

Table des matières

[1. Analyse préliminaire 2](#_Toc184311171)

[1.1 Introduction 2](#_Toc184311172)

[1.2 Objectifs 2](#_Toc184311173)

[1.3 Gestion de projet 2](#_Toc184311174)

[1.4 Planification initiale 2](#_Toc184311175)

[1.5 Compte rendu 4](#_Toc184311176)

[2. Développement 4](#_Toc184311177)

[2.1 Configuration de l’environnement 4](#_Toc184311178)

[2.2 Journal des commits 5](#_Toc184311179)

[2.3 Développement et test 6](#_Toc184311180)

[7. Conclusion 8](#_Toc184311181)

[7.1 Résultats finaux et conclusion 8](#_Toc184311182)

[7.2 Utilisation de l’IA 8](#_Toc184311183)

# Analyse préliminaire

## Introduction

Ce rapport est la seconde partie du projet Install\_In\_Nutshell qui est effectuée dans le cadre du module ICT 324. Cette partie a pour but d’explique la partie « OPS » du projet, elle passera donc en revue la mise en place des tests unitaires du script ainsi que les processus de déploiement du script préalablement créé dans la partie « Dev ».

## Objectifs

Les objectifs de la partie « OPS » d’Install\_In\_Nutshell sont les suivants :

**Plateforme de téléchargement du script** :

1. Mise en place d’une plateforme web permettant le téléchargement du script « Install\_In\_Nutshell ».
2. Mise en place d’une ressource « Azure Web app Service » ayant pour but d’hébergé la plateforme de téléchargement.
3. Mise en place d’un « serveur de base de données flexible » sur Azure cloud.
4. Création d’un workflow GitHub Action permettant d’automatiser le déploiement de la plateforme de téléchargement.

**Framework de test unitaire :**

1. Création d’un script de test unitaire PowerShell nommé « unitTest.ps1 » permettant de tester certaines fonctionnalités du code.

**Workflow de test unitaire :**

1. Mise en place d’un workflow

## Gestion de projet

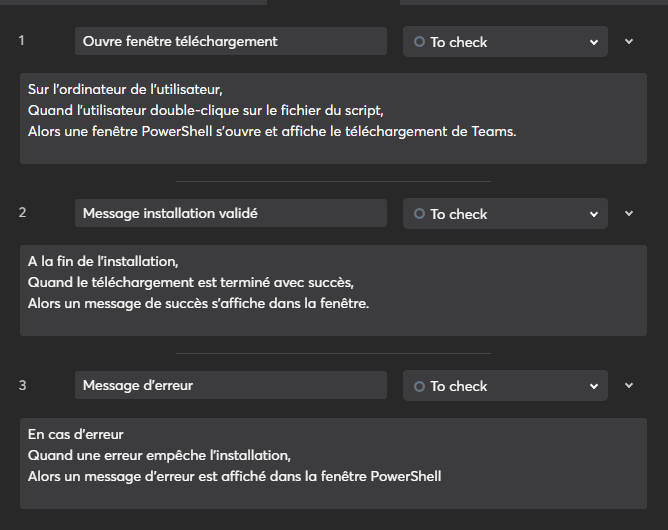
Ce projet est réalisé en binôme. Pour cela, nous utilisons l'outil IceScrum afin de créer des user stories et de définir des objectifs clairement ciblés. Nous utilisons également GitHub Desktop et des commits pour que chacun puisse accéder à une version toujours à jour du projet.  
Un journal personnel est également maintenu sous forme de fichier CSV.

Ce journal permet de suivre l'avancement du projet et de recenser les problèmes rencontrés. Ainsi, en cas d'absence de l'un des membres du binôme, il reste possible de suivre l'avancement du projet. L'objectif est qu'aucun événement ne vienne ralentir ou perturber le bon déroulement du projet.

## Planification initiale

Afin de bien débuter et de ne pas être perdus dans les étapes à venir, nous avons d’abord créé des user stories pour chaque fonctionnalité que nous souhaitions implémenter. Chacune de ces user stories a été accompagnée de tests d’acceptation qui nous permettront de bien comprendre les attentes du projet.

