Maintenance de stockage via PowerShell

Une image contenant automate, machine, jouet, caméra

Description générée automatiquement

Figure - Image générée par Bing Copilot représentant le projet

Yann Mangiagli – FIN2

Vennes

88H

Gaël Sonney

Experts : Daniel Berney, Gabriel Maret

Table des matières

[1 Résumé de la documentation / résumé du rapport 4](#_Toc168316866)

[1.1.1 Situation de départ 4](#_Toc168316867)

[1.1.2 Mise en œuvre 4](#_Toc168316868)

[1.1.3 Résultat 4](#_Toc168316869)

[2 Spécifications 5](#_Toc168316870)

[2.1 Titre 5](#_Toc168316871)

[2.2 Description 5](#_Toc168316872)

[2.3 Matériel et logiciels à disposition 5](#_Toc168316873)

[2.4 Prérequis 5](#_Toc168316874)

[2.5 Cahier des charges 6](#_Toc168316875)

[2.5.1 Objectifs et portée du projet (objectifs SMART et CQQCOQP) 6](#_Toc168316876)

[2.5.2 Caractéristiques des utilisateurs et impacts 6](#_Toc168316877)

[2.5.3 Fonctionnalités requises (du point de vue de l’utilisateur) 7](#_Toc168316878)

[2.5.4 Contraintes 7](#_Toc168316879)

[2.5.5 Travail à réaliser par l'apprenti 7](#_Toc168316880)

[2.5.6 Si le temps le permet … 8](#_Toc168316881)

[2.5.7 Méthodes de validation des solutions 8](#_Toc168316882)

[2.6 Les points suivants seront évalués 8](#_Toc168316883)

[2.7 Validation et conditions de réussite 8](#_Toc168316884)

[3 Planification Initiale 8](#_Toc168316885)

[3.1.1 Méthode des 6 pas 8](#_Toc168316886)

[3.1.2 Mises à jour du planning 9](#_Toc168316887)

[3.1.3 Diagramme de Gantt 10](#_Toc168316888)

[3.1.4 Bilan graphique 15](#_Toc168316889)

[4 Analyse 16](#_Toc168316890)

[4.1 Opportunités 16](#_Toc168316891)

[4.1.1 Approfondir 16](#_Toc168316892)

[4.1.2 Difficultés potentielles 16](#_Toc168316893)

[4.1.3 Matériel à exploiter 16](#_Toc168316894)

[4.1.4 Recherche d’informations particulière 16](#_Toc168316895)

[4.1.5 Solutions possibles 16](#_Toc168316896)

[4.2 Document d’analyse et conception 17](#_Toc168316897)

[4.2.1 Schéma de conception 17](#_Toc168316898)

[4.2.2 Script 1 17](#_Toc168316899)

[4.2.3 Script 2 21](#_Toc168316900)

[4.3 Conception des tests 23](#_Toc168316901)

[4.4 Planification détaillée 23](#_Toc168316902)

[5 Réalisation 27](#_Toc168316903)

[5.1 Dossier de Réalisation 27](#_Toc168316904)

[5.1.1 Version des outils logiciels utilisés 27](#_Toc168316905)

[5.1.2 Comptes utilisés 28](#_Toc168316906)

[5.1.3 Configuration spéciale des serveurs 28](#_Toc168316907)

[5.1.4 Déploiement et exécution détaillée 32](#_Toc168316908)

[5.1.5 En tête commune aux 2 scripts 33](#_Toc168316909)

[5.1.6 Réalisation du script de setup 33](#_Toc168316910)

[5.1.7 Réalisation du script de maintenance 37](#_Toc168316911)

[5.2 Modifications 42](#_Toc168316912)

[6 Tests 42](#_Toc168316913)

[6.1 Dossier des tests 42](#_Toc168316914)

[7 Conclusion 43](#_Toc168316915)

[7.1 Bilan des fonctionnalités demandées 43](#_Toc168316916)

[7.2 Bilan de la planification 44](#_Toc168316917)

[7.3 Bilan personnel 44](#_Toc168316918)

[7.3.1 Remerciements 45](#_Toc168316919)

[8 Divers 45](#_Toc168316920)

[8.1 Journal de travail 45](#_Toc168316921)

[8.2 Webographie 51](#_Toc168316922)

[8.3 Table des illustrations 52](#_Toc168316923)

[9 Glossaire 54](#_Toc168316924)

[10 Annexes 55](#_Toc168316925)

# Résumé de la documentation / résumé du rapport

### Situation de départ

Deux machines distantes et une machine locale sont reliées par Ethernet sur un réseau LAN. Deux scripts doivent être créés, un pour la maintenance du stockage des pc, l’autre pour sa mise en place. Le script de maintenance, appelé script 1 dans ce rapport, permet de supprimer les fichiers OVA de plus de 3 Go. Le script de mise en place, surnommé script 2, sert à copier à distance le script 1 en utilisant les adresses IP du réseau en paramètre pour identifier l’ordinateur de destination.

### Mise en œuvre

La mise en œuvre du projet commence par un travail fait en amont en paramétrant 2 serveurs Windows Server 2022, qui ont été reliés via le réseau de l’école par câble Ethernet. Les pare-feux ont ensuite été désactivés et le Windows remote activé (avec ajout en hôte trusté via PowerShell). Par la suite une planification initiale a été réalisée, ainsi qu’un schéma illustrant comment les scripts vont procéder. Les scripts sont ensuite réalisés petit à petit et postés sur GitHub afin d’avoir un versioning. Le rapport est rédigé en parallèle du projet, comme le journal de travail.

### Résultat

Les 2 scripts fonctionnent, malgré certains chemins en dur pour la copie du script. Le script 2 copie le script 1 dans le dossier C:\Scripts que le script 2 crée si le répertoire n’existe pas déjà. Le script 2 crée une tâche planifiée à 16h qui lance le script de maintenance tous les jours à l’heure convenue. Le script 1 analyse le stockage de l’ordinateur sur lequel il est. SI le taux de stockage atteint plus de 80%, il nettoie le disque en supprimant tous les fichiers OVA de plus de 3Go du disque C. Il envoie dans le même temps un mail d’avertissement à l’adresse électronique tpiymetml@gmail.com. Le mail contient la version de l’OS, la version des mises à jour, l’espace libre en octets et l’espace utilisé en octets.

# Spécifications

## Titre

Maintenance automatique et régulière du stockage des postes de travail à distance

## Description

Les étudiants de la section informatique sont amenés à utiliser de nombreuses machines virtuelles pendant leur formation qu’ils et elles laissent sur leur lecteur (H :) qui leur servent de répertoire utilisateur sans les utiliser de nouveau. Ces machines virtuelles sont donc stockées en vain sur les machines de l’ETML plutôt que sur le SSD des étudiants et prennent beaucoup de place avec le temps. Le but de ce projet est de limiter l’impact des machines virtuelles sur le stockage des postes de l’école en avertissant quand le stockage d’un poste atteint 80% et en envoyant un mail à l’administrateur qui pourra identifier la machine pendant que le script supprime les fichiers OVA de plus de 3Go des répertoires des élèves s’étant connecté à cette machine.

## Matériel et logiciels à disposition

PC standard de l’ETML sous Windows 10

PowerShell

PowerShell ISE

Deux machines distantes (WINSERV2022) pour le déploiement du script à distance avec les droits d’administrateur configurées par l’apprenti en amont.

## Prérequis

Module 431 - Planification de projets et de tâches

Module 122 – PowerShell

Module 117 – Réseau (Windows Server 2022, Câblage simple, Remote Desktop)

## Cahier des charges

### Objectifs et portée du projet (objectifs SMART et CQQCOQP)

**CQQCOQP**

Combien : Le projet ne coûte rien à produire, il rapporte de l’efficacité en supprimant une tâche redondante et permet aux administrateurs système de faire autre chose et augmenter leur productivité. Cela consomme peu car la tâche est faite une fois par jour brièvement, mais il ne faut pas oublier que les mails restent dans une boite mail et qu’une trop grande quantité est polluante. C’est pour cela qu’il faut se débarrasser des mails qui ne sont plus utiles.

Quoi : Les actions à effectuer sont la création de 2 scripts différents. L’un des scripts permet d’installer l’autre, pendant que celui-ci surveille le stockage des élèves.

Qui : Un administrateur système va réaliser le projet. Cela concerne les élèves ayant des machines virtuelles sur leur disque et les administrateurs système, pour faciliter la vie des administrateurs système du service informatique de l’ETML.

Comment : Utilisation des compétences acquises durant le CFC.

Où : Les ordinateurs distants sont à l’étage au-dessus de la salle A13 dans la salle des serveurs de Vennes. La machine locale est en salle A13.

Quand : Du 13 mai au 3 juin 2024.

Pourquoi : Améliorer la productivité et soulager la charge de travail du service informatique de l’ETML tout en libérant de l’espace de stockage pour les élèves.

**SMART**

**S**pécifique : Réaliser 2 scripts dont un capable de copier l’autre script sur une autre machine à distance, pendant que l‘autre script analyse le stockage du pc et envoie des informations par mail sur le pc sur lequel s’exécute le script.

**M**esurable : Chaque jour le progrès est enregistré sur un journal de travail, puis tous les mardi et vendredi un rendu « complet » est fait.

**A**tteignable : Utilisation des compétences apprises durant le CFC.

**R**éaliste : Tous les élèves doivent quand même pouvoir poser des fichiers volumineux sans que ceux-ci ne se fassent supprimer s’ils ne sont pas des ova.

**T**emporel : Le projet doit être fait en 88H , du lundi 13 mai 2024 au lundi 3 juin 2024.

### Caractéristiques des utilisateurs et impacts

Les utilisateurs du script seraient des administrateurs système et réseau désirant optimiser le stockage des postes de travail supportant des machines virtuelles, comme le service informatique de l’ETML. Le but est donc d’automatiser les scripts tout en laissant aux administrateurs le moyen de personnaliser l’utilisation du script avec des paramètres. Par exemple, choisir la taille maximum tolérée par le script (par défaut 3Go comme mentionné plus haut).

### Fonctionnalités requises (du point de vue de l’utilisateur)

Pour le premier script, l’utilisateur doit pouvoir récupérer les informations concernant l’espace disque utilisé ainsi que l’espace libre, le nom de l’ordinateur, la version de son système, ainsi que les versions des mises à jour de l’OS. Le script envoie un courriel sur le mail d’entreprise de l’administrateur un avertissement quand 80% du stockage est utilisé, et doit pouvoir supprimer les fichiers trop volumineux, avec éventuellement un paramètre permettant de décider ce qui est considéré comme « trop volumineux ». Le deuxième script est plus simple et contient moins de fonctionnalités. L’utilisateur a juste besoin de le lancer pour qu’il copie le script 1 sur une machine à distance ne le possédant pas pour qu’il crée la tâche planifiée correspondante. Le script peut gérer une liste de 10 ordinateurs en simultané pour son exécution.

### Contraintes

Plusieurs contraintes peuvent exister avec des scripts comme ceux-là. Par exemple, le mail envoyé par le script a une chance non négligeable de finir dans les spam, car il n’y a pas le temps d’installer un serveur SMTP et c’est pour cela qu’une adresse Gmail sera utilisée, car il est plus simple et plus rapide de se connecter via PowerShell sur ce serveur SMTP. Comme la sécurité n’est pas mentionnée dans le cahier des charges, la sécurité du script ne sera pas forcément optimale car cela ne sera pas une priorité. Par exemple, pour automatiser l’enregistrement des identifiants en utilisant la commande « get-credential », il faudra passer par écrire le mot de passe en clair dans le script pour ne pas perdre du temps là-dessus. Cela pourrait être un point à améliorer s’il reste du temps quand les 2 scripts seront prêts.

### Travail à réaliser par l'apprenti

L’apprenti doit réaliser 2 scripts ainsi qu’un rapport, une planification et un journal de travail. Le premier script doit récupérer des informations système comme le nom, version de l’OS, Espace disque libre & utilisé, version des mises à jour et envoyer un mailcontenant les informations permettant d’identifier le poste de travail à l’administrateur quand le stockage utilisé égale ou dépasse les 80%. Il sera ensuite possible de supprimer en priorité les fichiers de machine virtuelles, en particulier les fichiers avec une extension ova. Le deuxième script est une automatisation. Il permet de sélectionner une liste jusqu’à 10 ordinateurs distants, et d’implémenter le premier script afin qu’il puisse scanner et surveiller le stockage des postes de travail sur lesquels il vient de s’installer, tout en créant une tâche planifiée qui planifiera ce scan. Le rapport doit contenir toute la documentation qu’il a créée durant le projet ainsi que les tests réalisés.

### Si le temps le permet …

Il serait judicieux d’améliorer la sécurité pour rendre le script utilisable par une entreprise ou une école, ainsi qu’installer un serveur SMTP pour ne pas avoir à utiliser une adresse Gmail mais une adresse eduvaud dans l’exemple de l’ETML utilisé comme prétexte pour ce TPI.

### Méthodes de validation des solutions

Chaque fonctionnalité demandée dans le script sera testée en premier lieu seule pour être sûr que la solution fonctionne avant de l’incorporer dans le script 1 ou 2 suivant l’instruction(s) testée(s). Selon la commande, elle devra être testée sur une machine distante, comme la commande de suppression des fichiers ova de 3Go. Le maximum de résultats de tests seront sujet à une capture d’écran montrant le résultat. Ils seront aussi documentés dans la partie réalisation du rapport.

## Les points suivants seront évalués

* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Planification Initiale

Le seul jour spécial est pentecôte, ce qui rend le lundi 20 mai 2024 chômé. Le lendemain, l’expert 2 vient voir l’avancée du projet vers 13h30.

### Méthode des 6 pas

Le projet commence le 13 mai 2024 et se termine le lundi 3 juin 2024. La méthode des 6 pas a été choisie pour ce projet. Cette méthode de planification consiste à se fixer des objectifs

**S’informer :**

Comprendre et enregistrer la tâche à accomplir, se renseigner sur le travail, sur ce qui est attendu, ce qui est « libre ». C’est dans ce moment-là qu’on se renseigne sur l’échéance, sur si on est capable ou non de réaliser ce projet, le matériel à disposition… Dans ce projet, on peut faire le parallèle avec la prise de connaissance du cahier des charges, les questions posées au chef de projet et aux experts en rapport avec ce document, la configuration des deux machines Windows server 2022 utilisées dans le projet comme machines distantes.

**Planifier :**

C’est tout ce qui concerne la planification du projet pour en arriver à bout. Il faut donc se fixer des priorités dans le projet, faire en premier lieu ce qui est le plus important comme les fonctionnalités principales du script avant la gestion des erreurs par exemple. En gros, quelles tâches réaliser en premier pour avoir le plus vite possible un prototype fonctionnel. Il est aussi décidé quelle méthode de gestion de projet est utilisée, comme agile, 6 pas, etc. Il faut aussi estimer le temps que chaque tâche prendra. C’est un exercice très difficile car on ne peut pas prévoir tous les problèmes qu’un projet va inévitablement rencontrer, ni leur gravité, comme une perte totale des données par exemple.

**Décider :**

Que le projet soit à réaliser seul(e) ou à plusieurs, il est impératif d’avoir un consensus avec tous les partis impliqués dans le projet. À plusieurs, il faut être d’accord entre collègues, pour pas que chacun fasse quelque chose de différent qui fasse juste perdre du temps.

**Réaliser / exécuter :**

Partie principale du projet. C’est à ce moment qu’on travaille sur le projet en le réalisant. Dans l’idéal, il faudrait que ça se déroule exactement comme dans la planification. Pour faire un parallèle avec ce projet, c’est la création des scripts.

**Contrôler :**

On regarde le travail réalisé plus haut et on le vérifie. On reteste, on relit la documentation et on la modifie si besoin.

