Ansible

- NEOPOLEAN

INTRODUCTION

- Ansible is an IT DevOps tool that automates provisioning, configuration management, application deployment, intra-service orchestration, continuous delivery, and many other IT processes.

- Ansible is designed for multi-tier deployments. Instead of managing systems individually, it models your IT infrastructure by describing the inter-relationships among all your systems.

- Ansible is an open source IT configuration management, deployment, and orchestration tool.

- A key advantage to Ansible over other automation engines is that it uses no agents and no additional custom security infrastructure, which simplifies deployment. Ansible uses a very simple, human-readable language called *YAML* for Ansible playbooks, to manage configuration, deployment, and orchestration tasks.

- Ansible works by connecting to your nodes and running small programs, called *Ansible modules*, to configure the resource for your system. Ansible executes these modules over Secure Shell (SSH) by default, and removes them when finished.

Benefits

➔ **Efficient :** Agentless, minimal setup

➔ **Fast :** Easy to learn/to remember, simple declarative language

➔ **Scalable :** Can managed thousands of nodes

➔ **Secure :** SSH transport

➔ **Large community** : thousands of roles on Ansible Galaxy

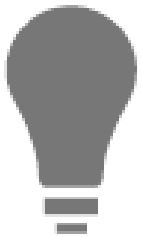
ARCHITECTURE, AGENTS, AND SECURITY

- One of the primary differentiators between Ansible and many other tools in this space is its architecture.
- Ansible is an agentless tool that runs in a ‘push’ model - no software is required to be installed on remote machines to make them manageable.
- Ansible by default manages remote machines over SSH (Linux and UNIX) or WinRM (Windows), using the remote management frameworks that already exist natively on those platforms.

Cont.

- Ansible builds on this by not requiring dedicated users or credentials - it respects the credentials that the user supplies when running Ansible.
- Ansible does not require administrator access, leveraging sudo, su, and other privilege escalation methods on request when necessary.
- By running in an agentless manner, no resources are consumed on managed machines when Ansible is not managing them.
- These attributes together make Ansible ideal for high-security environments or high-performance cases where there are concerns about stability or permanence of a management agent, but are generally useful attributes in all computing areas.

Cont.



SIMPLE

Human readable automation

No special coding skills needed

Tasks executed in order

Get productive quickly

POWERFUL

App deployment

Configuration management

Workflow orchestration

Orchestrate the app lifecycle

AGENTLESS

Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

More efficient & more secure

Use Cases

- Provisioning
- Configuration management
- Application deployments
- Rolling upgrades – Continuous Deployment
- Security and Compliance
- Orchestration

Cont.

- The number of servers managed by an individual administrator has risen dramatically in the past decade, especially as virtualization and growing cloud application usage has become standard fare.
- As a result, admins have had to find new ways of managing servers in a streamlined fashion.
- A systems administrator has many tasks on day-to-day operation:
  - Apply patches and updates via yum, apt, and other package managers.
  - Check resource usage (disk space, memory, CPU, swap space, network).
  - Check log files.
  - Manage system users and groups.
  - Manage DNS settings, hosts files, etc.
  - Copy files to and from servers.
  - Deploy applications or run application maintenance.
  - Reboot servers.
  - Manage cron jobs.
- Nearly all of these tasks can be (and usually are) at least partially automated—but some often need a human touch, especially when it comes to diagnosing issues in real time. And in today's complex multi-server environments, logging into servers individually is not a workable solution.
- Ansible allows admins to run ad-hoc commands on one or hundreds of machines at the same time, using the ansible command.

Establish Prerequisites

The easiest way to install Ansible is by adding a third-party repository named EPEL (Extra Packages for Enterprise Linux), which is maintained over at

http://fedoraproject.org/wiki/EPEL

rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

-U => Upgrade

-v => Print verbose information

-h => Hash

No other software is required as Ansible utilizes SSH to interact with remote servers.

Installation

- Now that we've added the EPEL repository, we're ready to install Ansible! This can be done by running **yum -y install ansible** on the command line. This will install a bunch of python dependencies during the process, but will only take around 30 seconds to complete.

- Once the above has completed, you can confirm that Ansible is installed and ready to go by running **ansible --version** command.

  Latest Ansible version : **2.9**

- To upgrade Ansible using the command **pip install --upgrade ansible**

Installation Methods

Ansible's only real dependency is **Python**.

Method – I : Install with yum

- $ wget http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-9.noarch.rpm
- $ yum localinstall epel-release-7-9.noarch.rpm
- $ yum --enablerepo=epel install ansible

Method – II : Install using pip

- $ pip install ansible

Ansible Layout

- **/etc/ansible** — The main configuration folder which encompasses all Ansible config
- **/etc/ansible/hosts** — This file holds information for the hosts/and host groups you will configure
- **/etc/ansible/ansible.cfg** — The main configuration file for Ansible
- **/etc/ansible/roles** — This folder allows you to create folders for each server role, web/app/db, etc.

How Ansible Works