Easy Save

Introduction

livrable 1 contains the first functional version of EasySave, as specified by ProSoft.

This version initiates the development of a .NET Core console application for managing backup tasks.

Prerequisites

1. Installation process

- Clone the livrable-1 branch and open the project with Visual Studio 2022.
- Make sure you have installed .NET Core 6.0.

2. Software dependencies

- .NET Core 6.0
- Visual Studio 2022

Execute Easy Save

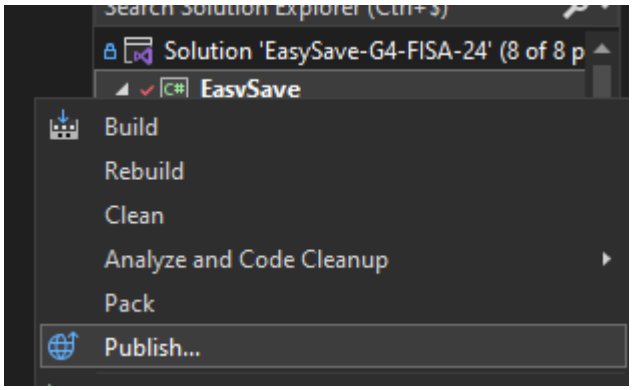
Windows

Run EasySave.exe in directory `../G4-FISA-24/EasySave-G4-FISA-24/EasySave/bin/Debug/net6.0/EasySave.exe`.

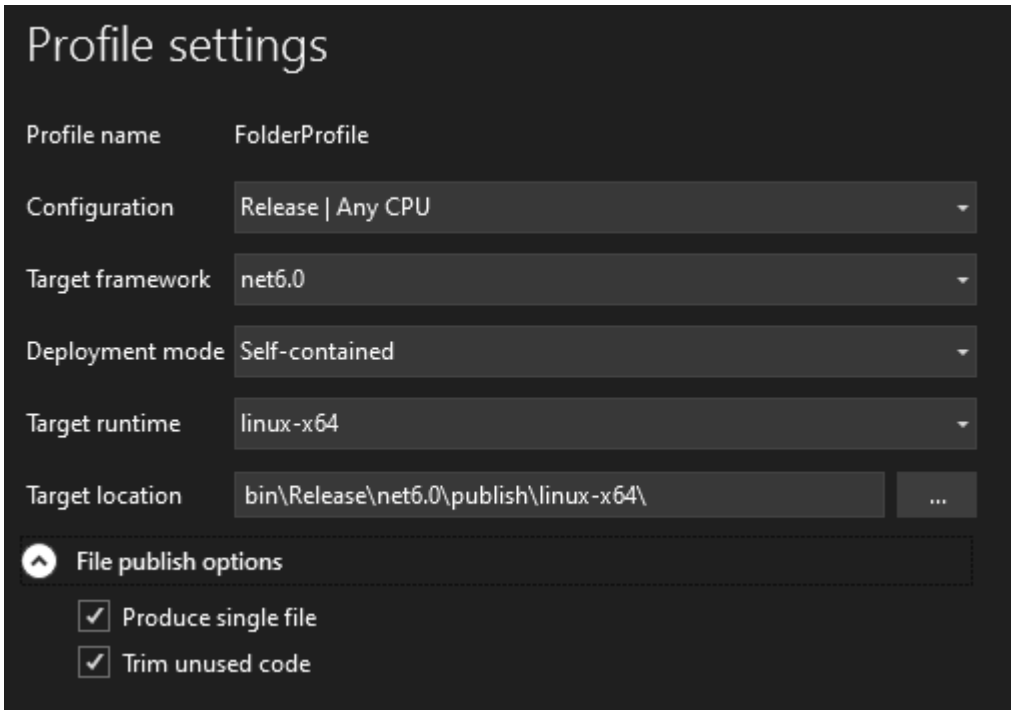
Linux

Run `./EasySave` after publishing from visual studio.

Publishing EasySave

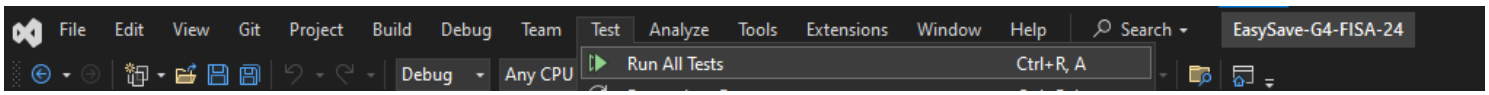


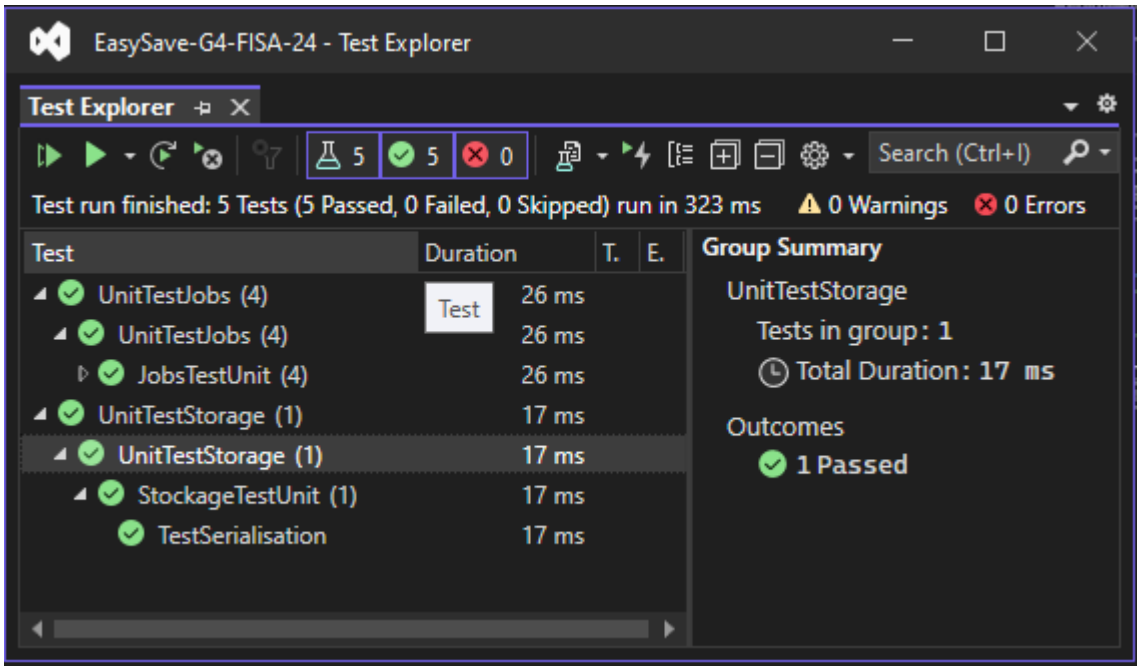
You need to specify the 'Target runtime' according to the operating system.



