Rapport de Stage

Au sein de l’entreprise **Wedia**

Par **Rémi Albertucci**

Remerciements :

Merci à **Thierry Martinez** pour son accueil et ses conseils, merci également à **Geoffrey Declerck** pour son temps et un grand à merci à **Cédric Alcisiadi** grâce à qui j'ai pu venir en entreprise et progresser

Table des matières

I Intro et contexte

Dans le cadre de ma reconversion et par le fait que je n’ai pas trouvé d’alternance pour ma première année en BTS SIO, j’ai effectué un stage en entreprise. Ce stage constitue une première expérience du monde du travail dans l’univers de l’informatique.

Au cours de cette période, j’ai pu observer et participer à diverses activités liées à l’informatique, comprendre l’organisation interne, et découvrir les processus et outils utilisés pour la gestion des projets et des systèmes. Cette première expérience constitue donc une étape importante dans ma formation, me permettant d’appréhender les exigences du monde professionnel et de réfléchir à mes futurs choix de spécialisation.

Durant ce stage, j’ai également été amené à me familiariser avec certains outils et technologies couramment utilisés dans le secteur informatique. J’ai pu observer le déploiement d’applications conteneurisées, la gestion des environnements de travail et le suivi des systèmes en fonctionnement. Même si je n’ai pas participé à toutes les opérations techniques, cette immersion m’a permis de comprendre l’importance de la coordination entre les différents services, la gestion des priorités et la mise en place de procédures permettant de garantir la disponibilité et la fiabilité des applications. Ces observations m’ont donné un aperçu concret des pratiques professionnelles et m’ont aidé à mieux situer mes acquis théoriques dans un contexte réel.

II Présentation de l’entreprise

L’entreprise

Wedia est une entreprise proposant une solution de Digital Asset management (DAM).

Un DAM permet de stocker et traiter les assets, à comprendre les images, photos, vidéos… d’une entreprise dans le but de pouvoir les exploiter, à des fins principalement publicitaire : spots TV, pubs internet, spots radios, publicité sur le lieu de vente (PLV)...

Une plateforme en ligne permet aux développeurs web, graphiste, ou tout autre publicitaire au sens large de récupérer les assets pour les intégrer dans leur communication. La plateforme permet de centraliser les assets et de créer un accès à différents corps de métiers pour collaborer, même à distance.

En somme, on peut voir le DAM comme un Cloud géant.

Mais qu’est ce qui différencie un DAM d’un simple Cloud ?
Le traitement.

Les assets ne sont pas seulement stockés mais ils sont surtout traités. Différents types de traitements existent pour différents assets, nous prendrons donc l’exemple d’une photo promotionnelle.

Une photo issue d’un shooting professionnel se présente souvent sous la forme d’un fichier RAW. Comme son nom l’indique, il s’agit d’un fichier brut, non compressé, et pouvant peser plusieurs dizaines voire centaines de mégaoctets. Ses dimensions sont souvent aussi impressionnantes que sa taille, avec des fichiers en équivalent 16K : un de fichier sous ce format est inexploitable pour de la promotion, trop lourd à charger sur un page web par exemple.

Le DAM va donc, de manière automatisée, convertir ce fichiers en différents formats : on peut imaginer un exemplaire redimensionné en 1920x1080p et convertit en JPEG, un autre au format carré, un troisième en 854x480p…

De cette manière, le fichier original est conservé mais il est possible d’exploiter cette photo aisément. Il suffit de se connecter à la plateforme et de récupérer l’image au format désiré.

Il en va de même pour les fichiers vidéos par exemple, qui représentent une grosse partie des traitements car ces fichiers demandent beaucoup plus de ressources.

Chaque « publicitaire » pourra donc directement exploiter les fichiers qu’il désire sans avoir à effectuer les traitements et conversions lui-même.

Donc en plus de rendre les assets accessibles, un DAM permet d'exploiter ces fichiers directement. Il est donc aujourd'hui indispensable pour une entreprise voulant faire de la publicité à grande échelle de disposer d'un DAM.

