Ansible Tower Follow Up 

Table of Contents

[**THE ENTERPRISE ENVIRONMENT**](#_fws6tgn7fngp) **8**

[**RED HAT INC. MANAGEMENT PORTFOLIO**](#_3znysh7) **9**

[**RED HAT INC. MANAGEMENT PORTFOLIO (CONTINUED)**](#_1t3h5sf) **10**

[**ANSIBLE**](#_2s8eyo1) **11**

[**ANSIBLE ENGINE**](#_j1rwwlttfmab) **11**

[**ANSIBLE TOWER**](#_17dp8vu) **11**

[Delegate](#_3rdcrjn) 11

[Access Reporting](#_26in1rg) 11

[Schedule Workflow](#_lnxbz9) 12

[Automate Through an API](#_35nkun2) 12

[**BEFORE INSTALLING ANSIBLE TOWER – ARCHITECTURAL OPTIONS**](#_44sinio) **13**

[Standalone](#_z337ya) 13

[Ansible Tower Isolated Nodes](#_3j2qqm3) 13

[**TOWER BASE REQUIREMENTS**](#_4i7ojhp) **14**

[Single Ansible/Ansible Tower Host](#_2xcytpi) 14

[Linux Clients](#_1ci93xb) 14

[Windows Clients](#_3whwml4) 14

[Ansible Official Documentation:](#_2bn6wsx) 14

[Firewall](#_qsh70q) 15

[**INSTALLING TOWER STANDALONE**](#_1pxezwc) **15**

[**SETTING UP A JUMP HOST TO USE WITH TOWER**](#_49x2ik5) **16**

[**INSTALL ANSIBLE TOWER CLUSTER. (INCLUDING POSTGRESQL REPLICATION) (NOT SUPPORTED BUT SOME CUSTOMERS STILL DO IT )**](#_147n2zr) **17**

[Install Tower Cluster](#_3o7alnk) 17

[Prepare Slave Postgresql Database For Replication](#_23ckvvd) 18

[Verify Replication](#_ihv636) 19

[**BACKUP ANSIBLE TOWER**](#_32hioqz) **20**

[**ANSIBLE TOWER DASHBOARD AND USER INTERFACE WALK THROUGH**](#_1hmsyys) **22**

[Projects](#_41mghml) 22

[Inventories](#_2grqrue) 23

[Templates](#_vx1227) 23

[Jobs](#_3fwokq0) 23

[Users](#_1v1yuxt) 24

[Setup](#_4f1mdlm) 24

[ORGANIZATIONS, TEAMS & USERS](#_19c6y18) **25**

[User](#_3tbugp1) 25

[Role Based Access Control (RBAC)](#_28h4qwu) 25

[Teams](#_nmf14n) 25

[CREDENTIALS](#_1mrcu09) **26**

[INVENTORIES](#_46r0co2) **27**

[Smart Inventory](#_juustvynjhta) 27

[Dynamic Inventory](#_p4j9vmnm0h5i) 28

[Building Dynamic and Smart Inventories with Fact Caching](#_4k668n3) 29

[Gathering Fact Playbook - Edward Quail](#_2zbgiuw) 29

[Example: FACT GATHERING PLAYBOOK](#_fi1scej1yi1v) 30

[AD-HOC COMMANDS IN TOWER](#_3ygebqi) **32**

[PROJECTS](#_2dlolyb) **33**

[TEMPLATES](#_sqyw64) **33**

[**WORKFLOW TEMPLATE**](#_owwlr6xjus1m) **34**

[Standard Job Template](#_1rvwp1q) 34

[Workflow Editor](#_4bvk7pj) 34

[User Surveys (Self-Service)](#_1664s55) 35

[JOB SCHEDULING](#_3q5sasy) **36**

[NOTIFICATIONS](#_25b2l0r) 36

[JOBS](#_kgcv8k) 36

[ADVANCED CONFIGURATION OPTIONS](#_34g0dwd) **37**

[Configure](#_1jlao46) 37

[Authentication](#_ioj28znq28bp) 37

[Jobs](#_43ky6rz) 37

[Logging aggregation](#_2iq8gzs) 38

[**IF YOU WOULD RATHER DO THINGS IN CLI**](#_xvir7l) **39**

[**Tower-cli examples**](#_3hv69ve) **39**

[Installation & Preparation](#_1x0gk37) 39

[Config Examples](#_4h042r0) 39

[Credentials Creation](#_2w5ecyt) 39

[Inventories](#_1baon6m) 40

[Add Projects](#_3vac5uf) 40

[Create A Job Templates](#_2afmg28) 40

[Custom Credential Types](#_pkwqa1) 41

[Add Users To Teams](#_39kk8xu) 41

[**MODULES**](#_48pi1tg) **42**

[Writing Modules](#_2nusc19) 42

[**ANSIBLE EXTRAS**](#_319y80a) **43**

[The Workshop](#_1gf8i83) 43

[Play Book Library](#_40ew0vw) 43

[Links](#_2fk6b3p) 43

[Decks Used Demo](#_upglbi) 43

[Request A License](#_3ep43zb) 43

[Additional Links](#_1tuee74) 43

[Free Red Hat Training](#_4du1wux) 43

[Backing Up Ansible Tower](#_2szc72q) 43

[Non Red Hat Tutorials](#_184mhaj) 44

[Open Source Apps](#_3s49zyc) 44

[ARA](#_279ka65) 44

[More About ARA](#_meukdy) 44

[Tower Workflow Approvals](#_36ei31r) 44

[AIX/Solaris](#_1ljsd9k) 44

[Microsoft](#_45jfvxd) 44

[VMware](#_zu0gcz) 45

[Terraform](#_2y3w247) 45

[Terraform Vs Ansible](#_1d96cc0) 45

[**Servicenow, Satellite, And Ansible Tower Integration**](#_3x8tuzt) **46**

[**SECURITY**](#_3bj1y38) **47**

[Ansible](#_1qoc8b1) 47

[Ansible Galaxy](#_4anzqyu) 47

[RHEL 7](#_14ykbeg) 47

[RHEL 6](#_338fx5o) 47

[General CIS](#_2hio093) 47

[STIG](#_1vsw3ci) 47

[Ubuntu](#_2uxtw84) 47

[Windows](#_3u2rp3q) 47

[Center For Internet Security](#_1nia2ey) 47

[**VENDOR SPECIFIC RESOURCES**](#_2mn7vak) **48**

[Pure](#_11si5id) 48

[Sysprep](#_3ls5o66) 48

[Workstation - Win And Mac](#_20xfydz) 48

[VMware](#_4kx3h1s) 48

[GIT](#_302dr9l) 48

[F5](#_1f7o1he) 49

[Cisco Network](#_3z7bk57) 49

[Cisco UCS](#_2eclud0) 50

[Brocade](#_thw4kt) 50

[Dell EMC Isilon](#_3dhjn8m) 50

[Hatachi](#_1smtxgf) 50

[IBM Qradar](#_4cmhg48) 50

[Veritas Storage Foundation](#_2rrrqc1) 51

[Data Guard](#_16x20ju) 51

[Oracle](#_3qwpj7n) 51

[Active Directory](#_261ztfg) 51

[IBM Tivoli](#_l7a3n9) 51

[Cyberark](#_356xmb2) 51

[Palo Alto](#_ira5dl17yrfr) 52

[Aruba](#_u5dgdrub44fb) 52

[Rubric](#_1kc7wiv) 52

[Oracle and Ansible - From Oracle](#_f33yktiikt73) 52

[Ansible Galaxy Oracle](#_p6w7ryx5boui) 52

[Community Oracle](#_95wxphtz6cr2) 53

[**USING LINKLIGHT AND SKYLIGHT WITH ANSIBLE TOWER**](#_44bvf6o) **54**

[Required](#_2jh5peh) 54

[Repos](#_dpq5tlulu3p) 54

[Install prerequisites](#_ymfzma) 54

[Sendgrid](#_3im3ia3) 54

[**CREATE A PROJECT FOR LINKLIGHT**](#_1xrdshw) **55**

[Open Ansible Tower In Your Browser And Goto “Projects”](#_4hr1b5p) 55

[Create A Free Sendgrid Account (Optional)](#_2wwbldi) 55

[Sign Into Your AWS Account](#_1c1lvlb) 55

[Boto](#_3w19e94) 55

[**LOGIN**](#_k9cznh762xm2) **56**

[License](#_mqe74d3jfxx9) 57

[Welcome To Ansible Tower](#_yrhb5mx96faw) 58

[Navigation Menu](#_zgs9jfly44hu) 58

[**SETTINGS**](#_eh4vdxct5qrm) **59**

[Authentication:](#_p495f3cufvxm) 59

[Jobs:](#_ld65smupakb7) 59

[System:](#_1wzcw46t2l10) 60

[User Interface:](#_j3j4y7susalb) 60

[License](#_xm0ivdytic3m) 60

[**ORGANIZATIONS**](#_xo1516jnbut1) **61**

[Users](#_xtfac68qqeml) 61

[**CREDENTIALS**](#_kevi0yaqsxe9) **63**

[**Select SAVE DYNAMIC INVENTORIES**](#_ci7myxb35kll) **65**

[AWS Inventory](#_lndqrl6lk7a2) 65

[**PROJECTS**](#_wfc1wcpbsohi) **67**

[**CREDENTIAL TYPES**](#_s1wgcaxu1lz9) **69**

[Create A Credential For Skylight – Windows](#_ak6ki79htk96) 70

[**TEMPLATES**](#_ivtuzwxxgutw) **72**

[PROVISION](#_pdwfxnhlev8i) 73

[TEARDOWN](#_fsmfxqs2a4l9) 75

[**SERVICENOW**](#_4tm9omut8hu3) **77**

[ServiceNow Dynamic Inventory](#_tjgwpq2slhlz) 77

[Create The Inventory](#_97je8np2cogt) 80

[**CREATING AN APPLICATION IN ANSIBLE TOWER**](#_i40d8l2jhbun) **81**

[**EXTRA – CREATING A CATALOG ITEM**](#_65cb4i6wbubr) **90**

[**DOCUMENTATION**](#_325gsm2pe98y) **94**

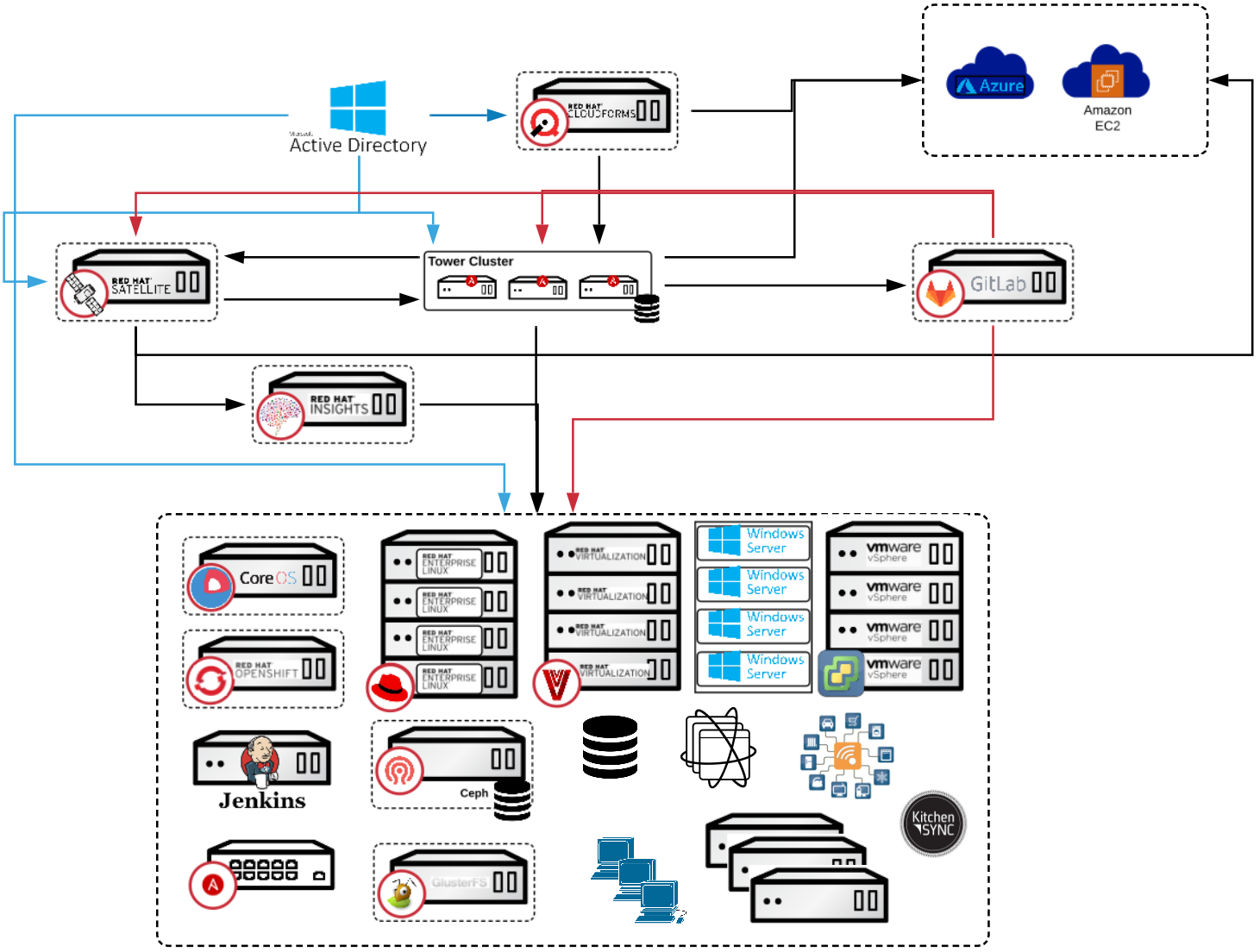
[**RED HAT SUPPORT**](#_tugyyoqs6c9b) **94**

# 

# THE ENTERPRISE ENVIRONMENT

In every environment there is a core set of servers/services that are required to allow your users to interface with the systems they use daily in seamless operation. The Red Hat Management Portfolio can provide you with the tools you need from a User Front End, Provisioning, Orchestration, Automation, and Management for your environment. These will allow you to provide that next level of IT service to your end users/customers.

Today we may be discussing one or more of these components within your environment (CloudForms, Satellite, and Ansible Tower) and how they can benefit your IT and user teams.



# RED HAT INC. MANAGEMENT PORTFOLIO

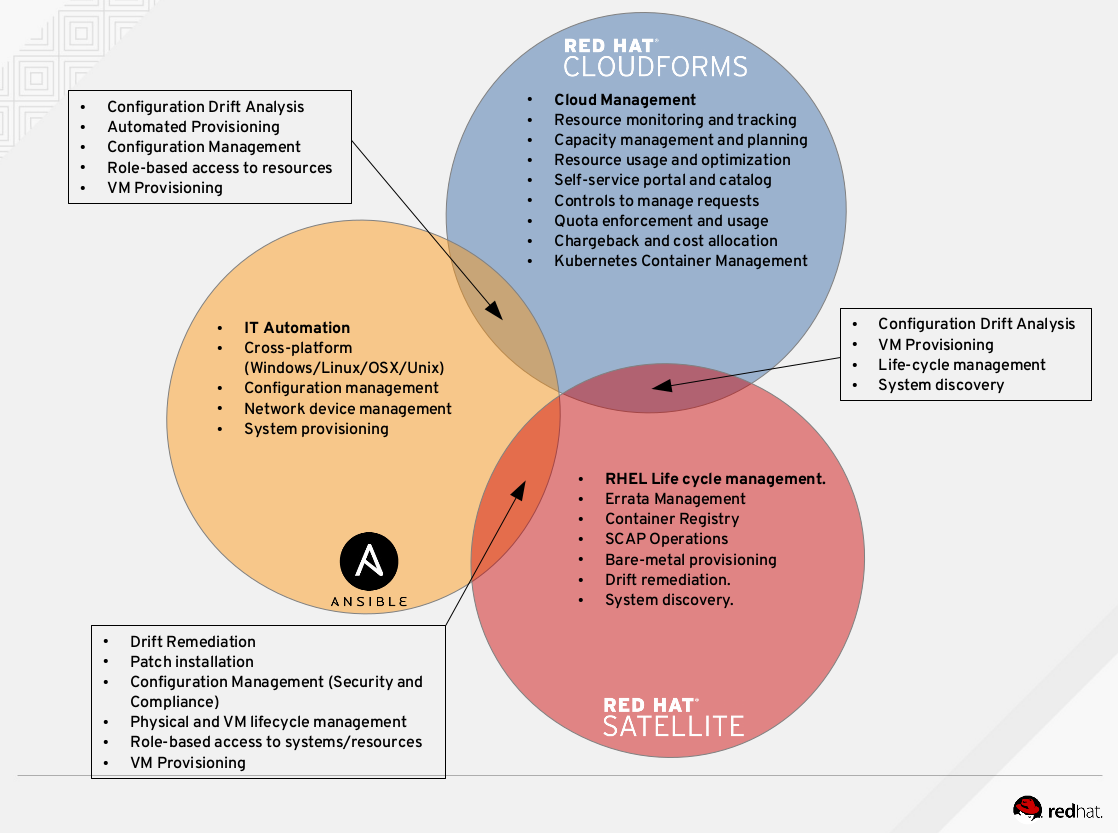
The key to success is always in the planning! The venn diagram below depicts the primary functions between tools and the overlap between to help you decide where you would like to situate the functions, and assist in integrating the tools within your IT environment(s)

**CloudForms:** A comprehensive IaaS cloud management platform that provides self-service for your virtual and cloud infrastructures while maintaining security and compliance. Lets you focus on enabling services versus managing systems so you can deliver services across your cloud environments with ease.

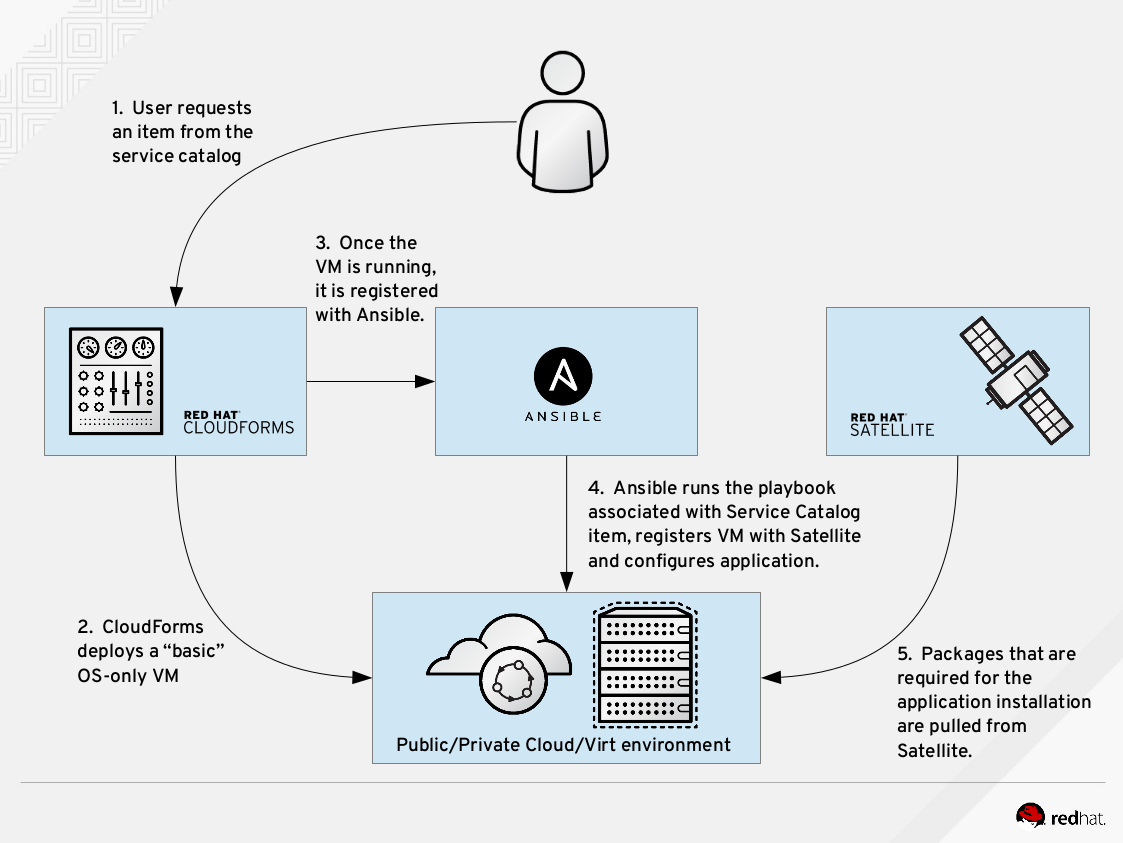
**Satellite:** The easiest way to manage your Red Hat infrastructure for efficient and compliant IT operations. Establish trusted content repos and processes that help you build a standards-based, secure Red Hat environment.

**AnsibleTower:** Simple, agentless automation platform that can improve your current processes, migrate apps for better optimization, and provide a single language for DevOps practices across your organization. Ansible Tower by Red Hat is a centralized API for your Ansible automation and a graphical user interface for Ansible.

**InSights:** A predictive analytics tool with real-time, in-depth analysis of your Red Hat infrastructure, letting you predict and prevent problems before they occur.



# RED HAT INC. MANAGEMENT PORTFOLIO (CONTINUED)

Working in tandem, this is what the optimal Management system would look like from a Red Hat perspective.

# 

# ANSIBLE

Ansible is a configuration management software implemented in Python and licensed under the GPL version 3.0 (GNU Public License). It sequentially runs a predefined set of tasks on a set of hosts and provides abstractions for selectively running different commands on a per-use case basis.

Instead of defining a state you want the machine to be in after execution, you write tasks that bring about that desired state. This makes Ansible an ideal system not only for prototyping but also for more complex systems involving several tens of hosts serving different functions.

Commands are executed by using a “push” model. You will run Ansible on your computer, on a jump host you ssh into, or through a continuous integration system you trust. Ansible then pushes commands to the remote host.

# ANSIBLE ENGINE

Ansible Engine supplies support for select Ansible Core Modules and Playbook support. This link (below) provides more information on Ansible Engine.

[https://www.Ansible.com/Ansible-engine](https://www.ansible.com/ansible-engine)

# ANSIBLE TOWER

Ansible Tower is a web-based solution that makes Ansible even more easy to use for IT teams of all kinds. It's designed to be the hub for all of your automation tasks. Ansible Tower is an enterprise framework for controlling, securing and managing your Ansible automation – with a UI and RESTful API

* Role-based access control keeps environments secure, and teams efficient
* Non-privileged users can safely deploy entire applications with push-button deployment access
* All Ansible automatons are centrally logged, ensuring complete auditability and compliance

## **Delegate**

Central to Ansible Tower is the ability to create users, and group them into teams. You can then assign access and rules to inventory, credentials, and playbooks at an individual level or team level. This makes it possible to setup push-button access to complex automation, and control who can use it, and where they can run it.

For example, when developers need to stand up a new environment, they don’t need to add another task to your already overbooked schedule. Instead, you can empower them to log into Ansible Tower, and run the provisioning playbook on their own. You can even configure Ansible Tower to prompt for parameters, providing sane defaults and reasonable choices.

## Access Reporting

Now that you have users, you’ll need tools for tracking playbook runs and troubleshooting problems. Ansible Tower adds a custom callback plugin to Ansible playbook runs that captures event and output data in real time. A high-level summary of the stored output is presented on a dashboard, providing an overview of job executions, failures, and successes, as well as a breakdown of inventory successes and failures. You’ll immediately know what playbooks have run and any trouble spots needing attention, and you can immediately click into the details.

At the playbook level, you’ll be able to access the results and output down to an individual task on a specific host. So if that new environment the development team attempted to create didn’t provision exactly as expected, you can quickly troubleshoot, and correct the problem.

## Schedule Workflow

You’ll bring together credentials, playbooks and inventory by creating a job template. The template represents the command line execution of ‘Ansible-playbook,’ except you no longer need to touch the command line. Instead, you can run the job template on demand, and watch real-time playbook output in your web browser. Or, schedule it to run later, or even on a recurring basis, and get full access to the output whenever you need it.

In addition to running a single playbook, you can create a job template with the workflow editor that chains together multiple playbook runs. Add a step that reads the latest set of hosts from your public cloud inventory, add another that automatically pulls the latest copy of the playbooks from your source repository, and add a final step to send notifications when everything is done. And just as before, run it on demand, or schedule it in the future.

## Automate Through an API

At the heart of Ansible Tower is a powerful restful API. In fact, the user interface communicates with the backend entirely through the API, demonstrating its power and completeness. Anything you can do through the web browser, you can affect through API calls.

It’s fully browsable, and self-documenting. Point your browser to the API at *http://<Tower* *server* *name>/api/*, and you can literally click your way through every endpoint. At the top of each page is the endpoint title. Click the **?** just to the right of it, and you’ll see a detailed description for each data element, indications of which elements are required when creating a new object, and how to search, sort and filter existing data.

The [Tower CLI](https://github.com/ansible/tower-cli) can also be used to make calls directly to the API from a shell script. It alleviates the need to make API calls using lower level tools like *curl* or *wget*, and provides a further source of examples, and documentation. It‘s written in Python, so in addition to being a useful tool in its own right, the source code is a how-to guide for programming API calls.

With full access to the API, and help from the Tower CLI, it’s possible to integrate Ansible Tower with other applications like ticketing tools, source code management, continuous integration systems, and shell scripts. It gives you the ability to easily automate deployment, maintenance, and mitigation tasks that previously would have been difficult, if not impossible, with just an Ansible Playbook.

# BEFORE INSTALLING ANSIBLE TOWER – ARCHITECTURAL OPTIONS

Ansible Tower is available in several license types: Self-Support, Enterprise: Standard, and Enterprise: Premium. These licenses vary in price, support-levels, and features. Enterprise: Standard and Enterprise: Premium license users will have access to some extended Tower features that Self-Support users will not have available. For more information about licenses and features, refer to [Licensing, Updates, and Support](http://docs.ansible.com/ansible-tower/3.2.4/html/installandreference/updates_support.html" \l "ir-license-support) in the Tower Installation and Reference Guide.

## Standalone

1. Tower can be installed on a single system (KVM or Bare metal) with the minimum requirements. In this configuration the DB and Tower are a single standalone system

## Ansible Tower Isolated Nodes

1. A Tower Isolated Node is a headless Ansible Tower node that can be used for local execution capacity, either in a constrained networking environment such as a DMZ or VPC, or in a remote datacenter for local execution capacity. All you need is SSH connectivity from your Tower Cluster to the Isolated Node. The Tower Cluster will send all jobs for the relevant inventory to the Isolated Node, run them there, and then pull the job details back into Ansible Tower for viewing and reporting.

[http://docs.Ansible.com/Ansible Tower/3.2.1/html/administration/clustering.html](http://docs.ansible.com/ansible-tower/3.2.1/html/administration/clustering.html)

https://www.Ansible.com/blog/deep-dive-red-hat-Ansible Tower-3-2

[https://developers.redhat.com/blog/2017/12/20/understanding-Ansible Tower-isolated-nodes/](https://developers.redhat.com/blog/2017/12/20/understanding-ansible-tower-isolated-nodes/)

[https://www.Ansible.com/blog/Ansible Tower-feature-spotlight-instance-groups-and-isolated-nodes](https://www.ansible.com/blog/Ansible-tower-feature-spotlight-instance-groups-and-isolated-nodes)

# TOWER BASE REQUIREMENTS

## Single Ansible/Ansible Tower Host

* 1. Because the requirements are so few it is easy to set up a system on a VM with an LVM file system with just root boot and swap

Source https://docs.Ansible.com/Ansible Tower/latest/html/quickinstall/prepare.html

Linux RHEL7

Ansible 2.x

4GB Ram (MIN)

30GB Total Storage

umask 022

**NOTE:** The system mask can be set back to 075 or 077 after install however if you update the system or use pip with Ansible it will disable your Ansible Tower. For safety if you do require your Ansible Tower system to have a mask other than 022 it is recommended that you put a playbook on tower for pip packages that sets umask to 0022 install PKG and then goes back to 075. The pip package playbook should be set to the source of tower env.

## Linux Clients

SSH w/keys shared with clients

Python 2.7 ^

## Windows Clients

* 1. Step by step setup examples:
  2. <https://argonsys.com/learn-microsoft-cloud/articles/configuring-Ansible-manage-windows-servers-step-step/>

<https://opensource.com/article/19/2/ansible-windows-admin?utm_medium=Email&utm_campaign=weekly&sc_cid=701f2000000RRDuAAO>

## **Ansible Official Documentation:**

https://docs.ansible.com/ansible/latest/user\_guide/windows.html

WinRm

Powershell 3.0^

Allow WinRM incoming connections (TCP/5986)

Chocolatey – [https://chocolatey.org](https://chocolatey.org/) (For installing applications)

<https://github.com/deekayen/Ansible-role-chocolatey.git>

## Firewall

Ports and instances used by Tower are as follows:

* 80, 443 (normal Tower ports)
* 22 (ssh)
* 5432 (database instance - if the database is installed on an external instance, needs to be opened to the tower instances)

Clustering/RabbitMQ ports:

* 4369, 25672 (ports specifically used by RabbitMQ to maintain a cluster, needs to be open between each instance)
* 15672 (if the RabbitMQ Management Interface is enabled, this port needs to be opened (optional)

# 

# INSTALLING TOWER STANDALONE

The easiest thing to install at Red Hat

Just change the passwords in the snippet below to match your requirements cut paste and walk away for a single node install. ( Clustered and Isolated Nodes just require a few changes to the install inventory file. )

1. Run as “root” the following to complete the install
2. yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
3. subscription-manager repos --disable '\*'
4. subscription-manager repos --enable rhel-7-server-ansible-2.8-rpms
5. subscription-manager repos --enable rhel-7-server-optional-rpms
6. subscription-manager repos --enable rhel-7-server-rpms
7. yum install -y ansible
8. yum-config-manager --disable epel
9. yum -y install wget
10. wget https://releases.ansible.com/ansible-tower/setup/ansible-tower-setup-latest.tar.gz
11. tar -zxvf ansible-tower\*.tar.gz
12. cd ansible-tower\*
13. #NOTE: Change the Password from "r3dh4t7!" to whatever you want
14. sed -i s/admin\_password="''"/admin\_password="'r3dh4t7!'"/g inventory
15. sed -i s/redis\_password="''"/redis\_password="'r3dh4t7!'"/g inventory
16. sed -i s/pg\_password="''"/pg\_password="'r3dh4t7!'"/g inventory
17. sed -i s/rabbitmq\_password="''"/rabbitmq\_password="'r3dh4t7!'"/g inventory
18. sh setup.sh

NOTE: next are things that are “Common Requests”, Extras, and Things you may find useful.

# SETTING UP A JUMP HOST TO USE WITH TOWER

[https://docs.ansible.com/Ansible Tower/latest/html/administration/tipsandtricks.html](https://docs.ansible.com/ansible-tower/latest/html/administration/tipsandtricks.html)

Credentials supplied by Tower will not flow to the jump host via ProxyCommand. They are only used for the end-node once the tunneled connection is set up.

