MERLIN

Version: 02.00.00

Getting Started

MERLIN Getting Started Guide

www.arcadsoftware.com

Document Version	:	**02.00.00**
Publication Date	:	**March 11, 2024**
Author	:	**FGR**

North America & LATAM
70 Main Street, Suite 203
Peterborough NH 03458
USA
1-800-676-4709 (toll free)
1-603-371-9074
sales-us@arcadsoftware.com

EMEA (HQ)
55, rue Adrastée - Parc Altaïs
F-74650 Chavanod/Annecy
FRANCE
+33 450 578 396
sales-eu@arcadsoftware.com

Asia Pacific
c/o Pramex Intl Ltd
1 Austin Rd West Intl Commerce Centre
7107B 71/F Tsim Sha Tsui HONG KONG
Yau Ma tei
Hong Kong
+852 3618 6118
sales-asia@arcadsoftware.com

Table of Contents

1 Goal

The goal of this document is to help the end user to get started with MERLIN.

It describes how to configure an IBM i application in Merlin and how to develop with the product.

This is not a training guide, meaning that all features are not listed here.

This Getting Started Guide is based on the ARCAD Demo application.

2 Important information

2.1 Prerequisites

Before you start, make sure that you:

- have access to the Merlin environment.
- Know how to use Git.
- have some knowledge of ARCAD.

2.2 ARCAD Products used in MERLIN

The products installed and used in MERLIN are the following:

- ARCAD Server and Elias.
  - ARCAD Server 23.02.11
  - Elias 2.0.1
- ARCAD Extension vsix.
- ARCAD Observer.
- ARCAD Builder Server.
- ARCAD Builder Client.
- ARCAD Skipper.
- ARCAD Transformer RPG.

2.3 ARCAD concepts

2.3.1 ARCAD application

An ARCAD application brings together the libraries forming a coherent whole (libraries of source files, libraries of objects, libraries of data files, etc.) more in the logistical sense than in the functional sense.

An application has a code, some operational attributes, in particular the application libraries and a liblist.

This liblist contains the application libraries and all other libraries needed to compile the source member of the application.

This list of libraries will be used to load the cross references between components.

2.3.2 ARCAD environment

An ARCAD environment is a complete and independent work context.

There are different types of environments:

- **Production** (or operational): environment in which users work.
- **Reference**: environment that includes the elements used in the production environment. It is used to create the ARCAD repository.
- **Development**: environment in which programmers develop. They also perform their unit tests in this environment.
- **Tests**: environment that allows you to carry out tests while remaining independent of the production environment.

2.3.3 ARCAD version

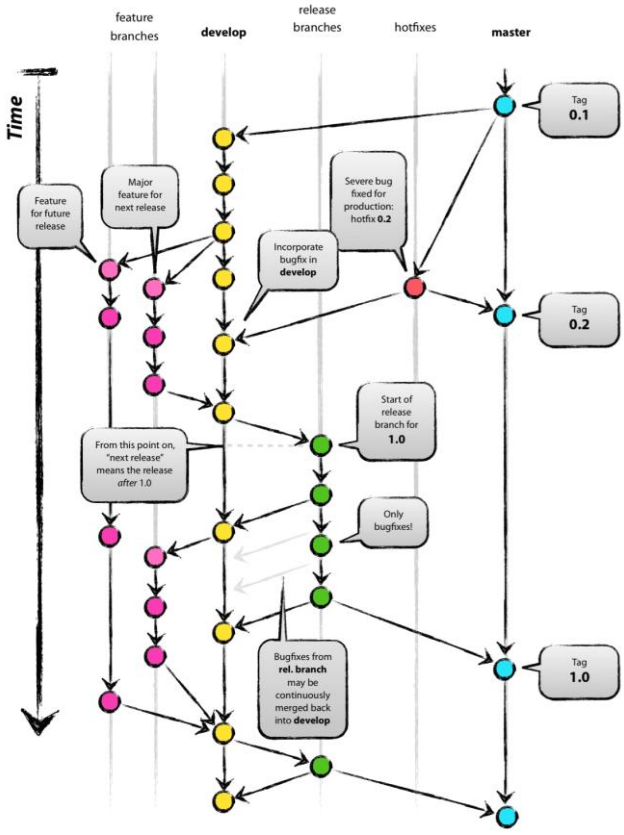
An ARCAD version is a library linked to a development environment, and, with MERLIN, linked to a Git branch.

This library will be used to receive the source members modified in a Git environment and to recompile them.

This library will be used for unit tests and also as a starting point to send these compiled objects to a test or a production environment.

2.4 Development workflow with Merlin

The development with Merlin is freely inspired on the GitFlow workflow but you can adapt it to your own needs.



Gitflow is based on Git Branches.

There are two branches where direct changes are forbidden:

- The first one is the ***master*** branch which is a picture of the production sources.
- The second one is the ***development*** branch where all new features will be brought together.

Three other types of branches can be used:

- ***Feature***: used by developers to create new feature for an application.
  New features are made in feature branches. When a development is done, it can be merged into the development branch, and the feature branch can be deleted.
- ***Release***: used to bring together several features. This branch is used by developers to apply patches after tests.

A new release receives changes from the development branch, and when the tests are done, the new release can be merged into the master branch, as well as the development branch to keep it up to date. The release branch can then be deleted.

- ***Hotfix***: used by developers to fix a bug in production.
  A hotfix is made to fix a production bug. When it is done, the hotfix can be merged into the master branch, as well as the development branch to keep it up to date. The fix branch can then be deleted.

With Merlin, each new branch will automatically create a new ARCAD version.

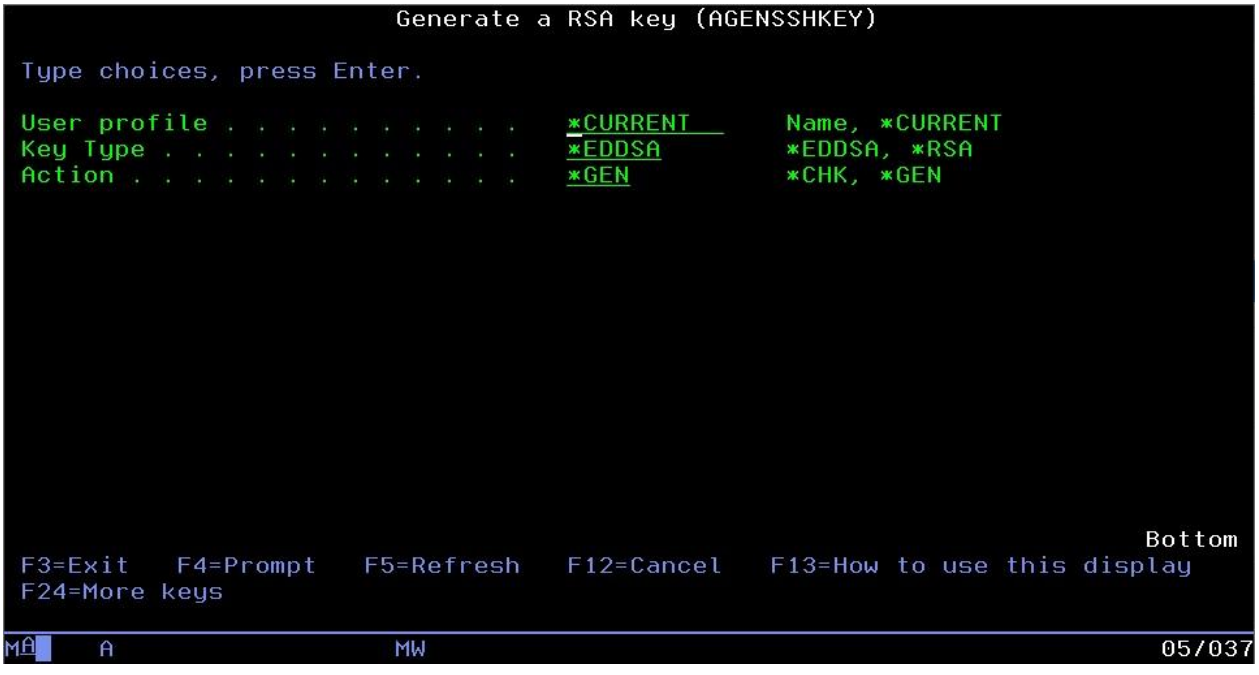
2.5 Developing with MERLIN

2.5.1 Generate SSH keys and add them to your Git profile

If you didn't have SSH keys for your profile on your IBM i, you need to calculate one and then add it to your Git profile.

To create SSH keys, you can use the ARCAD Command AGENSSHKEY or the command SSH-KEYGEN in a QSH session.

With the ARCAD Command, just fill the Key type you want to generate, the Action ***GEN** and then press ENTER.

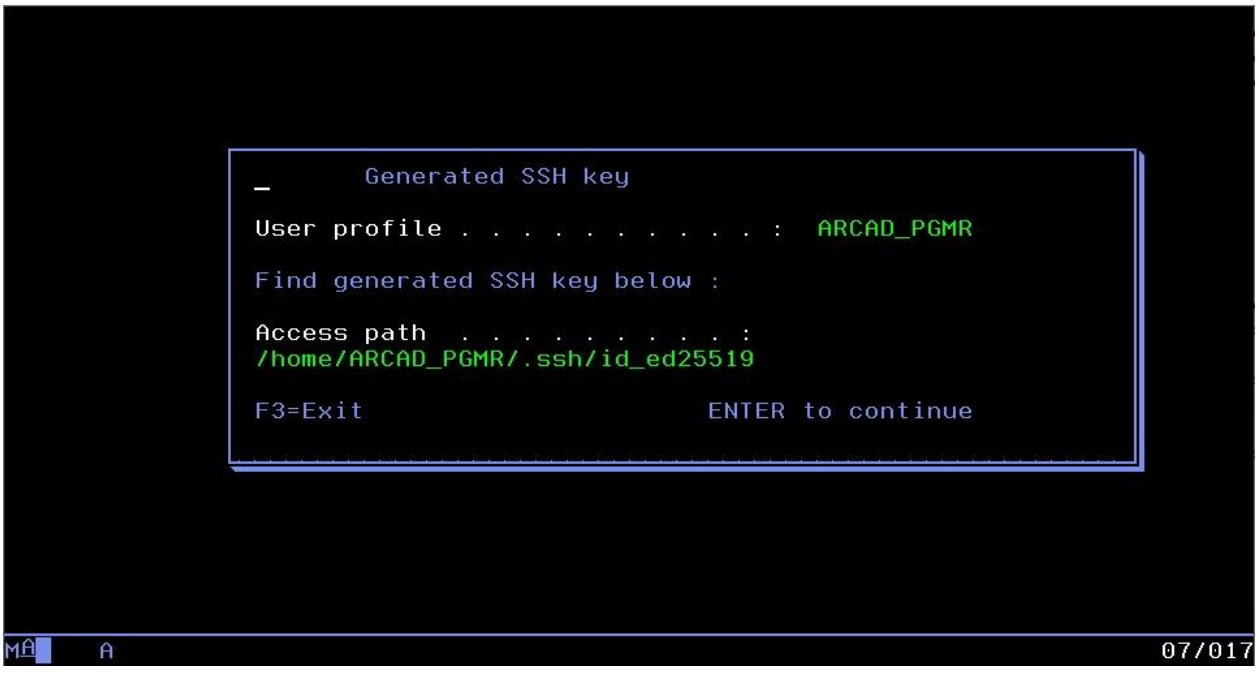


You can use this command to check if SSH Keys already exist or not.
In that case this command will display these kind of messages (for example for the user profile ARCAD_PGMR):

- id_ed25519 private key was not found in IFS folder /home/ARCAD_PGMR/.ssh.
- An EdDSA key pair already exists in IFS folder /home/ARCAD_PGMR/.ssh.
- id_rsa private key was not found in IFS folder /home/ARCAD_PGMR/.ssh.
- An RSA key pair already exists in IFS folder /home/ARCAD_PGMR/.ssh.

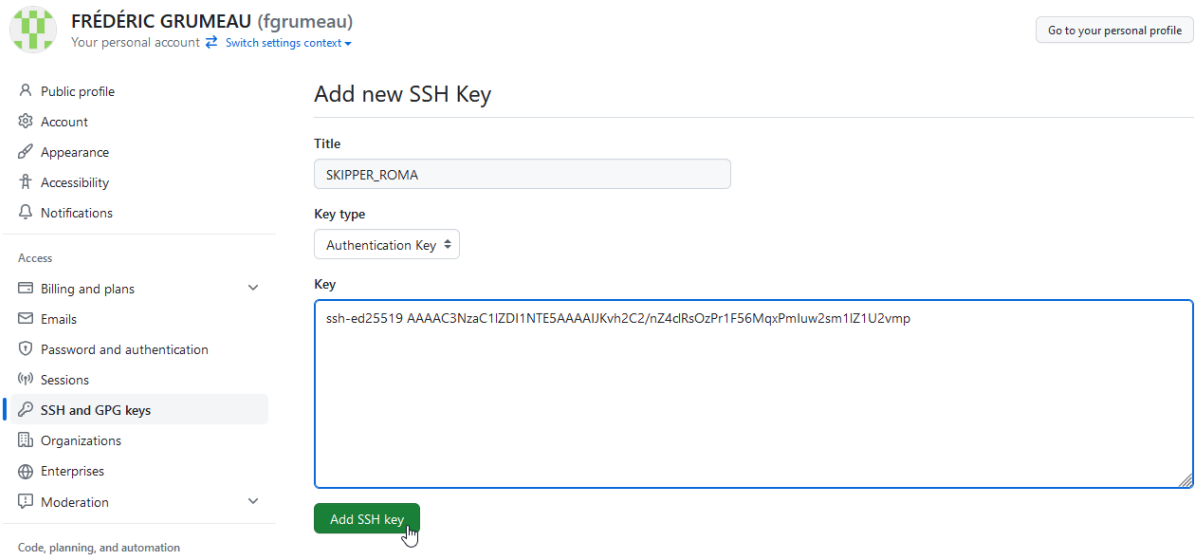
The command indicates that the Keys are generated and where you can find them.

Press ENTER to finish this creation.



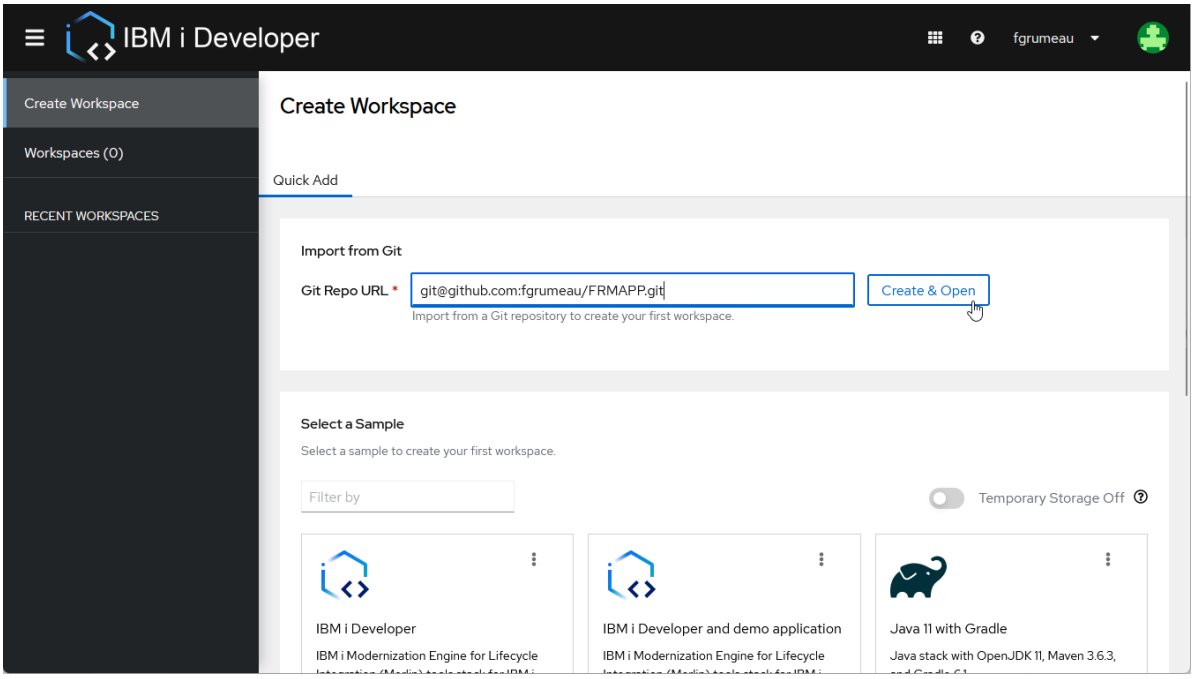
Display and copy the public key, for example with the command cat in a QSH session (or the WRKLNK command).

Paste this key in your Git profile.

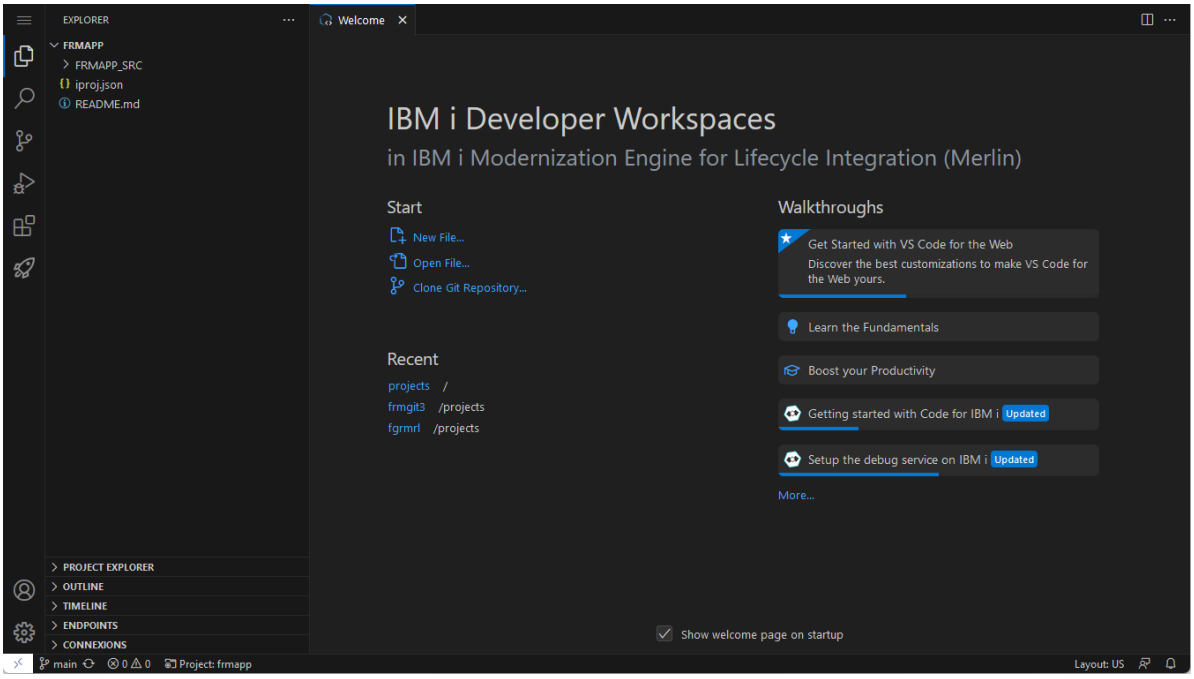


2.5.2 Clone the Git repository in your workspace

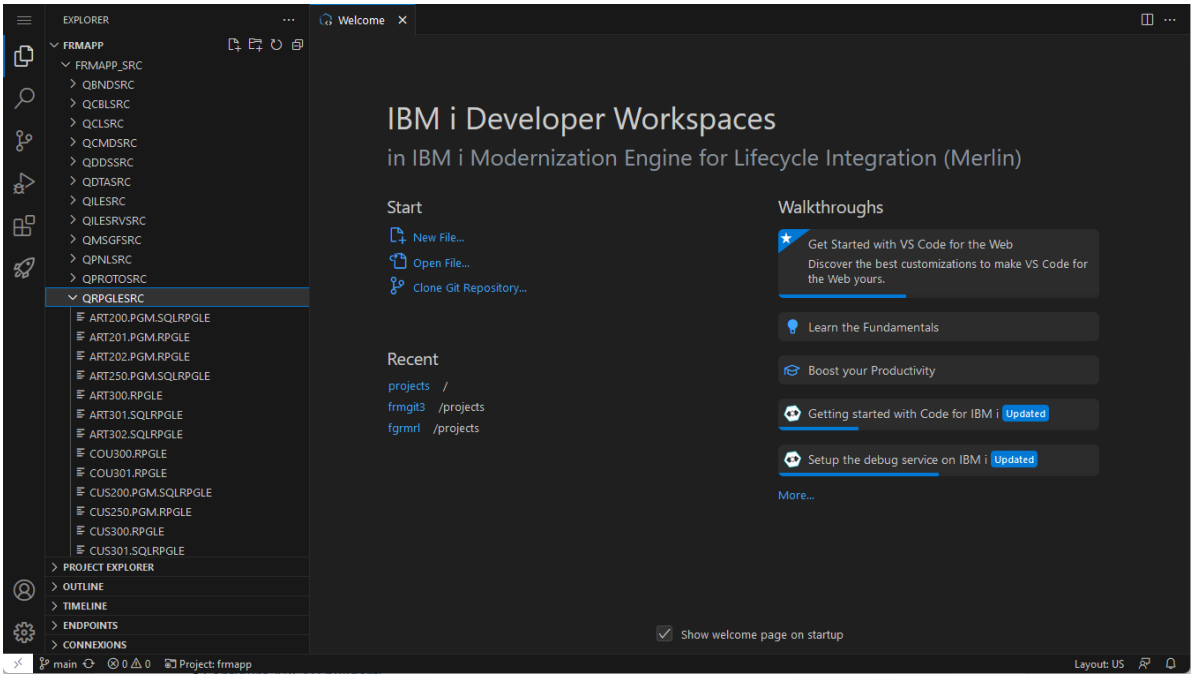
Go to the IBM i Developer IDE in Merlin, select the option Create Workspace, paste the URL of your git repository in the part **Import from Git** and click on **Create & Open**.



When it is finished, your Git repo was cloned.

