TEST AUTOMATION AND DIGITAL QA SUMMIT

SOMMET SUR L'AUTOMATISATION DES TESTS ET L'ASSURANCE QUALITÉ NUMÉRIQUE

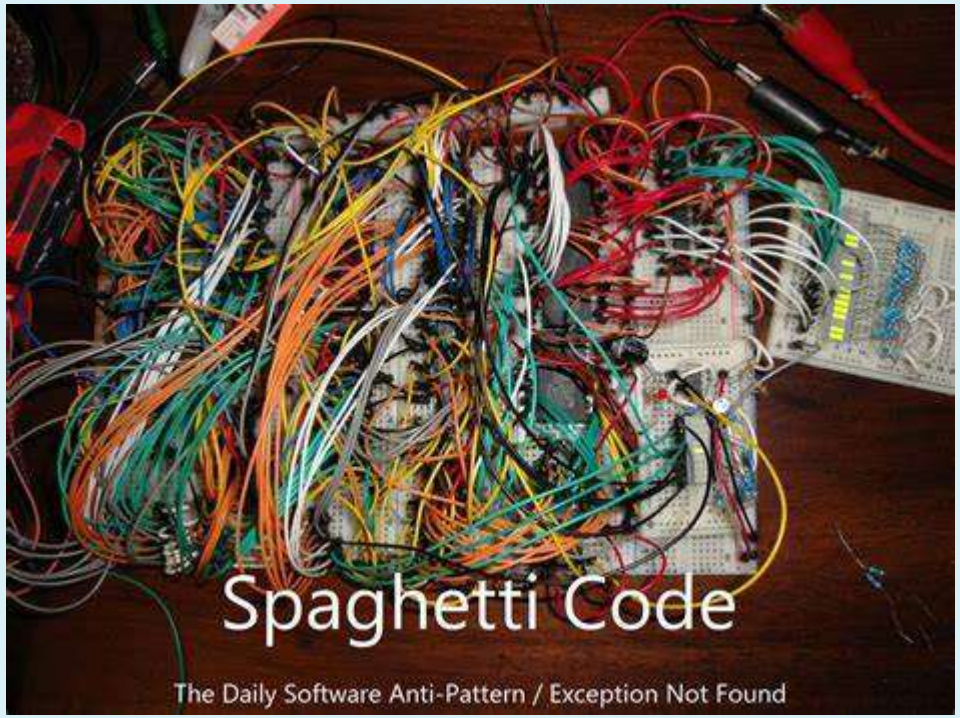
Draft 1.0

Montréal , QC, Canada
11 Juin 2025

BIENVENUE

- A propos de moi
  - **Expérience :** 25 Ans en Technologies de l'information.
  - Fait ses débuts comme Programmeur-analyste (Développeur).
  - S'est impliqué dans l'assurance de la qualité en 2016, pour rapprocher les QA et DEV par l'artisanat du Logiciel « software craftsmanship ».
  - Maintenant Architecte Sénior en Ingénierie de Qualité chez Slalom Build
  - **Passe-temps :** Jouer de la musique, voyager et étudier l'histoire et les nouvelles tendances en TI.
  - A travaillé à Paris comme coach en « software craftsmanship » pendant quelques années.
  - **Qu'est-ce que j'aime dans l'informatique?** – Possibilité d'inventer/innover assez facilement.

ÉPOQUE DU CODE "SPAGHETTI"

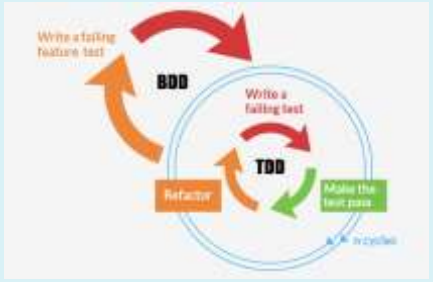


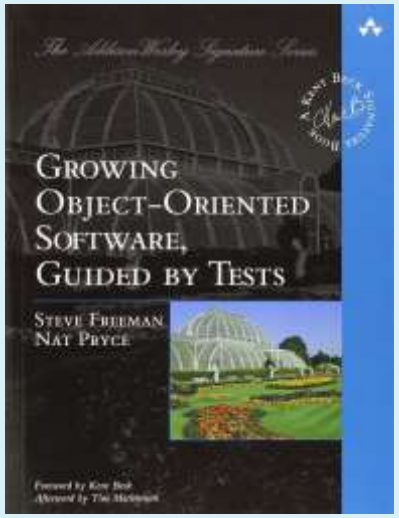
- En tant que développeur
  - As du « galérer » par manque de cohésion, d'agilité, inexpérience et manque de mentorat solide
  - Tout était testé manuellement sans test automatisé, parfois même sans pipeline de « build »
  - ASP et Access ou SQL Server, MTS était un luxe
  - Enregistrements déconnectes ADO XML
  - Sécurité minimale

- **Était fier d'être développeur?**
  - La réponse est OUI et NON.

    « Non car il était impossible de faire du design par le feedback de test, les conventions de code étaient rarement respectées et l'Assurance Qualité quasi-inexistante ou en silo en dernier juste avant la release et Oui car nous réussissions a avoir du succès quand même »

DÉCOUVERTE DE L'ARTISANAT DU LOGICIEL





- Découverte / Design du code guidé par le test
  - As été introduit au TDD en France par certains mentors
    - Oncle Bob (Robert C Martin)
    - Kent Beck (Inventeur de JUnit)
    - Collègues dans une grande banque européenne
  - Le BDD fut la suite évidente, ensuite
  - Web 2.0 , micro-services et DDD
  - Clean code et clean architecture
  - Et finalement Agile XP et le « software craftsmanship »
  - Devenir coach et rendre service a la communauté.

