

Formateur : Ihab ABADI

Outils DevOps de Microsoft :



Azure Boards

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



Azure Repos

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



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 Test Plans

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



Azure Artifacts

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



Azure Boards

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

- Gestion des tâches
- Suivre
- Statut
- Planification

- Backlogs & Sprints
- Queries
- Plans aligned with backlogs

## Azure Repos



- source code
- Version control

- Files in the project
- TFVC or Git (commits)

## Azure Repos

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

- Suivre des changements → Pushes and Branches
- Contrôle → 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 secrets
- Customize the pipeline
- Machines (Windows or Linux)
- Old building model



Ihab ABADI - UTOPIOS



### 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

Ihab ABADI - UTOPIOS



### Azure Artifacts



- Création de nos propres packages

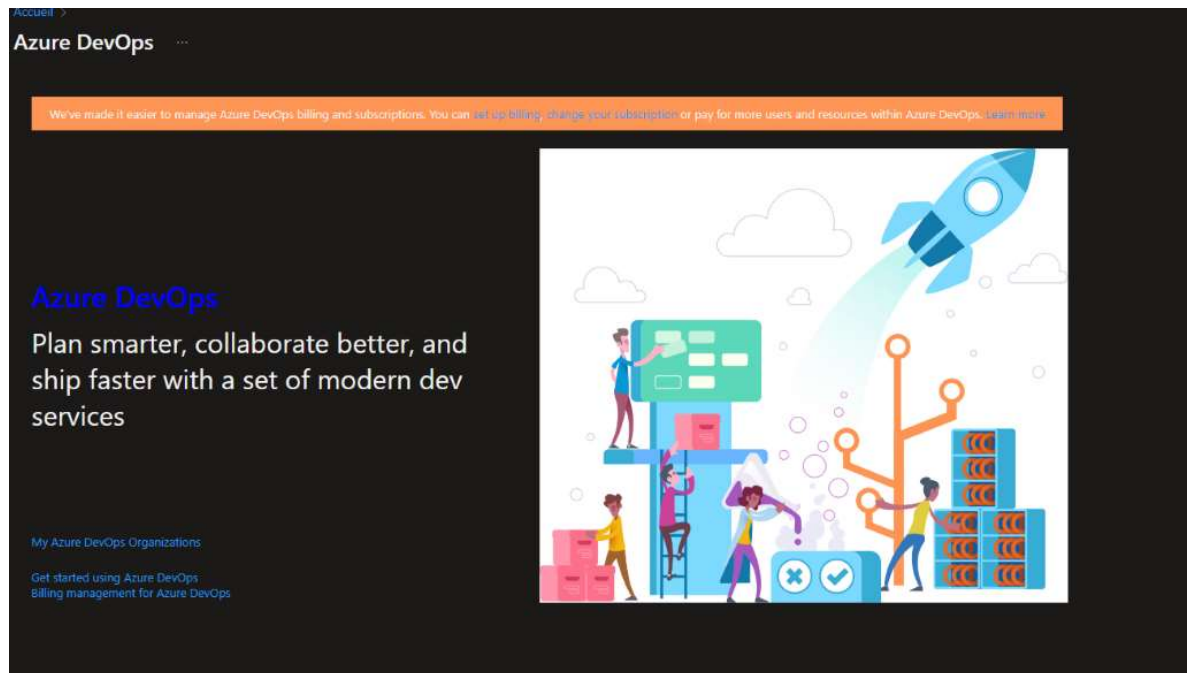
- NuGet

- .NET packages

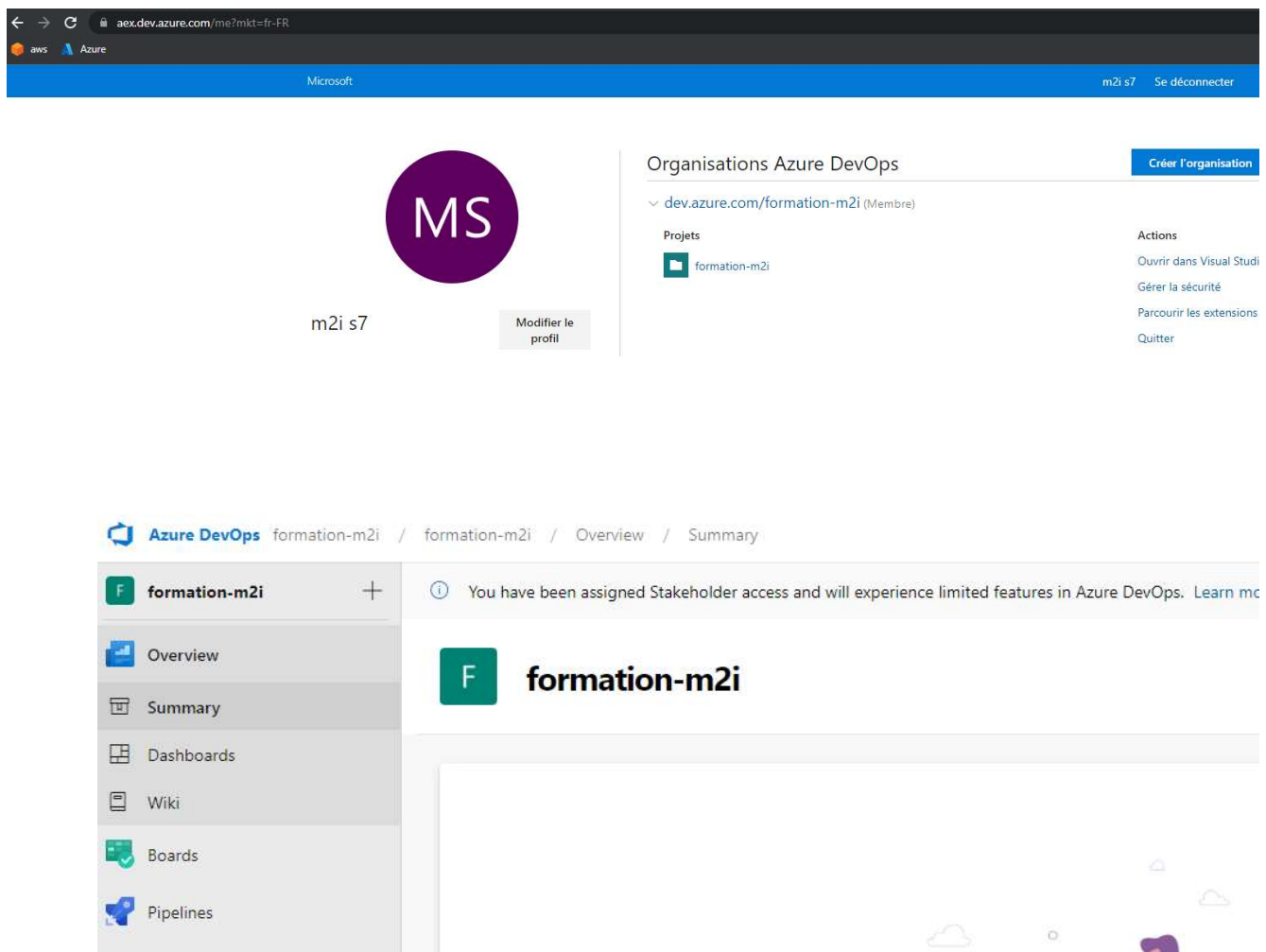
Create, host, and share packages. Easily add artifacts to CI/CD pipelines.