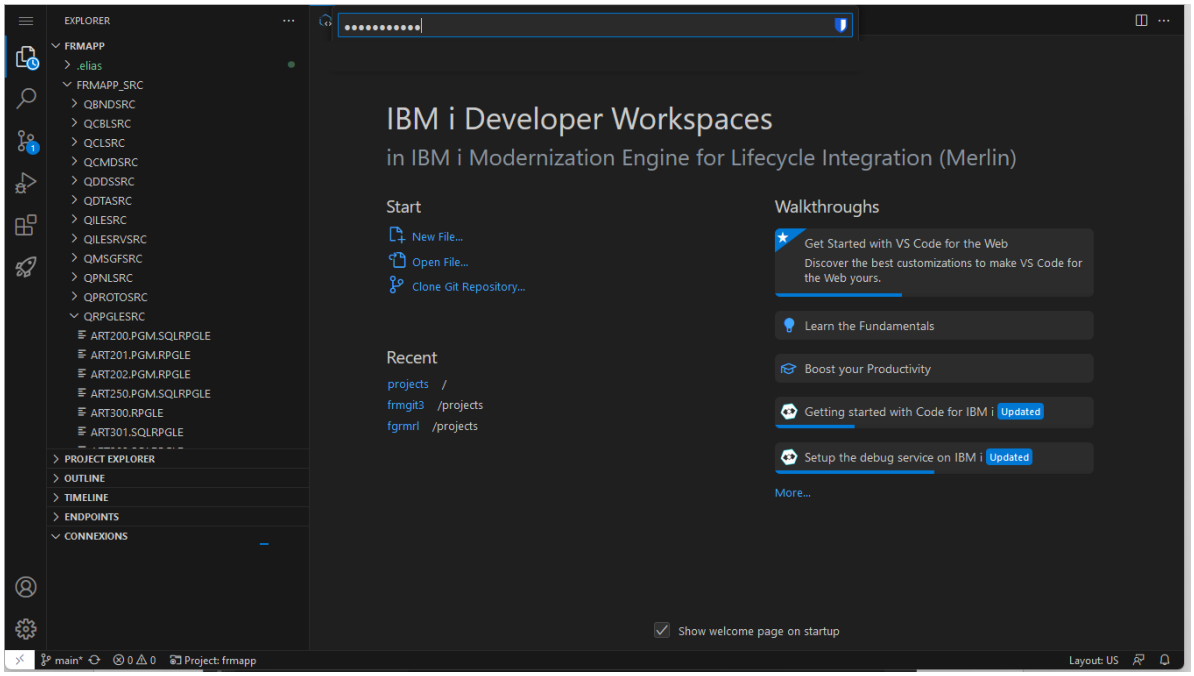


And you can see all the source members in your workspace.

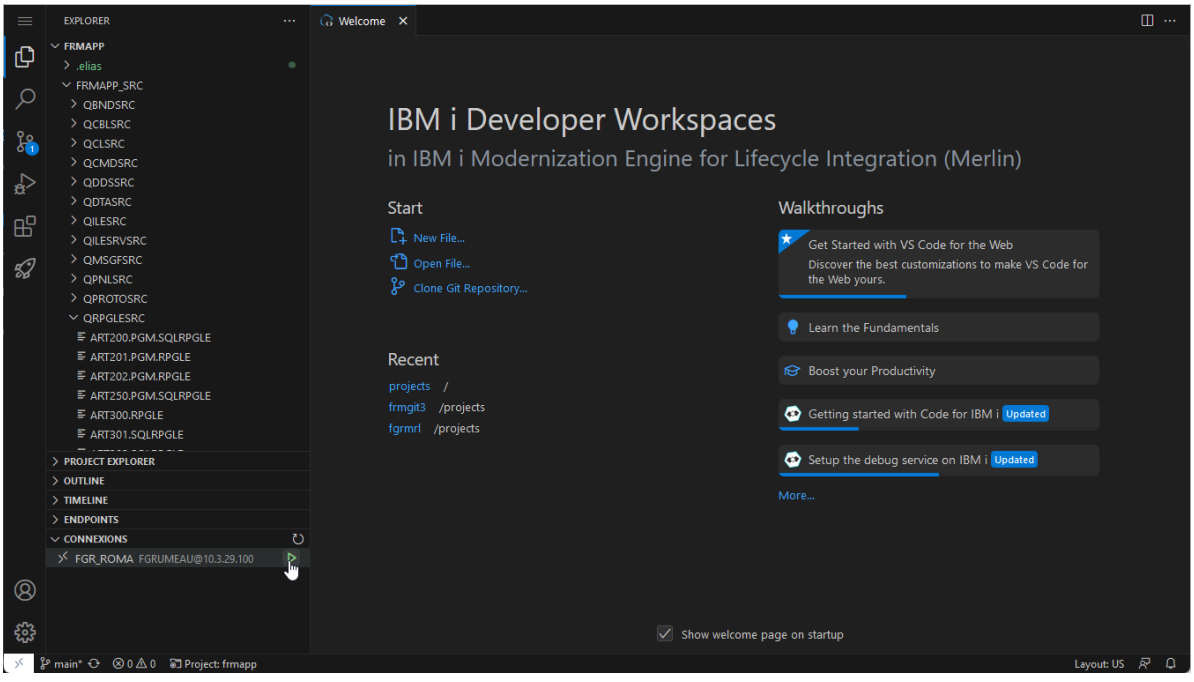


Connect to the IBM i

Go to the node **CONNECTIONS** and enter your Merlin password to see the connection to IBM i you can used.



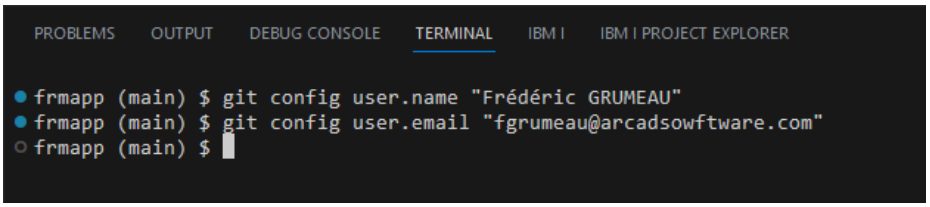
Then click on the triangle icon to do the connection.



When the connection is done, you are ready to work with Merlin.

Before working with Merlin, you probably need to configure your name and email address for Git.

To do this, simply go to a terminal and run the commands **git config user.name "Your Name"** and **git config user.email "Your email"**

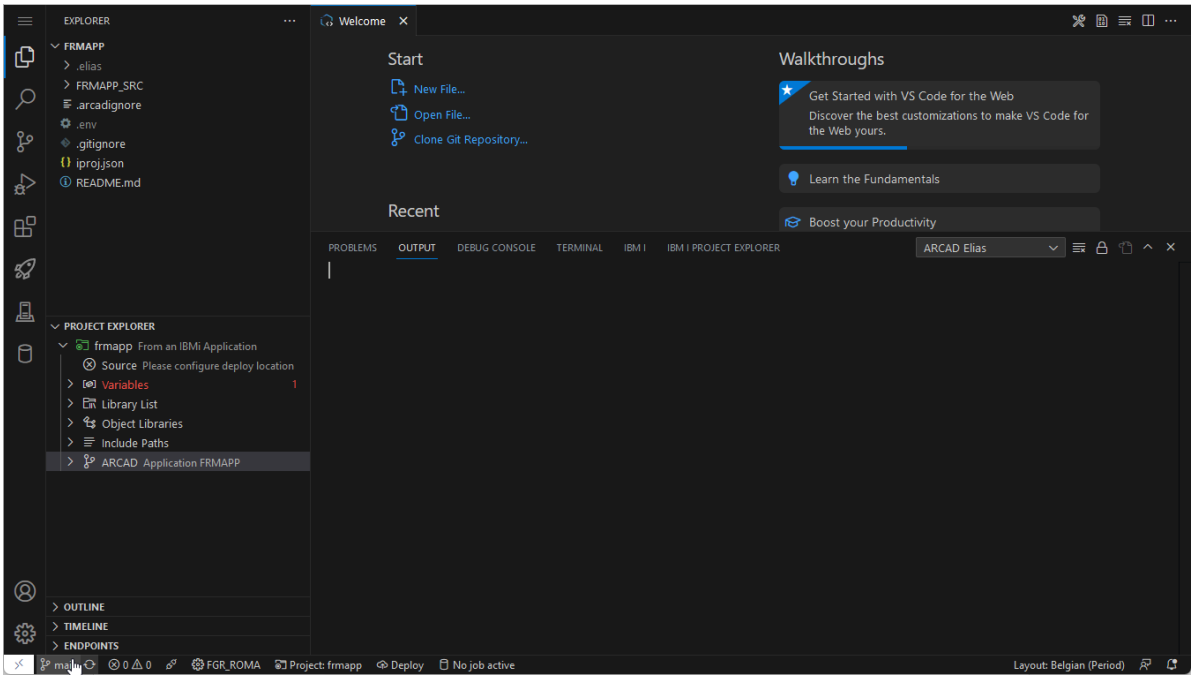


2.5.3 Creating a branch

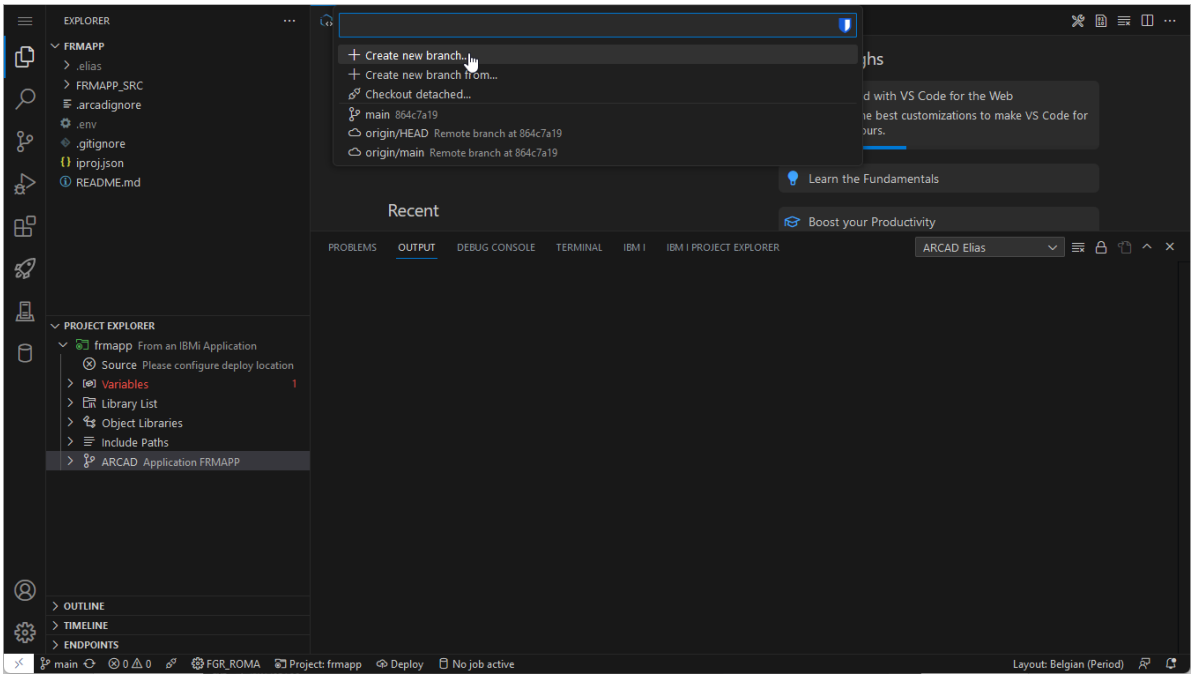
The first thing to do is to create a feature branch.

This action can be performed in your Git repository or directly in your workspace. If you do this action in your Git repository then you will have to select this branch in your workspace. In this procedure, we will do this action directly in the IBM i Developer workspace.

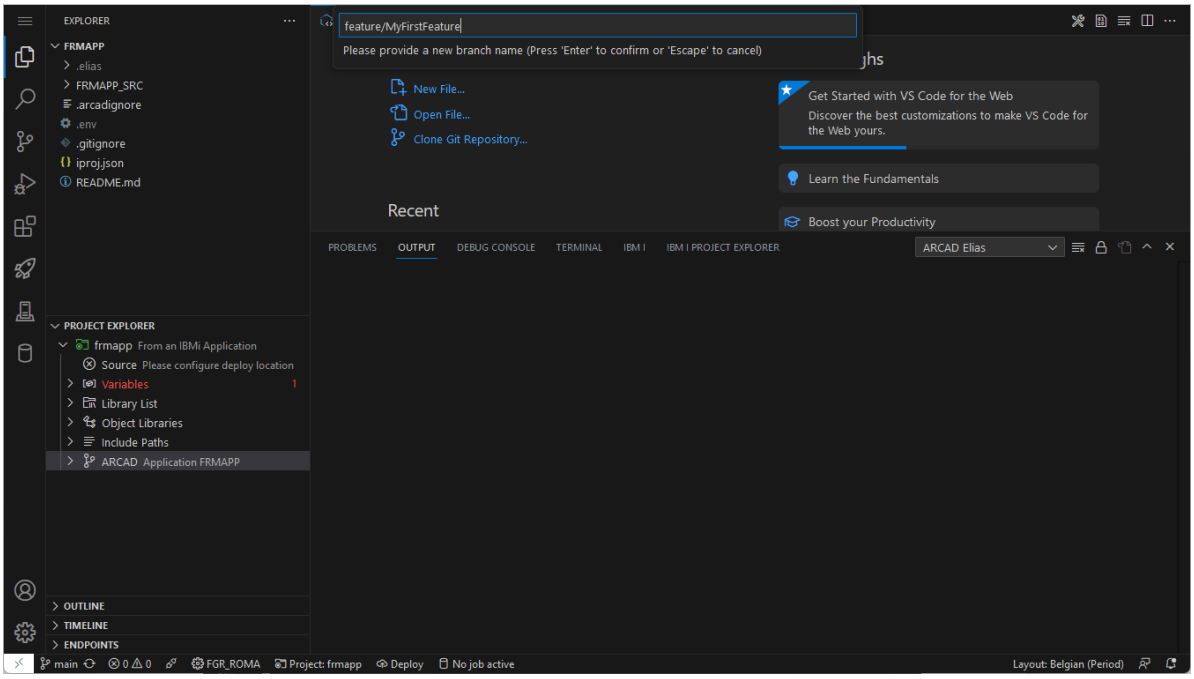
Click on the ***main*** (or on the name of your default branch) which is at the bottom left of your workspace.



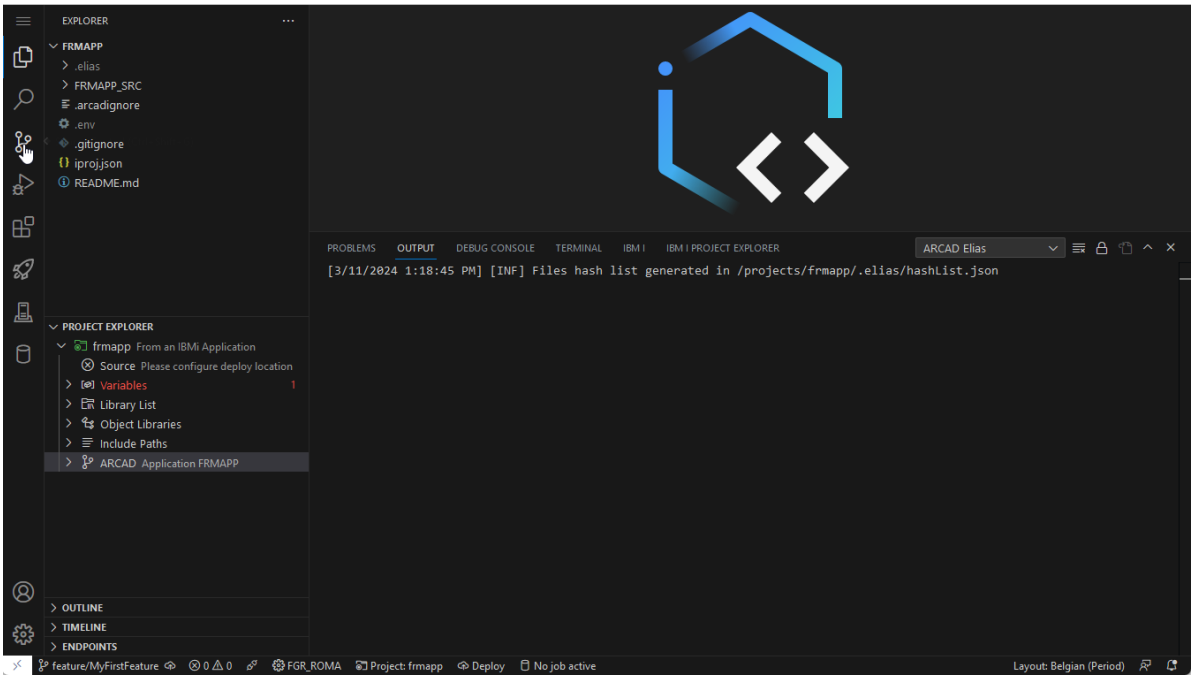
On the top of your workspace a menu appears, select **Create new branch**.



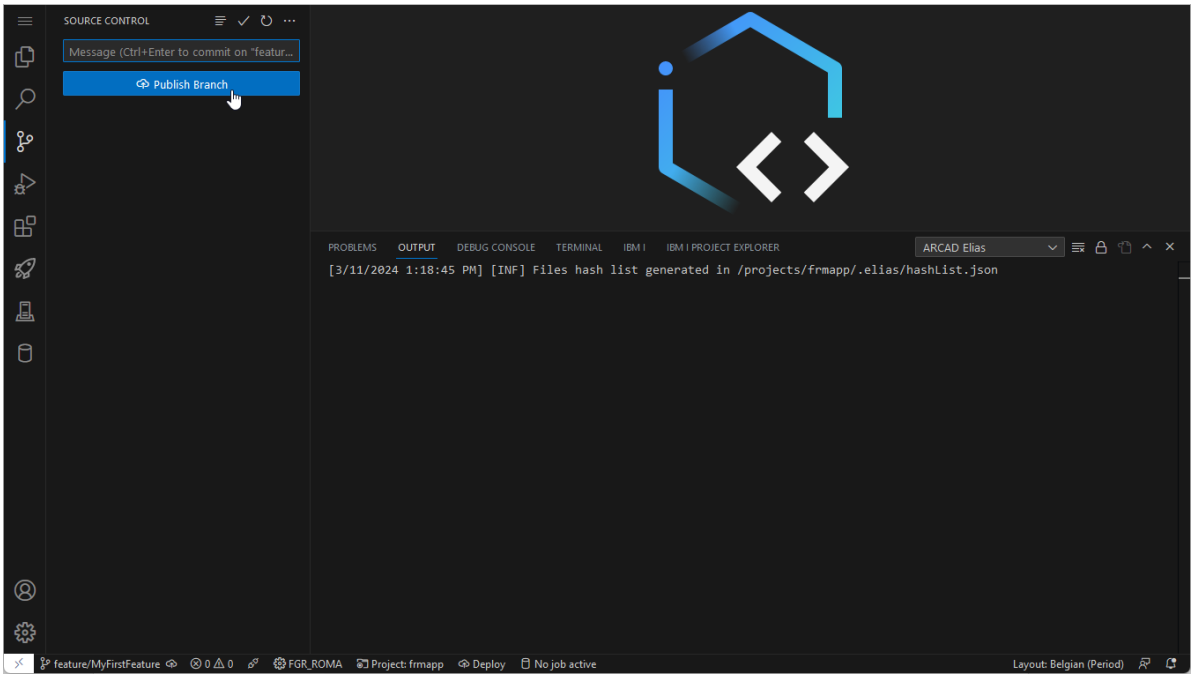
Enter the name of your new branch, this name must begin with “feature/”, and press ENTER to create a new local branch.



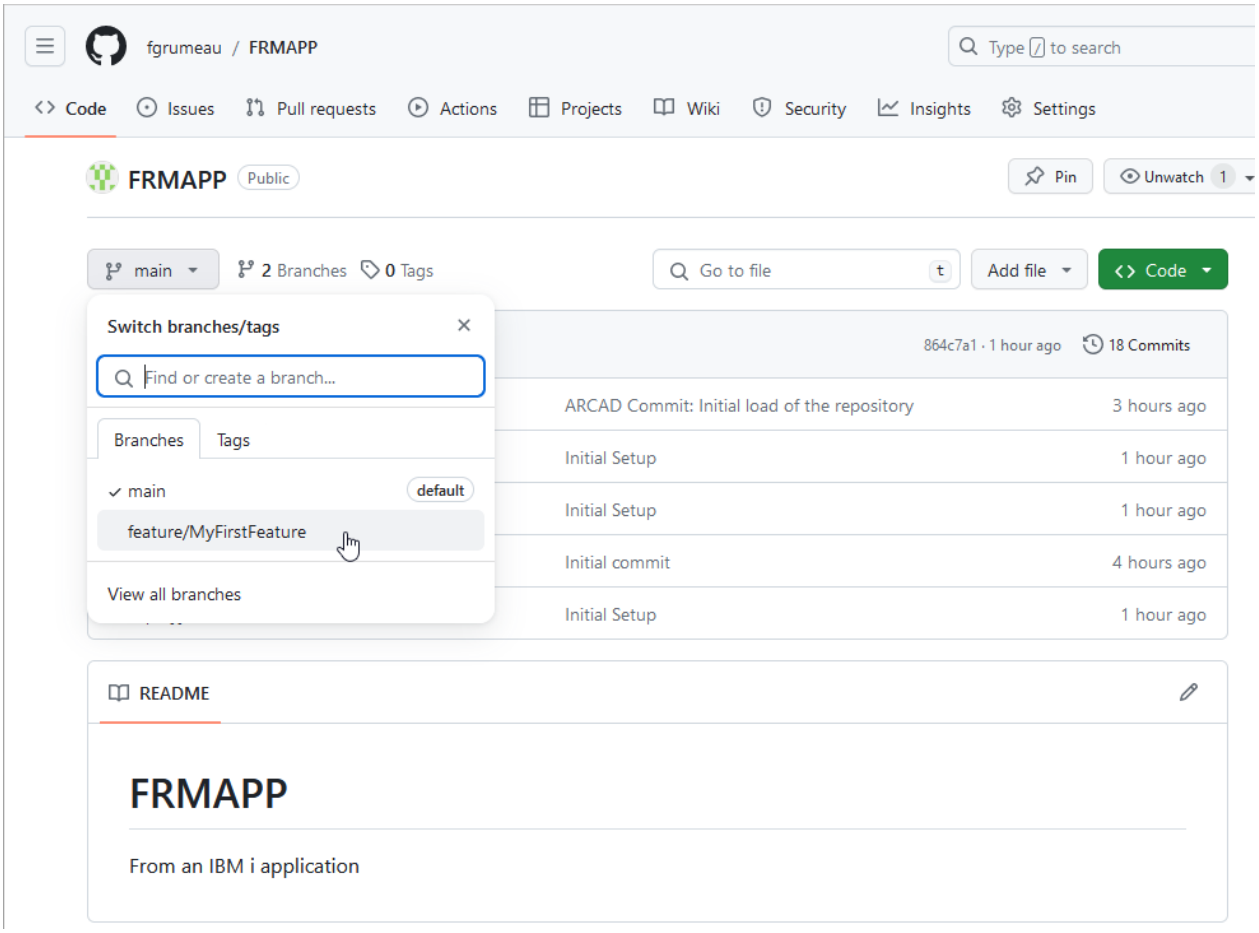
Now you can see the name of your branch in the bottom left of the workspace. Click on the **Source Control:Git** icon to open the Source Control view and push your feature local branch into your Git repository.