Tests

Open the Test Explorer in Visual Studio and run the unit tests.





All tests are tested during the CI/CD phase, but it's best to run the tests before pushing.

Latest versions

Please consult the "releases" tab to access the latest version of livrable 1 (Pipelines -> Releases):

https://dev.azure.com/faikmehmeti/G4-FISA-24/_release?_a=releases&view=mine&definitionId=1

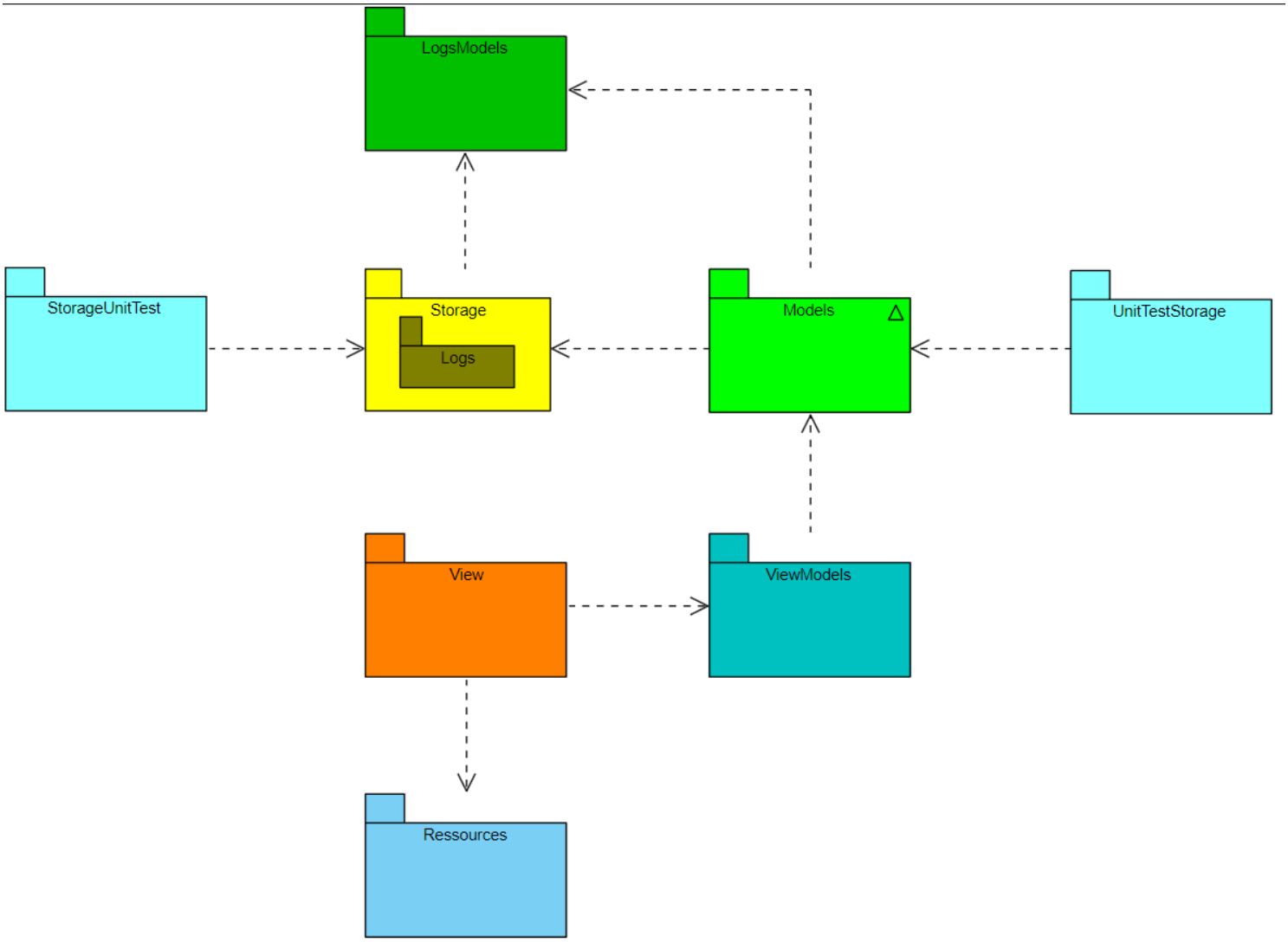
Features

- Sequential backup
- Daily logs
- Status logsv
- Select language
- List jobs
- Load job configuration
- Create a job
- Delete a job
- Launch jobs

Console application entry point

```
using View = EasySave.Views.View;
namespace EasySave // Note: actual namespace depends on the project name.
{
    /// <summary>
    /// Application entry point
    /// </summary>
    internal class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
            View pView = new View();
            //Lance le program principale
            pView.Run();
        }
    }
}
```

Architecture



The architecture is based on an MVVM model with :

- **Models** : classes representing data (CJob, CJobManager, CSettings, etc.)
- **Views**: classes representing views (BaseView, View, JobView, etc.).
- **ViewModels**: classes linking models and views.

Log System

Default location of CJobManager and CSettings and location of log model stored using the logging classes in the Logs package.

The current location of event logs is stored in the `Logs` folder.

The current location of user settings is stored in the root directory `Settings.json`.

The location of backup jobs is stored in the `Jobs` folder.

Logs with names formatted as `Logs - 2024-02-15` are daily logs.

The `Logs.json` file is the log status.

Continuous Integration (CI) and Continuous Deployment (CD) processes using Azure DevOps

