



Ansible on Windows Fundamentals

Automating Tasks with Ansible Playbooks

Writing a Simple Playbook

Objective

- Review how to write a basic Ansible playbook and store it in a Git repository.

Writing Playbooks

- An Ansible Playbook is the main way to automate tasks in Ansible.
- A *playbook* is a YAML-based text file containing a list of one or more plays to run in a specific order.
- A *play* is an ordered list of *tasks* run against specific hosts within an inventory.
- Each task runs a module that performs some simple action on or for the managed host.
- Most tasks are idempotent and can be safely run a second time without problems.
- Playbooks can change lengthy, complex manual administrative tasks into an easily repeatable routine with predictable and successful outcomes.

Formatting an Ansible Playbook

- A playbook is saved using the standard file extension **.yml**.
- Indentation with space character indicates the structure of the data in the file.
- YAML does not place strict requirements on how many spaces are used for the indentation, but there are two basic rules.
 - Data elements at the same level in the hierarchy (such as items in the same list) must have the same indentation.
 - Items that are children of another item must be indented more than their parents.
- Only the space character can be used for indentation. Tab characters are not allowed.

Formatting an Ansible Playbook

- A playbook begins with a line consisting of three dashes (---) as the start of a document marker.
- It may end with three dots (...) as the end of a document marker, although in practice these are often omitted.
- In between those markers, the playbook is defined as a list of plays.
- A list item in YAML starts with a single dash followed by a space.
- For example, a YAML list might appear as follows:

```
- apple  
- orange  
- grape
```

Formatting an Ansible Playbook

- The play itself is a collection of key-value pairs.
- Keys in the same play should have the same indentation.
- The following example shows a YAML snippet which is a list item (the play) with three keys. The first two keys (**name** and **host**) have simple values. The third key (**tasks**) has a list of three items as a value:

```
- name: just an example
  hosts: webserver
  tasks:
    - first
    - second
    - third
```

Example: A simple playbook with one play, of one task

- ❶ A name for the play to help document its intended purpose.
- ❷ Hosts or groups on which the play will run, taken from the inventory.
- ❸ The beginning of the list of tasks in the play.
- ❹ Clear descriptive names for each task help document what each task does.
- ❺ Each task uses one module to perform the work, in this case **win_service**.
- ❻ Argument for the **win_service** module to specify which service to manage.
- ❼ Another argument for **win_service**. Indented the same amount as the preceding line.

```
---  
- name: Converted ad hoc command example ❶  
  hosts: win1.example.com ❷  
  tasks: ❸  
    - name: Make sure the spooler service is started ❹  
      win_service: ❺  
        name: spooler ❻  
        state: started ❼
```


Example: Playbook excerpt of three tasks from a play

The order in which plays and tasks are listed in a playbook is important, because Ansible runs them in the same order.

```
tasks:
- name: Ensure that WinRM is started when the system has settled
  win_service:
    name: WinRM
    start_mode: delayed

- name: Registry path MyClassroom and its entries are absent
  win_regedit:
    path: HKCU:\Software\MyClassroom
    state: absent
    delete_key: yes

- name: Region format is set to English (United States)
  win_region:
    format: en-US
```

The first task in the list uses the **win_service** module to make sure the WinRM service is configured to start at boot.

Next, the second task uses the **win_regedit** module to ensure a specific Registry key is not present.

Finally, the third task uses the **win_region** module to set the system's region to US English.

