AKAT

Rapport SAÉ 5.02

Piloter un projet informatique

Arnaud Kastner :	Chef de projet
Thomas Mirbey :	Architecte solution
Kyllian Cuevas :	Responsable Administration Réseaux et Sécurité
Alexandre Berot-Armand :	Responsable Administration Système et Automatisation

Sommaire

1. Gestion de projet
    a. Organisation
    b. Travail collaboratif
2. Cahier des charges
3. Plan d'adressage IP
4. Architecture
    a. Coeur du réseau
    b. Site 0
    c. Site 1
5. Services mis en place
    a. Base de données
    b. DNS
    c. Active Directory
    d. NAS
    e. DHCP
    f. Sauvegarde
    g. Zabbix
    h. Radius
    i. Broker MQTT
    j. Lecteurs de badge pour contrôle d'accès
    k. Gestion de parc
    l. Serveur Web
6. Scripts *
    a. Gestion des données
    b. Tests de redondance
7. Sécurité
    a. Vlan
    b. Access-lists
    c. Pare-feux
    d. Radius
    e. SSL/TLS et Certificats
    f. VPN
8. Redondance et continuité de service
    a. VRRP/HSRP
    b. SLB : Server load Balancing
    c. Spanning-tree
    d. Sauvegarde 321
9. Améliorations
10. Devis
11. Conclusion

1. Gestion de projet

1.a Organisation

Pour l'organisation du projet nous avons créé un Trello que nous avons mis à jour pendant l'ensemble du projet.

Trello est un outil de gestion de projet qui est particulièrement utile dans la réalisation de projets agiles. Il permet de créer des tableaux de bord virtuels où vous pouvez organiser et suivre les différentes tâches et étapes du projet. Chaque tâche est représentée par une carte que vous pouvez déplacer d'une liste à une autre pour indiquer son statut (à faire, en cours, terminée, etc.). Cela facilite la visualisation de l'avancement du projet et assure une meilleure coordination entre les membres de l'équipe.

Trello offre également des fonctionnalités telles que la création de listes de vérification, l'assignation de responsabilités aux membres de l'équipe, l'ajout d'échéances et la collaboration en temps réel. Il permet ainsi de planifier, d'organiser et de suivre efficacement les projets agiles.

En résumé, Trello est un outil précieux pour la gestion et le suivi des projets agiles, en offrant une visualisation claire des tâches, une coordination améliorée et une collaboration en temps réel.

Nous avons séparé le trello en 3 parties : "À faire", " En cours" et "Terminé".

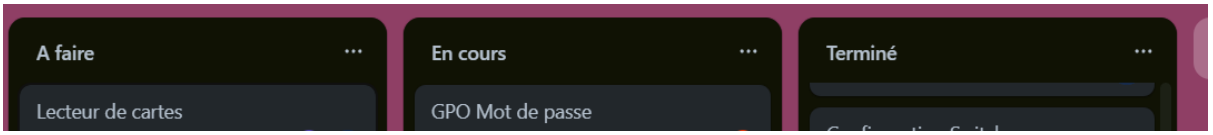


Figure 1: Organisation du trello en 3 parties

Nous avons ensuite créé les différentes tâches qui composent notre projet comme par exemple : "Serveur DHCP", "Serveur ADDS", "Coeur de Réseau", "Paramétrer les lecteurs de cartes","Faire le devis", etc

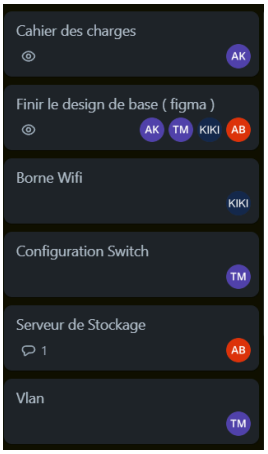


Figure 2: Tâches à réaliser dans Trello

Nous avons ensuite accordé une priorité plus ou moins élevée à chaque carte ce qui nous a permis de toujours garder en vue les éléments les plus importants

Figure 3: Tâches importantes du Trello sur le site web

Une fois cela fait, nous nous sommes répartis les tâches en fonction des capacités de chacun.

A chaque fin de demi-journée, nous faisions le point sur le trello pour voir les évolutions du projet et nous focaliser sur les choses essentielles en mettant à jour les priorités de l'ensemble des tâches.

1.b Travail collaboratif

- Discord

Comme principal moyen de communication nous avons choisi d'utiliser le logiciel gratuit Discord.

Nous avons ensuite créé un serveur sur lequel nous avons invité uniquement les membres de notre équipe.

Un serveur Discord offre de nombreux avantages. Il permet une communication en temps réel entre les participants, facilitant ainsi la coordination et l'organisation. De plus, il offre des fonctionnalités telles que la possibilité de créer des canaux textuels et vocaux dédiés à différents sujets, ce qui permet une discussion plus structurée. Un serveur Discord peut également être utilisé pour partager des ressources, des liens et des fichiers pertinents pour la partie. En résumé, la création d'un serveur Discord est un moyen efficace de centraliser la communication et de faciliter la collaboration pendant la partie.

Nous avons décidé de structurer le nôtre en 7 parties, 5 écrites et 2 vocales.

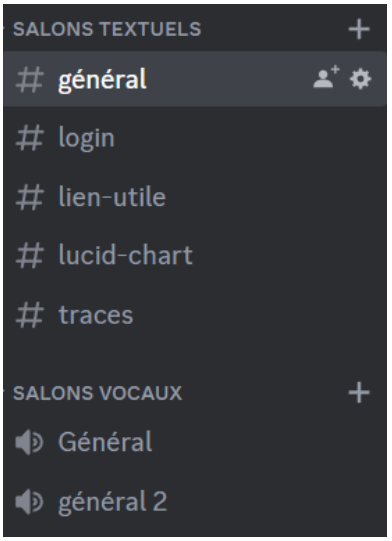


Figure 4: Channels Discord

Les 5 parties écrites ont été créés de la façon suivantes :

- Un channel général : Il a servi à planifier nos différentes réunions à distance ainsi que de partager différentes informations sur le cahier des charges, il était aussi l'endroit où on posait l'ensemble de nos questions

- Un channel "Login" : On y stockait les logins des différentes machines / VM que nous avons utilisées

- Un channel "Lien Utiles" : On y inscrivait l'ensemble des liens pouvant nous êtres utiles comme des tutoriels d'installations et de configuration d'équipement, le cahier des charges que les clients nous ont transmit etc

- Un channel "Lucid Chart" : Dans lequel on retrouve notre plan réseau, on a créé un channel dédié au plan réseau car c'est le document le plus important du projet, on y revient toujours peu importe ce qu'on est en train de faire

- Un channel "Traces" : dans lequel nous avons stocké l'ensemble des traces qui nous serviront pour la SAé Portfolio et qui seront par la suite misent sur notre site internet de présentation.

Nous avons aussi créé 2 channels vocaux : "Général" et "Général 2" pour organiser des réunions et travailler ensemble à distance. Le channel "Général 2" était extrêmement utile quand nous devions travailler en binôme sur un des aspects du projet, un binôme se trouvait dans chacun des salons vocaux ce qui permettait de ne pas se gêner mutuellement.

- Figma

Pour réaliser le schéma de notre réseau ainsi que le template de nos bases de données et de notre site internet nous avons utilisé l'outil Figma.

Figma est un logiciel de conception graphique et d'interface utilisateur (UI) en ligne. Il permet aux designers de créer des maquettes, des prototypes interactifs et de collaborer en temps réel avec leur équipe. Figma est particulièrement apprécié par ses utilisateurs pour sa facilité d'utilisation, sa capacité à travailler sur des projets en mode collaboratif, son accessibilité depuis n'importe quel appareil connecté à Internet, et son outil de prototypage avancé.

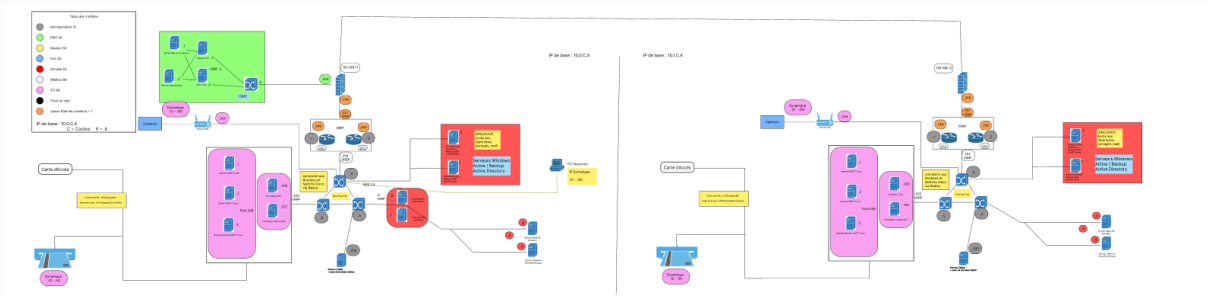


Figure 5: Schéma du réseau complet des deux Hôpitaux

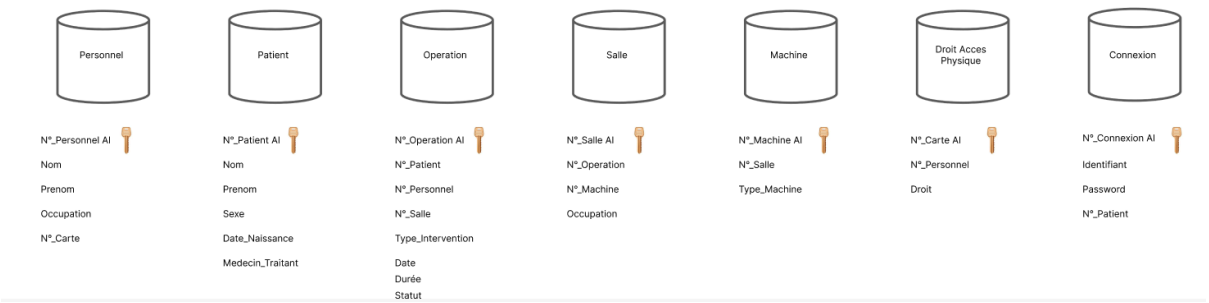


Figure 6: Modèle Conceptuel de données de la Base de Données

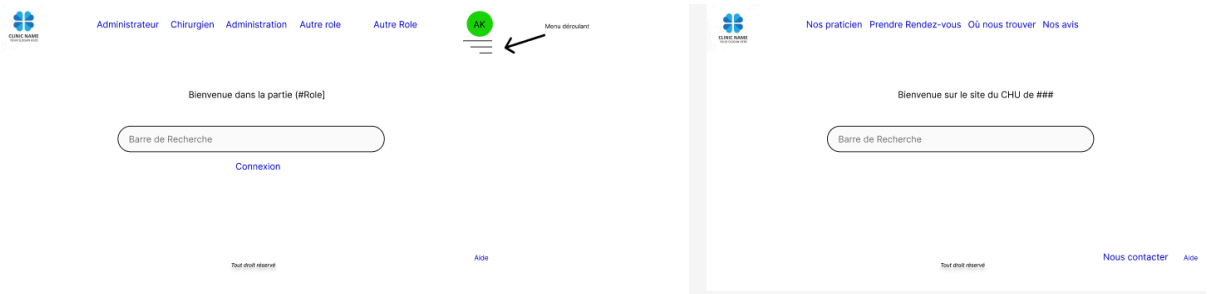


Figure 7: Modèle pour le site web sur Figma

- Github

Enfin nous avons décidé d'utiliser le site en ligne github pour tout ce qui est dépôt de fichier et versionnement.

Nous avons créé un projet sur github spécialement pour ce projet ce qui nous a permis que chacun puisse éditer les différents fichiers sur lesquels il a travaillé.

GitHub offre aussi des outils de suivi des problèmes, des fonctionnalités de demande de fusion, et un système de contrôle de version Git, favorisant un développement plus efficace et transparent.

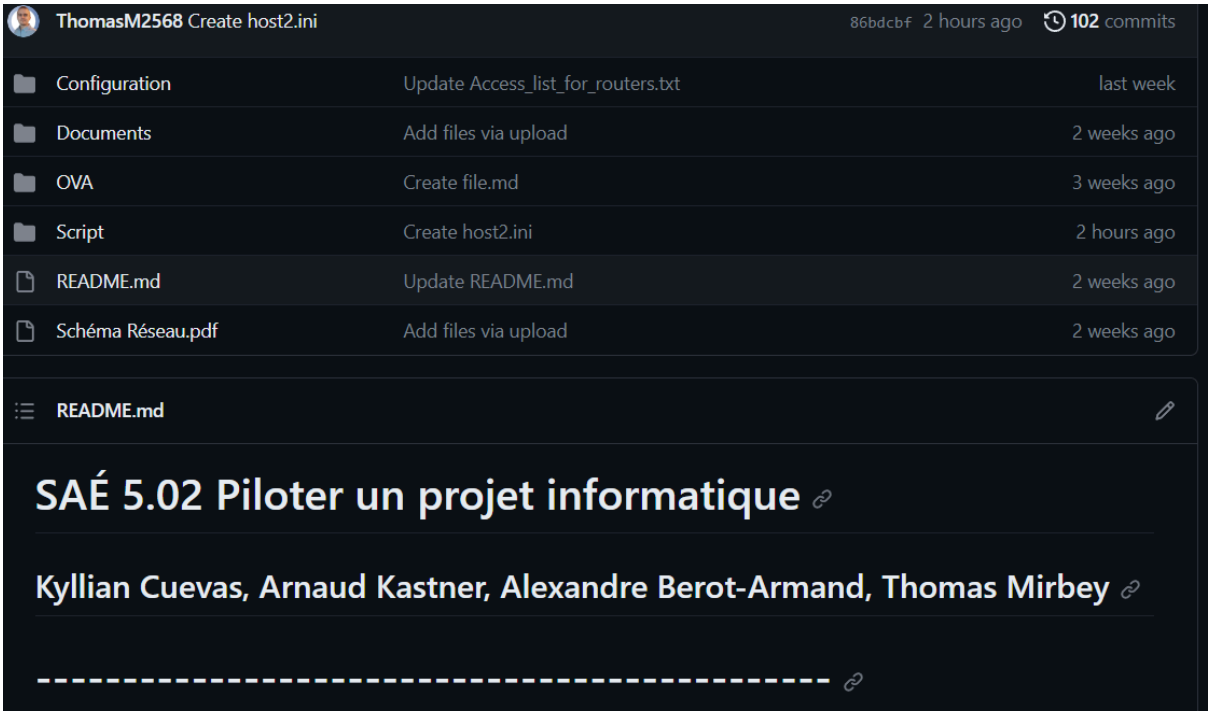


Figure 8: Github de la Saé

2. Cahier des Charges

Une des premières choses qui a été réalisée en parallèle du plan réseaux était le cahier des charges. Pour le réaliser, nous avons commencé par analyser en détail les consignes qui nous ont été données puis nous avons réfléchi aux composantes essentielles du projet.

a. Introduction

L'objectif de ce cahier des charges est de définir les exigences et les tâches pour la sécurisation du système informatique d'un hôpital. Compte tenu des récentes attaques cyber et de la sensibilité des données médicales, il est essentiel de mettre en place une infrastructure informatique robuste et sécurisée. Le projet consistera à créer une architecture réseau cloisonnée, à sécuriser l'accès aux données médicales, à établir une connexion VPN avec un autre hôpital, et à documenter l'ensemble du système.

