keenable

Customising Redmine and Superset to Allow Automatic Login with Gmail through Keycloak

Task Description:

To integrate Redmine and Apache Superset with Keycloak for easy Gmail login, follow these steps:

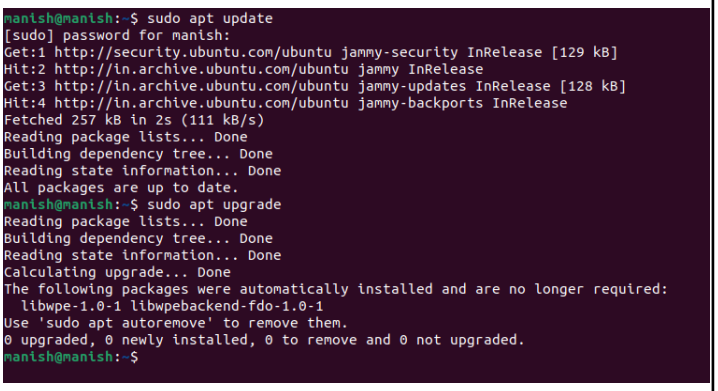
1. Start by setting up Keycloak. Create a realm and set up Google as an identity provider using OAuth credentials from the Google API Console.
2. Next, create separate clients within Keycloak for Redmine and Superset, making sure to specify the right redirect URIs.
3. Then, install the necessary OAuth plugin in Redmine, set it up with Keycloak client details, and update Redmine's configuration file.
4. For Superset, install the required Python packages and update the superset_config.py file to include OIDC configuration for Keycloak.
5. Finally, restart both Redmine and Superset servers to apply the changes.

To verify the integration, access Redmine and Superset using Google credentials, and check that user information is correctly populated.

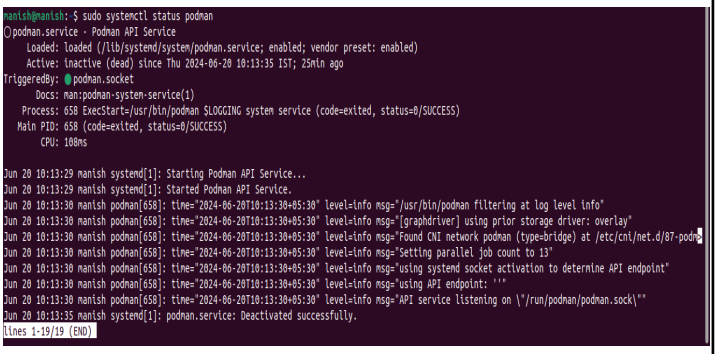
Table of content

Test cases

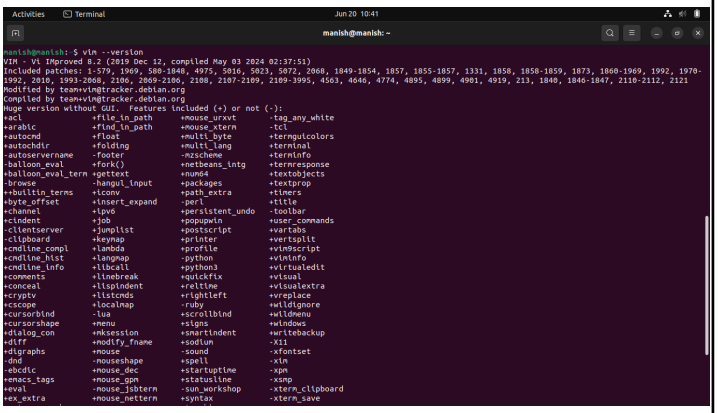
Task: 1

Scenario: Ensure the system is up to date	
Given: User has a terminal open.	**Output:** 
When: User run "sudo apt update"	
And: User run "sudo apt upgrade"	
Then: User system packages updated to the latest versions	
Result: Passed	

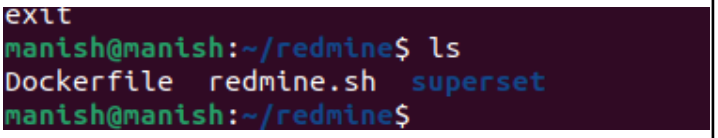
Task: 2

Scenario: Install Podman	
Given: User system packages are up to date	**OUTPUT:** 
When: User run "sudo apt install podman"	
Then: Podman installed on the system	
Result: Passed	

Task: 3

Scenario: Install Vim	
Given: User system packages are up to date	**OUTPUT:** 
When: User run "sudo apt install vim"	
Then: Vim installed on the system	
Result: Passed	

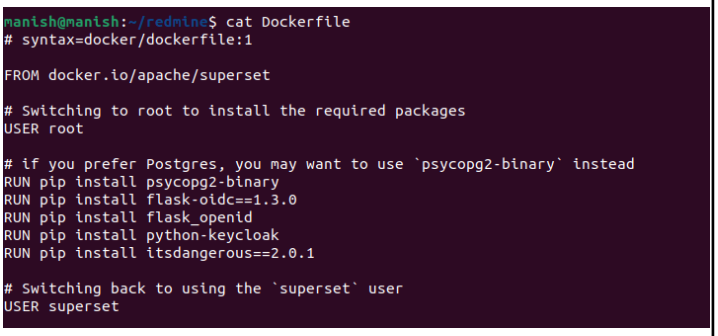
Task: 4

<table>
<tr><td colspan="2">Scenario: Create necessary directories and files for Redmine setup</td></tr>
<tr><td>Given: User Unix-like operating system environment</td><td rowspan="4">Output:
  </td></tr>
<tr><td>When: User run the following commands:

mkdir redmine
touch redmine.sh
vim redmine.sh
cat redmine.sh
touch Dockerfile</td></tr>
<tr><td>Then: User following conditions met:

A directory named redmine is created.
An empty file named redmine.sh is created.
The file redmine.sh can be edited using vim.
The contents of redmine.sh can be displayed using cat.
An empty file named Dockerfile is created.</td></tr>
<tr><td>Result: Passed</td></tr>
</table>

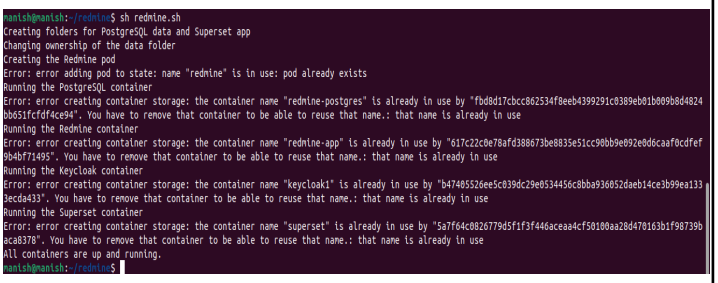
Task: 5

<table>
<tr><td colspan="2">Scenario: Define the Dockerfile for setting up Superset with additional packages</td></tr>
<tr><td>Given: User empty Dockerfile</td><td rowspan="4">Output:
 
</td></tr>
<tr><td>When: User add the following content to the Dockerfile with ss in the output box.</td></tr>
<tr><td>Then: User Dockerfile up a Superset environment with the required packages installed.</td></tr>
<tr><td>Result: Passed</td></tr>
</table>

Task: 6

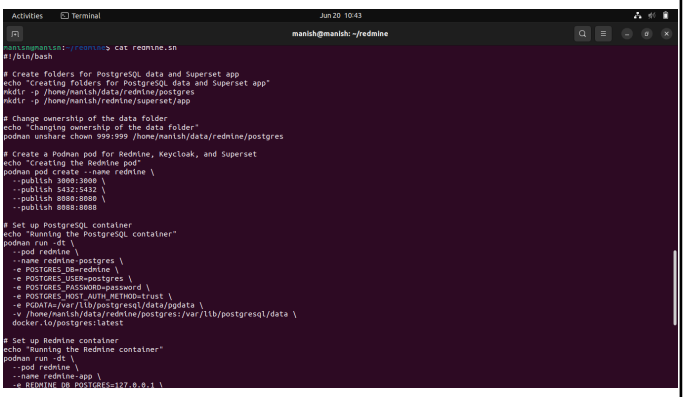
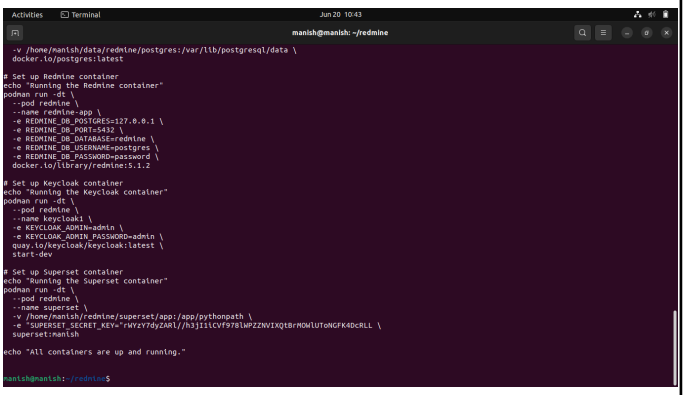
<table>
<tr><td colspan="2">Scenario: Create and run the redmine.sh script to set up containers using Podman</td></tr>
<tr><td>Given: User properly configured redmine.sh script with the following content in the output box.

echo "All containers are up and running."</td><td rowspan="3">Output:

 
  </td></tr>
<tr><td>When: User execute <code>./redmine.sh</code></td></tr>
<tr><td>Then: User following conditions met:

• Directories for PostgreSQL data and Superset app are created.
• Ownership of the PostgreSQL data directory is changed.
• A Podman pod named <code>redmine</code> is created with the specified ports published.
• Containers for PostgreSQL, Redmine, Keycloak, and Superset are created and running within the <code>redmine</code> pod.
• A message indicating "All containers are up and running" is displayed.</td></tr>
<tr><td>Result: Passed</td><td></td></tr>
</table>

Task: 7

Scenario: Set up the Redmine environment and install the OAuth plugin	
Given: User Unix-like operating system with Podman installed	**Output:**  
And: User `redmine.sh` script that sets up necessary directories and containers	
And: User Redmine container connected to a PostgreSQL database	
When: User execute the command sh redmine.sh	
Then: User the script should: • Create directories for PostgreSQL data and the Superset app • Change ownership of the PostgreSQL data directory • Create a Podman pod named redmine with specified ports published • Run containers for PostgreSQL, Redmine, Keycloak, and Superset within the redmine pod • Display a message indicating "All containers are up and running"	
Result: Passed	

