Ansible & TOSCA Essentials

Doina Cristina Duma (aiftim<at>infn.it)

Alessandro Costantini (acostantini<at>infn.it)

Big Data Analytics

9-12 Dic. 2019, Bologna

Outline

- What is, how it works, architecture
- Key components
- Ad-hoc commands
- Roles, their structure
- Ansible-Galaxy & Galaxy, Roles use and re-use
- Playbooks & roles
- Advanced usage (cenni): debug, optimization

Bit…s of History

- «**Ansible**»
  - 1966 – **Ursula K. Le Guin**, *«Rocannon's World»*
    - *«**answerable**»: device that allow its users to **receive answers** to their **messages** in a **reasonable** amount of **time**, even over **interstellar distances***
  - 1977,1985 – **Orson Scott Card**, *«Ender's Game»*
    - *«**Philotic Parallax Instantaneous Communicator**»: machine capable of **communicating** across **infinite distances** with **no time delay***
  - **2012** - **Michael DeHaan** **,** RH Emerging Technologies: ***«work on basically whatever they thought people needed»***
    - Cobbler & Func
    - AnsibleWorks, Inc. => Ansible, Inc. => RedHat (2015)
    - ***«a simple deployment, model-driven configuration management, and command execution framework»***

ANSIBLE

What is?

*«Ansible is an **automation** and **configuration** management technology used to **provision**, **deploy**, and **manage** compute infrastructure across **cloud**, **virtual**, and **physical** environments»*

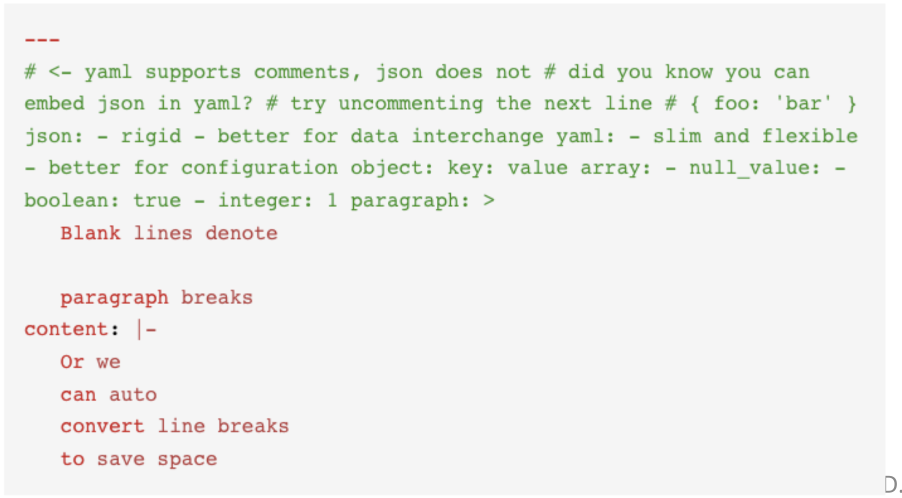
- ➢Automation language that can describe an IT application infrastructure, in Ansible Playbooks => YAML

- ➢Automation Engine that runs Ansible Playbooks

(YAML = YAML Ain't Markup Language)

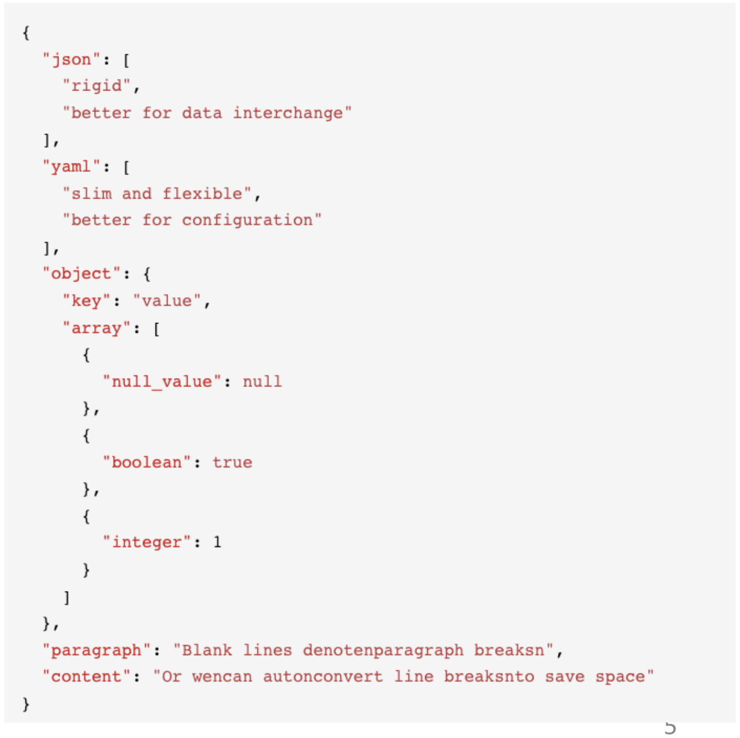
- **Human friendly (readble) data-serialization standard for all programming languages**
- Can be used with nearly any application that needs to **store** or **transmit** data
- **Flexible** = bits and pieces from **other languages**:
  - Scalars, lists, associative arays <- **Perl**
  - Document separator, «—» <- **MIME**
  - Whitespace wrapping <- **HTML**
  - Escape sequences <- **C**
  - uses both **Python**-style indentation to indicate nesting

YAML version



```
---
# <- yaml supports comments, json does not # did you know you can
embed json in yaml? # try uncommenting the next line # { foo: 'bar' }
json: - rigid - better for data interchange yaml: - slim and flexible
- better for configuration object: key: value array: - null_value: -
boolean: true - integer: 1 paragraph: >
   Blank lines denote

   paragraph breaks
content: |-
   Or we
   can auto
   convert line breaks
   to save space
```

JSON version



```
{
  "json": [
    "rigid",
    "better for data interchange"
  ],
  "yaml": [
    "slim and flexible",
    "better for configuration"
  ],
  "object": {
    "key": "value",
    "array": [
      {
        "null_value": null
      },
      {
        "boolean": true
      },
      {
        "integer": 1
      }
    ]
  },
  "paragraph": "Blank lines denotenparagraph breaksn",
  "content": "Or wencan autonconvert line breaksnto save space"
}
```

Ansible is …

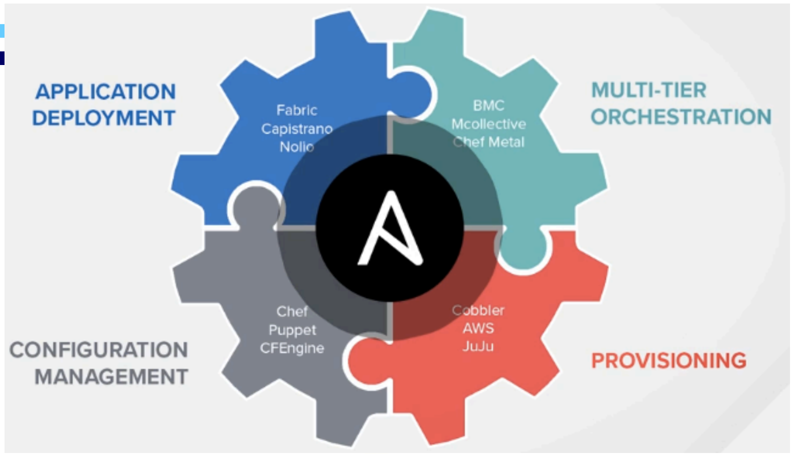
- **Simple**
  - ➢ Human readable automation
  - ➢ No special coding skills needed
  - ➢ Tasks executed in order
  - ➢ Get productive quickly
- **Powerful**
  - ➢ Application deployment
  - ➢ Configuration management
  - ➢ Workflow orchestration
  - ➢ Orchestrate the application lifecycle
- **Cross-platform**
  - ➢ Agentless support for all major OS, physical, virtual, cloud and network

- **Works with existing toolkits**
  - ➢ Homogenize existing env. by leveraging current toolsets and update mechanisms
- **«Batteries Included»**
  - ➢ Comes bundled with > 450 modules

- Cloud	- Monitoring	- System
- Containers	- Network	- Testing
- Databases	- Notifications	- Utilities
- Files	- Packaging	- Web Infrastructure
- Messaging	- Source Control	

- **Community powered**
  - ➢ the **most popular open source automation** tool on **GitHub**
    - ▪ Downloads ~250k/month
    - ▪ People – 3500 people contributing modules, 1200 users on IRC

Use cases



CONFIG MANAGEMENT

Enforce a baseline state for all of our servers
- right packages the right configuration files

APP DEPLOYMENT

«Take code and push it to servers»
- deploy multi-tier applications reliably and consistently, all from one common framework

PROVISIONING

Latest cloud platforms, virtualized hosts and hypervisors, network devices and bare-metal servers

CONTINUOUS DELIVERY

Push-based architecture allows very fine-grained control over operations

SECURITY & COMPLIANCE

Apply and enforce security standards that adapt to meet internal and external security guidelines

ORCHESTRATION

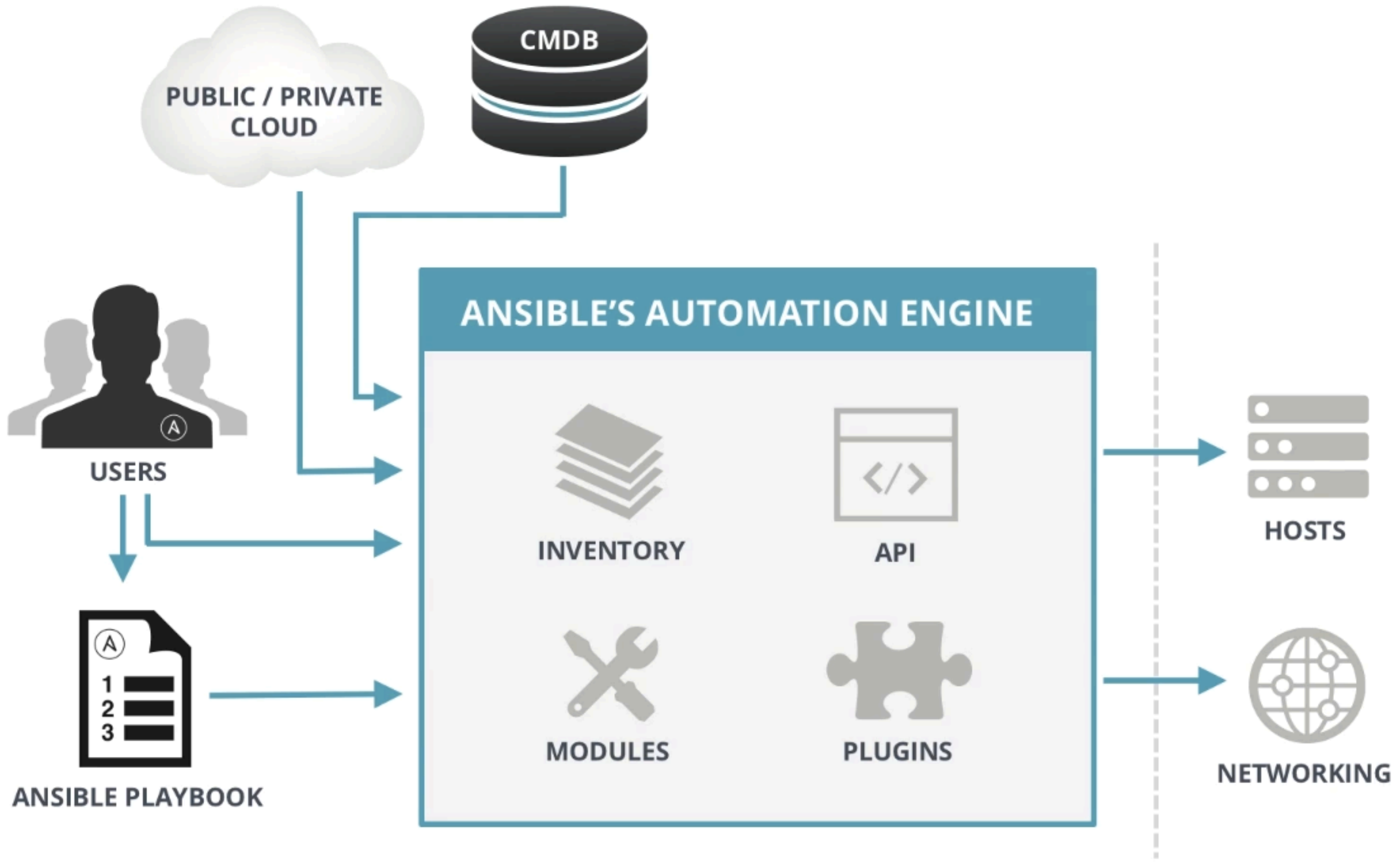
«Bring order into caos»
- targeting the right servers in the right order at any time

Ansible concepts

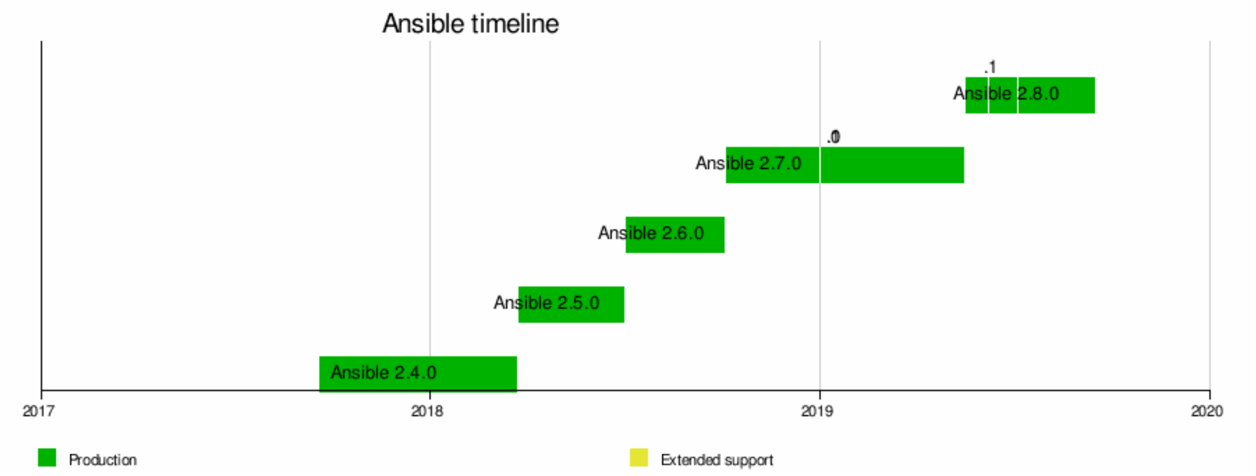
- ***Control Node***
  - ➢ Any machine with Ansible installed
- ***Managed Nodes*** = hosts
  - ➢ Servers one manages **with** Ansible
  - ➢ No Ansible installed
- ***Inventory*** = hostfile
  - ➢ List of managed hosts
  - ➢ Groups – hosts with common features (web server, rack)
- ***Modules***
  - ➢ units of code Ansible executes
- ***Tasks***
  - ➢ units of action in Ansible
- ***Playbook***
  - ➢ Ordered lists of tasks, and variables
  - ➢ Written in YAML

- **`Playbook`** is a YAML file which consists in a list of Plays.
  - ➢ A **`Play`** in a playbook is a list of Tasks.
    - ➢ A **`Task`** in a play contains Modules and its arguments.
      - ➢ **`Modules`** are the ones that do the actual work.

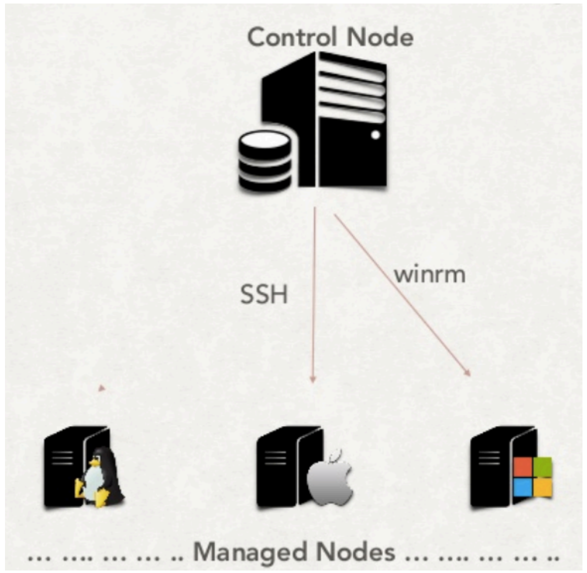
Ansible Architecture



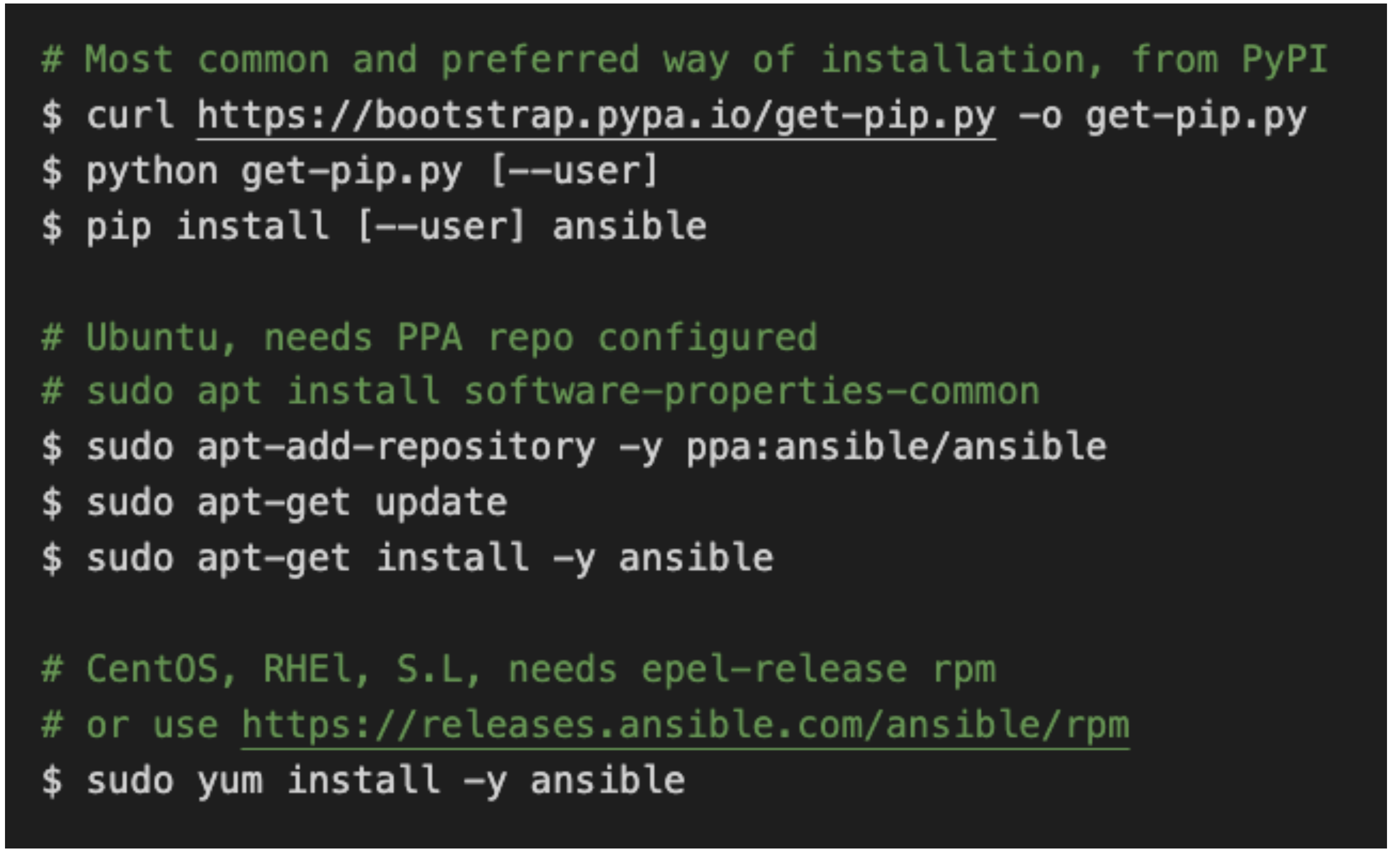
Installation



- Version:
  - ➢latest
- Requirements:
  - ➢***Control Node***
    - **Python** 2 (v. 2.7) or Python 3 (v. 3.5 and higher)
    - Red Hat, Debian, CentOS, macOS, any of the BSDs, etc
      - ❖ **No Windows**
    - Nearness/closeness
  - ***Managed Nodes***
    - **Python** 2 (v. 2.7) or Python 3 (v. 3.5 and higher)
    - a way to communicate => **ssh**

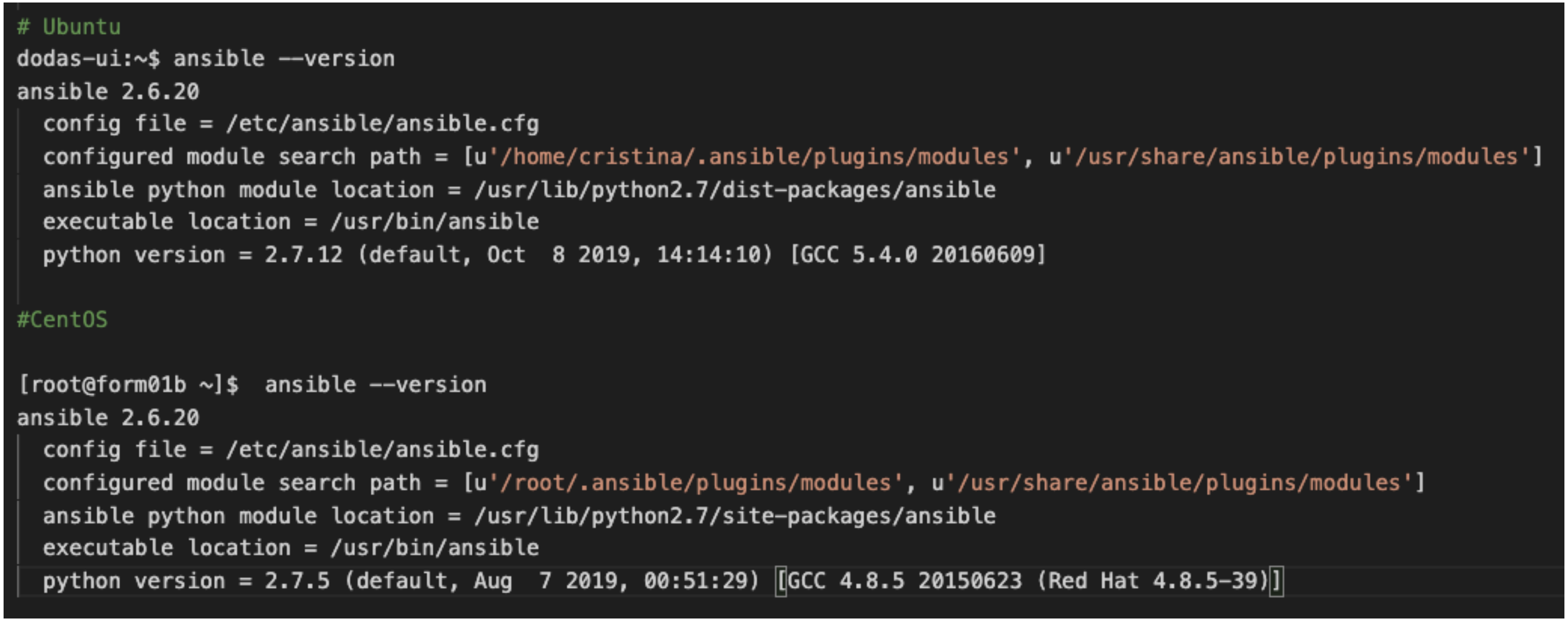


Installation (2)



```
# Most common and preferred way of installation, from PyPI
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
$ python get-pip.py [--user]
$ pip install [--user] ansible

# Ubuntu, needs PPA repo configured
# sudo apt install software-properties-common
$ sudo apt-add-repository -y ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install -y ansible

# CentOS, RHEl, S.L, needs epel-release rpm
# or use https://releases.ansible.com/ansible/rpm
$ sudo yum install -y ansible
```

Version, config files

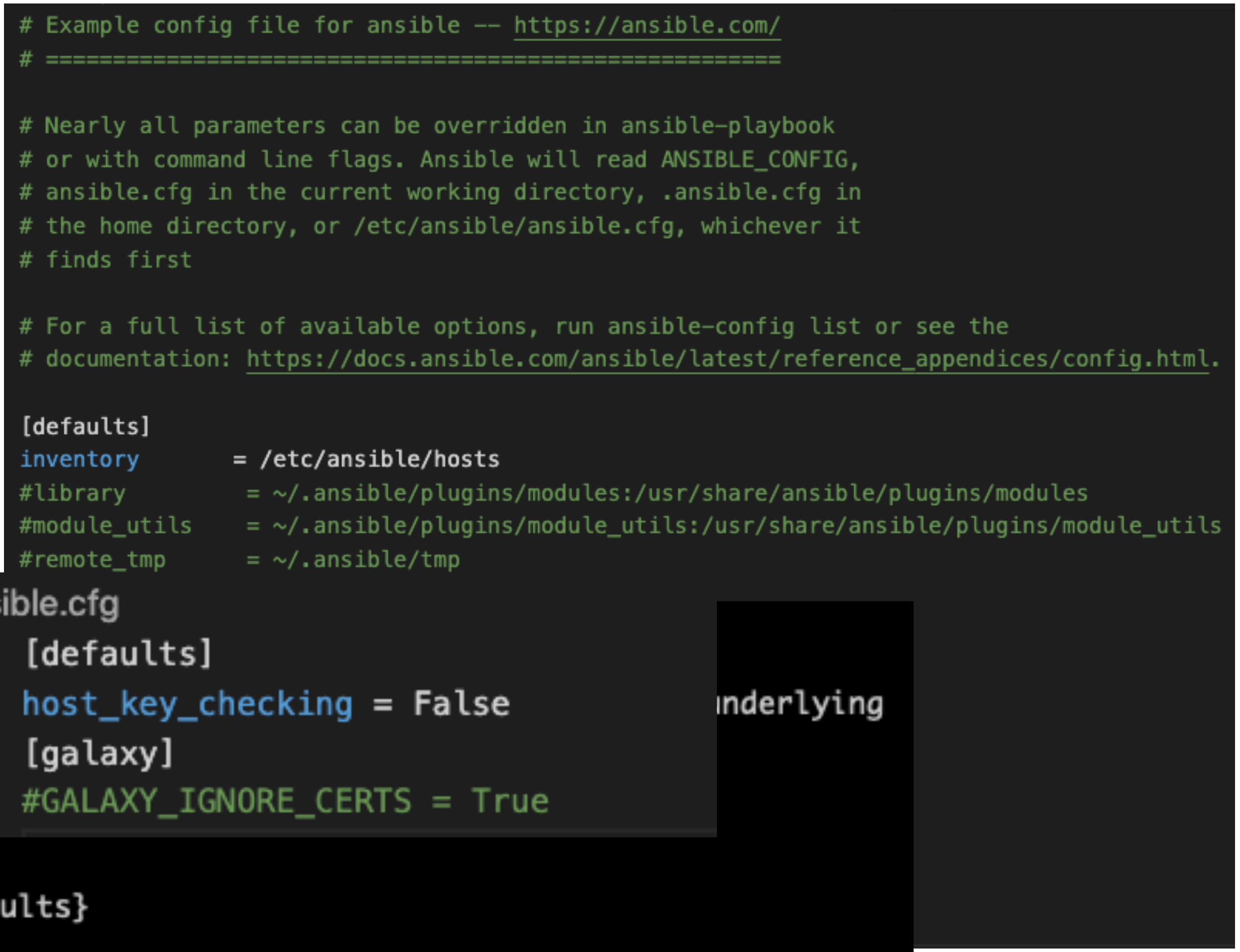


```
# Ubuntu
dodas-ui:~$ ansible --version
ansible 2.6.20
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/cristina/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.12 (default, Oct  8 2019, 14:14:10) [GCC 5.4.0 20160609]

#CentOS

[root@form01b ~]$  ansible --version
ansible 2.6.20
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Aug  7 2019, 00:51:29) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]
```

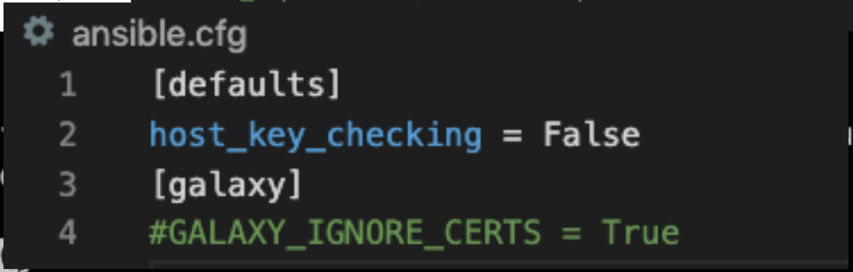
Config files

- $ANSIBLE_CONFIG
- {$PWD}/ansible.cfg
- ~/.ansible.cfg
- /etc/ansible/ansible.cfg

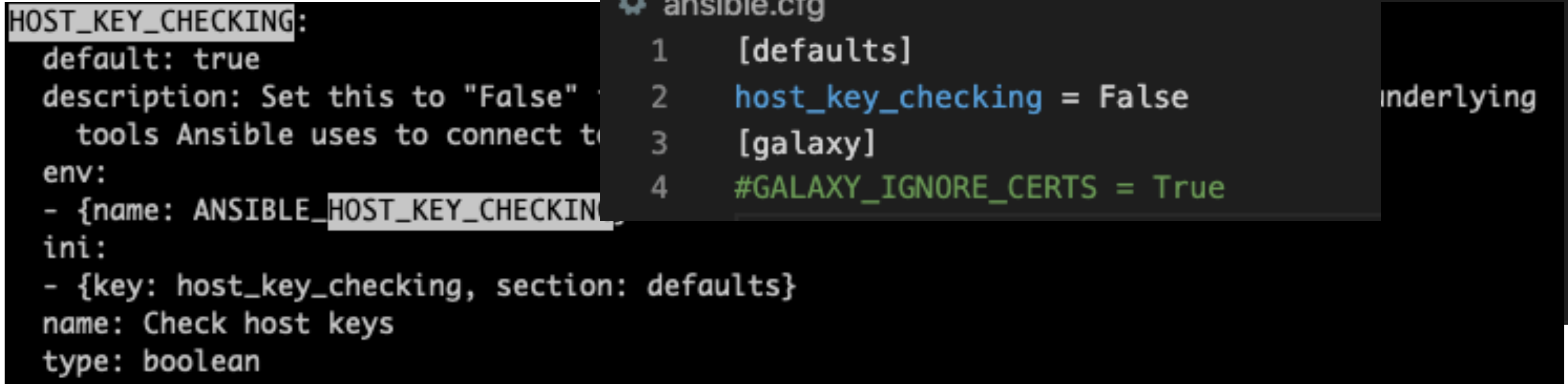
`$ ansible-config list`



```
# Example config file for ansible -- https://ansible.com/
# ===============================================

# Nearly all parameters can be overridden in ansible-playbook
# or with command line flags. Ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory, or /etc/ansible/ansible.cfg, whichever it
# finds first

# For a full list of available options, run ansible-config list or see the
# documentation: https://docs.ansible.com/ansible/latest/reference_appendices/config.html.

[defaults]
inventory       = /etc/ansible/hosts
#library         = ~/.ansible/plugins/modules:/usr/share/ansible/plugins/modules
#module_utils    = ~/.ansible/plugins/module_utils:/usr/share/ansible/plugins/module_utils
#remote_tmp      = ~/.ansible/tmp
```



```
ansible.cfg
[defaults]
host_key_checking = False
[galaxy]
#GALAXY_IGNORE_CERTS = True
```

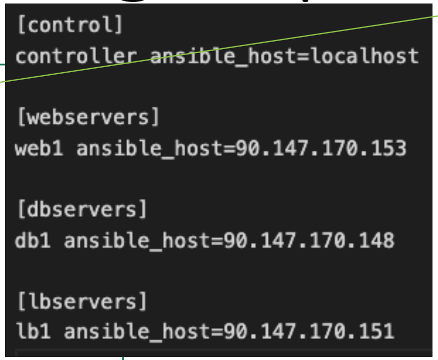


```
HOST_KEY_CHECKING:
  default: true
  description: Set this to "False"                            underlying
    tools Ansible uses to connect t
  env:
  - {name: ANSIBLE_HOST_KEY_CHECKIN
  ini:
  - {key: host_key_checking, section: defaults}
  name: Check host keys
  type: boolean
```

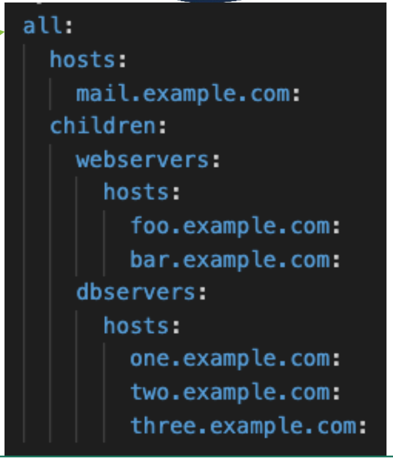
CLI

- ansible - Define and run a single task ‘playbook’ against a set of hosts
- ansible-config - View ansible configuration
- ansible-console - REPL console for executing Ansible tasks
- ansible-doc - Plugin documentation tool
- ansible-galaxy - Perform various Role and Collection related operations
- ansible-inventory - Display or dump the configured inventory as Ansible sees it
- ansible-playbook-Runs Ansible playbooks, executing the defined tasks on the targeted hosts.
- ansible-pull - pulls playbooks from a VCS repo and executes them for the local host
- ansible-vault - encryption/decryption utility for Ansible data files

Inventory: formats, hosts, groups



```
[control]
controller ansible_host=localhost

[webservers]
web1 ansible_host=90.147.170.153

[dbservers]
db1 ansible_host=90.147.170.148

[lbservers]
lb1 ansible_host=90.147.170.151
```



```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

- ***Formats***:
  - INI
  - YAML
- ***Hosts***
  - Remote nodes managed by Ansible
  - Can have individual **variables** (host name, service port number, etc, see ex…)
  - **Ranges:**
    - www[01:50].example.com
    - db-[a:f].example.com
  - **Vars:**

```
[group1]
host1 http_port=80
maxRequestsPerChild=808
host2 http_port=303
maxRequestsPerChild=909
```

- ***Groups***
  - Used to clasify hosts, hosts in multiple groups
    - **what** hosts you are controlling at **what** times and for **what** purpose.
  - Default groups:
    - ***«all»***, ***«ungrouped»***

Inventory: formats, hosts, groups, **vars**

- Vars:
  - Host vars
  - Group vars
    - Assigning a variable to **many** machines
    - Ansible **flattens** vars at level of host

```
[atlanta]
host1
host2

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

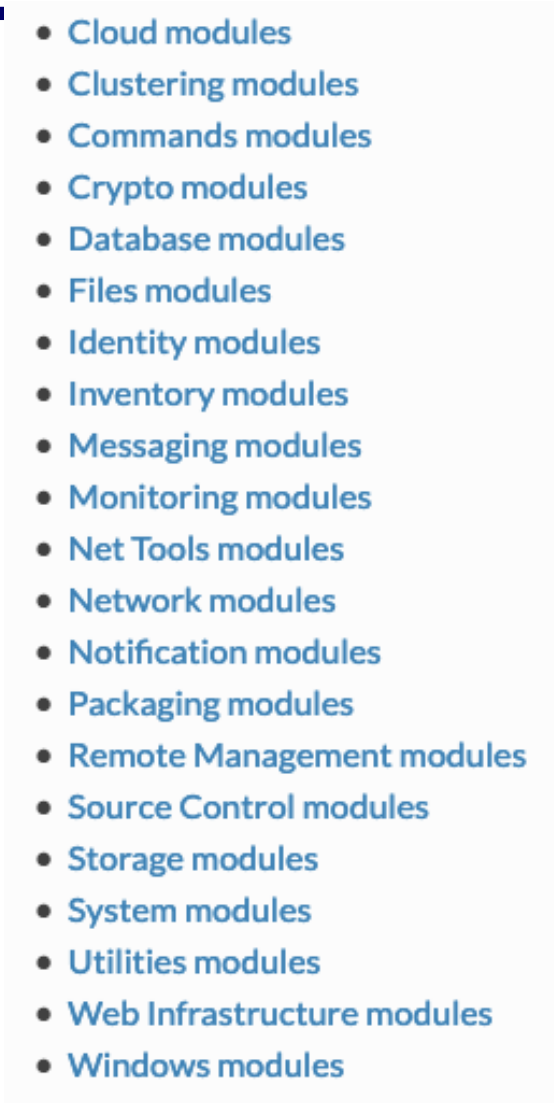
- ➢ internal rules for merging => order/precedence:
  - ➢ all group
  - ➢ parent group
  - ➢ child group
  - ➢ Host
- ➢ When multiple inventory are used => their order is important

```
a_group:
    testvar: a
    ansible_group_priority: 10
b_group:
    testvar: b
```

