Introduction to Ansible

What is Ansible?

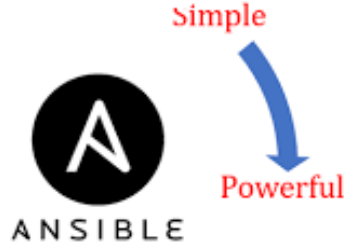
- Ansible is an **Open Source** Automation Platform.
- **Simple Automation Language** that can perfectly describe an IT application infrastructure in **Ansible Playbooks**.
- **Automation engine** that runs Ansible Playbooks

Why Ansible?

Ansible is Simple

- Ansible Playbooks provide **Human-Readable Automation**.
- 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, which allows people new to Ansible to get **productive quickly**.

Ansible Is Powerful

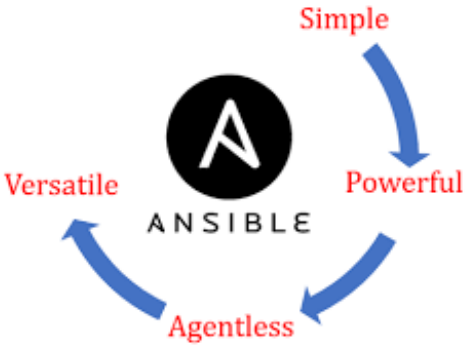


- You can use Ansible to deploy applications for:
    1. **Configuration Management**
    2. **Workflow Automation**
    3. **Network Automation**
- Ansible can be used to **Orchestrate** the entire **Application life cycle**.

Ansible Is Agentless

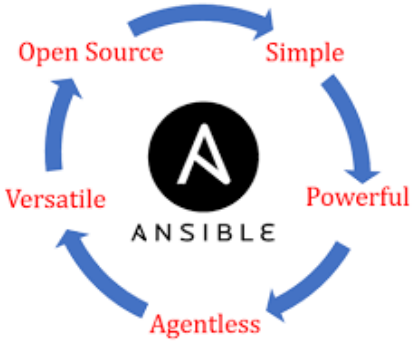
- Ansible is built around an **Agentless Architecture**.
- Ansible connects to the hosts it manages using **OpenSSH** or **WinRM** and runs tasks, often (but not always) by pushing out small programs called **Ansible modules** to those hosts.
- You can start using Ansible almost immediately because no 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 is Versatile



- Ansible provides Agentless support for **Linux**, **Windows**, **UNIX**, and **Network devices**,
- Ansible provides Agentless support for **Physical**, **Virtual**, **Cloud**, and **Container** environments.
- Ansible Playbooks, written as **YAML text files**, are easy to read and help ensure that everyone understands what they will do.
- Ansible Playbooks and projects are plain text. They can be treated like source code and placed in your existing **version control system**.

