

Ansible

Introduction

- System Admins used to manage servers by hand, installing software, changing configurations, and administering services on individual servers.
- System administrators manage servers by logging into them via SSH, making changes, and logging off
- Some of these changes would be documented, some would not.
- If an admin needed to make the same change to many servers (for example, changing one value in a config file), the admin would manually log into each server and repeatedly make this change
- Most servers have complicated firewalls and dozens of tweaked configuration files
- Even with change documentation, the manual process usually results in some servers or some steps being forgotten

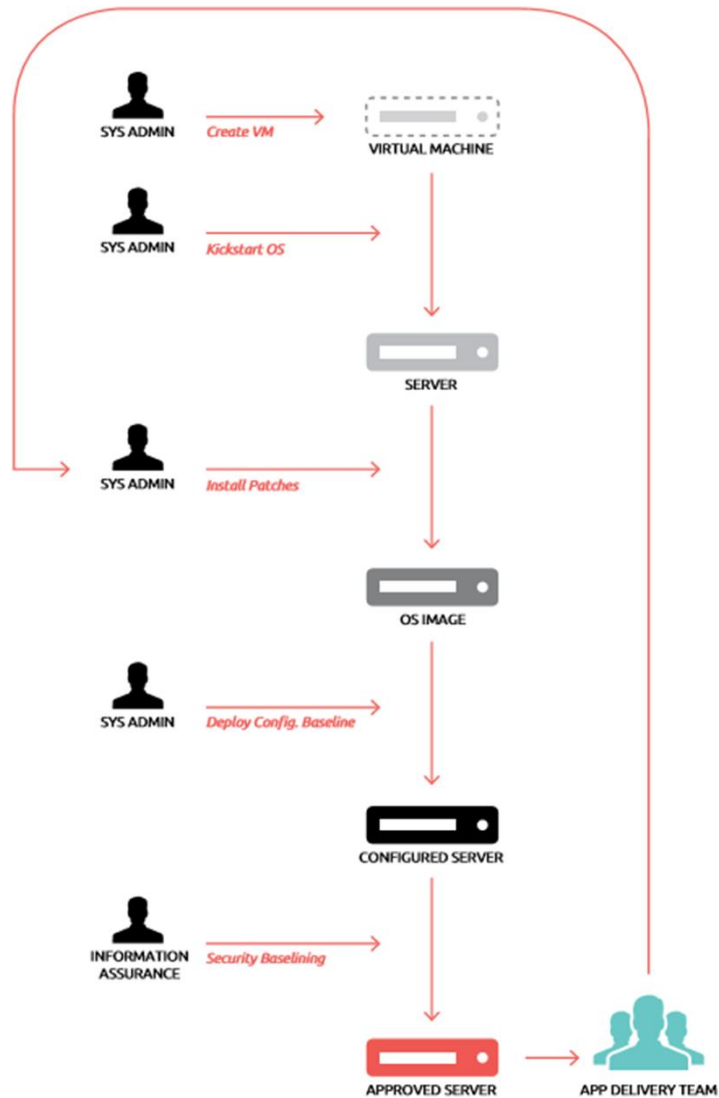
Introduction contd..

- If the admins at these companies wanted to set up a new server exactly like one that is currently running, they would need to spend a good deal of time going through all of the installed packages, documenting configurations, versions, and settings; and they would spend a lot of unnecessary time manually reinstalling, updating, and tweaking everything to get the new server to run close to how the old server did
- Some admins may use shell scripts to try to reach some level of sanity - : it's simple and easy-to-use, and they've had years of experience using bash and command-line tools
- Ansible was built by developers and sysadmins who know the command line—and want to make a tool that helps them manage their servers exactly the same as they have in the past, but in a repeatable and centrally managed way.

