Terraform

- Créer conteneur lxc dans proxmox
- □□ Architecture Infrastructure as Code
- □□ Workflow Terraform : Gestion des Conteneurs LXC

Créer conteneur lxc dans proxmox

- Déployer un LXC
- Comprendre ce qui s’est passé

? 1. Installer Terraform

Sur ta Surface Pro (Windows probablement) :

□□ Télécharge Terraform depuis le site officiel (zip)

Ensuite :

- Dézippe
- Mets `terraform.exe` dans un dossier (ex: `C:\terraform`)
- Ajoute ce dossier au PATH

Puis dans un terminal :

```
terraform version
```

□□ Si tu vois une version → OK

? 2. Créer un accès API Proxmox

Dans Proxmox :

a. Créer un utilisateur Terraform

Datacenter → Permissions → Users

```
user : terraform@pve
```

b. Créer un API Token

Datacenter → Permissions → API Tokens

- Token ID : tf-token
- User : terraform@pve

□□ Note bien :

- Token ID
- Secret

c. Donner les droits

□□ Important sinon ça ne marchera pas

Permissions → Add

- Path : /
- User : terraform@pve
- Role : Administrator (pour commencer simple)

? 3. Créer ton projet Terraform

```
mkdir terraform-proxmox cd terraform-proxmox
```

Crée un fichier :

```
main.tf
```

?? 4. Configurer Terraform + Provider

Dans `main.tf` :

```
resource "proxmox_virtual_environment_container" "test_lxc" {
  node_name = "pve"

  vm_id = 101
  hostname = "test-lxc"

  initialization {
```

```
    hostname = "test-lxc"
  }

  operating_system {
    template_file_id = "local:vztmpl/debian-12-standard_12.0-1_amd64.tar.zst"
  }

  cpu {
    cores = 2
  }

  memory {
    dedicated = 512
  }

  network_interface {
    name = "eth0"
    bridge = "vmbr0"
  }

  rootfs {
    storage = "local-lvm"
    size    = "8G"
  }
}
```

□□ Remplace :

- IP
- token

? 5. Ajouter un LXC

⚠ Avant ça :

□□ Tu dois avoir un template LXC déjà téléchargé dans Proxmox (ex: Debian 12)

Ajoute dans `main.tf` :

```
resource "proxmox_virtual_environment_container" "test_lxc" { node_name = "pve" vm_id = 101 hostname = "test-lxc" initialization { hostname = "test-lxc" } operating_system { template_file_id = "local:vztmpl/debian-12-standard_12.0-1_amd64.tar.zst" } cpu { cores = 2 } memory { dedicated = 512 } network_interface { name = "eth0" bridge = "vmbr0" } rootfs { storage = "local-lvm" size = "8G" } }
```

? 6. Lancer Terraform

Dans le dossier :

Initialiser

```
terraform init
```

□□ Ça télécharge le provider Proxmox

Voir ce qu’il va faire

```
terraform plan
```

□□ Très important : ça te montre le résultat AVANT

Appliquer

```
terraform apply
```

□□ Tape `yes`

? Résultat

□□ Ton LXC est créé automatiquement dans Proxmox

Sans interface web. Juste du code.

? Ce que tu viens d’apprendre

- Terraform = décrit un état
- Provider Proxmox = parle à l’API
- `apply` = rend réel ce que tu as décrit
- `state` = garde la trace

?? Architecture Infrastructure as Code

Le document détaille la structure et le fonctionnement du dépôt de configuration Terraform utilisé pour piloter le cluster Proxmox. L'objectif est d'assurer une gestion 100% automatisée, versionnée et sécurisée.

1. Structure du Dépôt (Pattern Live/Modules)

Nous utilisons la séparation entre les **Modules** (plans de fabrication) et le **Live** (instances réelles).

```
terraform/
├── modules/                # --- L'USINE (Templates réutilisables) ---
│   ├── proxmox_lxc/        # Code générique pour créer un conteneur
│   ├── proxmox_vm/         # Code générique pour créer une machine virtuelle
│   └── synology_vm/        # Spécificités pour les VMs sur stockage Synology
└── live/                   # --- LA PRODUCTION (Ce qui tourne vraiment) ---
    ├── 01-core/            # Fondations : Pools, Réseaux, SDN
    ├── 02-vms/             # Instances de serveurs (Bastion, Plex, etc.)
    └── 03-k8s/             # Cluster Kubernetes spécifique
```

2. Pourquoi ce découpage ?

- **Modules :** Évite de répéter le code (DRY - Don't Repeat Yourself). Si on améliore la sécurité des LXC, on le fait dans le module et tous les serveurs en profitent.
- **Live (01, 02, 03) :** Réduit le "Blast Radius" (rayon d'explosion). Si une erreur est faite dans le dossier K8s, les fondations (Core) ne sont pas impactées car elles ont leur propre fichier d'état (State).

