Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Ansible

2017

People matter, results count.

Agenda

1. Ansible Introduction
2. Architecture and Process Flow
3. Creating Environment
4. Ansible Inventory and Configuration
5. Ansible Modules
6. Plays and Playbooks
7. Roles
8. Summary

Ansible Introduction

Ansible Introduction

What is Ansible?

Change Management

Provisioning

Automation

Orchestration

Change Management

- Define a “System State”
  - Enforce the System State
- System State
  - Apache Web Installed
  - Apache Web at version x.xx.x
  - Apache Web Started

CHANGE EVENT

Change Management

- Did the process failed?
- Did someone not verify?
- Did someone goof up?

Now that's what I like to see?

Change Management

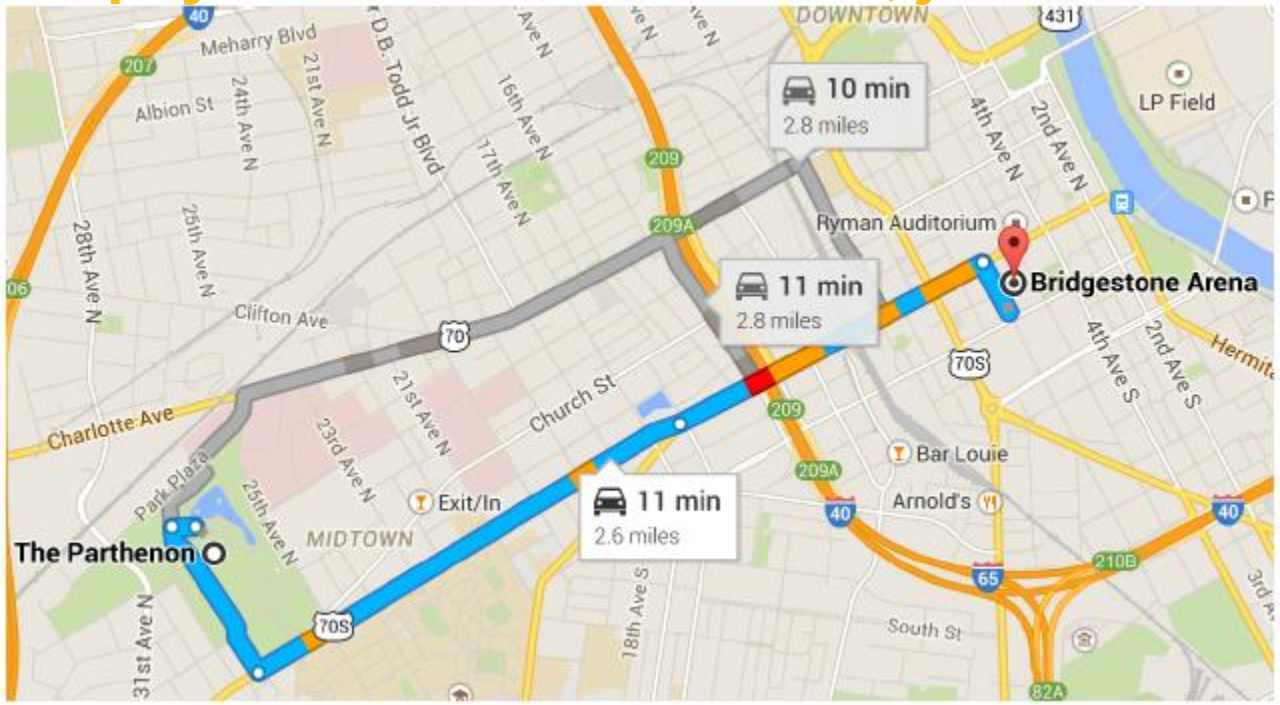
- A function is idempotent if repeated
- applications has the same affect as a
- single application

- IDEMPOTENCE

Change Management

Defining State

Don’t pay attention to the ‘How’, just the ‘What’



Provisioning

- Prepare a system to make it ready
  - Transition from one state to a different state
- Examples
  - Make an FTP Server
  - Make an Email Server
  - Make a DB Server

Provisioning

1. Install web software
2. Copy configurations
3. Copy web files
4. Install security updates
5. Start web service

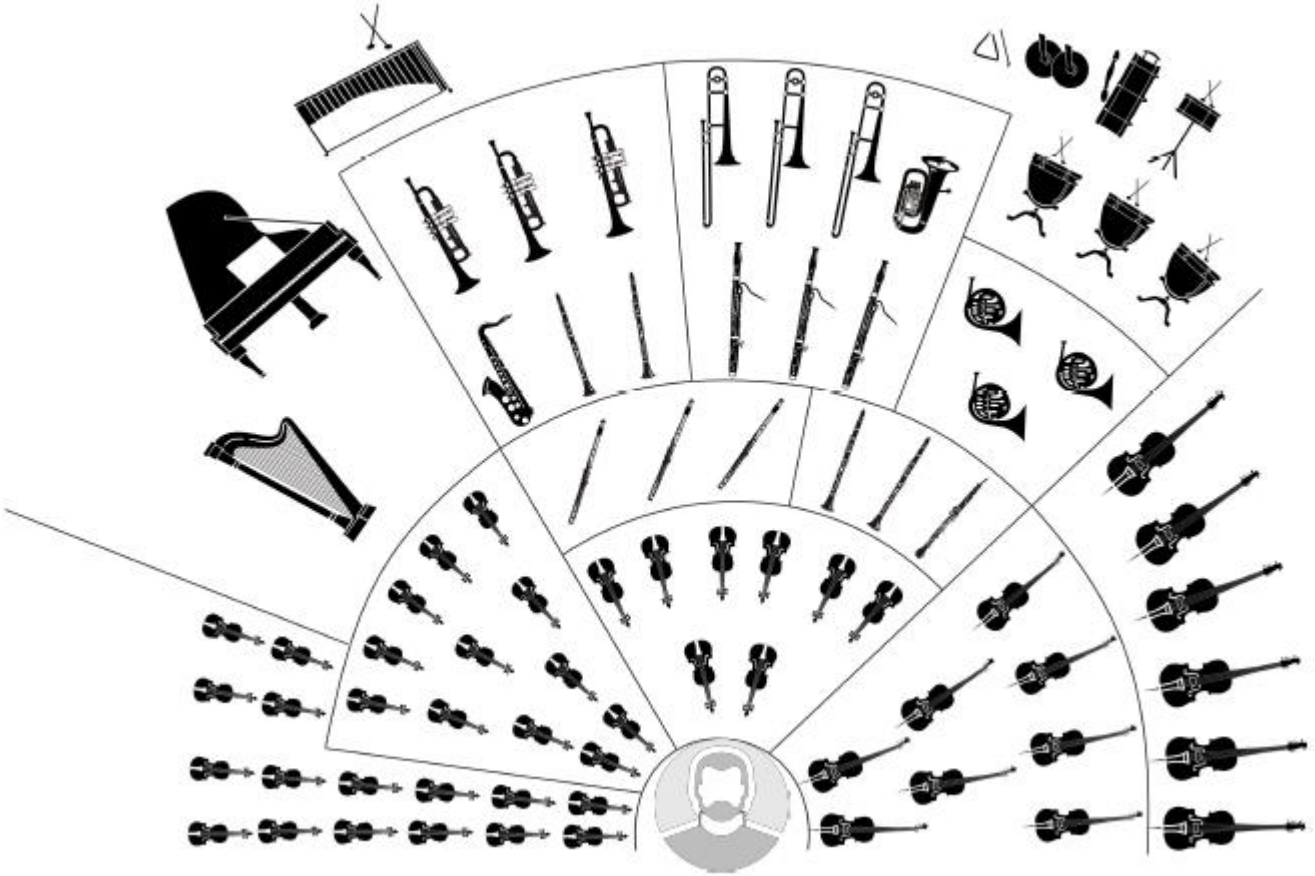
Automation

- Define tasks to be executed automatically
  - Ordered Tasks
  - Make decisions
  - Ad-hoc tasks
- Set it and Forget it
  - Run the task
  - Get a cup of coffee
  - Walk back to desk seeing tasks finished
  - Sip your coffee and feel productive

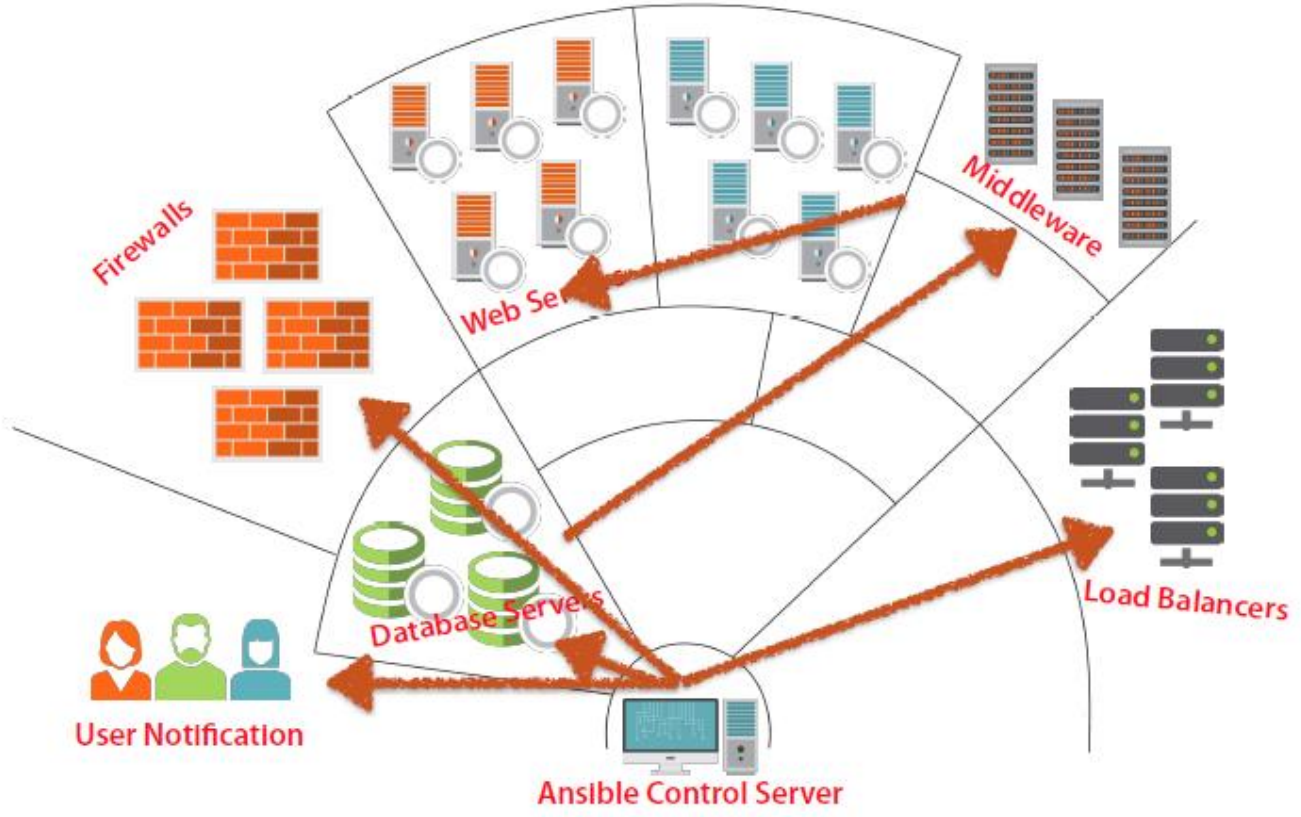
Orchestration

- Coordinates automation BETWEEN systems
  - Task 1 - System 1
  - Task 2 - System 2
  - Task 3 - System 3
  - Task 4 - System 1

Ansible



Ansible



Why Ansible?

- What makes it so different?
  - It’s clean!
  - No agents
  - No database
  - No residual software
  - No complex upgrades

YAML

- Ansible Execution
  - No programming required
  - NOT a markup language
  - Structured
  - Easy to read and write

Security

- Built-in security
  - Uses SSH
  - Root / Sudo usage
  - Encrypted vault
  - No PKI needed

Easy to Extend

- Easy to extend
  - URL / RESTful calls
  - Shell Commands
  - Scripts
  - Ansible-Galaxy

Summary

- Ansible Introduction
- Change Management
- Provisioning
- Automation
- Orchestration
- Easy to implement
- Easy to Program
- Inherently Secure
- Very extendable

Architecture and Process Flow

Architecture

Ansible Requirements

Ansible Requirements
(Control Server)

Python 2.6+

Must be *NIX (Linux/Unix/Mac)