- npm
- Maven
- Gradle
- Universal

- JavaScript packages
- Java packages
- Java packages
- different packages



La plateforme azure devops se trouve à l'extérieur d'azure : <https://aex.dev.azure.com/>





**Welcome to the project!**

What service would you like to start with?

Azure – Pipeline :

Application qui permet le Build / Test / Deploy

## 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 continu

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

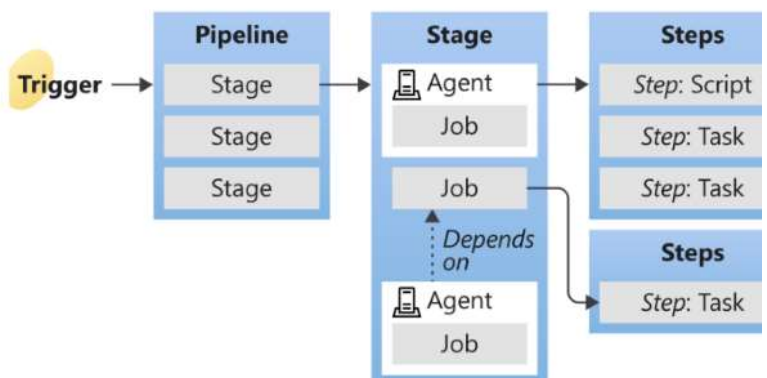


# Azure pipelines – Langages pris en charge

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

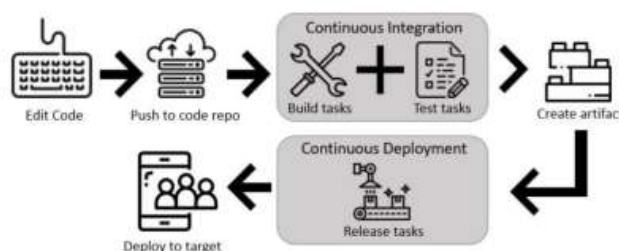
## Azure pipelines – fonctionnement

- 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.

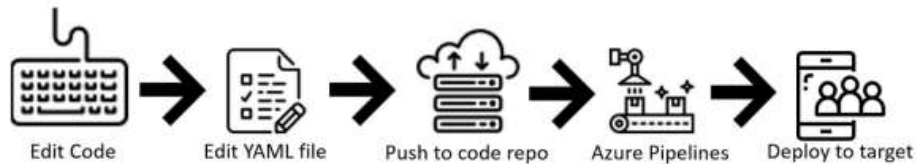


## Azure pipelines – utilisation

- 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



Création Pipeline :

Connect Select Configure Review

New pipeline

## Where is your code?



Azure Repos Git YAML  
Free private Git repositories, pull requests, and code search



Bitbucket Cloud YAML  
Hosted by Atlassian



GitHub YAML  
Home to the world's largest community of developers



GitHub Enterprise Server YAML  
The self-hosted version of GitHub Enterprise



Other Git  
Any generic Git repository



Subversion  
Centralized version control by Apache

[Use the classic editor](#) to create a pipeline without YAML.

✓ Connect ✓ Select **Configure** Review

New pipeline

## Configure your pipeline



Docker  
Build a Docker image



Docker  
Build and push an image to Azure Container Registry



Deploy to Azure Kubernetes Service  
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service



Deploy to Kubernetes - Review app with Azure DevSpaces  
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Services and setup Review App with Azure DevSpaces



Node.js  
Build a general Node.js project with npm.

# Docker

Build a Docker image

## Dockerfile

\$(Build.SourcesDirectory)/result/Dockerfile

formation-m2i / azure-pipelines.yml \*

```

1  # Docker
2  # Build a Docker image
3  # https://docs.microsoft.com/azure/devops/pipelines/languages/docker
4
5  trigger:
6  - master
7
8  resources:
9  - repo: self
10
11  variables:
12  - tag: '$(Build.BuildId)'
13
14  stages:
15  - stage: Build
16    displayName: Build image
17    jobs:
18    - job: Build
19      displayName: Build
20      pool:
21        vmImage: ubuntu-latest
22      steps:
23        - task: Docker@2
24          displayName: Build an image
25          inputs:
26            command: build
27            dockerfile: '$(Build.SourcesDirectory)/result/Dockerfile'
28            tags: |
29              $(tag)
30

```

Azure Pipeline propose un assistant pour créer directement les bloc YAML :

## Tasks





## Azure App Service deploy

Deploy to Azure App Service a web, mobile, or AP...



## Azure Functions for container

Update a function app with a Docker container

```
Settings
5  - task: Docker@2
6    Inputs:
7      command: 'login'
8
```

Exécuter le YAML :

## Save and run

Saving will commit azure-pipelines.yml to the repository.

Commit message

Set up CI with Azure Pipelines

Optional extended description

Add an optional description...

- ☒ Commit directly to the master branch  
☐ Create a new branch for this commit

TP :

## Exercice 1

L'objectif de cet exercice est de créer un pipeline de déploiement sur Azure DevOps qui permettra de déployer une application Node.js dans un environnement Azure.

L'adresse du dépôt : <https://github.com/frankhokevinho/simple-express-app>

Import du repository :

## Import a Git repository

Repository type

Git

Clone URL \*

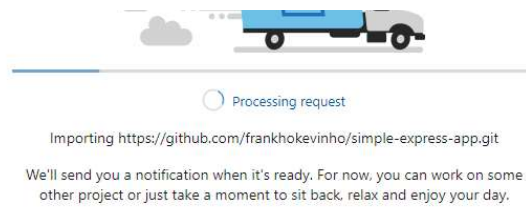
<https://github.com/frankhokevinho/simple-express-app.git>

☐ Requires Authentication

On its way!







Correction :

## Node.js Express Web App to Linux on Azure

Build a Node.js Express app and deploy it to Azure as a Linux web app.

