

Projet d'interconnexion de systèmes autonomes

Realisé par:

Mohamed El Yessefi
Yahya Younes
Christophe Chen
Abderrahmane Eddahbi
Othman Ait Ouakrim
Abderrahim Abdellaoui
Ibrahim Al Gharras
Nabil Ait Alaya

Groupe 90'

Département Sciences du Numérique

2022-2023

Contents

1	Introduction	3
2	Principaux choix	3
2.1	Architecture du réseau :	3
2.2	Plateforme : Docker	4
2.3	Distribution des tâches	4
3	Observations	5
3.1	Compatibilité de l'architecture avec les contraintes imposées . . .	5
3.2	Premiers pings	6
3.3	Déroulement et tests de fonctionnement	6
3.3.1	Serveurs FTP et Web : Abderrahmane Eddahbi et Abder- rahim Abdellaoui	7
3.3.2	DHCP et automatisation des scripts : Christophe Chen . .	9
3.3.3	Interconnexion et serveur FTP : Othman Ait Ouakrim . .	9
3.3.4	Interconnexion et adressage : Ibrahim Al Gharras	9
3.3.5	Routage OSPF et Interconnexion : Nabil Ait Alaya	10
3.3.6	Premiers Pings, command.sh et Service DNS : Yahya Younes	10
3.3.7	Interconnexion, command.sh et conception de l'architecture :Mohammed El Yessefi	11
4	Difficulté principale	11
5	Conclusion	12

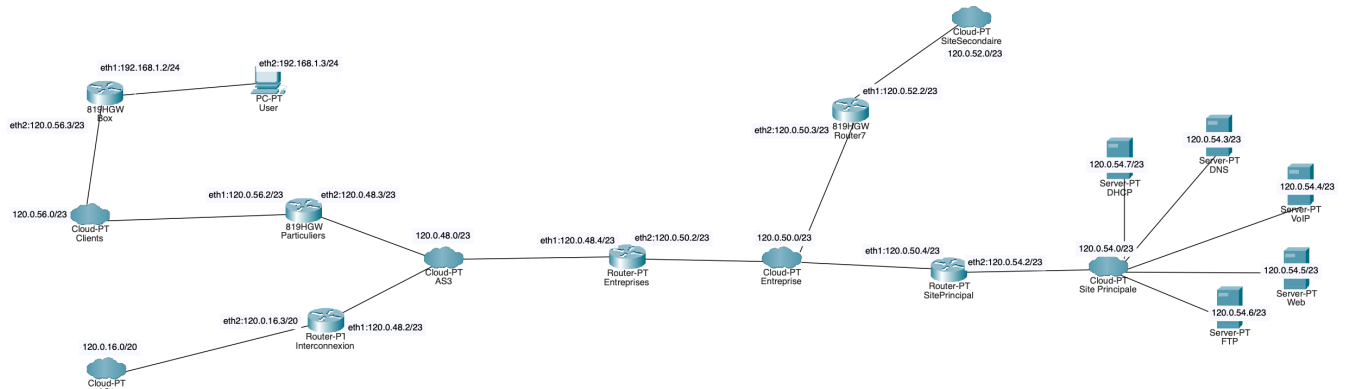
1 Introduction

Le projet en question vise à établir un réseau d'entreprise multi-sites en adoptant une approche holistique qui consiste à concevoir des protocoles de communication efficaces, à comprendre leur fonctionnement et à les observer en situation réelle. L'objectif ultime est de permettre la connexion de ce réseau à d'autres systèmes autonomes, pour assurer une intégration fluide et une communication optimale entre les différents sites de l'entreprise. Pour atteindre cet objectif, il est nécessaire de disposer d'une expertise solide en matière de réseaux informatiques, ainsi que d'une avancée significative dans les recherches sur les technologies et les supports techniques IP les plus adaptés à utiliser.

2 Principaux choix

2.1 Architecture du réseau :

Avant de se pencher sur des solutions potentielles, il était d'abord nécessaire de définir une architecture appropriée pour un système autonome quelconque. Bien que plusieurs options aient été considérées au départ, certaines étant abstraites et incomplètes, nous avons finalement opté pour le modèle suivant :



Voici les composantes principales mises en place :

AS : le réseau central de l'architecture, qui a une politique de routage interne cohérente.

