Ansible

Dia 2: PlayBooks

Xavier Sala Pujolar

Universitat de Girona

Maig 2021

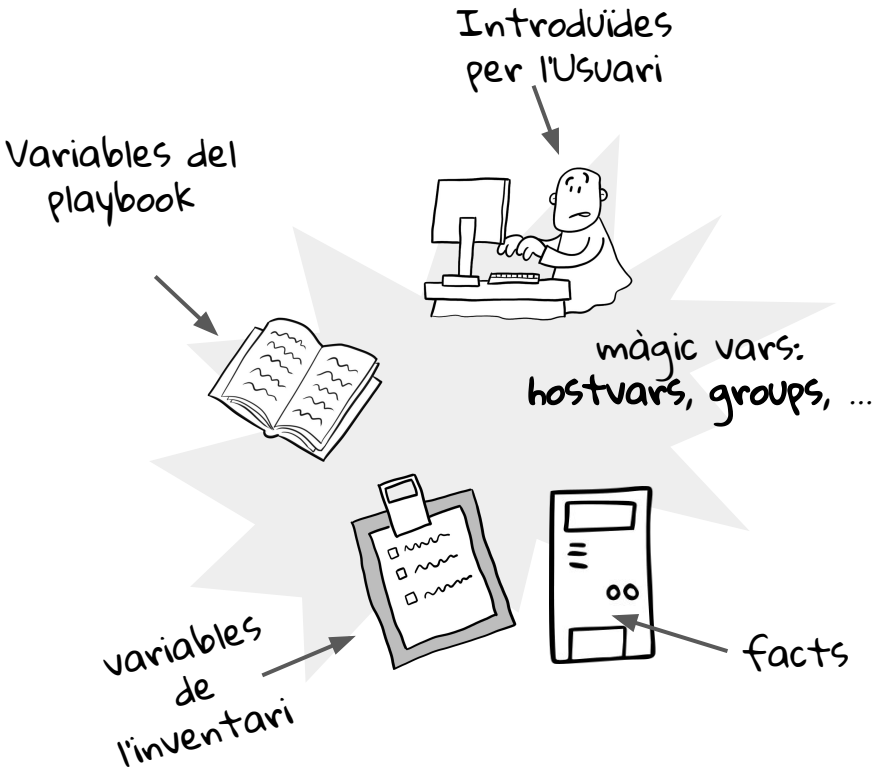
Ansible playbooks

Generalment en automatització hi ha tasques que es repeteixen diverses vegades però sovint és més d'una execució

Els playbooks permeten definir accions formades per diverses passes

Variables

Els playbooks s'alimenten de **metadades** en forma de variables

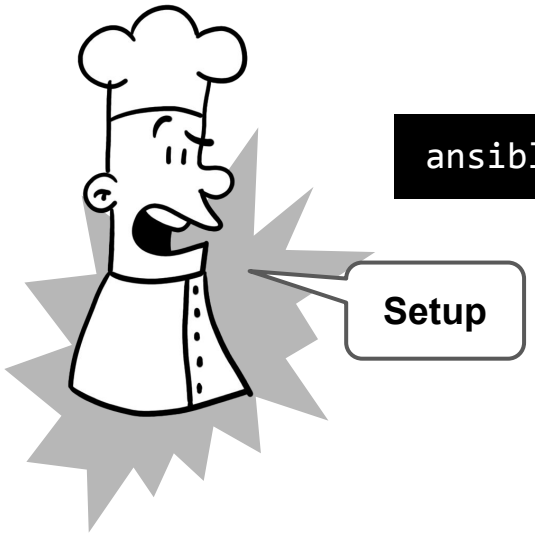


Hi ha uns 20 llocs diferents ...

