Red Hat

Managing Ansible with Red Hat Ansible Tower

Configuring and Running Automation

Objectives

This module demonstrates how to:

- Create static inventories, machine credentials, projects, and job templates.
- Launch jobs, use the API, and control access with Role-Based Access Controls.

Controlling Access with Users and Role-Based Access Control

Users and Teams

- Users allow you to control which users can configure resources and run jobs using specific job templates and inventories.
- You can assign users to teams if a group of users all need the same access to certain resources.
- This can be useful to restrict users from using specific credentials or inventories so that they can only run jobs on specific sets of managed hosts.
- You can configure job templates so that users can just launch jobs using the template as-is, or so that they can customize the template before launching it.

User Types

By default, the Ansible Tower installer creates an admin user account with full control of the Ansible Tower installation. Using the special admin account, an Ansible Tower administrator can log in to the web interface and create additional users.

The three user types in Ansible Tower are:

- System Administrator
- System Auditor
- Normal User

Role-Based Access Controls

- Users and teams are assigned *roles* that grant permissions on an object in Ansible Tower.
  - Inventories, credentials, projects, job templates, and so on.

- All users in a team inherit the permissions granted by the team's roles.

- Common roles include
  - **Admin** (granting administrative control over the object)
  - **Read** (granting read access to the object)
  - **Execute** (on a job template, grants permission to launch it)

Organizations

- An *organization* is a logical collection of teams, projects, and inventories.
- All users must belong to an organization.
- Organizations, allow you to configure a collection of users and teams to work with only those Ansible Tower resources that you want them to use.
- As part of the Red Hat Ansible Tower installation, a default organization is created.

Organization Roles

- Newly created users inherit specific roles from their organization, based on their user type.
- You can assign additional roles to a user after creation to grant permissions to view, use, or change other Ansible Tower objects.
- An organization is itself one of these objects.
- There are four roles that users can be assigned on an organization:
    - Organizational Admin
    - Auditor
    - Member
    - Read

Teams

- Teams make managing roles on objects more efficient than managing them for each user separately.

- Rather than assigning the same roles to multiple users, an administrator can assign roles to the team representing a group of users.

- In Ansible Tower, users exist as objects at an Ansible Tower-wide level.
    - Users can be assigned roles in multiple organizations.

- A team, by contrast, belongs to exactly one organization.

Team Roles

- Users can be assigned roles for a particular team.
- These roles control whether the user is considered part of that team, can administer it, or can view its membership.
- A user can be assigned one or more of the following team roles:
    - Admin
    - Member
    - Read

Creating a Static Inventory

Preparing Red Hat Ansible Tower to Manage Hosts

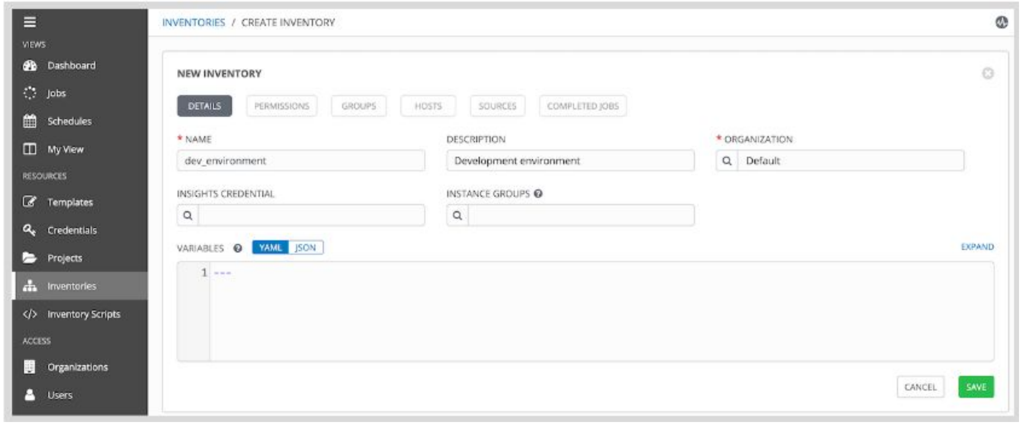
- An **inventory** is a collection of hosts and groups of hosts which are managed by Ansible
- Before you can run a playbook with Ansible Tower, you have to configure it with an inventory
- You can also set variables for those hosts which affect playbook execution
- More than one inventory can be configured
- Different inventories can be used for different purposes

Creating a Static Inventory in Ansible Tower

- The simplest kind of inventory to set up is a static inventory that you manually write
- Two ways to set up a static inventory:
  - Manage it from Ansible Tower's web-based user interface
  - Use an existing inventory file stored in a version control system
- This presentation will illustrate the first option.
- For more information on the second option, see https://docs.ansible.com/ansible-tower/latest/html/administration/scm-inv-import.html

Creating an Inventory in the Ansible Tower Web UI

- Log into Ansible Tower (the admin user will work for this example).
- Click on **Inventories**.
- In the INVENTORIES window, click the **+** button.
- Enter a NAME for the inventory and its ORGANIZATION (often “Default”)



* NAME: dev_environment

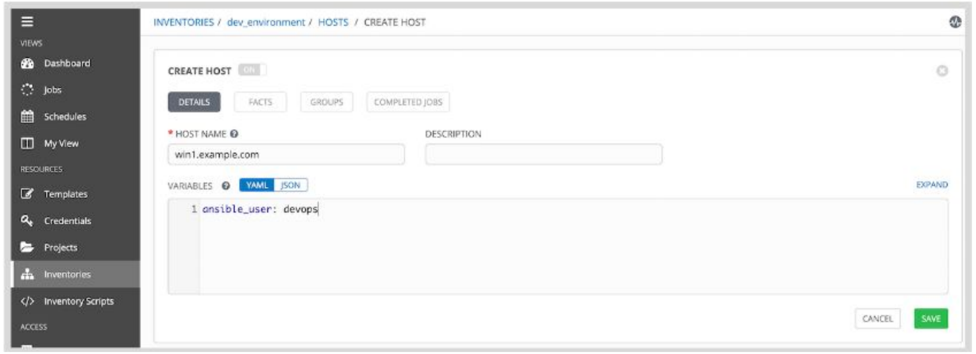
DESCRIPTION: Development environment

* ORGANIZATION: Default

```
1 ---
```

Adding a Host to an Inventory

- In the Ansible Tower GUI, click the **Inventories** menu, then click on the name of the inventory.
  - Click the **HOSTS** button, then click on **+**. This displays the “Create a new host” tooltip.
  - In the HOST NAME field enter the hostname or IP address of the managed host.
  - In the VARIABLES text box, you can set values for variables that apply only to this host.
  - Click **SAVE**.



