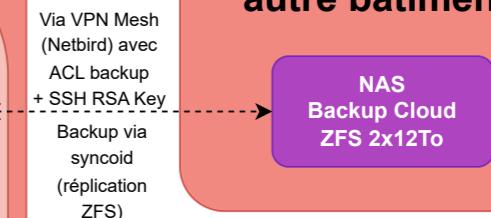


VLAN 10 : 192.168.10.0/24
 VLAN 20 : 192.168.20.0/24
 VLAN 50 : 192.168.50.0/24
 VLAN 100 : 192.168.100.0/24

Segmentation VLAN 100 HOMELAB :

- 176.16.100.1-20 : VMs FreeIPA + VIP, LXC Postgres, LXC NFSv4 Share, autres dans le futur...
- 176.16.100.100-199 : Cluster K3S

Réseau distant, autre bâtiment



Réflexions autour de PostgreSQL :
 Voulant atteindre la haute disponibilité pour l'extrême majorité des services (ADDS, VPN Mesh, reverse proxy...) pour avoir le moins de point de défaillances, la BDD est un défi.

Ici, le choix s'est finalement porté sur **un seul container LXC** (donc pas de HA), avec un script qui relance le container en cas d'erreur. Plusieurs solutions ont été envisagées, mais dans le cadre d'une infrastructure homelab avec des ressources limitées (une seule machine) pour des raisons budgétaires et d'économie électrique, celles-ci ont été écartées.

Les solutions envisagées :

- **CockroachDB** : une BDD designée pour la scalabilité et compatible Postgres, seulement, certaines fonctions de Postgres ne sont pas compatibles et des applications comme Keycloak (OIDC provider) en ont besoin.
- **YugabyteDB** : Plus compatible que CockroachDB avec la majorité des apps demandant une BDD Postgres, celle-ci est néanmoins plus lourde, voire trop lourde pour notre infrastructure ;

- **PostgreSQL avec Patroni** : Cette solution apparaît comme la plus robuste des trois, car elle repose sur des instances PostgreSQL standards, totalement compatibles avec l'ensemble des applications déployées. Chaque instance Postgres est supervisée et contrôlée par Patroni, un leader entre les BDD est élu par consensus via etcd/Consul (on ne peut avoir qu'une seule BDD maître, où l'on écrit). La seconde BDD est bloquée en standby par son Patroni et réplique les données de la première instance, les nouvelles élections se font en cas de panne (donc on a un failover automatique, on atteint notre objectif). Bien que cela semble être la solution idéale, deux instances de BDD sont lancées en même temps, augmentant la charge sur le serveur. **Cette solution reste envisagée à l'avenir.**