Ansible Tower

Managing Ansible
Automation Platform

Introduction

- This course provides an introduction to Ansible Tower AKA Ansible Automation Platform
- For licensing reasons, the AWX upstream project is used for demos
- Knowledge about Ansible Fundamentals is required
- This course is hands on, if you haven't done it yet, set up the following minimal lab requirement
  - 1 VM with 8 GB RAM and 4 vCPU's and 40 GB disk space
  - 2 VMs with 1 GB RAM, 1 vCPU and 20GB disk space

Configuration Requirements

- The Ansible Tower (AWX) machine
  - 8 (4) GB RAM
  - 4 (2) vCPUs
  - 40 (20) GB disk space
  - RHEL/CentOS latest version
- The Managed machines
  - 1 GB RAM
  - 1 vCPU
  - 10 GB disk space

Poll Question 1

- How would you rate your own Ansible knowledge
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5

Poll Question 2

- Did you attend any of my Ansible classes?
  - no
  - Ansible in 4 hours
  - Ansible in 3 Weeks
  - RHCE EX/294
  - Attended another course
  - No need for class

Poll Question 3

- Where are you from?
  - North/Central America
  - South America
  - Netherlands
  - India
  - Asia
  - Europe
  - Australia/Pacific
  - Africa

Ansible Tower

1. Understanding Ansible Tower / Automation Controller

Understanding Ansible Tower / AAC

- Ansible Tower (now known as Ansible Automation Controller) provides a web based interface that brings Ansible to large environments, offering several features
  - workflow design
  - activity logging
  - scalability
  - notifications
  - scheduling
  - remote execution
  - REST API and Tower CLI tool

Ansible Automation Platform

- Red Hat Automation Platform is Ansible Engine + Tower rebranded, connecting to Automation Hub
- Ansible Tower Automation Hub was announced in Oct. 2020
- Automation Platform integrates Ansible Tower with OpenShift to provide access to Ansible Content Collections (=marketing blah)
- Automation Hub content collections are subscriber only (=$$$)
- Ansible Tower is now known as Ansible Automation Controller

Ansible Tower versus AWX

- Ansible Automation Platform is the Red Hat licensed web-based Ansible management interface
  - The free developers.redhat.com license allows management of 16 nodes
  - The demo version can be used for free for a limited number of days
- AWX is the unlicensed alternative
  - Rather complex installation on top of OpenShift / Kubernetes
  - Scripted installation is provided in this course

Ansible Tower

2. Setting up Ansible Automation Platform or AWX

Installing AWX

- AWX is installed on Docker (deprecated) or as an operator on top of Kubernetes (recommended)
- To install on Kubernetes, use minikube
- In this course, a script is provided to set up Minikube easily on top of Ubuntu Desktop
- Make sure the Ubuntu Desktop VM has 8GB RAM and 4 vCPUs

Installing AWX

- **sudo apt install git vim -y**
- **git clone https://github.com/sandervanvugt/tower**
- **cd tower; ./minikube-docker-setup.sh**
- **minikube start --cpus=4 --memory=6g --addons=ingress --vm-driver=docker**
- **curl -s "https://raw.githubusercontent.com/kubernetes-sigs/kustomize/master/hack/install_kustomize.sh" | bash**
- **sudo mv kustomize /usr/local/bin/**
- Check version number by reading kustomization.yaml and then run **kustomize build . | kubectl apply -f -**
- Verify, using **kubectl get pods -n awx**
- **kubectl config set-context --current --namespace=awx**

Installing AWX

- Verify contents of **awx-demo.yaml** in course git repo
- Modify the **kustomization.yaml** file to add the following extra line below the **resources:**

```
...
resources:
  - github.com/ansible/awx-operator/config/default?ref=1.1.3
  - awx-demo.yaml
...
```

- Run **kustomize build . | kubectl apply -f -** again
- Type **kubectl get pods,svc** and verify that you have the AWX pods and services running (will take a few minutes)

