

# Formation AzureDevops

Ihab ABADI / UTOPIOS

# Objectifs

- Mettre en place une intégration continue avec Azure
- Maîtriser la chaîne de déploiement continu : les bons réflexes, les outils, les rituels collaboratifs
- Développer le culte de la mesure et mettre en œuvre les bonnes pratiques

# Sommaire

- 1. Intégration continue et Azure DevOps**
- 2. Le contrôleur de code source (Azure Repos)**
- 3. Les Builds (Azure Pipeline)**
- 4. Fonctionnement de l'agent pool (Azure Pipeline)**
- 5. Mettre en place des tests (Azure Test Plan)**
- 6. La gestion des releases (Azure Artifacts)**
- 7. Les release et leurs déploiements (Azure Artifacts)**
- 8. Validation et déclenchement des déploiements (Azure Artifacts)**

# Intégration continue et Azure DevOps



# CI-CD

- Le **CI/CD** est un acronyme qui combine les concepts de l'Intégration Continue (Continuous Integration, CI) et du Déploiement Continu (Continuous Deployment) ou de la Livraison Continue (Continuous Delivery, également CD).
- Ces pratiques sont centrales dans le domaine du génie logiciel, surtout dans le cadre du développement agile et de la DevOps.
- L'approche **CI/CD** permet d'augmenter la fréquence de distribution des applications grâce à l'introduction de l'automatisation au niveau des étapes de développement des applications.

# CI-CD

## Intégration continue

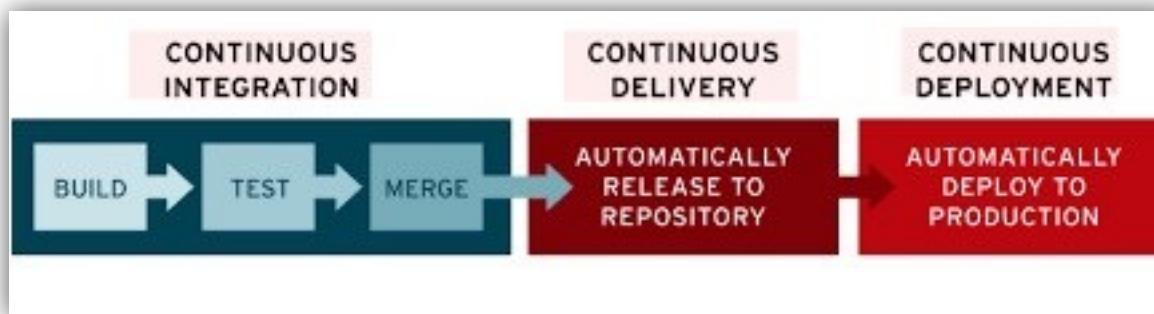
Une méthode de développement logiciel dans laquelle le logiciel est reconstruit et testé à chaque modification apportée par un programmeur.

## Livraison continue

La livraison continue est une approche dans laquelle l'intégration continue associée à des techniques de déploiement automatiques assurent une mise en production rapide et fiable du logiciel.

## Déploiement continu

Le déploiement continu est une approche dans laquelle chaque modification apportée par un programmeur passe automatiquement toute la chaîne allant des tests à la mise en production. Il n'y a plus d'intervention humaine.



# Intégration continue

## Intégration Continue (CI) :

- Cela fait référence à l'automatisation de l'intégration des changements de code dans un projet logiciel.
- Chaque fois qu'un développeur soumet des modifications (commits) dans le répertoire de code source (par exemple, sur GitHub ou Bitbucket), ces modifications sont automatiquement testées à l'aide de scripts d'automatisation.
- Cela permet de détecter rapidement les erreurs, les incompatibilités ou les problèmes de performance avant qu'ils ne se propagent dans le projet.

# Livraison Continue

## **Livraison Continue (CD - Continuous Delivery) :**

- Cette pratique va un peu plus loin que la CI en automatisant non seulement le test du code, mais en préparant également ce code pour être déployé dans un environnement de production.
- Cela signifie que, après les tests automatiques, le code est mis dans un état où il peut être déployé manuellement avec le moins d'efforts possible.

# Déploiement continue

## **Déploiement Continu (CD - Continuous Deployment) :**

- Cela représente l'étape suivante de la Livraison Continue, où l'étape de déploiement est également automatisée.
- Si le code passe tous les tests automatiques, il est directement déployé dans l'environnement de production sans intervention humaine.
- Cela permet de s'assurer que les modifications apportées au code sont disponibles pour les utilisateurs finaux aussi rapidement que possible.

# Principes

## Principes :

- **Automatisation** : Automatiser le processus de build et de test pour le code source.
- **Intégration fréquente** : Encourager les développeurs à intégrer souvent leur code, idéalement plusieurs fois par jour.
- **Version de code source** : Utiliser un système de contrôle de version pour suivre et gérer les changements.
- **Feedback rapide** : Fournir un retour immédiat sur l'état de la base de code après chaque intégration.

# Objectifs

## Objectifs :

- **Réduire les risques** : Diminuer les problèmes liés à l'intégration en détectant et en résolvant les conflits rapidement.
- **Améliorer la qualité du produit** : Assurer une qualité constante grâce à des tests automatiques.
- **Accélérer le temps de mise sur le marché** : Réduire le temps nécessaire pour livrer de nouvelles fonctionnalités et corrections.

# Avantages

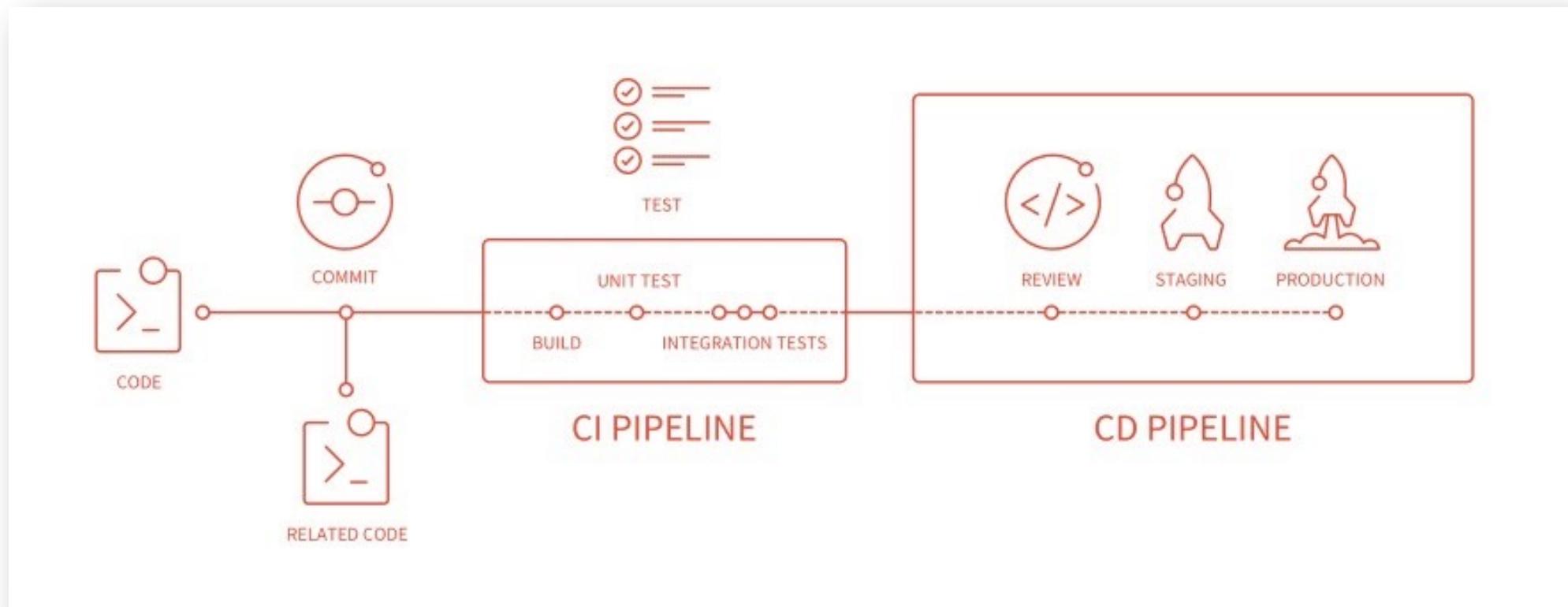
## Avantages :

- **Détection précoce des erreurs** : Les erreurs et les problèmes de compatibilité sont identifiés plus tôt.
- **Meilleure collaboration** : Facilite la collaboration entre les membres de l'équipe en intégrant le travail régulièrement.
- **Transparence** : Offre une visibilité sur l'état du développement et la qualité du code.

# Les différents composants de l'intégration continue

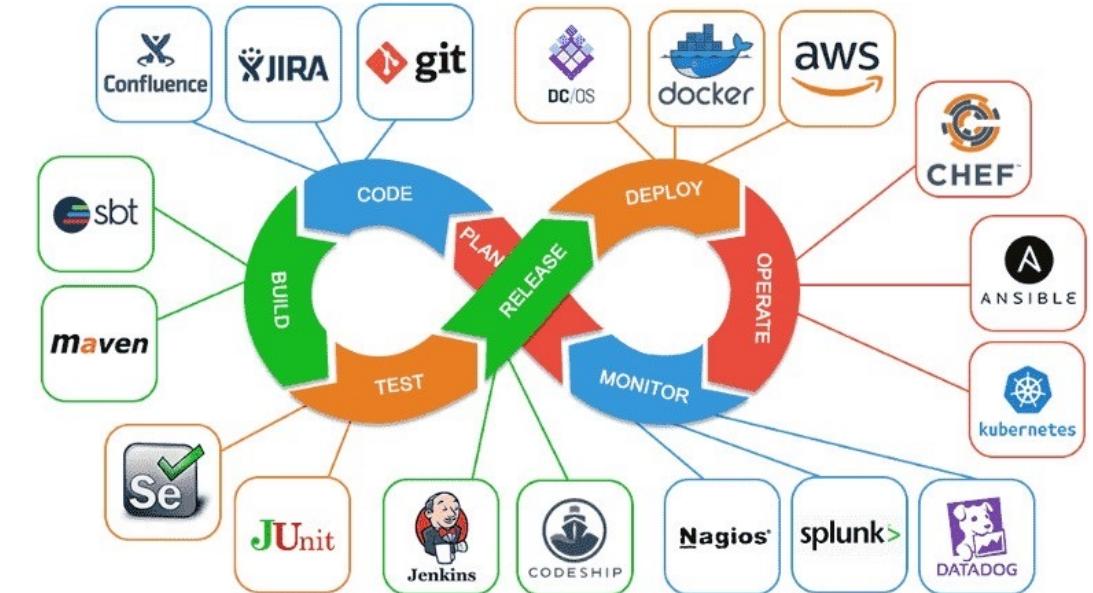
- 1. Système de contrôle de version** : Git, SVN, ou tout autre outil de gestion de versions pour stocker le code source.
- 2. Serveur d'intégration continue** : Jenkins, Travis CI, Azure DevOps, etc., pour orchestrer le processus d'intégration.
- 3. Automatisation des tests** : Cadres et outils de test pour automatiser les tests unitaires, d'intégration, de performance, etc.
- 4. Gestion des artefacts** : Systèmes comme Artifactory ou Nexus pour gérer les artefacts générés lors des builds.
- 5. Outils de notification** : Systèmes pour informer l'équipe des succès ou des échecs de build, comme les e-mails, Slack, etc.