Voici donc les user stories et les technical stories qui ont été créées :

Comme mentionné précédemment, chaque user story a été créée avec des tests d’acceptance. Voici un exemple concret de tests d’acceptance qui ont pu être mis en place sur les user stories :

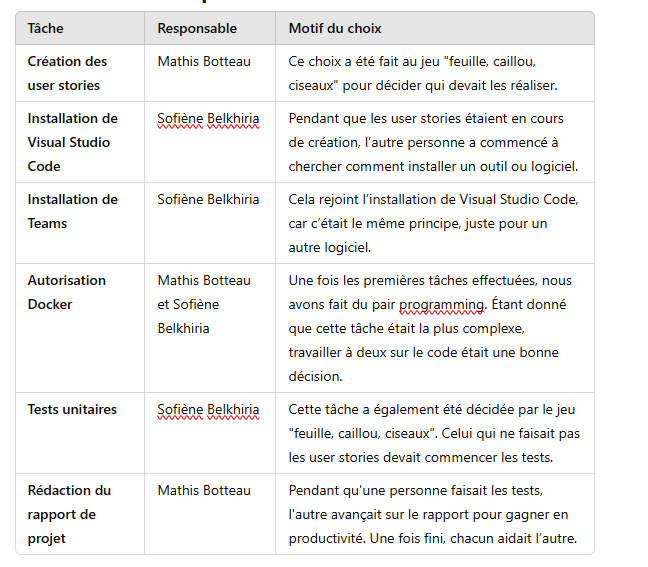
Une fois les user stories créées et acceptées, nous avons commencé à réfléchir à la manière dont nous allions concevoir le script et organiser tout le déroulement du projet : quelles technologies utiliser, comment structurer le script, quelles seraient les étapes à suivre, et à quel moment nous commencerions à rédiger le rapport de projet, etc.

Il a été décidé d’utiliser PowerShell pour créer le script, étant donné que nous avions déjà eu un module en deuxième année sur PowerShell et la création de scripts. Ce module couvrait des aspects tels que la récupération d’informations d’une machine distante ou locale, ou encore l’installation de logiciels sur une machine. C’est pour cela que nous avons choisi une technologie avec laquelle nous avions quelques bases.

Ensuite, nous avons décidé que les étapes d’installation des outils se feraient dans l’ordre que nous avions imaginé lors de la création du projet et de ses fonctionnalités. L’installation se déroule tout simplement comme cela :

1. Installation de Teams
2. Installation de Visual Studio Code
3. Vérification des permissions de l’utilisateur
4. Vérification de l’installation de Docker
5. Ajout de l’utilisateur au groupe Docker

## Compte rendu

Etant donne que ce projet à été fait par deux, nous avons dû nous répartir les taches et prendre des décisions. Voici qui a fait quoi et comment nous avons réparti les taches et pourquoi ces choix :

# Développement

## Configuration de l’environnement

L’environnement qui a été mis en place pour travailler sur ce projet est le suivant :

* Installation de extension PowerShell sur Visual Studio Code
* Installation de PowerShell version 7 depuis le Microsoft Store
* Installation de Github Desktop et clonage du répertoire du projet

## Journal des commits

Etant donne que nous avons utilisé github pour versionner notre projet, nous avons décidé d’utiliser une structure de commits comme celle-ci : « type(thème) : Description »

Voici notre journal de commits :

|  |  |  |
| --- | --- | --- |
| Auteur | Date | Message |
| Mathis Botteau | Thu Dec 5 16:13:03 2024 +0100 | Merge branch 'main' of https://github.com/termimi/Install\_in\_NutShell |
| termimi | Thu Dec 5 16:09:10 2024 +0100 | [doc] (journaux de travaux): Update journaux de travaux |
| termimi | Thu Dec 5 16:01:12 2024 +0100 | [Test] (unitTest.ps1): Ajout des tests unitaires pour l'ajout d'un utilisateur au groupe Docker |
| termimi | Thu Dec 5 15:02:06 2024 +0100 | [Feat] (install\_in\_nutshell): Ajout de paramètres pour préciser l'utilisateur et les logiciels à installer |
| Mathis Botteau | Thu Dec 5 14:33:03 2024 +0100 | Merge branch 'main' of https://github.com/termimi/Install\_in\_NutShell |
| termimi | Thu Dec 5 14:14:45 2024 +0100 | [Feat] (install\_in\_nutshell): Ajout d'un paramètre permettant de ne pas installer Teams et VSCode |
| termimi | Thu Dec 5 13:52:42 2024 +0100 | [fix] (install\_in\_nutshell): Modification des messages d'erreur pour inclure des erreurs PowerShell |
| termimi | Thu Dec 5 13:46:04 2024 +0100 | [Refactor] (install\_in\_nutshell): Ajout de messages de confirmation pour les installations terminées |
| termimi | Thu Dec 5 13:32:26 2024 +0100 | [doc] (journal de travail): Update journal de travail |
| Mathis Botteau | Thu Dec 5 13:32:25 2024 +0100 | [doc] (rapport de projet): Création du rapport de projet |
| termimi | Thu Dec 5 13:25:57 2024 +0100 | [feat]: Installation de Docker |
| termimi | Thu Nov 28 20:33:32 2024 +0100 | [doc] (journal de travail): Update du journal de travail |
| termimi | Thu Nov 28 20:31:33 2024 +0100 | [Feat] (install\_in\_nutshell): Ajout de l'installation de VSCode |
| termimi | Thu Nov 28 17:23:56 2024 +0100 | [doc] (journal de travail): Update journal de travail |
| termimi | Thu Nov 28 17:22:40 2024 +0100 | Merge branch 'main' of https://github.com/termimi/Install\_in\_NutShell |
| termimi | Thu Nov 28 17:22:37 2024 +0100 | [Feat]: Installation de Teams via le script |
| Mathis Botteau | Thu Nov 28 17:17:44 2024 +0100 | [doc] (journal de travail): Ajout des heures du journal de travail du 28/11/2024 |
| termimi | Thu Nov 28 16:16:21 2024 +0100 | [infra] (journal de travail): Update du journal de travail |
| termimi | Thu Nov 28 16:14:25 2024 +0100 | [Infra]: Mise en place des fichiers/dossiers de base du repo |
| termimi | Thu Nov 28 15:55:59 2024 +0100 | Initial commit |
|  |  |  |

