*Ansible in 24 hours*

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

<!@>//||\\<@!>/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\<!@>//||\\<@!>

***Preface:***

Dive into DevOps tools with this immersive 24-hour course with Ansible.

Ansible is the most powerful or one of the most used configuration management.

***Course Contents:***

1. Introduction

* What is Ansible, an Introduction?
* Ansible vs. Other Configuration Management Tools
* Ansible and DevOps

1. Installation and Configuration:

* Download Ansible
* Set up Ansible Environment
* Deploy Ansible Server
* Configure Ansible Server

1. Ansible Playbooks:

* Delegations
* Local Actions
* Rolling Updates

1. Ansible Modules:

* Module Community
* Installing a module
* Common modules
* Module Demo

1. Ansible Tower
2. Ansible Galaxy
3. How to test with Ansible
4. How to read Yaml
5. Ansible Use Cases

* Website Config Deployment: Apache
* Website Config Deployment: Nginx
* Deployment code: git
* Managing and deploying Docker

***Introduction to Ansible:***

* Ansible in short is an IT automation, configuration management and provisioning tools.
* It uses Playbooks to deploy manage build test and configure anything form full server environments to websites to custom compiled source code for applications.
* It brings together aspects of environment management that have been traditionally separate and managed independently.
* Ansible is the easiest way to deploy, manage, and orchestrate computer systems you've ever seen. You can get started in minutes.
* Ansible is an extra-simple tool/framework/API for doing 'remote things' over SSH.
* 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.

***Background***

* Ansible was created and is run by Michael DeHaan, a Raleigh, NC based software developer and architect, who also created the popular open-source DevOps install server Cobbler. Cobbler is used to deploy mission critical systems all over the planet, in industries ranging from core Internet infrastructure, finance, chip design, massively multiplayer gaming, and more.
* Michael also co-authored Func, a precursor to Ansible, which is used to orchestrate systems in lots of diverse places. He’s worked on systems software for IBM, Motorola, Red Hat’s Emerging Technologies Group, Puppet Labs, and is now with rPath. Reach Michael by email here
* Ansible is open source software (GPLv3), and is developed by a large group of industry-experts from all over the world.

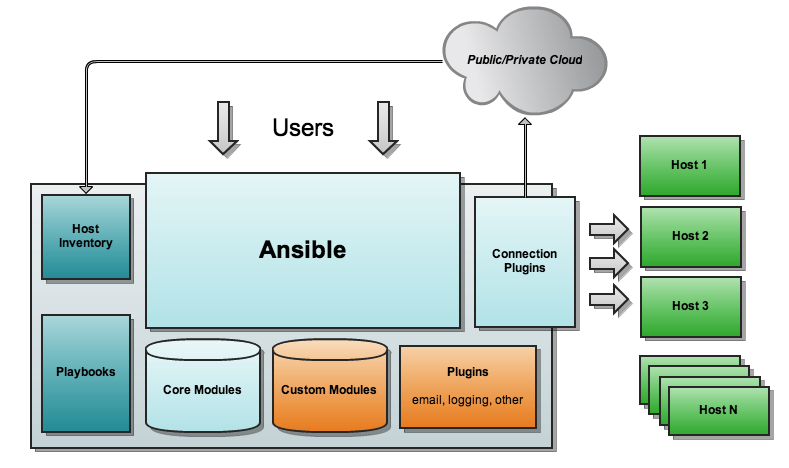
***Features***

* Management over SSH  
  you don't need to install any daemons. You don't need to wrestle with a custom PKI solution. Add a Ruby stack? No need. Ansible manages machines over SSH, something you've already got out there, and you know to be secure. Need to use Kerberos or SSH Jump Hosts? No problem! Want to go faster? Ansible 0.8 can bootstrap an ephemeral 0mq connection!
* Developer Friendly  
  Configurations are text. Ansible modules can be written in any language -- if you would like to add extensions in bash, Python, Ruby, or even C, you are welcome to do so. Inventory can be pulled in from external sources like EC2, Open Stack, and Cobbler!
* Dead simple  
  In addition to not requiring any daemons or bootstrapping, Ansible Playbook language is the simplest systems management language out there. It reads like English. We believe you have other work to do, so we want you to get things done quickly and get out of your way.

***Ansible Architecture:***

Ansible runs on a central computer. Playbooks define configuration policy and orchestration workflows. Ansible then uses SSH to execute modules on remote machines without having to install any systems management software. Ansible comes with a large selection of modules for automating common tasks, and users can also write their own in their choice of favorite language. Inventory can be sourced from simple text files, the cloud, or configuration management databases (CMDBs). Results can be stored and processed into a variety of systems.

