Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2022

DevSecOps for Microsoft .net

Etudiant : Quentin Beeckmans

Professeur : David Wannier

Déposé le : 23 janvier 2023

SOURCES DES ILLUSTRATIONS DE LA PAGE DE TITRE

https://fr.wikipedia.org/wiki/Fichier:Microsoft-Azure.png

i. RÉSUMÉ

L'objectif de ce document est de définir l'état de l'art du déploiement d'une application de façon automatisée et sécurisée sur la plateforme cloud de Microsoft Azure.

Depuis quelques années, les questions de sécurité gagnent de l'importance dans le développement de logiciels. Les termes de Cyberattaques, ou hacking deviennent monnaies courantes dans l'actualité. Cette aggravation massive de cybercriminalité depuis 2020 est dû notamment à l'augmentation importante du télétravail qui a obligé les entreprises à rendre disponibles leurs ressources internes en dehors du réseau de l'entreprise. Cette mise en place de solution d'urgence a montré des failles techniques de sécurité.

Cette situation ne fait que renforcer un besoin d'intégrer la sécurité au plus tôt dans le cycle de vie d'une application. La méthodologie DevSecOps tend à répondre à cela en traitant les problèmes de sécurité dès leur apparition.

De cette manière, ce travail consiste à déterminer quels sont les outils de sécurité de la méthodologie DevSecOps il est bon de mettre en place pour automatiser le processus de déploiement d'une application.

Mots-clés : DevSecOps, Azure, Pipeline, Cloud, Security

ii. AVANT-PROPOS

Ce travail de Bachelor a été réalisé dans le cadre de la formation en Informatique de Gestion à la Haute École Spécialisée de Suisse Occidentale Valais/Wallis validant les quatre années d'études à temps partiel.

Ce projet a été proposé et suivi par le professeur David Wannier de la Haute École Spécialisée de Suisse Occidentale, dans la filière informatique de gestion. Il vise principalement à automatiser le déploiement de Microsoft .net sur Microsoft Azure incluant les lancements de tests automatisés et une surveillance du système pour un projet de storemanager de panneaux photovoltaïques au Technopole de Sierre.

L'ensemble de l'accomplissement de celui-ci nécessite un total de 360 heures. Il est constitué d'un rapport écrit, dans lequel nous élaborons un état de l'art des outils DevSecOps sur Azure présents lors de la rédaction de celui-ci, d'une prise de position dans le choix des outils et d'un proof of concept qui permet la livraison de versions sécurisées selon les standards de la méthodologie DevSecOps.

Les principales difficultés rencontrées sont notamment la mise en place d'un système d'intégration et de déploiement de a à z sur une plateforme cloud de façon sécurisée et la mise en place d'outils d'analyse de sécurité de sources externes à la plateforme.

iii. REMERCIEMENTS

En premier lieu, je tiens à adresser mes remerciements au professeur David Wannier pour ses conseils et remarques tout au long de ce travail ainsi que pour la proposition de ce sujet très intéressant et actuel.

Je souhaite également remercier Jérémie Vianin et le professeur Alain Duc de la HES-SO Valais-Wallis pour leur disponibilité et pour m'avoir fourni les ressources et informations nécessaires à la réalisation de ce travail.

Je tiens à exprimer ma reconnaissance à Madame Nadine Lacrosse pour son aide précieuse dans la relecture du document.

Enfin, je tiens à remercier mes deux enfants et mon épouse pour leur patience ainsi que leur soutien moral qu'ils ont eu durant tout le long de ce travail de Bachelor.

iv. TABLE DES MATIÈRES

V. LISTE DES FIGURES VIII

VI. LISTE DES TABLEAUX XI

VII. LISTE DES ABRÉVIATIONS XII

INTRODUCTION 1

1.1. CONTEXTE 1
1.2. CAS D'ÉTUDE : STORAGEMANAGER 2
1.3. BUT DE L'ÉTUDE 3

2. ETAT DE L'ART 5

2.1. MÉTHODOLOGIE 5
2.2. DEVOPS 7
2.3. LES HUIT PHASES DU PIPELINE DEVOPS 9
2.3.1. Plan 10
2.3.2. Code 10
2.3.3. Build 10
2.3.4. Test 10
2.3.5. Release 10
2.3.6. Deploy 11
2.3.7. Operate 11
2.3.8. Monitor 11
2.4. DEVSECOPS 11
2.5. AZURE DEVOPS 12
2.5.1. Azure Board 14
2.5.2. Azure Repos 21
2.5.3. Azure Pipelines 22
2.5.3.1. Environnement 25
2.5.3.2. Release 26
2.5.3.3. Library 27
2.5.3.4. Task groups 27
2.5.3.5. Deployment groups 28
2.5.4. Azure Test plans 29
2.5.5. Azure Artefacts 30
2.5.6. Visual Studio marketplace 30

3. ANALYSE DES COMPOSANTS DU DEVSECOPS POUR AZURE DEVOPS 31

3.1. Pre-commit Hooks 32
3.2. Software Composition Analysis (SCA) 33
3.2.1. OWASP Dependency Check 33
3.2.2. Mend Bolt 33
3.2.3. Veracode 34
3.2.4. Tableau de Comparaison 34
3.2.5. Choix et justification 34
3.3. Static Application Security Testing (SAST) 35
3.3.1. SonarQube 35
3.3.2. SonarCloud 36
3.3.3. Checkmarx CxSAST 37
3.3.4. Tableau de Comparaison 38
3.3.5. Choix et justification 38
3.4. Dynamic Application Security Testing (DAST) 39
3.4.1. OWASP ZAP Scanner 39
3.4.2. Micro Focus Fortifier - Fortify WebInpect 40
3.4.3. Tableau de Comparaison 40
3.4.4. Choix et justification 40
3.5. Azure monitor 40
3.6. Synthèse des choix des composants 41

4. DEPLOIEMENT DU STORAGEMANAGER SUR AZURE SUIVANT UNE METHODOLOGIE DEVSECOPS ... 42

4.1. Architecture existante 42
4.2. Implémentation de la méthodologie Agile avec Azure Board 45
4.3. Implémentation des outils DevOps 49
4.3.1. Azure Pipelines (CI) 49
4.3.2. Tests unitaires 54
4.3.3. Azure Pipeline Release (CD) 57
4.3.4. Selenium (Tests fonctionnels) 62
4.3.5. Azure Test Plan 68
4.4. Implémentation des outils DevSecOps 71
4.4.1. OWASP Dependency Check (SCA) 71
4.4.2. SonarQube (SAST) (Solution abandonnée au profit de SonarCloud) 75
4.4.3. SonarCloud (SAST) 77
4.4.4. OWASP ZAP (DAST) 84
4.5. Synthèse des choix DevOps et DevSecOps 92
4.6. Dashboard (Surveillance) 92

5. AMELIORATIONS POSSIBLES 94

CONCLUSION 95

REFERENCES 96

ANNEXE I : PRÉPARATION DE L'ENVIRONNEMENT SUR AZURE 104

ANNEXE II : PROCÉDURE D'OBTENTION DE L'AUTORISATION D'EXÉCUTER UN PIPELINE GRATUIT SUR UN POOL D'AGENT HÉBERGÉ PAR MICROSOFT 106

ANNEXE III : INSTALLATION DE SONARQUBE VIA UNE INSTANCE DE CONTAINER (ACI) 107

ANNEXE IV : BURNDOWN CHART 112

ANNEXE V : PIPELINE YAML OWASP ZAP 114

ANNEXE VI : CONFIGURATION WEB APP 117

ANNEXE VII : WORK ITEM 118

ANNEXE VIII : BACKLOG DES SPRINTS 120

ANNEXE IX : PIPELINE YAML PRINCIPALE (CI, UNIT TEST, SONARCLOUD, OWASP DC) 122

ANNEXE X : AZURE DASHBOARDS 125

DÉCLARATION DE L'AUTEUR 126

V. LISTE DES FIGURES

Figure 1 - Interface du storagemanager 3
Figure 2 - Schéma waterfall 5
Figure 3 - Etape d'un pipeline CI/CD 9
Figure 4 - Cycle DevOps 9
Figure 5 - Parts de marché Mondial du Cloud Computing en 2022 13
Figure 6 - Licences Utilisateur Azure DevOps 13
Figure 7 - Concordance des services Azure avec le pipeline DevOps 14
Figure 8 - Fonctionnalités Azure Board 14
Figure 9 - Modèle basique Azure Board 15
Figure 10 - Modèle Agile Azure Board 15
Figure 11 - Modèle Scrum Azure Board 16
Figure 12 - Modèle CMI Azure Board 17
Figure 13 - Onglet Work Item Azure Board 17
Figure 14 - Onglet Boards Azure Board 18
Figure 15 - Onglet Backlogs Azure Board 18
Figure 16 - Onglet Sprints Azure Board 19
Figure 17 – Onglets Queries Azure Board : Exemple d'état des Bug 19
Figure 18 – Onglets Queries Azure Board : Graphique de requête 20
Figure 19 - Exemple de Azure Board : Exemple de plan de délivrable 20
Figure 20 – Fonctionnalités Azure Repos 21
Figure 21 - Exemple de commits sur Azure Repos 22
Figure 22 - Fonctionnalité Azure Pipeline 23
Figure 23 - Création d'un groupe de tâche 27
Figure 24 - Onglet Tasks groups Azure Pipeline 28
Figure 25 - Ajout d'un groupe de tâche 28
Figure 26 - Exemple d'un Azure Test Plan 29
Figure 27 - Carte de tarifs Azure Artefacts 30
Figure 28 - Top 10 OWASP 2021 31
Figure 29 - Représentation des outils DevSecOps 32
Figure 30 - Tableau des prix SonarCloud par lignes de code 37
Figure 31 - Notes comparatives Gartner Checkmarx vs SonarQube 38
Figure 32 - Vue page web monitoring storagemanager 42
Figure 33 - Vue page web admin storagemanager 43
Figure 34 - Exemple de mock de données 1/2 44
Figure 35 - Exemple de mock de données 2/2 44
Figure 36 - Structure des fichiers du storagemanager 45
Figure 37 – Importer des Work Items CSV dans Azure Board 47

Figure 38 - Erreur d'importation User Story 47
Figure 39 – Story Points du Sprint 3 48
Figure 40 – Exemple d'estimation en heure de l'effort de réalisation d'une tâche 48
Figure 41 - Burndown chart Azure Sprint 3 49
Figure 42 - Pipeline CI 49
Figure 43 - Résultat du Pipeline CI 54
Figure 44 - Tests unitaires 54
Figure 45 - Structure Visual Studio Unit Test 55
Figure 46 - Résultat Unit Test 56
Figure 47 - Pipeline release CD 57
Figure 48 – Tag du pipeline *Test* 57
Figure 49 - Configuration Default documents App Service 58
Figure 50 - Overview App Service 59
Figure 51 - Pipeline de release 59
Figure 52 - Déclencheur déploiement continu 60
Figure 53 - Condition de pré déploiement filtre d'artefact tags 60
Figure 54 - Condition de pré-déploiement filtre d'artefact manuelle 61
Figure 55 - Azure App Service deploy 61
Figure 56 - Configuration Azure App Service deploy 62
Figure 57 - Pipeline Selenium 62
Figure 58 - Animation au clic de l'image de la batterie 1/2 63
Figure 59 - Animation au clic de l'image de la batterie 2/2 63
Figure 60 - NuGet Package Selenium Visual Studio 64
Figure 61 - Structure Visual Studio Selenium 64
Figure 62 - Intégration d'une tâche VSTest pour Selenium 67
Figure 63 - Configuration de la tâche de test Selenium dans le pipeline de release 67
Figure 64 - Résultat du test Selenium dans le pipeline release 68
Figure 65 - Création d'un Test Plan via le Boards 69
Figure 66 - Exemple de Tests Case 69
Figure 67 - Exécution d'un test plan 1/2 70
Figure 68 - Formulaire de Test Case 70
Figure 69 - Pipeline SCA 71
Figure 70 - Artefact produit par OWASP Dependency Check (SCA) 72
Figure 71 - Résultat du test OWASP Dependency Check dans le pipeline 72
Figure 72 - Problème critique trouvé avec OWASP Dependency Check 73
Figure 73 - NuGet Package Manager log4net 73
Figure 74 - Git push Update Dependency Log4Net 74
Figure 75 - Résultat du test OWASP Dependency Check dans le pipeline après correction erreur 74

Figure 76 - Pipeline SonarQube abandonné 75
Figure 77 - Pipeline SonarCloud 77
Figure 78 - Génération d'un token Azure pour SonarCloud 1/2 79
Figure 79 - Génération d'un token Azure pour SonarCloud 2/2 79
Figure 80 - SonarCloud extension 80
Figure 81 - Nouvelle connexion de service SonarCloud 80
Figure 82 - Résultat de notre build avec SonarCloud 82
Figure 83 - Résultat de la première analyse SonarCloud 83
Figure 84 - Bug identifiés par SonarCloud 83
Figure 85 - Résultat SonarCloud après la correction du bug 84
Figure 86 - Pipeline DAST 84
Figure 87 - Tâche OWASP ZAP Scanner 85
Figure 88 - Résultat du test rapide OWASP ZAP 1/2 89
Figure 89 - Résultat du test OWASP ZAP 2/2 90
Figure 90 - Résultat du test agressif OWASP ZAP 2/2 90
Figure 91 - Menu triggers du pipeline 91
Figure 92 - Configuration du déclencheur pour le build OWASP ZAP 91
Figure 93 - Programmation du build OWASP ZAP 91
Figure 94 - Pipeline complet DevSecOps 92
Figure 95 - Dashboards 92
Figure 96 - Requête état de bug 93
Figure 97 - Résultats de la requête dans Overview > Dashboards 93
Figure 98 – Commande git à utiliser pour envoyer son code sur Azure 105
Figure 99 - Formulaire de demande 106
Figure 100 - Créer un groupe de ressource 107
Figure 101 - Configuration Container Instance étape 1/2 108
Figure 102 - Configuration Container Instance étape 2/2 109
Figure 103 - Azure Cloud Shell interface 110
Figure 104 - Instance d'un container SonarQube 111
Figure 105 - Burndown Chart Sprint 1 112
Figure 106 - Burndown Chart Sprint 2 112
Figure 107 - Burndown Chart Sprint 3 113
Figure 108 - Configuration Web App 117
Figure 109 - Sprint 1 120
Figure 110 - Sprint 2 120
Figure 111 - Sprint 3 121
Figure 112 - Dashboard Azure 125

VI. LISTE DES TABLEAUX

Tableau 1 - Pipeline hébergé par Microsoft 24
Tableau 2 - Pipeline auto hébergé 25
Tableau 3 - Comparaison des outils SCA 34
Tableau 4 - Tableau des prix SonarQube Developer Edition par lignes de code 36
Tableau 5 - Comparaison des outils SAST 38
Tableau 6 - Comparaison des outils DAST 40
Tableau 7 - Tarification Azure Monitor 41
Tableau 8 - Azure Pipeline hébergé sur des image de machine virtuelle 52
Tableau 9 - Outils open-source Microsoft Security DevOps 94

VII. LISTE DES ABRÉVIATIONS

AAD	Azure Active Directory
ACI	Azure Container Instances
CMI	Capability Maturity Model Integration
CSP	Content Security Policy
CD	Continuous Delivery
CI	Continuous Integration
DevOps	Development Operations
DevSecOps	Development Security Operations
DER	Distributed Energy Resources
DAST	Dynamic Application Security Testing
XP	Extrême Programming
IDE	Integrated Development Environment
IIS	Internet Information Services
LOC	Lines of code
Npm	Node Package Manager
QA	Quality Assurance
SAAS	Software as a Service
SCA	Software Composition Analysis
SMA	Solar Technology AG
SAST	Static Application Security Testing
VCS	Systèmes de contrôle de version
TFS	Team Foundation Server
TFVC	Team Foundation Version Control
OT	technologies operational
VM	Virtual machine
VSTS	Visual Studio Team Services
YAML	YAML Ain't Markup Language
ZAP	Zed Attack Proxy

INTRODUCTION

1.1. Contexte

Avec les récents évènements occasionnant l'augmentation des prix de l'électricité et le spectre d'une pénurie cet hiver, il est normal d'observer une augmentation significative des installations solaires. En 2021, la Suisse a enregistré une augmentation de 43% des installations par rapport à l'année précédente. Pour sortir des énergies fossiles et du nucléaire, le pays aurait besoin de 13 fois plus de puissance solaire qu'aujourd'hui selon l'Office Fédérale de l'énergie (OFEN) (Les installations solaires ont augmenté de 43% en 2021, 2022).

Au fur et à mesure que ces installations solaires sont ajoutées au réseau, de nombreux d'outils apparaissent pour fournir des informations sur la production d'énergie solaire, leur gestion ainsi que leur stockage.

Ces outils sont de plus en plus accessibles par internet ce qui permet leur gestion à distance. Ils constituent néanmoins une opportunité pour des accès malveillants. Les appareils à technologie d'exploitation[1] tels que les onduleurs solaires[2], lorsqu'ils sont connectés à internet, présentent des risques plus élevés par rapport à ceux qui ne le sont pas.

Il existe de nombreux outils de sécurité mal intégrés, peu flexibles et rapidement obsolètes. Ces outils ont souvent des processus d'installation et de maintenance complexes et fastidieux. Les difficultés techniques sont amplifiées par le manque d'experts en sécurité dans ce domaine et par le manque de communication entre les opérations informatique et l'entreprise (Tristan, 2022).

« Historiquement, le cyber-risque pour le solaire était relativement mineur, étant donné le peu de systèmes déployés et parce que la plupart des onduleurs solaires ne communiquaient pas pour la surveillance ou le contrôle » (Bureau de U.S. DEPARTMENT OF ENERGY, 2022).

Les onduleurs sont l'interface entre les panneaux solaires et le réseau ce qui fait que sans une mise à jour régulière de leur logiciel, ces données pourraient être interceptées et manipulées.

Les hacker sont sans cesse à la recherche de systèmes vulnérables avec des objectifs bien souvent de nature criminelle qui visent à nuire au bon fonctionnement des entreprises.

[1] Les technologies d'exploitation font référence à l'utilisation de matériel et de logiciels pour contrôler des équipements industriels.

[2] Un onduleur est un appareil permettant de transformer le courant continu des panneaux solaires en courant alternatif.

Selon les directives de Solar Technology AG (SMA)[3], aucun système de communication de données ne devrait être connecté à internet sans que des mesures de protections des installations photovoltaïque et autres équipements n'aient été prévues pour éviter des attaques.

Dans le cas contraire, les systèmes non protégés peuvent être utilisés pour pénétrer dans le réseau derrière le routeur internet et accéder de cette manière à presque tous les appareils du réseau. Les risques possibles sont l'espionnage des données confidentielles, la manipulation des données ou de mener d'autres attaques par ce biais. Les conséquences potentielles sont notamment l'usurpation d'identité, des pertes financières dues à une production d'énergie minime, des erreurs de tarif de consommation et d'injection sur le réseau ou encore à un endommagement des appareils (Information technique CYBERSÉCURITÉ PUBLIQUE Directives pour une communication sûre avec les installations, 2022).

Les modifications non autorisées des commandes de l'onduleur ou des communications sont appelées des failles de sécurité cyber-physique. En effet, le résultat est un changement de la tension ou du courant électrique que l'onduleur injecte dans les maisons ou le réseau ce qui pourrait par exemple occasionner des incendies (Bureau de U.S. DEPARTMENT OF ENERGY, 2022).

SMA donne une liste de précautions pour les techniciens qui installent et utilisent leurs produits, outre ces bonnes pratiques telles que faire des sauvegardes régulièrement ou par exemple utiliser une connexion VPN pour accéder au système depuis l'extérieur, il est nécessaire de réfléchir à la sécurité des programmes destinés à utiliser ou à modifier le comportement des appareils dès leur conception en l'intégrant à toutes les étapes du développement.

1.2. Cas d'étude : storagemanager

Le storagemanager est un projet informatique développé en .NET[4] et permet la visualisation des flux d'énergies des panneaux photovoltaïques installés sur les toits des bâtiments du Technopole à Sierre en Suisse.

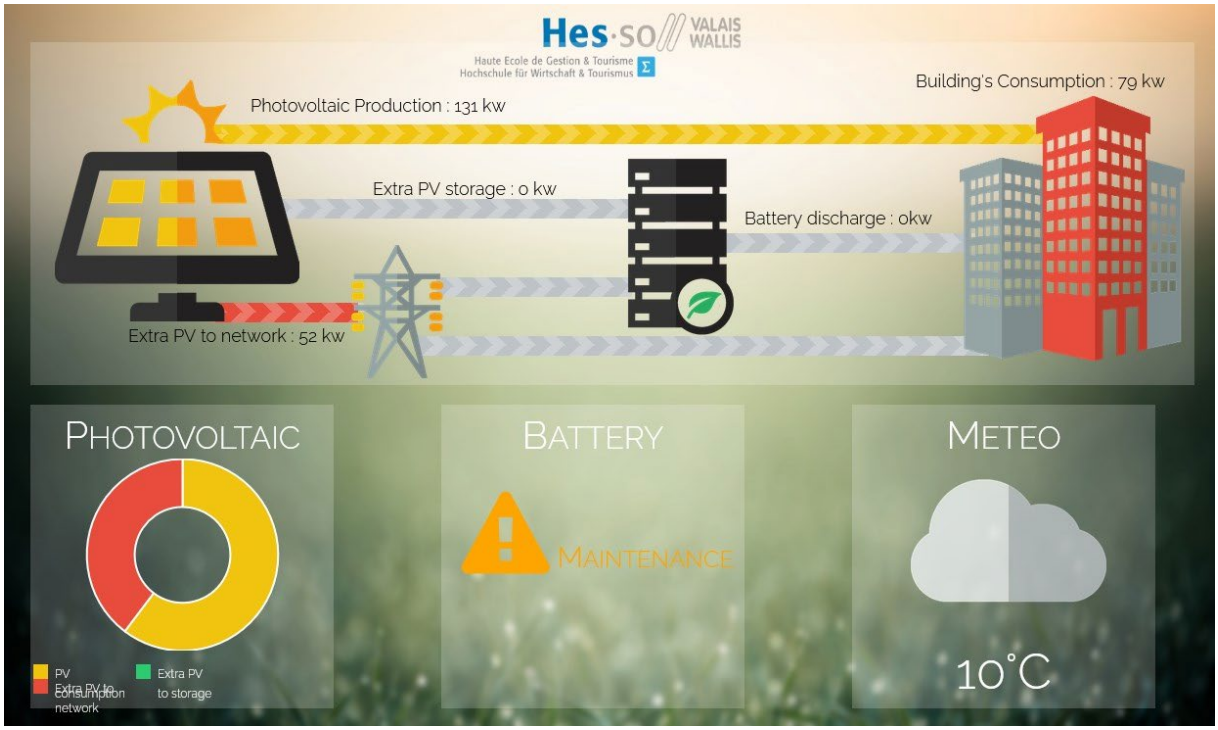
Cette interface permet également de contrôler la recharge et la décharge de la batterie reliée à ces panneaux de sorte d'optimiser la production et la consommation des bâtiments (Wannier, 2017).

[3] SMA : Solar Technology AG est un fabricant allemand d'onduleurs pour installations photovoltaïques destinées à l'injection du courant sur le réseau, à des systèmes autonomes ou à des systèmes de secours. Il s'agit du premier fabricant mondial par son chiffre d'affaires, et le plus connu en France.

[4] .NET est un framework de développement de logiciels créé par Microsoft. Il permet aux développeurs de créer des applications pour Windows, MacOs, Linux, iOS et Android en utilisant des langages tels que C#, F#, Visual Basic ou encore C++.

Si la production des panneaux est supérieure à la demande, le stockage de l'énergie peut s'activer en fonction des prochaines heures. A l'inverse, lorsque la production photovoltaïque est trop faible pour la consommation du bâtiment, la batterie est utilisée pour compenser le besoin supplémentaire de puissance. Lorsque la batterie est vide ou lorsqu'il fait nuit, le système utilisera l'énergie du réseau électrique (HES-SO Valais, 2017).

Figure 1 - Interface du storagemanager



Source : Données de l'auteur

1.3. But de l'étude

Ces dernières années, nous avons connu de nouvelles méthodologies de travail dans le développement logiciel qui ont permis d'augmenter la vitesse de développement pour répondre à une demande constante aux entreprises afin d'être les plus compétitifs. Cette pression de rendement a fait augmenter également les vulnérabilités des logiciels car les tests de sécurité sont effectués à la fin du cycle de développement et rarement après qu'ils soient en production, les logiciels peuvent donc présenter des risques sans que personne ne s'en aperçoive (Penot, 2021, p. 4).

Cette méthodologie de coordination entre les équipes de développement et des opérations appelée DevOps, présente donc des manquements au niveau de la sécurité.

Il est donc nécessaire que le développement des applications bénéficie d'un renforcement de la sécurité.

Le DevSecOps vise à répondre à ces manquements. « Le DevSecOps est un changement de culture dans l'industrie du logiciel qui vise à intégrer la sécurité dans des cycles de développement rapide et itératif qui sont typiques du développement et du déploiement d'applications modernes, également connu sous le nom de mouvement DevOps » (Penot, 2021, p. 1).

Actuellement, le projet storagemanager est déployé manuellement. L'objectif principal de ce document est la mise en place de la méthodologie DevSecOps en automatisant le déploiement du storagemanager sur une plateforme de « cloud computing » appelé aujourd'hui Microsoft Azure.

Ce faisant, un état de l'art des outils de DevSecOps sur Azure est réalisé afin de décrire ce qui est proposé actuellement sur la plateforme.

Nous passerons ensuite à la partie analyse où des critères nous permettrons de décider quels sont les outils retenus dans notre intégration.

Enfin, une mise en pratique de ces outils constituera la solution retenue pour le proof of concept.

2. ETAT DE L'ART

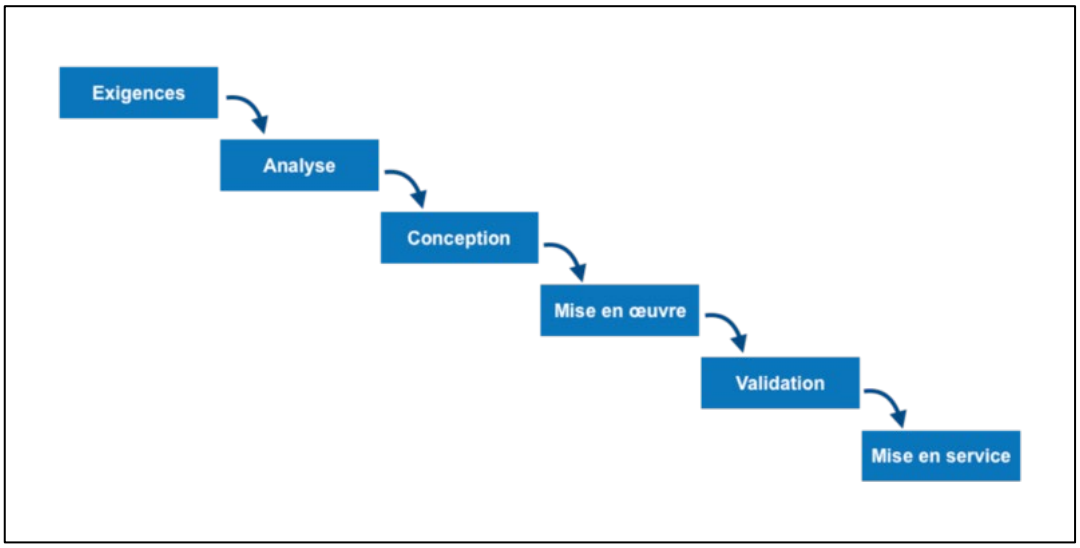
Avant d'aborder l'état de l'art des différents outils liés au DevSecOps sur Azure, la plateforme cloud de Microsoft, nous allons passer en revue les étapes historiques qui ont fait du DevSecOps un besoin de changement de culture dans la société du développement logiciel.

2.1. Méthodologie

2.1.1. Waterfall[5]

L'approche waterfall est une méthodologie classique de développement caractérisé par une suite d'étapes. Cette méthode part du principe que toutes les exigences peuvent être rassemblées à l'avance. A la fin de chaque étape, l'équipe en charge du projet passe à la suivante sans (ou peu de) retour en arrière. C'est au total une succession de six étapes : recueillir les besoins, analyse des exigences, conception du produit, réalisation du produit sur base des spécifications, validation du produit pour s'assurer de sa conformité aux exigences, la mise en service.

Figure 2 - Schéma waterfall



Source : (Modèle en cascade générique, 2019)

L'inconvénient majeur de cette méthodologie est son manque de flexibilité à cause de son déroulement séquentiel, car tout changement ou problème non anticipé provoquera des retards et des coûts supplémentaires (Kai, Wohlin, & Baca , 2009, p. 3).

Cette méthode reste l'une des plus utilisées dans la gestion de projet du fait de son fonctionnement simple et logique qui permet d'établir facilement le budget, le temps et le travail

[5] Waterfall : traduction en cascade

nécessaire. Le client quant à lui exprimera son besoin durant la première phase pour ensuite recevoir le produit final à la fin sans son intervention durant le processus. Le risque est que le produit ne corresponde pas réellement à ses attentes.

2.1.2. Agile

La méthodologie agile est une approche dite itérative qui permet de découper un projet en plusieurs étapes réalisables sur une durée de plusieurs semaines fixées à l'avance appelé Sprint.

Les story points sont une méthode utilisée pour estimer l'effort nécessaire pour réaliser une tâche ou un projet, ce sont des outils clés pour planifier les sprints ainsi que pour évaluer l'avancement du projet. Pour utiliser les story points, une équipe de développement doit d'abord identifier les tâches ou les user stories qui doivent être accomplies. Ensuite, l'équipe attribue un nombre de story points à chaque tâche en se basant sur sa complexité, son incertitude et son effort estimé. Cette estimation est généralement faite en utilisant une technique de planification de poker.

La planification de poker est une technique d'estimation collaborative pour attribuer des story points aux tâches ou aux user stories. Chacun des membres de l'équipe reçoit des cartes marquées avec des nombres (selon la suite de Fibonacci), et chacun donne son estimation individuellement. Ensuite, l'équipe discute pour arriver à un consensus sur le nombre de story points pour la tâche ou l'histoire.

Durant chaque itération, une partie du projet final est développée et présentée à la fin pour être validé ou non par le client. C'est sous forme de fonctionnalité que le projet se construit jusqu'à l'achèvement total du produit.

Cette méthode accepte plus facilement les changements ou les imprévus que la méthode traditionnelle waterfall en s'adaptant au fur à mesure des cycles d'itération. La validation client tout au long du projet assure que le produit correspondra à ses attentes lorsqu'il sera terminé.

2.1.3. Extrême Programming

La méthodologie extrême programming (XP) est une méthode de gestion de projet qui applique à l'extrême les principes du développement agile, en se concentrant sur les besoins du client, en mettant en place un développement itératif et une intégration continue (Qu'est-ce que la méthode eXtreme Programming ?, 2017).

L'Extrême programming repose sur cinq valeurs fondamentales : la communication, la simplicité, le feedback, le courage et le respect qui se déclinent en une liste de treize pratiques (Kent Beck, 2004, p. 57).

« Cette méthode ne peut pas fonctionner avec des grandes équipes, si le travail en binôme est impossible ou encore si les feedback sont longs et difficiles à obtenir » (Qu'est-ce que la méthode eXtreme Programming ?, 2017).

2.2. DevOps

Traditionnellement, le cycle de vie du développement d'un logiciel s'établit entre des équipes cloisonnées prenant chacune en charge des tâches spécifiques. D'une part l'équipe de développement responsable de l'écriture du code, des tests, de l'assurance qualité et de la préparation du déploiement et de l'autre l'équipe d'exploitation chargée de déployer le code sur les serveurs, de la coordination avec les clients et de fournir un retour d'information de ceux-ci aux développeurs (Machiraju & Gaurav, 2018, p. 1). Cette méthodologie présente plusieurs inconvénients dû au manque de communication.

Le DevOps est une méthodologie de développement et un ensemble de pratiques qui soutiennent une culture organisationnelle ce qui permet d'améliorer la collaboration et la communication entre les équipes. Elle permet une unification des processus et des outils sur une orientation client (DevOps, 2022).

Cet ensemble de processus et d'outils automatisés s'appelle un pipeline. Bien que chaque pipeline puisse différer d'une organisation à l'autre, elles utilisent des composants fondamentaux similaires tels que l'intégration continue (CI), le déploiement continu (CD), les tests automatisés, une phase de validation et de reporting (Hall, 2022).

Cette simplification du développement logiciel, du déploiement et des opérations permet de réduire les délais de mise sur le marché, le taux d'échec des nouvelles versions et les délais entre les correctifs.

2.2.1. Intégration continue

L'intégration continue est une pratique qui consiste à automatiser l'intégration des changements de code réalisés par plusieurs développeurs dans un seul et même projet via un outil de gestion de version.

Chaque intégration est ensuite vérifiée par un build[6] et par des tests automatisés pour vérifier que chaque modification n'entraine pas de régression[7]. Il s'agit d'une bonne pratique Agile et DevOps (Duvall, Matyas, & Gépver, 2007).

Fusionner régulièrement le code permet de réduire le risque d'énormes conflits d'intégration à la fin d'un projet ou d'un cycle. Cela permet donc de gagner du temps en identifiant et en résolvant très tôt ces conflits.

2.2.2. Livraison continue

La livraison continue est l'extension de l'intégration continue. C'est une pratique qui consiste à ce que les modifications de code soient automatiquement testées et préparées à être déployées dans un environnement de production. « Chaque révision apportée déclenche un flux automatique qui crée, teste et planifie la mise à jour » (Qu'est-ce que la livraison continue ?, s.d.). Le développeur décide ensuite manuellement dans quel environnement déployer.

La livraison continue permet aux développeurs d'automatiser les tests au-delà des simples tests unitaires , il peut s'agir de tests de charge, d'intégration, etc. afin de vérifier différents aspects d'une mise à jour avant de la déployer (Qu'est-ce que la livraison continue ?, s.d.).

Ce processus a pour avantage d'automatiser le processus de publication de logiciel, de trouver et corriger les bug plus rapidement et de livrer des mises à jour plus rapidement en disposant toujours d'un artefact[8] prêt au déploiement.

2.2.3. Déploiement continu

Le déploiement continu est le processus qui complète la livraison continue en automatisant le lancement d'une application dans un environnement de production. Avec cette pratique, chaque modification réussie selon les tests mis en place seront publiés aux utilisateurs finaux sans aucune autre intervention humaine. Le déploiement continu accélère considérablement la boucle de rétroaction avec les utilisateurs finaux en répondant le plus rapidement aux modifications de ceux-ci (Pittet , 2022).

Cette approche présente toutefois des risques, elle nécessite un investissement de départ considérable pour que les tests automatisés puissent s'adapter à un large éventail d'étapes, mais si

[6] Build ou construction est le processus de conversion des fichiers de code source en artefact logiciel autonome pouvant être exécuté sur un ordinateur, ou le résultat de cette opération.

[7] Régression logicielle est un bug logiciel qui fait qu'une fonctionnalité cesse de fonctionner après un certain évènement (par exemple une mise à jour du système, un patch du système, ou un changement d'heure).

[8] Un artefact est un fichier ou collection de fichiers qui représente une version d'une application.

des erreurs ne sont pas détectées précédemment, celles-ci seront déployées en production et impacteront directement les utilisateurs (Qu'est-ce que l'approche CI/CD ?, 2018).

Figure 3 - Etape d'un pipeline CI/CD

Source : (Qu'est-ce que l'approche CI/CD ?, 2018)

2.2.4. Outils d'intégration continue et de déploiement continu (CI/CD)

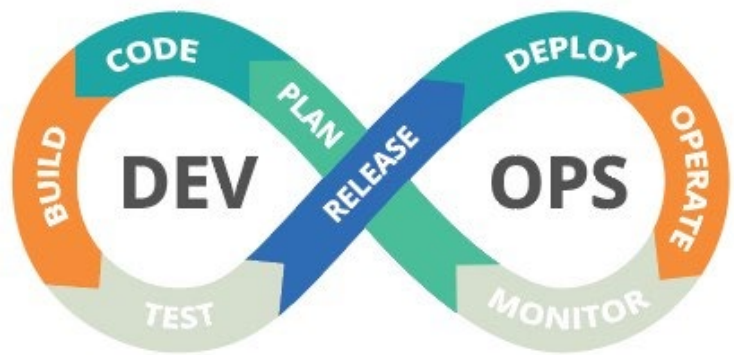
Dans notre cas d'étude, l'outil pipeline de Azure permet à lui seul la mise en place d'une intégration et d'une livraison continue pour générer et tester le code de manière persistante et livrer un produit performant de haute qualité.

Le principal concurrent de Azure Pipeline est Jenkins qui peut être lui aussi intégré dans Azure. Jenkins est un outil open source écrit en Java qui permet également de mettre en place un CI/CD. Malgré ce constat, l'utilisation de plusieurs outils nécessite un investissement supplémentaire en formation et maintenance. Dans notre cas, l'utilisation du service interne à Azure DevOps permet de réduire le nombre d'outils. De plus, la possibilité de définir le pipeline sous forme de code (YAML) offre l'avantage d'être géré comme un fichier source et donc de passer par le processus de révision de code standard, augmentant ainsi la qualité (Brown, 2022).

2.3. Les huit phases du pipeline DevOps

En raison de sa nature continue, les phases du cycle de vie du DevOps sont représentées sous la forme d'une boucle infinie.

Figure 4 - Cycle DevOps



Sources : Données de l'auteur

2.3.1. Plan

La phase de planification couvre tout ce qui se passe avant que les développeurs ne commencent à écrire du code. Le product owner recueille les exigences et commentaires auprès des parties prenantes et des clients pour en créer une feuille de route afin de guider le développement. Cette feuille de route est ensuite décomposée en User Stories qui permettront de planifier les sprints en les allouant à l'équipe de développement (Jakob, 2019).

2.3.2. Code

Durant la phase de code, les développeurs vont coder les tâches qu'ils se sont assignées à l'étape précédente. L'équipe suit alors des standards pour faciliter le processus de développement et aider à appliquer un style de code cohérent compréhensible pour les autres développeurs (Jakob, 2019).

2.3.3. Build

La phase de Build ou construction est l'étape la plus importante du DevOps. Une fois que le développeur a terminé une tâche, il valide son travail dans un référentiel de code partagé, il demande alors une fusion de son nouveau code avec la base de code partagée. Généralement, un autre développeur passe en revue les modifications qu'il a apporté et après vérification qu'il n'y a pas de problème, il approuve la demande de fusion. Cela déclenche un processus automatisé qui construit la base de code et exécute des tests unitaires et d'intégration écrit par le développeur pour identifier toute régression. En cas d'échec, le développeur est averti que la fusion est refusée afin de résoudre le problème (Jakob, 2019).

2.3.4. Test

Le build est ensuite déployé dans un environnement intermédiaire pour effectuer des tests fonctionnels plus approfondis. Cet environnement ressemble au plus possible à l'environnement de production pour permettre de faire des tests manuels et automatisés en conditions réelles. Des utilisateurs peuvent utiliser l'application comme le ferait le client (Test d'acceptation utilisateur) pour mettre en évidence les problèmes ou améliorations avant le déploiement en production. Des analyses de sécurité et de performances peuvent être également effectuées à différents niveaux (tests unitaires, tests d'intégration, tests de vulnérabilités, tests de charge, etc.) (Jakob, 2019).

2.3.5. Release

Après avoir subi toute une série de tests, la phase de release est le moment où une version est prête à être déployée dans l'environnement de production. En fonction du besoin, le déploiement peut être automatisé à partir de cette étape mais une organisation peut vouloir contrôler le moment où les build sont mis en production (Jakob, 2019).

2.3.6. Deploy

Dans cette phase, l'application est déployée dans l'environnement de production et mise à disposition des utilisateurs. Cet environnement a la même infrastructure que l'environnement de test.

2.3.7. Operate

Les équipes d'opération s'assurent du bon fonctionnement de l'application en adaptant par exemple l'hébergement à la charge des pics de connections des utilisateurs, en faisant des backup et en collectant les retours utilisateurs pour aider à améliorer l'application.

2.3.8. Monitor

Cette dernière phase sert à surveiller l'environnement en se basant sur les données collectées à partir du comportement des utilisateurs, des performances, des erreurs. Ces informations sont ensuite transmises au responsable du projet et à l'équipe de développement.

