



Red Hat Ansible Tower Workshop

lightbulb.rhdemo.io

Workshop's Guide
(The Language of DevOps Automation)

Creation date:	01/25/17
Date of last modification:	02/11/19
Version:	2.2

INDEX

REVIEW	2
SUMMARY	3
Business Scenario	3
Workshop's Architecture	3
Workshop's Design	3
WORKSHOP: RED HAT ANSIBLE TOWER	4
Lab 1: Setting Up Ansible Tower: Organizations, Teams, Users and Credentials	4
Lab 2: Creating Ansible Tower Projects, Inventories and Job Templates	9
Lab 3: Automating IT Process using Ansible Tower Jobs	16
Lab 4: Automating IT Process Using Ansible Tower Workflows	20
Lab 5: Running Ad-Hoc Commands Using Ansible Tower	23
Lab 6: Automating Windows	25
Lab 7: Automating Network Devices	29

REVIEW



Change's log

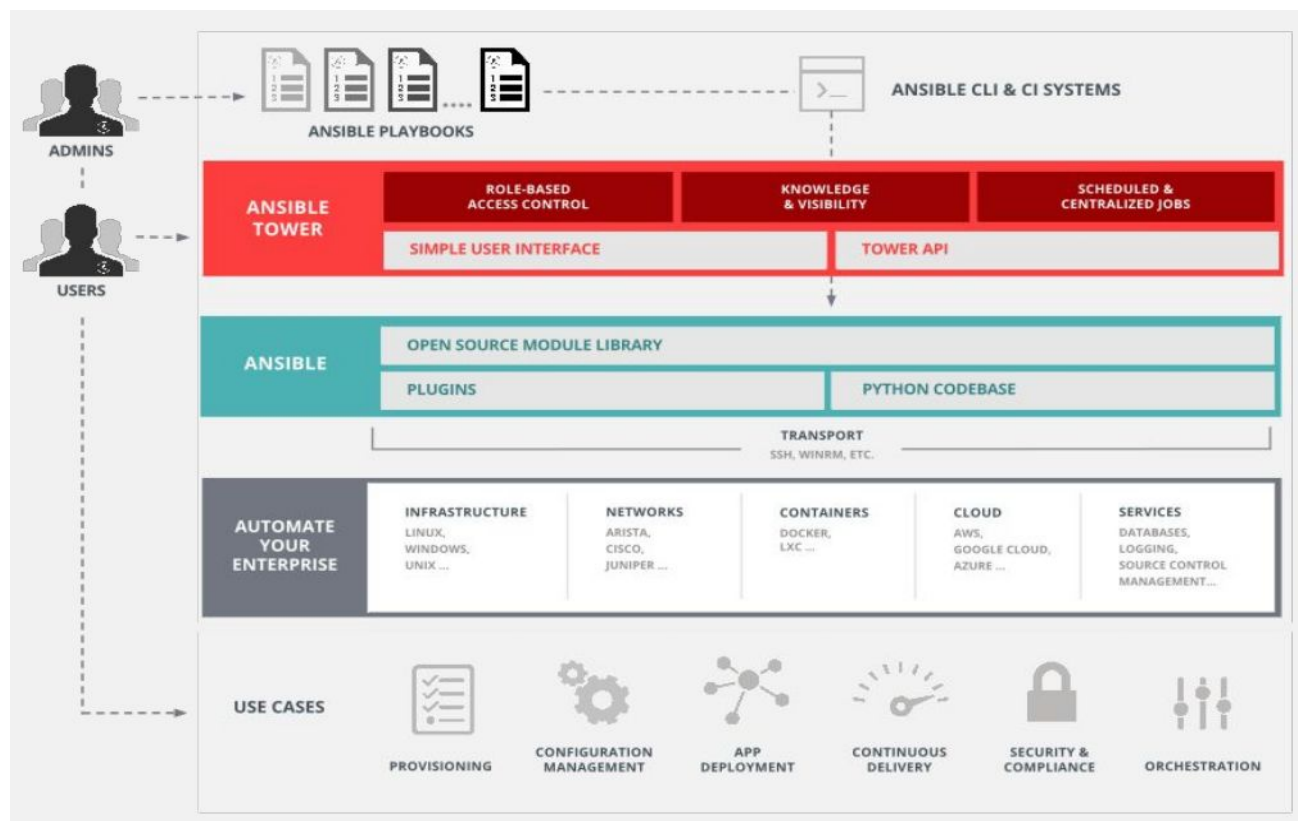
Fecha	Autor	Versión	Referencias
Sep/22/1017	Robert J. Calva	2.1	v2.0
Jan/08(2019	Robert J. Calva	2.2	v2.1

SUMMARY

Business Scenario

Ansible is the first **automation language** that can be read and written across IT. **Ansible** is the only **automation engine** that can automate the entire and continuous **delivery pipeline**.

Workshop's Architecture



Workshop's Design

This workshop has hands-on labs and presentations to provide a basic first-hand contact with the product:

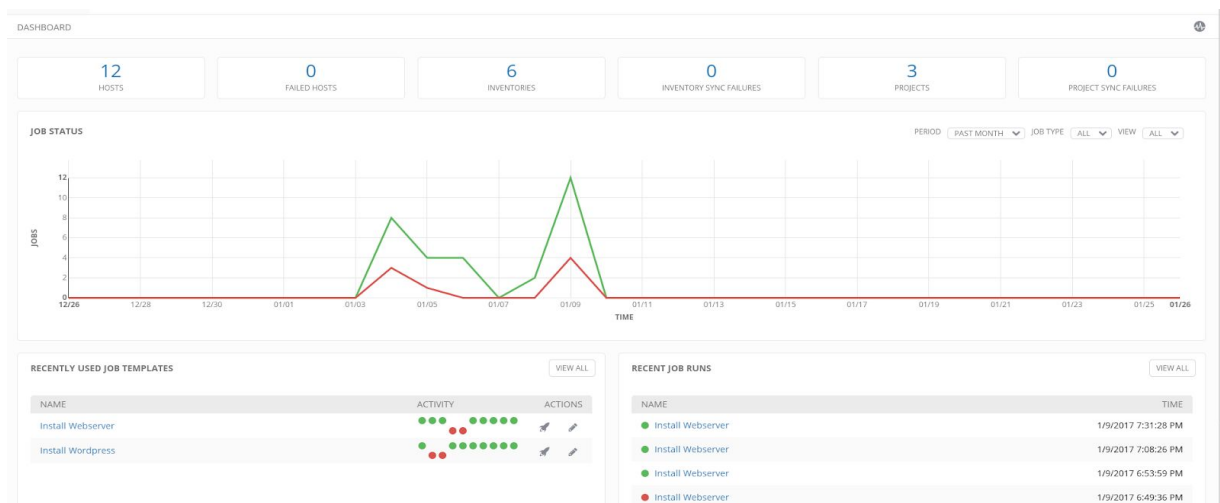
- Introductions ~ 30 min
- Ansible Tower General Presentation ~ 30 min
- Exploring the Dashboard and Tower Interface ~ 30 min
- Setting Up Ansible Tower: Credentials **(With Lab)** ~ 40 min
- Creating Ansible Tower Projects, Inventories and Job Templates **(With Lab)** ~ 40 min
- Automating IT Process using Ansible Tower Jobs **(with Lab)** ~ 40 min
- Automating IT Process using Workflows **(with Lab)** ~ 40 min
- Running Ad-Hoc Commands using Ansible Tower **(with Lab)** ~ 40 min
- Automating Windows **(with Lab)** ~ 40 min
- Automating Network Devices **(with Lab)** ~ 40 min




WORKSHOP: RED HAT ANSIBLE TOWER























Access to Ansible Tower Web Portal,

User: admin / Password: r3dh4t



1G. Create Credentials:

Navigate within the **Ansible Tower Portal** and click on **Credentials** (within RESOURCES Section). Click on  to add new Credentials:

CREDENTIALS 7			
SEARCH		KEY	
NAME ^	KIND	OWNERS	ACTIONS
AWS Credentials	Amazon Web Services	Default	  
CloudForms Credentials	Red Hat CloudForms	Default	  
Localhost Credentials	Machine	Default	  
OpenStack Admin Tenant	OpenStack	admin	  
SSH OSP Servers	Machine	admin	  
SSH Physical Servers	Machine	Default	  
VMware Credentials	VMware vCenter	admin	  

Create the new Credentials as follows:

Name: SSH Server Credentials

Description: SSH Server Credentials

Organization: Default

Type: Machine

Username: student# ⇐=(Use your student number here)

Password: r3dh4t

Click on **SAVE** to save the changes.