Installing AWX

- Use the following to get the minikube service URL: **minikube service -n awx awx-demo-service --url**
- Get the AWX admin password using **kubectl get secret awx-demo-admin-password -o jsonpath="{.data.password}" | base64 --decode**
- Copy the string that is printed, it is your admin password

Setting up Ansible Automation Controller

- Free AAC evaluation licenses are available from RedHat – see tower.ansible.com
  - Download the tar-ball
  - Request an evaluation license or use your RedHat developer subscription (developers.redhat.com)

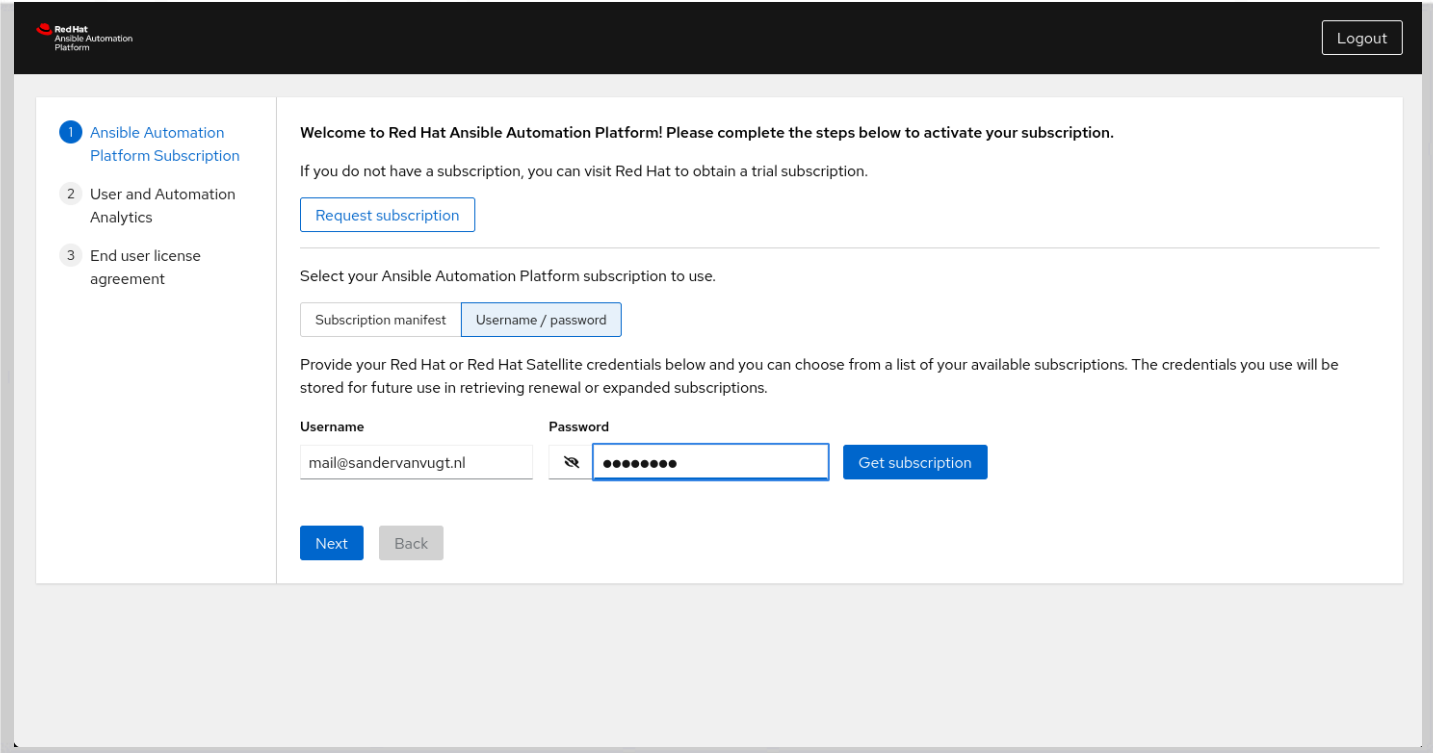
Configuration Requirements

- The Ansible Tower (AWX) machine
  - 8 GB RAM
  - 4 vCPUs
  - 40 GB disk space
  - Red Hat family 8 or later