2.4. DevSecOps

De plus en plus d'organisations adoptent une approche DevOps, ce qui permet un déploiement plus rapide des applications mais souvent au détriment de la sécurité qui se retrouve en fin de processus. Les équipes de développement en charge de la sécurité sont alors confrontés à des deadlines très courtes. La découverte d'un grave problème de sécurité avant la sortie prévue d'une application prolonge les cycles de développement de celui-ci. Ce que les entreprises essaient justement d'éviter. L'intégration de la sécurité dans l'ensemble de l'organisation est devenue plus critique que jamais. C'est de ce besoin que le DevSecOps a vu le jour.

Le DevSecOps est une combinaison des premières lettres de développement, sécurité et opération. C'est la fusion des processus de développement et des opérations (DevOps) avec les processus de sécurité que l'on intègre à tous les acteurs de l'organisation.

Selon Glenn Wilson (2020, p. 37), le DevSecOps est plus compliqué que d'intégrer des ingénieurs sécurité dans le développement DevOps car ceux-ci ont chacun un rôle spécifique dans la sécurité et il y a rarement assez d'ingénieurs sécurité dans une organisation pour les affecter à chaque équipe DevOps. Le DevSecOps est plutôt un changement de mentalité des différentes équipes qui participent au développement logiciel car la sécurité doit être considérée comme un aspect essentiel de la qualité d'un logiciel. De plus, si les vulnérabilités de sécurité sont découvertes plus tôt dans le processus cela permettra de réduire les coûts de leur correction.

On appelle également cette pratique le *shift left* (déplacement vers la gauche), pour le fait de décaler cette sécurité vers le début du cycle de développement vu de façon linéaire.

2.5. Azure DevOps

Microsoft a lancé en 2013 le logiciel Visual Studio Online en tant qu'offre de service de Visual Studio sur la plateforme Microsoft Azure. Il fut renommé en 2015 en « Visual Studio Team Services » (VSTS) et par la suite, en Azure DevOps Services en 2018 qui est l'appellation actuelle pour la version cloud (Visual Studio, 2022). Le cloud computing utilise des serveurs distants sur internet pour gérer, stocker et traiter des données au lieu d'utiliser un serveur local.

La version sur site, anciennement appelée *Team Foundation Server* (TFS) et *Visual Studio Team System* (VSTS) s'appelle actuellement quant à elle : *Azure Devops Server*.

Azure DevOps fournit une grande variété de services aux équipes DevOps afin qu'elles puissent planifier le travail, de collaborer sur le développement de code, de construire et déployer des applications (Zaal, Demiliani, & Malik, 2020, p. 8).

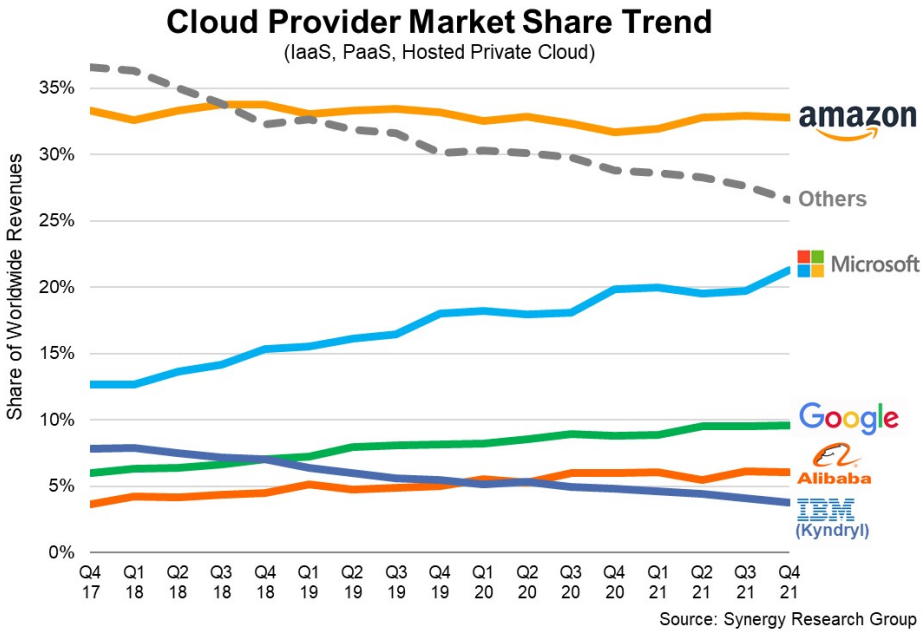
Selon un article du site Xenonstack (Gursimran, 2021), les fonctionnalités du cloud computing comme Azure Devops Service sont :

- Libre-service à la demande : les consommateurs peuvent disposer de capacités informatiques selon leurs besoins sans nécessiter d'interaction humaine avec chaque fournisseur de services.
- Large accès au réseau : le cloud offre à son utilisateur la facilité d'accès au réseau et aux services de n'importe quelle partie du monde.
- Mutualisation des ressources : le cloud est un vaste réservoir de ressources. Il propose une large gamme de produits et services.
- Élasticité rapide : l'élasticité est la capacité d'évoluer à la demande vers le haut ou vers le bas selon les besoins afin que l'utilisateur ne subisse aucune interruption.
- Services mesurés : il ne faut payer que ce qui est utilisé. Le cloud ne facture pas ses utilisateurs inutilement et leur offre l'avantage de ne payer que les ressources qui ont été utilisées.
- Multi-ténacité : le Cloud offre à son utilisateur la fonctionnalité de multi-tenant. La multi-ténacité est une architecture logicielle qui autorise les logiciels dans plusieurs serveurs, qui peuvent être utilisés pour servir plusieurs locataires.

Les principaux concurrents de Azure DevOps est AWS[9] et Google.

[9] AWS : Amazon Web Services : division du groupe américain de commerce électronique Amazon, spécialisée dans les services de cloud computing à la demande pour les entreprises et particuliers.

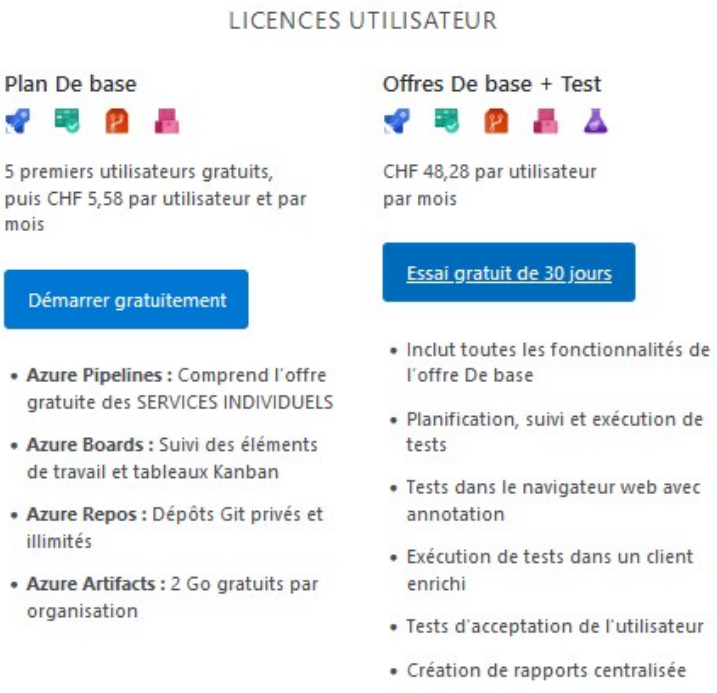
Figure 5 - Parts de marché Mondial du Cloud Computing en 2022



Source : (Alors que les dépenses trimestrielles dans le cloud grimpent à plus de 50 milliards de dollars, Microsoft occupe une place plus importante dans le rétroviseur d'Amazon, 2022)

Azure DevOps Services propose deux plans de licences utilisateurs possibles :

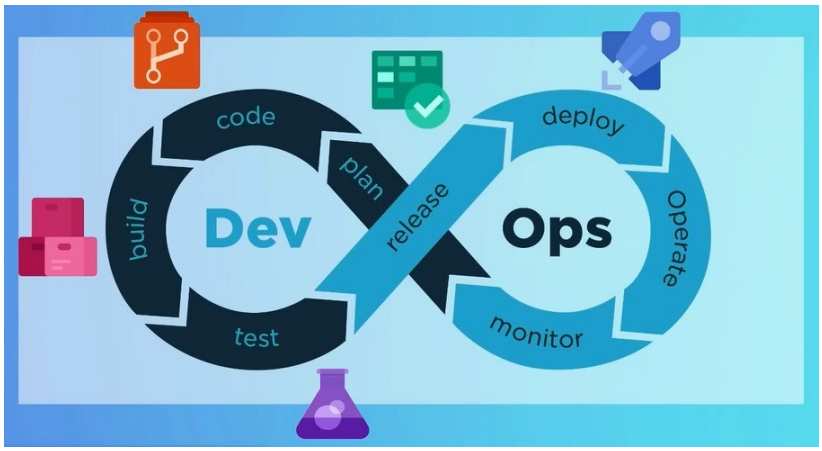
Figure 6 - Licences Utilisateur Azure DevOps



Source : (Tarification Azure DevOps, 2022)

Azure DevOps dans sa version complète comporte cinq services qui peuvent être utilisé pour supporter les équipes de développement tout le long du cycle de vie de l'application : Azure Board, Azure repos, Azure Pipeline, Azure Artefact et Azure Test Plan.

Figure 7 - Concordance des services Azure avec le pipeline DevOps

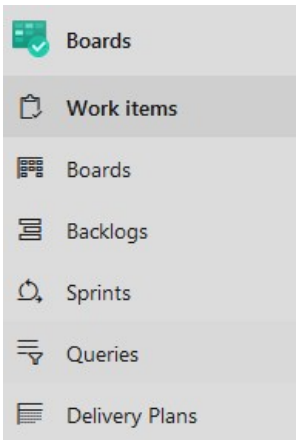


Source : (Nana, 2022)

2.5.1. Azure Board

Azure Board est un outil de gestion de projet proposé par Azure DevOps qui permet aux équipes de développement de planifier, de suivre et de livrer le travail de l'équipe. Il offre une vue d'ensemble du travail en cours et des objectifs à atteindre, ainsi que des indicateurs de progression et de performance de l'équipe.

Figure 8 - Fonctionnalités Azure Board



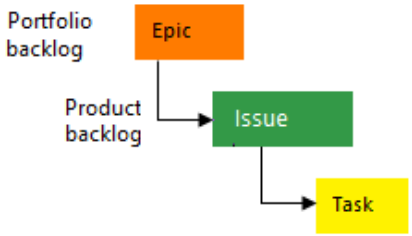
Source : Données de l'auteur

Au démarrage d'un projet, les équipes doivent décider quel modèle de processus doit être utilisé (Zaal, Demiliani, & Malik, 2020, p. 43) :

- Basic

Il s'agit du modèle le plus simple permettant d'utiliser les Epics, les problèmes, les tâches pour suivre le travail.

Figure 9 - Modèle basique Azure Board

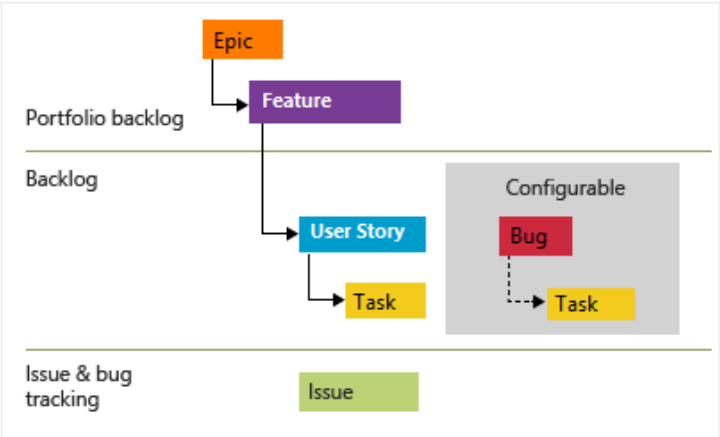


Source : (Kathryn, et al., Choisissez un flux de processus ou un modèle de processus pour travailler dans Azure Boards, 2022)

- Agile

Le modèle Agile permet de suivre les différents type de travaux tels que les fonctionnalités, les User Stories et les tâches. Ce modèle utilise le tableau Kanban[10] pour suivre les user stories et les bugs.

Figure 10 - Modèle Agile Azure Board



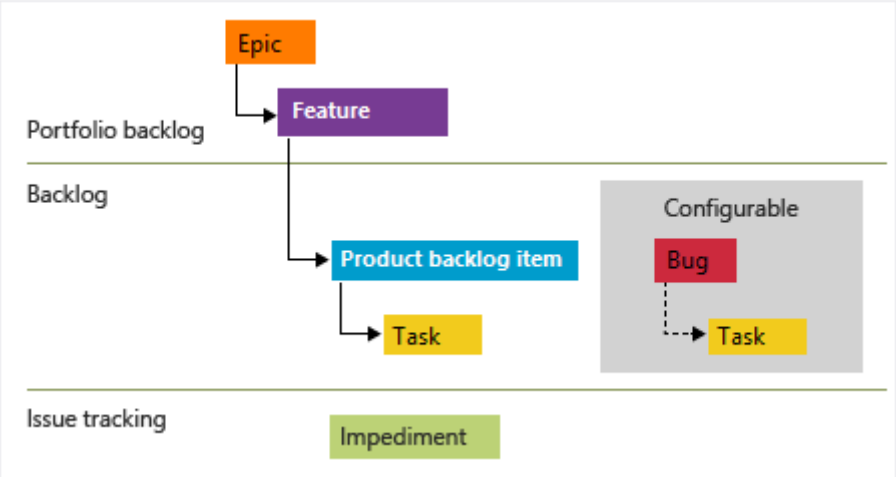
Source : (Kathryn, et al., Choisissez un flux de processus ou un modèle de processus pour travailler dans Azure Boards, 2022)

[10] Kanban est une méthode de gestion de projet à partir d'un système de carte à déplacer selon l'état de la tâche en cours

- Scrum[11]

Lorsque l'équipe pratique le SCRUM, ce modèle permet de suivre les éléments des products backlogs[12] et les bugs sur le tableau Kanban ou de décomposer les products backlogs et les bugs en tâches dans le tableau des tâches.

Figure 11 - Modèle Scrum Azure Board



Source : (Kathryn, et al., Choisissez un flux de processus ou un modèle de processus pour travailler dans Azure Boards, 2022)

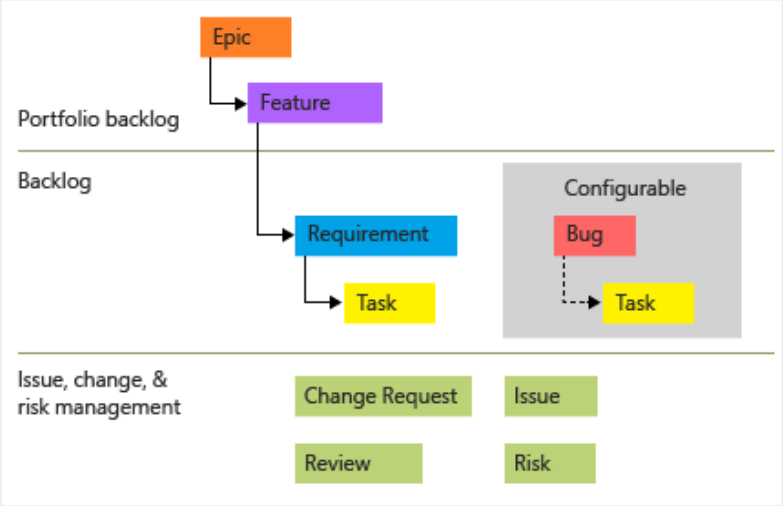
- Capability Maturity Model Integration (CMI)

Le processus CMI est adapté pour suivre les méthodes de projet plus formelles exigeant un cadre pour l'amélioration des processus et un enregistrement vérifiable des décisions. Il est alors possible de suivre les exigences et les demandes de changements, les risques et les révisions.

[11] Scrum est une méthodologie agile consacrée à la gestion de projet.

[12] Product backlog : est une liste d'éléments ou de fonctionnalités nécessaires pour atteindre les objectifs, le tout classé par ordre de priorité. Elle permet à ses membres de suivre leurs tâches.

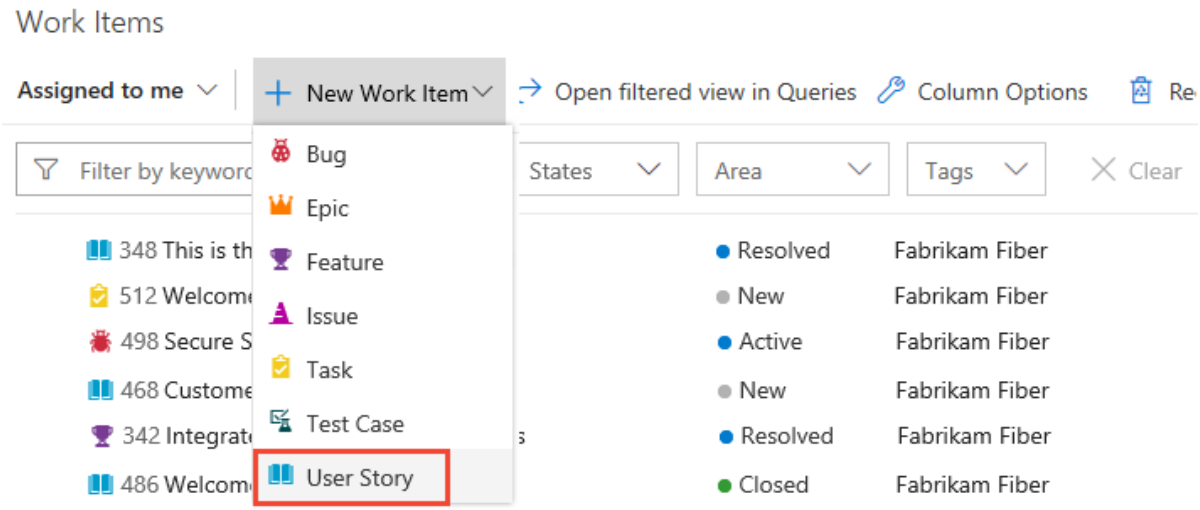
Figure 12 - Modèle CMI Azure Board



Source : (Kathryn, et al., Choisissez un flux de processus ou un modèle de processus pour travailler dans Azure Boards, 2022)

L'onglet **Work Item** permet d'ajouter et rechercher rapidement des éléments de travail en filtrant efficacement.

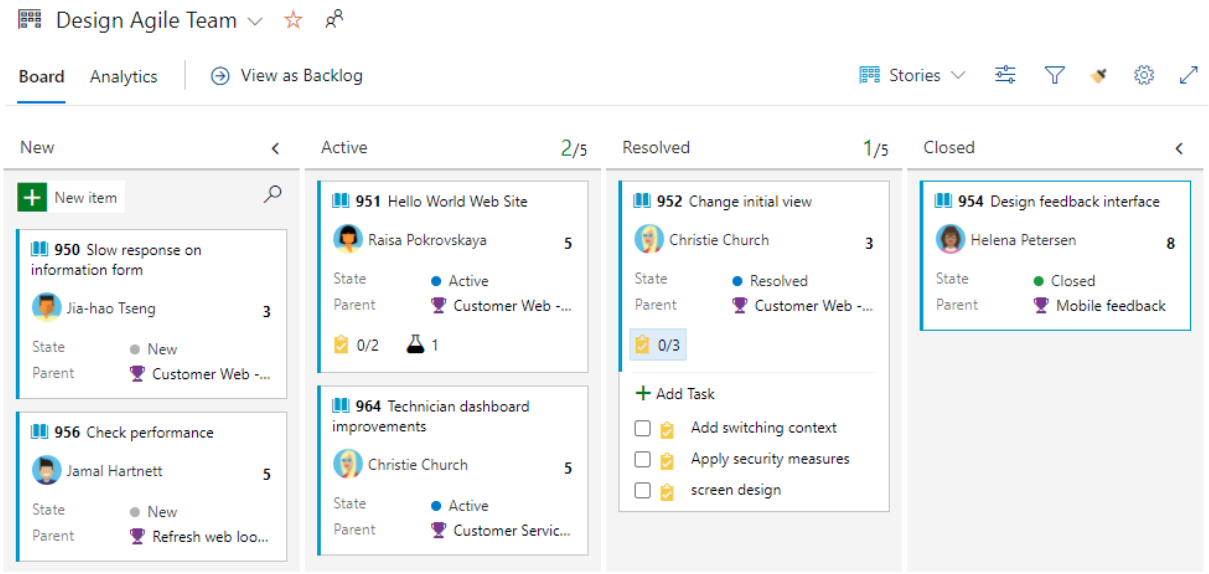
Figure 13 - Onglet Work Item Azure Board



Source : (Kathryn, et al., 2022)

L'onglet **Boards** permet d'afficher les éléments de travail en tant que cartes ainsi que d'effectuer des mises à jour d'état rapidement via un glisser-déplacer.

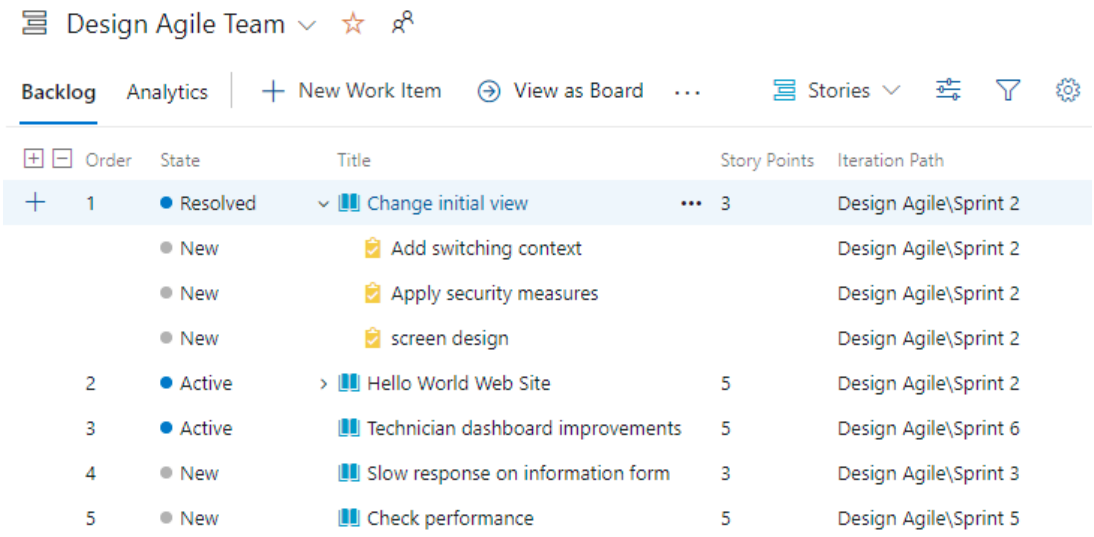
Figure 14 - Onglet Boards Azure Board



Source : (Kathryn, et al., 2022)

L'onglet **Backlogs** permet d'afficher, planifier, commander et organiser des éléments de travail (Kathryn, et al., 2022).

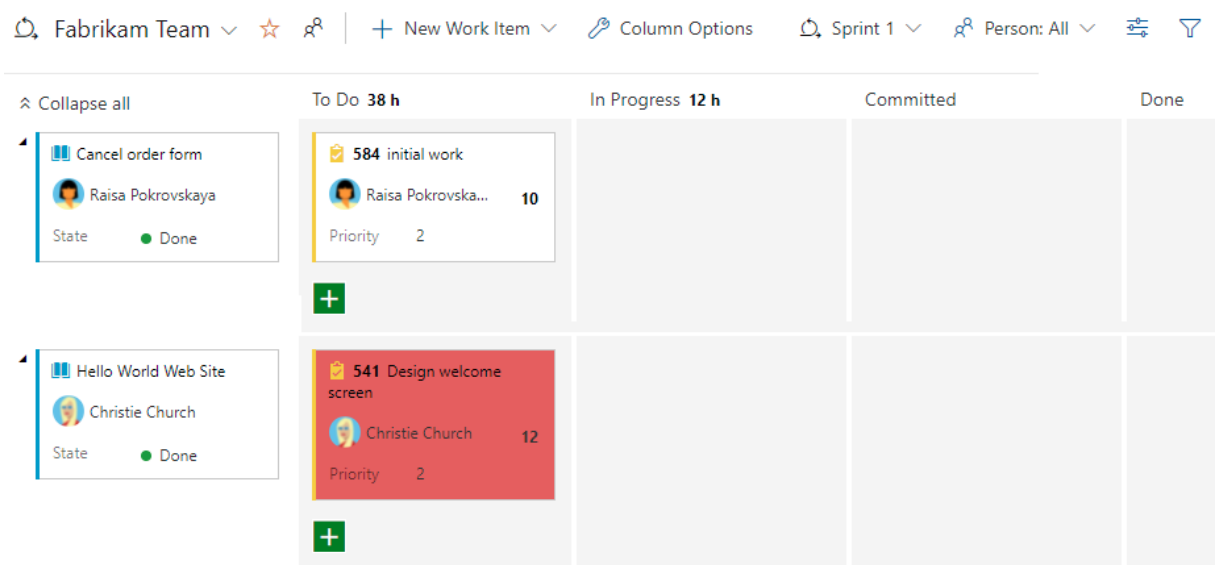
Figure 15 - Onglet Backlogs Azure Board



Source : (Kathryn, et al., 2022)

L'onglet **Sprints** permet d'afficher les éléments de travail en fonction d'un sprint et d'une équipe et affecter le travail en faisant un glisser-déplacer (Kathryn, et al., 2022).

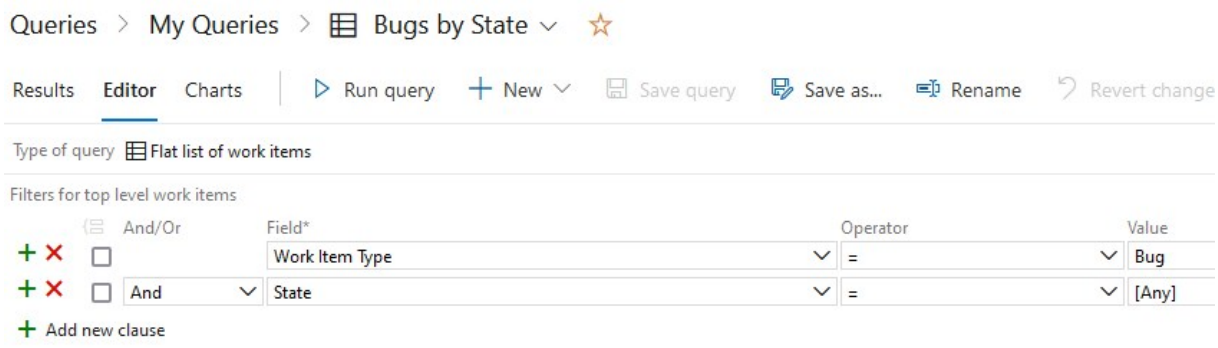
Figure 16 - Onglet Sprints Azure Board



Source : (Kathryn, et al., 2022)

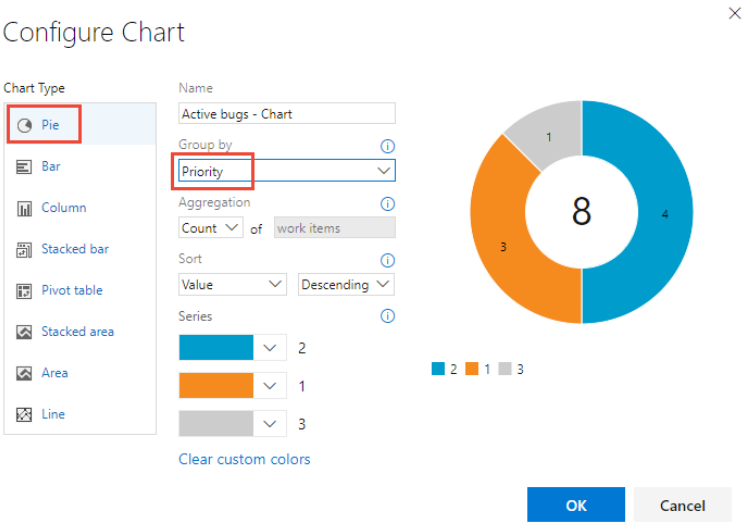
L'onglet **Queries** permet de sélectionner et d'afficher sous forme de listes et graphiques un ensemble de données spécifiques à partir du projet. Cela peut être utilisé pour afficher des données de suivi de travail telles que des tâches, des bugs mais également pour afficher des données de projet de build et de release. Les requêtes peuvent être enregistrées et partagées pour faciliter le suivi et la gestion du travail.

Figure 17 - Onglets Queries Azure Board : Exemple d'état des Bug



Source : Données de l'auteur

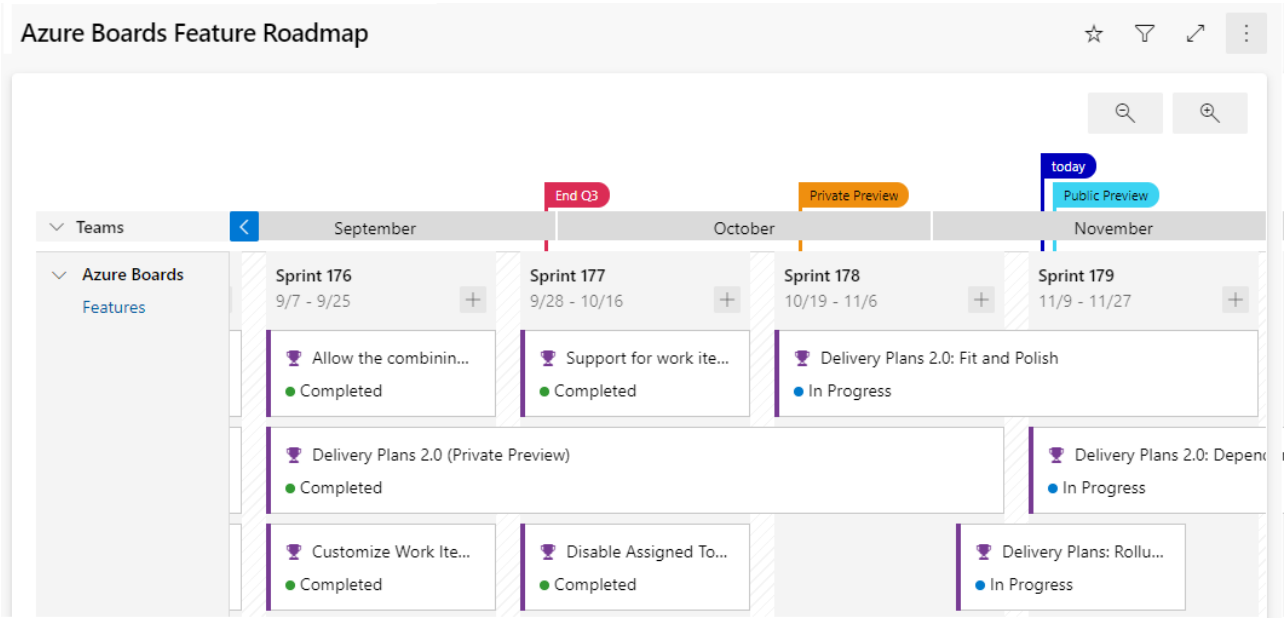
Figure 18 - Onglets Queries Azure Board : Graphique de requête



Source : (Kathryn, et al., 2022)

L'onglet **Delivery Plans** offre une vue d'ensemble du travail en cours et des objectifs à atteindre, ainsi que des indicateurs de progression et de performance de l'équipe.

Figure 19 - Exemple de Azure Board : Exemple de plan de délivrable



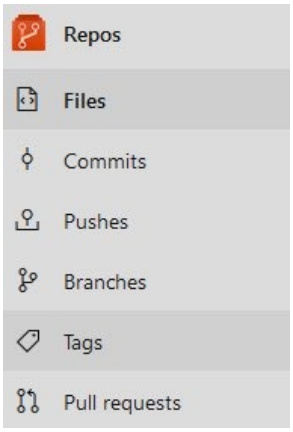
Source : (Kathryn, et al., 2022)

Etant donné la liberté dans le choix de la méthodologie, nous avons estimé que la méthode agile serait la plus complète et adaptable à la grandeur du projet storagemanager.

2.5.2. Azure Repos

Azure Repos est un service de gestion de code. Au démarrage d'un projet, les équipes doivent choisir quel type de dépôt central va être utilisé pour stocker le code.

Figure 20 - Fonctionnalités Azure Repos



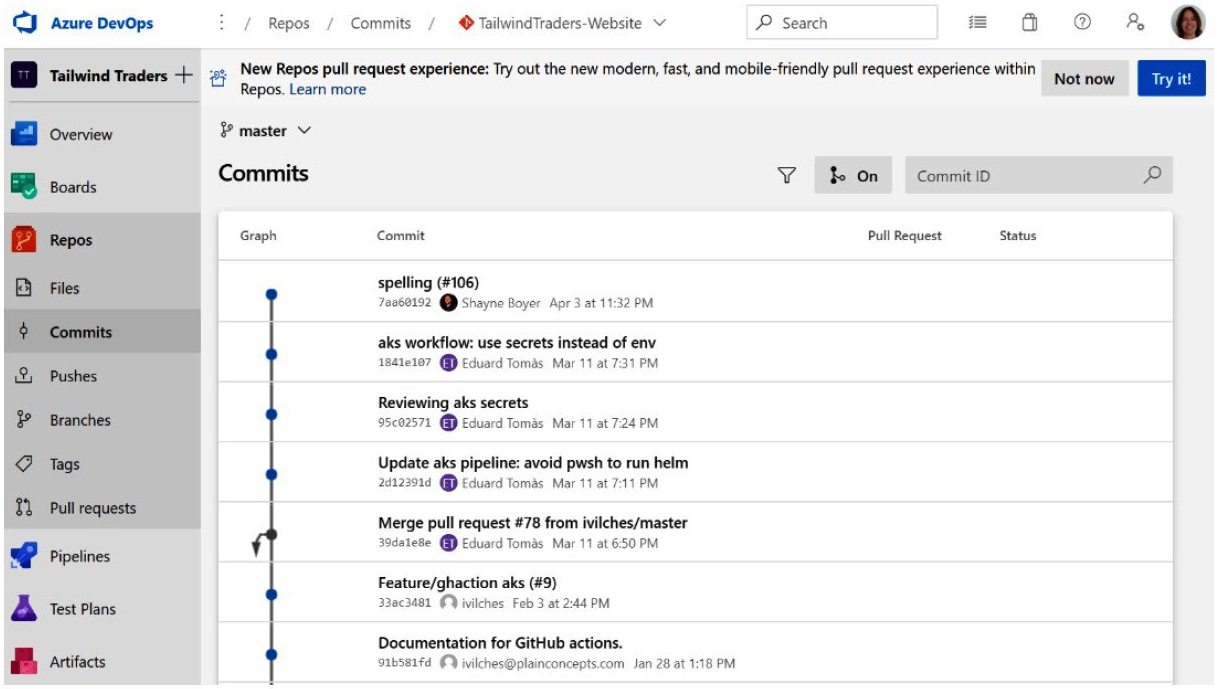
Source : Données de l'auteur

Azure Repos prend en charge les référentiels Git[13] et le contrôle du Team Foundation Version Control (TFVC) pour le contrôle de code source. Il offre un ensemble d'outils de contrôle de versions qui peuvent être utilisés pour gérer le code source de chaque projet de développement.

Les systèmes de contrôle de version (VCS) sont des logiciels qui aident à suivre les modifications apportées dans le code au fil du temps. Lorsque l'on modifie le code, cela indique au système de contrôle de version de prendre un instantané des fichiers. Le système de gestion de version enregistre cette capture instantanée définitivement afin que l'on puisse le rappeler ultérieurement si nécessaire (Machiraju, et al., 2022). Cette pratique permet de maintenir une source unique de codes à travers différents développeurs travaillant sur un même projet logiciel.

[13] Git est un système de contrôle de version décentralisé. C'est un logiciel libre et gratuit.

Figure 21 - Exemple de commits sur Azure Repos



Source : (Zaal, Demiliani, & Malik, 2020, p. 14)

Choix VCS

Microsoft a défini Git comme système de contrôle de version par défaut pour les nouveaux projets. Git est un système distribué, autrement dit, chaque utilisateur a une copie complète des fichiers du dépôt. TFVC est quant à lui un système centralisé qui stocke toutes les versions des fichiers dans un dépôt central sur un serveur unique.

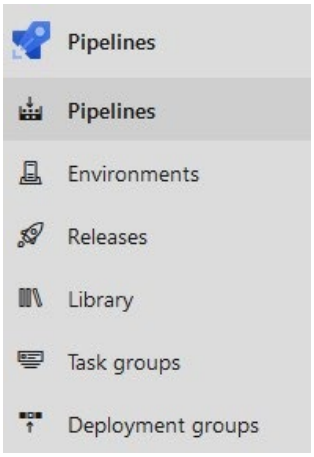
Selon la documentation, « Azure DevOps maintiendra la compatibilité avec TFVC, mais Git recevra tous les investissements futurs » (Maxim, et al., 2022). De plus, Git est devenu une référence pour la plupart des développeurs et administrateurs de bases de données.

Par conséquent il est plus judicieux pour nous d'utiliser Git.

2.5.3. Azure Pipelines

Azure Pipelines permet de créer, tester et déployer automatiquement des projets de code pour les mettre à la disposition des autres. Il fonctionne avec n'importe quel langage ou type de projet. C'est l'outil le plus important de Azure DevOps (Julia, et al., 2022).

Figure 22 - Fonctionnalité Azure Pipeline



Source : Données de l'auteur

Azure Pipelines combine l'intégration continue (CI) et le déploiement continu (CD) afin d'exécuter les unit tests, et générer le code et de l'expédier vers n'importe quelle cible tel que des machines virtuelles, des conteneurs, des environnements, des plates-formes on-premises[14] et cloud (Julia, et al., 2022).

Un pipeline peut être défini depuis l'éditeur classique ou en utilisant la syntaxe YAML.

YAML[15] est un langage de sérialisation des données souvent utilisé pour coder des fichiers de configuration. Le YAML est basé sur l'indentation pour définir la structure des données, il utilise des symboles tels que des tirets, des points et des accolades pour définir différents éléments de données.

La définition du pipeline YAML est stockée dans un fichier YAML (.yml) dans le code source du projet, cela signifie qu'elle est versionnée et gérée avec le reste du code. Cela permet une meilleure visibilité sur les modifications apportées au pipeline au fil du temps. Contrairement au pipeline classique qui n'est pas intégré au code source et n'est pas versionné.

Pour générer le code et déployer le logiciel, il est nécessaire d'avoir au moins un agent. Un agent est une infrastructure informatique avec un logiciel agent installé qui exécute une tâche à la fois (Steve, et al., 2022).

Deux possibilités s'offrent alors : un agent hébergé par Microsoft ou un agent auto-hébergé.

[14] On-premise : sur site

[15] YAML est l'acronyme de YAML Ain't Markup Language (« YAML n'est pas un langage de balisage »)

Travaux hébergés par Microsoft

La maintenance et les mises à niveau sont prises en charge par Microsoft. Chaque fois qu'un pipeline est exécuté, une nouvelle machine virtuelle est créée pour chaque tâche du pipeline. La machine virtuelle est supprimée après une tâche (ce qui signifie que toute modification apportée par une tâche au système de fichiers de la machine virtuelle ne sera pas disponible pour la tâche suivante). Les agents hébergés par Microsoft peuvent exécuter des tâches directement sur la machine virtuelle ou dans un conteneur (Steve, et al., 2022).

Facturation

Dans le cadre d'un projet privé, Azure pipeline ne peut exécuter qu'une seule tâche gratuitement pouvant durer jusqu'à 60 minutes à chaque fois pour une limite maximale de 1800 minutes (30 heures) par mois.

Dans un projet public, Azure pipeline peut exécuter jusqu'à 10 tâches parallèles gratuitement et jusqu'à 360 minutes (6 heures) à chaque fois sans limite de temps global par mois.

Il est malgré tout nécessaire de faire une demande explicite (voir Annexe II) afin de pouvoir bénéficier de la gratuité proposée.

Tableau 1 - Pipeline hébergé par Microsoft

	Nombre de travaux parallèles	**Limite de temps**
Projet public	Jusqu'à 10 travaux parallèles gratuits hébergés par Microsoft qui peuvent s'exécuter jusqu'à 360 minutes (6 heures) à chaque fois	Aucune limite de temps globale par mois
Projet privé	Un travail gratuit qui peut s'exécuter jusqu'à 60 minutes à chaque foi	1 800 minutes (30 heures) par mois

Source : (Configurer et payer pour les travaux parallèles, 2022)

Travaux auto-hébergés

Un agent auto-hébergé est un agent qu'il faut configurer et gérer soi-même pour exécuter des travaux, en d'autres termes, Azure Pipeline orchestre les build et les versions mais utilisera les machines personnelles pour les exécuter. Cela laisse plus de contrôle pour installer des logiciels dépendants nécessaires aux build et déploiements. Les principaux inconvénients sont la maintenance et l'évolutivité. Il faut par exemple s'assurer que l'agent ne remplit pas le stockage sur le disque (Steve, et al., 2022).