Windows not supported

Ansible Requirements

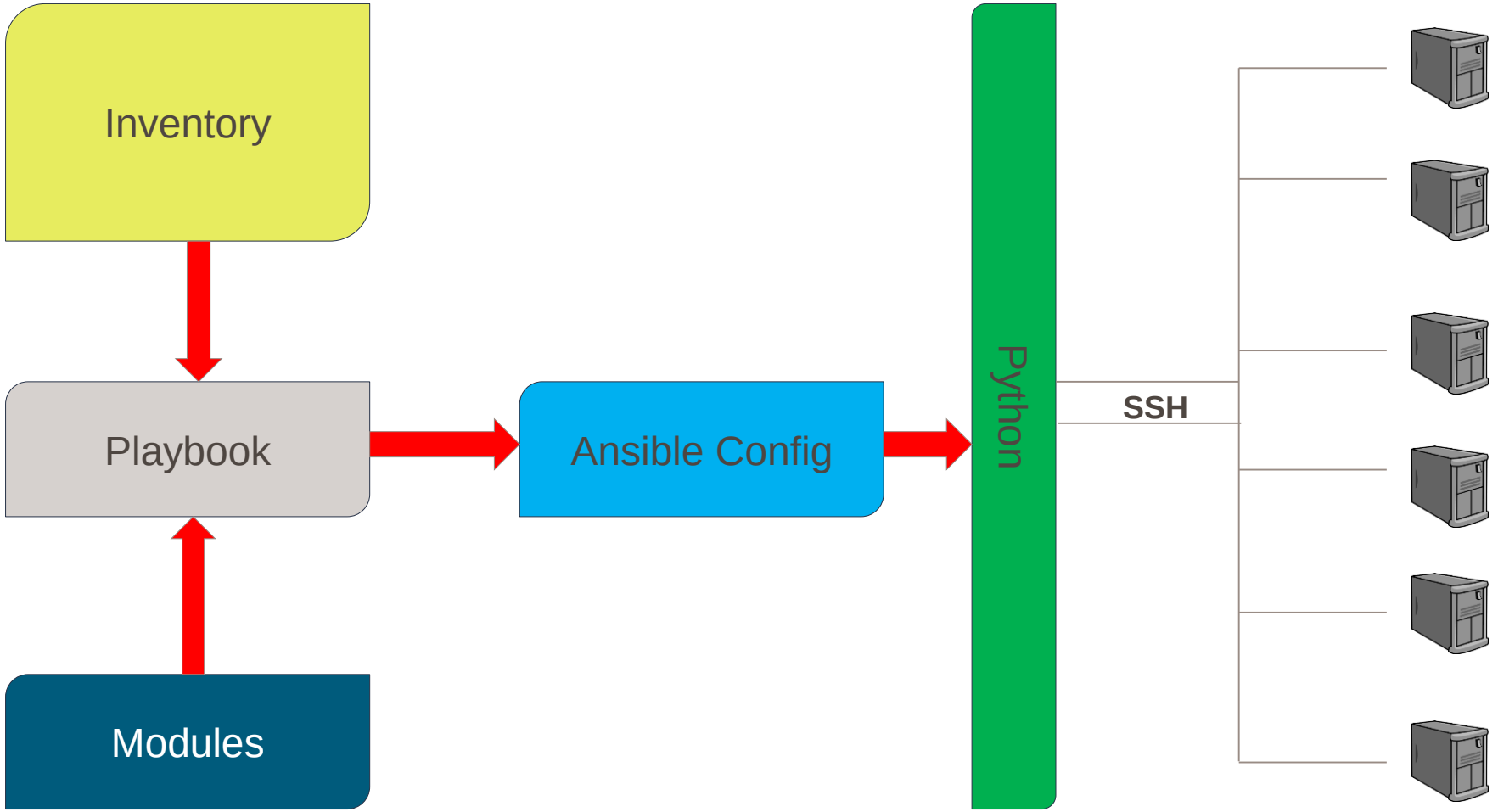
Ansible Requirements
(Remote Server)