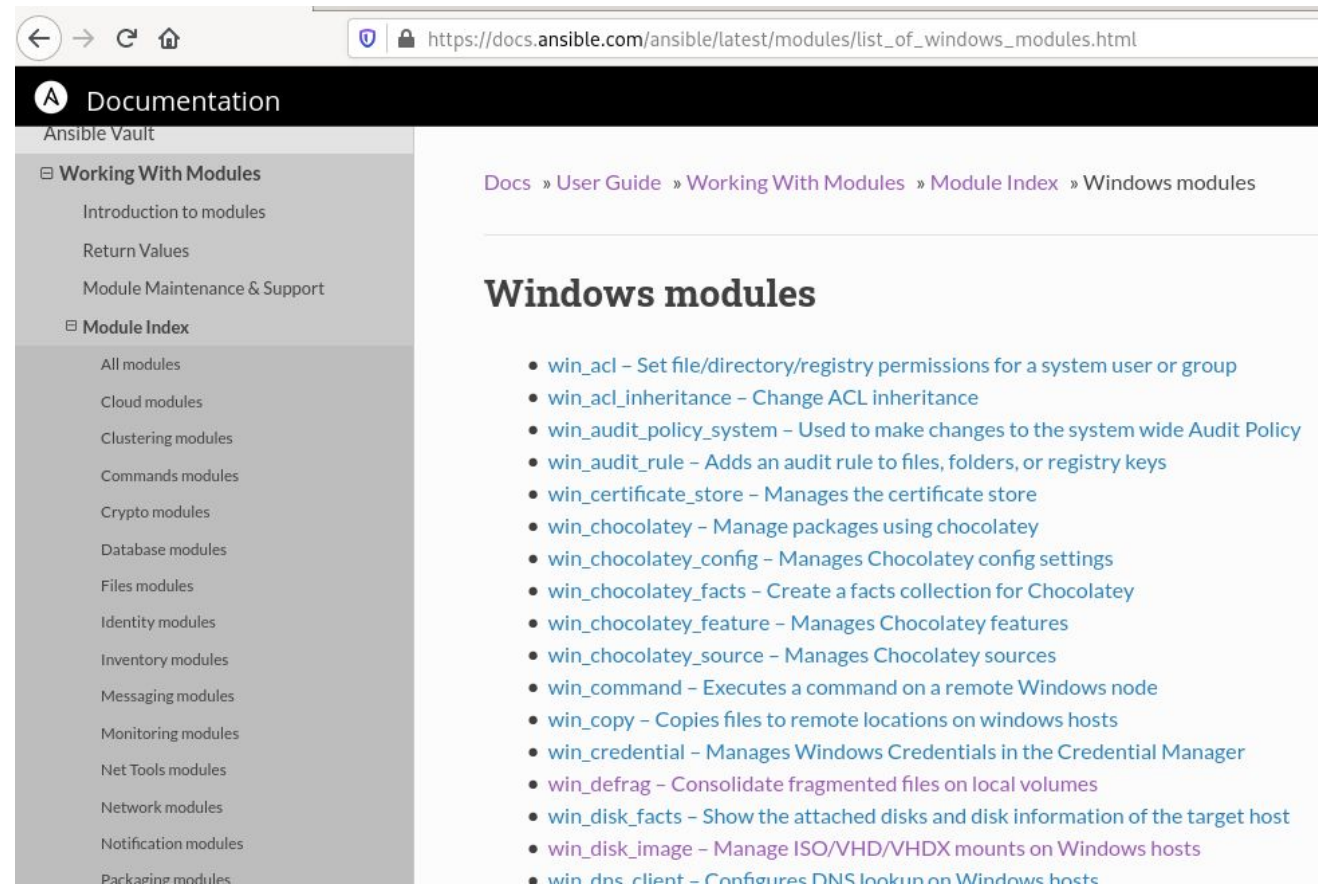
Finding Documentation for Windows modules

https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html

Other modules might be useful in your tasks as well, but these are specifically designed for Windows management.

Each link will go to documentation on and examples of how to use each of the listed modules.

A selected number of these modules will be used in examples in this course.



The screenshot shows a web browser displaying the Ansible documentation page for Windows modules. The browser's address bar shows the URL: https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html. The page has a dark header with the 'Documentation' logo. A left sidebar contains a navigation menu with categories like 'Working With Modules' and 'Module Index'. The main content area shows a breadcrumb trail: 'Docs » User Guide » Working With Modules » Module Index » Windows modules'. Below this, the title 'Windows modules' is followed by a bulleted list of module names and their brief descriptions, such as 'win_acl - Set file/directory/registry permissions for a system user or group' and 'win_certificate_store - Manages the certificate store'.