b. Architecture réseau sécurisée

Le réseau de l'hôpital sera structuré en trois espaces séparés : la direction, l'administration et la gestion des données médicales. Chacun de ces espaces devra être isolé des autres. Le standard 802.1X sera utilisé pour sécuriser l'accès au réseau, avec une authentification centralisée via Radius. Les droits d'accès seront définis comme suit :

- Le personnel de la direction pourra accéder aux données de la direction et de l'administration.
- Le personnel de l'administration pourra accéder aux données de l'administration.
- Le personnel médical pourra accéder aux données médicales.

c. Sécurisation de l'accès Internet et VPN

L'accès à Internet sera bloqué dans les deux sens, à l'exception de la connexion VPN établie avec un autre hôpital. Cette connexion permettra uniquement la consultation des données médicales, sans accès à Internet. Les tâches comprennent :

- Mise en place d'un VPN avec l'autre hôpital.
- Blocage complet de l'accès à Internet.
- Configuration pour permettre uniquement la consultation des données médicales via le VPN.

d. Sécurité physique

L'accès à la salle d'opération sera restreint aux personnels médicaux pendant les opérations, tandis que le personnel d'entretien y aura accès en dehors des opérations médicales.

e. Gestion des données

La création de deux bases de données distinctes sera nécessaire pour stocker les informations sur le personnel et les patients. Un serveur sera dédié à la direction, et les données des serveurs seront sauvegardées régulièrement pour permettre une restauration efficace en cas de perte de données. Des procédures de réinstallation automatisée des serveurs seront mises en place pour réactiver un serveur dont le système est effacé ou inutilisable. De plus, une procédure d'échange des rôles entre serveur principal et serveur de secours sera établie pour garantir la continuité des services en cas de défaillance.

f. Documentation

Toutes les étapes du projet, y compris la configuration du réseau, l'installation des serveurs, les procédures de secours, les politiques d'accès et de sécurité, devront être minutieusement documentées. Cette documentation sera essentielle pour permettre au personnel du service informatique/réseau d'accomplir les tâches. Il est recommandé d'utiliser GitHub pour stocker la documentation et les configurations du réseau.

g. Autres tâches

En plus des tâches mentionnées ci-dessus, il sera nécessaire de procéder à l'installation des services suivants :

- Active Directory
- Serveur DHCP
- Serveur DNS
- Serveur de Stockage
- Base de données
- Serveur Radius
- Script d'automatisation
- Contrôleur SLB (Server Load Balancer)
- Dispositifs de caméra
- Lecteurs et graveurs de cartes

g. Documentation pour les utilisateurs, administrateurs et programmeurs

Une documentation complète devra être créée pour les utilisateurs, les administrateurs et les programmeurs, couvrant tous les aspects du système informatique, de la sécurité et des procédures d'entretien.

Ce cahier des charges servira de référence pour la planification, la mise en œuvre et la documentation du projet visant à sécuriser le système informatique de l'hôpital. Il est essentiel de respecter toutes les réglementations de sécurité et de protéger les données médicales sensibles.

3. Plan d’adressage IP

Afin d’améliorer la sécurité et faciliter l’administration du réseau nous avons créé plusieurs VLANs. Les VLANs (Virtual Local Area Network) sont des groupes permettant ainsi de segmenter et de cloisonner un réseau physique en plusieurs réseaux virtuels. Ainsi nous avons créé ici 8 VLANs :

- VLAN 1 : Liaisons Pare-feu/Routeurs
- VLAN 10 : Administration du réseau.
- VLAN 20 : VLAN DMZ (Une DMZ (zone démilitarisée) est une zone réseau intermédiaire entre un réseau interne sécurisé et un réseau externe non sécurisé, utilisée pour héberger des services accessibles depuis Internet tout en isolant le réseau interne des menaces potentielles.). Ici la DMZ correspond au serveur Web Apache.
- VLAN 30 : Plage d’adresse IP du personnel de gestion du site (Secrétaire, Concierge, etc.).
- VLAN 40 : VLAN voix dédié à la téléphonie sur IP.
- VLAN 50 : VLAN données. Ce VLAN est dédié à la base de données et au serveur DNS, DHCP et Active Directory dont nous détaillerons les fonctions plus tard dans ce rapport.
- VLAN 60 : VLAN Médical. Plage d’adresse IP dédiée au personnel médical (médecins, infirmiers/ères, etc.)
- VLAN 90 : VLAN réservé aux bornes WIFI et aux différents capteurs et lecteurs de badges.

Nous avons aussi configuré un trunk interVLAN. Un trunk interVLAN est une liaison réseau qui permet le passage de données entre différents VLANs très pratique pour l’administration du réseau.

Concernant le plan d’adressage du réseau nous avons suivi un plan logique de type : 10.0.V.X

V correspondant au VLAN et X à l’adresse de l’appareil dans ce VLAN.

A noter qu’avec ce plan d’adressage, chaque VLAN peut contenir jusqu’à 254 appareils ce qui laisse une grande marge d’extension du réseau déjà en place. De plus, si un manque d’adresse se fait ressentir, il suffira de créer un nouveau VLAN et de modifier les règles de routage.

4. Architecture

Cette partie va détailler l'architecture réseau mise en place durant cette Saé.

Les réseaux mis en place pour chacun des sites de l'Hôpital (site 0 et 1) se basent sur la même architecture à quelques différences près, celles-ci seront détaillées dans la suite de ce rapport.

Nous avons utilisés différents matériels :

- 4 routeurs Cisco 1900 Series
- 6 commutateurs Cisco Catalyst 3750 series PoE-24
- 2 commutateurs Cisco Catalyst 3750 v2 series
- 1 pare-feu Stormshield SN210W
- 1 pare-feu IPfire Linux
- 2 bornes Wifi WRT54GL
- Des ordinateurs Windows et Debian pour nos serveurs

Nous avons configuré le protocole SSH sur chaque équipement pour pouvoir y accéder à distance et les configurer plus facilement.

4.a. Coeur du réseau

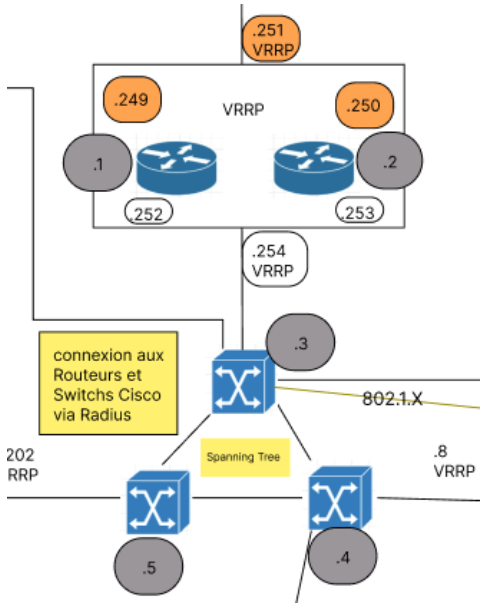


Figure 9: Coeur du réseau d'un Hôpital

Le cœur du réseau de chacun des sites se repose sur la même architecture. Chaque réseau est composé de 2 routeurs et de 3 commutateurs.

Pour permettre aux clients du réseau de sortir de leur VLAN pour accéder au réseau nous avions besoin de mettre en place des passerelles par défaut qui gèrent le routage des paquets sur le réseau mais également le routage inter VLAN.

Pour permettre le lien entre les routeurs et le VLAN, nous avons configuré les interfaces en encapsulation 802.1q. Le but est de créer des sous interface virtuelle pour chaque VLAN sur une interface physique. Ces interfaces serviront de passerelles pour leur VLAN respectif.

Nous avons donc configuré les interfaces côté réseaux internes de nos routeurs avec cette encapsulation.

Pour permettre à nos clients du réseau de récupérer une configuration réseau de manière dynamique via notre serveur Active Directory, nous avons configuré un agent de relais DHCP permettant aux requêtes DHCP de nos clients d'accéder au serveur sur un VLAN différent.

Pour permettre une continuité de service en cas de panne ou de maintenance d'un des deux routeurs, ceux-ci se partagent une adresse IP virtuelle grâce au protocole HSRP.

L'intérêt de la passerelle est que les clients du réseau contactent la même adresse IP indépendamment de quel routeur traite les paquets. En cas de panne, l'adresse IP partagée bascule automatiquement ce qui évite les coupures au niveau du réseau.

```
interface GigabitEthernet0/0
 description Vers SWAKAT01
 no ip address
 duplex auto
 speed auto
!
interface GigabitEthernet0/0.10
 encapsulation dot1Q 10
 ip address 10.0.10.1 255.255.255.0
 ip helper-address 10.0.50.1
 ip helper-address 10.0.50.2
 standby 10 ip 10.0.10.254
 standby 10 preempt
!
interface GigabitEthernet0/0.20
 encapsulation dot1Q 20
 ip address 10.0.20.252 255.255.255.0
 ip helper-address 10.0.50.1
 ip helper-address 10.0.50.2
 standby 20 ip 10.0.20.254
 standby 20 preempt
!
interface GigabitEthernet0/0.30
 encapsulation dot1Q 30
 ip address 10.0.30.252 255.255.255.0
 ip helper-address 10.0.50.1
 ip helper-address 10.0.50.2
 standby 30 ip 10.0.30.254
 standby 30 preempt
!
```

Figure 10: Configuration des interfaces des routeurs

Pour gérer les séparations du réseau, nous avons configuré des VLAN sur nos commutateurs. Ceux-ci étaient connectés entre eux en formant une boucle pour permettre de la redondance, cette partie sera davantage détaillée dans la partie 8.c. Spanning-tree.

```
interface FastEthernet1/0/3
 description Vers RAKAT02
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1,10,20,30,40,50,60,90
 switchport mode trunk
```

Figure 11: Configuration d'un trunk

Chaque port était associé à un VLAN.

Les liens vers nos routeurs quant à eux étaient configurés en trunk pour permettre de regrouper les VLAN sur un même lien pour ne pas avoir à créer un lien spécifique pour chacun.

Pour permettre le routage des informations sur le réseau, nous avons choisi de mettre en place le protocole OSPF (Open Shortest Path First). Nous avons déclaré les différents réseaux à router dans la configuration de chacun de nos routeurs et nous avons validé le bon fonctionnement de cette configuration en faisant des ping et des traceroutes vers des équipements d'autres sous réseaux (VLANs différents).

Nous avons également redistribué les routes directes qui correspondent aux périphériques directement connectés à notre équipement mais également les routes statiques que nous avons renseignées.

La route par défaut de chaque routeur est le pare-feu qui gère le lien entre le réseau local, la dmz et les réseaux distants (Internet et réseaux de l'autre hôpital).

Le choix d'utiliser OSPF plutôt qu'un autre protocole de routage comme BGP s'est fait sur plusieurs critères.

```
router ospf 1
 redistribute connected
 redistribute static
 network 10.0.1.0 0.0.0.255 area 0
 network 10.0.10.0 0.0.0.255 area 0
 network 10.0.20.0 0.0.0.255 area 0
 network 10.0.30.0 0.0.0.255 area 0
 network 10.0.40.0 0.0.0.255 area 0
 network 10.0.50.0 0.0.0.255 area 0
 network 10.0.60.0 0.0.0.255 area 0
 network 10.0.90.0 0.0.0.255 area 0
```

Figure 12: Configuration du protocole OSPF

OSPF est un protocole facile à mettre en place et qui ne nécessite pas une grosse configuration. Dans un projet en temps restreint comme cette Saé, le choix de la facilité de configuration prend tout son sens.

De plus, OSPF ne nécessite pas de maillage entre tous nos équipements et converge vite ce qui permet de facilement pouvoir tester son routage sur une petite infrastructure avant de la complexifier.

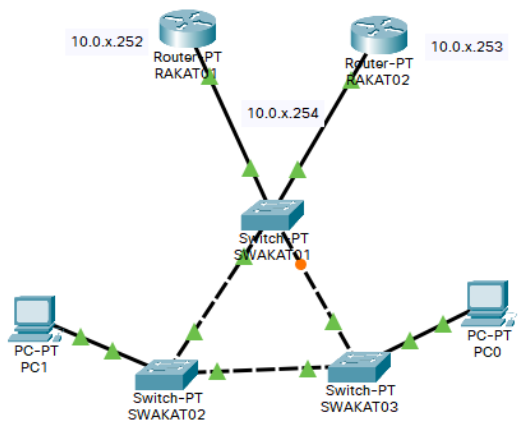


Figure 13: Infrastructure de test sur Packet Tracer

L'entièreté de la configuration du cœur du réseau a été réalisée sur Packet Tracer avant d'être mise en place physiquement.

Chaque fonctionnalité configurée (OSPF, Spanning-tree, VLAN, Trunk) a été testée et validée sur l'infrastructure de test Packet Tracer et nous a permis de valider la cohérence de notre infrastructure.

Mettre en place notre infrastructure de manière virtuelle avant de la réaliser physiquement a été bénéfique sur plusieurs points.

Tout d'abord cela nous a permis de vérifier que tout ce qui était configuré fonctionnait comme souhaité.

Cela nous a également permis de pouvoir reprendre notre configuration sur Packet Tracer et de la mettre sur les équipements réseaux de manière assez facile tout en évitant les erreurs de configuration ou de recopiage.

Finalement, nous avons pu tester des modifications ou des ajouts sur l'infrastructure virtuellement et valider le bon fonctionnement de ceux-ci avant de passer cette configuration sur nos équipements. Cela nous permettait donc de dédouaner le réseau en cas d'incident technique car celui-ci avait été validé et était fonctionnel en test. Nous avons donc pu plus facilement identifier les problématiques et les corriger ce qui représentait un réel gain de temps.

4.b. Site 0

Comme expliqué plus tôt, chaque site avait la même architecture. Néanmoins la seule différence entre le site 0 et les autres résidait dans les serveurs. La majorité de ceux-ci se trouvaient sur le site principal (site 0) et les serveurs actifs se basant sur le fonctionnement actif backup se trouvaient sur le site 0.

L'ensemble des services mis en place seront détaillés dans la partie 5 : Services mis en place.

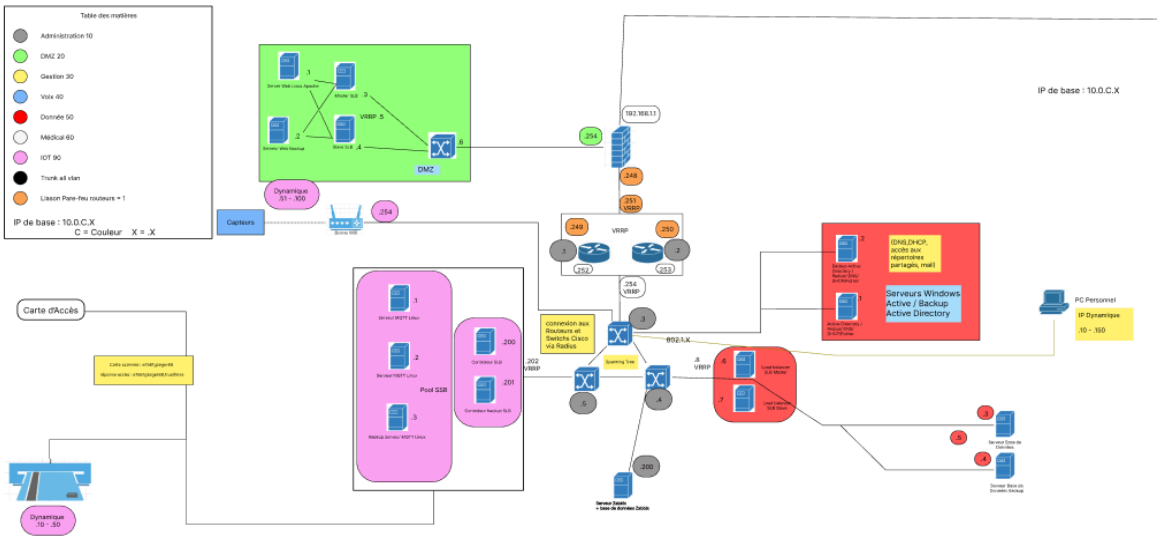


Figure 14: Schéma du site 0

4.c. Site 1

Le site 1 est donc similaire au site 0 mais abrite moins de serveurs car ceux-ci sont majoritairement hébergés sur le site principal (site 0). Les serveurs de ce site étaient majoritairement des serveurs de secours ou des serveurs secondaires ce qui permettait d'avoir une redondance plus importante. Notre site1 pouvait également être autonome.

