

Formation Azure

Ihab ABADI / UTOPIOS

SOMMAIRE – Partie 1

1. Approvisionnement de machines virtuelles dans Azure
2. Azure Resource Manager
3. Azure Container Registry
4. Azure Container Instances

SOMMAIRE – Partie 2

1. Persistance fichier Azure
2. Persistance Base de données relationnelles
3. Utilisation des Queues Messages

Approvisionnement de machines virtuelles dans Azure

- Exploration des machines virtuelles Azure
- La disponibilité des machines virtuelles
- Scalabilité des vms
- Exercice

Création d'une machine virtuelle

- La création d'une machine virtuelle nécessite l'existence d'un groupe de ressources.
- La création d'une machine virtuelle peut se faire à l'aide de azure cli
- Azure nous permet de créer des machine virtuelle à partir d'une multitude d'image.
- Chaque machine virtuelle peut être définie avec une taille en fonction de la région.
- Azure nous offre la possibilité de faire un redimensionnement des machines virtuelles
- Par défaut, le seul port ouvert est le port ssh.
- Nous avons la possibilité de modifier la configuration réseau de la machine.

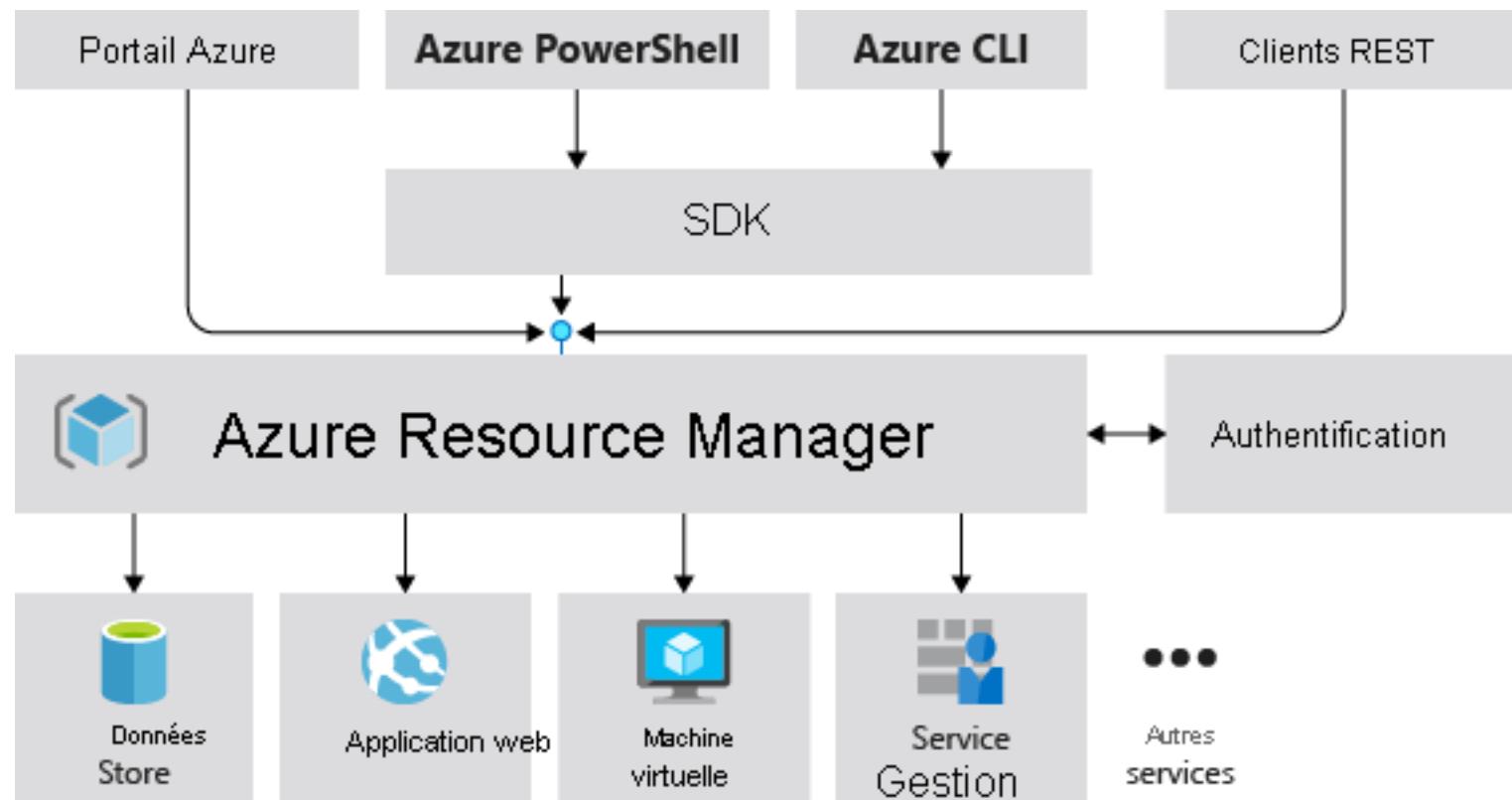
Exercice machine virtuelle

- A l'aide du cli azure,
- Créer une machine virtuelle à partir d'une image ubuntu.
- Se connecter à la machine et installer un service nginx.
- Ouvrir le port par défaut nginx sur la machine.
- Tester la connexion du serveur web nginx.

Azure Resource Manager

- Introduction
- Utilisation d'Azure Resource manager
- Déploiement de solutions avec Azure resource manager
- TP

Introduction ARM



Introduction ARM

- ARM offre la possibilité de créer des ressources azure tel que les Vms, les infrastructure réseau, système de stockage ou n'importe quel type de service d'une façon déclarative.
- ARM permet de déployer notre infrastructure d'une façon reproductibles tout au long du cycle de vie de nos applications.
- ARM offre une abstraction des commandes et des opérations à effectuer pour le déploiement.

Utilisation des ARM

- Un modèle Azure Resource Manager peut être en format json.
- Un modèle Azure Resource Manager est composé de :
 - Une partie paramètres
 - Une partie Variables
 - Une partie Ressources
 - Une partie Sorties.

Un modèle ARM peut également comporter des fonctions personnalisées

Utilisation des ARMs

- Démo

Mode de déploiement des ARMS

- Mode incrémentiel (Mode par défaut)
 - Ce mode permet ne pas supprimer les ressources dans le cas d'une mise à jour des fichiers de déploiements
- Mode complet
 - Ce mode va avoir comme conséquence la création de la totalité des ressources et la suppression des anciennes.
- Démo

Exercice ARM

- Dans un déploiement ARM Créer les ressources suivantes:
- Une ressource de type storage
- Une ressource de type networkSecurityGroups
- Une ressource de type PublicIPAddresses
- Une ressource de type virtualMachine
- Créer un fichier de configuration séparé.
- Déployer votre ARM

Azure Container Registry

- Acr est un service de registre d'image de conteneur privé.
- Acr propose trois niveaux de service base, Standard et premium.
- Acr fournit un mécanisme d'authentification basé sur Azure active Directory.
- Acr fournit un service gestion automatisé des images pour faciliter le build des images d'une pipeline d'intégration continue.
- Acr prend en charge des images linux et windows
- Démo

Azure container service - Exercice

- A partir d'une application web (api rest ou autre).
- Créer une image pour votre application.
- Déployer votre image sur notre acr.

Azure Container Service

- Acs est un service qui permet de démarrer et déployer des conteneurs en tant que service à partir de nos images.
- Acs est un service pour des conteneurs isolés.
- Acs permet d'exposer les applications à l'aide d'une adresse ip public.
- Acs permet le montage de volume à partir de azure file par exemple.
- Acs permet de gérer les ressources physique.
- Acs permet le déploiement de groupes de conteneurs dans un réseaux virtuelle.
- Démo.

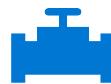
Azure Container Service - Exercice

- Créer un conteneur à partir de l'image créée dans l'exercice crs à l'aide du cli.
- Créer une ressource ARM pour créer un service acs, toujours à partir de notre image.
- Créer une application 2 qui communique avec notre application de l'image 1.
- Créer une image 2.
- Déployer un groupe de conteneur :
- Conteneur 1 à partir de l'image 1
- Conteneur 2 à partir de l'image 2

Azure App Service



Tight integration w/
Docker Hub, Azure Container Registry



Built-in CI/CD w/
Deployment Slots



Intelligent diagnostics &
troubleshooting, remote debugging

Fully managed platform



Automatic scaling
and load balancing



High availability
w/ auto-patching



Backup & recovery

Flexibility & choices



From CLI, portal, or
ARM template



Single Docker image,
multi container w/ Docker compose,



IntelliJ, Jenkin, Maven Visual Studio family

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Contoso Hotels ▾

Resource Group * ⓘ

 ▾

[Create new](#)

Instance Details

Name *

Web App name.

.azurewebsites.net

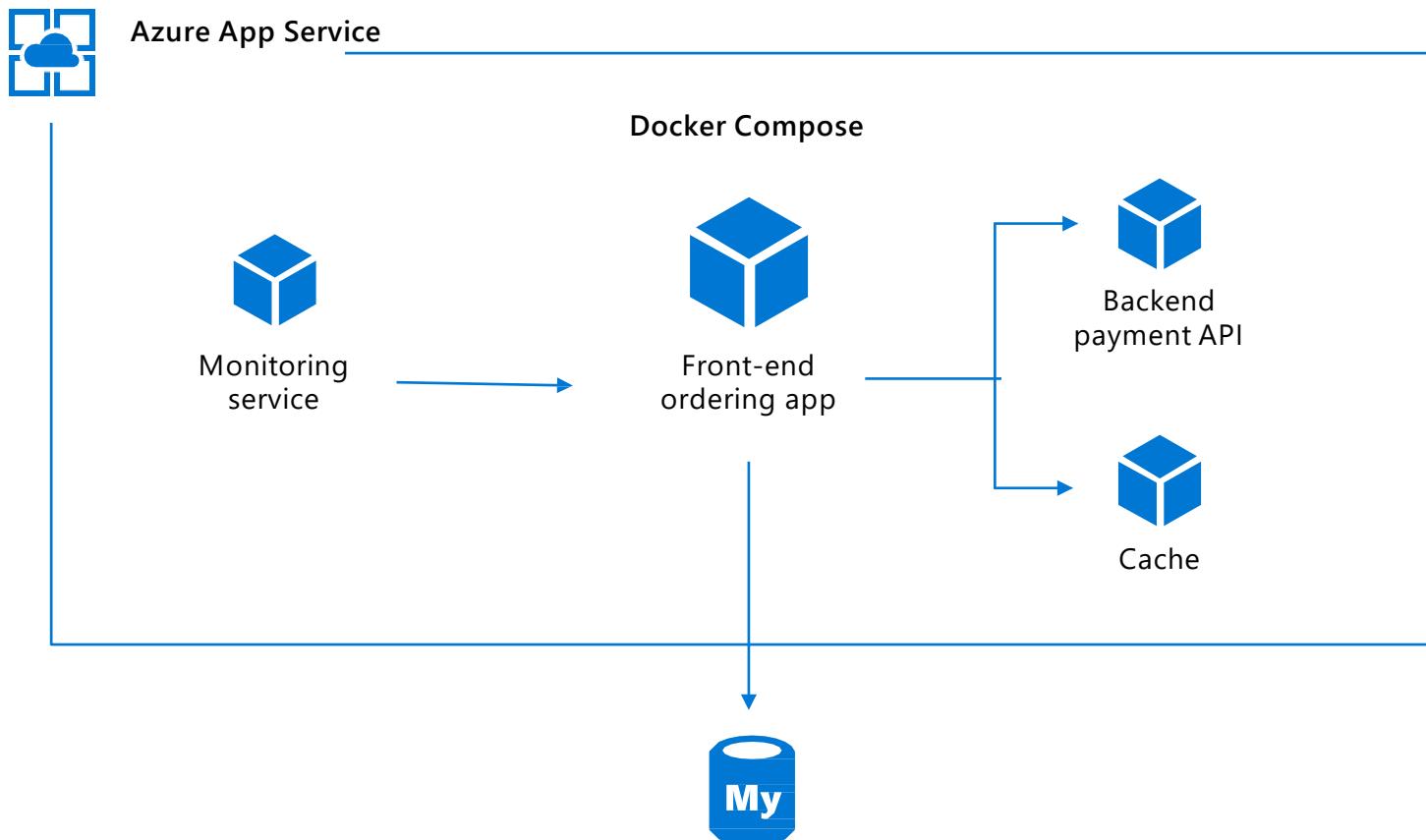
Publish *

Code Docker Container
Ihab ABADI - UTOPIOS

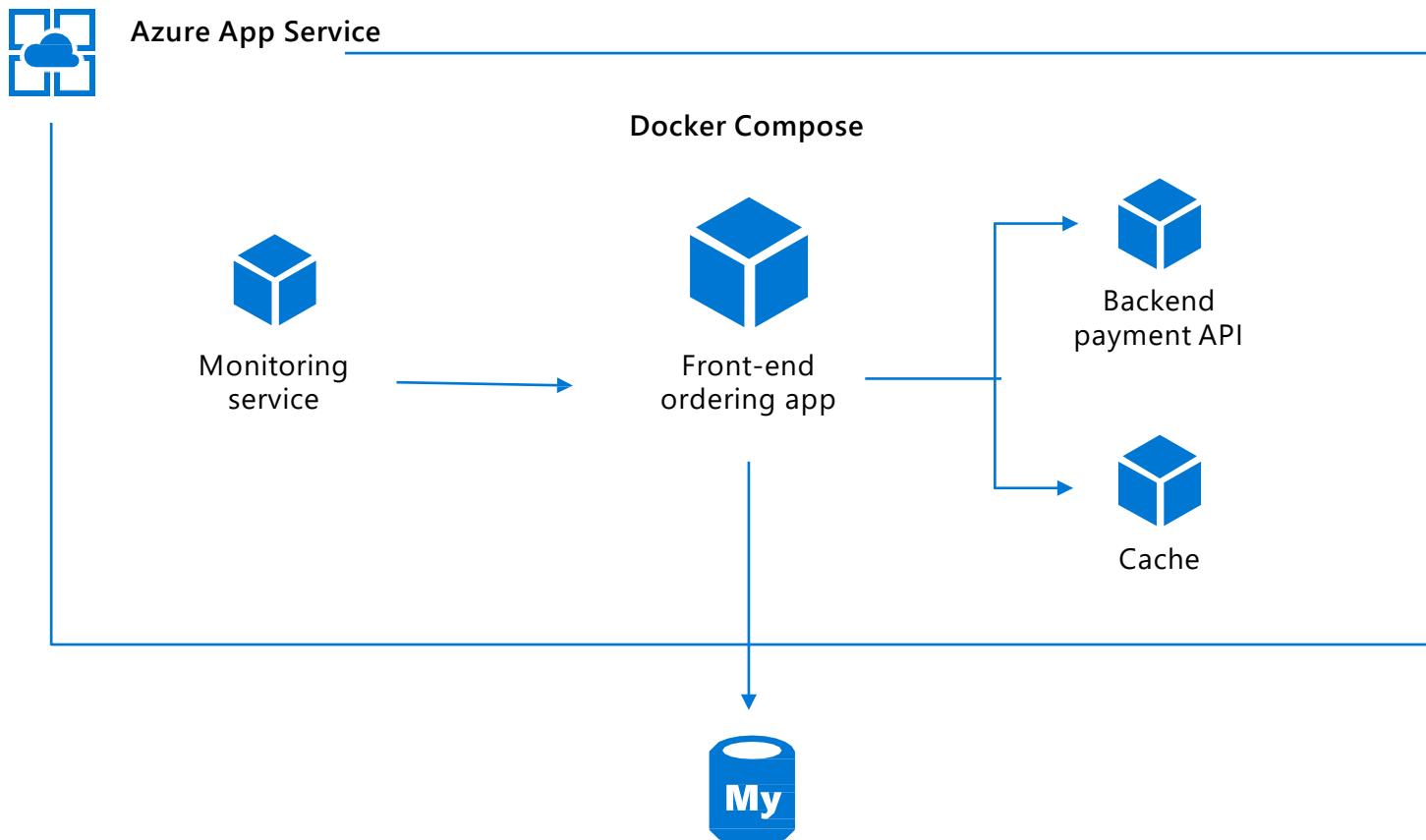
Azure App Service – Azure CLI

- \$ az webapp create --name \$appName --plan AppServiceLinuxDockerPlan --deployment-container-image-name \$dockerHubContainerPath - -resource-group \$myResourceGroup

Azure App Service – Multi-conteneur



Azure App Service – Multi-conteneur



Azure App Service – Azure CLI

- ```
$ az webapp create --resource-group
$myResourceGroup --plan
AppServiceLinuxDockerPlan --name $appName --
multicontainer-config-type compose --
multicontainer-config-file docker-compose-
wordpress.yml
```

# AppServiceLinuxDocker17303 | Container settings (Classic)

App Service | Directory: Microsoft

Search (Ctrl+ /)

Public Private

Deployment Center (Classic)

## Settings

Configuration

Container settings (Classic)

Authentication / Authorization

Authentication (preview)

Application Insights

Identity

Backups

Custom domains

TLS/SSL settings

Networking

Scale up (App Service plan)

Scale out (App Service plan)

WebJobs

Push

MySQL In App

Properties

Locks

Service plan

Configuration File

Choose File No file chosen

## Configuration

```
wordpress:
depends_on:
 - db
image: wordpress:latest
ports:
 - "8000:80"
restart: always
environment:
 WORDPRESS_DB_HOST: db:3306
 WORDPRESS_DB_USER: wordpress
 WORDPRESS_DB_PASSWORD: ****
```

## Continuous Deployment

On Off

Webhook URL [show url](#)

\*\*\*\*

[Copy](#)

## Logs

```
2021-03-07T11:22:11.094Z INFO - Pull image successful, time taken: 2 minutes and 27 seconds
2021-03-07T11:22:11.094Z INFO - Starting container for site
2021-03-07T11:22:11.094Z INFO - docker run -d -p 1371:80 --name appservicelinuxdocker17303_wordpress_0_e737865d -e WEBSITE_SITE_NAME=AppServiceLinuxDocker17303 -e WEBSITE_AUTH_ENABLED=False -e WEBSITE_ROLE_INSTANCE_ID=0 -e WEBSITE_HOSTNAME=appservicelinuxdocker17303.azurewebsites.net -e WEBSITE_INSTANCE_ID=abf0220645ea22573568610313bec0b920fd62831eeab85513f40eb93050918a
wordpress:latest
```

```
2021-03-07T11:22:11.094Z INFO - Logging is not enabled for this container.
```

```
Please use https://aka.ms/linux-diagnostics to enable logging to see container logs here.
```

```
2021-03-07T11:23:40.106Z INFO - Started multi-container app
```

```
2021-03-07T11:23:40.126Z INFO - Initiating warmup request to container appservicelinuxdocker17303_wordpress_0_e737865d for site appservicelinuxdocker17303
```

```
2021-03-07T11:23:40.144Z INFO - Container appservicelinuxdocker17303_wordpress_0_e737865d for site appservicelinuxdocker17303 initialized successfully and is ready to serve requests.
```

Download

Refresh

# Azure Storage Platform – Introduction

- Microsoft propose Azure Storage Platform en tant que solution de stockage dans le cloud pour tous les scénarios de stockage de données modernes.
- Il offre un ensemble d'objets hautement évolutif pour les objets de données.
- Un ensemble de messagerie fiable basé sur une file d'attente.
- Un ensemble NoSQL.
- Un stockage sur disque pour les machines virtuelles Azure et une plate-forme de partage de fichiers basée sur le cloud.

# Azure Storage Platform

- Azure propose un stockage :
  - Durable et hautement disponible.
  - Sécurisé.
  - Scalable.
  - Managé.
  - Facilement accessible.

# Azure Storage Platform

- Azure propose Les types de stockage suivants:
  - **Azure Blobs.**
  - **Azure Queues.**
  - **Azure Tables.**
  - **Azure Disks.**
  - **Azure Files.**
- Démo

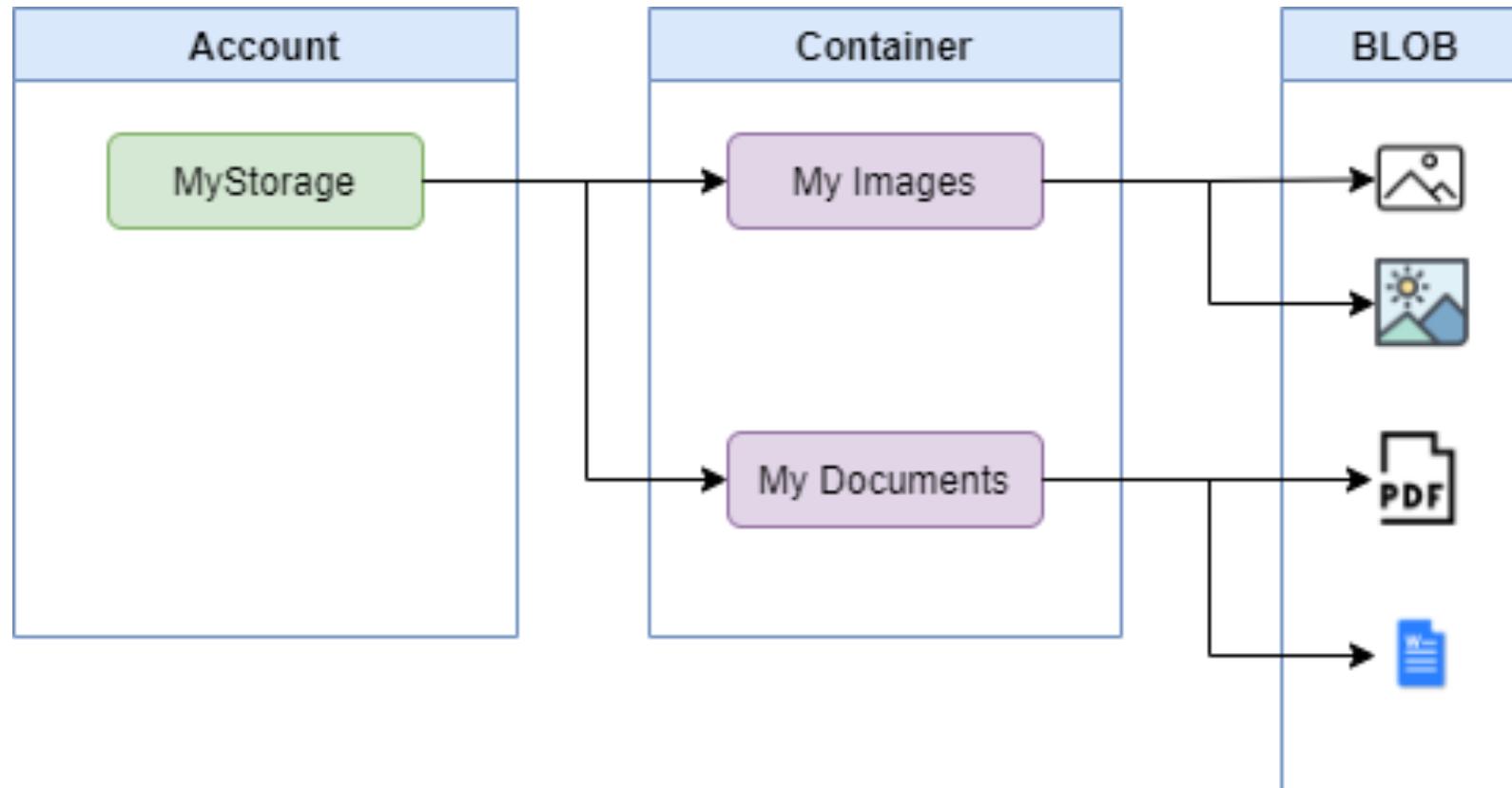
# Azure Storage Platform – Montage de volumes

- Le stockage Azure peut être utilisé comme une solution de montage de volume pour les conteneurs :
- Conteneur de service.
- Web App Service.
- Azure Kubernetes Service.
- Démo.