Remember to Push Your Work!

- When your playbook is complete, save your work
- Remember to commit your new or changed files to your local Git repository
- Remember to push your committed changes to the remote Git repository so others see them
- Once your playbook is in the remote repository, Ansible Tower can retrieve and run it

Running Ansible Playbooks

Objectives

- How to prepare a job template for a playbook in Ansible Tower
- How to run a playbook and evaluate the results in Ansible Tower

Overview: Preparing Playbooks to Run in Ansible Tower

- Have an inventory and machine credential for your managed hosts
- Create a Source Control credential if needed for your Git repository
- Create a Project for your Git repository that
 - Specifies the Git repository URL
 - Uses the Source Control credential
 - Gets the latest revision (or specifies a branch, tag, or commit to get)
- Create a Job Template to run the playbook specifying
 - The inventory and machine credential(s) to use
 - The project and playbook to use
- The Job Template can then be run whenever you want, to run the playbook

Creating a Source Control Credential

- Log in to the web-based UI for Ansible Tower.
- Click **Credentials** to enter the credentials management interface.
- In the CREDENTIAL pane, click + to create a new credential.
- In the CREATE CREDENTIAL pane, enter the required information for the new credential.
 - Enter a unique NAME for the credential.
 - Specify your ORGANIZATION if needed (often “Default”).
 - In the CREDENTIAL TYPE list, select **Source Control**.
 - Under TYPE DETAILS, enter the authentication information needed for your Git repository.
- Click **SAVE**.

The screenshot shows the 'Create Credential' form in Ansible Tower. The form is divided into several sections:

- * NAME**: A text input field containing 'testing-repo'.
- DESCRIPTION**: A text input field containing 'Credential for the testing repository'.
- ORGANIZATION**: A dropdown menu with 'Default' selected.
- * CREDENTIAL TYPE**: A dropdown menu with 'Source Control' selected.
- TYPE DETAILS**: A section containing:
 - USERNAME**: A text input field containing 'testing'.
 - PASSWORD**: A text input field with masked characters (dots) and a toggle icon.
 - SCM PRIVATE KEY**: A large text area with a hint: 'HINT: Drag and drop private file on the field below.' and a search icon.
 - PRIVATE KEY PASSPHRASE**: A text input field with masked characters (dots) and a toggle icon.
- Buttons**: 'CANCEL' and 'SAVE' buttons at the bottom right.

Creating a Project

- In Ansible Tower, the *project* resource represents a repository that is available in a version control system, and has at least one playbook and its associated resources, such as files and templates.
- The design of Ansible Tower assumes that most Ansible projects are in a version control system.
- It can automatically retrieve updated materials for a project from several commonly used version control systems.
- Ansible Tower support the ability to download and automatically get updates of project materials from SCMs using Git, Subversion, or Mercurial.

Creating a Project for a Git Repository

- Log in to the web-based UI of Ansible Tower.
- Select **Projects** to go to the list of projects. Click **+** to create a new project.
- Enter a unique NAME for the project.
- To assign the project to a specific organization, click the magnifying glass icon for the ORGANIZATION field, and then select your preferred organization.
- From the SCM TYPE list, select the **Git** item.
- Enter the location of the Git repository in the SCM URL field.
- In SCM CREDENTIAL, select the name of the Source Control credential to use (if any is needed).
- Finally, configure how the project gets updates from the version control system (SCM). Available settings are **Clean**, **Delete on Update**, and **Update Revision on Launch**.

Creating a Project for a Git Repository

NEW PROJECT

DETAILS

PERMISSIONS

JOB TEMPLATES

SCHEDULES

* NAME

Example Project

DESCRIPTION

example description

* ORGANIZATION

Q

Default

* SCM TYPE

Git

SOURCE DETAILS

* SCM URL ?

https://gitlab.example.com/student/playbooks.git

SCM BRANCH/TAG/COMMIT

SCM CREDENTIAL

Q

GitLab

SCM UPDATE OPTIONS

☐ CLEAN ?

