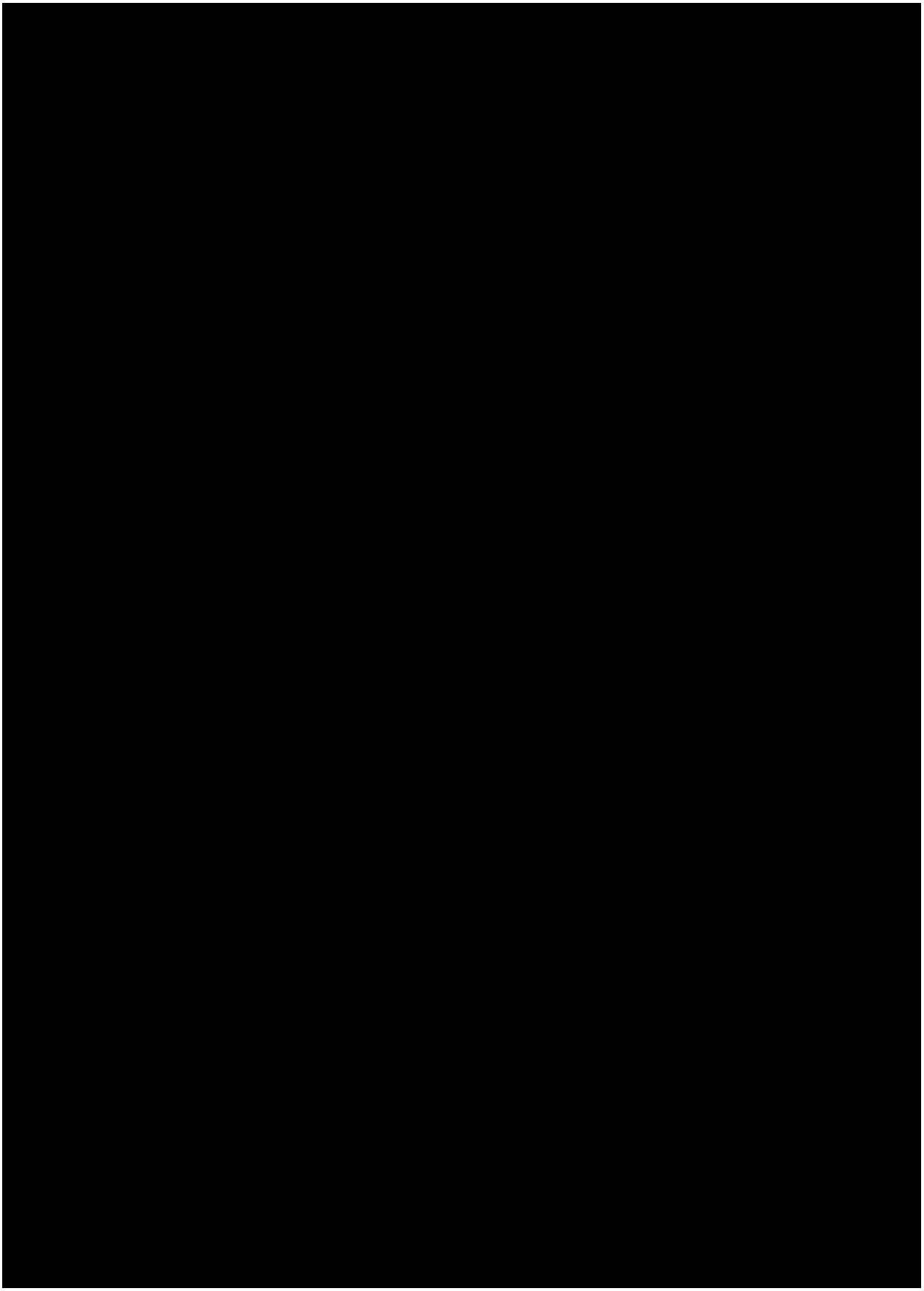
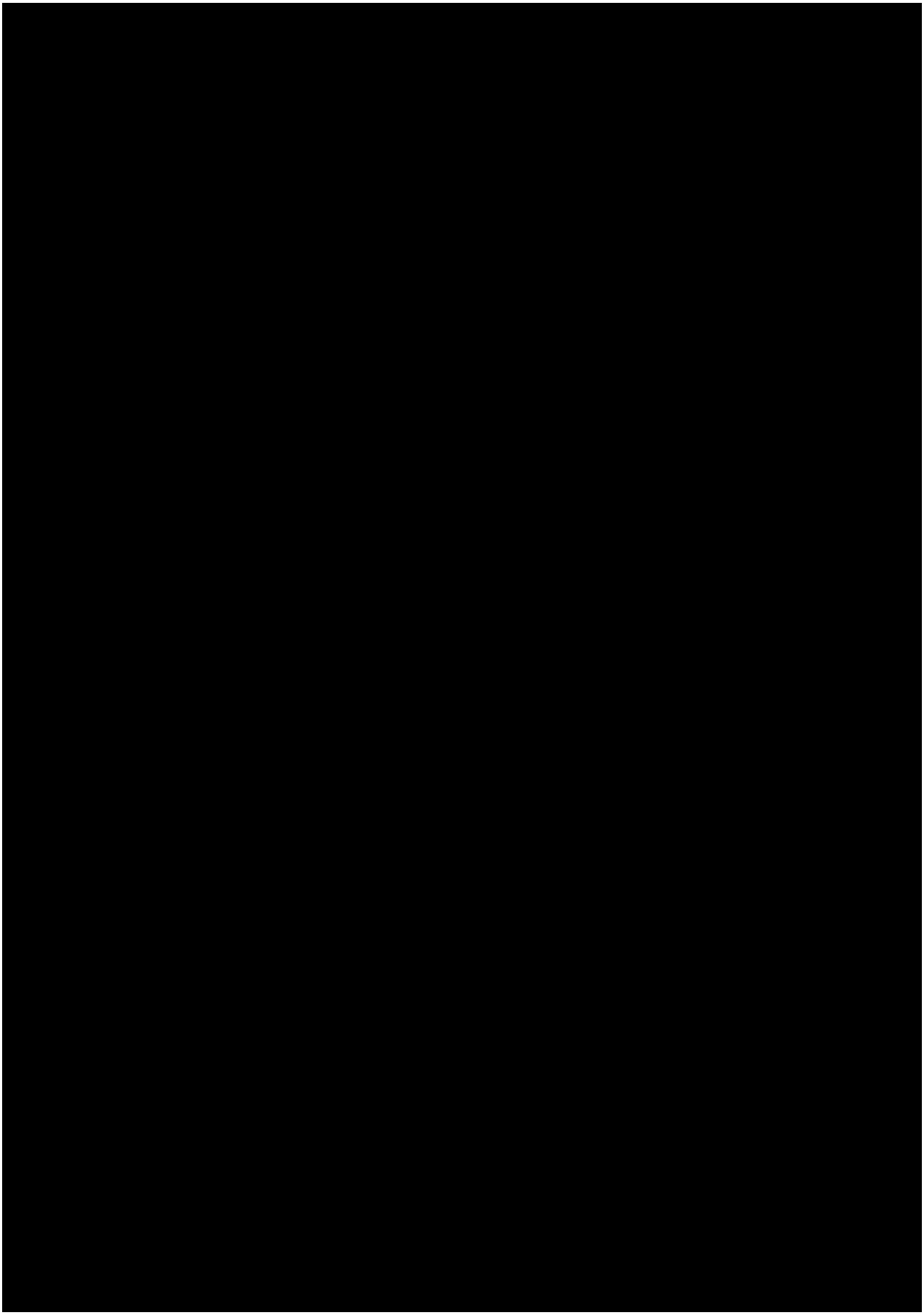
DOSSIER DE SYNTHESE E5

Matt MOREAU
Alternant Service Infrastructure, Pôle AdminLAN
Durée : 2 années

Tuteur en entreprise	Aurélien LALLEMAND
Entreprise d'accueil	Alphalink
Responsable d'Alternance	Olivier CHIRON
Formation	BTS SIO (Services Informatiques aux Organisations) Option : SISR (Solutions d'Infrastructure, Systèmes et Réseaux)
Établissement	Fab'Academy, 9 Rue de l'Halbrane, 44340 Bouguenais

2024/2026





BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2026**

Tableau de synthèse des réalisations professionnelles

NOM et prénom : MOREAU Matt	**N° candidat : 02542583108**
Centre de formation : FAB ACADEMY	**Option :** ☑ **SISR** ☐ **SLAM**
Adresse URL du portfolio : https://matt.moreau.formation-esiac.fr/	

Compétences mises en œuvre ***Réalisations professionnelles*** ***(intitulé et liste des documents et productions associés)***	**Période (sous la forme du JJ/MM/AA au JJ/MM/AA)**	**Gérer le patrimoine informatique** •Recenser et identifier les ressources numériques •Exploiter des référentiels, normes et standards adaptés au patrimoine informatique •Mettre en place et vérifier les conditions de la continuité d'un service informatique •Identifier le risque et mettre en place les procédures correctives	**Répondre aux incidents et aux demandes d'assistance et d'évolution** •Collecter, suivre et orienter des demandes •Traiter des demandes concernant les services, applicatifs •Traiter des incidents concernant les applications	**Développer la présence en ligne de l'organisation** •Participer à la valorisation de l'image de l'organisation sur les médias numériques en •Référencer les services en ligne de l'organisation et mesurer leur visibilité. •Participer à l'évolution d'un site Web exploitant les données de l'organisation.	**Travailler en mode projet** •Analyser les enjeux et les modalités d'organisation d'un projet •Évaluer les indicateurs de suivi d'un projet et analyser les écarts	**Mettre à disposition des utilisateurs un service informatique** •Réaliser les tests d'intégration et de déploiement d'un service •Déployer un service •Accompagner les utilisateurs dans la mise en place d'un service	**Organiser son développement professionnel** •Mettre en place son environnement d'apprentissage/autoformation •Mettre en œuvre des outils de veille stratégique de veille informationnelle •Développer son réseau professionnel •Développer son projet professionnel
Réalisation en cours de formation							
Mise en place d'un hyperviseur de type 1		x	x			x	
Mise en place d'un cluster RDS avec AD, partages réseau et un serveur de fichier		x	x			x	
Installation et configuration d'un serveur GLPI avec déploiement d'agents et intégration Active Directory		x	x			x	
Déploiement d'un serveur Active Directory (AD) et Windows Server Update Services (WSUS)		x	x			x	
MAQUETTE OASIS → Mise en place d'un serveur AD + DHCP + DNS + GLPI + NEXTCLOUD		x	x			x	
MAQUETTE OASIS → Mise en place de Vlans et hyperviseur de type 1 (switch /RODC /Relais DHCP /cluster serveur web WORDPRESS HA)		x	x			x	
Mise en place du VLSM, OSPF, DHCP centralisé ,DMZ avec ACL, syslog et SNMP sur Cisco Packet tracer		x	x			x	
Création d'un site web intranet à l'aide de Wordpress				x	x		x
Mise en place d'un compte Linkedin							x
Mise en place veille technologique							x
Création d'un site Web pour entreprise pédagogique				x			
Réalisations en milieu professionnel en cours de première année							
Mise en place d'une solution de gestion des logiciels		x	x		x	x	
Installation d'un poste		x	x			x	
Mise en place d'un filtrage Mac sur Wifi		x	x			x	
Renouvellement du parc informatique Mac Intel > Mac Puce M							
Gestion de comptes Extranet						x	
Réalisations en milieu professionnel en cours de seconde année							
Mise en place d'un serveur de sauvegarde		x	x		x	x	
Documentation Utilisateur pour gestion des adresses Mac		x					
Création de comptes Azure AD						x	
Mise en place d'une application de virtualisation windows sous Mac		x	x			x	
Mise en place de switch		x	x				

Table des matières

Remerciements

Tout d'abord, je tiens à remercier l'ensemble des personnes qui travaillent au sein d'Alphalink pour m'avoir consacré de leur temps lors des problèmes informatiques qu'elles ont rencontrés. Je souhaite également remercier l'ensemble du Service Systèmes d'Information pour leur accueil et pour les différentes connaissances qu'ils ont pu m'apporter. Enfin, je tiens à remercier tout particulièrement M. LALLEMAND, mon tuteur, pour m'avoir aidé, m'avoir transmis ses connaissances et avoir pris de son temps tout au long de cette alternance pour me former.

Introduction

Je m'appelle Matt MOREAU, je suis actuellement étudiant en BTS SIO (Services Informatiques aux Organisations) à la Fab'Academy de Bouguenais.

J'ai auparavant réalisé un BAC professionnel SN (Systèmes Numériques) option RISC (Réseaux Informatiques et Systèmes Communicants) en 3 ans, que j'ai obtenu avec mention très bien et les félicitations du jury. Ensuite, j'ai commencé une formation en alternance sur 2 ans, le BTS, afin de découvrir le fonctionnement des entreprises et d'approfondir mes connaissances avec des professionnels dans le domaine de l'informatique.

Mes objectifs pour cette alternance étaient de pratiquer et d'améliorer l'ensemble de mes compétences, que ce soient les soft skills ou les compétences techniques. Je voulais également m'exercer au sein d'un service informatique, voir plus précisément le métier d'administrateur réseau et me conforter dans mes choix de futures études supérieures.

Pour terminer, j'aimerais, à la suite de mon BTS, réaliser la formation ASR (Administrateur Systèmes et Réseaux) à l'ENI afin de continuer mes études et d'évoluer en compétences.

Tout d'abord, je présenterai l'entreprise au sein de laquelle j'ai pu réaliser mon alternance, avec les situations géographique, juridique et économique, ainsi que la présentation de son secteur d'activité et de l'organigramme de l'entreprise. Ensuite, je détaillerai les différentes activités que j'ai réalisées au cours de cette alternance, puis je terminerai par le bilan de ces deux années au sein de l'entreprise.

Présentation de l'entreprise

a) Localisation

Lors de cette alternance, j'ai travaillé au siège social d'Alphalink, qui se situe à Pornic, près du Val Saint-Martin, à côté du lycée du Pays de Retz. Le siège social se trouve exactement rue Jules Ferry, au Boismain, à Pornic.

Nous avons d'autres sites que le siège social de Pornic : nous avons Lyon, Avignon et Courbevoie, où se situe notre propre datacenter de 1500 m². Enfin, nous avons également 21 points de présence (points de connexion pour les connexions internet) dans toute la France.

b) Activités + Fillières

Alphalink qui est le leader sur le marché de la télécommunication indirecte, propose différentes offres.

Tout d'abord, nous proposons des offres de téléphonie mobile et fixe. Nous sommes le 1er opérateur sur les centres d'appel et le 2e opérateur en portabilité.

En plus de ces offres, nous avons établi un partenariat avec Cisco pour la plateforme Webex, qui permet de gérer la messagerie et la téléphonie comme la plateforme Teams de Microsoft.

Ensuite, nous avons des offres de fibre optique avec notre propre réseau fibre et Eurofiber. L'entreprise propose un service standard et premium avec des débits allant de 10 Mb/s à 1 Gb/s.

Après, nous avons des offres d'hébergement pour les entreprises dans notre propre Datacenter.

Enfin, nous avons une offre de cybersécurité, appelée Cyber 360, qui a été lancée en 2024. Cette offre comprend plusieurs services. Tout d'abord, des services de protection XDR. L'XDR est une solution de sécurité qui permet de faire de la détection sur les mails, réseaux, serveurs, applications cloud et Endpoint. Ensuite, de la mise en place de Firewall NextGEN pour protéger les cœurs de réseau. Enfin, un service de scan de vulnérabilité. L'ensemble de cette offre est géré par un SOC (Security Operations Center).

c) Évolution économique

Actuellement, le marché de la télécommunication indirecte est en pleine croissance, il représente 15 % du marché du télécom, le reste du marché étant occupé par les géants Orange, Bouygues et SFR à 80 %. Le marché des opérateurs indirects a pour prévision d'atteindre les 2 Mds € en 2025. Alphalink avait une croissance de 20 % par an de 2013 jusqu'à 2021, l'entreprise a environ 190 salariés et son chiffre d'affaires était de 64 M € en 2024.

d) Statut juridique

Alphalink a été créé le 1 juillet 1999 par deux associés, c'est une SAS (Société par Actions Simplifiés).

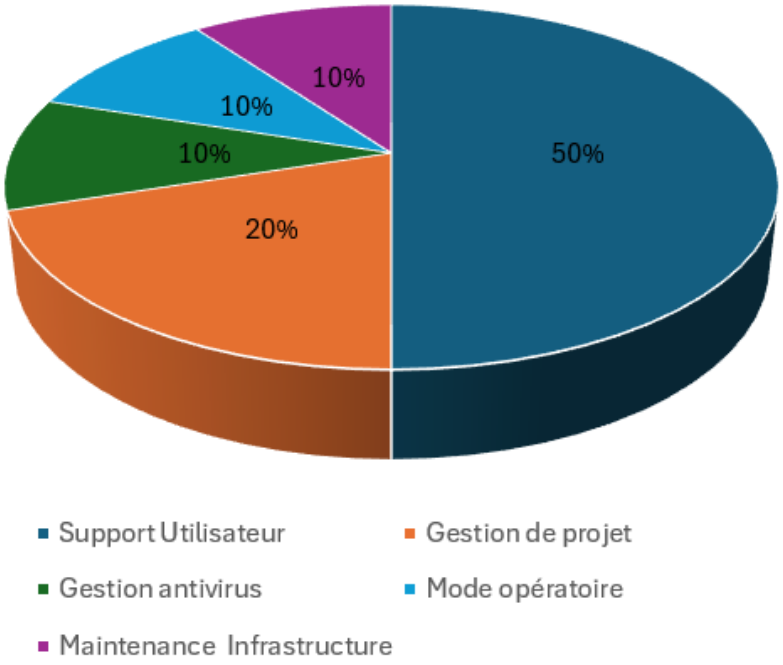
e) Organigramme

Nous pouvons voir sur cette organigramme la direction ainsi que le service infrastructure avec 13 membres, mon tuteur, Aurélien LALLEMAND se trouve également dans ce service infrastructure.

f) Présentation du service

Missions réalisées en entreprise

Différentes tâches effectuées



Nous pouvons voir ci-dessus un secteur en 3D. Dans celui-ci, nous pouvons retrouver les différentes activités que j'ai pu réaliser en entreprise, telles que du support utilisateur à 50 %, de la gestion de projet à 20 %, de la gestion d'antivirus à 10 %, de l'écriture de mode opératoire à 10 % ainsi que de la maintenance de notre infrastructure à 10 %.

Mission 1 : Support Utilisateur

1) Contexte et objectifs

Dans le cadre de cette mission, nous devions faire du ticketing pour tout ce qui concerne l'accueil et le départ des salariés. Nous devions également régler tout ce qui est incident, problèmes des collaborateurs ainsi que toutes les failles de sécurité pouvant concerner les utilisateurs. L'objectif de cette mission de support utilisateur était de s'occuper de l'ensemble des activités techniques que nous pouvons avoir en interne avec nos collaborateurs.

2) Outil

L'outil utilisé dans le cadre de cette mission était la plateforme Easyredmine, qui nous permettait de faire du ticketing avec nos collaborateurs en interne. Nous pouvons voir ci-dessous une image de la plateforme avec différentes tâches.

3) Missions concrètes réalisées

Sur la partie support utilisateur j'ai pu concrètement réaliser différentes missions :

- Gestion des non-conformités pour les utilisateurs via nos outils de gestion de logiciels ou de mises à jour

- Gestion de comptes sur nos outils en interne et particulièrement sur office 365, création, suppression et réinitialisation de mot de passe

- Gestion de postes utilisateurs afin de régler les problèmes liés aux réseaux, aux appareils en Bluetooth ou même aux fichiers

4) Compétences acquises

Au cours de cette mission, j'ai pu apprendre à faire un diagnostic ainsi que de la résolution d'incident. Ensuite, j'ai pu travailler sur ma communication à l'oral avec les collaborateurs en interne. Enfin, cette mission m'a permis d'apprendre à m'organiser et à prioriser les différentes tâches que j'ai pu réaliser.

Mission 2 : Gestion de Projet

1) Contexte et objectifs

Dans le cadre de cette mission, j'ai pu m'occuper de faire de la gestion de projet, c'est-à-dire que nous avons pu réaliser un ensemble d'activités afin d'arriver à une finalité qui nous permet de faire évoluer notre infrastructure afin de répondre à des besoins que nous avons pu identifier au préalable. L'objectif de cette gestion de projet était de réaliser le livrable via des tâches planifiées tout en respectant des contraintes de temps et de budget.

2) Outil

L'outil que j'ai pu utiliser dans le cadre de cette gestion de projet était Easyredmine comme pour la partie Ticketing car sur celui-ci nous avons également un partie Gestion de Projet.

3) Missions concrètes réalisées

Dans le cadre de cette mission, j'ai pu concrètement réaliser différentes tâches telles que :

- La gestion de tâches planifiées avec Easyredmine via un GANTT
- La réalisation d'études de solutions afin de savoir quelle solution pourrait correspondre à notre besoin
- La réalisation d'évaluations des risques afin de connaître la faisabilité du projet et de savoir quel est l'impact de celui-ci

4) Compétences acquises

Lors de cette mission j'ai pu acquérir différentes compétences. Tout d'abord j'ai pu apprendre la méthodologie de la gestion de projet. Ensuite j'ai pu voir comment organiser différentes tâches et mettre des priorités sur les différentes activités que j'ai pu réaliser. Enfin j'ai appris à faire une analyse des risques par rapport à la mise en place d'un projet dans une infrastructure.

Mission 3 : Gestion Antivirus

1) Contexte et objectifs

Dans le cadre de cette mission, j'ai pu m'occuper de faire de la gestion d'antivirus, c'est-à-dire que j'ai effectuée différentes tâches afin de mettre en conformité nos différents appareils qui sont liés à l'antivirus et j'ai pu mettre en place des scripts afin de récupérer des informations ou d'en enlever.

L'objectif de cette mission était de mettre en conformité nos différents appareils au sein de l'entreprise.

2) Outil

L'outil que j'ai pu utiliser lors de cette mission était Sophos qui est un anti-virus Mac et Windows qui permet de bloquer les ransomware et différentes attaques ainsi que de chiffrer nos différents appareils pour éviter toute perte de données.

3) Missions concrètes réalisées

Dans le cadre de cette mission, j'ai pu concrètement réaliser différentes tâches telles que :

- Mise en conformité de postes utilisateurs que ce soit en effectuant des mises à jour ou en désinstallant des applications interdites

- Mise en place de scripts afin de déployer des packages

- Mise en place de stratégies utilisateurs pour des téléphones

4) Compétences acquises

Au cours de cette mission, j'ai pu acquérir quelques compétences. Tout d'abord de la gestion de parc informatique via l'Endpoint de Sophos. Ensuite j'ai pu comprendre et analyser les risques qu'il pouvait y'avoir autour d'un utilisateur et d'un poste informatique. Enfin cela m'a permis de travailler sur mes compétences de Scripting Bash.

Projet 1 : Gestion de logiciels

a) Problématique

b) Veille effectuée avant le projet

Avant de démarrer la mise en place de la solution, nous avons réalisé avec mon tuteur une veille informatique. Celle-ci consistait à étudier le fonctionnement de l'outil, vérifier s'il était toujours maintenu à jour, s'il répondait correctement à nos besoins et s'il correspondait à notre environnement.

Grâce à cette veille informatique, nous avons pu conclure que Munki correspondait à nos besoins. Avec cette solution, nous pouvions installer et mettre à jour les logiciels souhaités, et même mettre à jour le système d'exploitation MacOs. Nous pouvions également choisir le type de mise à jour : invisible pour les utilisateurs, nécessitant une action de leur part, ou forcée à partir d'une certaine date.

Cette veille nous a également permis de constater que la communauté Munki était active. En effet, la dernière version majeure disponible au 07/07/2025 datait du 11 février 2025, et la dernière version en bêta datait du 1er juillet 2025.

c) Planification

Le projet pour la mise en place de la solution Munki a été séparé en divers étapes, ces étapes sont détaillées dans les différentes parties tel que la problématique la veille effectuée avant le projet ou encore la mise en place du projet.

1) Présentation du Projet (1 semaine)

Cette partie correspondait à toute la présentation du projet à mon n+2 (Deux personnes au-dessus de moi dans l'organigramme).

2) Etude de la faisabilité du Projet (1 semaine)

Dans cette partie nous avons étudié ce qui pouvait bloquer le projet ainsi les besoins matériels et immatériels.

3) Mise en place et configuration du matériel (1 semaine)

Cette partie correspondait à l'ensemble de la configuration que nous avons réalisé sur la machine physique afin qu'elle soit facilement accessible.

4) Installation et configuration de la solution Munki (3 semaines)

Cette étape correspondait à toute l'installation de la solution que nous avions choisie et la configuration que nous avons réalisée afin d'intégrer Munki dans notre infrastructure.

5) Documentation (1 semaine)

Cette partie correspondait à toute la documentation à faire pour les personnes qui sont voués à utiliser l'outil de gestion de logiciel.

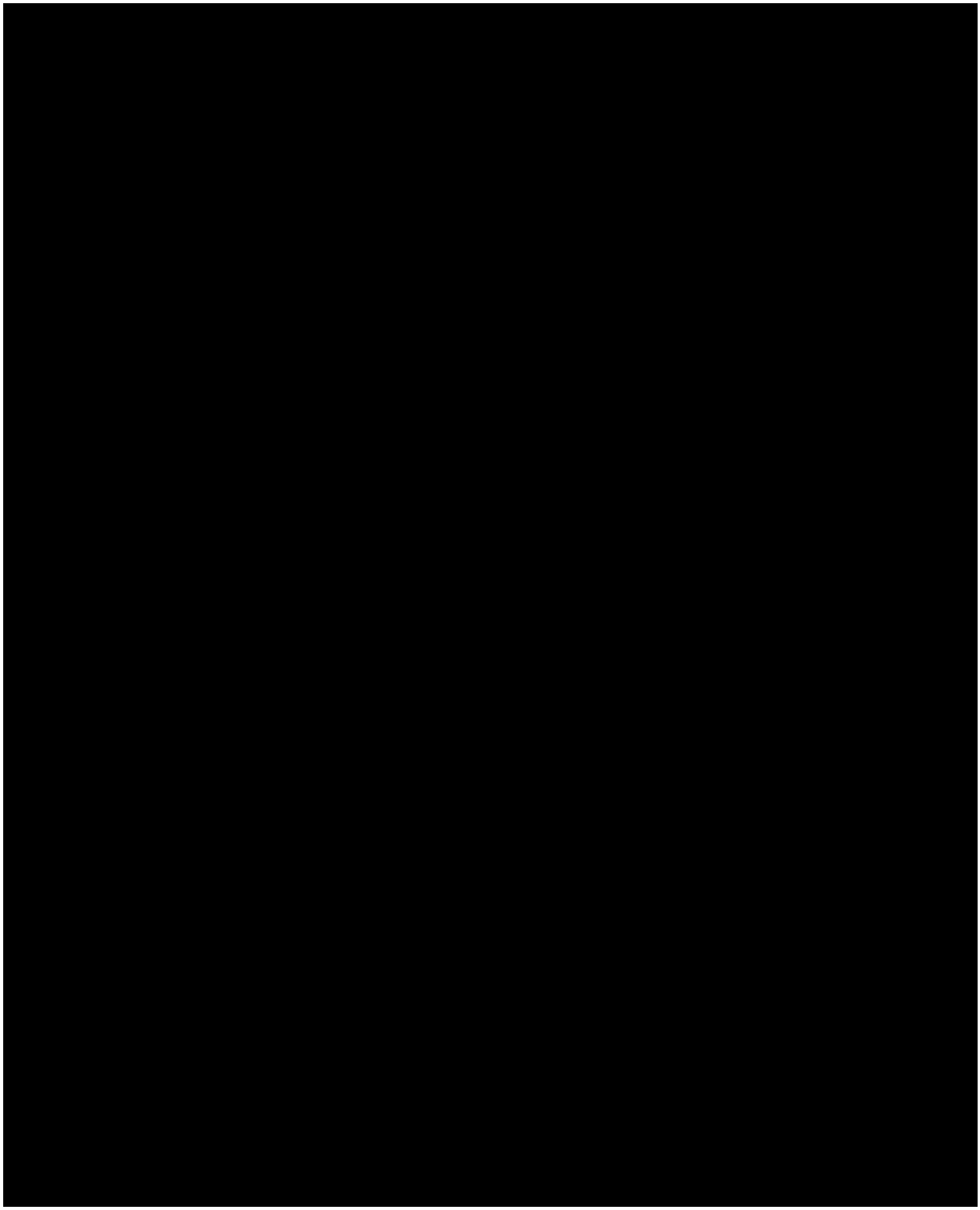
6) Mise en Production de la solution

La mise en production correspondait aux actions que nous avons réalisé pour installer le Centre de Gestion de logiciels côté utilisateur (collaborateurs en interne).