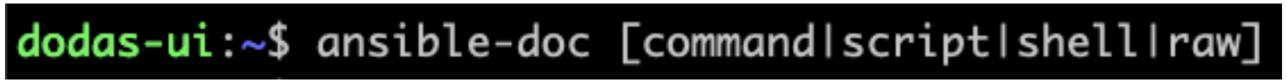
```
ansible-playbook get_logs.yml -i staging -i production
```

Modules & Run Commands

- ***Modules*** = units of code executed by Ansible
  = **«Ansible toolbox»**
  - Written in Python
  - Extensive library:
    - [Web module index]
    - **`# ansible-doc –l`**
  - ***(run)-commands => Ad-hoc commands***
    - ***command***
      - Exec commands on targets
    - ***shell***
      - Exec shell commands on targets
    - ***script***
      - Runs a local script on a remote node after transferring it
    - ***raw***
      - Exec a command without going throughthe Ansible module subsystem

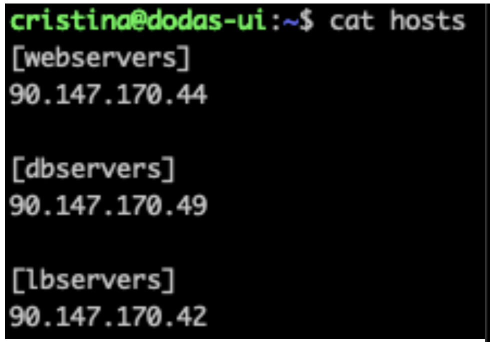


- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules
- Windows modules

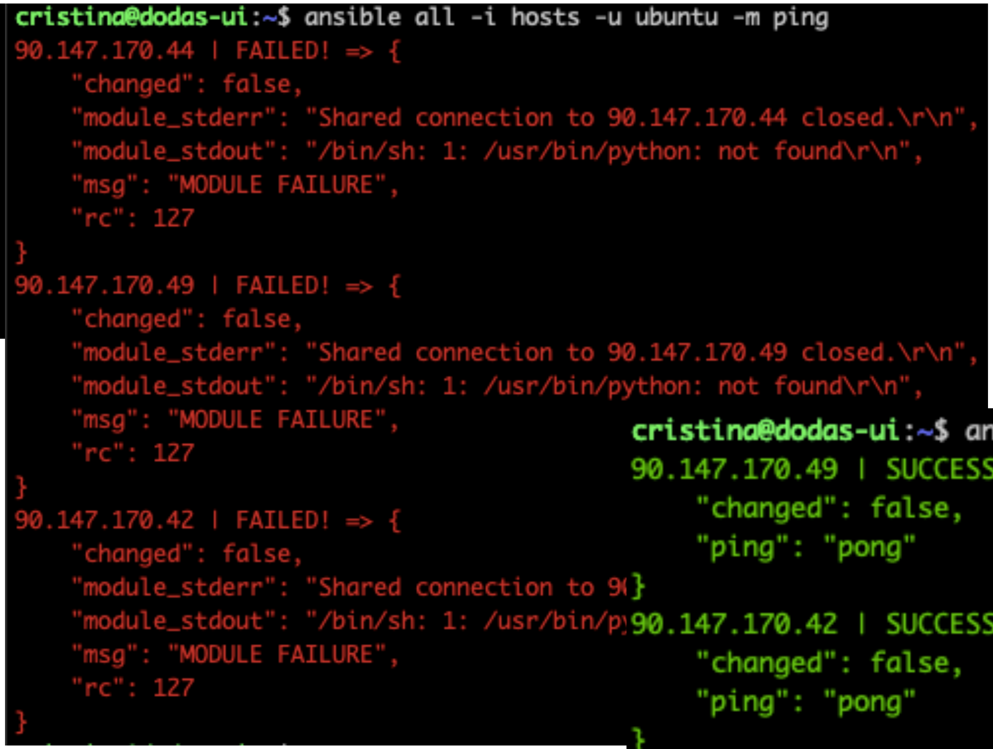


```
dodas-ui:~$ ansible-doc [command|script|shell|raw]
```

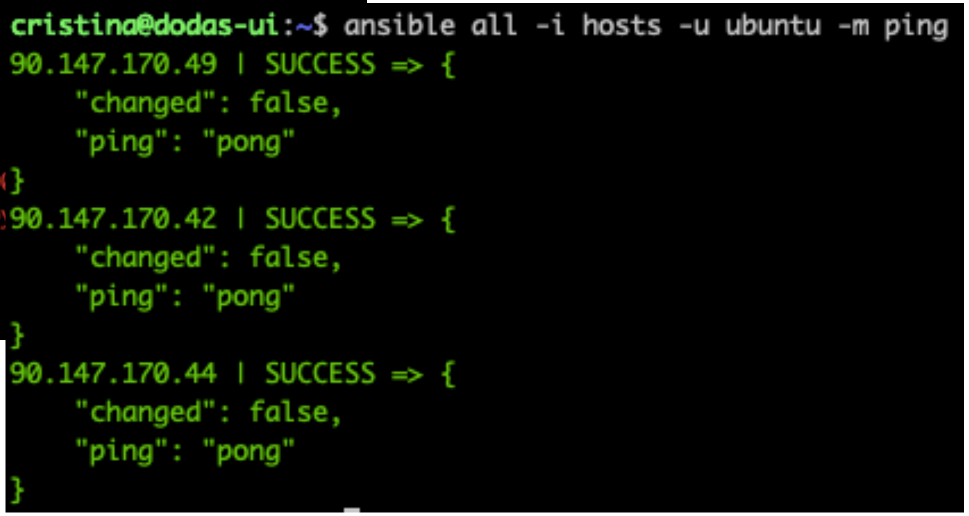
Ad-hoc Commands & Discovered Facts



```
cristina@dodas-ui:~$ cat hosts
[webservers]
90.147.170.44

[dbservers]
90.147.170.49

[lbservers]
90.147.170.42
```

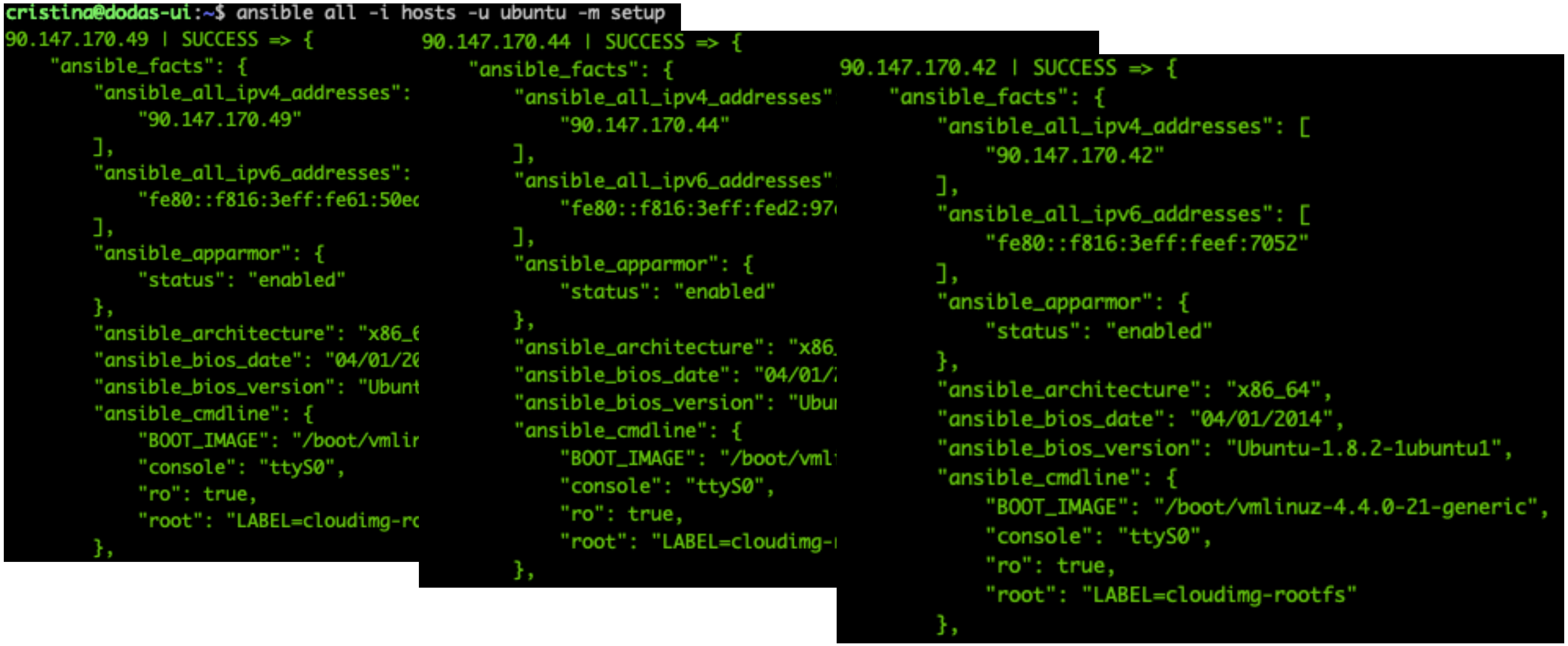


```
cristina@dodas-ui:~$ ansible all -i hosts -u ubuntu -m ping
90.147.170.44 | FAILED! => {
    "changed": false,
    "module_stderr": "Shared connection to 90.147.170.44 closed.\r\n",
    "module_stdout": "/bin/sh: 1: /usr/bin/python: not found\r\n",
    "msg": "MODULE FAILURE",
    "rc": 127
}
90.147.170.49 | FAILED! => {
    "changed": false,
    "module_stderr": "Shared connection to 90.147.170.49 closed.\r\n",
    "module_stdout": "/bin/sh: 1: /usr/bin/python: not found\r\n",
    "msg": "MODULE FAILURE",
    "rc": 127
}
90.147.170.42 | FAILED! => {
    "changed": false,
    "module_stderr": "Shared connection to 9
    "module_stdout": "/bin/sh: 1: /usr/bin/p
    "msg": "MODULE FAILURE",
    "rc": 127
}
```



```
cristina@dodas-ui:~$ ansible all -i hosts -u ubuntu -m ping
90.147.170.49 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
90.147.170.42 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
90.147.170.44 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Ad-hoc Commands & Discovered Facts (2)

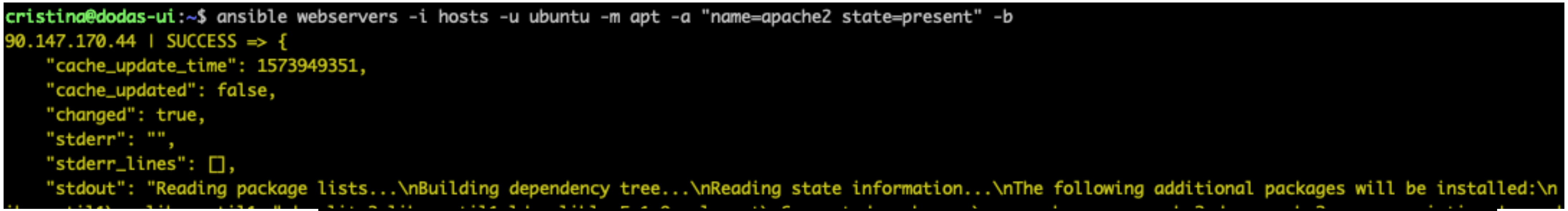


```
cristina@dodas-ui:~$ ansible all -i hosts -u ubuntu -m setup
90.147.170.49 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses":
            "90.147.170.49"
        ],
        "ansible_all_ipv6_addresses":
            "fe80::f816:3eff:fe61:50ec
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_6
        "ansible_bios_date": "04/01/20
        "ansible_bios_version": "Ubunt
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlir
            "console": "ttyS0",
            "ro": true,
            "root": "LABEL=cloudimg-rc
        },
```

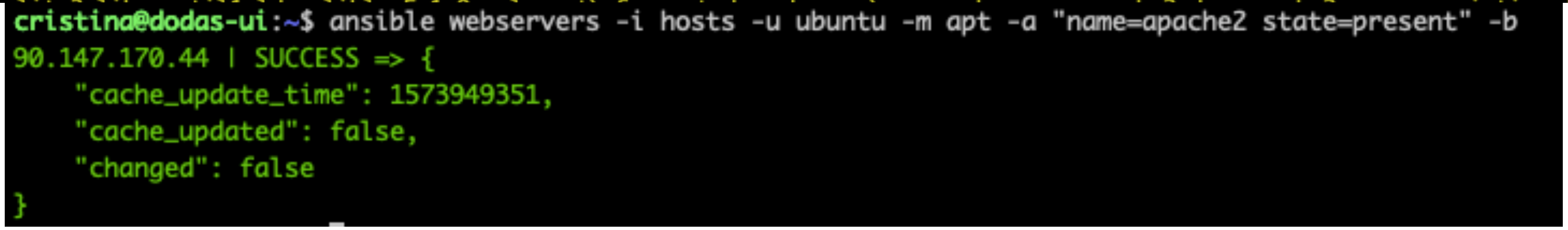
```
90.147.170.44 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses"
            "90.147.170.44"
        ],
        "ansible_all_ipv6_addresses"
            "fe80::f816:3eff:fed2:97
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86
        "ansible_bios_date": "04/01/
        "ansible_bios_version": "Ubu
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vml
            "console": "ttyS0",
            "ro": true,
            "root": "LABEL=cloudimg-
        },
```

```
90.147.170.42 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "90.147.170.42"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::f816:3eff:feef:7052"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "04/01/2014",
        "ansible_bios_version": "Ubuntu-1.8.2-1ubuntu1",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-4.4.0-21-generic",
            "console": "ttyS0",
            "ro": true,
            "root": "LABEL=cloudimg-rootfs"
        },
```

Ad-hoc Commands & Discovered Facts (3)



```
cristina@dodas-ui:~$ ansible webservers -i hosts -u ubuntu -m apt -a "name=apache2 state=present" -b
90.147.170.44 | SUCCESS => {
    "cache_update_time": 1573949351,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following additional packages will be installed:\n
```



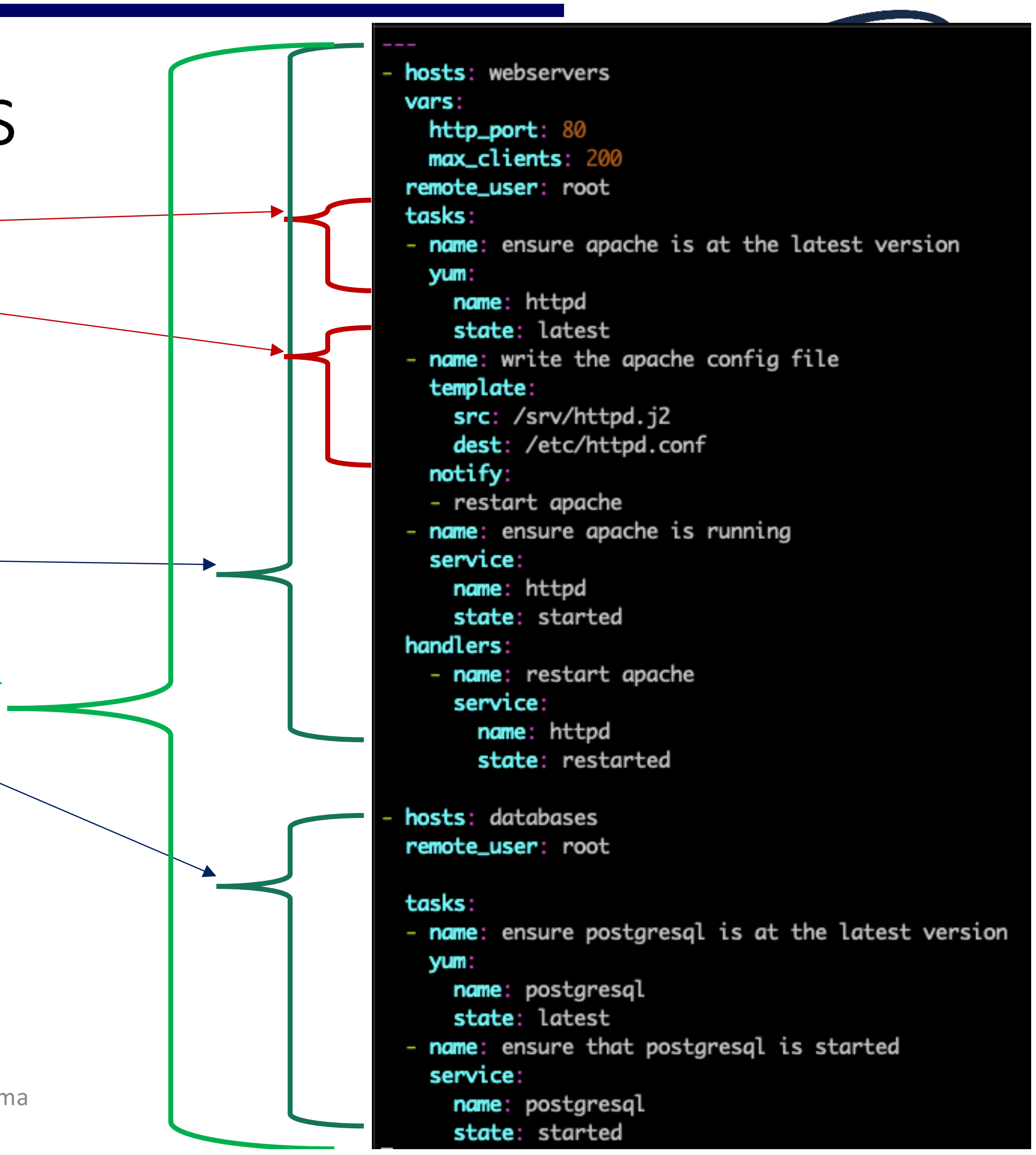
```
cristina@dodas-ui:~$ ansible webservers -i hosts -u ubuntu -m apt -a "name=apache2 state=present" -b
90.147.170.44 | SUCCESS => {
    "cache_update_time": 1573949351,
    "cache_updated": false,
    "changed": false
}
```

Playbooks, plays, tasks

- ***Task***
  - Application of a single module on one or more hosts
  - Each task ends in a well-defined state

- ***Play***
  - A set of ordered tasks, associated with a group of hosts

- ***Playbook***
  - Associate the hosts with the desired state of the infrastructure, defining the set of tasks to be performed
  - They therefore allow orchestration and deployment
  - Collection of plays



```yaml
---
- 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
    service:
      name: httpd
      state: started
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted

- 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
```

Creating Reusable Playbooks - Roles

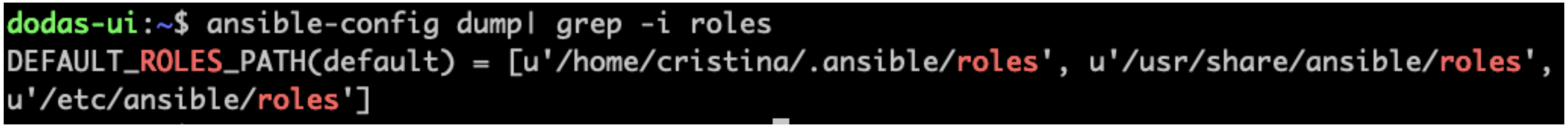
- Roles
  - decompose **complex jobs** into **smaller** pieces
    - organizing **multiple**, **related** Tasks and **encapsulating data** needed to accomplish those Tasks
      - Variables, handlers, modules, plugins
  - **special kind of Playbooks**, fully **self-contained**, with tasks, variables, configuration templates, other supporting files
    - **cannot** be executed
  - provide a **skeleton** for an **independent** and **reusable collection** of variables, tasks, templates, files, and modules which can be **automatically loaded** into the playbook.
    - Playbooks are a **collection of roles**
    - Every role has **specific functionality**

Roles vs. Playbooks

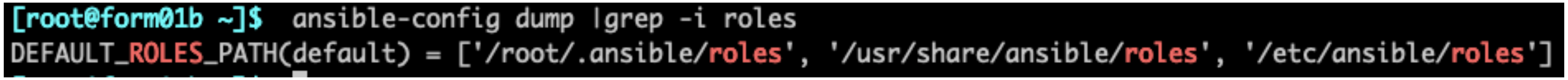
- Each role is typically **limited to a particular theme** or desired end result, with all the necessary **steps to reach that result** either within the role itself or in other roles listed as dependencies.
- Roles themselves are **not playbooks**. There is no way to directly execute a role.
- Roles have **no setting for which host** the role will apply to.
- Top-level playbooks are the **glue** that binds the hosts from your inventory to roles that should be applied to those hosts

Roles - Location

- ***Location***:
  - ➢Search path
    - A ***roles***/ directory, relative to the playbook file.
    - By default, in ***/etc/ansible/roles***
  - ➢Defined in the configuration, can be customized

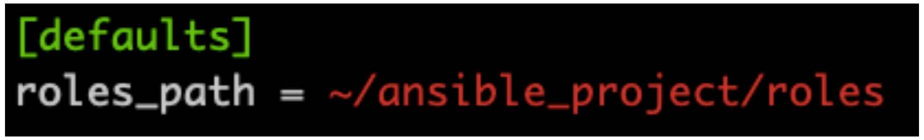


```
dodas-ui:~$ ansible-config dump| grep -i roles
DEFAULT_ROLES_PATH(default) = [u'/home/cristina/.ansible/roles', u'/usr/share/ansible/roles',
u'/etc/ansible/roles']
```



```
[root@form01b ~]$  ansible-config dump |grep -i roles
DEFAULT_ROLES_PATH(default) = ['/root/.ansible/roles', '/usr/share/ansible/roles', '/etc/ansible/roles']
```

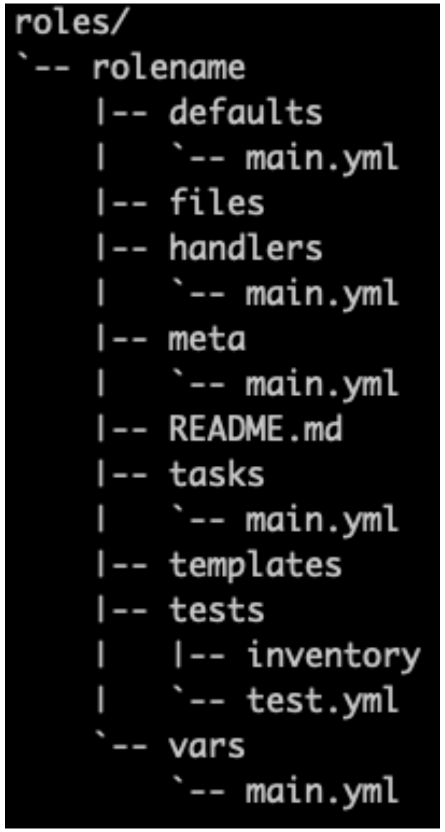
  - ➢Best-practice => define it (*ansible.cfg*) in a ***«project»*** related directory



```
[defaults]
roles_path = ~/ansible_project/roles
```

Roles - Directory Structure

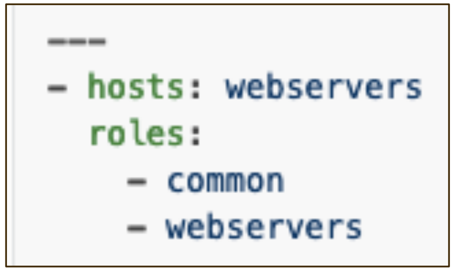
- Expect files to be in certain **directory names**
  - **At least one** of the listed directories
  - When exists – mut contain «***main.yml***»
- Content:
  - ***tasks*** - main list of tasks to be executed by the role.
  - ***handlers*** - handlers, which may be used by this role or even anywhere outside this role.
  - ***defaults*** - default variables for the role (see Using Variables for more information).
  - ***vars*** - other variables for the role
  - ***files*** - contains files which can be deployed via this role
  - ***templates*** - templates which can be deployed
  - ***meta*** - defines some meta data for this role.



```
roles/
`-- rolename
    |-- defaults
    |   `-- main.yml
    |-- files
    |-- handlers
    |   `-- main.yml
    |-- meta
    |   `-- main.yml
    |-- README.md
    |-- tasks
    |   `-- main.yml
    |-- templates
    |-- tests
    |   |-- inventory
    |   `-- test.yml
    `-- vars
        `-- main.yml
```

Roles – how to use

- Classic/original - via the *roles:* option for a given play

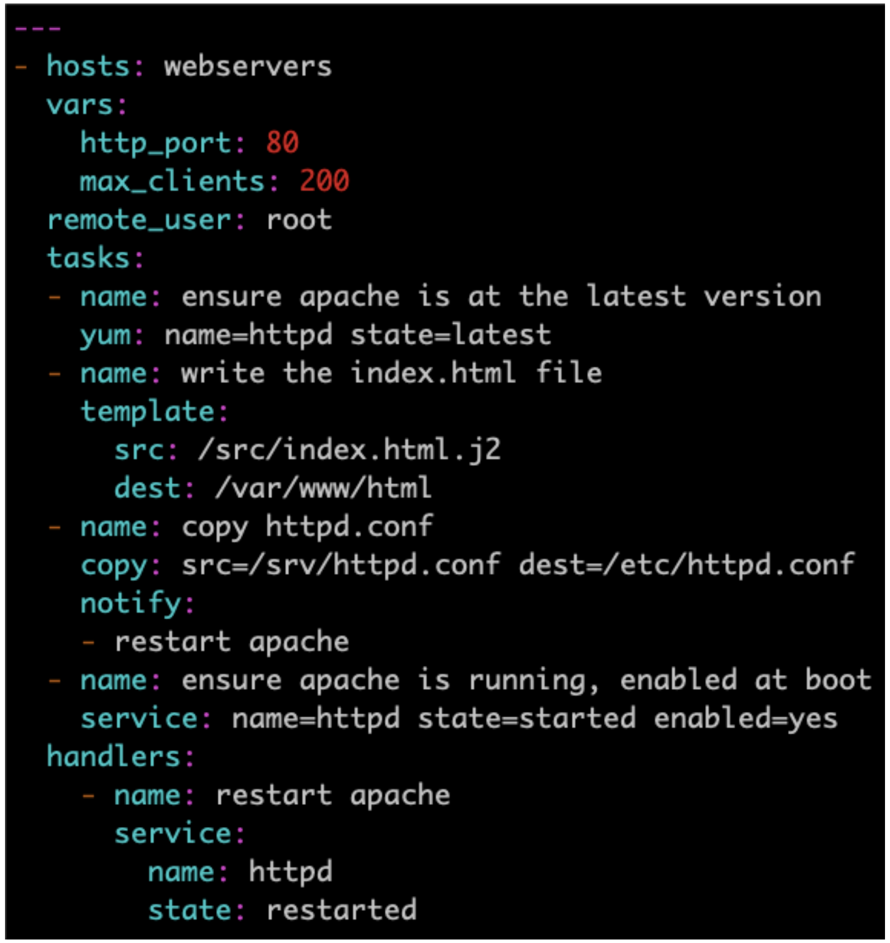


```
---
- hosts: webservers
  roles:
    - common
    - webservers
```

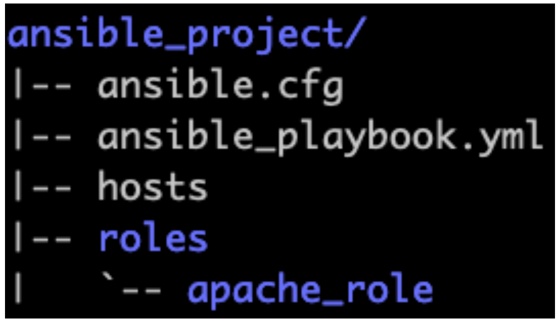
- Order to **add in the play/playbook**:
  - roles/x/***tasks/main.yml***
  - roles/x/***handlers/main.yml***
  - roles/x/***vars/main.yml***
  - roles/x/***meta/main.yml***
  - Any copy, script, template can reference files in roles/x***/{files,templates,tasks}/***

- Order of **execution** of the playbook
  - ➢ ***Each role*** listed in *roles*
    - ➢ ***Any role dependencies*** defined in the *meta/main.yml*
  - ➢ ***Any tasks*** defined in the play.
  - ➢ ***Any handlers*** triggered so far will be run.

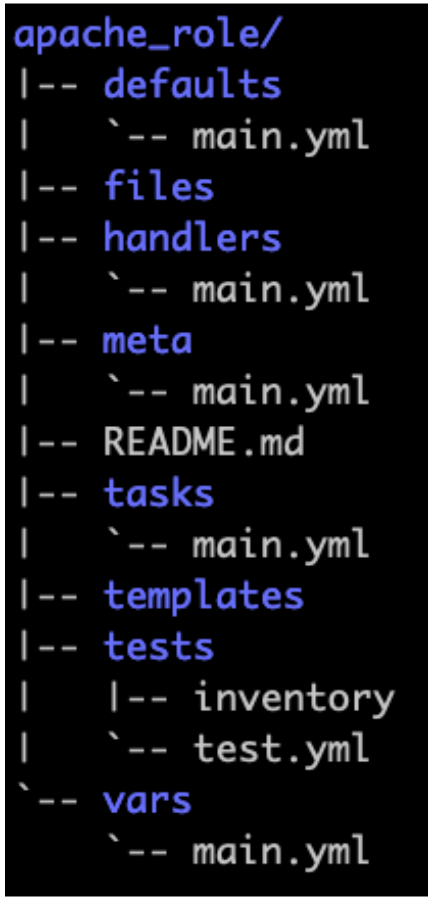
From monolithic playbook to roles



```yaml
---
- 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 index.html file
    template:
      src: /src/index.html.j2
      dest: /var/www/html
  - name: copy httpd.conf
    copy: src=/srv/httpd.conf dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running, enabled at boot
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

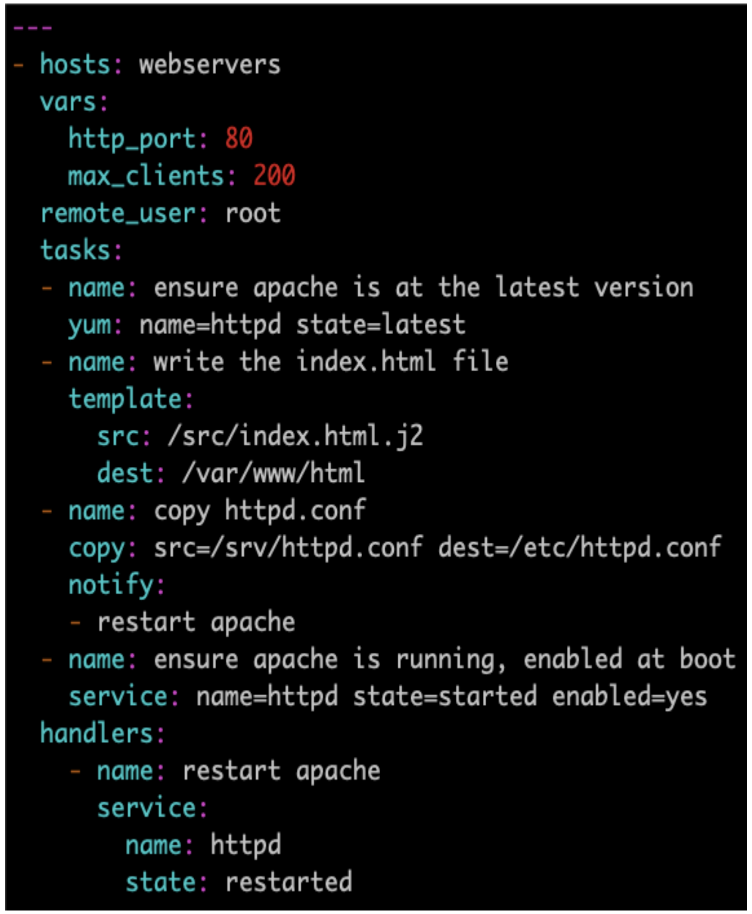


```
ansible_project/
|-- ansible.cfg
|-- ansible_playbook.yml
|-- hosts
|-- roles
|   `-- apache_role
```

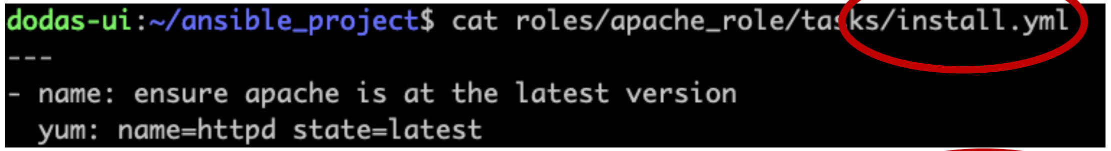


```
apache_role/
|-- defaults
|   `-- main.yml
|-- files
|-- handlers
|   `-- main.yml
|-- meta
|   `-- main.yml
|-- README.md
|-- tasks
|   `-- main.yml
|-- templates
|-- tests
|   |-- inventory
|   `-- test.yml
`-- vars
    `-- main.yml
```

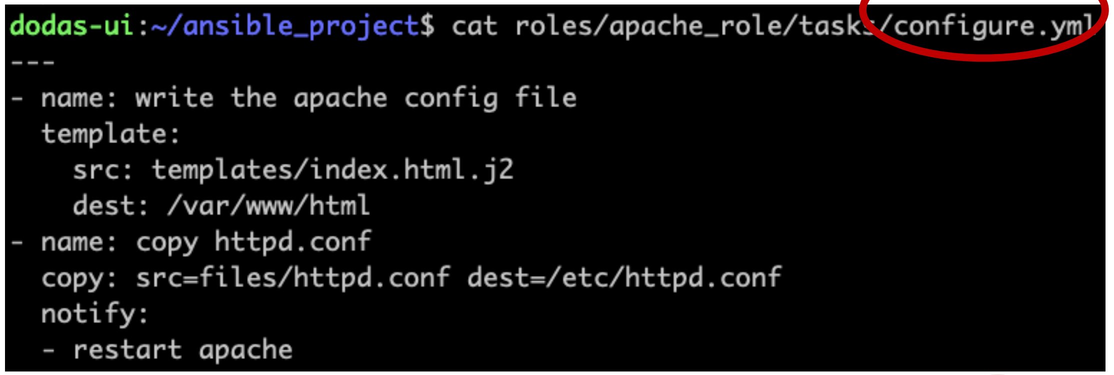
Extracting Tasks



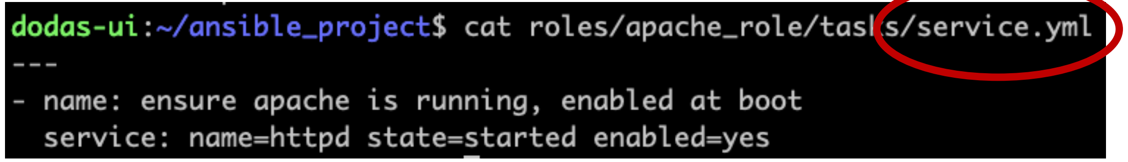
```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the index.html file
    template:
      src: /src/index.html.j2
      dest: /var/www/html
  - name: copy httpd.conf
    copy: src=/srv/httpd.conf dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running, enabled at boot
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```



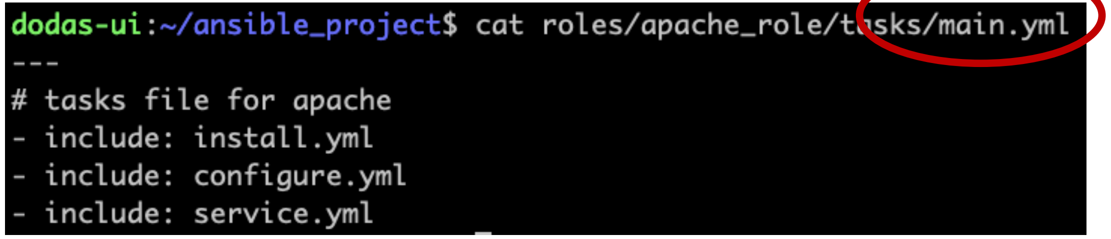
```
dodas-ui:~/ansible_project$ cat roles/apache_role/tasks/install.yml
---
- name: ensure apache is at the latest version
  yum: name=httpd state=latest
```