☐ DELETE ON UPDATE ?

☒ UPDATE REVISION ON LAUNCH ?

CACHE TIMEOUT (SECONDS) ?

0

CANCEL

SAVE

Project “SCM Update Options”

- **Clean**

- This SCM update option removes local modifications to project materials on Ansible Tower before getting the latest revision from the source control repository.

- **Delete on Update**

- This SCM option completely removes the local project repository on Ansible Tower before retrieving the latest revision from the source control repository. For large repositories, this operation takes longer than Clean.

- **Update on Launch**

- This SCM option automatically updates the project from the source control repository each time you use the project to launch a job. Ansible Tower tracks this update as a separate job. If this option is not selected, you must update the project manually.

Creating a Job Template

- A job template is used to run playbooks.
- It combines the project containing the playbook with an inventory, the machine credentials for authenticating to hosts, and parameters that control how the playbook runs.
- Once the template is created it can be reused to run the playbook again, as often as needed.
- Job templates are only set up once but run whenever a playbook is updated.
- They can also be run to ensure expected host configuration and conform to defined standards.
- The **Launch** button, or icon, for a job template is used to run its playbook.

Creating a Job Template

- Log in to the web-based UI for Ansible Tower.
- Click **Templates** to go to the template management interface. Click **+**, and then select **Job Template** to create a new job template.
- Enter a NAME for the job template.
- Select **Run** as the job type.
- Click the magnifying glass icon for the INVENTORY field, then select the inventory to use.
- In the CREDENTIAL field, select the machine credential to use to authenticate to managed hosts.
- Click the magnifying glass icon for the PROJECT field, then select the project containing the playbook that the job template will run.
- Select the playbook that the job will run from the PLAYBOOK list.
The list shows all the playbooks that exist in the project.
- Optionally, you can limit the hosts that will be used with the LIMIT field.
- Click **SAVE**.

Creating a Job Template

The screenshot shows the 'Create Job Template' interface in Ansible Tower. The left sidebar contains navigation links: Jobs, Schedules, My View, Resources, Templates (highlighted), Credentials, Projects, Inventories, Inventory Scripts, Access, Organizations, Users, Teams, Administration, Credential Types, and Notifications. The main form has tabs for DETAILS, PERMISSIONS, NOTIFICATIONS, COMPLETED JOBS, and SCHEDULES. The DETAILS tab is active, showing fields for NAME, DESCRIPTION, JOB TYPE, INVENTORY, PROJECT, PLAYBOOK, CREDENTIAL, FORKS, LIMIT, VERBOSITY, JOB TAGS, SKIP TAGS, INSTANCE GROUPS, JOB SLICING, SHOW CHANGES, and OPTIONS. The 'DevOps' credential is selected in the CREDENTIAL dropdown, and the '3 (Debug)' verbosity level is selected in its dropdown. The 'SHOW CHANGES' toggle is set to 'OFF'.