NEW CREDENTIAL

DETAILS

PERMISSIONS

* NAME ?

SSH Server Credentials

DESCRIPTION ?

SSH Server Credentials

ORGANIZATION ?

Workshop

* CREDENTIAL TYPE ?

Machine

TYPE DETAILS

USERNAME

root

PASSWORD

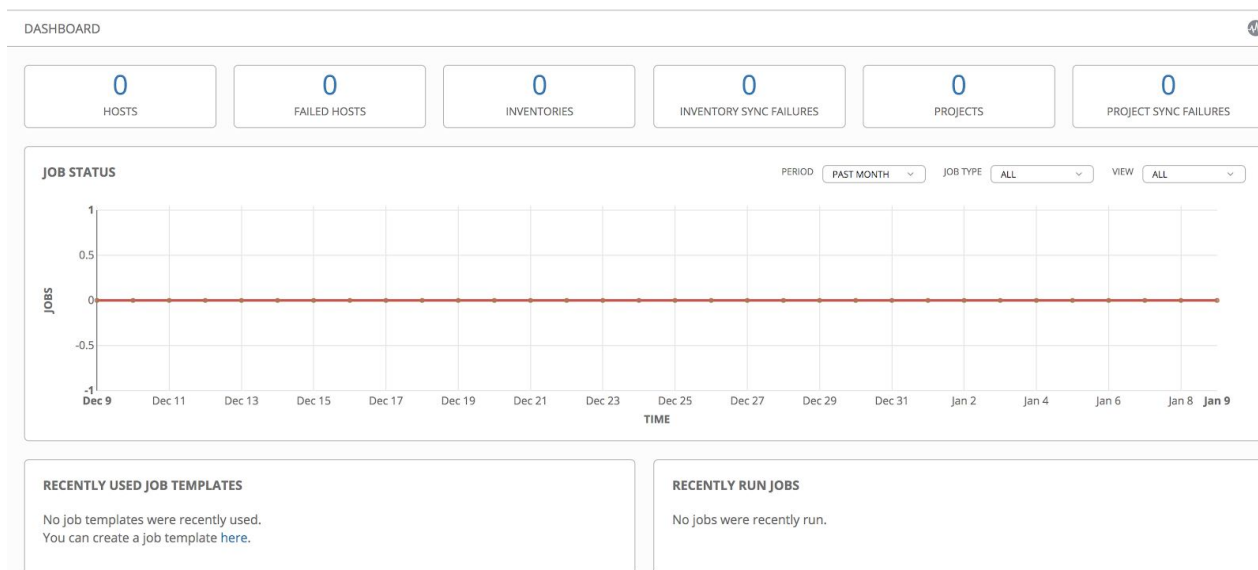
☐ Prompt on launch

SHOW

.....

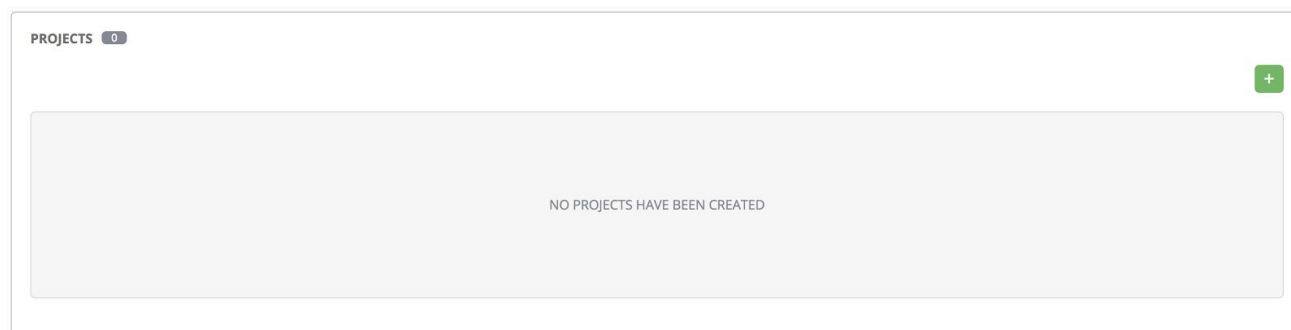
Lab 2: Creating Ansible Tower Projects, Inventories and Job Templates


Provisioning refers to the capacity an infrastructure has to deliver a resource and manage its life-cycle.



NOTE: You can see that you have no hosts, no projects, no Inventories, etc. because we are working within our new **Workshop Organization**.

2B. Create 2 new Projects:



Click on **Projects** (within RESOURCES Section) and then click on  button. Create the new App Server Project using next information:

NAME: App Project




DESCRIPTION: An Application Server Project

ORGANIZATION: Default


SCM TYPE: Git

SOURCE DETAILS (SCM URL): <https://github.com/leerich/ansible-examples.git>

SCM UPDATE OPTIONS:

- ☐ Clean 
- ☐ Delete on Update 
- ☒ Update Revision on Launch 

Click on **SAVE** to create the App Server Project.

Click on **Projects** (within RESOURCES Section) and then click on  button. Create the new Examples Project using next information:

NAME: Examples Project




DESCRIPTION: A Project with many examples

ORGANIZATION: Default

SCM TYPE: Git

SOURCE DETAILS (SCM URL): <https://github.com/leerich/ansible-examples.git>


SCM UPDATE OPTIONS:


- ☐ Clean 
- ☐ Delete on Update 
- ☒ Update Revision on Launch 













Click on **SAVE** to create the Examples Project.

Click on **Projects** (within RESOURCES Section). You will see your two new projects:

PROJECTS 2






NAME ^	TYPE ^	REVISION ^	LAST UPDATED ^	ACTIONS
 App Project	Git	4631505 	1/9/2019 11:18:32 PM	   
 Examples Project	Git	fca475b 	1/9/2019 11:19:17 PM	   

ITEMS 1 - 2


2C. Create a new Inventory:

INVENTORIES

HOSTS



PLEASE ADD ITEMS TO THIS LIST

Click on **Inventories** (within RESOURCES Section) and then click on  button and select “Inventory” (not “Smart Inventory”). Create the new Inventory using next information:

NAME: Physical Inventory

DESCRIPTION: An Inventory with physical hosts

ORGANIZATION: Workshop

Click on **SAVE** to create the new Inventory.

Physical Inventory

DETAILS

PERMISSIONS

GROUPS

HOSTS

SOURCES

COMPLETED JOBS

* NAME

Physical Inventory

DESCRIPTION

An Inventory with physical hosts


* ORGANIZATION

Q

Workshop

INSIGHTS CREDENTIAL

Q

INSTANCE GROUPS 

Q

2D. Create 3 new Groups within Physical Inventory:

Within our recently created **Physical Inventory**, click on **GROUPS**:

Physical Inventory

DETAILS

PERMISSIONS


GROUPS

HOSTS

SOURCES


COMPLETED JOBS

RUN COMMANDS



PLEASE ADD ITEMS TO THIS LIST

Now we need to **add three groups** as follows:

Click on  button to create new group **lamp-servers**:

NAME: lamp-servers

DESCRIPTION: A Group of Webservers

Click on **SAVE** to create the new group.

lamp-servers

DETAILS

GROUPS

HOSTS

* NAME

lamp-servers

DESCRIPTION

A Group of Webservers

VARIABLES

YAML

JSON

1

CANCEL

SAVE

Physical Inventory

DETAILS

PERMISSIONS

GROUPS

HOSTS

SOURCES

COMPLETED JOBS

SEARCH

Q

KEY

RUN COMMANDS

+

GROUPS

lamp-servers

ACTIONS

ITEMS 1 - 1

Click on  button again to create new group **wordpress-servers** (Physical Inventory will appear below the page):

NAME: wordpress-servers
DESCRIPTION: A Group of Wordpress Servers

Click on **SAVE** to create the new group.

wordpress-servers

DETAILS

GROUPS

HOSTS

* NAME

wordpress-servers

DESCRIPTION

A Group of Wordpress Servers

VARIABLES

YAML

JSON

1

CANCEL

SAVE

Physical Inventory

DETAILS

PERMISSIONS

GROUPS

HOSTS

SOURCES

COMPLETED JOBS

SEARCH

Q

KEY

RUN COMMANDS

+

GROUPS

lamp-servers

wordpress-servers

ACTIONS

ITEMS 1 - 2

Click on  button again to create the new group **appservers-group** (Physical Inventory will appear below the page):

NAME: appservers-group
DESCRIPTION: A Group of JBoss Application Servers

Click on **SAVE** to create the new group.