Facturation

La facturation dépendra du nombre de tâches exécutées en parallèles et ce sans aucune limite de temps.

Tableau 2 - Pipeline auto hébergé

	Nombre de travaux parallèles	Limite de temps
Projet public	Illimité	None
Projet privé	Un travail auto-hébergé ; Pour chaque abonné actif Visual Studio Enterprise membre de votre organisation, vous obtenez un travail parallèle auto-hébergé supplémentaire.	None

Source : (Configurer et payer pour les travaux parallèles, 2022)

Dépassé cette durée, CHF 38,56 est facturé par travail supplémentaire CI/CD hébergé par Microsoft ou CHF 14,46 par travail parallèle supplémentaire CI/CD auto-hébergé avec un nombre illimité de minutes (Tarification Azure DevOps, 2022).

Choix

Dans notre choix d'hébergement pour le proof of concept storagemanager, l'utilisation d'un pipeline YAML hébergé par Microsoft a été privilégié car notre pipeline ne dépassera pas 60 minutes et a besoin d'un agent unique pour fonctionner.

2.5.3.1. Environnement

Un environnement est une collection de ressources, couramment nommé Dev, Test, QA, Production, sur lequel on peut déployer une application à partir d'un pipeline (Julia, et al., 2022). Ils peuvent être utilisés pour représenter tout type d'environnement de déploiement.

Les environnements permettent de garder un historique de déploiement de pipeline, une traçabilité des validations et des éléments de travail, de vérifier l'intégrité des ressources et de définir des autorisations d'accès aux environnements. Cela permet de savoir quelle fonctionnalité ou correctif de bug a été déployée (Chandrasekara & Herath, 2020, p. 23).

Il existe trois niveaux d'autorisation différents pour contrôler l'accès à l'administration de l'environnement: Creator, Reader et User. Creator permet de tout faire, Reader permet de voir et User permet de créer des environnements. En dehors de ces autorisations, l'environnement ne peut être utilisé qu'avec les pipelines autorisés.

Deux types d'environnement sont possibles : les ressources de kubernetes[16] et de machine virtuelle.

Lors de la création d'un environnement dans le cas d'une machine virtuelle, un script PowerShell est généré contenant un token valable pendant trois heures. Ce script est à exécuter sur une machine virtuelle. Dans le cas d'un environnement Kubernetes, il est nécessaire de posséder des ressources Kubernetes.

2.5.3.2.Release

Une release est une construction qui contient un ensemble versionné d'artefacts spécifiés dans un pipeline CI/CD. Il comprend un instantané de toutes les informations requises pour effectuer toutes les tâches et actions dans le pipeline de publication, telles que les étapes, les tâches, les stratégies telles que les déclencheurs et les approbateurs, et les options de déploiement. (Nair, et al., Versions dans Azure Pipelines, 2022).

Les releases peuvent être créés manuellement ou via un déclencheur de déploiement. Lorsque les conditions du déclencheur sont remplies, le pipeline est ainsi déployé.

Il existe différents types de déclencheur :

- Les déclencheurs de déploiement continu permettent de créer une mise en production chaque fois qu'un nouvel artefact de build est disponible
- Les déclencheurs planifiés permettent de programmer les jours et heures des déploiements
- Les déclencheurs par pull request créent une mise en production chaque fois qu'une nouvelle version d'artefact est disponible
- Les déclencheurs d'étape permettent de configurer des conditions spécifiques pour un déploiement à une étape spécifique

Lorsque la release est lancée, cela démarre chaque déploiement prévu dans les paramètres du pipeline d'origine (Nair, et al., 2022).

[16] Kubernetes (parfois appelé K8s) est une plateforme open source populaire qui orchestre les systèmes d'exécution de conteneurs sur un cluster de ressources en réseau. Kubernetes peut être utilisé avec ou sans Docker.

2.5.3.3. Library

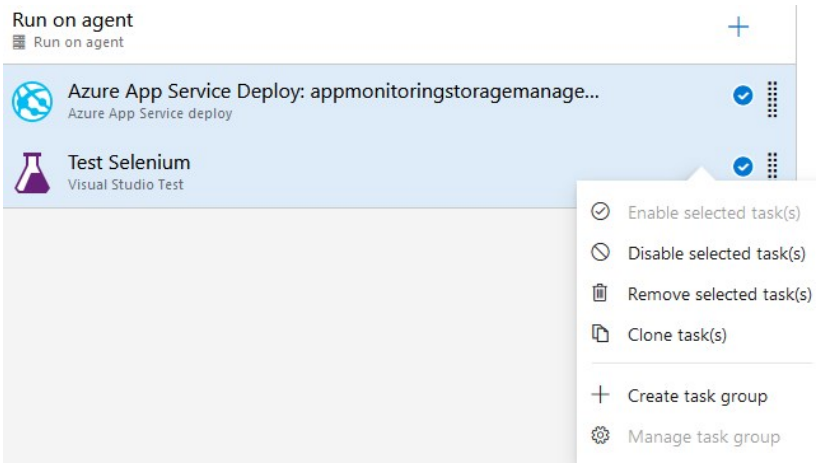
Les library (bibliothèque en français) est une collection de ressource de build et de mise en production. Ces ressources peuvent être utilisées dans plusieurs pipelines. La bibliothèque contient deux types de ressources : les groupes de variables et les fichiers sécurisés.

Les groupes de variables permettent de stocker des valeurs et des secrets pour les rendre disponibles sur plusieurs pipelines. Les fichiers sécurisés permettent de stocker des fichiers tels que des certificats de signatures, des profils de provisionnement Apple, des fichiers keystore Android ou des clés SSH. Le contenu de ces fichiers est crypté et ne peut être utilisé qu'à partir d'une tâche (Vijay, Steve, Eric, Julia, & Diana, 2022).

2.5.3.4. Task groups

Selon la documentation (Nair, et al., 2022), un groupe de tâches permet d'encapsuler une séquence de tâches, déjà définie dans un build ou un pipeline de release, en une seule tâche réutilisable qui peut être ajouté à un pipeline de build ou de publication, comme n'importe quelle autre tâche.

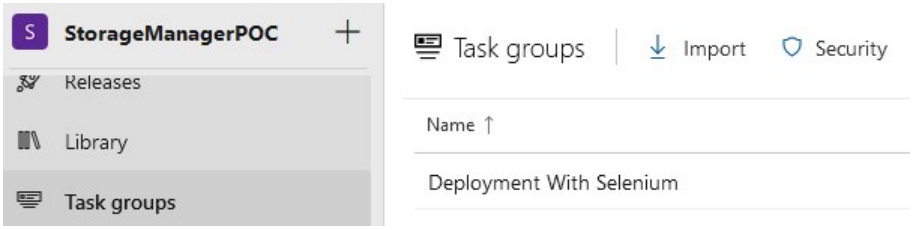
Figure 23 - Création d'un groupe de tâche



Source : (Nair, et al., 2022)

Tous les groupes de tâches créés dans le projet sont répertoriés dans l'onglet *Tasks Groups* :

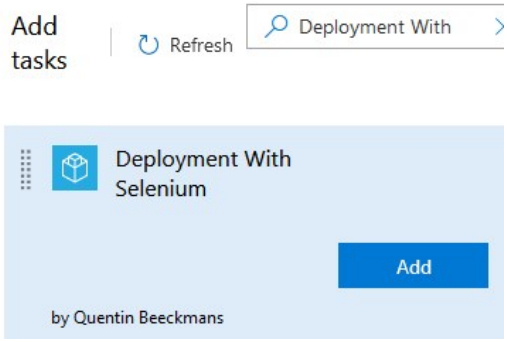
Figure 24 - Onglet Tasks groups Azure Pipeline



Source : Données de l'auteur

Le groupe de tâche peut être ajouté à un pipeline comme une tâche classique :

Figure 25 - Ajout d'un groupe de tâche



Source : Données de l'auteur

Les groupes de tâches ne sont pas pris en charge dans les pipelines YAML.

2.5.3.5. Deployment groups

Selon la documentation (Nair, et al., Provisionner des groupes de déploiement, 2022), un groupe de déploiement est un ensemble logique de machines cibles de déploiement sur lesquelles des agents sont installés. Les groupes de déploiement représentent les environnements physiques ; par exemple, environnement "Dev", "Test" ou "Production". En effet, un groupe de déploiement n'est qu'un autre regroupement d'agents, un peu comme un pool d'agents.

Lorsque l'on crée un groupe de déploiement, Azure génère un script pour Windows ou Linux. Ce script est à exécuter sur chaque machine cible du groupe de déploiement.

Les groupes de déploiement ne sont pas pris en charge dans les pipelines YAML.

2.5.4. Azure Test plans

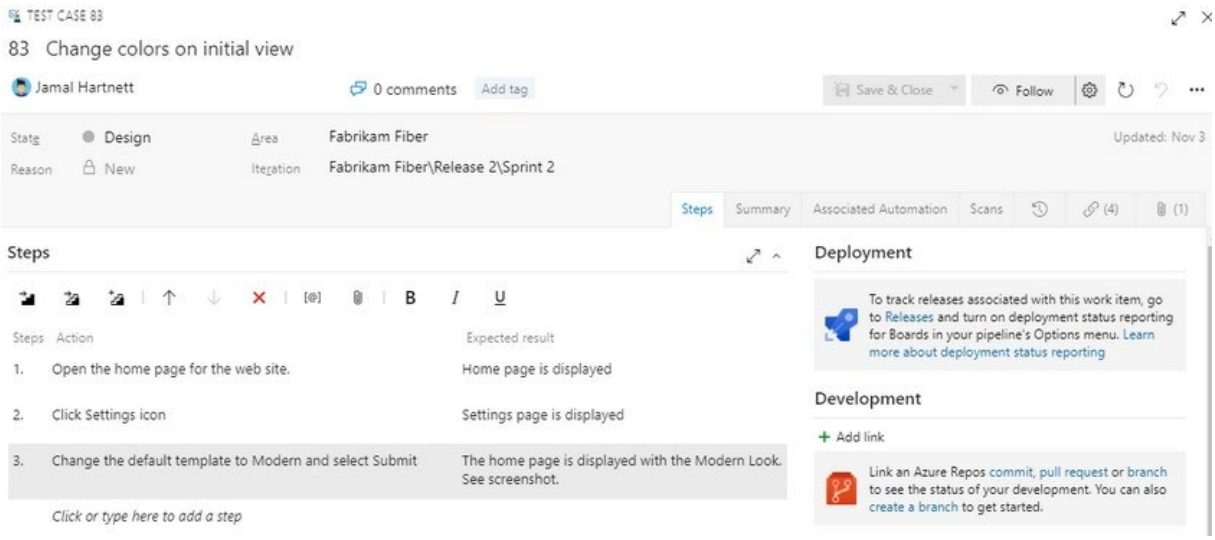
Azure Test Plan fourni des fonctionnalités pour les tests manuels planifiés, les tests exploratoires[17], les tests d'acceptation par les utilisateurs et sert également à recueillir les réactions des parties prenantes.

Les utilisateurs peuvent ajouter et exécuter des tests à partir du tableau Kanban ou directement dans le hub Plan de test pour définir des plans de test (conteneur dans lequel on regroupe des suites de tests) et des suites de tests (ensemble de cas de test).

Les testeurs peuvent directement créer une tâche de bug lorsqu'un test échoue.

Les résultats sont affichables via des graphiques configurables et peuvent être ajouté au tableau de bord.

Figure 26 - Exemple d'un Azure Test Plan



Source : (Kathryn, et al., 2022)

Comme vu précédemment, Azure Test Plan ne fait pas partie de la version Azure DevOps de base. Il y a une période d'essai de 30 jours, au-delà de laquelle il sera facturé CHF 50,13 par utilisateur et par mois.

Choix :

Azure Test Plan offre une multitude de tests d'acceptation de façon intégrée dans Azure DevOps. La taille de notre projet actuelle ne justifie pas l'utilisation de cet outil coûteux à ce niveau. A des fins de test nous allons l'implémenter durant la période d'essai offerte.

[17] Test exploratoire : est une approche de test logiciel mêlant l'apprentissage, le design et l'exécution de tests

2.5.5. Azure Artefacts

Azure Artefacts permet aux développeurs de partager efficacement leur code et de gérer tous leurs packages à partir d'un seul endroit. Les développeurs peuvent publier des packages dans leurs flux et les partager au sein de la même équipe, entre les organisations et même publiquement. Ils peuvent également consommer des packages provenant de différents flux et registres publics (Bououni, et al., 2022). Azure Artifacts prend en charge plusieurs types de packages tels que NuGet, npm, Python, Maven et les packages universels[18].

Au niveau de la tarification, Azure Artefact propose deux Gio[19] gratuits puis un tarif dégressif comme sur l'image qui suit :

Figure 27 - Carte de tarifs Azure Artefacts

Source : (Tarification Azure DevOps, 2023)

2.5.6. Visual Studio marketplace

Visual Studio marketplace permet de télécharger des extensions pour Azure DevOps. Ces modules complémentaires peuvent être gratuits ou payants et permettent de personnaliser l'utilisation de Azure DevOps. Elles peuvent notamment étendre la planification et le suivi des éléments de travail, des tests et du suivi du code, des flux de construction et de publication du pipeline ainsi que la collaboration entre les membres de l'équipe (Zaal, Demiliani, & Malik, 2020, p. 17).

Lien d'accès au marketplace : https://marketplace.visualstudio.com/

[18] Un package universel est une collection de fichiers qui ne correspond pas parfaitement aux autres types de package pris en charge.

[19] Gio : gibioctet, unité de mesure de quantité, 1Gio = 1 073 741 824 octets à ne pas confondre avec 1Go= 1000 000 000 octets.

3. ANALYSE DES COMPOSANTS DU DEVSECOPS POUR AZURE DEVOPS

DevSecOps applique la sécurité de l'innovation par l'intégration des processus et outils de sécurité dans le processus de développement DevOps. Il existe une multitude de ressources permettant d'identifier les vulnérabilités existantes, comme par exemple l'Open Web Application Security Project (OWASP).

L'OWASP est une fondation à but non lucrative qui se consacre à l'amélioration de la sécurité logiciel. C'est la source pour les développeurs et les technologues pour sécuriser le Web.

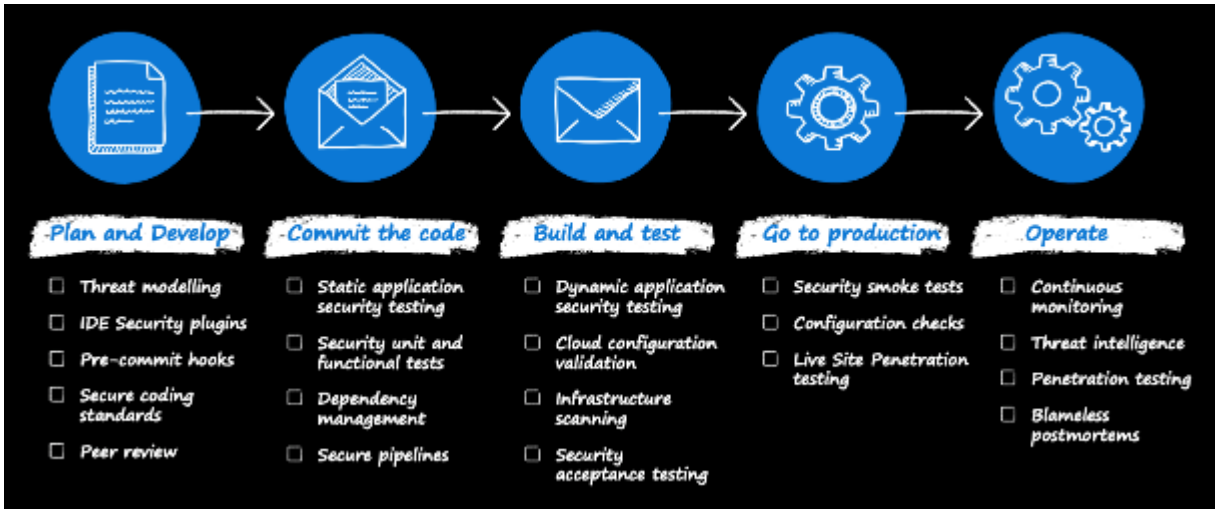
L'OWASP publie sur son site internet le Top 10 des risques de sécurité des applications web, cela regroupe les dix principaux risques qui peuvent être rencontrés et pour chacun d'entre eux, des conseils sur la façon d'y remédier.

Figure 28 - Top 10 OWASP 2021

Source : (OWASP Top Ten, 2021)

Un article de Microsoft (Buck, Liz, Gary, & Victoria, 2022) décrit les différentes étapes qui peuvent être suivies pour améliorer la sécurité dans un processus DevSecOps. Ces étapes permettent de détecter les problèmes de sécurité plus tôt dans le processus, ce qui rend leur résolution plus simple et moins coûteuse. Cependant, il est important de noter que toutes ces étapes ne sont pas nécessaires pour tous les projets, il faut donc sélectionner les étapes les plus appropriées en fonction des besoins spécifiques de chaque projet.

Figure 29 - Représentation des outils DevSecOps



Source : (Buck, Liz, Gary, & Victoria, 2022)

Les outils de test de sécurité des applications les plus populaires mis en œuvre par les entreprises dans leurs cycles de développement sont les tests de sécurité des applications statiques (SAST), l'analyse de la composition logicielle (SCA) et les tests de sécurité des applications dynamiques (DAST) (Simon, 2021).

L'intégration de ces extensions dans un pipeline Azure DevOps se fait via le marketplace de Visual Studio.

3.1. Pre-commit Hooks

Bien que cette étape ne soit pas utilisée sur la plateforme Azure, elle fait partie du processus de la méthodologie DevSecOps. Les precommit-Hooks sont la première ligne de défense contre les vulnérabilités. Ce sont des outils/plugins qui peuvent être intégrés dans l'IDE[20] d'un développeur et sont généralement utilisés pour trouver et résoudre rapidement les problèmes de code avant même qu'un développeur ne soumette le code dans le dépôt de contrôle de version. Ces outils peuvent analyser les fichiers soumis pour détecter la présence de clés secrètes, comme des jetons d'accès, des mots de passe ou des clés API[21], et bloquer la soumission en cas de détection. Cela permet de protéger les clés secrètes contre l'exposition accidentelle en les empêchant d'être stockées dans un dépôt public ou partagé avec des personnes non autorisées (Vivekanand, 2020).

[20] IDE (Integrated Development Environment) est un logiciel qui regroupe les outils nécessaires pour développer des logiciels.

[21] API (application programming interface) est une interface logicielle qui permet de connecter un logiciel ou un service à un autre logiciel ou service afin d'en échanger des données et des fonctionnalités.

3.2. Software Composition Analysis (SCA)

Selon Microsoft (Buck, Liz, Gary, & Victoria, 2022), jusqu'à 90% du code des applications actuelles contient des éléments de package et des bibliothèques externes. Ces dépendances présentes des problèmes de sécurité.

Les SCA permettent d'analyser le composant open source en détectant les licences logicielles, les dépendances obsolètes ainsi que les vulnérabilités connues et les code d'exploitation[22] potentiels dans une base de code afin de s'assurer de leur conformité. De cette façon, il est possible de trouver des défauts de sécurité plus tôt dans le cycle de développement. Ce qui permet d'arrêter le pipeline de build et de déploiement dès que des problèmes critiques sont détectés.

3.2.1. OWASP Dependency Check

OWASP Dependency Check est un utilitaire gratuit qui identifie les dépendances du projet et vérifie s'il existe des vulnérabilités connues et divulguées publiquement. Actuellement dans sa version 6.0.0.4 du 07.08.2019 et un note moyenne de 4.2/5 (6 votes) pour 5890 installations.

Il fonctionne avec Azure DevOps Services, Team Foundation Server (TFS) dès 2017 et Azure DevOps Server dès 2019 (OWASP Dependency Check, 2022).

3.2.2. Mend Bolt

Mend Bold, anciennement appelé WhiteSource est un outil gratuit qui scanne et détecte les composants open source, leur licence et les vulnérabilités connues tout en proposant des correctifs.

Il propose de gérer et sécuriser l'utilisation de l'open source, de recevoir des alertes des vulnérabilités en temps réel, d'assurer la conformité des licences et de fournir des rapports d'inventaire complet et précis basé sur l'utilisation de l'open source et ce à chaque build.

Actuellement dans sa version 22.7.1 du 24.07.2022 et un note moyenne de 3.6/5 (37 votes) pour 11472 installations.

Il fonctionne uniquement sur le service cloud de Azure DevOps (Mend Bolt (formerly WhiteSource), 2022).

[22] Un exploit ou code d'exploitation est, dans le domaine de la sécurité informatique, un élément de programme permettant à un individu ou à un logiciel malveillant d'exploiter une faille de sécurité informatique dans un système informatique.

3.2.3. Veracode

Veracode est un outil permettant de trouver et de corriger les défauts de sécurité dans le pipeline de Azure DevOps en analysant la composition du logiciel. Cet outil semblait gratuit dans son descriptif sur marketplace mais lors de son implémentation dans la phase de déploiement, il s'est avéré nécessaire de payer une licence pour l'utiliser.

Actuellement dans sa version 3.14.0 du 28.08.2022 et un note moyenne de 3.8/5 (14 votes) pour 8802 installation.

Il fonctionne sur le service cloud et sur site de Azure DevOps (Veracode, 2022).

3.2.4. Tableau de Comparaison

Tableau 3 - Comparaison des outils SCA

	OWASP Dependency Check (a)	Mend Bolt (b)	Veracode (c)
Evaluation	4,2/5 (6 votes)	3,6/5 (37 votes)	3.8/5 (14 votes)
Installations	5 891	11,472	8 802
Coût	Gratuit	Gratuit	Gratuit Payant
Hébergement	TFS/On-premise/Cloud	Cloud	On-premise/Cloud
Dernière mise à jour	18.05.2021	24.07.2022	28.08.2022

Source : Tableau de l'auteur provenant de sources multiples

a. (OWASP Dependency Check, 2022)
b. (Mend Bolt (formerly WhiteSource), 2022)
c. (Veracode, 2022)

3.2.5. Choix et justification

Dans le choix, l'hébergement hybride a été retenu afin de laisser plus de latitude au projet. OWASP Dependency Check et Veracode proposent des services similaires au niveau des technologies supportées et de leurs simplicités d'installation. Veracode avait été retenu dans un premier temps du fait de sa version plus récemment mis à jour et d'un suivi à jour des questions posées par les utilisateurs. Mais à la vue de la complexité de la mise en place de celui-ci et sa non gratuité, OWASP Dependency Check est l'outil retenu.

3.3. Static Application Security Testing (SAST)

Les outils de test de sécurité des applications statiques (SAST) sont utilisés pour détecter les vulnérabilités de sécurité dans le code source de l'application avant même qu'il ne soit exécuté. Ils s'intègrent dans le processus de développement pour détecter les faiblesses de sécurité dès le début, garantissant ainsi une meilleure sécurité de l'application. Ces outils sont utilisés dans l'intégration continue pour s'assurer que les vulnérabilités sont détectées le plus tôt possible (Buck, Liz, Gary, & Victoria, 2022).

3.3.1. SonarQube

SonarQube est un outil gratuit dans son édition Community et développé par SonarSource. Il permet de détecter les bug, les vulnérabilités et les smell code[23] dans les branches du projet et les pull request.

Actuellement dans sa version 5.8.1 pour Azure du 11.10.2022 et a un note moyenne de 2,9/5 (pour 75 votes) pour 93365 installations.

Il fonctionne avec Azure DevOps Services, Team Foundation Server (TFS) dès 2017 et Azure DevOps Server dès 2019 (SonarQube, 2022).

L'édition Community de SonarQube permet les fonctionnalités (Télécharger SonarQube, 2022) :

- Analyse de code statique pour 17 langues : Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML et VB.NET
- Détecter les bugs et les vulnérabilités
- Examiner les points d'accès de sécurité
- Suivre les smell code et corriger la dette technique[24]
- Métriques et historique de la qualité du code
- CI/CD intégration
- Extensible, avec plus de 50 plugins communautaires

[23] Un code smell (odeur de code en français) est une expression utilisée en développement logiciel pour décrire un aspect du code qui peut indiquer qu'il y a un problème de conception ou de qualité. Ce ne sont pas des erreurs à proprement parlé mais des indicateurs qui peuvent indiquer des problèmes potentiels à venir.

[24] Une dette technique est un terme utilisé pour décrire le montant de travail nécessaire pour maintenir et mettre à jour un système informatique ou un projet de développement logiciel.

L'édition Developer de SonarQube permet en plus de la version Community, de (Télécharger SonarQube, 2022):

- Travailler avec d'autres langages de programmation tel que : C, C++, Obj-C, Swift, ABAP, T-SQL, PL/SQL support.
- De détecter les failles d'injection en Java, C#, PHP, Python, JavaScript, TypeScript.
- D'analyser des branches de fonctionnalité et de maintenance. Un résumé des résultats apparait également dans l'interface des plateformes DevOps tels que GitHub, Bitbucket, Azure DevOps, GitLab.

La licence Developer est tarifé par instance[25] et par an et basé sur le nombre de lignes de code (Détails des prix, 2022).

Tableau 4 - Tableau des prix SonarQube Developer Edition par lignes de code

Jusqu'à lignes de code	Prix par an en $
100,000	$150
250,000	$1,200
500,000	$2,400
1 million	$4,000
2 millions	$8,000
5 millions	$23,000
10 millions	$48,000
20 millions	$65,000

Source : (Détails des prix, 2022)

3.3.2. SonarCloud

SonarCloud est développé par SonarSource, il est le principal service en ligne de SonarQube. Il est totalement gratuit pour les projets open source et prend en charge tous les principaux langages de programmation, notamment C#, VB .Net, JavaScript, TypeScript, C/C++.

Actuellement dans sa version 1.34.2 pour Azure du 14.11.2022 et a un note moyenne de 3.6/5 (pour 12 votes) pour 53003 installations.

Il fonctionne uniquement avec Azure DevOps Services.

[25] Instance : installation de SonarQube

SonarCloud propose un plan payant pour exécuter des analyses privées en fonction du nombre de lignes de code (LOC) de chaque projet analysé. Le nombre est seulement lié au nombre réel de lignes de code et n'est pas influencé par la fréquence d'analyse de notre projet. (SonarCloud, 2022).

La tarification SonarCloud commence à 10 €/mois pour un maximum de 100 000 LOC.

Figure 30 - Tableau des prix SonarCloud par lignes de code

Source : (Business solutions, 2022)

3.3.3. Checkmarx CxSAST

Checkmarx est un outil SAST permettant d'identifier, suivre et corriger des failles de sécurité techniques et logiques du code.

Actuellement dans sa version 2022.2.1 du 11.05.2022 et un note moyenne de 4,4/5 (pour 12 votes) pour 7045 installations.

Il fonctionne avec Azure DevOps Services et Azure DevOps Server (Checkmarx CxSAST, 2022).

3.3.4. Tableau de Comparaison

Tableau 5 - Comparaison des outils SAST

	SonarQube (a)	SonarCloud (b)	Checkmarx CxSAST (c)
Evaluation	2,9/5 (75 votes)	3.6/5 (12 votes)	4,4/5 (12 votes)
Installations	93492	53003	7057
Coût	Gratuit (pour les projets open source)	Gratuit (pour les projets public), payant (pour les projets privés)	Payant (14 jours gratuit)
Hébergement	On-premise/TFS/Cloud	Cloud	On-premise/Cloud
Dernière mise à jour	11.10.2022	14.11.2022	17.10.2022

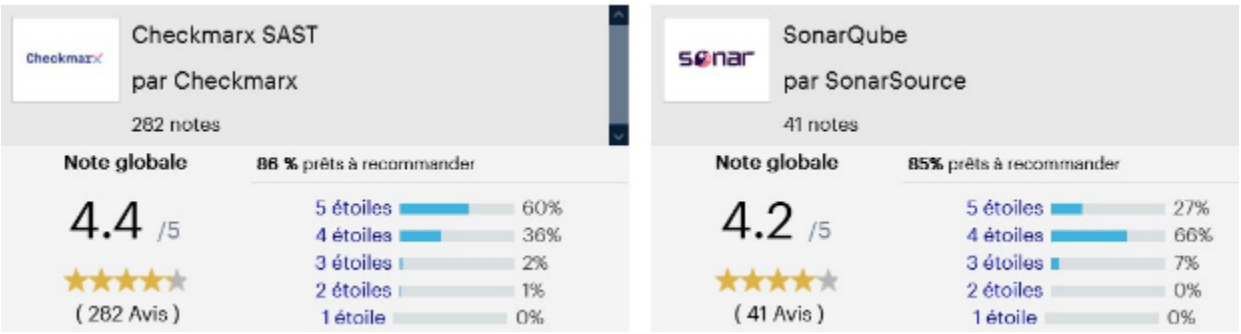
Source : Tableau de l'auteur provenant de sources multiples

a. (SonarQube, 2022)
b. (SonarCloud, 2022)
c. (Checkmarx CxSAST, 2022)

3.3.5. Choix et justification

Dans le choix, SonarQube et Checkmarx CxSAST proposent des services similaires au niveau des technologies supportées. SonarQube se montre plus polyvalent en revanche, en utilisant l'environnement Team Foundation Server 2017. Malgré cela, Checkmarx SAST est selon les données de classement pour 2022 de Gartner, l'outil d'analyse de code statique préféré de la communauté par rapport à son concurrent.

Figure 31 - Notes comparatives Gartner Checkmarx vs SonarQube



Source : (Checkmarx SAST contre SonarQube, 2022)

Il en ressort également que SonarQube est critiqué pour faire beaucoup plus de faux négatif que son concurrent (Checkmarx SAST contre SonarQube, 2022). Bien que présentant plusieurs avantages, Checkmarx manque de transparence au niveau du prix. Il est d'ailleurs critiqué par les utilisateurs pour son manque de convivialité et ses prix élevés.

Nous avions décidé de faire confiance à SonarQube pour notre projet pour sa grande communauté, son efficacité et sa transparence. Son implémentation nous a révélé qu'il était nécessaire de choisir SonarCloud pour le problème rencontré et analysé dans ce rapport. Ce système apporte de nombreux avantages comme de ne pas devoir mettre en place une machine virtuel tout en profitant de l'analyse SonarQube. De plus, SonarCloud obtient une note de huit sur dix selon les données de classement pour 2022 de *PeerSpot* (Avis sur Sonar Cloud, 2022).

3.4. Dynamic Application Security Testing (DAST)

Les outils de tests dynamiques de sécurité des applications (DAST) analysent les applications depuis l'extérieur (test de boîte noir), c'est-à-dire qu'ils explorent les pages web, localisent les entrées et les sorties sans regarder dans le code source.

Ces outils simulent des tests de pénétration comme des attaques pour découvrir des vulnérabilités exploitables et des problèmes de logique métier[26] du point de vue d'un pirate (Simon, 2021). Contrairement aux outils SAST, ils peuvent détecter les défauts d'exécution.

L'inconvénient de ces tests est qu'ils arrivent en fin de processus lorsque l'application est déjà en cours d'exécution, la découverte d'une vulnérabilité d'exécution oblige les développeurs à les corriger rapidement et entraine des coûts supplémentaires. Un autre inconvénient est que ces tests ne peuvent pas situer ces vulnérabilités dans le code car ils n'y ont pas accès, ils rapportent seulement les comportements non prévu de l'application.

3.4.1. OWASP ZAP Scanner

OWASP ZAP est un scanner d'application gratuit qui permet d'identifier les vulnérabilités au cours du processus de développement à partir des rapports de l'Open Web Application Security Project.

Il permet une analyse active sur une cible avec des seuils de risque de sécurité et la possibilité de générer le rapport d'analyse. Il est l'outil DAST le plus populaire du marché.

Actuellement dans sa version 1.0.4 du 25.10.2019 et un note moyenne de 3.2/5 (pour 13 votes) pour 3790 installations.

[26] Le terme "logique métier" désigne toutes les règles ou tests de validation personnalisés appliquées aux données avant de les insérer, de les mettre à jour ou de les supprimer de la base de données.

Il fonctionne uniquement avec Azure DevOps Services (OWASP ZAP Scanner, 2022).

3.4.2. Micro Focus Fortifier - Fortify WebInpect

Micro Focus Fortifier rassemble plusieurs outils d'analyse de sécurité. La solution Fortify WebInpect de Micro Focus est la solution DAST qui fournit une détection complète des vulnérabilités en simulant des attaques de sécurité externes réelles sur une application en cours d'exécution afin d'identifier les problèmes et de les classer par ordre de priorité pour une analyse des causes profondes (Microfocus, 2022).

Actuellement dans sa version 8.8.2 du 26.07.2022 et un note moyenne de 3.8/5 (pour 12 votes) pour 8481 installations.

Il fonctionne avec Azure DevOps Services et Azure DevOps Server (Micro Focus Fortify, 2022).

3.4.3. Tableau de Comparaison

Tableau 6 - Comparaison des outils DAST

	OWASP ZAP (a)	Fortify WebInpect (b)
Evaluation	3.2/5 (pour 4 votes)	3.8/5 (pour 12 votes)
Installations	3790	8481
Coût	Gratuit	Payant
Hébergement	Cloud	On-premise/Cloud
Dernière mise à jour	25.10.2019	26.07.2022

Source : Tableau de l'auteur provenant de sources multiples

a. (OWASP ZAP Scanner, 2022)
b. (Micro Focus Fortify, 2022)

3.4.4. Choix et justification

La préférence est donnée à OWASP ZAP qui a l'avantage d'être un outil soutenu par l'OWASP bien que sa dernière release ait été faite en 2019. WebInspect n'apporte pas une vision claire de son tarif bien qu'affiché comme gratuit dans son ajout dans un pipeline sur marketplace, celui-ci nécessite un compte sur leur plateforme.

3.5. Azure monitor

Azure Monitor est un service Azure qui permet de surveiller les applications et les infrastructures. Il identifie les problèmes de code et les activités et anomalies potentiellement suspectes.

Azure Monitor s'intègre aux pipelines de mise en production dans Azure Pipelines pour permettre l'approbation automatique des contrôles de qualité ou de la restauration de version en fonction des données de supervision (Tarification Azure Monitor, 2022).

Il existe deux types tarifications : paiement à l'utilisation ou par niveaux d'engagement.

Avec les niveaux d'engagement, des frais fixes et prévisibles sont facturés à partir de 100 Go par jour. Les données ingérées au-dessus du niveau d'engagement sont facturées au même prix par Go que le niveau actuel. Cela offre une remise sur l'ingestion de données en fonction du niveau d'engagement sélectionné. Les niveaux d'engagement ont une période d'engagement de 31 jours (Tarification Azure Monitor, 2022).

Tableau 7 - Tarification Azure Monitor

Niveau de tarification	Prix	Prix effectif par Go [1]	Économies par rapport au paiement à l'utilisation
Paiement à l'utilisation	CHF 4'197 par Go (5 Go par compte de facturation par mois inclus)	CHF 4'197 par Go	N / A
100 Go par jour	CHF 248.15 par jour	CHF 2.49 par Go	41%
200 Go par jour	CHF 465.92 par jour	CHF 2.33 par Go	44%

Source : (Tarification Azure Monitor, 2022)

3.6. Synthèse des choix des composants

	SCA	SAST	DAST
Nom :	Owasp Dependency Check	SonarCloud	OWASP ZAP Scanner
Sources :	(OWASP Dependency Check, 2022)	(SonarCloud, 2022)	(OWASP ZAP Scanner, 2022)

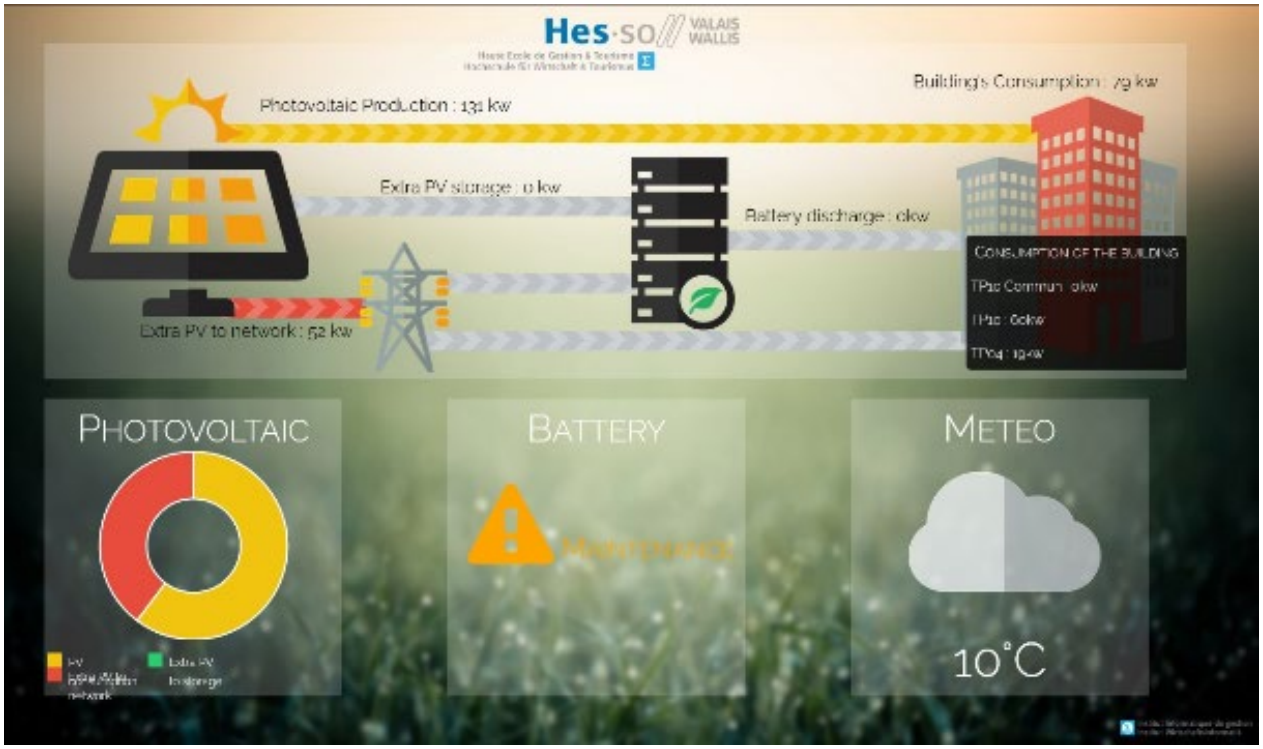
4. DEPLOIEMENT DU STORAGEMANAGER SUR AZURE SUIVANT UNE METHODOLOGIE DEVSECOPS

Dans cette partie du travail de Bachelor, nous allons intégrer les différents outils nécessaires afin de déployer le projet storagemanager sur la plateforme cloud Azure sous la forme d'un proof of concept afin de valider les recherches effectuées durant l'état de l'art.