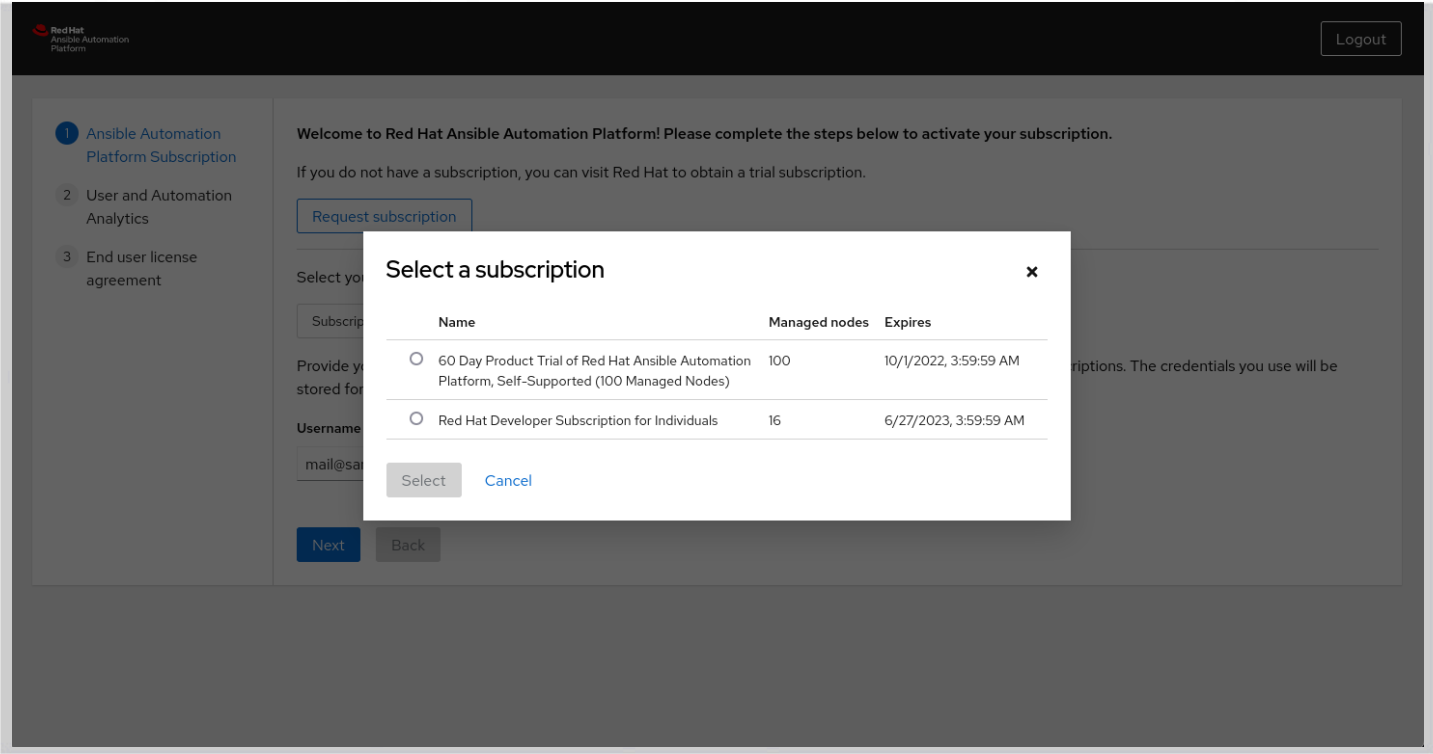
Installing AAP Evalution Version

- Get the AAC software from access.redhat.com
- Extract the evaluation version tarball
- Ensure your aac host has a fixed IP address and host name resolving is set up
- Create SSH keys and enable (root) SSH key based login to the server where you want to install AAC (**ssh-copy-id**)
- Modify all password fields in the inventory file and add an automationcontroller name or IP address
- **sudo ./setup.sh**
- Will take about 15 minutes to complete
- Log in to the web UI: https://localhost
- login in with your Red Hat credentials or provide the credentials file

Selecting a Subscription



Selecting a Subscription



Ansible Tower

3. Understanding a Tower Managed Environment

Managing Machines with Tower

- To reach out to managed machines with tower, things are not really different from managing machines with Ansible Engine from the command line
- Identifying the managed machines
  - On the tower host, setup /etc/hosts name resolving (or DNS)
  - On minikube based AWX, use **minikube ssh** and edit /etc/hosts there
- On the managed machines
  - Ensure sshd is running and accepts incoming connections (firewall)
  - Need a user account with sudo privileges
  - Need to set up password / SSH keys

Understanding Core Components

- Organization: a collection of managed devices
- Users: administrative users that can be granted access to specific tasks
- Inventories: managed servers. Can be created statically or dynamically
  - Click Settings > License and check Host Remaining
- Credentials: credentials that are used to log in to a managed machine. Think of user with sudo privileges
- Project: a collection of playbooks obtained from a certain location (such as Github)
- Template: the job definition with all of its parameters. Must be launched or scheduled

Ansible Tower

3. Running a First Project with Tower

Required steps

- (optional) Define an organization
- Create an Inventory
- Configure Credentials
- Set up a Project
- Define a Job Template
- Run the Job

4: Advanced Tower Usage

4.1 Working with Users and Teams

- The git repo for tower is https://github.com/sandervanvugt/tower

Understanding Tower Users

- Tower users are used by people that need access to the Tower interface