INVENTORIES / dev_environment / HOSTS / CREATE HOST

CREATE HOST

HOST NAME: win1.example.com

DESCRIPTION:

VARIABLES: YAML

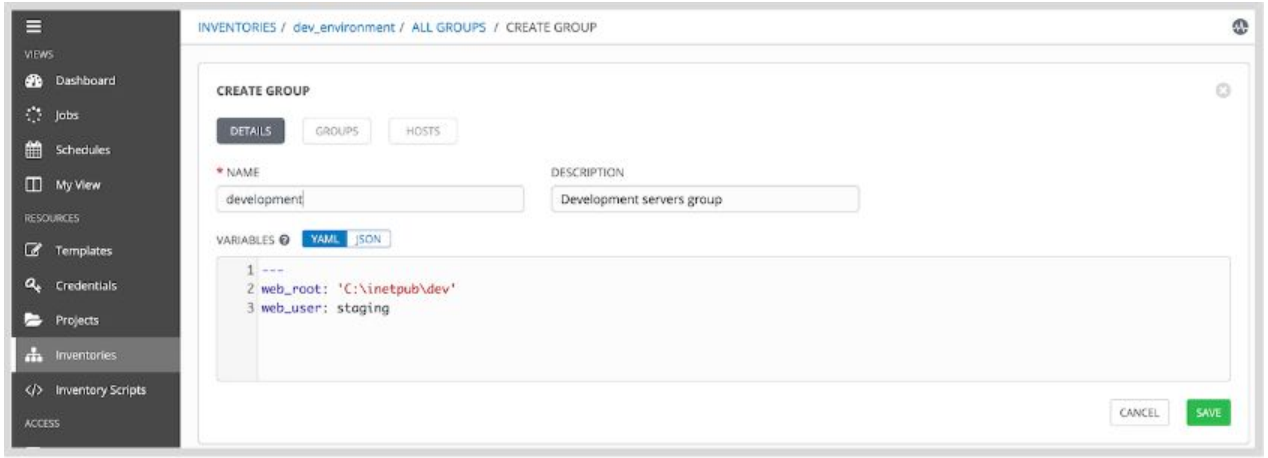
```
ansible_user: devops
```

Organizing Hosts into Groups

- Groups allow you to organize hosts into a set that can be managed together
- Hosts may be in multiple groups at the same time
  - All hosts that are in a particular data center
  - All hosts that have a particular purpose
  - Dev / Test / Prod hosts can be grouped
- Groups can be nested
  - The *europe* group might include a *paris_dc* group and a *london_dc* group
- This allows you to run playbooks on particular groups
- This allows you to set a variable to a specific value for all hosts in a group

Creating a Group

- In the Ansible Tower GUI, click the **Inventories** menu, and click on the inventory to edit.
- Click the **GROUPS** button, then click on **+**. This will open the "Create a new group" tooltip.
- In the NAME field, enter the name of the group.
- Define any values for variables
- Click **SAVE**.



INVENTORIES / dev_environment / ALL GROUPS / CREATE GROUP

CREATE GROUP

DETAILS GROUPS HOSTS

* NAME: development

DESCRIPTION: Development servers group

VARIABLES: YAML JSON

```
---
web_root: 'C:\inetpub\dev'
web_user: staging
```

CANCEL SAVE

Adding a New Host to a Group

- In the Ansible Tower GUI, click the **Inventories** menu, and click on the inventory to edit.
- Click the **GROUPS** button, then click on the group to edit.
- Click the **HOSTS** button, then click on **+**. This will open the “Add a host” tooltip. Select “New Host”.
- In the HOST NAME field, enter the hostname or IP address of the managed host to add.
- Define any values for variables that affect only that host (overriding any group variables).
- Click **SAVE**.

Inventory Roles

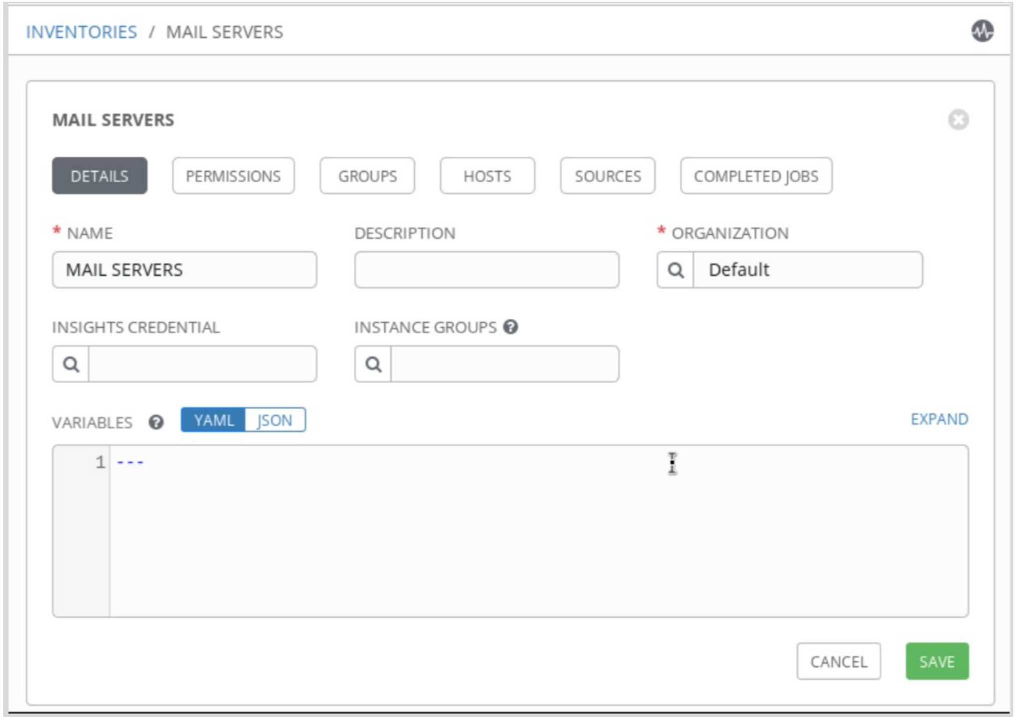
Role	Description
Admin	The inventory **Admin** role grants users full permissions over an inventory. These permissions include deletion and modification of the inventory. In addition, this role also grants permissions associated with the inventory roles **Use**, **Ad Hoc**, and **Update**.
Use	The inventory **Use** role grants users the ability to use an inventory in a job template resource. This controls which inventory is used to launch jobs using the job template's playbook.
Ad Hoc	The inventory **Ad Hoc** role grants users the ability to use the inventory to execute ad hoc commands.
Update	The inventory **Update** role grants users the ability to update a dynamic inventory from its external data source.
Read	The inventory **Read** role grants users the ability to view the contents of an inventory.

