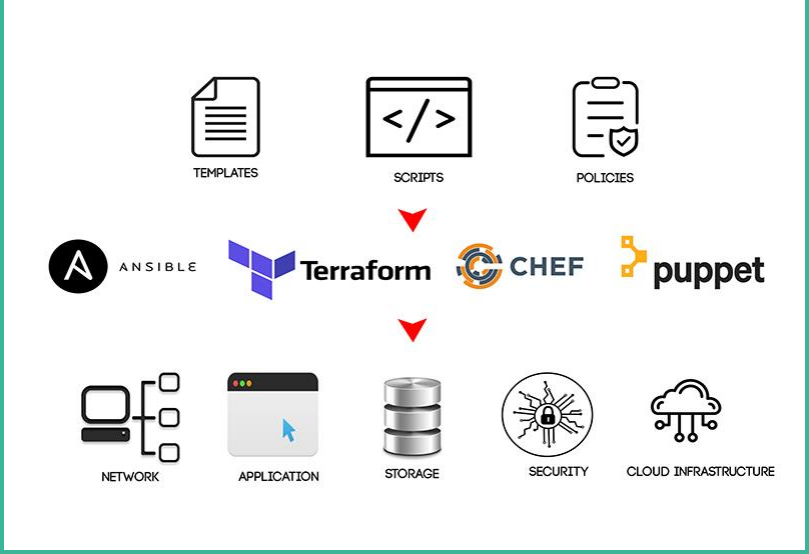
Infrastructure Automation

Infrastructure as Code (IaC) – Ansible

Sven Knockaert



Odisee
DE CO-HOGESCHOOL

Infrastructure Automation

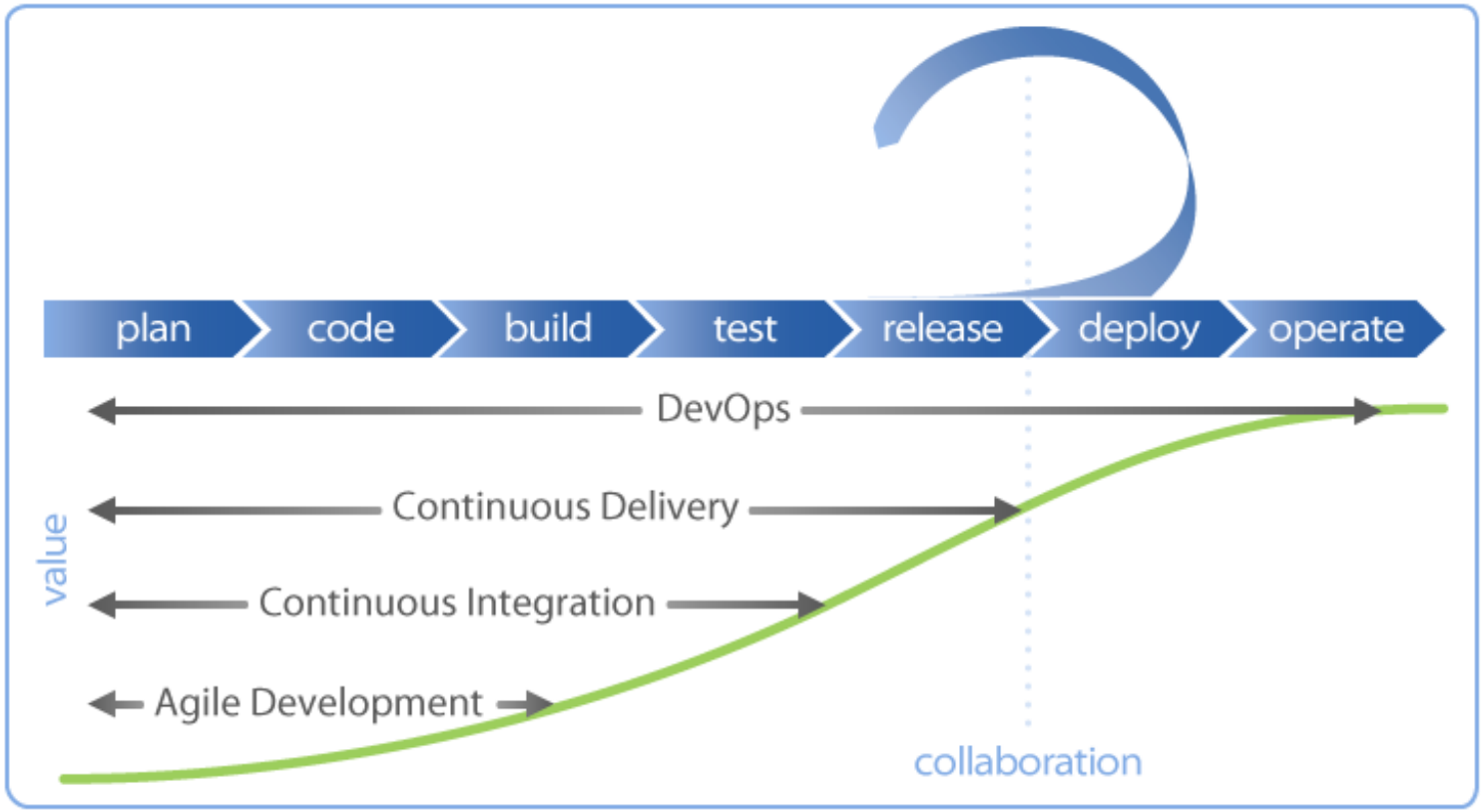
Infrastructure Automation is het proces waarbij we onze IT-infrastructuur **via scripts** gaan automatiseren met als doel oa. **snel** en **herhaaldelijk** items configureren. *(typisch ingezet bij DevOps)*