Entreprise : un réseau d'entreprise multi-sites, comprenant un site principal et un site secondaire selon le cadre habituel d'un réseau d'entreprise aléatoire.

Particulier : le réseau du client qui utilise les services de l'entreprise.

Serveur DHCP : protocole de réseau qui assure la configuration automatique des paramètres IP d'une station ou d'une machine, notamment en lui attribuant automatiquement une adresse IP et un masque de sous-réseau.

Serveur DNS : service informatique distribué utilisé pour traduire les noms de domaine Internet en adresses IP ou autres enregistrements.

Serveur FTP : permet de transférer des fichiers sur Internet ou sur un réseau informatique local. Toute personne ayant l'autorisation peut télécharger et envoyer des fichiers sur un ordinateur distant exécutant un tel serveur.

Serveur Web : répond aux requêtes du World Wide Web sur un réseau public ou privé, principalement en utilisant le protocole HTTP.

Machine client Particulier : pour tester l'adaptabilité du tunnel interne.

Routeurs : selon le service et le réseau concerné.

Pare-feu : pour faire respecter la politique de sécurité du réseau, définissant les types de communications autorisés sur ce réseau informatique. Il surveille et contrôle les applications et les flux de données.

2.2 Plateforme : Docker

L'une des étapes clés du projet a été de choisir un outil pour empaqueter la base de données complète. Cela impliquait de suivre des formations spécifiques pour chaque logiciel ou application, en fonction des dépendances des systèmes et des particularités des conteneurs isolés des serveurs. Le déploiement d'unités normalisées nécessitait également de rechercher des configurations spécifiques pour un environnement d'exécution adapté. Cependant, la méthode standard qui pouvait gérer la plupart des applications opérationnelles, de la même manière qu'une machine virtuelle, ne pouvait être garantie que par l'utilisation de la plateforme Docker. Cela permettait des économies considérables et des migrations intersystèmes faciles.

2.3 Distribution des tâches

Après avoir déterminé l'outil principal présenté dans la section précédente, il était important de partager efficacement les rôles, que ce soit pour la mise en oeuvre globale ou pour une interprétation spécifique de la visualisation interne. Des tâches individuelles devaient être attribuées, sans pour autant créer une isolation, car plusieurs composantes restent plus ou moins liées pour le bon fonctionnement général du réseau conçu.

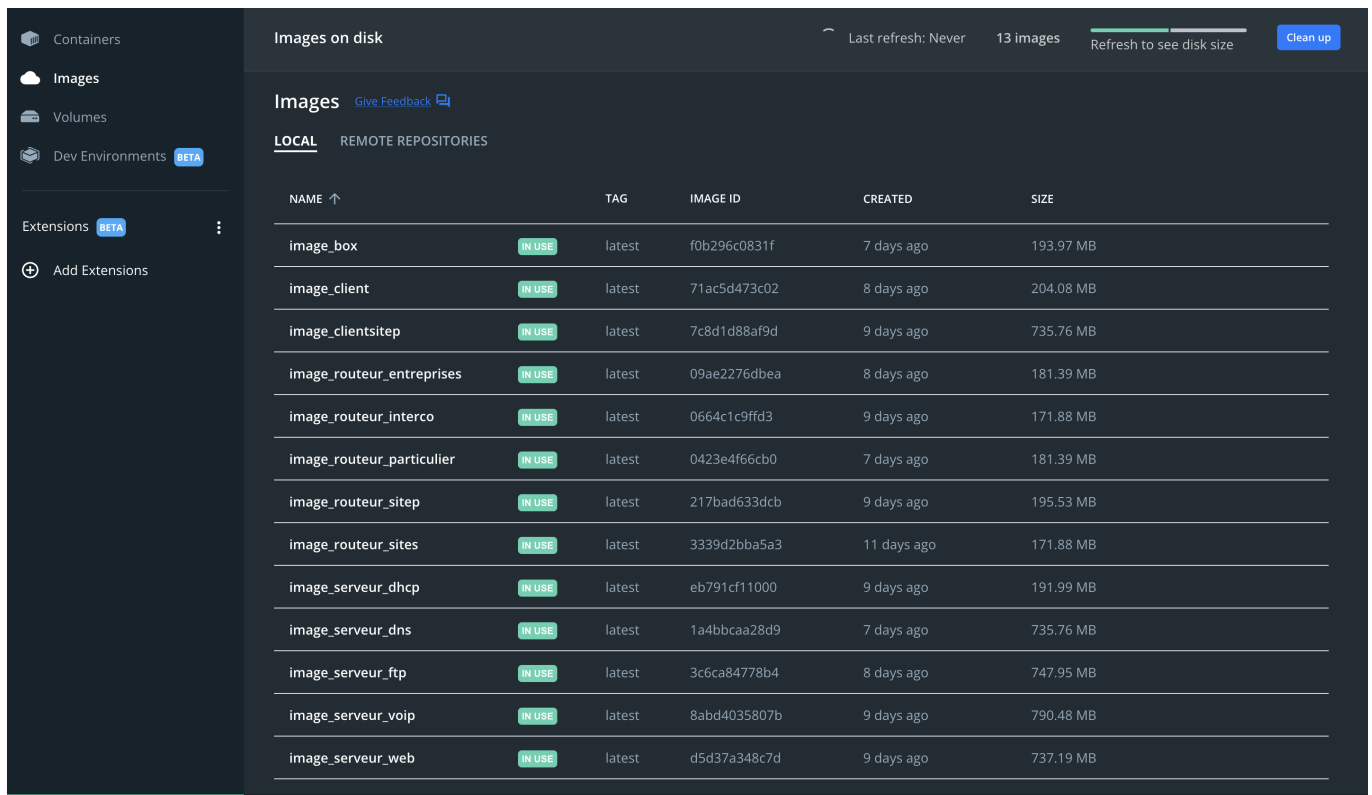
Voici comment le travail a été partagé au sein du groupe :