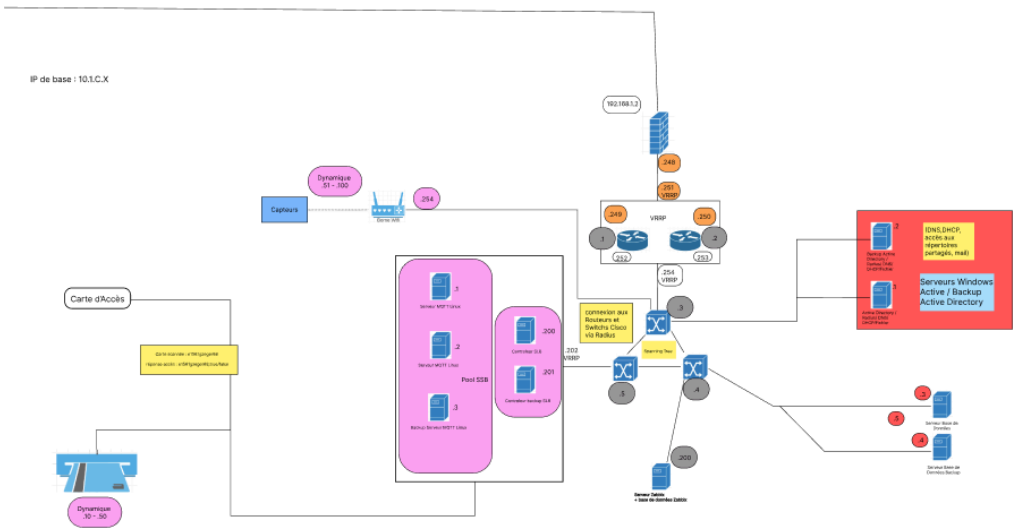


Figure 15: Schéma du site 1

5. Services mis en place

5. a. Base de données

Nous avons commencé par réfléchir aux rôles les plus importants ce qui nous a donné les tables suivantes

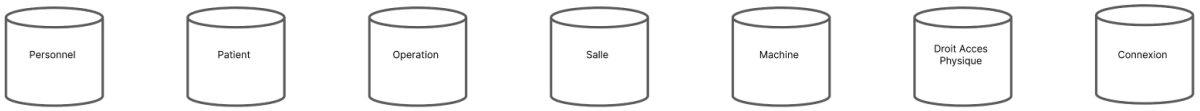


Figure 16: Schéma des tables de la base de données

Nous avons ensuite réfléchi aux éléments que chaque table devait comporter. Avec tout d'abord un identifiant unique pour chacune d'entre elles puis on les a répliqués dans les autres tables en fonction des liens logiques.

Par exemple, la table opération possède son propre ID ainsi que celui du médecin qui va opérer dedans et celui du patient qui va se faire opérer.

En répétant cette logique plusieurs fois et en ajoutant d’autres éléments essentiels comme l'âge et le nom, prénom des patients nous sommes arrivés au résultat suivant :

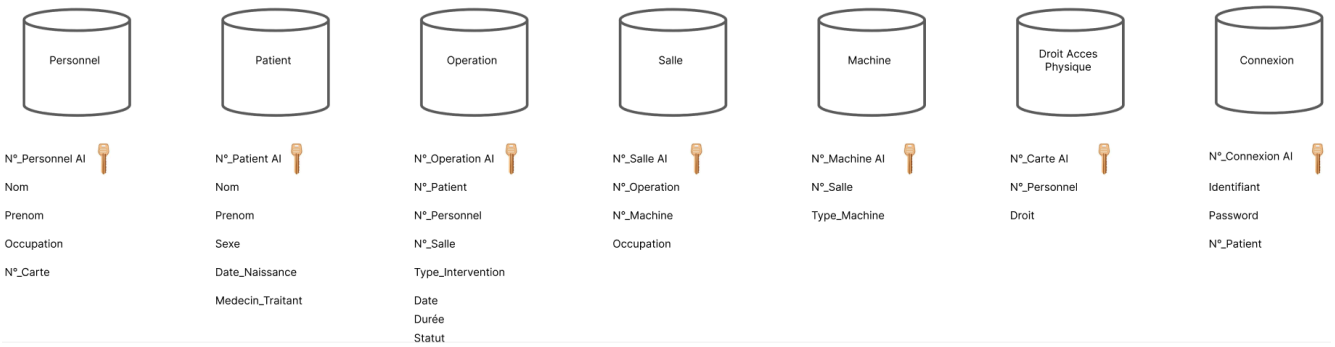


Figure 17: Modèle Conceptuel de Bases de données

5. b. DNS

Un serveur DNS (pour Domaine Name Service) ou serveur de noms de domaine est un service qui sert de répertoire de correspondances entre les noms de domaines conviviaux (comme www.google.fr) et les adresses IP numériques (comme 8.8.8.8).

Ils jouent un rôle clé dans la résolution des noms de domaine et facilitent la communication au sein des réseaux, en permettant aux appareils de se localiser mutuellement en utilisant des noms conviviaux au lieu d'adresses IP.

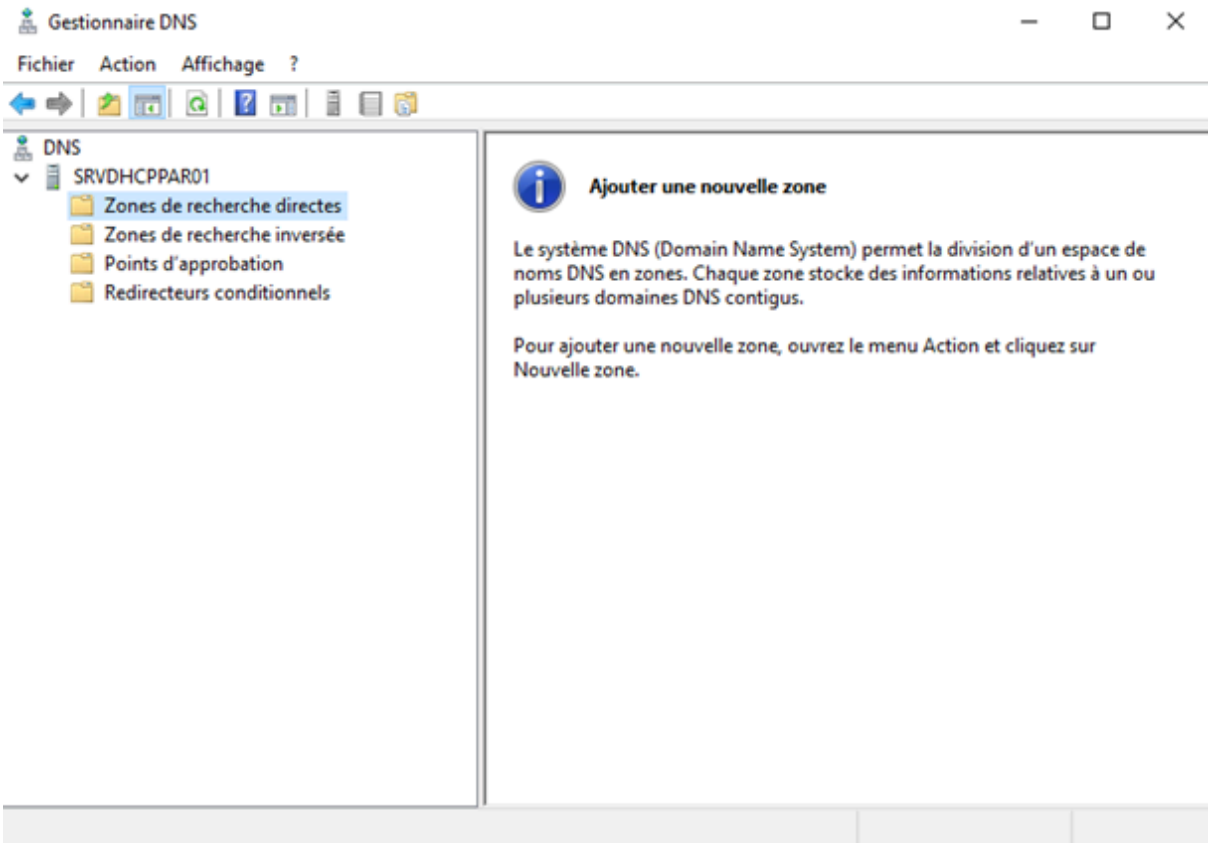


Figure 18: Page de configuration du DNS Windows Server

Le module DNS s'installe facilement sur Windows Server et est obligatoire pour installer la fonctionnalité Active Directory.

Dans notre cas, notre serveur DNS principal est celui de la Clinique 0 qui servira de guide. Le serveur DNS de la Clinique 1 sera un serveur de réplique qui copiera les informations depuis le serveur principal

Avec l'ajout de la fonctionnalité DNS, nous ajoutons le domaine AKAT.FR qui représente la forêt informatique des cliniques

5. c. Active Directory

Un serveur Active Directory est un service de gestion d'annuaire qui stocke et organise des informations sur les utilisateurs, les groupes, les ordinateurs et d'autres ressources au sein d'un réseau. Il permet de centraliser la gestion des utilisateurs, des autorisations et de simplifier l'administration des systèmes, assurant ainsi l'authentification, l'autorisation et la gestion des politiques de sécurité au sein d'un réseau d'entreprise.

Dans le but d'accélérer la mise en place de l'ensemble des serveurs, c'est un script en PowerShell qui s'occupe de mettre en place l'entièreté du serveur Active Directory (AD):
[Lien vers le GitHub qui héberge le script AD]

Figure 19: Script de configuration d'Active Directory sur Github

Dans un souci d'explication, l'image ne montre que la partie Active Directory du Script qui intègre aussi la partie serveur NAS mais celle-ci sera détaillée dans la suite de ce rapport.

Pour la partie AD, le script commence par importer le fichier CSV tiré du serveur de base de données.
Ensuite, nous créons une Unité Organisationnelle « Clinique » dans le domaine AKAT.FR pour organiser les objets.

Puis nous créons le groupe « EQUIPES » pour gérer les différentes équipes composants les cliniques.
Après cela, nous avons une boucle de traitement qui crée un groupe par équipe dans le CSV et qui l'ajoute au groupe « EQUIPES » existant.
Pour finir, il y a une seconde boucle qui s'opère pour créer chaque utilisateur présent sur le CSV dans l'OU « CLINIQUE » et qui lui attribue l'équipe dont il fait partie.

Le serveur principal de la clinique 0 servira de serveur Active Directory principal. Ce sera donc le serveur sur lequel le script sera lancé puis le serveur de la clinique 1 servira de

réplique AD au cas où le serveur principal viendrait à tomber en activant manuellement l'option.

5.d. NAS

Un serveur NAS (Network Attached Storage) est un appareil de stockage de données connecté à un réseau. Il offre un moyen centralisé et pratique de sauvegarder des données, de partager des fichiers et d'accéder à des contenus multimédias en toute simplicité.

La configuration du serveur NAS se fait dans le script AD dont le lien se trouve à la sous-rubrique précédente.

```powershell
# Spécifiez le chemin du fichier CSV contenant les données
$CSVFile = "C:\BDD-Script\add_personnel.csv"
$CSVData = Import-CSV -Path $CSVFile -Delimiter "," -Encoding UTF8

# Créez les répertoires "PERSO" et "EQUIPES" sur le lecteur C :
New-Item -Path "C:\" -Name "PERSO" -ItemType "Directory" -Force
New-Item -Path "C:\" -Name "EQUIPES" -ItemType "Directory" -Force

# Boucle de traitement des données du CSV
foreach ($add_personnel in $CSVData) {
    # Récupérez le nom d'équipe à partir du CSV
    $EquipesPersonnel = $add_personnel.occupation

    # Créez un répertoire pour l'équipe sous C:\EQUIPES
    New-Item -Path "C:\EQUIPES\" -Name "$EquipesPersonnel" -ItemType "Directory" -Force

    # Configurez les autorisations du répertoire
    $acl = Get-ACL -Path "C:\EQUIPES\$EquipesPersonnel"
    $acl.SetAccessRuleProtection($true, $false)
    $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("Administrateur", "FullControl", "Allow")
    $acl.AddAccessRule($AccessRule)
    $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("EQUIPES", "ReadAndExecute", "Allow")
    $acl.AddAccessRule($AccessRule)
    $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("$EquipesPersonnel", "Modify", "Allow")
    $acl.AddAccessRule($AccessRule)
    $acl | Set-Acl -Path "C:\EQUIPES\$EquipesPersonnel"
}

# Créez des partages SMB pour "EQUIPES" et "PERSO" avec un accès complet pour tout le monde
New-SmbShare -Name "EQUIPES" -Path "C:\EQUIPES\" -FullAccess "Everyone"
New-SmbShare -Name "PERSO" -Path "C:\PERSO\" -FullAccess "Everyone"

```

Figure 20: Script de configuration du NAS

Toujours dans un souci d'information, l'image suivante ne montre que la partie NAS du script Active Directory.

La première partie est la même que pour le script Active Directory avec l'importation du CSV tiré du serveur de base de données.

Par la suite, il crée les dossiers « PERSO » et « EQUIPES » sur le lecteur C.
Puis il crée un répertoire par équipe dans le dossier « EQUIPES » puis un répertoire par personne dans le dossier « PERSO ».

Enfin, le script attribue les droits sur les différents dossiers aux personnes autorisées.

Ce script étant compris dans le script Active Directory, il ne sera déployé que sur le serveur principal.

5.e. DHCP

Un serveur DHCP (Dynamic Host Configuration Protocol) est un composant réseau qui attribue automatiquement des adresses IP et d'autres paramètres de configuration réseau, tels que la passerelle et les serveurs DNS, aux dispositifs connectés à un réseau, comme les ordinateurs, les smartphones et les imprimantes.

Il simplifie la gestion en évitant la nécessité de configurer manuellement chaque appareil.

Pour augmenter la rapidité de déploiement ainsi que la rapidité de modification des plages DHCP, c'est aussi un script PowerShell qui s'occupera de générer le serveur DHCP.

