Execute PODBA using Ansible Automation Platform 2 (AAP2)

Overview: Ansible Automation Platform Version 2 offers new features and components. This readme outlines the process of executing the PODBA playbooks using AAP2 GUI.

Reference to AAP2: https://www.ansible.com/blog/introducing-ansible-automation-platform-2

AAP2 installation instructions can be found here:

Chapter 3. Installing Red Hat Ansible Automation Platform Red Hat Ansible Automation Platform 2.3 | Red Hat Customer Portal

Prepare the Execution Environment: This is a one-time setup before executing playbooks from AAP2 GUI.

- Login to the Ansible controller with “awx” user.
- Install python version 3.8 or later [# dnf install python3]
- Install podman using dnf [# dnf install podman]
- Install ansible-builder [$ pip install ansible-builder]
- Install ansible-navigator [$ pip install ansible-navigator], this is not mandatory for GUI.
- In any present working directory, create a directory named "context".
- Download and extract Oracle Instant client software from Oracle site: https://www.oracle.com/database/technologies/instant-client/downloads.html
- Note: For Linux on Power, click on “other platforms” in the above URL.
- Inside the "context" directory place the extracted oracle client software directory with the name "oracle_client".
- Create a file (example: create_podba.yml) with the following content.

```
version: 3

images:
  base_image:
    name: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8:latest
options:
  package_manager_path: /usr/bin/microdnf
additional_build_steps:
  append_base:
   - RUN microdnf install gcc python39-devel libnsl* libaio* find* which* sudo dnf
   - RUN pip3 install wheel
   - RUN python3.9 -m pip install cx_Oracle --upgrade
   - RUN ln -s /usr/lib64/libnsl.so.2 /usr/lib64/libnsl.so.1
   - COPY oracle_client /oracle_client_sw
   - COPY ansible-automation-platform-managed-ca-cert.crt /etc/pki/ca-trust/source/anchors
dependencies:
  galaxy: requirements.yml
```

- Make the requirements.yml file

```
collections:
  - ibm.power_aix
```

- Run the following command to build the execution environment image.

```
$ ansible-builder build -t powerodba -f create_podba.yml Running
command:
  podman build -f context/Containerfile -t powerodba context
Complete! The build context can be found at: /var/lib/awx/aap2/context
```

- List the images:

```
$ podman images
REPOSITORY                                                  TAG         IMAGE ID      CREATED            SIZE localhost/powerodba
latest      e04948d6013a  About a minute ago  908 MB
registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8  latest      b2d26de2d8de  4 months ago
1.79  GB  registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8       latest          c239714e9480   4
months ago        380 MB
```

Execute playbooks from CLI using ansible-navigator:

1. Create a file called ansible-navigator.yml inside the {{ collection_name }}/playbooks directory with the following content.

```
$ cat ansible-navigator.yml
---
ansible-navigator:       execution-
environment:        enabled: True
  image: powerodba:latest        # Name of the REPOSITORY:TAG
```

2. Follow the readme files under "docs" to understand how to update the required variables for each task.
3. Run the following command to execute the playbooks. The following example shows execution of manage-db-directories.yml playbook.

```
ansible-navigator run <playbook name> --pp=missing --m stdout -i <name of inventory file>
Example:
$ ansible-navigator run manage-db-directories.yml --pp=missing --m stdout -i hosts.yml
```

To use escalated privileges, please use "--playbook-artifact-enable false" at the end of the command.

Example:

ansible-navigator run db-opatch.yml --pp=missing --m stdout -i hosts.yml --ask-become-pass --playbookartifact-enable false