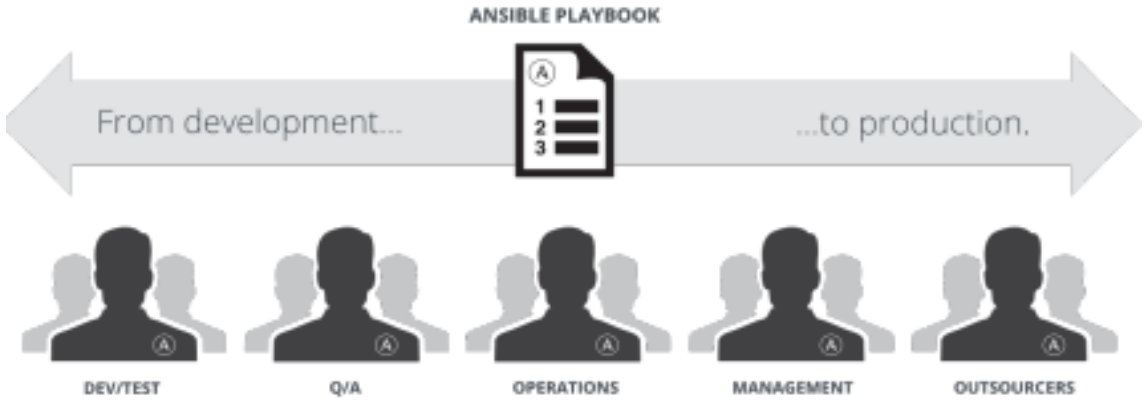
Ansible is OpenSource



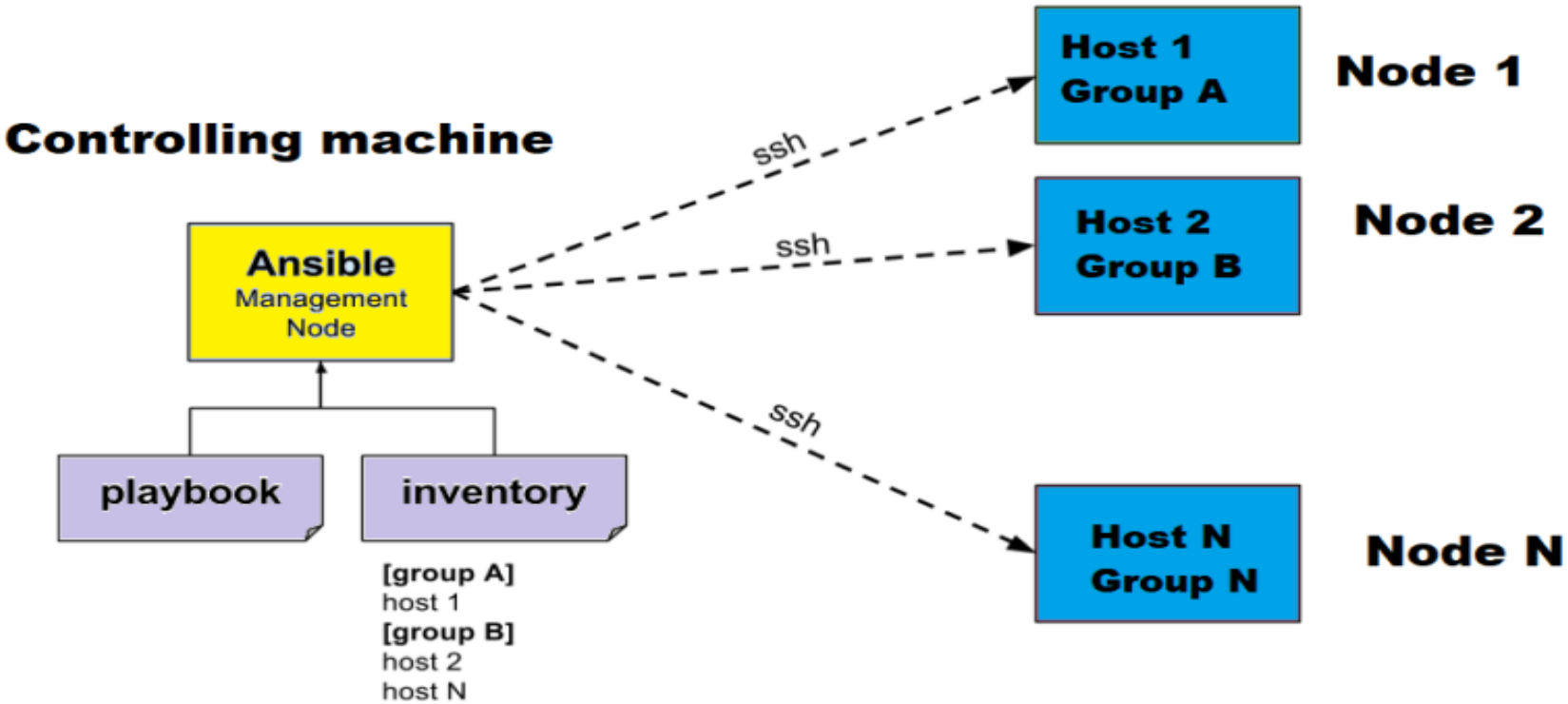
- **Perfect description of applications**: Every change can be made by Ansible Playbooks, and every aspect of your application environment can be described and documented.
- **Support for dynamic inventories**: The list of machines that Ansible manages can be dynamically updated from external sources 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.



