Ansible Automation Platform Quick Setup Guide

Introduction to Ansible

Ansible is an open-source automation tool used for configuration management, application deployment, and IT orchestration. It simplifies the management of complex IT environments by automating repetitive tasks, enabling faster deployments, and ensuring consistency across systems.

Key Features of Ansible:

1. Agentless Architecture: Ansible operates without the need for agent installation on target machines. It uses SSH for communication, making it lightweight and easy to set up.
2. Declarative Syntax: Using YAML-based playbooks, Ansible provides a simple, human-readable language to describe automation workflows.
3. Idempotency: Ansible ensures tasks are executed only when changes are required, preventing redundant operations and maintaining system stability.
4. Extensibility: It supports custom modules, plugins, and integrations with third-party tools for specialized use cases.
5. Cross-Platform Support: Ansible can manage multiple platforms, including Linux, Windows, cloud environments (AWS, Azure, GCP), containers, and networking devices.

Use Cases:

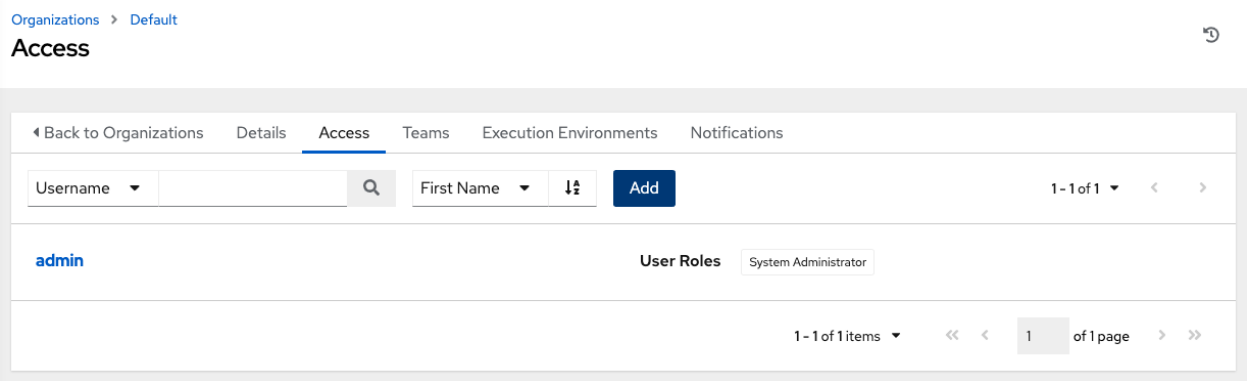
1. Configuration Management: Automate the setup and maintenance of server configurations.
2. Application Deployment: Simplify the deployment of applications across multiple servers.
3. Provisioning: Spin up virtual machines, containers, or cloud instances.
4. Orchestration: Coordinate complex workflows involving multiple systems and applications.

Advantages of Ansible:

- Easy to learn and use due to its simple syntax.
- Agentless design reduces overhead and security concerns.
- Large community and extensive documentation.
- Seamless integration with CI/CD pipelines and DevOps workflows.

CREATE A USER

Users associated with an organization are shown in the **Access** tab of the Organization.



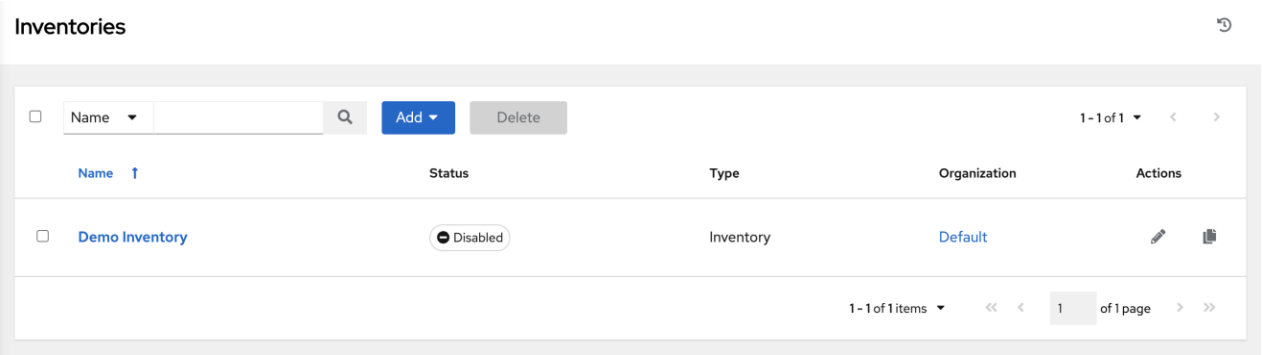
A default *admin* user with the role of System Administrator has been automatically created and is available to all users of automation controller. It can be used as is or edited later as needed. You may add other users to an organization, but you must create the user(s) first. Refer to the Users section in the *Automation Controller User Guide* for detail.

