

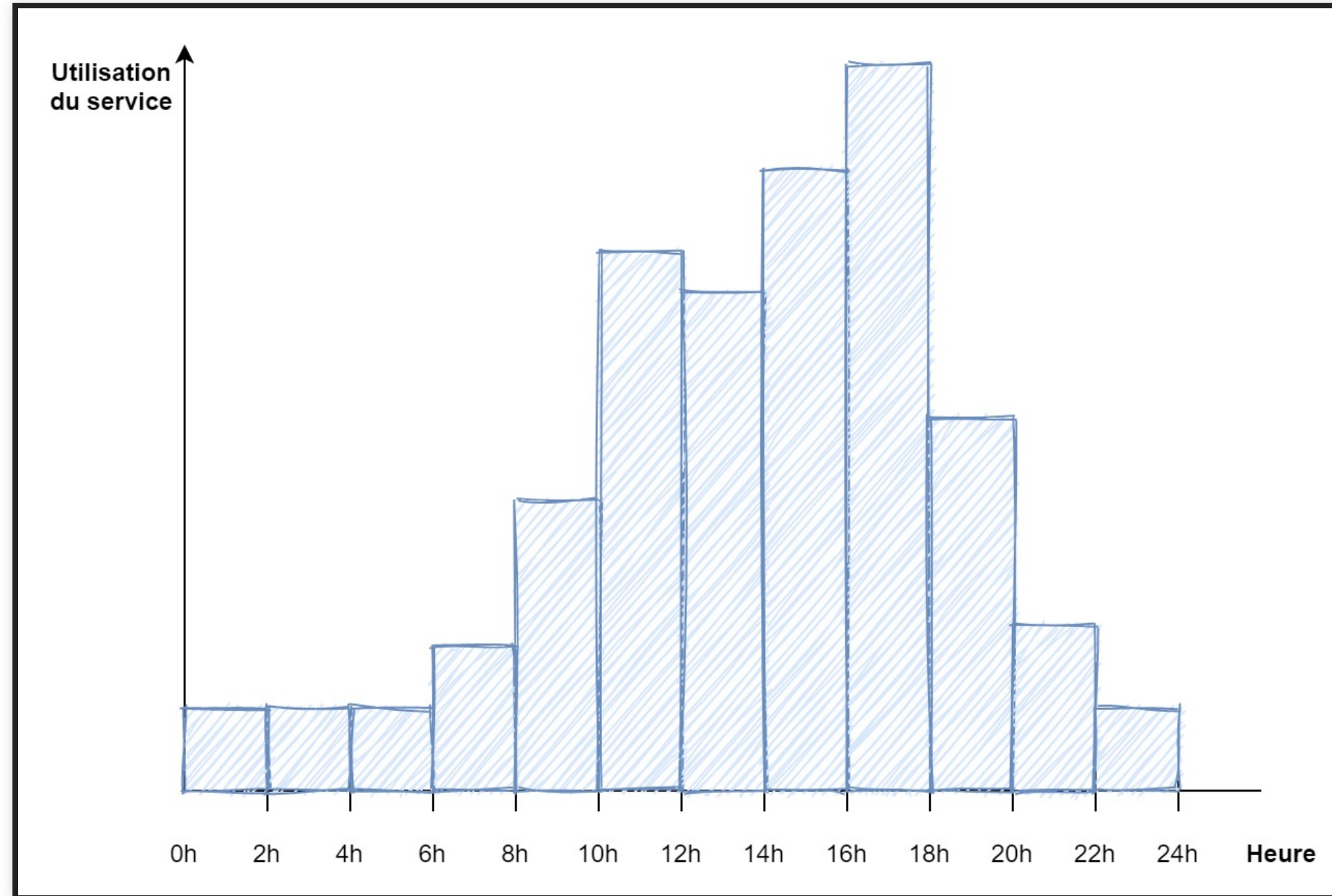
CLOUD COMPUTING, CM2

Pépin Rémi, Ensai, 2023