3. Sécurité : Authentification par Token

Ne jamais écrire de mots de passe dans les fichiers `.tf`. On utilise les variables d'environnement.

```
# Export du token API Proxmox (Indispensable avant chaque run)
# Format : 'USER@REALM!TOKENID=UUID'
export PROXMOX_VE_API_TOKEN='terraform@pam!awx-provisioning=c20efe74-ec94-4623-a53d-613ad538fcc9'
```

4. Commandes Vitales

Commande	Action
`terraform init`	Télécharge les connecteurs (Providers) Proxmox. À faire une seule fois par dossier.
`terraform plan`	**L'aperçu :** Compare le code avec la réalité. Ne modifie rien.
`terraform apply`	**L'exécution :** Applique les changements sur Proxmox. Demande "yes".

5. Comment modifier l'infrastructure ?

A. Ajouter ou renommer un Pool (Dossier Proxmox)

Modifier le fichier `live/01-core/main.tf` :

```
resource "proxmox_virtual_environment_pool" "mon_nouveau_pool" {
  pool_id = "NOM_DU_POOL"
  comment = "Description du contenu"
}
```

B. Mettre à jour des ressources

Il suffit de changer la valeur (ex: RAM ou CPU) dans le fichier `.tf` et de relancer un `terraform apply`. Terraform ne supprimera pas la machine, il modifiera ses paramètres à chaud si possible.

6. Résultats attendus (Proxmox)

Après l'exécution du **01-core**, les "Pools" apparaissent dans Proxmox. Pour les voir, passez la vue Proxmox de "*Server View*" à **"Folder View"**.

- **INFRASTRUCTURE** : Services vitaux (Réseau, DNS, Proxy)
- **KUBERNETES** : Cluster K8s
- **SERVICES** : Plex, Bookstack, etc.
- **TESTS** : Laboratoire temporaire

--- ### Pourquoi est-ce que c'est "Magique" ? Si demain ton Proxmox brûle ou si tu achètes un deuxième serveur : 1. Tu installes Proxmox. 2. Tu crées le token API. 3. Tu lances Terraform. 4. **En 10 secondes, toute ton arborescence, tes réseaux et tes pools sont recréés à l'identique.** ### Et maintenant, la suite ? Maintenant que nous avons les "pièces" (les Pools), nous allons pouvoir passer au dossier **`02-vms`**. C'est ici que nous allons créer nos premiers serveurs en utilisant les **Modules**.

? Workflow Terraform : Gestion des Conteneurs LXC

Ce guide explique comment transformer un conteneur Proxmox existant (ou nouveau) en un objet géré par le code (Infrastructure as Code).

1. Actions à faire UNE SEULE FOIS (Setup)

A. Préparation de l'environnement (Terminal)

Avant de lancer Terraform, il faut charger les accès de sécurité dans la session actuelle :

```
# Token API Proxmox
export PROXMOX_VE_API_TOKEN='terraform@pam!awx-provisioning=c20efe74-ec94-4623-a53d-613ad538fcc9'

# Variables pour les fichiers Terraform (Préfixe TF_VAR_)
export TF_VAR_lxc_root_password='Ghilghame$h8863!'
# Si lancement depuis code-server. A Adapter selon qui lance
export TF_VAR_admin_ssh_key='ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGa4tMA4dLSDNSL2x6YnqJ+L3g9UPGEeNP1bKywGvr1g code-server-homelab-2025-10-22'
```

B. Initialisation du dossier Live

Se placer dans `infra/terraform/live/02-lxcs/` et télécharger le connecteur Proxmox :

```
terraform init
```

2. Actions à RÉPÉTER (Pour chaque conteneur)

Pour chaque nouveau service (ex: Bitwarden, Bookstack), suivez ce cycle : **Coder -> Adopter -> Vérifier**.

Étape A : Créer le fichier de configuration

Créer un fichier par service (ex: `bitwarden.tf`) dans `live/02-lxcs/` :

```
module "bitwarden" {
  source           = "../../modules/proxmox_lxc"
  hostname         = "bitwarden"
  vmid             = 119
  target_node      = "proxmox"
  pool             = "SERVICES"

  cores            = 2
  memory           = 1024
  disk_size        = 8
  disk_datastore   = "ssd1tb"

  ip_address       = "10.151.151.231/24"
  gateway          = "10.151.151.1"

  password         = var.lxc_root_password
  ssh_public_keys  = var.admin_ssh_key
  template_file_id = "ssd1tb:vztmpl/debian-12-standard..."
}
```

Étape B : L'adoption (Uniquement pour l'existant)

Si le conteneur existe déjà sur Proxmox, il faut le "capturer" dans la mémoire de Terraform :

```
# Syntaxe : terraform import module.NOM_DU_MODULE.proxmox_virtual_environment_container.lxc
NOEUD/VMID
```

```
terraform import module.bitwarden.proxmox_virtual_environment_container.lxc proxmox/119
```

Étape C : La vérification

Comparer le code avec la réalité. Le résultat doit être "0 to add, 0 to change" :

```
terraform plan
```

3. Récapitulatif des fichiers types (.tf)

Fichier	Rôle	Contenu type
`providers.tf`	Connexion API	`provider "proxmox" { ... }`
`variables.tf`	Déclaration des entrées	`variable "lxc_root_password" { ... }`
`main.tf` (ou `service.tf`)	Le bon de commande	`module "nom" { source = "..." }`

⚠ **Rappel crucial :** La commande `import` ne se fait **qu'une seule fois** par machine. Une fois importée, toute modification doit se faire dans le fichier `.tf` suivi d'un `terraform apply`.