The image shows two screenshots from a web interface. The top screenshot is the configuration form for a group named 'appservers-group'. It has tabs for 'DETAILS', 'GROUPS', and 'HOSTS'. The 'DETAILS' tab is active, showing fields for 'NAME' (appservers-group) and 'DESCRIPTION' (A Group of JBoss Application Servers). There is also a 'VARIABLES' section with 'YAML' and 'JSON' tabs. The bottom screenshot shows the 'Physical Inventory' section with tabs for 'DETAILS', 'PERMISSIONS', 'GROUPS', 'HOSTS', 'SOURCES', and 'COMPLETED JOBS'. The 'GROUPS' tab is active, displaying a list of groups: 'appservers-group', 'lamp-servers', and 'wordpress-servers'. Each group has a checkbox and an 'ACTIONS' column with edit and delete icons.

2E. Create 3 new Hosts, each one within the corresponding Group:

Click on **Inventories** → **Physical Inventory** → **GROUPS** → **lamp-servers** and then click on **HOSTS**. After that, click on  button (selecting *New Host*) and add one host as follows:


HOST NAME: <workshop name>-student#-node1.rhdemo.io
DESCRIPTION: Web Server

Click on **SAVE** to create the new host.

Click on **Inventories** → **Physical Inventory** → **GROUPS** → **wordpress-servers** and then click on **HOSTS**. After that, click on  button (selecting *New Host*) and add one host as follows:

HOST NAME: <workshop name>-student#-node2.rhdemo.io
DESCRIPTION: Wordpress Server

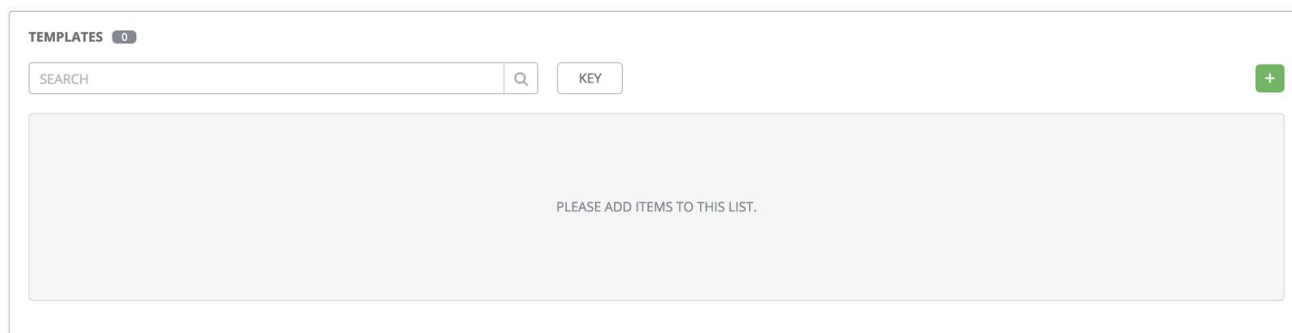
Click on **SAVE** to create the new host.


Click on **Inventories** → **Physical Inventory** → **GROUPS** → **appservers-group** and then click on **HOSTS**. After that, click on  button (selecting *New Host*) and add one host as follows:

HOST NAME: <workshop name>-student#-node3.rhdemo.io
DESCRIPTION: JBoss Application Server

Click on **SAVE** to create the new host.

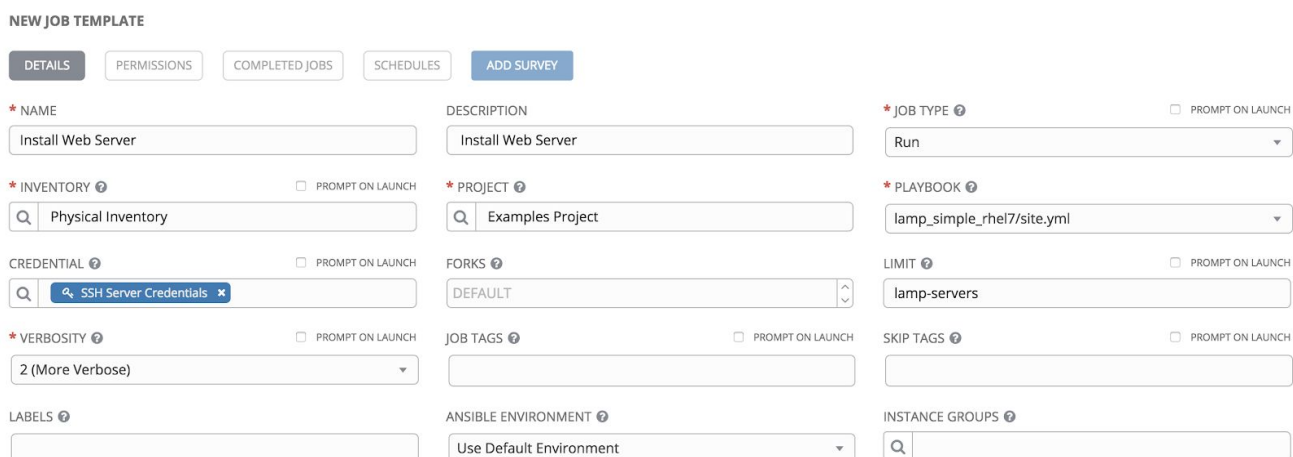
2F. Create 3 new Job Templates:




Click on **Templates** (within RESOURCES Section) and then click on  button (selecting *Job Template*) to create a new Job Template as follows:

NAME: Install Web Server
DESCRIPTION: Install Web Server
JOB TYPE: Run
INVENTORY: Physical Inventory
PROJECT: Examples Project
PLAYBOOK: lamp_simple_rhel7/site.yml
MACHINE CREDENTIAL: SSH Server Credentials
LIMIT: lamp-servers
VERBOSITY: 2 (More Verbose)
Enable Privilege Escalation: Checked

Click on **SAVE** to create the new Job Template




Click again on **Templates** and then click on  button (selecting *Job Template*) to create another Job Template as follows:

NAME: Install Wordpress Server
DESCRIPTION: Install Wordpress Server
JOB TYPE: Run
INVENTORY: Physical Inventory
PROJECT: Examples Project
PLAYBOOK: wordpress_nginx_rhel7/site.yml
MACHINE CREDENTIAL: SSH Server Credentials
LIMIT: wordpress-servers
VERBOSITY: 2 (More Verbose)
Enable Privilege Escalation: Checked

Click on **SAVE** to create the new Job Template

NEW JOB TEMPLATE

NEW JOB TEMPLATE				
DETAILS	PERMISSIONS	COMPLETED JOBS	SCHEDULES	ADD SURVEY
* NAME	DESCRIPTION		* JOB TYPE	<input type="checkbox"/> PROMPT ON LAUNCH
Install Wordpress Server	Install Wordpress Server		Run	
* INVENTORY	<input type="checkbox"/> PROMPT ON LAUNCH	* PROJECT	* PLAYBOOK	
Physical Inventory		Examples Project	wordpress-nginx_rhel7/site.yml	
CREDENTIAL	<input type="checkbox"/> PROMPT ON LAUNCH	FORKS	LIMIT	<input type="checkbox"/> PROMPT ON LAUNCH
SSH Server Credentials		DEFAULT	wordpress-servers	
* VERBOSITY	<input type="checkbox"/> PROMPT ON LAUNCH	JOB TAGS	SKIP TAGS	<input type="checkbox"/> PROMPT ON LAUNCH
2 (More Verbose)				
LABELS	ANSIBLE ENVIRONMENT		INSTANCE GROUPS	
	Use Default Environment			

Click again on **Templates** and then click on  button (selecting *Job Template*) to create another Job Template as follows:

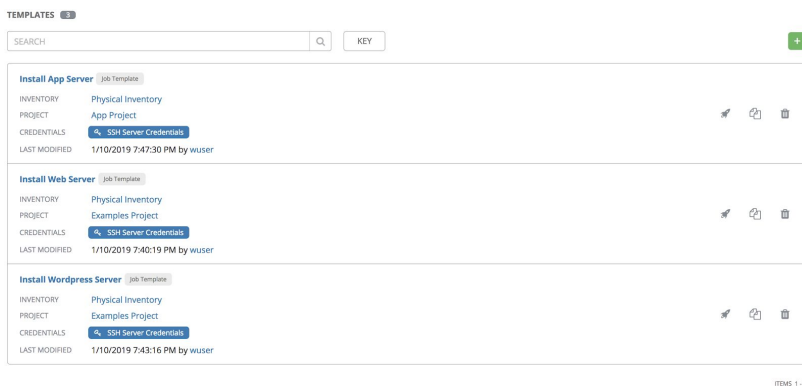
NAME: Install App Server
DESCRIPTION: Install JBoss Application Server
JOB TYPE: Run
INVENTORY: Physical Inventory
PROJECT: App Project
PLAYBOOK: jboss-standalone/site.yml
MACHINE CREDENTIAL: SSH Server Credentials
LIMIT: appservers-group
VERBOSITY: 2 (More Verbose)
Enable Privilege Escalation: Checked