Task: 8

Scenario: Access the Redmine container to install the OAuth plugin

Given: User redmine.sh script has been executed and all containers are running

When: User run the command `podman exec -it redmine-app bash`

Then: User gain interactive shell access to the Redmine container

When: User navigate to the plugins directory with `cd plugins/`

And: User list the contents of the directory with `ls`

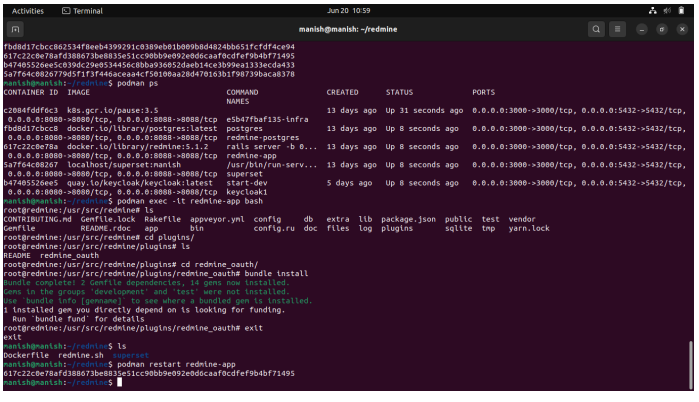
Then: User directory empty or show the current plugins

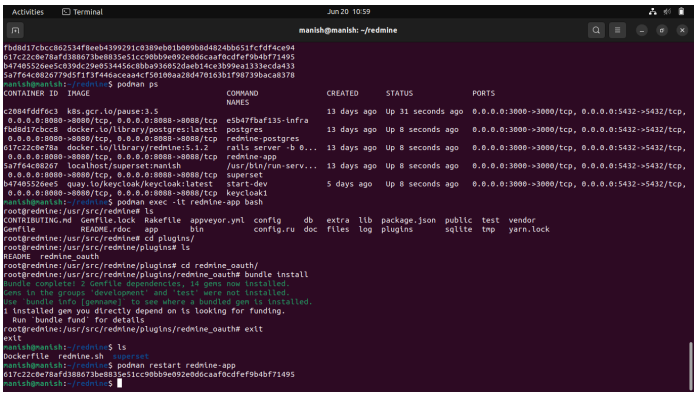
When: User clone the OAuth plugin with git clone https://github.com/kontron/redmine_oauth.git

And: User list the contents of the directory with `ls`

Output:

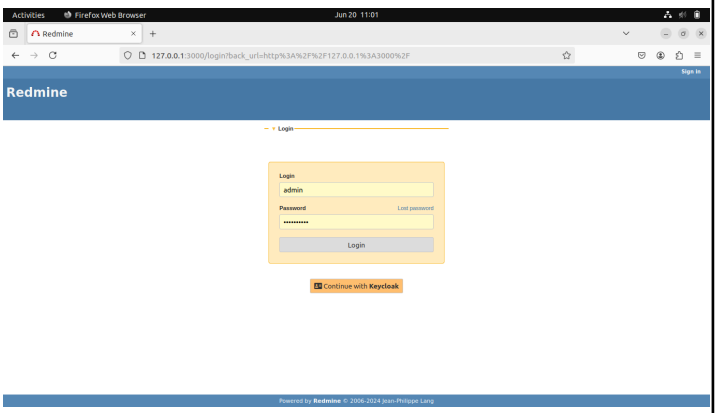
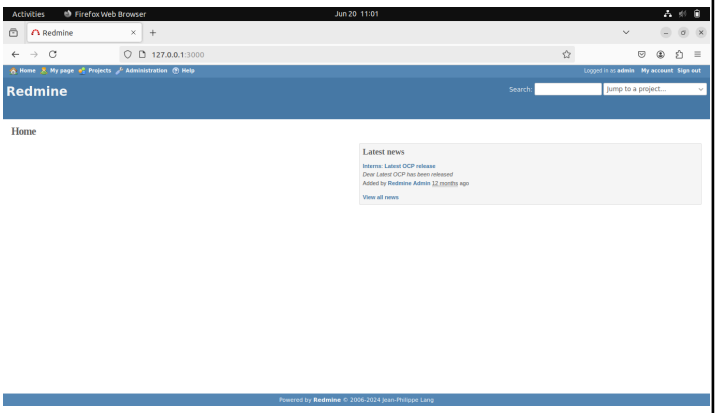
```
manish@manish:~/redmine$ podman exec -it redmine-app bash
root@redmine:/usr/src/redmine# ls
CONTRIBUTING.md  Gemfile.lock  Rakefile  appveyor.yml  config    db   extra  lib  package.json  public  test  vendor
Gemfile          README.rdoc   app       bin           config.ru  doc  files  log  plugins       sqlite  tmp   yarn.lock
root@redmine:/usr/src/redmine# cd plugins/
root@redmine:/usr/src/redmine/plugins# ls
README  redmine_oauth
root@redmine:/usr/src/redmine/plugins# cd redmine_oauth/
root@redmine:/usr/src/redmine/plugins/redmine_oauth# bundle install
Bundle complete! 2 Gemfile dependencies, 14 gems now installed.
Gems in the groups 'development' and 'test' were not installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
1 installed gem you directly depend on is looking for funding.
  Run `bundle fund` for details
root@redmine:/usr/src/redmine/plugins/redmine_oauth#
```

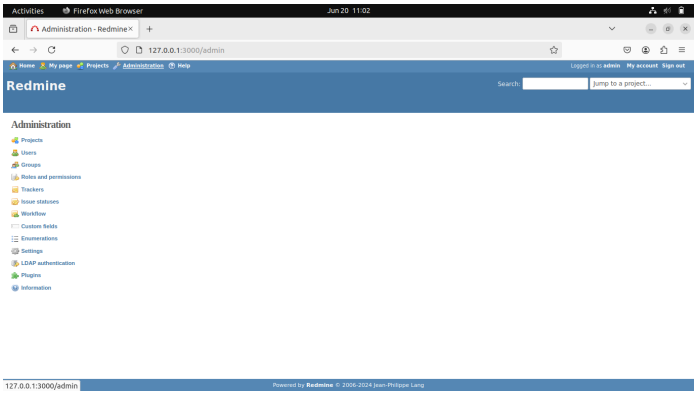
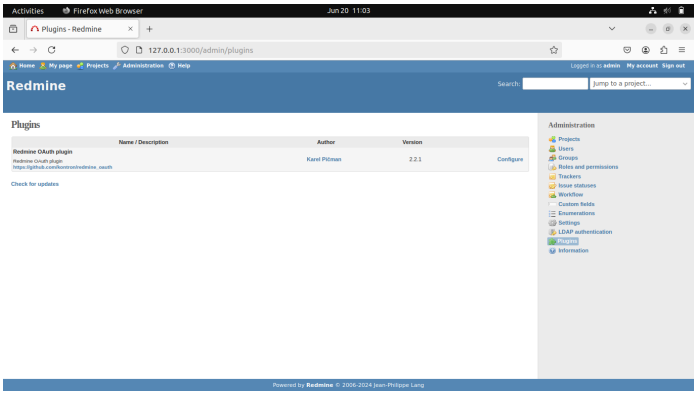
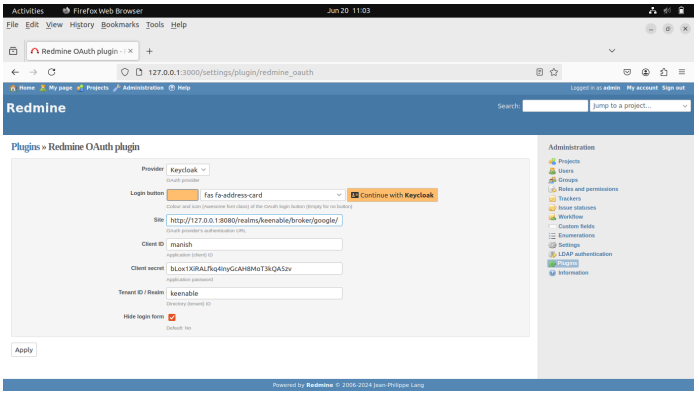


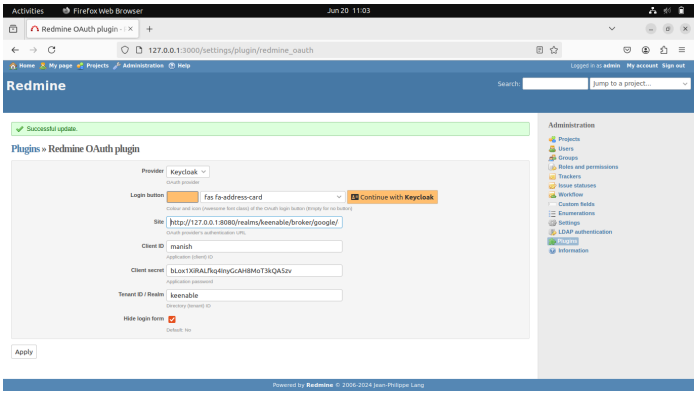
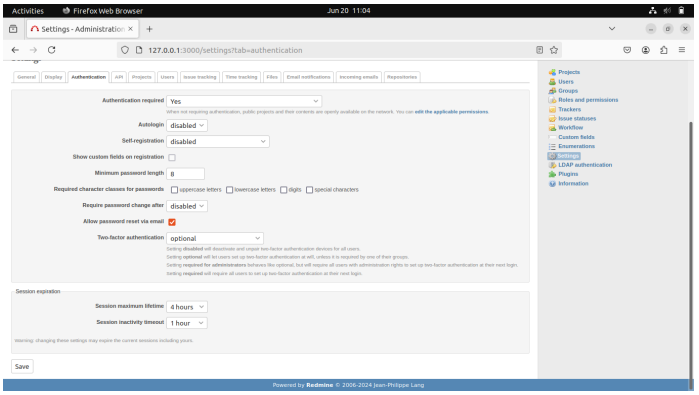
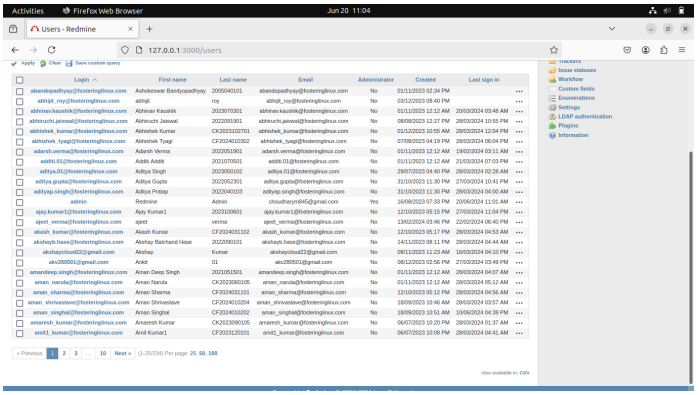
Then: User redmine_oauth directory present	
When: User navigate into the `redmine_oauth` directory with `cd redmine_oauth/`	
And: User install the bundle with `bundle install`	
Then: User necessary dependencies for the `redmine_oauth` plugin installed	
When: USer navigates back to the plugins directory with `cd ..`	
And: User install the bundle in the plugins directory with `bundle install`	
Then: User necessary dependencies for all plugins installed	
When: User restart the Redmine container with `podman restart redmine-app`	

Then: User Redmine application restart with the newly installed `redmine_oauth` plugin	
Result: Passed	