- **Était fier d'être développeur?**
  - La réponse est maintenant juste OUI.
    « Oui car j'ai appris avec beaucoup de pratique et kata de code, a voir les patterns de code qui émergent de certain tests en appliquant les GOF»

L'ARCHITECTE EN QUALITÉ DORMANT…

- Pourquoi devenir ensuite un consultant Assurance Qualité
  - En 2016 je deviens progressivement consultant AQ car le département bancaire pour lequel j'était coach me supporta dans ma démarche et alors j'ai été exposé a la gestion des test, campagnes de test, benchmarking d'outils de test et design d'application, de traçabilité agile etc….
  - Et progressivement j'ai été appelé a coacher aussi des consultants AQ et autres participant des méthodes agile i.e coach agiles sur le côté pratique du développement agile et le craftsmanship
  - Travail sur de très grosse application informatique
  - Travail avec des DevOps
  - Aujourd'hui je suis Architecte en qualité logicielle

ÉNONCÉ DE MISSION

Le client demande : « Pouvez-vous nous aider à tester notre plateforme infonuagique Azure et avez-vous une idée de comment s'y prendre car nous n'avons aucune idée sur comment le faire? »

AVANT DE COMMENCER, VOICI LES CONCEPTS CLÉS DE LA PRÉSENTATION

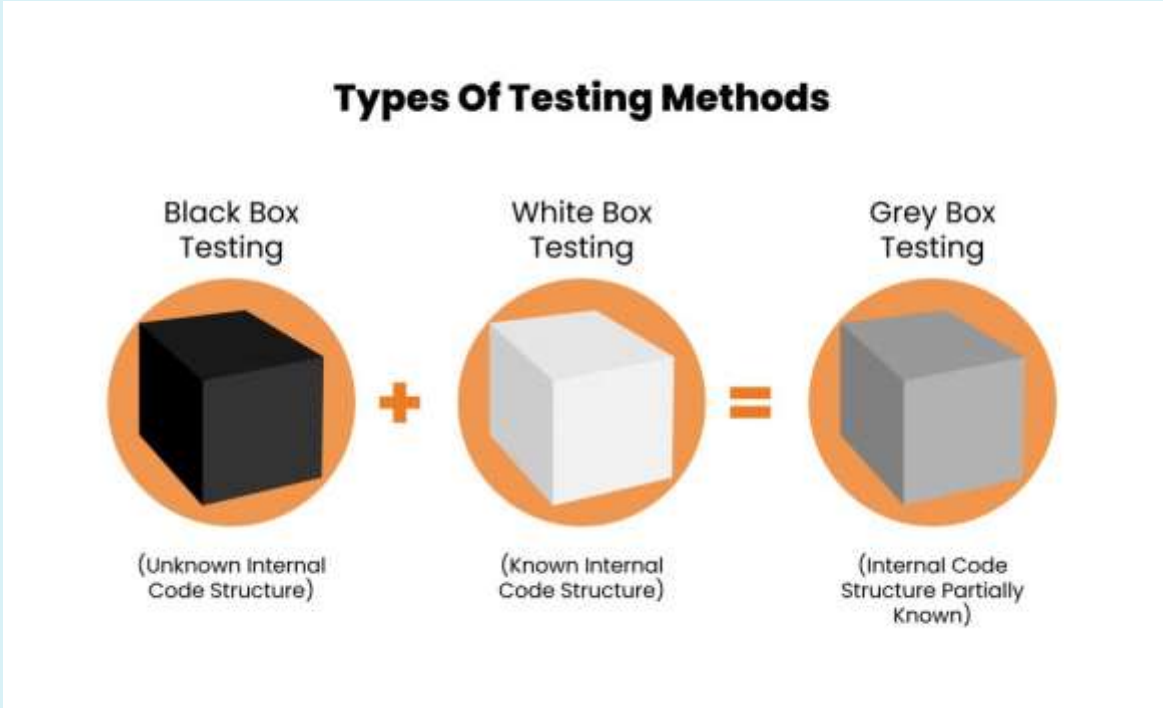
- **Comment accepter une mission en Assurance Qualité?**
  - Soyez honnête, si vous ne savez pas, dites
    - « Je ne sais pas, mais je vais trouver la solution qui conviens au projet ».
  - Prévoyez le temps pour comprendre le problème à résoudre.
  - Appliquer les bases de l'AQ et du développement agile.
  - Effectuez des recherches et même si vos informations sont partielles et floues, rassemblez-les.
  - Utilisez vos expériences passées pour vous inspirer
  - Quels « prompts » sur un LLM peut accélérer le prototype.
  - Ayez une attitude "CAN-DO" et listez les spécifications.

IDENTIFIER LE TYPE DE SYSTÈME À TESTER

Choisir la bonne méthode

- Architectes et ingénieurs DevOps sont t'ils disponible pour collaborer?
- Vont 'il collaborer avec L'AQ ?
- Allons nous travailler en mode incrémental (Scrum)?
- Avons-nous des spécifications claires?

Finalement sachant que tester les systèmes en mode **boite noire** conviens très bien car nous testeront des spécifications d'un déploiement «Azure Bicep ou Terraform» d'une manière itérative et synchronisée en méthode Agile /Scrum avec des tests fonctionnels selon les règles d'affaires de sécurité, rbacs, gouvernance, réseautiques en testant les résultats de configuration d'un déploiement.



