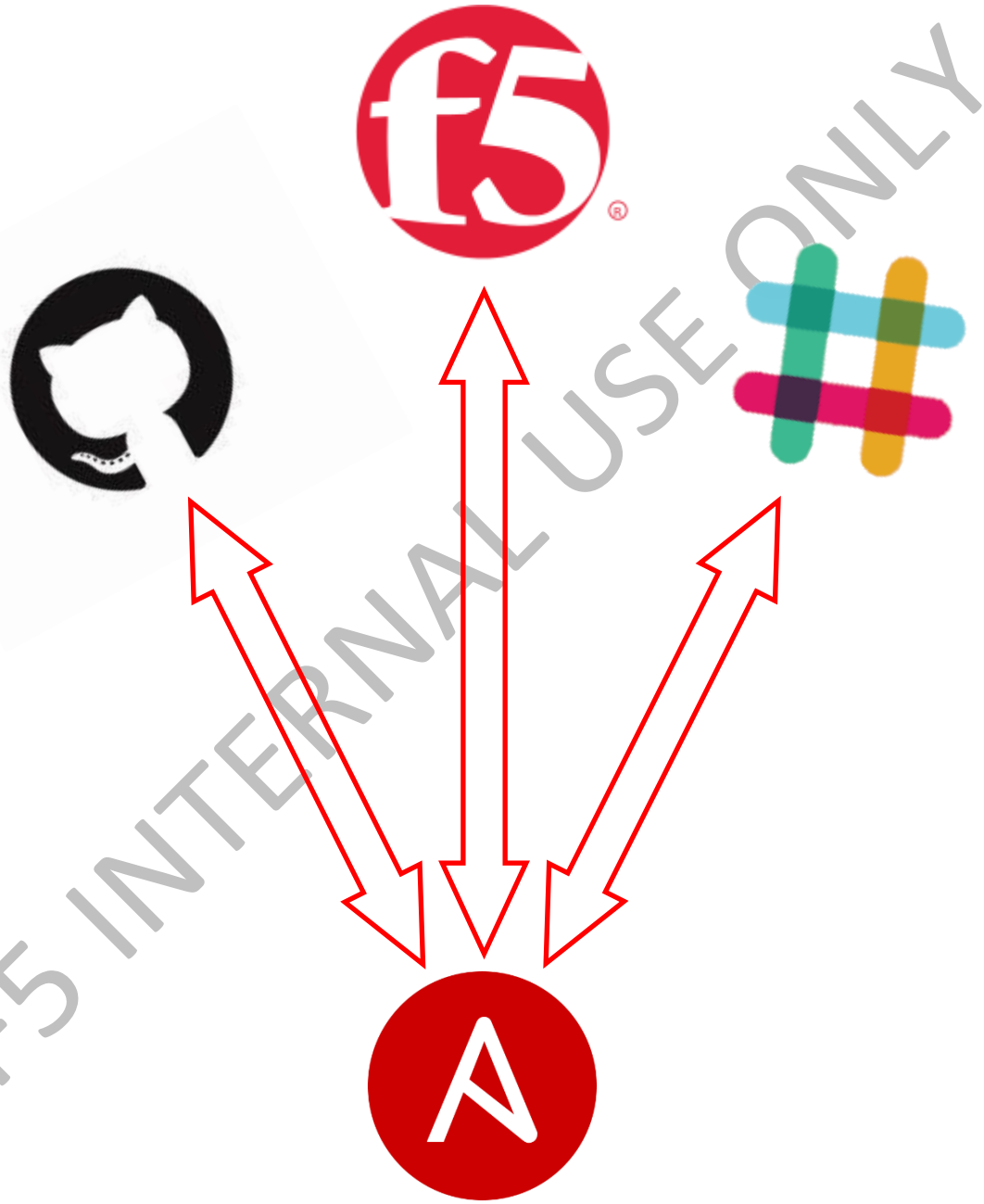
Ansible Tower with Github, Slack and BIGIP



Contents

A. Full Disclosure and References

All product names, logos, and brands are property of their respective owners. All company, product and service names used in this website are for identification purposes only. Use of these names, logos, and brands does not imply endorsement.

Materials (wordings and imagery) from the following sites may have been reproduced in this document :

https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/

https://help.github.com/articles/adding-a-file-to-a-repository-using-the-command-line/

https://git-scm.com/docs/git-config

https://git-scm.com/docs/git-pull

https://git-scm.com/docs/git-add

https://git-scm.com/docs/git-commit

https://help.github.com/articles/set-up-git/

https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/

http://releases.ansible.com/ansible-tower/setup/

http://docs.ansible.com/ansible-tower/3.2.3/html/userguide/

http://docs.ansible.com/ansible-tower/3.2.3/html/administration/

This document is intended for internal use only and shall not be distributed outside of F5 Networks.

1. Installation and Creation of Github Account and Repository

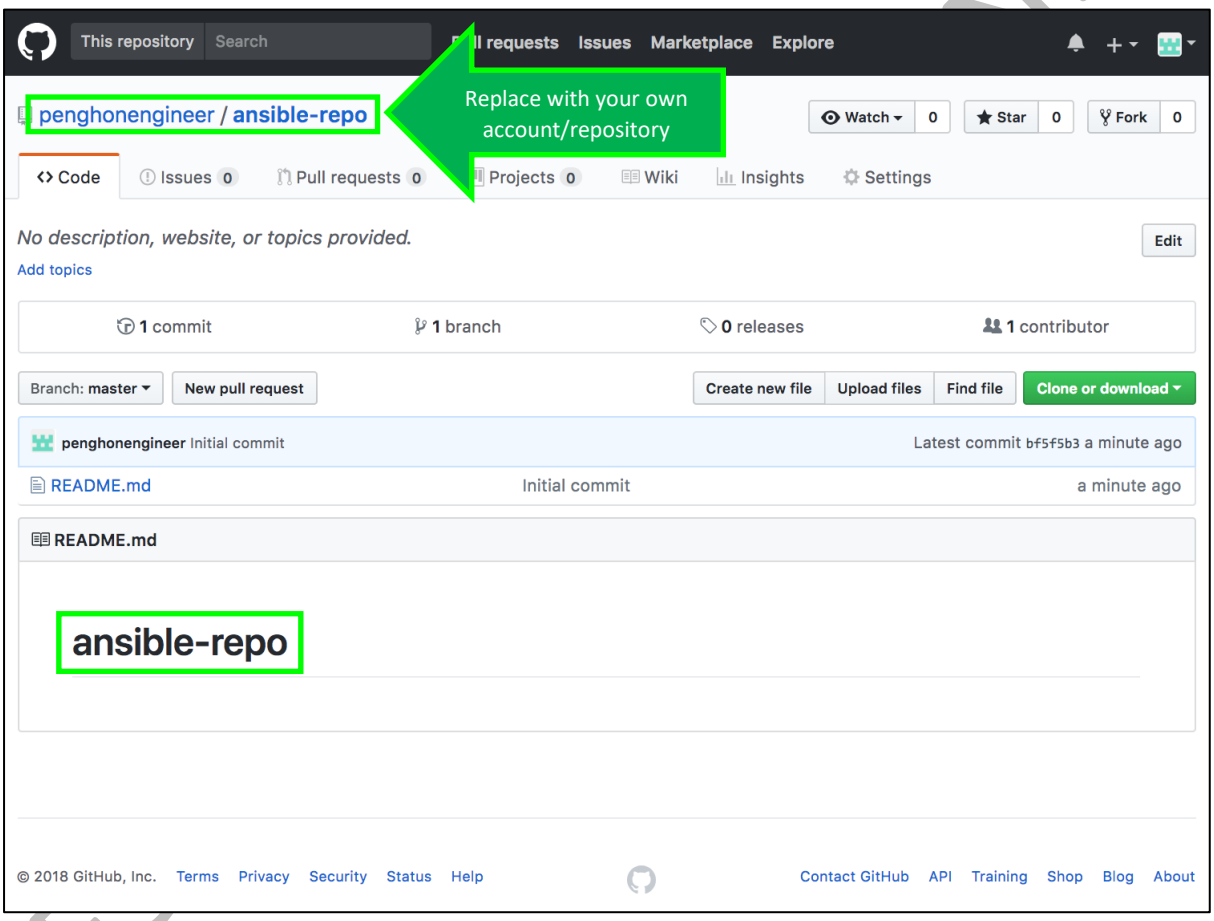
1.1 Create your very own Github Account!

Access https://github.com/ and sign up for your *Unlimited* (free) public repositories Github account. Complete the email verify process and start a project. Remember your repository name (you will need it later) and initialize the repository with a README. Your README can contain any string that would describe your Github repository.

Note

If you already have a Github account and you would like to use it for this Lab, skip to <>

Your new Github repository should look like this:



1.2 Create your SSH keypair

Using the provided root credentials SSH to your assigned Ansible Tower Controller/Git Local Repository and generate a keypair. This keypair will be used to automatically authenticate to Github when you push the latest changes from your local repository to your Github master branch:

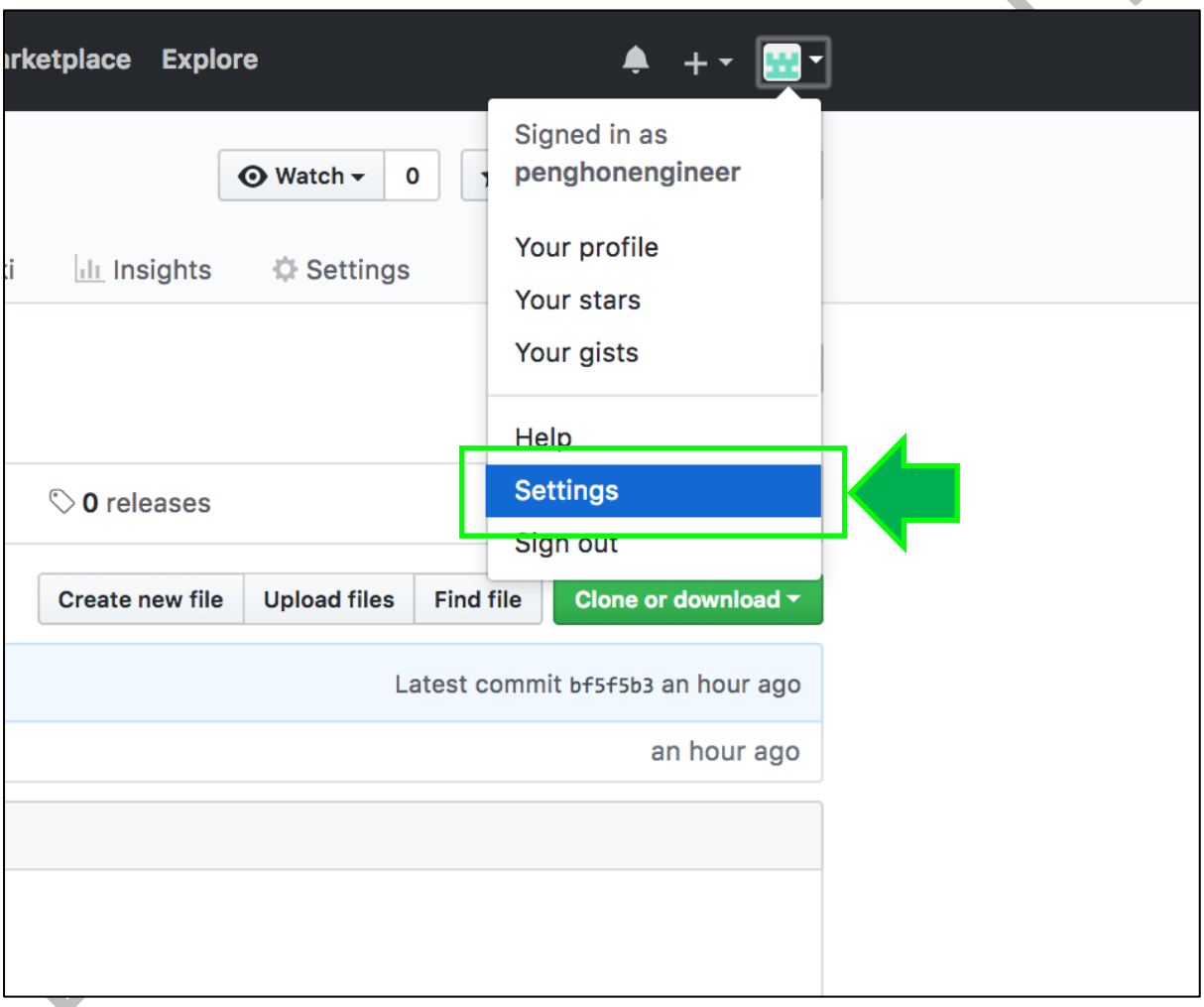
```
[root@localhost ~ ]# ssh-keygen -t rsa -b 4096
```

Note

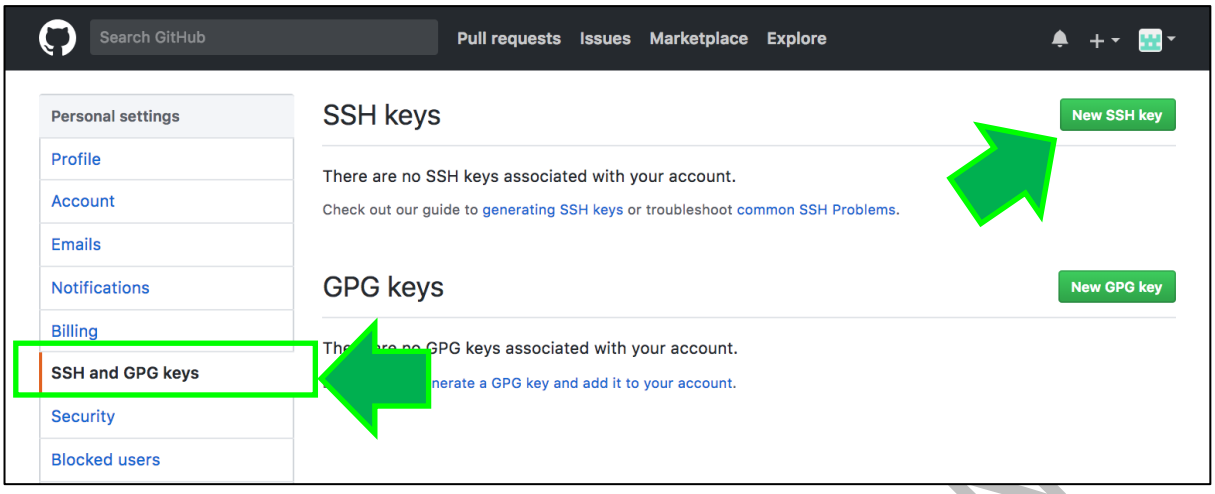
For the purpose of this Lab do not use any SSH passphrase and accept all default parameters when generating your SSH private key.

Copy out the contents of the **SSH public key** to a text editor and log back into your Github account. We will be adding the SSH public key from your assigned Ansible Tower Controller/Git Local Repository to your Github account.

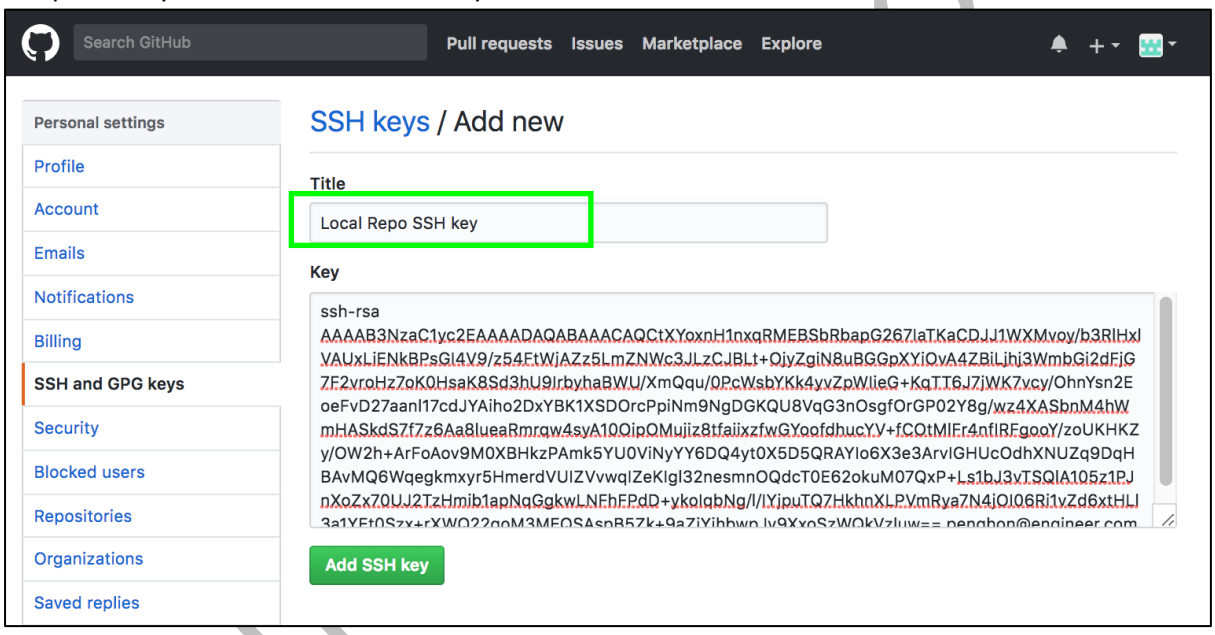
From your Github account select **Settings** from the menu as shown below:



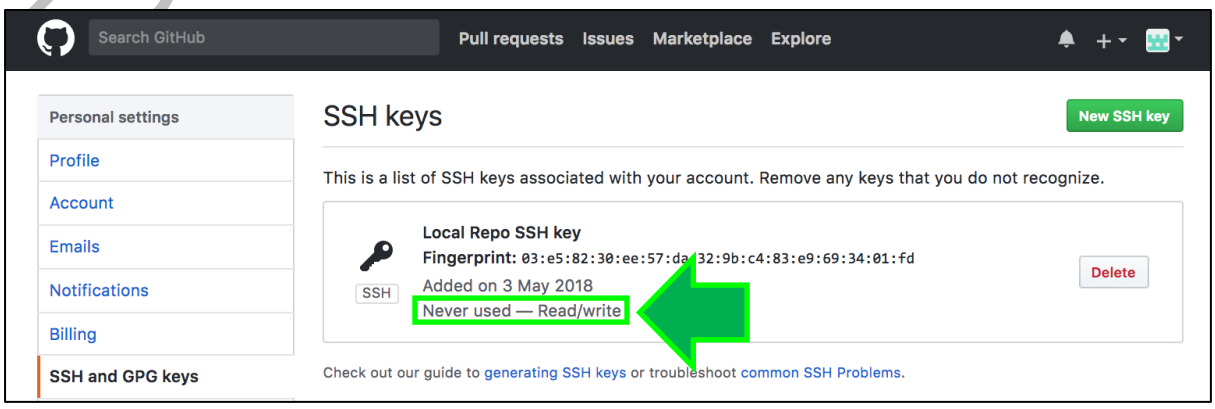
Select “SSH and GPG keys” from the left menu and click on New SSH key:



Add in the previously copied SSH public key starting with “ssh-rsa” and provide a meaningful title for this public key and click on “Add SSH key”:



Once Github accepts the public key, you should be able to see a summary page of the available SSH keys associated to your Github account. Notice that your new key has not been used before. This is expected and will change once you are able to sync and commit file changes from your local repo to Github:



[Optional] Check that the fingerprint of the public key that you have uploaded is the same as what you have on the Ansible Controller:

```
[root@localhost ~]# ssh-keygen -lf ~/.ssh/id_rsa.pub
4096 03:e5:82:30:ee:57:da:32:9b:c4:83:e9:69:34:01:fd  penghon@engineer.com (RSA)
```

1.3 [Optional] Download and install Git

We do not need the latest and greatest Git. Download the available Git (git **version** 1.8.3.1) from the yum repositories.

Using root credentials, SSH to your Ansible Tower Controller and install the latest available Git package from the yum repositories:

```
[root@localhost ~]# yum install git -y
```

Check that git has been installed successfully:

```
[root@localhost ~]# git --version
```

We will now create a project directory to use as local repository. You will use this local repository to push your playbooks and inventory files to Github.

Create your local repository. You can replace the path */git/ansible-repo/* with any directory that you would like to use instead. Throughout this document we will use */git/ansible-repo/* has the local repository. If you would like to use a different folder, just remember to replace any instance of */git/ansible-repo* with your desired directory:

```
[root@localhost ~]# mkdir -p /git/ansible-repo
```

Access your local repository:

```
[root@localhost ~]# cd /git/ansible-repo
```

Initialize your local repository:

```
[root@localhost ansible-repo]# git init .
```

Add the URL for the remote repository where you will be pushing from your local repository. We will be using SSH to securely push commits to Github. Remember to change the strings that are highlighted in red to your own Github account and repository URL:

```
[root@localhost ansible-repo]#
git remote add origin git@github.com:penghonengineer/ansible-repo.git
```

Setting push.default to simple, which is same as upstream except that if the branch and the local branch are different, git will not push the commits.

```
[root@localhost ansible-repo]#
git config --global push.default simple
```

Setting the default user name to be recorded in any newly created commits.

```
[root@localhost ansible-repo]#
git config --global user.name penghonengineer
```

Setting the default user email to be recorded in any newly created commits.

```
[root@localhost ansible-repo]#
git config --global user.email penghon@engineer.com
```

Merging Github master branch with local repository:

```
[root@localhost ansible-repo]#
git pull git@github.com:penghonengineer/ansible-repo master
```

Create *testfile* in local repository:

```
[root@localhost ansible-repo]# touch testfile
```

Adding new content to git index and prepare content to be staged for the next commit:

```
[root@localhost ansible-repo]# git add testfile
```

Adding a commit message and storing the the contents of the index in a new commit:

```
[root@localhost ansible-repo]#
git commit -m "Uploading test file" testfile
```

Set the branch to master once and for subsequent commits you can omit the branch that you are merging your local repository to:

```
[root@localhost ansible-repo]#
git push --set-upstream origin master
```

Once you can see the *testfile* appearing in your Github repository, you are all set!

If you have made a mistake with any of your git config command, you can edit it via:

```
git config --global --unset { parameter }
```

Or:

```
Git config --global --edit
```

Or:

```
git config --global --replace-all { parameter } "{ correct value }"
```

1.4 Adding Local Repository to Github

We will be using sample (but working) BIGIP playbooks in this lab in order to save time.

From your Ansible Tower Controller/Local Git Repository, copy the sample BIGIP playbook ***/var/tmp/sample_files/change_bigip_hostname.yaml*** to your git local repo directory ***/git/ansible-repo/*** and perform similar steps to earlier section on Github:

```
[root@localhost ~]#
cp /var/tmp/sample_files/change_bigip_hostname.yaml /git/ansible-repo/
[root@localhost ~]# cd /git/ansible-repo/
[root@localhost ansible-repo]# ls
change_bigip_hostname.yaml  README.md
[root@localhost ansible-repo]# git add change_bigip_hostname.yaml
[root@localhost ansible-repo]#
git commit -m "Sample BIGIP playbook" change_bigip_hostname.yaml
[master 95a68c5] Sample BIGIP playbook
 1 file changed, 17 insertions(+)
 create mode 100644 change_bigip_hostname.yaml
[root@localhost ansible-repo]# git push
Counting objects: 4, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 529 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:penghonengineer/ansible-repo.git
   1bf32af..95a68c5  master -> master
[root@localhost ansible-repo]#
```

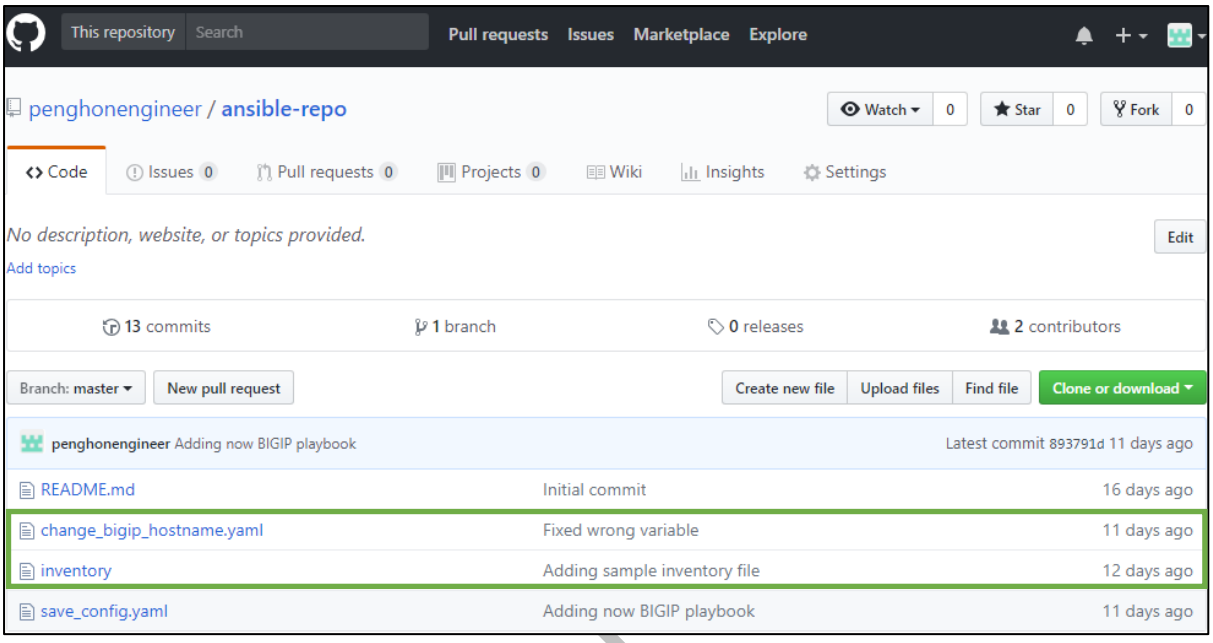
1.5 Inventory File from Local Repository to Github

Copy the ***/var/tmp/sample_files/inventory*** to your local git repository and upload to your Github account:

```
[root@localhost ansible-repo]#
cp /var/tmp/sample_files/inventory /git/ansible-repo/
[root@localhost ansible-repo]# git add inventory
[root@localhost ansible-repo]#
git commit -m "Adding sample inventory file"
[master 3b1d64d] Adding sample inventory file
 1 file changed, 2 insertions(+)
 create mode 100644 inventory
[root@localhost ansible-repo]# git push
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
```

```
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To git@github.com:penghonengineer/ansible-repo.git
   95a68c5..3b1d64d  master -> master
```

Once the git push has been completed, validate that the files have been uploaded to Github:



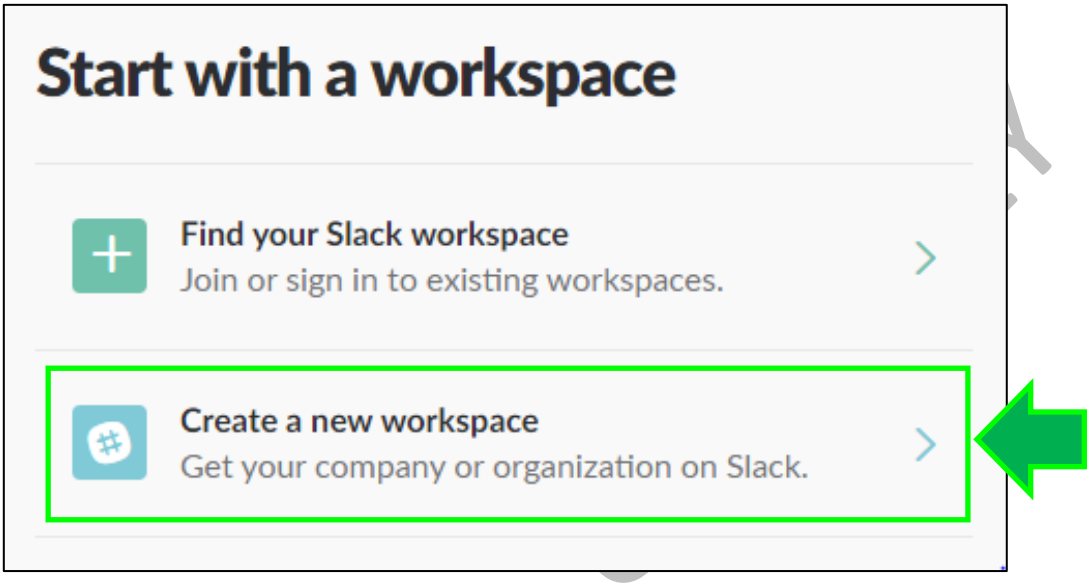
Repeat the above steps by adding ***/var/tmp/sample_files/save_config.yaml*** to your Github repository.