- Mohammed El Yessefi** : Interconnexion + command.sh + conception de l'architecture.
- Nabil Ait Alaya** : routage OSPF + interconnexion.
- Christophe Chen** : Service DHCP + Automatisation des scripts.
- Yahya Younes** : Premiers Pings + command.sh + Service DNS
- Ibrahim Al Gharras** : Conception de l'architecture + Adressage
- Othman Ait Ouakrim** : Interconnexion + serveur FTP
- Abderrahmane Eddahbi et Abderrahim Abdellaoui** : Serveur FTP + Serveur Web

3 Observations

3.1 Compatibilité de l'architecture avec les contraintes imposées

Avant de procéder aux tests de fonctionnement du système, il était crucial d'avoir des **images Docker** appropriées, qui permettent de regrouper des applications et des environnements de serveur pré-configurés sous forme d'ensemble d'instructions pour créer un conteneur pouvant s'exécuter sur la plateforme. Les recherches menées sur le sujet nous ont permis de déduire les modèles affichés dans la figure ci-contre :



NAME ↑	TAG	IMAGE ID	CREATED	SIZE
image_box	latest	f0b296c0831f	7 days ago	193.97 MB
image_client	latest	71ac5d473c02	8 days ago	204.08 MB
image_clientsitep	latest	7c8d1d88af9d	9 days ago	735.76 MB
image_routeur_entreprises	latest	09ae2276dbea	8 days ago	181.39 MB
image_routeur_interco	latest	0664c1c9ffd3	9 days ago	171.88 MB
image_routeur_particulier	latest	0423e4f66cb0	7 days ago	181.39 MB
image_routeur_sitep	latest	217bad633dcb	9 days ago	195.53 MB
image_routeur_sites	latest	3339d2bba5a3	11 days ago	171.88 MB
image_serveur_dhcp	latest	eb791cf11000	9 days ago	191.99 MB
image_serveur_dns	latest	1a4bbcaa28d9	7 days ago	735.76 MB
image_serveur_ftp	latest	3c6ca84778b4	8 days ago	747.95 MB
image_serveur_voip	latest	8abd4035807b	9 days ago	790.48 MB
image_serveur_web	latest	d5d37a348c7d	9 days ago	737.19 MB

Après cela, il était nécessaire de vérifier que les images étaient correctement installées et utilisées de manière appropriée. Lors d'un premier test, les résultats attendus étaient

obtenus (environnement approprié et fenêtre de contrôle bash visible et utilisable) :