Nous avons maintenant défini ce que fait un DAM, mais Wedia dans tout ça ?

Wedia est une entreprise fondée en 2010, constituée d'une centaine d'employés répartis sur 4 sites (Paris, Montpellier, Limoges et Francfort). Elle recense plus de 4500 clients dont des groupes comme Décathlon, le Crédit Agricole ou encore Kiabi ou Renault Trucks.

Le site de Montpellier, où j'ai effectué mon stage, est composé de trois personnes. Il s'agit concrètement centre de développement et débugage de certaines fonctionnalité, notamment le traitement des assets : une grande partie des mécanismes de traitement ont été conçus par l'équipe sur place.

Wedia se démarque par les possibilités de personnalisation de son DAM, qu'elle peut décliner sous plusieurs formes en fonctions des besoins réels de ses clients. C'est un point fort qui la démarque son principal concurrent et actuellement premier sur le marché, l'entreprise Bynder.

Les locaux de Montpellier

Les locaux de Montpellier sont en fait situés à Mauguio, dans une zone industrielle.

Ce sont des locaux simples, constitués d'un open space comprenant des espaces de travail (des bureaux) ainsi que de deux pièces. Les pièces servent pour l'une de bureau d'appoint, fermé pour plus de discrétion lors de visioconférences ou autre, et l'autre sert principalement de placard pour le matériel au rebus.

Les lieux sont habités la journée par 3 personnes : Cédric, Thierry ainsi que le nouvel arrivant, Geoffrey.

Les deux premiers étant dans l'entreprise depuis de nombreuses années, leur titre officiel de « resposable produit » et « directeur produit » ne se reflète pas dans leur travail, et ils occupent plutôt une fonction de développeur senior, experts sur les différents processus, et s'occupent de l'implémentation de nouvelles fonctionnalités ainsi que du débugage lorsque nécessaire. Issus de parcours d'ingénieurs logiciels et avec de nombreuses années d'expérience, ils constituent des piliers de l'entreprise.

Geoffrey quant à lui est un jeune homme sorti d'étude. Disposant d'un master en développement, il occupe les mêmes fonctions que Cédric et Thierry, tout en continuant d'apprendre les spécificités du système et les rouages déjà en place.

III Mes missions

J’ai pu effectuer différentes missions assez variées lors de mon passage dans les locaux de Wedia. Aucune mission spécifique ne m’a été donnée, il m’a juste été donné un cadre pour progresser dans différents domaines et permettre une montée en compétences.

III.1 Kubernetes

L’entreprise est en train de muter.

Actuellement, le déploiement d’un DAM pour un nouveau client se fait sur des machines virtuelles principalement hébergées sur le cloud Amazon Web Services (AWS).

Prochainement, les DAM seront hébergés toujours sur AWS mais via kubernetes (k8s).

Kubernetes est une solution open source qui permet de déployer, gérer et mettre à l’échelle des applications conteneurisées. Cela présente plusieurs avantages, notamment un déploiement accéléré, de la scalabilité ainsi qu’une disponibilité accrue. En effet, les services peuvent se relancer automatique et lorsque ce n’est pas possible, pour peu que les données soient accessibles sur un SAN ou autre, relancer un service revient à relancer un docker-compose.

Un cluster k8s est composé de différents nodes qui éxécutent des pods.
Les pods sont la plus petite unité dans k8s : ils contiennent un ou plusieurs conteneurs et partagent les mêmes ressources réseau et de stockage.
Chaque pod possède sa propre adresse IP et peut être identifié par un nom.

Il existes deux types de nodes : le master node (ou control plane) et les worker nodes. Le premier sert à contrôler les seconds.

Les nodes ont eux même différents composants autres que les pods :

- Kubelet : l’agent qui communique avec le control plane et s’assure que les pods tournent correctement.
- Kube-proxy : gère les règles réseau et permet la communication interne et externe des pods.
- Container runtime (Docker, containerd, CRI-O…) : lance et exécute les conteneurs

Les nodes jouent également le rôle de load balancers pour répartir la charge lors d’un fonctionnement en cluster d’applications.