Configuring Inventory Variables

When you manage a static inventory in the Ansible Tower web UI, you may define inventory variables directly in the inventory objects.

- Variables set in the inventory details affect all hosts in the inventory.
- Variables set in a group's details are the equivalent of **`group_vars`**.
- Variables set in a host's details are the equivalent of **`host_vars`**.

Configuring Variables for an Inventory



* NAME: MAIL SERVERS

DESCRIPTION:

* ORGANIZATION: Default

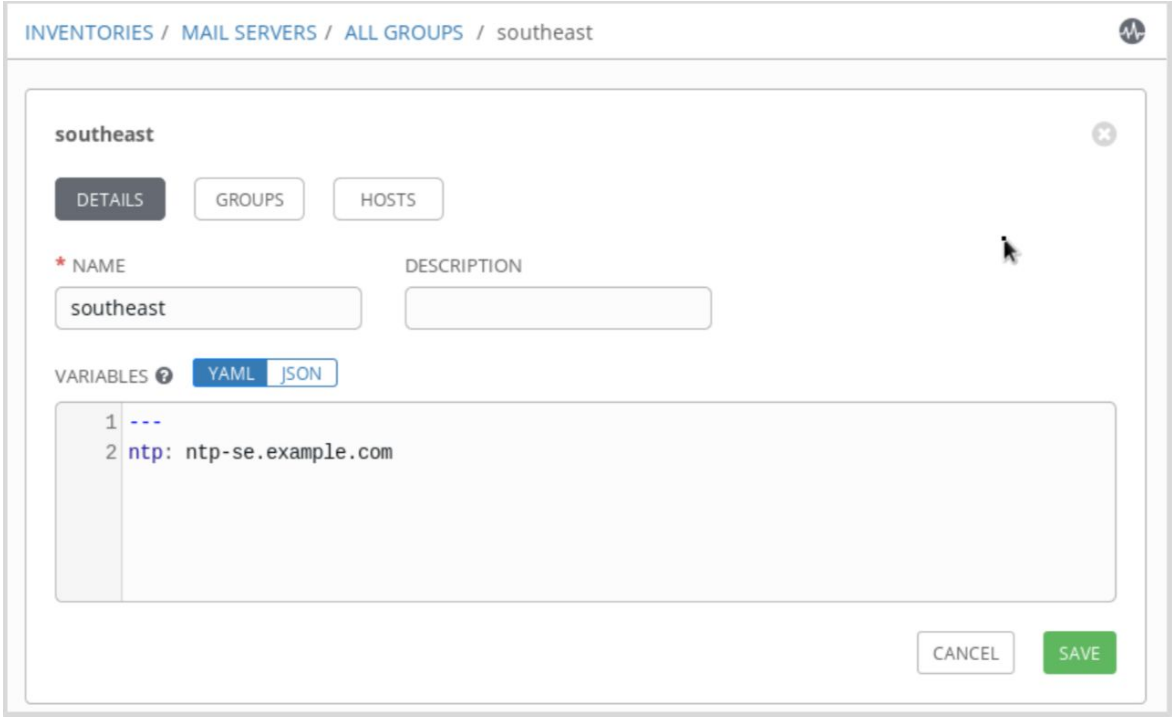
INSIGHTS CREDENTIAL:

INSTANCE GROUPS:

VARIABLES:

```
---
```

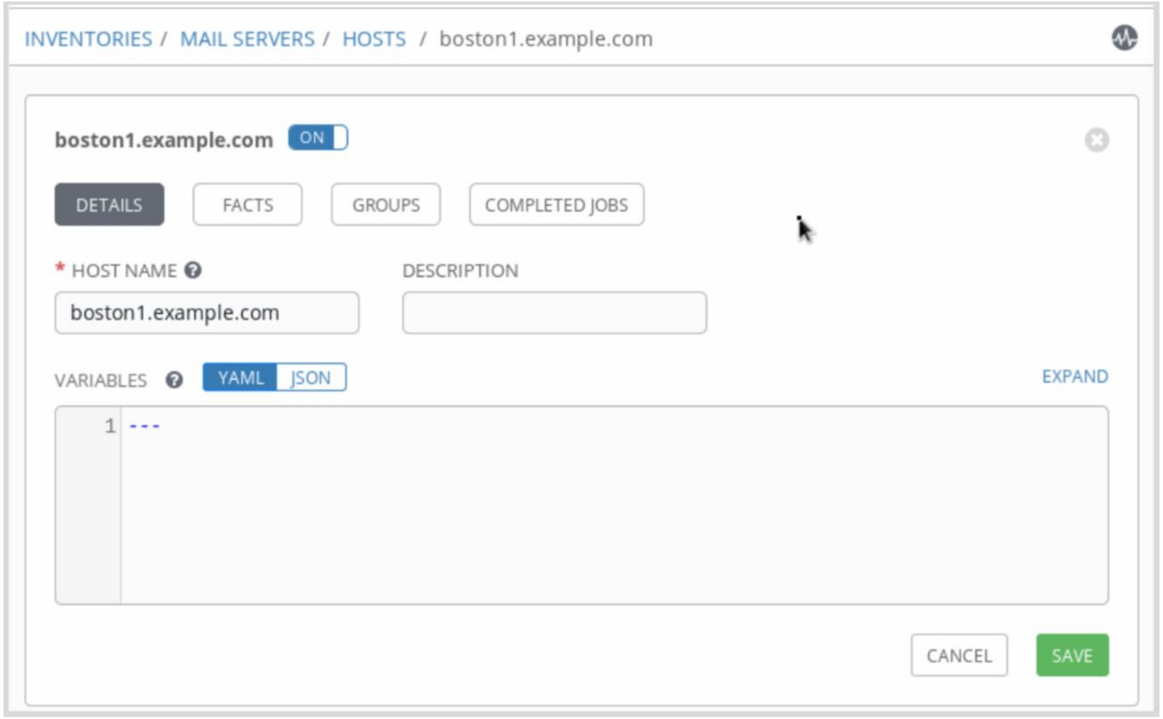
Configuring Inventory Variables for a Group



