… et comment s’en passer

- Vagrant (Ruby)
- Packer (Go)
- Terraform (Go)
- Consul (Go)
- Vault (Go)
- Nomad (Go)

HashiCorp Terraform

- oss ➡ freemium (racheté par IBM)
- initialement Mozilla Public License (MPL 2.0)
  - depuis août 2023 Business Source License (BUSL)
    - Terraform != Terraform cloud
      - Weighing Terraform Cloud alternatives after the recent price hike?
- OpenTofu (Linux Foundation / OpenTofu)

HashiCorp
Terraform

- IaC pour IaaS
- providers
  - ressource types
  - data sources
- state
  - cache
    - code VS reality => conform
  - json

HashiCorp
Terraform

HCL *.tf

- https://github.com/hashicorp/hcl
- utiliser
  - config
  - description
- fichiers .tf
  - `apply`
    - mise en conformité / state

- Telmate/terraform-provider-proxmox
- Telmate/proxmox-api-go
  - Provisionner des VM avec Terraform
- claudusd/terraform-windows-network
  - Active Directory (DHCP/DNS)

Cluster hypervision != Cloud

Stack cli (legacy)

- cli wrapper
  - terraform
  - ansible (racheté par IBM)
  - vault
  - bash
- workflow
  - build (terraform)
  - play (ansible)
  - destroy (terraform)

Stack cli (legacy)

```
workspaces/
├── ansible/
│   ├── host_vars/
│   ├── group_vars/
│   ├── playbooks/
│   │   └── my_project.yml
│   └── inventory.yml
├── terraform/
│   ├── my_project/
│   │   ├── ci-my_project.template
│   │   ├── ci-my_project.tf
│   │   ├── dhcp-my_project.tf
│   │   ├── dns-my_project.tf
│   │   └── vm-my_project.tf
```

Gestion massive de ressources ~600

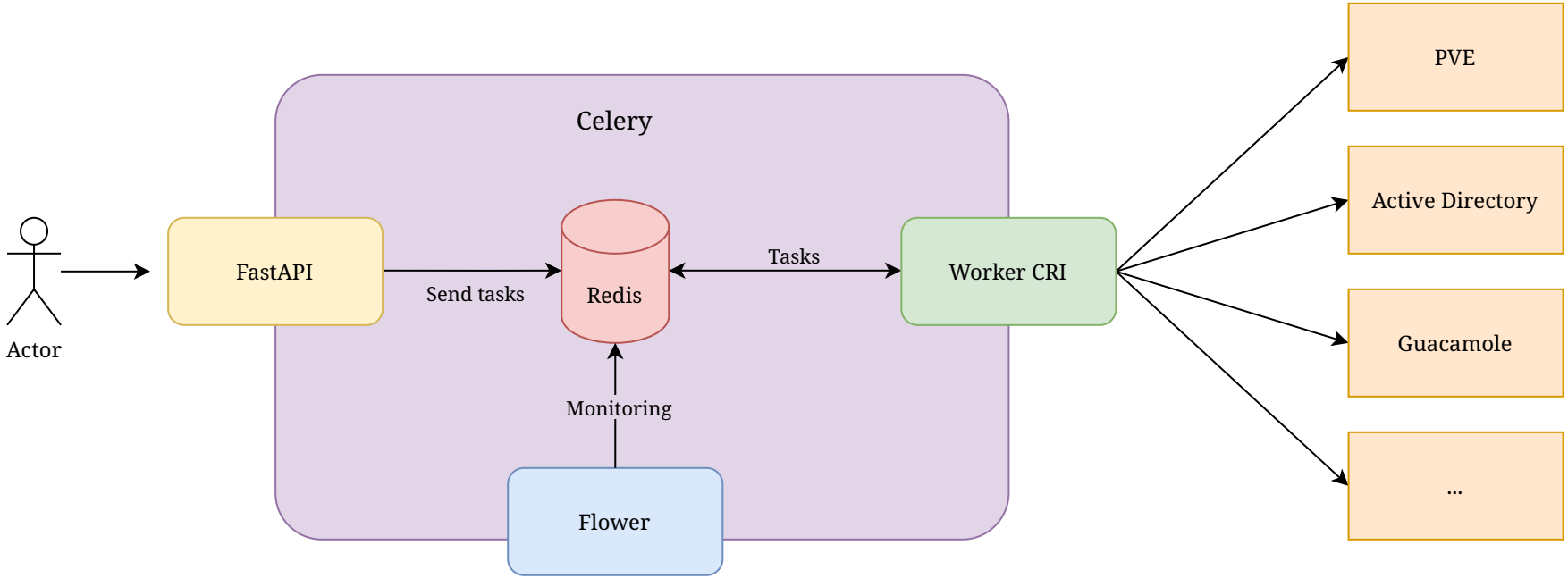
- création en masse VMs étudiantes :/
- overhead list pools ceph
  - temps de réponse API Proxmox
- backend consul (état partagé) limité
- git pull *.tf pour le collaboratif
- temps de création / VM
  - parallélisation ~5