Ces composants travaillent de concert pour garantir que les applications déployées sur le cluster fonctionnent de manière fiable et efficace. Ils permettent également au cluster de rester flexible et de s’adapter aux variations de charge ou aux interruptions, tout en offrant un environnement stable pour les applications et les services conteneurisés. Cette organisation assure que chaque node peut gérer plusieurs pods simultanément, surveiller

leur état et appliquer automatiquement des correctifs ou des redémarrages si nécessaire, ce qui est essentiel pour maintenir la disponibilité et la performance des services.

Il m'a fallu tout d'abord comprendre les différents concepts présents derrière k8s. Je me suis donc appuyé sur la documentation kube disponible en ligne mais également de nombreux tutoriels, notamment ceux Tech World with Nana qui rend disponible beaucoup de documentation pour les DevOps et donc beaucuop de tutoriels dockers ou kubernetes.

Dans un second temps j'ai aussi du me familiariser avec les services AWS sur lesquels sont et seront hébergés les services de l'entreprise.

Prise en main d'AWS

AWS propose de nombreux services d'hébergement, et dispose de plusieurs data center partout dans le monde. La tarification des services dépend du service en lui même bien sûr, mais aussi de sa localisation. Par exemple, le data center de Dublin coutera souvent moins cher qu'un data center situé en France.

Les principaux services que j'ai pu utiliser étaient :

- IAM (Identity and Access Management) : gestion des utilisateurs, permissions et sécurité
- EKS (Elastic Kubernetes Service) : service managé pour déployer et gérer des clusters Kubernetes
- SAN : systèmes de stockage partagés pour que plusieurs instances ou pods accèdent aux mêmes données
- S3 Glacier / S3 Glacier Deep Archive : stockage d'archives à long terme, économique et sécurisé, récupération lente car surt bande magnétique

La facturation de ces services se fait soit au nombre d'accès, soit à la bande passante, soit les deux.

Il est donc important de bien scaler un projet avant de le déployer sur le cloud Amazon pour éviter les surprises de facturation. Une fiche des tarifs est disponible et permet de budgétiser l'utilisation en vue de la refacturer au client final ou juste d'héberger ses services dans le cloud.

Cela peut amener à se poser la question d'un hébergement en cloud, car beaucoup d'entreprises pourront se contenter d'un serveur classique déployé localement, notamment pour du stockage ou de l'hébergement de services « légers ». Une économie est donc ici réalisable en fonction des entreprises et de l'échelle de celles-ci.