Task: 9

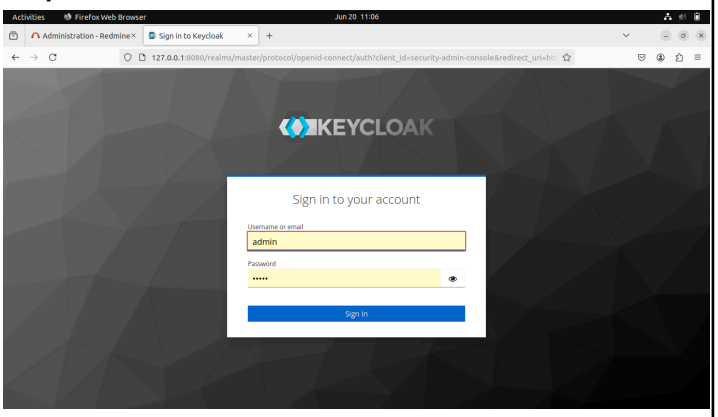
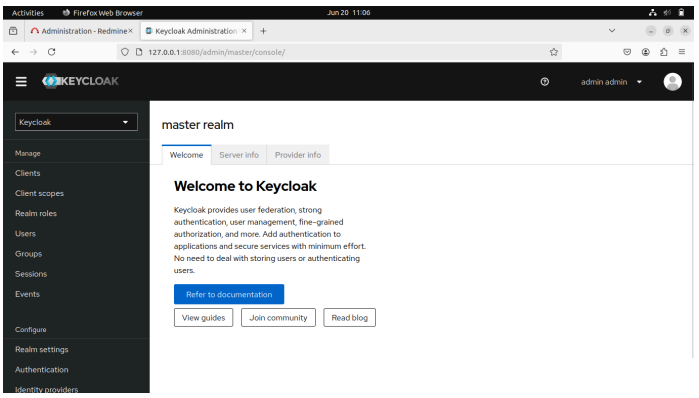
Scenario: Access Redmine web interface and configure OAuth plugin	
Given: User Redmine and Keycloak containers are running	**Output:** 
When: User open a web browser and navigate to `http://127.0.0.1:3000`	
Then: User Redmine login page displayed	

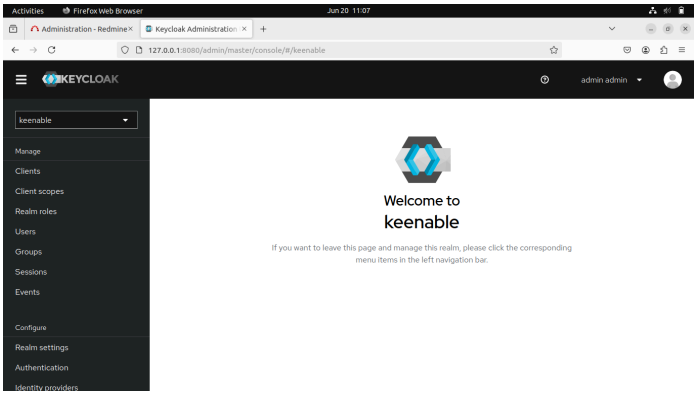
When: User login to Redmine with the username "admin" and password "admin"	
Then: User gain access to the Redmine dashboard	
When: User click the "Administration" option	
Then: User administration page should be displayed	

When: USer click the "Plugins" option	
Then: USer list of installed plugins displayed	
When: User click the "Configure" option for the OAuth plugin	
Then: User plugin configuration page displayed	
When: User click the "Provider" dropdown and select the "Keycloak" option	

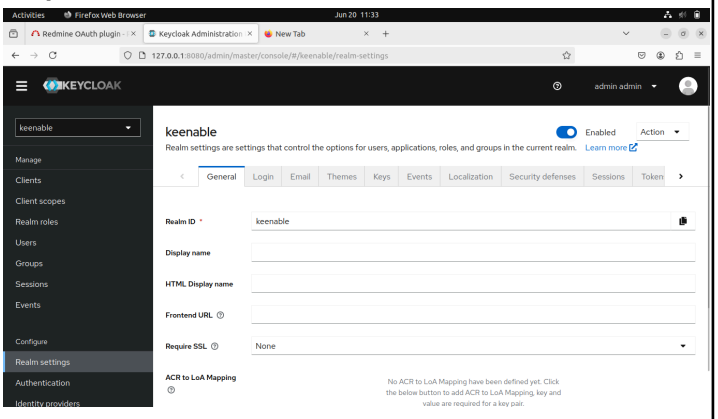
Then: User Keycloak option selected as the OAuth provider	
Result: Passed	

Task: 10

Scenario: Access Keycloak web interface	
Given: User Keycloak container is running	**Output:** 
When: User open a web browser and navigate to http://127.0.0.1:8080	
Then: User Keycloak login page displayed	
When: User log in to Keycloak with the username "admin" and password "admin"	

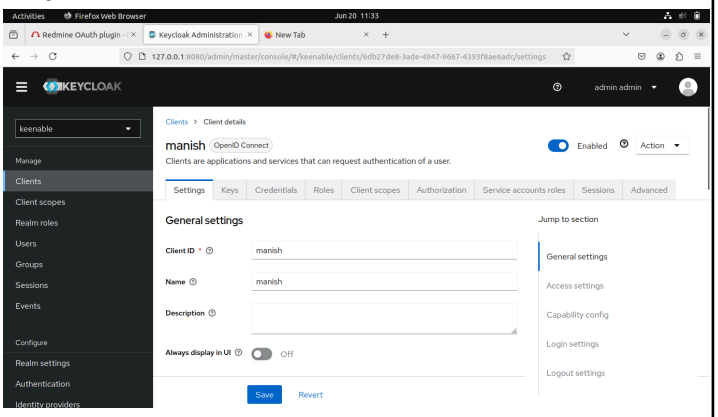
Then: User gain access to the Keycloak admin console	
Result: Passed	

Task: 11

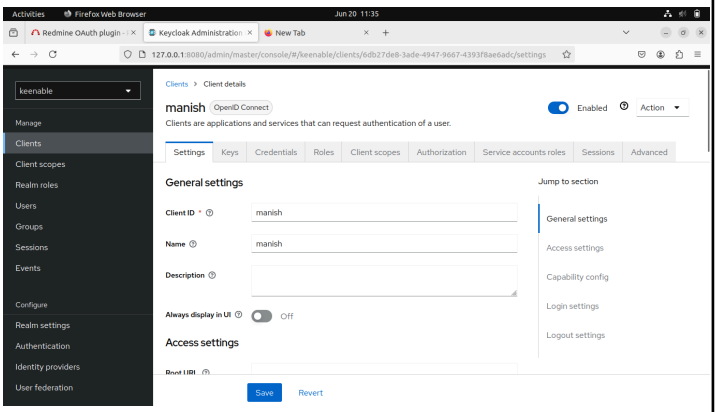
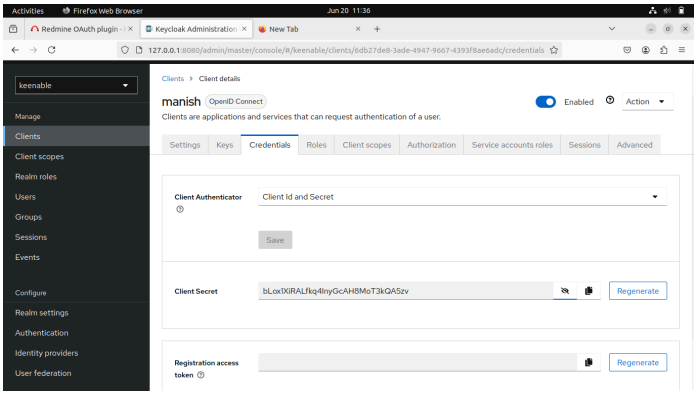
Scenario: Access Keycloak and create a realm	
Given: User logged in to the Keycloak admin console	**Output:** 
When: User navigate to the "Realms" section	
And: User create a new realm with the necessary details	

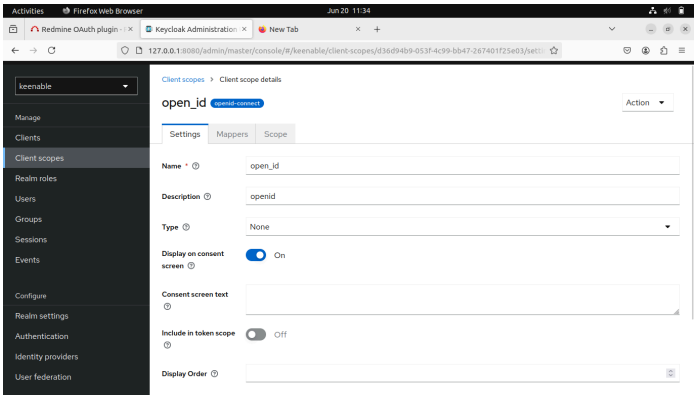
Then: User new realm created and listed in the realms section	
Result: Passed	

Task: 12

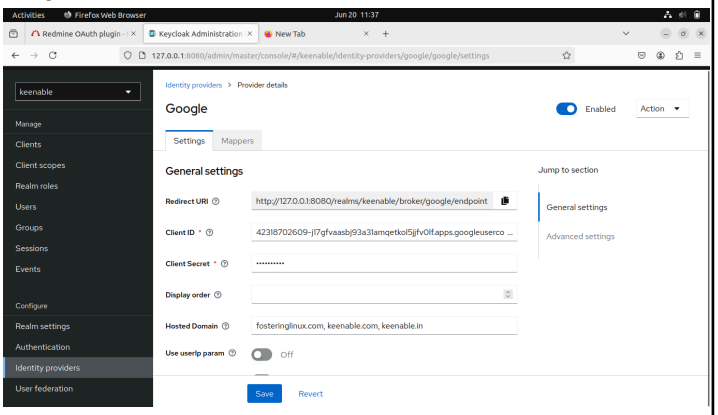
Scenario: Create a client in Keycloak	
Given: User have created a realm in Keycloak	**Output:** 
And: User create a new client with the necessary details	
Then: User new client created and listed in the clients section	
Result: Passed	

Task: 13

Scenario: Configure the client and obtain the client secret	
Given: User have created a client in Keycloak	**Output:** 
When: User click the "Next" button to proceed with the client configuration	
Then: User taken to the client settings page	
When: User navigate to the "Credentials" tab for the client	
Then: User see the client secret	

And: User copy the client secret	
Result: Passed	

Task: 14

Scenario: Configure Identity Provider in Keycloak	
Given: User have copied the client secret	**Output:**
When: User navigate to the "Identity Providers" section within the realm	
Then: User able to configure the identity provider with the necessary details	
Result: Passed	

Task: 15

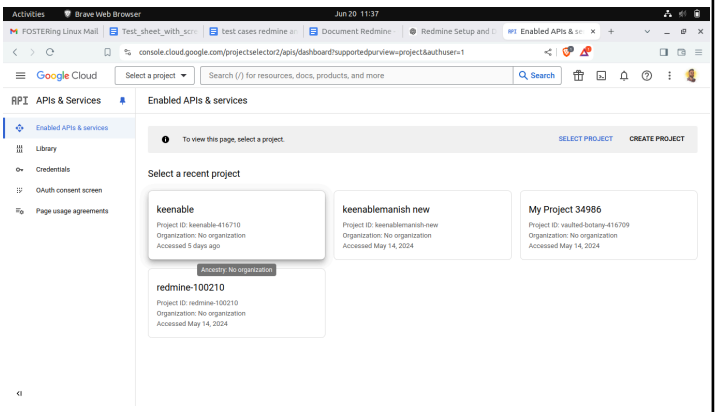
Scenario: Access Google API Dashboard and create OAuth credentials