Gestion massive de ressources ~600

- création en masse VMs étudiantes :/
- overhead list pools ceph
  - temps de réponse API Proxmox
- backend consul (état partagé) limité
- git pull *.tf pour le collaboratif
- temps de création / VM
  - parallélisation ~5

What Next? Hack together!

Architecture logicielle



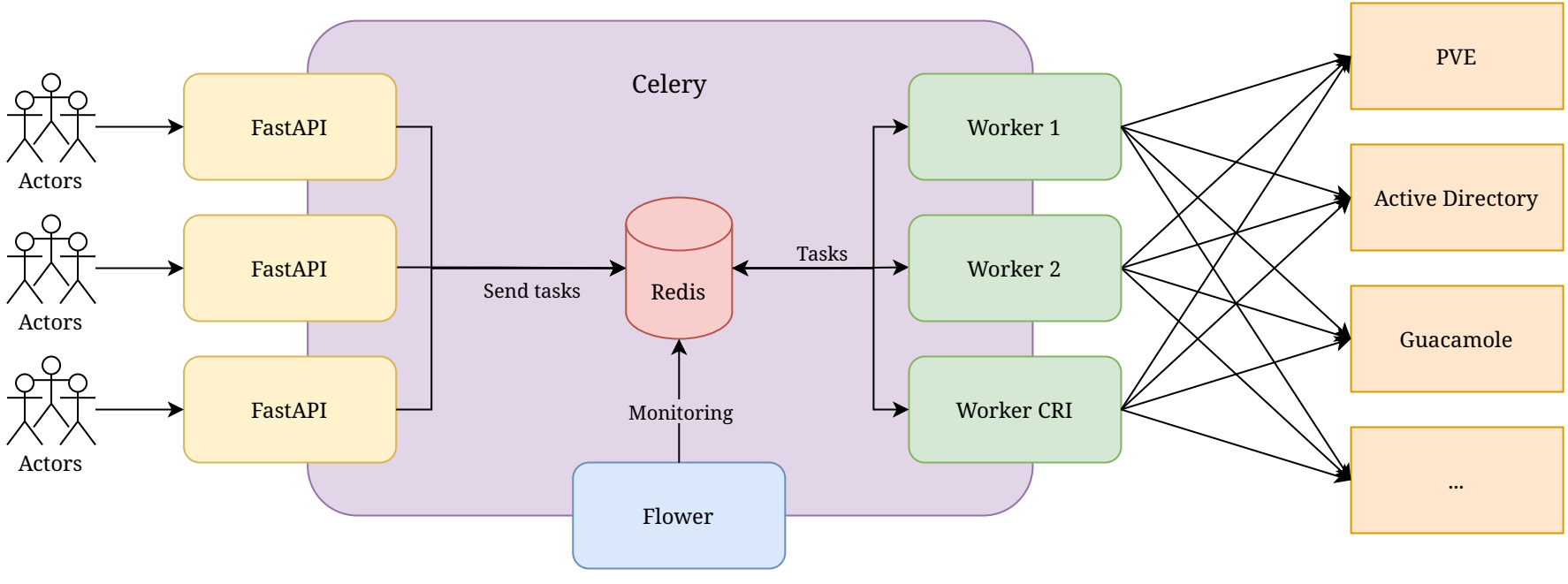
- Utilisation via fastAPI
- Celery = Ordonnanceur de tâches parallèles
- Tâches peuvent être longues

bruno

- Open source API Client
- Gitable
- Test suite
- Jeune mais prometteur (v0.1.0 - Octobre 2022)