# Azure Storage Platform – Azure Blobs

- Le stockage Blob est idéal pour de nombreux scénarios tels que :
  - Servir des images ou des documents directement via le navigateur.
  - Stockage de fichiers pour un accès à partir de plusieurs emplacements et services.
  - Streaming de fichiers vidéo et audio.
  - Stockage des données pour les opérations de sauvegarde et de restauration, la reprise après sinistre, l'archivage, etc.
  - Stockage pour l'analyse des données par les services sur site et hébergés par Azure.

# Azure Storage Platform – Azure Blobs



# Azure Storage Platform – Azure Blobs

- Création d'une application DotNet Core avec Azure Blobs.

# Introduction Microsoft Azure SQL Database

- Microsoft Azure SQL Database est une base de données relationnelle en tant que service fiable et sécurisée, qui offre des performances élevées sans avoir à se soucier d'aucune infrastructure.



# Introduction Microsoft Azure SQL Database

- Azure Sql Database prend en charge les structures de données relationnelles, JSON, XML et spatiales.
- Microsoft Azure SQL Database propose trois options de déploiement :
  - Base de données unique
  - Elastic pool
  - Instances gérées

# Azure Sql Vs Sql Server

- Azure SQL est une plate-forme de base de données relationnelle présente dans le cloud où les utilisateurs peuvent héberger les données et les utiliser en tant que service. Vous pouvez payer pour ce que vous avez utilisé, comme tous les autres services cloud.
- Azure SQL est construit sur la base de SQL Server et il peut donc être assez déroutant de faire un choix entre les deux car ils partagent des qualités similaires.
- Bien qu'ils soient familiers, il existe des différences assez évidentes qui peuvent vous aider à décider lequel choisir.

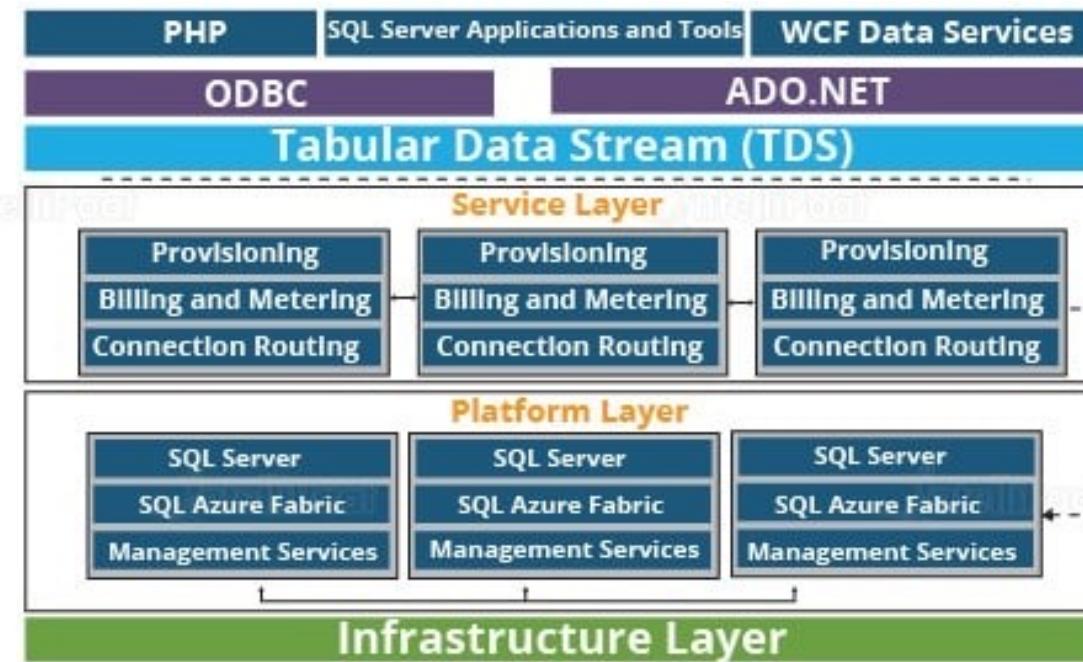
# Azure Sql Vs Sql Server

| Azure SQL                                                                           | SQL Server                                                 |
|-------------------------------------------------------------------------------------|------------------------------------------------------------|
| Une base de données peut héberger plusieurs bases de données de différents clients. | Les bases de données sont les seuls objets sur le serveur. |
| Il utilise le protocole Tabular Data Stream (TDS).                                  | Il utilise le protocole TCP/IP pour la communication.      |
| La communication directe n'est pas possible en raison de l'architecture complexe.   | Une communication directe peut avoir lieu.                 |
| La gestion et l'administration sont faciles.                                        | Il est difficile à mettre en place et à administrer.       |
| Il est facile à utiliser car vous n'avez besoin d'aucun matériel physique.          | Utilise un système physique                                |
| Sauvegarde automatique                                                              | Planification des sauvegardes manuelles                    |

# Sql Azure Architecture

- Il existe quatre couches dans Azure SQL Architecture :
  - Couche Client
  - Couche de service
  - Couche de plate-forme
  - Couche d'infrastructure

# Sql Azure Architecture



IntelliPaat

# Azure Sql database

- Démo

# Azure Messaging Services

- Microsoft Azure propose une multitude de service pour le broadcast de messages:
- **Storage Queue**
- **Service Bus :: Enterprise Messaging**
- **Event Hub :: Streaming platform**
- **Event Grid :: Event Distribution**

# Azure Messaging Services

|                                                                                   |                                                                                   |                                                                                    |                                                                                     |                                                                                     |                                                                                     |
|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|  |  |  |  |  |  |
| <b>Notification Hubs</b>                                                          | <b>Service Bus &amp; Azure Queues</b>                                             | <b>Event Hubs</b>                                                                  | <b>Event Grid</b>                                                                   | <b>IoT Hub</b>                                                                      | <b>Relay</b>                                                                        |
| Mobile push notifications                                                         | Cloud messaging                                                                   | Telemetry stream ingestion                                                         | Event distribution                                                                  | IoT messaging and management                                                        | Discovery, Firewall/NAT Traversal                                                   |

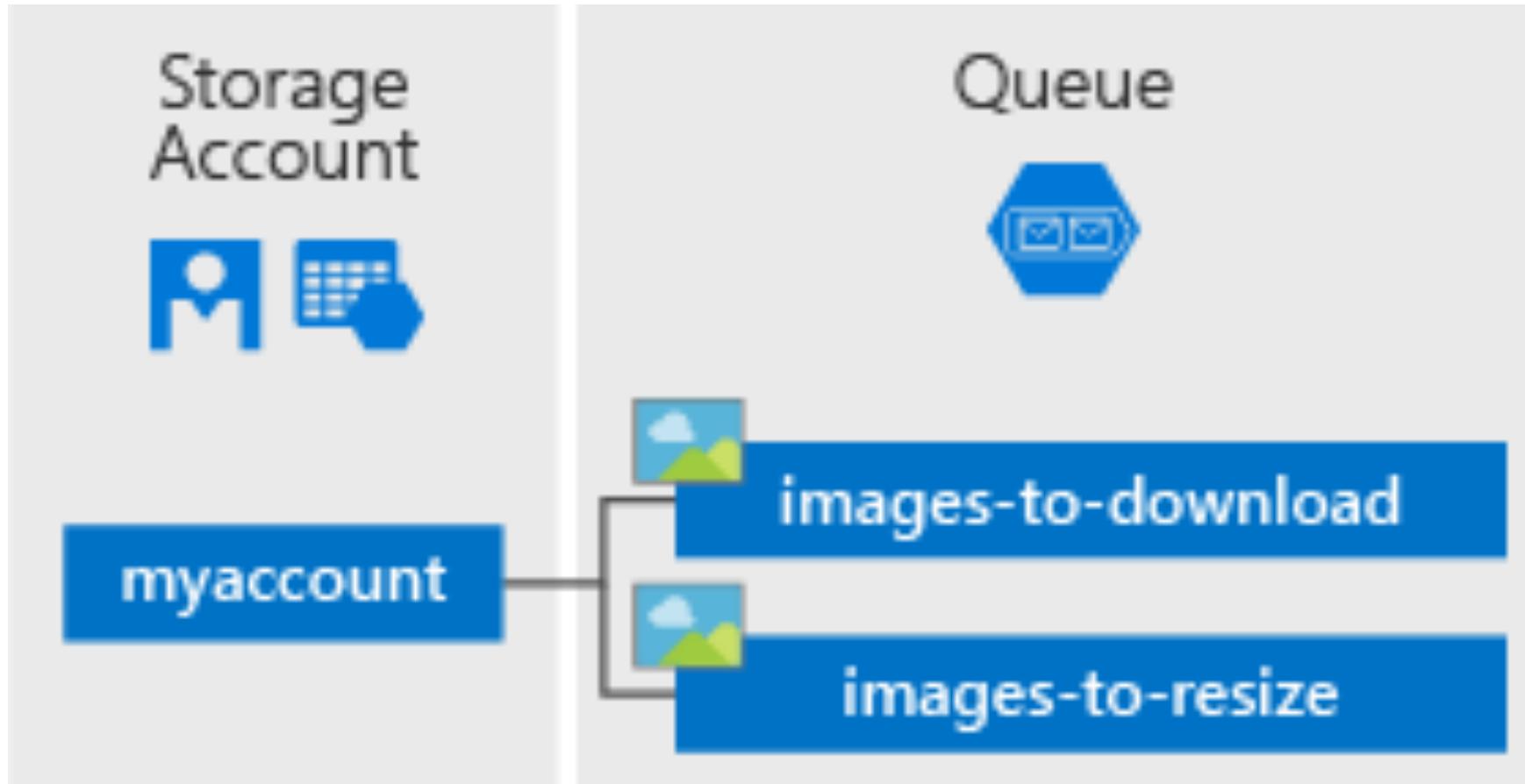


Microsoft Azure

# Azure Messaging Services – Storage Queue

- Storage Queue est mécanisme de stockage de message
- Simple queue
- Max. 64 KB par message.
- Max. 7 days.
- FIFO.
- Classement non garanti.

# Azure Messaging Services – Storage Queue



# Azure Messaging Services – Storage Queue

- Les opérations autorisées sont :
  - Put
  - Get
  - Peek
  - Delete
  - Clear
  - Update

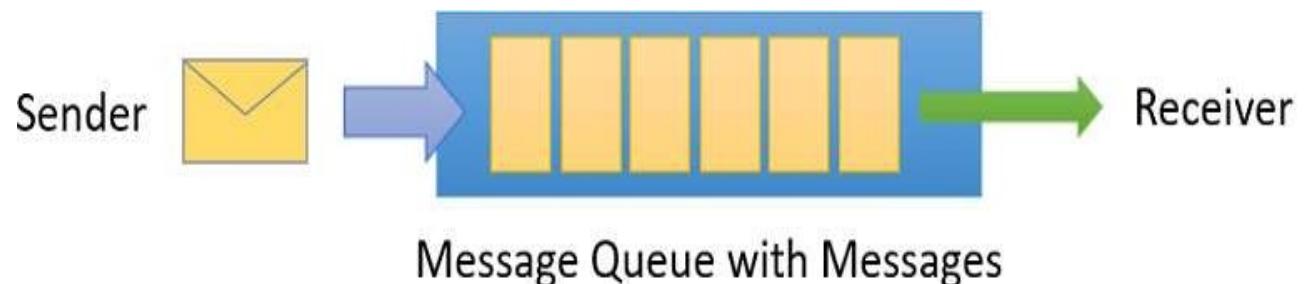
# Azure Messaging Services – Storage Queue

- Démo

# Azure Service Bus | Components

- Namespace : Définit le scopes des messages de notre application

- Queue >> point-to-point messaging
- Topic
- Subscription



- Gestion des composants ASB se fait :

- Portail
- Code
- PowerShell
- Http/Rest
- Azure Resource Manager



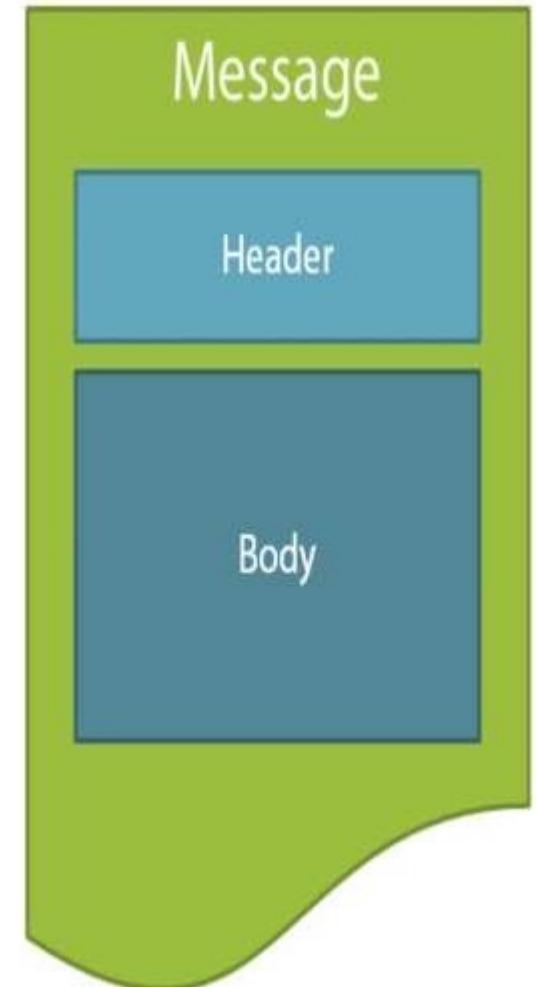
# Demo #1

- Création d'un Namespace
- Création d'une Queue
- Création Topic & Subscription



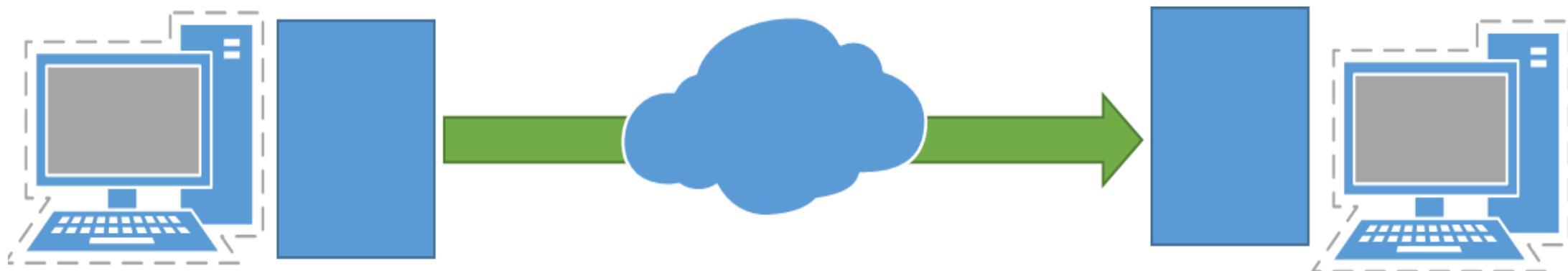
# Azure Service Bus | Message

- Header - Properties
  - MessageId
  - SessionId
  - Label
  - CorrelationId
  - UserProperties <Dictionary>
    - SequenceNumber.
    - EnqueuedSequenceNumber.
- Body
  - Content (Binary) » Serialized Object
- Limitation
  - Size = 256KB >> 1MB
  - Header Size = 64 KB



# Demo : Message

- Création Message
- Envoie et réception
- Modes de réceptions
- Réception de batch
- Détection de duplication

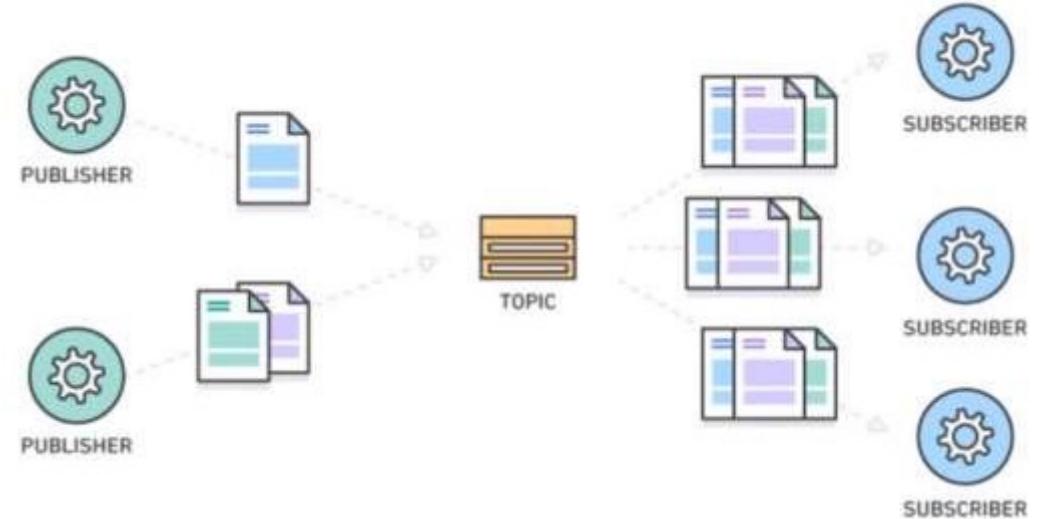


# Message | Pub/Sub

- Publish-Subscribe messaging
- Topic / Subscription
- Subscription Rules (Filters & Actions)

- Filters
- Actions

- NOT - AND - { = | <> | != | > | >= | < | <= }
- IS [NOT] NULL - [NOT] IN () - [NOT] LIKE
- sys.Label LIKE '%bus%' OR `user.tag` IN ('queue', 'topic', 'subscription')
- sys.Label = 'important'



# Azure Service Bus

- Message sessions »
- Auto-forwarding »
- Dead-lettering »
- Scheduled delivery »
- Message deferral »
- Batching »
- Transactions »
- Filtering and actions »
- Auto-delete on idle »
- Duplicate detection »
- Handling Large Messages » claim-check pattern

# Azure DevOps



Azure  
Boards



Azure  
Repos



Azure  
Pipelines



Azure  
Test Plans



Azure  
Artifacts

Plan, track, and discuss work across teams, deliver value to your users faster.

Unlimited cloud-hosted private Git repos. Collaborative pull requests, advanced file management, and more.

CI/CD that works with any language, platform, and cloud. Connect to GitHub or any Git provider and deploy continuously to any cloud.

The test management and exploratory testing toolkit that lets you ship with confidence.

Create, host, and share packages. Easily add artifacts to CI/CD pipelines.

# Azure Boards



Azure  
Boards

Plan, track, and discuss  
work across teams,  
deliver value to your  
users faster.

- **Gestion des tâches**
  - **Suivie**
  - **Statut**
  - **Planification**
- Backlogs & Sprints
- Queries
- Plans aligned with backlogs

# Azure Repos



Azure  
Repos

Unlimited cloud-hosted private Git repos. Collaborative pull requests, advanced file management, and more.

- **source code**
  - **Version control**
  - **Suivie des changements**
  - **Contrôle**
- ➔ Files in the project
  - ➔ TFVC or Git (commits)
  - ➔ Pushes and Branches
  - ➔ Pull Requests
  - ➔ Documentation

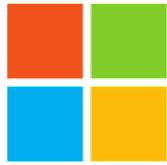
# Azure Pipeline



Azure  
Pipelines

CI/CD that works with any language, platform, and cloud. Connect to GitHub or any Git provider and deploy continuously to any cloud.

- Compile the app
- Release the app
- Library
- Task Group
- Deployment gr.
- XAML



- ➔ Builds from the repository
- ➔ Releases
- ➔ Manage variables or secure files
- ➔ Customize the pipeline
- ➔ Machines (Windows or Linux)
- ➔ Old building model



# Azure Test Plan



Azure  
Test Plans

The test management and exploratory testing toolkit that lets you ship with confidence.

- Test Plans
- Parameters
- Configurations
- Runs
- Load test

- ➔ Test Suites & Test Cases
- ➔ Customize the test cases data
- ➔ Different OS, browsers, i.e.
- ➔ Execute the tests
- ➔ Check responsiveness of the app

# Azure Artifacts



Azure  
Artifacts

Create, host, and  
share packages.  
Easily add artifacts  
to CI/CD pipelines.



- Crédit de nos propres packages

- NuGet → .NET packages
- npm → JavaScript packages
- Maven → Java packages
- Gradle → Java packages
- Universal → different packages

# Azure Boards

- Azure Boards est un ensemble d'outils interactifs pour gérer un projet logiciel.
- Azure Boards fournit un ensemble de fonctionnalités pour la gestion des processus en :
  - Agile.
  - Scrum.
  - Kanban.
- Azure Boards permet également un suivi pour :
  - Les tâches.
  - Les problèmes et bugs.
  - Les défauts de code.

# Azure Boards – Rappel Kanban

- Kanban est un board qui permet de visualiser le workflow des tâches à effectuer dans un projet.
- Un tableau Kanban transforme un backlog linéaire en un tableau interactif à deux tableau dimensionnel, fournissant un flux visuel de travail.
- Au fur et à mesure que le travail progresse, l'équipe met à jour le éléments au tableau.
- Cette visualisation rend transparente la progression de l'élément en cours ou manque de progrès.