Click on **SAVE** to create the new Job Template

NEW JOB TEMPLATE

NEW JOB TEMPLATE				
DETAILS	PERMISSIONS	COMPLETED JOBS	SCHEDULES	ADD SURVEY
* NAME	DESCRIPTION		* JOB TYPE	<input type="checkbox"/> PROMPT ON LAUNCH
Install App Server	Install JBoss Application Server		Run	
* INVENTORY	<input type="checkbox"/> PROMPT ON LAUNCH	* PROJECT	* PLAYBOOK	
Physical Inventory		App Project	jboss-standalone/site.yml	
CREDENTIAL	<input type="checkbox"/> PROMPT ON LAUNCH	FORKS	LIMIT	<input type="checkbox"/> PROMPT ON LAUNCH
SSH Server Credentials		DEFAULT	appservers-group	
* VERBOSITY	<input type="checkbox"/> PROMPT ON LAUNCH	JOB TAGS	SKIP TAGS	<input type="checkbox"/> PROMPT ON LAUNCH
2 (More Verbose)				
LABELS	ANSIBLE ENVIRONMENT		INSTANCE GROUPS	
	Use Default Environment			


Click again on **Templates**. You will see three Job Templates as follows:



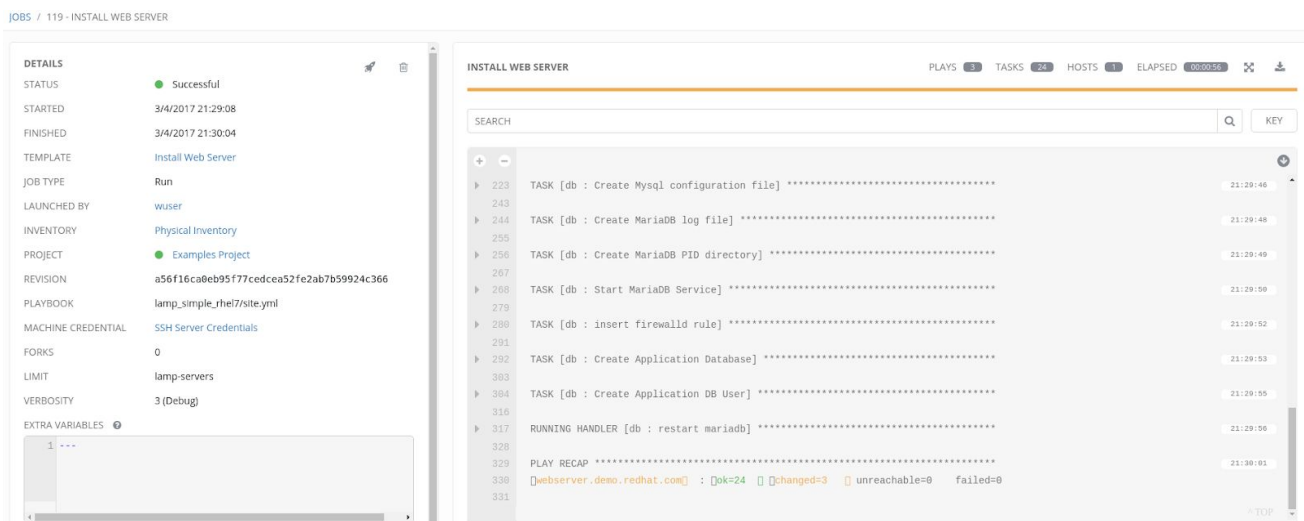
Lab 3: Automating IT Process using Ansible Tower Jobs

3A. Create new Apache Webserver via Job Template:

First, click on your **Web Server URL** to be sure it's not installed.

Then, access to Ansible Tower. Click on **Templates** and find the “**Install Web Server**” Job Template. Click  on to start a job creation using this template.

This Action will launch the **Install Web Server** Job Template, starting the Webserver installation. You will be redirected to a dynamic Job Output page:



You will see the Status of the Job, Started / Finished Time, Tasks, etc.

If your **Job** finished **Ok**, with Status **Successful**, go to your **Web Server URL** and make sure your Web Server is Up & Running.

Hello World!

My Web App was deployed via **Red Hat Ansible Tower**.

We will be taking a look at the Ansible Playbook we are using for this Job Template, following this link:

https://github.com/leerich/ansible-examples/blob/master/lamp_simple_rhel7/site.yml

3B. Create new Wordpress Website via Job Template:

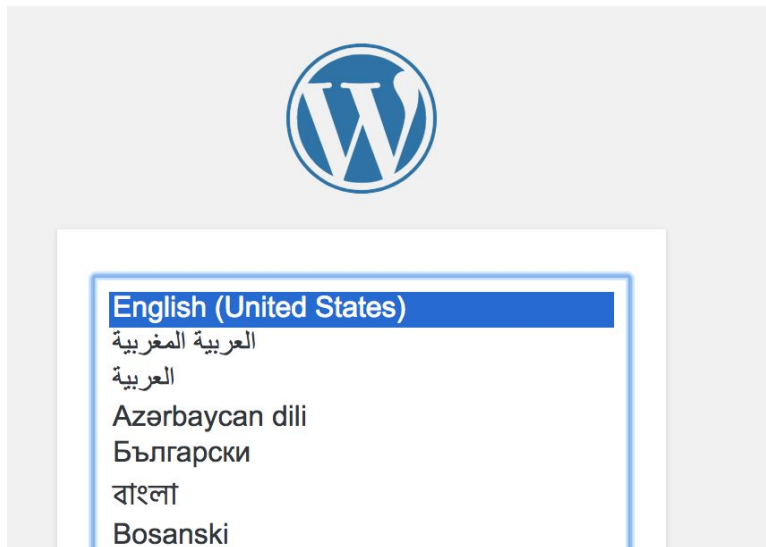
First, click on your **Wordpress Server URL** to be sure it's not installed.

Then, access to Ansible Tower. Click on **Templates** and find the “**Install Wordpress Server**” Job Template. Click on **Run** to start a job creation using this template.

This Action will launch the **Install Wordpress Server** Job Template, starting the Website installation. You will be redirected to a dynamic Job Output page:

You will see the Status of the Job, Started / Finished Time, Tasks, etc.

If your **Job** finished **Ok**, with Status **Successful**, go to your **Wordpress Server URL** and make sure your Wordpress Server is Up & Running. **Play a moment with your new Wordpress!**




We will be taking a look at the Ansible Playbook we are using for this Job Template, following this link:

https://github.com/leerich/ansible-examples/blob/master/wordpress-nginx_rhel7/site.yml

3C Create new JBoss Application Server via Job Template:

First, click on your **JBoss Application Server URL** to be sure it's not installed.

Then, access to Ansible Tower. Click on **Templates** and find the **"Install App Server"** Job Template. Click  on to start a job creation using this template.

This Action will launch the **Install App Server** Job Template, starting the JBoss App Server installation. You will be redirected to a dynamic Job Output page:

The screenshot displays the Ansible Tower interface for a job titled "INSTALL APP SERVER". On the left, the "DETAILS" panel shows the job status as "Successful", started on 3/4/2017 at 21:56:25, and finished at 21:58:52. It lists the template as "Install App Server", job type as "Run", and launched by "wuser". The inventory is "Physical inventory", project is "App Project", and revision is "28173530d51c7e31616696a2913d0bba33e5aba2". The playbook is "jboss-standalone/site.yml", machine credential is "SSH Server Credentials", forks are 0, limit is "appservers-group", and verbosity is 0 (Normal). The "EXTRA VARIABLES" section is empty.