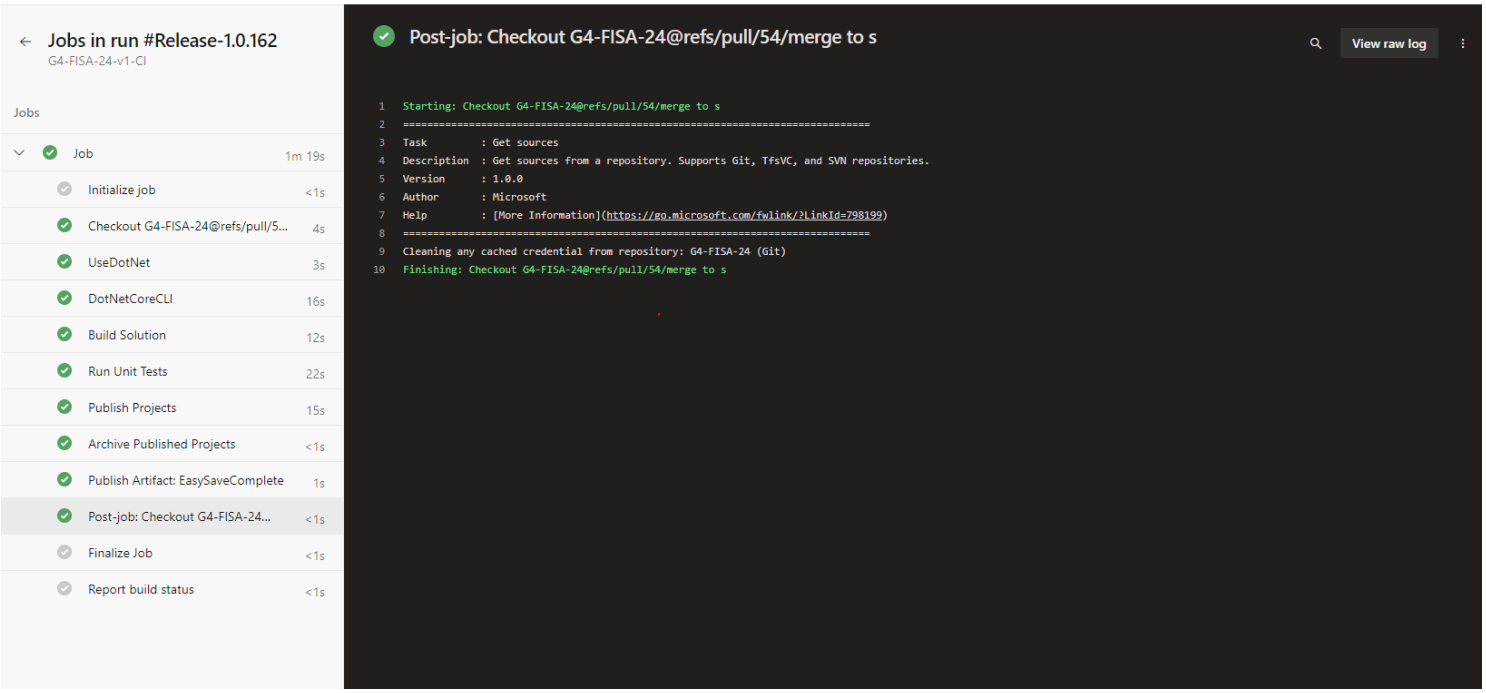
Continuous Integration configuration

Triggering pipelines

Continuous integration pipelines are configured to trigger automatically when a pull request is made to the main branch. Continuous integration validation is a prerequisite for any code merge.

Integration process

- Build: Compilation of source code into an executable or deployable package.
- Automated Testing: Execution of tests to verify code quality and functionality.
- Release Publishing: Creation and storage of release artifacts in Azure DevOps.

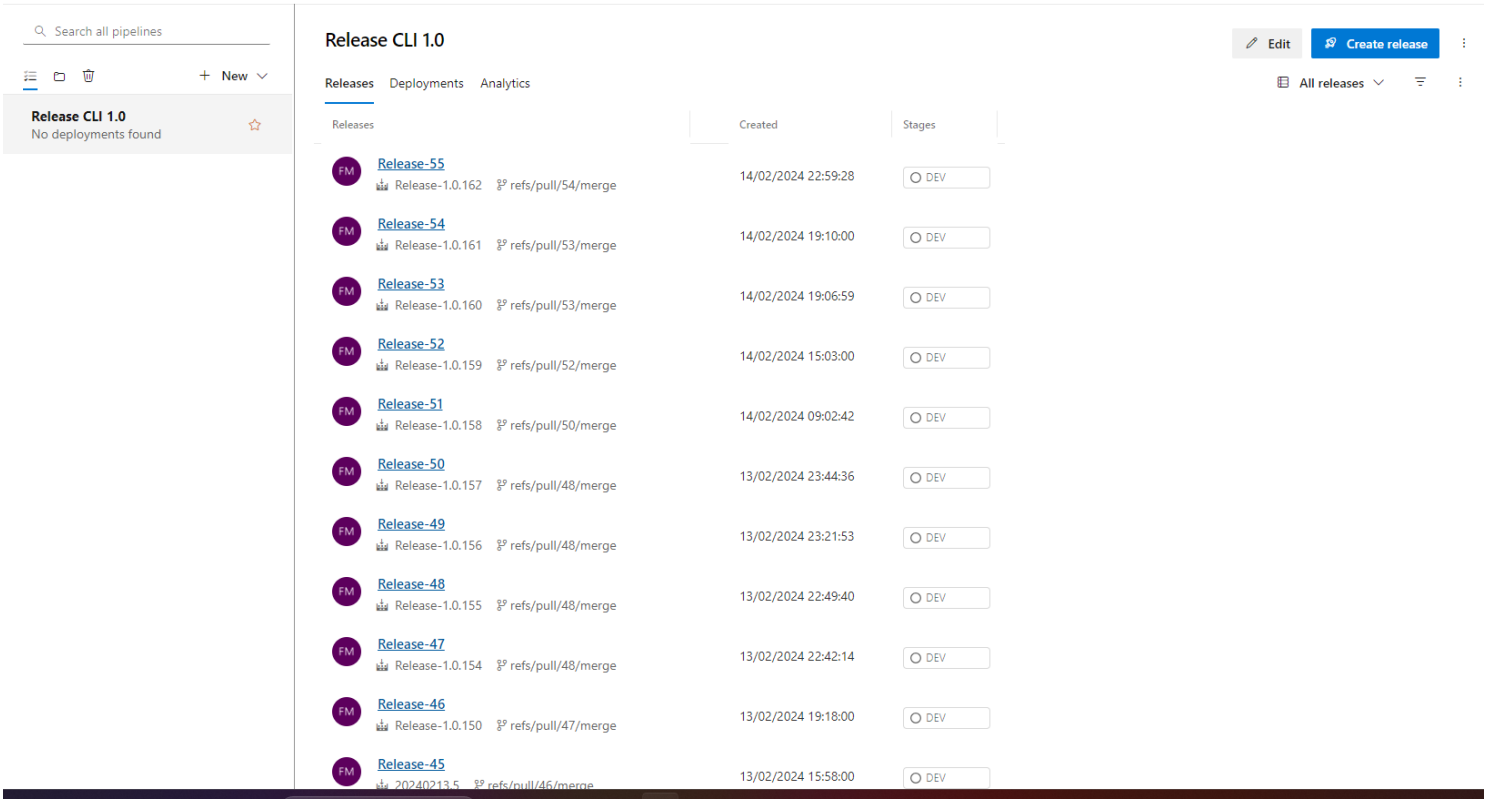


Continuous Deployment Management

Although continuous integration is automated, continuous deployment currently requires manual intervention.