Web App name

No results found

Project Settings  
formation-m2i

General

Overview

Teams

Permissions

Notifications

Service hooks

Dashboards

Boards

Project configuration

Team configuration

GitHub connections

Pipelines

Agent pools

Parallel jobs

Settings

Test management

Release retention

Service connections

XAML build services

Repos

Repositories

Service connections

Filter by keywords:

Git connection 1

New service connection

Choose a service or connection type

Search connection types

☒ Azure Classic

☐ Azure Repos/Team Foundation Server

☐ Azure Resource Manager

☐ Azure Service Bus

☐ Bitbucket Cloud

☐ Cargo

☐ Chef

☐ Docker Host

☐ Docker Registry

☐ Generic

☐ GitHub

☐ GitHub Enterprise Server

☐ Incoming WebHook

☐ Jenkins

Learn more

Next

## New Azure service connection

Azure Resource Manager using service principal

Scope level

- ☒ Subscription
- ☐ Management Group

Machine Learning Workspace

Subscription

Abonnement Azure 1 (c49e632f-5dd4-4c37-)

Resource group

m2i-formation

Details

Service connection name

azure devops

Description (optional)

Security

☒ Grant access permission to all pipelines

[Learn more](#)

[Troubleshoot](#)

Code partagé par Ihab :

```
{
  "subscriptionId": "c49e632f-5dd4-4c37-b1a6-578c7faa36fd",
  "appId": "ac865e4d-92cd-43e9-a3ac-fb3e5dc483ef",
  "displayName": "service-principal-aks",
  "password": "Jv38Q~IKp-otYhUem.DVcL.UlsymJpW6CnWJccXO",
  "tenant": "5016feea-b6ad-4017-aa7b-03ae57220502"
}
```

correction-nodejs-express / azure-pipelines.yml \* ⌵

```
1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5
6 trigger:
7 - main
8
9 pool:
10   vmImage: ubuntu-latest
11
12 steps:
13 - script: echo Hello, world!
14   displayName: 'Run a one-line script'
15
16 - script: |
17   echo Add other tasks to build, test, and deploy your project.
18   echo See https://aka.ms/yaml
19   displayName: 'Run a multi-line script'
20
```

← Azure App Service deploy ⓘ

Connection type \* ⓘ

Azure Resource Manager

Azure subscription \* ⓘ

ihab-app

App Service type \* ⓘ

Web App on Linux

App Service name \* ⓘ

qualid-nodejs-webapp

☐ Deploy to Slot or App Service Environment ⓘ

Package or folder \* ⓘ

\$(System.DefaultWorkingDirectory)/\*\*/\*.zip

Runtime Stack ⓘ

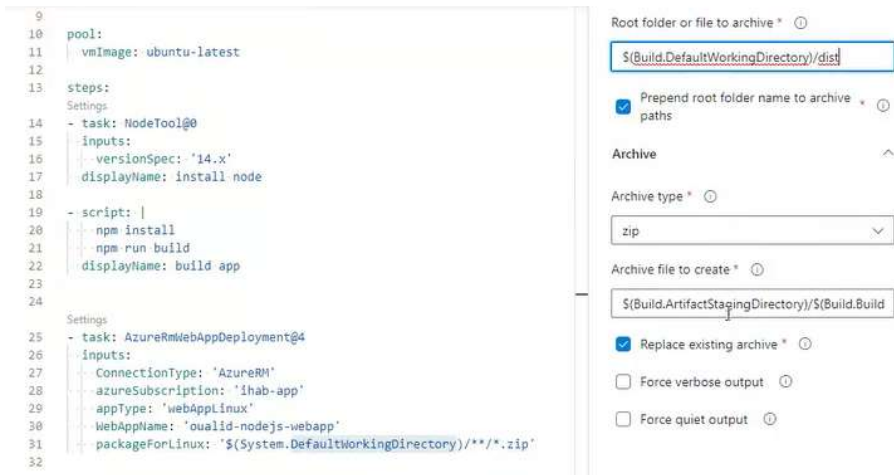
Review your pipeline YAML

Variables

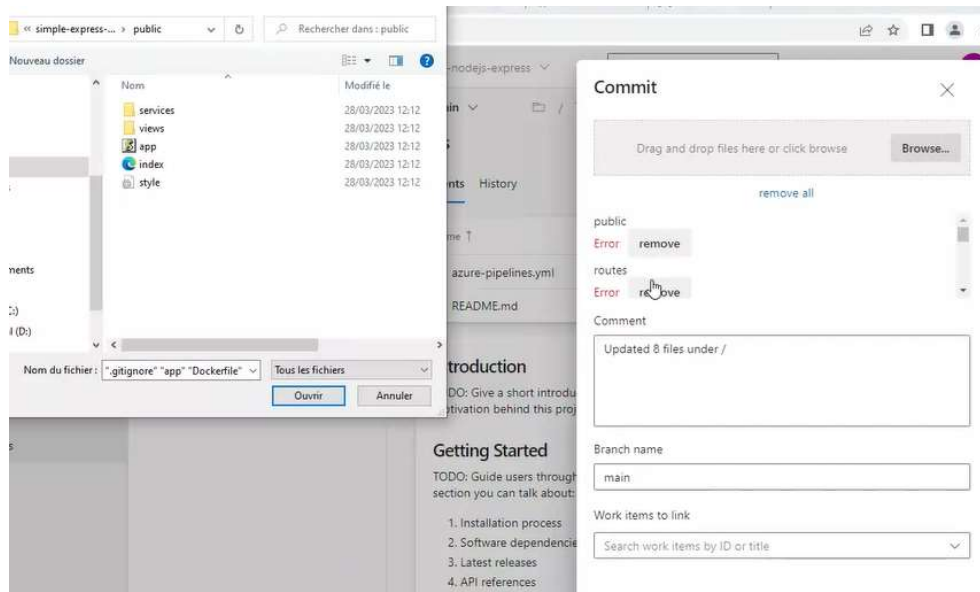
Save and run

correction-nodejs-express / azure-pipelines.yml \* ⌵

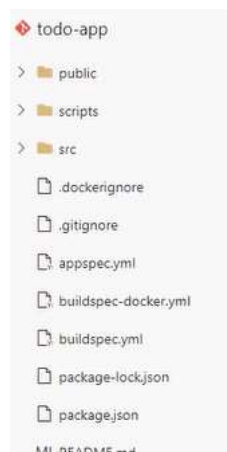
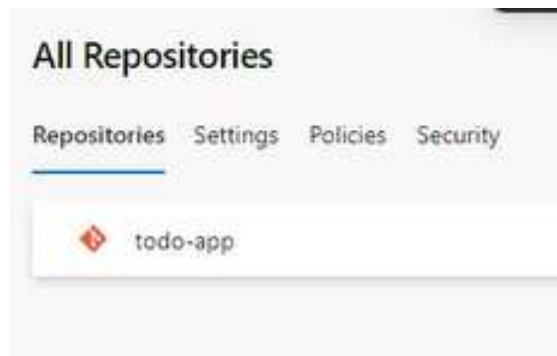
← Archive files ⓘ



Rajout du code dans le dossier du code pipeline :



Autre exemple avec un projet Angular :



Création Pipeline :

New pipeline

Review your pipeline YAML

Variables Save and run

todo-app / azure-pipelines.yml

```
1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy your code.
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5
6 trigger:
7   - main
8
9 pool:
10  vmImage: ubuntu-latest
11
12 steps:
13   - task: NodeTool@0
14     inputs:
15       versionSource: 'spec'
16       versionSpec: '14.x'
17
18   - script: |
19     npm install
20     npm build
21     displayName: 'Run a multi-line script'
```

Archive files

Root folder or file to archive \*

\$(Build.BinariesDirectory)

