenkins And install the required plugins from available section in it.

1. Git & Maven



The screenshot shows the Jenkins Plugin Manager interface. The browser address bar indicates the URL is `devops-hiren.local:10380/pluginManager/installed`. The Jenkins logo and a search bar are at the top. The main heading is "Jenkins > Plugin Manager". On the left, there are links for "Back to Dashboard" and "Manage Jenkins". The main content area has a "filter" search bar and tabs for "Updates", "Available", "Installed", and "Advanced". The "Installed" tab is selected, showing a table of installed plugins.

Enabled	Name	Version
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin	4.5.10-
<input checked="" type="checkbox"/>	bouncycastle API Plugin	2.18
<input checked="" type="checkbox"/>	Command Agent Launcher Plugin	1.4
<input checked="" type="checkbox"/>	Credentials	2.3.1
<input checked="" type="checkbox"/>	Credentials Binding Plugin	1.23
<input checked="" type="checkbox"/>	Display URL API	2.3.2
<input checked="" type="checkbox"/>	Git client plugin	3.3.0
<input checked="" type="checkbox"/>	Git plugin	4.3.0
<input checked="" type="checkbox"/>	Javadoc Plugin	1.5
<input checked="" type="checkbox"/>	JSch dependency plugin	

- Click on the project and open config project
- In the SCM section select Git and add your GitHub URL in it.

The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' radio button is selected. Under 'Repositories', the 'Repository URL' is set to 'https://github.com/ricardoandre97/jenkins-ci-cd.git' and 'Credentials' is set to '- none -'. There are 'Advanced...' and 'Add Repository' buttons. Under 'Branches to build', the 'Branch Specifier (blank for \'any\')' is set to '*/master'. There is a red 'X' icon and an 'Add Branch' button. The 'Repository browser' is set to '(Auto)'. At the bottom, there is an 'Additional Behaviours' section with an 'Add' button.

- Go to Build section and add the following details

The screenshot shows the 'Build' configuration page in Jenkins. The 'Root POM' is set to 'build/simple-java-maven-app/pom.xml'. The 'Goals and options' field contains '-DaltDeploymentRepository=hirenRepo::default::http://nexus:8081/repository/my-app-repo deploy'. The 'MAVEN_OPTS' field is empty. There are help icons (?) next to each field.

- In the image above **hirenRepo** is the ID that we created in settings.xml file for nexus repository
- <http://nexus:8081/repository/my-app-repo> is the path where we want to push the deployed jar/war/pom file

- **“Deploy”** is a maven command to deploy the project.

⇒ Result

The screenshot shows the Sonatype Nexus Repository Manager interface. The browser address bar indicates the URL: `devops-hiren.local:10680/#browse/search=keyword%3Dmy-app:my-app-repo%3Acom.mycompany.app%3Amy-app%3A1.0-SNAPSHOT`. The page title is "Sonatype Nexus Repository Manager OSS 3.24.0-02". The search bar contains "my-app". The left sidebar shows the "Search" menu item selected. The main content area displays the search results for "my-app".

Search Results:

Repository	my-app-repo	Group	com.mycompany.app	Most popular version	
Format	maven2	Name	my-app	Age	
		Version	1.0-SNAPSHOT	Popularity	

Buttons: [Browse SNAPSHOT\(s\)](#) [Analyze application](#)

File List:

Name
com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-20200630.063556-2.jar
com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-20200630.063556-2.jar.md5
com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-20200630.063556-2.jar.sha1
com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-20200630.063556-2.pom
com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-20200630.063556-2.pom.md5
com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-20200630.063556-2.pom.sha1

⇒ Building, Testing, analyzing your Java Maven project using Jenkins and SonarQube.

- Open sonarqube and Jenkins in your browser on the following ports which we binded in our docker-compose.yml file and login with the following credentials as shown above.

Jenkins:



Jenkins



sonarqube

The screenshot shows the Jenkins web interface for a project named 'FirstJob'. The browser address bar indicates the URL is 'devops-hiren.local:10380/job/FirstJob/'. The Jenkins logo and a search bar are at the top. A left sidebar contains navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Rename'. The main content area is titled 'Project FirstJob' and includes links for 'Workspace' and 'Recent Changes'. Below this is a 'Permalinks' section with a list of build links: 'Last build (#12), 3 min 16 sec ago', 'Last stable build (#12), 3 min 16 sec ago', 'Last successful build (#12), 3 min 16 sec ago', 'Last failed build (#11), 10 hr ago', and 'Last unsuccessful build (#11), 10 hr ago'. At the bottom left, a 'Build History' table shows build #12 on Jun 30, 2020 at 6:04 AM.

Build History
#12 Jun 30, 2020 6:04 AM

SonarQube

The screenshot shows the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is on the right. The main content area shows a project named 'my-app' with a 'master' branch. The 'Issues' tab is selected, displaying a list of 3 issues. The first issue is highlighted: 'Make this final field static too.' (Code Smell). The second issue is highlighted: 'Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.' (Code Smell). The third issue is highlighted: 'Replace this use of System.out or System.err by a logger.' (Code Smell). The right sidebar shows the source code of 'src/main/java/com/mycompany/app/App.java' with line numbers 1 to 17. The code includes a package declaration, a comment, a class definition, and a main method. The issues are linked to specific lines in the code: 'Make this final field static too.' is linked to line 9, 'Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.' is linked to line 11, and 'Replace this use of System.out or System.err by a logger.' is linked to line 14.

