



ANSIBLE
TOWER
by Red Hat®

Contents

| | |
|--|----|
| Introduction to Ansible and Ansible Tower | 3 |
| Ansible Tower Dashboard | 3 |
| Enable Self-Service IT | 3 |
| Job Scheduling and Multi-Playbook Workflows..... | 4 |
| Manage and Track Your Entire Inventory | 5 |
| Integrate Tower Using the REST API and CLI Tool | 6 |
| About the test drive | 7 |
| Deployment Architecture | 8 |
| Getting Started..... | 9 |
| Lab 1: Getting Started with Ansible Tower | 9 |
| Access Ansible tower | 9 |
| Tower Dashboard..... | 9 |
| Add Credentials..... | 10 |
| Add Projects | 12 |
| Add Inventory and Test Connectivity..... | 14 |
| Lab 2: Deploy Web Servers on Linux VM using Ansible Playbook | 21 |
| Deploy Nginx Web Server | 21 |
| Remove Nginx Web Server | 24 |
| Deploy Apache Web Server | 27 |
| Lab 3: Manage Windows VM using Ansible Playbook | 30 |
| Create a Local User Account in Windows VM..... | 30 |
| Install a Package..... | 34 |
| Deploy IIS Web Server..... | 38 |

Introduction to Ansible and Ansible Tower

Ansible, an open source community project sponsored by Red Hat, is a **simple automation language** that can perfectly describe an IT application infrastructure in the form of an Ansible Playbook. Ansible is also an **automation engine** that runs Ansible Playbooks.

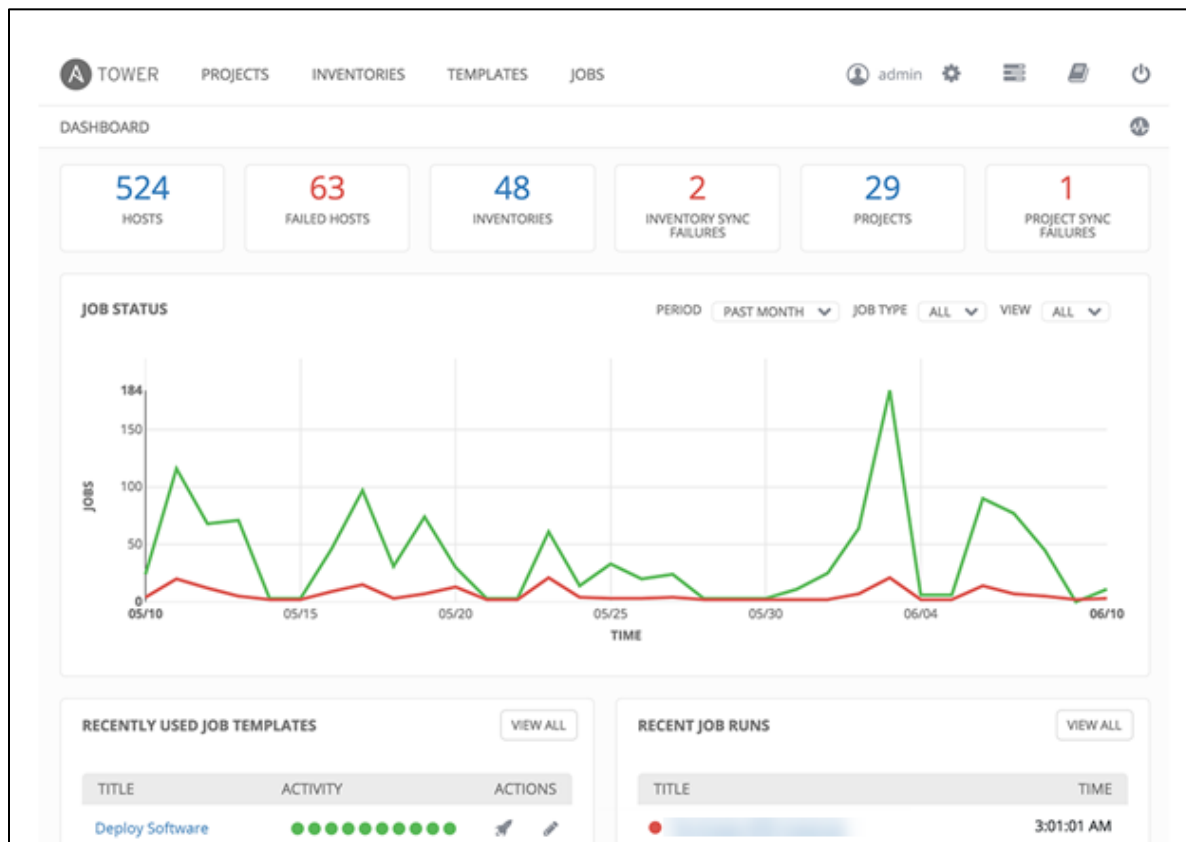
Ansible Tower by **Red Hat** is a commercial offering that helps teams control, secure and manage their Ansible automation. Tower builds on the underlying Ansible automation engine by adding a visual dashboard, role-based access control (RBAC), job scheduling and graphical inventory management.

Using Ansible Tower, all Ansible automations are **centrally logged**, ensuring **complete auditability and compliance**, across the organization.

Ansible Tower Dashboard

The Tower dashboard provides a heads-up NOC-style display for everything going on in your Ansible environment.

As soon as you log in, you'll see your host and inventory status, all the recent job activity and a snapshot of recent job runs. Adjust your job status settings to graph data from specific job and time ranges.



Enable Self-Service IT

Tower lets you launch Playbooks with just a single click. **Role-based access control (RBAC)** keeps environments secure, and teams efficient. Non-privileged users can **safely deploy** entire applications with **push-button deployment** access.

Tower's simplified portal mode and survey features allow IT administrators to delegate automation job runs to users across the organization - synchronized directly from corporate directories such as LDAP, Active Directory or delegated SAML authentication.

With Tower RBAC, developers or QA departments can provision their own development and test environments. Customer service agents can provision a new demo environment. Or junior admins can run simple jobs - like changing passwords - all at the press of a button.

LAUNCH JOB | DEPLOY SOFTWARE

INVENTORY
CREDENTIAL
SURVEY

* ENTER NUMBER OF SERVICE INSTANCES.

* PLEASE SELECT THE SERVICE OWNER.

* ENTER PASSWORD FOR DEPLOYED CERTIFICATE.

SHOW

INVENTORY
Cloud staging servers

CREDENTIAL
Staging ssh key

CANCEL
LAUNCH

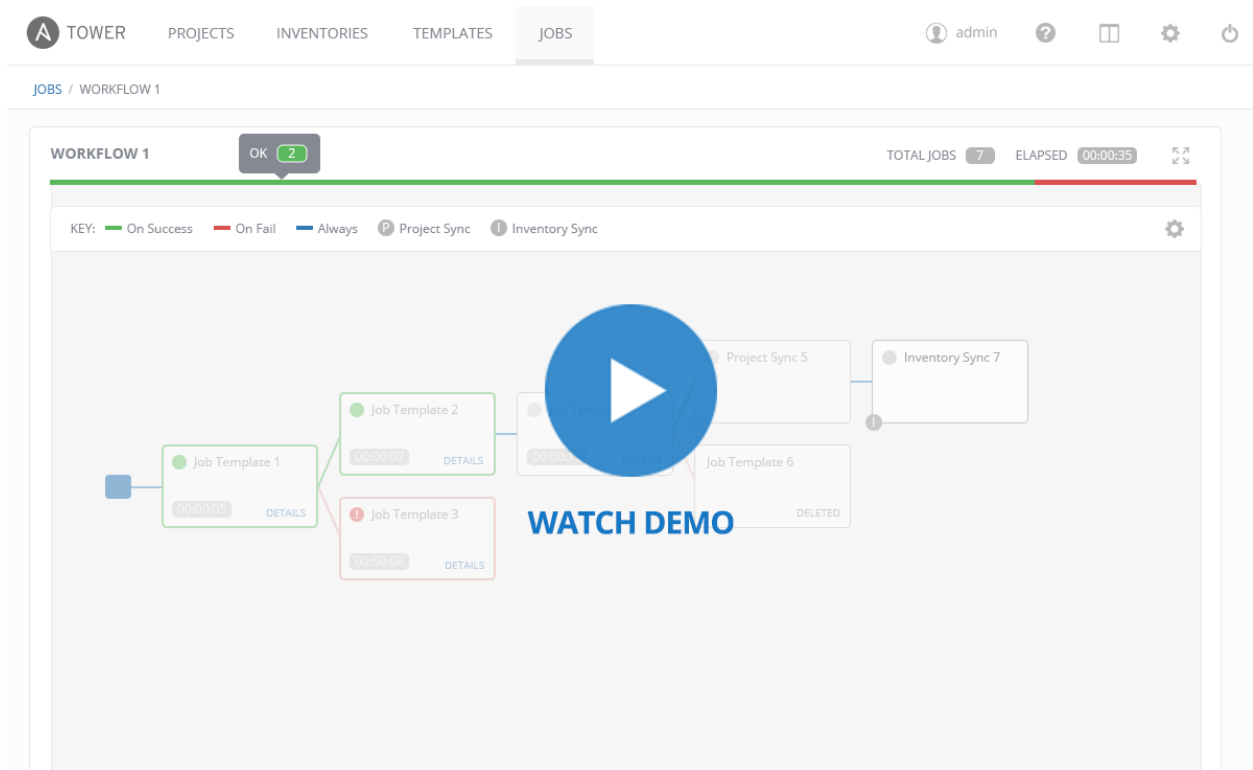
Job Scheduling and Multi-Playbook Workflows

Playbook runs, cloud inventory updates, and source control updates can be scheduled inside Tower - run now, run later, or run forever.

Set up occasional tasks like nightly backups, periodic configuration remediation for compliance, or a full continuous delivery pipeline with just a few clicks.

Tower's multi-Playbook workflows chain any number of Playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

Tower workflows allow for many complex operations. You can build a provisioning workflow that provisions machines, applies a base system configuration, and deploys an application, all with different Playbooks maintained by different teams. You can build a CI/CD testing workflow that builds an application, deploys it to a test environment, runs tests, and automatically promotes the application based on test results. Set up different Playbooks to run in case of success or failure of a prior workflow Playbook. Easily model complex processes with Tower's intuitive workflow editor.



Manage and Track Your Entire Inventory

Tower helps you manage your entire infrastructure. Easily pull your inventory from public cloud providers such as Amazon Web Services, Microsoft Azure, and more. Synchronize from your local OpenStack cloud or VMware environment. Connect your inventory directly to your Red Hat Satellite or Red Hat CloudForms environment. Or connect Tower directly to your custom CMDB.

Tower can keep your cloud inventory in sync, and Tower's powerful provisioning callbacks allow nodes to request configuration on demand, enabling auto scaling.

TOWER

PROJECTS

INVENTORIES

TEMPLATES

JOB

admin

INVENTORIES / MANAGE CLOUD STAGING SERVERS / EDIT

CLOUD SERVERS

DETAILS

NOTIFICATIONS

*NAME

Cloud servers

DESCRIPTION

SOURCE

Amazon EC2

CLOUD CREDENTIAL

Q

Amazon keys

REGIONS

US East (Northern Virginia)

INSTANCE FILTERS

tag:Name=*staging*

ONLY GROUP BY

UPDATE OPTIONS

☒ Overwrite

☒ Overwrite Variables

☐ Update on Launch

VARIABLES

YAML

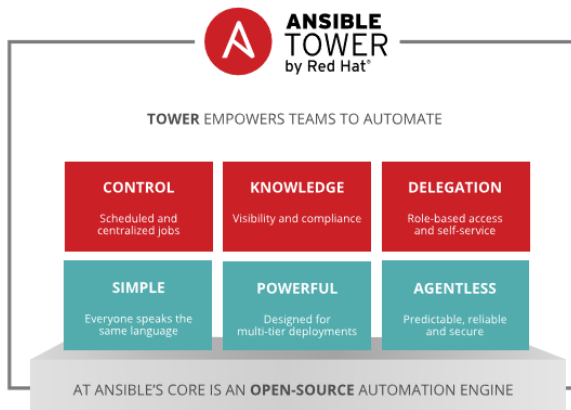
JSON

1

Integrate Tower Using the REST API and CLI Tool

Every feature of Tower is available via Tower's REST API, providing the ideal API for a systems management infrastructure to build against. Call Tower jobs from your existing build tools, show Tower information in your custom dashboards and more. Get API usage information and best practices with built-in documentation.