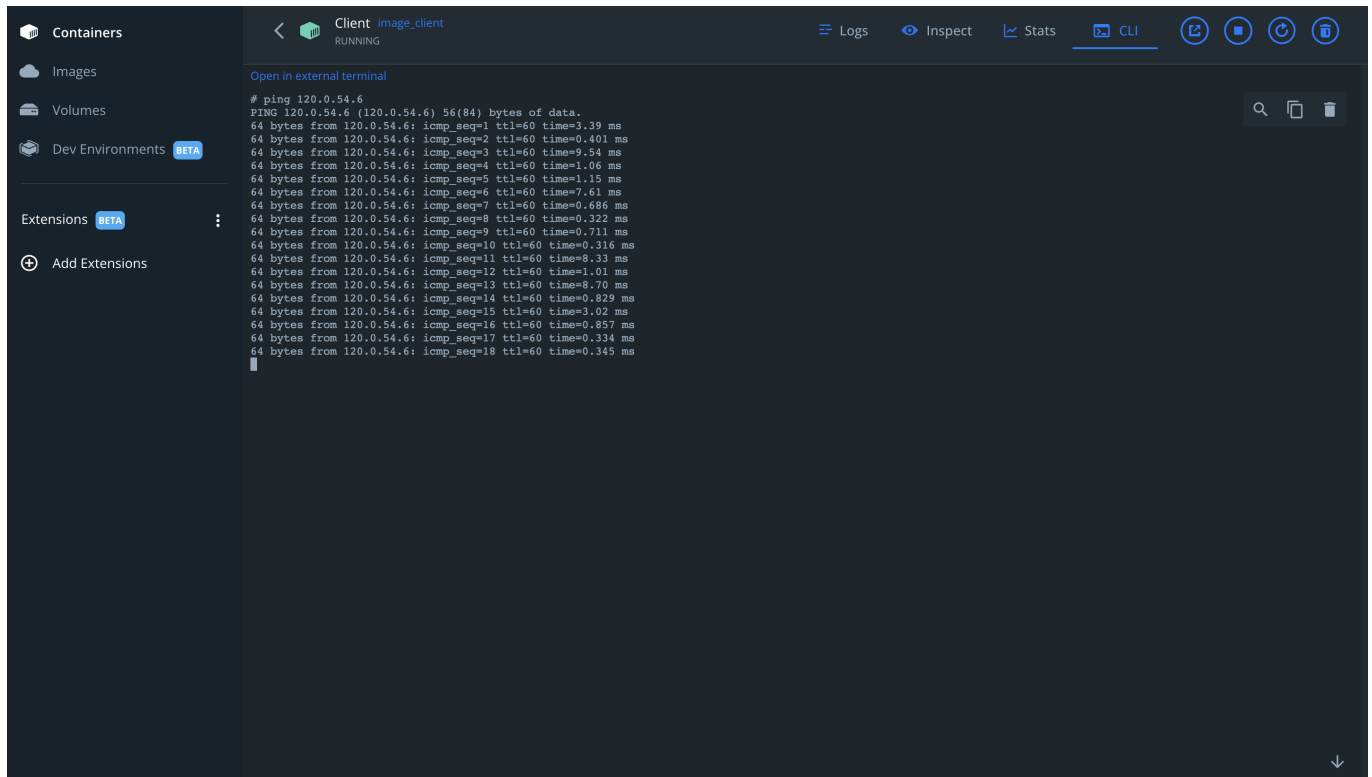
	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	Routeur_Entreprises b1724887a742	image_routeur_entreprises:latest	Running	49462	2 minutes ago	
<input type="checkbox"/>	Routeur_SiteP 863f515068c6	image_routeur_sitep:latest	Running	49463	2 minutes ago	
<input type="checkbox"/>	Routeur_SiteS f139f6d07eb3	image_routeur_sites:latest	Running	49464	2 minutes ago	
<input type="checkbox"/>	BOX ffa66fa2f737	image_box:latest	Running	49465	2 minutes ago	
<input type="checkbox"/>	Routeur_Interco e528bc1b9c8a	image_routeur_interco:latest	Running	49466	2 minutes ago	
<input type="checkbox"/>	Routeur_Particulier 9d8c12efae10	image_routeur_particulier:latest	Running	49467	2 minutes ago	
<input type="checkbox"/>	Serveur_WEB 02784473a9ab	image_serveur_web:latest	Running	8080	2 minutes ago	
<input type="checkbox"/>	Serveur_FTP f2c2c9f067a3	image_serveur_ftp:latest	Running	49468	2 minutes ago	
<input type="checkbox"/>	Serveur_VOIP d8a6c7e131d6	image_serveur_voip:latest	Running	5060	2 minutes ago	
<input type="checkbox"/>	Serveur_DHCP ecacda4f20a8	image_serveur_dhcp:latest	Running	5061	2 minutes ago	
<input type="checkbox"/>	Serveur_DNS b039a1b4cb8a	image_serveur_dns:latest	Running	49469	2 minutes ago	
<input type="checkbox"/>	Client					