Une fois mes identifiants créés et une fois connecté à AWS, j'ai entrepris de me faire une cheatsheet sur les différentes commandes AWS et Kubernetes pour pouvoir naviguer plus aisément. Voici les principales commandes que j'ai pu utiliser lors de mon passage chez Wedia :

```
====================
1. CONFIGURATION AWS
====================

# Configure l'AWS CLI (profil par défaut)
aws configure

# Configure un profil spécifique
aws configure --profile monprofil

# Vérifier les identifiants utilisés
aws sts get-caller-identity
aws sts get-caller-identity --profile monprofil

# Voir les fichiers de configuration
cat ~/.aws/credentials
cat ~/.aws/config

====================
2. EKS - CLUSTER K8S
====================

# Lister les clusters EKS
aws eks list-clusters

# Ajouter le cluster à kubeconfig
aws eks update-kubeconfig --region eu-west-1 --name mon-cluster

# Avec un profil AWS
aws eks update-kubeconfig --region eu-west-1 --name mon-cluster --profile monprofil

# Vérifier l'URL du cluster Kubernetes
kubectl config view -o jsonpath='{.clusters[*].cluster.server}'

# Tester l'accès au cluster
kubectl cluster-info
kubectl get nodes

====================
3. KUBECTL - DE BASE
====================

# Voir les noeuds du cluster
kubectl get nodes

# Voir les pods (namespace courant / tous)
kubectl get pods
kubectl get pods --all-namespaces

# Voir les services
kubectl get svc

# Voir tous les objets (namespace courant)
kubectl get all

# Changer de namespace
kubectl config set-context --current --namespace=mon-namespace

====================
4. INTERACTION PODS
====================

# Décrire un pod
kubectl describe pod nom-du-pod

# Voir les logs d'un pod
kubectl logs nom-du-pod

# Exec dans un pod (terminal interactif)
kubectl exec -it nom-du-pod -- /bin/bash

# Port forwarding
kubectl port-forward svc/mon-service 8080:80

====================
5. DÉPLOIEMENTS YAML
====================

# Appliquer un fichier YAML
kubectl apply -f fichier.yaml

# Supprimer des ressources via YAML
kubectl delete -f fichier.yaml

# Recharger un déploiement (rollout restart)
kubectl rollout restart deployment nom-de-deployment

# Voir les événements récents
kubectl get events --sort-by='.metadata.creationTimestamp'
```

```
====================
6. DEBUG KUBECONFIG
====================

# Voir tous les contextes
kubectl config get-contexts

# Changer de contexte
kubectl config use-context nom-du-contexte

# Voir la config complète
kubectl config view

# Obtenir l'IP/API server (souvent utile pour debug)
kubectl cluster-info

====================
7. S3 RAPIDE
====================

# Lister les buckets
aws s3 ls

# Uploader un fichier
aws s3 cp monfichier.txt s3://mon-bucket/

# Télécharger un fichier
aws s3 cp s3://mon-bucket/monfichier.txt ./

====================
8. EC2 RAPIDE
====================

# Lister les instances
aws ec2 describe-instances

# Démarrer / arrêter une instance
aws ec2 start-instances --instance-ids i-xxxxxx
aws ec2 stop-instances --instance-ids i-xxxxxx

====================
9. ECR (Docker Registry)
====================

# Login à ECR
aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin
<account>.dkr.ecr.eu-west-1.amazonaws.com

# Lister les dépôts
aws ecr describe-repositories

# Tag et push une image Docker
docker tag monimage:latest <account>.dkr.ecr.eu-west-1.amazonaws.com/monrepo:latest
docker push <account>.dkr.ecr.eu-west-1.amazonaws.com/monrepo:latest

==========================
10. RÉCUPÉRER LES ADRESSES IP
==========================

# IP d'un POD
kubectl get pod nom-du-pod -o wide
kubectl get pods -o wide

# IP d'un NODE
kubectl get nodes -o wide
kubectl describe node nom-du-node

# IP d'un SERVICE
kubectl get svc nom-du-service -o wide
kubectl describe svc nom-du-service

# Pour voir les IPs de tous les services
kubectl get svc

# Exemple de sortie pour un LoadBalancer:
# NAME         TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)
# mon-service  LoadBalancer   10.0.0.123      34.201.45.67    80:32112/TCP
```

Prise en main de Kubernetes

Le déploiement d’un cluster k8s se fait par le biais d’un fichier .yaml, à la manière d’un docker compose.