NAME: southeast

```
---
ntp: ntp-se.example.com
```

Configuring Inventory Variables for a Host



INVENTORIES / MAIL SERVERS / HOSTS / boston1.example.com

boston1.example.com ON

DETAILS FACTS GROUPS COMPLETED JOBS

* HOST NAME	boston1.example.com
DESCRIPTION	
VARIABLES	YAML

```
---
```

Creating Machine Credentials

Creating Machine Credentials

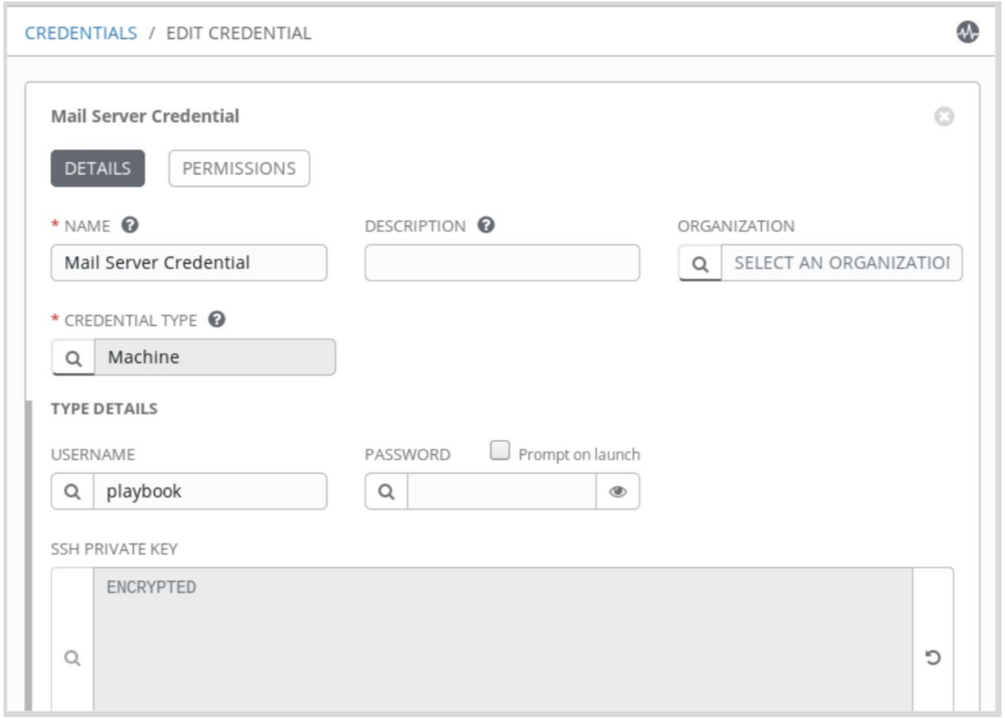
- To securely provide the ansible_user and ansible_password setting for managed hosts, create a machine credential
- Users of Ansible Tower can use these credentials, but cannot retrieve the value of the password directly
- The machine credential’s password is stored in encrypted form in Ansible Tower

Creating Machine Credentials

- In the Ansible Tower web UI, click **Credentials**, then click **+** to “Create a new credential”.
- For NAME, enter a name for your machine credential.
- Select your ORGANIZATION (often “Default”).
- For CREDENTIAL TYPE, select **Machine**, then click **SELECT**.
- For USERNAME, enter the name of the Windows user you use to authenticate.
- For PASSWORD, enter the password for that Windows user.
- Click **SAVE**.

- When you set up your Ansible Playbook’s job template, it can use this credential.
- It is simpler if you can use the same user name and password for a large number of hosts.

Machine Credential Example


CREDENTIALS / EDIT CREDENTIAL

Mail Server Credential

DETAILS | PERMISSIONS

* NAME: Mail Server Credential

DESCRIPTION:

ORGANIZATION: SELECT AN ORGANIZATIOI

* CREDENTIAL TYPE: Machine

TYPE DETAILS

USERNAME: playbook

PASSWORD: ☐ Prompt on launch

SSH PRIVATE KEY: ENCRYPTED

Credential Roles

Role	Description
Admin	The Credential **Admin** role grants Users full permissions on a Credential. These permissions include deletion and modification of the Credential, as well as the ability to use the Credential in a Job Template.
Use	The Credential **Use** role grants Users the ability to use a Credential in a Job Template.
Read	The Credential **Read** role grants Users the ability to view the details of a Credential. This still does not allow them to decrypt the secrets which belong to that Credential through the web interface.

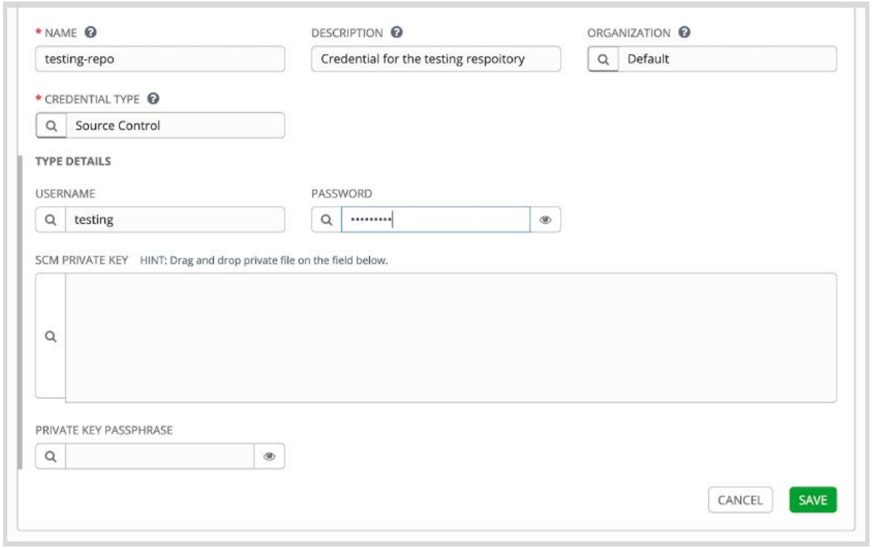
Configuring a Project

Overview: Preparing a Project

- 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)

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**.



* NAME: testing-repo

DESCRIPTION: Credential for the testing respoitory

ORGANIZATION: Default

* CREDENTIAL TYPE: Source Control

TYPE DETAILS

USERNAME: testing

PASSWORD: ••••••••••

SCM PRIVATE KEY HINT: Drag and drop private file on the field below.