7) Clôture du Projet

Pour terminer, la clôture du projet correspondait à la fin du projet, celle-ci est signé lorsque tout a été mis en place et que cela correspond complètement aux attentes et à la problématique.

Ci-dessous, nous pouvons voir les tâches du diagramme de GANTT ainsi que le diagramme correspondant, celui-ci permet de voir et de comprendre l'ensemble des tâches du projet, nous pouvons également voir à qui elles sont attribués et le temps de réalisation de ces différentes tâches.

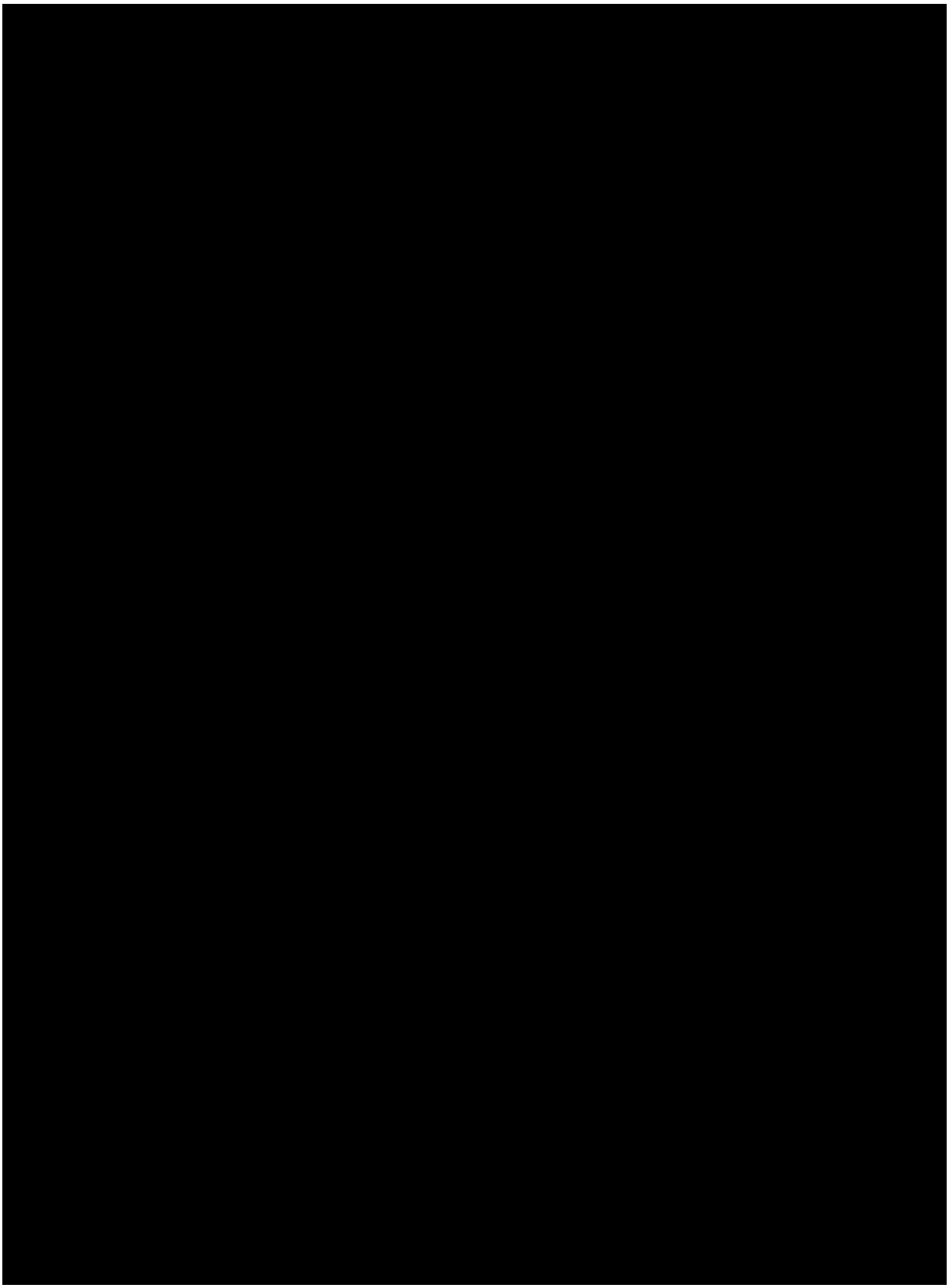


d) Liste du matériel à disposition

Dans ce projet, nous avons eu du matériel physique mis à notre disposition afin de déployer la solution Munki. Voici la liste du matériel qui a été utilisé pour le projet :

- Apple Mac Mini
- Câble d'alimentation Apple Mac Mini
- Câble RJ45

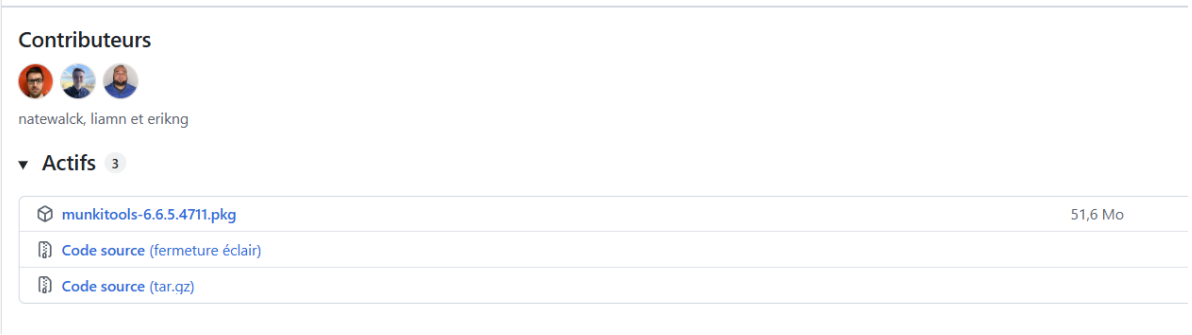
e) Configuration du matériel



f) Installation de la solution

Ensuite, pour la mise en place de solution, nous devons d'abord faire l'installation de la solution Munki sur le Mac Mini, pour installer celle-ci, il faut télécharger le .pkg correspondant à la solution directement sur GitHub au lien ci-dessous.

https://github.com/munki/munki/releases



Mais avant de lancer ce package, nous devons créer la structure de répertoire qui va être utilisée par l'outil, l'action recommandée par les développeurs est de créer dans /Users/Shared les différents répertoires qui vont être utilisés par l'outil, c'est donc ce qu'il faut faire comme nous pouvons le voir ci-dessous.

```
cd /Users/Shared
mkdir munki_repo
mkdir munki_repo/catalogs
mkdir munki_repo/icons
mkdir munki_repo/manifests
mkdir munki_repo/pkgs
mkdir munki_repo/pkgsinfo
```

```
% cd /Users/Shared
Users/Shared % mkdir munki_repo
Users/Shared % mkdir munki_repo/catalogs
Users/Shared % mkdir munki_repo/icons
Users/Shared % mkdir munki_repo/manifests
Users/Shared % mkdir munki_repo/pkgs
Users/Shared % mkdir munki_repo/pkgsinfo
Users/Shared %
```

Dans cette structure, chaque répertoire va avoir son utilité, tout d'abord le répertoire catalogs va permettre de contenir les différents catalogues que nous allons créer qui sont des listes de l'ensemble des packages disponible.

Ensuite, le répertoire icons va permettre de stocker les icônes de chaque application ajoutée dans Munki.

Après le répertoire manifests va permettre de définir ce que chaque client/groupe doit installer.

Puis le répertoire pkgs va permettre de contenir les fichiers d'installation des applications qui seront ajoutés, les .pkg ou les .dmg pour Mac par exemple.

Enfin, le répertoire pkgsinfo contient dans un fichier XML les paramètres que nous voulons pour l'application Munki qui est configuré, par exemple si on veut désinstaller des appareils une application ou si nous voulons que la session de l'utilisateur soit fermée pour l'installation du package.

Après avoir configuré la structure, il faut lancer le package qui a été téléchargé précédemment. Ce paquet va installer les outils Shell de Munki, ce qui va nous servir sur le serveur pour l'administration mais il va également installer le client Munki car tout est dans le même package (Center.app dans les applications).

Grâce à cela nous avons maintenant un dépôt Munki fonctionnel mais il est vide et inutile.

Afin de pouvoir l'utiliser, il faut configurer le dépôt de l'application avec la commande ci-dessous qui nous permet de définir l'emplacement de notre dépôt Munki, l'extension que l'on veut pour nos fichiers de configuration, l'éditeur qui sera utilisé pour les fichiers de configuration, le Catalog que l'on va utiliser par défaut ou encore le dossier pour les plugins.

```
/usr/local/munki/munkiimport --configure
```

```
Repo URL (example: afp://munki.example.com/repo): file:///Users/Shared/munki_repo
pkginfo extension (Example: .plist):
pkginfo editor (examples: /usr/bin/vi or TextMate.app; leave empty to not open an editor after import): TextEdit.app
Default catalog to use (example: testing): testing
Repo access plugin (defaults to FileRepo):
```

g) Configuration de Munki

1) Import d'un package dans l'outil

Après avoir terminé l'ensemble de l'installation, il est possible d'ajouter un package dans Munki en ligne de commande afin de tester le bon fonctionnement de l'application. Pour cela, il faut ajouter le paquet correspondant, par exemple Chrome avec la commande ci-dessous.

```
/usr/local/munki/munkiimport ~/Downloads/googlechrome.dmg
```

```
Analyzing installer item...
This item is similar to an existing item in the repo:
            Item name: GoogleChrome
         Display name: Google Chrome
          Description: Chrome is a fast, simple, and secure web browser, built for the modern web.
              Version: 143.0.7499.170
  Installer item path: apps/Google/GoogleChrome-143.0.7499.170.pkg

Use existing item as a template? [y/N] y
Copying name: GoogleChrome
Copying blocking_applications: (
    "Google Chrome"
)
Copying unattended_install: True
Copying category: MacOS
Copying developer: Google
[           Item name: GoogleChrome
[        Display name: Google Chrome
[         Description: Chrome is a fast, simple, and secure web browser, built for the modern web.
[             Version: 139.0.7258.128
[            Category: MacOS
[           Developer: Google
[  Unattended install: True
[Unattended uninstall: False
[            Catalogs: Catalog

Import this item? [y/N] y
[Upload item to subdirectory: apps/Google
No existing product icon found.
Attempt to create a product icon? [y/N] y
Attempting to extract and upload icon...
Imported icons/GoogleChrome.png
Copying googlechrome.dmg to repo...
Copied googlechrome.dmg to pkgs/apps/Google/googlechrome-139.0.7258.128.dmg.
Edit pkginfo before upload? [y/N]: n
Saved pkginfo to pkgsinfo/apps/Google/GoogleChrome-139.0.7258.128.
Rebuild catalogs? [y/N] y
```

Grâce à cela le paquet Google Chrome est ajouté dans le dépôt Munki, celui-ci est dans le catalog testing, il est possible de vérifier cela dans le fichier correspondant au catalog testing dans « munki_repo/catalogs/ » en allant modifier le fichier avec la commande ci-dessous et comme nous pouvons le voir celui-ci est présent dedans.

```
nano /Users/Shared/munki_repo/catalogs/testing
```

```
<dict>
        <key>autoremove</key>
        <false/>
        <key>blocking_applications</key>
        <array>
                <string>Google Chrome</string>
        </array>
        <key>catalogs</key>
        <array>
                <string>Install</string>
                <string>Production</string>
        </array>
        <key>category</key>
        <string>MacOS</string>
        <key>description</key>
        <string>Chrome is a fast, simple, and secure web browser, built for the modern web.</string>
        <key>developer</key>
        <string>Google</string>
        <key>display_name</key>
        <string>Google Chrome</string>
        <key>installed_size</key>
        <integer>652123</integer>
        <key>installer_item_hash</key>
        <string>db08f843cdbac2bf6c917866c56dd0d44568d1d39324eec1e84893e728e8a263</string>
        <key>installer_item_location</key>
        <string>apps/Google/GoogleChrome-143.0.7499.170.pkg</string>
        <key>installer_item_size</key>
        <integer>221800</integer>
        <key>installs</key>
        <array>
```

Maintenant que le paquet est ajouté dans le catalogue voulu, Munki doit savoir quoi installer sur telle machine, pour cela il faut configurer un Manifest client. Pour la configuration, il faut utiliser les commandes ci-dessous qui nous permettent d'ajouter notre application dans le manifest par défaut (site_default).

```
manifestutil
add-catalog --manifest site_default testing
add-pkg GoogleChrome --manifest site_default
exit
```

```
Entering interactive mode... (type "help" for commands, "exit" to quit)
> add-catalog --manifest site_default testing
Added 'testing' to catalogs of manifest site_default.
> add-pkg GoogleChrome --manifest site_default
Added 'GoogleChrome' to managed_installs of manifest site_default.
> exit
```

Après l'avoir ajouté, il faut vérifier qu'il est bien présent dans le manifest qui a été utilisé précédemment.

```
cat /Users/Shared/munki_repo/manifests/site_default
```

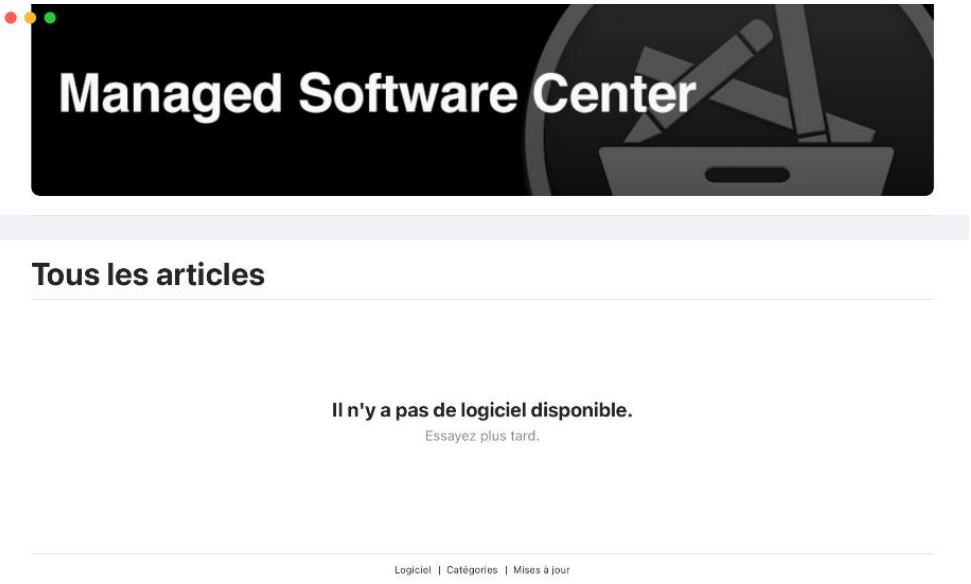
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.appl
<plist version="1.0">
<dict>
        <key>catalogs</key>
        <array>
                <string>Production</string>
        </array>
        <key>conditional_items</key>
        <array/>
        <key>default_installs</key>
        <array/>
        <key>included_manifests</key>
        <array/>
        <key>managed_installs</key>
        <array>

                <string>GoogleChrome</string>




        </array>
        <key>managed_uninstalls</key>
        <array/>
        <key>managed_updates</key>
        <array/>
        <key>optional_installs</key>
        <array>



        </array>
</dict>
</plist>
```

2) Test d'un client Munki

Maintenant que le serveur est configuré, nous devons configurer le client avec un package.

Pour l'installation, il faut tout d'abord télécharger le même paquet Munki que nous avons téléchargé pour le serveur, le seul changement est que l'on ne va pas configurer la partie serveur. Ce package va nous installer le « Centre de Gestion des Logiciels » dans les applications comme nous pouvons le voir ci-dessous.

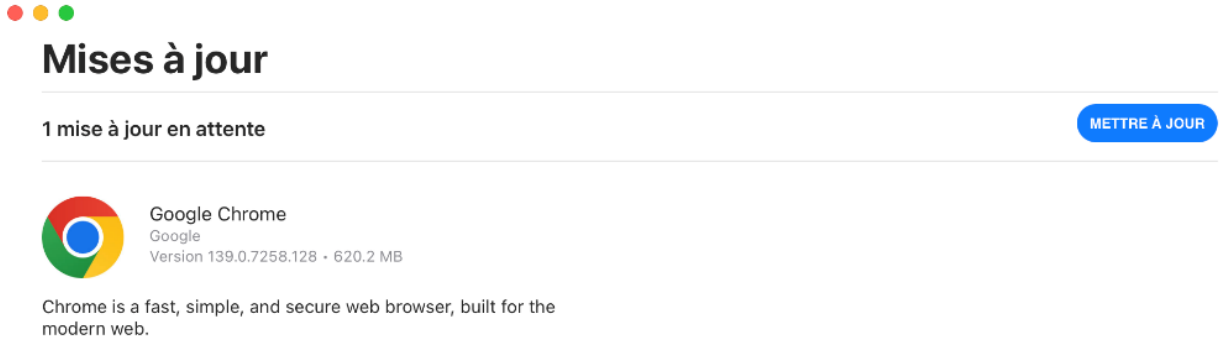
Centre de gestion
des logiciels

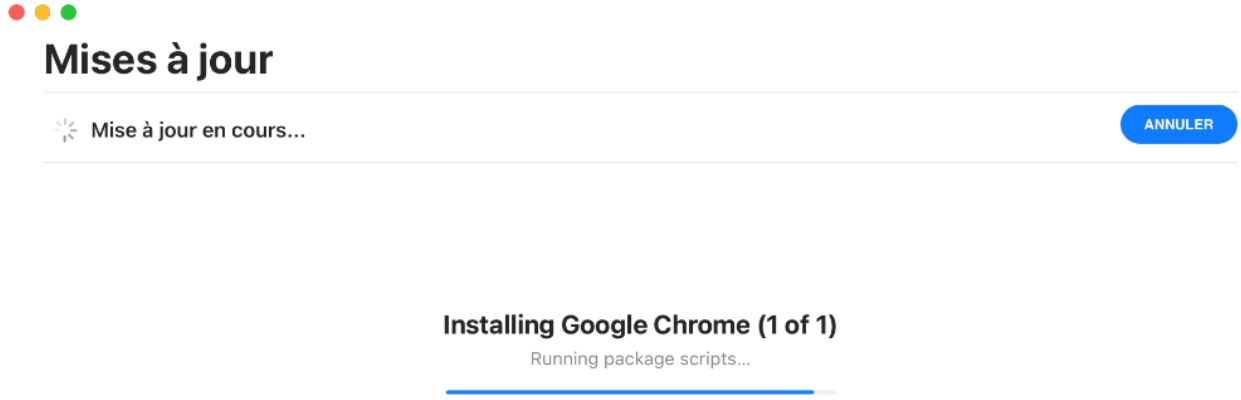


Après que l'application « Centre de gestion des logiciels » est installée, nous devons configurer sur le client la communication avec le serveur, pour cela, il faut taper la ligne de commande ci-dessous qui configure sur quel serveur nous allons aller et sur quel manifest.