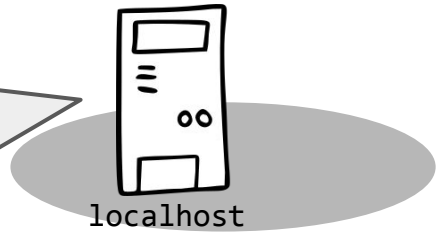
màquina

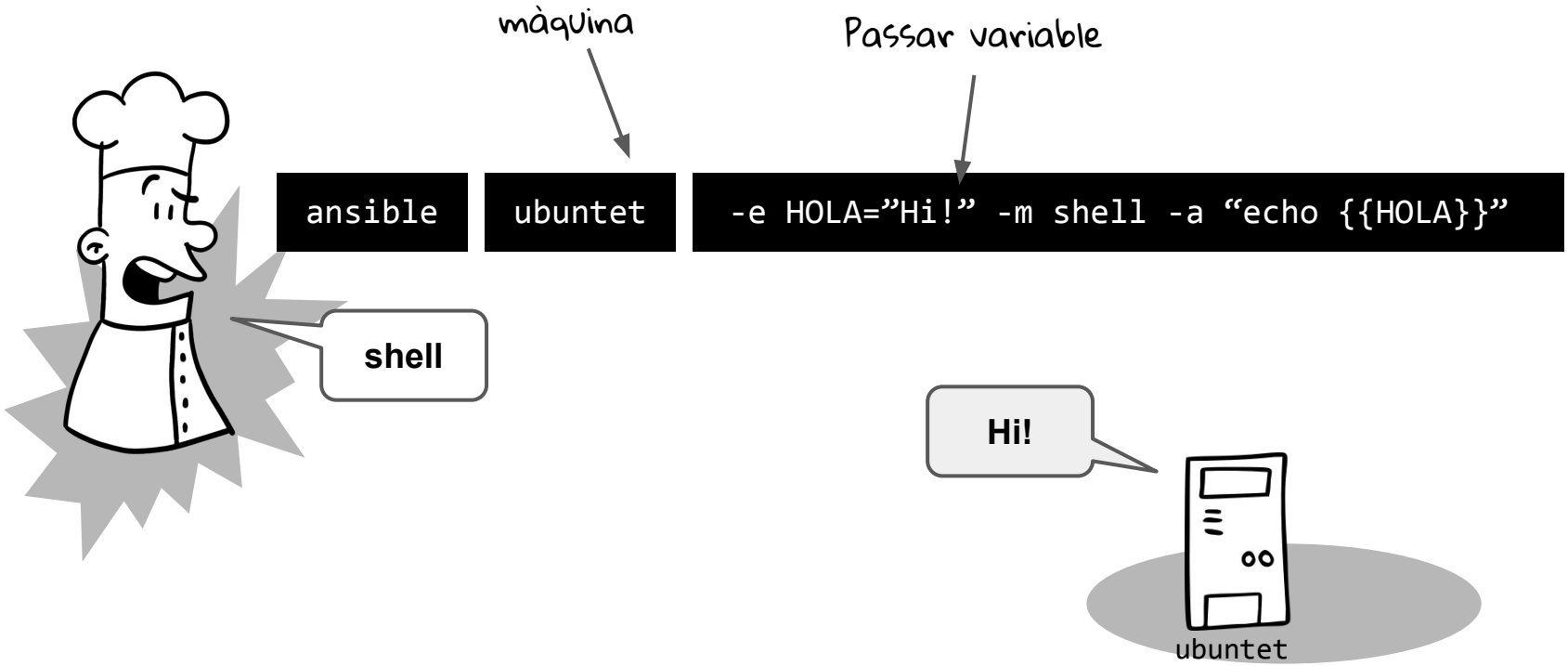
Obtenir facts

```
ansible   xinxan   -m setup
```



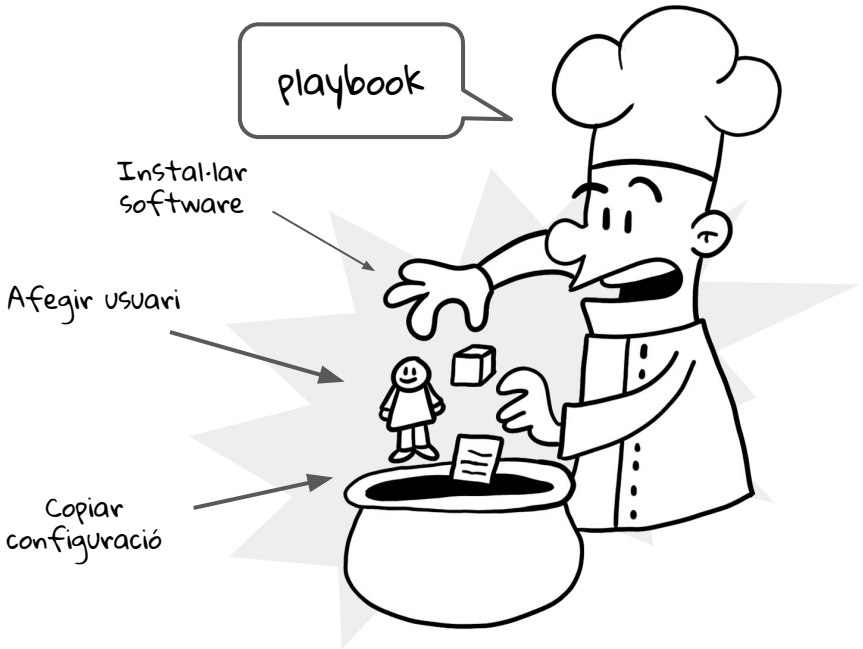

```
xinxan | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.0.108"
        ],
        ...
        "ansible_nodename": "xinxan",
        "ansible_os_family": "Debian",
        "ansible_pkg_mgr": "apt",
        …
}
```