To make this work, configure a fixed user/keyfile in the AWX user’s SSH config in the ProxyCommand definition that sets up the connection through the jump host. For example:

Host tampa

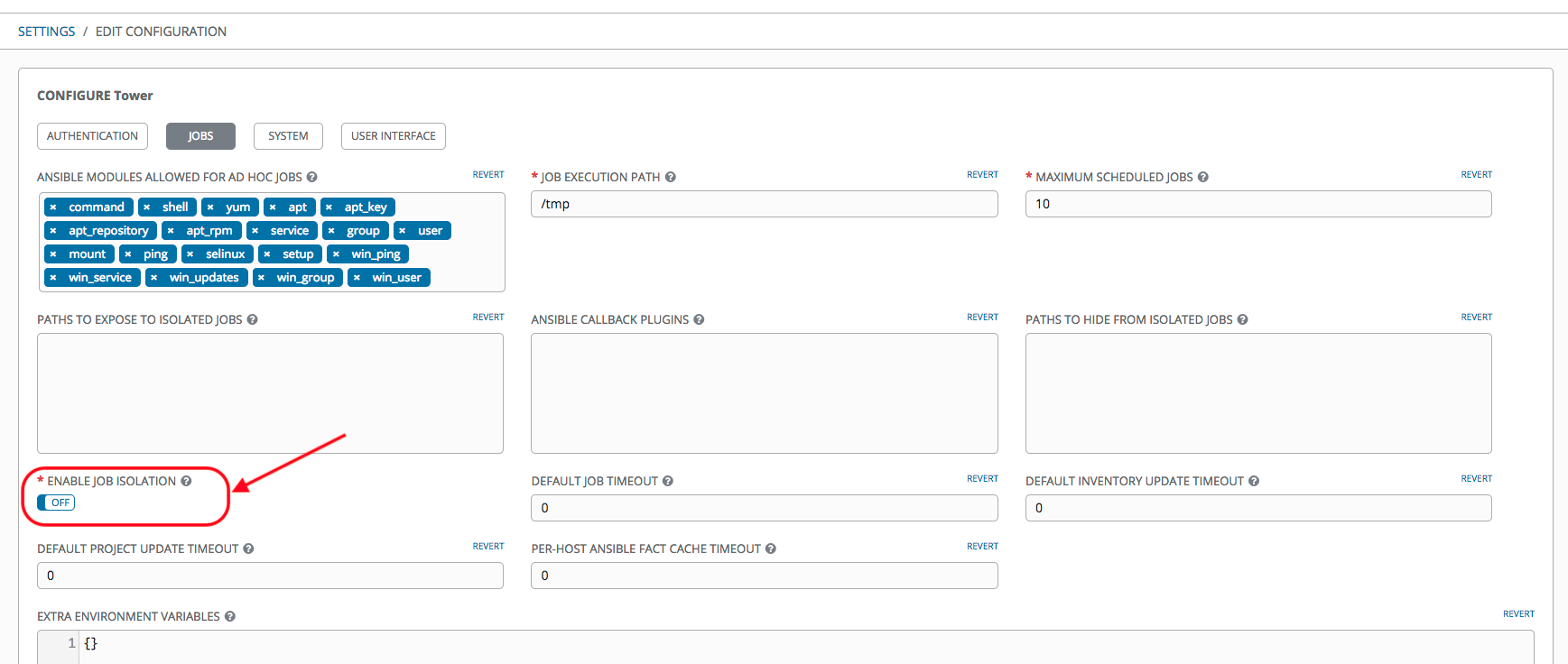
Hostname 10.100.100.11

IdentityFile [privatekeyfile]

Host 10.100..

Proxycommand ssh -W [jumphostuser]@%h:%p tampa

NOTE: You must disable PRoot by default if you need to use a jump host. You can disable PRoot through the Configure Tower user interface by setting the Enable Job Isolation toggle to OFF from the Jobs tab:



You can also add a jump host to your Tower instance through Inventory variables. These variables can be set at either the inventory, group, or host level. To add this, navigate to your inventory and in the variables field of whichever level you choose, add the following variables:

ansible\_user: <user\_name>

ansible\_connection: ssh

ansible\_ssh\_common\_args: '-o c <user\_name>@<jump\_server\_name>"'

# 

# INSTALL ANSIBLE TOWER CLUSTER. (INCLUDING POSTGRESQL REPLICATION) (NOT SUPPORTED BUT SOME CUSTOMERS STILL DO IT )

## Install Tower Cluster

Download Ansible 3.x Bundle on the first Tower Instance

Edit Ansible Tower-setup-3.x.x/inventory to specify list of hosts where Ansible Tower packages shall be deployed and add the hostname for external Postgres database will be deployed and configured. You also need to specify the password for Ansible Tower \*admin user, postgres user and rabbitmq user.

Execute setup script to install Ansible Tower and external database server

./setup.sh

Prepare master Postgresql database for replication

Replace postgreservermaster with the Mater PostgreSQL Server

1) Edit postgresql.conf file to include custom configuration directory

ansible postgreservermaster -m lineinfile -a "line='include\_dir = 'conf.d'' path=/var/lib/pgsql/9.6/data/postgresql.conf"

2) Create the custom configuration directory under /var/lib/pgsql/9.6/data/ with the name conf.d

ansible postgreservermaster -m file -a 'path=/var/lib/pgsql/9.6/data/conf.d state=directory'

3) Create new file tower-postgresql.conf file with all the custom settings needed for replication and remote copy it

cat << EOF > tower-postgresql.conf

wal\_level = hot\_standby

synchronous\_commit = local

archive\_mode = on

archive\_command = 'cp %p /var/lib/pgsql/9.6/data/archive/%f'

max\_wal\_senders = 2

wal\_keep\_segments = 10

synchronous\_standby\_names = 'slave01'

EOF

ansible postgreservermaster -m copy -a "src=/root/tower-postgresql.conf dest=/var/lib/pgsql/9.6/data/conf.d/tower-postgresql.conf"

4) Append the file /var/lib/pgsql/9.6/data/pg\_hba.conf to add line which configures master node to accept connections from slave server.

ansible postgreservermaster -m lineinfile -a "line='host replication replica 0.0.0.0/0 md5' path=/var/lib/pgsql/9.6/data/pg\_hba.conf"

5) Restart postgresql-9.6 service and check for any errors. If there is an error in restarting service please check the log files in directory /var/lib/pgsql/9.6/data/pg\_log/ on support1 host. Fix them and restart postgresql-9.6 service before proceeding.

ansible postgreservermaster -m service -a"name=postgresql-9.6 state=restarted"

6) Create user replica which will be used by slave server to replicate database from master server

ansible postgreservermaster -m postgresql\_user -a "name=replica password=r3dh4t1! role\_attr\_flags=REPLICATION state=present" --become-user=postgres

## Prepare Slave Postgresql Database For Replication

1)Install postgresql on the slave server

ansible postgresqlslave -m yum -a "name=postgresql96-server state=present"

2) run pg\_basekup to copy all the configuration files and database files from master server with replica user privileges

ansible postgresqlslave -m shell -a "export PGPASSWORD=r3dh4t1! && pg\_basebackup -h postgreservermaster -U replica -D /var/lib/pgsql/9.6/data/ -P --xlog" --become-user=postgres

3) Edit the configuration file /var/lib/pgsql/9.6/data/postgresql.conf to setup the slave as standby postgresql user database server

ansible postgresqlslave -m lineinfile -a "line='hot\_standby = on' path=/var/lib/pgsql/9.6/data/postgresql.conf"

4) Create a new file recovery.conf and remote copy it to /var/lib/pgsql/9.6/data/ directory on the slave server for setting up the slave for recovery.

cat << EOF > recovery.conf

restore\_command = 'scp support1.${GUID}.internal:/var/lib/pgsql/9.6/data/archive/%f %p'

standby\_mode = on

primary\_conninfo = 'host=support1.${GUID}.internal port=5432 user=replica password=PASSWORD application\_name=slave01'

EOF

ansible postgresqlslave -m copy -a "src=/root/recovery.conf dest=/var/lib/pgsql/9.6/data/recovery.conf"

5) Start and enable postgres 9.6 service on support2 host

ansible postgresqlslave -m service -a "name=postgresql-9.6 state=started enabled=true"

## Verify Replication

ansible postgresqlmaster -m shell -a "psql -c 'select application\_name, state, sync\_priority, sync\_state from pg\_stat\_replication;'" --become-user postgres

# BACKUP ANSIBLE TOWER

If a repository of the Ansible Tower installation package already exists skip to step 2.

1. Create a repository in bitbucket/git/gitlab of the Ansible Tower installation.

yum insrall -y <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

yum install -y git

#Skip the wget and untar steps in **RED** if you just set up your tower

wget [https://releases.Ansible.com/Ansible Tower/setup/Ansible Tower-setup-latest.tar.gz](https://releases.ansible.com/ansible-tower/setup/ansible-tower-setup-latest.tar.gz)

tar -xzvf Ansible\*.tar.gz

cd ~/Ansible Tower\*

git init

git remote add origin <bitbucket\_repo\_url>

git add .

git commit -m “Initial Commit”

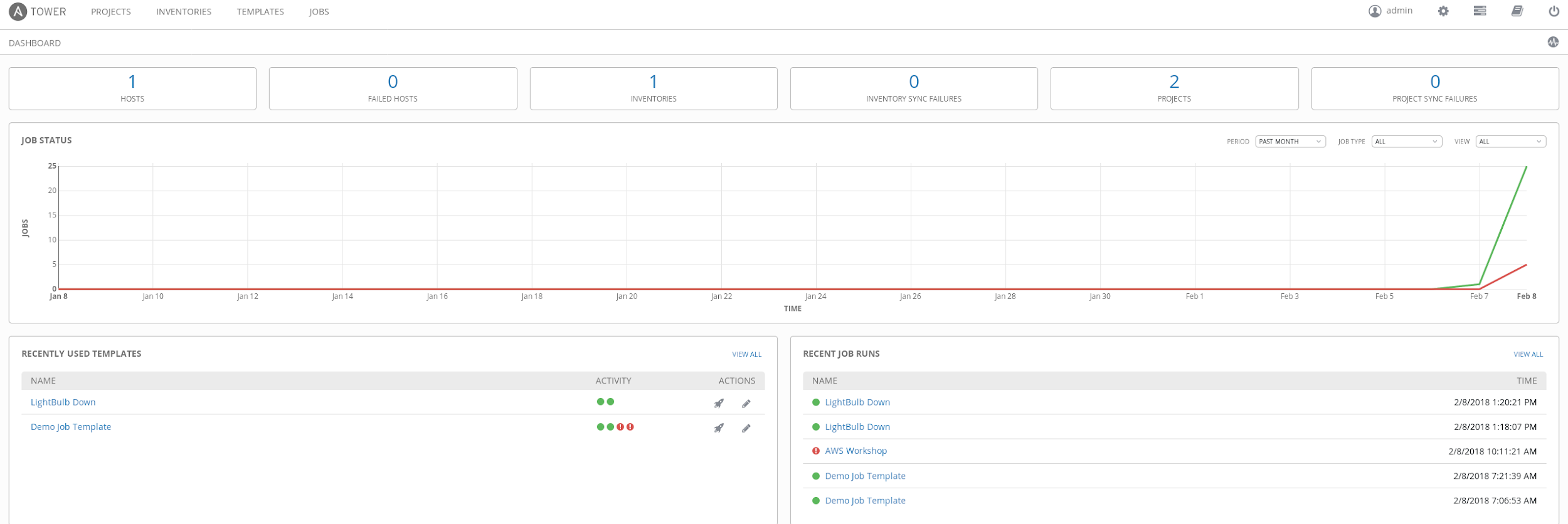
git push -u origin master

1. The backup role in the Ansible Tower installation will need to be slightly modified in order to run backups via Ansible Tower.
2. Edit and add the following to line 45 of roles/backup/tasks/download.

become: false

1. Create an Ansible Tower project if one does not already exist based on your Ansible Tower-setup repository.
2. Create an Ansible Tower inventory if one does not already exist based on your Ansible Tower-setup/inventory file used during the initial setup/upgrade of your Ansible Tower installation.
3. Create an Ansible Tower template utilizing the above objects created.
4. Utilize root credentials or select a privileged account.
5. If using a privileged account you will need to Enable Privilege Escalation on your Ansible Tower Job Template.
6. Select Ansible Tower-setup project.
7. Select backup.yml playbook.
8. In the EXTRA VARIABLES section set: backup\_dest: “/path/to/tower\_backups/”
9. Add job template schedule at the frequency needed. Likely every 30-60 minutes.

# ANSIBLE TOWER DASHBOARD AND USER INTERFACE **WALK THROUGH**

TOWER - returns you to main screen

Simple easy and clean user interface

Interface displays Tower, Projects, Inventories, Templates, Jobs, User, Gear Configure Icon, My View, Documentation, and Log out.

In the middle “JOB STATUS” graph shows simple Red and green success and failure status

At the bottom there is “RECENTLY USED TEMPLATES” showing Activity and “RECENT JOB RUNS” showing individual project runs with timestamp and success indicator in green or fail in red

Quick look/review at the first level screens of the interface

## Projects

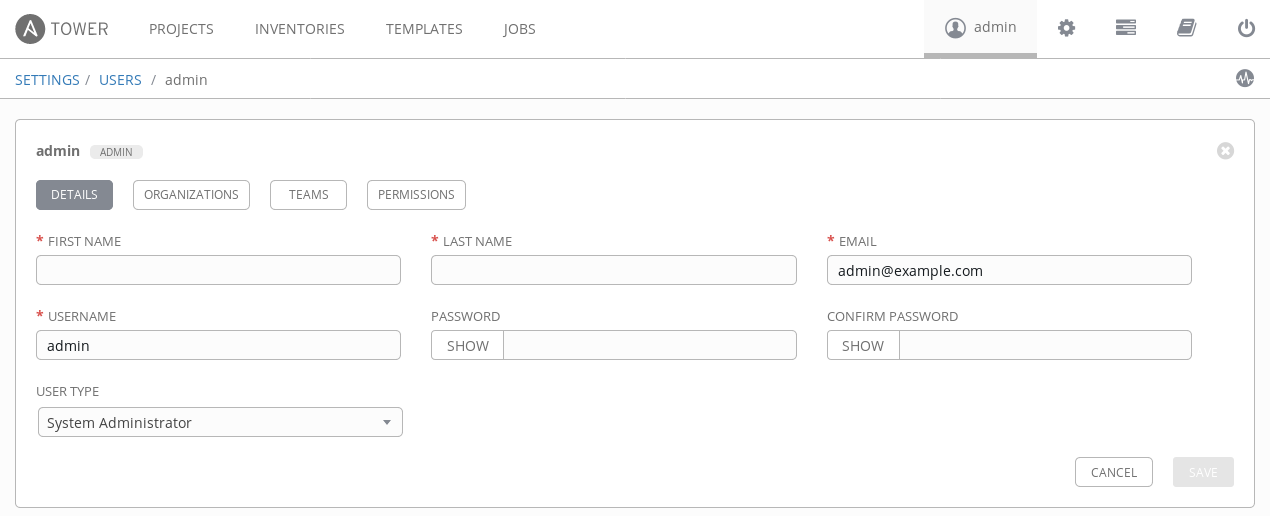
## Inventories

## Templates

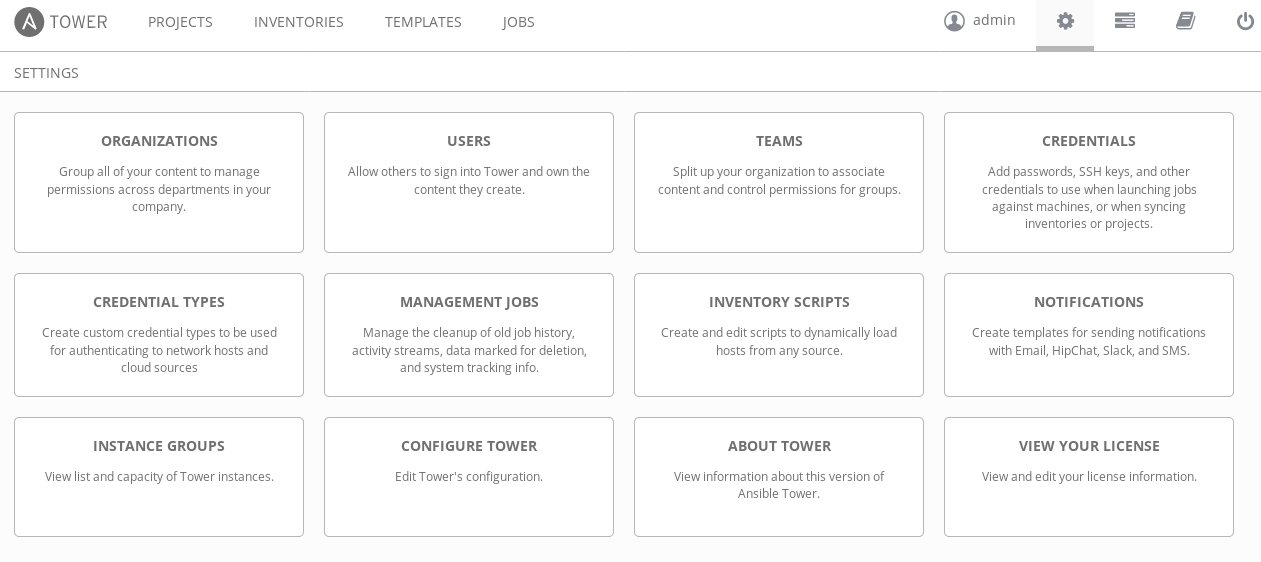
## Jobs

### 

## Users



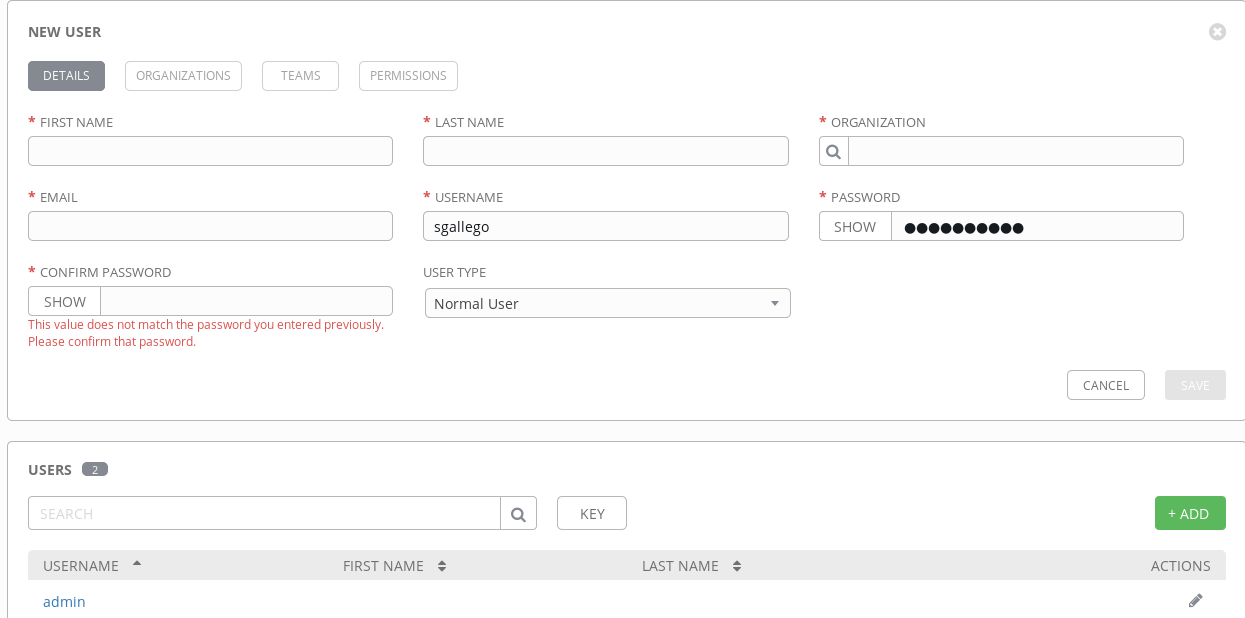
## Setup

1. Lets drill down into the items above...

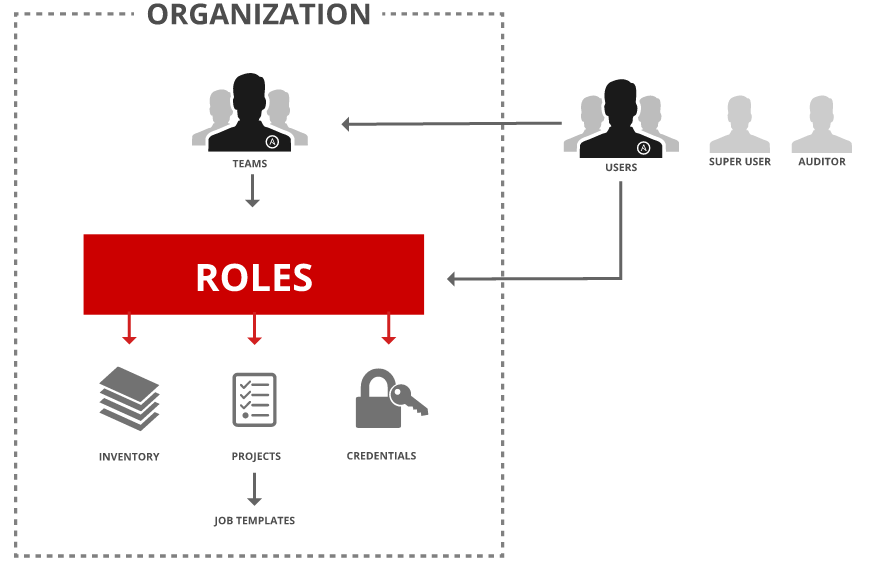
# ORGANIZATIONS, TEAMS & USERS

An organization is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization with the exception of users.

## User

A user is an account to access Tower and its services given the permissions granted to it.

## Role Based Access Control (RBAC)

Role-Based Access Controls are built into Ansible Tower and allow administrators to delegate access to server inventories, organizations, and more. These controls allow Tower to help you increase security and streamline management of your Ansible automation.

## Teams

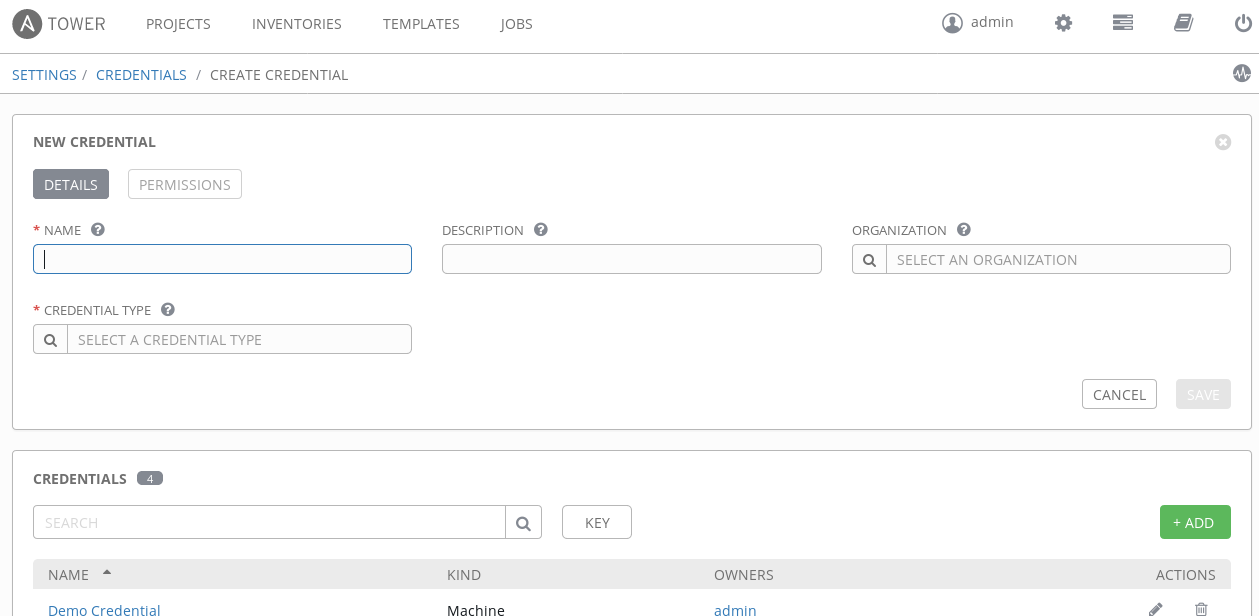
Teams provide a means to implement role-based access control schemes and delegate responsibilities across organizations.

# CREDENTIALS

Credentials are utilized by Tower for authentication with various external resources:

* Connecting to remote machines to run jobs
* Syncing with inventory sources
* Importing project content from version control systems
* Connecting to and managing networking devices

Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.



# INVENTORIES

Inventory is a collection of hosts (nodes) with associated data and groupings that Tower can connect to and manage.

* Hosts (nodes)
* Groups
* Inventory-specific data (variables)
* Static or dynamic sources

## Smart Inventory

A Smart Inventory is a collection of hosts defined by a stored search that can be viewed like a standard inventory and made to be easily used with job runs. Organization administrators have admin permission to inventories in their organization and can create Smart Inventories. A Smart Inventory is identified by KIND=smart. You can define a Smart Inventory using the same method being used with Tower Search. InventorySource is directly associated with an Inventory.

The Inventory model has the following new fields that are blank by default but are set accordingly for Smart Inventories:

* kind is set to smart for Smart Inventories
* host\_filter is set AND kind is set to smart for Smart Inventories.

The host model has a new field, smart\_inventories that uses a membership lookup table that identifies a set of all the Smart Inventory a host is associated with. The memberships are generated by a task. The task is launched when:

* + a new host is added
  + an existing host is modified (updated or deleted)
  + a new Smart Inventory is added
  + an existing Smart Inventory is modified (updated or deleted)

### 

## Dynamic Inventory

Dynamic inventory is a script that queries a service, like a cloud provider API or a management application. This data is formatted in an Ansible-specific JSON data structure and is used in lieu of static inventory files.

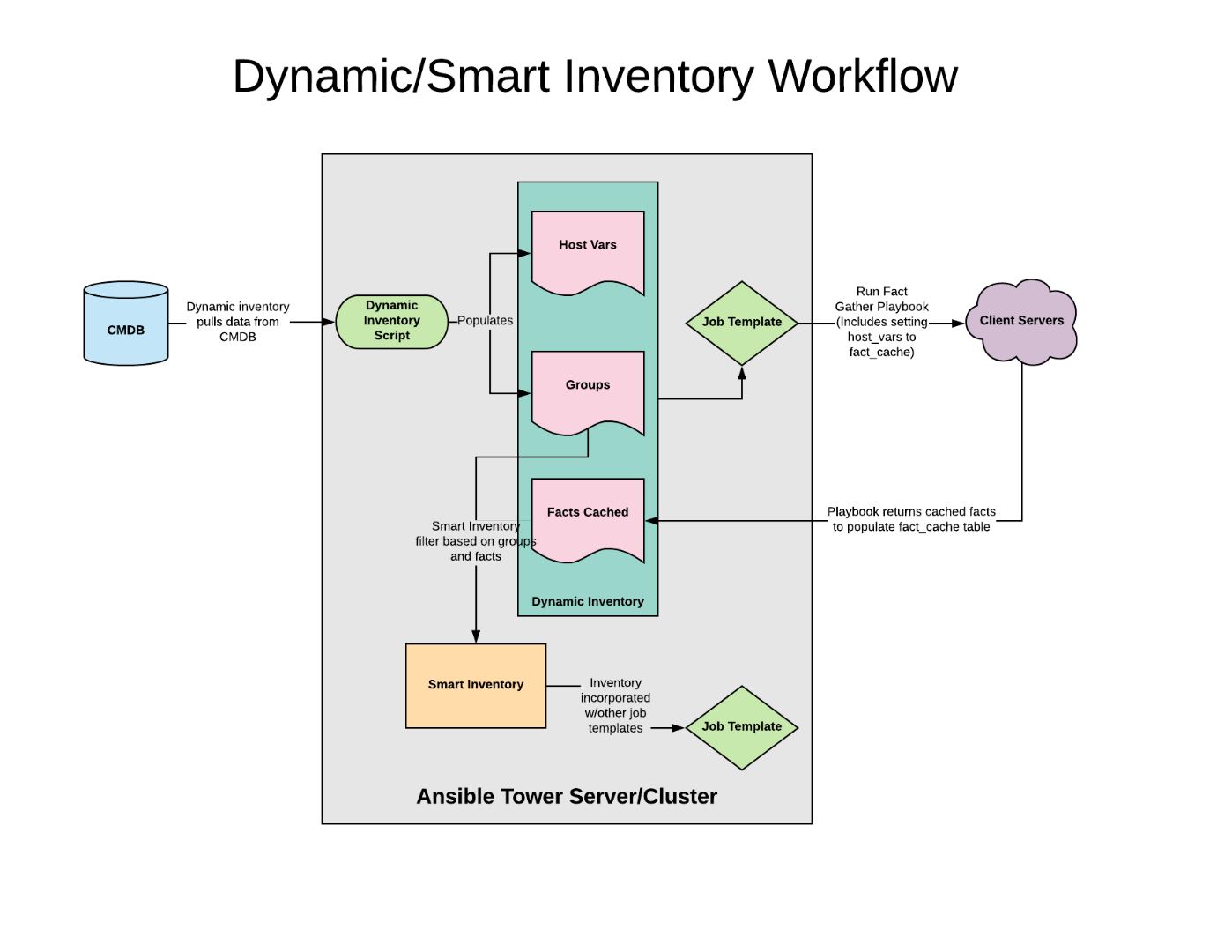
* + Groups are generated based on host metadata
  + Single source of truth saves time, avoids duplication and reduces human error
  + Dynamic and static inventory sources can be used together

1. **NOTE: If using static inventories or you are migrating from Ansible to Ansible Tower the ini based inventories can be imported to Ansible Tower.**

In Tower create an inventory called “Ansible Workshop Inventory” but do not add the nodes then on the command line:

sudo tower-manage inventory\_import --source=/home/<username>/lightbulb/lessons/lab\_inventory/<username>-instances.txt --inventory-name="Ansible Workshop Inventory"

## Building Dynamic and Smart Inventories with Fact Caching



## Gathering Fact Playbook - Edward Quail

1. There may be use cases where you need to collect specific facts based on a host. A recent customer mentioned that gather facts causes issues when collecting hardware facts on NFS shares that are stale mounted. Currently the only work around for this is to disable hardware via the setup module or gathering Ansible configuration by setting gather\_subset: !hardware.
2. A workaround for this is to add logic to your fact gathering playbook to check for stale mounts and conditionally exclude hardware with gather\_facts set to false. Additionally you may want to set the gather configuration to explicit to avoid users of Ansible tower from collecting facts by default and cause hanging jobs due to NFS stale mount issues.
3. Below is example code on how to create manual facts and conditionally check for NFS stale mounts. NOTE: additional information to come for gathering facts from facts.d.