```
sudo defaults write /Library/Preferences/ManagedInstalls SoftwareRepoURL
https://nom_srv_munki/munki_repo
```

Une fois cela fait, nous pouvons tester l'installation de notre package via le « Centre de gestion des logiciels », comme nous pouvons le voir ci-dessous, le logiciel a récupéré l'application et l'a téléchargée.





3) Administration Munki

Avec ce test client – serveur, nous avons pu voir comment ajouter un paquet en ligne de commande mais cela peut être un peu compliqué à administrer sur le long terme pour la partie serveur car de ce que nous avons vu ce n'est que de l'administration via un terminal.

Afin de faciliter cela, nous pouvons soit passer par une application Munki à mettre sur le serveur afin de réaliser l'administration ou mettre en place un serveur web pour l'administration.

Dans cette activité, nous allons uniquement voir la mise en place de l'application Munki à mettre sur le serveur car celle-ci est maintenue avec de nombreux contributeurs et il y a différentes versions, contrairement à la version web qui est encore en développement, aucune version n'a été publiée et en plus les contributeurs eux-mêmes déconseillent de mettre en place le serveur web en production car c'est un outil qui est en développement.

Pour mettre en place l'application de gestion Munki Admin, il faut télécharger le .dmg correspondant à la version stable au lien ci-dessous.

https://github.com/hjuutilainen/munkiadmin/releases?page=1

MunkiAdmin 1.10.1 Dernier

Après quelques années difficiles, je vous présente enfin une version de MunkiAdmin autre qu'une version bêta. Sa conception a été longue et de nombreuses modifications ont été apportées depuis la version stable précédente (1.8.1). Sans ordre particulier :

Modifications et correctifs :

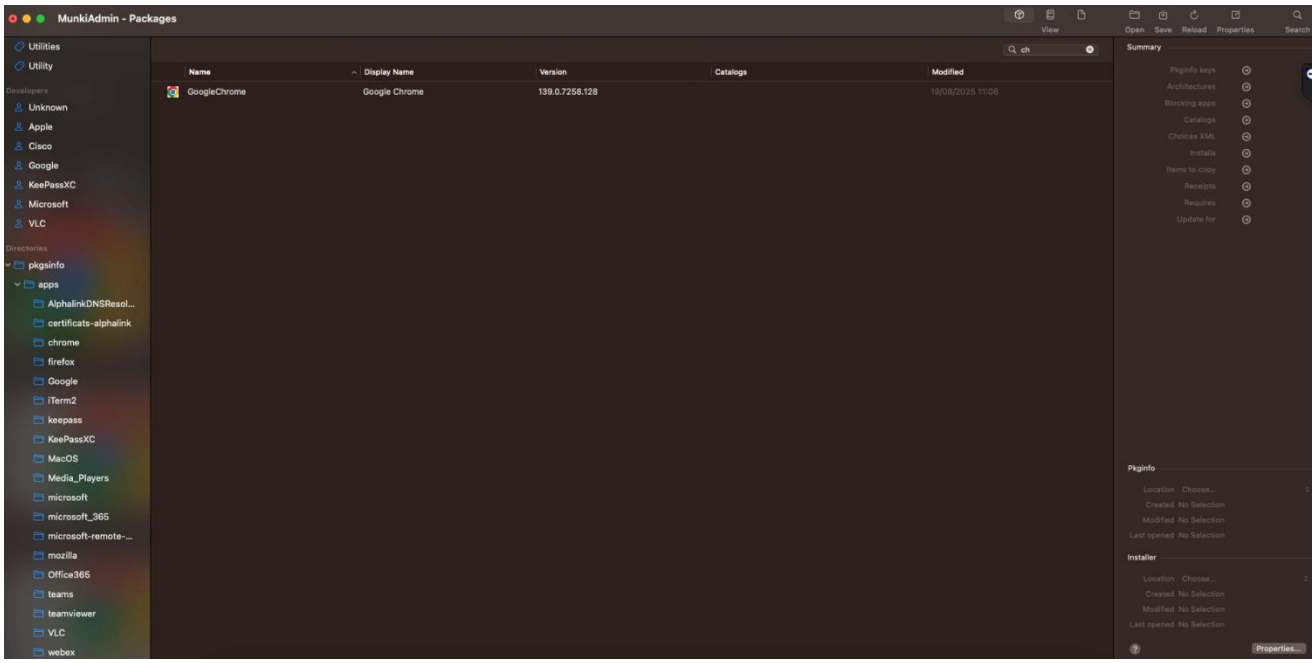
- Nouvelle icône d'application compatible Tahoe
- Refactorisation majeure de la barre latérale de la vue des manifestes pour utiliser NSOutlineView simple avec NSTreeController au lieu de PXSourceList obsolète
- Désactivez Liquid Glass pour le moment en définissant UIDesignRequiresCompatibility sur true
- Correction de la barre latérale de l'éditeur de manifeste affichant des nombres incorrects lorsque des éléments sont attribués à des conditions
- Supprimer le champ de texte d'emplacement de l'élément de désinstallation de l'éditeur pkginfo
- Ajouter 26.0 à la liste des versions minimales du système d'exploitation
- Ajouter les conditions de modèle macOS 15 et 26 à l'éditeur de manifeste
- Prise en charge de `version_script` la clé pkginfo ajoutée dans la version bêta de Munki 7
- Toutes les vues de texte de script dans l'éditeur pkginfo sont désormais regroupées dans une seule vue d'onglet Scripts
- Ajoutez un astérisque aux libellés de l'onglet Scripts lorsque le fichier pkginfo actuel contient des scripts. Cela devrait faciliter l'identification des éléments de l'onglet Scripts contenant du contenu.
- Pour les éléments d'installation, la clé de chemin est désormais requise uniquement pour les éléments de type fichier ou plist
- Le champ de texte de la méthode de désinstallation dans l'éditeur pkginfo est désormais une zone de liste déroulante qui complète automatiquement les valeurs
- Prise en charge de `default_installs` la clé manifeste ajoutée dans Munki 6.1
- Ajout d'un élément de barre latérale pour `stage_os_installer` le type pkginfo
- Conditions de modèle mises à jour et chaînes d'exigences du système d'exploitation avec les nouvelles versions de macOS
- Mettre à jour la dépendance CocoaLumberjack

Journal des modifications complet : `v1.8.1...v1.10.1`

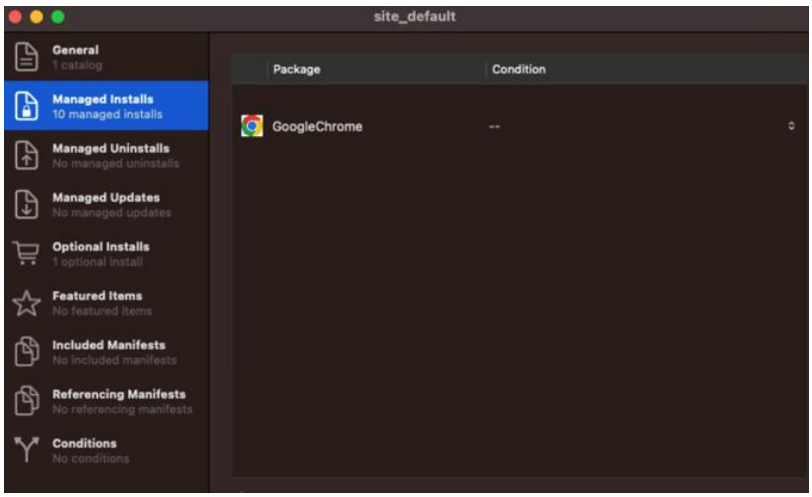
▾ Actifs 3

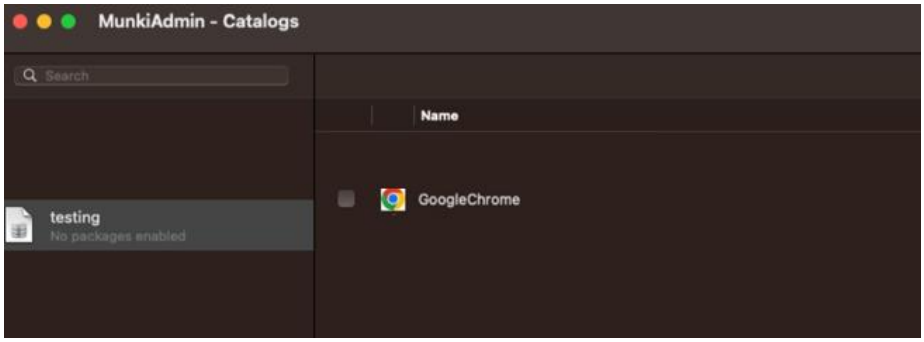
MunkiAdmin-1.10.1.dmg	sha256:29a4c5ba0ba9cddd...	4,71 Mo	il y a 4 jours
Code source (fermeture éclair)			il y a 4 jours
Code source (tar.gz)			il y a 4 jours

Après avoir installé l'outil et l'avoir ouvert, nous tombons sur la page ci-dessous qui est la page d'administration de Munki.



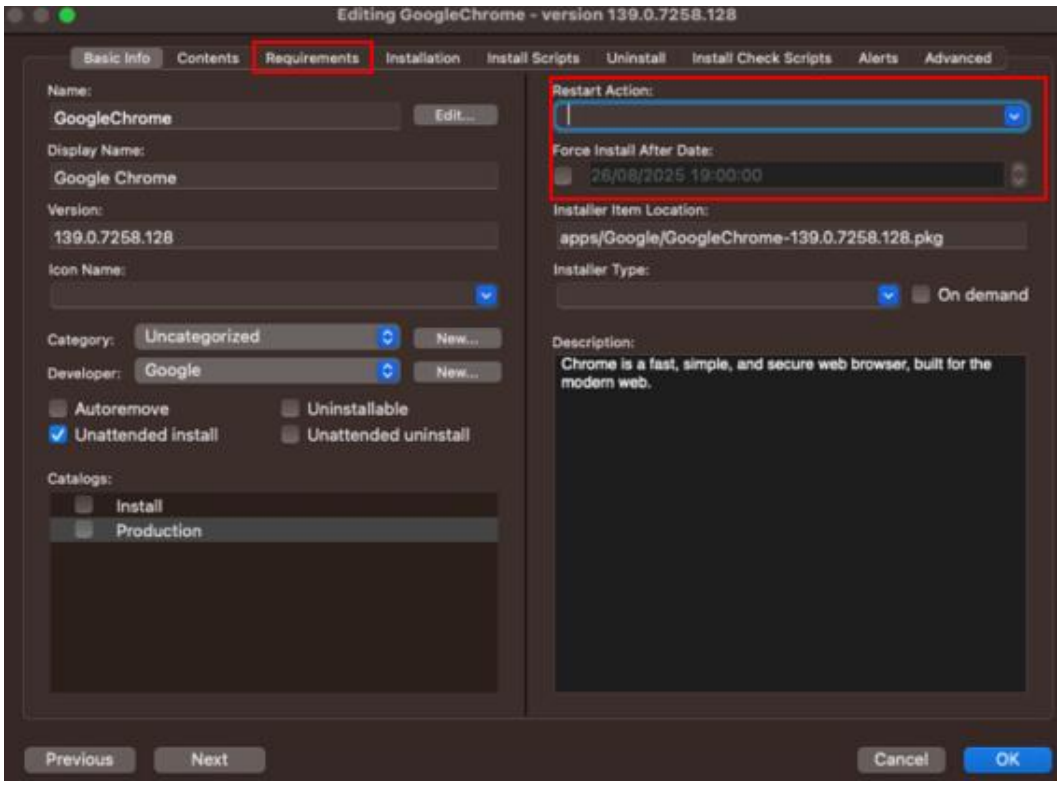
Dans cette interface, il est possible d'ajouter les packages facilement de façon graphique et nous pouvons également choisir le manifest que l'on veut comme sur la première image ci-dessous ou encore le Catalog voulu comme sur la deuxième image.





Grâce à cette application, nous pouvons également modifier la façon dont va être déployée l'application choisie, comme on peut le voir ci-dessous.

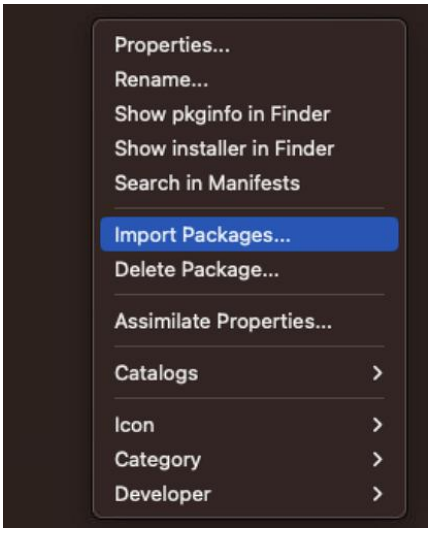
Par exemple il est possible de forcer le déploiement de l'application après une date donnée, il est également possible de configurer le besoin de redémarrer ou de déconnecter la session, nous pouvons même définir des prérequis pour l'installation de chaque application.



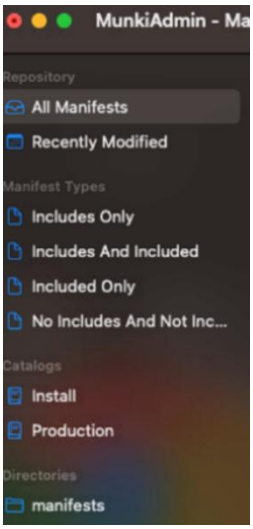
Grâce à cette nouvelle utilisation de Munki bien plus abordable et aux nouvelles fonctionnalités que nous avons pu voir ci-dessus, l'outil est désormais plus facile à utiliser.

h) Personnalisation de l'outil

Après avoir créé les catalogs, nous pouvons ajouter toutes les applications destinées aux utilisateurs. L'ajout de chaque application se fait via l'interface graphique en important les packages par un clic droit. Il est également possible de créer différentes catégories afin de classer les applications. Nous pouvons voir cela sur les deux images ci-dessous.



Enfin, il faut créer les différents manifests en respectant le même schéma que pour les catalogs, dans notre cas, nous n'avons pas créé un manifest par groupe d'utilisateurs car l'ensemble des personnes utilisent les mêmes applications, il n'y a pas vraiment d'exception. Nous pouvons voir les différents manifests ci-dessous.



i) Documentation

Après cette phase de personnalisation, la configuration et la personnalisation de Munki sont terminées mais afin que toutes les personnes comprennent correctement l'outil, une documentation a été réalisée. Cette documentation vise à décrire et expliquer l'application Munki et le Centre de gestion des logiciels afin que les personnes qui vont utiliser l'outil en administration puissent comprendre comment il fonctionne.

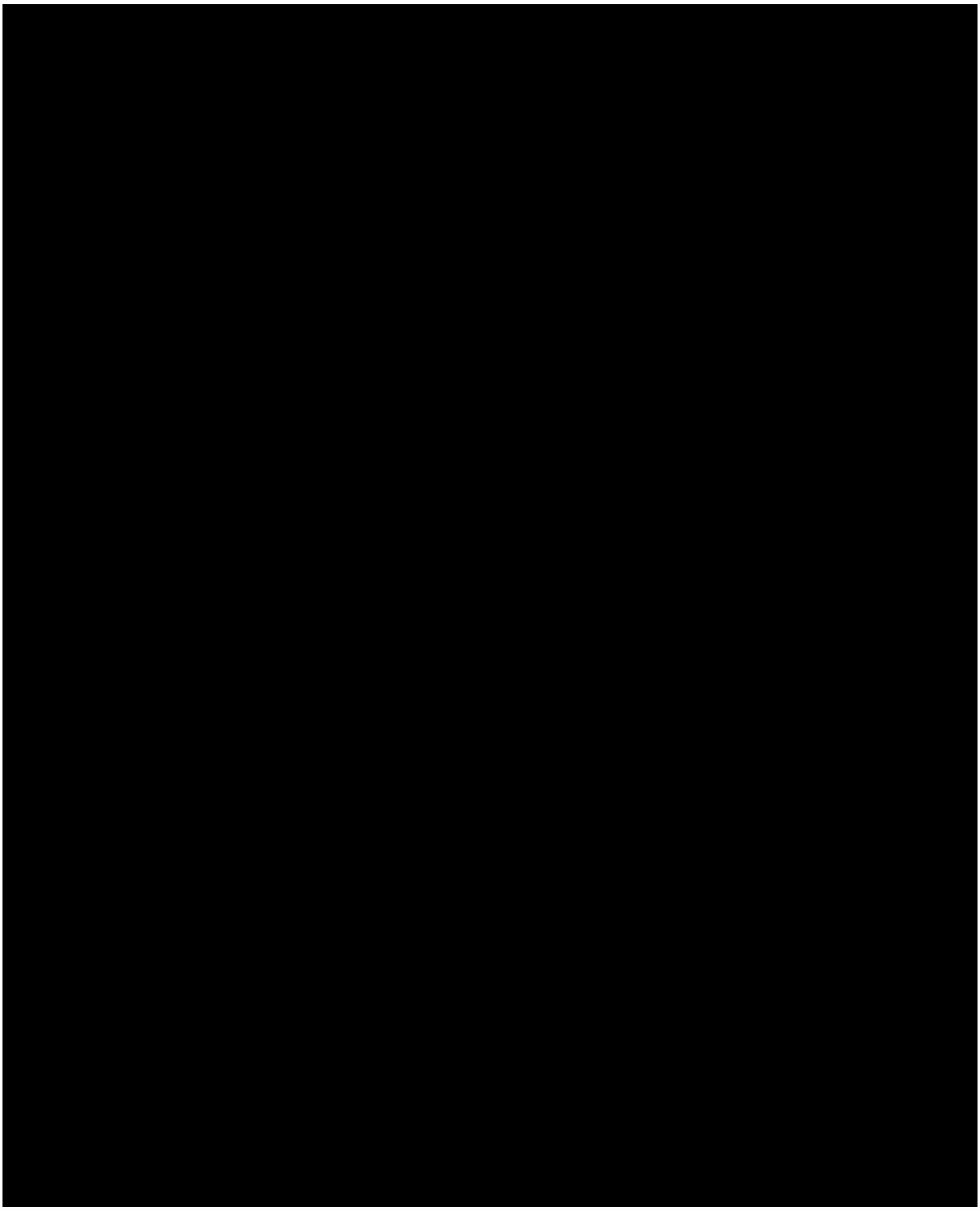
Une fois le fichier téléchargé et le logiciel installé et configuré, une nouvelle application apparaîtra sur votre ordinateur : le **Centre de gestion des logiciels**. C'est via cette application que vous pourrez installer d'autres paquets selon vos besoins.

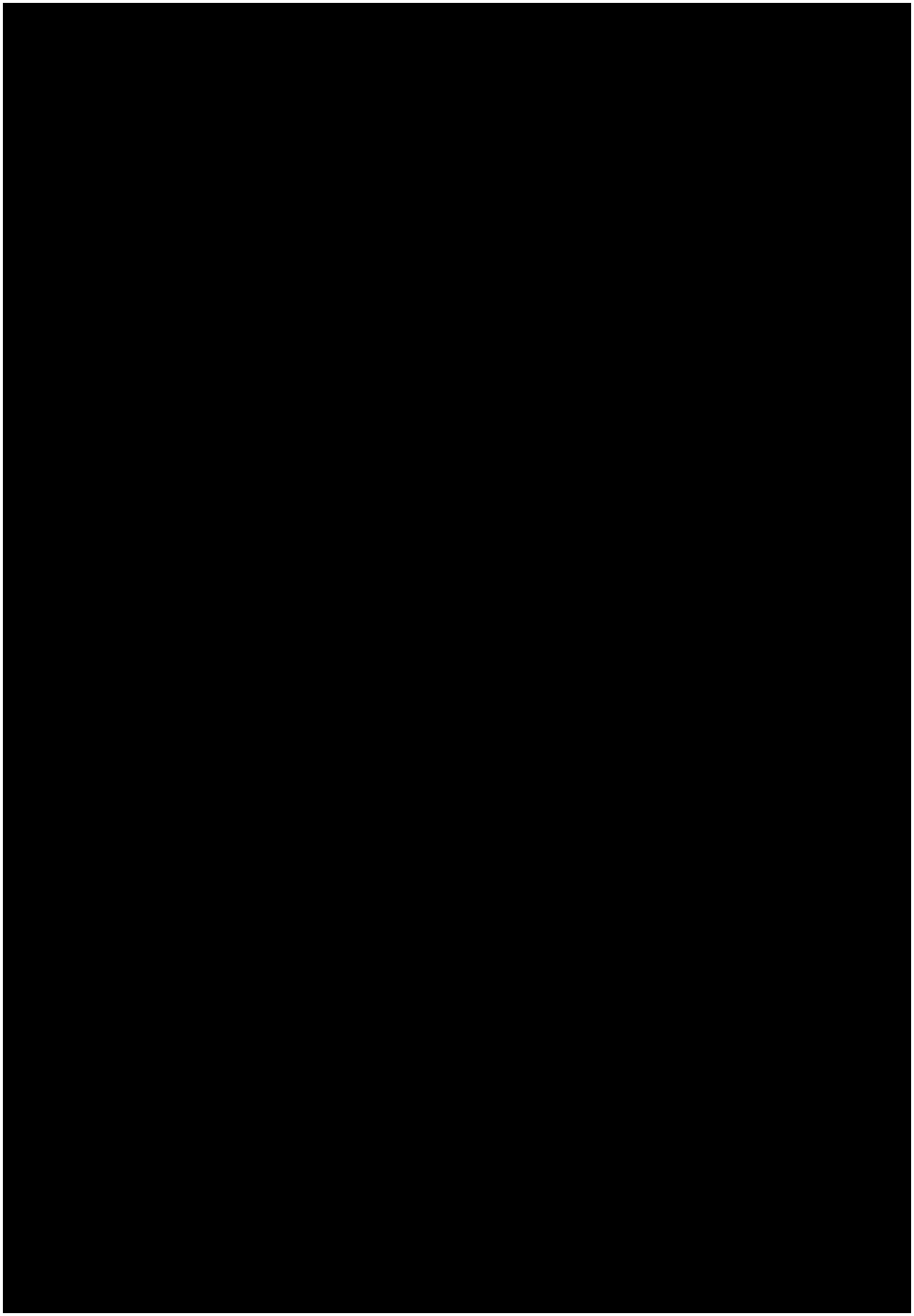
Centre de gestion
des logiciels

Lorsque vous ouvrez cette application, vous arrivez sur la page ci-dessous. Sur cette page, vous pouvez voir :

- Les logiciels qui doivent être installés ou ceux dont les mises à jour ne sont pas encore effectuées, dans la section **« Mise à jour »**. Par exemple, sur la première image ci-dessous, Firefox apparaît comme un logiciel à installer.
- Les logiciels optionnels, qui ne sont pas obligatoires mais peuvent répondre à certains besoins spécifiques, dans la section « **Logiciel** ». Par exemple, comme sur la seconde image, VLC est proposé.

Dans cette documentation, nous pouvons également retrouver toute une partie qui explique les différentes façons pour ajouter les packages ou encore pour gérer les manifests et les catalogs.



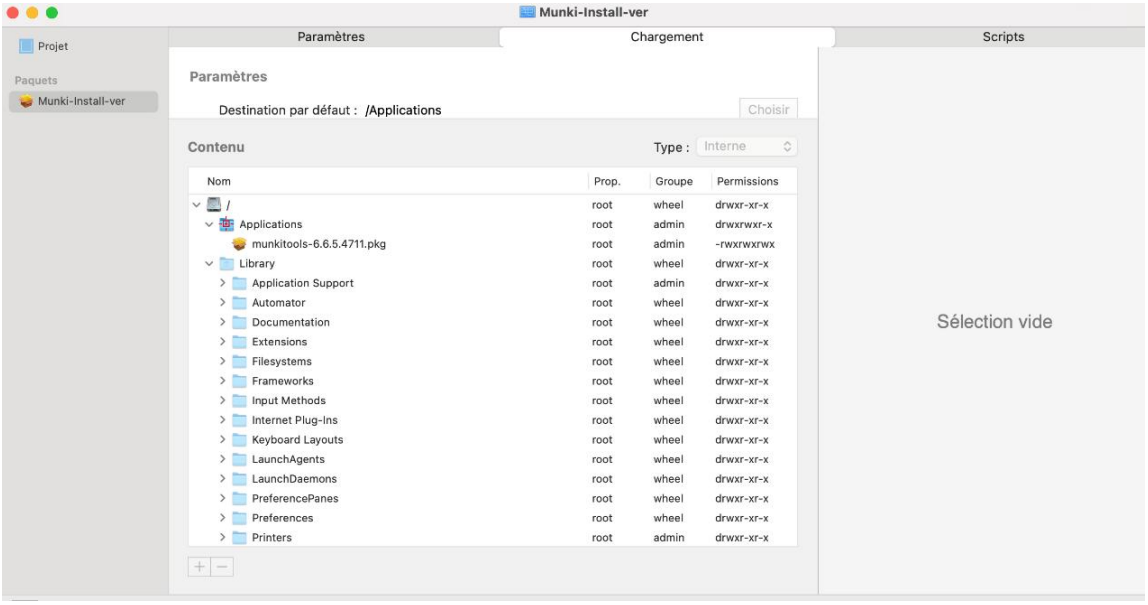


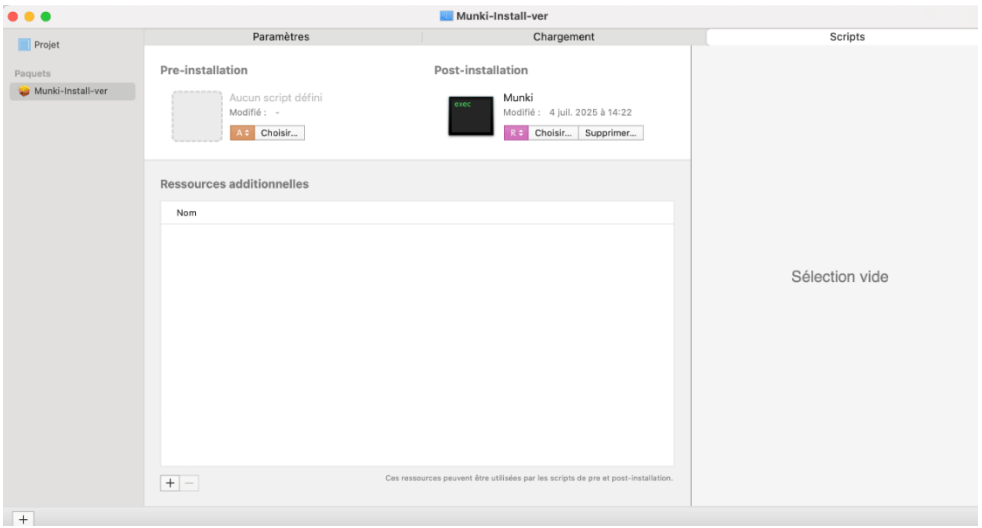
j) Mise en production

1) Installation de l'outil

Après avoir réalisé l'ensemble de la configuration de Munki, il faut réaliser la mise en production de l'outil. Afin de faire la mise en production du Centre de gestion des logiciels, il faut tout d'abord installer l'application sur l'ensemble des postes utilisateurs.