In the Source Control:Git part of your workspace, click on **Publish Branch**.

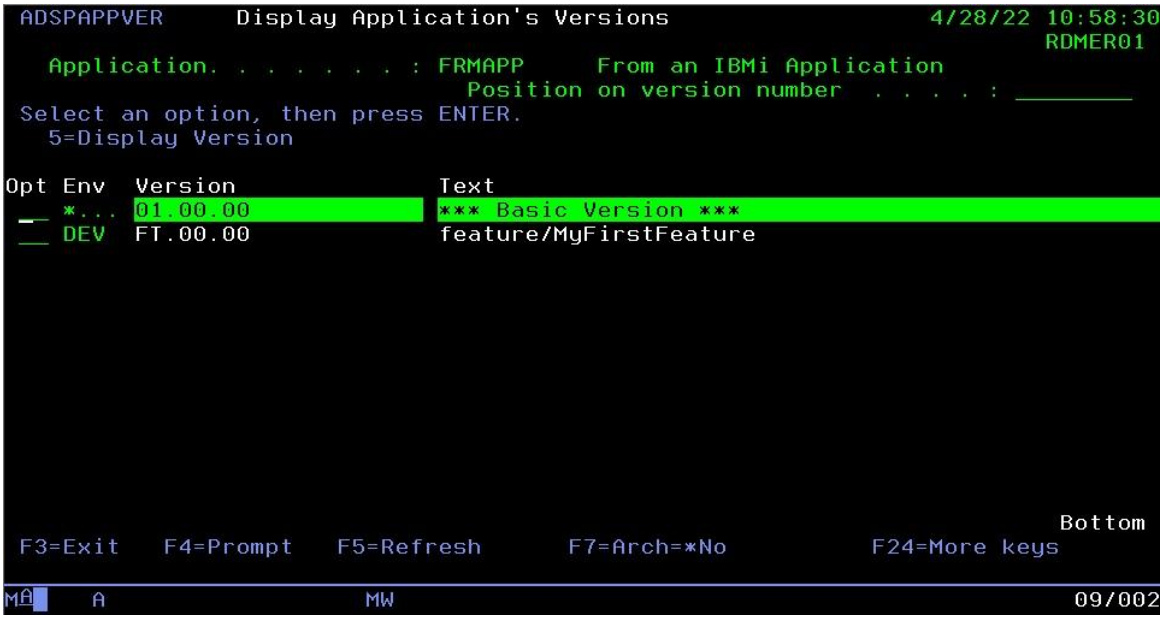


When the push action is complete, you can see in your Git repository the new feature branch.



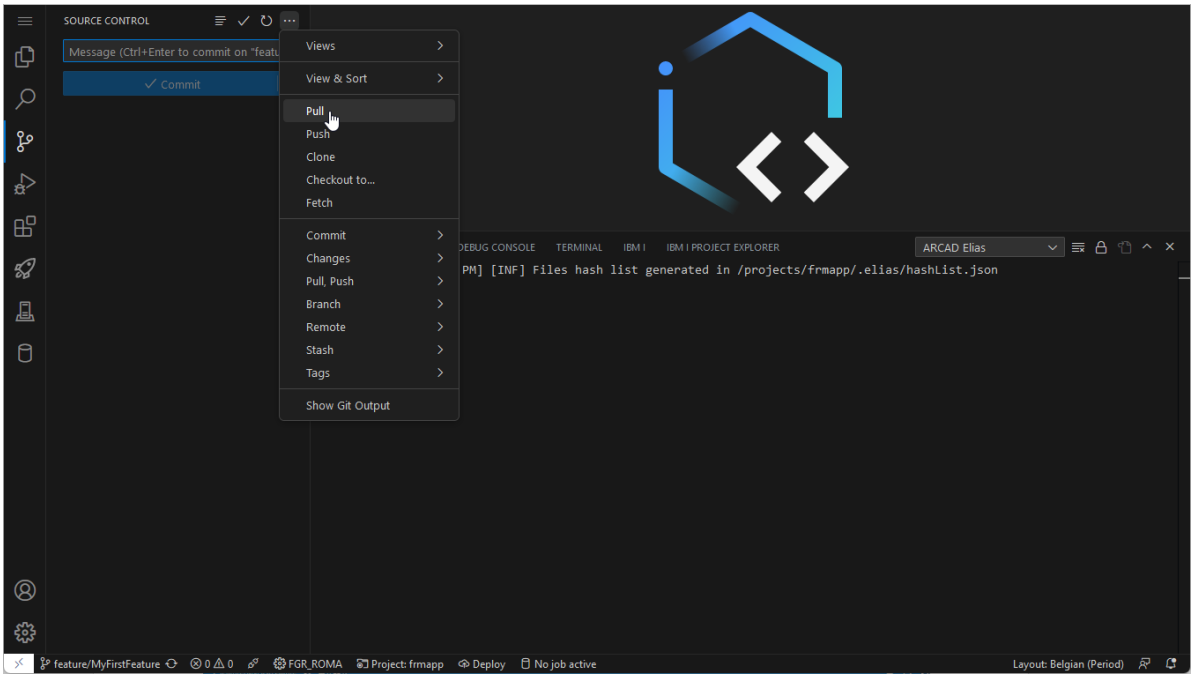
If you added Webhooks to your Git repository, the creation of the remote feature branch through the push action automatically created a new ARCAD Skipper version.

You can see this creation with the AWRKAPP command, option 17 Display application versions but we will see later in this guide how to have this same information directly in your IBM i Developer workspace.

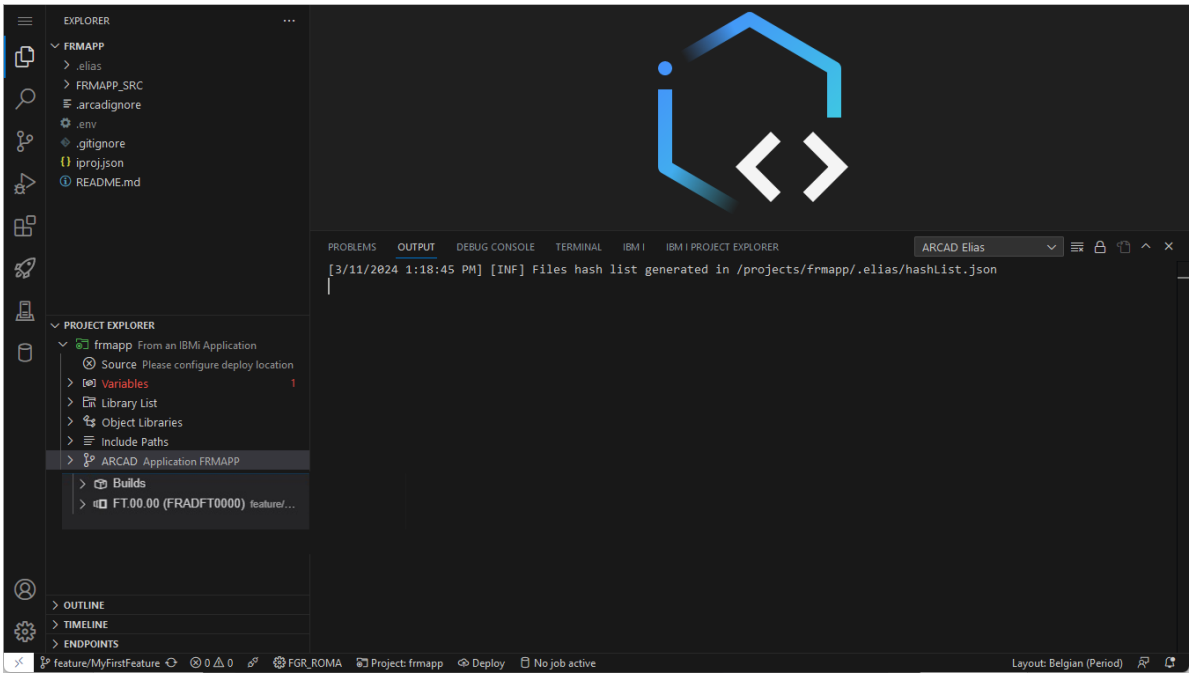


Once the version is successfully created by the webhook, it generates a commit named by default ARCAD Build Commit [ci-skip].

It is recommended to make a pull for your local repository by clicking on the menu and select **Pull**.

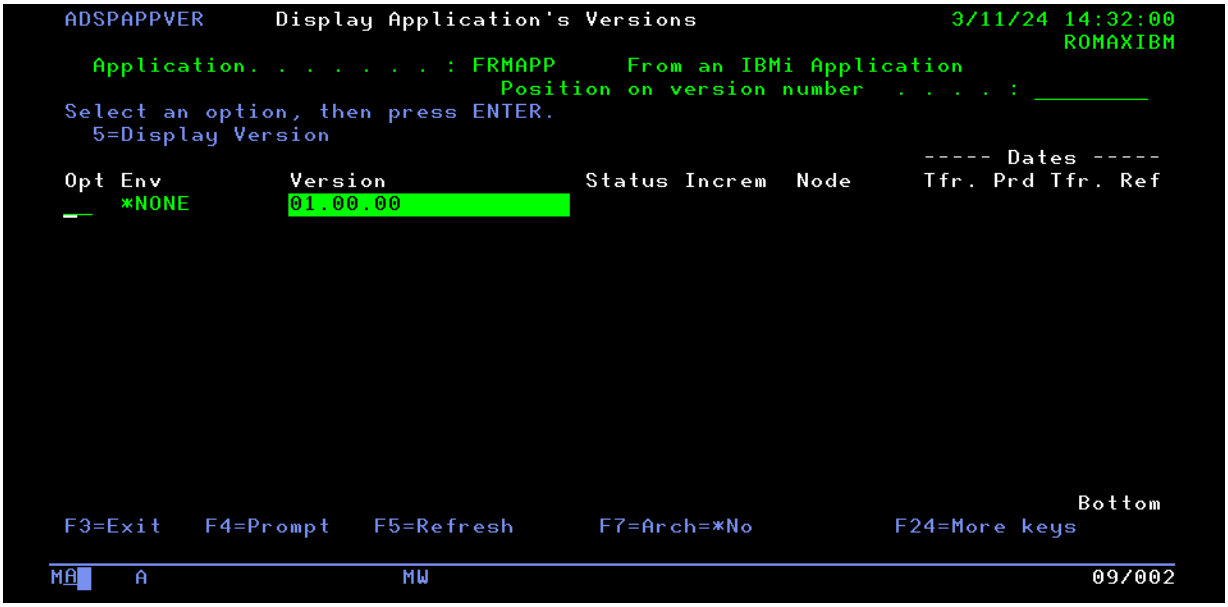


If you click on the ARCAD part of the PROJECT EXPLORER, you will see the version created by the webhook. This is the other way to check the creation of the ARCAD Skipper Version.

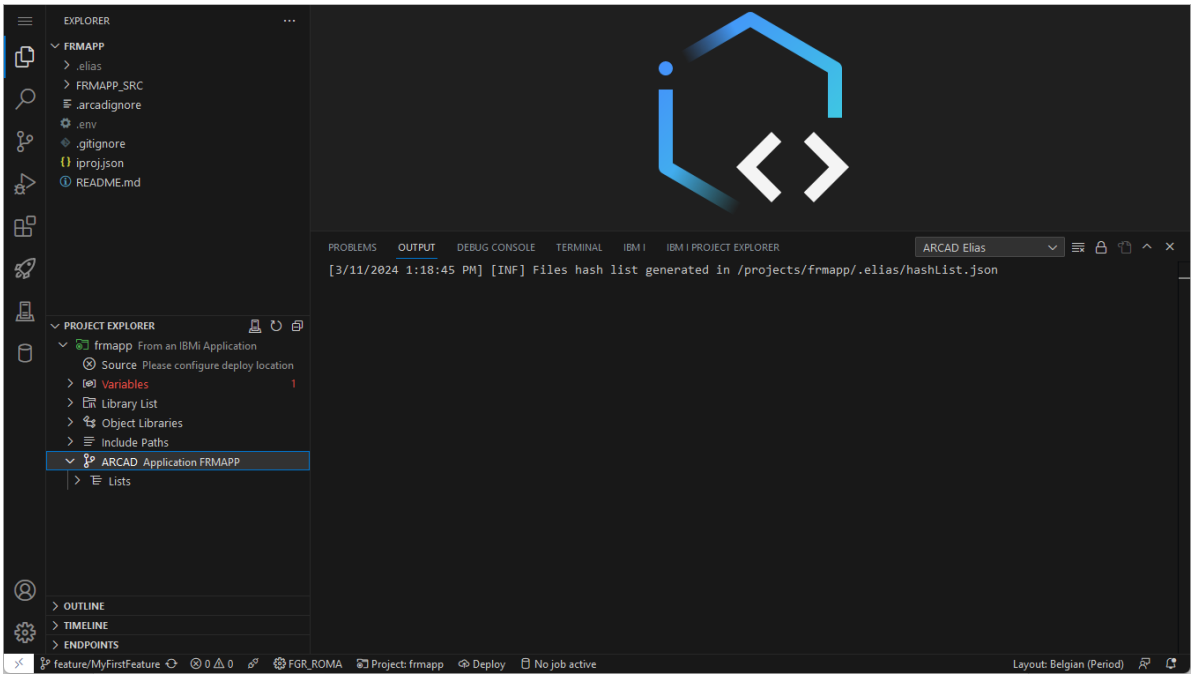


Getting Started – v 02.00.00

If you didn't add the webhook, then with the AWRKAPP command, option 17 you will only see the initial version.



And no version on the ARCAD part of the PROJECT EXPLORER.

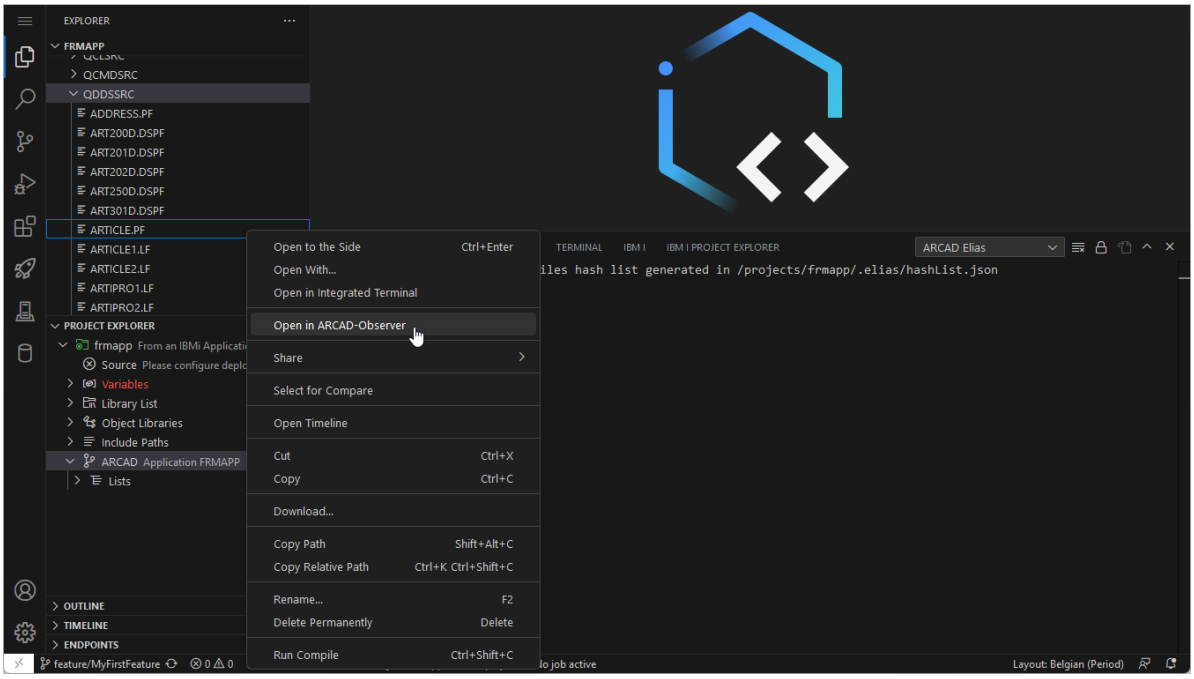


In any case, you are now ready to make changes in the application.

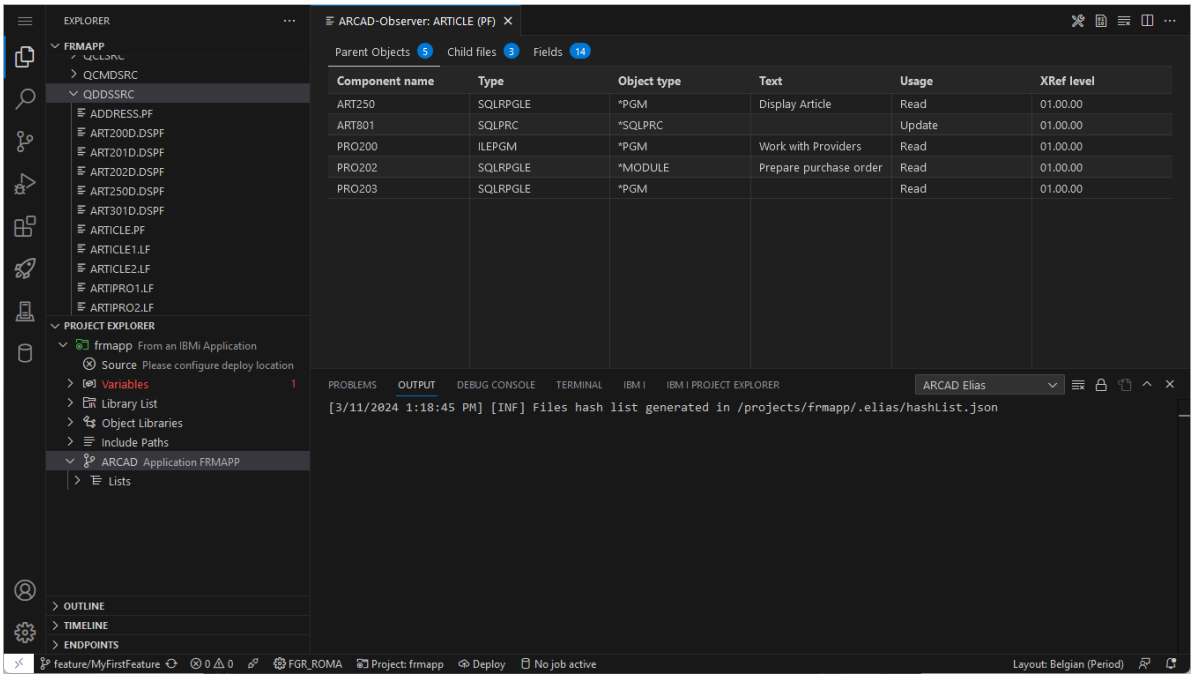
2.5.4 ARCAD actions *Display cross-references*

Before you start making changes, you can browse links between components by using the ARCAD Xrefs.

To do so, right-click on a source member and select the **Open in ARCAD-Observer** option.

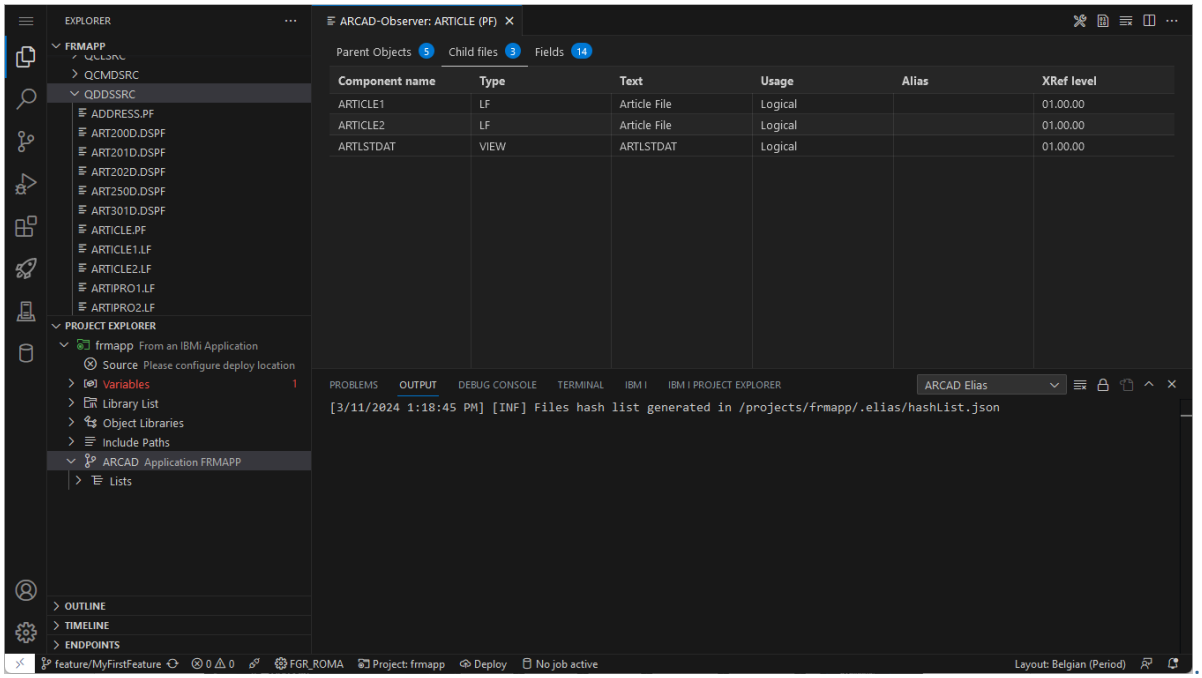


A view appears with links to programs (Parents Objects).