### Example: FACT GATHERING PLAYBOOK

- hosts: all

gather\_facts: false

tasks:

# Clears all gathered facts on hosts that have been previously

# gathered by Ansible.

- meta: clear\_facts

# This is an example of how to manually

# gather facts from a host by using set\_fact

- name: Gathering the release OS version

shell: "sed -e '/NAME/!d' -e '/CPE/d' -e 's,.\*=,,g' /etc/os-release"

register: release\_version

- name: Setting a fact

set\_fact:

cacheable: True

inscopeforsox: "{{ sn\_u\_inscopeforsox }}"

application: "{{ sn\_application }}"

my\_var: "variable\_value"

a\_host\_var: "{{ a\_host\_var }}"

my\_release\_version:

OS\_TYPE: "{{ release\_version.stdout\_lines[0] }}"

PRETTY\_NAME: "{{ release\_version.stdout\_lines[1] }}"

# Example of how to gather NFS mounts points

# and check whether or not they are stale before running

# setup module to gather the system's facts

- name: Gathering all NFS mount points

shell: 'awk "\$3 == \"nfs\" && /vers=3/ {print \$2}" /proc/mounts'

register: mount\_out

- name: Checking for NFS stale mounts

shell: 'stat -t "{{ item }}" > /dev/null 2>&1'

register: stat\_check

with\_items: "{{ mount\_out.stdout\_lines }}"

ignore\_errors: True

# When a stale NFS mount is detected from stat -d

# we skip the hardware fact gathering

- name: Running setup module without hardware

setup:

gather\_subset: '!hardware'

when: stat\_check|failed

# When no errors are detected in stat -d check of the

# mount setup module is ran normally to gather all the

# system's facts.

- name: Running setup module normally

setup:

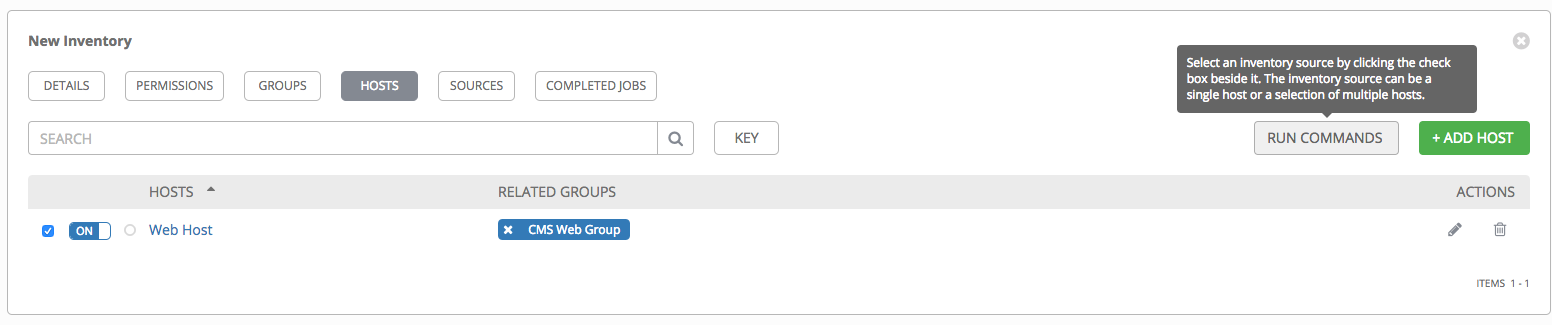
gather\_subset: all

when: stat\_check|succeeded

# AD-HOC COMMANDS IN TOWER

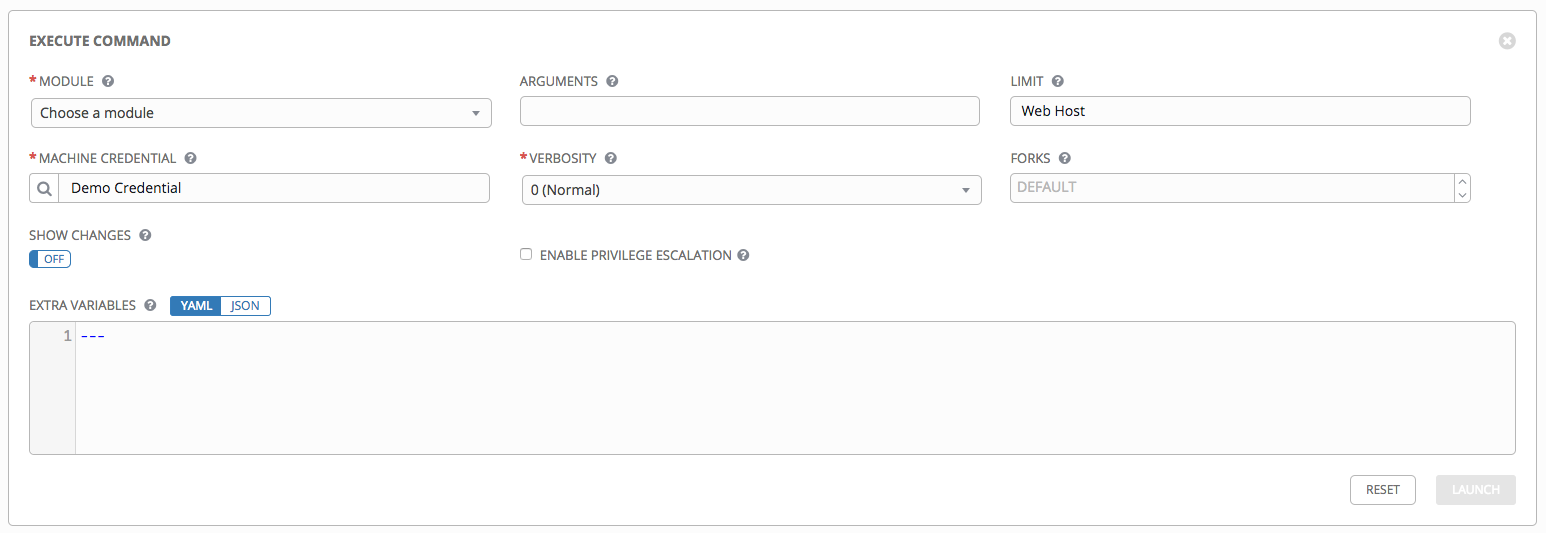
To run an ad hoc command in Tower:

Select an inventory source from the list of hosts or groups. The inventory source can be a single group or host, a selection of multiple hosts, or a selection of multiple groups.



Click the  button on right side of screen.

The Execute Command window opens.



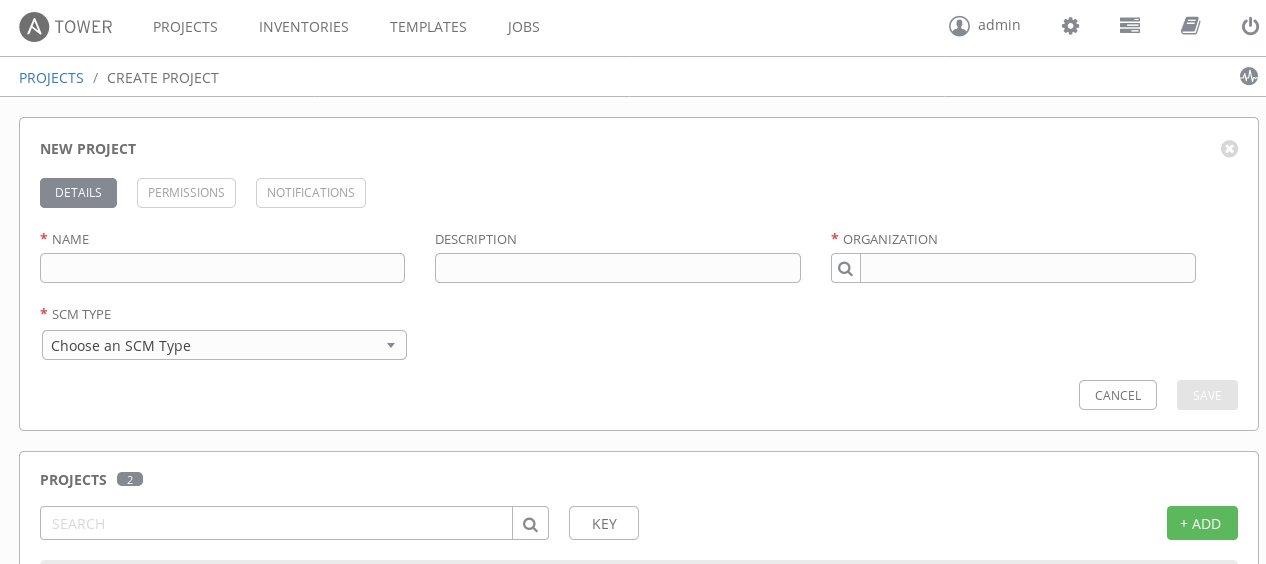
Fill in and use drop downs to provide details applicable to your environment

Choose LAUNCH in lower right hand of menu

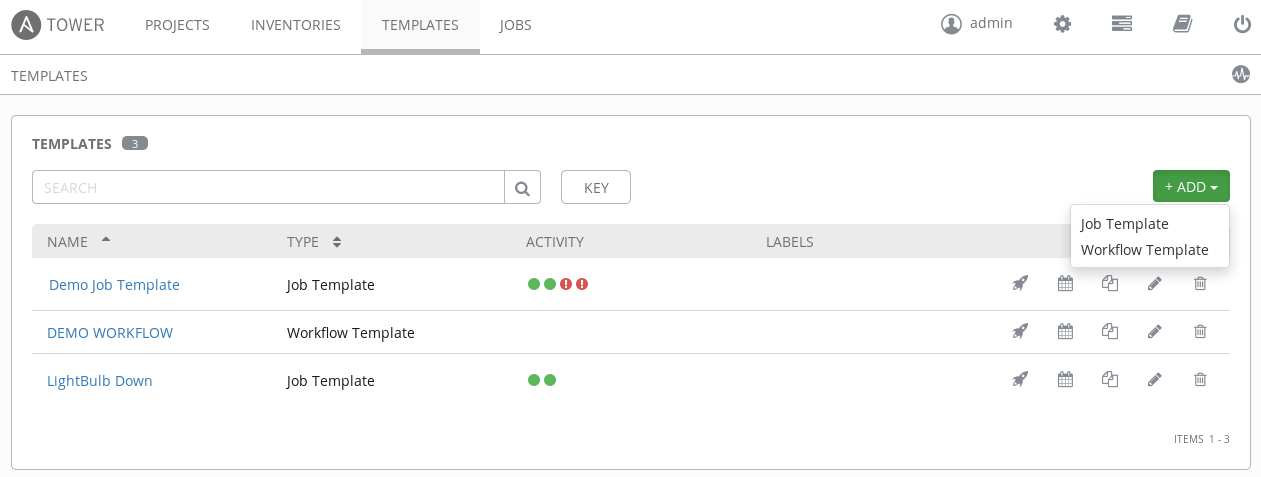
# PROJECTS

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

You can manage Playbooks and Playbook directories by placing them in a **source code management system** supported by Tower, including Git, Subversion, and Mercurial.



# TEMPLATES

A job template is a definition and set of parameters for running an Ansible Playbook.   
Job templates are useful to execute the same job many times and encourage the reuse of Ansible Playbook content and collaboration between teams.

# 

# WORKFLOW TEMPLATE

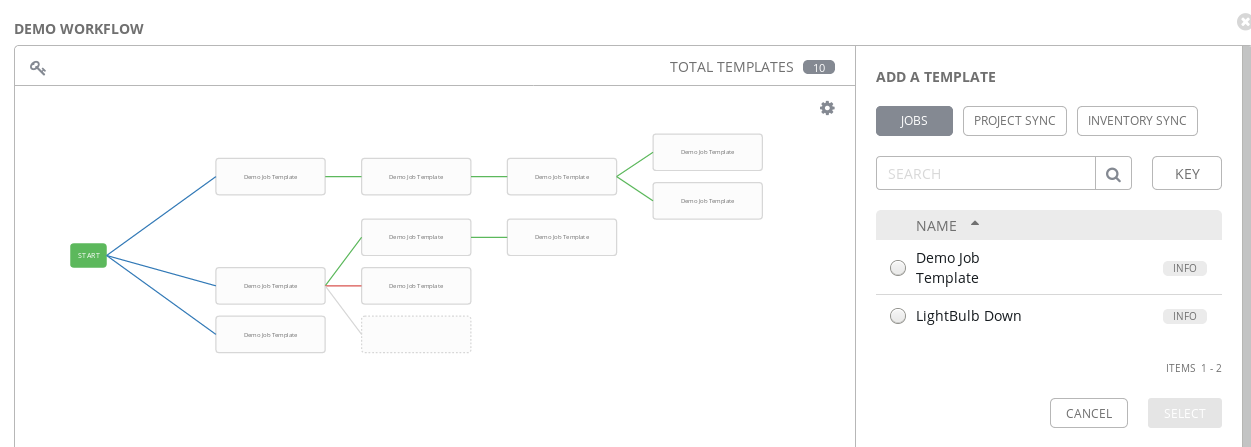
## Standard Job Template

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

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

## Workflow Editor

Workflows allow you to configure a sequence of disparate job templates that may or may not share inventory, playbooks, or permissions. However, workflows have ‘admin’ and ‘execute’ permissions, similar to job templates. A workflow accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit.

1. ****

**NOTE: Workflows are only available to those with Enterprise-level licenses.**

Job templates are linked together using a graph-like structure called nodes. Job template nodes are associated with job templates. A job template can be a part of different workflows or used multiple times in the same workflow. A copy of the graph structure is saved to a workflow job when you launch the workflow.

As the workflow runs, jobs are spawned from the node’s linked template. Nodes linking to a job template which has prompt-driven fields (job\_type, job\_tags, skip\_tags, limit) can contain those fields, and will not be prompted on launch. Job templates with promptable credential and/or inventory, WITHOUT defaults, will not be available for inclusion in a workflow.

A node can have only one parent and can only have children that is linked to a state of success, failure, or always. If always, then the state is neither success or failure. States apply at the node level, not at the workflow job template level. A workflow job will be marked as successful unless it is canceled or encounters an error.

If you attempt to launch a workflow job template that has the following missing pieces, the user interface will notify you as a warning but will still proceed:

* Job template deleted from the node
* A prompted field is provided, but the job template is not set to prompt on launch for the field

To edit and delete a workflow job template, you must have the admin role. To create a workflow job template, you must be an organization admin or a system admin. However, you can run a workflow job template that contains job templates you don’t have permissions for. Similar to projects, organization admins can create a blank workflow and then grant an admin\_role to a low-level user, after which they can go about delegating more access and building the graph. You must have execute access to a job template to add it to a workflow job template.

Other tasks such as the ability to make a duplicate copy and re-launch a workflow can also be performed, depending on what kinds of permissions are granted to a particular user. Generally, you should have permissions to all the resources used in a workflow (like job templates) before relaunching or making a copy.

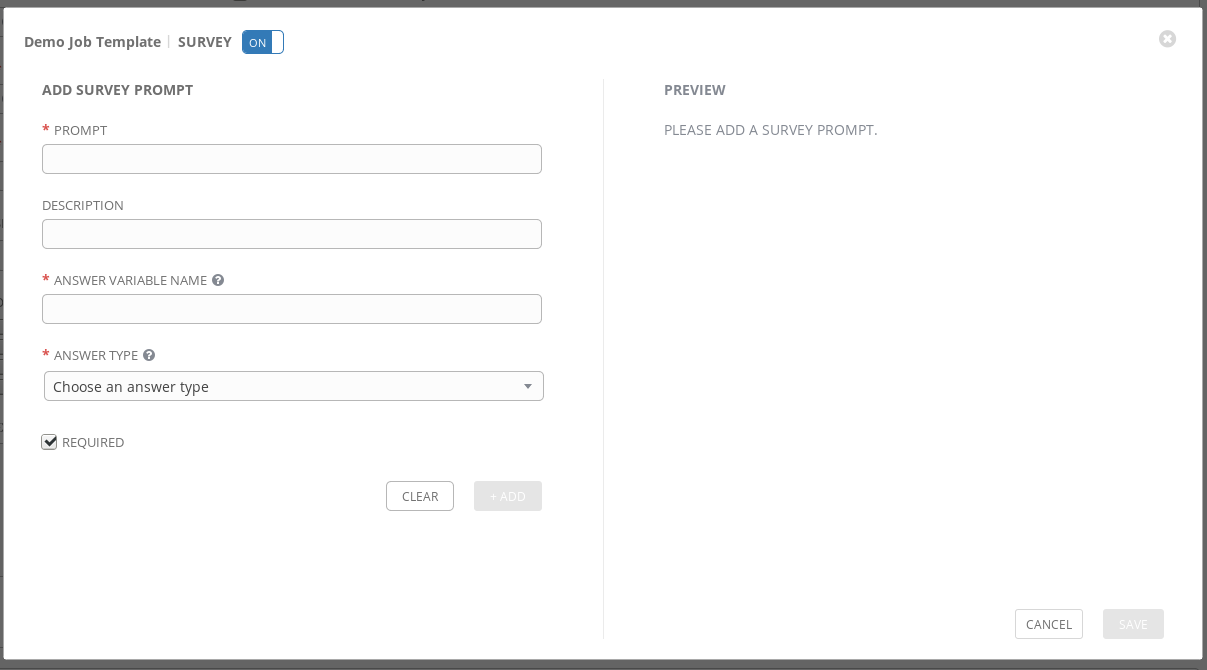
To edit and delete a workflow job template, you must have the admin role. To create a workflow job template, you must be an organization admin or a system admin. However, you can run a workflow job template that contains job templates you don’t have permissions for. Similar to projects, organization admins can create a blank workflow and then grant an admin\_role to a low-level user, after which they can go about delegating more access and building the graph. You must have execute access to a job template to add it to a workflow job template.

Other tasks such as the ability to make a duplicate copy and re-launch a workflow can also be performed, depending on what kind of permissions are granted to a particular user. Generally, you should have permissions to all the resources used in a workflow (like job templates) before relaunching or making a copy.

## User Surveys (Self-Service)

Surveys set extra variables for the playbook similar to ‘Prompt for Extra Variables’ does, but in a user-friendly question and answer way. Surveys also allows for validation of user input. If **Enable Survey** is checked, you can see a button to **Create Survey**.

Use cases for surveys are numerous. An example might be if operations wanted to give developers a “push to stage” button they could run without advanced Ansible knowledge. When launched, this task could prompt for answers to questions such as, “What tag should we release?”

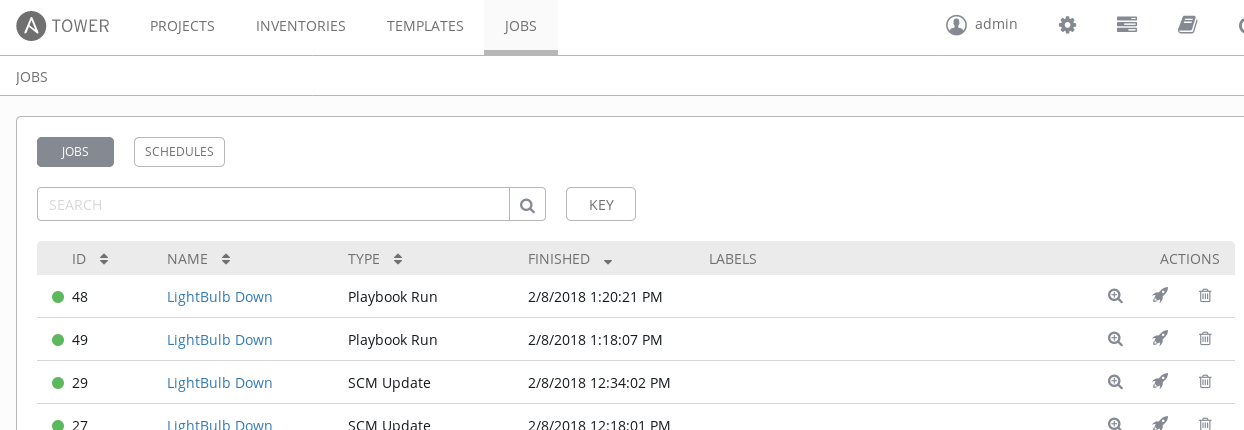
Many types of questions can be asked, including multiple-choice questions.

# JOB SCHEDULING

## NOTIFICATIONS

## JOBS

A job is an instance of Tower launching an Ansible Playbook against an inventory of hosts.

Job results can be easily viewed View the standard out for a more in-depth look

# ADVANCED CONFIGURATION OPTIONS

## Configure

Located in the setup screen you will find the option

Here you will find the most common Advanced Configuration Options:

* Authentication
* Jobs
* Logging Aggregation
* User Interface

## Authentication

## Jobs

## Logging aggregation

# IF YOU WOULD RATHER DO THINGS IN CLI

# Tower-cli examples

1. This is a collection of examples of how to set up Tower using tower-cli that I have collected.
2. More documentation can be found at:

[https://docs.Ansible.com/Ansible Tower/latest/html/towerapi/tower\_cli.html](https://mojo.redhat.com/external-link.jspa?url=https%3A%2F%2Fdocs.ansible.com%2Fansible-tower%2Flatest%2Fhtml%2Ftowerapi%2Ftower_cli.html)

## Installation & Preparation

1. Either install `python2-Ansible Tower-cli` from the OS' repos or use `pip install Ansible Tower-cli`.
2. To set up tower-cli you can use any of the methods described in its documentation, like, e.g.,
3. $ tower-cli config host tower.example.com
4. $ tower-cli config username leeroyjenkins
5. $ tower-cli config password myPassw0rd
6. If you're creating long scripts (e.q. to set up an entire tower) it makes sense to use
7. #! /bin/bash
8. set -e
9. so that the script exits if one of the commands fail

## Config Examples

1. Organization, team, and user
2. # Org
3. tower-cli organization create -n org-example --force-on-exists
4. # User(org-Admin):
5. tower-cli user create --username org-admin-username --password "987654321" --email a@example.com --first-name Org --last-name Admin --force-on-exists --is-superuser False --is-system-auditor False
6. tower-cli role grant --type admin --user org-admin-username --organization "org-example"
7. create teams for normal users and power users with additional permissions (see below):
8. # Teams
9. tower-cli team create -n team-example \
10. --organization org-example \
11. --force-on-exists
12. tower-cli team create -n team-PU-example \
13. --organization org-example \
14. --force-on-exists

## Credentials Creation

1. # see https://github.com/Ansible/tower-cli/issues/452
2. tower-cli credential create -n cred-example --credential-type Machine \
3. --organization org-example --team team-PU-example \
4. --force-on-exists \
5. --inputs 'username: some\_user
6. become\_method: sudo
7. ssh\_key\_data: |
8. '"$(sed 's/^/ /' ${TOWER\_RSA})"

## Inventories

1. This is how an inventory script is added to tower:
2. tower-cli inventory\_script create -n custinv-example-ucmdb \
3. --organization org-example \
4. --script "$(cat ${INVENTORY\_SCRIPT})" \
5. --force-on-exists
6. Here, we create the actual inventory (the Jinja2 templating here is an example that could be used if Ansible's template module would be used to set-up the tower. In that case, `test.inv.vars` would be provided in that context.
7. tower-cli inventory create -n inv-example \
8. --organization org-example \
9. --force-on-exists \
10. {% if test.inv.vars is defined %}
11. --variables "{ {% for item in test.inv.vars %}{{ item }}, {% endfor %} }"
12. {% endif %}
13. Add the script created earlier to the inventory:
14. tower-cli inventory\_source create --name src-example-ucmdb \
15. --inventory inv-example \
16. --source custom --source-script custinv-example-ucmdb \
17. --overwrite true --overwrite-vars true --update-on-launch true \
18. --force-on-exists
19. Now, allow teams to access/edit/admin the inventory
20. tower-cli role grant --type use --team team-PU-example -i inv-example
21. tower-cli role grant --type adhoc --team team-PU-example -i inv-example
22. tower-cli role grant --type update --team team-PU-example -i inv-example
23. tower-cli role grant --type update --team team-MA-example -i inv-example

## Add Projects

1. This example sets up a project with 'manual' SCM (better use git IRL).
2. tower-cli project create -n prj-example \
3. -d "Playbooks for example project" \
4. --organization org-example --scm-type manual \
5. --local-path playbooks-example \
6. --force-on-exists
7. As with the inventories above, we need to grant teams their particular rights.
8. tower-cli role grant --type admin --team team-PU-example --project "prj-example"
9. tower-cli role grant --type use --team team-PU-example --project "prj-example"

tower-cli role grant --type update --team team-PU-example --project "prj-example"

## Create A Job Templates

1. tower-cli job\_template create \
2. -n creative\_job\_name \
3. -i inv-example --playbook hack\_the\_planet.yml \
4. --job-type run --project prj-example \
5. --credential cred-example \
6. --host-config-key somerandomhostconfigkey \
7. --ask-variables-on-launch true \
8. --extra-vars "key1: value1" \
9. --extra-vars "key2: value2" \
10. --force-on-exists
11. As usual, we need to grant access:
12. tower-cli role grant --type admin --team team-PU-example --job-template "creative\_job\_name"
13. tower-cli role grant --type execute --team team-example --job-template "creative\_job\_name"

## Custom Credential Types

1. The following example creates a credential type to store username and password:
2. tower-cli credential\_type create \
3. -n "tower-cli-credential" -d "Custom credential for tower-cli" \
4. --kind cloud \
5. --inputs '{ "fields": [ { "type": "string", "id": "username", "label": "Username for Tower" }, { "label": "Tower Password", "secret": true, "type": "string", "id": "password" } ], "required": [ "username", "password" ] }' \
6. --injectors '{ "extra\_vars": { "tower\_cli\_user": "{{ username }}", "tower\_cli\_password": "{{password}}" } }' \
7. --force-on-exists
8. Now, as in one of the previous examples, we create an actual credential of that type:
9. tower-cli credential create \
10. -n "cred-example-tower-cli" --credential-type "tower-cli-credential" \
11. --organization org-example \
12. --force-on-exists \
13. --inputs '{ "username": "tower-cli-cred-user-example","password": "'"${TOWER\_CLI\_PASSWORD}"'"}'
14. And, we need to allow the power-users to use that credential
15. tower-cli role grant --type use --team team-PU-example --credential "cred-example-tower-cli"
16. Lastly, we associate the credential with a job-template, so that it is actually accessible from there
17. tower-cli job\_template associate\_credential \
18. --credential cred-example-tower-cli \
19. --job-template creative\_job\_name

## Add Users To Teams

1. tower-cli user create --username power\_user-example --password redhat --email eb@example.example.com --first-name El --last-name Barto --force-on-exists --is-superuser False --is-system-auditor False
2. tower-cli team associate --team team-PU-example --user power\_user-example
3. tower-cli user create --username normal\_user-example --password redhat --email bs@example.example.com --first-name Bart --last-name Simpson --force-on-exists --is-superuser False --is-system-auditor False
4. tower-cli team associate --team team-example --user normal\_user-example

# 

# **MODULES**

There are around 1,400 modules in Ansible if there are any that you specifically want to know about you can:

1. Go on-line at doc.Ansible.com
2. From the CLI on your Ansible host

Ansible-doc -l to list all

Ansible-doc -l |grep <part of name> to pull a list of modules specific modules i.e. win, net, virtual, ect…

From the moment you install Ansible you have them all and all of the same documentation you find online you can get right from the CLI.

For instance the copy module

Go to [http://docs.Ansible.com/Ansible/latest/copy\_module.html](http://docs.ansible.com/ansible/latest/copy_module.html) and you can see that the online documentation provides

* copy - Copies files to remote locations
* Synopsis
* Options
* Examples
* Return Values
* Notes
* Status
* Maintenance Info

On your Ansible host open a command line

**Ansible-doc copy**

Now scroll down with the down arrow and you can see that all the same including the examples are present

**Why is this so great?**

You no longer need to carry documentation into a datacenter to work with Ansible, you don't need an external connection to search stackoverflow or google for help. In a lot of cases you can obtain it right there!

## Writing Modules

[http://docs.Ansible.com/Ansible/latest/dev\_guide/developing\_modules.html](http://docs.ansible.com/ansible/latest/dev_guide/developing_modules.html)

[http://docs.Ansible.com/Ansible/latest/dev\_guide/index.html](http://docs.ansible.com/ansible/latest/dev_guide/index.html)

[http://docs.Ansible.com/Ansible/latest/dev\_guide/developing\_plugins.html](http://docs.ansible.com/ansible/latest/dev_guide/developing_plugins.html)

<https://blog.toast38coza.me/custom-Ansible-module-hello-world/>

https://github.com/pmarkham/writing-ansible-modules-in-bash/blob/master/ansible\_bash\_modules.md

# ANSIBLE EXTRAS

## The Workshop

* + 1. Just let me know how many students you will have and I can build a 4 node infrastructure for each student in AWS that will stay up for 5 days (The course takes 2 to 3 hours of dedicated time).

[https://network-automation.github.io/linklight/exercises/Ansible\_engine/](https://network-automation.github.io/linklight/exercises/ansible_engine/)

[https://network-automation.github.io/linklight/exercises/Ansible\_tower/](https://network-automation.github.io/linklight/exercises/ansible_tower/)

<https://network-automation.github.io/linklight/exercises/networking_v2/>

## Play Book Library

For about everything you can think of --> This is OPC (other people's code) so it is meant as an example only and is not supported and there is no implied warranty. HOWEVER, THIS WILL DECREASE YOUR DEVELOPMENT CYCLES!

