Step 1: Creating an Encrypted Secrets File

I started by creating a vault-encrypted YAML file where I stored the database password:

```
ansible-vault create secrets.yml
```

Inside the file, I added:

```
db_password: "SuperSecurePass123"
```

This file is now encrypted and cannot be opened without the vault password.

Step 2: Using the Encrypted Secret in My Playbook

Then, in my main playbook, I included this encrypted file using vars_files:

```
---
- name: Deploying Web App with Secure DB Access
  hosts: localhost
  vars_files:
    - secrets.yml

  tasks:
    - name: (Just for test) Print the password
      debug:
        msg: "The password is {{ db_password }}"

    - name: Connect to MySQL using the secret password
      shell: |
        mysql -u root -p{{ db_password }} -e "SHOW DATABASES;"
      register: result

    - name: Show database list
      debug:
        var: result.stdout_lines
```

Note: In a real-world project, I don't actually print passwords. The debug task above was just to test that the variable worked properly.

Step 3: Running the Playbook Securely

To run the playbook, I used:

```
ansible-playbook deploy-app.yml --ask-vault-pass
```

Or, to avoid entering the password every time, I also tested it with:

```
ansible-playbook deploy-app.yml --vault-password-file ~/.vault_pass.txt
```

Extra Thing I Tried

I also learned that I could encrypt just a single value (like an API key or password) instead of an entire file:

```
ansible-vault encrypt_string 'SuperSecurePass123' --name 'db_password'
```

It gave me an encrypted block like this:

```
db_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          383063346334656239623836623564346535393135633833623032343734353 8...
```

I then placed it directly in my playbook variables.