If it's easier for you to wrap a command line interface than write REST code, Tower's CLI tool is available for launching jobs from CI systems such as Jenkins, or when you need to integrate with other command line tools.



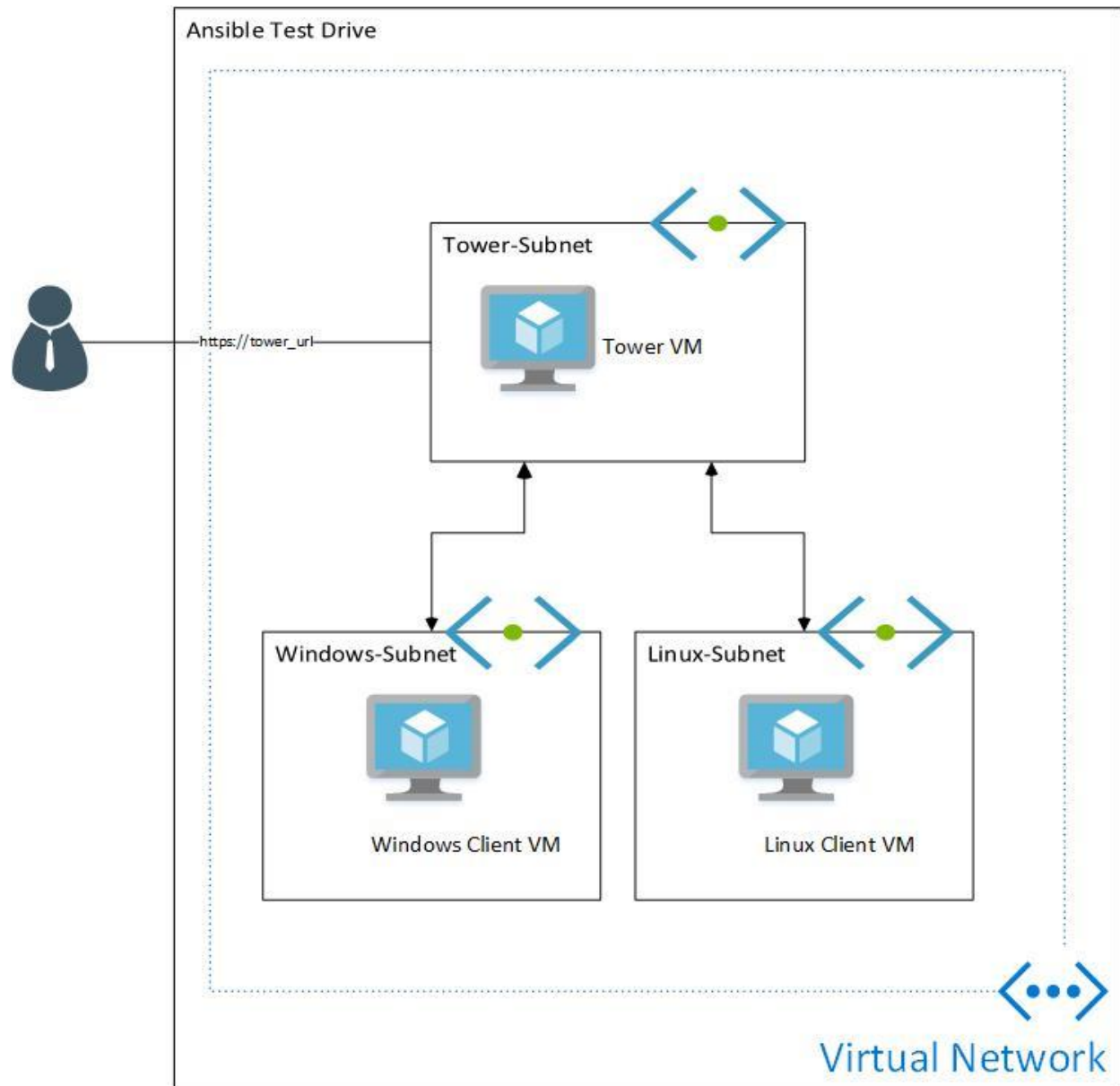
About the test drive

This test drive will help you experience the capabilities of Ansible Tower. As described in previous sections, Ansible Tower builds upon the Ansible Engine. To get the most out of this test drive, the user should be familiar with Ansible and how Ansible Playbooks work. Most users find Ansible incredibly easy to learn and are writing playbooks the same day.

The Ansible website hosts many resources for those wishing to learn more about Ansible (<https://www.ansible.com/get-started>).

Ansible Tower comes with a feature rich Dashboard to provide users with a nice interface to work with server management. We will be using Ansible Tower to deploy and manage web servers in Windows and Linux environments. Are you ready to take the Driver seat and experience the Red Hat Ansible Tower test drive?

Deployment Architecture



Getting Started

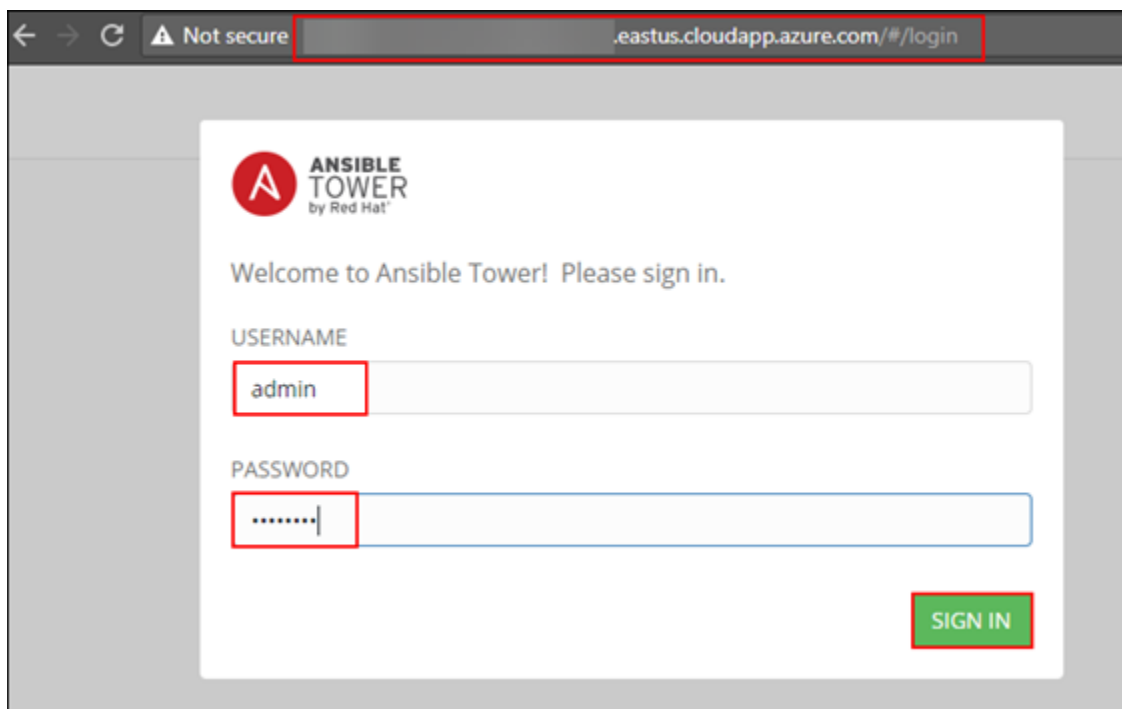
Once you have signed up/in, you will see the test drive launch screen. Once the test drive is deployed, you will receive the access information via email and also provided in the access information.

Lab 1: Getting Started with Ansible Tower

Access Ansible tower

During this lab, we are going to log in to Tower using the Ansible Tower DNS Name and other login details received via email or provided in the access information, browse to the Tower interface at: <http://<TowerDNSName>>

Log in using the Tower username and password details as provided. The default username set during installation is 'admin'. Password will be provided in the email.



Once we sign in, we will be directed to the Tower Dashboard page.

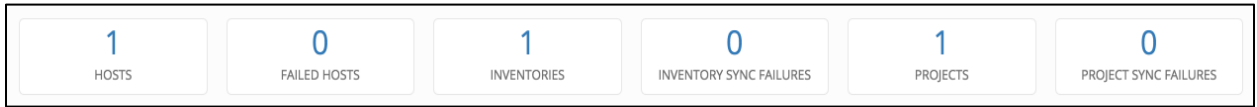
Tower Dashboard

The Tower Dashboard offers a friendly graphical framework for your IT orchestration needs. Across the top-left side of the Tower Dashboard, users can quickly navigate to their **Projects**, **Inventories**, **Job Templates**, and **Jobs**.

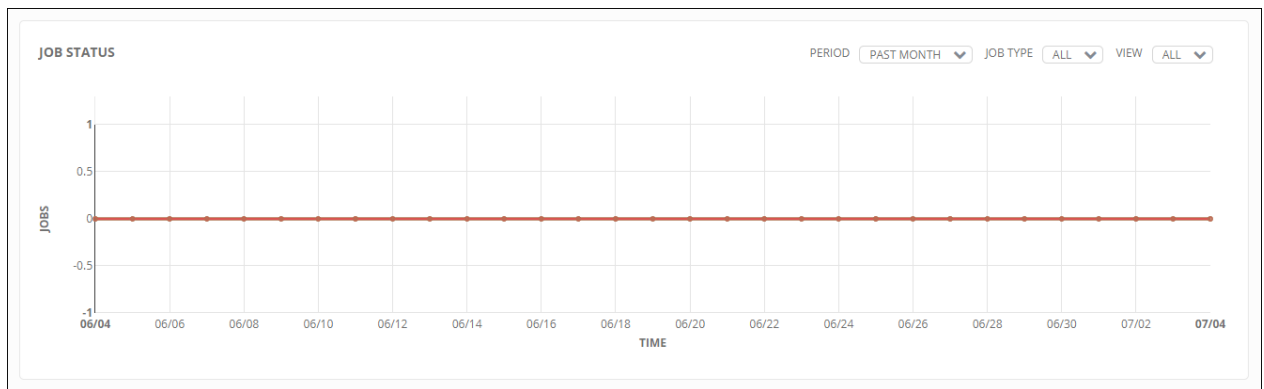
Across the top-right side of this interface, administrators can access the tools they need to configure organizations, users, groups, and permissions as well as view related documentation, access portal mode, and log out.




At the top of the Dashboard is a summary of your hosts, inventories, and projects. Each of these is linked to the corresponding object in Tower, for easy access.

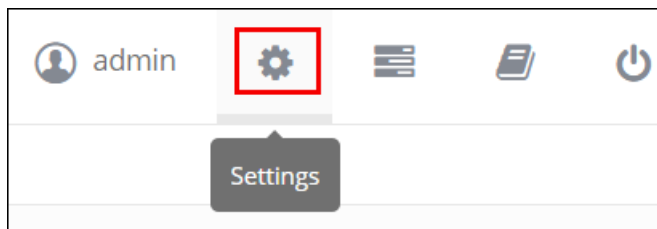


On the main Tower Dashboard screen, a summary appears listing your current **Job Status**. Also, available for review are summaries of **Recently Used Job Templates** and **Recently Run Jobs**.




Add Credentials

To enter the Settings Menu screen for Ansible Tower, click the  button. This screen allows you to create your organizations, add credentials, add users and teams, schedule management jobs, modify your Tower's configuration, and more. You can also view your license from the Settings Menu's 'View Your License' link.



Credentials are utilized by Tower for authentication when launching Jobs against machines, synchronizing with inventory sources, and importing project content from a version control system.