# Les différents composants de l'intégration continue

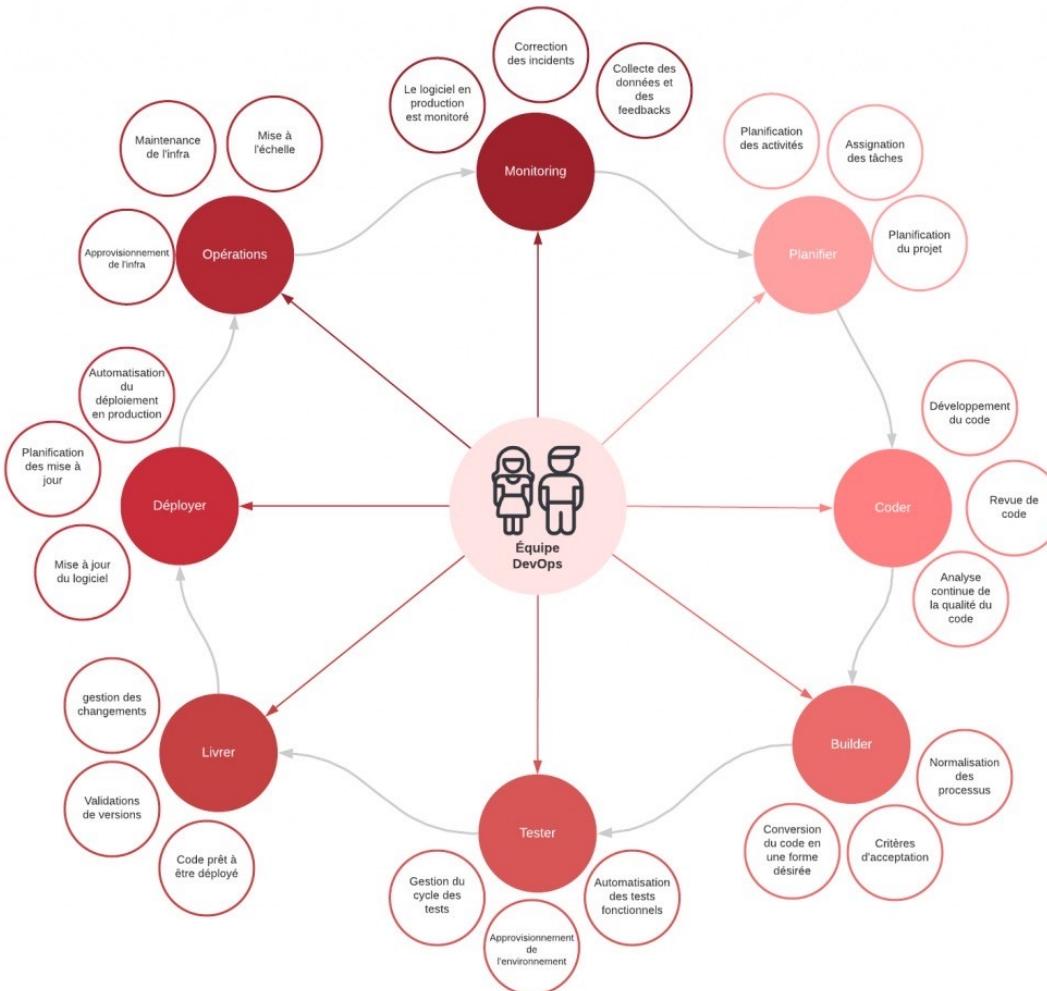


# Les différents composants de l'intégration continue

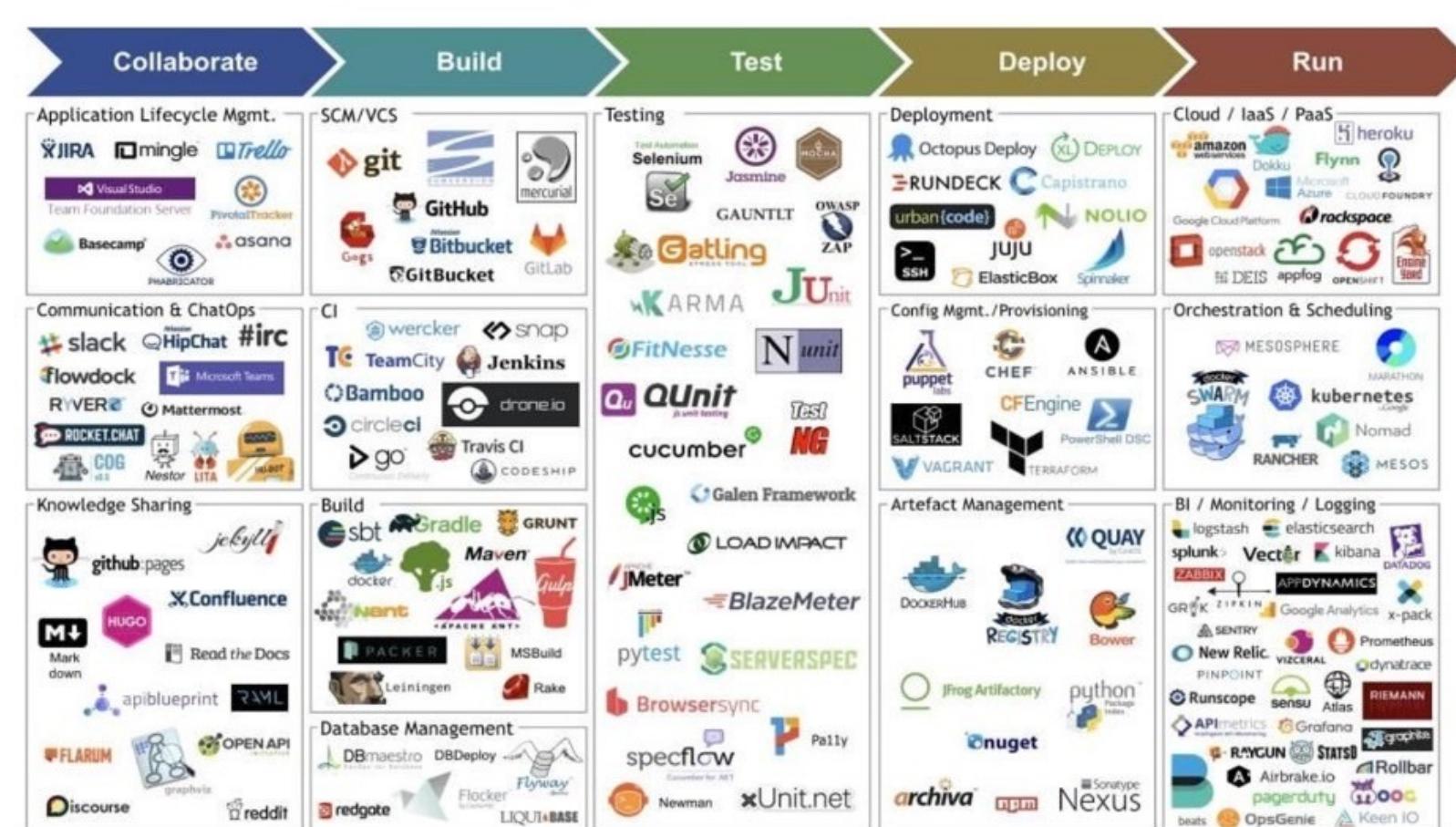
## CI/CD Process



# Les différents composants de l'intégration continue



# Les différents composants de l'intégration continue



# Azure DevOps server- Architecture physique et logique dans Azure DevOps

- **Architecture physique :**
- L'architecture physique d'Azure DevOps Server (on-premises) correspond à l'infrastructure matérielle et logicielle nécessaire pour héberger et faire fonctionner le serveur.
- **Composants principaux :**
  - **Serveur Azure DevOps :**
    - Héberge les services principaux d'Azure DevOps (projets, work items, pipelines, etc.).
    - Fonctionne sur un serveur Windows.
  - **Base de données SQL Server :**
    - Stocke toutes les données des projets, collections, utilisateurs, et configurations.
    - Chaque collection de projets a une base de données dédiée.
  - **Serveur Build/Release (optionnel) :**
    - Dédié aux tâches CI/CD pour réduire la charge sur le serveur principal.
  - **Clients :**
    - Visual Studio, navigateurs web, et intégrations API.
- **Topologie :**
  - Azure DevOps Server peut être déployé sur une seule machine ou réparti sur plusieurs serveurs pour une meilleure performance.
  - En environnement distribué, le serveur applicatif, le serveur SQL, et le serveur Build peuvent être séparés.
- **Haute disponibilité :**
  - Peut être configuré avec des clusters SQL pour la haute disponibilité.
  - Utilise des proxys applicatifs pour équilibrer la charge.

# Azure DevOps server - Architecture physique et logique dans Azure DevOps

- **Architecture logique :**
- L'architecture logique concerne la manière dont les données, les utilisateurs et les projets sont organisés dans Azure DevOps.
- **Niveaux d'organisation :**
  - **Organisation :**
    - Plus haut niveau de hiérarchie, regroupe toutes les collections et projets.
    - Utilisé pour définir les paramètres globaux (sécurité, billing, etc.).
  - **Collection de projets :**
    - Conteneur logique regroupant plusieurs projets.
    - Chaque collection dispose de sa propre base de données dans SQL Server.
    - Utilisé pour isoler les projets en fonction des équipes ou des besoins.
  - **Projet :**
    - Unité principale où sont gérés les work items, le code source, les pipelines, et autres ressources.
- **Avantages de cette architecture :**
  - Permet d'isoler les données sensibles entre les équipes.
  - Facilite la gestion des permissions et des configurations spécifiques à chaque collection ou projet.

# Azure DevOps server - Gestion des rôles et utilisateurs

- **Rôles dans Azure DevOps :**
- Azure DevOps utilise des rôles pour gérer les permissions et les accès aux différents composants du système.
- **Principaux rôles disponibles :**
  - **Administrateurs d'organisation :**
    - Gèrent les paramètres globaux, comme les collections de projets, les modèles de processus, et les utilisateurs.
  - **Administrateurs de collection :**
    - Gèrent une collection spécifique, y compris les projets qu'elle contient et les permissions associées.
  - **Contributeurs :**
    - Utilisateurs ayant le droit de modifier les work items, de commettre du code, et de lancer des pipelines.
  - **Lecteurs (Readers) :**
    - Accès en lecture seule aux projets et aux éléments associés.

# Azure DevOps server - Gestion des rôles et utilisateurs

- **Gestion des utilisateurs :**

1. **Ajout d'utilisateurs :**

1. Allez dans **Organization Settings > Users**.
2. Ajoutez les utilisateurs et assignez-leur des rôles au niveau de l'organisation ou des projets.

2. **Groupes de sécurité :**

1. Azure DevOps propose des groupes prédéfinis, comme **Project Administrators**, **Contributors**, et **Readers**.
2. Vous pouvez créer des groupes personnalisés avec des permissions spécifiques.

3. **Permissions granulaires :**

1. Les permissions peuvent être ajustées au niveau :
  1. Organisation
  2. Collection
  3. Projet
  4. Repository, Pipelines, ou Work Items.
2. Exemple : Un utilisateur peut avoir un accès en lecture seule au backlog, mais un accès en modification aux pipelines.

# Azure DevOps server - Définition des projets

- **Qu'est-ce qu'un projet dans Azure DevOps ?**
- Un projet est l'unité principale de travail dans Azure DevOps. Il regroupe tous les éléments nécessaires pour développer, tester, et livrer un produit ou une fonctionnalité.
- **Contenu d'un projet :**
  - **Boards** : Pour gérer les work items, backlogs, et sprints.
  - **Repos** : Pour stocker le code source avec Git ou TFVC.
  - **Pipelines** : Pour CI/CD (Intégration et Déploiement Continus).
  - **Test Plans** : Pour gérer les plans et cas de test.
  - **Artifacts** : Pour gérer les packages et les dépendances.
- **Étapes pour définir un projet :**
  1. **Créer un nouveau projet :**
    1. Allez dans **Organization Settings > New Project**.
    2. Donnez un nom au projet (ex. "E-commerce Platform").
    3. Sélectionnez un modèle de processus (Agile, Scrum, ou Basic).
    4. Configurez la visibilité (privé ou public).
  2. **Configurer le projet :**
    1. Ajoutez des équipes : Chaque projet peut contenir plusieurs équipes avec leurs propres sprints et backlogs.
    2. Configurez les repositories Git : Importez du code existant ou initialisez un nouveau repository.
    3. Créez des pipelines : Configurez une build pipeline pour automatiser les compilations.
  3. **Gérer les permissions :**
    1. Assignez des rôles aux utilisateurs au niveau du projet (Contributeur, Administrateur, Lecteur).
  4. **Définir les backlogs et sprints :**
    1. Configurez le backlog pour refléter les besoins du projet.
    2. Planifiez les sprints en fonction des objectifs à court terme.

# Concepts d'Azure DevOps

Azure DevOps est une suite de services de développement logiciel de Microsoft qui supporte le développement, le déploiement, et l'intégration continue de logiciels. Ses concepts clés incluent :

- **Boards** : *Outils de planification et de suivi du travail, utilisant des méthodologies agiles.*
- **Repos** : *Gestion de version avec Git pour le stockage et la gestion du code source.*
- **Pipelines** : *Automatisation des builds, des tests, et du déploiement.*
- **Test Plans** : *Planification, exécution, et suivi des tests manuels et automatiques.*
- **Artifacts** : *Gestion des artefacts et intégration avec des feeds de packages pour partager des packages entre différents projets et équipes.*

# Concepts d'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.

# Gestion de projet avec Azure DevOps

**Azure DevOps facilite la gestion de projet en permettant :**

1. La planification agile avec des boards personnalisables.
2. La gestion des sprints et des backlogs.
3. Le suivi des problèmes, des demandes d'évolution, et des bugs.
4. La génération de rapports pour suivre la progression et la productivité de l'équipe.

# Acteurs Intervenant avec Azure DevOps

- **Développeurs** : Intègrent le code dans les repos et participent aux processus de CI/CD.
- **Testeurs** : Utilisent les Test Plans pour planifier, exécuter, et suivre les tests.
- **Chefs de projet** : Suivent la progression du projet à travers les boards et les rapports.
- **Opérations/DevOps** : Configurent les pipelines pour automatiser le déploiement et la livraison.

# Création et Configuration d'un Nouveau Projet

- 1.Création du projet :** Via l'interface Azure DevOps, créez un nouveau projet en spécifiant son nom, sa description, et sa visibilité.
- 2.Configuration :** Configurez les boards, les repos, les pipelines, et les autres services selon les besoins du projet.

# Gestion des droits d'Accès au Projet

**Azure DevOps** permet de gérer finement les droits d'accès en attribuant des rôles et des permissions spécifiques aux membres de l'équipe pour contrôler l'accès aux différentes parties du projet (code, builds, déploiements, etc.).

# 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

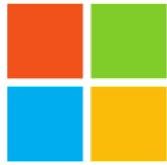
# Azure Pipeline



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

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

# 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 Jamal Hartnett</p> <p>398 Cancel order form Jamal Hartnett</p> <p>Phone Service Web 0/1</p>	<p>486 Welcome back page Raisa Pokrovskaya 3</p> <p>346 Add animated emoticons Christie Church 3</p> <p>Slow response on form Christie Church 8</p>	<p>344 Implement a factory which abstracts Jamal Hartnett 8</p> <p>0/1</p>	<p>405 GPS locator Jamal Hartnett 8</p>

# Azure Boards – Kanban- Checklists

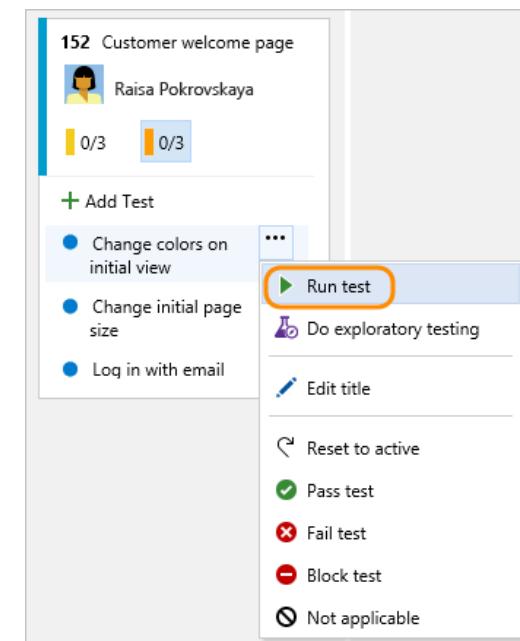
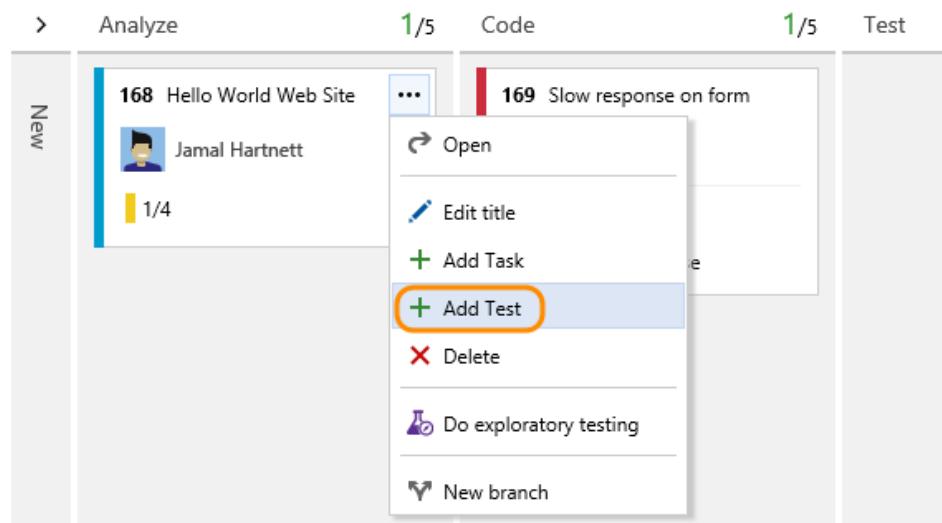
- Les checklists de tâches fournissent un moyen facile et visuel de suivre les détails du travail d'un élément du tableau
- Les tâches d'une checklists peuvent être:
  - Crée directement à partir du Tableau Kanban
  - Réorganiser en drag and drop
- Marqué comme terminé en cochant simplement la case

The screenshot shows a Kanban board in Azure Boards with three columns: Analyze, Develop, and Test. Each column has a card for a task with a checklist.

- Analyze:** Card for "352 Hello World Web Site" by Raisa Pokrovskaya (8).
  - Checklist items:
    - Service [1/3] (checkbox checked)
    - Web [3] (checkbox checked)
  - Buttons: Add Bug, Add Task.
  - Checklist items:
    - Fix performance issues (checkbox unchecked)
    - Auto-complete user's information (checkbox unchecked)
    - Auto-save (checkbox checked)
- Develop:** Card for "1069 Permissions".
  - Checklist items:
    - 0/1 (checkbox checked)
    - 0/1 (checkbox checked)
    - 1/3 (checkbox checked)
  - Buttons: Add Bug, Add Task.
  - Checklist items:
    - Performance issues (checkbox unchecked)
    - Secure sign-in (checkbox checked)
    - Check issues with permissions (checkbox unchecked)
- Test:** Card for "364 Slow response on information form" by Jamal Hartnett (8).
  - Checklist items:
    - Review [2/3] (checkbox checked)
    - Service [1] (checkbox checked)
  - Buttons: Add Bug, Add Task.
  - Checklist items:
    - Benchmark performance (checkbox unchecked)
    - Apply UI updates (checkbox checked)
    - Insert telemetry code (checkbox 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.

# Azure Boards - Gestion du backlog

- Le **backlog** est une liste priorisée des travaux à effectuer pour un projet. Il inclut les Epics, Features, User Stories, et Tasks. Le backlog vous aide à planifier et prioriser les éléments à livrer dans les sprints.
- **Étapes pour gérer un backlog :**
  - 1. Accéder au backlog :**
    1. Allez dans votre projet Azure DevOps.
    2. Cliquez sur **Boards > Backlogs**.
  - 2. Activer les niveaux de hiérarchie :**
    1. Cliquez sur  (engrenage) en haut à droite.
    2. Sous **Backlogs levels**, activez les niveaux souhaités, comme **Epics**, **Features**, ou **User Stories**.
  - 3. Ajouter des éléments :**
    1. Cliquez sur **+ New Work Item** pour ajouter un Epic, une Feature, ou une User Story.
    2. Remplissez les détails comme le titre, la description, et assignez-le à un membre.
  - 4. Organiser et prioriser :**
    1. Glissez-déposez les éléments pour les réorganiser.
    2. Ajoutez des liens pour montrer les dépendances (clic droit > **Add link to**).
  - 5. Planifier les sprints :**
    1. Cliquez sur **Sprints** dans le menu gauche.
    2. Ajoutez les éléments du backlog aux sprints en les glissant dans le sprint correspondant.