Vb:

- OS installeren
- Software deployen, configureren
- User provisionen
- Netwerk aanpassen
- Services opzetten (vb. web- en db-servers)
- …

Infrastructure Automation

- Automation / Infrastructure as Code ontontbeerlijk in DevOps.



Odisee
DE CO-HOGESCHOOL

Infrastructure Automation

Automation via:

Klassieke scripting

- Bash
- Powershell
- Python
- Ruby
- …

Automation Tools

puppet

CHEF

Terraform

CFEngine

ANSIBLE

juju

PowerShell
Desired State Configuration

RUDDER

SALTSTACK

CloudFormation

HashiCorp
Packer

Odisee
DE CO-HOGESCHOOL

Configuration management – Infrastructure as Code (IaC)

Met configuration management beschrijf je hoe de "server" moet geconfigureerd zijn. Je bouwt met Automation Tools een laag van abstractie boven de klassieke scripts en provisioning-tools.

De onderliggende technologie (met of zonder agent)

- Zorgt ervoor dat de 'desired state' actief is op de servers
- Monitort wijzigingen in configuratie of status (gehele lifecycle)
- Inventariseert & rapporteert

Configuration management – Infrastructure as Code (IaC)

- Infrastructure as Code
  - Zorgt voor “Single Source of Truth”
  - Configuraties bij te houden onder versiecontrole, VCS (Github, Gitlab, BitBicket,…)
    - Rollback mogelijk!
  - Zorgt voor “herhaalbaarheid”

Enkele concepten

- **Idempotency**: Automation tool produceert telkens de gewenste status, te bereiken door “aanpassingen” te doen => “convergentie”
  - **Enkel** als een requirement niet voldoet wordt aanpassing doorgevoerd
  - Geen (minder) gevaar om bestaande items “stuk” te maken
  - Vb: Ansible, Chef

- **Immutable**: Automation tool zal een verandering doorvoeren door een “vernietiging” en volledige “herconfiguratie” van gewenste item.
  - Grotere impact op de bestaande configuratie
  - Vb: Terraform, Cloudformation

Enkele concepten

- Proceduraal: Automation tool volgt imperatieve sequentie van commando's. (weinig intelligent, doelsysteem status moet gekend zijn.)

- Declaratief: Automation tool beschrijft een "Desired State", houdt rekening met huidige status

Enkele concepten

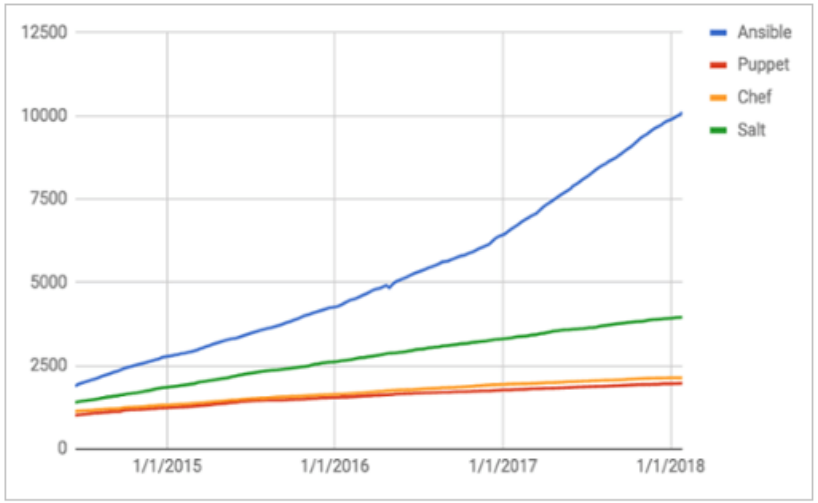
- **Stateless:** Automation werkt best als toepassingen stateless zijn. Dit heeft als gevolg dat redeployen geen verlies van info of data betekent.

- **Not stateless** (statefull): Info en data staat lokaal op het systeem of in een DB die onderwerp uitmaakt van de automation.