3.2 Premiers pings

Après avoir mis en place les équipements nécessaires, et avant de poursuivre dans les détails, un premier test de transfert de ping était utile pour évaluer la qualité de l'installation. Ce test s'est avéré être un succès:

3.3 Déroulement et tests de fonctionnement

Dans cette section, nous allons présenter les principaux progrès du projet en partageant directement l'expérience de chaque membre, accompagnée des explications nécessaires sur les raisons et les méthodes des résultats obtenus. Il est à noter que chaque membre s'est



The screenshot shows the Docker Desktop application. On the left sidebar, there are icons for Containers, Images, Volumes, Dev Environments (with a BETA badge), and Extensions (also with a BETA badge). The main area displays the 'Client image_client' container, which is in a 'RUNNING' state. Below the container name, there is a link to 'Open in external terminal'. The terminal window shows the output of a 'ping' command to the IP address 120.0.54.6. The output indicates that the ping is successful, with 56(84) bytes of data received and various response times (e.g., 3.39 ms, 0.401 ms, etc.).

```
# ping 120.0.54.6
PING 120.0.54.6 (120.0.54.6) 56(84) bytes of data.
64 bytes from 120.0.54.6: icmp_seq=1 ttl=60 time=3.39 ms
64 bytes from 120.0.54.6: icmp_seq=2 ttl=60 time=0.401 ms
64 bytes from 120.0.54.6: icmp_seq=3 ttl=60 time=9.54 ms
64 bytes from 120.0.54.6: icmp_seq=4 ttl=60 time=1.06 ms
64 bytes from 120.0.54.6: icmp_seq=5 ttl=60 time=1.15 ms
64 bytes from 120.0.54.6: icmp_seq=6 ttl=60 time=7.61 ms
64 bytes from 120.0.54.6: icmp_seq=7 ttl=60 time=0.686 ms
64 bytes from 120.0.54.6: icmp_seq=8 ttl=60 time=0.322 ms
64 bytes from 120.0.54.6: icmp_seq=9 ttl=60 time=0.711 ms
64 bytes from 120.0.54.6: icmp_seq=10 ttl=60 time=0.316 ms
64 bytes from 120.0.54.6: icmp_seq=11 ttl=60 time=8.33 ms
64 bytes from 120.0.54.6: icmp_seq=12 ttl=60 time=1.01 ms
64 bytes from 120.0.54.6: icmp_seq=13 ttl=60 time=8.70 ms
64 bytes from 120.0.54.6: icmp_seq=14 ttl=60 time=0.829 ms
64 bytes from 120.0.54.6: icmp_seq=15 ttl=60 time=3.02 ms
64 bytes from 120.0.54.6: icmp_seq=16 ttl=60 time=0.857 ms
64 bytes from 120.0.54.6: icmp_seq=17 ttl=60 time=0.334 ms
64 bytes from 120.0.54.6: icmp_seq=18 ttl=60 time=0.345 ms
```

chargé des fichiers.sh propres à sa partie en utilisant des fichiers Docker, avec l'ajout indispensable de fichiers de configuration dans les conteneurs lancés.

3.3.1 Serveurs FTP et Web : Abderrahmane Eddahbi et Abderahim Abdellaoui

Dans ce projet, Nous étions été chargés de configurer deux serveurs différents : un serveur FTP et un serveur web. Pour configurer le serveur FTP, nous avons utilisé le service proftpd, un serveur FTP populaire pour sa flexibilité et facilité d'utilisation, permettant de gérer les utilisateurs, les droits d'accès et les quotas. Pour configurer le serveur Web, nous avons utilisé Apache2, l'un des serveurs web les plus utilisés dans le monde, permettant de gérer les sites web, les configurations de sécurité et performance, et les redirections.