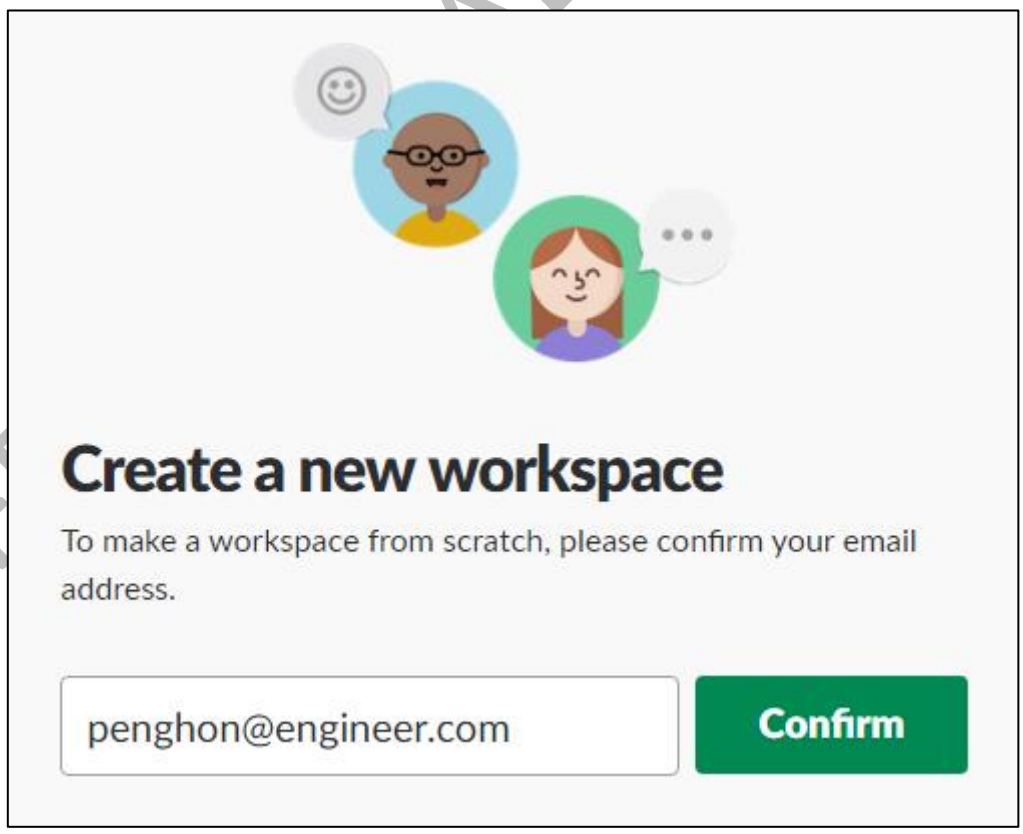
2. Slack Account Creation

2.1 Create your own Slack Workspace and Channel!

Using a browser access the following URL: https://slack.com

Follow the instructions after clicking on “Create a new workspace”. If you already have your own slack workspace you can skip the following few pages until reach the Legacy API token section.

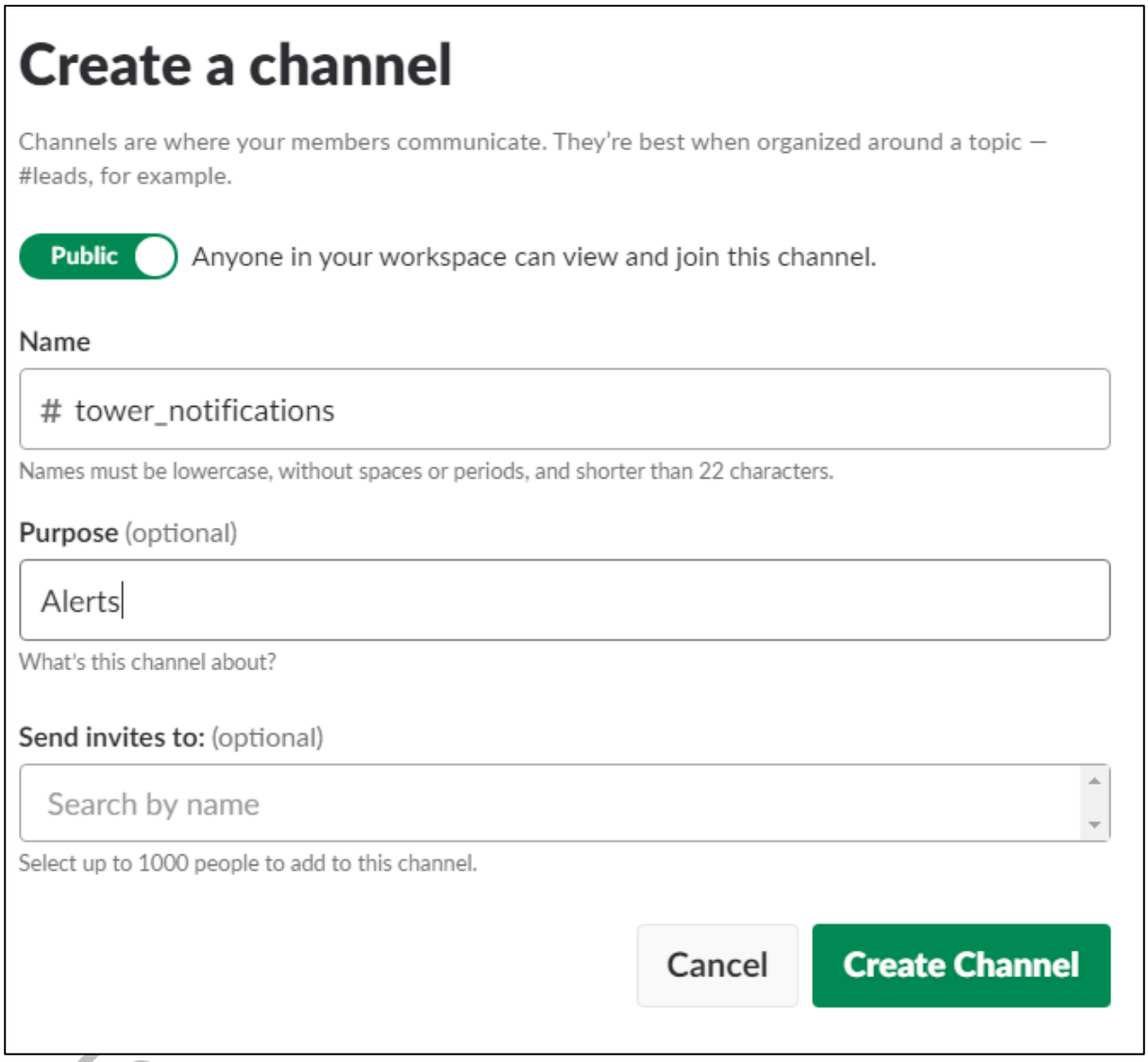




Key in the Stack confirmation code. Complete the profile field and continue to password configuration.

Key in your password, create the slack workspace and skip the “Send Invitations”.

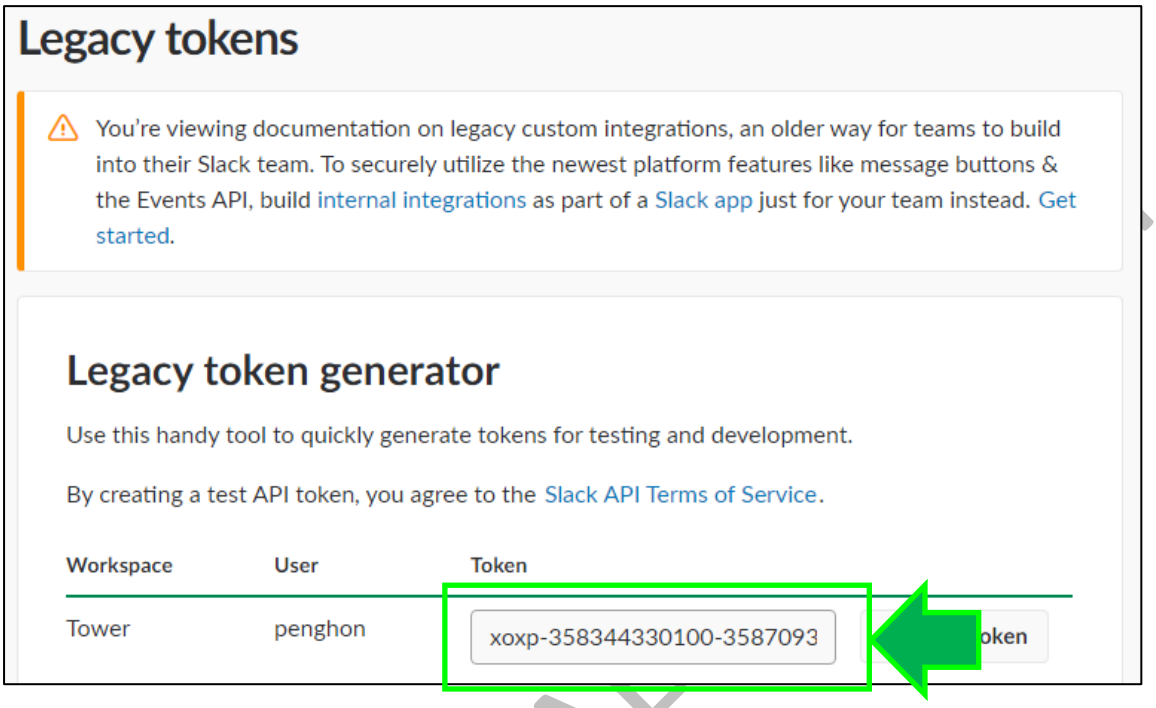
Create a channel within your newly created Slack workspace. We will be using this channel to send notifications from your Ansible Tower.

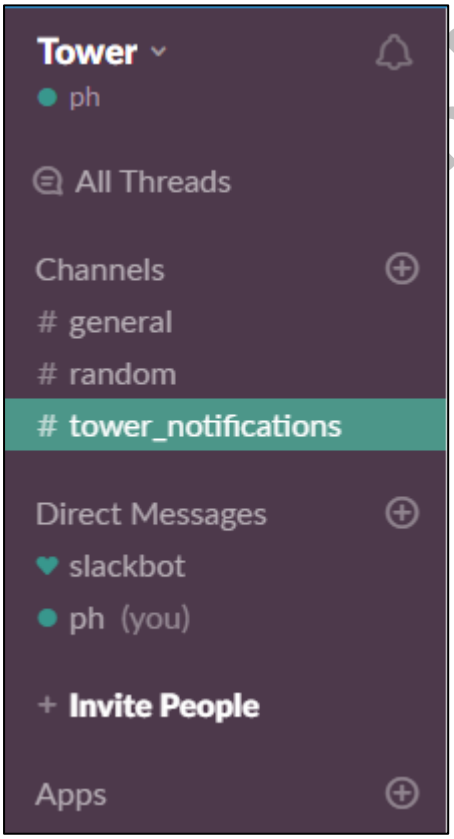


2.2 Slack Legacy API Token Creation

Using a new browser tab, access https://api.slack.com/custom-integrations/legacy-tokens and select "Create token".

Copy and save the value of this token to a text editor. You will need the value of this token when you configure Ansible Tower Slack notification.





Your Slack workspace have the #general and #random default channels configured together with the channel that you have created to receive notifications from your Ansible Tower.

[Optional] Download the slack app on your mobile phone and log into your newly created slack workspace.

3. Ansible Tower Installation and Configuration

We will be installing Ansible Tower 3.2.3 and will upgrade to 3.2.4 (latest version as of May 2018) as an optional task.

Just a quick word on the perquisites and requirements for a Tower installation.

Ansible Tower minimum requirements as of 3.2.3 are as follows:

- Supported Operating Systems:
  - Red Hat Enterprise Linux 7.2 or later 64-bit
  - CentOS 7.2 or later 64-bit
  - Ubuntu 14.04 LTS 64-bit
  - Ubuntu 16.04 LTS 64-bit
- Currently supported version of Mozilla Firefox or Google Chrome
  - Other HTML5 compliant web browsers may work but are not fully tested or supported.
- 2 CPUs minimum
  - 2 CPUs recommended per 20 forks
- 2 GB RAM minimum (4+ GB RAM recommended)
  - 2 GB RAM (minimum)
  - 4GB RAM per 100 forks
- 20 GB of dedicated hard disk space
  - 10GB of the 20 GB requirement must be dedicated to /var/ where Tower stores its files and working directories
  - Storage volume rated for minimum baseline of 750 IOPS.
- 20 GB of dedicated hard disk space for nodes containing a database (150 GB+ recommended)
  - The storage volume should be rated for a high baseline IOPS (1000 or more.)
- 64-bit support required
- PostgreSQL version 9.6 required for Ansible Tower 3.2 and later
- Ansible Engine version 2.2 required for Ansible Tower 3.2 and later
- For AWS EC2:
  - M3.medium or larger
  - For 100 managed hosts, m3.xlarge or larger is needed

For the purpose of this lab your Ansible Tower will be running on the following specifications:

- CentOS Linux release 7.3.1611 (Core)
- 1 CPU
- 2 GB RAM
- 30 GB of virtualized disk of questionable IOPS
- Single instance of Ansible Tower with integrated installation (PostgreSQL will be installed on the same machine as the Ansible Tower)

For ease of installation and avoidance of permission related issues every task in this lab will be using the root credentials.

3.1 Ansible Tower Installation

Download the Ansible Tower installation/upgrade tool from http://releases.ansible.com/ansible-tower/setup/

In order to save time this file has been already downloaded to the */var/tmp* folder of your designated Ansible Tower controller.

Extract the files and access the ansible-tower-setup-3.2.3 directory:

```
[root@localhost tmp]# tar xvzf ansible-tower-setup-3.2.3.tar.gz
[root@localhost tmp]# cd ansible-tower-setup-3.2.3
```

Modify the */var/tmp/ansible-tower-setup-3.2.3/inventory* file with the following modifications highlighted in red:

```
[tower]
localhost ansible_connection=local

[database]

[all:vars]
admin_password='abc123!@'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='abc123!@'

rabbitmq_port=5672
rabbitmq_vhost=tower
rabbitmq_username=tower
rabbitmq_password='abc123!@'
rabbitmq_cookie=cookiemonster

# Needs to be true for fqdns and ip addresses
rabbitmq_use_long_name=false

# Isolated Tower nodes automatically generate an RSA key for authentication;
# To disable this behavior, set this value to false
# isolated_key_generation=true
```

Once the necessary changes above has been made, run ./setup.sh:

```
[root@localhost ansible-tower-setup-3.2.3]# ./setup.sh
```

The setup.sh script will install all necessary dependencies such as epel-release and Ansible Engine.

To track the installation process, run the following

```
[root@localhost ~]# tail -f /var/log/messages
```

The following task will require quite a fair bit of time:

```
TASK [awx_install : Migrate the Tower database schema (may take awhile when upgradi
ng).]
```

Once the installation process has completed check that all necessary processes are up and running:

```
[root@localhost ansible-tower-setup-3.2.3]#
ansible-tower-service status | grep -i running
Redirecting to /bin/systemctl status  postgresql-9.6.service
   Active: active (running) since Tue 2018-05-08 22:34:22 SGT; 1 day 11h ago
Redirecting to /bin/systemctl status  rabbitmq-server.service
   Active: active (running) since Tue 2018-05-08 22:34:27 SGT; 1 day 11h ago
Redirecting to /bin/systemctl status  nginx.service
   Active: active (running) since Tue 2018-05-08 22:34:28 SGT; 1 day 11h ago
Redirecting to /bin/systemctl status  supervisord.service
   Active: active (running) since Tue 2018-05-08 22:34:28 SGT; 1 day 11h ago
```

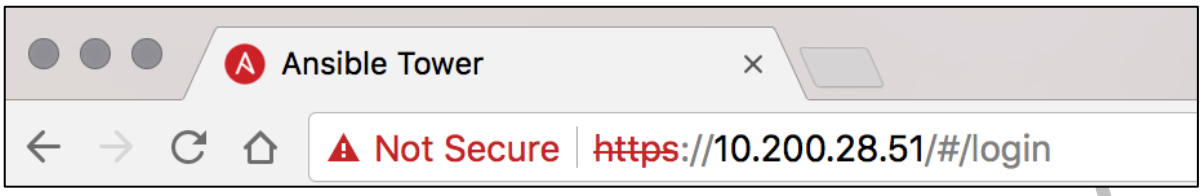
Four essential services must be in running:

- Postgresql
- Rabbitmq-server
- Nginx
- Supervisord

3.2 Ansible Tower Licensing

After your ansible Tower has been successfully installed, access the Tower GUI by using your browser and key in your Ansible Controller management IP.

For example:



Log into the Tower UI with the admin credentials that was specified in the ***/var/tmp/ansible-tower-setup-3.2.3/inventory*** file (check the file if you do not remember the admin password).

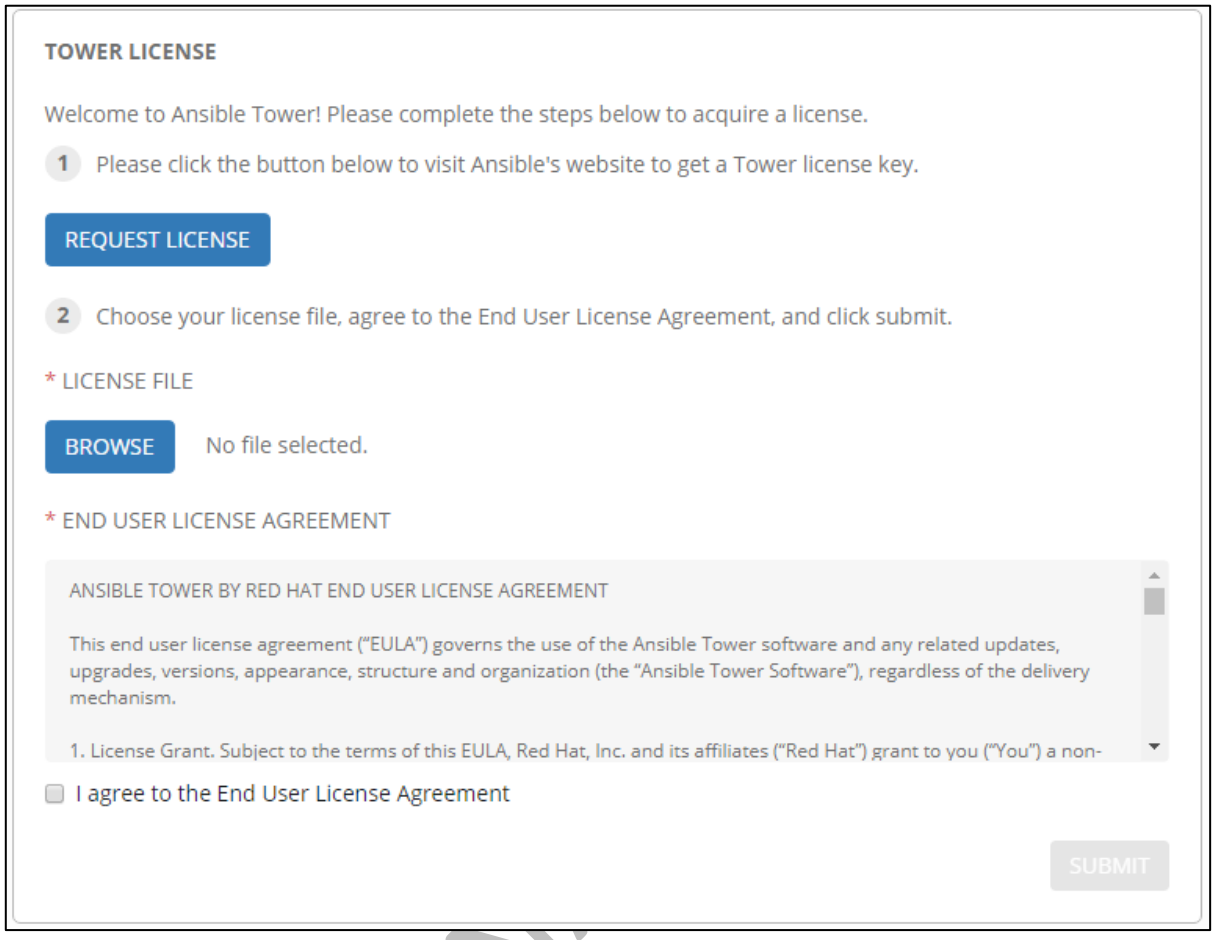
A copy of an NFR (Not for Resale) Ansible Tower Enterprise License has been made available on your Ansible Tower controller as */var/tmp/sample_files/ansible_tower_license.txt*.

Copy the contents of the above license file and create a local file. You will need to upload this file later via the Tower UI.

Ansible Tower login page looks like the following:

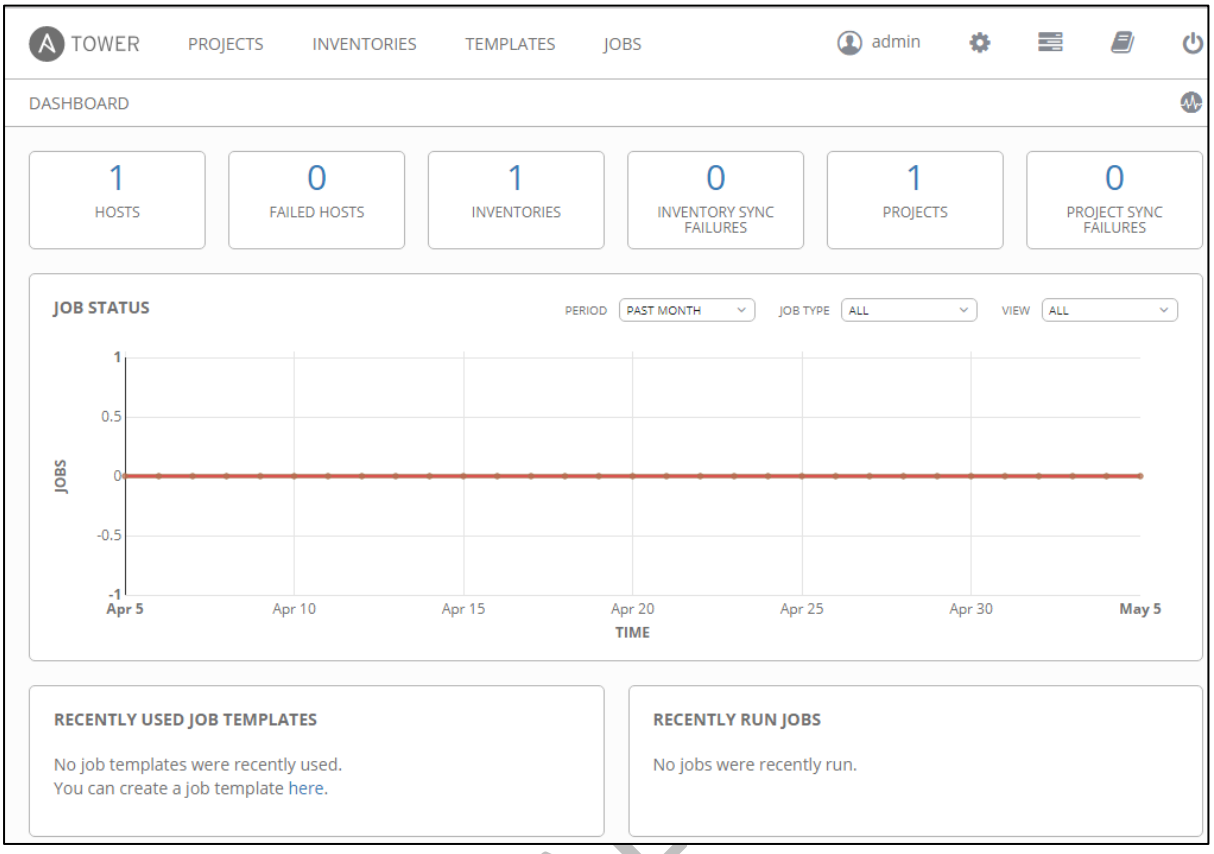
Key in the username “admin” and the password.

Select the browse button, upload the Ansible Tower License that you have created previously and agree to the EULA.



Once the system accepts the license file you will be brought to the Ansible Tower dashboard.

3.3 Ansible Tower Dashboard



Several objects are already been pre-defined (following list is not exhaustive):

- Demo Project
- Demo Inventory
- Default organization
- Demo Credentials
- Demo Job Template

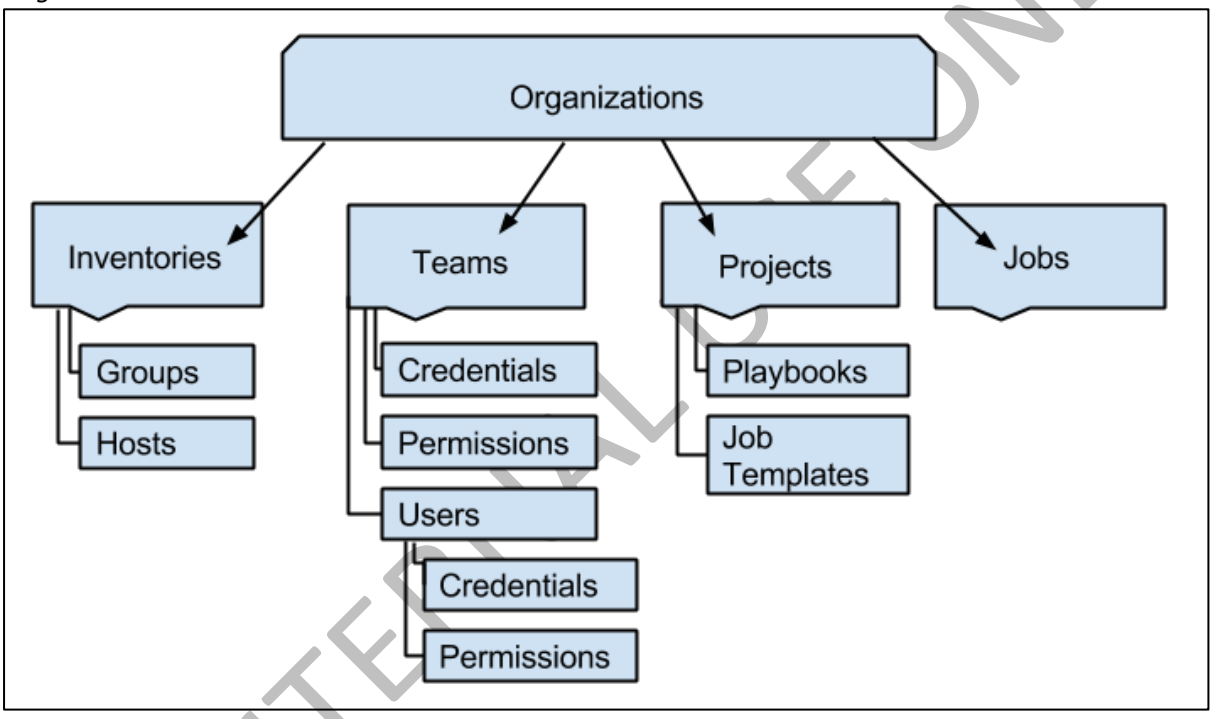
3.4 Exploring the Ansible Tower Dashboard and Interface

Take a minute to examine the Ansible Tower Dashboard and interface.

3.5 Explanation of Dashboard and interface objects

Quick Access Menu	Purpose and Funtionality
Projects	A Project is a logical collection of Ansible playbooks, represented in Tower.
Inventories	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.
Templates	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. A job is an execution of a job template.
Jobs	This shows the list of jobs that have run in the past.

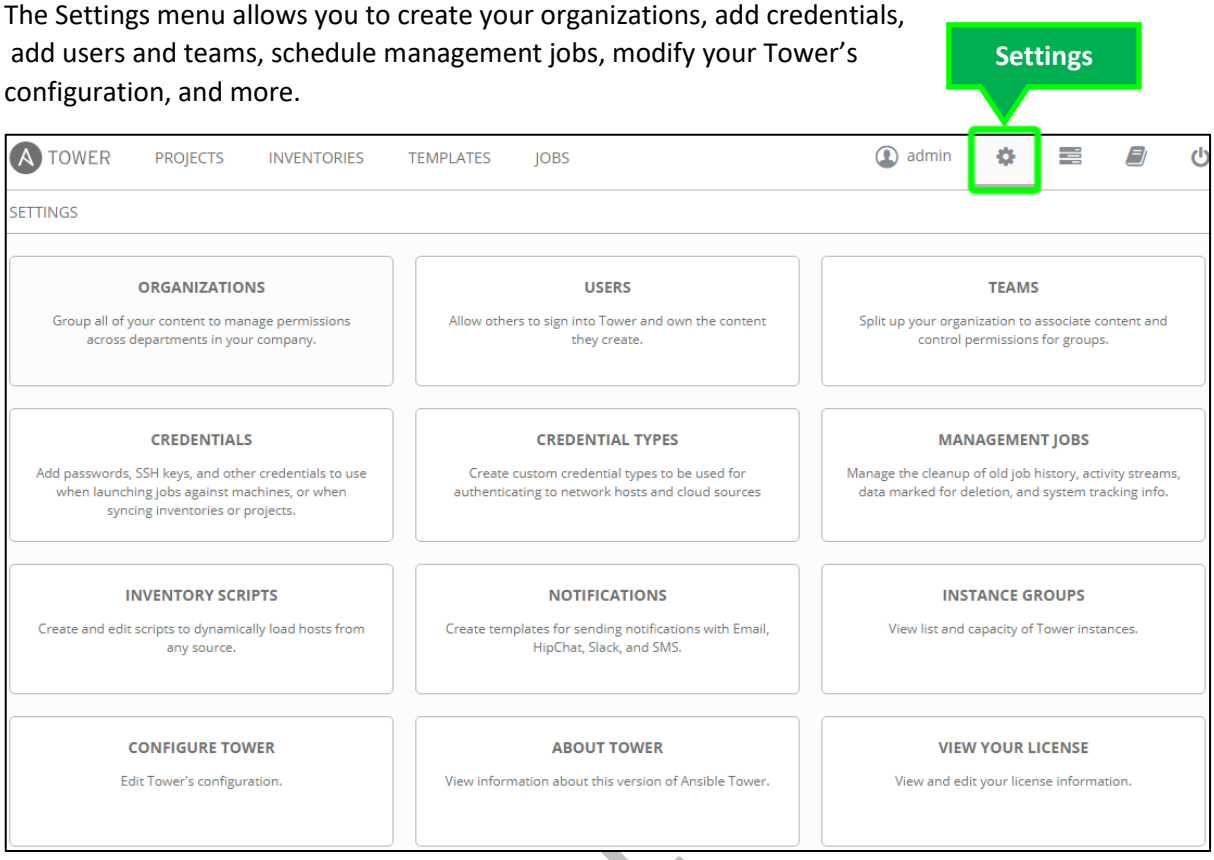
Ansible Tower with Enterprise license enables multitenancy through the use of different *Organizations*.