Full-stack Automation (infrastructure én applicaties)

Vereist stateless aanpak. => nadenken hoe “persistent” data dan wel kan bewaard worden? (aparte DB?, Cloudstorage?,….)

Marktspelers



puppet

CHEF

ANSIBLE

Terraform

	Source	Cloud	Type	Infrastructure	Language	Agent	Master	Community	Maturity
Chef	Open	All	Config Mgmt	Mutable	Procedural	Yes	Yes	Large	High
Puppet	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Large	High
Ansible	Open	All	Config Mgmt	Mutable	Procedural	No	No	Huge	Medium
SaltStack	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Large	Medium
CloudFormation	Closed	AWS	Provisioning	Immutable	Declarative	No	No	Small	Medium
Heat	Open	All	Provisioning	Immutable	Declarative	No	No	Small	Low
Terraform	Open	All	Provisioning	Immutable	Declarative	No	No	Huge	Low

Odisee
DE CO-HOGESCHOOL

Marktspelers

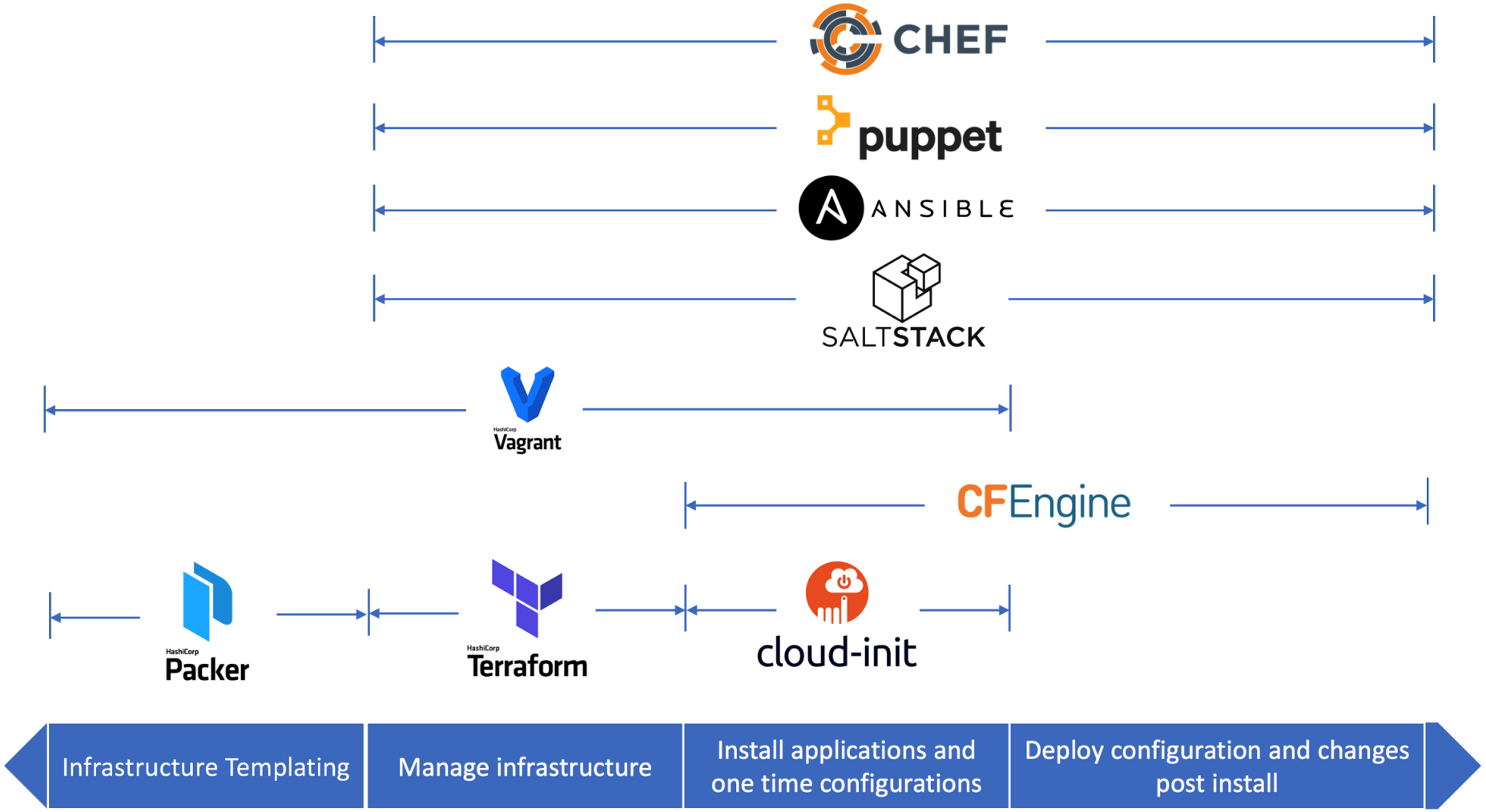
Open Source Configuration Management

	Puppet	Chef	Salt	Ansible
Commercial Support	Puppet Labs	Opscode	SaltStack	AnsibleWorks
Core Technology	Ruby	Ruby; Erlang	Python	Python
Communication	SSL	SSL	0mq	SSH; 0mq optional
Control Interface	Manifest: proprietary language	Recipe: Ruby	States: YAML and other standard template tools	Playbooks: JSON, YAML, INI text files
Dependency Awareness	Yes	No	Yes	No
Community Repository	Puppet Forge	Cookbooks	SaltStarters	ansible-examples on GitHub
List Price (annual/node)	Std: $88 / Prem: $152	Std: $72 / Prem: $?	*"contact sales"*	Std: $100 / Prem: $250
Date established	Founded 2005; February 2011 first commercial project	January 2009	March 2011	February 2012; AnsibleWorks March 2013
Ref customers	eBay, Google, Disney, many more	Facebook, Ancestry.com	LinkedIn, HP Cloud	Evernote, Rackspace
Strengths	Most mature: users, mindshare, integrations	No proprietary language; execution order instead of dependency	Execution speed	Few dependencies – easy to get started; agentless, leaves no trace on machines; more readable syntax
Headquarters	Portland	Seattle	Salt Lake City	Santa Barbara