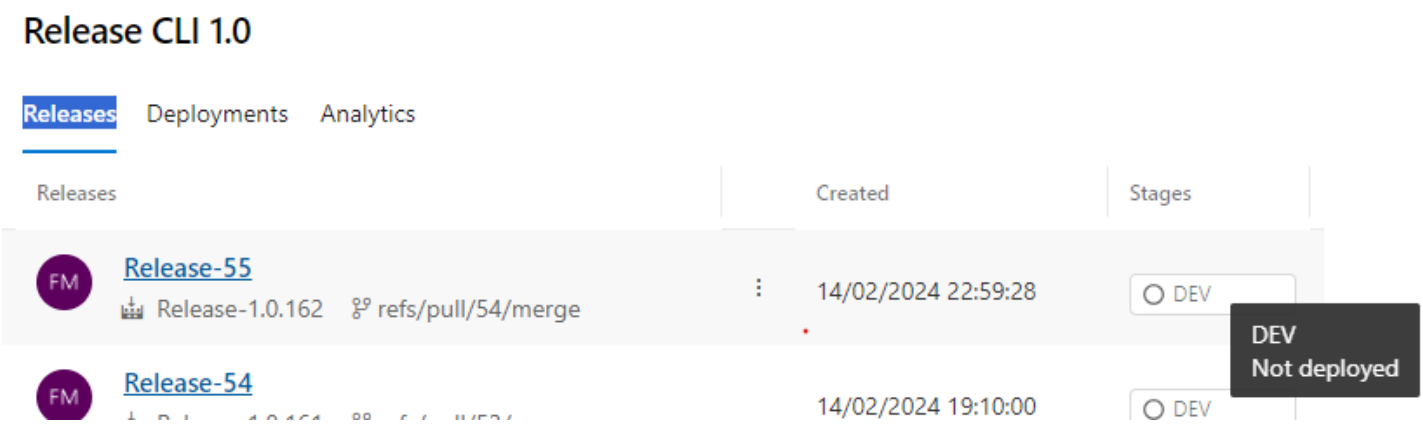
Access to Releases

https://dev.azure.com/faikmehmeti/G4-FISA-24/_release



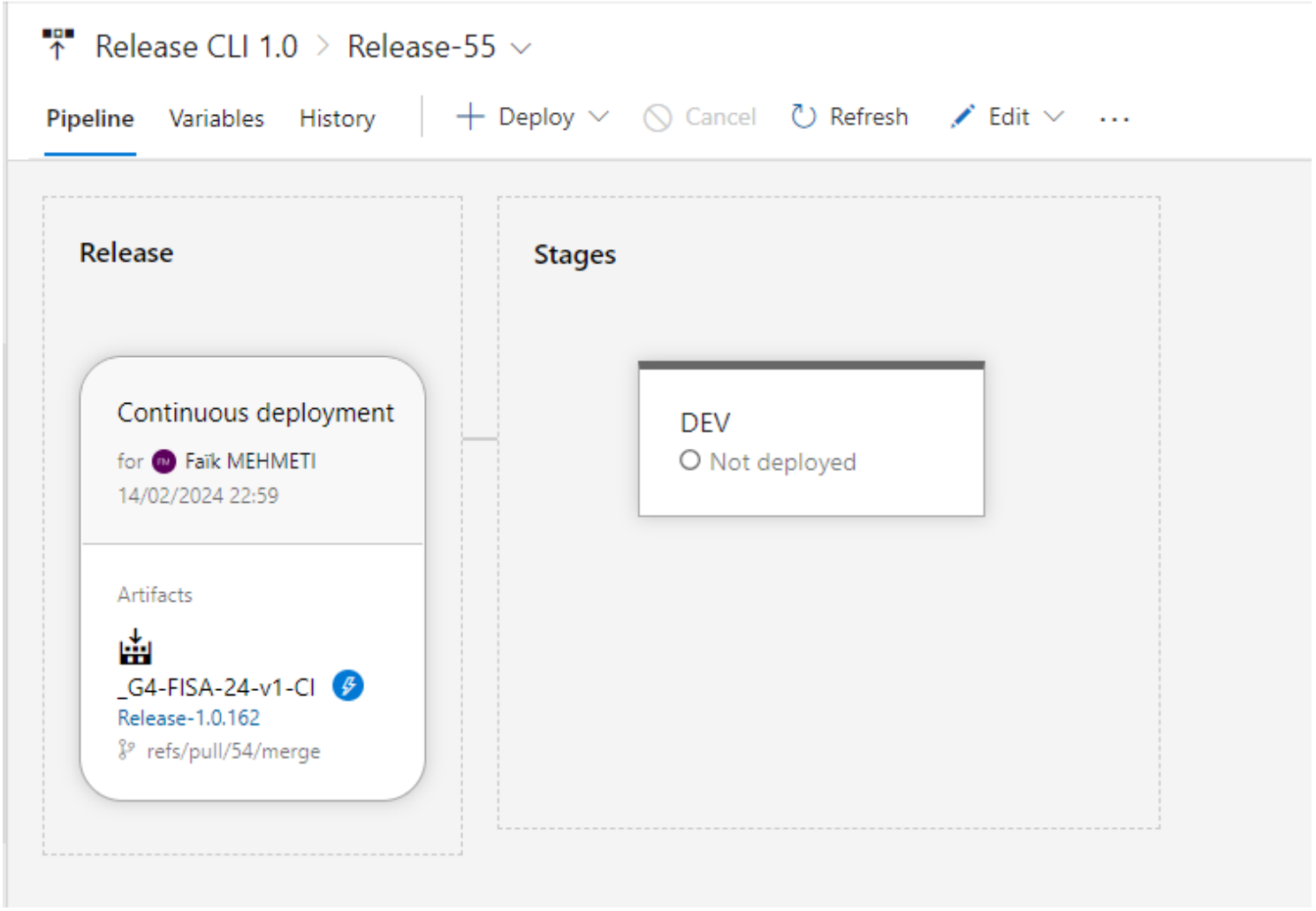
Here you can track the deployment status of each release.

This link allows you to follow the deployment status for each release.



Deployment Management

By selecting a release, you will be taken to a page where continuous deployment can be managed. Currently, deployments must be performed manually for each stage.


Release CLI 1.0
Release-55
Pipeline
Variables
History
Deploy
Cancel
Refresh
Edit
Release
Continuous deployment
for Faïk MEHMETI
14/02/2024 22:59
Artifacts
_G4-FISA-24-v1-CI
Release-1.0.162
refs/pull/54/merge
Stages
DEV
Not deployed

Easy Save

Introduction

Le livrable 1 contient la première version fonctionnelle d'EasySave, conformément aux spécifications de ProSoft.

Cette version initie le développement d'une application console .NET Core pour la gestion des tâches de sauvegarde.

Prérequis

1. Processus d'installation

- Cloner la branche livrable-1 et ouvrir le projet avec Visual Studio 2022.
- Assurez-vous d'avoir installé .NET Core 6.0.

