Morgan Stanley

Bringing Devops enablement to your mainframe developers

Calder Sacks, Executive Director

August 2026

Bio Page

CALDER SACKS

Executive Director
Enterprise Z
Phone: +1 212 276-1952
Calder.Sacks@morganstanley.com

Years of Experience: 34 Years
Years at Morgan Stanley: 27 Years

Biography

Worked in many roles within the mainframe technology landscape. Started as a Natural mainframe programmer and moved into the Adabas DBA role. Lead the database engineering team at Morgan Stanley for 10+ years and now lead the team working on Mainframe integration strategy and technologies for our Cobol/DB2 mainframe environment.

Been working on technologies between mainframe and distributed since 2003.

- One of the first companies to call Java RPC services from our mainframe Batch and CICS processes back in 2005
- Setup Sybase access from the mainframe.
- Implemented SQL & ACI access to mainframe Adabas and CICS
- Running Kafka Connectors running on the mainframe to distributed Kafka

Our goal is integrating mainframe to be just another platform and to leverage what has already been built where possible

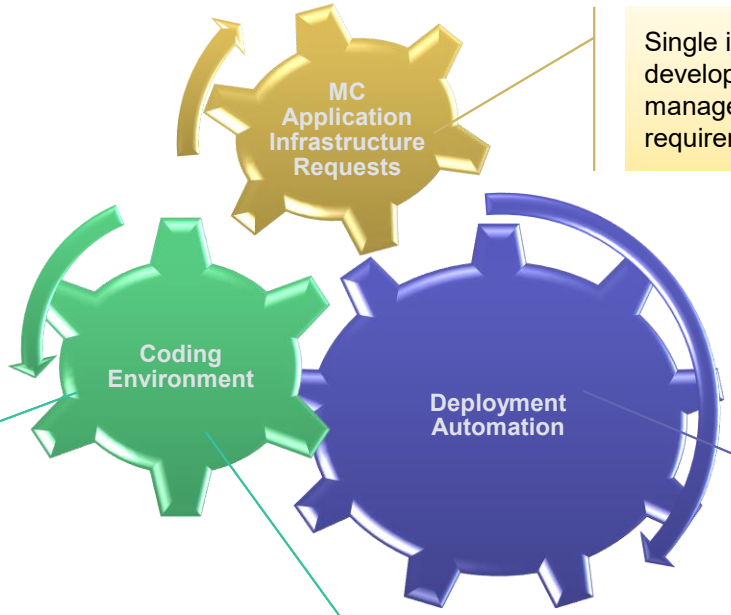
Table of Contents

Why do Git for mainframe applications?

- We were looking at ways to integrate our mainframe code base into the broader change management eco system
  - Mainframe change management systems have been around for many years
    - A number of bridges have been built over the years between mainframe change control systems and the newer distribute change control systems
    - Some of the mainframe change control systems have a limited interface to Git – some are more integrated but none of them can interface with our change control workflow
- Within our space the distributed change control process and workflows have now become the standard we have to adhere to
  - There are tighter controls and reporting required, both internally and externally
  - Newer reporting and measuring systems are built around distributed code and not for the mainframe
    - Many of our application owners now have mainframe and distributed code and they have to manage them separately
    - We cannot build bridges fast enough to talk to all the new systems in the change control workflow
- Change systems and workflows have greatly evolved in the Git and CI/CD environment

Project Goal

Simplify and unify the development environment, by providing a modern feature and function rich set of tools in an easy-to-use environment, to improve the mainframe developer experience when making requests for mainframe resources and deploying changes to the mainframe environment with full integration into the Firm standard CI/CD tool set



Single interface for developers to request and manage their infrastructure requirements

Visual Studio Code extensions to allow developers a feature rich development environment for creating and maintain Mainframe programs and objects within a Git repository

New languages Enablement of Python on the mainframe with full integration with the distributed Python environment – all coded next to your Cobol application

Integration with Firm standard tools - **JIRA, TRAIN, SPA, CDSS and TCM , MSVIZ**

Our approach to using Git

We had a few challenges to overcome

- How do you get only the changed programs ?
  - For Java we rebuild the whole app
  - For Mainframe we only want to build the 1 program that changed
- How do you copy from a distributed environment concept of ownership into a shared environment ?
  - Distributed apps tend to live in a certain set of directories that they own
  - Mainframe apps are all in a limited number of shared datasets
- How do you get your distributed CI/CD tooling to talk to the mainframe as just another node ?
- Getting distributed teams to understand mainframe call structures
- Getting mainframe people to understand distributed paradigms
- Do you put your code in Git on distributed side or keep it on the mainframe
  - Who becomes the Golden Source ?
  - And a few other lively debates….

Our approach – Key decisions first !

GIT WILL BE THE GOLDEN SOURCE

This decision was the driver behind much of the designs. We chose this approach because in our environment the distributed change control process is the new standard. To best integrate with it we need to be a part of it from the beginning of the application lifecycle.

- Support all pieces of an application
  - JCL, Procs, Parms, Control Cards, Stored Procedures, ISPF panels, REXX and many other component types