vmware

Odisee
DE CO-HOGESCHOOL

Toepassingsgebied verschillend

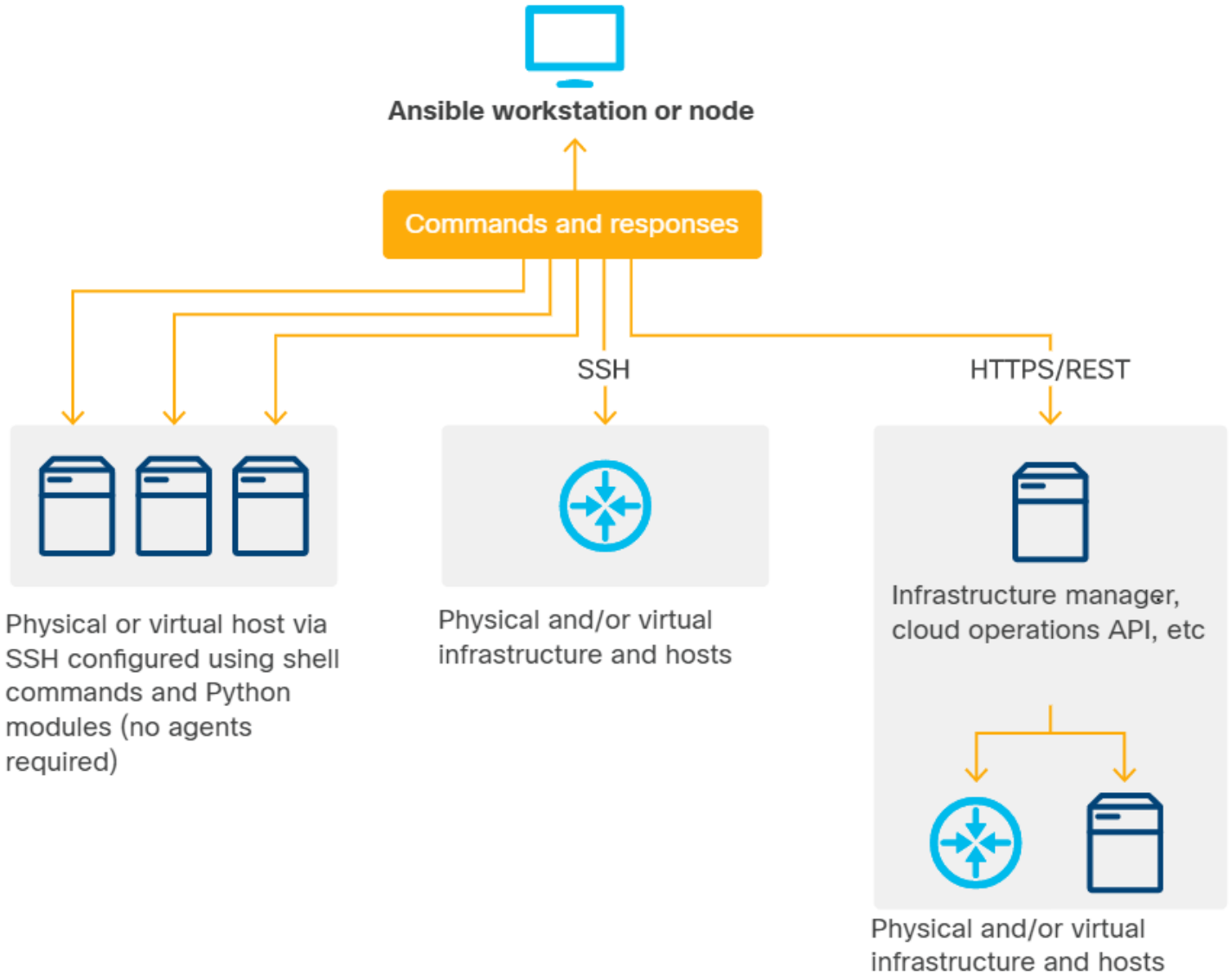


ANSIBLE

- Basis architectuur is vrij eenvoudig:
  - Control node werkt via SSH
    - Voert shell commands rechtstreeks uit of via REST interface
    - Injecteert Python scripts (uitvoeren en weer verwijderen)
  - Laat toe om taken op verschillende targets gelijktijdig uit te voeren (vb. op 100 servers, routers,…)
  - Plugins zorgen voor oa. “gathering facts” per specifiek platform.

    Maakt het mogelijk om configuratie uit te voeren op systemen die geen Python draaien (vb. via REST)
  - Ansible code structuur => oa. Tasks, Handlers beschreven in YAML syntax (.yml)

ANSIBLE



Odisee
DE CO-HOGESCHOOL

Typische toepassingen Ansible

- Provisioning Environments
  - "Infrastructuur" opbouwen, vb een omgeving opzetten in AWS (netwerken, Security policies,…)
- Configuring Operating Systems
  - Besturingssysteem aanpassen, vb Linux of Windows software installeren, OS patches, services starten/stoppen… => Desired State
- Deploying Applications
  - Stappen uitvoeren om een applicatie (vb eigen webcode) te installeren met alle afhankelijkheden.
- Performing Compliance Checks
  - Taken uitvoeren om een desired state te checken en te bereiken, vb: Firewall regels aanpassen, huidige status melden,…

Typische toepassingen Ansible

- Running Tasks with Ansible => 2 mogelijke manieren

  - Ad Hoc Commands => enkelvoudige opdracht

```
$ ansible hostsgent -m ping
$ ansible hostsgent -m command "/sbin/shutdown"
$ ansible hostsgent -m service -a "name=apache2 state=restarted"
```

*Naam van de groep met servers => **Inventory***

module

*Parameters bij de module **key/value** formaat*

  - Playbooks
    - Maakt het mogelijk om meerdere opdrachten uit te voeren
    - Opgemaakt in YAML met 1 of meerdere "plays"

ANSIBLE

- Inventory
  - Ansible gebruikt geen agents => specifiëren welke servers/devices je als target wenst te gebruiken
  - Deze info plaatsen we in een "Inventory File" (`/etc/ansible/hosts`) <= default locatie, kan elders

```
mail.example.com
```

Host name

```
[webservers]
foo.example.com
bar.example.com
```

Web servers group

```
[dbservers]
one.example.com
two.example.com
three.example.com