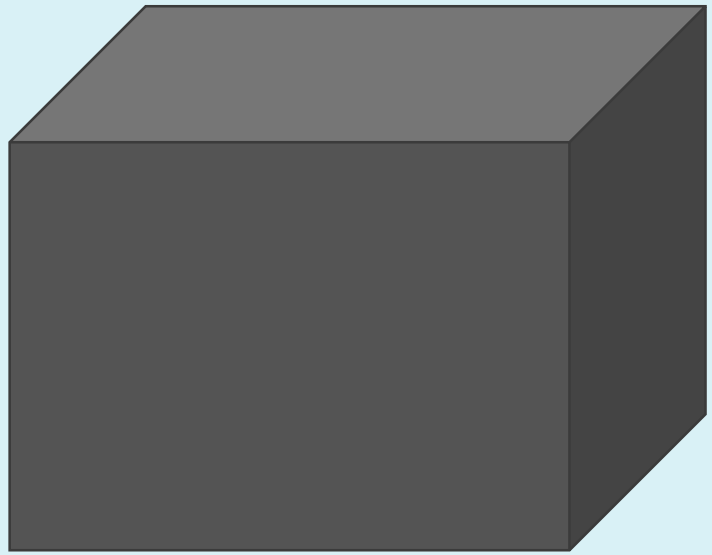
Azure Bicep

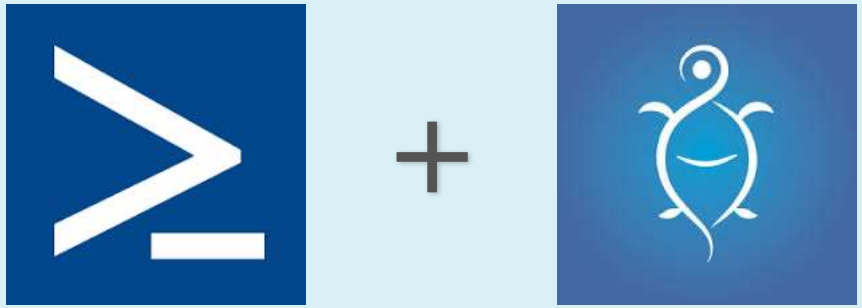
OU

Terraform

SOLUTION : BOITE NOIRE POWERSHELL ET PESTER

Finalement nous testeront des spécifications d'une manière qui est proche du BDD traditionnel mais sans que cela devienne lourd et dogmatique. Ce cas d'usage ne couvre pas les tests unitaires qui est selon moi une responsabilité DevOps et non AQ.





IDENTIFIER LES OUTILS POUR TESTER

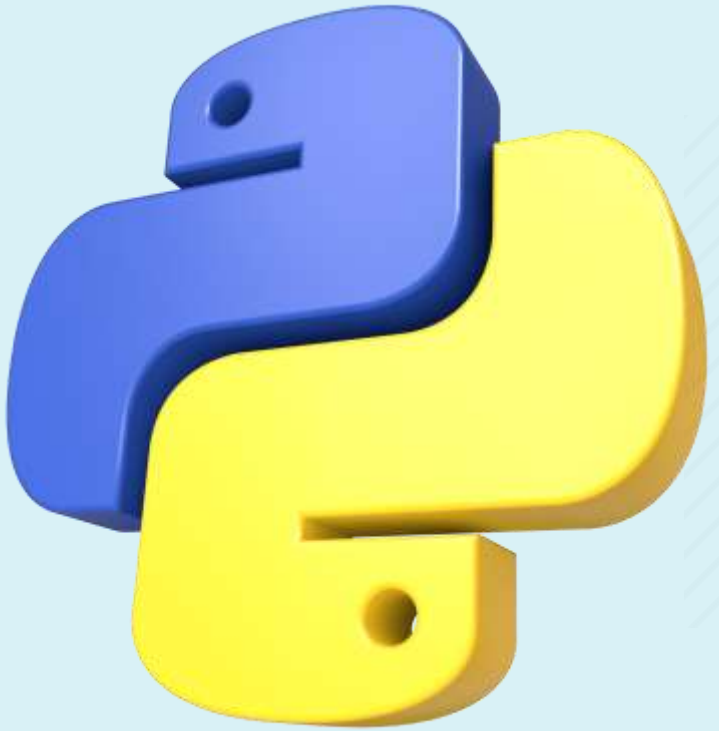
Choisir les bons outils pour notre cas d'usage

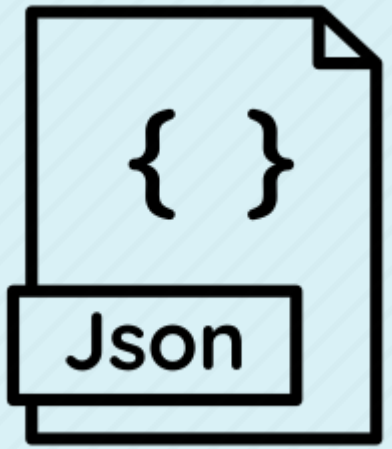
- Langage Microsoft avec la librairie Azure
  - PowerShell Core
  - Librairie d'assertion pour PowerShell (Pester)
- Pourquoi Pester et PowerShell?
  - Permet de faire des tests automatisés de type « Data-Driven Testing »
  - PowerShell est le « Bash » de Microsoft, facile a comprendre et doit être un langage interprété

AUTRES IDÉES COMPATIBLES AVEC LA SOLUTION

Choisir les bons outils pour notre cas d'usage peut aussi être:

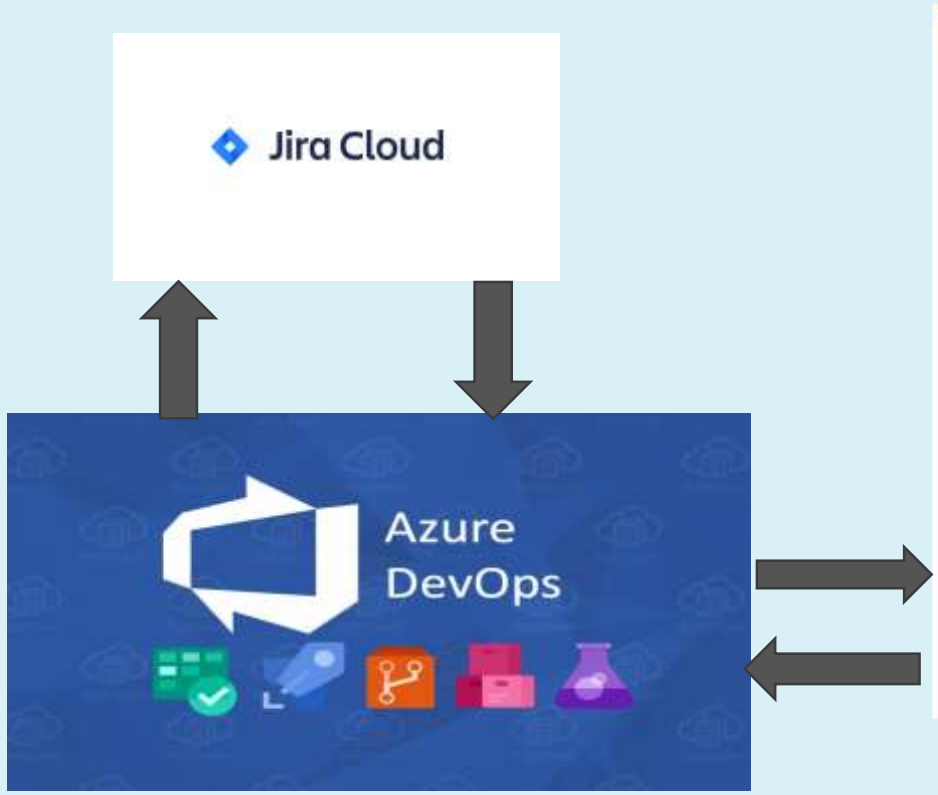
- Langage Python avec la librairie Azure
  - Azure librairies (SDK) for Python
  - Fixtures en JSON
- Pourquoi Python et JSON?
  - Permet de faire des tests automatisés de type « Data-Driven Testing »
    - py.test parametrize
  - Python est très populaire et est une bonne alternative a PowerShell

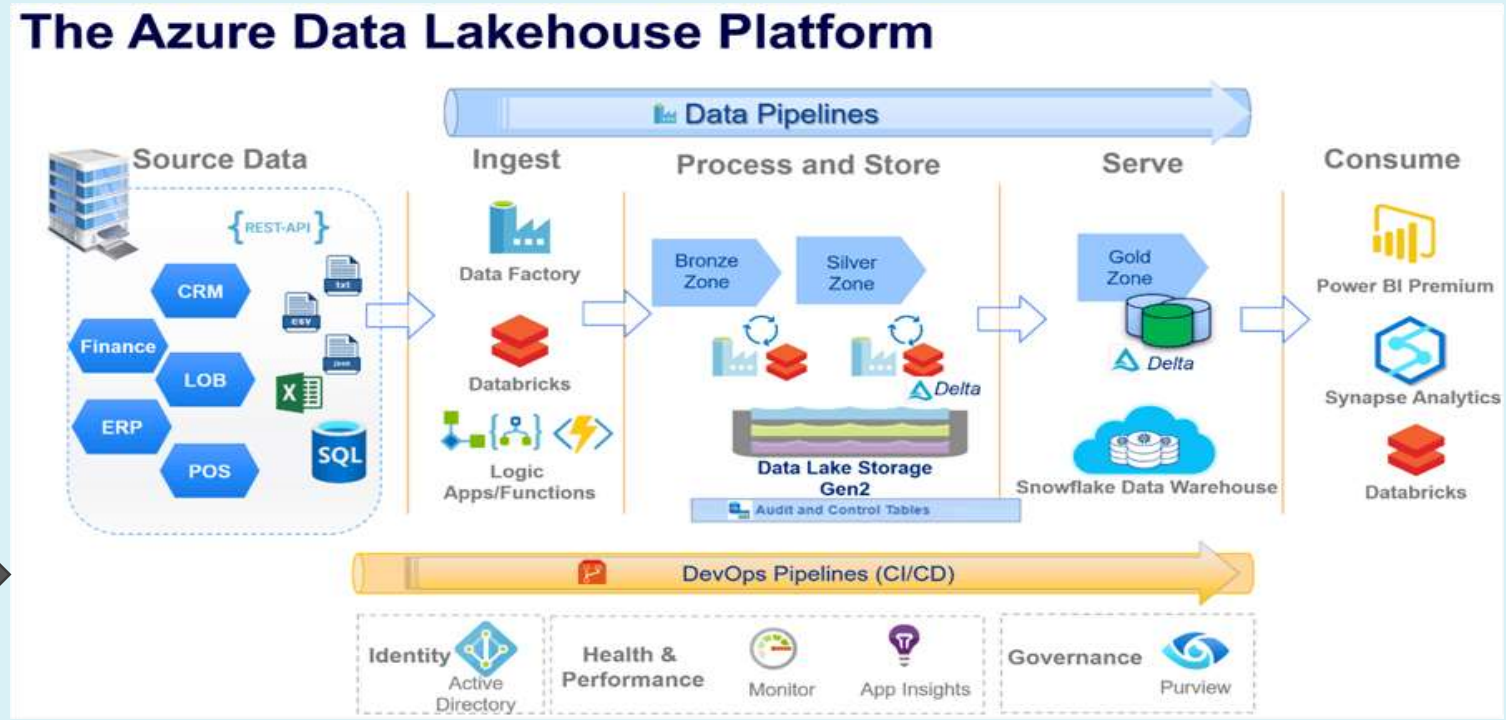




INTÉGRATION DES TESTS AUTOMATISÉS SUR DEVOPS AVEC UNE MÉTHODE AGILE

Un cas d'utilisation d'Azure « Lakehouse », qui contient Azure DevOps comme pipeline de test et synchronise avec Jira a chaque Build/Release