For the purpose of this Quick Start Guide, leave the default user as i

CREATE A NEW INVENTORY

An inventory is a collection of hosts managed by the controller. Organizations are assigned to inventories, while permissions to launch playbooks against inventories are controlled at the user and/or team level. For more information, refer to Inventories, Users - Permissions, and Teams - Permissions in the *Automation Controller User Guide*.

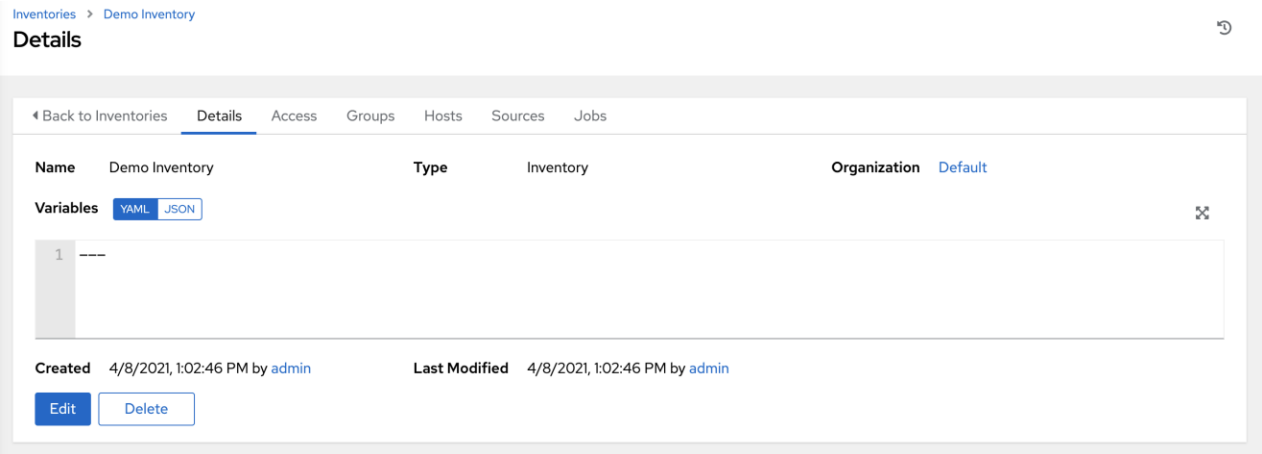
To view existing inventories, click **Inventories** from the left navigation bar.



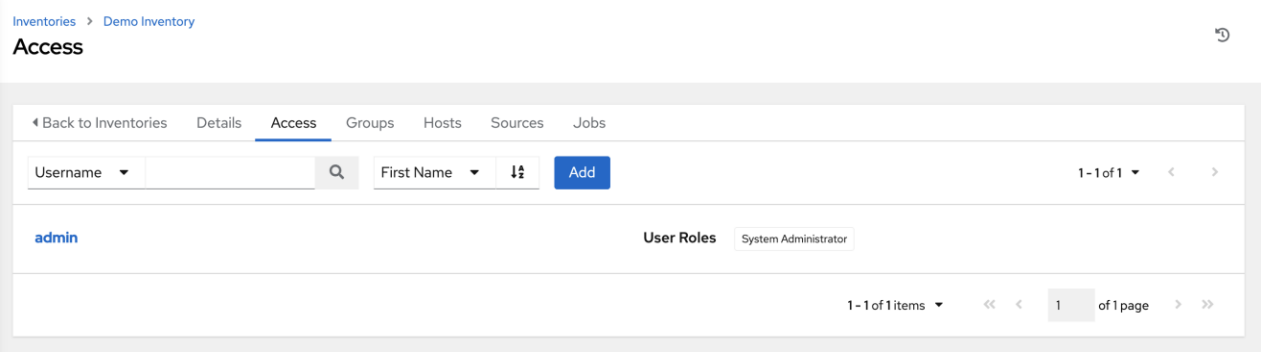
automation controller provides a demo inventory for you to use as you learn how the controller works. It can be used as is or edited later as needed. You may create another inventory if necessary. Refer to Add a new inventory in the *Automation Controller User Guide* for detail.

For the purpose of this Quick Start Guide, leave the default inventory as is.

Click **Demo Inventory** to view its details.



As with organizations, inventories also have associated users and teams that you can view through the **Access** tab.



A default admin user with the role of System Administrator has been automatically populated for this demo inventory.