<http://34.227.71.23/workshop/Ansible-CICD-Playbook-Library-OPC.tar.gz>

<http://34.227.71.23/workshop/Ansible-Linux-Playbook-Library-OPC.tar.gz>

<http://34.227.71.23/workshop/Ansible-NETWORK-Playbook-Library-OPC.tar.gz>

<http://34.227.71.23/workshop/Ansible-Satelite-Playbook-Library-OPC.tar.gz>

<http://34.227.71.23/workshop/Ansible-Windows-Playbook-Library-OPC.tar.gz>

<http://34.227.71.23/workshop/Ansible-infrastructure-Playbook-Library-OPC.tar.gz>

## Links

For use in setting up cloud (AWS) Demo

<https://github.com/mgmt-sa-tiger-team/lightbulb>

<https://github.com/network-automation/linklight>

## Decks Used Demo

[http://34.227.71.23/workshop/decks/Ansible-essentials.html#/](http://34.227.71.23/workshop/decks/ansible-essentials.html" \l "/)

[http://34.227.71.23/workshop/decks/intro-to-Ansible Tower.html#/](http://34.227.71.23/workshop/decks/intro-to-ansible-tower.html" \l "/)

## [Request A License](http://34.227.71.23/workshop/decks/intro-to-ansible-tower.html" \l "/)

[https://www.Ansible.com/license](https://www.ansible.com/license)

## **Additional Links**

[https://www.Ansible.com/](https://www.ansible.com/)

[http://docs.Ansible.com/](http://docs.ansible.com/)

## **Free Red Hat Training**

<http://redhatgov.io/>

## **Backing Up Ansible Tower**

[http://docs.Ansible.com/Ansible Tower/latest/html/administration/backup\_restore.html](http://docs.ansible.com/Ansible Tower/latest/html/administration/backup_restore.html)

## **Non Red Hat Tutorials**

[https://sysadmincasts.com/episodes/43-19-minutes-with-Ansible-part-1-4](https://sysadmincasts.com/episodes/43-19-minutes-with-ansible-part-1-4)

[https://sysadmincasts.com/episodes/45-learning-Ansible-with-vagrant-part-2-4](https://sysadmincasts.com/episodes/45-learning-ansible-with-vagrant-part-2-4)

[https://sysadmincasts.com/episodes/46-configuration-management-with-Ansible-part-3-4](https://sysadmincasts.com/episodes/46-configuration-management-with-ansible-part-3-4)

https://sysadmincasts.com/episodes/47-zero-downtime-deployments-with-Ansible-part-4-4

## **Open Source Apps**

ARA is an acronym. It means [Ansible Run Analysis](https://ara.readthedocs.io/en/latest/). It provides you with a detailed analysis of your Ansible playbook runs, which includes:

* Tasks description, status and output
* Play description and duration
* Hosts involved
* Files executed
* Parameters used

## **ARA**

[https://blog.octo.com/en/Ansible-reporting-with-ara-Ansible-run-analysis/](https://blog.octo.com/en/ansible-reporting-with-ara-ansible-run-analysis/)

### More About ARA

<https://dmsimard.com/2016/05/21/ara-an-idea-to-store-browse-and-troubleshoot-ansible-playbook-runs/>

## Tower Workflow Approvals

<https://github.com/wtcross/tower-workflow-approvals>

## A**IX**/Solaris

https://github.com/aixoss/Ansible-playbooks

http://docs.Ansible.com/Ansible/latest/modules/aix\_inittab\_module.html#aix-inittab-module

http://docs.Ansible.com/Ansible/latest/modules/aix\_lvol\_module.html#aix-lvol-module

http://docs.Ansible.com/Ansible/2.4/solaris\_zone\_module.html

[http://docs.Ansible.com/Ansible/latest/modules/dladm\_etherstub\_module.html#dladm-etherstub-module](http://docs.ansible.com/ansible/latest/modules/dladm_etherstub_module.html" \l "dladm-etherstub-module)

## [Microsoft](http://docs.ansible.com/ansible/latest/modules/dladm_etherstub_module.html" \l "dladm-etherstub-module)

* + - * 1. There are many azure and windows modules within Ansible, however Microsoft has been working hard at creating an extended set of modules to assist in Microsoft tasks. I have added playbook examples into the Linux tar playbook library above

<https://docs.microsoft.com/en-us/azure/Ansible/Ansible-overview>

https://github.com/mgmt-sa-tiger-team/skylight

### 

## VMware

<https://github.com/GoKEV/Lab-Playbooks.git>

[https://github.com/vmware/Ansible-role-vcenter.git](https://github.com/vmware/ansible-role-vcenter.git)

<https://github.com/zwindler/Ansible-deploy-vmware-guest>

## Terraform

<https://github.com/OpenStackSanDiego/osa-workshop>

<https://www.arctiq.ca/our-blog/2019/4/6/ansible-terraform-gcp-automation-goodness>

## Terraform Vs Ansible

Generally, Ansible and Terraform work differently - Terraform is declarative in a top-down format that describes what your infrastructure is. Ansible is declarative in a bottom-up way that describes how to build your infrastructure. With Terraform, to fix it you need to edit your terraform description of your infrastructure and redeploy; with Ansible you can fix in a variety of targeted ways.

Ansible is far more flexible in what it allows you to do - not just declare and provision infrastructure, but also do patching, deployment, one-off changes, network management, and more. Terraform is fairly static - it's "give me an infrastructure that fits this definition", while Ansible is a more general "perform this automation based on these steps

Ultimately, in order to make config and state changes on infrastructure, you'd still have to use something in addition to terraform. Second to that, terraform can't do the whole stack... Ie no network, no configuration, no self service, no reliable and stable API (due to immaturity), etc.

So the big things to focus on here with Ansible are ease-of-use and versatility. Tower will provide the ability to scale with Ansible and put controls in place with a compliance-heavy organization like Q2ebanking.

# Servicenow, Satellite, And Ansible Tower Integration

### 

**Playbooks For The Satellite Patching Automation From The Blogs Below Can Now Be Found At:**

https://github.com/p-avery/sat-ansible

<https://averytechguy.com/2019/04/22/ansible-tower-smarter-inventories/>

https://averytechguy.com/2018/10/22/add-custom-dynamic-inventory-in-ansible-tower-servicenow/

https://averytechguy.com/2018/12/22/auto-remediation-with-zabbix-and-ansible-tower-part-1/

<https://averytechguy.com/2019/01/19/auto-remediation-with-zabbix-and-ansible-tower-part-2/>

https://averytechguy.com/2018/09/14/auto-patch-schedule-with-satellite-6-3-and-ansible-tower-part-1/

https://averytechguy.com/2018/09/14/auto-patch-schedule-with-satellite-6-3-and-ansible-tower-part-2/

https://averytechguy.com/2018/09/17/auto-patch-schedule-with-satellite-6-3-and-ansible-tower-part-3/

<https://averytechguy.com/2019/01/24/set-custom-virtual-environments-and-ansible-versions-in-tower/>

<https://averytechguy.com/2017/06/26/run-ansible-playbook-during-service-provisioning/>

# SECURITY

## Ansible

https://ansiblelockdown.io/

https://github.com/ansible/ansible-lockdown

https://www.ansible.com/blog/security-baseline-configuration-automation

https://www.ansible.com/security-stig

## Ansible Galaxy

<https://galaxy.ansible.com/dev-sec>

## RHEL 7

<https://github.com/MindPointGroup/RHEL7-CIS>

https://galaxy.ansible.com/HarryHarcourt/Ansible-RHEL7-CIS-Benchmarks

http://www.biffsocko.com/?p=1240

## RHEL 6

https://github.com/major/cis-rhel-ansible

https://major.io/2015/08/05/automated-testing-for-ansible-cis-playbook-on-rhelcentos-6/

## General CIS

https://github.com/dev-sec/ansible-os-hardening

https://medium.com/@cosmin.ciobanu/the-quest-for-os-hardening-8affa979f36a

## STIG

https://iase.disa.mil/stigs/os/unix-linux/Pages/red-hat.aspx

## Ubuntu

https://blog.codybunch.com/2015/06/22/Ansible-CIS-and-Ubuntu/

## Windows

https://github.com/dev-sec/ansible-windows-hardening

https://github.com/juju4/ansible-harden-windows

https://github.com/Neutrollized/cis-windows-level2

## Center For Internet Security

https://www.cisecurity.org/cis-benchmarks/

# VENDOR SPECIFIC RESOURCES

## Pure

https://docs.ansible.com/ansible/2.6/modules/list\_of\_storage\_modules.html

## Sysprep

https://nathanwebster.me/2017/sysprep-ec2-with-ansible/

## Workstation - Win And Mac

https://opensource.com/article/18/3/manage-workstation-ansible

https://opensource.com/article/18/3/manage-your-workstation-configuration-ansible-part-2

https://opensource.com/article/18/5/manage-your-workstation-ansible-part-3

https://fedoramagazine.org/using-ansible-setup-workstation/

https://github.com/rothgar/ansible-workstation

https://github.com/chapmanu/ansible-workstation

https://blog.josephkahn.io/articles/ansible/

https://pbassiner.github.io/blog/automating\_my\_dev\_setup.html

https://github.com/cdown/ansible-desktop

https://github.com/geerlingguy/mac-dev-playbook

https://daemonza.github.io/2017/03/06/using-ansible-to-automate-my-macbook-setup/

https://medium.com/tensult/setting-up-a-mac-using-ansible-3dcf63c1d324

https://medium.com/@ngerasimatos/how-to-automate-your-apple-macos-setup-with-ansible-f06701971b2a

## VMware

https://blogs.vmware.com/vcloud/2018/06/ansible-modules-for-vcloud-director-released.html

https://github.com/ansible/community/wiki/VMware

https://raymii.org/s/software/Ansible\_\_Dynamic\_Inventory\_From\_VMware\_vCenter.html

https://www.justai.net/en/blog/using-ansible-to-manage-vmware-infrastructure/

https://everythingshouldbevirtual.com/creating-vsphere-vms-using-ansible/

https://docs.ansible.com/ansible/latest/vmware/vmware\_intro.html

https://vuptime.io/2018/02/07/ansible-vmware\_modules\_-\_first\_steps/

https://github.com/vmware/ansible-role-vcenter

https://github.com/MindPointGroup/ansible-vmware-provisioning

https://github.com/stone-payments/ansible-vmware

https://github.com/vmware/ansible-module-vcloud-director

https://github.com/zwindler/ansible-deploy-vmware-guest

https://everythingshouldbevirtual.com/creating-vsphere-vms-using-ansible/

http://virtualelephant.com/2017/11/27/infrastructure-as-code-ansible-for-vmware-nsx/

## GIT

https://docs.ansible.com/ansible/latest/modules/git\_module.html

https://ansible-manual.readthedocs.io/en/v1.8.4-doc/git\_module.html

https://www.jeffgeerling.com/blog/2018/cloning-private-github-repositories-ansible-on-remote-server-through-ssh

https://opensource.com/article/17/8/ansible-environment-management

https://codereviewvideos.com/course/ansible-tutorial/video/git-your-deploy-just-right

https://blog.rackspace.com/how-to-deploy-ansible-accessible-explanation

https://github.com/sovereign/sovereign

## F5

https://www.ansible.com/integrations/networks/f5

https://github.com/F5Networks/f5-ansible

https://clouddocs.f5.com/products/orchestration/ansible/devel/usage/playbook\_tutorial.html

https://network-automation.github.io/linklight/decks/ansible\_f5.pdf

https://media.readthedocs.org/pdf/f5-ansible-romain/latest/f5-ansible-romain.pdf

https://ansible-manual.readthedocs.io/en/v1.8.4-doc/bigip\_pool\_module.html

https://github.com/lukaszsedek/f5-ansible

https://www.ansible.com/hubfs/pdf/automating-f5-big-ip-platform-with-ansible.pdf?hsLang=en-us

https://www.ansible.com/blog/automating-f5-big-ip-using-ansible-webinar

https://www.ansible.com/hubfs/pdf/f5-ansible-overview.pdf?hsLang=en-us

## Cisco Network

https://www.ansible.com/integrations/networks/cisco

https://learningnetwork.cisco.com/blogs/vip-perspectives/2017/08/10/automating-cisco-using-ansible

https://developer.cisco.com/docs/nx-os/#automation-ansible

https://www.rogerperkin.co.uk/network-automation/ansible/ansible-tutorial-network-engineers/

https://packetpushers.net/ansible-cisco-snmp/

https://networklore.com/ansible-ios\_config/

https://xrdocs.io/application-hosting/tutorials/2018-08-06-comprehensive-guide-to-ansible-on-ios-xr/

https://github.com/colin-mccarthy/ansible-playbooks-for-cisco-ios

https://github.com/colin-mccarthy/cisco\_base\_config

https://github.com/colin-mccarthy/asa

https://github.com/colin-mccarthy/ansible\_upgrade

## Cisco UCS

https://blogs.cisco.com/datacenter/ansible-and-ucs-manager

https://github.com/CiscoUcs/ucsm-ansible

https://docs.ansible.com/ansible/latest/modules/ucs\_vlans\_module.html

https://docs.ansible.com/ansible/latest/modules/list\_of\_remote\_management\_modules.html

https://www.networkcomputing.com/networking/automating-ucs-manager-configuration-ansible/2034118773

https://www.univention.com/blog-en/2016/10/ansible-modules-for-the-automation-of-ucs-specific-tasks/

https://blogs.cisco.com/datacenter/scaling-to-pb-within-minutes-the-road-to-full-automation-for-scale-out-storage-with-cisco-ucs

https://github.com/btotharye/ansible-ucs

https://github.com/CiscoUcs/ucsm-ansible

https://github.com/CiscoUcs/ucsm-ansible-docker

https://github.com/CiscoUcs/ansible-role-ucs

https://github.com/CiscoUcs/imc-ansible

https://github.com/CiscoUcs/ansible-role-ucs-storage

https://github.com/CiscoUcs/ansible-role-ucs-lan

https://github.com/CiscoUcs/ansible-role-ucs-san

https://github.com/CiscoUcs/DockerEE\_UCS\_BM

## Brocade

https://github.com/brocade/ansible

https://docs.ansible.com/ansible/2.5/modules/ironware\_command\_module.html

https://github.com/LateRoomsGroup/ansible-module-brocade

## Dell EMC Isilon

https://ansible-dellos-docs.readthedocs.io/en/latest/

https://media.readthedocs.org/pdf/ansible-dellos-docs/latest/ansible-dellos-docs.pdf

https://github.com/Isilon

https://github.com/dell/Dell-EMC-Ansible-Modules-for-iDRAC

https://github.com/dell/redfish-ansible-module

https://community.emc.com/people/PaulCork/blog/2018/09/24/ansible-modules-for-vmax-and-powermax

## Hatachi

https://community.hitachivantara.com/docs/DOC-1007676-ansible-templates

https://docs.ansible.com/ansible/latest/modules/list\_of\_storage\_modules.html

https://github.com/miquelMariano/VSP\_Gx00

https://github.com/Akrog/ansible-role-storage

## IBM Qradar

https://github.com/josh-morin/qradar

<https://community.ibm.com/community/user/security/blogs/chris-schulz/2018/10/25/qradar-ansible-module>

## Veritas Storage Foundation

https://github.com/nkwd/ansible-playbooks

https://github.com/bradymholt/veritas

## Data Guard

https://hk.saowen.com/a/d846726cb47e8b9ee96760e1861495d9bf7bd50754677bcac7d6e233f49e7262

https://oracle-cloud-infrastructure-ansible-modules.readthedocs.io/en/latest/modules/oci\_data\_guard\_association\_facts\_module.html

## Oracle

https://oracle-cloud-infrastructure-ansible-modules.readthedocs.io/en/latest/modules/list\_of\_cloud\_modules.html

https://fritshoogland.wordpress.com/2014/09/14/using-ansible-for-executing-oracle-dba-tasks/

http://www.oracledbwr.com/ansible/oracle-automation-create-a-dba-user-using-ansible-tool/

http://www.oracledbwr.com/ansible/oracle-automation-oracle-database-creation-using-ansible-tool/

https://blog.pythian.com/cloning-oracle-grid-infrastructure-using-ansible/

https://www.nloug.nl/downloads/ogh20180613\_ilmar\_kerm\_mikael\_sandstr%C3%96m.pdf

## Active Directory

https://6uellerbpanda.gitlab.io/posts/2018/04-17-Active-Directory-usermanagement-with-Ansible/

https://docs.ansible.com/ansible/latest/modules/win\_domain\_computer\_module.html

https://docs.ansible.com/ansible/2.4/win\_domain\_user\_module.html

## IBM Tivoli

https://github.com/zrongh90/system\_monitor

https://github.com/mikesimos/tsm-client

https://gitlab.com/MathieuMD/ansible-role-tsm-client

http://kairo.eti.br/aix-powervm-automation-with-ansible.html

## Cyberark

https://github.com/cyberark/ansible-modules

https://github.com/cyberark/ansible-aim-provider

https://github.com/infamousjoeg/cyberark-ansible-lamp

https://github.com/jamboubou/ansible-cyberark-psmp

https://github.com/cyberark/pas-orchestrator

https://github.com/cyberark/pvwa

https://github.com/jellevandehaterd/ansible-role-cyberarkpasswordvault-lookup

https://github.com/jeepapichet/ansible-cyberark-demo

https://github.com/cyberark/ansible-conjur-host-identity

https://github.com/infamousjoeg/cyberark-ansible-demo

https://github.com/infamousjoeg/conjur-ansible-framework

https://github.com/cyberark/cpm

https://github.com/kikidun/personal-projects

https://github.com/cyberark/psm

https://github.com/CaptainFluffyToes/ansible\_aim\_demo

<https://github.com/infamousjoeg/provisioning>

## **Palo Alto**

<https://www.ansible.com/integrations/networks/palo-alto>

<https://github.com/PaloAltoNetworks/ansible-pan>

<http://paloaltonetworks.github.io/ansible-pan/examples.html>

<https://medium.com/@IrekRomaniuk/palo-alto-firewall-on-azure-8c3d52511166>

<https://libraries.io/github/PaloAltoNetworks/ansible-pan>

<https://ansible-pan.readthedocs.io/en/latest/>

<https://github.com/PaloAltoNetworks>

<https://github.com/PaloAltoNetworks>

<https://github.com/XenitAB/ansible-palo-alto-deploy.git>

<https://github.com/PaloAltoNetworks/terraform-ansible-intro>

## **Aruba**

<https://github.com/aruba>

<https://github.com/aruba/aruba-ansible-modules>

<https://github.com/aruba/aruba-switch-ansible>

<https://github.com/aruba/aruba-switch-ansible/wiki>

<https://github.com/aruba/aruba-switch-ansible/wiki/Ansible-Tower-Integration>

<https://github.com/HPENetworking/ansible-hpe-cw7>

## **Rubric**

<https://www.rubrik.com/wp-content/uploads/2017/10/DATA-SHEET-Rubrik-Automation-APIs.pdf>

## **Oracle and Ansible - From Oracle**

<https://blogs.oracle.com/cloud-infrastructure/autoscaling-oracle-e-business-suite-application-servers>

<https://docs.cloud.oracle.com/iaas/Content/API/SDKDocs/ansible.htm>

<https://blogs.oracle.com/oracle-systems/ansible-for-oracle-pca-v2-v2>

<https://blogs.oracle.com/cloud-infrastructure/announcing-oracle-cloud-infrastructure-ansible-modules>

<https://docs.cloud.oracle.com/iaas/Content/API/SDKDocs/ansibleinventoryscript.htm>

<https://docs.cloud.oracle.com/iaas/Content/API/SDKDocs/ansiblesamples.htm>

<https://blogs.oracle.com/oracle-systems/ansible-for-oracle-pca-v2-v2>

<https://blogs.oracle.com/oracleuniversity/how-to-use-ansible-to-set-up-a-vncserver-on-your-oracle-cloud-services>

[There is a lot more on Oracles web site!](https://blogs.oracle.com/oracleuniversity/how-to-use-ansible-to-set-up-a-vncserver-on-your-oracle-cloud-services)

## Ansible Galaxy Oracle

<https://galaxy.ansible.com/search?deprecated=false&keywords=Oracle&order_by=-relevance&page=1&page_size=100>

## [**Community Oracle**](https://galaxy.ansible.com/search?deprecated=false&keywords=Oracle&order_by=-relevance&page=1&page_size=100)

<https://www.thatfinnishguy.blog/2018/01/02/loading-customizations-to-e-business-suite-with-ansible/>

<https://www.shadandy.com/ansible-oracle-database-clones/>

<https://blog.pythian.com/automating-oracle-patching-with-an-ansible-module/>

<https://ronekins.wordpress.com/2018/07/12/oracle-database-automation-with-ansible-awx/>

# USING LINKLIGHT AND SKYLIGHT WITH ANSIBLE TOWER

## Required

The VM

2 Licenses (Request 2 Demo licenses from your sales team 30 days) 500 nodes for your ansible system and a 100 node license for the Windows provisioner

### Repos

subscription-manager repos --disable '\*'

subscription-manager repos --enable=rhel-7-server-rpms

subscription-manager repos --enable=rhel-server-rhscl-7-rpms

subscription-manager repos --enable=rhel-7-server-optional-rpms

subscription-manager repos --enable rhel-7-server-ansible-2.8-rpms

## Install prerequisites

(as root)

yum -q list installed epel-release-latest-7 &>/dev/null && echo "epel-release-latest-7 is installed" || yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm –skip-broken

yum-config-manager --enable epel

yum -q list installed git &>/dev/null && echo "git is installed" || yum install -y git --skip-broken

yum -q list installed python2-pip &>/dev/null && echo "python2-pip is installed" || yum install -y python2-pip --skip-broken

yum -q list installed ansible &>/dev/null && echo "ansible is installed" || yum install -y ansible –skip-broken

yum-config-manager –disable epel

pip install --upgrade pip

pip install boto3==1.9.111

pip install boto==2.49.0

pip install botocore==1.12.111

pip install awscli==1.16.121

## Sendgrid

SendGrid components are optional if you want to email your students when the environments are complete

pip install sendgrid==5.6.0

# CREATE A PROJECT FOR LINKLIGHT

LinkLight playbooks are great playbooks for example of AWS simplification and will automatically:

* Create a VPC
* Create Subnet
* Create Gateway
* Create Route
* Create ssh key

## Open Ansible Tower In Your Browser And Goto “Projects”

Create a LinkLight project (Linux)

* Choose the scm type “git”
* Use the following git url: [https://github.com/ansible/workshops.git](https://github.com/network-automation/linklight.git)

Create another for Skylight (Windows)

* Choose the scm type “git”
* Use the following git url: <https://github.com/mgmt-sa-tiger-team/skylight>

## [Create A Free Sendgrid Account (Optional)](https://github.com/network-automation/linklight.git)

This is if you want to email users that there env has completed once tower builds it.

[https://sendgrid.com/marketing/sendgrid-services/?cvosrc=PPC.Google.sendgrid&cvo\_cid=SendGrid%20-%20US%20-%20Brand%20-%20(English)&mc=Paid%20Search&mcd=AdWords&keyword=sendgrid&network=g&matchtype=e&mobile=&content=&search=1&gclid=CMzo9crH9NMCFcS6wAod8eMFLg](https://sendgrid.com/marketing/sendgrid-services/?cvosrc=PPC.Google.sendgrid&cvo_cid=SendGrid - US - Brand - (English)&mc=Paid Search&mcd=AdWords&keyword=sendgrid&network=g&matchtype=e&mobile=&content=&search=1&gclid=CMzo9crH9NMCFcS6wAod8eMFLg)

## Sign Into Your **AWS** Account

<https://console.aws.amazon.com/console/home?region=us-east-1>

## Boto

Create (FOR COMMAND LINE USE) a boto configuration file containing your AWS access key ID and secret access key. - [https://console.aws.amazon.com/iam/home?region=us-west-2#/users/](https://console.aws.amazon.com/iam/home?region=us-west-2" \l "/users/)<USERNAME?section=security\_credentials

The file should contain the following:

[default]

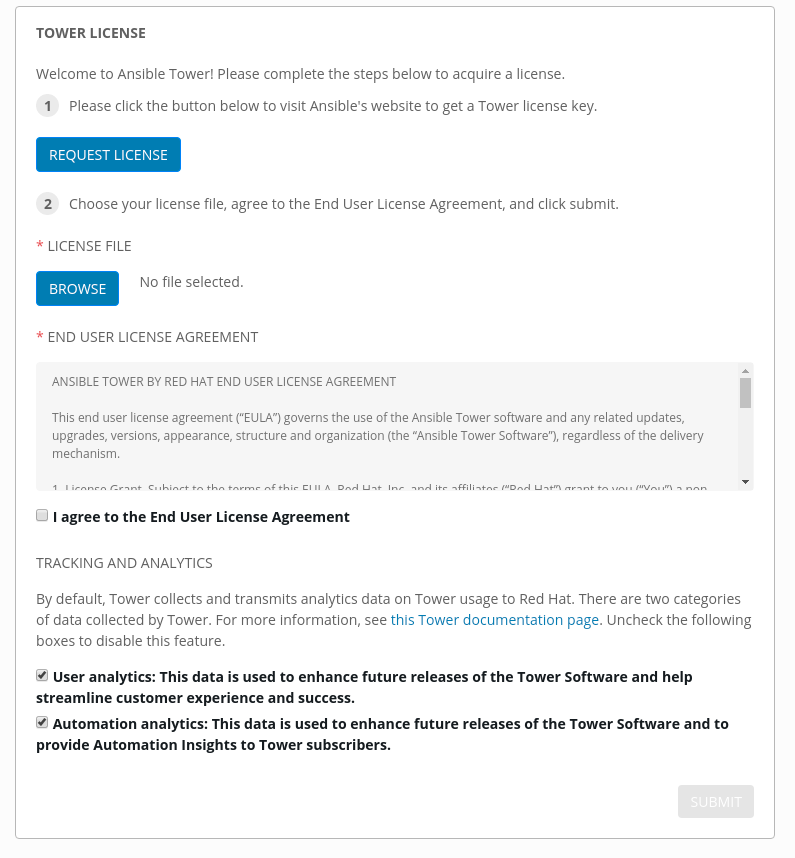
aws\_access\_key\_id = [Access Key Identifier always starts with AKI [https://console.aws.amazon.com/iam/home?region=us-east-1#/users/USERNAME?section=groups]](https://console.aws.amazon.com/iam/home?region=us-east-1" \l "/users/USERNAME?section=groups%5D)

aws\_secret\_access\_key = [secret key can be generated at the same location [https://console.aws.amazon.com/iam/home?region=us-east-1#/users/USERNAME?section=groups]](https://console.aws.amazon.com/iam/home?region=us-east-1" \l "/users/USERNAME?section=groups%5D)

# LOGIN

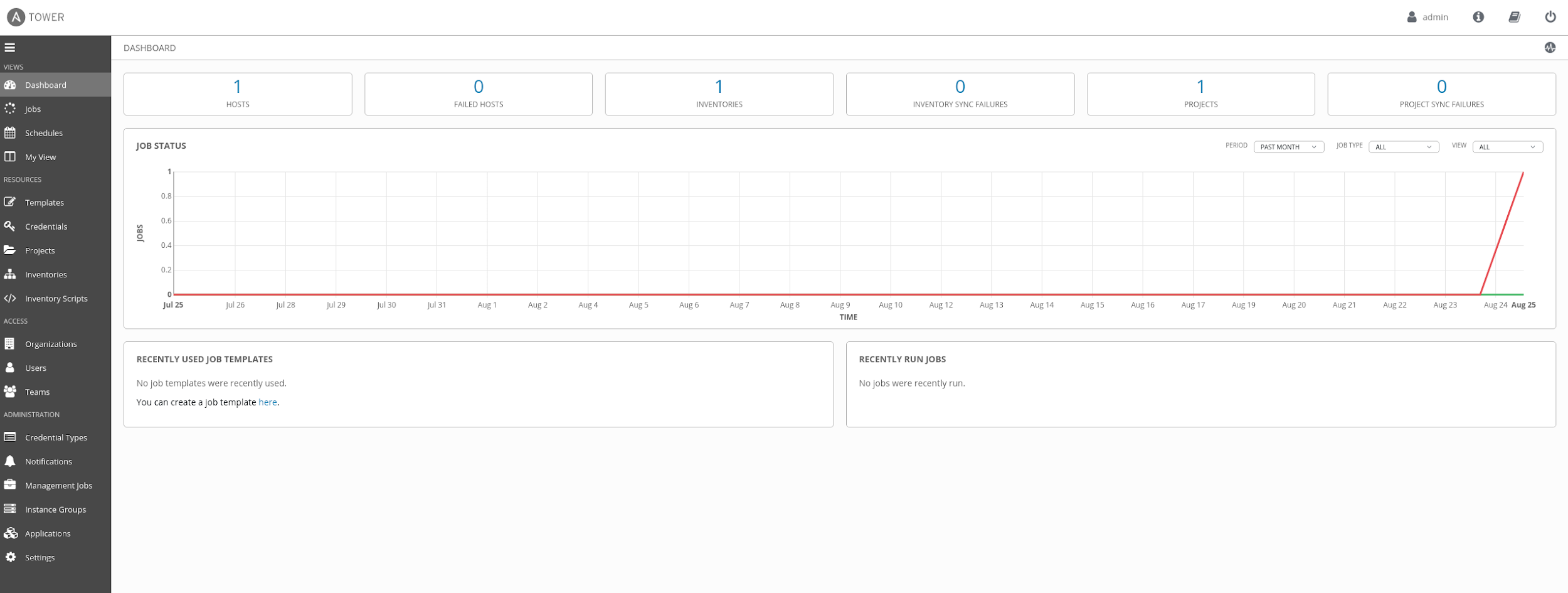
The first item you will come to is the login screen where you will enter your username (admin) and password (r3dh4t7!)

## License

When you install the system yourself the second screen is the License screen. Save your license to a place on your desktop and browse to find it. Then accept the EULA and select SUBMIT

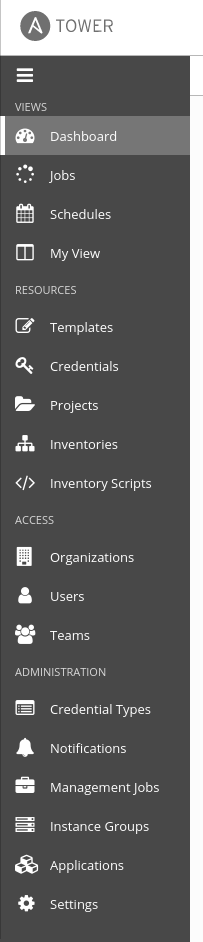
## Welcome To Ansible Tower

[https://<student>.<yourcompanyhere>.rhdemo.io/#/home](https://student1.bpx-poc.rhdemo.io/" \l "/home)

The First screen is your Tower home screen. Here you can see at a glance what your system is up to. Notice the menu on the left.

