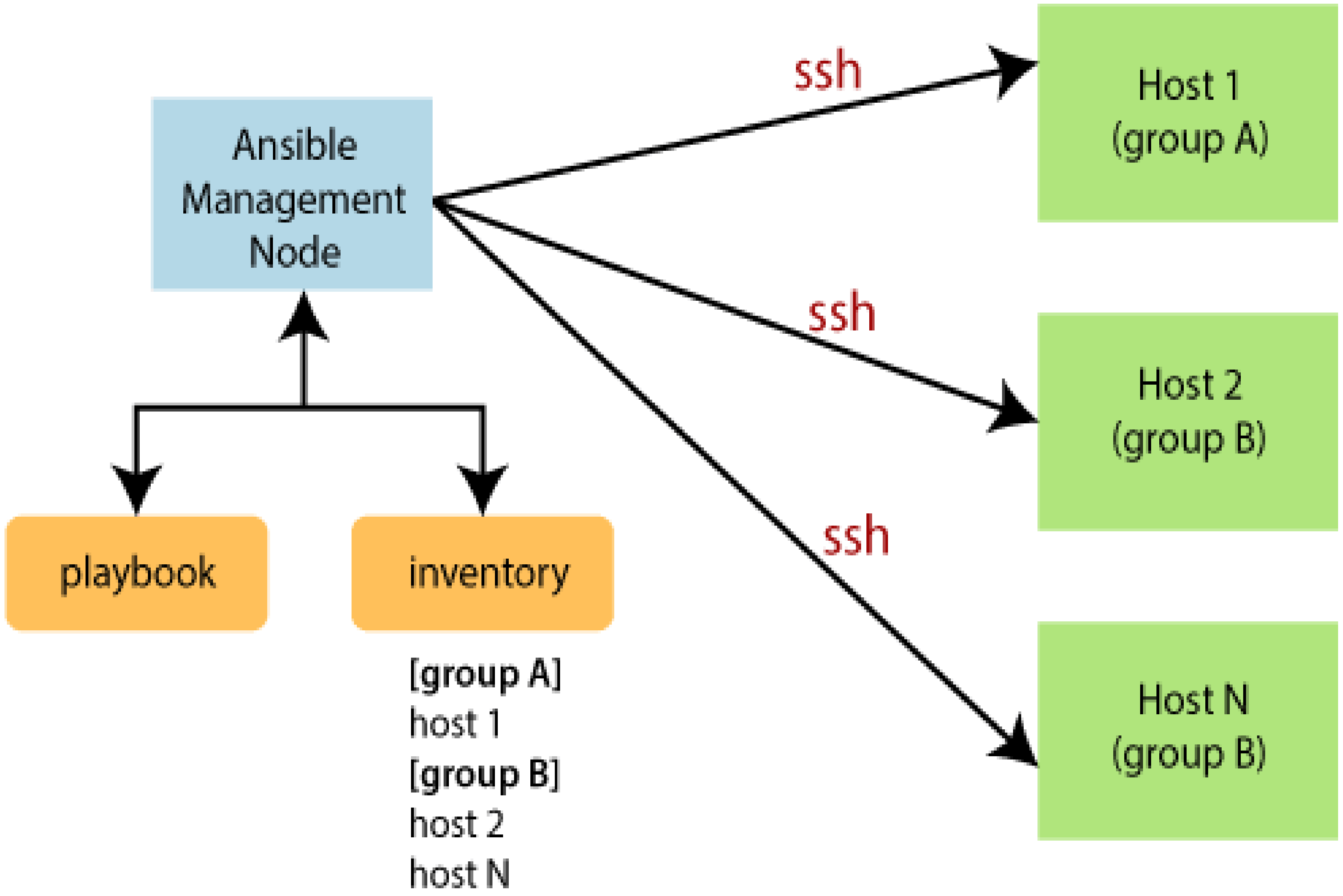
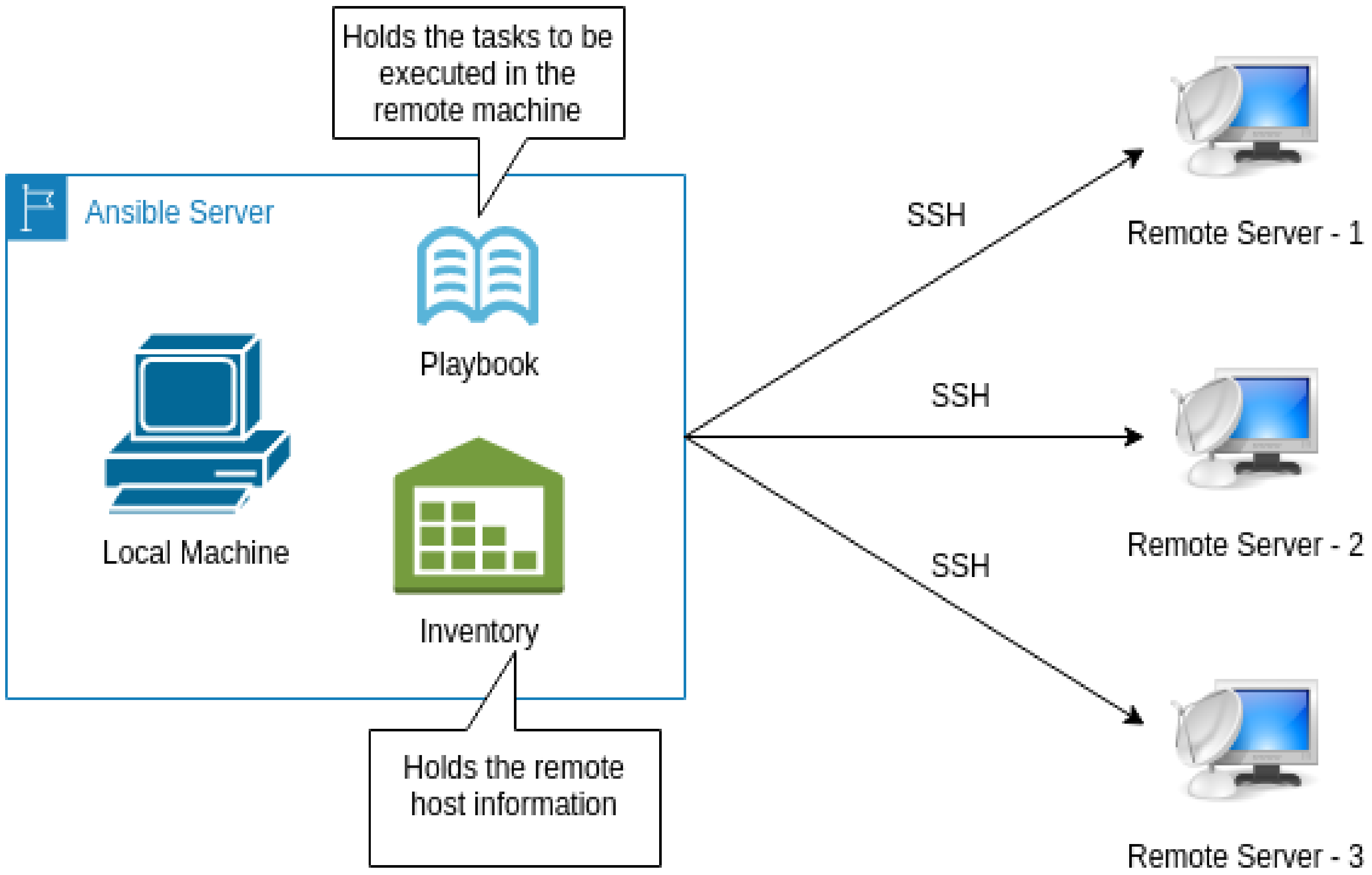
ANSIBLE

Ansible

- Ansible is an open-source platform used for automation and for various operations such as configuration management, application deployment, task automation, and IT orchestration.
- Ansible is easy to deploy because it does not use any agents or custom security infrastructure.
- It runs on Linux, Mac, or BSD.
- Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e.**YAML**
- Apart from the free version, it has an enterprise edition called 'Ansible Tower.'
- Ansible is completely agentless which means Ansible works by connecting your nodes through ssh(by default).



Ansible Management Node
playbook
inventory
[group A]
host 1
[group B]
host 2
host N
ssh
ssh
ssh
Host 1 (group A)
Host 2 (group B)
Host N (group B)


Holds the tasks to be executed in the remote machine
Ansible Server
Playbook
Local Machine
Inventory
Holds the remote host information
SSH
SSH
SSH
Remote Server - 1
Remote Server - 2
Remote Server - 3

What is Configuration Management

Configuration management in terms of Ansible means that it maintains configuration of the product performance by keeping a record and updating detailed information which describes an enterprise’s hardware and software.

Top 5 Configuration Management Tools
CHEF
puppet labs
ANSIBLE
CFEngine
SALTSTACK

Ansible Advantages

- ✓ 1) Ansible is that it is easy and free to use by everyone.
- ✓ 2) Ansible is very lightweight and consistent, and no constraints regarding the operating system or underlying hardware are present.
- ✓ 3) It is also very secure due to its agentless capabilities and due to the use of OpenSSH security features.
- ✓ 4) Its modularity regarding plugins, modules, inventories, and playbooks make Ansible the perfect companion to orchestrate large environments.

Linux Knowledge Required for Ansible

1) Linux Basic Commands

2) User and Group account

3) Compress and Decompress

4) File Security

5) Package management(yum and apt)

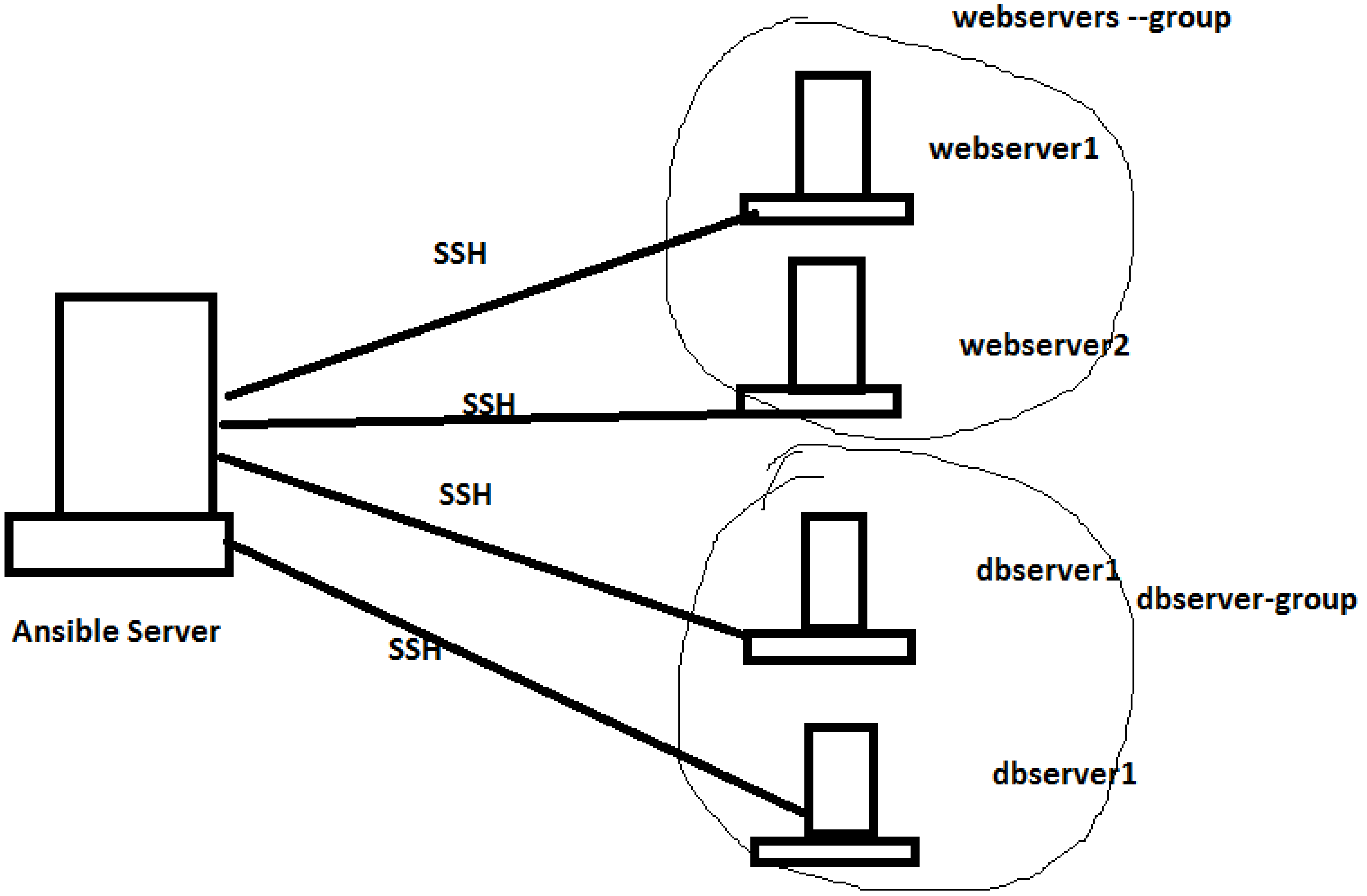
6) Firewall

7) Apache web server configuration

8) Tomcat web server configuration

9) SSH

10) SCP or RSYNC


webservers --group
webserver1
webserver2
SSH
SSH
SSH
SSH
Ansible Server
dbserver1
dbserver-group
dbserver1

Hands on: Steps Involved

1) Launch 3 Instance in AWS cloud –Connect it
2)Change Hostname
3)Change root password
4) Enable passwordlogin authentication
5) Configure passwordless Authentication
6) Ansible Installation
7) Adding Inventory
8) Ansible ad-hoc commands
9) Ansible modules
10) Running Playbooks

Hands on: Steps 1,2,3

1) Launch 3 Instances (RHEL based) in AWS ----- connect all using mobaextrem

2) Change hostname

```
# hostnamectl  set-hostname  ansibleserver

# hostnamectl  set-hostname  webserver

# hostnamectl  set-hostname dbserver
```

3) Change root password in all system

```
#passwd root
```

4) Enable passwordlogin authentication in webserver and dbserver(Remove #)

```
# vi /etc/ssh/sshd_config
PermitRootlogin yes
Passwordauthentication yes
```

Hands on: Steps 4,5

#systemctl restart sshd

# systemctl enable sshd

How to Check

From Ansible Server

# ssh   root@Webserver-private-IP

Yes -----Give password

#exit

Check same for remaining servers

5) Configure passwordless Authentication --do in Ansible server

# ssh-keygen

    4 times Enter

# ssh-copy-id   root@192.168.0.101   ---- Webserver private IP

# ssh-copy-id   root@192.168.0.102   ---- dbserver private IP

How to check

#ssh   root@Webserver-private-IP

#exit

Hands on: Steps 6 -Ansible Installation

Ansible Installation in Linux(RHEL/CentOS/Fedora)

```
# yum install epel-release

#  yum install -y ansible

# ansible  --version
```

Install ansible on Ubuntu/Debian systems

```
# sudo apt update

# sudo apt install software-properties-common

# sudo  apt-add-repository ppa:ansible/ansible

# sudo apt-get  update

 # sudo apt install ansible

# ansible  --version
```

Hands on: Steps 7 -- Adding Inventory

Adding Inventory

```
# vi  /etc/ansible/hosts
```

at the last of line add

```
[webservers]
192.168.0.101
192.168.0.102
[dbservers]
192.168.0.103
192.168.0.104
```

To check inventory

```
# ansible all --list-hosts
# ansible webservers  --list-hosts
```

Hands on: Steps 8 --Ansible ad-hoc commands

One of the simplest ways Ansible can be used is by using ad-hoc commands. These can be used when you want to issue some commands on a server or a bunch of servers. Ad-hoc commands are not stored for future uses but represent a fast way to interact with the desired servers.

```
# ansible -m ping   'webservers'

# ansible -m ping    'dbservers'

# ansible -m ping    'all'

# ansible -m command   -a "uptime"  all

# ansible -m command   -a "uname -r"    dbservers

# ansible -m command   -a "free -m"  'webservers‘
```

Here, m—module, a -- attribute

Hands on: Steps 8 --Ansible ad-hoc commands

```
# ansible -m command   -a "df - h"   192.168.0.102

#ansible  -m command   -a "df -th"    'dbservers'   >  /tmp/command-output.txt

# ansible -m command   -a  "useradd   deepak"    all

# ansible -m command   -a  "grep  deepak   /etc/passwd"    all

# ansible -m command   -a  "mkdir  /root/india"    all

# ansible -m command   -a  "ls  /root/"    all
```

Hands on: Steps 9 -Ansible modules

Modules (also referred to as “task plugins” or “library plugins”) are discrete units of code . Ansible executes each module, usually on the remote target node, and collects return values.

Ansible modules are standalone scripts that can be used inside an Ansible playbook. A playbook consists of a play, and a play consists of tasks

Ansible modules examples:

1) yum, dnf, apt : these modules can install, upgrade, downgrade, remove, and list packages.

2) Service: enables you to start, stop, and reload installed packages

3) Copy: copies a file from the local or remote machine to a location on the remote machine.

4) debug: prints statements during execution and can be useful for debugging variables or expressions without having to halt the playbook.

5) file: manages the file and its properties.

6) command: takes the command name followed by a list of space-delimited arguments.

7) git: manages git checkouts of repositories to deploy files or software.

Hands on: Steps 9 -Ansible modules

Some standalone examples of running ansible modules

```
#cd   /etc/ansible/
```

1) If you need to copy a file to multiple destinations rapidly, you can use the copy module in ansible which uses SCP

```
# ansible  -i  hosts all  -m copy -a  " src=/root/test_ansible/testfile
dest=/tmp/testfile"
```

2) How to install a package via the yum module on two Centos hosts.

```
# ansible  -i   hosts  all  -m  yum  -a 'name=httpd   state=present'
```

3) How to remove a package via the yum module on two Centos hosts.

```
# ansible -i  hosts  all -m  yum  -a 'name=httpd   state=absent'
```

Hands on: Steps 9 -Ansible modules

4) If you need detailed information about the systems to be modified via ansible, the next command can be used. The setup module gathers facts from the system variables.

```
# ansible -i hosts  all -m    setup
```

5) Restart http service in all webservers hosts

```
#ansible webservers  -m service -a "name=httpd state=started“
```

6) Ping and reboot

```
#ansible   webservers  -m ping
```

```
#ansible webservers  -m command -a "/sbin/reboot -t now"
```

Hands on: Steps 10 -Ansible Playbook

- ✓ Playbooks are the files where Ansible code is written.
- ✓ Playbooks are written in YAML format.
- ✓ YAML stands for Yet Another Markup Language.
- ✓ **Playbooks** are one of the core features of Ansible and tell Ansible what to execute.

Like the name is saying, a playbook is a collection of plays. Through a playbook, you can designate specific roles to some of the hosts and other roles to other hosts. By doing so, you can orchestrate multiple servers in very diverse scenarios, all in one playbook.

Different YAML Tags

1) name: pecifies the name of the Ansible playbook

2)hosts: targeting IP or group or all

3) vars: lets you define the variables

4) task: tasks are a list of actions one needs to perform.

Hands on: Steps 10 -Ansible Playbook Basic examples

1) Install httpd in centos

```
 # nano test1.yml
---- hosts: all
       tasks:
             - name: Install httpd
                yum: name=httpd state=present
```

2) UnInstall httpd in centos

```
 # nano test1.yml
---- hosts: all
       tasks:
             - name: Install httpd
                yum: name=httpd state=absent
```

Hands on: Steps 10 -Ansible Playbook Basic examples

3) Install httpd in ubuntu

```
 # nano test1.yml
---- hosts: all
      become: true
      tasks:
           - name: Update apt-cache
             apt: update_cache=yes
           - name: Install apache2
             apt: name=apache2 state=latest or present
```

4) For uninstalltion

Change the state : absent

How to run

```
#ansible-playbook   test1.yml
```

Hands on: Steps 10 -Ansible Playbook Basic examples

5) To allow http in ubuntu firewall

```
#nano test2.yml
---
- hosts: all
  become: true
- name: Allow all access to tcp port 80
      ufw:
        rule: allow
        port: '80'
        proto: tcp
```

How to run

```
#ansible-playbook   test1.yml
```