Pour réaliser cela, nous pouvons directement utiliser le package qui est fourni par le GitHub de Munki, ce même package que nous avons pu installer pour le test client-serveur. Avec ce package, nous devons ajouter un script Bash qui permet de configurer le nom DNS du serveur Munki, ensuite le package original de Munki et le script doivent être dans un nouveau package pour que cela puisse être déployé, nous pouvons voir la création de celui-ci ci-dessous et le script Bash en annexe.





Après avoir créé ce package, nous devons réaliser le déploiement de l'application sur l'ensemble du parc informatique.

Mais afin de respecter certaines normes de sécurité, nous avons tout d'abord pu passer cette action par le CAB (Change Advisory Board) ou Comité consultatif sur les changements en français.

Ce CAB regroupe l'ensemble de la direction technique de l'entreprise et il permet d'évaluer et d'approuver les changements apportés à l'infrastructure.

Des décisions ont été prises par ce CAB, celles d'avertir l'ensemble des utilisateurs du déploiement de ce nouveau logiciel sur leurs postes et de réaliser un déploiement progressif sur les différents services que nous pouvons avoir.

Tout d'abord pour la prévention utilisateur, nous avons fait une communication par mail pour expliquer l'outil et l'ordre de déploiement.

Bonjour à toutes et à tous,

Dans le cadre de l'amélioration continue de nos outils, nous débutons le déploiement progressif de la solution Munki. Le nom de l'application apparaîtra sous la forme suivante sur vos Mac : **Centre de gestion des Logiciels**.

Centre de gestion
des logiciels

Munki est un outil permettant d'automatiser les installations, les mises à jour et la gestion des logiciels sur les ordinateurs macOS. Cela permet :

- Une meilleure cohérence des versions d'applications entre les postes.
- Une réduction des interruptions liées à des mises à jour manuelles.
- Un gain de temps pour tous nos utilisateurs.

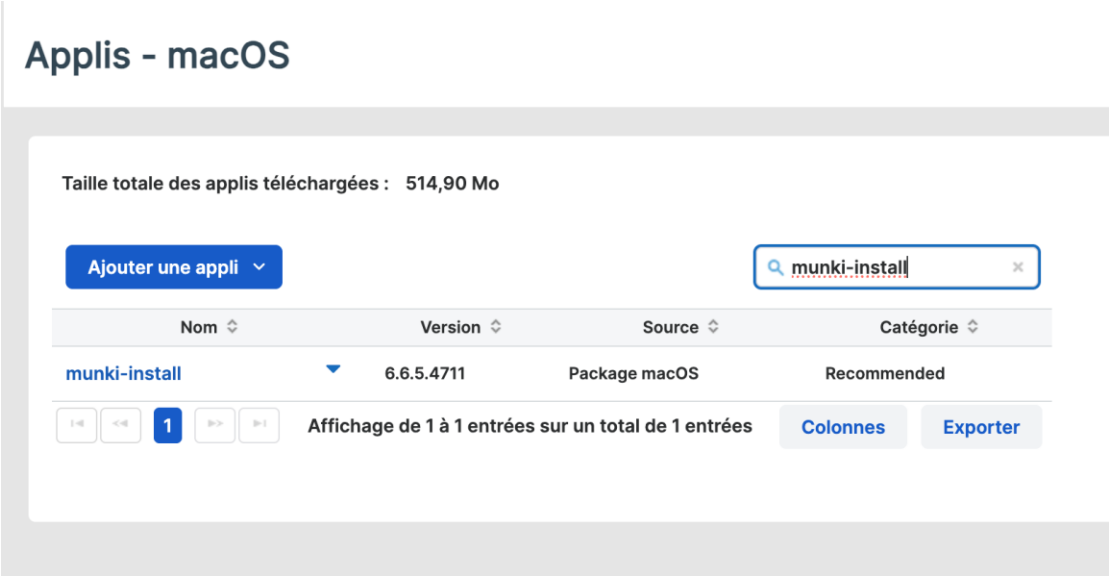
Comment cela va-t-il se dérouler ?

Le déploiement sera effectué progressivement par service, selon le planning suivant :

1. **Direction technique**
2. **Service ADV**
3. **Service Administratifs**
4. **Service Support**
5. **Commerce**

Le déploiement de la solution est prévu du **6 janvier 2025** au **13 janvier 2025**. Si vous rencontrez des dysfonctionnements pendant cette période, merci de revenir vers l'équipe AdminLan afin que nous puissions régler vos problèmes.

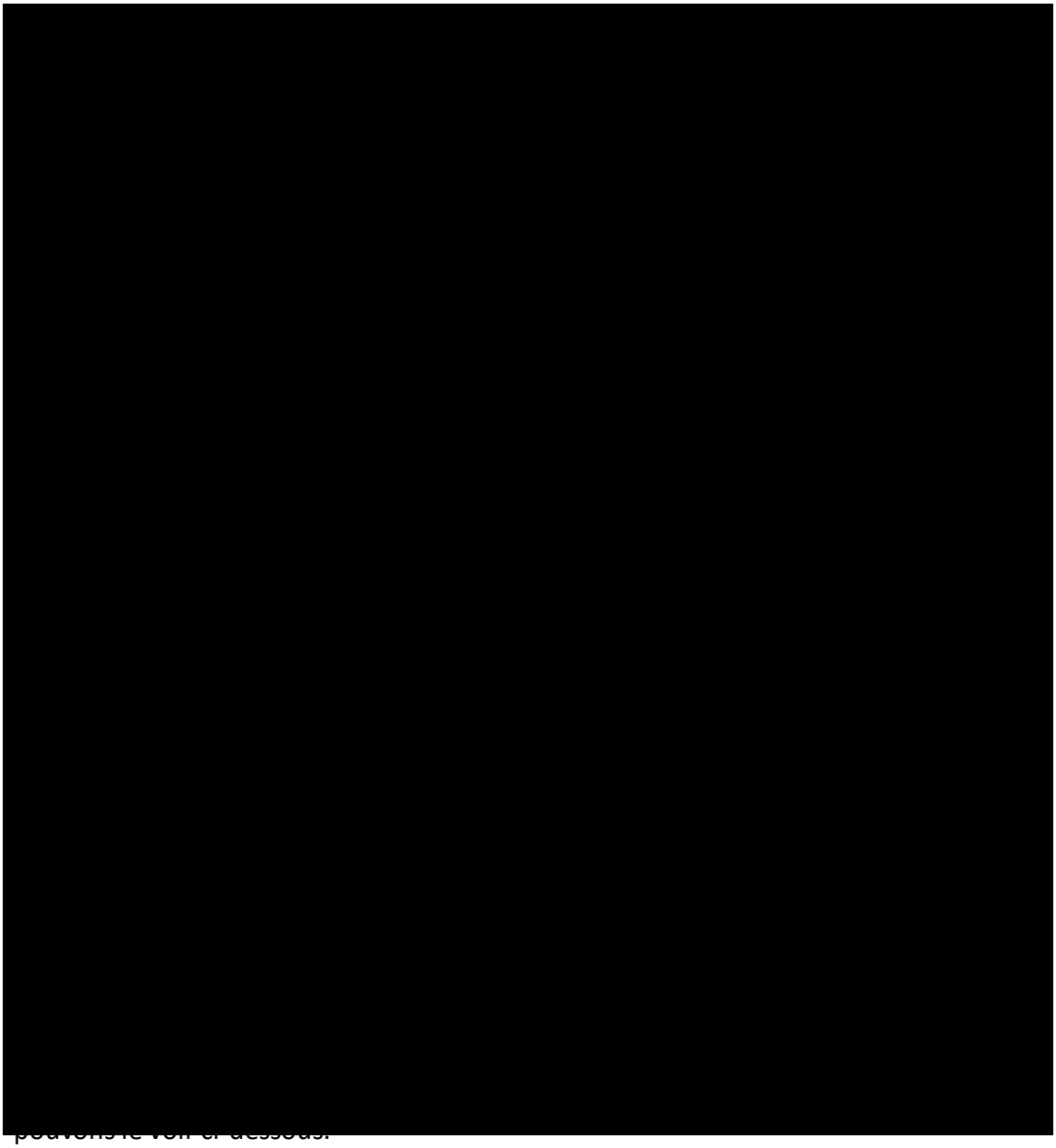
Maintenant, nous devons réaliser la configuration pour le déploiement de notre paquet, dans notre cas, nous l'avons fait via notre antivirus qui est Sophos. Nous avons pu appeler ce paquet « munki-install » comme nous pouvons le voir sur l'image ci-dessous.



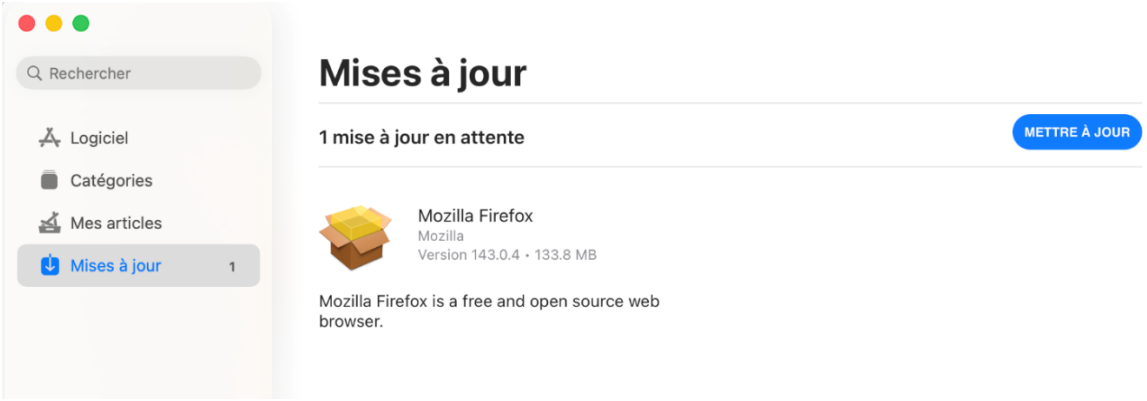
Une fois l'ajout fait, nous avons déployé le paquet service par service en commençant par la direction technique. Pour cela, nous avons choisi à chaque fois les différentes personnes de chaque service, comme nous pouvons le voir ci-dessous où j'ai pu me sélectionner.

Après avoir sélectionné les différentes personnes de chaque service, il faut regarder s'il n'y pas de tâches en erreur, si c'est le cas, il faut noter les tâches qui sont en erreur sur Sophos et celle qui sont en succès afin d'avoir une liste de ceux sur qui cela n'a pas forcément fonctionné car si des personnes sont en congés ou qu'il y a tout simplement une erreur sur le déploiement il faut alors déployer une nouvelle fois l'outil sur les postes concernés.

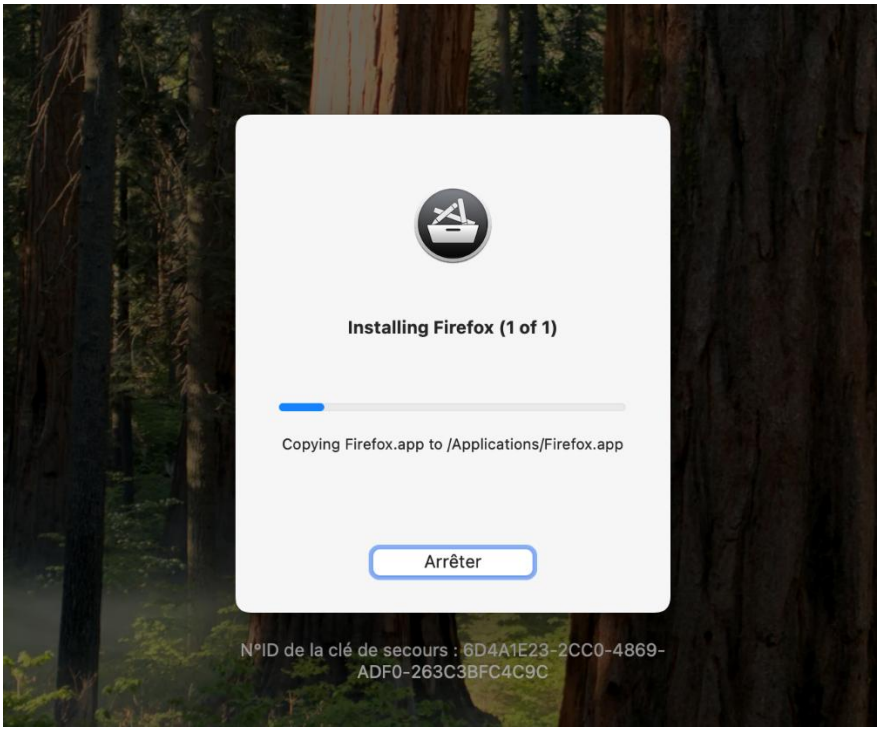
2) Mise en production d'applications



pouvons le voir ci-dessous.



Si un utilisateur ne met pas à jour l'application concerné malgré les pop-ups, après la date que nous avons définie ci-dessus, la mise à jour de l'application sera forcée comme nous pouvons le voir ci-dessous avec Firefox par exemple.



Grâce à ces différentes étapes, des applications ont été mises à jour via le centre de gestion des logiciels, celui-ci est en production et fonctionnel.

k) Automatisation ajout de paquet

Maintenant que nous avons réalisé la mise en production du centre de gestion des logiciels, nous allons voir comment automatiser la mise à jour de nos paquets dans Munki Admin car il n'est pas forcément pratique de devoir mettre constamment à jour nos paquets manuellement.

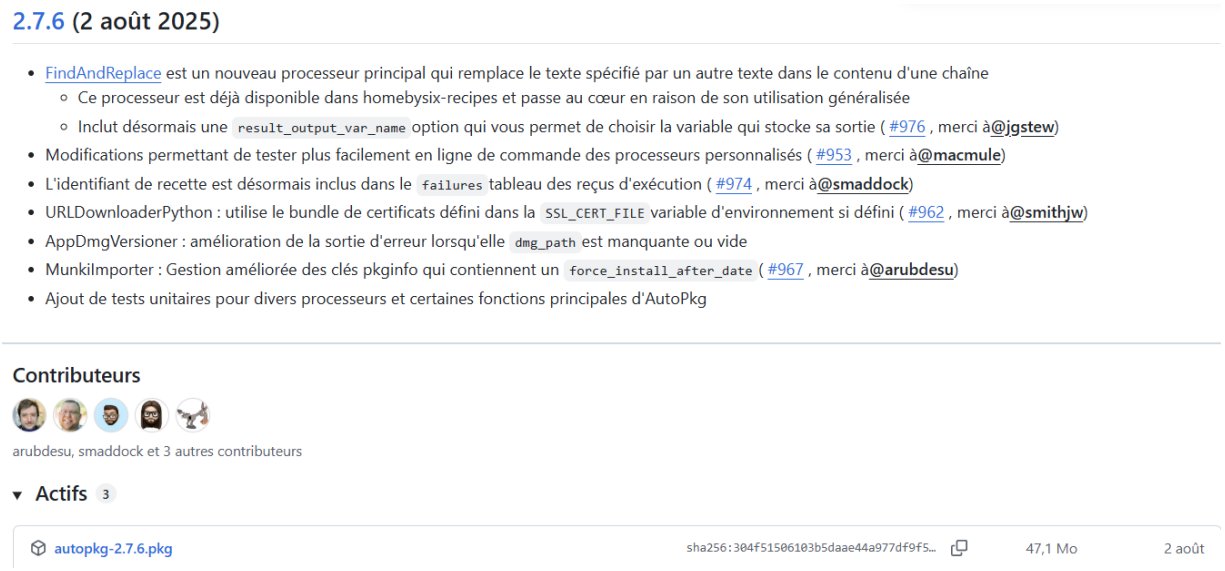
Dans cette activité, nous allons uniquement voir la mise en place de cette automatisation via Autopkg et des recipes.

Tout d'abord Autopkg est un Framework d'automatisation pour le packaging et la distribution de logiciels MacOs, orienté vers les tâches que l'on effectuerait normalement manuellement pour préparer des logiciels tiers pour un déploiement de masse comme Munki par exemple.

Ensuite, une "recipe" (recette) est un fichier de description utilisé par Autopkg pour automatiser, de bout en bout, l'intégration d'un logiciel dans Munki via une chaîne d'étapes que nous allons voir ci-dessous.

Dans un premier temps, nous allons installer Autopkg, pour cela, nous pouvons télécharger le .pkg de la dernière version officielle sur le GitHub de Autopkg avec le lien ci-dessous.

https://github.com/autopkg/autopkg/releases/



2.7.6 (2 août 2025)

- FindAndReplace est un nouveau processeur principal qui remplace le texte spécifié par un autre texte dans le contenu d'une chaîne
  - Ce processeur est déjà disponible dans homebysix-recipes et passe au cœur en raison de son utilisation généralisée
  - Inclut désormais une `result_output_var_name` option qui vous permet de choisir la variable qui stocke sa sortie (#976 , merci à**@jgstew**)
- Modifications permettant de tester plus facilement en ligne de commande des processeurs personnalisés (#953 , merci à**@macmule**)
- L'identifiant de recette est désormais inclus dans le `failures` tableau des reçus d'exécution (#974 , merci à**@smaddock**)
- URLDownloaderPython : utilise le bundle de certificats défini dans la `SSL_CERT_FILE` variable d'environnement si défini (#962 , merci à**@smithjw**)
- AppDmgVersioner : amélioration de la sortie d'erreur lorsqu'elle `dmg_path` est manquante ou vide
- MunkiImporter : Gestion améliorée des clés pkginfo qui contiennent un `force_install_after_date` (#967 , merci à**@arubdesu**)
- Ajout de tests unitaires pour divers processeurs et certaines fonctions principales d'AutoPkg

Contributeurs

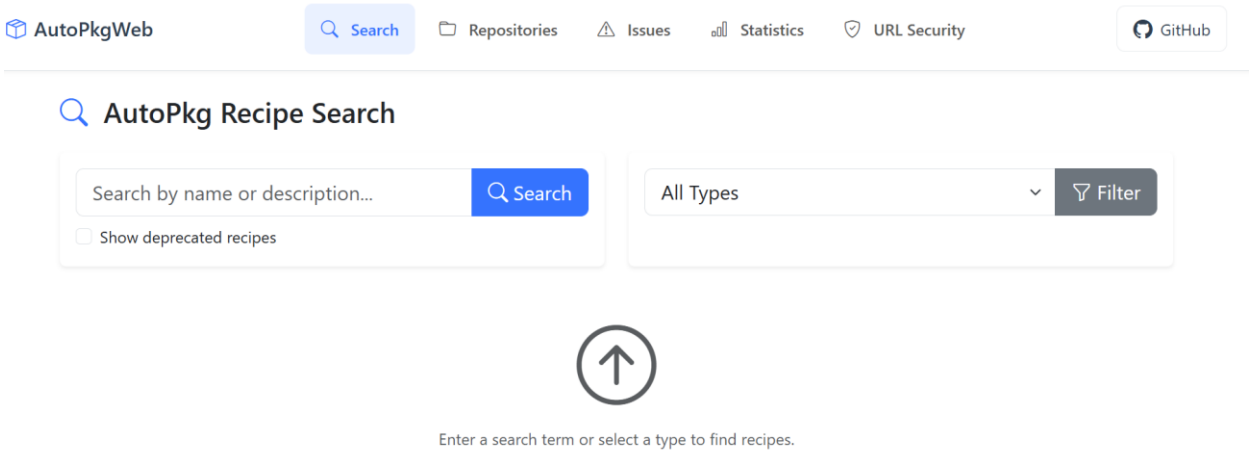
arubdesu, smaddock et 3 autres contributeurs

Actifs 3

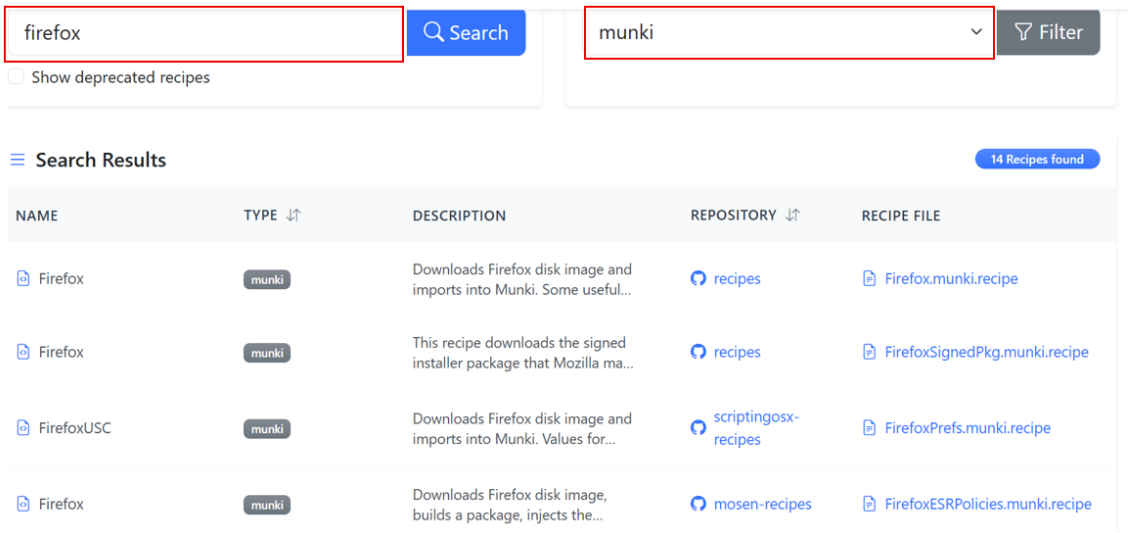
autopkg-2.7.6.pkg sha256:304f5150610 3b5daae44a977df9f5… 47,1 Mo 2 août

Après avoir réalisé l'installation de Autopkg, nous allons installer notre première recipe, pour cela, nous pouvons rechercher la recipe que nous voulons installer sur le site officiel de autopkg, au lien ci-dessous.

https://autopkgweb.com/

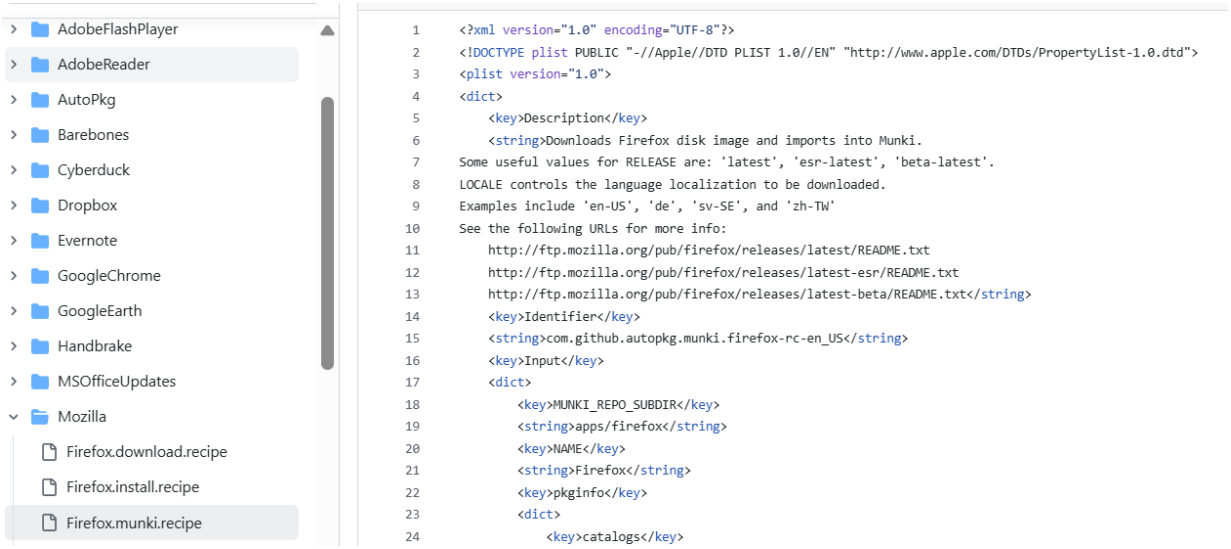


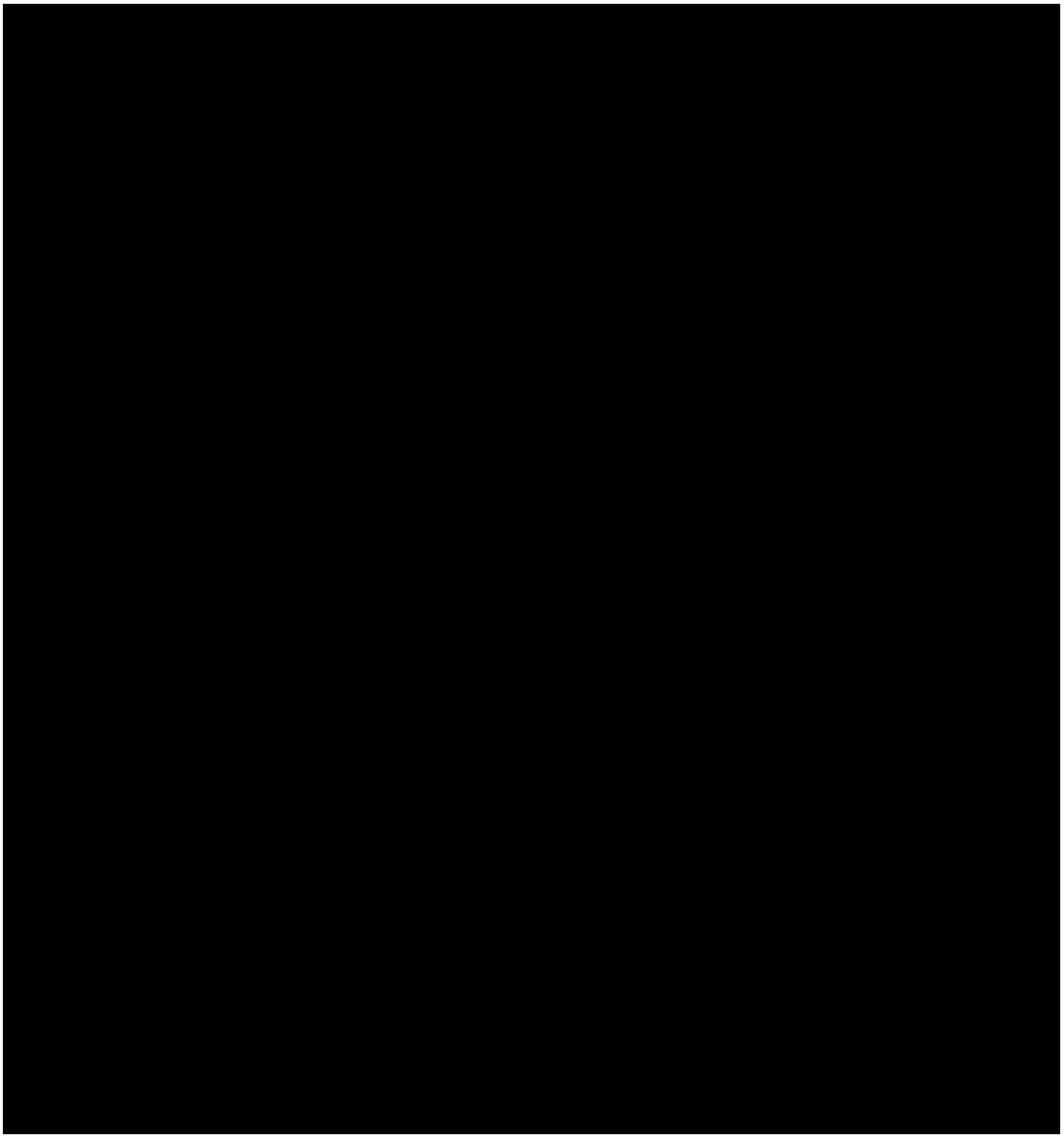
Dans notre exemple, nous allons utiliser une recipe pour Firefox, pour cela, il faut rechercher avec le nom Firefox et mettre le type Munki comme nous pouvons le voir ci-dessous.



Nous avons différentes recipes qui sont proposées, nous allons choisir la première qui correspond au recipe Firefox officiel, pour cela, il faut cliquer sur le lien correspondant au « RECIPE FILE ».

Ce lien va nous emmener sur le fichier « Firefox.munki.recipe » qu'il faut télécharger, cette recette correspond au script qui va nous permettre d'ajouter l'application dans Munki. Le script qui va nous permettre de télécharger l'application est « Firefox.download.recipe » mais nous n'avons pas besoin de le télécharger car le script d'ajout dans Munki va taper sur ce script de téléchargement.





Maintenant que la recipe est configurée, nous pouvons essayer de tester le bon fonctionnement de cette première recette, pour cela, il faut utiliser la commande ci-dessous.