```
dodas-ui:~/ansible_project$ cat roles/apache_role/tasks/configure.yml
---
- name: write the apache config file
  template:
    src: templates/index.html.j2
    dest: /var/www/html
- name: copy httpd.conf
  copy: src=files/httpd.conf dest=/etc/httpd.conf
  notify:
  - restart apache
```

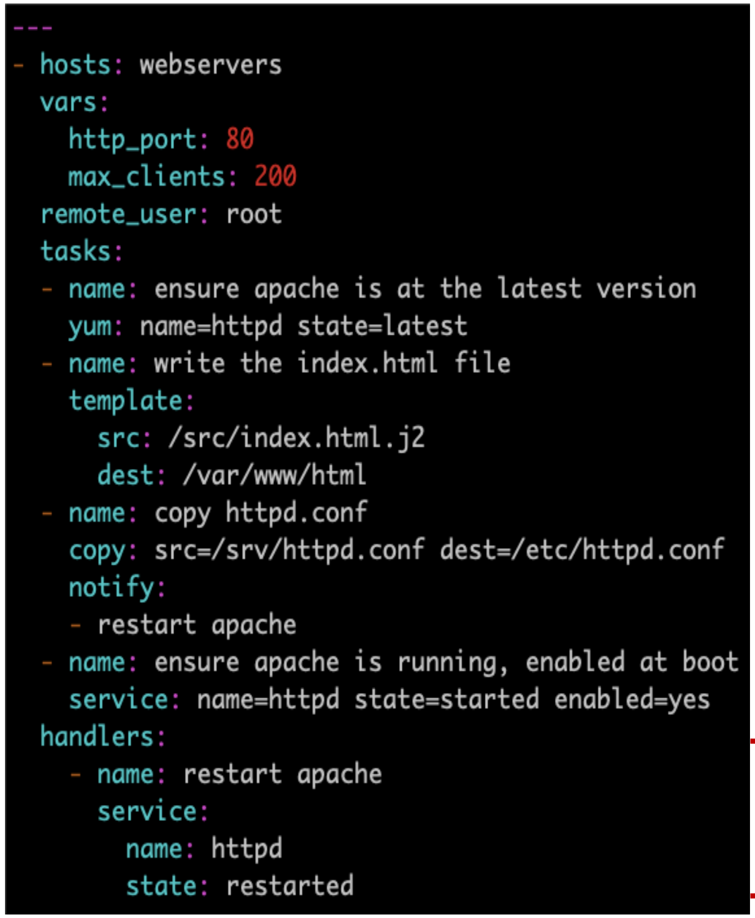


```
dodas-ui:~/ansible_project$ cat roles/apache_role/tasks/service.yml
---
- name: ensure apache is running, enabled at boot
  service: name=httpd state=started enabled=yes
```

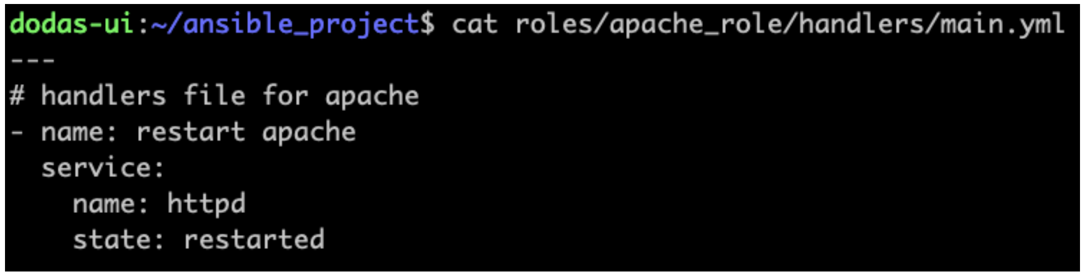


```
dodas-ui:~/ansible_project$ cat roles/apache_role/tasks/main.yml
---
# tasks file for apache
- include: install.yml
- include: configure.yml
- include: service.yml
```

Extracting handler



```
---
- 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 index.html file
    template:
      src: /src/index.html.j2
      dest: /var/www/html
  - name: copy httpd.conf
    copy: src=/srv/httpd.conf dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running, enabled at boot
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

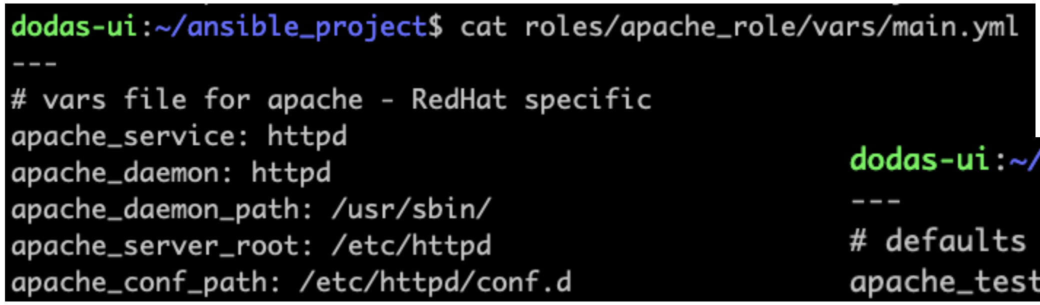


```
dodas-ui:~/ansible_project$ cat roles/apache_role/handlers/main.yml
---
# handlers file for apache
- name: restart apache
  service:
    name: httpd
    state: restarted
```

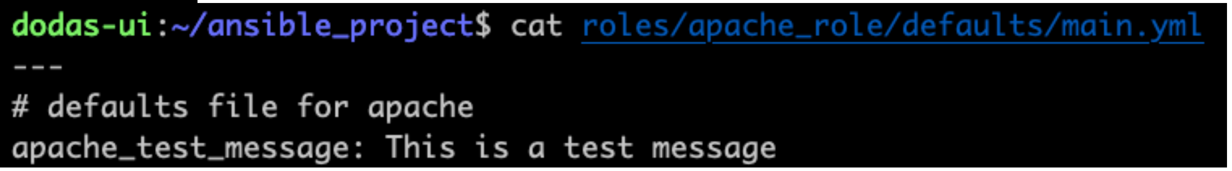
Variables

- two types of variables that can be defined in a role:
  - ***role variables***, loaded from *roles/<role_name>/vars/main.yaml*
    - used for example for system-specific constants that don't change much
  - ***role defaults***, which are loaded from *roles/<role_name>/defaults/main.yaml*
    - place holders for actual data, a reference of what variables a developer may be interested in defining with site-specific values
- Main difference – **precedence** order
  - *Defaults* - are the lowest order variables

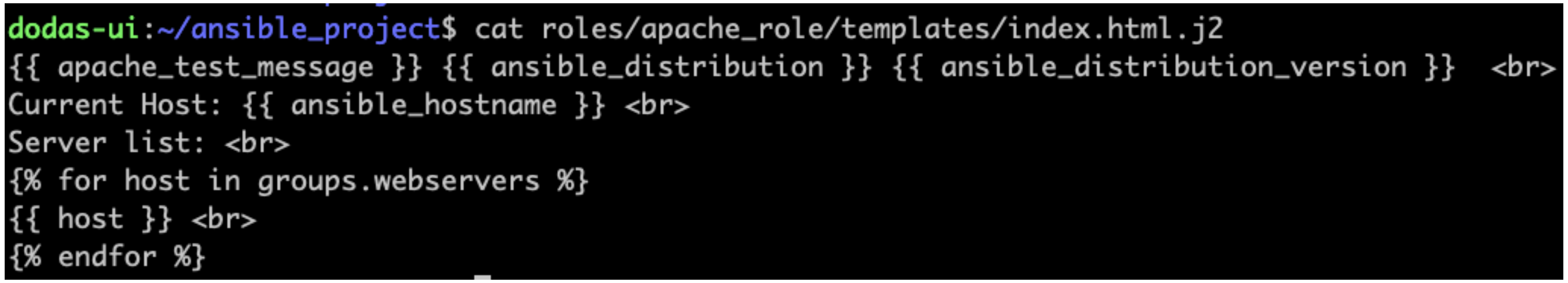
Variables in roles - examples



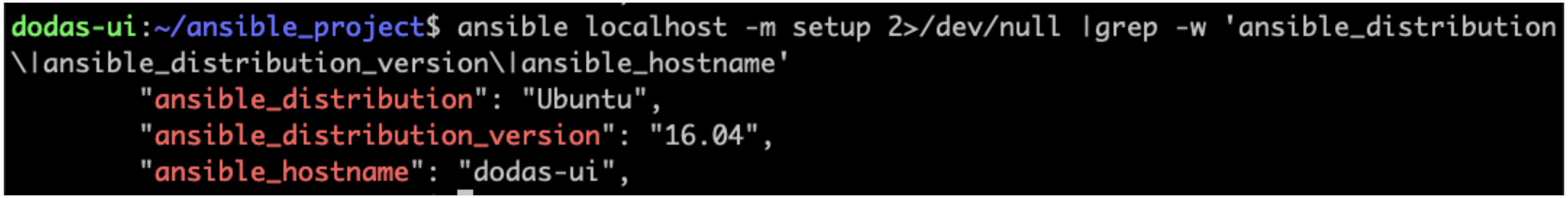
```
dodas-ui:~/ansible_project$ cat roles/apache_role/vars/main.yml
---
# vars file for apache - RedHat specific
apache_service: httpd
apache_daemon: httpd
apache_daemon_path: /usr/sbin/
apache_server_root: /etc/httpd
apache_conf_path: /etc/httpd/conf.d
```



```
dodas-ui:~/ansible_project$ cat roles/apache_role/defaults/main.yml
---
# defaults file for apache
apache_test_message: This is a test message
```

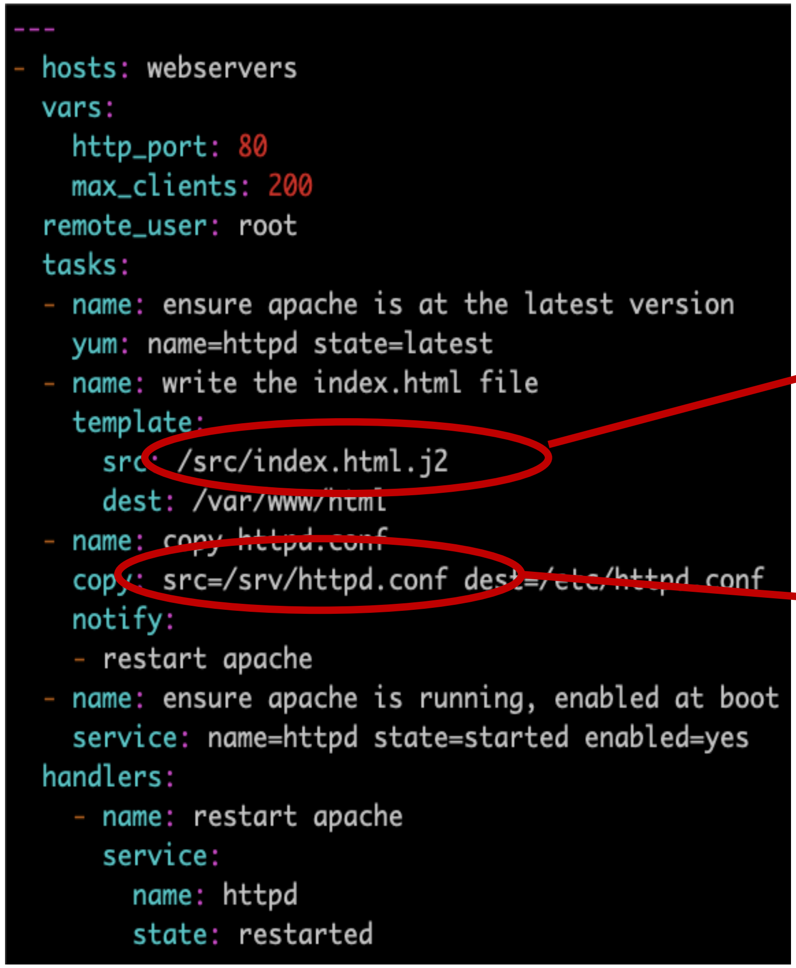


```
dodas-ui:~/ansible_project$ cat roles/apache_role/templates/index.html.j2
{{ apache_test_message }} {{ ansible_distribution }} {{ ansible_distribution_version }}  <br>
Current Host: {{ ansible_hostname }} <br>
Server list: <br>
{% for host in groups.webservers %}
{{ host }} <br>
{% endfor %}
```

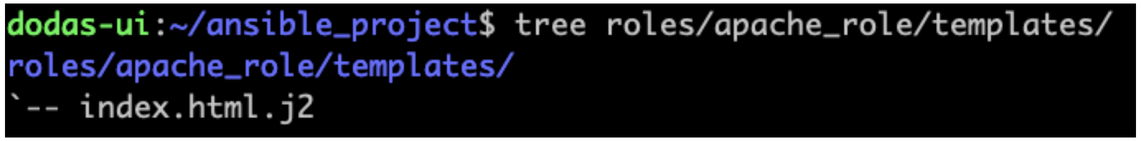


```
dodas-ui:~/ansible_project$ ansible localhost -m setup 2>/dev/null |grep -w 'ansible_distribution
\|ansible_distribution_version\|ansible_hostname'
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_version": "16.04",
        "ansible_hostname": "dodas-ui",
```

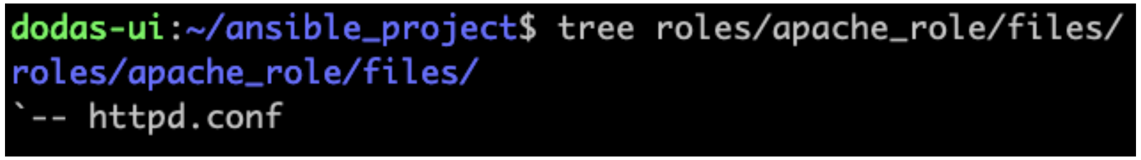
Moving config files



```
---
- 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 index.html file
    template:
      src: /src/index.html.j2
      dest: /var/www/html
  - name: copy httpd.conf
    copy: src=/srv/httpd.conf dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running, enabled at boot
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

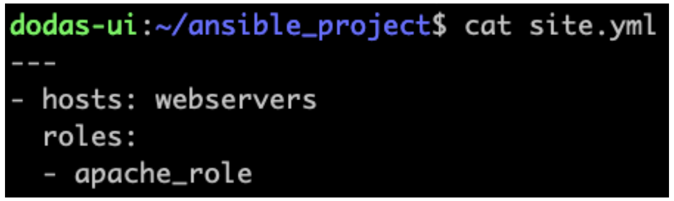


```
dodas-ui:~/ansible_project$ tree roles/apache_role/templates/
roles/apache_role/templates/
`-- index.html.j2
```



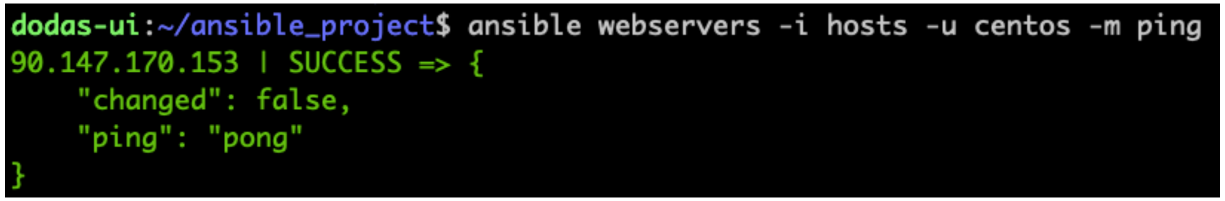
```
dodas-ui:~/ansible_project$ tree roles/apache_role/files/
roles/apache_role/files/
`-- httpd.conf
```

Using the New playbook that uses the New role



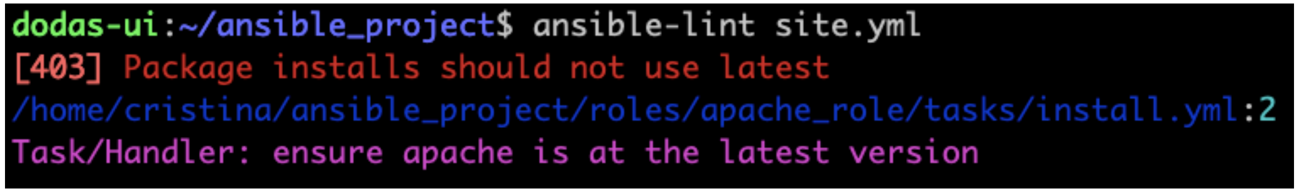
```
dodas-ui:~/ansible_project$ cat site.yml
---
- hosts: webservers
  roles:
  - apache_role
```

Check



```
dodas-ui:~/ansible_project$ ansible webservers -i hosts -u centos -m ping
90.147.170.153 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Check

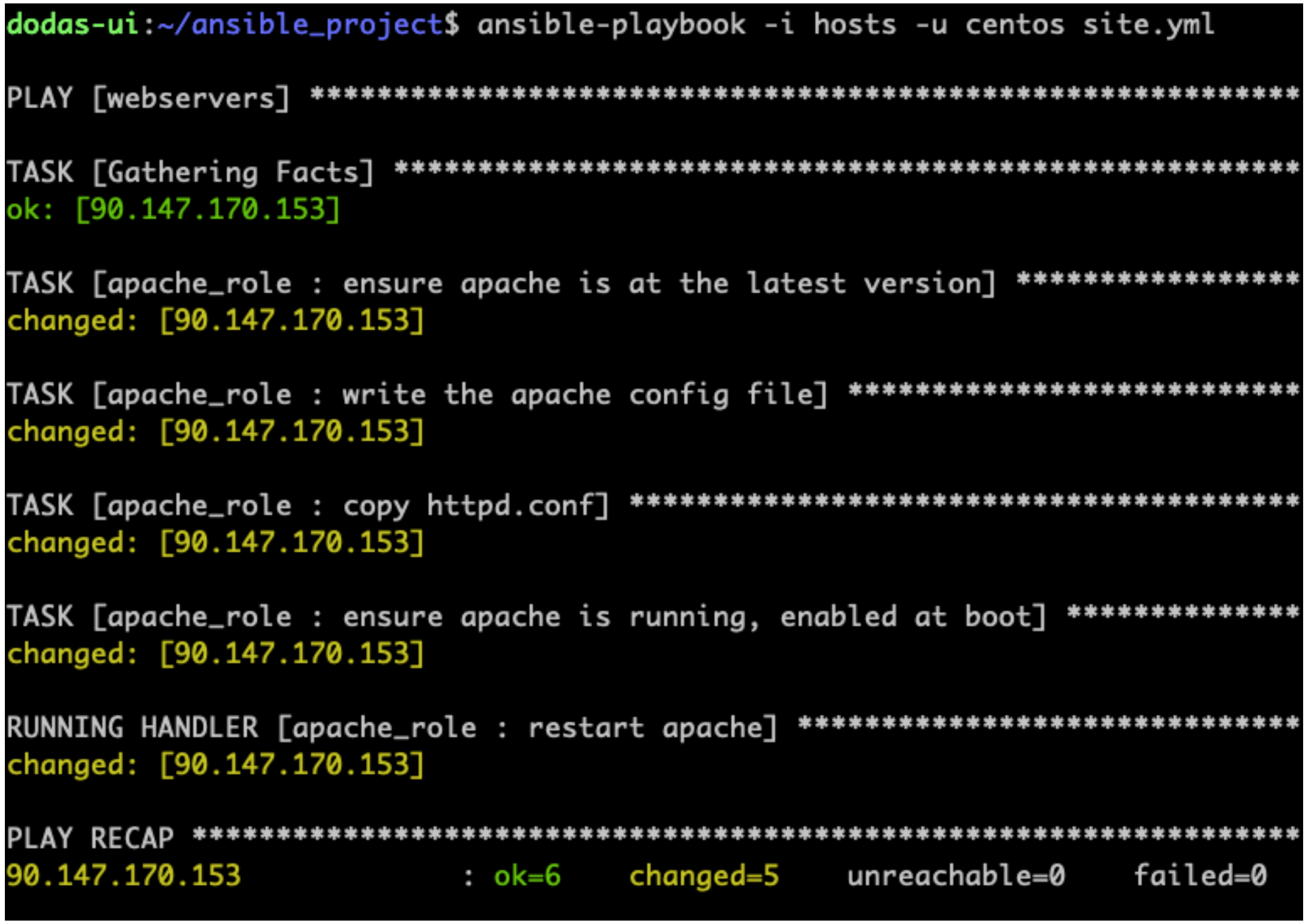


```
dodas-ui:~/ansible_project$ ansible-lint site.yml
[403] Package installs should not use latest
/home/cristina/ansible_project/roles/apache_role/tasks/install.yml:2
Task/Handler: ensure apache is at the latest version
```

PLAY

Reference: https://docs.ansible.com/ansible-lint/rules/default_rules.html

Using the New playbook that uses the New role

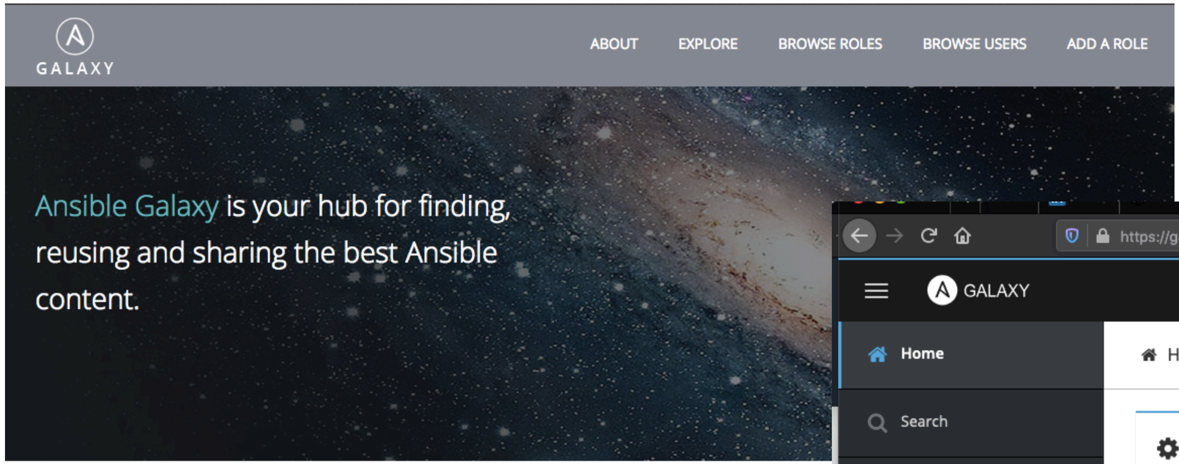


```
dodas-ui:~/ansible_project$ ansible-playbook -i hosts -u centos site.yml

PLAY [webservers] *************************************************************

TASK [Gathering Facts] ********************************************************
ok: [90.147.170.153]

TASK [apache_role : ensure apache is at the latest version] *****************
changed: [90.147.170.153]

TASK [apache_role : write the apache config file] ***************************
changed: [90.147.170.153]

TASK [apache_role : copy httpd.conf] ****************************************
changed: [90.147.170.153]

TASK [apache_role : ensure apache is running, enabled at boot] **************
changed: [90.147.170.153]

RUNNING HANDLER [apache_role : restart apache] ******************************
changed: [90.147.170.153]

PLAY RECAP ********************************************************************
90.147.170.153             : ok=6    changed=5    unreachable=0    failed=0
```

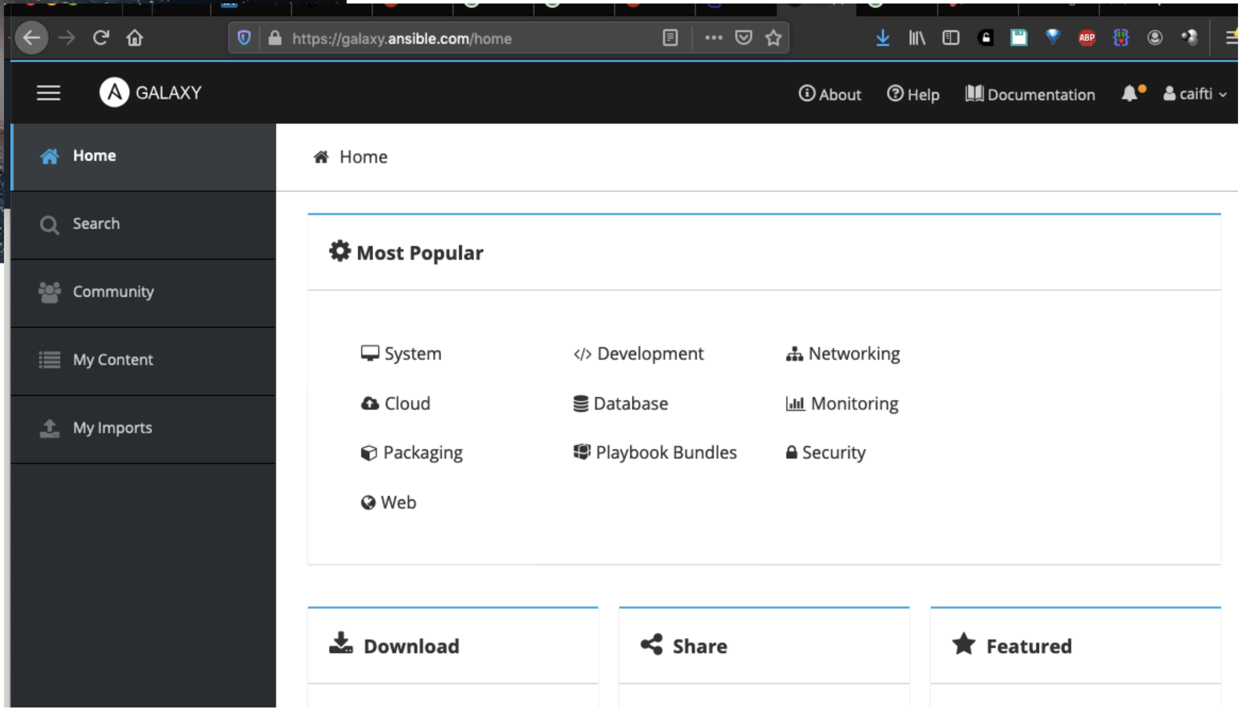
PLAY

• Ansible Galaxy – Reusing Roles

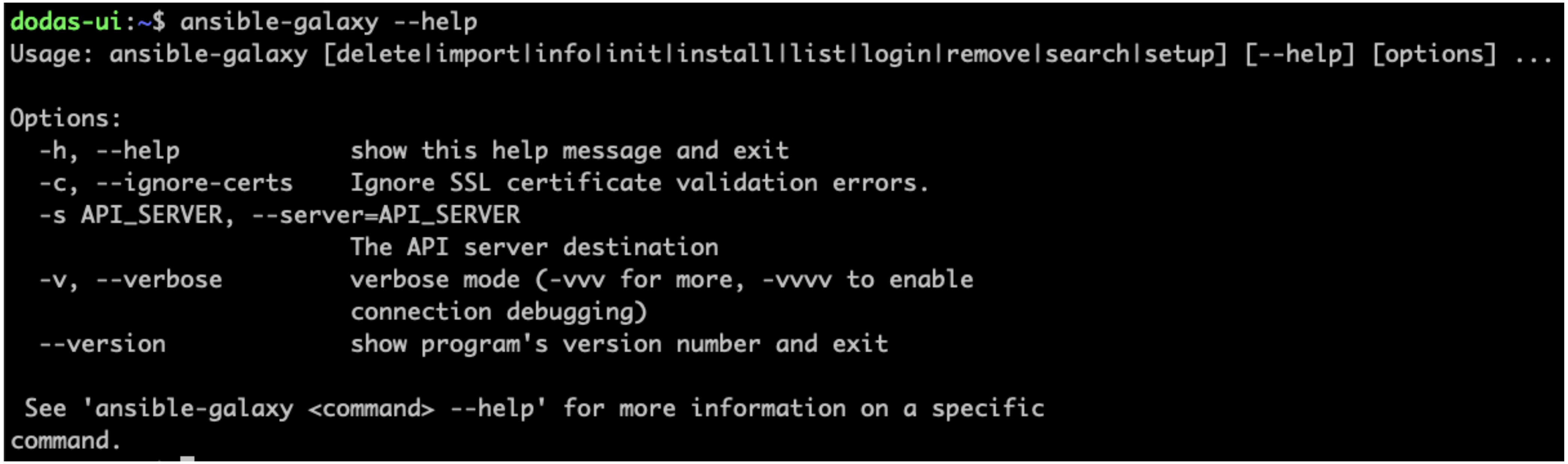
Ansible Galaxy – Reusing Roles



Ansible Galaxy is your hub for finding, reusing and sharing the best Ansible content.

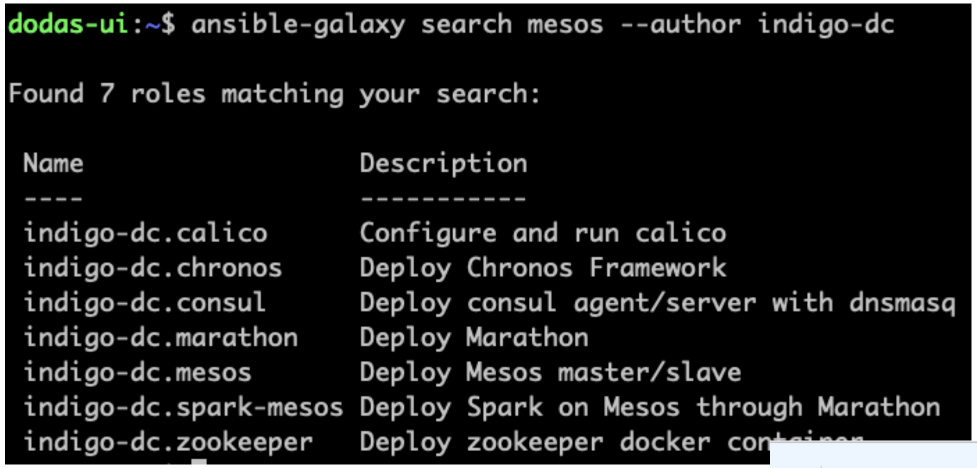


ansible-galaxy CLI tool

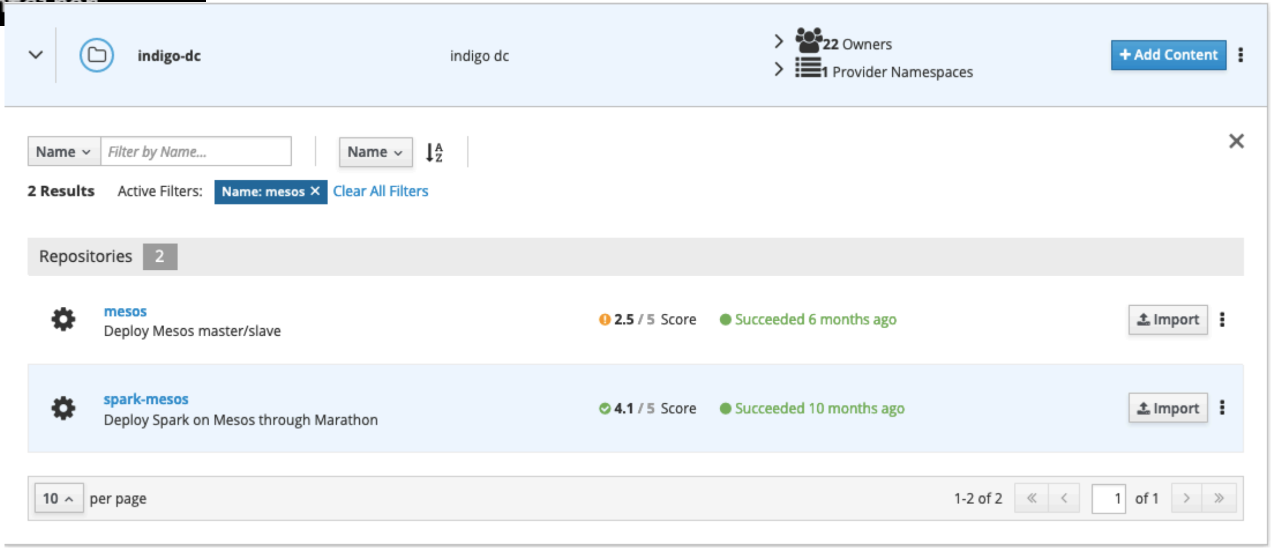


```
dodas-ui:~$ ansible-galaxy --help
Usage: ansible-galaxy [delete|import|info|init|install|list|login|remove|search|setup] [--help] [options] ...

Options:
  -h, --help            show this help message and exit
  -c, --ignore-certs    Ignore SSL certificate validation errors.
  -s API_SERVER, --server=API_SERVER
                        The API server destination
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
  --version             show program's version number and exit

 See 'ansible-galaxy <command> --help' for more information on a specific
command.
```

Search Roles



```
dodas-ui:~$ ansible-galaxy search mesos --author indigo-dc

Found 7 roles matching your search:

 Name                      Description
 ----                      -----------
 indigo-dc.calico          Configure and run calico
 indigo-dc.chronos         Deploy Chronos Framework
 indigo-dc.consul          Deploy consul agent/server with dnsmasq
 indigo-dc.marathon        Deploy Marathon
 indigo-dc.mesos           Deploy Mesos master/slave
 indigo-dc.spark-mesos     Deploy Spark on Mesos through Marathon
 indigo-dc.zookeeper       Deploy zookeeper docker con
```



indigo-dc　indigo dc　22 Owners　1 Provider Namespaces　+ Add Content

Name　Filter by Name...　Name

2 Results　Active Filters: Name: mesos　Clear All Filters

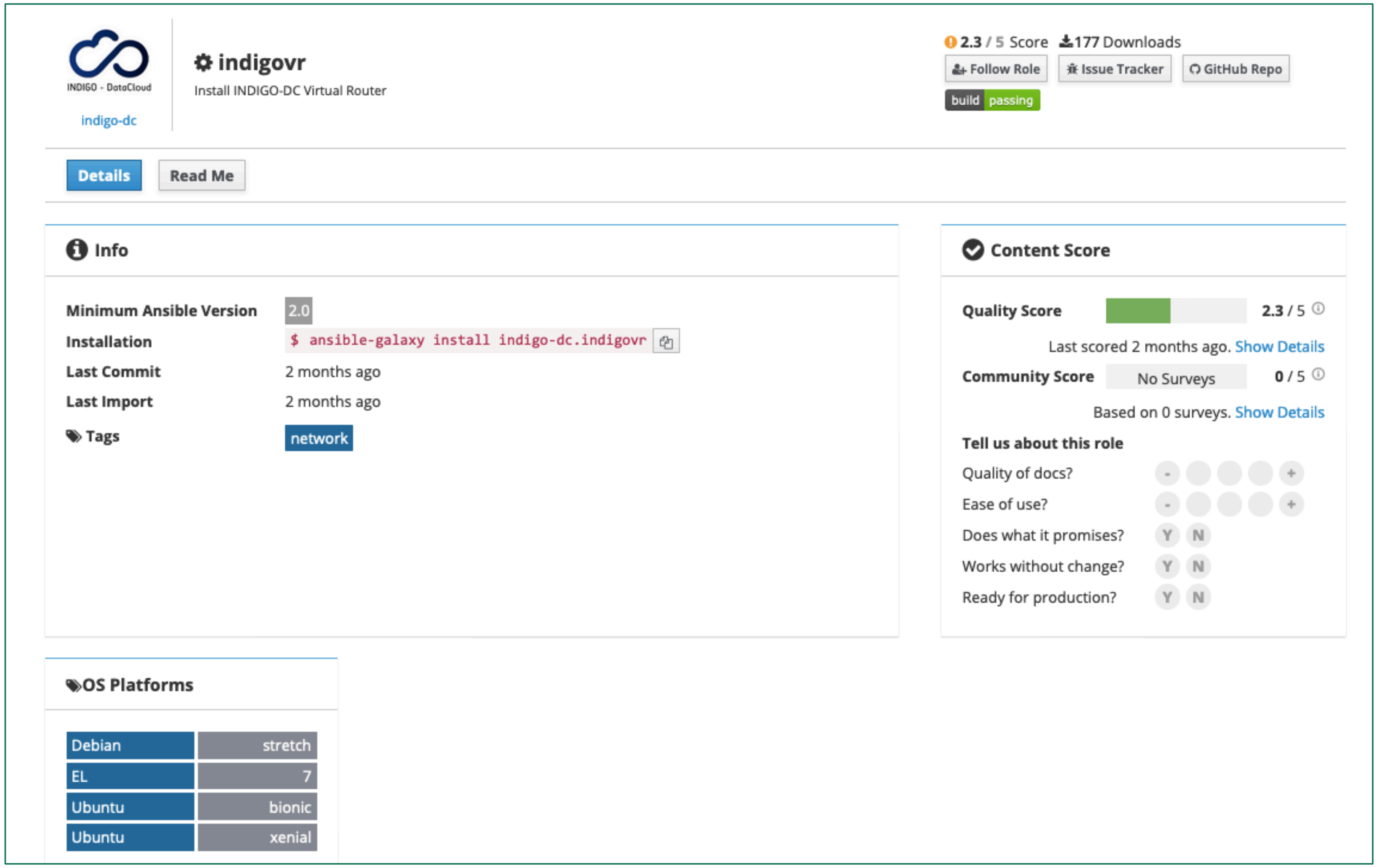
Repositories 2

mesos
Deploy Mesos master/slave
2.5 / 5 Score　Succeeded 6 months ago　Import

spark-mesos
Deploy Spark on Mesos through Marathon
4.1 / 5 Score　Succeeded 10 months ago　Import