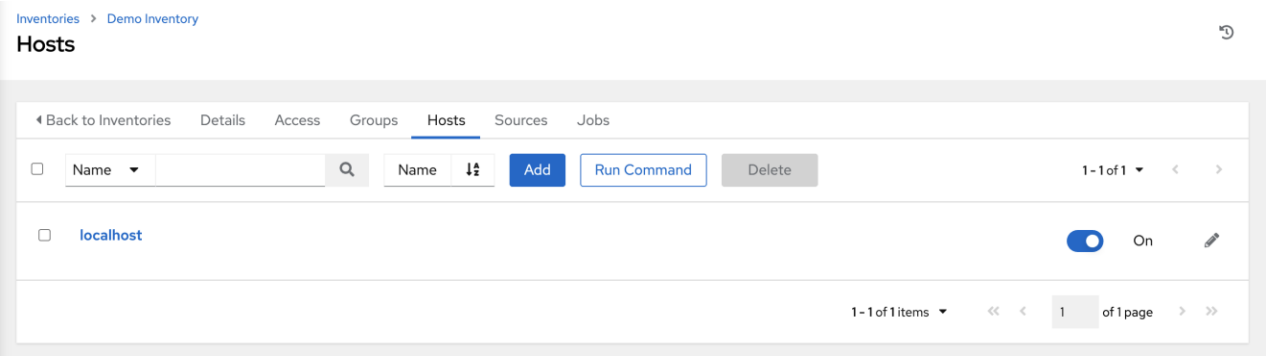
8.1 Groups and Hosts

Note that inventories are divided into groups and hosts. A group might represent a particular environment (e.g. “Datacenter 1” or “Stage Testing”), a server type (e.g. “Application Servers” or “DB Servers”), or any other representation of your environment. The groups and hosts that belong to the Demo inventory are shown in the **Groups** and **Hosts** tabs, respectively.

To add new groups, click the **Add** button in the Groups screen.

Similarly, in the **Hosts** tab, click the **Add** button to add hosts to groups.

For the purposes of this Quick Start and to test that the controller is setup properly, a local host has been added for your use.



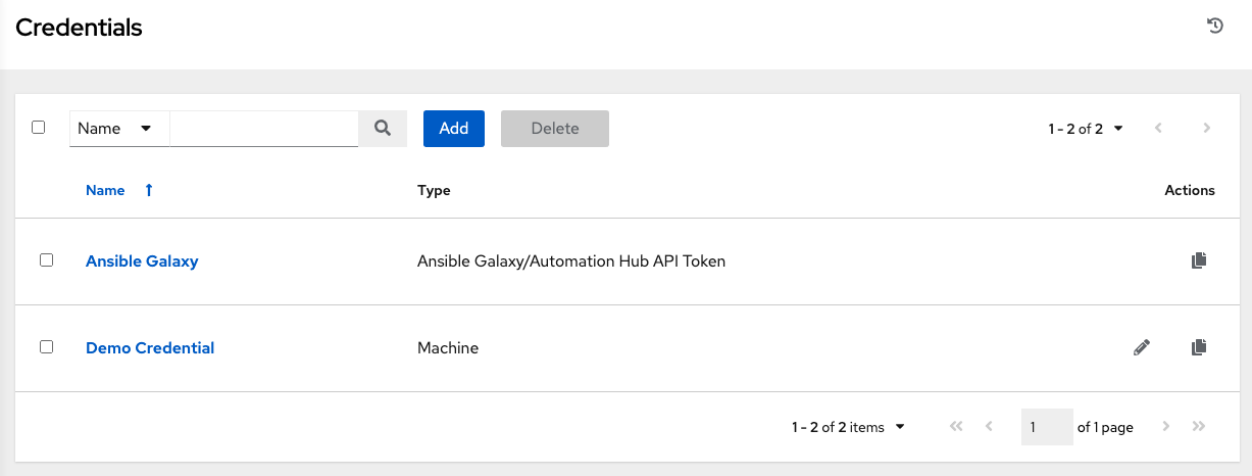
Suppose that the organization you created earlier has a group of web server hosts supporting a particular application. To add these hosts to the inventory, create a group and add the web server hosts.

Click **Cancel** (if no changes were made) or use the breadcrumb navigational links at the top of the automation controller browser to return to the Inventories list view. Clicking **Save** does not exit the Details dialog

CREATE A CREDENTIAL

Credentials authenticate the controller user to launch Ansible playbooks, which can include passwords and SSH keys, against inventory hosts. You can also require the controller user to enter a password or key phrase when a playbook launches using the credentials feature of the controller.

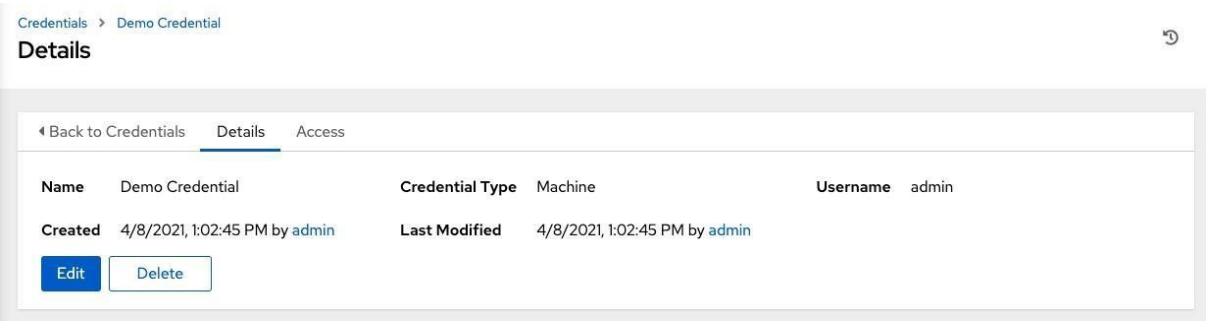
Access the list of credentials by clicking **Credentials** from the left navigation bar.