Afin de vérifier le bon fonctionnement de ces deux serveurs, nous avons utilisé Docker pour créer une architecture de petit réseau comprenant un client, un routeur et les deux serveurs (FTP et web) en spécifiant un nom d'utilisateur et un mot de passe. Nous avons alors utilisé des commandes FTP pour nous connecter au serveur FTP et y déposer un fichier, ainsi que des commandes pour lister les fichiers, changer de répertoire et supprimer des fichiers. Pour le serveur web, nous avons activé le serveur et utilisé la commande wget pour tester son fonctionnement et vérifier que les sites web sont accessibles, les pages web s'affichent correctement et que les liens et images fonctionnent.

Enfin, nous avons intégré ces deux serveurs dans l'architecture du grand réseau pour les utiliser efficacement. Nous avons vérifié que les serveurs sont accessibles depuis l'extérieur, que les ports sont ouverts et que les connexions sont sécurisées.

```
# wget 120.0.54.5
--2023-01-25 21:07:26-- http://120.0.54.5/
Connecting to 120.0.54.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html          100%[=====>] 10.45K  --.-KB/s  in 0s

2023-01-25 21:07:26 (106 MB/s) - 'index.html' saved [10701/10701]

# ls
bin boot dev etc home index.html lib media mnt opt proc root run sbin srv sys tmp usr var
#
```

```
converted 'ftp://120.0.54.6/msg.txt' (ANSI_X3.4-1968) -> 'ftp://120.0.54.6/msg.txt' (UTF-8)
--2022-01-22 17:04:52-- ftp://120.0.54.6/msg.txt
=> 'msg.txt'
Connecting to 120.0.54.6:21... connected.
Logging in as groupe2 ... Logged in!
==> SYST ... done. ==> PWD ... done.
==> TYPE I ... done. ==> CWD not needed.
==> SIZE msg.txt ... 17
==> PASV ... done. ==> RETR msg.txt ... done.
Length: 17 (unauthoritative)

msg.txt            100%[=====>] 17 --.-KB/s  in 0s

2022-01-22 17:04:52 (1.76 MB/s) - 'msg.txt' saved [17]
```


3.3.2 DHCP et automatisation des scripts : Christophe Chen

Je me suis renseigné sur internet les moyens de créer un dockerfile d'un serveur dhcp, j'ai trouvé un package intéressant. Lorsque les adressages sont bien configurés, j'ai lancé dhcpd (pour activer le serveur), mais ça ne fonctionnait pas, puis après lecture et réflexion sur les messages d'erreur, j'ai su qu'il fallait créer un script vide dhcpd.conf dans var/lib/dhcp/. Le serveur s'est bien lancé après modification mais on ne pouvait pas encore l'utiliser car il manquait un dhclient, après quelques détournements j'ai fini par réussir à créer le serveur dhcp.

J'ai écrit les commandes permettant de supprimer tout les images, ainsi que tout les conteneurs et réseau après chaque lancement du fichier. J'ai écrit des instructions (et configurations) pour pouvoir lancer le script de configuration de chaque conteneur docker quand on veut (sans forcément lors de la construction de l'image) C'était bien fonctionnelle.

J'ai également participé à la distribution des plages d'adresses aux machines. J'ai fourni les instructions pour faire de l'adressage dynamique des conteneurs dès lors que celle-ci se connecte au réseau. (Pour le routage statique on précisera le device plutôt que le gateway).

3.3.3 Interconnexion et serveur FTP : Othman Ait Ouakrim

Dans cette phase du projet, j'ai rencontré des difficultés liées à la sélection des images appropriées pour implémenter un service FTP en utilisant Proftpd. Comme les images disponibles étaient inadaptées et la configuration difficile à mettre en œuvre, nous avons choisi d'utiliser une version alternative, Vsftpd, qui remplit le même rôle. Nous avons également utilisé des fichiers de configuration spécifiques et testé l'implémentation en créant un répertoire de test, où nous avons téléchargé un message depuis le client. Le test a été réalisé via le terminal car la manipulation des interfaces graphiques Docker était difficile.

