RED HAT® ANSIBLE® Automation

ansible.com

WHITEPAPER

ANSIBLE IN DEPTH

“Ansible is quite fun to use right away. As soon as you write five lines of code it works. With SSH and Ansible I can send commands to 500 servers without having even used the servers before.”

MARK MAAS
UNIX/LINUX SYSTEMS ADMINISTRATOR
BINCKBANK

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. Furthermore, Ansible seeks to solve major unsolved IT challenges such as clear orchestration of complex multi- tier workflows and cleanly unifying OS configuration and application software deployment under a single banner.

Ansible is designed to be minimal in nature, consistent, secure, and highly reliable, with an extremely low learning curve for administrators, developers, and IT managers. Ansible seeks to keep descriptions of IT easy to build, and 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.

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.

Ansible builds on this by not requiring dedicated users or credentials - it respects the credentials that the user supplies when running Ansible. Similarly, Ansible does not require administrator access, leveraging sudo, su, and other privilege escalation methods on request when necessary.

This method allows Ansible to be more secure. By using the credentials passed by the user, those with access to the control server (or source control) cannot make content be pushed out to remote systems (or otherwise command them) without also having credentials on remote systems. Similarly, by operating in a push-based model where only needed code (called Ansible 'modules') are passed to remote machines, remote machines cannot see or affect how other machines are configured.

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.

PLAYBOOKS AND ROLES

Ansible performs automation and orchestration of IT environments via Playbooks. Playbooks are a YAML definition of automation tasks that describe how a particular piece of automation should be done. Like their namesake, Ansible Playbooks are prescriptive, yet responsive descriptions of how to perform an operation - in this case, IT automation, that clearly states what each individual component of your IT infrastructure needs to do, but still allows components to react to discovered information, and to operate in concert with each other.

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, many, or all of the hosts in the inventory. Each task is a call to an Ansible module - a small piece of code for doing a specific task. These tasks can be simple, such as placing a configuration file on a target machine, or installing a software package. They can be complex, such as spinning up an entire CloudFormation infrastructure in Amazon EC2. Ansible includes hundreds of modules, ranging from simple configuration management, to managing network devices, to modules for maintaining infrastructure on every major cloud provider.

Core included modules for Ansible are written in an manner to allow for easy configuration of desired state - they check that the task that is specified actually needs to be done before executing it. For example, if an Ansible task is defined to start a webserver, configuration is only done if the webserver is not already started. This desired state configuration, sometimes referred to as 'idempotency,' ensures that configuration can be applied repeatedly without side effects, and that configuration runs quickly and efficiently when it has already been applied.

Ansible also supports encapsulating Playbook tasks into reusable units called 'roles.' Ansible roles can be used to easily apply common configurations in different scenarios, such as having a common web server configuration role that may be used in development, test, and production automation. The Ansible Galaxy community site contains thousands of roles that can be used and customized to build Playbooks.

ADVANCED FEATURES

The Ansible Playbook language includes a variety of features that allow for complex automation flow, including conditional execution of tasks, the ability to gather variables and information from the remote system, ability to spawn asynchronous long running actions, ability to operate in either a push or pull configuration, a "check" mode to test for pending changes without applying change, and the ability to tag certain plays and tasks so that only certain parts of configuration can be applied. These features allow your applications and environments to be modelled simply and easily, in a logical framework that is easily understood not just by the automation developer, but by anyone from developers to operators to CIOs.

"In the beginning I didn't realize Ansible was good for orchestration as well, but found it out quickly. I really loved that as it beats competitors right there."

BEIER CAI
DIRECTOR OF TECHNOLOGY
HOOTSUITE MEDIA, INC.

USING PLAYBOOKS FOR COMPLEX ORCHESTRATION

By combining different tasks into a Playbook, complex automation can be achieved. As a detailed example, consider a traditional three-tier web application and its environment consisting of:

- Application servers
- Database servers
- Content servers
- Load balancers
- Plus, a monitoring system connected to an alert system such as a pager notification service

In this example, Ansible can easily be used to implement a complex, cluster-wide rolling update process that consists of:

- Consulting a configuration/settings repository for information about the involved servers
- Configuring the base OS on all machines and enforcing desired state
- Identifying a portion of the web application servers to update
- Signaling the monitoring system of an outage window prior to bringing the servers off line
- Signaling load balancers to take the application servers out of a load balanced pool
- Stopping the web application server
- Deploying or updating the web application server code, data, and content
- Starting the web application server
- Running appropriate tests on the new server and code
- Signaling the load balancers to put the application servers back into the load balanced pool
- Signaling the monitoring system to resume alerts on any detected issues on those servers
- Repeating this process for remaining application servers in a rolling update process
- Repeating these rolling update processes for other tiers such as database or content tiers
- Sending email reports and logging as desired when updates are complete

By running this Playbook across a cluster in groups of machines, an application update can be achieved with zero downtime.