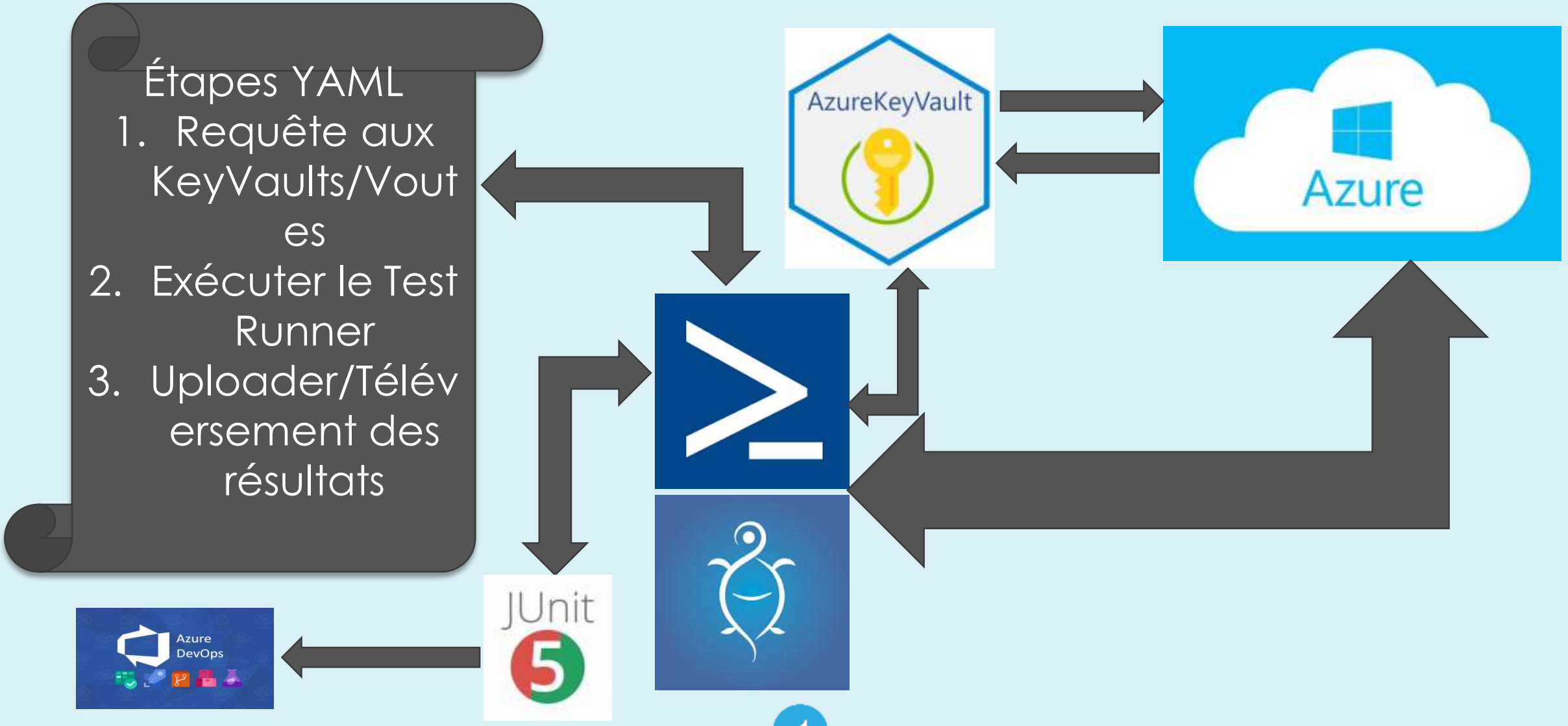




COMPOSANTS DE LA SOLUTION

- Configuration CI/CD YAML
- Le “runner” PowerShell Core pour Pester
  - Utilisation de la configuration et sessions Pester
- Le fichier DiscoveryPhase
- Les fixtures
- Les tests dynamiques
- Les dépendances
  - PowerShell Core (YAML et code)
    - Libraries Az.* (les appels REST sont encapsulés dans des “getters” Az)
- Rapports et Métriques avec le format JUnit pour uplaod vers Jira XRay et Azure DevOps
- Avoir un dossier de fonction réutilisables .i.e. Switch-Subscription, Assert-Exists, Assert-Contains, etc…

ILLUSTRATION DE LA SOLUTION



PESTER, LIBRAIRIE D'ASSERTION POUR POWERSHELL

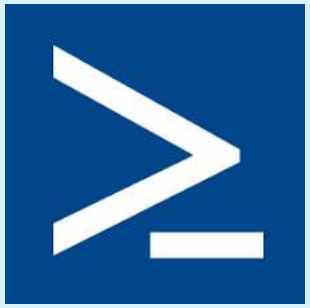
Pester vous permet

- D'utiliser le pattern « **Describe**, **It** » comme avec certaines librairies de test « Javascript »
- De profiter des évènements comme « **BeforeDiscovery** »
- De multiplier les tests avec des boucles « **ForEach** »