```

Database servers group

```
[atlanta]
host1
host2
```

Atlanta Group

```
[raleigh]
host2
Host3
```

Raleigh Group

```
[southeast:children]
atlanta
raleigh
```

South East Nested Group

```
[southeast:vars]
some_server=foo.southeast.example.com
halon_system_timeout=30
self_destruct_countdown=60
escape_pods=2
```

Variables for South East Group

```
[usa:children]
southeast
northeast
southwest
northwest
```

USA Nested Group

Variabelen voor host kunnen beter in aparte files: /etc/ansible/host_vars of group_vars

ANSIBLE

- Inventory
  - Standaard maakt Ansible gebruik van
    - SSH voor Linux
    - WinRM voor windows

  - Connection settings per host of per groep mogelijk.
  - Connection settings kunnen in de hosts file bepaald worden of in ansible.cfg (indien voor alle hosts identiek)

=> Bij Linux voorkeur ssh met Public/Private keypair!!

ANSIBLE

- Inventory
  - Opbouw “INI” of “YAML”
  - Naast default inventory (/etc/ansible/hosts.yaml) eigen inventories mogelijk
    - `ansible -i <path naar eigen inventory> ...`

```
mail.example.com
```

Host name

```
[webservers]
foo.example.com
bar.example.com
```

Web servers group

```
[dbservers]
one.example.com
two.example.com
three.example.com
```

Database servers group

INI ⟷ YAML

```yaml
all: # keys must be unique, i.e. only one 'hosts' per group
    hosts:
        test1:
        test2:
            host_var: value
    vars:
        group_all_var: value
    children:   # key order does not matter, indentation does
        other_group:
            children:
                group_x:
                    hosts:
                        test5   # Note that one machine will work without a colon
                #group_x:
                #    hosts:
                #        test5  # But this won't
                #        test7  #
                group_y:
                    hosts:
                        test6:  # So always use a colon
            vars:
                g2_var2: value3
            hosts:
                test4:
                    ansible_host: 127.0.0.1
        last_group:
            hosts:
                test1 # same host as above, additional group membership
            vars:
                group_last_var: value
```