3.3.4 Interconnexion et adressage : Ibrahim Al Gharras

Dans le cadre de ce projet de création d'un réseau d'entreprise multi-sites, j'ai participé à plusieurs tâches clés. Tout d'abord, j'ai contribué à la création de la topologie du réseau. Cela a impliqué la définition des différents éléments qui composent le réseau, tels que les routeurs, les commutateurs et les serveurs, ainsi que la manière dont ils sont connectés entre eux.

Ensuite, j'ai travaillé sur l'adressage de notre réseau en respectant les consignes du sujet. Cela a impliqué la définition des plages d'adresses IP qui seront utilisées pour chaque segment du réseau.

J'ai également participé à l'interconnexion entre deux AS. Cela a permis de connecter efficacement les différents sites entre eux, assurant une communication fluide entre les employés de différents emplacements. En plus de mes contributions mentionnées précédemment, j'ai également rencontré des difficultés avec le routage OSPF. J'ai importé le fichier Quagga depuis le sujet de la première année, mais j'ai rencontré des problèmes de compatibilité avec l'image Debian Jessie-slim que nous utilisions sur Docker. Malgré mes efforts

pour résoudre ces problèmes, je n'ai pas réussi à mettre en place le routage OSPF sur Docker. Cela a été un défi pour moi, mais j'ai appris beaucoup sur la configuration et la mise en place de routage OSPF.

En somme, j'ai contribué à plusieurs étapes clés de ce projet de création d'un réseau d'entreprise multi-sites, notamment la création de la topologie du réseau, l'adressage de notre réseau en respectant les consignes du sujet, et la réalisation de l'interconnexion avec un autre AS séparé en aidant mon groupe connectant d'abord nos conteneurs Docker à la carte réseau de la machine hôte et en interconnectant les deux machines.

3.3.5 Routage OSPF et Interconnexion : Nabil Ait Alaya

J'ai été affecté à la partie routage dynamique ospf, un protocole de routage interne (Interior Gateway Protocol) à état de liens. Malheureusement, l'implémentation de cette fonctionnalité sous Docker s'est avérée difficile, car la configuration d'OSPF est plus délicate en un conteneur Docker, qu'en un routeur physique. Ainsi, j'ai changé de tâche, et je me suis affecté à la tâche d'interconnexion entre deux AS : le routage du trafic d'un conteneur Docker vers la carte réseau physique de la machine. J'ai aussi participé à l'adressage et la mise en place de l'architecture du réseau.

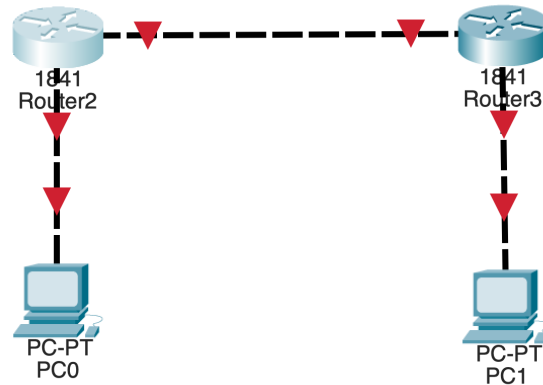
3.3.6 Premiers Pings, command.sh et Service DNS : Yahya Younes

Dans le cadre de mon rôle au sein de l'équipe, j'ai pris l'initiative de créer un projet minimal (voir figure ci dessous) pour permettre à mes collègues de tester les nouveautés sur une petite échelle, ce qui a facilité la compréhension et la rapidité des tests. Cette méthode de travail a considérablement économisé du temps lors de la création de conteneurs, de tests, de configurations de dockerfiles et d'analyses, en permettant de détecter facilement les erreurs et de les corriger rapidement. J'ai ainsi réussi à faire passer le premier ping dans le grand projet, ce qui représente une étape importante. Ensuite, j'ai été en charge de la configuration du service DNS dans le projet. J'ai utilisé BIND9 comme serveur de noms open-source et simple à configurer, en utilisant le fichier `/etc/named.conf` pour décrire la résolution de nom dans les environnements Linux. J'ai configuré les zones dans le fichier `/etc/bind/zones/db.groupe3.com` et j'ai testé l'accès de l'hôte 2 depuis l'hôte 1 en utilisant le DNS.

En outre, j'ai contribué à la réalisation du fichier `command.sh` qui automatise les commandes nécessaires pour lancer le projet, ce qui nous a facilité la préparation et la présentation du projet.