# Azure Boards - Kanban

- Les tableaux Kanban suivent les exigences, sont indépendants du sprint, fournissent un organigramme pour suivre la progression des différents éléments.
- Les tableaux Kanban permettent de suivre :
  - Des produits.
  - Des backlogs items.
  - Des user stories.
  - Des bugs.
- Les tableaux de tâches sont associés à un sprint et sont généralement utilisés par les équipes Scrum.

| Backlog                                                                                                                                                  | Active                                                                                                                                                               | Resolved                                                                     | Closed                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|----------------------------------------------------|
| <p>+ New item</p> <p>532 Hello World Web Site<br/>Jamal Hartnett</p> <p>398 Cancel order form<br/>Jamal Hartnett</p> <p>Phone Service Web</p> <p>0/1</p> | <p>486 Welcome back page<br/>Raisa Pokrovskaya</p> <p>346 Add animated emoticons<br/>Christie Church</p> <p>Slow response on form<br/>Christie Church</p> <p>0/1</p> | <p>344 Implement a factory which abstracts<br/>Jamal Hartnett</p> <p>0/1</p> | <p>405 GPS locator<br/>Jamal Hartnett</p> <p>8</p> |

# Azure Boards – Kanban - Checklists

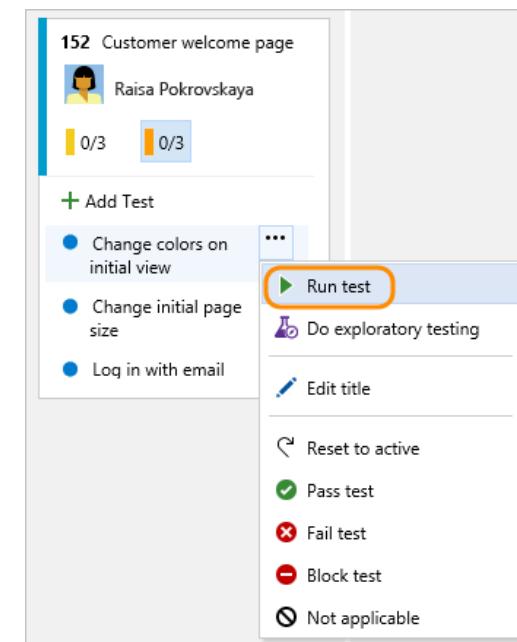
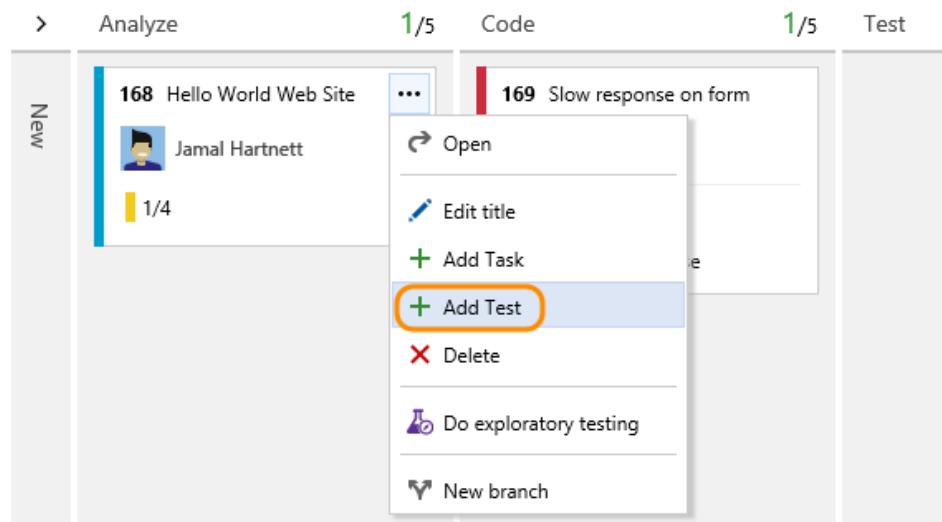
- Les checklists de tâches fournissent un moyen facile et visuel de suivre les détails du travail d'un élément du tableau
- Les tâches d'une checklists peuvent être:
  - Crée directement à partir du Tableau Kanban
  - Réorganiser en drag and drop
- Marqué comme terminé en cochant simplement la case

The screenshot shows a Kanban board with three columns: Analyze, Develop, and Test. Each column has a summary card at the top indicating the count of tasks.

- Analyze:** 1/10 tasks. A task titled "352 Hello World Web Site" is shown, assigned to Raisa Pokrovskaya, state Approved, iteration Sprint 3. It has a checklist with 1/3 items completed. The checklist includes: Fix performance issues (unchecked), Auto-complete user's information (unchecked), and Auto-save (checked).
- Develop:** 1/6 tasks. A task titled "1069 Permissions" is shown, unassigned, state Committed, iteration Sprint 3. It has a checklist with 0/1 items completed. The checklist includes: Performance issues (unchecked), Secure sign-in (checked), and Check issues with permissions (unchecked).
- Test:** 1/6 tasks. A task titled "364 Slow response on information form" is shown, assigned to Jamal Hartnett, state Committed, iteration Sprint 3. It has a checklist with 1/3 items completed. The checklist includes: Benchmark performance (unchecked), Apply UI updates (checked), and Insert telemetry code (checked).

# Azure Boards – Kanban - Tests

- De la même façon que les checklists, le tableau kanban permet de définir un ensemble de tests pour un élément.
- Les tests peuvent être démarrer à partir du tableau.
- Le tests utilisent Azure plan Tests.



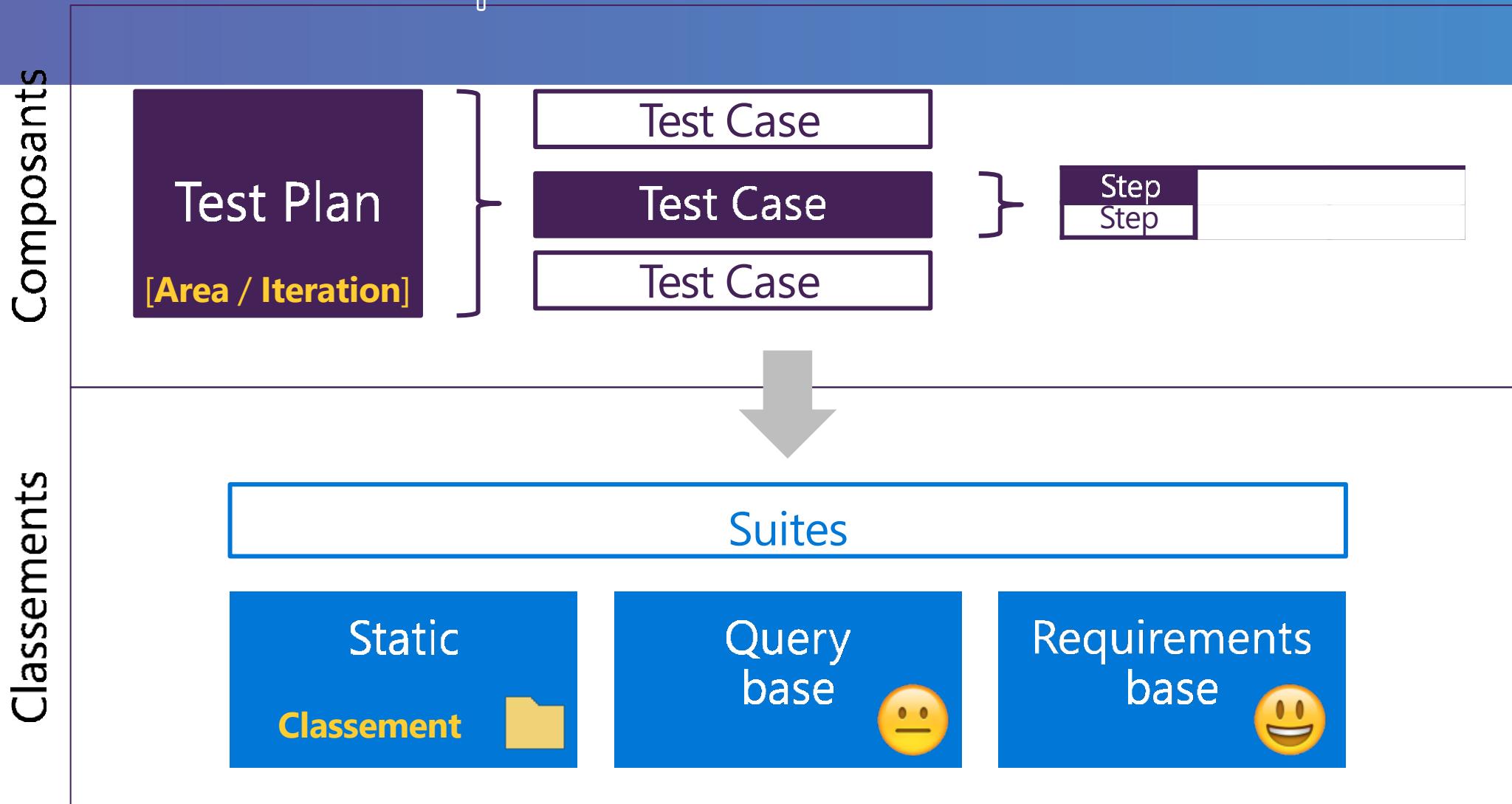
# Azure Boards – Rappel Scrum

- Scrum est un ensemble d'outils et de fonctionnalités pour gérer la progression du travail à l'intérieur d'une équipe.
- Scrum utilise un cycle de vie itératif appelé sprint.
- Scrum utilise un ensemble de rôle.
  - Product owner.
  - Scrum Master.
  - Equipe scrum.
- Scrum organise le travail sous forme d'une liste hiérarchisée appelée Backlog produits.

# Tests dans Azure DevOps

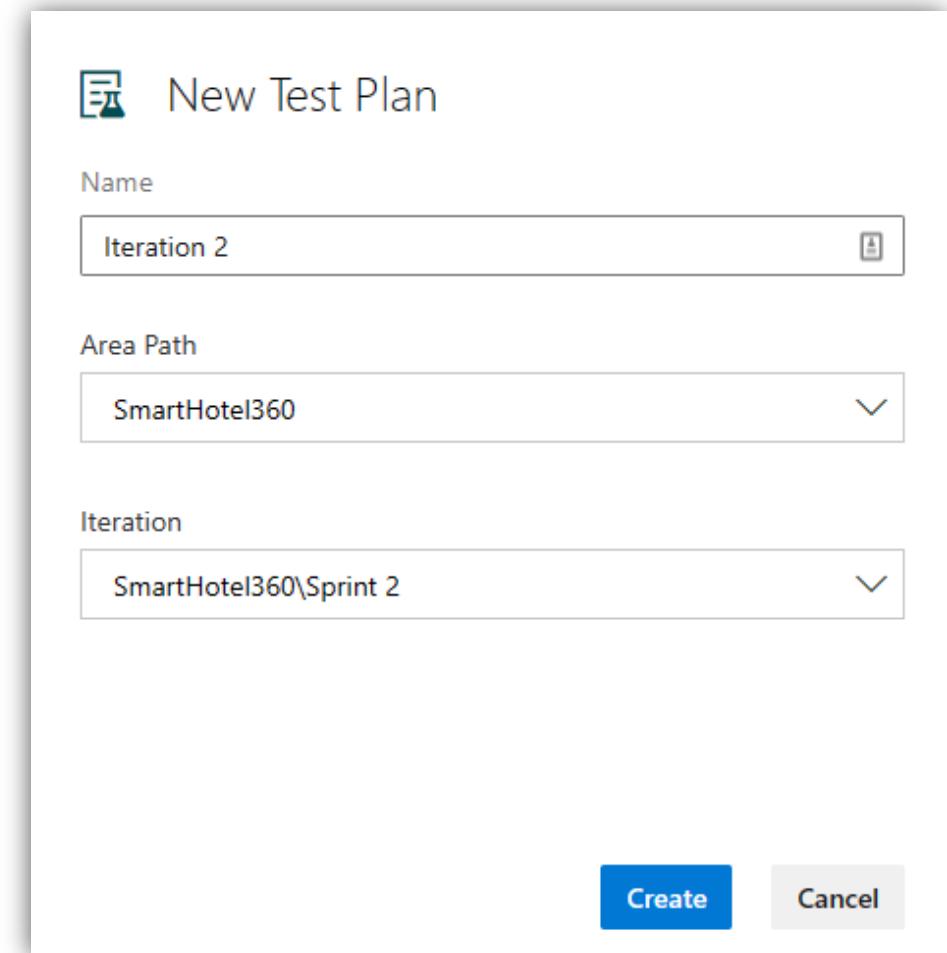
- Azure Test Plans est un service qui fournit un ensemble de fonctionnalités pour exécuter un ensemble de tests.
- Azure Test Plans prend en charge :
  - Tests manuels.
    - Tests manuels planifiés.
    - Test d'acceptation de l'utilisateur.
    - Tests exploratoires.
  - Tests automatisés.
  - Suivi de tests
  - Rapports et analyses.

# Structure des plans de tests



# Test Plans - Crédit

« Campagne de tests correspondant à un périmètre fonctionnel et de temps dans lequel les tests vont être créés et exécutés. »



# Les cas de Tests

The screenshot shows a software interface for managing test cases. On the left, a sidebar displays the current sprint (Sprint 1) which is past, running from Oct 27 - Nov 3, with 6% run and 0% passed. It lists three test suites: Sprint 1, 33 : Change initial view (7), 34 : Welcome back pa... (..), and 43 : Cancel order form (1). The second item, 33 : Change initial view (7), is selected and expanded. On the right, a detailed view of this test suite is shown with a title "33 : Change initial view (ID: 68)". Below it is a table titled "Test Cases (7 items)" with columns for Title, Order, Test Case Id, and State. The table lists seven test cases, all of which are currently in the "Design" state:

| Title                         | Order | Test Case Id | State  |
|-------------------------------|-------|--------------|--------|
| Test welcome page             | 1     | 76           | Design |
| Change initial view           | 2     | 79           | Design |
| Welcome back                  | 3     | 80           | Design |
| Resume                        | 4     | 81           | Design |
| Interim save on long form     | 5     | 82           | Design |
| Change colors on initial view | 6     | 83           | Design |
| Revert to previous version    | 7     | 84           | Design |

Product Backlog Items

Test Cases

# Les cas de Tests – Etapes

TEST CASE 83  
83 Change colors on initial view

Jamal Hartnett 0 comments Add tag Save & Close Follow ...

State: Design Area: Fabrikam Fiber Updated: Nov 3  
Reason: New Iteration: Fabrikam Fiber\Release 2\Sprint 2

Steps Summary Associated Automation Scans (4) (1)

Steps Deployment

To track releases associated with this work item, go to [Releases](#) and turn on deployment status reporting for Boards in your pipeline's Options menu. [Learn more about deployment status reporting](#)

Steps Action Expected result

1. Open the home page for the web site. Home page is displayed

2. Click Settings icon Settings page is displayed

3. Change the default template to Modern and select Submit The home page is displayed with the Modern Look. See screenshot.

Click or type here to add a step

Add link Development

Link an Azure Repos commit, pull request or branch to see the status of your development. You can also [create a branch](#) to get started.

Related Work

Add link Add an existing work item as a parent

Tests (2)

85 Settings page does not display Updated 11/5/2021, ● New

33 Change initial view Updated 5/22/2015, ● Active

Related (2)

111 Change colors on initial view Updated 11/10/2021, ● Design

144 Change colors on initial view Updated 11/17/2021, ● Design

Parameter values

Add a shared parameter set | Convert to shared parameters

# Les cas de Tests – Grille

The screenshot shows a test management interface with a sidebar on the left and a main content area on the right.

**Left Sidebar:**

- S: Iteration 2 (dropdown)
- Nov 4 - Nov 25 (date range)
- 5% run, 100% passed. [View report](#)
- + (New Test Suite)
- Test Suites
- Iteration 2 (selected):
  - 770 : As a room guest, I should be able to communicate with all smart devices from the app (ID: 892)
  - 771 : As a room guest, I should ... .
  - 780 : As a user, I should be a... .
  - 766 : As a customer, I should ... .
  - 769 : As a customer, I should ... .
  - 765 : As a customer, I should ... .
  - 764 : As a reservation agent, I... .
  - 763 : As a customer, I would l... .
- ⚙️ (Settings)

**Main Content Area:**

770 : As a room guest, I should be able to communicate with all smart devices from the app (ID: 892)

Define Execute Chart [Close Grid](#)

| ID  | Title                                                                                                              | Step Action                                          | Step Expected Result                                |
|-----|--------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|-----------------------------------------------------|
| 861 | Verify that a notification is auto-closed after X seconds                                                          | Click the notification icon #Browser                 | The notification should appear c                    |
|     |                                                                                                                    | Wait for X seconds with the notification window open | Once the X seconds are passed, window should close. |
| 864 | Verify that the application's display is adapted to the screen size and all buttons and menus are easily clickable | Launch the application                               |                                                     |
|     |                                                                                                                    | Check the screen size adaption                       |                                                     |
|     |                                                                                                                    | Check whether all buttons are working                |                                                     |
|     |                                                                                                                    | Log the results                                      |                                                     |
|     |                                                                                                                    | Close the application                                |                                                     |

# Les Suite de tests

← Iteration 2 ★

Nov 4 - Nov 25 Current

0% run. [View report](#)

**Test Suites**

Iteration 2

- 770 : As a room guest, I ...
- 771 : As a room guest, I ...
- 780 : As a user, I should ...
- 766 : As a customer, I sh ...
- 769 : As a customer, I sh ...
- 765 : As a customer, I should be able to remove a car r... ...
- 764 : As a reservation agent, I would like to send confir... ...
- 763 : As a customer, I would like to reserve a conferenc... ...

**⋮**

**⋮**

**New Suite >**

- Static suite
- Requirement based suite
- Query based suite

**⋮**

**⋮**

Static



Query base

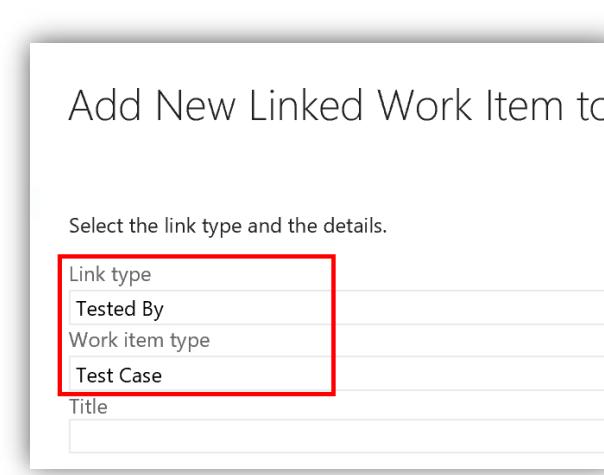
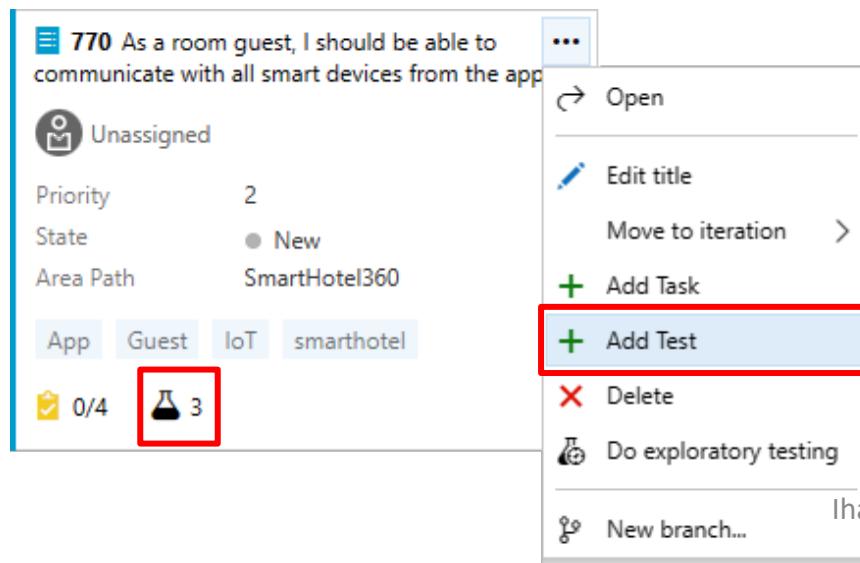


Requirements base



# Les Suite de tests – Bonne pratiques

- Test Case sont à associer aux users stories.
- Création d'un Requirement base suite (dynamique).
- Ajouter des tests qui seront liés à la Story.
- Ajouter des tests depuis le Azuez Boards.



# Les Suite de tests – Bonne pratiques

The screenshot shows a Kanban board in the Microsoft Azure DevOps Boards interface. The board is organized into four columns: New, Approved, Build and Test, and Deploy. Each column displays a list of backlog items (cards) with their status, priority, state, area path, and other details.

- New Column:** Contains 3 cards:
  - Card 763: As a customer, I would like to reserve a conference room. Status: Unassigned, Priority: 2, State: New, Area Path: SmartHotel360. Sub-tasks: Reservation (0/2), Lab (2).
  - Card 764: As a reservation agent, I would like to send confirmations to guest. Status: Unassigned, Priority: 2, State: New, Area Path: SmartHotel360. Sub-tasks: App (0/1), Guests (0/1), Lights (0/1), Rooms [List] (0/1).
  - Card 0/1: Status: Unassigned, Priority: 2, State: New, Area Path: SmartHotel360.
- Approved Column:** Contains 4 cards:
  - Card 785: As a room guest, I should be able to control my room's lighting within the app. Status: Unassigned, Priority: 2, State: Approved, Area Path: SmartHotel360. Sub-tasks: App (0/1), Guests (0/1), Lights (0/1), Rooms [List] (0/1).
  - Card 786: Search hotel works only for certain cities. Status: Unassigned, Priority: 2, State: Approved, Area Path: SmartHotel360. Sub-tasks: App (0/1), Guests (0/1), Lights (0/1), Rooms [List] (0/1).
  - Card 787: Search hotel bug. Status: Unassigned, Priority: 2, State: Approved, Area Path: SmartHotel360. Sub-tasks: App (0/1), Guests (0/1), Lights (0/1), Rooms [List] (0/1).
  - Card 788: Search hotel works only for certain cities. Status: Unassigned, Priority: 2, State: Approved, Area Path: SmartHotel360. Sub-tasks: App (0/1), Guests (0/1), Lights (0/1), Rooms [List] (0/1).
- Build and Test Column:** Contains 3 cards:
  - Card 789: Search hotel works only for certain cities. Status: Committed, Priority: 2, State: Committed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
  - Card 790: Search hotel bug. Status: Committed, Priority: 2, State: Committed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
  - Card 791: Search hotel works only for certain cities. Status: Committed, Priority: 2, State: Committed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
- Deploy Column:** Contains 5 cards:
  - Card 792: Search hotel works only for certain cities. Status: Deployed, Priority: 2, State: Deployed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
  - Card 793: Search hotel bug. Status: Deployed, Priority: 2, State: Deployed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
  - Card 794: Search hotel works only for certain cities. Status: Deployed, Priority: 2, State: Deployed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
  - Card 795: Search hotel bug. Status: Deployed, Priority: 2, State: Deployed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).
  - Card 796: Search hotel works only for certain cities. Status: Deployed, Priority: 2, State: Deployed, Area Path: SmartHotel360. Sub-tasks: Admin (1/5), Manager (1/5).