2. Dépendances logicielles

- .NET Core 6.0
- Visual Studio 2022

Executer Easy Save

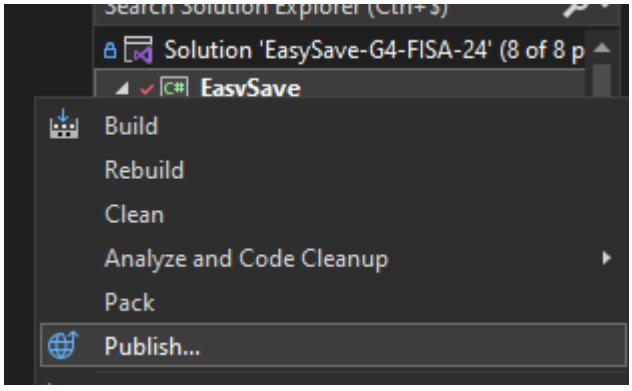
Windows

Lancer EasySave.exe dans le répertoire `../G4-FISA-24/EasySave-G4-FISA-24/EasySave/bin/Debug/net6.0/EasySave.exe`.

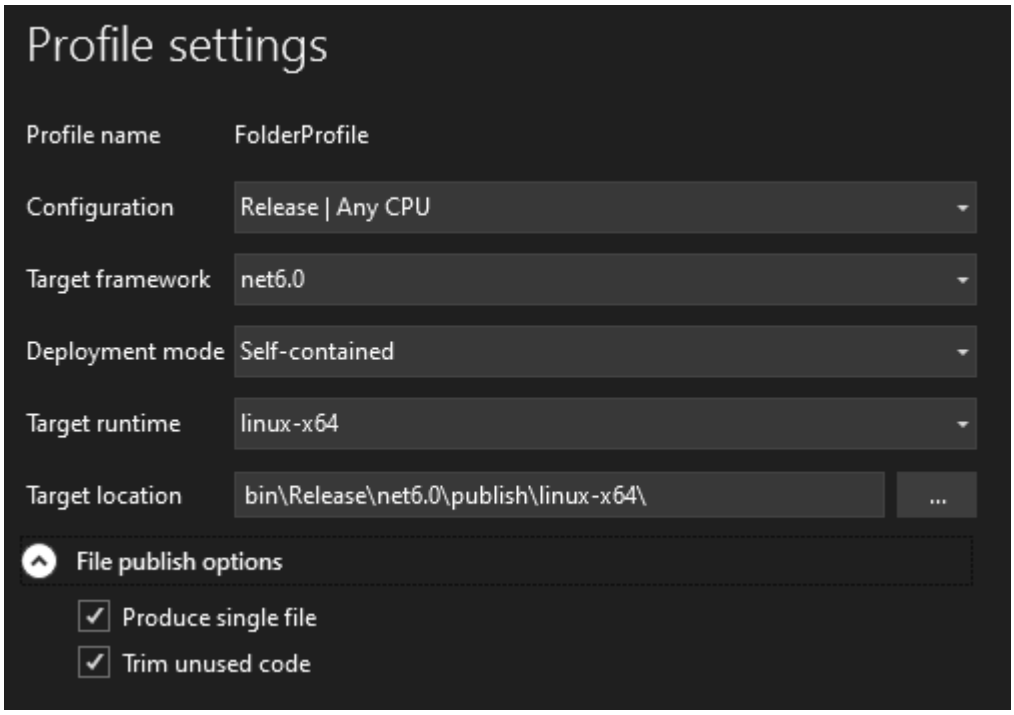
Linux

Exécutez `./EasySave` apres avoir fait la publication depuis visual studio.

Publication de EasySave

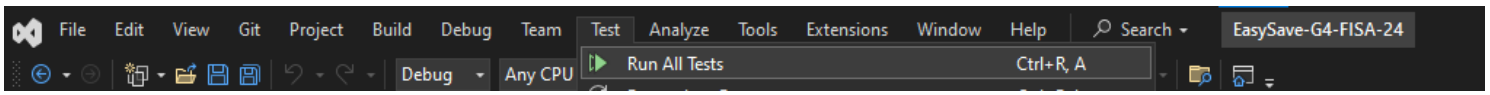


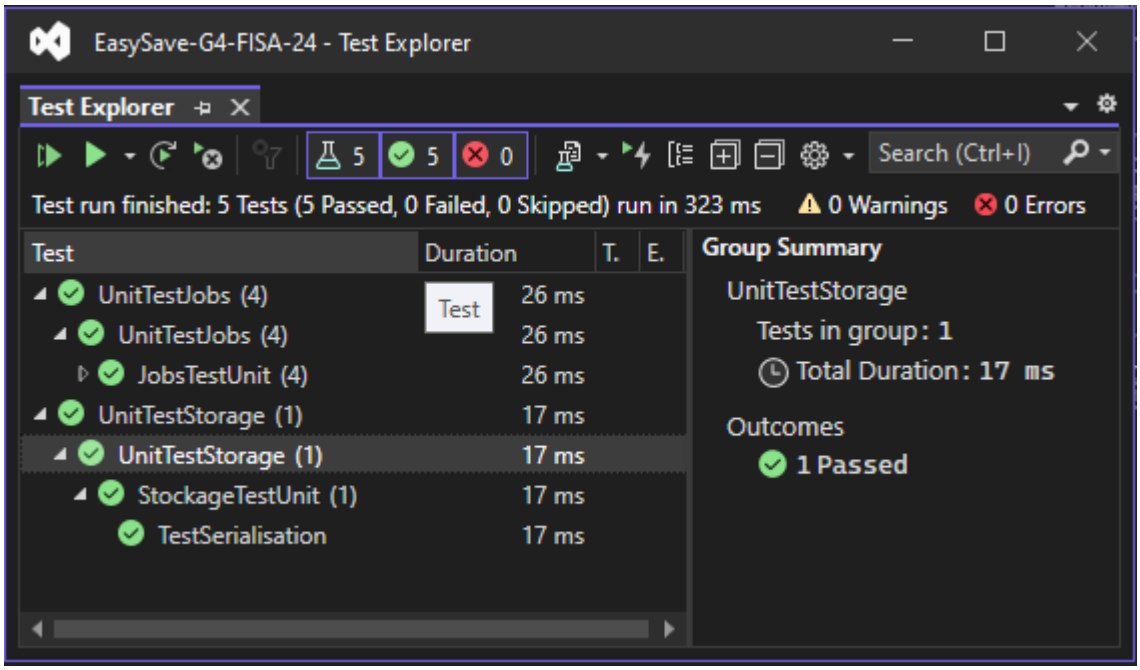
Il faut specifier le 'Target runtime' en fonction du système d'exploitation



Tests

Ouvrez l'explorateur de tests dans Visual Studio et exécutez les tests unitaires.





Tous les tests sont testés pendant la phase CI/CD, mais il est préférable d'exécuter les tests avant de 'push'.