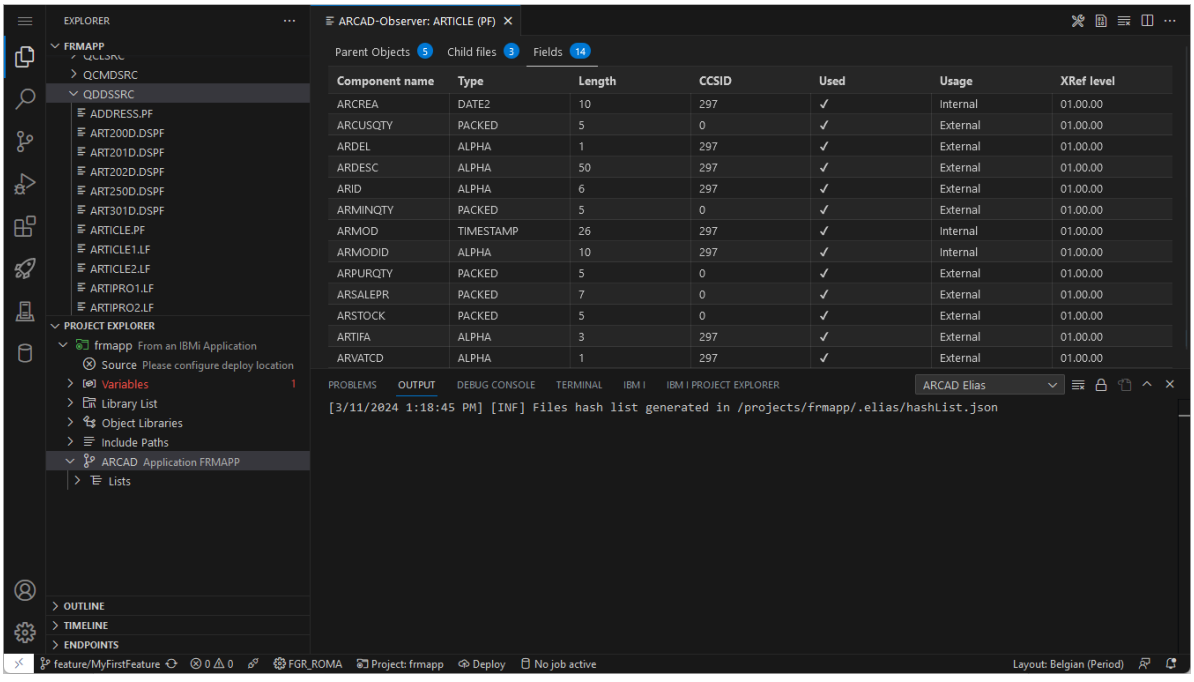


Getting Started – v 02.00.00

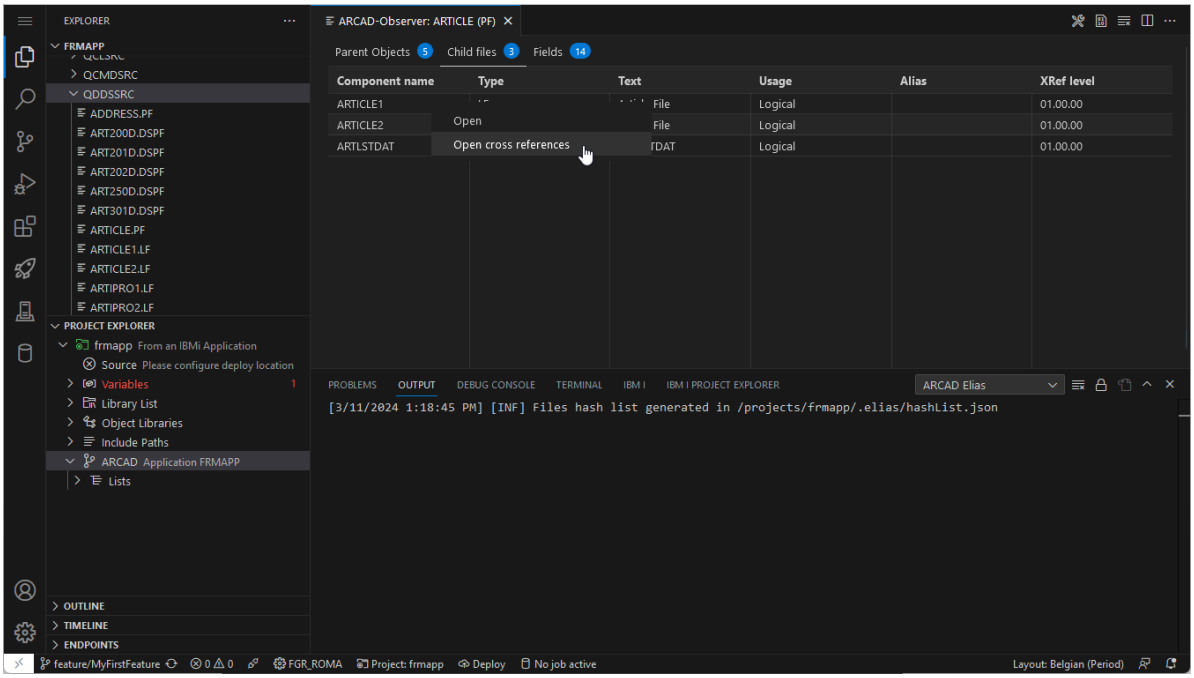
Links to files (Child files).

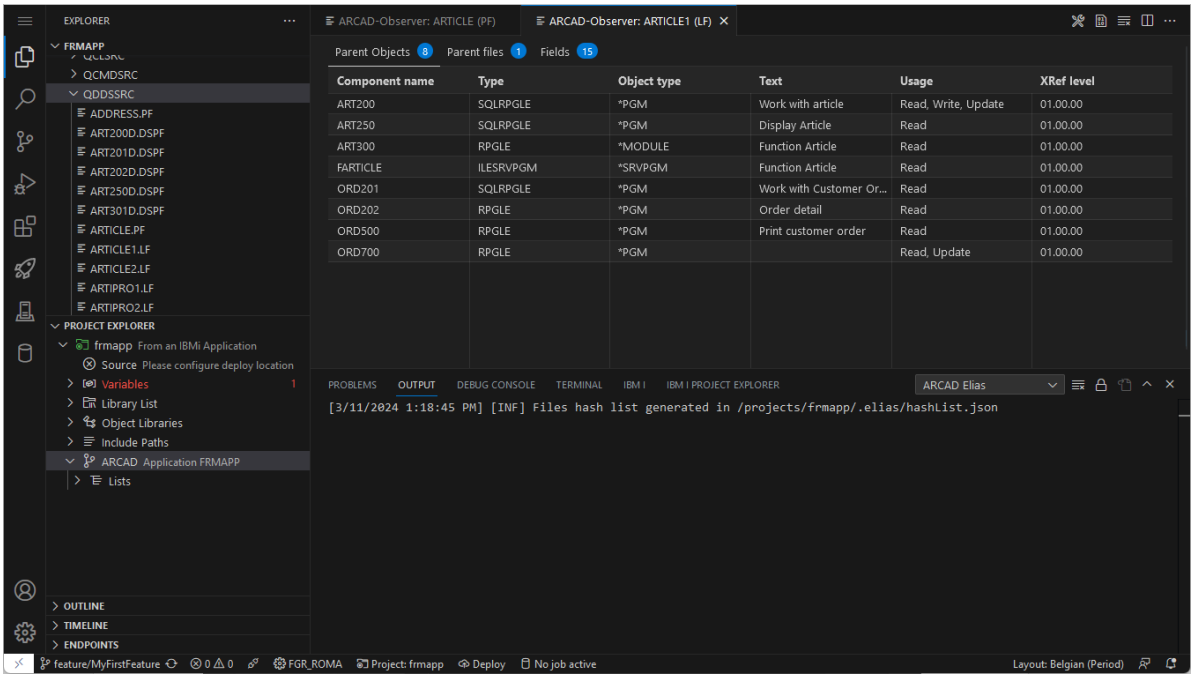


And links to fields.

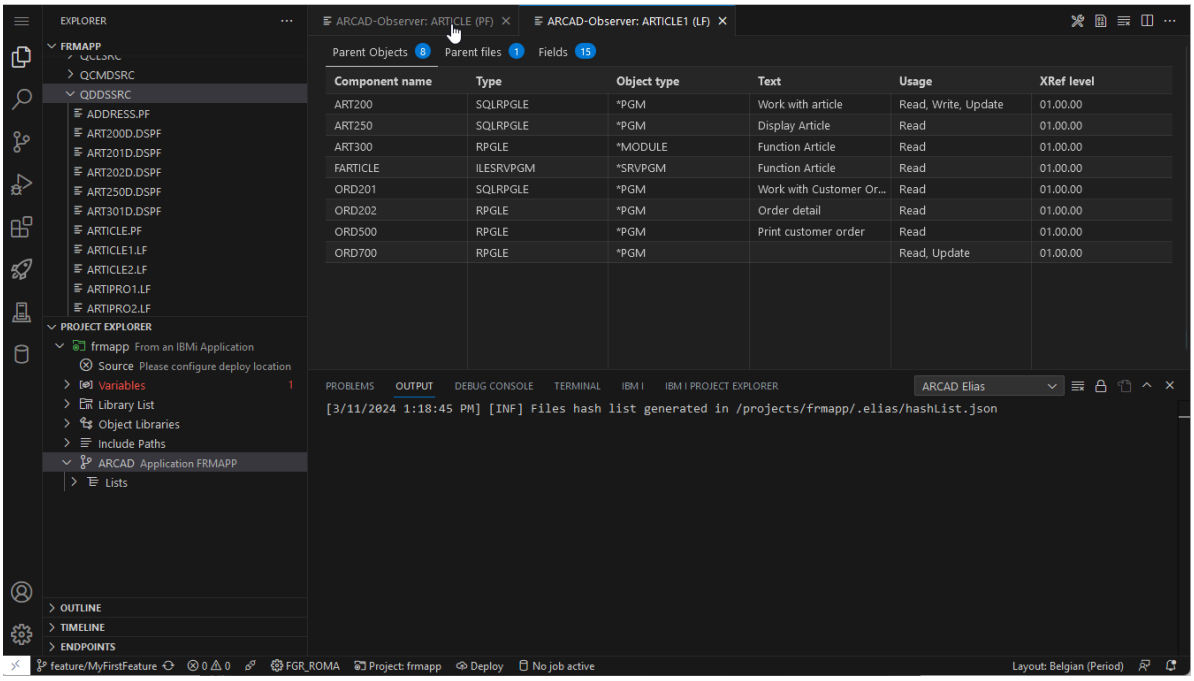


With a right click on a component, it is possible to follow links.

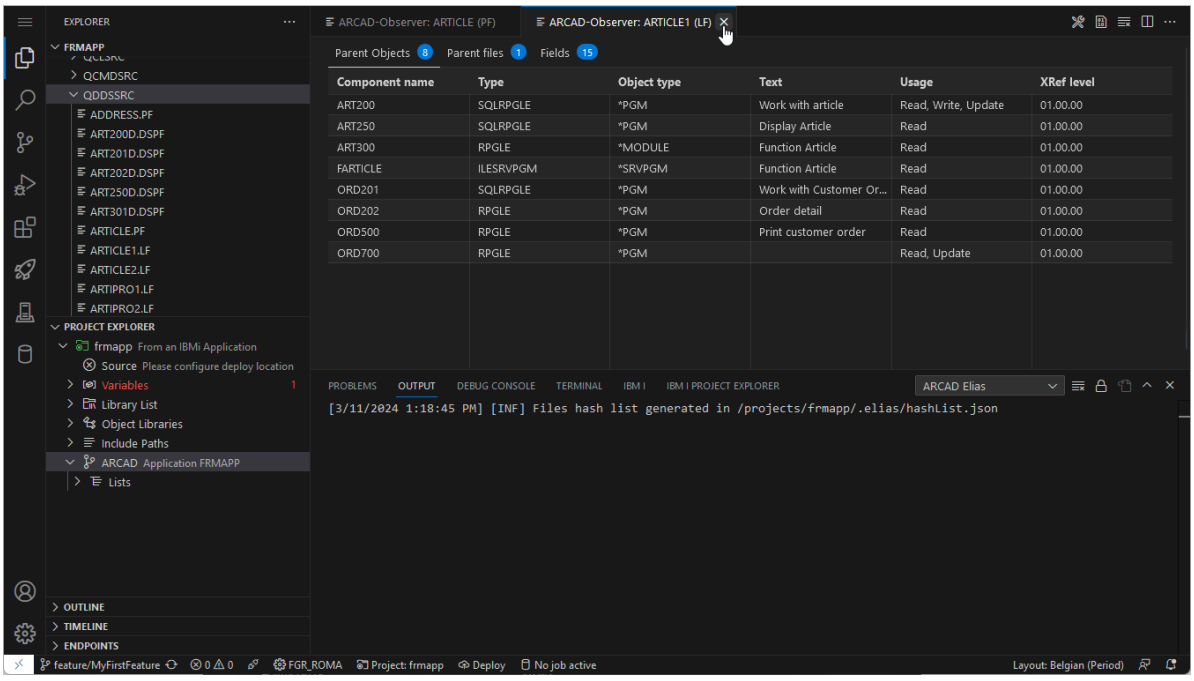




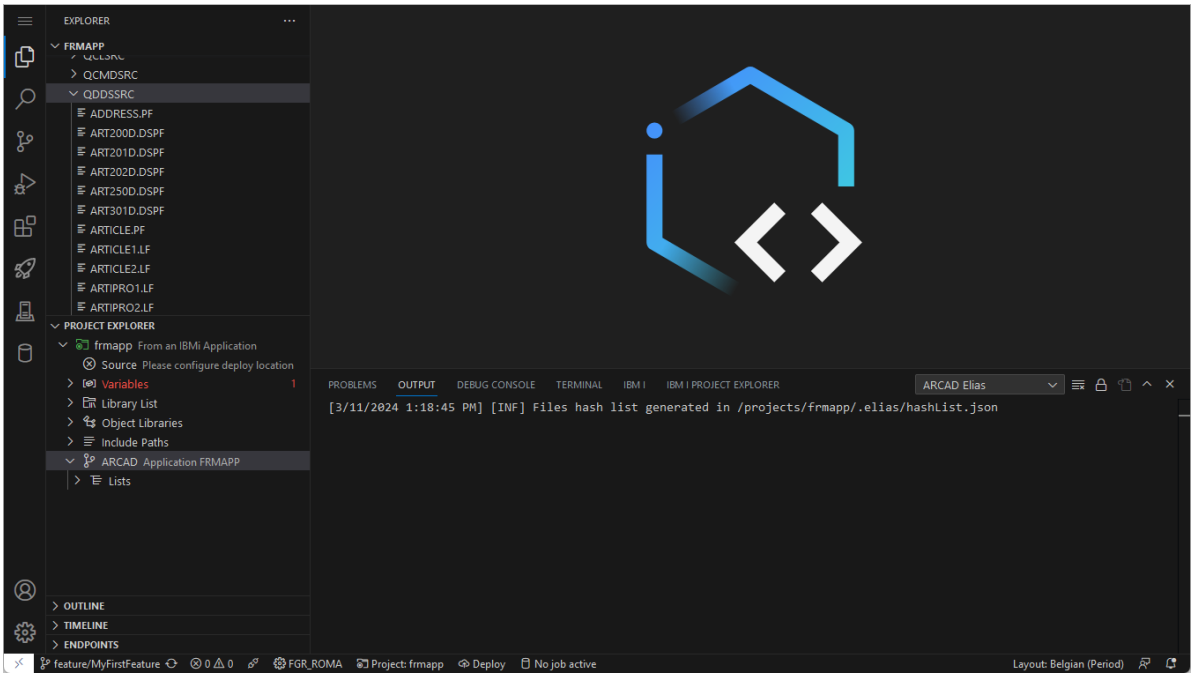
When you follow many links, it is possible to come back to a previous component with a click on its tab.



Close the view once you have finished navigating through it.

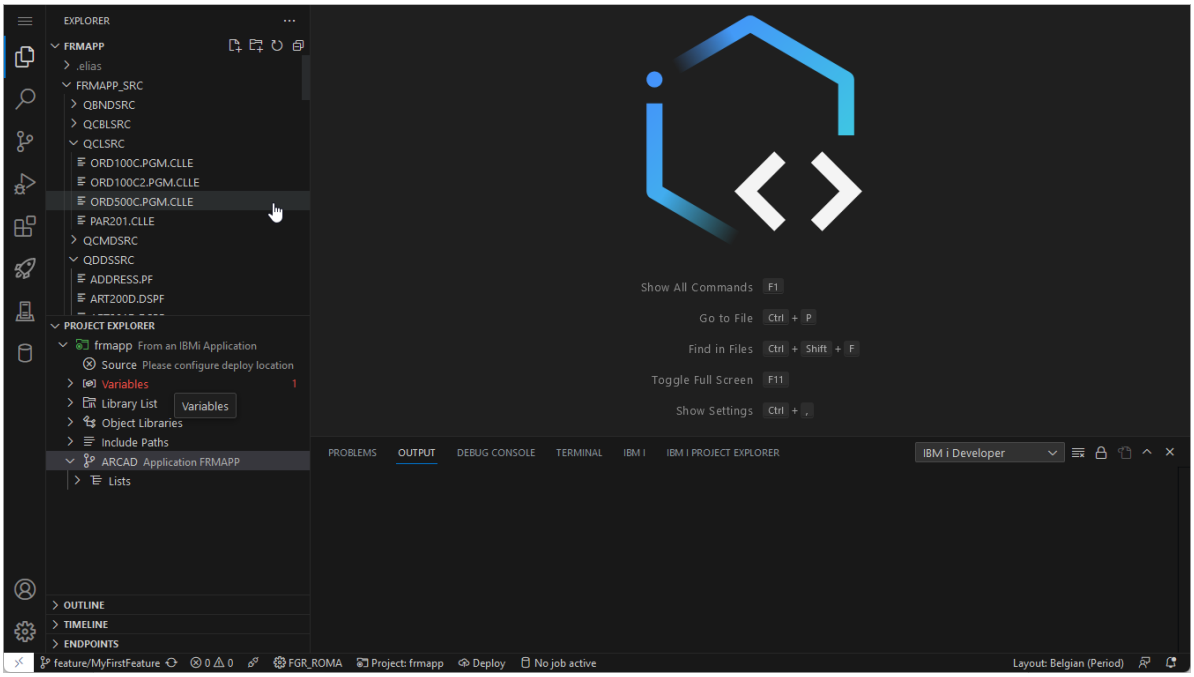


You can now make changes in the application.

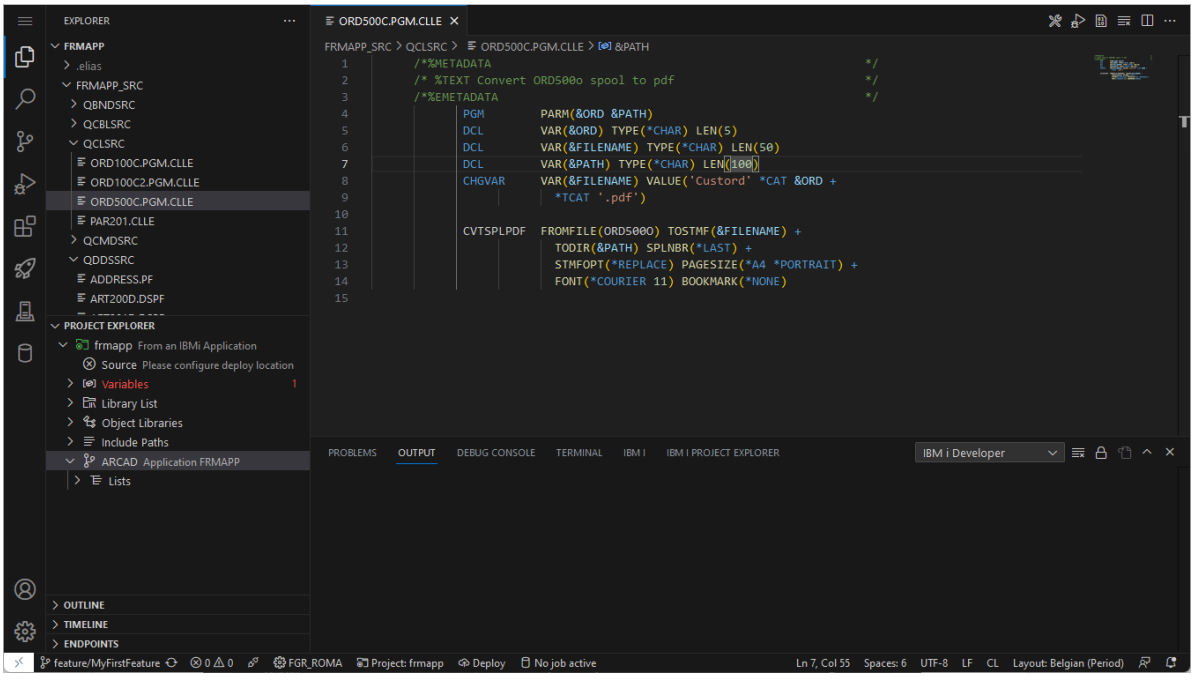


2.5.5 Modifying sources in MERLIN

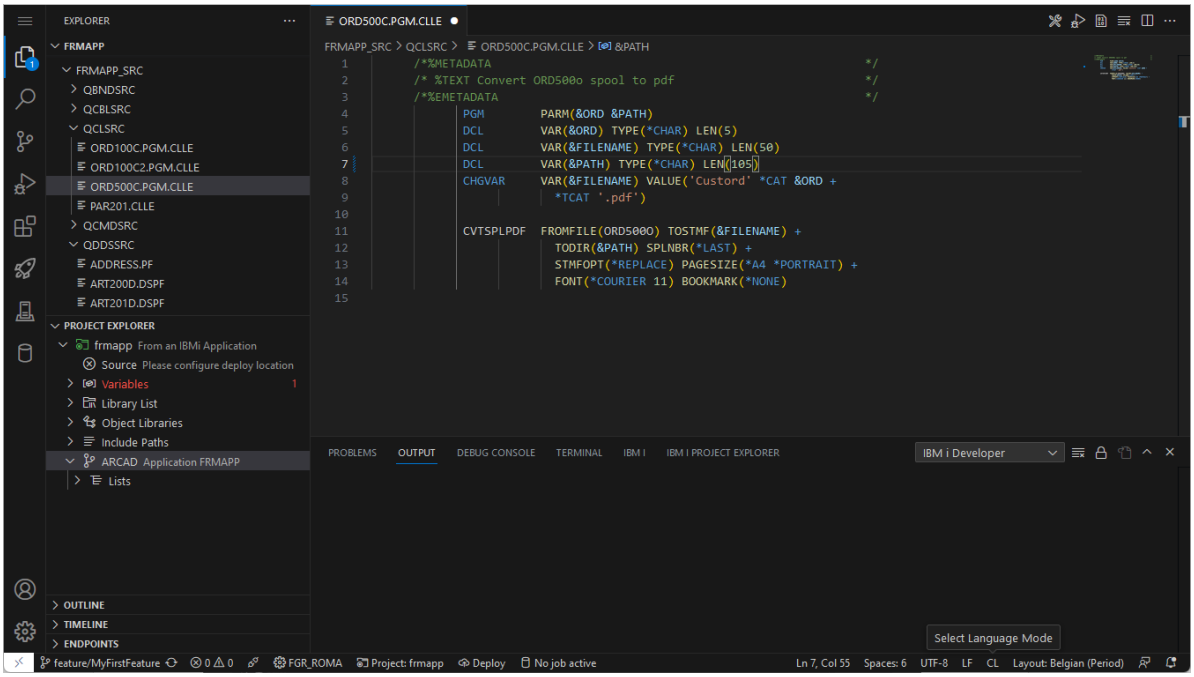
To edit a source, click on it.



