Ansible Tower

Managing Ansible Tower

Introduction

- This course provides an introduction to Ansible Tower AKA Ansible Automation Platform
- 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

Expectations

- As of Ansible Automation Platform 2.5, it uses a multi-server distributed installation with heavey resource requirements.
- There is an all-in-one container based installation which doesn't work well.
- For that reason, in this course you'll learn how to install AWX on top of Kubernetes.

AWX Configuration Requirements

- The Ansible Tower (AWX) machine
  - 8 GB RAM
  - 4 vCPUs
  - 40 GB disk space
  - Recent Ubuntu because of Kubernetes requirement
- 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

Ansible Automation Platform

Ansible Automation Platform consists of different components:

- Ansible Automation Controller: a web-based platform that makes working with Ansible easier in large-scale environments
- Event Driven Ansible Controller: triggers playbooks on specific events
- Ansible Automation Hub: the integrated platform to manage Ansible Content Collections
- Ansible Lightspeed AI (requires separate subscription)

Understanding Ansible Automation Controller

- Ansible Automation Controller (AAC) 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 AI (additional subscription required)
  - Event Driven Ansible

AAP versus AWX

- AAP is the Red Hat licensed web-based Ansible management interface
  - The free developers.redhat.com license allows management of 16 nodes
  - Distributed installation on containers or OpenShift
- AWX is the open source upstream (?) alternative
  - Container-based installation on top of Kubernetes or OpenShift

Ansible Tower

2. Setting up AAP

Setup Options

- AAP can be installed in 3 ways:
  - RPM based on virtual machines
  - In OpenShift
  - Container based
- All installations require different node roles, with 16 GiB and 4 vCPUs each:
  - Platform gateway
  - Control nodes
  - Execution nodes
  - Hop nodes
  - Automation hub
  - Database
  - EDA controller

Containerized Setup Bundle

- Containerized Setup Bundle is the only feasible installation for demo environments
- It is available through developers.redhat.com, scroll down a bit to find the appropriate file
- System Requirements:
  - 16GiB RAM
  - 4 CPUs
  - 60GiB disk

Preparing your host

- **sudo subscription-manager register**
- **sudo hostnamectl set-hostname yourhostname**
- **sudo dnf install -y ansible-core**
- Download the latest installer.tar file for "Ansible Automation Platform 2.5 Containerized Setup"
  - Use "Ansible Automation Platform 2.5 Containerized Setup Bundle" for offline installations
- **tar xzvf ansible-automation[Tab]**
- **podman login registry.access.redhat.com**
- **podman login registry.redhat.io**

Using an Inventory File

- The installation is controlled by using an inventory file
- Inventory files define the hosts and containers created, variables and more
- Example inventory files are provided:
  - inventory is for default distributed enterprise installation
  - inventory-growth is the all-in-one installation that you'll need
- Replace the < ... > placeholders with your local variables
- Check the README.md file in the installation directory for additional information

Starting the Installation

- As **sudo** prvileges are required during installation, use a dummy sudo command to generate the sudo authentication token (**sudo ls -l /root**)
- **ansible-playbook -i inventory-growth ansible.containerized_installer.install**
- Go have a break, this takes at least 15 minutes
- Access the platform UI using https://aap.example.com:443 and login as admin with the password defined

Setting up AWX

AWX Setup Requirements

To set up AWX according to the instructions found here, make sure you have the following

- One Ubuntu Workstation using the most recent Ubuntu LTS version
  - 8 GB RAM
  - 4 vCPUs
  - 40 GB disk space
- At least one server that is in a manageable state

Demo: 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/**
- Change 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**