Tower credentials are imported and stored encrypted in Tower, and are not retrievable in plain text on the command line by any user. Once a password or key has been entered into the Tower interface, it is encrypted and inserted into the Tower database, and cannot be retrieved from Tower. We can grant users and teams the ability to use these credentials, without exposing the credential to the user. If we have a user move to a different team or leave the organization, you don't have to re-key all our systems just because that credential was available in Tower.

Now, we will navigate to Credentials, by clicking on the Setting () button on the top right of the dashboard menu.

We will add new credentials in Ansible Tower for the Linux Virtual Machine as well as Windows Virtual Machine:

1. Click the **+ ADD** button located in the upper right corner of the **Credentials** screen.
2. Enter the following details:
 - **Name:** **LinuxVmCredential**
 - **Description:** **Credentials for Linux VM**
 - **Organization:** **default**
 - **Type:** **Machine**
 - **Type details:**
 - a. **Username:** (Enter the **vmUsername** for **VM** received via **email**)
 - b. **Password:** (Enter the **vmPassword** for **VM** received via **email**)
 - c. **Privilege escalation:** **SUDO**
 - d. **Privilege escalation Password:** (Provide the **vmPassword** for **VM** you received via **email**)

The screenshot shows the 'CREATE CREDENTIAL' form in the Red Hat Ansible Tower interface. The form is titled 'CREATE CREDENTIAL' and has two tabs: 'DETAILS' (selected) and 'PERMISSIONS'. The form contains several input fields, some of which are highlighted with red boxes: 'NAME' (LinuxVmCredential), 'DESCRIPTION' (Credentials for Linux VM), 'TYPE' (Machine), 'USERNAME' (redacted), 'PASSWORD' (SHOW button and redacted text), 'PRIVILEGE ESCALATION' (Sudo), and 'PRIVILEGE ESCALATION PASSWORD' (SHOW button and redacted text). The form also includes checkboxes for 'Ask at runtime?' and a search bar for 'ORGANIZATION'.

3. Click **Save** when done.
4. Again, click the **+ ADD** button located in the upper right corner of the **Credentials** screen.
5. Enter the given details into the following fields:
 - **Name:** **WindowsVmCredential**
 - **Description:** **Credentials for Windows VM**
 - **Organization:** **default**
 - **Type:** **Machine**
 - **Type details:**

- a. *Username:* (Enter the **vmUsername** for **VM** received via **email**)
- b. *Password:* (Enter the **vmPassword** for **VM** received via **email**)

The screenshot shows the 'WindowsVmCredential' configuration page in Red Hat Ansible Tower. The page has a top navigation bar with 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. The user is logged in as 'admin'. The breadcrumb trail is 'SETTINGS / CREDENTIALS / WindowsVmCredential'. The main form has two tabs: 'DETAILS' (selected) and 'PERMISSIONS'. The form contains the following fields:

- * NAME:** 'WindowsVmCredential' (highlighted with a red box)
- DESCRIPTION:** 'Credentials for Windows VM' (highlighted with a red box)
- ORGANIZATION:** A search bar with a magnifying glass icon.
- * TYPE:** A dropdown menu with 'Machine' selected.
- TYPE DETAILS:**
 - USERNAME:** A text input field (highlighted with a red box).
 - PASSWORD:** A text input field with a 'SHOW' button and masked characters (highlighted with a red box).
 - PRIVATE KEY PASSPHRASE:** A text input field with a 'SHOW' button (highlighted with a red box).
- PRIVILEGE ESCALATION:** A dropdown menu with 'Choose a privilege escalation' selected.
- VAULT PASSWORD:** A text input field with a 'SHOW' button.

6. Click **Save** when done.

Add Projects

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

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

Now you will create a new project for Linux Machines by clicking on PROJECTS on top of the dashboard menu:

1. Then, click the **+ ADD** button, which launches the **Create Project** dialog.
2. Enter the given details into the following fields:
 - **Name:** **LightBulb**
 - **Description:** **Project with Playbook examples**
 - **Organization:** **default**
 - **SCM Type:** **Git**
 - **Source details:**
 - a. **SCM URL:** <https://github.com/ansible/lightbulb>

The screenshot shows the 'LightBulb' project configuration page in Red Hat Ansible Tower. The page has a top navigation bar with 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. The user is logged in as 'admin'. The breadcrumb is 'PROJECTS / LightBulb'. The form has three tabs: 'DETAILS', 'PERMISSIONS', and 'NOTIFICATIONS'. The 'DETAILS' tab is selected. The form fields are as follows:

- * NAME:** LightBulb
- DESCRIPTION:** Project with Playbook examples
- * ORGANIZATION:** Default
- * SCM TYPE:** Git
- SOURCE DETAILS:**
 - * SCM URL:** https://github.com/ansible/lightbulb
 - SCM BRANCH:**
 - SCM CREDENTIAL:**
- SCM UPDATE OPTIONS:**
 - ☐ Clean
 - ☐ Delete on Update
 - ☐ Update on Launch

At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

- Click **Save** when done.

Now you will create another new project for Windows Machines by clicking on **PROJECTS** on top of the dashboard menu:

- Again, click the **+ ADD** button, which launches the **Create Project** dialog.
- Enter the given details into the following fields:
 - Name:** **WindowsProject**
 - Description:** **Project with Windows Playbooks**
 - Organization:** **default**
 - SCM Type:** **Git**
 - Source details:**
 - SCM URL:** <https://github.com/SpektraSystems/ansible-testdrive>

The screenshot shows the 'NEW PROJECT' form in Red Hat Ansible Tower. The form is titled 'NEW PROJECT' and has three tabs: 'DETAILS', 'PERMISSIONS', and 'NOTIFICATIONS'. The 'DETAILS' tab is active. The form contains several fields: 'NAME' (WindowsProject), 'DESCRIPTION' (Project with Windows Playbooks), 'ORGANIZATION' (Default), 'SCM TYPE' (Git), 'SCM URL' (https://github.com/SpetraSystems/ansible), 'SCM BRANCH' (empty), 'SCM CREDENTIAL' (empty), and 'SCM UPDATE OPTIONS' (Clean, Delete on Update, Update on Launch). The 'SAVE' button is highlighted in green.

6. Click **Save** when done.

Add Inventory and Test Connectivity

An Inventory is a collection of hosts against which jobs may be launched, the same as an Ansible inventory file. Inventories are divided into groups and these groups contain the actual hosts. Groups may be sourced manually, by entering host names into Tower, or from one of Ansible Tower's supported cloud providers.

Now, you will create a new Inventory and add Linux and Windows Machines as hosts by first clicking on **INVENTORIES** on top of the dashboard menu:

1. Then, click the **+ ADD** button, which launches the **Create Inventory** dialog.
2. Enter the given details into the following fields:
 - **Name:** Agent VM Inventory
 - **Description:** Inventory with Agent hosts
 - **Organization:** Default

TOWER PROJECTS INVENTORIES TEMPLATES JOBS admin

INVENTORIES / CREATE INVENTORY

* NAME: Agent VM Inventory DESCRIPTION: Inventory with Agent hosts * ORGANIZATION: Default

VARIABLES ? [YAML] [JSON]

1 ---

CANCEL SAVE

3. Click on **Save** when done.

4. Now, click the **+ ADD** button in GROUPS, which launches the **Create Group** dialog.

5. Enter the given details into the following fields:

- *Name:* **web**
- *Description:* **Group for Linux Hosts**
- *Source:* **Manual**

INVENTORIES / Agent VM Inventory / web

web

DETAILS NOTIFICATIONS

* NAME: web DESCRIPTION: Group for Linux Hosts SOURCE: Manual

VARIABLES ? [YAML] [JSON]

1 ---

CANCEL SAVE

6. Click on **Save** when done.

7. Now, click on the created group **web**

8. Then, click the **+ ADD** button in HOSTS, which launches the **Create Host** dialog.

9. Enter the given details into the following fields:

- *Host Name:* (Enter the **DNS Name** for **Linux VM** received via **email**)
- *Description:* **Linux Host**

10. Click on **Save** when done.

11. Now go back to Agent VM Inventory page by clicking on Agent VM Inventory

[INVENTORIES](#) / [Agent VM Inventory](#) / web

12. Now, click the  button in GROUPS, which launches the **Create Group** dialog.

13. Enter the given details into the following fields:

- *Name:* **windowsVM**
- *Description:* **Group for Windows Hosts**
- *Source:* **Manual**
- *Variables:* Add the following lines below ---
 ansible_connection: winrm
 ansible_winrm_server_cert_validation: ignore

INVENTORIES / Agent VM Inventory / CREATE GROUP

CREATE GROUP

DETAILS

NOTIFICATIONS

* NAME

windowsVM

DESCRIPTION

Group for Windows Hosts

SOURCE

Manual

VARIABLES

YAML

JSON

```

1 ---
2 ansible_connection: winrm
3 ansible_winrm_server_cert_validation: ignore
4

```

CANCEL

SAVE

14. Click on **Save** when done.

15. Now, click on the created group **windowsVM**

16. Then, click the **+ ADD** button in HOSTS, which launches the **Create Host** dialog.

17. Enter the given details into the following fields:

- **Host Name:** (Enter the **DNS Name** for **Windows VM** received via **email**)
- **Description:** **Windows Host**

INVENTORIES / Agent VM Inventory / windowsVM / CREATE HOST

CREATE HOST

ON

* HOST NAME

DESCRIPTION

Windows Host

VARIABLES

YAML

JSON

```

1 ---

```

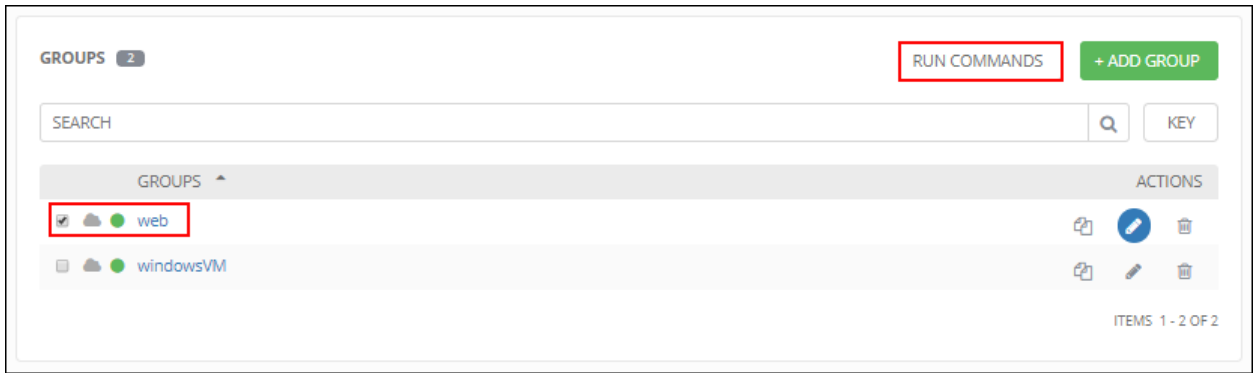
CANCEL

SAVE

18. Click on **Save** when done. Now we will test the connectivity to the added hosts.

19. Now go back to Agent VM Inventory page by clicking on Agent VM Inventory

20. Select **web** group and click on **Run Commands**, which launches the Execute Command dialog



21. Enter the given details into the following fields:

- Module: **ping**
- Machine Credential: **LinuxVmCredential**

The screenshot shows the 'EXECUTE COMMAND' form. The 'MODULE' dropdown is set to 'ping'. The 'MACHINE CREDENTIAL' dropdown is set to 'LinuxVmCredential'. The 'LIMIT' dropdown is set to 'web'. The 'FORKS' dropdown is set to '0'. The 'VERBOSITY' dropdown is set to '0 (Normal)'. The 'EXTRA VARIABLES' section is empty. At the bottom right, there are 'RESET' and 'LAUNCH' buttons. The 'LAUNCH' button is highlighted with a red box.