☒ Prepend root folder name to archive paths

Archive

Archive type \*

zip

Archive file to create \*

\$(Build.ArtifactStagingDirectory)/\$(Build.BuildId).zip

☒ Replace existing archive \*

☐ Force verbose output

☐ Force quiet output

```
# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
  - master

pool:
  vmImage: ubuntu-latest

steps:
  Settings
  - task: NodeTool@0
    inputs:
      versionSource: 'spec'
      versionSpec: '14.x'

  - script: |
    npm install
    npm build
    displayName: 'Run a multi-line script'

  Settings
  - task: ArchiveFiles@2
    inputs:
      rootFolderOrFile: '$(System.DefaultWorkingDirectory)/dist'
      includeRootFolder: true
      archiveType: 'zip'
      archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
      replaceExistingArchive: true
```

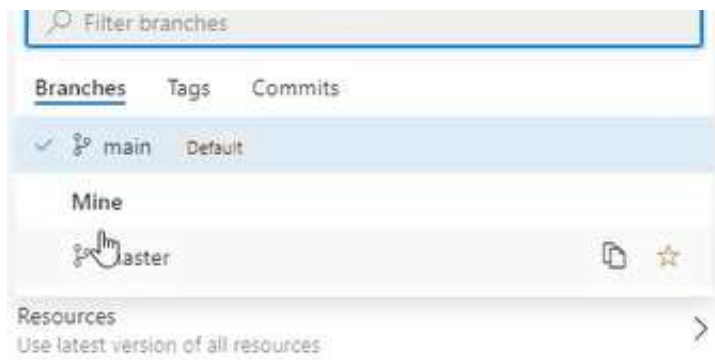
Run pipeline

Select parameters below and manually run the pipeline

Branch/tag

main

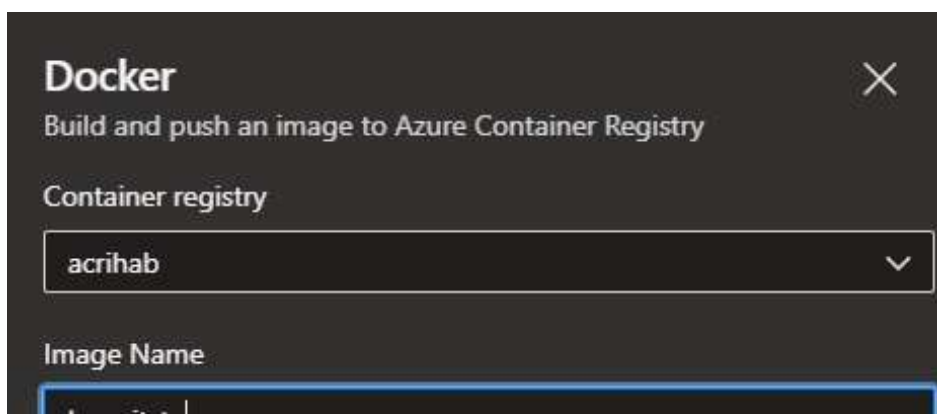
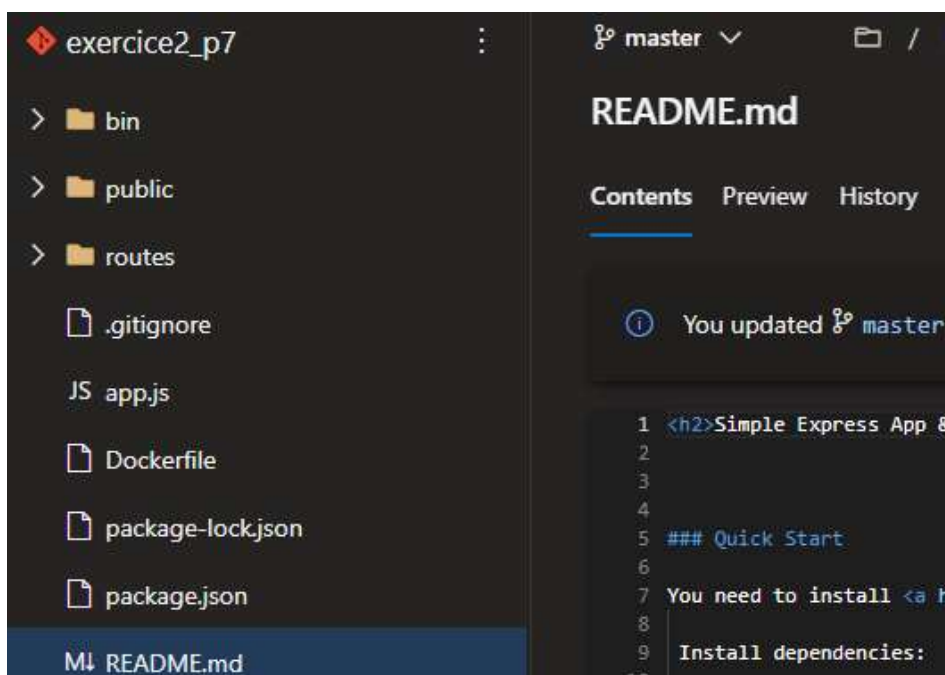


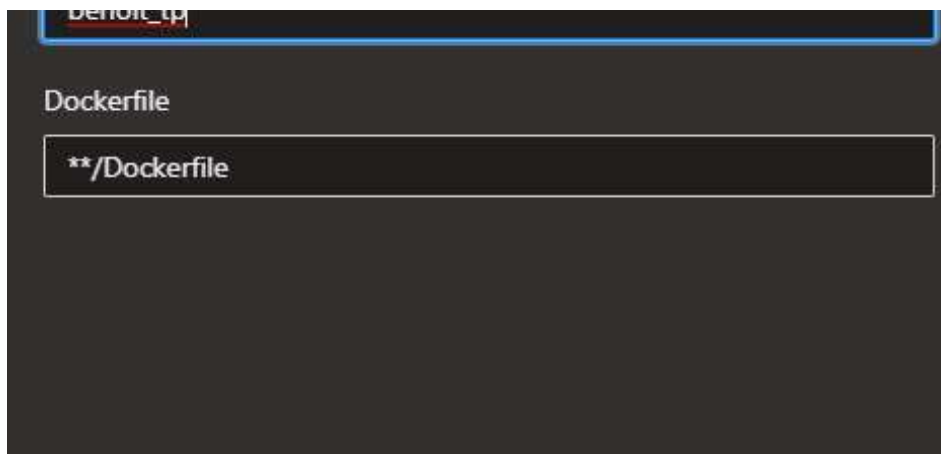


Exercice 2 :

Créer un nouveau repository + push le code

```
26 git clone https://github.com/frankhokevinho/simple-express-app.git
27 cd simple-express-app/
28 rm -rf .git
29 ls
30 git init
31 git remote add origin https://formation-m2i@dev.azure.com/formation-m2i/p7/_git/exercice2_p7
32 git status
33 git add .
34 git commit -m "tada !"
35 git push
36 git push -u origin master
```





Correction :

### Docker

Build and push an image to Azure Container Registry

Container registry

acrihab

Image Name

todoapp 1

Dockerfile

\$(Build.SourcesDirectory)/Dockerfile

Script généré grâce au Dockerfiles :