Demo: 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** (where ref= refers to the current version)**:**

```
...
resources:
  - github.com/ansible/awx-operator/config/default?ref=2.19.1
  - 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)

Demo: Troubleshooting

- March 2025 the following troubleshooting was required, as the awx-demo-web pod was not starting successfully because of a problem in the rsync container that it runs:
- **kubectl edit deploy awx-demo-web**
  - Change image: quay,.io/ansible/rsyslog:nn.n.n to rsyslog:latestgg
- Verify that kubectl get pods shows a running awx-demo-web Pod

Demo: 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
- Use **minikube ssh** followed by **sudo vim /etc/hosts** to set up host name resolution for the managed hosts
- Make sure the managed hosts are in a manageable state
  - They have an "ansible" user with sudo privileges that can use SSH to log in

Ansible Tower

4. 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 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

Lesson 12: Advanced Tower Usage

12.1 Working with Users and Teams

Notice

- These slides are from "Ansible from basics to guru", available on this platform

- 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

Demo

- Createusers and roles and assigning users to teams
- How these teams can be granted privileges to other tower objects

Lesson 12: Advanced Tower Usage

12.2 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

Lesson 12: Advanced Tower Usage

12.3 Configuring Notifications

Lesson 12: Advanced Tower Usage

12.4 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

demo

- Templates > New > create workflow template > Save
- Make sure we have a complete environment, including two templates, credentials and an inventory
  - add a survey on the workflow template (NOT job template) to query variable "something"
  - installsomething -> removesomething !-> onfailure
  - use ansibleinthreeweeks/tower git repo
- Set permissions
- Set notifications!

Lesson 12: Advanced Tower Usage

12.5 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

Demo

- templates > select any job > schedule
- schedules for an overview
- notifications > create notifications > show type > select email
  - host = localhost
  - job template > notifications > switch on

Lesson 11: Advanced Tower Usage

12.6 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"**

Demo

- ensure there is a project that is based on git repo svv/tower (should already be there), select " update on project update"
- create inventory, save it, go to sources > add source > sourced from a project > select inventory file. select "update on project update" and save. Check that the hosts are listed
- From the MBP, update the tower/inventory file, git push
- select projects > mygitproject and update it
- select inventories and check on hosts in the just added inventory

Lesson 12: Advanced Tower Usage

12.9 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

Demo

- create a vault credential
- create a job template based on github/tower/vaulted.yam
    - configure it with a machine credential as well as a vault credentials and run

Lesson 12: Advanced Tower Usage

12.10 Using the Tower API

Understanding Tower API

- Tower provides a REST API that allows controlling tower from playbooks
- This allows clients to perform actions using standard HTTP methods, such as GET, POST, PUT and DELETE
  - **curl -X GET https://towerbridge.example.com/api/ -k**
- The API is browsable, which means you can access it from a graphical browser and investigate the different elements by clicking on them
  - **https://towerbridge.example.com/api/**
- When using **curl**, pipe through **json_pp** for readable ("pretty print") output
  - **dnf install -y perl-JSON-PP**
  - **curl -X GET https://towerbridge.example.com/api/v2/ -ks | json_pp**

Understanding Tower API

- Some information requires proper authentication:
  - **curl -X GET https://towerbridge.example.com/api/v2/activity_stream/ -k**
  - **curl -X GET --user admin:password https://towerbridge.example.com/api/v2/activity_stream/ -k**
- When using a graphical browser, use the question mark icon to get more information about resources

Demo: Launching Job Templates using API

- Start by getting information about Jobs:
  - curl -X GET --user admin:password https://towerbridge.example.com/api/v2/job_templates/ -ks | json_pp | grep name
- Use POST and launch to launch any job
  - curl -X POST --user admin:password https://towerbridge.example.com/api/v2/job_templates/"updatecache"/launch/ -ks | json_pp
  - In the output, look for the job ID
- Use the Job ID to get a job status update:
  - curl -X GET --user admin:password https://towerbridge.example.com/api/v2/jobs/25/ -ks | json_pp

Using the API to launch Jobs from a Playbook

- The **uri** module can be used to run a job using the API
- See launchtowerjob.yaml in the https://github.com/sandervanvugt/tower git repository