ANSIBLE

- Inventory
  - Inventory testen =>

```
$ ansible-inventory -vvv -i hosts.yaml –graph
```

```
ansible-inventory 2.9.14
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/student/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible-inventory
  python version = 3.6.8 (default, Apr 16 2020, 01:36:27) [GCC 8.3.1 20191121 (Red Hat 8.3.1-5)]
Using /etc/ansible/ansible.cfg as config file
host_list declined parsing /home/student/ansible-test/hosts.yaml as it did not pass its verify_file() method
script declined parsing /home/student/ansible-test/hosts.yaml as it did not pass its verify_file() method
Parsed /home/student/ansible-test/hosts.yaml inventory source with yaml plugin
@all:
  |--@production:
  |  |--10.129.28.130
  |  |--10.129.38.174
  |--@testservers:
  |  |--10.129.38.225
  |  |--10.129.38.227
  |--@ungrouped:
```

ANSIBLE

- Playbooks
  - YAML file die “Plays” bevat
  - “Hart” van Ansible waar alles samen komt.
    (Tasks, Modules, Handlers,…)
  - Play is een serie van oa. Tasks

First Play

Playbook

Second Play

```yaml
---
- hosts: lamp
  remote_user: ubuntu
  become: true

  roles:
    - lamp

  tasks:
    - name: Download app
      git:
        repo: "{{app_repo}}"
        dest: "{{app_download_dest}}"

- hosts: database
  remote_user: ubuntu
  become: true

  roles:
    - mysql

  tasks:
    - name:
      mysql_user:
        name: appuser
        password: 94nfsUl7
        priv: "*.*:ALL"
        state: present

    - name:
      mysql_db:
        name: appdata
        state: present
```

ANSIBLE

- Tasks
  - Bestaat uit een **module** en bijhorende metadata
    - Name
    - Handler notifications
    - Ignore Errors
    - Conditionals
    - Loops
    - …