```
autopkg run Firefox.munki.recipe
```

Si nous vérifions dans Munki Admin, nous pouvons voir que le package Firefox a été ajouté.

Enfin, afin d'automatiser une fois par semaine le lancement des recettes, nous pouvons créer un cron. Un cron correspond à une tâche planifiée qui nous permet d'exécuter automatiquement une commande (ou un script). Pour cela, il faut utiliser la commande ci-dessous puis ajouter notre cron. Dans notre cas, nous allons exécuter le script Firefox.munki.recipe avec la commande autopkg puis envoyer le retour de la commande dans un fichier de log.

```
crontab -e
```

```
0 2 * * 2 /usr/local/bin/autopkg run firefox.munki.recipe >> /var/log/autopkg_firefox.log 2>&1
~
~
~
~
~
```

Pour terminer, afin de vérifier si la crontab est correctement configurée, nous pouvons utiliser la commande ci-dessous afin de voir si notre tâche planifiée a bien été ajoutée.

```
crontab -l
```

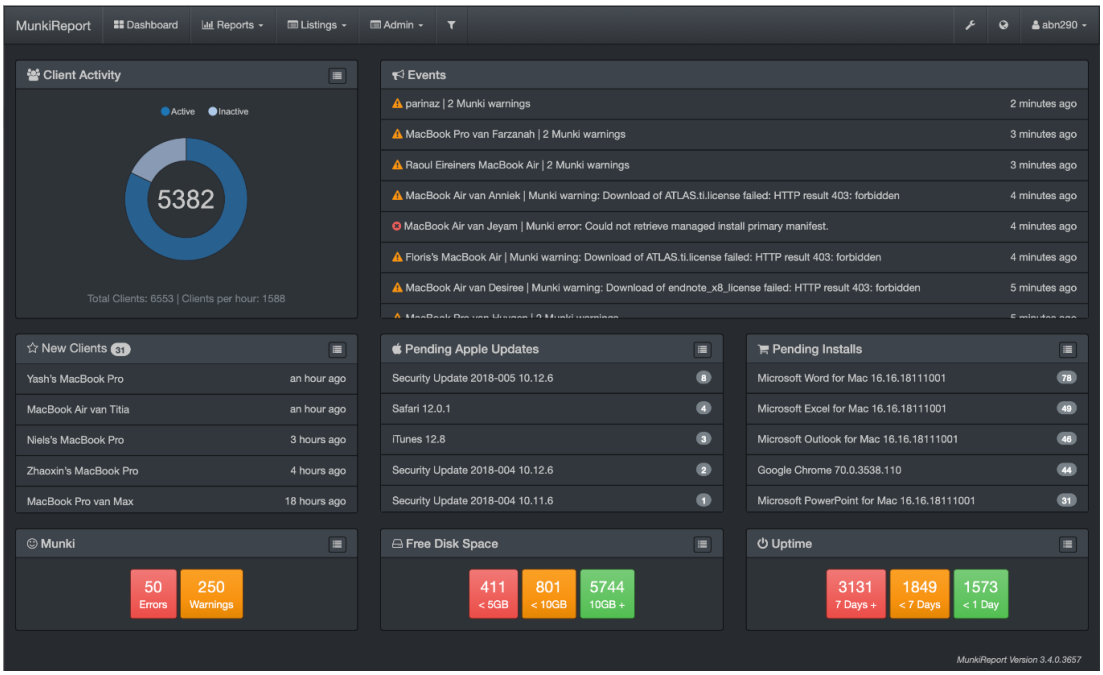
```
% crontab -l
0 2 * * 2 /usr/local/bin/autopkg run firefox.munki.recipe >> /var/log/autopkg_firefox.log 2>&1
%
```

Une fois ces différentes étapes réalisées, l'automatisation de Munki est terminée.

I) Axes d'améliorations

Pour ce projet, je pense qu'il y a différents axes d'amélioration. Tout d'abord, nous aurions pu implémenter une solution de supervision pour suivre l'état des différents déploiements, tel que MunkiReport-PHP. Cela nous aurait permis de savoir sur quels postes l'outil ne fonctionnait pas forcément.

https://github.com/munkireport/munkireport-php



Afin de nous faciliter la tâche, nous aurions également pu faire en sorte d'avoir des notifications Teams dès qu'un paquet devait être mis à jour. Cela nous aurait permis de détecter s'il y avait une erreur au niveau de l'ajout de certains packages.

m) Conclusion

En conclusion, j'ai trouvé que cette activité m'a permis d'apprendre de nombreuses choses. J'ai pu réaliser différentes tâches qui m'ont permis d'acquérir de nouvelles compétences sur le déploiement d'applications et sur l'environnement Mac.

Au niveau des difficultés rencontrées, j'ai pu rencontrer des difficultés lors du premier déploiement d'applications via Munki. Il m'a été compliqué de savoir si les applications étaient correctement à jour et si le Centre de Gestion des Logiciels fonctionnait correctement lors du premier test en production.

J'ai également rencontré des difficultés lors de l'automatisation des recipes dans Munki. Les applications n'étaient pas systématiquement importées correctement dans le catalogue. Pour résoudre ce problème de manière efficace, j'aurais dû approfondir mes recherches en consultant la documentation officielle d'AutoPkg et de Munki, analyser les logs d'exécution pour identifier les erreurs précises, et solliciter la communauté Munki Admin pour bénéficier de retours d'expérience. Cette difficulté m'a permis de comprendre l'importance d'une phase de recherche approfondie et de tests rigoureux.

Malgré ces difficultés, je pense que nous avons répondu à l'objectif principal du projet qui était de trouver une solution adaptée au parc informatique pour le déploiement d'applications. Je suis satisfait de ce qui a pu être réalisé sur ce projet.

Pour terminer, cette activité m'a permis de valider plusieurs compétences du référentiel BTS SIO SISR :

- Gérer le patrimoine informatique
- Répondre aux incidents et aux demandes d'assistance et d'évolution
- Travailler en mode projet
- Mettre à disposition des utilisateurs un service informatique

Projet 2 : Serveur de Sauvegarde

a) Problématique

b) Veille effectuée avant le projet

Avant de réaliser le projet, nous avions fait une veille sur les différentes solutions que l'on avait à notre disposition et sur les besoins matériels que nous avions par rapport à notre infrastructure.

Pour cela, nous avons réalisé une étude de faisabilité, dans cette étude, nous avons comparé différentes solutions, Bareos, Veeam ainsi que Amanda en fonction de ces trois critères :

- Comment la solution pouvait répondre à nos besoins de sauvegarde ?
- Quels étaient les besoins matériels de la solution ?
- Quel était le coût de la solution ?

Tout d'abord pour le premier critère par rapport à nos besoins de sauvegarde, nous avons regardé la documentation de chaque solution et nous avons pu remplir le tableau Excel que nous voyions ci-dessous avec les différentes fonctionnalités de chaque solution. Sur ce tableau, nous pouvions voir que la solution Bareos répond plus à nos critères.

Ensuite, pour le second et le troisième critère, par rapport aux besoins matériels et au coût de chaque solution, nous avons réalisé une comparaison entre les différentes solutions.

Nous pouvions voir sur ce tableau que les besoins matériels restaient globalement les mêmes, mais que les coûts n'étaient pas forcément pareils. Pour les solutions Veeam Backup et Amanda, il y a des coûts de licences.

Grâce à ces différentes comparaisons et à la veille que nous avons réalisée sur les documentations de chaque solution, nous avons conclu que la solution Bareos correspondait le plus à nos besoins. C'est donc cette solution a été mis en place.

c) Planification

Le projet pour la mise en place de la solution Bareos a été séparé en diverses étapes, ci-dessous les différentes étapes du projet :

1) Présentation du Projet (1 semaine)

Cette partie correspondait à toute la présentation du projet à mon n+2 (Deux personnes au-dessus de moi dans l'organigramme).

2) Etude de la faisabilité du Projet (1 semaine)

Dans cette partie nous avons étudié ce qui pouvait bloquer le projet ainsi que nos besoins matériels et immatériels.

3) Mise en place et configuration du matériel (1 semaine)

Cette partie correspondait à l'ensemble de la configuration que nous avons réalisé sur la machine physique, que ce soit la mise en baie, la mise en réseau, la mise en place des RAID ou encore l'installation du système d'exploitation.

4) Installation et test de la solution Bareos (10 semaines)

Cette étape correspondait à toute l'installation de la solution et aux différents tests que nous avons réalisés afin de savoir si Bareos pouvait être intégré dans notre infrastructure.

5) Documentation (1 semaine)

Cette partie correspondait à toute la documentation à faire pour les personnes qui sont vouées à utiliser l'outil de sauvegarde.

6) Mise en Production de la solution

La mise en production correspondait aux actions que nous avons effectuées pour mettre en place Bareos sur l'ensemble des machines qu'il y avait au siège social de l'entreprise.

7) Clôture du Projet

Pour terminer, la clôture du projet correspondait à la fin du projet, celle-ci est signée lorsque tout a été mis en place et que cela correspond complètement aux attentes et à la problématique.

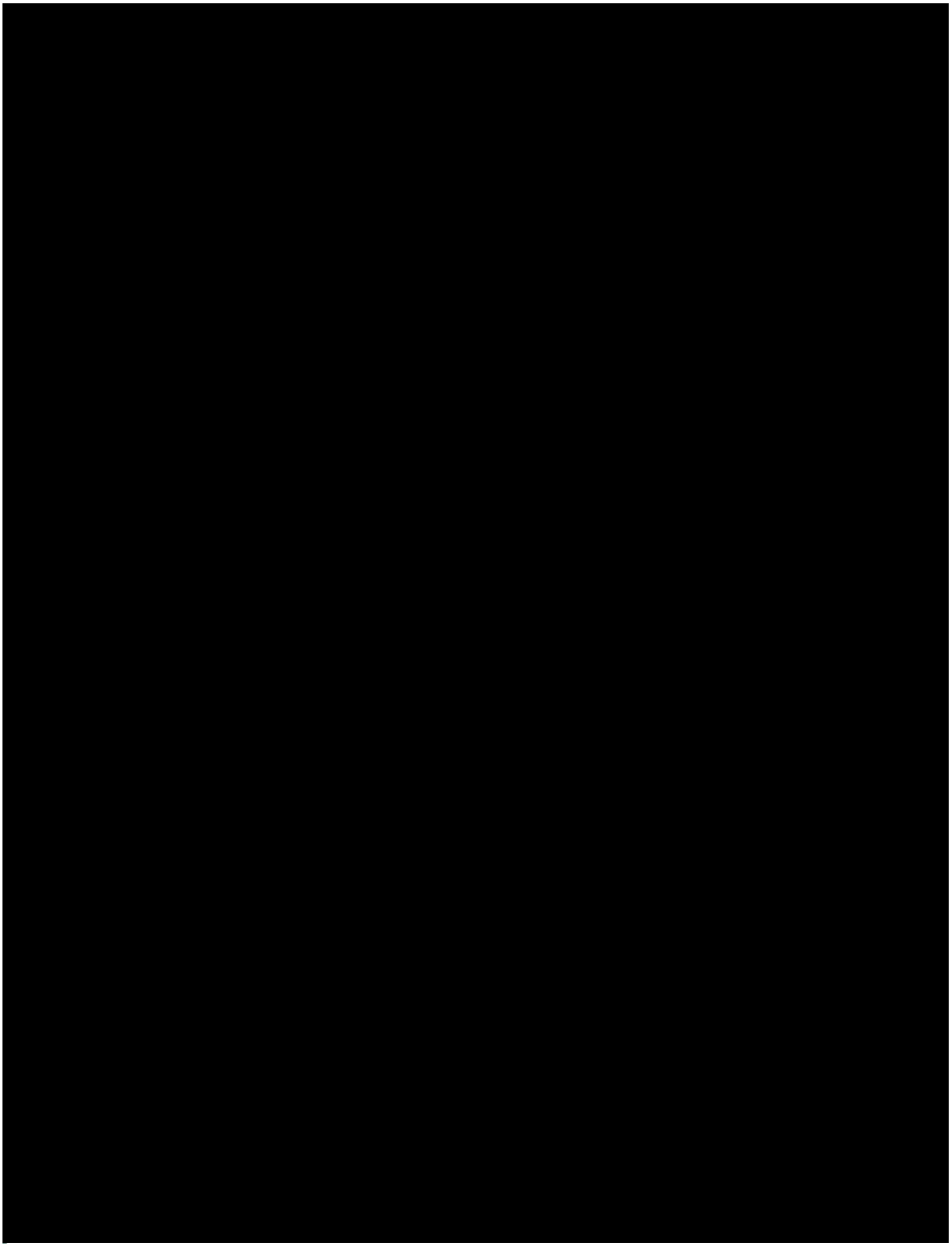
Pour terminer cette partie planification, ci-dessous les tâches du diagramme de GANTT ainsi que le diagramme correspondant, celui-ci permettait de voir et de comprendre l'ensemble des tâches du projet.

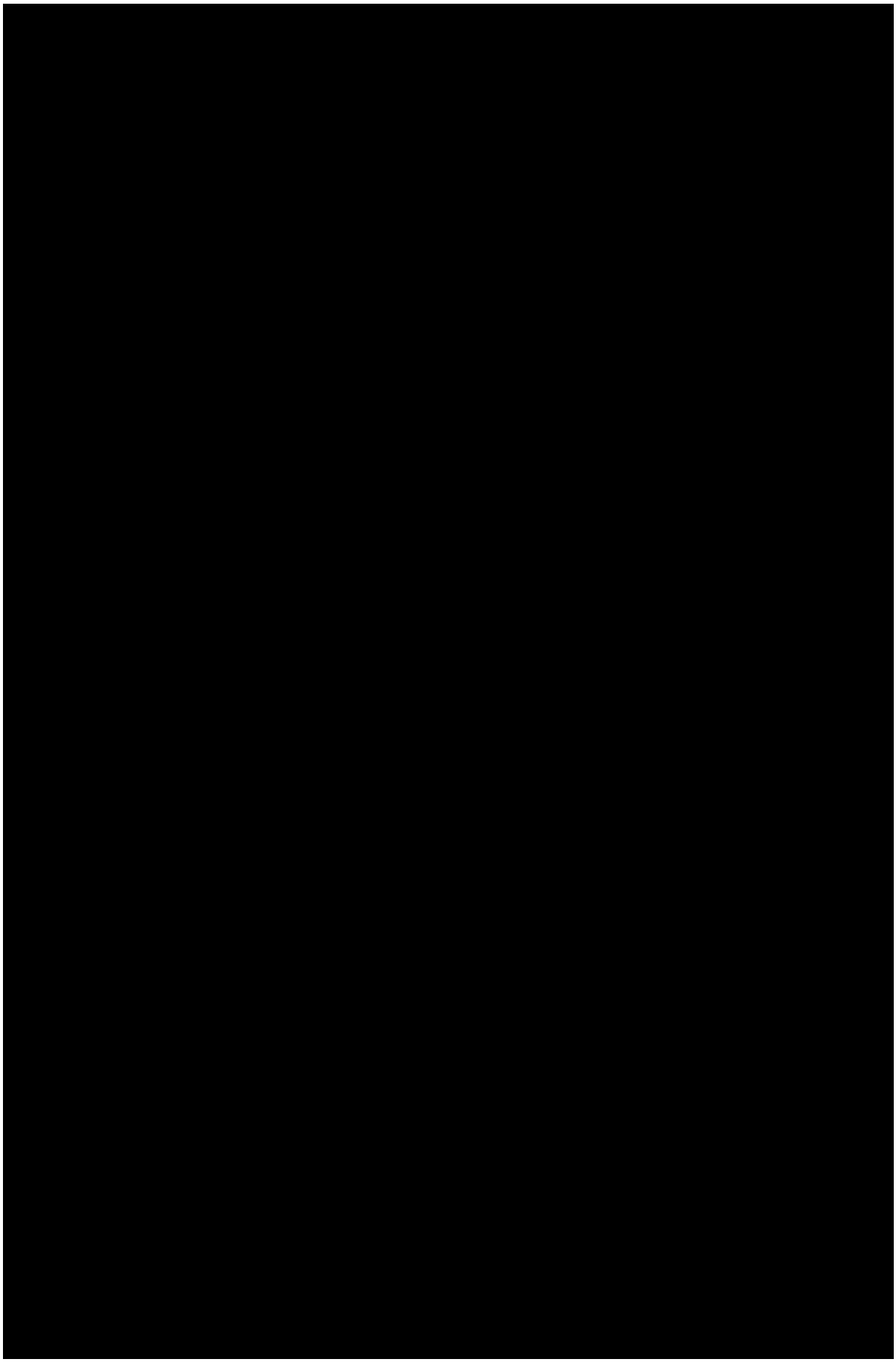
[redacted]

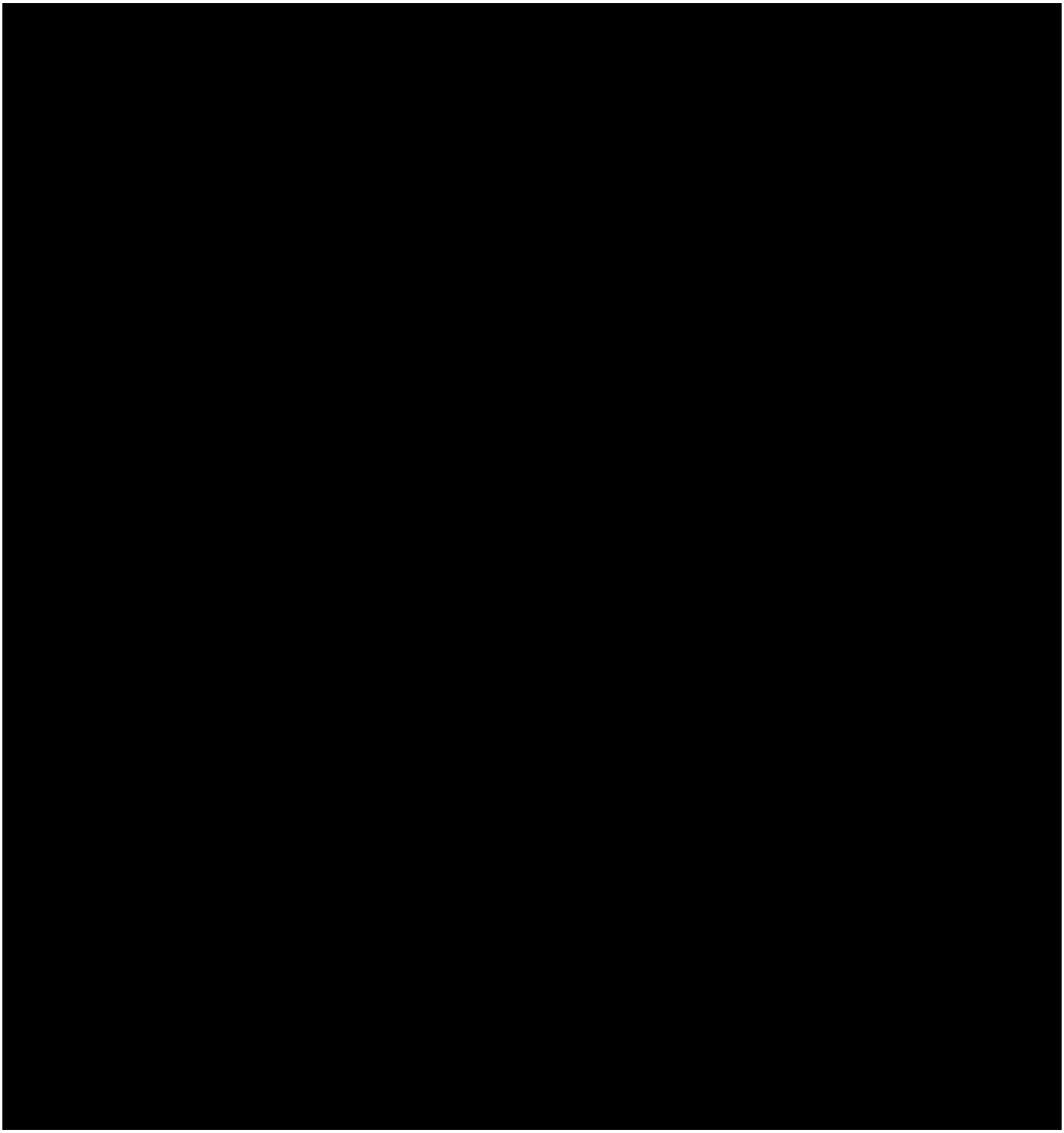
d) Liste du matériel à disposition

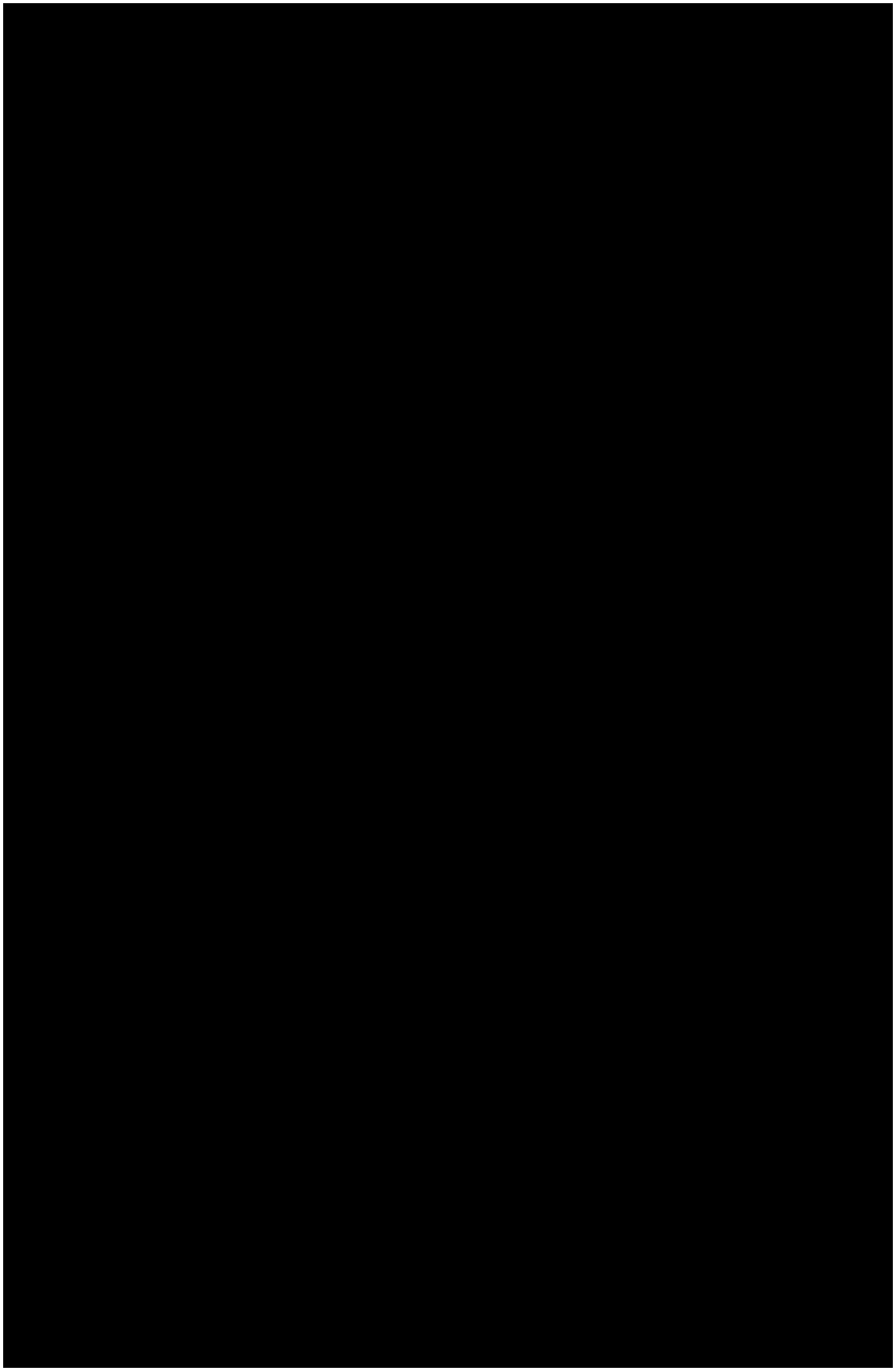
e) Configuration du matériel

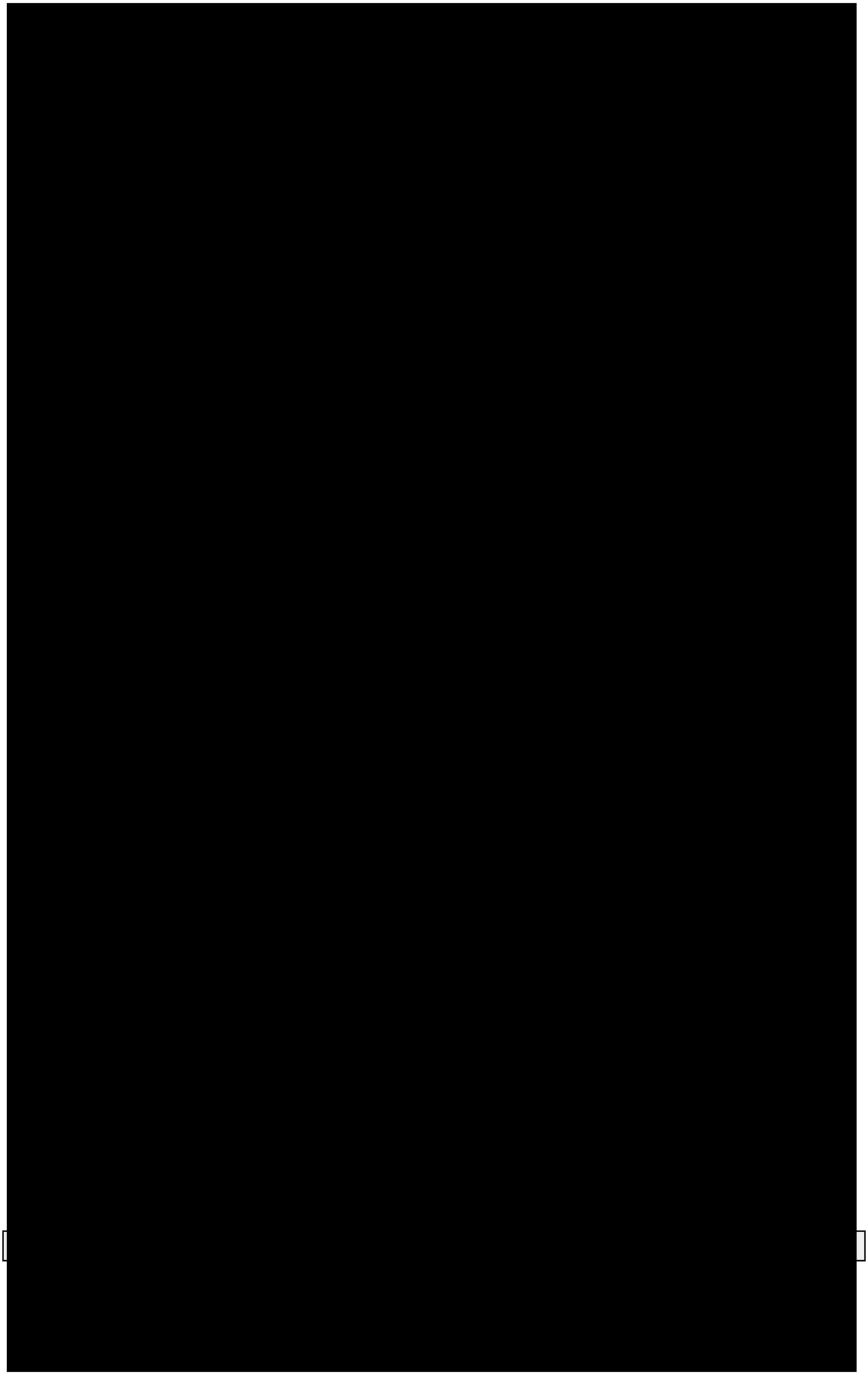
Dès réception du matériel, nous avons commencé par installer le serveur physique dans la baie. Pour cela, nous l'avons d'abord fixé sur des rails, comme le montre l'image ci-dessous.











f) Installation serveur Bareos

Dans cette partie consacrée à l'installation, nous allons mettre en place la solution Bareos afin de réaliser des sauvegardes et des restaurations sur différents types de systèmes : Debian, switchs, routeurs ISR et bases de données MariaDB.

Mais tout d'abord voyons différents concepts de Bareos. Globalement, bareos va utiliser trois services, bareos-fd, bareos-dir et bareos-sd.

Le bareos-dir (Director) est le programme de contrôle central de tous les autres démons. Il planifie et supervise toutes les opérations de sauvegarde, de restauration, de vérification et d'archivage.

Le bareos-fd (Filedaemon) est un programme qui doit être installé sur chaque machine cliente à sauvegarder. À la demande du directeur Bareos, il localise les fichiers à sauvegarder et les envoie au démon de stockage Bareos (bareos-sd).

Le bareos-sd (Storage Daemon) est chargé, à la demande du directeur Bareos, de recevoir les données du démon de fichiers Bareos et d'enregistrer les attributs et les données des fichiers sur les supports ou volumes de sauvegarde physiques.

Ensuite, Bareos va utiliser différents fichiers pour les sauvegardes et les restaurations.

Un FileSet est une ressource contenue dans un fichier de configuration qui définit les fichiers à sauvegarder. Il comprend une liste de fichiers ou de répertoires inclus, une liste de fichiers exclus et le mode de stockage des fichiers.

Une Job Bareos est une ressource de configuration qui définit les opérations que Bareos doit effectuer pour sauvegarder ou restaurer un client spécifique.

Maintenant que le serveur est complètement configuré, nous allons installer la solution Bareos. Pour cela, il faut tout d'abord mettre à jour ses paquets comme ci-dessous.

```
apt update && apt upgrade -y
```

```
apt upgrade && apt update -y
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Hit:1                                                InRelease
All packages are up to date.
```

Ensuite, il faut télécharger un script Bash sur le site officiel de Bareos qui va nous permettre d'ajouter les sources pour les dépôts Bareos comme ci-dessous où l'on télécharge le fichier add_bareos_repositories.sh avec la commande wget.

```
wget
https://download.bareos.org/current/Debian_13/add_bareos_repositories.sh
```

```
wget https://download.bareos.org/current/Debian_13/add_bareos_repositories.sh
--2025-10-18 15:17:19--  https://download.bareos.org/current/Debian_13/add_bareos_repositories.sh
Resolving download.bareos.org (download.bareos.org)... 2a03:4000:15:27f::1, 185.170.114.121
Connecting to download.bareos.org (download.bareos.org)|2a03:4000:15:27f::1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2874 (2.8K) [text/plain]
Saving to: 'add_bareos_repositories.sh'

add_bareos_repositories.sh      100%[=============================================>]   2.81K  --.-KB/s    in 0s

2025-10-18 15:17:19 (45.3 MB/s) - 'add_bareos_repositories.sh' saved [2874/2874]
```

Après avoir fait cela, il faut lancer le script Bash qui a été téléchargé, pour cela il faut utiliser la commande ci-dessous, le script va ajouter les sources Bareos.

```
sh add_bareos_repositories.sh
```

```
sh add_bareos_repositories.sh
Repository https://download.bareos.org/current/Debian_13 successfully added.
```

Maintenant, les sources des dépôts Bareos ont normalement été ajoutées, nous pouvons le vérifier en regardant le contenu du fichier contenant les sources comme ci-dessous.

```
cat /etc/apt/sources.list.d/bareos.sources
```

```
cat /etc/apt/sources.list.d/bareos.sources
Types: deb deb-src
URIs: https://download.bareos.org/current/Debian_13
Suites: /
Architectures: amd64
Signed-By: /etc/apt/keyrings/bareos-experimental.gpg
```

Une fois que l'on est sûr que les sources ont été ajoutées, nous pouvons utiliser la commande ci-dessous afin de les mettre à jour.

```
apt update
```

```
apt update
Get:1 https://download.bareos.org/current/Debian_13  InRelease [1,890 B]
Hit:2
Get:3 https://download.bareos.org/current/Debian_13  Sources [2,132 B]
Get:4 https://download.bareos.org/current/Debian_13  Packages [8,504 B]
Fetched 12.5 kB in 0s (33.5 kB/s)
All packages are up to date.
```

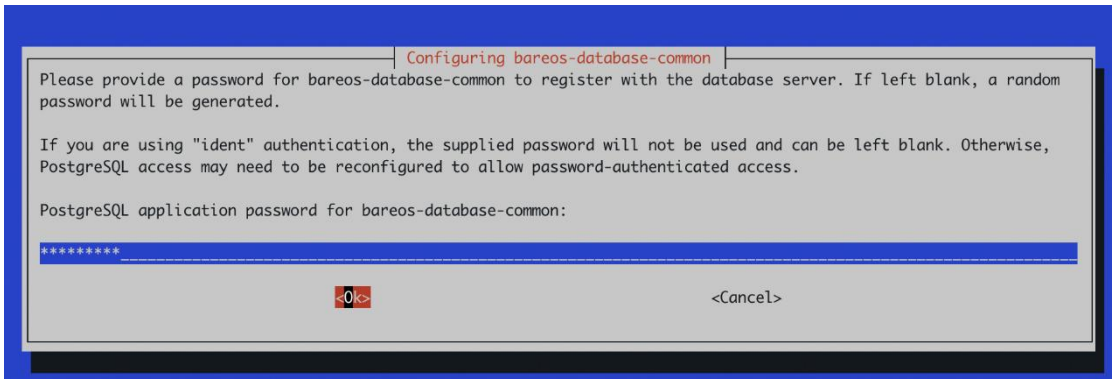
Nos dépôts sont mis à jour grâce à la commande précédente, maintenant, nous pouvons installer tous les paquets dont nous allons avoir besoin côté serveur.

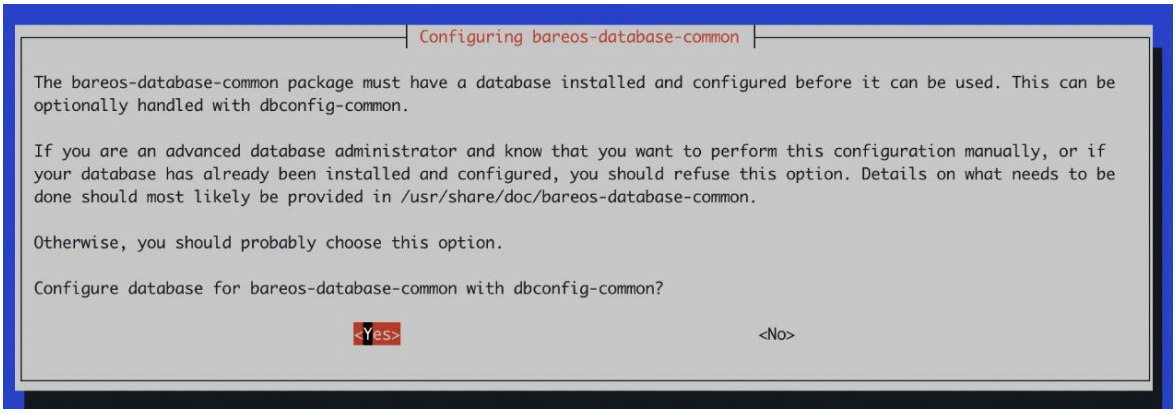
```
apt install bareos
```

```
apt install bareos
Installing:
  bareos

Installing dependencies:
  bareos-bconsole             bareos-filedaemon  exim4-daemon-light      liblockfile1  perl-modules-5.40
  bareos-client               bareos-storage     libevent-2.1-7t64       liblzo2-2     postgresql-client
  bareos-common               bsd-mailx          libfile-fcntllock-perl  libnsl2       postgresql-client-17
  bareos-database-common      dbconfig-common    libgdbm-compat4t64      libperl5.40   postgresql-client-common
  bareos-database-postgresql  dbconfig-pgsql     libgnutls-dane0t64      libpq5
  bareos-database-tools       exim4-base         libidn12                libunbound8
  bareos-director             exim4-config       liblockfile-bin         perl
```

Lorsqu'il a terminé d'installer le paquet « bareos », la page ci-dessous s'affiche, il faut sélectionner « Yes » afin qu'il configure automatiquement la base de données.

Une nouvelle page va s'afficher, sur celle-ci il faut renseigner le mot de passe qui va être utilisé par l'utilisateur PostgreSQL.





Après avoir renseigné le mot de passe, l'installation de Bareos est terminée mais il faut activer les différents services utilisés par l'outil comme nous pouvons le voir ci-dessous.

```
systemctl enable --now bareos-director.service
systemctl enable --now bareos-storage.service
systemctl enable --now bareos-filedaemon.service
```

systemctl enable --now bareos-director.service
systemctl enable --now bareos-storage.service
systemctl enable --now bareos-filedaemon.service

Pour terminer, il faut vérifier si les services ont correctement démarré, nous allons vérifier les services bareos-director et bareos-storage car ces deux services correspondent à la partie serveur.