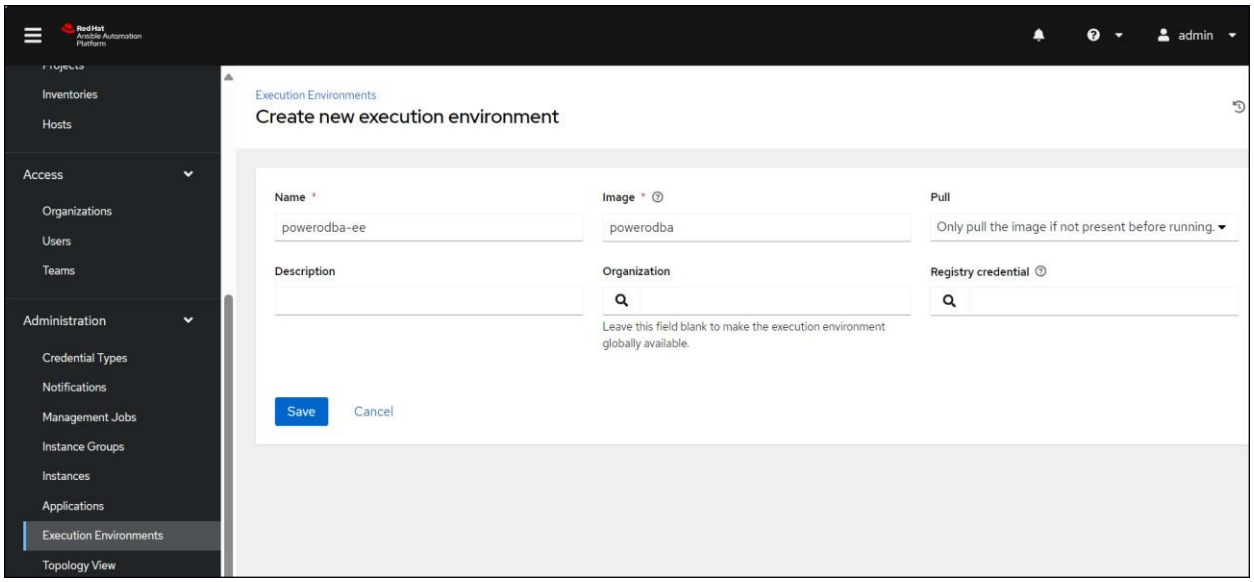
Execute playbooks from GUI:

PODBA has two types of modules – few of them uses “cx_Oracle” module which is a python connector to access Oracle databases and the others won’t use cx_Oracle, they require “ssh” connectivity.

The playbook template setup will be a little different between the two types of modules. Hence, we’re going to setup a project in AAP2 and demonstrate one example playbook which uses cx_Oracle and another one which doesn’t require cx_Oracle.

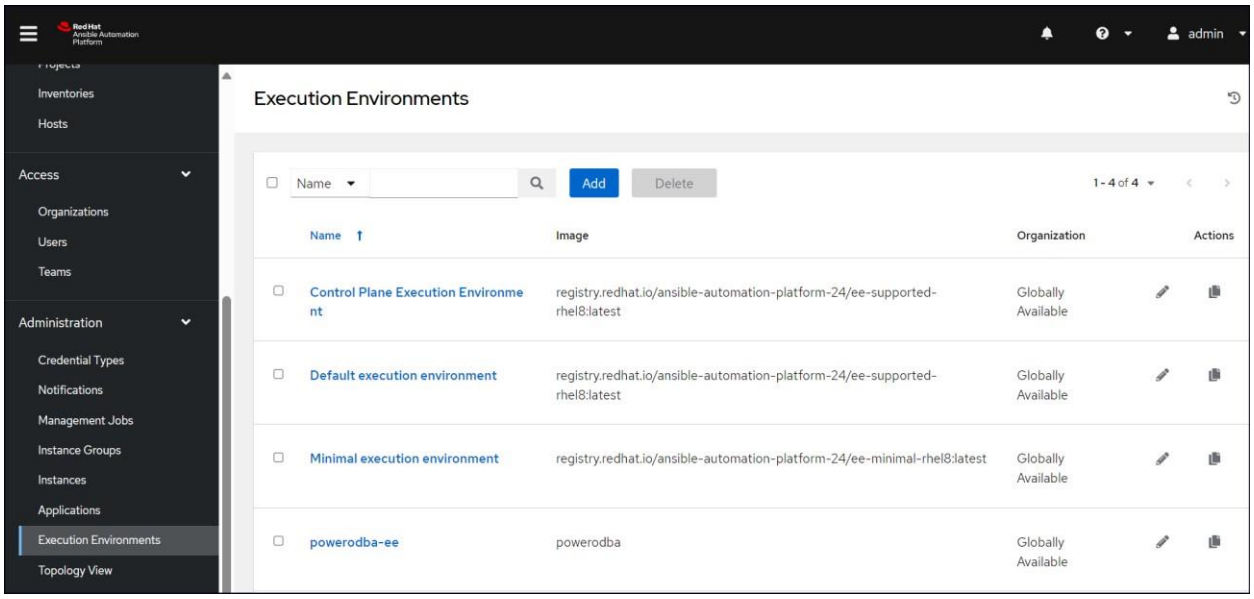
Create Project:

Step 1: Login to AAP2 and “Create new execution environment”.