Brief Concept of Ansible

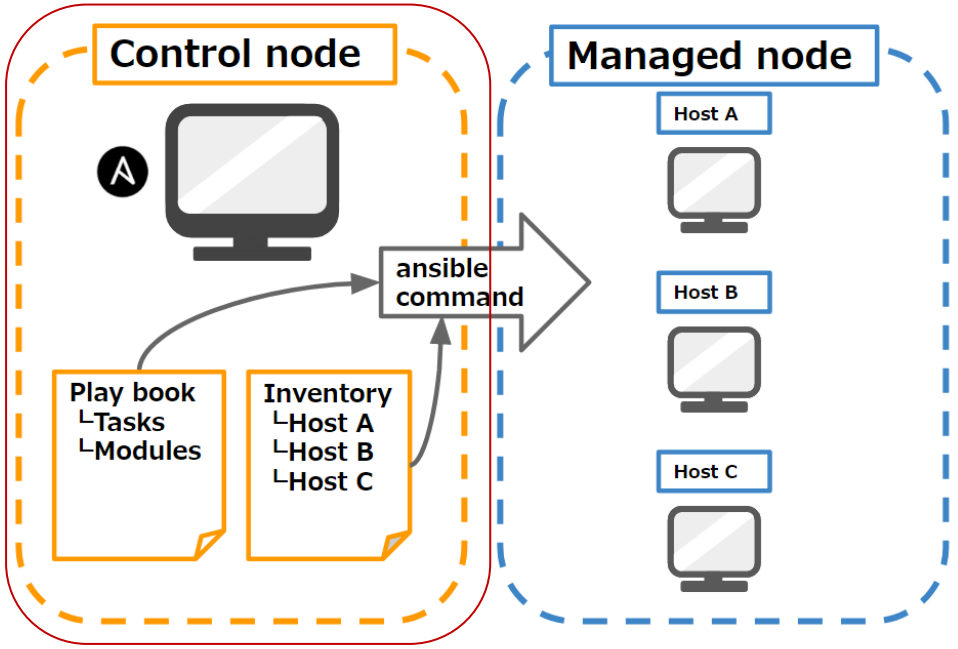


Ansible Concepts and Architecture(1-4)

- There are two types of machines in the Ansible architecture:
    1. **Control nodes**
    2. **Managed hosts**

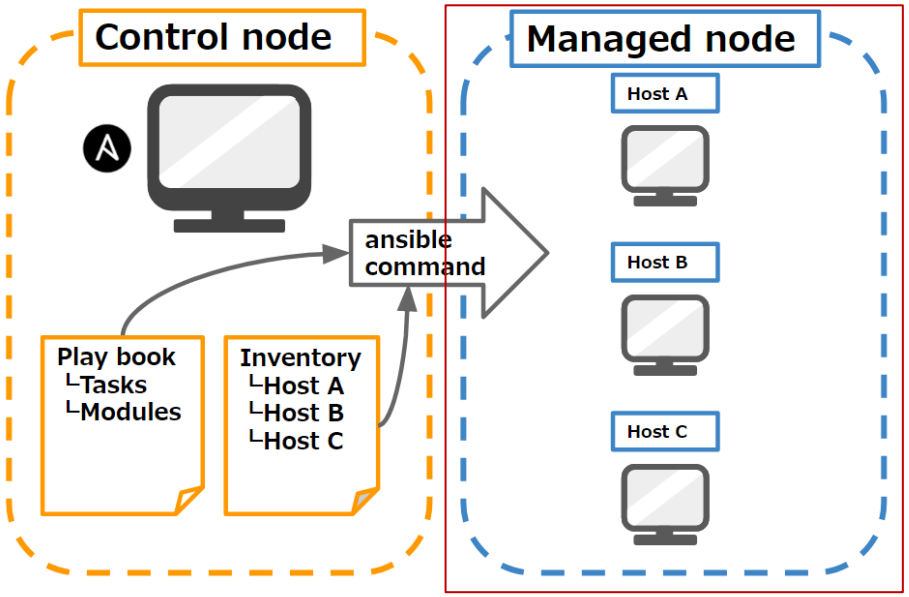
Control nodes (2-4)

- 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.**



Managed hosts (3-4)

- **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.



