**Exercise - Run an Ansible playbook from Azure Pipelines**

In this part, you run your Ansible playbook in Azure Pipelines. You configure the pipeline to run Ansible from your control machine, on the same Linux VMs that you set up earlier.

To do so, you:

* Create a repository on GitHub, and then push your local Git repository to GitHub.
* Create the project in Azure DevOps.
* Create a service connection that enables Azure Pipelines to access your Azure subscription.
* Install the Ansible extension for Azure Pipelines.
* Create the pipeline in Azure Pipelines, and add a task that runs Ansible from your control machine.
* Remove a user from one of your VMs, and then manually run the pipeline to see Ansible repair the configuration.

In previous modules, we provide a template that sets up a project in Azure DevOps for you. Here, you set up the project manually because you already have most of the pieces you need to run Ansible in Azure Pipelines.

The Ansible extension for Azure Pipelines provides the Ansible@0 task, which simplifies the process of connecting to your control machine and running your Ansible playbook.

**Create your repository on GitHub**

1. In a web browser, go to GitHub  and sign in.
2. From the **Repositories** pane, select **New**.
3. Under **Repository name**, enter *mslearn-ansible*.
4. Select **Create repository**.

**Push your local Git repository to GitHub**

Push your local Git repository to GitHub. To do so, you add a remote that points to your GitHub repository.

1. In Visual Studio Code, go to your Bash session.
2. Add and commit your inventory file, *azure\_rm.yml*, and your playbook, *users.yml*, to your repository.

**Bash**

**git add azure\_rm.yml users.yml**

**git commit -m "Initial"**

1. Add a remote named *origin* that points to your GitHub repository. Replace <you> with your GitHub username.

**Bash**

**git remote add origin https://github.com/<you>/mslearn-ansible.git**

1. Push your master branch to GitHub.

**Bash**

**git push -u origin master**

1. Refresh your repository on GitHub.

You see *azure\_rm.yml* and *users.yml* listed under your master branch.

**Create the project in Azure DevOps**

1. Go to Azure DevOps  and sign in.
2. Select your Azure DevOps organization.
3. Select **New project**.
4. Name your project *Microsoft Learn - Ansible*, and then select **Create**.

**Create a service connection**

Create a service connection that enables Azure Pipelines to access your Azure subscription. Azure Pipelines uses this service connection to securely connect to your control machine.

1. In Azure DevOps, go to your **Microsoft Learn - Ansible** project.
2. Select **Project settings** from the bottom corner of the page.
3. Under **Pipelines**, select **Service connections**.
4. Select **Create service connection**, then choose **SSH**, then choose **Next**.

The **New SSH service connection** dialog box appears.

1. In the dialog box, fill in these fields:

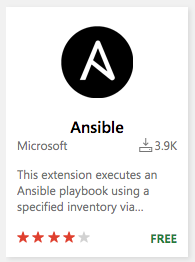
|  |  |
| --- | --- |
|  | |
| Field | **Value** |
| Host name | Your control machine's IP address, for example, *40.113.7.63* |
| Port number | *22* |
| Private key | The contents of your SSH private key, *ansible\_rsa* |
| User name | *azureuser* |
| Password/Passphrase | (Leave blank) |
| Connection name | *SSH - Ansible VM* |

1. Ensure that **Grant access permission to all pipelines** is selected.
2. Select **OK**.

**Install the Ansible extension for Azure Pipelines**

Here, you install an extension that provides access to the Ansible@0 task in Azure Pipelines.

1. From a new browser tab, go to marketplace.visualstudio.com .
2. On the **Azure DevOps** tab, search for "Ansible."
3. Select the **Ansible** extension.



1. Select **Get it free**.
2. Select your Azure DevOps organization.
3. Select **Install**.

**Create the pipeline in Azure Pipelines**

During this process, you create your pipeline configuration file, *azure-pipelines.yml*.

1. In Azure DevOps, go to your project.
2. Select **Pipelines**, either from the project page or from the left pane.
3. Select **Create Pipeline**.
4. On the **Connect** tab, select **GitHub**.

If prompted, enter your GitHub credentials.

1. On the **Select** tab, select your **mslearn-ansible** repository.

You might be redirected to GitHub to install the Azure Pipelines app. If so, select **Approve and install**.

1. On the **Configure** tab, select **Starter pipeline**.
2. On the **Review** tab, note the starter code for your pipeline configuration. Move your cursor to the bottom of the file.
3. On the **Review** tab, select **Show assistant**.
4. Select **Ansible** from the list of tasks shown. Then fill in these fields:
   * **Ansible location**: **Remote machine**
   * **Ansible SSH endpoint**: **SSH - Ansible VM**
   * **Playbook**:
     + **Source**: **Agent machine**
     + **Root**: *$(System.DefaultWorkingDirectory)*
     + **File path**: *users.yml*
   * **Inventory**:
     + **Inventory location**: **File**
     + **File source**: **Agent machine**
     + **File path**: *azure\_rm.yml*
   * **Advanced**:
     + **Additional parameters**: *--private-key ~/.ssh/ansible\_rsa*
5. Select **Add**.
6. As an optional step, remove the automatically generated comments and the tasks you don't need.