***NIX:**
Python 2.4 (simplejson)
Python 2.5+
SSH

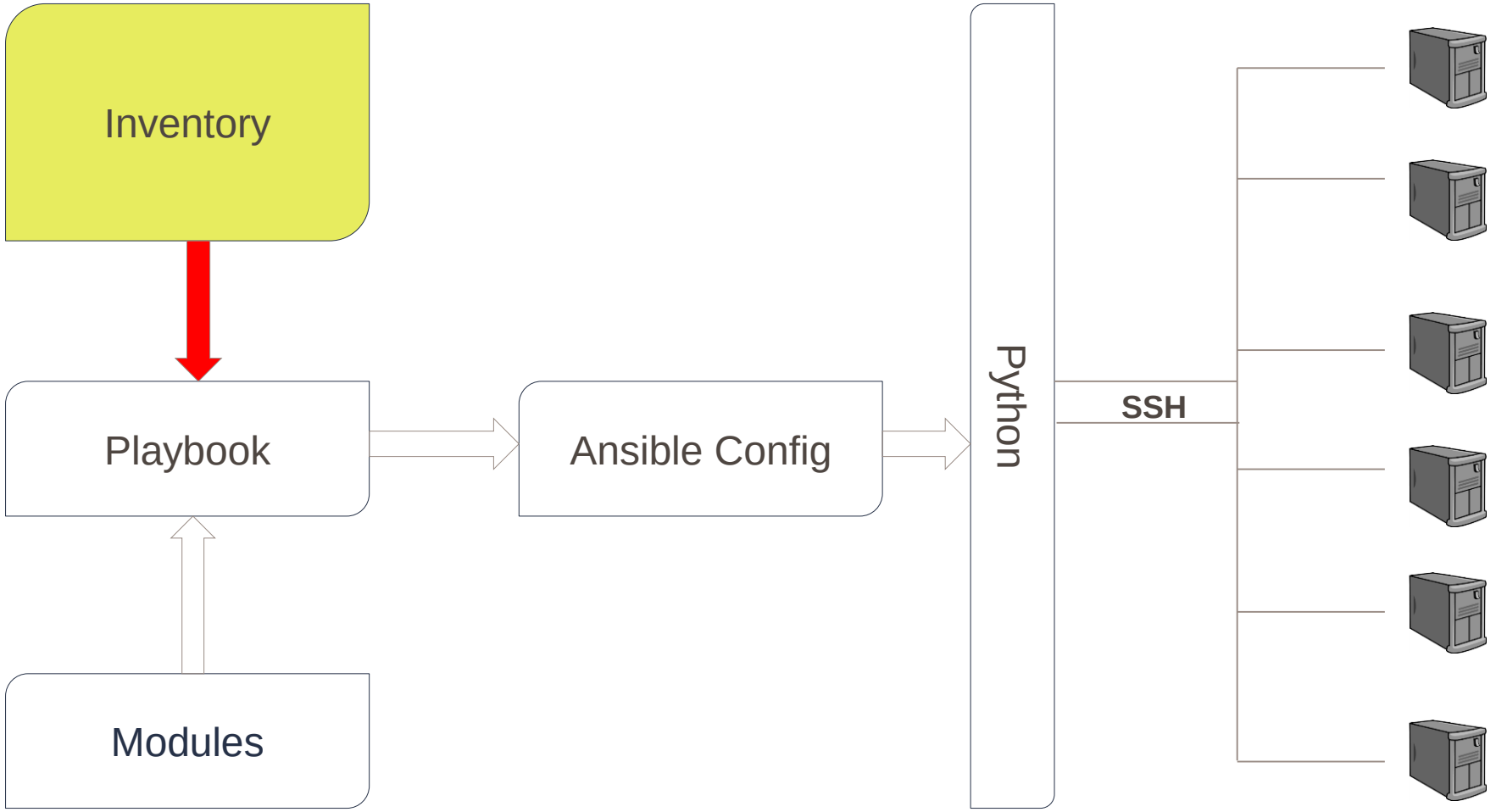
Windows:
Remote Powershell Enabled

***Note** : Python3.x is not an upgrade to Python 2.x; Python 3.x is not supported*

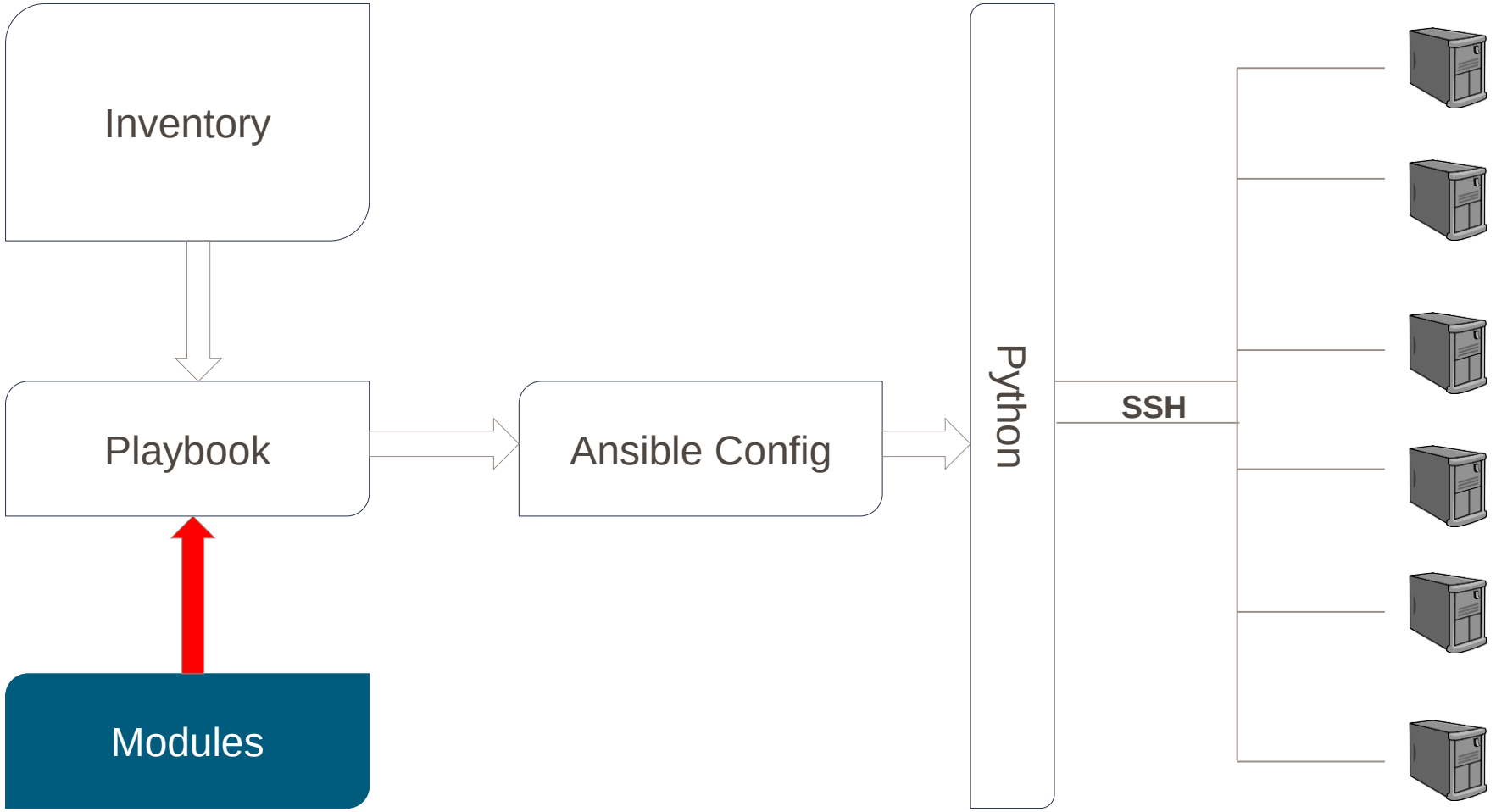
Architecture



Inventory



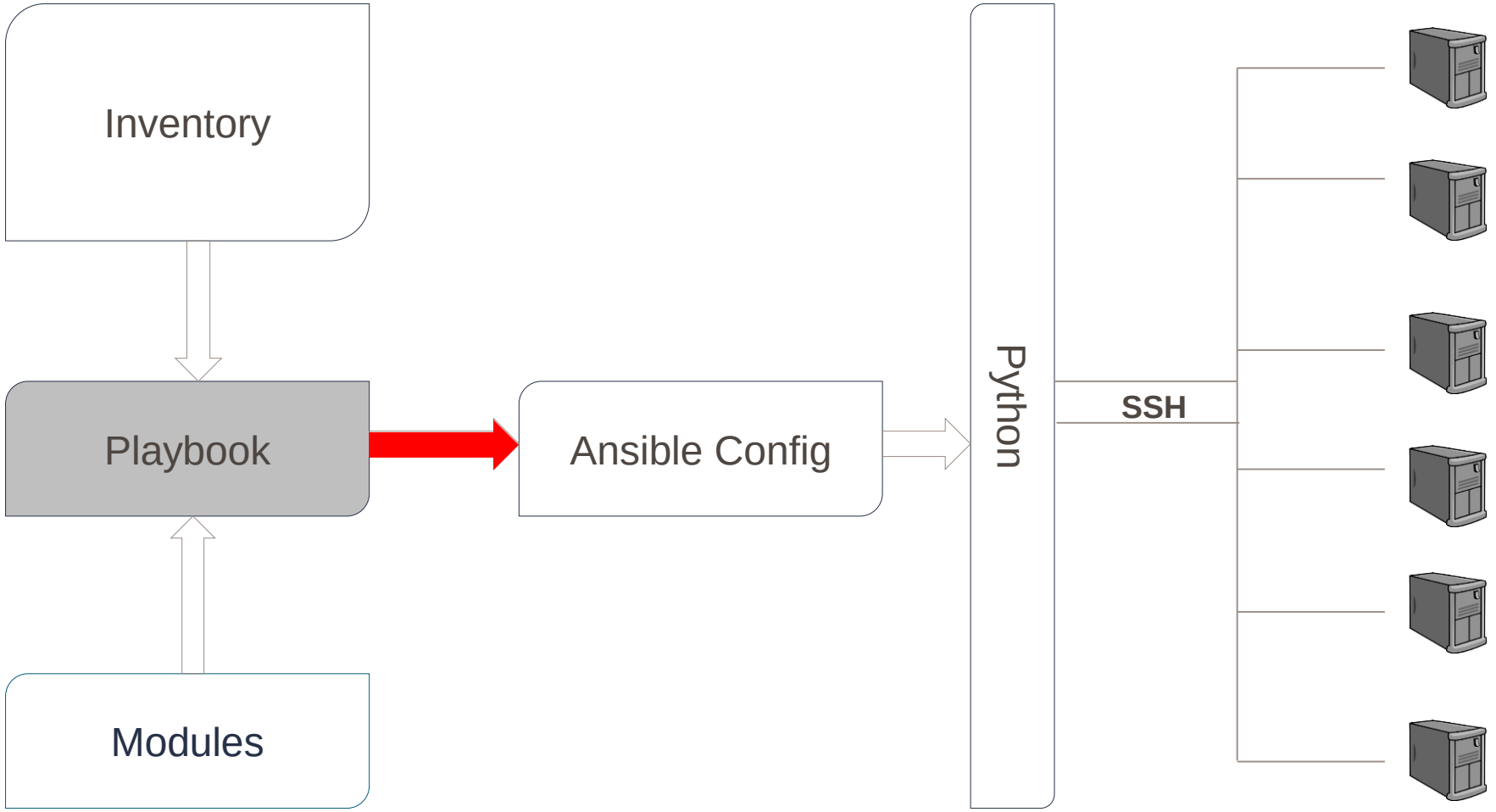
Modules



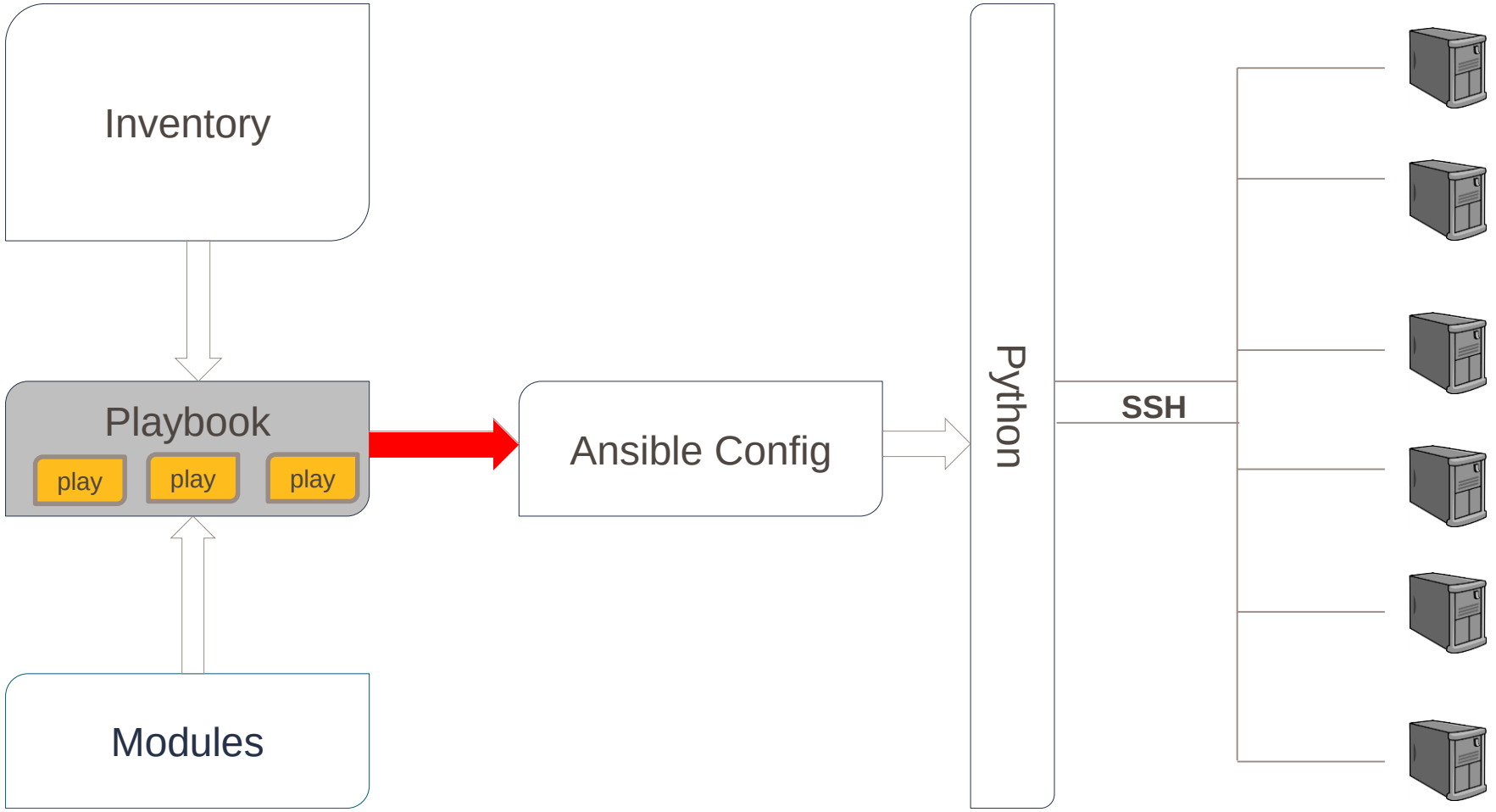
Modules

A programmed unit of work to be done.

Playbook



Playbook

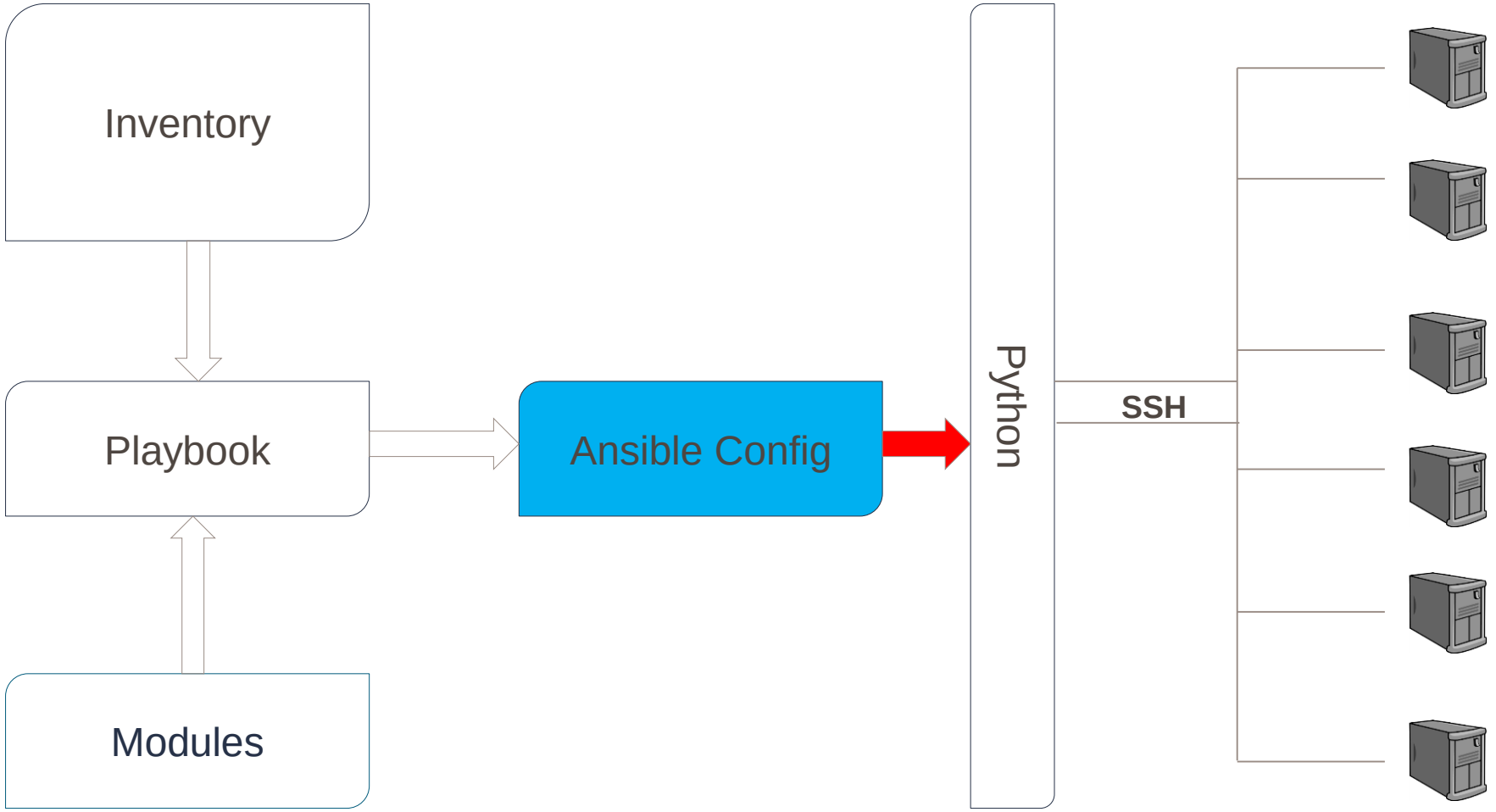


Playbook

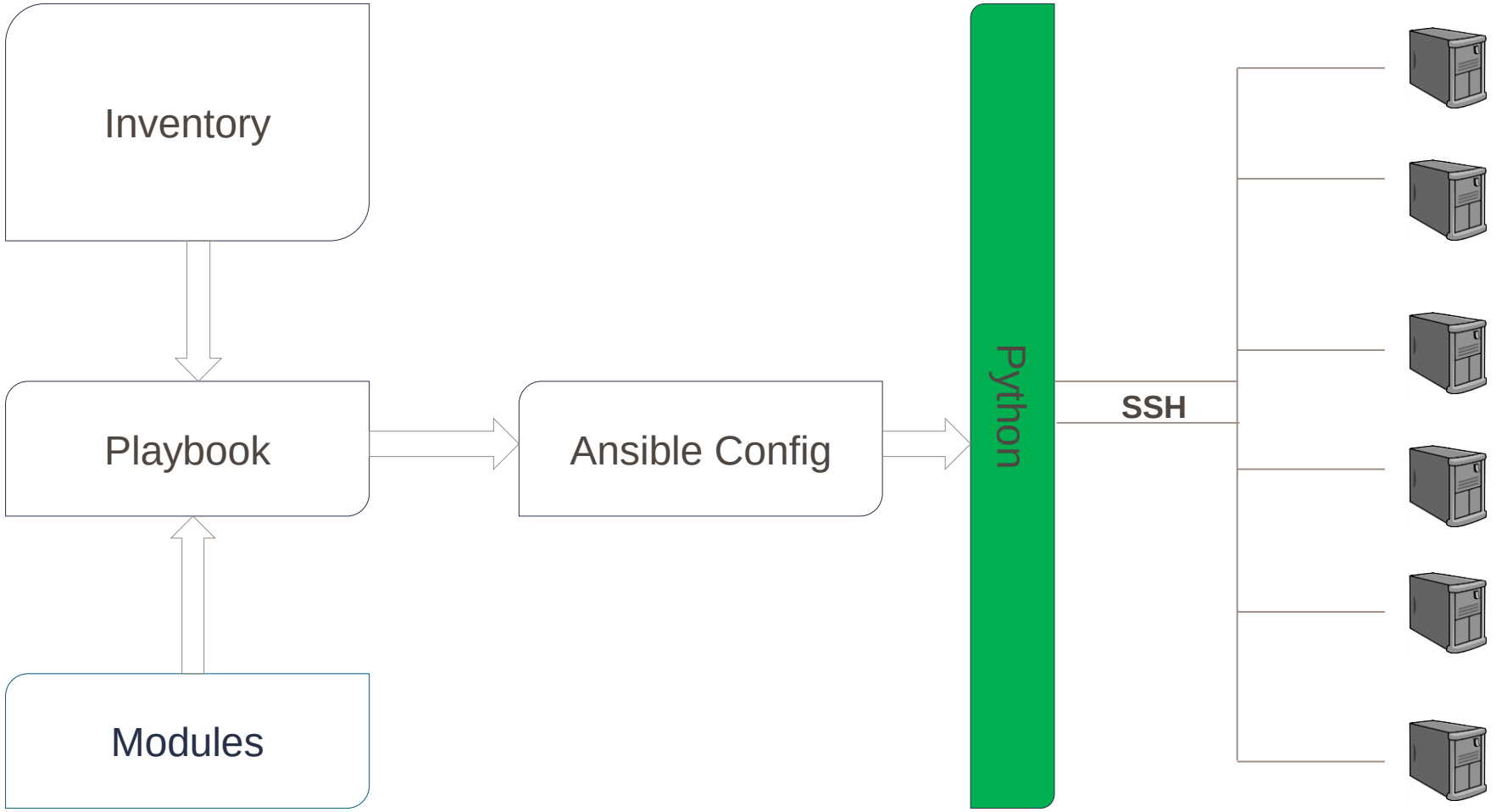
Play:
A single task from a module, executed on a host or set of hosts.

Playbook:
A set of plays built in specific order sequence to produce an expected outcome or outcomes.

Ansible Config



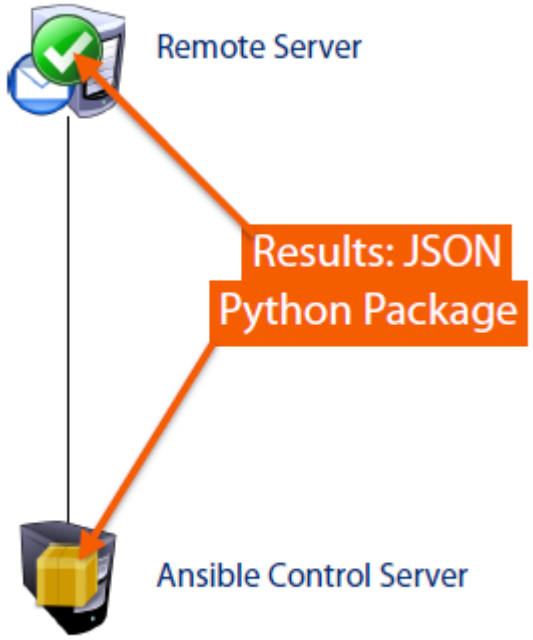
Python



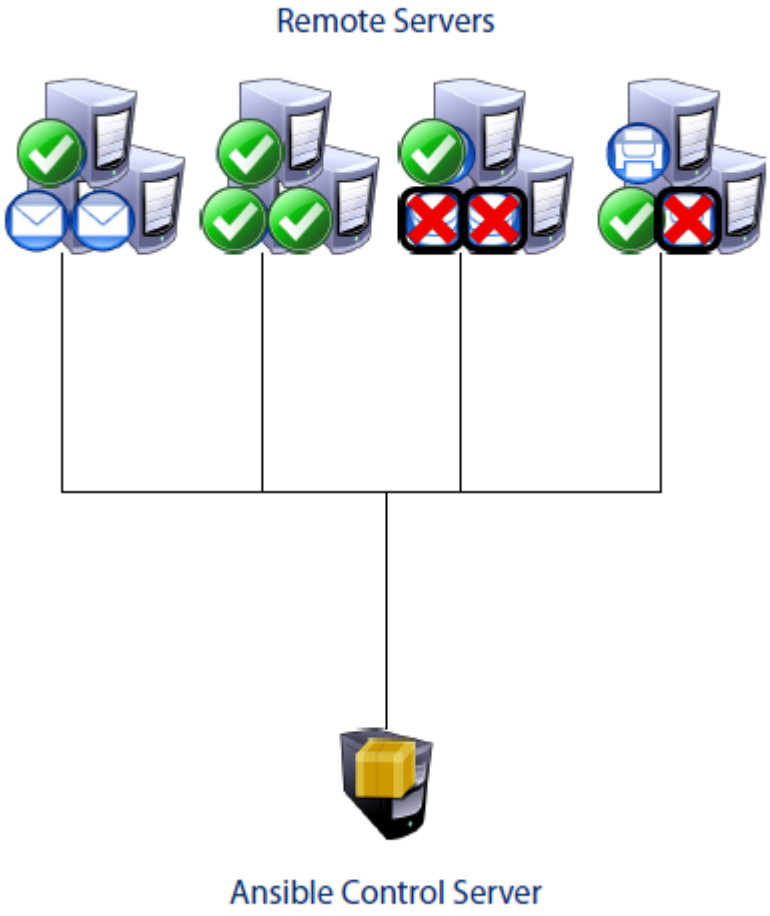
Variables

- Host Variables
  - Use variables defined in Inventory per host or group
- Facts
  - Use data gathered from the remote managed host
- Dynamic Variables
  - Use data gathered by tasks or created at runtime