4.1. Architecture existante

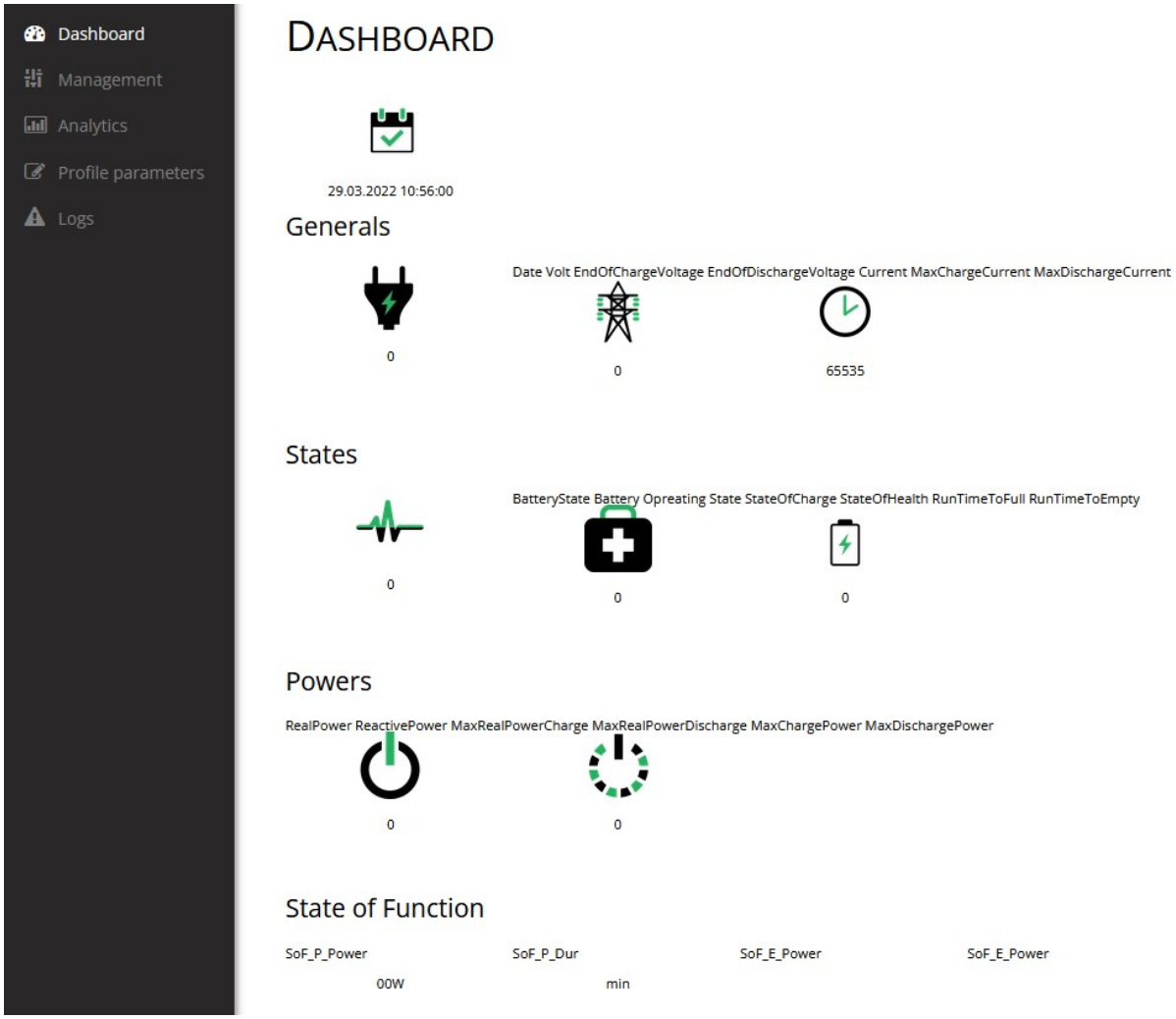
Il est important d'analyser l'architecture existante avant de la déployer dans le cloud, le projet en ligne actuellement se compose de deux pages web distincte présentant une page de moniteur (http://81.13.188.123:81/monitoring/home.aspx) et une page admin (http://81.13.188.123:81/AdminConsole/Login.aspx). La machine virtuelle du projet est hébergée au technopole de Sierre.

Figure 32 - Vue page web monitoring storagemanager



Source : (2022)

Figure 33 - Vue page web admin storagemanager



Source : (2022)

Les données affichées sur les pages web proviennent d'une base de données qui n'est plus mise à jour actuellement. Les données de la batterie datent du 31/12/2021 et les données météo datent du 31/08/2015. Dans le but de tester l'affichage du projet, il a été décidé de faire des mocks[27] pour simuler des données à afficher sur le monitoring.

[27] Les Mocks, en programmation, sont des objets simulés qui reproduisent le comportement d'objets réels de manière contrôlée.

Figure 34 - Exemple de mock de données 1/2

```
//Battery state
/* MOCK
LabelBatteryStateDate.Text = battery.Date;
LabelBatteryStateBatteryState.Text =  string.Format("<span id='batteryState'>{0}</span> ({1
LabelBatteryStateBatteryOperatingState.Text = string.Format("{0} ({1})", battery.BatteryOper
LabelBatteryStateVoltage.Text = battery.Volt.ToString();
LabelBatteryStateCurrent.Text = battery.Current.ToString();
LabelBatteryStateRealPower.Text = battery.RealPower.ToString();
LabelBatteryStateReactivePower.Text = battery.ReactivePower.ToString();
LabelBatteryStateMaxRealPowerCharge.Text = battery.MaxRealPowerCharge.ToString();
LabelBatteryStateMaxRealPowerDischarge.Text = battery.MaxRealPowerDischarge.ToString();
LabelBatteryStateEndOfChargeVoltage.Text = battery.EndOfChargeVoltage.ToString();
LabelBatteryStateMaxChargeCurrent.Text = battery.MaxChargeCurrent.ToString();
LabelBatteryStateEndOfDischargeVoltage.Text = battery.EndOfDischargeVoltage.ToString();
LabelBatteryStateMaxDischargeCurrent.Text = battery.MaxDischargeCurrent.ToString();
LabelBatteryStateMaxChargePower.Text = battery.MaxChargePower.ToString();
LabelBatteryStateMaxDischargePower.Text = battery.MaxDischargePower.ToString();
LabelBatteryStateRunTimeToFull.Text = battery.RunTimeToFull.ToString();
LabelBatteryStateRunTimeToEmpty.Text = battery.RunTimeToEmpty.ToString();
LabelBatteryStateStateOfCharge.Text = battery.StateOfCharge.ToString();
LabelBatteryStateStateOfHealth.Text = battery.StateOfHealth.ToString();
*/
```

Source : Source de l'auteur

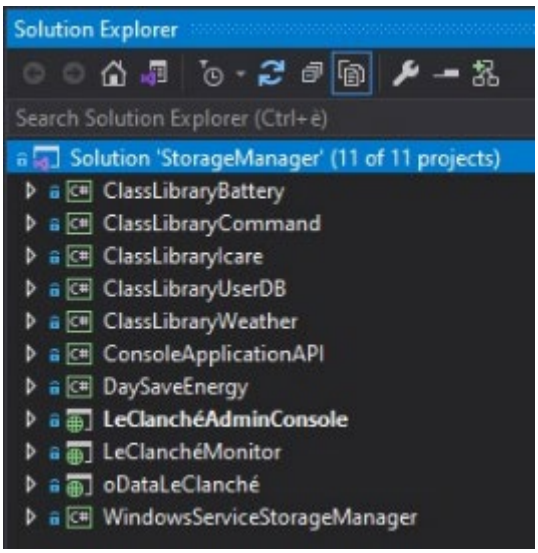
Figure 35 - Exemple de mock de données 2/2

```
LabelBatteryStateDate.Text = DateTime.Today.ToString();
LabelBatteryStateBatteryState.Text = string.Format("<span id='batteryState'>{0}</span> ({1})",
LabelBatteryStateBatteryOperatingState.Text = "3 (Idle)";
LabelBatteryStateVoltage.Text = "0";
LabelBatteryStateCurrent.Text = "0";
LabelBatteryStateRealPower.Text = "0";
LabelBatteryStateReactivePower.Text = "0";
LabelBatteryStateMaxRealPowerCharge.Text = "0";
LabelBatteryStateMaxRealPowerDischarge.Text = "0";
LabelBatteryStateEndOfChargeVoltage.Text = "0";
LabelBatteryStateMaxChargeCurrent.Text = "0";
LabelBatteryStateEndOfDischargeVoltage.Text = "0";
LabelBatteryStateMaxDischargeCurrent.Text = "0";
LabelBatteryStateMaxChargePower.Text = "0";
LabelBatteryStateMaxDischargePower.Text = "0";
LabelBatteryStateRunTimeToFull.Text = "0";
LabelBatteryStateRunTimeToEmpty.Text = "0";
LabelBatteryStateStateOfCharge.Text = "20";
LabelBatteryStateStateOfHealth.Text = "20";
```

Source : Source de l'auteur

La structure du projet est composée comme suite :

Figure 36 - Structure des fichiers du storagemanager



Sources : Données de l'auteur

4.2. Implémentation de la méthodologie Agile avec Azure Board

Azure Board permet entre autres d'importer et exporter des éléments de travail depuis un fichier au format CSV[28]. Il est également possible d'intégrer un outil à Excel pour l'importation (Kathryn, et al., Importer ou mettre à jour des éléments de travail en masse à l'aide de fichiers CSV dans Azure Boards, 2022).

Nous allons utiliser le format CSV qui ne nécessite pas de licence Office Excel.

La première ligne de notre CSV contient les définitions des champs que l'on va utiliser, une liste exhaustive se trouve à l'adresse : https://learn.microsoft.com/en-us/azure/devops/boards/work-items/guidance/work-item-field?view=azure-devops

Dans notre cas, nous avons utilisé :

```
ID,Work Item Type,Title,Assigned To,State,Area Path,Iteration Path,Tags.
```

- L'**ID** ne doit pas être rempli lors de l'ajout de nouveaux items, il n'est d'usage qu'en cas d'update.
- **Work Item** peut être de type : Bug, Epic, Feature, Product backlog item (Scrum)
- **Title** : est le titre qui sera affiché, il est préférable de ne pas utiliser des accents car ceux-ci ne sont pas reconnus lors de l'importation

[28] CSV (comma-separated values) est un format de texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.

- **Assigned To** : est le nom du membre de l'équipe qui possède actuellement l'élément de travail.
- **State** : Indique l'état de notre élément il peut être de type : New, Active, Resolved, Closed, Removed. Lors de la création d'un nouvel élément, seul le statut New est accepté.
- **Area Path** : regroupe les éléments de travail en fonctionnalités de produits ou en zones d'équipe. La zone doit être un nœud valide dans la hiérarchie du projet.
- **Item Path** : Regroupe les éléments de travail par sprints, l'itération doit être un nœud valide dans la hiérarchie du projet.
- **Tags** : le balisage aide à filtrer rapidement les éléments par catégorie lors de requêtes.

Notre fichier CSV rassemble ces champs pour les ajouter à notre itération 2 :

Exemple d'Users Stories au format CSV :

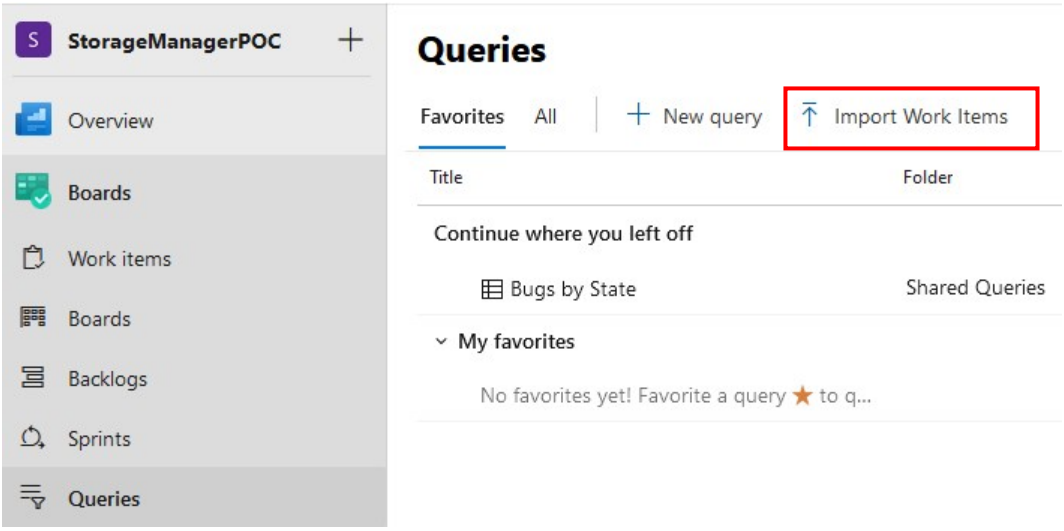
ID,Work Item Type,Title,Assigned To,State,Area Path,Iteration Path,Tags
,User Story,En tant que Développeur je veux intégrer intégrer un outil DAST afin d'intégrer la sécurité dans le pipeline,Quentin Beeckmans <quentin.beeckman@hes-so.ch>,New,StorageManagerPOC,StorageManagerPOC\Iteration 2,DAST
,User Story,En tant que Développeur je veux créer un pipeline de release afin de deployer l'application,Quentin Beeckmans <quentin.beeckman@hes-so.ch>,New,StorageManagerPOC,StorageManagerPOC\Iteration 2,CD
,User Story,En tant que Développeur je veux s'ouscrire à l'offre d'hébergement App Service afin d'en avoir les accès et de le paramétrer,Quentin Beeckmans <quentin.beeckman@hes-so.ch>,New,StorageManagerPOC,StorageManagerPOC\Iteration 2,AppService
,User Story,Test Iteration Path vide,Quentin Beeckmans <quentin.beeckman@hes-so.ch>,New,StorageManagerPOC,,AppService

Exemple de tâche au format CSV :

ID,Work Item Type,Title,Assigned To,State,Area Path,Iteration Path,Tags
,Task,Paramétrer l'app service,Quentin Beeckmans <quentin.beeckman@hes-so.ch>,New,StorageManagerPOC,StorageManagerPOC\Iteration 2,AppService

Pour importer le fichier csv dans Azure, allez dans *Boards* > *Queries* comme suit :

Figure 37 - Importer des Work Items CSV dans Azure Board



Source : Données de l'auteur

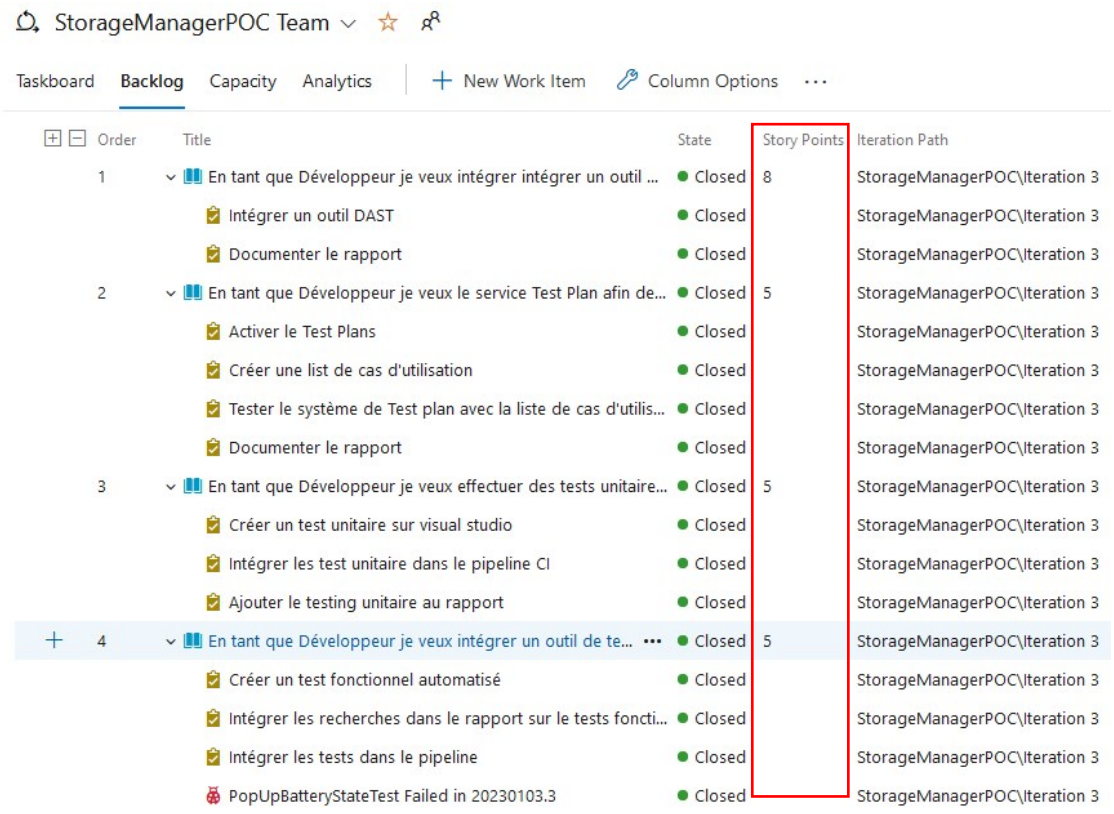
Le système analyse l'importation et met en évidence les erreurs en cas de problème. Par exemple, dans notre import, il manquait la valeur Item Path dans une User Story :

Figure 38 - Erreur d'importation User Story

Source : Données de l'auteur

Nous avons estimé l'effort nécessaire pour réaliser chaque User Stories par des story points. Cela nous permet également de faire une estimation de l'effort en heure pour chaque tâche.

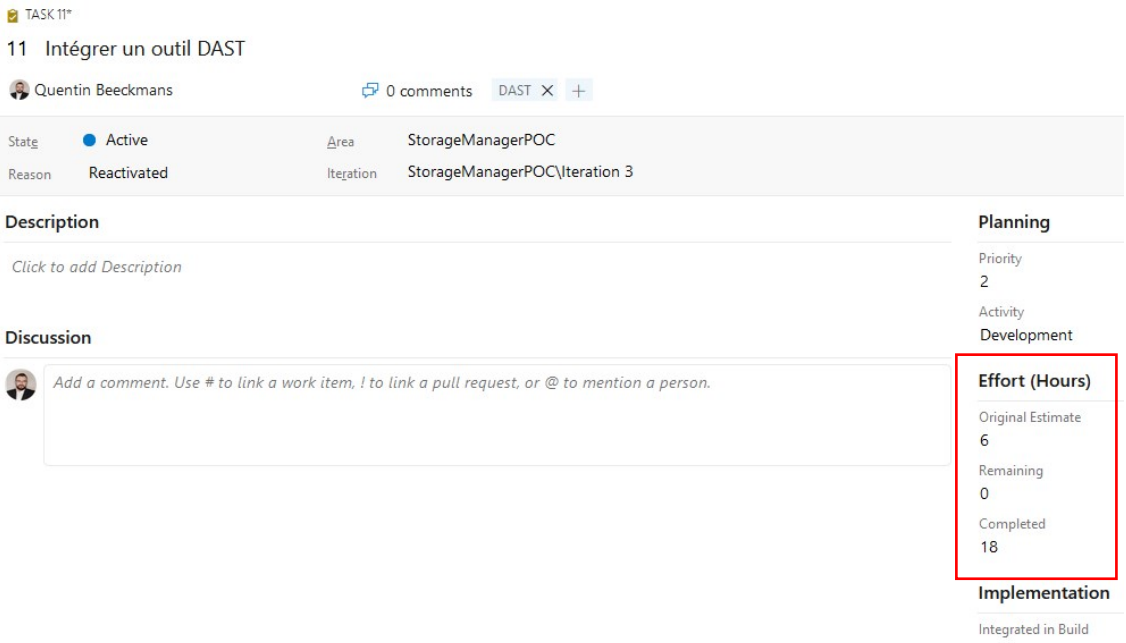
Figure 39 - Story Points du Sprint 3



Order	Title	State	Story Points	Iteration Path
1	En tant que Développeur je veux intégrer intégrer un outil ...	Closed	8	StorageManagerPOC\Iteration 3
	Intégrer un outil DAST	Closed		StorageManagerPOC\Iteration 3
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 3
2	En tant que Développeur je veux le service Test Plan afin de...	Closed	5	StorageManagerPOC\Iteration 3
	Activer le Test Plans	Closed		StorageManagerPOC\Iteration 3
	Créer une list de cas d'utilisation	Closed		StorageManagerPOC\Iteration 3
	Tester le système de Test plan avec la liste de cas d'utilis...	Closed		StorageManagerPOC\Iteration 3
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 3
3	En tant que Développeur je veux effectuer des tests unitaire...	Closed	5	StorageManagerPOC\Iteration 3
	Créer un test unitaire sur visual studio	Closed		StorageManagerPOC\Iteration 3
	Intégrer les test unitaire dans le pipeline CI	Closed		StorageManagerPOC\Iteration 3
	Ajouter le testing unitaire au rapport	Closed		StorageManagerPOC\Iteration 3
4	En tant que Développeur je veux intégrer un outil de te...	Closed	5	StorageManagerPOC\Iteration 3
	Créer un test fonctionnel automatisé	Closed		StorageManagerPOC\Iteration 3
	Intégrer les recherches dans le rapport sur le tests foncti...	Closed		StorageManagerPOC\Iteration 3
	Intégrer les tests dans le pipeline	Closed		StorageManagerPOC\Iteration 3
	PopUpBatteryStateTest Failed in 20230103.3	Closed		StorageManagerPOC\Iteration 3

Source : Données de l'auteur

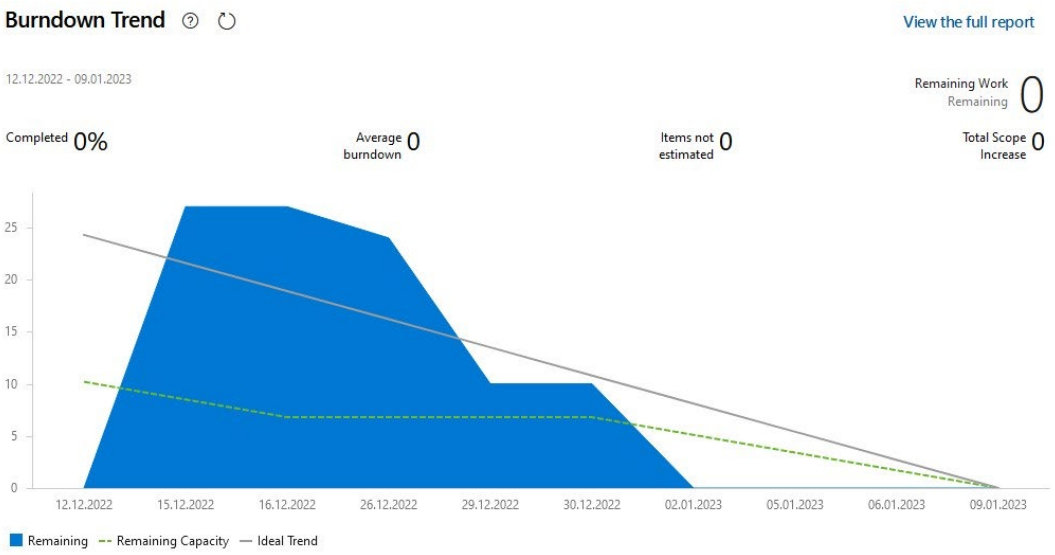
Figure 40 - Exemple d'estimation en heure de l'effort de réalisation d'une tâche



Source : Données de l'auteur

Le burndown chart est un graphique utilisé en conjonction avec les story points et l'estimation de l'effort des tâches restant pour suivre l'avancement d'un projet agile. Le graphique est divisé en deux axes : l'axe horizontal représente le temps (en jours de travail effectifs) et l'axe vertical représente l'effort restant.

Figure 41 - Burndown chart Azure Sprint 3



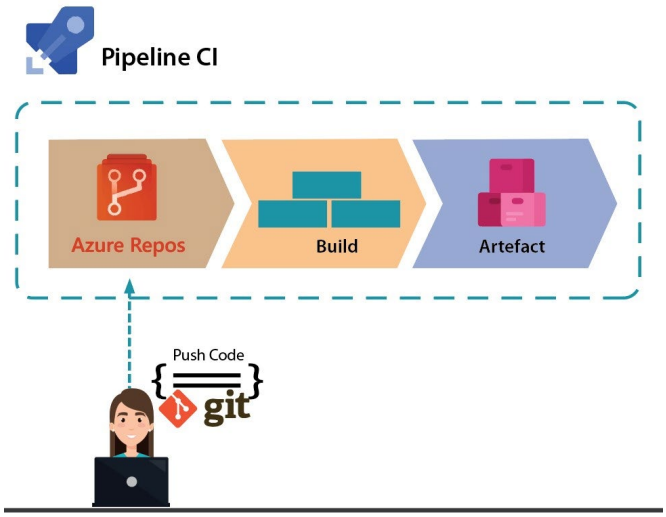
Source : Données de l'auteur

Burndown chart du projet (voir Annexe IV).

4.3. Implémentation des outils DevOps

4.3.1. Azure Pipelines (CI)

Figure 42 - Pipeline CI



Source : Données de l'auteur

Lorsqu'une mise à jour du code (git push) est effectuée sur la branche principale (main) du contrôle de version de code, le pipeline Azure est déclenché durant la nuit suivante[29], permettant de faire un build et de le tester. En cas d'échec, le pipeline est arrêté et une erreur est rapportée au développeur sur l'interface Azure DevOps et également par email. Ces notifications sont configurables dans :

Project setting > Notifications.

Le pipeline complet est configuré dans un fichier YAML comme suit :

```yaml
trigger:
- none

pool:
  vmImage: 'windows-2019'

variables:
  solution: '**\*.sln'
  project: '**\LeClanchéMonitor.csproj'
  buildPlatform: 'AnyCPU'
  buildConfiguration: 'Release'

steps:
- task: CmdLine@2
  displayName: Script to tag pipeline
  inputs:
    script: 'echo ##vso[build.addbuildtag]Test'

- task: NuGetToolInstaller@1
  inputs:
    versionSpec:
- task: NuGetCommand@2
  inputs:
```

[29] Nighty build(Construction nocturne) : la construction nocturne permet de s'assurer que les développeurs ne sont pas susceptibles de travailler sur le code source. Cela permet également de ne pas s'occuper du temps d'exécution plus long pour les tests de régression.

```
  restoreSolution: '$(solution)'

- task: MSBuild@1
  displayName: BUILD
  inputs:
    solution: '$(project)'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'
    msbuildArguments: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPackageLocation="$(build.artifactStagingDirectory)\WebApp.zip" /p:DeployIisAppPath="Default Web Site"'

- task: PublishBuildArtifacts@1
  displayName: Create Artefact
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

Analyse du pipeline

```
pool:
  vmImage: 'windows-2019'
```

Dans cette commande, nous spécifions un pool d'agents hébergé par Microsoft, ce pool d'agents est utilisé pour organiser les agents de construction. Un agent de construction exécute toutes les tâches définies dans le pipeline (Milindanath, 2019, p. 106).

Nous sélectionnons l'image de la machine virtuelle (VM) Windows 2019 correspondant à l'image de Windows Serveur 2019 avec Visual Studio 2019 qui correspond à notre projet.

Tableau 8 - Azure Pipeline hébergé sur des image de machine virtuelle

Virtual machine image	YAML label
Windows Server 2019 with Visual Studio 2019	**windows-latest OR windows-2019**
Windows Server 2016 with Visual Studio 2017	**vs2017-win2016**
Ubuntu 18.04	**ubuntu-latest OR ubuntu-18.04**
Ubuntu 16.04	**ubuntu-16.04**
macOS X Mojave 10.14	**macOS-latest OR macOS-10.14**

Source : (Milindanath, 2019, p. 106)

```
variables:
 solution: '**\*.sln'
 project: '**\LeClanchéMonitor.csproj'
 buildPlatform: 'AnyCPU'
 buildConfiguration: 'Release'
```

Dans l'initialisation d'un pipeline ASP.Net Framework, des variables sont produites pour les utiliser plus tard dans le pipeline. La variable buildPlatform vaut 'Any CPU' en temps normal, mais lorsque l'on veut construire à partir d'un projet .csproj[30] plutôt qu'une solution complète (.sln), cela provoque l'erreur :

```
Warning : The BaseOutputPath/OutputPath property is not set for project…
```

Il est nécessaire de supprimer l'espace entre Any et CPU dans la variable.

```
- task: NuGetToolInstaller@1
 inputs:
  versionSpec:
```

Cette tâche permet de rechercher, télécharger et mettre en cache une version spécifique de NuGet[31]. Lorsque l'on utilise les agents hébergés par Microsoft, il ne faut pas configurer la tâche car celle-ci

[30] Un projet .csproj (abréviation de C# Project) est un projet de développement de logiciel créé et géré par Microsoft Visual Studio.

[31] NuGet : gestionnaire de package .NET open source qui automatise la gestion des bibliothèques externes ainsi que leurs dépendances dans une application. Un package NuGet peut contenir des libraries .NET, des fichiers JavaScript, des images, des fichiers HTML ect.

alourdira le temps d'exécution pour n'apporter qu'une version mineure plus récente. Si elle n'est pas spécifiée, une version sera choisie automatiquement (Danielson & Santos, 2022).

```
- task: NuGetCommand@2
  inputs:
    restoreSolution: '$(solution)'
```

Cette tâche permet de restaurer les packages NuGet utilisés dans la source du projet et doit être ajoutée avant l'étape de build.

```
- task: MSBuild@1
  displayName: BUILD
  inputs:
    solution: '$(project)'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'
    msbuildArguments: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPackageLocation="$(build.artifactStagingDirectory)\WebApp.zip" /p:DeployIisAppPath="Default Web Site"'
```

Cette tâche permet de générer un build. Microsoft recommande d'utiliser MSBuild lorsque l'on souhaite créer un projet spécifique (*.csproj) plutôt qu'une solution complète. Dans l'autre cas, nous aurions utilisé la tâche VSBuild. L'input msbuildArguments permet d'ajouter des arguments supplémentaires au MSBuild :

- **DeployOnBuild** indique à MSBuild que ce projet Web doit être empaqueté/déployé dans le cadre de la construction.
- **WebPublishMethod** garantit que nous créons simplement un package de déploiement. Il existe d'autres options telles que la publication sur le système de fichiers ou ailleurs à l'aide de MSDeploy.
- **PackageAsSingleFile** compresse la sortie dans un seul fichier.
- **DesktopBuildPackageLocation** est l'endroit où le zip est enregistré
- **DeployIisAppPath** est le chemin pour le déploiement de l'application IIS[32]

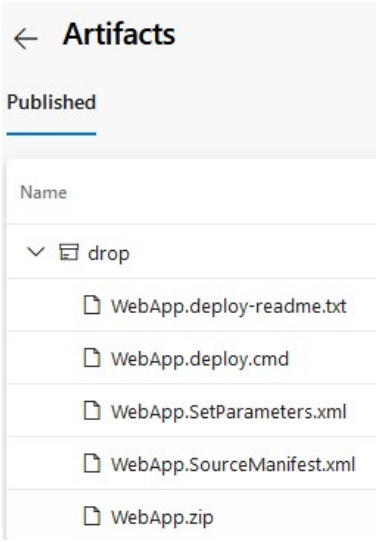
[32] IIS : serveur Web utilisé pour aider les utilisateurs de Windows à héberger divers types de contenu sur le Web, tels que des fichiers multimédias, des documents ou même des sites Web à part entière.

```
- task: PublishBuildArtifacts@1
  displayName: Create Artefact
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

Cette tâche permet de publier des artefacts de build dans le dossier *Build.ArtifactStagingDirectory* qui est le chemin d'accès local sur l'agent où tous les artefacts sont copiés avant d'être poussés vers leur destination.

Le résultat de la construction réussie de ce pipeline donne un Artefact :

Figure 43 - Résultat du Pipeline CI



Source : Données de l'auteur

4.3.2. Tests unitaires

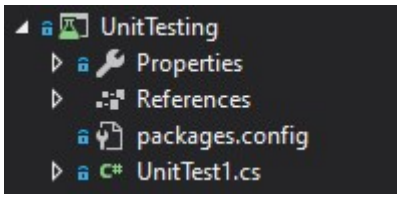
Figure 44 - Tests unitaires

Source : Données de l'auteur

Les tests unitaires permettent de tester la plus petite unité fonctionnelle d'une application. Le but est de valider que chaque unité du code logiciel fonctionne comme prévu. Ils sont effectués pendant la phase de codage d'une application.

Dans le projet storagemanager, il n'y avait pas de test unitaire, afin d'intégrer et d'afficher le fonctionnement dans le pipeline, un test unitaire a été créé dans Visual Studio.

Figure 45 - Structure Visual Studio Unit Test



Source : Données de l'auteur

Celui-ci s'assure que la méthode permettant d'afficher les informations d'énergie fonctionne comme prévu.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using LeClanchéMonitor;
using System;

namespace UnitTesting
{
  [TestClass]
  public class UnitTest1
  {
    Home homeService;
    [TestMethod]
    public void TestDisplayEnergyInfo()
    {
      try
      {
        homeService = new Home();
        homeService.displayEnergyInfo(100);
        Assert.IsTrue(true);
      }
      catch {
        Assert.IsTrue(false);
      }
    }
  }
   }
```

Ce test est intégré dans notre pipeline CI :

```
- task: VisualStudioTestPlatformInstaller@1
  inputs:
    packageFeedSelector: 'nugetOrg'
    versionSelector: 'latestPreRelease'

- task: VSTest@2
  displayName: Test Unit Test
  inputs:
    testSelector: 'testAssemblies'
    testAssemblyVer2: |
      UnitTesting\\bin\\Debug\\UnitTest*.dll
      !**\*TestAdapter.dll
      !**\obj\**
    searchFolder: '$(System.DefaultWorkingDirectory)'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'
```

La tâche **VisualStudioTestPlatformInstaller@1** permet d' installer le Visual Studio Test Platform et ses dépendances sur l'agent de build, ce qui permet à l'agent de build de trouver et d'exécuter les tests unitaires qui sont définis dans le code source.

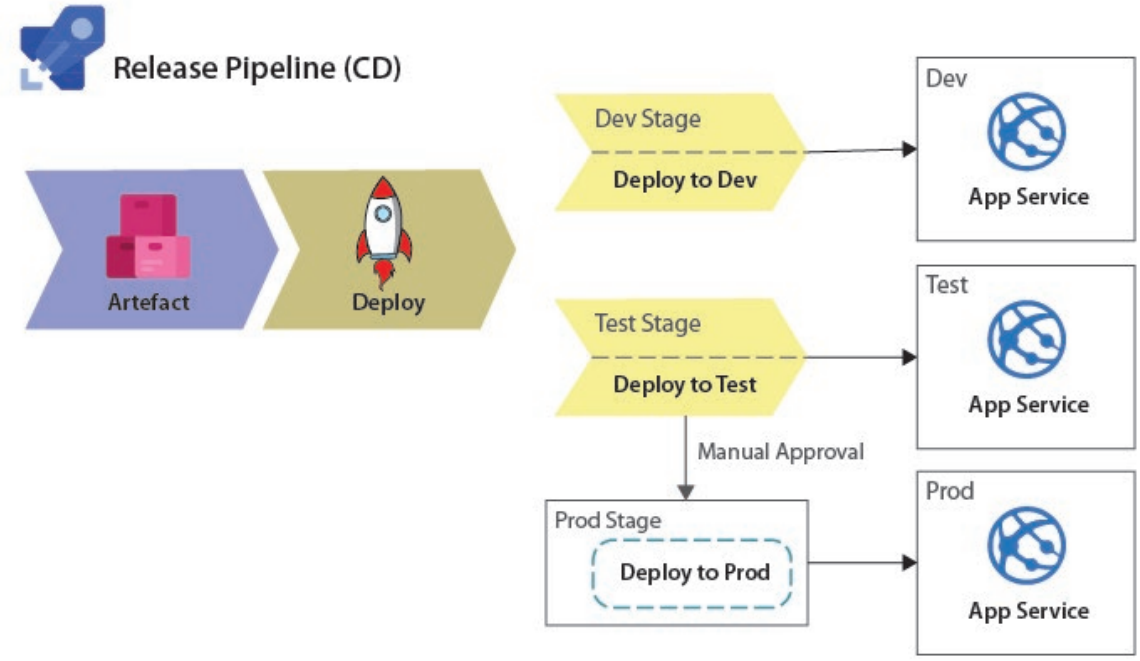
La tâche **VSTest** nous permet d'exécuter des tests unitaires. Elle prendra en entrée le fichier UnitTest*.dll qui est le résultat du build dans Visual Studio du projet UnitTesting. Tout en excluant les fichiers dll qui correspond à TestAdapter.dll ou les fichiers dll se trouvant dans le répertoire obj car ceux-ci ne contiennent pas de code de test unitaire valide ce qui évite des erreurs et améliore la qualité de test.

Figure 46 - Résultat Unit Test

Source : Données de l'auteur

4.3.3. Azure Pipeline Release (CD)

Figure 47 - Pipeline release CD



Source : Données de l'auteur

Nous avons ajouté une tâche à notre pipeline CI afin de mettre un tag sur l'artefact produit, cela va nous permettre de classer notre déploiement dans le bon environnement dans le release pipeline:

```
- script: |
  echo ##vso[build.addbuildtag]Test
```

Figure 48 - Tag du pipeline *Test*

Source : Données de l'auteur

Dès que notre build CI a été testé et que nous n'avons pas trouvé de vulnérabilités critiques de sécurité, la dernière étape du pipeline consiste à déployer les changements de façon continue (CD).

Le résultat de notre précédent pipeline a donné un Artefact qui est mis en entrée du pipeline de release afin d'être déployé dans un App Service.

Un App Service est un service proposé par Microsoft Azure qui facilite la création, le déploiement et la gestion d'applications web, mobiles et hybrides, sans avoir à gérer les infrastructures nécessaires.

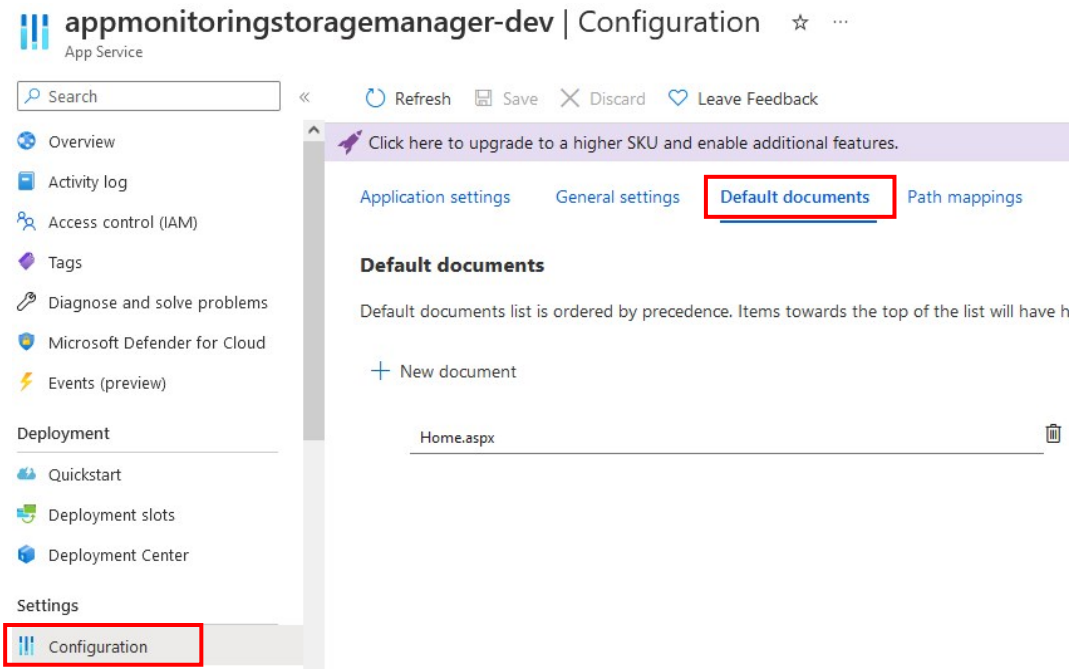
Installation d'un App Service sur Azure Portal

Tout d'abord, connectez-vous à Azure Portal et créez une nouveau App Services via *Home* > *App Services* > *Create*. Puis, créez un Groupe de ressources[33], définissez le plan de tarification et créez l'application web (voir Annexe VI).

Il faut ensuite déterminer les documents par défaut. Ce sont les pages Web affichées à la racine d'une application App Service. Le premier fichier correspondant de la liste est utilisé pour l'affichage (Lin, et al., 2022).

Enfin, sélectionnez *Configuration* > *Default documents* > *New document* et ajoutez le fichier .NET de votre page d'accueil. Dans notre projet storagemanager : *Home.aspx*.

Figure 49 - Configuration Default documents App Service

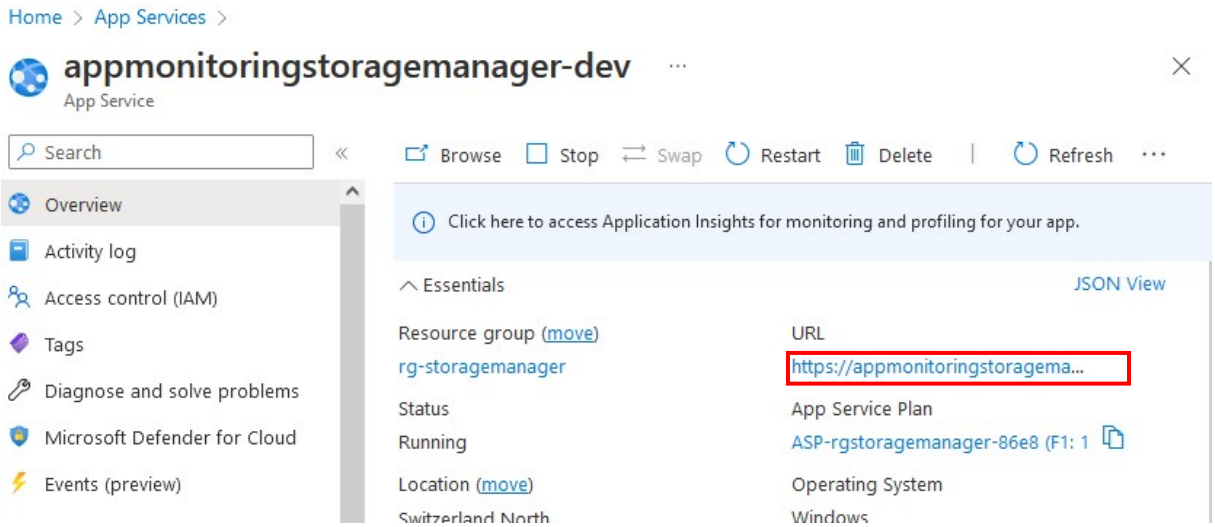


Source : Données de l'auteur

Vous pouvez à présent consulter le site à l'URL créée sur l'Overview de votre App service :

[33] Un groupe de ressources est un conteneur qui contient des ressources associées partageant le même cycle de vie afin de pouvoir facilement les déployer, les mettre à jour et les supprimer en tant que groupe. La localisation spécifie la région où sont stockés les métadonnées du groupe.

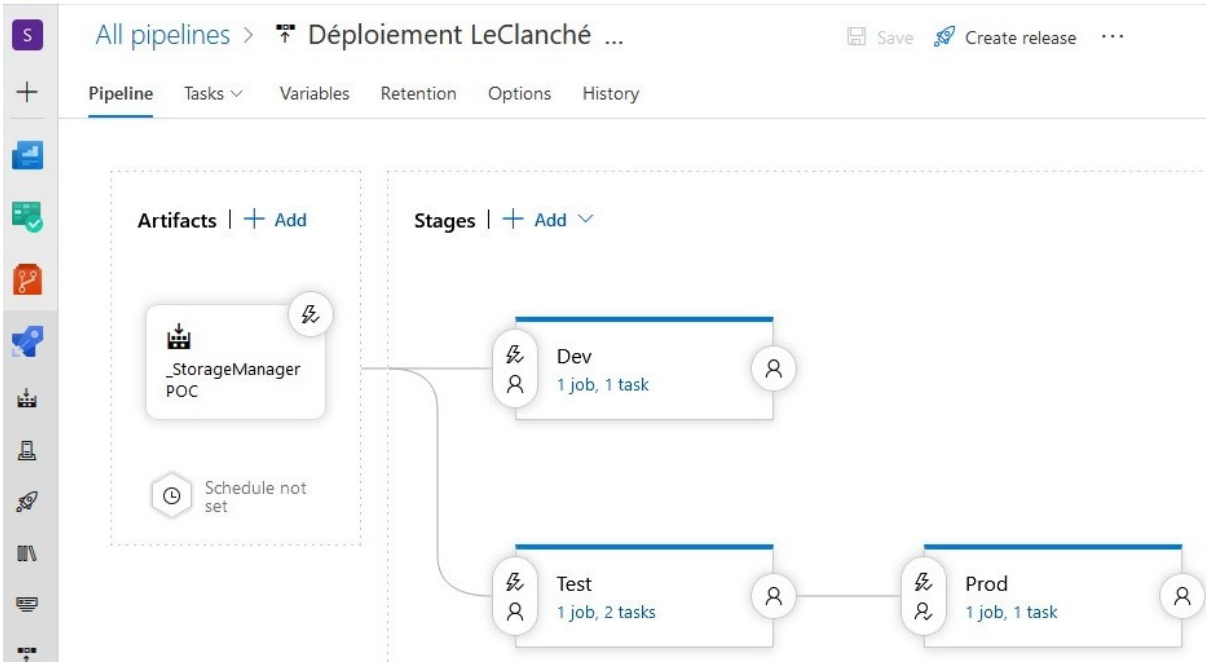
Figure 50 - Overview App Service



Source : Données de l'auteur

Configuration du pipeline de release sur Azure DevOps :

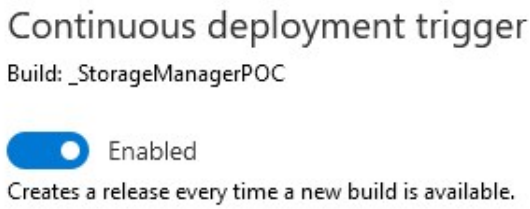
Figure 51 - Pipeline de release



Source : Données de l'auteur

Nous configurons le déclencheur  pour un déploiement continu, cela créera une release à chaque fois qu'un nouvel artefact est disponible :

Figure 52 - Déclencheur déploiement continu



Source : Données de l'auteur

Nous configurons également des conditions de pré-déploiement afin de diriger selon le tag de notre artefact dans la bonne étape du pipeline.

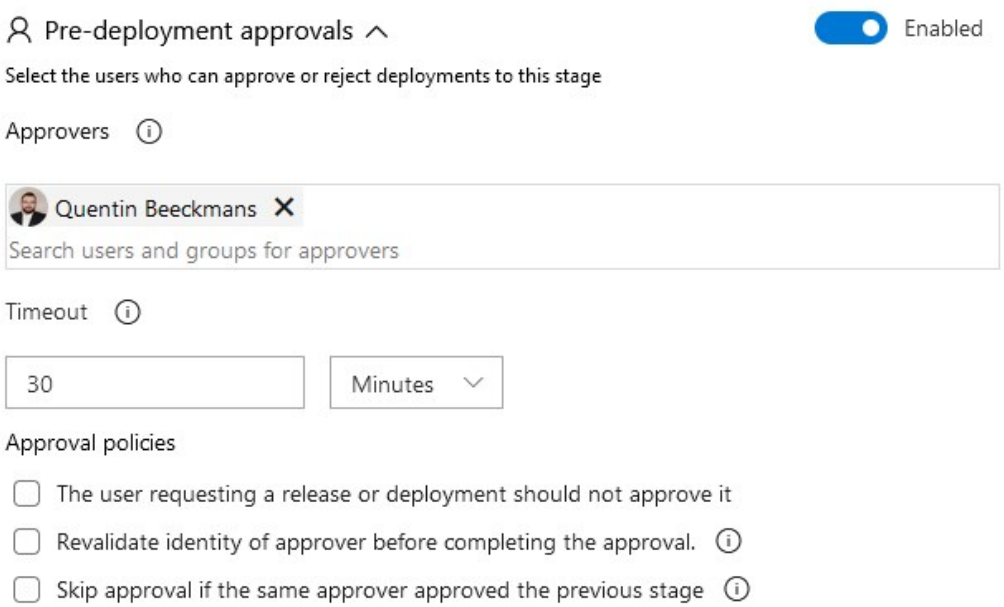
Si notre artefact a un tag appelé « Test », l'étape avec un filtre sur ce tag sera activée.

Figure 53 - Condition de pré déploiement filtre d'artefact tags

Source : Données de l'auteur

L'étape Prod est quant à elle configurée pour que seule une approbation manuelle soit possible :

Figure 54 - Condition de pré-déploiement filtre d'artefact manuelle



Source : Données de l'auteur

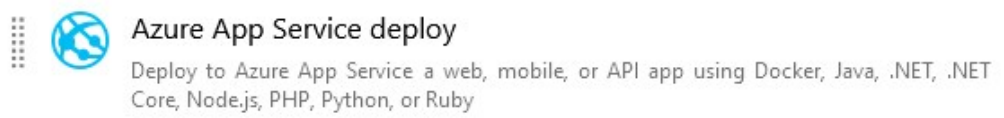
Cette étape permet de définir les personnes pouvant approuver le déploiement ainsi que le délai d'approbation avant d'échouer.

Il nous reste à ajouter la tâche *Azure App Service Deploy* qui est identique dans chaque étapes (Dev, Test, Prod) mais dont la cible diffère :

- https://appmonitoringstoragemanager-dev.azurewebsites.net/
- https://appmonitoringstoragemanager-test.azurewebsites.net/
- https://appmonitoringstoragemanager-prod.azurewebsites.net/

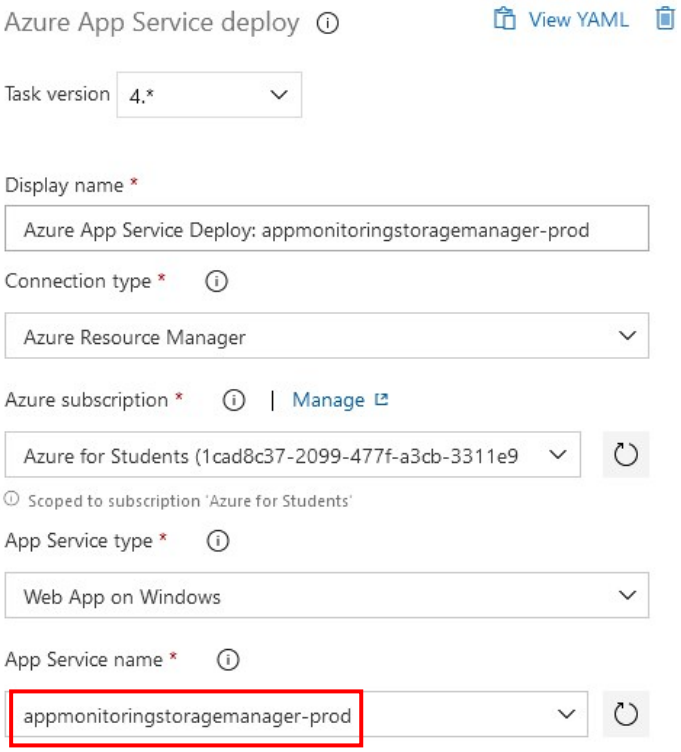
De cette manière, à chaque build réussi dans notre CI, cela déclenchera notre CD de façon automatisé et ce jusqu'à la publication de notre site avec les derniers changements.

Figure 55 - Azure App Service deploy



Source : Données de l'auteur

Figure 56 - Configuration Azure App Service deploy



Source : Données de l'auteur

4.3.4. Selenium (Tests fonctionnels)

Figure 57 - Pipeline Selenium

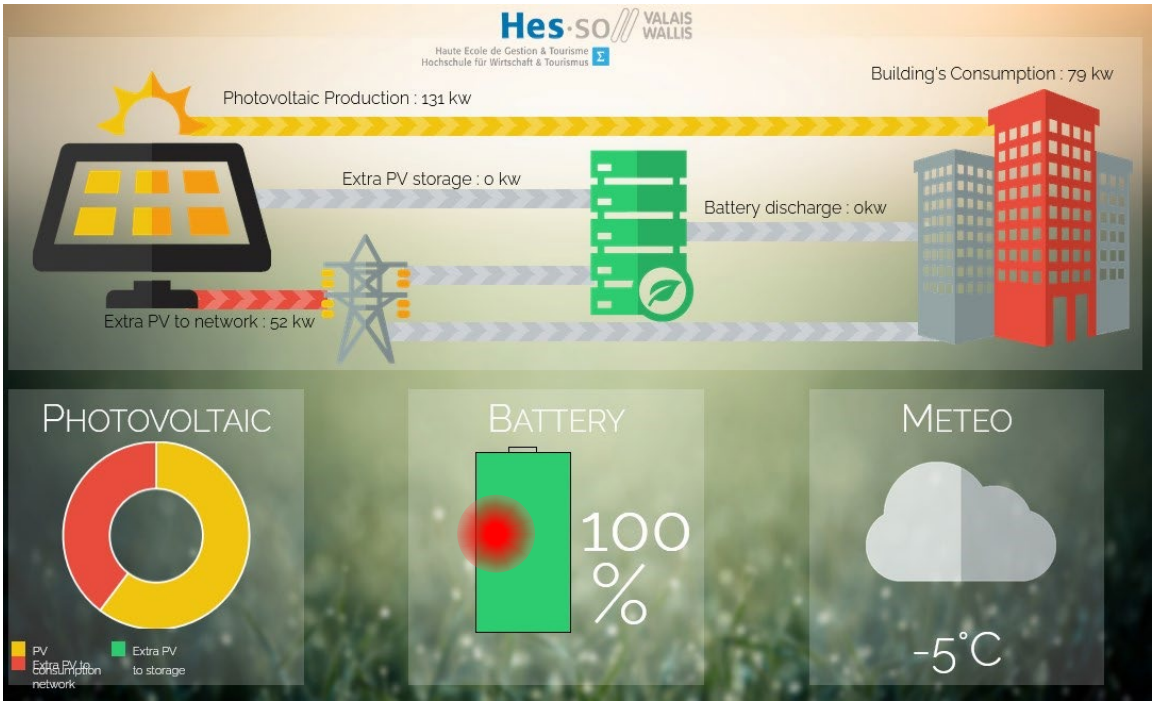
Source : Données de l'auteur

Selenium est un outil de test fonctionnel qui permet aux développeurs de simuler des actions de l'utilisateur final. Tel que la saisie de texte dans des champs de formulaire, la sélection de valeurs dans des listes déroulantes, cocher des cases et de cliquer sur les liens dans les documents. Selenium fournit également d'autres fonctionnalités comme le mouvement de la souris et l'exécution de code JavaScript (A deeper look at Selenium, 2022).

Au départ, il n'était pas prévu de créer de test Selenium pour ce projet. Cependant, nous avons décidé de les inclure pour montrer comment ils fonctionnent dans le pipeline Azure.

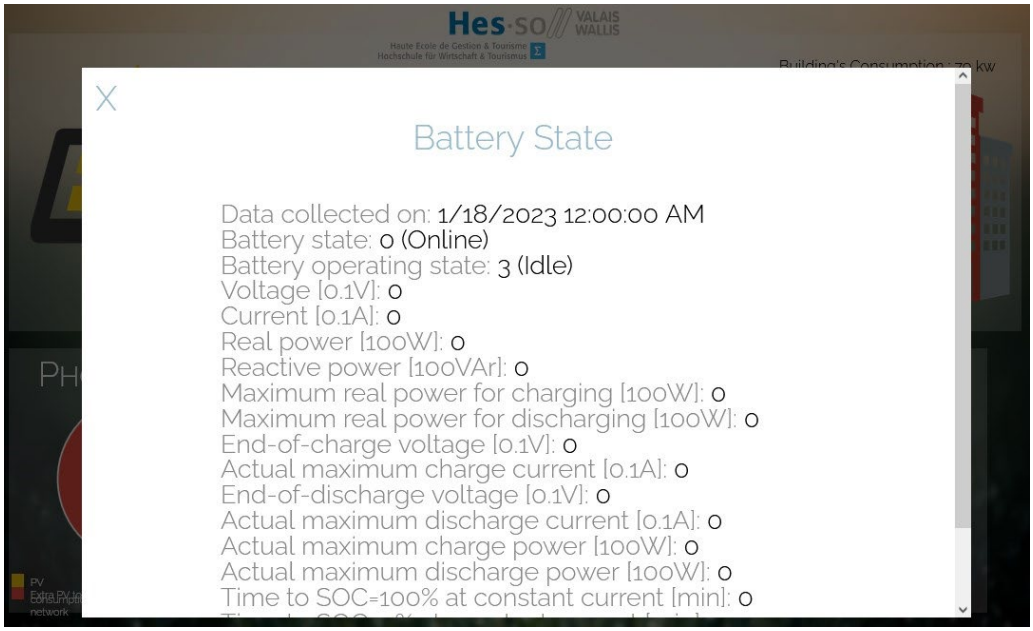
Nous avons créé un test Selenium dans Visual Studio pour nous assurer du bon fonctionnement du site sur chrome en simulant l'action de l'utilisateur final lorsqu'il clique sur l'image de la batterie, faisant apparaître de cette manière un pop-up avec les données de l'état de la batterie :

Figure 58 - Animation au clic de l'image de la batterie 1/2



Source : Données de l'auteur

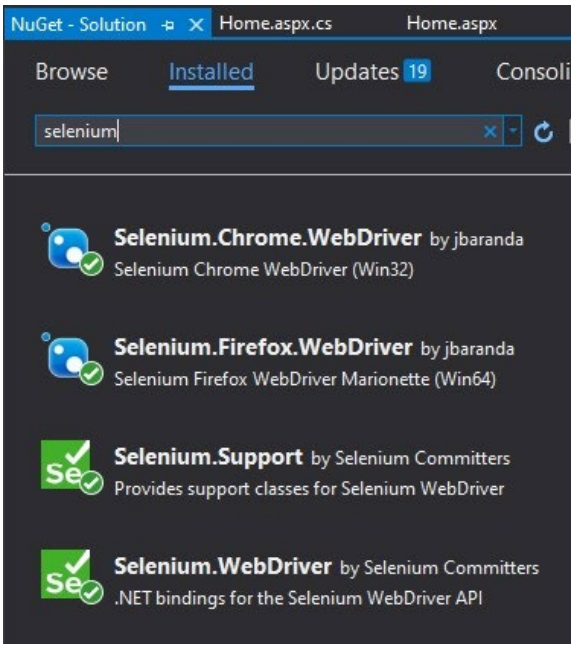
Figure 59 - Animation au clic de l'image de la batterie 2/2



Source : Données de l'auteur

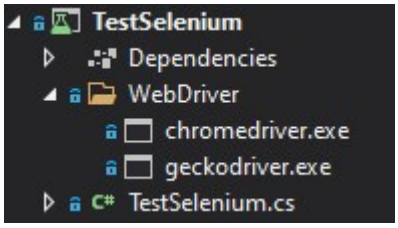
Pour créer ce test sur Visual Studio, il a été nécessaire de créer un nouveau projet NUnit Test, en y ajoutant les packages Selenium dans le gestionnaire NuGet Package :

Figure 60 - NuGet Package Selenium Visual Studio



Source : Données de l'auteur

Figure 61 - Structure Visual Studio Selenium



Source : Données de l'auteur

Le code de notre fichier de test Selenium:

```
using System;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Chrome;


namespace TestSelenium
{
    [TestClass]
    public class MySeleniumTests
    {
```

```
        private TestContext testContextInstance;
        private IWebDriver driver;
        private string appURL;

        public MySeleniumTests()
        {
        }

        [TestMethod]
        [TestCategory("Chrome")]
        public void PopUpBatteryStateTest()
        {
            driver.Navigate().GoToUrl(appURL + "/");
            driver.FindElement(By.Id("pileInfoPanel")).Click();
            Assert.IsTrue(driver.FindElement(By.Id("light")).Displayed, "Verified if popup work after
clic");
        }


        public TestContext TestContext
        {
            get
            {
                return testContextInstance;
            }
            set
            {
                testContextInstance = value;
            }
        }

        [TestInitialize()]
        public void SetupTest()
        {
            appURL = "https://appmonitoringstoragemanager-test.azurewebsites.net/";
            string browser = "Chrome";
            switch (browser)
            {
                case "Chrome":
                    driver = new ChromeDriver("**\\WebDriver\\chromedriver.exe");
                    break;
                case "Firefox":
                    driver = new FirefoxDriver("**\\WebDriver\\geckodriver.exe");
                    break;
                default:
                    driver = new ChromeDriver("**\\WebDriver\\chromedriver.exe");
                    break;
            }
        }

        [TestCleanup()]
        public void MyTestCleanup()
        {
            driver.Quit();
        }
    }
}
```

L' annotation **[TestMethod]** indique à Microsoft Visual Studio que la méthode est une méthode de test et **[TestCategory("Chrome")]** indique la catégorie de test associée à la méthode.

La méthode **PopUpBatteryStateTest()** est utilisée pour vérifier si un popup s'affiche correctement après un clic sur l'image de la batterie.

La méthode **SetupTest()** est appelée avant chaque méthode de test et sert à configurer l'environnement de test en définissant l'URL de l'application web à tester et en initialisant l'objet WebDriver pour le navigateur configuré.

La méthode **MyTestCleanup()** est appelée après chaque méthode de test et sert à nettoyer l'environnement de test en fermant le navigateur qui a été utilisé pour les tests.

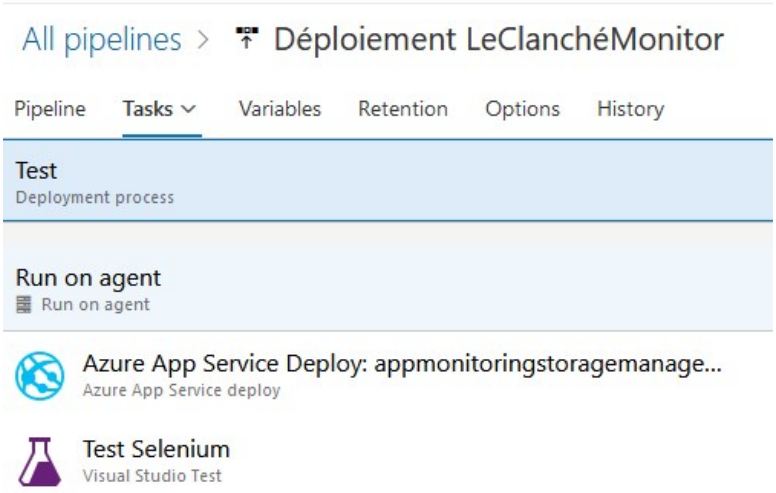
Intégration dans Azure :

Pour commencer, nous devons ajouter une étape de copie de fichier dans notre pipeline CI pour transférer le fichier DLL de notre test Selenium vers le pipeline de release :

```
- task: CopyFiles@2
  displayName: Copy content of bin for TestSelenium
  inputs:
    SourceFolder: '$(build.sourcesdirectory)'
    Contents: '**\TestSelenium\bin\**'
    TargetFolder: '$(build.artifactstagingdirectory)'
```

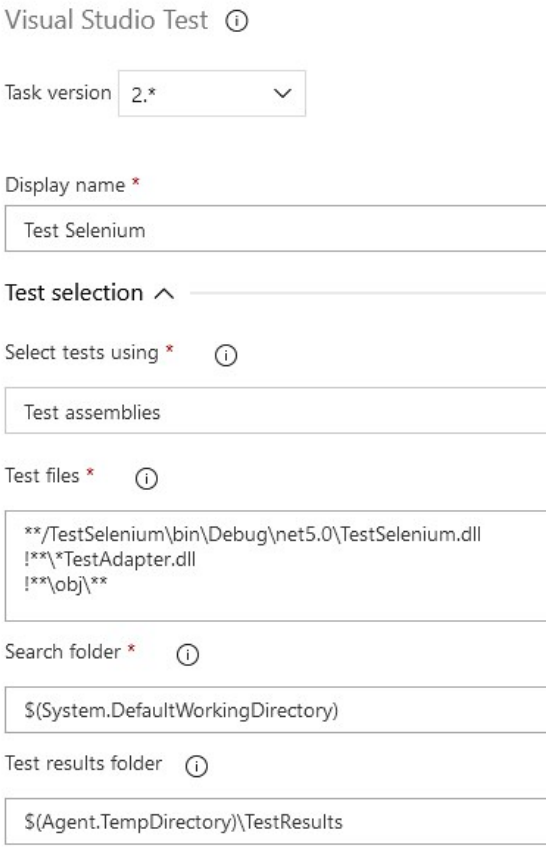
Ensuite, nous devons ajouter une tâche VSTest dans notre pipeline de release à la suite de notre application web nouvellement déployée :

Figure 62 - Intégration d'une tâche VSTest pour Selenium



Source : Données de l'auteur

Figure 63 - Configuration de la tâche de test Selenium dans le pipeline de release

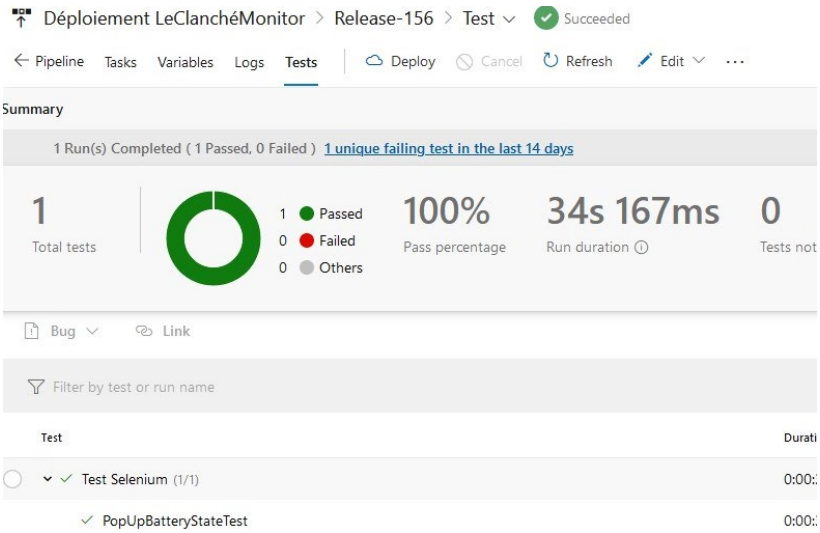


Source : Données de l'auteur

La tâche VSTest prendra en entrée le fichier TestSelenium.dll. TestAdapter.dll et les fichiers contenus dans le dossier obj ne doivent pas être inclus dans les résultats de test.

Le résultat de ce test est affiché dans le pipeline de release :

Figure 64 - Résultat du test Selenium dans le pipeline release



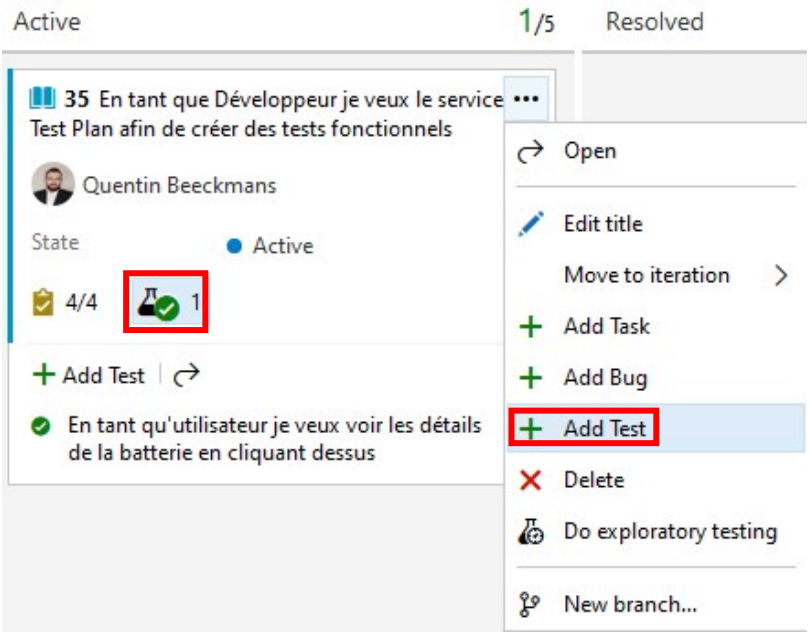
Source : Données de l'auteur

4.3.5. Azure Test Plan

Test Plan est une fonctionnalité de Azure DevOps permettant de planifier, suivre et gérer les tests d'un projet. Elle est disponible gratuitement à l'essai pendant 30 jours. Pour activer cet essai, il faut se rendre dans son Organisation Azure DevOps > Organization settings > Billing > Activer l'essai.

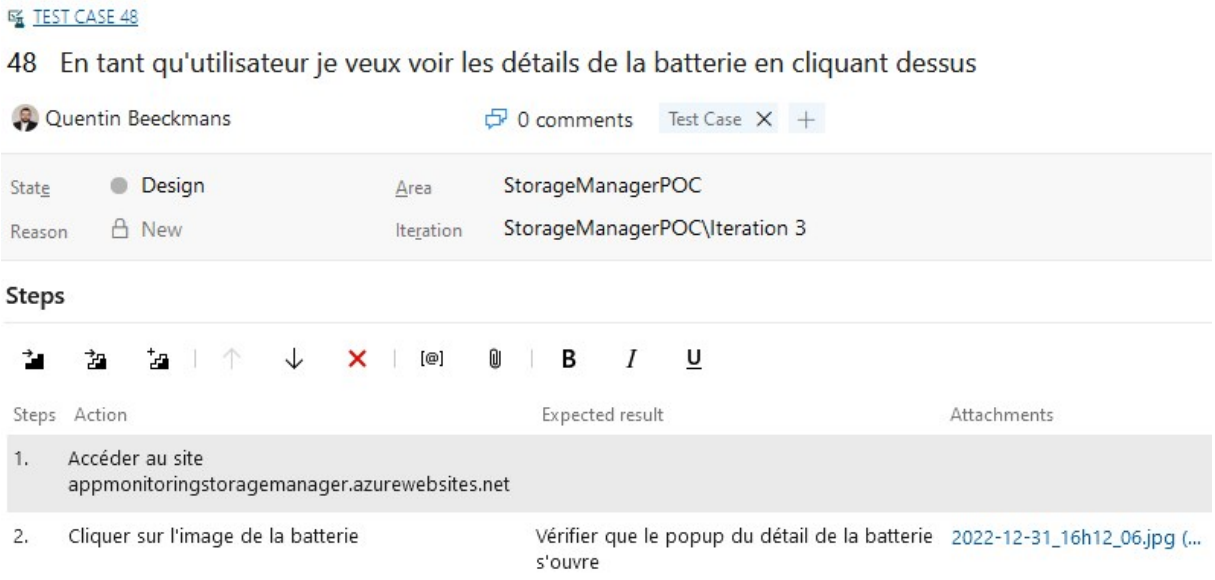
Nous pouvons par exemple créer directement des tests Plan dans la User Stories de notre Azure Boards :

Figure 65 - Création d'un Test Plan via le Boards



Source : Données de l'auteur

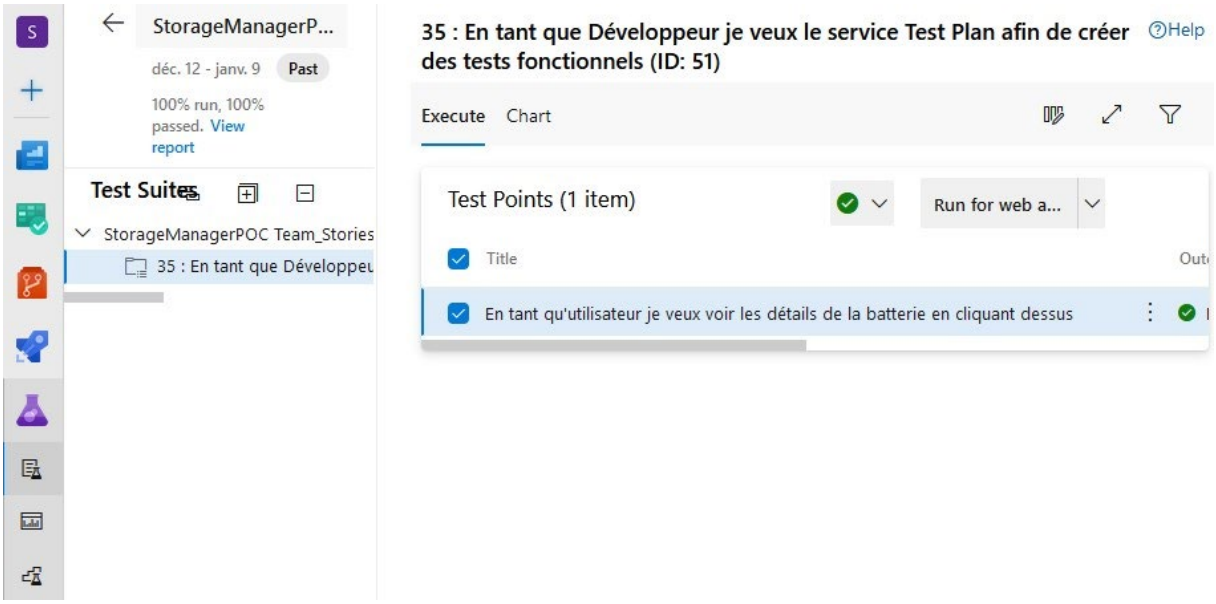
Figure 66 - Exemple de Tests Case



Source : Données de l'auteur

Le testeur peut lancer le test dans l'onglet Test Plans, sélectionner le test et en cliquant sur Run for web application :

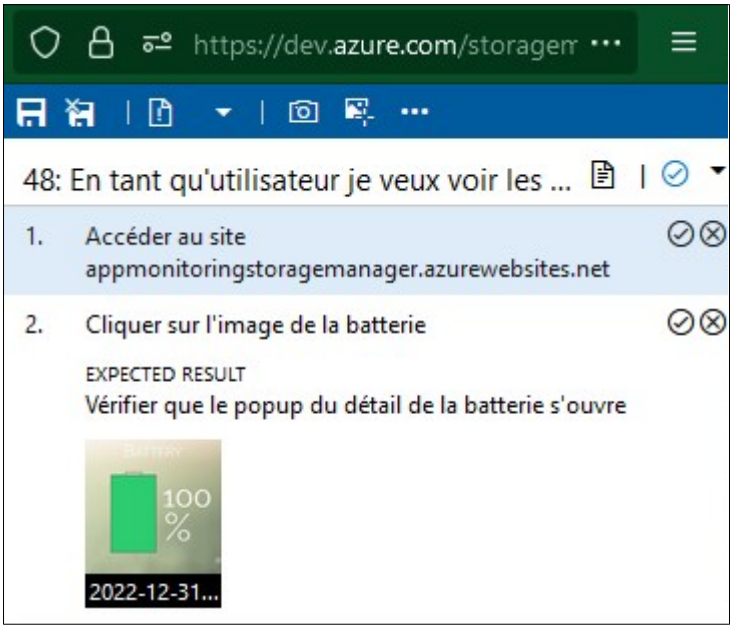
Figure 67 - Exécution d'un test plan 1/2



Source : Données de l'auteur

Une fois lancé une page de formulaire s'ouvre :

Figure 68 - Formulaire de Test Case



Source : Données de l'auteur

Il nous reste à vérifier les étapes manuellement et cocher les cases correspondantes. Il est en outre possible de créer directement une tâche de bug en cas de problème depuis cet interface.

4.4. Implémentation des outils DevSecOps

4.4.1. OWASP Dependency Check[34] (SCA)

Figure 69 - Pipeline SCA

Source : Données de l'auteur

Pour installer OWASP Dependency Check, il faut se rendre dans le marketplace et l'ajouter à notre projet :

https://marketplace.visualstudio.com/items?itemName=dependency-check.dependencycheck

Dans notre pipeline CI, il faut ajouter la tâche OWASP Dependency Check après la tâche MSBuild configurée comme suit:

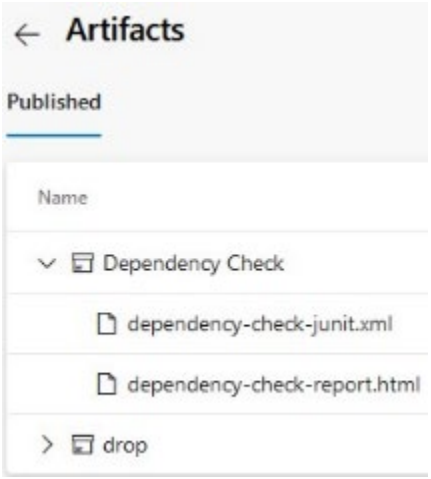
```
- task: MSBuild@1
[…]
- task: dependency-check-build-task@6
 inputs:
  projectName: 'Dependency Check'
  scanPath: '**/*.dll'
      format: 'HTML, JUNIT'
```

Le scanPath est défini pour rechercher tous les fichiers au format .dll conformément au paramétrage pour un projet dotnet (jeremylong.github, 2022).

Le format de sortie est ici défini en HTML et JUNIT, ces fichiers se retrouvent en tant qu'artefact tout comme le build du projet et peuvent y être consulté.

[34] https://marketplace.visualstudio.com/items?itemName=dependency-check.dependencycheck

Figure 70 - Artefact produit par OWASP Dependency Check (SCA)



Source : Données de l'auteur

La tâche « Publish Test Results » placé après la précédente tâche permet de récupérer le fichier junit.xml et de l'afficher dans l'interface des tests de façon intégré au pipeline.

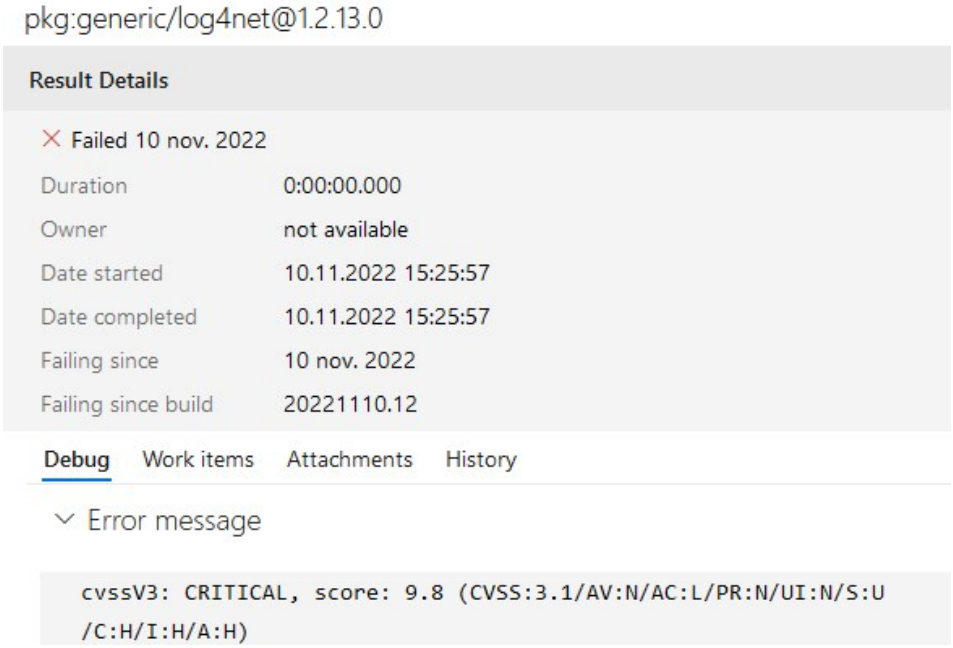
```
- task: dependency-check-build-task@6
[…]
- task: PublishTestResults@2
 inputs:
  testResultsFormat: 'JUnit'
  testResultsFiles: 'dependency-check\*junit.xml'
  searchFolder: '$(Common.TestResultsDirectory)'
  testRunTitle: 'Dependency Check'
```

Figure 71 - Résultat du test OWASP Dependency Check dans le pipeline

Source : Données de l'auteur

Le résultat du test met en évidence un package présentant un risque critique :

Figure 72 - Problème critique trouvé avec OWASP Dependency Check



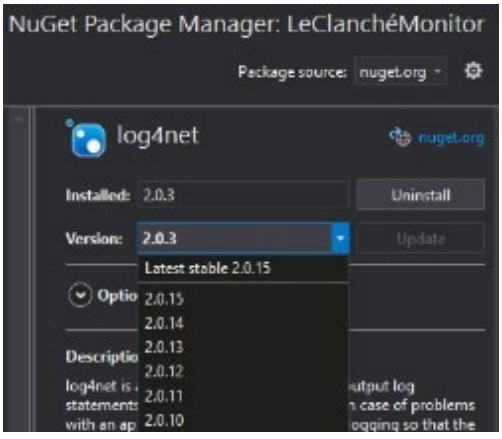
Source : Données de l'auteur

Le détail de l'erreur donne :

« Apache log4net versions before 2.0.10 do not disable XML external entities when parsing log4net configuration files. This allows for XXE-based attacks in applications that accept attacker-controlled log4net configuration files" (Tests OWASP Dependency Check, 2022).

Une mise à jour du NuGet résoudra le problème :

Figure 73 - NuGet Package Manager log4net



Source : Données de l'auteur

Après avoir mis à jour le package à la bonne version, nous effectuons un commit[35] du code pour mettre à jour le repository :

Figure 74 - Git push Update Dependency Log4Net

```
Quentin@LAPTOP-7KV28DN4 MINGW64 ~/source/Workspaces/StorageManager/StorageManager (main)
$ git commit -am "Update Dependency Log4Net"
[main f701cef] Update Dependency Log4Net
 14 files changed, 47 insertions(+), 12 deletions(-)
 rewrite "LeClanch\303\251Monitor/obj/Debug/DesignTimeResolveAssemblyReferencesInput.cache" (61%)