Organization is the highest level in the Tower object hierarchy as shown above.

3.6 Ansible Tower Settings Menu

The Settings menu allows you to create your organizations, add credentials, add users and teams, schedule management jobs, modify your Tower's configuration, and more.



Settings	Purpose and Functionality
Organizations	An Organization is a logical collection of Users, Teams, Projects, and Inventories, and is the highest level in the Tower object hierarchy. This Organizations displays all of the existing organizations for your installation of Tower and will allow you to manage new organizations to your Ansible Tower.
Users	A User is someone who has access to Tower with associated permissions and credentials. This Users link allows you to manage all Tower users.
Teams	A Team is a subdivision of an organization with associated users, projects, credentials, and permissions. Teams provide a means to implement role-based access control schemes and delegate responsibilities across organizations. . This Teams link allows you to manage Teams for Tower.
Credentials	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. You can add credential objects from the following predefined credential types: • Amazon Web Services • Ansible Tower • Google Compute Engine • Insights • Machine • Microsoft Azure Resource Manager • Network • OpenStack

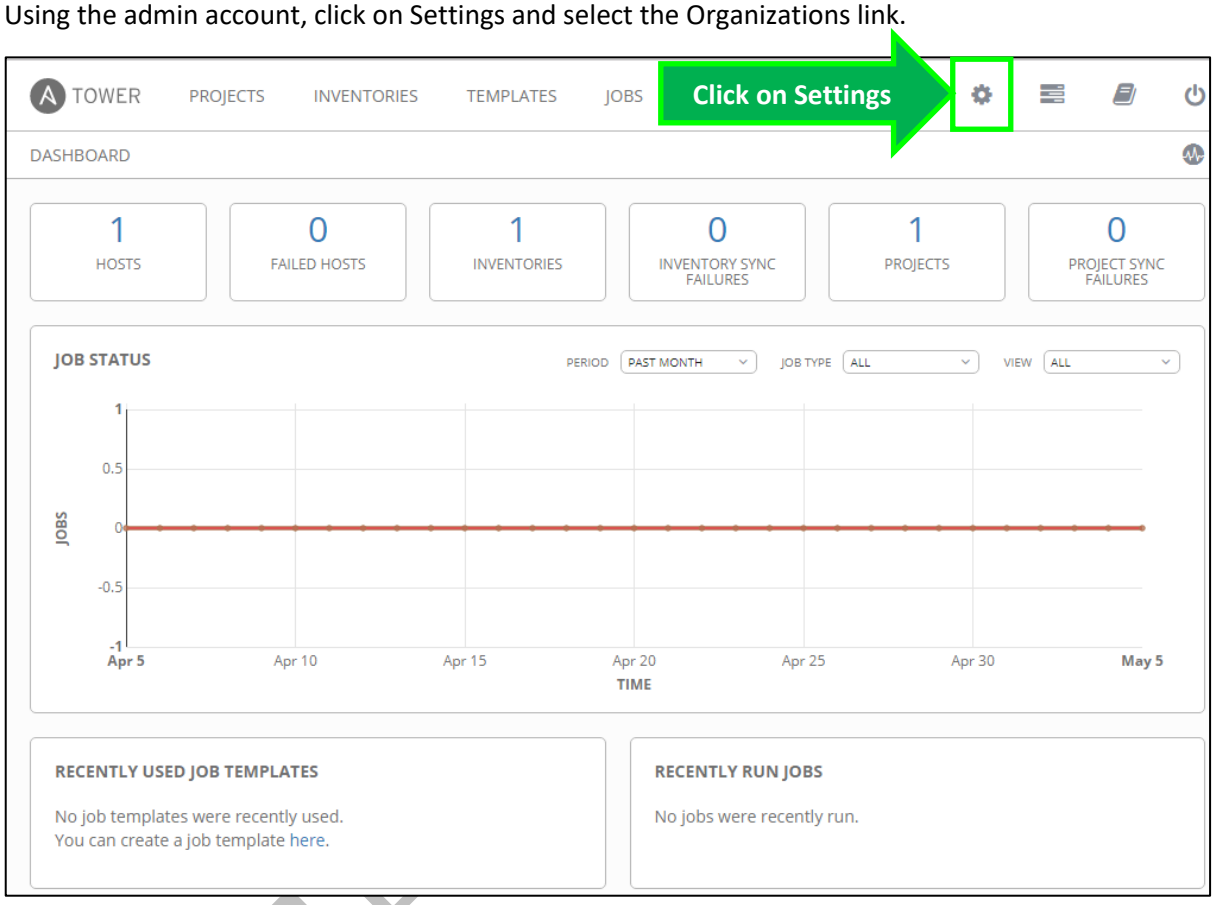
	• Red Hat CloudForms • Red Hat Satellite 6 • Red Hat Virtualization • Source Control • Vault • VMware vCenter
Credential Types	Creation and management of custom credential type in standard format using a YAML/JSON-like definition.
Management Jobs	Management Jobs assist in the cleaning of old data from Tower, including system tracking information, job histories, and activity streams. You can use this if you have specific retention policies or need to decrease the storage used by your Tower database.
Inventory Scripts	Tower includes built-in support for syncing dynamic inventory from cloud sources such as Amazon AWS, Google Compute Engine, among others. Tower also offers the ability to use a custom script to pull from your own inventory source.
Notifications	Creation and management of notification. You can configure Ansible Tower to send notification of job success or failure to the following predefined notification types: • Email • Slack • Twilio • PagerDuty • HipChat • Webhook • IRC
Instance Groups	Manage Tower cluster instance groups.
Configure Tower	In Ansible Tower version 3.1.0, you can configure various Tower settings within the Tower user interface, in the following tabs: Authentication: Enable simplified login for your Tower applications. Jobs: Update settings pertaining to Jobs within Tower. System: Define system-level features and functions. User Interface: Set the level of data collection for use in usability analytics; and configure custom logos and login messages.
About Tower	A funky picture that shows the version of the Ansible Tower.
View Your License	Management of your Ansible Tower license.

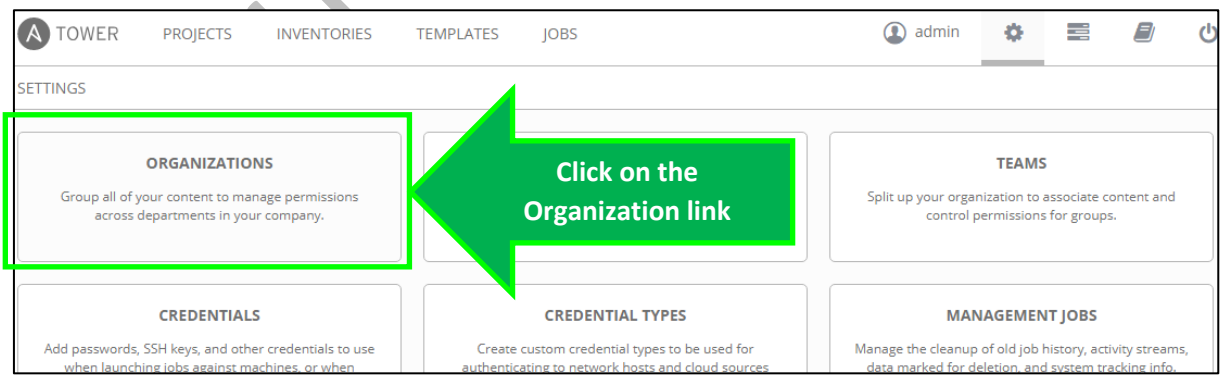
3.7 Ansible Tower Organization Creation

An Ansible Tower Organization is a logical collection of Users, Teams, Projects, and Inventories, and is the highest level in the Tower object hierarchy. Tower creates a default organization automatically and it is the only organization for non-Enterprise Tower installations.

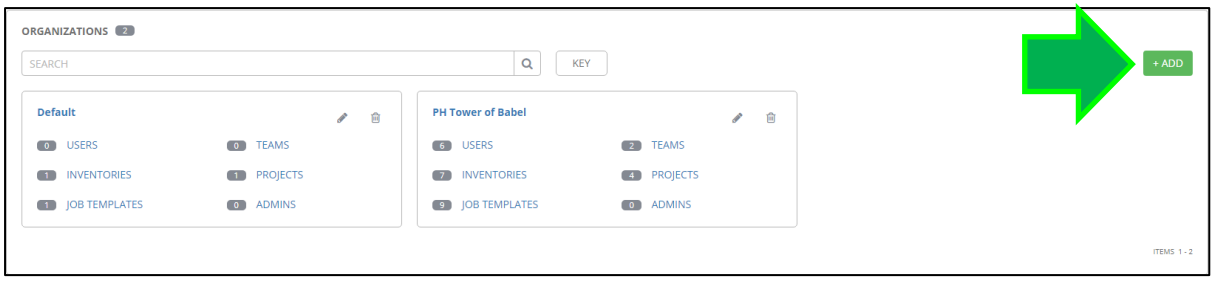
Since we are using Enterprise license for this lab, the first object that we will be creating will be your own Tower organization.

Using the admin account, click on Settings and select the Organizations link.

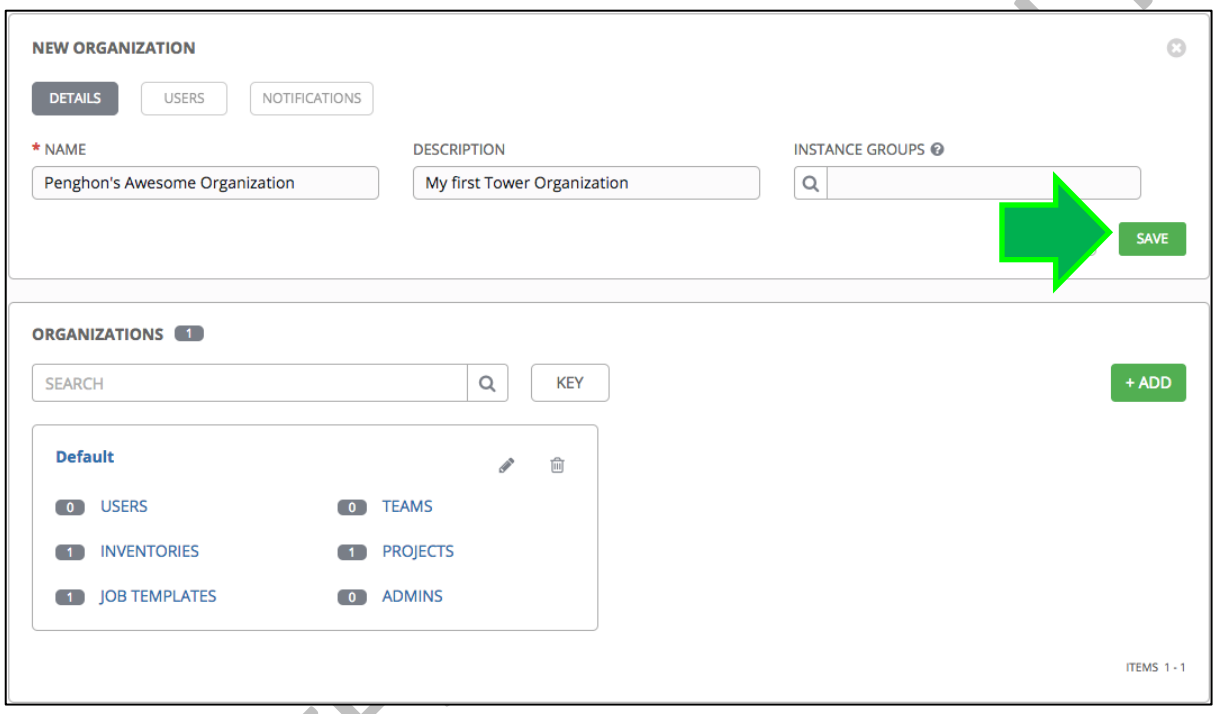




Select +ADD to create a new organization:

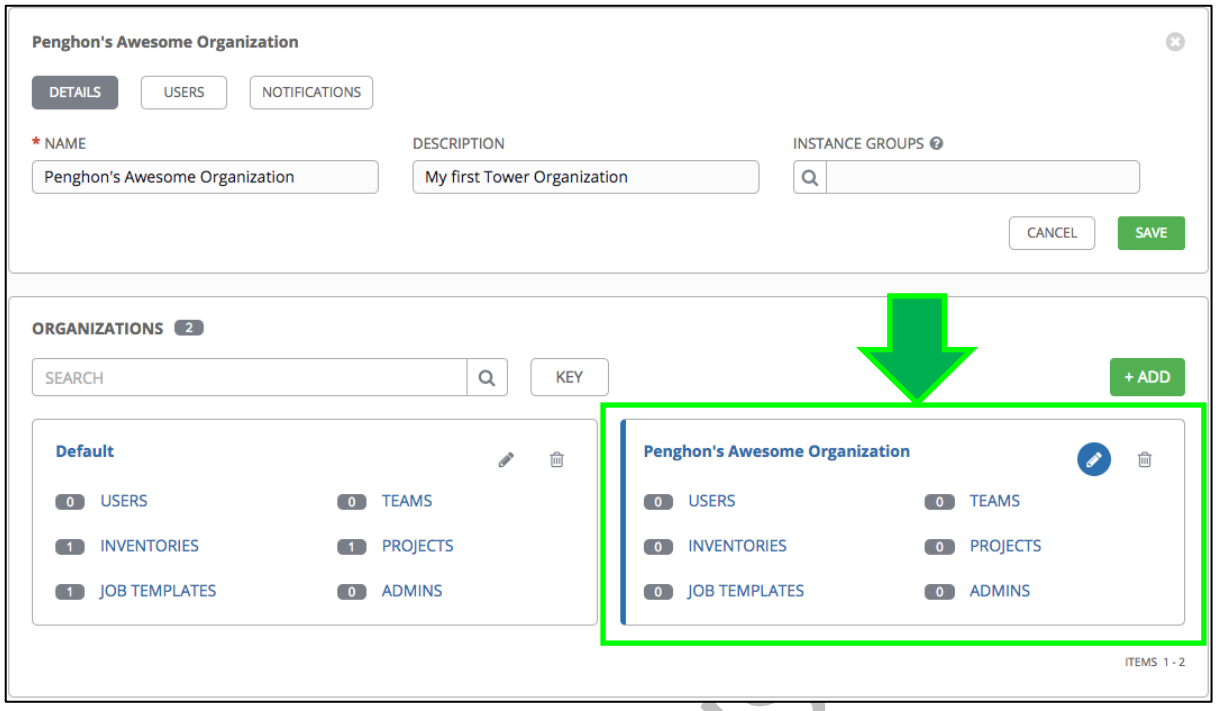


Fill the *Organization* name, and optionally its description, ignoring instance groups and click on SAVE.



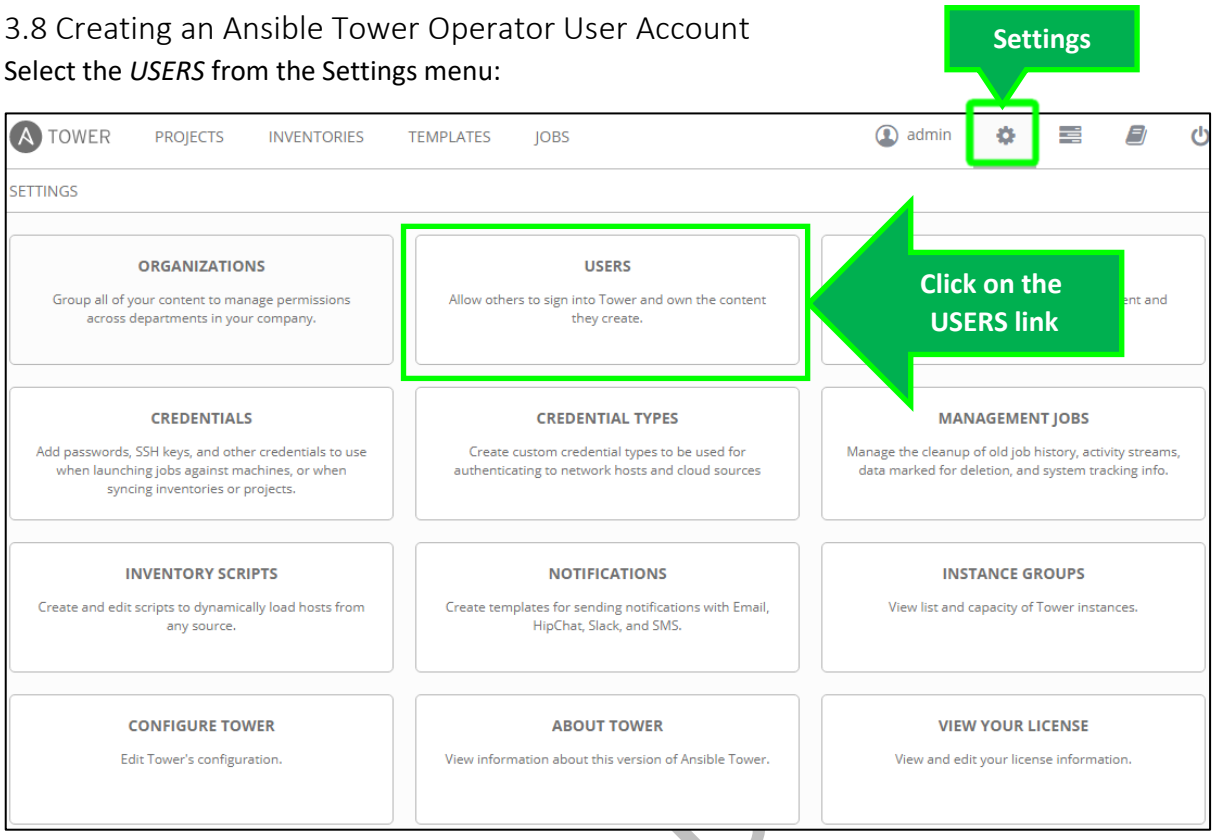
Tower Instance groups are instances of Ansible Tower that are grouped according to Organizations, Inventories or Job Templates for job execution load balancing. We will not be covering instance groups in this lab.

After saving the configuration, the newly created organization should look similar to the following:

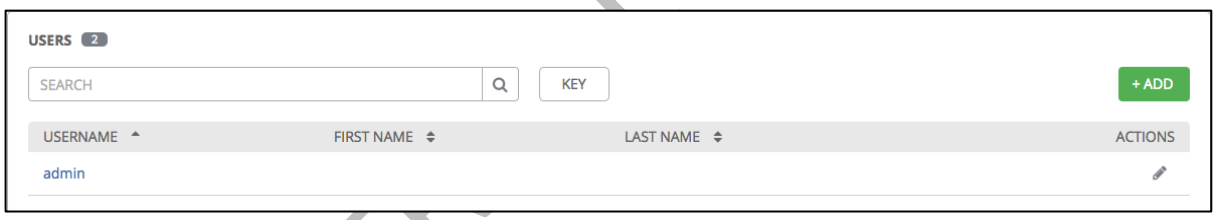


3.8 Creating an Ansible Tower Operator User Account

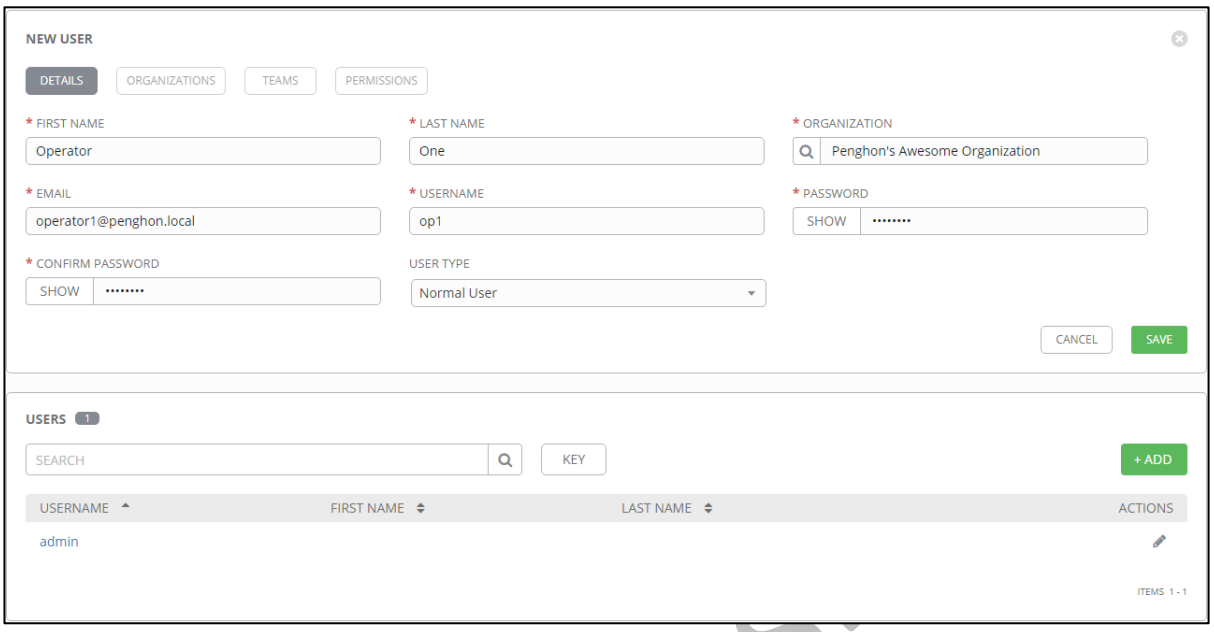
Select the *USERS* from the Settings menu:



Click **+ADD** to add a new user:



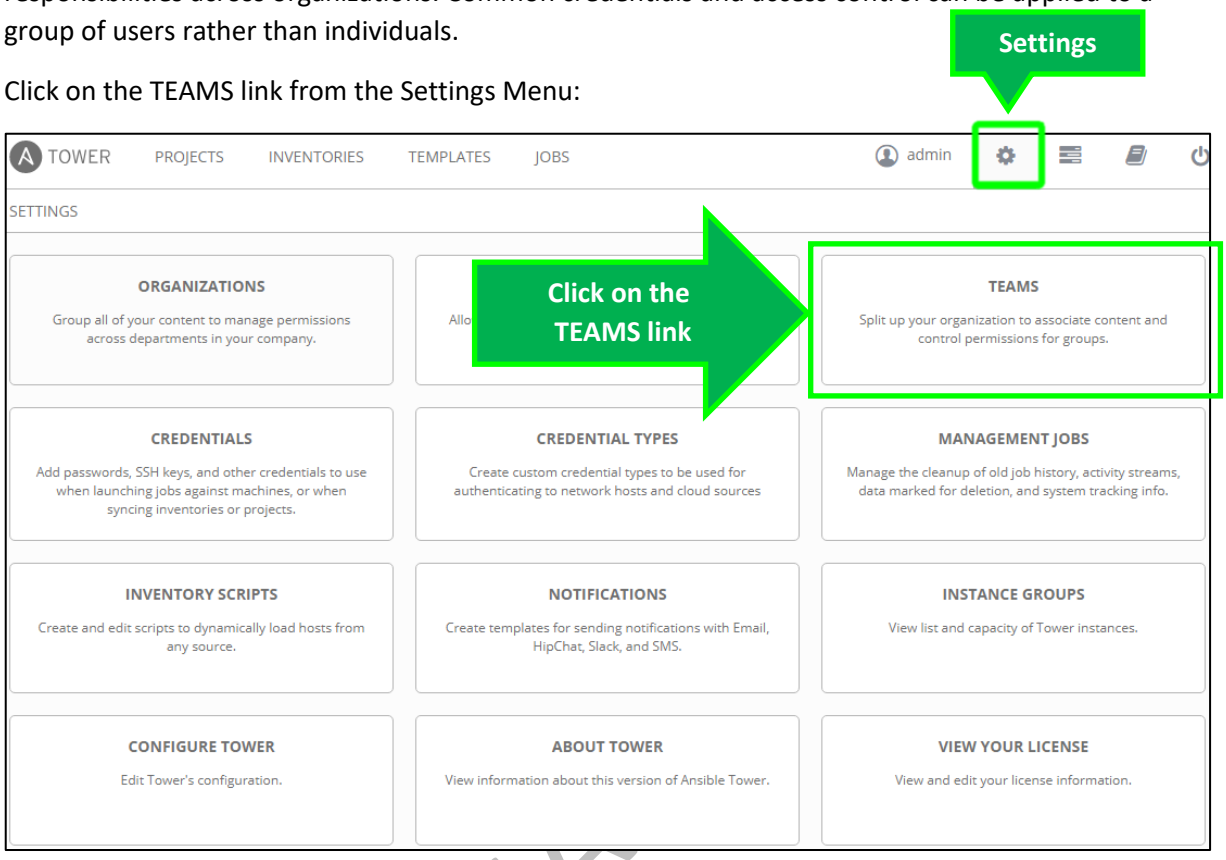
Fill up the various fields from the new user configuration form and click on SAVE. Remember the username and password you have created for this user as you will be logging in as this user later during the lab.



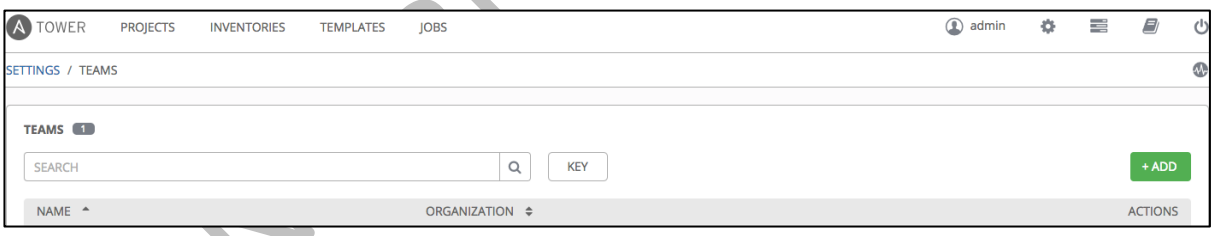
3.9 Creating an Ansible Tower Team

Teams provides a means to implement role-based access control schemes and delegate responsibilities across organizations. Common credentials and access control can be applied to a group of users rather than individuals.