For the purpose of this Quick Start, a demo credential and a Galaxy credential have been provided for your use. The provided Galaxy credential serves as a template and can only be copied and not edited. You may add more credentials as necessary. Refer to Add a New Credential in the *Automation Controller User Guide* for detail.

Note: When setting up additional credentials, keep in mind that the user you assign must have root access or be able to use SSH to connect to the host machine.

Click **Demo Credential** to view its details.



For the purpose of this Quick Start Guide, leave the default demo credential as is. But to edit the default credential later - from the Details tab, click **Edit**, or from the Credentials list view, click the Edit () button next to the credential name to edit the appropriate details, then save your changes.

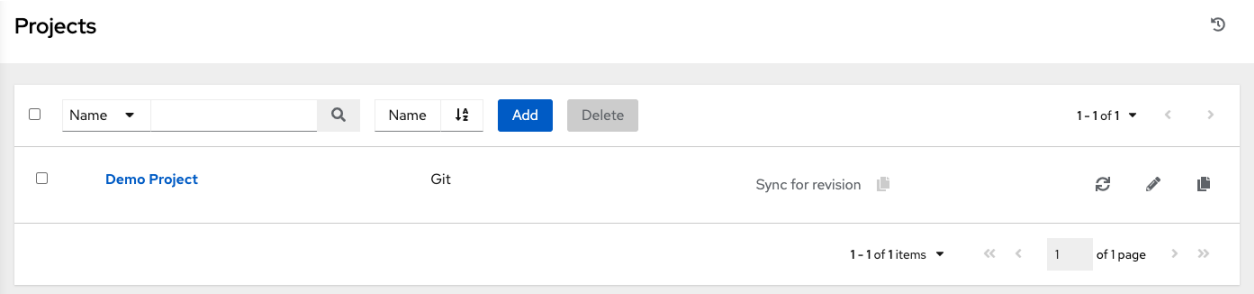
SETTING UP A PROJECT

A Project is a logical collection of Ansible playbooks, represented in the controller.

You can manage playbooks and playbook directories by either placing them manually under the Project Base Path on your controller server, or by placing your playbooks into a source code management (SCM) system supported by the controller, including Git, Subversion, and Mercurial.

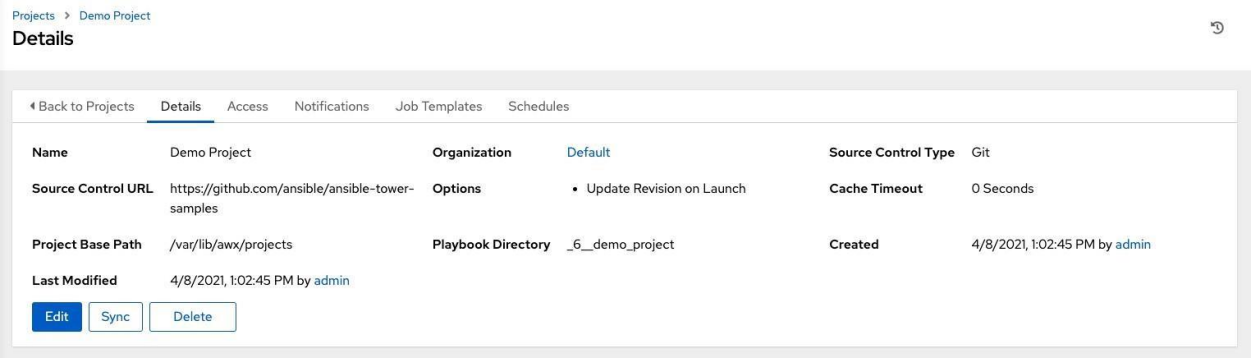
Note: It is recommended that, whenever possible, you use source control to manage your playbooks. This type of best practice provides the ability to treat your infrastructure as code and is in line with DevOps ideals. While this Quick Start Guide uses lightweight examples to get you up and running, we suggest using source control to manage playbook for production purposes.

To review existing projects, click **Projects** from the left navigation bar.



automation controller simplifies the startup process by providing you with a Demo Project to work with initially.