Src: https://terry.im/wiki/terry/attachments/29786135/37552129.png



History of traditional deployment issues in comparison with the current deployment advantages:

* I don’t need to have an agent
* My server can act as a client to other server
* Push based and running over there
* In the case of chef and puppet they are pull based
* It comes with core and custom modules. And there are many plugins that can be attached to the Ansible.
* Ex: You can use the password less Ssh plugin.
* UDeploy works based on the server and agent model.
* Ansible master will talk with the client.
* Depends on the situation a server can be a client and client can become a server.
* You created a golden image in your organization, by installing any other critical fixes.
* Python, Ansible
* Ansible uses ssh

***Traditional in UDeploy:***

Server --------🡪 Agent

|

|------------------🡪 Agent

|

|-----------------🡪 Agent

* Agent can communicate by using Http or TCP. No need to use SSH.
* The script can be passed over TCP/IP to the agent from the server and the script can be run on the agents.
* Commercial and cost played a crucial role.
* UDeploy is more specific to the J2EE Infrastructure.
* Ansible/Chef are open source.
* Chef is good for Java developers.
* Ansible is made over the python.
* The layer, which I’m going to write in Ansible, is written in Yaml and not python.
* By using Ansible as it a Push based
* Push based means you’re pushing data to the server in order to do some action.
* Pull based means the server is pulling from the agent.

Taking control of your environment with a single tool has a number of advantages.

With Ansible, you can control server deployment configuration, making everything consistent. With modules and plugins, you can build or ‘hook’ into other applications and control them as well!

Configuration management is a complex topic but it is the heart of the DevOps tools.

**Ansible vs. Others Tools:**

Configuration items, builds, databases backups and updates

Other tools which are available in the market.

* Jenkins
* Salt
* Puppet
* Chef
* Fabric
* Ansible is very often called an orchestration tool as it can function independently as well as ‘control’ one or more of the tools listed above.

**Ansible vs. World:**

* Puppet and chef will have a master or controller server in the server
* Ansible operations only with Ssh. Any system with Ansible can function in that role at any time based on the task or deployment type.

**Workflow: Push or Pull?**

* Since most configuration management tools have a ‘master’ server, they use the pull method.
* Ansible uses the push method.
* Puppet are not top to bottom
* Ansible are top to bottom and is easy to read.
* Many automation utilities are used in a variety of ways.
* Ansible is built on python.
* Puppet uses ruby.
* Chef uses ruby
* Puppet, Salt or Chef have a lot of modules.
* Chef has a lot modules included in it.
* Puppet is based on custom DSL while Ansible is based on YAML.
* There is no concept of functions in Ansible.
* Chef lo you’ll have functions, you’ll have wrapper class around it and it is used. You invoke these functions by writing another script.
* Ansible is based on YAML.

It brings together multiple tools.

Easy to understand because YAML is readable.

*Ansible + Docker + Jenkins*

***YAML***

Yet Another Markup Language 🡪 YAML

YAML Ain’t Markup Language

Human Readable data serialization format.

***Structure*:**

YAML most closely resembles an outline or list of things with basic descriptions.

---# Our Fav Movies of ALL time

- The Terminator

- Star Trek

- Star Wars

Python is very good at processing the text.

YAML is a very powerful configuration tool for our DevOps toolkit. It is perfect candidate for compiling and running complex configuration and deployment tasks in our Ansible Playbooks.

***Ansible***

***YAML EXAMPLES***

Simple Lists:

---# Grocery list

* Banana
* Apples
* Oranges
* Cereal
* Eggs

Inline formatting like:

[Bananas, apples, oranges, Cereal, Eggs]

Associative Arrays/Key Value pairs

---# Employee Info (List)

Name: John smith

Age: 44

HireDate: 09/01/2011

---# Employee Information (Inline)

{Name: John Smith, Age: 44, HireData: 09/01/2011}

Note: Strings does not require quotation

---# New Line preservation

typing: |

Now this is the time for all good men to

Come to the aid of their country. Now for a

Famous quote to display

Four score and seven years ago

---# New Line Folding

Each item in this

Will be folded into

A single paragraph

Note: “Blank line specifies a new paragraph”

***YAML Characteristics:***

Casting datatypes: Normally, Yaml will detect datatype, but you may want to specify

---  
A: 123 # an integer

B: “123” # a string

C: 123.0 # a float

D: !!Float 123 # also a float via explicit data type prefixed by !!