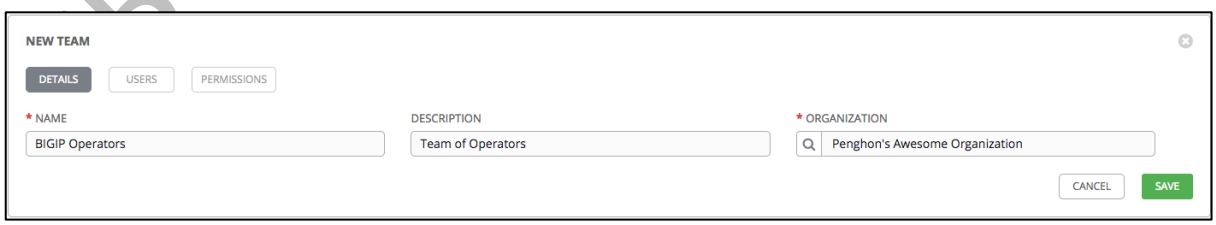
Click on the TEAMS link from the Settings Menu:



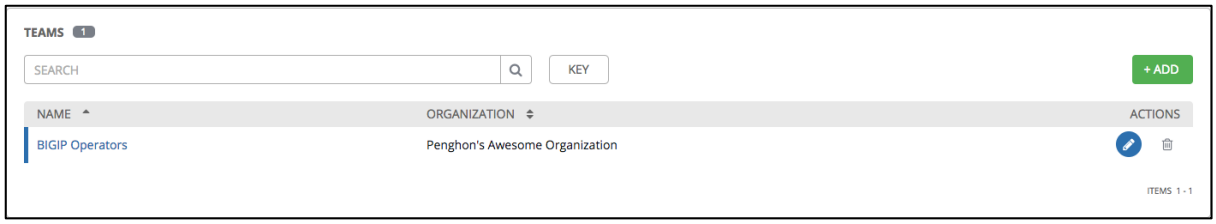
Select +ADD to create the Team:



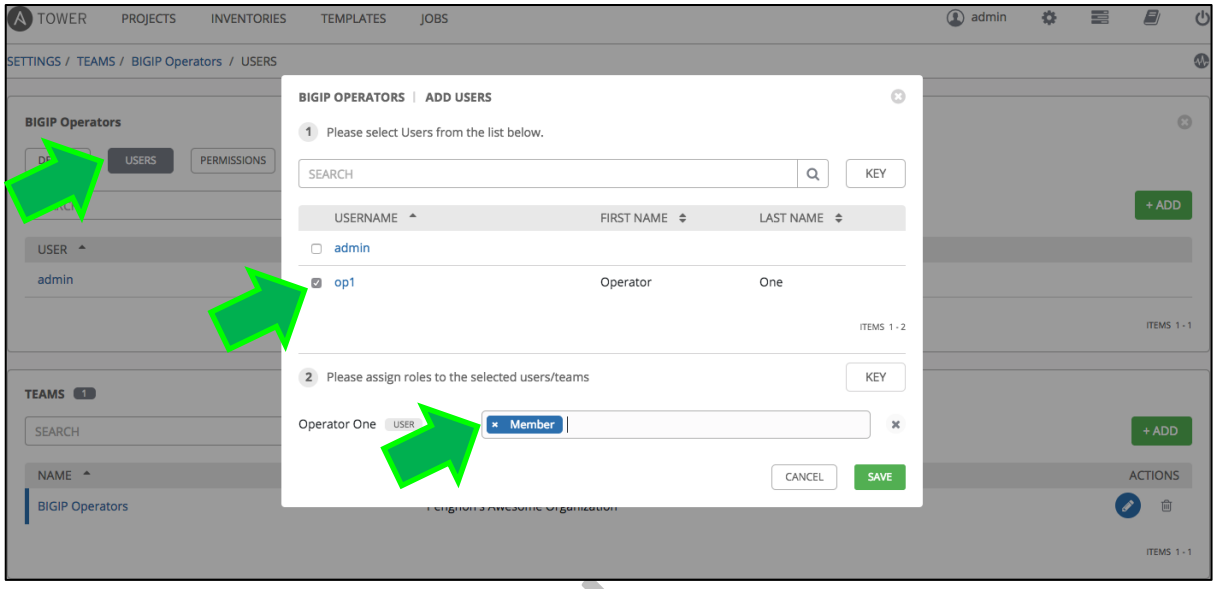
Fill in the *name* of your Team and select the *Organization* and click on SAVE:



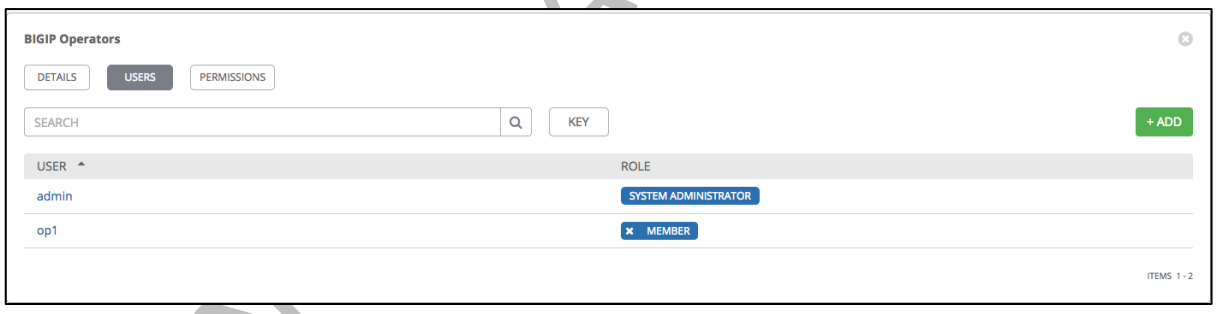
After adding the Team, it should be listed in the Team list:



Add the previously created Operator account as member of the Team:



The latest member of the Team should appear in the User list as shown below:

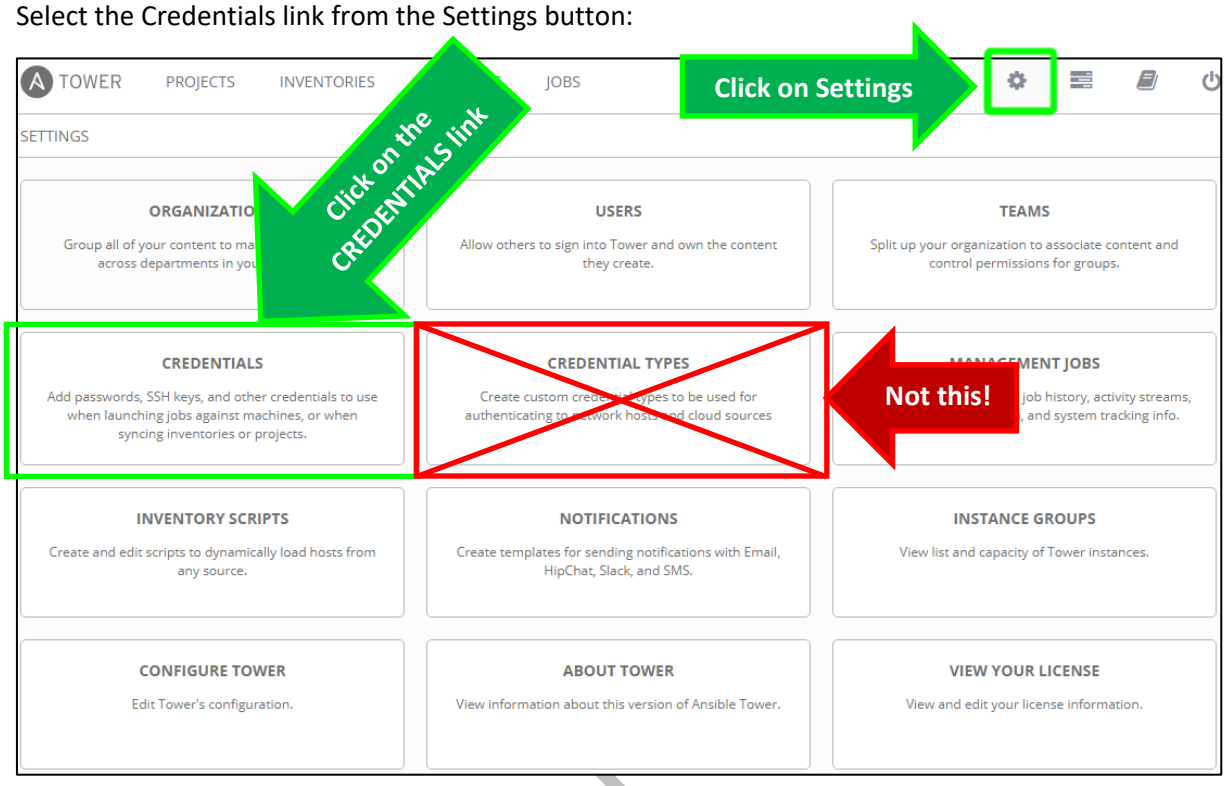


We will be adding permissions for the Team to the various objects such as Projects, Job Templates and Workflow Templates later in the lab.

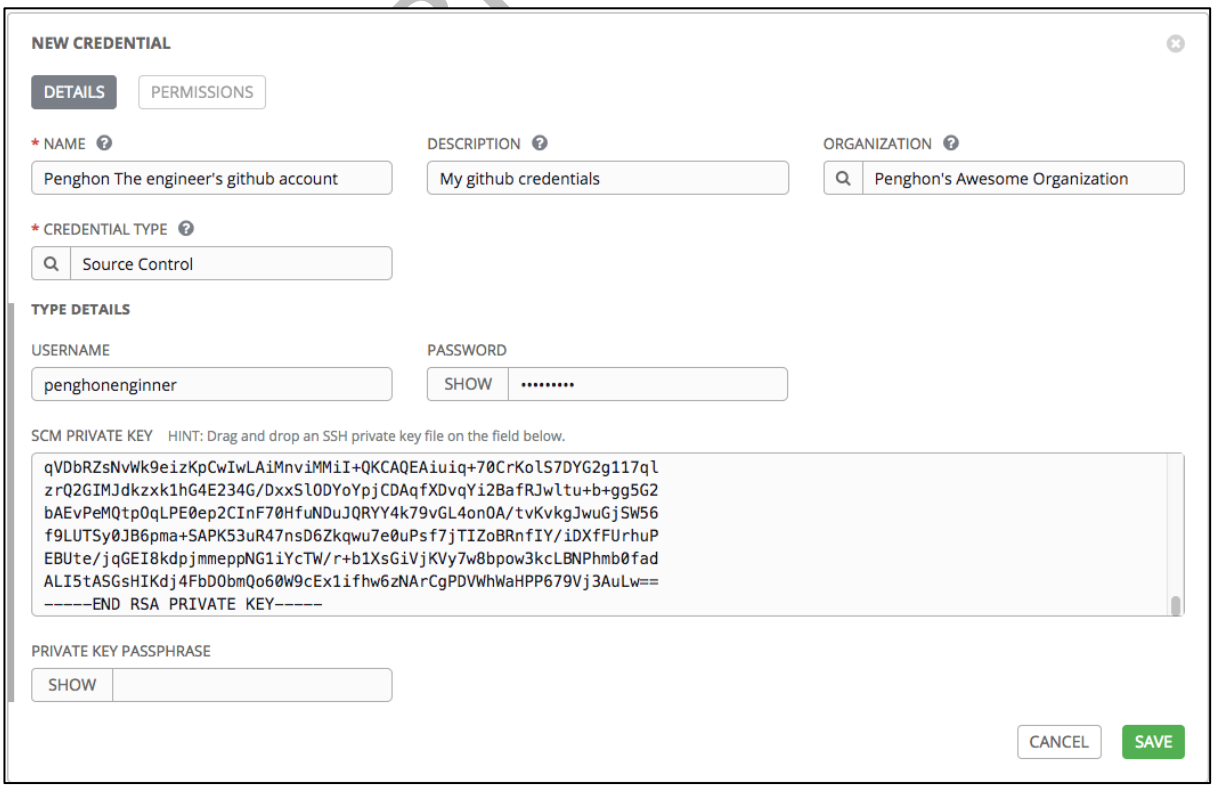
3.10 Ansible Tower Credential Creation for Source Control Management (Github)

Github will be used as the source of truth for the Project which you will create later.

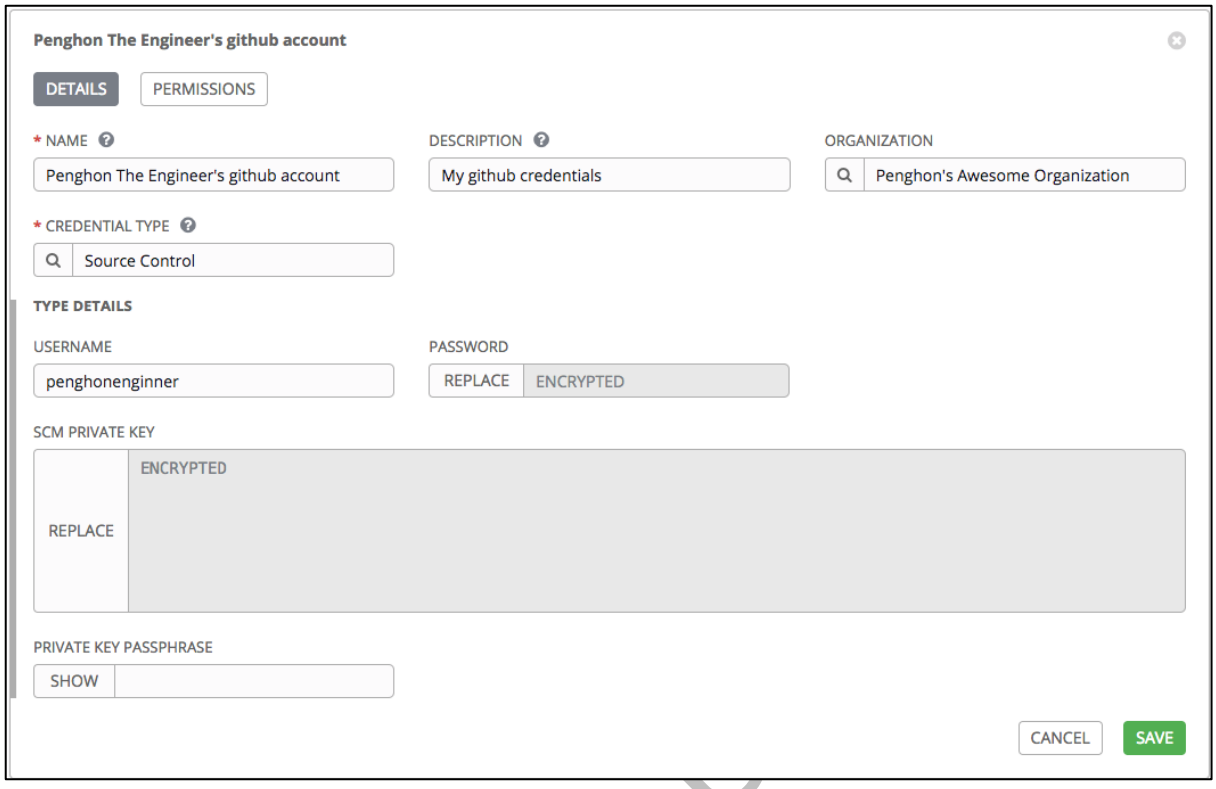
Select the Credentials link from the Settings button:



Select +ADD and fill in the various fields. Note that you will be storing the ***SSH private key*** from the keypair that you had previously generated, your ***Github username*** and ***GitHub password*** on your Ansible Tower:



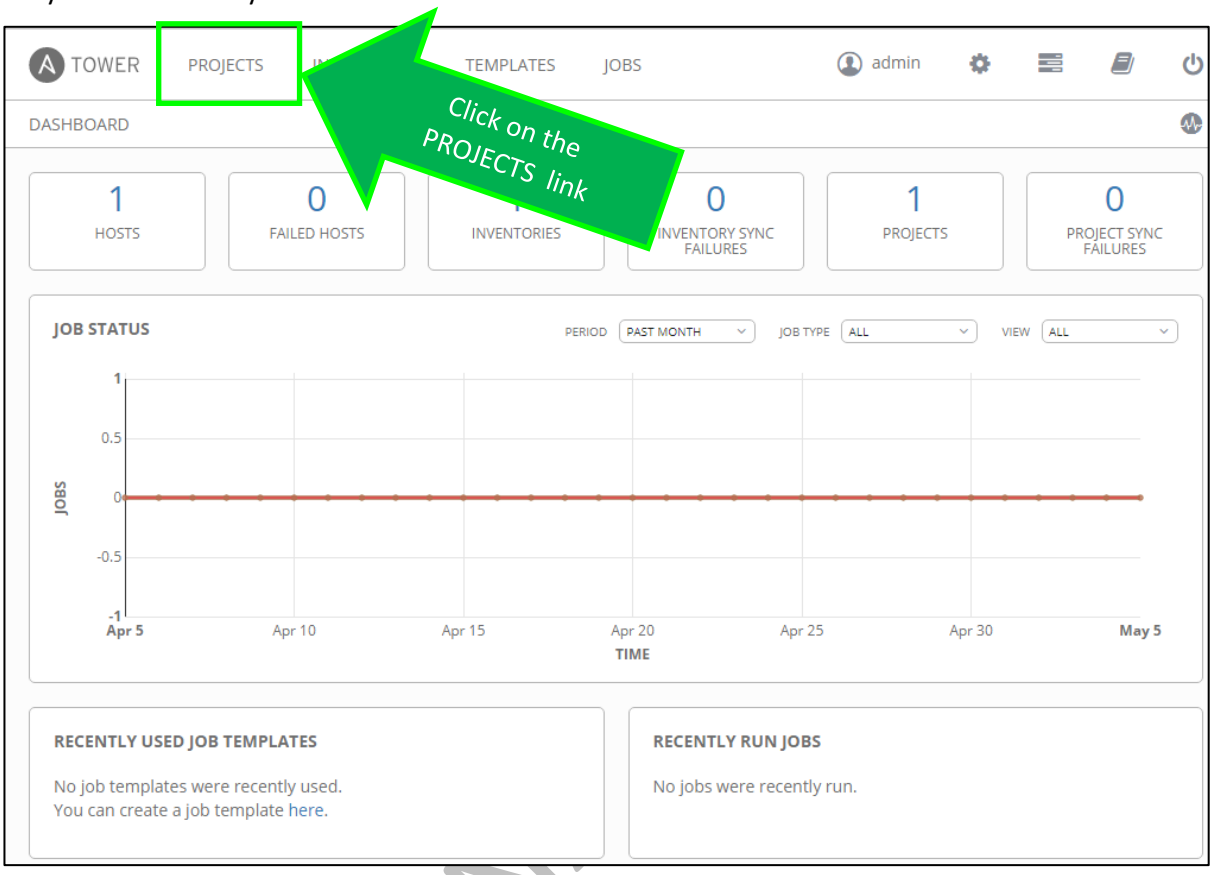
Click on SAVE once the various fields have been filled up:



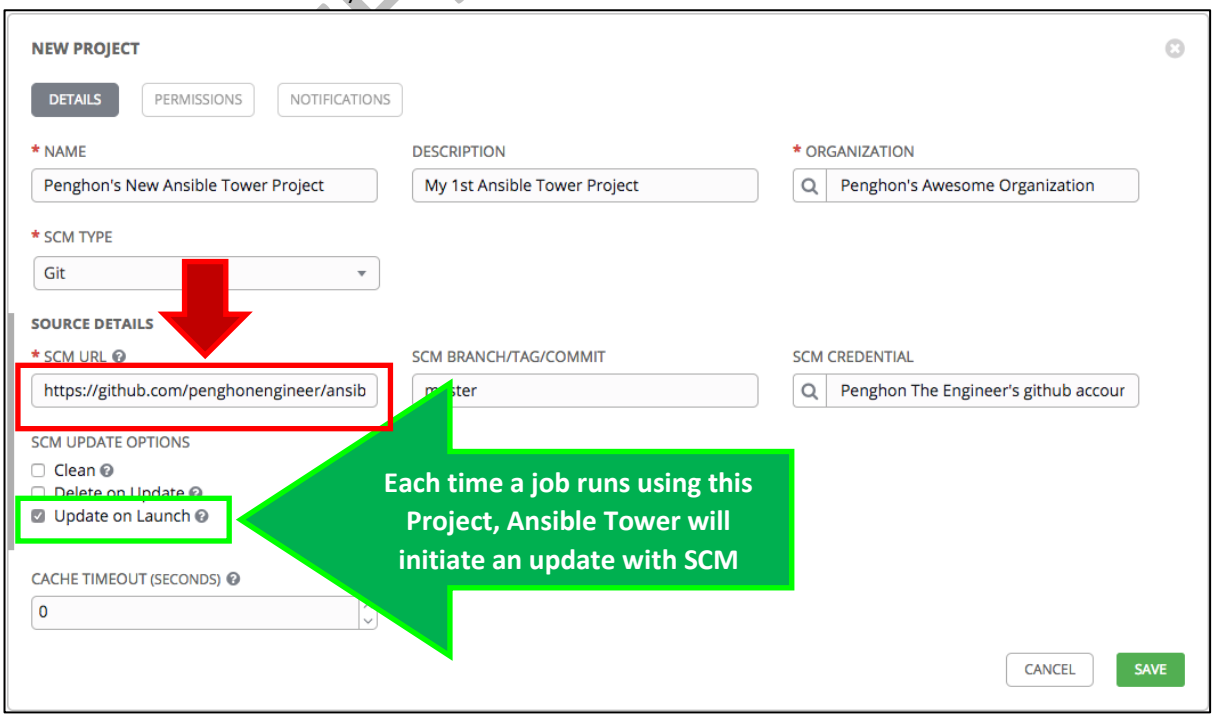
Notice that the Private Key and password field are encrypted immediately upon saving. Ansible Tower utilizes an encryption/decryption algorithm that uses a variation of Fernet: a symmetric encryption cipher utilizing AES-256 in CBC mode alongside a SHA-256 HMAC.

3.11 Ansible Tower Project Creation

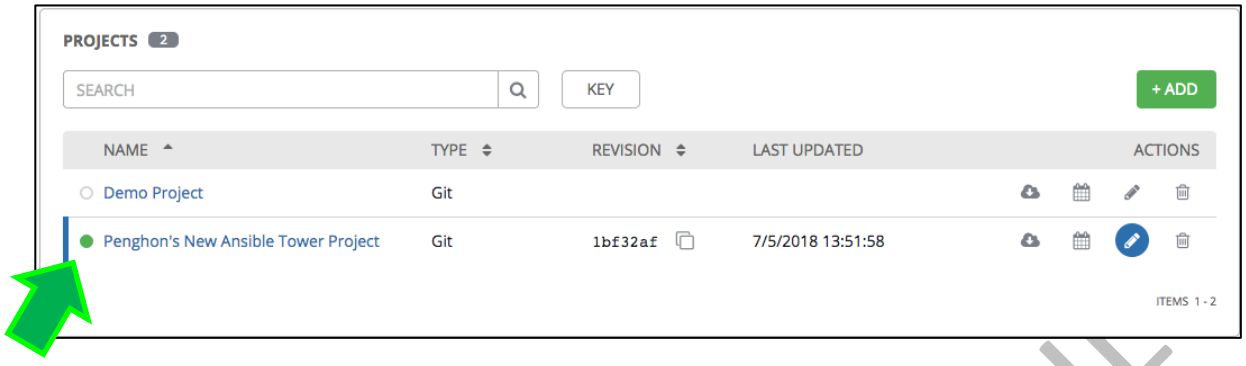
Project is a logical collection of Ansible playbooks, represented in Tower. Playbooks and playbook directories can either be source controlled with Git, Subversion, Mercurial or Red Hat insights or they can be manually added to Ansible Tower.

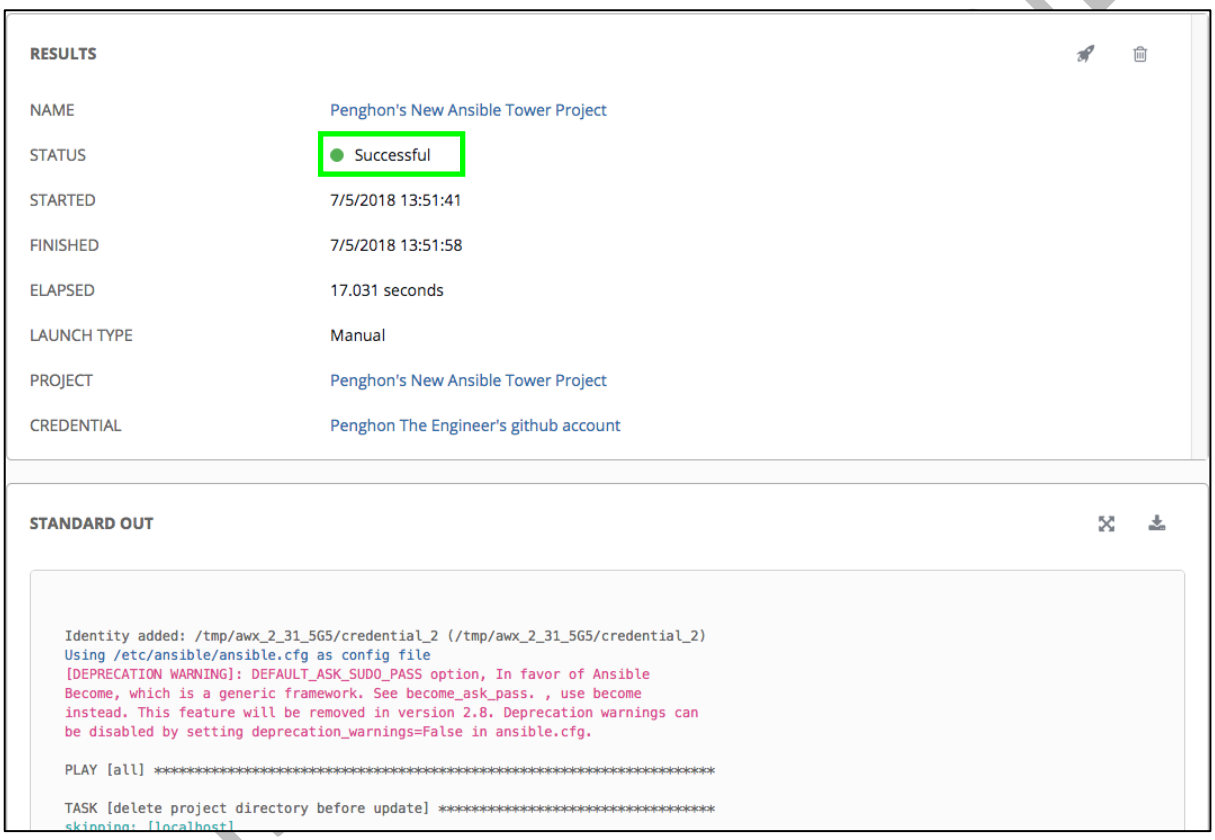


Select +ADD and fill in the various fields as follows (remember to substitute with your own URL in the field within the red box):



After saving the Project, Ansible Tower will sync and download the Playbooks and Inventory files from the SCM system (in this case Github). ***Click*** on the pulsating green dot to display the live updates of the initial sync between Ansible Tower and Github:





If the status shows successful, you have successfully validated and cloned the files from Github.

[Optional] To further validate whether Ansible Tower has successfully synced with Github, access your Ansible Tower Controller via SSH:

```
root@localhost ~]#
cd /var/lib/awx/projects/_6__penghons_new_ansible_tower_project/
[root@localhost _6__penghons_new_ansible_tower_project]# ls -la
```

3.12 [Optional] Adding a Project manually

In Ansible Tower the default path: */var/lib/awx/projects* is the "Project Base Path". As per the online help:

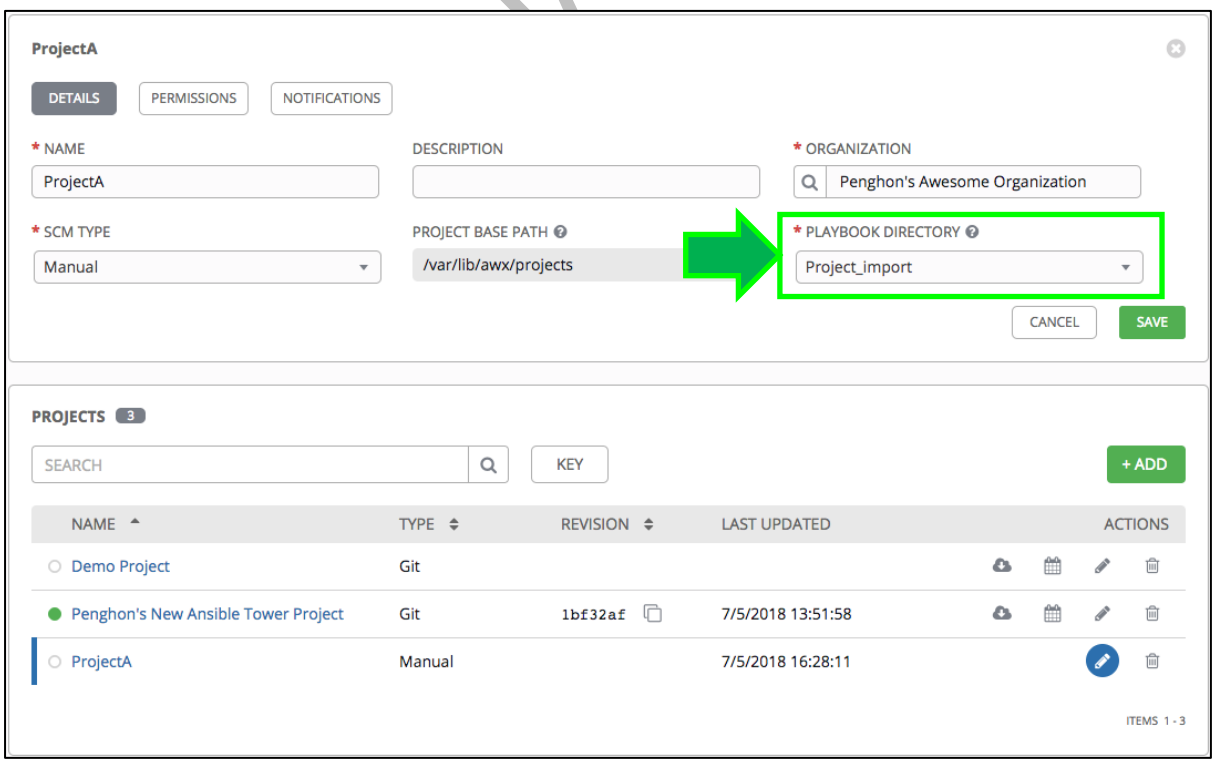
> *"Base path used for locating playbooks. Directories found inside this path will be listed in the playbook directory drop-down. Together the base path and selected playbook directory provide the full path used to locate playbooks. Change PROJECTS_ROOT under "Configure Tower" to change this location."*

Directories, under the ***PROJECTS_ROOT*** path, are considered projects. YAML files within the directories are considered as playbooks or inventories host files.

