Ansible

ANURAG KUMAR JAIN[1] AND GREGOR VON LASZEWSKI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: laszewski@gmail.com

Paper 1, February 26, 2017

Ansible is a powerful open source automation tool which can be used in configuration management, deployment, and orchestration tool.[1] This paper gives an in-depth overview of Ansible.

Keywords: Cloud, Ansible, Playbook, Roles

https://github.com/anurag2301/sp17-i524

INTRODUCTION

Ansible is an open source automation engine which can be used to automate cloud provisioning, configuration management, and application deployment. It is designed to be minimal in nature, secure, consistent, and highly reliable.[2] In many respects, it is unique from other management tools and aims to provide large productivity gains. It has an extremely low learning curve and seeks to solve major unsolved IT challenges under a single banner. Michael DeHaan developed the Ansible platform and Ansible, Inc. was the company set up to commercially support and sponsor Ansible. The company was later acquired by Red Hat Inc. It is available on Fedora, Red Hat Enterprise Linux, CentOS and other operating systems.[1]

ARCHITECTURE

One of the primary differences between Ansible and many other tools in the space is its architecture. Ansible is an agentless tool, it doesn't requires any software to be installed on the remote machines to make them manageable. By default is manages remote machines over SSH or WinRM, which are natively present on those platforms.[3]

Like many other configuration management software, Ansible distinguishes two types of servers: controlling machines and nodes. Ansible uses a single controlling machine where the orchestration begins. Nodes are managed by a controlling machine over SSH. The location of the nodes are described by the inventory of the controlling machine.[2]

Modules are deployed by Ansible over SSH. These modules are temporarily stored in the nodes and communicate with the controlling machine through a JSON protocol over the standard output.[3]

The design goals of Ansible includes consistency, high reliability, low learning curve, security and to be minimalistic in nature. The security comes from the fact that Ansible doesn't requires users to deploy agents on nodes and manages remote machines using SSH or WinRM. Ansible doesn't require dedicated users or credentials - it respects the credentials that the user supplies when running Ansible. Similarly, Ansible does not require administrator access, it leverages sudo, su, and other privilege escalation methods on request when necessary.[4] If needed, Ansible can connect with LDAP, Kerberos, and other centralized authentication management systems.[5]

ADVANCED FEATURES

The Ansible Playbook language includes a variety of features which allow complex automation flow, this includes 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, it also includes 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. [2]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. Ansible has low overhead and is much smaller when compared to other tools like Puppet. [6]

PLAYBOOK AND ROLES

Playbook is what Ansible uses to perform automation and orchestration. They are Ansible's configuration, deployment and orchestration language. They can be used to describe policy you need your remote systems to enforce, or a set of steps in a general IT process.[7]

At a basic level, playbooks can be used to manage configurations and deployments to remote machines. While at an advanced level, they can be utilized to sequence multi-tier rollouts which involves rolling updates, and can also be used to delegate actions to other hosts, interacting with monitoring servers and load balancers at the same time.

Playbooks consists of series of 'plays' which are used to define automation across a set of hosts, known as the 'inventory'. These 'play' generally consists of multiple 'tasks,' that can select one, many, or all of the hosts in the inventory where each task is a call to an Ansible module - a small piece of code for doing a specific task. The tasks may be complex, such as spinning up an entire cloud formation infrastructure in Amazon EC2. Ansible includes hundreds of modules which help it perform a vast range of tasks.[3]

Similar to many other languages Ansible supports encapsulating Playbook tasks into reusable units called 'roles.' These roles can be used to easily apply common configurations in different scenarios, such as having a common web server configuration role which can be used in development, test, and production automation. The Ansible Galaxy community site contains thousands of customizable rules that can be reused used to build Playbooks.