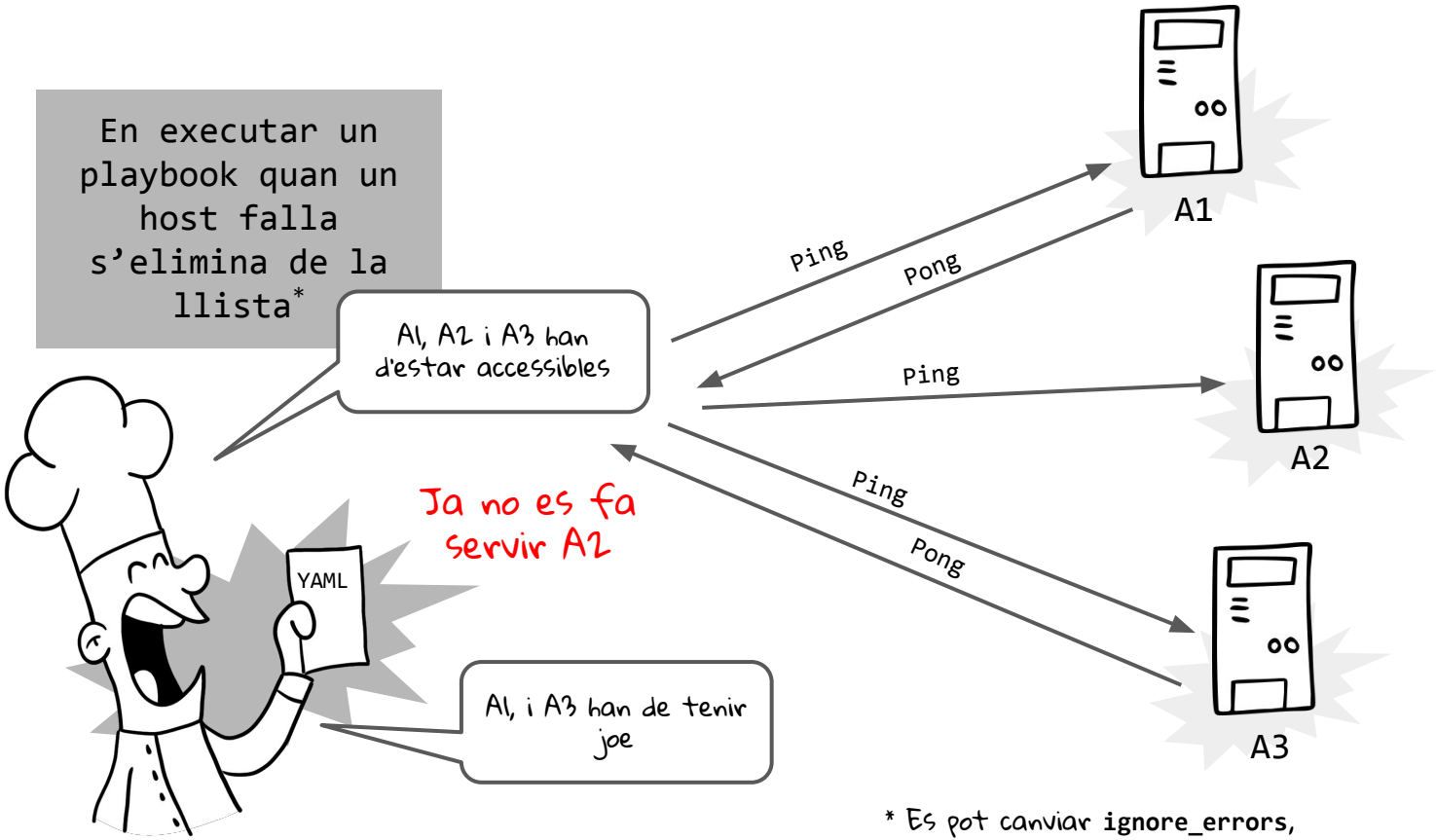

màquina
Passar variable
ansible
ubuntet
-e HOLA=”Hi!” -m shell -a “echo {{HOLA}}”
shell
Hi!
ubuntet

Playbooks

El gran poder d'Ansible ve en les seves **capacitats d'scripting**



En executar un playbook quan un host falla s'elimina de la llista*



* Es pot canviar **ignore_errors**, **any_errors_fatal**, ...

Yaml

Els playbooks es defineixen en **format YAML**

```
---
nom: “Filomeno”
edat: 23

llenguatges:
  - C#
  - Java

departament:
  nom:Informatica
  id: 12

descripció: |
   Ros
   Alt
```

seqüències

```
llenguatges: [c#,Java]
```

mapes

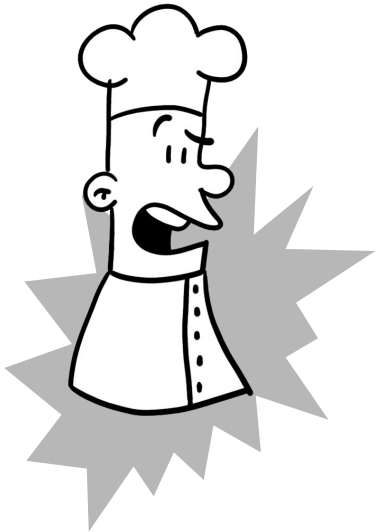
```
departament:
{nom:Informatica, id:12}
```

Playbook

alive.**yml**

```
---
- hosts: ubuntet
  user: pi

  tasks:
   - name: Comprovar que està viu
     ping:
```