# Azure Boards - Gérer des collections de projet

- Une **collection de projet** regroupe plusieurs projets dans une instance Azure DevOps Server. Cela permet de structurer et isoler les projets en fonction des besoins organisationnels.
- **Étapes pour gérer une collection (Azure DevOps Server uniquement) :**

## 1. Accéder à l'administration de la collection :

1. Dans l'interface d'administration Azure DevOps Server, allez dans **Collections**.

## 2. Créer une nouvelle collection :

1. Cliquez sur **New Collection**.
2. Donnez un nom à la collection (ex. "Projets E-commerce").
3. Sélectionnez les paramètres du serveur SQL (base de données associée).

## 3. Ajouter des projets à la collection :

1. Créez un projet et choisissez la collection où l'ajouter.
2. Vous pouvez également migrer un projet existant dans une autre collection.

## 4. Configurer les permissions :

1. Accédez aux paramètres de la collection.
2. Ajoutez des utilisateurs ou groupes avec des rôles spécifiques (Reader, Contributor, Admin).

# Azure Boards - Modèles de processus prédéfinis et héritage

- Azure DevOps propose des **modèles de processus** (comme Agile, Scrum, Basic) pour structurer vos projets. Ces modèles déterminent les types de work items disponibles, les workflows, et les champs par défaut.
- **Étapes pour choisir ou personnaliser un modèle de processus :**

## 1. Choisir un modèle pour un nouveau projet :

1. Lors de la création d'un projet, sélectionnez le modèle de processus :
  1. **Agile** : Pour des projets avec des User Stories, des Tasks, et des Bugs.
  2. **Scrum** : Avec des Product Backlogs Items et des Sprints.
  3. **Basic** : Simplifié, avec uniquement Issues et Tasks.

## 2. Personnaliser un modèle avec héritage :

1. Allez dans **Organization settings > Process**.
2. Dupliquez un modèle existant (ex. Agile) en cliquant sur **Customize**.
3. Modifiez les éléments (ajout de nouveaux champs, personnalisation des workflows).

# Azure Boards- Les work items

- Un **work item** est un élément de travail dans Azure DevOps. Il peut représenter une tâche (Task), une fonctionnalité (Feature), un bug, ou un Epic.
- **Étapes pour gérer les work items :**

## 1.Créer un work item :

1. Accédez à Boards > Work Items.
2. Cliquez sur **New Work Item**.
3. Sélectionnez le type (Epic, Issue, Task, Bug).
4. Renseignez les détails (titre, description, priorité, assigné à).

## 2.Lister et filtrer les work items :

1. Dans Boards > Work Items, utilisez les filtres pour rechercher par **Assigné à**, **État**, ou **Type**.

## 3.Configurer des relations entre les work items :

1. Ouvrez un work item.
2. Cliquez sur **Add Link** pour ajouter une relation (par exemple, un Epic lié à plusieurs Features).

# Azure Boards - Personnalisation des workflows

- Les **workflows** définissent les états possibles d'un work item (ex. Nouveau, En cours, Terminé) et les transitions entre ces états.
- **Étapes pour personnaliser un workflow :**

## 1. Accéder au modèle de processus :

1. Allez dans **Organization Settings > Process**.
2. Sélectionnez votre modèle de processus.

## 2. Modifier un type de work item :

1. Cliquez sur un type (ex. User Story).
2. Ajoutez un nouvel état (ex. "Validation en attente").
3. Définissez les transitions possibles (ex. Nouveau → Validation en attente → Terminé).

## 3. Sauvegarder et appliquer :

1. Appliquez les modifications pour les rendre disponibles dans vos projets.

# Azure Boards- Outils collaboratifs et wiki

- Azure DevOps inclut des outils comme :
- **Wiki** : Pour documenter des spécifications, guides, ou processus.
- **Dashboards** : Pour partager des rapports visuels.
- **Discussion dans les work items** : Pour collaborer sur des tâches.
- **Étapes pour utiliser le Wiki** :

## **1. Créer un Wiki :**

1. Allez dans **Wiki** dans le menu gauche.
2. Cliquez sur **Create Wiki**.
3. Ajoutez une page et remplissez-la avec votre contenu.

## **2. Ajouter des pages et structurer :**

1. Cliquez sur **+ New Page**.
2. Utilisez Markdown pour ajouter des tableaux, des images, et des liens.

## **3. Collaborer :**

1. Invitez les membres de l'équipe à éditer ou commenter les pages.
2. Suivez l'historique des modifications pour traquer les changements.

# Azure Boards

## Point

**Gestion du backlog**

**Collections de projet**

**Modèles prédéfinis et héritage**

**Work Items**

**Personnalisation des workflows**

**Outils collaboratifs et wiki**

## Où le faire

Boards > Backlogs

Administration Azure DevOps Server

Organization Settings > Process

Boards > Work Items

Organization Settings > Process > Workflow

Wiki

## Actions clés

Ajouter des éléments, prioriser, planifier.

Créer/organiser des collections.

Personnaliser les modèles (Agile, Scrum).

Créer, assigner, et relier des work items.

Ajouter des états, transitions, champs.

Créer et structurer la documentation.

# Azure Repos



Azure  
Repos

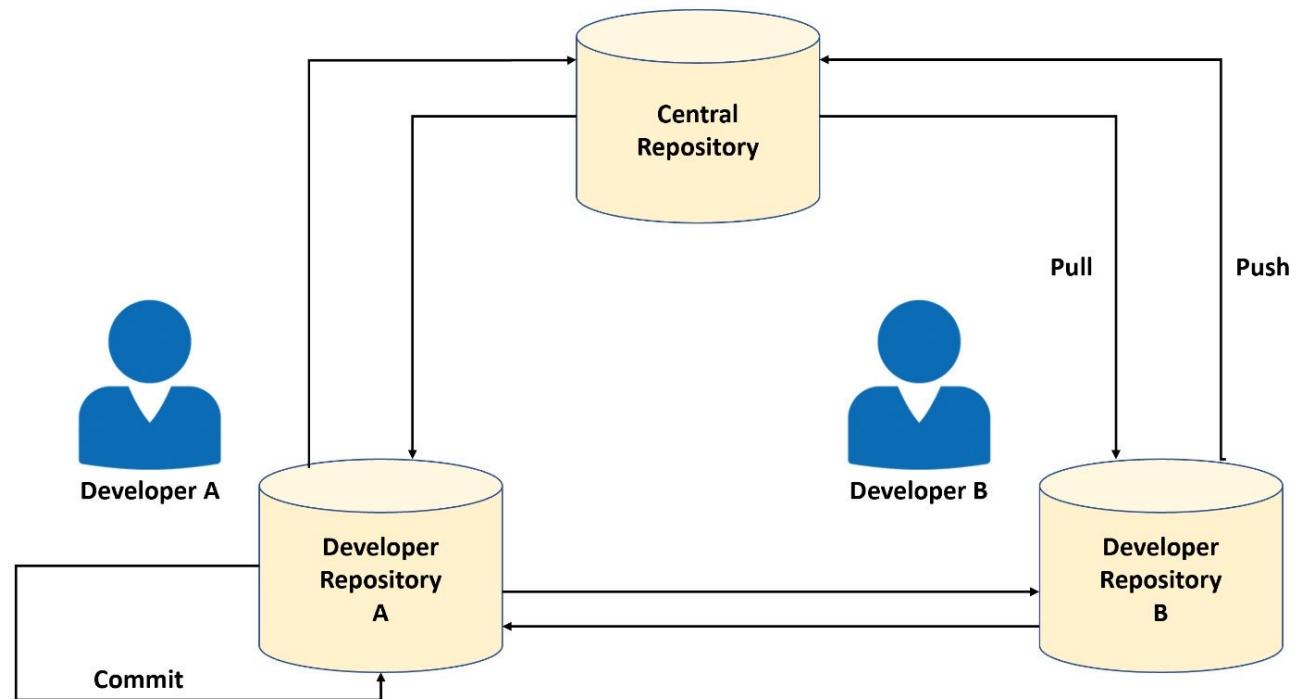
Unlimited cloud-hosted private Git repos. Collaborative pull requests, advanced file management, and more.

# 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).



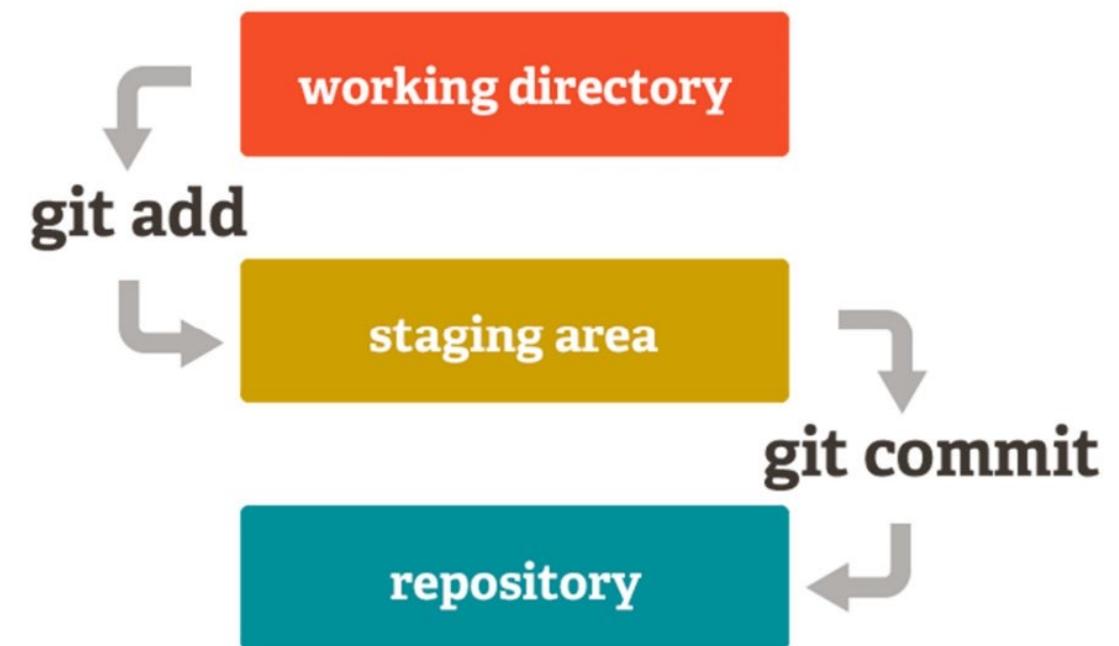
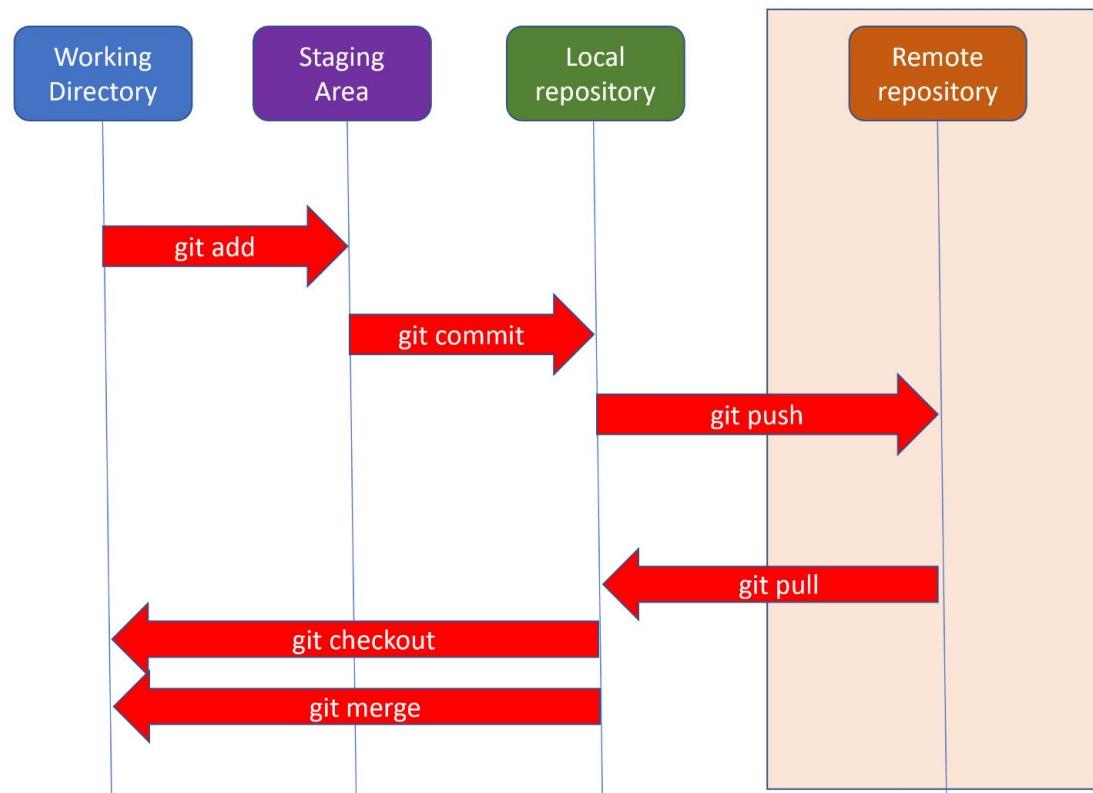
# Azure Repos - Git

- 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

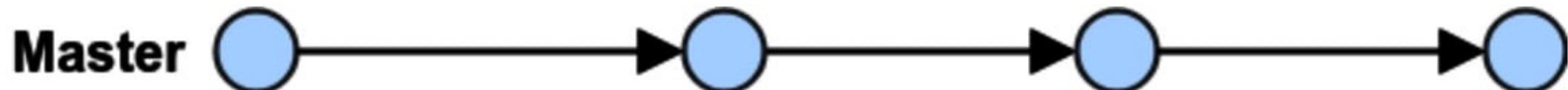


# Azure Repos – Rappel commande base git

- **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)