https://pester.dev

LES TESTS 'DATA-DRIVEN' PESTER

En prenant avantage des boucles de code et des wildcards / placholders

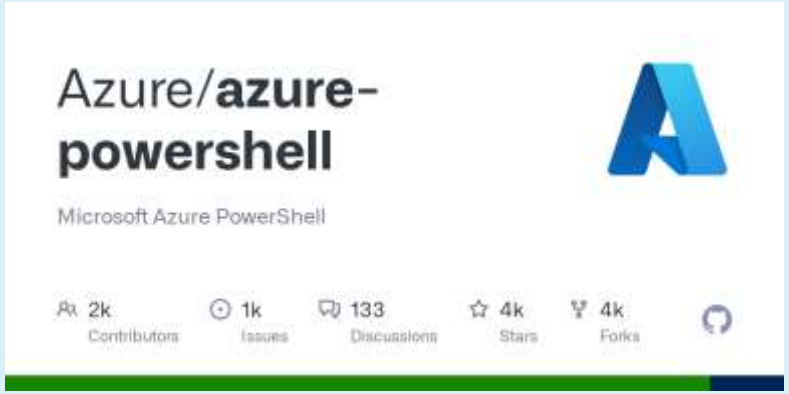


https://pester.dev/docs/usage/data-driven-tests

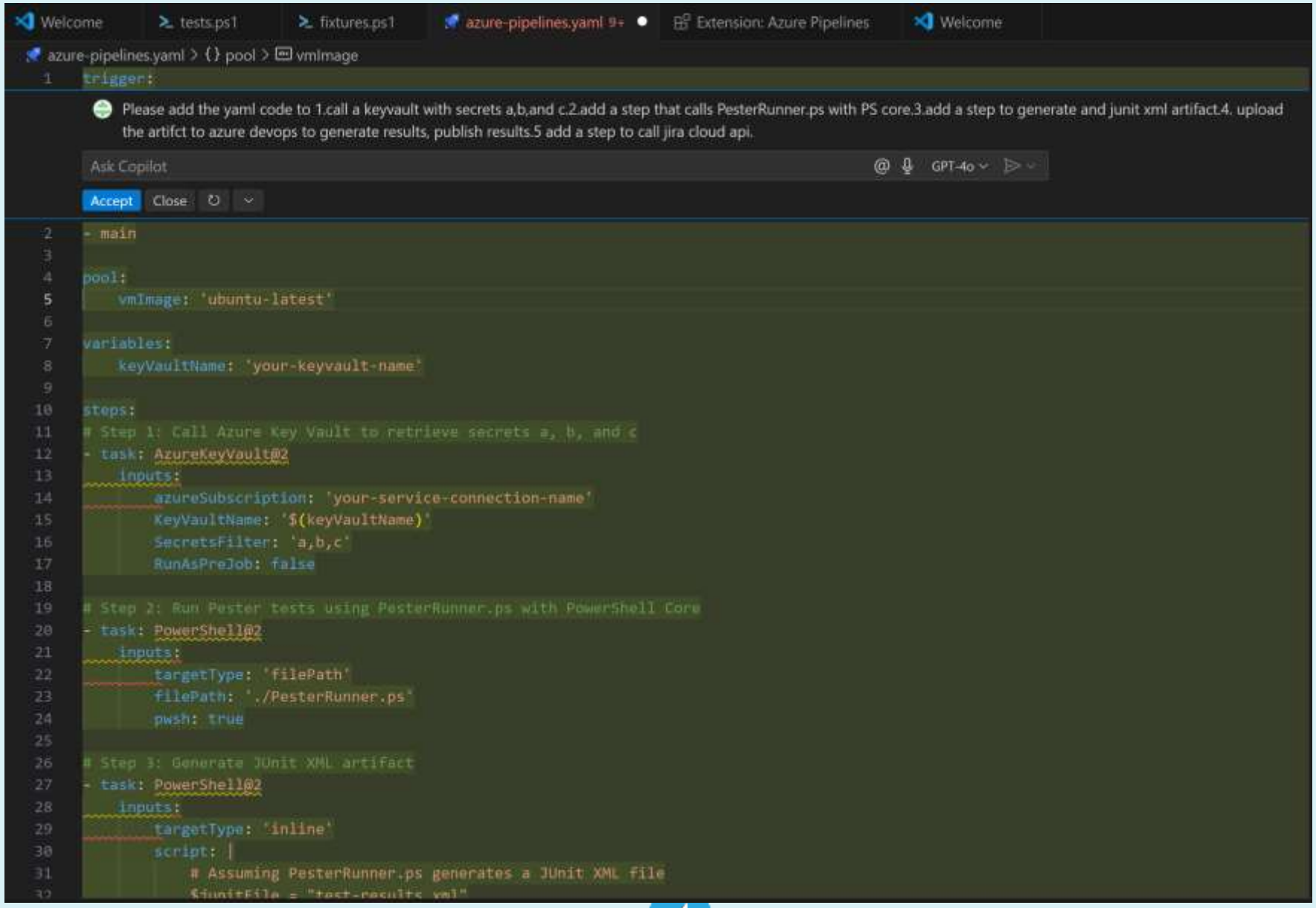
```
#In this case we will execute 4 tests, 2 for each storage account. The first test will check if the storage account exists
# and the second one will check if it is secure.
$expectedInfraStorageAccounts = @(
    @{ Name = "storageaccount1"; Config = @{ AllowBlobAccess = $false } },
    @{ Name = "storageaccount2"; Config = @{ AllowBlobAccess = $false } }
)

Describe "Storage Account deployment" {
    Context "<_> is deployed and online " -ForEach $expectedInfraStorageAccounts  {
        It "<_.FriendlyName> exists" {
            $actualStorageAccount = $actualStorageAccounts | Where-Object { Name -eq $_.Name }
            $actualStorageAccount | Should -Not -BeNullOrEmpty
        }

        It "<_.FriendlyName> is secure" {            #filter : hastable in the array by AllowBlobAccess is an attribute
            $actualStorageAccount = $actualStorageAccounts | Where-Object { Name -eq $_.Name  -and $_.AllowBlobAccess -eq $false }
            $actualStorageAccount | Should -Not -BeNullOrEmpty
        }
    }
}
```

PESTER ET L'ÉVÈNEMENT BEFOREDISCOVERY

```
#this file is called DiscoveryPhase.ps1 and all #Az.* references are in the PesterRunner.ps1 #file
BeforeDiscovery {
$actualStorageAccounts = Get-AzStorageAccount
#all test files will include/dot source this file, #you could use this event in  each test file, we #prefer to centralize all Azure Cloud AZ queries #from here
 }
```



PROMPT GPT-O AVEC GITHUB COPILOT POUR GÉNÉRER LE FICHIER YAML



FIXTURES DE TEST

Tableau (array)

- CONTIENS : Tables de hachage (hashtable)

Exemple : InfrastructureFixtures.ps1 contiens les objets Az et leurs configurations afin de valider les spécifications. Assertion « $actuals vs $expected »

```
#hashtable fixtures
$expectedSTIngestionConfig = @{
    Name                = "stIngestion"
    FriendlyName        = "stockage d'ingestion"
    AllowPublicBlobAccess = $false
}

$expectedSTLOBConfigurations = @{
    Name                = "stLOBConfigurations"
    FriendlyName        = "stockage des configuratition d'une ligne d'affaire"
    AllowPublicBlobAccess = $false
}

$expectedSTBlogPosts = @{
    Name                = "stBlogPosts"
    FriendlyName        = "stockage des post de blob de l'organisation"
    AllowPublicBlobAccess = $false
}
#array fixture
$global:expectedInfraStorageAccounts = @($expectedSTIngestionConfig,$expectedSTLOBConfigurations,$expectedSTBlogPosts)
```