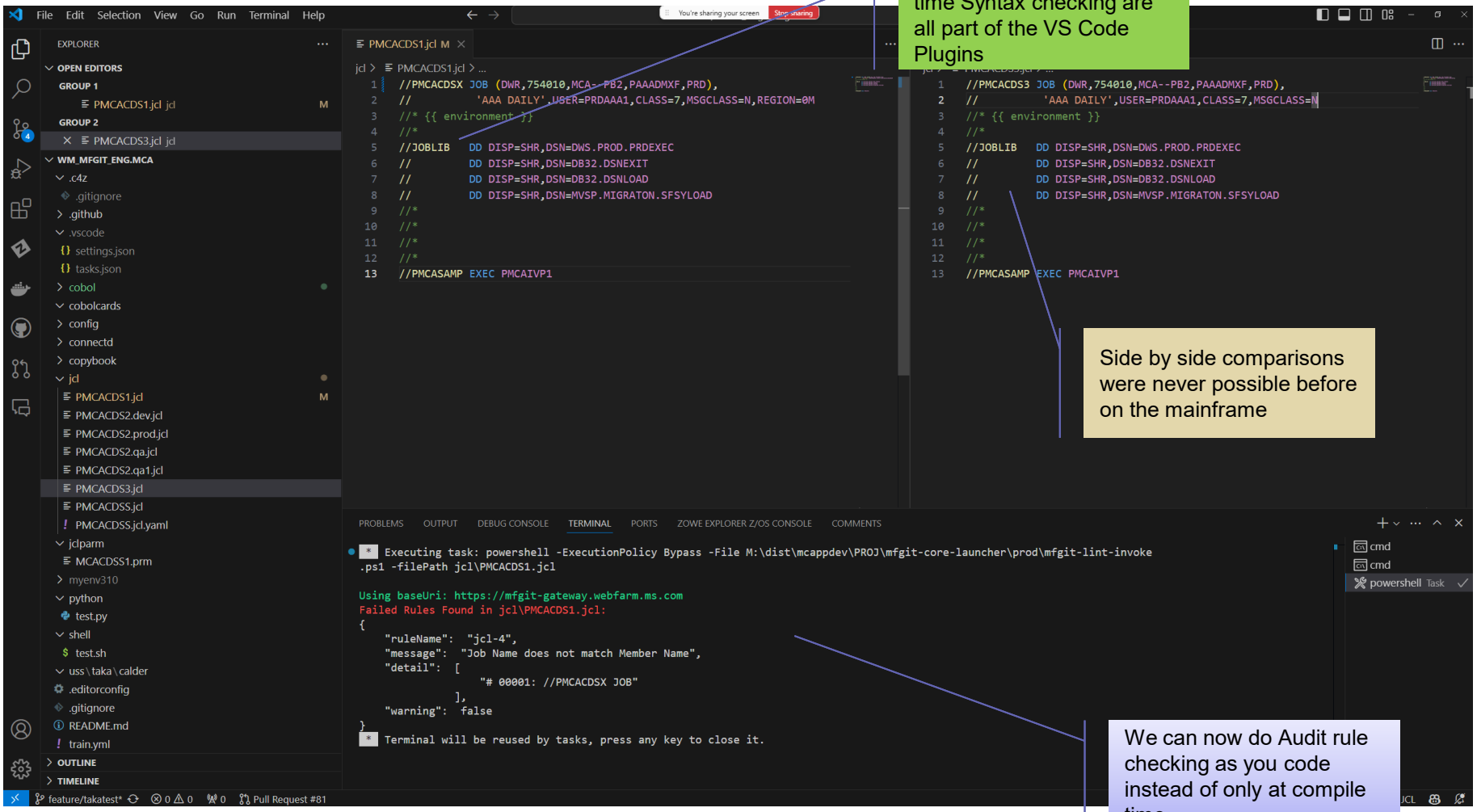
INTEGRATE WITH THE EXISTING SYSTEMS

Don’t treat the mainframe as different. In the end it is just computer code, treat it the same as any other code.