[Lien vers le GitHub qui héberge le script DHCP]

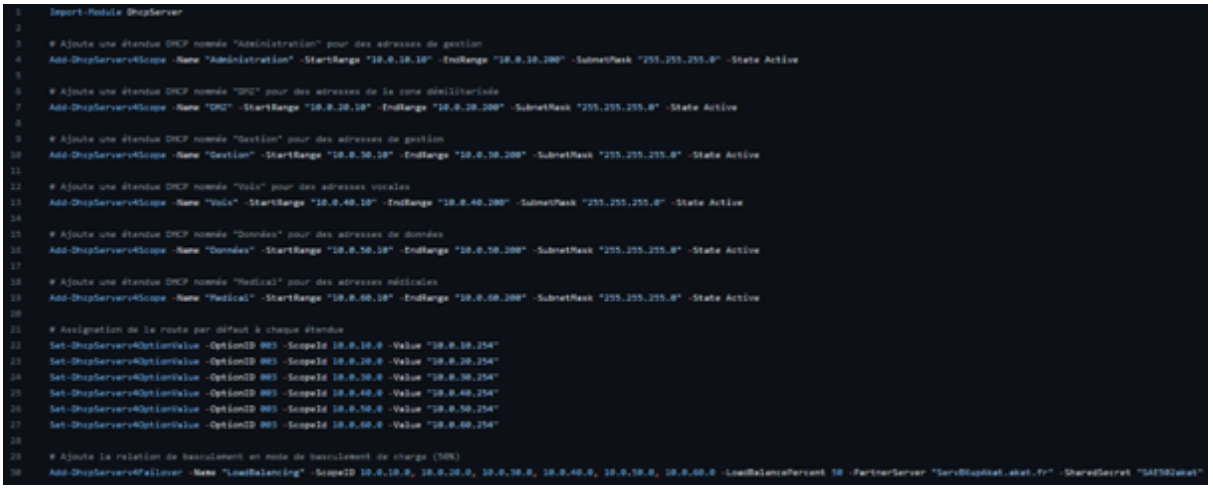


Figure 21: Script de configuration du DHCP

Le script commence par importer le module DHCPServer pour gérer les services DHCP. Ensuite, il ajoute les différentes étendues DHCP avec les noms et les plages d'adresses IP prédéfinies.

Il configure ensuite la route par défaut pour attribuer la bonne passerelle à chaque étendue. Pour finir, le script active une relation de basculement entre le serveur principal de Clinique 0 et celui de Clinique 1 avec un basculement de charge de 50%.

La dernière partie permet d'activer automatiquement le service DHCP sur les deux serveurs à l'aide d'un seul script et de créer une relation de basculement. Cela ajoute aussi un effet naturel de FailOver ce qui permet, en cas de panne d'un des deux serveurs, qu'un seul serveur peut s'occuper seul de l'étendu DHCP.

5.f. Sauvegarde

Un serveur de sauvegarde, ou serveur de backup, est un système informatique conçu pour stocker des copies de données provenant d'autres systèmes, applications ou serveurs afin de les protéger contre la perte de données, les pannes matérielles ou les sinistres. Il assure la redondance des données et permet leur récupération en cas de besoin.

A la différence des autres scripts, celui la sera déployé sur le serveur de la Clinique 1 puisqu'il servira de serveur de sauvegarde pour les dossiers des équipes et du personnel présent uniquement sur le serveur principal (Le serveur de la clinique 0)

[Lien vers le GitHub qui héberge le script Backup]

```powershell
# Définition des chemins source et de destination
$src1 = "\\ServMainAKAT\PERSO"
$src2 = "\\ServMainAKAT\EQUIPES"
$src3 = "\\ServMainAKAT\BDD-Script"
$dst = "C:\BACKUPS"

# Boucle infinie pour effectuer des sauvegardes à intervalles réguliers
while ($true) {
    # Obtenez la date et l'heure actuelles au format "yyyyMMdd-HHmm"
    $time = (Get-Date).ToString("yyyyMMdd-HHmm")

    # Créez un répertoire de sauvegarde avec le nom basé sur la date/heure actuelle
    $backup_path = "$dst\$time"
    New-Item -ItemType Directory -Path $backup_path

    # Copiez les dossiers sources dans le répertoire de sauvegarde en utilisant Robocopy
    Robocopy "$src1" "$backup_path\PERSO" /E /MIR
    Robocopy "$src2" "$backup_path\EQUIPES" /E /MIR
    Robocopy "$src3" "$backup_path\BDD-Script" /E /MIR

    # Calculez la date limite pour la suppression des sauvegardes (7 jours en arrière)
    $delete_before = (Get-Date).AddDays(-7)

    # Supprimez les dossiers de sauvegarde datant de plus de 7 jours de manière récursive
    Get-ChildItem $dst | Where-Object {$_.LastWriteTime -lt $delete_before} | Remove-Item -Recurse

    # Attendez pendant 3600 secondes (1 heure) avant de lancer la sauvegarde suivante
    Start-Sleep -Seconds 3600
}
```

Figure 22: Script de configuration du sauvegarde

Ce script commence par définir les sources des dossiers à copier sur le serveur principal.
Par la suite, le script ouvre une boucle infinie qui s'exécute en continue :

- Cette boucle crée un répertoire avec la date et l'heure actuelle puis elle copie les dossiers sources vers le répertoire de sauvegarde.
- Ensuite, elle supprime les sauvegardes inutiles (sans changement de fichiers pendant plus de 7 jours.
- Et pour finir, elle attend 1 heure avant de relancer la sauvegarde suivante.

La ligne BDD-Script contient une sauvegarde physique des Scripts AD, DHCP et Backup présent sur le GitHub.

5.g Zabbix

Zabbix est une solution open-source, développée en langage PHP de surveillance et de gestion de la performance des systèmes informatiques.

Open-source (ou code source ouvert en français) se réfère à un logiciel ou à un programme informatique dont le code source est disponible pour tous les utilisateurs. Cela signifie que toute personne peut consulter, modifier et distribuer le code source selon ses propres besoins. PHP (acronyme de Hypertext Preprocessor) est un langage de programmation libre, principalement utilisé pour développer des applications web.

Zabbix est utilisée pour surveiller de nombreuses métriques et performances telles que la disponibilité, l'utilisation des ressources, la capacité, les erreurs, les temps de réponse, etc. Le tout de manière centralisée.

Figure 23: Logo de Zabbix

Cette solution de supervision dispose d'une architecture client-serveur, où les agents Zabbix sont installés sur les systèmes à surveiller et envoient des données de surveillance au serveur Zabbix.

Le serveur collecte ces données, les stocke et les analyse pour générer des alertes en cas de dépassement de seuils définis ou de rupture d'une liaison par exemple. Les alertes peuvent être envoyées par e-mail, SMS, messagerie instantanée, être affichées sur une page web en direct, etc.

Le serveur dispose également d'une interface web personnalisable pour la configuration, la visualisation et l'analyse des données de surveillance. Il peut être utilisé pour surveiller différents types de systèmes tels que les serveurs, les réseaux, les applications, les bases de données, les dispositifs de stockage, etc.

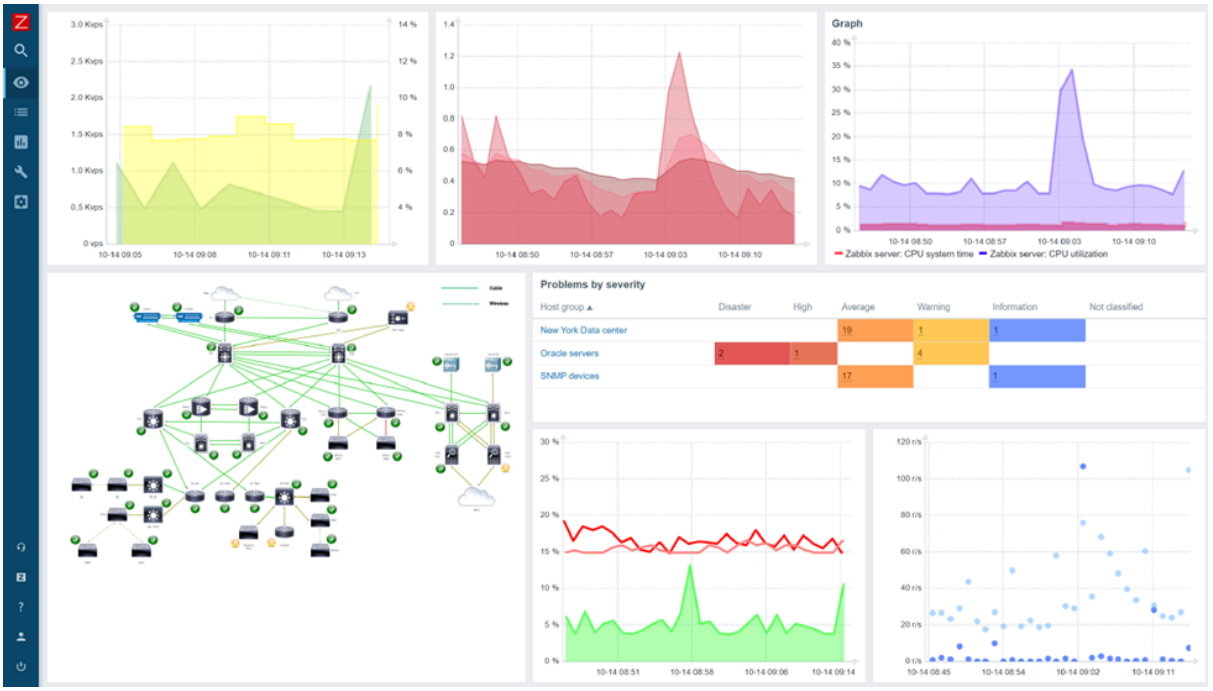


Figure 24: Interface Web de Zabbix

La solution Zabbix peut être utilisée pour surveiller les systèmes critiques et aider les équipes informatiques à détecter rapidement les problèmes, à diagnostiquer les causes profondes et à résoudre les problèmes avant qu'ils ne deviennent des pannes majeures.

Nous avons ici utilisé cette solution de supervision afin de surveiller le réseau des hôpitaux. Nous avons reconstitué la cartographie de notre réseau sur Zabbix en la découpant par VLAN utile à superviser.

Il est très facile de naviguer entre les différentes cartographie de notre réseau car il suffit de cliquer sur l'icône de la cartographie désirée pour s'y rendre.

Par exemple sur l'image ci-dessous, en cliquant sur l'icône en haut à gauche on rejoindra la cartographie de la partie DMZ (VLAN 20).

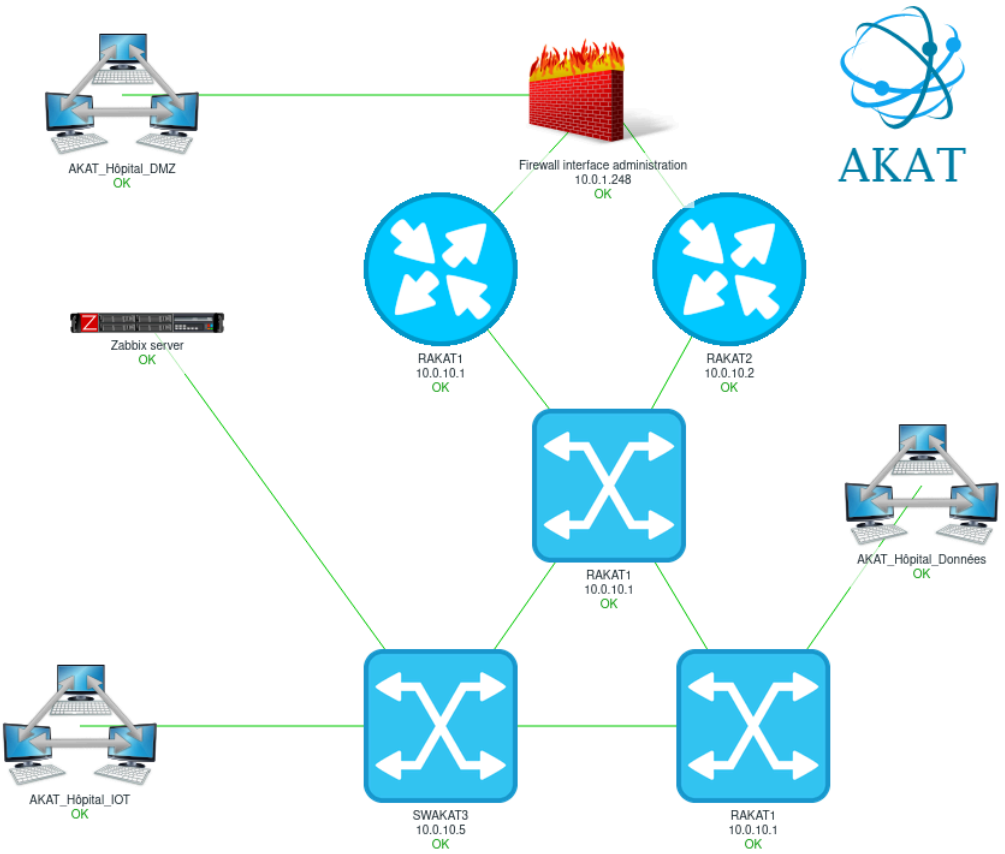


Figure 25: Cartographie de la partie DMZ (VLAN 20)

Voici ici notre coeur de réseau relié à 3 autres cartographies:

- La partie DMZ (VLAN 20)
- La partie IOT (VLAN 90)
- La partie Données (VLAN 50)

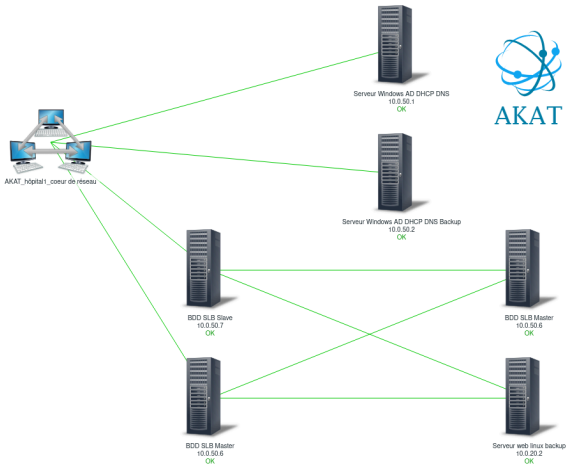


Figure 26: Cartographie de la partie Données (VLAN 50)

Voici la cartographie de la partie Données (VLAN 50).

On retrouve les différents éléments de notre topologie physique que l'on peut monitorer.

L'intérêt est que l'on n'a pas besoin de connaître l'infrastructure physique mise en place et les liens entre les éléments du réseau pour pouvoir récupérer des informations utiles.

On peut simplement se référer à ce qui a été mis en place sur Zabbix.

Voici la cartographie de la partie DMZ (VLAN 20).

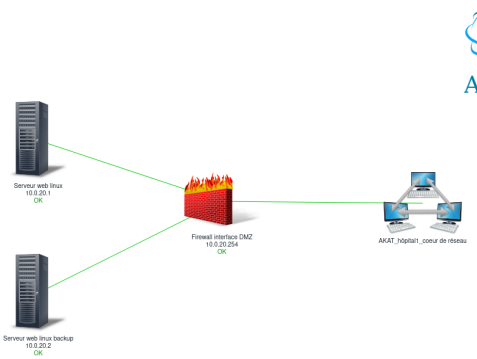
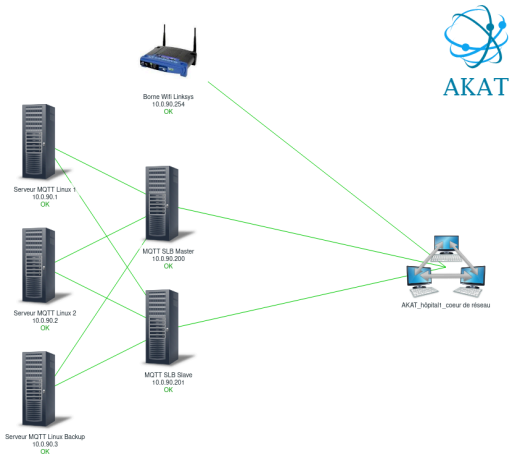


Figure 27: Cartographie de la partie DMZ (VLAN 20)



Voici la cartographie de la partie IOT (VLAN 90)

Figure 28: Cartographie de la partie IOT (VLAN 90)

5.h. Radius

Un serveur RADIUS (Remote Authentication Dial-In User Service) est un système d'authentification et d'autorisation utilisé pour sécuriser l'accès aux réseaux que ce soit l'accès physique ou l'accès à distance.