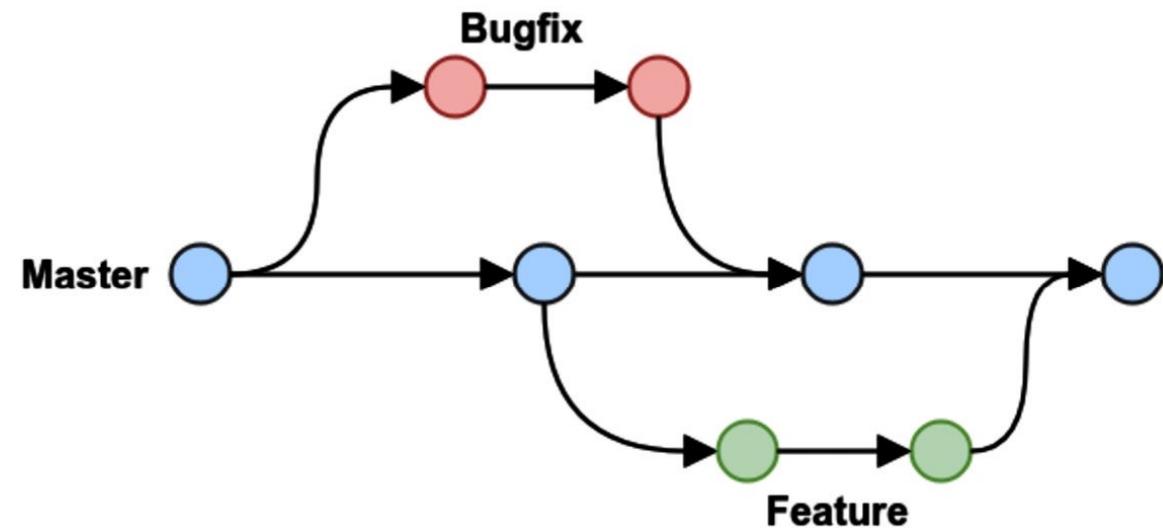


# Azure Repos - Branches

- 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

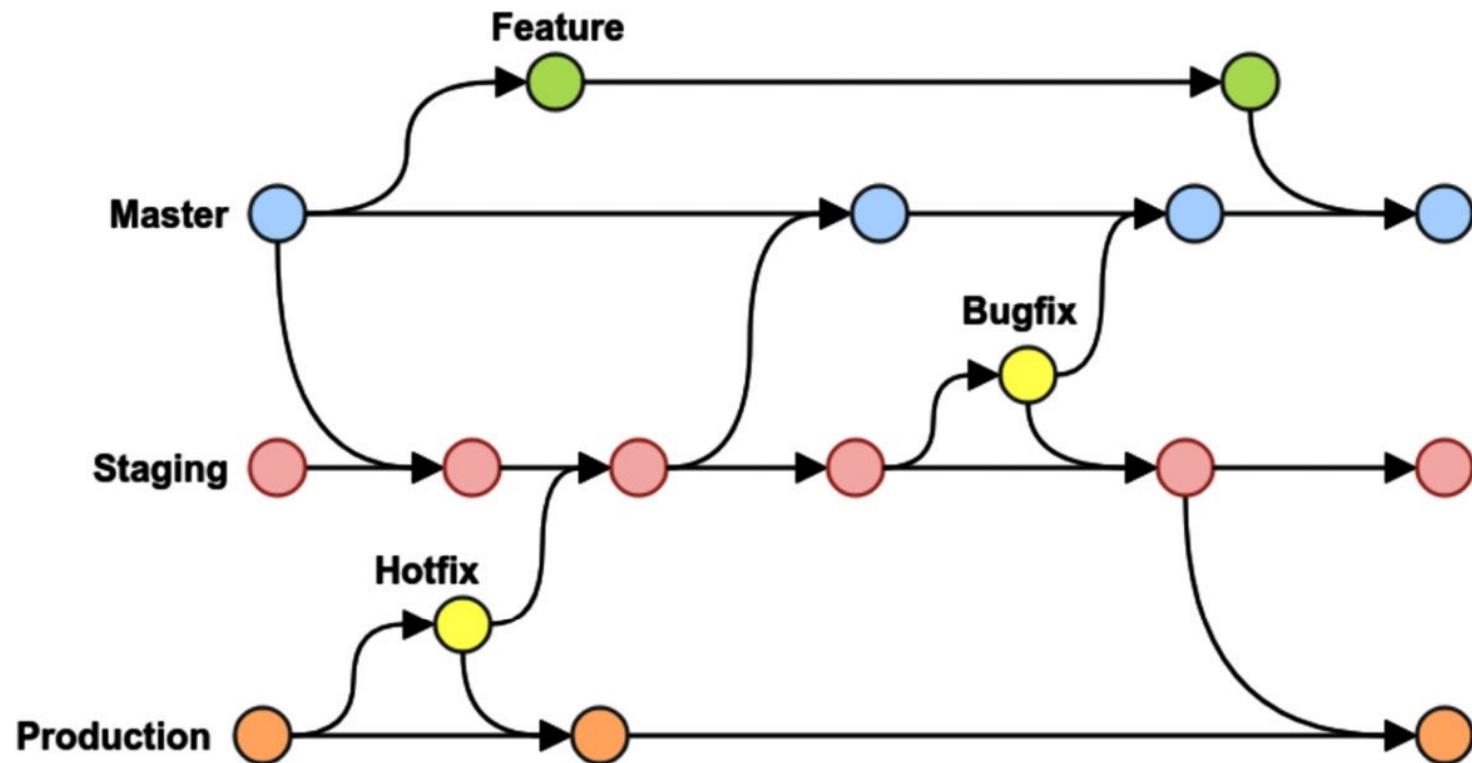
# Azure Repos – Branches – GitHub Flow

- 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



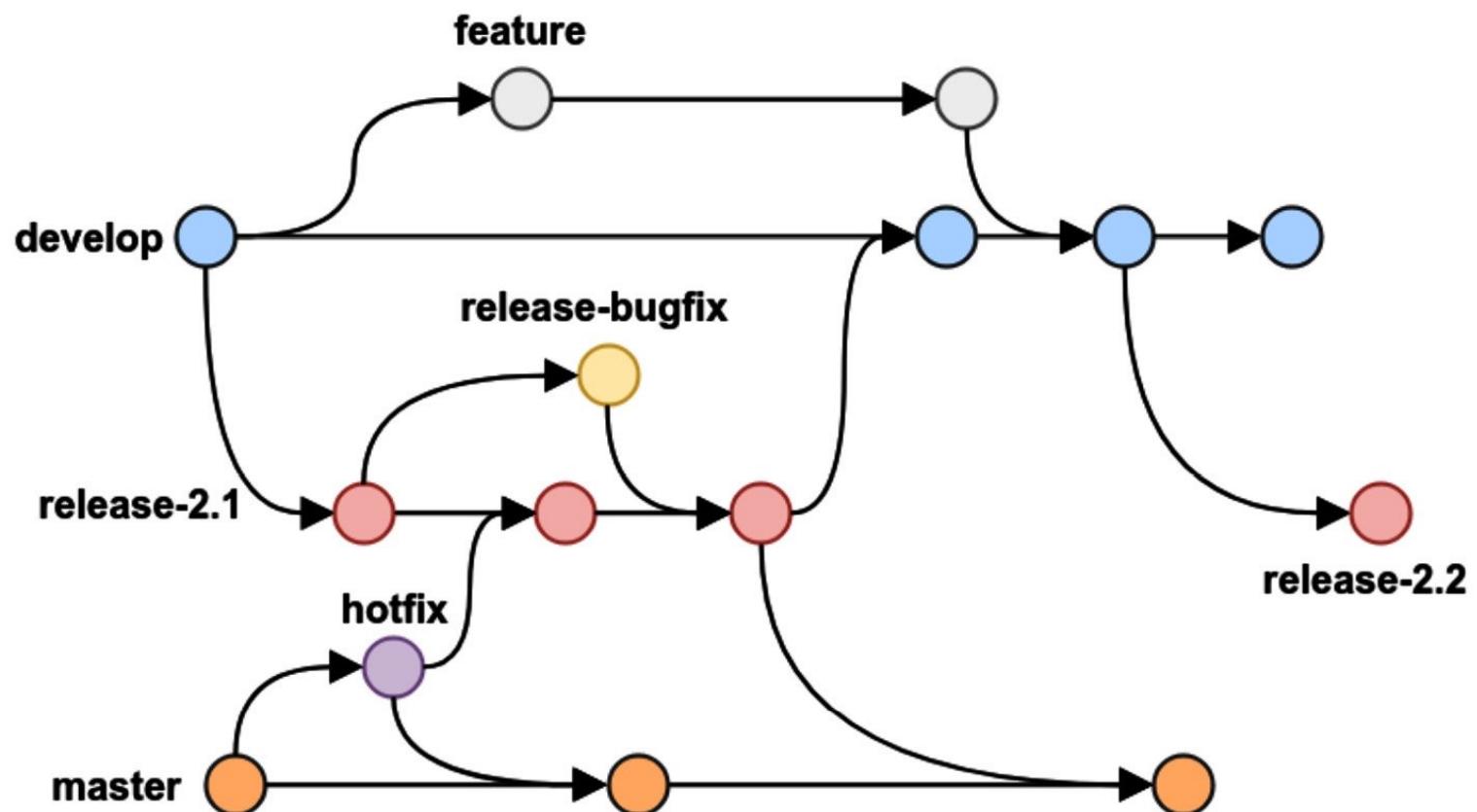
# Azure Repos – Branches – GitLab Flow

- 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 – Crédation 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:**

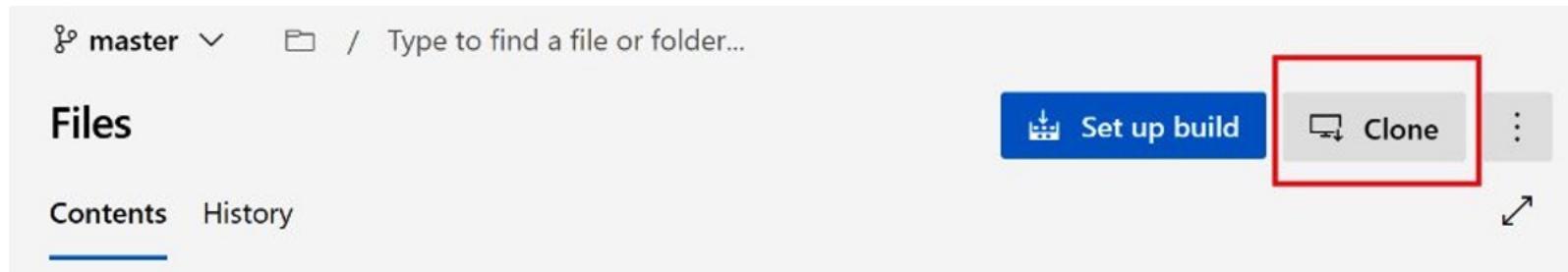
- Project name \***: A text input field.
- Description**: A text input field.
- Visibility**:
  - Public**: Anyone on the internet can view the project. Certain features like TFVC are not supported.
  - Private**: Only people you give access to will be able to view this project. This option is selected.
- Note**: Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).
- Advanced** section:
  - Version control**: A dropdown menu showing "Git" (selected) and "Team Foundation Version Control".
  - Work item process**: A dropdown menu showing "Agile" (selected).

**Repos Hub Sidebar (Left Panel):**

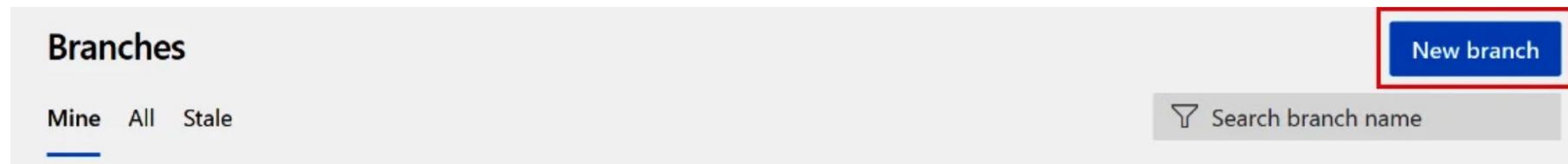
- Overview
- Boards
- Repos** (Selected)
- Files
- Commits
- Pushes
- Branches
- Tags
- Pull requests
- Pipelines
- Test Plans
- Artifacts

# 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 interface for managing branches. The top navigation bar includes 'Repos' and 'Branches' under the 'MyHealthClinic' repository. A search bar and user profile icon are also present. The main area displays a table of branches with columns: Branch, Commit, Author, Authored ..., Behind | Ahead. The 'master' branch is highlighted with a red box and labeled 'Default'. A context menu is open on the right, listing options like 'New branch', 'New pull request', 'Delete branch', 'View files', 'View history', 'Compare branches', 'Set as compare branch', 'Lock', and 'Branch policies'. The 'Branch policies' option is highlighted with a red box. At the bottom right of the interface are a star icon and a three-dot menu icon.

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**

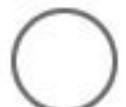
Check to see that all comments have been resolved on pull requests.

### 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.

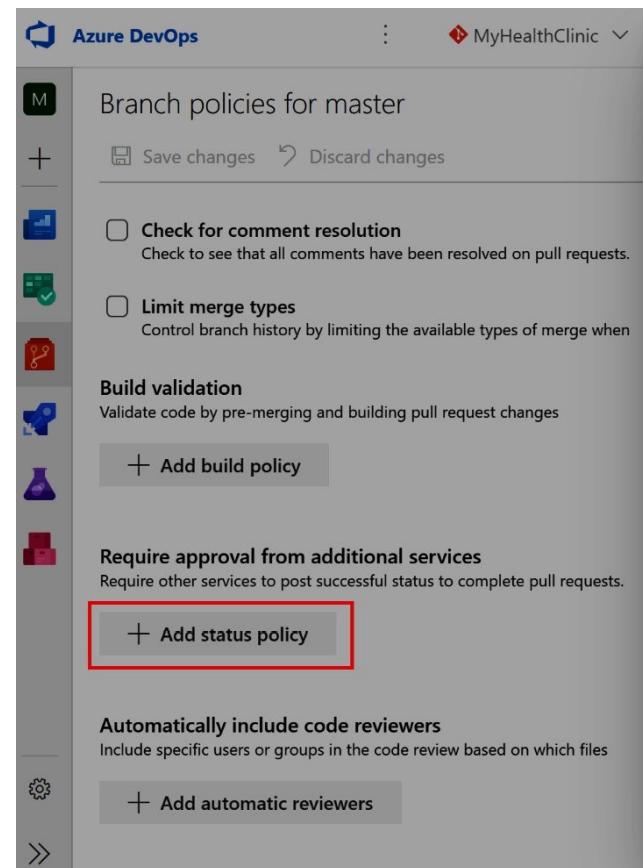
# Azure Repos –Validation des builds

- 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



**Add status policy**

Status policies are passing when a "succeeded" status is posted to the pull request.

Status to check  \*

Policy requirement

Required  
A status of "succeeded" is needed to complete pull requests.

Optional  
A status of "failed" will not block completion of pull requests.

**Advanced**

Authorized organization

Any identity can post this status

Only the following identity can post this status

Search users

Reset conditions

Reset status whenever there are new changes

Save Cancel

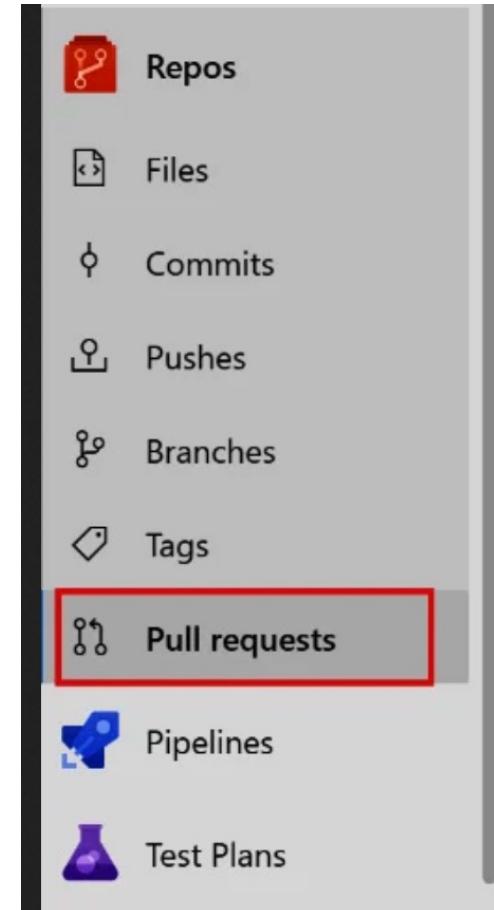
# Azure Repos – Code Review

- 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

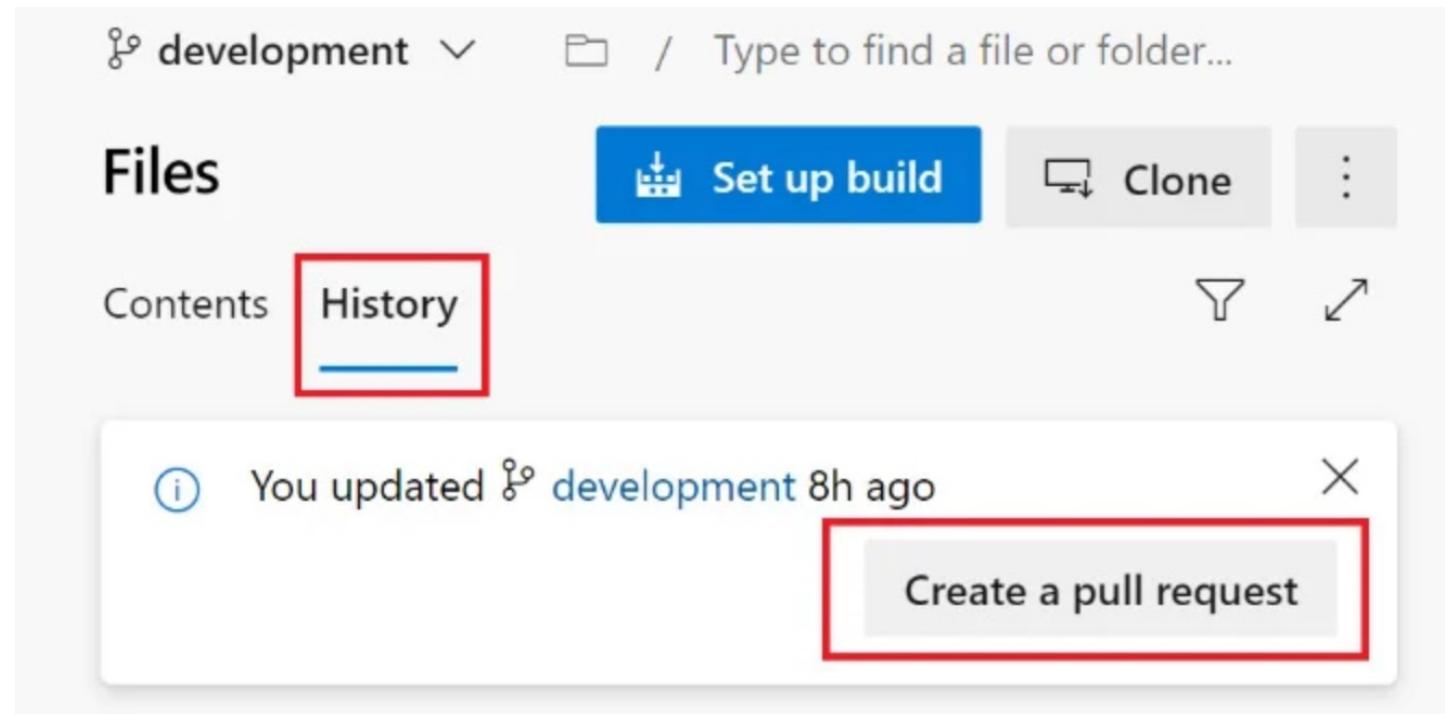
# Azure Repos – Cr ation des PullRequests – Work item

- 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. The main area contains fields for 'Description', 'Priority' (2), 'Remaining Work', and 'Activity'. On the right, there's a 'Deployment' section with a note about tracking releases. Below the main details, the 'Development' section is expanded, showing a 'Create a pull request' button also highlighted with a red box. Other buttons in this section include 'Add link' and 'development'. The 'Related Work' section is partially visible at the bottom.