To manually import your playbooks and/or inventory files, create a directory under the default */var/lib/awx/projects* folder. Copy over the sample BIGIP playbook */var/tmp/sample_files/change_bigip_hostname.yaml* to your newly created directory. For example:

```
root@localhost projects]#
mkdir /var/lib/awx/projects/Project_import
root@localhost projects]# cp /var/tmp/sample_files/change_bigip_hostname.yaml /var/lib
/awx/projects/Project_import/.
root@localhost projects]# chown awx.awx -R /var/lib/awx/projects/Project_import/Project
_import/
```

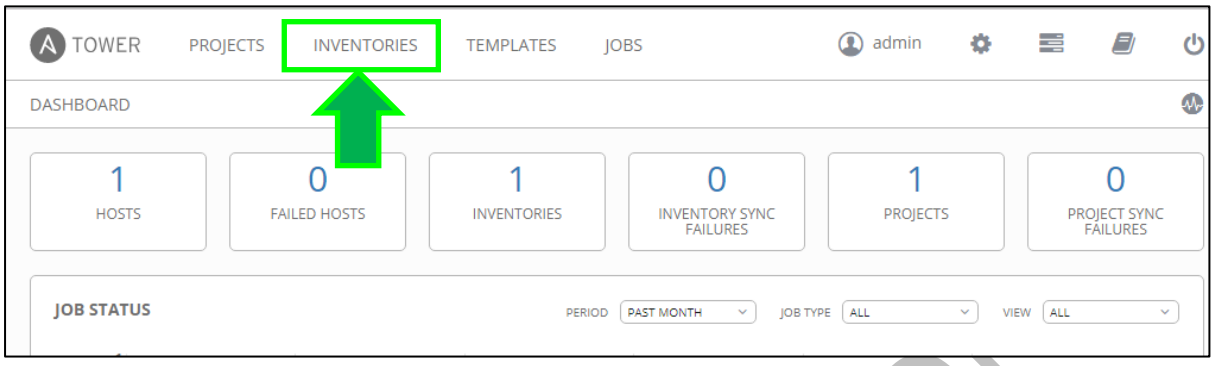
The Ansible Tower Project name does not have to be the same as the directory under the ***PROJECTS_ROOT*** folder. From Ansible Tower create a new Project and select the appropriate playbook directory:



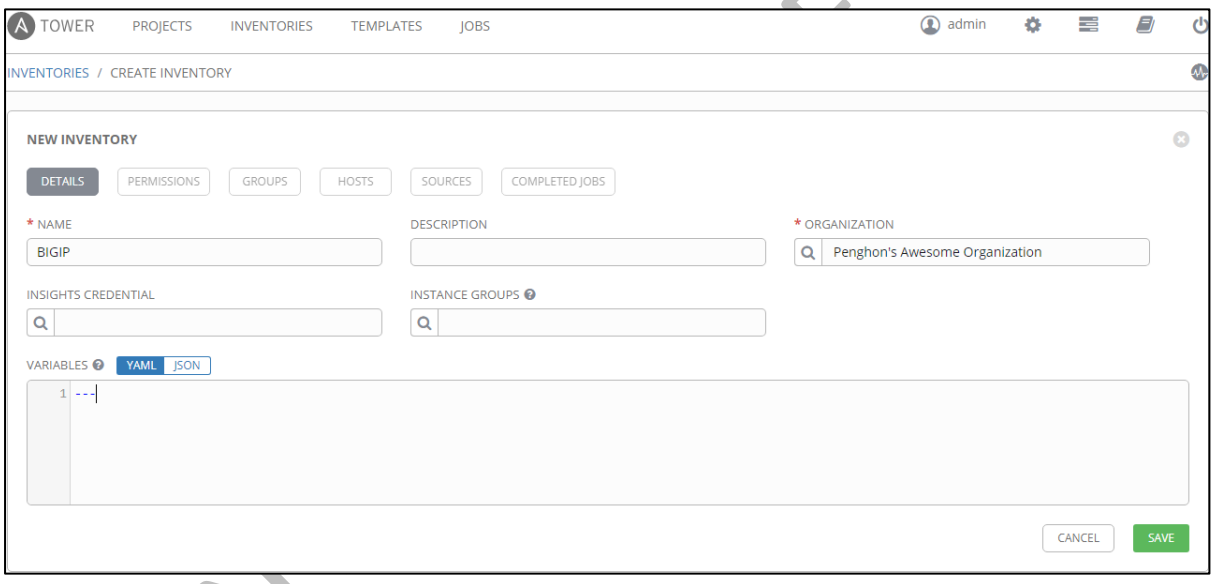
Once the new Project has been created, you will be able to use the inventory host files or the playbooks to create Job Templates.

3.13 Creation of an Ansible Tower Inventory

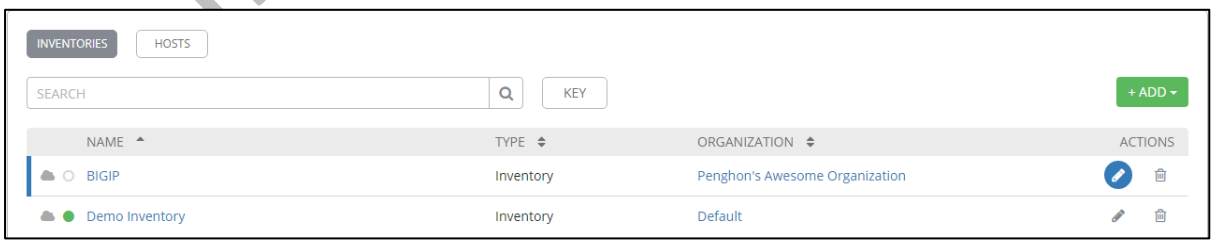
An Ansible Tower Inventory is similar in concept as an Ansible Engine inventory file. Select the INVENTORIES link as per below:



Click on +ADD, provide a meaningful name for the inventory and associate to the right organization. Click on SAVE:



Your inventory should look similar to the following:

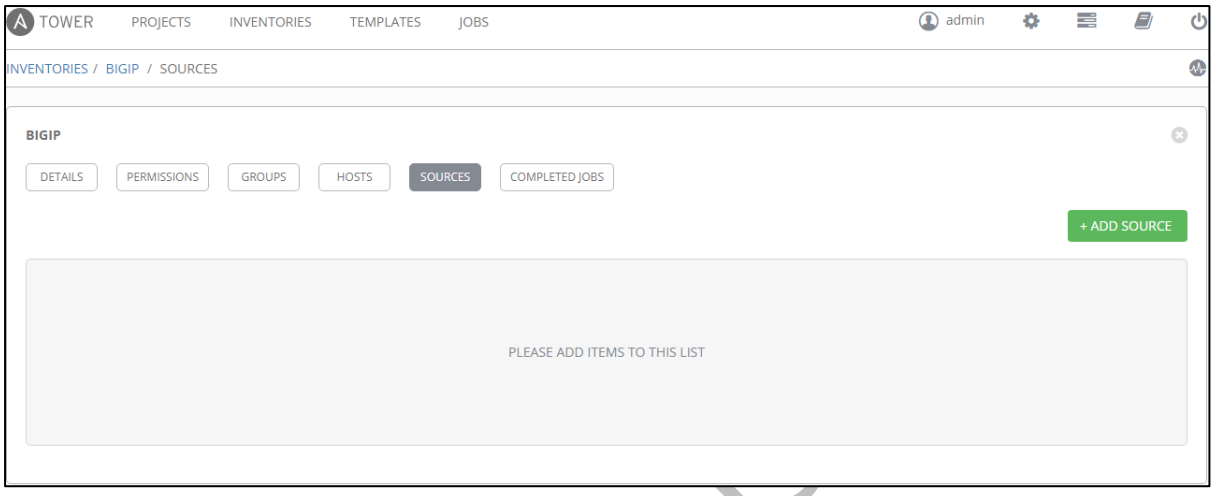


There are 5 types of permissions for Inventory:

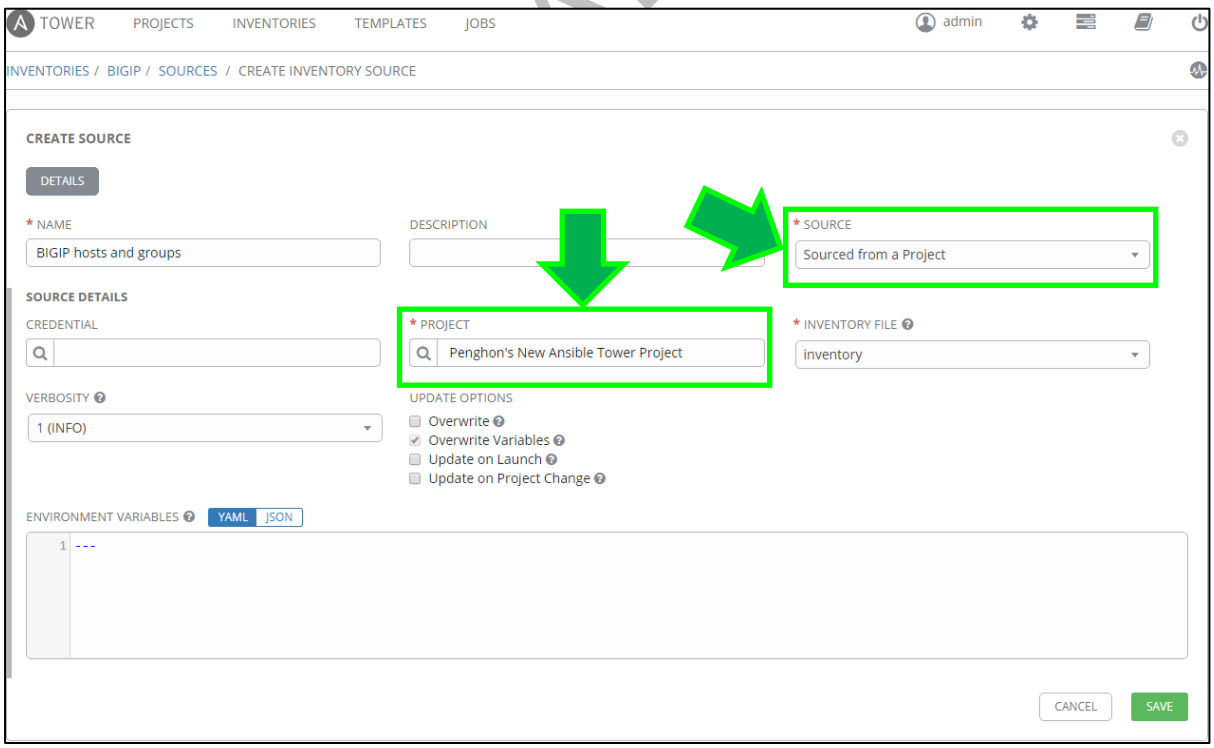
- **Admin** allows read, run, and edit privileges (applies to all resources)
- **Use** allows use of a resource in a job template (applies all resources except job templates)
- **Update** allows updating of project via the SCM Update (applies to projects and inventories)
- **Ad Hoc** allows use of Ad Hoc commands (applies to inventories)
- **Execute** allows launching of a job template (applies to job templates)

For the purpose of this Lab, the only permissions that matter would be applied to the Job Template, which is to allow push button permission to execute the Job Template. We will be covering this in the later section of the lab.

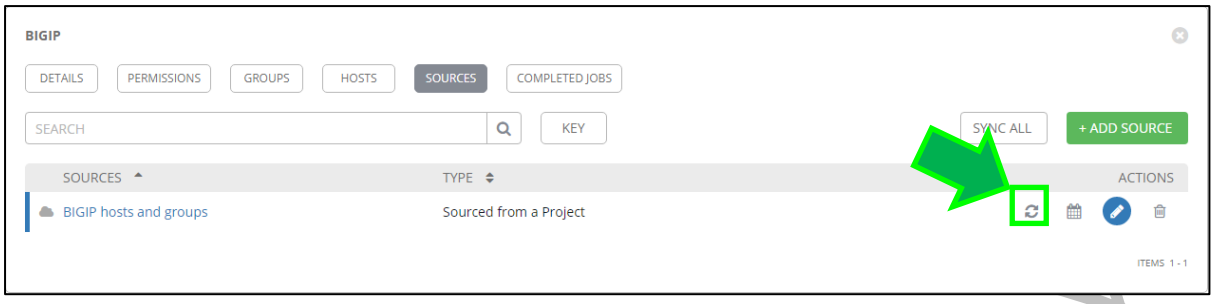
From your Ansible Tower, from the Inventory that you had created, select SOURCES and select the +ADD SOURCE:



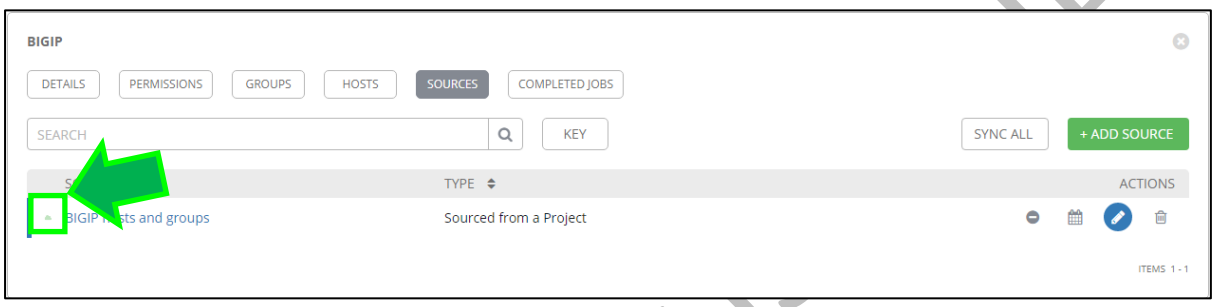
Configure source of the inventory file as “Source from a Project”, and select the Project that was created previously. The inventory file which was synced from Github previously will appear for selection:



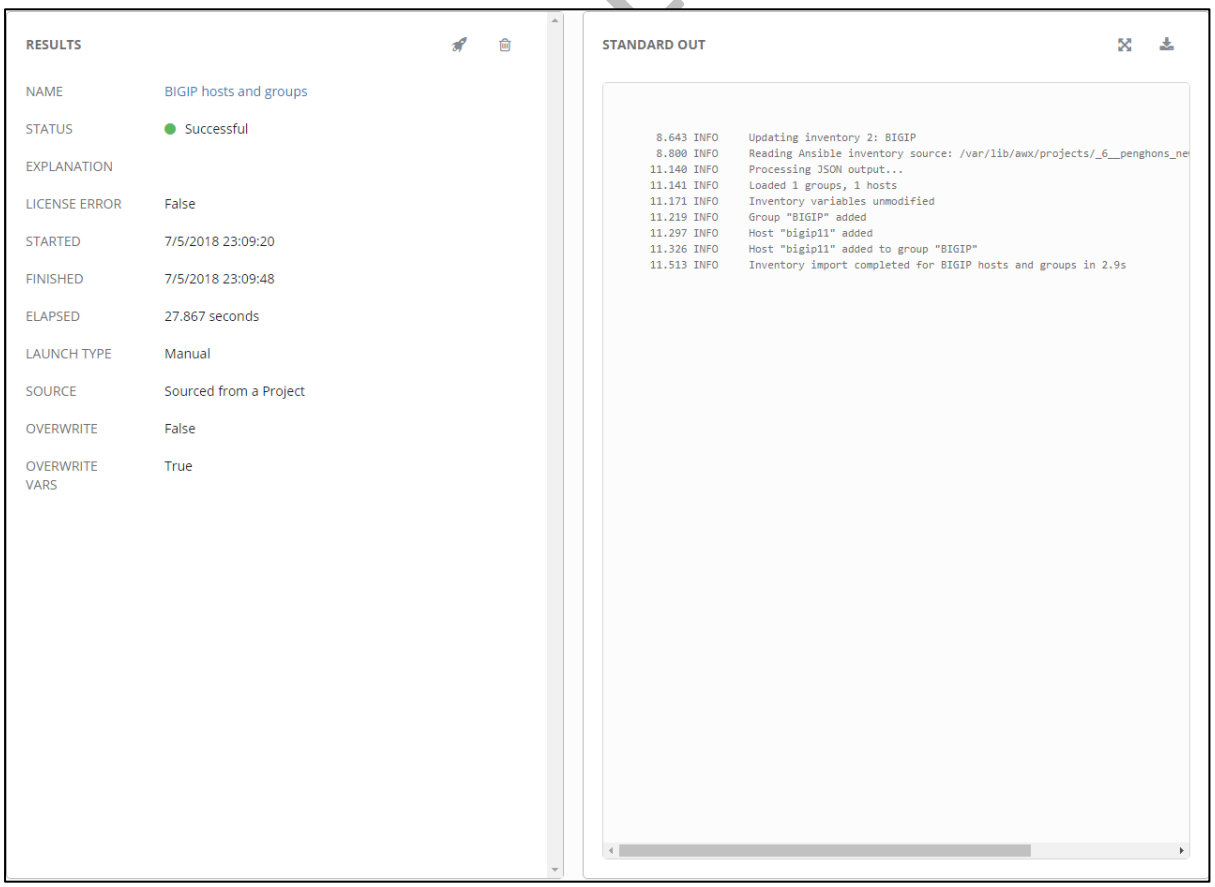
Click on SAVE, and click on the sync symbol as shown:



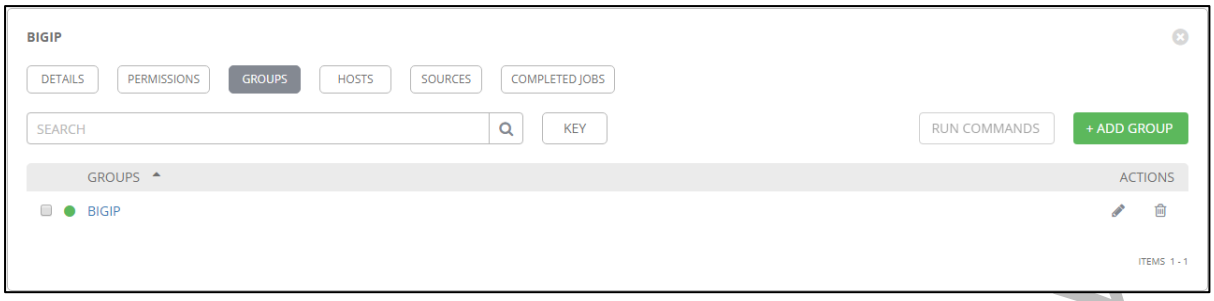
Click on the pulsing cloud as shown below to see the result of the job:

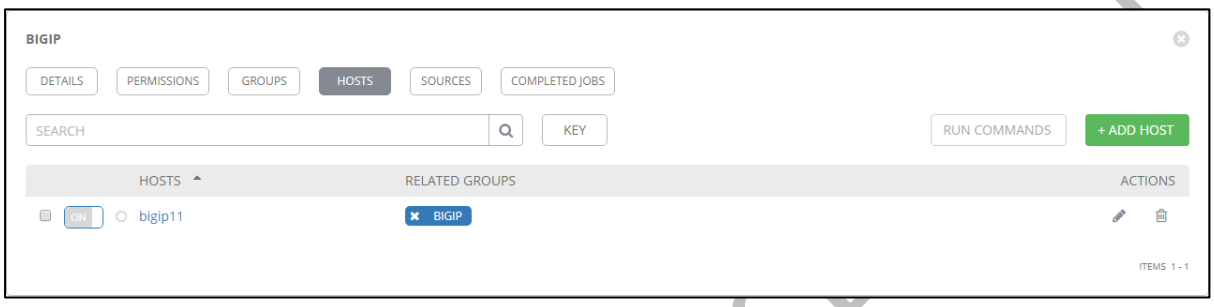


Examine the result of the inventory sync. Once successful, Ansible Tower will import the hosts from the source into your newly created Tower inventory:

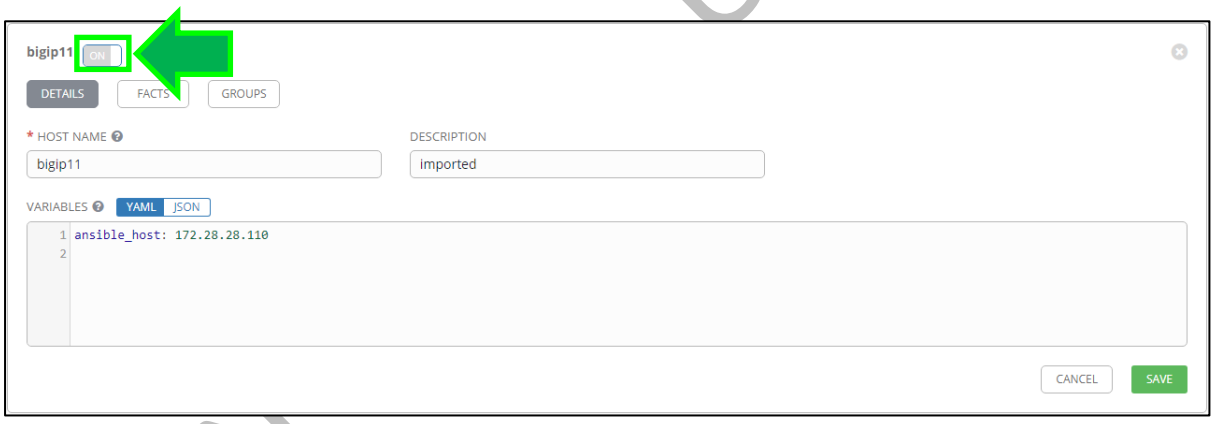


Validate that both group and host have been imported successfully:





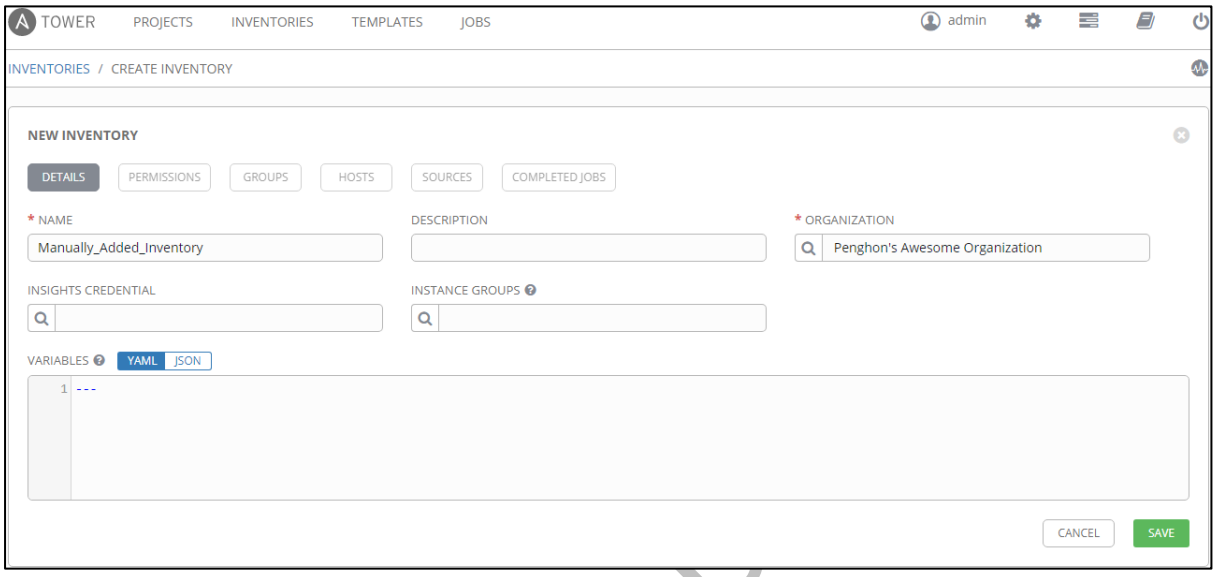
Click on the host link for further details. Notice that SCM controlled hosts cannot be turned off via Ansible Tower. Why?



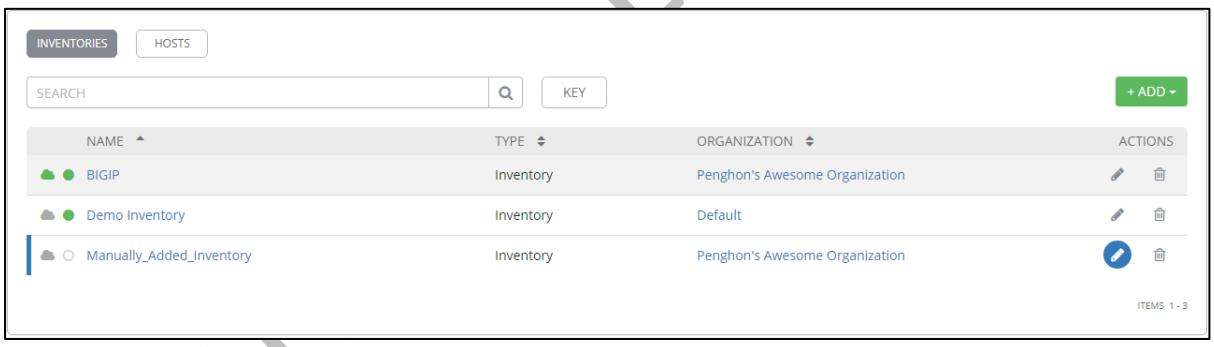
3.14 [Optional] Using awx-manage to manually import inventory file

For the purpose of this lab task you will be using awx-manage to manually import the inventory file to your inventory hosts.

Create the inventory on Ansible Tower:



Result:

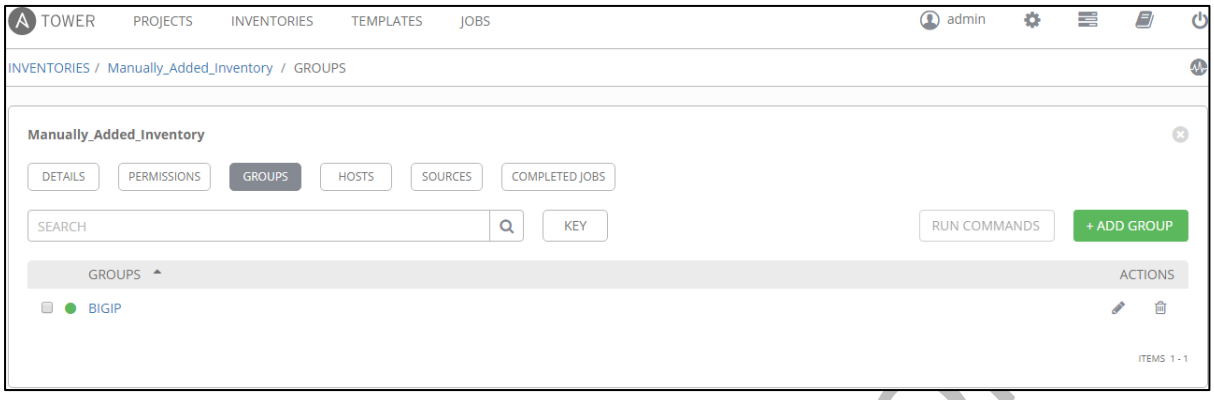


Copy the sample inventory file ***/var/tmp/sample_files/inventory*** into the previously created project folder ***/var/lib/awx/projects/Project_import/*** :

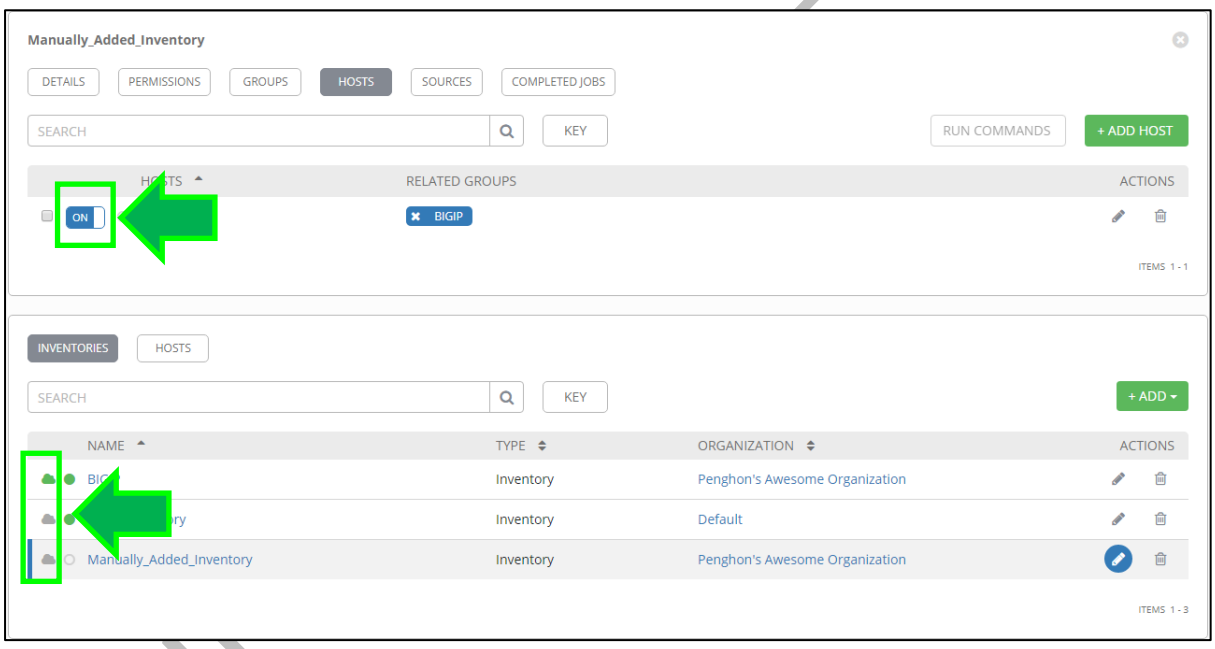
```
[root@localhost Project_import]# awx-manage inventory_import --source /var/lib/awx/pro
jects/Project_import/inventory --inventory-name Manually_Added_Inventory
    3.916 INFO     Updating inventory 3: Manually_Added_Inventory
    4.055 INFO     Reading Ansible inventory source: /var/lib/awx/projects/Project_import/i
nventory
    5.640 INFO     Processing JSON output...
    5.641 INFO     Loaded 1 groups, 1 hosts
    5.649 INFO     Inventory variables unmodified
    5.670 INFO     Group "BIGIP" added
    5.688 INFO     Host "bigip11" added
```

```
5.706 INFO    Host "bigip11" added to group "BIGIP"
5.824 INFO    Inventory import completed for  (Manually_Added_Inventory - 9) in 1.9s
```

Check that the group and host are added into the ***Manually_Added_Inventory*** inventory



Notice that a manually added host can be switched off and cannot be synced:



3.15 Modification of playbook to use Ansible Vault

We will be using the Ansible Vault encrypt_string to create a single encrypted variable which is the password for the admin user on BIGIP.

