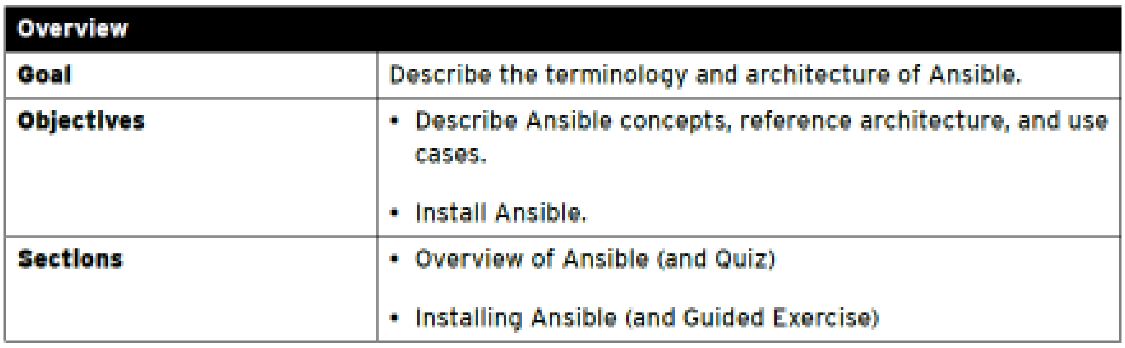
INTRODUCING ANSIBLE



Overview	
Goal	Describe the terminology and architecture of Ansible.
Objectives	• Describe Ansible concepts, reference architecture, and use cases. • Install Ansible.
Sections	• Overview of Ansible (and Quiz) • Installing Ansible (and Guided Exercise)

What is Ansible?

Ansible is an open source automation platform. It's a *simple automation language* that can perfectly describe an IT application infrastructure in Ansible Playbooks. It's also an *automation engine* that runs **Ansible Playbooks.**

Ansible can manage powerful automation tasks, and can adapt to many different workflows and environments. At the same time, new users of Ansible can very quickly use it to become productive.

Ansible Is Simple

Ansible Playbooks provide human-readable automation. This means that your playbooks are automation tools that are also easy for humans to read, comprehend, and change. No special coding skills are required to write them. Playbooks execute tasks in order. The simplicity of playbook design makes them usable by every team. This allows people new to Ansible to get productive quickly.

Ansible Is Powerful

You can use Ansible to deploy applications, for configuration management, for workflow automation, and for network automation. Ansible can be used to orchestrate the entire application life cycle.

Ansible Is Agentless

Ansible is built around an agentless architecture. Typically, Ansible connects to the hosts it manages using OpenSSH or WinRM and runs tasks, often (but not always) by pushing out small programs called Ansible modules to those hosts. These programs are used to put the system in a specific desired state. Any modules pushed are removed when Ansible is finished with its tasks. It is possible to start using Ansible almost immediately, because no special agents need to be approved for use and then deployed to the managed hosts. Because there are no agents and no additional custom security infrastructure, Ansible is more efficient and more secure than other alternatives.

Ansible has a number of important strengths:

• Cross platform support: Ansible provides agentless support for Linux, Windows, UNIX, and network devices, in physical, virtual, cloud, and container environments.

• Human-readable automation: Ansible Playbooks, written as YAML text files, are easy to read and help ensure that everyone understands what they will do

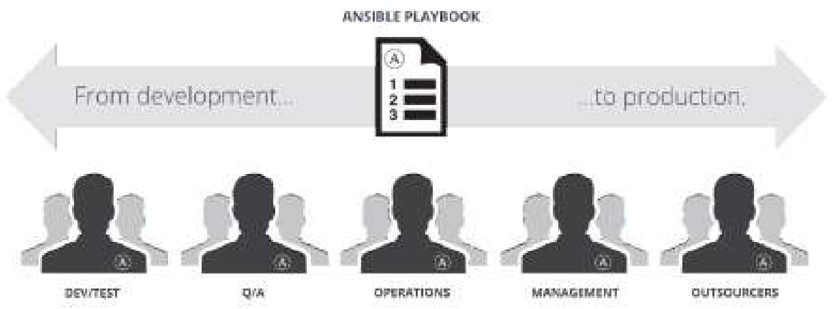
• Perfect description of applications: Every change can be made by Ansible Playbooks, and every aspect of your application environment can be described and documented.

• Easy to manage in version control: Ansible Playbooks and projects are plain text. They can be treated like source code and placed in your existing version control system.

• Support for dynamic inventories: The list of machines that Ansible manages can be dynamically updated from external sources in order to capture the correct, current list of all managed servers all the time, regardless of infrastructure or location.

• Orchestration that integrates easily with other systems: HP SA, Puppet, Jenkins, Red Hat Satellite, and other systems that exist in your environment can be leveraged and integrated into your Ansible workflow.

Ansible: The Language of DevOps



Communication is the key to DevOps.
Ansible is the first automation language that can be read and written across IT. It is also the only automation engine that can automate the application life cycle and continuous delivery pipeline from start to finish.