The main panel shows the "PLAY RECAP" and "TASKS" for the job. The tasks are as follows:

- TASK [jboss-standalone : Enable JBoss to be started at boot] ***** [ok: [jboss.demo.redhat.com]] 21:57:51
- TASK [jboss-standalone : deploy iptables rules] ***** [skipping: [jboss.demo.redhat.com]] 21:57:53
- TASK [jboss-standalone : Ensure that firewall is installed] ***** [changed: [jboss.demo.redhat.com]] 21:57:53
- TASK [jboss-standalone : Ensure that firewall is started] ***** [changed: [jboss.demo.redhat.com]] 21:58:38
- TASK [jboss-standalone : deploy firewall rules] ***** [changed: [jboss.demo.redhat.com] => (item=8080/tcp)] [changed: [jboss.demo.redhat.com] => (item=8443/tcp)] 21:58:48
- RUNNING HANDLER [jboss-standalone : restart jboss] ***** [changed: [jboss.demo.redhat.com]] 21:58:44

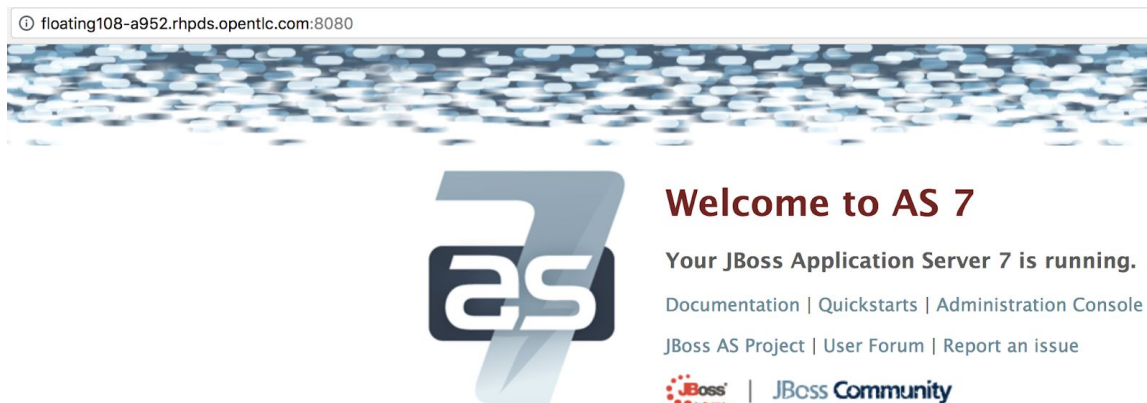
The "PLAY RECAP" shows: [jboss.demo.redhat.com] : [ok=16] [changed=14] [unreachable=0] [failed=0] 21:58:52

You will see the Status of the Job, Started / Finished Time, Tasks, etc.

If your **Job** finished **Ok**, with Status **Successful**, go to your **JBoss Application Server URL** <http://<node3 IP address>:8080> and make sure your App Server is Up & Running, **accessing via :8080 Port!**

Click on **Administration Console** link. You can access to the **JBoss Administration Console**:

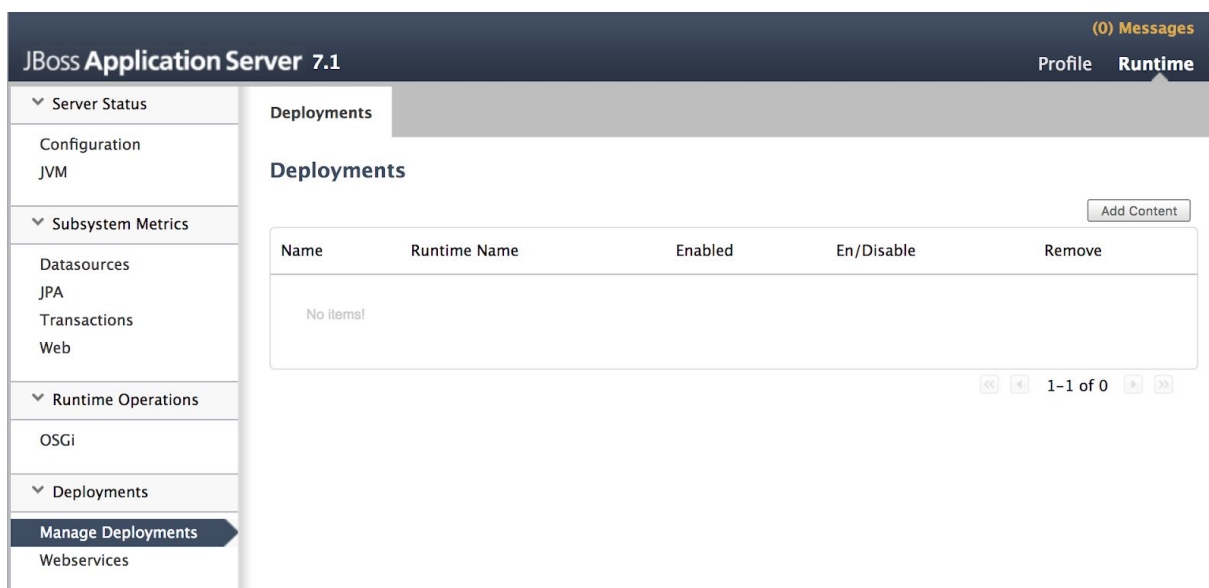
User: admin / **Password:** r3dh4t



We will be taking a look at the Ansible Playbook we are using for this Job Template, following this link:


<https://github.com/leerich/ansible-examples/blob/master/jboss-standalone/site.yml>

Now click on **Deployments** → **Manage Deployments**. You will see that we have no deployments yet. We will be creating a workflow template to deploy two new Java Applications on JBoss in Lab 4.



Lab 4: Automating IT Process Using Ansible Tower Workflows

First as **wuser**, we need to create another Ansible Tower Job to deploy a Java Application within our JBoss App Server:

Click on **Templates** and then click on  button (selecting *Job Template*)→ **Job Template** to create a new Job Template as follows:

NAME: Install Java App

DESCRIPTION: Install a Java App within JBoss

JOB TYPE: Run

INVENTORY: Physical Inventory

PROJECT: App Project

PLAYBOOK: jboss-standalone/deploy-application.yml
MACHINE CREDENTIAL: SSH Server Credentials
LIMIT: appservers-group
VERBOSITY: 2 (More Verbose)
Enable Privilege Escalation: Checked


Click on **SAVE** to create the new Job Template

NEW JOB TEMPLATE

DETAILS PERMISSIONS COMPLETED JOBS SCHEDULES ADD SURVEY

* NAME Install Java App	DESCRIPTION Install a Java App within JBoss	* JOB TYPE ? Run <input type="checkbox"/> PROMPT ON LAUNCH
* INVENTORY ? <input type="checkbox"/> PROMPT ON LAUNCH Physical Inventory	* PROJECT ? App Project	* PLAYBOOK ? jboss-standalone/deploy-application.yml
CREDENTIAL ? <input type="checkbox"/> PROMPT ON LAUNCH SSH Server Credentials	FORKS ? DEFAULT	LIMIT ? <input type="checkbox"/> PROMPT ON LAUNCH appservers-group
* VERBOSITY ? <input type="checkbox"/> PROMPT ON LAUNCH 1 (Verbose)	JOB TAGS ? <input type="checkbox"/> PROMPT ON LAUNCH	SKIP TAGS ? <input type="checkbox"/> PROMPT ON LAUNCH
LABELS ?	ANSIBLE ENVIRONMENT ? Use Default Environment	INSTANCE GROUPS ?

Then, we need to create a new **Workflow Job Template**, so we can later create a workflow of Job Templates:

Click on **Templates** and then click on  button (selecting *Workflow Template*)→ **Workflow Template** to create a new Workflow Template as follows:

NAME: Deploy a Java App within JBoss
DESCRIPTION: Deploying a Java App within JBoss
ORGANIZATION: Default

NEW WORKFLOW JOB TEMPLATE

DETAILS PERMISSIONS COMPLETED JOBS SCHEDULES ADD SURVEY WORKFLOW VISUALIZER

* NAME Deploy a Java App within JBoss	DESCRIPTION Deploying a Java App within JBoss	* ORGANIZATION Workshop
LABELS ?	OPTIONS <input type="checkbox"/> Enable Concurrent Jobs ?	
EXTRA VARIABLES ? YAML JSON 1 ---		

CANCEL SAVE

Click on **SAVE**. Then, create a workflow within our new Workflow Job Template, so click on **WORKFLOW VISUALIZER**:

Deploy a Java App within JBoss

DETAILS PERMISSIONS COMPLETED JOBS SCHEDULES ADD SURVEY WORKFLOW VISUALIZER

Next, click on **START** button: 

We can now select our first initial Job Template. Select **Install Java App** Job Template and click on **SELECT**.