```
systemctl status bareos-dir
systemctl status bareos-sd
```

Grâce à ces différentes actions, Bareos est correctement installé sur le serveur.

g) Configuration Debian

1) Configuration bareos-fd

Lorsque la machine Debian est installée, nous pouvons ajouter les sources Bareos sur le client concerné, pour cela, il est possible de télécharger les sources via la commande wget https://download.bareos.org/current/ comme ci-dessous pour une Debian 13.

```
wget https://download.bareos.org/current/Debian_13/add_bareos_repositories.sh
```

Ensuite, il faut lancer le script Shell pour ajouter les sources Bareos comme nous pouvons le voir ci-dessous.

```
sh ~/add_bareos_repositories.sh
```

```
root@Test-Deb-Bareos:~# sh ~/add_bareos_repositories.sh
Repository https://download.bareos.org/current/Debian_13 successfully added.
root@Test-Deb-Bareos:~#
```

Une fois cela fait, les sources Bareos sont ajoutées dans le fichier /etc/apt/sources.list.d/bareos.sources, afin de pouvoir télécharger le paquet Bareos pour le client, nous allons devoir mettre à jour nos sources avec la commande « apt update ».

```
apt update
```

```
root@Test-Deb-Bareos:~# apt update
Get:1 https://download.bareos.org/current/Debian_13  InRelease [1,890 B]
Hit:2 http://                                        /debian trixie InRelease
Get:3 https://download.bareos.org/current/Debian_13  Sources [2,136 B]
Get:4 https://download.bareos.org/current/Debian_13  Packages [8,512 B]
Fetched 12.5 kB in 0s (26.4 kB/s)
All packages are up to date.
```

Après, nous pouvons installer le client Bareos (Filedaemon) comme nous pouvons le voir ci-dessous.

```
apt install bareos-filedaemon -y
```

```
root@Test-Deb-Bareos:~# apt install bareos-filedaemon -y
Installing:
  bareos-filedaemon

Installing dependencies:
  bareos-common  liblzo2-2

Summary:
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading: 0
  Download size: 1,316 kB
  Space needed: 4,591 kB / 15.5 GB available

Get:1                                  /debian trixie/main amd64 liblzo2-2 amd64 2.10-3+b1 [55.1 kB]
Get:2 https://download.bareos.org/current/Debian_13  bareos-common 24.0.7~pre6.6b71fc81e-107 [1,012 kB]
Get:3 https://download.bareos.org/current/Debian_13  bareos-filedaemon 24.0.7~pre6.6b71fc81e-107 [249 kB]
Fetched 1,316 kB in 4s (361 kB/s)
Selecting previously unselected package liblzo2-2:amd64.
(Reading database ... 31604 files and directories currently installed.)
Preparing to unpack .../liblzo2-2_2.10-3+b1_amd64.deb ...
Unpacking liblzo2-2:amd64 (2.10-3+b1) ...
Selecting previously unselected package bareos-common.
Preparing to unpack .../bareos-common_24.0.7~pre6.6b71fc81e-107_amd64.deb ...
Unpacking bareos-common (24.0.7~pre6.6b71fc81e-107) ...
Selecting previously unselected package bareos-filedaemon.
Preparing to unpack .../bareos-filedaemon_24.0.7~pre6.6b71fc81e-107_amd64.deb ...
Unpacking bareos-filedaemon (24.0.7~pre6.6b71fc81e-107) ...
Setting up liblzo2-2:amd64 (2.10-3+b1) ...
Setting up bareos-common (24.0.7~pre6.6b71fc81e-107) ...
Setting up bareos-filedaemon (24.0.7~pre6.6b71fc81e-107) ...
Info: replacing 'XXX_REPLACE_WITH_LOCAL_HOSTNAME_XXX' with 'Test-Deb-Bareos' in /etc/bareos/bareos-fd.d/client/myself.conf
Info: replacing 'XXX_REPLACE_WITH_CLIENT_PASSWORD_XXX' in /etc/bareos/bareos-fd.d/director/bareos-dir.conf
Info: replacing 'XXX_REPLACE_WITH_CLIENT_MONITOR_PASSWORD_XXX' in /etc/bareos/bareos-fd.d/director/bareos-mon.conf
Created symlink '/etc/systemd/system/bareos-fd.service' → '/usr/lib/systemd/system/bareos-filedaemon.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/bareos-filedaemon.service' → '/usr/lib/systemd/system/bareos-filed
aemon.service'.
Processing triggers for libc-bin (2.41-12) ...
Processing triggers for man-db (2.13.1-1) ...
root@Test-Deb-Bareos:~#
```

Une fois que le Filedaemon est installé, un dossier bareos est créé dans /etc/bareos, pour que le client soit correctement configuré, il faut modifier le fichier /etc/bareos/bareos-fd.d/client/myself.conf afin de changer le nom du client de hostname-fd à juste hostname comme nous pouvons le voir ci-dessous. Nous passons de Test-Deb-Bareos-fd à juste Test-Deb-Bareos.

```
vim /etc/bareos/bareos-fd.d/client/myself.conf
```

```
Client {
  Name = Test-Deb-Bareos

  # remove comment from "Plugin Directory" to load plugins from specified directory.
  # if "Plugin Names" is defined, only the specified plugins will be loaded,
  # otherwise all filedaemon plugins (*-fd.so) from the "Plugin Directory".
  #
  # Plugin Directory = "/usr/lib/bareos/plugins"
  # Plugin Names = ""
}
```

Ensuite, il faut configurer un second fichier, pour cela, nous allons modifier le fichier /etc/bareos/bareos-fd.d/director/bareos-dir.conf et nous pouvons soit garder le mot de passe présent et le noter afin de le configurer sur le serveur dans les étapes suivantes ou alors le changer avec le mot de passe que l'on veut.

```
vim /etc/bareos/bareos-fd.d/director/bareos-dir.conf
```

```
Director {
  Name = bareos-dir
  Password =
  Description = "Allow the configured Director to access this file daemon."
}
```

Une fois cela fait, nous devons redémarrer le service « bareos-fd » mais avant cela nous pouvons vérifier que l'on n'a pas de problème de configuration Bareos grâce à la commande « bareos-fd -t », ensuite nous redémarrons le service et nous vérifions le statut de celui-ci comme ci-dessous.

```
bareos-fd -t
systemctl restart bareos-fd
systemctl status bareos-fd
```

```
root@Test-Deb-Bareos:~# bareos-fd -t
root@Test-Deb-Bareos:~# systemctl restart bareos-fd
root@Test-Deb-Bareos:~# systemctl status bareos-fd
● bareos-filedaemon.service - Bareos File Daemon service
     Loaded: loaded (/usr/lib/systemd/system/bareos-filedaemon.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-10-24 15:21:37 CEST; 2min 7s ago
 Invocation: 878702828f3f42d9818d9fdf963bc056
       Docs: man:bareos-fd(8)
   Main PID: 3344 (bareos-fd)
      Tasks: 2 (limit: 2236)
     Memory: 1.7M (peak: 2.1M)
        CPU: 32ms
     CGroup: /system.slice/bareos-filedaemon.service
             └─3344 /usr/sbin/bareos-fd -f

Oct 24 15:21:37 Test-Deb-Bareos systemd[1]: Started bareos-filedaemon.service - Bareos File Daemon service.
```

Le service est actif, ce qui veut dire que le client Bareos est fonctionnel.

2) Configuration bareos-dir

Après avoir fait la configuration côté client, nous pouvons réaliser la configuration côté serveur.

Dans un premier temps, il faut créer le client sur lequel a été installé le paquet Bareos-fd sur le serveur Bareos, pour cela, nous devons, sur le serveur aller dans la console Bareos avec la commande « bconsole ».

```
bconsole
```

```
# bconsole
Connecting to Director localhost:9101
 Encryption: TLS_CHACHA20_POLY1305_SHA256 TLSv1.3
1000 OK: bareos-dir Version: 24.0.8~pre8.dbdd292a4 (13 November 2025)
Bareos community build (UNSUPPORTED).
Get professional support from https://www.bareos.com
You are connected using the default console

Enter a period (.) to cancel a command.
*
```

Ensuite, nous devons ajouter le client, pour cela il suffit de taper la commande ci-dessous dans la console Bareos. Il faut renseigner le mot de passe récupéré sur le client Bareos puis redémarrer le service bareos-dir.

```
configure add client name=Test-Deb-Bareos address= « IP » password=secret
reload
```

```
*configure add client name=Test-Deb-Bareos address=            password=secret
Exported resource file "/etc/bareos/bareos-dir-export/client/Test-Deb-Bareos/b
Director {
  Name = "bareos-dir"
  Password = "[md5]5ebe2294ecd0e0f08eab7690d2a6ee69"
}
Created resource config file "/etc/bareos/bareos-dir.d/client/Test-Deb-Bareos.
Client {
  Name = "Test-Deb-Bareos"
  Address =
  Password = "secret"
}
You have messages.
*reload
reloaded
```

Maintenant que le client est ajouté, nous pouvons tester pour voir si notre serveur communique, pour cela, il faut utiliser la commande « status » dans la bconsole, comme ci-dessous.

```
status client=Test-Deb-Bareos
```

```
*status client=Test-Deb-Bareos
Connecting to Client Test-Deb-Bareos at
Probing client protocol... (result will be saved until config reload)
 Handshake: Immediate TLS, Encryption: TLS_CHACHA20_POLY1305_SHA256 TLSv1.3

Test-Deb-Bareos Version: 24.0.8~pre8.dbdd292a4 (13 November 2025)  Debian GNU/Linux 13 (trixie)
Daemon started 15-Nov-25 10:22. Jobs: run=0 running=0, Bareos community binary
 Sizeof: boffset_t=8 size_t=8 debug=0 trace=0 bwlimit=0kB/s

Running Jobs:
bareos-dir (director) connected at: 15-Nov-25 10:23
No Jobs running.
```

Une fois cela fait, il faut configurer les différents éléments permettant de réaliser la sauvegarde.

Tout d'abord le FileSet : il faut créer un fichier à ce chemin /etc/bareos/bareos-dir.d/fileset/exemple.conf dans lequel nous allons définir ce que nous voulons sauvegarder.

Nous pouvons également définir ce qui ne sera pas sauvegardé, mais tout ce qui n'est pas défini ne sera pas sauvegardé par défaut.

Dans notre cas, nous allons utiliser un FileSet permettant de sauvegarder globalement notre machine Debian, le FileSet LinuxAll comme nous pouvons le voir ci-dessous.

```
# cat LinuxAll.conf
FileSet {
  Name = "LinuxAll"
  Description = "Backup all regular filesystems,
  Include {
    Options {
      Signature = XXH128 # calculate checksum pe
      One FS = No        # change into other fil
      FS Type = btrfs
      FS Type = ext2     # filesystems of given
      FS Type = ext3     # others will be ignore
      FS Type = ext4
      FS Type = reiserfs
      FS Type = jfs
      FS Type = vfat     # UEFI
      FS Type = xfs
      FS Type = zfs
    }
    File = /
  }
  # Things that usually have to be excluded
  # You have to exclude /var/lib/bareos/storage
  # on your bareos server
  Exclude {
    File = /var/lib/bareos
    File = /var/lib/bareos/storage
    File = /proc
    File = /dev
    File = /sys
    File = /tmp
    File = /var/tmp
    File = /.journal
    File = /.fsck
  }
}
```

Ensuite, pour les JobDefs, il faut créer un fichier à ce chemin /etc/bareos/bareos-dir.d/jobdefs/exemple.conf dans lequel nous allons définir le type de sauvegarde que l'on va faire, le FileSet que l'on va utiliser, la priorité ou encore le stockage utilisé.

Il est possible de réutiliser une JobDefs déjà utilisée ou tout simplement d'en recréer une. Ci-dessous, un exemple de JobDefs créée pour notre client Test-Deb-Bareos.

```
# cat Test-Deb-Bareosjob.conf
JobDefs {
  Name = "Test-Deb-Bareosjob"
  Type = Backup
  Level = Incremental
  FileSet = "LinuxAll"                    # selftest fileset
  Schedule = "WeeklyCycle"
  Storage = File
  Messages = Standard
  Pool = Incremental
  Priority = 10
  Write Bootstrap = "/var/lib/bareos/%c.bsr"
  Full Backup Pool = Full                 # write Full Backups into "Full" Pool
  Differential Backup Pool = Differential  # write Diff Backups into "Differential" Pool
  Incremental Backup Pool = Incremental    # write Incr Backups into "Incremental" Pool
}
```

Enfin, nous devons configurer la tâche. Pour cela, il faut créer un fichier à ce chemin /etc/bareos/bareos-dir.d/job/exemple.conf qui va être attribué au client voulu, dans notre cas à la machine Test-Deb-Bareos.

Nous pouvons voir ci-dessous un exemple de configuration pour une tâche Debian.

```
# cat backup-Test-Deb-Bareos.conf
Job {
  Name = "backup-Test-Deb-Bareos"
  JobDefs = "Test-Deb-Bareosjob"
  Client = "Test-Deb-Bareos"
}
```

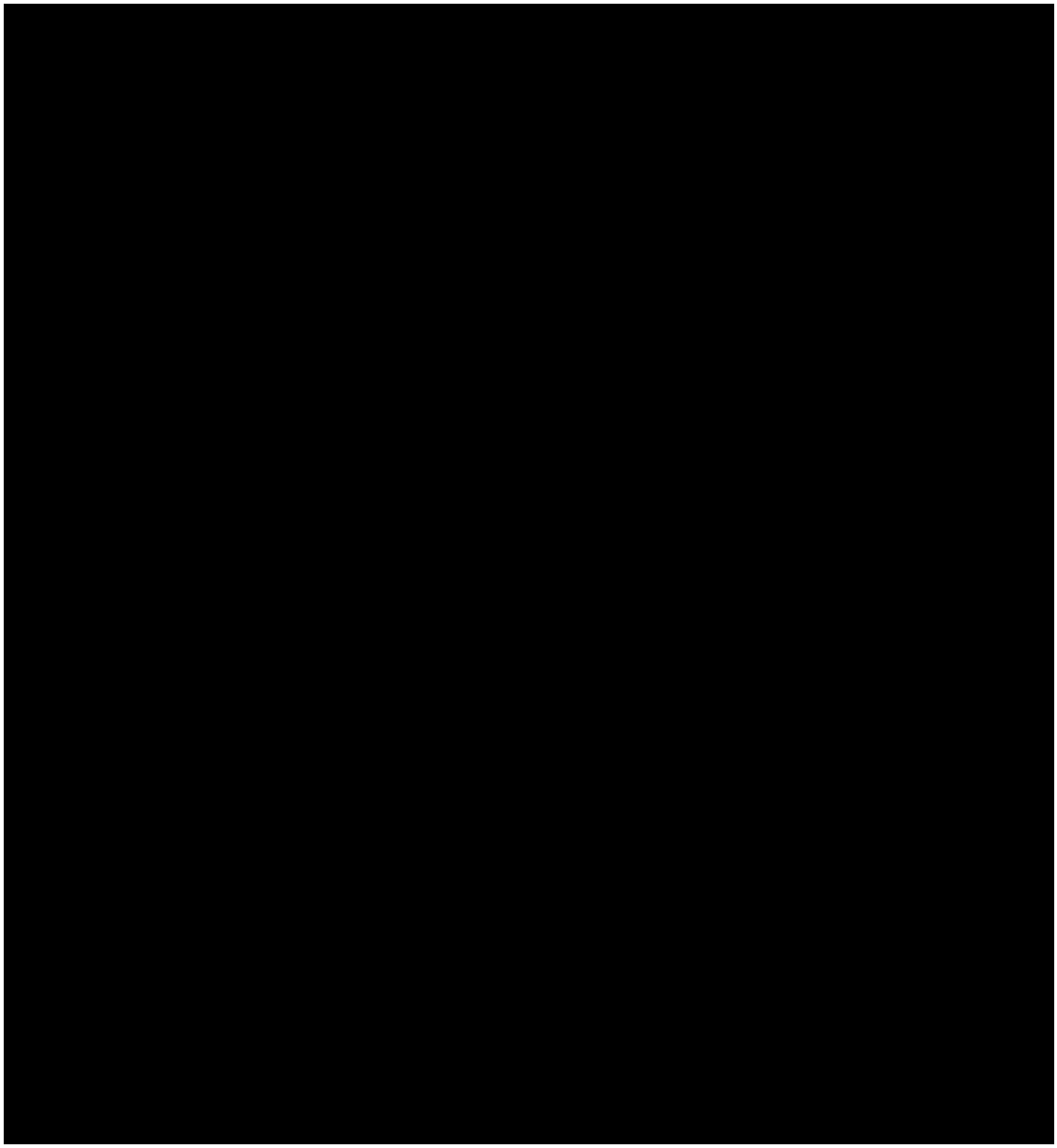
Les différentes configurations pour réaliser une sauvegarde sont maintenant faites. Afin que cela fonctionne, il faut uniquement redémarrer le Director Bareos pour qu'il prenne en compte les changements, comme ci-dessous.

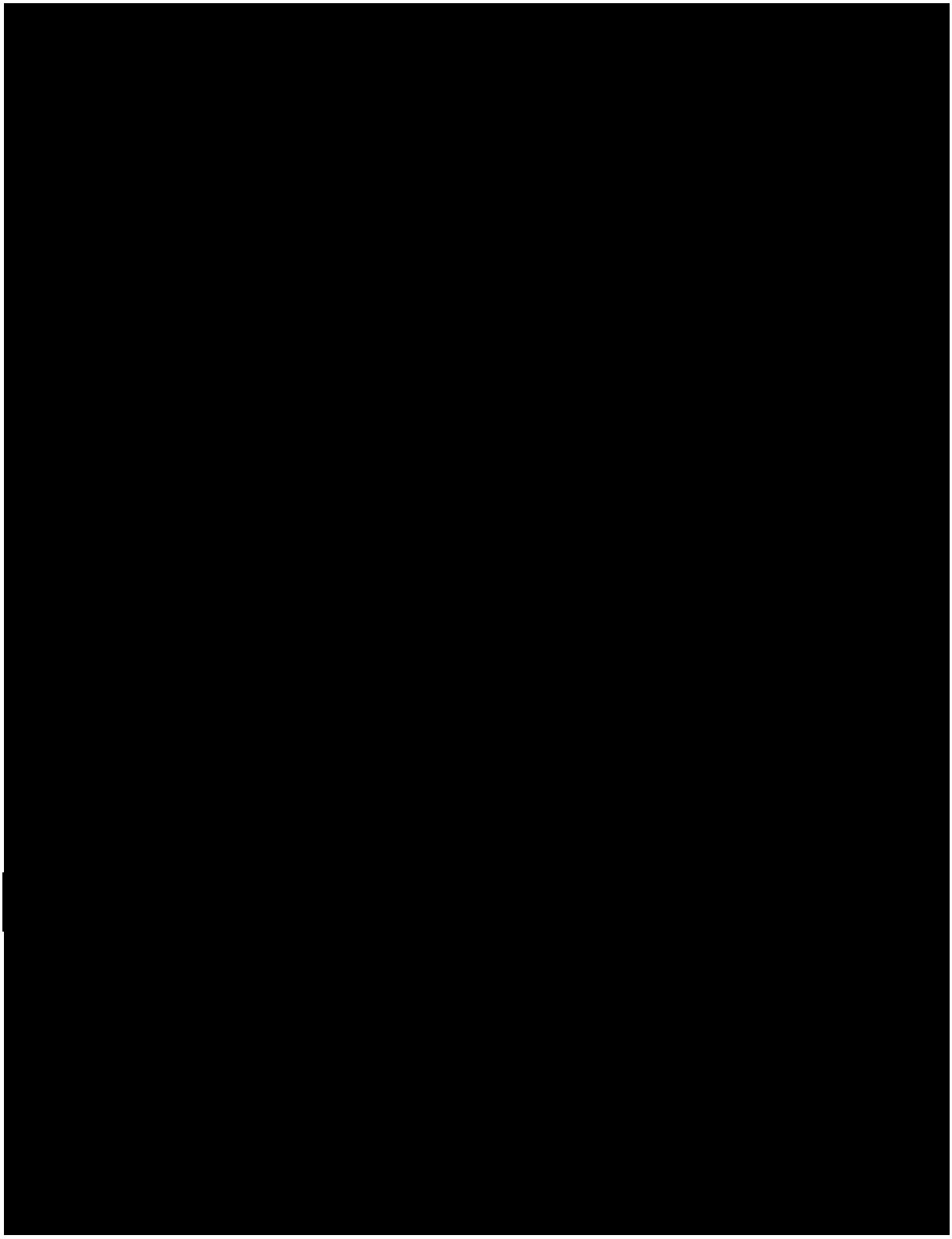
```
systemctl restart bareos-dir && systemctl status bareos-dir
```

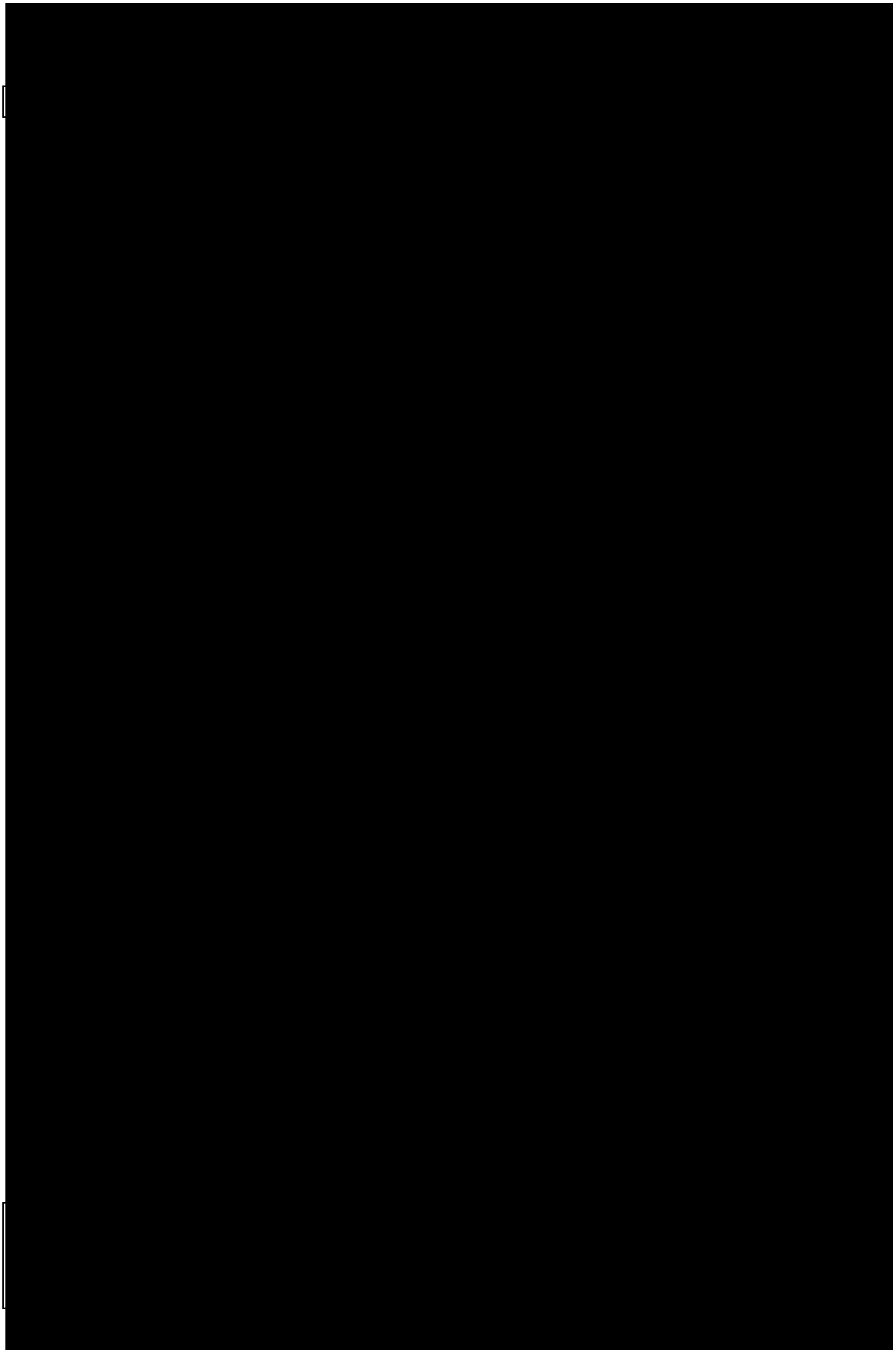
```
# systemctl restart bareos-dir && systemctl status bareos-dir
● bareos-director.service - Bareos Director Daemon service
     Loaded: loaded (/usr/lib/systemd/system/bareos-director.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-11-15 11:00:24 CET; 12ms ago
 Invocation: e29cd209eaf34d1e96406fb7b857f301
       Docs: man:bareos-dir(8)
   Main PID: 15768 ((reos-dir))
      Tasks: 1 (limit: 18507)
     Memory: 1.7M (peak: 1.7M)
        CPU: 6ms
     CGroup: /system.slice/bareos-director.service
             └─15768 "(reos-dir)"