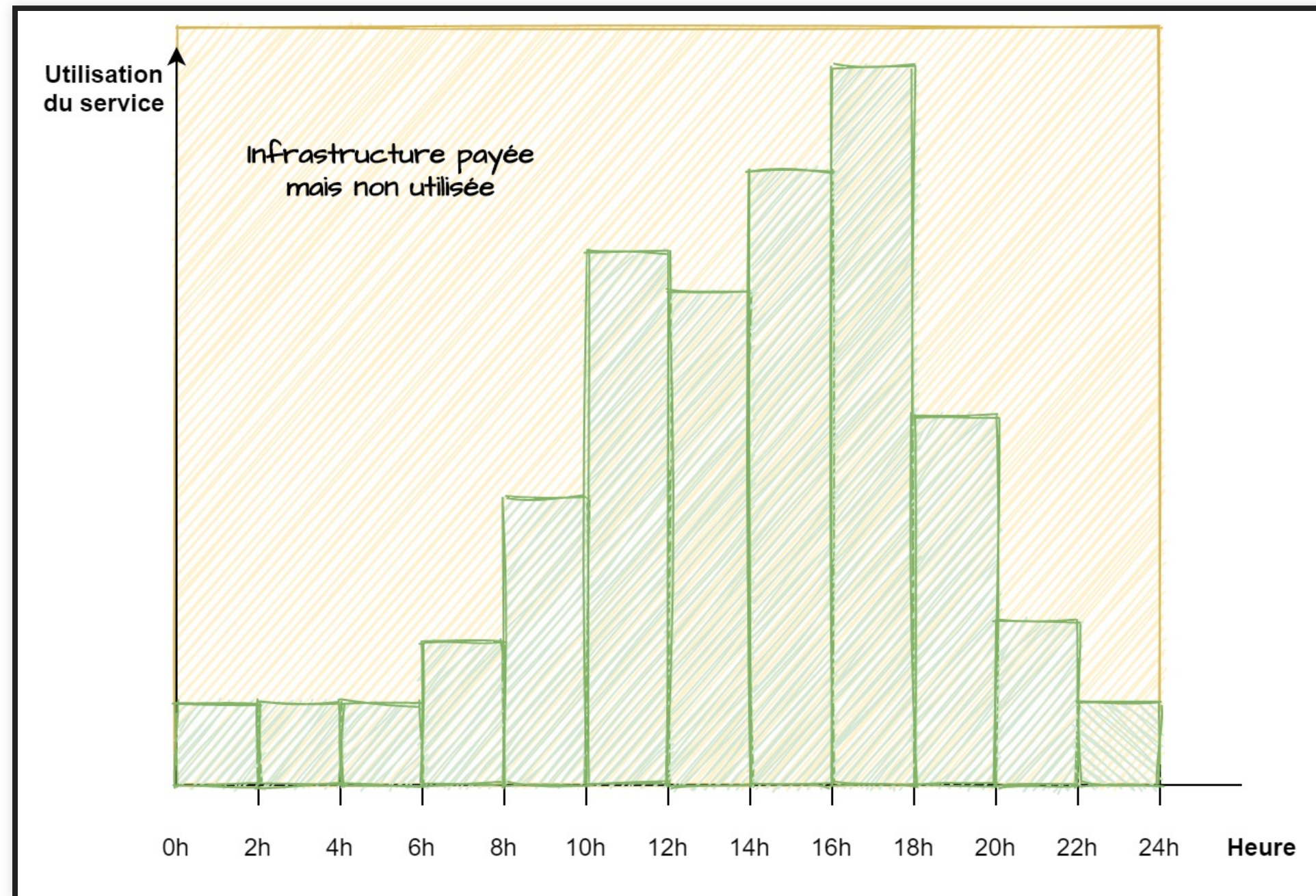
remi.pepin@ensai.fr

HAUTE DISPONIBILITÉ