DETAILS PERMISSIONS NOTIFICATIONS COMPLETED JOBS SCHEDULES **ADD SURVEY**

*** NAME** **DESCRIPTION** *** JOB TYPE** ☐ PROMPT ON LAUNCH

*** INVENTORY** ☐ PROMPT ON LAUNCH *** PROJECT** *** PLAYBOOK**

CREDENTIAL ☐ PROMPT ON LAUNCH **FORKS** **LIMIT** ☐ PROMPT ON LAUNCH

*** VERBOSITY** ☐ PROMPT ON LAUNCH **JOB TAGS** ☐ PROMPT ON LAUNCH **SKIP TAGS** ☐ PROMPT ON LAUNCH

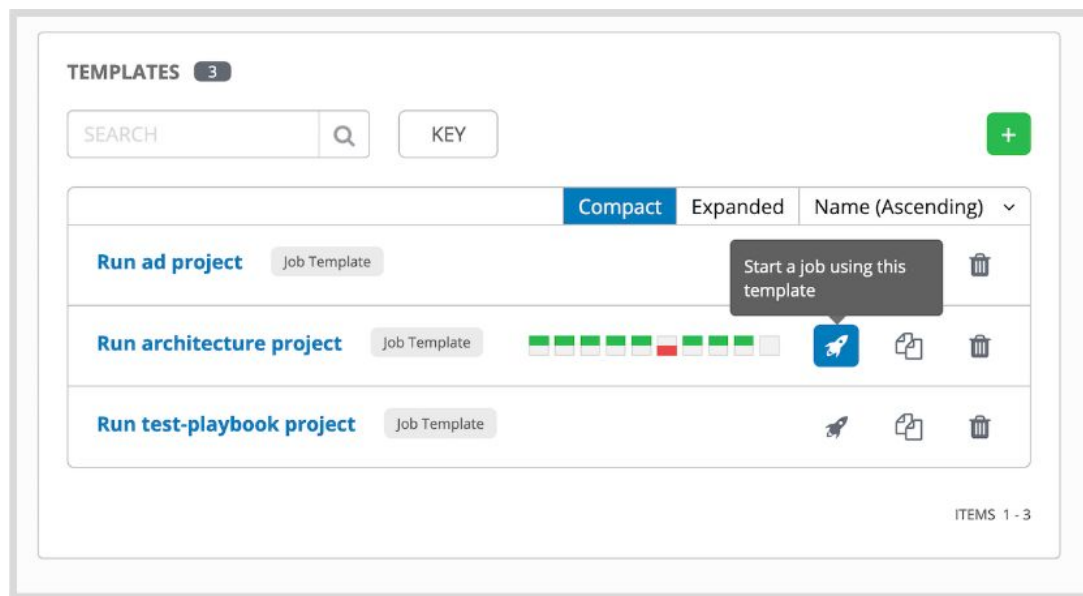
Choose a verbosity
0 (Normal)
1 (Verbose)
2 (More Verbose)
3 (Debug)
4 (Connection Debug)

INSTANCE GROUPS **JOB SLICING**

SHOW CHANGES ☐ PROMPT ON LAUNCH **OPTIONS**
☐ ENABLE PRIVILEGE ESCALATION ☐ ALLOW PROVISIONING CALLBACKS ☐ ENABLE CONCURRENT JOBS ☐ USE FACT CACHE

Launching Jobs

- Click **Templates** to access the list of templates.
- Locate the job template to run from the list of templates, then click the rocket icon beneath the ACTIONS column to launch the job.
- If you have enabled the **Prompt on launch** option for any of the fields, Ansible Tower will prompt you for input before executing the job. After responding, click **LAUNCH** to launch the job.



Evaluating Job Results

- After launching a playbook, Ansible Tower will load the detail page for the running job
- The DETAILS pane displays details of the job's parameters: who ran the job, when it ran, and so on
- The right pane is contains the output of the job, just like the **ansible-playbook** command-line tool. This can be used to review the effects of the job and to troubleshoot it.
- Across the top of the right pane are some summary statistics: how many plays, tasks, and hosts were involved in the playbook run, and how long it took to complete (or has taken so far if it is still running).

The screenshot displays the Ansible Tower web interface for a job titled "Test API Template". The interface is divided into two main panes. The left pane, labeled "DETAILS", shows job metadata: Status is "Successful", Started at "1/31/2020 10:49:34 PM", Finished at "1/31/2020 10:49:47 PM", Job Template is "Test API Template", Job Type is "Run", Launched By is "bonnevil", Inventory is "Demo Inventory", Project is "Test API Project", Revision is "592af14", Playbook is "api.yml", and Credential is "API Vault Credential". It also shows "1" Forks, Environment as "/var/lib/awx/venv/ansible", Execution Node as "localhost", Instance Group as "tower", and Extra Variables as "1 ---". The right pane shows the job output, with a search bar at the top. The output is a JSON response from an API, showing details about a job and its related entities. At the bottom of the right pane, a "PLAY RECAP" section summarizes the results: "localhost" is "ok=3", "changed=0", "unreachable=0", "failed=0", "skipped=0", and "rescued=0".