Une fois de plus, j’ai synthétisé la structure autour d’un squelette réutilisable : cela m’a permis de gagner du temps car la strucutre d’un fichier yaml est rigide. L’indentation est primordiale d’une part, mais l’enchaînement de commandes aussi. Avoir un squelette de déploiement m’a permis de modifier plus rapidement le fichier que l’on m’a fourni et donc de comprendre plus rapidement le fonctionnement. Voici mon exemple de yaml :

```
# ==========================
# 11. EXEMPLES FICHIERS YAML
# ==========================

# POD SIMPLE
---
apiVersion: v1
kind: Pod
metadata:
  name: mon-pod
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80

# DEPLOYMENT
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mon-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      app: mon-app
  template:
    metadata:
      labels:
        app: mon-app
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80

# SERVICE TYPE ClusterIP (par défaut, interne au cluster)
---
apiVersion: v1
kind: Service
metadata:
  name: mon-service
spec:
  selector:
    app: mon-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80

# SERVICE TYPE NodePort (accessible depuis l'extérieur via un port du node)
---
apiVersion: v1
kind: Service
metadata:
  name: mon-service-nodeport
spec:
  type: NodePort
  selector:
    app: mon-app
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30080

# SERVICE TYPE LoadBalancer (nécessite un cloud provider)
---
apiVersion: v1
kind: Service
metadata:
  name: mon-service-lb
spec:
  type: LoadBalancer
  selector:
    app: mon-app
  ports:
  - port: 80
    targetPort: 80
```

```yaml
# INGRESS (avec annotation pour nginx ingress controller)
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: mon-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: monapp.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: mon-service
            port:
              number: 80
```

On peut distinguer le « deployment » des « services » :

- Pod : la plus petite unité déployable, contenant un ou plusieurs conteneurs et partageant ressources réseau et stockage.
- Deployment : définit la manière dont les pods doivent être créés et gérés. Il précise le nombre de réplicas, l'image du conteneur et les stratégies de mise à jour.
- Service : expose les pods et gère la communication réseau. Types principaux :
  - ClusterIP : service interne au cluster, accessible uniquement depuis d'autres pods
  - NodePort : expose le service sur un port spécifique de tous les nodes pour un accès externe simple
  - LoadBalancer : crée un point d'accès externe via un load balancer (souvent utilisé sur le cloud)
- Ingress : gère l'accès HTTP/HTTPS externe aux services du cluster, avec routage basé sur l'URL ou le nom de domaine

Face à la complexité de k8s, un fichier de ce type m'a grandement aidé à faire face à un fichier de configuration déjà en place. J'ai pu appréhender le document, m'y retrouver, et lancer moi même un mini kluster sur les services AWS.

III.2 Remise en route des serveurs

Les locaux montpelliérains de Wedia regorgent de vieux matériel.

Certains sont aussi obsolètes que mise au rebus, comme le dénommé Zeus qui fonctionne sous disquettes et dispose d'une RAM impressionnante de 256mb, le tout piloté par windows 95.

D'autres sont des machines certes âgées mais qui peuvent encore exécuter des tâches simples, comme les deux Dell PowerEdge T300 qu'il m'a été donné de remettre en fonctionnement.

Les serveurs sur place servent principalement à faire tourner des fonction lambdas sans encombrer les machines de travail. Le travail de fond peut être lancé sur ces appareils et libérer de la puissance sur les machines de bureau utilisées tout au long de la journée.

Ces machines étaient entreposées dans la « baie de serveurs » (une étagère dans une pièce séparées, reliée à un switch) car tombés en panne il y a plusieurs années. Elles sont à l'origine dotées d'un processeur Intel Xeon quad core 2.5Ghz, de 8Go de ram ECC en DDR2 ainsi que d'une carte RAID physique associée à 4 disques durs de 256Go tournant à 5400rpm pour un total de 512Go en raid 5.

Ce sont des serveurs au format tour, semblables à des PC au format ATX, mais avec des caractéristiques destinées à un usage en tant que serveurs : carte RAID physique, processeur XEON, et ventilateurs tournant jusqu'à 7000rmp (autour de 4000 lors de charge faible). L'optimisation de l'airflow est faite par des caches en plastique acheminant l'air des ventilateurs vers les composants critiques. Les ventilateurs sont bien sûr très bruyants et j'ai donc choisi de les débrancher lorsque je travaillais sur les machines, dans un soucis de ne pas perturber le travail des collègues présents. J'ai donc bien sûr ignoré les avertissements relatifs aux ventilateurs lors de l'installation des machines.

Il était possible d'accéder au bios des machines mais impossible d'aller plus loin.