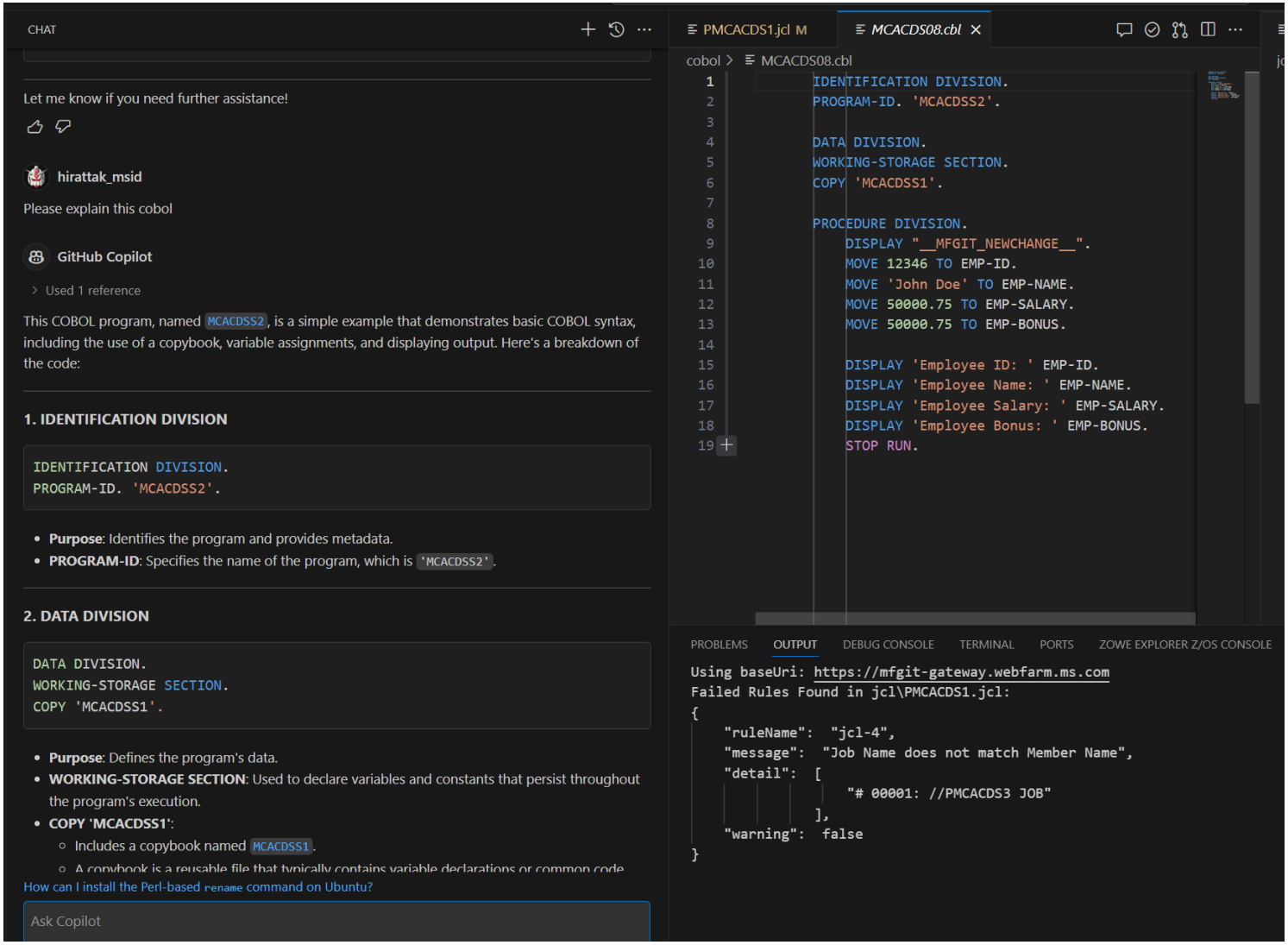
BUILD ONLY WHAT IS REQUIRED

Mainframe is not distributed and Distributed Is not mainframe, identifying the differences and catering only for these helped us build targeted approaches such as our gateway.

It starts with a Modern IDE



Using Copilot to explain Cobol code



Using VS Code and leveraging your code in GITHUB gives you access to Copilot and more of the extended tools available

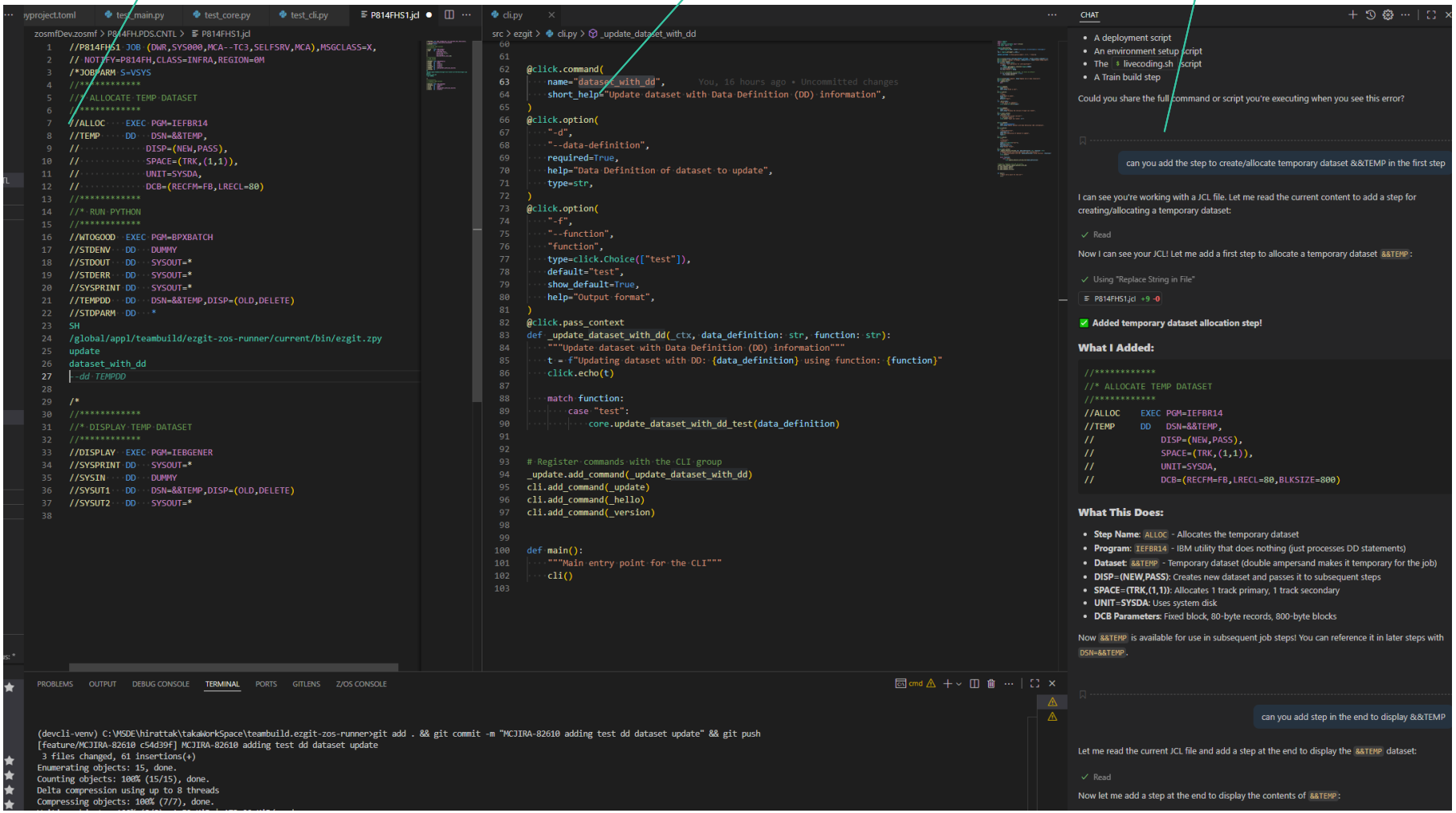
There are an array of other plugins that you can now use to dramatically enhance the overall developer experience