Ansible

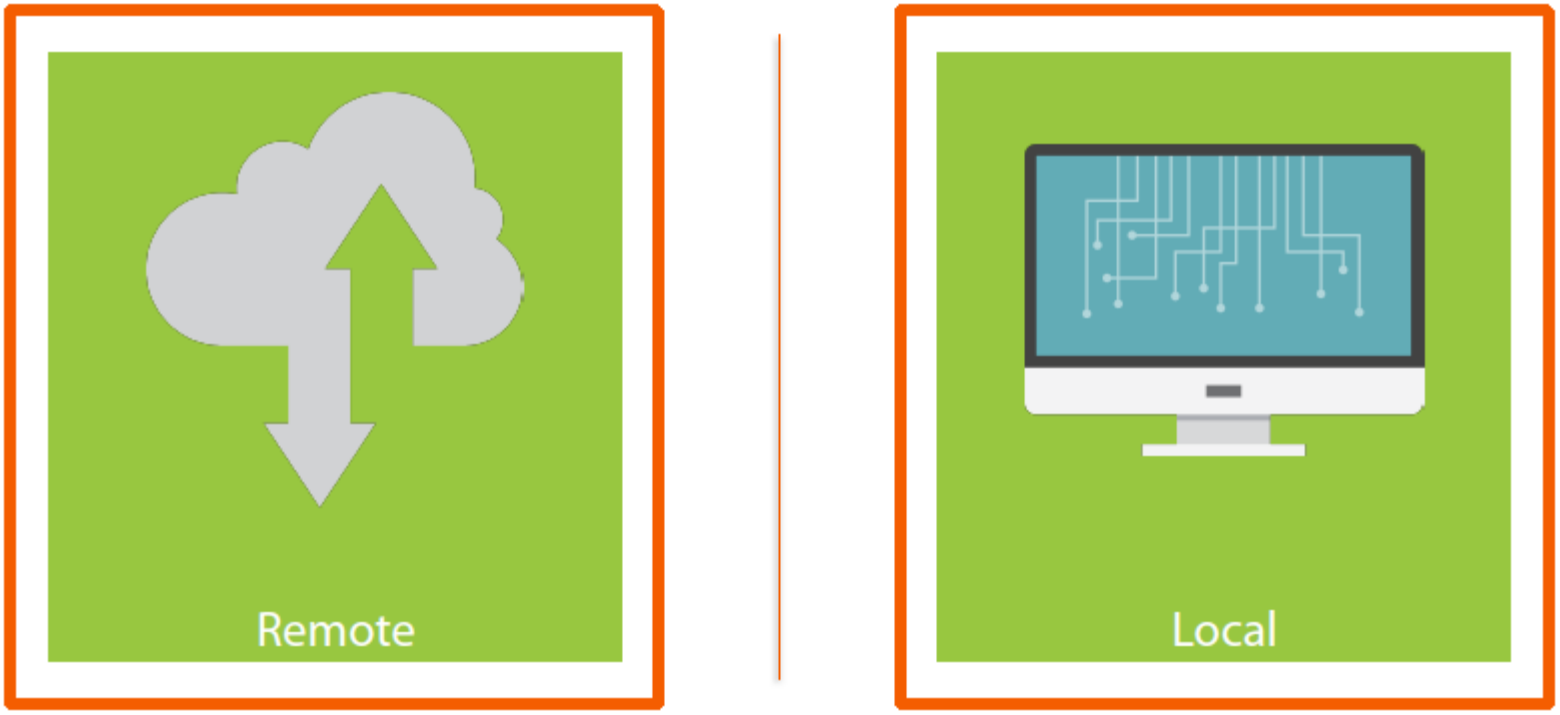


Ansible

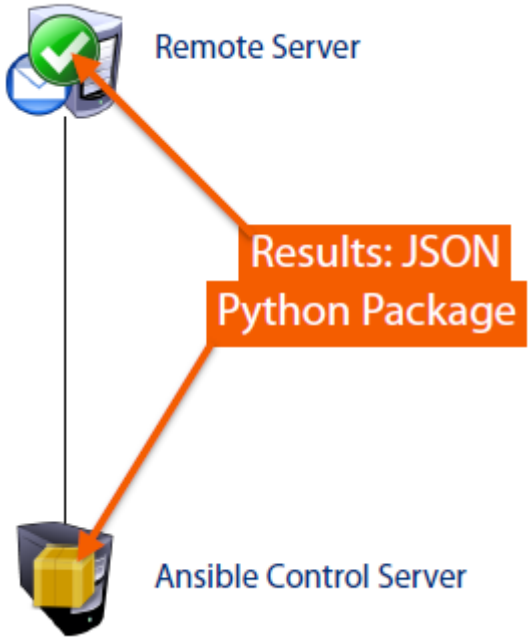


Ansible

Execution Types

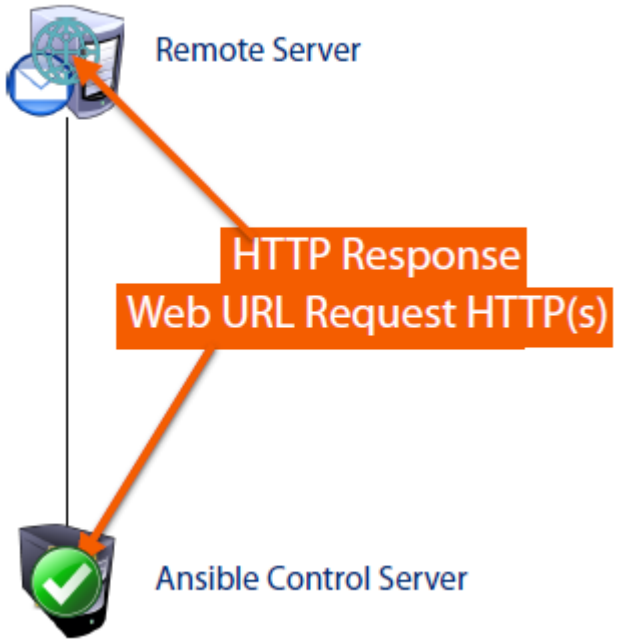


Ansible



Packaged Tasks Executed on Remote-End

Ansible



Packaged Tasks Executed on Ansible Server
Mostly used for webservice/API calls

Summary

Ansible Architecture

- Inventory Maps hosts
- Configuration sets Ansible Parameters
- Modules define action
- Playbooks to coordinate multiple tasks
- Python on build the execution
- SSH to deliver the task

Execution Types

- Remote
  - Execution of playbooks
- Local
  - When remote box is not executing plays

Creating Environment

Creating Environment

Creating Environment

Vagrant: Virtual Machine Controller

Define VM's to startup, and initial configs (ip, hostname, etc)

Creating Environment

VirtualBox: Virtual Machine Provider

Environment to run virtual machines

Creating Environment

Ansible: Automation / Provisioning Application to push configuration and automation to remote systems.

Installation

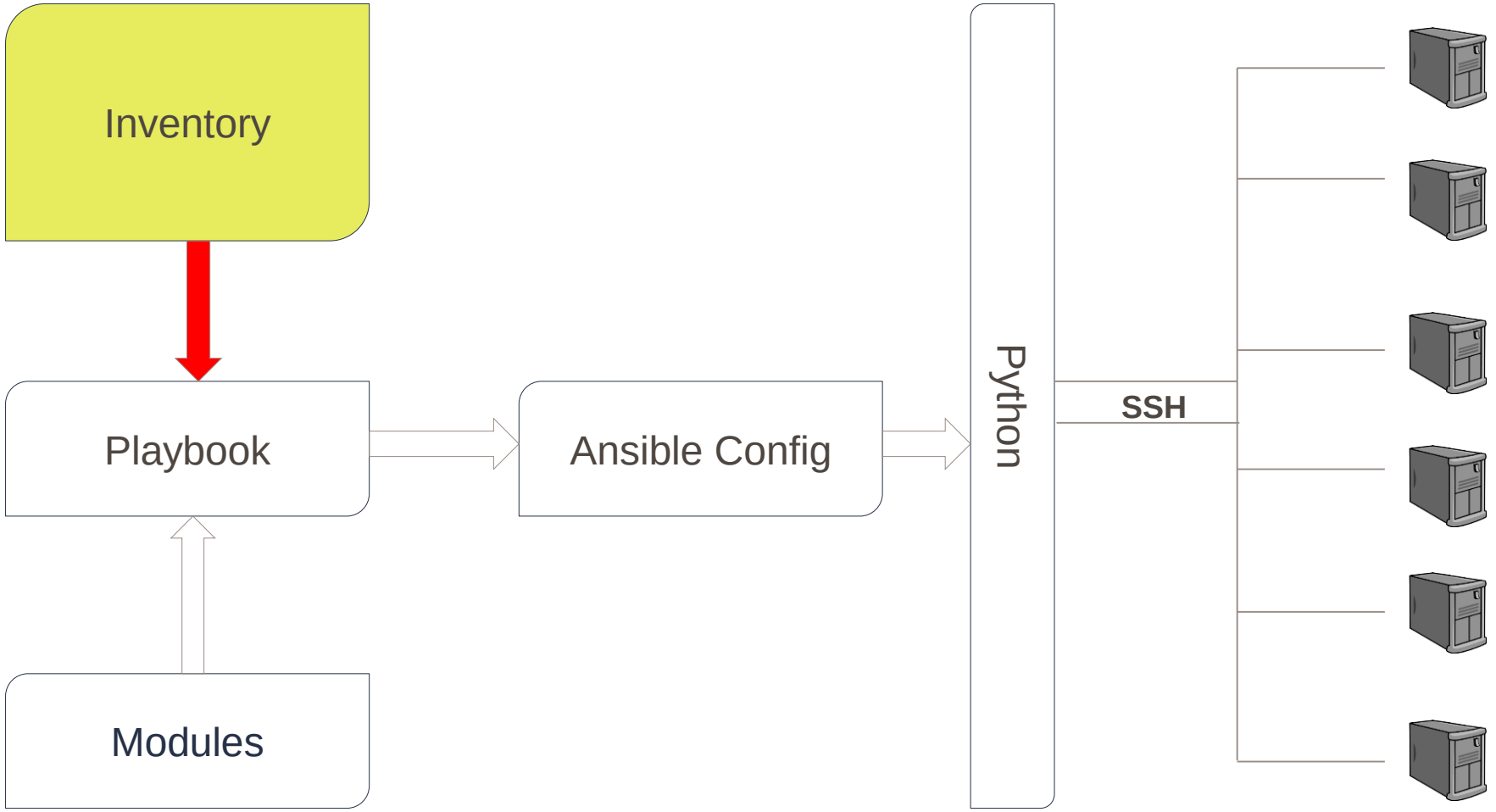
Install Ansible (Debian)	$ sudo apt-get install ansible
Install Ansible (CentOS)	$ sudo yum install epel-release $ sudo yum install ansible
PIP Install (All others)	Install Libraries (gcc, python-devel) Install Python SetupTools Install Ansible

Summary

- Environment Review
- Vagrant : Environment Controller
- Virtual Box : Server Hypervisor
- Ansible : Automation

Ansible Inventory and Configuration

Inventory



Inventory Features

1. Behavioral Parameters
2. Groups
3. Groups of Groups
4. Assign variables
5. Scaling out using multiple files
6. Static / Dynamic

Inventory File

[db]
db1.company.com ansible_ssh_user=aaron ansible_ssh_pass=123
db2.company.com ansible_python_interpreter=/usr/bin/python

Inventory File

```
[db]
db1.company.com ansible_ssh_user=aaron ansible_ssh_pass=123
db2.company.com ansible_python_interpreter=/usr/bin/python

[datacenter-west:children]
db
```

Inventory File

[db]
db1.company.com ansible_ssh_user=aaron ansible_ssh_pass=123
db2.company.com ansible_python_interpreter=/usr/bin/python

[datacenter-west]
db

Inventory File

```
[db]
db1.company.com ansible_ssh_user=aaron ansible_ssh_pass=123
db2.company.com ansible_python_interpreter=/usr/bin/python

[datacenter-west:children]
db
```

Inventory File

[db]
db1.company.com ansible_ssh_user=aaron ansible_ssh_pass=123
db2.company.com ansible_python_interpreter=/usr/bin/python

[datacenter-west:children]
db

[datacenter-west:vars]
ansible_ssh_user=ansible_user
ansible_ssh_pass=#45e!@Gh
ntp-server=5.6.7.8

Creating Inventory File

- Add Behavioral Parameters
- Create host-based variables
- Create a Group
- Create group-based variables