600 utilisateurs potentiels

- Trop lent
- Parallèliser le code

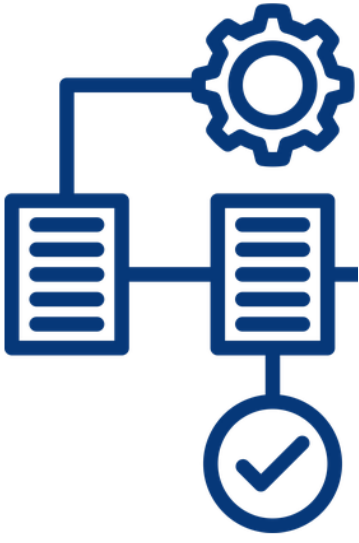


Le drame

- Création de VM avec le même VMID
- Timeout du lock sur les templates
- Démarrage simultané

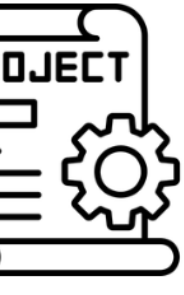
–> Utilisation de verrous via Redis

Création de Workflows



VM Etudiant

- Réservation VMID
- Clonage du template
- Autorisation DHCP
- Enregistrement dans guacamole



Projet

- Création d’une VM
- Enregistrement DNS
- Playbook ansible

VM étudiant / LXC Debian

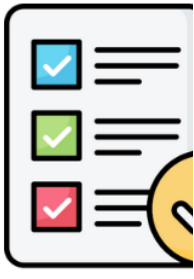
- 2 types de conteneurs
  - clone d'un template préparé maison avec GUI (cinammon)
  - debian12 de base

=> call api PVE différent entre les deux

- Personnalisation des vm au 1er boot
  - hook lxc mount
  - hook lxc start

FastStack -> Cluster PVE

PROXMOX



- Ensemble de petites tâches
- Requêtes d’API
  - Détails du cluster pve
  - Récupération du prochain vmid disponible
  - Création de conteneur LXC
  - Démarrage/arrêt des machines virtuelles
  - etc.

FastStack -> Cluster PVE

PROXMOX

- Ensemble de petites tâches
- Commandes systèmes par ssh
  - Mise à jour des fichiers de config lxc
  - Téléversement des scripts de provisioning (cloud-init/lxc hook)
  - ...

Configuration LXC (par ssh)

- Mappage high uid/gid
  - Même UID hors et dans le conteneur
  - Requis pour la connexion et pour l’accès au dossier personnel

- Ajout dossier personnel
  - Montage NFS sur le PVE
  - Montage du dossier PVE -> Conteneur LXC

- hook scripts LXC
  - Personnalisation du conteneur

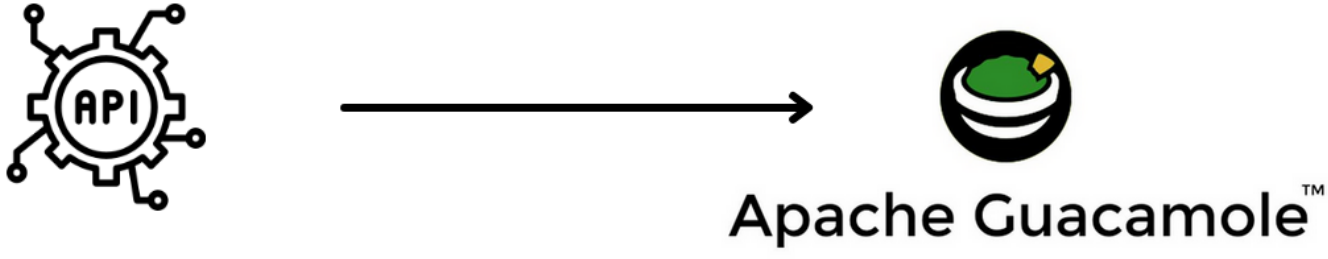
Script de démarrage LXC (lxc.hook.start)

- Mise à jour système
- Installation de paquets
- Intégration au domaine
- Restriction d’authentification
- dhclient -i : use DUID (dhcp server create dns entry when dhcp offer)