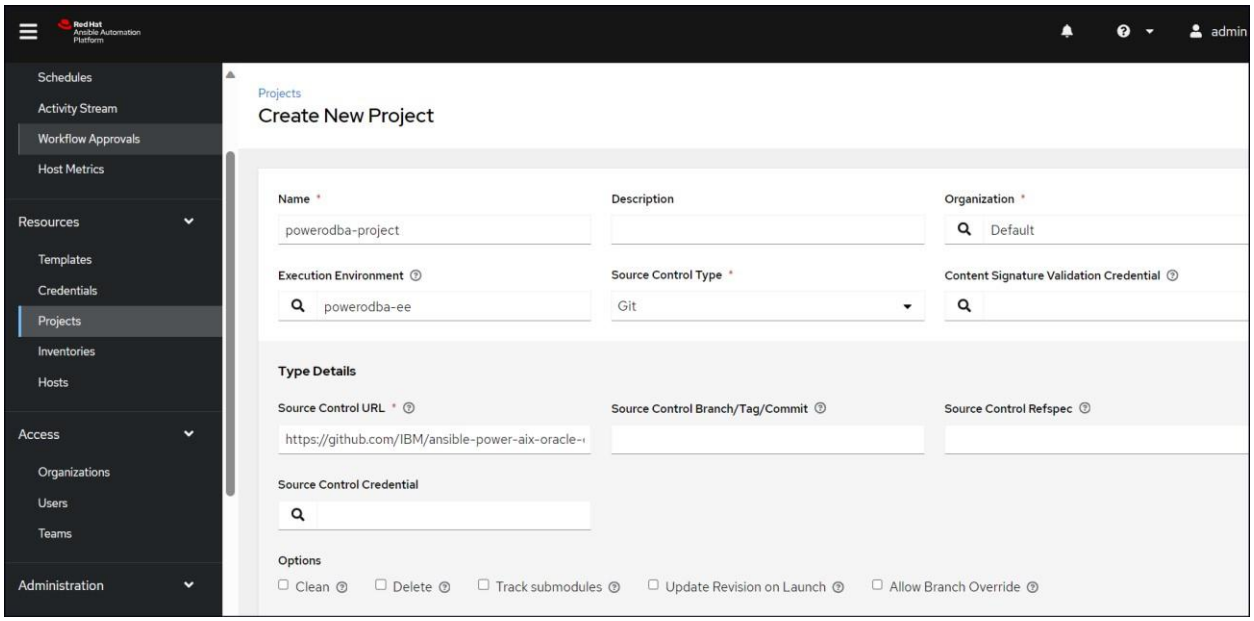


Name: Any desired name for identification.