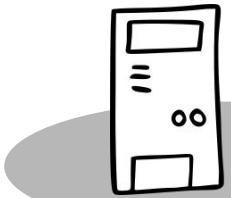
Executar playbooks



```
ansible-playbook   -i hosts   alive.yml
```

playbook

inventari



Ubuntet

Un fitxer pot tenir diversos plays

qualsevol.yml

```
---

- hosts: xinxan.local
  user: pi

  ...

- hosts: bulma.local
  user: xavier

  ...
```

Els **plays** es formen amb diferents seccions

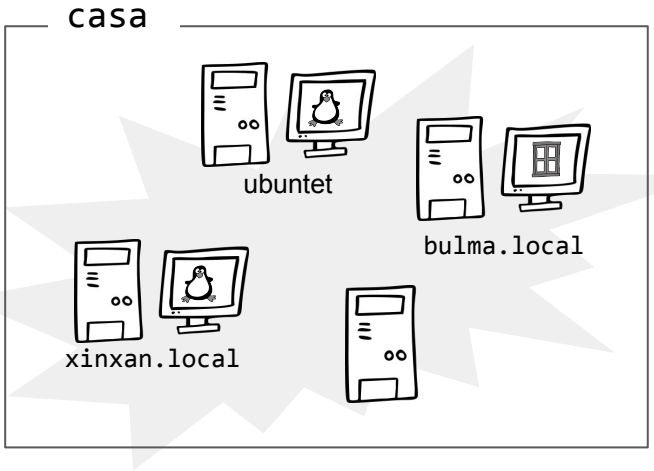
Play

joe.yml

```
---
- hosts: xinxan.local
  user: pi

  vars:
   - editor: joe


  tasks:
   - name: Comprovar que està viu
     ping:

   - name: Instal·lar {{editor}}
     apt:
        name={{editor}}
        state=present
     become: yes
```

Target section

Defineix on i de quina forma connectar



Grups, llistes de hosts, ...

```
- hosts: casa
```

```
- hosts:
  - xinxan.local
  - ubuntet
```

```
- hosts: xinxan.local
  user: pi
  become: yes
```