E: !!Str 123 # a string disambiguiltated by explicit type

F: !!Str Yes #a string via explicit type

G: Yes # A Boolean True (yaml1.1), string “Yes”(yaml1.2)

H: Yes sir we have no bananas today # Yes and No disambiguated by context

PlayBook Example (Ansible)

---# PlayBook Description

-hosts: appserver

tasks:

-name: Install Lynx on App Server

yum: pkg=lynx state=installed update\_cache=true

-name: Querying for Telnet

yum: pkg=telnet state=present update\_cache=true

This Playbook, when run, will look for a group of hosts called ‘appserver’ and then try to install lynx and query if telnet is installed using the ‘yum’ repository.

***Ansible Documentation***

/usr/local/bin

ansbile is installed here.

🡪 Ansible DOC EC2

3 instances

3 users

Test

Ansible2

Ansible3

Create id\_rsa, id\_rsa.pub

Cat id\_rsa.pub >> authorized\_keys

[ansible2@ip-172-31-27-88 .ssh]$ rm id\_rsa.pub

[ansible2@ip-172-31-27-88 .ssh]$ vi id\_rsa.pub

[ansible2@ip-172-31-27-88 .ssh]$ vi id\_rsa

[ansible2@ip-172-31-27-88 .ssh]$ cat id\_rsa.pub >> authorized\_keys

[ansible2@ip-172-31-27-88 .ssh]$ more authorized\_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCeGD0MSOVVCOa3TmZVuK3I0Z/BX+h/FQw54CVmgZAHcAjNLb5m/1jIZLFX9ATmu5etWqqv2pVbTJ1BtErkdr2buB0MYPJofUjmgl19SzBPPWAIjlWvLpncG4AKxis+6hEVbC4rcmP+Afxma

abkWzovO6iQegSNwskpxVemQC9LD/R2qaUC1ymFly8qtM0ZsZ9HNjlonSd+H4UpFQ3BM2DfNZu7crRslkIaWm/vzys+xMd4xBXo8/I/l6jKzQYttPeaiDN7eh4qkX4zqk9yYIKCGjq+GGWzM1O2IsPDEJaVwdTOOSBHCOvBE31xDGYptYaBcE