## Review your pipeline YAML

todo-app / azure-pipelines-2.yml

```
12 # Container registry service connection established during pipeline creat
13 dockerRegistryServiceConnection: '14596151-1c6b-4be1-a3ba-504454619adc'
14 imageRepository: 'todoapp'
15 containerRegistry: 'acrihab.azurecr.io'
16 dockerfilePath: '$(Build.SourcesDirectory)/Dockerfile'
17 tag: '$(Build.BuildId)'
18
19 # Agent VM image name
20 vmImageName: 'ubuntu-latest'
21
22 stages:
23 - stage: Build
24   displayName: Build and push stage
25   jobs:
26   - job: Build
27     displayName: Build
28     pool:
29       vmImage: $(vmImageName)
```

```

30 | steps:
    |   Settings
31 |   - task: Docker@2
32 |     displayName: Build and push an image to container registry
33 |     inputs:
34 |       command: buildAndPush
35 |       repository: $(imageRepository)
36 |       dockerfile: $(dockerfilePath)
37 |       containerRegistry: $(dockerRegistryServiceConnection)
38 |       tags: |
39 |         $(tag)
40 |
41 |

```

```

# Docker
# Build and push an image to Azure Container Registry
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

```

```

trigger:
- main

```

```

resources:
- repo: self

```

```

variables:
# Container registry service connection established during pipeline creation
dockerRegistryServiceConnection: '14596151-1c6b-4be1-a3ba-504454619adc'
imageRepository: 'todoapp'
containerRegistry: 'acrihab.azurecr.io'
dockerfilePath: '$(Build.SourcesDirectory)/Dockerfile'
tag: '$(Build.BuildId)'

```

```

# Agent VM image name
vmImageName: 'ubuntu-latest'

```

```

stages:
- stage: Build
  displayName: Build and push stage
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: $(vmImageName)
    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        command: buildAndPush
        repository: $(imageRepository)
        dockerfile: $(dockerfilePath)
        containerRegistry: $(dockerRegistryServiceConnection)
        tags: |
          $(tag)

    - task: AzureCLI@2
      inputs:
        azureSubscription: 'ihab-app'
        scriptType: 'bash'
        scriptLocation: 'inlineScript'
        inlineScript: 'az container create -g m2i-formation --name todo-app --image todoapp:199 --cpu 1 --memory 1 --registry-login-sec
registry-username "acrihab" --registry-password "gFGm0VE+JvvZBebx6K7uUGkqDXuLMKMOhj9lJuzFD5+ACRB25Or9" --ports 80 --ip-address Public

```

Shell

Script Location \* ⓘ

Inline script

### Inline Script \* ⓘ

```
az container create -g m2i-formation --name  
todo-app --image todoapp:197 --cpu 1 --  
memory 1 --registry-login-server  
"acrihab.azurecr.io" --registry-username  
"acrihab" --registry-password  
"gFGm0VE+JvvZBebx6K7uUGkqDXuLMKMO  
hj9lJuzFD5+ACRB25Or9" --ports 80 --ip-  
address Public
```

### Script Arguments ⓘ

### Advanced ^

☐ Access service principal details in script ⓘ

☐ Use global Azure CLI configuration ⓘ

### Working Directory ⓘ

☐ Fail on Standard Error ⓘ

```
Settings  
- task: AzureCLI@2  
  inputs:  
    azureSubscription: 'ihab-app'  
    scriptType: 'bash'  
    scriptLocation: 'inlineScript'  
    inlineScript: 'az container create -g m2i-formation --name
```

```
- task: AzureCLI@2  
  inputs:  
    azureSubscription: 'ihab-app'  
    scriptType: 'bash'  
    scriptLocation: 'inlineScript'  
    inlineScript: 'az container create -g m2i-formation --name todo-app --image todoapp:199 --cpu 1 --memory 1 --registry-login-s  
registry-username "acrihab" --registry-password "gFGm0VE+JvvZBebx6K7uUGkqDXuLMKMOhj9lJuzFD5+ACRB25Or9" --ports 80 --ip-address Public
```

On peut créer des dépendances entre les job (condition)

```
1 # Starter pipeline  
2 # Start with a minimal pipeline that you can customize to build and de  
3 # Add steps that build, run tests, deploy, and more:  
4 # https://aka.ms/yaml  
5  
6 trigger:
```



```

7   - main
8
9   pool:
10    vmImage: ubuntu-latest
11
12  jobs:
13    - job: job1
14      steps:
15        - script: echo "je suis le job 1"
16    - job: job2
17      dependsOn: job1
18      #condition: succeeded()
19      condition: failed()
20      steps:
21        - script: echo "je suis le job 2"

```

Ici, l'option `dependsOn` permet de faire démarrer un job uniquement si le job précédent a démarré

On peut aller encore plus loin avec l'option `condition`

```

# Starter pipeline
# Start with a minimal pipeline that you can customize to build and dep
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- main

pool:
  vmImage: ubuntu-latest

jobs:
  - job: job1
    steps:
      - script: exit 1
  - job: job2
    dependsOn: job1
    #condition: succeeded()
    condition: and(failed(), eq(variables['Build.SourceBranch'], 'refs/
    steps:
      - script: echo "je suis le job 2"

```

main ▾ todo-app / azure-pipelines.yml \*

```

1 Starter pipeline
2 Start with a minimal pipeline that you can customize to build and dep
3 Add steps that build, run tests, deploy, and more:
4 https://aka.ms/yaml
5
6 trigger:
7   main
8
9 pool:
10  vmImage: ubuntu-latest
11
12 jobs:
13   - job: job1
14     steps:
15       - script: |
16         exit 1
17         echo "##vso[task.setvariable variable=result;isOutput=true]1"
18   - job: job2
19     dependsOn: job1
20     #condition: succeeded()
21     #condition: and(failed(), eq(variables['Build.SourceBranch'], 'refs/
22     condition: eq(dependencies.job1.outputs.result, 1)