Quentin@LAPTOP-7KV28DN4 MINGW64 ~/source/Workspaces/StorageManager/StorageManager (main)
$ git push
Enumerating objects: 54, done.
Counting objects: 100% (54/54), done.
Delta compression using up to 12 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (29/29), 6.38 KiB | 261.00 KiB/s, done.
Total 29 (delta 19), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (29/29) (55 ms)
remote: Storing packfile... done (70 ms)
remote: Storing index... done (66 ms)
To https://dev.azure.com/storagemanager/StorageManagerPOC/_git/StorageManagerPOC
   bdd2b0f..f701cef  main -> main
```

Source : Données de l'auteur

Résultat :

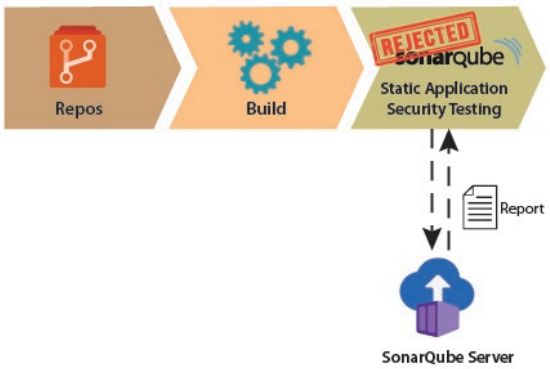
Figure 75 - Résultat du test OWASP Dependency Check dans le pipeline après correction erreur

Source : Données de l'auteur

[35] Un commit est une action qui permet de sauvegarder des modifications dans un dépôt de code source. Les commits peuvent être considérés comme des instantanés ou des étapes importantes dans la chronologie d'un projet Git.

4.4.2. SonarQube (SAST) (Solution abandonnée au profit de SonarCloud)

Figure 76 - Pipeline SonarQube abandonné



Source : Données de l'auteur

Pour installer SonarQube, il faut se rendre dans le marketplace et l'ajouter à notre projet :

https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarqube

Nous avons tout d'abord installé le plugin dans notre projet Azure, ensuite il est obligatoire d'avoir un serveur SonarQube en fonctionnement pour recevoir le rapport d'analyse, sans quoi la tâche dans le pipeline échoue. La solution SonarQube gratuite Community permet, via son image Docker, de la déployer dans un container.

Azure Container Instances (ACI) est un moyen simple et efficace d'exécuter un conteneur dans le cloud Azure en éliminant le besoin de gérer des machines virtuelles ou d'utiliser des services d'orchestration de conteneurs plus complexes. De plus, ACI est basé sur un modèle sans serveur et il est idéal pour les charges de travail simples pour des applications à plus petite échelle.

Le serveur SonarQube une fois installé a permis d'échanger des informations avec la tâche SonarQube de notre pipeline DevOps pour en traiter les données et retourner une analyse approfondie des erreurs présentes dans le code. A cette dernière étape, nous pouvons alors éteindre le serveur SonarQube pour arrêter la tarification à la seconde, ayant prouvé son fonctionnement.

Problème :

Nous avons rencontré un problème lorsqu'il a été nécessaire de redémarrer le serveur SonarQube pour une analyse postérieur. Il semble que la version actuelle de SonarQube 9.7 ne permet plus la persistance des données[36].

Tentatives de résolutions :

Un système de partage de fichier a été mis en place pour mapper les différents dossier SonarQube : /opt/sonarqube/data, /opt/sonarqube/extensions et /opt/sonarqube/logs ainsi qu'une connexion à une base de données SQL pour permettre une persistance des données.

Le script testé :

```
az container create --location westus --resource-group rg-storagemanager --name sonarqubeserver --image sonarqube:9.8.0-community --os-type Linux --ip-address Public --dns-name-label storagemanager --ports 9000 --cpu 2 --memory 3.5 \

--environment-variables SONAR_JDBC_USERNAME=[redacted] SONAR_JDBC_PASSWORD=[redacted] SONAR_JDBC_URL='jdbc:sqlserver://storagemanager.database.windows.net:1433;database=sonar;user=[redacted];password=[redacted];encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;' \

--azure-file-volume-account-name stastoragemanager --azure-file-volume-account-key \"UUo0lOGGUw2mTVmuufG/cIqYsRFVOgmoGgtbhvzagekX99SxYL/m5Hp27NzIwPcScmuu5jrejeH+AStUV+1gQ==\" \

--azure-file-volume-share-name data --azure-file-volume-mount-path /opt/sonarqube/data \

--azure-file-volume-share-name extensions --azure-file-volume-mount-path /opt/sonarqube/extensions \

--azure-file-volume-share-name logs --azure-file-volume-mount-path /opt/sonarqube/logs \ --log-analytics-workspace 50e11b76-5fd9-4aed-9a81-ee77d209db8f \

 --log-analytics-workspace-key bKH+PlOp4xBEtyB/jywjDK1tkV60DaFkLw6DeQ6zX2QMwVmOIBbjJM8n2ZFJwp8jTSNxs57oKg27GUYxhNTSUQ==
```

Le but de ce script est de créer un ACI de l'image docker SonarQube, la base de données SQL Azure avec la connexion, les fichiers de partages et une connexion à un service de log Azure.

Malheureusement, au lancement de l'instance, celui-ci comporte des erreurs empêchant le démarrage.

[36] La persistance des données en programmation désigne l'ensemble des techniques qui permettent de stocker des données.

Pistes de solution :

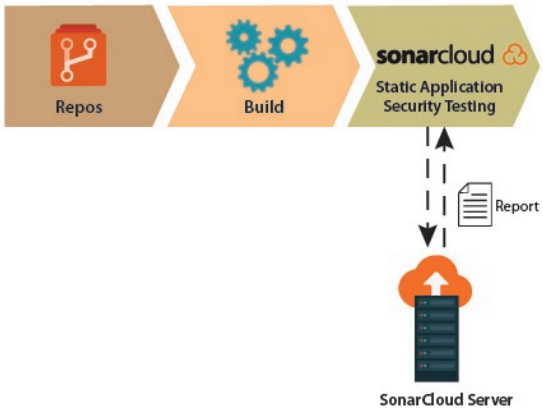
Dans la multitude de logs nous avons isolé un problème récurrent causé par elasticsearch[37] qui est un processus de recherche utilisé par SonarQube. C'est un élément redondant dans les problèmes antérieurs trouvés sur les forums mais qui n'a pas trouvé de solution pour notre cas.

Décision :

Pour pallier au problème de persistance de SonarQube et pouvoir bénéficier malgré tout de son expertise d'analyse , il a été décidé d'utiliser son homologue cloud SonarCloud dans sa version open source.

4.4.3. SonarCloud[38] (SAST)

Figure 77 - Pipeline SonarCloud



Source : Données de l'auteur

SonarCloud évalue le code source de notre projet par rapport à un ensemble de règles appelées profils de qualité. Ces règles permettent d'évaluer la maintenabilité, la fiabilité, la sécurité, la couverture du code et les lignes dupliquées. Il fournit ensuite des rapports sur la qualité du code.

Pour son implémentation dans le pipeline, SonarCloud fonctionne de la même manière que SonarQube en intégrant trois tâches presque identiques, notamment :

[37] Elasticsearch : https://www.elastic.co/fr/about/

[38] http://sonarcloud.io/

```
# SonarCloud
- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: 'SonarCloud Connection'
    organization: 'storagemanager'
    scannerMode: 'MSBuild'
    projectKey: 'storagemanager_StorageManagerPOC'
    projectName: 'StorageManagerPOC'
    extraProperties: sonar.prokectBaseDir=$(Agent.TempDirectory\\**\coverage.cobertura.xml
# SonarQube
#- task: SonarQubePrepare@5
#  inputs:
#    SonarQube: 'SonarQubeServer Connection'
#    scannerMode: 'MSBuild'
#    projectKey: 'StorageManagerPOC_StorageManagerPOC_AYSkMzj_0KDvBk7hOf1t'
#    projectName: 'sonar.Storagemanager'
#    extraProperties: 'sonar.cs.opencover.reportsPaths=$(Agent.TempDirectory)/*/coverage.op
encover.xml'
```

Documentation de l'installation de SonarCloud :

https://docs.sonarcloud.io/getting-started/azure-devops/

Afin d'utiliser SonarCloud dans sa version gratuite, il a été nécessaire de rendre le projet Azure public. Pour ce faire, il faut se rendre tout d'abord dans *Organization Setting* > *Policies* et activer *Allow public projects*.

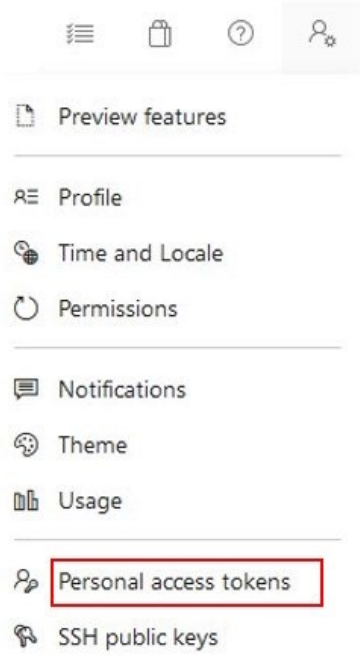
Ensuite dans notre projet, dans Project Settings > Overview > Visibility , changer la valeur sur Public.

N.B. Lors du changement de l'état de visibilité d'un projet avec un agent autohébergé gratuit, il est obligatoire de refaire une demande d'accès via le formulaire et attendre quelques jours avant que la demande soit traitée.

En premier lieu, il est nécessaire de générer un jeton d'accès personnel[39] sur Azure pour pouvoir lier SonarCloud dans *User settings* > *Personal access tokens* > *New Token* :

[39] Jeton d'authentification (Token) : est un dispositif matériel utilisé pour permettre l'accès à une ressource protégée électroniquement.

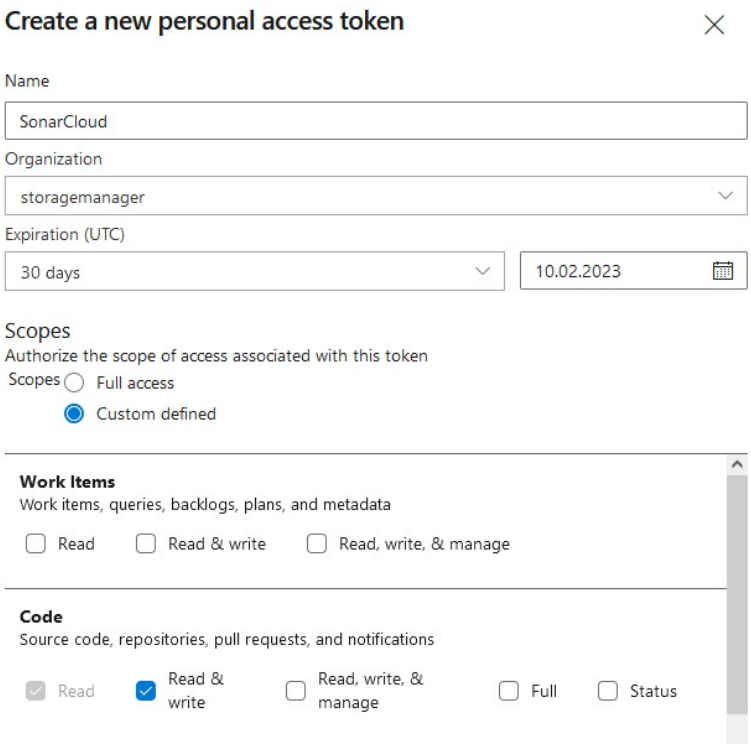
Figure 78 - Génération d'un token Azure pour SonarCloud 1/2



Source : Données de l'auteur

Attention : SonarCloud nécessite un token avec au minimum la permission Read/Write du code :

Figure 79 - Génération d'un token Azure pour SonarCloud 2/2



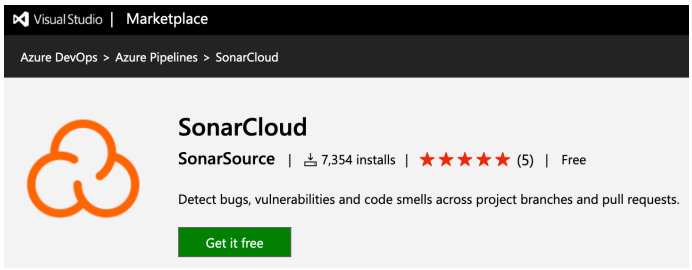
Source : Données de l'auteur

Une fois le token créé, copiez celui-ci dans un autre endroit car celui-ci ne sera plus récupérable par la suite.

Connectez-vous à https://sonarcloud.io/ et créez un compte avec votre token Azure DevOps et configurez votre organisation.

Ajoutez l'extension SonarCloud au projet Azure via le marketplace à l'adresse : https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarcloud

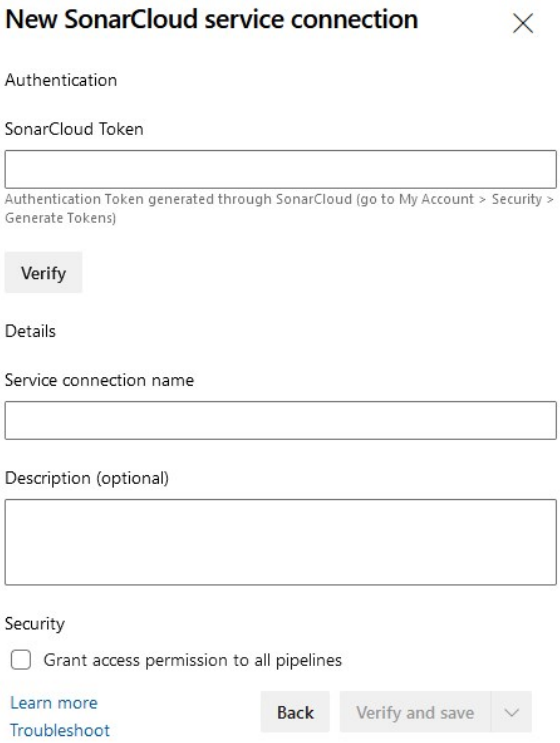
Figure 80 - SonarCloud extension



Source : (sonarcloud, 2022)

Créez ensuite un point d'accès dans Azure pour SonarCloud en allant dans *Project settings* > *Service connections* afin de valider le lien entre Azure Pipeline et SonarCloud en utilisant le token fourni par SonarCloud.

Figure 81 - Nouvelle connexion de service SonarCloud



Source : Données de l'auteur

Terminez en récupérant les informations fournies par SonarCloud dans la dernière étape à savoir : la clé et le nom du projet.

Ajoutez ces trois tâches dans le pipeline de notre projet :

- **Prepare Analysis on SonarCloud** doit être configurée avant la tâche de Build de notre pipeline. Ajoutez la clé de projet générée à partir de SonarCloud plus tôt.
- **Run Code Analysis** après la tâche de build.
- **Publish Quality Gate Result** à la suite.

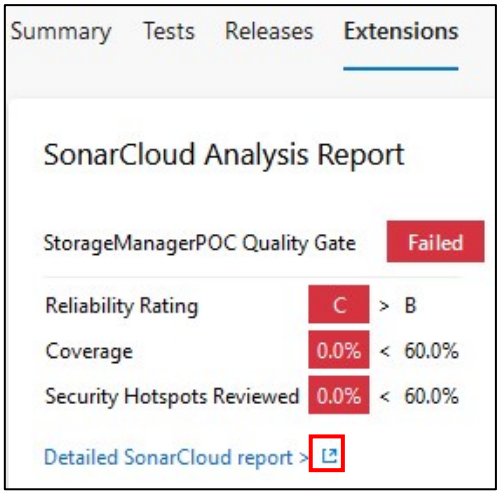
Résultat de l'intégration des tâches SonarCloud dans notre pipeline Azure:

```
trigger:
- main

pool:
  vmImage: 'windows-2019'

variables:
  solution: '**\*.sln'
  project: '**\LeClanchéMonitor.csproj'
  buildPlatform: 'AnyCPU'
  buildConfiguration: 'Release'

steps:
- task: NuGetToolInstaller@1
  inputs:
    versionSpec:
- task: NuGetCommand@2
  inputs:
    restoreSolution: '$(solution)'

- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: 'SonarCloud Connection'
    organization: 'storagemanager'
    scannerMode: 'MSBuild'
    projectKey: 'storagemanager_StorageManagerPOC'
    projectName: 'StorageManagerPOC'

- task: MSBuild@1
  displayName: BUILD
  inputs:
    solution: '$(project)'
    platform: '$(buildPlatform)'
```

```
  configuration: '$(buildConfiguration)'
  msbuildArguments: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPackageLocation="$(build.artifactStagingDirectory)\WebApp.zip" /p:DeployIisAppPath="Default Web Site"'

- task: SonarCloudAnalyze@1

- task: SonarCloudPublish@1
  inputs:
    pollingTimeoutSec: '300'

- task: PublishBuildArtifacts@1
  displayName: Create Artefact
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

A la fin de notre build, un résumé du rapport et un lien sont affichés dans la tabulation *Extensions :*

Figure 82 - Résultat de notre build avec SonarCloud



Source : Données de l'auteur

Le lien nous permet d'afficher le rapport détaillé sur le plateforme SonarCloud :

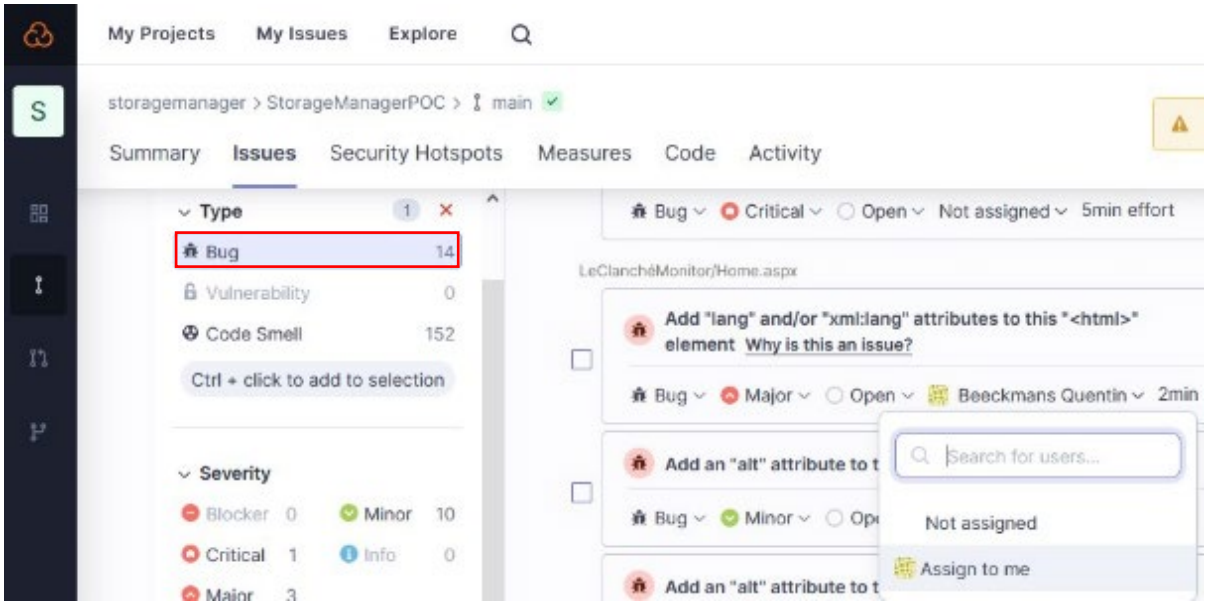
Figure 83 - Résultat de la première analyse SonarCloud

Source : Données de l'auteur

Lien : https://sonarcloud.io/organizations/storagemanager/projects

Dans ce rapport SonarCloud a identifié 14 bugs :

Figure 84 - Bug identifiés par SonarCloud



Source : Données de l'auteur

Nous nous attribuons une tâche de correction pour un bug simple : l'attribut langue est manquant dans la balise html du fichier home.aspx .

```
[…]
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
[…]
```

Une fois cette erreur corrigée, l'analyse montre qu'il y a un bug de moins :

Figure 85 - Résultat SonarCloud après la correction du bug

Source : Données de l'auteur

SonarCloud nous a permis de mettre en place des règles d'analyse de code statique protégeant de cette manière notre application en s'assurant de générer du code de qualité et de fournir un retour rapide sur l'état de sécurité de celui-ci.

4.4.4. OWASP ZAP[40] (DAST)

Figure 86 - Pipeline DAST

Source : Données de l'auteur

OWASP ZAP est la dernière étape de test de sécurité, appelé test de sécurité dynamique (ZAP) pour ensuite nous fournir un rapport.

Nous souhaitons effectuer cette analyse sur notre site web déployé précédemment à l'adresse https://appmonitoringstoragemanager-prod.azurewebsites.net .

OWASP ZAP a besoin d'une image Ubuntu[41] dans le pipeline pour fonctionner, nous avons été contraints de sortir l'outil du pipeline principal qui lui demandait un environnement Windows pour fonctionner.

```
pool:
  vmImage: 'ubuntu-20.04'
```

[40] https://www.zaproxy.org/

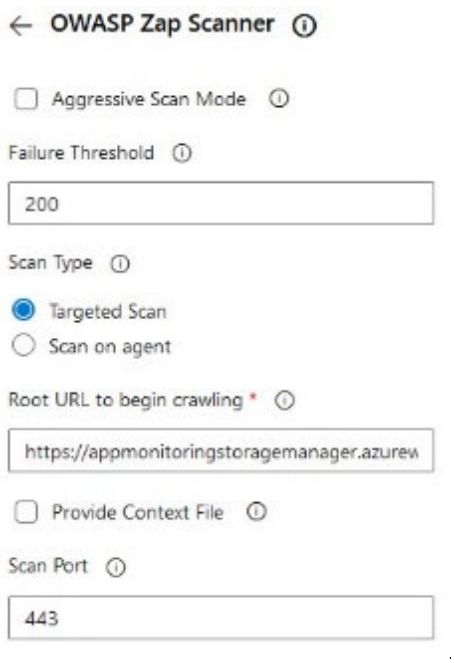
[41] Ubuntu est un système d'exploitation basé sur Linux, il est libre et open-source

Cela reste néanmoins utile car le pipeline peut être exécuté indépendamment du reste. Il peut être exécuté de nuit afin d'être utilisé à son maximum sans déranger les développeurs.

Pour installer OWASP ZAP, il existe un plugin Azure à l'adresse :

https://marketplace.visualstudio.com/items?itemName=CSE-DevOps.zap-scanner

Figure 87 - Tâche OWASP ZAP Scanner



Source : Données de l'auteur

Problème :

Malheureusement, l'analyse avec cette tâche OWASP ZAP n'a pas fonctionné pour les sites hébergés par Azure (.azurewebsites.net).

L'erreur retournée est que le site est inaccessible :

```
Automation plan failures:
Job spider failed to access URL http://appmonitoringstoragemanager.azurewebsites.net:443 : Read timed out

##[error]ENOENT: no such file or directory, open '/home/vsts/work/1/s/owaspzap/report.json'
```

Après de multiples recherches, nous avons constaté que d'autres utilisateurs avaient rencontré ce problème et avaient ouvert un sujet sur le GitHub de l'application OWASP ZAP Scanner pour Azure DevOps. Aucune solution n'a été proposée jusqu'à présent.

https://github.com/microsoft/CSEDevOps/issues

Nous avons effectué des essais sur des sites autres que « azurewebsite » et l'outil fonctionnait correctement comme par exemple sur l'adresse **https://quentinbeeckmans.dev** .

Solution :

D'innombrable tentatives ont été effectuées pour résoudre ce problème via notamment la documentation https://www.zaproxy.org/docs/docker/diagnosing-problems/, nous ne retiendrons pour une meilleur lecture que la solution qui a fonctionné.

Nous avons décidé d'utiliser une image docker en ligne de commande au lieu de la tâche OWASP ZAP présente dans le marketplace pour avoir une plus grande flexibilité dans la configuration.

Documentation OWASP ZAP Docker : https://www.zaproxy.org/docs/docker/baseline-scan/ .

Le premier script permet de préparer les dossiers pour le Docker, un dossier owaspzap avec tous les droits chmod 777 (lecture, écriture et exécution) :

```
- task: CmdLine@2
  displayName: 'Prepare OWASPZAP output directory for Docker'
  inputs:
    script: |
      sudo mkdir $(System.DefaultWorkingDirectory)/owaspzap
      sudo chmod -R 777 $(System.DefaultWorkingDirectory)/owaspzap
```

Le second script :

```
- task: CmdLine@2
  displayName: 'Run OWASPZAP Docker Image and scan https://appmonitoringstoragemanager.azure
websites.net'
  inputs:
    script: 'docker run -t -
v $(System.DefaultWorkingDirectory)/owaspzap:/zap/wrk:rw owasp/zap2docker-stable zap-
baseline.py -t https://appmonitoringstoragemanager.azurewebsites.net/ -J report.json -i -
r report.html --autooff'
#    script: 'docker run -t -
v $(System.DefaultWorkingDirectory)/owaspzap:/zap/wrk:rw owasp/zap2docker-stable zap-full-
scan.py -t https://appmonitoringstoragemanager.azurewebsites.net/ -J report.json -i -r report.html'
```

- **docker run** exécute une image Docker
- **-t** indique à Docker d'allouer un terminal virtuel pour le conteneur qui sera exécuté. Cela permet d'interagir avec le conteneur en utilisant la ligne de commande.

- **-v** indique de monter un volume, répertoire du système de fichiers, dans le conteneur. Cela permet de partager des fichiers. Le volume est créé en utilisant le chemin d'accès $(System.DefaultWorkingDirectory)/owaspzap sur le système hôte et en le mappant au chemin /zap/wrk dans le conteneur. L'option :rw indique que le volume est en lecture-écriture.
- **owasp/zap2docker-stable** est le nom de l'image Docker à exécuter.
- **zap-baseline.py** est le nom du script Python à exécuter dans le conteneur pour une analyse rapide. La version agressive de l'analyse est le script zap-full-scan.py.
  Deux niveaux d'analyses sont donc possibles : une analyse de base exécutant l'araignée[42] ZAP contre la cible pendant une minute. Ce mode est adapté pour une intégration continue. Tandis que l'analyse complète « Agressive » exécute de véritables attaques pendant une longue période. Cette dernière est mise en commentaire (pour ne pas être exécutée) et doit être déclenchée de préférence durant la nuit pour ne pas déranger les développeurs et peut nécessiter d'avoir accès à un temps exécution de pipeline supérieur à une heure.
- **-t https://appmonitoringstoragemanager.azurewebsites.net/** : indique l'URL de l'application à analyser au script python.
- **-J report.json** : indique le nom du fichier JSON dans lequel le rapport doit être enregistré.
- **-i** indique que le script doit effectuer une analyse passive de l'application
- **–r report.html** : indique le nom du fichier HTML dans lequel le rapport doit être enregistré
- **--autooff** : indique que le script doit arrêter ZAP une fois l'analyse terminée.

Le scanner ZAP comprend plusieurs options de rapport sous forme d'artefacts téléchargeables. Pour pouvoir voir le rapport produit par l'analyse Owasp Zap, utilisez ce script à la suite des tâches précédentes :

```
- task: CmdLine@2
  displayName: 'Owasp Nunit Template'
  inputs:
    script: |
      sudo npm install -g handlebars-cmd

        cat <<EOF > owaspzap/nunit-template.hbs
        {{#each site}}
        <test-run
          id="2"
```

[42] L'araignée est un outil qui est utilisé pour découvrir automatiquement de nouvelles ressources (URL) sur un site particulier. Il commence par une liste d'URL visité pour ensuite identifier tous les hyperliens présents dans les pages et propage de cette manière la découverte de nouveaux URL (Spider, 2022).

```
    name="Owasp test"
    start-time="{{../[@generated]}}" >
    <test-suite
      id="{{@index}}"
      type="Assembly"
      name="{{[@name]}}"
      result="Failed"
      failed="{{alerts.length}}">
      <attachments>
        <attachment>
          <filePath>$(Build.SourcesDirectory)/owaspzap/report.html</filePath>
        </attachment>
      </attachments>
    {{#each alerts}}<test-case
      id="{{@index}}"
      name="{{alert}}"
      result="Failed"
      fullname="{{alert}}"
      time="1">
        <failure>
          <message>
            <![CDATA[{{{desc}}}]]>
          </message>
          <stack-trace>
            <![CDATA[
Solution:
{{{solution}}}

Reference:
{{{reference}}}

instances:{{#each instances}}
* {{uri}}
  - {{method}}
  {{#if evidence}}- {{{evidence}}}{{/if}}
              {{/each}}]]>
          </stack-trace>
        </failure>
    </test-case>
    {{/each}}
    </test-suite>
</test-run>
{{/each}}
```

Npm (Node Package Manager) est le gestionnaire de paquets par défaut pour l'environnement d'exécution JavaScript Node.js.

Handlebars-cmd est un module npm qui permet de travailler avec des templates Handlebars. Handlebars est un moteur de template qui permet de générer du HTML en utilisant des expressions et des structures de contrôle de boucle et de condition.

Ensuite, pour générer un rapport au format NUnit[43] permettant sa publication dans l'onglet de test de Azure pipeline, utilisez ce script :

```
- task: CmdLine@2
  displayName: 'Generate NUnit type file'
  inputs:
    script: 'handlebars owaspzap/report.json < owaspzap/nunit-template.hbs > owaspzap/test-results.xml'
```

Et finalement, ajoutez la tâche Publier les résultats du test. Cette tâche permettra d'afficher dans l'onglet Tests les résultats à la fin de notre pipeline :

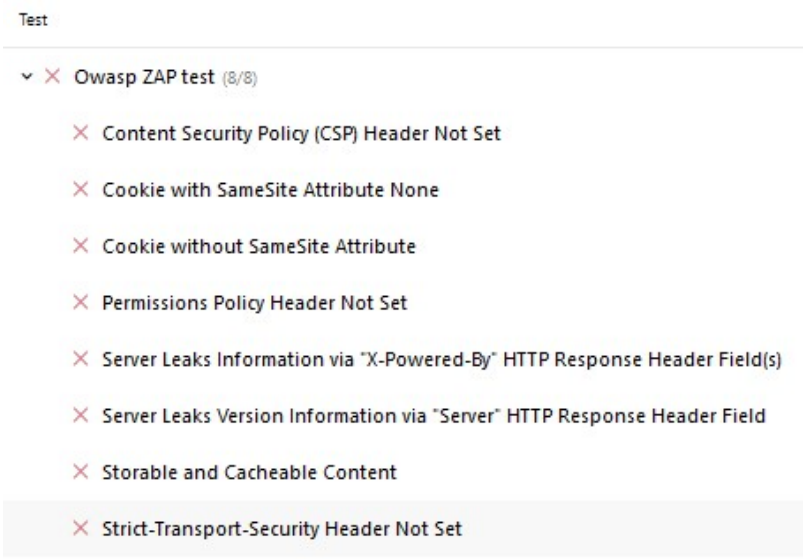
```
- task: PublishTestResults@2
  displayName: 'Publish Test Results'
  inputs:
    testResultsFormat: 'NUnit'
    testResultsFiles: 'owaspzap/test-results.xml'
```

Figure 88 - Résultat du test rapide OWASP ZAP 1/2

Source : Données de l'auteur

[43] NUnit est un framework de test unitaire pour tous les langages .Net. Initialement porté à partir de JUnit.

Figure 89 - Résultat du test OWASP ZAP 2/2



Source : Données de l'auteur

Le rapport ZAP nous fait état de huit erreurs, la première par exemple est le Content Security Policy (CSP) qui n'a pas été défini. C'est une fonctionnalité de sécurité qui indique aux navigateurs les règles de sécurité à respecter lors du chargement de contenu sur le site.

Lorsqu'un CSP n'est pas configuré correctement ou pas du tout, il peut y avoir des risques comme : l'injection de code malveillant, la fuite de données sensibles et la dégradation de la performance du site.

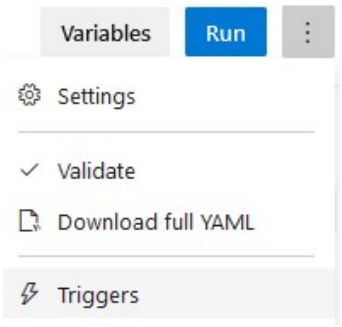
L'exécution du mode agressif nous fait état de deux erreurs supplémentaires :

Figure 90 - Résultat du test agressif OWASP ZAP 2/2

Source : Données de l'auteur

Afin de déclencher le pipeline OWASP ZAP de façon automatique après l'exécution réussie du build SM_OWASPDC_SonarCloud_UnitTest, un déclencheur est configuré dans le menu du pipeline :

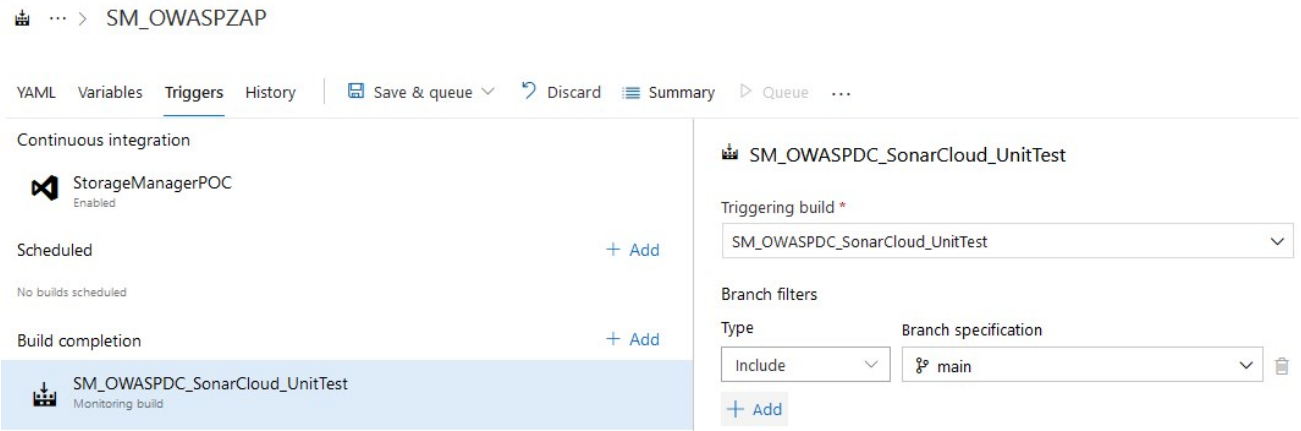
Figure 91 - Menu triggers du pipeline



Source : Données de l'auteur

Il reste ensuite à paramétrer le *Build completion* et le *scheduled* pour que le pipeline se déclenche après le build du pipeline principal et durant la nuit.

Figure 92 - Configuration du déclencheur pour le build OWASP ZAP



Source : Données de l'auteur

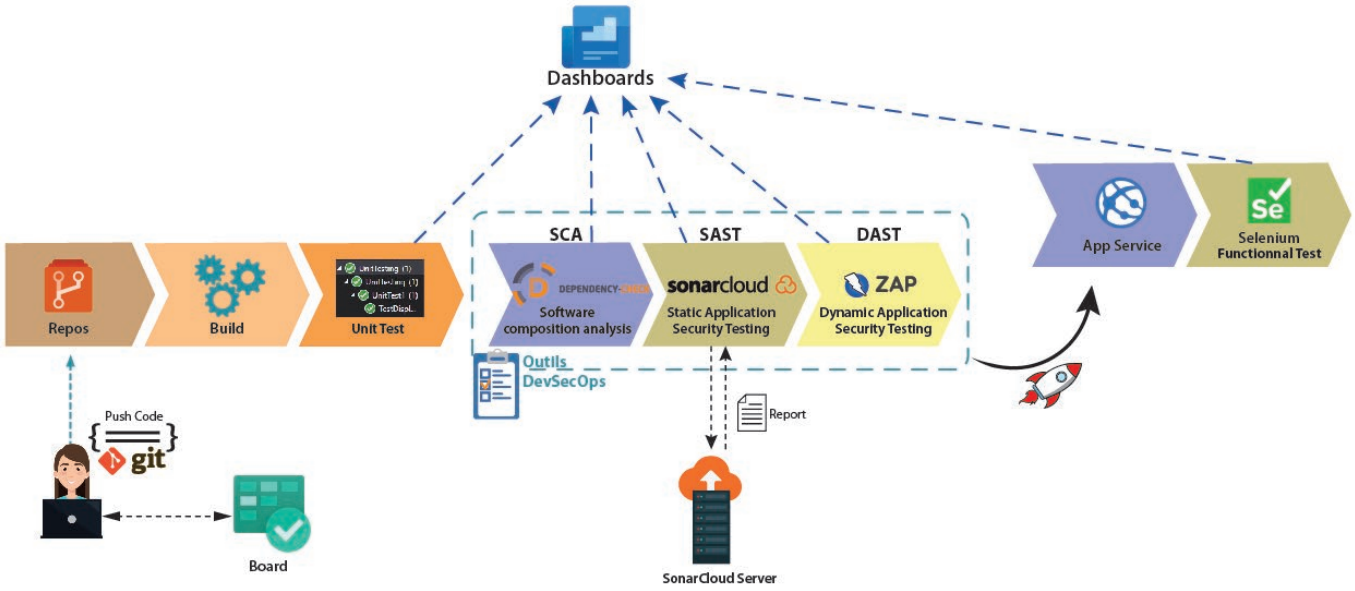
Résultat du déclencheur :

Figure 93 - Programmation du build OWASP ZAP

Source : Données de l'auteur

4.5. Synthèse des choix DevOps et DevSecOps

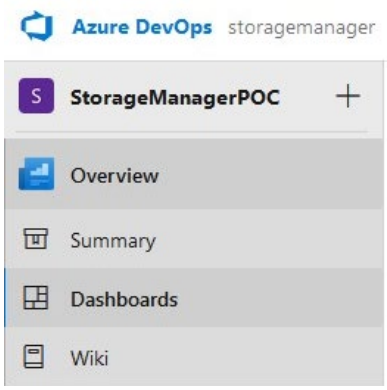
Figure 94 - Pipeline complet DevSecOps



Source : Données de l'auteur

4.6. Dashboard (Surveillance)

Figure 95 - Dashboards



Source : Données de l'auteur

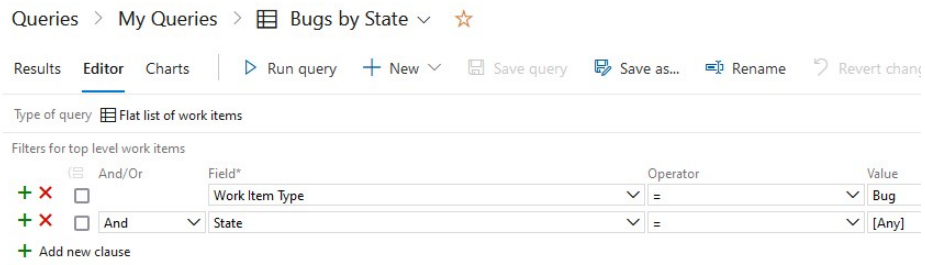
Les tableaux de bord dans Azure DevOps sont des outils de visualisation de données qui permettent de créer des graphiques et widgets personnalisés pour afficher des informations en temps réel sur une équipe ou un projet. Les graphiques sont basés sur des requêtes ou des tendances, tandis que les widgets sont des informations configurables affichées sur les tableaux de bord. Il est également possible d'ajouter des widgets supplémentaires via le marketplace Azure DevOps.

Les rapports, quant à eux, sont des graphiques générés automatiquement par le système pour afficher des informations spécifiques, comme la vitesse d'équipe, le burndown de sprint ou le diagramme de flux cumulé, qui dérivent tous des données d'Analytics.

Azure Dashboards intègre l'affichage de données à l'aide de Power BI[44] (Staheli, et al., 2022).

Par exemple nous souhaitons afficher tous les bugs et leur état à partir d'une requête qui serait:

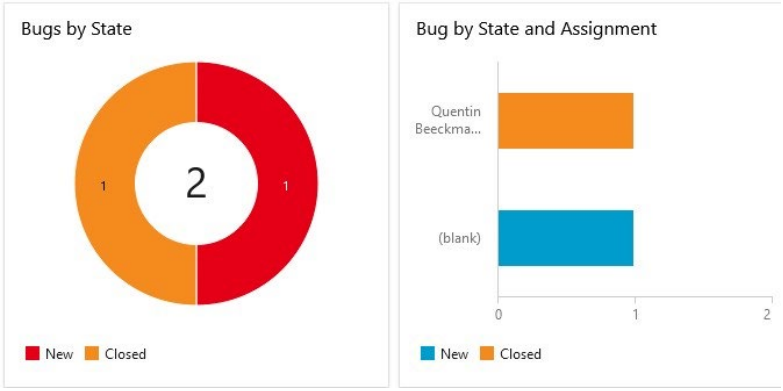
Figure 96 - Requête état de bug



Source : Données de l'auteur

Le résultat donnera (sur deux widgets différents) :

Figure 97 - Résultats de la requête dans Overview > Dashboards



Source : Données de l'auteur

Une liste complète des widgets natifs à Azure DevOps Dashboards se trouve à l'adresse :

https://learn.microsoft.com/fr-ch/azure/devops/report/dashboards/overview?view=azure-devops

Vue complète du Dashboard (voir Annexe X).