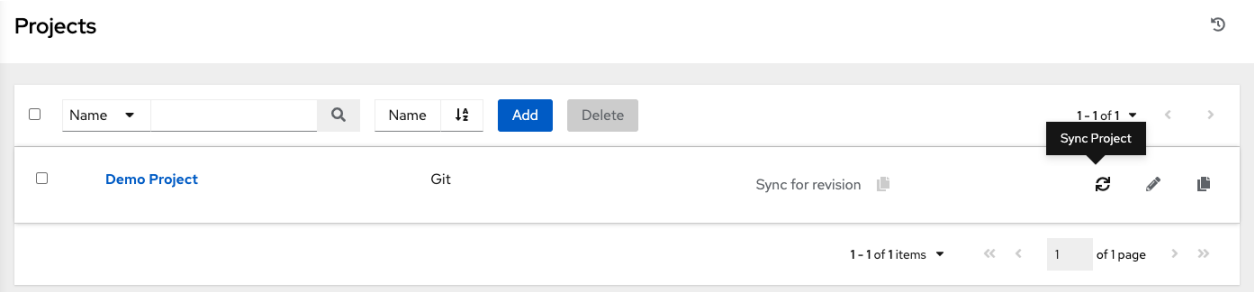
Click on **Demo Project** to view its details.



For the purpose of this Quick Start Guide, leave the default demo project as is. But to edit the default project later - from the Details tab, click **Edit**, or from the Projects list view, click the Edit ( ) button next to the project name

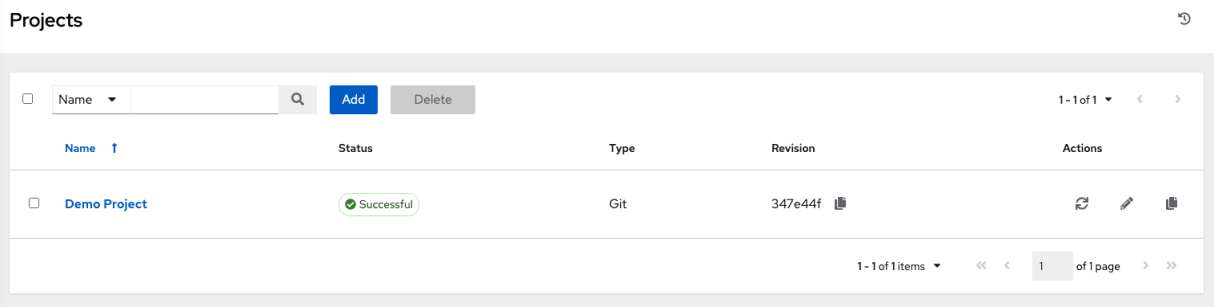
to edit the appropriate details, then save your changes.

Before this project can be used in a job template, you must manually start an SCM sync for this project. Update the SCM-based demo project from the Details tab, click **Sync**, or from the Projects list view, click the Sync ( ) button next to the project name.



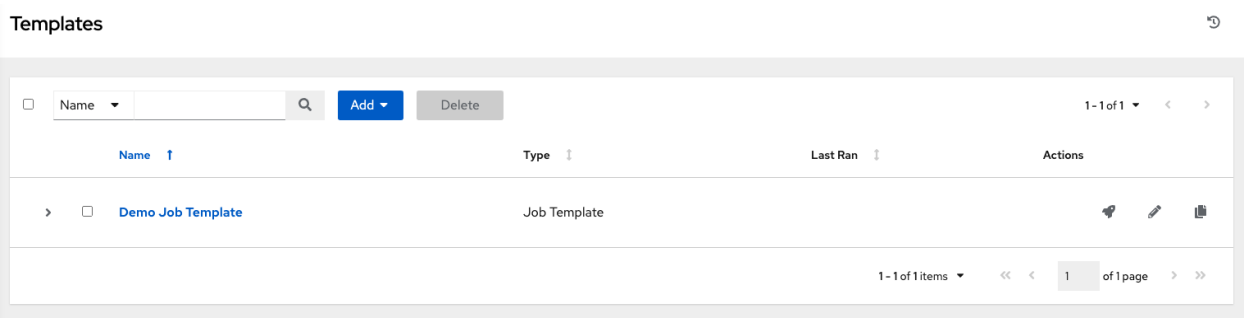
Note: Please note that immediately after adding new projects setup to use source control, a "sync" automatically starts that fetches the project details from the configured source control. Because the Demo project is pre-stocked, however, you must manually start the inventory sync in order for this project to be used in a job template.

Notice that the status indicator beside the name of the project and the revision details update once the sync has completed.



