Automate IT with

Maria Dominguez

LUNATECH

Nov. 2017

Agenda

Day 1

- What / Why / When Ansible?
- Unix / YAML
- Components
- Best practices

Day 2

- Cluster provisioning
- Application deployment

What / Why / When Ansible?

- Ansible is an IT Automation engine where you define your configuration.
- Ansible loves repetitive work that we hate :)

Why	When
Human readable	Provisioning
No special coding skills	Configuration Management
Tasks executed in order	App deployment
Powerful	Continuous Delivery
Flexible	Security & Compliance
Positive team impact	Orchestration

How Ansible works

1. First, your admin client connects to the target servers via **`ssh`**. These servers are managed in a **INI** file.

   Agents are not required. What you only need is **python** and a user that can **`ssh`** to the server (**`root`** user is not required, you can login with any user and then **`su`**/**`sudo`** to any other user).

2. Then Ansible starts gathering **facts** about the machine (OS running in the server, what packages are installed, which services are running, etc.).

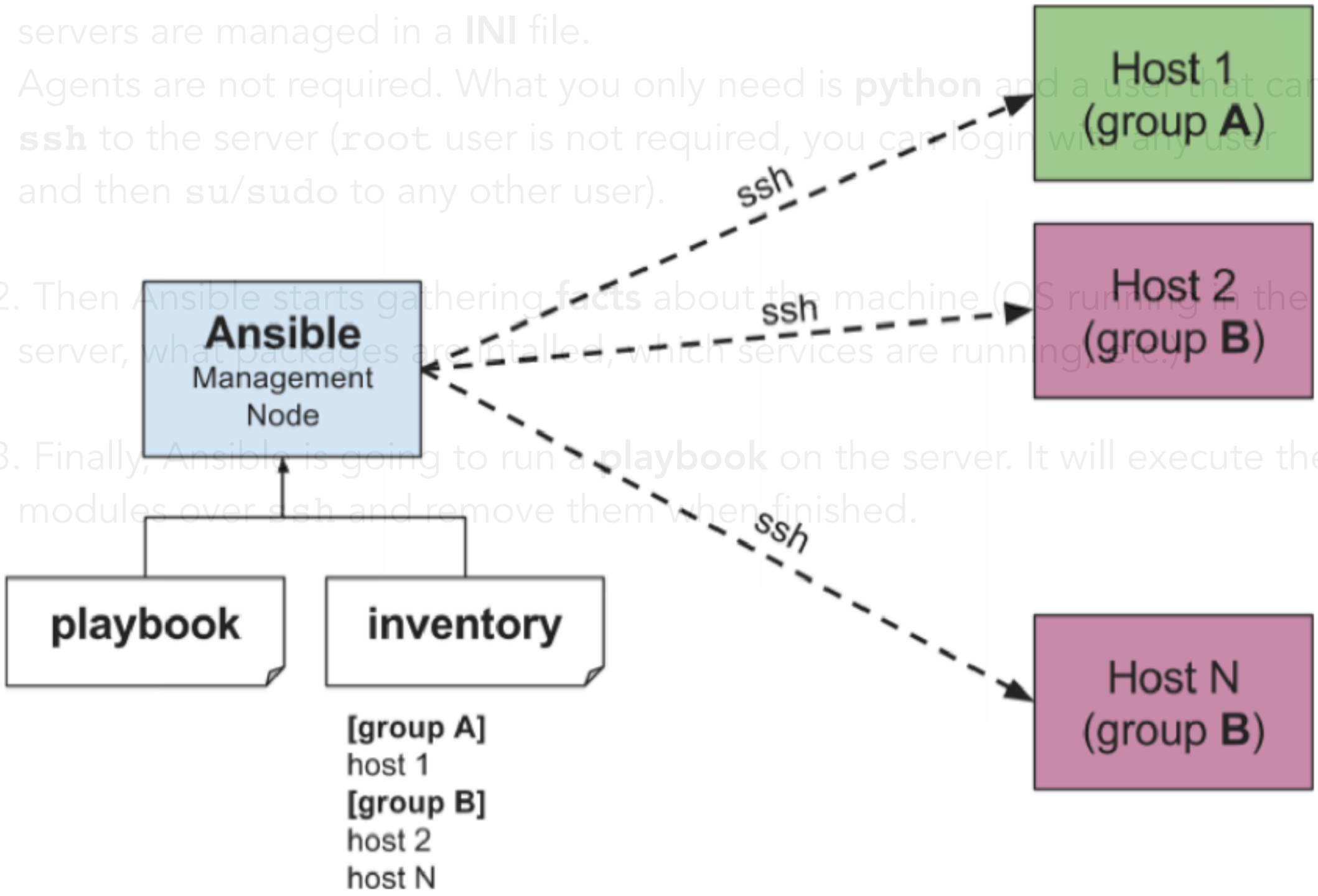
3. Finally, Ansible is going to run a **playbook** on the server. It will execute the modules over **`ssh`** and remove them when finished.

How Ansible works

1. First, your admin client connects to the target servers via **`ssh`**. These servers are managed in a **INI** file.
   Agents are not required. What you only need is **python** and a user that can **`ssh`** to the server (**`root`** user is not required, you can login with any user and then **`su`**/**`sudo`** to any other user).

2. Then Ansible starts gathering **facts** about the machine (OS running in the server, what packages are installed, which services are running, etc.)

3. Finally, Ansible is going to run a **playbook** on the server. It will execute the modules over ssh and remove them when finished.



A tiny bit of UNIX

- **OpenSSH**: By default, will try to use native OpenSSH for remote communication when possible.
  - ✦ *It is the premier connectivity tool for remote login with SSH protocol.*
  - ✦ *It encrypts all traffic.*
  - ✦ *Provides a large suite of secure tunnelling capabilities, several authentication methods and sophisticated configuration options.*

- **SSH keys** for authentication. To set up SSH agent to avoid retyping passwords, you can do

```
ssh-agent bash
ssh-add ~/.ssh/id_rsa
```

- `/etc/ansible/hosts`: Edit and put one or more remote systems in it.

```
localhost ansible_connection=local
```

A bit of YAML

file - extension	`.yaml  or  .yml`	`ansible_workshop.yml`
file - beginning	`---`	
file - end	…	
comment	#	`# This is a comment`

dictionary	`key:value`	**`presenter:`**
		`    `**`name:`** `Maria Dominguez`
		`    `**`job:`** `Developer`
		`    `**`skill:`** `Ansible`
		
		`presenter:` `{`**`name:`** `Maria Dominguez,` **`job:`** `Developer,` **`skill:`** `Ansible}`

list	`-`
	`[ ]`
		`lunatech_offices:`
		`    - Amsterdam`
		`    - Rotterdam`
		
		`lunatech_offices:` `['Amsterdam', 'Rotterdam']`

multiple lines	`|`
	`>`
		`include_newlines:` `|`
		`    exactly as you see, it will appear`
		`    two lines of text`
		
		`ignore_newlines:` `>`
		`    this is just a single line of text`
		`    despite appearances`

variables	“{{ variable }}”	**`name:`** `"{{ presenter.name }}"` **`description:`** `"{{ presenter.name }} is a {{ presenter.job }}”`

A bit of YAML

employee.yml

```
---
# An employee record
name: Maria Dominguez
job: Developer
joined: 2015
employed: true
offices:
    - Amsterdam
    - Rotterdam
technologies:
    scala: advanced
    ansible: advanced
    aws: novice
career: |
    Machine learning & Spark
    Scala, Akka & DevOps
...
```

Installation

```
$ sudo easy_install pip
$ sudo pip install ansible
```

```
$ ansible --version

ansible 2.4.0
  config file = None
  configured module search path = [u'/Users/mariadominguez/.ansible/plugins/modules', u'/usr/share/
ansible/plugins/modules']
  ansible python module location = /Library/Python/2.7/site-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 2.7.10 (default, Feb  7 2017, 00:08:15) [GCC 4.2.1 Compatible Apple LLVM 8.0.0
(clang-800.0.34)]
```

Ansible commands

`ansible`	Define and run a single task 'playbook' against a set of hosts
`ansible-playbook`	Runs Ansible playbooks, executing the defined tasks on the targeted hosts
`ansible-config`	View, edit, and manage ansible configuration
`ansible-console`	REPL console for executing Ansible tasks
`ansible-doc`	Documentation tool
`ansible-galaxy`	Downloads roles from other ansible users
`ansible-vault`	Encryption/decryption utility for Ansible data files

Components

Playbook

Modules

Roles

Variables

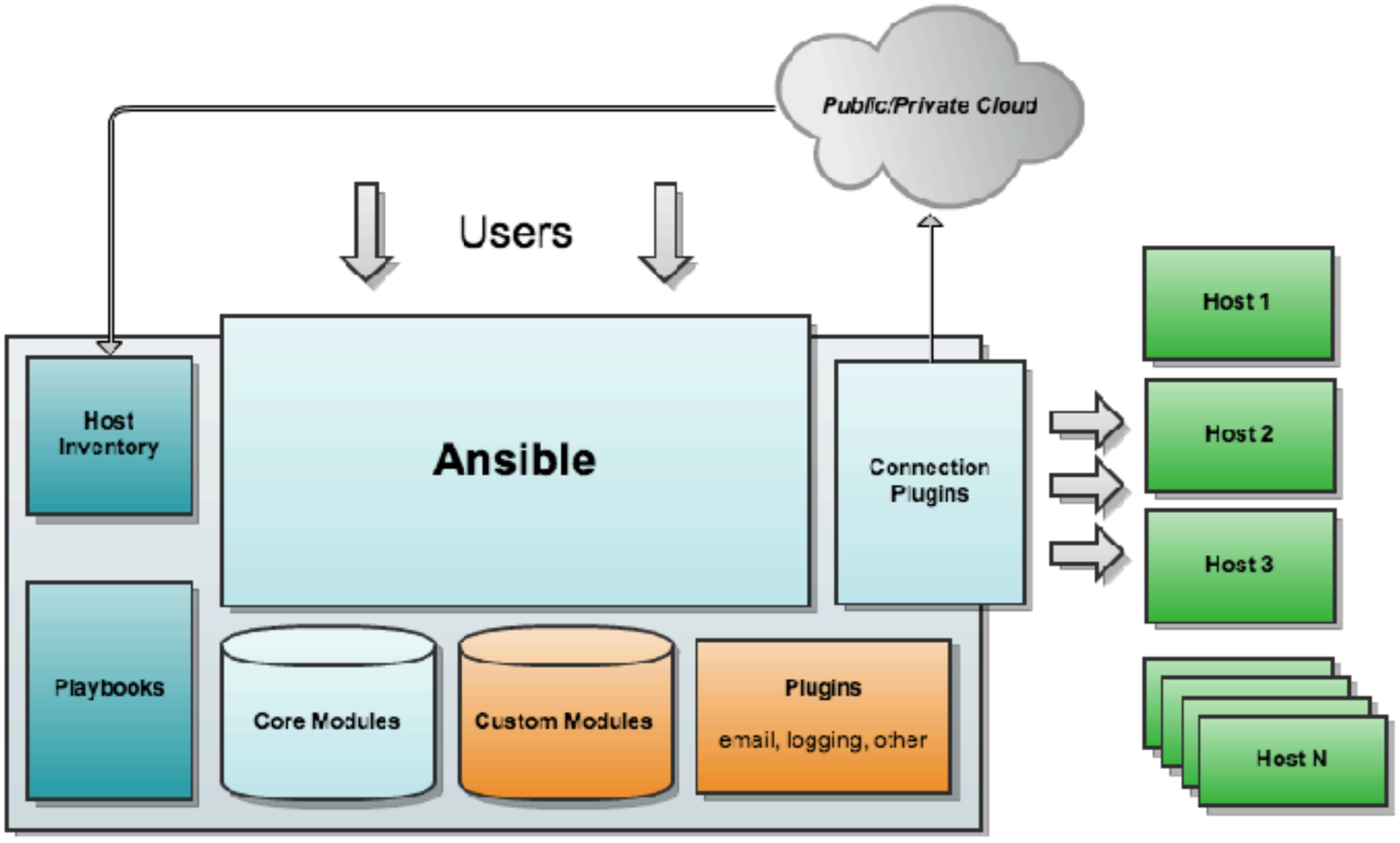
Templates

Jinja2

Inventory

Configuration file

Vault



https://galaxy.ansible.com/

Components: **Playbook**

Playbook is the entry point in Ansible. It defines the description of actions that you want to apply to your system.

Basics: manage configurations of and deployments to remote machines.

Advanced: sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way.

Each playbook is composed of one or more '**plays**' in a list.

The goal of a **play** is to map a group of hosts to some well defined **roles**, represented by things Ansible calls **tasks**.

At a basic level, a **task** is nothing more than a call to an Ansible **module**.

```
ansible-playbook playbook.yml --syntax-check
```

Components: Playbook

1 playbook which contains 1 play

```
---
- hosts: webservers
  vars:
    http_port: 80
    max clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Components: **Playbook - Play - Task**

Each **play** contains a **list of tasks**.

Tasks are executed **in order**, one at a time, against **all machines** matched by the host pattern, before moving on to the next task.

It is important to understand that, within a play, all hosts are going to get the same task directives. It is the purpose of a play to map a selection of hosts to tasks.

When running the playbook, which runs **top to bottom**, hosts with failed tasks are taken out of the rotation for the entire playbook. If things fail, simply correct the playbook file and rerun.

The goal of each **task** is to **execute a module**, with very specific arguments.

Components: Playbook

1 playbook which contains 2 plays

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf

- hosts: databases
  remote_user: root
  tasks:
  - name: ensure postgresql is at the latest version
    yum: name=postgresql state=latest
  - name: ensure that postgresql is started
    service: name=postgresql state=started
```

```
become: yes
become_user: spark
become_method: sudo
```

Components: **Playbook**

```
ansible-playbook [options] playbook.yml [playbook2 ...]
```

Options	Connection options
--check	--ask-pass
--syntax-check	--user=REMOTE_USER
--verbose	--connection=CONNECTION
--list-hosts	--ssh-common-args=SSH_COMMON_ARGS
--list-tasks	--ssh-extra-args=SSH_EXTRA_ARGS
--extra-vars=EXTRA_VARS	--become
--start-at-task=START_AT_TASK	--become-user=BECOME_USER

Components: **Playbook - include / import**

Statements can be reused by other playbooks or tasks..

```
---
- import_playbook: webservers.yml
- import_playbook: databases.yml
```

- All `import*` statements are pre-processed at the time playbooks are parsed. *[static]*

- All `include*` statements are processed as they encountered during the execution of the playbook. *[dynamic]*

common_tasks.yml

```
---
- name: placeholder foo
  command: /bin/foo
- name: placeholder bar
  command: /bin/bar
```

```
tasks:
- import_tasks: common_tasks.yml
# or
- include_tasks: common_tasks.yml
```

Components: **Modules**

All work in Ansible is performed by the Modules.
User can define his own modules.
Documentation: **`ansible-doc command`** or http://docs.ansible.com

- Nearly all modules take `key=value` arguments, space delimited.

- Some modules take no arguments.

- The `command`/`shell` modules simply take the string of the command you want to run.

```
- name: reboot the servers
  command: /sbin/reboot -t now
```

```
- name: restart webserver
  service:
    name: httpd
    state: restarted
```

Components: **Modules**

List of available modules

[http://docs.ansible.com/ansible/latest/modules_by_category.html]

- **Cloud**: Amazon, Docker, VMWare
- **Commands**: command, script, shell
- **Files**: copy, find, file, replace, [un]archive, template
- **Notification**: mail, hipchat, slack, telegram
- **Packaging**: easy_install, maven_artifact, nom, pip, apt, homebrew
- **Source control**: git, github, subversion
- **System**: cron, user, group, hostname, mount, ping, service, sysctl, systemd
- **Utilities**: debug, set_fact, set_stats, wait_for, fail, pause, import_role
- **Web infrastructure**: jira, jenkins, nginx

Components: **Modules**

- All modules technically return JSON format data.

Return values	Description
`changed`	True if the task had to make changes, false otherwise.
`failed`	True if the task was failed, false otherwise.
`msg`	Generic message for the user.
`rc`	Return code from command line utilities (shell, command, etc.).
`results`	For loops, normal module result per item.
`stderr[_lines]`	Error from command line utilities (shell, command, etc.).
`stdout[_lines]`	Output from command line utilities (shell, command, etc.).
`exception`	Traceback information caused by an exception in a module. (-vvv)
`warnings`	Information presented to the user.

"If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material."

Components: **Exercise**

Create a playbook that creates a folder in the current directory, and copies a random file to the newly created folder.

```
.
├── playbook-dirs.yml
└── test-dir
    └── test-file
```

Components: **Exercise**

Create a playbook that creates a folder in the current directory, and copies a random file to the newly created folder.

The user of the directory must be root and only this root user can have write/read/execute permissions on the file created (other will have no access at all).

```
.
├── playbook-dirs.yml
└── test-dir
    └── test-file
```

```
-rw-r--r--  mariadominguez  staff   playbook.yml
drw-rw-r--  root            staff   test-dir

-rwx------  root            wheel   test-file
```

Components: **Roles**

Roles are ways of automatically loading certain functionality.

Roles expect files to be in certain directory names.

Directories must contain a **`main.yml`** file with relevant info:

- **tasks**: list of tasks executed by the role.
- **defaults**: default variables for the role.
- **vars**: other variables for the role.
- **files**: files which can be deployed via this role.
- **templates**: templates which can be deployed via this role.
- **meta**: meta data for this role.
- **handlers**: list of tasks referenced by a globally unique name.

```
---
- hosts: webservers
  roles:
     - common
     - webservers
```

```
site.yml
webservers.yml
fooservers.yml
roles/
   common/
     tasks/
     files/
     templates/
     vars/
     ...
   webservers/
     ...
```

Each role listed in `roles` will execute in turn.

Any role dependencies defined in the roles `meta/main.yml` will be run first.

`Tasks` defined in the play will be run.

Components: **Roles**

- Roles can appear inline with `import_role`, or `include_role`.

```
---
- hosts: webservers
  tasks:
  - debug:
      msg: "before we run our role"
  - import_role:
      name: example1
    when: "ansible_os_family == 'RedHat'"
  - include_role:
      name: example2
  - debug:
      msg: "after we ran our role"
```

- Roles can accept parameters:

```
---
- hosts: webservers
  roles:
    - common
    - { role: foo_app_instance, dir: '/opt/a', app_port: 5000 }
    - { role: foo_app_instance, dir: '/opt/b', app_port: 5001 }
```

```
meta/main.yml
allow_duplicates: true
```

- Roles name can be a qualified path: `{ role: '/path/to/my/roles/common' }`

Components: **Roles**

Role dependencies: automatically pull in other roles when using a role.

Stored in the `meta/main.yml`.

Always executed before.

Dependencies may be recursive.

Ansible searches for roles in (1) `roles/` and (2) `/etc/ansible/roles`.

```
---
dependencies:
  - { role: common, some_parameter: 3 }
  - { role: apache, apache_port: 80 }
  - { role: postgres, dbname: blarg, other_parameter: 12 }
```

```
---
dependencies:
- { role: wheel, n: 1 }
- { role: wheel, n: 2 }
- { role: wheel, n: 3 }
- { role: wheel, n: 4 }
```

```
---
dependencies:
- { role: tire }
- { role: brake }
```

```
tire(n=1)
brake(n=1)
wheel(n=1)
tire(n=2)
brake(n=2)
wheel(n=2)
...
car
```

Components: **Variables**

Valid name: letters, numbers and underscores. Always start with a letter.

On dictionaries:

```
foo:
  bar: one
  baz: two
```

```
foo['bar']
foo.bar
```

Where do we define variables?

- Inventory
- Playbook
- Included files and roles

```
[dev]
master ansible_host=master.dev.com ansible_port=22
slave1 ansible_host=slave1.dev.com ansible_port=22
slave2 ansible_host=slave2.dev.com ansible_port=22

[dev:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

```
---
- hosts: webservers
  vars:
    http_port: 80
```

```
---
- hosts: webservers
  tasks:
  - include_role:
      name: foo_app_instance
    vars:
      dir: '/opt/a'
      app_port: 5000
  ...
```

```
---
dir: '/opt/a'
app_port: 5000
```

Components: **Variables - Jinja2 Filters**

Filters in *Jinja2* are a way of transforming template expressions from one kind of data into another.

```
- hosts: app_servers
  vars:
       app_path: "{{ base_path }}/22"
```

Variable scopes:

- **Global**: Set by config, environment variables and command line.
- **Play**: Each play and contained structures, var entries, role defaults and vars.
- **Host**: Directly associated to a host (inventory, include_vars, facts, registered tasks outputs)

Components: **Variables**

Variable precedence:

- role defaults
- inventory file or script group vars
- inventory group_vars/all
- playbook group_vars/all
- inventory group_vars/*
- playbook group_vars/*
- inventory file or script host vars
- inventory host_vars/*
- playbook host_vars/*
- host facts
- play vars
- play vars_prompt
- play vars_files
- role vars (defined in role/vars/main.yml)
- block vars (only for tasks in block)
- task vars (only for the task)
- role (and include_role) params
- include params
- include_vars
- set_facts / registered vars
- extra vars (always win precedence)

Components: **Jinja2 - Filters**

Filters: `{{ my_variable | filter }}`

- **Formatting data**: to_json, to_yaml, to_nice_json, from_json
- **Variable definition**: mandatory, default(value), default(omit)
- **Lists**: min, max
- **Theory**: union(another_list), intersect(another_list)
- **Random**: random, random(step), random(seed)
- **Network**: ipaddr, ipv4, ipv6
- **Hashing**: hash('md5'), checksum
- **URL**: urlsplit, urlsplit('hostname')
- **Regex**: regex_search, regex_replace, regex_escape
- Others: ternary, basename, realpath, b64decode, to_uuid, type_debug

Components: **Exercise**

Given the variable `user_data`, write a set of tasks which perform the following actions:

```
user_data:
  name: Maria Dominguez
  job: Developer
  joined: 2015
  employed: true
  offices:
    - Amsterdam
    - Rotterdam
  technologies:
    scala: advanced
    ansible: advanced
    aws: novice
  career: |
    Machine learning & Spark
    Scala, Akka & DevOps
```

- Save the `user_data` as json in /tmp/user

- Extract the level of `scala` and save it to a variable

- If the level of `scala` is “advanced”, print career.

- Extract the level of a not given technology and provide a default message instead.

- Check if “Leiden” is one of the offices.

Components: **Templating**

Ansible uses Jinja2 templating to enable dynamic expressions and access to variables.

All templating happens on the Ansible controller before the task is sent and executed on the target machine.

```
#mytemplates/foo.j2

{% if app_args is defined %} {{app_args | replace(“,”, “ ”) }} {% endif %}
base_url=http://example.com
user=maria
role=presenter
```

http://jinja.pocoo.org/docs/2.10/templates/

```
- template:
    src: /mytemplates/foo.j2
    dest: /etc/file.conf
    owner: bin
    group: wheel
    mode: 0644
```

Components: **Exercise**

Using templates, create a playbook that generates a configuration file **`kafka.conf`** with the format

```
bootstrap_server: localhost
port: 9092
topic: test-topic
replication_factor: 3
```

and places it under **`/tmp/kafka/conf/`**

Rules:

- Ensure that **`bootstrap_server`**, **`port`** and **`topic`** are present (**`replication_factor`** is optional, and should appear in the file only if it is defined).
- **`bootstrap_server`** must appear in lowercase.

Components: Jinja2 - Tests

Strings

```
- debug: msg="A includes B"
  when: a|issuperset(b)
```

Lists

```
- debug: "msg='matched pattern 1'"
  when: url | match("http://example.com/users/.*/resources/.*")
```

Paths

```
- debug: msg="path is a directory"
  when: mypath|is_dir

- debug: msg="path is a file"
  when: mypath|is_file
```

Task results

```
- shell: /usr/bin/foo
  register: result
  ignore_errors: True

- debug: msg="it failed"
  when: result|failed

- debug: msg="it changed"
  when: result|changed

- debug: msg="it succeeded in Ansible >= 2.1"
  when: result|succeeded

- debug: msg="it succeeded"
  when: result|success

- debug: msg="it was skipped"
  when: result|skipped
```

Components: **Conditionals**

Sometimes you will want to skip a particular step on a particular host: `when`

```
tasks:
  - name: "shut down CentOS 6 and Debian 7 systems"
    command: /sbin/shutdown -t now
    when: (ansible_distribution == "CentOS" and ansible_distribution_major_version == "6") or
          (ansible_distribution == "Debian" and ansible_distribution_major_version == "7")
```

```
tasks:
  - name: "shut down CentOS 6 systems"
    command: /sbin/shutdown -t now
    when:
      - ansible_distribution == "CentOS"
      - ansible_distribution_major_version == "6"
```

```
tasks:
  - command: /bin/false
    register: result
    ignore_errors: True

  - command: /bin/something
    when: result|failed

  - command: /bin/something_else
    when: result|succeeded

  - command: /bin/still/something_else
    when: result|skipped
```

Components: **Loops**

Combining `when` with `with_items`.

!! Be aware that the when statement is processed separately for each item **!!**

```
tasks:
    - command: echo {{ item }}
      with_items: [ 0, 2, 4, 6, 8, 10 ]
      when: item > 5
```

```
# This will run debug three times since
# the list is flattened

- debug:
    msg: "{{ item }}"
  vars:
    nested_list:
      - - one
        - two
        - three
  with_items: "{{ nested_list }}"
```

```
# This will run debug once with the three
# items

- debug:
    msg: "{{ item }}"
  vars:
    nested_list:
      - - one
        - two
        - three
  with_items:
    - "{{ nested_list }}"
```

Components: Loops over...

Hashes

```yaml
---
users:
  maria:
    name: Maria Dominguez
    mb: "+31652883890"
  bob:
    name: Bob Bananarama
    telephone: 987-654-3210
---
tasks:
  - name: Print phone records
    debug:
      msg: "{{ item.value.name }}: {{ item.value.mb }}"
    with_dict: "{{ users }}"
```

Files

```yaml
---
- hosts: all

  tasks:
    # debug with the content of each file.
    - debug:
        msg: "{{ item }}"
      with_file:
        - first_example_file
        - second_example_file
```

Parallel sets of data

```yaml
---
keys: [ 'a', 'b', 'c', 'd' ]
values:  [ 1, 2, 3, 4 ]

---
tasks:
  - debug:
      msg: "{{ item.0 }} and {{ item.1 }}"
    with_together:
      - "{{ keys }}"
      - "{{ values }}"
```

Random choices

```yaml
- debug:
    msg: "{{ item }}"
  with_random_choice:
     - "go through the door"
     - "drink from the goblet"
     - "press the red button"
     - "do nothing"
```

Components: Loops over...

Do until

```
- shell: /usr/bin/foo
  register: result
  until: result.stdout.find("all systems go") != -1
  retries: 5
  delay: 10
```

Results of a program execution

```
- name: Example of looping over a command
       result
  shell: "/usr/bin/frobnicate {{ item }}"
  with_lines:
    - "/usr/bin/frobnications_per_host --param
      {{ inventory_hostname }}"
```

First found

```
- name: INTERFACES | Create Ansible header for
       /etc/network/interfaces
  template:
    src: "{{ item }}"
    dest: "/etc/foo.conf"
  with_first_found:
    - "{{ ansible_virtualization_type }}_foo.conf"
    - "default_foo.conf"
```

Others:

```
- with_indexed_items
- with_nested
- with_filetree
- with_fileglob
- with_subelements
- with_sequence
- with_indexed_items
- with_ini
- with_flattened
```

Components: **Exercise**

Create a playbook that performs the following tasks:

- create a /tmp/test-loops-source and /tmp/test-loops-target directories

- creates 2 files on /tmp/test-loops-source

- copy the 2 different files to /tmp/test-loops-target

- for one of the files on /tmp/test-loops-source, rename the file to “file1-readable” only if the file is readable

Components: **Inventory**

Ansible needs an inventory file where the server(s) definition is specified so it can connect to the hosts to perform the required actions.

It can be in different formats (INI, YAML)

```
[dev]
master ansible_host=master.dev.com ansible_port=22 ansible_user='admin' ansible_private_key_file='~/.ssh/id_rsa'
slave1 ansible_host=slave1.dev.com ansible_port=22 ansible_user='admin' ansible_private_key_file='~/.ssh/id_rsa'
slave2 ansible_host=slave2.dev.com ansible_port=22 ansible_user='admin' ansible_private_key_file='~/.ssh/id_rsa'
```

ansible_connection	Connection type to the host. This can be the name of any of ansible's connection plugins. SSH protocol types are **smart**, **ssh** or **paramiko**. The default is **smart**.
ansible_host	The name of the host to connect to, if different from the alias you wish to give to it.
ansible_port	The ssh port number, if not 22
ansible_user	The default ssh user name to use.
[…]	http://docs.ansible.com/ansible/latest/intro_inventory.html#list-of-behavioral-inventory-parameters

Components: **Configuration file**

Certain settings in Ansible are adjustable via a configuration file.

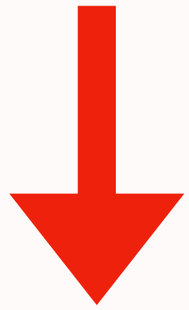
Changes can be made and used in a configuration file which will be processed in the following order:

ANSIBLE_CONFIG (an environment variable)

ansible.cfg (in the current directory)

.ansible.cfg (in the home directory)

/etc/ansible/ansible.cfg



https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg

http://docs.ansible.com/ansible/latest/intro_configuration.html#general-defaults

Components: **Vault**

To keep sensitive data encrypted (passwords, keys, etc.)

ansible-vault	--ask-vault-pass --vault-password-file

What can be encrypted with vault?:

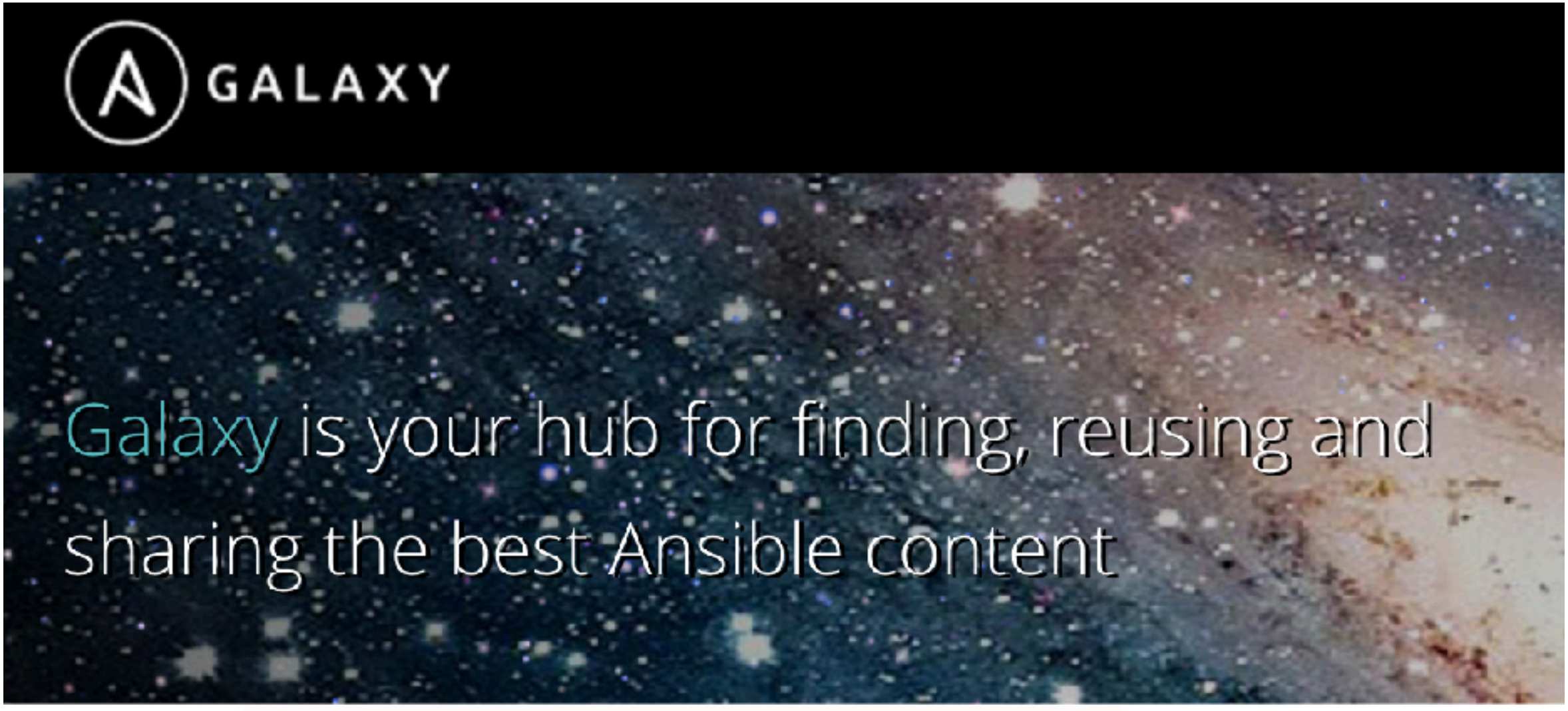
- Variables (hosts, passed by command line),
- tasks,
- files,
- etc.

Components: **Vault**

Create encrypted file	`ansible-vault create foo.yml`
Edit encrypted file	`ansible-vault edit foo.yml`
Encrypt unencrypted file	`ansible-vault encrypt foo.yml`
Decrypt encrypted file	`ansible-vault decrypt foo.yml`
Rekeying encrypted file	`ansible-vault rekey foo.yml`
View content of encrypted file	`ansible-vault view foo.yml`

Running a playbook with vault	`ansible-playbook site.yml --ask-vault-pass`
	`ansible-playbook site.yml --vault-password-file ~/.vault_pass.txt`
	`ansible-playbook site.yml --vault-password-file ~/.vault_pass.py`

Ansible Galaxy



[https://galaxy.ansible.com/]

Keep It Simple !!!!

If something feels complicated, it probably is, and may be a good opportunity to simplify things.

Best practices: Recommended directory layout

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group_1                # here we assign variables to particular groups
host_vars/
   hostname_1             # if systems need specific variables, put them here

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (opt)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/main.yml    #  <-- tasks file can include smaller files if warranted
        handlers/main.yml #  <-- handlers file
        templates/
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/main.yml     #  <-- variables associated with this role
        defaults/main.yml #  <-- default lower priority variables for this role
        meta/main.yml     #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above
    monitoring/
```

Best practices: Well-organised inventory. Dynamic for cloud

```
[atlanta-webservers]
www-atl-1.example.com
www-atl-2.example.com

[boston-webservers]
www-bos-1.example.com
www-bos-2.example.com

[atlanta-dbservers]
db-atl-1.example.com
db-atl-2.example.com

[boston-dbservers]
db-bos-1.example.com

# webservers in all geos
[webservers:children]
atlanta-webservers
boston-webservers

# dbservers in all geos
[dbservers:children]
atlanta-dbservers
boston-dbservers

# everything in the atlanta geo
[atlanta:children]
atlanta-webservers
atlanta-dbservers

# everything in the boston geo
[boston:children]
boston-webservers
boston-dbservers
```

```
[servers]
www-atl-1.example.com
www-atl-2.example.com
www-bos-1.example.com
www-bos-2.example.com
db-atl-1.example.com
db-atl-2.example.com
db-bos-1.example.com
```

Best practices: Group and host variables

```
---
# file: group_vars/all
ntp: ntp-boston.example.com
backup: backup-boston.example.com
```

```
---
# file: host_vars/db-bos-1.example.com
foo_agent_port: 86
bar_agent_port: 99
```

```
---
# file: group_vars/atlanta
ntp: ntp-atlanta.example.com
backup: backup-atlanta.example.com
```

```
---
# file: group_vars/webservers
apacheMaxRequestsPerChild: 3000
apacheMaxClients: 900
```

Best practices: **Group by roles / playbook**

Roles are ways of automatically loading certain vars_files, tasks, and handlers based on a known file structure.

Grouping content by roles also allows easy sharing of roles with other users.

```
---
# file: site.yml
- import_playbook: webservers.yml
- import_playbook: dbservers.yml
```

```
---
# file: webservers.yml
- hosts: webservers
  roles:
    - common
    - webtier
```

```
ansible-playbook site.yml --limit webservers
ansible-playbook webservers.yml
```

Best practices: Always mention the `state` (when possible)

```
- name: Ensure apache is present
  yum:
    name: httpd
    state: present
```

```
- name: Ensure apache is running
  service:
    name: httpd
    state: started
```

```
- name: Ensure apache is at the latest version
  yum: name=httpd state=latest
```

```
- name: Installs nginx web server
  apt: pkg=nginx state=installed
  notify:
    - start nginx
```

```
- name: Restart memcached
  service: name=memcached state=restarted
```

Best practices: Use `creates` and `removes` (when possible)

```
- name: Change the working directory to somedir/ before executing the command.
  shell: somescript.sh >> somelog.txt
  args:
    chdir: somedir/
```

```
- name: This command will change the working directory to somedir/ and will
        only run when somedir/somelog.txt does not exist.
  shell: somescript.sh >> somelog.txt
  args:
    chdir: somedir/
    creates: somelog.txt
```

```
- name: This command will change the working directory to somedir/ and will
        only run when somedir/somelog.txt does exist.
  shell: somescript.sh >> somelog.txt
  args:
    chdir: somedir/
    removes: somelog.txt
```

Best practices: **Always name tasks**

```
- file:
    path: /tmp/app/conf
    state: directory
```

```
TASK [file] ********************************************************
ok: [localhost] => {"changed": false, "failed": false, "gid": 0, "group":
"wheel", "mode": "0755", "owner": "mariadominguez", "path": "/tmp/app/conf",
"size": 68, "state": "directory", "uid": 501}
```

```
- name: Create directory to store configuration files
  file:
    path: /tmp/app/conf
    state: directory
```

```
TASK [Create directory to store configuration files] ***********************
changed: [localhost] => {"changed": true, "failed": false, "gid": 0, "group":
"wheel", "mode": "0755", "owner": "mariadominguez", "path": "/tmp/app/conf",
"size": 68, "state": "directory", "uid": 501}
```

Best practices: Increase verbosity (-v, -vv, -vvv, -vvvv)

```
---
- hosts: localhost
  tasks:
    - name: Create directory to store configuration files
      file:
        path: /tmp/app/conf
        state: directory
```

```
ansible-playbook playbook-dirs.yml
ansible-playbook playbook-dirs.yml -v
ansible-playbook playbook-dirs.yml -vv
ansible-playbook playbook-dirs.yml -vvv
ansible-playbook playbook-dirs.yml -vvvv (connection debugging)
```

Best practices: **No Verbosity**

```
PLAY [localhost] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [Create directory to store configuration files] ***************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

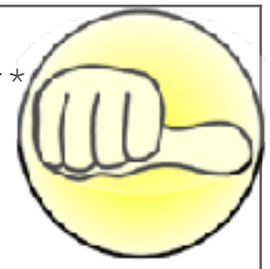
Best practices: **Verbosity -v**

```
Using /Users/mariadominguez/.ansible.cfg as config file

PLAY [localhost] ***************************************************

TASK [Gathering Facts] *********************************************
ok: [localhost]

TASK [Create directory to store configuration files] ***************
changed: [localhost] => {"changed": true, "failed": false, "gid": 0, "group":
"wheel", "mode": "0755", "owner": "mariadominguez", "path": "/tmp/app/conf", "size":
68, "state": "directory", "uid": 501}

PLAY RECAP *********************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

Best practices: **Verbosity -vv**

```
ansible-playbook 2.4.0.0
  config file = /Users/mariadominguez/.ansible.cfg
  configured module search path = [u'/Users/mariadominguez/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /Library/Python/2.7/site-packages/ansible
  executable location = /usr/local/bin/ansible-playbook
  python version = 2.7.10 (default, Feb  7 2017, 00:08:15) [GCC 4.2.1 Compatible
Apple LLVM 8.0.0 (clang-800.0.34)]
Using /Users/mariadominguez/.ansible.cfg as config file

PLAYBOOK: playbook-dirs.yml ****************************************************
1 plays in playbook-dirs.yml

PLAY [localhost] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]
META: ran handlers

TASK [Create directory to store configuration files] ***************************
task path: /Users/mariadominguez/ansible-workshop/playbook-dirs.yml:17
changed: [localhost] => {"changed": true, "failed": false, "gid": 0, "group":
"wheel", "mode": "0755", "owner": "mariadominguez", "path": "/tmp/app/conf", "size":
68, "state": "directory", "uid": 501}
META: ran handlers
META: ran handlers

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

Best practices: **Verbosity -vvv**



```
[...]
TASK [Create directory to store configuration files] ***************************************
task path: /Users/mariadominguez/ansible-workshop/playbook-dirs.yml:17
Using module file /Library/Python/2.7/site-packages/ansible/modules/files/file.py
<localhost> ESTABLISH LOCAL CONNECTION FOR USER: mariadominguez
<localhost> EXEC /bin/sh -c 'echo ~ && sleep 0'
<localhost> EXEC /bin/sh -c '( umask 77 && mkdir -p "` echo /Users/mariadominguez/.ansible/tmp/ansible-
tmp-1510245609.56-39641320619915 `" && echo ansible-tmp-1510245609.56-39641320619915="` echo /Users/
mariadominguez/.ansible/tmp/ansible-tmp-1510245609.56-39641320619915 `" ) && sleep 0'
<localhost> PUT /var/folders/qf/5jdmsd3d31j7q50p0gppn7z40000gn/T/tmpXMvC2R TO /Users/
mariadominguez/.ansible/tmp/ansible-tmp-1510245609.56-39641320619915/file.py
<localhost> EXEC /bin/sh -c 'chmod u+x /Users/mariadominguez/.ansible/tmp/ansible-
tmp-1510245609.56-39641320619915/ /Users/mariadominguez/.ansible/tmp/ansible-
tmp-1510245609.56-39641320619915/file.py && sleep 0'
<localhost> EXEC /bin/sh -c '/usr/bin/python /Users/mariadominguez/.ansible/tmp/ansible-
tmp-1510245609.56-39641320619915/file.py; rm -rf "/Users/mariadominguez/.ansible/tmp/ansible-
tmp-1510245609.56-39641320619915/" > /dev/null 2>&1 && sleep 0'
changed: [localhost] => {
    "changed": true,
    "diff": {
        "after": {
            "path": "/tmp/app/conf",
            "state": "directory"
        },
        "before": {
            "path": "/tmp/app/conf",
            "state": "absent"
        }
    },
    "failed": false,
    "gid": 0,
    "group": "wheel",
    "invocation": {
        "module_args": {
            "attributes": null,
            "backup": null,
            [...]
```