q/RzOm9AlY3AYpmLL7 test@ip-172-31-21-234.ec2.internal

[ansible2@ip-172-31-27-88 .ssh]$ chmod 600 authorized\_keys

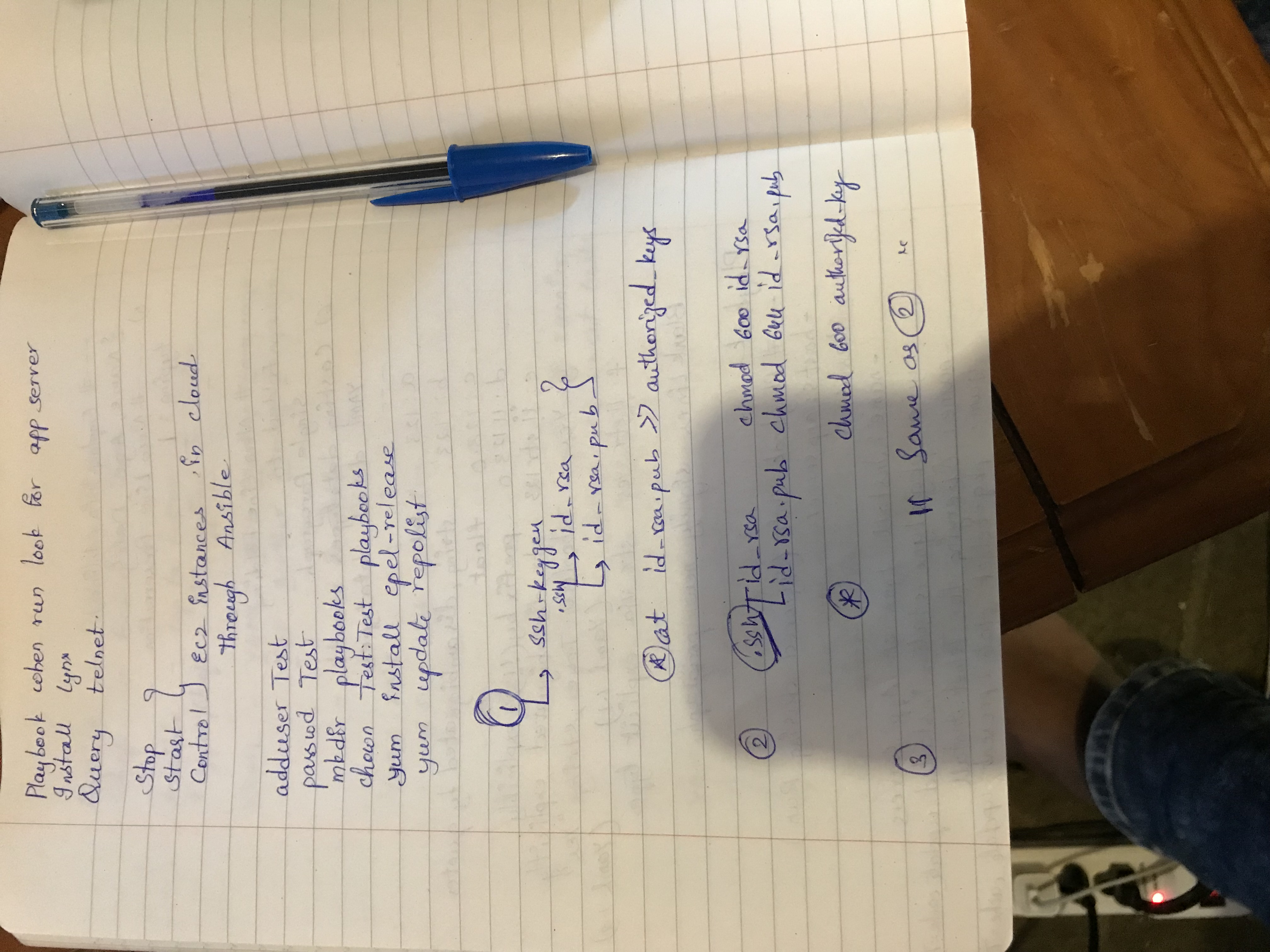
[ansible2@ip-172-31-27-88 .ssh]$ chmod 600 id\_rsa

[ansible2@ip-172-31-27-88 .ssh]$ chmod 644 id\_rsa.pub

[ansible2@ip-172-31-27-88 .ssh]$ ssh test@172.31.21.234

Mkdir playbooks in three users home directory and change the ownership of the directory.

http://nokitel.im/index.php/2016/11/09/how-to-install-ansible-on-red-hat-enterprise-linux-7-rhel-7/



***Step 0 – Prepping RHEL 7***

If you have just downloaded the RHEL 7 ISO as part of the developer subscription model we will need to perform some initial tasks to get it up to scratch

Authenticate your subscription  
subscription-manager register --username your\_username --password your\_password

Update the OS  
yum update && yum upgrade

Let’s install some classics such as ifconfig & wget

Shell

|  |  |
| --- | --- |
| 1  2 | yum install net-tools  yum install wget |

We also need access to the “[Extra Packages for Enterprise Linux](https://access.redhat.com/solutions/3358)” for downloading packages outside the normal channels

Shell

|  |  |
| --- | --- |
| 1  2  3 | wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm  rpm -ivh epel-release-latest-7.noarch.rpm  yum install epel-release |

So at this point we should have a RHEL 7 instance ready to go.

***Step 1 – Installing Python, Python Tooling & Ansible***

We’ll install Python first  
sudo yum install python

Next we’ll install PIP which is a “package management system used to install and manage software packages written in Python”  
sudo yum install python-pip

Now we need to upgrade it  
sudo pip install --upgrade pip

Next we’ll use PIP to install some liraries and modules for python & Ansible

Shell

|  |  |
| --- | --- |
| 1  2  3 | sudo pip install markupsafe  sudo pip install xmltodict  sudo pip install pywinrm |

And finally we will install Ansible  
sudo yum install Ansible

Sudo pip install Ansible

To confirm that it has been installed correctly we’ll execute  
ansible --version  
Which should generate the following output

Shell

|  |  |
| --- | --- |
| 1  2  3  4 | [asecor@labansc]$ ansible --version  ansible 2.1.2.0    config file = /etc/ansible/ansible.cfg    configured module search path = Default w/o overrides |

***Ansible Configuration File***

Cd /etc/Ansible

Ansible.config

Hosts

Roles

Any client can function as a server, any server can function as a client.

The Ansible structure can be modified in the Ansible configuration file.

Network shared drive between your server so that playbooks can be used in any of the server

Hosts file consists of hosts and that can be expressed as groups

**Ansible.cfg 🡪**

Ansible can be configured to be used by user or system.