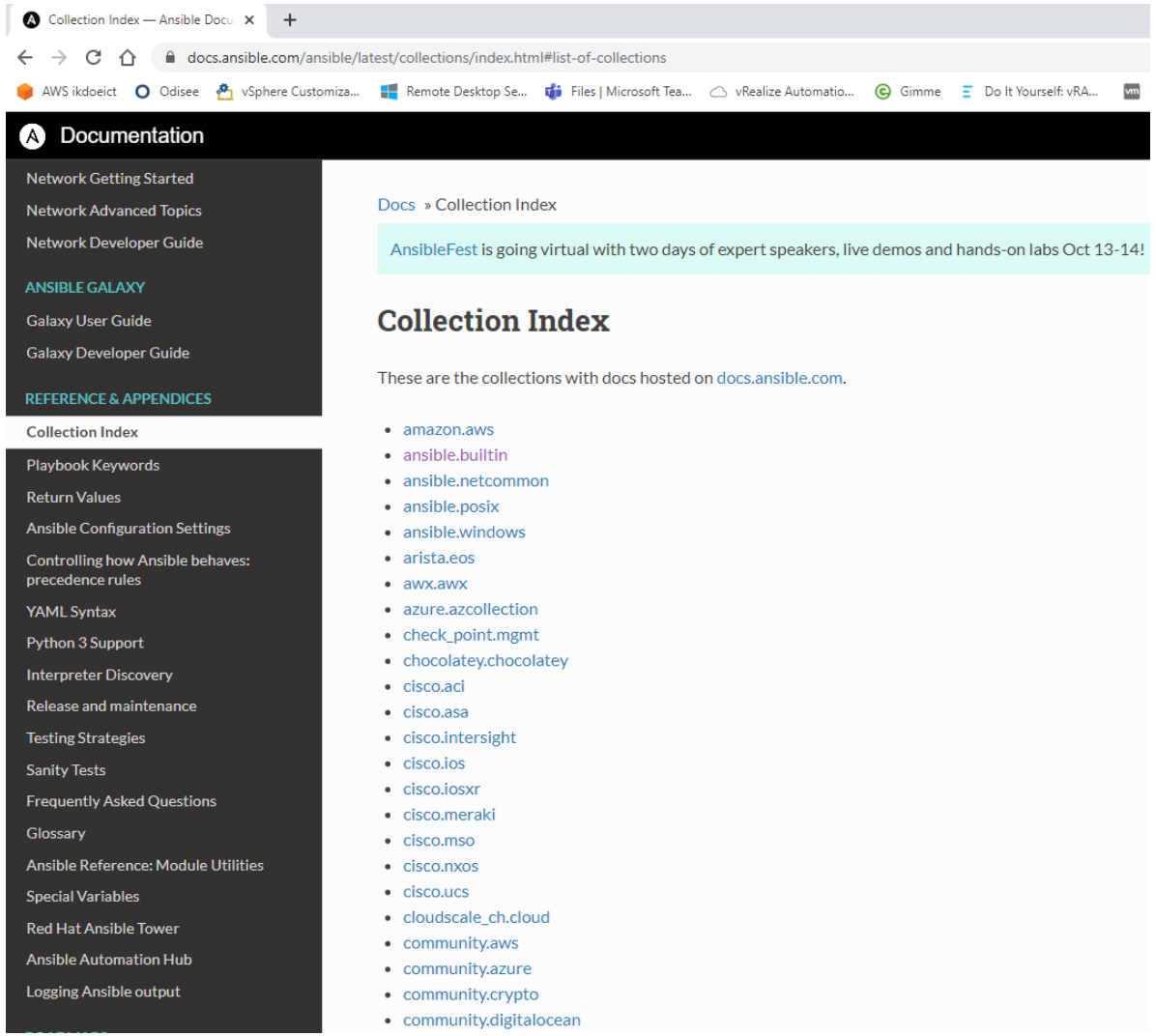
First Task

Second Task

```yaml
---
- hosts: localhost
  strategy: debug
  remote_user: ubuntu
  become: true
  connection: local

  tasks:

    - name: Install nmap - A network mapping tool
      apt:
        name: "nmap"
        state: present
      register: nmap

    - name: Stat testerror file
      stat:
        path: /bin/testerror
      register: stat
```

ANSIBLE

- Modules
  - Bevatten de uitvoering van taken
    - Files aanmaken
    - Services starten
    - Firewall regels aanpassen
    - …
  - 1000+ via https://docs.ansible.com
  - Eventueel zelf te schrijven



https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections

Odisee
DE CO-HOGESCHOOL

ANSIBLE

- Modules

Copy Module

```
- copy: src: /srv/myfiles/foo.conf
  dest: /etc/foo.conf
  owner: foo
  group: foo
  mode: 0644
```

Git Module

```
- git:
  repo: git://url.org/repo.git
  dest: /srv/checkout
  version: release-0.22
```

2 verschillende mogelijkheden van noteren

```yaml
---
- hosts: lamp
  remote_user: root

  tasks:
    - name: install apache
        apt:
            name: apache2
            state: present

    - name: write the apache config file
        template: src=apache.j2 dest=/etc/apache/sites-enabled/000-default.conf

    - name: ensure apache is running (and enable it at boot)
        service: name=httpd state=started enabled=yes


# Try installing the latest version of Apache
- hosts: test
  remote_user: root

  tasks:
    - name: install apache
        apt: name=apache2 state=latest
    - name: ensure apache is running (and enable it at boot)
        service: name=httpd state=started enabled=yes
```

Odisee
DE CO-HOGESCHOOL

ANSIBLE

Handlers, Facts, Variables, Templates

- Handlers => repetitieve tasks vereenvoudigen (cfr functions)

```
  - name: Copy the apache configuration file
    copy:
      src: "apache.conf"
      dest: /etc/apache2/sites-available/000-default.conf
    notify: restart apache


handlers:
  - name: restart apache
    service:
      name: apache2
      state: restarted
```

Vb: telkens apache herstarten enkel als 000-default.conf werd veranderd (notify change)

ANSIBLE

Handlers, Facts, Variables, Templates

- Variables =>
  - Te definiëren in oa. playbook, inventory,…

```yaml
---
- hosts: localhost
  strategy: debug
  remote_user: ubuntu
  become: true
  connection: local

  vars:
    username: "ben lambert"

  tasks:
    - name: Test variable override
      debug:
        msg: "{{username}}"
```

from **least** to greatest (⚠️ command line option have the least importance !!

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

Odisee
DE CO-HOGESCHOOL

ANSIBLE

Handlers, Facts, Variables, Templates

- Facts =>
  - Informatie, eigenschappen van een doel-systeem, te gebruiken als variabele

    vb: `$ ansible -i hosts.yaml testservers -m setup`


```
},
"ansible_fibre_channel_wwn": [],
"ansible_fips": false,
"ansible_form_factor": "Other",
"ansible_fqdn": "EntInfra-TEST1-sven-knockaert-1870.vsphere.local",
"ansible_hostname": "EntInfra-TEST1-sven-knockaert-1870",
"ansible_hostnqn": "",
"ansible_interfaces": [
    "ens192",
    "lo"
```

ANSIBLE

Handlers, Facts, Variables, Templates

- Templates =>
  - Zorgt voor “pre-processing” van files
  - Gebruikt **Jinja2** engine
  - Typisch om real-time variabelen in config-files in te vullen.

https://jinja.palletsprojects.com/en/2.11.x/templates/#synopsis

```
- hosts: all
  vars:
    variable_to_be_replaced: 'Hello world'
    inline variable: 'hello again'
  tasks:
    - name: Ansible Template Example
      template:
        src: hello_world.j2
        dest: /Users/mdtutorials2/Documents/Ansible/hello_world.txt
```

```
hello_world.j2
--------------
{{ variable_to_be_replaced }}
This line won't be changed
Variable given as inline - {{ inline_variable }} - :)

output - hello_world.txt
------
Hello world
This line won't be changed
Variable given as inline - hello again - :)
```