Nous avons ici choisi l'option freeradius.

Voici un exemple de configuration de freeradius qui dans un premier temps définit les réseaux sur lesquels le serveur radius va opérer dans le fichier clients.conf.

```
  GNU nano 5.4                /etc/freeradius/clients.conf *
client 10.0.10.0/24 {
        secret = ***********
        nastype = cisco
        shortname = switch
}
client 10.0.20.0/24 {
        secret = ***********
        nastype = cisco
        shortname = switch
}
client 10.0.30.0/24 {
        secret = ***********
        nastype = cisco
        shortname = switch
}
client 10.0.40.0/24 {
        secret = ***********
        nastype = cisco
        shortname = switch
}
```

Figure 29: Fichier clients de Freeradius

On définit ensuite dans le fichier users les différents utilisateurs du réseau qui devront s'authentifier pour y avoir accès avec un certain nombre de privilèges. Ici nous avons créé un administrateur et un opérateur.

```
  GNU nano 5.4                    /etc/freeradius/users *
admin   Cleartext-Password := *********
        Service-Type = Administrative-User,
        Cisco-AVPair = "shell:roles = network-admin",
        Cisco-AVPair += "shell:priv-lvl=15"

opérateur       Cleartext-Password := *********
                Service-Type = NAS-Prompt-User,
                Cisco-AVPair = "shell:roles = network-operation",
                Cisco-AVPair = "shell:priv-lvl=1"
```

Figure 30: Fichier users de Freeradius

Il est très facile d'ajouter des utilisateurs tout en leur attribuant des droits adaptés et des clients (réseaux) dans les fichiers de configuration.

L'intérêt de mettre en place un dispositif type Radius est que l'on peut centraliser la sécurité du réseau et la distribuer à nos équipements réseaux par le biais du fichier clients. On n'a donc pas à configurer les droits d'accès localement et ceux-ci sont communs pour chaque groupe d'équipements.

5.i. Broker MQTT

Un broker MQTT est un serveur ou un logiciel qui facilite la communication dans le protocole MQTT (Message Queuing Telemetry Transport). Il reçoit, stocke et achemine les messages entre les clients MQTT, permettant ainsi la mise en place de systèmes de communication à faible consommation d'énergie et adaptés à l'Internet des objets (IoT).

Nous allons détailler son fonctionnement plus en détail :

Publication et Abonnement : Le protocole MQTT fonctionne sur la base d'un modèle de publication/abonnement (pub/sub). Les clients MQTT peuvent être des appareils IoT, des capteurs, des applications, etc. Un client peut "publier" des messages sur un sujet spécifique, et d'autres clients peuvent “s'abonner” à ce sujet pour recevoir les messages. De la même manière qu’un utilisateur le ferait sur les réseaux sociaux par exemple.

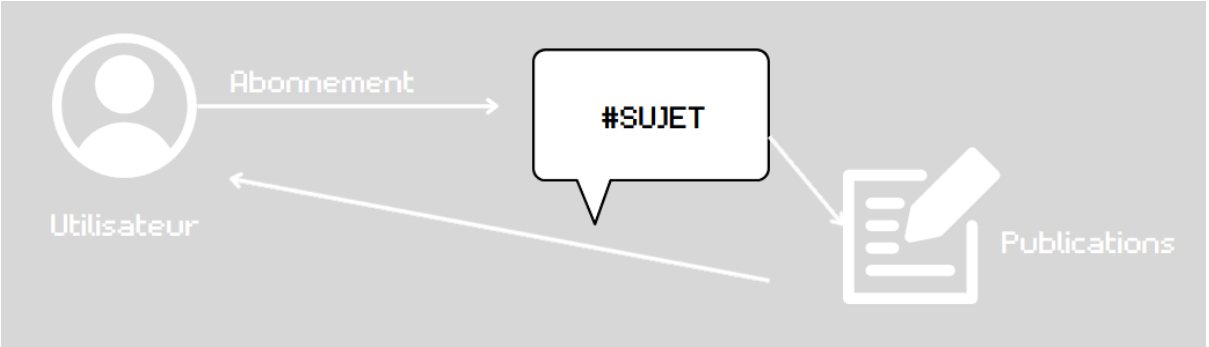


Figure 31: Schéma du fonctionnement de MQTT en faisant un parallèle avec le fonctionnement des réseaux sociaux

Broker MQTT : Le broker MQTT est le serveur centralisé qui gère la distribution des messages entre les clients. Lorsqu'un client publie un message sur un sujet donné, le broker s'assure que ce message est transmis aux clients abonnés à ce même sujet. Il maintient une liste des sujets, des abonnements et des clients connectés.

Communication asynchrone : Le broker MQTT permet une communication asynchrone, ce qui signifie que les clients n'ont pas besoin d'établir une connexion directe les uns avec les autres pour communiquer. Au lieu de cela, ils communiquent via le broker, qui relaie les messages aux destinataires appropriés.

Qualité de service (QoS) : MQTT propose différents niveaux de qualité de service (QoS) pour la livraison des messages. Les niveaux QoS déterminent le niveau de garantie de livraison des messages, allant de 0 (au plus rapide, sans garantie) à 2 (livraison garantie). Le broker gère la gestion de ces niveaux QoS.

Répartition des messages : Le broker MQTT est conçu pour gérer efficacement un grand nombre de messages et de clients. Il peut stocker temporairement les messages destinés aux clients indisponibles jusqu'à ce qu'ils se reconnectent.

Sécurité : Les brokers MQTT prennent en charge divers mécanismes de sécurité, tels que l'authentification des clients, le chiffrement des communications et la défense contre les attaques par déni de service.

En résumé, un broker MQTT est un élément clé de l'architecture MQTT, permettant aux appareils IoT et aux clients de communiquer de manière efficace et asynchrone en utilisant le modèle de publication/abonnement. Il assure la gestion, la distribution et la sécurité des messages au sein du réseau MQTT, ce qui en fait un composant essentiel de nombreuses applications IoT et de messagerie.

5.j. Lecteurs de badge pour contrôle d’accès

Nous utilisons un système de contrôle d'accès basé sur des lecteurs de badges RFID USB pour sécuriser l'accès aux différentes salles de l'hôpital. Le système permet l'identification des utilisateurs autorisés et l'enregistrement de leurs entrées.

Figure 32: Illustration d’un lecteur de Badges RFID USB

Configuration des Lecteurs de Badges RFID USB

Les lecteurs de badges RFID USB sont connectés à des clients MQTT via des ports USB. Ils sont configurés pour lire les identifiants de badges RFID et les transmettre à la base de données via le broker MQTT pour traitement. Les lecteurs sont capables après traitement des informations de détecter les badges autorisés et non autorisés.

Communication avec un Broker MQTT

Pour la transmission des données en temps réel, nous utilisons le protocole MQTT. Chaque lecteur de badge RFID est connecté à un client MQTT configuré pour communiquer avec le broker MQTT. Lorsqu'un badgeage est effectué, les données associées sont publiées sur le broker MQTT. Cela permet une communication rapide entre les lecteurs de badges et la base de données.

Vérification des Badges Autorisés

Lorsqu'un badgeage est effectué, on vérifie d'abord si l'identifiant du badge est présent dans la liste des badges autorisés. Cette liste est stockée dans la base de données. Si le badge est autorisé, l'accès est accordé. Dans le cas contraire, un accès non autorisé est enregistré.

```
MariaDB [Hopital]> select * from Personnel;
+---------------+--------+----------+-------------+-----------+
| N°_Personnel  | Nom    | Prenom   | Occupation  | N°_Carte  |
+---------------+--------+----------+-------------+-----------+
|             1 | Greys  | Annatomy | Secrétaire  |         1 |
|             2 | Tomy   | Anne     | Urgentiste  |       666 |
|             3 | Lepauv | Maccabee | Chirurgien  |       777 |
```

Figure 33: Base de données des cartes

Enregistrement des Données de Badgeage

```
ale5g6reg86,10h30,salle106
g95e6g59ela,11h00,salle104
```

Figure 34: Enregistrement des badges

Chaque badgeage enregistré comprend l'identifiant du badge, l'heure à laquelle il a été utilisé, et le lieu de l'accès.

Ces données sont ensuite stockées dans la base de données et dans un fichier CSV pour assurer un suivi précis de l'utilisation des badges et permettre une analyse ultérieure.

Ainsi l'intégration des lecteurs de badges RFID USB avec le broker MQTT et la vérification des badges autorisés dans la base de données permettent la mise en place d'un système de contrôle d'accès fiable et efficace.

Ce système garantit que seuls les utilisateurs autorisés puissent accéder aux différentes salles de l'hôpital, tout en enregistrant les données essentielles liées à chaque badgeage.

Nous avons réalisé un premier schéma nous permettant de décider quelles données devaient être échangées entre nos lecteurs de cartes, nos broker MQTT et nos bases de données MariaDB.

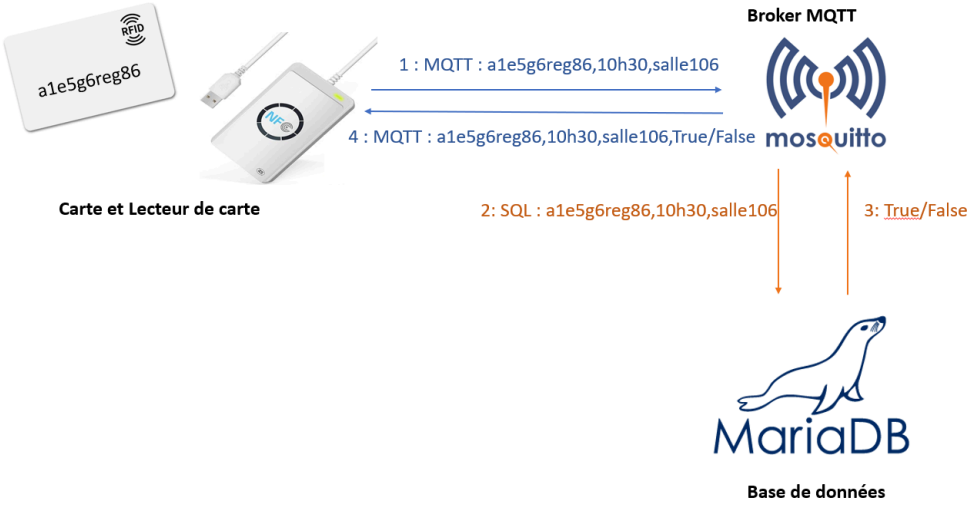


Figure 35: Échanges lors d'un accès par carte à une salle

5.k Outil de gestion de parc

Pour pouvoir gérer le plus facilement les différents appareils du réseau nous avons décidé de mettre en place un outil de gestion de parc nommé Fog Project. Installer un outil de ce type permet :

- Déploiement d'images système : Fog Project permet de créer et de déployer des images système sur plusieurs ordinateurs simultanément. Cela simplifie la tâche de configuration initiale des machines, l'installation du système d'exploitation et des logiciels.

- Inscription et suivi des machines : Il offre la possibilité d'inscrire et de suivre les ordinateurs du parc informatique, ce qui permet de gérer les informations matérielles et logicielles de manière centralisée.

- Gestion des mises à jour : Fog Project facilite la gestion des mises à jour logicielles en permettant de déployer des correctifs et des mises à jour sur l'ensemble du parc informatique.

- Exécution de scripts : L'outil permet d'exécuter des scripts sur un groupe d'ordinateurs, ce qui est utile pour automatiser des tâches spécifiques sur les machines.

- Réduction des coûts et du temps : En automatisant de nombreuses tâches de gestion, Fog Project réduit les coûts opérationnels et le temps nécessaire pour maintenir un parc informatique.

L'un des principaux avantages de cette solution est son interface graphique très simple d'utilisation.

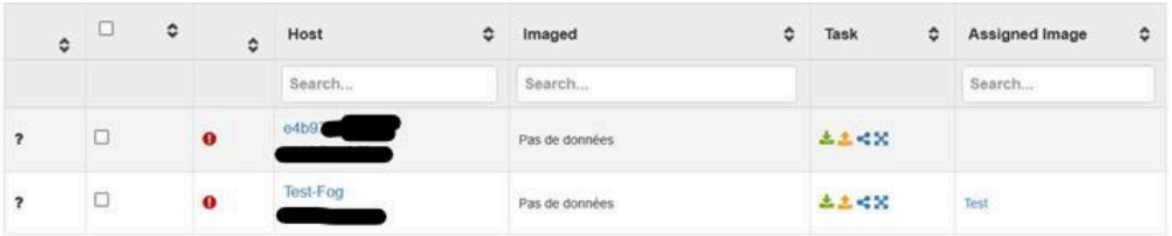


Figure 36: Interface serveur Fog Project

Pour déployer une image nous avons juste besoin d'appuyer sur la petite flèche verte et pour capturer sur le logo jaune .

Nous l'avons installé avec les paramètres suivants :

- Storage Mode = Normal
- Network Interface = No
- Router Address = No
- DHCP to handle DNS = No
- DHCP Service = No
- Language packs = Yes
- HTTPS enable = No
- Change Hostname = Yes --> Fog-srv.akat.fr

Nous avons ensuite mis en place les différents scripts nécessaires au bon fonctionnement des machines lors de leur déploiement.

- fog.custominstall qui exécute le script fog.postdownload
- fog.postdownlaod qui appelle fog.copydriver et funcs.sh
- fog.copydriver qui copie les drivers sur la machine que l'on déploie
- funcs.sh

Pour que les drivers soient correctement copiés sur la machine il faut respecter l'architecture de dossier suivantes sur le serveur fog :

```
/images
├─/drivers
  ├─$machine
      ├─$osn
        └─$arch
```

Ce qui donne dans le cas d'une vraie machine

```
/images
├─/drivers
  ├─Optiplex5050
      ├─win10
        └─x64
```

5.I. Serveur Web

Pour notre serveur web, nous avons choisi d'installer Apache2. Notre serveur web est doublé permettant de répartir la charge entre eux. Dans le temps imparti, nous n'avons pas eu le temps de développer cette partie. Nous avons néanmoins configuré chaque serveur et nous avons développé une première page de connexion qui devait permettre aux utilisateurs authentifiés de prendre des rendez-vous.

Cette page a été développée en PHP et Javascript.