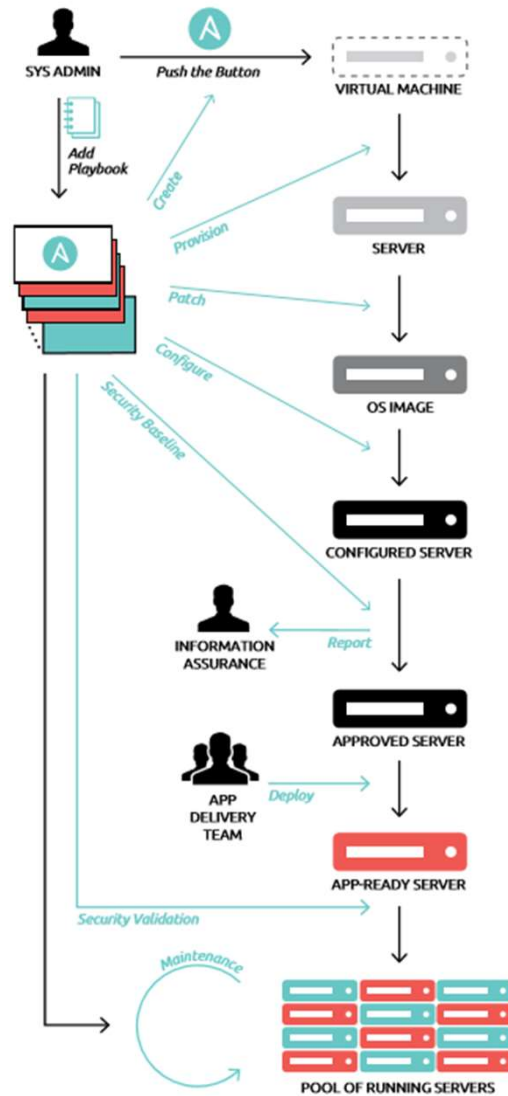
What is Ansible?

- Ansible is an open source IT Configuration Management, Deployment & Orchestration tool
- Ansible was released in 2012 by Michael DeHaan , a developer who has been working with configuration management and infrastructure orchestration
- One of Ansible's greatest strengths is its ability to run regular shell commands verbatim, so you can take existing scripts and commands and work on converting them into idempotent playbooks
- Some admins who were comfortable with the command line, but never became proficient in more complicated tools like Puppet or Chef (which needs knowledge of Ruby), Ansible comes to rescue
- Ansible works by pushing changes out to all your servers and requires no extra software to be installed on your servers (thus no extra memory footprint, and no extra daemon to manage), unlike most other configuration management tools