Premiere machine

La première machine affichait plusieurs erreurs dans les menus du bios, personnalisé par Dell : batterie CMOS hors service, et groupe Raid inutilisable… Plusieurs tentatives de restaurer le RAID ont échouées, car le système ne pouvait reconstruire la base de donnée. Après avoir tenté d'échanger les câbles entre les machines et n'avoir constaté aucune évolution, le problème devait venir d'un des disques durs.

J'ai donc laissé tourner la machine et constaté qu'un des disques durs chauffait bien plus que les autres, signe d'un malfonctionnement.

J’ai également trouvé le menu de contrôle du RAID et pu constater qu’un des disques n’était pas reconnu… et donc que les contrôles lors du démarrage renvoyaient une erreur et empêchaient le système de continuer.

Après avoir enlevé le disque défectueux (et un autre, car la machine ne supporte que le raid 1 et 5), j’ai pu reconstruire le raid et lancer un boot de la machine.

J’ai donc voulu installer l’os que les techniciens installent par défaut : ubuntu. Après téléchargé l’iso et flashé une clé à l’aide de rufus, j’ai pu lancer le boot de la machine via la clé USB pour installer Ubuntu.

Malheureusement, le serveur plantait et je ne pouvais pas accéder à l’installation, sous forme graphique ou non.

J’ai donc essayé une version plus ancienne (20.4 alors que nous sommes aujourd’hui en 24.xx) avec le même résultat.

Ubuntu étant un dérivé du système Debian, j’ai donc re-formaté ma clé avec Debian Bookworm (12) netinst, partant du principe que Debian offrirait certainement une meilleur compatibilité avec un système ancien. En effet, ce système d’exploitation est un gage de stabilité, parfois au détriment de nouveautés car les release « stable » accusent un certain retard vis à vis des derniers noyaux (kernel), rendant souvent les systèmes récents incompatibles nativement (même s’il est possible de contourner le problème en installant un kernel plus récent, au risque de voir des incompatibilités).

Pari gagnant, j’ai donc pu installer Debian avec un environnement de bureau LXDE. J’ai choisi une installation graphique légère et compatible avec les machines anciennes et limitées en puissance. J’aurais aussi pu me passer d’une installation graphique, la machine étant principalement contrôlée via SSH, mais je souhaitais me rapprocher au maximum de la configuration ubuntu qui était utilisée de base sur la machine.

J’ai effectué une configuration simple de la machine, avec serveur ssh, fixé l’IP pour la retrouver facilement sur le réseau et installé des paquets indispensables comme Docker ou htop.

J’ai donc pu remonter l’appareil, et suis allé l’installer à coté de la machine utilisée actuellement pour les tests (un autre serveur DELL au format tour, beaucoup plus récent et performant).

Seconde machine

Maintenant plus expérimenté, j’ai pu m’attaquer à la seconde machine de manière plus ordonnée. J’ai donc directement changé la pile sur la carte mère, et suis allé dans le menu de gestion des disques du bios.

Tous les disques étaient reconnus mais le raid n’arrivait pas à reconstituer l’image de démarrage. La faute sans doute à la pile du CMOS déchargée qui n’alimentait donc plus la carte de contrôle du RAID, ce qui a engendré une dégradation de ce dernier. Potentiellement

une corruption silencieuse a pu avoir lieu, mais la machine ne disposant pas de données importantes, formater les disques et reconstruire un nouveau groupe RAID était la solution la plus rapide et efficace pour retrouver une machine fonctionnelle dans les meilleurs délais.

Malgré les faibles spécifications de la machine, formater un groupe raid à base de 4 HDD de 256Go a été rapide et j'ai donc pu installer la même configuration que la machine précédente.

L'ayant déjà fait auparavant, la configuration de la machine a été faite rapidement et j'ai pu l'installer aux cotés de l'autre.

Résultats

Les deux machines sont maintenant reliées au réseau local et sont accessibles depuis le réseau interne via SSH. Elles remplissent leur fonction de force de calcul d'appoint.

C'est important de noter que même de vieilles machines, au caractéristiques aujourd'hui perçues comme obsolètes, peuvent faire tourner des applications, des fonctions, ou trouver une fonction dans un environnement domestique ou de bureau.