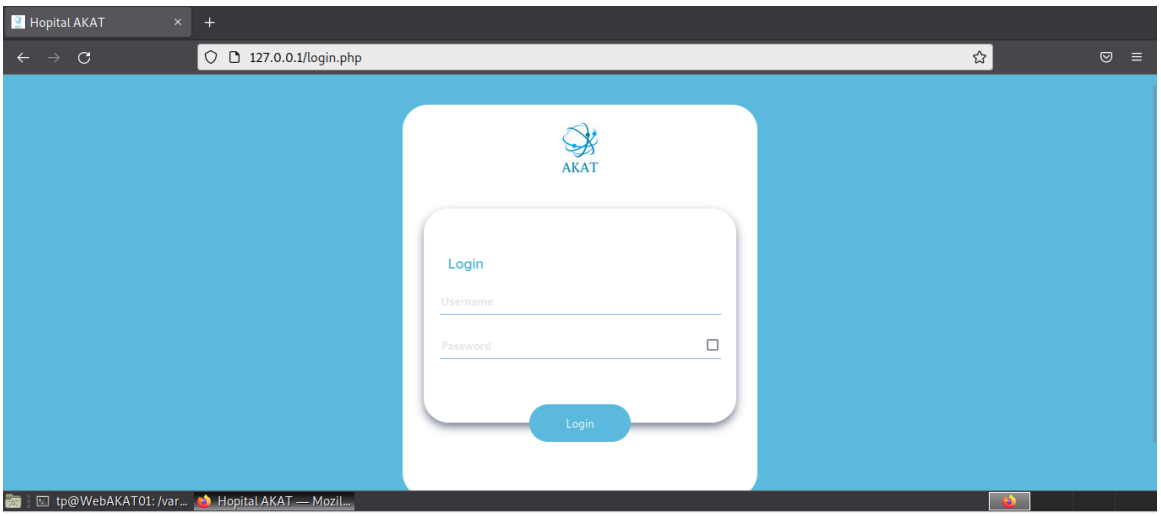


Figure 37: Page de connexion du site Internet

L'accès au serveur web se fait via un contrôleur SLB en mode leastconn. L'intérêt est d'envoyer le trafic sur le serveur le moins chargé pour équilibrer la charge.

6. Scripts

```yaml
database:
  user: "user"
  password: "pass"
  host: "127.0.0.1"
  name: "Hopital"
  raise_on_warnings: true
  dbname: 'db.csv'
```

Figure 38: Fichier YAML utilisé par un script

Pour chacun de nos scripts, nous avons choisi de les décomposer en deux parties.

Les informations de connexions tels que les identifiants mot de passe et les adresses IP se trouvaient dans un fichier YAML.

Cela permet de ne pas avoir à modifier directement le code lorsqu'on veut changer les informations et cela permet également de partager le script plus facilement sans avoir à en modifier le contenu.

Chaque script était déposé dans un dépôt github nous permettant de pouvoir journaliser ceux-ci et de garder des versions fonctionnelles facilement accessibles.

L'intérêt était aussi de permettre de récupérer ces scripts de manière automatique si l'on souhaite en développant un script qui irait chercher directement les programmes dans le dépôt associé.

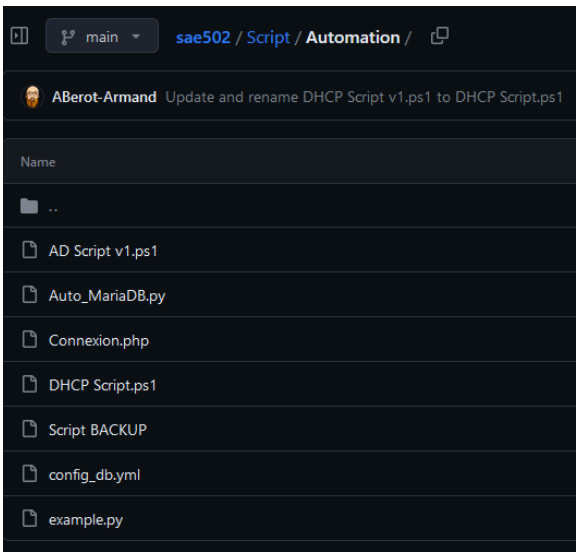


Figure 39: Dépôt github des scripts.

6.a. Gestion des données

```
fichier_csv = '/home/tp/add_personnel.csv'

try:
    connection = aaa

    if connection.is_connected():
        cursor = connection.cursor()

        with open(fichier_csv, 'r') as fichier:
            csv_reader = csv.DictReader(fichier)
            for ligne in csv_reader:
                print(ligne)
                print(ligne["Occupation"])
                action = ligne['Action']
                nom = ligne['Nom']
                prenom = ligne['Prenom']
                occupation = ligne["Occupation"]
                numero_carte = ligne['N°_Carte']

                #------------------------------------------------------------
                #     ▉              Modifying database
                #------------------------------------------------------------
                #Action add +
                if (action == '+'):

                    query = "INSERT INTO Personnel (Nom, Prenom, Occupation, N°_Carte) VALUES (%s, %s, %s, %s)"
                    values = (nom, prenom, occupation, numero_carte)
                    cursor.execute(query, values)
                    connection.commit()
                    print(f"Insertion : {nom} {prenom} - {occupation}")
```

Figure 40: Dépôt github des scripts.

Ce script sert à la fois à ajouter des données dans une table mariadb et à en supprimer.
Il va extraire les informations d'un fichier csv pour ensuite les transformer en la fonction "input mariadb".

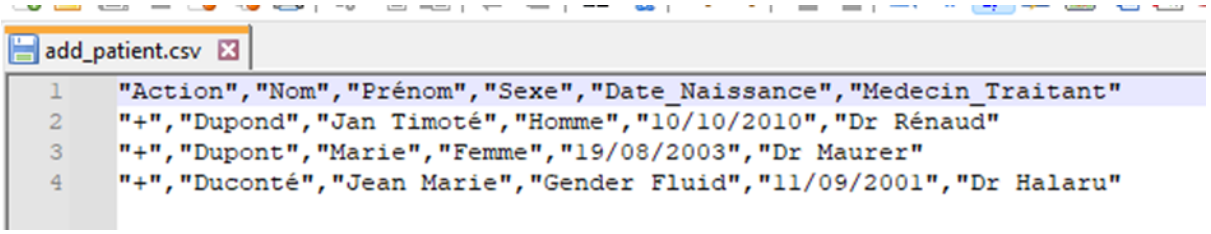


Figure 41: Exemple de CSV

Le script va créer un objet mysql permettant de se connecter à la base de données MariaDB d'après les informations renseignées dans le fichier YAML.

Si la connexion à la base de données est fonctionnelle, alors il va lire le fichier csv contenant les informations à ajouter ou supprimer de la table contenant les utilisateurs. En fonction des valeurs d'action (+ ou -) on modifie les données de la table.

6.b. Tests de redondance

Pour tester plus facilement les redondances nous avons créé deux playbook ansible. Ansible est un langage d'automatisation qui nécessite uniquement un accès SSH aux équipements sur lesquels il doit travailler.

Le gros avantage est que l'on n'a pas besoin d'avoir une API ou un module sur un langage de programmation classique puisque les commandes sont passées comme on le ferait à la main.

```
- hosts: machines_a_eteindre
  become: yes
  tasks:
    - name: Shutdown
      command: shutdown -h 0
```

Figure 42: Playbook ansible

Un playbook ansible se présente sous la forme d'un fichier YAML qui comporte les différentes instructions à exécuter.

Ici la playbook a pour but d'éteindre une machine Linux.

Nous allons l'utiliser pour tester le bon fonctionnement du Server Load Balancing et du VRRP.

Le script sera exécuté sur deux machines différentes donc on renseigne les adresses IP.

```
[machines_a_eteindre]
10.0.90.1
10.0.90.200
```

Figure 43: Fichier ini

Nous avons un deuxième playbook qui a pour but de redémarrer un ou plusieurs routeurs pour faire basculer le trafic.

```
---
- hosts: routeurs_cisco
  gather_facts: no
  tasks:
    - name: reload Cisco Routers
      ios_command:
        commands:
          - reload
      register: result
    - name: Action
      wait_for:
        host: "{{ inventory_hostname }}"
        port: 22
        delay: 300
        state: started
      when: "'Please confirm' in result.stdout_lines | join(' ')"
```

Figure 44: Playbook ansible pour redémarrer des équipements Cisco

Il y a une légère différence dans ce playbook.

En effet, pour redémarrer un équipement Cisco il faut confirmer la commande, on a donc une ligne when qui s'exécute comme une condition dans un langage de programmation qui va permettre de valider la commande.

On voit que la commande envoyée pour redémarrer l'équipement est la même que l'on utiliserait manuellement.

7. Sécurité

La cybersécurité est essentielle pour un hôpital, devenant un enjeu critique en raison de la fréquence des cyberattaques et de leur dangerosité.

Aujourd'hui, les fuites de données confidentielles sont devenues un problème majeur, et les cyberattaques peuvent paralyser des hôpitaux, mettant en danger leur fonctionnement.

Il est impératif de protéger le système informatique hospitalier pour éviter de tels risques et préserver la sécurité des patients et des opérations médicales.

7.a. VLAN

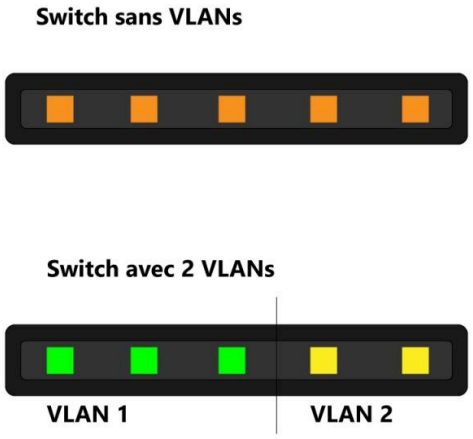


Figure 45: Schéma de deux Switch, un sans VLAN et l'autre avec 2 VLAN

Comme expliqué dans la partie 3. Plan d'adressage IP, nous avons mis en place des VLANs. Leur but est d'améliorer la sécurité et faciliter l'administration du réseau en segmentant et en cloisonnant un réseau physique en plusieurs réseaux virtuels.

En effet, avec cette segmentation, les clients du VLAN vert ne peuvent pas directement contacter des clients du VLAN jaune. Ce n'est pas le cas sans les VLANs. Le passage d'un VLAN à un autre n'est possible que si l'on met en place du routage inter vlan.

Mettre des VLANs permettant de séparer les clients en fonction de leur rôle (Administrateur, Médecin, Secrétaire etc…) permet donc une première limitation de l'accès aux ressources par les personnes concernées qu'il faudra compléter avec des droits d'accès par la suite.

On comprend donc mieux en quoi les VLANs contribuent à un premier niveau de sécurité.

7.b. Access-lists

Comme expliqué dans la partie précédente, nous avons mis en place des VLANs. Pour permettre aux clients de pouvoir accéder à des serveurs qui ne se trouvent pas dans le même VLAN qu'eux, nous avons dû configurer du routage inter vlan. Le problème de cette fonctionnalité est que les clients qui étaient virtuellement séparés entre eux peuvent maintenant y accéder.

Cela veut également dire que les utilisateurs ont accès au réseau d'Administration et peuvent théoriquement se connecter aux équipements réseaux.

Nous avons donc mis en place des Access-lists pour filtrer les accès en fonction de plusieurs paramètres :

- Le réseau / host source
- Le port source
- Le réseau / host destination
- Le port destination
- Le protocole de transport utilisé

Il faut savoir que les access-list fonctionnent de manière hiérarchique, c'est-à-dire que lorsqu'un paquet va arriver sur l'équipement réseau, chaque ligne sera testée et la première qui correspond sera utilisée.

Cela veut donc dire qu'il faut penser à mettre les autorisations avant les blocages. Il faut également expressément créer un blocage à la fin car tout ce qui n'est pas refusé est autorisé.

C'est pour cette raison qu'il faut absolument une ligne "deny ip any any" à la fin de chaque access-list.

```
!--- IOT subnet to database
permit ip 10.0.90.0 0.0.0.255 host 10.0.50.3
permit ip 10.0.90.0 0.0.0.255 host 10.0.50.4
permit ip 10.0.90.0 0.0.0.255 host 10.0.50.5
permit ip 10.0.90.0 0.0.0.255 host 10.0.50.6
permit ip 10.0.90.0 0.0.0.255 host 10.0.50.7
permit ip 10.0.90.0 0.0.0.255 host 10.0.50.8
permit ip 10.0.90.0 0.0.0.255 host 10.1.50.3
permit ip 10.0.90.0 0.0.0.255 host 10.1.50.4
permit ip 10.0.90.0 0.0.0.255 host 10.1.50.5

!--- Deny all
deny ip any any
```

Figure 46: Une partie de l'access-list de nos routeurs

Lors de nos tests nous nous sommes rendu compte d'une erreur, en effet, tout ce qui n'est pas autorisé est bloqué. Nos clients n'arrivaient donc pas à joindre le serveur DHCP pour demander une adresse IP. Nous avons résolu ce problème en ajoutant les lignes suivantes :

```
permit any 255.255.255.255
permit 255.255.255.255 any
```

Après avoir fait des tests, cela semble avoir corrigé le problème.

7.c. Pare-feux

La mise en place d'un pare-feu en bordure d'un réseau d'hôpital est d'une importance cruciale. Nous avons déployé ce dispositif de sécurité pour sécuriser l'accès au réseau. Le pare-feu joue un rôle essentiel en filtrant les flux de données entrants et sortants, ce qui permet de détecter et de contrer les menaces potentielles via un ensemble de règles que nous avons défini, tout en garantissant la confidentialité, l'intégrité et la disponibilité des informations médicales.

Cette mesure de sécurité est essentielle pour assurer le bon fonctionnement du réseau et protéger les données sensibles des patients. De plus, elle contribue à la conformité aux réglementations du secteur de la santé en matière de sécurité des données.

Notre pare-feu permettait également de rendre accessible nos serveurs web sur Internet sans pour autant compromettre la sécurité globale de notre réseau.

Par manque de matériel, nous n'avons pas pu utiliser deux pare-feux Stormshield, nous avons donc configuré et mis en place un système d'exploitation Linux (IPfire) qui jouait le rôle de notre deuxième pare-feu. Nous avons choisi cette solution car elle est assez similaire aux pare-feux Stormshield dans leur manière de configurer les règles et dans le fonctionnement des interfaces.

Dans le temps imparti, nous n'avons pas eu le temps de configurer entièrement IPfire, nous avons néanmoins pu l'installer et lui configurer ses adresses IP.

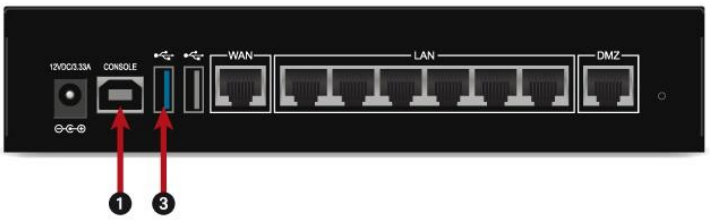


Figure 47: Illustration des ports d'un pare-feu Stormshield

En effet, les pare-feux Stormshield possèdent trois types d'interfaces. L'interface out/wan est reliée à Internet, les interfaces in/LAN permettent d'accéder au réseau local.

Le but d'une DMZ est de rendre accessible une partie du réseau sur Internet sans mettre en danger le réseau local.

Dans notre contexte, nous avons utilisé l'interface DMZ pour permettre l'accès à nos serveurs web.