Here's what the most basic configuration looks like:

**yml**

trigger:

- master

pool:

vmImage: 'ubuntu-latest'

steps:

- task: Ansible@0

inputs:

ansibleInterface: 'remoteMachine'

connectionOverSsh: 'SSH - Ansible VM'

playbookSourceRemoteMachine: 'agentMachine'

playbookRootRemoteMachine: '$(System.DefaultWorkingDirectory)'

playbookPathLinkedArtifactOnRemoteMachine: 'users.yml'

inventoriesRemoteMachine: 'file'

inventoryFileSourceRemoteMachine: 'agentMachine'

inventoryFileLinkedArtifactOnRemoteMachine: 'azure\_rm.yml'

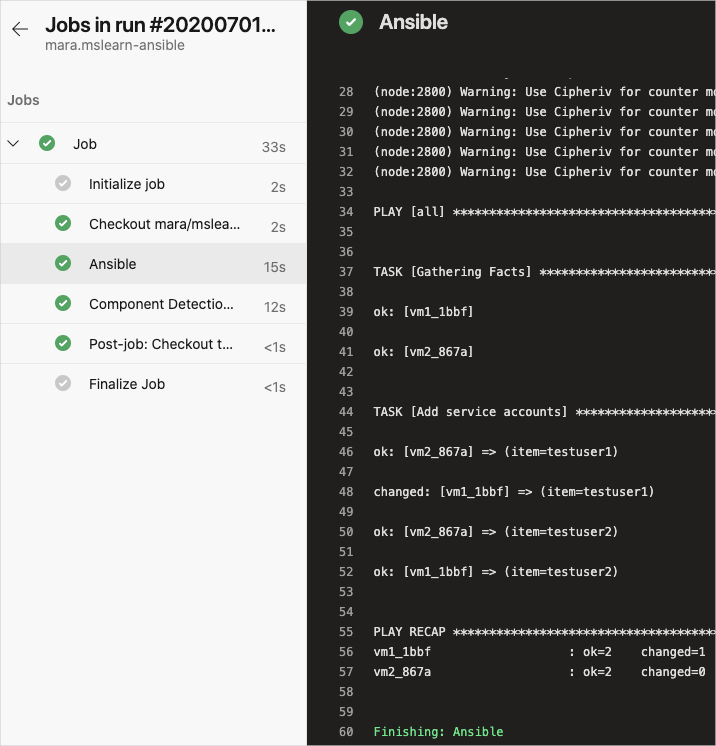
args: '--private-key ~/.ssh/ansible\_rsa'

1. Select **Save and run**. Then select **Save and run** again.

**Watch the pipeline run**

On the build screen, select the job and trace the pipeline as it runs.

You see that the **Ansible** task runs, and produces the same output as before:



**See Ansible repair the configuration**

Here, you remove the *testuser1* user from your VM named *vm1*. You then run the pipeline a second time to see Ansible repair the configuration on the VM.

This process helps you see how Ansible can repair your configuration if it's accidentally changed by an outside person or process.

1. In Visual Studio Code, switch to your Azure Cloud Shell session.
2. Get the IP address of your VM named *vm1*:

**Azure CLI**

IP\_ADDRESS=$(az vm list-ip-addresses \

--resource-group learn-ansible-rg \

--name vm1 \

--query [0].virtualMachine.network.publicIpAddresses[0].ipAddress \

--output tsv)

1. Verify that the user *testuser1* exists on the VM:

**Bash**

ssh -i ~/.ssh/ansible\_rsa azureuser@$IP\_ADDRESS "/usr/bin/getent passwd testuser1"

1. Remove the user *testuser1* from the VM:

**Bash**

ssh -i ~/.ssh/ansible\_rsa azureuser@$IP\_ADDRESS "sudo /usr/sbin/userdel testuser1"

1. Verify that the user *testuser1* no longer exists on the VM:

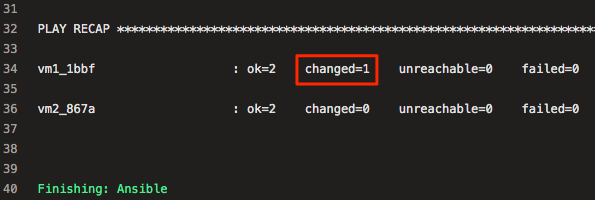
**Bash**

ssh -i ~/.ssh/ansible\_rsa azureuser@$IP\_ADDRESS "/usr/bin/getent passwd testuser1"

You don't see any output, which tells you the user isn't present.

1. Manually run the pipeline. To do so, go to your pipeline in Azure Pipelines, and select **Run pipeline**. Make sure **master** is the selected branch, and then select **Run**.

You see that the user is added again to the VM:



Ansible doesn't make configuration changes to *vm2*, because you removed *testuser1* only from the first VM, *vm1*.

As an optional step, return to Cloud Shell and verify that the user *testuser1* exists on *vm1*.