COMPLEX ORCHESTRATION USING PLAYBOOKS

Playbook can be used to combine multiple tasks to achieve complex automation.[7] Playbook and Ansible can be easily used in implementing a cluster-wide rolling update 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. It can also be used in signaling the monitoring system of an outage window prior to bringing the servers offline and signaling load balancers to take the application servers out of a load balanced pool. The usage doesn't ends here and it can be utilized to start and stop server, running appropriate tests on the new server or even deploying/updating the server code, data, and content. Ansible can also be used to send email reports and as a logging tool.[3]

EXTENSIBILITY

Tasks in Ansible are performed by Ansible 'modules' which are pieces of code that run on remote hosts. Although there are a vast set of modules which cover a lot of things which a user may require there might be a need to implement a new module to handle some portion of IT infrastructure. Ansible makes it simpler and by allowing the modules to be written in any language, with the only requirement that they are required to take JSON as input and product JSON as output.[3]

We can also extend Ansible through it's dynamic inventory which allows Ansible Playbooks to run against machines and infrastructure discovered during runtime. Out of the box Ansible includes support for all major cloud providers, it can also be easily extended to add support for new providers and new sources as needed.

ONE TOOL TO DO IT ALL

Ansible is designed to make IT configurations and processes both simple to read or write, the code is human readable and can be read even by those who are not trained in reading those configurations. Ansible is different from genrally used software programming languages, it uses basic textual descriptions of desired states and processes, while being neutral to the types of processes described. Ansible's simple, task-based nature makes it unique and it can be applied to a variety of use cases which includes configuration management, application deployment, orchestration and need basis test execution.

Ansible combines these approaches into 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 the above mentioned use cases without modifying existing infrastructure that may already be in use.

INTEGRATION OF CLOUD AND INFRASTRUCTURE

Ansible can easily deploy workloads to a variety of public and on-premise cloud environments. This capability includes 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 and the capability doesn't ends here. As noted, further integrations are easy, and more are added to each new Ansible release. And as Ansible is open source anyone can make his/her contributions.[3]

AUTOMATING NETWORK

Ansible can easily automate traditional IT server and software installations, but it strives to automate IT infrastructure entirely, including areas which are not covered by traditional IT automation tools. Due to the Ansible's agentless, task-based nature it can easily be utilized in the networking space, and the inbuilt support is included with Ansible for automating networking from major vendors such as Cisco, Juniper, Hewlett Packard Enterprise, Cumulus and more.[2]

By harnessing this networking support, network automation is no longer a task required to be done a separate team, with separate tools, and separate processes. It can easily be done by the same tools and processes used by other automation procedures you already have. Ansible tasks also include configuring switching and VLAN for a new service. So now users don't need a ticket filed whenever a new service comes in.

CONCLUSION

Ansible is an open source community project sponsored by Red Hat. It is one of the easiest way to automate IT. Ansible code is easy to read and write and the commands are in human readable format. The power of Ansible language can be utilized across entire IT teams – from systems and network administrators to developers and managers. Ansible includes thousands of reusable modules which makes it even more user friendly and users can write new modules in any language which makes it flexible too. Ansible by Red Hat provides enterprise-ready solutions to automate your entire application lifecycle – from servers to clouds to containers and everything in between. Ansible Tower by Red Hat is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

REFERENCES

[1] Wikipedia, "Ansible (software)," Web Page, Feb. 2017, accessed: 2017-02-20. [Online]. Available: https://en.wikipedia.org/wiki/Ansible_(software)

[2] Red Hat Inc., "Ansible in depth," Web Page, Feb. 2016, accessed: 2017-02-23. [Online]. Available: https://www.ansible.com/ansible-in-depth-whitepaper

[3] Red Hat Inc., “How ansible works,” Web Page, Feb. 2016, accessed: 2017-02-14. [Online]. Available: https://www.ansible.com/how-ansible-works
[4] Michael DeHaan, “Ansible,” Code Repository, Feb. 2017, accessed: 2017-02-21. [Online]. Available: https://github.com/ansible/ansible
[5] Red Hat Inc., “Ansible documentation,” Web Page, Feb. 2016, accessed: 2017-02-15. [Online]. Available: https://docs.ansible.com/ansible/index.html
[6] UpGuard Inc., “Ansible vs puppet,” Web Page, Feb. 2017, accessed: 2017-02-24. [Online]. Available: https://www.upguard.com/articles/ansible-puppet
[7] Red Hat Inc., “Playbooks,” Web Page, Feb. 2016, accessed: 2017-02-24. [Online]. Available: https://docs.ansible.com/ansible/playbooks.html