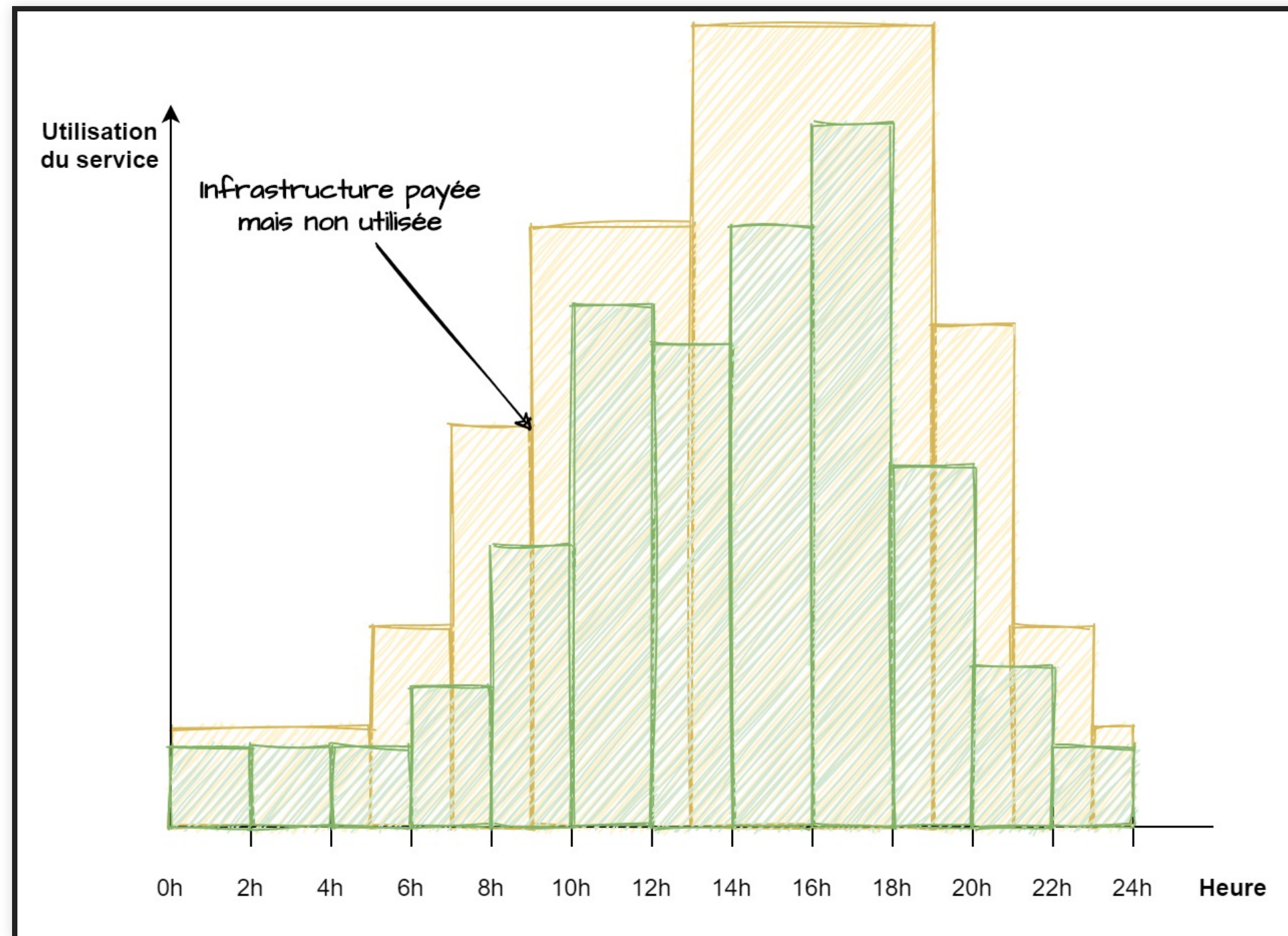
UTILISATION D'UN SERVICE AU COURS DU TEMPS