Dades extres

Vars section

Variables que es poden usar en tots els hosts que participin en el play

```
- vars:
  - usuari: pep
  - programes:
     - joe
     - tilix
     - exa
```

Fitxers amb les variables →

```
- vars_files:
  - fitxer_1.yml
```

Demana la variable →

```
- vars_prompt:
   - name: contrasenya
     prompt: contrasenya?
     private: yes
```

Tasks section

Sempre s'executen en ordre

joe.yml

```
---
- hosts: xinxan.local
  user: pi

  vars:
   - editor: joe

  tasks:
   - name: Comprovar que està viu
     ping:

   - name: Instal·lar {{editor}}
     apt:
       name={{editor}}
       state=present
     become: yes
```

Handlers section

Permet "disparar" accions en cas d'èxit d'una tasca

joe.yml

```
---
- hosts: xinxan.local
  user: pi
  become: yes

  tasks:
  - name: instal·lar apache
     apt:
       name: apache2
       state: present
     notify: reiniciar apache

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

Condicionals i bucles

install_joe.yml

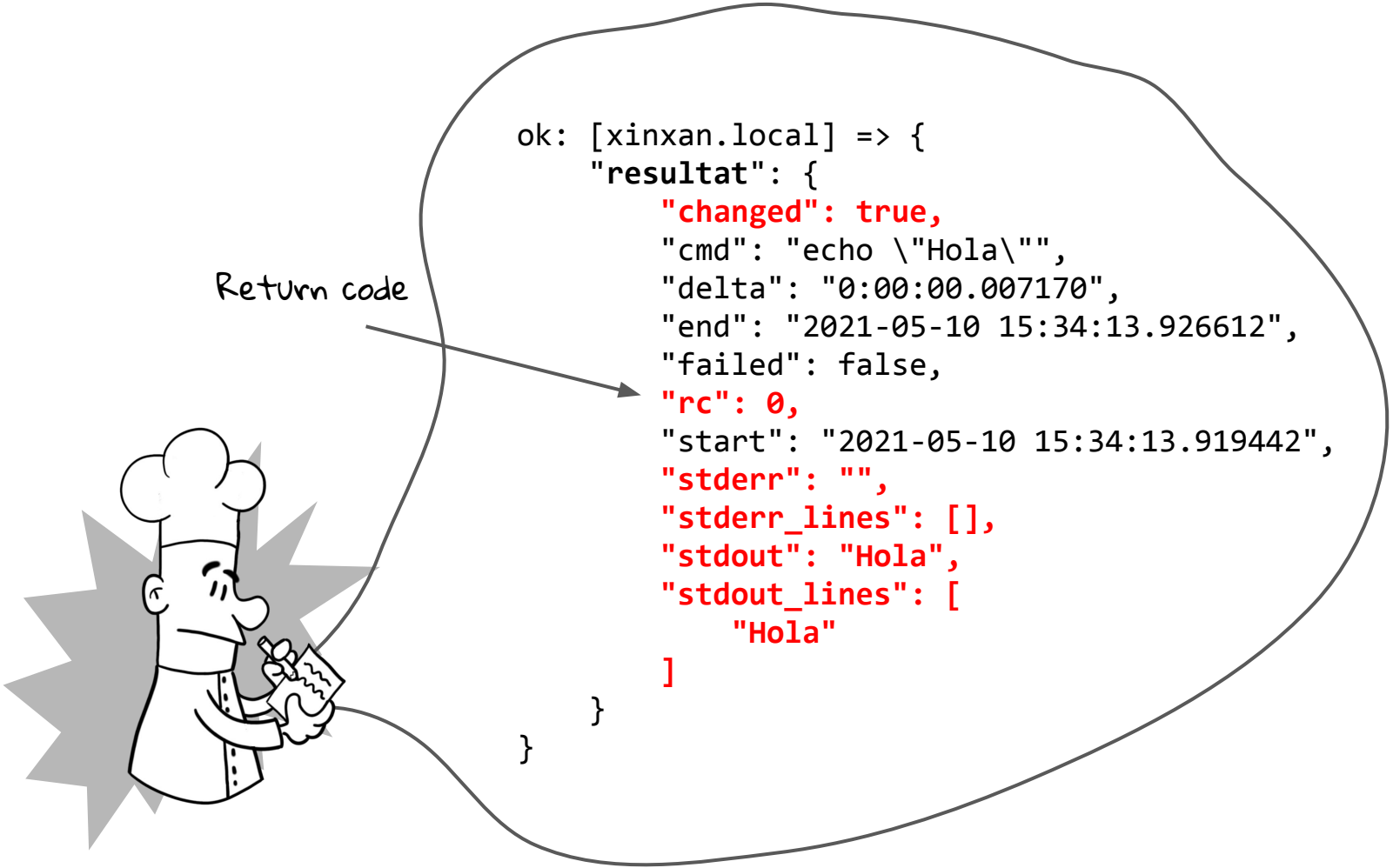
```
---
- hosts: xinxan.local
  user: pi