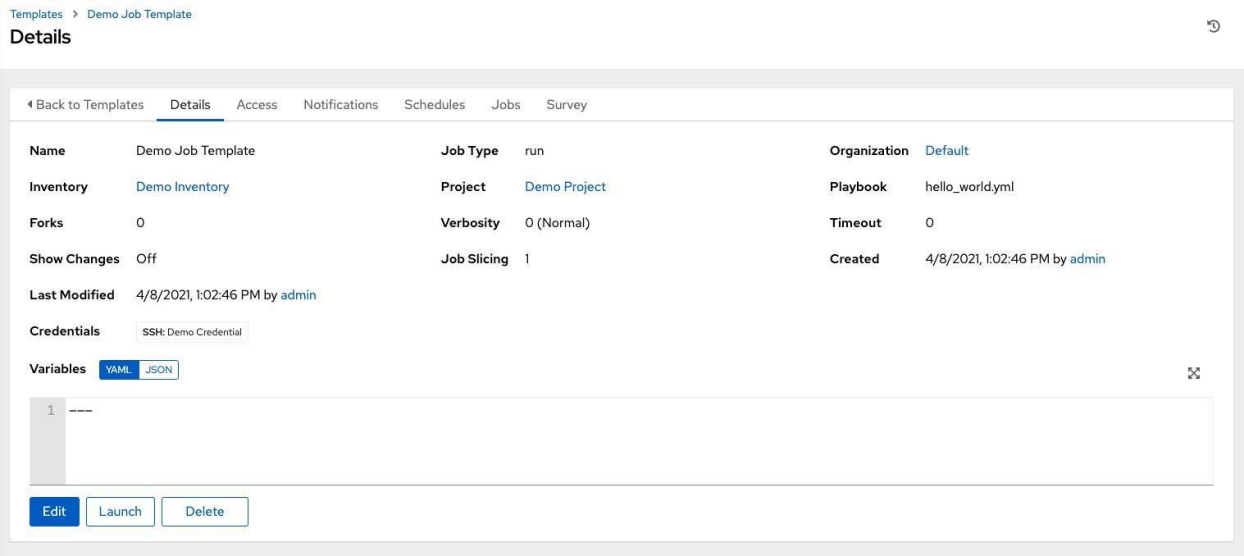
CREATE A NEW JOB TEMPLATE

A job template combines an Ansible playbook from a project and the settings required to launch it. Review existing job templates by clicking **Templates** from the left navigation bar.