Configuration LXC créée :

```
CT perso pour Raphael AMATO - cri
rch: amd64
ores: 4
eatures: nesting=1
ostname: vm-rapamato
emory: 12288
et0: name=eth0,bridge=vmbr285,hwaddr=BC:24:11:B3:C1:3D,type=veth
stype: debian
ootfs: disk1:vm-50003-disk-1,size=50G
wap: 512
nprivileged: 1
xc.idmap: u 0 100000 65536
xc.idmap: g 0 100000 65536
xc.idmap: u 600000000 600000000 99999999
```

FastStack -> Serveur Guacamole



- Requêtes d’API (pyguacamole)

FastStack -> Serveurs Windows (LDAP/DHCP/DNS)

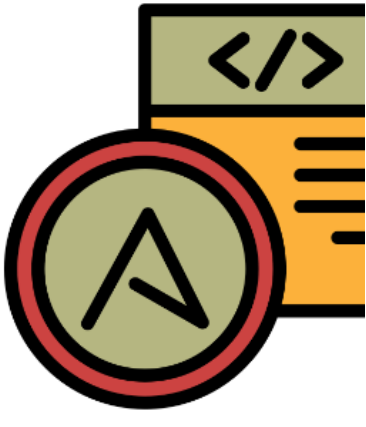
Windows Server

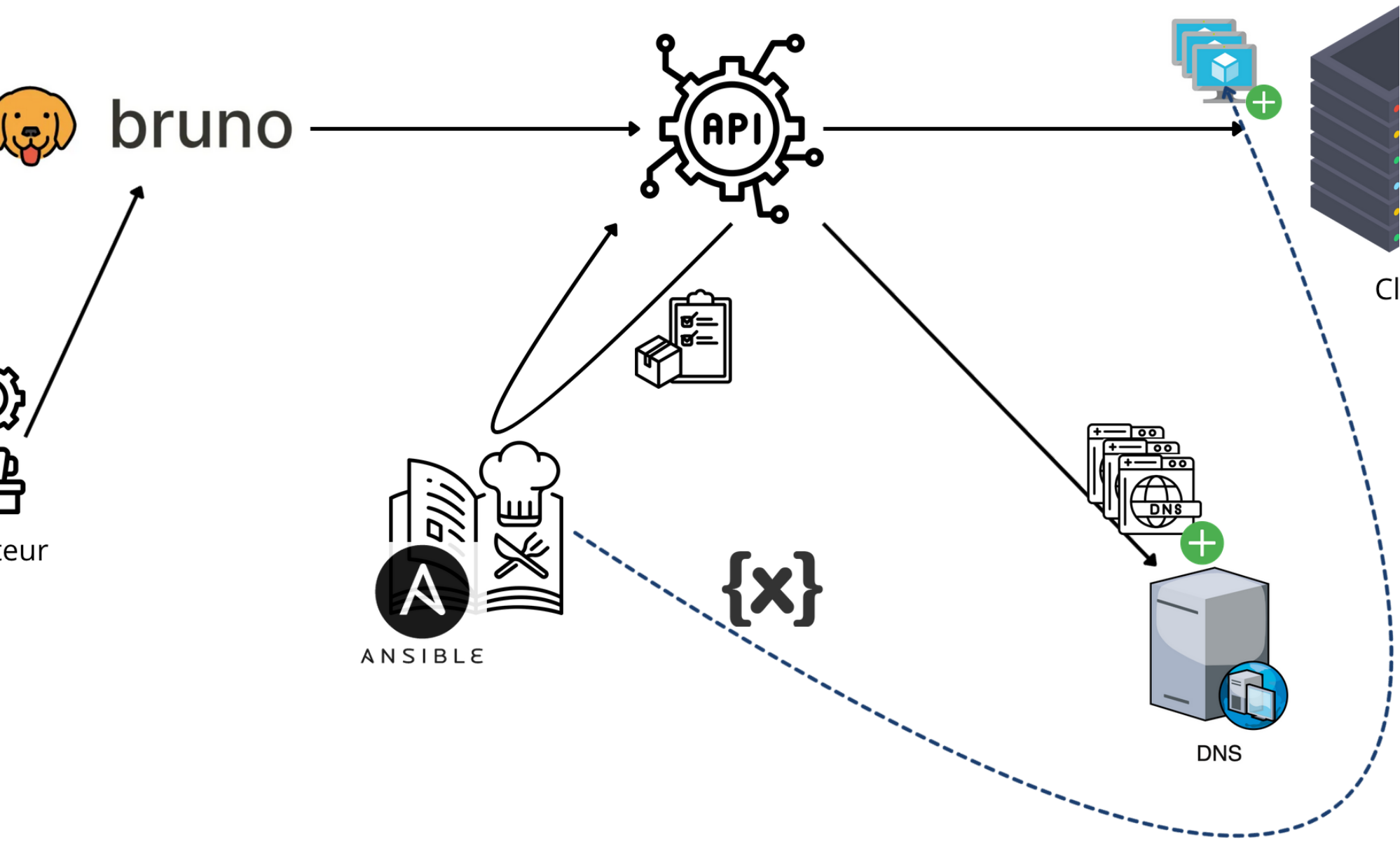
- Connexion WinRM (pywinrm)
  - commandes systèmes PowerShell