# Configuration & Paramètres

- Un test peut être répété plusieurs fois en utilisant différents environnements d'exécution (Win10, Win11, Chrome, Firefox, ...).
- Un test peut être répété plusieurs fois en utilisant des paramètres (locaux / partagés).
- Azure test plan permet de créer des étapes utilisées par plusieurs cas de tests.

# Configuration & Paramètres

The screenshot shows the Microsoft Test Plan interface. On the left, the navigation bar includes: Overview, Boards, Repos, Pipelines, Test Plans (selected), Test plans, Progress report, Parameters, Configurations (selected), Runs, and Load test. The main area displays 'All test configurations' with entries for 'Win10 + Edge' and 'Win10 + Firefox'. Below this, 'All configuration variables' are listed under 'Operating System' and 'Browser'. A configuration variable 'Win10 + Edge' is selected and shown in detail. The details pane includes fields for Name (highlighted with a red box), Description, State (Active), and an 'Assign to new test plans' checkbox. Under 'Configuration variables', there are entries for 'Browser' (Microsoft Edge) and 'Operating System' (Windows 10). A link to 'Add configuration variable' is also present.

| Name             | Value          |
|------------------|----------------|
| Browser          | Microsoft Edge |
| Operating System | Windows 10     |

# Configuration & Paramètres

The screenshot shows the SmartTest 2020 application interface. On the left, the navigation bar includes 'GOVOLUTION / SMARTTEST 2020 / TEST SUITES / ITERATION 2'. The main area displays a requirement titled '770 : As a room guest, I should be able to communicate with all smart devices from the app (ID: 892)'. Below it, there are tabs for 'Define', 'Execute' (which is selected), and 'Chart'. A sub-section titled 'Test Points (3 items)' lists several options like 'Title', 'Verify that a notification', etc. To the right, a modal window titled 'Assign configurations to test suite - 770 : As a room guest, I ...' is open. It contains a search bar and a table with two rows. The first row is 'Win10 + Edge' with values 'Browser:Microsoft Edge;Operating System:Windows 10'. The second row is 'Win10 + Firefox' with values 'Browser:FireFox;Operating System:Windows 10'. Both rows have checkboxes next to them, and the first one is checked. The entire 'Assign configurations' option in the context menu is highlighted with a red box.

| Configuration name ↑                                | Configuration values                               |
|-----------------------------------------------------|----------------------------------------------------|
| <input checked="" type="checkbox"/> Win10 + Edge    | Browser:Microsoft Edge;Operating System:Windows 10 |
| <input checked="" type="checkbox"/> Win10 + Firefox | Browser:FireFox;Operating System:Windows 10        |

Save Cancel

# Configuration & Paramètres

- Un test peut être répété plusieurs fois en utilisant des paramètres

Steps

| Steps | Action                                                                                           | Expected result                                                                                 | Attachments |
|-------|--------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|-------------|
| 1.    | Launch the application and login into the application as a customer using @Login.                | Customer should be able to login into the application with his credentials                      |             |
| 2.    | Click on the Profile and select Saved Cards                                                      | Customer should get navigated to Saved cards page.                                              |             |
| 3.    | Click on ADD button to add new card details and enter invalid card details, Click on Save button | Customer should not be allowed to save the invalid card details. The error @Error is displayed. |             |

*Click or type here to add a step*

Parameter values

Add a shared parameter set | Convert to shared parameters

| Login | Error                   |
|-------|-------------------------|
| Admin | Error 0x47156 "Inv..."  |
| Jack  | This card is invalid... |

# Configuration & Paramètres

- Exécutions multiples

The screenshot shows a software interface for test execution. At the top, there's a toolbar with icons for file operations and a dropdown menu labeled "Test 1 of 1: Iteration 1". Below the toolbar, a test case is listed with the ID "873: Verify that user is not allowed to save...". The test steps are:

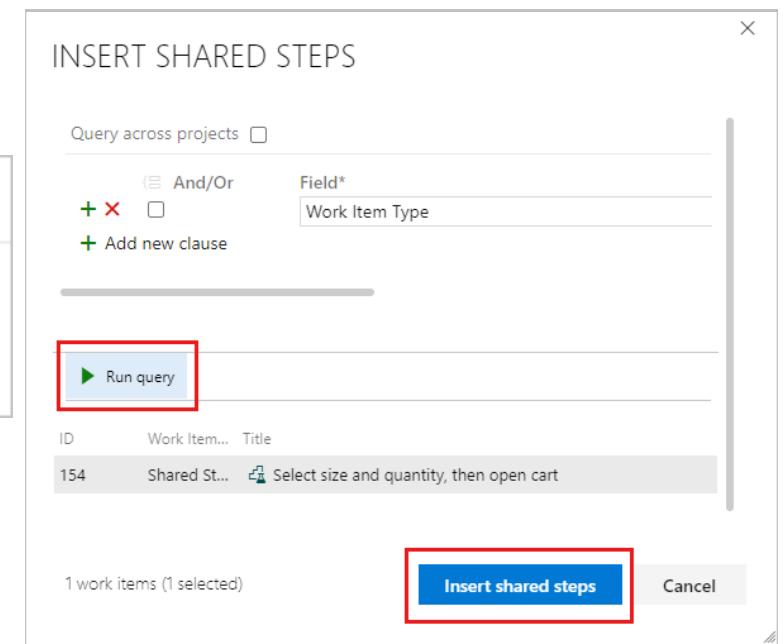
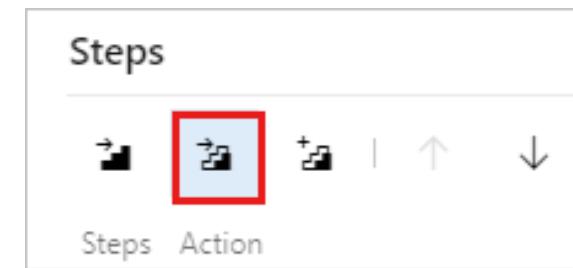
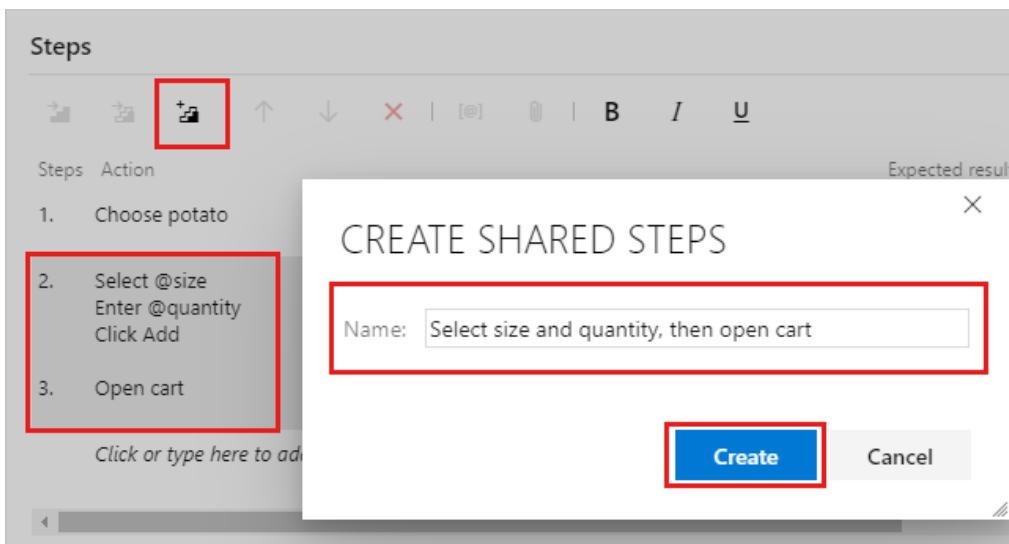
1. Launch the application and login into the application as a customer using @Login.  
EXPECTED RESULT  
Customer should be able to login into the application with his credentials.  
Login = Admin
2. Click on the Profile and select Saved Cards  
EXPECTED RESULT  
Customer should get navigated to Saved cards page.
3. Click on ADD button to add new card details and enter invalid card details, Click on Save button  
EXPECTED RESULT  
Customer should not be allowed to save the invalid card details. The error @Error is displayed.  
Error = Error 0x47156 "Invalid card"

The screenshot shows a software interface for test execution. At the top, there's a toolbar with icons for file operations and a dropdown menu labeled "Test 1 of 1: Iteration 2". Below the toolbar, a test case is listed with the ID "873: Verify that user is not allowed to save...". The test steps are:

1. Launch the application and login into the application as a customer using @Login.  
Login = Jack  
EXPECTED RESULT  
Customer should be able to login into the application with his credentials
2. Click on the Profile and select Saved Cards  
EXPECTED RESULT  
Customer should get navigated to Saved cards page.
3. Click on ADD button to add new card details and enter invalid card details, Click on Save button  
EXPECTED RESULT  
Customer should not be allowed to save the invalid card details. The error @Error is displayed.  
Error = This card is invalid. Contact the helpdesk

# Etapes partagées

- Les étapes partagées définissent une séquence d'étapes qui peuvent être référencées par de nombreux cas de test différents.



# Exécution des tests

- L'exécution des tests manuels se fait à l'aide Test Runner.
- Test Runner permet de :
  - Exécuter la totalité des tests actifs.
  - Exécuter un test spécifique.
  - Modifier les tests lors de l'exécution.
- L'exécution des tests peut être pour :
  - Une application web.
  - Une application bureau.

# Exécution des tests

The screenshot shows the Microsoft Test Plan interface. On the left, the navigation bar includes 'Fabrikam Fiber' (selected), 'Overview', 'Boards', 'Repos', 'Pipelines', 'Test Plans' (selected), 'Test plans' (highlighted with a red box), and 'Progress report'. The main area displays 'Sprint 2' (Nov 2 - Nov 9, Past) with a progress of 15% run, 40% passed. A 'Test Suites' section shows 'Sprint 2 (15)' containing 'Basic testing (8)' and 'Special testing (10)' (highlighted with a red box). A detailed view of 'Special testing (ID: 142)' is shown, with tabs for 'Define', 'Execute' (highlighted with a red box), and 'Chart'. Under 'Test Points (10 items)', several checkboxes are listed: 'Title', 'Interim save on long form', 'Interim save on long form', 'Change colors on initial view', 'Change colors on initial view', and 'Revert to previous version'. To the right, two windows are open: 'Test Suites' (Sprint 2 (6)) and 'Test Points (6 items)'. The 'Test Points' window has a dropdown menu set to 'Run for web application' (highlighted with a red box). Under 'Title', the checkbox 'Change initial view' is checked (highlighted with a red box).

# Exécution des tests

Runner - Test Plans - Work - Microsoft Edge  
https://fabrikamprime.visualstudio.com/Fabrikam%20Fi...

Test 1 of 2

83: Change colors on initial view

1. Open the home page for the web site.
- EXPECTED RESULT  
Home page is displayed
2. Click Settings icon
- EXPECTED RESULT  
Settings page is displayed
3. Change the default template to Modern and select Submit
- EXPECTED RESULT  
The home page is displayed with the Modern Look. See screenshot.



home-page-modern...

Runner - Test Plans - Work - Microsoft Edge  
https://fabrikamprime.visualstudio.com/Fabrikam%20Fi...

Test 1 of 2

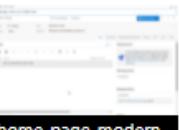
83\*: Change colors on initial view

1. Open the home page for the web site.
- EXPECTED RESULT  
Home page is displayed
2. Click Settings icon
- EXPECTED RESULT  
Settings page is displayed

COMMENT

Settings page does not open

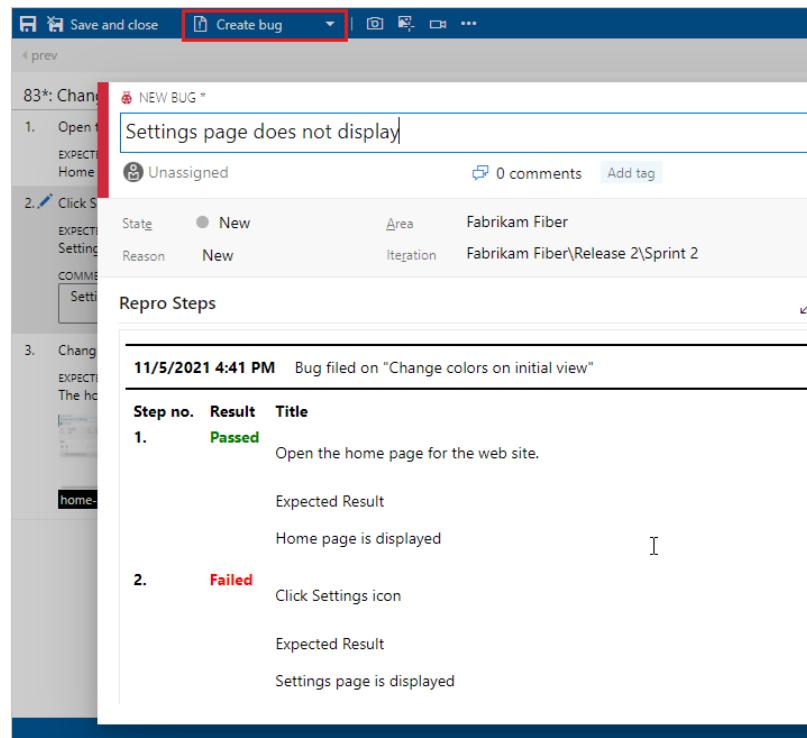
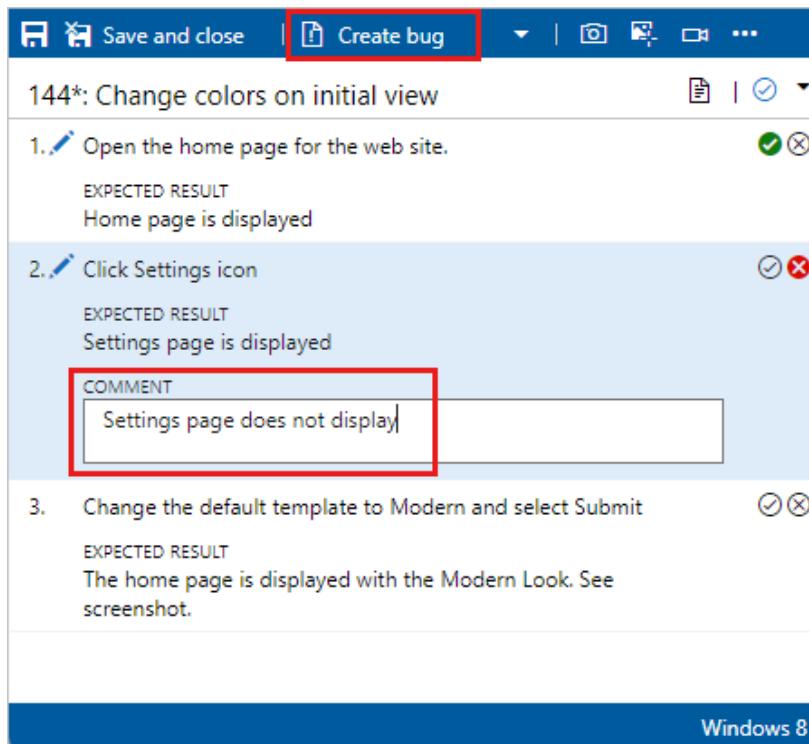
3. Change the default template to Modern and select Submit
- EXPECTED RESULT  
The home page is displayed with the Modern Look. See screenshot.



home-page-modern...

# Exécution des tests – Cr éation de bugs

- Test Plans permet d'envoyer des bugs et des commentaires en cas d'échec de tests.



# Exécution des tests – Capture d'action

- Test Plans permet également de faire soit des :
  - Capture d'images.
  - Capture d'enregistrement.

The screenshot shows a software interface for executing test cases. At the top, there's a toolbar with icons for Save and close, Create bug, and other functions. Below the toolbar, a list of steps is displayed for a specific test case:

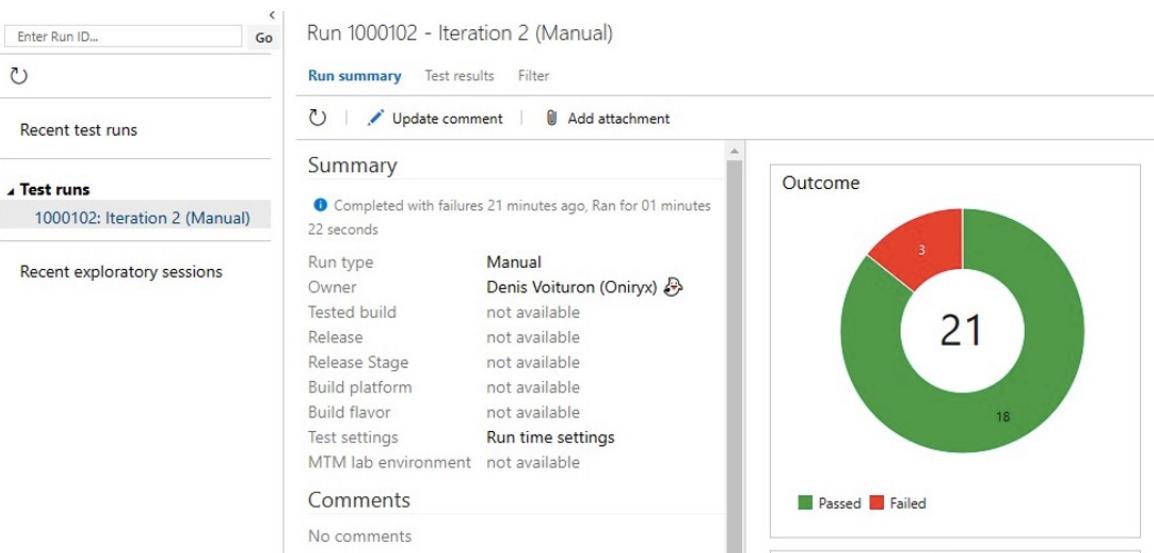
| Step | Action                               | Status |
|------|--------------------------------------|--------|
| 1.   | Open the home page for the web site. | ✓✗     |
| 2.   | Click Settings icon.                 | ✓✗     |
| 3.   | Select Use previous version          | ✓✗     |

This screenshot is identical to the one above, showing the same list of steps and their execution status for the same test case. The steps are:

| Step | Action                               | Status |
|------|--------------------------------------|--------|
| 1.   | Open the home page for the web site. | ✓✗     |
| 2.   | Click Settings icon.                 | ✓✗     |
| 3.   | Select Use previous version          | ✓✗     |

# Résultat d'exécution

- Test Plans permet :
  - D'avoir une visualisation globale sur les états des tests (réussite, échec, ...).
  - D'avoir un suivi détaillé des résultats test par test.
  - D'exporter les résultats sous forme de graphiques.



Run 1000102 - Iteration 2 (Manual) 21 results (1 selected)

Run summary Test results Filter

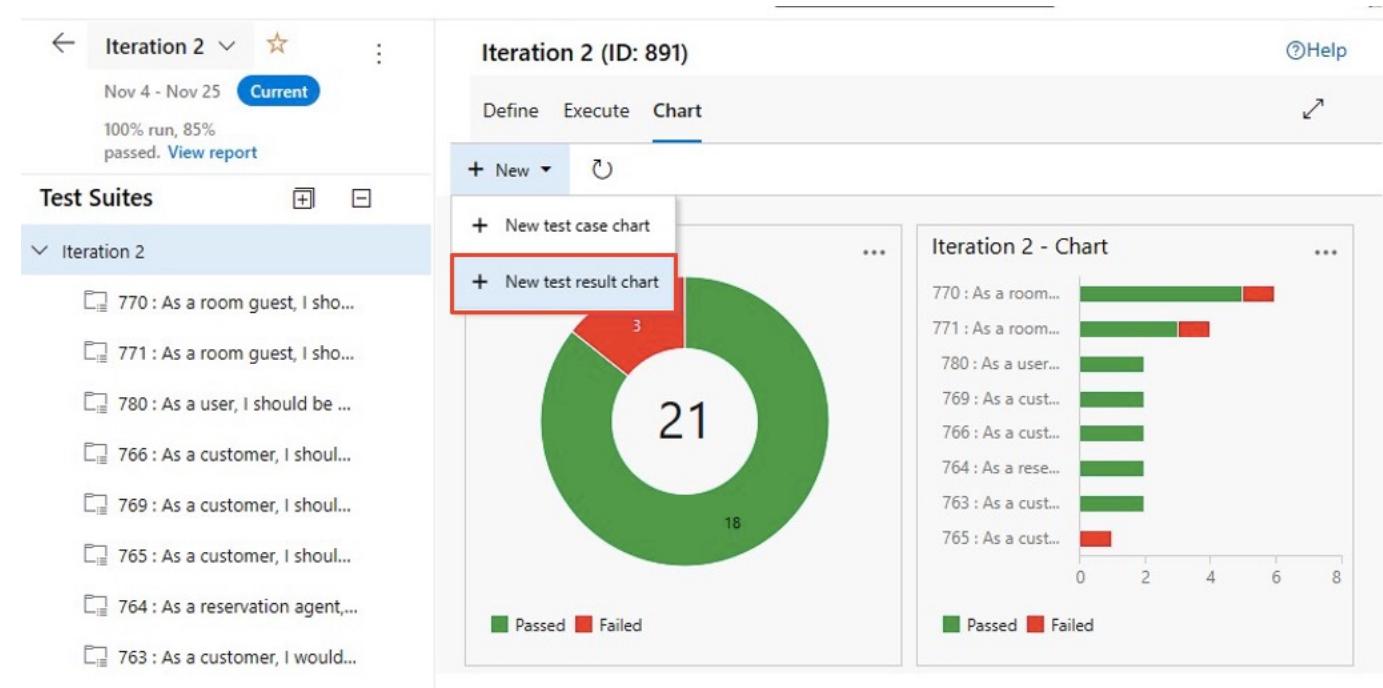
Recent test runs

Test runs 1000102: Iteration 2 (Manual)

Recent exploratory sessions

| Outcome | Test Case Title                                                                 | Priority | Duration    | Owner |
|---------|---------------------------------------------------------------------------------|----------|-------------|-------|
| Passed  | Verify that someone new to the app can log in with Facebook id                  | 2        | 0:00:05.000 | Denis |
| Passed  | Verify whether the application has been launched successfully or not.           | 2        | 0:00:08.790 | Denis |
| Passed  | Verify that a notification is auto-closed after X seconds                       | 2        | 0:00:10.770 | Denis |
| Failed  | Verify that a notification is auto-closed after X seconds                       | 2        | 0:00:14.130 | Denis |
| Passed  | Check that each screen is appropriately displayed in each display mode ...      | 2        | 0:00:17.720 | Denis |
| Passed  | Verify that the application's display is adapted to the screen size and all ... | 2        | 0:00:20.110 | Denis |
| Passed  | Verify that the application's display is adapted to the screen size and all ... | 2        | 0:00:22.270 | Denis |
| Passed  | Verify that a changed Facebook password will ask for re-login in the ap...      | 2        | 0:00:24.566 | Denis |
| Passed  | Verify that system allow user to update the fields in the reservation form.     | 2        | 0:00:26.686 | Denis |
| Passed  | Verify that System should allow user to checkout late with a later time v...    | 2        | 0:00:29.350 | Denis |
| Failed  | Verify that the reservation gets cancelled after click on Cancel order nu...    | 2        | 0:00:31.606 | Denis |

# Résultat d'exécution



# Azure pipelines

- Azure pipelines est un service qui, à partir d'un système de gestion de version (Github ou Azure repos), permet de :
  - Générer automatiquement les projets.
  - Tester automatiquement les projets.
  - Déployer automatiquement les projets.
- Azure pipelines combine L'intégration continue (CI) et le déploiement continu (CD).