Changing our JCL scripts to Python for Git processing

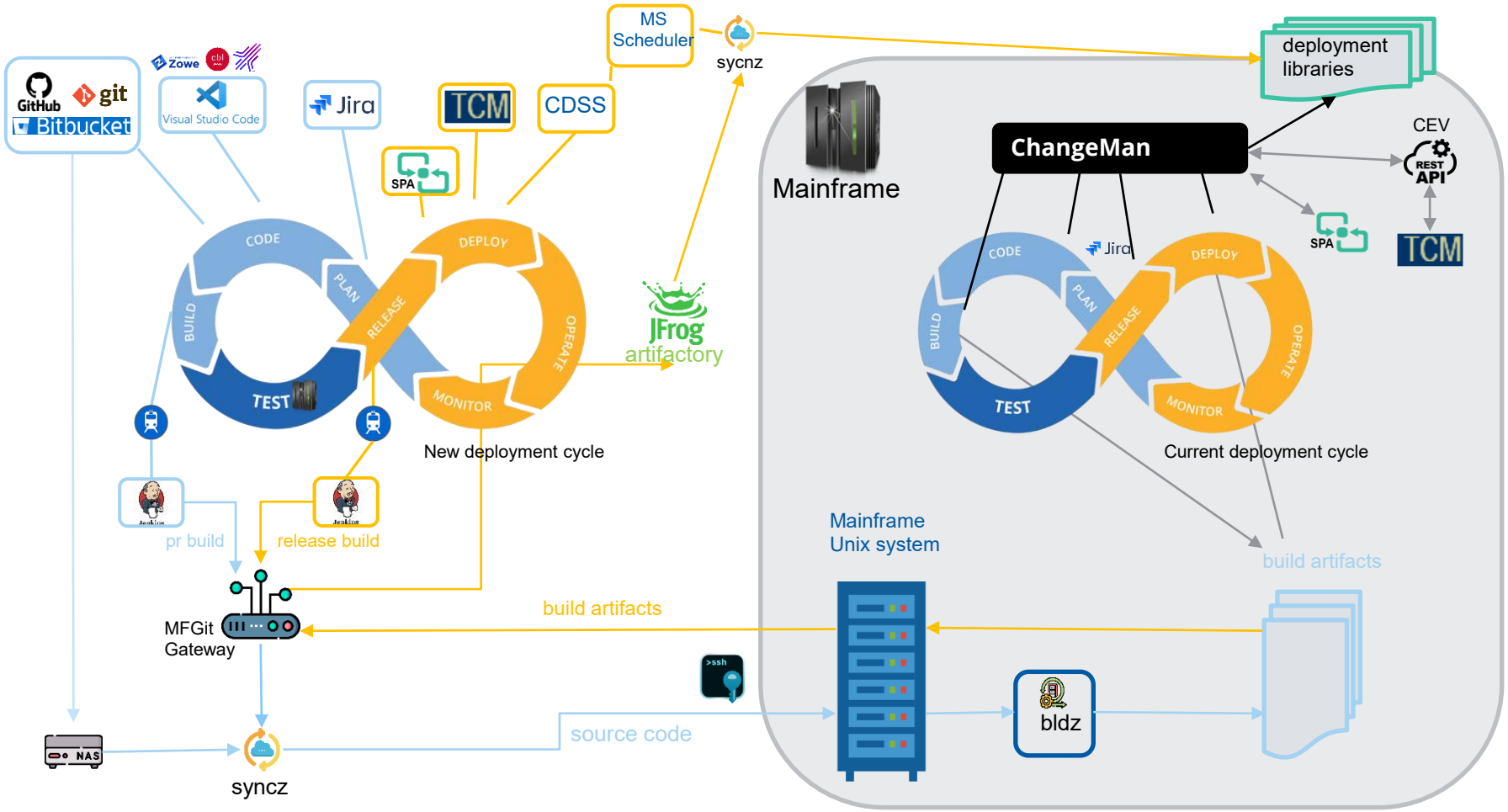
Existing JCL loaded with ZOWE

z/Python scripting

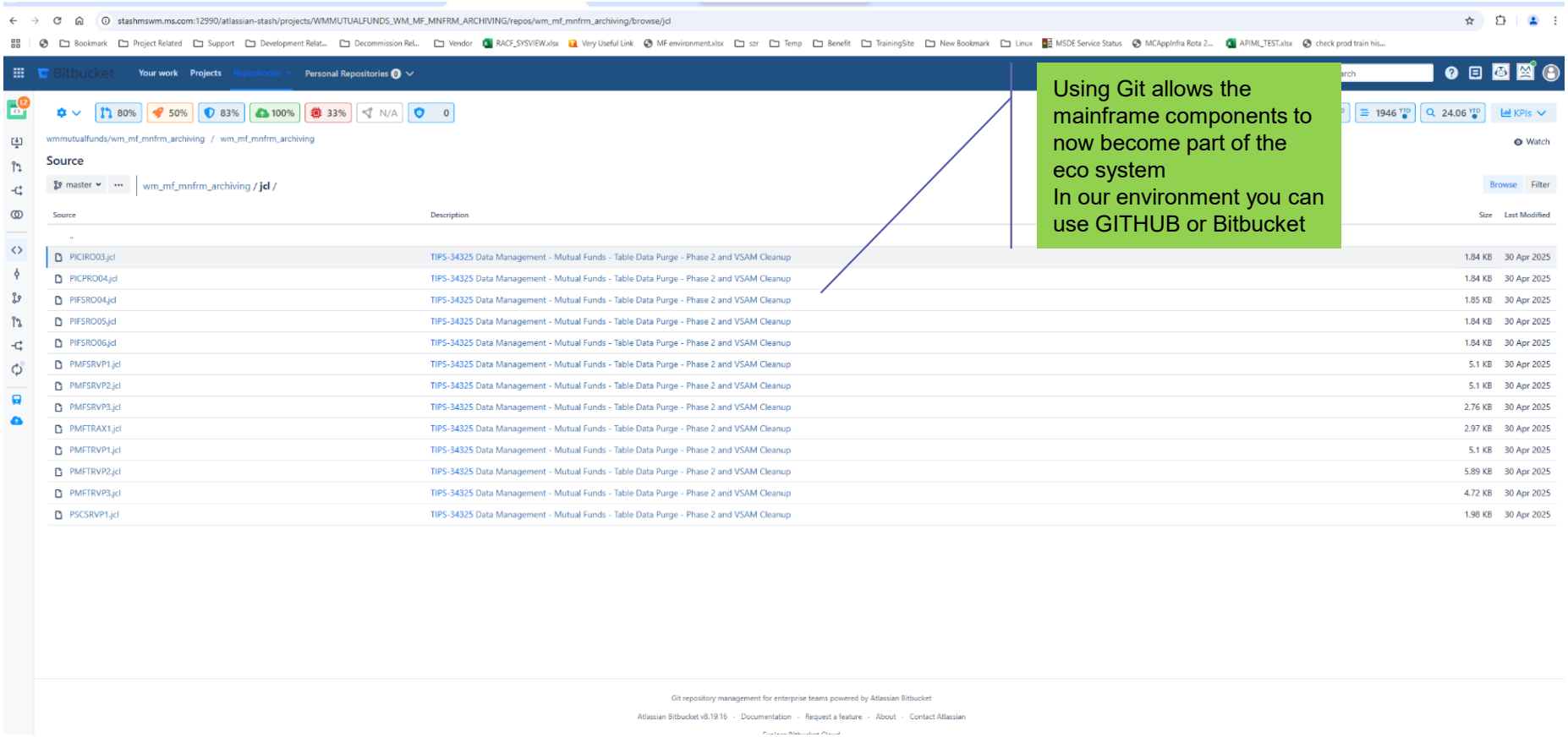