WorkFlow Projet

- Injection de fichier Yaml
  - Définition des ressources VM
  - Définition des playbook ansible
  - Ajout d’ extravars Ansible

- Création de VM QEMU/LXC
- Enregistrement DNS
- Lancement de playbook Ansible




bruno
API
teur
ANSIBLE
{x}
DNS
DNS
Cl

Workflow IaaC : import yaml

ct/vm definition

```
ct:
  - description: "Profan - Cloud - PROD"
    node: "pve2"
    memory: "4096"
    cores: "4"
    hostname: "cloud.prod.pft.cri.local.isima.fr"
    base: "dirs-nfs:debian-11-standard"
    storage: "vol-ceph-1"
    net0: "vmbr279"
    rootfs: "300"
    ip: "192.168.79.170/24"
    gw: "192.168.79.254"
    ssh-public-keys: "limosadm"
```

Section Ansible avec extravars

```
cluster: cri
workspace: projects
ansible:
  group: "profan_prod"
  playbook: profan-transfert.yml
  tags: [ 'sudoer', 'pip', 'docker']
  extravars:
    ssh:
        port: 22
        permit_root_login: 'yes'
        allow_agent_forwarding: 'yes'
```

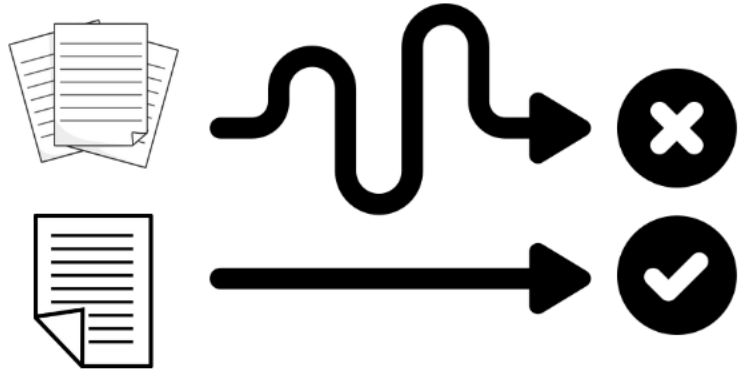
Bénéfices FastStack vs oldStack (terraform)

- Gain de temps déploiement VM :
  - oldStack : + d’1h pour ~100vms
  - newstack : ~15minutes pour ~100vms

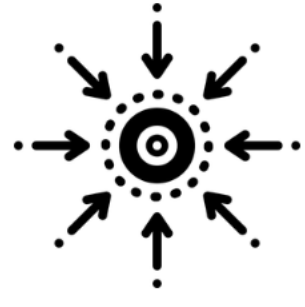
- Autonomie des étudiants (création VM, gestion start/stop/snapshot)

Bénéfices FastStack vs oldStack (terraform)

- Simplification pour l’équipe info
  - 1 seul fichier de déclaration/projet



- Centralisation des tâches



What’s next ?

- Gestion certificat
- Gestion sauvegardes
- Gestion secret vault
- Consolidation msg retour FastStack
- Délégation Ansible Tower ?

Merci

Des questions ?