22. Click on **Launch** when done, which launches the Job Results page.

JOBS / ping

RESULTS

| | |
|-----------------|---|
| NAME | ping |
| STATUS | ● Successful |
| STARTED | 5/7/2017 20:25:18 |
| FINISHED | 5/7/2017 20:25:22 |
| ELAPSED | 4.407 seconds |
| INVENTORY | Agent VM Inventory |
| CREDENTIAL | LinuxVmCredential |
| LAUNCHED BY | admin |
| FORKS | 0 |
| LIMIT | web |
| VERBOSITY | 0 |
| EXTRA VARIABLES | 1 --- |

STANDARD OUT

```
SSH password:
SUDO password[defaults to SSH password]:
.eastus.cloudapp.azure.com | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

We can see status of the Job in that page, and if that job was successful, status will be **Successful** with a green icon next to it.

23. Now go back to Agent VM Inventory page by clicking on INVENTORIES from the top menu and then click on Agent VM Inventory.
24. Select **windowsVM** group and click on **Run Commands**, which launches the Execute Command dialog

GROUPS 2

RUN COMMANDS **+ ADD GROUP**

SEARCH

| GROUPS | ACTIONS |
|--|--|
| <input type="checkbox"/> ● web | <input type="button" value="Q"/> <input type="button" value="KEY"/> <input type="button" value="X"/> |
| <input checked="" type="checkbox"/> ● windowsVM | <input type="button" value="Q"/> <input type="button" value="KEY"/> <input type="button" value="X"/> |

ITEMS 1 - 2 OF 2

25. Enter the given details into the following fields:
 - Module: Select **win_ping**
 - Machine Credential: **WindowsVmCredential**

INVENTORIES / Agent VM Inventory / RUN COMMAND

EXECUTE COMMAND

* MODULE

win_ping

ARGUMENTS

LIMIT

windowsVM

* MACHINE CREDENTIAL

WindowsVmCredential

ENABLE PRIVILEGE ESCALATION

* VERBOSITY

0 (Normal)

* FORKS

0

EXTRA VARIABLES

YAML JSON

```
1 ---
```

RESET

LAUNCH

26. Click on **Launch** when done, which launches the Job Results page.

JOBS / win_ping

RESULTS

| | |
|-----------------|---------------------------|
| NAME | win_ping |
| STATUS | ● Successful |
| STARTED | 5/7/2017 16:57:07 |
| FINISHED | 5/7/2017 16:57:21 |
| ELAPSED | 14.226 seconds |
| INVENTORY | Agent VM Inventory |
| CREDENTIAL | WindowsVmCredential |
| LAUNCHED BY | admin |
| FORKS | 0 |
| LIMIT | windowsVM |
| VERBOSITY | 0 |
| EXTRA VARIABLES | <pre>1 ---</pre> |

STANDARD OUT

```
SSH password: .eastus.cloudapp.azure.com | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

We can see status of the Job in that page, and if that job was successful, status will be **Successful** with a green icon next to it.

END OF LAB

Lab 2: Deploy Web Servers on Linux VM using Ansible Playbook

Deploy Nginx Web Server


A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. Job templates also encourage the reuse of Ansible playbook content and collaboration between teams. While the REST API allows for the execution of jobs directly, Tower requires that you first create a job template.

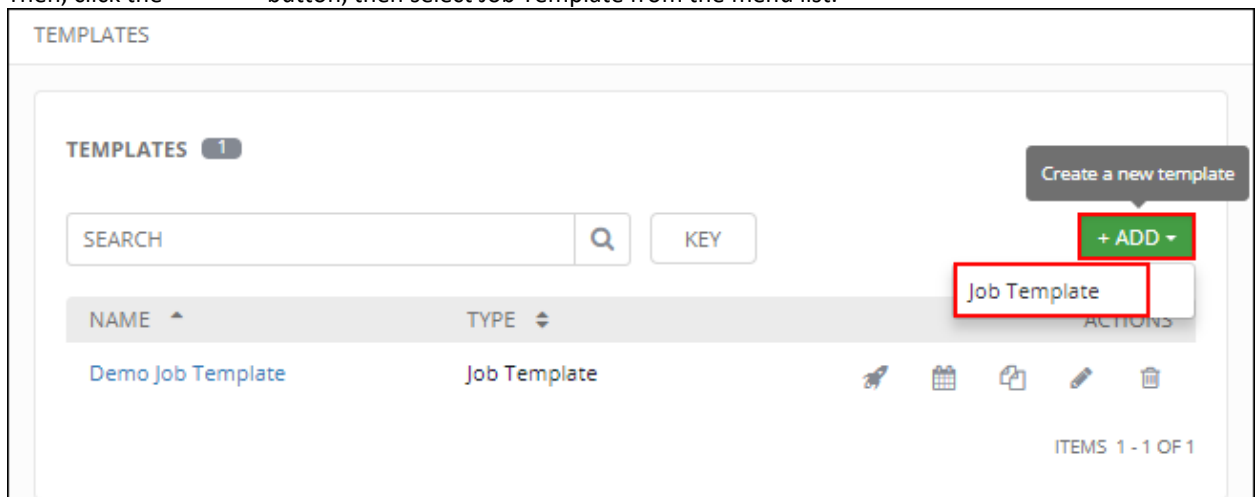
We can access Job Template page by clicking on TEMPLATES on the menu on top of the dashboard.



This menu opens a list of the job templates that are currently available. The job template list may be sorted and searched by **Name** or **Description**. The **Templates** tab also enables the user to launch, schedule, modify, and remove a job template.

Now, you will create a new template for deploying Nginx Web Server by first clicking on TEMPLATES on top of the dashboard menu:

1. Then, click the  button, then select Job Template from the menu list.



2. Enter the given details into the following fields:

- **NAME:** Install Nginx
- **JOB TYPE:** Run
- **INVENTORY:** Agent VM Inventory
- **PROJECT:** LightBulb
- **PLAYBOOK:** examples/nginx-basic-playbook/site.yml
- **MACHINE CREDENTIAL:** LinuxVmCredential
- **LIMIT:** web
- **Enable Privilege Escalation:** Tick

Install Nginx

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

* NAME:

DESCRIPTION:

* JOB TYPE:

☐ Prompt on launch

* INVENTORY:

* PROJECT:

* PLAYBOOK:

☐ Prompt on launch

* MACHINE CREDENTIAL:

CLOUD CREDENTIAL:

NETWORK CREDENTIAL:

☐ Prompt on launch

FORKS:

LIMIT:

☐ Prompt on launch

* VERBOSITY:

JOB TAGS:

SKIP TAGS:

OPTIONS

- ☒ Enable Privilege Escalation
- ☐ Allow Provisioning Callbacks
- ☐ Enable Concurrent Jobs

3. Click on **Save** when done.

Saving the template does not exit the job template page but remains on the Job Template Details view for further editing, if necessary. The **Details** tab of a saved job allows you to review, edit, and add a survey (if the job type is not a scan).

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.

TEMPLATES 2

SEARCH KEY

| NAME | TYPE | ACTIONS |
|-------------------|--------------|---------|
| Demo Job Template | Job Template | |
| Install Nginx | Job Template | |

ITEMS 1 - 2 OF 2

Now, we will launch a Job Template by clicking the launch icon. We will be directed to the Job Results Page. The Jobs page shows details of all the tasks and events for that playbook run.

The screenshot displays the Red Hat Ansible Tower interface. On the left, the 'DETAILS' panel shows the job status as 'Successful' (highlighted with a red box). The job was started on 5/7/2017 at 22:34:13 and finished at 22:34:38. The template used is 'Install Nginx', and the job type is 'Run'. The job was launched by 'admin' and is associated with the 'Agent VM Inventory'. The project is 'LightBulb', and the revision is 'd319ec4103c2816309879fd0a9f3e2000a555f4'. The playbook is 'examples/nginx-basic-playbook/site.yml', and the machine credential is 'LinuxVmCredential'. There are 0 forks, a limit of 'web', and a verbosity of 0 (Normal). The 'EXTRA VARIABLES' section is empty.

The main panel shows the execution details for the 'Install Nginx' job. It lists the tasks and their status:

- 1 SSH password:
- 2 SUDO password[defaults to SSH password]:
- 3
- 4 PLAY [install and start nginx with wsgi] *****
- 5
- 6 TASK [Gathering Facts] *****
- 7 Ok: [redacted.eastus.cloudapp.azure.com]
- 8
- 9 TASK [nginx packages are present] *****
- 10 changed: [redacted.eastus.cloudapp.azure.com] => (11 u'python-devel', u'gcc')]
- 11
- 12 TASK [uwsgi package is present] *****
- 13 Ok: [redacted.eastus.cloudapp.azure.com]
- 14
- 15 TASK [latest default.conf is present] *****
- 16 Ok: [redacted.eastus.cloudapp.azure.com]
- 17
- 18 TASK [latest index.html is present] *****
- 19 Ok: [redacted.eastus.cloudapp.azure.com]
- 20
- 21 TASK [nginx service is started and enabled] *****
- 22 Ok: [linagentyomfjqagpwm1.eastus.cloudapp.azure.com]
- 23

We can see the status as successful for Job to deploy Nginx Web Server in Linux VM.

Now to verify that Nginx is installed on Linux Virtual Machine and is accessible through Public DNS Name, we will open a browser and navigate to the Linux VM DNS name and check if the Ansible Tower is coming on the page as shown below.



Remove Nginx Web Server

Now, you will create a new template for removing Nginx Web Server by first clicking on TEMPLATES on top of the dashboard menu:

1. Then, click the **+ ADD** button, then select Job Template from the menu list.
2. Enter the given details into the following fields:
 - **NAME:** Remove Nginx
 - **JOB TYPE:** Run
 - **INVENTORY:** Agent VM Inventory
 - **PROJECT:** LightBulb
 - **PLAYBOOK:** examples/nginx-remove-playbook/site.yml
 - **MACHINE CREDENTIAL:** LinuxVmCredential
 - **LIMIT:** web
 - **Enable Privilege Escalation:** Tick

NEW JOB TEMPLATE

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

* NAME: DESCRIPTION:

* JOB TYPE: Prompt on launch

* INVENTORY: Prompt on launch

* PROJECT: Prompt on launch

* PLAYBOOK:

* MACHINE CREDENTIAL: Prompt on launch

CLOUD CREDENTIAL:

NETWORK CREDENTIAL:

FORKS: Prompt on launch

LIMIT: Prompt on launch

* VERBOSITY:

JOB TAGS: Prompt on launch

SKIP TAGS: Prompt on launch

OPTIONS

- ☒ Enable Privilege Escalation
- ☐ Allow Provisioning Callbacks
- ☒ Enable Concurrent Jobs

3. Click on **Save** when done.

Saving the template does not exit the job template page but remains on the Job Template Details view for further editing, if necessary. The **Details** tab of a saved job allows you to review, edit, and add a survey (if the job type is not a scan).

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.

TEMPLATES 3

SEARCH

| NAME | TYPE |
|-------------------|--------------|
| Demo Job Template | Job Template |
| Install Nginx | Job Template |
| Remove Nginx | Job Template |

Now, we will launch a Job Template by clicking the launch icon. We will be directed to the Job Results Page. The Jobs page shows details of all the tasks and events for that playbook run.

The screenshot displays the Red Hat Ansible Tower interface. On the left, the 'DETAILS' panel shows the job status as 'Successful' (highlighted with a red box). The job is titled 'Remove Nginx' and was launched by 'admin' on '5/7/2017 22:34:43'. The inventory used is 'Agent VM Inventory' and the project is 'LightBulb'. The playbook is 'examples/nginx-remove-playbook/site.yml' and the machine credential is 'LinuxVmCredential'. The job type is 'Run' and the limit is 'web'. The verbosity is set to '0 (Normal)'. The 'EXTRA VARIABLES' section is empty.

On the right, the 'Remove Nginx' job execution details are shown. The top bar indicates 'PLAYS: 1', 'TASKS: 5', 'HOSTS: 1', and 'ELAPSED'. The execution log shows the following tasks:

- PLAY [removes nginx with wsgi]
- TASK [Gathering Facts]
- TASK [nginx service is stopped]
- TASK [nginx package is absent]
- TASK [uwsgi package is absent]
- TASK [files created by nginx-simple are absent]

The log also shows the 'PLAY RECAP' section, indicating that the job was successful with 5 tasks, 3 changes, and 0 failures.