# 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

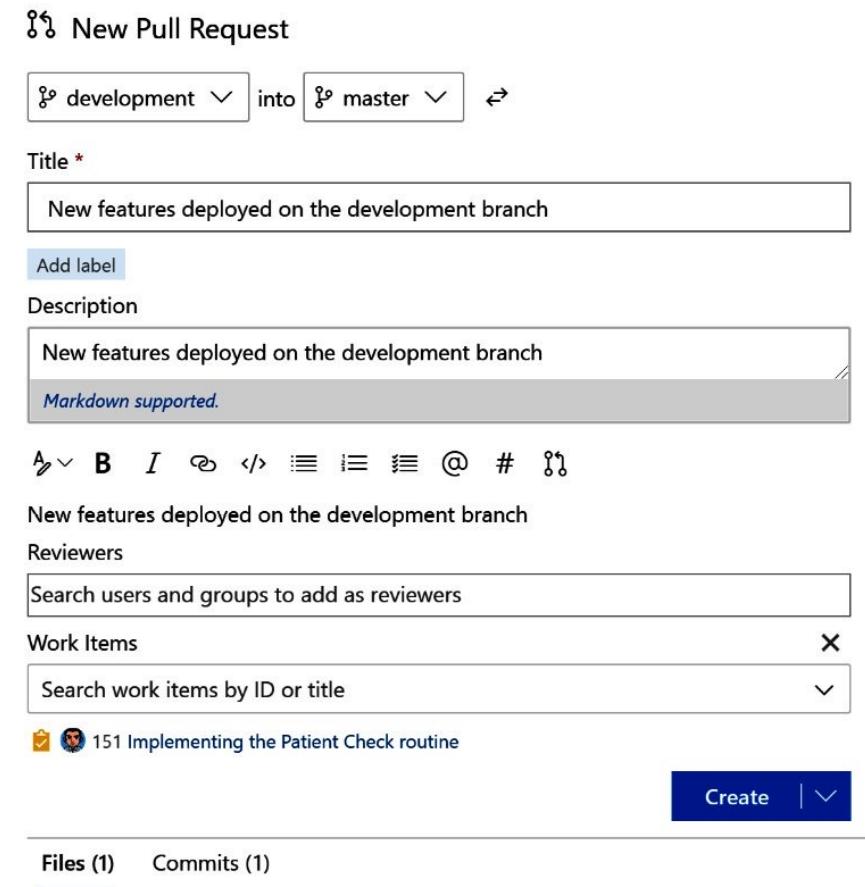
Work Items

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 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.

# 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

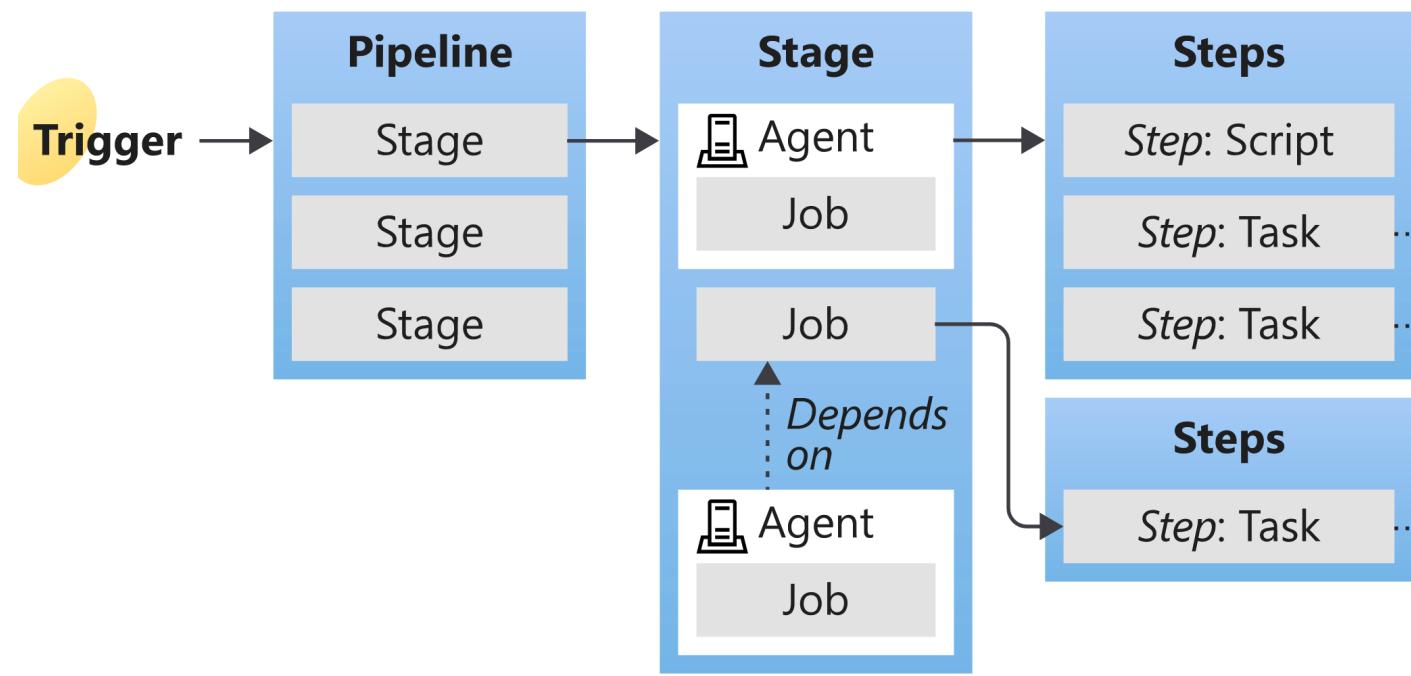
- Le déploiement continue permet de générer, tester et déployer nos applications sur un ou plusieurs environnements.
- Le déploiement continue consomme les artefacts produit par CI.
- Le déploiement continue permet d'améliorer la visibilité sur l'ensemble du processus à l'aide des systèmes de surveillance.

# 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.

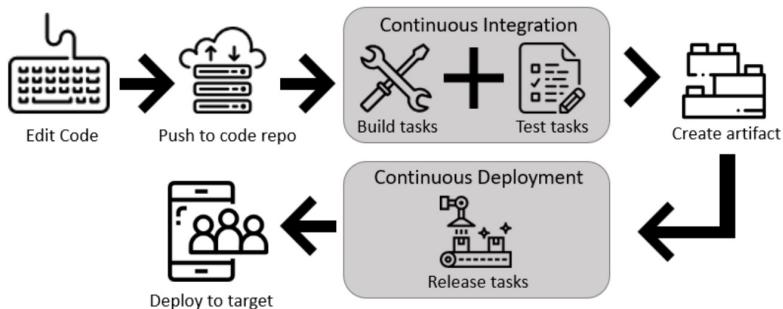
# Azure pipelines – fonctionnement

- 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.

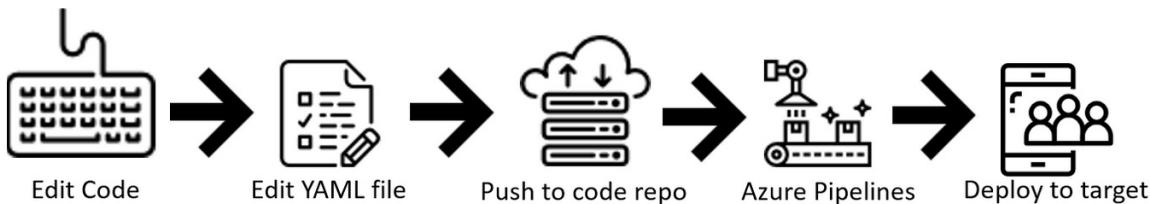


# Azure pipelines – utilisation

- 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.

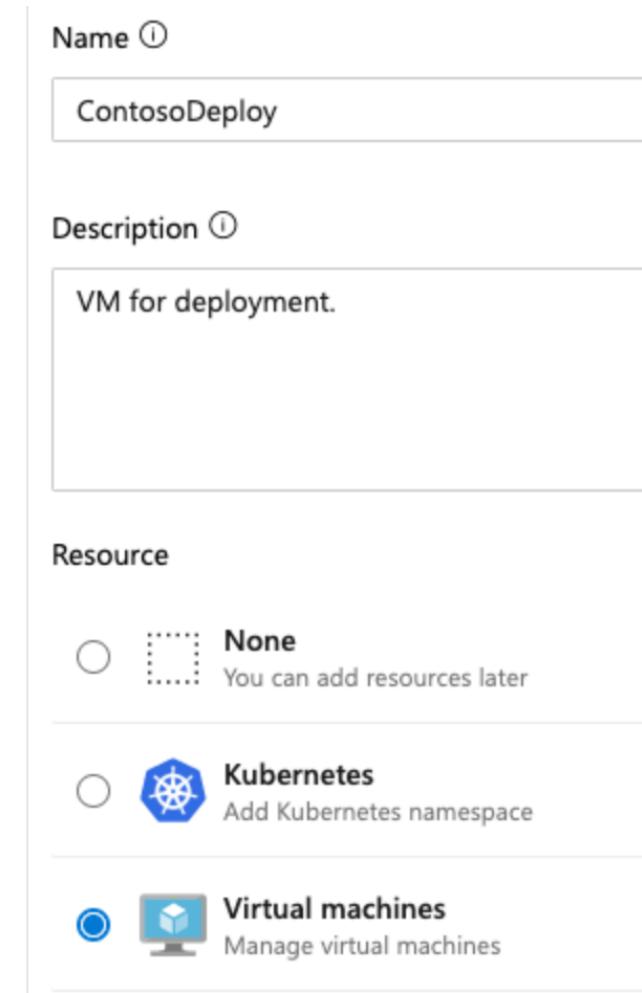
```
task: Docker@2
displayName: Push image
inputs:
containerRegistry: |
$(dockerHub)
repository: $(imageName)
command: push
tags: |
test1
test2
```