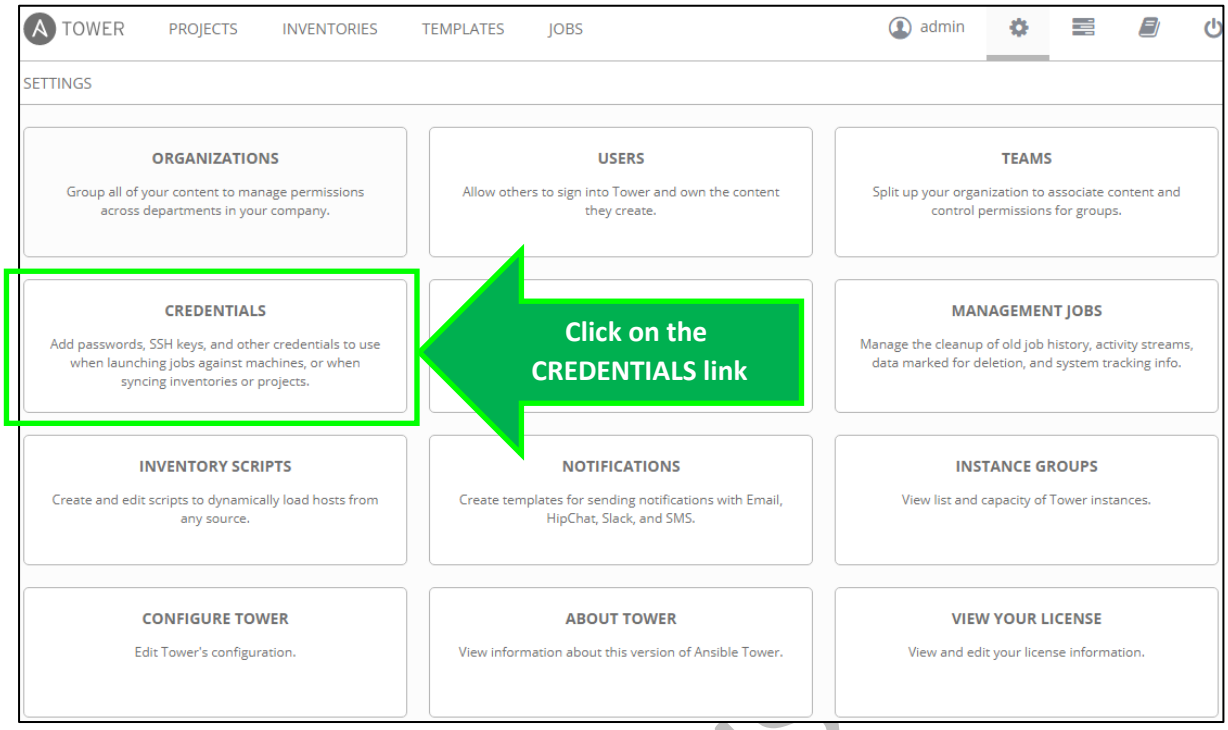
SSH to your Ansible Tower Controller and run the following:

```
[root@localhost ~]# echo -n 'admin' | ansible-vault encrypt_string
New Vault password:
Confirm New Vault password:
Reading plaintext input from stdin. (ctrl-d to end input)
!vault |
          $ANSIBLE_VAULT;1.1;AES256          396664323636363138323631613364396638323766663131346239343663663937396664333565616636613634613866383837303431336639333664623738 0a353235333063656566253838353632          666364343466383539663030313631663931353166623764643031643164346634323766303438393          336230313133363465 0a396439343361333331383831313665656166633663565393165383136623763333
Encryption successful
```

Remember the vault secret that was used to encrypt the BIGIP admin password and copy out the highlighted lines in red to a text editor.

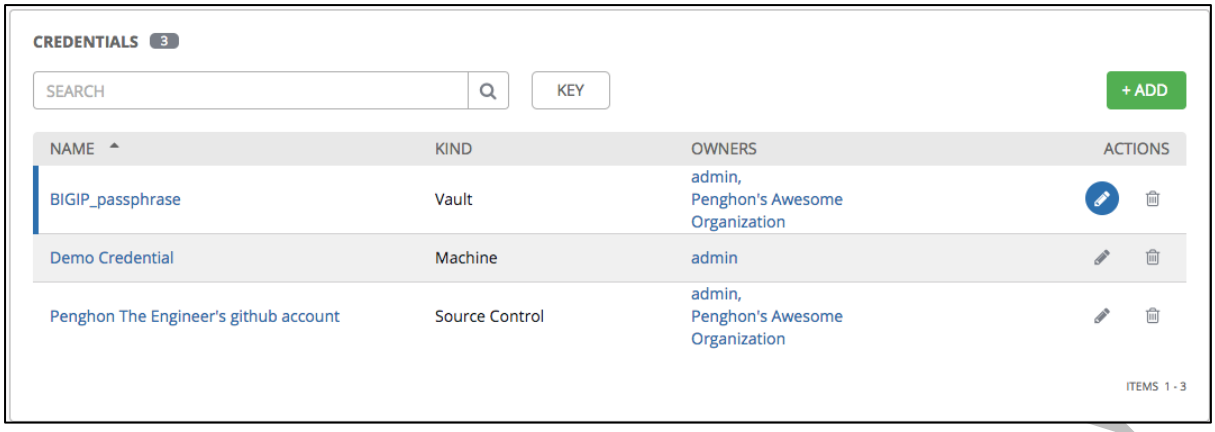
Ansible Vault uses AES shared secret to encrypt the string. The above represents the encrypted form of your BIGIP password '***admin***'. You will now need to create an Ansible Vault credential on Ansible Tower with the shared secret.

From Ansible Tower, select the Credentials link:



Create a new credential object and use the pre-defined Credential Type of Vault. Use the vault secret that you had keyed in earlier:

Select SAVE and the newly created credential will appear similar to what you see below:



We will now need to modify the existing playbook to use the encrypted variable:

From your local git repository, modify and add the following lines highlight red below from the ***/git/ansible-repo/change_bigip_hostname.yaml*** file:

```
---

- name: Testing importing group and host vars via Tower
  hosts: all
  gather_facts: no
  connection: local
  vars:
    bigip_admin_credentials: !vault |
          $ANSIBLE_VAULT;1.1;AES256                396664323636363138323631336433
966383237666631313462393436636639373966643335656 1          6636366136346138
663838373034313366393336646237380a353235333063656565623838353632666364
343466383539663030313631663931353166623764643031643164346634323766303 4
383933623031313336346 50a396439343361333331383831313665656166336635653 9
316538313662376333
```

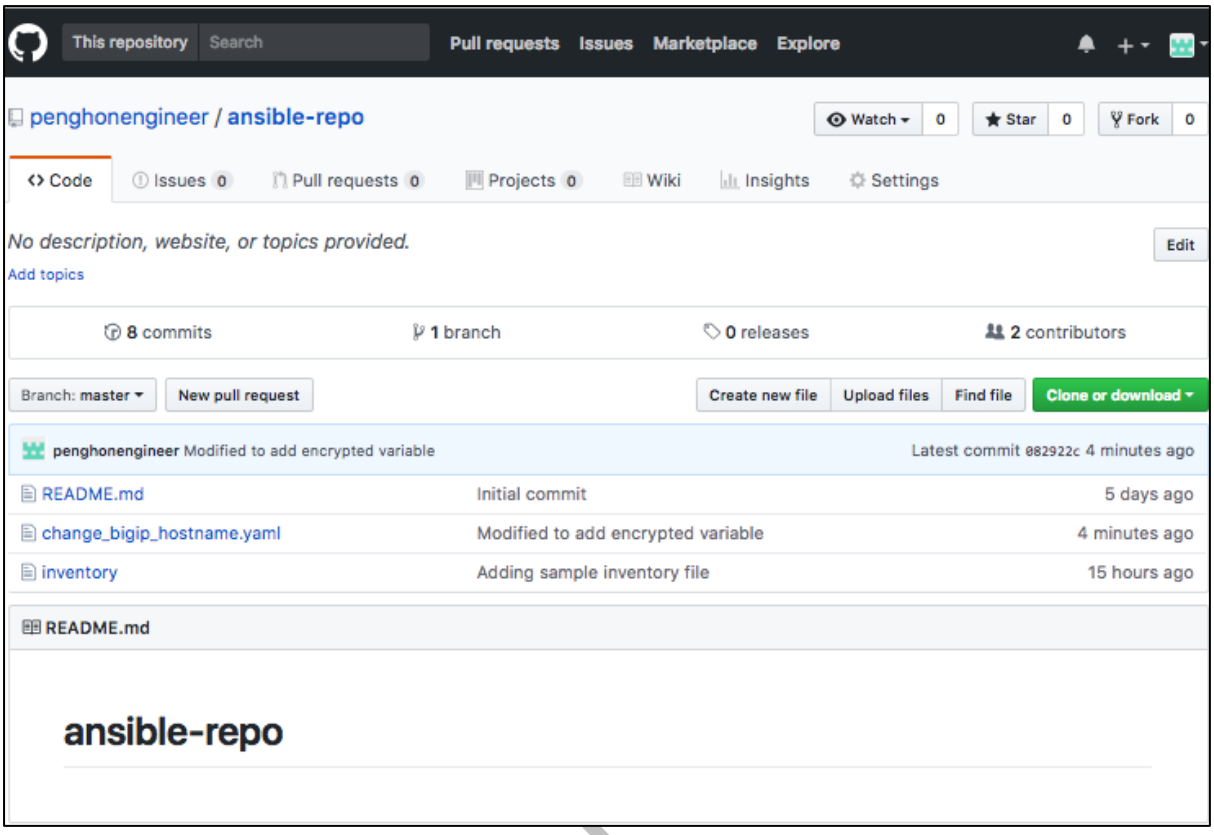
```
  tasks:
    - name: Setting the hostname of the target BIGIP
      bigip_hostname:
        hostname: "{{ hostname }}"
        provider:
          user: admin
          server: "{{ ansible_host }}"
          password: "{{ bigip_admin_credentials }}"
          validate_certs: no
```

After modification, commit the file changes and push to Github:

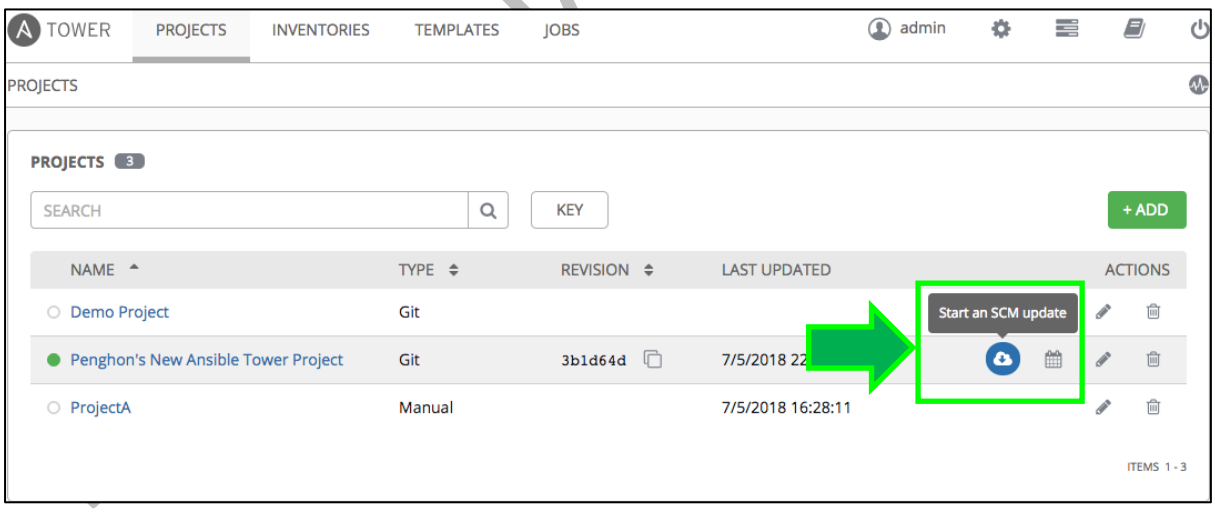
```
[root@localhost ansible-repo]# git commit -m "Modified to add encrypted variable" chang
e_bigip_hostname.yaml
[master 082922c] Modified to add encrypted variable
 1 file changed, 9 insertions(+), 1 deletion(-)
[root@localhost ansible-repo]# git push
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 798 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:penghonengineer/ansible-repo.git
   3b1d64d..082922c  master -> master
```

Repeat the above steps for ***/git/ansible-repo save_config.yaml***

Validate that the file changes have been made on the playbook on Github:



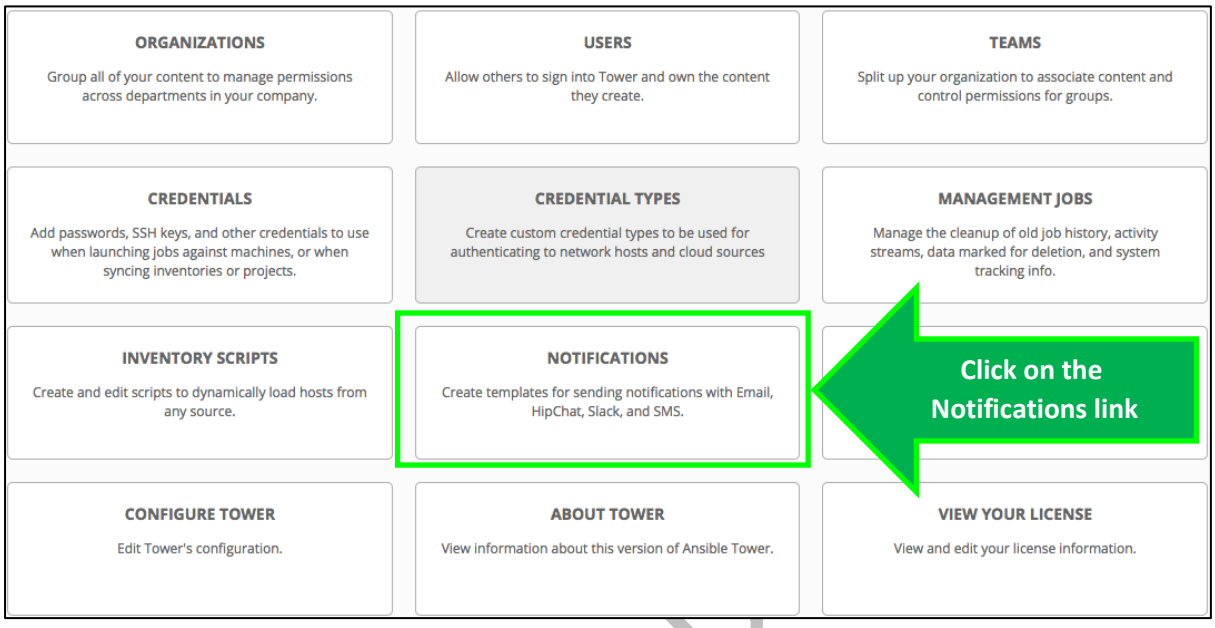
Issue the SCM update from Ansible Tower:



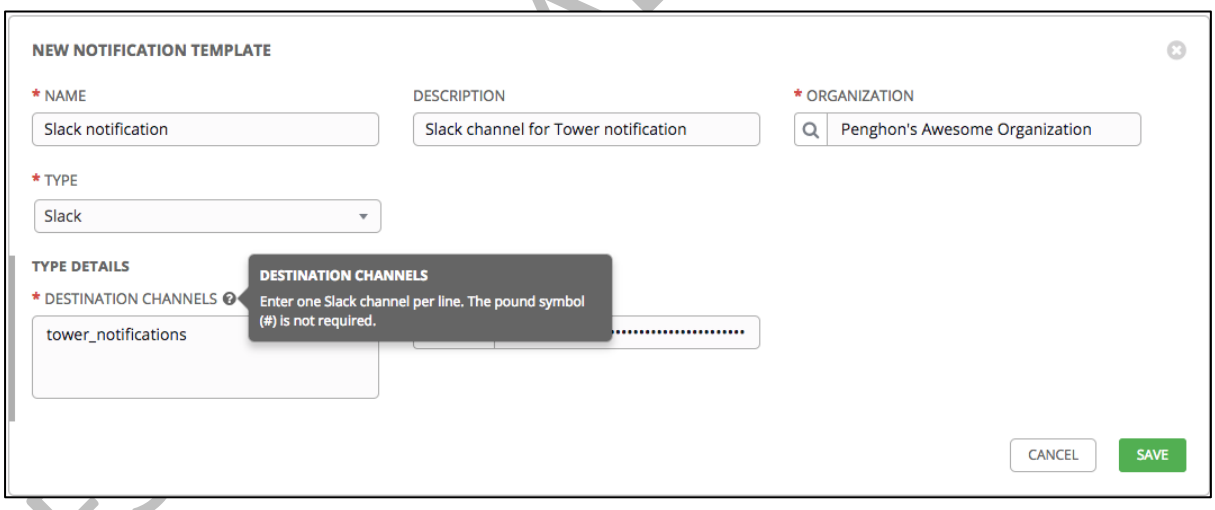
3.16 Ansible Tower Notification Configuration with Slack

You will be configuring a Slack notifier so that you can be notified of the job status whenever a job template completes.

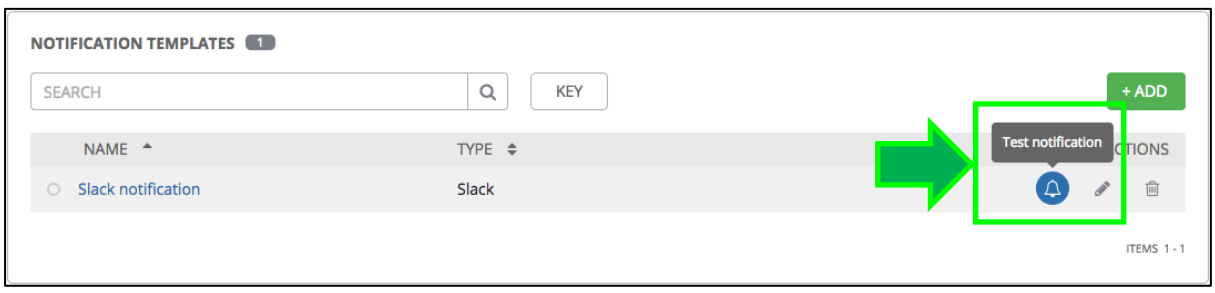
Select NOTIFICATIONS from Settings:



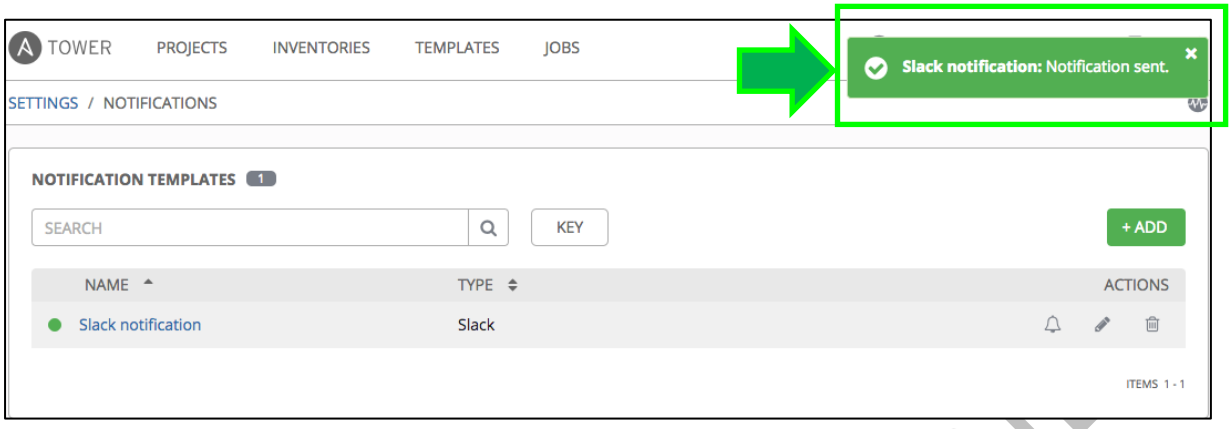
Select +ADD and fill in the fields and ensure that slack is selected as the type.
The destination channel is the previous channel that you had created in Slack:



Click SAVE and the following should appear. Select the “test nofitication” button:



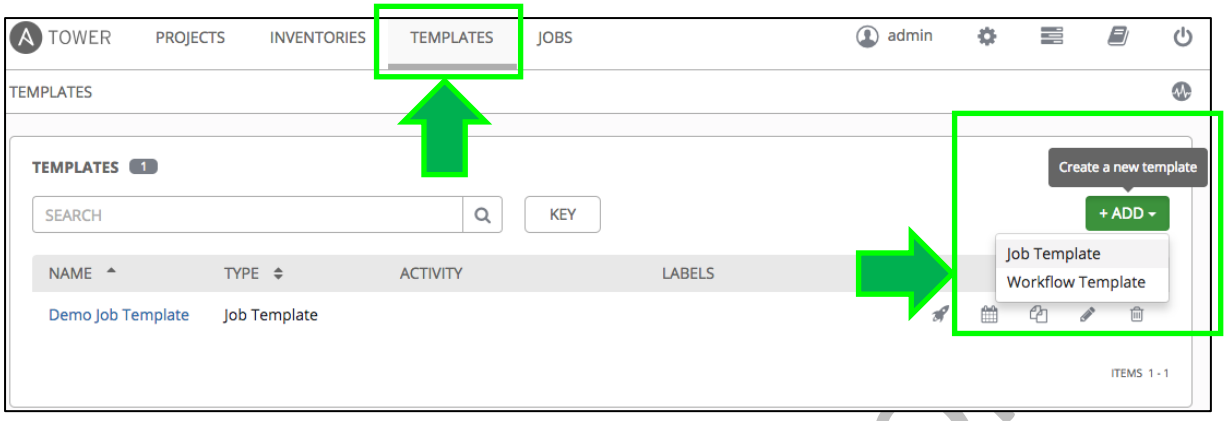
Ansible Tower will attempt to send a test message to your configured Slack channel. Once Tower is able to send the message, the following popup will be seen:



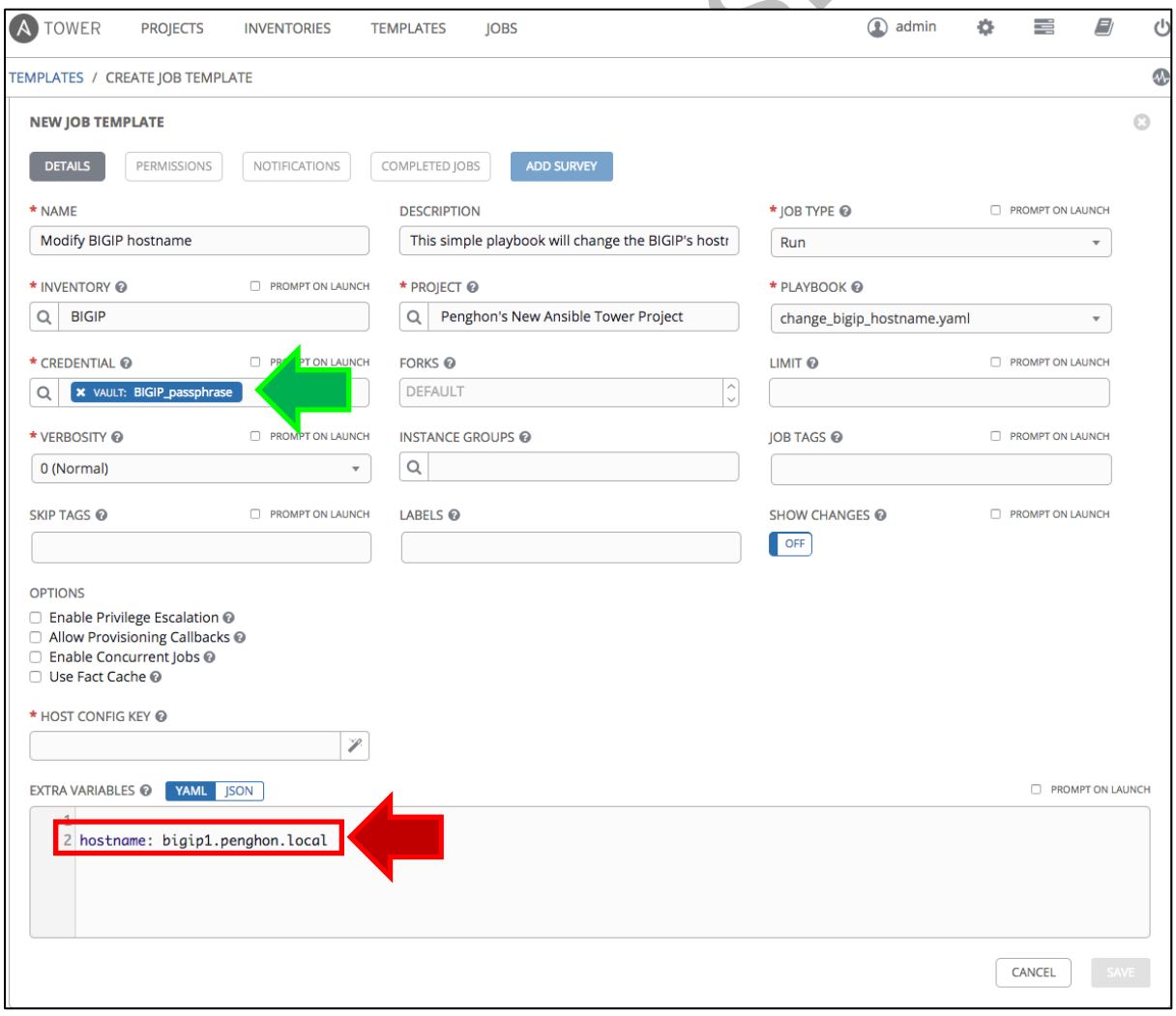
Check your Slack channel that you have received the test message from Ansible Tower:

3.17 Ansible Tower Job Template Configuration

An Ansible Tower job template defines playbook execution with RBAC permissions, reusability with surveys, job notifications, and several other parameters. From the Ansible Tower dashboard, select TEMPLATES and +ADD to create a Job Template:

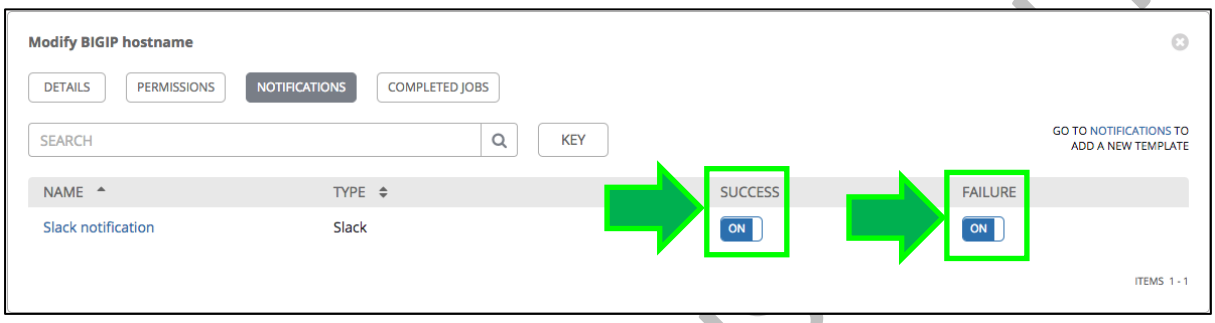


Fill in the fields as shown below and select the ***change_bigip_hostname.yaml*** Playbook, and the previously created vault credentials. In the EXTRA VARIABLES section, key in the ***hostname*** variable and replace the value with a hostname of your own choosing as shown below:

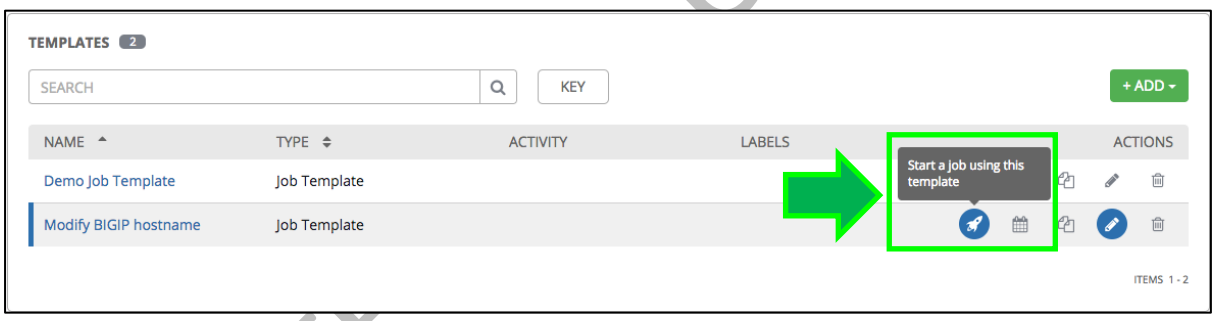


Select SAVE and the new Job Template will be shown as below:

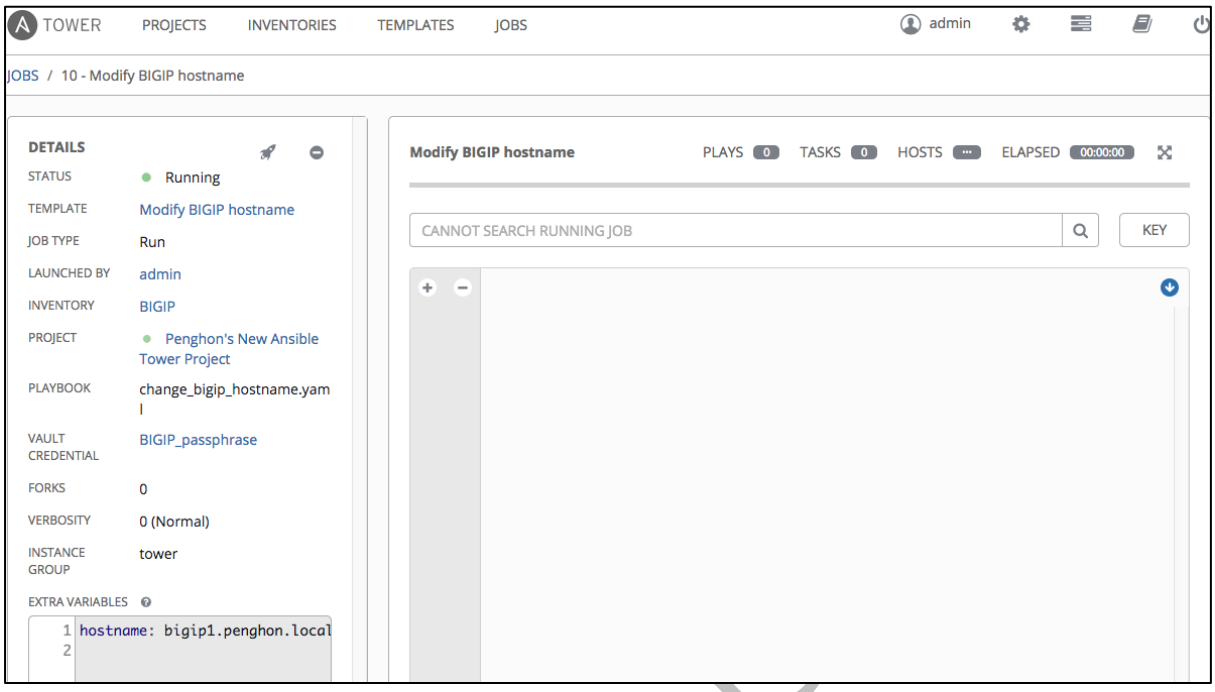
Modify the inherited notification to notify you when the Job instance is successful or when it failed:



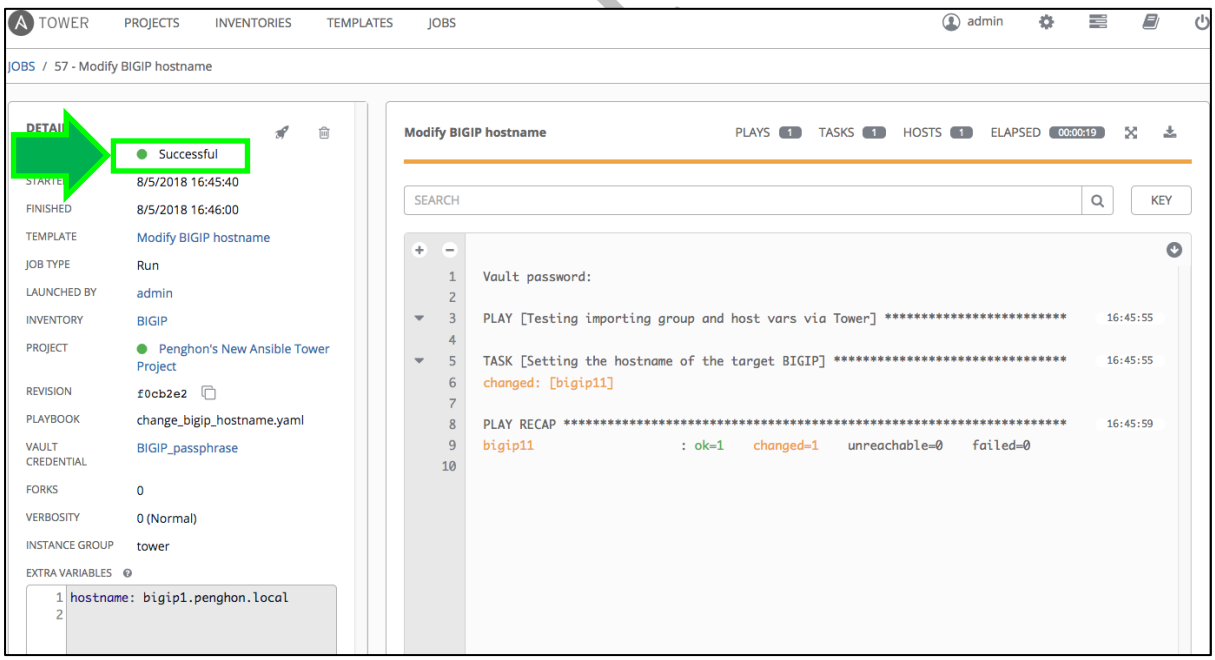
Launch the template job by selecting the rocket icon:



Ansible Tower will switch to the live update of the Job instance after the Job Template has been executed. Notice that the playbook hasn't been run. Why?



Upon successful execution of the job you would see the standard output as you would run ansible-playbook via Ansible Engine.

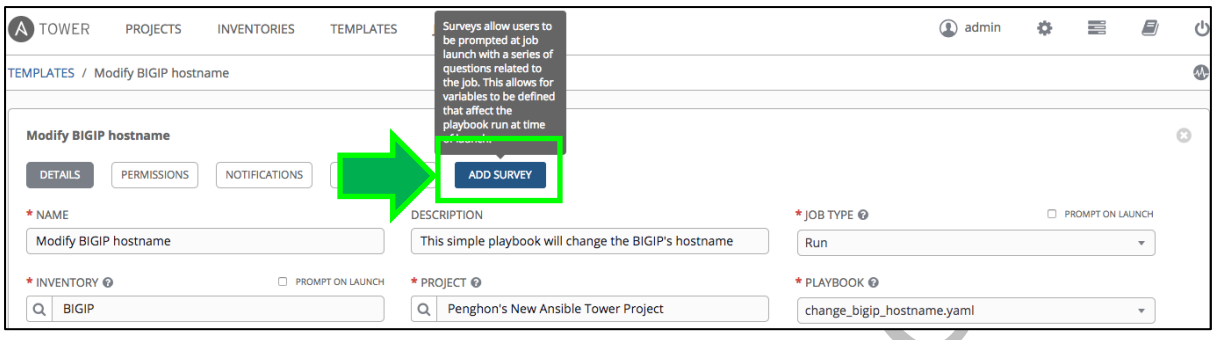


Check that your slack channel has the job success/failure notification.

3.18 Configuring Ansible Tower Surveys

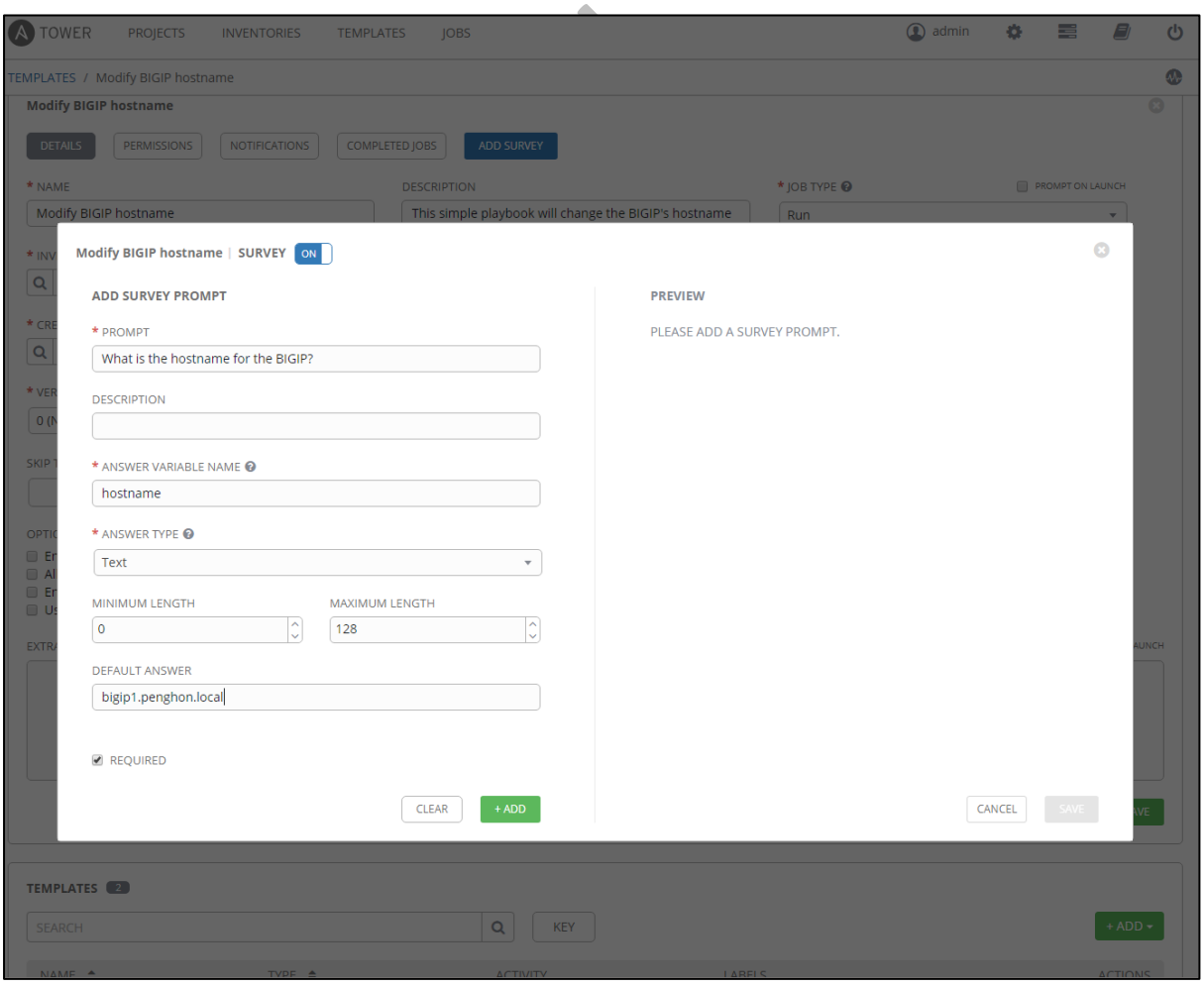
Ansible Tower Surveys only available for Enterprise-level licenses. Surveys set the extra variables but in a user-friendly question-and-answer manner. You can set limits or user input validation or create multiple-choice questions.

Access the previously created Job Template and click on ADD SURVEY:

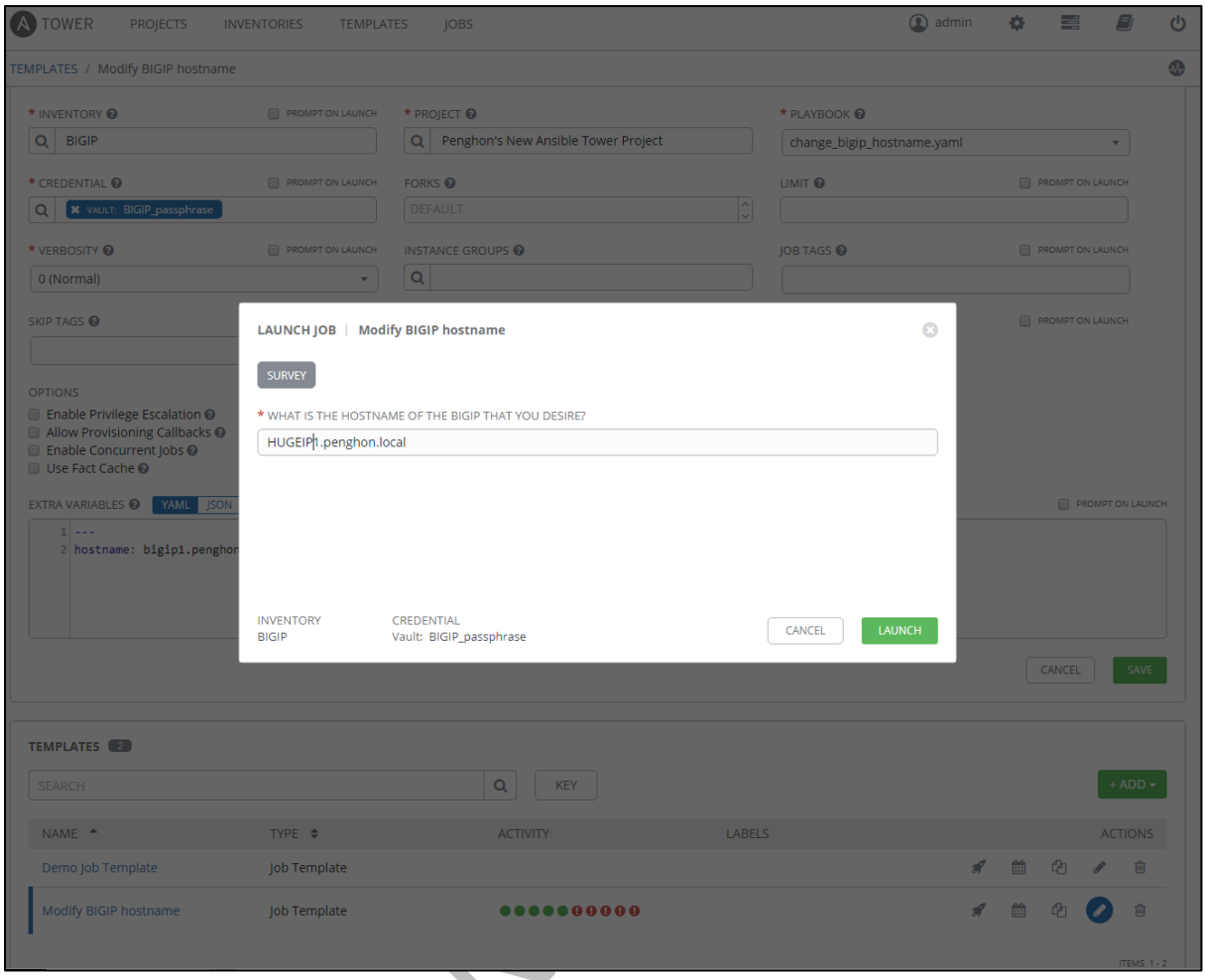


The sample playbook ***change_bigip_hostname.yaml*** sets the BIGIP hostname using the bigip_hostname module. This module requires the ***hostname*** parameter. We will use this to demonstrate the purpose of the Ansible Template Survey.

Fill in the fields below to create a user-friendly survey that asks for the desired hostname for your BIGIP. Complete the survey by clicking on +ADD and subsequently the SAVE button.



Launch the Job Template and key in your desired hostname for your BIGIP.



Complete the job run and check that it succeeds.

Access your Slack channel to determine if the job runs successful or not.

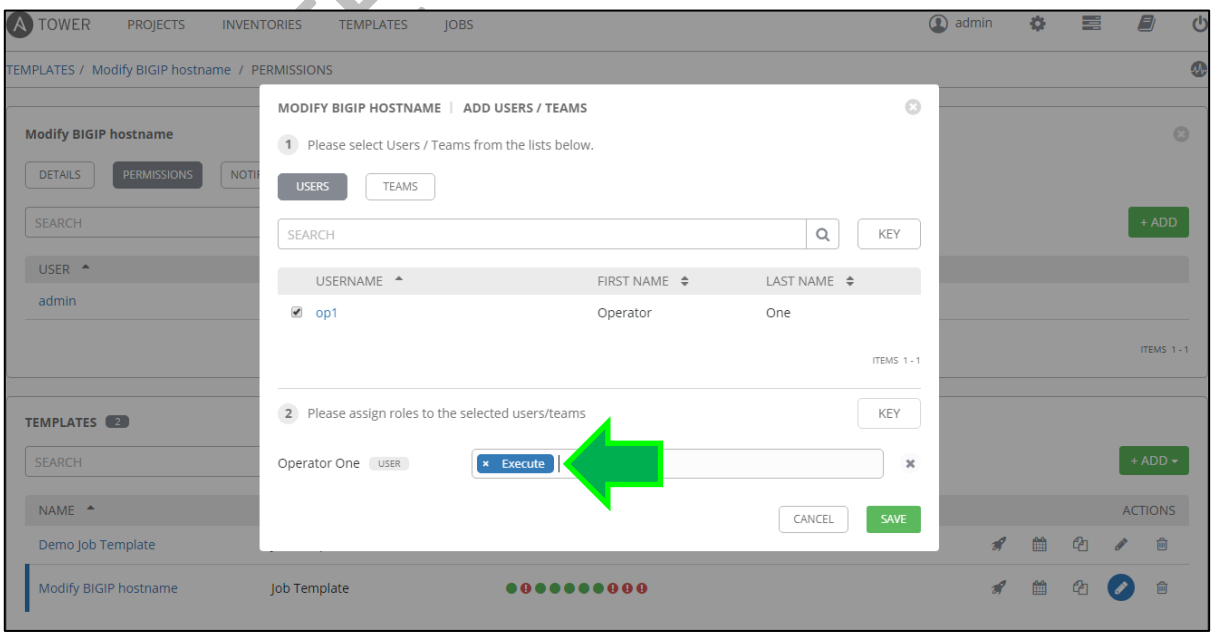
From your Ansible Tower, SSH to your assigned BIGIP to determine if the hostname has been changed or not.

3.19 Role-based Access Control

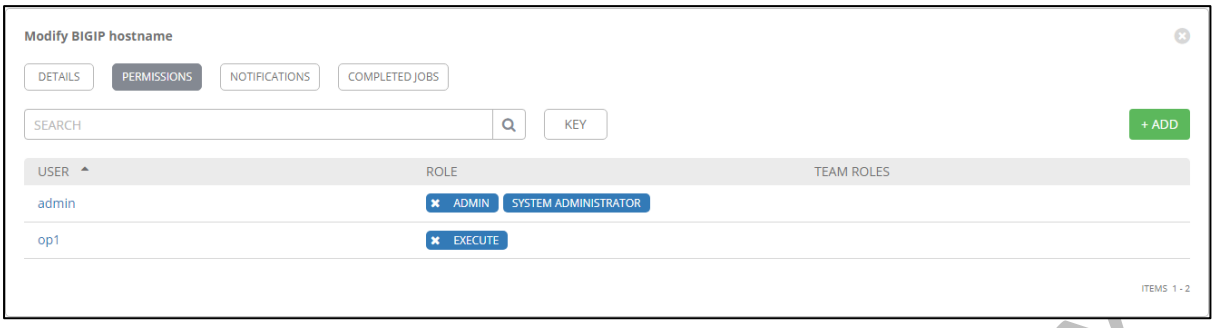
Ansible Tower has been designed primarily with RBAC as one of its main features. Role-Based Access Controls (RBAC) are built into Tower and allow Tower administrators to delegate access to server inventories, organizations, and more. Administrators can also centralize the management of various credentials, allowing end users to leverage a needed secret without ever exposing that secret to the end user. RBAC controls allow Tower to help you increase security and streamline management.

Now log out as admin. Log back in as user that you have created previously:

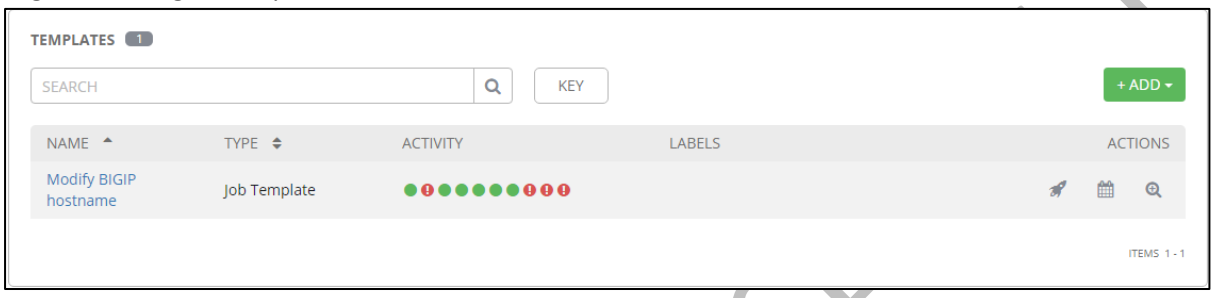
Does this user have access to the Job template previously created? Log out as Op1, log back in as admin. Enable permissions for the Job template for the User or Team that you have created previously:



If you have added execute permissions for the User, you will see the summary of the permissions as follows:



Log out and log in as Op1 user.



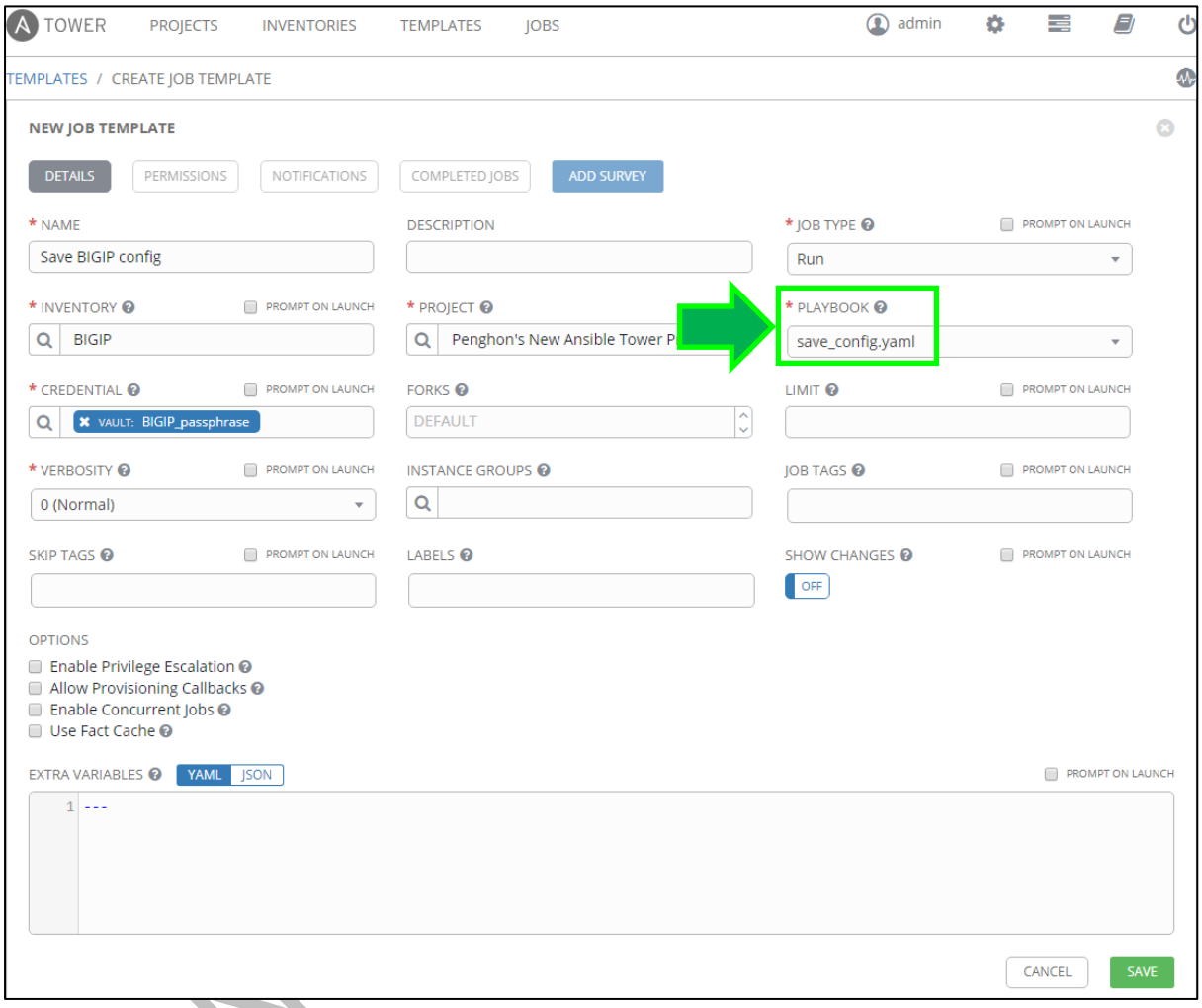
Launch the Job. Does it work?

3.20 Configuring an Ansible Workflow Template

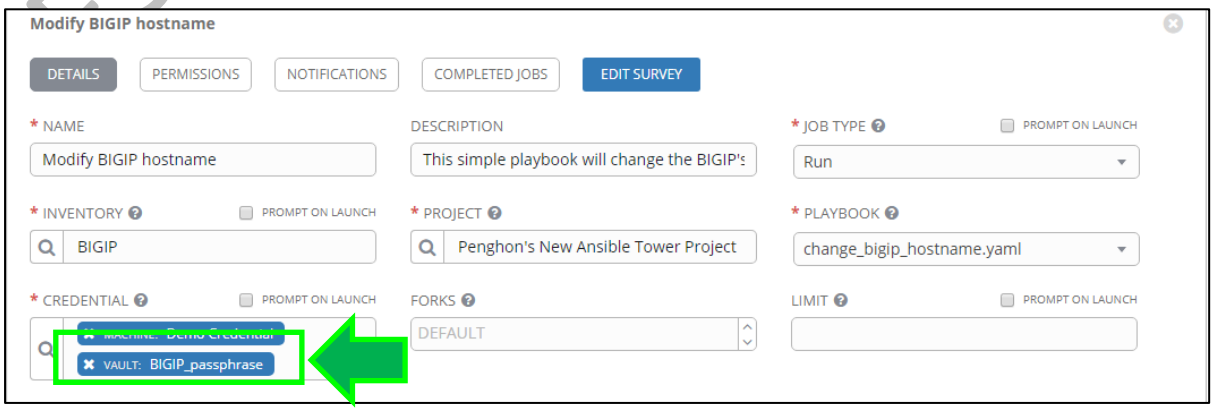
A workflow job template links together a sequence of disparate job templates and accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit.

We will be chaining the execution of 2 playbooks in a single Job Workflow.