We can see the status as successful for Job to deploy Nginx Web Server in Linux VM.

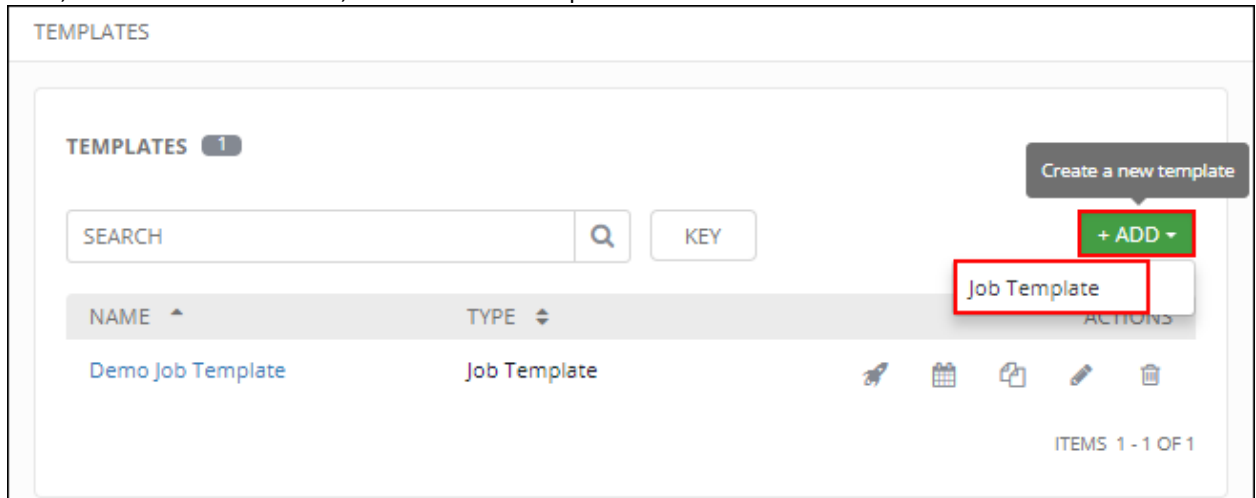
Now to verify that Nginx Webserver is removed from the Linux Virtual Machine and is not accessible through Public DNS Name, we will open a browser and navigate to the Linux VM DNS name and check if the page is unreachable



Deploy Apache Web Server

Now, you will create a new template for deploying Apache Web Server by first clicking on TEMPLATES on top of the dashboard menu:

1. Then, click the **+ ADD** button, then select Job Template from the menu list.



2. Enter the given details into the following fields:
 - **NAME:** Install Apache
 - **JOB TYPE:** Run
 - **INVENTORY:** Agent VM Inventory
 - **PROJECT:** LightBulb
 - **PLAYBOOK:** examples/apache-simple-playbook/site.yml
 - **MACHINE CREDENTIAL:** LinuxVmCredential
 - **LIMIT:** web
 - **Enable Privilege Escalation:** Tick

Install Apache

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

* NAME: DESCRIPTION:

* JOB TYPE:
 ☐ Prompt on launch

* INVENTORY: * PROJECT: * PLAYBOOK:
 ☐ Prompt on launch

* MACHINE CREDENTIAL: CLOUD CREDENTIAL: NETWORK CREDENTIAL:
 ☐ Prompt on launch

FORKS: LIMIT: * VERBOSITY:
 ☐ Prompt on launch

JOB TAGS: SKIP TAGS:
 ☐ Prompt on launch

OPTIONS:
 ☒ Enable Privilege Escalation
 ☐ Allow Provisioning Callbacks
 ☐ Enable Concurrent Jobs

3. Click on **Save** when done.

Saving the template does not exit the job template page but remains on the Job Template Details view for further editing, if necessary. The **Details** tab of a saved job allows you to review, edit, and add a survey (if the job type is not a scan).

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.

| TEMPLATES 4 | |
|-------------------|--------------|
| SEARCH | |
| NAME | TYPE |
| Demo Job Template | Job Template |
| Install Apache | Job Template |
| Install Nginx | Job Template |
| Remove Nginx | Job Template |

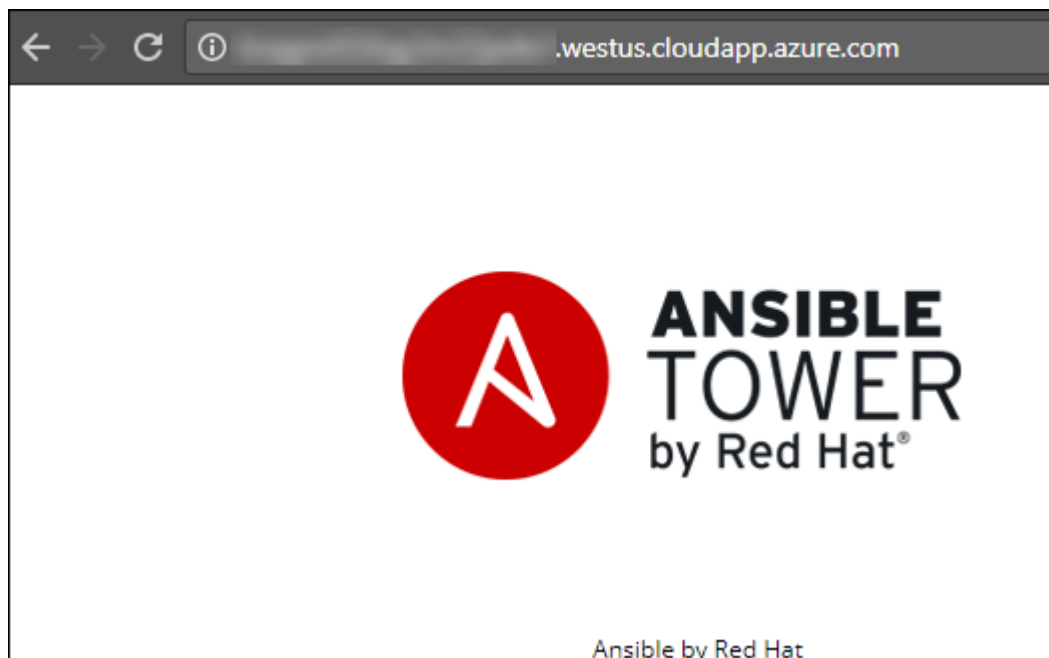
Now, we will launch a Job Template by clicking the launch icon. We will be directed to the Job Results Page. The Jobs page shows details of all the tasks and events for that playbook run.

The screenshot displays the Red Hat Ansible Tower web interface. On the left, the 'DETAILS' sidebar shows the job status as 'Successful' (highlighted with a red box). The job is titled 'Install Apache' and was launched by 'admin' on '5/7/2017 20:28:56'. The inventory used is 'Agent VM Inventory', and the project is 'LightBulb'. The playbook is 'examples/apache-simple-playbook/site.yml' using the 'LinuxVmCredential'.

The main panel shows the execution progress for 'Install Apache'. It lists tasks such as 'Gathering Facts', 'httpd package is present', 'latest httpd.conf file is present', and 'httpd is started'. The output shows that the package was installed and the service was started successfully. The final summary indicates 4 OK, 3 changed, and 0 unreachable or failed.

We can see the status as successful for Job to deploy apache Web Server in Linux VM.

Now to verify that apache is installed on Linux Virtual Machine and is accessible through Public DNS Name, we will open a browser and navigate to the Linux VM DNS name and check if the Ansible Tower is coming on the page as shown below.




END OF LAB

Lab 3: Manage Windows VM using Ansible Playbook

Create a Local User Account in Windows VM

Now, you will create a new template for creating a local User account in Windows VM by first clicking on TEMPLATES on top of the dashboard menu:

1. Then, click the  button, then select Job Template from the menu list.
2. Enter the given details into the following fields:
 - **NAME:** Create User
 - **JOB TYPE:** Run
 - **INVENTORY:** Agent VM Inventory
 - **PROJECT:** WindowsProject
 - **PLAYBOOK:** windows/create-user.yml
 - **MACHINE CREDENTIAL:** WindowsVmCredential
 - **LIMIT:** windowsVM

NEW JOB TEMPLATE

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

* NAME: DESCRIPTION:

* JOB TYPE:
☐ Prompt on launch

* INVENTORY: * PROJECT: * PLAYBOOK:
☐ Prompt on launch

* MACHINE CREDENTIAL: CLOUD CREDENTIAL: NETWORK CREDENTIAL:
☐ Prompt on launch

FORKS: LIMIT:
☐ Prompt on launch

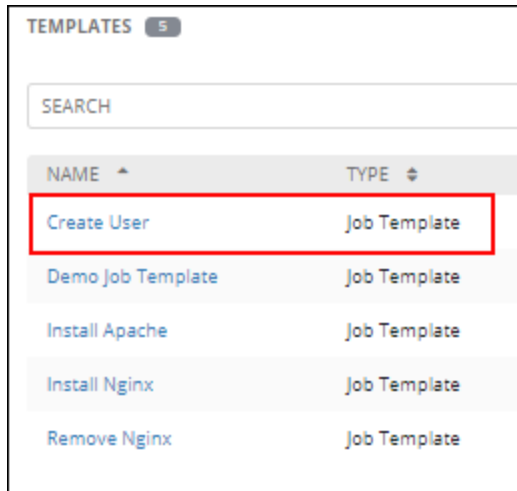
JOB TAGS: SKIP TAGS:
☐ Prompt on launch

* VERBOSITY:
☐ Enable Privilege Escalation
☐ Allow Provisioning Callbacks
☐ Enable Concurrent Jobs

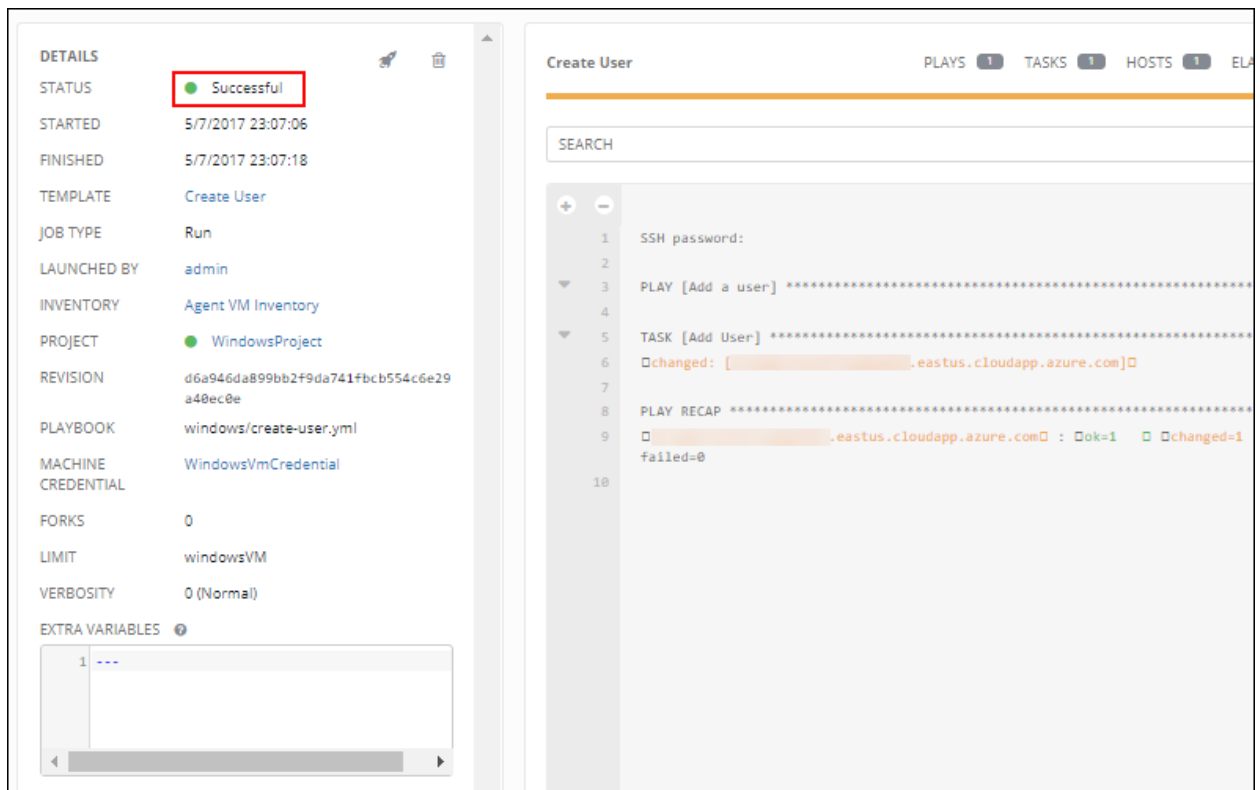
3. Click on **Save** when done.