  tasks:
   - name: Instal·lar Joe apt
     apt: pkg=joe state=present
     become: yes
     when: ansible_os_family == “Debian”

   - name: Instal·lar Joe yum
     yum: pkg=joe state=present
     become: yes
     when: ansible_os_family == “RedHat”
```

register permet guardar informació del resultat d'execucions

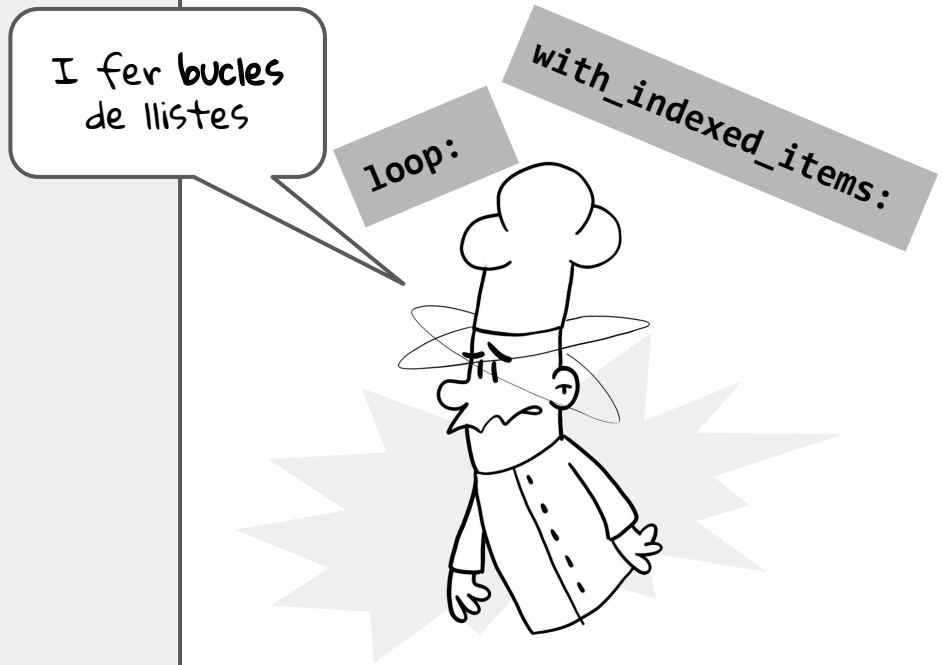
prova_register.yml

```
---
- hosts: xinxan.local
  user: pi

  tasks:
   - name: pintar Hola
     shell:
       echo Hola
     register: resultat

   - name: Ha funcionat tot bé
     debug:
       var: resultat
     when: resultat.rc == 0
```

Que es pot usar en altres tasques


Return code
ok: [xinxan.local] => {
"resultat": {
"changed": true,
"cmd": "echo \"Hola\"",
"delta": "0:00:00.007170",
"end": "2021-05-10 15:34:13.926612",
"failed": false,
"rc": 0,
"start": "2021-05-10 15:34:13.919442",
"stderr": "",
"stderr_lines": [],
"stdout": "Hola",
"stdout_lines": [
"Hola"
]
}
}