# Azure pipelines – Rappel Intégration continue

- L'intégration continue permet aux développeurs d'automatiser la fusion et le test du code.
- CI permet de détecter les problèmes et bugs au début du cycle de développement.
- CI produit les artifacts qui permettent la mise en production et un déploiement fréquent.

# Azure pipelines – Rappel déploiement continue

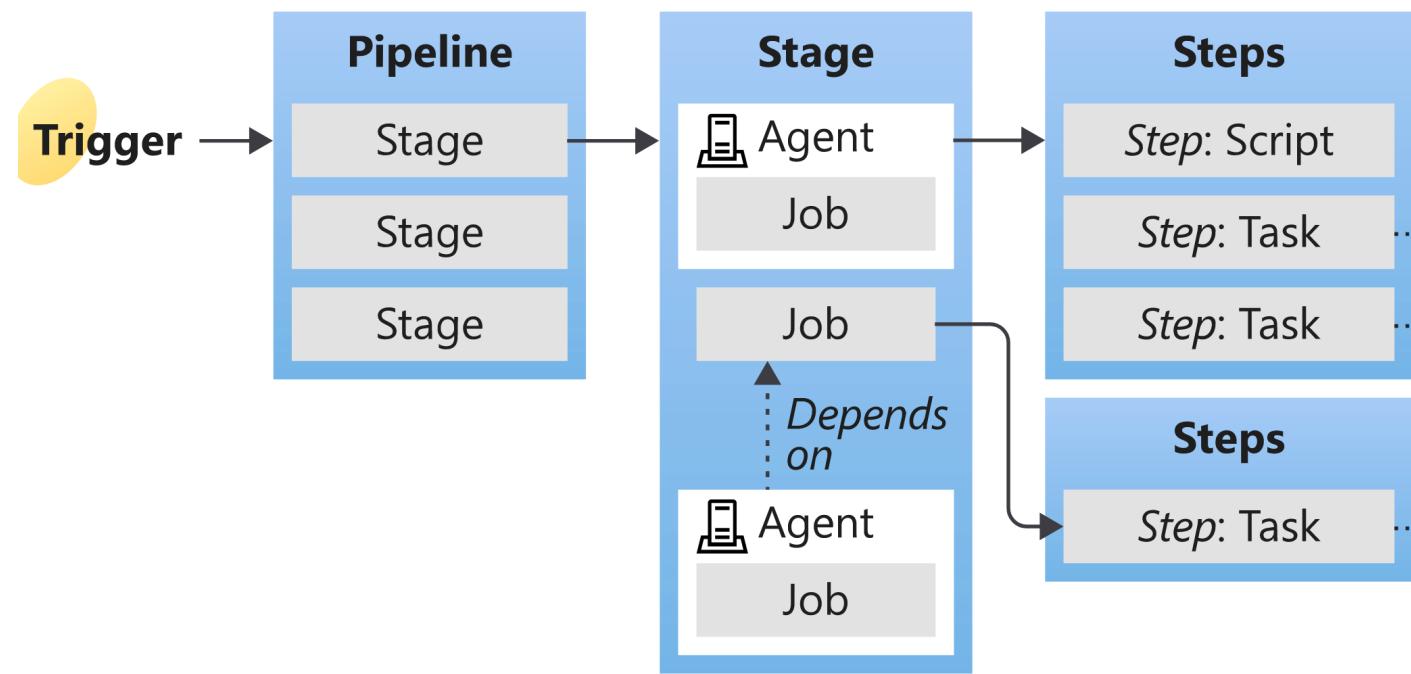
- Le déploiement continue permet de générer, tester et déployer nos applications sur un ou plusieurs environnements.
- Le déploiement continue consomme les artefacts produit par CI.
- Le déploiement continue permet d'améliorer la visibilité sur l'ensemble du processus à l'aide des systèmes de surveillance.

# Azure pipelines – Langages pris en charge

- Azure pipelines permet la génération d'applications développées à partir d'un ensemble de technologies prisent en charge.
- Actuellement les langages prisent en charge sont :
  - C#
  - C++
  - JAVA
  - PHP
  - Javascript
  - Python
  - Ruby
  - Golang
- Azure devops utilise le mécanisme de tâche pour générer nos applications.

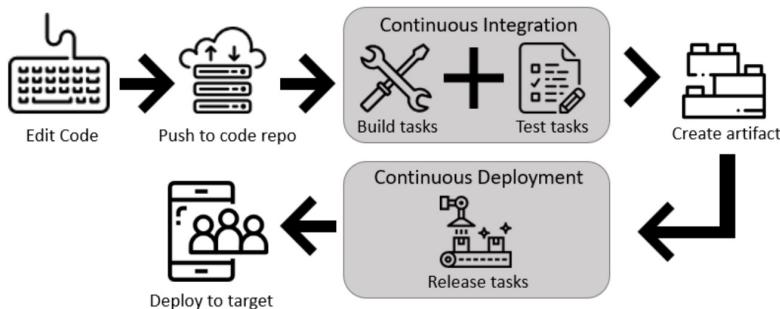
# Azure pipelines – fonctionnement

- Azure pipelines démarre l'exécution à partir d'un déclencheur (un push sur un dépôt, un autre build...).
- Une pipeline est constitué d'un ou plusieurs stages.
- Une pipeline est déployé sur un ou plusieurs environnements.
- Les stages ont un ou plusieurs jobs.
- Un job est exécuté sur un agent et est constitué de steps.
- Chaque step est une tâche ou un script.
- Une tâche est un script pré-build qui effectue une action.

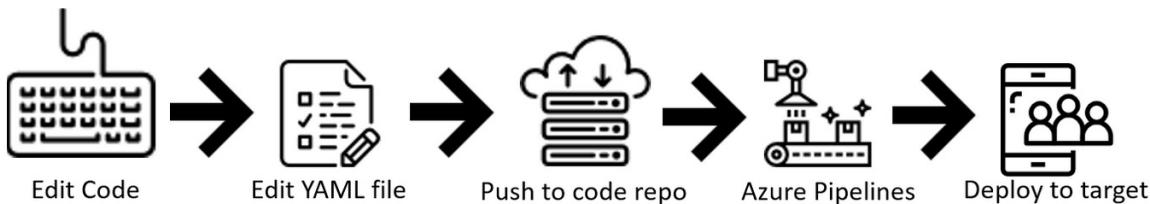


# Azure pipelines – utilisation

- Azure pipeline fonctionne :
  - A partir de l'éditeur azure pipelines de l'interface web.



- A partir d'un fichier YAML appelé `azure-pipelines.yml`.
- Ce fichier est à placer à l'intérieur du gestionnaire de version de notre application (dépôt git).



# Azure pipelines – utilisation

- Certaines fonctionnalités d’Azure pipelines ne sont pas disponibles avec à la fois YAML et le build classique.
- Exemple les travaux de conteneur sont disponible uniquement en YAML

# Azure pipelines – Déploiement des images.

- Pour déployer une image de conteneur dans un registre de conteneur tel que Azure container registry, nous pouvons découper le travail en deux étapes.
- Etape 1: générer l'image.
- Etape 2: déployer l'image dans un Azure container registry.

# Azure pipelines – générer des images.

- La génération d'une image docker nécessite un fichier dockerfile dans notre dépôt.
- Dans notre pipelines, nous définissons :
  - Un trigger (branche main par exemple).
  - L'agent à utiliser pour la génération.
  - Un step pour la tâche de génération de l'image.
  - Nous pouvons utiliser la tâche Docker@2 qui permet de construire une image à partir d'un dockerfile.

```
trigger:
- main

pool:
vmImage: 'ubuntu-latest'

variables:
imageName: 'pipelines-javascript-docker'

steps:
- task: Docker@2
displayName: Build an image
inputs:
repository: $(imageName)
command: build
Dockerfile: app/Dockerfile
```

# Azure pipelines – déployer des images.

- Suite à la génération d'une image, nous pouvons créer une étape pour le push dans notre pipeline.
- Nous utilisons toujours la tâche Docker@2 pour envoyer notre image dans un Azure container registry.
- Pour l'envoie dans ACR, nous pouvons utiliser le mécanisme d'Azure de connexion de service pour l'authentification.

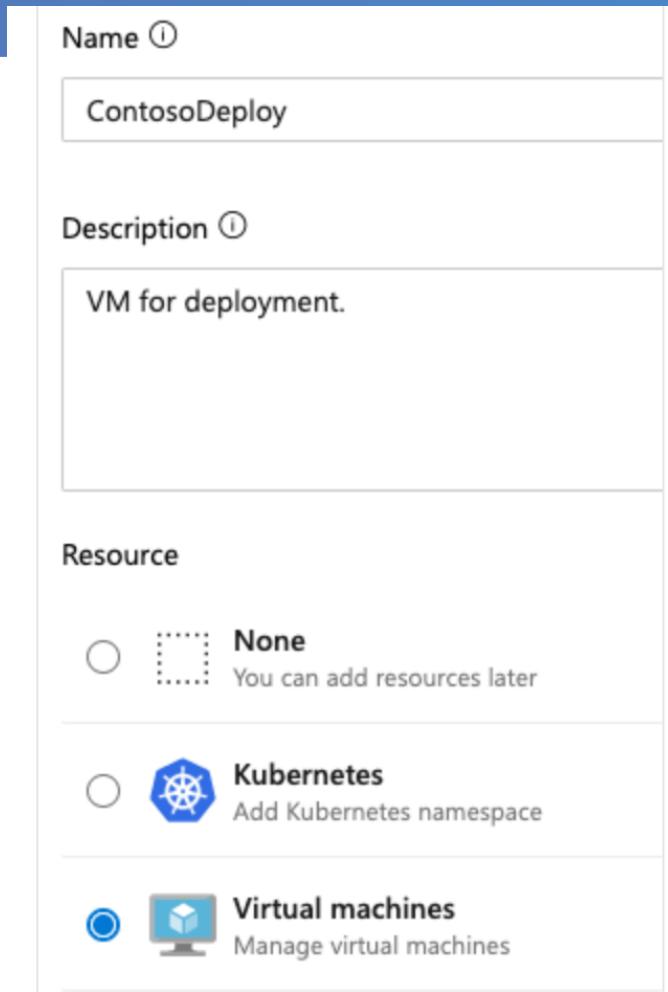
```
task: Docker@2
displayName: Push image
inputs:
containerRegistry: |
$(dockerHub)
repository: $(imageName)
command: push
tags: |
test1
test2
```

# Azure pipelines – déploiements vers d'autre ressources.

- Azure pipelines permet le déploiement d'application vers d'autre ressources tel que :
  - Une machine virtuelle
  - Un cluster kubernetes.
- Nous avons besoin que :
  - La ressource soit créer.
  - Le dépôt connecté.

# Azure pipelines – Création de la ressource.

- La première étape consiste à la création de la ressource (machine virtuelle par exemple) dans un nouvel environnement.
- Après la création de la ressource, un script est fourni pour être exécuter sur chaque machine virtuelle cible.
- Ce script permet d'inscrire nos machine virtuelle en tant que ressource.
- Après l'inscription la machine sera visible dans l'onglet ressources de l'environnement.



# Azure pipelines – Utilisation des ressources dans des pipelines.

- Après l'inscription de nos machines dans nos ressources, nous pouvons les cibler en référençant notre environnement.
- Les jobs de la pipelines seront exécuter par défaut sur la totalité des machines de l'environnement.
- Nous pouvons utiliser le mécanisme des tags pour spécifier uniquement une partie des machines virtuelles.

```
trigger:
- main

pool:
vmImage: ubuntu-latest

jobs:
- deployment: VMDeploy
displayName: Deploy to VM
environment:
name: ContosoDeploy
resourceType: VirtualMachine
strategy:
runOnce:
deploy:
steps:
- script: echo "Hello world"
```

# Azure pipelines – Plusieurs étapes.

- Une Azure pipeline peut être organiser en plusieurs étapes successives.
- Cette organisation est possible à l'aide des stages, chaque stage a un nom et plusieurs job.
- Chaque stage peut avoir un agent différent.
- Chaque stage peut être conditionné à l'exécution d'autre stage.
  - La dépendance entre stages peut se faire à l'aide du mot dependsOn, cette dépendance peut avoir, également, une condition.
  - Les conditions peuvent être réussite ou échec par exemple.
- Dans des pipelines en build classique (et non en YAML), nous pouvons spécifier également une stratégie de mise en attente.
- Azure pipelines permet également un contrôle manuel à l'aide d'un mécanisme d'approbation.

# Azure pipelines – Plusieurs étapes.

- Un exemple d’Azure pipelines en plusieurs :

```
stages:
- stage: A

stage B runs if A fails
- stage: B
 condition: failed()

stage C runs if B succeeds
- stage: C
 dependsOn:
- A
- B
 condition: succeeded('B')
```

# Azure pipelines – Tâches de test

- Azure pipelines permet d'exécuter différents types de tests à l'aide d'un ensemble de tâches disponibles.
- Quelque exemple de tâche de tests.
- App center test pour exécuter des tests sur un fichier binaire.
- L'exécution des tests fonctionnel à l'aide de la tâche « RunVisualStudioTestsusingTestAgent »
- L'exécution des tests unitaires à l'aide de la tâche « VSTest@2 »

# Azure pipelines – App Center test

```
App Center test
Test app packages with Visual Studio App Center
- task: AppCenterTest@1
 inputs:
 appFile:
 #artifactsDirectory: '$(Build.ArtifactStagingDirectory)/AppCenterTest'
 #prepareTests: true # Optional
 #frameworkOption: 'appium' # Required when prepareTests == True#
 Options: appium, espresso, calabash, uitest, xcuitest
 #appiumBuildDirectory: # Required when prepareTests == True &&
 Framework == Appium
 #espressoBuildDirectory: # Optional
 #espressoTestApkFile: # Optional
 #calabashProjectDirectory: # Required when prepareTests == True &&
 Framework == Calabash
 #calabashConfigFile: # Optional
 #calabashProfile: # Optional
 #calabashSkipConfigCheck: # Optional
 #uiTestBuildDirectory: # Required when prepareTests == True &&
 Framework == Uitest
 #uitestStorePath: # Optional
 #uiTestStorePassword: # Optional
 #uitestKeyAlias: # Optional
 #uiTestKeyPassword: # Optional
 #uiTestToolsDirectory: # Optional
 #signInfo: # Optional
 #xcUITestBuildDirectory: # Optional
 #xcUITestIpaFile: # Optional
 #prepareOptions: # Optional
 #runTests: true # Optional
```

# Azure pipelines – Test fonctionnel

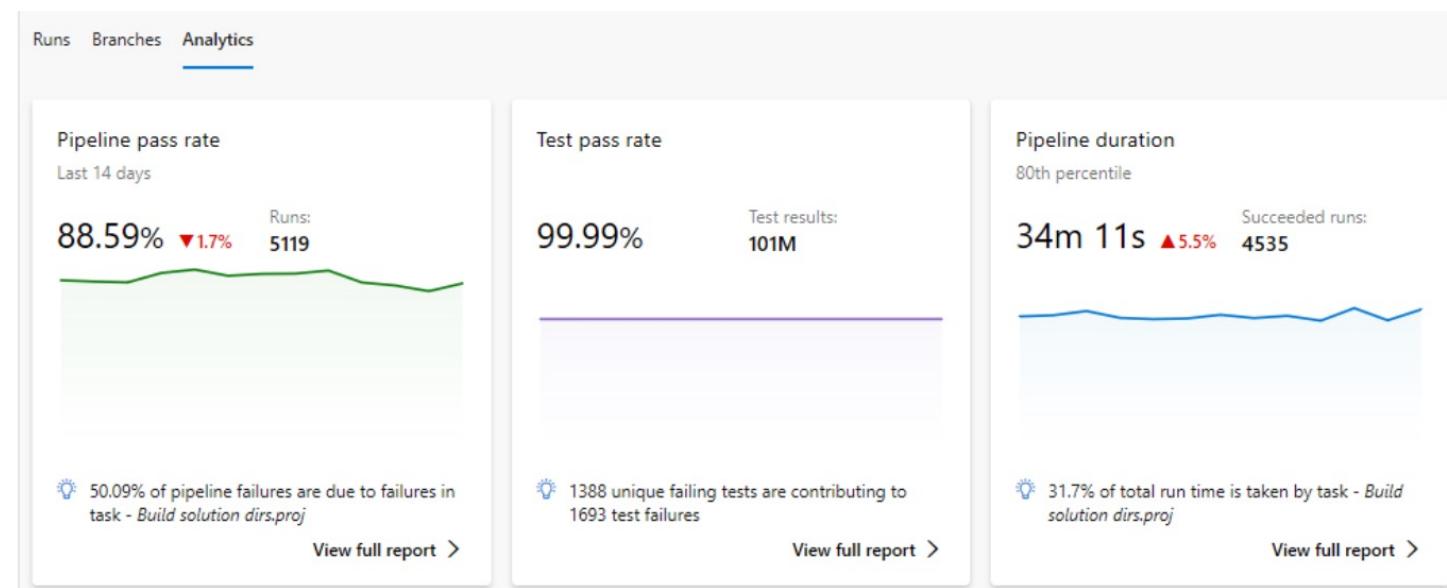
```
Run functional tests
- task: RunVisualStudioTestsusingTestAgent@1
 inputs:
 testMachineGroup:
 dropLocation:
 #testSelection: 'testAssembly' # Options: testAssembly, testPlan
 #testPlan: # Required when testSelection == TestPlan
 #testSuite: # Required when testSelection == TestPlan
 #testConfiguration: # Required when testSelection == TestPlan
 #sourcefilters: '***test*.dll' # Required when testSelection ==
 TestAssembly
 #testFilterCriteria: # Optional
 #runSettingsFile: # Optional
 #overrideRunParams: # Optional
 #codeCoverageEnabled: false # Optional
 #customSlicingEnabled: false # Optional
 #testRunTitle: # Optional
 #platform: # Optional
 #configuration: # Optional
 #testConfigurations: # Optional
 #autMachineGroup: # Optional
```

# Azure pipelines – Test unitaire

```
Visual Studio Test
Run unit and functional tests (Selenium, Appium, Coded UI test,
etc.) using the Visual Studio Test (VsTest) runner.
- task: VSTest@2
inputs:
#testSelector: 'testAssemblies' # Options: testAssemblies, testPlan,
testRun
#testAssemblyVer2: | # Required when testSelector == TestAssemblies
***test*.dll
!***TestAdapter.dll
!**\obj**
#testPlan: # Required when testSelector == TestPlan
#testSuite: # Required when testSelector == TestPlan
#testConfiguration: # Required when testSelector == TestPlan
#tcmTestRun: '$(test.RunId)' # Optional
#searchFolder: '$(System.DefaultWorkingDirectory)'
#testFiltercriteria: # Optional
#runOnlyImpactedTests: False # Optional
#runAllTestsAfterXBuilds: '50' # Optional
#uiTests: false # Optional
```

# Azure pipelines – Rapport

- Azure pipelines permet d'afficher un résumé des taux de réussite et la durée d'exécution dans l'onglet analytique.
- Azure pipelines permet de réaliser un export des rapports ainsi qu'un filtrage suivant plusieurs critères.



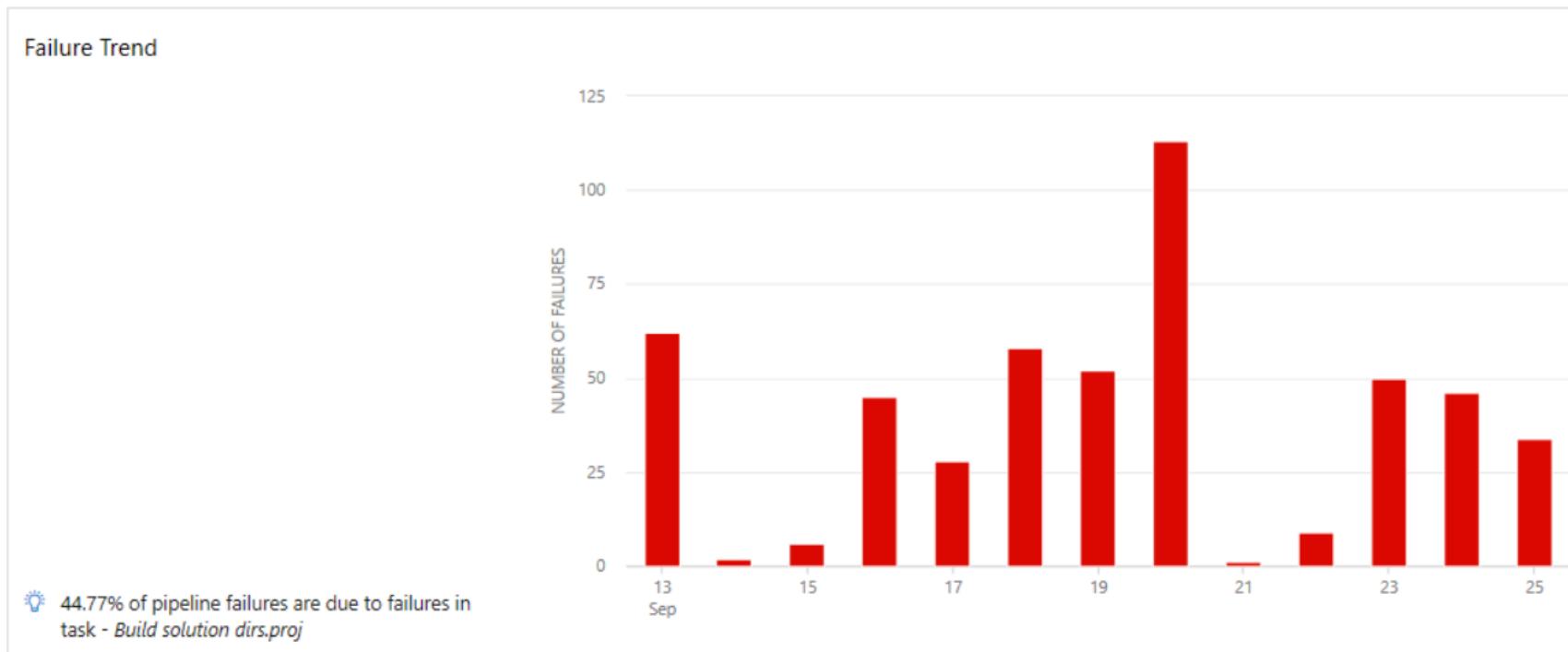
# Azure pipelines – Rapport réussite, échec

- Azure pipelines permet d'afficher également un taux de réussite et d'échec, ainsi qu'une tendance au fil du temps.
- Azure pipelines permet également de faire un focus sur l'échec d'une tâche bien spécifique pour apporter des corrections spécifiques.



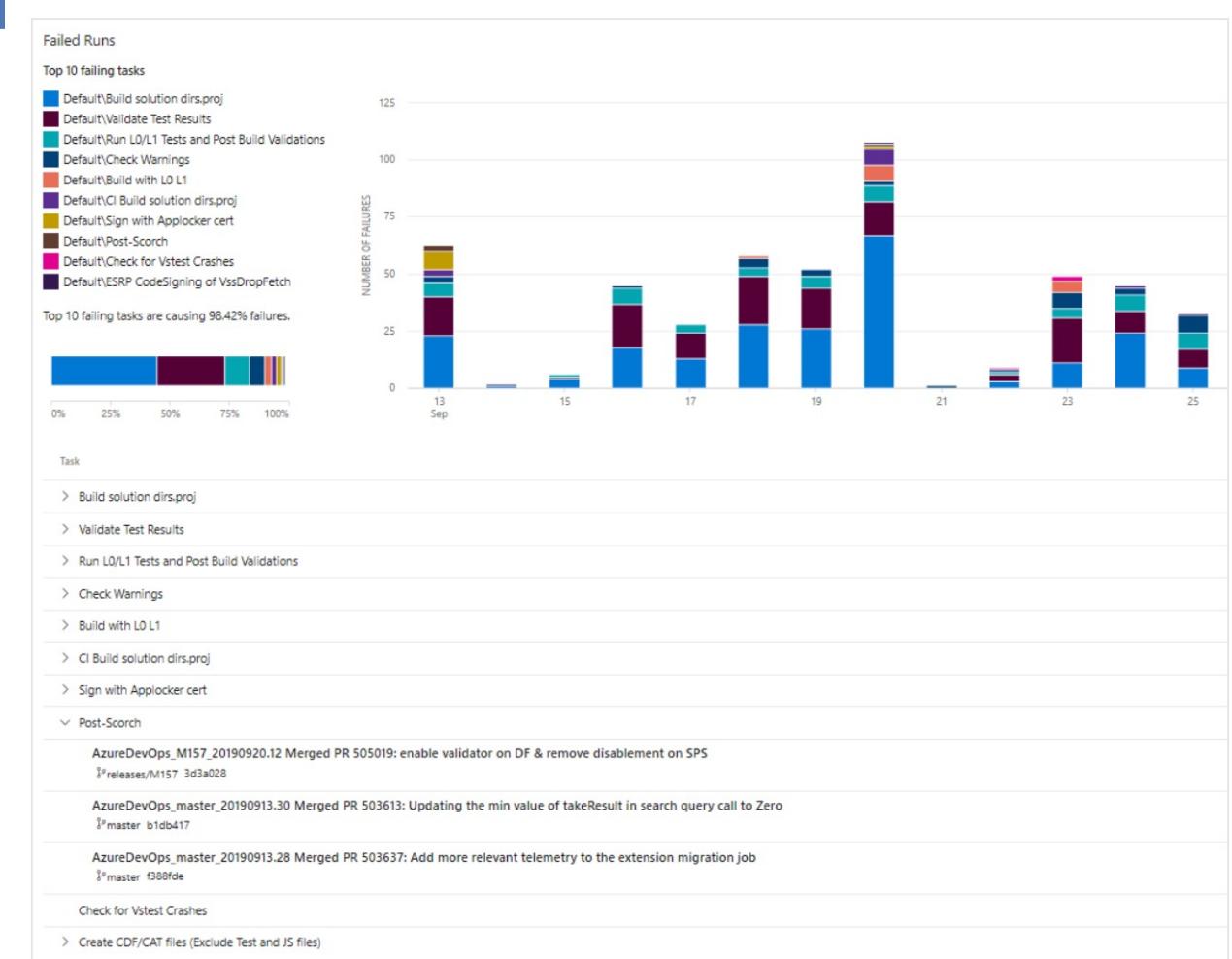
# Azure pipelines – Rapport réussite, échec

- Exemple tendance d'échec.



# Azure pipelines – Rapport réussite, échec

- Focus sur les tâches en échec.

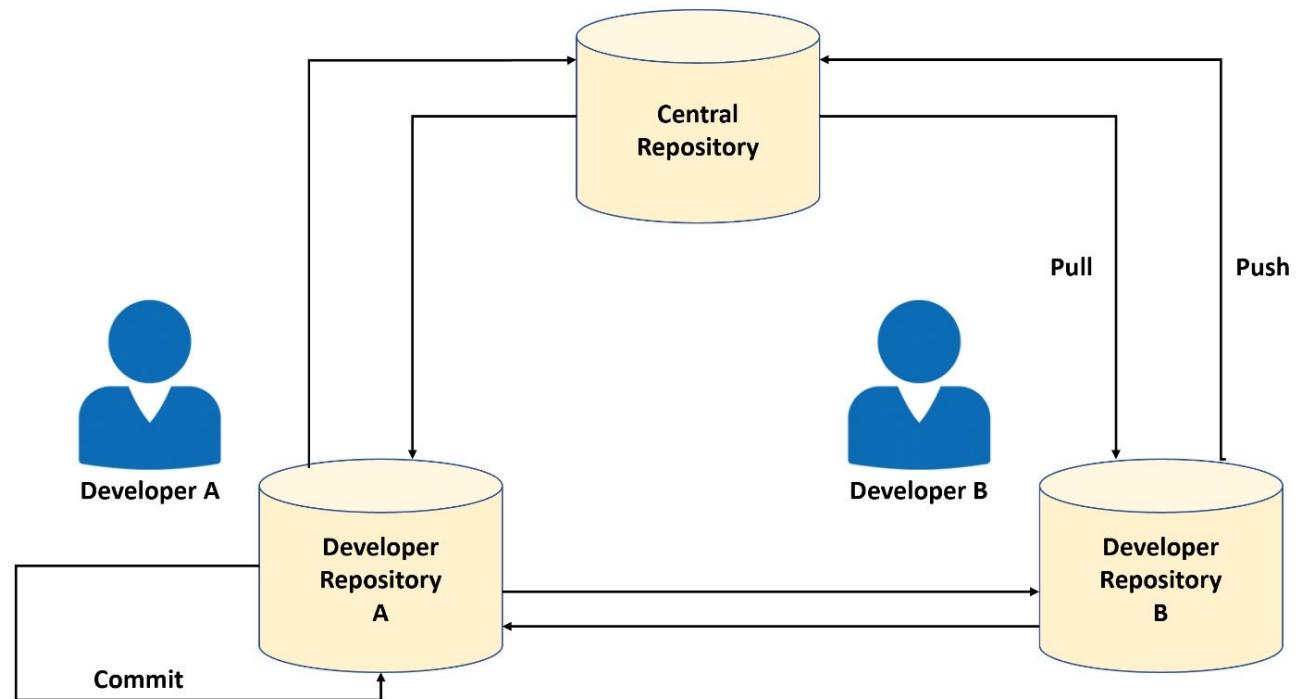


# Source Control Management Azure DevOps - Repos

- La gestion du contrôle des sources (SCM) est un élément essentiel de toute entreprise qui développe des logiciels de manière professionnelle, mais aussi de tout développeur qui souhaite disposer d'un moyen sûr de stocker et de gérer son code.
- Lorsque nous travaillons en équipe, il est absolument nécessaire d'avoir un référentiel central sécurisé où tout votre code est stocké. Il est également nécessaire d'avoir un système qui garantit que le code est partagé en toute sécurité entre les développeurs et que chaque modification est inspectée et fusionnée sans générer de conflits.

# Azure Repos – SCM

- Le contrôle de code source (ou contrôle de version) est une pratique logicielle utilisée pour suivre et gérer les modifications apportées au code source.
- Il s'agit d'une pratique extrêmement importante car elle permet de maintenir une seule source de code entre différents développeurs et aide à collaborer sur un seul projet logiciel (où différents développeurs travaillent sur la même base de code).



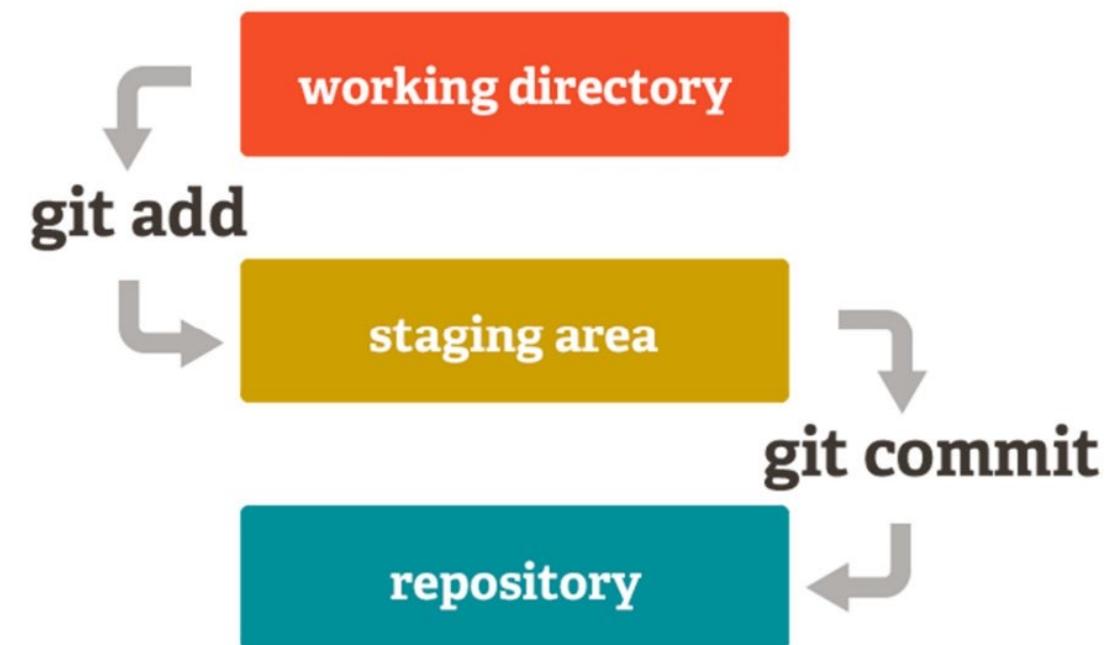
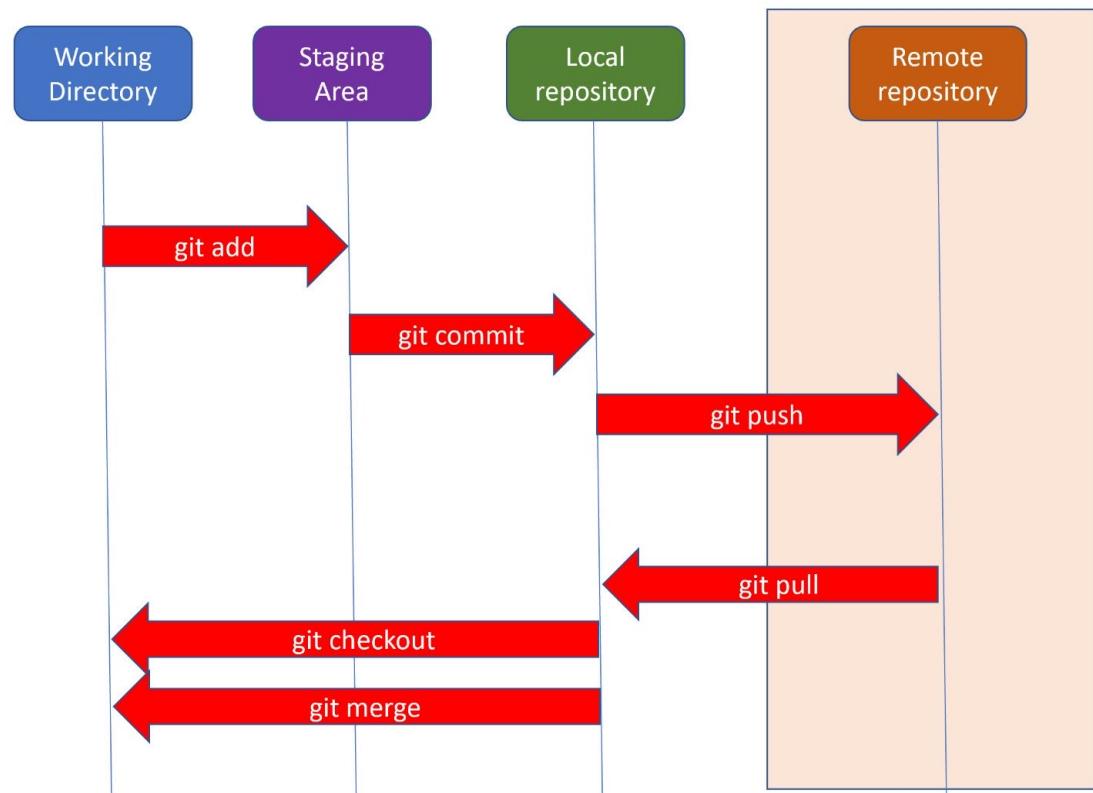
# Azure Repos - Git

- Git est absolument l'un des systèmes SCM les plus populaires sur le marché.
- Git a été créé en 2005 par Linus Torvalds pour aider au développement du noyau Linux. Git est gratuit, open source et entièrement basé sur des fichiers, donc aucun logiciel supplémentaire n'est requis pour gérer SCM, à l'exception du moteur Git lui-même.
- Git a un workflow qui peut être résumé comme suit :
  - Vous créez un référentiel pour votre projet sur votre système d'hébergement Git.
  - Vous copiez (ou clonez) le référentiel sur votre ordinateur de développement local.
  - Vous créez un nouveau fichier dans votre référentiel local, puis vous enregistrez les modifications localement (étape et validation).
  - Vous poussez les modifications vers le référentiel distant (push).
  - Vous récupérez les modifications du référentiel distant vers le référentiel local (pour aligner votre code avec le référentiel distant si d'autres développeurs ont apporté des modifications).
  - Vous fusionnez les modifications avec votre référentiel local.

# Azure Repos – Rappel Git

- Les snapshots sont la façon dont Git garde une trace de l'historique de votre code.
- Un snapshot enregistre essentiellement à quoi ressemblent tous vos fichiers à un moment donné.
- La validation est l'acte de créer un snapshot. Dans un projet, vous créez différents commits. Un commit contient trois ensembles d'informations :
  - -- Détails sur la façon dont les fichiers ont changé depuis la version précédente
  - -- Une référence au commit parent (commit survenu précédemment)
  - -- Un nom de code de hachage
- Les repositories sont des collections de tous les fichiers nécessaires et de leur historique. Un repos peut se trouver sur une machine locale ou sur un serveur distant.
- Le clonage consiste à copier un référentiel à partir d'un serveur distant.
- Le pulling est le processus de téléchargement de validations qui n'existent pas sur votre machine à partir d'un référentiel distant.
- Le push est le processus d'ajout de vos modifications locales à un référentiel distant.
- Les branches sont des "versions" de votre base de code. Tous les commits dans Git vivent dans une branche et vous pouvez avoir différentes branches. La branche principale d'un projet est connue sous le nom de master ou main

# Azure Repos – Rappel Git

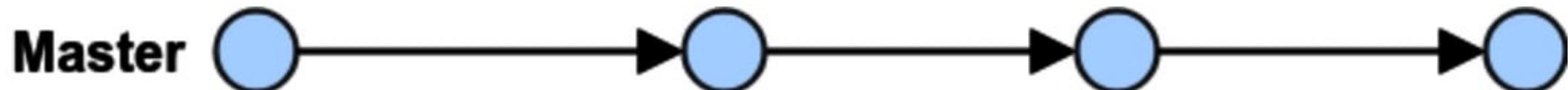


# Azure Repos – Rappel commande base git

- **git clone <https://github.com/user/yourRemoteRepo.git>.**
- **git add . git commit -m "my commit message".**
- **git pull**
- **git push**
- **git checkout -b 'branch1'**
- **git add .**
- **git commit –m 'update from branch1'**
- **git checkout master**
- **git merge branch1**
- **git push**

# Azure Repos – Branches

- Une branche est une version de votre code stockée dans un système SCM. Lorsque vous utilisez SCM avec Git, le choix de la meilleure stratégie de branche à adopter pour votre équipe est crucial car cela vous aide à disposer d'une base de code fiable et d'une livraison rapide.
- Avec SCM, si vous n'utilisez pas de branche, vous avez toujours une seule version de votre code (branche master ou main)

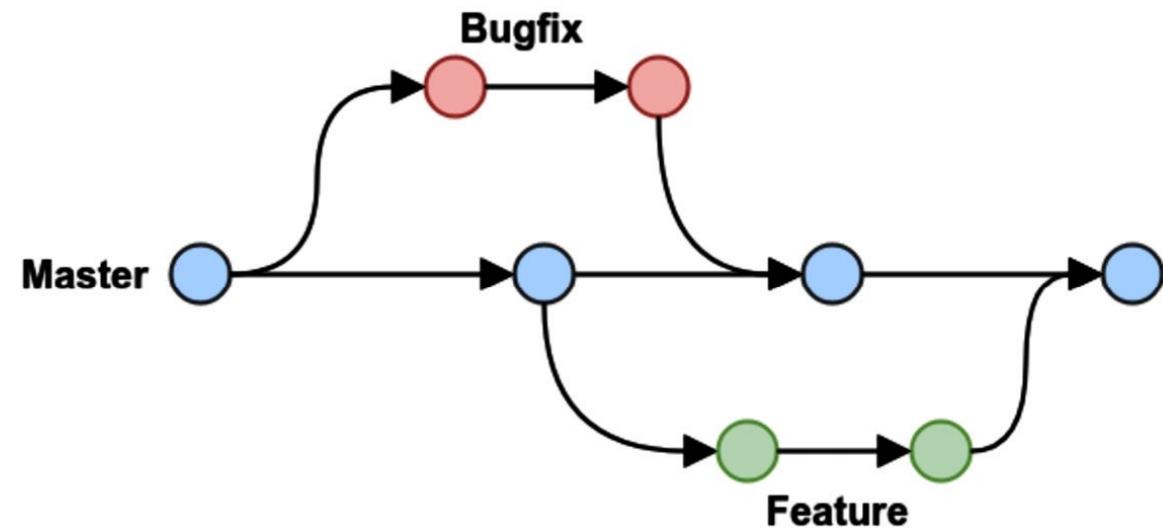


# Azure Repos - Branches

- Il existe différents workflows de branchement (stratégies) que vous pouvez adopter pour votre équipe.
- Avec Git, il existe trois principales stratégies de branchement que vous pouvez adopter:
  - GitHub Flow
  - GitLab Flow
  - Git Flow

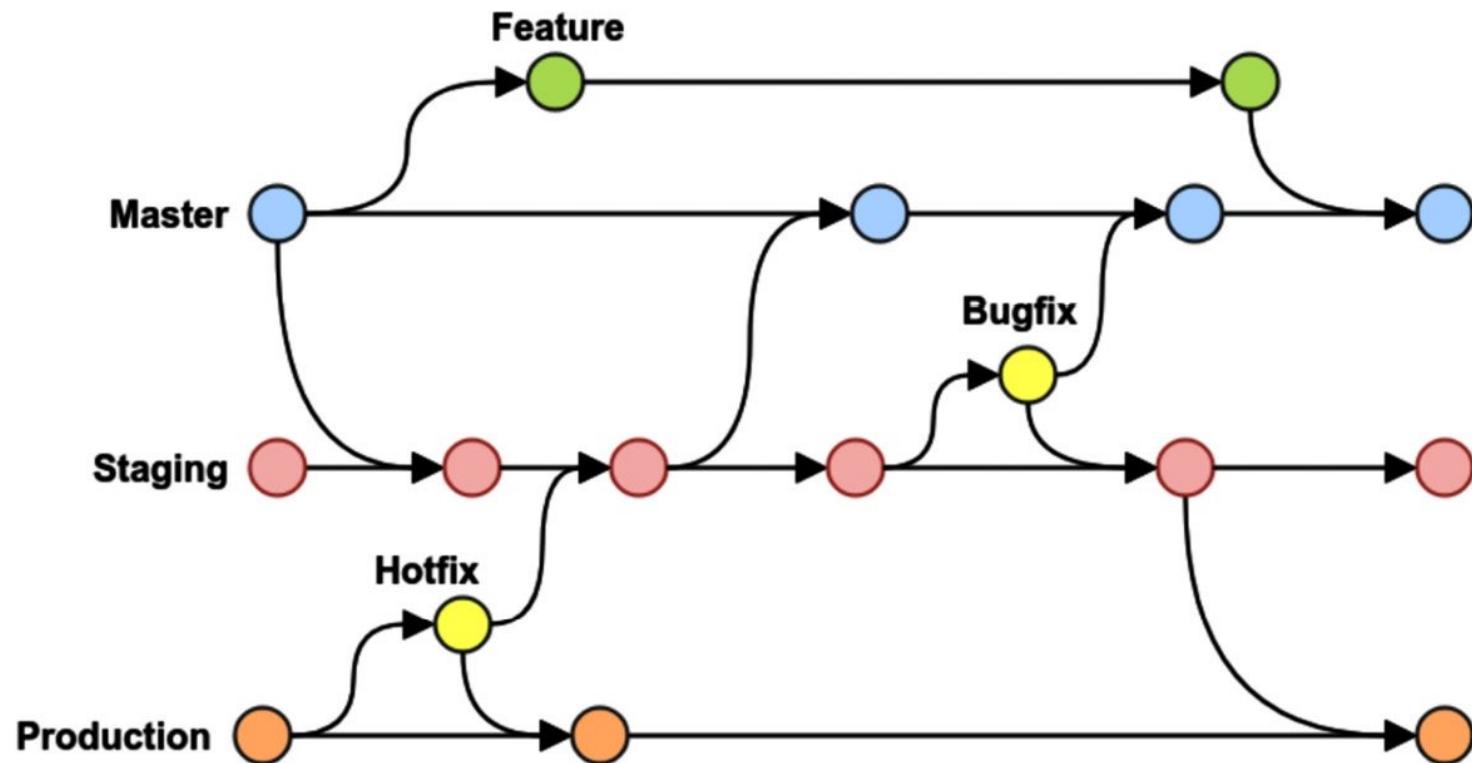
# Azure Repos – Branches – GitHub Flow

- GitHub Flow est l'une des stratégies de création de branches les plus utilisées et est assez simple à adopter.
- Selon ce workflow, vous partez d'une branche main (qui contient toujours le code déployable).
- Lorsque vous commencez à développer une nouvelle fonctionnalité, vous créez une nouvelle branche et vous vous engagez régulièrement dans cette nouvelle branche. Lorsque le travail de développement est terminé, vous créez une pull request pour fusionner la branche secondaire avec la branche main



