

Cloud Computing Applications and Services

(Aplicações e Serviços de Computação em Nuvem)

System Provisioning

University of Minho
2022/2023



Provisioning

Provisioning, the action of providing or supplying something for use.

- Server provisioning
- Storage provisioning
- Network provisioning
- VM provisioning
- User provisioning

Deployment

Deployment, the process of installing or upgrading an *application/service* into a server.

- Installing or upgrading a web application
 - Files
 - Migrations
 - Assets
- Network service installation or upgrade
- The scope is the **service** or **application**

Provisioning and Deployment is...

Boring (after first iteration)

- Repetitive process
 - So, it's a great target for automation
- May spread across multiple and heterogeneous systems
 - Better keep a systems inventory and run tasks sequentially or parallelized
- Will probably require tweaks overtime
 - Well, versioning might be a good idea
- Sometimes a time consuming task
 - Let the machine do it and go do something else

Configuration Management

A way of handling systematic system changes while maintaining integrity throughout its lifecycle.

- Express configuration through a common dialect
- Predictable configuration result
- Configuration evolves with the infrastructure
- Infrastructure documentation as a positive side effect
- Full history of changes overtime when used with source code management
- Changes are observable
- Process Automation
- Each unit of work is expressed as a recipe

Recipes / Reuse / Automation

Define task automation via a set of directives expressed in a given language.

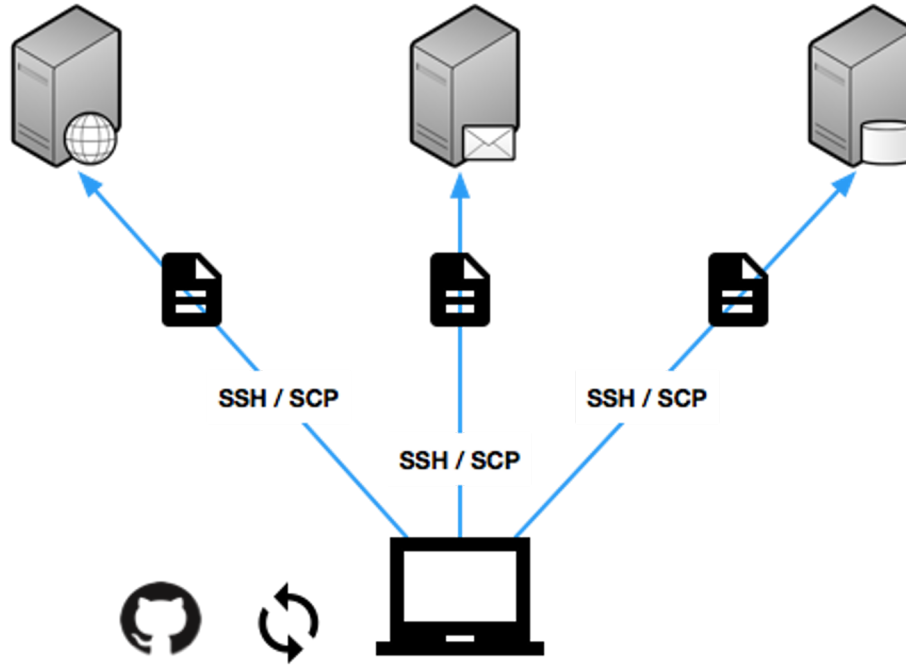
```
#!/bin/sh
username=deployer
apt-get -y update
apt-get -y upgrade
apt-get -y install vim-nox openntpd sudo whois aptitude
useradd -G sudo -p "password" -s /bin/bash -m $username
mkdir -p /home/$username/.ssh
chmod 700 /home/$username/.ssh
chown $username: /home/$username/.ssh
echo "public_key" >> /home/$username/.ssh/authorized_keys
chmod 600 /home/$username/.ssh/authorized_keys
chown $username: /home/$username/.ssh/authorized_keys
```

Tools of the Trade

	Language	Agent	Agentless	SSH
Ansible	YAML	No	Yes	Yes
Chef	Ruby	Yes	Supported	Yes
Puppet	Puppet's Dec. Lang.	Yes	Supported	Yes
SaltStack	YAML	Yes	Supported	Yes



Provisioning and Deployment Workflow



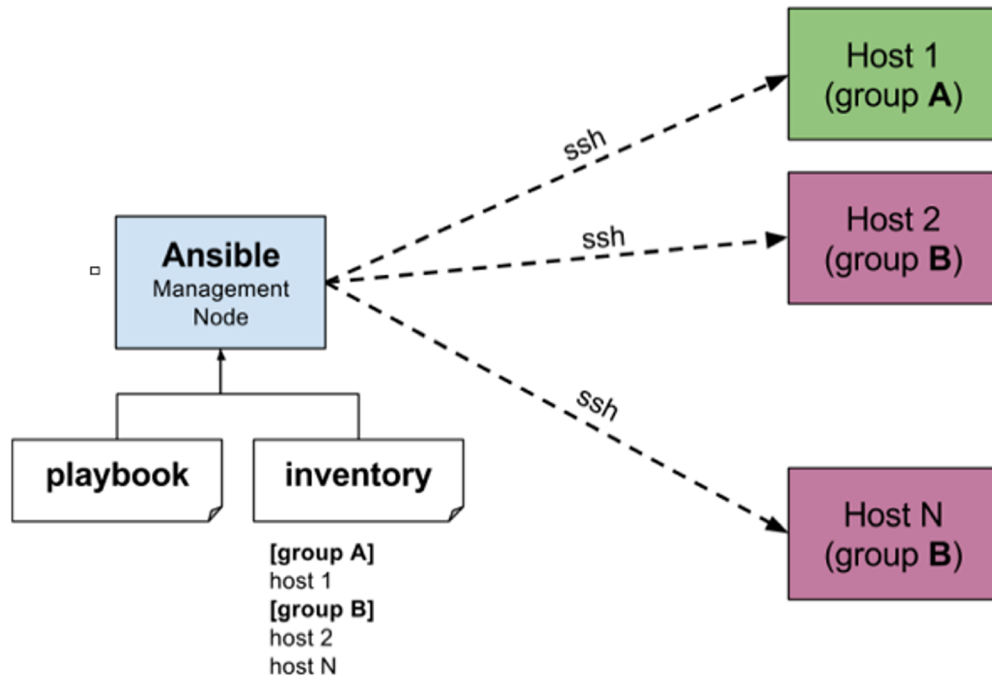


Vocabulary

- **Inventory**
 - Grouped deployment targets (hosts)
- **Module**
 - Reusable work unit distributed with Ansible or developed for it
- **Task**
 - Combination of a module and given arguments in order to create an action
- **Handlers**
 - Special kind of task that responds to a notification
- **Templates**
 - Enable the creation of dynamic configuration (leverages Jinja2, the Python template engine)
- **Role**
 - Reusable component that encapsulates variables, templates, tasks, handlers... (configurable)
- **Playbook**
 - Describe policies for remote systems to enforce (set of roles / tasks)

Overview

- Agentless recipe execution via SSH or locally
- Recipes are expressed in YAML
- Recipes are created via module and task directives
- Recipes are organized into roles and playbooks
- Target hosts are defined in the inventory
- Tasks only run if the target differs from the expected result (idempotency)



Inventory

hosts.inv:

[web]

host-[01:99].example.com

[database]

db-01.example.com

Staging.example.com

Playbook

- hosts: all

vars:

username: someuser

shell: /bin/bash

tasks:

- name: create unprivileged user

user: ←

Module

name: '{{username}}'

password: 'secretpasswordhash'

shell: '{{shell}}'

- name: Set SSH authorized_key

authorized_key:

user: '{{username}}'

state: present

key: "{{ lookup('file', '/home/' + someuser + '/.ssh/id_rsa.pub') }}"

Shell Script

```
#!/bin/sh
username=someuser
useradd -G sudo -p "password" -s /bin/bash -m $username
mkdir -p /home/$username/.ssh
chmod 700 /home/$username/.ssh
chown $username: /home/$username/.ssh
echo "public_key" >> /home/$username/.ssh/authorized_keys
chmod 600 /home/$username/.ssh/authorized_keys
chown $username: /home/$username/.ssh/authorized_keys
```

Role

hosts.inv

provision.yml

roles

- |
- | - role
 - | - files (static files)
 - | - templates (Jinja2 templates)
 - | - **tasks** (task definition - main.yml)
 - | - handlers (handlers that trigger on notify - main.yml)
 - | - vars (role scoped variables - main.yml)
 - | ... (defaults, meta, etc)

See examples at: <https://github.com/ansible/ansible-examples!>

Playbook With Roles

- hosts: webserver

- roles:

- config

- users

- ssh-server

- ntp-client

- ...

Execution

Without ansible.cfg:

```
ansible-playbook playbook.yml -b -i hosts.inv -u someuser -K --private-key=/path/to/private_key
```


With ansible.cfg (ansible.cfg):

```
ansible-playbook playbook.yml -b -K
```

Flags:

- b -> become, privilege elevation with *sudo* command
- i -> inventory file to use
- u -> login username
- K -> ask *sudo* password

ansible.cfg:



```
[defaults]  
hostfile = hosts.inv  
remote_user = someusername  
private_key_file = ../../someuser_private_key
```

Level Up

Handlers

- name: template configuration file

template:

src: template.j2

dest: /etc/nginx/nginx.conf

notify:

- restart nginx

handlers:

- name: restart nginx

service:

name: nginx

state: restarted

Conditionals

- name: Common Debian
include_tasks: debian.yml
when: ansible_os_family == "Debian"
 - name: Common RedHat
include_tasks: rh.yml
when: ansible_os_family == "RedHat"
-

when:

- condition
- condition

when: condition or condition

when: somevar|failed

when: somevar is defined

Loops

- name: Install Packages
apt: name="{{ item }}" update_cache=yes state=latest
loop:
 - vim-nox
 - aptitude
- name: Install Packages
apt: name="{{ item }}" update_cache=yes state=latest
loop: "{{ list_variable_with_packages }}"
- name: Copy Files
copy: src="{{ item.source }}" dest="{{ item.destination }}"
loop:
 - { source: 'motd', destination: '/etc/motd' }
 - { source: 'sshd', destination: '/etc/ssh/sshd_config' }

More at https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html

Host Facts

Facts can be accessed and used within tasks and templates allowing for more dynamic playbooks.

```
{% for host in groups['webservers'] %}  
    {{ hostvars[host]['ansible_all_ipv4_addresses'] | join }}  
{% endfor %}
```

```
{{ ansible_distribution }}  
{{ ansible_os_family }}  
{{ ansible_processor_vcpus }}
```

Vault

- Allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles.
- More at https://docs.ansible.com/ansible/2.6/user_guide/vault.html

Provisioning GCP With Ansible

- Ansible can be used to provision different GCP services
 - VMs, Kubernetes, ...
- Ansible Modules (examples...)
 - [gcp_compute_disk](#)
 - [gcp_compute_network](#)
 - [gcp_compute_address](#)
 - [gcp_compute_firewall](#)
 - [gcp_compute_instance](#)
 - [gcp_container_cluster](#)
- https://docs.ansible.com/ansible/latest/scenario_guides/guide_gce.html

Dynamic Inventory

- When provision happens dynamically the addresses are unknown
- Problem has two solutions
 - Manually (go into the console and look at the addresses)
 - Automatic (use a dynamic inventory)

Further reading

- <https://docs.ansible.com>
- K. Morris. **Infrastructure as Code: Managing Servers in the Cloud**. O'Reilly, 2016

Questions?