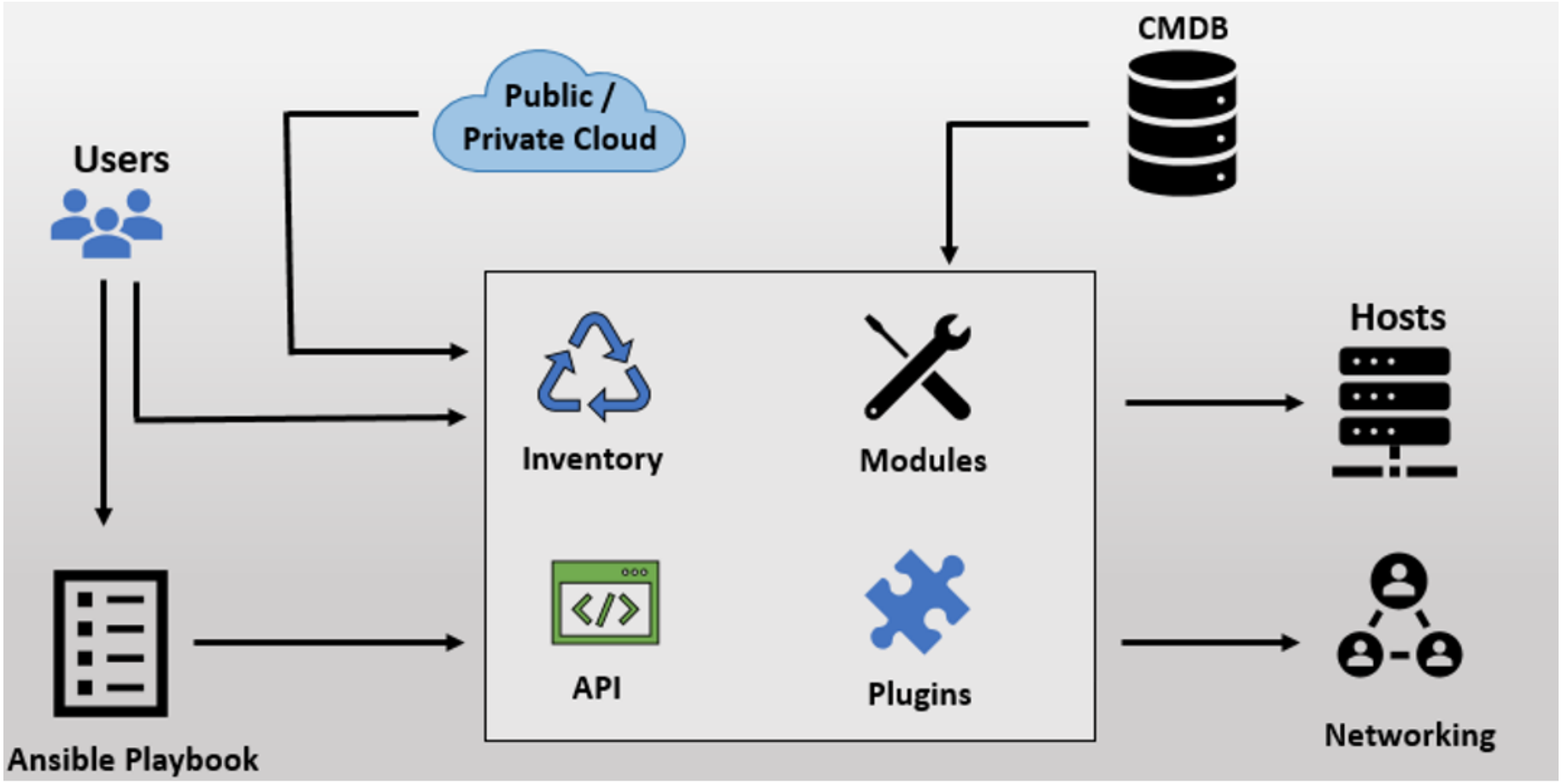
Ansible Concepts and Architecture (4-4)

- Instead of writing complex scripts, Ansible users create high-level plays to ensure a host or group of hosts are in a particular state.
- play performs a series of tasks on the 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**.

