**Configuration Management**

What so you need to build good software? **System, good environment**

Configuration Management is process to setup such system and environment in well documented manner.

CM, when applied over the life cycle of a system, provides visibility and control of its performance, functional and physical attributes.

CM make sure that system is behaving as intended and documented to support project life cycle

Cm manages system information and system changes to improve performance, reliability, or maintainability; extend life; reduce cost; reduce risk and liability; or correct defects

Two types of CM are 1) Hardware 2) Software

We are dealing with software part.

Two ways of configuration management - 1) Manual 2) Automatic

**Automated configuration management:**  eliminates a lot of manual work, and creates greater dependability and predictability

**Manual:**  custom scripts, and golden images to accomplish repetitive tasks. In large environments, or with multiple team members, these methods are difficult to scale, track and maintain, and can create several issues, including configuration drift, non-compliance, and decreased productivity and responsiveness.

**Before Ansible**

**Shell Script**

* + - Even for small task you have to write very long and tedious script
    - Difficult to maintain script if task grows
    - You have to manually run those script on remote machine one at a time
    - Needs lots of time for large scale configuration
    - Difficult to track these scripts
    - You need to linux expert for writing these scripts
    - Does not provide proper way to handle hybrid infrastructure which consist of cloud,ec2 etc.
    - Difficult to maintain same environment all time
    - Lack of proper documentation

**After Ansible**

* + - Short scripts
    - Easy to maintain
    - No need of linux expert
    - Anyone can write and understand scripts
    - Easy to track
    - Easy to handle hybrid infrastructure which include cloud
    - 1000s off nodes can be configured at a time
    - Saves lots of time
    - Can maintain same environment all the time.
    - Proper documented
    - Automated

**Infrastructure as Code**

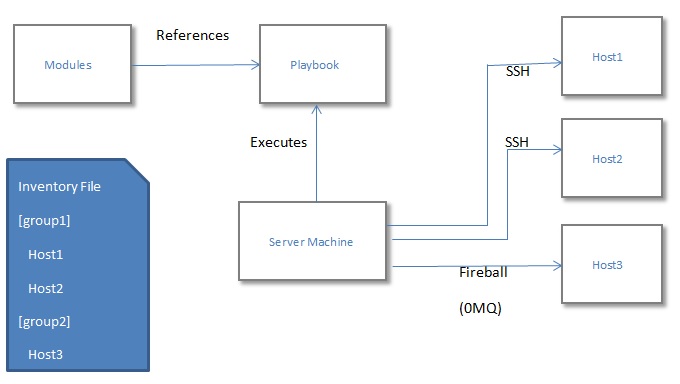
Up till now we were dependent on IS team for providing us proper infrastructure for software development, testing and deployment. Have you ever thought that developer is writing code for maintaining infrastructure. But, now it is possible with the help of ansible.

In ansible we maintain Infrastructure in the form of code. So, that you can track any changes in your infrastructure and easily update if you want some changes.

**What is Ansible?**

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

* Initial Release February 2012
* Configuration Management
* Application Deployment
* Cloud Provisining
* AD-HOC Task execution
* Continuous Deployment

**Architecture**

**Working**

1. Install Ansible and setup its environment
2. Make entry of remote host in Inventory file
3. Write playbook for your task.
4. Run playbook on main machine which calls ansible modules from module library for performing tasks
5. When you execute playbook, ansible contacts to remote machine through SSH and pushes its modules over ssh to perform desired task i.e agentless execution
6. When task is done it deletes pushed modules from remote system and leave no traces.
7. If you want to boost execution speed you can use other execution modes like fireball, pipelining

**Key Concept**

* Inventory
* Modules
* Ad Hoc Commands
* Playbooks
  + Roles
  + Tasks
  + Variables
  + Templates
  + Handlers

**Inventory**

* Define a list of target hosts
* usually located in /etc/ansible/hosts

**Modules**

* They are like libraries
* Playbook internally calls these modules
* can be written in any language as long as they output JSON
* take parameters and conditions to define desired state
* handles processing of system resources, services, packages, files, etc. in idempotent fashion
* ansible comes preloaded with a lots of modules

**Ad Hoc Commands**

* These are ansible simple commands for simple tasks, such as making sure a service is running, or to trigger updates and reboots.
* run a single, one-off command
* run on a full or partial inventory
* run on a single host
* no need to save for later
* Format : ansible <host pattern> <module> <arguments>