Now click on **Install Java App** Job box:



Click on **Green Plus** to add a sequential Job Template. Select **Install App Server** Job Template and select **On Failure** as **TYPE**. This Job will run if our first Job *Install Java App* has a **failed status**, so *Install App Server* Job will ensure to have all the necessary JBoss App Server infrastructure.

NAME ^	
<input checked="" type="radio"/> Install App Server	INFO
<input type="radio"/> Install Java App	INFO
<input type="radio"/> Install Web Server	INFO
<input type="radio"/> Install Wordpress Server	INFO

ITEMS 1 - 4 OF 4

* TYPE

☐ On Success

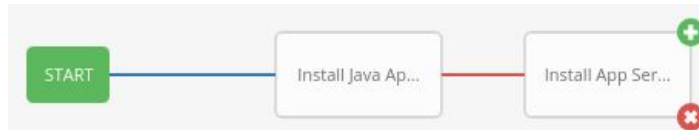
☒ On Failure

☐ Always

Click on **SELECT** to select the Job

Template.

Now click on on **Install App Server** Job box:



Click on **Green Plus** to add a sequential Job Template. Select **Install Java App** Job Template and select **On Success** as **TYPE**. This Job will run if our *Install App Server* Job has a **successful status**, so *Install Java App* Job will install the Java Application on our JBoss App Server infrastructure.

NAME ^	
<input type="radio"/> Install App Server	INFO
<input checked="" type="radio"/> Install Java App	INFO
<input type="radio"/> Install Web Server	INFO
<input type="radio"/> Install Wordpress Server	INFO

ITEMS 1 - 4 OF 4

* TYPE

☒ On Success

☐ On Failure

☐ Always

Click on **SELECT** to select the Job Template.

You will have something like this:



Then click on **SAVE** to create the new Workflow. You will be coming back to the main **workflow template** screen:

Deploy a Java App within JBoss

DETAILS PERMISSIONS COMPLETED JOBS SCHEDULES ADD SURVEY WORKFLOW VISUALIZER

* NAME: Deploy a Java App within JBoss DESCRIPTION: Deploying a Java App within JBoss * ORGANIZATION: Workshop

LABELS: OPTIONS: ☐ Enable Concurrent Jobs

EXTRA VARIABLES: [YAML](#) [JSON](#)

```

1 ---
  
```

CANCEL SAVE

Now, click on **SAVE** to save **Deploy a Java App within JBoss** again.

NOTE: Make sure your **JBoss Java Application URL** is not available (**HTTP Status 404 - :8080/ticket-monster**) before starting you workflow! → Access your URL using **:8080 Port!**

Now Click on **Templates** and find the “**Deploy a Java App in JBoss**” Workflow Template. Click  on to start a Workflow Job creation using this workflow template.

Deploy a Java App within JBoss Workflow Template

LAST MODIFIED 1/11/2019 7:12:59 PM by [wuser](#)

This Action will launch the **Install Java App** Job Template, starting the Java App Server installation. You will be redirected to a dynamic Workflow Job Output page:

[JOBS](#) / 183 - DEPLOY A JAVA APP IN JBOSS

DETAILS

STATUS: ● Successful

STARTED: 4/4/2017 14:14:57

FINISHED: 4/4/2017 14:16:06

TEMPLATE: Deploy a Java App in JBoss

LAUNCHED BY: [wuser](#)

EXTRA VARIABLES: [?](#)

```

1 ---
  
```

DEPLOY A JAVA APP IN JBOSS

KEY: — On Success — On Fail — Always P Project Sync I Inventory Sync

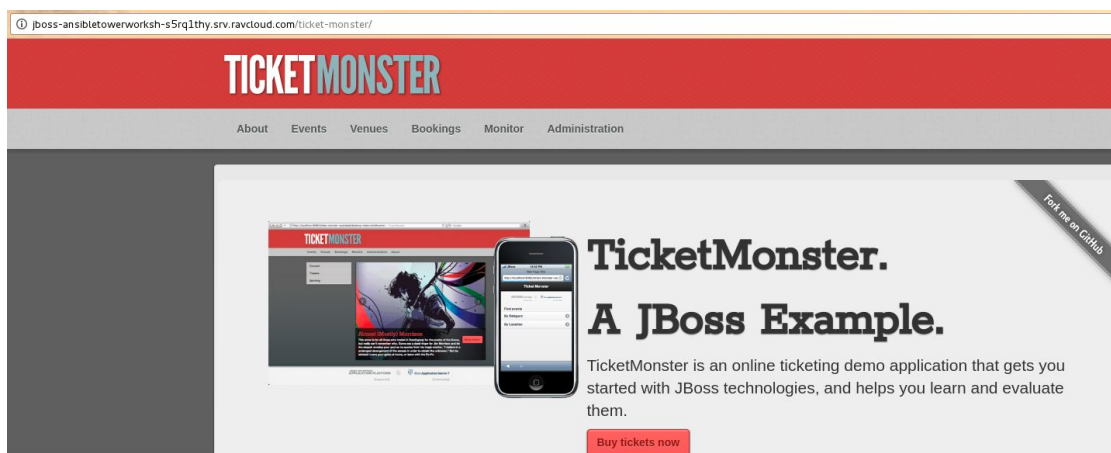
```

graph LR
    InstallJavaA1[Install Java A...] -- blue --> InstallAppSe[Install App Se...]
    InstallAppSe -- red --> InstallJavaA2[Install Java A...]
  
```

If our first Job Template has a **Successful status**, Workflow finishes, but if it has a **Failure status**, workflow will continue until finishes.

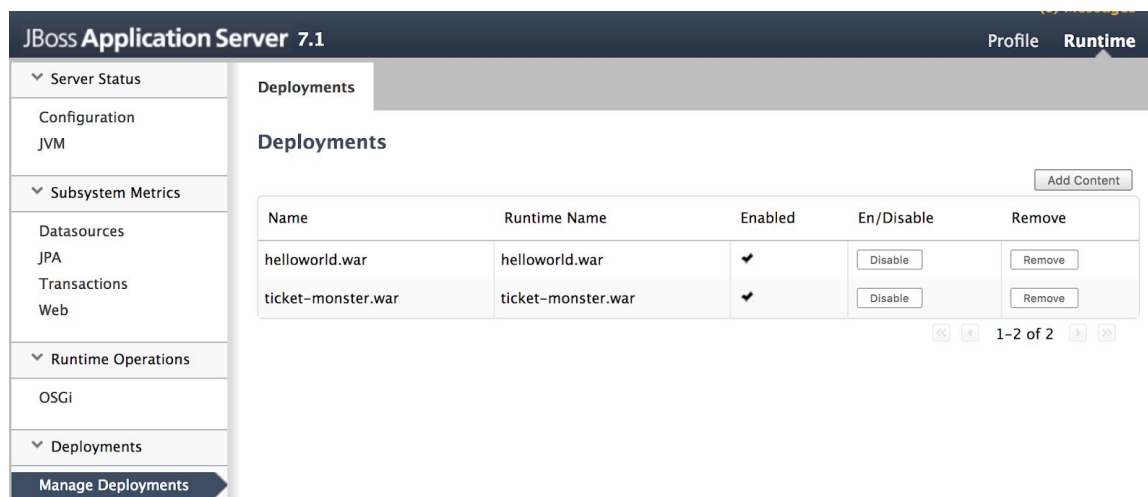
If everything finishes **Ok**, you can access to your **JBoss Java Application URL** (with **:8080/ticket-monster** context root and then try with **:8080/helloworld**)

You can access to **TicketMonster Web Application Server** to buy some tickets:



Just play a moment with your new Java Application!

Now come back to your **Administration Console** link. Then click on **Deployments** → **Manage Deployments**. Refresh page if necessary. You will see your new deployments:



Lab 5: Running Ad-Hoc Commands Using Ansible Tower

Scenario

An **Ad-Hoc command** is something that you might type in to do something really quick, but don't want to save for later. We will use **Ansible Tower** to run some **Ad-Hoc commands**, such as **date**, **last**, **whoami** and so on, into some different servers at the same time.

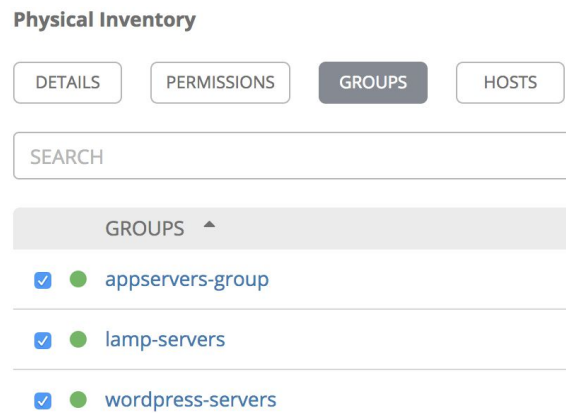
Tasks

Using our servers in **Physical Inventory**, run some commands using different **Ansible Modules**.