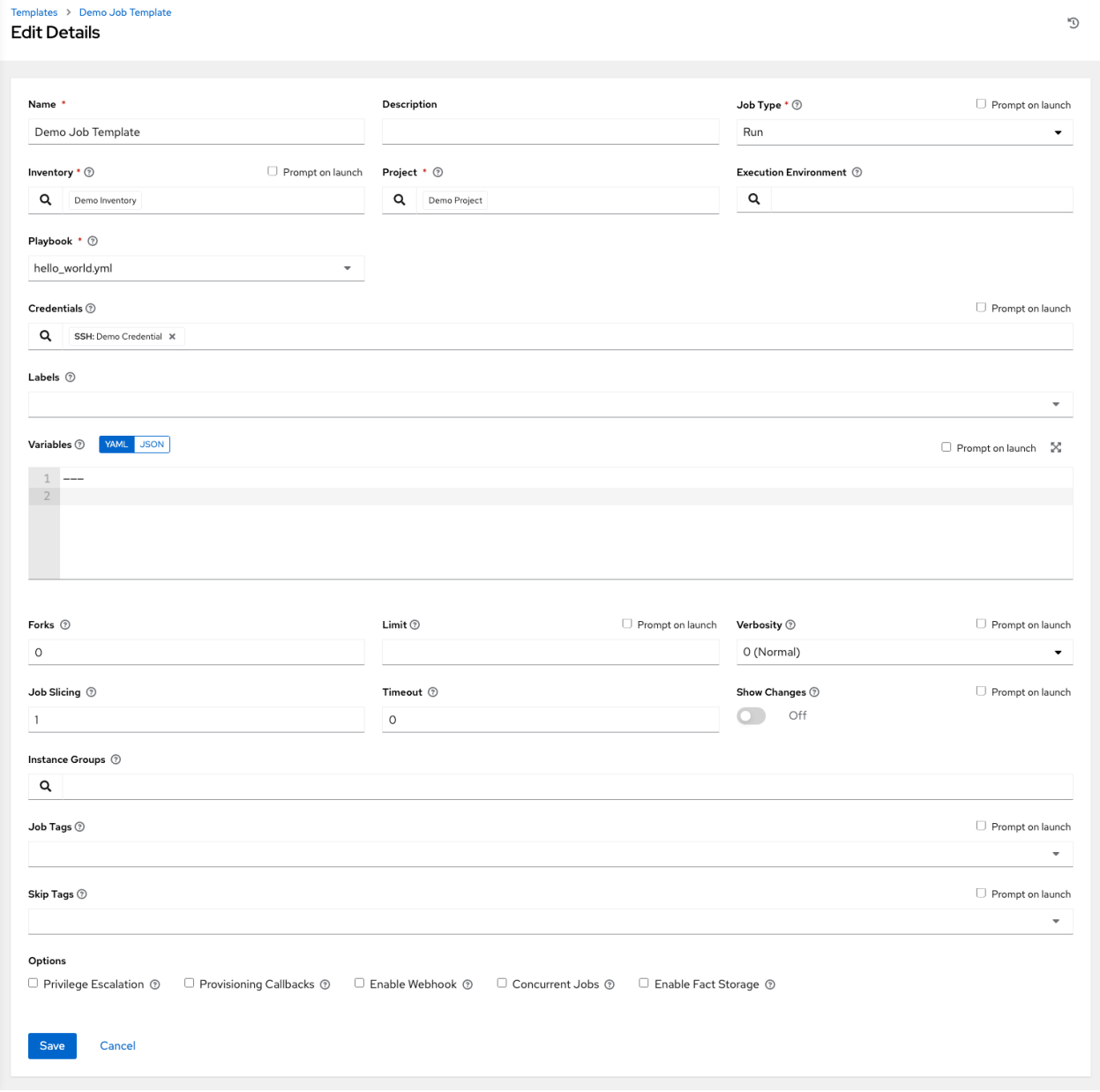


For the purpose of this Quick Start, a Demo Job Template has been created for your initial use.

Click **Demo Job Template** to view its details.



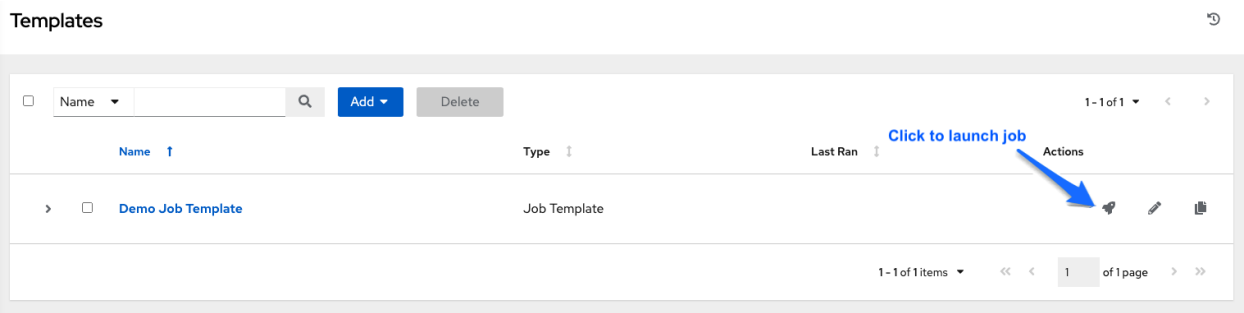
For the purpose of this Quick Start Guide, leave the default demo job template as is. But to edit the default template later - from the Details tab, click **Edit**, or from the Templates list view, click the Edit ( ) button next to the template name to edit the appropriate details, then save your changes.



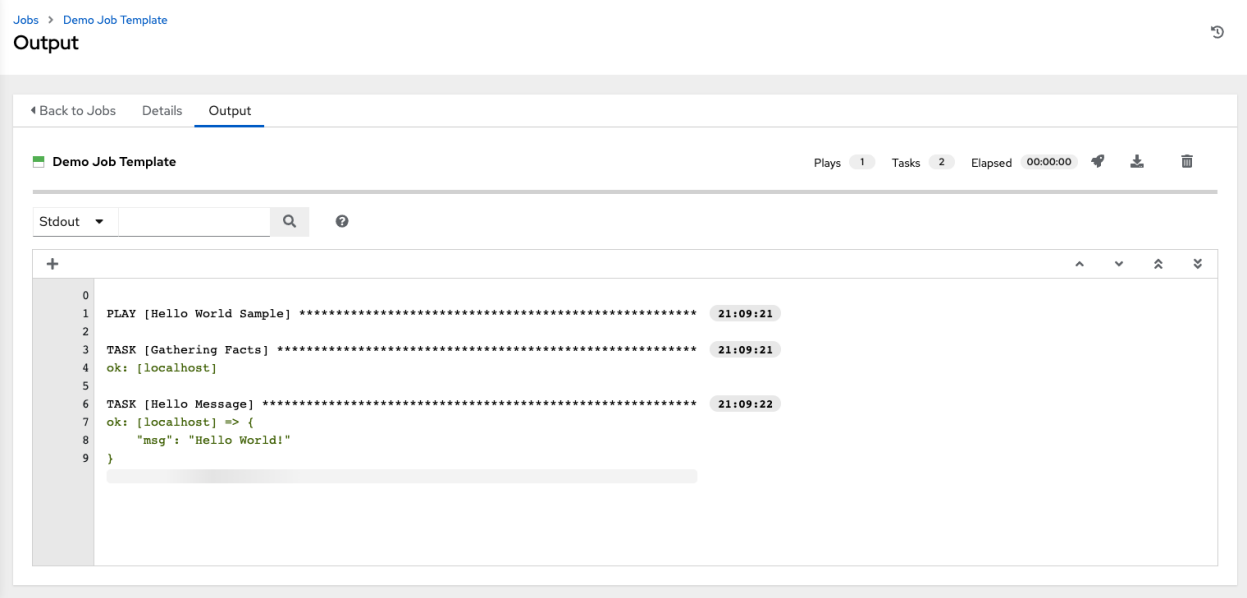
Click **Cancel** (if no changes were made) or use the breadcrumb navigational links at the top of the automation controller browser to return to the Templates list view. Clicking **Save** does not exit the Details dialog.