What are modules

- 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**.

Ansible Architecture



The Ansible Way(1-3)

- **Complexity kills productivity**

Simpler is better. Ansible is designed so that its tools are simple to use and automation is simple to write and read. You should take advantage of this to strive for simplification in how you create your automation.

The Ansible Way (2-3)

- **Optimize For Readability**

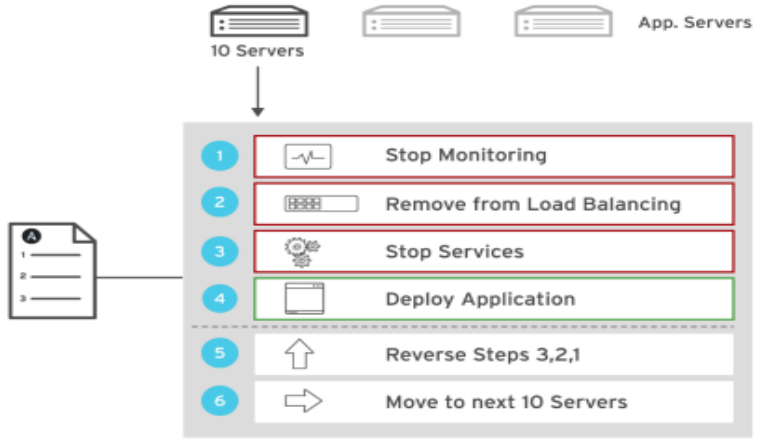
The Ansible automation language is built around simple, declarative, text-based files that are easy for humans to read. Written properly, Ansible Playbooks can clearly document your workflow automation.

The Ansible Way (3-3)

- **Think Declaratively**

Ansible is a **desired-state** engine. It approaches the problem of how to automate IT deployments by expressing them in terms of the state that you want your systems to be in. Ansible's goal is to put your systems into the desired state, only making changes that are necessary. Trying to treat Ansible like a **scripting language** is not the **right approach**.

Ansible provides complete automation:



Vast application of Ansible

Unlike some other tools, Ansible combines and unites orchestration with configuration management, provisioning, and application deployment in one easy-to-use platform.

Some applications of Ansible include:

I. **Configuration Management**
II. **Application Deployment**
III. **Provisioning**
IV. **Continuous Delivery**
V. **Security and Compliance**
VI. **Orchestration**

Configuration Management:

Centralizing configuration file management and **deployment** is a common use case for Ansible.

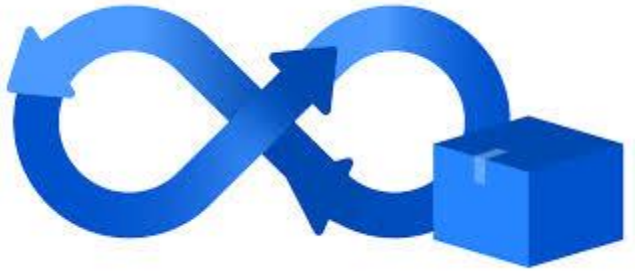
Application Deployment:

When you define your application with Ansible, and manage the deployment with Red Hat Ansible Tower, teams can effectively manage the entire application life cycle from development to production.

Provisioning

- Applications have to be deployed or installed on systems.
- Ansible and Red Hat Ansible Tower can help streamline the process of provisioning systems, whether you are PXE booting and kickstarting bare-metal servers or virtual machines, or creating virtual machines or cloud instances from templates.
- Applications have to be deployed or installed on systems.