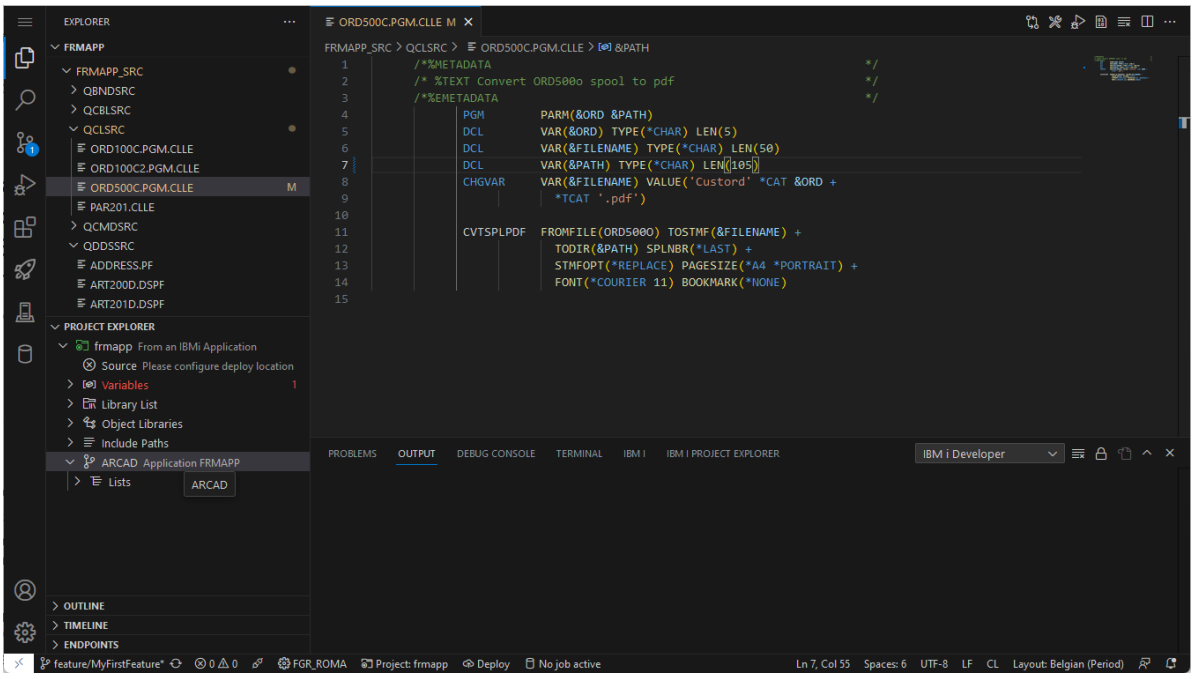
The editor view appears, so it is possible to put the cursor where you must do your modification.



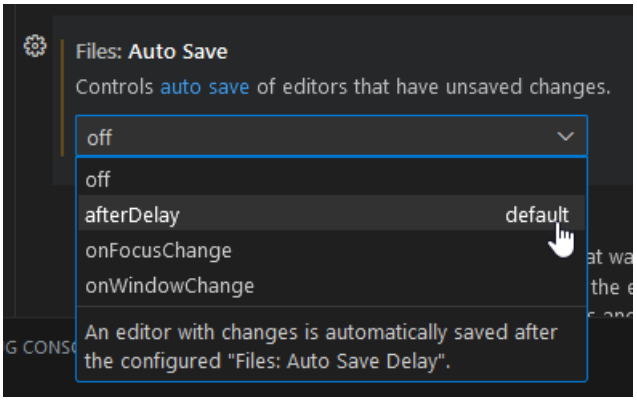
When a modification is done, the icon on the tab to the right of the component name is changed from the closing cross to a full circle and a badge appears on the Explorer icon.



Press Ctrl+S to save your modifications. The full circle disappears, the badge disappears on the Explorer icon, a badge appears on the Git icon and a M (modified) appears to the right of the component name and to the right of the folder names.



You will have this behavior if Auto Save in preference is off

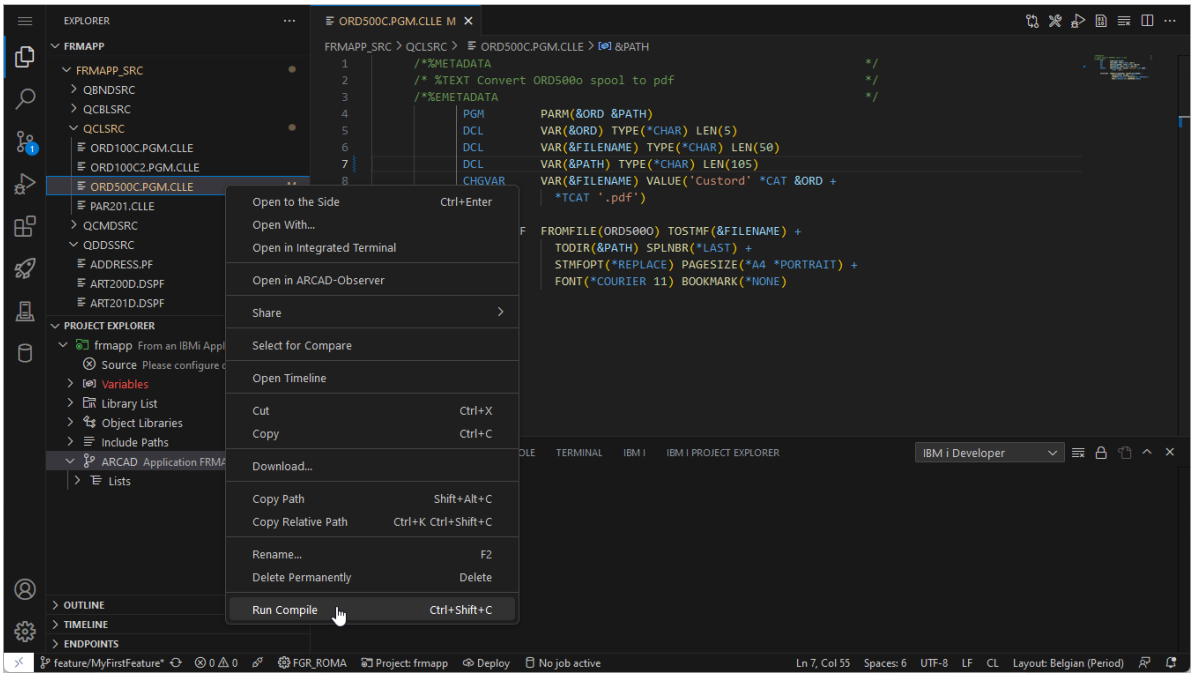


When a modification is done, it is possible to directly close the editor view and, in that case, you will be asked if you want to save the changes or not.

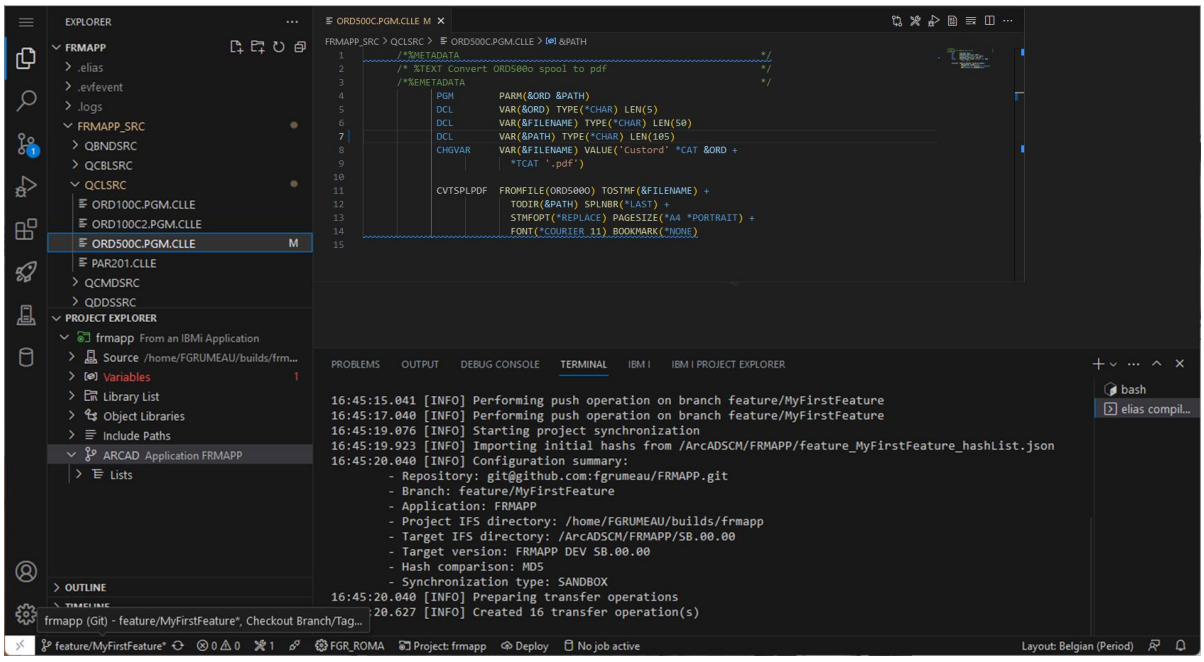
2.5.6 Compiling in a sandbox

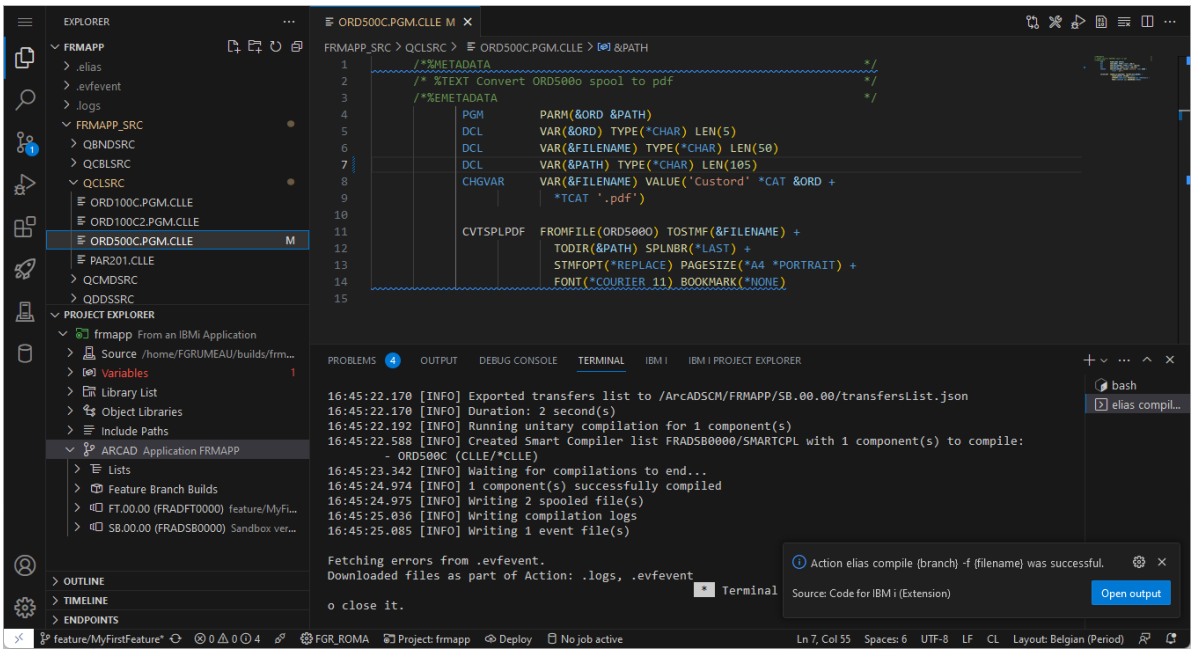
During a development, it can be interesting to compile a source member to check that the compilation works and to be able to do some unit tests.

To do that, do a right click on a source member and select **Run Compile**



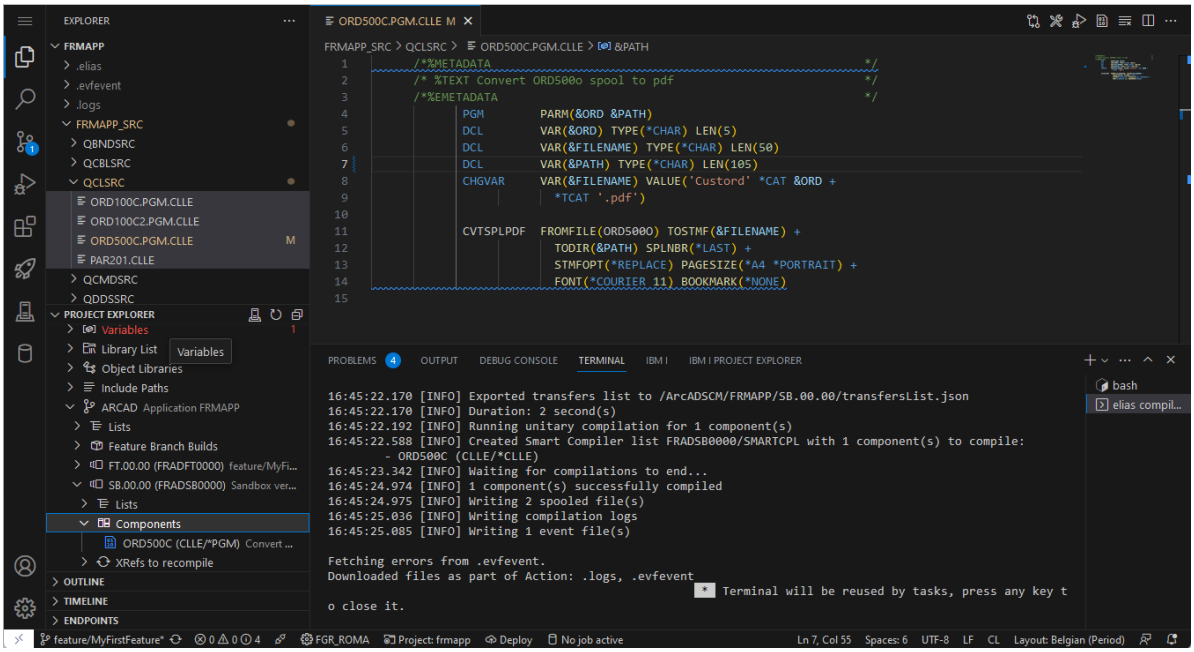
The compiling job starts.
It is possible to follow the process.
This job creates a new version, a sandbox version linked to the feature version only for the developer.
The compilation is done in this version.
And here, because the webhook was not added, the feature version was also created before.



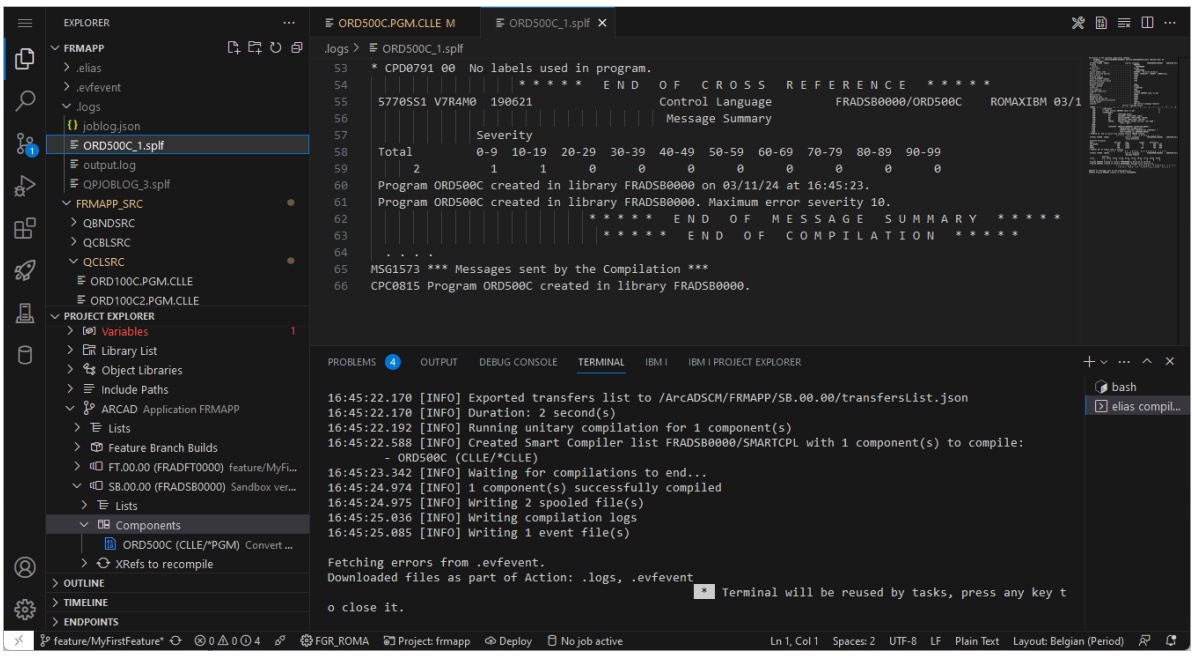


When the compilation is finished, many things appear in your IBM i Developer workspace.

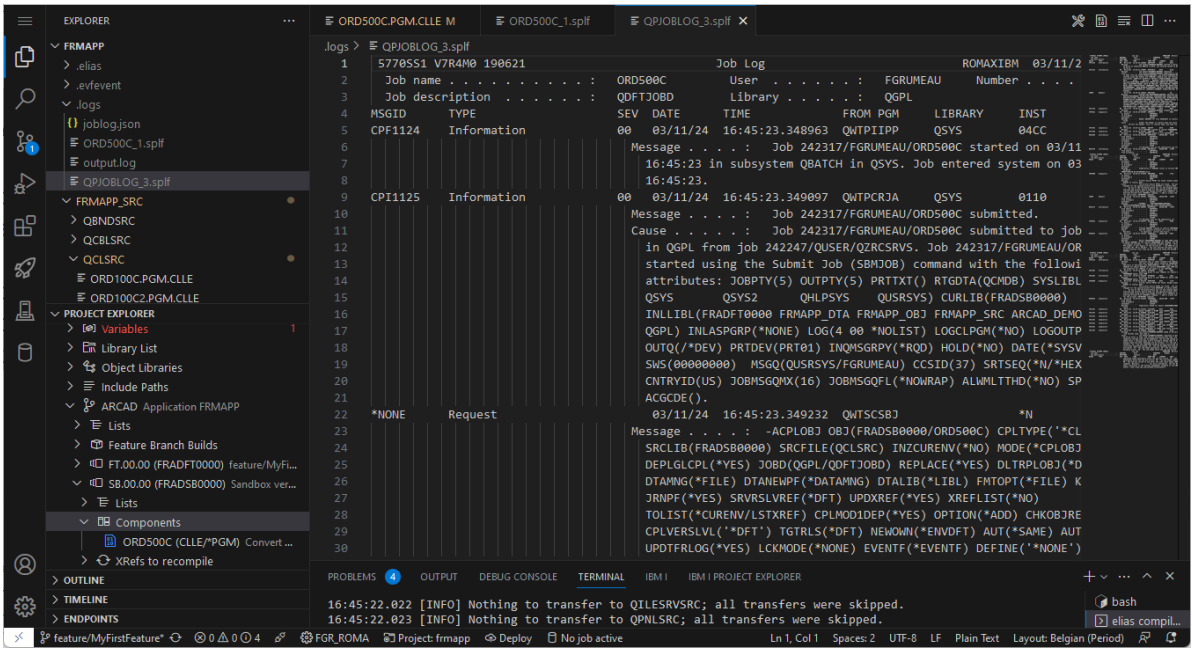
In the ARCAD part of the PROJECT EXPLORER it is possible to see the new versions (their names and the name of the library linked) and the compiled object.