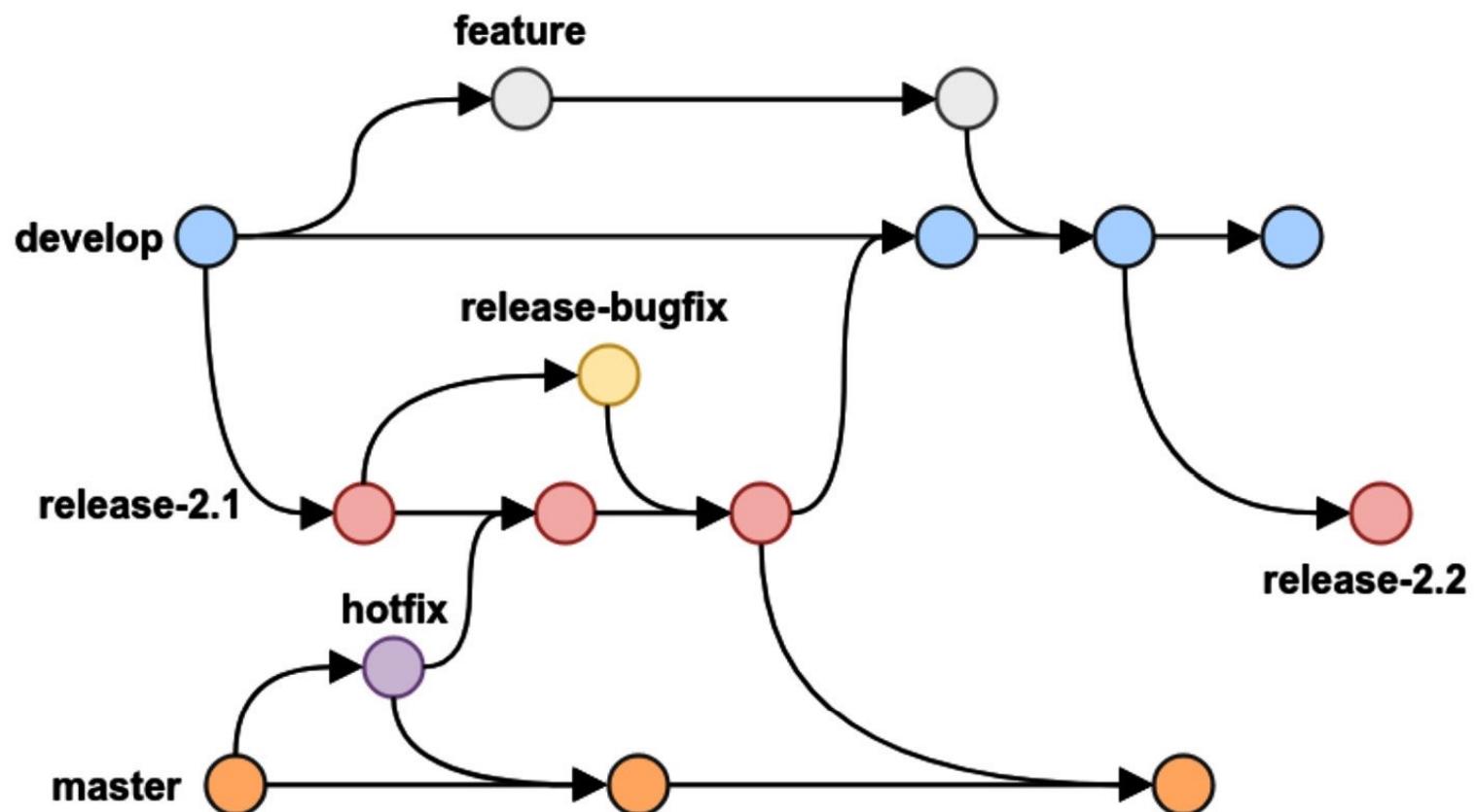
# Azure Repos – Branches – GitLab Flow

- GitLab Flow est une autre stratégie de branchement populaire largement utilisée, en particulier lorsque vous devez prendre en charge plusieurs environnements (tels que la production, la mise en scène, le développement, etc.) dans votre processus SCM.



# Azure Repos – Branches – Git Flow

- Git Flow est un workflow utilisé lorsque vous avez un cycle de publication planifié.



# Azure Repos

- Azure DevOps prend en charge deux types de gestion de contrôle source :
  - Git : il s'agit d'un système de contrôle de version distribué et est le fournisseur de contrôle de version par défaut dans Azure DevOps lorsque vous créez un nouveau projet.
  - Team Foundation Version Control (TFVC) : il s'agit d'un système de contrôle de version centralisé où les développeurs n'ont qu'une seule version d'un fichier localement, les données sont stockées sur un serveur et des branches sont créées sur le serveur.

# Azure Repos – Crédit d'un repository

- La première étape lorsque vous travaillez avec Azure DevOps consiste à créer un nouveau projet au sein de votre organisation. Lorsque vous créez un nouveau projet avec Azure DevOps, vous êtes invité à choisir le système de contrôle de version que vous souhaitez utiliser.
- En cliquant sur le bouton OK, le nouveau projet sera créé dans votre organisation Azure DevOps.
- Une fois le projet provisionné, vous pouvez gérer vos référentiels en vous rendant dans le hub Repos sur la barre de gauche dans Azure DevOps (voir la capture d'écran suivante). C'est là que vos fichiers seront stockés et où vous pourrez commencer à créer des référentiels et à gérer des branches, des demandes d'extraction, etc.

The screenshot shows the 'Create new project' dialog box and the 'Repos' hub sidebar in Azure DevOps.

**Create new project** Dialog:

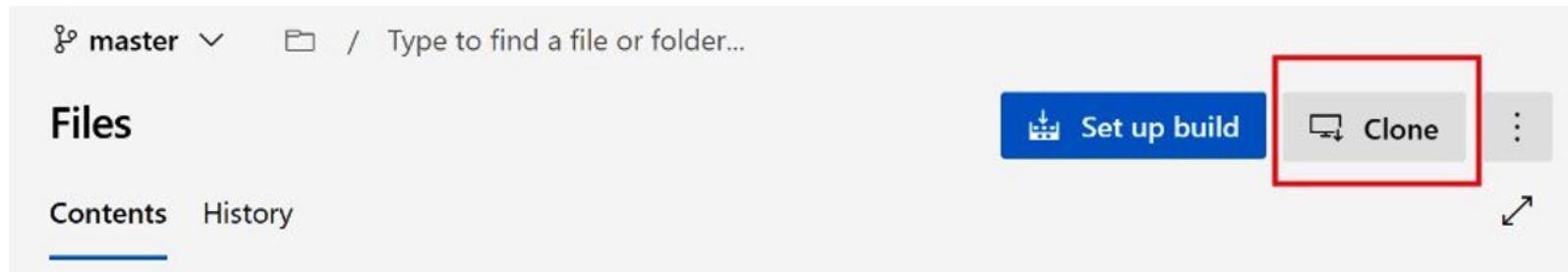
- Project name \***: An empty input field.
- Description**: An empty input field.
- Visibility**:
  - Public**: A greyed-out option with a note: "Anyone on the internet can view the project. Certain features like TFVC are not supported."
  - Private**: The selected option, highlighted with a blue border. Note: "Only people you give access to will be able to view this project."
- Advanced** section:
  - Version control**: A dropdown set to **Git**, with **Git** highlighted in blue and **Team Foundation Version Control** listed below it.
  - Work item process**: A dropdown set to **Agile**.

**Repos** Hub Sidebar (highlighted with a red box):

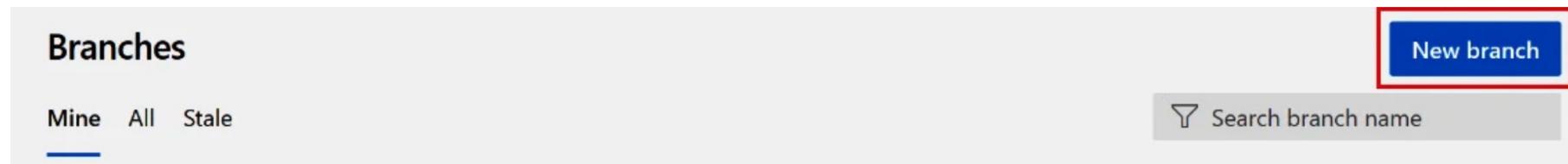
- Overview
- Boards
- Repos** (highlighted)
- Files
- Commits
- Pushes
- Branches
- Tags
- Pull requests
- Pipelines
- Test Plans
- Artifacts

# Azure Repos

- Pour clone d'un projet, nous pouvons récupérer l'url du repository à partir de l'interface du repos.



- Nous pouvons ajouter des branches également à partir de l'interface du repos



# Azure Repos – Protection des branches

- Lorsque vous travaillez avec différents développeurs et lorsque vous utilisez des branches, il est extrêmement important de protéger les branches critiques que vous avez dans votre référentiel (comme la branche principale) avec des règles qui peuvent garantir que la branche sera toujours dans un état sain.
- Pour cette portée, Azure DevOps vous permet de spécifier un ensemble de stratégies pour vos branches critiques.
- Les stratégies de branche dans Azure DevOps vous permettent d'effectuer les opérations suivantes :
  - Limiter les contributeurs à une branche spécifique
  - Spécifiez qui peut créer des branches
  - Spécifier un ensemble de conventions de dénomination pour les branches
  - Inclure automatiquement les réviseurs de code pour chaque changement de code dans la branche
  - Appliquer l'utilisation des pull requests.
  - Démarrer une pipeline de build avant de valider le code dans la branche

# Azure Repos – Protection des branches

The screenshot shows the Azure Repos Branches page for the repository "MyHealthClinic". The "master" branch is selected and highlighted with a red box. A context menu is open on the right side, listing various actions: "New branch", "New pull request", "Delete branch", "View files", "View history", "Compare branches", "Set as compare branch", "Lock", "Branch policies" (which is highlighted with a red box), and "Branch security".

| Branch          | Commit   | Author   | Authored ... | Behind   Ahead |
|-----------------|----------|----------|--------------|----------------|
| AddingContactUs | 924ac579 | Chris... | 4 mag 2...   | 12   2         |
| CopyrightUpdate | b6f5e2b5 | Chris... | 4 mag 2...   | 12   1         |
| development     | 69be6c36 | Stef...  | 2h ago       | 0   1          |
| master          | 27439428 | Stef...  | Yesterday    |                |

# Azure Repos – Protection des branches



## Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers

2

- Allow requestors to approve their own changes
- Allow completion even if some reviewers vote to wait or reject
- Reset code reviewer votes when there are new changes

# Azure Repos – Protection des branches



## **Check for linked work items**

Encourage traceability by checking for linked work items on pull requests.

### Policy requirement



Required

Block pull requests from being completed unless they have at least one linked work item.



Optional

Warn if there are no linked work items, but allow pull requests to be completed.

# Azure Repos – Protection des branches



## **Check for comment resolution**

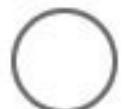
Check to see that all comments have been resolved on pull requests.

### Policy requirement



Required

Block pull requests from being completed while any comments are active.



Optional

Warn if any comments are active, but allow pull requests to be completed.

# Azure Repos – Protection des branches Limitation des merges

- **Basic merge** : cette option fusionne l'historique des commits de la branche source et crée un commit de fusion dans la branche cible. L'historique non linéaire complet des commits qui se produisent pendant le développement est préservé.
- **Squash merge** : cela crée un seul commit dans la branche cible en compressant les commits de la branche source (historique linéaire).
- **Rebase and fast-forward** : Un rebase permet l'intégration d'une branche pull request dans la branche master. Chaque commit sur la pull request est fusionné individuellement dans la branche cible (historique linéaire).
- **Rebase with merge commit** : cela crée un historique semi-linéaire en remplaçant les commits de la branche source dans la branche cible, puis en créant un merge commit.

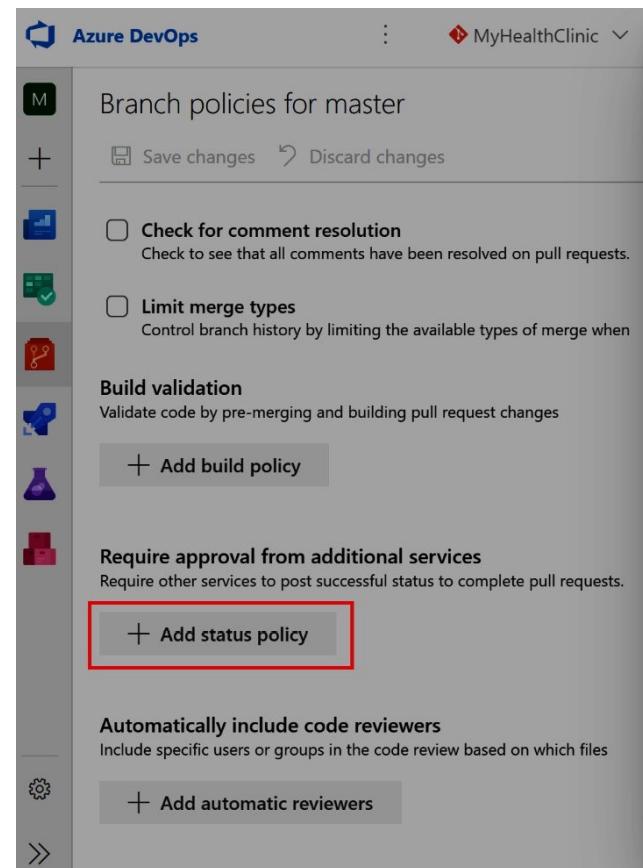
# Azure Repos –Validation des builds

- Cette section vous permet de spécifier un ensemble de règles pour la construction de votre code avant que pullRequests ne puisse être complétée (utile pour détecter les problèmes plus tôt).
- Vous pouvez spécifier quelle définition de pipeline de construction vous souhaitez appliquer et si elle doit être déclenchée automatiquement lorsque la branche est mise à jour ou manuellement

The screenshot shows the 'Branch policies for master' page in Azure DevOps. On the left, there's a sidebar with icons for Merge, Pull Request, Work items, and Pipelines. The main area displays 'Protect this branch' settings, including options for review numbers, linked work items, comment resolution, and merge types. Below this is the 'Build validation' section, which contains a button labeled '+ Add build policy'. This button is highlighted with a red box. To the right, a modal window titled 'Add build policy' is open, showing configuration fields for a build pipeline, trigger (Automatic), policy requirement (Required), and build expiration (After 12 hours). The 'Save' and 'Cancel' buttons are at the bottom right of the modal.

# Azure Repos –Validation des builds

- Cette option vous permet de connecter des services externes (via les API de requête pull Azure DevOps) afin de participer au flux de pullRequests



The screenshot shows the 'Branch policies for master' page in Azure DevOps. On the left, there's a sidebar with icons for Merge, Add, Build, Approve, and Review. The main area lists several policies: 'Check for comment resolution', 'Limit merge types', 'Build validation' (with a '+ Add build policy' button), 'Require approval from additional services' (with a '+ Add status policy' button highlighted by a red box), and 'Automatically include code reviewers' (with a '+ Add automatic reviewers' button). To the right, a modal window titled 'Add status policy' is open, containing fields for 'Status to check \*' (a dropdown menu), 'Policy requirement' (radio buttons for 'Required' or 'Optional'), 'Authorized organization' (radio buttons for 'Any identity can post this status' or 'Only the following identity can post this status' with a 'Search users' input field), and 'Reset conditions' (checkbox for 'Reset status whenever there are new changes').

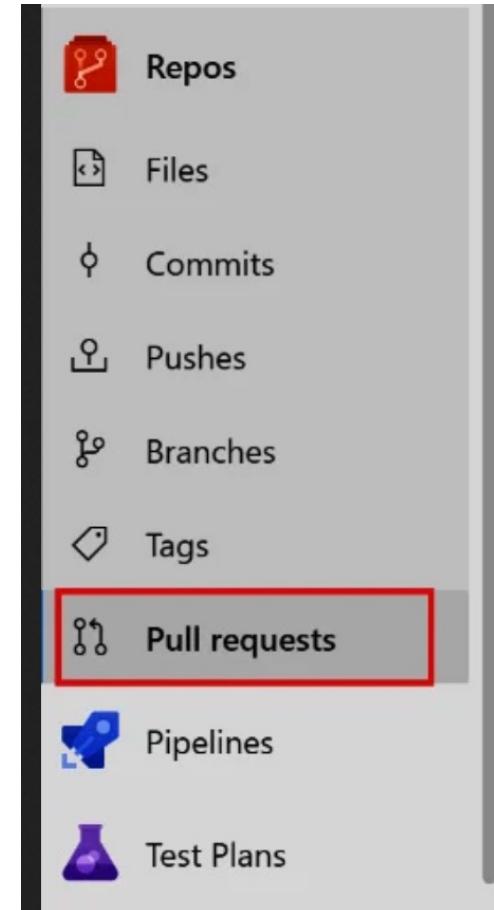
# Azure Repos – Code Review

- Cette stratégie vous permet d'inclure des utilisateurs ou des groupes spécifiques dans le processus du code review

The screenshot shows the Azure DevOps interface for managing branch policies. On the left, under 'Branch policies for master', there are sections for 'Check for comment resolution', 'Limit merge types', 'Build validation', and 'Require approval from additional services'. Below these is a section titled 'Automatically include code reviewers' with a note: 'Include specific users or groups in the code review based on which files'. A red box highlights the 'Add automatic reviewers' button. To the right, a modal dialog titled 'Automatically include reviewers' is open. It contains a search bar for 'Include the following reviewer(s) \*' and a dropdown for 'Policy requirement' where 'Required' is selected. It also includes a 'For pull requests affecting these folders' field set to 'No filter set', a 'Completion options' section with a checked checkbox for 'Allow requestors to approve their own changes', and an 'Activity feed message' field. At the bottom right of the dialog are 'Save' and 'Cancel' buttons.

# Azure Repos – Utilisation des PullRequests

- Les pullRequests permettent d'informer les membres de votre équipe qu'une nouvelle implémentation est terminée et doit être fusionnée avec une branche spécifiée.
- En utilisant les pullRequests, les membres de votre équipe peuvent réviser votre code (en parcourant les fichiers et voir les modifications introduites par un commit particulier), fournir des commentaires de révision sur des problèmes mineurs et approuver ou rejeter ces modifications. Il s'agit de la pratique recommandée à utiliser lors de l'utilisation de la gestion du contrôle de code source avec Azure DevOps.
- Vous pouvez afficher les pullRequests entrantes pour un repos spécifique sur Azure DevOps en sélectionnant le menu pullRequests dans le hub Repos.



# Azure Repos – Cr ation des PullRequests

- Un PullRequest peut  tre cr  e de diff  entes mani res :
  - Manuellement   partir de la page des pull requests Azure DevOps
  - A partir d'un work item li    une branche (l'onglet D veloppement)
  - Lors d'une mise   jour   une branche de fonctionnalit 
  -   partir de l'interface de ligne de commande Azure DevOps Services

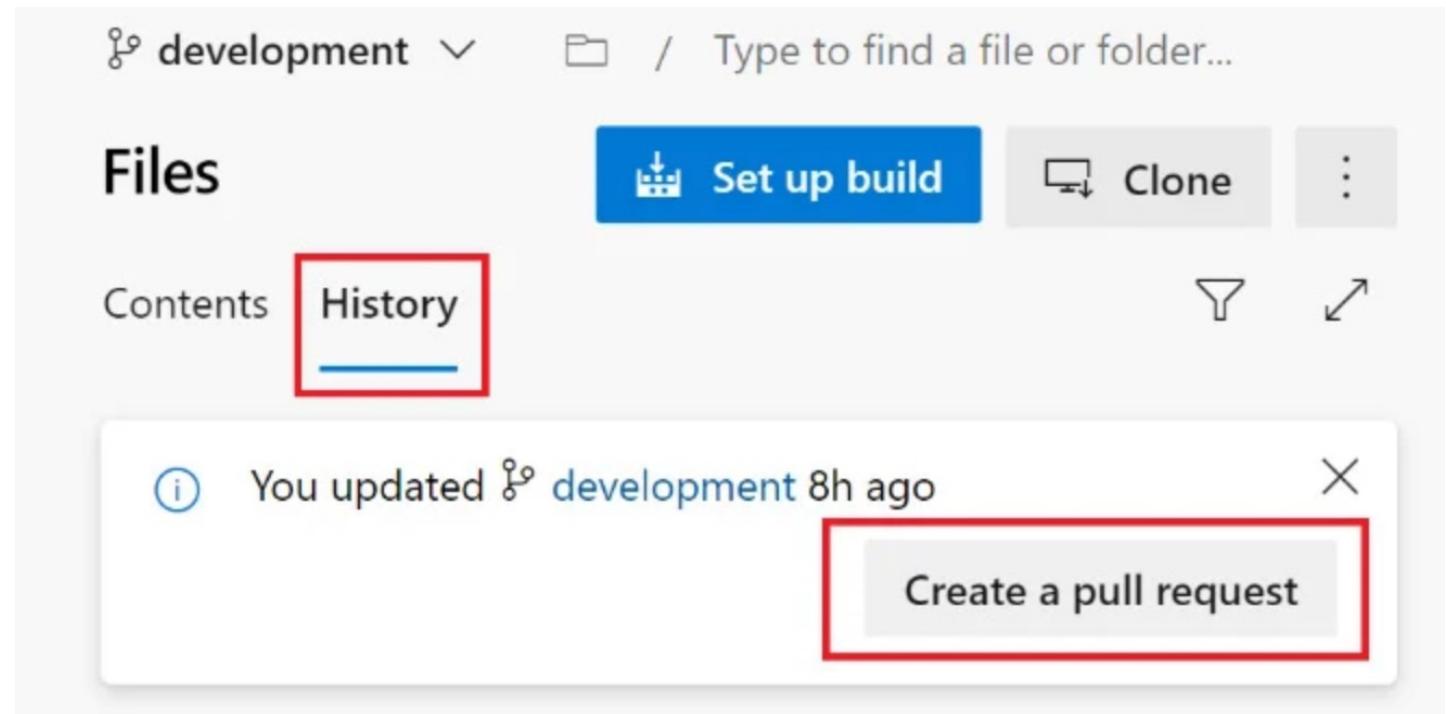
# Azure Repos – Cr ation des PullRequests – Work item

- Dans la vue Backlogs des work items de votre  quipe,
- Vous pouvez s lectionner un work item avec une branche li e.
- Acc der   la zone D veloppement de l'item.
- Cr er une pullRequest.