On peut alors se poser la question de l'obsolescence quand des machines proposant des performances brutes correctes ne peuvent plus faire tourner un système d'exploitation comme Windows 11, car l'installation normale nécessite une puce TPM2.0 ainsi que 8Go de ram pour fonctionner correctement. Il est donc possible d'installer un os moins gourmand sur de vieilles machines et de leur redonner vie dans une fonction moins gourmande, comme un serveur de fichier dans une TPE/PME ou un serveur multimédia domestique.

Le nombre de possibilité pour exploiter de vieux appareil sont trop nombreuses pour être toutes citées.

III.3 Prise en main d’Opensuse

OpenSuse Tumbleweed est la version rolling release d’OpenSuse. C’est la distribution principalement utilisée chez Wedia, et les ordinateurs fournis sont souvent équipés de celle-ci.

Contrairement à Debian ou autre en release fixe, il n’y pas de versions tous les 6 mois ou 1 an à mettre à jour, le système est constamment mis à jour. On reçoit donc les dernières versions des logiciels et du noyau Linux dès qu’elles sont stables. C’est idéal pour les développeurs car on a donc toujours les dernières bibliothèques, outils de développement et environnements graphiques. Tumbleweed est testée via openQA avant chaque mise à jour, ce qui limite les plantages.

OpenSuse utilise zypper pour les paquets et YaST pour la configuration système, et il est bien sur possible d’utiliser flatpak pour compléter la liste des paquets.

Faisant d’habitude tourner mes machines sous Debian avec un bureau KDE, OpenSuse est un changement mitigé pour ma part : outre mes habitudes chamboulées par un format aussi similaire que différent, j’ai trouvé plusieurs avantages et plusieurs défauts à l’utilisation.

Il est à noter qu’OpenSuse tumbleweed est pensé comme une distribution « de bureau », pour une machine de travail par exemple.

Le principal avantage dans mon cas était la nouveauté du noyau. En effet, mon pc portable disposant d’une carte réseau « récente », elle n’était pas reconnue par Debian 12 car son kernel est antérieur à la sortie de cette carte. Il faut donc faire un rétroportage du noyau et c’est donc s’exposer à des bugs et à des incompatibilités.

OpenSuse s’est installé sans problème, et pouvoir prendre un version netinst d’une distribution et ne pas avoir à effectuer des manipulations complexes, laborieuses et risquées pour avoir accès au wifi est pour moi un point fort de cette distribution. On peut donc installer facilement OpenSuse sur du matériel récent pour remplacer Windows.

Installation de paquets .deb

Le problème s’est par contre posé pour les paquets. Bien que zypper soit bien fourni, certains paquets dont je me sers quotidiennement ou dont j’ai besoin dans le cadre de mes études ne sont disponibles qu’en .deb, et donc incompatibles avec OpensUse.

Pour rappel, un fichier .deb est un fichier de configuration compressé avec une architecture toujours similaire. On peut décomposer le de cette façon :

- debian-binary : fichier texte qui indique la version du format du paquet
- control.tar.gz : contient les métadonnées du paquet (nom, version, dépendances, scripts de pré/post installation)

- data.tar.gz (ou .xz) : contient les fichiers réels du logiciel qui seront installés sur le système (binaires, bibliothèques, configuration, etc.)

Il est donc tout de même possible d'installer certains .deb sur OpenSuse. En effet, les systèmes Linux reposent sur une hiérarchie de répertoires standards remplis de fichiers textes dans différents formats : /usr/bin pour les exécutables, /usr/lib pour les bibliothèques, /etc pour les fichiers de configuration, etc.
En décompressant un fichier .deb, on retrouve cette même organisation. Il suffit alors de copier manuellement les fichiers extraits dans les bons répertoires du système.