**Évaluer :**

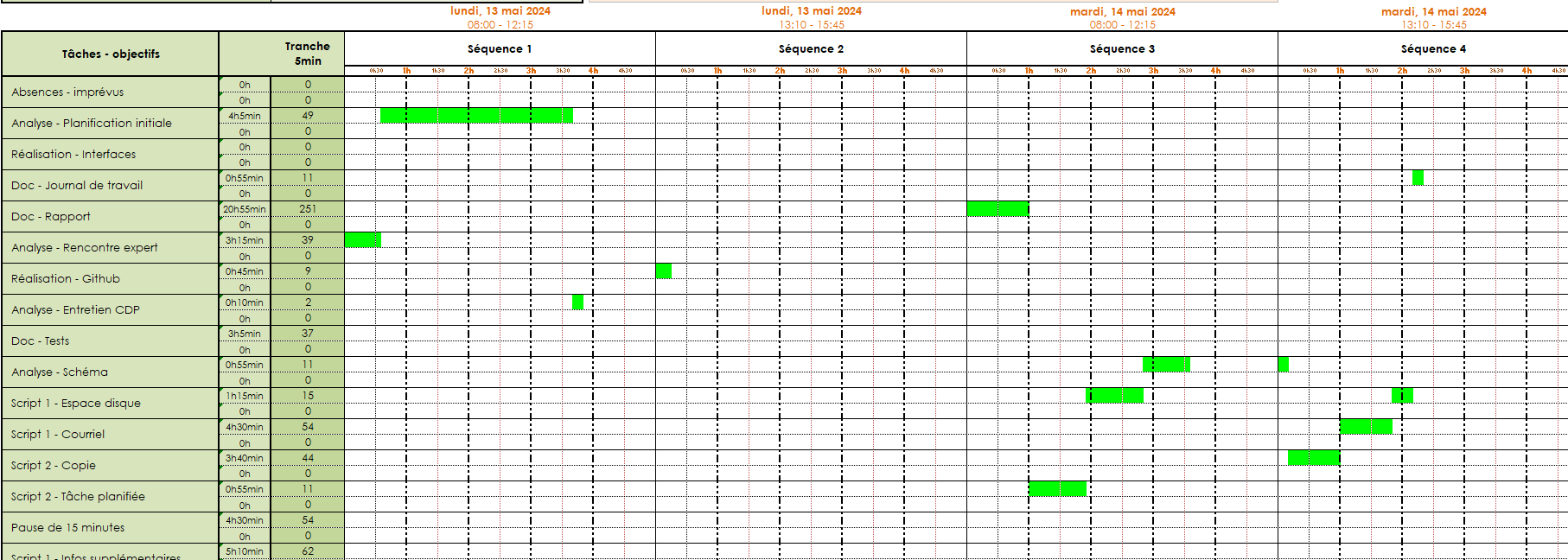
Passage en revue du projet effectué, on parle de ce qui a été bien fait, de ce qui a été mal fait, de ce qui aurait pu être amélioré. Dans ce rapport on pourrait assimiler cette étape au bilan des fonctionnalités demandées dans la conclusion.

### Mises à jour du planning

14.05.24 :

Modification de la rencontre avec expert 2, décidée arbitrairement dans la planification initiale par l’apprenti pour le 23 mai avant d’être modifié au 21 quand l’expert 2 et l’élève se sont mis d’accord.

### Diagramme de Gantt



Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

Une image contenant capture d’écran, ligne, Tracé, diagramme

Description générée automatiquement

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

Une image contenant capture d’écran, texte, ligne, Tracé

Description générée automatiquement

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

### Bilan graphique



# Analyse

## Opportunités

### Approfondir

Il faudra approfondir certaines connaissances en PowerShell pour mener à bien le projet.

* Commande « New-ScheduledTaskAction » à apprendre à utiliser
* Apprendre à exécuter les scripts depuis un autre ordinateur (Copy-Item -ToSession)

### Difficultés potentielles

Plusieurs difficultés peuvent survenir, surtout à cause d’imprévus.

* 1) Malgré le script d’envoi de mail fonctionnel réalisé en amont, rien ne promet qu’il fonctionnera dans un contexte différent
* 2) Comment vérifier que le script 1 s’exécute bien à 80% alors que les machines à disposition ont des disques vides ?
* 3) La copie du script sur une liste de machines n’a pas l’air simple à mettre en place
* 4) Pour le script de copie, il faut copier un script sur la machine principale et le coller sur la machine distante
* 5) Un serveur qui ne réagit pas comme l’autre
* Il faut internet pour envoyer un mail

### Matériel à exploiter

Le matériel exploitable peut lui aussi causer des difficultés, par exemple un serveur mal paramétré en amont, ou encore si le réseau de l’école est changé. C’est pour cela qu’il est impossible d’utiliser le script depuis une machine distante sur la machine locale, car elle a besoin de droit administrateur que je ne possède pas, et elle est protégée derrière docker kubernetes.

### Recherche d’informations particulière

Comme PowerShell et ses erreurs sont bien documentées, il y’a peu de chances de rencontrer des difficultés de ce côté-ci. En revanche, un problème d’environnement rendrait tout de suite la tâche plus ardue.

### Solutions possibles

* 2) Réduction du quota de 80% à 3% par exemple
* 3) Une boucle « foreach » sur un tableau ayant la liste des ordinateurs
* Supprimer Invoke-Command, car on essayait d’accéder à la session depuis la session, c’est donc normal que ça ne fonctionnait pas
* 5) Les pare-feux des 2 serveurs n’étaient pas identiques. Le premier était désactivé et pas l’autre.

## Document d’analyse et conception

### Schéma de conception

Une image contenant texte, diagramme, Plan, ligne

Description générée automatiquement

Figure - Schéma démontrant comment les scripts ont été pensés

### Script 1

Pour le script 1, des tests ayant été faits durant un projet précédent ont mis en exergue le fait qu’utiliser Send-MailMessage était obsolète et n’était plus supporté par Microsoft. Malgré le fait que des alternatives existent (MimeKit & MailKit), il a été décidé de rester sur cette façon de faire car elle est plus simple bien que moins sécurisée. Je suis parti du principe que vu que le script 2 permettra de copier le script à distance et de le faire s’exécuter périodiquement via une tâche planifiée, le script 1 n’a pas besoin d’être utilisable à distance ; il suffira de lancer le script 2 avant pour le copier sur la machine distante.

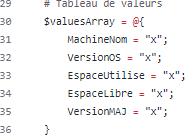


Figure - Création du tableau

Un tableau associatif est créé afin de lier chaque valeur récupérée dans le script pour pouvoir tout imprimer dans le mail d’un coup grâce à une boucle foreach. Bien qu’il soit possible de directement mettre les valeurs dans le tableau, j’ai préféré les rajouter plus tard afin d’avoir un code plus lisible.



Figure - Variable d'environnement du nom de l'ordinateur

Le nom de la machine est stocké dans une variable qu’on appelle variable d’environnement, d’où le $env : devant. Elles existent déjà dans le système, et la variable $env :COMPUTERNAME donne le nom de l’ordinateur qui exécute le script. Elle est utilisée ici pour être entrée dans le tableau associatif la liant à l’indice MachineNom.



Figure - Variables d'environnement de la version de l'OS

Même concept que pour la valeur de la ligne 39, mais avec deux valeurs. $env :OS donne le nom du système d’exploitation (dans le contexte de la machine sur laquelle je fais mes tests, cela équivaut à Windows NT, tandis que la seconde valeur contient le numéro de build de la version). Cela permet d’avoir un résultat plus esthétique, comme suit sans les guillemets : « Version de l’OS : Windows\_NT 19045 ».



Figure - Enregistrement de l'espace libre

Une valeur temporaire est créée à la ligne 45 qui sera réinstanciée plus tard pour économiser la mémoire en ne créant pas trop de variables inutiles. Cette instruction récupère l’espace libre stocké dans Get-PSDrive puis stocke la partie Free du tableau de PSDrive dans $valuesArray avec la clé EspaceLibre.



Figure - Enregistrement de l'espace utilisé

Comme la valeur temporaire a été utilisée, et que sa valeur a été inscrite dans le tableau, on peut la réutiliser sans perdre les données. Le script utilise donc les mêmes instructions que pour l’espace libre mais on remplace la valeur recherchée par Used. On aura donc l’espace utilisé dans les disques.



Figure - Enregistrement de l'historique des versions de mise à jour

La variable $tempValue est de nouveau réinitialisée en mettant une nouvelle valeur à l’intérieur qui représente la liste de toutes les mises à jour réalisées sur le système.



Figure - Calcul du stockage utilisé

Dans l’intervalle des lignes de l’image ci-dessus (56-57) se trouve une valeur permettant de lancer un mail même si le stockage n’atteint pas 80% à des fins de débogage. Pour la ligne 59, l’instruction renvoie plusieurs valeurs.



Figure - Contenu de $usedSpace

Il faudra donc rajouter un « .Sum » à la fin de $usedSpace pour avoir la valeur voulue.



Figure - Calcul de l'espace libre entre tous les disques

La même stratégie est appliquée pour l’espace libre.



Figure - Calcul du stockage total

Vient ensuite l’addition de la somme du stockage plein et du stockage vide afin d’avoir le stockage total pour calculer le pourcentage du stockage utilisé en octets, qui est ensuite stocké dans une variable.



Figure - Calcul en % de l'espace utilisé par rapport au stockage total

Le calcul pour récupérer le pourcentage est simple. On multiplie par 100 la somme du stockage utilisée, puis on divise par le stockage total. Par exemple, sur la machine de l’école en conditions réelles, le stockage est utilisé à environ 63.62%.



Figure - Condition pour faire fonctionner la suite du script

Le script effectue une vérification du pourcentage de stockage utilisé. S’il est de 80% ou plus, la suite du script s’exécute, sinon le script se termine. Tout ce qui suit est donc contenu dans le bloc de la condition. On pourrait améliorer les performances du script en faisant en sorte que le script aille chercher les infos que si le pourcentage de stockage autorisé et dépassé, comme ça pour la plupart du temps le script se fera plus vite.



Figure - Informations du serveur SMTP

On donne ici les premières informations importantes sur le serveur comme le port qui doit être à 587 pour fonctionner avec -UseSSL afin que Gmail permette d’envoyer le mail via leur serveur smtp.



Figure - Informations de l'expéditeur

Pour ce qui est de l’expéditeur et du destinataire, il s’agit juste de 2 strings classiques qui seront affectés par la suite dans le Send-MailMessage à la fin du script pour aérer un peu cette dernière commande.

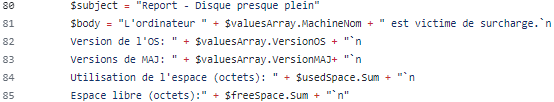


Figure - Sujet et corps de mail

Cette partie du code est la partie où le mail est rédigé $ body pourrait être optimisé avec une boucle foreach.



Figure - Mot de passe d'application Google utilisé

La variable du mot de passe est composée d’un mot de passe d’application généré aléatoirement par Google qui permet de se connecter à son compte Google depuis n’importe quelle application.



Figure - Sécurisation du mot de passe

Le mot de passe Google est converti en string sécurisé, un type créé exprès pour les mots de passe. Sans ça, Google considérera que la connexion n’est pas sécurisée même avec SSL et le mail ne s’enverra pas. Comme le mot de passe est en clair dans le code ce n’est pas sécurisé.



Figure - Création d'un objet de crédentité

Un objet spécial pour les logins est créé prenant en paramètre le nom d’utilisateur et le mot de passe d’un compte. Ici, le compte est l’adresse électronique, d’où le $sender qui est utilisé, suivi par le mot de passe d’application Google.



Figure - Envoi du mail via la commande Send-MailMessage

Envoi du mail en précisant le serveur smtp, son port, l’utilisation de SSL, l’adresse email et son mot de passe, puis une précision de l’expéditeur ainsi que du destinataire. Le sujet du mail et le corps sont définis aussi.



Pour finir, le script supprime les fichiers OVA de plus de 3 Go de tout le disque.



Le script se termine sur un message annonçant que le script est terminé.

### Script 2

Le deuxième script se focalise sur une façon d’utiliser le premier. En effet, celui-ci copie le premier script depuis la machine locale et le colle dans un répertoire spécifique sur la ou les machines distantes (jusqu’à un maximum de 10). C’est par celui-ci que le projet a été commencé, jugé plus simple à implémenter et tester.



Figure - Instanciation du tableau

Un tableau est créé comme paramètre afin de pouvoir mettre plusieurs adresses IP.



Figure - Instanciation d'un deuxième tableau

Un autre tableau d’une taille de 10 éléments est créé afin de pouvoir stocker les paramètres entrés dans $remoteip.



Figure - placeholder - boucle foreach

La boucle foreach permet de naviguer dans le tableau $remoteip et de récupérer toutes les adresses IP pour les ajouter dans $computerList mentionné ci-dessus. Out-Null est utilisé afin de masquer l’output fait par la boucle à chaque passage dans celle-ci.



Figure - Conversion du mot de passe

C’est dans cette partie du script que le mot de passe est choisi et stocké pour l’utiliser par la suite. Cette instruction cause des problèmes de sécurité car avoir un mot de passe en clair permet à n’importe quelle personne ayant accès au script de voir le mot de passe. Pour pallier à cette problématique, une solution serait de créer un fichier avec les mots de passe hachés à l’intérieur, puis de venir le récupérer avec une commande spécifique.



Figure - Récupération du nom d'utilisateur

Le nom d’utilisateur est récupéré ici. Comme le mot de passe et le nom d’utilisateur sont identiques pour les 2 machines distantes, ils sont renseignés en dur dans le script. Si ce n’était pas le cas, des paramètres auraient été utilisés. Cela crée un problème de sécurité similaire à celui du mot de mot de passe mentionné précédemment.



Figure - Implantation des chemins

Pour ce qui est des chemins, ils sont aussi en dur. Pour régler ce problème il faudra créer une valeur avec le nom du fichier et faire une concaténation avec $PWD pour rendre le script plus indépendant de la machine utilisée. $startpath est le chemin à optimiser, pour que le script respecte entièrement le cahier des charges. C’est lui qui va chercher le script 1 pour le copier plus tard. $destinationPath est le chemin de destination où sera stocké le script copié plus tôt. La lettre C:\ a été choisie car c’est la lettre par défaut d’un stockage. On pourrait améliorer ça en permettant à l’administrateur de choisir le chemin de destination via un paramètre.

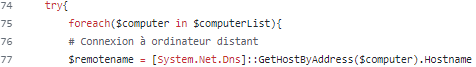


Figure - Début de la boucle

Cette boucle foreach commence par aller chercher le nom DNS des machines qui sera utile pour créer une session par la suite. C’est dans cette boucle qu’on utilise le tableau $computerList créé plus tôt.



Une session est créée ici pour chaque passage dans la boucle décidé par le nombre d’éléments dans le tableau $computerList cette session contient un id, un nom d’ordinateur, l’état de la connexion, le nom de la configuration qui s’apparente à comment la session a été configurée, et si la connexion et disponible ou non, comme le montre la capture d’écran ci-dessous.

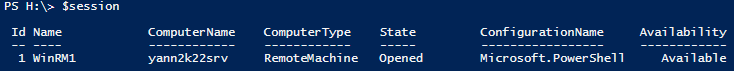


Figure - Informations d'une session ouverte



Cette instruction copie le fichier situé au chemin de départ et se connecte à l’ordinateur distant grâce au l’instruction -ToSession, puis colle le script dans le chemin de destination situé dans l’ordinateur distant.



Figure - Argument servant à lancer la tâche planifiée

Écriture du chemin complet du script dans une variable afin de pouvoir l’utiliser comme argument pour créer une tâche planifiée par la suite à l’intérieur de « Invoke-Command ». Celle-ci permet d’exécuter une commande sur la machine distante d’une manière plus efficace qu’Enter-PSSession, car Enter PSSession doit d’abord se connecter à la machine distante, tandis que invoke-command permet de rester sur la machine locale. Les guillemets simples permettent d’écrire les guillemets doubles et que ce ne soit pas considéré comme la façon d’écrire un string.



Figure - Création de la tâche planifiée

Une variable de tâche dénommée $task est créée en utilisant New-ScheduledTaskAction. On précise que la tâche utilisera PowerShell en le notant dans -Execute et on précise le script qui sera lancé grâce à son chemin.