Given: User have access to the Google API Dashboard using my personal Gmail account

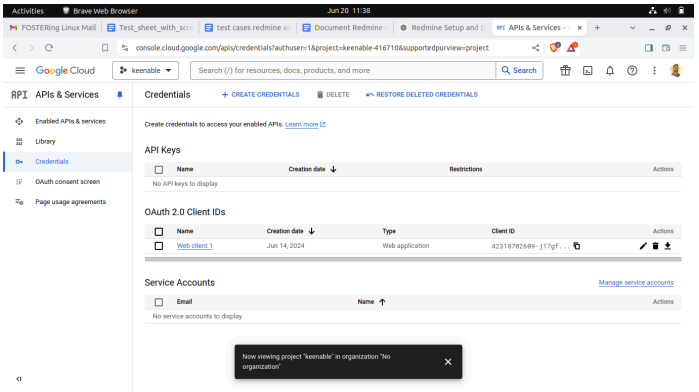
When: User navigate to **https://console.cloud.google.com/projectselector2/apis/dashboard?supportedpurview=project&authuser=1**

Then: User see a list of projects associated with my account

Output:

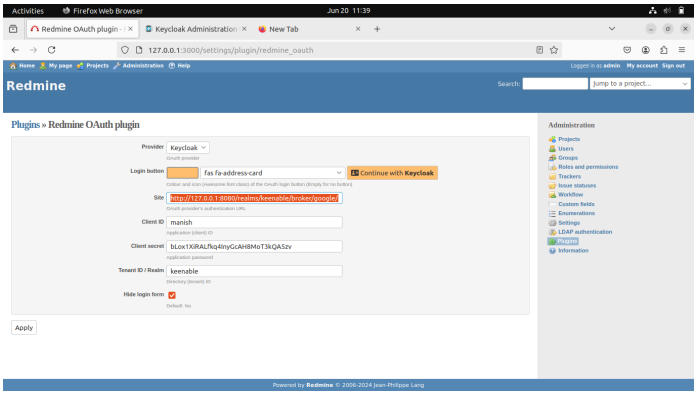


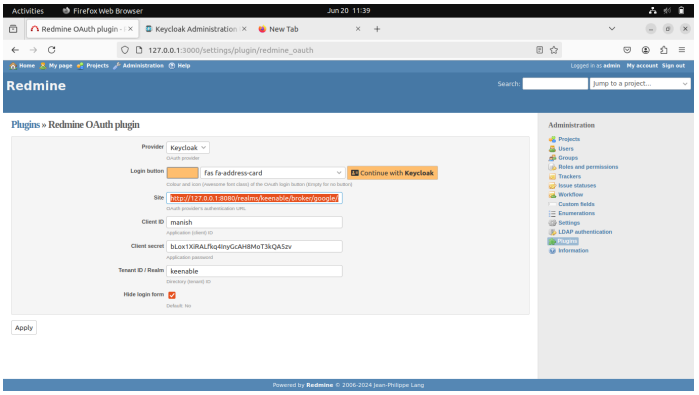
When: User select the keenable project



Then: User project dashboard should be displayed

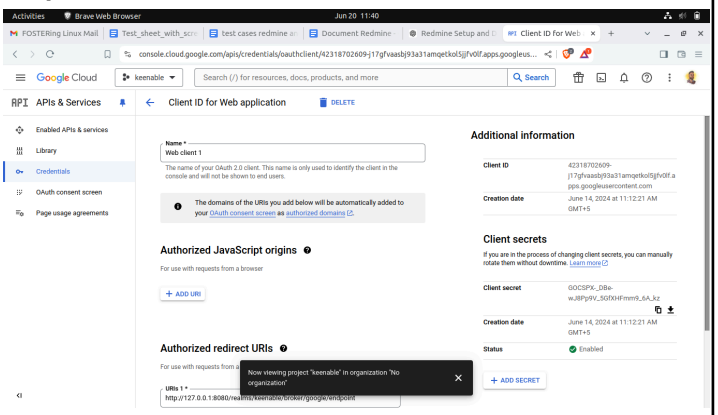
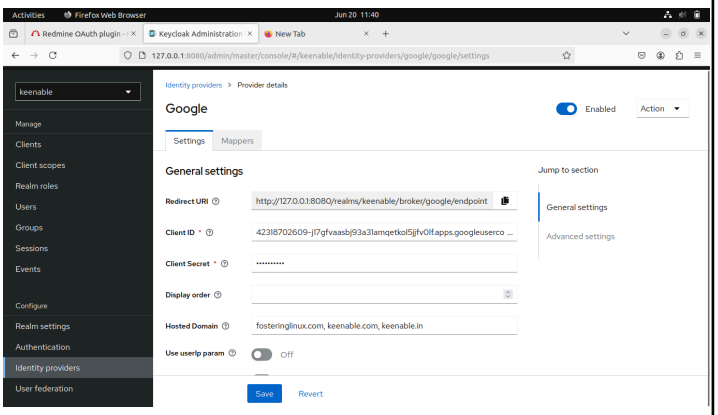
And: User select "Create credentials"



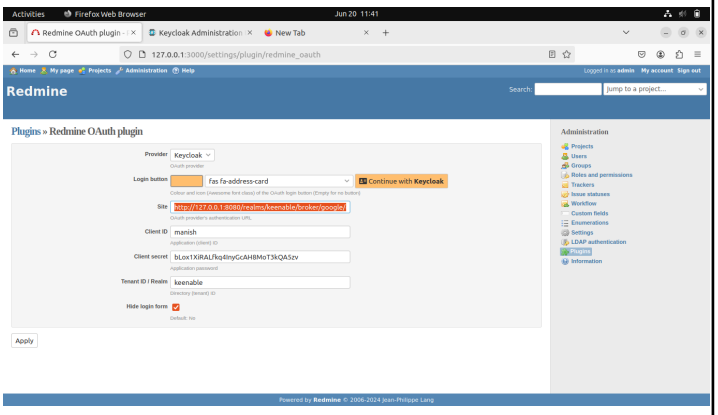
And: User choose "OAuth 2.0 Client IDs"	
Then: User prompted to configure OAuth credentials	
When: User select "Web application" as the application type	
And: User configure the required fields with the necessary information, including the redirect URI http://127.0.0.1:3000/oauth2callback	
And: User create the credentials	
Then: User see the client ID and client secret	
When: User copy the client ID and client secret	
Then: User able to paste these credentials into Keycloak	

Result: Passed	

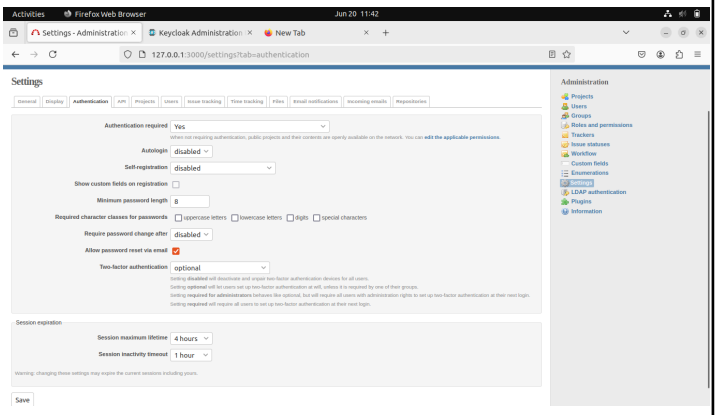
Task: 16

Scenario: Configure Identity Provider in Keycloak with Google OAuth	
Given: User have the client ID and client secret from Google	**Output:** 
When: User navigate to the "Identity Providers" section in Keycloak	
And: User select "Google" as the identity provider	
And: User paste the client ID and client secret into the appropriate fields in Keycloak	
And: User save the configuration	
Then: Keycloak configured to use Google OAuth for authentication	
Result: Passed	

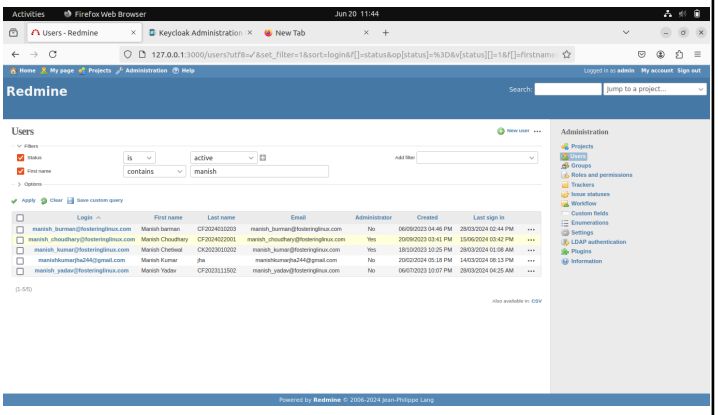
Task: 17

Scenario: Configure Redmine to use Keycloak OAuth plugin	
Given: User logged in to Redmine as an admin	**Output:** 
When: User navigate to the "Administration" section	
And: User select the "Plugins" option	
And: User fill in the necessary configuration details for the Keycloak OAuth plugin	
Then: User plugin configured correctly	
Result: Passed	

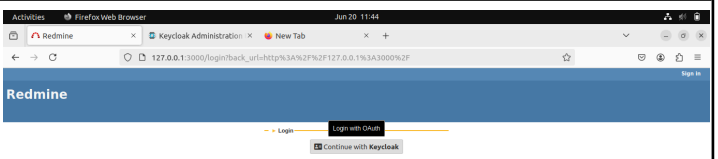
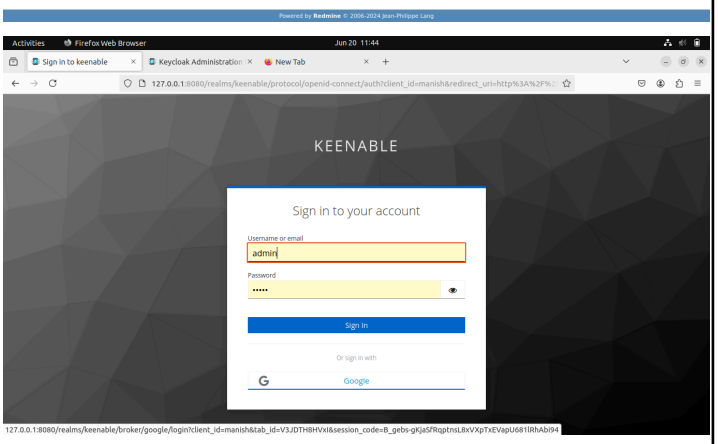
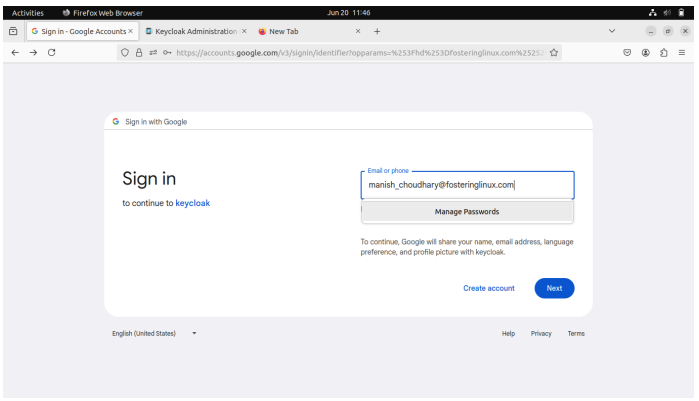
Task: 18