# 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 organisée 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

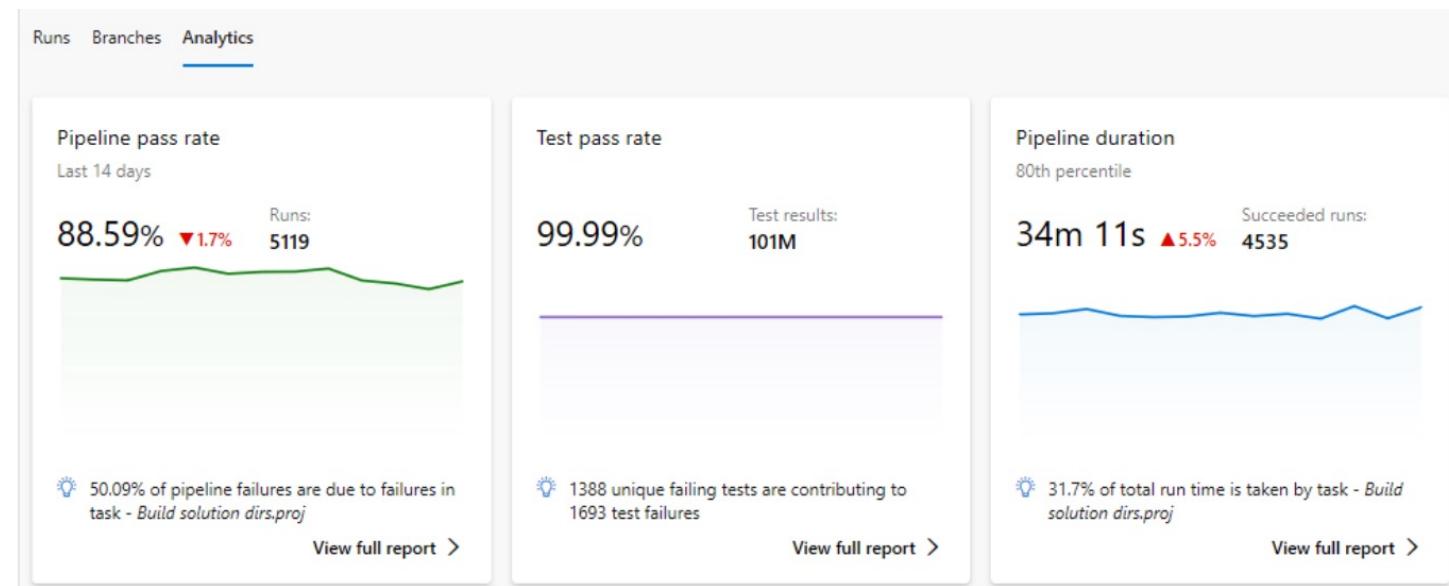
```
# Run functional tests
- task: RunVisualStudioTestsusingTestAgent@1
  inputs:
    testMachineGroup:
    dropLocation:
    #testSelection: 'testAssembly' # Options: testAssembly, testPlan
    #testPlan: # Required when testSelection == TestPlan
    #testSuite: # Required when testSelection == TestPlan
    #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.



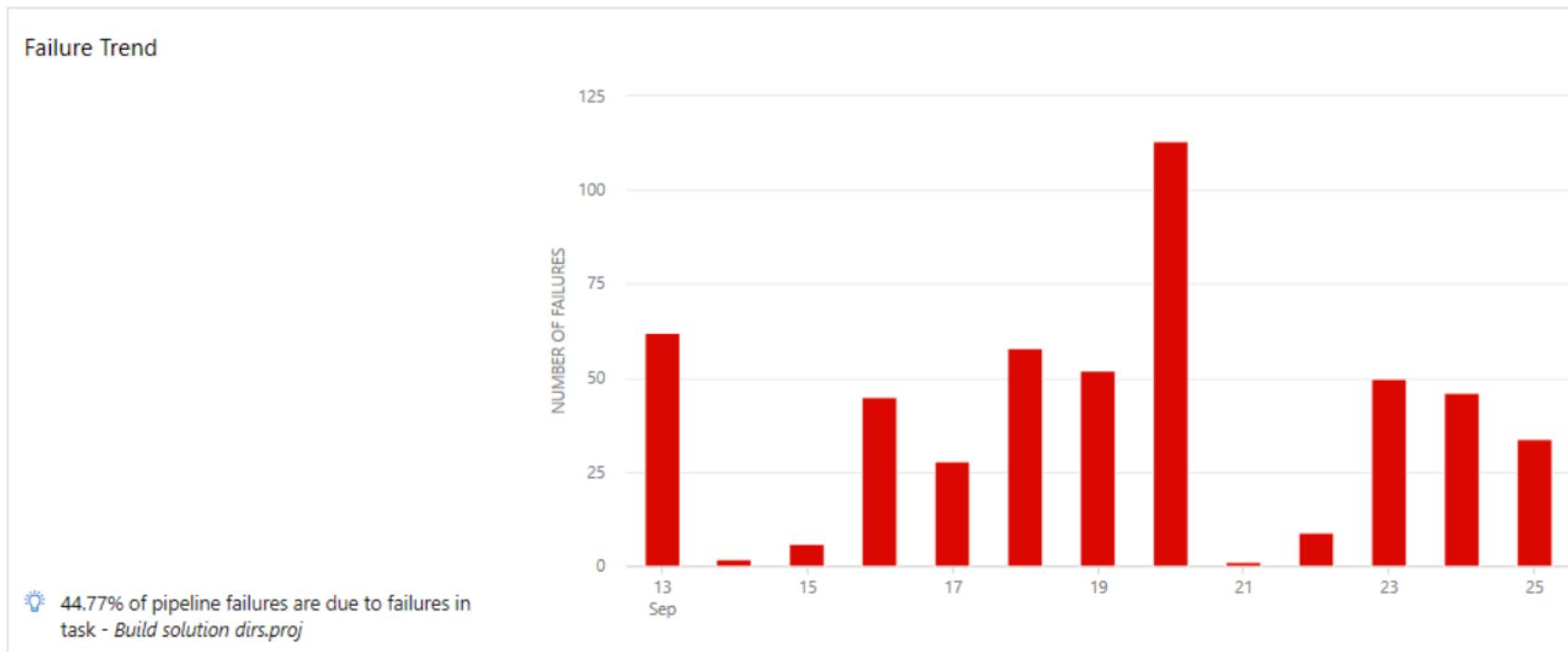
# Azure pipelines – Rapport réussite, échec

- 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.



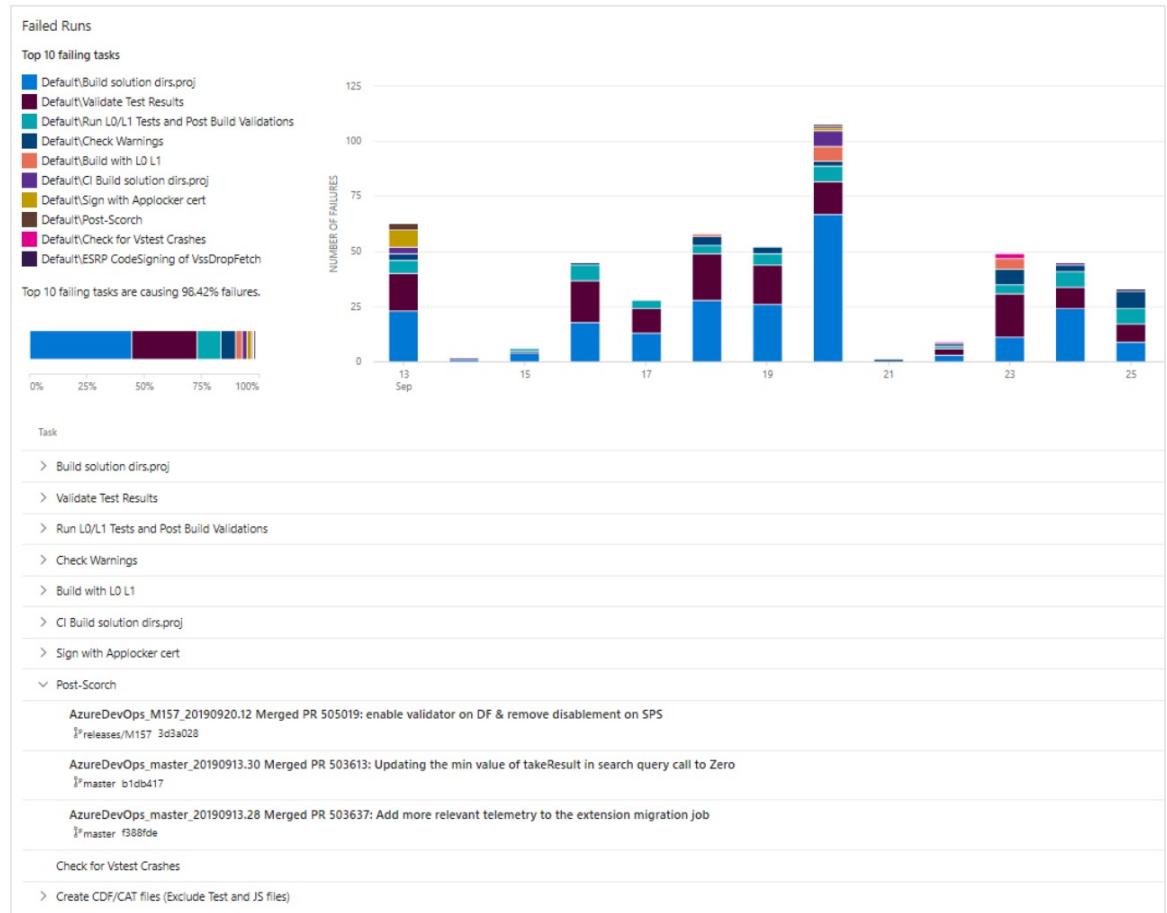
# Azure pipelines – Rapport réussite, échec

- Exemple tendance d'échec.



# Azure pipelines – Rapport réussite, échec

- Focus sur les tâches en échec.



# Azure pipelines – Agent Pool

Un **pool d'agents** est une collection d'agents ayant une configuration similaire et qui sont utilisés pour exécuter des jobs. Dans Azure DevOps, les pools d'agents peuvent être :

- **Globaux** : Disponibles pour tous les projets dans une organisation Azure DevOps.
- **Spécifiques à un projet** : Limités à un projet particulier.

Les pools d'agents permettent aux administrateurs de gérer et d'allouer des ressources de build et de déploiement de manière flexible et efficace. Ils peuvent créer différents pools pour différents types de jobs, exigences de sécurité, ou environnements de développement.

# Azure pipelines – Gestion de la file d'attente

- Quand un pipeline est déclenché (par exemple, par un push dans un dépôt Git), un job est créé et placé dans la file d'attente d'un pool d'agents.
- La file d'attente de jobs est gérée par Azure DevOps, qui sélectionne un agent disponible dans le pool pour exécuter le job.
- Si tous les agents sont occupés, le job attend dans la file d'attente jusqu'à ce qu'un agent se libère.
- Les jobs sont généralement traités selon le principe du premier arrivé, premier servi, mais la priorité peut être ajustée par les administrateurs.

# Azure pipelines – Pipeline concurrents | avec les agents hébergés

- Azure DevOps offre des **agents hébergés** avec un certain nombre de pipelines concurrents, qui dépendent du plan d'abonnement de l'organisation.
- Un pipeline concurrent fait référence à la capacité d'exécuter plusieurs instances de pipelines simultanément.
- Si le nombre maximal de pipelines concurrents est atteint, les pipelines supplémentaires sont mis en file d'attente jusqu'à ce qu'une des exécutions en cours se termine.
- Pour augmenter le nombre de pipelines concurrents, une organisation peut acheter des capacités supplémentaires ou optimiser l'utilisation des pipelines par des stratégies telles que le regroupement des jobs ou l'utilisation d'agents auto-hébergés.

# Azure pipelines – Les agents privés

Les **agents privés (auto-hébergés)** offrent plus de contrôle sur l'environnement de build et de déploiement, permettant aux équipes de configurer des agents avec des outils spécifiques, des versions de logiciels, ou des configurations de sécurité personnalisées. Pour ajouter un agent privé :

- 1. Créer un Pool d'Agents :** *D'abord, créez un nouveau pool d'agents dans Azure DevOps si nécessaire, ou utilisez un pool existant.*
- 2. Télécharger le Logiciel de l'Agent :** *Dans Azure DevOps, naviguez vers la section de gestion des agents et téléchargez le package d'installation de l'agent pour votre système d'exploitation.*
- 3. Installer et Configurer l'Agent :** *Exécutez le script d'installation sur votre machine cible, en suivant les instructions pour configurer l'agent afin qu'il communique avec Azure DevOps et s'ajoute au pool d'agents sélectionné.*
- 4. Vérifier la Connexion :** *Une fois l'installation terminée, vérifiez dans Azure DevOps que l'agent apparaît comme en ligne et prêt à exécuter des jobs.*

# Azure Test Plans



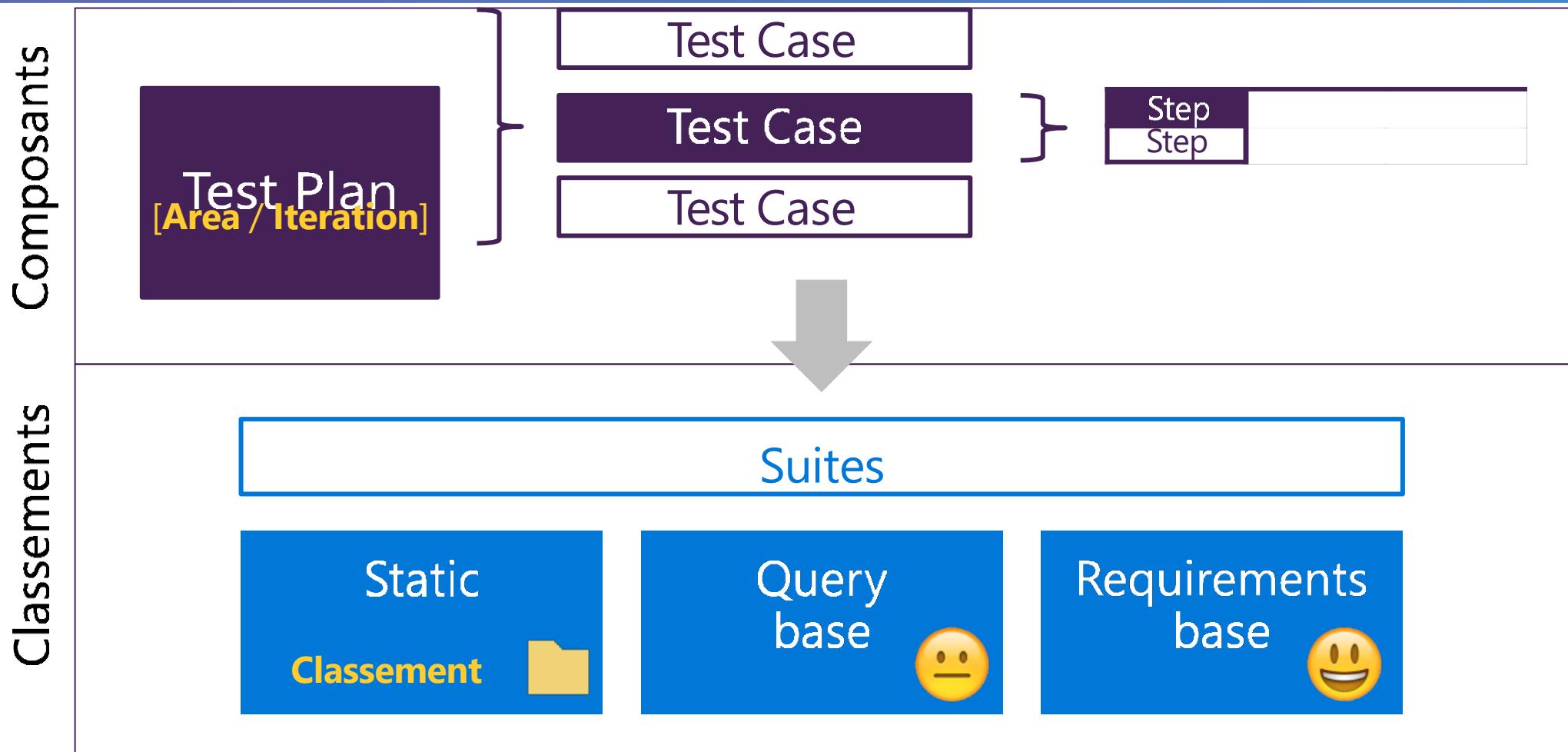
## Azure Test Plans

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

# Tests dans Azure DevOps

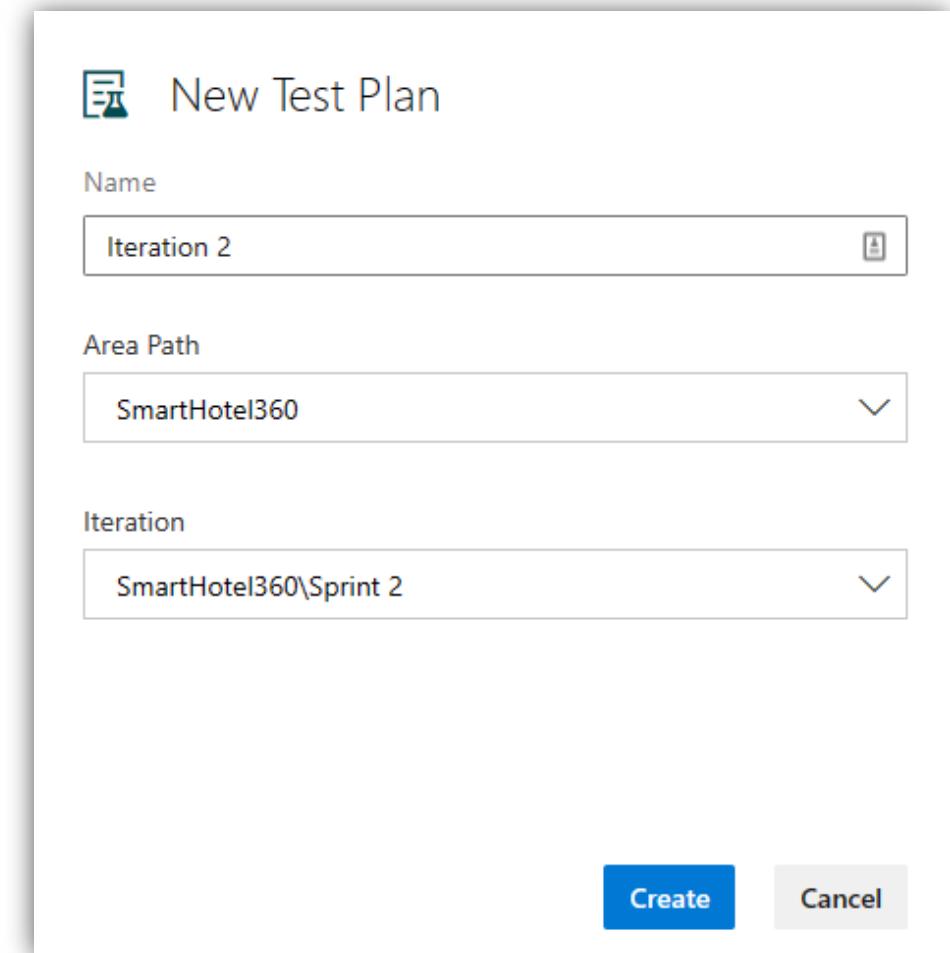
- Azure Test Plans est un service qui fournit un ensemble de fonctionnalités pour exécuter un ensemble de tests.
- Azure Test Plans prend en charge :
  - Tests manuels.
    - Tests manuels planifiés.
    - Test d'acceptation de l'utilisateur.
    - Tests exploratoires.
  - Tests automatisés.
  - Suivi 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, under 'Sprint 1', there are 'Test Suites' listed: '33 : Change initial view (7)', '34 : Welcome back pa...', and '43 : Cancel order form (1)'. The first item is selected. On the right, the details for '33 : Change initial view (ID: 68)' are shown. The 'Define' tab is active, displaying a table of 'Test Cases (7 items)'. The table has columns for Title, Order, Test Case Id, and State. All test cases are currently in the 'Design' state.

	Title	Order	Test Case Id	State
<input type="checkbox"/>	Test welcome page	1	76	Design
<input type="checkbox"/>	Change initial view	2	79	Design
<input type="checkbox"/>	Welcome back	3	80	Design
<input type="checkbox"/>	Resume	4	81	Design
<input type="checkbox"/>	Interim save on long form	5	82	Design
<input type="checkbox"/>	Change colors on initial view	6	83	Design