Cette méthode est assez artisanale et doit être utilisée avec précaution, car elle ne gère pas les dépendances ni les mises à jour. Mais dans certains cas précis (logiciels non disponibles sous zypper ou flatpak), elle permet de forcer l'installation et de rendre l'application utilisable sur OpenSuse.

Certains paquets propriétaires sont par contre impossibles à installer, notamment Cisco Packet Tracer pourtant indispensable à la poursuite de mes études.

Prise en main de Wine et Winetricks

Une solution pour installer des paquets est de passer par Wine.

Wine permet de lancer une machine virtuelle Windows directement sur le système, et donc d'installer des .exe et de les lancer comme une application normale.

Winetricks permet de customiser des profils Wine, notamment pour adapter la version de windows ou pour installer diverses dépendances. Certains .exe ne tournent en effet que sur une version de windows précise, ou d'autres nécessitent des dépendances exotiques. Il est possible d'installer différents profils Wine et de faire cohabiter plusieurs configurations.

La configuration de Wine est comme beaucoup de choses sous linux : des fichiers textes dans une hiérarchie de dossier. Lors de la création d'une configuration, un profil dans /home/user est créé sous .wine.

Il s'agit donc d'une solution pour rendre OpenSuse plus versatile en venant étoffer le catalogue d'applications disponibles. Combiné à flatpak et à l'installation de .deb vue plus haut, on peut donc bénéficier d'un système récent constamment mis à jour avec la majorité des applications disponibles.

IV Mon ressenti dans et sur l’entreprise

J’ai tout d’abord découvert un cadre bienveillant.

Bien que n’ayant pas de mission fixe, les personnes sur place ont pris le temps de répondre à mes questions et de trouver des sujets pour me faire progresser. J’ai en effet conscience qu’avec leurs 30ans d’expérience dans le métier et des diplômes d’ingénieurs, je n’ai pas pu leur apporter une grande aide.

J’ai tout de même pu remettre en service les deux serveurs ce qui constitue une contribution modeste, mais une contribution.

J’ai de plus été dans un cadre très libre : pas d’horaire d’arrivée, pas d’horaire de sortie (outre celle du dernier car je n’avais bien sûr pas les clefs!). Chacun se gère donc mais cette souplesse permet d’éviter un stress parfois inutile lorsque les transports sont bloqués ou qu’un impératif se fait savoir dans l’après midi. Le fait de faire confiance aux collaborateurs permet selon moi que le travail soit fait dans de bonnes conditions, **sans faux semblants**.

C’est un constat que j’ai déjà fait dans le monde du travail, certaines entreprises sont plus regardantes des apparences que du travail réellement effectué, et évoluer dans un cadre à l’opposé de ces à priori est quelque chose de fort appréciable et m’a conforter dans l’idée de continuer dans cette lancée.

J’ai également pu mesurer l’expertise des membres présent, qui connaissent leur sujet lorsqu’ils parlent de développement et naviguent avec aisance dans des concepts que j’appréhende difficilement.

Vis à vis de l’entreprise, j’ai eu le sentiment qu’ils cherchent à fournir un produit personnalisé aux clients, pour correspondre au mieux à ses attentes. C’est une approche à contre courant de certains gros acteurs, et je salue cette initiative dans un monde qui tend naturellement à tout réguler et formater pour effacer la complexité de l’individualité.

V Conclusion

Ce stage m’a permis de découvrir le fonctionnement concret d’une organisation et de mettre en pratique mes connaissances dans un contexte professionnel. J’ai pu approfondir certains aspects techniques (déploiement d’applications, gestion des environnements, suivi de procédures) tout en développant des compétences transversales comme l’autonomie, la rigueur et la communication.

Au-delà de la dimension purement technique, cette expérience m’a montré l’importance du travail en équipe, de la documentation et de l’adaptabilité face aux imprévus. Elle constitue une étape utile dans mon parcours de formation et contribuera à mieux préparer mon intégration future dans le monde professionnel.

Je tiens enfin à remercier l’entreprise pour son accueil ainsi que l’équipe pédagogique pour l’accompagnement.