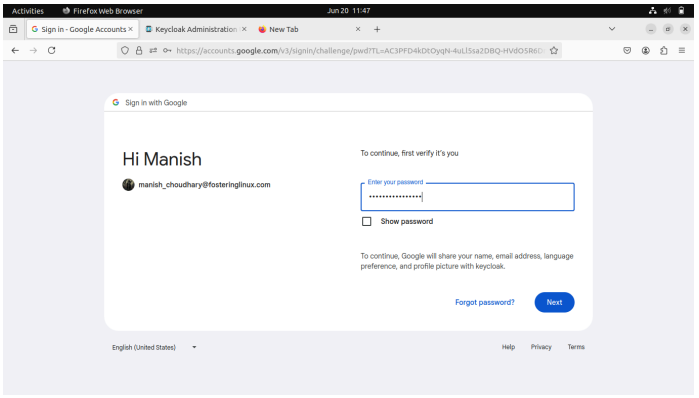
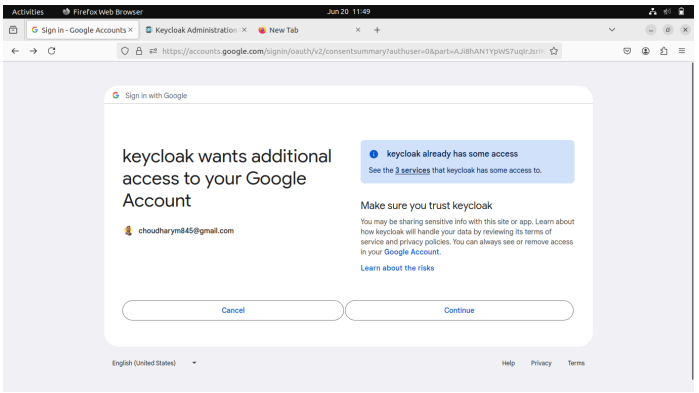
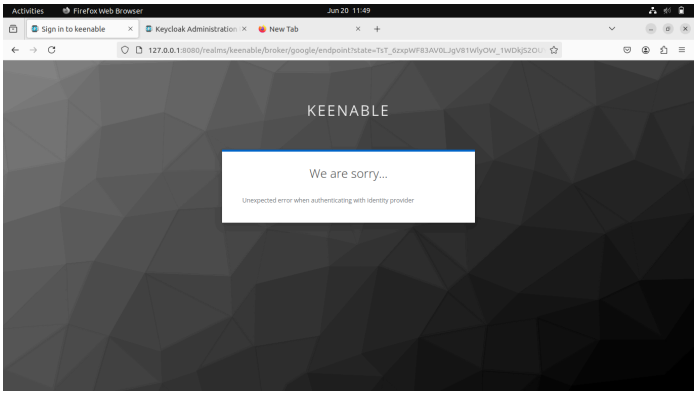
Scenario: Configure Redmine authentication settings	
Given: User have configured the Keycloak OAuth plugin in Redmine	**Output:** 
And: User select the "Authentication" tab	
And: User configure the authentication settings to use Keycloak	
Then: User authentication settings updated correctly	
Result: Passed	

Task: 19

Scenario: Create a user in Redmine	
Given: User am logged in to Redmine as an admin	**Output:** 
When: User navigate to the "Users" section	
And: User create a new user with the email "manish_choudhary@fosteringlinux.com" and other required details	
Then: user created successfully	
Result: Passed	

Task: 20

Scenario: Authenticate with Keycloak and Google OAuth	
Given: User have logged out of Redmine	
When: User navigate to the Redmine login page	
And: User select "Continue with Keycloak"	
Then: User redirected to the Keycloak login page	
When: User select the Google authentication option in Keycloak	
Then: User redirected to the Google login page	

When: User log in with the email "manish_choudhary@fosteringlinux.com"	
When: User log in with the email "choudharym845@gmail.com"	
Then: User see an error indicating that the user does not exist in Redmine	
Result: Passed	

Task: 21

Scenario: Build and run the Superset Docker image

Given: User have a Dockerfile for Superset

When: User run the command "podman build -t superset:manish ."

Then: The Docker image for Superset built and tagged as "superset:manish"

When: User run the command "podman images"

Then: User see the "superset:manish" image listed

When: User generate a secret key with the command "openssl rand -base64 42"

Then: User receive a secret key string

Output:

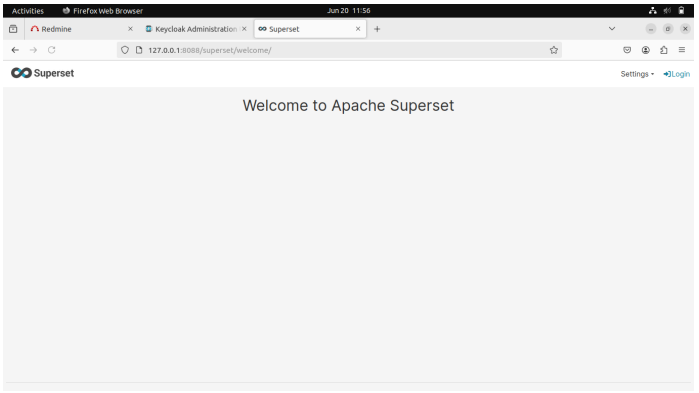
```
manish@manish:~/redmine$ cat Dockerfile
# syntax=docker/dockerfile:1

FROM docker.io/apache/superset

# Switching to root to install the required packages
USER root

# if you prefer Postgres, you may want to use `psycopg2-binary` instead
RUN pip install psycopg2-binary
RUN pip install flask-oidc==1.3.0
RUN pip install flask_openid
RUN pip install python-keycloak
RUN pip install itsdangerous==2.0.1

# Switching back to using the `superset` user
USER superset

manish@manish:~/redmine$
```

```
manish@manish:~/redmine$ podman build -t superset:manish .
STEP 1/8: FROM docker.io/apache/superset
STEP 2/8: USER root
--> Using cache 36e175c728f443baa7df0aaa87167271e3cbbb6b0e13d5012272f452898fde5a
--> 36e175c728f
STEP 3/8: RUN pip install psycopg2-binary
--> Using cache ddfbe70cb58ab327cd0ab2fcc9dad1168f84d451d6a7a1e56516e9cc51c99bd7
--> ddfbe70cb58
STEP 4/8: RUN pip install flask-oidc==1.3.0
--> Using cache d5ca2db9520f90e4ce86f92626284eea14dd3fe18119c325bf20aa08e40c3882
--> d5ca2db9520
STEP 5/8: RUN pip install flask_openid
--> Using cache 6a5b581a11d1557ee3b98add0d7dd929f3567c3ddfa730787ae8d9efde4ba0fc
--> 6a5b581a11d
STEP 6/8: RUN pip install python-keycloak
--> Using cache eea8bddcfe716d36f2cf54b4586e4730c6324a563231d9fbd5533b17f28d5897
--> eea8bddcfe7
STEP 7/8: RUN pip install itsdangerous==2.0.1
--> Using cache 03a7cf64f9ed03a2731371565ba1ab3b07a2e7881bc4869db2bf2177960d5fd2
--> 03a7cf64f9e
STEP 8/8: USER superset
--> Using cache e8693c41fde86be0a0b04e37e6ef93986ff3accc76976a81134dcfd7d148150b
COMMIT superset:manish
--> e8693c41fde
Successfully tagged localhost/superset:manish
e8693c41fde86be0a0b04e37e6ef93986ff3accc76976a81134dcfd7d148150b
manish@manish:~/redmine$
```

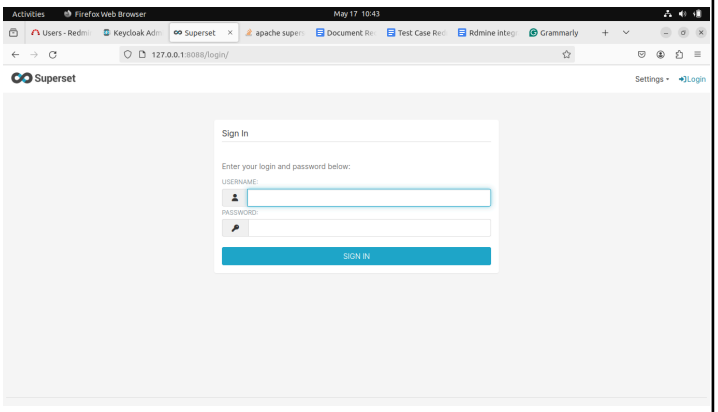
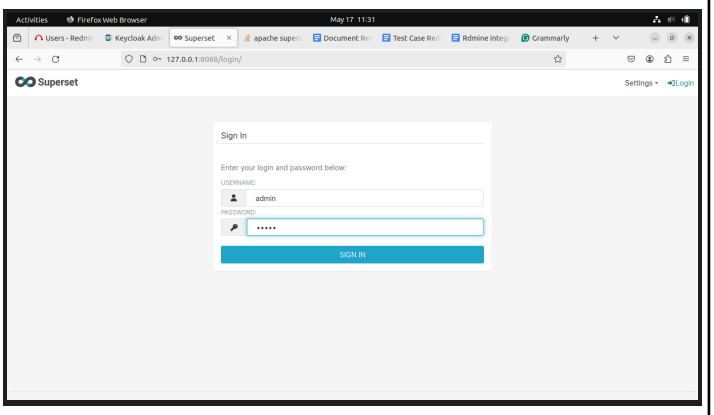
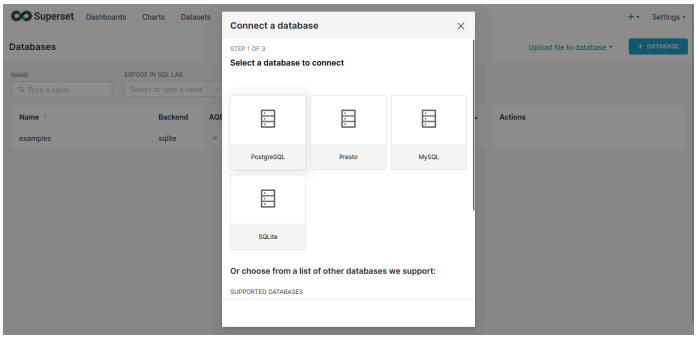
openssl rand -base64 42

