# Handling Passwords with Azure Key Vault

## Introduction

Azure Key Vault is a cloud service for securely storing and accessing secrets. This document describes how to add secrets to an existing Azure Key Vault and retrieve them in an Azure DevOps pipeline.

## 1. Adding Secrets to Azure Key Vault

1. Navigate to the Key Vault:  
- Log in to the Azure portal.  
- Go to All services and search for Key Vaults.  
- Select your existing Key Vault.

2. Add a New Secret:  
- In the Key Vault, navigate to the Secrets section.  
- Click on Generate/Import.  
- Enter the Name for the secret (e.g., network-device-password).  
- Enter the Value for the secret (e.g., the password you want to store).  
- Click Create.

## 2. Configuring Azure DevOps Pipeline to Use Key Vault Secrets

1. Create or Edit Your Pipeline:  
- In Azure DevOps, go to your project.  
- Navigate to the Pipelines section.  
- Create a new pipeline or edit an existing one.

2. Add Azure Key Vault Task:  
- In your pipeline YAML file, add a task to retrieve secrets from Azure Key Vault. Ensure your pipeline has access to the Key Vault through a service connection.  
- Example YAML snippet:

```yaml  
stages:  
- stage: Deploy  
 jobs:  
 - job: DeployJob  
 steps:  
 - task: AzureKeyVault@2  
 inputs:  
 azureSubscription: '<Your Service Connection Name>'  
 KeyVaultName: '<Your Key Vault Name>'  
 SecretsFilter: '\*'  
 RunAsPreJob: true  
```

3. Use Retrieved Secrets in Your Pipeline:  
- Secrets retrieved from Azure Key Vault are available as environment variables in the pipeline.  
- Example: If the secret name is network-device-password, it will be available as $(network-device-password).

4. Pass Secrets to Ansible Playbook:  
- You can pass the retrieved secrets as environment variables to your Ansible playbook.  
- Example Ansible playbook usage:

```yaml  
- hosts: all  
 vars:  
 ansible\_password: "{{ lookup('env', 'NETWORK\_DEVICE\_PASSWORD') }}"  
 tasks:  
 - name: Ensure the device is reachable  
 ping:  
```

## 3. Example Full Pipeline YAML

```yaml  
trigger:  
- main  
  
pool:  
 vmImage: 'ubuntu-latest'  
  
variables:  
 # Add other variables if needed  
  
stages:  
- stage: Deploy  
 jobs:  
 - job: DeployJob  
 steps:  
 - task: AzureKeyVault@2  
 inputs:  
 azureSubscription: '<Your Service Connection Name>'  
 KeyVaultName: '<Your Key Vault Name>'  
 SecretsFilter: '\*'  
 RunAsPreJob: true  
  
 - script: |  
 echo "Running Ansible Playbook"  
 ansible-playbook -i inventory.yml playbook.yml  
 env:  
 NETWORK\_DEVICE\_PASSWORD: $(network-device-password)  
```

## Conclusion

By integrating Azure Key Vault with your Azure DevOps pipeline, you ensure that sensitive information like passwords is securely managed and accessed. This approach improves the security and reliability of your automation processes.