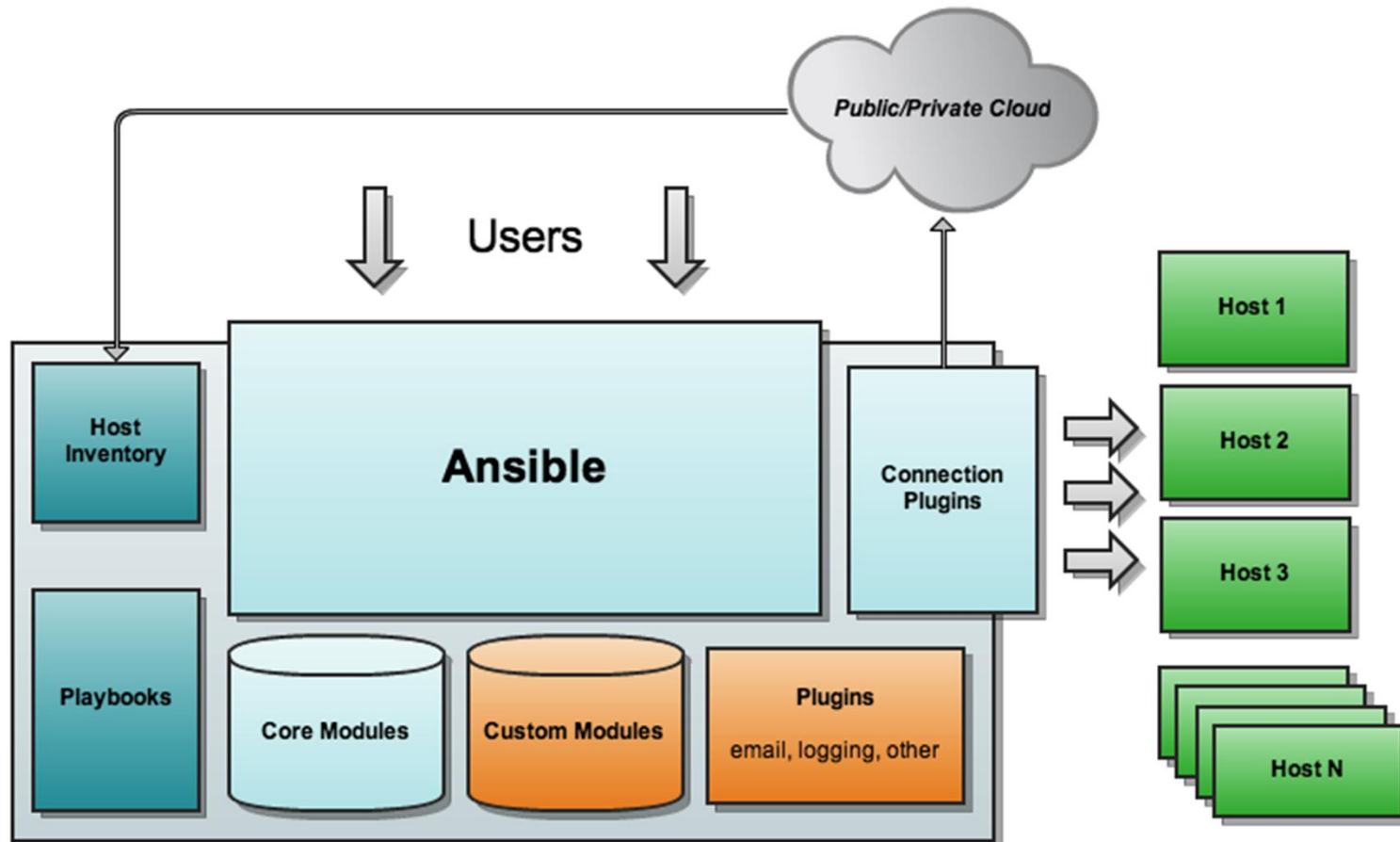
Without Ansible



With Ansible



Ansible Architecture



Common Queries

- I don't have Python in my environment and introducing Ansible would bring in Python. What do I do?
 - on most Linux systems, a version of Python is present at boot time and you don't have to explicitly install Python
- Can I use Ansible if I am starting afresh, have no automation in my system?
 - Yes, Ansible is perfect for you. Its simple and has easier learning curve than other tools
- I have other tools in my environment. Can I still use Ansible?
 - A case in point is a puppet shop that uses Ansible for orchestration and provisioning of new systems but continues to use Puppet for configuration management

Hands On – Provision the machines

- Create a directory called vms/centos7
- Copy and paste the Vagrantfile - provided

```
Vagrant.configure("2") do |config|
  config.ssh.insert_key = false
  config.vm.box_check_update = false

  config.vm.provider :virtualbox do |vb|
    vb.customize ["modifyvm", :id, "--memory", "256"]
  end

  # Controller.
  config.vm.define "controller" do |cont|
    cont.vm.hostname = "controller.dev"
    cont.vm.box = "geerlingguy/centos7"
    config.vm.network "public_network"
  end

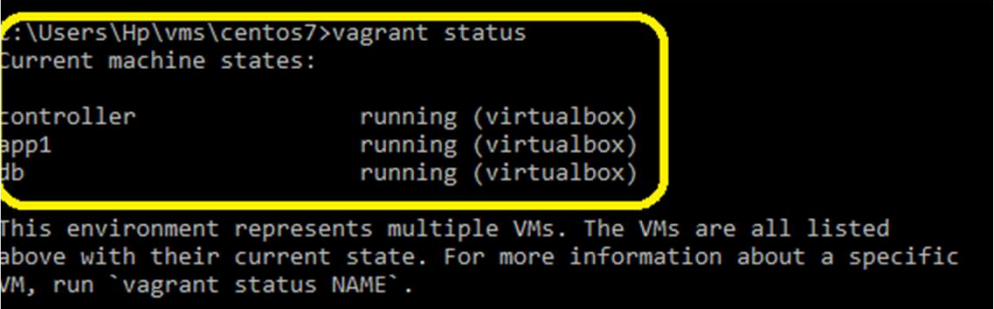
  # App server.
  config.vm.define "app1" do |app|
    app.vm.hostname = "app1.dev"
    app.vm.box = "geerlingguy/centos7"
    config.vm.network "public_network"
  end
end
```



C:\Users\Hp\vm\
-dhcp-public\Vagr

Hands On – Provision the machines contd

- Change directory to centos7 and type the following command
- *Vagrant up*



```
C:\Users\Hp\vm\centos7>vagrant status
Current machine states:

controller          running (virtualbox)
app1                running (virtualbox)
db                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

- *Wait till 2 VMs show up as “running”*
- *One machine will be used as controller – on this machine Ansible will be installed.*

Check ip Address of Controller

- Vagrant ssh controller

```
C:\Users\Hp\vm\centos>vagrant ssh controller
Last login: Mon Feb 26 15:58:42 2018 from 10.0.2.2