Dernières versions

Veuillez consulter l'onglet "releases" pour accéder à la dernière version du livrable 1 (Pipelines -> Releases) :

https://dev.azure.com/faikmehmeti/G4-FISA-24/_release?_a=releases&view=mine&definitionId=1

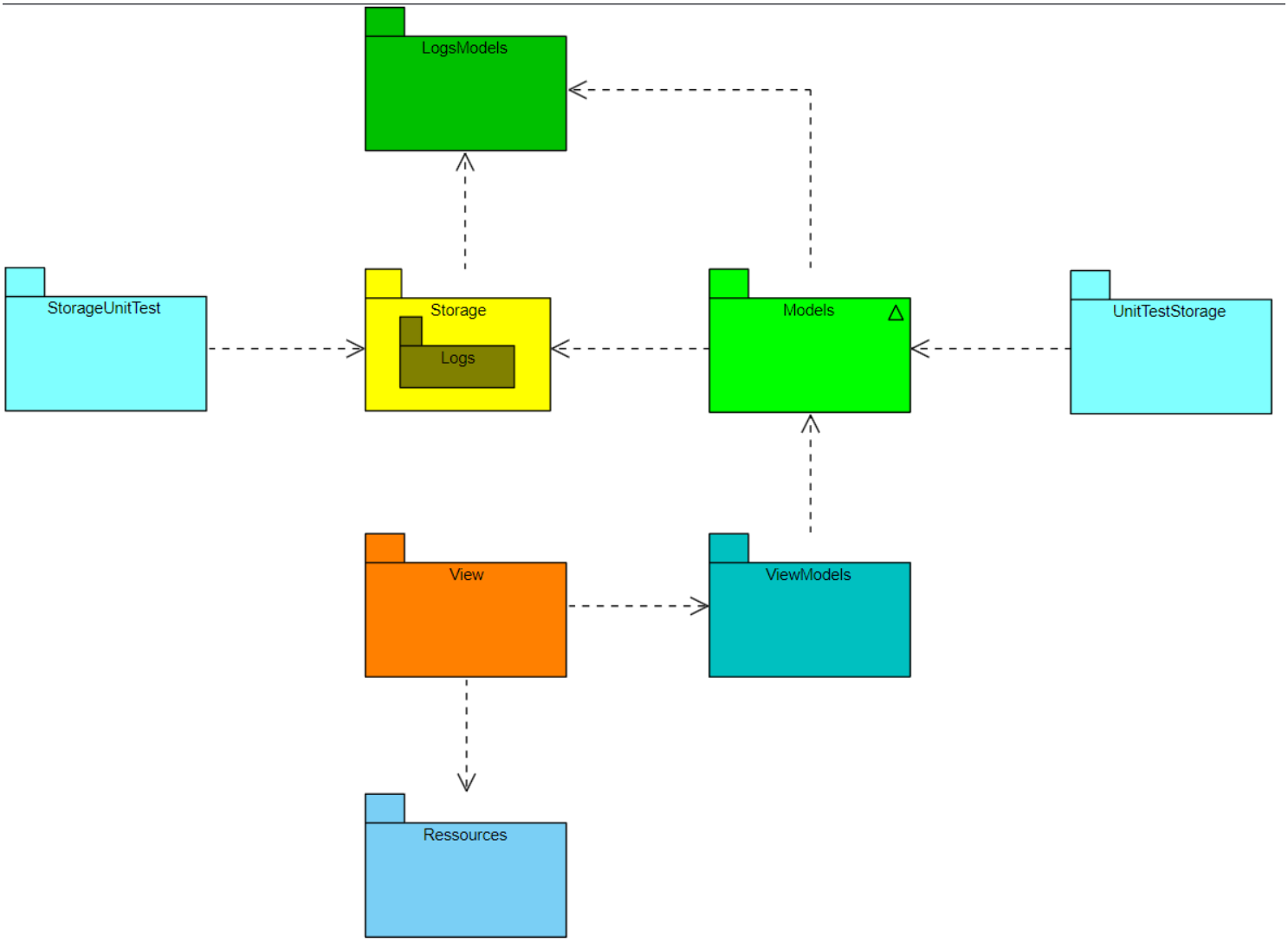
Fonctionnalités

- Sauvegarde séquentielle
- Journaux quotidiens
- Journaux d'état
- Choisir la langue
- Lister les jobs
- Charger une configuration de Jobs
- Créer un job
- Supprimer un job
- Lancer les jobs

Point d'entrée de l'application console

```
using View = EasySave.Views.View;
namespace EasySave // Note: actual namespace depends on the project name.
{
    /// <summary>
    /// Application entry point
    /// </summary>
    internal class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
            View pView = new View();
            //Lance le program principale
            pView.Run();
        }
    }
}
```

Architecture



L'architecture est basée sur un modèle MVVM avec :

- **Modèles** : classes représentant les données (CJob, CJobManager, CSettings, etc.)
- **Vues** : classes représentant les vues (BaseView, View, JobView, etc.)
- **ViewModels** : classes faisant le lien entre modèles et vues.

Systeme de Log

Emplacement par defaut du CJobManager et CSettings et l'emplacement des modèle de logs stockée grace aux classes de journalisation dans le package Logs

L'emplacement actuel des journaux d'événements est stocké dans le dossier `Logs`.

L'emplacement actuel des paramètres de l'utilisateur est stocké dans le répertoire racine `Settings.json`.

L'emplacement des tâches de sauvegarde est stocké dans le dossier `Jobs`.

Les logs dont le nom est formaté dans ce type `Logs - 2024-02-15` sont des logs journalier.

Le fichier `Logs.json` est l'état du journal.

Processus d'intégration continue (CI) et de déploiement continu (CD) utilisant Azure DevOps