Ex. ansible all -m user -a "name=joe password=wat"

**Playbook**

* It is collection of tasks (plays)
* For more powerful configuration management
* Kept in source control, developed, validated
* Declare configurations of more complex mutli-system enviornments
* Arrange and run tasks synchronously or asynchronously

Ex. Smple playbook => my\_plybook.yml

**- - -**

**- hosts: webservers**

**user: root**

**vars:**

**http\_port: 80**

**max\_clients: 200**

**tasks:**

**- name: ensure apache is at the latest version**

**action: yum pkg=httpd state=latest**

**- name: write the apache config file**

**action: template src=httpd.j2 dest=/etc/httpd.conf**

**notify:**

**- restart apache**

**- name: ensure apache is running**

**action: service name=httpd state=started**

**handlers:**

**- name: restart apache**

**action: service name=httpd state=restarted**

Running Playbook

**ansible-playbook mysite.yml -f 10**

Ansible Project Directory Structure for best practices :



**Roles**

* A feature of Ansible for encouraging reuse of code and best practices.
* Encapsulate a reusable service definition including Actions ,Variables, Templates, Files
* Defined in standard directory structure (subproject-like)
* A host / group can have multiple roles
* A role can be applied to multiple groups

**Role study links:**

<http://labs.qandidate.com/blog/2013/11/21/installing-a-lamp-server-with-ansible-playbooks-and-roles/>

<https://www.digitalocean.com/community/tutorials/how-to-use-ansible-roles-to-abstract-your-infrastructure-environment>

**Tasks**

* Each play contains a list of tasks.
* Tasks are executed in order, one at a time, against all machines matched by the host pattern, before moving on to the next task
* These task calls ansible modules to perform task.

**Variables**

* Vars can be defined at many levels (default, role ,playbook)

**Templates**

* Ansible uses jinja2 templating format
* Templates are interpreted by jinja2 template engine
* It allows Ansible to use some programing language features
* fill variables in differently depending on conditions
* Powerful conditionals
* Loops and iterators

**Handlers**

* The things listed in the ‘notify’ section of a task are called handlers.
* Handlers are how you can perform various post-deployment tasks like restating some services when some of its configuration files changed.  Handlers only get triggered if the state of the line actually changed
* These ‘notify’ actions are triggered at the end of each block of tasks in a playbook, and will only be triggered once even if notified by multiple different tasks.
* Notify handlers are always run in the order written.

**Best Practices**

1. Inventory: While making entry in inventory you should define groups based on purpose of the host (roles) and also geography or datacenter location (if applicable):

Ex [atlanta-webservers]

www-atl-1.example.com

www-atl-2.example.com

1. Group and Host variable: If you are using particular thing to whole group or particular set of host , then you should use Group or host variables with meaningful names.

Ex # file: group\_vars/webservers

apacheMaxRequestsPerChild: 3000

apacheMaxClients: 900

1. Directory structure: You should follow standard directory structure suggested in ansible doc
2. Roles : you should use roles structure while writing playbooks

**Puppet ,Chef are already there Then Why Ansible? (features)**

**Saying: If you can’t figure out in 10 minutes how to use it you should not be in IT. Ansible follows it.**

1) Agent less Architecture : It don’t have agent concept. Modules are pushed to hosts over SSH and after worked done those modules are erased from hosts

2)Python based : Core modules are written in python and python comes by default in most of linux flavours.

3)YAML Syntax : YAML is human readable data serialization syntax. In Ansible it is used to write configuration code.

4)Modules can be written in any language : So, no need to learn python.

5) Push based : As it is pushed based, It does not dependent on hosts to contact master. So, it executes modules whenever need arises.

6) Supports pull mode : If you want to use ansible pull mode, you can configure it

7)Fireball mode : If you have to manage 1000s of nodes in short time you can configure ansible for fireball mode. This mode uses 0MQ message transfer protocol which is 10 times faster than SSH.

8) Encryption and security built in : As it runs on SSH.