DÉPLOIEMENT SUR UNE SEULE MACHINE



DÉPLOIEMENT SUR UNE FLOTTE ÉLASTIQUE DE MACHINE



MODÈLE DE DÉPLOIEMENT

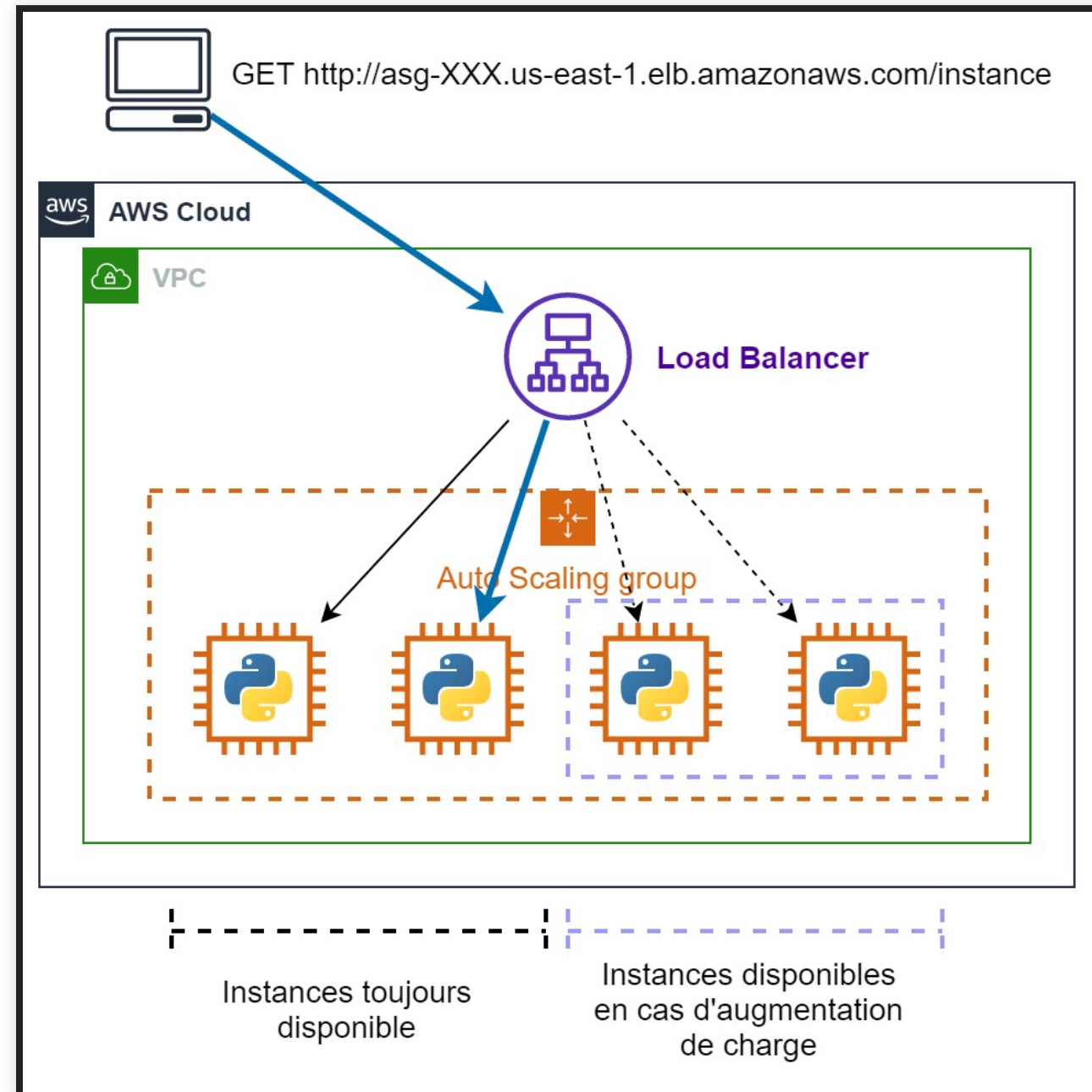
Machine unique

- **Désavantages**
 - Facturation élevée
 - Reprise sur erreur
- **Avantages**
 - Simple à mettre en place
 - Application statefull

Flotte de machine

- **Désavantages**
 - Ajoute de la complexité
 - Monitorer une flotte de machine
 - Application stateless
- **Avantages**
 - Permet de faire des économies
 - Permet haute disponibilité