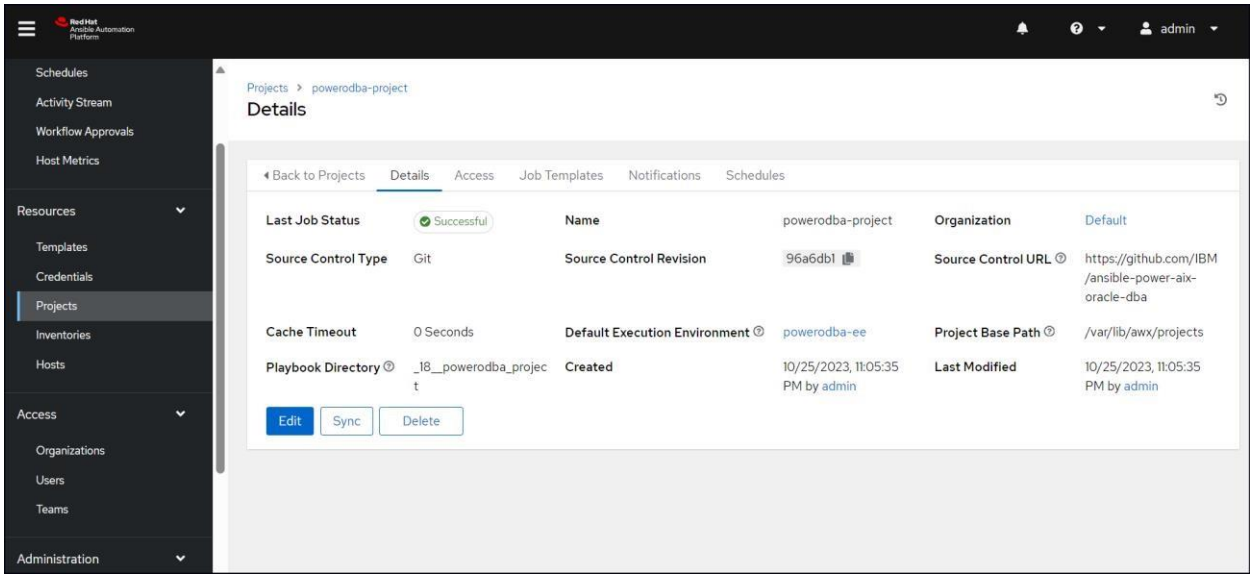
Image: Provide the name of the image which was created earlier in this document using “ansible-builder”.



Step 2: Create a new project.



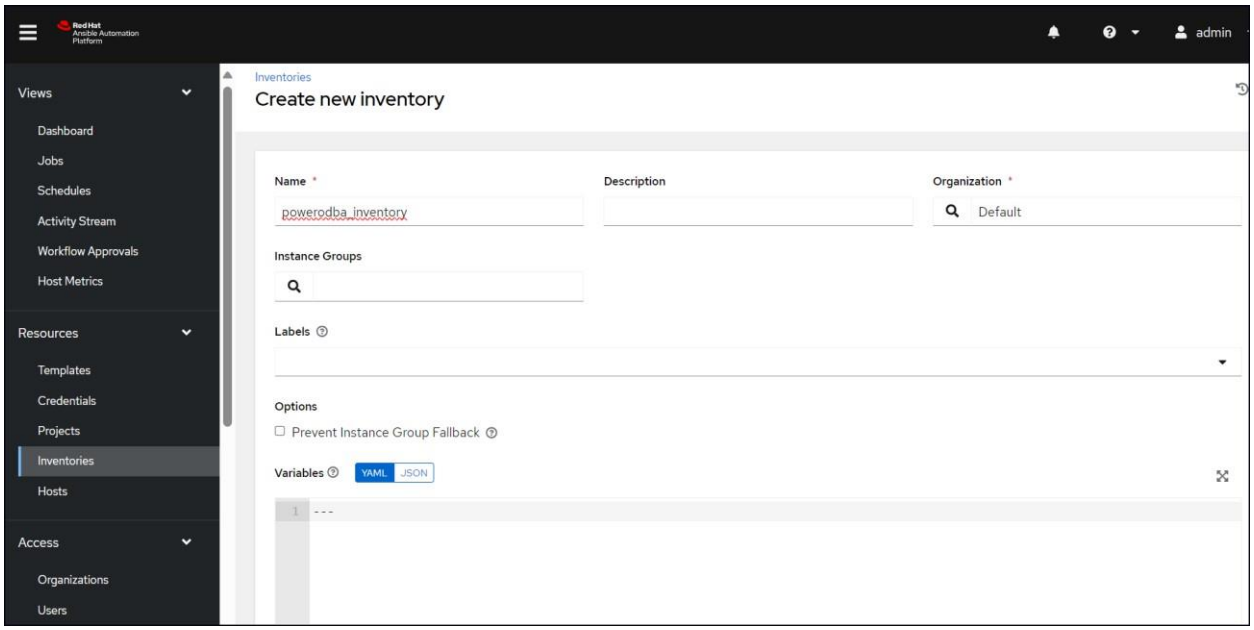
Name: Any desired name for identification.
Organization: Your existing Organization name or leave it “Default”
Execution Environment: Name of the Execution Environment created in Step 1.
Source Control Type: Git
Source Control URL: https://github.com/IBM/ansible-power-aix-oracle-dba
Click “save”
The “Last Job Status” must show Successful as shown below.