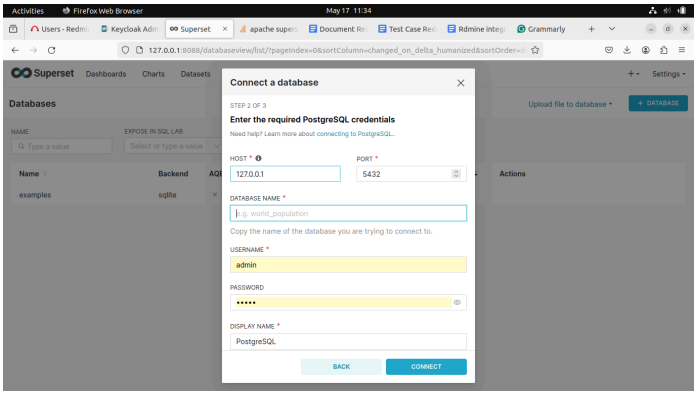
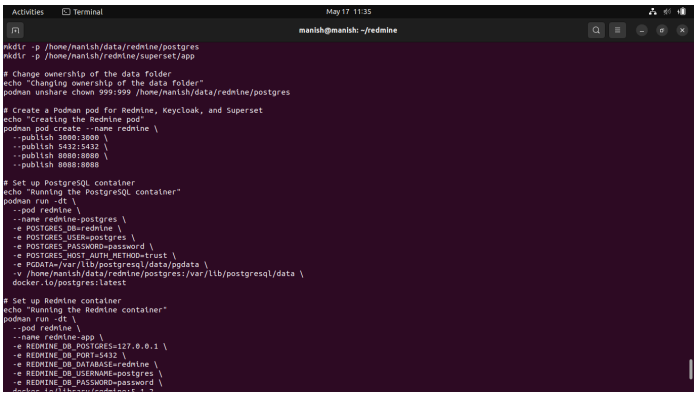
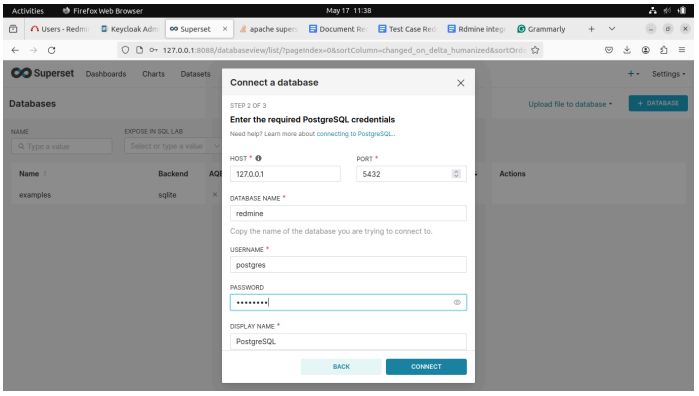
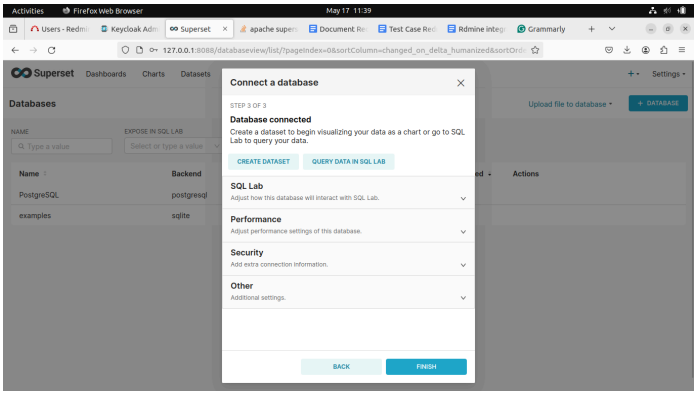
“N/m/iGhhcsoadUnvIlHMqHiRbApy7Z1CT9ymQXwbnBSQl50c7xNaH10+”

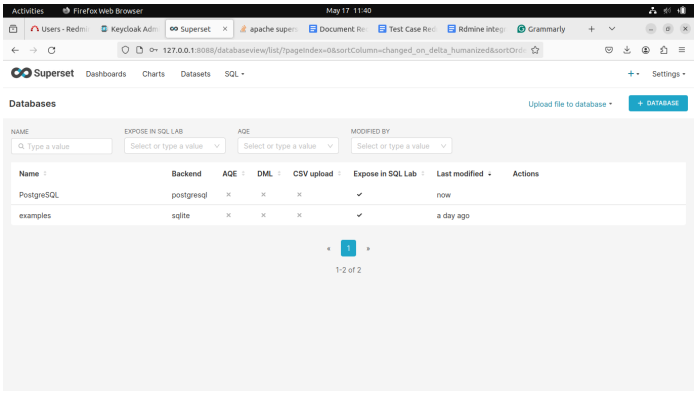
```
manish@manish:~/redmine/superset/app$ podman images
REPOSITORY                  TAG         IMAGE ID      CREATED       SIZE
localhost/superset          manish      e8693c41fde8  13 days ago   1.38 GB
quay.io/keycloak/keycloak   latest      07caf7ea07ac  2 weeks ago   462 MB
docker.io/apache/superset   latest      bf71819e68c5  5 weeks ago   1.37 GB
docker.io/library/postgres  latest      cff6b68a194a  5 weeks ago   439 MB
docker.io/library/redmine   5.1.2       375ba4478aa8  2 months ago  653 MB
k8s.gcr.io/pause            3.5         ed210e3e4a5b  3 years ago   690 kB
manish@manish:~/redmine/superset/app$
```

When: User navigate to "http://127.0.0.1:8088" in a web browser	
Then: The Superset login page displayed	
Result: Passed	

Task 22:

Scenario: Create Superset admin user and initialise the database	
Given: The Superset container is running	**Output:** 
When: User run the command "podman exec -it superset superset fab create-admin --username admin --firstname Superset --lastname Admin --email admin@superset.com --password admin"	
Then: Then an admin user with the username "admin" and password "admin"created	
When: User run the command "podman exec -it superset superset db upgrade"	
Then: The local database migrated to the latest version	

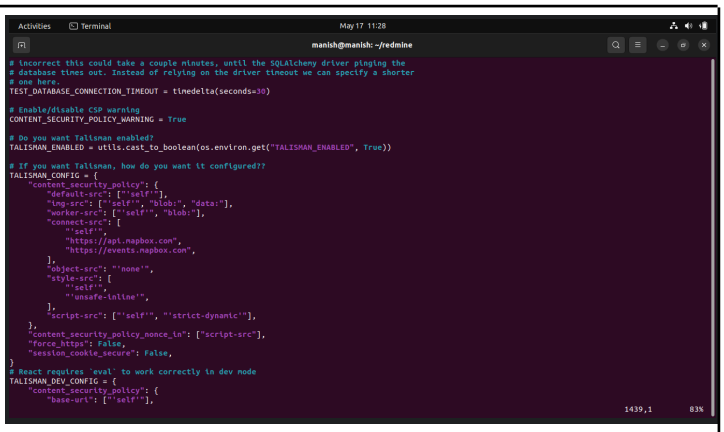
When: User run the command "podman exec -it superset superset init"	
Then: The Superset instance initialised	
When: User log in to Superset with the username "admin" and password "admin"	
Then: User gain access to the Superset dashboard	

Result: Passed	

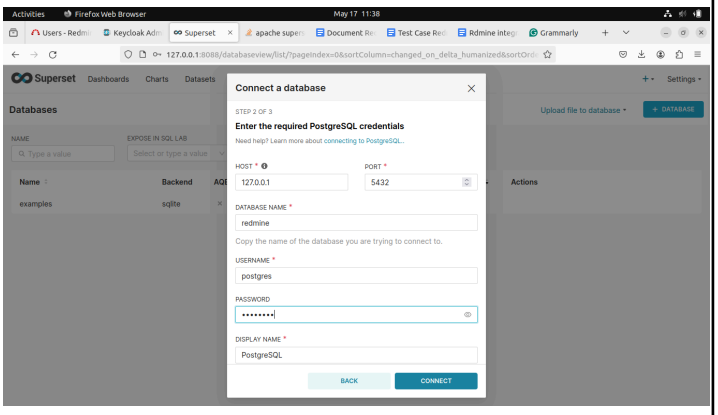
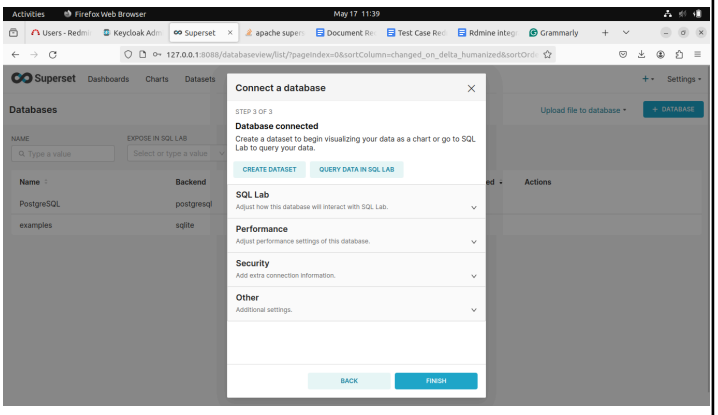
Task: 23

Scenario: Update Superset configuration	
Given: User need to disable the CSP in Superset	**Output:**
When: User run the command "podman exec -it -u root superset bash"	
And: User run "apt-get update" and "apt install vim" inside the container	
And: User navigate to the "superset/" directory and edit "config.py" using "vim"	
And: User set "TALISMAN_ENABLED" to "False" in "config.py"	

```
manish@manish:~/redmine/superset/app$ podman exec -it superset superset db upgrade
Loaded your LOCAL configuration at [/app/pythonpath/superset_config.py]
logging was configured successfully
2024-05-17 06:19:23,351:INFO:superset.utils.logging_configurator:logging was configured
successfully
2024-05-17 06:19:23,355:INFO:root:Configured event logger of type <class
'superset.utils.log.DBEventLogger'>
/usr/local/lib/python3.10/site-packages/flask_limiter/extension.py:293: UserWarning: Using
the in-memory storage for tracking rate limits as no storage was explicitly specified. This
is not recommended for production use. See:
https://flask-limiter.readthedocs.io#configuring-a-storage-backend for documentation about
configuring the storage backend.
  warnings.warn(
WARNI [alembic.env] SQLite Database support for metadata databases will         be removed
in a future version of Superset.
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.

manish@manish:~/redmine/superset/app$ podman exec -it superset superset db upgrade
Loaded your LOCAL configuration at [/app/pythonpath/superset_config.py]
logging was configured successfully
2024-05-17 06:21:00,555:INFO:superset.utils.logging_configurator:logging was configured
successfully
2024-05-17 06:21:00,560:INFO:root:Configured event logger of type <class
'superset.utils.log.DBEventLogger'>
/usr/local/lib/python3.10/site-packages/flask_limiter/extension.py:293: UserWarning: Using
the in-memory storage for tracking rate limits as no storage was explicitly specified. This
is not recommended for production use. See:
https://flask-limiter.readthedocs.io#configuring-a-storage-backend for documentation about
configuring the storage backend.
  warnings.warn(
WARNI [alembic.env] SQLite Database support for metadata databases will         be removed
```