AUTO-SCALING GROUP + LOAD BALANCER



AUTO-SCALING GROUP + LOAD BALANCER

- Auto-Scaling Group : gère une flotte d'EC2. Peut en lancer/éteindre en fonction de conditions (heures, utilisation CPU, RAM), ou si une machine est considérée comme ko
- Load Balancer : composant qui répartit le trafic entre plusieurs machines. Permet un point d'entrée unique pour accéder à une flotte. Équilibre le trafic, et peut renvoyer un utilisateur vers la même machine (sticky session)






AUTO-SCALING GROUP + LOAD BALANCER

- Haute disponibilité
- Élasticité horizontale
- Permet de faire des économies
- Nécessite d'être pris en compte dans le code
- Pour la partie traitement, pas stockage

Pattern courant pour assurer haute disponibilité

INFRASTRUCTURE AS CODE (IAC)

UN CONSTAT

-  Fastidieux
-  Réplication ?
-  Versionnable ?
-  Orchestration ?
-  Ai-je bien tout éteint ?

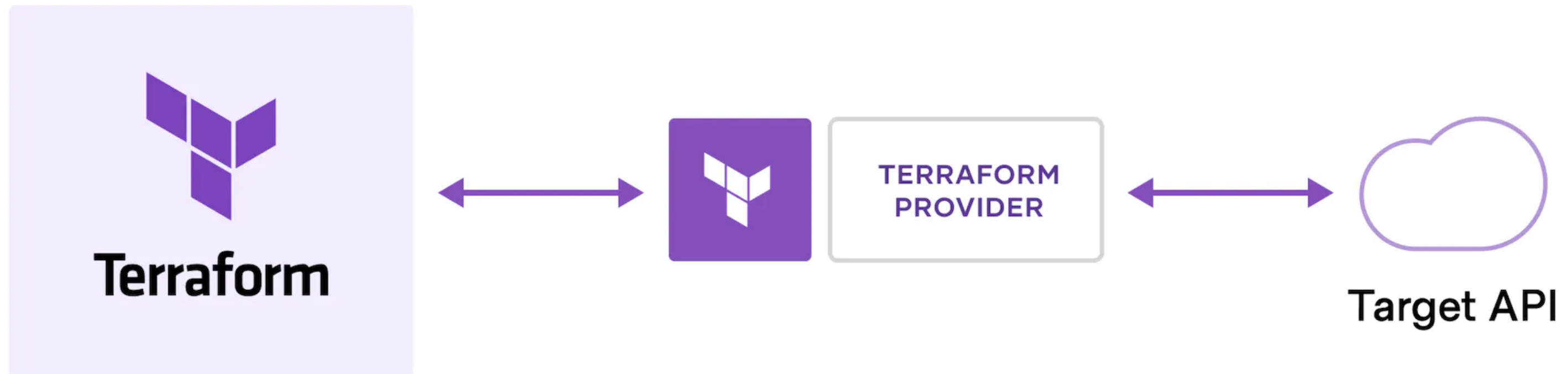
INFRASTRUCTURE AS CODE

L'Infrastructure as Code (IaC) consiste à définir son architecture informatique comme du code. Au lieu de naviguer sur une interface, les services à déployer seront définis dans des fichiers via un langage spécifique. Ce code pourra ainsi être versionné et redéployé à l'infini.

- Terraform (cross plateforme)
- Amazon Cloud Formation, Amazon Serverless Application Manager
- Azure Resource Manager
- Google Cloud Deployment Manager