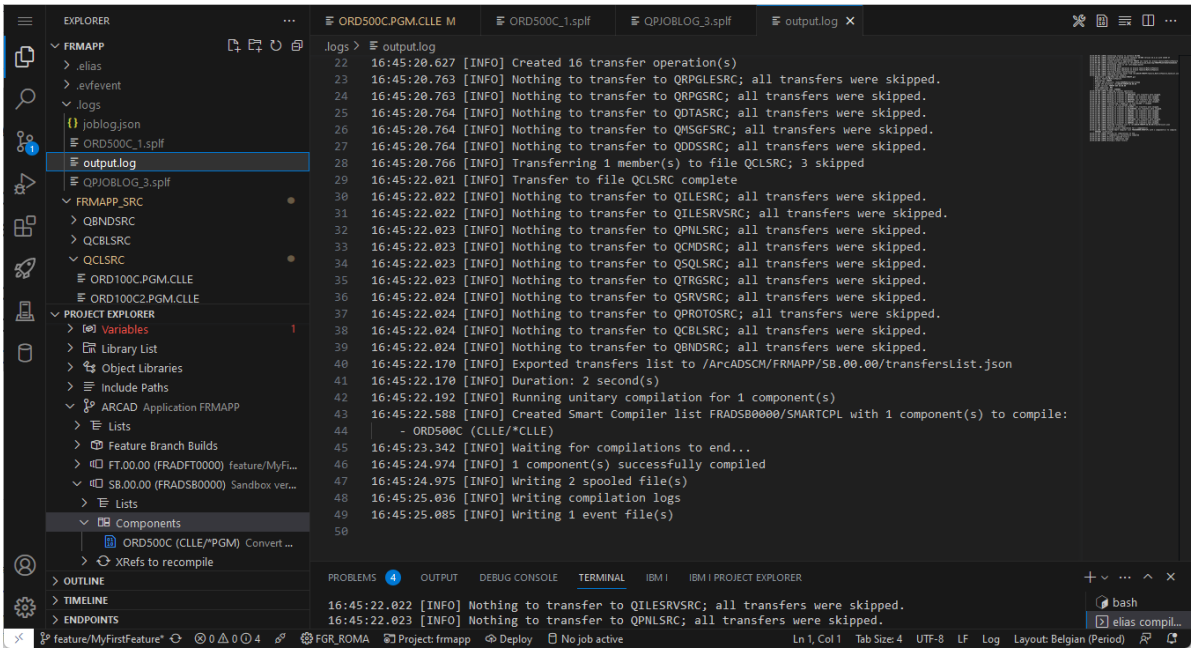
In the PROJECT Part, it is possible to see the creation of “logs” files.
You can display the spooled file of the compilation.



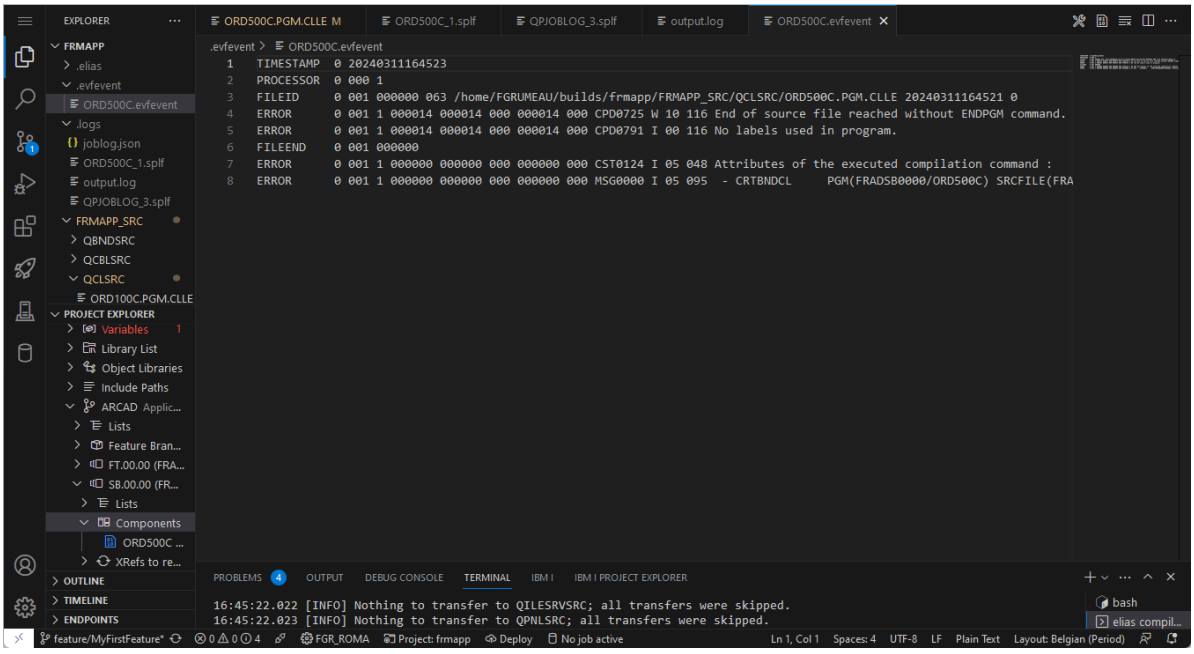
You can display the joblog of the compilation job executed in the IBM i.



You can also display the output.log of the compilation job.

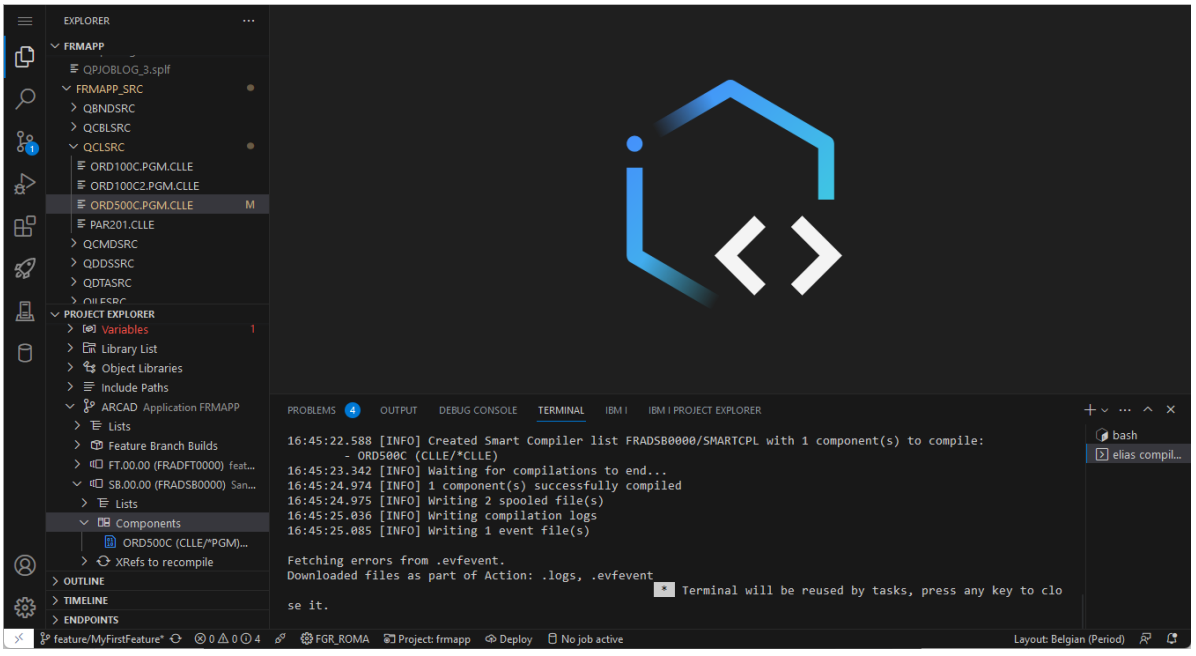


In this example, the compilation of this source member generates a warning.
You can display the evfevent file to see the cause of this warning.



These files are important when a compilation fails.

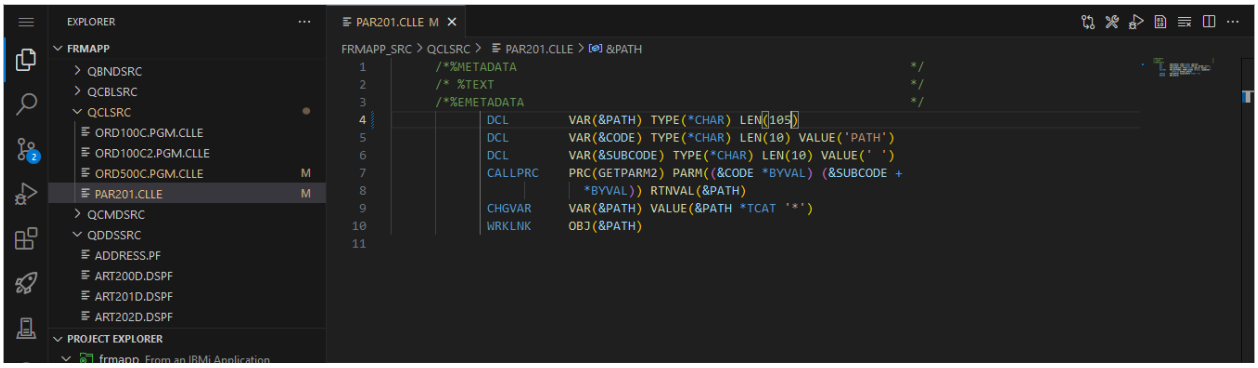
It is possible to make changes in the source member and relaunch a compilation process in case tests fail.
Here, you can just close all tabs.



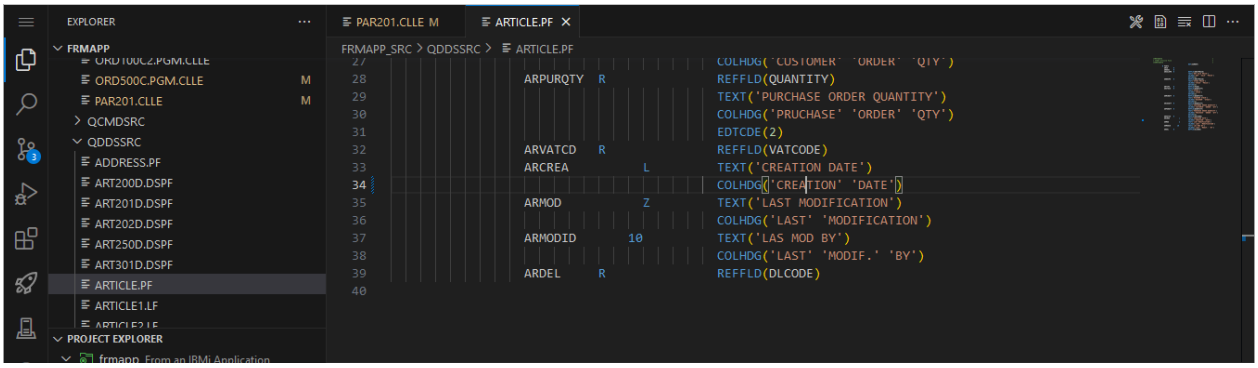
2.5.7 Building in a sandbox

In this section, we edit two other source members, and we add a modification, but this time, we will launch a build.

Edit the CLLE PAR201 to change the length of the PATH field if you use the ARCAD Demo application for this getting started.

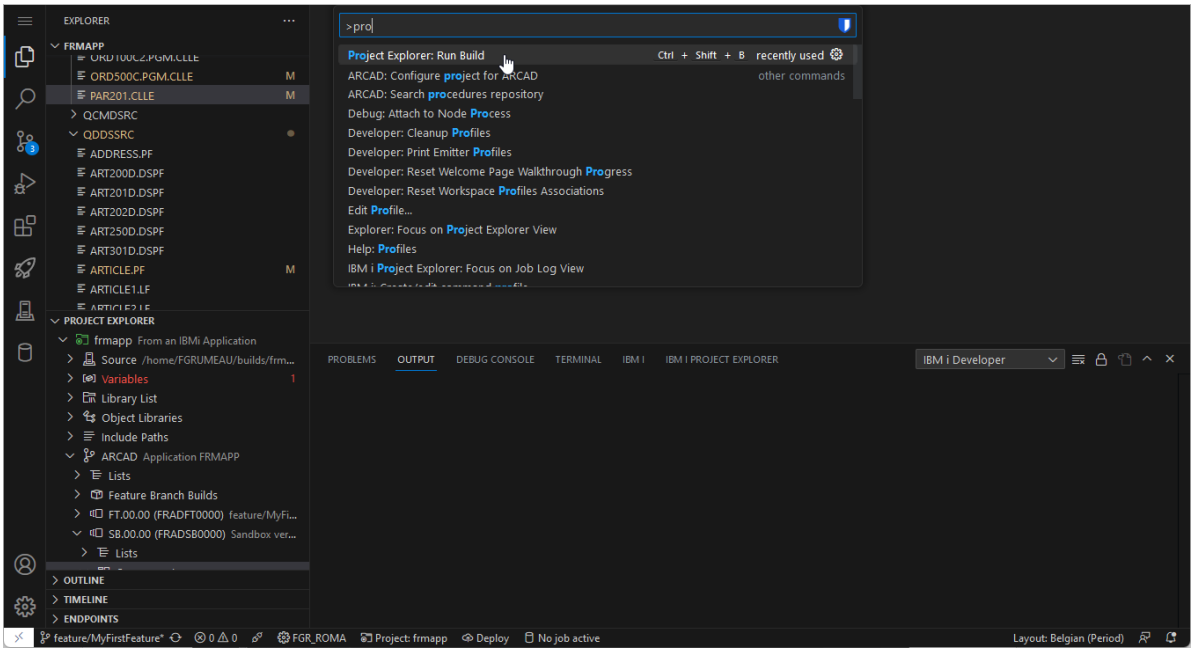


And edit for example the PF ARTICLE to change the error on the COLHDG for the ARCREA field if you use the ARCAD Demo application for this getting started.



Close the two tabs and save the changes if asked.

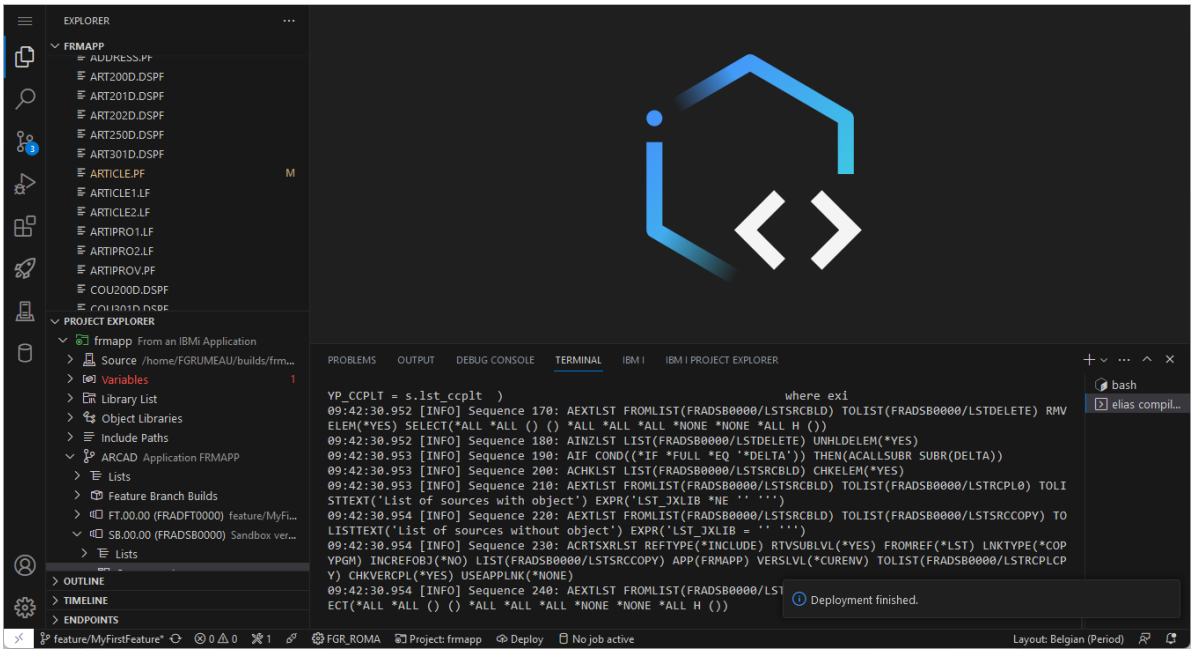
Now press F1 and select the **Project Explorer: Run Build** option.



The build job starts.

As for the compilation, it is possible to follow it.

This job uses the same sandbox version (it is created if it does not exist already) and this time, the job will compile all the modified source members and all the components impacted by the change in the PF (logical, linked programs).

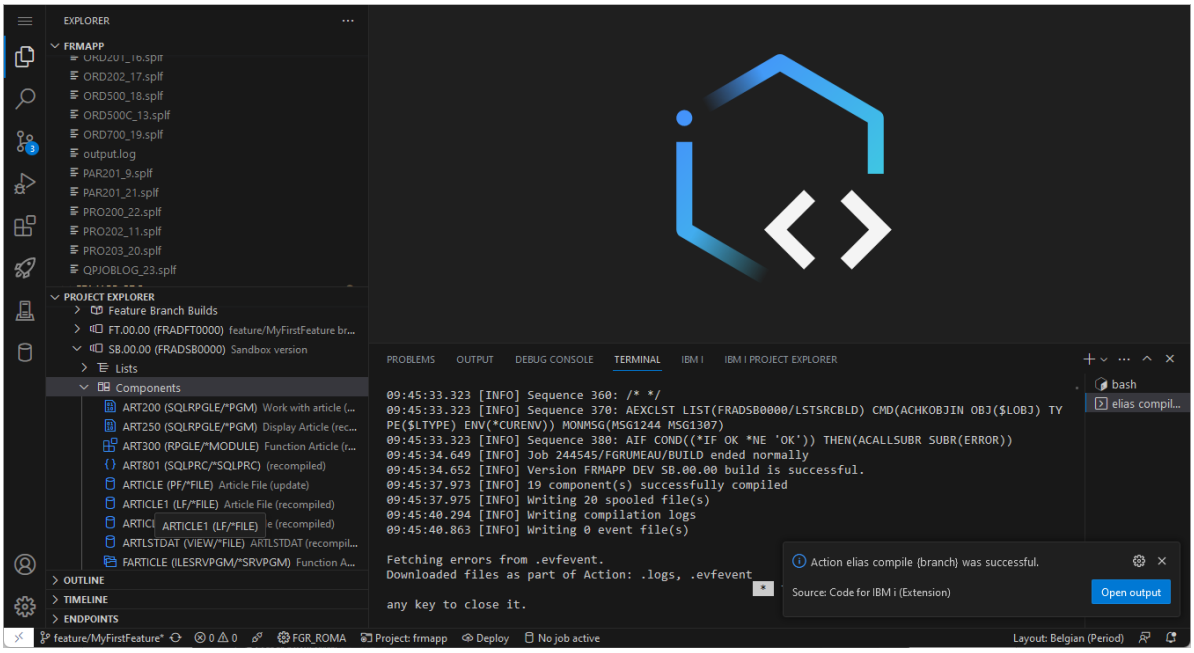


When the build is finished, many things appear in your IBM I Developer workspace like for a compilation.

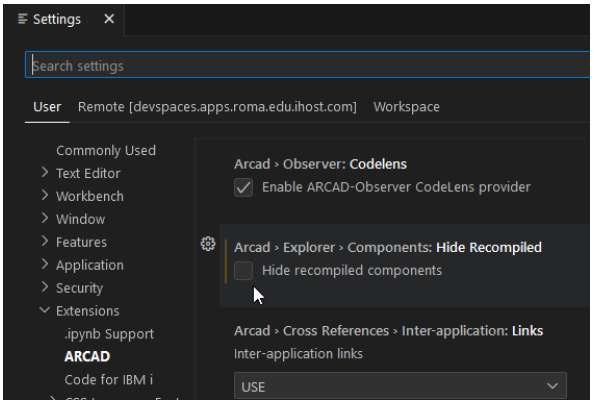
It is possible to display, if needed the spooled files, the joblogs, the evfevent files, etc.

And if you go to the ARCAD part of the PROJECT EXPLORER, for the sandbox version, you will see more than three source members recompiled, those with the information Update.

You can also see all the automatic recompilations done by the build, (those with the information recompiled).



You will see all recompiled component only if the option **Hide recompiled components** in Settings is unchecked.



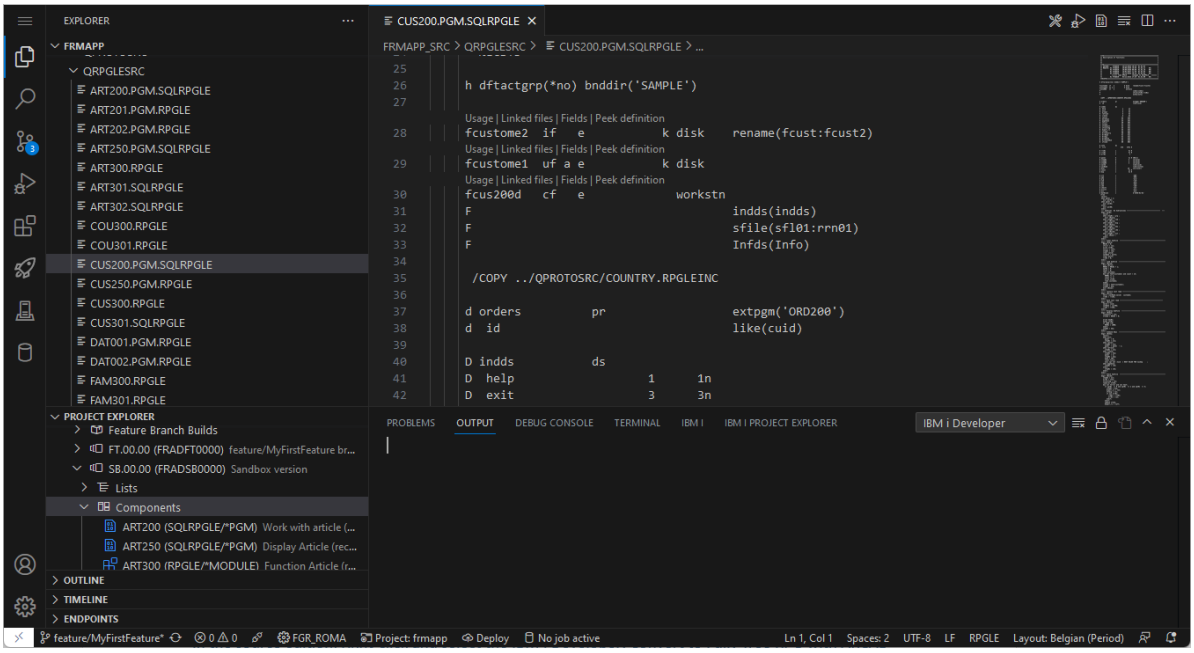
Like for the compilation, all the modified components and recompiled components are in the version library. It is possible to do unit test and make additional changes if needed.