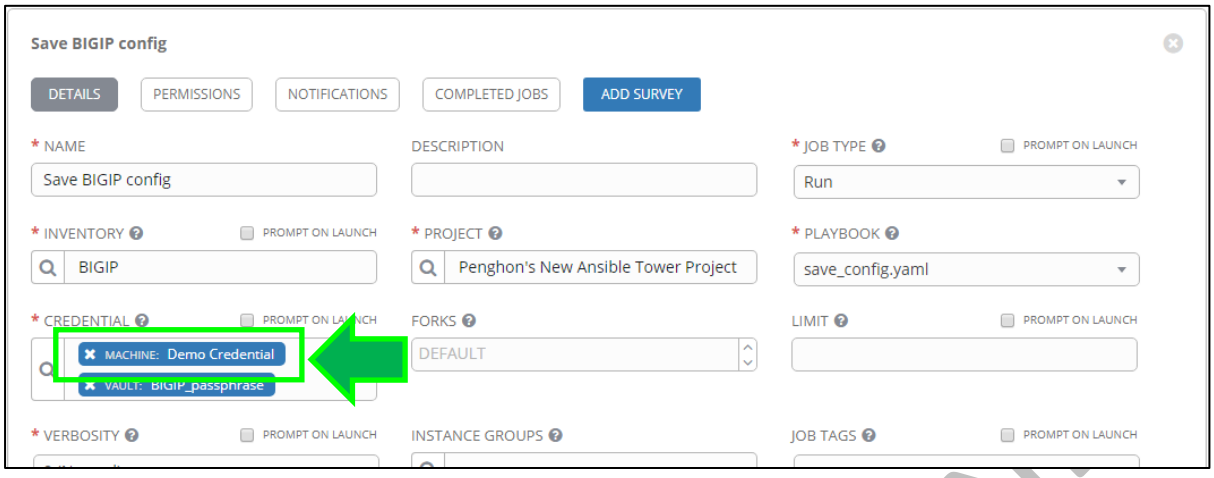
Create a new Job Template by referencing the ***save_config.yaml*** playbook:



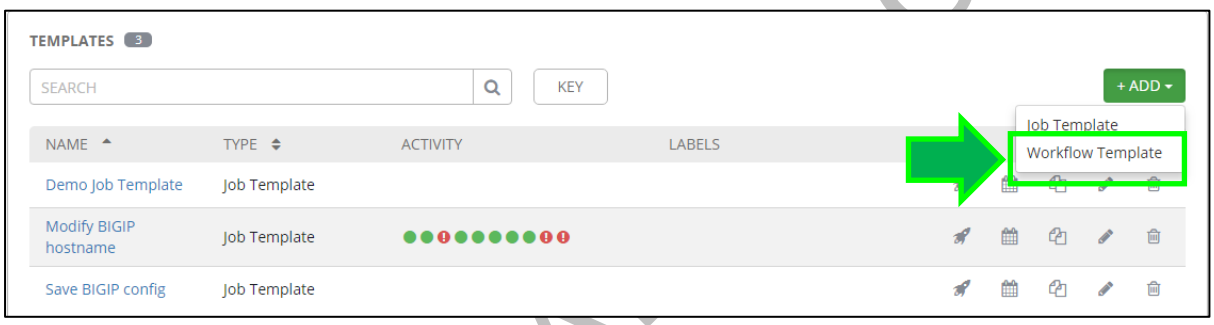
In order for the Workflow Editor to be able to see the templates for inclusion into a Workflow you need to modify the 2 Job Templates and include the Demo Credential. Select the previously created ob Template and add the Demo Credential to the Credential field:



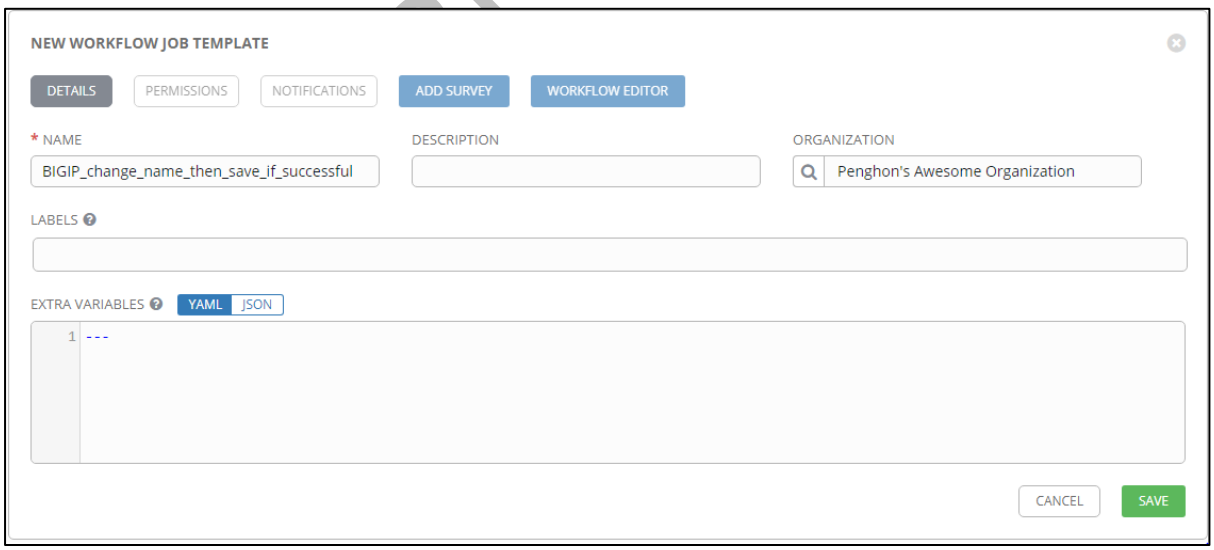
Repeat for the other Job Template:



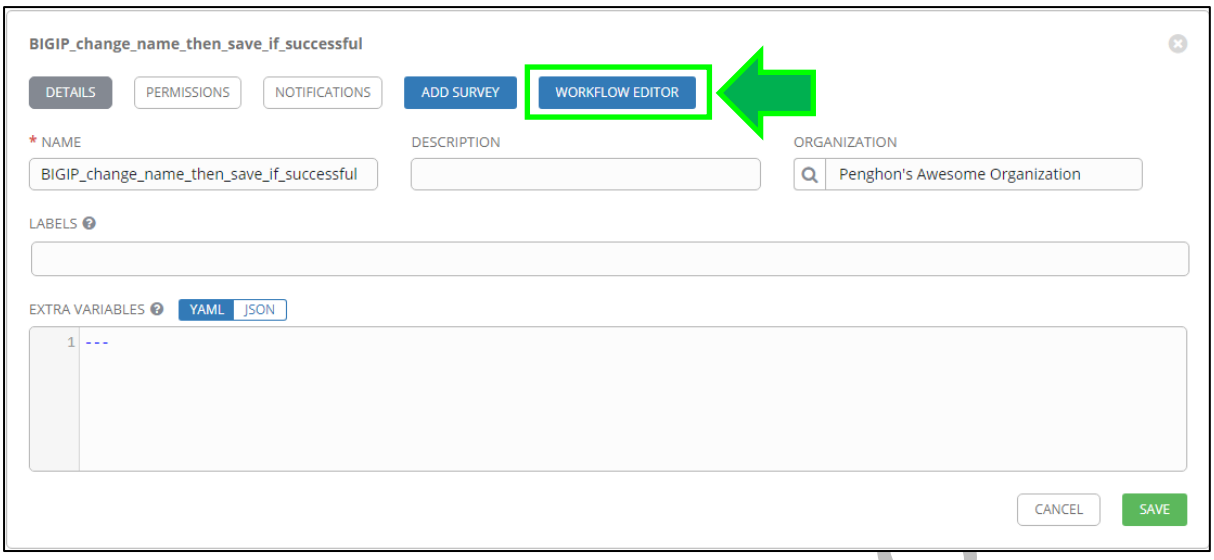
From Templates, click on +ADD and select Workflow Template:



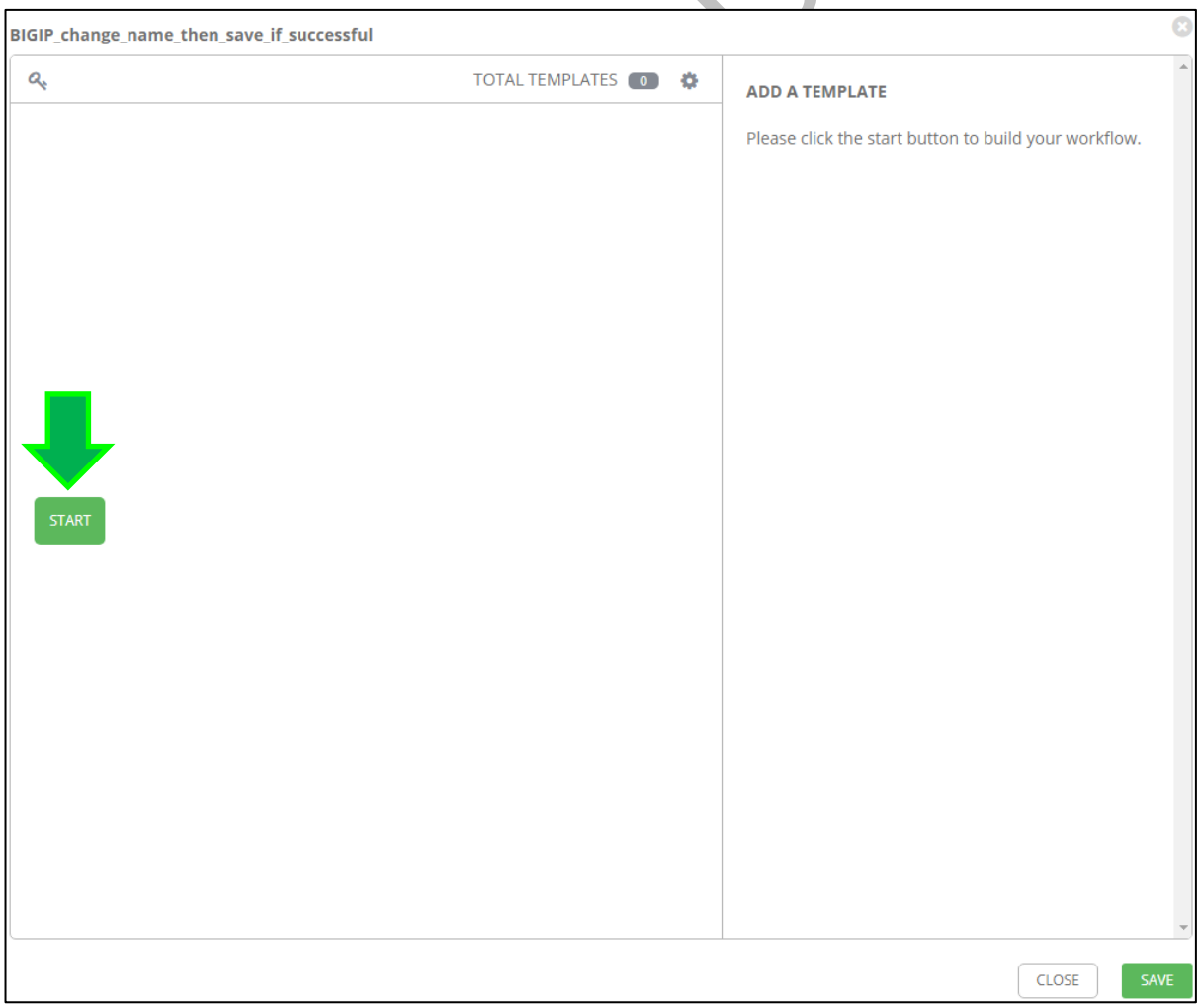
Create a new Workflow Template by filling in the necessary fields and saving the configuration:

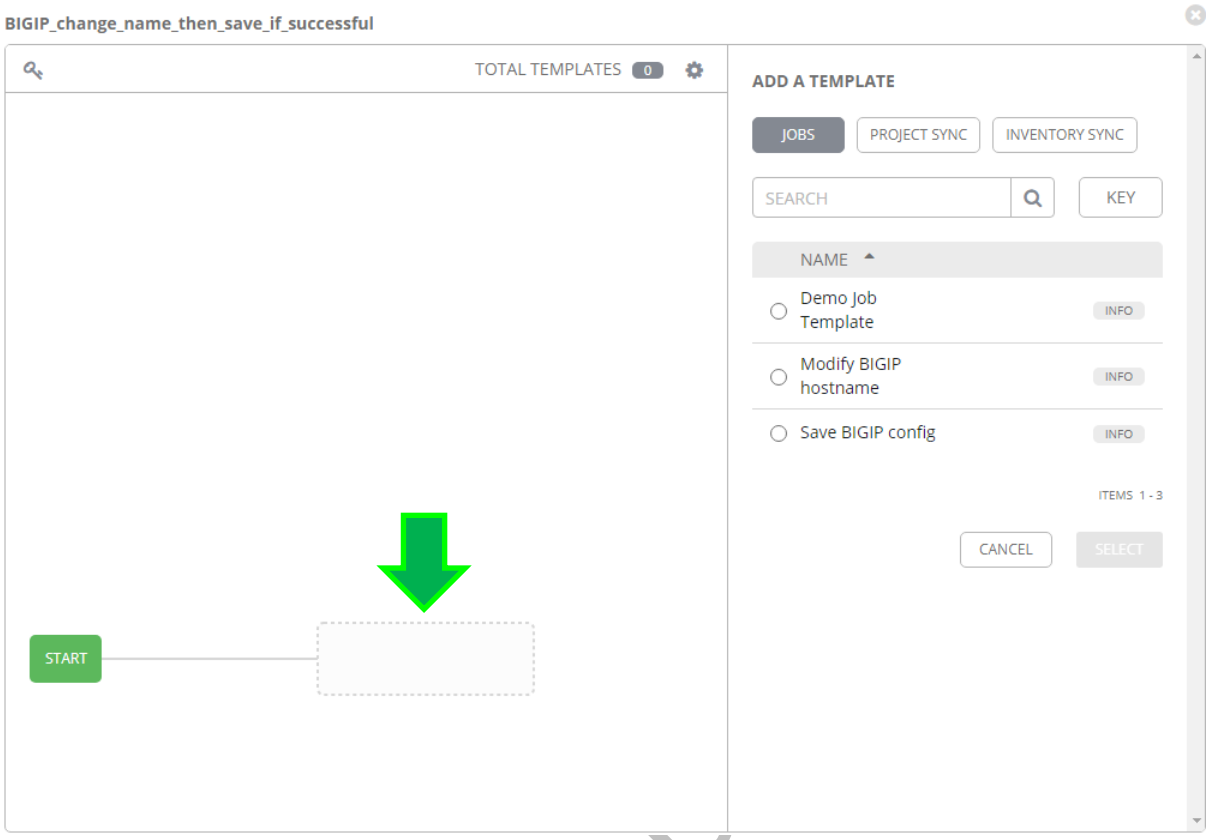


Select the WORKFLOW EDITOR:

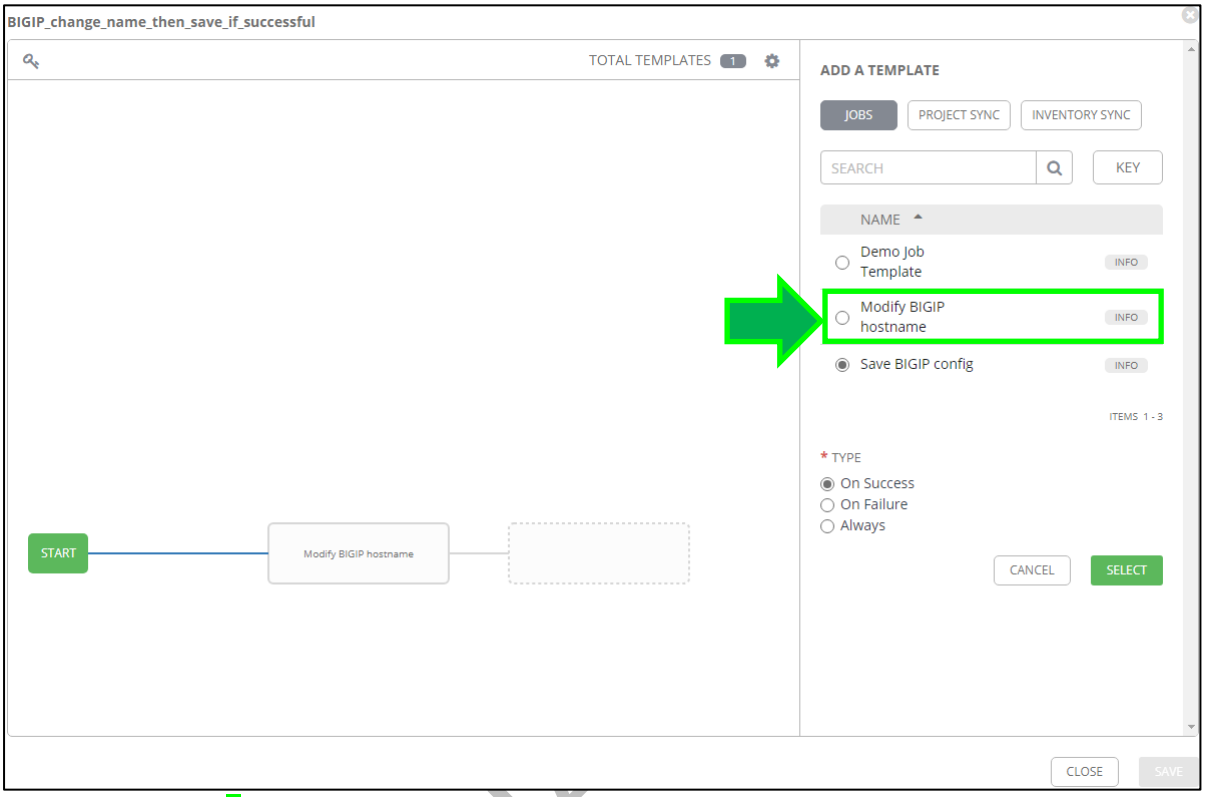


The Workflow editor will appear and you will configure the desired workflow by clicking on the green start button:

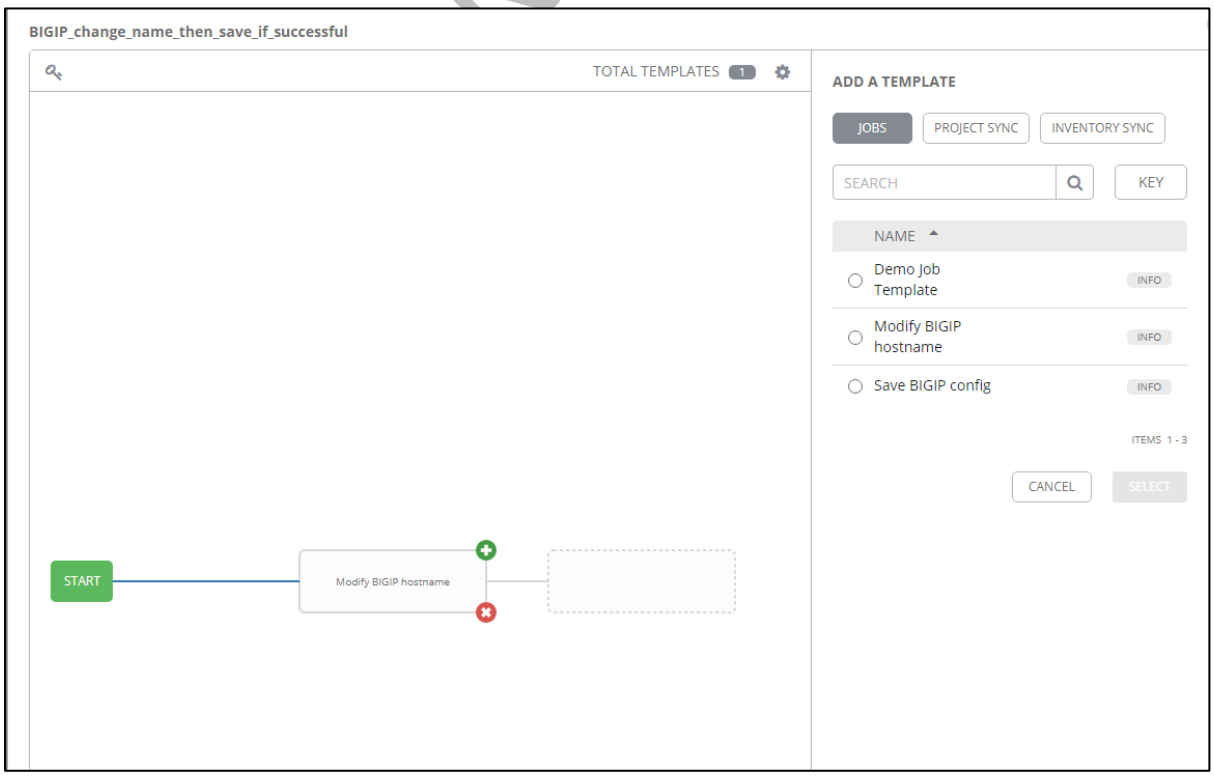



BIGIP_change_name_then_save_if_successful
TOTAL TEMPLATES 0
START
ADD A TEMPLATE
JOBS
PROJECT SYNC
INVENTORY SYNC
SEARCH
KEY
NAME
Demo Job Template
INFO
Modify BIGIP hostname
INFO
Save BIGIP config
INFO
ITEMS 1 - 3
CANCEL
SELECT

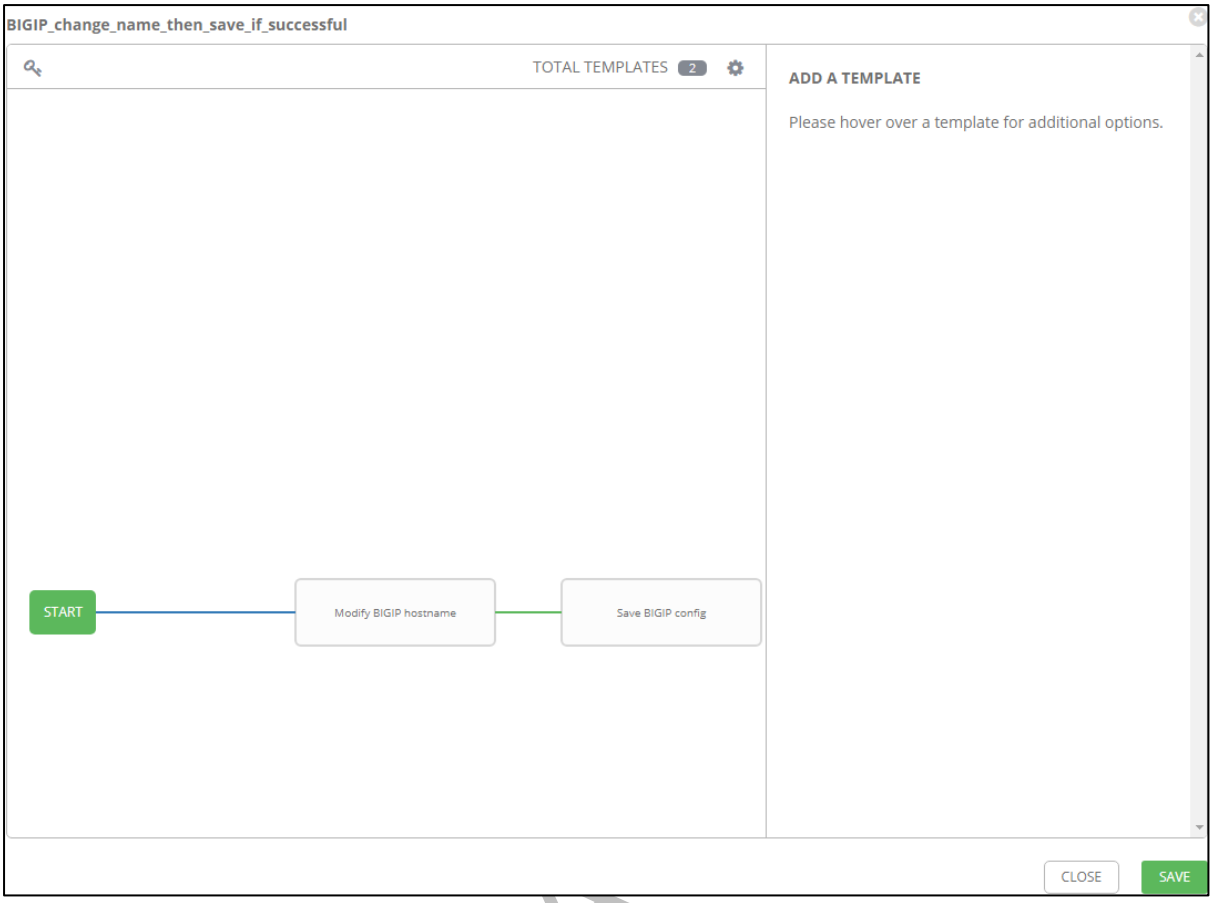
Workflows allow you to determine which Job Template to run based on the state of the previously executed Job. For the purpose of this lab task we will be chaining two Job Templates to be run one after another based on success of the initial Job Template. Select the “Modify BIGIP hostname” Job template (substitute with your Job Template if it is named differently).:



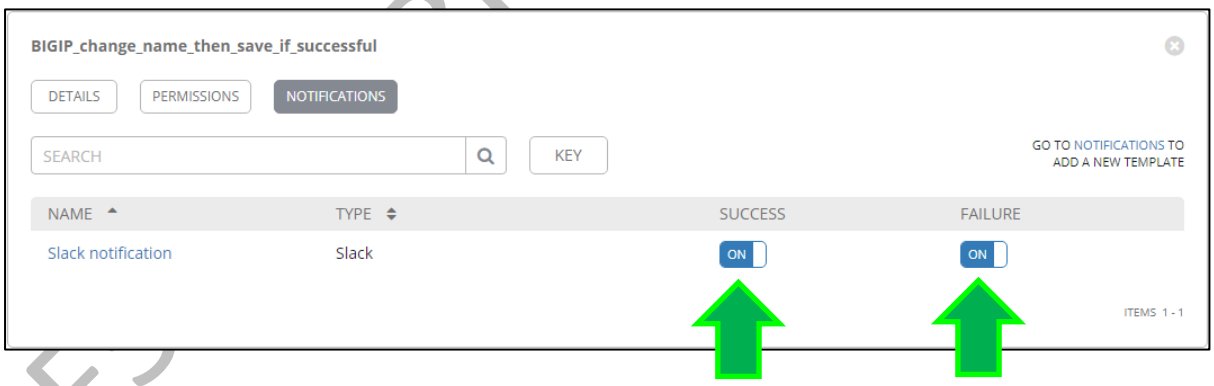
Click on the green + sign and select the second Job Template:



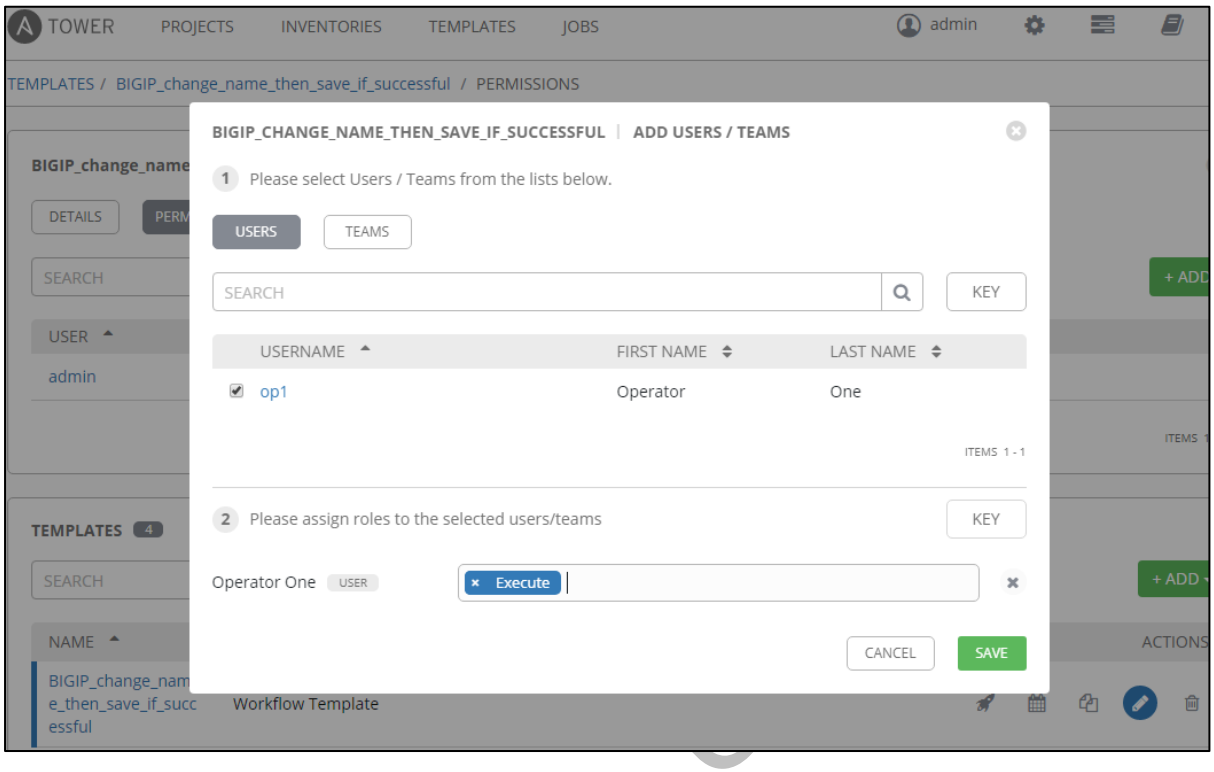
Once your Workflow Template looks like the following, click SAVE:



Select Notifications and enable the Slack notifier to notify on both success and failure of the job run:



Edit the permissions of the Workflow Template to allow the Operator User/Team to execute the Workflow Template:



Logout then log in as ***op1***. Run the workflow template. Was there any survey prompt?

3.21 Running *dev* F5 modules

Running F5 dev ansible modules is the same as how you would do for Ansible Engine/CLI. Create ansible.cfg within the specific project folder and specify the library first path of module discovery. Ensure that the dev module exists within the library.

3.22 Upgrading

Upgrading the Ansible Tower is as easy as installing it. Unpack the installation files and ensure that the ***inventory*** file is copied over from the previous installation and just execute the ***./setup.sh***. The installation process determines by default whether it is a fresh install or an upgrade.