vagrant@controller ~]$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe82:7325 prefixlen 64 scopeid 0x20:::
    ether 08:00:27:82:73:25 txqueuelen 1000 (Ethernet)
    RX packets 875 bytes 88689 (86.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 658 bytes 85967 (83.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.42 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe4c:48b2 prefixlen 64 scopeid 0x20:::
    ether 08:00:27:4c:48:b2 txqueuelen 1000 (Ethernet)
    RX packets 170 bytes 26167 (25.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25 bytes 2746 (2.6 KiB)
```

- Once the IP address is known – use putty to login to server
- Ping google.co.in to check connectivity to internet

Installing Ansible on Controller Machine

- Ansible's only real dependency is Python. Once Python is installed, the simplest way to get Ansible running is to use pip, a simple package manager for Python
- `$ sudo pip install ansible`
- Fedora/RHEL/CentOS:
 - `$ sudo yum install epel-release`
 - `$ sudo yum -y install ansible`
- Debian/Ubuntu:
 - `$ sudo apt-add-repository -y ppa:ansible/ansible`
 - `$ sudo apt-get update`
 - `$ sudo apt-get install -y ansible`
- Once Ansible is installed, make sure it's working properly by entering `ansible --version` on the command line
- `$ ansible --version`

Ansible – Inventory File

- Ansible uses an inventory file (basically, a list of servers) to communicate with your servers
- Ansible inventory file matches servers (IP addresses or domain names) to groups
- Create (if it does not exist) a file at [/etc/ansible/hosts](#) (the default location for Ansible's inventory file)
- Install a user friendly editor like vim
 - Sudo yum install -y vim
- Make the following entry into the file and save
 - [app]
 - 192.168.1.39
- Try your first Ansible command
 - ansible app -a "hostname"
- You will get an error as permission denied.
 - ansible app -b -k -a "hostname"
 - Option -b is become Super User, option -k is ask for password
 - You will be prompted for password, then command successfully runs

Ansible ad-hoc commands

- Instead of typing out the password, this can be configured in the hosts file
- Make the following entry inside the `/etc/ansible/hosts` file
 - `[db]`
 - `192.168.1.40`

 - `[multi:children]`
 - `app`
 - `db`

 - `[multi:vars]`
 - `ansible_connection=ssh`
 - `ansible_ssh_user=vagrant`
 - `ansible_ssh_pass=vagrant`
- Now you need not use the `–s` and `–k` option
 - `ansible multi -a "hostname"`

Parallelism of Ansible

- `ansible multi -a "hostname"`
- Notice Parallelism of Ansible. You may have noticed that the command was not run on each server in the order you'd expect
- By default, Ansible will run your commands in parallel, using multiple process forks, so the command will complete more quickly
- Run the same command again, but this time, add the argument `-f 1` to tell Ansible to use only one fork (basically, to perform the command on each server in sequence):
- `ansible multi -a "hostname" -f 1`

Ansible – Modules

- Copying file to all nodes
 - `ansible app1 -m copy -a "src=.... dest=....."`
- Removing a directory on all nodes
 - `Ansible app1 -m file -a "dest=/home/vagrant/to-delete state=absent"`

Using Ansible Modules

- Install the NTP daemon on the server to keep the time in sync
 - *ansible multi -b -m yum -a "name=ntp state=present"*
- Now we'll make sure the NTP daemon is started and set to run on boot
 - *ansible multi -b -m service -a "name=ntpd state=started enabled=yes"*
- # also can be done using
 - *ansible multi -b -a "service ntpd restart"*
- Now lets write a simple play book to do the same
 - *Create a file by the name ntpplaybook.yml*
 - *Make the following entries in the playbook*

First Ansible playbook contd..

- hosts: all

 - become: yes

 - tasks:

 - name: Ensure NTP (for time synchronization) is installed.

 - yum: name=ntp state=present

 - name: Ensure NTP is running.

 - service: name=ntpd state=started enabled=yes

Running the playbook

- `ansible-playbook ntpplaybook.yml`

```
[vagrant@controller ~]$ ansible-playbook ntpplaybook.yml

PLAY [multi] *****

TASK [Gathering Facts] *****
ok: [192.168.1.43]
ok: [192.168.1.47]

TASK [Ensure NTP (for time synchronization) is installed.] *****
ok: [192.168.1.47]
ok: [192.168.1.43]

TASK [Ensure NTP is running.] *****
changed: [192.168.1.43]
changed: [192.168.1.47]

PLAY RECAP *****
192.168.1.43      : ok=3    changed=1    unreachable=0    failed=0
192.168.1.47      : ok=3    changed=1    unreachable=0    failed=0

[vagrant@controller ~]$
```

Playbook - explained

- ---
 - This first line is a marker showing that the rest of the document will be formatted in YAML
- - hosts: all
 - This line tells Ansible to which hosts this playbook applies. all works here, since Vagrant is invisibly using its own Ansible inventory file
- become: yes
 - we need privileged access to install NFS and modify system configuration, this line tells Ansible to use sudo for all the tasks in the playbook
- tasks:
 - All the tasks after this line will be run on all hosts (or, in our case, our one VM). Ansible uses "modules" to accomplish most of its Tasks