10 per page　1-2 of 2　1 of 1

Get Info



indigo-dc

indigovr

Install INDIGO-DC Virtual Router

2.3 / 5 Score 177 Downloads

Follow Role | Issue Tracker | GitHub Repo

build passing

Details | Read Me

Info

Minimum Ansible Version	2.0
Installation	`$ ansible-galaxy install indigo-dc.indigovr`
Last Commit	2 months ago
Last Import	2 months ago
Tags	network

Content Score

Quality Score 2.3 / 5

Last scored 2 months ago. Show Details

Community Score No Surveys 0 / 5

Based on 0 surveys. Show Details

Tell us about this role

Quality of docs? - +

Ease of use? - +

Does what it promises? Y N

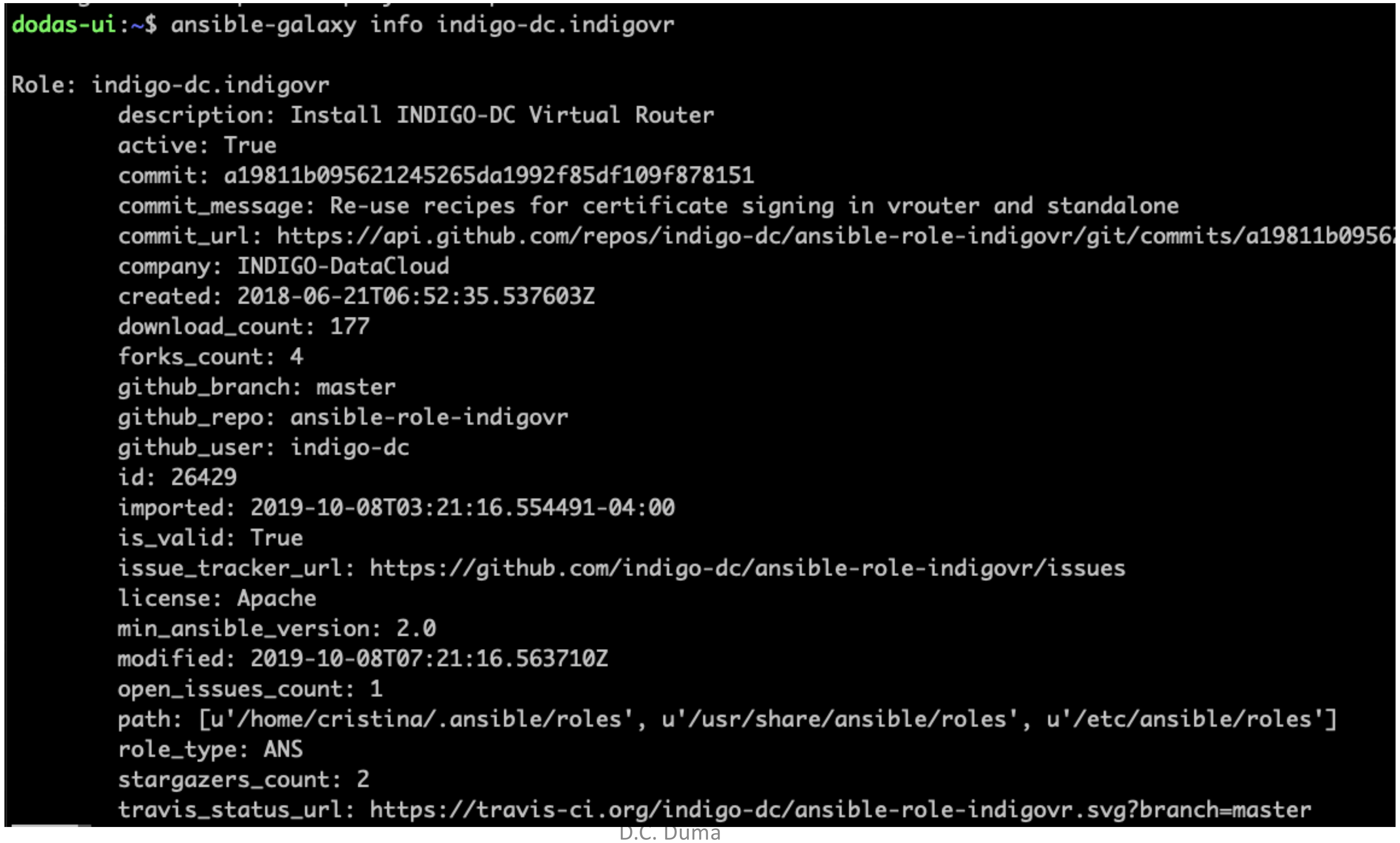
Works without change? Y N

Ready for production? Y N

OS Platforms

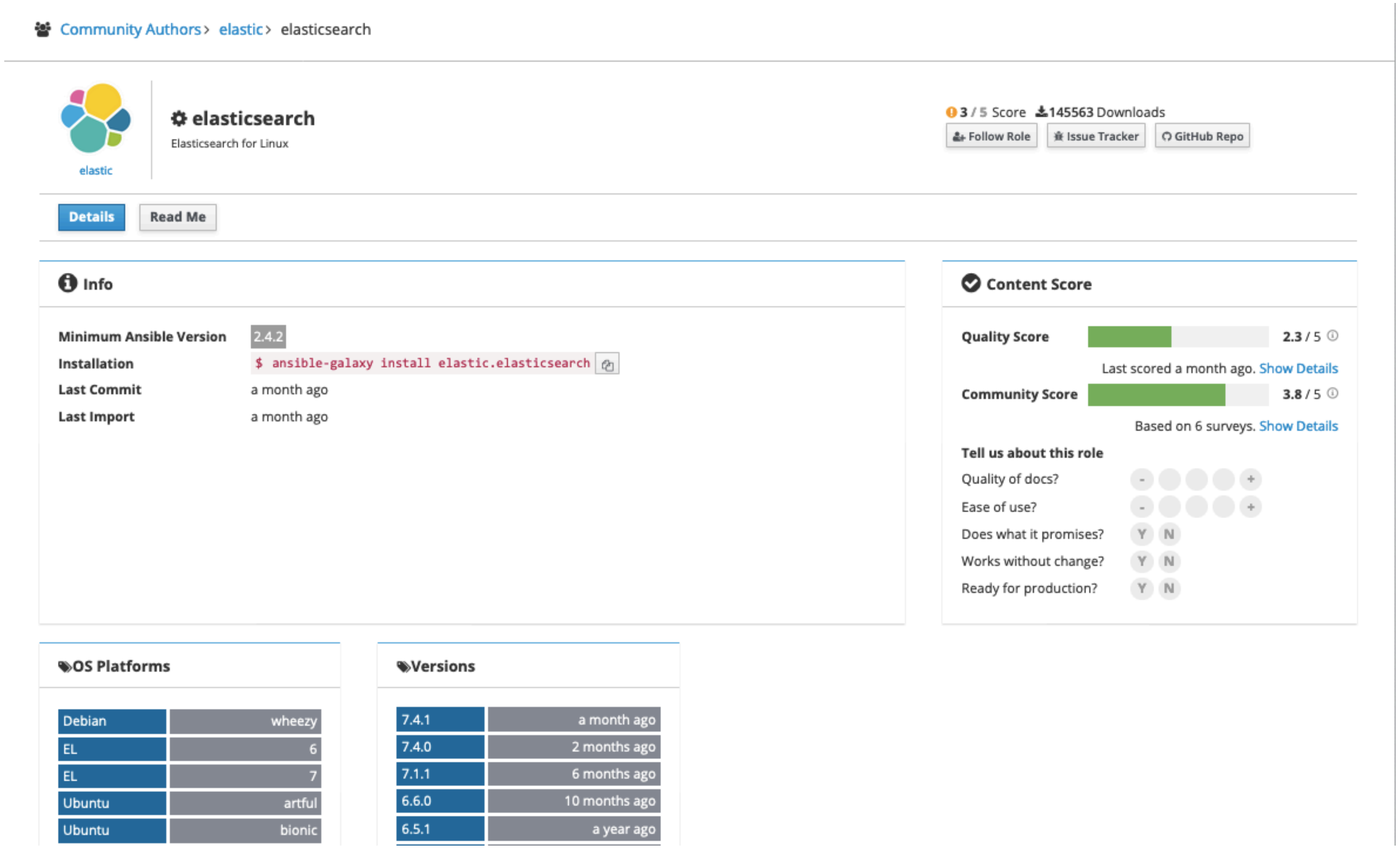
Debian	stretch
EL	7
Ubuntu	bionic
Ubuntu	xenial

Get Info (2)



```
dodas-ui:~$ ansible-galaxy info indigo-dc.indigovr

Role: indigo-dc.indigovr
        description: Install INDIGO-DC Virtual Router
        active: True
        commit: a19811b095621245265da1992f85df109f878151
        commit_message: Re-use recipes for certificate signing in vrouter and standalone
        commit_url: https://api.github.com/repos/indigo-dc/ansible-role-indigovr/git/commits/a19811b0956
        company: INDIGO-DataCloud
        created: 2018-06-21T06:52:35.537603Z
        download_count: 177
        forks_count: 4
        github_branch: master
        github_repo: ansible-role-indigovr
        github_user: indigo-dc
        id: 26429
        imported: 2019-10-08T03:21:16.554491-04:00
        is_valid: True
        issue_tracker_url: https://github.com/indigo-dc/ansible-role-indigovr/issues
        license: Apache
        min_ansible_version: 2.0
        modified: 2019-10-08T07:21:16.563710Z
        open_issues_count: 1
        path: [u'/home/cristina/.ansible/roles', u'/usr/share/ansible/roles', u'/etc/ansible/roles']
        role_type: ANS
        stargazers_count: 2
        travis_status_url: https://travis-ci.org/indigo-dc/ansible-role-indigovr.svg?branch=master
```

One more!!



Community Authors › elastic › elasticsearch

elasticsearch
Elasticsearch for Linux

elastic

3 / 5 Score 145563 Downloads
Follow Role | Issue Tracker | GitHub Repo

Details | Read Me

Info

Minimum Ansible Version	2.4.2
Installation	`$ ansible-galaxy install elastic.elasticsearch`
Last Commit	a month ago
Last Import	a month ago

Content Score

Quality Score 2.3 / 5
Last scored a month ago. Show Details

Community Score 3.8 / 5
Based on 6 surveys. Show Details

Tell us about this role
Quality of docs? - +
Ease of use? - +
Does what it promises? Y N
Works without change? Y N
Ready for production? Y N

OS Platforms

Debian	wheezy
EL	6
EL	7
Ubuntu	artful
Ubuntu	bionic

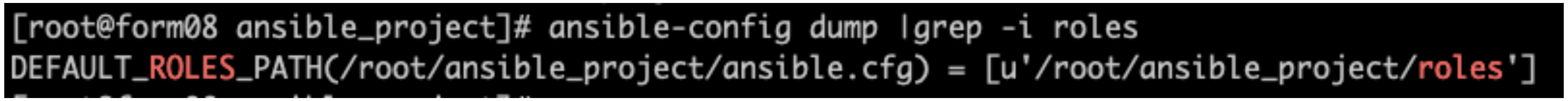
Versions

7.4.1	a month ago
7.4.0	2 months ago
7.1.1	6 months ago
6.6.0	10 months ago
6.5.1	a year ago

Download and Install – from Galaxy

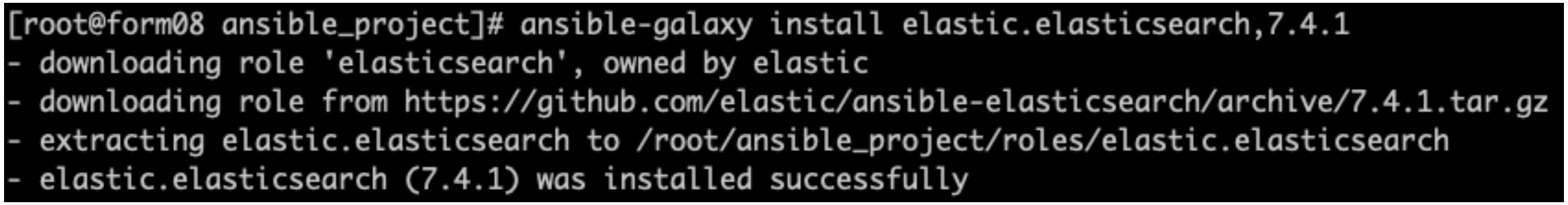
```
ansible-galaxy install role_name(s)[,version]
```

Where:



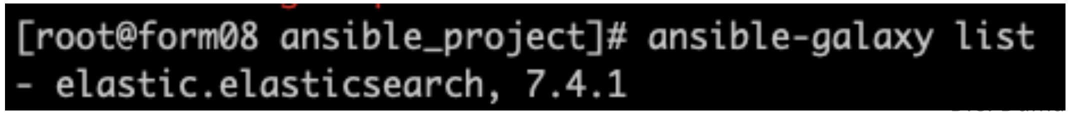
```
[root@form08 ansible_project]# ansible-config dump |grep -i roles
DEFAULT_ROLES_PATH(/root/ansible_project/ansible.cfg) = [u'/root/ansible_project/roles']
```

Do:



```
[root@form08 ansible_project]# ansible-galaxy install elastic.elasticsearch,7.4.1
- downloading role 'elasticsearch', owned by elastic
- downloading role from https://github.com/elastic/ansible-elasticsearch/archive/7.4.1.tar.gz
- extracting elastic.elasticsearch to /root/ansible_project/roles/elastic.elasticsearch
- elastic.elasticsearch (7.4.1) was installed successfully
```

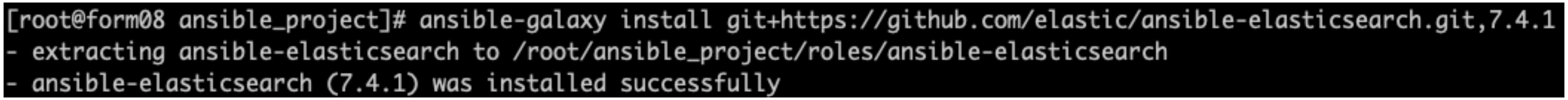
Check:



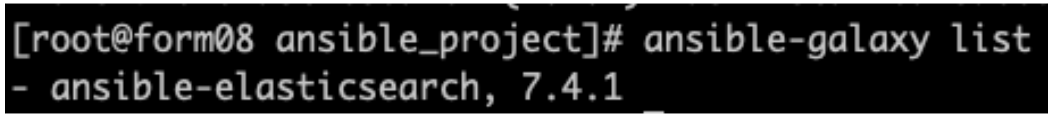
```
[root@form08 ansible_project]# ansible-galaxy list
- elastic.elasticsearch, 7.4.1
```

Download and Install – from Github

```
ansible-galaxy install scm+role_repo_url[,version]
```



```
[root@form08 ansible_project]# ansible-galaxy install git+https://github.com/elastic/ansible-elasticsearch.git,7.4.1
- extracting ansible-elasticsearch to /root/ansible_project/roles/ansible-elasticsearch
- ansible-elasticsearch (7.4.1) was installed successfully
```



```
[root@form08 ansible_project]# ansible-galaxy list
- ansible-elasticsearch, 7.4.1
```

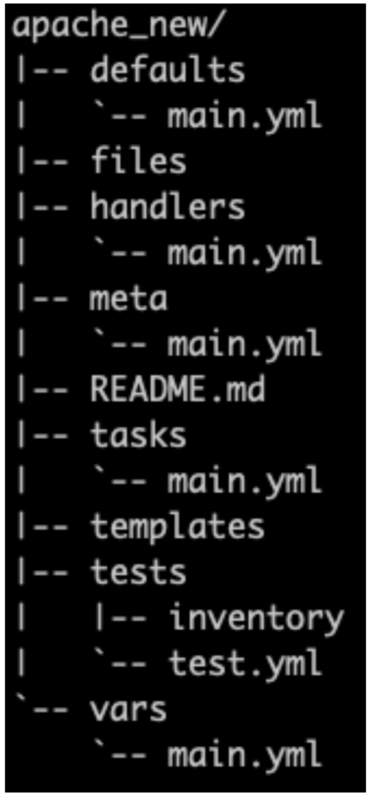
Creating roles with ansible-galaxy

- **ansible-galaxy** tool can also be used to generate ***scaffolding***, an initial set of files and directories involved in a role:

```
[ansible_project]# ansible-galaxy init apache_new
-   apache_new was created successfully
```

Importing roles – using CLI & WebUI

```
[ansible_project]# ansible-galaxy list
- ansible-elasticsearch, 7.4.1
-  apache_new, (unknown version)

[ansible_project]# ansible-galaxy init --init-path=INIT_PATH
apache_new
```



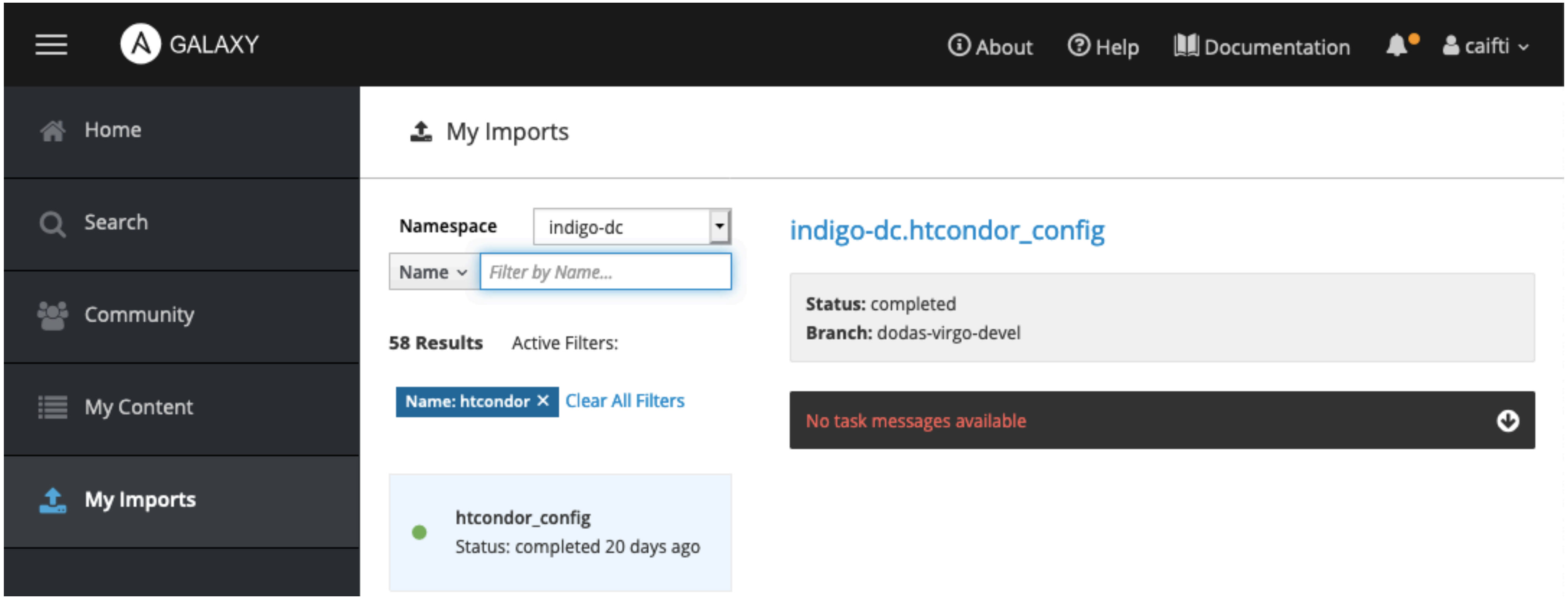
```
apache_new/
|-- defaults
|   `-- main.yml
|-- files
|-- handlers
|   `-- main.yml
|-- meta
|   `-- main.yml
|-- README.md
|-- tasks
|   `-- main.yml
|-- templates
|-- tests
|   |-- inventory
|   `-- test.yml
`-- vars
    `-- main.yml
```

Importing roles – using CLI & WebUI

- CLI
  - GitHub repository for new role
  - login to Ansible Galaxy
  - ansible import

```
# ansible-galaxy import -h
Usage: ansible-galaxy import [options] github_user github_repo


Options:
  --branch=REFERENCE    The name of a branch to import. Defaults to the epository's default
           branch (usually master)
  -h, --help            show this help message and exit
  -c, --ignore-certs    Ignore SSL certificate validation errors.
  --no-wait             Don't wait for import results.
  --role-name=ROLE_NAME
                        The name the role should have, if different than the
                        repo name
  -s API_SERVER, --server=API_SERVER
                        The API server destination
  --status              Check the status of the most recent import request for
                        given github_user/github_repo.
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
  --version             show program's version number and exit
```

Import using Ansible Galaxy Web GUI



My Imports

Namespace: indigo-dc

Name: Filter by Name...

58 Results Active Filters:

Name: htcondor ✕ Clear All Filters

htcondor_config
Status: completed 20 days ago

indigo-dc.htcondor_config

Status: completed
Branch: dodas-virgo-devel

No task messages available

Ansible – advanced usage

- Debbuging
- Optmization

Verbose & debug

- Verbose flag: ***-vvv*** or ***–verbose***
    - prints all the values that were returned by each module after it runs

```
# ansible-playbook --verbose playbook.yml
```

- ***debug*** module - prints statements during execution and can be useful for debugging variables or expressions without necessarily halting the playbook. Useful for debugging together with the 'when:' directive.

```
- debug: var=myvariable
- debug: msg="The value of myvariable is {{ var }}"
- debug:
    msg: "System {{ inventory_hostname }} has gateway {{ 
ansible_default_ipv4.gateway }}"
  when: ansible_default_ipv4.gateway is defined
```

assert & pause

- ***Assert module*** - module asserts that given expressions are true

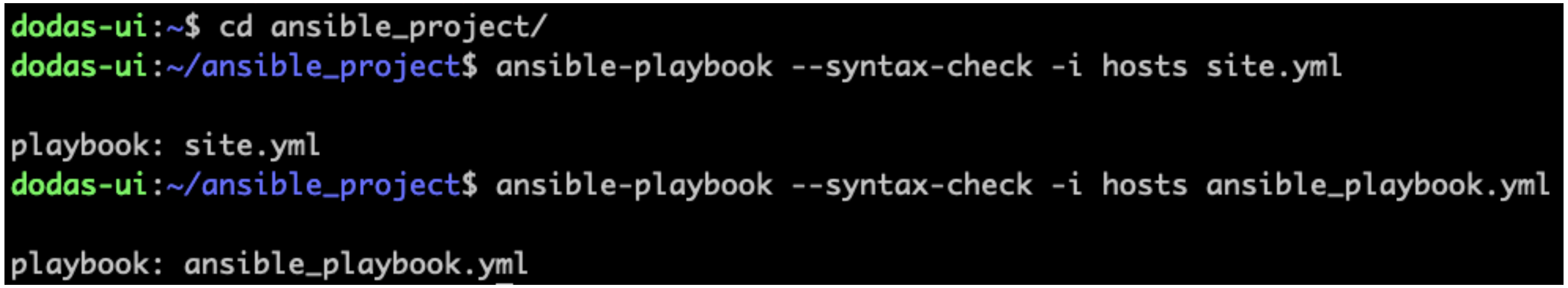
```
- assert: { that: "ansible_os_family != 'RedHat’» }
```

- ***Pause module*** - pauses playbook execution for a **set amount of time**, or until a prompt is acknowledged
  - default behavior is to **pause with a prompt**

```
# Pause for 5 minutes to build app cache.
- pause:
    minutes: 5
```

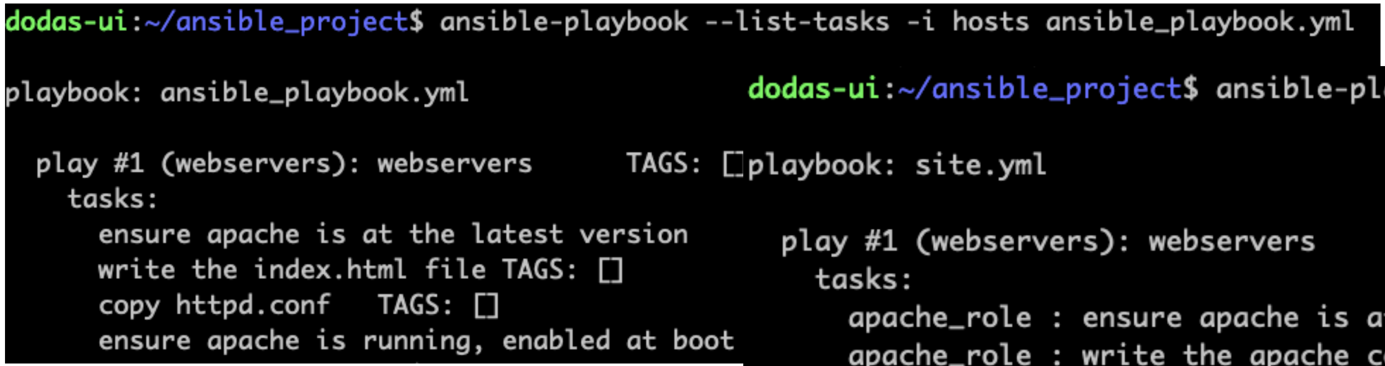
syntax check & list tasks

- ***«--syntax-check»*** perform a syntax check on the playbook, but do not execute it

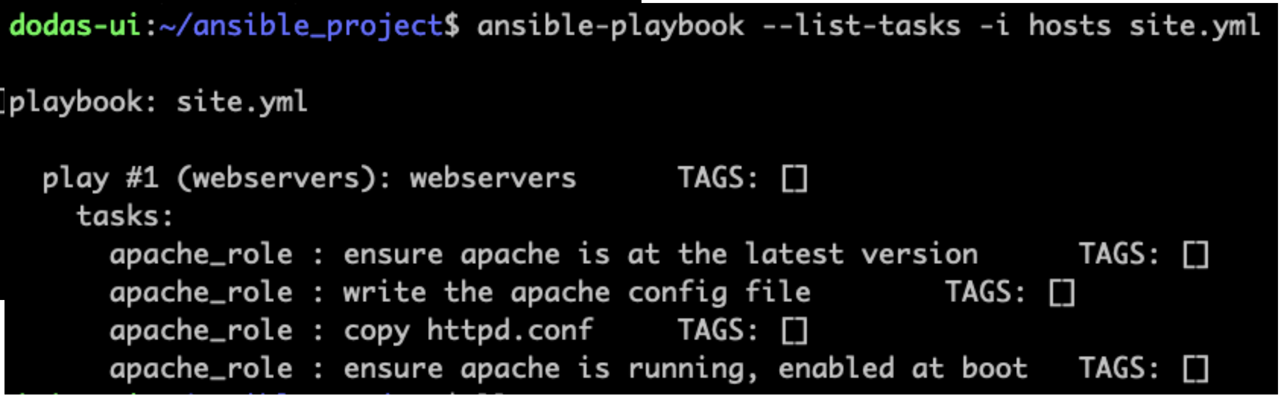


```
dodas-ui:~$ cd ansible_project/
dodas-ui:~/ansible_project$ ansible-playbook --syntax-check -i hosts site.yml

playbook: site.yml
dodas-ui:~/ansible_project$ ansible-playbook --syntax-check -i hosts ansible_playbook.yml

playbook: ansible_playbook.yml
```

- ***«--list-tasks»*** list all tasks that would be executed



```
dodas-ui:~/ansible_project$ ansible-playbook --list-tasks -i hosts ansible_playbook.yml

playbook: ansible_playbook.yml

  play #1 (webservers): webservers      TAGS: []
    tasks:
      ensure apache is at the latest version
      write the index.html file TAGS: []
      copy httpd.conf    TAGS: []
      ensure apache is running, enabled at boot
```



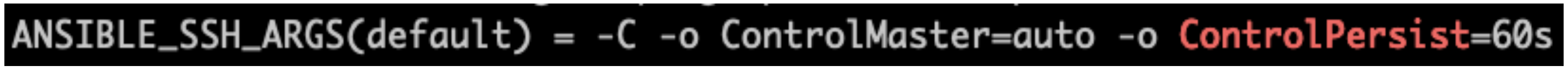
```
dodas-ui:~/ansible_project$ ansible-playbook --list-tasks -i hosts site.yml

playbook: site.yml

  play #1 (webservers): webservers      TAGS: []
    tasks:
      apache_role : ensure apache is at the latest version      TAGS: []
      apache_role : write the apache config file        TAGS: []
      apache_role : copy httpd.conf     TAGS: []
      apache_role : ensure apache is running, enabled at boot   TAGS: []
```

Optimization (1)

- ***SSH multiplexing & ControlPersist***
  - When Ansible runs a playbook, it will make **many SSH connections**, in order to do things such as copy over files and run commands.
  - Each time Ansible makes a new SSH connection to a host, it has to pay the **negotiation penalty**.
  - OpenSSH supports an optimization called ***SSH multiplexing***, which is also referred to as ***ControlPersist***:
    - ➢ a master connection is opened for each host and a control socket is used to communicate with the remote host instead of making a new TCP connection

➢In Ansible:

```
ControlMaster default=auto
ControlPath default=$HOME/.ansible/cp/ansible-ss-%h-%p-%r
ControlPersist 60s
```



```
ANSIBLE_SSH_ARGS(default) = -C -o ControlMaster=auto -o ControlPersist=60s
```

Optimization (2)

- ***Pipelining***
  - When Ansible executes a task
    - It generates a Python script based on the module being invoked
    - Then it copies the Python script to the host
    - Finally, it executes the Python script
  - Enabling **pipelining** reduces the number of SSH operations required to execute a module on the remote server
    - by **executing many** ansible modules **without** actual **file transfer**.
    - this can result in a **very significant performance improvement** when enabled
    - however when using "***sudo***:" operations you must first ***disable 'requiretty'*** in /etc/sudoers on all managed hosts.

Optimization (3)

- Facts caching
  - When a fact cache is enabled and there is valid data for a host, Ansible will use that rather than running an implicit setup job on a remote host.
  - Plugins => ***# ansible-doc -t cache -l***

jsonfile	JSON formatted files.
memcached	Use memcached DB for cache
memory	RAM backed, non persistent
mongodb	Use MongoDB for caching
pickle	Pickle formatted files.
redis	Use Redis DB for cache
yaml	YAML formatted files.

```
- [defaults]
- gathering = smart fact
- _caching_timeout = 86400
  fact_caching = ... .
```

Example – using elastic.elasticsearch module

- Connect to your VM & become root

```
# ssh -i ~/<path>/devopskeyXX -l centos devopsXX.cloud.cnaf.infn.it
```

- Get project from baltig

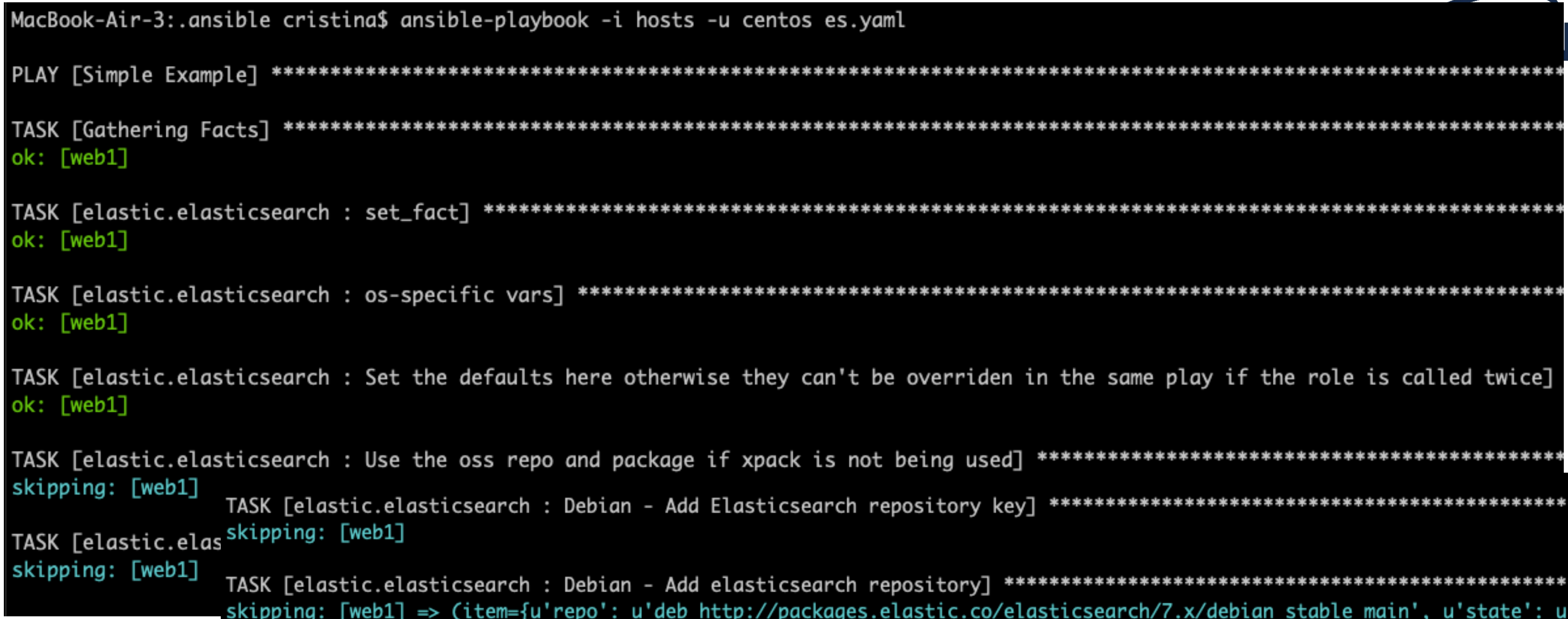
```
# git clone
https://baltig.infn.it/corsi_formazione_ccr/corso_bd_2019.git
```

- Update files to meet your environment – ansible.cfg, hosts…
- Install *elasticsearch* role

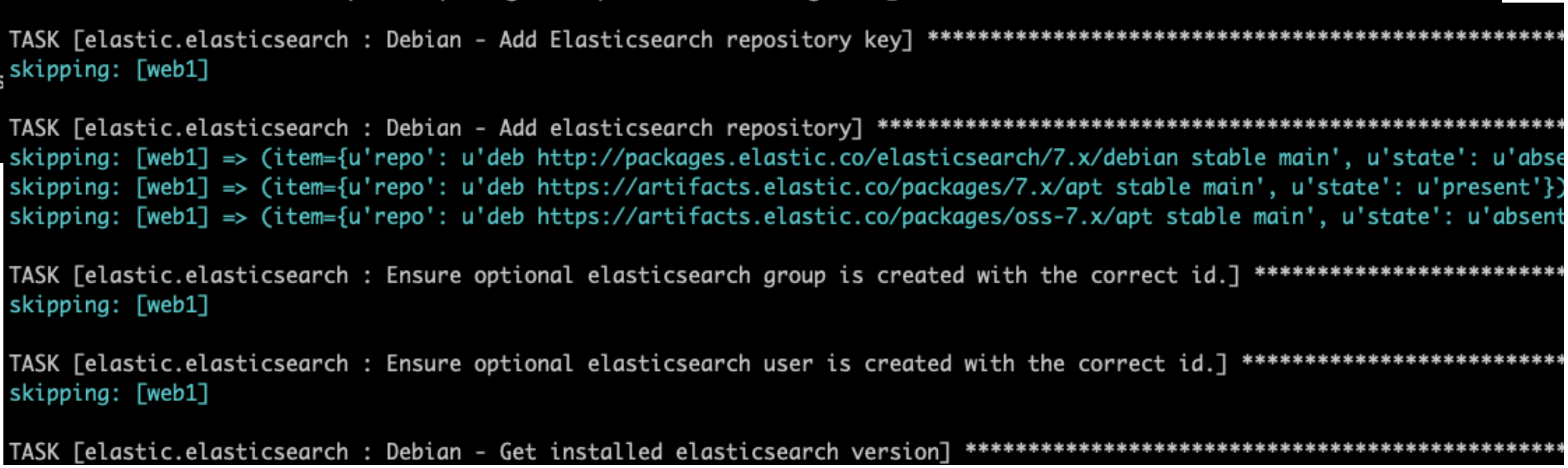
```
# ansible-galaxy install elastic.elasticsearch,7.4.1
```