The play create-user.yml has commands for creating a local user named 'ansible' with password "@ns1bl3"

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.



Now, we will launch a Job Template by clicking the launch icon. We will be directed to the Job Results Page. The Jobs page shows details of all the tasks and events for that playbook run.

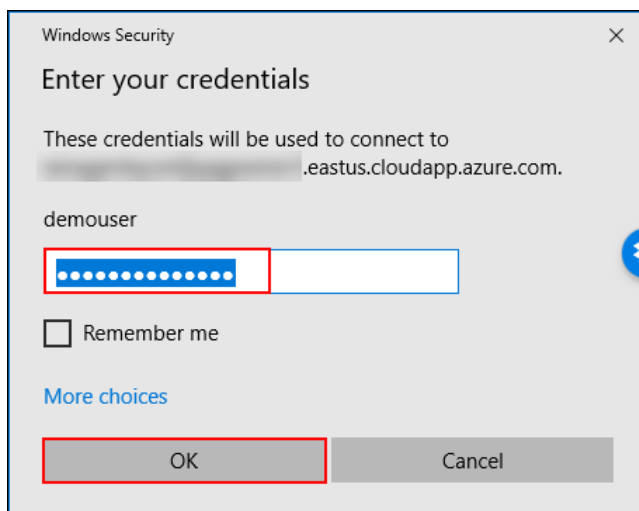
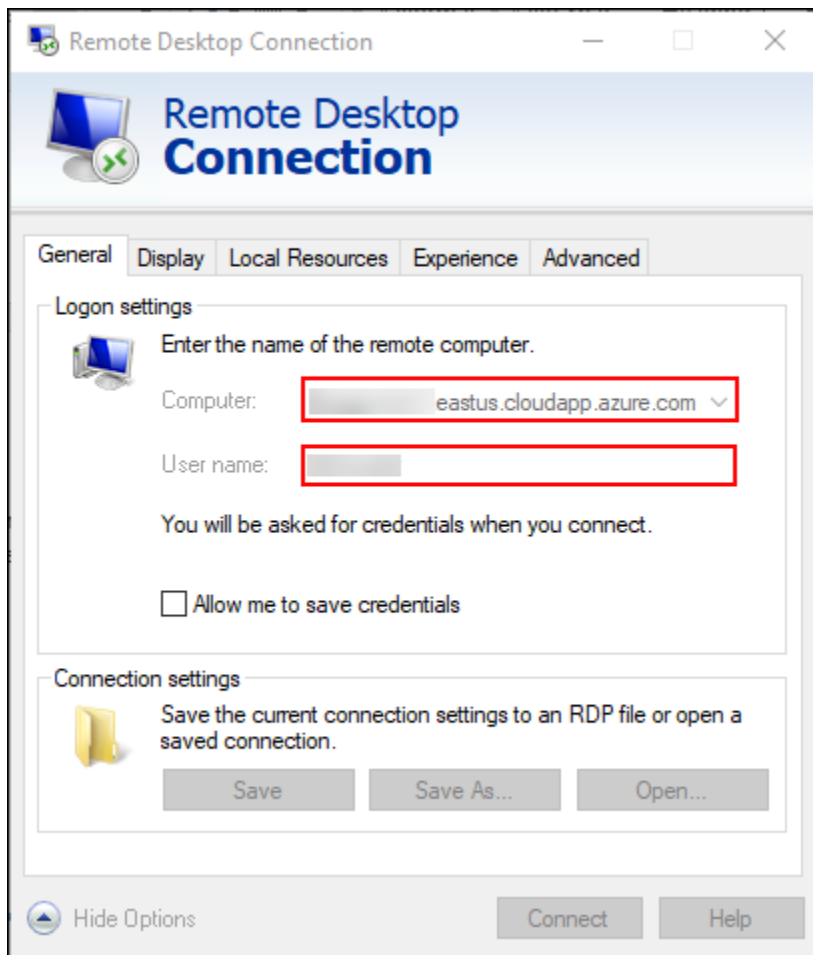


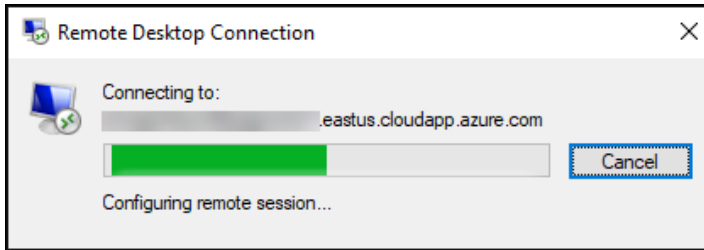
We can see the status as successful for Job to create a local user account in Windows VM.

Now to verify that a user account with username 'ansible' is created on Windows Virtual Machine, we will access the Virtual Machine using a RDP client and check.

Using RDP Client, DNS Name of Windows VM, username and password of VM we received via email, log in to the Windows VM.

From Windows, using the built-in RDP Client, we can connect like given below:



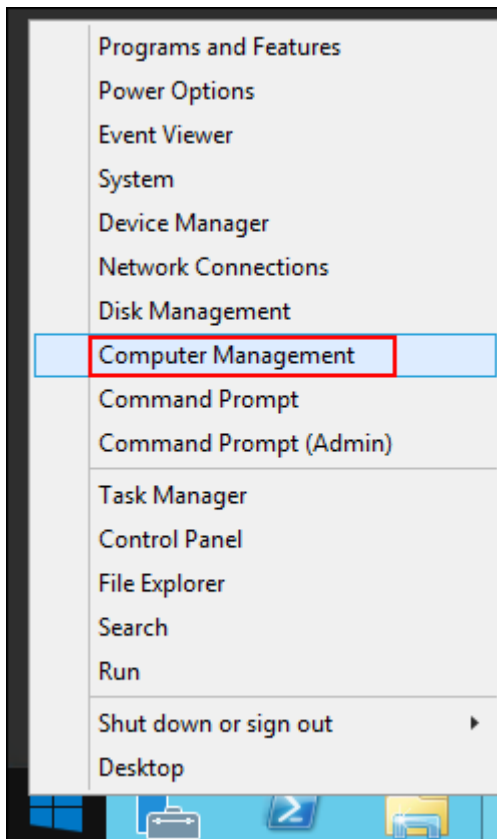


Once we are logged in to the virtual machine, we should go to the User accounts inside Control Panel.

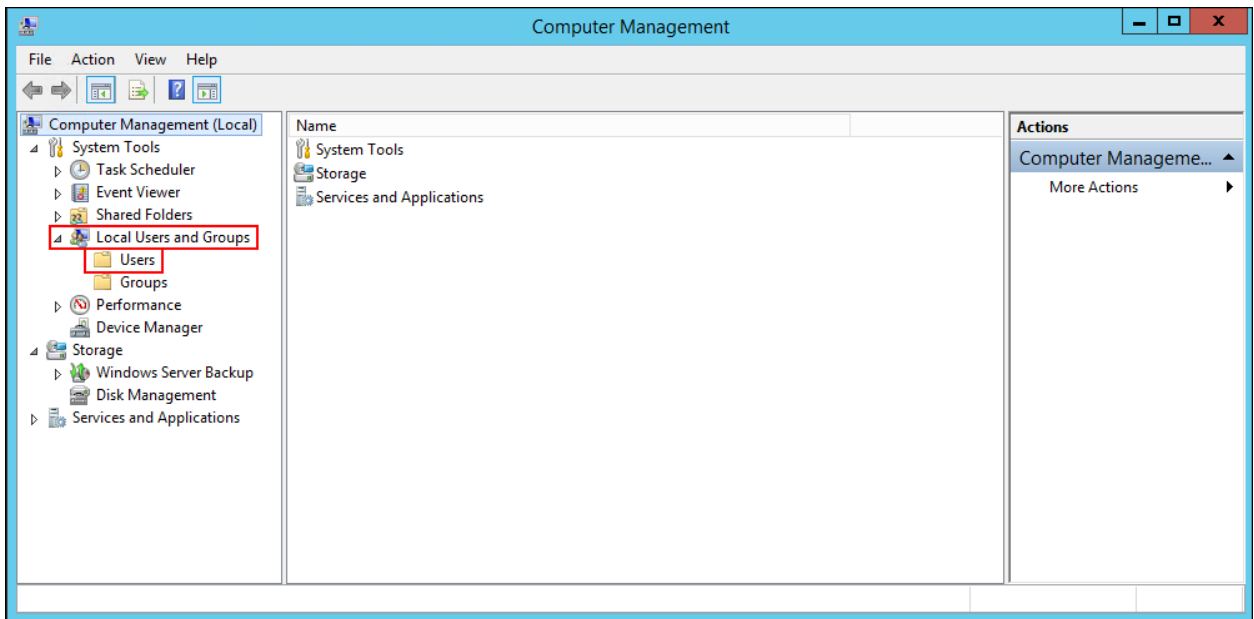
For that, we will Right Click on Windows button the bottom left of the windows vm.



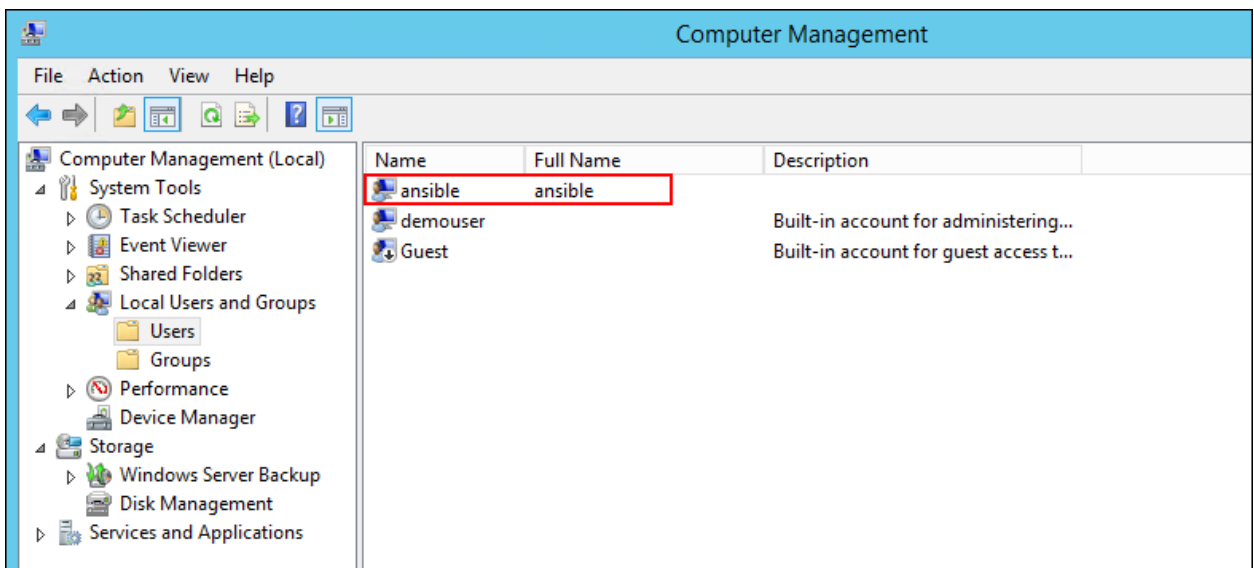
Then Select Computer Management from the options.