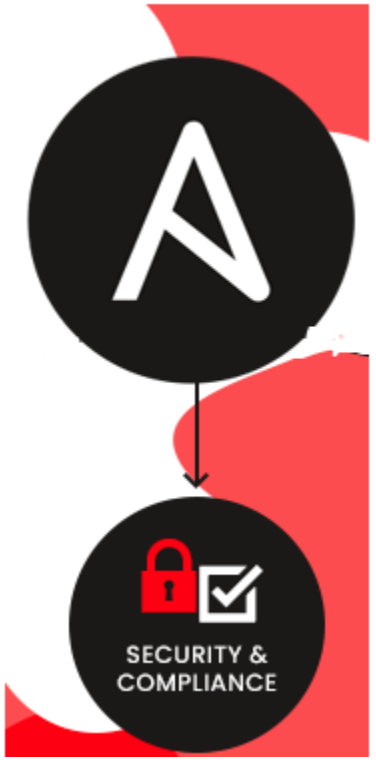
Continuous Delivery



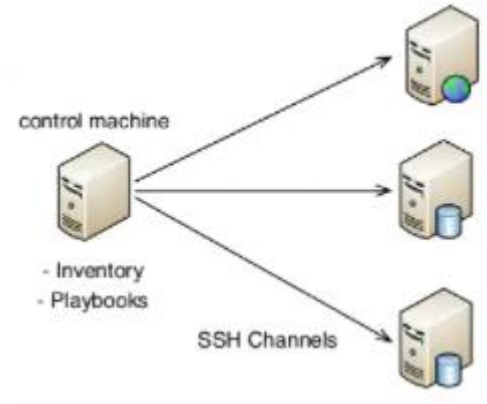
- Creating a CI/CD pipeline requires coordination and buy-in from numerous teams. You cannot do it without a simple automation platform that everyone in your organization can use. Ansible Playbooks keep your applications properly deployed and managed throughout their entire life cycle.

Security and Compliance

- When your security policy is defined in Ansible Playbooks, **scanning** and **remediation** of sitewide security policies can be integrated into other automated processes.
- Instead of being an afterthought, it is an integral part of everything that is deployed.



Orchestration



Configurations alone do not define your environment. You need to define how multiple configurations interact, and ensure that the disparate pieces can be managed as a whole

Installing Ansible

- **Ansible and Red Hat Ansible Automation Platform:**

Red Hat provides a fully supported version of Ansible through Red Hat Ansible Automation Platform. Ansible Automation Platform provides the core Ansible toolset plus additional certified and supported content, tools, and cloud services. Customers with a valid subscription can use the available repository, install the additional tools, and consume certified content from the cloud services.

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.**

Installing the core Ansible toolset involves relatively few steps and has minimal requirements.

Requires a **Red Hat Enterprise Linux 8.2** or later system, with a minimum of **two CPUs**, **4 GiB** of RAM, and **20 GiB** of available disk space.

Python 3 (version 3.5 or later) or Python 2 (version 2.7 or later) needs to be installed on the **control node**.

Control Nodes

You need a valid Red Hat Ansible Automation Platform subscription to install the core toolset on your control node. The installation process is as follows:

If you have activated Simple Content Access for your organization in the Red Hat Customer Portal, then you do not need to attach the subscription to your system. The installation process is as follows

- Register your system to Red Hat Subscription manager:

```
[root@ansible-master ~]# subscription-manager register
```

- Enable Red Hat Ansible Engine repository:

```
[root@ansible-master ~]#  subscription-manager repos \
> > --enable ansible-2-for-rhel-8-x86_64-rpms
```

- Inatall Red Hat Ansible Engine:

```
[root@ansible-master ~]# yum install ansible
```

Managed Hosts

One of the benefits of Ansible is that managed hosts do not need to have a **Special Agent installed**. The Ansible control node connects to managed hosts using a standard network protocol to ensure that the systems are in the specified state.

Managed hosts might have some requirements depending on how the control node connects to them and what modules it will run on them.

Linux and UNIX managed hosts need to have Python 2 (version 2.6 or later) or Python 3 (version 3.5 or later) installed for most modules to work.

For Red Hat Enterprise Linux 8, you may be able to depend on the platform-python package. You can also enable and install the python36 application stream

Managed Network Devices

You can also use Ansible automation to configure managed network devices such as routers and switches. Ansible includes a large number of modules specifically designed for this purpose. This includes support for Cisco IOS, IOS XR, and NX-OS; Juniper Junos; Arista EOS; and VyOS-based networking devices, among others.