Getting Co-pilot help



The next step : Change the Mainframe Deployment Process

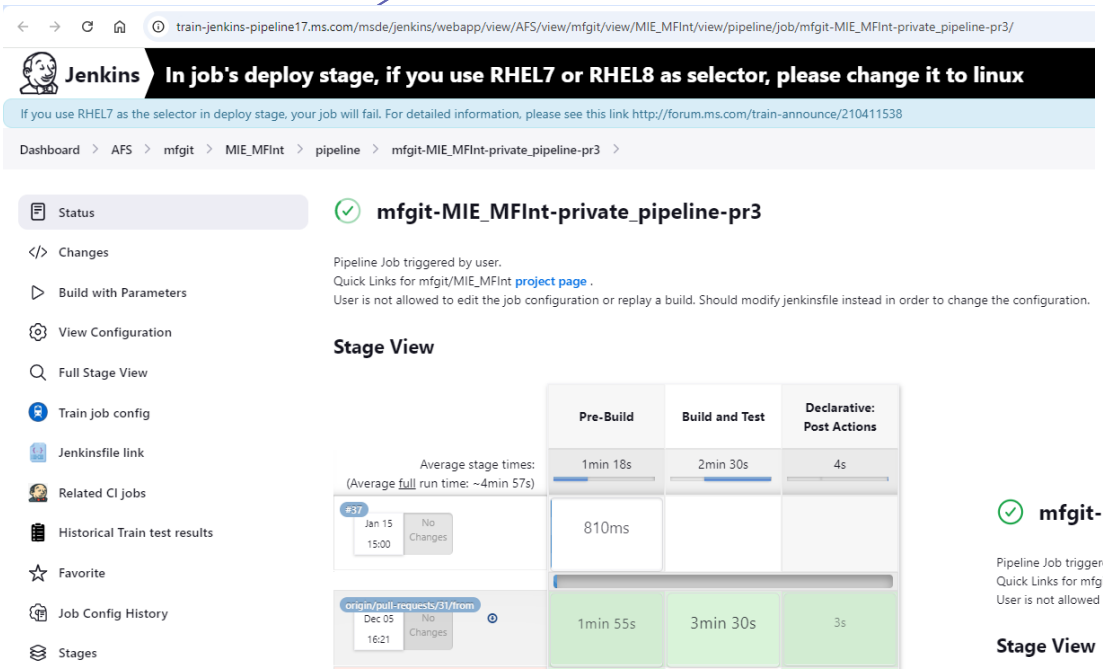


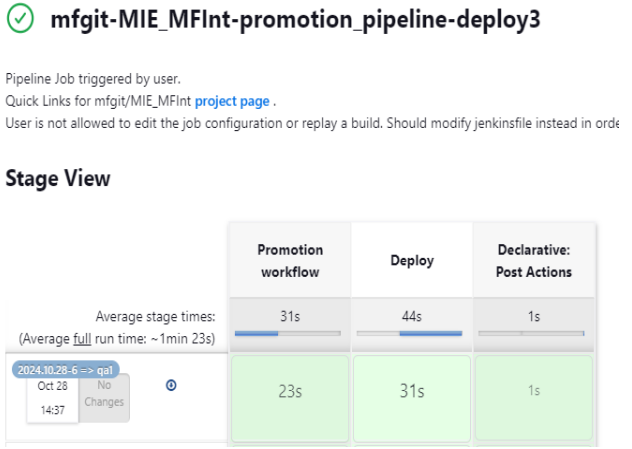
The next step is to move your code to Git