Figure - Création de la répétitivité de la tâche

$taskInterval consiste à décider de la fréquence à laquelle le script sera exécuté. Il s’exécutera donc tous les jours à 16h00. La suite du script donne un nom à la tâche et le stocke dans une variable du nom de $taskName.

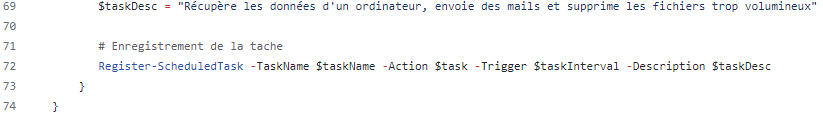


Figure - Description de la tâche et enregistrement de cette tâche

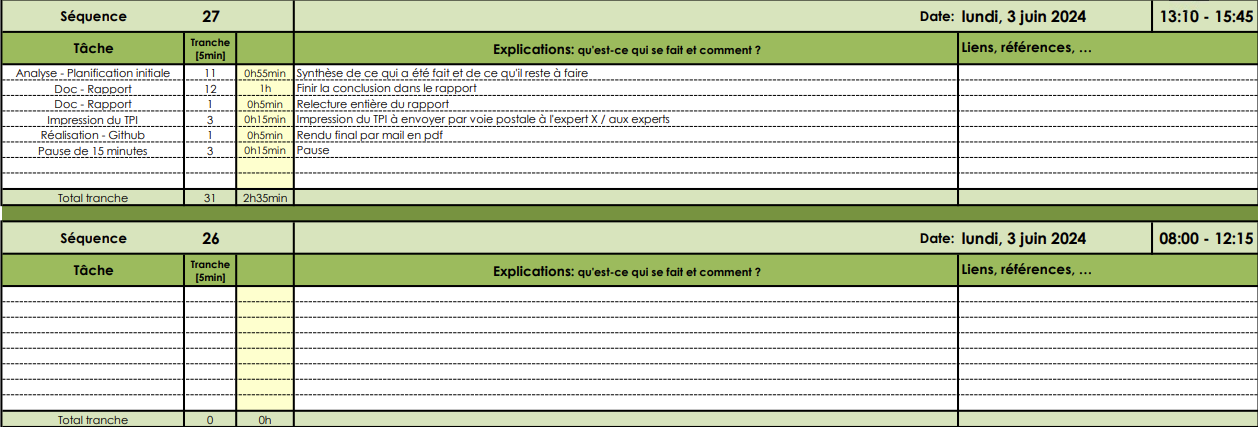
Le script donne une description à la tâche, puis l’enregistre définitivement ensuite. Les 2 accolades permettent de refermer invoke-command ainsi que le foreach.

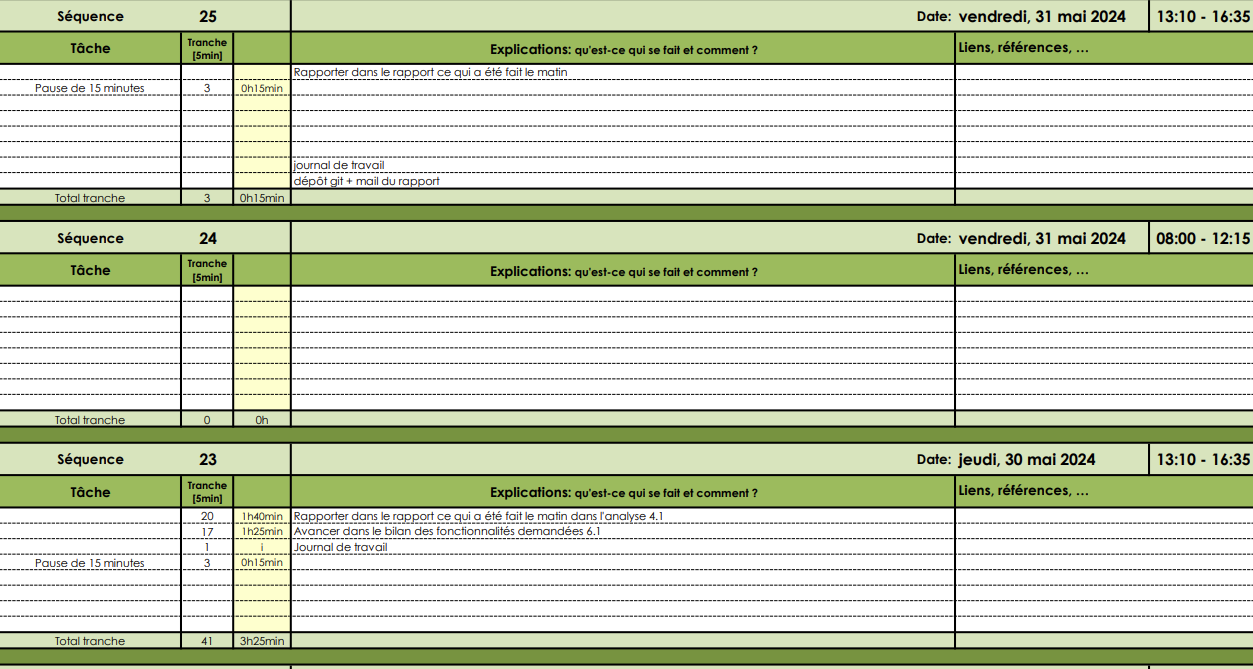
## Conception des tests

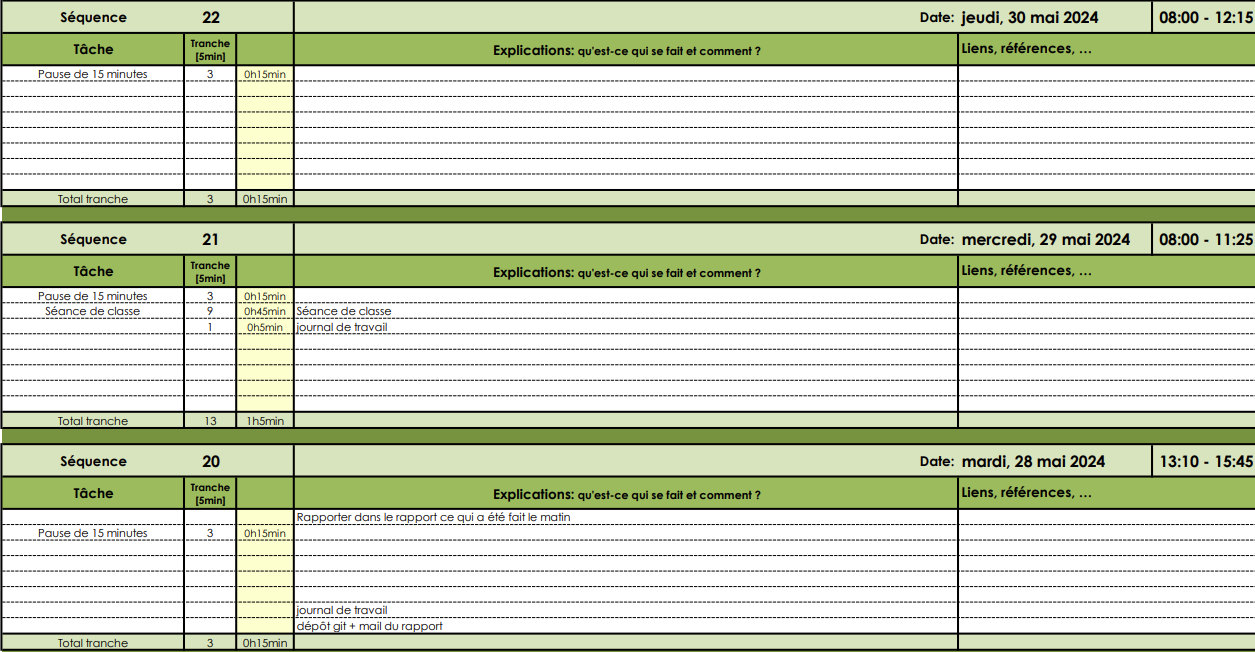
La stratégie de test employée est triviale ; Chaque ligne est testée indépendamment des autres pour avoir une suite d’instructions qui fonctionnent. Une fois ces instructions mises ensembles, elles sont retestées par groupe jusqu’à qu’elles fonctionnent elles-aussi. Cette opération sera effectuée tant que le script ne fonctionne pas entièrement.

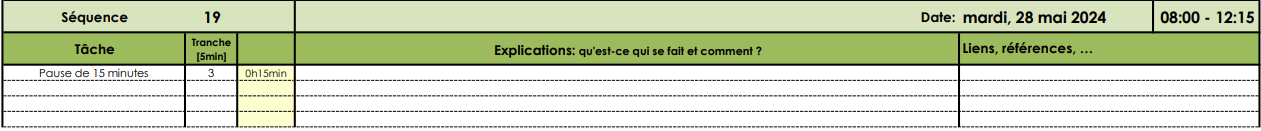
## Planification détaillée

La planification se lit de bas en haut, car c’est comme cela qu’apparait la planification dans le journal de travail