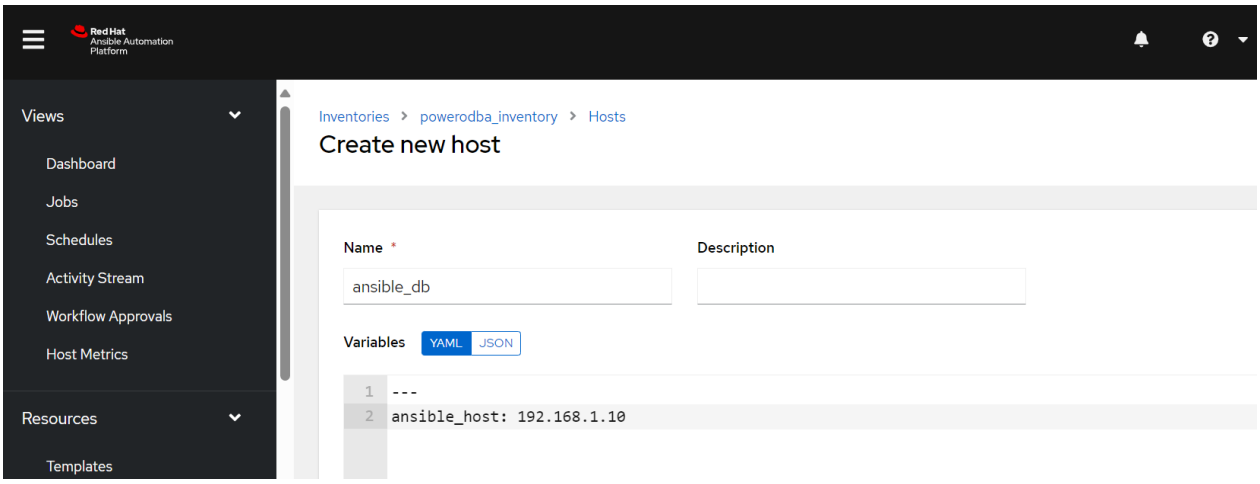
From the above fig, the collection is installed in “/var/lib/awx/projects”

Example 1: The following setup is for the playbooks which use cx_Oracle. The “manage-users.yml” playbook will create two users testuser1 and testuser2 in a Non-CDB environment “atsdb”.

Step 1: Create an inventory with a desired name and save it.



Step 2: Add the hostname where the database is running in the inventory and save it.