You can write Ansible Playbooks for network devices using the same basic techniques that you use when writing playbooks for servers. Because most network devices cannot run Python, Ansible runs network modules on the control node, not on the managed hosts. Special connection methods are also used to communicate with network devices, typically using either CLI over SSH, XML over SSH, or API over HTTP(S)

Ansible vs SaltStack vs Chef vs Puppet

Pros & cons of Ansible

Pros	Cons
• No dependency on agents • Easy learning curve • Simple playbook structure • Streamlined code base	• Lacks UI • No notion of state • Nascent windows support • Lacks consistency between formats

Pros & cons of Puppet

Puppet is a full-fledged configuration automation and deployment orchestration solution. It's an open-source tool based on Ruby. For working, it counts on a customized Domain Scripting Language (DSL) nearer to JSON. It runs as a master-client setup and uses a model-driven approach. Large enterprises use it widely to automate sysadmins who spend ages configure, provision, troubleshoot, and maintain server operations.

Pros	Cons
• Great community support • Simple installation • Dynamic and idempotent server configuration • Runs on nearly every OS	• Ruby-based CLI • Ruby support is declining • Code base is bit complex • Less control compared to code-driven method

Pros & cons of Saltstack

SaltStack configuration tool relies on a master-client setup model or a non-centralized model. SaltStack is available in Python programming language and uses the push model for executing commands via SSH protocol. The platform also allows to group together clients and configuration templates to control the environment easily. It enables low-latency and high-speed communication for remote execution and data collection in sysadmin environments

Pros	Cons
• Feature-rich DSL • Enormously flexible • Great plugin API • Consistent input, output and configs- all YAML • Strong community support	• No immutable infrastructure • Difficult to set up • Challenging documentation • Poor support for non-Linux OS

Pros & cons of Chef

The chef is an automation platform that provides an effective way to configure and manage infrastructure. The chef works on Ruby and DSL language for writing the configurations. Its architecture is like the Puppet master-agent model. It also uses a pull-based approach and an additional logical Chef workstation to control configurations from the master to agents. It provides a configuration in a Ruby DSL using a client-server architecture

Pros
• Feature-rich DSL • Enormously flexible • Great plugin API • Consistent input, output and configs- all YAML • Strong community support

Cons
• No immutable infrastructure • Difficult to set up • Challenging documentation • Poor support for non-Linux OS

A Glimpse on Tool Capabilities

Each DevOps tool has its own set of capabilities that makes it unique. Have a look

Ansible	Puppet	Saltstack	Chef
Streamlined provisioning	Orchestration	Automation for CloudOps	Infrastructure automation
Configuration management	Automated provisioning	Automation for ITOps	Cloud automation
App deployment	Role-based access control	Continuous code integration and deployment	Compliance and security management
Automated workflow for Continuous Delivery	Visualization and reporting	DevOps toolchain workflow automation with support for Puppet, Chef, Docker, Jenkins, and Git.	Automated workflow for Continuous Delivery
Security and Compliance policy integration	Configuration automation	Application monitoring and auto-healing	Chef-Server using RabbitMQ, AMQP protocol.
Simplified orchestration	Code and node management	Orchestration	Automation for DevOps workflow

Chef vs. Puppet vs. Ansible vs. SaltStack:

A Quick Comparison to Know the Difference

Parameters	Chef	Puppet	Ansible	Saltstack
Availability	Yes	Yes	Yes	Yes
Configuration Language	DSL (Ruby)	DSL(PuppetDSL)	YAML (Python)	YAML (Python)
Setup and Installation	Moderate	Moderate	Very Easy	Moderate
Ease of Management	Tough	Tough	Easy	Easy
Scalability	HighlyScalable	HighlyScalable	HighlyScalable	HighlyScalable
Interoperability	High	High	High	High
Pricing	$13700	$11200-$19900	$10,000	$15,000(approx.)
Cloud Support	All	All	All	All