Here, we just go to another feature, the capability to convert an RPGLE in free format.

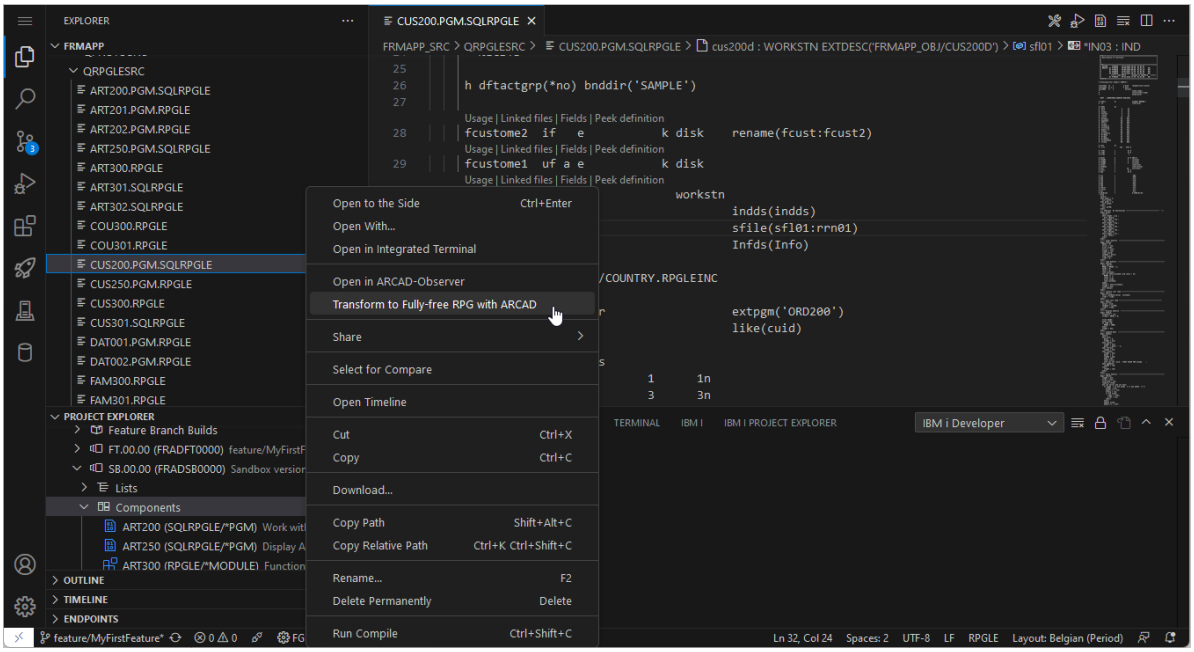
2.5.8 Free RPG transformations

In this section, we will edit a (SQL)RPGLE to convert it into fully-free format.

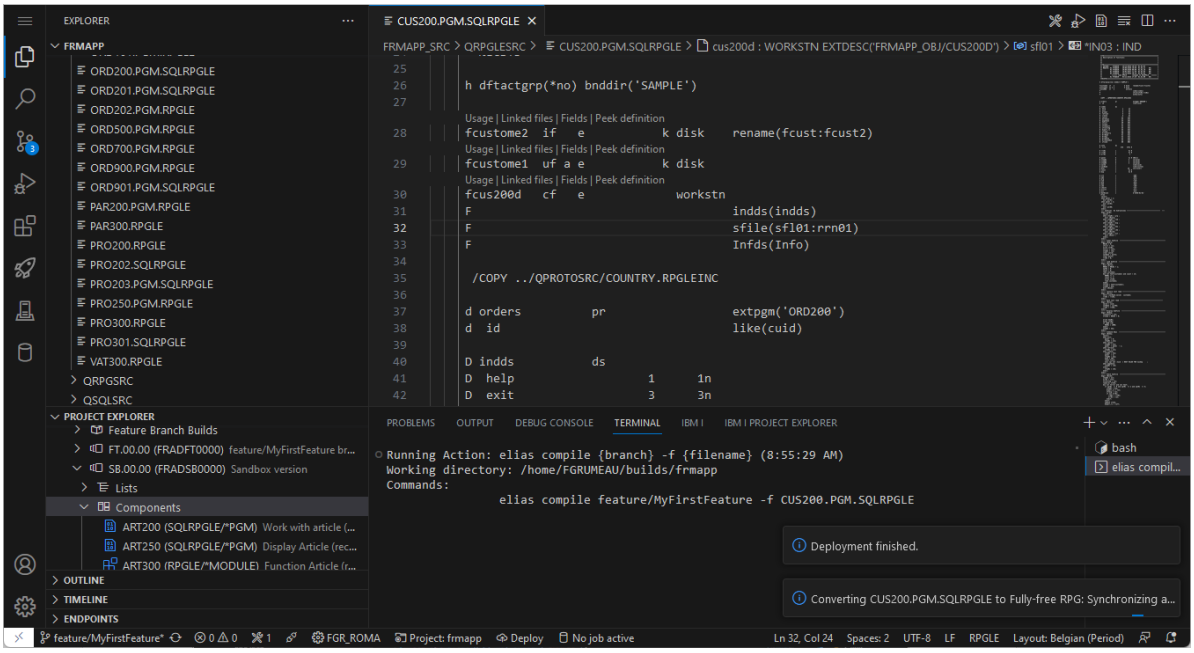
Edit for example the RPGLE CUS200.



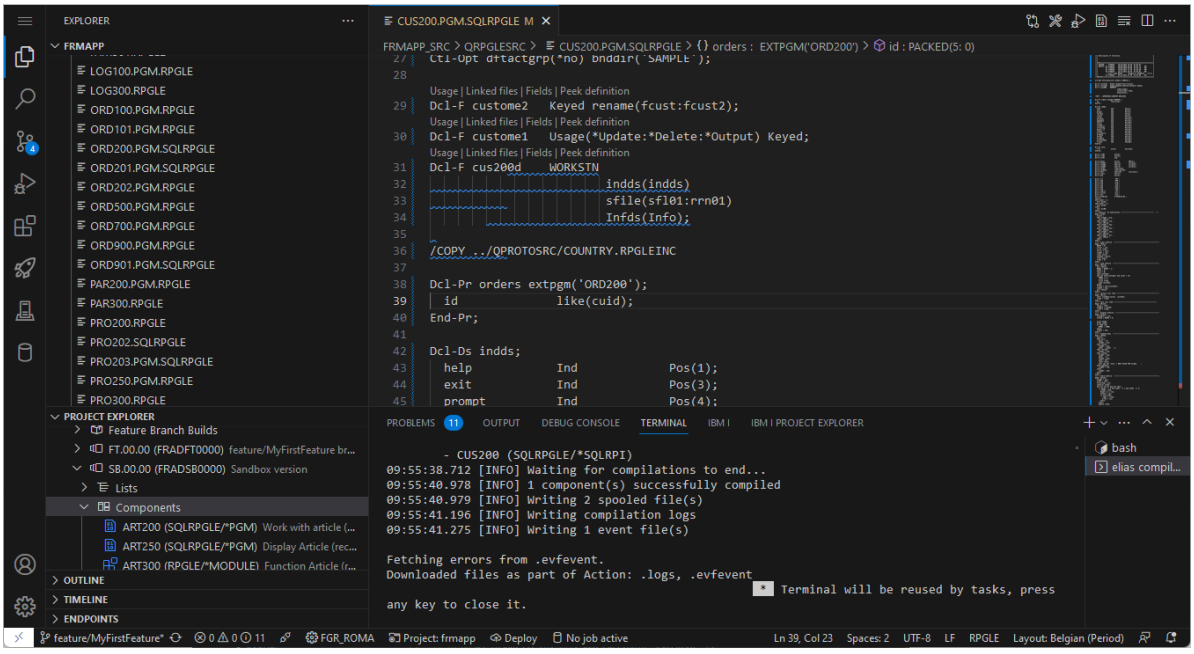
On the source member or in the source edition, do a right-click and select the **Transform to Fully-free RPG with ARCAD** option.



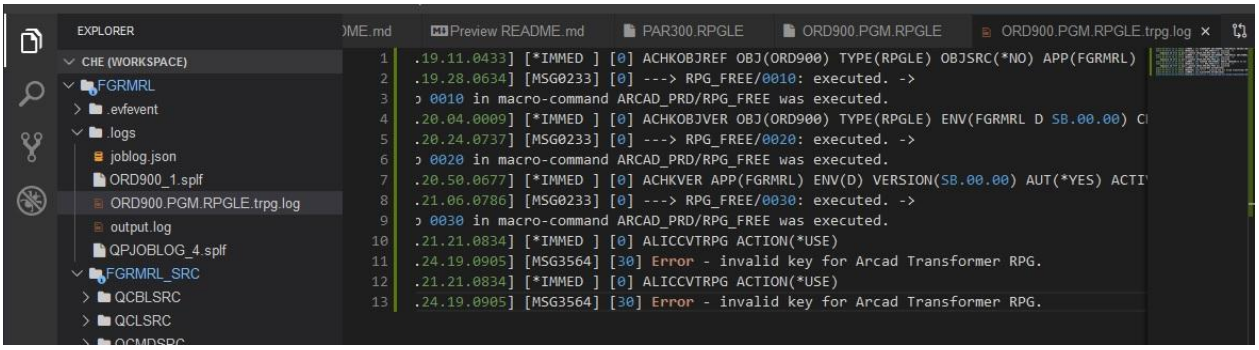
The conversion starts.



When the conversion is finished, the editor view is automatically refreshed.



After the conversion, the component is automatically recompiled. So, in case of an error, check the compilation log for more information, and in case of a problem during the conversion, a conversion log is generated with the**.trg.log** extension.

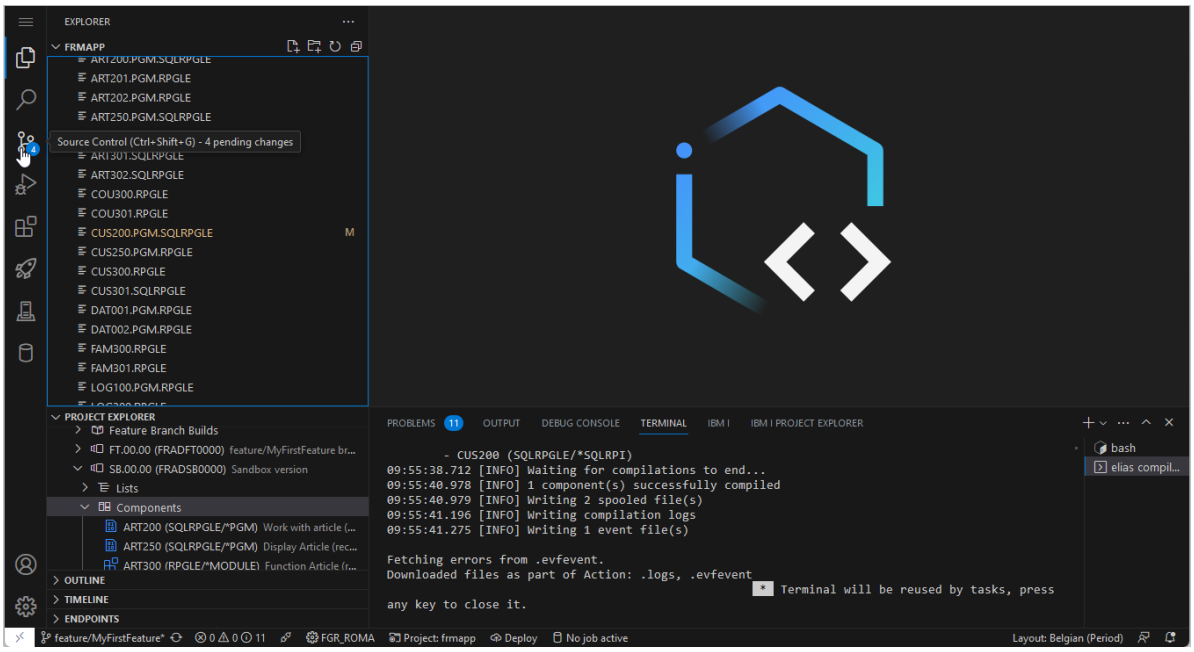


In this example, the conversion is successfully completed, and all modifications are done, so now we can close all tabs and build the feature.

2.5.9 Commit and push

When all the modifications are done, you need to commit them and then to push them to the Git repository, before launching the building process.

To do that, click on the **Source Control: Git** icon.

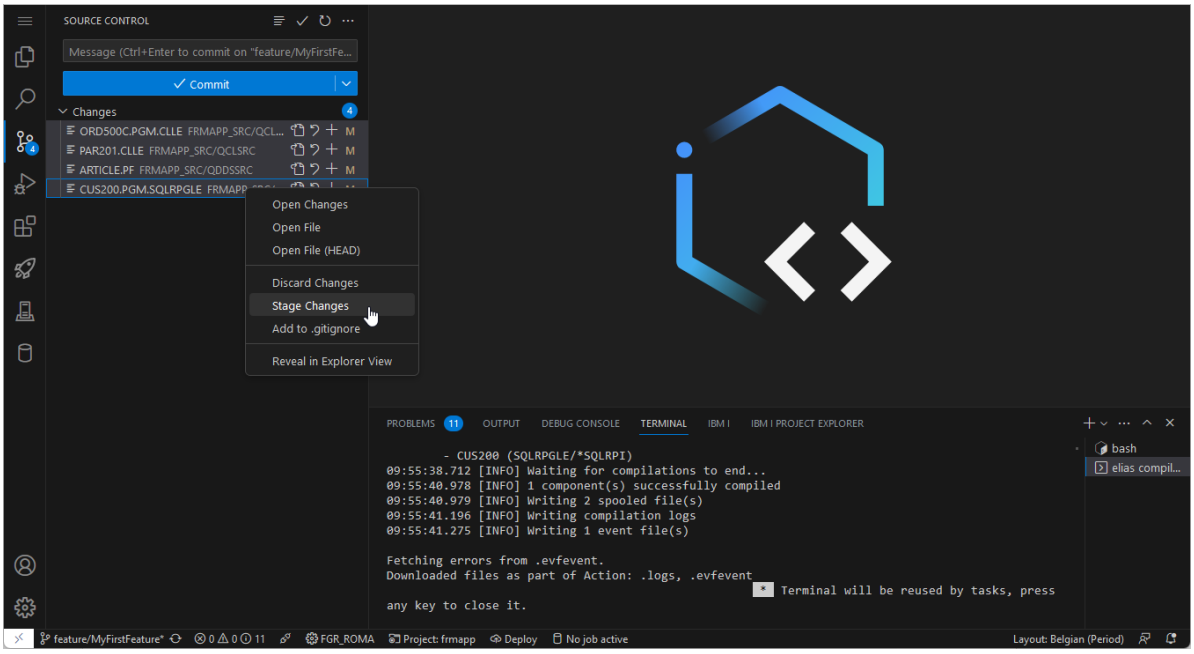


You will see all the modifications.

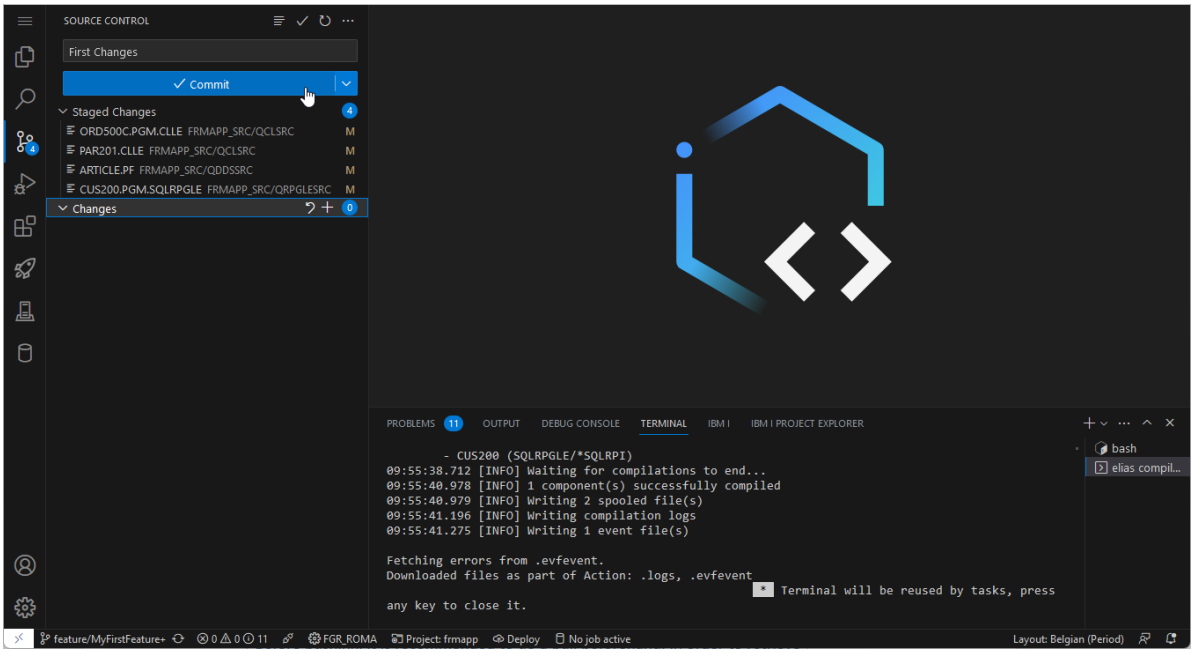
You can stage them one by one with a right-click or by using the **+** icon.

You can click on a source to display the difference.

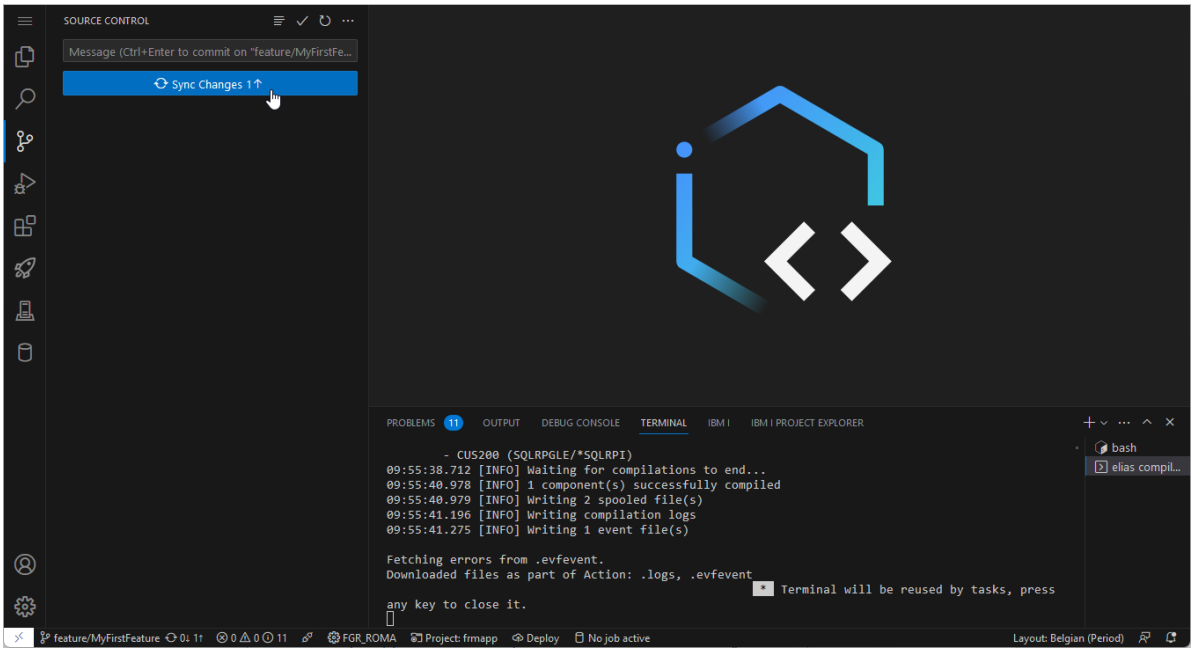
But here just select all source members, right-click and select the **Stage Changes** option.



Now you can add a commit message and launch the commit.



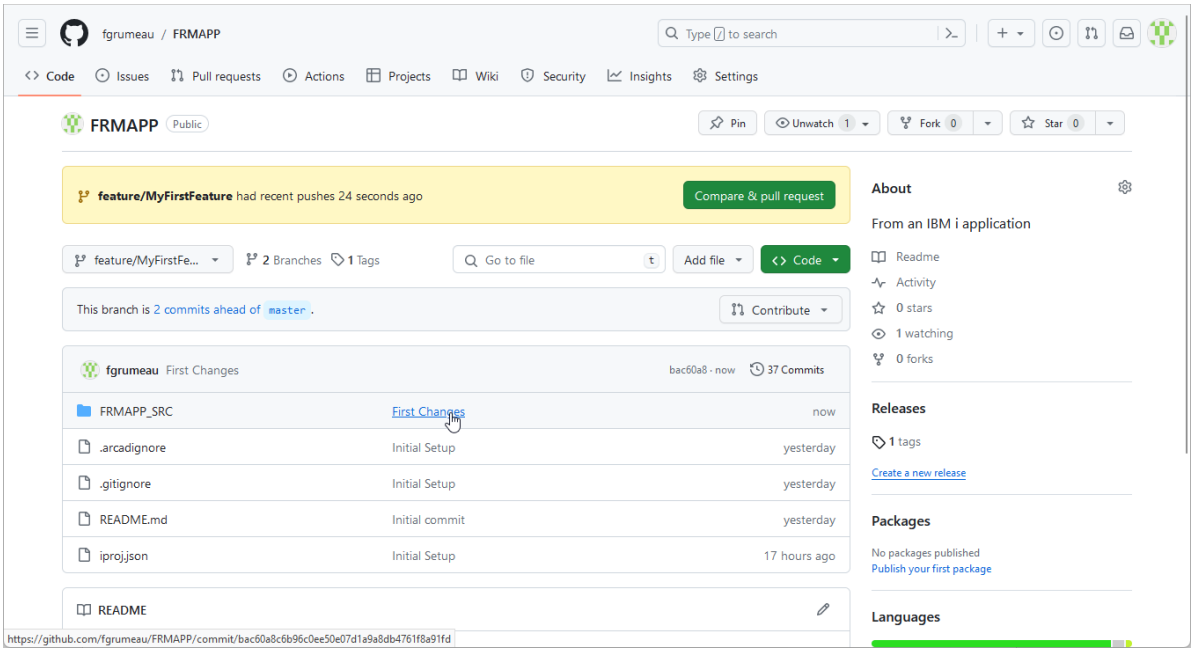
Click on **Sync Changes** to send the changes to the Git repository.