- Click on **Administration** tab then Click on **Marketplace** and install the required plugins.
- For example for maven I have installed **"SonarJava & Bugs"**

The screenshot shows the SonarQube Administration page. The 'Administration' tab is selected in the top navigation bar. The 'Marketplace' sub-tab is selected in the left sidebar. The main content area displays the 'Marketplace' section. It states 'You are currently running a Community Edition.' and 'Discover more features with SonarSource Editions:'. There are three cards for different editions: 'Developer Edition', 'Enterprise Edition', and 'Data Center Edition'. Each card lists features and has a link to 'Ask for more information'. The 'Developer Edition' features include: Branch and Pull Requests analysis, Analysis of additional languages: C/C++, Objective-C, PL/SQL, ABAP, TSQL, Swift, Detection of security vulnerabilities, and SonarLint notifications. The 'Enterprise Edition' features include: Portfolio management, Executive reporting, Analysis of additional languages: COBOL, PL/I, RPG & VB6, and Parallel processing of analysis reports. The 'Data Center Edition' features include: Support for High-Availability. Allow every node of SonarQube to be redundant, in order to keep the service up at all times without worrying about downtime or interruption. Below the Marketplace section, there is a 'Plugins' section. It has tabs for 'All', 'Installed', and 'Updates Only'. A search bar is present with the text 'quboo' entered.

- Click on the project and open config project
- In the SCM section select Git and add your GitHub URL in it.

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Additional Behaviours

- Go to Build section and add the following details

Build

Invoke top-level Maven targets

Maven Version

Goals

POM

Properties

- Save the project and click on **Build** to see the magic
- Visit sonarqube and hit refresh and you can see your project is uploaded and analyzed

The screenshot displays the SonarQube web interface. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A search bar is present on the right. The main content area is divided into two panels. The left panel, titled '1 / 3 issues', lists three code smells: 'Make this final field static too.', 'Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.', and 'Replace this use of System.out or System.err by a logger.'. The right panel shows the source code of 'App.java' with three issues highlighted. Each issue includes a description, a 'See Rule' link, a severity level (Minor, Critical, Major), a status (Open, Not assigned), an effort estimate (2min, 5min, 10min), and a comment section. The issues are: 1. 'Make this final field static too.' (Minor, Open, 2min effort, convention tag). 2. 'Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.' (Critical, Open, 5min effort, suspicious tag). 3. 'Replace this use of System.out or System.err by a logger.' (Major, Open, 10min effort, bad-practice, cert tag).

```
1 package com.mycompany.app;
2
3 /**
4  * Hello world!
5  */
6 public class App
7 {
8
9     private final String message = "Hello World from an update2!";
10
11     public App() {}
12
13     public static void main(String[] args) {
14         System.out.println(new App().getMessage());
15     }
16
17     private final String getMessage() {
18         return message;
19     }
20
21 }
22
```

Whole Docker-Compose YAML FILE Review

```
hiren-si@localhost:~/software-l
version: '3.5'
services:
  jenkins:
    image: "jenkins/jenkins:lts"
    ports:
      - "10380:8080"
    restart: "always"
    volumes:
      - "./jenkins:/var/jenkins_home"

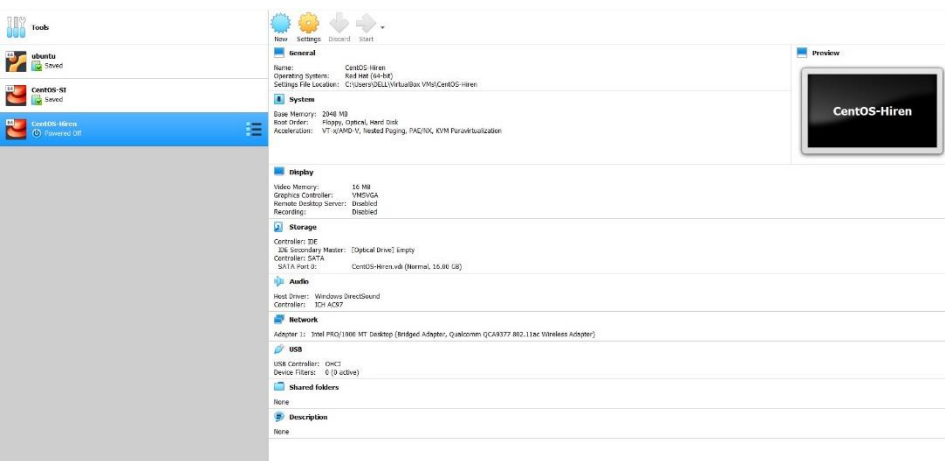
  nexus:
    image: "sonatype/nexus3"
    ports:
      - "10680:8081"
    volumes:
      - "./nexus:/nexus-data"

  gitea:
    image: "gitea/gitea:latest"
    ports:
      - "10122:22"
      - "10180:3000"
    volumes:
      - "./gitea:/data"

  sonarqube:
    image: "sonarqube:lts"

    ports:
      - "10580:9000"

    volumes:
      - "./sonarqube/conf:/opt/sonarqube/conf"
      - "./sonarqube/data:/opt/sonarqube/data"
      - "./sonarqube/logs:/opt/sonarqube/logs"
      - "./sonarqube/extensions:/opt/sonarqube/extensions"
      - "./sonarqube/bundled-plugins:/opt/sonarqube/lib/bundled-plugins"
```



Thank You