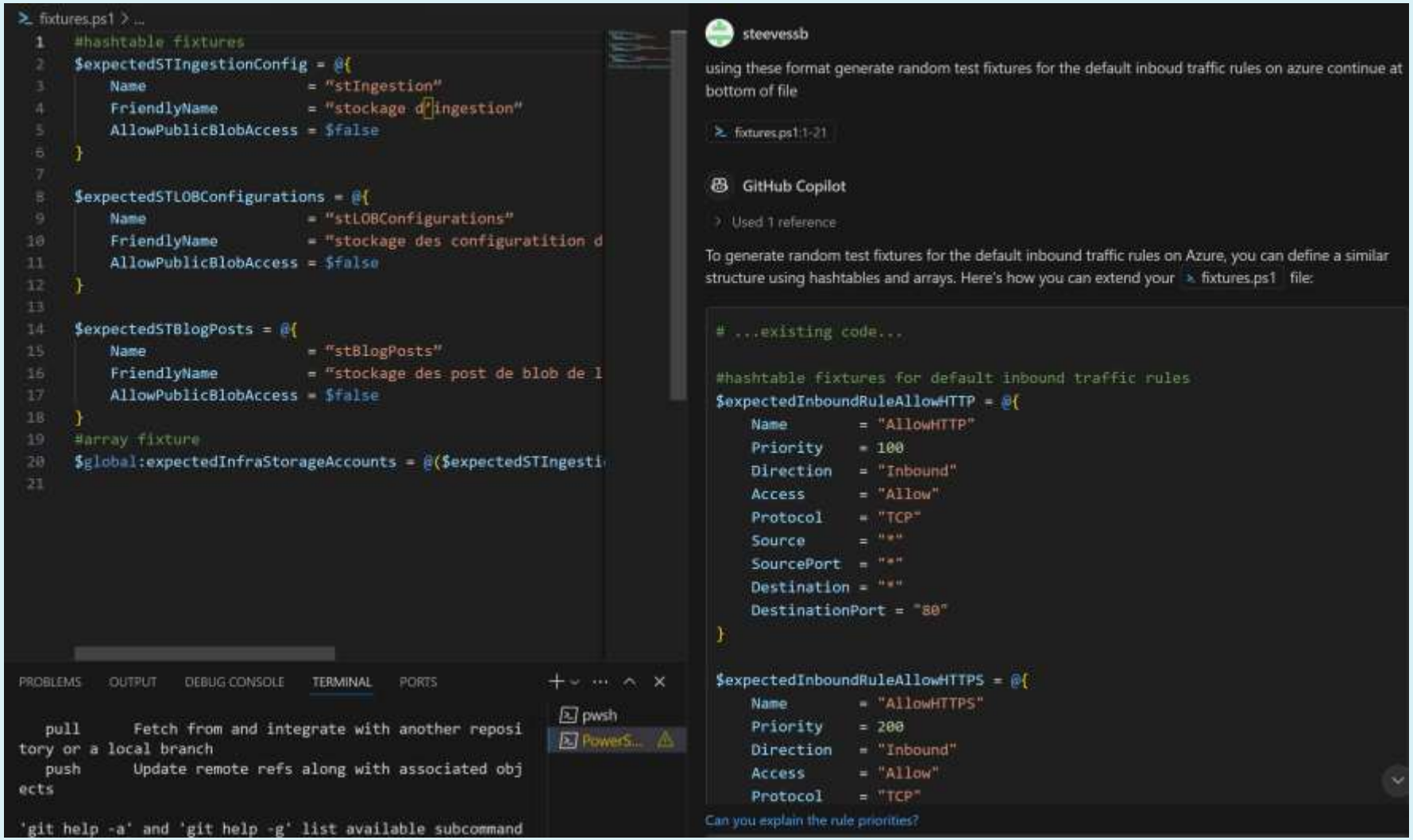
FIXTURES DE TEST –EXPLIQUÉES AVEC FONCTIONS

- L'intention des fixtures est de comparer l'attendu avec l'actuel qui est déployé sur le nuage
- Aussi d'être capable d'itérer sur les tableau et « hastables » et de les charger a l'exécution « runtime » dynamique afin de remplacer les
  - « Wildcard » $._ et <_> par les valeurs de fixtures et les comparer avec les $actuals qui sont chargées sur l'évènement BeforeDiscovery

```
Describe "Storage Account deployment" {
    Context "<_> is deployed and online " -ForEach $expectedInfraStorageAccounts  {
        It "<_> exists" {
            $actualStorageAccount = $actualStorageAccounts | Where-Object { Name -eq $_.Name }
            $actualStorageAccount | Should -Not -BeNullOrEmpty
        }

        It "<_> is secure" {                              #here Config is a hastable in the array AllowBlobAccess is an attribute
            $actualStorageAccount = $actualStorageAccounts | Where-Object { Name -eq $_  -and $_.Config.AllowBlobAccess -eq $false }
            $actualStorageAccount | Should -Not -BeNullOrEmpty
        }
    }
}
```