From the options listed, we should go to Local Users and Groups and select Users to view the User accounts present in the Virtual Machine.




In the Users page, we can see that the user `ansible` has been created.



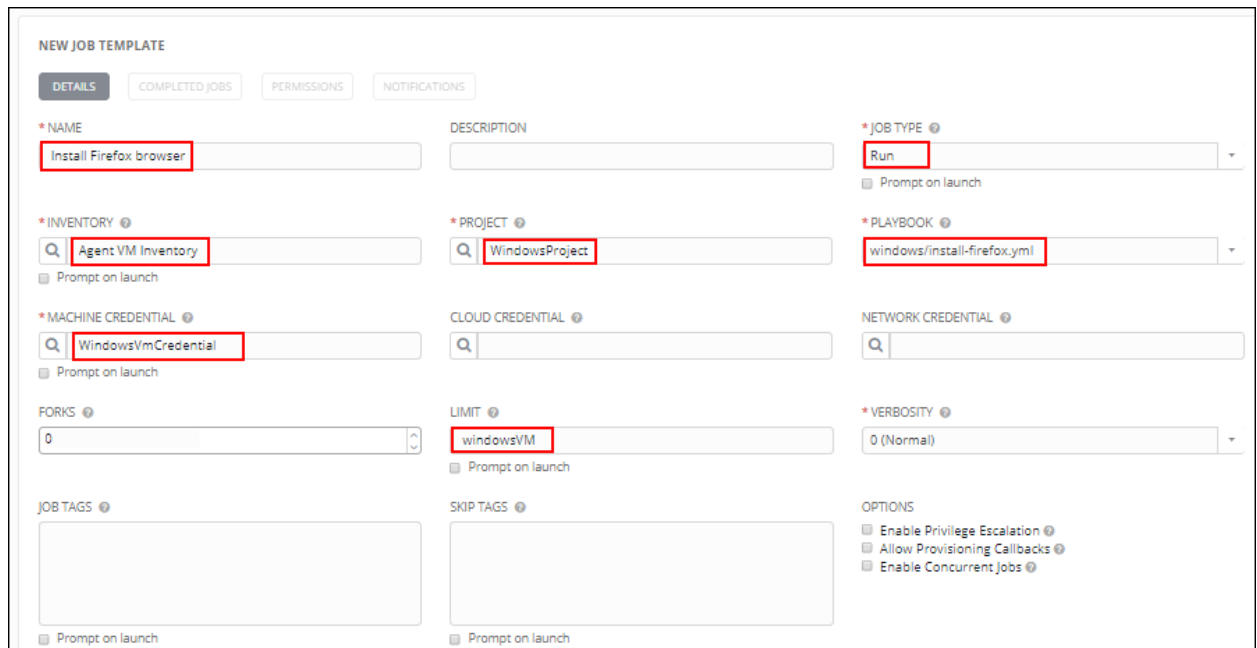
Install a Package

Now, you will create a new template for installing Firefox Browser by first clicking on **TEMPLATES** on top of the dashboard menu:

1. Then, click the  button, then select Job Template from the menu list.

2. Enter the given details into the following fields:

- **NAME:** Install Firefox browser
- **JOB TYPE:** Run
- **INVENTORY:** Agent VM Inventory
- **PROJECT:** WindowsProject
- **PLAYBOOK:** windows/install-firefox.yml
- **MACHINE CREDENTIAL:** WindowsVmCredential
- **LIMIT:** windowsVM



NEW JOB TEMPLATE

DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

* NAME: DESCRIPTION:

* JOB TYPE:
 ☐ Prompt on launch

* INVENTORY: * PROJECT: * PLAYBOOK:
 ☐ Prompt on launch

* MACHINE CREDENTIAL: CLOUD CREDENTIAL: NETWORK CREDENTIAL:
 ☐ Prompt on launch

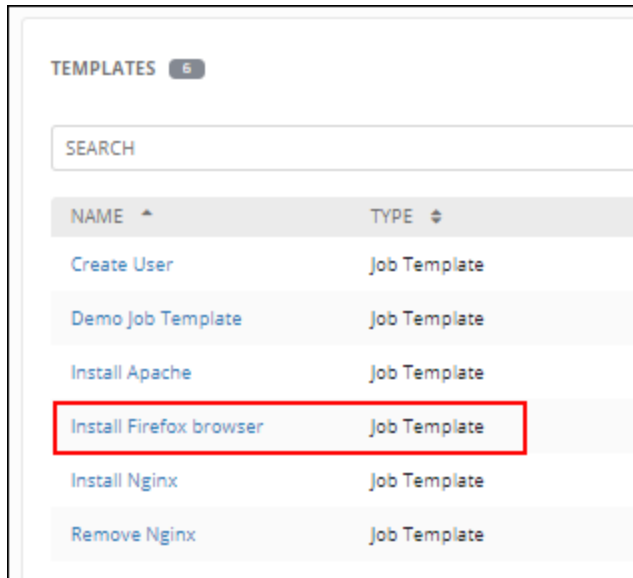
FORKS: LIMIT: * VERBOSITY:
 ☐ Prompt on launch

JOB TAGS: SKIP TAGS:
 ☐ Prompt on launch

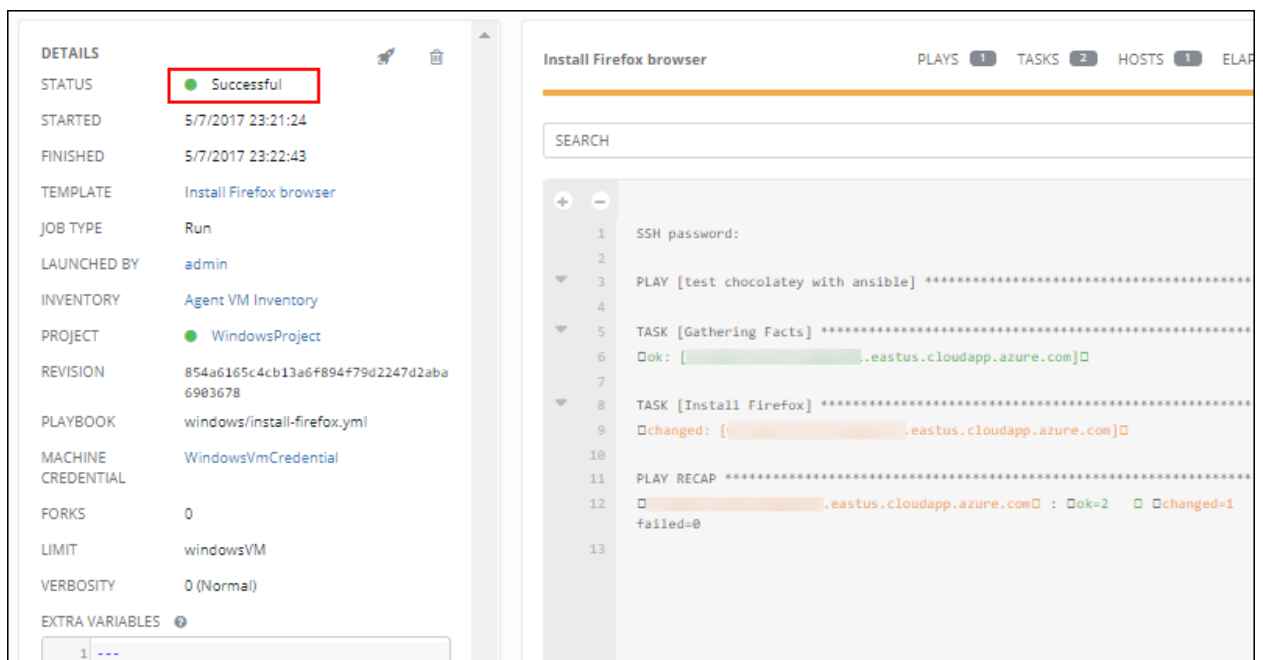
OPTIONS:
 ☐ Enable Privilege Escalation
 ☐ Allow Provisioning Callbacks
 ☐ Enable Concurrent Jobs

3. Click on **Save** when done.

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.



Now, we will launch a Job Template by clicking the launch icon. We will be directed to the Job Results Page. The Jobs page shows details of all the tasks and events for that playbook run.

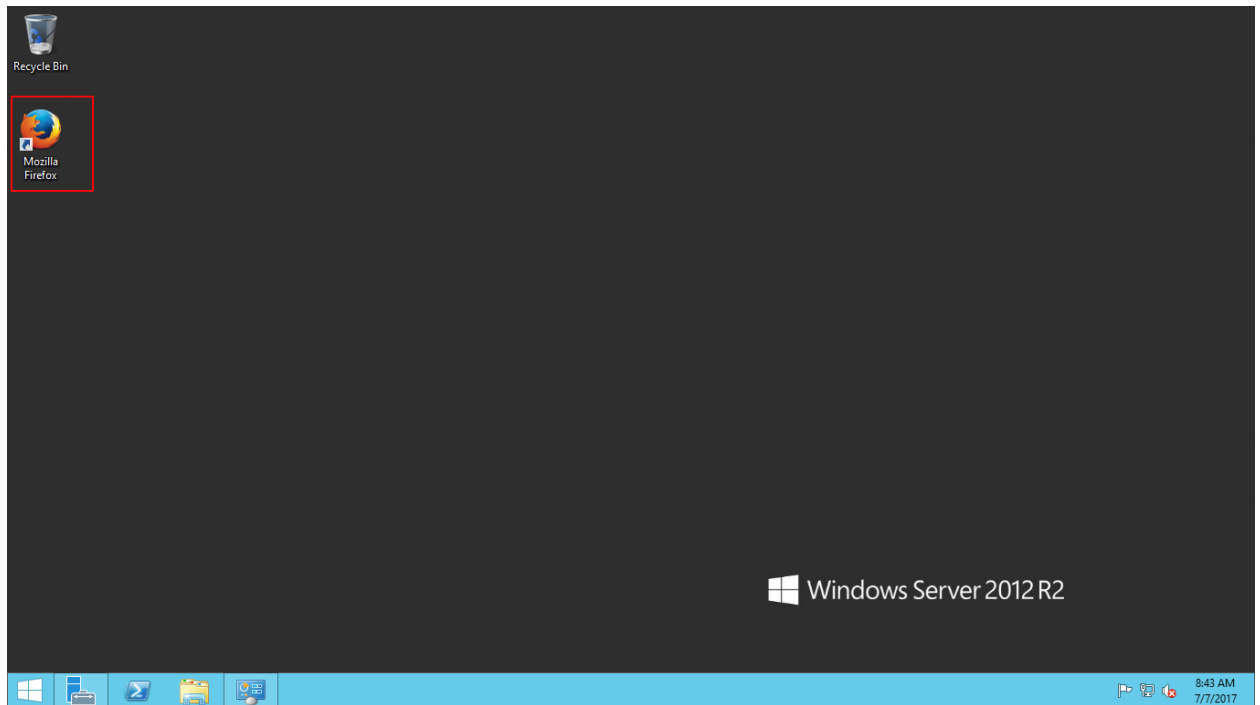


We can see the status as successful for Job to install Firefox Browser on the windows virtual machine.

Now to verify that Firefox is installed on the Windows Virtual Machine, we will access the Virtual Machine using a RDP client and check.