## Navigation Menu

The navigation menu lists all the components we will be working with



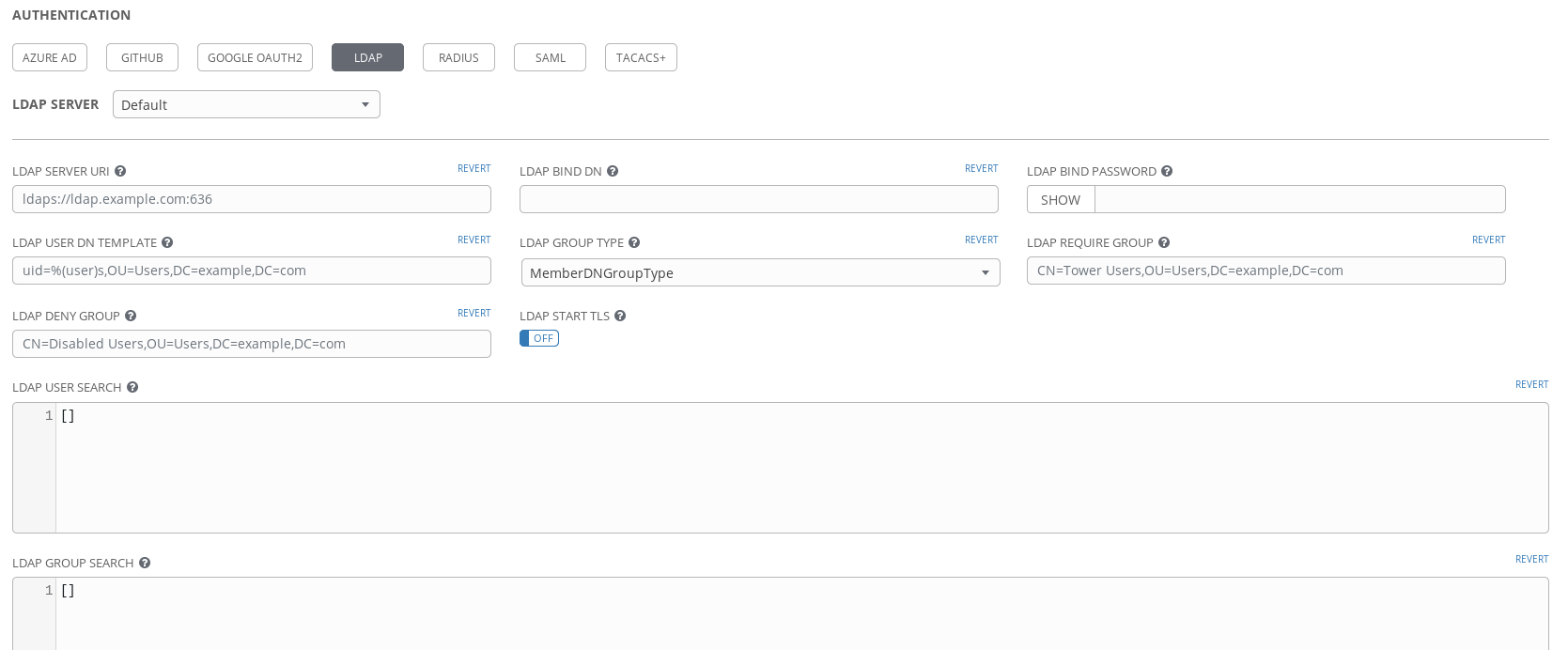
# SETTINGS

[Last item on menu](https://student1.bpx-poc.rhdemo.io/" \l "/settings) on the left

[https://<student>.<yourcompanyhere>.rhdemo.io/#/settings](https://student1.bpx-poc.rhdemo.io/" \l "/settings)

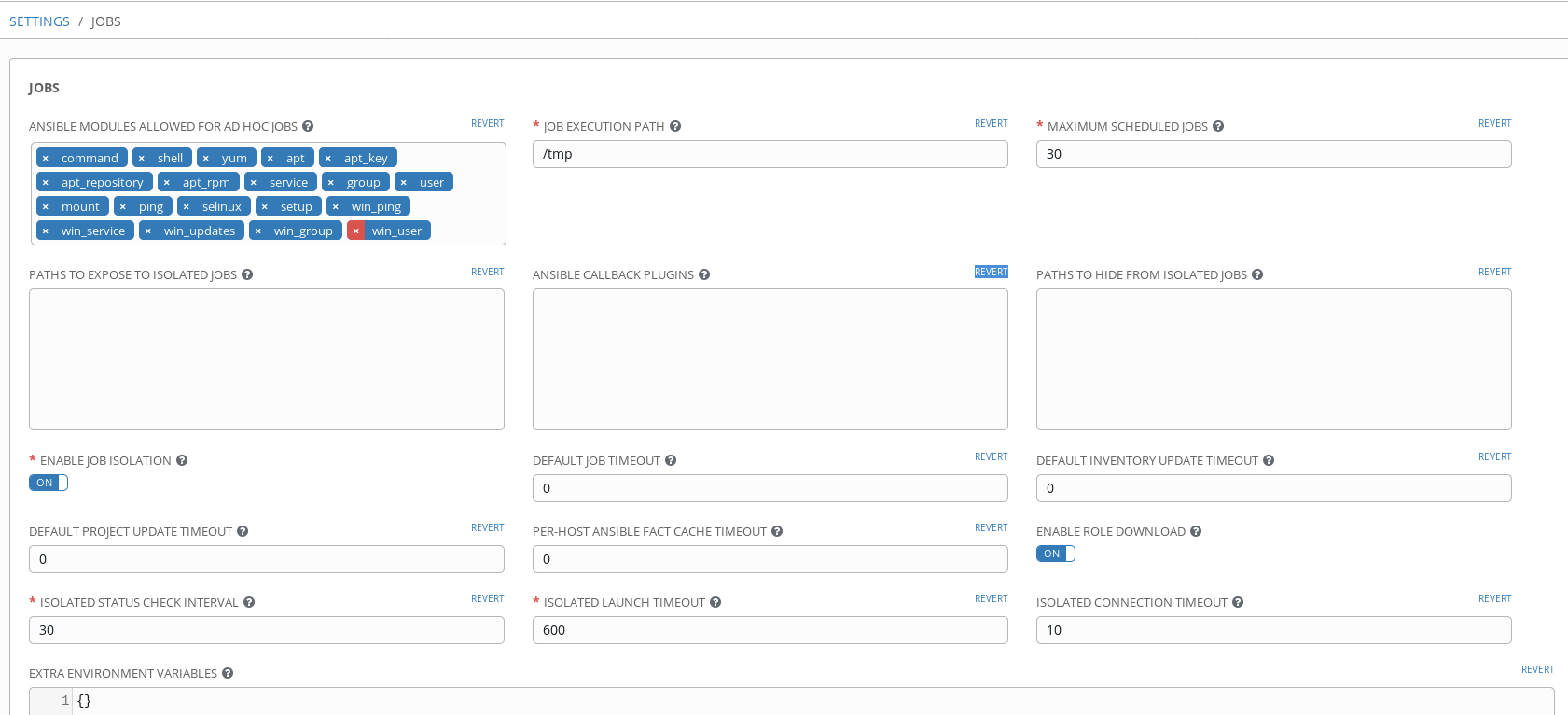
## Authentication:

This is where you will setup your Enterprise auth.



## Jobs:

Here we can add ad-hoc modules to be run, change Max sched jobs, check, and timeout intervals.



## System:

Here is where you will be able to setup your logging and some other minor settings

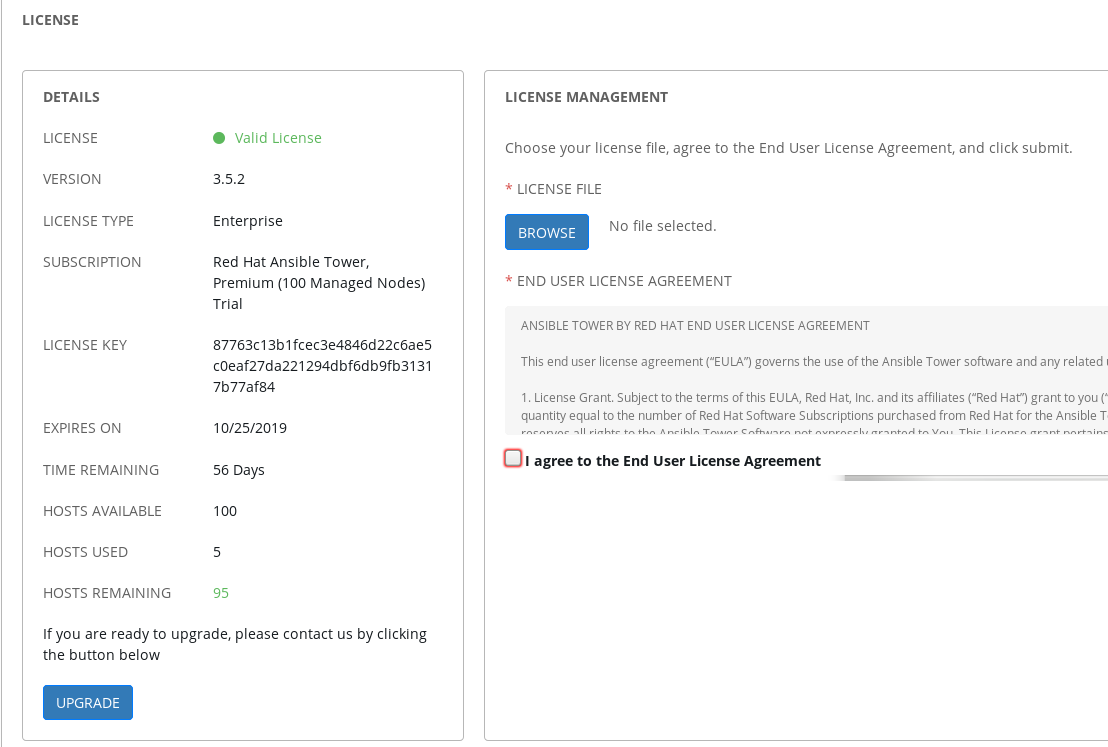
### 

## User Interface:

Here you can set the motd msg and add your logo to the login screen

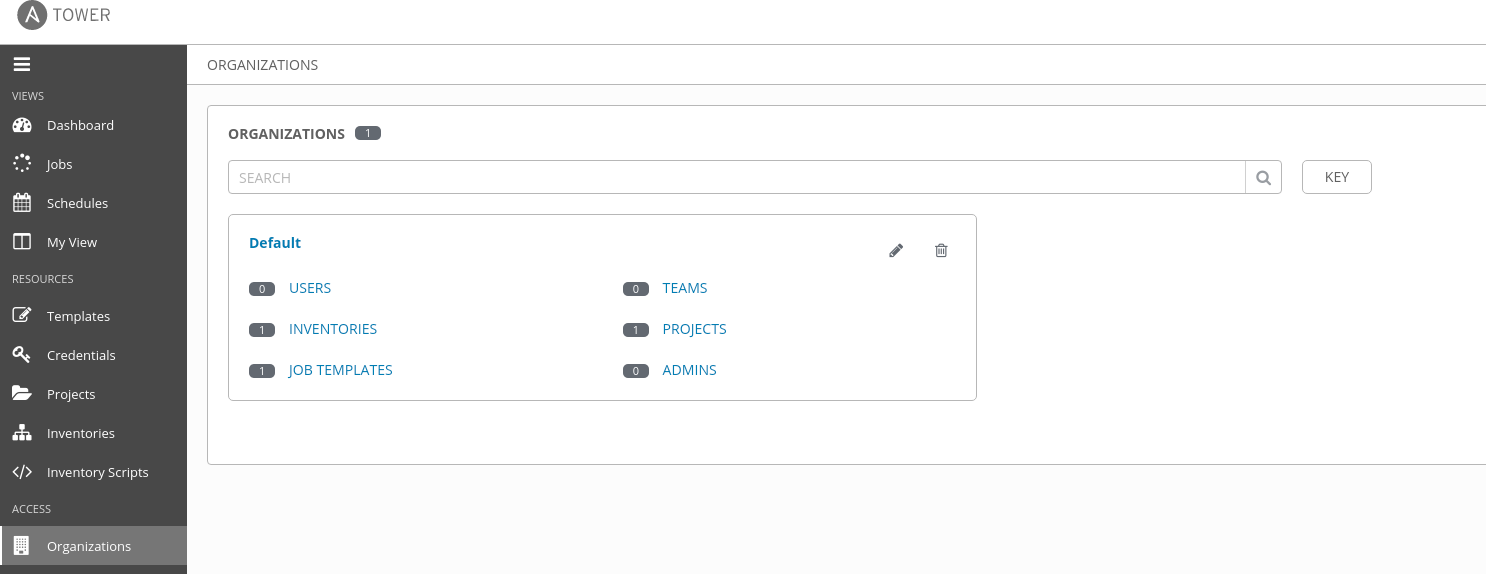
## License

Here you can see the status of your license and update/add a license at any time

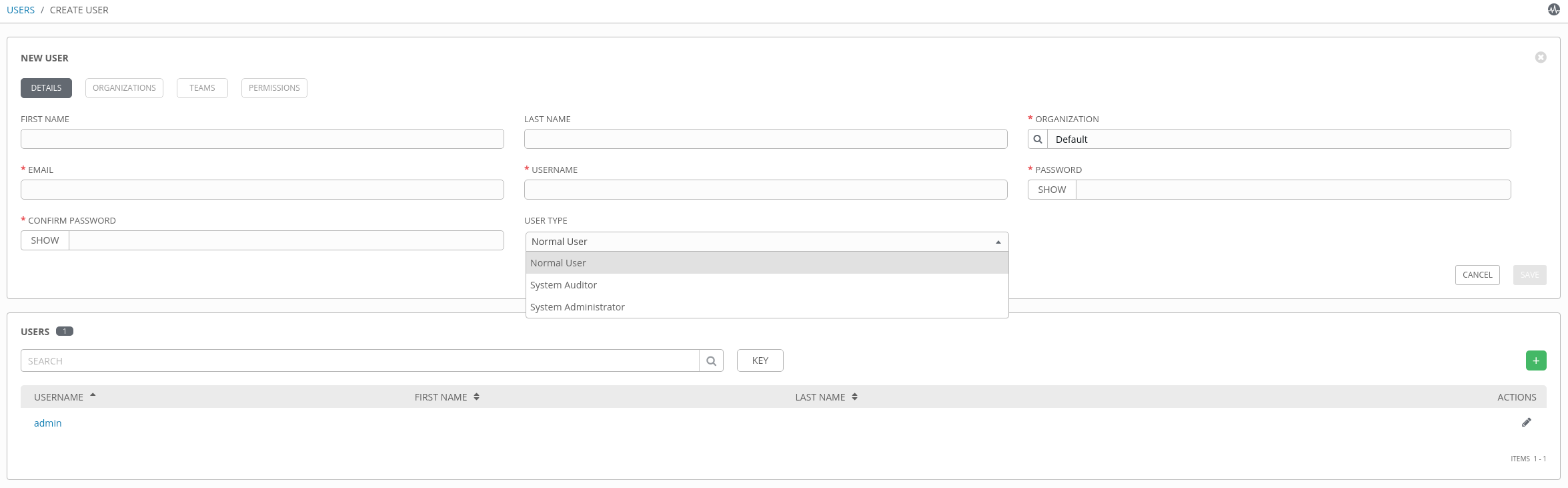


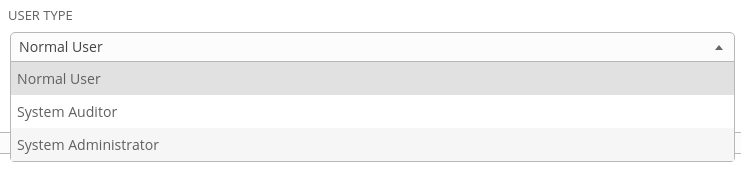
# ORGANIZATIONS

[https://<student>.<yourcompanyhere>.rhdemo.io/#/organizations](https://student1.bpx-poc.rhdemo.io/" \l "/organizations)

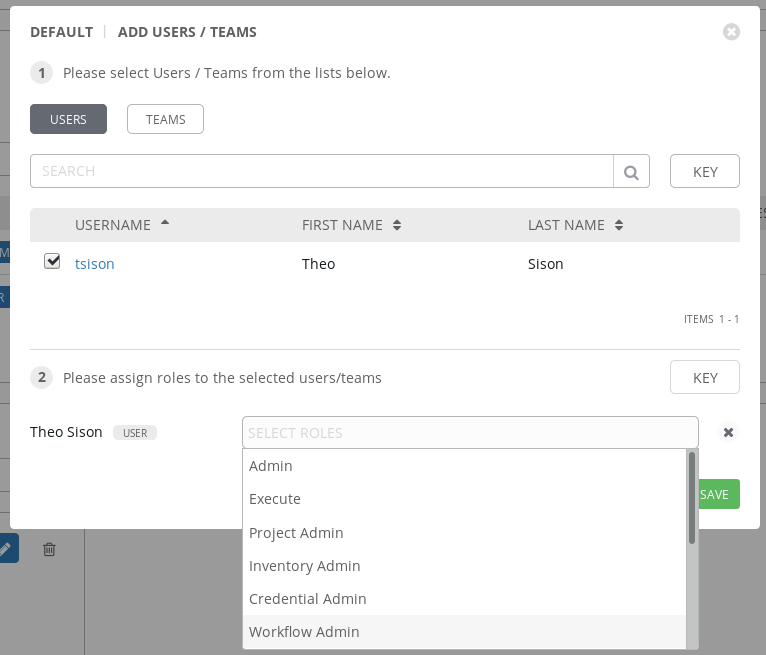
If you are using LDAP or if you are adding users manually, This is where you will create your groups of users organized into Teams or individual users within each org you create. Users can live in multiple Organizations or Teams.

## Users

https://<student>.<yourcompanyhere>.rhdemo.io/#/users/add

There are only 3 types of users on a system Normal, Auditor, and Admin. Only use admin for the few users that have NO RESTRICTIONS in your environment, and you want these people to seeor change everything

If you have selected the "Normal User" you can now assign the level of access that they have within the system by selecting the ORGANIZATION → PERMISSIONS → USER and select the level of authority the user will have. You may select multiple options for your users.



Select "SAVE"

# CREDENTIALS

[https://<student>.<yourcompanyhere>.rhdemo.io/#/credentials](https://student1.bpx-poc.rhdemo.io/" \l "/credentials)

For setting up and working with AWS we need 2 types of credentials

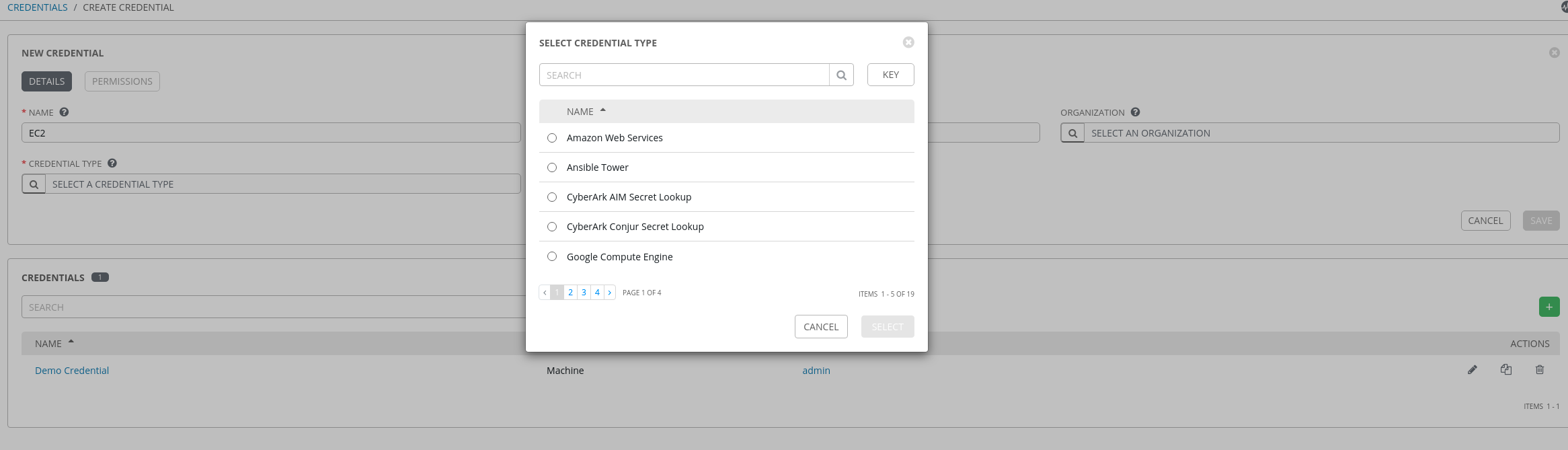
We need the credential for connection to AWS and we need a credential for working with the nodes

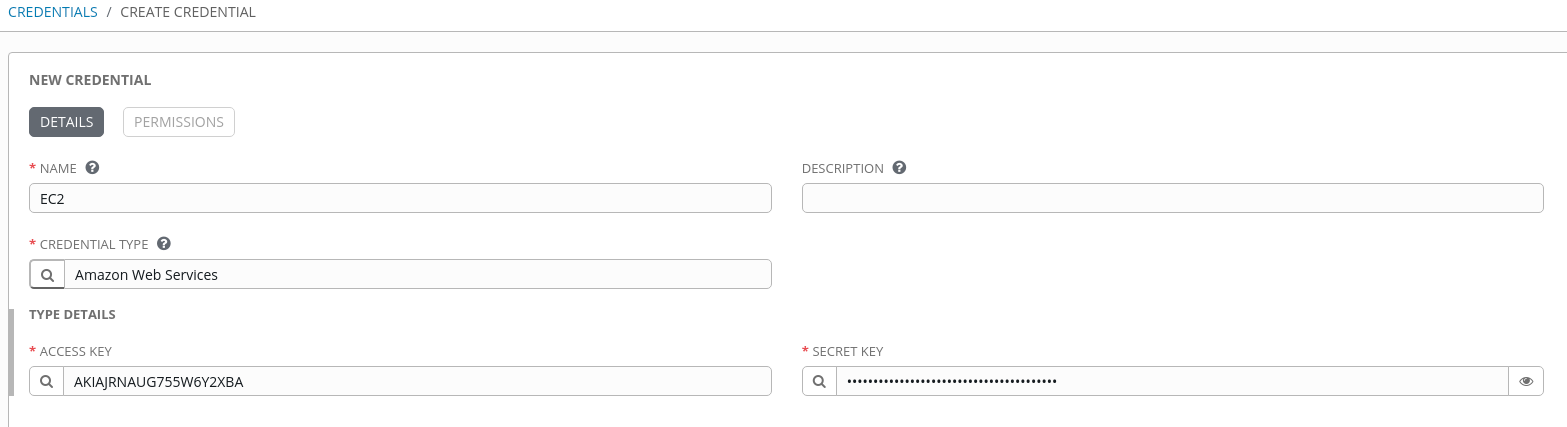
For AWS we need

* ACCESS KEY
* SECRETKEY
* PEM KEY

Select CREDENTIALS from the side menu

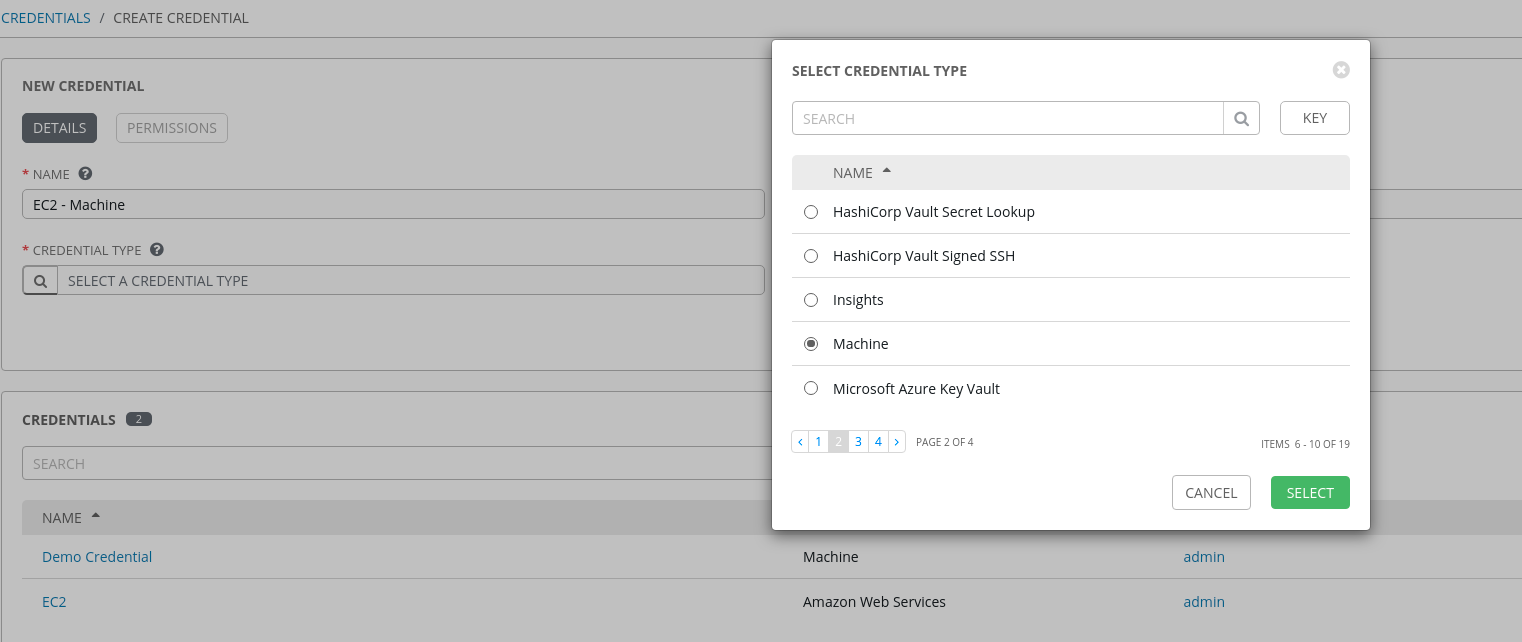
* Click the add button, which launches the CREATE CREDENTIAL



* In the CREDENTIAL TYPE choose "Amazon Web Services"
* Now insert your ACCESS KEY and SECRETKEY
* Select SAVE

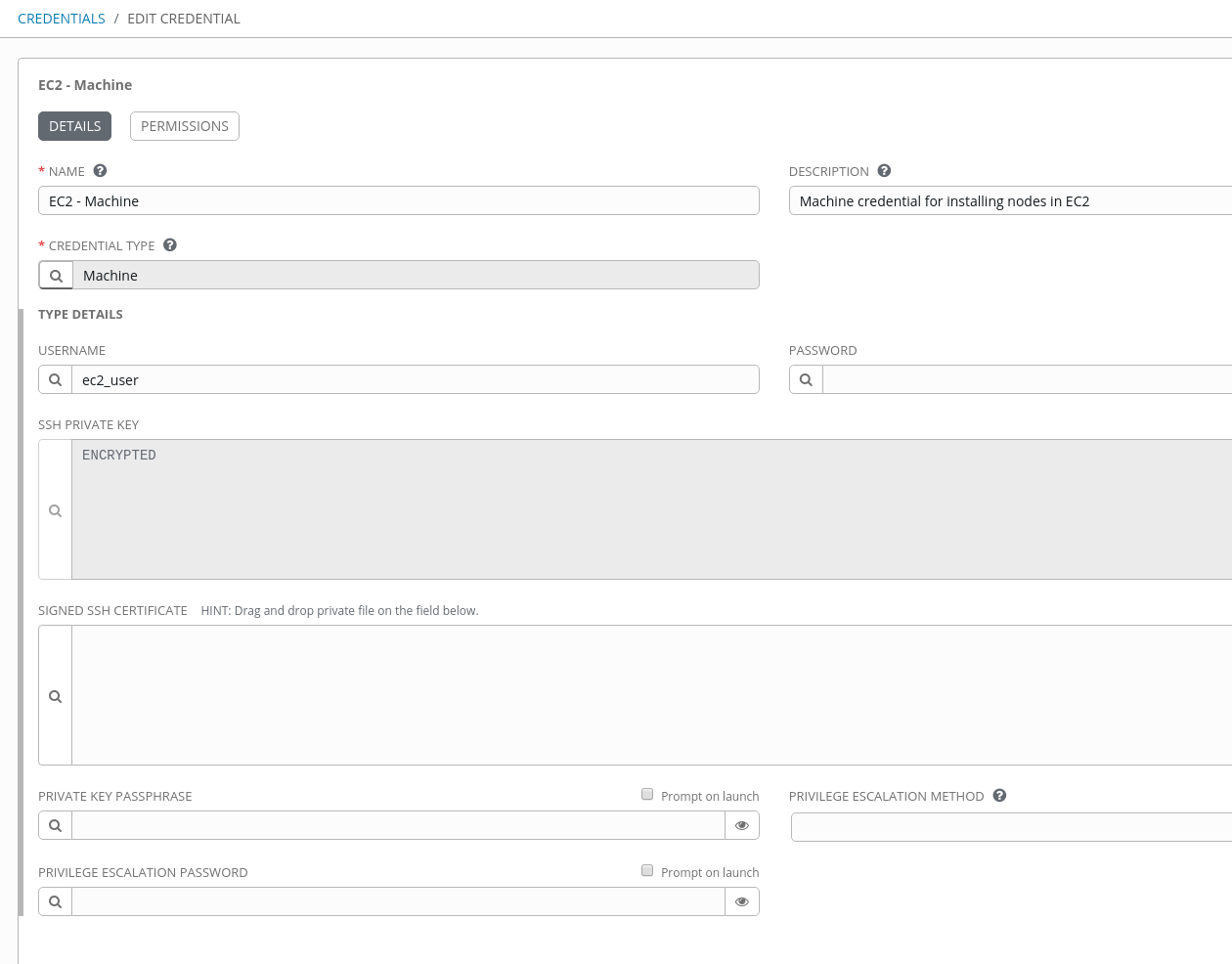
Now for the machine credential

* Click the add button, which launches the CREATE CREDENTIALS.
* In the CREDENTIAL TYPE choose "Machine"



NOTE: The root user for installing in AWS is ec2\_user so add that to the "USERNAME" section

* Now open your PEM key you received from AWS and copy and paste it in the "SSH PRIVATE KEY" section.



# Select SAVE DYNAMIC INVENTORIES

[https://<student>.<yourcompanyhere>.rhdemo.io/#/inventories](https://student1.bpx-poc.rhdemo.io/" \l "/inventories?inventory_search=page_size:20;order_by:name)

Ansible inventory tends to fluctuate over time, with hosts spinning up and shutting down in response to business demands, the static inventory solutions described in Working with Inventory will not serve your needs. You may need to track hosts from multiple sources: cloud providers, LDAP, Cobbler, and/or enterprise CMDB systems.

Ansible integrates all of these options via a dynamic external inventory system. Ansible supports two ways to connect with external inventory: Inventory Plugins and inventory scripts.

Inventory plugins take advantage of the most recent updates to Ansible’s core code. We recommend plugins over scripts for dynamic inventory. You can write your own plugin to connect to additional dynamic inventory sources.

You can still use inventory scripts if you choose. When we implemented inventory plugins, we ensured backwards compatibility via the script inventory plugin. The examples below illustrate how to use inventory scripts.

If you’d like a GUI for handling dynamic inventory, the Red Hat Ansible Tower inventory database syncs with all your dynamic inventory sources, provides web and REST access to the results, and offers a graphical inventory editor. With a database record of all of your hosts, you can correlate past event history and see which hosts have had failures on their last playbook runs.

We will be working with a couple different Dynamic inventories in this example (AWS and ServiceNow) We will start with the AWS component and address the ServiceNow portion in the ServiceNow section.