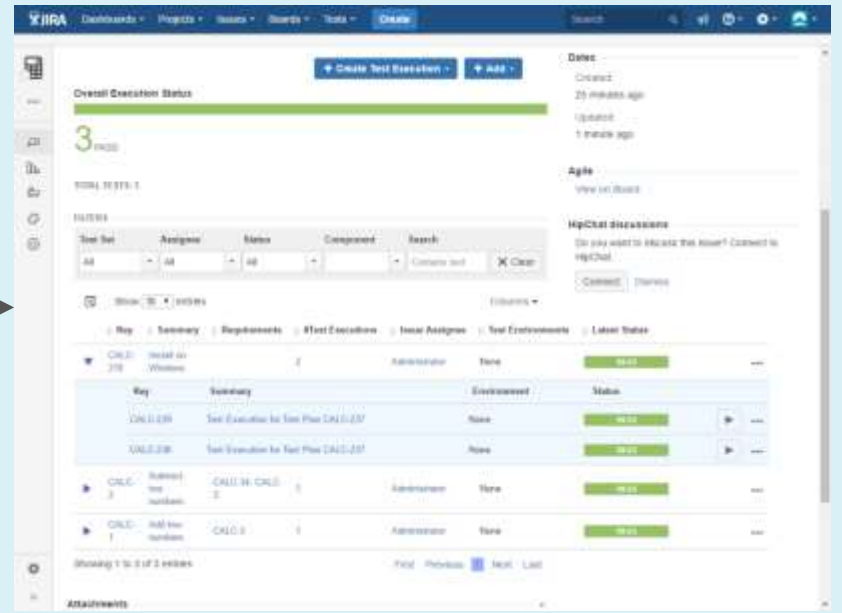
UTILISATION DE L'AI AVEC GOOGLE COPILOT

LE # PROMPT EST TRÈS PUISSANT POUR GÉNÉRER DES FIXTURES ET FONCTIONS D'ASSERTIONS

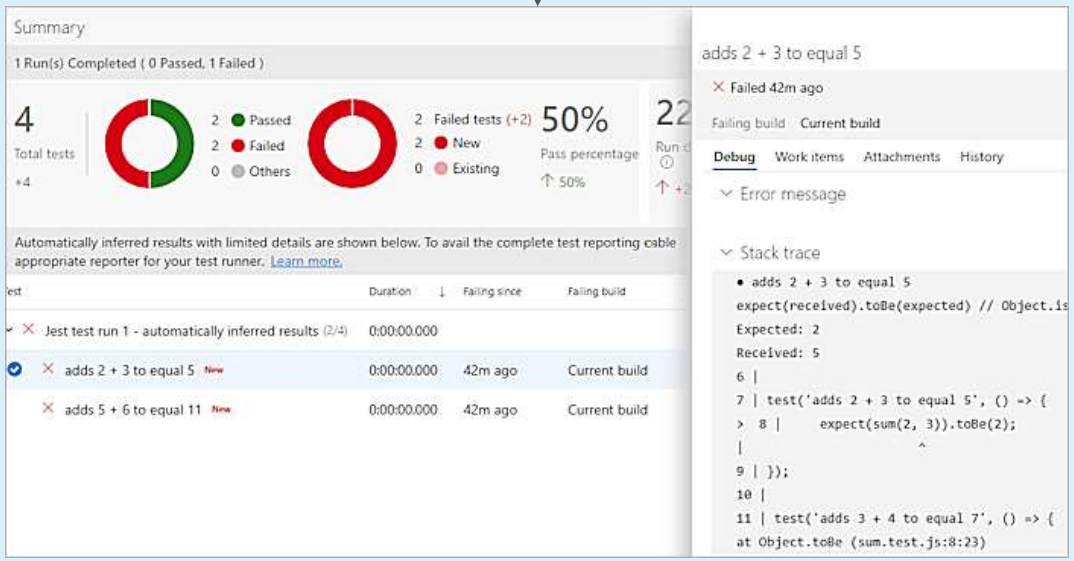


RAPPORTS EN 3 ASPECTS

1. Sortie I/O en console
2. Rapports Azure DevOps
3. Rapports XRAY Jira



```
Pester v5.3.0-alpha4

Starting discovery in 2 files.
Discovery found 4 tests in 20ms.
Running tests.

Running tests from '/workspaces/Pester/Samples/demoOutput.tests.ps1'
Describing Output sample
 Context Demo context
   [+] Successfull test 6ms (3ms|3ms)
  [+] Outer test 4ms (3ms|2ms)

Running tests from '/workspaces/Pester/Samples/demoOutput2.tests.ps1'
Describing Output sample
 Context Demo context
   [+] Successfull test 6ms (3ms|2ms)
  [-] failing test 14ms (13ms|1ms)
   Expected 2, but got 1.
   at testFunc | Should -Be 2, /workspaces/Pester/Samples/demoOutput2.tests.ps1:11
   at <ScriptBlock>, /workspaces/Pester/Samples/demoOutput2.tests.ps1:11
Tests completed in 129ms
Tests Passed: 3, Failed: 1, Skipped: 0 NotRun: 0
```



ORGANISATION DU TRAVAIL

- Planification et exécution du Sprint
  - Je conseille de faire un sprint 0 pour faire un « POC » et expérimentation car une solution comme celle-ci nécessite au moins un ou deux “spike” pour ensuite itérer dans le Testing/développement sachant que la forme Describe / IT se prête bien a cela, c’est une forme de BDD (Behavior Driven Design)
  - Faire des échanges avec ses pairs selon la stratégie de test.
  - Extraire le savoir de la base de connaissance du projet
    - Confluence
  - Relier les ‘issues’ Jira avec les plan de tests
    - 1 exécution par issue / story Jira
    - Ajouter des « requirements » Xray pour avoir:
      - Couverture fonctionnelle de test
      - Estimation de l’avancement d’une Epic.

NE PAS FAIRE

- Cree des issues de type bug systématiquement quand il y a des erreurs dans les tests automatisés car parfois
  - Une fonctionnalité n'est pas finie d'être implémentée
  - Je conseille de laisser le temps prévu par le sprint.
  - Si ensuite il y a encore un bogue ou une régression
    - Cree une issue Jira de type « Bug » et la rattacher a la user story
    - Mettre aussi le bug en Backlog si la prioritée est Triviale
      - Ceci de la DOD et de la stratégie de test

- Ajouter des fonctionnalités de Testing avant d'en avoir de besoin
- Répéter du code
- Ne pas utiliser un repo git comme le reste du code de deployement

Vos Question S.V.P ?