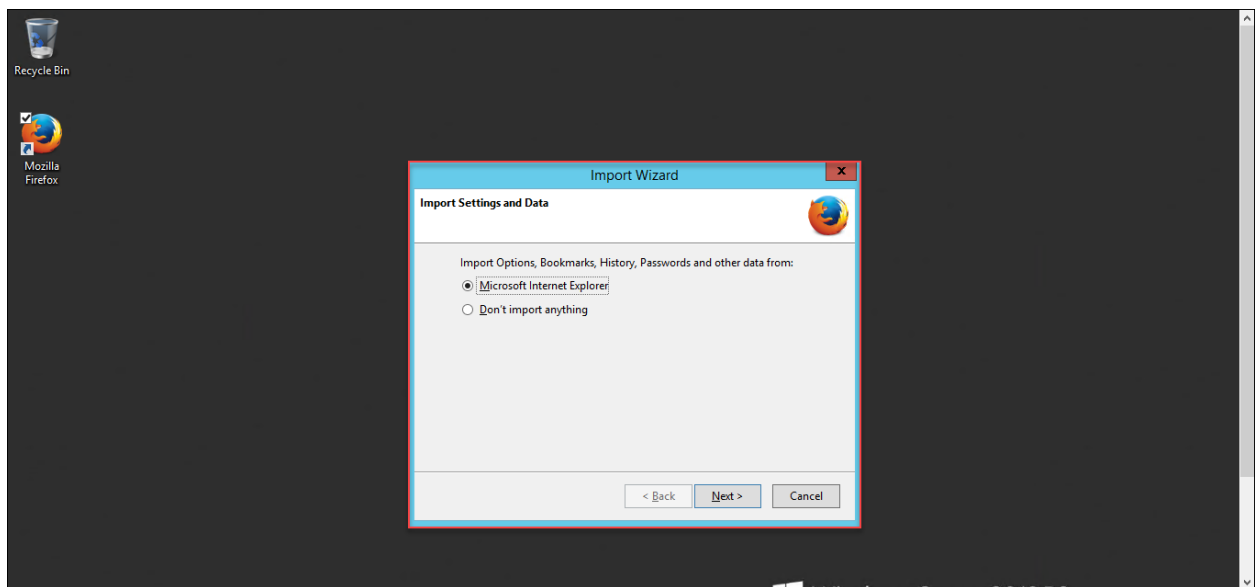
Using RDP Client, DNS Name of Windows VM, username and password of VM we received via email, we will log in to the Windows VM.

Once we are logged in to the virtual machine, we can see the Mozilla Firefox browser icon on the Desktop Home Screen.

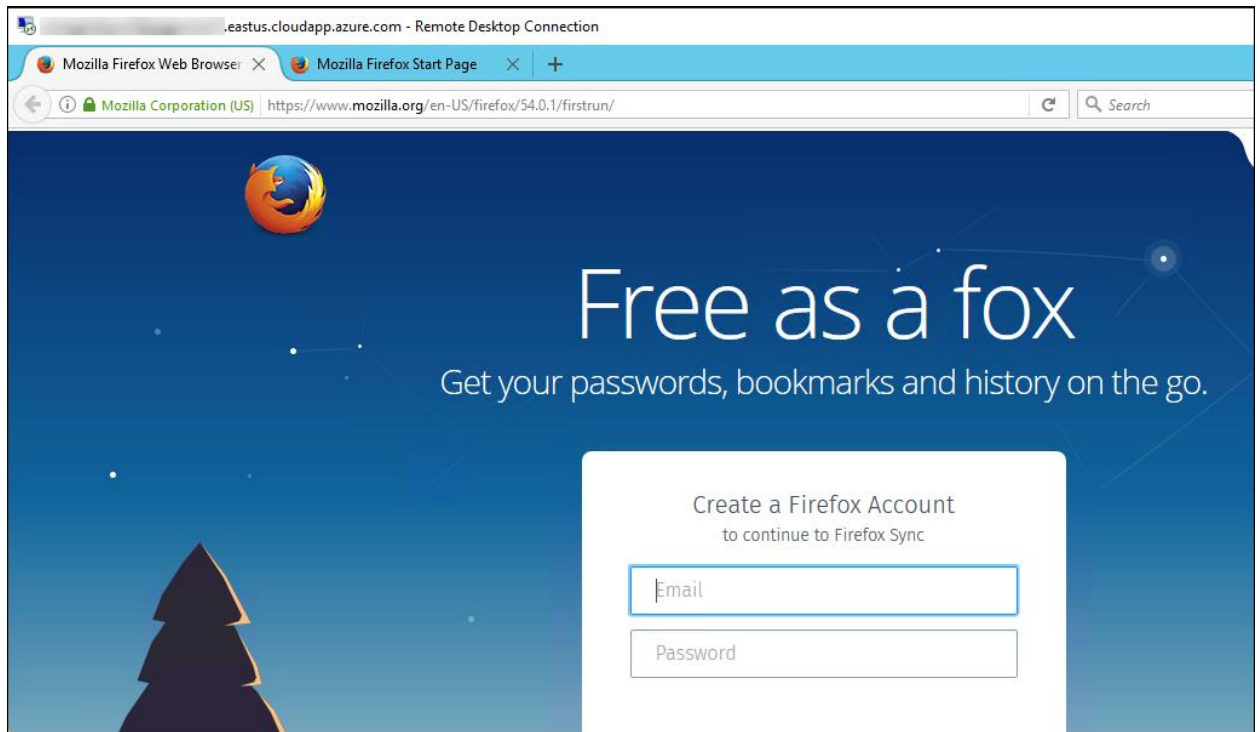


To verify that browser was properly installed, we will open the browser and check.

An Import Wizard will come up to import any existing bookmarks.




Once the initial configuration is done, the browser will be launched.



Deploy IIS Web Server

Now, we will create a new template for deploying IIS Web Server by first clicking on TEMPLATES on top of the dashboard menu:

1. Then, click the  button, then select Job Template from the menu list.
2. Enter the given details into the following fields:
 - **NAME:** Install IIS
 - **JOB TYPE:** Run
 - **INVENTORY:** Agent VM Inventory
 - **PROJECT:** WindowsProject
 - **PLAYBOOK:** windows/enable -iis.yml
 - **MACHINE CREDENTIAL:** WindowsVmCredential
 - **LIMIT:** windowsVM

NEW JOB TEMPLATE

DETAILS COMPLETED JOBS PERMISSIONS NOTIFICATIONS

* NAME **Install IIS** DESCRIPTION DESCRIPTION * JOB TYPE **Run**
☐ Prompt on launch

* INVENTORY **Agent VM Inventory** * PROJECT **WindowsProject** * PLAYBOOK **windows/enable-iis.yml**
☐ Prompt on launch

* MACHINE CREDENTIAL **WindowsVmCredential** CLOUD CREDENTIAL NETWORK CREDENTIAL
☐ Prompt on launch

FORKS **0** LIMIT **windowsVM** * VERBOSITY **0 (Normal)**
☐ Prompt on launch

JOB TAGS SKIP TAGS OPTIONS
☐ Prompt on launch

- ☐ Enable Privilege Escalation
- ☐ Allow Provisioning Callbacks
- ☐ Enable Concurrent Jobs

- Click on **Save** when done.

You can verify the template is saved when the newly created template appears on the list of templates at the bottom of the screen.

TEMPLATES 7

SEARCH

| NAME | TYPE |
|-------------------------|---------------------|
| Create User | Job Template |
| Demo Job Template | Job Template |
| Install Apache | Job Template |
| Install Firefox browser | Job Template |
| Install IIS | Job Template |
| Install Nginx | Job Template |
| Remove Nginx | Job Template |

Now, we will launch a Job Template by clicking the launch icon. We will be directed to the Job Results Page. The Jobs page shows details of all the tasks and events for that playbook run.

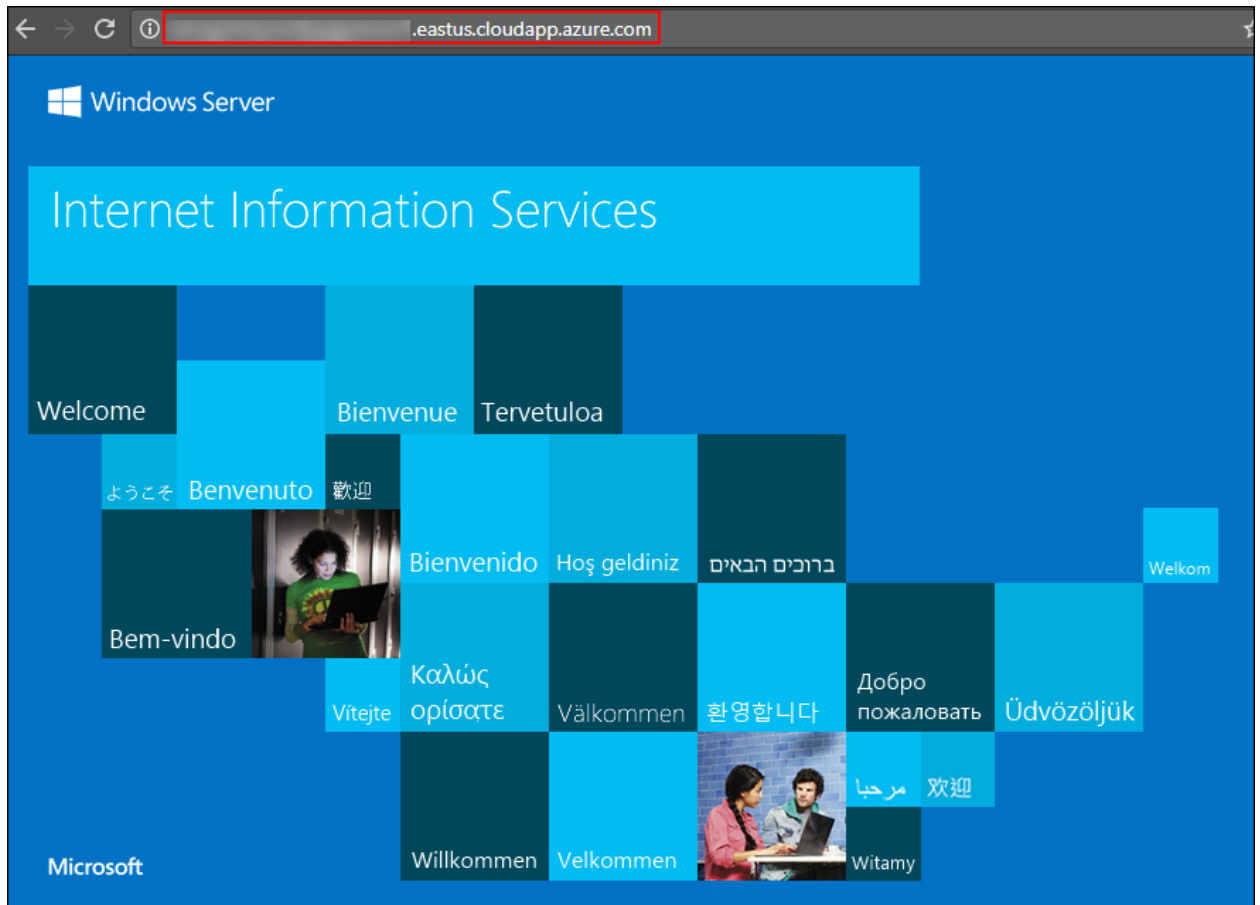
The screenshot displays the Red Hat Ansible Tower web interface. On the left, the 'DETAILS' sidebar shows the job status as 'Successful' (highlighted with a red box). The job is titled 'Install IIS' and was launched by 'admin' on '6/7/2017 00:22:58'. The inventory used is 'Agent VM Inventory', and the project is 'WindowsProject'. The playbook is 'windows/enable-iis.yml'. The job type is 'Run', and the machine credential is 'WindowsVmCredential'. The job completed successfully at '6/7/2017 00:23:14'.

The main panel shows the 'Install IIS' job execution details. It includes a search bar and a list of tasks. The tasks are as follows:

- 1 SSH password:
- 2
- 3 PLAY [installIIServer] *****
- 4
- 5 TASK [Gathering Facts] *****
- 6 ok: [redacted].eastus.cloudapp.azure.com
- 7
- 8 TASK [Install IIS Web-Server with sub features and management tools] *****
- 9 ok: [redacted].eastus.cloudapp.azure.com
- 10
- 11 PLAY RECAP *****
- 12 ok: [redacted].eastus.cloudapp.azure.com : ok=2 changed=0
- 13 failed=0

We can see the status as successful for Job to deploy IIS Web Server in Windows VM.

Now to verify that IIS is installed on Windows Virtual Machine and is accessible through Public DNS Name, we will open a browser and navigate to the Windows VM DNS name and check if the IIS Server default page is coming up as shown below.



END OF LAB

Thank You for following the test drive.