Nov 15 11:00:24          systemd[1]: Started bareos-director.service - Bareos Director Daemon service.
```

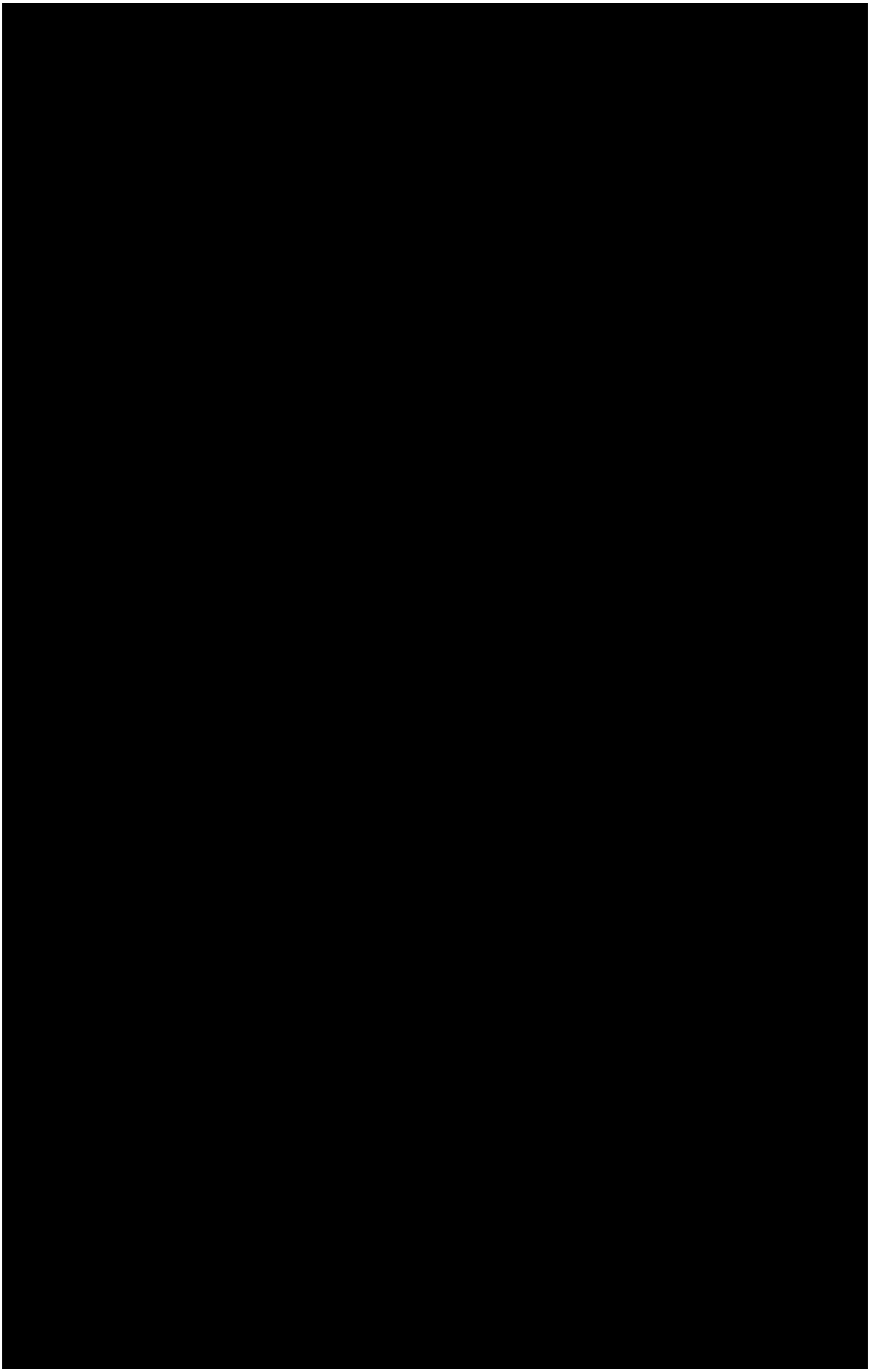
Nous pouvons maintenant réaliser une sauvegarde, comme détaillé dans la partie « Test de Sauvegarde ».

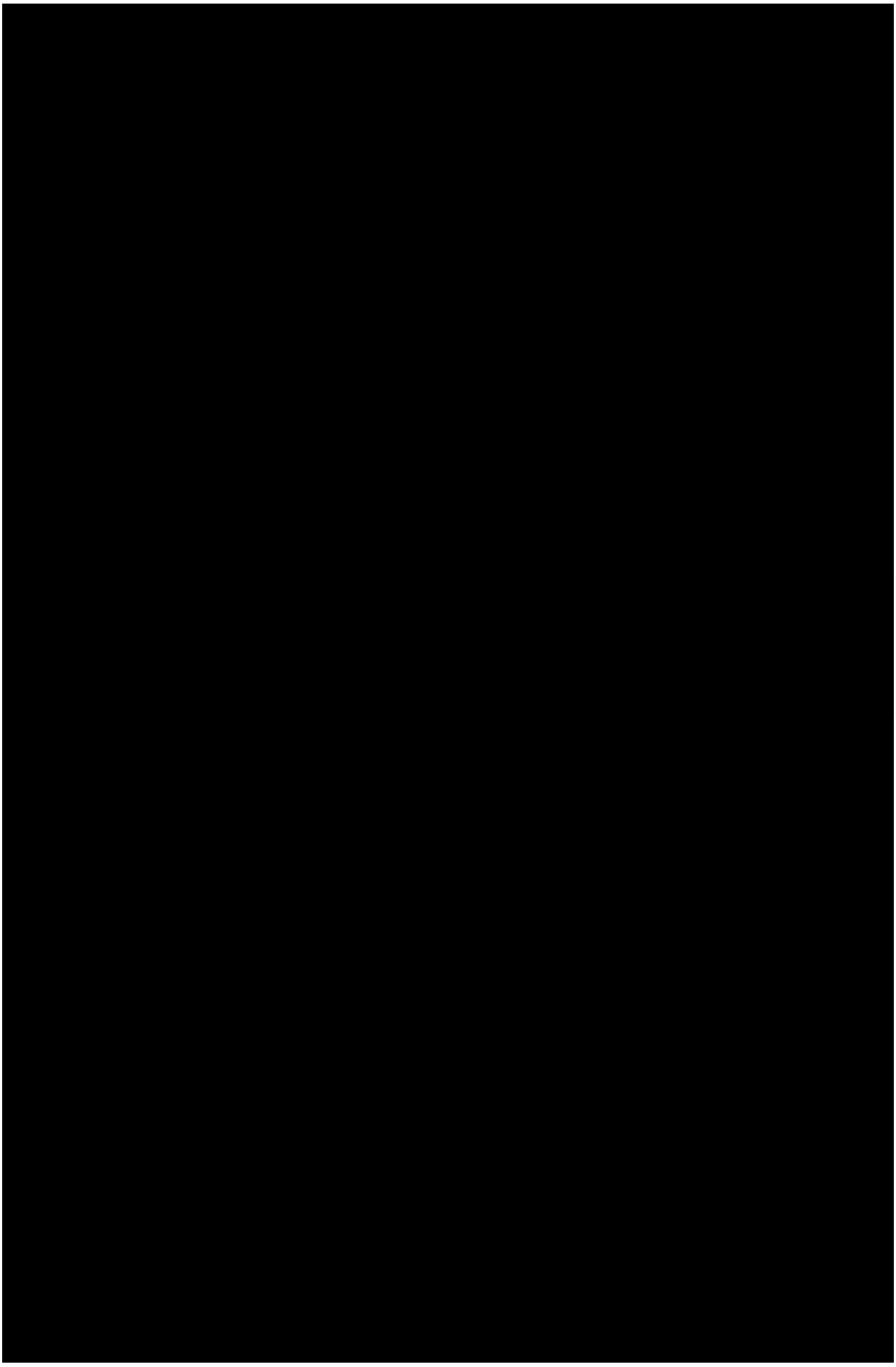
h) Configuration Switch & ISR











```
FileSet {
  Name = "Switch"
  Include {
    Options {
      signature = SHA1
      compression = LZ4
    }
    File =
  }
}
```

Ensuite, il faut configurer la planification de sauvegarde. Une sauvegarde complète (Full) quotidienne est suffisante. Les sauvegardes incrémentielles ne sont pas nécessaires, car les fichiers sources sont supprimés après chaque sauvegarde réussie.

```
# cat /etc/bareos/bareos-dir.d/schedule/SwitchSchedule.conf
Schedule {
  Name = "SwitchCycle"
  Run = Level=Full daily at 22:00
}
```

Après, nous configurons la tâche en créant le fichier /etc/bareos/bareos-dir.d/jobdefs/switch-job.conf. Cette configuration réutilise le FileSet et la planification définis précédemment, et ajoute un script de post-traitement (Run Script) qui permet la suppression automatique des fichiers à la fin de la tâche.

```
# cat /etc/bareos/bareos-dir.d/jobdefs/switchjob.conf
JobDefs {
  Name = "switchjob"
  Type = Backup
  Level = Full
  FileSet = "Switch"
  Schedule = "SwitchCycle"
  Storage = File
  Pool = Full
  Messages = Standard

  Run Script {
    Runs When = After
    Runs On Success = Yes
    Runs On Client = Yes
    Fail Job On Error = No

    # Simple et efficace
    Command = "find                          -type f -delete"
  }
}
```

Enfin, nous avons la tâche qui va être utiliser par notre jobdefs et qui va l'appliquer à notre client, dans notre cas, bareos-fd.

```
# cat /etc/bareos/bareos-dir.d/job/backup-switch.conf
Job {
  Name = "backup-switch"
  JobDefs = "switchjob"
  Client = "bareos-fd"
}
```

Les différentes configurations pour réaliser une sauvegarde sont maintenant faites. Afin que cela fonctionne, il faut uniquement redémarrer le Director Bareos pour qu'il prenne en compte les changements, comme ci-dessous.

```
# systemctl restart bareos-dir && systemctl status bareos-dir
● bareos-director.service - Bareos Director Daemon service
     Loaded: loaded (/usr/lib/systemd/system/bareos-director.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-11-15 12:36:52 CET; 11ms ago
 Invocation: c1a754e7f3d546dfb2efb446f3217991
       Docs: man:bareos-dir(8)
   Main PID: 17072 ((reos-dir))
      Tasks: 1 (limit: 18507)
     Memory: 1.7M (peak: 1.7M)
        CPU: 6ms
     CGroup: /system.slice/bareos-director.service
             └─17072 "(reos-dir)"

Nov 15 12:36:52           systemd[1]: Started bareos-director.service - Bareos Director Daemon service.
```

i) Configuration MariaDB

1) Configuration Client MariaDB

Pour la configuration du client, celui-ci est sous base Debian, pour l'ajouter dans Bareos, il faut voir dans la partie « Configuration Client Debian ».

Pour la partie MariaDB, il faut utiliser le plugin MariaBackup qui est intégré directement dans Bareos.

Pour cela, nous devons tout d'abord installer le plugin MariaBackup et mariadb-backup.

```
apt install bareos-filedaemon-mariabackup-python-plugin mariadb-backup -y
```

```
root@Test-Deb-Bareos:/# apt install bareos-filedaemon-mariabackup-python-plugin mariadb-backup -y
The following package was automatically installed and is no longer required:
  linux-image-6.12.38+deb13-amd64
Use 'apt autoremove' to remove it.

Installing:
  bareos-filedaemon-mariabackup-python-plugin  mariadb-backup
```

Ensuite, nous pouvons vérifier que le plugin est bien présent dans le dossier, comme ci-dessous.

```
root@Test-Deb-Bareos:/# ls -l /usr/lib/bareos/plugins/
total 152
-rw-r--r-- 1 root root 18656 Nov 16 10:07 autoxflate-sd.so
-rw-r--r-- 1 root root  8201 Nov 16 09:53 bareos-fd-local-fileset.py
-rw-r--r-- 1 root root 22873 Nov 16 09:53 bareos-fd-mariabackup.py
-rw-r--r-- 1 root root 19724 Nov 16 09:53 BareosFdPluginBaseclass.py
-rw-r--r-- 1 root root 11749 Nov 16 09:53 BareosFdPluginLocalFilesBaseclass.py
-rw-r--r-- 1 root root  3529 Nov 16 09:53 BareosFdWrapper.py
-rw-r--r-- 1 root root 23112 Nov 16 10:07 bpipe-fd.so
-rw-r--r-- 1 root root 35688 Nov 16 10:07 python3-fd.so
```

Après, il faut configurer notre client Bareos pour qu'il sache où se trouvent ses plugins. Pour cela, il faut ajouter deux lignes dans le fichier « myself.conf » de notre configuration Bareos, comme ci-dessous.

```
Plugin Directory = /usr/lib/bareos/plugins
Plugin Names = "python3"
```

```
root@Test-Deb-Bareos:/# cat /etc/bareos/bareos-fd.d/client/myself.conf
Client {
  Name = Test-Deb-Bareos

  # remove comment from "Plugin Directory" to load plugins from specified directory.
  # if "Plugin Names" is defined, only the specified plugins will be loaded,
  # otherwise all filedaemon plugins (*-fd.so) from the "Plugin Directory".
  #
  Plugin Directory = /usr/lib/bareos/plugins
  Plugin Names = "python3"
}
```

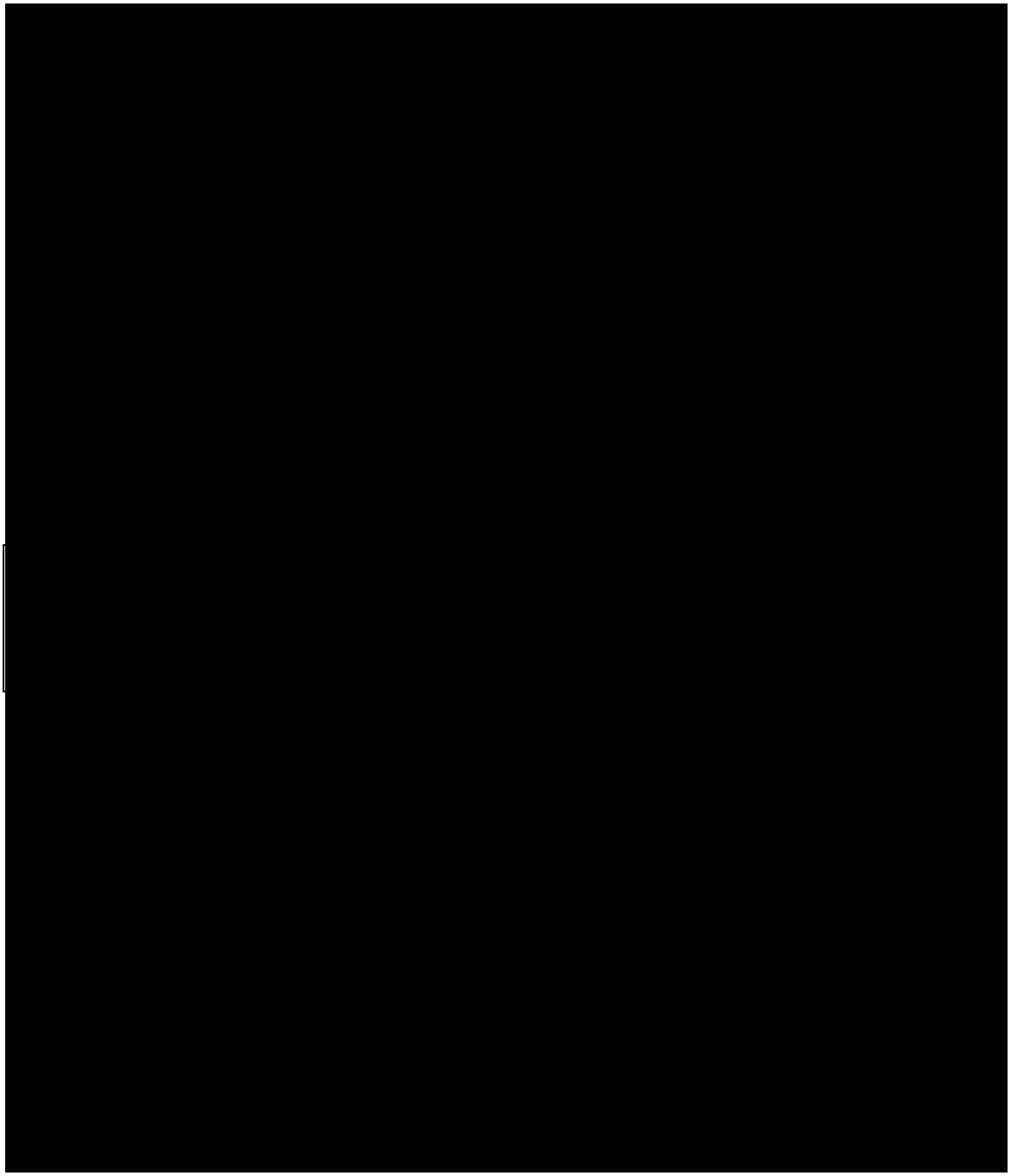
Une fois cela fait, il faut redémarrer le service « bareos-fd » pour qu'il prenne en compte les modifications.

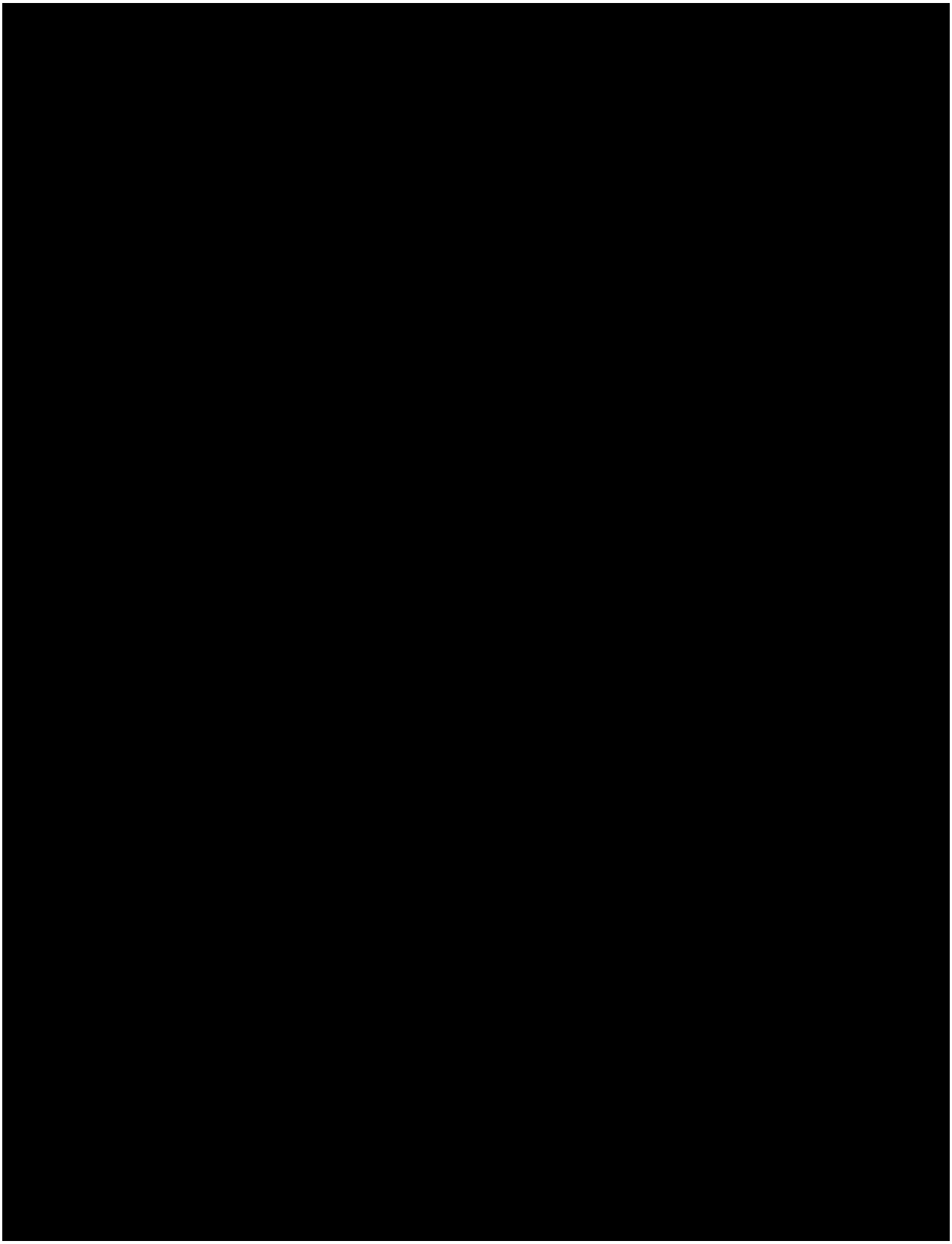
```
systemctl restart bareos-fd && systemctl status bareos-fd
```

```
root@Test-Deb-Bareos:/# systemctl restart bareos-fd && systemctl status bareos-fd
● bareos-filedaemon.service - Bareos File Daemon service
     Loaded: loaded (/usr/lib/systemd/system/bareos-filedaemon.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-11-16 11:47:49 CET; 22ms ago
 Invocation: 0fcf6bf68997470893da27f25222fbcc
       Docs: man:bareos-fd(8)
   Main PID: 31183 (bareos-fd)
      Tasks: 1 (limit: 2236)
     Memory: 1.5M (peak: 1.7M)
        CPU: 9ms
     CGroup: /system.slice/bareos-filedaemon.service
             └─31183 /usr/sbin/bareos-fd -f
```

Avec ces différentes actions, la partie Bareos est configurée, mais le serveur doit avoir un accès direct à la base de données. Pour cela, il faut créer un utilisateur avec uniquement un minimum de droits afin qu'il puisse uniquement lire les bases de données, comme ci-dessous avec un utilisateur Bareos.

```
root@Test-Deb-Bareos:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 11.8.3-MariaDB-0+deb13u1 from Debian -- Please help get to 10k
ariaDB/Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```





Après avoir réalisé ces différentes modifications, il faut redémarrer le Director Bareos afin que l'ensemble des changements soit pris en compte.

```
systemctl restart bareos-dir && systemctl status bareos-dir
```

```
# systemctl restart bareos-dir && systemctl status bareos-dir
● bareos-director.service - Bareos Director Daemon service
     Loaded: loaded (/usr/lib/systemd/system/bareos-director.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-11-16 12:28:28 CET; 11ms ago
 Invocation: 2fc88bb34c764cd3a19f9933554235eb
       Docs: man:bareos-dir(8)
   Main PID: 2631 ((reos-dir))
      Tasks: 1 (limit: 18508)
     Memory: 1.7M (peak: 1.7M)
        CPU: 6ms
     CGroup: /system.slice/bareos-director.service
             └─2631 "(reos-dir)"
```

Une fois ces différentes actions réalisées, la sauvegarde est fonctionnelle.

j) Sauvegarde de machines

1) Sauvegarde via CLI

Nous pouvons réaliser nos sauvegardes en ligne de commande, via l'API intégrée à Bareos, la bconsole.

Pour cela, il faut aller sur le serveur Bareos et taper la commande « bconsole », comme ci-dessous.

```
bconsole
```

```
# bconsole
Connecting to Director localhost:9101
 Encryption: TLS_CHACHA20_POLY1305_SHA256 TLSv1.3
1000 OK: bareos-dir Version: 24.0.8~pre8.dbdd292a4 (13 November 2025)
Bareos community build (UNSUPPORTED).
Get professional support from https://www.bareos.com
You are connected using the default console

Enter a period (.) to cancel a command.
*
```

Ensuite, pour lancer une tâche de sauvegarde, il faut utiliser la commande « run » et ajouter les options que l'on veut.

```
run job=backup-Test-Deb-Bareos
```

```
*run job=backup-Test-Deb-Bareos
accurate=       catalog=        fileset=        level=
pluginoptions=  since=          verifyjob=      where=
backupclient=   client=         job=            migrationjob=
pool=           spooldata=      verifylist=     yes
backupformat=   comment=        jobid=          nextpool=
priority=       storage=        when=
```

Dans notre cas, notre job a déjà toutes les configurations que l'on veut, alors nous pouvons directement la lancer, comme ci-dessous. Une fois lancé, Bareos nous affiche les différentes options de notre job et nous pouvons confirmer le lancement de celle-ci en faisant « yes ».

```
run job=backup-Test-Deb-Bareos
```

```
*run job=backup-Test-Deb-Bareos
Using Catalog "MyCatalog"
Run Backup job
JobName:  backup-Test-Deb-Bareos
Level:    Incremental
Client:   Test-Deb-Bareos
Format:   Native
FileSet:  Test-Deb-Bareos
Pool:     Incremental (From Job IncPool override)
Storage:  File (From Job resource)
When:     2025-11-16 15:24:57
Priority: 10
OK to run? (yes/mod/no): yes
Job queued. JobId=461
You have messages.
```

Ensuite, nous pouvons regarder l'état du client, afin de voir si la tâche est toujours en cours, s'il y a des erreurs, des warnings ou si celle-ci est terminée, comme ci-dessous.

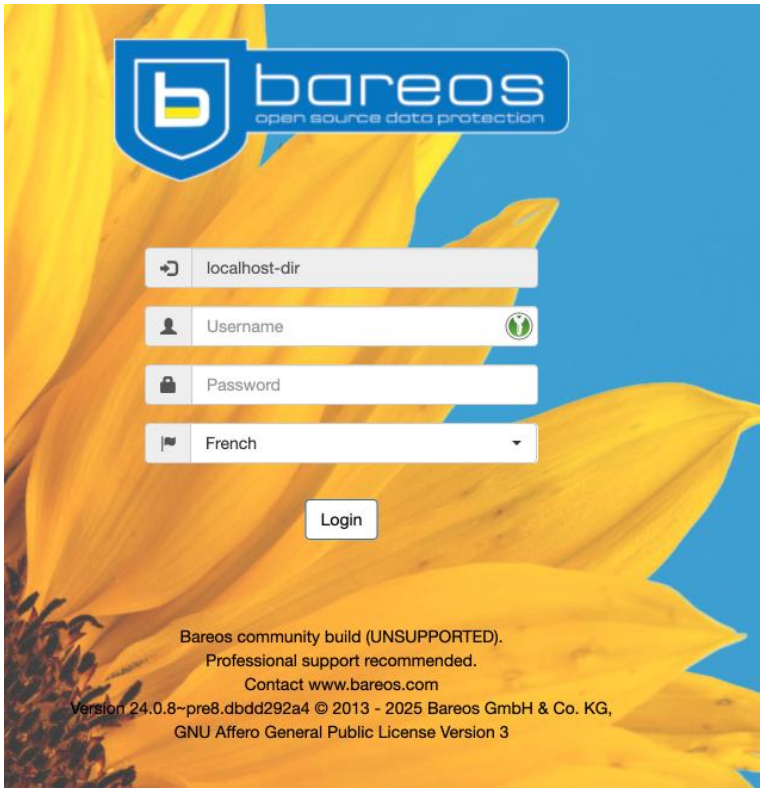
```
status client=Test-Deb-Bareos
```

```
*status client=Test-Deb-Bareos
Connecting to Client Test-Deb-Bareos at
 Handshake: Immediate TLS, Encryption: TLS_CHACHA20_POLY1305_SHA256 TLSv1.3

Test-Deb-Bareos Version: 24.0.8~pre13.9a9c0d299 (16 November 2025)  Debian GNU/Linux 13 (trixie)
Daemon started 16-Nov-25 12:34. Jobs: run=5 running=0, Bareos community binary
 Sizeof: boffset_t=8 size_t=8 debug=0 trace=0 bwlimit=0kB/s
Plugin Info:
 Plugin     : python3-fd.so
 Description: Python File Daemon Plugin
 Version    : 4 (May 2020)
 Author     : Bareos GmbH & Co. KG
 License    : Bareos AGPLv3
 Usage      : python3:module_name=<python-module-to-load>:module_path=<path-to-python-modules>:.

  module_name: The name of the Python module.
  module_path: Python search path for the module.
               The path '/usr/lib/bareos/plugins' is always checked for modules.
  Additional parameters are plugin specific.



Running Jobs:
bareos-dir (director) connected at: 16-Nov-25 15:28
No Jobs running.
====

Terminated Jobs:
 JobId  Level    Files      Bytes   Status   Finished        Name
====================================================================
   433  Full         37    64.48 M  Cancel   16-Nov-25 09:33 backup-Test-Deb-Bareos
   441  Full         37    72.87 M  Cancel   16-Nov-25 10:09 backup-Test-Deb-Bareos
   456  Full          1         0   Error    16-Nov-25 12:31 backup-Test-Deb-Bareos
   457  Full          1         0   Error    16-Nov-25 12:35 backup-Test-Deb-Bareos
   458  Full          1         0   Error    16-Nov-25 14:32 backup-Test-Deb-Bareos
   459  Full          2    1.235 M  OK       16-Nov-25 14:35 backup-Test-Deb-Bareos
   460                1    48.69 M  OK       16-Nov-25 14:36 RestoreFiles
   461  Incr          2    1.035 M  OK       16-Nov-25 15:25 backup-Test-Deb-Bareos
```

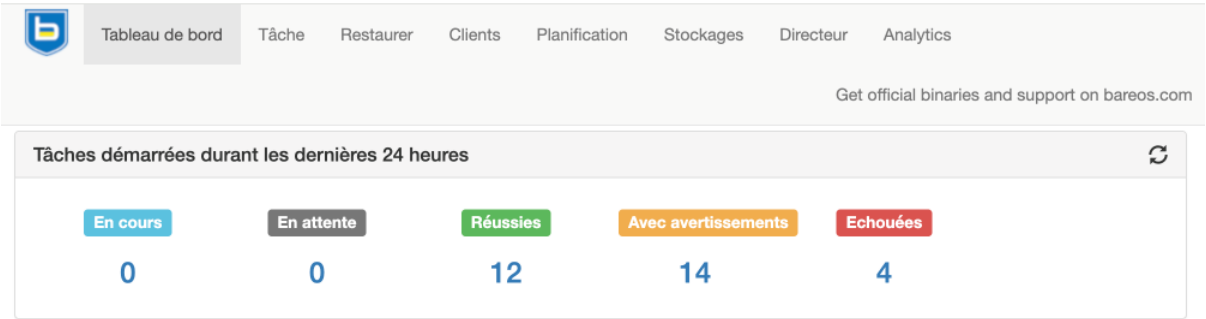
2) Sauvegarde via interface Web

Grâce au Bareos-Webui, nous pouvons également lancer des tâches et vérifier les tâches en cours.