And: User save the changes and exit	
And: User restart the Superset container with "podman restart superset"	
Then: Then the CSP should be disabled	
When: User navigate to "http://127.0.0.1:8088" in a web browser	
Then: The Superset login page should be displayed	
Result: Passed	

Task: 24

Scenario: Connect Superset to PostgreSQL	
Given: PostgreSQL is running and configured with the details from "redmine.sh"	**Output:** 
When: User log in to Superset and navigate to the "Settings" section	
And: User add a new database connection	
And: User select "Postgres" as the database type	
And: User fill in the connection details (hostname, port, database name, username, and password)	
Then: The PostgreSQL database connected successfully	
Result: Passed	

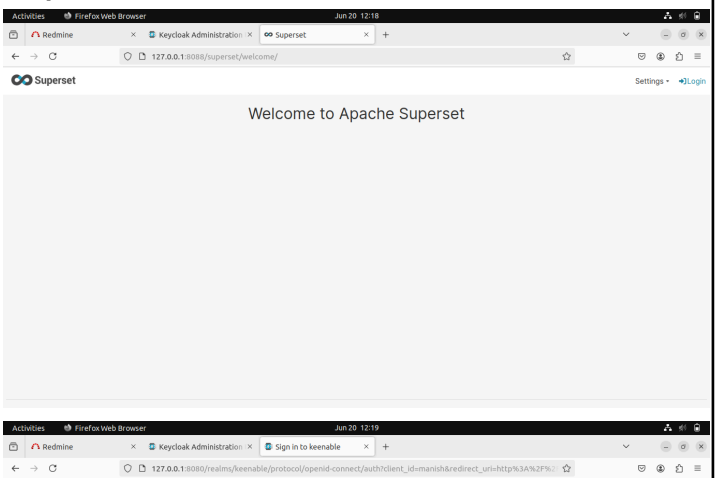
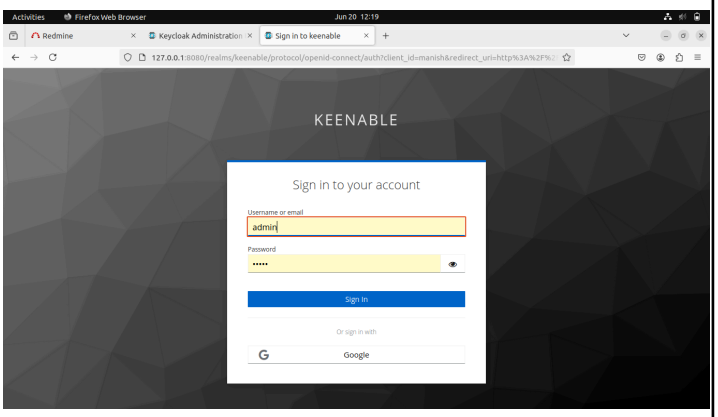
Task: 25

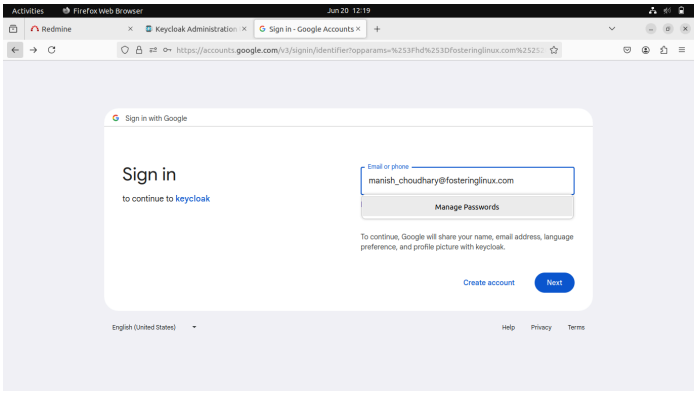
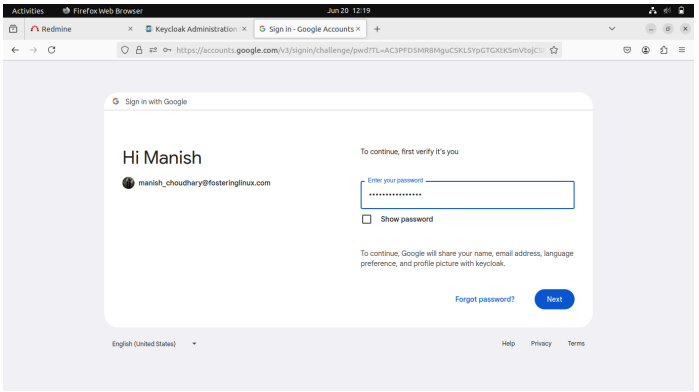
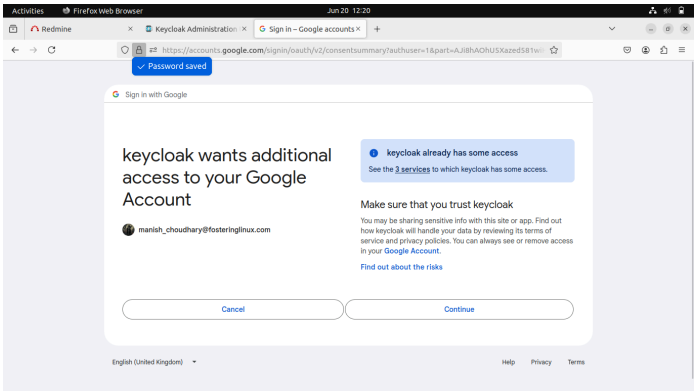
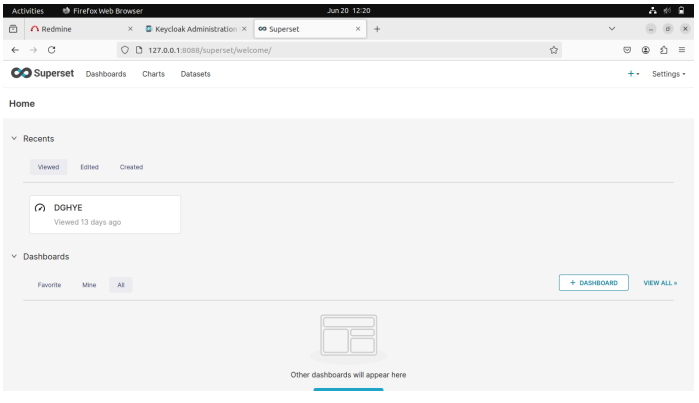
Scenario: Configure Superset for Keycloak SSO

Given: User have created the "client_secret.json" file with Keycloak client details	**Output:**
And: User have created the "keycloak_security_manager.py" file with custom security manager code	
And: User have created the "superset_config.py" file with Keycloak configuration	
When: User navigate to the Superset application directory	
And: User update the database with "podman exec -it superset superset db upgrade"	
And: User initialise the Superset instance with "podman exec -it superset superset init"	
And: User restart the Superset container with "podman restart superset"	
Then: Superset configured to use Keycloak for authentication	

Result: Passed	

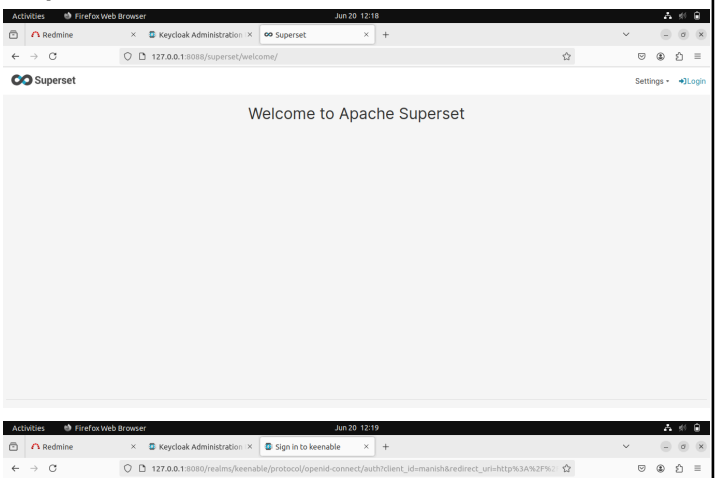
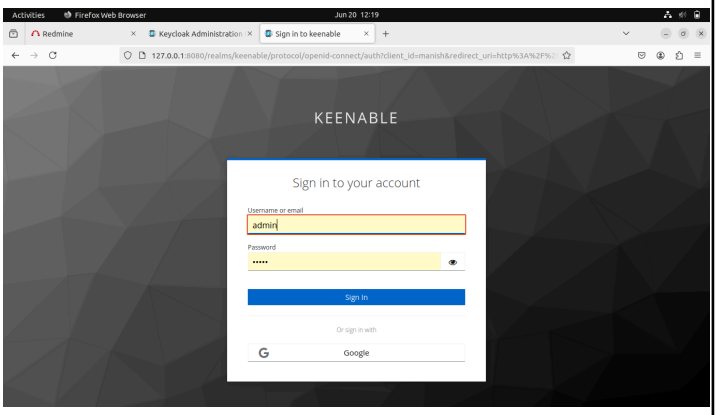
Task: 26