Ansible Concepts and Architecture

There are two types of machines in the Ansible architecture: the control nodes and managed hosts. Ansible is installed and run from a control node, and this machine also has copies of your Ansible project files. A control node could be an administrator's laptop, a system shared by a number of administrators, or a server running Ansible Tower.
Managed hosts are listed in an inventory, which also organizes those systems into groups for easier collective management. The inventory can be defined in a static text file, or dynamically determined by scripts that get information from external sources.
Instead of writing complex scripts, Ansible users create high-level plays to ensure a host or group of hosts are in a particular state. A play performs a series of tasks on the host or hosts, in the order specified by the play. These plays are expressed in YAML format in a text file. A file that contains one or more plays is called a playbook.
Each task runs a module, a small piece of code (written in Python, PowerShell, or some other language), with specific arguments. Each module is essentially a tool in your toolkit. Ansible ships with hundreds of useful modules that can perform a wide variety of automation tasks. They can act on system files, install software, or make API calls.
When used in a task, a module generally ensures that some particular thing about the machine is in a particular state. For example, a task using one module may ensure that a file exists and has particular permissions and contents, while a task using a different module may make certain that a particular file system is mounted. If the system is not in that state, the task should put it in that state. If the system is already in that state, it should do nothing. If a task fails, Ansible's default behaviour is to abort the rest of the playbook for the hosts that had a failure. Tasks, plays, and playbooks should be idempotent. This means that you should be able to run a playbook on the same hosts multiple times safely, and when your systems are in the correct state the playbook should make no changes when run.

References

Ansible
https://www.ansible.com
How Ansible Works
https://www.ansible.com/how-ansible-works

Installing Ansible

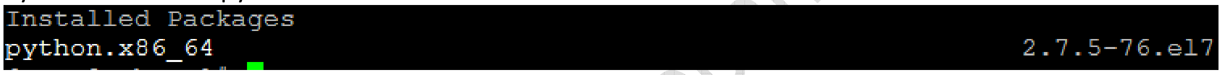
Control Nodes

Ansible is simple to install. The Ansible software only needs to be installed on the control node (or nodes) from which Ansible will be run. Hosts that are managed by Ansible do not need to have Ansible installed. This installation involves relatively few steps and has minimal requirements.
The control node should be a Linux or UNIX system. Microsoft Windows is not supported as a control node, although Windows systems can be managed hosts. Python 2 (version 2.6 or later) needs to be installed on the control node. (Use of Python 3 with Ansible is still in technology preview and should not be used in production.) To see whether the appropriate version of Python is installed on a Red Hat Enterprise Linux system, use the **yum** command.

Control Node: robo
Managed node: robo2

Install RHEL 7 epel repo on both node
#rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

#yum list installed python



```
Installed Packages
python.x86_64                                                    2.7.5-76.el7
```

References

ansible(1) man page
Installation — Ansible Documentation
http://docs.ansible.com/ansible/intro_installation.html
Windows Support
http://docs.ansible.com/ansible/intro_windows.html
Networking Support
http://docs.ansible.com/ansible/intro_networking.html

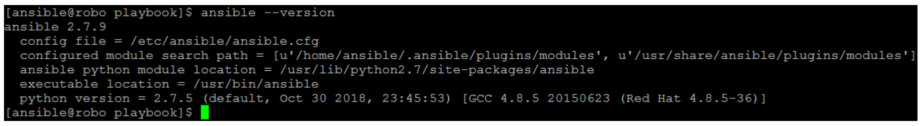
Guided Exercise: Installing Ansible

:-Install ansible on control node (robo)
#yum list installed python
#yum install -y ansible

Configuring Ansible

The behaviour of an Ansible installation can be customized by modifying settings in the Ansible configuration file. Ansible chooses its configuration file from one of several possible locations on the control node.
Using **/etc/ansible/ansible.cfg**
The *ansible* package provides a base configuration file located at **/etc/ansible/ansible.cfg**. This file is used if no other configuration file is found.

#ansible –version



```
[ansible@robo playbook]$ ansible --version
ansible 2.7.9
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ansible/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Oct 30 2018, 23:45:53) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]
[ansible@robo playbook]$
```

Managing Settings in the Configuration File

<u>Make sure of below settings in ansible configuration file.</u>

```
#grep -v "#" /etc/ansible/ansible.cfg
[defaults]
inventory      = /home/ansible/playbook/inventory
fact_path = /etc/ansible/facts.d/
forks          = 5
poll_interval  = 90
roles_path    = /home/ansible/playbook/roles
host_key_checking = False
deprecation_warnings=False

[privilege_escalation]
become=no
become_method=sudo
become_user=root
become_ask_pass=true

[colors]
highlight = white
verbose = blue
warn = bright purple
error = red
debug = dark gray
deprecate = purple
skip = cyan
unreachable = red
ok = green
changed = yellow
diff_add = green
diff_remove = red
diff_lines = cyan
```

Create User “ansible” with sudo privilege on both node

```
#useradd ansible && passwd ansible
# usermod -G wheel ansible
# ssh-keygen
# ssh-copy-id ansible@robo2   → Run from control node
# ssh-copy-id ansible@robo    → Run from managed node
```

ansible(1), **ssh-keygen**(1), and **ssh-copy-id**(1) man pages
Configuration file: Ansible Documentation
http://docs.ansible.com/ansible/intro_configuration.html