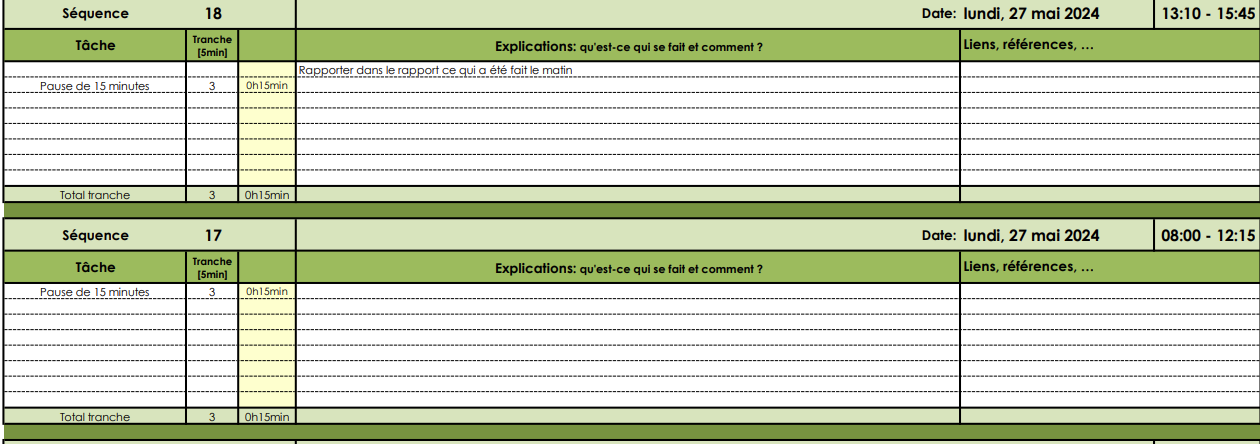


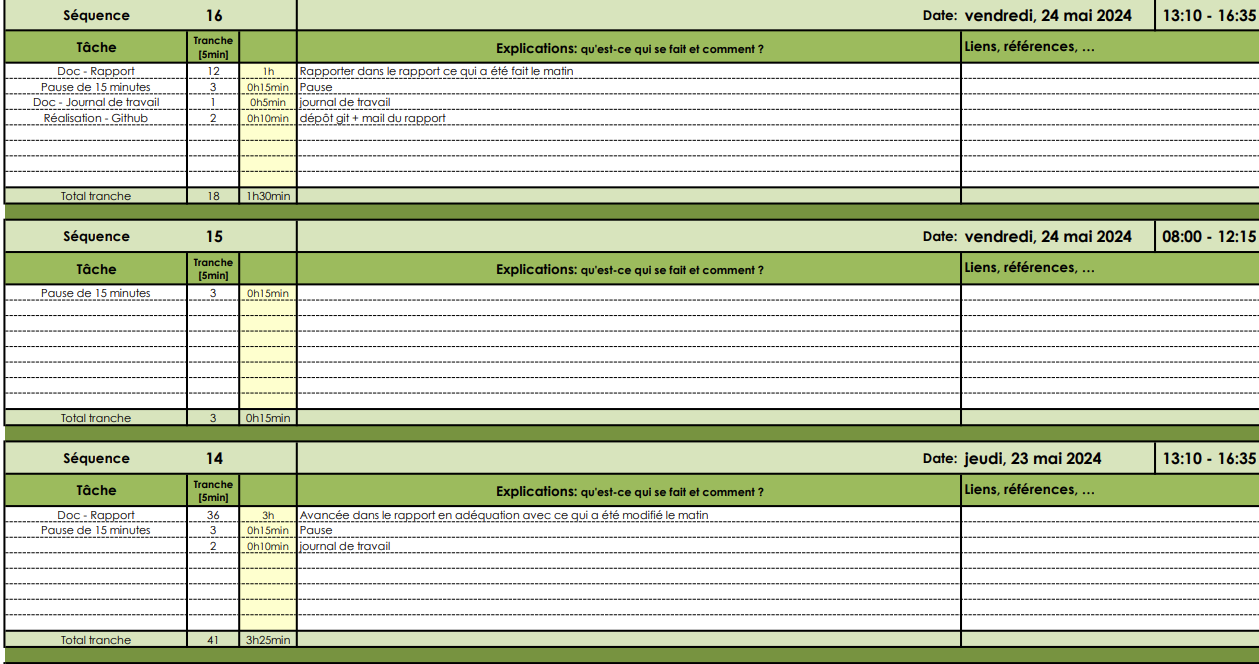


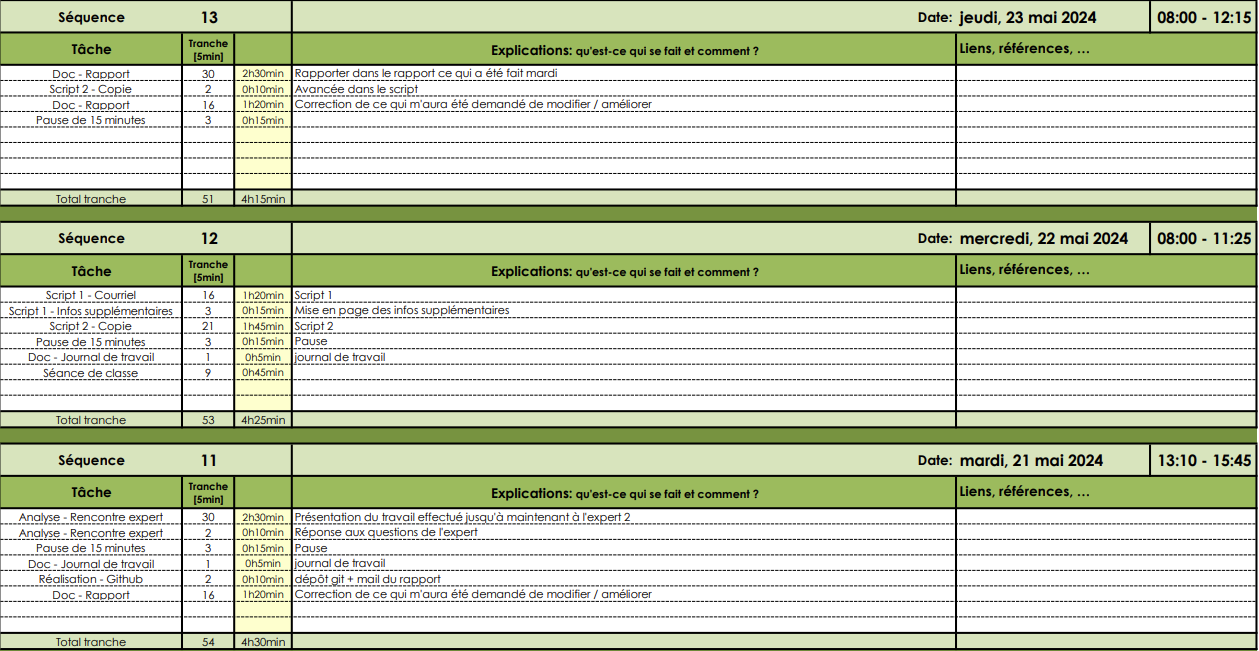


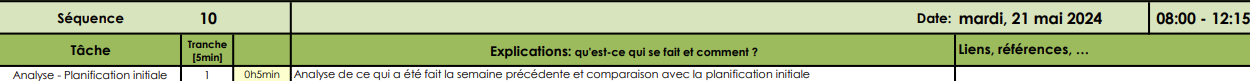


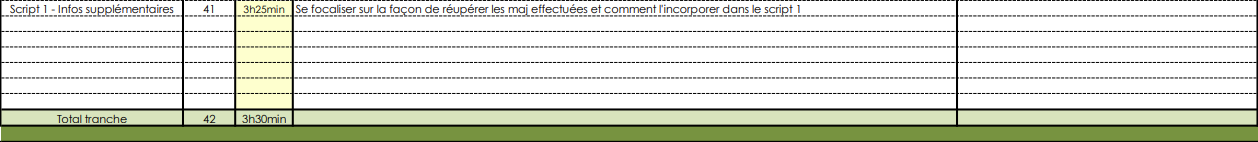


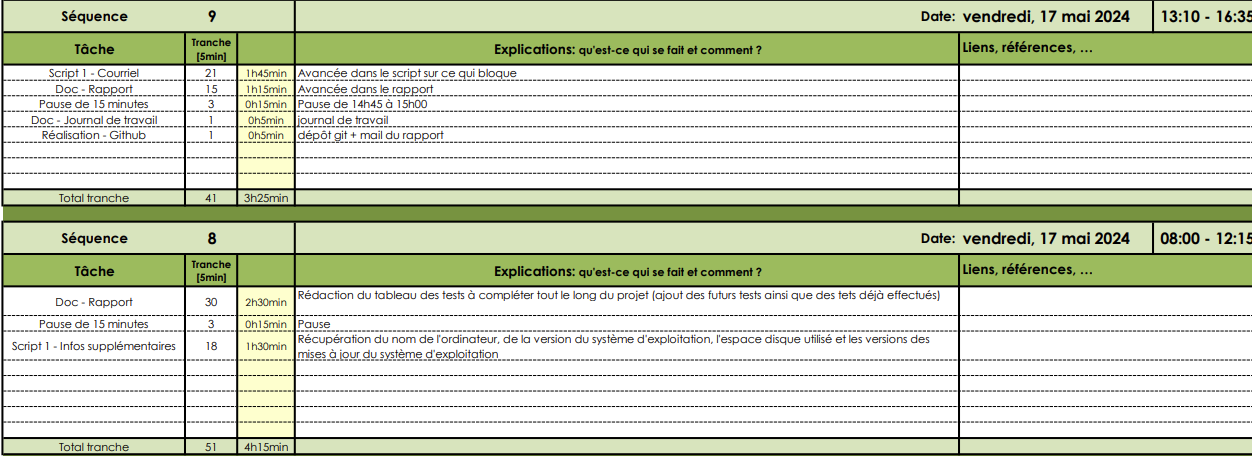


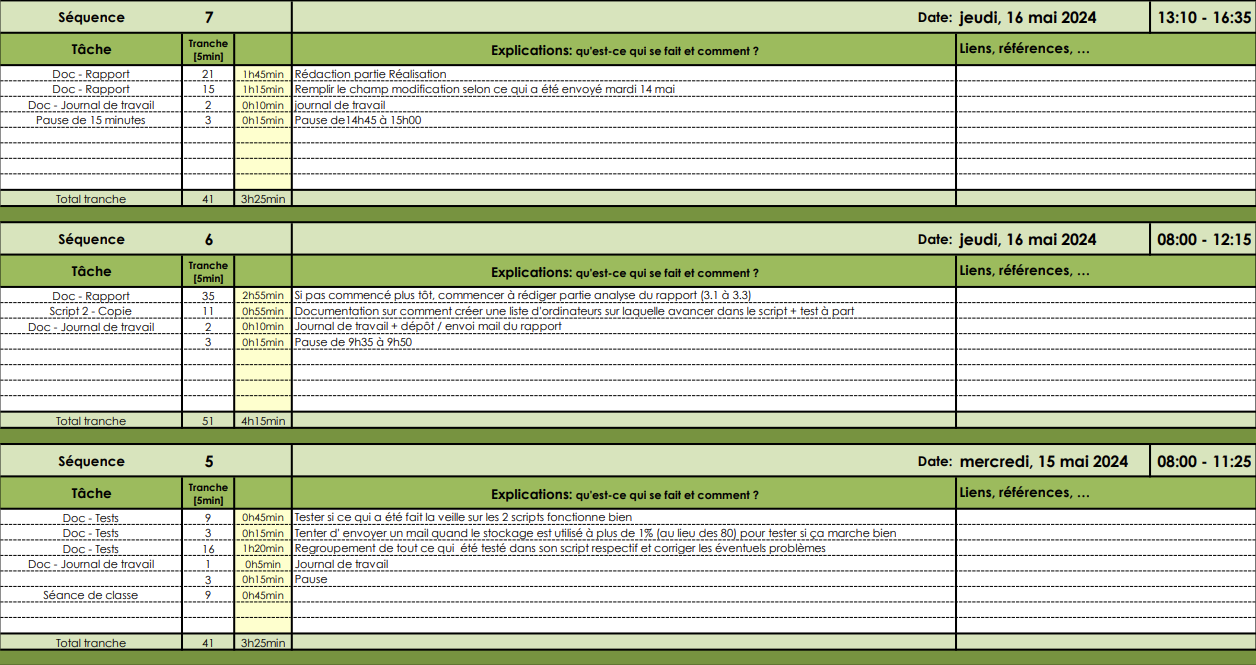




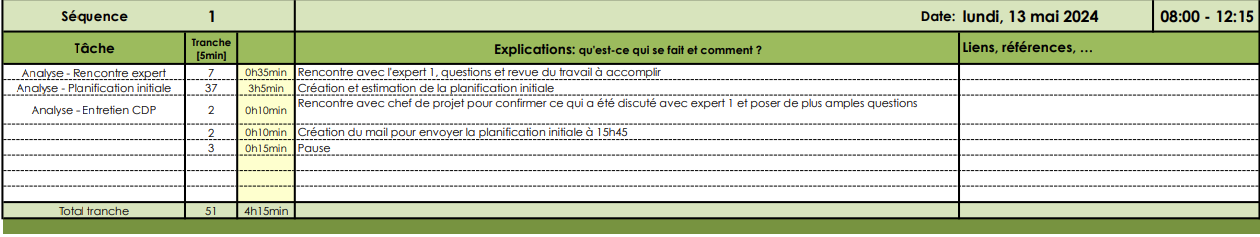












# Réalisation

## Dossier de Réalisation

### Version des outils logiciels utilisés

Machine locale

|  |  |
| --- | --- |
| **Logiciel** | **Version** |
| PowerShell Desktop | 5.1.19041.4291 |

Yann2k22srv

|  |  |
| --- | --- |
| **Logiciel** | **Version** |
| PowerShell Desktop | 5.1.20348.558 |

Yann2k22srv2

|  |  |
| --- | --- |
| **Logiciel** | **Version** |
| PowerShell Desktop | 5.1.20348.558 |

### Comptes utilisés

|  |  |  |
| --- | --- | --- |
| **Utilisation** | **Compte** | **Mot de passe** |
| Envoi du mail automatisé | pappro2mail@gmail.com | .Etml-123 |
| Réception du mail automatisé | tpiymetml@gmail.com | .Etml-123 |
| Compte serveur YANN2K22SRV | Administrateur | .Etml-123 |
| Compte serveur YANN2K22SRV2 | Administrateur | .Etml-123 |

### Configuration spéciale des serveurs

Tout ce qui n’est pas précisé est un paramètre par défaut. Les 3 machines ont la police d’exécution déterminée sur « Unrestricted »

Les 2 machines distantes sont configurées de la même manière, la seule différence est que yann2k22srv2 a une autre adresse ip (192.168.10.53). Toutes les captures d’écran proviennent de yann2k22srv. Les deux machines ont leur pare-feu désactivé.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Spécifications de Windows Server

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure - Configuration du pare-feu

Les pare-feux sont désactivés car ils bloquent l’utilisation de la commande

$remotename = [System.Net.Dns]::GetHostByAddress($remoteip).Hostname

Cela empêche donc le script 2 de résoudre le DNS et de récupérer le nom de la machine qui est utilisé par la suite pour créer une session dans laquelle il sera possible de coller le script 1. En bloquant le pare-feu, on s’affranchit des problèmes du genre. Malgré tout, cela cause d’énormes problèmes de sécurité. Le plus optimal serait de créer une règle qui autoriserait la commande de s’exécuter, et dans le même temps il faudrait désactiver toutes les règles qui coupent le script.

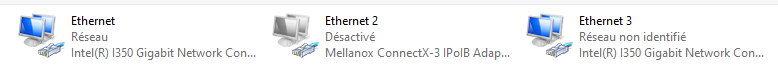


Figure - Configuration des cartes réseau

La partie réseau est décidée par le **réseau** de l’école et la partie hôte a été décidée arbitrairement. Comme c’est un masque /24, seulement la dernière partie est une partie hôte. Voici un tableau regroupant les adresses IP utilisées et à quelles machines elles correspondent :

|  |  |
| --- | --- |
| yann2k22srv | **192.168.10**.51 |
| INF-A13-M215 | **192.168.10**.52 |
| yann2k22srv2 | **192.168.10**.53 |

La carte réseau activée a 2 connexions : une pour que la machine soit connectée à internet et l’autre pour avoir un réseau LAN avec la machine physique.

Une image contenant texte, logiciel, Police, affichage

Description générée automatiquement

Figure - Protocole IPV6 désactivé

Le protocole IPV6 est désactivé afin d’éviter les problèmes d’incompatibilité qui pourraient survenir en utilisant winRM.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Activation du bureau à distance

Il a fallu activer le bureau à distance sur les serveurs physiques afin de pouvoir s’y connecter plus tard.

Une image contenant texte, Police, Bleu électrique, capture d’écran

Description générée automatiquement

Figure - Configuration de WinRM

Pour configurer celui-ci, il suffit d’ajouter les autres machines comme des hôtes de confiance (TrustedHosts avec PowerShell. Si le pare-feu n’est pas désactivé, cela ne fonctionnera pas. La valeur est une étoile pour autoriser n’importe quelle machine.

La prochaine étape est de créer un mot de passe d’application pour utiliser un compte Google Pour créer un mot de passe d’application, il faut accéder à l’A2F de Google dans les réglages du compte, puis sélectionner « mot de passes d’application ». Il suffit ensuite d’entrer un nom et d’appuyer sur « créer »

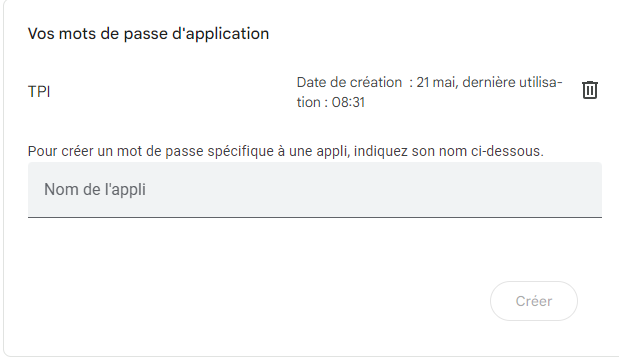


Figure - Page d'accueil de la page de création de mots de passe d'application Google

Google générera un mot de passe composé de 19 caractères en comptant les espaces comme celui ci-dessous.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Mot de passe généré

C’est un mot de passe comme celui-ci qu’il faut entrer dans le script pour que celui-ci puisse se connecter à Google.

### Déploiement et exécution détaillée

Maintenant que l’environnement est apte à accueillir et exécuter tout ce que les scripts sont susceptibles de lui demander, il faut effectuer la procédure suivante :

1. Avoir les 2 scripts sur la machine dans le même répertoire de la machine locale.
2. Lancer ensuite le script nommé « x-yanmangiagl-setup.ps1 » avec en argument les adresses IP des machines sur lesquelles vous voulez que le script 1 s’installe.
3. Par défaut, le script 2 crée un dossier nommé Scripts se trouvant dans la partition C. (C:\Scripts). Pour le changer il faut aller changer sa valeur dans le script directement.
4. Si aucune adresse n’est renseignée, le script 2 copiera le script monitoring dans C:/Scripts sur la machine qui exécute le script 2.
5. Le script 1 va se copier dans le dossier créé et créer une tâche planifiée qui fera fonctionner le script tous les jours à 16h00.

### En tête commune aux 2 scripts



Figure - En-tête du script 2

L’en-tête est utilisé pour fournir des explications sur le script à la personne qui l’utilise. Le .Notes communique des informations sur l’auteur, les modifications du script ainsi que les raisons. .SYNOPSIS donne ce que fait le script en gros, tandis que .DESCRIPTION décrit comment le script effectue le synopsis. .EXAMPLE donne un exemple, .PARAMETER les paramètre s’ils existent et .LINK un lien important pour le script. L’en-tête se trouve dans les 2 scripts, avec quelques différences. En effet, contrairement au script 1, le script 2 contient des paramètres. Cela veut dire que le .Parameter n’est pas nécessaire dans l’en-tête du script 1.

### Réalisation du script de setup

Comme dit plus haut dans la partie analyse, le script 2 a été réalisé avant le premier, ce qui a causé un problème au moment de la réalisation du script 1 mais ce sera expliqué plus tard dans ce rapport. Le script a d’abord été fait avec le minimum requis. C’est-à-dire la copie d’un item dans une session, la création d’une tâche et des chemins en dur. J’ai eu des problèmes en réalisant la partie de la copie, car une erreur mentionnant un chemin vide était affichée. J’ai d’abord essayé de créer une liste d’arguments dans le invoke-cammand, mais ça ne fonctionnait pas non plus. Il s’est avéré que le problème venait du fait que la commande qui posait problème devait être en dehors du bloc d’invoke-cammand, et que je devais utiliser -ToSession pour faire la copie du fichier plutôt qu’essayer de me connecter à la session depuis cette-même session, vu que c’est comme ça qu’Invoke-Command fonctionne.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Le message d'erreur qui apparaissait avant de modifier copy-item

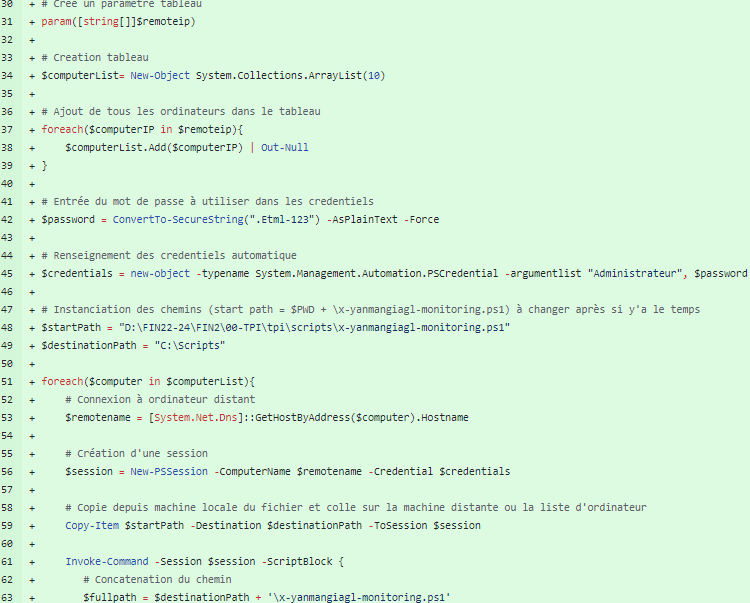


Figure - Première partie du code source original

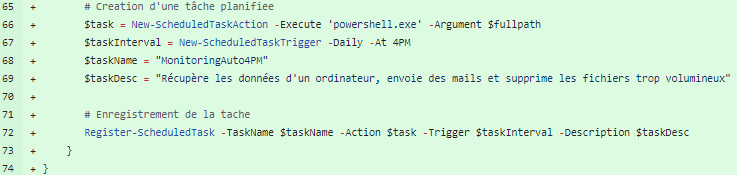


Figure - Seconde partie du code source original

On peut voir dans cette première version plusieurs problèmes ; les chemins en dur ne respectent pas le cahier des charges et il n’y a aucune gestion d’erreur. Les exemples aussi sont faux, c’est la première chose qui est modifiée.

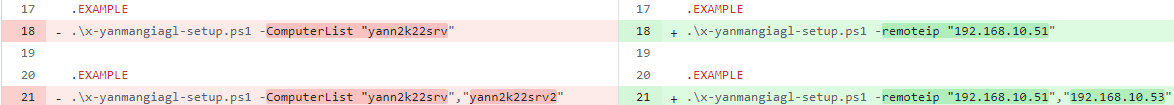


Figure - Modification des exemples

Cependant, un problème inattendu s’est déclaré qui a forcé une nouvelle version du script. En effet, le script 1 était bien ajouté en argument de la tâche planifiée, mais elle n’avait pas le bon format. Celui-ci devait non seulement contenir le chemin entier, mais en plus il devait contenir un argument nommé -File pour que le script s’exécute bien correctement.



Figure - Modification de l'argument du script 2

J’ai su que je devais faire ça en créant une tâche planifiée manuellement sur yann2k22srv, et dans les options du déclencheur de tâche, voilà ce qui était montré :

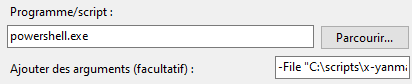


Figure - Résultat voulu de $fullpath

Une fois cette erreur corrigée, j’en ai profité pour mettre en place la gestion d’erreurs et les limites du script 2.

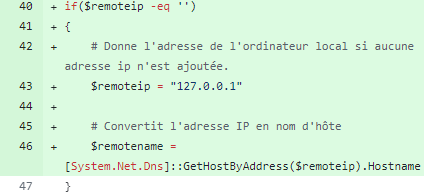


Figure - Ajout de gestion d'erreurs pour la conversion DNS

Si aucune adresse IP n’est spécifiée, le diagnostic est fait sur la machine locale. Attention cependant, cela ne fonctionnera pas sur la machine locale de l’école INF-A13-M215 car elle reste gérée par l’école qui bloque la requête du script.

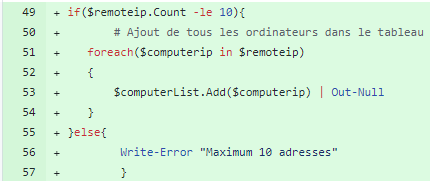


Figure - Ajout d'une limite de 10 machines

Comme il est demandé dans le cahier des charges que le nombre d’adresses maximum soit de 10 machines, le script limite le tableau d’adresses IP à 10 éléments maximum.

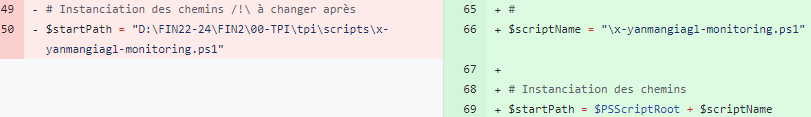


Figure - Modification d'un chemin en dur en chemin automatique

Ici, le chemin a été changé pour être automatique avec la variable $PSScriptRoot qui récupère le chemin duquel le script est exécuté, ce qui permet de se débarrasser d’un chemin en dur. De plus, ce chemin en dur causait des problèmes pour créer la tâche planifiée comme il faut. Un test a été fait pour s’assurer qu’on pouvait concaténer les valeurs ce qu’on ne peut pas faire avec $PWD par exemple.

Une image contenant texte, capture d’écran, Police, Bleu électrique

Description générée automatiquement

Figure - La concaténation ne se fait pas

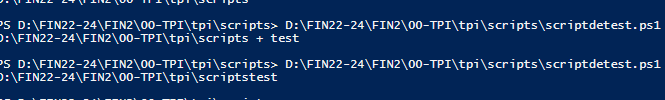


Figure - La concaténation se fait



Figure - Ajout d'une gestion d'erreurs pendant la connexion

Une gestion d’erreur est effectuée avec un try catch pour sécuriser tout le code, même si cela ralentira un peu l’exécution du code.

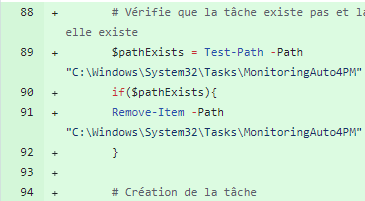


Figure - Vérification de si la tâche existe déjà, et la supprime si c'est le cas

Avant de créer la tâche, j’ai incorporé un test permettant de checker si la tâche existe déjà en me basant sur le chemin par défaut de Windows.

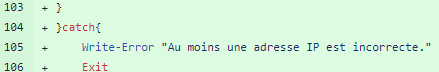


Figure – Bloc de code si une adresse IP est incorrecte

Si le script a une erreur, par exemple une adresse IP non valide, et bien un message d’erreur a été rajouté. Je pourrais améliorer ça en faisant en sorte que le script mette de côté cette adresse IP et fasse quand même les adresses IP valides plutôt que juste quitter le programme. Le script est donc fonctionnel comme démontré ci-dessous :

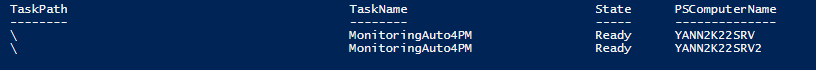


Figure - Sortie du script une fois les tâches créées

### Réalisation du script de maintenance

Le script de maintenance est le script qui se charge de vérifier le stockage et de récupérer les variables qui seront envoyées par mail si le stockage atteint 80%. Il y’a eu pas mal de problèmes au début pour faire fonctionner l’envoi de mail qui m’a bloqué pendant quasiment une journée complète. En effet, j’avais réalisé en amont du projet un script qui envoyait un mail sécurisé avec MimeKit et MailKit en utilisant Nuget, qui est un gestionnaire de package pour .NET [[1]](#footnote-1). Nuget a donc été installé avec MailKit et MimeKit.



Figure - Installation de MimeKit et MailKit

Cependant, une erreur proclamait qu’il manquait des dépendances et le script s’arrêtait. Il a donc été décidé que Send-MailMessage serait utilisé, mais ce n’était pas plus facile pour autant.

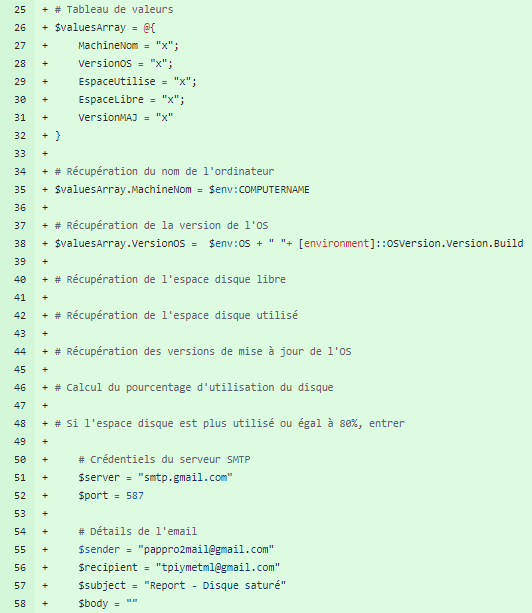


Figure - Première partie du script 1

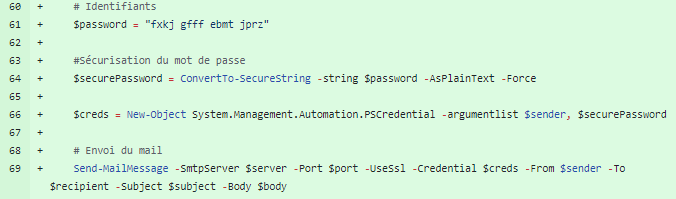


Figure - Seconde partie du script 1

Dans cette première version du script, seules les variables d’environnement et les valeurs nécessaires à l’envoi du mail sont réalisées, car c’est ce que j’ai eu le temps de faire à cause du blocage créé à cause de ma première idée avec NuGet, puis les problèmes qui ont suivi avec Send-MailMessage.



Figure - Capture d'écran du message d’erreur dû à Google

Ce message d’erreur a été un grand ralentissement dans ma progression, car j’avais renseigné le bon port, le bon mot de passe, le bon nom d’utilisateur, et spécifié le SSL, désactivé le pare-feu du serveur et la machine était connectée à Internet. Ce qui faisait bloquer le mail était qu’il fallait en faire un secure-string. J’avais pensé que l’instruction créait un objet identitaire le faisait nativement, mais non. Google ne considérait la connexion sans sécurité, même si elle était en SSL. J’ai donc dû continuer le script le jour suivant.



Figure - Enlèvement de la lettre du disque

J’ai commencé par enlever la lettre du disque car il n’y en a pas besoin dans l’immédiat, puis j’ai ajouté les valeurs qui manquaient pour l’espace libre et utilisé.

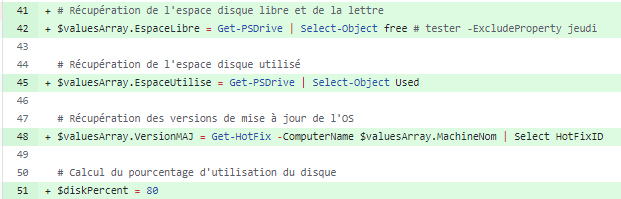


Figure - Ajout des valeurs dans le script

Je ne l’ai pas implémenté correctement, il fallait une variable ou préciser, par exemple $valuesArray.EspaceUtilise.Used, mais ça faisait répétitif et ce n’était pas très lisible. J’ai corrigé cela plus tard.



Figure - Création de la condition de 80%

Un début de code pour la suppression des fichiers volumineux est créé. C’est dans ce bloc que sera rajouté plus tard toutes les commandes de suppression.



Figure - Création du corps du mail

Le corps du mail est rempli pour ne plus être vide, et on y met déjà les valeurs qu’on veut recevoir dans le mail.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Mails de tests reçus

Dans ce mail cependant, on peut remarquer un problème d’encodage. Pour contourner le problème, j’ai décidé de reformuler mon texte pour ne pas avoir à me soucier de ce problème handicapant, comme dans cet exemple.

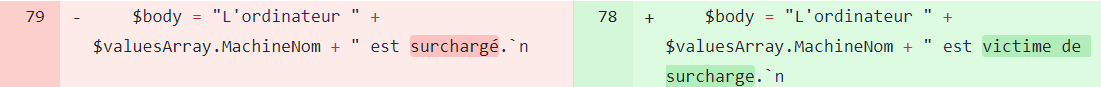


Figure - Contournement des problèmes d'encodage

Les autres changements sont plus importants ; L’ajout des versions de mise à jour et la suppression des fichiers OVA de plus de 3 Gb [[2]](#footnote-2).



Figure - Ajout des versions de MAJ



Figure - Corps du mail final

L’espace libre et l’utilisation de l’espace sont bien notés d’une manière qui fonctionne, et plus lisible qu’avant.



Figure - Commande permettant de supprimer les fichiers ova de plus de 3Go

La commande cherche dans tous les dossiers que contient C : ainsi que C : lui-même tous les fichiers dont l’extension est .ova, et qui pèsent plus de 3Gb, puis les supprime. Le -Force est là pour effacer les .ova qui pourraient être cachés. Pour améliorer le script il serait possible de mettre une valeur plutôt que ‘.ova’, pourquoi pas dans un foreach pour rajouter d’autres extensions, comme les .avi par exemple.

## Modifications

Aucune modification n’a été demandée par le chef de projet ou les experts. Il y’a eu un conseil de mettre en forme les mises à jour en tant que liste dans le mail plutôt que la forme « à la suite » qui est donnée afin de rendre le tout plus lisible. Après avoir essayé de l’implémenter sans succès, il a été décidé de revenir dessus plus tard s’il restait du temps.

# Tests

## Dossier des tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro | Description | Procédure | Résultat attendu | Résultat obtenu |
| 1 | Vérification synchronisation GitHub – GitHub desktop | Faire un commit GitHub en utilisant GitHub Desktop afin de voir si le commit passe bien sur GitHub | OK | OK |
| 2 | Lancement d’un script depuis son chemin à l’intérieur d’un autre script | Entrer la commande ./x-pappromail.ps1 pour voir si le script s’exécute | KO | OK |
| 3 | Affichage de l’aide (script 2) | Dans l’en-tête des scripts, renseigner .SYNOPSIS, .DESCRIPTION, .EXAMPLE et .PARAMETER, puis faire les commandes  Get-Help x-yanmangiagl-setup.ps1 et  Get-Help yanmangiagl-setup.ps1 -Examples | L’aide s’affiche sans problème | L’aide s’affiche sans problème |
| 4 | Déplacement des variables (script 2) | Faire bouger les variables $startPath et $destinationPath dans le scriptblock d’invoke command pour ne plus avoir de x dite $null alors qu’elle ne l’est pas | Les variables sont instanciées et permettent au script de passer à la suite | Les 2 variables ont l’air de fonctionner maintenant mais $session est $null |
| 5 | Supprimer d’ invokecommand copy-item | Bouger Copy-Item du bloc d’invoke-command pour l’isoler | OK | OK |
| 6 | Tenter 192.168.10.53 sans l’autre comme paramètre du script 2 (.51 fonctionne) | Restart-Service winrm, suivi d’un redémarrage du serveur, puis effectuer la commande | Marche pour toujours | Fonctionne seulement une fois après un redémarrage du serveur si la commande est faite assez vite |
| 8 | Désactiver pare-feu serveur 2 | La premier serveur a un pare-feu éteint et pas le deuxième, tester de le désactiver pour voir si ça fonctionne | Marche pour toujours | Marche pour toujours |
| 9 | Envoi de mail avec Send-MailMessage | Lancer la commande permettant d’envoyer un mail | La commande peut s’exécuter sans problème | La commande ne s’exécute pas et retourne une erreur |
| 10 | Encoder avec Powershell | Rajouter [System.Text.Encoding ::UTF8] sur la variable $body | Affichage de éà dans le mail | Affichage de ?? dans le mail |
| 11 | Test du script 1 fini | Mise en commentaire de la commande mettant le vrai pourcentage et lancement du script 1 depuis la machine locale | Le script fonctionne comme prévu et le mail est envoyé | Le script fonctionne comme prévu et le mail est envoyé |
| 12 | Exécution du script depuis la tâche planifiée | Même test mais sur yann2k22srv | Le script fonctionne comme prévu et le mail est envoyé | Le mail n’arrive pas > Problème vient sûrement de la création de la tâche dans script 2 |
| 13 | Création d’une tâche manuelle | Création d’une tâche qui exécute le script manuellement pour comprendre d’où vient le problème en comparant les 2 tâches après | Dans les arguments on voit c:\scripts… | Dans les arguments on voit -File c:\scripts… |
| 14 | Calcul du stockage correct | Appel des valeurs $usedSpace.Sum et $totalspace pour calculer en % le stockage utilisé | Le pourcentage apparait normalement | Le pourcentage apparait normalement |
| 15 | Connecter serveur à internet | Réactiver une des cartes réseau | OK | OK |
| 16 | Lancer un mail depuis les 2 machines en même temps | Lancer le script 2 avec les 2 machines en paramètres avec le script de debug pour voir comment se lance les mails | Chaque machine fait son propre mail | Chaque machine fait son propre mail |
| 17 | Test taille réelle | OVA de 3 Go, OVA de 2 Go, deux serveurs prêts à accueillir les scripts | OVA de 3go suppr, ova de 2 go gardé, mail envoyé | -Path est null pour le truc que j’ai réglé dans les 5 min, tout le reste a fonctionné |

# Conclusion

## Bilan des fonctionnalités demandées

Pour ce qui est des fonctionnalités demandées dans le cahier des charges, un point sur toutes les fonctionnalités va être effectué.

**Exécution sans erreur sur une machine quelconque avec indication de la bonne exécution des tâches :** Fonctionnel à 100%

**Envoi d’un courriel avec les informations de la machine :** Fonctionnel à 100%, le mail s’envoie parfaitement sans délai et ne se retrouve pas dans les spam de la boite de réception.

**Récupération des données pertinentes** **:** Fonctionnel à 100%, l’interface pourrait être plus conviviale, mais toutes les informations demandées sont consultables dans le mail. Une amélioration souhaitable serait de convertir en Gb les octets utilisés pour rendre le mail plus lisible.

**Suppression des dossiers trop volumineux :** Fonctionnel à 100%, les fichiers OVA de plus de 3GB sont supprimés.

**Exécution de la tâche planifiée avec succès :** Fonctionnel à 100%, la tâche planifiée est créée et s’exécute tous les jours à 16h00 sans accroc.

**Copie sur une machine distante :** Fonctionnel à 100%, le script 2 copie le script 1 correctement sans erreur.

**Procédure de déploiement et exécution détaillée dans le rapport :** 90 % Je considère avoir dit ce qu’il y’avait à dire dans le rapport, mais j’ai l’impression d’oublier quelque chose à ce sujet.

## Bilan de la planification

La planification n’a pas vraiment été suivie. J’ai été trop optimiste sur le temps que prendrait les tâches à accomplir et suis parti du principe que je n’aurai pas d’erreur en créant les scripts, ce qui était peu probable. En conséquence, beaucoup de tâches ont été réalisées en plus de temps que prévu, et des tâches ont même dues être rajoutées. Les tâches qui ont causé du retard sur ma planification sont toutes les tâches importantes, comme la création de scripts ou encore la réalisation du rapport. La tâche où j’ai pris de l’avance était la rencontre avec l’expert 2, où j’avais pensé que je devrais lui montrer tout le projet dans le détail et que ça prendrait beaucoup plus de temps que la réalité. Avec une semaine de plus, je pense que j’aurais pu réussir à magnifier le script en rajoutant les chemins en paramètre du script 2 ainsi que des paramètres pour l’heure voulue pour la tâche planifiée. Pour ce qui est du temps réalisé, j’ai travaillé pendant les séances de classe des mercredis ainsi que chez moi 20 minutes le samedi 1er juin, ce qui expliquerai pourquoi j’arrive à plus de 90h sur les 88 demandées. Le journal de travail se trouve dans les divers du rapport.

## Bilan personnel

Si le projet devait être refait, j’accorderai plus de temps au code, et plus d’importance aux chemins. J’ai peut-être passé trop de temps sur le rapport. Il y’a aussi beaucoup de captures d’écran que j’ai faites qui ne sont pas utilisées alors qu’elles pourraient et devraient être utiles. De plus, la dernière semaine j’ai eu beaucoup de problèmes à rester concentré sur mon travail pour finir le rapport. En revanche, ce que je voudrais garder telle quelle c’est ma capacité à résoudre les problèmes de script plutôt rapidement, et avoir su poser des questions à mon chef de projet et aux experts quand j’en avais besoin. Ce projet m’a appris à mieux gérer mon temps de travail, de nouvelles méthodes pour envoyer un mail via PowerShell, debugger efficacement sur PowerShell et oser poser des questions, ce qui n’était pas forcément toujours le cas pendant les projets précédents. Pour améliorer le projet, il y’a plusieurs voies : d’abord il faudrait améliorer les chemins des scripts, bien que ceux-ci marchent sur les machines testées (toutes les machines testées avaient leur partition principale sur la lettre C) Une autre chose à améliorer serait d’avoir une meilleure attitude de travail. En effet, à plusieurs reprises il m’est arrivé de lâcher prise et de faire d’autres choses pendant mon travail, ou alors encore de regarder dans le vide sans avancer. Cela arrive souvent après que je sois bloqué sur un élément, comme un script qui ne fonctionne pas comme attendu. Là où je le ressens le plus est pendant la rédaction du rapport, où je pouvais mettre 20 minutes pour écrire une phrase simple. Les difficultés principales que j’ai rencontrées, c’était que pour certains problèmes, il était compliqué de savoir si le problème venait du script ou de l’environnement dans lequel il évoluait.

### Remerciements

Mr Gaël Sonney, mon chef de projet qui m’a aidé à de nombreuses reprises.

Mr Daniel Berney, mon second expert venu me render visite mardi 21 mai pour s’assurer que le projet se déroulait correctement.

Mr Gabriel Maret, mon premier expert qui s’est assuré que le cahier des charges soit bien compris et qui a répondu à mes questions.

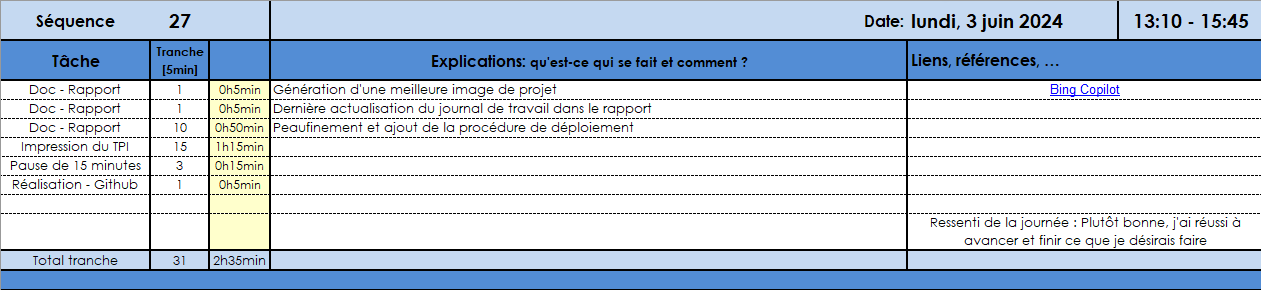
Mr Leonar Dupuis, un camarade de classe qui m’a aidé à arranger mon journal de travail et qui a relu mon travail.

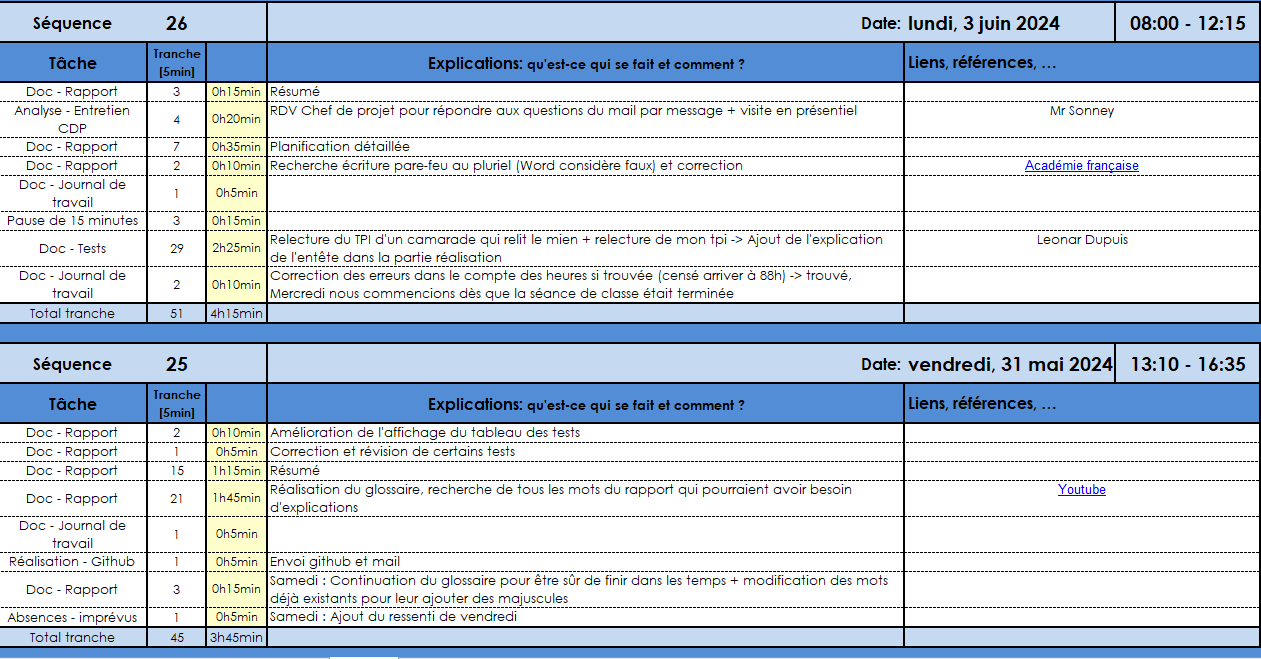
Mr Cédric Schaffter, qui m’a aidé à me préparer au TPI pendant les projets d’approfondissement.

# Divers

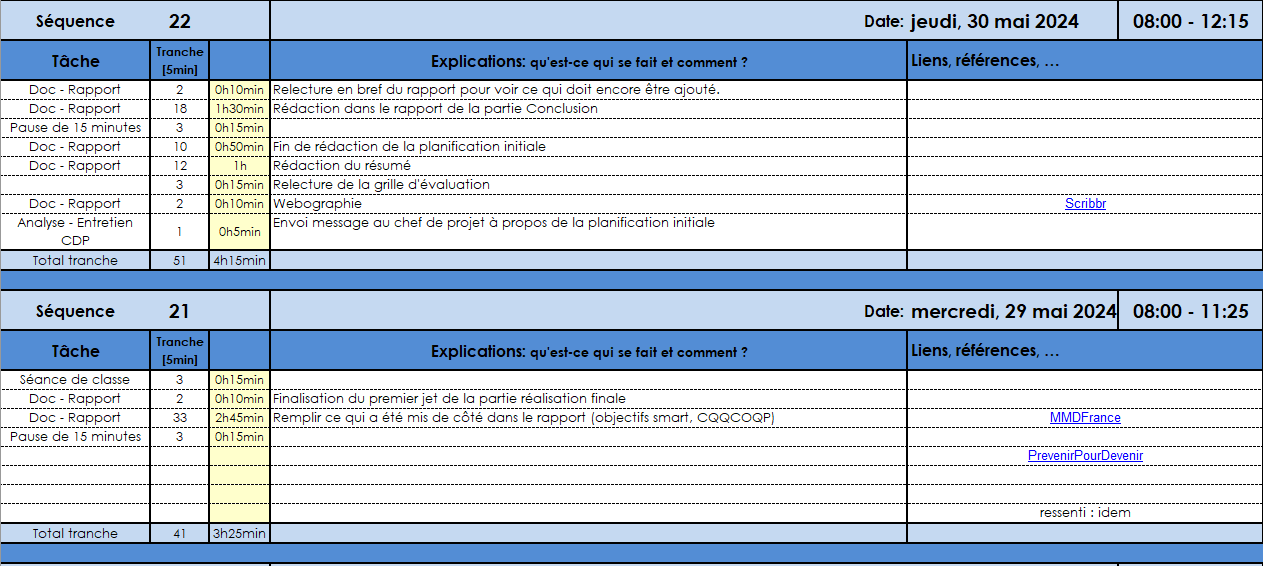
## Journal de travail

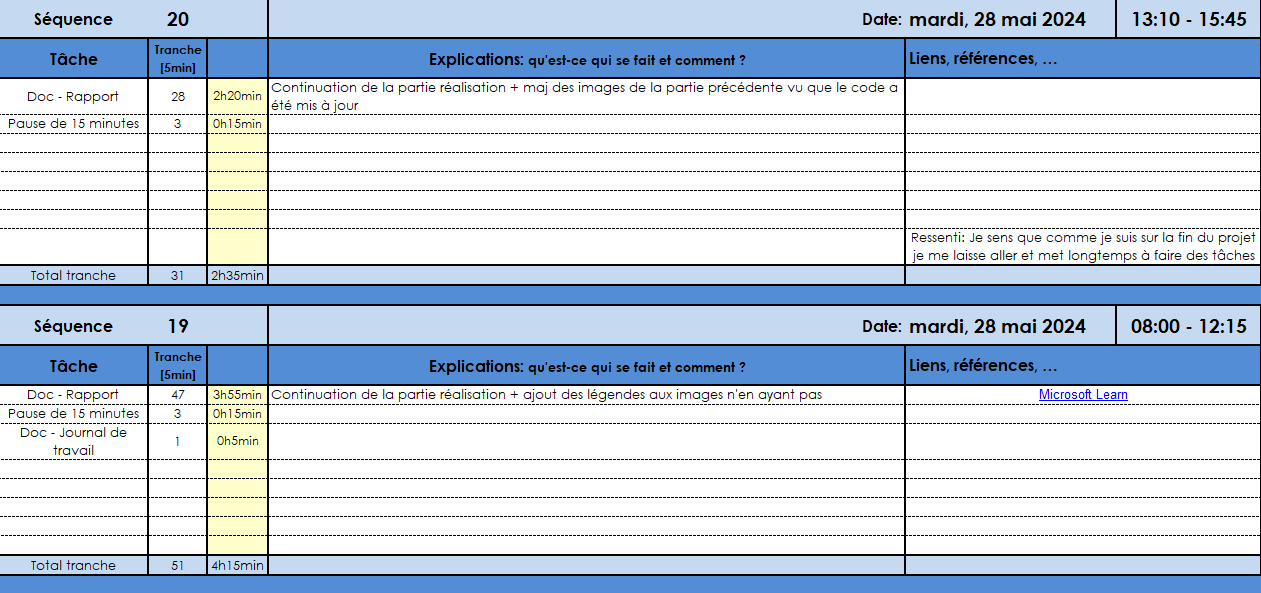
Le journal de travail a été réalisé en parallèle du projet pendant tout son déroulement. Dès que je sentais un petit blocage dans mon travail, je profitais pour regarder l’heure et noter ce qui avait été fait depuis le dernier petit blocage, ce qui rend les tranches très fidèles à la réalité.

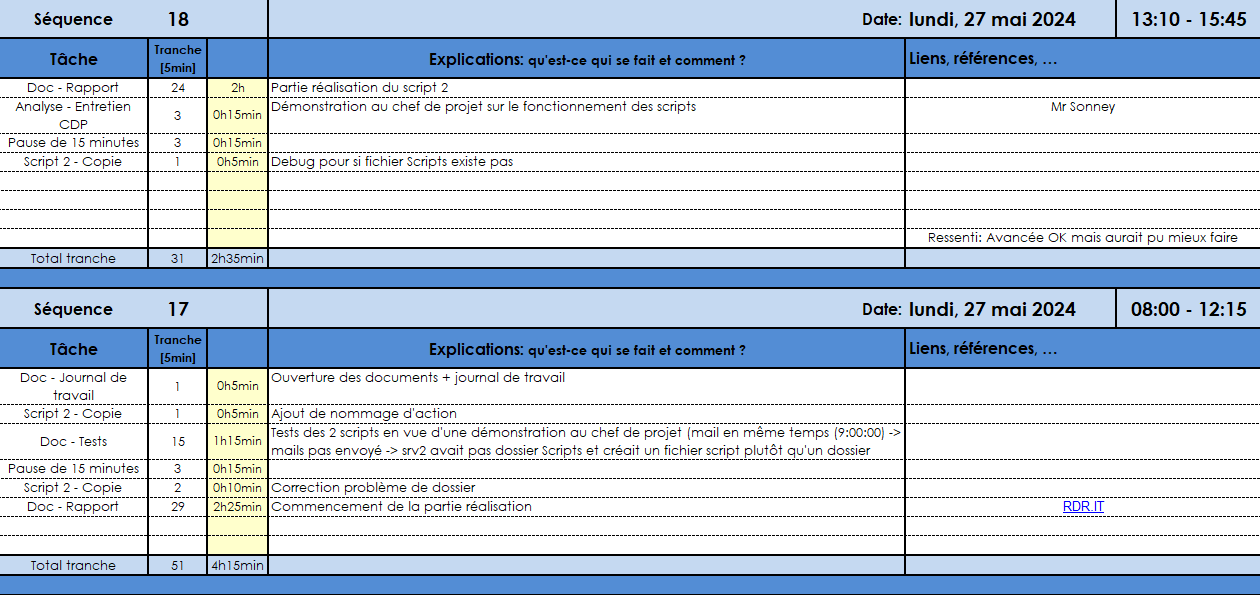


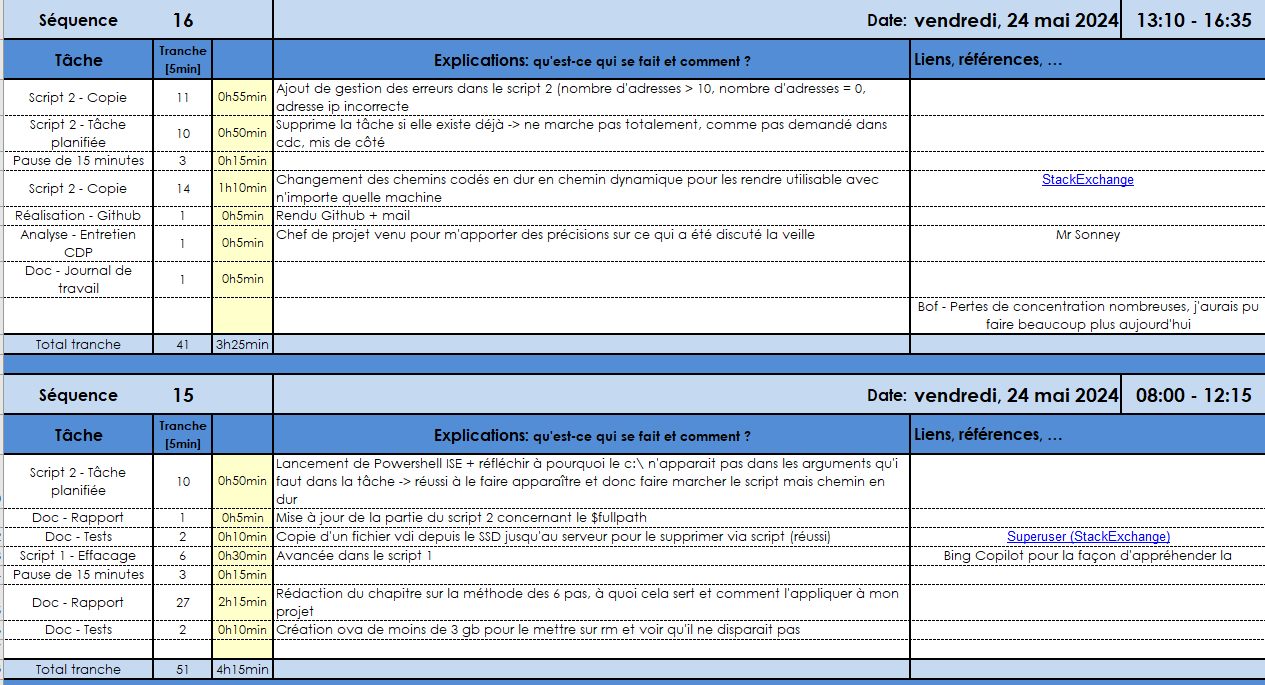


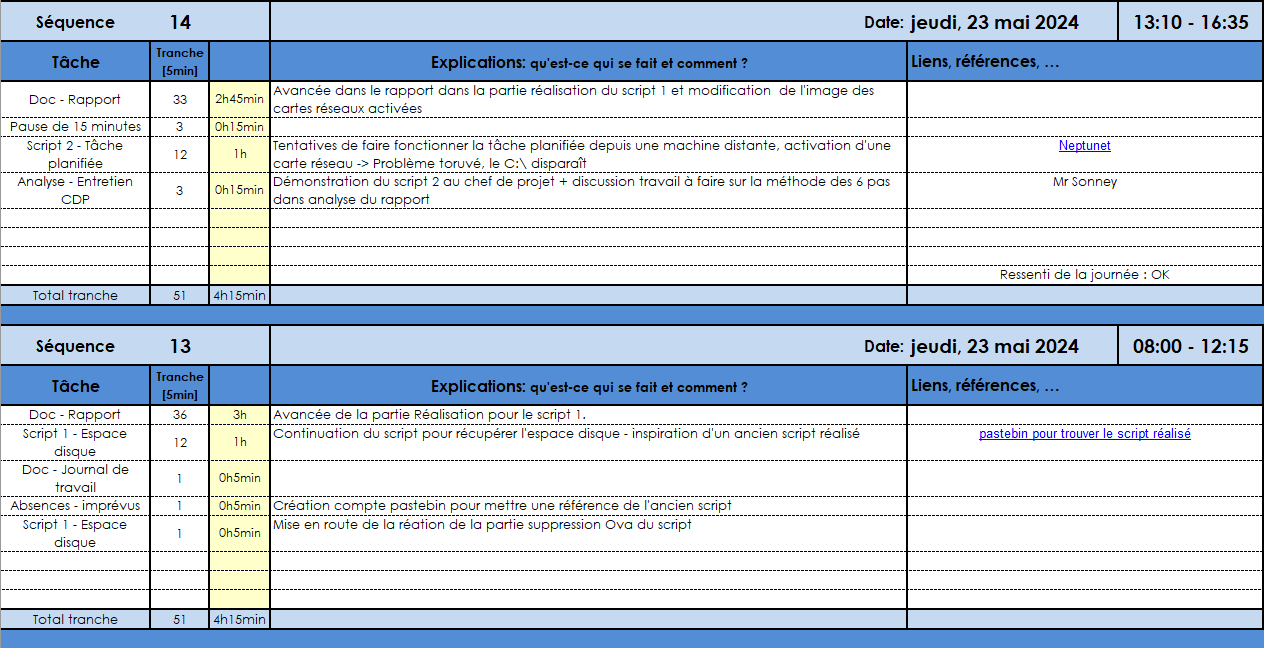


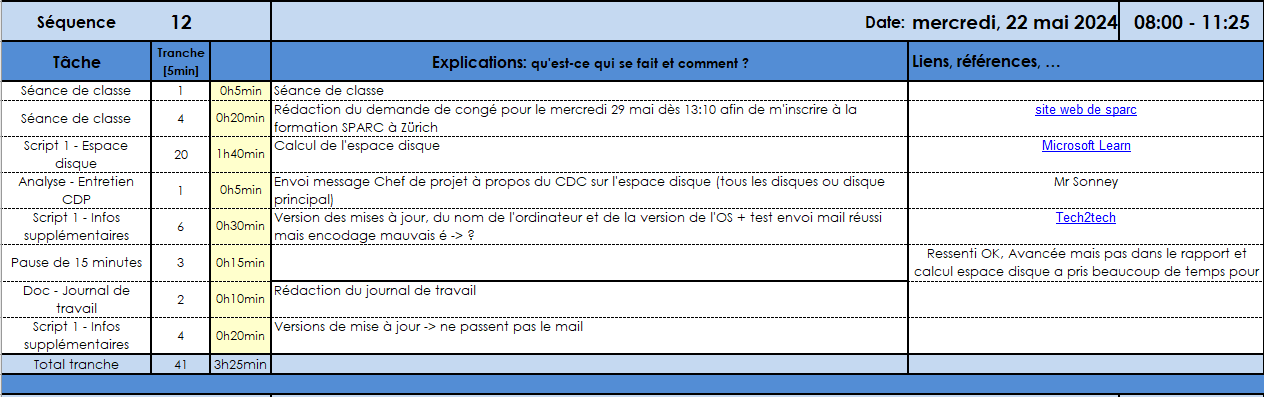


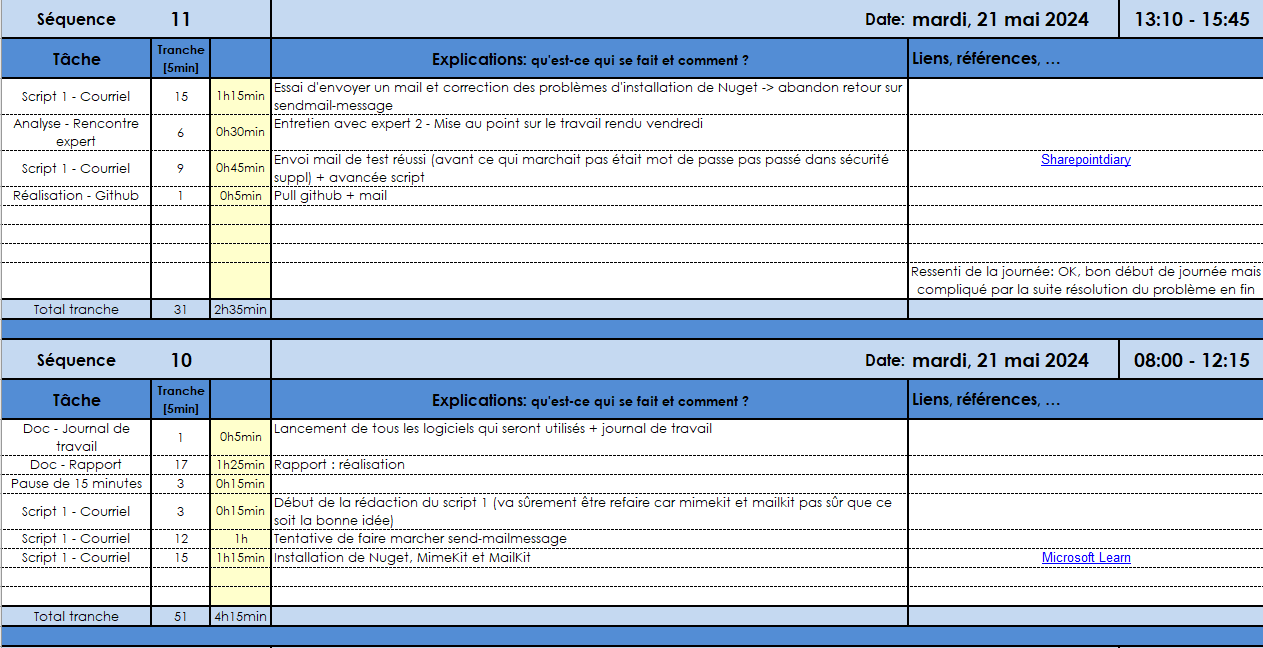


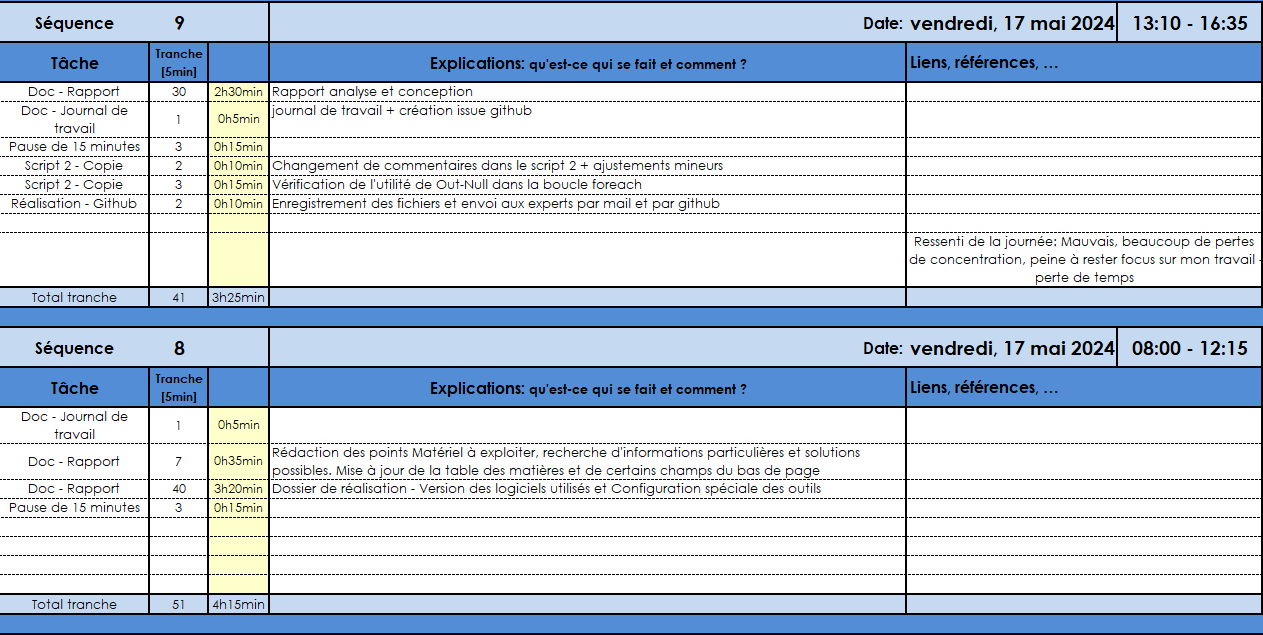


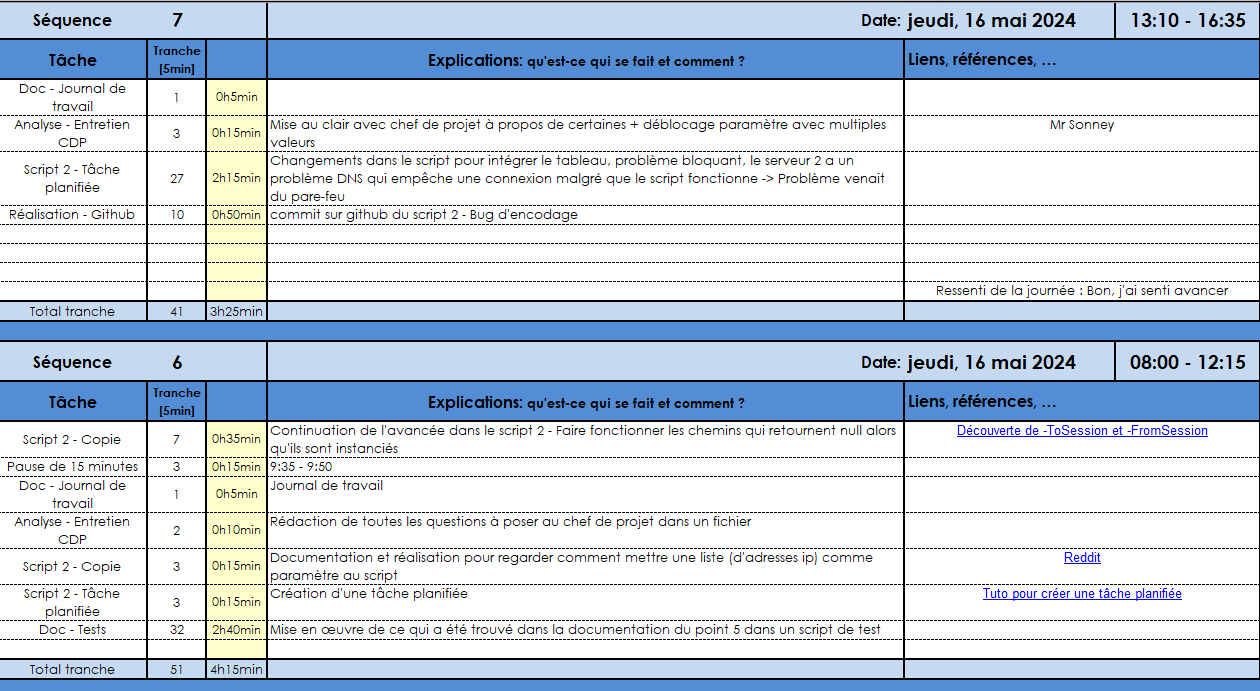


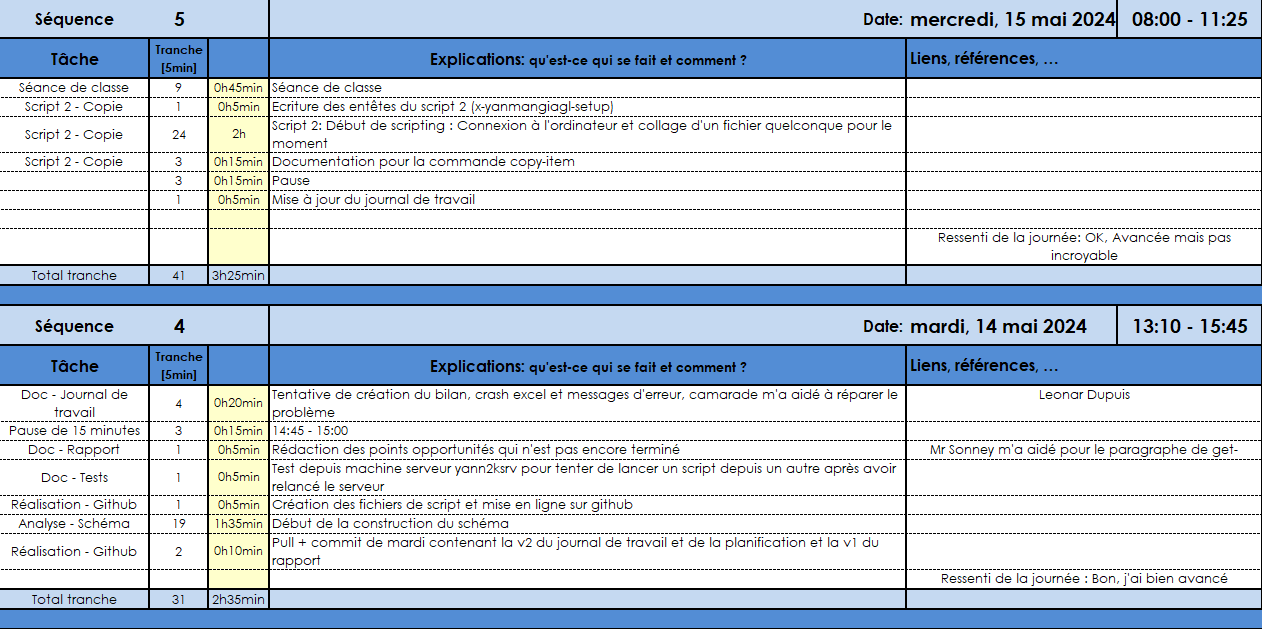


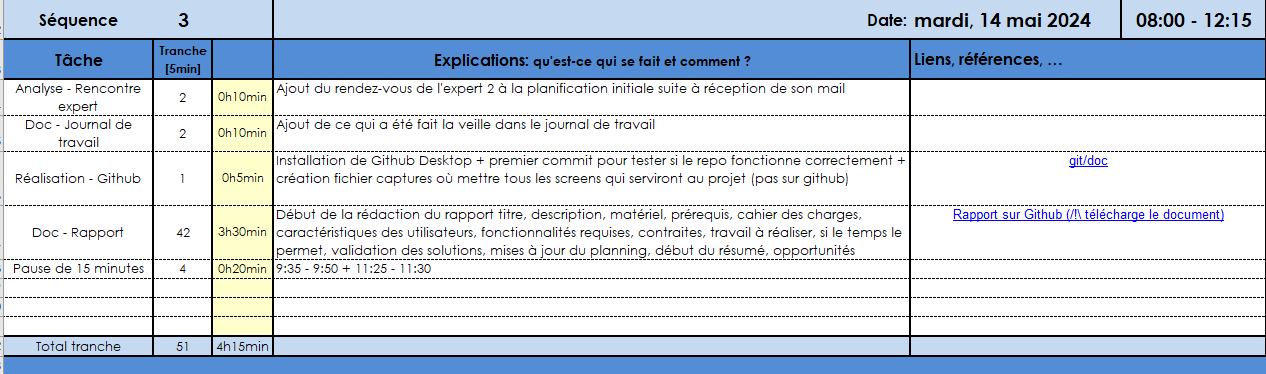


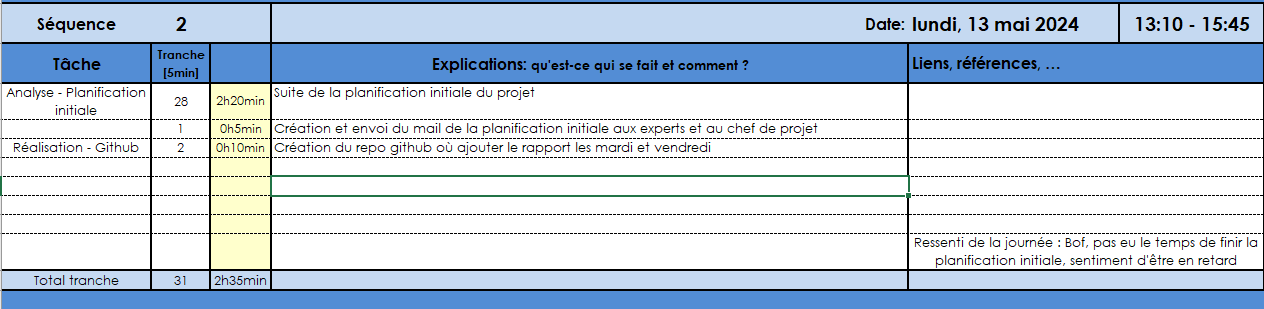


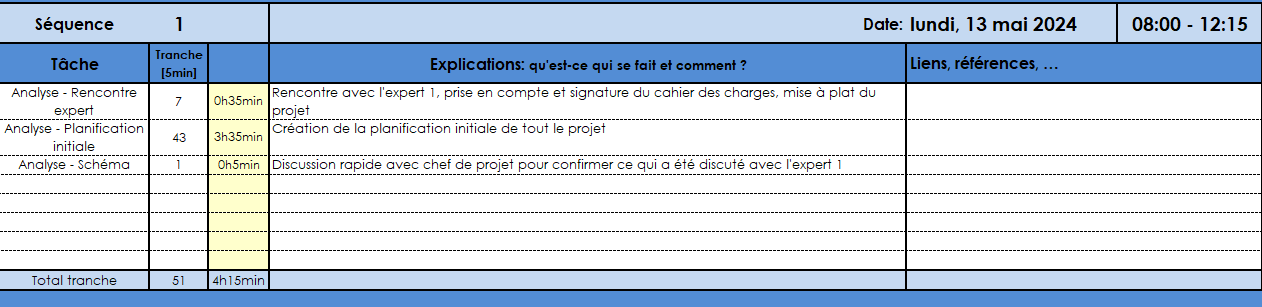












## Webographie

La Webographie a été réalisée avec un outil dénommé Scribbr, permettant de générer la webographie de manière académique en y collant le lien.

Comment créer une tâche planifiée en PowerShell. Malekal. https://www.malekal.com/comment-creer-une-tache-planifiee-en-powershell/. Published 4 août 2022.

Russr. (n.d.). Multi-thread Run CMD on multiple PC’s at the same time? : r/PowerShell. https://www.reddit.com/r/PowerShell/comments/hdrxfb/comment/fvmu4l0/?utm\_source=share&utm\_medium=web3x&utm\_name=web3xcss&utm\_term=1&utm\_content=share\_button

Tips For IT Pros. (2020, May 5). Copy file to and from Powershell remote session [Video]. YouTube. https://www.youtube.com/watch?v=P1zDkxSpzaQ

JonDouglas. (2024, May 21). Install NuGet client tools. Microsoft Learn. https://learn.microsoft.com/en-us/nuget/install-nuget-client-tools?tabs=windows

Rajack, S. (2024, May 22). How to Send Emails in PowerShell using Send-MailMessage? SharePoint Diary. https://www.sharepointdiary.com/2020/08/send-email-powershell.html

Guillerm, M. (2016, December 1). Récupérer la liste des updates Windows installées via PowerShell. Tech2Tech | News, Astuces, Tutos, Vidéos Autour De L’informatique. https://www.tech2tech.fr/recuperer-la-liste-des-updates-windows-installees-via-powershell/

Sdwheeler. (n.d.). Measure-Object (Microsoft.PowerShell.Utility) - PowerShell. Microsoft Learn. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/measure-object?view=powershell-7.4

SPARC. (n.d.). https://www.sparc-cyberdefence.ch/fr/

Poséidon. (2022, January 27). [Tips] Exécuter un script Powershell depuis le planificateur de tâches de Windows – NEPTUNET.FR. https://neptunet.fr/task-powershell/

Powershell to delete all files with a certain file extension. (n.d.). Super User. https://superuser.com/questions/1091344/powershell-to-delete-all-files-with-a-certain-file-extension

Concatenate filename with current directory path. (n.d.). Stack Overflow. https://stackoverflow.com/questions/57610821/concatenate-filename-with-current-directory-path

Romain. (2023, September 12). Créer un dossier en PowerShell - RDR-IT. RDR-IT. https://rdr-it.io/creer-un-dossier-en-powershell/

JonDouglas. (n.d.). Documentation de NuGet. Microsoft Learn. https://learn.microsoft.com/fr-fr/nuget/

Prévenir pour Devenir. (n.d.). Fichier de planification d’objectifs SMART à télécharger. Prévenir Pour Devenir. https://prevenirpourdevenir.ca/fichier-de-planification-dobjectifs-smart-a-telecharger/

Masucci, F. (2023, February 13). La méthode CQQCOQP en Mind Mapping. MMD France. https://mmdfrance.fr/methode-cqqcoqp/

SJ Abgrall. (2017, April 3). Créer un Index ou un Glossaire sous Word [Video]. YouTube. https://www.youtube.com/watch?v=srlNOQmQ4Ig

## Table des illustrations

[Figure 1 - Image générée par Bing Copilot représentant le projet 1](#_Toc168316926)

[Figure 2 - Schéma démontrant comment les scripts ont été pensés 17](#_Toc168316927)

[Figure 3 - Création du tableau 17](#_Toc168316928)

[Figure 4 - Variable d'environnement du nom de l'ordinateur 18](#_Toc168316929)

[Figure 5 - Variables d'environnement de la version de l'OS 18](#_Toc168316930)

[Figure 6 - Enregistrement de l'espace libre 18](#_Toc168316931)

[Figure 7 - Enregistrement de l'espace utilisé 18](#_Toc168316932)

[Figure 8 - Enregistrement de l'historique des versions de mise à jour 18](#_Toc168316933)

[Figure 9 - Calcul du stockage utilisé 18](#_Toc168316934)

[Figure 10 - Contenu de $usedSpace 19](#_Toc168316935)

[Figure 11 - Calcul de l'espace libre entre tous les disques 19](#_Toc168316936)

[Figure 12 - Calcul du stockage total 19](#_Toc168316937)

[Figure 13 - Calcul en % de l'espace utilisé par rapport au stockage total 19](#_Toc168316938)

[Figure 14 - Condition pour faire fonctionner la suite du script 19](#_Toc168316939)

[Figure 15 - Informations du serveur SMTP 19](#_Toc168316940)

[Figure 16 - Informations de l'expéditeur 19](#_Toc168316941)

[Figure 17 - Sujet et corps de mail 20](#_Toc168316942)

[Figure 18 - Mot de passe d'application Google utilisé 20](#_Toc168316943)

[Figure 19 - Sécurisation du mot de passe 20](#_Toc168316944)

[Figure 20 - Création d'un objet de crédentité 20](#_Toc168316945)

[Figure 21 - Envoi du mail via la commande Send-MailMessage 20](#_Toc168316946)

[Figure 22 - Instanciation du tableau 21](#_Toc168316947)

[Figure 23 - Instanciation d'un deuxième tableau 21](#_Toc168316948)

[Figure 24 - placeholder - boucle foreach 21](#_Toc168316949)

[Figure 25 - Conversion du mot de passe 21](#_Toc168316950)

[Figure 26 - Récupération du nom d'utilisateur 21](#_Toc168316951)

[Figure 27 - Implantation des chemins 22](#_Toc168316952)

[Figure 28 - Début de la boucle 22](#_Toc168316953)

[Figure 29 - Informations d'une session ouverte 22](#_Toc168316954)

[Figure 30 - Argument servant à lancer la tâche planifiée 22](#_Toc168316955)

[Figure 31 - Création de la tâche planifiée 23](#_Toc168316956)

[Figure 32 - Création de la répétitivité de la tâche 23](#_Toc168316957)

[Figure 33 - Description de la tâche et enregistrement de cette tâche 23](#_Toc168316958)

[Figure 34 - Spécifications de Windows Server 28](#_Toc168316959)

[Figure 35 - Configuration du pare-feu 29](#_Toc168316960)

[Figure 36 - Configuration des cartes réseau 29](#_Toc168316961)

[Figure 37 - Protocole IPV6 désactivé 30](#_Toc168316962)

[Figure 38 - Activation du bureau à distance 30](#_Toc168316963)

[Figure 39 - Configuration de WinRM 30](#_Toc168316964)

[Figure 40 - Page d'accueil de la page de création de mots de passe d'application Google 31](#_Toc168316965)

[Figure 41 - Mot de passe généré 31](#_Toc168316966)

[Figure 42 - En-tête du script 2 33](#_Toc168316967)

[Figure 43 - Le message d'erreur qui apparaissait avant de modifier copy-item 34](#_Toc168316968)

[Figure 44 - Première partie du code source original 34](#_Toc168316969)

[Figure 45 - Seconde partie du code source original 34](#_Toc168316970)

[Figure 46 - Modification des exemples 35](#_Toc168316971)

[Figure 47 - Modification de l'argument du script 2 35](#_Toc168316972)

[Figure 48 - Résultat voulu de $fullpath 35](#_Toc168316973)

[Figure 49 - Ajout de gestion d'erreurs pour la conversion DNS 35](#_Toc168316974)

[Figure 50 - Ajout d'une limite de 10 machines 36](#_Toc168316975)

[Figure 51 - Modification d'un chemin en dur en chemin automatique 36](#_Toc168316976)

[Figure 52 - La concaténation ne se fait pas 36](#_Toc168316977)

[Figure 53 - La concaténation se fait 36](#_Toc168316978)

[Figure 54 - Ajout d'une gestion d'erreurs pendant la connexion 36](#_Toc168316979)

[Figure 55 - Vérification de si la tâche existe déjà, et la supprime si c'est le cas 37](#_Toc168316980)

[Figure 56 – Bloc de code si une adresse IP est incorrecte 37](#_Toc168316981)

[Figure 57 - Sortie du script une fois les tâches créées 37](#_Toc168316982)

[Figure 58 - Installation de MimeKit et MailKit 38](#_Toc168316983)

[Figure 59 - Première partie du script 1 39](#_Toc168316984)

[Figure 60 - Seconde partie du script 1 39](#_Toc168316985)

[Figure 61 - Capture d'écran du message d’erreur dû à Google 40](#_Toc168316986)

[Figure 62 - Enlèvement de la lettre du disque 40](#_Toc168316987)

[Figure 63 - Ajout des valeurs dans le script 40](#_Toc168316988)

[Figure 64 - Création de la condition de 80% 40](#_Toc168316989)

[Figure 65 - Création du corps du mail 40](#_Toc168316990)

[Figure 66 - Mails de tests reçus 41](#_Toc168316991)

[Figure 67 - Contournement des problèmes d'encodage 41](#_Toc168316992)

[Figure 68 - Ajout des versions de MAJ 41](#_Toc168316993)

[Figure 69 - Corps du mail final 41](#_Toc168316994)

[Figure 70 - Commande permettant de supprimer les fichiers ova de plus de 3Go 41](#_Toc168316995)

# Glossaire

A

Administrateur système

Personne responsable des serveurs d'une entreprise 6

C

CFC

Certificat fédéral de capacité 6

D

Débogage

Action de corriger des bugs informatique 18

Docker kubernetes

Outil de managment de docker 16

E

Environnement

Ensemble matériel et logiciels système 16

Ethernet

Protocole de communication 4

H

Hachage

Algorythme permettant de convertir en suite de nombres une chaîne de caractère 21

M

Machine virtuelle

Machine simulée sur de vraies machines 5

O

OVA

Extension de fichiers souvent volumineux utilisés pour les machines virtuelles 4

P

PowerShell

Solution multiplateforme d'automatisation de tâches 5

S

Script

Ensemble d'instructions dans un langage de programmation 4

Serveur SMTP

Ordinateur qui gère les courriels sortants 7

Session

Connexion d'un pc à un autre via PowerShell 22

SSL

Protocole de sécurité Internet 20

Système d’exploitation

Ensemble de programmes spécialisés qui travaillent ensemble (WIndows, Linux) 18

T

Tableau associatif

Tableau en programmation liant une clé à une valeur 17

# Annexes

Code source du script 1

Code source du script 2

Journal de travail

1. Citation venant de Microsoft Learn [↑](#footnote-ref-1)
2. Instruction de suppression générée par Bing Copilot et modifiée pour convenir au contexte [↑](#footnote-ref-2)