PRIVATE KEY PASSPHRASE

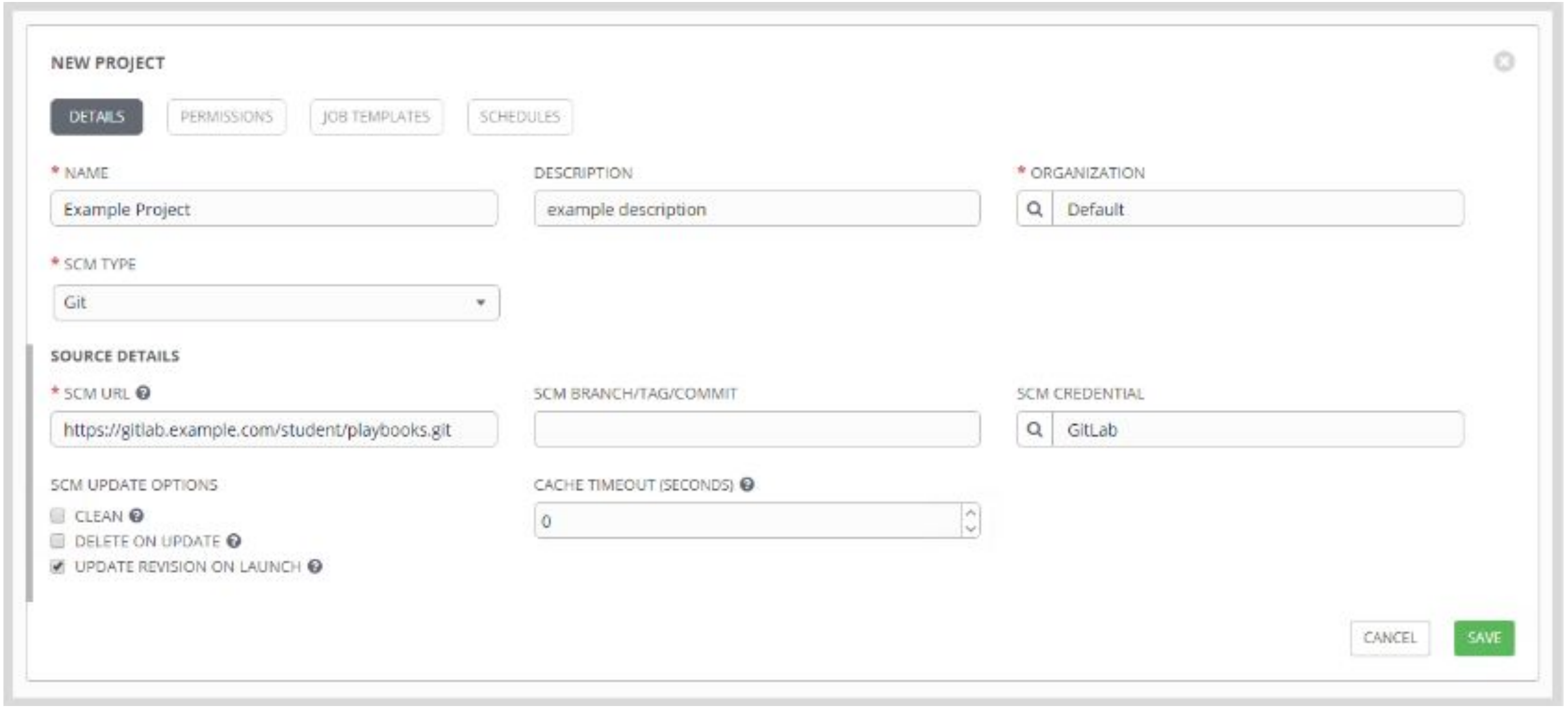
Creating a Project

- 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

* NAME: Example Project	DESCRIPTION: example description	* ORGANIZATION: Default
* SCM TYPE: Git		

SOURCE DETAILS

* SCM URL: https://gitlab.example.com/student/playbooks.git	SCM BRANCH/TAG/COMMIT:	SCM CREDENTIAL: GitLab

SCM UPDATE OPTIONS
- [ ] CLEAN
- [ ] DELETE ON UPDATE
- [x] UPDATE REVISION ON LAUNCH

CACHE TIMEOUT (SECONDS): 0

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.

Project Roles

Role	Description
Admin	The **Admin** role grants users full access to a Project. When granted this role on a Project, a user can delete the Project and modify its properties, permissions included. In addition, this role also grants users the **Use**, **Update**, and **Read** roles.
Use	The **Use** role grants users the ability to use a Project in a Template resource. This role also grants users the permissions associated with the Project **Read** role.
Update	The **Update** role grants users the ability to manually update or schedule an update of a Project's materials from its SCM source. This role also grants users the permissions associated with the Project **Read** role.
Read	The **Read** role grants users the ability to view the details, permissions, and notifications associated with a Project.