The screenshot shows the 'Details' view of a work item in Azure DevOps. The top navigation bar includes 'State' (To Do), 'Area' (MyHealthClinic), 'Reason' (New task), 'Iteration' (MyHealthClinic), and a 'Details' button which is highlighted with a red box. Below the navigation, there are sections for 'Description' (empty), 'Details' (Priority: 2, Remaining Work: 0, Activity: 0), and 'Deployment' (instructions to track releases). The main content area is titled 'Development' and contains a 'Create a pull request' button, which is also highlighted with a red box. Other buttons in this section include 'Add link' and 'development'. Below the development section is a 'Related Work' section.

# Azure Repos – Cr ation des PullRequests – Apr s un push

- Lorsque vous validez du code dans une branche de d veloppement dans Azure DevOps, vous  tes automatiquement invit    cr er une pull request



# Azure Repos – Gestion des pullRequests

- Dans Azure DevOps, la fenêtre pull Request permet de renseigner les détails d'activité du pull request.
- Une fois la demande créée, vous pouvez compléter la demande en cliquant sur le bouton Terminer dans le coin supérieur droit de la fenêtre de la demande de tirage (vous pouvez le faire après la phase d'approbation facultative et après avoir passé les règles de la branche)

New Pull Request

From development into master ↗

Title \*

New features deployed on the development branch

Add label

Description

New features deployed on the development branch

Markdown supported.

>New features deployed on the development branch

Reviewers

Search users and groups to add as reviewers

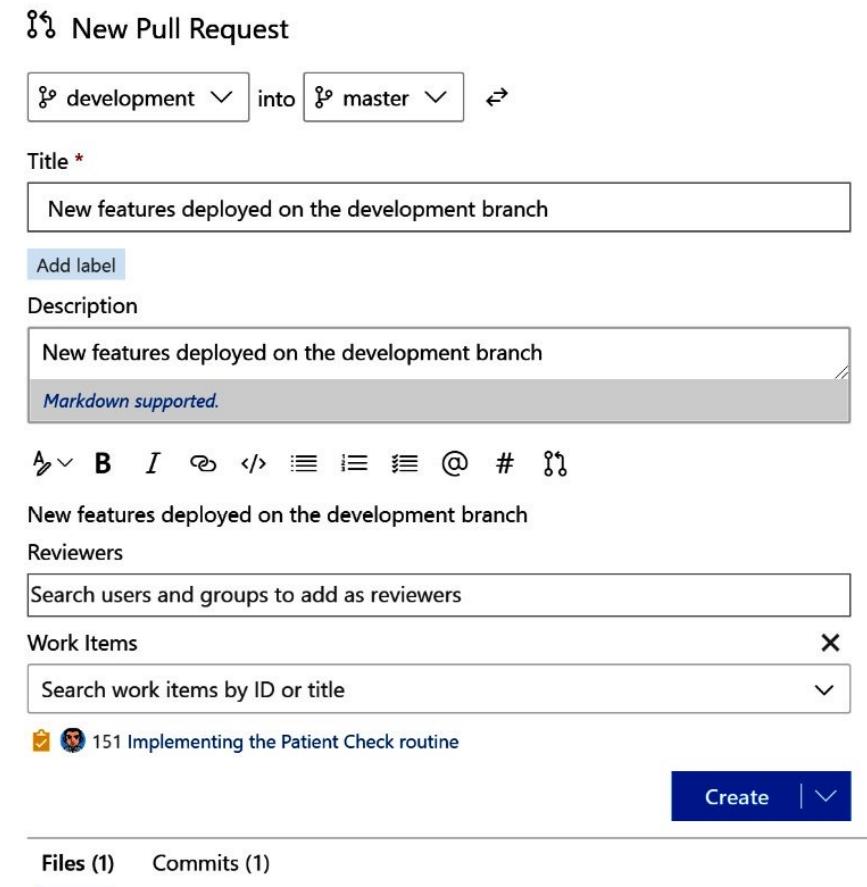
Work Items

Search work items by ID or title

151 Implementing the Patient Check routine

Create | ↗

Files (1) Commits (1)



# Azure Repos - Tags

- Les tags Git sont des références qui pointent vers des points spécifiques de l'historique de Git.
- Les tags sont utilisées dans Azure DevOps pour marquer une version (ou branche) particulière avec un identifiant qui sera partagé en interne dans votre équipe pour identifier, par exemple, la « version » de votre base de code.
- Pour utiliser des tags pour vos branches, dans le hub Repos d'Azure DevOps, accédez au menu tags.

# Azure Artifacts

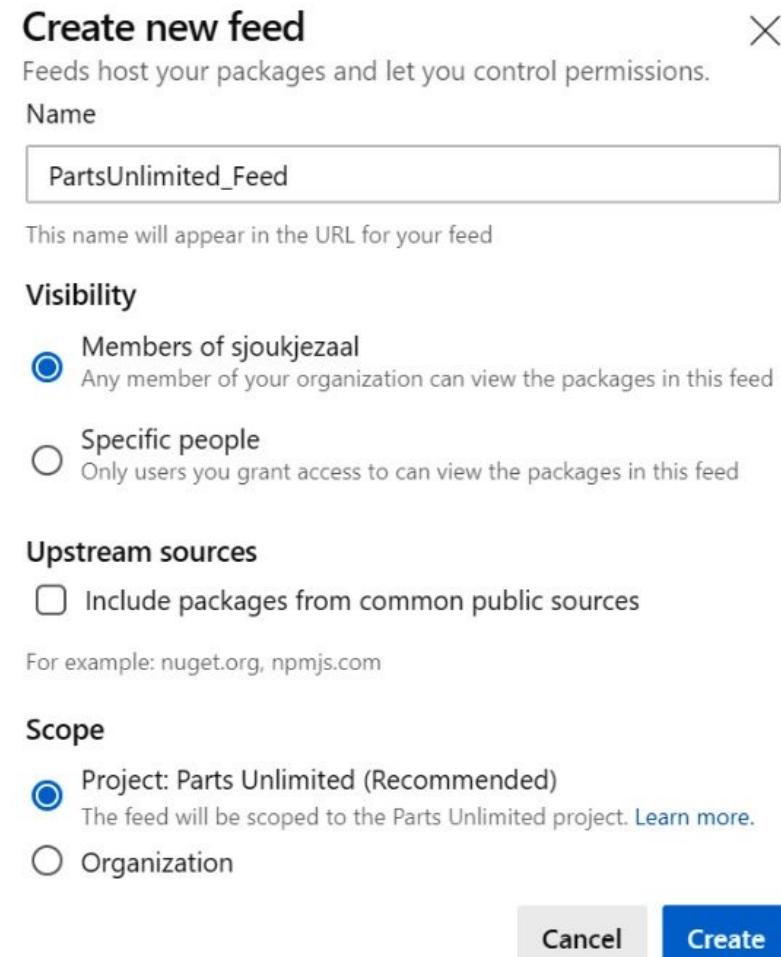
- Il est probable que chaque développeur ait utilisé un package tiers ou open source dans son code pour ajouter des fonctionnalités supplémentaires et accélérer le processus de développement de son application.
- L'utilisation de composants populaires pré-construits qui ont été utilisés et testés par la communauté vous aidera à faire avancer les choses plus facilement.
- Les fonctionnalités, les scripts et le code créés par différentes équipes de votre organisation sont souvent réutilisés par d'autres équipes et dans différents projets de développement logiciel. Ces différents artefacts peuvent être déplacés dans une bibliothèque ou un package afin que d'autres puissent en bénéficier.
- Il existe différentes manières de créer et d'héberger ces packages. Par exemple, vous pouvez utiliser NuGet pour héberger et gérer des packages pour la plate-forme de développement Microsoft ou npm pour les packages JavaScript, Maven pour Java, etc.
- Azure Artifacts propose des fonctionnalités vous permettant de partager et de réutiliser facilement des packages.
- Dans Azure Artifacts, les packages sont stockés dans des flux.
- Un flux est un conteneur qui vous permet de regrouper des packages et de contrôler qui y a accès.

# Azure Artifacts

- Vous pouvez stocker des packages dans des flux créés par vous-même ou par d'autres équipes, mais il dispose également d'un support intégré pour les sources en amont. Avec les sources en amont, vous pouvez créer un flux unique pour stocker à la fois les packages produits par votre organisation et les packages consommés à partir de flux distants, tels que NuGet, npm, Maven, Chocolatey, RubyGems, etc.
- Il est fortement recommandé d'utiliser Azure Artifacts comme source principale pour la publication de packages internes et de flux distants.
- En effet, cela vous permet de garder une vue d'ensemble complète de tous les packages utilisés par l'organisation et les différentes équipes.
- Le flux connaît la provenance de tous les packages enregistrés à l'aide de ressources en amont ; les packages sont enregistrés dans le flux même lorsque la source d'origine tombe en panne ou que le package est supprimé.
- Les packages sont versionnés et vous référez généralement à un package en spécifiant la version du package que vous souhaitez utiliser dans votre application.
- De nombreux packages permettent un accès illimité, sans que les utilisateurs aient besoin de se connecter. Cependant, certains packages nous obligent à nous authentifier à l'aide d'une combinaison de nom d'utilisateur et de mot de passe ou d'un jeton d'accès. Concernant ce dernier, les jetons d'accès peuvent être configurés pour expirer après une période de temps donnée.

# Azure Artifacts – Cr ation d'un flux

- Pour cr er un flux d'artifacts dans Azure Artifacts, les packages sont stock s dans des flux, qui sont essentiellement des constructions organisationnelles qui nous permettent de regrouper les packages et de g rer leurs autorisations. Chaque type de package (NuGet, npm, Maven, Python et Universal) peut  tre stock  dans un seul flux.
- Dans le menu Artifacts



# Azure Artifacts – Cr ation d'un package avec une pipeline

- Après la cr ation du flux, nous allons cr er un pipeline de build qui cr e automatiquement un package lors de la construction du projet. Pour cet exemple, on choisira un projet nomm  « PartsUnlimited » en .net.
- Nous allons ajouter tous les mod les   un package et le distribuer   partir d'Artifacts. De cette fa on, vous pouvez facilement partager le mod le de donn es entre diff rents projets.

# Azure Artifacts – Cr ation d'un package avec une pipeline

- La premi re  tape consiste   importer le r   entiel GitHub dans l'organisation PartsUnlimited dans Azure DevOps.
  - 1- Acc dez au projet PartsUnlimited dans Azure DevOps et acc dez   D p t > Fichiers.
  - 2- S lectionnez Importer le r   entiel dans la liste d roulante PartsUnlimited.
  - 3- Entrez l'URL du r   entiel source dans la zone URL de clonage et ajoutez un nom pour votre nouveau r   entiel GitHub
  - 4- Cliquez sur Importer.
- Apr s avoir import  le projet PartsUnlimited.Models dans un r   entiel Azure DevOps, nous pouvons l'utiliser dans un pipeline de build pour en cr er un package NuGet.

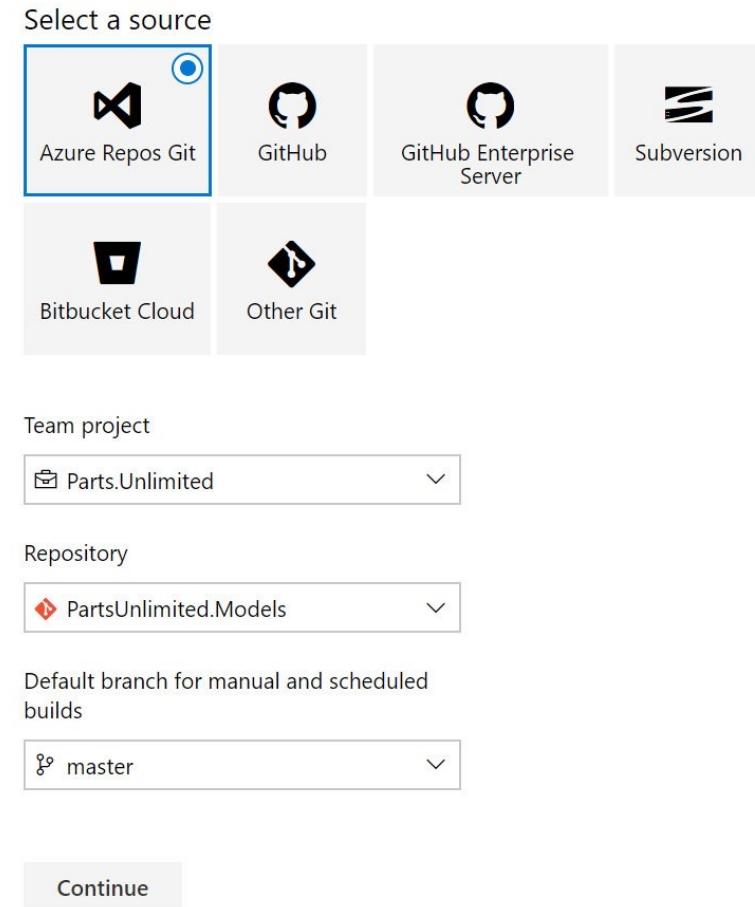
# Azure Artifacts – Cr ation d'un package avec une pipeline

- Apr s que le projet est  t  ajout  au r  f rentiel, nous pouvons cr er le pipeline de build.

1- Acc dez   Azure DevOps et ouvrez   nouveau le projet

PartsUnlimited.Models. Dans le menu de gauche, cliquez sur Pipelines.

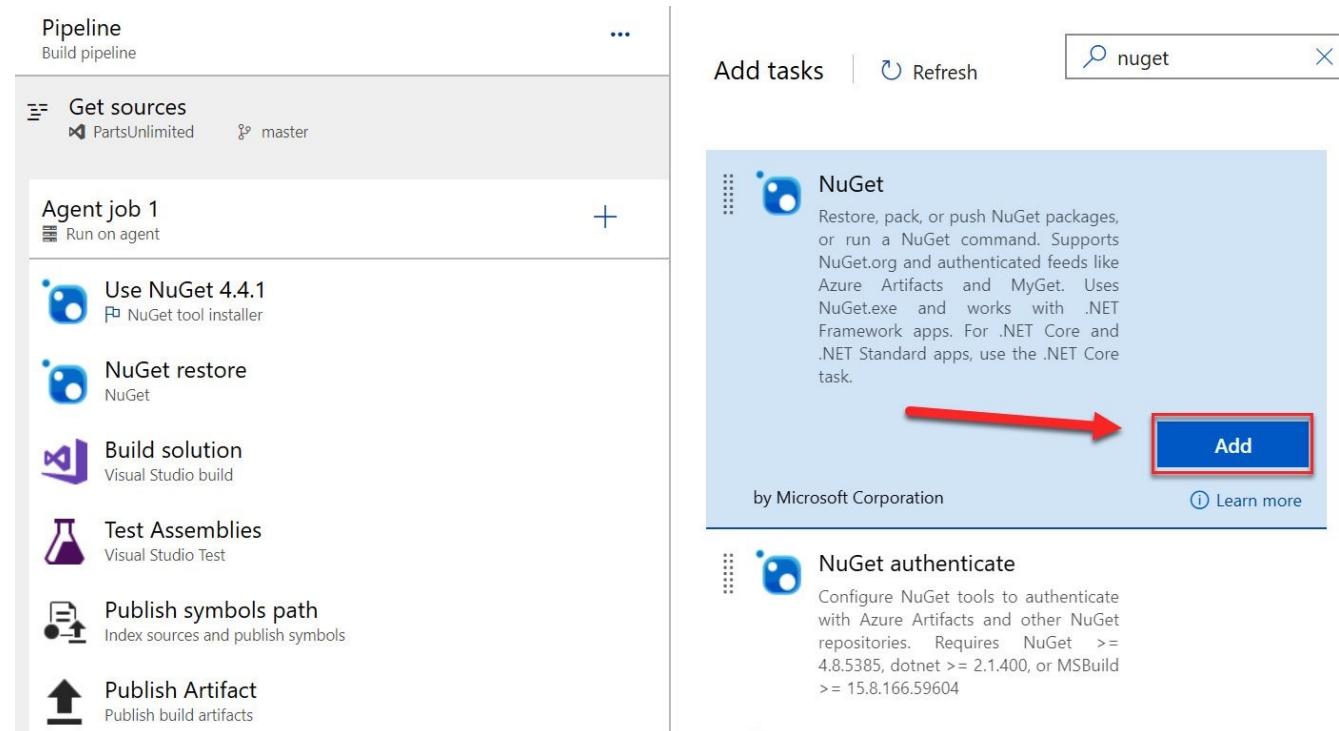
2- Cliquez sur Nouveau pipeline dans le menu en haut   droite et s lectionnez Utiliser l diteur classique sur le premier  cran de l'assistant.



# Azure Artifacts – Cr ation d'un package avec une pipeline

3- Selectionnez ASP.NET sur l' cran suivant de l'assistant et cliquez sur Appliquer.

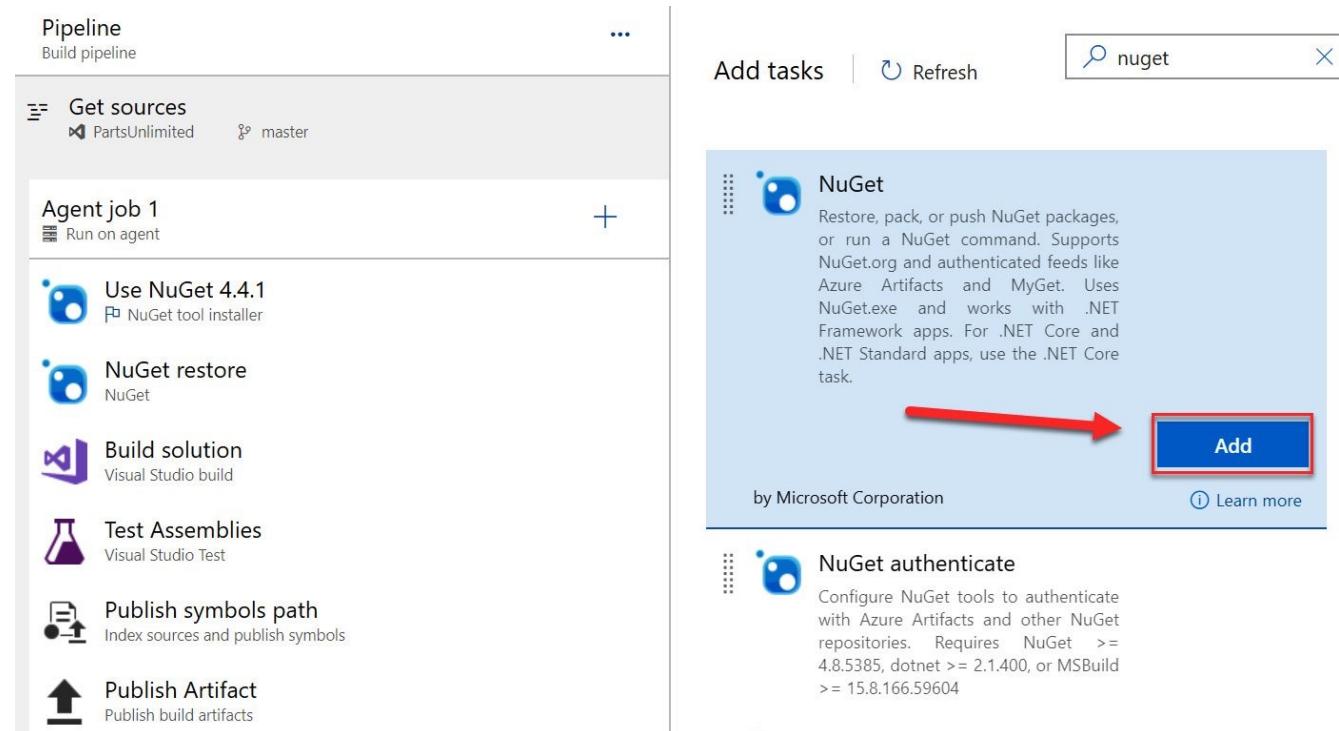
4- Cliquez sur le signe +   droite de la t che d'agent 1 et recherchez NuGet. Ajouter la t che NuGet au pipeline



# Azure Artifacts – Cr ation d'un package avec une pipeline

3- Selectionnez ASP.NET sur l' cran suivant de l'assistant et cliquez sur Appliquer.

4- Cliquez sur le signe +   droite de la t che d'agent 1 et recherchez NuGet. Ajouter la t che NuGet au pipeline



# Azure Artifacts – Publication d'un artifacts

- Après avoir construit l'application et le package à partir de notre pipeline de build, nous pouvons publier le package dans le flux que nous avons créé.
- Pour cela, nous devons définir les autorisations requises sur le flux. L'identité sous laquelle la compilation sera exécutée doit disposer des autorisations de contributeur sur le fil. Une fois ces autorisations définies, nous pouvons étendre notre pipeline pour pousser le package vers le flux.

# Azure Artifacts – Publication d'un artifacts

- Maintenant que l'identité du pipeline de construction dispose des autorisations requises sur le flux, nous pouvons lui envoyer le package pendant sa construction.
- Publication du package
- Nous sommes maintenant prêts à étendre notre pipeline de construction et à pousser le package de celui-ci vers le flux. Pour ce faire, nous devons effectuer les étapes suivantes :
  - Accédez à Azure DevOps et ouvrez le projet PartsUnlimited.Models. Cliquez sur Pipelines dans le menu de gauche.
  - Sélectionnez le pipeline de build que nous avons créé à l'étape précédente et cliquez sur le bouton Modifier, qui se trouve dans le menu en haut à droite.
  - Cliquez à nouveau sur le bouton + à côté de la tâche d'agent 1 et recherchez NuGet. Ajoutez la tâche au pipeline.
  - Faites glisser la tâche nouvellement ajoutée sous la tâche NuGet que nous avons créée à l'étape précédente. Apportez les modifications suivantes aux paramètres de la tâche :
    - --Nom d'affichage : poussée NuGet
    - --Commande : push
    - --Chemin d'accès au(x) package(s) NuGet à publier :  
\$(Build.ArtifactStagingDirectory)/\*\*/\*.nupkg;!\$(Build.ArtifactStagingDirectory)/\*\*/\*.\*symbols.nupkg
    - --Emplacement cible du flux : cette organisation/cette collection
    - --Flux cible : PacktLearnDevOps