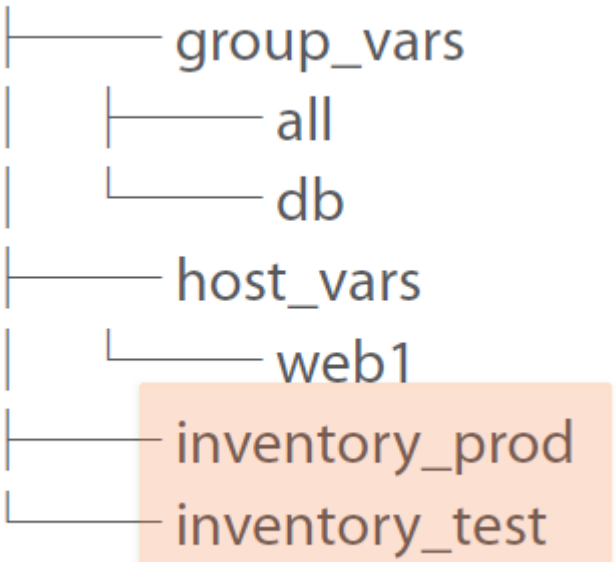
Scaling-out Inventory File

Using Directories

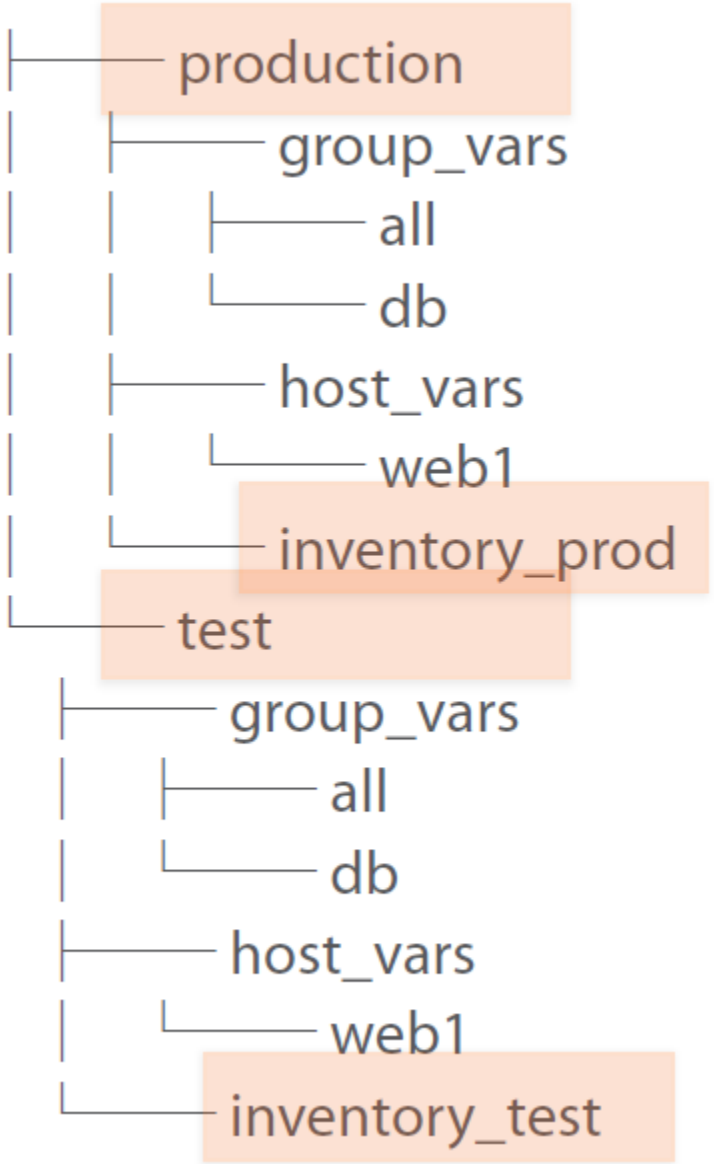
Can use to break-out long-running inventory files.

Very useful when dealing

Directory Structure



Basic Directory Structure



Multi-Environment Directory Structure

Order-of-Operations (Precedence)

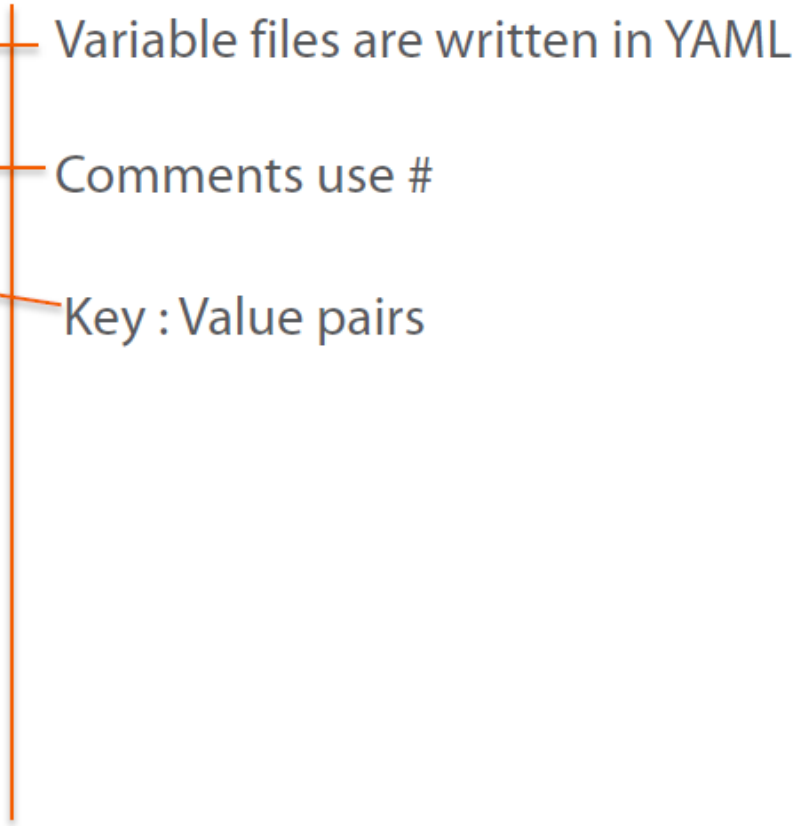
1). (Group_Vars) All

2). (Group_Vars) GroupName

3). (Host_Vars) HostName

Variable file Example

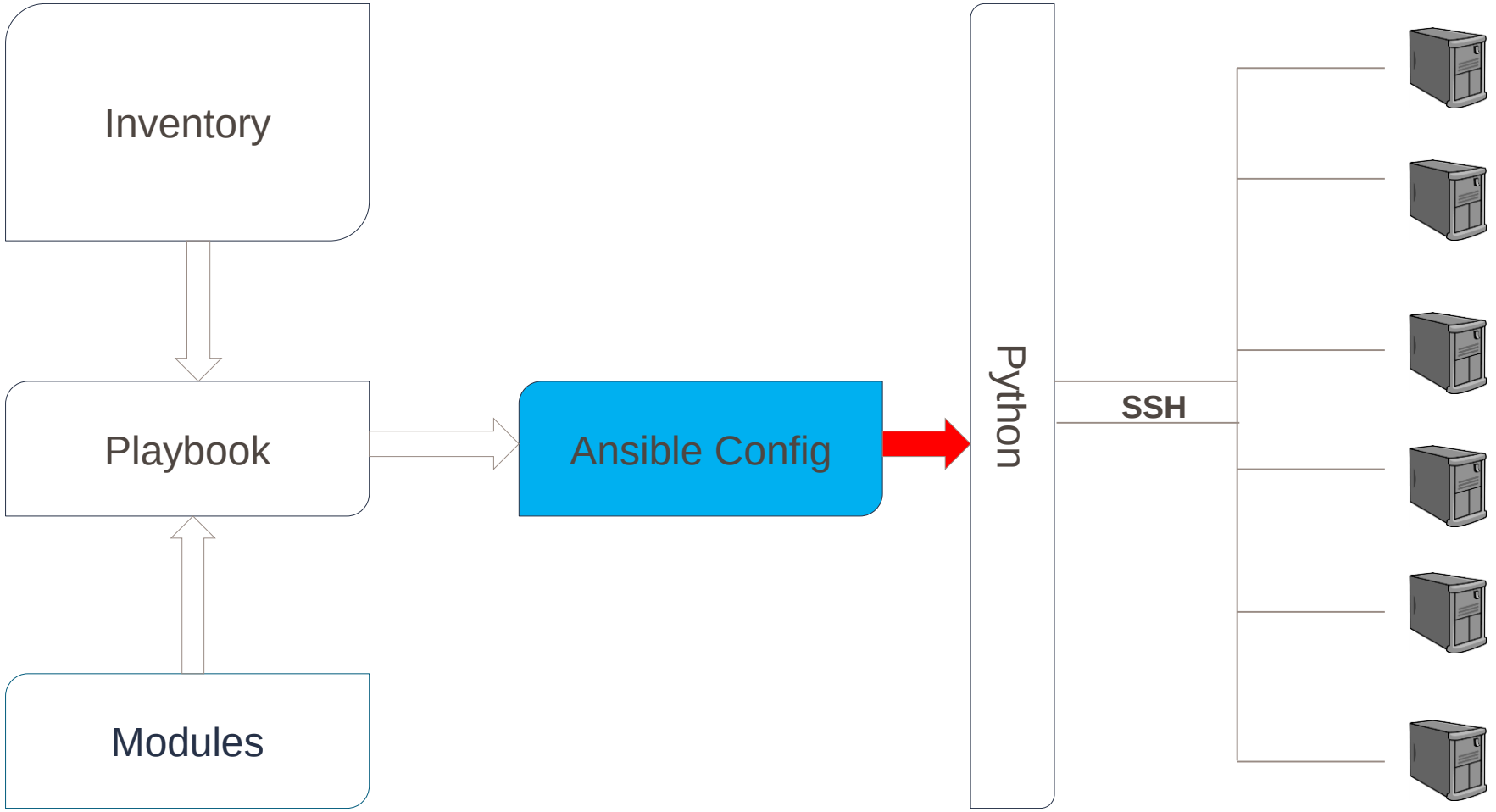
```
---
# file: group_vars/dc1-west
ntp: ntp-west.company.com
syslog: logger-west.company.com
```



Scaling Variable Files

- Create Group Variables in separate file
- Show Order-of-Precedence

Ansible Config



Ansible Config

Configuration Settings Order-of-Operations

1. $ANSIBLE_CONFIG
2. ./ansible.cfg
3. ~/.ansible.cfg
4. /etc/ansible/ansible.cfg

Note: Configuration files are not merged. First one wins!

Ansible Config

Environment variable overrides

$ANSIBLE_<configsetting>

Override specific settings by prefixing ANSIBLE_ to the name

 # export ANSIBLE_FORKS=10

Great way to override specific settings on-the fly!

Note: Configuration files are not merged. First one wins!

Ansible Config

[defaults] forks

Default set to 5

Total number of parallel operations Ansible executes

Production Recommendation: 20

Start at 20, and go up or down depending on performance

Ansible Config

[defaults]
host_key_checking

Default set to True

For Production environments, do not change

Development Environment: set to False

Due to the dynamic environment of Dev, keeps it easy

Ansible Config

[defaults]
log_path

Default set to Null

Write information on Ansible executions

Set path to log file

Make sure all users of Ansible has permissions

to write

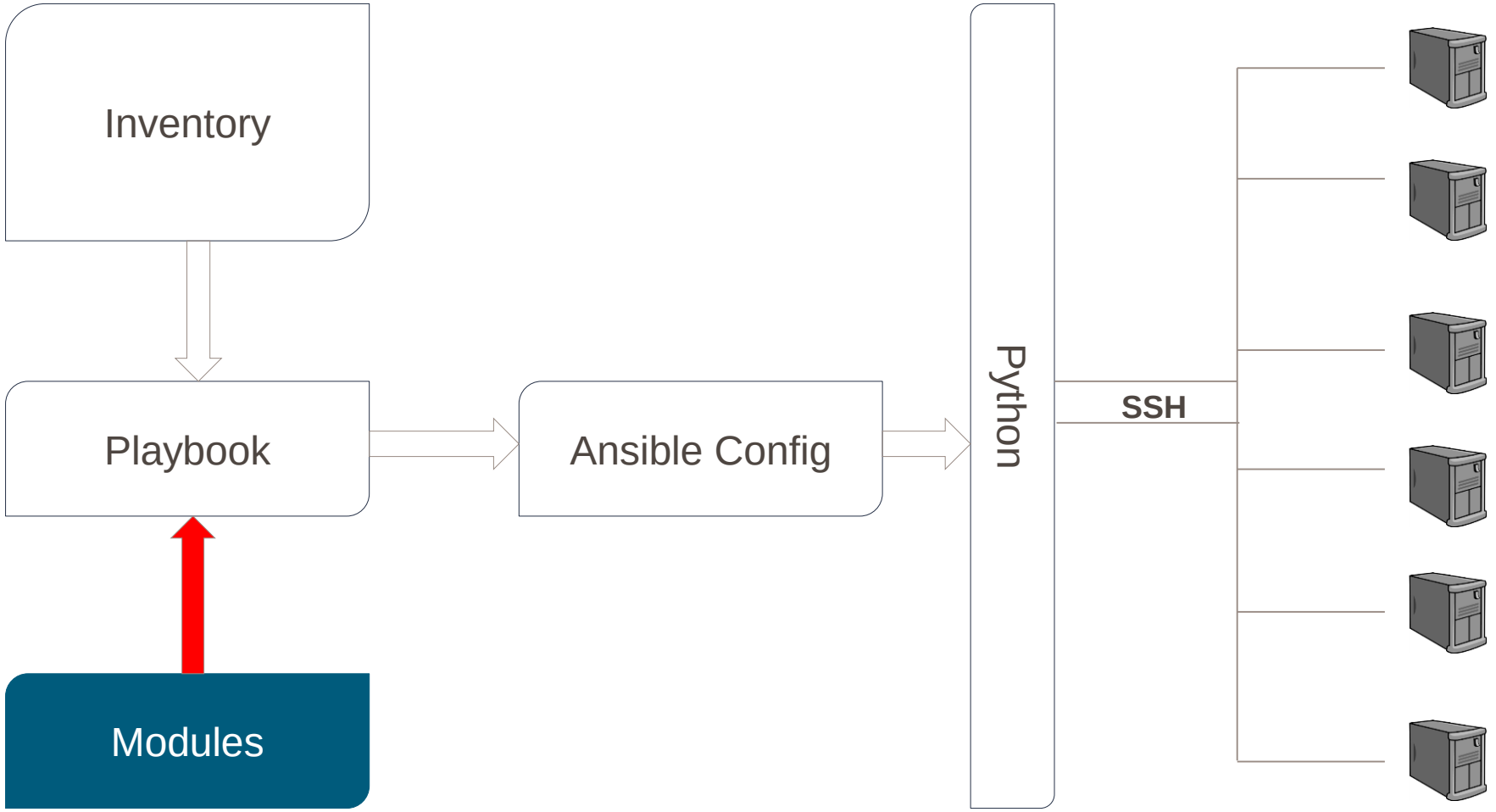
Ansible Config

Editing Configuration

- Define settings in configuration file
- Override setting in environment variable