<input type="checkbox"/>	Revert to previous version	7	84	Design

**Product Backlog Items**

Ihab A B A D I - U T O P I O S

**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 (1) (4)

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

Development

Add link

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 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

**⋮**

**⋮**



# 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 Azure Boards.

The image shows two screenshots from the Azure Boards interface. On the left, a requirement card for user story 770 is displayed. The card includes details like 'As a room guest, I should be able to communicate with all smart devices from the app', priority 2, state New, and area path SmartHotel360. A red box highlights the 'Test' icon (a flask) in the bottom right corner. A context menu is open over the card, with the 'Add Test' option highlighted by a red box. On the right, a modal dialog titled 'Add New Linked Work Item to' is shown, prompting the user to select a link type and details. The 'Test Case' link type is selected and highlighted with a red box. The footer of the slide contains the text 'Ihab ABADI - UTOPIOS'.

# 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 contains multiple backlog items (cards) with various details such as priority, state, and area path.

- New Column:** Contains 7 backlog items.
  - Card 763: As a customer, I would like to reserve a conference room. Priority: 2, State: New, Area Path: SmartHotel360. Status: Unassigned, 0/2 tasks completed.
  - Card 764: As a reservation agent, I would like to send confirmations to guest. Priority: 2, State: New, Area Path: SmartHotel360. Status: Unassigned, 0/4 tasks completed.
- Approved Column:** Contains 15 backlog items.
  - Card 785: As a room guest, I should be able to control my room's lighting within the app. Priority: 2, State: Approved, Area Path: SmartHotel360. Status: Unassigned, 0/2 tasks completed.
  - Card 786: Search hotel works only for certain cities. Priority: 2, State: Approved, Area Path: SmartHotel360. Status: Unassigned, 1/4 tasks completed.
- Build and Test Column:** Contains 5 backlog items.
  - Card 787: Search hotel bug. Priority: 2, State: Committed, Area Path: SmartHotel360. Status: Admin, Manager, 1/5 tasks completed.
- Deploy Column:** Contains 4 backlog items.
  - Card 775: be able to search for rooms. Priority: 1, State: Not Started, Area Path: SmartHotel360. Status: 5/5 tasks completed.
  - Card 776: be able to search for guests. Priority: 1, State: Not Started, Area Path: SmartHotel360. Status: 4/4 tasks completed.

# 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 'Win10 + Edge' and 'Win10 + Firefox'. Below this, 'All configuration variables' are listed under 'Browser' and 'Operating System'. A configuration variable 'Win10 + Edge' is selected and shown in detail. The 'Name' field contains 'Win10 + Edge' (highlighted with a red box). The 'Description' field is 'Default operating system for testing'. The 'State' is set to 'Active'. A checked checkbox 'Assign to new test plans' is present. Under 'Configuration variables', there are two entries: 'Browser' with value 'Microsoft Edge' and 'Operating System' with value 'Windows 10'. A link '+ Add configuration variable' is visible at the bottom.

# Configuration & Paramètres

The screenshot shows the SmartTest 2020 application interface. On the left, the navigation bar includes 'GOVOLUTION / SMARTTEST2020 / 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 on the left 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:

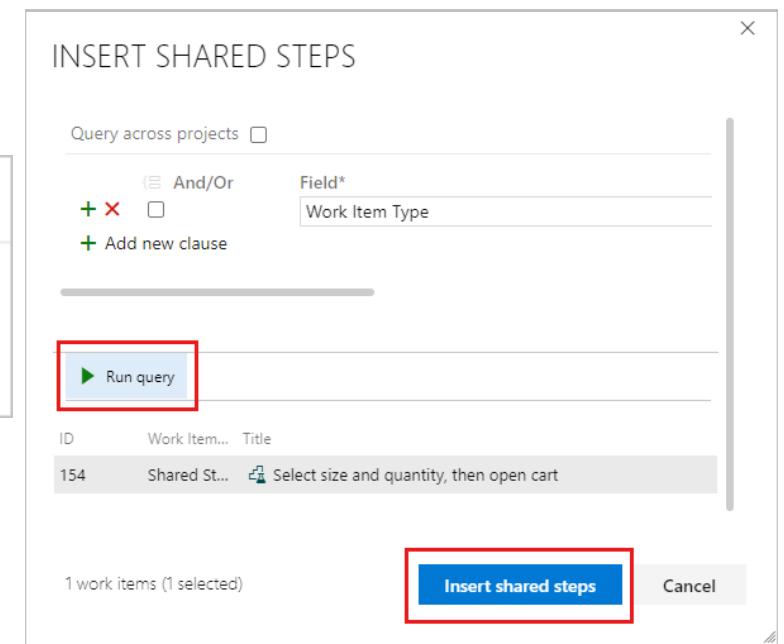
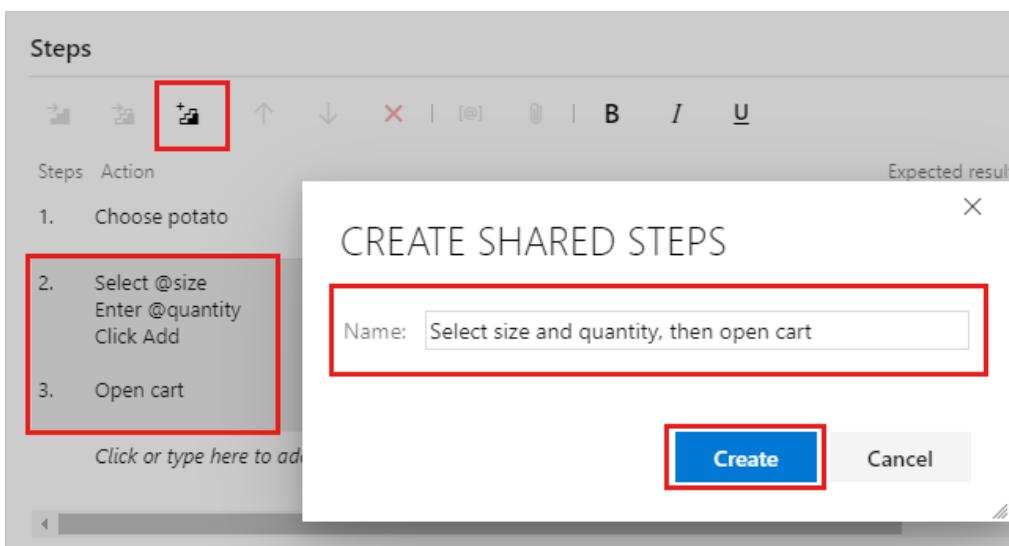
1. Launch the application and login into the application as a customer using @Login.  
EXPECTED RESULT  
Customer should be able to login into the application with his credentials.  
Login = Admin
2. Click on the Profile and select Saved Cards  
EXPECTED RESULT  
Customer should get navigated to Saved 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 'View report' link is present. Under 'Test Suites', 'Sprint 2 (15)' is expanded, showing 'Basic testing (8)' and 'Special testing (10)' (highlighted with a red box). The 'Special testing (10)' card has tabs for 'Define', 'Execute' (highlighted with a red box), and 'Chart'. Below is a list of 'Test Points (10 items)':

- Title
- Interim save on long form
- Interim save on long form
- Change colors on initial view
- Change colors on initial view
- Revert to previous version

On the right, there are two cards: 'Test Suites' (Sprint 2 (6)) and 'Test Points (6 items)'. The 'Test Points' card has a dropdown menu set to 'Run for web application' (highlighted with a red box). The list of test points includes:

- Change initial view
- Welcome back
- Resume
- Interim save on long form
- Change colors on initial view
- Revert to previous version

# 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

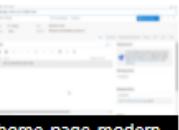
83\*: Change colors on initial view

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

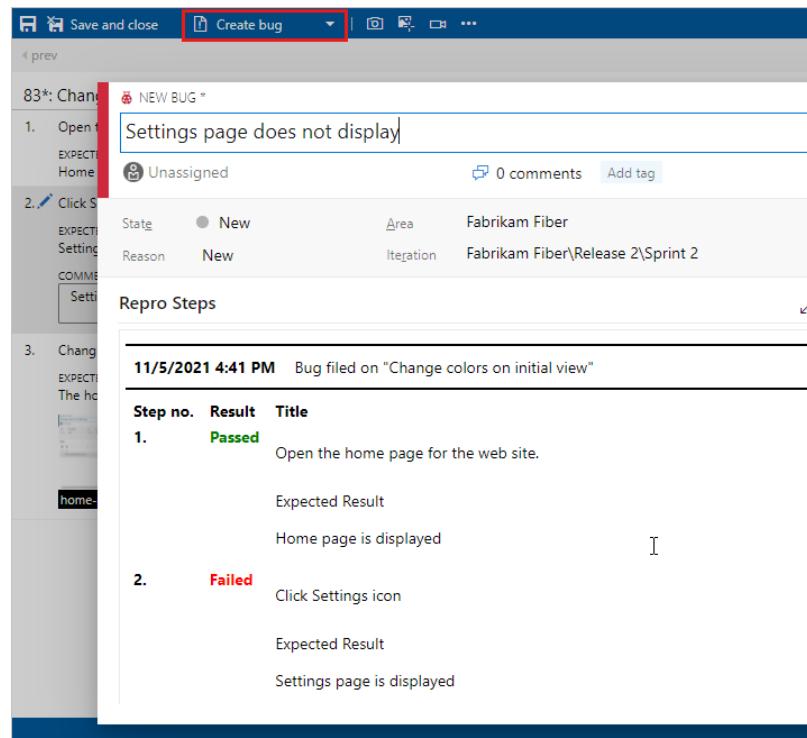
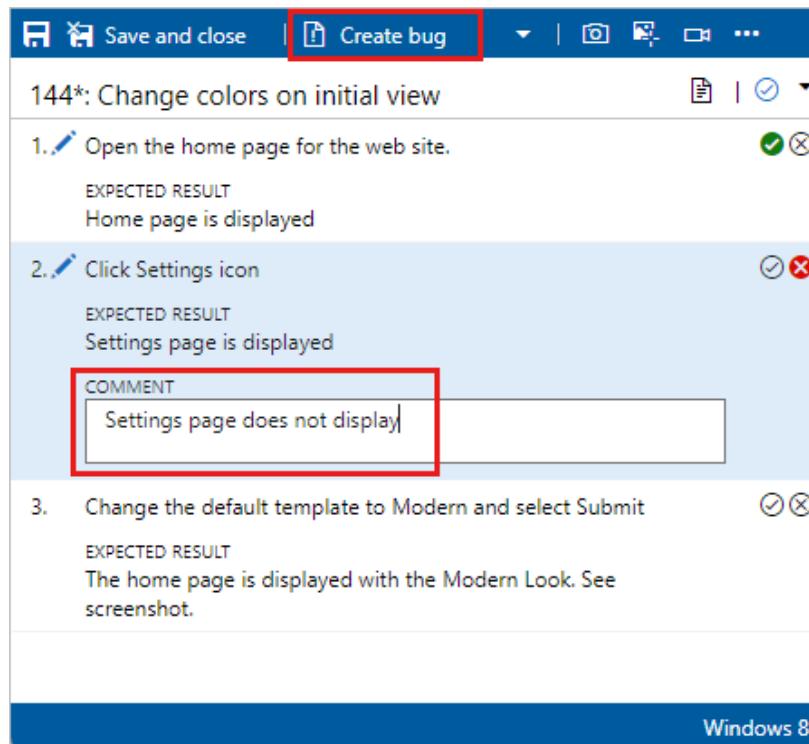
3. Change the default template to Modern and select Submit
- EXPECTED RESULT  
The home page is displayed with the Modern Look. See screenshot.