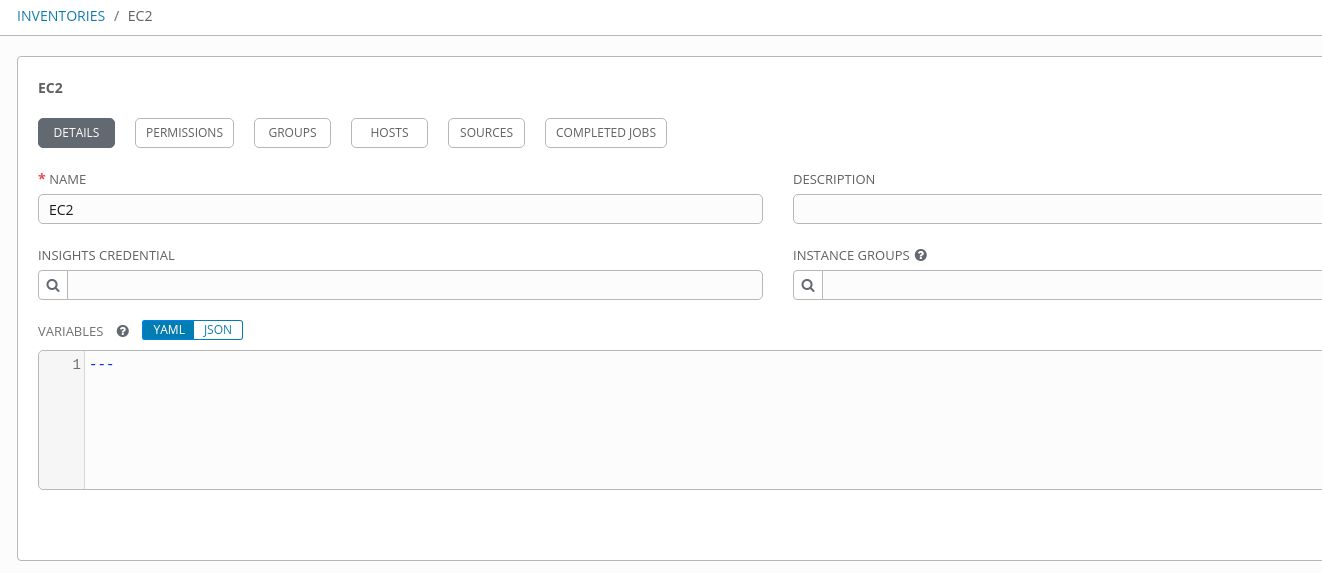
## **AWS** Inventory

From the menu select "Inventories"

Click the add button, which launches the "Inventory" from the drop down

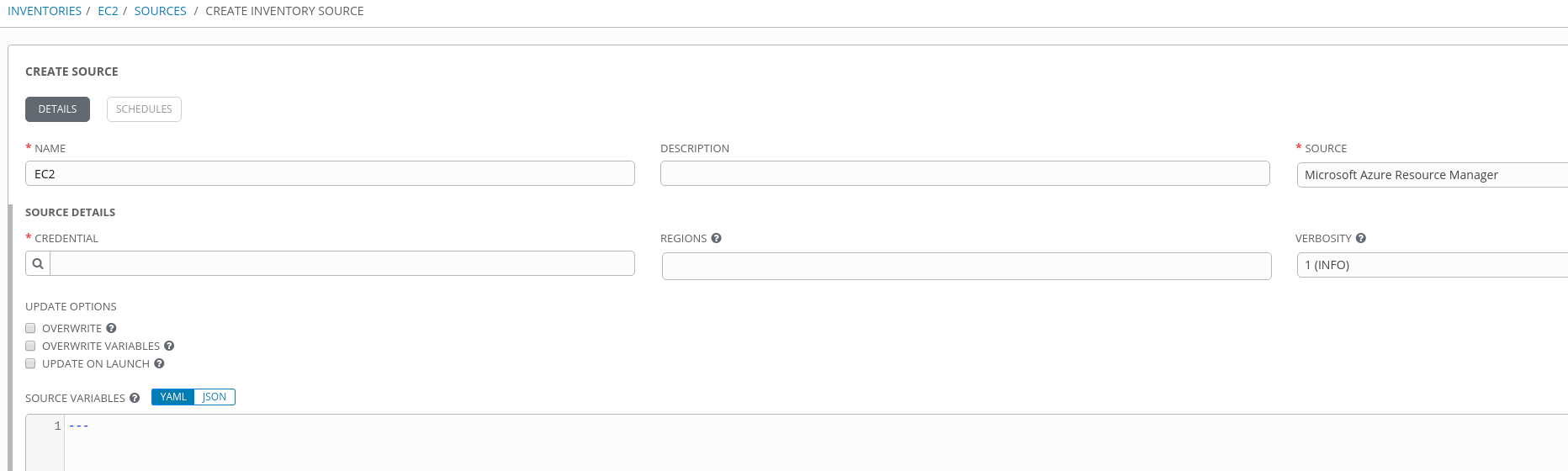
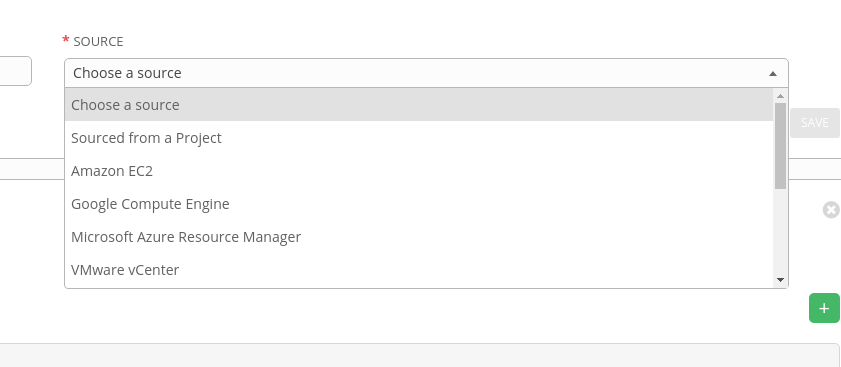
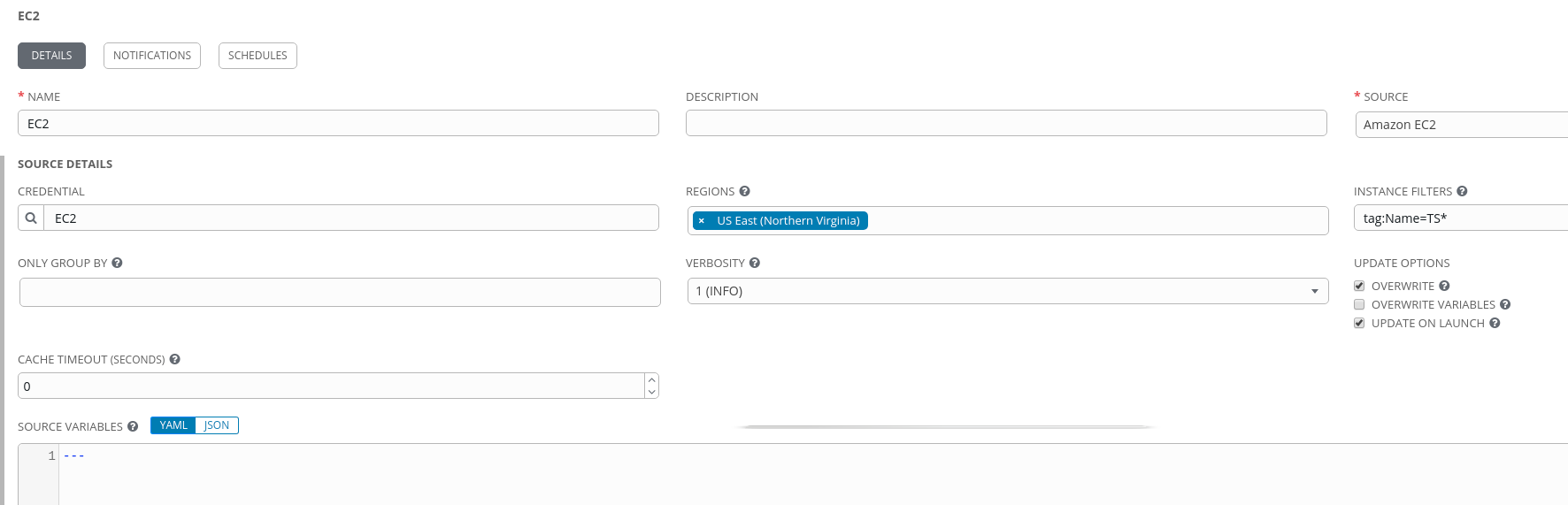
Insert the name of the Inventory (EC2)

Select SAVE



* Then choose the "SOURCES" buttonSOURCE

[https://<student>.<yourcompanyhere>.rhdemo.io/#/inventories/inventory/2/inventory\_sources/add?](https://student1.bpx-poc.rhdemo.io/" \l "/inventories/inventory/2/inventory_sources/add?inventory_search=page_size:20;order_by:name)

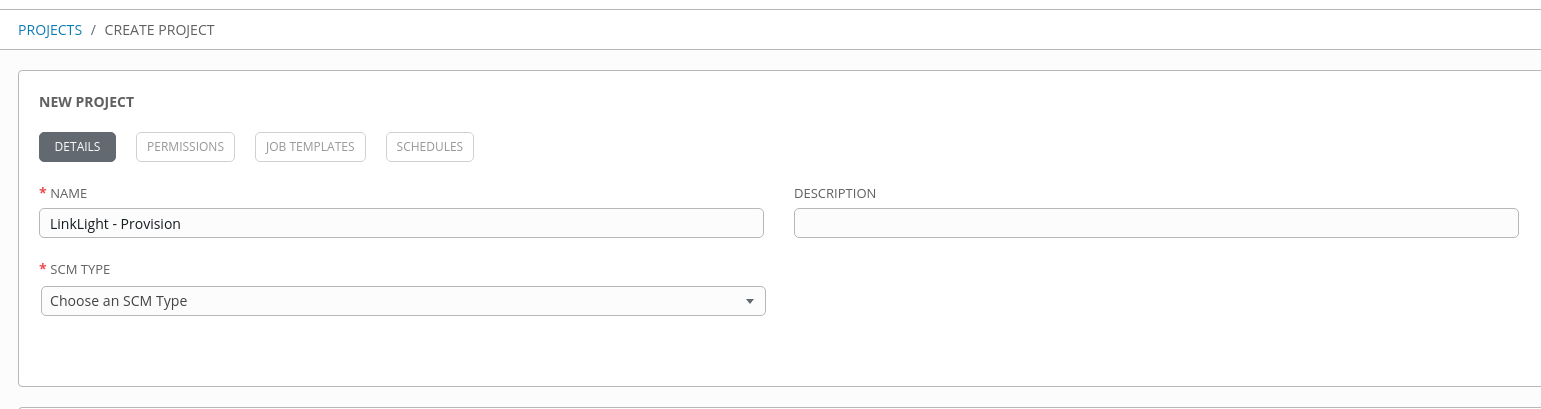
* Enter the name (I chose EC2 for consistency)
  + 
* Choose the "SOURCE" "Amazon EC2"
* Choose "CREDENTIAL" (EC2) Choose "REGIONS"
* (extra) add ‘INSTANCE FILTERS" This will allow you to filter for items you want to see. (tagNametag:Name=<yourcompanyhere>\*) this example filters all instances that were prefaced by "<yourcompanyhere>"
* Choose the boxes "OVERWRITE" and "UPDATE ON LAUNCH"
* Choose "SAVE"

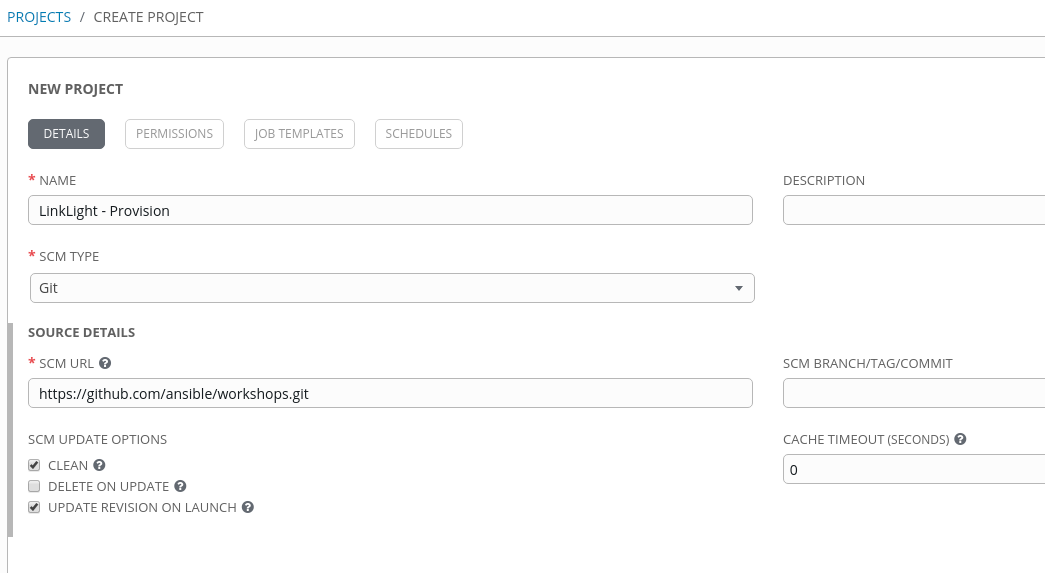
# PROJECTS

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

You 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, Mercurial, and Red Hat Insights. To create a Red Hat Insights project, refer to Setting up an Insights Project.

[https://<student>.<yourcompanyhere>.rhdemo.io/#/projects](https://student1.bpx-poc.rhdemo.io/" \l "/projects)

* Click the add button, which launches the Create Project dialog.
* Name the PROJECT (LinkLight - Linux)
* Choose the SCM TYPE (Git)
* Add your "SCM URL" (https://github.com/ansible/workshops.git)
* Choose the boxes "CLEAN" and "UPDATE REVISION ON LAUNCH"
* Choose SAVE

Repeat the process for creating a project and create a "SkyLight – Windows" project using [https://github.com/mgmt-sa-tiger-team/skylight.git](https://github.com/mgmt-sa-tiger-team/skylight.git) as the SCM URL.

# CREDENTIAL TYPES

[https://docs.ansible.com/ansible-tower/latest/html/userguide/credential\_types.html#getting-started-with-credential-types](https://docs.ansible.com/ansible-tower/latest/html/userguide/credential_types.html" \l "getting-started-with-credential-types)

As a Tower administrator with superuser access, you can define a custom credential type in a standard format using a YAML/JSON-like definition, allowing the assignment of new credential types to jobs and inventory updates. This allows you to define a custom credential type that works in ways similar to existing credential types. For example, you could create a custom credential type that injects an API token for a third-party web service into an environment variable, which your playbook or custom inventory script could consume.

Custom credentials support the following ways of injecting their authentication information:

* Environment variables
* Ansible extra variables
* File-based templating (i.e., generating .ini or .conf files that contain credential values)

You can attach one SSH and multiple cloud credentials to a Job Template. Each cloud credential must be of a different type. In other words, only one AWS credential, one GCE credential, etc., are allowed. In Ansible Tower 3.2 and later, vault credentials and machine credentials are separate entities.

In the provided example we will need to create 2 of our own credentials. The first for deploying Windows in SkyLight and the second for ServiceNow

[https://<student>.<yourcompanyhere>.rhdemo.io/#/credential\_types](https://student1.bpx-poc.rhdemo.io/" \l "/credential_types)

From the menu on the left select "Credential Types"

* Click the add button, which launches the NEW CREDENTIAL TYPE
* Name the CREDENTIAL TYPE
* In the field marked INPUT CONFIGURATION cut and paste

*fields:*

*- id: tower\_license*

*type: string*

*label: Tower License*

*required:*

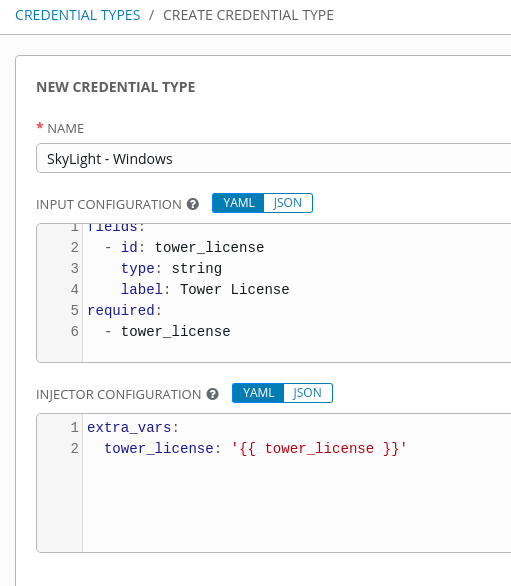
*- tower\_license*

* In the field marked INJECTOR CONFIGURATION cut and paste

*extra\_vars:*

*tower\_license: '{{ tower\_license }}'*

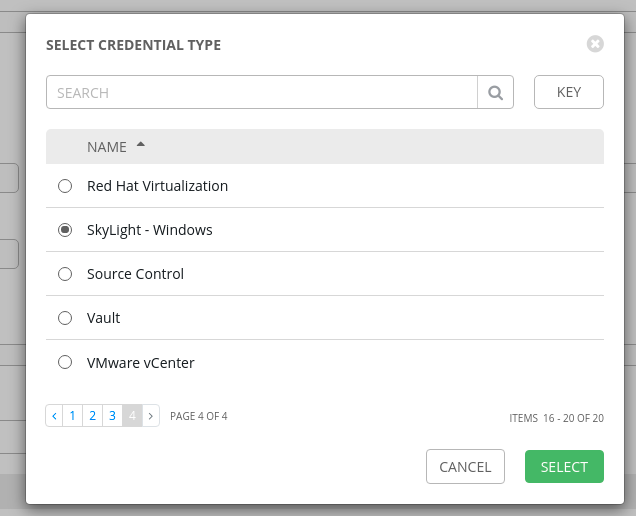
Example below:

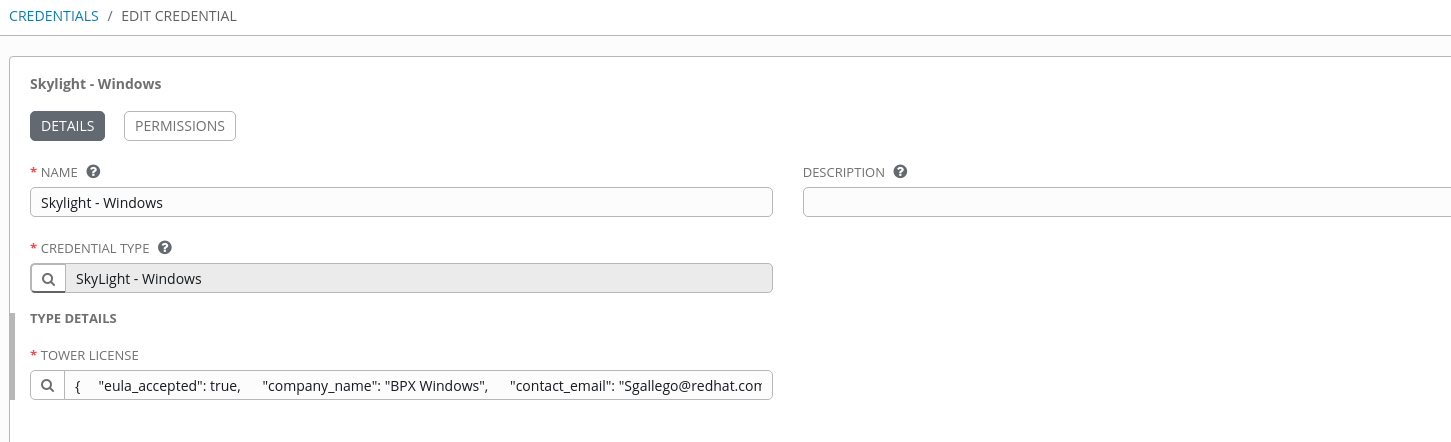
* Choose "SAVE"

### Create A Credential For Skylight – Windows

[https://<student>.<yourcompanyhere>.rhdemo.io/#/credentials](https://student1.bpx-poc.rhdemo.io/" \l "/credentials)

Open the 50 node license in a text editor

* Above the "company\_name" insert ""eula\_accepted": true," make it look like the graphic below
* Highlight and copy the text.
* Select "Credentials" from the menu
* Click the add button, which launches the NEW CREDENTIAL
* Name the Credential (Skylight – Windows)
* Select the Credential Type that we just created (Skylight - Windows)
* Paste the license into the TOWER LICENSE field



* Choose "SAVE"

What this has done is, it has created an example of a way you can push a license to a system as it is built .

# TEMPLATES

A [job template](http://docs.ansible.com/ansible-tower/3.5.2/html/installandreference/glossary.html" \l "term-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.

The () menu opens a list of the job templates that are currently available. The default view is collapsed (Compact), showing the template name, template type, and the statuses of the jobs that ran using that template, but you can click Expanded to view more information. This list is sorted alphabetically by name, but you can sort by other criteria, or search by various fields and attributes of a template.

[https://<student>.<yourcompanyhere>.rhdemo.io/#/templates](https://student1.bpx-poc.rhdemo.io/" \l "/templates)

Now for each of the environments that are built there needs to be a way to build them when needed and to tear them down when you are done, so the following templates must be created.

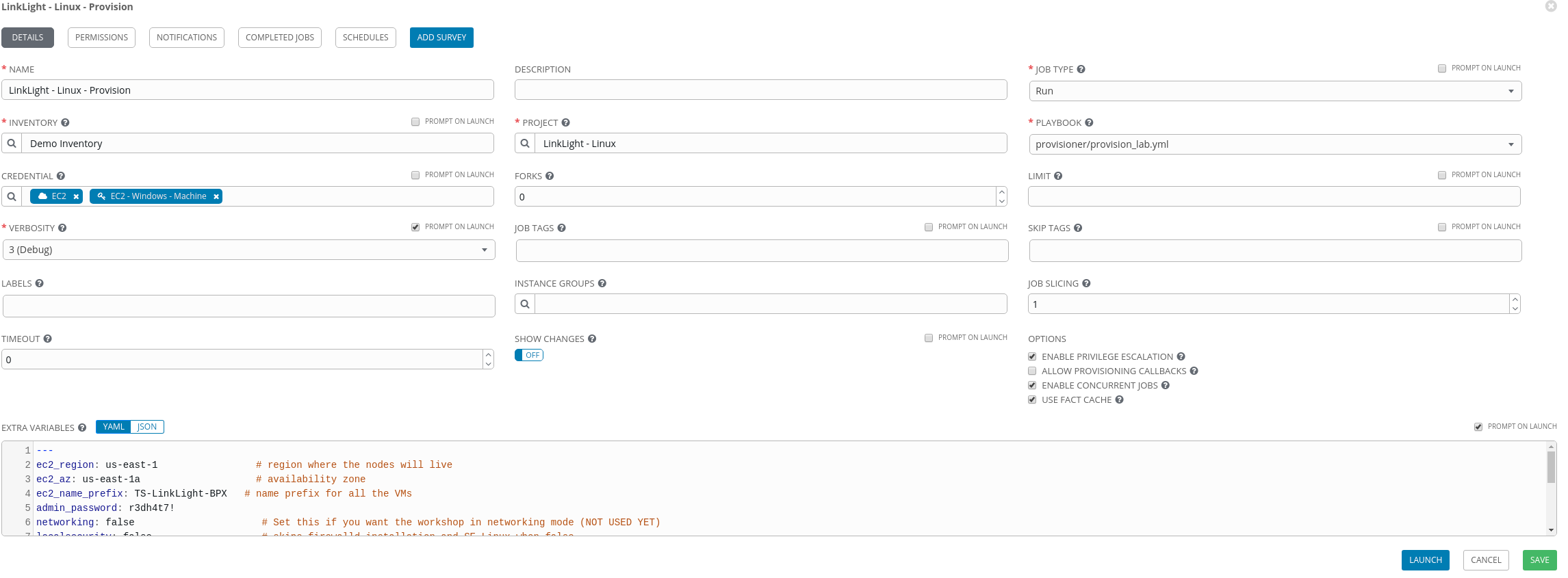
1. LinkLight – Linux – Provision
2. SkyLight – Windows – Provision
3. LinkLight – Linux – Teardown
4. SkyLight - Windows – Teardown

* From the menu on the left select "Templates"
* Click the add button, Chose "Job Template"
* Follow the tables below to complete each template.

NOTE: LinkLight and SkyLight playbook examples are designed to install groups of 4 servers for Linux and 7 system for Windows for each student. Be sure to check the box Prompt on launch so you may change the “EXTRA VARIABLES” you need for creating each node set. By default the vars that are usually changed are “ec2\_name\_prefix: <yourcompanyhere>-LinkLight-Test“ and “student\_total: 1“ you can set these up as SURVEY questions if you like. Try running with the default first and then look in the inventory to see what has been created. You will need to use the same <yourcompanyhere>-LinkLight-Test in your tear down template

### PROVISION

|  |  |
| --- | --- |
| **FIELD** | **ENTRY** |
| NAME: | LinkLight – Linux – Provision |
| DESCRIPTION: | Provision Linux Environments |
| JOB TYPE: | Run |
| INVENTORY: | Demo Inventory |
| PROJECT: | LinkLight |
| PLAYBOOK: | provisioner/provision\_lab.yml |
| CREDENTIAL(s): | EC2, EC2\_Machine |
| ENABLE PRIVILEGE ESCALATION: | X |
| ENABLE CONCURRENT JOBS: | X |
| USE FACT CACHE: | X |
| EXTRA VARIABLES: | ---  ec2\_region: us-east-1 # region where the nodes will live  ec2\_az: us-east-1a # availability zone  ec2\_name\_prefix: <yourcompanyhere>-LinkLight-Test # name prefix for all the VMs  admin\_password: r3dh4t7!  localsecurity: false # skips firewalld installation and SE Linux when false  student\_total: 1 # automatically creates students if you don’t define a user.yml  create\_login\_page: true  towerinstall: true # If you want to install an HTML landing page  f5workshop: false  xrdp: true  workshop\_type: rhel |

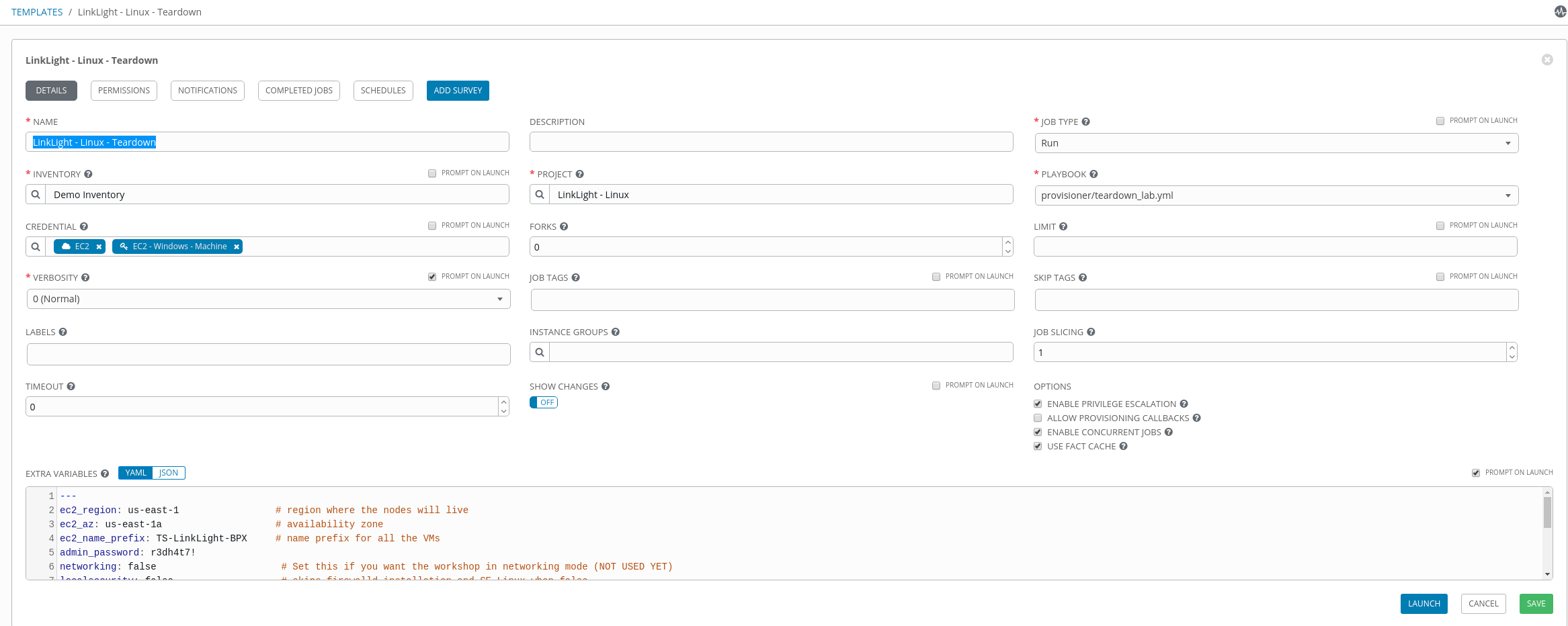


|  |  |
| --- | --- |
| **FIELD** | **ENTRY** |
| NAME: | SkyLight – Windows – Provision |
| DESCRIPTION: | Provision Windows Environments |
| JOB TYPE: | Run |
| INVENTORY: | Demo Inventory |
| PROJECT: | SkyLight |
| PLAYBOOK: | provision.yml |
| CREDENTIAL(s): | EC2, EC2\_Machine, SkyLight |
| ENABLE PRIVILEGE ESCALATION: |  |
| ENABLE CONCURRENT JOBS: | X |
| USE FACT CACHE: | X |
| EXTRA VARIABLES: | ---  dns\_domain\_name: ansibleworkshop.com  domain\_admin\_password: MyP@ssw0rd21  ec2\_region: us-east-1  name\_prefix: <yourcompanyhere>-Skylight-TestHS  root\_user: ec2-user  user\_count: 1  user\_prefix: student  users\_password: AnsibleWorkshop21# |

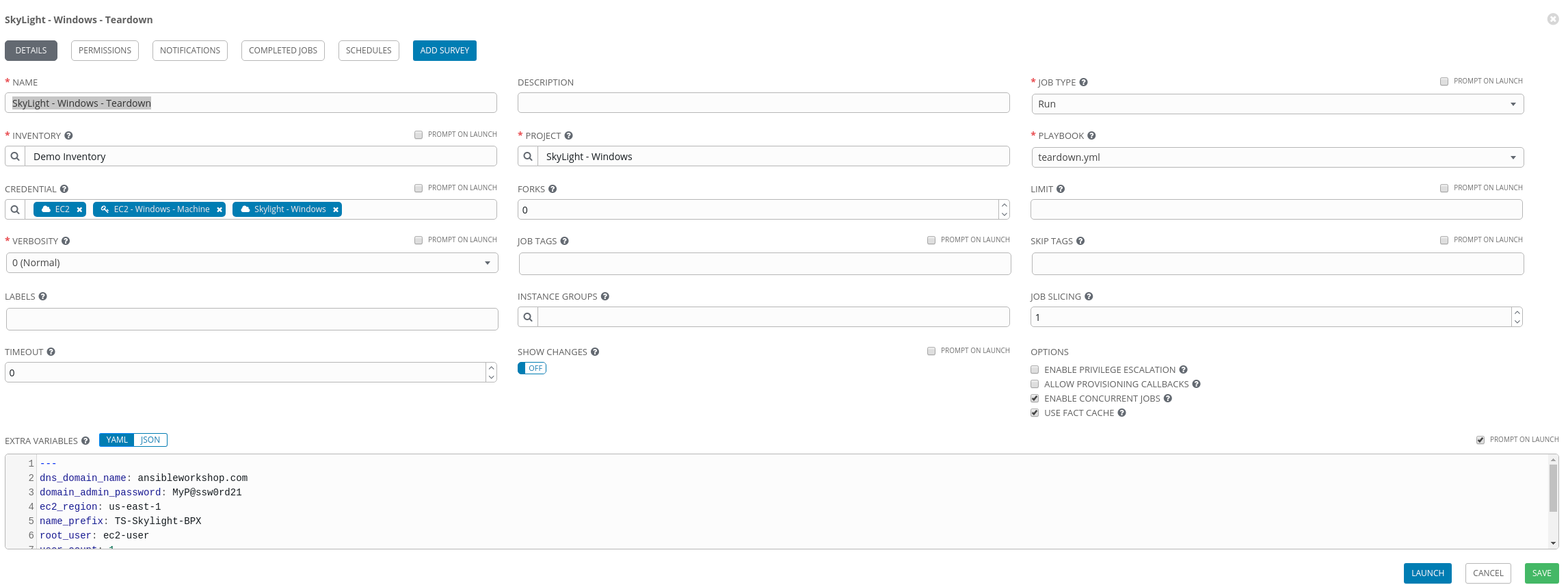
### 

### TEARDOWN

|  |  |
| --- | --- |
| **FIELD** | **ENTRY** |
| NAME: | LinkLight – Linux – Teardown |
| DESCRIPTION: | Destroy Linux Environment |
| JOB TYPE: | RUN |
| INVENTORY: | Demo Inventory |
| PROJECT: | LinkLight |
| PLAYBOOK: | provisioner/teardown.yml |
| CREDENTIAL(s): | EC2, EC2\_Machine |
| ENABLE PRIVILEGE ESCALATION: | X |
| ENABLE CONCURRENT JOBS: | X |
| USE FACT CACHE: | X |
| EXTRA VARIABLES: | ---  ec2\_region: us-east-1 # region where the nodes will live  ec2\_az: us-east-1a # availability zone  ec2\_name\_prefix: <yourcompanyhere>-LinkLight-<yourcompanyhere> # name prefix for all the VMs  admin\_password: r3dh4t7!  localsecurity: false # skips firewalld installation and SE Linux when false  student\_total: 1 # automatically creates students if you don’t define a user.yml  create\_login\_page: true  towerinstall: true  f5workshop: false  xrdp: true  workshop\_type: rhel |



|  |  |
| --- | --- |
| **FIELD** | **ENTRY** |
| NAME: | SkyLight - Windows – Teardown |
| DESCRIPTION: | Destroy Windows Environment |
| JOB TYPE: | RUN |
| INVENTORY: | SkyLight |
| PROJECT: | SkyLight |
| PLAYBOOK: | teardown.yml |
| CREDENTIAL(s): | EC2, EC2\_Machine, SkyLight |
| ENABLE PRIVILEGE ESCALATION: |  |
| ENABLE CONCURRENT JOBS: | X |
| USE FACT CACHE: | X |
| EXTRA VARIABLES: | ---  dns\_domain\_name: ansibleworkshop.com  domain\_admin\_password: MyP@ssw0rd21  ec2\_region: us-east-1  name\_prefix: <yourcompanyhere>-Skylight-Test  root\_user: ec2-user  user\_count: 1  user\_prefix: student  users\_password: AnsibleWorkshop21# |



# SERVICENOW

Obtain a developer ServiceNow account

[https://developer.servicenow.com/app.do#!/home](https://developer.servicenow.com/app.do" \l "!/home)

## ServiceNow Dynamic Inventory

We are going to set up this dynamic inventory for ServiceNow but the following method can be used with any CMDB.