Ansible Modules

Modules



Modules

3 Types of Modules

Core

Extras

Deprecated

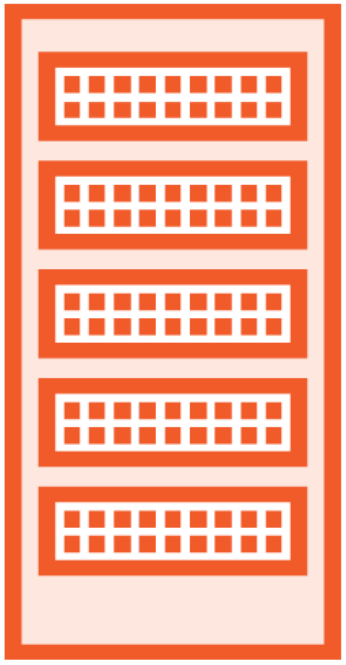
Modules

Module Doc

$ ansible-doc -l	$ ansible-doc -s <name>	$ ansible-doc <name>

Modules

Module Categories



- Manage Servers
- Deploy Configurations

Module Categories

- Configure Network Equipment

Modules

Module Categories

- Maintain Virtual Servers

Module Categories

- Manage Databases and tables

Module Categories

- Deploy load-balancer configurations

Modules

Copy Module

- Copies a file from local box to remote system
- Has “backup” capability
- Can do validation remotely

Modules

Fetch Module

- Pulls a file from remote host to local system
- Can use md5 checksums to validate

Modules

Apt Module

- Manages installed applications on Debian-based systems
- Can install, update, or delete packages
- Can update entire system

Modules

Yum Module

- Manages installed applications on Redhat-based systems
- Can install, update, or delete packages
- Can update entire system

Modules

Service Module

- Can stop, start, or restart services
- Can enable services to start on boot

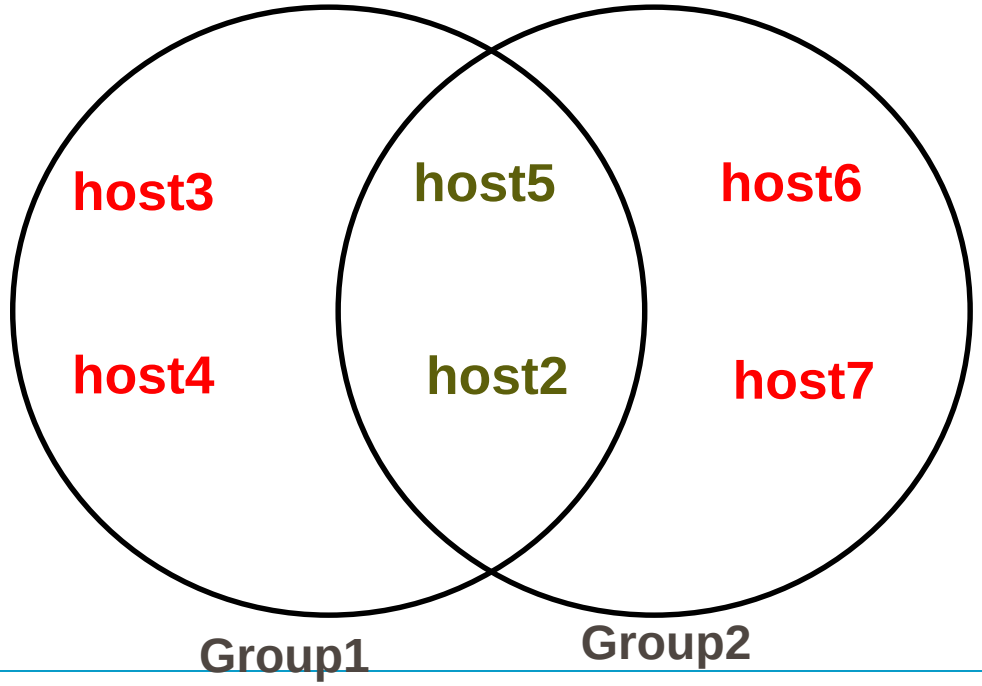
Modules

Host/Group Target Patterns

OR (group1:group2)	NOT (!group2)
Wildcard (web*ex.com)	Regex (~web[0-9]+)

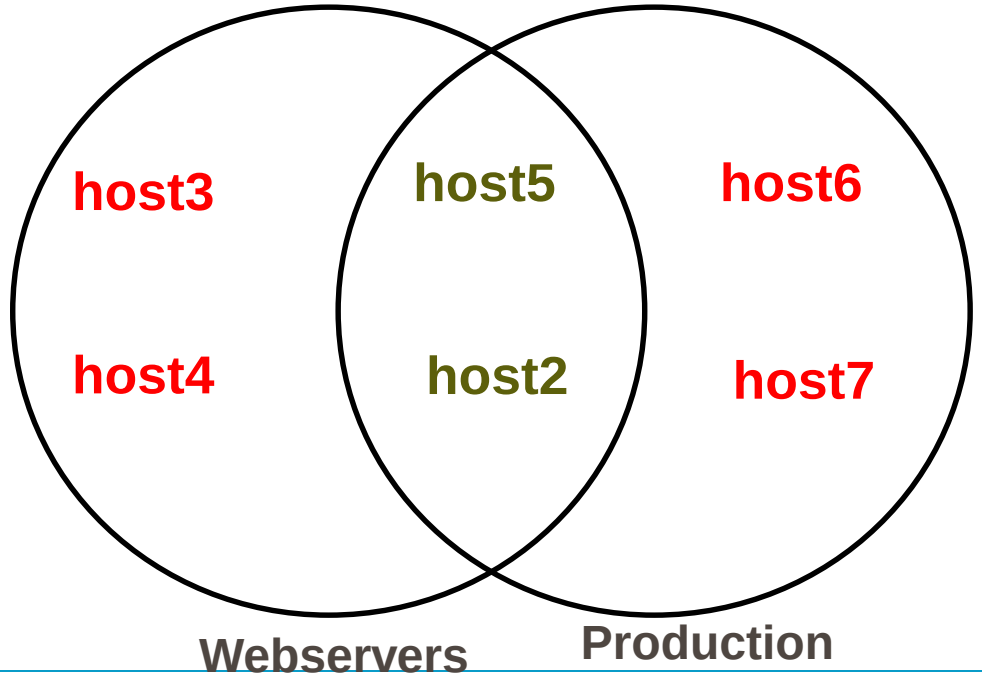
Complex Patterns

AND
(group1 : &group2)



Complex Patterns

AND
(Webservers : &Production)



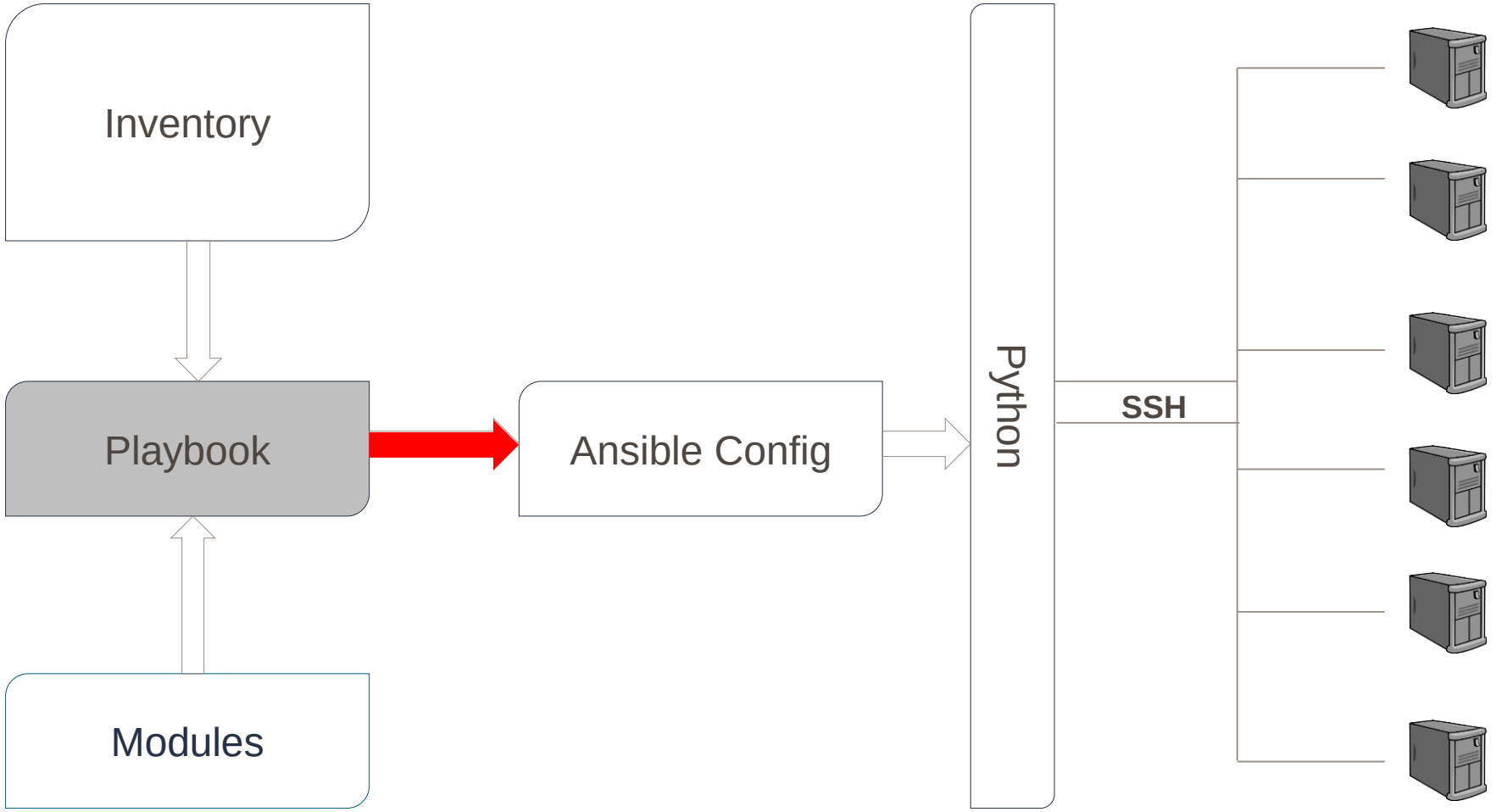
Complex Patterns

Webservers : &prod:!python3

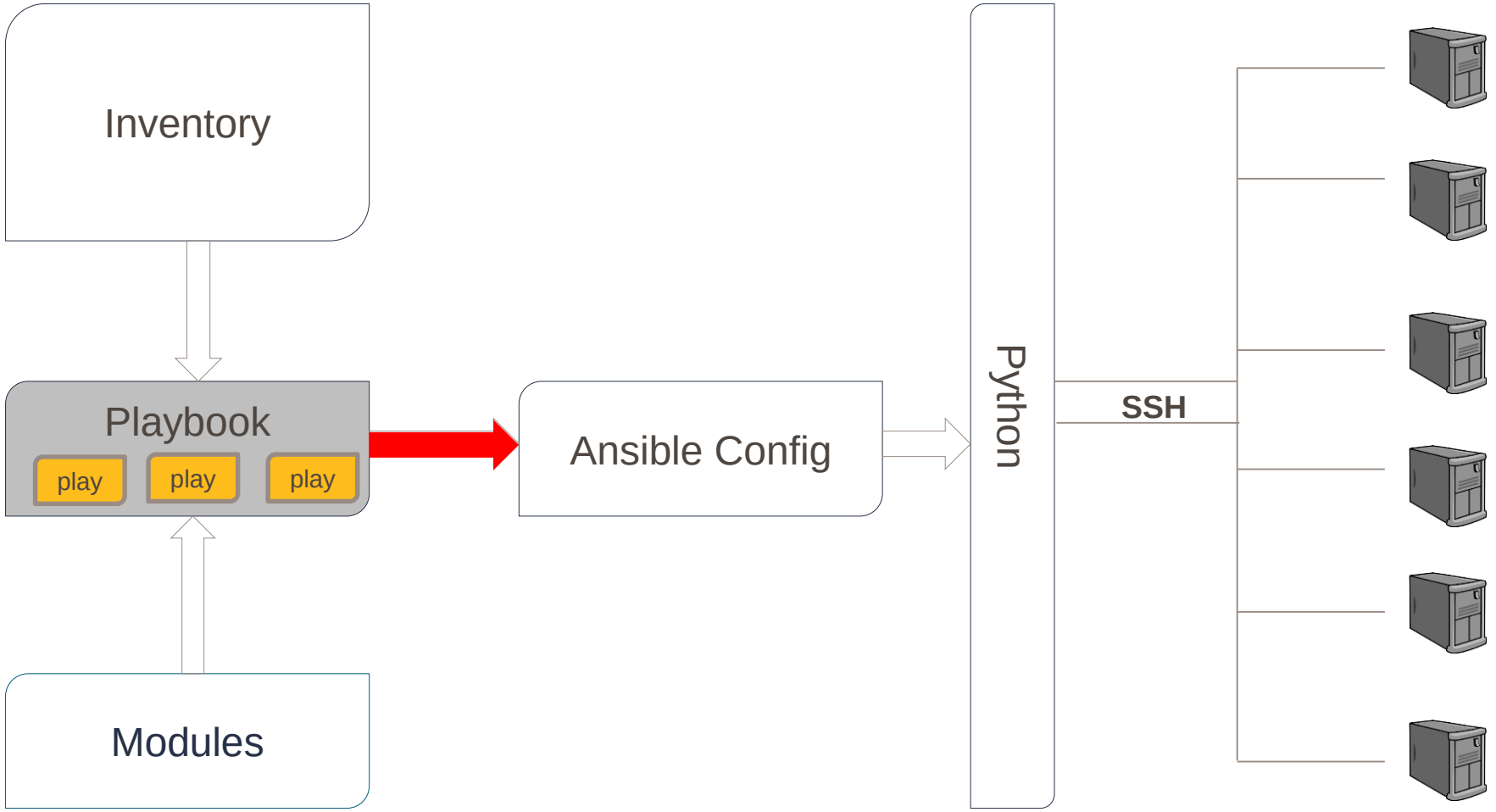
Hosts defined in Webservers AND Prod but NOT in Python3