9) Idempotent : means if run Plybook and configured hosts but you forget one task to add in playbook. Now, If you add that task and re-run playbook, then only that task will get executed and other won’t execute

**Cons**

* GUI is not well developed
* Came in 2012 , so not used by many
* No support for Windows

**Ansible Window Support:**

* Starting in version 1.7, Ansible also contains support for managing Windows machines. This uses native powershell remoting, rather than SSH.
* In order for Ansible to manage your windows machines, you will have to enable Powershell remoting configured.
* A Linux control machine will be required to manage Windows hosts.

**Case Studies**

* **HootSuite Media, Inc. -Ansible Case Study**

HootSuite is a social media management system for businesses and Organizations. More than 75% of Fortune 1000 companies now use HootSuite for social media management, social marketing, social customer service, and social selling. They are using ansible for managing their all servers.

<http://cdn2.hubspot.net/hub/330046/file-480366621-pdf/pdf_content/Hoot_Suite_Case_Study.pdf?t=1409060021116>

* **Binck (BinckBank) -Ansible Case Study**

BinckBank is an online bank for investors based in Amsterdam and is ranked among the top five investment banks in Europe. It has 600 UNIX servers in house. They are managing 600 servers using ansible

<http://cdn2.hubspot.net/hub/330046/file-480395301-pdf/pdf_content/Binck_Bank_Case_Study.pdf?t=1409060021116>

Useful links:

<http://cdn2.hubspot.net/hub/330046/file-479069823-pdf/pdf_content/Achieving_Rolling_Updates_and_Continuous_Deployment_with_Zero_Downtime.pdf?t=1414679671862>

<http://cdn2.hubspot.net/hub/330046/file-479013288-pdf/pdf_content/The_Benefits_of_Agentless_Architecture.pdf?t=1414679671862>

<http://cdn2.hubspot.net/hub/330046/file-480366556-pdf/pdf_content/Ansible_in_Depth.pdf?t=1414679671862>

http://cdn2.hubspot.net/hub/330046/file-476015569-pdf/pdf\_content/Scaling\_and\_Performance\_of\_the\_Ansible\_Management\_Toolchain.pdf?t=1414679671862

**Advance Concepts:**

**Fireball mode:**

1. Deprecated
2. Used to sped up ansible working.
3. Alternatives are Pipelining and Accelerated mode.

**Accelerated Mode:**

1. Use only if you are unable to use pipelining feature of ansible.
2. Accelerated mode can be anywhere from 2-6x faster than SSH and 10x faster than paramiko.
3. By default, Ansible will use port 5099 for the accelerated connection.
4. Accelerated mode offers several improvements over the (deprecated) original fireball mode

**Pipelining:**

1. Enabling pipelining reduces the number of SSH operations required to execute a module on the remote server, by executing many ansible modules without actual file transfer.
2. This can result in a very significant performance improvement when enabled.
3. pipelining is better than accelerate mode for nearly all use cases.

**Asynchronous Mode**

1. By default, when you run playbook , the ssh connection remains open until the task is done on each node.
2. This may not always be desirable because you are blocking ssh connection or you may be running operations that take longer than the SSH timeout.
3. To resolve this problem Asynchronous Mode is used.
4. Here we kick off all modules at once and then poll until they are done.

**Check Mode**

When ansible-playbook is executed with - -check it will not make any changes on remote systems. Instead will report what changes they would have made rather than making them.

**Specific Task for Specific host in Group**

1. Suppose you running a playbook for group of 10 hosts. But you want particular task in playbook should run only on particular host ,then you should use delegate\_to option.

Ex. - task:

-Name: install nginx

apt: name=nginx state=present

delegate\_to: 172.27.59.17

1. Suppose you running a playbook for group of 10 hosts. But you want particular task in playbook should run only once on particular host ,then you should use delegate\_to option.

Ex. - task:

-Name: install nginx

apt: name=nginx state=present

run\_once: true

delegate\_to: 172.27.59.17

**Prompts**

When running a playbook, you may wish to prompt the user for certain input, and can do so with the ‘vars\_prompt’ section.

Ex. ---

- hosts: all

remote\_user: root

vars:

from: "camelot"

vars\_prompt:

name: "what is your name?"

quest: "what is your quest?"

favcolor: "what is your favorite color?"

**Tags**

If you have a large playbook it may become useful to be able to run a specific part of the configuration without running the whole playbook.

**Vault**

“Vault” is a feature of ansible that allows encryption of ansible files and data