Access to **Ansible Tower Portal**, using our new **Workshop User**:

User: wuser / **Password:** r3dh4t

Click on **Inventories** and then click on **Physical Inventory** → **GROUPS**. Then select (**check**) our 3 different Groups: **appservers-group**, **lamp-servers** and **wordpress-servers**:



After that, click on **RUN COMMANDS** button:



5A. Using Command Module:

Within **EXECUTE COMMAND** section, run **date** command as follows:

MODULE: command

MACHINE CREDENTIAL: SSH Server Credentials

ARGUMENTS: date

LIMIT: appservers-group:lamp-servers:wordpress-servers ←(keep it as is)

Click on **LAUNCH** to run the command. Observe the results.

You will observe within **STANDARD OUT** section something like this:

```
webserver.demo.redhat.com | CHANGED | rc=0 >>
Sat Jan 12 01:35:27 UTC 2019

jboss.demo.redhat.com | CHANGED | rc=0 >>
Sat Jan 12 01:35:27 UTC 2019

wordpress.demo.redhat.com | CHANGED | rc=0 >>
Sat Jan 12 01:35:27 UTC 2019
```

NOTE: You can click on **Download Output**  to download the results.

Click on **Inventories** and then click on **Physical Inventory** → **GROUPS**. Select the same Groups and click on **RUN COMMANDS** again.

Within **EXECUTE COMMAND** section, run **last** command as follows:

MODULE: command

MACHINE CREDENTIAL: SSH Server Credentials

ARGUMENTS: last

LIMIT: appservers-group:lamp-servers:wordpress-servers ←(keep it as is)

Click on **LAUNCH** to run the command. Observe the results.

You will observe within **STANDARD OUT** section something like this:

```
jboss.demo.redhat.com | CHANGED | rc=0 >>
root      pts/0      ansible.demo.red Sat Jan 12 01:37   still logged in
root      pts/0      ansible.demo.red Sat Jan 12 01:35 - 01:35   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:19   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)
root      pts/0      ansible.demo.red Sat Jan 12 01:18 - 01:18   (00:00)...
```

5B. Using Yum Module:

Click on **Inventories** and then click on **Physical Inventory** → **GROUPS**. Select the same Groups and click on **RUN COMMANDS** again.

Within **EXECUTE COMMAND** section, using **yum module**, install **telnet** package as follows:

MODULE: yum

MACHINE CREDENTIAL: SSH Server Credentials

ARGUMENTS: name=telnet state=latest

LIMIT: appservers-group:lamp-servers:wordpress-servers ←(keep it as is)

Privilege Escalation: Make sure this is checked.

Click on **LAUNCH** to run the command. Observe the results.

You will observe within **STANDARD OUT** section something like that:

```
SSH password:
jboss.demo.redhat.com | SUCCESS => {
  "changed": true,
  "msg": "",
  "rc": 0,
  "results": [
    "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\n * base: mirror.cogentco.com\n * extras: mirror.netdepot.com\n * updates: mirror.vcu.edu\nResolving Dependencies\n--> Running transaction check\n--> Package telnet.x86_64 1:0.17-60.el7 will be installed\n--> Finished Dependency Resolution\n\nDependencies Resolved\n\n===== Package Arch\nVersion      Repository      Size\nx86_64        1:0.17-60.el7   base           63 k\n\nTransaction Summary\n\nInstall 1 Package\n\nTotal download size: 63 k\nInstalled size: 113 k\n\nDownloading packages:\nRunning transaction check\nRunning transaction test\nTransaction test succeeded\nRunning transaction\n  Installing : 1:telnet-0.17-60.el7.x86_64 1/1 \n\nInstalled:\n  telnet.x86_64 1:0.17-60.el7\n\nComplete!\n"
```

5C. Using User Module:

Click on **Inventories** and then click on **Physical Inventory** → **GROUPS**. Select the same Groups and click on **RUN COMMANDS** again.

Within **EXECUTE COMMAND** section, using **User module**, create **demo** user as follows:

MODULE: user

MACHINE CREDENTIAL: SSH Server Credentials

ARGUMENTS: name=demo comment="Demo User" password=\$1\$ctRQ8kmb\$PMF.2YAjQrdjiDGFuE4uw0 ←(needs to be crypted)

LIMIT: appservers-group:lamp-servers:wordpress-servers ←(keep it as is)

Privilege Escalation: Make sure this is checked.

Click on **LAUNCH** to run the command. Observe the results.

You will observe within **STANDARD OUT** section something like that:

```
SSH password:
webserver.demo.redhat.com | SUCCESS => {
  "changed": true,
  "comment": "Demo User",
  "createhome": true,
  "group": 1001,
  "home": "/home/demo",
  "name": "demo",
  "password": "NOT_LOGGING_PASSWORD",
  "shell": "/bin/bash",
  "state": "present",
  "system": false,
  "uid": 1001
}
```

5D. Using Ping Module:

Click on **Inventories** and then click on **Physical Inventory** → **GROUPS**. Select the same Groups and click on **RUN COMMANDS** again.

Within **EXECUTE COMMAND** section, using **Ping module**, ping hosts in Groups as follows:

MODULE: ping

MACHINE CREDENTIAL: SSH Server Credentials

ARGUMENTS:

LIMIT: appservers-group:lamp-servers:wordpress-servers ←(keep it as is)

Click on **LAUNCH** to run the command. Observe the results.

You will observe within **STANDARD OUT** section something like that:

```
SSH password:
wordpress.demo.redhat.com | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
webserver.demo.redhat.com | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
jboss.demo.redhat.com | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Lab 6: Automating Windows

Ansible knew that the key was to bring the same simple, agentless paradigm to managing Windows, while still feeling native to Windows administrators. Ansible's native **Windows support** uses Windows PowerShell remoting to manage Windows like Windows in the same Ansible agentless way that Ansible manages Linux like Linux.

Plus, with Ansible's easy extensibility, you can write your own modules in **PowerShell** and extend Ansible for whatever other functionality you need. Ansible users have written modules for managing filesystem ACLs, managing Windows Firewall, and managing hostname and domain membership, and more.

In this lab we will be automating many tasks (such as creating users, copying files, configuring services, etc.) within a **Windows 2012 Server**.

Access to Ansible Tower as **wuser** and follow next steps:

6A. Create Windows Credentials:

Navigate within the **Ansible Tower Portal** and click on **Credentials** (within RESOURCES Section). Click on  to add new Windows Credentials as follows:

Name: Windows Credentials
Description: Windows Credentials
Organization: Workshop
Type: Machine
Username: Administrator
Password: r3dh4t

Click on **SAVE** to save the changes.

6B. Create Windows Project:

Click on **Projects** (within RESOURCES Section). Click on  button. Create the new **Windows Project** using next information:


NAME: Windows Project
DESCRIPTION: Windows Project
ORGANIZATION: Workshop
SCM TYPE: Git
SOURCE DETAILS (SCM URL): <https://github.com/leerich/windows-ansible.git>
SCM UPDATE OPTIONS:

SCM UPDATE OPTIONS

- ☐ Clean ?
- ☐ Delete on Update ?
- ☒ Update Revision on Launch ?

Click on **SAVE** to create the new Project.

6C. Create windows-servers Group:

Click on **Inventories** → **Physical Inventory** → **GROUPS** and then click on  to add **windows-servers** group as follows:

NAME: windows-servers
DESCRIPTION: Windows Servers Group
VARIABLES (YAML):


ansible_connection: winrm
ansible_ssh_port: 5986
ansible_winrm_server_cert_validation: ignore

Click on **SAVE** to create the new group.

The screenshot shows the 'WINDOWS-SERVERS' group configuration page in Ansible Tower. At the top, there are tabs for 'DETAILS' and 'NOTIFICATIONS'. Below these, there are three input fields: 'NAME' (containing 'windows-servers'), 'DESCRIPTION' (containing 'windows-servers'), and 'SOURCE' (set to 'Manual'). Below the input fields, there is a section for 'VARIABLES' with radio buttons for 'YAML' (selected) and 'JSON'. A text area contains the following YAML code:

```
1 ---
2 ansible_connection: winrm
3 ansible_ssh_port: 5986
4 ansible_winrm_server_cert_validation: ignore
```

6D. Add a new host to windows-servers Group:

Click on **Inventories** → **Physical Inventory** → **GROUPS** → **windows-servers** and then click on **HOSTS**. Now click on  (select *New Host*) to add one host as follows:

HOST NAME: windows.demo.redhat.com
DESCRIPTION: Windows 2012

Click on **SAVE** to create the new host.

6E. Create a Windows Job Template:

Click on **Templates** and then click on  button (selecting *Job Template*) to create a new Job Template as follows:

NAME: Automating Windows
DESCRIPTION: Automating Windows
JOB TYPE: run
INVENTORY: Physical Inventory
PROJECT: Windows Project
PLAYBOOK: tower-ansible-automating-windows.yml
MACHINE CREDENTIAL: Windows Credentials
LIMIT: windows-servers
VERBOSITY: 0 (Normal)

Click on **SAVE** to create the new Job Template

6F. First, verify that our Windows 2012 has not been yet automated:

Access to **Windows 2012** via RDP:

Username: Administrator / **Domain:** local / **Password:** r3dh4t

TASK: Verify that **Telnet Client** and **IIS Web-Server** has **not** been yet installed:

- Click on **Server Manager** → **Add roles and features**.
- Then click on **Server Selection** and then click on **Server Roles**. Verify that **Web Server IIS** has **not** been yet installed.
- Then click on **Features** and verify that **Telnet Client** has **not** been yet installed


TASK: Verify that **Ansible User** has **not** been yet created:

- Click on **Control Panel** → **User Accounts** → **User Accounts** → **Manage Another Account**. Verify that **Ansible User** has **not** been yet created

TASK: Verify that **nerd.jpg** image does **not** exist:

- Click on **File Explorer** and then click on **Downloads**. Verify that **nerd.jpg** image does **not** exist.

6G. Run the Windows Job Template:

Click on **Templates** and find the “**Automating Windows**” Job Template. Click  on **Run** to start a job creation using this template.

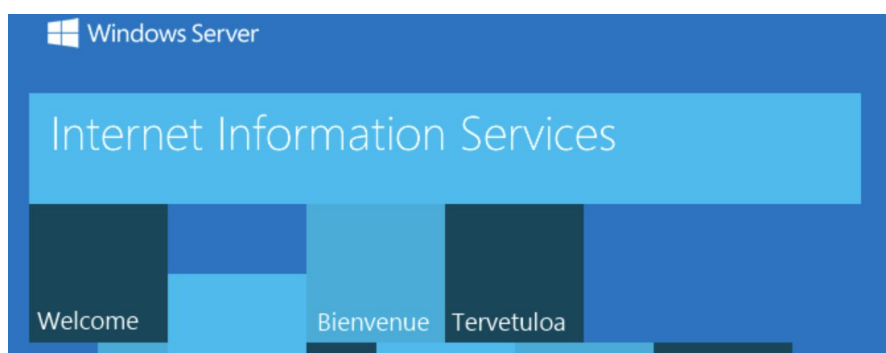
Automating Windows Job Template	
INVENTORY	Physical Inventory
PROJECT	Windows Project
LAST MODIFIED	1/11/2019 8:09:13 PM by wuser

This Action **will automate** some tasks on **Windows 2012**, such as:

- Add a Windows Admin User (Ansible User)
- Install Software (Telnet and IIS Server)
- Copy a JPG archive (neerdd.jpg as nerd.jpg)
- Run a PowerShell script (Hello World)
- Run some Windows commands (ipconfig)
- Check Status of file win.in

If your **Job Template** ran **successfully**, verify again steps in **6F** section, but now you will find that **Windows 2012** has been automated!

Access to **Windows 2012** URL but now via **Web Browser**, just to make sure **Web Server IIS** is **Up** and **Running**!



Now just take a look at the Ansible Playbook:

<https://github.com/leerich/windows-ansible/blob/master/tower-ansible-automating-windows.yml>

Lab 7: Automating Network Devices


7a. Create VyOS Credentials:

Navigate within the **Ansible Tower Portal** and click on **Credentials** (within RESOURCES Section). Click on  to add new Network Credentials as follows:

NAME: VyOS Credentials
DESCRIPTION: VyOS Credentials
ORGANIZATION: Workshop
TYPE: Network
USERNAME: vyos
PASSWORD: r3dh4t

Click on **SAVE** to save the changes.

7B. Create VyOS Project:

Click on **Projects** (within RESOURCES Section). Click on  button. Create the new **VyOS Project** using next information:

NAME: VyOS Project
DESCRIPTION: a VyOS Project
ORGANIZATION: Workshop
SCM TYPE: Git
SCM_URL: <https://github.com/leerich/vyos-ansible>
SCM UPDATE OPTIONS:

SCM UPDATE OPTIONS

- ☐ Clean ?
- ☐ Delete on Update ?
- ☒ Update Revision on Launch ?

Click on **SAVE** to create the new Project.


7C. Create vyos-servers group:

Click on **Inventories** → **Physical Inventory** → **GROUPS**. Now click on  to add **vyos-servers** group as follows:

NAME: vyos-servers
DESCRIPTION: VyOS Servers Group

Click on **SAVE** to create the new group.

7D. Add a new host to vyos-servers group:

Click on **Inventories** → **Physical Inventory** → **GROUPS** → **vyos-servers** and then click on **HOSTS**. Now click on  button (selecting *New Host*) to add one host as follows:


HOST NAME: vyos.demo.redhat.com
DESCRIPTION: VyOS Network Appliance

Click on **SAVE** to create the new host.

7E. Create a VyOs Job Template:

Click on **Templates** and then click on  button (selecting *Job Template*) to create a new Job Template as follows:


NAME: Automating VyOS
DESCRIPTION: Automating VyOS
JOB TYPE: run
INVENTORY: Physical Inventory
PROJECT: VyOS Project
PLAYBOOK: tower-ansible-automating-vyos.yml
MACHINE CREDENTIAL: SSH Server Credentials
NETWORK CREDENTIAL: VyOS Credentials
LIMIT: vyos-servers
VERBOSITY: 0 (Normal)



TIP: Selecting Machine Credential: just click on  icon within **CREDENTIAL** field:


CREDENTIAL  ☐ PROMPT ON LAUNCH





Select **Machine** as **CREDENTIAL TYPE** and now select **SSH Server Credentials**:

CREDENTIALS 

SELECTED:  SSH Server Credentials 

CREDENTIAL TYPE: Machine 

SEARCH  KEY

NAME 


☒ SSH Server Credentials


☐ Windows Credentials




ITEMS 1 - 2

CANCEL SELECT

Finally click on **SELECT** button.

TIP: Selecting Network Credential: just click again on  icon within **CREDENTIAL** field:

CREDENTIAL  ☐ PROMPT ON LAUNCH

  SSH Server Credentials 

Select **Network** as **CREDENTIAL TYPE** and now select **VyOS Credentials**:

CREDENTIALS ✕

SELECTED: 🔍 SSH Server Credentials ✕ 👤 VyOS Credentials ✕

CREDENTIAL TYPE: Network ▼

SEARCH 🔍 KEY

NAME ▲

☒ VyOS Credentials

ITEMS 1 - 1

CANCEL SELECT

Finally click on **SELECT** button.


You will have your CREDENTIAL field as follows:

CREDENTIAL ? ☐ PROMPT ON LAUNCH

🔍 🔍 SSH Server Credentials ✕ 👤 VyOS Credentials ✕

Click on **SAVE** to create the new Job Template

7F. Run the “Automating VyOS” Job Template:

Click on **Templates** and find the “**Automating VyOS**” Job Template. Click  on **Run** to start a job creation using this template.

This Action **will automate** some tasks on **VyOS Network Virtual Appliance**, providing a **NAT Gateway** for this device with two interfaces, as described within the VyOS Quick Start Guide:

https://wiki.vyos.net/wiki/User_Guide#Quick_Start_Guide

If your **Job Template** ran **successfully**, just click on some tasks to verify that **your VyOS has been automated!**

Now take a look at the Ansible Playbook:

<https://github.com/leerich/vyos-ansible/blob/master/tower-ansible-automating-vyos.yml>

If you finished the Workshop, please answer this Survey:

Select **Ansible Tower Automation** as **Test Drive Name**:

https://docs.google.com/forms/d/e/1FAIpQLSdauHtguNMYICRE5x1nrEOY11ASfDNnptSEqLZi_TCsNgb2g/viewform