- Check ….
- Run

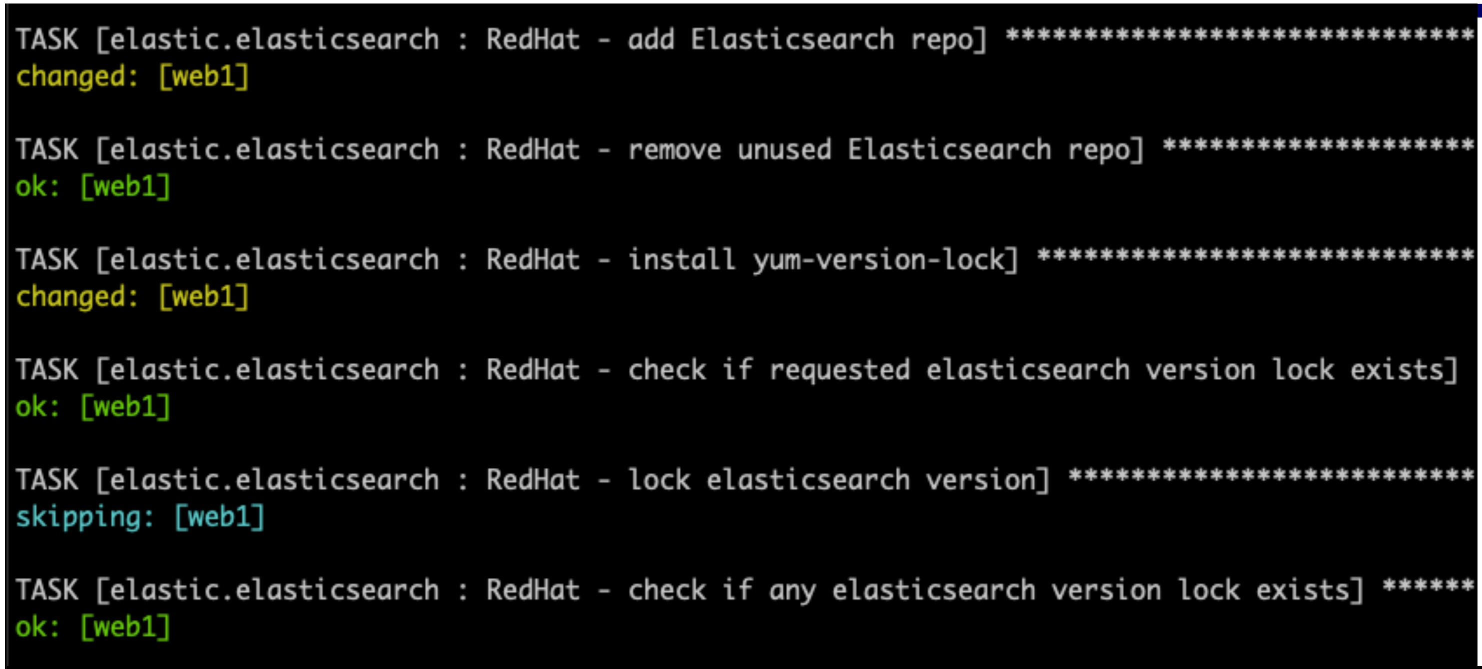
```
# ansible-playbook -i hosts es.yaml
```



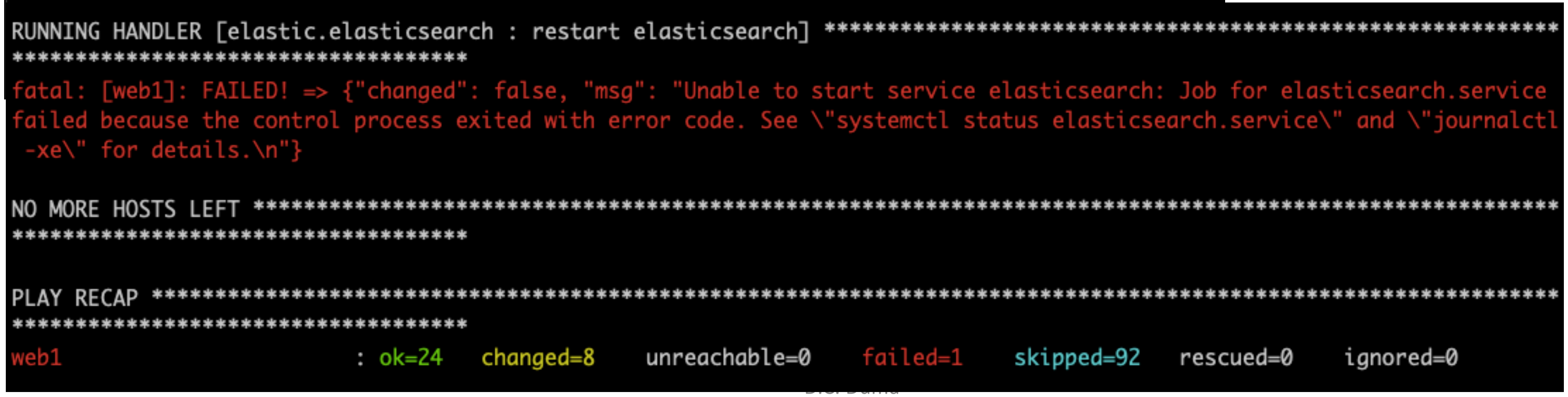
```
MacBook-Air-3:.ansible cristina$ ansible-playbook -i hosts -u centos es.yaml

PLAY [Simple Example] ***************************************************************************

TASK [Gathering Facts] **************************************************************************
ok: [web1]

TASK [elastic.elasticsearch : set_fact] *********************************************************
ok: [web1]

TASK [elastic.elasticsearch : os-specific vars] *************************************************
ok: [web1]

TASK [elastic.elasticsearch : Set the defaults here otherwise they can't be overriden in the same play if the role is called twice]
ok: [web1]

TASK [elastic.elasticsearch : Use the oss repo and package if xpack is not being used] **********
skipping: [web1]

TASK [elastic.elas
skipping: [web1]
```



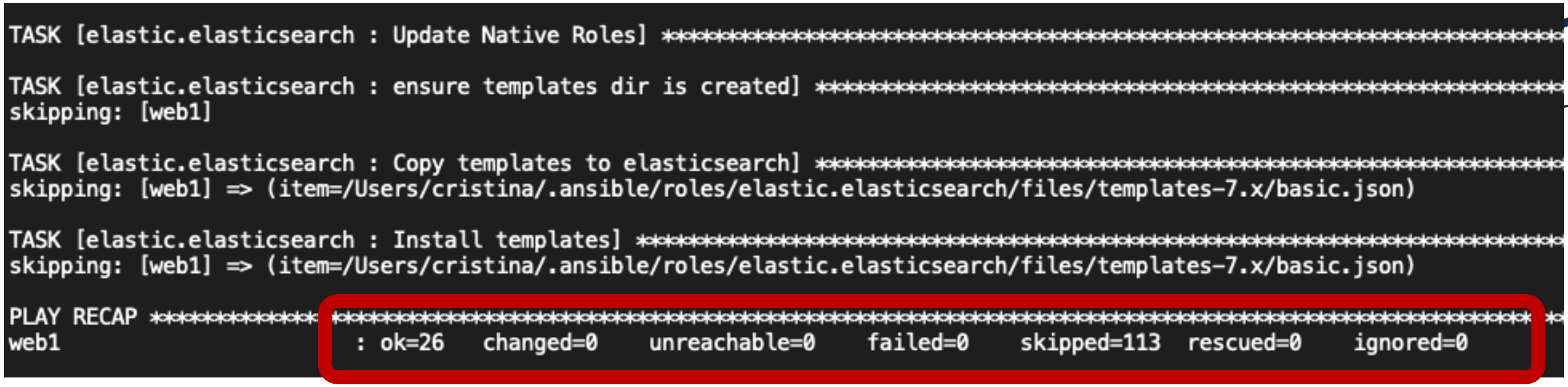
```
TASK [elastic.elasticsearch : Debian - Add Elasticsearch repository key] ************************
skipping: [web1]

TASK [elastic.elasticsearch : Debian - Add elasticsearch repository] ****************************
skipping: [web1] => (item={u'repo': u'deb http://packages.elastic.co/elasticsearch/7.x/debian stable main', u'state': u'abse
skipping: [web1] => (item={u'repo': u'deb https://artifacts.elastic.co/packages/7.x/apt stable main', u'state': u'present'})
skipping: [web1] => (item={u'repo': u'deb https://artifacts.elastic.co/packages/oss-7.x/apt stable main', u'state': u'absent

TASK [elastic.elasticsearch : Ensure optional elasticsearch group is created with the correct id.] **********************
skipping: [web1]

TASK [elastic.elasticsearch : Ensure optional elasticsearch user is created with the correct id.] ***********************
skipping: [web1]

TASK [elastic.elasticsearch : Debian - Get installed elasticsearch version] *********************************************
```



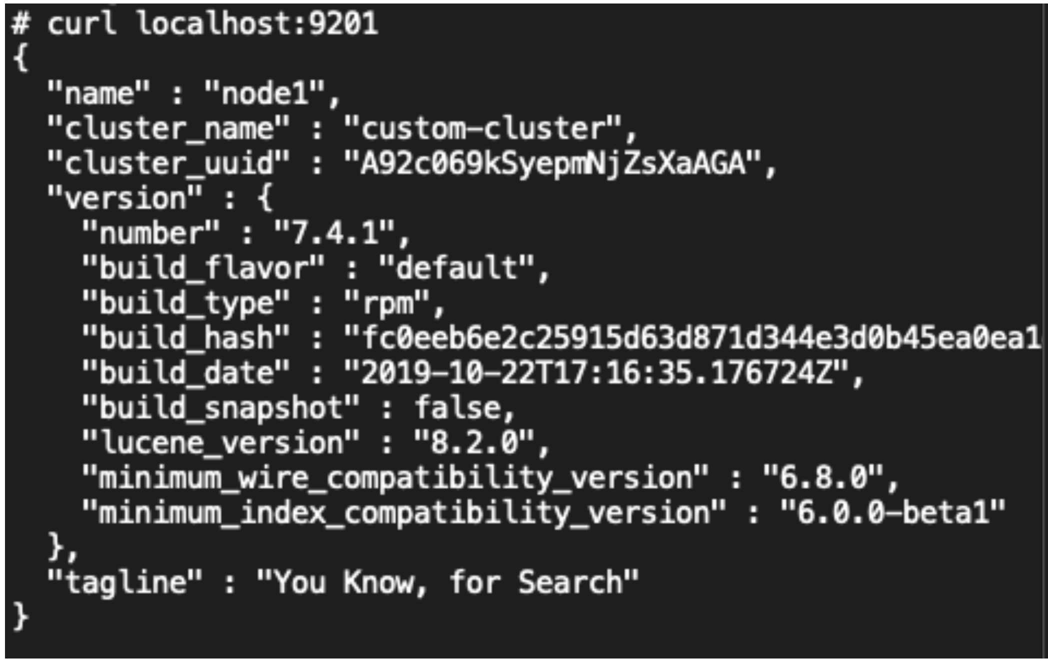
```
TASK [elastic.elasticsearch : RedHat - add Elasticsearch repo] ******************************
changed: [web1]

TASK [elastic.elasticsearch : RedHat - remove unused Elasticsearch repo] ********************
ok: [web1]

TASK [elastic.elasticsearch : RedHat - install yum-version-lock] ****************************
changed: [web1]

TASK [elastic.elasticsearch : RedHat - check if requested elasticsearch version lock exists]
ok: [web1]

TASK [elastic.elasticsearch : RedHat - lock elasticsearch version] **************************
skipping: [web1]

TASK [elastic.elasticsearch : RedHat - check if any elasticsearch version lock exists] ******
ok: [web1]
```



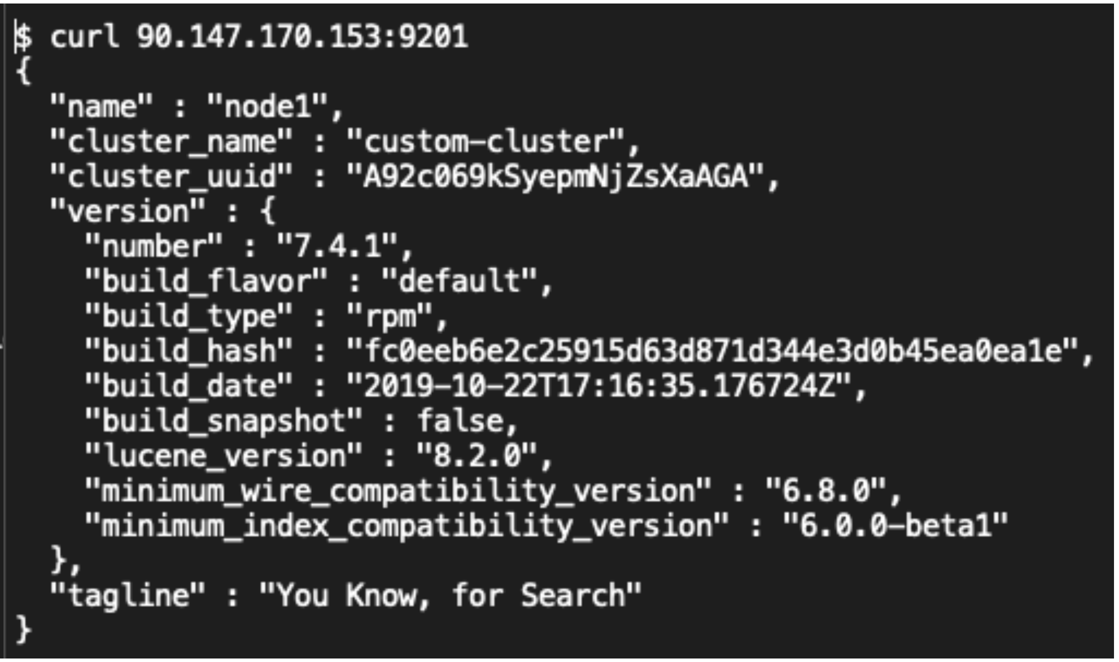
```
RUNNING HANDLER [elastic.elasticsearch : restart elasticsearch] *************************************************************
************************************
fatal: [web1]: FAILED! => {"changed": false, "msg": "Unable to start service elasticsearch: Job for elasticsearch.service
failed because the control process exited with error code. See \"systemctl status elasticsearch.service\" and \"journalctl
 -xe\" for details.\n"}

NO MORE HOSTS LEFT **********************************************************************************************************
************************************

PLAY RECAP ******************************************************************************************************************
************************************
web1                       : ok=24    changed=8    unreachable=0    failed=1    skipped=92    rescued=0    ignored=0
```



```
TASK [elastic.elasticsearch : Update Native Roles] ***************************************************

TASK [elastic.elasticsearch : ensure templates dir is created] ***************************************
skipping: [web1]

TASK [elastic.elasticsearch : Copy templates to elasticsearch] ***************************************
skipping: [web1] => (item=/Users/cristina/.ansible/roles/elastic.elasticsearch/files/templates-7.x/basic.json)

TASK [elastic.elasticsearch : Install templates] *****************************************************
skipping: [web1] => (item=/Users/cristina/.ansible/roles/elastic.elasticsearch/files/templates-7.x/basic.json)

PLAY RECAP *******************************************************************************************
web1                       : ok=26   changed=0    unreachable=0    failed=0    skipped=113  rescued=0    ignored=0
```



```
# curl localhost:9201
{
  "name" : "node1",
  "cluster_name" : "custom-cluster",
  "cluster_uuid" : "A92c069kSyepmNjZsXaAGA",
  "version" : {
    "number" : "7.4.1",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "fc0eeb6e2c25915d63d871d344e3d0b45ea0ea1
    "build_date" : "2019-10-22T17:16:35.176724Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

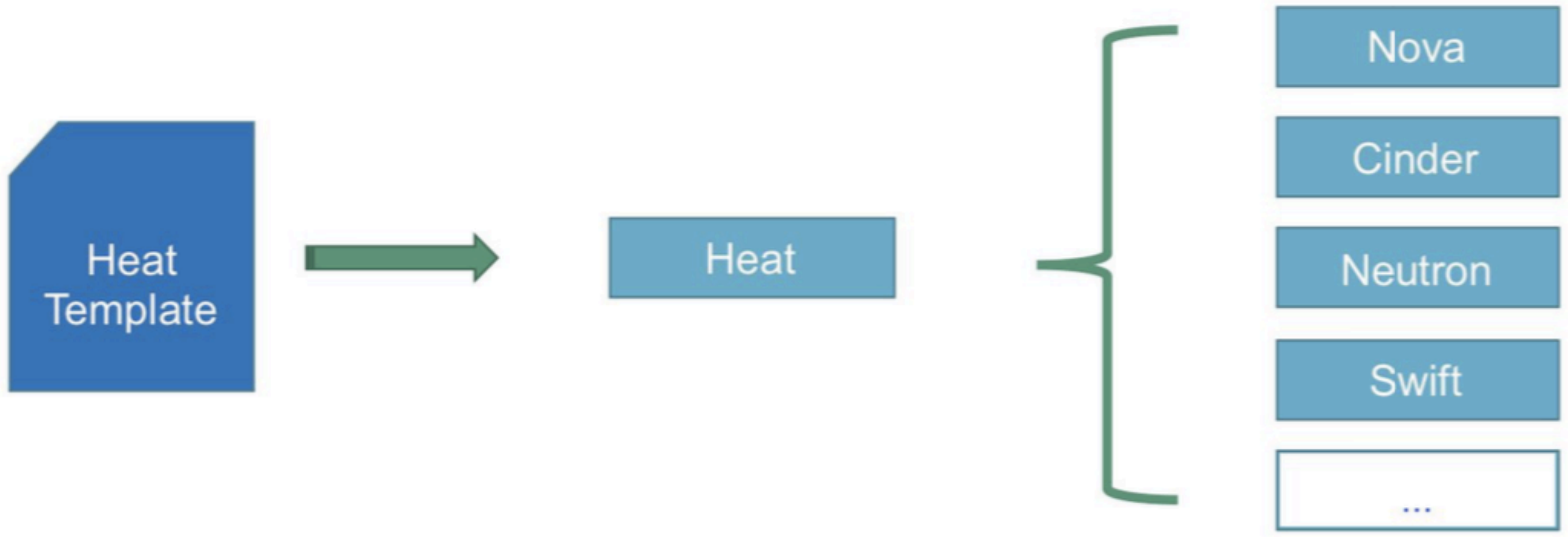


```
$ curl 90.147.170.153:9201
{
  "name" : "node1",
  "cluster_name" : "custom-cluster",
  "cluster_uuid" : "A92c069kSyepmNjZsXaAGA",
  "version" : {
    "number" : "7.4.1",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "fc0eeb6e2c25915d63d871d344e3d0b45ea0ea1e",
    "build_date" : "2019-10-22T17:16:35.176724Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Ansible – Use Cases

- Ansible and HEAT
- Ansible and TOSCA
- Ansible and Docker

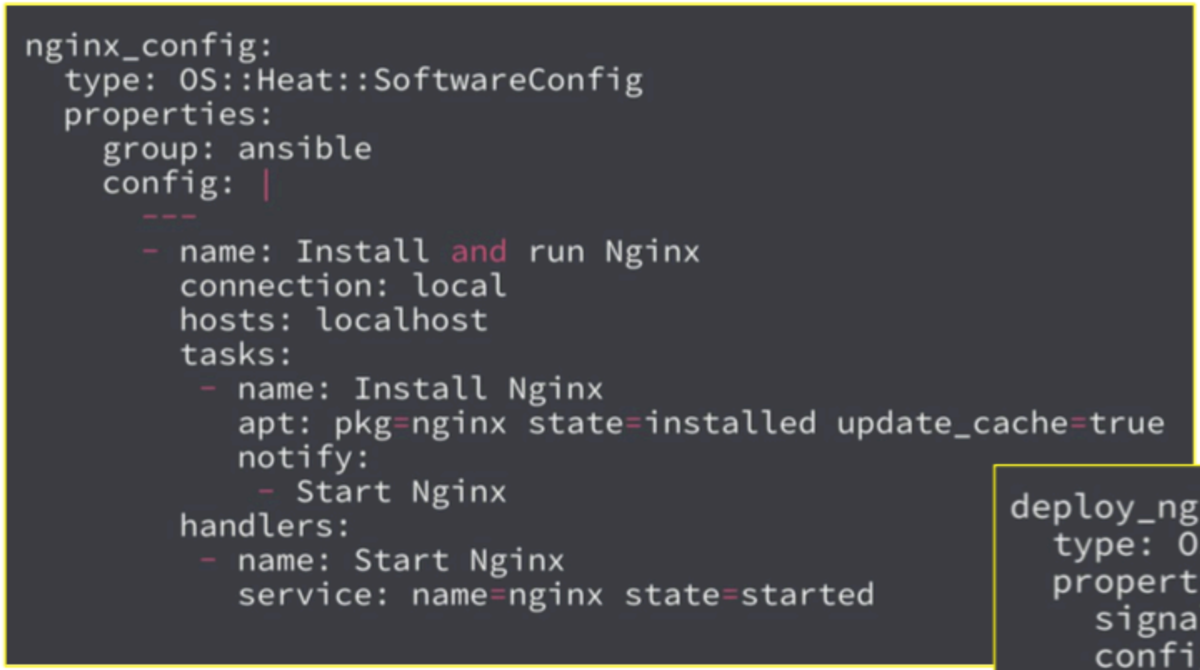
HEAT & HOT templates



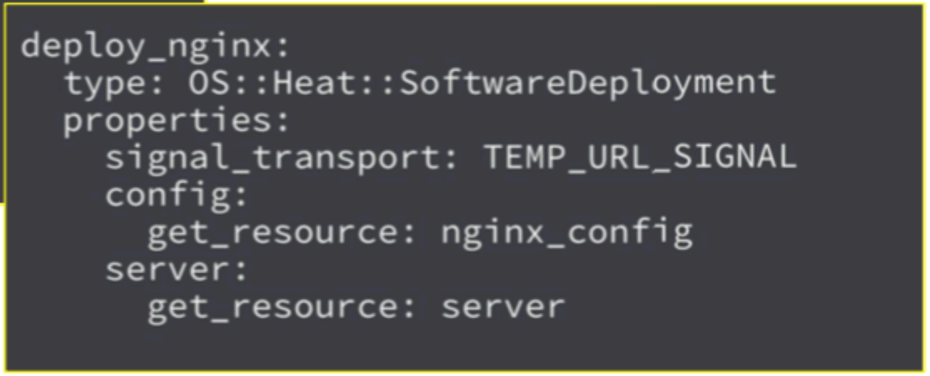
Software configuration

- There are **two main ways** for running SW configuration scripts in VMs:
  - User-data + cloudinit
    - Run once after instance first boot
  - Software Deployment resources
    - Run on every stack create/update
    - Send a signal back to Heat when finished
    - You can define dependencies among different scripts
    - Requires special services (hooks) running in the V

Example



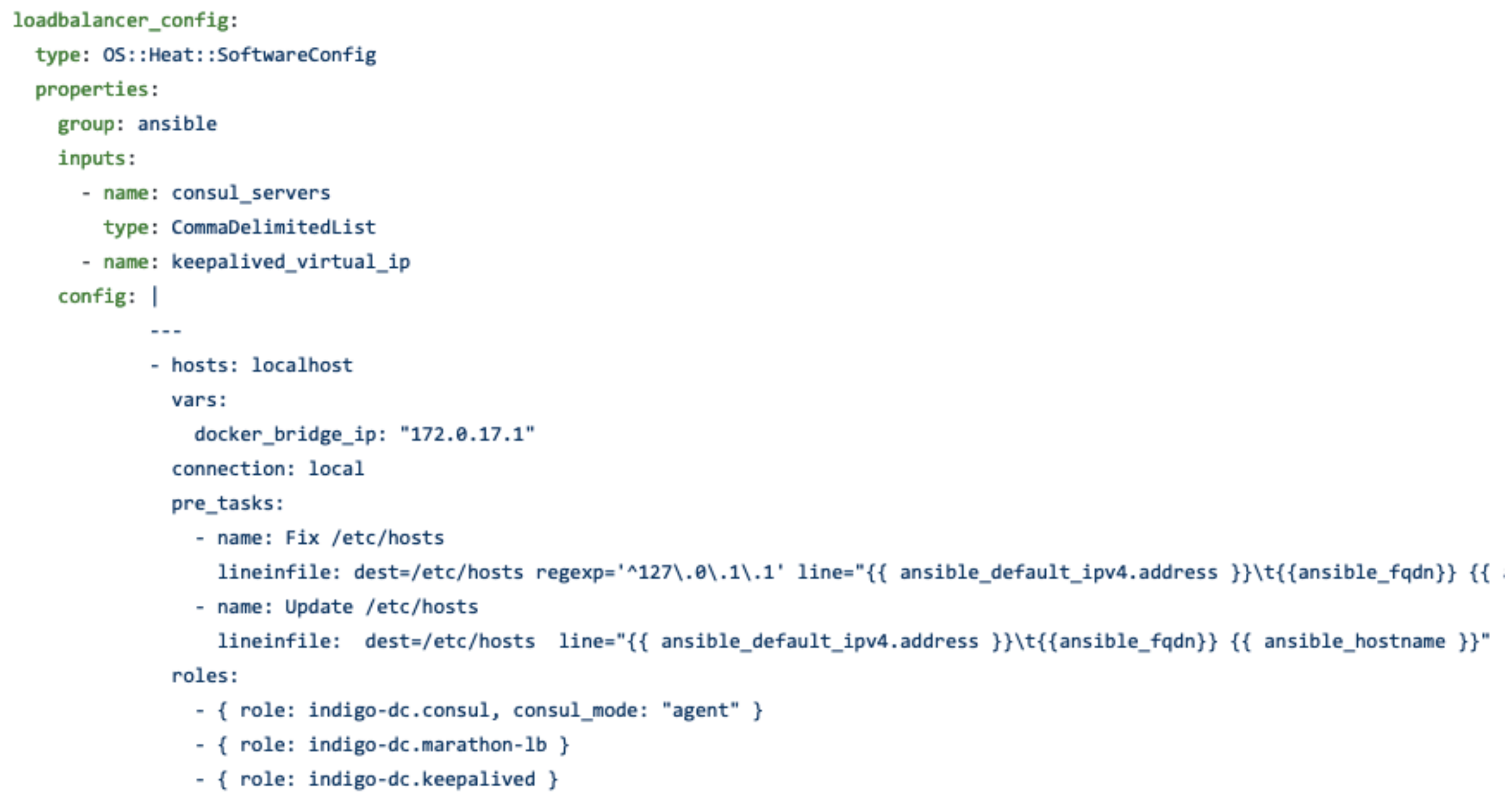
```
nginx_config:
  type: OS::Heat::SoftwareConfig
  properties:
    group: ansible
    config: |
      ---
      - name: Install and run Nginx
        connection: local
        hosts: localhost
        tasks:
          - name: Install Nginx
            apt: pkg=nginx state=installed update_cache=true
            notify:
              - Start Nginx
        handlers:
          - name: Start Nginx
            service: name=nginx state=started
```



```
deploy_nginx:
  type: OS::Heat::SoftwareDeployment
  properties:
    signal_transport: TEMP_URL_SIGNAL
    config:
      get_resource: nginx_config
    server:
      get_resource: server
```

INDIGO Mesos templates

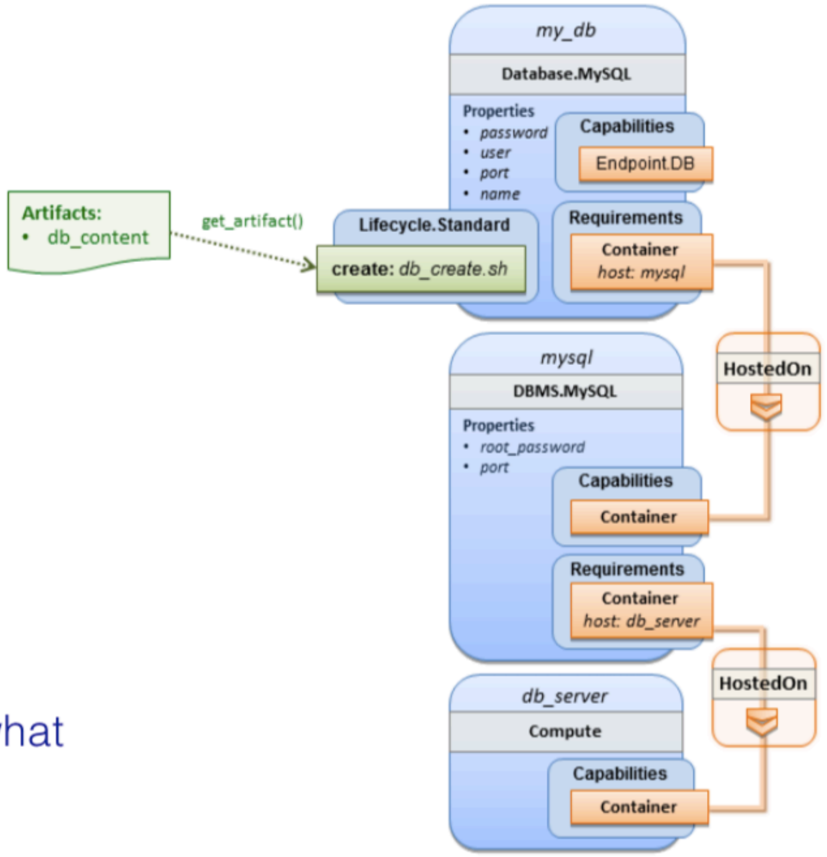
- https://github.com/indigo-dc/mesos-cluster/tree/master/deploy/openstack-heat



```
loadbalancer_config:
  type: OS::Heat::SoftwareConfig
  properties:
    group: ansible
    inputs:
      - name: consul_servers
        type: CommaDelimitedList
      - name: keepalived_virtual_ip
    config: |
            ---
            - hosts: localhost
              vars:
                docker_bridge_ip: "172.0.17.1"
              connection: local
              pre_tasks:
                - name: Fix /etc/hosts
                  lineinfile: dest=/etc/hosts regexp='^127\.0\.1\.1' line="{{ ansible_default_ipv4.address }}\t{{ansible_fqdn}} {{
                - name: Update /etc/hosts
                  lineinfile:  dest=/etc/hosts  line="{{ ansible_default_ipv4.address }}\t{{ansible_fqdn}} {{ ansible_hostname }}"
              roles:
                - { role: indigo-dc.consul, consul_mode: "agent" }
                - { role: indigo-dc.marathon-lb }
                - { role: indigo-dc.keepalived }
```

TOSCA – Infrastructure as Code

- **3 layers**
  - **Infrastructure**
    (Cloud or DC objects)
  - **Platform or Middleware**
    (App containers)
  - **Application modules, schemas and configurations**
- **Relationships** between components:
  - What's hosted on what or installed on what
  - What's connected to what



Tosca types

- Normative types
- Custom types
  - NDIGO custom types:
    - https://github.com/indigo-dc/tosca-types/blob/master/custom_types.yaml new types have been defined for elastic clusters, Marathon applications, Chronos jobs, etc

The artifacts are ansible playbooks that use indigo-dc ansible roles

Ansible & Docker

- Running ansible playbooks in the Dockerfile:

```
RUN apt-get update -y
RUN apt-get install software-properties-common -y
RUN apt-add-repository ppa:ansible/ansible
RUN apt-get update && \
    apt-get install -y ansible && \
    rm -rf /var/lib/apt/lists/*
RUN ansible-galaxy install indigo-dc.oneclient && \
    ansible-playbook
/etc/ansible/roles/indigo-dc.oneclient/tests/test.yml
```

The same ansible recipes can be used for configuring bare-metal, cloud servers and containers

Managing Docker containers with Ansible

Ansible provides some modules to manage containers:

docker_service: Consumes docker compose to start, shutdown and scale services

docker_container: Manage the life cycle of docker containers

docker_image: Build, load or pull an image, making the image available for creating containers. Also supports tagging an image into a repository and archiving an image to a .tar file.

docker_image_facts: inspect images, returning an array of inspection result

docker_login: Authenticate with a docker registry and add the credentials to your local Docker config file

docker_volume: Create/remove Docker volumes

ansible-container (NEW): a tool for building Docker images and orchestrating containers using Ansible playbooks

TOSCA Orchestration essentials

(many of the slides – courtesy of Marica Antonacci)

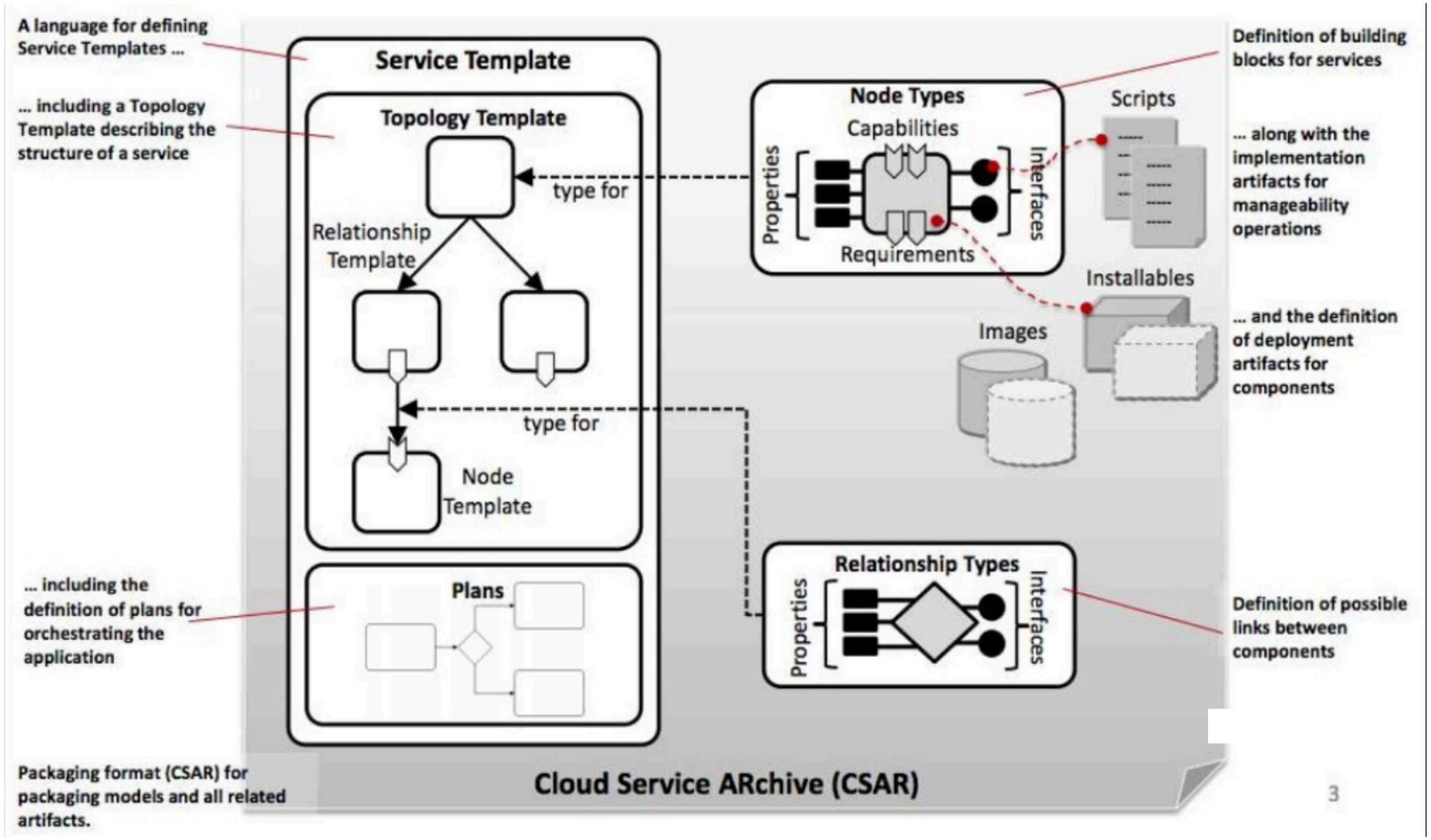
Outline

- What is TOSCA
    - Goals, topology, composition, portability, lifecycle
- INDIGO PaaS Orchestrator
- Hands-on