TERRAFORM

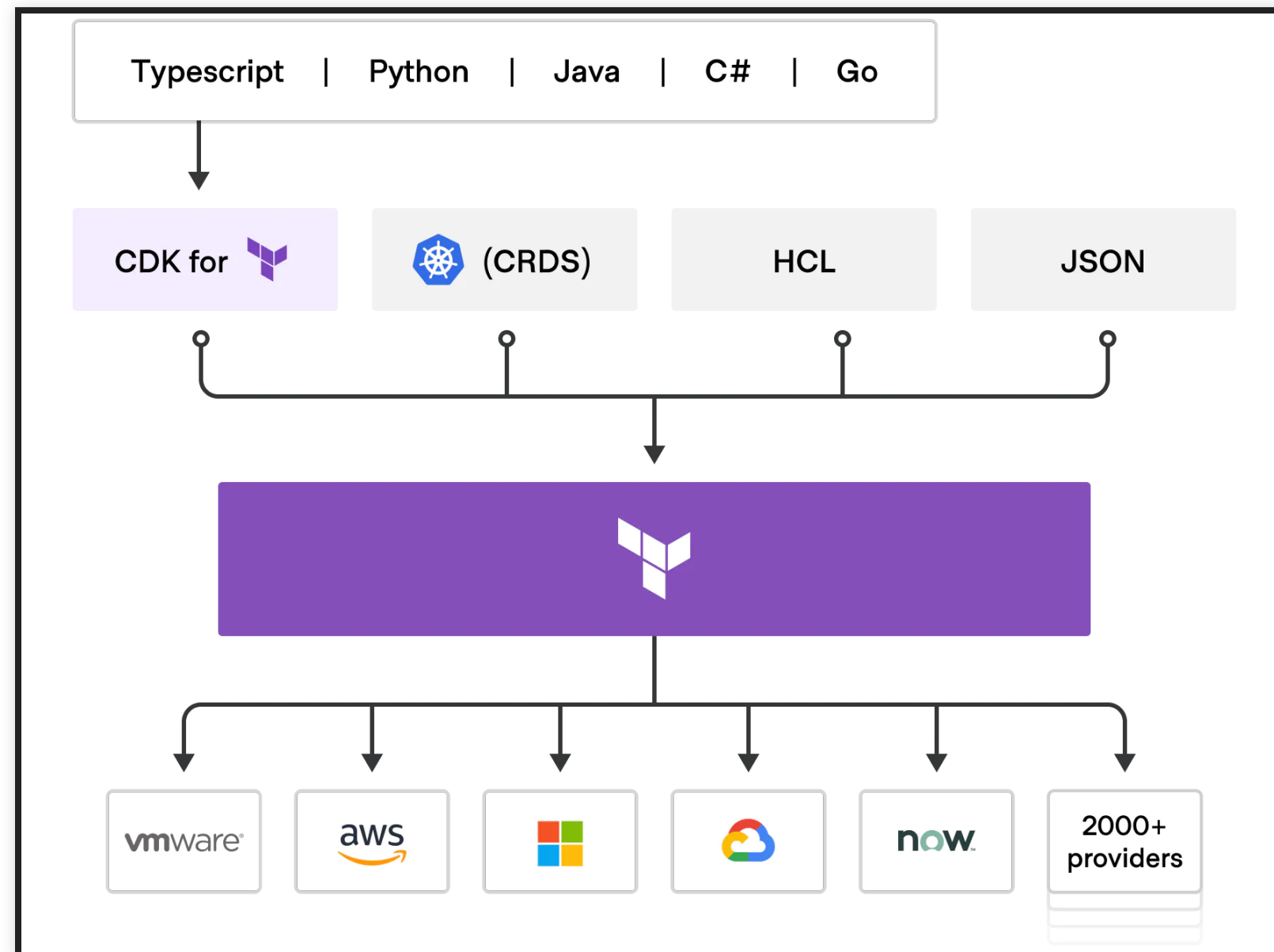
- Projet open source crée par HashiCorp
- Langage de configuration / CDK pour infra
- Terraform provider pour traduction
- Utilise les API des différents systèmes



