1. Introduction

1. Introduction

Ansible is an open source IT configuration management, deployment, and orchestration tool. It is unique from other management tools in many respects, aiming to provide large productivity gains to a wide variety of automation challenges as a more productive drop-in replacement for many core capabilities in other automation solutions.

Ansible seeks to keep automation tasks easy to understand, such that new users can be quickly brought into new IT projects, and longstanding automation content is easily understood even after months of being away from a project. Ansible seeks to make things powerful for expert users, but equally accessible for all skill levels, ensuring a quicker time to market for IT projects and faster, less-error prone turnaround on IT configuration change.

Ansible uses YAML files as its main source of information. YAML is a human readable data language that is commonly used for configuration or dataset files. It does come with some quirks though, such as being whitespace sensitive.

Ansible and all modules are Python 2.6 compatible, which means that they'll work with any version of Python2 above version 2.6. Therefore, there are no additional dependencies on the machines that you want to manage.

Not only are there no additional language dependencies for your machines, but also there are no additional dependencies at all. Ansible works by running commands via SSH or WinRM, so there is no need to install any additional software on the managed nodes. This is a huge benefit for two reasons:

1. The systems you are automating do not use additional resources because of daemons running in the background.

2. All the features that SSH provides can be used. You can use advanced features, such as Control Persist, Kerberos, and jump hosts. In addition, there is no need to implement your own authentication mechanism

1.1 Infrastructure as code

Infrastructure as code, also referred to as IaC, is a type of IT setup wherein developers or operations teams automatically manage and provision the technology stack for their systems through software, rather than using a manual process to configure discrete hardware devices and operating systems. Infrastructure as code is sometimes referred to as programmable or software-defined infrastructure.

The concept of infrastructure as code is similar to programming scripts, which are used to automate IT processes. However, scripts are primarily used to automate a series of static steps that must be repeated numerous times across multiple servers. Infrastructure as code uses higher-level or descriptive language to code more versatile and adaptive provisioning and deployment processes.

Ansible is a key example of an automation tool implementing infrastructure as code provisioning.

1.2 Architecture

Ansible is installed on a control node, which is the machine that communicates with the managed nodes. The managed nodes are the end devices you want to automate.
One of the primary differentiators between Ansible and many other automation tools is the architecture. Ansible is an agentless tool that runs in a 'push' model; no software is required to be installed on the managed nodes.

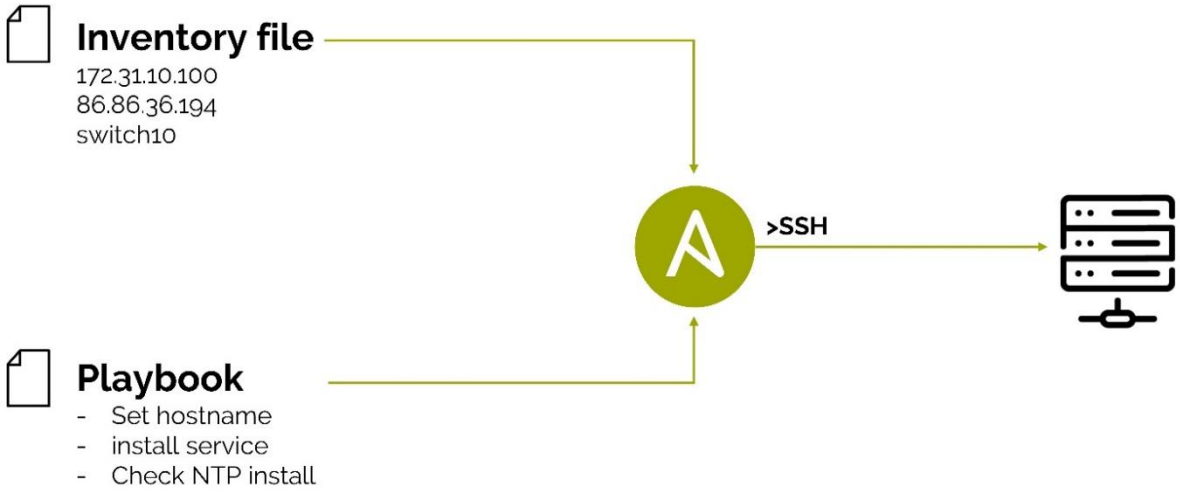
Ansible by default manages remote machines over SSH (Linux and UNIX) or WinRM (Windows), using the remote management frameworks that already exist natively on those platforms. Ansible builds on this by not requiring dedicated users or credentials; it just uses the credentials that the user supplies. Similarly, Ansible does not require administrator access, leveraging sudo, su, and other privilege escalation methods only on request.

1.3 Playbooks

Ansible performs automation and orchestration tasks using Playbooks. Playbooks are a YAML definition of automation tasks that describes how a particular result should be.

Like their namesake, Ansible Playbooks are prescriptive, yet responsive descriptions of how to perform an operation. It clearly states what each individual component of the IT infrastructure needs to do, but still allows components to react to discovered information.

Ansible Playbooks consist of series of 'plays' that define automation across a set of hosts, known as the 'inventory'. Each 'play' consists of multiple 'tasks', that can target one or more machines from the 'Inventory' to perform a specific task.



Each task uses an Ansible module; a python script for executing a specific task. These tasks can be simple, such as copying a configuration file from the control node to the managed node, or installing a software package. They can be complex, such as implementing a complete Virtual Routing domain within an Enterprise network. Ansible includes hundreds of modules, ranging from simple configuration management, to managing network devices, to modules for maintaining infrastructure on every major cloud provider.

The core Ansible modules allow for easy configuration of desired state; they check that the task that is specified needs to be performed before executing it. For example, if an Ansible task is defined to create a VLAN, the VLAN is only created when not yet existent. This desired state configuration, sometimes referred to as 'idempotency', ensures that the goal of each task is achieved while configuration can be applied repeatedly without side effects.

Ansible also supports encapsulating Playbook tasks into reusable units called 'roles.' Ansible roles can be used to apply common configurations in different scenarios, such as having a common switch configuration role that may be used for all switches in the IT environment. The Ansible Galaxy community site contains thousands of roles that can be used and customized to build Playbooks.

1.4 Modules

As noted above, tasks in Ansible are performed by Ansible 'modules'; small pieces of code that run on the control node to create configuration data. If you have the need for a new module to handle a specific task of automation your IT infrastructure that is not covered by Ansible's included set of 450+ modules, Ansible can be extended by writing your own modules. While the modules that ere included with Ansible are implemented in Python and PowerShell, Ansible modules can be written in any language and are only required to take JSON as input and produce JSON as output.

1.5 Network Automation

Ansible strives to automate not just traditional IT server and software, but the entirety of IT infrastructure, including areas not covered by traditional IT automation tools.

On most network devices it is not possible to install custom software, such as the agent daemon used by most automation tools in a PULL model. Ansible's agentless nature makes it possible to automate network devices, and support is included with Ansible for automating networking from major vendors such as Cisco, Hewlett Packard Enterprise, Juniper and more.

By leveraging this networking support, network automation no longer needs to be done by a separate team but can be done by the same tools and processes used by other automation already implemented. Configuring a new VLAN or access control list becomes just a few additional Ansible tasks in the deployment Playbook, rather than a ticket filed with the separate networking team.