Adding secrets to Ansible Secrets

This document outlines the procedure for adding new secrets to the credential store using the `add-secret.sh` utility. This script automates the encryption process, ensuring consistency and security.

Adding Passwords

Here is the complete, step-by-step process for adding a new password—for example, for a secret named `mssql_db`.

Step 1: Run the add-secret.sh Utility

The add-secret.sh script handles the creation and GPG encryption of the new secret file. It must be run from the Ansible project directory.

```
# Ensure you are in the project root directory
cd /opt/ansible_secrets

# Run the script, providing the name of the secret you
want to add.
# The script will prompt you to enter the password
securely.

./add-secret.sh mssql_db
```

After the script runs successfully, a new encrypted file named `mssql_db_pswd.txt.gpg` will be created in the `/opt/ansible_secrets/files/` directory with the correct ownership.

Step 2: Update the Ansible Playbook

Edit your playbook, `/opt/ansible_secrets/deploy_secrets.yml`, to add the new filename to the list of files to be deployed. Find this section in `deploy_secrets.yml`:

```
vars:
  secrets_target_dir: "/opt/credential_store"
  service_user: "service_account"
  secret_access_group: "appsecretaccess"
  encrypted_secret_files:
    - green_dm_pswd.txt.gpg
    - ldap_ro_pswd.txt.gpg
    - yellow_dm_pswd.txt.gpg
```

Add the new filename to the encrypted_secret_files list:vars:

```
  secrets_target_dir: "/opt/credential_store"
  service_user: "service_account"
  secret_access_group: "appsecretaccess"
  encrypted_secret_files:
    - green_dm_pswd.txt.gpg
    - ldap_ro_pswd.txt.gpg
    - yellow_dm_pswd.txt.gpg
    - mssql_db_pswd.txt.gpg  # <-- Add the new file here
```

Save the playbook file.

Step 3: Re-run the Ansible Playbook

Now, deploy the change.

```
# Ensure you are in the project root and your venv is
active
cd /opt/ansible_secrets
source venv/bin/activate

# Run the playbook
ansible-playbook deploy_secrets.yml
```

Re-running the Playbook

Because Ansible is idempotent, it will be very efficient. It will check the state of the existing files and find that they are already correct (they will show up as "ok"). It will only perform the action for the new file, copying `mssql_db_pswd.txt.gpg` to `/opt/credential_store` and setting its permissions (this will show up as "changed").

Step 4: Use the New Secret in Your Scripts

Your reusable helper scripts are designed to use the new secret without any changes to them.

- In a Bash script:

```
#!/bin/bash
# Get the MSSQL password into a variable
MSSQL_PASS=$(/usr/local/bin/get_secret.sh mssql_db)
if [[ -z "$MSSQL_PASS" ]]; then
```

```
    echo "Failed to retrieve MSSQL password." >&2
    exit 1
fi
echo "Successfully retrieved password. Now connecting
to database..."
# ... use "$MSSQL_PASS"
unset MSSQL_PASS # Clean up
```

- In a Python script: This example shows the necessary code to reliably locate and import the `secret_retriever` module before using it.

```
#!/usr/bin/env python3
import sys
import os

# --- Start of required modification ---
# Define the path to the helper library.
HELPER_LIB_PATH =
"/usr/local/lib/ansible_secret_helpers"

# Add the path to the system path list so Python can
find the module.
# This makes the script work reliably from anywhere
(including cron).
if HELPER_LIB_PATH not in sys.path:
    sys.path.append(HELPER_LIB_PATH)

try:
    # Now that the path is set, the import will
succeed.
  import secret_retriever
```

```
except ImportError:
    print(f"CRITICAL ERROR: Could not import
'secret_retriever'.", file=sys.stderr)
    print(f"Ensure '{HELPER_LIB_PATH}' exists and is
readable.", file=sys.stderr)
    sys.exit(1)
# --- End of required modification ---

mssql_pass = None
try:
    # Use the helper to get the mssql_db password
    mssql_pass =
secret_retriever.get_password("mssql_db")
    print(f"Successfully retrieved MSSQL DB password of
length {len(mssql_pass)}")
    # ... now use the mssql_pass variable to connect to
the database

except Exception as e:
    print(f"Error retrieving secret: {e}",
file=sys.stderr)

finally:
    # Clear the variable reference for security
    mssql_pass = None
```