Creating Job Templates and Launching Jobs

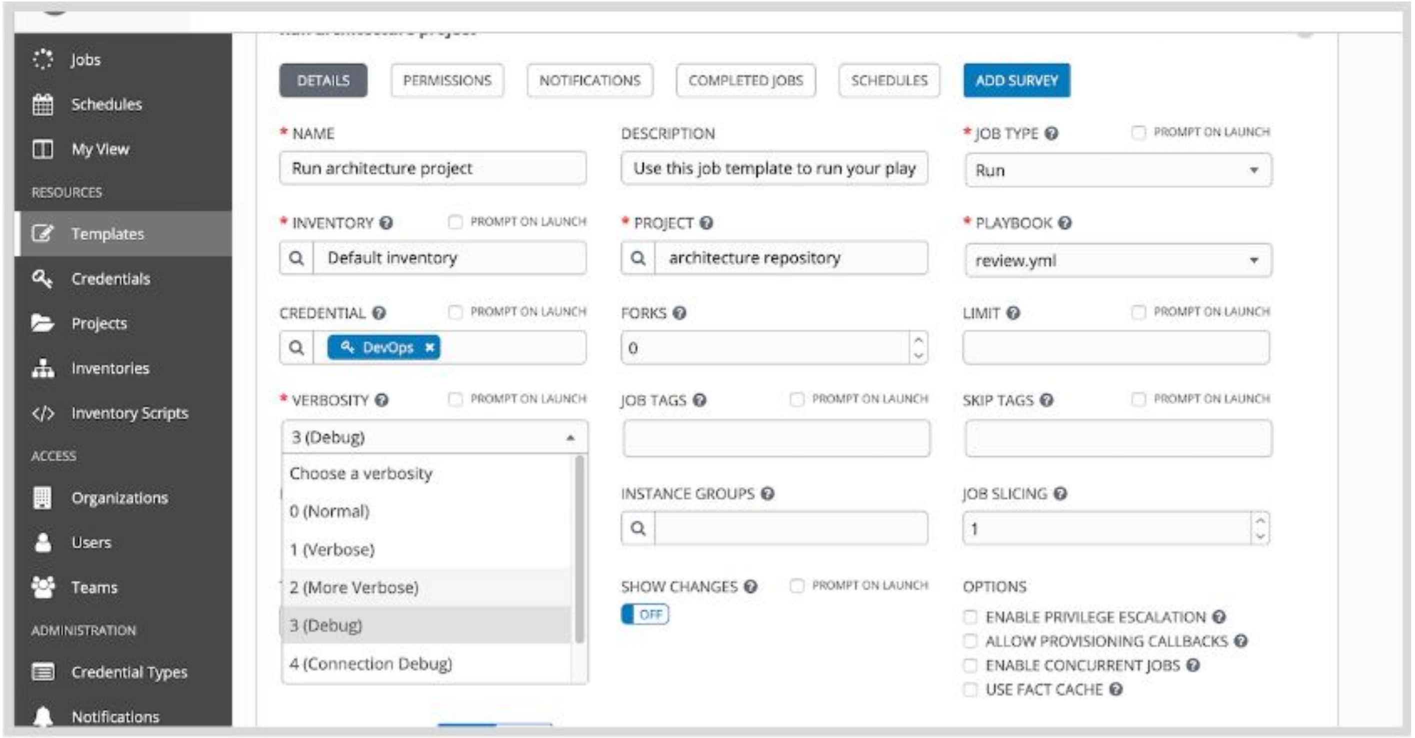
Creating a Job Template

- 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



* NAME: Run architecture project	DESCRIPTION: Use this job template to run your play	* JOB TYPE (PROMPT ON LAUNCH): Run
* INVENTORY (PROMPT ON LAUNCH): Default inventory	* PROJECT: architecture repository	* PLAYBOOK: review.yml
CREDENTIAL (PROMPT ON LAUNCH): DevOps	FORKS: 0	LIMIT (PROMPT ON LAUNCH):
* VERBOSITY (PROMPT ON LAUNCH): 3 (Debug)	JOB TAGS (PROMPT ON LAUNCH):	SKIP TAGS (PROMPT ON LAUNCH):
	INSTANCE GROUPS:	JOB SLICING: 1
	SHOW CHANGES (PROMPT ON LAUNCH): OFF	OPTIONS: ENABLE PRIVILEGE ESCALATION; ALLOW PROVISIONING CALLBACKS; ENABLE CONCURRENT JOBS; USE FACT CACHE

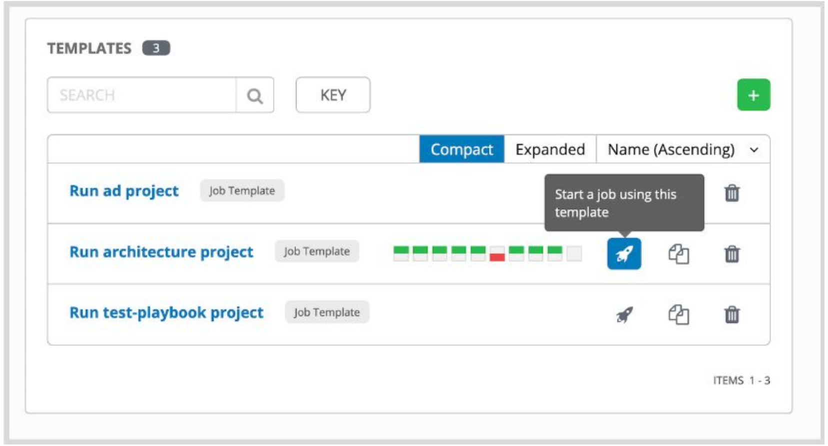
Choose a verbosity
0 (Normal)
1 (Verbose)
2 (More Verbose)
3 (Debug)
4 (Connection Debug)

Job Template Roles

Role	Description
Admin	The **Admin** role provides a user the ability to delete a Job Template or edit its properties, including its associated permissions. This role also grants permissions associated with the Job Template **Execute** and **Read** roles.
Execute	The **Execute** role grants users permission to execute a job using the Job Template. It also grants the Users permission to schedule a job using the Job Template. This role also grants permissions associated with the Job Template **Read** role..
Read	The **Read** role grants users read-only access to view the properties of a Job Template. It also grants access to view other information related to the Job Template, such as the list of jobs executed using the Job Template, as well as its associated permissions and notifications.

Launching Jobs

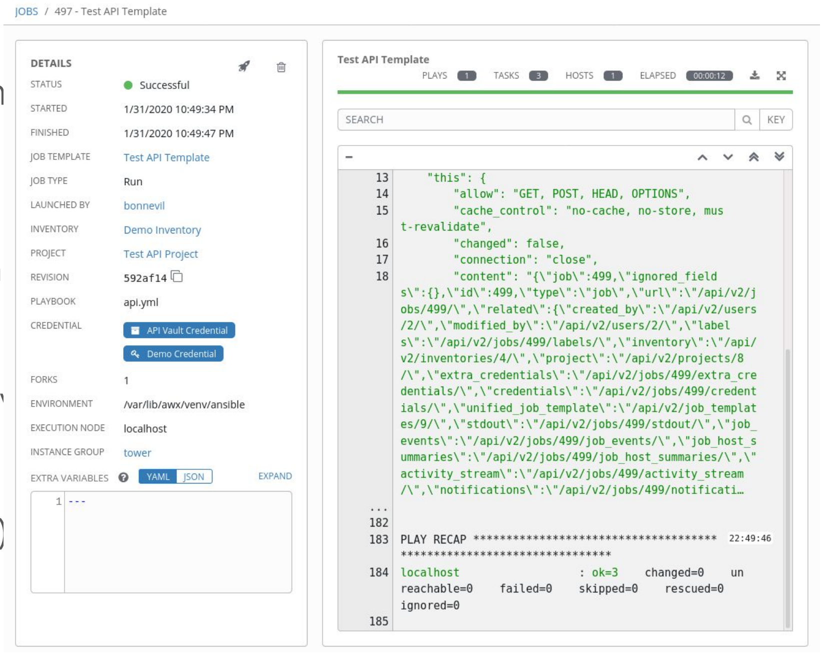
- Click **Templates** to access the list of templates.
- Locate the job template to run from the list of templates, then click the rocket icon beneath the ACTIONS column to launch the job.
- If you have enabled the **Prompt on launch** option for any of the fields, Ansible Tower will prompt you for input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



Run ad project — Job Template
Run architecture project — Job Template
Run test-playbook project — Job Template

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 an to troubleshoot it.
- Across the top of the right pane are some summar 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)



DETAILS

STATUS	● Successful
STARTED	1/31/2020 10:49:34 PM
FINISHED	1/31/2020 10:49:47 PM
JOB TEMPLATE	Test API Template
JOB TYPE	Run
LAUNCHED BY	bonnevil
INVENTORY	Demo Inventory
PROJECT	Test API Project
REVISION	592af14
PLAYBOOK	api.yml
CREDENTIAL	API Vault Credential Demo Credential
FORKS	1
ENVIRONMENT	/var/lib/awx/venv/ansible
EXECUTION NODE	localhost
INSTANCE GROUP	tower
EXTRA VARIABLES	YAML JSON EXPAND

```
---
```