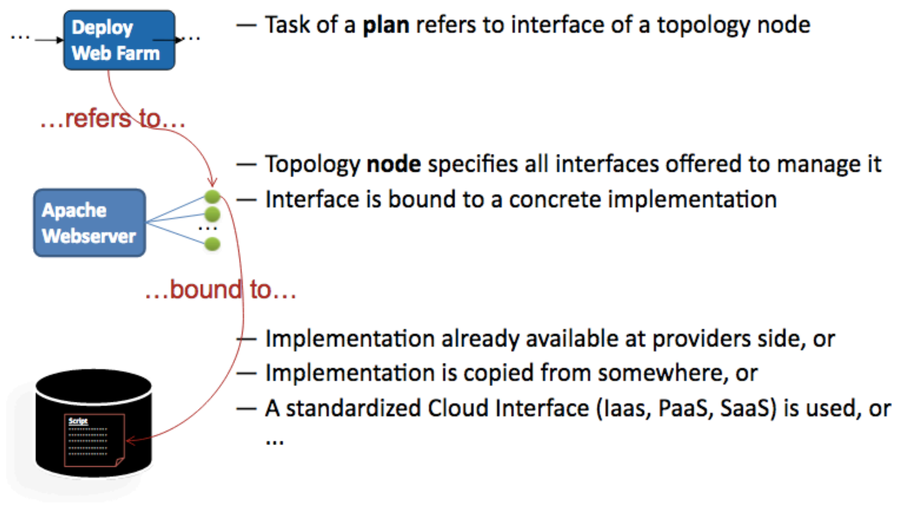
TOSCA

- **T**opology and **O**rchestration **S**pecification for **C**loud **A**pplications Standardizes the language to describe
  - The structure of an ITService (its topology model)
  - How to orchestrate operational behavior (plans such as build, deploy, patch, shutdown, etc.)
    - Leveraging the BPMN standard
  - Declarative model that spans applications, virtual and physical infrastructure
- **Main Goals**
  - Automated Application Deployment and Management
  - Portability of Application Descriptions and their Management
  - Interoperability and Reusability of Components

TOSCA in a nutshell


A language for defining Service Templates ...
... including a Topology Template describing the structure of a service
... including the definition of plans for orchestrating the application
Packaging format (CSAR) for packaging models and all related artifacts.
Service Template
Topology Template
type for
Relationship Template
type for
Node Template
Plans
Node Types
Capabilities
Properties
Interfaces
Requirements
Scripts
Installables
Images
Relationship Types
Properties
Interfaces
Cloud Service ARchive (CSAR)
Definition of building blocks for services
... along with the implementation artifacts for manageability operations
... and the definition of deployment artifacts for components
Definition of possible links between components
3

Vision



— Task of a **plan** refers to interface of a topology node

— Topology **node** specifies all interfaces offered to manage it

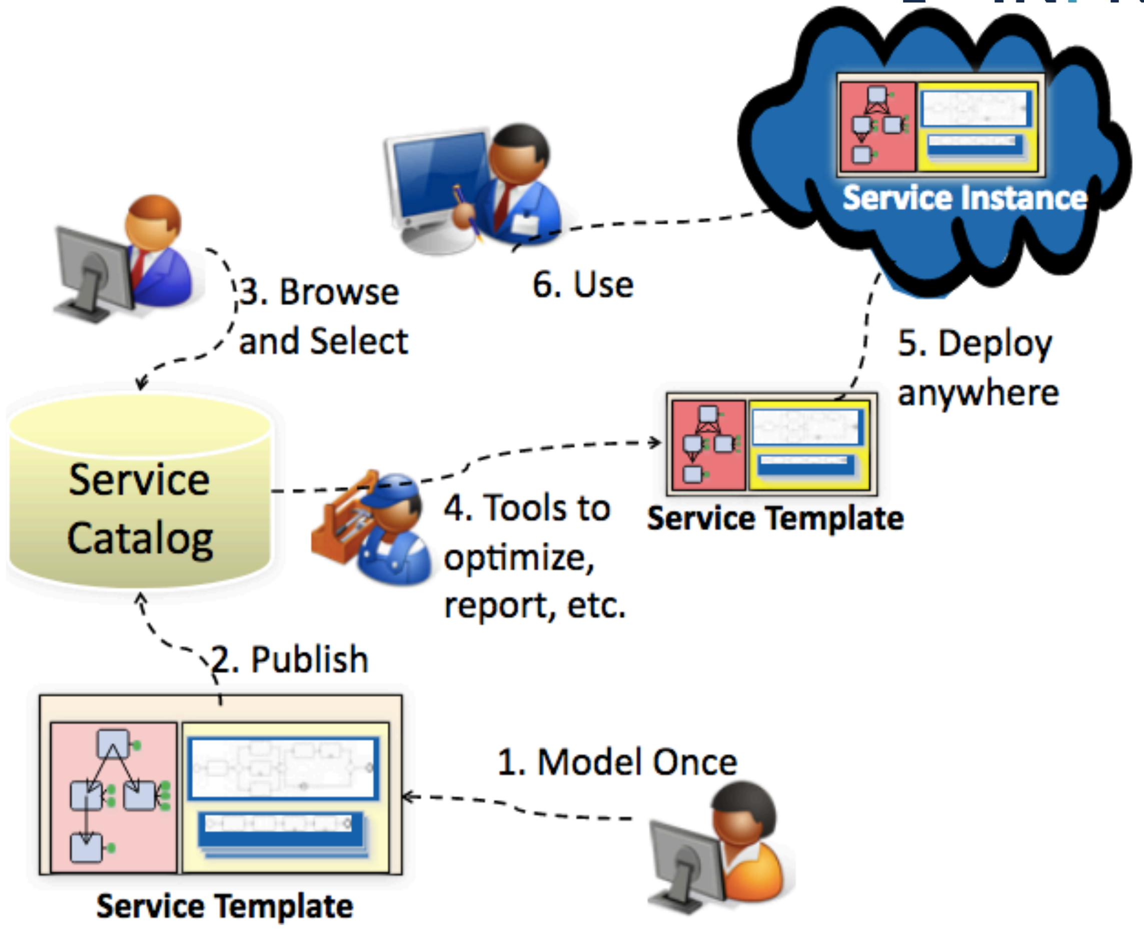
— Interface is bound to a concrete implementation

— Implementation already available at providers side, or

— Implementation is copied from somewhere, or

— A standardized Cloud Interface (Iaas, PaaS, SaaS) is used, or

…



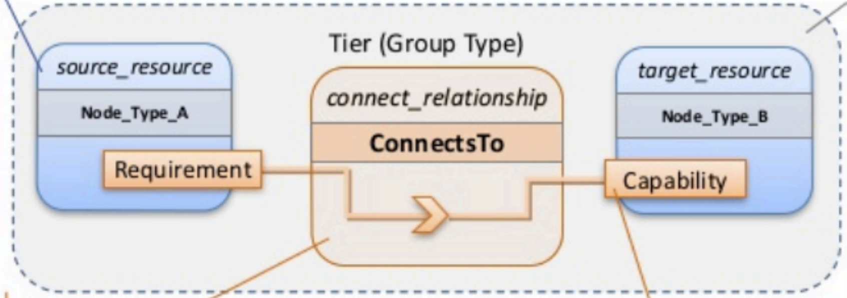
TOSCA Topology

TOSCA is used first and foremost to describe the topology of the **deployment view** for cloud applications and services

- ✓ ***Node templates*** *to describe components in the topology structure*
- ✓ ***Relationship templates*** *to describe connections, dependencies, deployment ordering*

Nodes *- are the resources or components that will be materialized or consumed in the deployment topology*

Groups
Create Logical, Management or Policy groups (1 or more nodes)

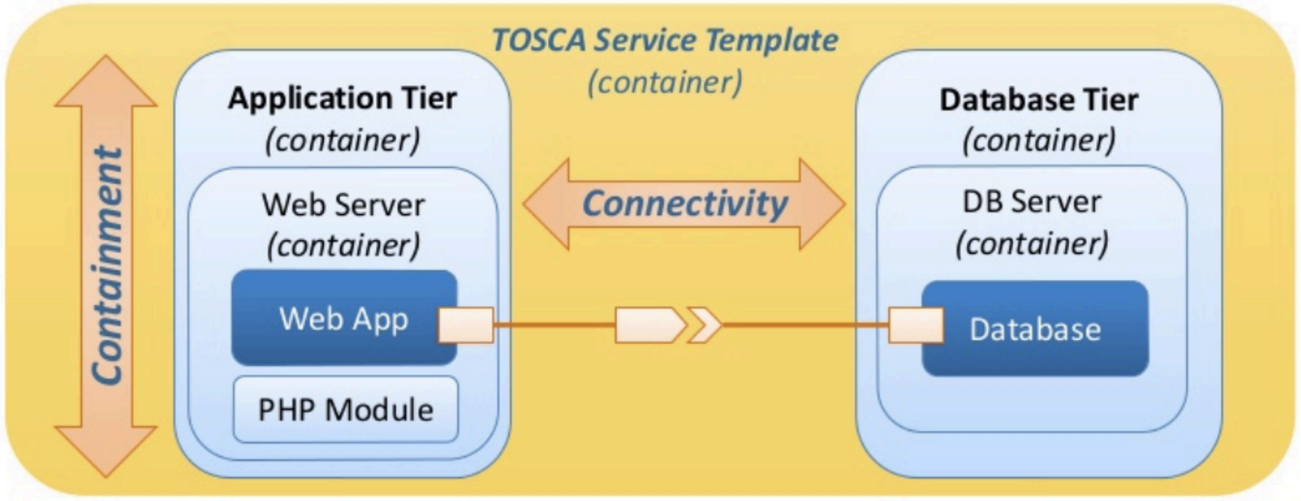


Relationships
express the dependencies between the nodes (not the traffic flow)

Requirement - Capability
Relationships can be customized to match specific source requirements to target capabilities

Topology (2)

Example: a simple, 2-Tier Cloud application expressed in a TOSCA Service Template

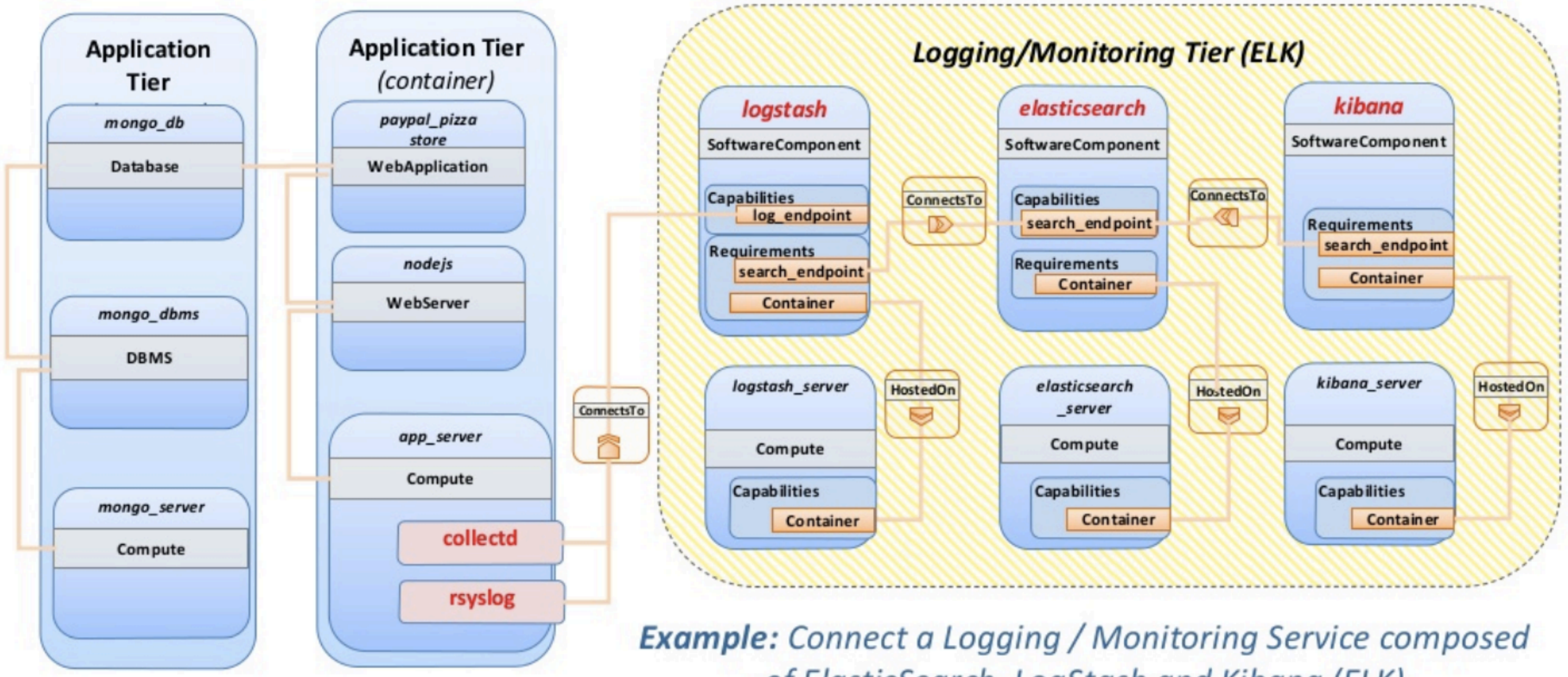


Service Templates provide the "container" to exchange and reuse topologies:

- ***Reusable models*** extend investments by making it easy to *compose* more valuable and complex apps from existing apps
- Determines dependency boundaries to **maximize parallelism** of deployments
- Models (dependencies) can be ***validated by automation*** to ensure application-aware, policy-aligned configuration, deployment and operational semantics

Composition

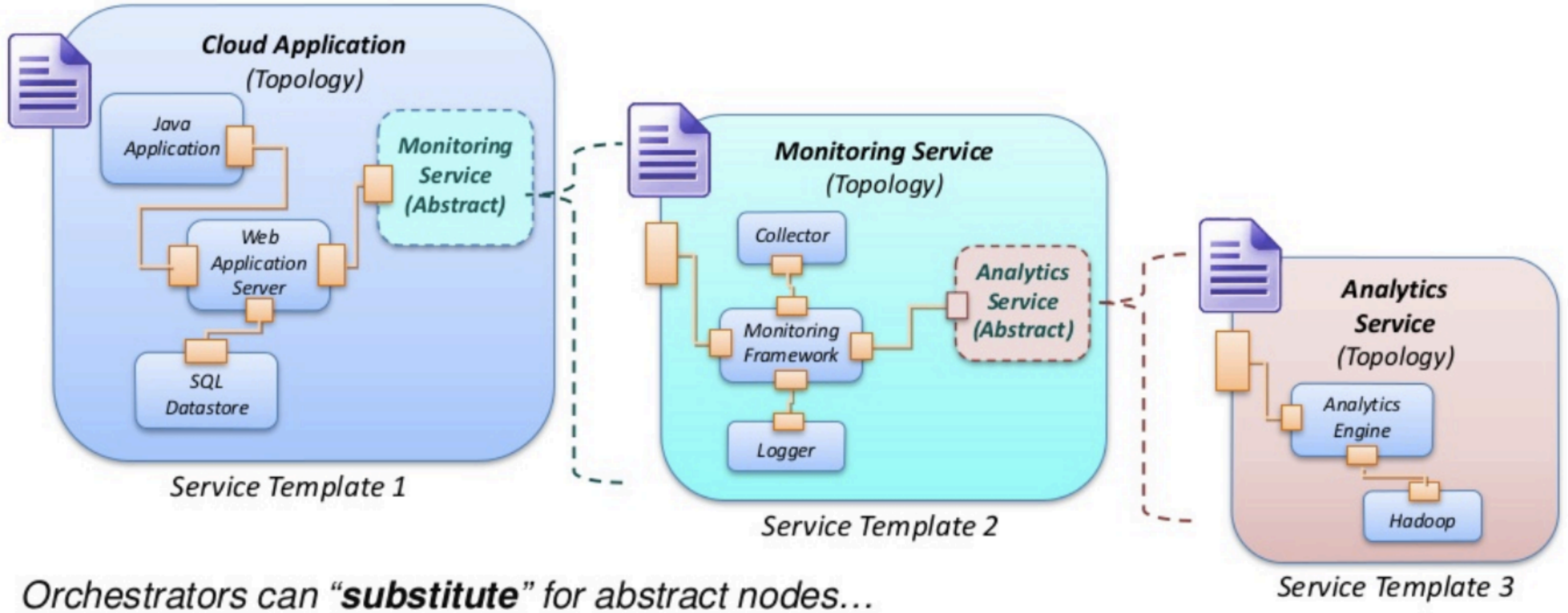
Enabling the description of complex, multi-tier (hybrid) Cloud applications



Example: *Connect a Logging / Monitoring Service composed of ElasticSearch, LogStash and Kibana (ELK)*

Composition (2)

Abstract nodes in one TOSCA topology can be substituted with another topology

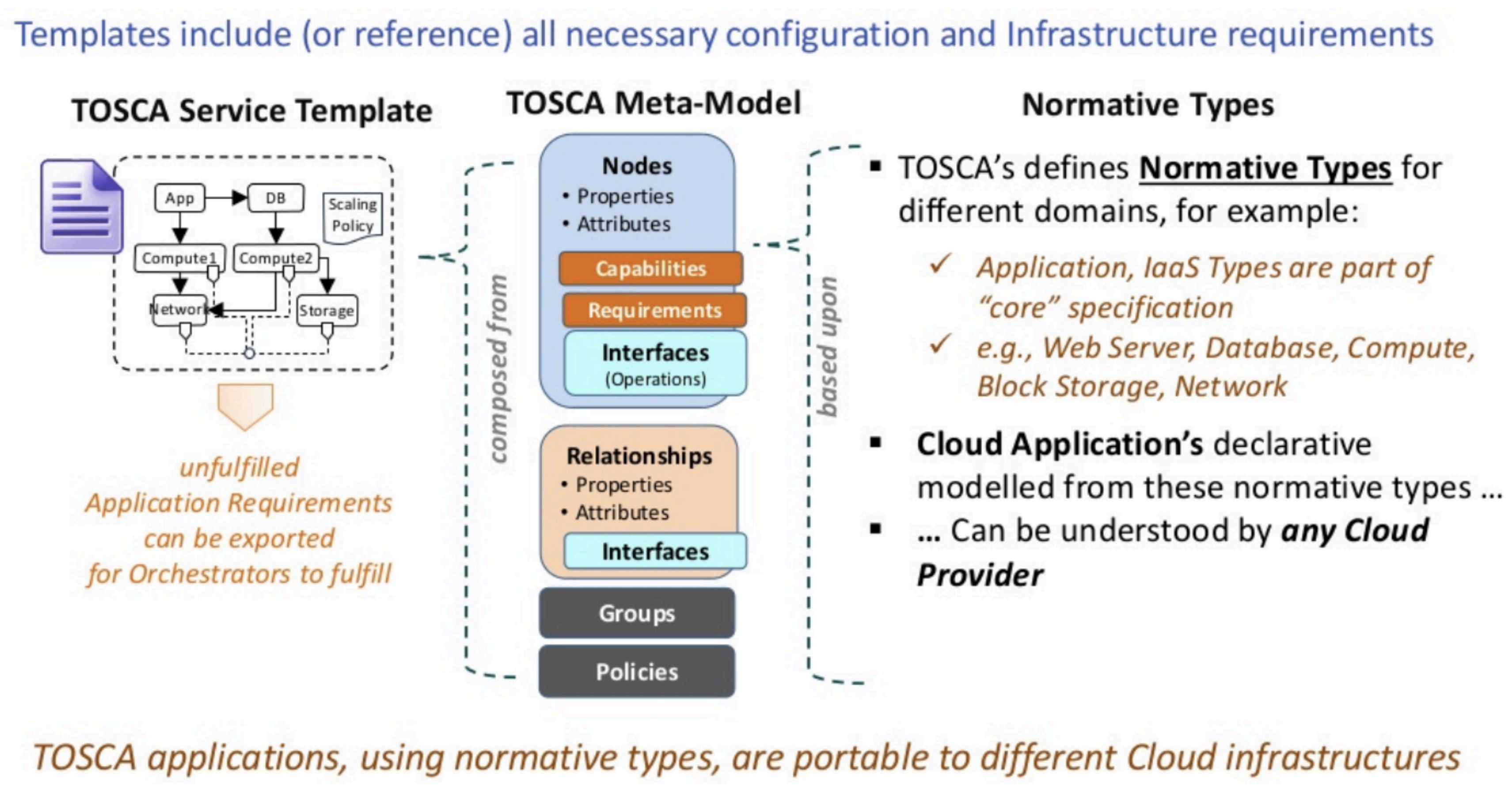


*Orchestrators can "**substitute**" for abstract nodes…*

*… as long as all declared "**requirements**" are met:*

- ***Monitoring Service** can be **substituted** in **Cloud Application***
- ***Analytics Service** can be **substituted** in **Monitoring Service***

Portability



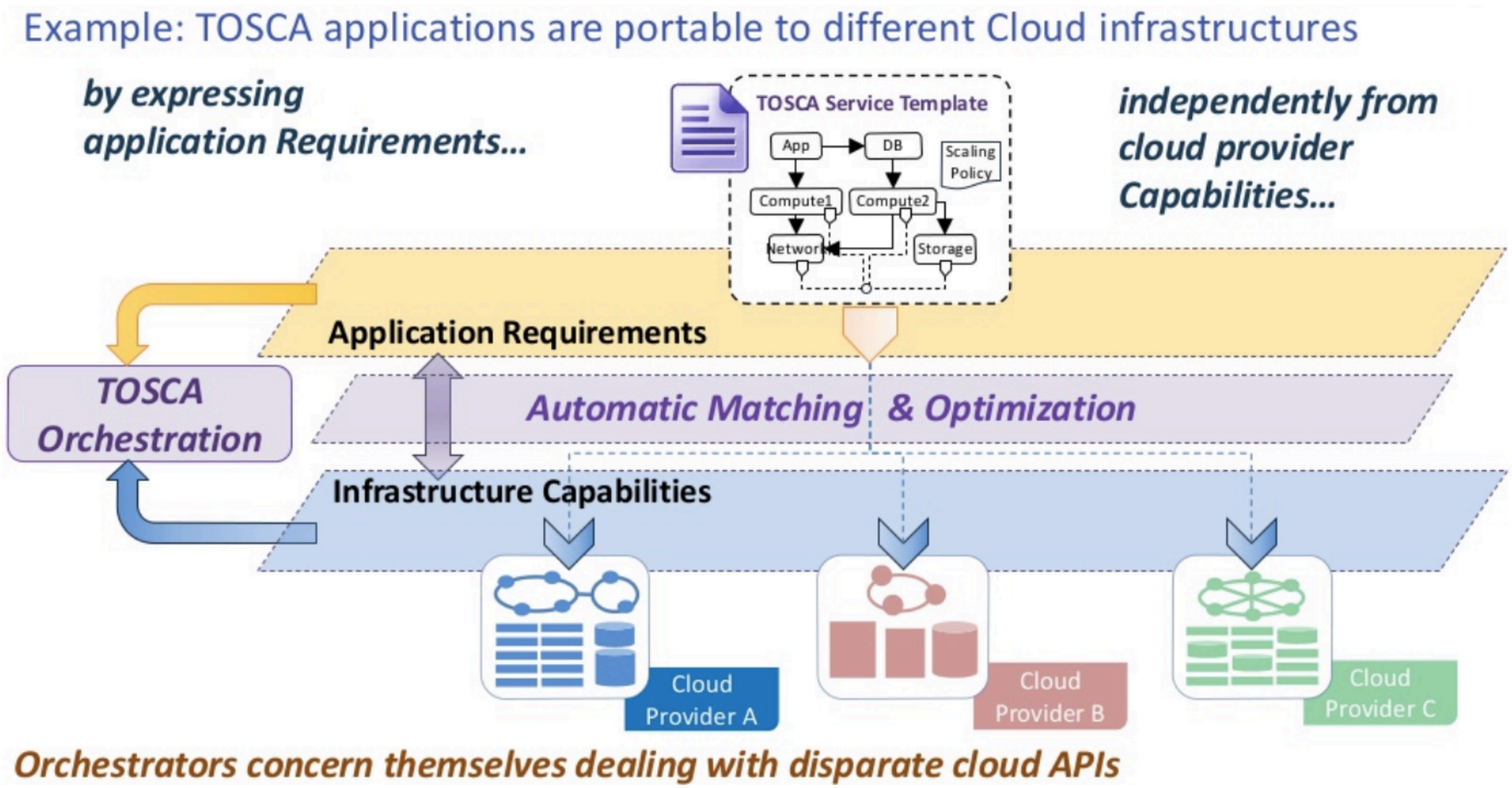
Templates include (or reference) all necessary configuration and Infrastructure requirements

Normative Types

- TOSCA's defines **Normative Types** for different domains, for example:
  - ✓ *Application, IaaS Types are part of "core" specification*
  - ✓ *e.g., Web Server, Database, Compute, Block Storage, Network*
- **Cloud Application's** declarative modelled from these normative types ...
- **...** Can be understood by ***any Cloud Provider***

TOSCA applications, using normative types, are portable to different Cloud infrastructures

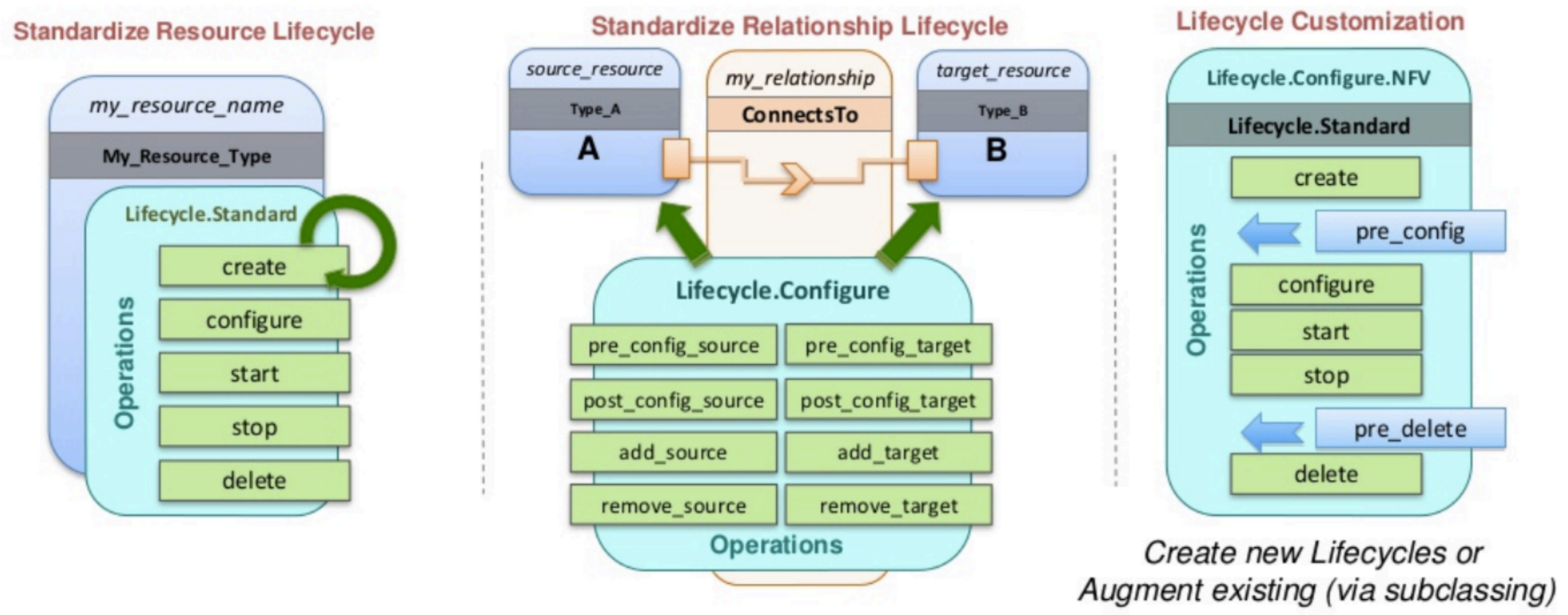
Portability (2)


Example: TOSCA applications are portable to different Cloud infrastructures
by expressing application Requirements…
TOSCA Service Template
App
DB
Scaling Policy
Compute1
Compute2
Network
Storage
independently from cloud provider Capabilities…
Application Requirements
TOSCA Orchestration
Automatic Matching & Optimization
Infrastructure Capabilities
Cloud Provider A
Cloud Provider B
Cloud Provider C
Orchestrators concern themselves dealing with disparate cloud APIs

Lifecycle – State based Orchestration

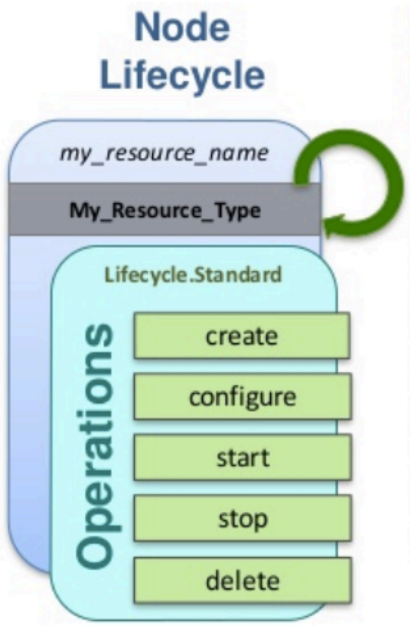
TOSCA models have a consistent view of state-based lifecycle

- ✓ have **Operations** (implementations) that can be sequenced against state of any dependent resources
- ✓ *fits into any **Management Framework** or **Access Control System***



Create new Lifecycles or Augment existing (via subclassing)

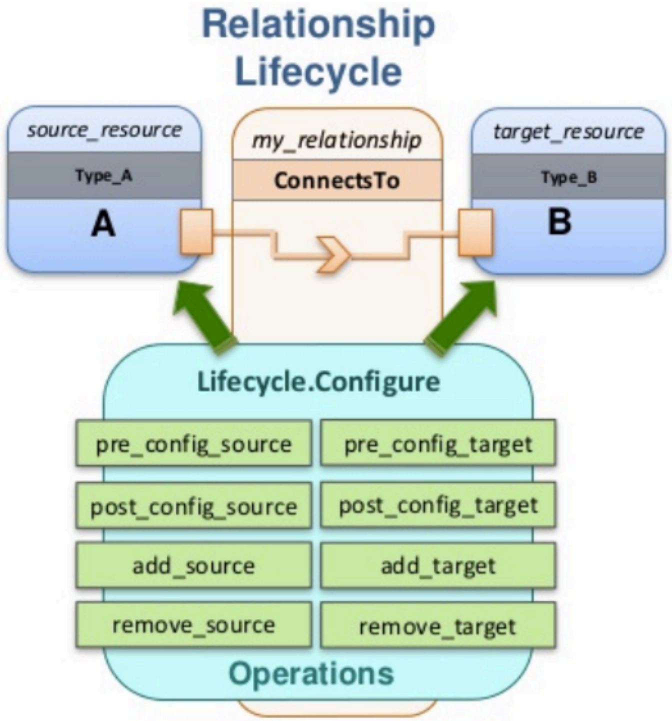
Lifecycle (2)



__Nodes__ have their own __Lifecycle Operations__ which are invoked in order to achieve a __target state__

***__Relationships__** also have their own __Lifecycle Operations__ to configure or allocate and de-configure or deallocate __Node__ related resources*

***__Implementations__** (e.g., imperative scripts) can be bound to operations.*



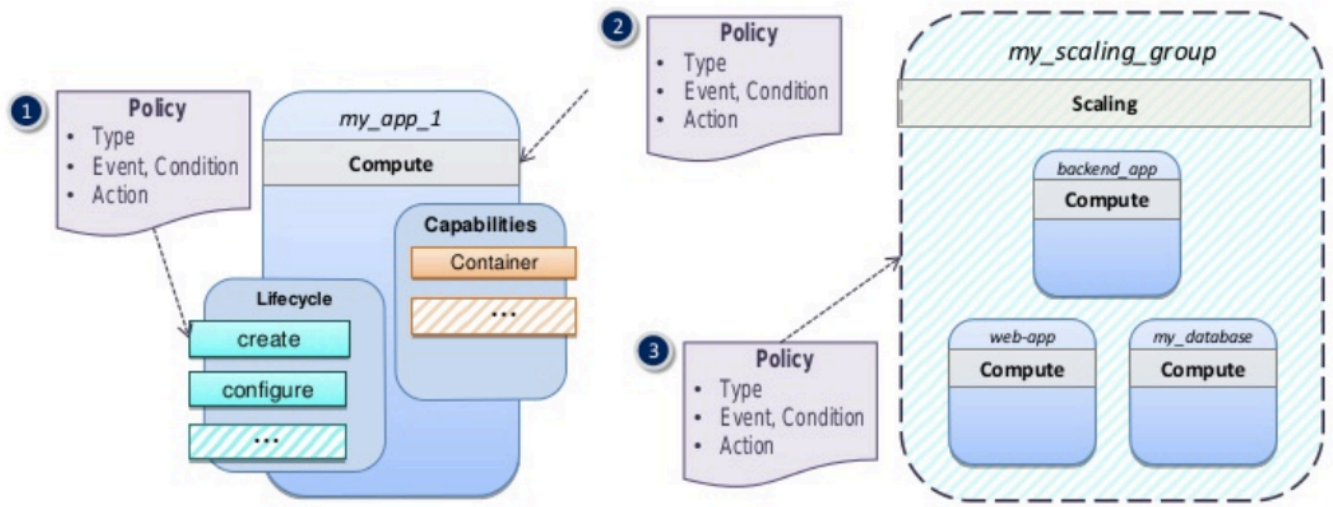
The __Orchestrator__ moves the nodes through their __Lifecycle States__ by executing their __Lifecycle Operations__ in __topological order__

- *Orchestrators can work to deploy nodes in parallel based upon node relationships*

Policy - Operational Policies

v1.0 includes the groundwork for Placement (Affinity), Scaling and Performance Policies

- *Orchestrators can evaluate **Conditions** based on **Events** that trigger Automatic or Imperative **Actions***



Policies can be declared independently and ttached to various points in your models

1. *That can be attached to **Interfaces** or specific **Operations**,*
2. ***Nodes** and*
3. ***Groups** of Nodes*

"Policies are non-functional Requirements independent of nodes"

TOSCA Eco-System

ARIA

Multi-Cloud Orchestration
(Amazon, Azure, VMware, OpenStack)
Open Sourced from Cloudify
http://ariatosca.org//