```
---
- hosts: xinxan.local
  user: pi

  vars:
    programs:
      - joe
      - exa

  tasks:
   - name: Instal·lar
     apt:
       name: {{ item }}
       state=present
     become: yes
     with_items: "{{ programs }}"
```



Template Module

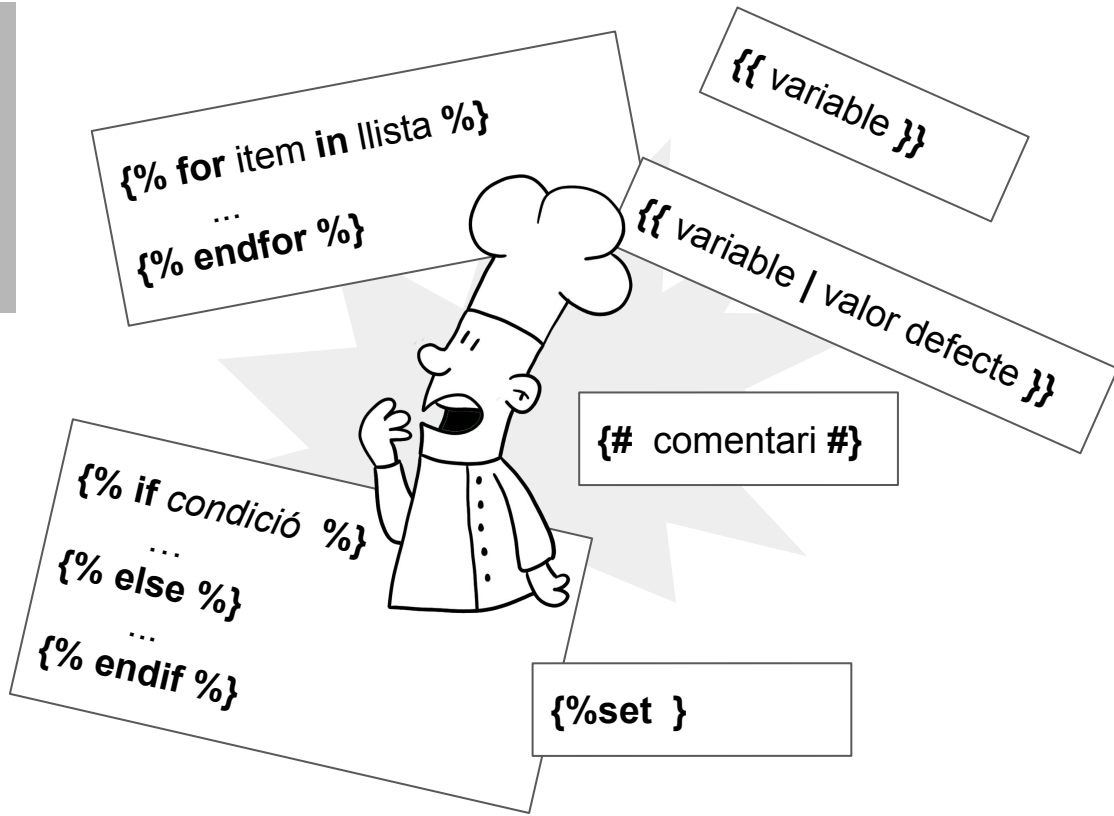
Permet definir **plantilles** que Ansible emplenarà en temps d'execució

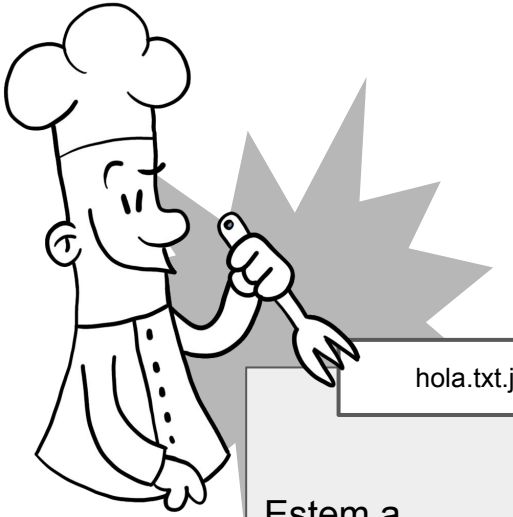
Es defineixen en format Jinja2

fitxer.conf.j2