Ansible.cfg in cwd || in home directory. /Ansible || /etc/Ansible folder

Nearly all parameters can be overridden in Ansible-playbook

or with command line flags. Ansible will read ANSIBLE\_CONFIG,

Ansible.cfg in the current working directory, .Ansible.cfg in

The home directory or /etc/Ansible/Ansible.cfg, whichever it

Finds first

**Python and its Dependencies in the Ansible in our environment:**

Control Server --- Web Client --- Application Client

Python 2.4 is the min requirement

Ansible 1.9.2 does not work with Python 3

Whereis python

Which python

#! /usr/bin/python

print “hello world”;

**python helloworld.py**

output: hello world

* Copy the hosts onto hosts.original
* Create a new hosts file
* [local]
* localhost
* localhost.localdomain
* 127.0.0.1
* Just use one in the above three!
* 172.31.39.145 🡪 Control server
* 172.31.45.99 🡪 Web Client
* 172.31.42.237 🡪 Application Client
* **ansible all --list-hosts**
* 🡪 Shows the hosts
* 172.31.45.99
* 172.31.42.237
* localhost

**Over riding the hosts file:**

~/hosts

[appserver1]

172.31.45.99

ansible appserver1 -i ~/hosts -m ping

This box is the file where it can find the appserver1.

Normally without the –I it’ll look in the path for the appserver1 specified in the Ansible.cfg in the /etc/Ansible/

*Order of Execution:*

* *Env\_var*
* *Current directory*
* *.ansible/Ansible.cfg*
* */etc/Ansible/Ansible.cfg*

***ANSIBLE PLAYBOOKS***

Push based system is Ansible.

Sudo visudo

Root: ALL(ALL) ALL

Sagar: ALL(ALL) NOPASSWD: ALL

Restart

Or else: /etc/sudoers 🡪 ADD the user and say this as the root

--------------------------------------------------------------------------------------------------------------------------

Ansible.cfg make the edit to the roles so that the roles or the playbooks in your home directory wil be run first rather than the one in the system etc

/home/sagar/Playbooks/Roles/roles :/etc/Ansible/Roles

# Additional paths to search for rolesin, colons separated

roles\_path = /home/test/Playbooks/Roles/roles:/etc/ansible/roles

ansible apacheweb -s -m shell -a 'apt list installed | grep python'

--------------------------------------------------------------------------------------------------------------------------

**System Facts**:

1. Ansible all –m ping

Will give the ping of all the hosts defined

1. Ansible appserver –m ping

Will run on that particular server and gives pong

Ansible local –m setup | more

It will read all the facts! What are the ip address what is the architecture system?

**Ansible local –m setup –tree /tmp/facts**

**Ansible local –m setup –t /tmp/facts**

**Ansible apacheweb –m setup –t /tmp/facts\_apacheweb**

**Ansible appserver –m setup –a ‘filter=\*ipv4’**

**Ansible appserver –m setup |grep ipv4**

When we run the facts we get a JSON file.

We get ipv4 address. This will be very useful.

Ansible all 🡪 gives a whole bunch of facts for all three and will be difficult to read.

For example if you have a lot and lots and servers.

There may be bunch of information that you need to know.

Distribution and version in all Servers I need to Know.

Lets do this:

Ansible apacheweb –m setup –a ‘filter=Ansible\_architecture’

Review:

-m 🡪 ping, shell, setup, yum/apt,

-s 🡪 sudo

--------------------------------------------------------------------------------------------------------------------------

**Writing the first playbook:**

**Appserver.yml**

- Hosts: appserver

Tasks:

- Name: Install Lynx on App Server

Apt: pkg=lynx state=installed update\_cache=true

Without the playbook to install:

ansible apacheweb –s –m yum –a ‘pkg=lynx state=installed update-cache=true’

**🡪 Ansible-playbook –s appserver.yml**

Facts:

ansible apacheweb -m setup -a "filter=ansible\_\*"

Facts related to distribution:

ansible apacheweb -m setup -a "filter=ansible\_architecture”

**Variables in playbooks:**

- Hosts: appserver

vars-files:

author\_name: Test user

vars:

control\_server: x.x.x.x

web\_root: /var/www/html

or

vars:

- vars.yml

Tasks:

- Name: Install Lynx on App Server

Apt: pkg=lynx state=installed update\_cache=true

Vars.yml 🡪

Control\_server: x.x.x.x

Web\_root: /var/www/html

* hosts:

user:

sudo:

connection:

gather\_facts:

The host is connected via connection to the user.