Enfin, j'ai contribué à l'interconnexion en trouvant une méthode pour envoyer des ping depuis un conteneur client sur Docker jusqu'à la carte réseau d'un autre ordinateur, où nous avons pu visualiser sur Wireshark le trafic et vérifier le bon fonctionnement du NAT.

Figure du projet minimal et simpliste pour tester chaque nouvelle implémentation



3.3.7 Interconnexion, command.sh et conception de l'architecture :Mohammed El Yessefi

Partie gestion : En tant que leader du groupe 90', j'étais chargé de superviser l'avancement de l'équipe chaque semaine, en attribuant les tâches et en organisant des réunions entre les différents membres. En outre, j'ai joué un rôle clé pour assurer que les objectifs et les délais de l'équipe soient respectés, en coordonnant et en communiquant efficacement avec tous les membres de l'équipe, en fournissant des conseils et un soutien selon les besoins. De plus, j'étais chargé de détecter et de résoudre les problèmes ou les obstacles qui se sont présentés au cours du projet, afin de maintenir l'élan et la performance globale de l'équipe. Grâce à mon leadership, j'ai visé à favoriser un environnement de travail positif et productif, où tous les membres de l'équipe pouvaient s'épanouir et livrer leur meilleur travail.

Partie technique : En tant que membre de l'équipe de projet, j'ai été chargé de contribuer à la conception de l'architecture de notre système d'automatisation (AS). Mon premier objectif était de s'assurer que l'adressage réseau était correctement mis en place pour éviter tout problème de confusion entre les différents réseaux. Pour cela, j'ai proposé d'utiliser la technique VLSM pour attribuer les adresses IP. En plus de cela, j'ai également participé à la rédaction du fichier command.sh, en collaboration avec mes collègues, dans lequel j'ai construit les images, les réseaux et les conteneurs nécessaires pour notre système d'automatisation en utilisant les paramètres appropriés pour assurer un bon fonctionnement de notre système. Ajoutons à cela que chaque membre de l'équipe m'envoyait le travail qu'il a fait dans une petite architecture alors c'était à moi d'intégrer leurs travaux dans notre projet.

4 Difficulté principale

Au cours de ce projet, nous avons rencontré des difficultés liées à la configuration du serveur VoIP. Cela a impliqué une configuration par fichiers, dans laquelle nous avons dû

déposer les fichiers de configuration correctement et ajouter deux utilisateurs pour finaliser l'opération. Pour tester cette configuration, nous avons dû utiliser l'interface graphique d'un logiciel appelé EKIGA, qui utilise les protocoles de communication standards et ouverts H.323 et SIP3. Cela rend le logiciel compatible et interopérable avec d'autres logiciels et appareils qui utilisent les mêmes protocoles. Malheureusement, malgré notre engagement à poursuivre cette initiative ambitieuse, nous avons été confrontés à des contraintes de temps et techniques qui ont finalement rendu difficile la réalisation de cette partie. Ces contraintes ont eu un impact significatif sur notre capacité à mettre en œuvre cette initiative, qui était certes ambitieuse mais également complexe.

5 Conclusion

Ce projet a été extrêmement bénéfique pour notre développement professionnel en tant qu'ingénieurs en réseaux informatiques. Il nous a donné l'occasion de mettre en pratique les concepts d'architectures de réseaux autonomes et de comprendre les enjeux de notre futur métier. Il a mis en évidence l'importance de prendre des initiatives, de respecter les délais et de travailler en équipe pour atteindre les objectifs.

En outre, ce projet nous a permis d'appliquer nos connaissances en matière de réseaux et de protocoles à un domaine pratique qui est devenu de plus en plus important en raison de l'augmentation des besoins des entreprises en outils de conteneurisation. Les principaux défis auxquels nous avons été confrontés ont été liés à la collecte de données, qui étaient souvent incompatibles tant en termes d'images que de structure globale. Cela nous a fait prendre conscience de la difficulté de extrapoler les données disponibles pour obtenir des résultats cohérents sur l'ensemble du domaine étudié, ce qui sera probablement un défi dans notre futur métier. Cependant, si une telle étude devait être menée à terme à plus ou moins long terme par une entreprise de consulting, Docker pourrait figurer parmi les programmes adaptatifs proposés.