EXTENSIBILITY

Ansible can be extended in many ways.

As noted above, tasks in Ansible are performed by Ansible 'modules' - small pieces of code that run on remote hosts. If there is a need for a new module to handle some portion of IT infrastructure that is not covered by Ansible's included set of 450+ modules, extending Ansible is easy. While the modules shipped 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.

Ansible can also be extended through its support for dynamic inventory. Dynamic inventory lets Ansible Playbooks be run against a set of machines and infrastructure discovered at runtime rather than statically defined. Typically, this is done by reading information from a public or private cloud provider, a virtualization provider, or a CMDB. Ansible includes support for all major cloud providers out of the box, and can be easily extended to add support for new providers and new sources of truth as needed - just by writing a custom program in any scripting or compiled language that can return a JSON inventory definition.

The runtime behavior of Ansible itself is also easily extended, as many features are implemented via a plugin mechanism. Ansible's normal output display is handled via a set of callback plugins that processes events as jobs are run - new callback plugins can log to log aggregators, send messages to notification services, and more. Ansible's use of SSH and WinRM to connect to managed machines are done through connection plugins - new connection plugins can be written to access managed nodes in new ways if desired. New lookup plugins can be created for looking up variables, passwords, and credentials from new credential storage mechanisms or CMDBs at runtime.

This extensibility of Ansible means that no matter how custom your environment may be, it is easy to make it work with Ansible. Ansible's open source nature also means that new features are added to extend Ansible in every release, and it's easy to contribute your changes back.

TOOL UNIFICATION

Ansible is designed to make IT configurations and processes both simple to read or write, even by those untrained in reading those configurations. While Ansible can accomplish all types of automation tasks, Ansible does not resemble software programming languages, but rather basic textual descriptions of desired states and processes, while being neutral to the types of processes described.

With other traditional approaches, users have typically had to combine many different tools together to cover the basics of managing IT operating systems and software configurations. However, Ansible's simple, task-based nature makes it uniquely applicable to a variety of use cases, including:

- Configuration management - for describing the desired state of a system, called manually or via provisioning tools
- Application deployment - for pushing out hosted application software to one machine or a series of machines
- Orchestration - for coordinating a multi-machine process such as a rolling cluster upgrade
- As-needed task execution - for performing tasks immediately that do not fit into the previous models, such as batch server rebooting.

Ansible makes these approaches available in a single tool. This not only allows for integrating multiple disparate processes into a single framework for easy training, education, and operation, but also means that Ansible seamlessly fits in with other tools - it can be used for any of these use cases without disturbing existing infrastructure that may already be in use.

CLOUD AND INFRASTRUCTURE INTEGRATION

Ansible is capable of easily deploying workloads to a variety of public and on-premise cloud environments, including but not limited to cloud providers such as Amazon Web Services, Microsoft Azure, Rackspace, and Google Compute Engine, and local infrastructure such as VMware, OpenStack, and CloudStack. This includes not just compute provisioning, but storage and networks as well. As noted, further integrations are easy, and more are added to each new Ansible release.

“The work the Ansible team is doing… is something the entire industry should be paying attention to.”

LEW TUCKER
VP & CTO, CLOUD COMPUTING
CISCO

NETWORK AUTOMATION

Ansible strives to automate not just traditional IT server and software installations, but the entirety of IT infrastructure, including areas not covered by traditional IT automation tools. Ansible’s task-based, agentless nature makes it easily applicable to the networking space, and support is included with Ansible for automating networking from major vendors such as Cisco, Juniper, Hewlett Packard Enterprise, Cumulus and more.

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

ABOUT RED HAT ANSIBLE AUTOMATION

Ansible, an open source community project sponsored by Red Hat, is the simplest way to automate IT. Ansible is the only automation language that can be used across entire IT teams – from systems and network administrators to developers and managers. Ansible by Red Hat provides enterprise-ready solutions to automate your entire application lifecycle – from servers to clouds to containers and everything in between. Red Hat Ansible Tower is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

READY TO AUTOMATE? ansible.com • info@ansible.com