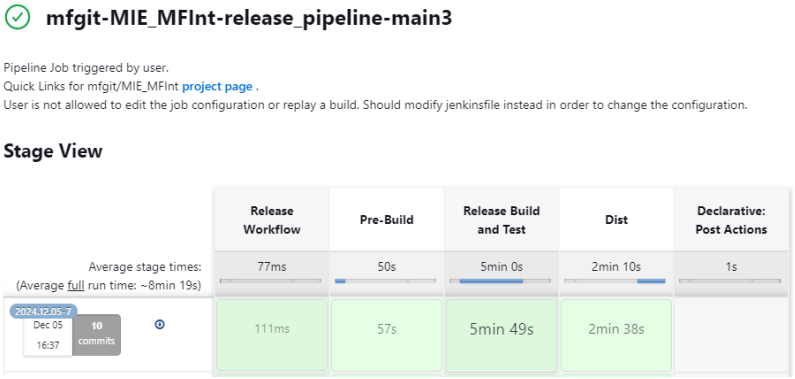


Setup the Jenkins Pipelines to run

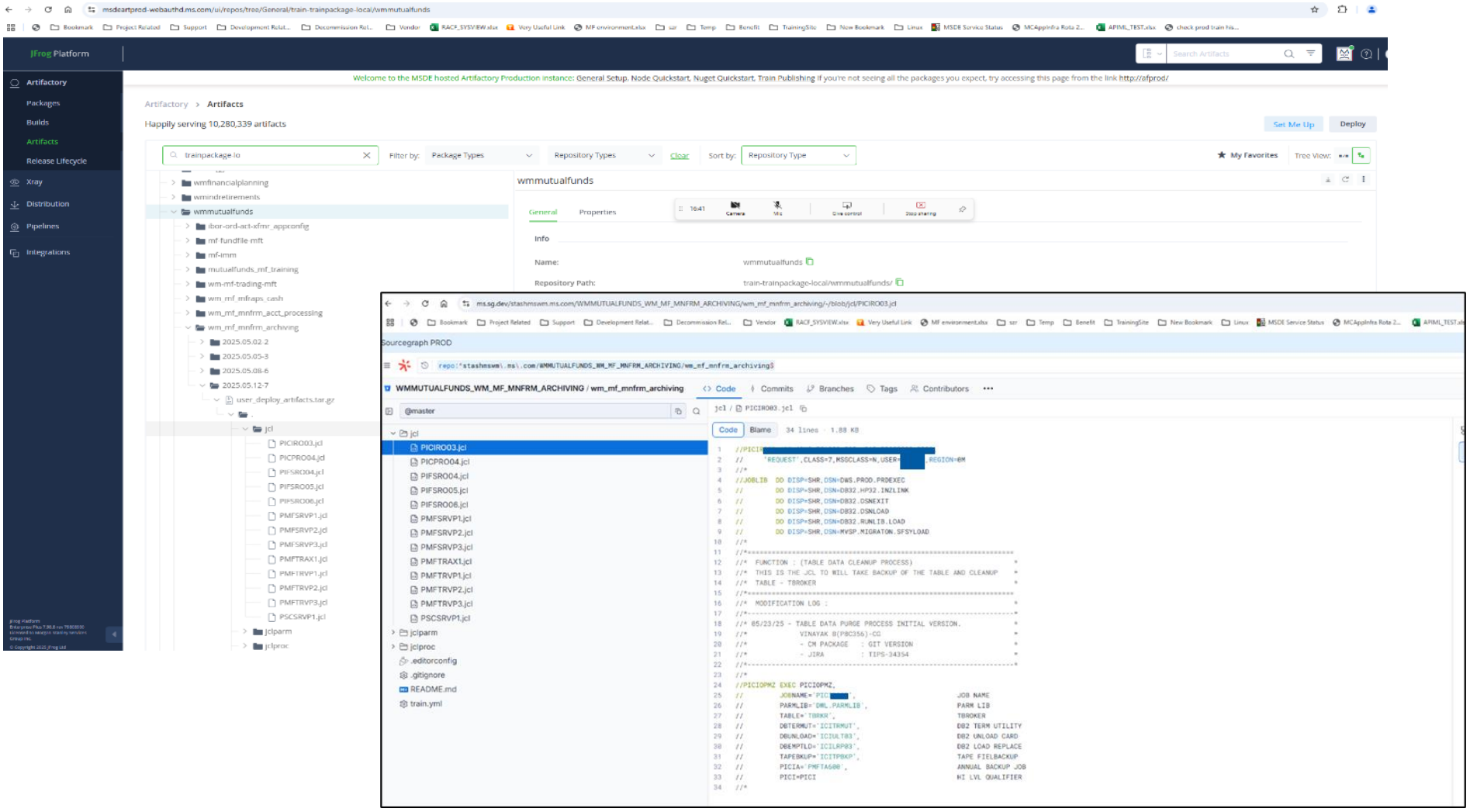
This is where we call our gateway



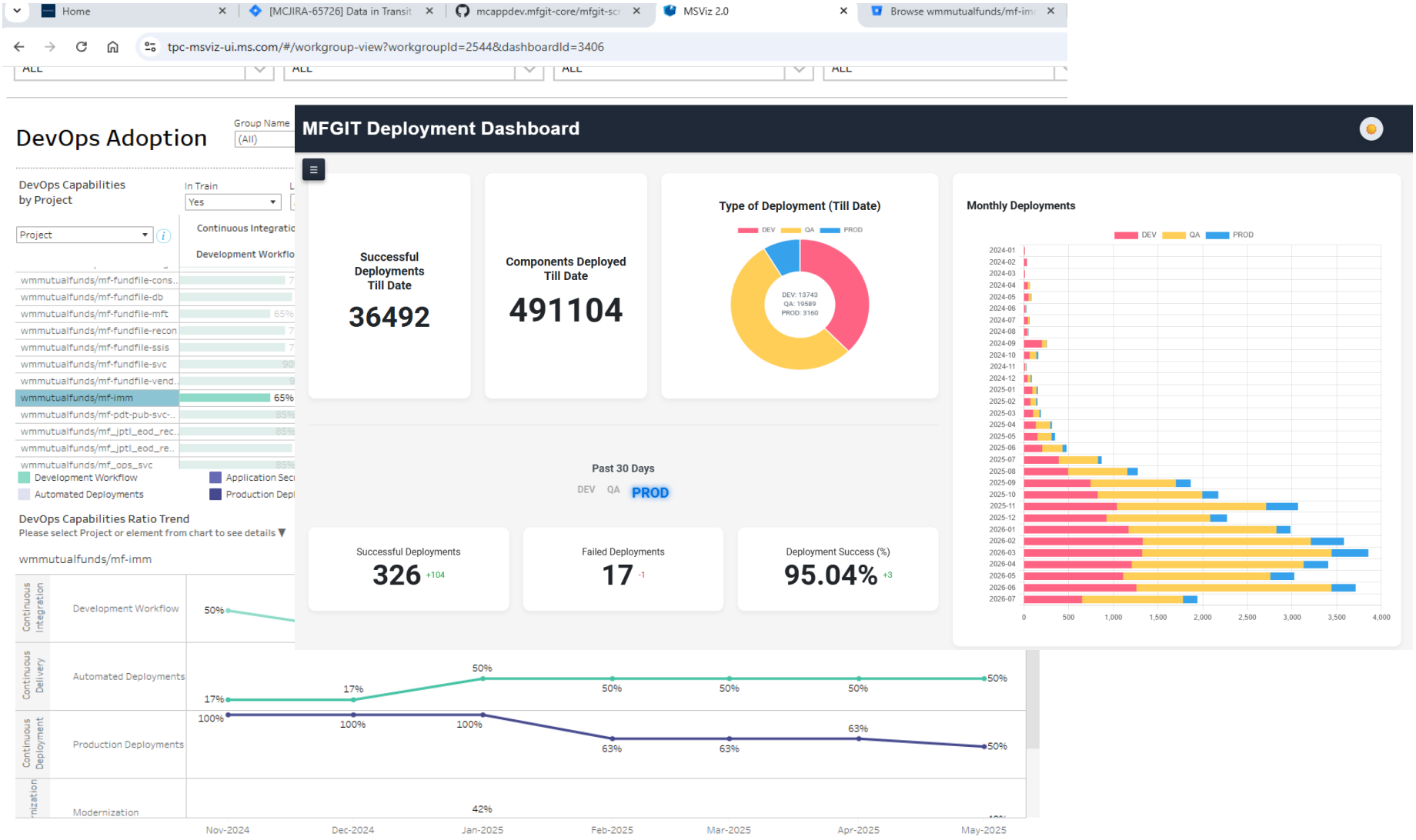




The end result is that the Mainframe code can now be seen in tools such as Artifactory or Sourcegraph