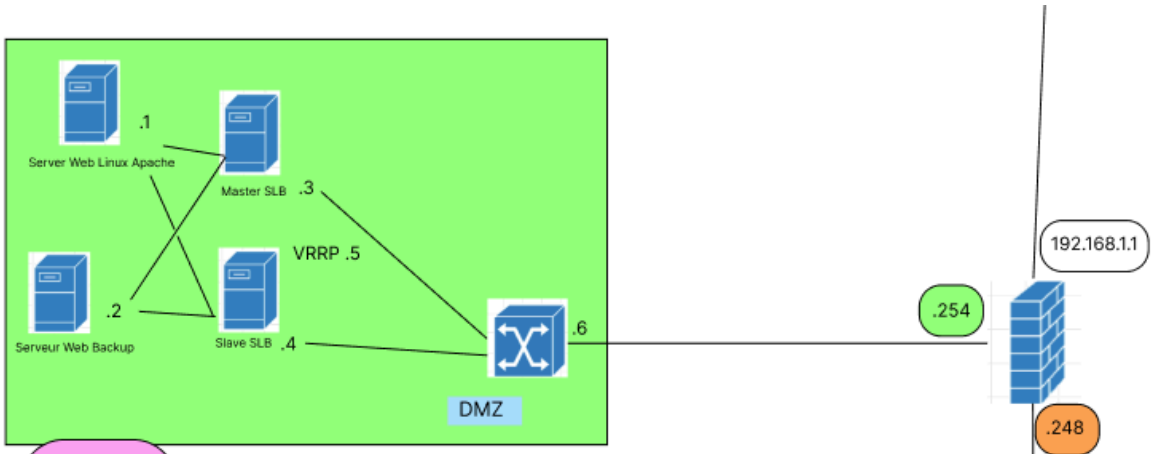


Figure 47: Schéma du pare-feu et de la DMZ

Avec IPfire, les interfaces in, out, et dmz sont remplacées par les interfaces red, green et orange mais le fonctionnement reste le même.

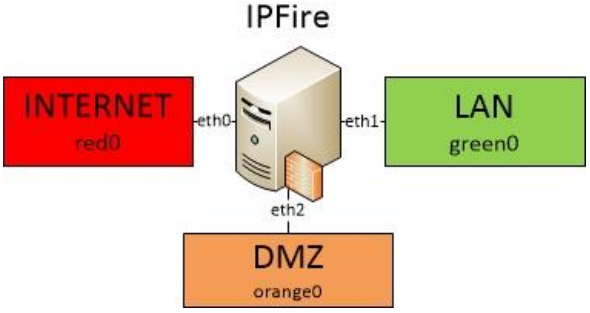


Figure 48: Schéma des interfaces du pare-feu IPfire

Figure 49: Interfaces du pare-feu Stormshield

Sur notre pare-feu Stormshield, nous avons configuré différents éléments.

Tout d'abord nous avons renseigné les adresses IP de chaque interface. Nous avons ensuite mis en place les différentes routes permettant à notre pare-feu de joindre nos différents réseaux. Chaque pare-feu est passerelle par défaut de l'autre (default gateway), de cette manière nous pouvons accéder aux réseaux des deux sites.

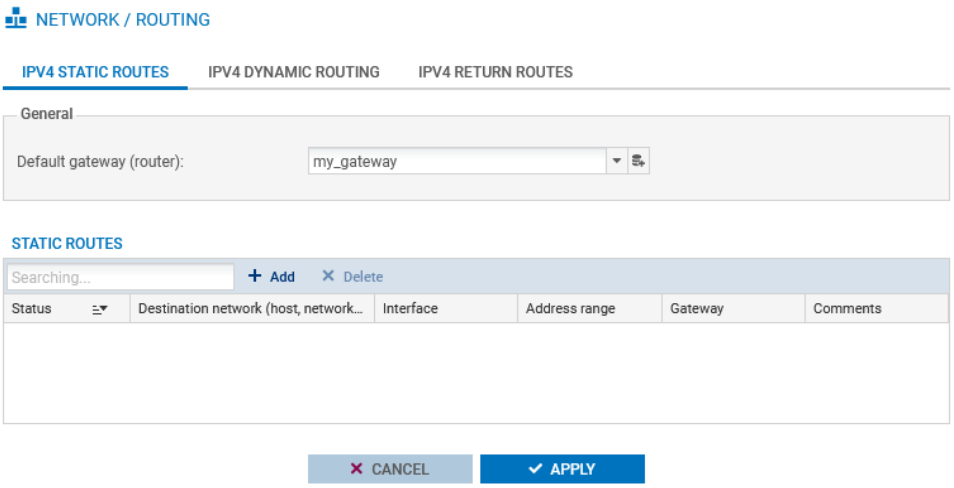


Figure 50: Page de configuration des routeurs sur le pare-feu Stormshield

Nous avons par la suite créé un ensemble d'objets pour nos machines et réseaux. Ceux-ci nous ont été utilisés lors de la création des différentes règles.

Ces règles se décomposent en 4 sections.

Tout d'abord les règles pour les réseaux distants (site 1) vers nos réseaux locaux (site 0).

Ensuite la règle autorisant tout le trafic en HTTP et HTTPS vers notre DMZ.

Puis les règles pour les réseaux locaux (site 0) vers les réseaux distants (site 1).

Enfin un block_all, cette règle est très importante pour interdire tout trafic ne correspondant à aucune règle définie.

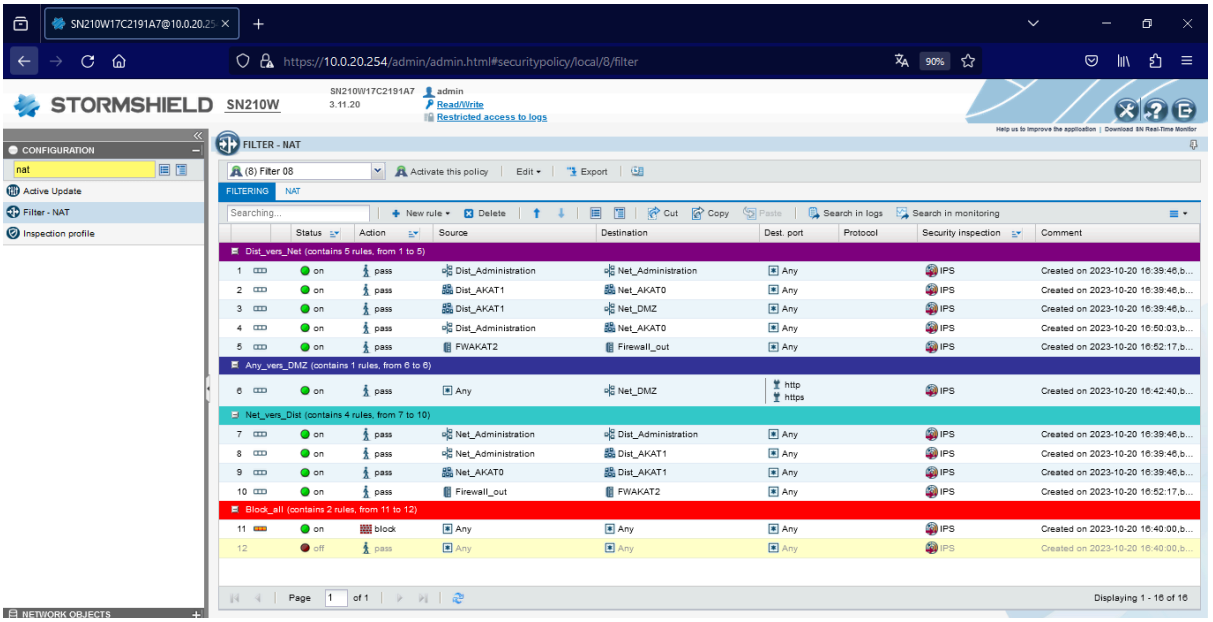


Figure 51: Page de configuration des règles de sécurité sur le pare-feu Stormshield

Le fonctionnement des règles est hiérarchique, c'est à dire que chaque règle est testée une par une et si aucune ne correspond alors le block_all se trouvant en dernière ligne bloquera le trafic. Le fait d'avoir séparé les règles en sections permet de les rendre plus lisibles.

7.d. Radius

Comme expliqué dans la partie 5.g. Radius est un service permettant de gérer les droits d'accès au réseau de manière centralisée.

Il est impératif de souligner l'importance de restreindre l'accès aux données sensibles, en particulier dans le secteur médical et hospitalier. L'utilisation de solutions telles que Radius s'avère cruciale pour garantir la confidentialité des informations des patients et la sécurité des données.

En mettant en place des mécanismes de contrôle d'accès robustes, le réseau garantit une bonne gestion de l'accès aux informations sensibles.

7.e. SSL/TLS et Certificats

SSL/TLS est un protocole de sécurité qui permet de chiffrer les communications, assurant la confidentialité et l'intégrité des données échangées.

Les certificats SSL/TLS sont des fichiers numériques émis par une autorité de certification (CA) qui vérifient l'identité d'un serveur et assurent que les communications avec celui-ci soient sécurisées. Ils contiennent des informations sur le propriétaire du site, la clé publique du serveur et d'autres détails, permettant ainsi de garantir l'authenticité et la sécurité des connexions.

Avant d'entrer davantage dans les détails de la configuration, il faut bien comprendre comment fonctionnent les certificats et leur environnement.

Une autorité de certification (AC) délivre des certificats pour les composants d'une infrastructure. Dans notre cas, nous avons créé notre propre CA : AKAT_CA

Les serveurs/équipements soumettent une demande de certificat pour vérification d'identité.

L'AC intermédiaire doit être vérifiée par l'autorité de certification racine en suivant trois étapes: création de clés, génération d'une demande de certificat, et vérification de l'identité de l'ACI.

1. La hiérarchie de certification peut comporter jusqu'à trois niveaux : une AC unique, une AC avec une ou plusieurs ACI.

   Dans notre cas, les certificats pour le web étaient signés par plusieurs AC que nous avons créés. Mais dans le contexte de MQTT, les certificats ne peuvent pas être signés par plus d'une AC.

2. Les composants de l'architecture disposent de leurs clés privées et certificats. HTTPS se limite généralement au niveau serveur, tandis que MQTTS peut s'étendre jusqu'au niveau client.

3. La mise en place de cette structure exige une compréhension avancée de la sécurité réseau, prend du temps à déployer, et nécessite la régénération des certificats pour remplacer les anciens. La durée de validité des certificats doit être planifiée avec attention, et un mécanisme de révocation doit être prévu en cas de besoin.

La mise en place de ces deux fonctionnalités est très importante lorsque l'on souhaite sécuriser les échanges sur le réseau. Dans notre cas nous avons utilisé les certificats SSL dans le cadre de nos serveurs web et de nos broker et clients MQTT.

Nous avons donc commencé par créer le certificat de notre CA : AKAT_CA. Celle-ci signera nos certificats.

```
tp@debian:/tmp$ openssl x509 -in cacert.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            06:9c:a5:0b:5c:c3:78:b6:1e:28:cf:e7:61:74:c3:5f:5f:54:38:bc
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FR, ST = France, L = Montb\C3\83\C2\A9liard, O = AKAT_CA, OU = AKAT, CN = AKAT, emailAddress = certificate@akat.com
        Validity
            Not Before: Nov  4 14:40:17 2023 GMT
            Not After : Nov  1 14:40:17 2033 GMT
        Subject: C = FR, ST = France, L = Montb\C3\83\C2\A9liard, O = AKAT_CA, OU = AKAT, CN = AKAT, emailAddress = certificate@akat.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

Figure 52: Création d'un certificat CA root

Ci-dessous se trouve le certificat créé pour nos serveurs web. Le nom DNS (akat.fr) est porté par l'adresse VRRP partagée entre nos serveurs et des alias ont été créés pour que le certificat soit valable sur chaque serveur.

```
tp@debian:/tmp$ openssl x509 -req -in serveur.csr -CA cacert.pem -CAkey caroot.key -CAcreateserial -out serveur.crt -days 365
Signature ok
subject=C = FR, ST = France, L = Montb\C3\83\C2\A9liard, O = Clinique_AKAT, OU = Clinique_AKAT0, CN = akat.fr, emailAddress = certificate@akat.com
Getting CA Private Key
CA certificate and CA private key do not match
140098791417152:error:0B080074:x509 certificate routines:X509_check_private_key:key values mismatch:../crypto/x509/x509_cmp.c:303:
```

Figure 53: Création d'une demande de certificat

Il faut néanmoins penser à créer des clés privées et publiques pour nos serveurs web pour permettre le bon fonctionnement du chiffrement.

7.f. VPN

Pour mettre en place notre VPN nous avions plusieurs possibilités, nous aurions pu mettre en place un VPN IPsec site à site si nous avions eu deux pare-feux Stormshield.

Nous avons plutôt choisi de nous rabattre sur deux autres options :

- Tunnels MPLS over OSPF
- VPN IPsec gérer par les routeurs Cisco

Nous avons fait le choix de configurer un VPN IPsec avec nos routeurs et nous avons préparé une configuration pour des tunnels MPLS que nous n'avons pas eu le temps de mettre en place. La configuration se fait en miroir et sera faite sur les 4 routeurs en utilisant l'adresse VRRP partagée.

Il faut tout d'abord configurer la méthode d'authentification (ici clé partagée) et les mécanismes d'encryption utilisés.

```
crypto isakmp policy 10
 authentication pre-share
 encryption aes
 hash sha
 group 2
!
```

Ensuite, on doit préciser avec qui on partage notre clé et son contenu.

```
crypto isakmp key cle_akat address 10.1.1.251
!
crypto ipsec transform-set akat_vpn esp-aes esp-sha-hmac
!
```

On crée une carte crypto nommée "AKAT_CRYPTO_MAP" avec un numéro de séquence 10. La carte crypto est une liste d'ensemble de politiques de sécurité qui seront appliquées au trafic réseau. On utilise l'access-list 35 pour gérer le trafic.

```
crypto map AKAT_CRYPTO_MAP 10 ipsec-isakmp
 set peer 10.1.1.251
 set transform-set akat_vpn
 match address 35
```

On doit donc penser à créer l'access-list 35 pour autoriser le trafic venant du réseau du routeur pair. Dans la mesure où le trafic est déjà filtré par le pare-feu côté extérieur et les access-list côté intérieur, on simplifie l'access-list.

```
access-list 100 permit ip any 192.168.2.0 0.0.0.255
```

On modifie ensuite la configuration de notre interface qui nous lie à l'autre routeur en ajoutant la crypto map utilisée pour le tunnel IPsec.

```
interface GigabitEthernet0/1
 description Vers FWAKAT01
 ip address 10.0.1.249 255.255.255.0
 standby 1 ip 10.0.1.251
 standby 1 preempt
 duplex auto
 speed auto
 crypto map AKAT_CRYPTO_MAP
```

8. Redondance et continuité de service

La redondance et la continuité de service sont tous deux des points essentiels pour un réseau informatique. Mais ils le sont encore plus dans un réseau avec un usage critique comme celui d'un hôpital. On doit pouvoir accéder aux données en continue sans blocage pour ne pas compromettre le bon fonctionnement de l'hôpital. C'est pour cette raison que nous avons choisi de mettre l'accent sur ce point.

8.a. VRRP/HSRP

```
interface GigabitEthernet0/0.50
 encapsulation dot1Q 50
 ip address 10.0.50.253 255.255.255.0
 ip helper-address 10.0.50.1
 ip helper-address 10.0.50.2
 standby 50 ip 10.0.50.254
 standby 50 priority 110
 standby 50 preempt
```

Figure 54: Configuration HSRP sur le routeur Passif

VRRP « virtual router redundancy protocol » est un protocole réseau permettant de partager une adresse IP entre plusieurs équipements du même groupe.

L'intérêt est que si l'équipement actif qui porte l'adresse IP virtuelle tombe en panne un autre équipement du même groupe prendra le trafic et assurera la continuité de service et la redondance.

Cela peut également permettre de faire la maintenance d'un équipement sans pour autant couper le trafic en basculant celui-ci sur un autre membre du groupe VRRP.

Nous avons également mis en place la fonctionnalité HSRP sur nos routeurs Cisco, cette fonctionnalité se repose sur VRRP.