Odisee
DE CO-HOGESCHOOL

ANSIBLE

- Roles
  - Zorgt voor “reusable” code
  - “bundelen” van functionaliteiten die bij elkaar horen => roles
  - “Roles” verwacht specifieke folder-structuur!!
  - Folderstructuur kan aangemaakt worden met

```
$ ansible-galaxy init <naam van de rol>
```

  - vb: Rol “LAMP-stack” installeert de componenten nodig voor Lamp (www, db,…)
  - Role aanmaken in een map “roles” relatief t.o.v. locatie playbook

ANSIBLE

- Roles

```
$ ansible-galaxy init lamp
```

```
.
└── lamp
    ├── defaults
    │   └── main.yml
    ├── files
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── README.md
    ├── tasks
    │   └── main.yml
    ├── templates
    ├── tests
    │   ├── inventory
    │   └── test.yml
    └── vars
        └── main.yml
```

Placeholders: is de naam van de file die ansible verwacht bij het uitvoeren van een rol

- `tasks/main.yml` - the main list of tasks that the role executes.
- `handlers/main.yml` - handlers, which may be used within or outside this role.
- `library/my_module.py` - modules, which may be used within this role (see Embedding modules and plugins in roles for more information).
- `defaults/main.yml` - default variables for the role (see Using Variables for more information). These variables have the lowest priority of any variables available, and can be easily overridden by any other variable, including inventory variables.
- `vars/main.yml` - other variables for the role (see Using Variables for more information).
- `files/main.yml` - files that the role deploys.
- `templates/main.yml` - templates that the role deploys.
- `meta/main.yml` - metadata for the role, including role dependencies.

ANSIBLE

Roles

Playbook =>

```
- hosts: webservers
  roles:
     - lamp
     - webservers
```

This designates the following behaviors, for each role ‘x’:

- If roles/x/tasks/main.yml exists, tasks listed therein will be added to the play
- If roles/x/handlers/main.yml exists, handlers listed therein will be added to the play
- If roles/x/vars/main.yml exists, variables listed therein will be added to the play
- If roles/x/defaults/main.yml exists, variables listed therein will be added to the play
- If roles/x/meta/main.yml exists, any role dependencies listed therein will be added to the list of roles (1.3 and later)
- Any copy, script, template or include tasks (in the role) can reference files in roles/x/{files,templates,tasks}/ (dir depends on task) without having to path them relatively or absolutely

ANSIBLE

- Lookups, Loops, conditionals,

Conditional

```
- hosts: all
  vars:
    loop_1: "hello1"
  tasks:
  - name: Ansible loop with conditional example
    debug:
      msg: "{{ item }}"
    with_items:
      - "hello1"
      - "hello2"
      - "hello3"
    when: item == "{{ loop_1 }}"
```

```
tasks:
  - name: Shut down Debian flavored systems
    ansible.builtin.command: /sbin/shutdown -t now
    when: ansible_facts['os_family'] == "Debian"
```

```
vars:
  file_contents: "{{lookup('file', 'path/to/file.txt')}}"
```

Lookup

```
- name: Ansible Loop example
  apt:
    name: "{{ item }}"
    state: present
  with_items:
      - python3
      - ca-certificates
      - git
```

```
- name: Add several users
  ansible.builtin.user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
      - testuser1
      - testuser2
```

Loop

```
- name: Non-optimal yum, slower and may cause issues with interdependencies
  ansible.builtin.yum:
    name: "{{  item  }}"
    state: present
  loop: "{{  list_of_packages  }}"
```

Odisee
DE CO-HOGESCHOOL

ANSIBLE

- Directory-structuur
  - 2 manieren van aanpak

Eenvoudige inventory ⇔ Complexere inventory

ANSIBLE

Eenvoudige inventory

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

group_vars/
   group1                 # here we assign variables to particular groups
   group2                 # ""
host_vars/
   hostname1              # if systems need specific variables, put them here
   hostname2              # ""

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

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

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

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Odisee
DE CO-HOGESCHOOL

ANSIBLE

Complexere inventory

```
inventories/
   production/
      hosts                # inventory file for production servers
      group_vars/
         group1            # here we assign variables to particular groups
         group2            # ""
      host_vars/
         hostname1         # if systems need specific variables, put them here
         hostname2         # ""

   staging/
      hosts                # inventory file for staging environment
      group_vars/
         group1            # here we assign variables to particular groups
         group2            # ""
      host_vars/
         stagehost1        # if systems need specific variables, put them here
         stagehost2        # ""

library/
module_utils/
filter_plugins/

site.yml
webservers.yml
dbservers.yml

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

Odisee
DE CO-HOGESCHOOL