```

```

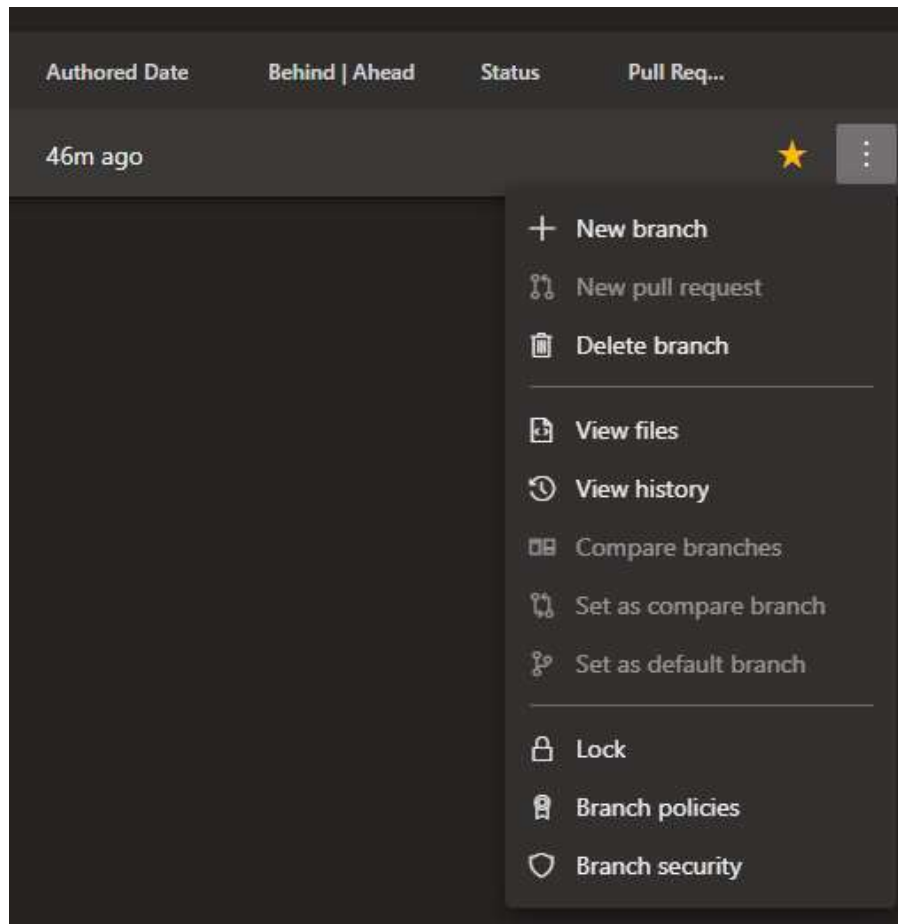
23 | steps:
24 |   - script: echo "je suis le job 2"

```

Si création de deux branche (main/master) à la création du projet -> voici des éléments de résolutions

- Tu renomes ta branche : `git branch -m master main`
- Si ton repo à distance est aussi sur une branche nommée master, change directement sur la plateforme (il y a un bouton défilant avec master, main...)
- Puis `git push -u origin main` / puis : `git push origin --delete master`

Changer la branche par défaut dans le repository/menu branche/set a default branche



Correction exercice :

On déclare directement nos variables dans le pipeline

```

todo-app / azure-pipelines.yml
1  # Starter pipeline
2  # Start with a minimal pipeline that you can customize to build and deploy your code.
3  # Add steps that build, run tests, deploy, and more:
4  # https://aka.ms/yaml
5
6  trigger:
7  - main
8
9  pool:
10 | vmImage: ubuntu-latest
11
12  variables:
13  - name: 'appName'
14  - value: 'todoapp'
15  - name: 'buildDirectory'
16  - value: '$(workingDirectory)'
17

```

Jobs:

Tâche : installation Nodejs

```
jobs:
- job: build
  steps:
    Settings
    - task: NodeTool@0
      inputs:
        versionSpec: '14.x'
        displayName: Install nodejs
```

Tâche : installation des dépendances :

```
Settings
- task: Npm@1
  inputs:
    command: 'install'
    workingDir: $(buildDirectory)
    displayName: Install des dépendances
```

Script : build de l'application :

```
- script: |
  npm run build
  displayName: build app
```

Job test de l'application :

```
- job: test
  dependsOn: build
  condition: succeeded()
  # #condition: and(failed(), eq(variables['Build.SourceBranch'], 'refs/heads/master'))
  # condition: eq(dependencies.job1.outputs['job1step1.result'], 1)
  steps:
    - script: echo "je suis le job 2"
```

Job deploiement de l'application :

```
- job: deploy
  dependsOn: test
  steps:
    Settings
    - task: ArchiveFiles@2
      inputs:
        rootFolderOrFile: '$(buildDirectory)'
        includeRootFolder: true
        archiveType: 'zip'
        archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
        replaceExistingArchive: true
        displayName: Création du zip
```

Task :

```
Settings
- task: AzureRmWebAppDeployment@4
  inputs:
    ConnectionType: 'AzureRM'
    azureSubscription: 'ihab-app'
    appType: 'webAppLinux'
    WebAppName: '$(appName)'
    packageForLinux: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
    SlotName: 'production'
    condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/pull/$(System.PullRequest.PullRequestId)/merge'))
    displayName: deploy test
```

Code utilisé :

```
# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- main

pool:
  vmImage: ubuntu-latest

variables:
- name: 'appName'
  value: 'todoapp'
- name: 'buildDirectory'
  value: '$(workingDirectory)'

jobs:
- job: build
  steps:
  - task: NodeTool@0
    inputs:
      versionSpec: '14.x'
    displayName: Install nodejs

  - task: Npm@1
    inputs:
      command: 'install'
      workingDir: $(buildDirectory)
    displayName: Install des dépendances

  # - task: Npm@1
  #   inputs:
  #     command: 'custom'
  #     customCommand: 'run build'
  #     workingDir: $(buildDirectory)
  #     displayName: Install des dépendances

  - script: |
      npm run build
    displayName: build app

- job: test
  dependsOn: build
  condition: succeeded()
  # #condition: and(failed(), eq(variables['Build.SourceBranch'], 'refs/heads/master'))
  # condition: eq(dependencies.job1.outputs['job1step1.result'], 1)
  steps:
  - task: Npm@1
    inputs:
      command: 'custom'
      customCommand: 'run test'
      workingDir: $(buildDirectory)
    displayName: Install des dépendances

- job: deploy
  dependsOn: test
  steps:
  - task: ArchiveFiles@2
    inputs:
      rootFolderOrFile: '$(buildDirectory)/public'
      includeRootFolder: true
      archiveType: 'zip'
      archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
      replaceExistingArchive: true
    displayName: Création du zip

  - task: AzureRmWebAppDeployment@4
    inputs:
      ConnectionType: 'AzureRM'
```



```

    azureSubscription: 'ihab-app'
    appType: 'webAppLinux'
    WebAppName: '${appName}'
    packageForLinux: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
    SlotName: "production"
condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/main'))
displayName: deploy prod

- task: AzureRmWebAppDeployment@4
  inputs:
    ConnectionType: 'AzureRM'
    azureSubscription: 'ihab-app'
    appType: 'webAppLinux'
    WebAppName: '${appName}'
    packageForLinux: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
    SlotName: "production"
condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/pull/${System.PullRequest.PullRequestId}/merge'))
displayName: deploy test

```

## Exercice 4

Vous devez créer un pipeline Azure qui permet de construire et tester une image Docker, puis la déployer dans Azure Container Registry (ACR) et Azure Kubernetes Service (AKS). Le pipeline doit être configuré pour exécuter les étapes suivantes :

- Récupérer le code source de l'application à partir d'un référentiel Git.
- Construire une image Docker à partir de l'application.
- Exécuter les tests de l'application.
- Si les tests réussissent, déployer l'image Docker dans Azure Container Registry (ACR).
- Déployer l'image Docker depuis ACR dans Azure Kubernetes Service (AKS).