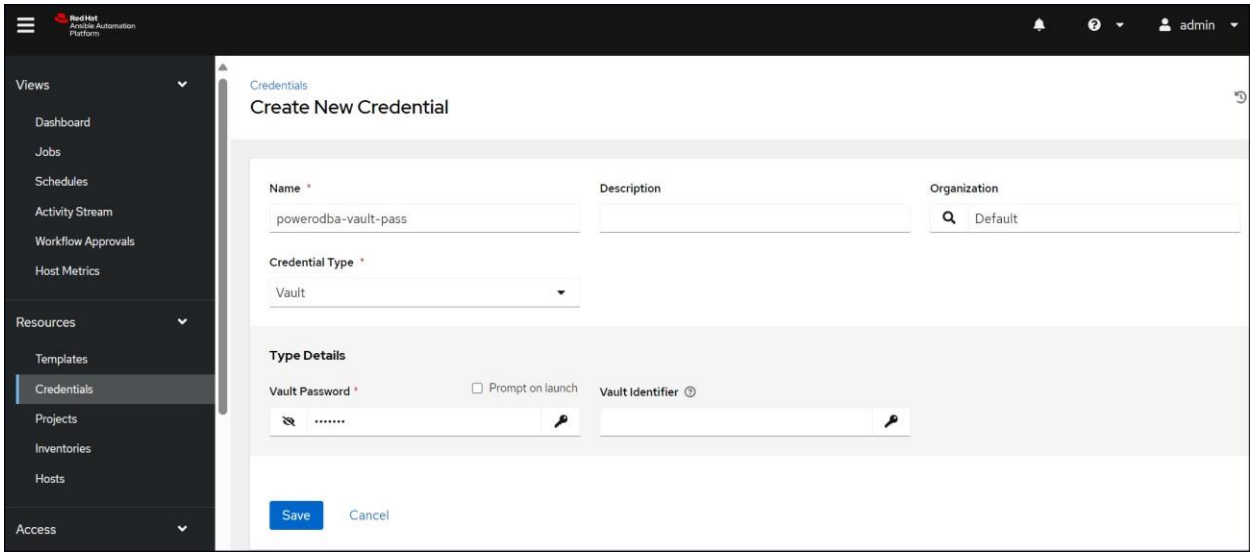


Step 3: Set SYS user password in the vault.yml file for “default_dbpass” variable and encrypt it.

[awx@localhost vars]$ pwd
/var/lib/awx/projects/_18__powerodba_project/playbooks/vars
[awx@localhost vars]$ cat vault.yml default_gipass:
Oracle4u # ASM sys user password
default_dbpass: Oracle4u # Sys user password

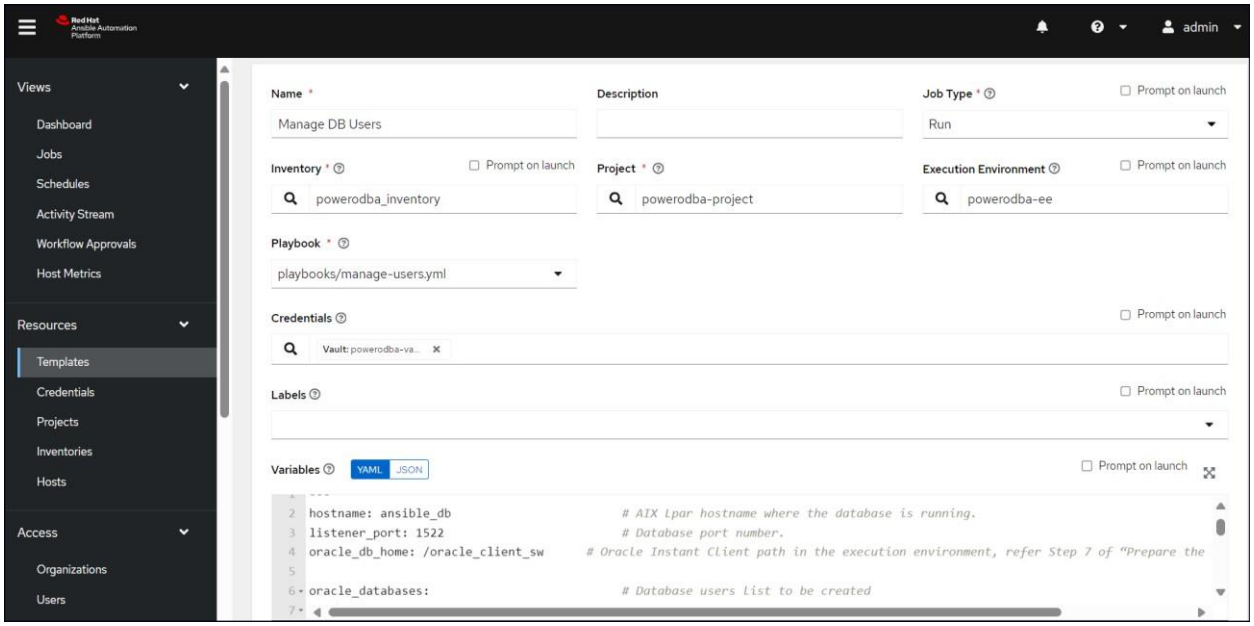
[awx@localhost vars]$ ansible-vault encrypt vault.yml
New Vault password:
Confirm New Vault password:
Encryption successful

Step 4: Set the vault password in “Create New Credentials” tab.



Name: Any desired name for identification.
Credential Type: vault
Vault Password: Update the password used in Step 5.

Step 5: Create a template for manage-users.yml playbook, this will create database users and grants privileges to it.