The custom inventory script has already been made and can be available here.

<https://raw.githubusercontent.com/ServiceNowITOM/ansible-sn-inventory/master/now.py>

From the main menu select "Inventory Scripts"

* Click the add button, which launches the "NEW CUSTOM INVENTORY**"**
* Name the script "ServiceNow"
* Copy and paste the raw now.py into the "CUSTOM SCRIPT" field.
* Choose "SAVE"
* Now pass a credential to connect to ServiceNow the following 3 environment variables are needed.

SN\_INSTANCE  
SN\_USERNAME  
SN\_PASSWORD

Now we will create a custom credential type so the credentials will be encrypted.

From the main menu select "Credential Types"

* Click the add button, which launches the "NEW CREDENTIAL TYPE"
* Name the credential type "ServiceNow"
* Now enter the following into the "INPUT CONFIGURATION"

*fields:*

*- id: username*

*type: string*

*label: Username*

*- id: password*

*type: string*

*label: Password*

*secret: true*

*- id: instance*

*type: string*

*label: Instance*

NOTE: if you want the password to be encrypted and not stored in plain text makesure you use the " *- secret: true"*

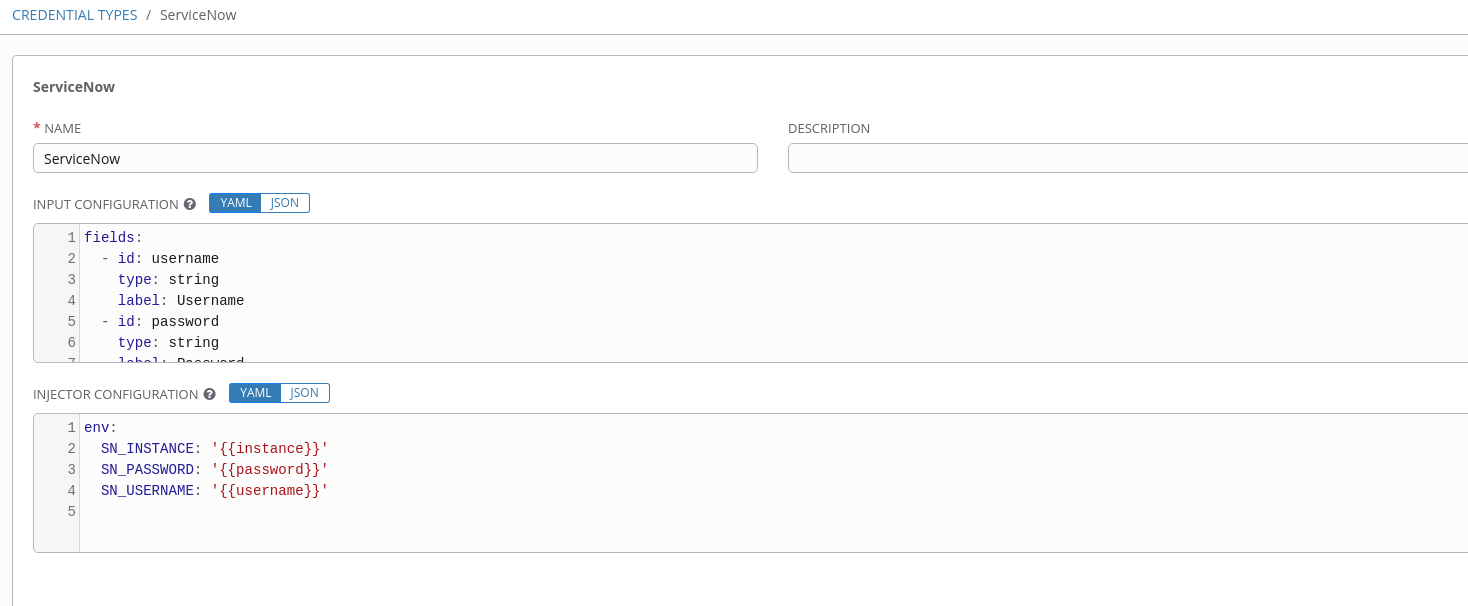
* Now enter the following into the "Injector Configuration" we will add in the relevant ServiceNow environment tags.

*env:*

*SN\_INSTANCE: '{{instance}}'*

*SN\_PASSWORD: '{{password}}'*

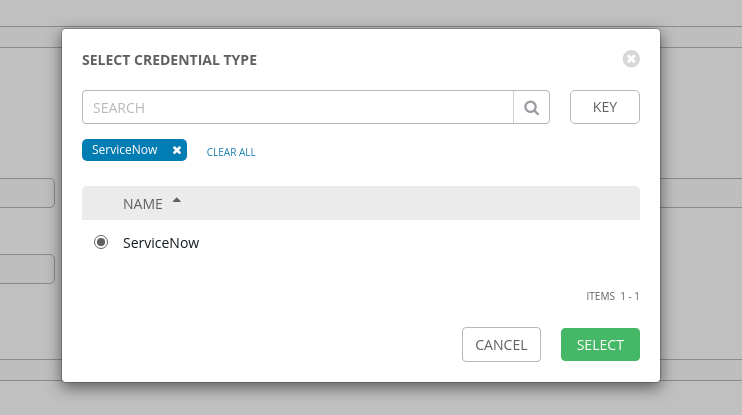
*SN\_USERNAME: '{{username}}'*

* Choose "SAVE"

Now create a credential for ServiceNow, navigate to credentials

From the main menu select "Credentials"

* Click the add button, which launches the "NEW CREDENTIAL"
* Name the credential "ServiceNow"
* Select a "CREDENTIAL TYPE" search for "ServiceNow" and select "ServiceNow"



* Enter the "TYPE DETAILS"

USERNAME: admin

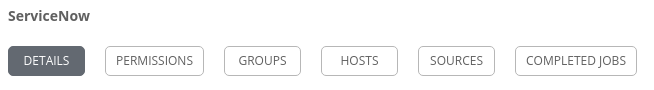
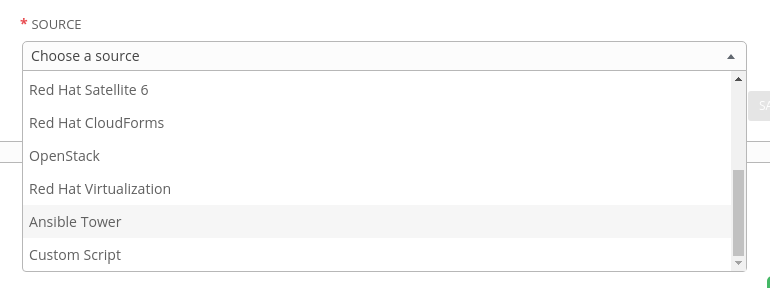
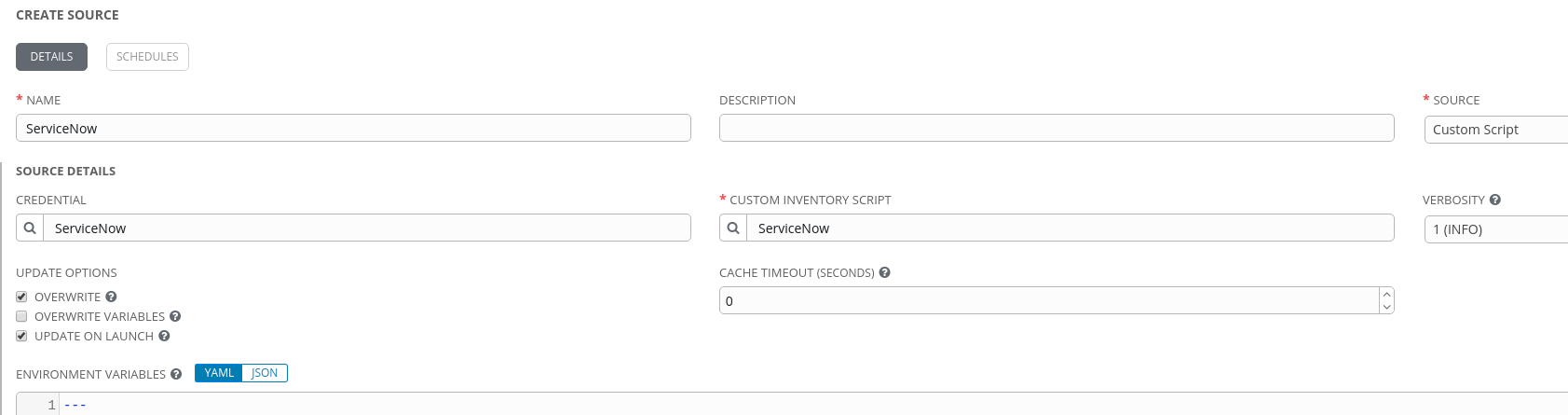
PASSWORD: r3dh4t7!SN

SERVICENOW INSTANCE: <name>.service-now.com/

* Choose "SAVE”

## Create The Inventory

From the main menu select "INVENTORIES"

* Click the add button, which launches the "NEW INVENTORY"
* Name the inventory "ServiceNow"
* Choose "SAVE"
* Now the buttons at the top of the "INVENTORIES / ServiceNow" screen should be selectable
* Choose "SOURCES"
* Click the add button, which launches the "CREATE SOURCE"
* Name the source "ServiceNow"
* Use the drop down "SOURCE" and select "Custom Script"
* Under "CREDENTIAL" choose "ServiceNow"
* Under "CUSTOM INVENTORY SCRIPT" choose "ServiceNow"
* Choose the checkboxes "OVERWRITE" and "UPDATE ON LAUNCH"
* Choose "SAVE"

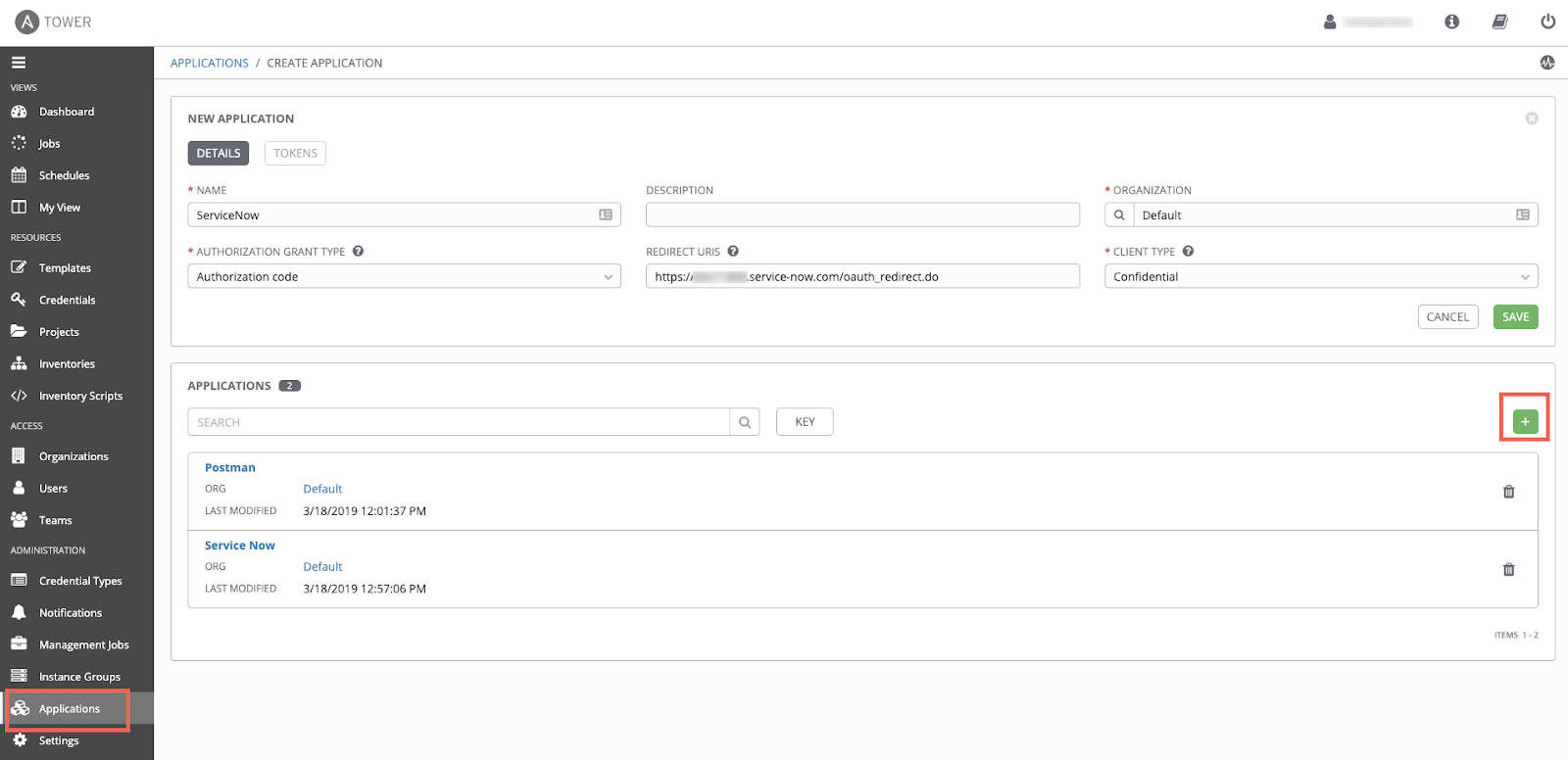
Now you will be able to use ServiceNow as a CMDB to control nodes and actions in Ansible Tower.

# CREATING AN APPLICATION IN ANSIBLE TOWER

USERNAME: admin

PASSWORD: r3dh4t7!SN

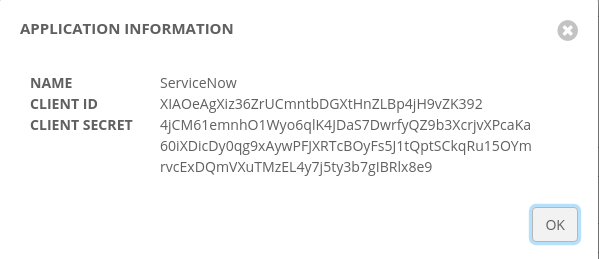
SERVICENOW INSTANCE: <name>[.service-now.com](https://dev90709.service-now.com/oauth_redirect.do)



In Ansible Tower, navigate to Applications on the left side of the screen.

* Click the green plus button, which will present you with a New Application dialog screen. Fill in the following fields:
* Click the add button, which launches the "CREATE APPLICATION"
* Name the source "ServiceNow"
* Use the magnifying glass icon to choose the "ORGANIZATION" that will own the application
* Select the "AUTHORIZATION GRANT TYPE" and when you select the text box a menu will drop down choose "Authorization code"
* Select the "Redirect URIS" [https://<name>.service-now.com/oauth\_redirect.do](https://dev88611.service-now.com/oauth_redirect.do)
* Then to the right under the "CLIENT TYPE" choose Confidential
* Choose "SAVE"

NOTE: A window will pop up, presenting you with the Client ID and Client Secret needed for ServiceNow to make API calls into Tower. This will only be presented ONCE, so capture these values for later use.



Next, navigate to Settings->System on the left side of the screen. You’ll want to toggle the Allow External Users to Create Oauth2 Tokens option to on. Click the green Save button to commit the change.

* Moving over to ServiceNow, Navigate to System Definitions->Certificates. This will take you to a screen of all the certificates ServiceNow uses. Click on the blue New button, and fill in these details:

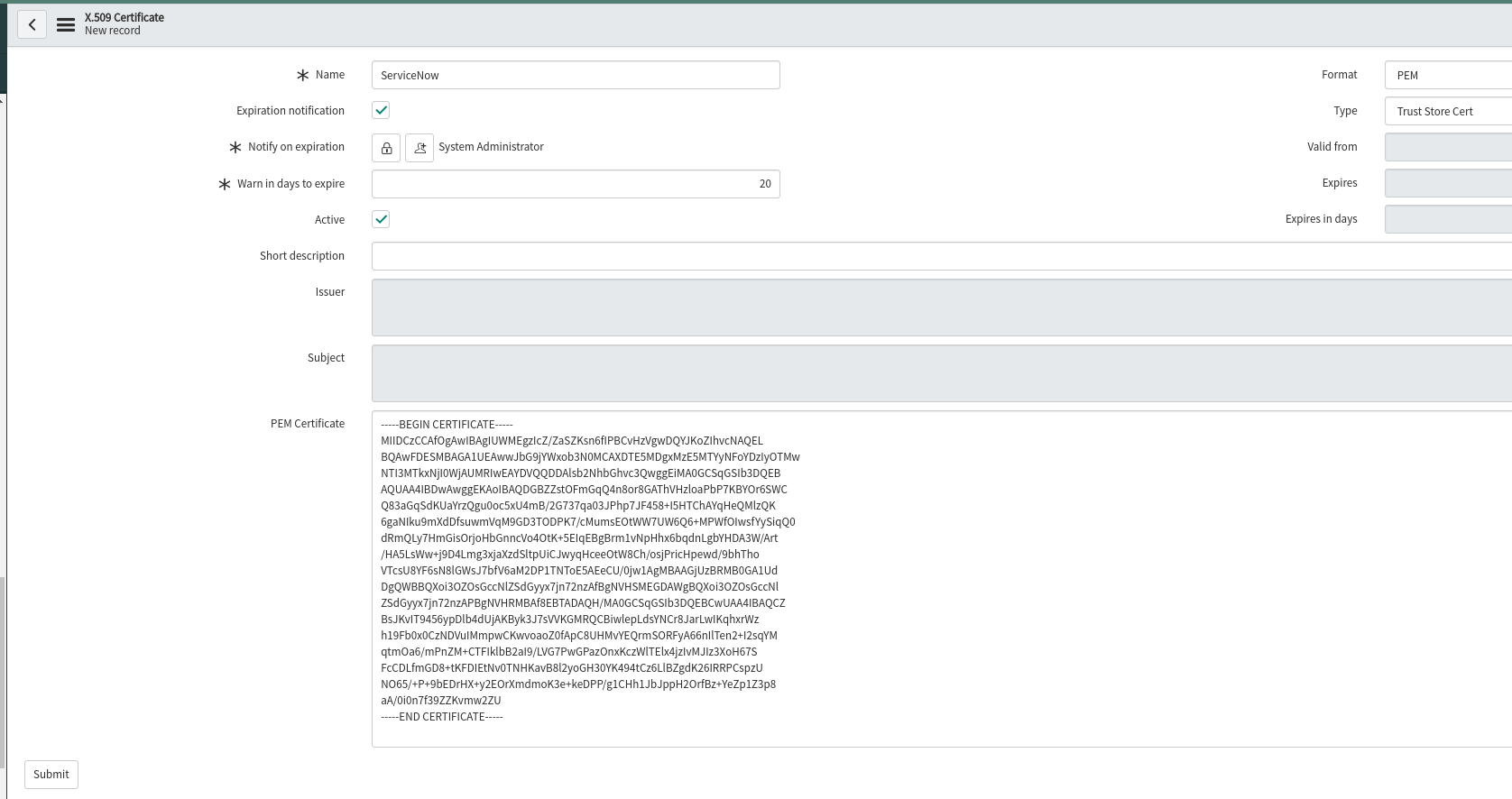
Name: Descriptive name of the certificate

Format: PEM

Type: Trust Store Cert

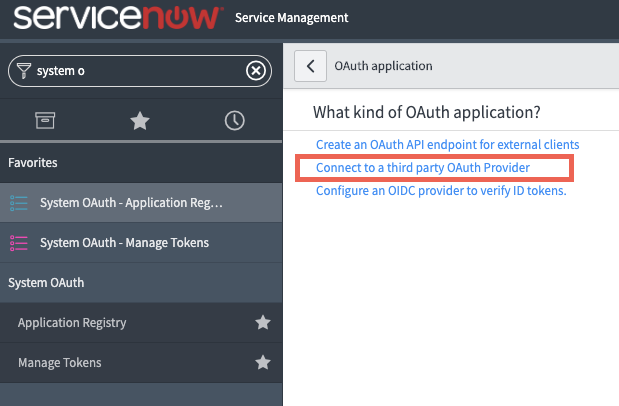
PEM Certificate: The certificate to authenticate against Ansible Tower with. You can use the built-in certificate on your Tower server, located at /etc/tower/tower.cert. Copy the contents of this file into the field in ServiceNow.

* Click the Submit button at the bottom.



In ServiceNow, Navigate to System OAuth->Application Registry. This will take you to a screen of all the Applications ServiceNow communicates with.

* Click on the blue New button, and you will be asked What kind of Oauth application you want to set up.
* Select Connect to a third party Oauth Provider.



* On the new application screen, fill in these details:

Name: Ansible Tower

Client ID: XIAOeAgXiz36ZrUCmntbDGXtHnZLBp4jH9vZK392

Client Secret: 4jCM61emnhO1Wyo6qlK4JDaS7DwrfyQZ9b3XcrjvXPcaKa60iXDicDy0qg9xAywPFJXRTcBOyFs5J1tQptSCkqRu15OYmrvcExDQmVXuTMzEL4y7j5ty3b7gIBRlx8e9

Default Grant Type: Authorization Code

Authorization URL: [https://<student>.<yourcompanyhere>.rhdemo.io/api/o/authorize/](https://student1.bpx-poc.rhdemo.io/api/o/authorize/)

Token URL: [https://<student>.<yourcompanyhere>.rhdemo.io/api/o/token/](https://student1.bpx-poc.rhdemo.io/api/o/token/)

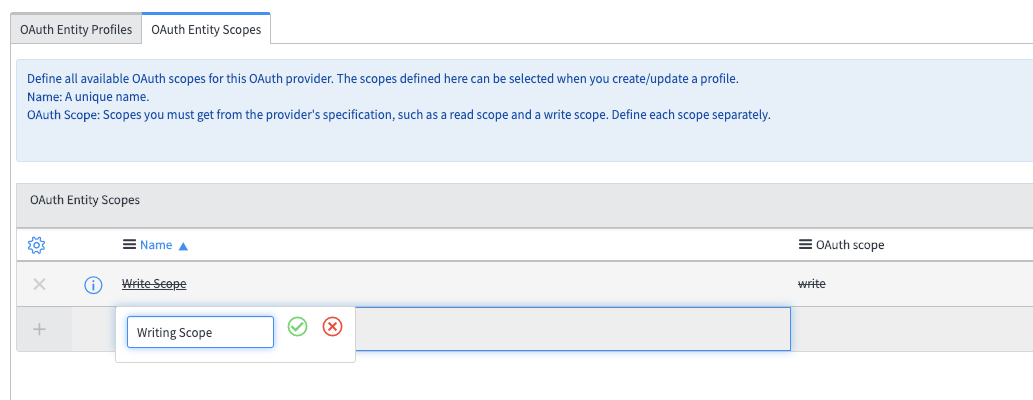
Redirect URL: [https://https://<name>.service-now.com//oauth\_redirect.do](about:blank)

* Click the Submit button at the bottom.



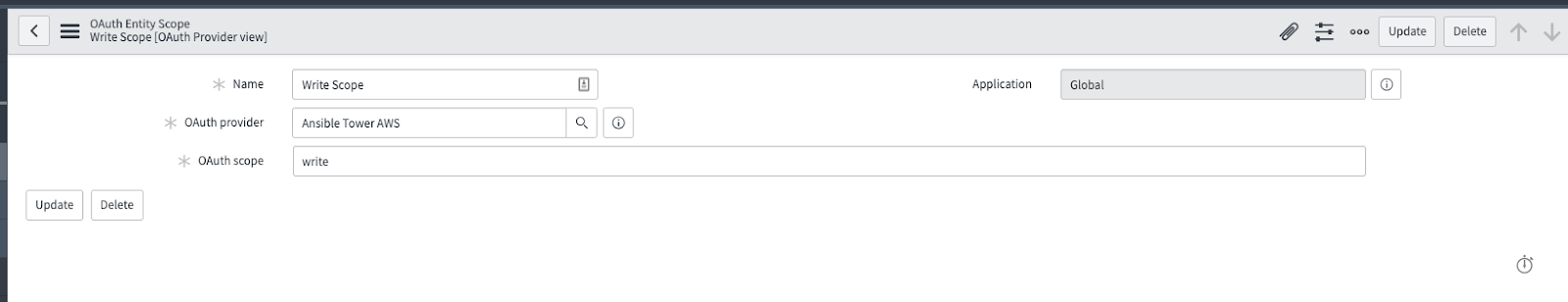
You should be taken out to the list of all Application Registries.

* Click back into the Application you just created. At the bottom, there should be two tabs:
* Click on the tab Oauth Entity Scopes.
* Under here, there is a section called Insert a new row….
* Double click here, and fill in the field to say Writing Scope.
* Click on the green check mark to confirm this change.
* Then, right-click inside the grey area at the top where it says Application Registries and click Save in the menu that pops up.



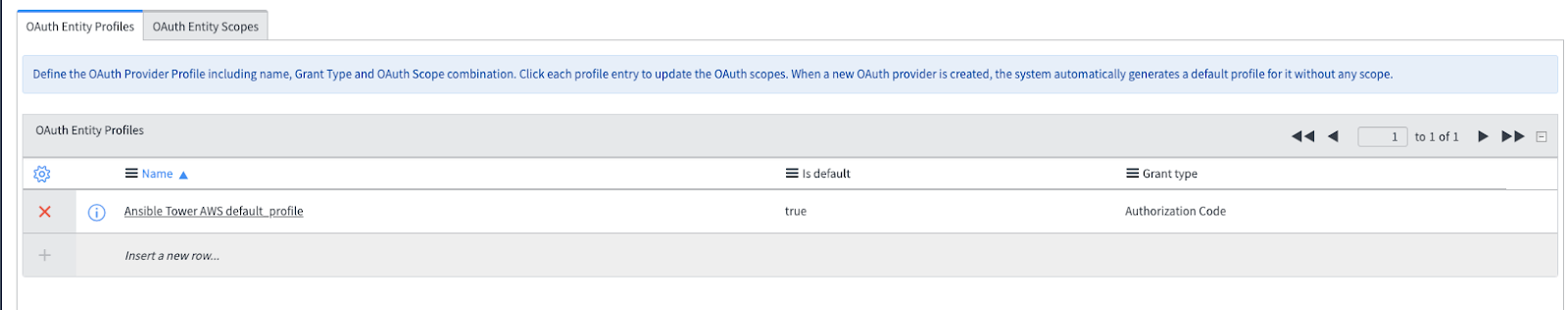
The writing scope should now be Clickable.

* Click on it, and in the dialog window that you are taken to, type write in the Oauth scope box.
* Click the Update button at the bottom.

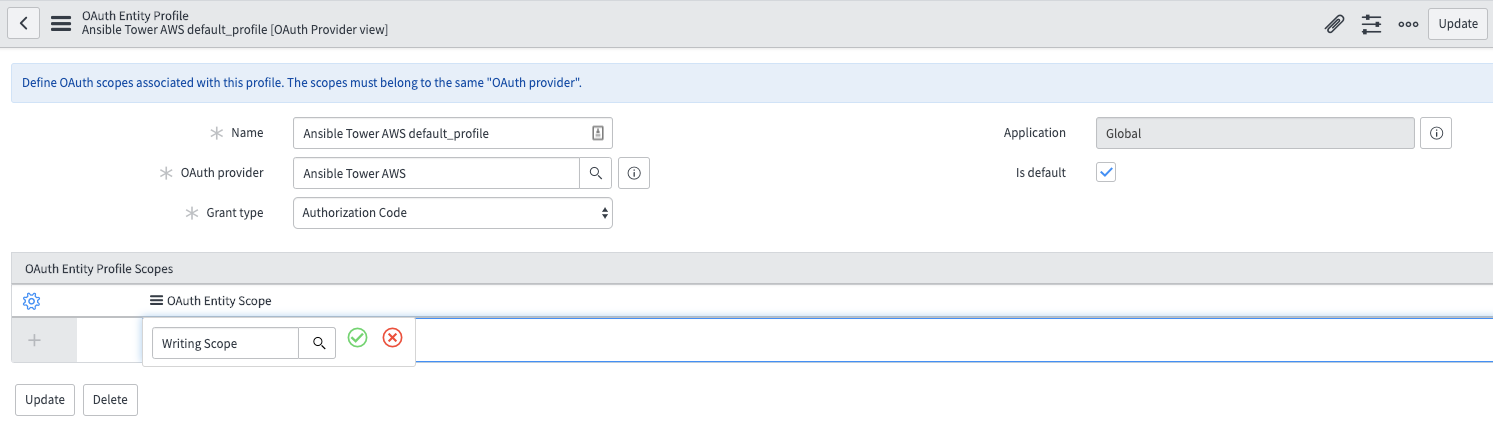


Back in the Application Settings page, scroll back to the bottom.

* Click the Oauth Entity Profiles tab.

There should be an entity profile populated - click into it.

You will be taken to the Oauth Entity Profile Window. At the bottom,

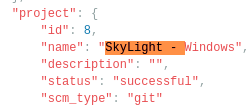
* Type “Writing Scope” into the Oauth Entity Scope field.
* Click the green check mark and update.

Navigate to System Web Services→ Outbound → REST Messages. Click the blue New button. In the resulting dialog window.