Test API Template

PLAYS 1 TASKS 3 HOSTS 1 ELAPSED 00:00:12

```
    "this": {
        "allow": "GET, POST, HEAD, OPTIONS",
        "cache_control": "no-cache, no-store, mus
t-revalidate",
        "changed": false,
        "connection": "close",
        "content": "{\"job\":499,\"ignored_field
s\":{},\"id\":499,\"type\":\"job\",\"url\":\"/api/v2/j
obs/499/\",\"related\":{\"created_by\":\"/api/v2/users
/2/\",\"modified_by\":\"/api/v2/users/2/\",\"label
s\":\"/api/v2/jobs/499/labels/\",\"inventory\":\"/api/
v2/inventories/4/\",\"project\":\"/api/v2/projects/8
/\",\"extra_credentials\":\"/api/v2/jobs/499/extra_cre
dentials/\",\"credentials\":\"/api/v2/jobs/499/credent
ials/\",\"unified_job_template\":\"/api/v2/job_templat
es/9/\",\"stdout\":\"/api/v2/jobs/499/stdout/\",\"job_
events\":\"/api/v2/jobs/499/job_events/\",\"job_host_s
ummaries\":\"/api/v2/jobs/499/job_host_summaries/\",\"
activity_stream\":\"/api/v2/jobs/499/activity_stream
/\",\"notifications\":\"/api/v2/jobs/499/notificati…
...

PLAY RECAP ************************************  22:49:46
********************************
localhost                  : ok=3    changed=0    un
reachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

Controlling Ansible Tower with the API

Red Hat Ansible Tower's REST API

- Red Hat Ansible Tower provides a Representational State Transfer (REST) API.
- This provides a mechanism that allows control of Ansible Tower outside of the web UI.
- Custom scripts or external applications can access the API using standard HTTP messages.
- The REST API is useful when integrating Ansible Tower with other programs.
- Any programming language, framework, or system that supports HTTP can use the API.
- This provides an easy way to automate repetitive tasks and integrate other enterprise IT systems with Red Hat Ansible Tower.

Using the REST API from the CLI

Use the **`curl`** command to make the request from the command line. The output of the API request is in JSON format.

```
[user@demo ~]$ curl -X GET https://tower.lab.example.com/api/ -k
{"description":"AWX REST API","current_version":"/api/v2/","available_versions":
{"v1":"/api/v1/","v2":"/api/v2/"},"oauth2":"/api/
o/","custom_logo":"","custom_login_info":""}
```

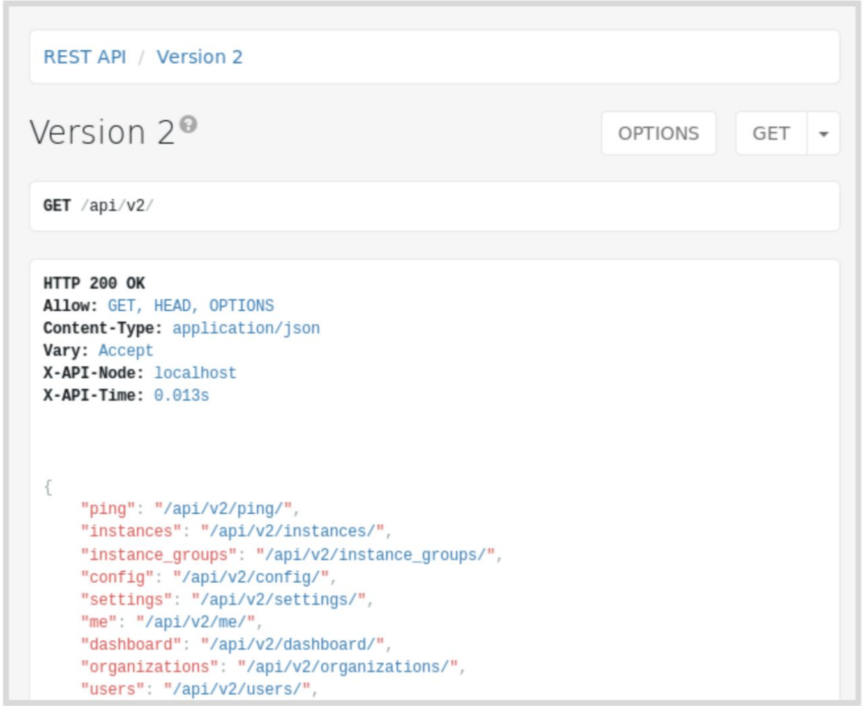
Using the REST API from the Browser

The Red Hat Ansible Tower API is browsable.

- You can access the browsable API at https://tower.lab.example.com/api/.
- You can click the **/api/v2/** link on that page to browse information specific to version 2 of the API.

Click the **?** icon next to an API endpoint name to get documentation on the access methods for that endpoint.

The documentation also provides information on what data those methods return.



```
GET /api/v2/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: localhost
X-API-Time: 0.013s


{
    "ping": "/api/v2/ping/",
    "instances": "/api/v2/instances/",
    "instance_groups": "/api/v2/instance_groups/",
    "config": "/api/v2/config/",
    "settings": "/api/v2/settings/",
    "me": "/api/v2/me/",
    "dashboard": "/api/v2/dashboard/",
    "organizations": "/api/v2/organizations/",
    "users": "/api/v2/users/",
```

Launching a Job Using the API

Launching a job template from the API is done in two steps.

First, access it with the GET method to get information about any parameters or data that you need to launch the job.

```
[user@demo ~]$ curl -X GET --user admin:redhat \
> https://tower.lab.example.com/api/v2/job_templates/"Demo Job Template"/launch/ -
k -s | json_pp
{
   "job_template_data" : {
      "name" : "Demo Job Template",
      "id" : 5,
      "description" : ""
   },
```

Launching a Job Using the API

Then, access it with the POST method to launch the job.

```
[user@demo ~]$ curl -X POST --user admin:redhat \
> https://tower.lab.example.com/api/v2/job_templates/"Demo Job Template"/launch/ -
k -s | json_pp
{
   "result_traceback" : "",
   "passwords_needed_to_start" : [],
   "elapsed" : 0,
   "ignored_fields" : {},
   "skip_tags" : "",
   "summary_fields" : {
      "unified_job_template" : {
         "unified_job_type" : "job",
         "name" : "Demo Job Template",
         "id" : 5,
         "description" : ""
      },
```

Launching a Job Using the API from an Ansible Playbook

- You can launch a job template through Ansible Tower's API with an Ansible Playbook.

- Ansible Tower can even run that playbook from a job template and use it to launch another job template as one of its tasks.

- In the playbook, use a task with the **uri** module to talk to the API.
  - Think of this as equivalent to the **curl** command in the earlier example.
  - You must specify the URL for the job template, using its ID or named URL.
  - You must authenticate to Ansible Tower as a user that has permission to launch the job.
  - The task must run on a managed host that can talk to Ansible Tower, but it does not need to run on the Tower itself.

Launching a Job Using the API from an Ansible Playbook

❶ To access the API, Ansible requires the login credentials for a user that is allowed to launch a job from a job template.

❷ The playbook also requires the URL of the actual API to which Ansible must connect. This example uses the named URL of the Demo Job Template. Notice how the spaces in the job template's name have been specified using the `%20` code. You can also use the **urlencode** filter: `tower_job: "{{ 'Demo Job Template' | urlencode }}"`

❸ Ansible can access the Ansible Tower API from the Ansible Tower server by using the **uri** module.