[44] Power BI est une plateforme de visualisation de données qui permet de créer des tableaux de bord et des rapports à partir de données. Il est développé par Microsoft.

5. AMELIORATIONS POSSIBLES

Bien que nous ayons atteint nos objectifs, il y a toujours des points qui peuvent être améliorés et de nouvelles fonctionnalités qui peuvent être proposées.

Nous avons retenu ci-dessous des possibilités de développement et d'amélioration de ce projet :

Pipeline parallèle

La création de pipelines parallèles permettrait une réduction du temps de construction et de déploiement. De cette façon il serait possible d'utiliser plus efficacement les ressources de build ou de déploiements disponibles, ce qui permettrait de réduire les coûts et accélérer les pipelines.

Microsoft Security DevOps pour Azure DevOps

Microsoft Security DevOps est une extension pour Azure DevOps qui a été lancé le 11 octobre 2022.

https://marketplace.visualstudio.com/items?itemName=ms-securitydevops.microsoft-security-devops-azdevops&ssr=false#overview

Cette extension permettrait d'utiliser des outils open source très intéressants tels que:

Tableau 9 - Outils open-source Microsoft Security DevOps

Name	Language	License
Bandit	python	Apache License 2.0
BinSkim	binary - Windows, ELF	MIT License
CredScan	code, artifacts	-
ESlint	JavaScript	MIT License
Template Analyzer	Infrastructure-as-code (IaC), ARM templates, Bicep files	MIT License
Terrascan	Infrastructure-as-Code (IaC), Terraform (HCL2), Kubernetes (JSON/YAML), Helm v3, Kustomize, Dockerfiles, CloudFormation	Apache License 2.0
Trivy	Container Images, Infrastructure as Code (Iac)	Apache License 2.0

Source : (Readme, 2022)

Selon la documentation Azure (Krieger, et al., 2023), Defender pour DevOps fournirait aux équipes de sécurité un aperçu de haut niveau des problèmes de sécurité. C'est donc un outil DevSecOps qui aurait été intéressant de pouvoir mettre en place.

CONCLUSION

En conclusion, ce travail de Bachelor nous a permis de découvrir la méthodologie DevSecOps pour répondre à la demande d'automatiser le déploiement du projet storagemanager avec cette méthodologie qui consiste à intégrer les préoccupations de sécurité dès les premières étapes du cycle de développement, afin de garantir la sécurité de l'application tout au long de son cycle de vie.

Nous avons pour cela réalisé un proof of concept qui représente la synthèse des connaissances acquises lors de l'étape de l'état de l'art par laquelle nous avons notamment analysé les différentes solutions proposées par Microsoft Azure. Et cela a été mis en œuvre dans une intégration des outils de DevOps et DevSecOps choisis durant cette étude.

Des tests ont été créés pour améliorer la qualité du code en utilisant des outils qui n'étaient pas présents initialement dans le projet comme les tests unitaires et les tests fonctionnels permettant de s'assurer du bon fonctionnement d'une partie précise de l'application.

En ce qui concerne les outils DevSecOps, nous avons mis en place l'outil OWASP Dependency Check pour effectuer l'analyse de composition de logiciels (SCA). Les tests de sécurité statiques (SAST) ont été réalisés avec l'outil SonarCloud et les tests de sécurité dynamiques ont quant à eux été faits avec l'outil OWASP ZAP.

La plateforme de développement de logiciels, Azure DevOps, avec laquelle nous avons mis en place ce proof of concept, nous a apporté de grandes satisfactions. Celle-ci propose une grande variété d'outils pour la planification, le suivi, la livraison de projet ainsi que pour la gestion de la sécurité. De plus, la documentation est régulièrement mise à jour par une grande communauté ce qui permet de mieux comprendre son fonctionnement et son utilisation.

Pour conclure, il est important de souligner l'importance de l'approche DevSecOps afin de prévenir les vulnérabilités de sécurité dans les applications et les systèmes en les détectant et en les corrigeant avant leur mise en production. Notre proof of concept a permis d'optimiser le processus de développement et de déploiement du storagemanager tout en garantissant une sécurité accrue de l'application.

REFERENCES

(2017). Consulté le 10 07, 2022, sur HES-SO Valais: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.hevs.ch%2Fmedia%2Fdocument%2F2%2Fhes4_campus2017_v4.pdf&psig=AOvVaw3zMJFC7NSzz7vY6eGY2YuZ&ust=1665220294087000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCNjchpTjzfoCFQAAAAAdAAAAABAE

(2022, 11 04). (HESSO) Récupéré sur http://81.13.188.123:81/monitoring/home.aspx

(2022, 11 04). (HESSO) Récupéré sur http://81.13.188.123:81/AdminConsole/Home.aspx

A deeper look at Selenium. (2022, 9 4). Consulté le 12 29, 2022, sur selenium: https://www.selenium.dev/documentation/overview/details/

Alors que les dépenses trimestrielles dans le cloud grimpent à plus de 50 milliards de dollars, Microsoft occupe une place plus importante dans le rétroviseur d'Amazon. (2022, 02 03). Consulté le 11 06, 2022, sur srgresearch: https://www.srgresearch.com/articles/as-quarterly-cloud-spending-jumps-to-over-50b-microsoft-looms-larger-in-amazons-rear-mirror

Andrews, S., Buck, A., Nelson, J., Onaldi, H., & Ghanayem, M. (2022, 10 12). *Qu'est-ce qu'Azure Database pour PostgreSQL ?* Consulté le 10 12, 2022, sur Microsoft: https://learn.microsoft.com/fr-fr/azure/postgresql/single-server/overview

Avis sur Sonar Cloud. (2022, 11). Consulté le 12 01, 2022, sur peerspot: https://www.peerspot.com/products/sonarcloud-reviews#products-show

Azure DevOps Parallelism Request. (2023). Consulté le 01 18, 2023, sur forms.office: https://forms.office.com/pages/responsepage.aspx?id=v4j5cvGGr0GRqy180BHbR63mUWPlq7NEsFZhkyH8jChUMlM3QzdDMFZOMkVBWU5BWFM3SDI2QlRBSC4u

Bououni, R., Kathryn, Danielson, S., Holvoet, S., Jacobs, M., Eskelid, S., & Wilson, C. (2022, 05 10). *Présentation des artefacts Azure*. Consulté le 10 01, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/artifacts/start-using-azure-artifacts?view=azure-devops

Brown, C. (2022, 07 27). *Jenkins or Azure Pipelines?* Consulté le 10 22, 2022, sur Pratikgroup: https://www.praktikgroup.com/jenkins-azure-pipelines/

Buck, A., Liz, C., Gary, M., & Victoria. (2022, 09 22). *Contrôles DevSecOps*. Consulté le 10 17, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-fr/azure/cloud-adoption-framework/secure/devsecops-controls

Bureau de U.S. DEPARTMENT OF ENERGY. (2022, 09 20). *Cybersécurité solaire*. Récupéré sur energy: https://www.energy.gov/eere/solar/solar-cybersecurity

Business solutions. (2022). Consulté le 12 01, 2022, sur sonarsource: https://www.sonarsource.com/plans-and-pricing/#sonarcloud

Chandrasekara, C., & Herath, P. (2020). *Hands-on Azure Pipelines Understanding Continuous Integratuib abd Deployment in Azure DevOps*. Sri Lanka: Apress. doi:https://doi.org/10.1007/978-1-4842-5902-3

Checkmarx CxSAST. (2022). Consulté le 10 17, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=checkmarx.cxsast

Checkmarx SAST contre SonarQube. (2022). Consulté le 10 19, 2022, sur Gartner: https://www.gartner.com/reviews/market/application-security-testing/compare/product/checkmarx-sast-vs-sonarqube

Concepts de base d'Azure Key Vault. (2022). Consulté le 10 01, 2022, sur Microsoft: https://learn.microsoft.com/fr-fr/azure/key-vault/general/basic-concepts

Configurer et payer pour les travaux parallèles. (2022, 10 24). Consulté le 10 27, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-ch/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops&tabs=ms-hosted

Danielson, S., & Santos, B. (2022, 09 08). *Tâche du programme d'installation de l'outil NuGet*. Consulté le 11 10, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/tool/nuget?view=azure-devops

Détails des prix. (2022). Consulté le 11 07, 2022, sur sonarqube: https://www.sonarqube.org/trial-request/developer-edition/

DevOps. (2022). Consulté le 10 12, 2022, sur Atlassian: https://www.atlassian.com/fr/devops

Duvall, P. M., Matyas, S., & Gépver, A. (2007). *Continuous Integration*. Boston: Addison-Wesley.

Geeksforgeeks. (2022, 06 14). Consulté le 10 19, 2022, sur Qu'est-ce que le Proxy Zed Attack ?: https://www.geeksforgeeks.org/what-is-zed-attack-proxy/

Gérer la dette technique avec SonarQube et Azure DevOps . (2022). Consulté le 11 11, 2022, sur azuredevopslabs: https://www.azuredevopslabs.com/labs/vstsextend/sonarqube/#exercise-2-modify-the-build-to-integrate-with-sonarqube

Gursimran, S. (2021, 12 15). *Construire un pipeline de build Microsoft Azure DevOps*. Consulté le 11 06, 2022, sur xenonstack: https://www.xenonstack.com/blog/microsoft-azure-devops-pipeline

Hall, T. (2022). *Pipeline DevOps*. Consulté le 10 12, 2022, sur Atlassian: https://www.atlassian.com/fr/devops/devops-tools/devops-pipeline

Information technique CYBERSÉCURITÉ PUBLIQUE Directives pour une communication sûre avec les installations. (2022). Consulté le 10 06, 2022, sur sma: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj-9_-nscz6AhV1xAIHHccdALEQFnoECBEQAQ&url=https%3A%2F%2Ffiles.sma.de%2Fdownloads%2FCyberSecurity-TI-fr-10.pdf&usg=AOvVaw2pCH7ER0VGMxVosEl0rtUo

Jakob, P. (2019, 07 18). Consulté le 10 13, 2022, sur Medium: https://medium.com/taptuit/the-eight-phases-of-a-devops-pipeline-fda53ec9bba

jeremylong.github. (2022, 10 19). Consulté le 11 10, 2022, sur Assembly Analyzer: https://jeremylong.github.io/DependencyCheck/analyzers/assembly-analyzer.html

Julia, K.-M., Kathryn, Steve, D., Dennis, R., Shannon, L., Bryan, S., & Ray, A. (2022, 10 05). *Qu'est-ce qu'Azure Pipelines ?* Consulté le 09 30, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-fr/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops

Julia, K.-M., William, A. R., Kathryn, Danielson, S., Jeff, B., & Manuel, G. (2022, 09 27). *Créer et cibler un environnement*. Récupéré sur learn.microsoft: https://learn.microsoft.com/fr-fr/azure/devops/pipelines/process/environments?view=azure-devops

Kai, P., Wohlin, C., & Baca , D. (2009). *Le modèle en cascade dans le développement à grande échelle*. (G. Translate, Trad.) Springer-Verlag Berlin Heidelberg: Fraunhofer IIESE. doi:10.1007/978-3-642-02152-7_29

Kathryn, Leavitt, S., Casey, L., Mabee, D., Mutta, E., Coulter, D., . . . Jackson-South, J. (2022, 10 04). *Qu'est-ce qu'Azure Boards ?* Consulté le 09 29, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-fr/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops

Kathryn, Mabee, D., Casey, L., Coulter, D., Sharkey, K., Jacobs, M., . . . Danielson, S. (2022, 04 10). *Choisissez un flux de processus ou un modèle de processus pour travailler dans Azure Boards*. Consulté le 10 27, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-

ch/azure/devops/boards/work-items/guidance/choose-process?view=azure-devops&tabs=agile-process

Kathryn, Mabee, D., Casey, L., Petersen, T., Jacobs, M., & Hellem, D. (2022, 04 10). *Importer ou mettre à jour des éléments de travail en masse à l'aide de fichiers CSV dans Azure Boards*. Consulté le 11 26, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-ch/azure/devops/boards/queries/import-work-items-from-csv?view=azure-devops

Kathryn, Steve, D., Rahul, J., David, C., Dennis, L., Mike, J., & Alex, H. (2022, 04 10). *Qu'est-ce qu'Azure Test Plans ?* Consulté le 10 01, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/test/overview?view=azure-devops#exploratory-testing

Kent Beck, C. A. (2004). *Extreme Programming Explained: Embrace Change*. (A. W. Professional, Éd.) Consulté le 11 24, 2022

Krieger, E., Mansheim, B., Mel, Spiller, L., Downer, R., Buck, A., & Powers, B. (2023, 01 04). *Quoi de neuf dans Microsoft Defender pour Cloud ?* . Récupéré sur learn.microsoft: https://learn.microsoft.com/en-us/azure/defender-for-cloud/release-notes#defender-for-devops-preview

Les installations solaires ont augmenté de 43% en 2021. (2022, 07 14). Consulté le 10 06, 2022, sur TdG: https://www.tdg.ch/les-installations-solaires-ont-augmente-de-43-en-2021-784169600736

Lin, C., Sangapu, M., Selig, J., Fox, P., Gailey, G., Tardif, B., . . . Keller, L. (2022, 10 22). *Configurer une application App Service*. Consulté le 01 10, 2023, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/app-service/configure-common?tabs=portal

Machiraju, S., & Gaurav, S. (2018). *DevOps for Azure Applications*. Washington: Apress.

Machiraju, V., Sherer, T., Kathryn, Jagtap, R., Danielson, S., Petersen, T., . . . Jackson, S. (2022, 09 27). *Qu'est-ce que Azure Repos ?* Consulté le 09 30, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-fr/azure/devops/repos/get-started/what-is-repos?view=azure-devops

Maxim, M., Kathryn, Theano, P., Rahul, J., Shannon, L., Christopher, M., . . . Sam, G. (2022, 10 19). *Choisir le bon contrôle de version pour votre projet*. Consulté le 10 28, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/comparison-git-tfvc?view=azure-devops

Mend Bolt (formerly WhiteSource). (2022). Consulté le 10 27, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=whitesource.whiteSource-bolt-v2

Micro Focus Fortify. (2022). Consulté le 10 27, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=fortifyvsts.hpe-security-fortify-vsts#wi

Microfocus. (2022). Consulté le 10 19, 2022, sur Fortifier WebInspect: https://www.microfocus.com/pnx/media/data-sheet/webinspect_automated_dynamic_application_security_testing_ds.pdf

Milindanath, H. (2019). *A pratical guide to Azure DevOps*. Learn by doing. Consulté le 11 17, 2022

Modèle en cascade générique. (2019, 06 01). Consulté le 10 27, 2022, sur commons.wikimedia: https://commons.wikimedia.org/w/index.php?curid=79394662

Nair, R., Bououni, R., Danielson, S., Kathryn, Jacobs, M., bansal, s., . . . Staheli, D. (2022, 08 23). *Versions dans Azure Pipelines*. Consulté le 10 15, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/releases?view=azure-devops

Nair, R., Bououni, R., Danielson, S., Kulla-Mader, J., Jacobs, M., Kathryn, . . . Staheli, D. (2022, 10 29). *Provisionner des groupes de déploiement*. Consulté le 11 13, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/deployment-groups/?view=azure-devops

Nair, R., Danielson, S., Kulla-Mader, J., Kathryn, Simpson, D., Alberts, M., . . . Toliver, K. (2022, 04 20). *Groupes de tâches pour les builds et les versions (classique)*. Consulté le 11 13, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/task-groups?view=azure-devops

Nair, R., Downer, R., Bououni, R., Danielson, S., Kathryn, Petersen, T., & Jacobs, M. (2022, 09 27). *Déclencheurs de mise en production*. Consulté le 10 15, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-ch/azure/devops/pipelines/release/triggers?view=azure-devops

Nana, J. (2022, 07 19). *Azure DevOps Tutorial for Beginners | CI/CD with Azure Pipelines*. Consulté le 11 03, 2022, sur youtube: https://www.youtube.com/watch?v=4BibQ69MD8c

Noiuamane, C. (2021, 03 16). *DevOps ou DevSecOps : pourquoi sont-ils importants aujourd'hui et comment s'intègrent-ils ensemble ?* Consulté le 10 27, 2022, sur journaldunet:

https://www.journaldunet.com/solutions/dsi/1498661-devops-ou-devsecops-pourquoi-sont-ils-importants-aujourd-hui-et-comment-s-integrent-ils-ensemble/

OWASP Dependency Check. (2022). Consulté le 10 27, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=dependency-check.dependencycheck

OWASP Top Ten. (2021). Consulté le 10 27, 2022, sur owasp: https://owasp.org/www-project-top-ten/

OWASP ZAP Scanner. (2022). Consulté le 10 27, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=CSE-DevOps.zap-scanner

Penot, A. (2021, août 20). DevSecOps on the Hotmaps Cloud Plateform (DigitalOcean). *Travail de Master*. Neuchâtel, Suisse.

Pittet , S. (2022). *Intégration continue vs livraison vs déploiement*. Consulté le 10 12, 2022, sur atlassian: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Présentation d'Azure DevOps. (2022, 10 11). Consulté le 09 27, 2022, sur learn.microsoft.com: https://learn.microsoft.com/fr-fr/azure/devops/user-guide/what-is-azure-devops?toc=%2Fazure%2Fdevops%2Fget-started%2Ftoc.json&bc=%2Fazure%2Fdevops%2Fget-started%2Fbreadcrumb%2Ftoc.json&view=azure-devops

Qu'est-ce qu'Azure Active Directory. (2022, 09 27). Consulté le 10 01, 2022, sur Microsoft: https://learn.microsoft.com/fr-fr/azure/active-directory/fundamentals/active-directory-whatis

Qu'est-ce que la livraison continue ? (s.d.). Consulté le 10 12, 2022, sur AWS: https://aws.amazon.com/fr/devops/continuous-delivery/

Qu'est-ce que la méthode eXtreme Programming ? (2017, 08 08). Consulté le 11 24, 2022, sur planzone: https://www.planzone.fr/blog/quest-ce-que-la-methodologie-extreme-programming

Qu'est-ce que l'approche CI/CD ? (2018, 04 12). Consulté le 10 12, 2022, sur redhat: https://www.redhat.com/fr/topics/devops/what-is-ci-cd

Readme. (2022, 10 19). Récupéré sur github: https://github.com/microsoft/security-devops-azdevops

Simon, R. (2021, 03 30). *SAST, DAST, SCA : qu'est-ce qui convient le mieux aux tests de sécurité des applications ?* Consulté le 10 17, 2022, sur outpost24: https://outpost24.com/blog/SAST-DAST-SCA-what-s-best-for-application-security-testing

sonarcloud. (2022). Consulté le 12 07, 2022, sur Analyze with Azure Pipelines: https://sonarcloud.io/project/configuration?id=Storagemanager_StorageManagerPOC&analysisMode=AzurePipe

SonarCloud. (2022, 11 14). Consulté le 12 01, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarcloud

SonarQube. (2022). Consulté le 10 17, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarqube&ssr=false#overview

Spider. (2022). Consulté le 12 04, 2022, sur zaproxy: https://www.zaproxy.org/docs/desktop/addons/spider/

Staheli, D., Kathryn, Ovhal, P., Parente, J., Kramer, J., Jacobs, M., . . . Boer, G. (2022, 11 17). *À propos des tableaux de bord, des graphiques, des rapports, & des widgets*. Consulté le 01 03, 2023, sur learn.microsoft: https://learn.microsoft.com/fr-ch/azure/devops/report/dashboards/overview?view=azure-devops

Steve, D., Buck, A., Kathryn, haibo, z., Dan, M., Kristine, T., . . . Shayki, A. (2022). *Agents Azure Pipelines*. Consulté le 10 01, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser

Tarification Azure DevOps. (2022). Consulté le 09 30, 2022, sur Azure: https://azure.microsoft.com/fr-fr/pricing/details/devops/azure-devops-services/

Tarification Azure DevOps. (2023). Consulté le 01 10, 2023, sur azure.microsoft: https://azure.microsoft.com/fr-fr/pricing/details/devops/azure-devops-services/

Tarification Azure Monitor. (2022). Consulté le 10 19, 2022, sur Microsoft: https://azure.microsoft.com/fr-fr/pricing/details/monitor/

Télécharger SonarQube. (2022). Consulté le 11 07, 2022, sur sonarqube: https://www.sonarqube.org/downloads/

Tests OWASP Dependency Check. (2022, 11 10). Récupéré sur dev.azure: https://dev.azure.com/storagemanager/StorageManagerPOC/_build/results?buildId=106&vi

ew=ms.vss-test-web.build-test-results-tab&runId=22&resultId=100040&paneView=attachments

Tom, F., Diana, R., Shohei, M., Allen, S., Raphael, B., David, S., . . . Carl, R. (2022, 11 11). *Qu'est-ce qu'Azure Resource Manager ?* Consulté le 11 11, 2022, sur learn.microsoft: https://learn.microsoft.com/fr-ch/azure/azure-resource-manager/management/overview#resource-groups

Tristan, C. (2022). *CYBER SECURITE ET ENERGIES VERTES*. Consulté le 10 06, 2022, sur CYBERCOVER: https://www.cyber-cover.fr/cyber-documentation/cyber-securite/cyber-securite-et-energies-vertes

Veracode. (2022). Consulté le 10 27, 2022, sur marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=Veracode.veracode-vsts-build-extension

Vijay, M., Steve, D., Eric, M., Julia, K.-M., & Diana, I. (2022, 08 09). *Utiliser des fichiers sécurisés*. Consulté le 10 15, 2022, sur learn.microsoft: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/secure-files?view=azure-devops

Visual Studio. (2022, 10 01). Consulté le 09 29, 2022, sur wikipedia: https://en.wikipedia.org/wiki/Visual_Studio#Azure_DevOps_Services

Vivekanand, R. (2020, 10 28). *DevSecOps with Azure DevOps*. Consulté le 10 22, 2022, sur dev.to: https://dev.to/vivekanandrapaka/devsecops-with-azure-devops-32ho

Wannier, D. (2017). *DevSecOps for Microsoft .net*. Information de thèse de Bachelor, Sierre. doi:FO.2.2.02.28.EC

Wheeler, S., Smatlak, D., Buck, A., Hitchcock, A. M., Berry, D., Maertens, D., . . . Keller, L. (2022, 10 12). *Présentation d'Azure Cloud Shell*. Consulté le 11 12, 2022, sur learn.microsoft: https://learn.microsoft.com/en-in/azure/cloud-shell/overview

Wilson, G. (2020). *DevSecOps A leader's guide to producing secure software without compromising flow, feedback and continuous improvement*. Rethink.

Zaal, S., Demiliani, S., & Malik, A. (2020). *Azure DevOps Explained*. Birmingham: Packt Publishing Ltd. Consulté le 11 17, 2022

Annexe I : Préparation de l'environnement sur Azure

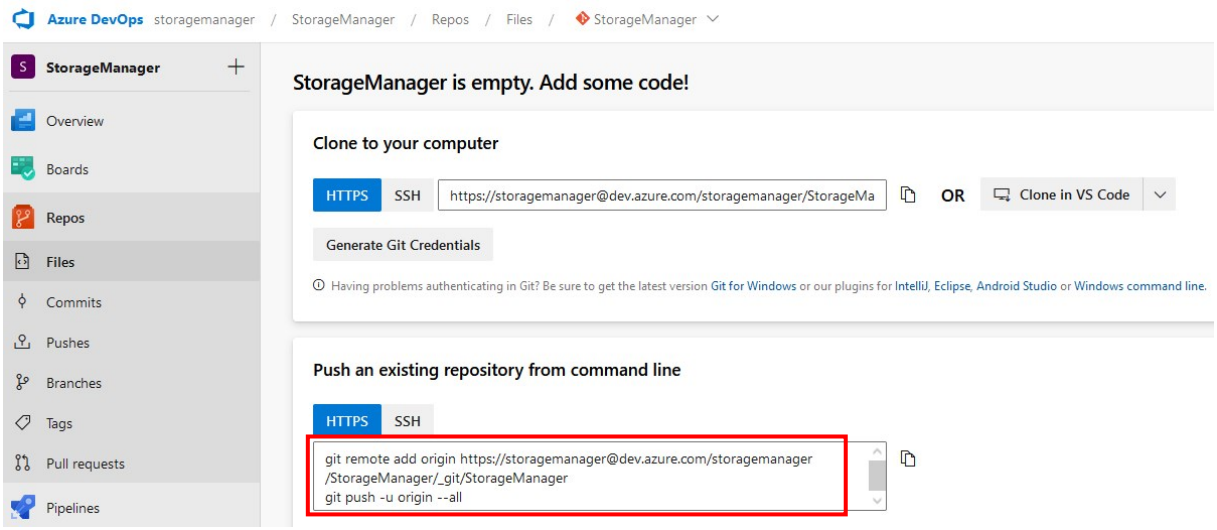
Pour créer un projet Azure DevOps :

1. Créez un compte Azure DevOps si vous n'en avez pas déjà un.
2. Connectez-vous à votre compte Azure DevOps.
3. Créez une organisation
4. Cliquez sur le bouton "New Project" se trouvant sur la page de votre organisation.
5. Entrez un nom, une description, la visibilité, le version contrôle et le processus pour votre projet, puis cliquez sur "Créer".
6. Vous pourrez ensuite configurer les paramètres de votre projet, tel qu'ajouter des utilisateurs et autorisations dans Organization settings.

Pour mettre le code de votre ordinateur sur le dépôt Repos de Azure :

1. Installer Git si cela n'est pas fait
2. Ouvrez un terminal Git et naviguez jusqu'à l'emplacement de votre projet sur votre ordinateur
3. Initialisez un dépôt Git local en exécutant la commande
   ```
   git init.
   ```
4. Ajoutez tous les fichiers de votre projet au dépôt local en utilisant la commande
   ```
   git add .
   ```
5. Effectuez un "commit" en utilisant la commande
   ```
   git commit -m "Votre message de commit"
   ```
6. Rendez-vous dans l'onglet Repos de votre projet Azure
7. Copiez la commande HTTPS (voir image suivante) et collez là dans votre terminal Git
8. Envoyez votre code vers Azure Repos en utilisant la commande
   ```
   git push
   ```

Figure 98 - Commande git à utiliser pour envoyer son code sur Azure



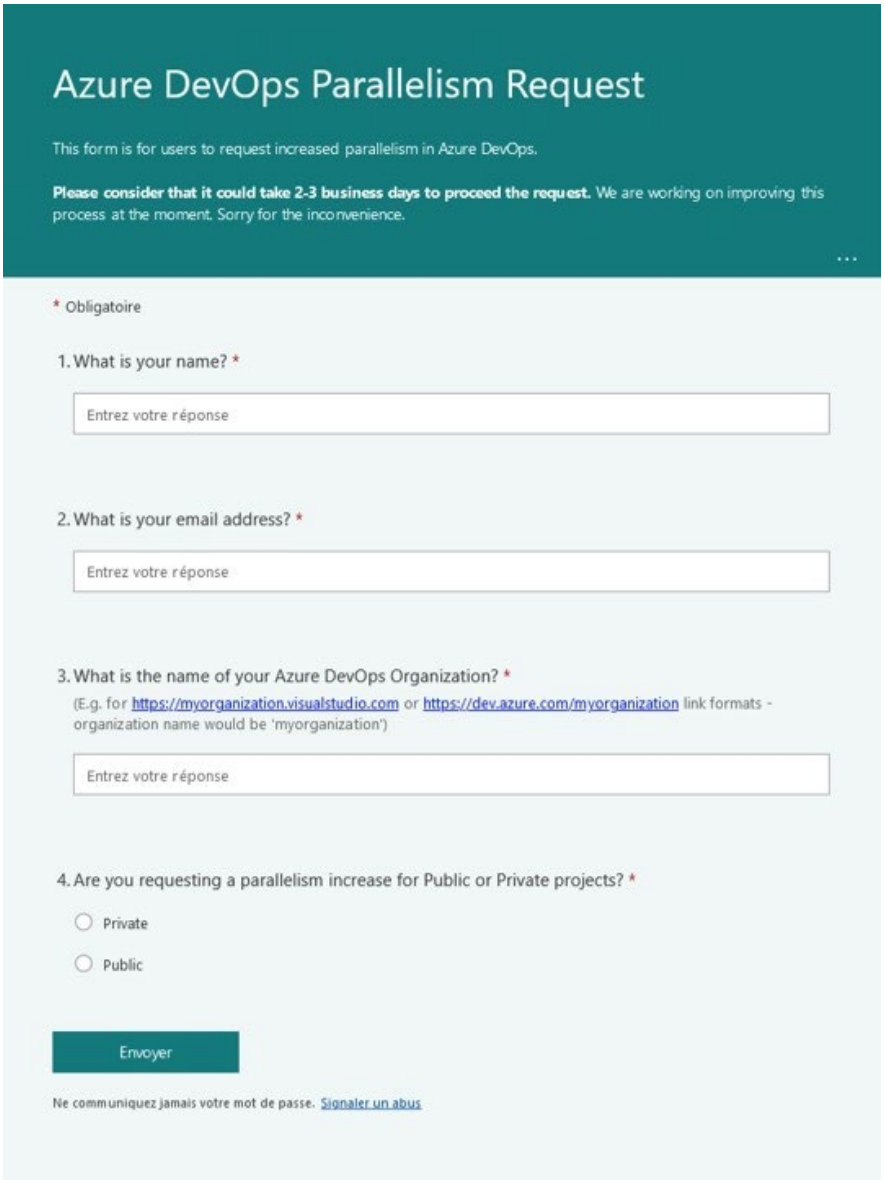
Source : Données de l'auteur

Annexe II : Procédure d'obtention de l'autorisation d'exécuter un pipeline gratuit sur un pool d'agent hébergé par Microsoft

Azure DevOps Parallelism Request. (2023). Récupéré sur forms.office: https://forms.office.com/pages/responsepage.aspx?id=v4j5cvGGr0GRqy180BHbR63mUWPlq7NEsFZhkyH8jChUMlM3QzdDMFZOMkVBWU5BWFM3SDI2QlRBSC4u

Pour obtenir l'autorisation d'exécuter pour la première fois un pipeline gratuit dans votre projet, il est nécessaire d'en faire la demande via ce formulaire :

Figure 99 - Formulaire de demande



La demande sera alors examinée et validée si tout est en ordre dans les deux à trois jours.

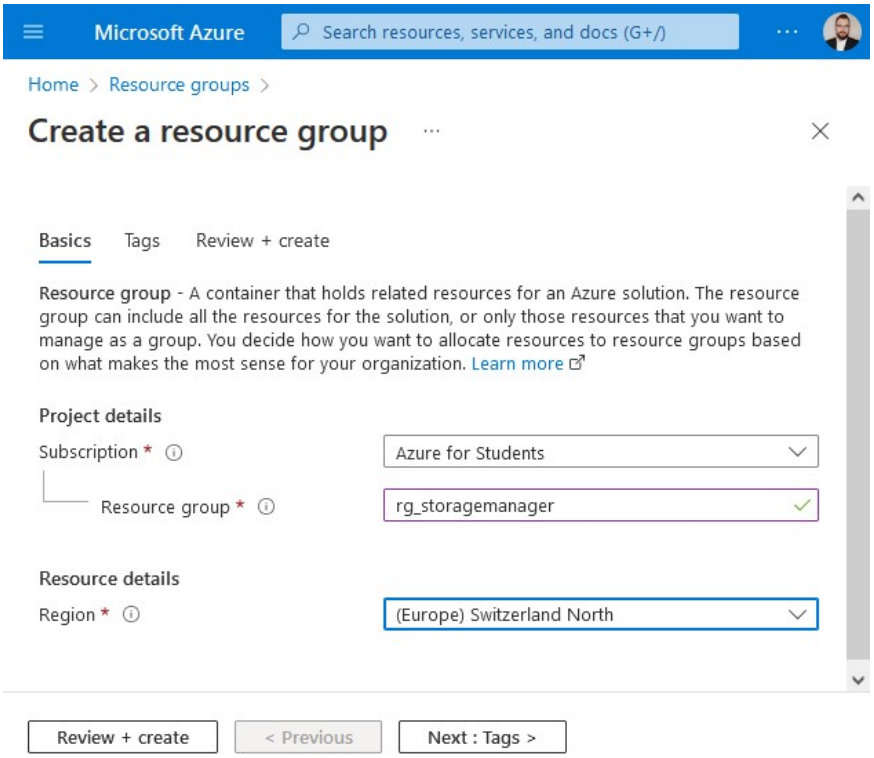
Annexe III : Installation de SonarQube via une instance de container (ACI)

1. Installation via l'interface graphique

Afin d'installer SonarQube sur Azure DevOps il est nécessaire dans un premier temps de créer un groupe de ressource. Un groupe de ressource fournit une couche de gestion permettant de créer, de mettre à jour et de supprimer des ressources.

Depuis Azure > Ressource groups , configurez comme suit :

Figure 100 - Créer un groupe de ressource



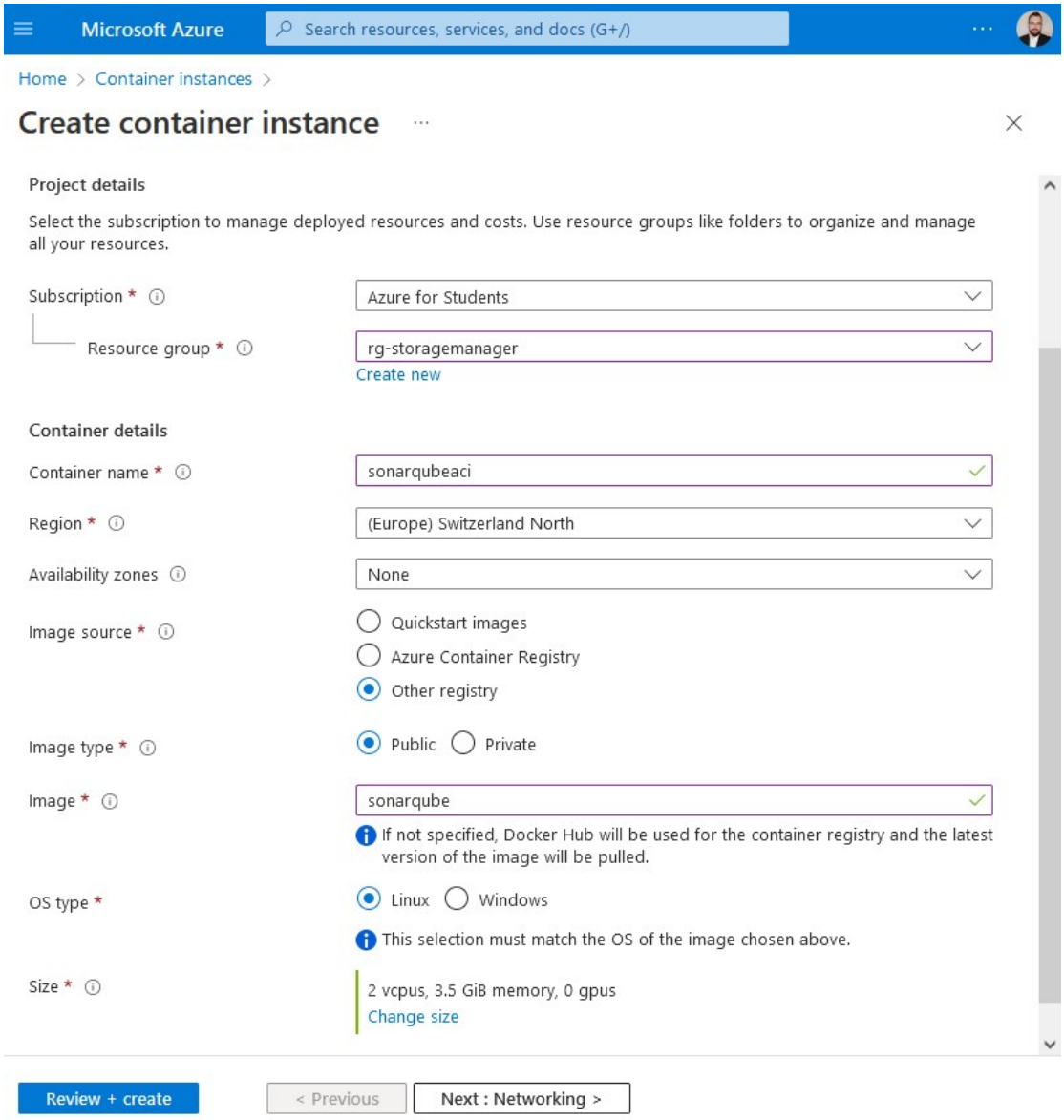
Source : Données de l'auteur

La région indique où les métadonnées sur les ressources sont stockées. Les ressources ont également leur propre région qui peuvent être différentes du groupe de ressource.

Créez ensuite une ressource de type *Container Instances* afin d'héberger et de configurer un container SonarQube. Cette méthode est choisie afin de réduire un maximum les coûts pour un usage ponctuel.

Azure Container Instances (ACI) permet d'exécuter rapidement et facilement des conteneurs sur Azure sans gérer de serveurs ni avoir à apprendre de nouveaux outils. ACI propose une facturation à la seconde pour minimiser le coût d'exécution des conteneurs sur le cloud.

Figure 101 - Configuration Container Instance étape 1/2



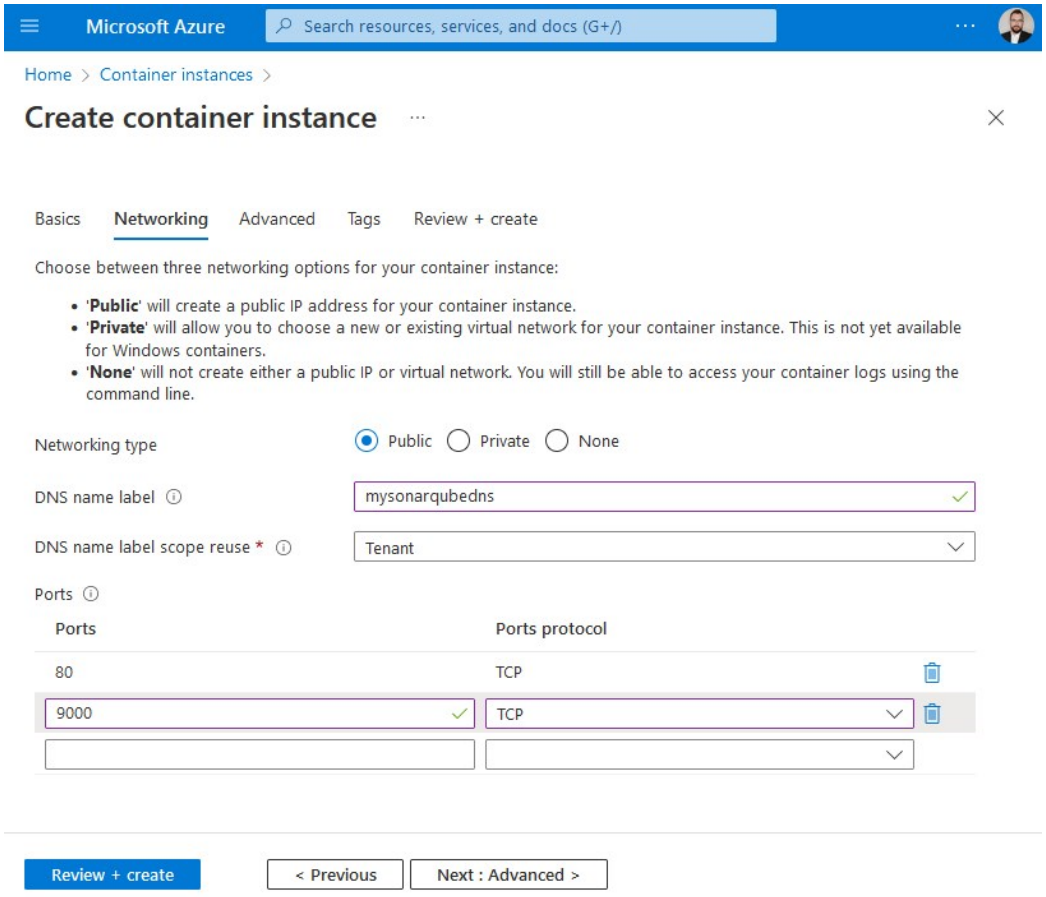
Source : Données de l'auteur

Le nom de l'image correspond au nom figurant sur la plateforme Docker hub :

https://hub.docker.com/_/sonarqube

Pour un fonctionnement correct de SonarQube, il est nécessaire de modifier la taille et de mettre 2 cpu et 3.5 Gio de mémoire. Next.

Figure 102 - Configuration Container Instance étape 2/2



Source : Données de l'auteur

Ajoutez un port 9000 en TCP et passez à l'étape de création 

.