- Tower users are used with Role Based Access Control (RBAC) to grant users access to specific roles
- Roles can be assigned to individual users or teams
- Depending on the RBAC settings granted to an Ansible user, the user will be able to view, use, change or remote Ansible objects

Understanding Organizations

- An organization is a collection of teams, projects and inventories
- Organizations make sense in very large deployments, as they allow users and teams to be configured with access to specific sets of resources
- Ansible Tower comes with one organization, named Default
- According to the Ansible tower usage license, additional organizations may be created
- Users exist at the Ansible Tower level and can have roles in multiple organizations

Understanding User Types

- By default, there are three types of users
  - System Administrator has read/write access to the entire tower installation
  - System Auditor has read-only access to the entire installation
  - Normal user starts with minimal access, and must be provided with access by adding roles to the user

Understanding Teams

- A team is a group of users
- Teams exist at an organization level
- System Administrator users can assign the team roles on resources in different organizations
- Teams cannot get roles on the organization object

Organization Roles

- Different roles are available and can be connected to the users
  - Organizational Admin
  - Project Admin
  - Inventory Admin
  - Credential Admin
  - Notification Admin
  - Workflow Admin
  - Job Template Admin
  - Auditor
  - Member
  - Read
  - Execute
- Roles are assigned with an organization scope or a project scope

4: Advanced Tower Usage

4.2 Understanding Execution Environments

Understanding Execution Environments

- An execution environment is a container image that has all dependencies to run a job
- Without execution environments, you may have to install software dependencies on all nodes that run specific jobs
- By using execution environments this is no longer required
- In this case, the execution environment serves as an Ansible control node
- To create an execution environment, use **ansible-builder**, to use an execution environment, use **ansible-runner** (or AWX)
- To install **ansible-builder**, use **pip install ansible-builder**