Cloudify

Service Orchestration & Management
http://getcloudify.org/

alien4cloud

Topology, Type & LCM Design
http://alien4cloud.github.io/

CERN INNO INDIGO POLICY

Data/computing platform targeted at scientific communities
http://information-technology.web.cern.ch/about/projects/eu/indigo-datacloud

seaclouds

Open, Multi-Cloud Management
www.seaclouds-project.eu/media.html

OPNFV Parser

Deployment Template Translation
https://wiki.opnfv.org/display/parser/Parser

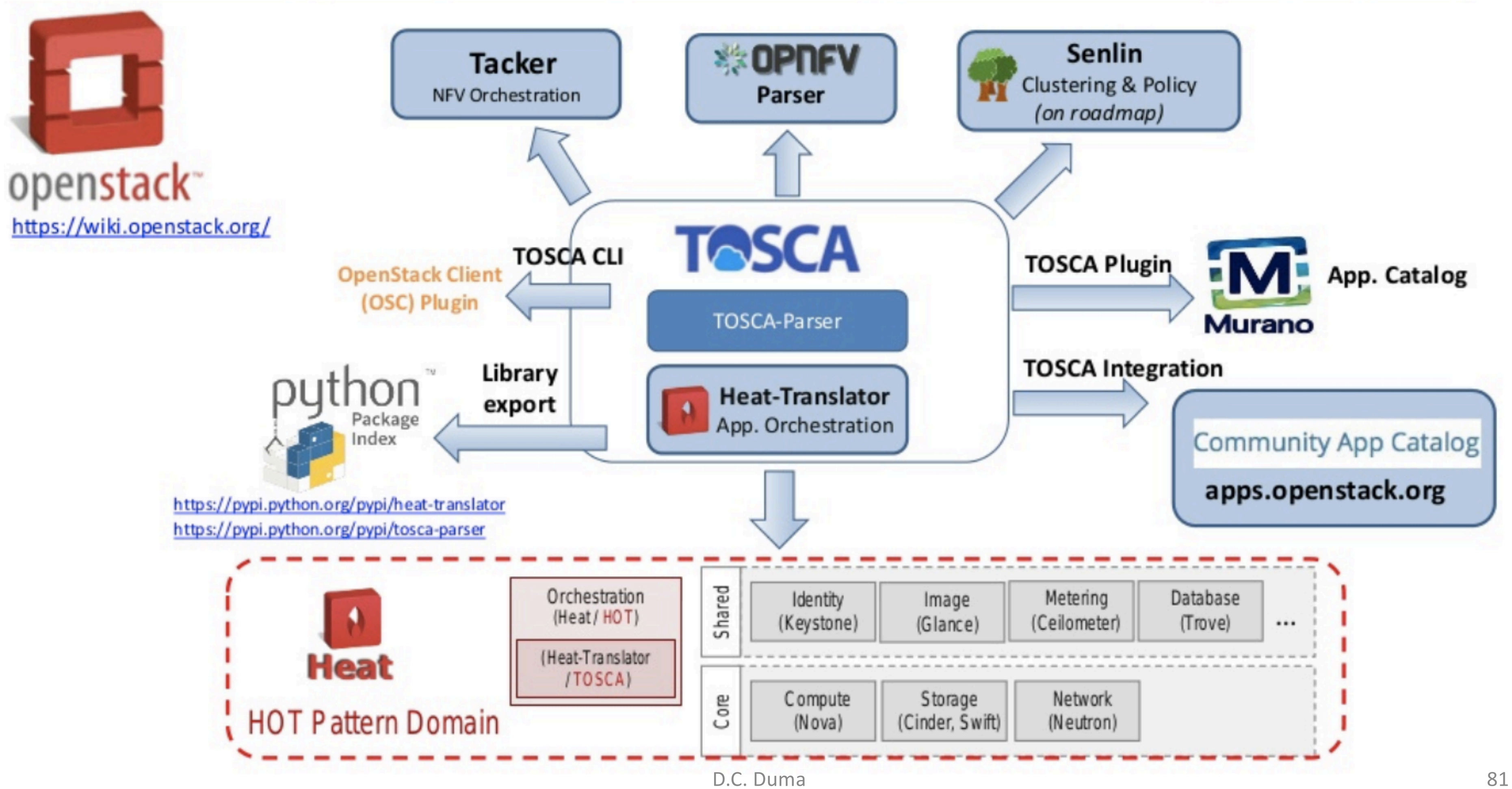
***Note**: ETSI NFV ack. TOSCA can be used as an input model/format*

openstack

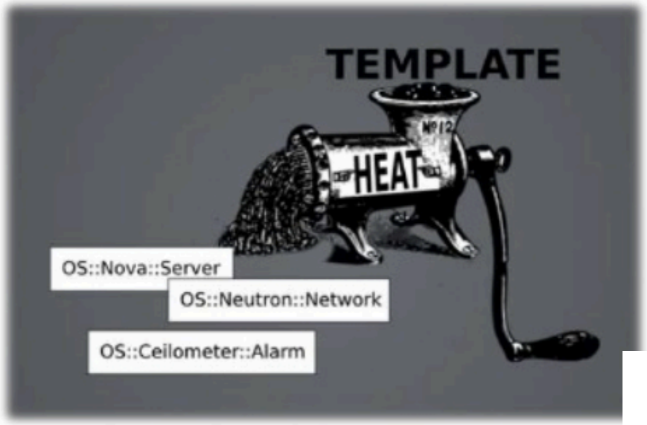
Heat-Translator
(IaaS, App Orchestration)
Tacker
(Network Function Orchestration)
https://wiki.openstack.org/

Ref: https://wiki.oasis-open.org/tosca/TOSCA-implementations

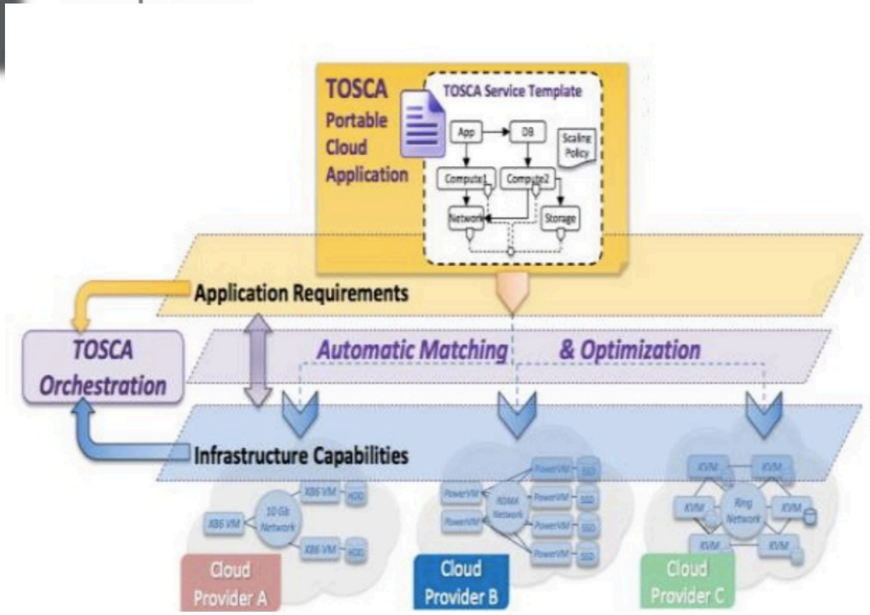
TOSCA – Openstack Integration



HEAT vs TOSCA

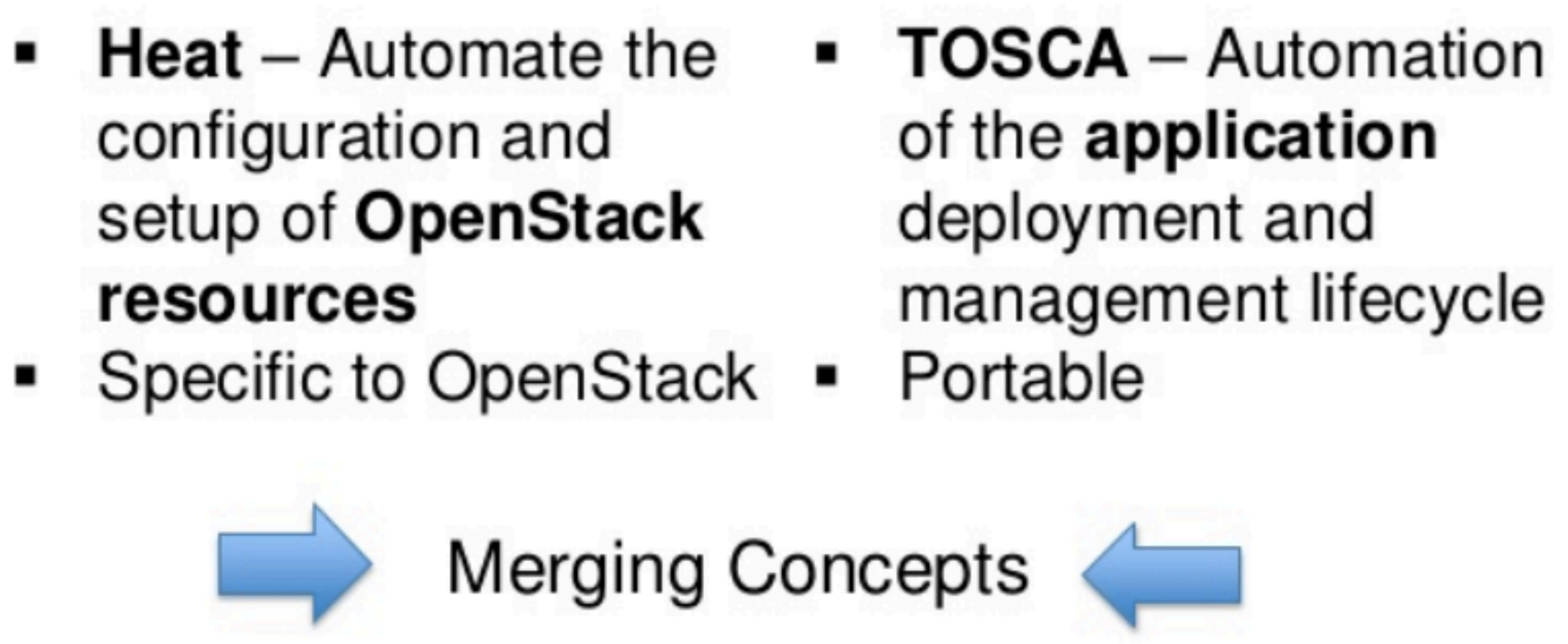


Heat provides a mechanism for orchestrating OpenStack resources through the use of modular templates.



TOSCA defines the interoperable description of applications; including their components, relationships, dependencies, requirements, and capabilities….

Comparing TOSCA & HEAT



- **Heat** – Automate the configuration and setup of **OpenStack resources**
- Specific to OpenStack

- **TOSCA** – Automation of the **application** deployment and management lifecycle
- Portable

Merging Concepts

TOSCA topology

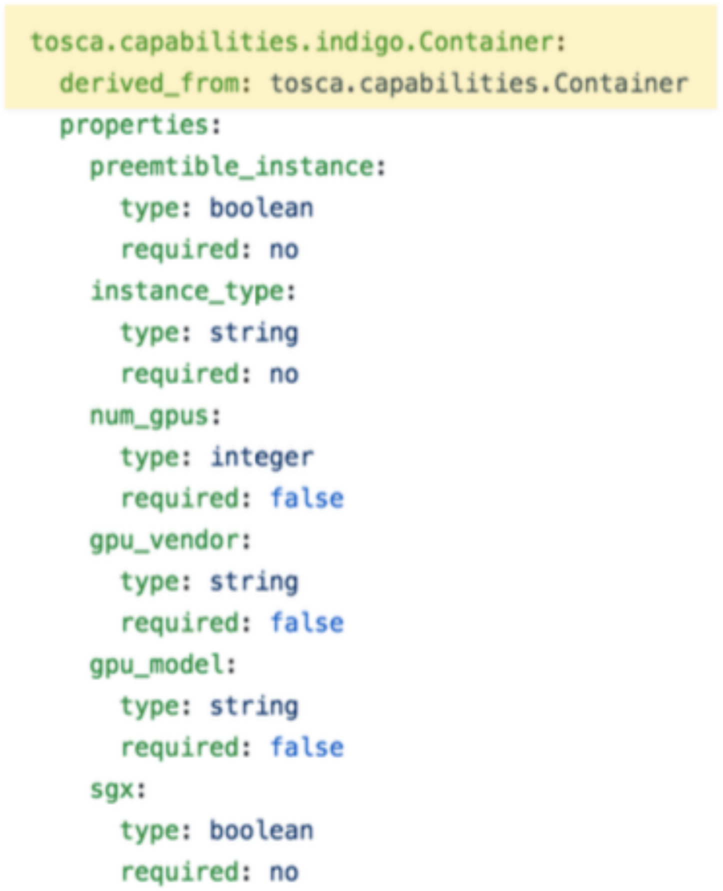
- Components in the topology are called ***Nodes***
- Each Node has a ***Type*** (e.g. Host, DB, Web server).
  - The Type is abstract and hence portable
  - The Type defines ***Properties*** and ***Interfaces***
- An ***Interface*** is a set of hooks (named ***Operations***)
- Nodes are connected to one another using ***Relationships***
- Both Node Types and Relationship Types can be **derived**

Normative Types

- The **TOSCA Simple Profile in YAML** specifies a rendering of TOSCA to provide a more accessible syntax and a more concise expressiveness of the TOSCA DSL
- It provides a rich set of **base** types (node types and relationship types): e.g. '**Compute**' node type
- Some **non-normative types** are provided as well but implementations of this specification are not required to support these types for conformance.

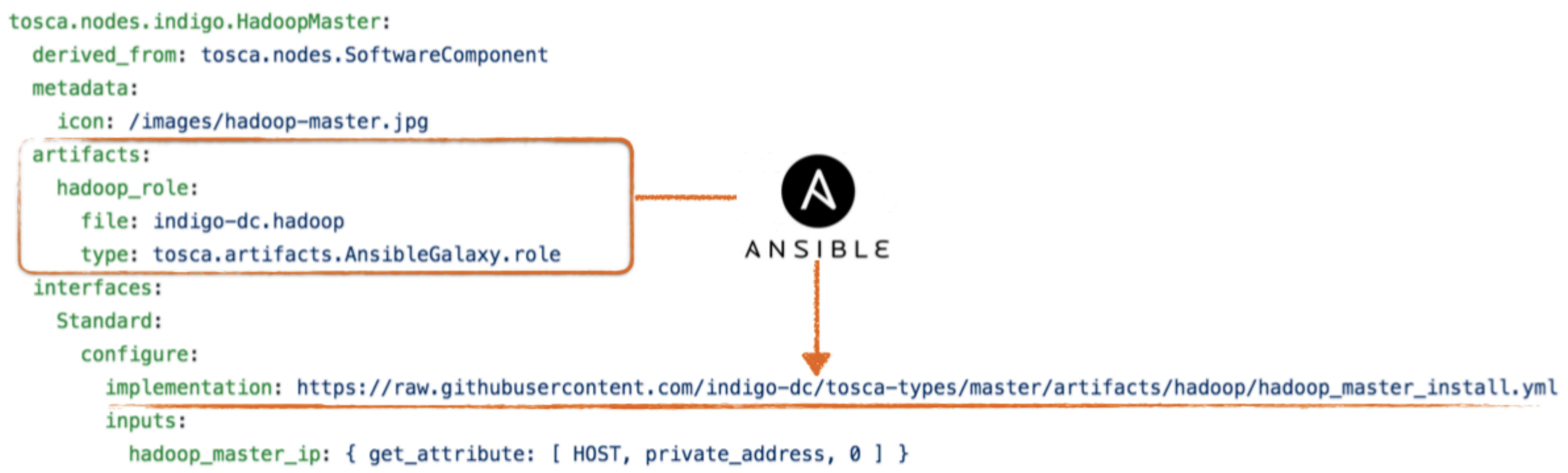
Custom Types

- TOSCA is highly versatile
  - ➢ One can define custom types for nodes, relationships, and capabilities —> can be used in different domains
  - ➢ Indigo custom types
    - ➢ https://github.com/indigo-dc/tosca-types



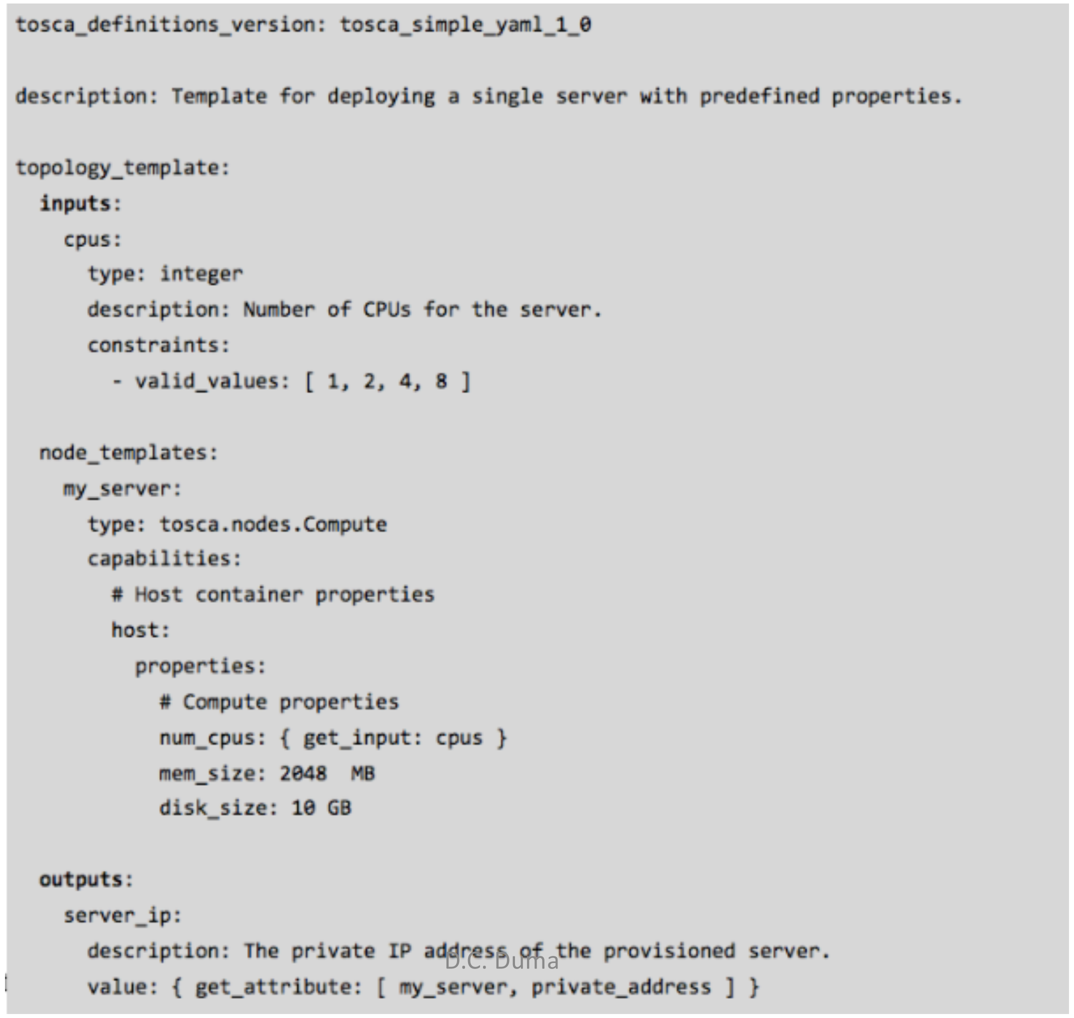
```
tosca.capabilities.indigo.Container:
  derived_from: tosca.capabilities.Container
  properties:
    preemtible_instance:
      type: boolean
      required: no
    instance_type:
      type: string
      required: no
    num_gpus:
      type: integer
      required: false
    gpu_vendor:
      type: string
      required: false
    gpu_model:
      type: string
      required: false
    sgx:
      type: boolean
      required: no
```

INDIGO-DC custom type example



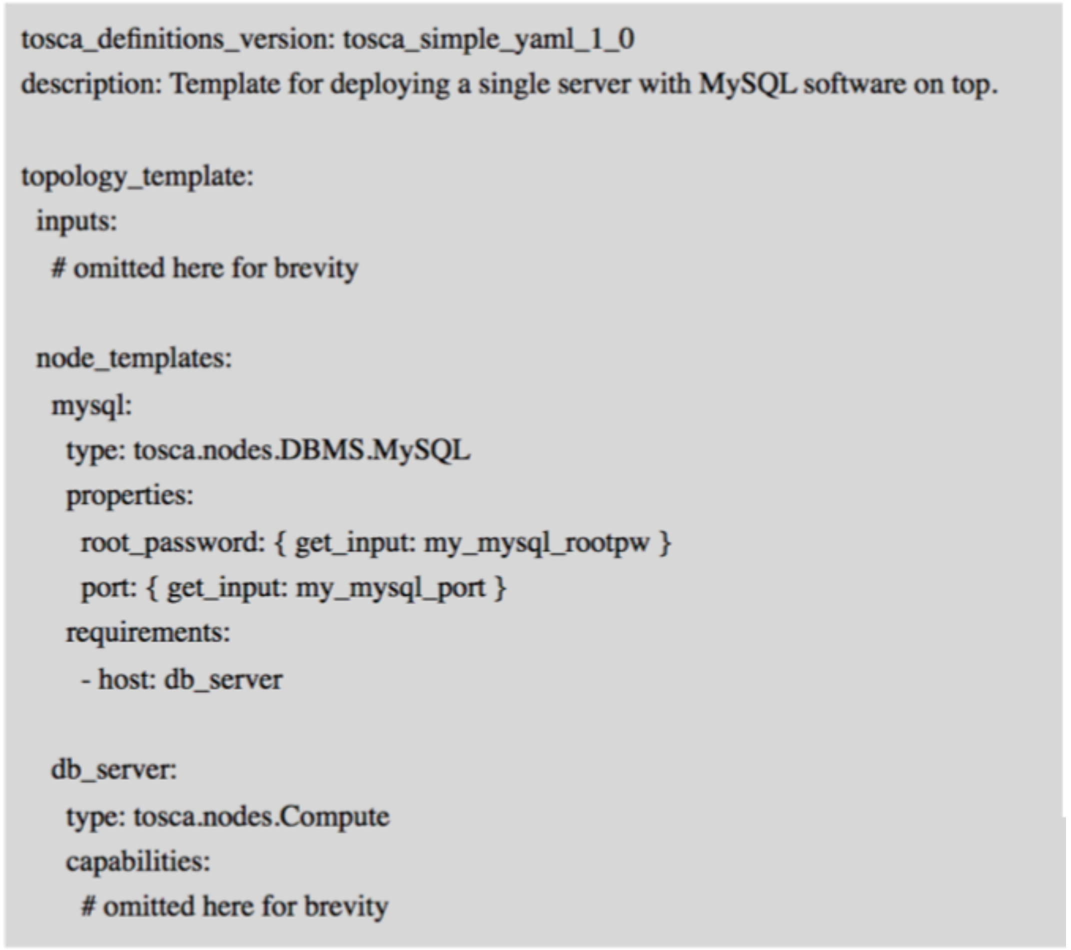
```
tosca.nodes.indigo.HadoopMaster:
  derived_from: tosca.nodes.SoftwareComponent
  metadata:
    icon: /images/hadoop-master.jpg
  artifacts:
    hadoop_role:
      file: indigo-dc.hadoop
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      configure:
        implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/artifacts/hadoop/hadoop_master_install.yml
        inputs:
          hadoop_master_ip: { get_attribute: [ HOST, private_address, 0 ] }
```

Topology Template Example 1

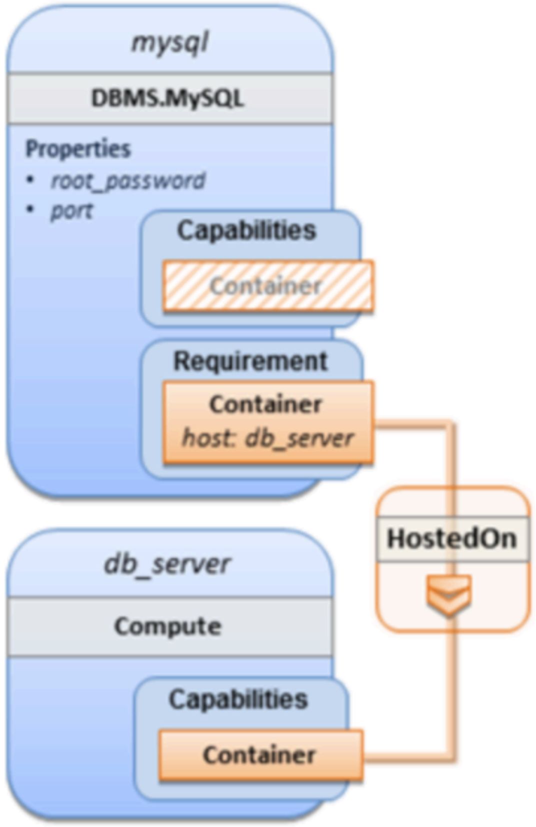


```
tosca_definitions_version: tosca_simple_yaml_1_0

description: Template for deploying a single server with predefined properties.

topology_template:
  inputs:
    cpus:
      type: integer
      description: Number of CPUs for the server.
      constraints:
        - valid_values: [ 1, 2, 4, 8 ]

  node_templates:
    my_server:
      type: tosca.nodes.Compute
      capabilities:
        # Host container properties
        host:
          properties:
            # Compute properties
            num_cpus: { get_input: cpus }
            mem_size: 2048  MB
            disk_size: 10 GB

  outputs:
    server_ip:
      description: The private IP address of the provisioned server.
      value: { get_attribute: [ my_server, private_address ] }
```

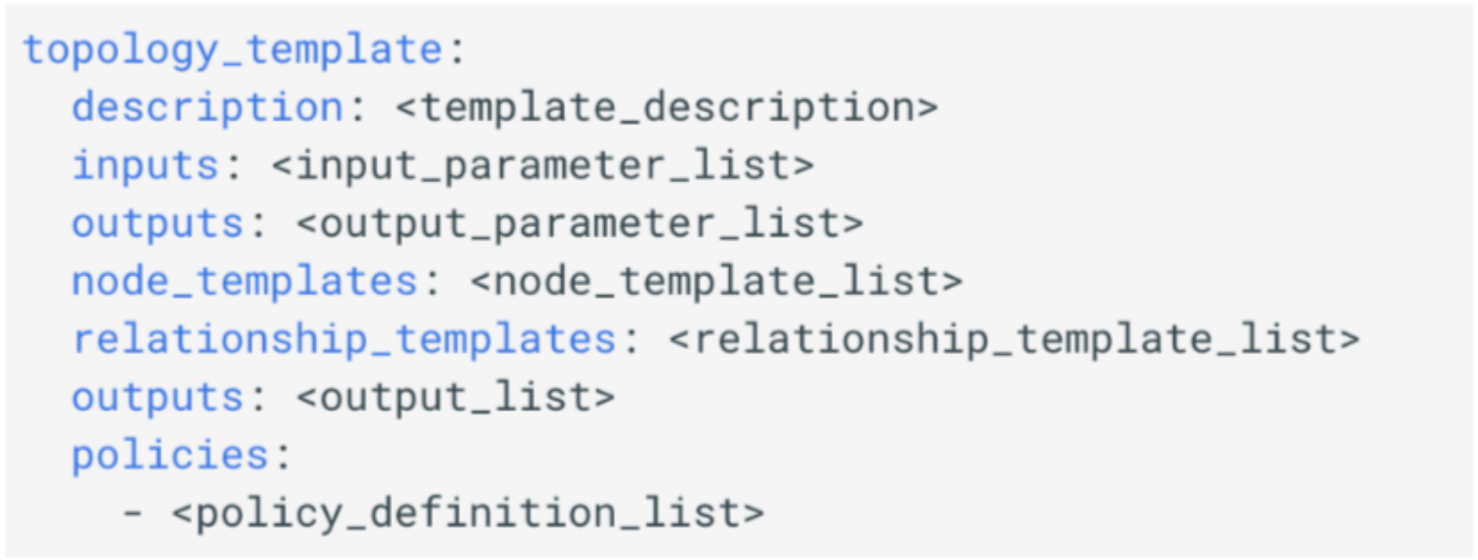
Topology Template Example 2



```
tosca_definitions_version: tosca_simple_yaml_1_0
description: Template for deploying a single server with MySQL software on top.

topology_template:
  inputs:
    # omitted here for brevity

  node_templates:
    mysql:
      type: tosca.nodes.DBMS.MySQL
      properties:
        root_password: { get_input: my_mysql_rootpw }
        port: { get_input: my_mysql_port }
      requirements:
        - host: db_server

    db_server:
      type: tosca.nodes.Compute
      capabilities:
        # omitted here for brevity
```



Simplified Topology Template Structure

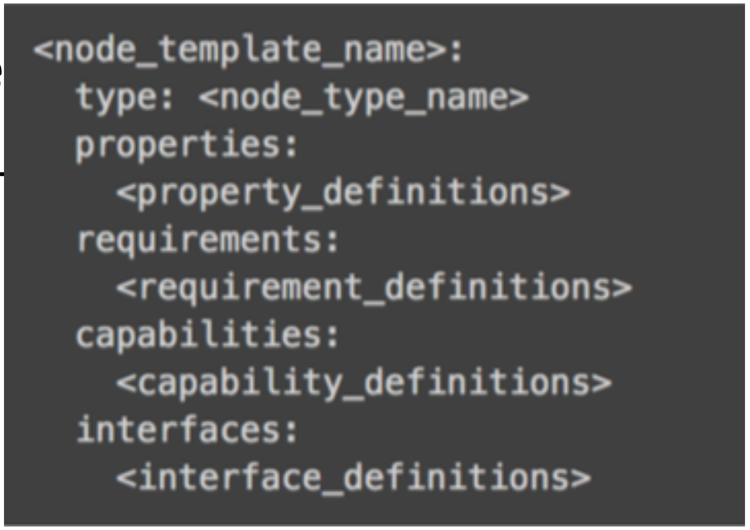


```
topology_template:
  description: <template_description>
  inputs: <input_parameter_list>
  outputs: <output_parameter_list>
  node_templates: <node_template_list>
  relationship_templates: <relationship_template_list>
  outputs: <output_list>
  policies:
    - <policy_definition_list>
```

Node template

- A **Node template** is an **instance** of a specified **Node Type** and can provide customized properties, constraints or operations which override the defaul provided by its Node Type and its implementations

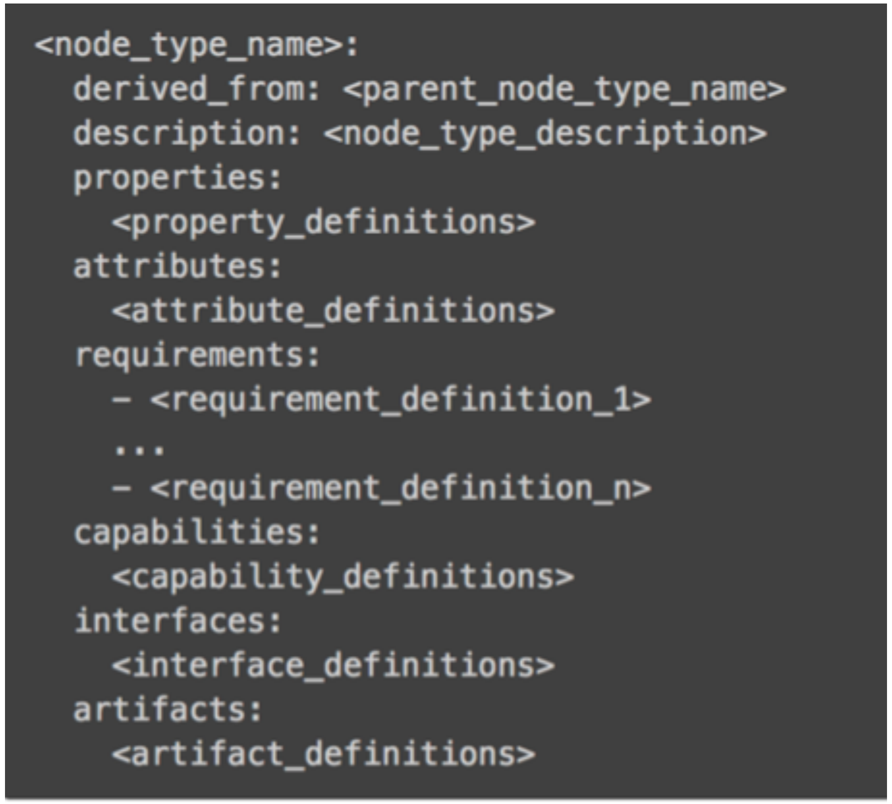
- An instance of a type (like Object to Class)
  - ➢Has specific properties
  - ➢Has artifacts:
    - What to install
    - How to install (mapped to interface hooks)
  - ➢Has requirements and capabilities (or relationships)



```
<node_template_name>:
  type: <node_type_name>
  properties:
    <property_definitions>
  requirements:
    <requirement_definitions>
  capabilities:
    <capability_definitions>
  interfaces:
    <interface_definitions>
```

Node Type

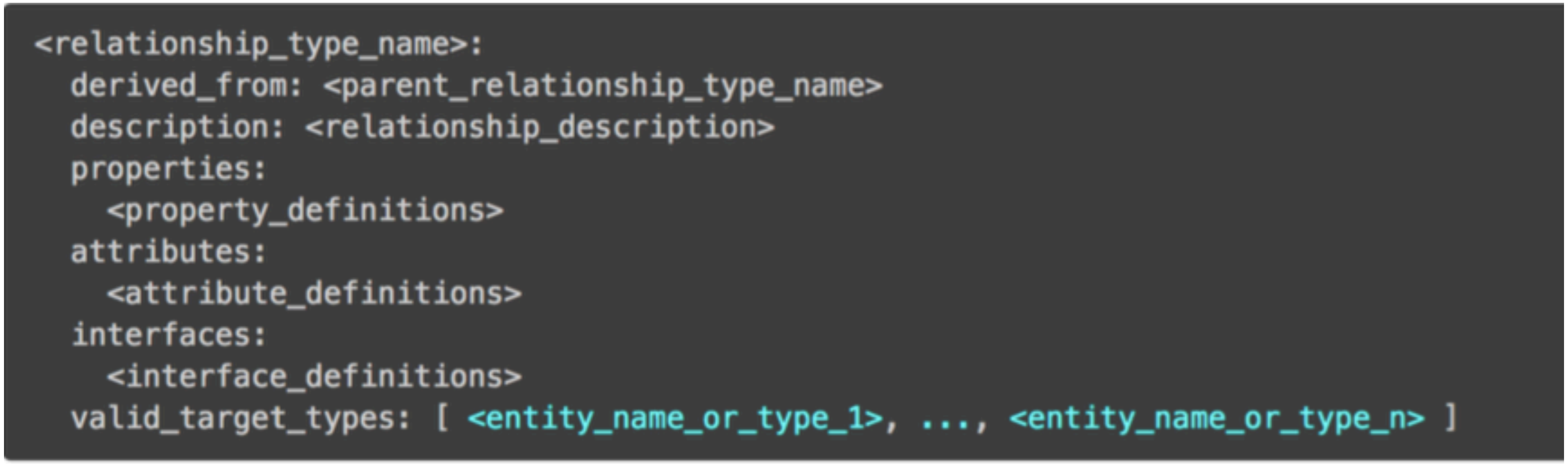
- Describes a Cloud or Software type (e.g. Server or Apache)



```
<node_type_name>:
  derived_from: <parent_node_type_name>
  description: <node_type_description>
  properties:
    <property_definitions>
  attributes:
    <attribute_definitions>
  requirements:
    - <requirement_definition_1>
    ...
    - <requirement_definition_n>
  capabilities:
    <capability_definitions>
  interfaces:
    <interface_definitions>
  artifacts:
    <artifact_definitions>
```

Relationship Type

- The basic relationship types are:
  - **dependsOn** – abstract type and its sub types:
  - **hostedOn** – a node is contained within another
  - **connectsTo** – a node has a connection configured to another

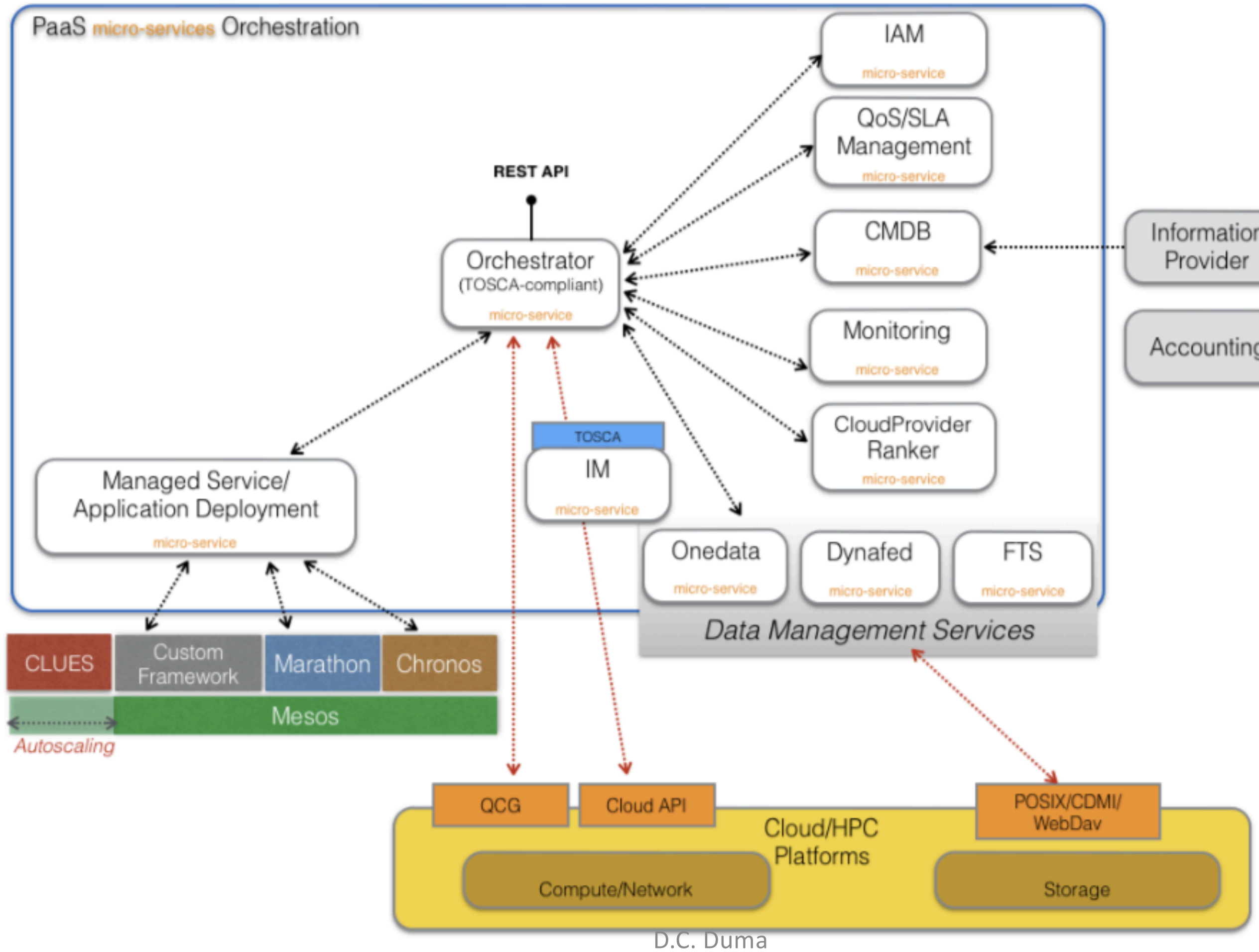


```
<relationship_type_name>:
  derived_from: <parent_relationship_type_name>
  description: <relationship_description>
  properties:
    <property_definitions>
  attributes:
    <attribute_definitions>
  interfaces:
    <interface_definitions>
  valid_target_types: [ <entity_name_or_type_1>, ..., <entity_name_or_type_n> ]
```