2. Installation en ligne de commande

Source : Wheeler, S., Smatlak, D., Buck, A., Hitchcock, A. M., Berry, D., Maertens, D., . . . Keller, L. (2022, 10 12). *Présentation d'Azure Cloud Shell*. Consulté le 11 12, 2022, sur learn.microsoft: https://learn.microsoft.com/en-in/azure/cloud-shell/overview

Il est également possible d'accélérer le processus en utilisant Azure Cloud Shell qui permet de gérer les ressources Azure en utilisant des commandes Shell. Il existe deux types de commandes Shell dans Azure: les commandes Azure CLI et les commandes PowerShell. Nous utiliserons la première façon. Azure Shell engendre des frais de stockage réguliers.

Accédez au menu via l'onglet :

Figure 103 - Azure Cloud Shell interface

Sources : Données de l'auteur

Il vous sera ensuite demandé de configurer un compte Shell.

Commande de création en Bash d'un groupe de ressource :

```
az group create --name rg-storagemanager --location switzerlandnorth
```

- **--name rg-storagemanager** : spécifie le nom que vous souhaitez donner au groupe de ressources que vous créez.
- **--location switzerlandnorth** : spécifie l'emplacement géographique où vous souhaitez créer le groupe de ressources.

Pour lister tous les emplacements disponibles utilisez la commande :

```
az account list-locations -o table
```

Commande de création de l'ACI :

```
az container create --location switzerlandnorth --resource-group rg-storagemanager --name sonarqubeserver --image sonarqube:9.7.1-community --os-type Linux --ip-address Public --dns-name-label storagemanager --ports 9000 --cpu 2 --memory 3.5
```

- **--resource-group rg-storagemanager** : spécifie le groupe de ressources utilisé.
- **--name sonarqubeserver** : spécifie le nom donné au conteneur.
- **--image sonarqube:9.7.1-community** : est l'image Docker Hub utilisée.
- **--os-type Linux** : spécifie le système d'exploitation linux utilisé pour le conteneur.
- **--ip-address Public** : indique une adresse IP publique pour accéder au conteneur
- **--dns-name-label storagemanager** : est le nom DNS pour accéder au conteneur.
- **--ports 9000** : permet d'ouvrir un port pour accéder au conteneur.
- **--cpu 2** : spécifie le nombre de coeurs CPU allouer au conteneur.
- **--memory 3.5** : spécifie la quantité de mémoire allouer au conteneur.

La ressource est créée :

Figure 104 - Instance d'un container SonarQube

Source : Donnée de l'auteur

Dès que le container est en fonctionnement, accédez au nom de domaine en y ajoutant le port 9000 à la fin de l'url.

Dans notre exemple :
http://sonarqubedns.g5hugwgbhfhjeyfd.switzerlandnorth.azurecontainer.io:9000

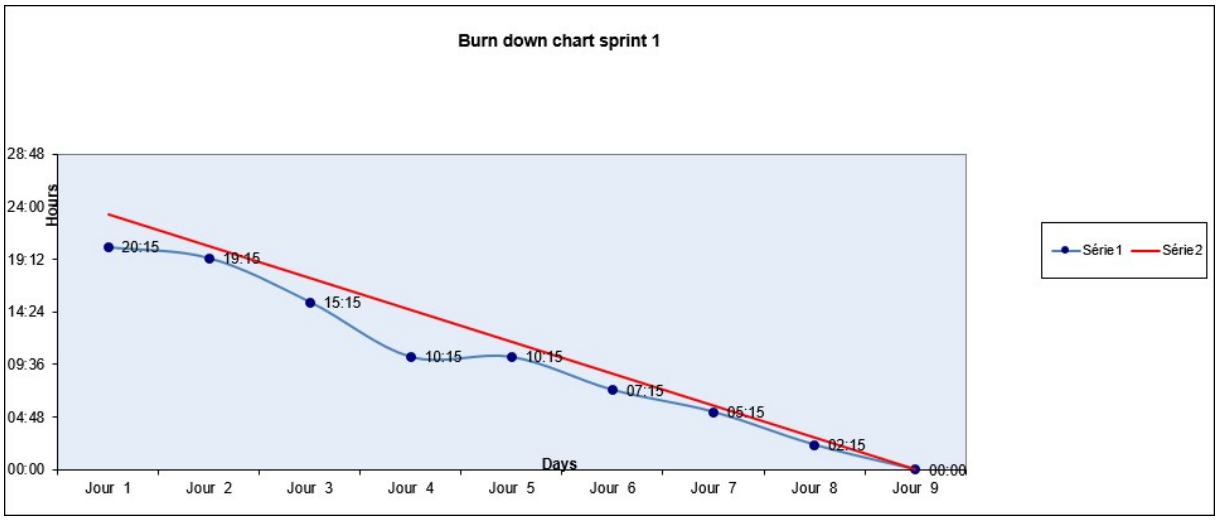
Le nom de compte et de mot de passe par défaut de SonarQube est admin/admin.

Remarque : Cette installation ne permet pas la persistance des données de configuration lorsque l'instance du container sera arrêtée.

Annexe IV : Burndown chart

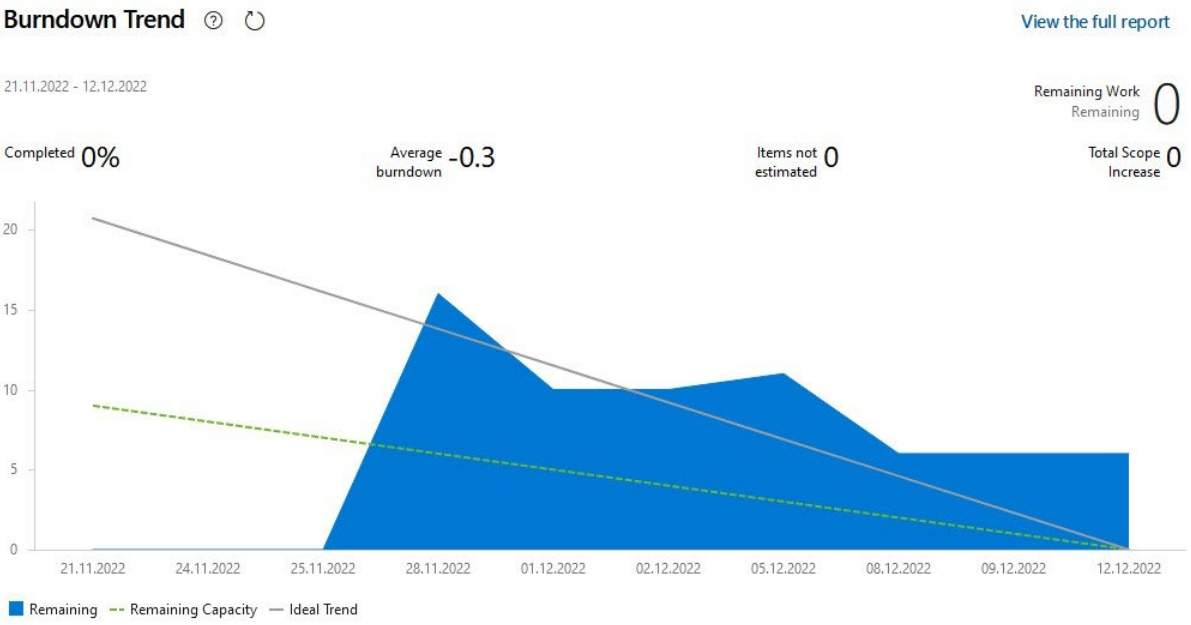
Le premier Sprint du projet a été fait sur Excel, les Sprints suivant ont été fait sur Azure Boards.

Figure 105 - Burndown Chart Sprint 1



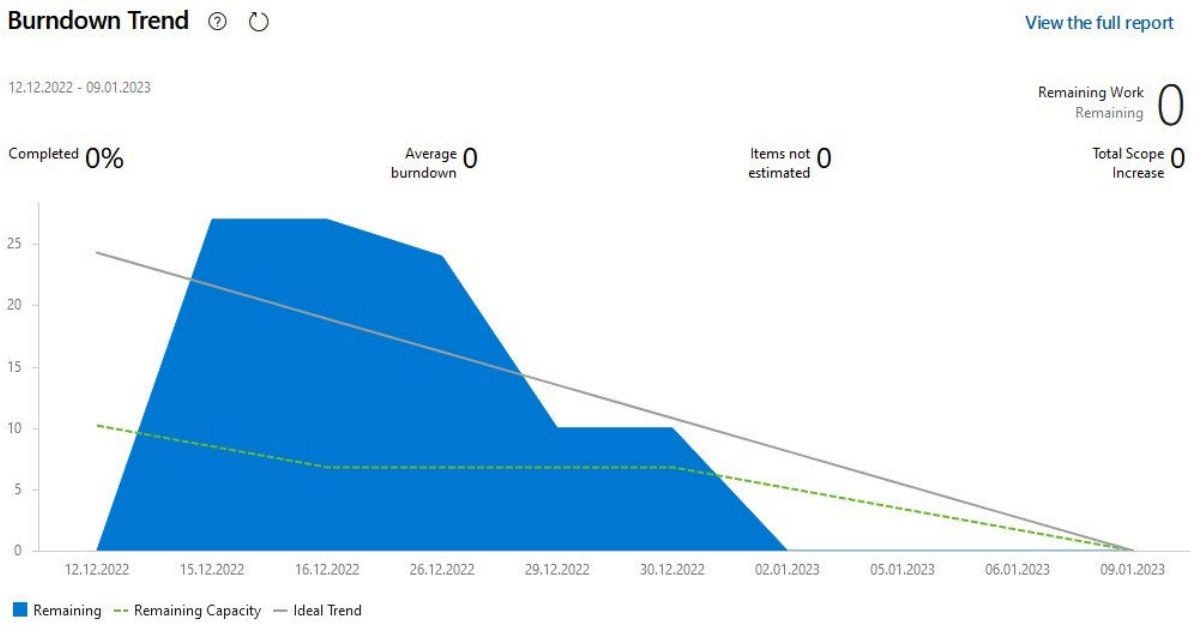
Source : Données de l'auteur

Figure 106 - Burndown Chart Sprint 2



Source : Données de l'auteur

Figure 107 - Burndown Chart Sprint 3



Source : Données de l'auteur

Annexe V : Pipeline YAML OWASP ZAP

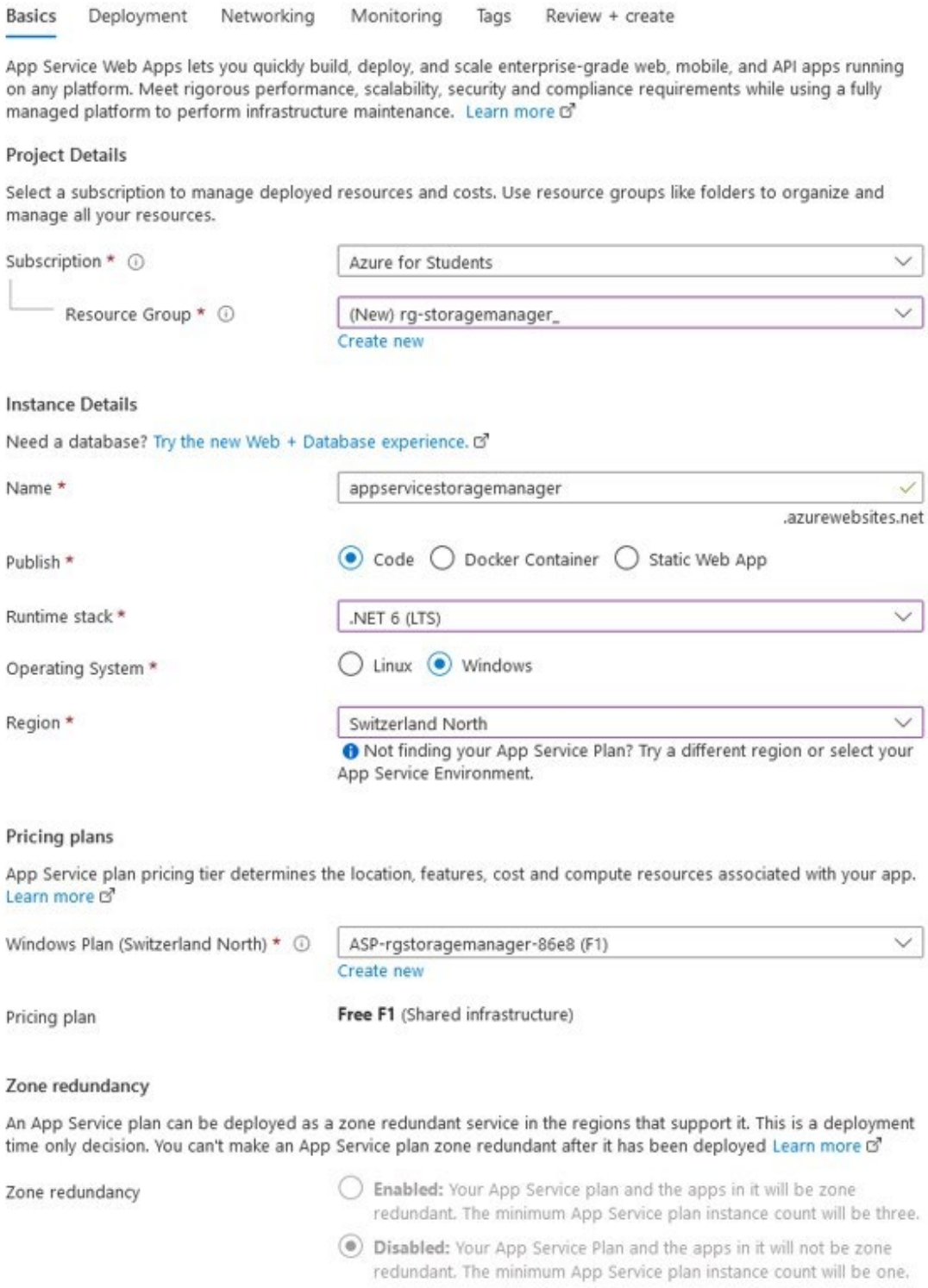
```
trigger:
- none

pool:
  vmImage: 'ubuntu-latest'

steps:
# Tâche de préparation du dossier pour l'analyse OWASP ZAP
- task: CmdLine@2
 displayName: 'Prepare OWASPZAP output directory for Docker'
 inputs:
  script: |
   sudo mkdir $(System.DefaultWorkingDirectory)/owaspzap
   sudo chmod -R 777 $(System.DefaultWorkingDirectory)/owaspzap

# Tâche d'exécution du script docker pour l'analyse OWASP ZAP
- task: CmdLine@2
 displayName: 'Run OWASPZAP Docker Image and scan https://appmonitoringstoragemanager.azure
websites.net'
 inputs:
  script: 'docker run -t -
v $(System.DefaultWorkingDirectory)/owaspzap:/zap/wrk:rw owasp/zap2docker-stable zap-
baseline.py -t https://appmonitoringstoragemanager.azurewebsites.net/ -J report.json -i -
r report.html --autooff'
#   script: 'docker run -t -
v $(System.DefaultWorkingDirectory)/owaspzap:/zap/wrk:rw owasp/zap2docker-stable zap-full-
scan.py -t https://appmonitoringstoragemanager.azurewebsites.net/ -J report.json -i -r report.html'

# Tâche d'exécution du script de template permettant de mettre en forme le résultat du test OWASP
ZAP
- task: CmdLine@2
 displayName: 'Owasp Nunit Template'
 inputs:
  script: |
   sudo npm install -g handlebars-cmd

     cat <<EOF > owaspzap/nunit-template.hbs
     {{#each site}}

     <test-run
       id="2"
```

```
    name="Owasp test"
    start-time="{{../[@generated]}}" >
    <test-suite
      id="{{@index}}"
      type="Assembly"
      name="{{[@name]}}"
      result="Failed"
      failed="{{alerts.length}}">
      <attachments>
        <attachment>
          <filePath>$(Build.SourcesDirectory)/owaspzap/report.html</filePath>
        </attachment>
      </attachments>
    {{#each alerts}}<test-case
      id="{{@index}}"
      name="{{alert}}"
      result="Failed"
      fullname="{{alert}}"
      time="1">
        <failure>
          <message>
            <![CDATA[{{{desc}}}]]>
          </message>
          <stack-trace>
            <![CDATA[
Solution:
{{{solution}}}

Reference:
{{{reference}}}

instances:{{#each instances}}
* {{uri}}
  - {{method}}
  {{#if evidence}}- {{{evidence}}}{{/if}}
              {{/each}}]]>
          </stack-trace>
        </failure>
    </test-case>
    {{/each}}
    </test-suite>
</test-run>
{{/each}}
```

```
# Tâche d'exécution du script de formatage du rapport selon le template
- task: CmdLine@2
  displayName: 'Generate NUnit type file'
  inputs:
    script: 'handlebars owaspzap/report.json < owaspzap/nunit-template.hbs > owaspzap/test-
results.xml'

# Tâche de publication des résultats du test OWASP ZAP
- task: PublishTestResults@2
  displayName: 'Publish Test Results'
  inputs:
    testResultsFormat: 'NUnit'
    testResultsFiles: 'owaspzap/test-results.xml'
    testRunTitle: 'Owasp ZAP test'
    publishRunAttachments: true
```

Annexe VI : Configuration Web App

Figure 108 - Configuration Web App



Source : Donnée de l'auteur

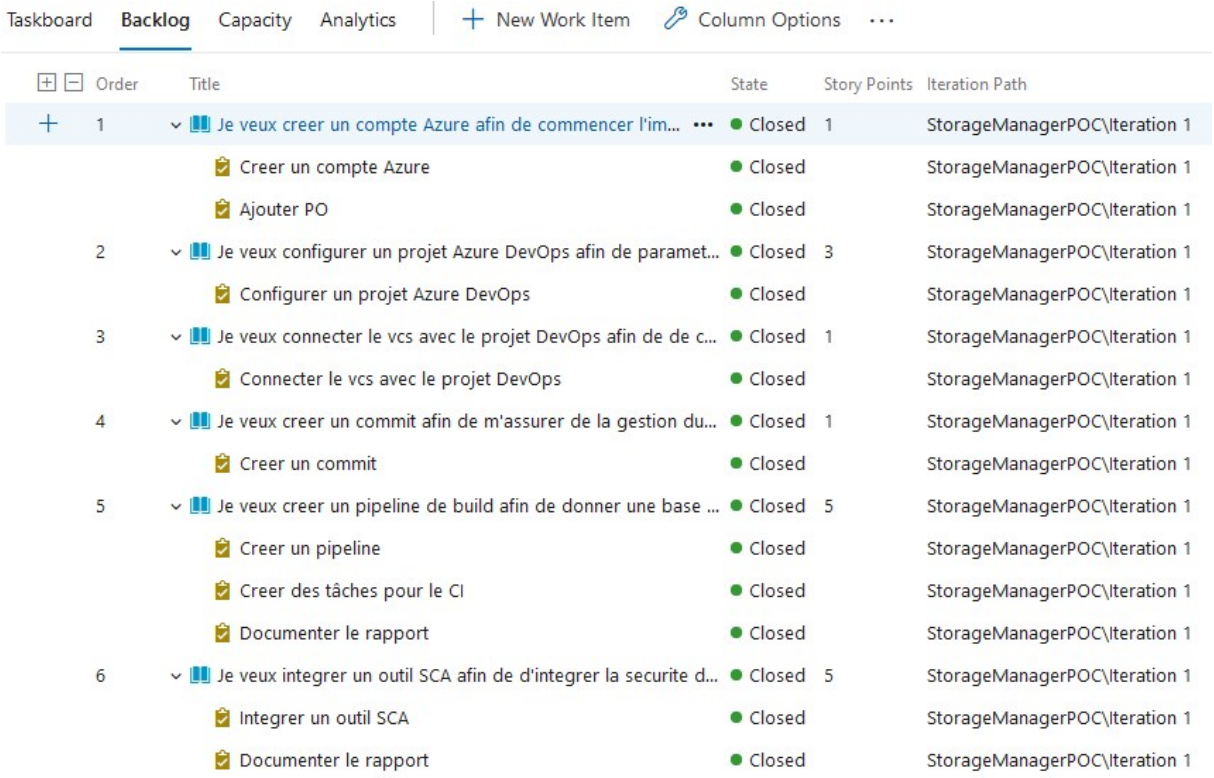
Annexe VII : Work Item

Work Item Type	ID	Title	Sprint
Task	19	Créer un compte Azure	1
Task	26	Ajouter PO	1
Task	27	Configurer un projet Azure DevOps	1
Task	28	Connecter le vcs avec le projet DevOps	1
Task	29	Créer un commit	1
Task	30	Créer un pipeline	1
Task	31	Créer des tâches pour le CI	1
Task	32	Documenter le rapport	1
Task	33	Intégrer un outil SCA	1
Task	34	Documenter le rapport	1
User Story	20	Je veux créer un compte Azure afin de commencer l'implémentation	1
User Story	21	Je veux configurer un projet Azure DevOps afin de paramétrer le projet	1
User Story	22	Je veux connecter le vcs avec le projet DevOps afin de de créer un lien	1
User Story	23	Je veux créer un commit afin de m'assurer de la gestion du versioning et du branching	1
User Story	24	Je veux créer un pipeline de build afin de donner une base de travail	1
User Story	25	Je veux intégrer un outil SCA afin de d'intégrer la sécurité dans le pipeline	1
Task	9	Intégrer un outil SAST	2
Task	10	Documenter le rapport	2
Task	13	Créer un pipeline	2
Task	14	Créer des tâches pour le CD	2
Task	15	Documenter le rapport	2
Task	16	Souscrire à l'offre d'hébergement App Service	2
Task	17	Paramétrer l'app service	2
User Story	1	En tant que Développeur je veux intégrer un outil SAST afin d'intégrer la sécurité dans le pipeline	2
User Story	7	En tant que Développeur je veux créer un pipeline de release afin de déployer l'application	2
User Story	8	En tant que Développeur je veux souscrire à l'offre d'hébergement App Service afin d'en avoir les accès et de le paramétrer	2
Bug	58	PopUpBatteryStateTest Failed in 20230103.3	3
Task	11	Intégrer un outil DAST	3
Task	12	Documenter le rapport	3

Task	36	Activer le Test Plans	3
Task	37	Créer une liste de cas d'utilisation	3
Task	38	Tester le système de Test plan avec la liste de cas d'utilisation	3
Task	40	Créer un test unitaire sur Visual Studio	3
Task	41	Intégrer les test unitaire dans le pipeline CI	3
Task	42	Ajouter le testing unitaire au rapport	3
Task	44	Créer un test fonctionnel automatisé	3
Task	45	Intégrer les recherches dans le rapport sur le test fonctionnel automatisé	3
Task	46	Documenter le rapport	3
Task	47	Intégrer les tests dans le pipeline	3
Test Case	48	En tant qu'utilisateur je veux voir les détails de la batterie en cliquant dessus	3
Test Case	55	Nouveau test	3
User Story	6	En tant que Développeur je veux intégrer un outil DAST afin d'intégrer la sécurité dans le pipeline	3
User Story	35	En tant que Développeur je veux le service Test Plan afin de créer des tests fonctionnels	3
User Story	39	En tant que Développeur je veux effectuer des tests unitaires afin d'intégrer la qualité du code	3
User Story	43	En tant que Développeur je veux intégrer un outil de test fonctionnel automatisé afin d'intégrer le test automatisé au pipeline	3

Annexe VIII : Backlog des Sprints

Figure 109 - Sprint 1

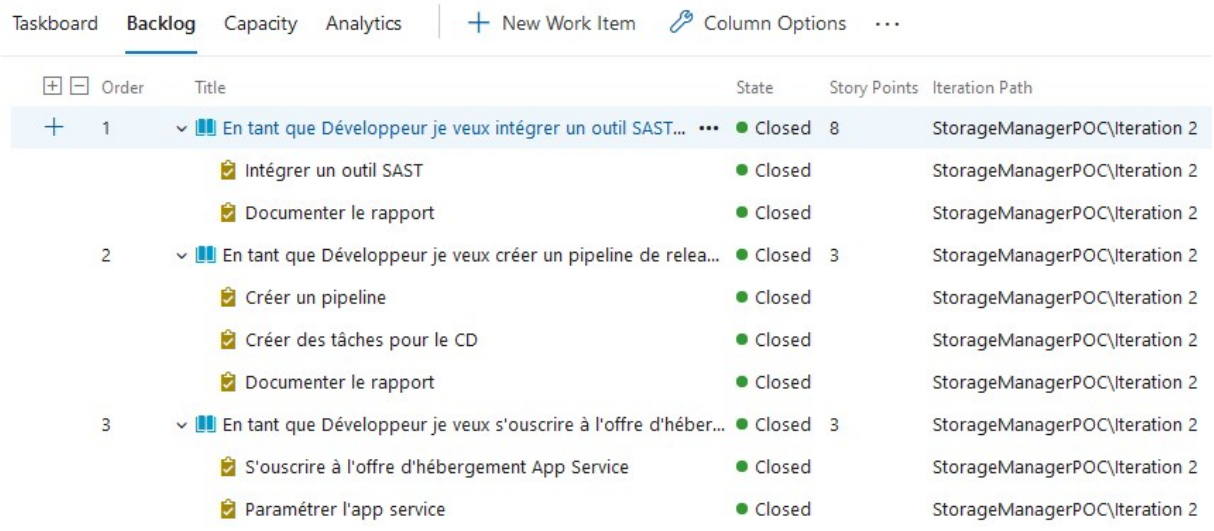


Taskboard | Backlog | Capacity | Analytics | + New Work Item | Column Options | ...

Order	Title	State	Story Points	Iteration Path
1	Je veux creer un compte Azure afin de commencer l'im...	Closed	1	StorageManagerPOC\Iteration 1
	Creer un compte Azure	Closed		StorageManagerPOC\Iteration 1
	Ajouter PO	Closed		StorageManagerPOC\Iteration 1
2	Je veux configurer un projet Azure DevOps afin de paramet...	Closed	3	StorageManagerPOC\Iteration 1
	Configurer un projet Azure DevOps	Closed		StorageManagerPOC\Iteration 1
3	Je veux connecter le vcs avec le projet DevOps afin de de c...	Closed	1	StorageManagerPOC\Iteration 1
	Connecter le vcs avec le projet DevOps	Closed		StorageManagerPOC\Iteration 1
4	Je veux creer un commit afin de m'assurer de la gestion du...	Closed	1	StorageManagerPOC\Iteration 1
	Creer un commit	Closed		StorageManagerPOC\Iteration 1
5	Je veux creer un pipeline de build afin de donner une base ...	Closed	5	StorageManagerPOC\Iteration 1
	Creer un pipeline	Closed		StorageManagerPOC\Iteration 1
	Creer des tâches pour le CI	Closed		StorageManagerPOC\Iteration 1
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 1
6	Je veux integrer un outil SCA afin de d'integrer la securite d...	Closed	5	StorageManagerPOC\Iteration 1
	Integrer un outil SCA	Closed		StorageManagerPOC\Iteration 1
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 1

Source : Données de l'auteur

Figure 110 - Sprint 2

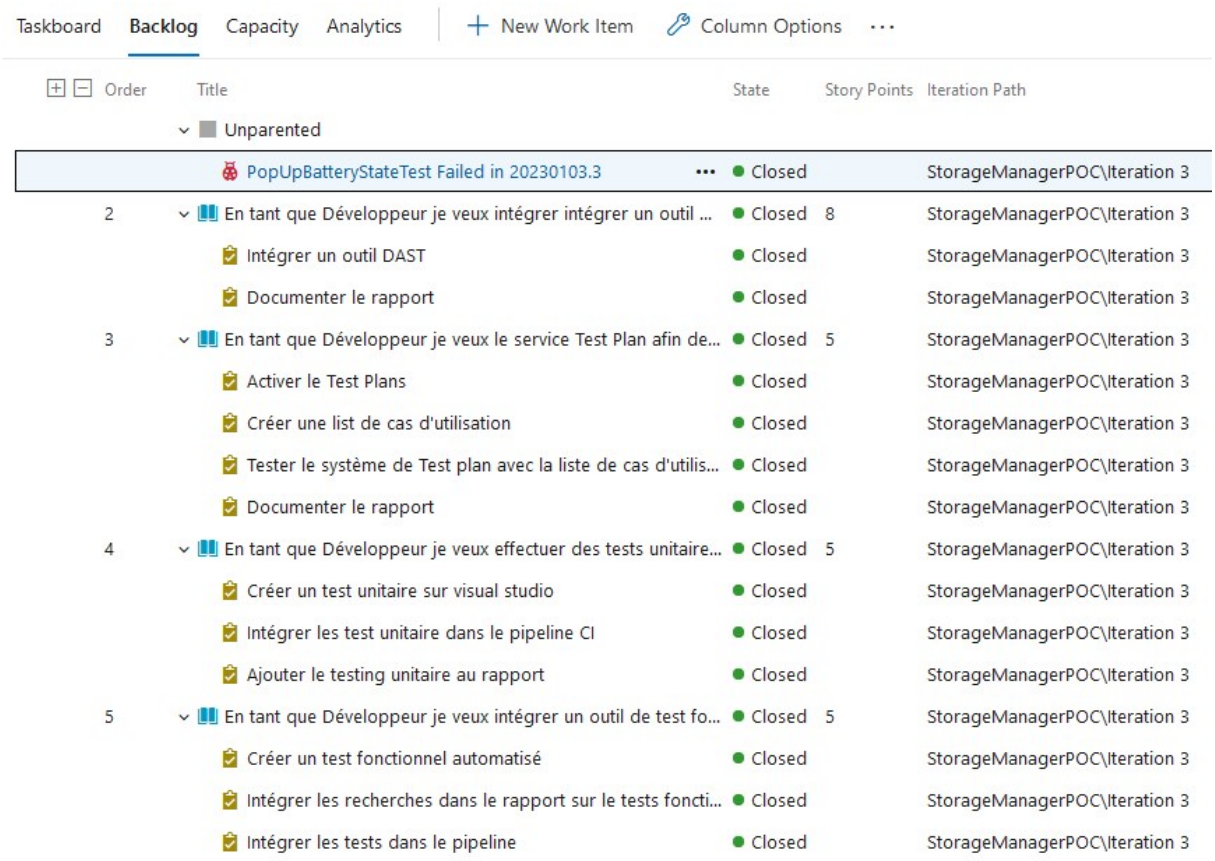


Taskboard | Backlog | Capacity | Analytics | + New Work Item | Column Options | ...

Order	Title	State	Story Points	Iteration Path
1	En tant que Développeur je veux intégrer un outil SAST...	Closed	8	StorageManagerPOC\Iteration 2
	Intégrer un outil SAST	Closed		StorageManagerPOC\Iteration 2
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 2
2	En tant que Développeur je veux créer un pipeline de relea...	Closed	3	StorageManagerPOC\Iteration 2
	Créer un pipeline	Closed		StorageManagerPOC\Iteration 2
	Créer des tâches pour le CD	Closed		StorageManagerPOC\Iteration 2
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 2
3	En tant que Développeur je veux s'ouscrire à l'offre d'héber...	Closed	3	StorageManagerPOC\Iteration 2
	S'ouscrire à l'offre d'hébergement App Service	Closed		StorageManagerPOC\Iteration 2
	Paramétrer l'app service	Closed		StorageManagerPOC\Iteration 2

Source : Données de l'auteur

Figure 111 - Sprint 3



Taskboard | Backlog | Capacity | Analytics | + New Work Item | Column Options | ...

Order	Title	State	Story Points	Iteration Path
	Unparented			
	PopUpBatteryStateTest Failed in 20230103.3	Closed		StorageManagerPOC\Iteration 3
2	En tant que Développeur je veux intégrer intégrer un outil ...	Closed	8	StorageManagerPOC\Iteration 3
	Intégrer un outil DAST	Closed		StorageManagerPOC\Iteration 3
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 3
3	En tant que Développeur je veux le service Test Plan afin de...	Closed	5	StorageManagerPOC\Iteration 3
	Activer le Test Plans	Closed		StorageManagerPOC\Iteration 3
	Créer une list de cas d'utilisation	Closed		StorageManagerPOC\Iteration 3
	Tester le système de Test plan avec la liste de cas d'utilis...	Closed		StorageManagerPOC\Iteration 3
	Documenter le rapport	Closed		StorageManagerPOC\Iteration 3
4	En tant que Développeur je veux effectuer des tests unitaire...	Closed	5	StorageManagerPOC\Iteration 3
	Créer un test unitaire sur visual studio	Closed		StorageManagerPOC\Iteration 3
	Intégrer les test unitaire dans le pipeline CI	Closed		StorageManagerPOC\Iteration 3
	Ajouter le testing unitaire au rapport	Closed		StorageManagerPOC\Iteration 3
5	En tant que Développeur je veux intégrer un outil de test fo...	Closed	5	StorageManagerPOC\Iteration 3
	Créer un test fonctionnel automatisé	Closed		StorageManagerPOC\Iteration 3
	Intégrer les recherches dans le rapport sur le tests foncti...	Closed		StorageManagerPOC\Iteration 3
	Intégrer les tests dans le pipeline	Closed		StorageManagerPOC\Iteration 3

Source : Données de l'auteur

Annexe IX : Pipeline YAML principale (CI, Unit test, SonarCloud, OWASP DC)

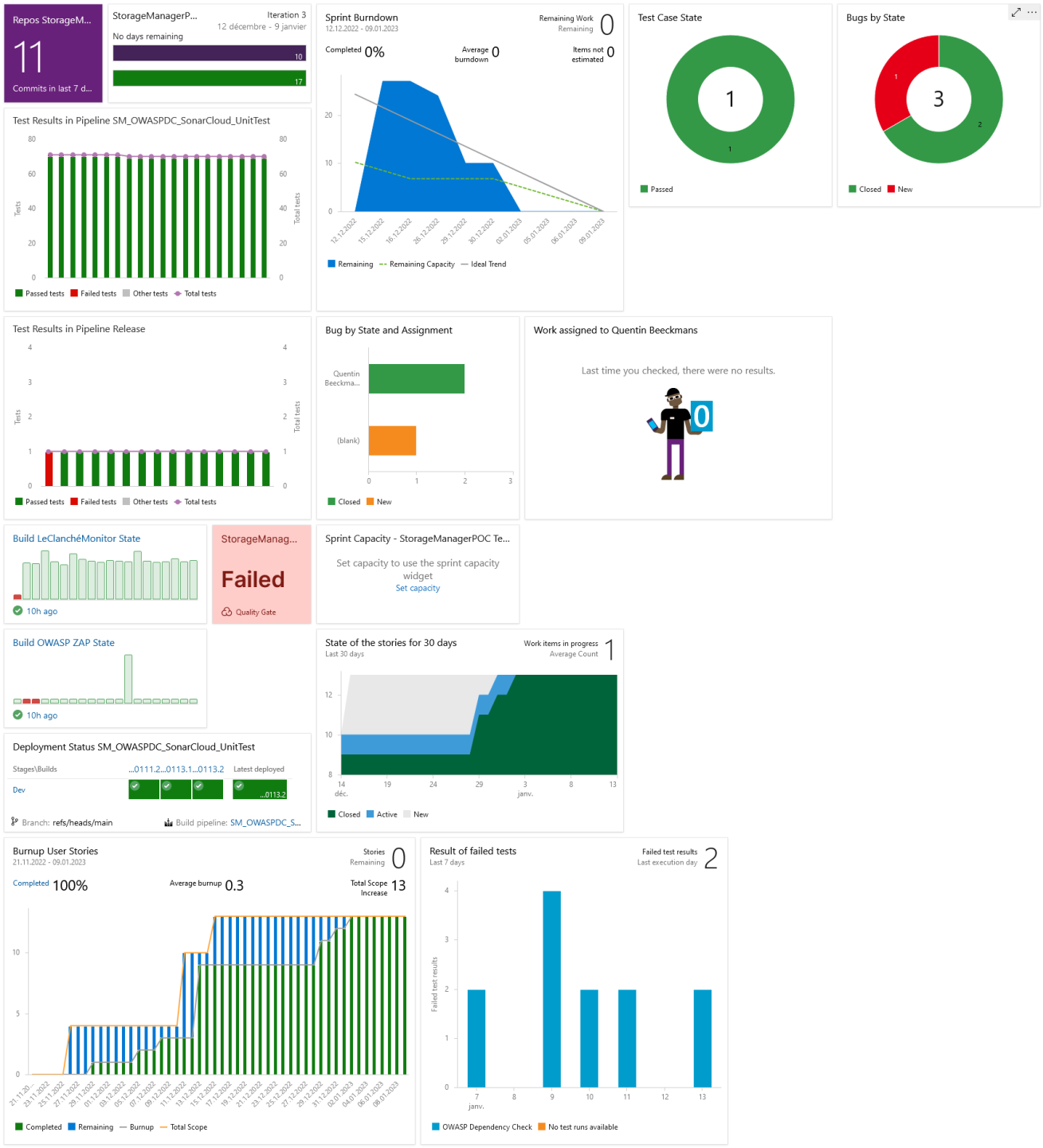
```yaml
trigger:
- main

pool:
  vmImage: 'windows-2019'

variables:
 solution: '**\*.sln'
 project: '**\LeClanchéMonitor.csproj'
 buildPlatform: 'AnyCPU'
 buildConfiguration: 'Release'

steps:
# Script permettant de mettre un tag sur le pipeline
- script: |
   echo ##vso[build.addbuildtag]Test

- task: NuGetToolInstaller@1
 inputs:
  versionSpec:

- task: NuGetCommand@2
 inputs:
  restoreSolution: '$(solution)'

# Tâche de préparation/configuration du test SonarCloud
- task: SonarCloudPrepare@1
 inputs:
  SonarCloud: 'SonarCloud Connection'
  organization: 'storagemanager'
  scannerMode: 'MSBuild'
  projectKey: 'storagemanager_StorageManagerPOC'
  projectName: 'StorageManagerPOC'
  extraProperties: |
   # Additional properties that will be passed to the scanner,
   # Put one key=value per line, example:
   # sonar.exclusions=**/*.bin
   sonar.prokectBaseDir=$(Agent.TempDirectory\\**\coverage.cobertura.xml

# Tâche de Build du projet
- task: MSBuild@1
```

```
  displayName: BUILD
  inputs:
    solution: '$(project)'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'
    msbuildArguments: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPackageLocation="$(build.artifactStagingDirectory)\WebApp.zip" /p:DeployIisAppPath="Default Web Site"'

# Tâche d'installation de l'environnement de test Visual Studio Test Platform et ses dépendances (Préparation Test Unitaire)
- task: VisualStudioTestPlatformInstaller@1
  inputs:
    packageFeedSelector: 'nugetOrg'
    versionSelector: 'latestPreRelease'

# Tâche de test Unitaire
- task: VSTest@2
  displayName: Test Unit Test
  inputs:
    testSelector: 'testAssemblies'
    testAssemblyVer2: |
      UnitTesting\\bin\\Debug\\UnitTest*.dll
      !**\*TestAdapter.dll
      !**\obj\**
    searchFolder: '$(System.DefaultWorkingDirectory)'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'

# Tâche d'analyse SonarCloud
- task: SonarCloudAnalyze@1

# Tâche de test OWASP Dependency Check
- task: dependency-check-build-task@6
  displayName: Test Owasp Dependency Check
  inputs:
    projectName: 'Dependency Check'
    scanPath: '**/*.dll'
    format: 'HTML, JUNIT'

# Tâche de publication des résultats du test OWASP Dependency Check
- task: PublishTestResults@2
  displayName: Publish Owasp Dependency Check result
```

```
  inputs:
    testRunTitle: 'OWASP Dependency Check'
    testResultsFormat: 'JUnit'
    testResultsFiles: 'dependency-check\*junit.xml'
    searchFolder: '$(Common.TestResultsDirectory)'
    failTaskOnFailedTests: false

# Tâche de publication des résultats du test SonarCloud
- task: SonarCloudPublish@1
  inputs:
    pollingTimeoutSec: '300'

# Tâche qui copie le fichier de test Selenium dans le dossier des artefacts pour pouvoir l'utiliser dans l
e Pipeline de Release
- task: CopyFiles@2
  displayName: Copy content of bin for TestSelenium
  inputs:
    SourceFolder: '$(build.sourcesdirectory)'
    Contents: '**\TestSelenium\bin\**'
    TargetFolder: '$(build.artifactstagingdirectory)'

# Tâche qui publie l'artefact généré par notre tâche de build dans le dossier des artefacts
- task: PublishBuildArtifacts@1
  displayName: Create Artefact
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

Annexe X : Azure Dashboards

Figure 112 - Dashboard Azure



Source : Données de l'auteur

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

David Wannier, Professeur à la HES-SO Valais/Wallis

Lieu et date : Signature :

Sierre, le 19 janvier 2023