Building an Execution Environment

- The execution environment should contain multiple components
  - Ansible
  - Ansible Runner
  - Ansible Collections
  - Python and optional other system dependencies
- As an execution environment is a container image, a container engine must be available as well
  - Podman as well as Docker are supported
  - Use the **--container-runtime** option to specify which engine to use
- To build an execution environment, use a dedicated machine that has Ansible as well as the container engine installed
- Having detailed knowledge about execution environments is not required for working with AWX

4: Advanced Tower Usage

4.3 Creating Job Template Surveys

Understanding Job Templates

- **vars_prompt** from Ansible Engine is not supported in Tower
- An alternative is provided by Job Template surveys
- On a job, use **EXTRA VARIABLES** to define variables on the job
- Select **PROMPT ON LAUNCH** to prompt for variable values while launching the job template
- These options make sense for a skilled Ansible user
- To make it easy for anyone to provide variables, a Job Template Survey can be used
- Job Template Surveys prompt for variables when the job is started
- Variables from a survey have the highest priority

Defining Survey Answer Types

- In surveys the variable types can be defined as one of the following
  - Text: this is text on a single line
  - Textarea: text on multiple lines
  - Password: treated as sensitive information
  - Multiple choise (single select): a list of options where one can be selected
  - Multiple choise (multiple select): a list of options where one or more can be selected
  - Integer: an integer number
  - Float: a foating-point decimal
- While creating surveys, a default answer can be specified
- Questions can also be marked as required: an answer must be provided

Creating Surveys

- A survey cannot be created during creation of the template
- Create the Job Template first, save it, and next add the Survey to it

4: Advanced Tower Usage

4.4 Using Workflow

Understanding Workflow

- A Workflow Job Template is used to run multiple job templates in a sequence
- Using workflows makes it easier to work with playbooks (job templates) that are provided from different teams
- In a Workflow complex relations between jobs can be defined, where the next job is started depending on the result of the previous job
  - On success
  - On failure
  - Always
- Before creating a Workflow, a Workflow Job Template has to be defined
- After defining the Workflow Job Template, the Workflow Visualizer is used to define the actual workflow

4: Advanced Tower Usage

4.5 Scheduling Jobs

Understanding Scheduled Jobs

- Scheduled Jobs allow you to run Job Templates on a cron-like schedule
- After Job execution, results can be consulted in Completed Jobs
- Also, notification templates can be configured to send information about job success or failure in an automated way
- To use notifications, you'll first create the notification template and next add it to a job template for execution

4: Advanced Tower Usage

4.6 Importing Static Inventories

Importing Static Inventories

- Static inventories can easily be imported if they are in Git or any other external system
- Local static inventory files are imported with the **awx-manage** cli utility on the tower server:
  - **awx-manage inventory_import --source=/root/myinventory --inventory-name="myinventory"**

4: Advanced Tower Usage

4.7 Creating and Updating Dynamic Inventories

Dynamic Inventory and Tower

- Tower comes with a set of dynamic inventories to connect to common external environments like AWS, Azure, OpenStack and More
- Custom inventory scripts can also be used

4: Advanced Tower Usage

4.8 Using Smart Inventories

Understanding Smart Inventory

- Smart inventory is dynamically created from other inventory sources by using a filter
- The filter is using ansible facts that are discovered from the different hosts
- Smart inventory uses fact cache, so you'll have to create a job template with the use fact cache option and run it periodically

Understanding Smart Inventory Filters

- A filter may look like **ansible_facts.ansible_distribution:RedHat**
- In this filter, pre Ansible 2.5 notation is used (modern notation is not yet supported)
- **ansible_facts** indicates the filter applies to ansible_facts, and not a host name or something else
- Also notice there is no white space between the colon and the value you want to match

4: Advanced Tower Usage

4.9 Using Vault in Tower

Tower and Vault

- To use Vault encrypted files, you need to create a vault credential
- Job templates must be configured with both the vault credential as the machine credential to run the job