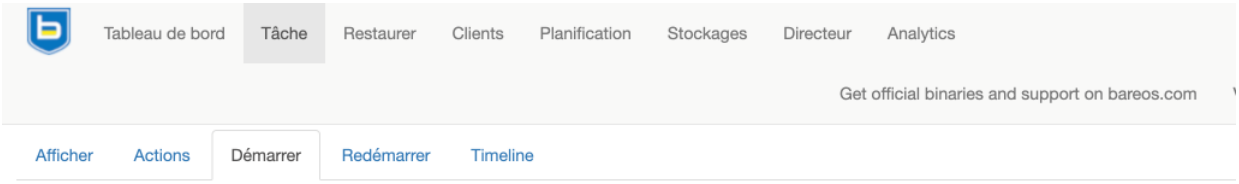
Pour aller sur l'interface Web, il suffit d'ouvrir un navigateur web, de taper le nom FQDN (Fully Qualified Domain Name) de la machine suivi de /bareos-webui et la page ci-dessous devrait s'afficher.



Ensuite, il suffit de se connecter avec un compte Bareos sur la page de connexion et la fenêtre ci-dessous devrait s'afficher.

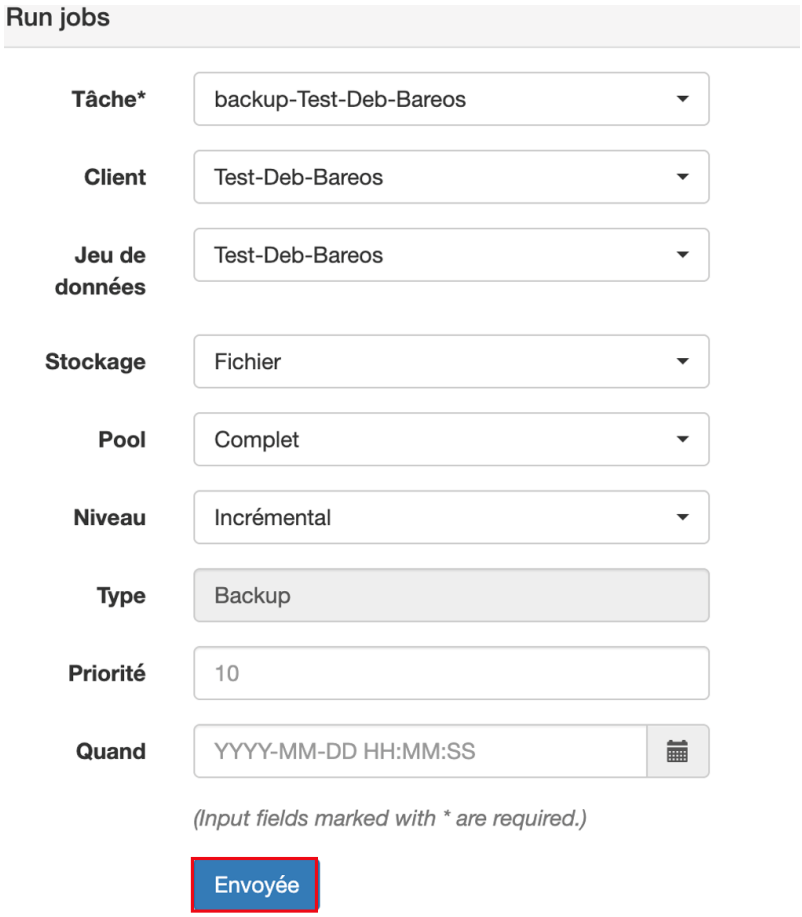


Sur cette page, nous pouvons aller dans l'onglet « Tâche » puis l'onglet « Démarrer » afin de pouvoir ensuite démarrer une tâche de sauvegarde.

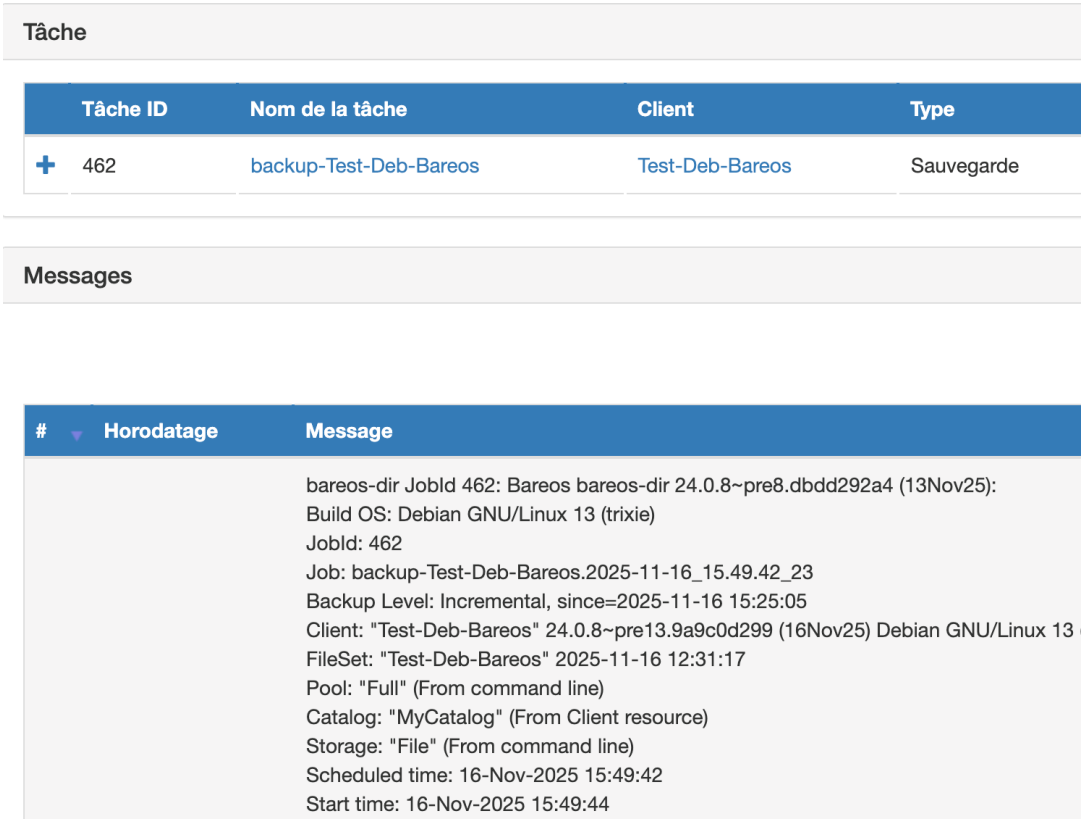


Dans cet onglet « Démarrer », nous pouvons choisir la tâche (job) que l'on veut démarrer, le client, le stockage, si on veut une sauvegarde incrémentielle ou une sauvegarde complète ou encore la priorité de la tâche.

Pour commencer celle-ci, il faut cliquer sur le bouton « Envoyer ».



Une fois cela fait, Bareos nous redirige sur une nouvelle page avec la tâche en cours, les logs de celle-ci, son état et de façon générale toutes les informations sur la tâche.



k) Restauration de machines

1) Restauration en CLI

Pour restaurer des fichiers ou une machine complète via la console Bareos (bconsole), on utilise la commande « restore ».

Cette commande permet de choisir quel client restaurer, quelle sauvegarde utiliser, quels fichiers/dossiers restaurer et l'emplacement de restauration.

Avec cette commande, nous devons tout d'abord choisir la sauvegarde que l'on veut restaurer, dans notre cas, la dernière sauvegarde réalisée.

```
restore
```

```
*restore
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"

First you select one or more JobIds that contain files
to be restored. You will be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.

To select the JobIds, you have the following choices:
 1: List last 20 Jobs run
 2: List Jobs where a given File is saved
 3: Enter list of comma separated JobIds to select
 4: Enter SQL list command
 5: Select the most recent backup for a client
 6: Select backup for a client before a specified time
 7: Enter a list of files to restore
 8: Enter a list of files to restore before a specified time
 9: Find the JobIds of the most recent backup for a client
10: Find the JobIds for a backup for a client before a specified time
11: Enter a list of directories to restore for found JobIds
12: Select full restore to a specified Job date
13: Cancel
Select item:  (1-13): 5
```

Ensuite, il faut définir le client sur lequel nous allons réaliser la restauration, dans notre cas, la machine Test-Deb-Bareos.

```
5: Test-Deb-Bareos
```

Après, il faut définir le FileSet correspondant à notre restauration, dans notre cas, celui qui correspond à notre sauvegarde de base de données, « Test-Deb-Bareos ».

```
The defined FileSet resources are:
1: LinuxAll
2: Test-Deb-Bareos
Select FileSet resource (1-2): 2
```

Une fois cela fait, nous devons choisir les dossiers/fichiers que l'on veut restaurer. Dans notre cas, l'ensemble des fichiers/dossiers sauvegardés, nous choisissons cela avec « mark * » puis nous validons notre choix avec « done ».

```
mark *
done
```

```
cwd is: /
$ mark *
3 files newly marked.
$ done
```

À la suite des différents choix, Bareos va faire un récapitulatif de la configuration de notre restauration. Nous pouvons voir qu'il n'y a pas de fichier de destination correspondant au « Where », afin de changer cela, nous pouvons renseigner « mod », pour modification.

```
mod
```

```
Using Catalog "MyCatalog"
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /var/lib/bareos/bareos-dir.restore.2.bsr
Where:
Replace:         Always
FileSet:         LinuxAll
Backup Client:   Test-Deb-Bareos
Restore Client:  Test-Deb-Bareos
Format:          Native
Storage:         File
When:            2025-11-16 16:17:19
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no): mod
```

Nous allons alors devoir modifier le « Where », celui-ci correspond au numéro 10.

```
Parameters to modify:
 1: Level
 2: Storage
 3: Job
 4: FileSet
 5: Restore Client
 6: Backup Format
 7: When
 8: Priority
 9: Bootstrap
10: Where
11: File Relocation
12: Replace
13: JobId
14: Plugin Options
Select parameter to modify (1-14): 10
```

Une fois le « Where » configuré, dans notre cas dans le /tmp/, nous pouvons valider la restauration avec « yes ». Suite à cette action, Bareos va démarrer la tâche de restauration sur le client défini.

```
Select parameter to modify (1-14): 10
Please enter the full path prefix for restore (/ for none): /tmp/
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /var/lib/bareos/bareos-dir.restore.2.bsr
Where:           /tmp/
Replace:         Always
FileSet:         LinuxAll
Backup Client:   Test-Deb-Bareos
Restore Client:  Test-Deb-Bareos
Format:          Native
Storage:         File
When:            2025-11-16 16:17:19
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no): yes
Job queued. JobId=463
You have messages.
*
```

Nous pouvons vérifier que la tâche s'est déroulée correctement avec la commande ci-dessous. Nous pouvons voir que notre tâche s'est lancée, qu'elle a duré 3 secondes, que 3 fichiers ont été restaurés et qu'elle est terminée (T).

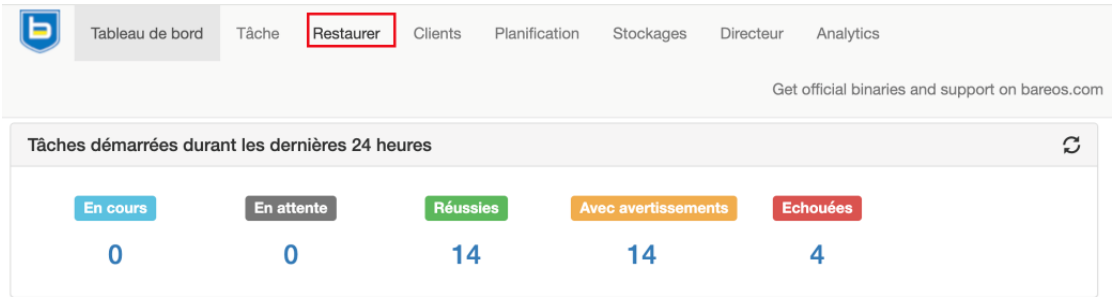
```
|   463 | RestoreFiles                    | Test-Deb-Bareos                          |
+-------+---------------------------------+------------------------------------------+-

| 00:00:03 | R    | F     |         3 |     57,660,152 | T          |
+----------+------+-------+-----------+----------------+------------+
```

Il est également possible de vérifier directement sur le client que les fichiers concernés ont été correctement restaurés, comme nous pouvons le voir ci-dessous. Bareos nous a bien restauré notre configuration MariaDB avec MariaBackup.

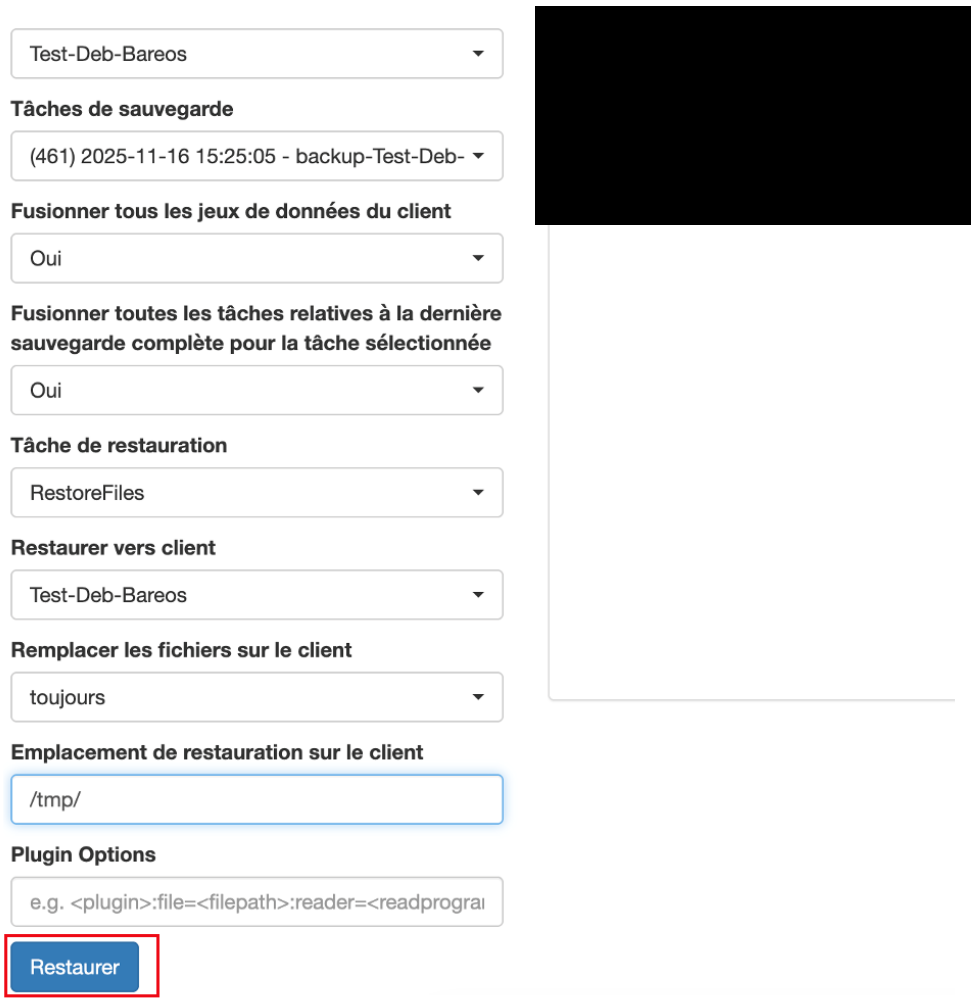
```
root@Test-Deb-Bareos:/# ls -l /tmp/
total 0
drwxr-xr-x 4 root bareos 80 Nov 16 16:18 _mariadb-backup
```

2) Restauration via interface Web

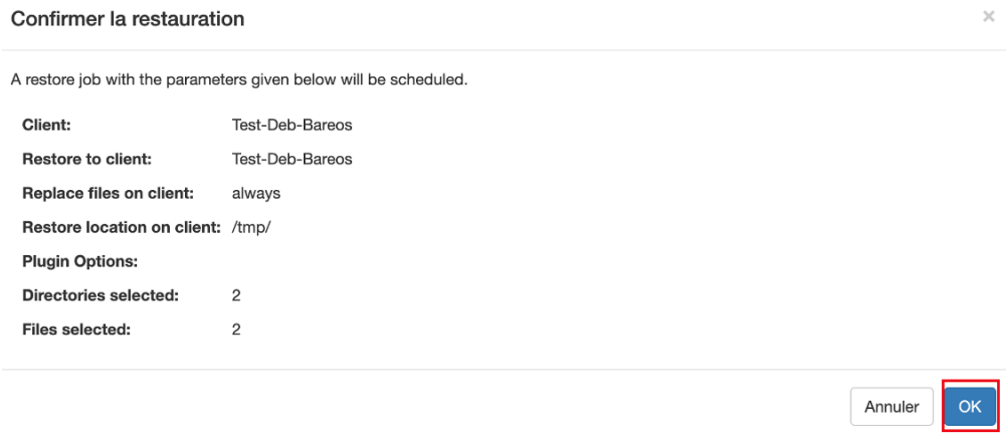
Comme pour la sauvegarde, il faut se connecter sur l'interface Web (bareos-webui) pour réaliser les actions. Sur celle-ci, nous pouvons aller directement dans le menu « Restaurer ».



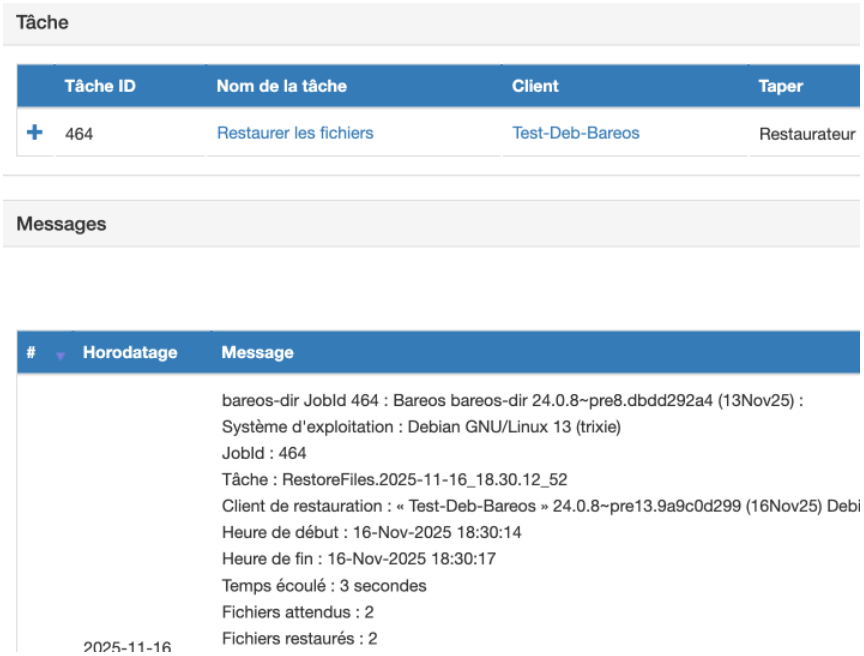
Dans le menu de restauration, nous pouvons choisir le backup du client que l'on veut restaurer, sur quel client on veut restaurer, si on veut remplacer les doublons, le dossier où l'on veut restaurer ou encore si l'on veut utiliser des plugins. Ci-dessous, nous avons choisi de restaurer l'ensemble de notre dernière sauvegarde dans le /tmp/ de notre machine. Une fois les options configurées, il faut cliquer sur le bouton « Restaurer ».



Ensuite, nous devons confirmer la restauration en cliquant sur le bouton « OK ».



Une fois cela fait, Bareos affiche un rapport de la tâche, cela nous permet de vérifier que celle-ci s'est correctement déroulée.



Nous pouvons également vérifier directement sur le client que le dossier « _mariadb-backup » a bien été créé, comme ci-dessous.

```
ls -l /tmp/
```

I) Axes d’amélioration

Bien que le projet ait atteint ses objectifs initiaux, il y a plusieurs axes d'amélioration sur ce projet.

Au niveau de la solution, il y a différents axes d'amélioration. Lors du premier déploiement de la solution en production, celle-ci n'a pas été déployée sur Windows car Bareos ne prenait pas en charge la restauration de fichiers système, l'outil prenait en charge uniquement la restauration de fichiers sur Windows.

Mais une nouvelle solution de Bareos est sortie à la suite du premier déploiement, la version Bareos 25.0 qui prend en charge la restauration du système d'exploitation complet de Windows et intègre des modules d'extension sur Hyper-V et Proxmox pour les sauvegardes et restaurations.

Bareos 25 : Nouvelles fonctionnalités

Voici quelques-unes des caractéristiques les plus importantes du Bareos 25 :

- **Restauration système complète Windows** : restauration complète du système Windows, y compris la création et la restauration d’images avec protection système complète.
- **Module d'extension Hyper-V** : Intégrez Bareos à Microsoft Hyper-V, sauvegardez les machines virtuelles et restaurez-les directement.
- **Prise en charge de Proxmox** : sauvegardes complètes des machines virtuelles et conteneurs Proxmox, restauration directe sur la VM/CT ou sur disque via l’interface graphique de Proxmox, sauvegarde de plusieurs VM en une seule opération.

m) Conclusion

En conclusion, j'ai trouvé que cette activité était très enrichissante, j'ai réalisé des tâches que je n'avais jamais faites auparavant, cette activité était intéressante par sa diversité ainsi que par sa recherche.

J'ai pu rencontrer quelques difficultés au niveau de l'organisation du projet, celui-ci était ma première expérience en tant que chef de projet et je n'ai pas forcément su organiser l'ensemble des tâches qui m'étaient attribuées. J'ai pu également rencontrer des difficultés sur la communication du projet, il m'a été difficile de communiquer sur mes différents besoins ou même de communiquer avec les personnes de l'équipe technique.

Malgré les difficultés rencontrées au cours de ce projet, je pense avoir réussi à atteindre l'objectif de celui-ci qui était de mettre en place une nouvelle solution de sauvegarde qui serait conforme à nos différents besoins. Je suis satisfait de ce qui a pu être réalisé et de l'aide que j'ai pu apporter.

Pour terminer, cette activité m'a permis de valider plusieurs compétences du référentiel BTS SIO SISR :

- Gérer le patrimoine informatique
- Répondre aux incidents et aux demandes d'assistance et d'évolution
- Travailler en mode projet
- Mettre à disposition des utilisateurs un service informatique

Portfolio

Un portfolio a été réalisé au cours de notre formation afin de nous présenter, montrer les différents dossiers que nous avons pu réaliser, montrer notre parcours professionnel et scolaire ainsi que nos compétences. Ce portfolio permet de répondre à la compétence « **Organiser son développement professionnel** » du BTS SIO

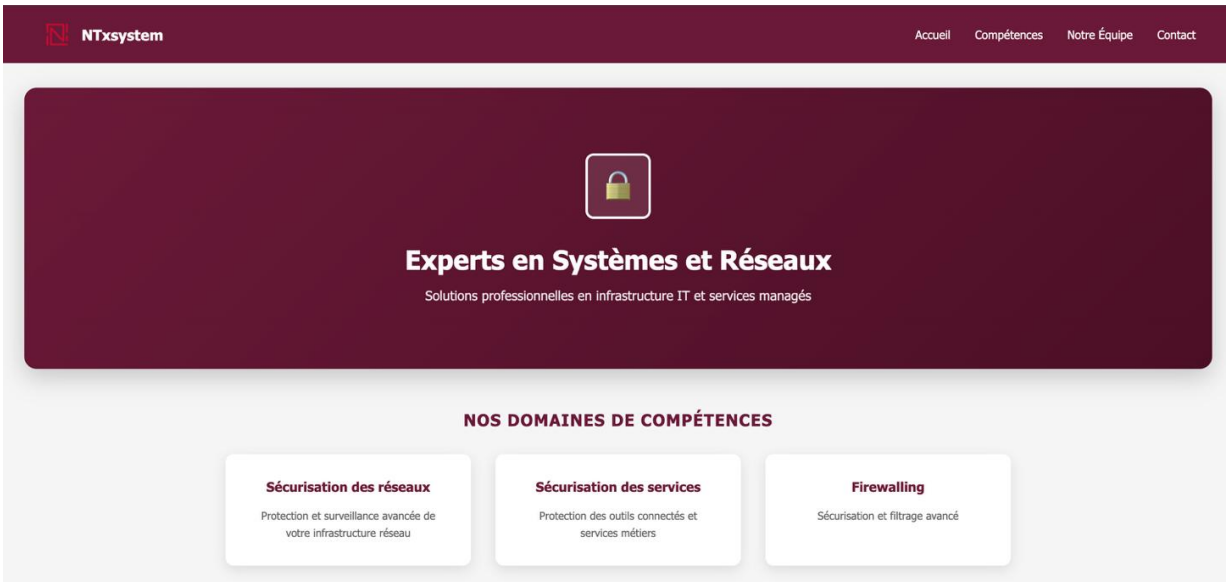
Lien du Portfolio : https://matt.moreau.formation-esiac.fr/

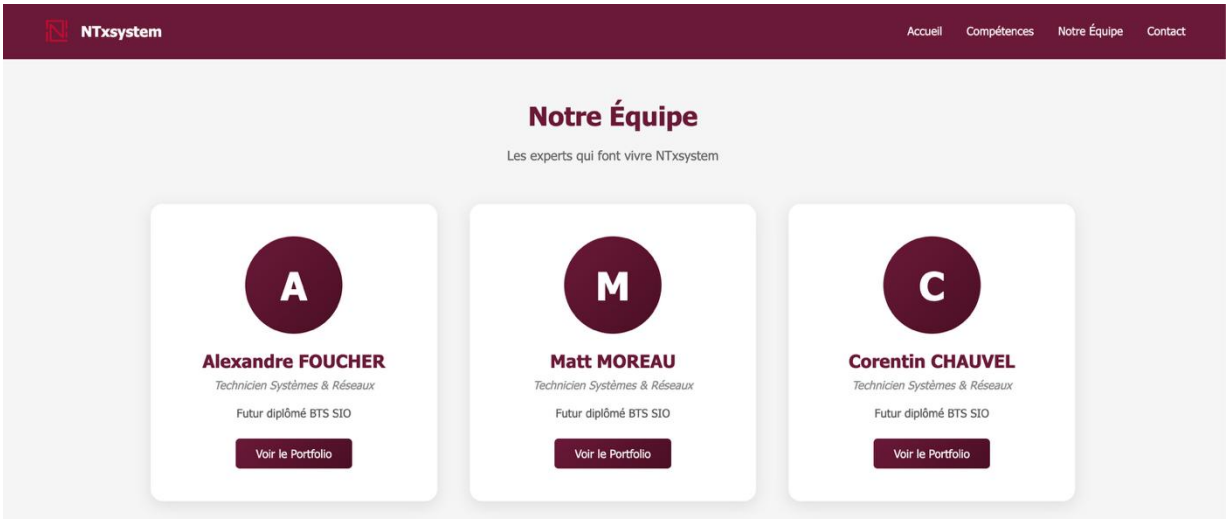
Site Entreprise

Dans le cadre de notre formation, nous avons pu monter une entreprise pédagogique afin de mettre en place une baie pédagogique et étudier le concept d'entreprise en droit.

Lors de cette période de projet, nous avons pu mettre en place un site web pour cette entreprise pédagogique, ce site web permet de répondre à la compétence « **Développer la présence en ligne de l'organisation** » pour le BTS SIO. Ci-dessous des images du site web.

Lien du site web : http://ntxsystem.fr/





Bilan de l'alternance

Cette alternance a été très enrichissante, elle m'a permis de découvrir particulièrement le métier d'administrateur LAN, ce qui m'a considérablement plu, j'ai pu apprendre à faire du support utilisateur, de la préparation d'ordinateur ou encore de la configuration de switchs.

Cette alternance m'a également permis de découvrir le monde du travail, d'évoluer en équipe et de découvrir le lien qu'il y a entre chaque service d'une entreprise.

Je tire un bilan très positif de cette partie en entreprise, cela m'a permis d'appliquer en conditions réelles les techniques et méthodes apprises au sein de la Fab'Academy, cela m'a aussi permis d'apprendre des compétences que l'on ne travaille pas à l'école, toutes ces approches m'ont permis d'évoluer en maturité et en autonomie.

Enfin cette alternance m'a conforté dans l'idée de poursuivre ma formation vers le titre professionnel ASR (Administrateur Systèmes et Réseaux) en alternance afin de continuer à découvrir le monde de l'informatique et de résoudre des erreurs de plus en plus complexes.