```
---
- name: Tower API
  hosts: localhost
  become: false

  vars:
    tower_user: admin❶
    tower_pass: redhat
    tower_host: demo.example.com
    tower_job: Demo%20Job%20Template❷

  tasks:
    - name: Launch a new Job
      uri:❸
        url: https://{{ tower_host }}/api/v2/job_templates/{{ tower_job }}/launch/
        method: POST
        validate_certs: no
        return_content: yes
        user: "{{ tower_user }}"
        password: "{{ tower_pass }}"
        force_basic_auth: yes
        status_code: 201
```

Troubleshooting Ansible Tower and Reviewing Logs

Objectives

- Perform simple troubleshooting of the Red Hat Ansible Tower installation.
- Identify log files for Red Hat Ansible Tower services.

Troubleshooting Red Hat Ansible Tower

- In this presentation, we will look at the Red Hat Ansible Tower service itself.
- This will focus on the components that make up Red Hat Ansible Tower, not playbook debugging.
- When troubleshooting playbooks, look at your job output first.

Components of Red Hat Ansible Tower

There are four main components managed together:

● Nginx	Provides web server for the UI and API
● PostgreSQL	Internal relational database server
● **supervisord**	Process control system that manages the application: running jobs, etc.
● *rabbitmq-server*	AMQP message broker supporting signalling by application components

In addition, a **memcached** memory caching daemon is used as a local caching service.

Network Ports and Component Communication

- These services communicate with each other using normal network protocols.

Nginx	80/tcp and 443/tcp
PostgreSQL	5432/tcp
rabbitmq-server	**beam** listens on 5672/tcp, 15672/tcp, and 25672/tcp

- If you are running a multi-machine clustered Ansible Tower installation, these may need to be exposed to allow the different servers of your Ansible Tower cluster to talk to each other.

- This is why setting good passwords for PostgreSQL and RabbitMQ in your setup inventory is important.

- In the single-machine integrated database installation demonstrated here, you only need to expose 80/tcp and 443/tcp.

Controlling and Checking Ansible Tower Services

The **ansible-tower-service** command can be used to check the status of Ansible Tower services and control them:

ansible-tower-service status	Check if the services are running properly
ansible-tower-service start	Start all services
ansible-tower-service stop	Stop all services
ansible-tower-service restart	Restart all services

Controlling and Checking Ansible Tower Services

- The Red Hat Ansible Tower web application is a collection of Django-based components managed by **supervisord**.
- You can use **supervisord status** to check the status of these services:

```
[root@tower ~]# supervisord status
exit-event-listener                     RUNNING  pid 4111, uptime 0:42:55
tower-processes:awx-callback-receiver   RUNNING  pid 4116, uptime 0:42:55
tower-processes:awx-celeryd             RUNNING  pid 4118, uptime 0:42:55
tower-processes:awx-celeryd-beat        RUNNING  pid 4117, uptime 0:42:55
tower-processes:awx-channels-worker     RUNNING  pid 4112, uptime 0:42:55
tower-processes:awx-daphne              RUNNING  pid 4115, uptime 0:42:55
tower-processes:awx-uwsgi               RUNNING  pid 4113, uptime 0:42:55
```

Key Configuration and Data Files

- The main configuration files for Ansible Tower are kept in the **/etc/tower** directory.
  - The most important is **settings.py**, which specifies the locations for job output, storage, and other key directory locations.
  - You should not need to edit these manually, use **setup.sh**.

- A number of key data files are kept in the **/var/lib/awx** directory:
  - **/var/lib/awx/projects** is where your project files are downloaded
  - **/var/lib/awx/job_status** stores job status output from playbook runs
  - **/var/lib/awx/public/static** is the static root directory for the Django applications
  - You should not need to manage these manually either, but your Ansible Tower can have problems if the storage device for **/var/lib/awx** becomes full.

Red Hat Ansible Tower Log Files

- The **/var/log/tower** directory stores logs for the key servers supporting the application:
  - **tower.log** The main log file for Ansible Tower.
  - **task_system.log** Logs various system housekeeping tasks.
  - **setup*.log** Log files from setup.sh when run to install, backup, or restore.

- The **/var/log/supervisor** directory stores logs for the Django-based applications:
  - **supervisord.log** The main log file for **supervisord**.
  - Other files contain information about the activity of various applications.

- You can configure Ansible Tower to send information to log aggregation services.
  - See https://docs.ansible.com/ansible-tower/latest/html/administration/

Common Troubleshooting Scenarios

- **Problems running playbooks:**
  - Playbooks are confined to /tmp on the Ansible Tower server when run.
  - This limits resources they can access and can impact tasks delegated to localhost.

- **Problems connecting to managed hosts:**
  - Verify that you can establish an SSH connection (or WinRM with Windows hosts) to the managed host.
  - Review your inventory file and check host names, IP addresses, and connection variables.

- **Playbook in the project does not appear in the list when configuring a job template:**
  - Make sure the YAML syntax of the playbook is correct and can be parsed by Ansible
  - **/var/lib/awx/projects** should allow user awx to view the files

Common Troubleshooting Scenarios

- **Playbook stays in "Pending" state:**
  - Ensure that Ansible Tower has enough memory available.
  - Use **supervisord status** to make sure the Django applications are running.
  - Ensure that **/var** has at least 1 GB of free space.
  - Try **ansible-tower-service restart**

- **Provided hosts list is empty ("Skipping: No Hosts Matched"):**
  - Make sure the **hosts** declaration in your play matches the group or host names in the inventory.
  - Make sure group names have no spaces in them. Underscores are valid.
  - If you specified a limit in the job template, make sure its syntax is valid and matches something in your inventory.

Common Troubleshooting Scenarios

If you need to change the password for the built-in admin superuser:

- Log in to the Ansible Tower server as the Linux user **root** or **awx**.
- Run **awx-manage changepassword admin**.

```
[awx@tower ~]$ awx-manage changepassword admin
Changing password for user 'admin'
Password:
Password (again):
Password changed successfully for user 'admin'
[awx@tower ~]$
```

Conclusion

Learn More about Red Hat Training and Certification

- Congratulations on completing this course! Want to learn more? Visit the Red Hat Training and Certification page to explore Red Hat courses and certifications.
- Join the Red Hat Learning Community to ask questions and access a collaborative learning environment that enables open source skill development.

Thank you

Red Hat is the world's leading provider of

enterprise open source software solutions.

Award-winning support, training, and consulting

services make

Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat