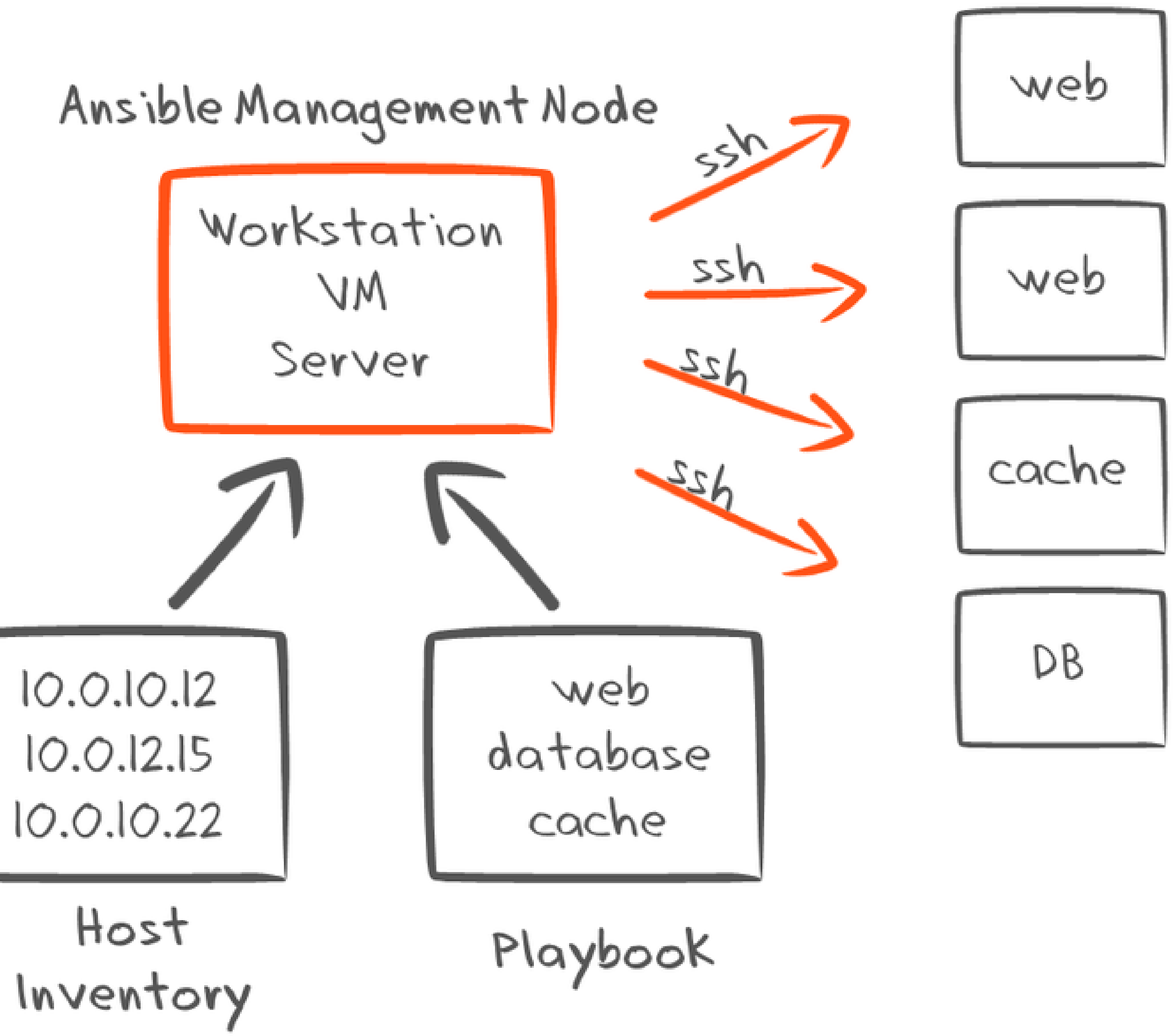
Introduction to Ansible


Ansible Management Node
Workstation
VM
Server
ssh
ssh
ssh
ssh
web
web
cache
DB
10.0.10.12
10.0.12.15
10.0.10.22
Host
Inventory
web
database
cache
Playbook

What is Ansible?

- A *configuration management* tool
- Applies changes to your system to bring it to a desired state
- Similar applications include puppet, chef, salt, juju etc

Why to choose Ansible?

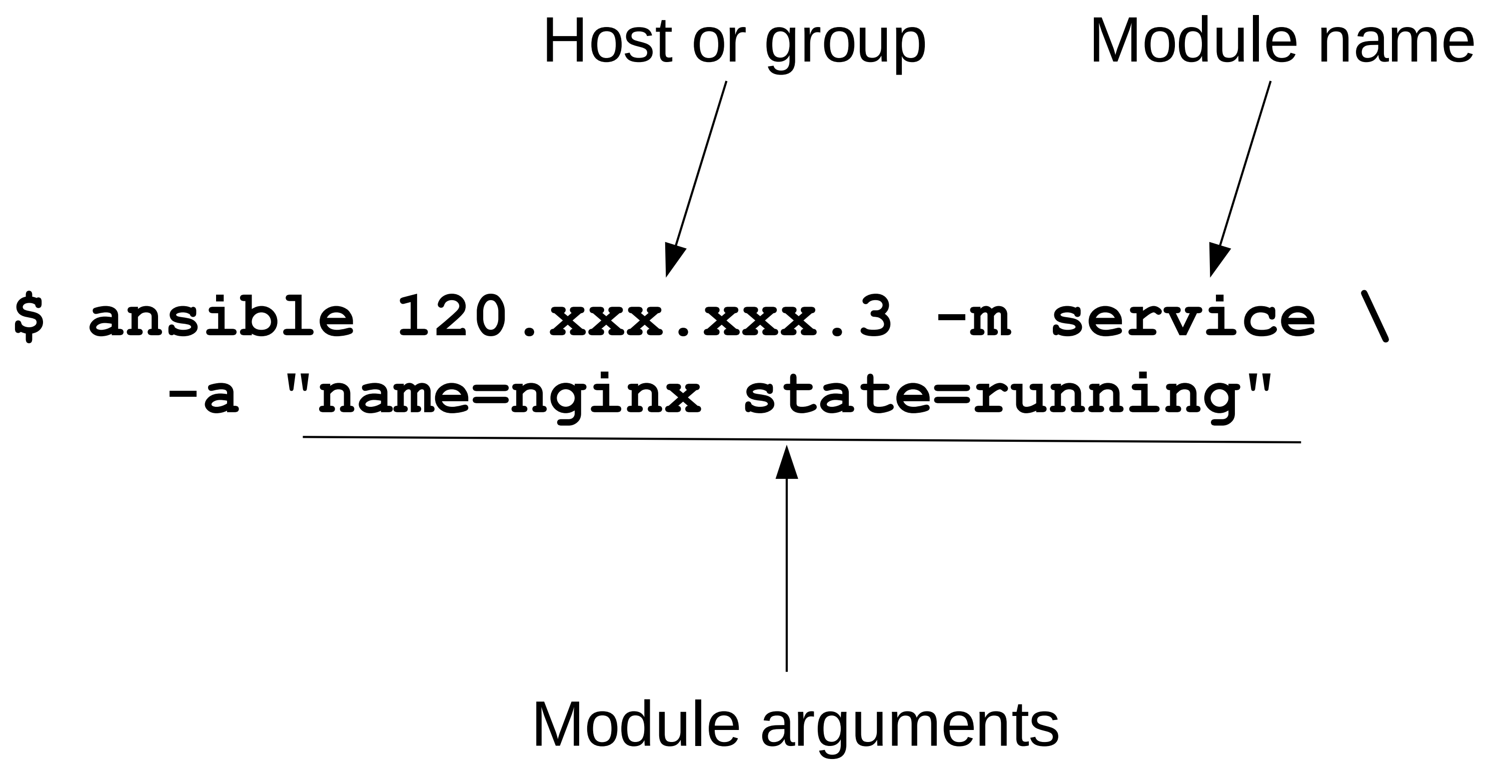
- Target system requires only sshd and python
  - No daemons or agents to install

- Security
  - Relies on ssh
- Easy to get started, compared to the others!

Modules

- Ansible “modules” are small pieces of code which perform one function
  - e.g. copy a file, start or stop a daemon
- Most are “idempotent”: means that they only do something when a change is required
- Many modules supplied as standard

Invoking modules from shell



```
$ ansible 120.xxx.xxx.3 -m service \
    -a "name=nginx state=running"
```

Configuring Ansible Behavior

- ***Tasks*** are modules called with specific arguments
- ***Handlers*** are triggered when something changes
  - e.g. restart daemon when a config file is changed
- ***Roles*** are re-usable bundles of tasks, handlers and templates
- All defined using YAML

YAML

- A way of storing structured data as text
- Conceptually similar to JSON
  - String and numeric values
  - Lists: ordered sequences
  - Hashes: unordered groups of key-value pairs
- String values don't normally need quotes
- Lists and hashes can be nested
- Indentation used to define nesting

YAML list (ordered sequence)

- Single line form

```
[apple, grape, banana]
```

- Multi-line form

```
- apple
- grape
- banana
```

Space after dash required

YAML hash (key-value pairs)

- Single line form

```
{item: shirt, color: red, size: 42}
```

Space after colon required

- Multi-line form

```
item: shirt
color: red
size: 42
description: |
  this is a very long multi-line
text field which is all one value
```

Nesting: list of hashes

- Compact

```
- {item: shirt, colour: red,
  size: 42}
- {item: shirt, colour: blue,
  size: 44}
```

- Multi-line

Note alignment

```
- item: shirt color: red
  size: 42
```

```
- item: shirt  color: blue
  size: 44
```

More complex YAML example

A list with 3 items

Each item is a hash (key-value pairs)

```
- do: laundry
  items:
   - trousers -> list value(note indentation)
   - shirts
- do: polish
  items:
   - shoes
   - buckle
- do: relax
```

```
eat:
- chocolate
- chips
```

Ansible playbook

Top level: a list of "plays"

Each play has "hosts" plus "tasks" and/or "roles"

```
- hosts:
   - pc1.example.com
   - pc3.example.com
  tasks:
   - name: install Apache
     action: yum pkg=apache2 state=present
   - name: ensure Apache is running
     action: service name=apache2 state=running
```

```
- hosts: dns_servers
  roles:
    - dns_server
```

Roles

- A bundle of related tasks/handlers/templates

```
roles/<rolename>/tasks/main.yml
roles/<rolename>/handlers/main.yml
roles/<rolename>/defaults/main.yml
roles/<rolename>/files/...
roles/<rolename>/templates/...

### Recommended way to make re-usable configs
```

Tags

- Each role or individual task can be labelled with one or more "tags"
- When you run a playbook, you can tell it only to run tasks with a particular tag: `-t <tag>`
- Lets you selectively run parts of playbooks

Inventory

- Lists all hosts which Ansible may manage
- Can define groups of hosts
- Default is `/etc/ansible/hosts`
  - *We will instead use `./hosts.local`*
  - *Can override using `-i <filename>`*

Inventory (hosts) example

```
[dns_servers] <--- Name of group
pc1.example.com  <--- hosts in the group
pc2.example.com
[misc]
pc3.example.com
pc4.example.com
# Note: the same host can be listed under
# multiple groups.
# Group "all" is created automatically.
```

Inventory variables

- You can set variables on hosts or groups of hosts
- Variables can make tasks behave differently when applied to different hosts
- Variables can be inserted into templates
- Some variables control how Ansible connects

"Facts"

- Facts are variables containing information collected automatically about the target host
- Things like what OS is installed, what interfaces it has, what disk drives it has
- Can be used to adapt roles automatically to the target system
- Gathered every time Ansible connects to a host (unless playbook has "gather_facts: no")

Showing facts

Invoke the "setup" module

```
$ ansible 10.128.555.16 -m setup | less
10.128.555.16 | success >> {
    "ansible_facts": {
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_version": "12.04",
        "ansible_domain": "a.bdcd.org",
        "ansible_eth0": {
            "ipv4": {
                "address": "10.10.0.241",
                "netmask": "255.255.255.0",
                "network": "10.10.0.0"
```

jinja2 template examples

- Insert a variable into text

```
INTERFACES="{{ dhcp_interface }}"
```

- Looping over lists

```
search a.bdcd.org
{% for host in use_dns_servers %}
nameserver {{ host }}
{% endfor %}
```

Many other cool features

- Conditionals

```
- action: apt pkg=apache2 state=present
when: ansible_os_family=='Debian'
```

- Loops

```
action: yum pkg={{item}} state=present
with_items:
  - openssh-server
  - acpid
  - rsync
  - telnet
```

Hands On

- To learn Ansible basics and create a simple Ansible playbook to install a ngnix server.

- As we know Ansible is a modern IT automation tool which makes your life easier by managing your servers for you.
- You just need to define the configuration in which you are interested and ansible will go ahead and do it for you, be it installing a package or configuring a server application or even restarting a service.
- Ansible is always ready to manage your servers.

How to install Ansible?

- We will install the Ansible by pip. Package managers like dnf, yum and apt can be used.

On CentOS machines

- # yum install epel-release
- # yum install ansible

Check the ansible version

# Ansible –version

Playbooks

- Playbooks are a description of policies that you want to apply to your systems. They consist of a listing of modules and the arguments that will run on your system so that ansible gets to know the current state.
- They are written in YAML. They begin with “- - -”, followed by the group name of the hosts where the playbook would be run.

Examples:

```
---
hosts: localhost
- name: install nginx
  yum: name=nginx state=installed
- name: start nginx
  service: name=nginx state=started
```

```
- name: install git
  yum: name=git state=installed
- name: create the user
  user: name=Aditya
```

- Git module can be used to clone or pull the code from a repository

```
- name: fetch application
  git: repo=https://github.com/1.git dest=/opt/demo
```

Playbook example demo

https://www.digitalocean.com/community/tutorials/how-to-create-ansible-playbooks-to-automate-system-configuration-on-ubuntu