```
# {{ ansible_managed }}
options {
   listen-on port 53 {
      127.0.0.1;
      {%for ip in ansible_all_ipv4_addresses %}
           {{ ip }};
      {% endfor % }
   };
   directory "/var/named";
```

Jinja2 permet fer servir expressions dinàmiques i mostrar variables





hola.txt.j2

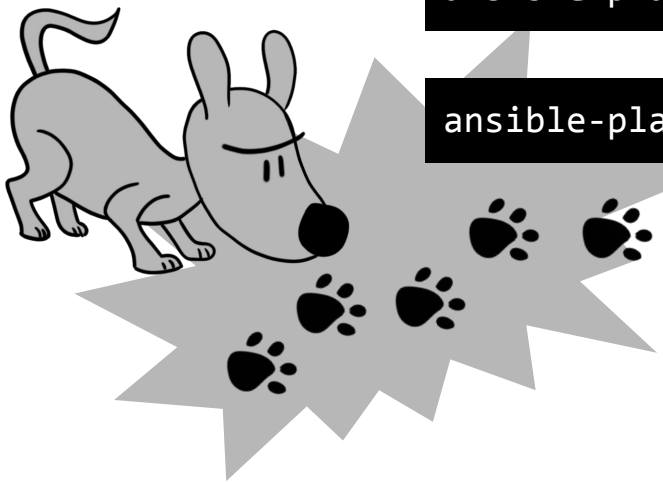
Estem a
{{ansible_nodename}}

templating.yml

```
---
- hosts: xinxan.local
  user: pi

  vars:
   - user: ansible_user

  tasks:
   - name: Copiar fitxer identity
     template:
        src: /templates/hola.txt.j2
        dest: /home/{{user}}/hola.txt
        owner: {{user}}
        mode: 0400
```

Debug Playbooks


```
---
- hosts: xinxan.local

  tasks:
   - name: Copiar fitxer identity
     debug:
        msg: “{{ansible_nodename}}!”

   - name: imprimir variable
     debug:
        var: ansible_user
```

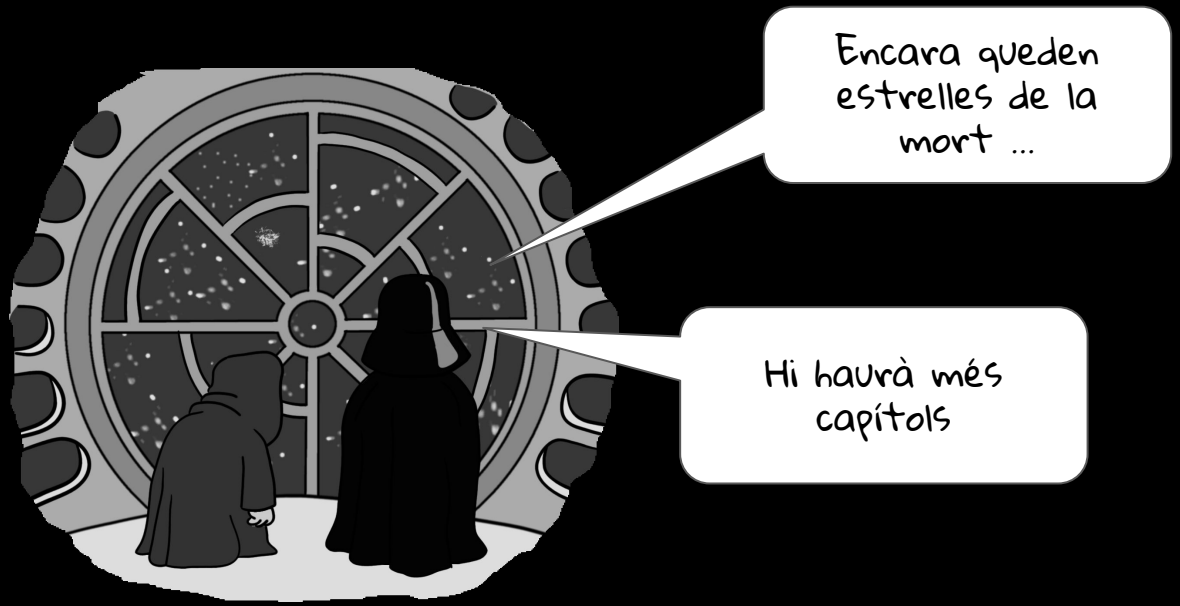


```
ansible-playbook fitxer.yml --syntax-check
```

```
ansible-playbook fitxer.yml --check --diff
```

Verifica sintaxi

Simula l'execució



2021