* Fill in the following fields:

Name: Ansible Tower

Endpoint: The url endpoint of the Ansible Tower action you wish to do. This can be taken from the browsable API at [https://<student>.<yourcompanyhere>.rhdemo.io/api/v2/job\_templates/](https://student1.bpx-poc.rhdemo.io/api/v2/job_templates/) search for the template SkyLight – Windows and you should find

Above that you should find the launch api " "launch": "[/api/v2/job\_templates/12/launch/](https://student1.bpx-poc.rhdemo.io/api/v2/job_templates/12/launch/)",

* Fill in the endpoint https://<student>.<yourcompanyhere>.rhdemo.io/api/v2/job\_templates/12/launch/

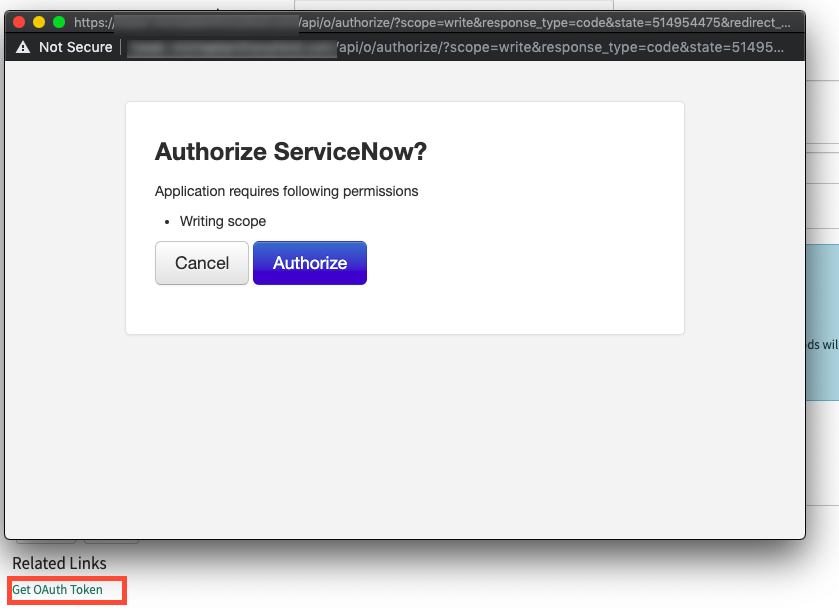
Authentication Type: Oauth 2.0

Oauth Profile: Select the Oauth profile you created

* Right-click inside the grey area at the top; click Save.
* Click the Get Oauth Token button on the REST Message screen.

This will generate a pop-up window asking to authorize ServiceNow against your Tower instance/cluster.

* Click Authorize. ServiceNow will now have an Oauth2 token to authenticate against your Ansible Tower server.



Under the HTTP Methods section at the bottom,

* Click the blue New button. At the new dialog window that appears,
* Fill in the following fields:

HTTP Method type POST

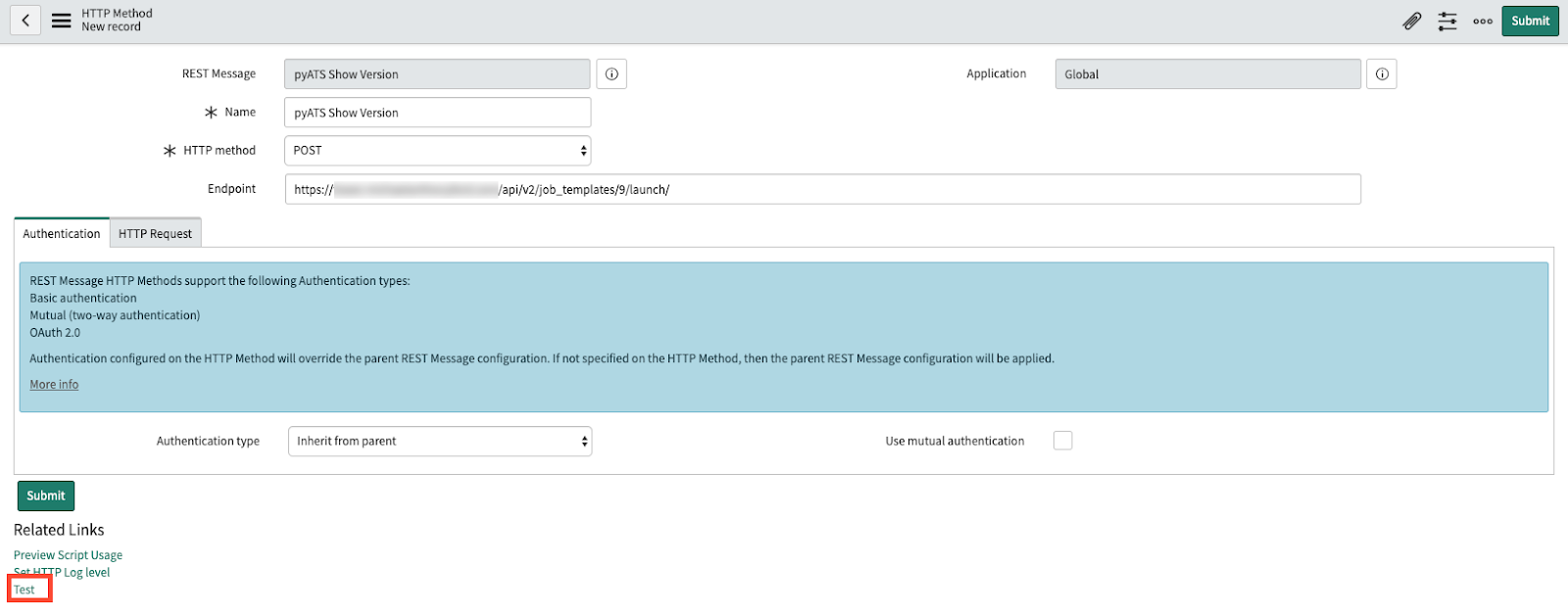
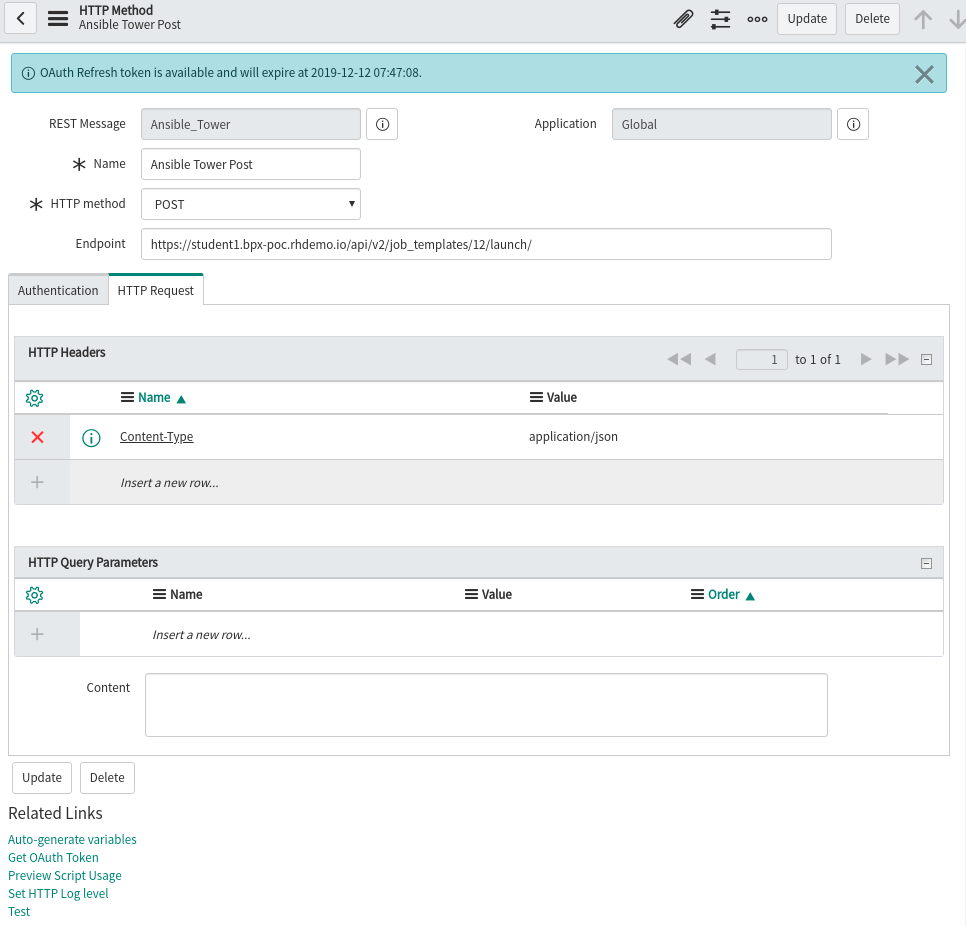
Name: Descriptive HTTP Method Name

Endpoint: The url endpoint of the Ansible Tower action you wish to do. This can be taken from the browsable API at https://<tower\_url>/api

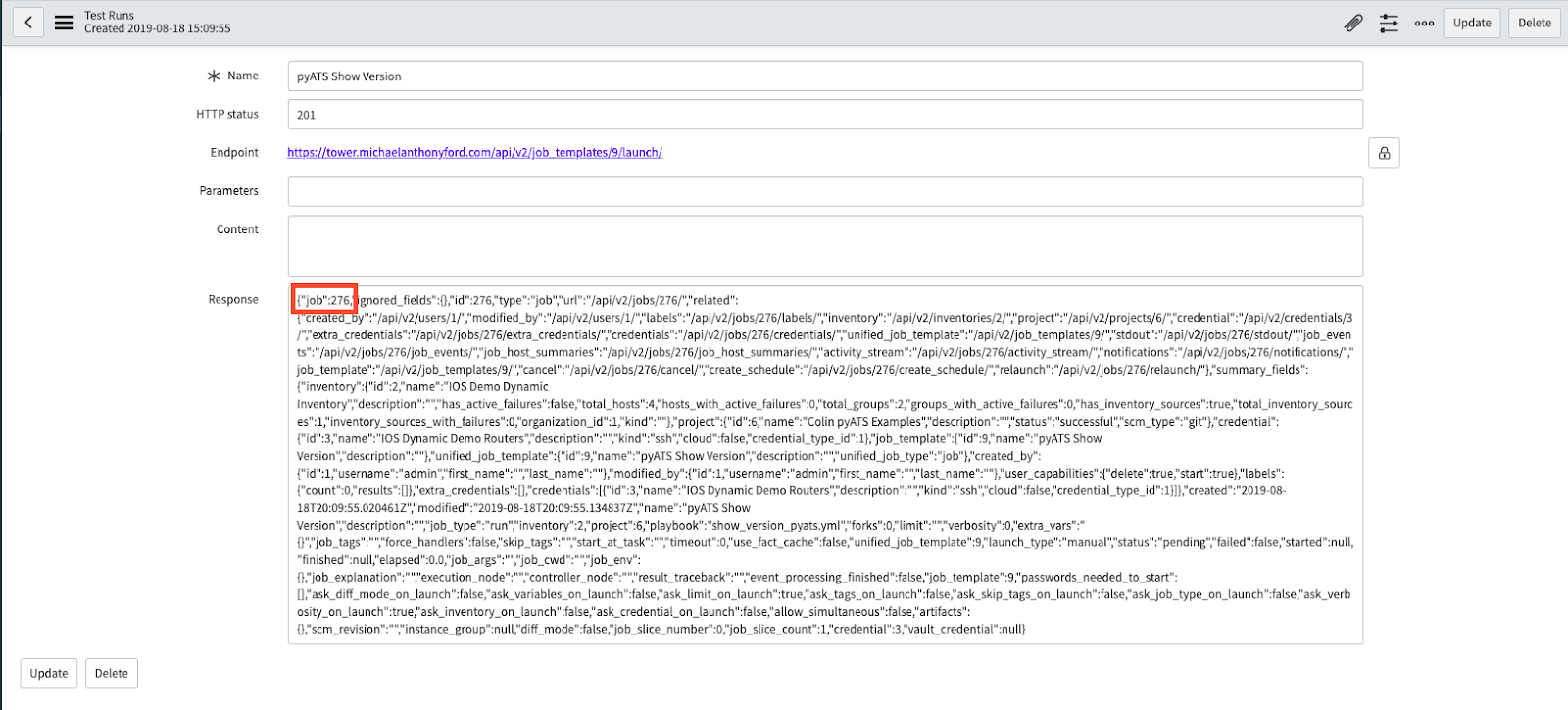
HTTP Headers (under the HTTP Request tab picture below)

The only HTTP Header that should be required is *Content-Type: application/json*

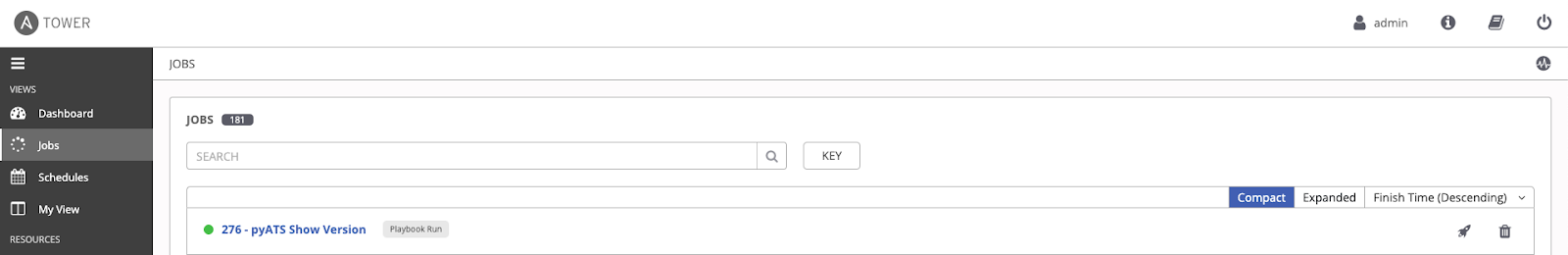
You can kick off a RESTful call to Ansible Tower using these parameters with the Test link.

NOTE: Before you click test open your Ansible Tower jobs window so you can watch it kick off.

* Click the Test link will take you to a results screen, which should indicate that the Restful call was sent successfully to Ansible Tower. In this example, ServiceNow kicks off an Ansible Tower job Template, and the response includes the Job ID in Ansible Tower: 276.



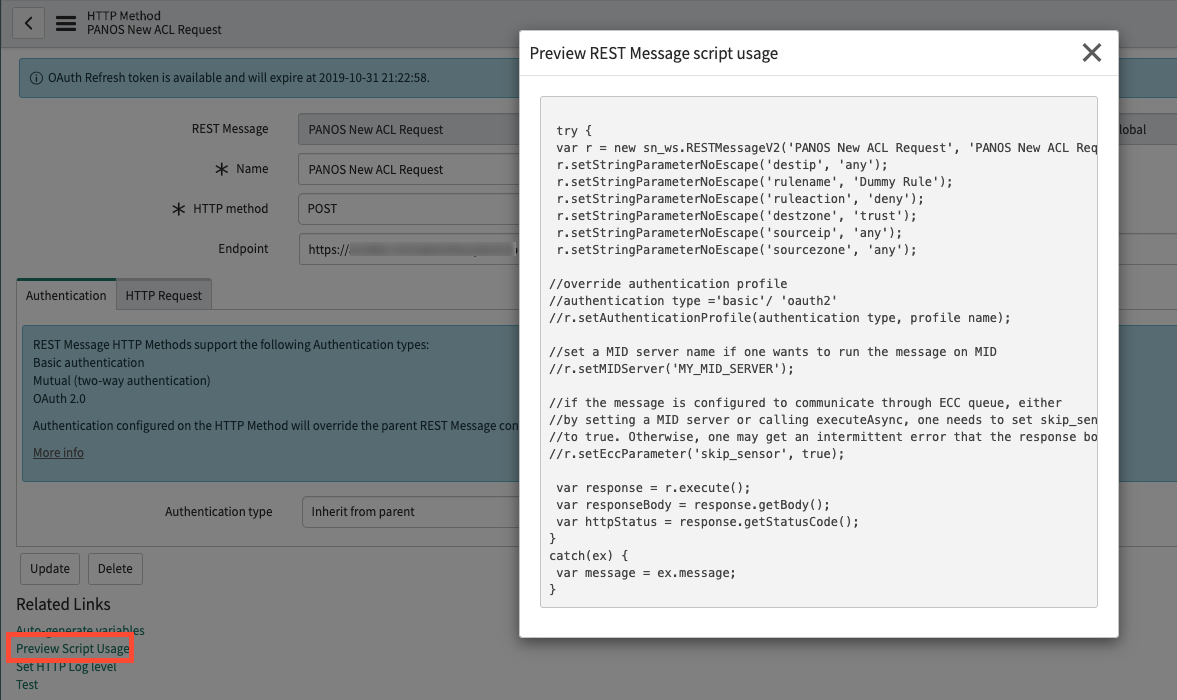
You can confirm that this Job Template was in fact started by going back to Ansible Tower and clicking the Jobs section on the left side of the screen; a Job with the same ID should be in the list (and, depending on the playbook size, may still be in process):

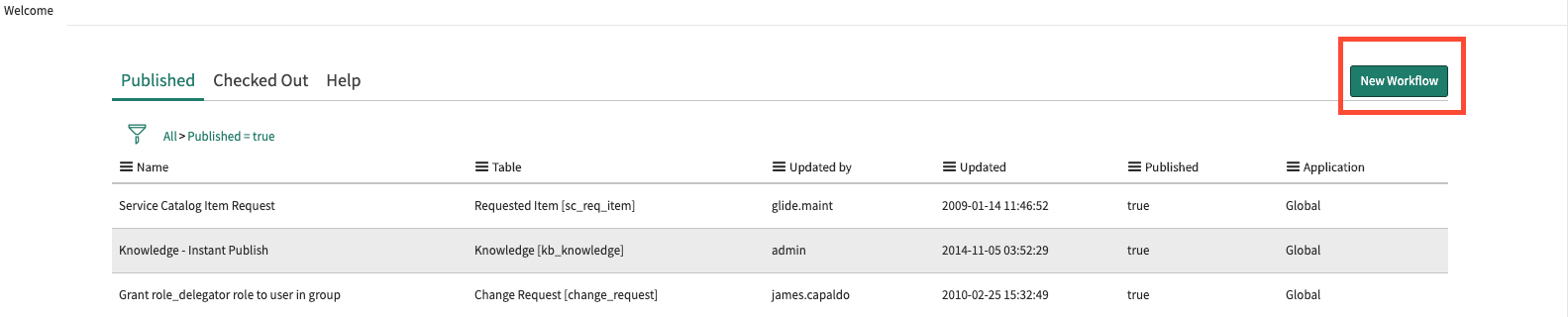


At this point you can see that we can make calls from ServiceNow to Ansible Tower.

# EXTRA – CREATING A CATALOG ITEM

Now that you are able to make outbound RESTful calls from ServiceNow to Ansible Tower, it’s time to create a catalog item for users to select in ServiceNow in a production self-service fashion. While in the HTTP Method options.

* Click the Preview Script Usage link:
* Copy the resulting script the appears, and paste it into a text editor to reference later.
* In ServiceNow, navigate to Workflow->Workflow Editor. This will open a new tab with a list of all existing ServiceNow workflows. Click on the blue New Workflow button:



In the New Workflow dialog box that appears, fill in the following options:

* Name: A descriptive name of the workflow
* Table: Requested Item [sc\_req\_item]

Everything else can be left alone.

* Click the Submit button.



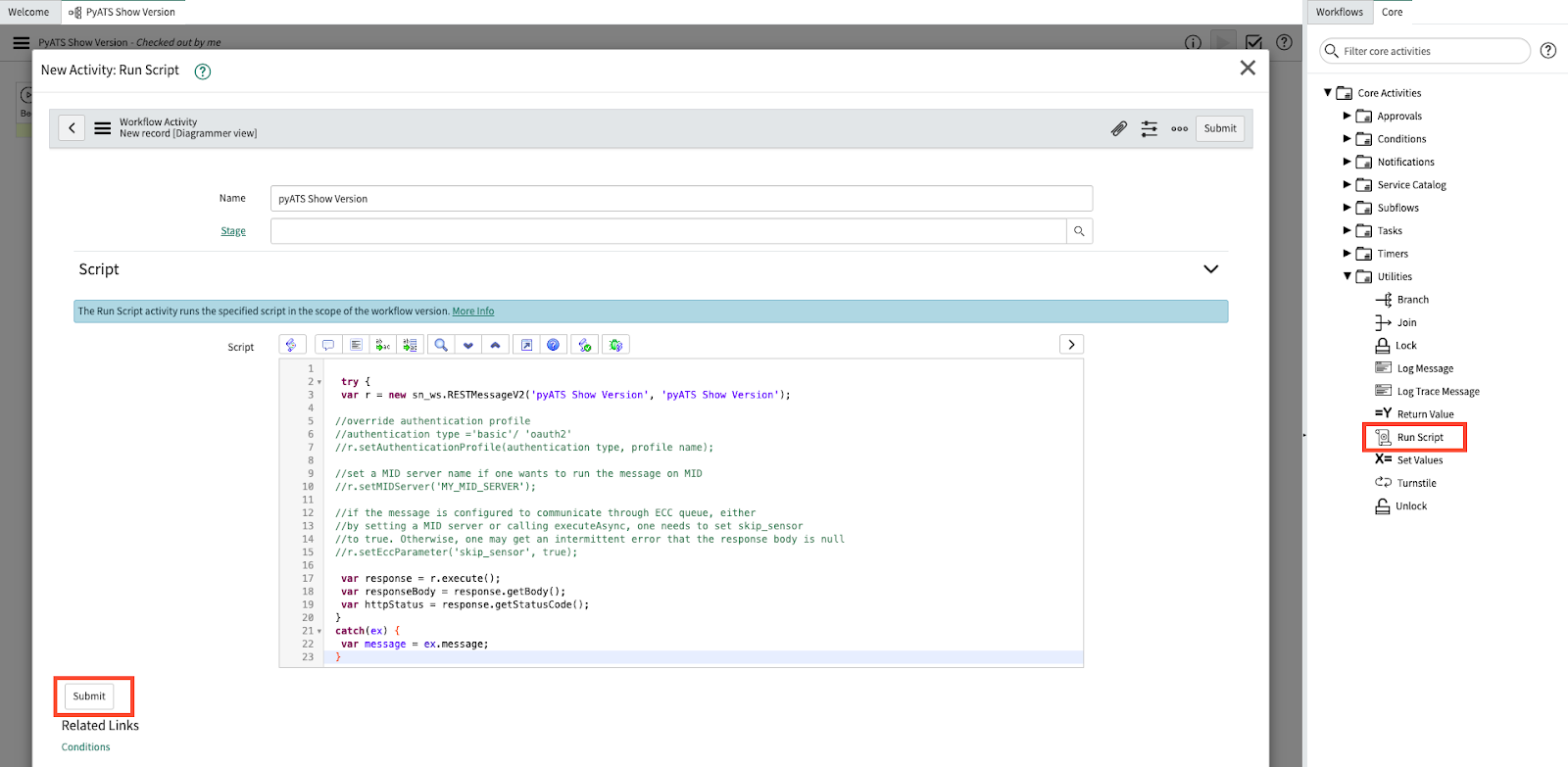
The resulting Workflow Editor will have only a Begin and End box.

* Click on the line (it will turn blue to indicate it has been selected), then press delete to get rid of it.



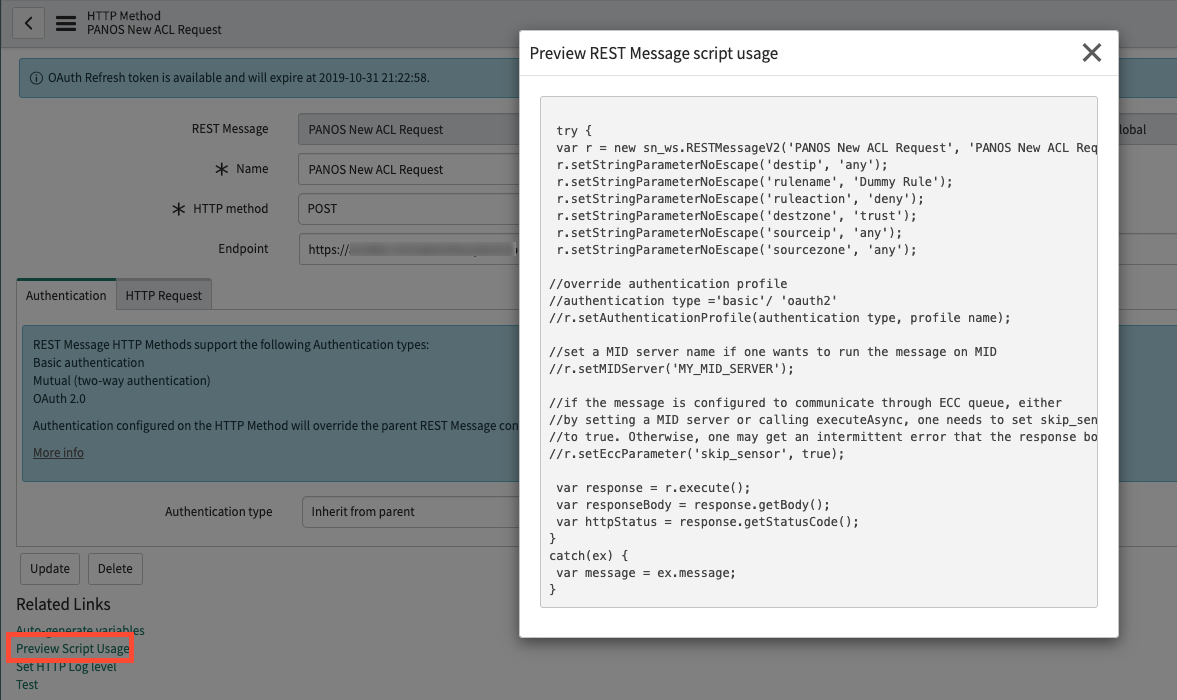
On the right side of the Workflow Editor Screen, select the Core tab and, under Core Activities → Utilities, drag the Run Script option into the Workflow Editor. In the new dialog box that appears.

* Type in a descriptive name, and paste in the script you captured from before. Click Submit to save the Script.



* Draw a connection from Begin, to the newly created Run Script Box, and another from the Run Script box to End. Afterward,
* click on the three horizontal lines to the left of the Workflow name, and select the Publish option.

You are now ready to associate this workflow with a catalog item.



Navigate to Service Catalog → Catalog Definitions→Maintain Items.

* Click the blue New button on the resulting item list. In the resulting dialog box, fill in the following fields:

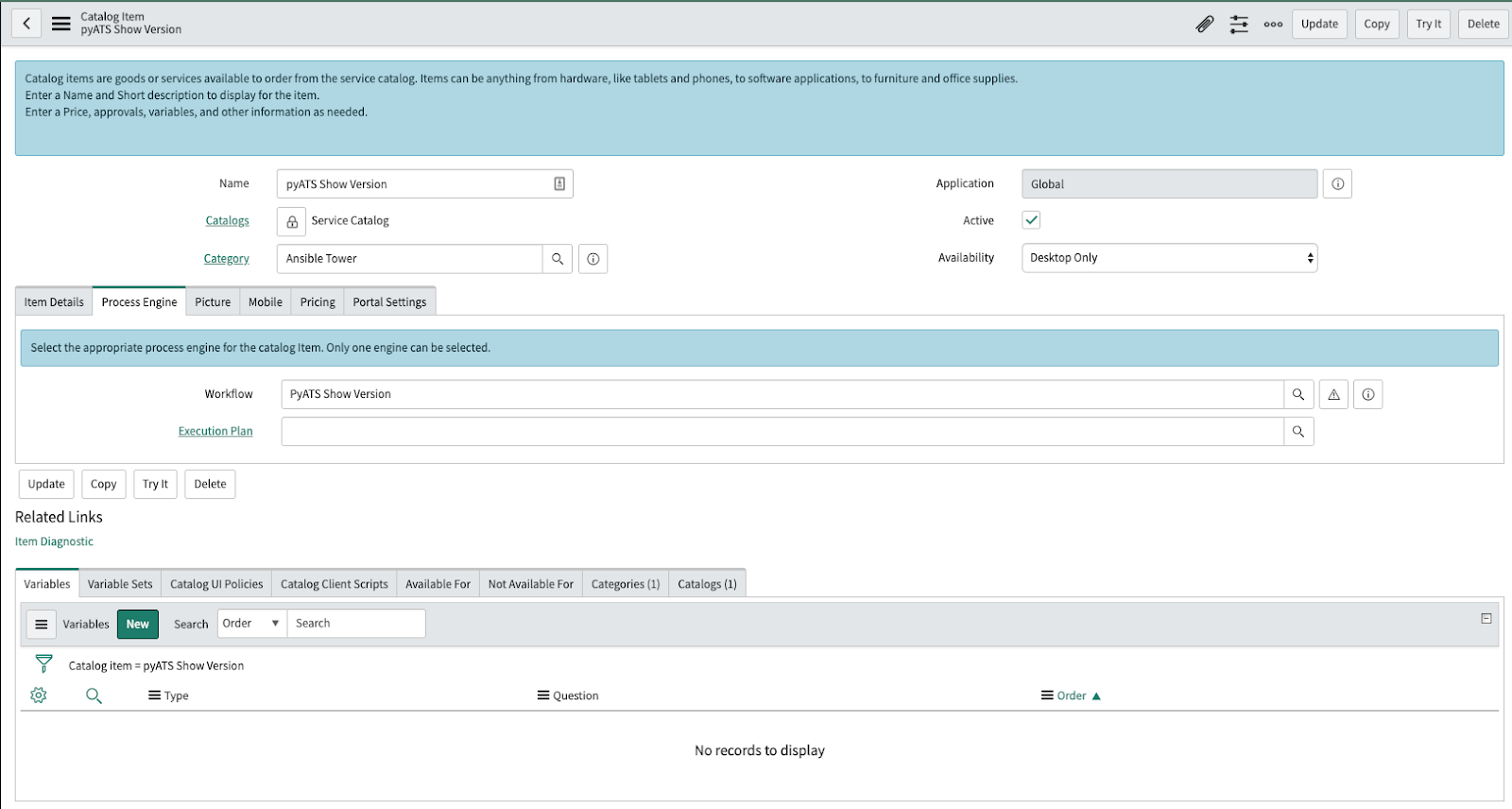
Name: Descriptive name of the Catalog Item

Catalog: The catalog that this item should be a part of

Category: Required if you wish users to be able to search for this item

In the Process Engine tab, populate the Workflow field with the Workflow you just created.

* Click the Submit Button. You’ve not created a new catalog item!



Lastly, to run this catalog item, navigate to Self-Service → Homepage and search for the catalog item you just created. Once found,

Click the order now button. You can see the results page pop up in ServiceNow, and you can confirm that the Job is being run in Ansible Tower.

# DOCUMENTATION

1. <https://docs.ansible.com/>

# RED HAT SUPPORT

* 1. **LAST BUT NOT LEAST**

1. RED HAT Inc..
2. 1-888-REDHAT-1 or 1-919-754-3700, then select the Menu Prompt for Customer Service
3. Spanish: 1-888-REDHAT-1 Option 5 or 1-919-754-3700 Option 5
4. Fax: 919-754-3701 (General Corporate Fax)
5. Web: [https://access.redhat.com/support/cases/#/case/new?intcmp=hp%7Ca%7Ca3%7Ccase](https://access.redhat.com/support/cases/" \l "/case/new?intcmp=hp|a|a3|case)
6. Email address: [customerservice@redhat.com](mailto:customerservice@redhat.com)