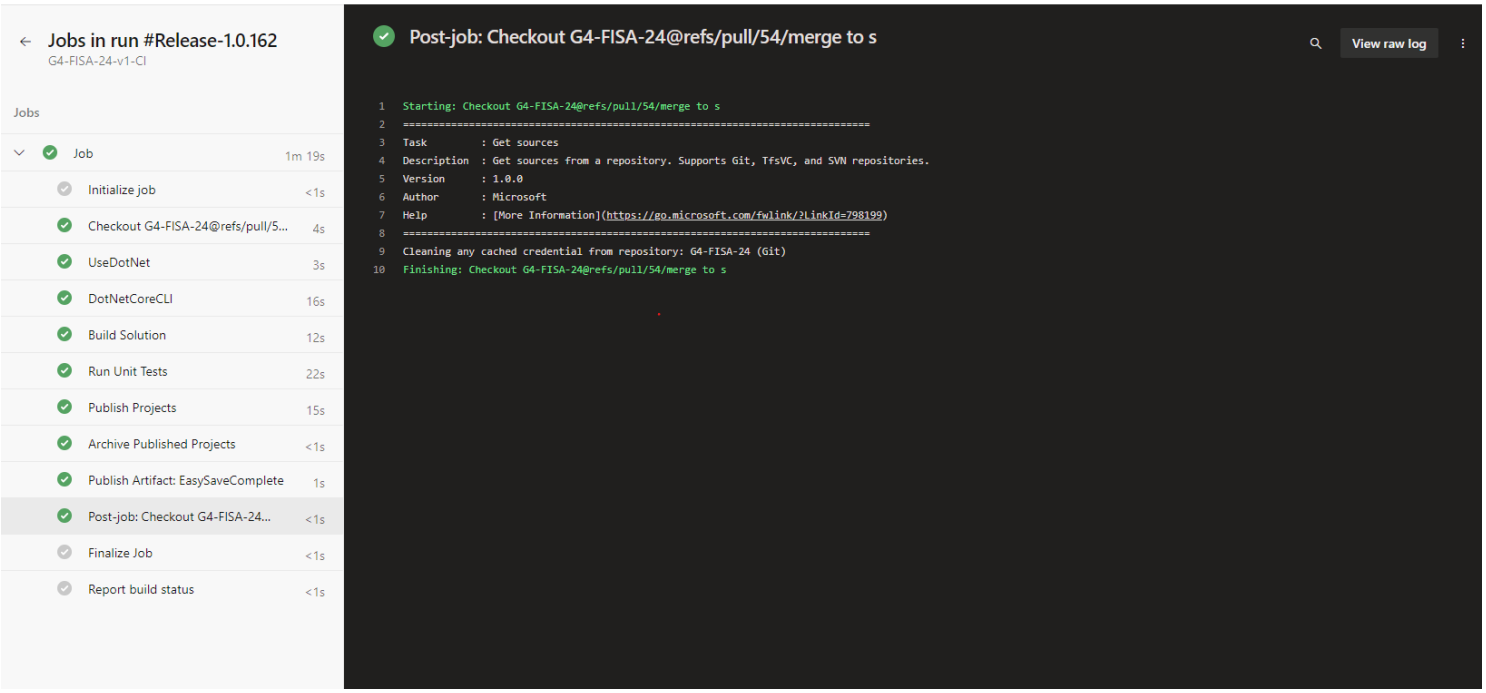
Configuration de l'Intégration Continue

Déclenchement des Pipelines

Les pipelines d'intégration continue sont configurés pour se déclencher automatiquement lors d'un pull request vers la branche principale. La validation de l'intégration continue est un prérequis avant toute fusion de code.

Processus d'Intégration

- Build : Compilation du code source en un exécutable ou en un paquet déployable.
- Tests Automatisés : Exécution de tests pour vérifier la qualité et la fonctionnalité du code.
- Publication des Releases : Création et stockage des artefacts de release dans Azure DevOps.

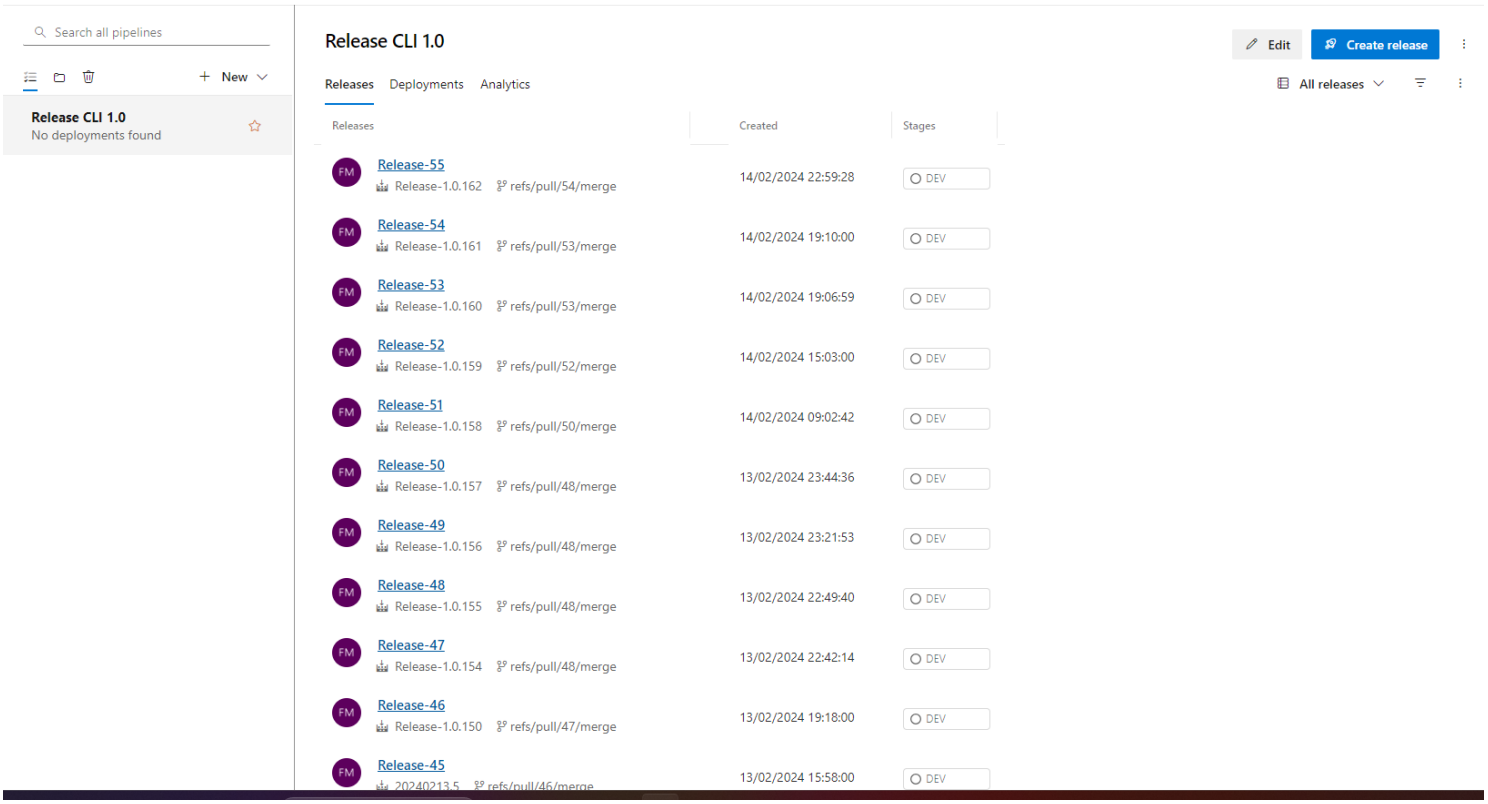


Gestion du Déploiement Continu

Bien que l'intégration continue soit automatisée, le déploiement continu nécessite actuellement une intervention manuelle.