Assurez-vous que le pipeline ne déploie l'image Docker que si toutes les étapes précédentes ont réussi.

Correction :

Deux possibilité pour les test : build l'image et faire les test / faire les test et déployer

Ne pas oublier la connexion des services : (ACR et AKS)

### New Kubernetes service connection

Authentication method

☐ KubeConfig
☐ Service Account
☒ Azure Subscription

Azure Subscription

Abonnement Azure 1 (c49e632f-5dd4-4c37-b1a6-578c7faa36f...

Cluster

hayet (m2i-formation)

Namespace

default

☐ Use cluster admin credentials

Details

Service connection name

aks-ihab

Description (optional)

Security

☒ Grant access permission to all pipelines

Learn more
Troubleshoot

Setting up connection...

```

# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- main

pool:
  vmImage: ubuntu-latest

variables:
- name: 'appName'
  value: 'todoapp'
- name: 'buildDirectory'
  value: '$(workingDirectory)'

jobs:
- job: build
  steps:
  - task: Docker@2
    inputs:
      containerRegistry: 'acrihab'
      repository: 'image-app'
      command: 'buildAndPush'
      Dockerfile: '**/Dockerfile'
      displayName: build image app

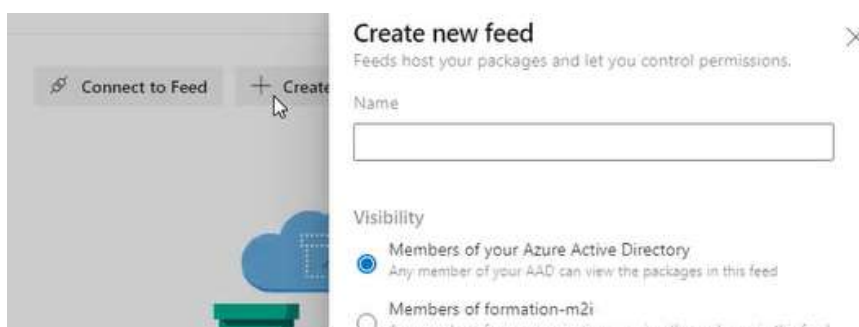
- job: test
  dependsOn: build
  condition: succeeded()
  # condition: and(failed(), eq(variables['Build.SourceBranch'], 'refs/heads/master'))
  # condition: eq(dependencies.job1.outputs['job1step1.result'], 1)
  steps:
  - task: Docker@2
    inputs:
      containerRegistry: 'acrihab'
      command: 'start'
      arguments: 'npm run test'
      Dockerfile: '**/Dockerfile'

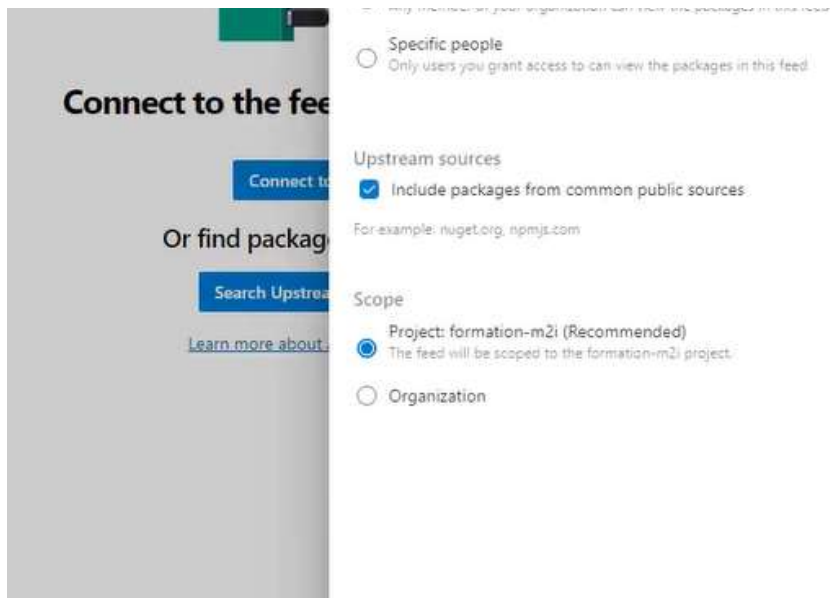
- job: deploy
  dependsOn: test
  steps:
  - task: KubernetesManifest@0
    inputs:
      action: 'deploy'
      kubernetesServiceConnection: 'ihab-aks'
      manifests: 'deploy.yml'

```

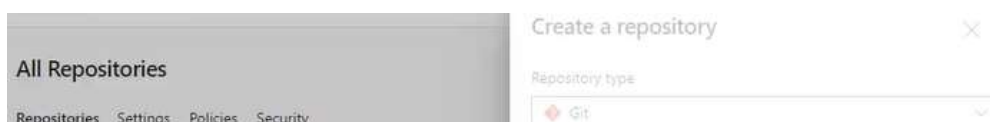
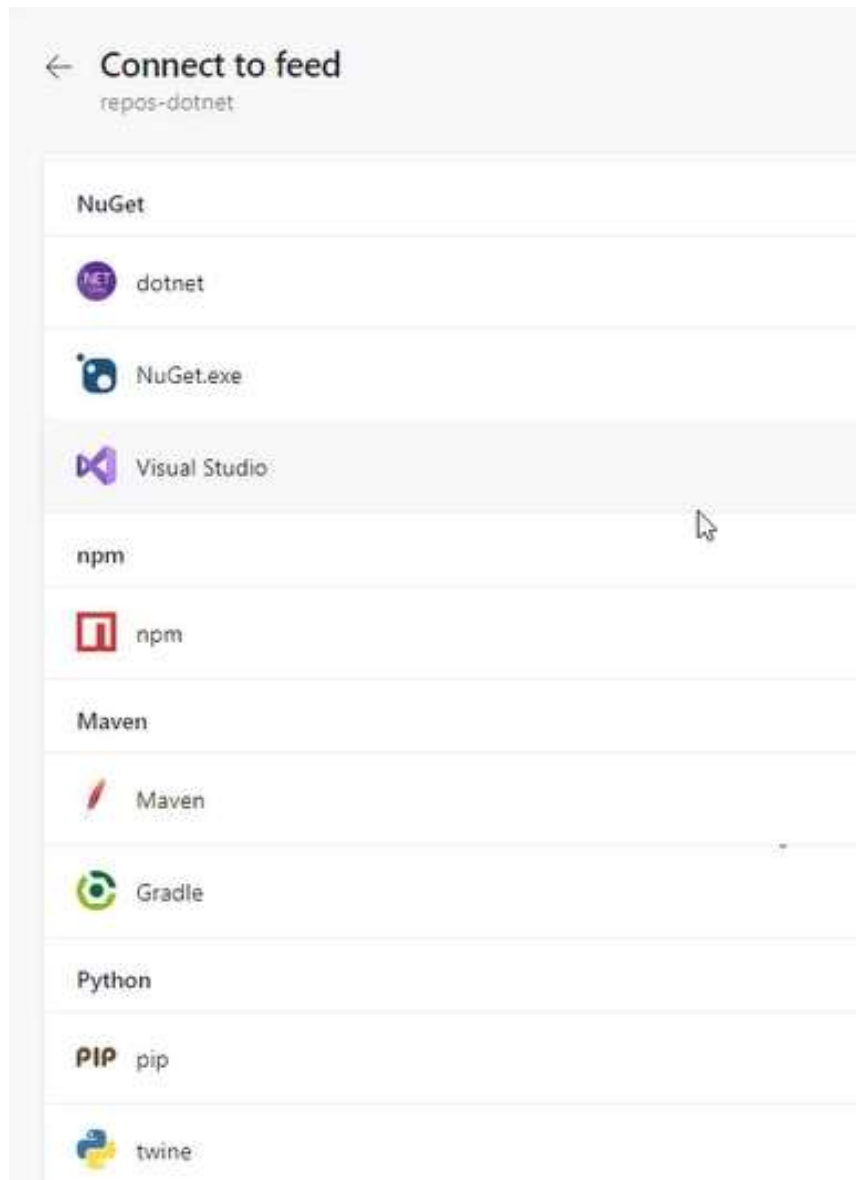
Déployer Pipeline à Artifact :