--- # My first YAML Playbook for Ansible

- hosts: apacheweb

user: sagar

sudo: yes

connection: ssh

gather\_facts: no

vars:

playbook\_version: 0.1b

vars\_files:

- conf/copyright.yml

- conf/webdefaults.yml

vars\_prompt:

- name: web\_domain

prompt: Web Domain

tasks:

- name: Install HTTPD

action: apt name=apache2 state=installed

notify: Restart HTTPD

handlers:

- name: Restart HTTPD

action: service name=apache2 state=restarted

~/Playbooks/conf/copyright.yml

---

message: Copyright 2015 by pitem

author: Prashanth

~/Playbooks/conf/webdefaults.yml

apache\_version: 2.6

apache\_mod\_req: mod\_ssl

**Handler Section:**

*Notify invokes the handlers.*

Once run and it wont trigger again.!!

The above is the example!

**Skeleton of a example playbook:**

- webservers

- sagar user

- sudo rights

- install the apache web server

start the web service

-verify that the web service is running

- Install client software

- telnet

- lynx

- log all apackages install in the system!

- date/time stamp for when the playbook ends

To see the modules:

Ansible-doc

**Local Actions:**

--- #Local Action playbook

- hosts: 127.0.0.1

connection: local

tasks:

- name: Install Telnet Client

apt: pkg=telnet state=latest

* 127.0.0.1 is a loopback ip address.
* Install on local machine

**Create a List of Users:**

--- #Loop Playbook Example

- hosts: apacheweb

user: sagar

sudo: yes

connection: ssh

gather\_facts: no

tasks:

- name: Add a list of users

user: name={{ item }} state=present

with\_items:

- user1

- user2

- user3

* Three users will be created at the location.

**Until Example:**

--- # UNTIL Example

- hosts: apacheweb

sudo: yes

connection: ssh

user: sagar

gather\_facts: no

tasks:

- name: Installing apache web server

apt: pkg=apache2 state=latest

- name: Verify Service Status

shell: systemctl status apache2

register: result

until: result.stdout.find("active (running)") != -1

retries: 5

delay: 5

- debug: var=result

***Ansible Security***

Ansible enables us to encrypt the yaml files to enhance security.

* To create a encrypted file
* Ansible-vault create secure.yml
* Give a password
* Enter the data and then after it is saved the file is encrypted using AES 256 bit encryption
* To decrypt or edit the file
* Ansible-vault edit secure.yml
* Now the decrypted file opens after you enter the password
* To change the password
* Ansible-vault rekey --ask-vault-pass secure1.yml
* Give the old key and then it’ll ask for the new password
* To encrypt an existing file
* Ansible-vault encrypt A.yml B.yml
* You can encrypt for multiple files or single files after they are created.

**Commands**

* **ansible all --list-hosts**
* **ansible appserver1 -i ~/hosts -m ping**
* **Ansible all –m ping**
* **python helloworld.py**
* **ansible x.x.x.x -s -m shell -a 'apt list installed | grep python' \*\*\*\*\*\*\*\*\***
* **Ansible local –m setup | more**
* **ansible apacheweb -m setup -a 'filter=\*ipv4'**
* **ansible apacheweb -m setup -a 'filter=ansible\_distribution'**
* **ansible apacheweb -m setup -a 'filter=ansible\_distribution\_version'**
* **ansible apacheweb -m setup -a 'filter=ansible\_domain'**
* **ansible apacheweb -m setup -a 'filter=ansible\_fqdn'**
* **ansible apacheweb -m setup -a 'filter=ansible\_architecture'**
* **ansible apacheweb -m setup -a 'filter=ansible\_interfaces'**
* **ansible apacheweb -m setup -a 'filter=ansible\_kernel'**
* **ansible apacheweb -m setup -a 'filter=ansible\_memtotal\_mb'**
* **ansible apacheweb -m setup -a 'filter=ansible\_proc\*'**
* **ansible local -s -m shell -a 'apt-get install lynx'**
* **ansible apacheweb -s -m apt -a 'pkg=lynx state=installed update\_cache=true'**
* **Ansible-playbook –s appserver.yml**
* **Ansible-playbook –s –v test.yml**
* **Ansible all –m shell –a ‘apt-get install telnet’**
* **ansible-vault create secure.yml**
* **Ansible-vault edit secure.yml**
* **ansible-vault rekey --ask-vault-pass secure1.yml**