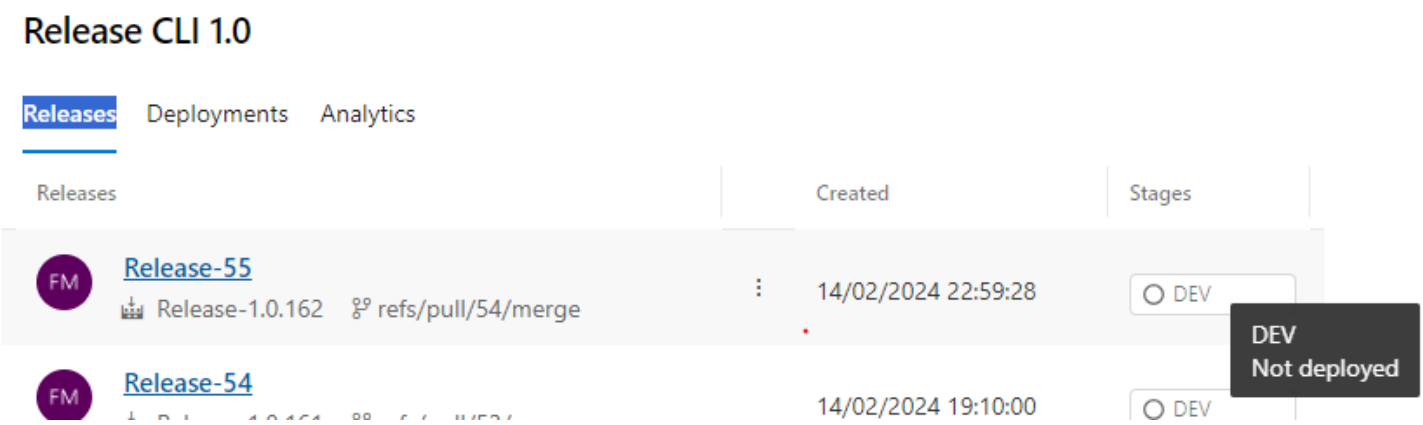
Accès aux Releases

https://dev.azure.com/faikmehmeti/G4-FISA-24/_release



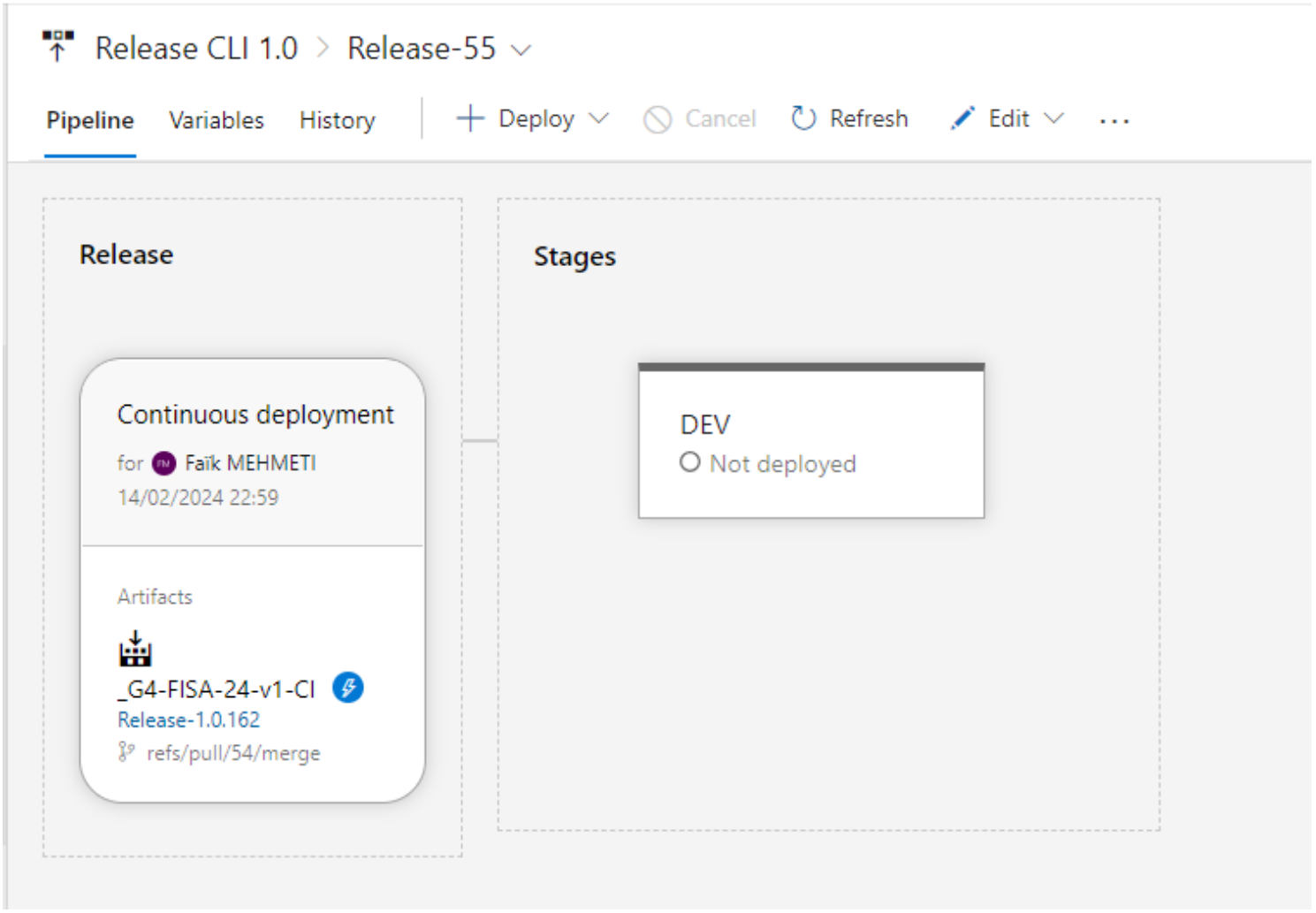
Vous pourriez suivre ici l'état de déploiement pour chaque release

Ce lien permet de suivre l'état de déploiement pour chaque release.



Gestion du Déploiement

En sélectionnant une release, vous serez dirigé vers une page où le déploiement continu peut être géré. Actuellement, les déploiements doivent être effectués manuellement pour chaque étape (Stages).


Release CLI 1.0
Release-55
Pipeline
Variables
History
Deploy
Cancel
Refresh
Edit
Release
Continuous deployment
for Faïk MEHMETI
14/02/2024 22:59
Artifacts
_G4-FISA-24-v1-CI
Release-1.0.162
refs/pull/54/merge
Stages
DEV
Not deployed