```
interface GigabitEthernet0/0.50
 encapsulation dot1Q 50
 ip address 10.0.50.252 255.255.255.0
 ip helper-address 10.0.50.1
 ip helper-address 10.0.50.2
 standby 50 ip 10.0.50.254
 standby 50 preempt
```

Figure 55: Configuration HSRP sur le routeur Actif

Le routeur prioritaire est celui qui a le chiffre le plus bas, par défaut la valeur est de 100 et n'est pas renseignée dans la configuration de l'actif.

L'intérêt est que l'on peut renseigner une passerelle par défaut unique pour nos clients du réseau même si nous disposons de plusieurs routeurs. L'avantage est que nous n'aurons pas besoin de modifier la configuration si une panne avait lieu.

On peut observer cette bascule avec un ping vers la passerelle en la débranchant du réseau pendant le test.

```
ping 10.0.50.254

Envoi d’une requête 'Ping'  10.0.50.254  avec 32 octets de données :

Réponse de 10.0.50.254 : octets=32 temps=3 ms TTL=64 (Routeur Actif)

Réponse de 10.0.50.254 : octets=32 temps=3 ms TTL=64 (Routeur Actif)

Réponse de 10.0.50.254 : octets=32 temps=4 ms TTL=64 (Routeur Actif)

Réponse de 10.0.50.254 : octets=32 temps=18 ms TTL=64 (Routeur Actif)

Réponse de 10.0.50.254 : Impossible de joindre l’hôte de destination. (Bascule)

Réponse de 10.0.50.254 : Impossible de joindre l’hôte de destination. (Bascule)

Réponse de 10.0.50.254 : octets=32 temps=8 ms TTL=64 (Routeur Passif)

Réponse de 10.0.50.254 : octets=32 temps=6 ms TTL=64 (Routeur Passif)
```

Nous avons une bascule automatique presque instantanée ce qui limite fortement l’impact d’une panne sur les clients du réseau.

Nous avons aussi configuré le protocole VRRP sur nos serveurs qui disposaient d’une redondance et sur nos load balancer (partie 8.b. Server Load Balancing). Cela permettait d’avoir le même mécanisme d’actif passif.

Dans le cas de nos serveurs web, pour permettre un basculement invisible pour les clients, le nom DNS sur serveur (akat.fr) était porté par l’adresse VRRP.

Voici un autre exemple de l’usage de VRRP avec notre base de données côté Hôpital 0 grâce au paquet Keepalived sur Linux.

On retrouve les mêmes paramètres que sur la configuration Cisco avec le groupe (virtual router id) et les priorités.

```
vrrp_instance VI_2 {
        state MASTER
        interface enp0s8 //Changer la carte si besoin
        virtual_router_id 52
        priority 100
        advert_int 1
        virtual_ipaddress {
                10.0.50.5/24
        }
}
```

Figure 56: Configuration de Keepalived sur le serveur Actif (Master)

```
vrrp_instance VI_2 {
        state BACKUP
        interface enp0s8 //Changer la carte si besoin
        virtual_router_id 52
        priority 200
        advert_int 1
        virtual_ipaddress {
                10.0.50.5/24
        }
}
```

Figure 57: Configuration de Keepalived sur le serveur Passif (Backup)

8.b. SLB : Server Load Balancing

Le SLB est un protocole réseau permettant de répartir la charge entre plusieurs équipements via un algorithme de répartition de charge. L'équilibrage de charge est important dans un réseau pour permettre d'éviter de monopoliser un serveur réduisant ses performances et augmentant son temps de réponse en répartissant le travail entre plusieurs serveurs.

Pour permettre la mise en place de cette fonctionnalité, nous avons installé le paquet HAproxy sur nos machines Linux.

Dans le contexte de HAproxy, un équipement qui gère le Server Load Balancing est appelé Contrôleur SLB.

L'exemple suivant vous présentera une configuration pour la base de données.

Nous avons choisi le fonctionnement suivant :

- Le trafic est envoyé sur l'adresse VRRP partagée entre nos deux serveurs MariaDB de l'Hôpital 0. L'intérêt est qu'on a déjà une première redondance au sein du même site.

- Dans le cas où les deux serveurs de bases de données de l'Hôpital 1 ne seraient pas accessibles, alors on enverrait le trafic vers l'adresse VRRP partagée entre nos deux serveurs côté Hôpital 1.

On ne bascule sur l'Hôpital 1 qu'en cas de défaillance des deux serveurs de l'Hôpital 0. On utilise donc un modèle actif backup.

```
listen mariadb
  bind *:3306
  mode tcp
  timeout client 3h
  timeout server 3h
  option tcplog
  balance roundrobin
  server bdd1 10.0.50.5:3306 check
  server bdd2 10.1.50.5:3306 check backup
```

Figure 58: Configuration de HAproxy pour nos serveurs de bases de données

On peut faire gérer plusieurs Pool SLB à un contrôleur mais nous avons fait le choix d'en mettre en place un actif et un passif par usage pour mieux répartir le trafic.

On renseigne donc le réseau et le port d'écoute. Ici on gère la répartition de charge de nos serveurs MariaDB.

On renseigne nos membres du pool, ici l'adresse VRRP des serveurs de l'Hôpital 0 et l'adresse VRRP des serveurs de l'Hôpital 1 en Backup.

Pour cette configuration l'algorithme de répartition de charge à utiliser doit être obligatoirement renseigné mais ne sera pas utilisé car on a précisé que l'on faisait de l'actif backup.

Nous avons donc configuré les contrôleurs sur le même modèle. L'algorithme de répartition de charge a été choisi spécifiquement en fonction de l'usage du Load Balancing.

- Leastconn pour nos serveurs web, le serveur le moins chargé prend le trafic.
- Round Robin pour nos broker MQTT, les requêtes vont sur le serveur 1 puis 2 puis 3 pour chaque client.
- Actif / Backup pour la base de données.

Avec nos Load Balancer, nous pouvons garantir un équilibrage de charge conséquent. Mais avec un seul contrôleur SLB la redondance reste faible et en cas de panne le bon fonctionnement du réseau serait compromis.

C'est pour cette raison que nous avons configuré plusieurs Load Balancer pour le même usage (Base de données, serveur web etc…) qui se partagent une adresse IP grâce au protocole VRRP présenté dans la partie précédente. L'intérêt est que cela permet une vraie robustesse du réseau et une tolérance aux pannes très élevée permettant une bonne continuité de service.

Physiquement les contrôleurs SLB sont en parallèles des serveurs mais ils se trouvent devant ceux-ci logiquement. En effet, on ne passe plus directement par les serveurs pour les contacter mais par les Load Balancer qui nous renvoient sur le serveur choisi en fonction de l'algorithme.

Par exemple pour le serveur web, logiquement on aurait :

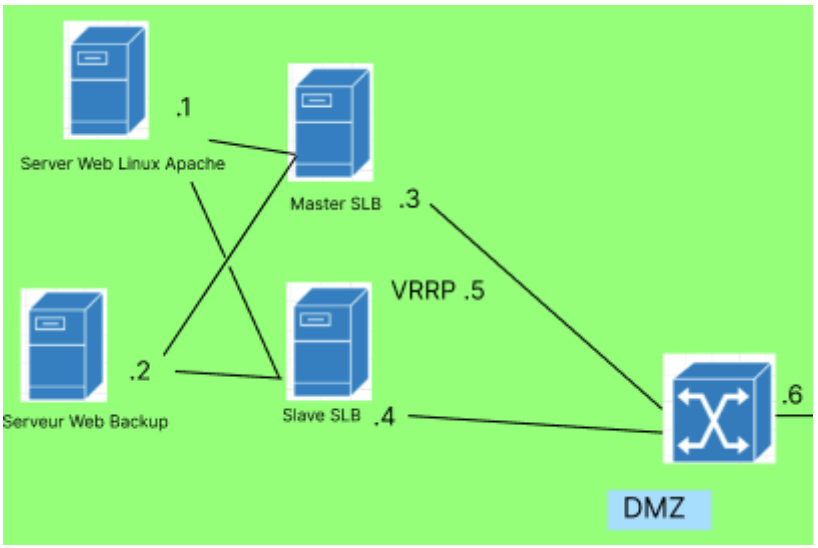


Figure 59: Schéma logique des Load Balancer pour les serveurs Web

Alors que physiquement tous ces équipements sont en parallèle.

8.c. Spanning-Tree

Le Spanning-Tree (STP) est un protocole réseau utilisé pour éviter les boucles dans les réseaux Ethernet commutés.

Le fonctionnement du STP repose sur l'élection d'un commutateur racine (Root Bridge) parmi l'ensemble des commutateurs du réseau. Une fois le commutateur racine élu, le protocole détermine les chemins les plus courts de chaque commutateur vers le commutateur racine, désignant ainsi un unique chemin de communication pour chaque paire de commutateurs. Les autres chemins, qui créeraient des boucles, sont bloqués.

Le Spanning-Tree en tant que tel ne contribue pas à la continuité de service et à la redondance. Mais son utilisation prend tout son sens dans ces domaines.

En effet, nous avons mis en place une boucle de 3 commutateurs sur chaque réseau. Le but était de pallier les pannes d'un des équipements.

Le STP permet d'exploiter efficacement la redondance des liaisons dans un réseau. En bloquant les chemins redondants, le réseau reste opérationnel tout en évitant les boucles. Si l'une des liaisons principales venait à tomber en panne, le STP peut réactiver rapidement une liaison redondante pour maintenir la connectivité.

Le Spanning-tree est donc utilisé pour les liens entre nos switchs Cisco. Nous avons mis en place la configuration suivante :

```
spanning-tree mode rapid-pvst
spanning-tree extend system-id
spanning-tree vlan 1,10,20,30,40,50,60,90 priority 24576
```

Figure 60: Configuration du STP

On doit renseigner le mode de STP utilisé, ici rapid-pvst.

On ajoute ensuite l'ensemble des VLAN utilisés sur les liaisons et on précise la priorité du protocole sur chaque équipement.

Le switch avec la priorité la plus basse prend le trafic. Lorsque cette fonctionnalité est mise en place, on peut voir qu'un port de la boucle se ferme automatiquement pour empêcher la boucle de parasiter le réseau.

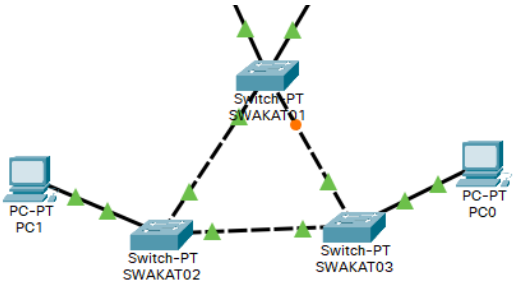


Figure 61: Blocage d'un port (en orange) grâce au STP

8.d. Sauvegarde 321

La sauvegarde **321** est une approche pour la redondance des données. Elle se base sur **3** copies d'un fichier, **2** supports différents (NAS et local) et **1** copie hors site (usb_drive).

```
#!/bin/bash

source_directory="/root/script"
nas_backup_directory="//10.0.50.1/partage/backup/"
local_backup_directory="/root/save/script/"
github_backup_directory="/root/save/github/"
backup_filename="backup_$(date +'%Y%m%d_%H%M%S').tar.gz"

############################
#    Local save copy       #
############################
tar -czvf "$local_backup_directory$backup_filename" "$source_directory"

############################
#         NAS save         #
############################
cp "$local_backup_directory$backup_filename" "$nas_backup_directory$backup_filename"

############################
#         Github           #
############################
cd "$github_backup_directory"
git add "$backup_filename"
git commit -m "Nouvelle sauvegarde $(date +'%Y-%m-%d %H:%M:%S')"
git push https://github.com/ThomasM2568/sae502/script/automation.git master

############################
#     Local save           #
############################
additional_local_backup_directory="/mnt/usb_drive"
cp "$local_backup_directory$backup_filename" "$additional_local_backup_directory$backup_filename"


echo "Done"
```

Figure 62: Script de sauvegarde 321 en bash

Nous avons développé ce script en Bash car cela nous permettait de facilement passer les commandes de copies et de git sans avoir à utiliser de modules comme en Python par exemple.

L'intérêt d'un tel dispositif est que l'on a toujours accès à une sauvegarde des données (ici des scripts) sur plusieurs supports ce qui permet de faire face à de nombreux événements.

9. Améliorations

Avec plus de temps, nous aurions pu développer davantage plusieurs points :

1. Créer des règles de pare-feux et des access-lists plus poussées. En effet, nous avons créé des règles qui fournissent déjà une première protection du réseau mais nous aurions pu filtrer plus finement le trafic pour n'autoriser que le nécessaire.

2. Automatiser davantage de tâches, par exemple des exports de bases de données plus poussés qui feraient des copies sur plusieurs serveurs de sauvegarde et serait capable de remplir la base de données à partir de ces mêmes fichiers.

3. Mettre en place les tunnels MPLS.

4. Développer davantage la partie lecteur de carte RFID. Nous aurions pu par exemple ajouter une LED rouge et une LED verte à allumer en fonction de l'autorisation ou non de l'accès à une salle. Nous aurions pu le compléter avec un écran pour afficher en cas de refus d'accès les raisons et les personnes à contacter en cas de besoin.

5. Nous avions préparé un réseau pour la téléphonie sur IP mais nous n'avons pas eu le temps de l'implémenter dans le temps imparti, nous aurions pu continuer à avancer sur ce point.

10. Devis

En pièce-jointe.

11. Conclusion

La mise en place d'un réseau hospitalier est une tâche qui nécessite une bonne planification. Tout au long de cette Saé, nous avons examiné de nombreux aspects, de la gestion de projet à la mise en œuvre de services essentiels, en passant par la sécurité et la redondance pour garantir la continuité des services.

La gestion de projet joue un rôle central dans la réalisation de cette Saé. L'organisation et le travail collaboratif sont essentiels pour le succès d'un projet de ce type. Une bonne communication et une collaboration efficace entre les membres de l'équipe sont nécessaires pour garantir un projet abouti.

Le cahier des charges et le plan d'adressage IP établissent des bases solides pour la conception du réseau. Ils définissent les exigences et les plages d'adresses IP, ce qui permet une planification précise de l'architecture du réseau.

L'architecture du réseau, y compris le cœur du réseau et les différents sites, a été pensée pour répondre aux besoins spécifiques de l'hôpital. Les services essentiels, tels que la base de données, le DNS, l'Active Directory et d'autres, sont mis en place pour assurer un fonctionnement fluide de l'infrastructure.

La sécurité est un enjeu crucial dans le domaine médical, avec la mise en place de VLAN, d'access-lists, de pare-feux, de Radius, de SSL/TLS et de VPN pour protéger les données sensibles et garantir l'intégrité du réseau.

La redondance et la continuité de services sont assurées grâce à des mécanismes tels que VRRP/HSRP, le Server Load Balancing (SLB) et Spanning-Tree, ce qui garantit un fonctionnement ininterrompu du réseau, même en cas de panne matérielle ou de défaillance.

Enfin, des dispositifs de sauvegarde et des procédures de test de redondance ont été mis en place pour prévenir la perte de données et valider le bon fonctionnement du réseau.

En conclusion, nous avons pu démontrer la faisabilité de la mise en place de notre infrastructure à travers notre preuve de concept (**P**roof **O**f **C**oncept).