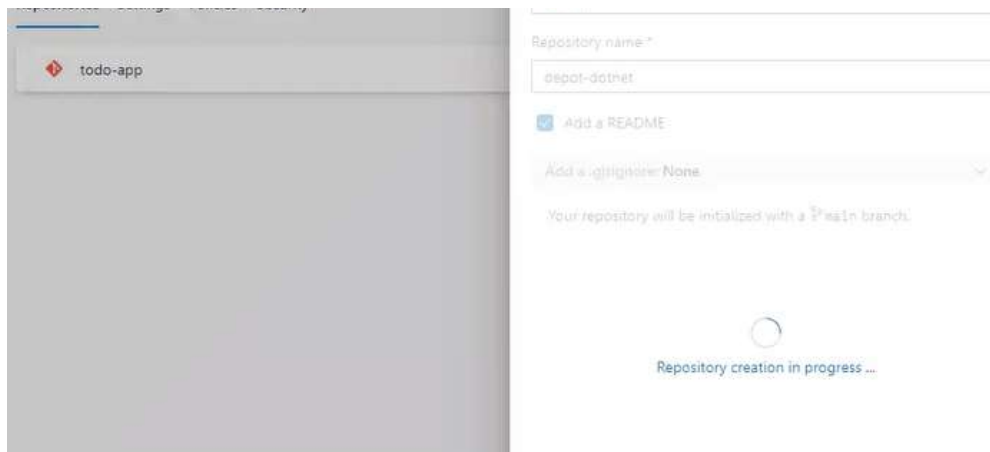
Création d'un feed :



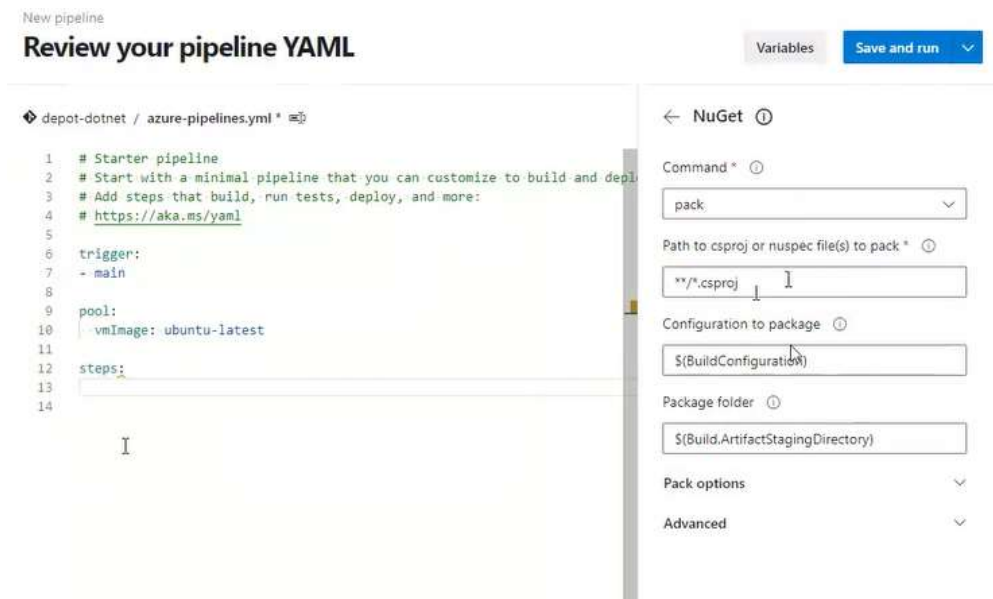


Sélection techno'





Review du pipeline :

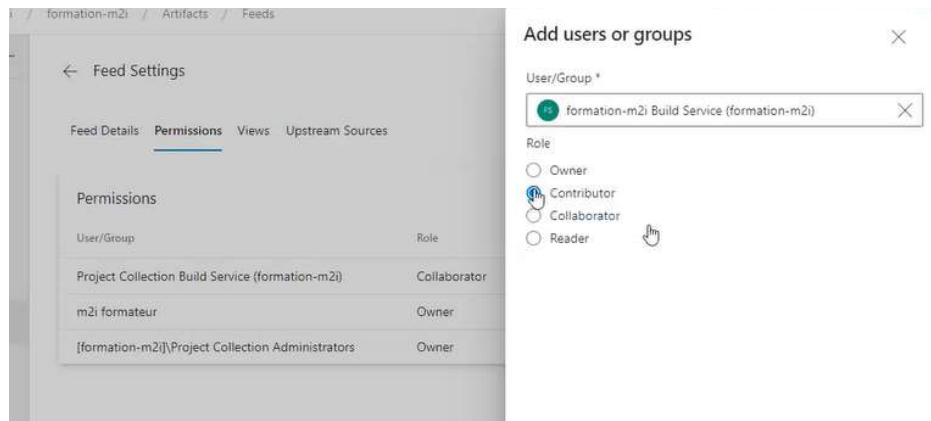


Run du job :



Ne pas oublier de rajouter les droits sur le pipeline :





TP :

## La Partie CI/CD

- Vous aurez besoin de mettre en place les services azure devops suivants :
  - Azure repos
  - Azure pipeline
- A chaque fois qu'un commit sera pushé sur la branche main sur azure repos, azure pipeline se chargera de générer une nouvelle image. Cette nouvelle image sera stockée dans ACR.
- Elle viendra ensuite être mise à jour dans votre cluster.
- A chaque fois qu'une branche est mise à jour, le chef de projet recevra une notification.
- Il faudra partir du principe que chacun des microservices sera géré par une équipe différente (Developpeur).
- Le projet est pleinement fonctionnel lorsque vous allez effectuer une modification sur le titre de la page du projet dans le header en passant « Online Boutique » à « M2i Boutique ».
- Cette modification sera prise en compte de manière automatique par votre système juste après avoir mis à jour le micro-service sur azure repos.