Plays and Playbooks

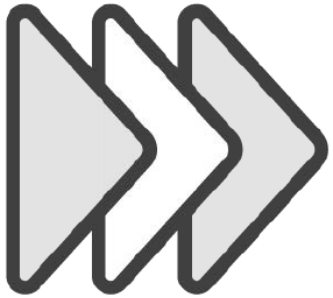
Playbook



Playbook



Playbook

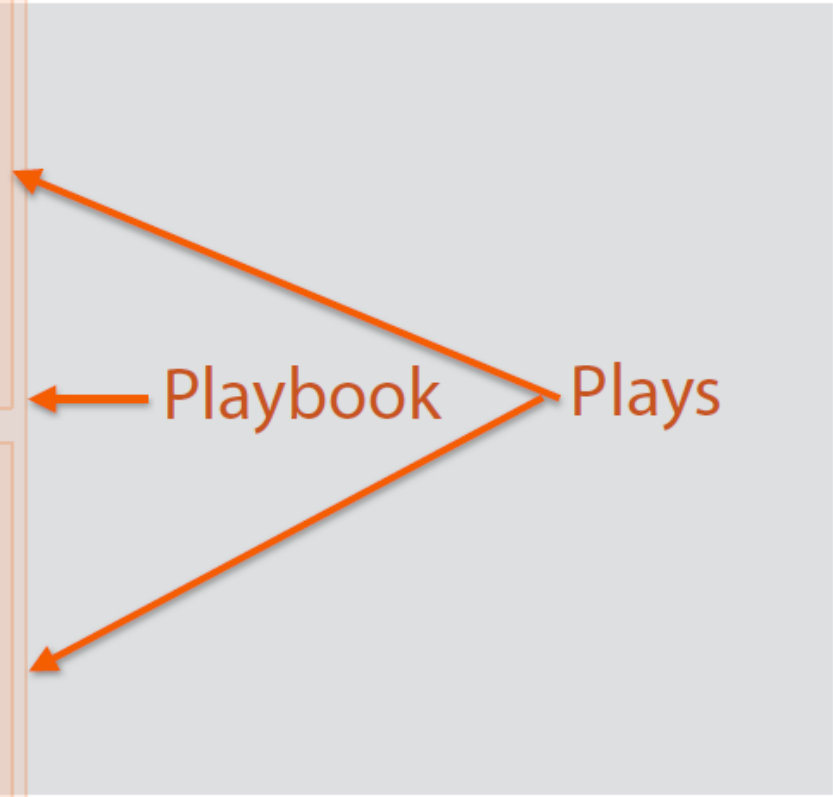


- Plays map hosts to tasks
- A play can have multiple tasks
- A playbook can have multiple plays

Playbook

Playbook Breakdown

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: Install Apache
    yum: name=httpd state=present
  - name: Start Apache
    service: name=httpd state=started

- hosts: dbservers
  remote_user: root
  tasks:
  - name: Install MySQL
    yum: name=mysql-server state=present
  - name: Start MySQL
    service: name=mysqld state=started
```



Playbook

YAML whitespace

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: Install Apache
    yum: name=httpd state=present
  - name: Start Apache
    service: name=httpd state=started

- hosts: dbservers
  remote_user: root
  tasks:
  - name: Install MySQL
    yum: name=mysql-server state=present
  - name: Start MySQL
    service: name=mysqld state=started
```

Note : Whitespace is crucial!

Playbook

Playbook Breakdown

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: Install Apache
    yum: name=httpd state=present
  - name: Start Apache
    service: name=httpd state=started
```

Playbook

Playbook Breakdown

```
---
- hosts: webservers
  remote_user: root
  tasks:
    - name: Install Apache
      yum: name=httpd state=present
    - name: Start Apache
      service: name=httpd state=started
```

Global Play Declaration

Playbook

Playbook Declarations

```
- hosts: webservers
  vars:
    git_repo: https://github.com/repo.git
    http_port: 8080
    db_name: wordpress
  sudo: yes
  sudo_user: wordpress_user
  gather_facts: no
```

Playbook

Playbook Declarations

```
- hosts: webservers
  vars:
    git_repo: https://github.com/repo.git
    http_port: 8080
    db_name: wordpress
  sudo: yes
  sudo_user: wordpress_user
  gather_facts: no
```

Declare Variables per Play

Playbook

Playbook Declarations

```
- hosts: webservers
  vars:
    git_repo: https://github.com/repo.git
    http_port: 8080
    db_name: wordpress
  sudo: yes
  sudo_user: wordpress_user
  gather_facts: no
```

Declare user to run tasks

Playbook

Playbook Declarations

```
- hosts: webservers
  vars:
    git_repo: https://github.com/repo.git
    http_port: 8080
    db_name: wordpress
  sudo: yes
  sudo_user: wordpress_user
  gather_facts: no
```

Don't gather facts on hosts

Playbook

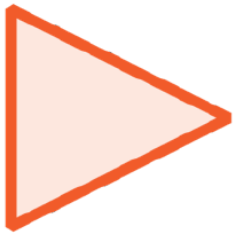
- Tasks are executed in order - top down
- Tasks use modules

Playbook

Tasks

```
tasks:
  - name: Name this task for readability
    module: parameters=go_here

  - name: Deploy Apache Configuration File
    copy: src=/ansible/files/conf/httpd.conf
          dest=/etc/httpd/conf/
```

Playbook



Execution of playbooks:
$ ansible-playbook playbook.yml

Playbook

If a host fails a task, that host is removed from the rest of the playbook execution

Playbook

Retrying failed Host Executions

```
PLAY RECAP *********************************************************************
        to retry, use: --limit @/home/vagrant/ping.retry

db1                        : ok=0    changed=0    unreachable=1    failed=0
web1                       : ok=2    changed=0    unreachable=0    failed=0
```

Playbook

Including Files

Include Files to Extend Playbook

```
tasks:
  - include: wordpress.yml
    vars:
      sitename: My Awesome Site
  - include: loadbalancer.yml
  - include_vars: variables.yml
```

- Breaks up long playbooks
- Use to add external variable files
- Reuse other playbooks

Playbook

Registering Task Output

Grab output of task for another task

```
tasks:
  - shell: /usr/bin/whoami
    register: username
  - file: path=/home/myfile.txt
          owner={{ username }}
```

- Useful to use tasks to feed data into other tasks
- Useful to create custom error trapping

Playbook

Debug Module

Add debug to tasks

```
tasks:
  - debug: msg="This host is
{{ inventory_hostname }} during
execution"

  - shell: /usr/bin/whoami
    register: username
  - debug: var=username
```

- Useful to send output to screen during execution
- Helps find problems

Playbook

Promoting for Input

Prompt user during execution

```
- hosts: web1

  vars_prompt:
   - name: "sitename"
     prompt: "What is new site name?"

  tasks:
   - debug: msg="The name is {{ sitename }}"
```

- Creates Dynamic Playbooks

Playbook

Playbook handlers

- Tasks with asynchronous execution
- Only runs tasks when notified
- Tasks only notify when state=changed
- Does not run until all playbook tasks have executed
- Most common for restarting services to load changes (if changes are made)

Playbook

Handlers

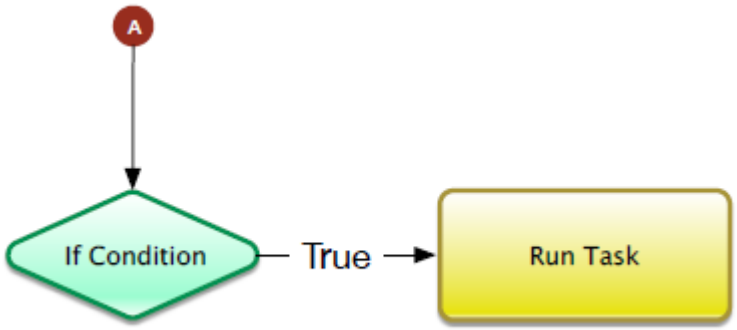
Notify handlers from your tasks

```
tasks:
   - copy: src=files/httpd.conf
           dest=/etc/httpd/conf/
     notify:
        - Apache Restart
 handlers:
   - name: Apache Restart
     service: name=httpd state=restarted
```

- Copies config file to host
- If state=change on "COPY", tell "Apache Restart"
- Run "Service" module.

Playbook

Conditional Execution



Use the clause “when” to choose if task should run.

Playbook

Conditional Clause

Choose when to execute tasks

```
tasks:
   - yum: name=httpd state=present
     when: ansible_os_family == "RedHat"

   - apt: name=apache2 state=present
     when: ansible_os_family == "Debian"
```

- Uses YUM if OS is RedHat
- Uses APT if OS is Debian

Playbook

Conditional Clause Based on Output

Choose when to execute tasks

```
tasks:
  - command: ls /path/doesnt/exist
    register: result
    ignore_errors: yes

  - debug: msg="Failure!"
    when: result|failed
```

- Track whether previous task ran
- Searches JSON result for status
- Status Options:
  - success
  - failed
  - skipped

Playbook

Templates

Uses Jinja2 Engine
Insert variables into static files
Creates and copies dynamic files
Deploy custom configurations

Playbook

Template Module

Modify Template and Copy

```
tasks:
  - template:
      src=templates/httpd.j2
      dest=/etc/httpd/conf/httpd.conf
      owner=httpd
```

- Takes a file with pre-defined variable names
- Inserts variable values in file
- Copies file to destination

Playbook

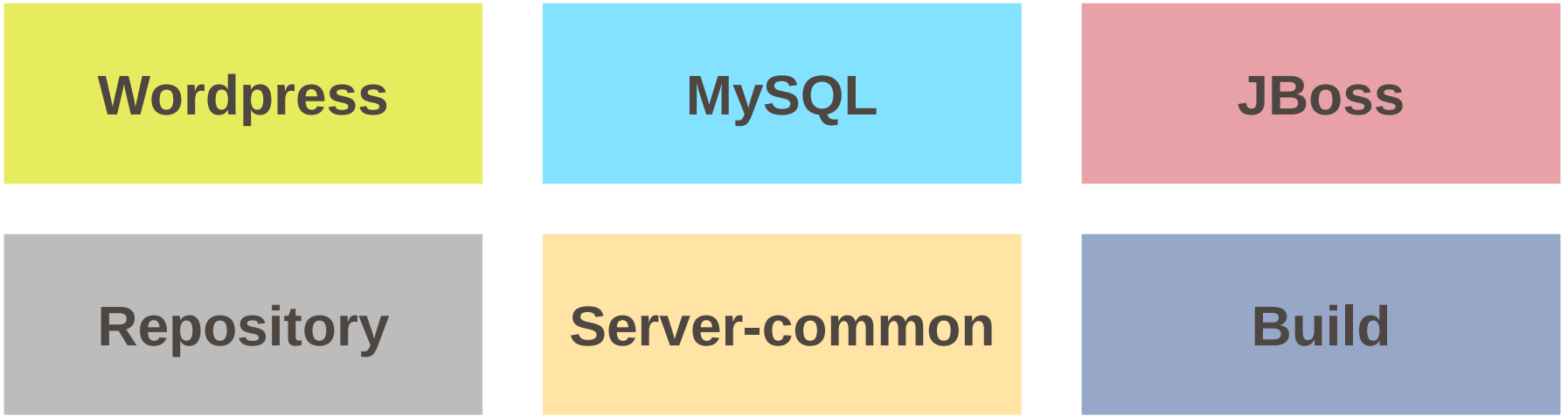
httpd.j2

```
……

<VirtualHost *:80>
    ServerAdmin {{ server_admin }}
    DocumentRoot {{ site_root }}
    ServerName {{ inventory_hostname }}
</VirtualHost>

……
```

Roles

Roles

Roles Examples



Roles

Current Playbook

ROLE: Builder

```
- tasks:
  - name: install gcc
  - name: install jdk
  - name: setup git
```

ROLE: Server-Common

```
- tasks:
  - name: configure SNMP
  - name: configure SYSLOG
  - name: configure NTP
```

ROLE: Repo