LAUNCH IT!

From the Templates list view, click the Launch ( ) button to run the Demo Job Template.



The initial job launch returns a status page which updates automatically using the controller's Live Event feature until the job is complete. Once complete, the job results look like the following:



For more details on the job results, refer to Jobs.

Congratulations! Your controller installation is officially setup and running properly.

Setting Up a Ansible Playbook

1. Prerequisites

- **Ansible Installed:** Ensure Ansible is installed on your control machine. Install it using pip install ansible or your system's package manager.
- **Target Hosts:** Have a server(s) with SSH access and appropriate privileges for deploying and configuring Nginx.
- **Inventory File:** An inventory file listing the target server(s) where you want to deploy Nginx.

---

2. Plan Your Playbook

An Ansible playbook for Nginx typically involves:

- Installing Nginx.
- Starting and enabling the Nginx service.
- Deploying custom configurations (optional).
- Opening required firewall ports (optional).

---

3. Create the Inventory File

Create an inventory file (inventory.yml) specifying your target hosts:

```
all:
  hosts:
    nginx-server:
      ansible_host: <IP_ADDRESS>
      ansible_user: <USERNAME>
      ansible_ssh_private_key_file: /path/to/private/key
```

Replace <IP_ADDRESS>, <USERNAME>, and the path to your private SSH key with your details.

---

4. Write the Ansible Playbook

Create a playbook file (nginx_setup.yml) with the following content:

server.js | nginx_setup.yml | package.json

nginx_setup.yml

```yaml
---
- name: Install and configure Nginx
  hosts: nginx-server
  become: true
  tasks:
    - name: Update apt cache
      ansible.builtin.apt:
        update_cache: yes

    - name: Install Nginx
      ansible.builtin.apt:
        name: nginx
        state: present

    - name: Start and enable Nginx service
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true

    - name: Copy custom Nginx configuration (optional)
      ansible.builtin.copy:
        src: /path/to/local/nginx.conf
        dest: /etc/nginx/nginx.conf
        owner: root
        group: root
        mode: "0644"
      notify:
        - Restart Nginx

    - name: Open HTTP port in the firewall (optional)
      ansible.builtin.ufw:
        rule: allow
        port: "80"
        proto: tcp

    - name: Open HTTPS port in the firewall (optional)
      ansible.builtin.ufw:
        rule: allow
        port: "443"
```

server.js | nginx_setup.yml | package.json

nginx_setup.yml

```yaml
        state: present

    - name: Start and enable Nginx service
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true

    - name: Copy custom Nginx configuration (optional)
      ansible.builtin.copy:
        src: /path/to/local/nginx.conf
        dest: /etc/nginx/nginx.conf
        owner: root
        group: root
        mode: "0644"
      notify:
        - Restart Nginx

    - name: Open HTTP port in the firewall (optional)
      ansible.builtin.ufw:
        rule: allow
        port: "80"
        proto: tcp

    - name: Open HTTPS port in the firewall (optional)
      ansible.builtin.ufw:
        rule: allow
        port: "443"
        proto: tcp

  handlers:
    - name: Restart Nginx
      ansible.builtin.service:
        name: nginx
        state: restarted
```

5. Prepare Custom Configuration (Optional)

If you're deploying a custom nginx.conf, prepare it on your control machine and reference its path in the src field in the copy task.

---

6. Run the Playbook

Execute the playbook with the following command:

```
ansible-playbook -i inventory.yml nginx_setup.yml
```

---

7. Verify the Setup

- **Check Nginx Status:** SSH into the target server and run:
- systemctl status nginx
- **Verify HTTP Service:** Open your server's public IP in a browser or use curl:
- curl http://<SERVER_IP>

---

8. (Optional) Automate Testing

- Use a module like ansible.builtin.uri to verify that Nginx is serving content:
- - name: Verify Nginx is running
- ansible.builtin.uri:
- url: http://localhost
- return_content: yes

---

9. Advanced Customizations

- Add tasks for setting up SSL certificates using Let's Encrypt or custom certificates.
- Use roles for modularizing the playbook for larger setups.
- Configure logging or monitoring for Nginx.