```
manish@manish:~/redmine$ cat redmine.sh
#!/bin/bash

# Create folders for PostgreSQL data and Superset app
echo "Creating folders for PostgreSQL data and Superset app"
mkdir -p /home/manish/data/redmine/postgres
mkdir -p /home/manish/redmine/superset/app

# Change ownership of the data folder
echo "Changing ownership of the data folder"
podman unshare chown 999:999 /home/manish/data/redmine/postgres

# Create a Podman pod for Redmine, Keycloak, and Superset
echo "Creating the Redmine pod"
podman pod create --name redmine \
  --publish 3000:3000 \
  --publish 5432:5432 \
  --publish 8080:8080 \
  --publish 8088:8088

# Set up PostgreSQL container
echo "Running the PostgreSQL container"
podman run -dt \
  --pod redmine \
  --name redmine-postgres \
  -e POSTGRES_DB=redmine \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=password \
  -e POSTGRES_HOST_AUTH_METHOD=trust \
  -e PGDATA=/var/lib/postgresql/data/pgdata \
  -v /home/manish/data/redmine/postgres:/var/lib/postgresql/data \
  docker.io/postgres:latest

# Set up Redmine container
echo "Running the Redmine container"
podman run -dt \
  --pod redmine \
  --name redmine-app \
  -e REDMINE_DB_POSTGRES=127.0.0.1 \
  -e REDMINE_DB_PORT=5432 \
  -e REDMINE_DB_DATABASE=redmine \
  -e REDMINE_DB_USERNAME=postgres \
  -e REDMINE_DB_PASSWORD=password \
  docker.io/library/redmine:5.0.5
```

```
# Set up Keycloak container
echo "Running the Keycloak container"
podman run -dt \
  --pod redmine \
  --name keycloak \
  -e KEYCLOAK_ADMIN=admin \
  -e KEYCLOAK_ADMIN_PASSWORD=admin \
  quay.io/keycloak/keycloak:latest \
  start-dev

# Set up Superset container
echo "Running the Superset container"
podman run -dt \
  --pod redmine \
  --name superset \
  -v /home/manish/redmine/superset/app:/app/pythonpath \
  -e
"SUPERSET_SECRET_KEY="rWYzY7dyZARl//h3jI1iCVf978lWPZZNVIXQtBrMOWlUToNGFK4
DcRLL \
  superset:manish

echo "All containers are up and running."


manish@manish:~/redmine$ cd superset/app/
manish@manish:~/redmine/superset/app$ cat superset_config.py
import os
from keycloak_security_manager import OIDCSecurityManager
from flask_appbuilder.security.manager import AUTH_OID
import logging

#---------------------------KEYCLOAK ----------------------------

curr = os.path.abspath(os.getcwd())
AUTH_TYPE = AUTH_OID
SECRET_KEY = 'xkAh+zcYLmlRugMkBRyW3VJThAyYF8r0U1VP2dHPXbAiSstVYFDg0P8/'
OIDC_CLIENT_SECRETS = os.path.join(curr, 'pythonpath', 'client_secret.json')
OIDC_ID_TOKEN_COOKIE_SECURE = False
OIDC_REQUIRE_VERIFIED_EMAIL = False
OIDC_OPENID_REALM = 'fosteringlinux'
OIDC_INTROSPECTION_AUTH_METHOD = 'client_secret_post'
CUSTOM_SECURITY_MANAGER = OIDCSecurityManager
AUTH_USER_REGISTRATION = True
```

```
AUTH_USER_REGISTRATION_ROLE = 'Public'

#---------------------------------------------------------------

SUPERSET_WEBSERVER_DOMAINS = ["http://fosteringlinux.com", "http://keenable.in"]
ENABLE_PROXY_FIX = True
SUPERSET_WEBSERVER_PROTOCOL = "http"
SUPERSET_WEBSERVER_PORT = 8088
PUBLIC_ROLE_LIKE = "Gamma"

# Additional Superset configurations
# ...

# Setting the logging level to DEBUG for detailed output
logging.basicConfig(level=logging.DEBUG)

manish@manish:~/redmine/superset/app$ cat client_secret.json
{
        "web": {
        "issuer": "http://127.0.0.1:8080/realms/fosteringlinux/broker/google/endpoint",
        "auth_uri": "http://127.0.0.1:8080/realms/fosteringlinux/protocol/openid-connect/auth",
        "client_id": "manishch",
        "client_secret": "Oywvz2J89HbtJ3o2Cw05onDVyUWkwWyh",
        "redirect_uris": [
        "http://127.0.0.1:8088/"

        ],
        "userinfo_uri":
"http://127.0.0.1:8080/realms/fosteringlinux/protocol/openid-connect/userinfo",
        "token_uri": "http://127.0.0.1:8080/realms/fosteringlinux/protocol/openid-connect/token",
        "token_introspection_uri":
"http://127.0.0.1:8080/realms/fosteringlinux/protocol/openid-connect/token/introspect"
        }
}

manish@manish:~/redmine/superset/app$ cat keycloak_security_manager.py
from flask_appbuilder.security.manager import AUTH_OID
from superset.security import SupersetSecurityManager
from flask_oidc import OpenIDConnect
from flask_appbuilder.security.views import AuthOIDView
from flask_login import login_user
from urllib.parse import quote
from flask_appbuilder.views import expose
from flask import redirect, request
```

```
import logging

class AuthOIDCView(AuthOIDView):
    @expose('/login/', methods=['GET', 'POST'])
    def login(self, flag=True):
        sm = self.appbuilder.sm
        oidc = sm.oid

        @self.appbuilder.sm.oid.require_login
        def handle_login():
            user = sm.auth_user_oid(oidc.user_getfield('email'))

            if user is None:
                info = oidc.user_getinfo(['preferred_username', 'given_name', 'family_name', 'email'])
                user = sm.add_user(
                    info.get('preferred_username'),
                    info.get('given_name'),
                    info.get('family_name'),
                    info.get('email'),
                    sm.find_role('Gamma')
                )

            login_user(user, remember=False)
            return redirect(self.appbuilder.get_url_for_index)

        return handle_login()

    @expose('/logout/', methods=['GET', 'POST'])
    def logout(self):
        oidc = self.appbuilder.sm.oid
        logging.debug("OIDC logout initiated")

        # Ensure 'id_token' is available in credentials_store
        id_token = oidc.credentials_store.get('id_token')

        if id_token:
            # Perform the local logout
            oidc.logout()
            super(AuthOIDCView, self).logout()

            # Construct the post_logout_redirect_uri
            redirect_url = "http://127.0.0.1:8088/"
            logging.debug(f"Redirect URL after logout: {redirect_url}")
            # Construct the full Keycloak logout URL
            keycloak_logout_url = oidc.client_secrets.get('issuer') + '/protocol/openid-connect/logout'
            full_logout_url =
f"{keycloak_logout_url}?id_token_hint={id_token}&post_logout_redirect_uri={quote(redirect_url)}
"
            logging.debug(f"Full logout URL: {full_logout_url}")

            return redirect(full_logout_url)
        else:
            logging.warning("No 'id_token' found in credentials_store during logout.")
            # Perform local logout without id_token handling
            oidc.logout()
            super(AuthOIDCView, self).logout()
            return redirect("http://127.0.0.1:8088/")

class OIDCSecurityManager(SupersetSecurityManager):
    authoidview = AuthOIDCView

    def __init__(self, appbuilder):
        super(OIDCSecurityManager, self).__init__(appbuilder)
        if self.auth_type == AUTH_OID:
            self.oid = OpenIDConnect(self.appbuilder.get_app)
```