```
- tasks:
  - name: git
  - name: configure
  - name: pull latest
```

Roles

Efficient Role Design

BUILD: Compiler/Unit Test Role

Install GCC

Install JDK

Install Unit Testing

Repo: Code Repository Role

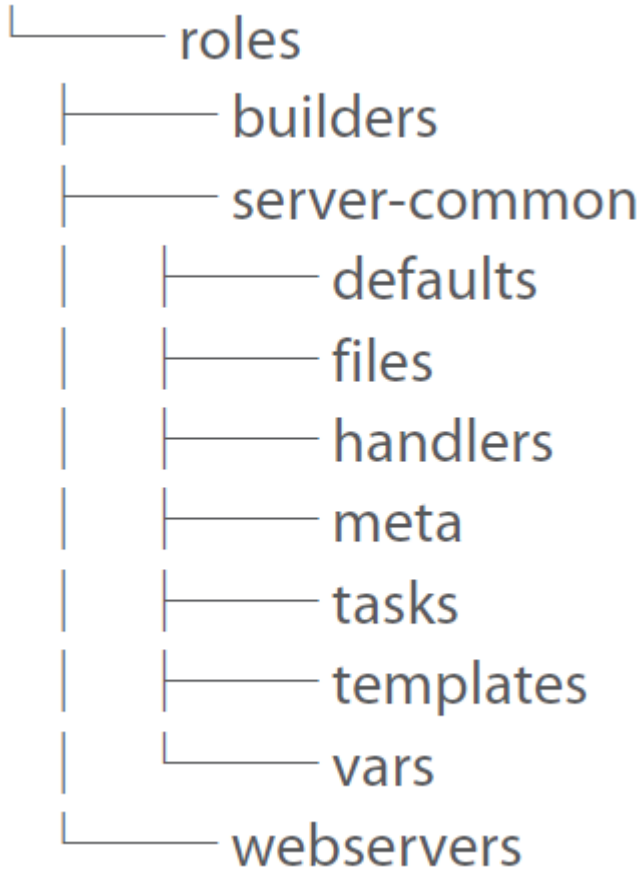
Install Git

Configure Git

Schedule hourly pulls

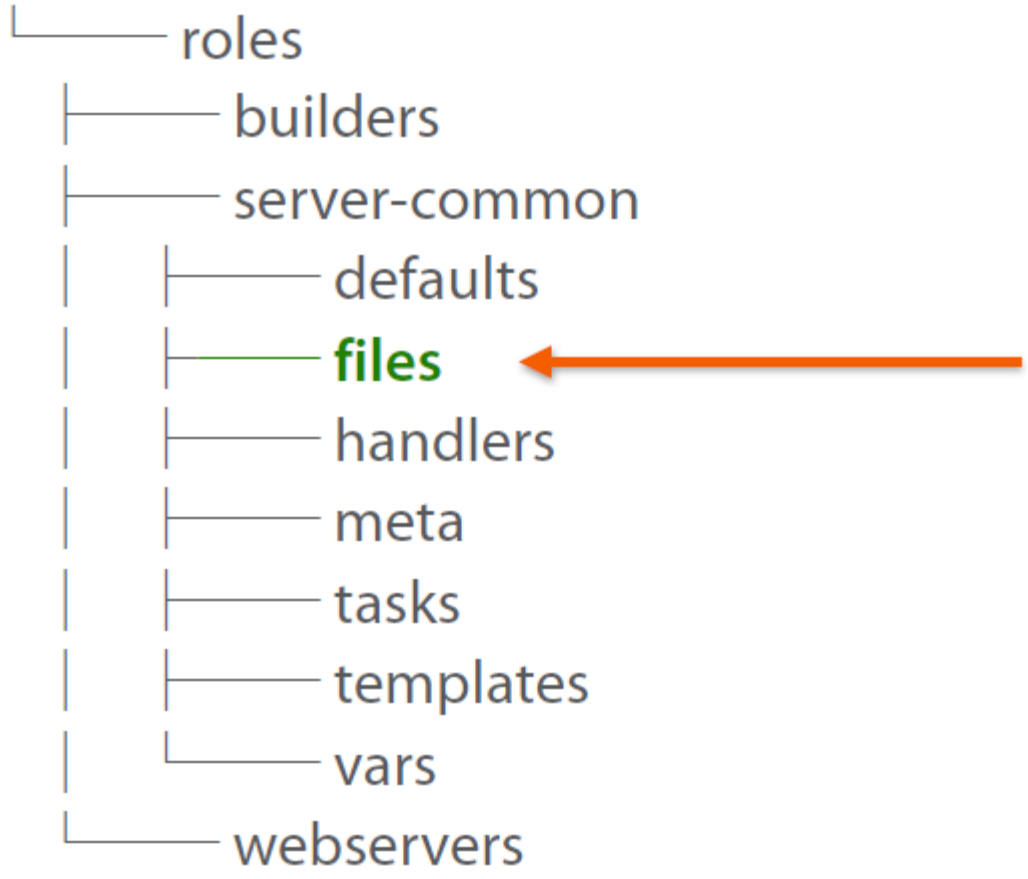
Roles

Directory Structure



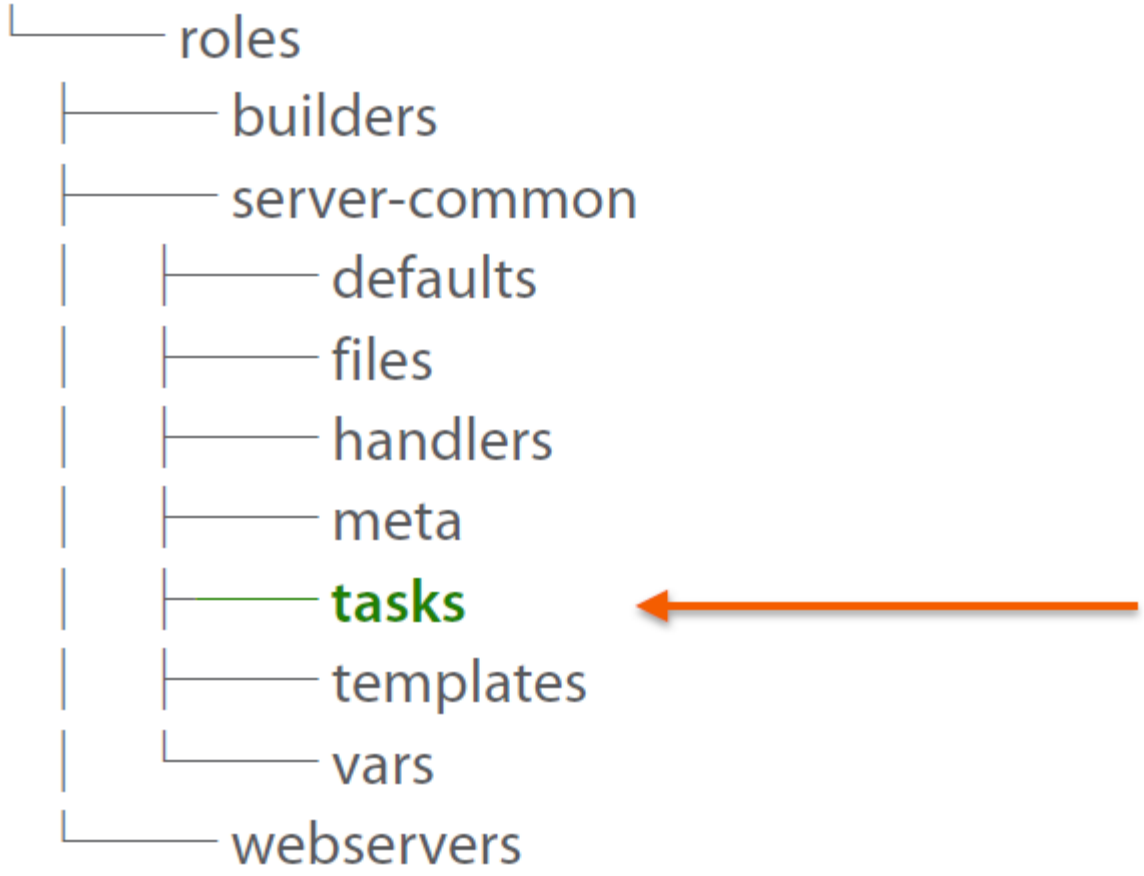
Roles

Directory Structure



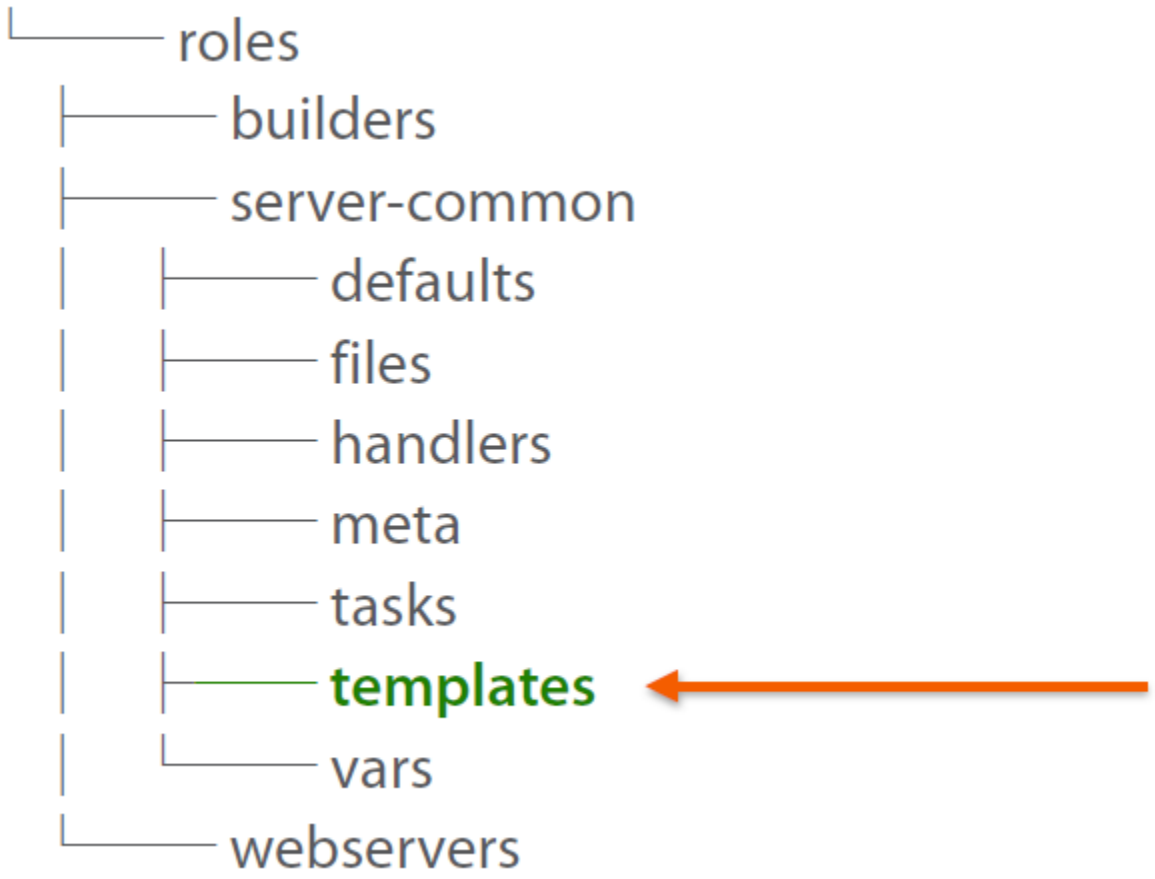
Roles

Directory Structure

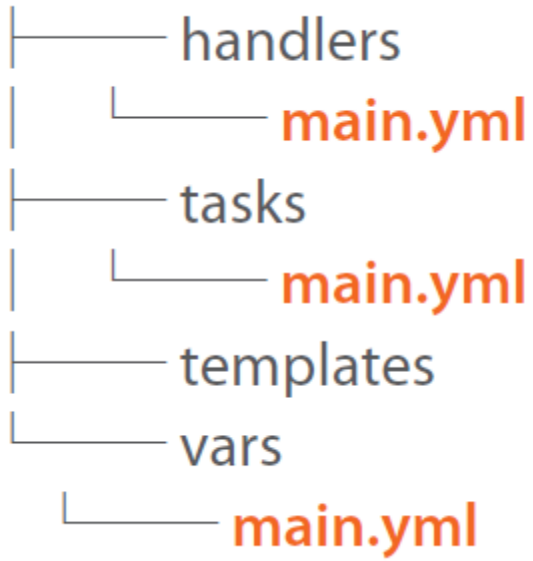


Roles

Directory Structure



Roles



main.yml

Primary file that Ansible looks for

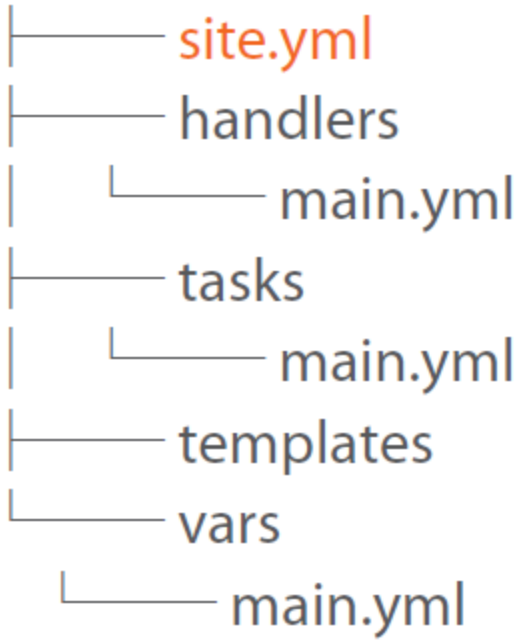
Roles

tasks/main.yml

```
---
- include: webservers.yml
- include: dbservers.yml
```

Can add includes to break-up long files

Roles



site.yml

Primary file to include entire infrastructure

Roles

site.yml

```
---
- include: webservers.yml
- include: dbservers.yml
```

Can add includes to break-up long files

Roles

site.yml

```
---
- include: webservers.yml tags=web
- include: dbservers.yml tags=db
```

Use tags to define categories within your playbooks

Tagging Task

```
tasks:

debug: msg="This will only run on tag 'debug'"

tags:

- debug

debug: msg="You can also use multiple tags"

tags:

- debug

- ubercool
```

Adding Roles to Playbook

```
---
- hosts: code-dev

gather_facts: no

roles:
- server-common
-    builders

tasks:
# Build your extra tasks here like
# creating users, or deploying a specific config
```

Roles

Pre-task and Post-task

pre_tasks:

- Executes plays BEFORE roles
  - Use-Cases
    - Setup of maintenance windows
    - Removing servers from Loadbalancers
    - Silencing alarms

post_tasks:

- Executes plays AFTER roles
  - Use-Cases
    - Clearing of maintenance windows
    - Adding servers to Loadbalancers
    - Enabling Alarms

Adding Pre and Post Tasks

```
---
- hosts: webservers
      pre_tasks:
            - # Remove from load-balancer
      roles:
            - server-common
            - jboss
      post_tasks:
            - # Add to load-balancer
      gather_facts: no
```

Executing Roles - Basic

Basic execution of roles:

```
$ ansible-playbook site.yml
```

Executing Roles - Tags

Tagged execution of roles:

```
$ ansible-playbook site.yml
                —tags “web”
```

Executing Roles – Tags with limits

Limited tagged execution of roles:

```
$ ansible-playbook site.yml
        —tags “web”
        —limit atlanta
```

Getting Roles

Create your own roles

Perfect for proprietary applications or workflows

Find roles to download

Look for others that had the same requirement and shared their work

Roles

Ansible Galaxy

Download

Share

Review

Installing Galaxy Roles

Use username.role

```
$ ansible-galaxy install username.role
```

Installing Galaxy Roles

Use username.role

```
$ ansible-galaxy install apaxson.ultimate
```

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

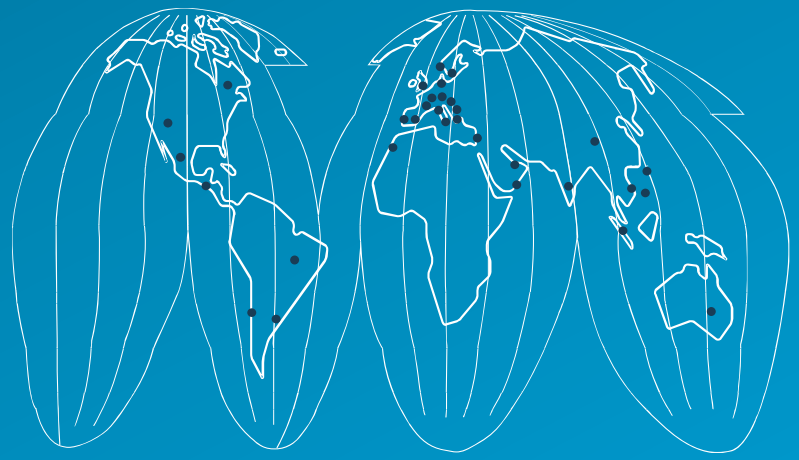
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com