INDIGO PaaS Orchestration

- The **PaaS Orchestrator** is based on the developments carried out during the INDIGO-DataCloud project
  - advanced features and important enhancements are being implemented in the framework of three projects: **DEEP-Hybrid DataCloud**, **eXtreme-DataCloud** and **EOSC-Hub**
- It allows to coordinate the **provisioning** of *virtualized* compute and storage resources on different Cloud Management Frameworks (like OpenStack, OpenNebula, AWS, etc.) and the **deployment** of dockerized services and jobs on Mesos clusters.
- The PaaS orchestrator features advanced **federation** and **scheduling** capabilities ensuring the **transparent access** to heterogeneous cloud environments and the **selection of the best resource providers** based on criteria like user's SLAs, services availability and data location

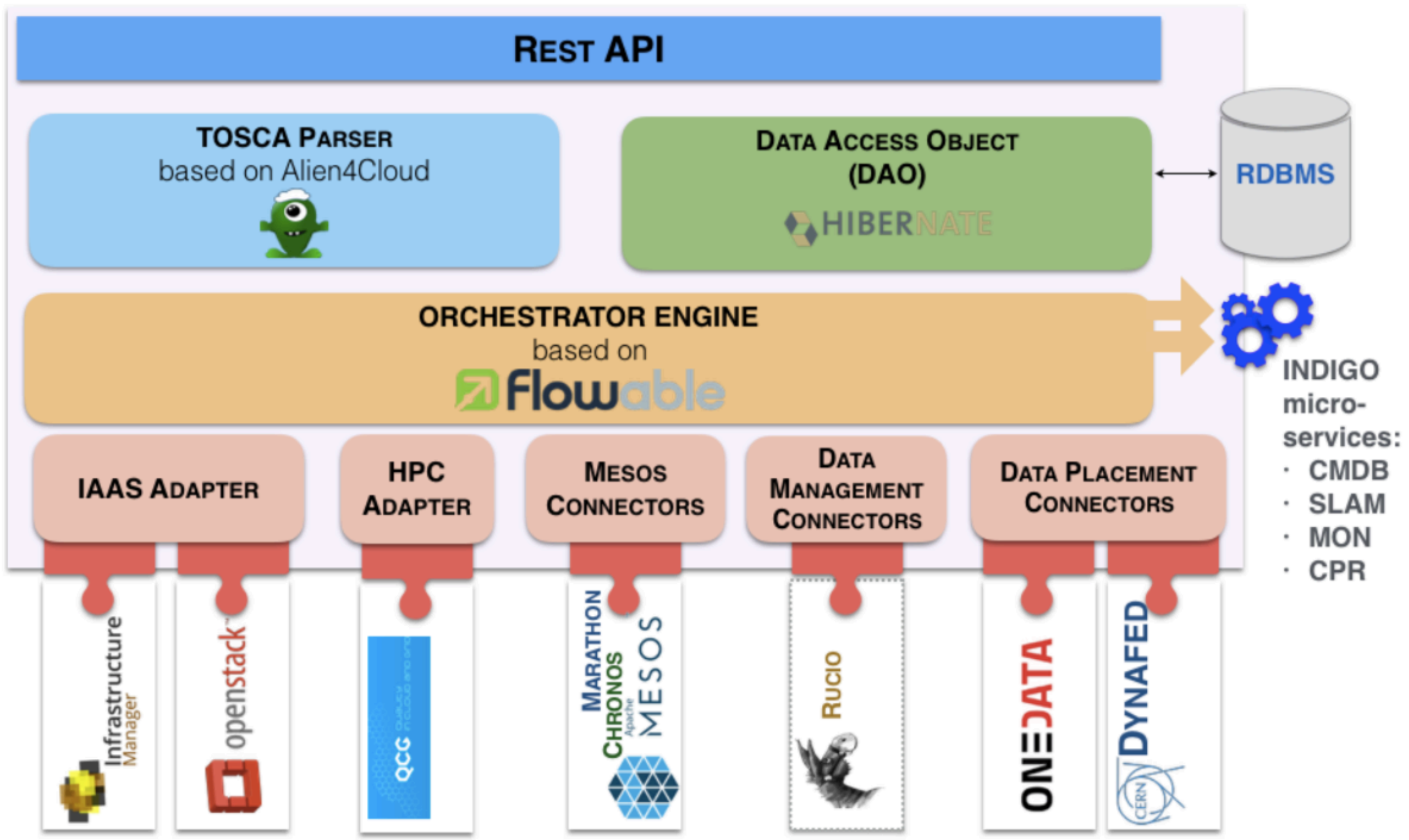
INDIGO Platform as a Service



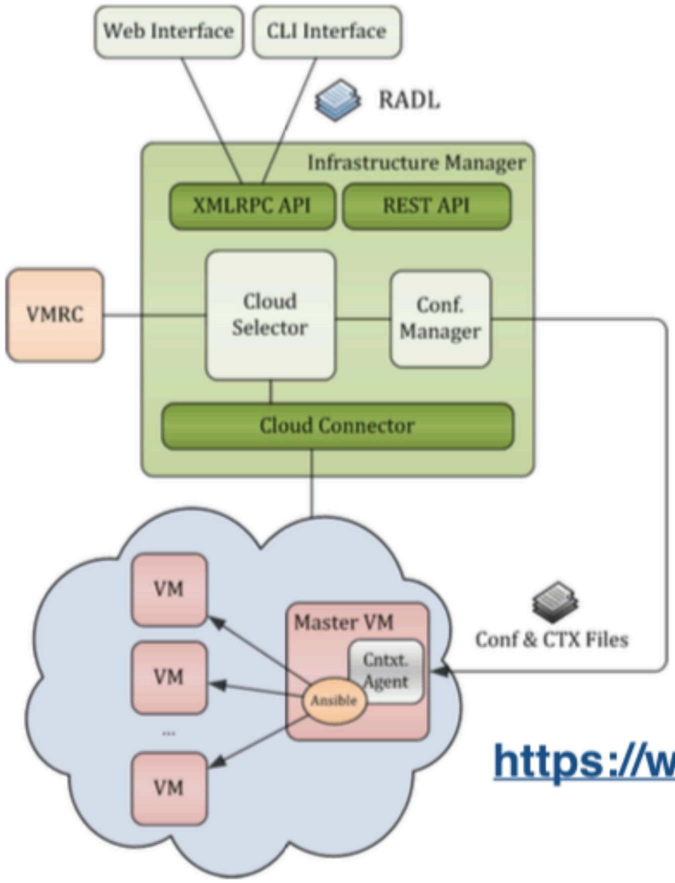
The deployment workflow

- The Orchestrator **receives** the deployment request (TOSCA template)
- The Orchestrator collects all the information needed to deploy the virtual infra/service/job consuming others PaaS µServices APIs:
  - **SLAM Service**: get the prioritized list of SLAs per user/group;
  - **Configuration Management DB**: get the the capabilities of the underlying IaaS platforms;
  - **Data Management Service**: get the status of the data files and storage resources needed by the service/application
  - **Monitoring Service**: get the IaaS services availability and their metrics;
  - **CloudProviderRanker Service** (Rule Engine): sort the list of sites on the basis of configurable rules
- The orchestrator delegates the deployment to IM, Mesos or QCG-Computing based on the TOSCA template and the list of sites.
- Cross-site deployments are also possible.

PaaS Orchestrator architecture



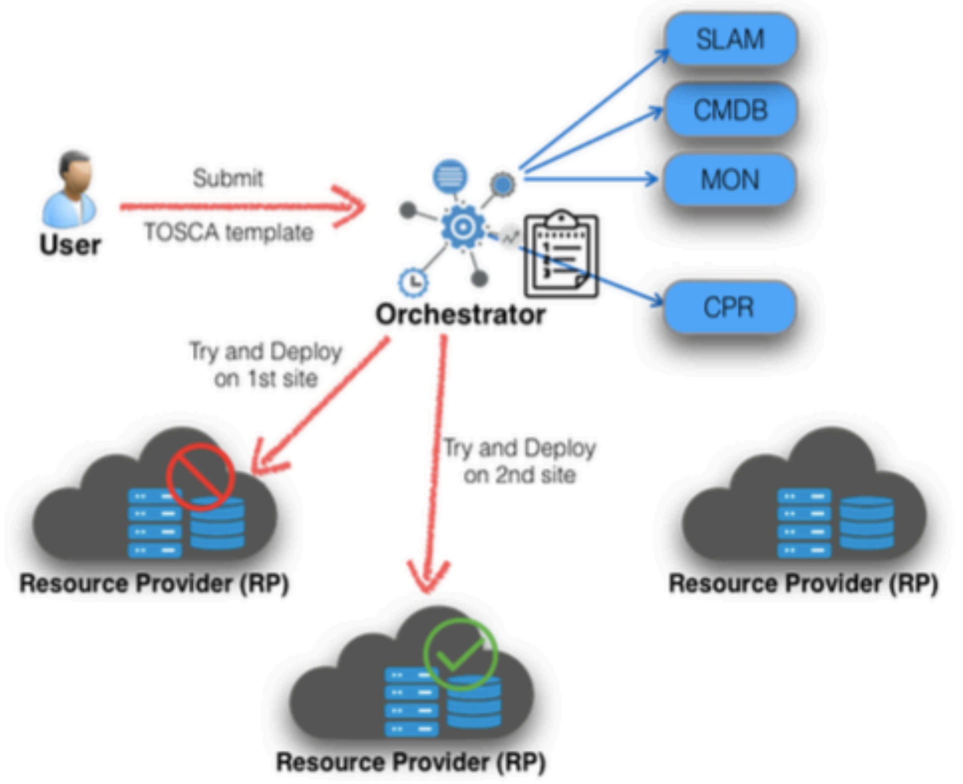
Infrastructure Manager Architecture



https://www.grycap.upv.es/im/index.php

Deployment retry strategy

- The Orchestrator implements a **trial-and-error** mechanism that allows to re-schedule the deployment on the next available cloud provider from the list of candidate sites.
- Example:
    - deployment fails because of exceeding the quota on the chosen site



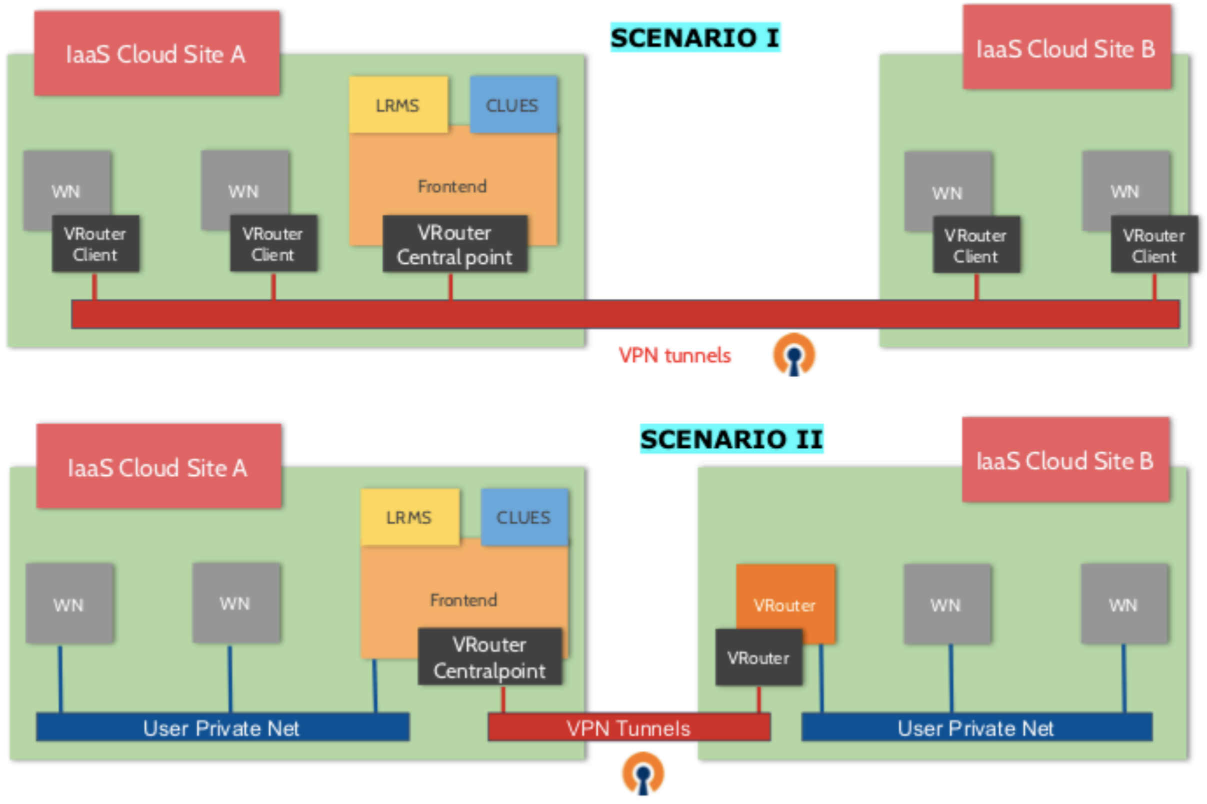
GPU scheduling and HPC integration

- The PaaS Orchestrator supports the deployment of virtual machines and containers that need to **access specialised hardware devices**, namely GPUs, to provide the processing power required by tasks like Machine Learning algorithms
  - the GPU requirements (num, vendor, model) can be specified in the TOSCA template
  - the Orchestrator automatically selects the sites/services that provide the needed capabilities (flavors, gpu support)
- The Orchestrator includes a plugin for **submitting jobs to HPC facilities**
  - exploits the **QCG-Computing** service (PSNC) that exposes REST APIs to submit jobs to the underlying batch systems

Support for hybrid deployments of elastic clusters

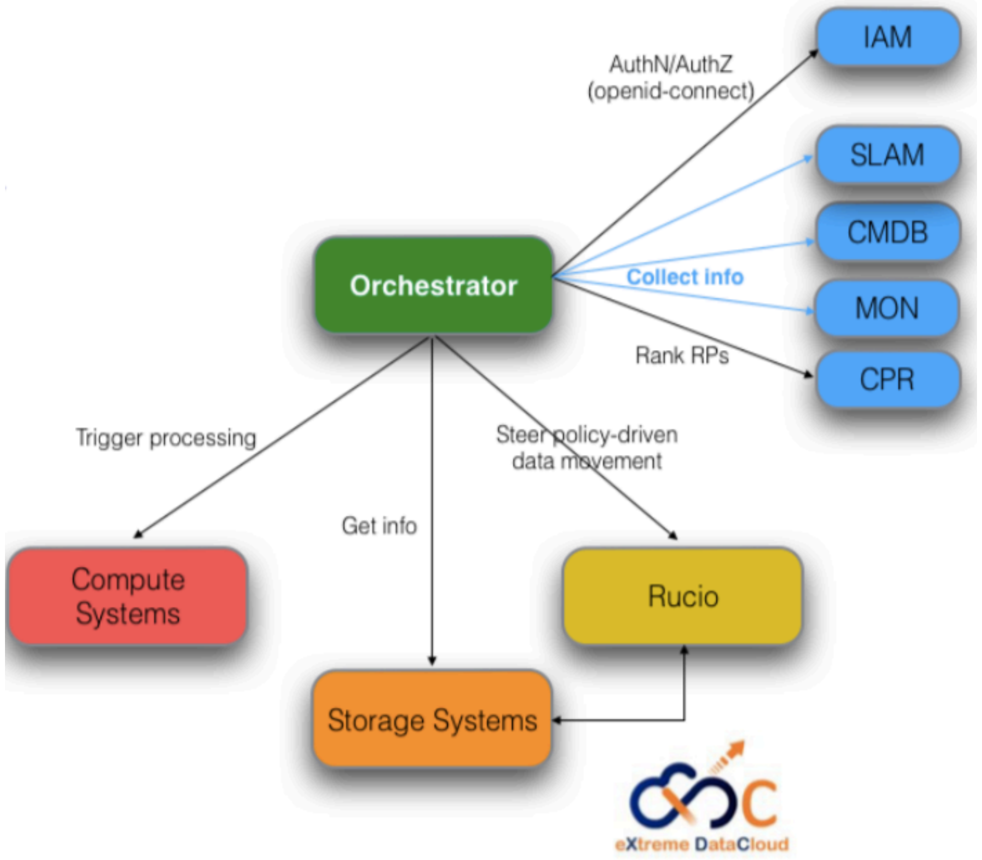
- Scenario I:
  - exploits L2 network providec by the site

- Scenario II:
  - dedicated private nets are automatically provisioned



Further features and enhancements

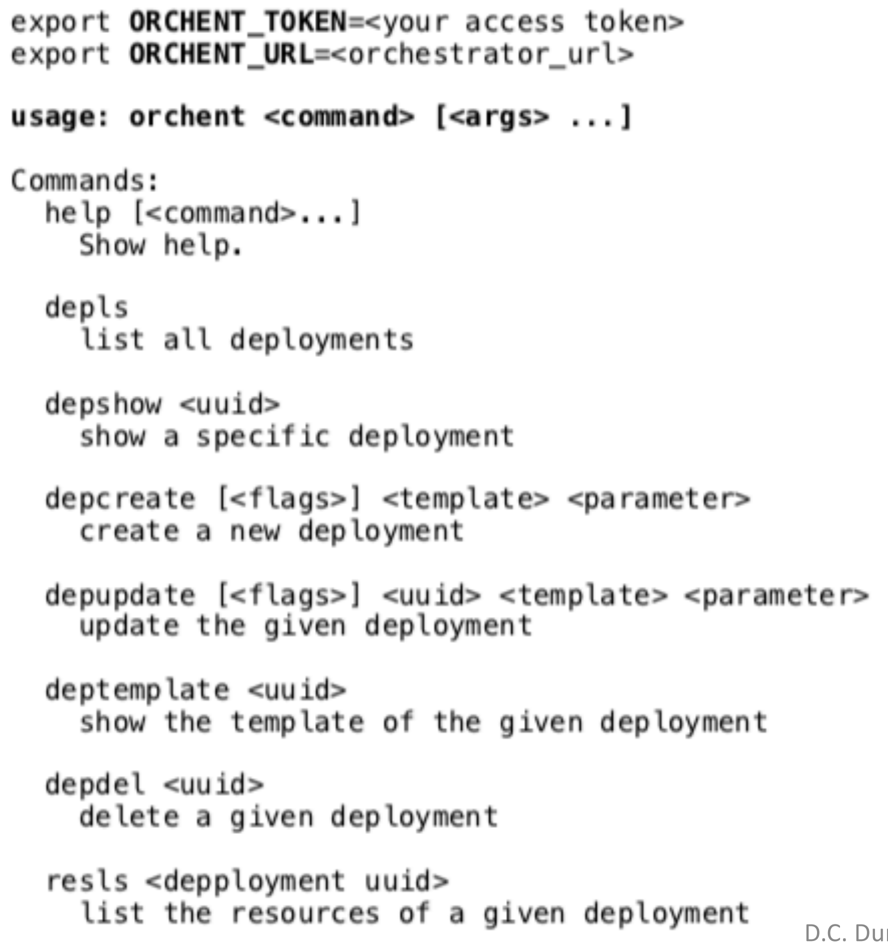
- The Paas Orchestrator has been enhanced to **schedule** the processing **jobs near** the **data**
- The PaaS orchestrator is been extended in order to:
  - Integrate a **data management policy engine** (QoS and Data Life Cycle)
    - Move data between distributed storages
    - Specify different QoSfor replicas
    - Support workflows for data pre-processing and ingestion



Orchestrator REST APIs

- **Create a deployment:**
  - POST request to /deployments - parameters:
    - template: string containing a TOSCA YAML-formatted template
    - parameters: the input parameters of the deployment (map of strings
- **Get deployment details**
  - GET request to /deployments:
    curl 'http://localhost:8080/deployments/<uuid>'
- **Delete deployment**
  - DELETE request
    curl ‘http://localhost:8080/deployments/<uuid>'
- Documentation:
  - http://indigo-dc.github.io/orchestrator/restdocs/#overview

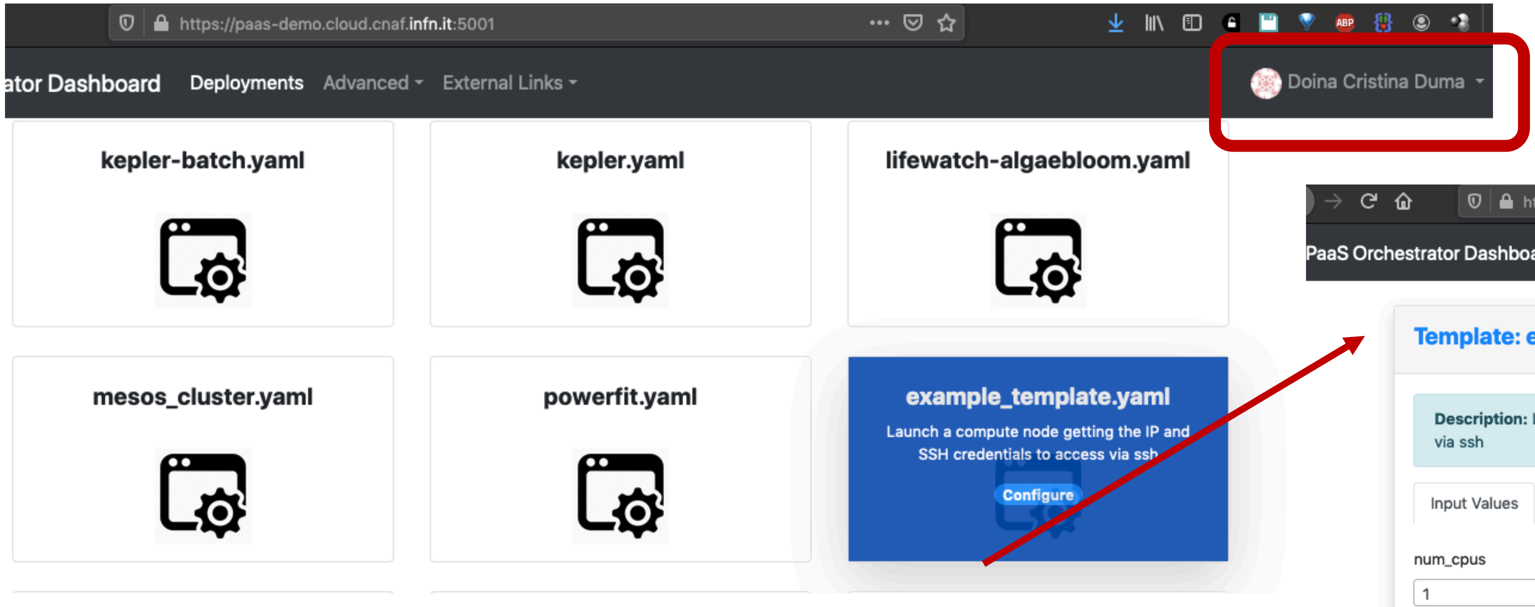
Orchent: the Orchestrator CLI

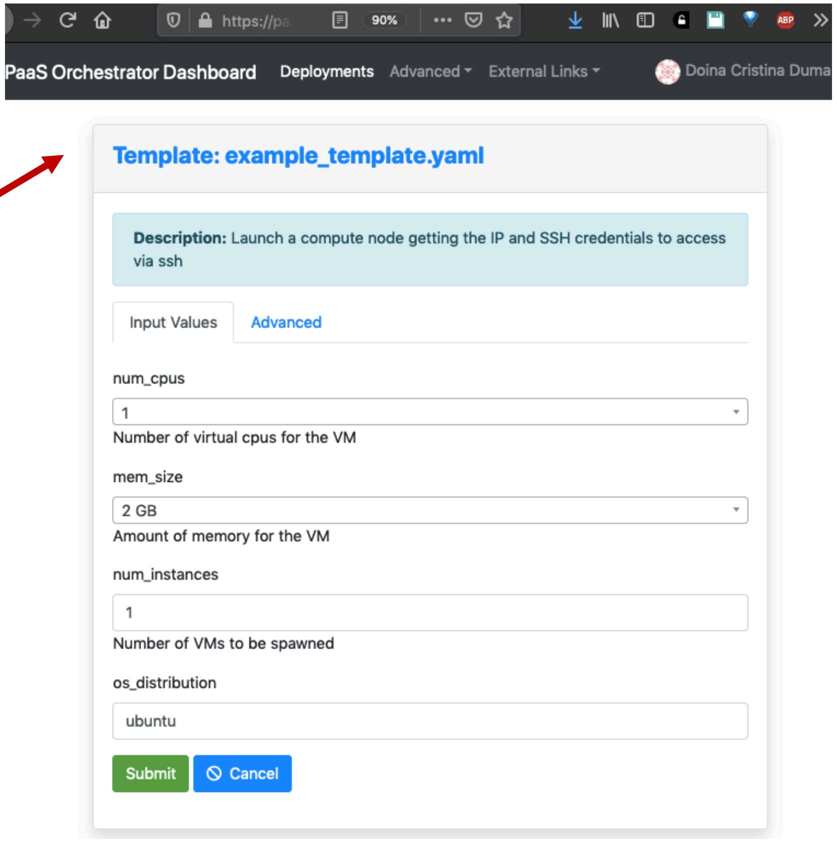


```
export ORCHENT_TOKEN=<your access token>
export ORCHENT_URL=<orchestrator_url>

usage: orchent <command> [<args> ...]

Commands:
  help [<command>...]
    Show help.

  depls
    list all deployments

  depshow <uuid>
    show a specific deployment

  depcreate [<flags>] <template> <parameter>
    create a new deployment

  depupdate [<flags>] <uuid> <template> <parameter>
    update the given deployment

  deptemplate <uuid>
    show the template of the given deployment

  depdel <uuid>
    delete a given deployment

  resls <depployment uuid>
    list the resources of a given deployment
```

Orchestrator dashboard

Authentication via INDIGO IAM





Template: example_template.yaml

Description: Launch a compute node getting the IP and SSH credentials to access via ssh

Input Values | Advanced

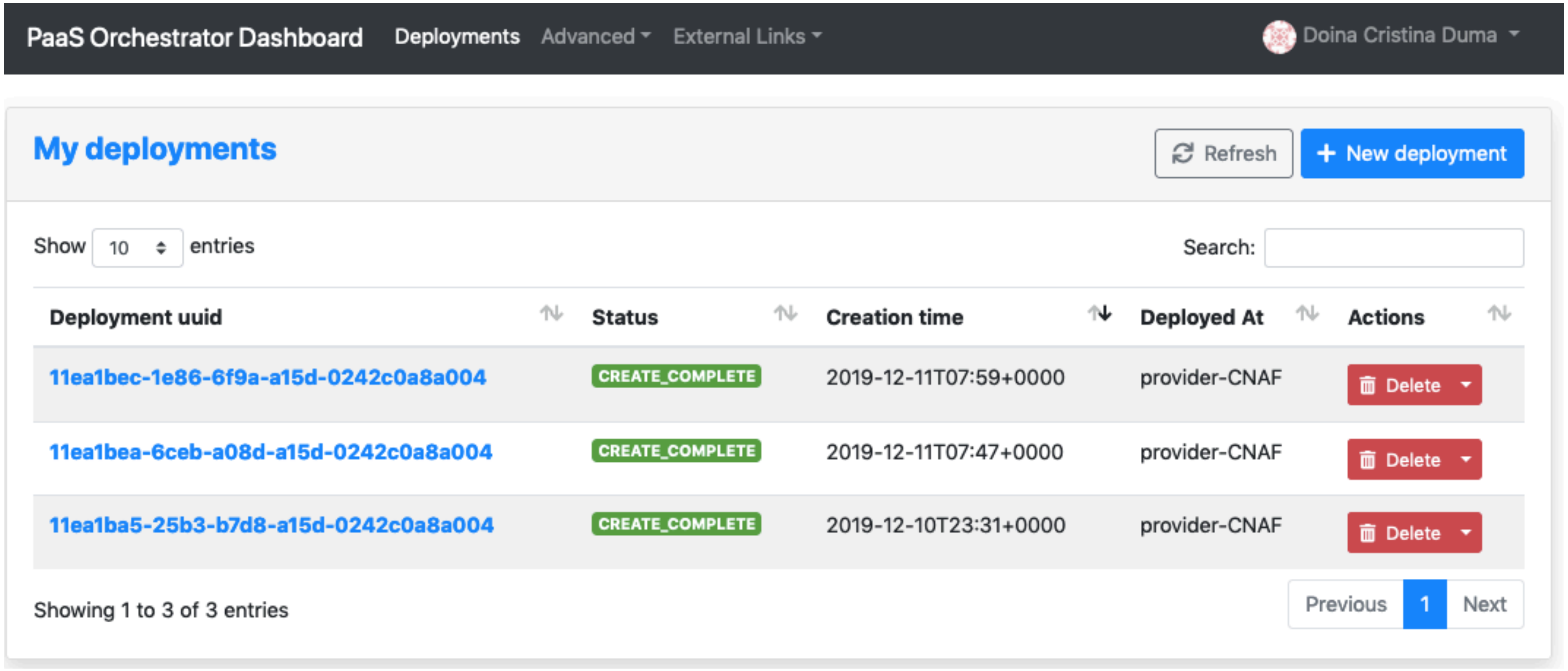
num_cpus: 1
Number of virtual cpus for the VM

mem_size: 2 GB
Amount of memory for the VM

num_instances: 1
Number of VMs to be spawned

os_distribution: ubuntu

List own deployments



PaaS Orchestrator Dashboard   Deployments   Advanced   External Links   Doina Cristina Duma

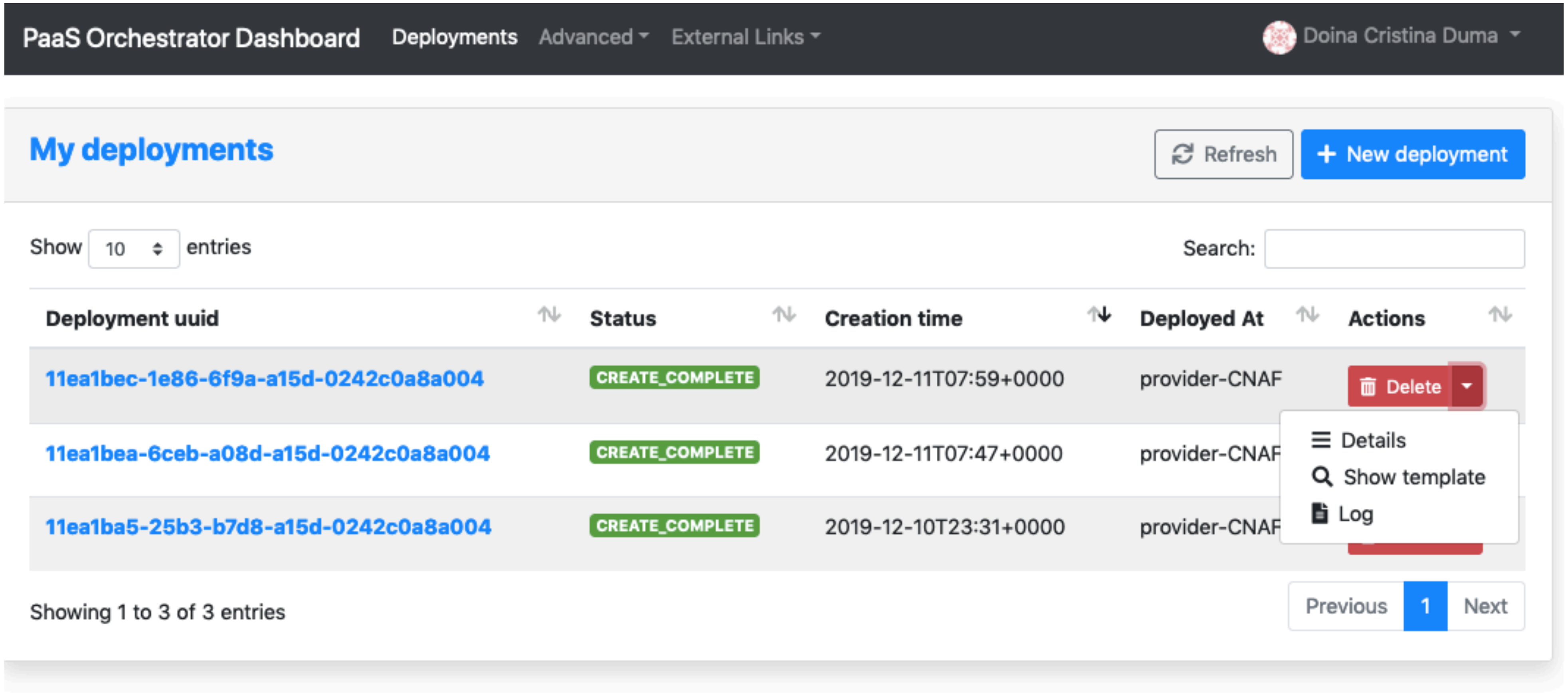
My deployments   Refresh   + New deployment

Show 10 entries   Search:

Deployment uuid	Status	Creation time	Deployed At	Actions
11ea1bec-1e86-6f9a-a15d-0242c0a8a004	CREATE_COMPLETE	2019-12-11T07:59+0000	provider-CNAF	Delete
11ea1bea-6ceb-a08d-a15d-0242c0a8a004	CREATE_COMPLETE	2019-12-11T07:47+0000	provider-CNAF	Delete
11ea1ba5-25b3-b7d8-a15d-0242c0a8a004	CREATE_COMPLETE	2019-12-10T23:31+0000	provider-CNAF	Delete

Showing 1 to 3 of 3 entries   Previous 1 Next

Get deployments details and outputs



PaaS Orchestrator Dashboard Deployments Advanced External Links Doina Cristina Duma

My deployments

Refresh | New deployment

Show 10 entries Search:

Deployment uuid	Status	Creation time	Deployed At	Actions
11ea1bec-1e86-6f9a-a15d-0242c0a8a004	CREATE_COMPLETE	2019-12-11T07:59+0000	provider-CNAF	Delete
11ea1bea-6ceb-a08d-a15d-0242c0a8a004	CREATE_COMPLETE	2019-12-11T07:47+0000	provider-CNAF	
11ea1ba5-25b3-b7d8-a15d-0242c0a8a004	CREATE_COMPLETE	2019-12-10T23:31+0000	provider-CNAF	

Showing 1 to 3 of 3 entries

INDIGO Resources

- https://github.com/indigo-dc/tosca-templates
- https://github.com/indigo-dc/tosca-types

- https://galaxy.ansible.com/indigo-dc/

- https://hub.docker.com/u/indigodatacloud/dashboard/

References

- **TOSCA Simple Profile in YAML Version 1**
  - http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csprd02/TOSCA-Simple-Profile-YAML-v1.0-csprd02.html
- **Cloud Portability, Lifecycle Management and more** https://www.slideshare.net/CloudStandardsCustomerCouncil/oasis-tosca-cloud-portability-and-lifecycle-management
- TOSCA presentation
  - https://www.slideshare.net/CloudStandardsCustomerCouncil/oasis-tosca-cloud-portability-and-lifecycle-management

Hands-on Outline

- Goal: **submit some simple TOSCA template through the PaaS Orchestrator**
  1. Deploy a VM
  2. Deploy a JupytherHub on top of a Kubernetes cluster

https://baltig.infn.it/corsi_formazione_ccr/corso_bd_2019/tree/master/ansible_tosca/tosca