## Développement et test

***Objectifs :***

Le projet a pour but d’automatiser et d’installer les logiciels suivants :

* Teams
* Visual Studio Code
* Docker Desktop

***Etapes du développement :***

1. **Mise en place des paramètres**

Le script contient de plusieurs paramètres, non obligatoire :

* + *CompleteInstall :* Indique si l’installation de teams et vs code sont nécessaires (valeur par défaut = true)
  + *UserToAdd* : Indique l’utilisateur à ajouté dans le groupe docker (a noté que si la valeur est vide, le script proposera à l’utilisateur d’indiquer l’user a ajouté)
  + *InstallPath :* Indique le répertoire d’installation du logiciel (si rien n’est indiqué, la valeur est l’emplacement du script)

2. **Vérification de la version de powershell**

Cette partie récupère la version actuelle de PowerShell, afin de vérifier si le script est capable de s’exécuter correctement, dans le cas contraire un message d’erreur est envoyé à l’utilisateur.

1. **Installation Teams & Visual Studio Code**

Cela est la partie principale du script, elle va en premier lieu récupérer l’installer de Teams et VS Code depuis internet, et ensuite les exécuter.

1. **Installation / Authorization de docker**

Cette étape est divisée en plusieurs sections que voici :

* + Vérification des permissions de l’utilisateurs afin de vérifier ses permissions, dans le cas contraire un message est envoyé un l’utilisateur et le script est stoppé.
  + Vérification de l’existence du groupe docker-users.

Une fois ces étapes effectuées, deux étapes se propose à nous :

* + Groupe docker existant : Cela ajoute l’utilisateur choisi à l’intérieur.
  + Groupe docker inexistant : Cela installe l’installer de docker et lance l’installation.

1. **Test unitaires**

Cette partie est constituée d’un script appart, cela va nous permettre de tester si nous le voulons les différentes fonctionnalités du script d’installation. Afin de faire cela nous avons du créer notre propre Framework de test, celui-ci, exécute le script principale avec des paramètres préalablement définit, et compare les informations attendues avec celles retournées.

Dans le cas ou celles-ci correspondent avec le test, ils sont considérés comme réussis.

1. **Problèmes rencontrés**

Durant ces étapes de développements, plusieurs problèmes on fait surfaces.

Le premier, était un problème qui venait de la manière dont on essayait de vérifier l’installation de docker. Nous sommes d’abord parties dans l’optique de vérifier si l’exe de docker était sur le pc, mais cela prenait trop de temp et nécessitait des permission administrateurs pour certains fichiers.

Nous avons donc décidé de vérifier l’existence du groupe docker-users, qui est automatiquement créer à l’installation de docker.

L’autre problèmes majeur, était le fait de récupérer le résultat de la commande permettant d’ajouté un utilisateur sur docker. Cela nous empêchait de faire valider et fonctionner nos tests, car certains résultats des commandes était déclaré comme « Write Host » et non « Write Output ».

Nous avons donc contourné ce problème, en redirigeant toutes les sorties dans notre variable de résultat, afin de la comparer avec le résultat attendu

# Conclusion

## Résultats finaux et conclusion

Les résultats finaux de notre projet consistent en un script nommé « install\_in\_nutshell » qui fonctionne sur Windows permettant d’installer automatiquement Teams, Visual Studio Code & Docker Desktop et en un second script nous permettant de vérifier le fonctionnement interne des fonctionnalités du premier script.

## Utilisation de l’IA

Bien que PowerShell ne nous était pas totalement inconnu, nous avons utiliser l’IA afin de nous remémorer certaines de ses exceptions et syntaxes particulière de plus nous nous sommes servi de l’intelligence artificielle afin de corriger les différentes fautes d’orthographes de notre rapport.