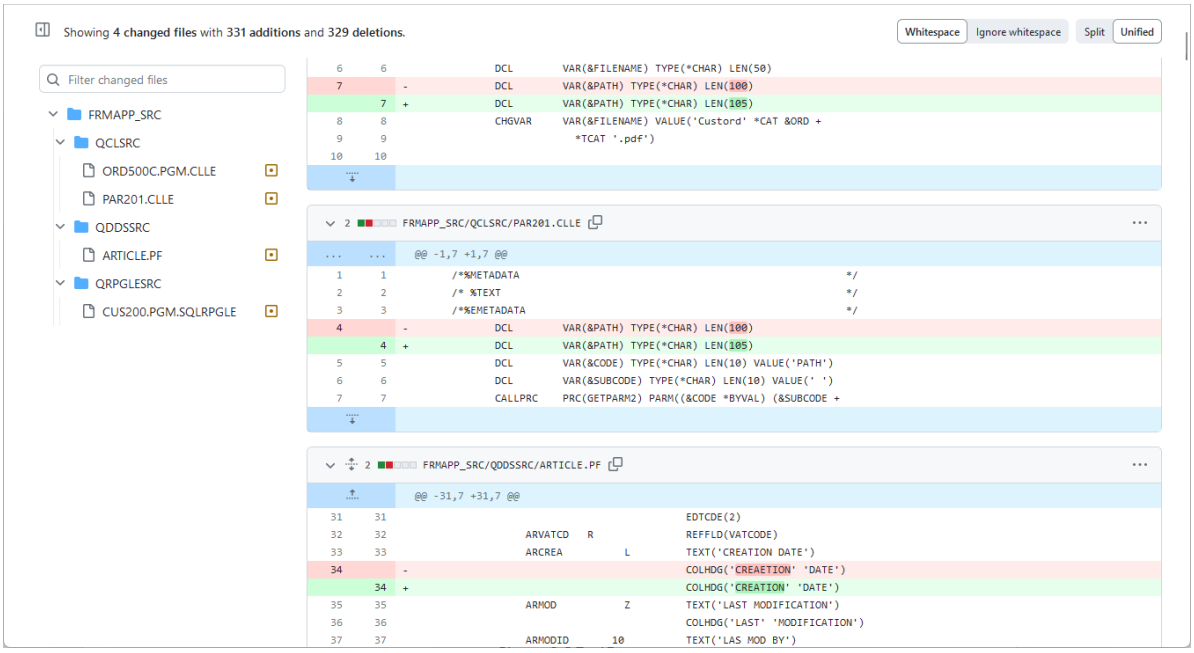
Before pushing, it is recommended to do a pull beforehand, in order to retrieve the commits of other developers working on the same feature.

And after the push, you can also do a pull to retrieve the last commit.

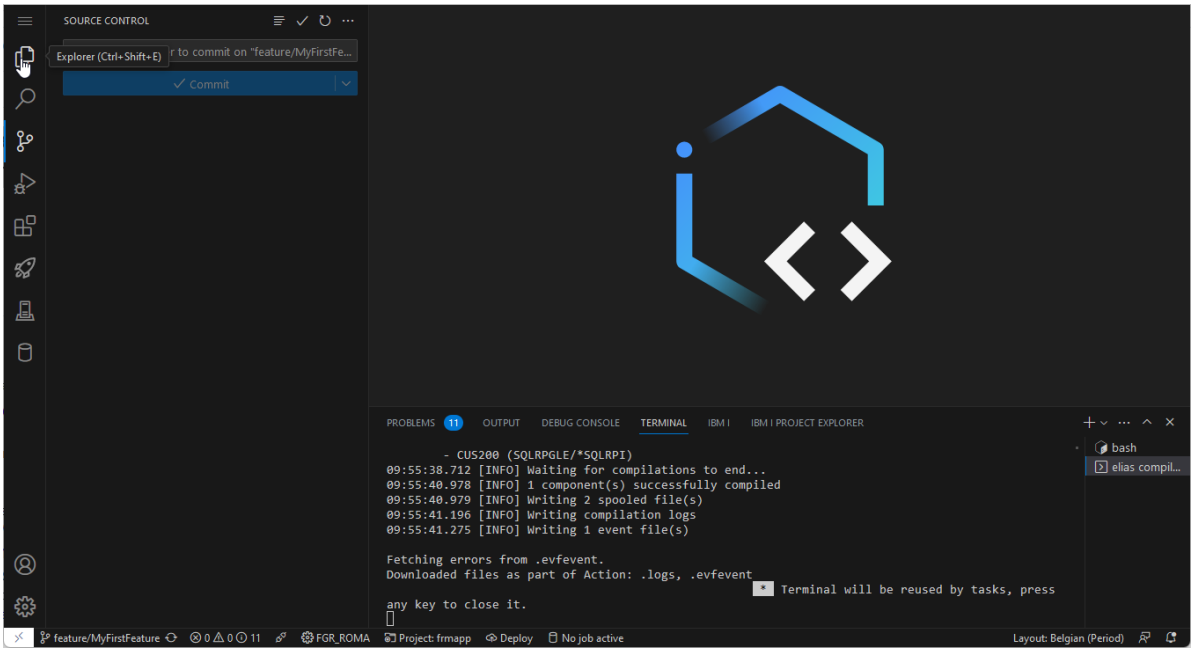
When the push is finished, it is possible to see the commit done from the IBM i Developer workspace in the Git repository on the feature branch.



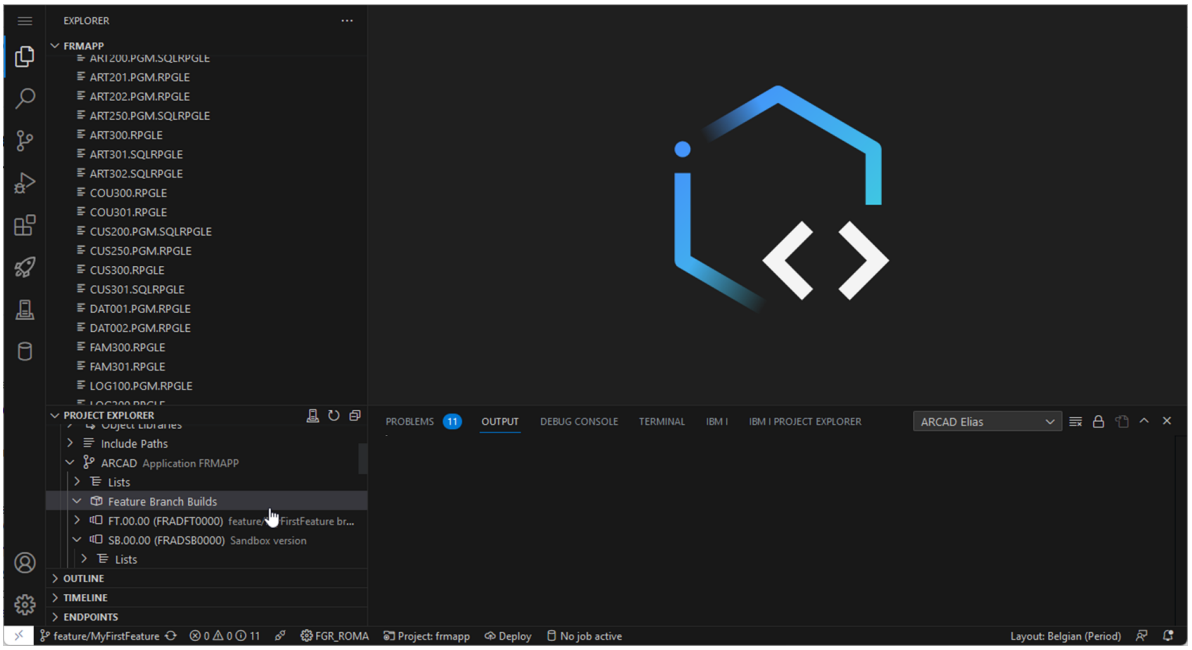
And if you click on the commit message, you will see the changes made.



To launch the build of the feature, click on the **Explorer** icon.

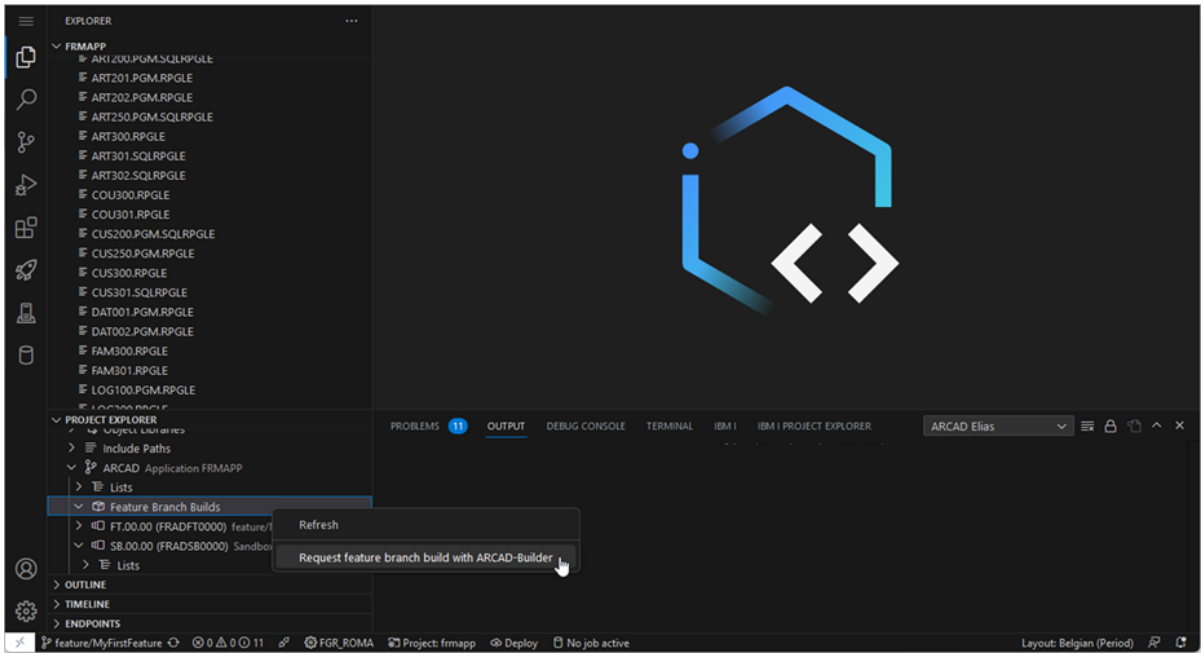


Select the **Feature Branch Builds** node on the PROJECT EXPLORER part.



2.5.10 Building your version with ARCAD Builder

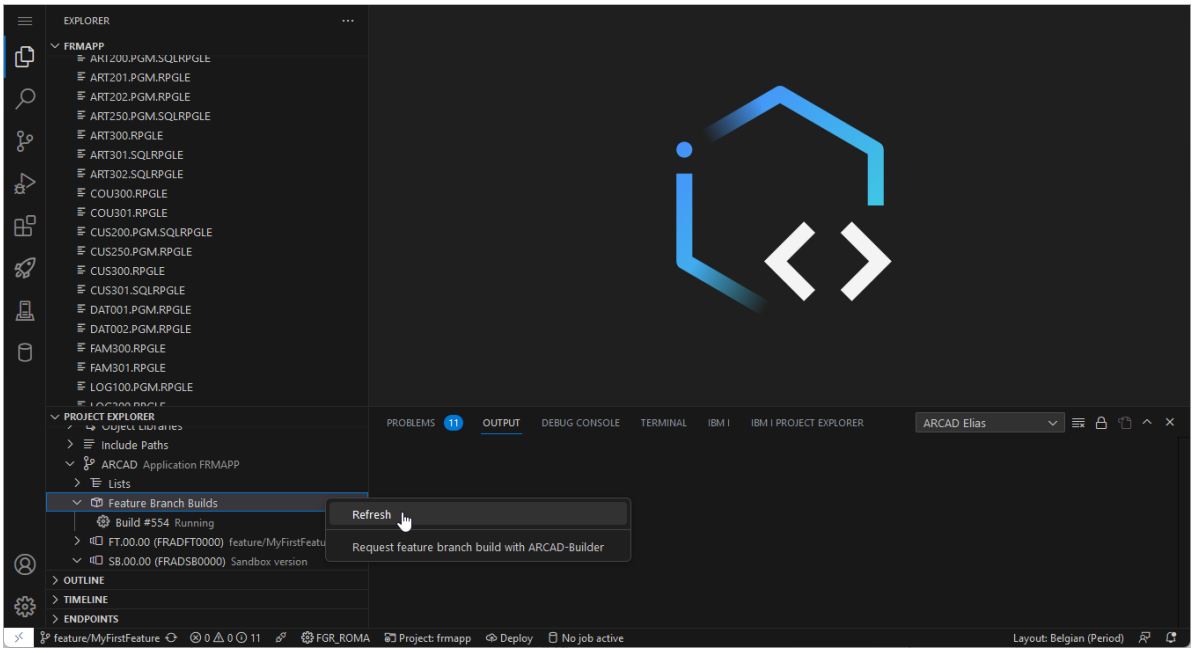
To launch a build of the feature with ARCAD Builder, do a right-click on the **Feature Branch Builds** and select the option Request feature build with ARCAD-Builder.



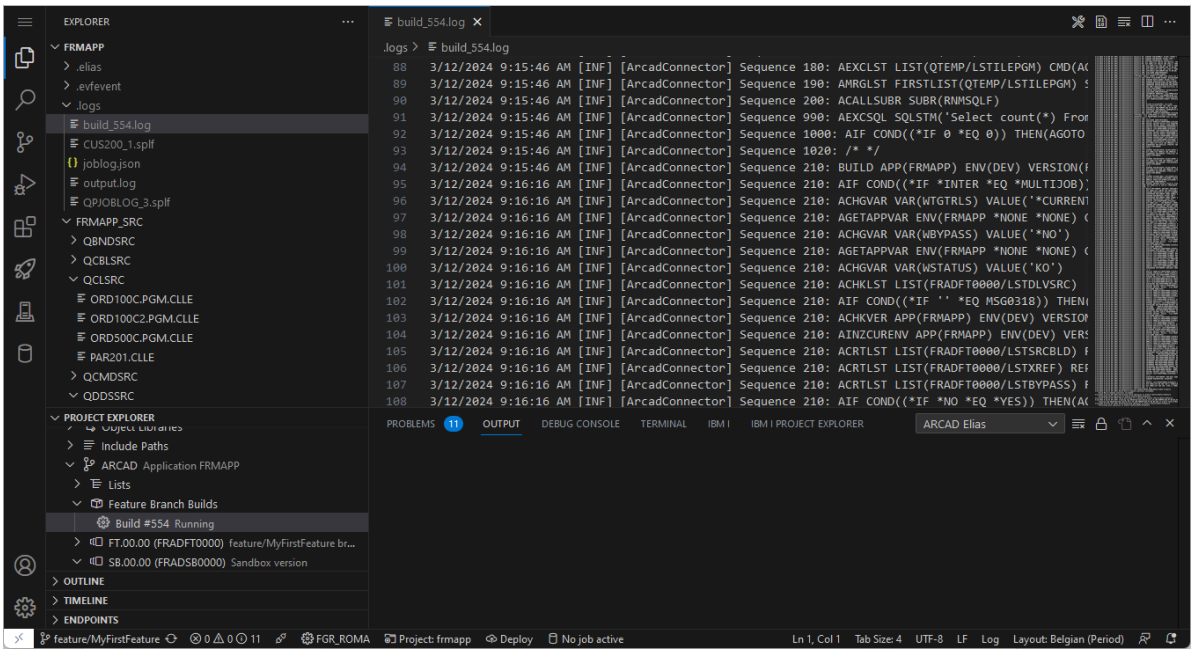
You can also use the ARCAD Builder Client to do that manually, or if you check the Enable webhook support option for pushCommit operation, the build is launched automatically, and you can use your IBM i Developer workspace to follow the process.

After using the option, you can explore the **Feature Branch Builds** node and see a running build.

It is possible to refresh the information by doing a right click and select the option **Refresh**.



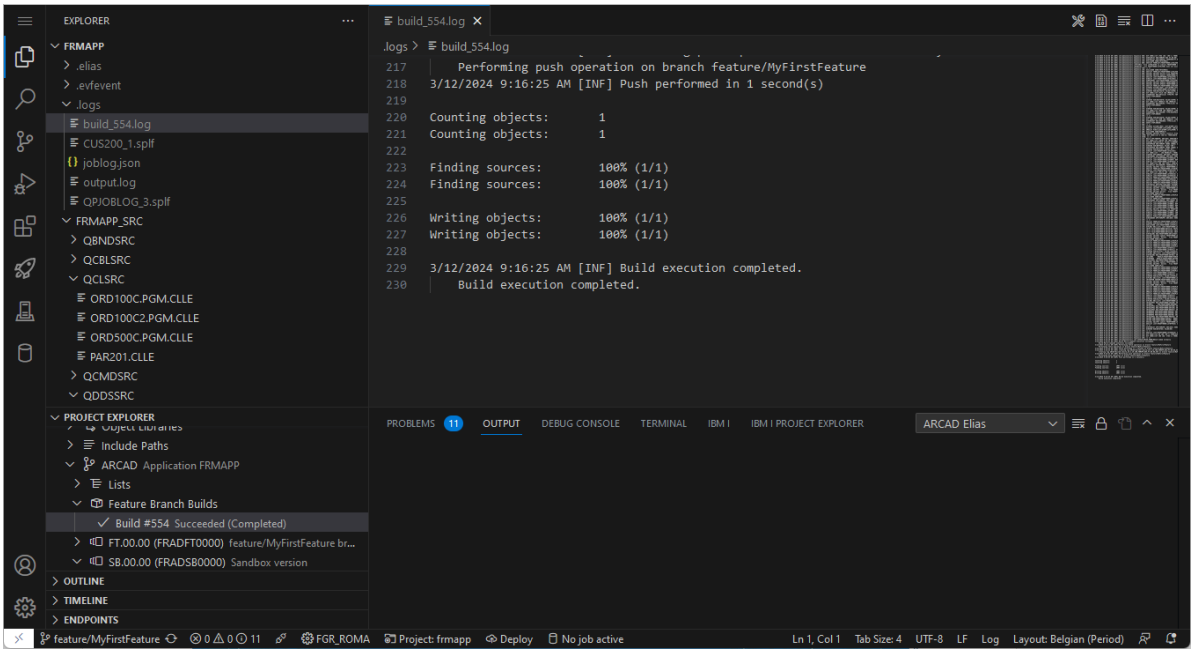
You can click on a running build to display the running log



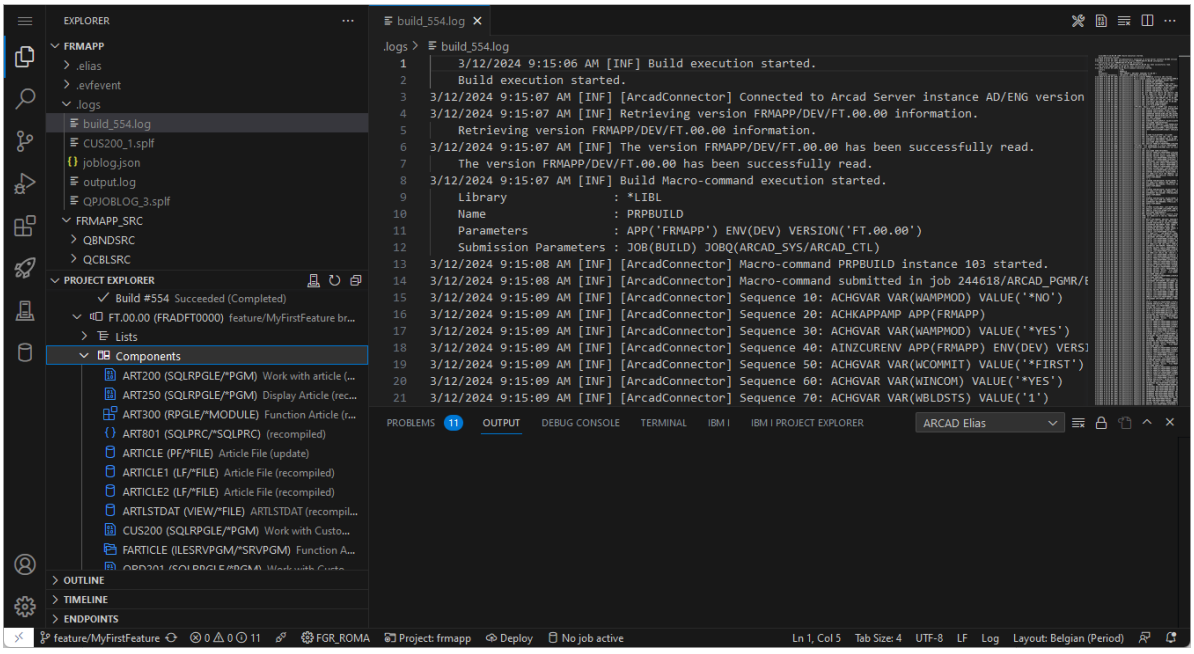
This log is important if there is a problem during the build.

When the build is finished, you can have the result directly on your workspace.

The workspace displays a **Failed** message if the build fails, and a **Succeeded** (completed) message if the build is completed successfully. If there is nothing to build, it displays an **Empty** message (this happens when you launch the build without having pushed new changes).



If you click on the **Components** node for the feature version, you will see the components of the feature.



2.6 The next step – The release branch

The next logical step is to create a new release branch, that will create automatically a new ARCAD release version.

Then it will be possible to merge some features into this release and launch an ARCAD build to have all the components in this new version.

When it is done, it will be possible to test all the releases, and if needed it is possible to do some corrections, for example with a feature with the same process.