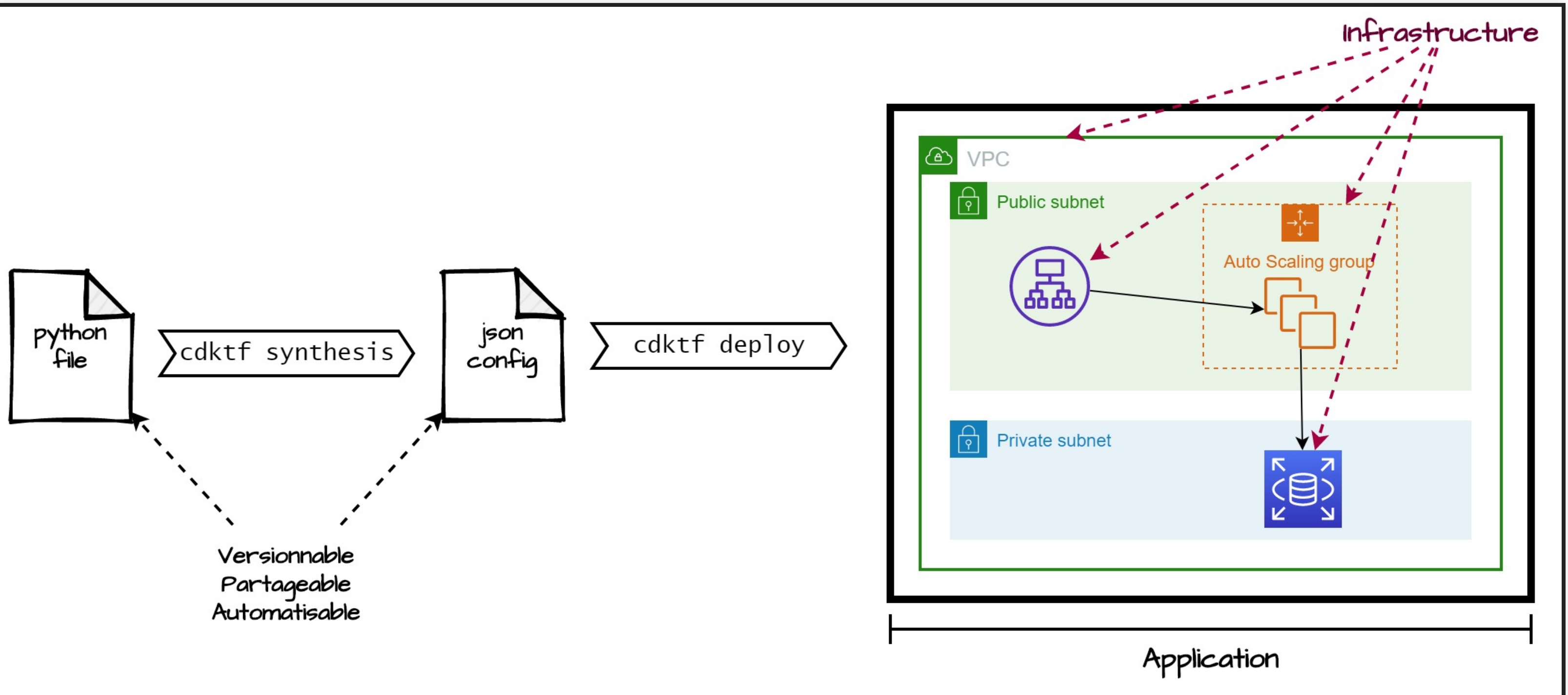
CONFIGURATION LANGUAGE VS CDKTF

- HashiCorp Configuration Language (HCL) : pas json ni yaml (existe une version json)
- CDKTF : bibliothèques pour coder son architecture dans son langage de prédilection

Pour ne pas apprendre un nouveau langage, nous allons utiliser le CDKTF



CDKTF



EXAMPLE

```
from constructs import Construct
from cdktf import App, NamedRemoteWorkspace, TerraformStack, TerraformOutput, RemoteBackend
from cdktf_cdktf_provider_aws.provider import AwsProvider
from cdktf_cdktf_provider_aws.instance import Instance

class MyStack(TerraformStack):
    def __init__(self, scope: Construct, ns: str):
        super().__init__(scope, ns)

        AwsProvider(self, "AWS", region="us-west-1")

        instance = Instance(self, "compute",
                             ami="ami-01456a894f71116f2",
                             instance_type="t2.micro",
```

Plus qu'a faire un "cdktf deploy" pour déployer l'architecture

CDKTF : TP 2 - PRISE EN MAIN

Ce que vous allez faire

1. Terminer le TP 1 si ce n'est pas fait
2. Refaire le TP 1 en utilisant le CDK TerraForm

THAT'S ALL FOLKS