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 several others. Below the toolbar is a list of steps for a specific test case:

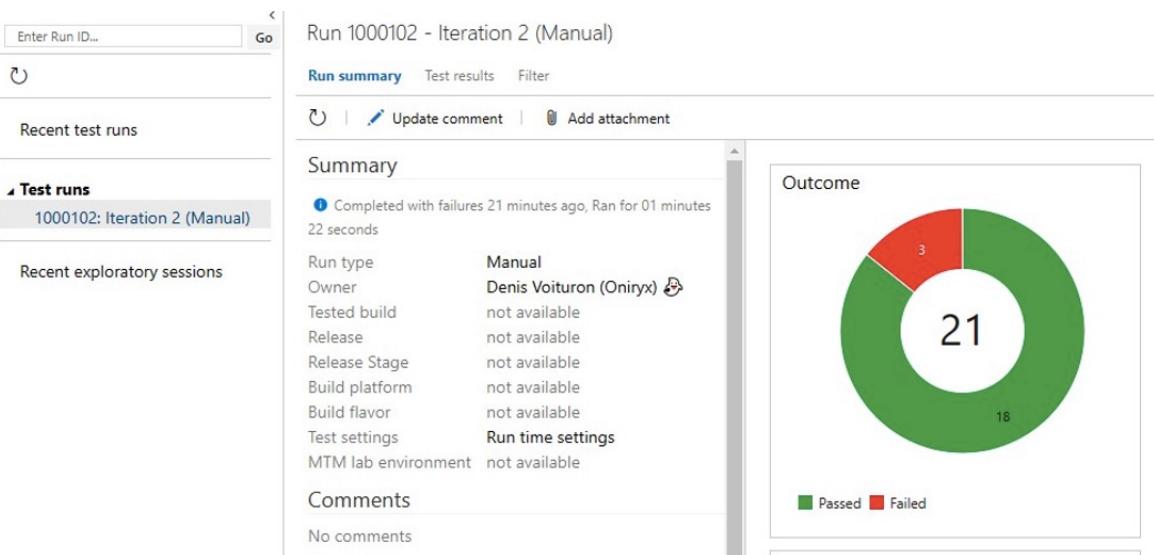
Step	Action	Status
1.	Open the home page for the web site.	✓✗
2.	Click Settings icon.	✓✗
3.	Select Use previous version	✓✗

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

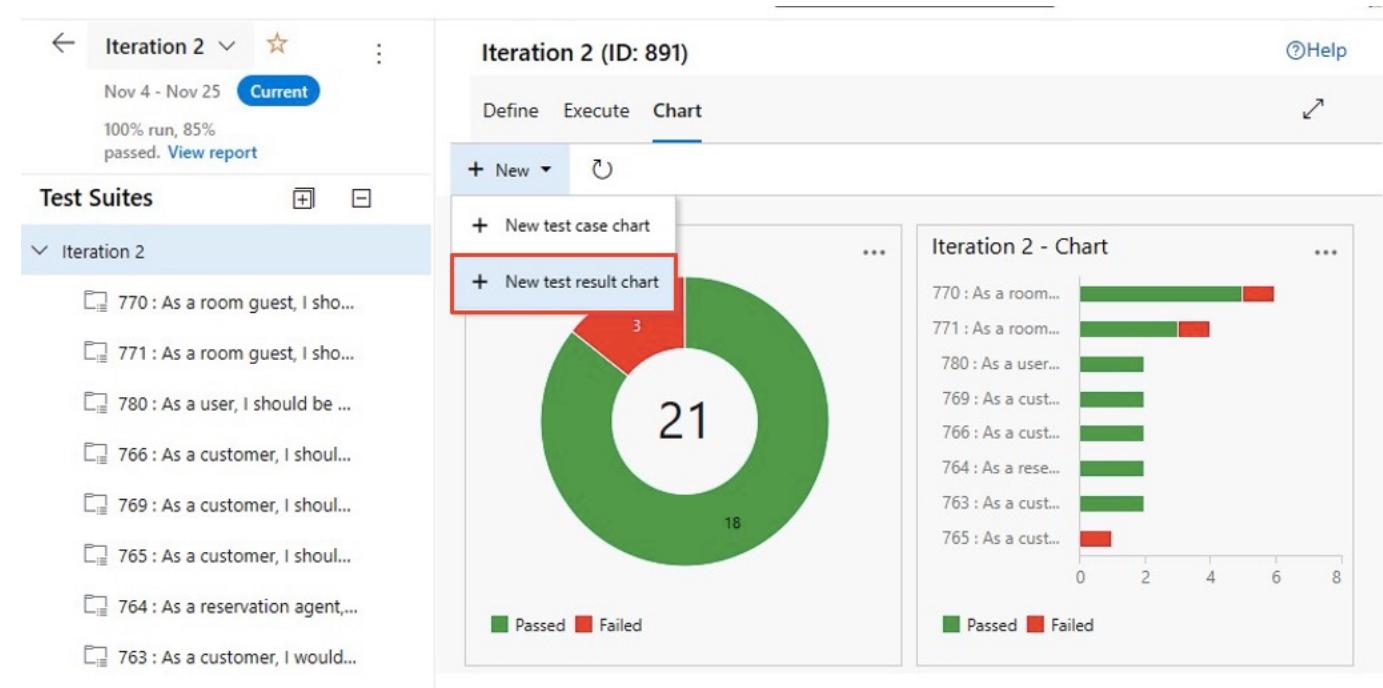
Test runs

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 Artifacts



Azure  
Artifacts

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

# Azure Artifacts

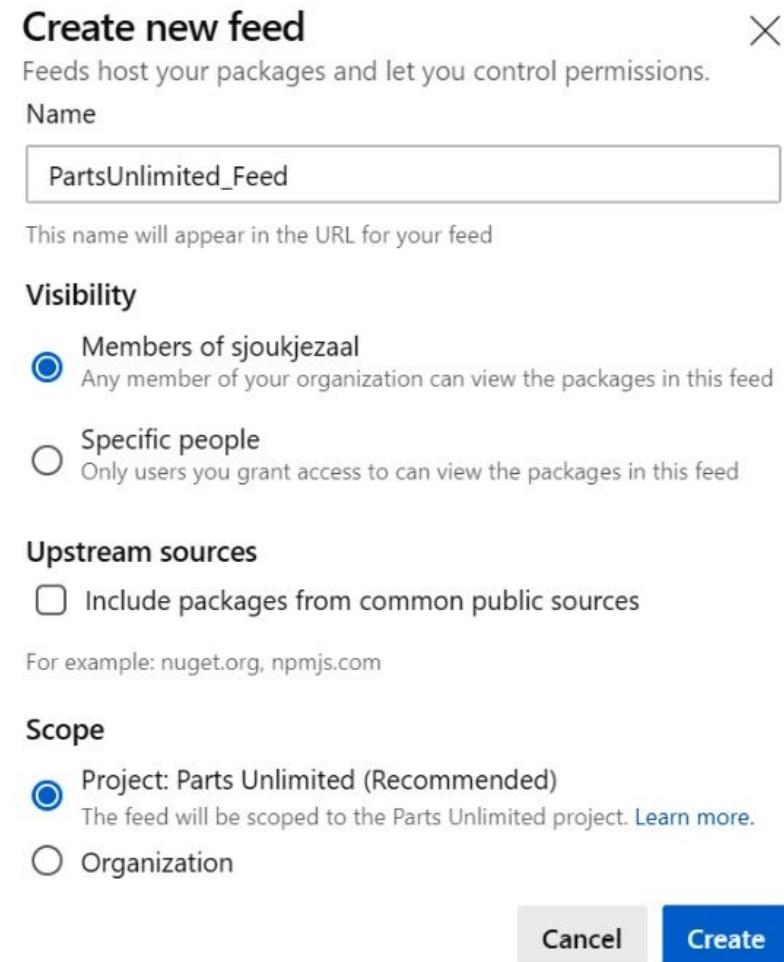
- Il est probable que chaque développeur ait utilisé un package tiers ou open source dans son code pour ajouter des fonctionnalités supplémentaires et accélérer le processus de développement de son application.
- L'utilisation de composants populaires pré-construits qui ont été utilisés et testés par 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 ef rentiel 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 ef rentiel dans la liste d roulante PartsUnlimited.
  - 3- Entrez l'URL du r ef rentiel source dans la zone URL de clonage et ajoutez un nom pour votre nouveau r ef rentiel GitHub
  - 4- Cliquez sur Importer.
- Apr s avoir import  le projet PartsUnlimited.Models dans un r ef rentiel Azure DevOps, nous pouvons l'utiliser dans un pipeline de build pour en cr er un package NuGet.

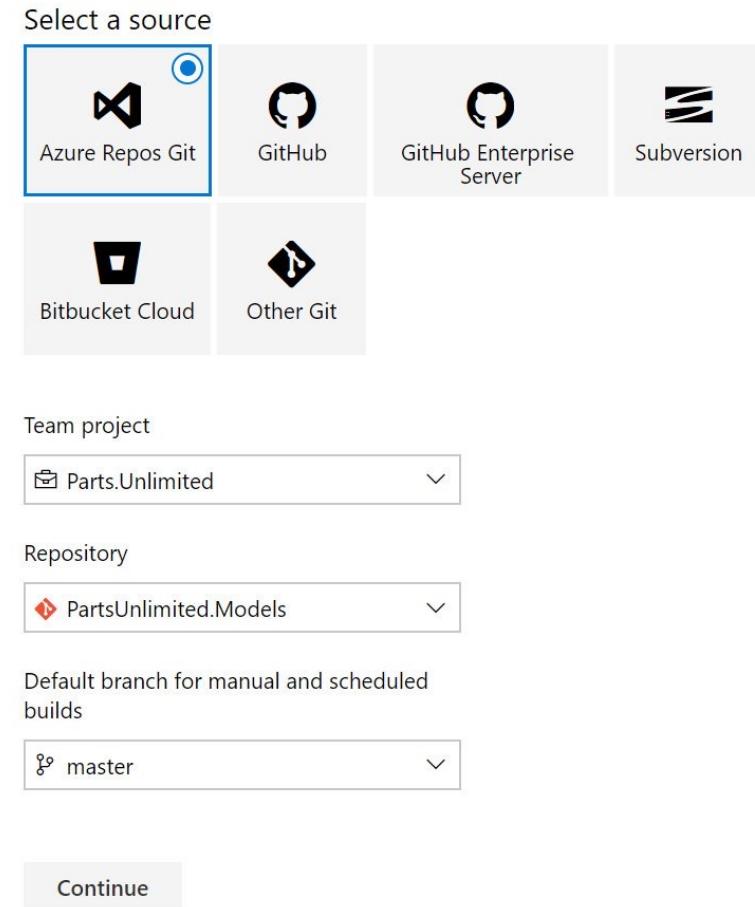
# Azure Artifacts – Cr ation d'un package avec une pipeline

- 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.

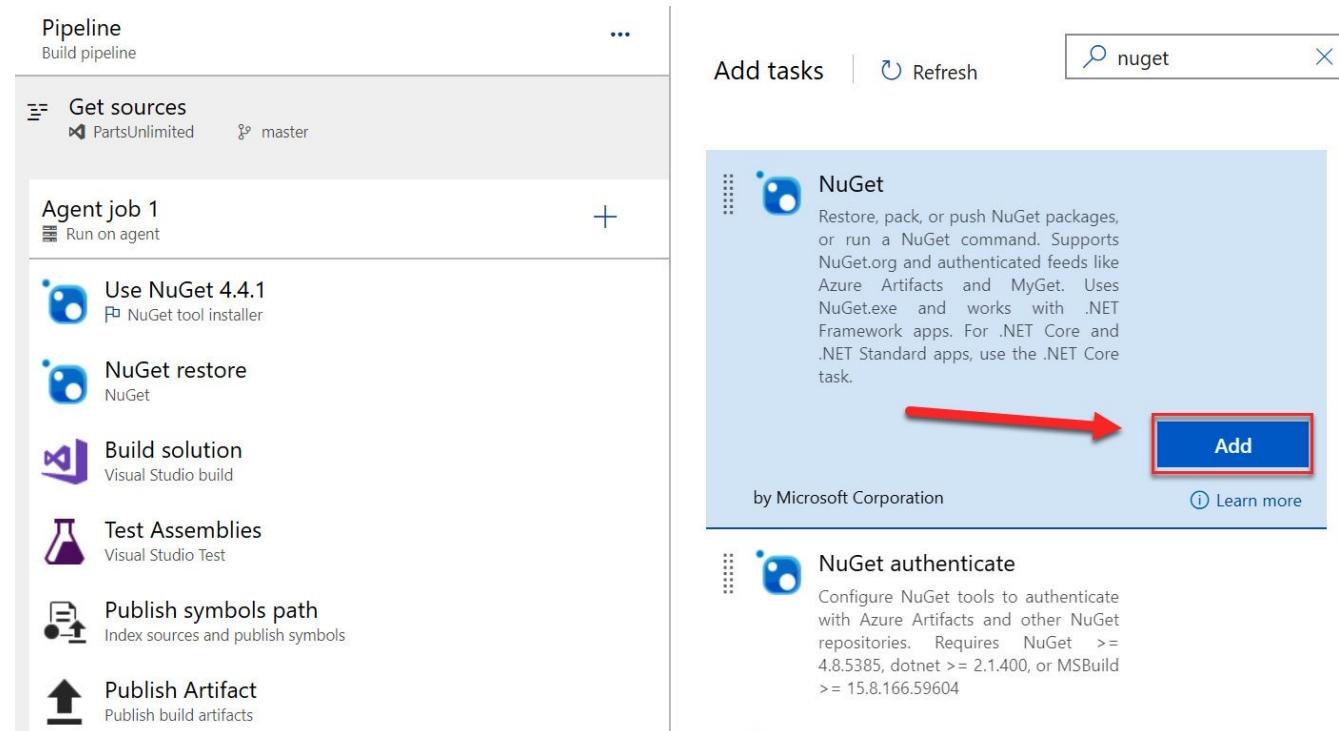
2- Cliquez sur Nouveau pipeline dans le menu en haut   droite et s lectionnez Utiliser l diteur classique sur le premier  cran de l'assistant.



# 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.

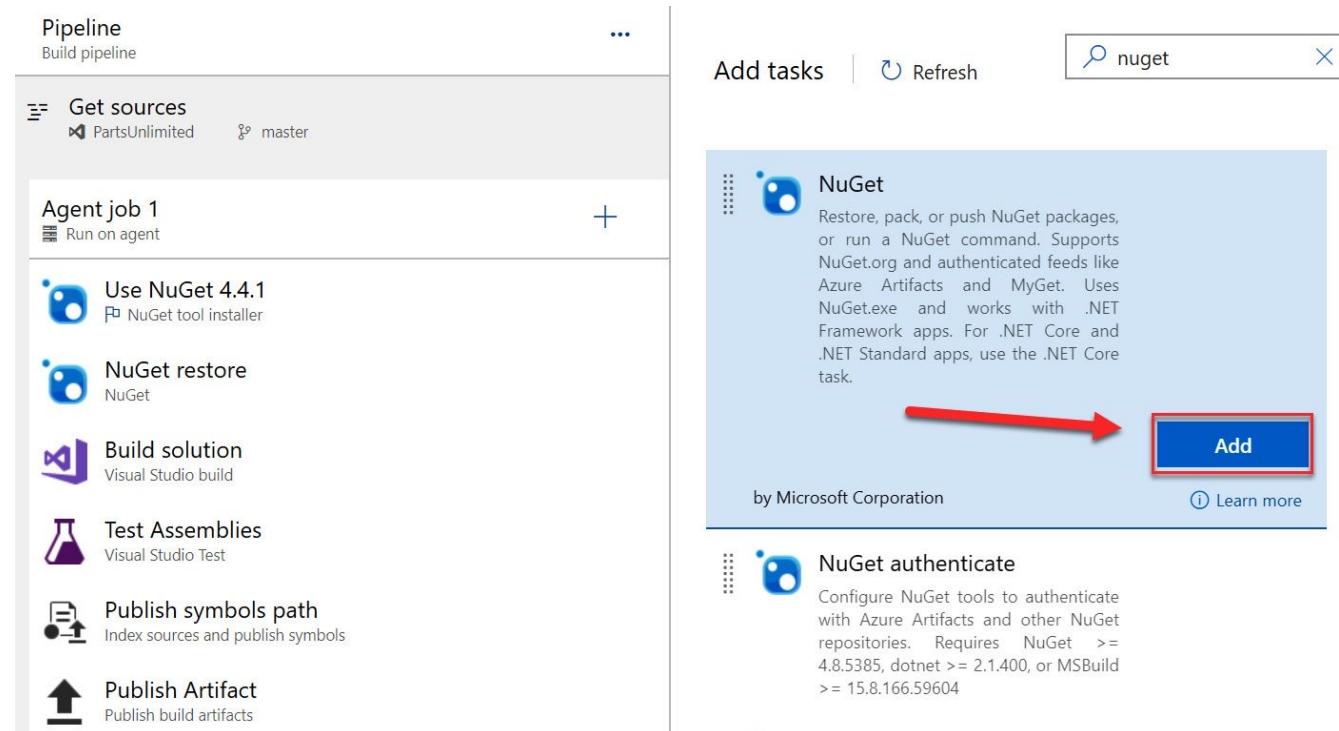
4- Cliquez sur le signe +   droite de la t che d'agent 1 et recherchez NuGet. Ajouter la t che NuGet au pipeline



# Azure Artifacts – Crédit d'un package avec une pipeline

3- Sélectionnez ASP.NET sur l'écran suivant de l'assistant et cliquez sur Appliquer.

4- Cliquez sur le signe + à droite de la tâche d'agent 1 et recherchez NuGet. Ajouter la tâche 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

# Azure DevOps – Déploiement

## 1. Approuver un déploiement

Dans Azure DevOps, le processus d'approbation des déploiements est crucial pour contrôler la qualité et la sécurité des releases avant qu'elles ne soient déployées dans les environnements cibles. Vous pouvez configurer des approbations manuelles à différents stades du pipeline de release :

- **Gates pré-déploiement** : Avant qu'un déploiement ne commence, vous pouvez exiger une ou plusieurs approbations manuelles. Ces approbations peuvent être requises de la part des membres de l'équipe spécifiques, garantissant ainsi que seuls les déploiements vérifiés et validés sont exécutés.
- **Configuration** : Dans Azure DevOps, configurez les approbations pré-déploiement en accédant à l'onglet "Pipelines" > "Releases", sélectionnez votre pipeline, puis modifiez le stage pour ajouter une approbation pré-déploiement dans l'onglet "Pré-déploiement".

# Azure DevOps – Déploiement

## 2. Automatisation des déploiements

L'automatisation des déploiements est au cœur d'Azure DevOps, permettant une mise en production rapide et fiable des applications :

- **Pipelines de CI/CD** : Créez des pipelines d'intégration continue (CI) pour automatiser la construction et le test de votre code dès qu'un commit est effectué. Puis, avec les pipelines de déploiement continu (CD), automatisez le déploiement de votre application dans différents environnements (développement, test, production).
- **Tâches et scripts** : Utilisez des tâches intégrées ou des scripts personnalisés dans vos pipelines pour exécuter des déploiements, gérer des configurations, et effectuer des tâches de post-déploiement.

# Azure DevOps – Déploiement

## 3. Déploiement conditionnel

Le déploiement conditionnel dans Azure DevOps permet de contrôler plus finement le processus de déploiement en définissant des conditions spécifiques pour l'exécution des déploiements :

- **Conditions d'environnement** : Définissez des conditions basées sur des variables d'environnement, des résultats de tâches précédentes, ou des expressions personnalisées pour déterminer si un stage de déploiement doit être exécuté.
- **Stratégies de déploiement** : Utilisez des stratégies comme le déploiement canari, blue/green, ou le déploiement par lots pour introduire la nouvelle version de l'application progressivement dans l'environnement de production, en minimisant les risques.

# Azure DevOps – Déploiement

## Configuration

Pour mettre en place ces fonctionnalités dans Azure DevOps :

- **Approbations** : Accédez à votre pipeline de release, sélectionnez le stage de déploiement, puis configurez les approbations et les contrôles dans les options de pré-déploiement.
- **Automatisation** : Créez et configurez des pipelines de CI/CD en utilisant YAML ou l'éditeur visuel pour automatiser la construction, les tests, et le déploiement de vos applications.
- **Déploiement conditionnel** : Dans les paramètres de chaque stage de votre pipeline de release, configurez les conditions sous lesquelles le stage doit être exécuté.