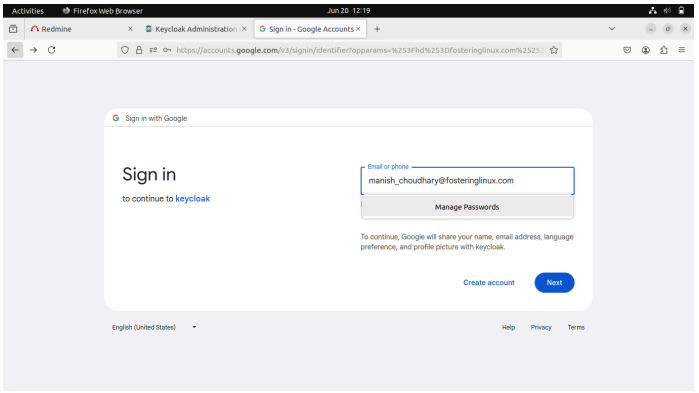
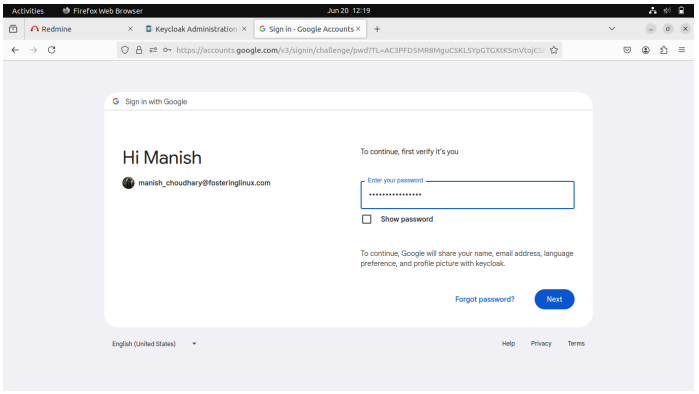
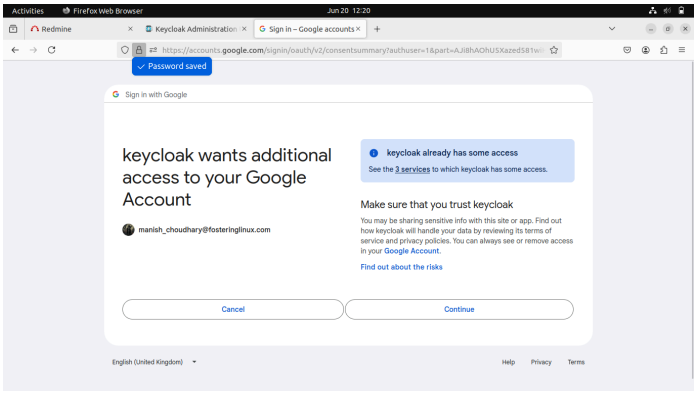
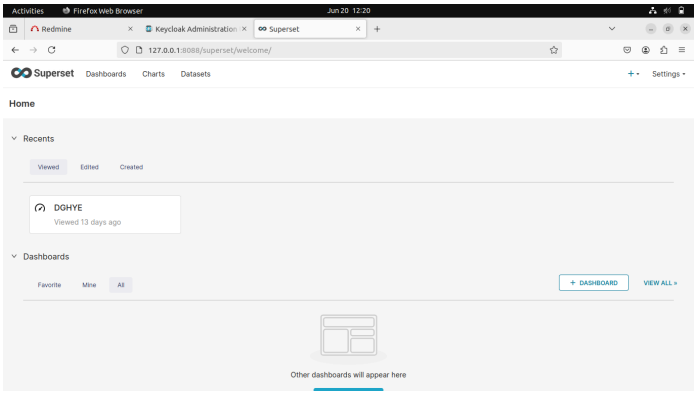
Scenario: Login to Superset via Keycloak	
Given: The Superset container is running and configured for Keycloak	**Output:**  
When: User navigate to "http://127.0.0.1:8088" in a web browser	
Then: User redirected to the Keycloak sign-in page	

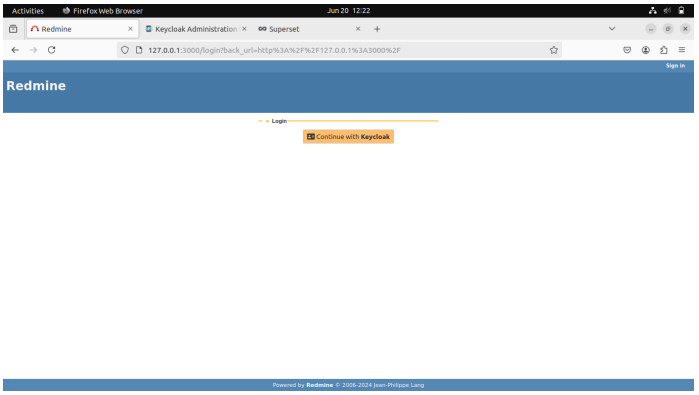
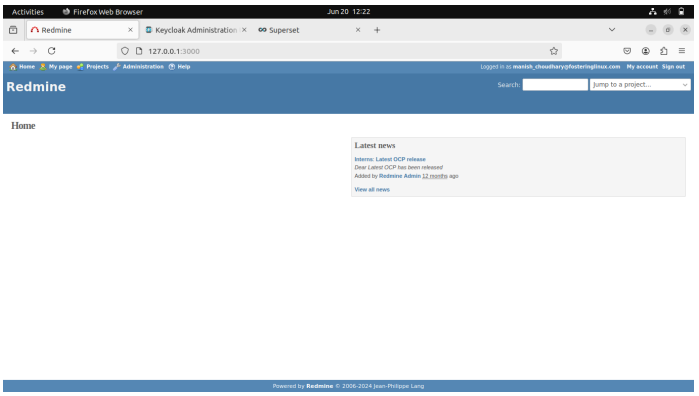
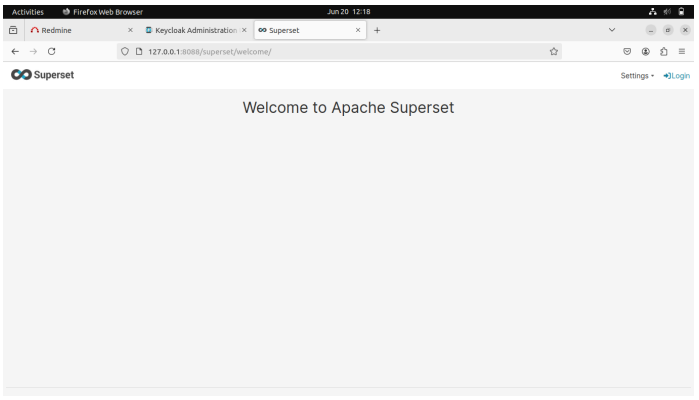
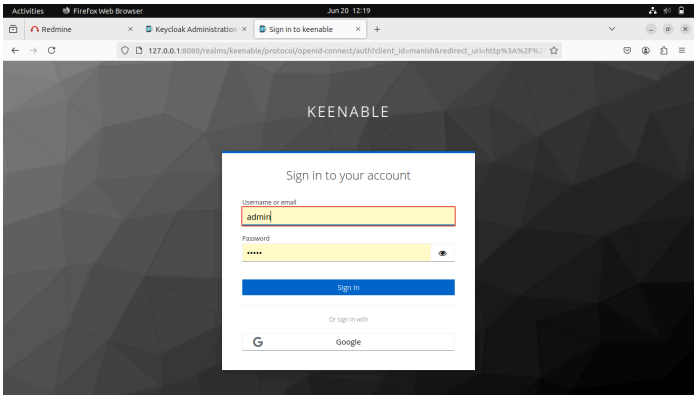
When: User click on the Google login option	
And: User authenticate using my Google account	
Then: User redirected back to Superset and be logged in	 

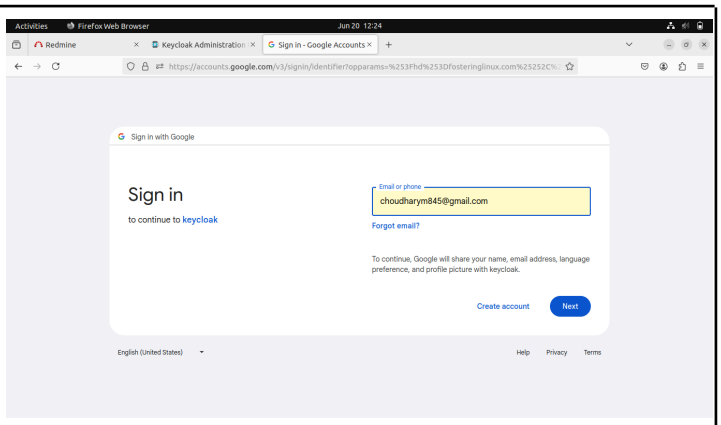
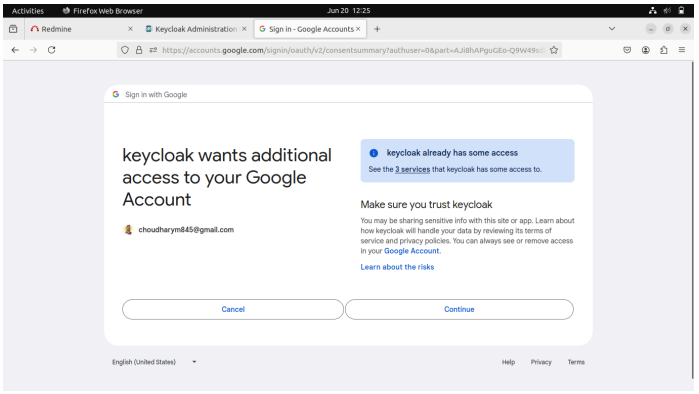
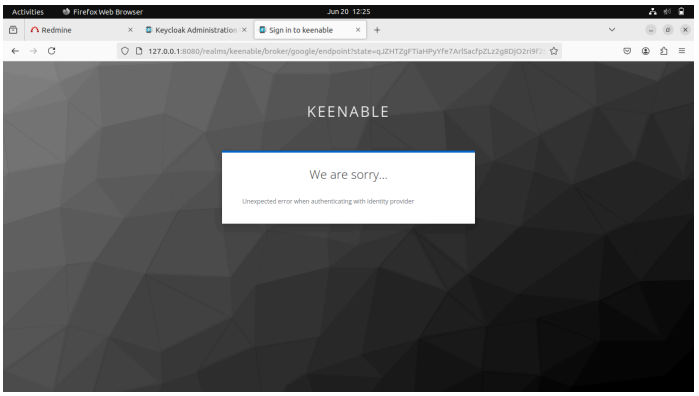
Result: Passed	

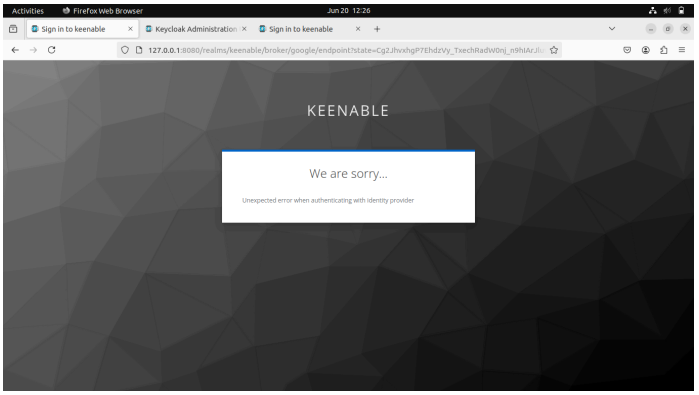
Task: 27

Scenario: Test SSO Integration with Redmine and Superset	
Given: Both Redmine and Superset are configured to use Keycloak for SSO	**Output:**  
When: I log in to Redmine with Keycloak and Google authentication	
Then: User logged in successfully if the user exists in Redmine	

When: User log in to Superset with Keycloak and Google authentication	
Then: User logged in successfully if the user exists in Superset	
When: User try to log in to Redmine with an email not registered in Redmine	 

	 
Then: User see an authentication error	
When: User try to log in to Superset with an email not registered in Superset	
Then: User see an authentication error	

	
Then: User logged in successfully	
When: User try to log in to Superset with the email "manish_choudhary@fosteringlinux.com"	
Then: User logged in successfully	
When: User try to log in to Redmine with the email "choudharym845@gmail.com"	**Redmine:**

	
Then: User see an authentication error	
When: User try to log in to Superset with the email "choudharym845@gmail.com"	
Then: User see an authentication error	
Result: Passed	