Name: Manage DB Users

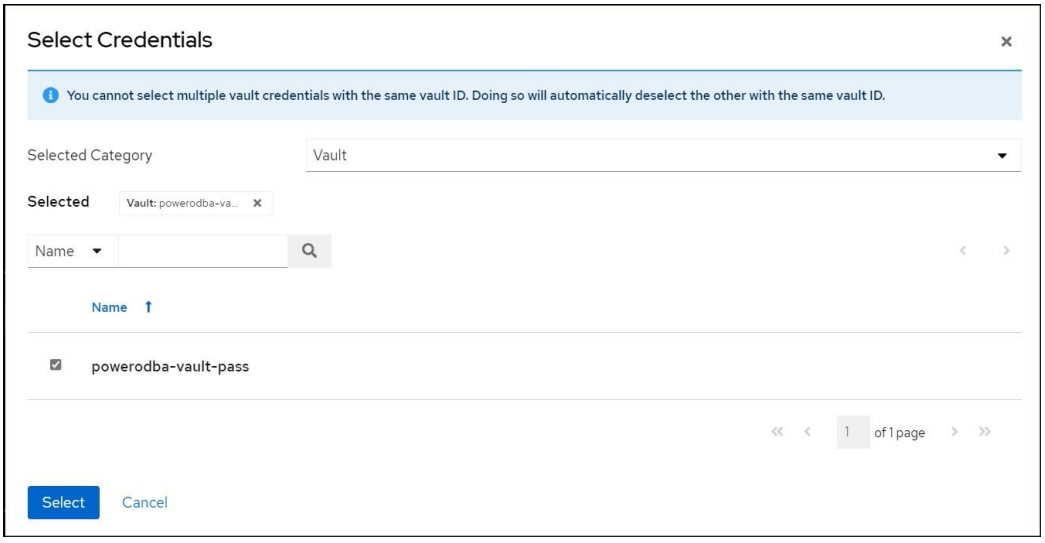
Inventory: Select the inventory created in Step 3 of “Create Project” section.

Project: Select the project created in Step 2 for “Create Project” section.

Execution Environment: Select the Execution environment used in Step 1

Playbook: From the drop down, select manage-users.yml

Credentials: Select the vault credentials created earlier.



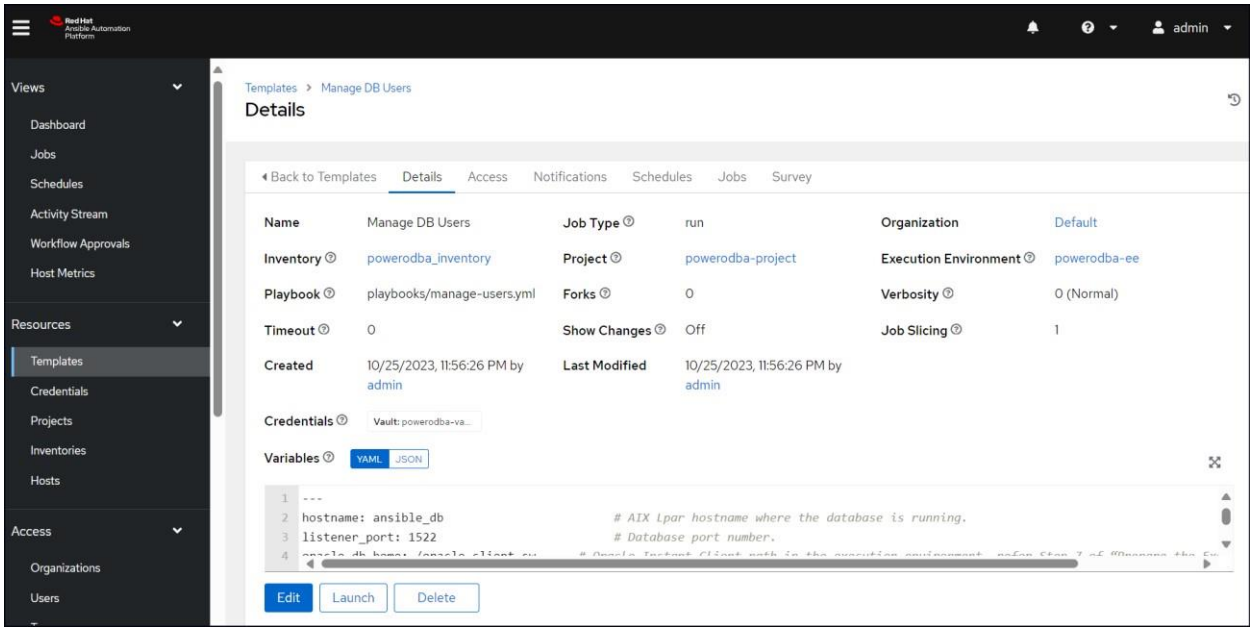
Variables:

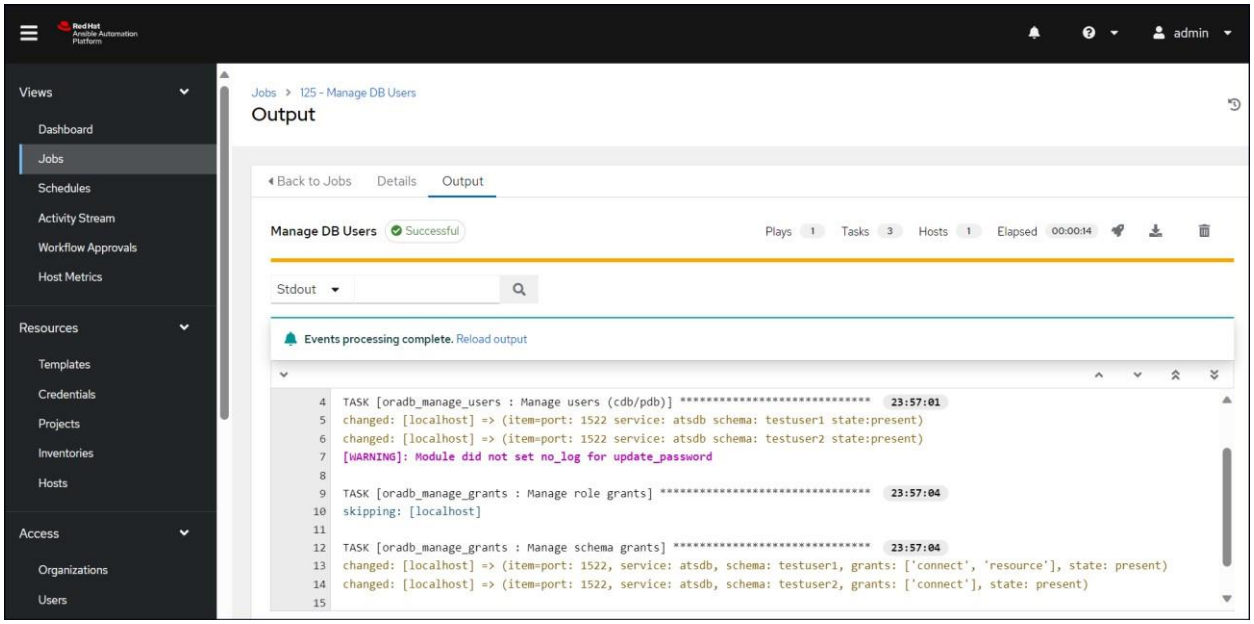
```
hostname: ansible_db                        # AIX Lpar hostname where the database is running.
listener_port: 1521                         # Database port number.
oracle_db_home: /oracle_client_sw           # Oracle Instant Client path on the ansible controller / execution environment.

oracle_databases:                           # Database users list to be created
      - users:
         - schema: testuser1               # Username to be created.
        default_tablespace: users           # Default tablespace to be assigned to the user.
        service_name: devpdb                 # Database service name.
        schema_password: oracle3            # Password for the user.
        grants_mode: enforce                # enforce|append.
        grants:
         - connect                          # Provide name of the privilege as a list to grant to the user.
         - resource
        state: present                       # present|absent|locked|unlocked [present: Creates user, absent: Drops user]
# Multiple users can be created with different attributes as shown below.
      - users:
         - schema: testuser2
        default_tablespace: users
        service_name: devpdb
        grants_mode: enforce
        grants:
         - connect
        schema_password: oracle4
        state: present
```

Save the template.

Step 6: Click Launch





We can see two users testuser1 and testuser2 have been created and granted privileges.