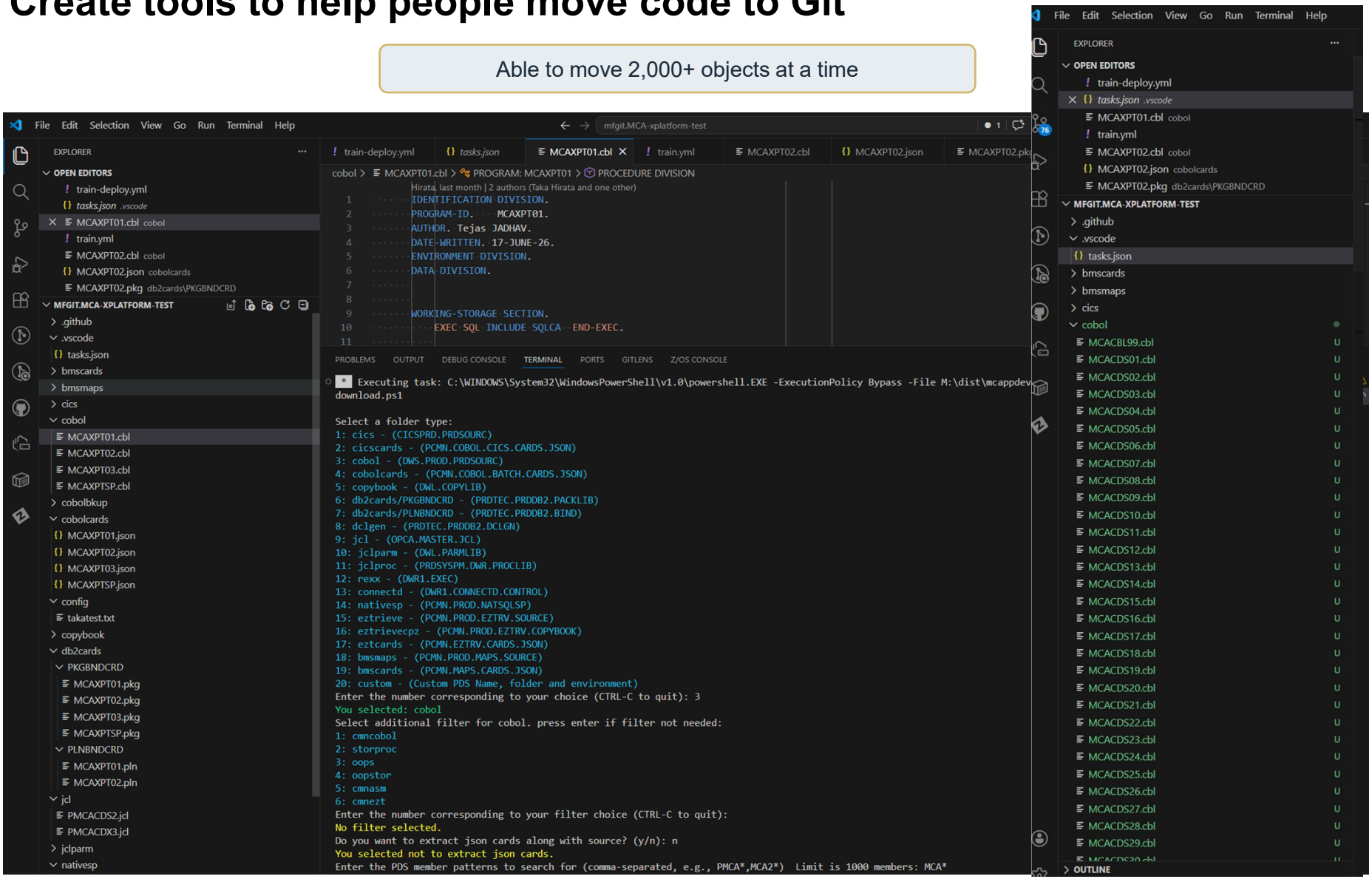


Mainframe code can now be measured in standard reporting tools

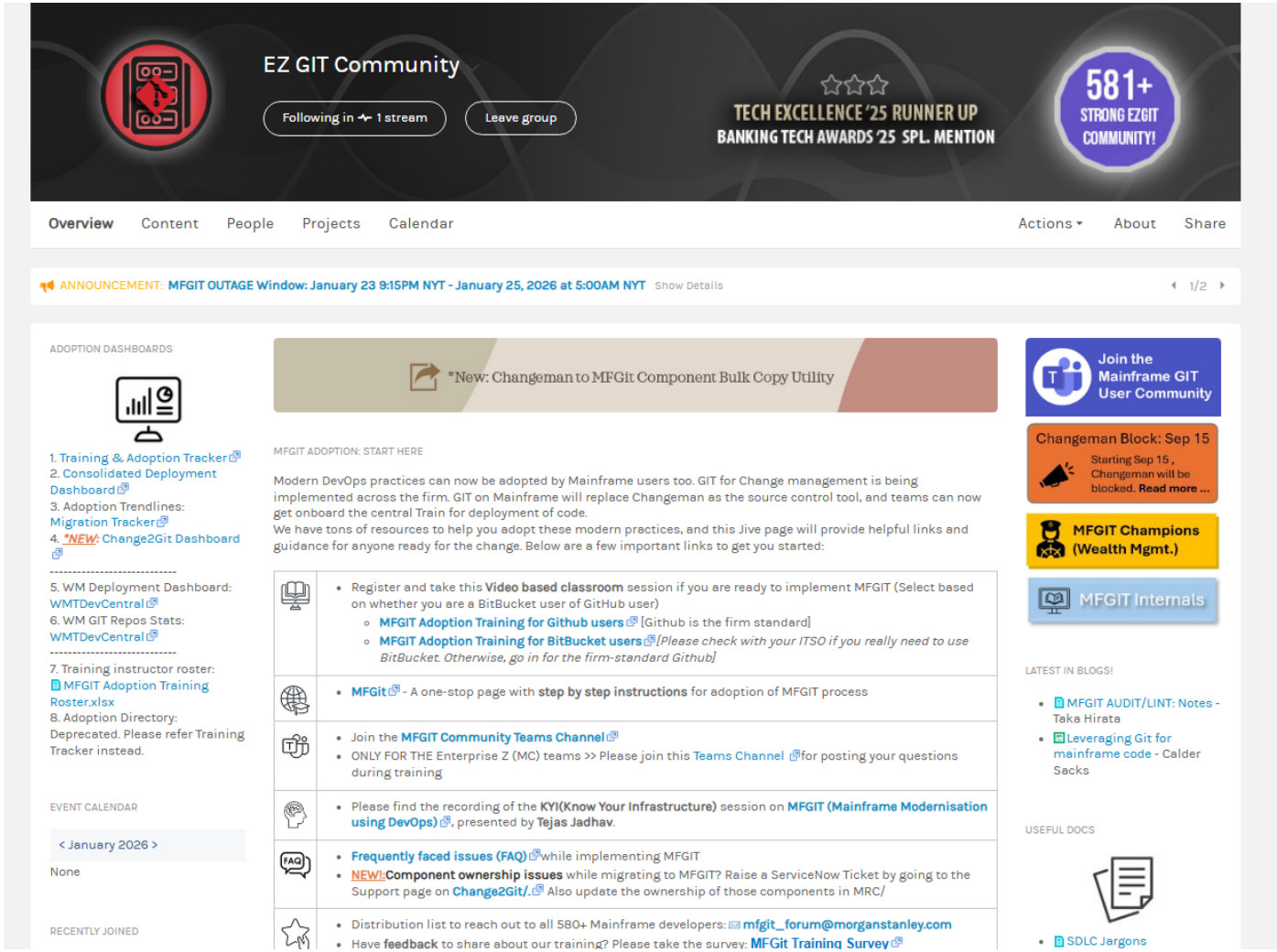


Create tools to help people move code to Git

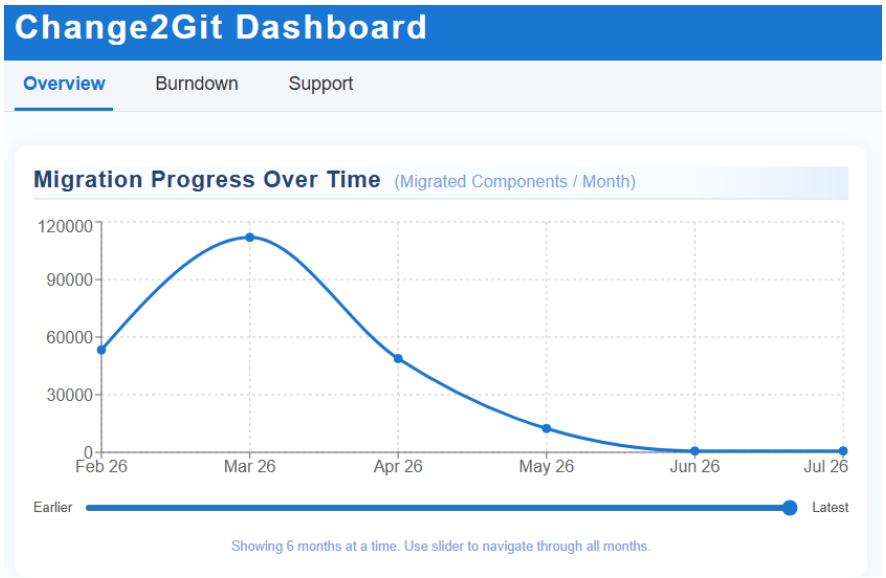
Able to move 2,000+ objects at a time

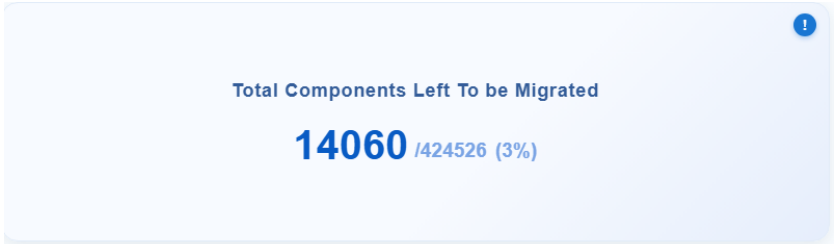


Create dashboards and metrics for people to track progress and get help



Create dashboards and metrics for people to track progress and get help





Lessons learnt

- Start with an IDE to get people more comfortable
- It is a journey
  - We started 2.5 years ago with VS Code and the plugins
- Replace functions one step at a time
  - Change control systems on the mainframe are purpose built
  - Pick the core components to start
  - Build new function support as you mature on the journey
- People need time to adapt and learn
  - Some teams will be quicker than others to adopt
  - It is a complete paradigm shift for developers
  - Mainframe engineering teams need to learn a lot of new concepts as well
    - Setup and structure is very different and requires in depth understanding

Lessons learnt

- Technologies we used
  - Cobol and JCL
  - Zowe and the mediation layer
  - Python and Java Script with extensive YAML files
  - USS on the mainframe
  - Team Build

- Most importantly - Setup a cross functional team to build it
  - Our team consists of 5 core people
    - Mainframe change control people that know the structure of the systems
    - Distributed people that know much of the distributed environment
    - The key is to identify people that are willing to learn both mainframe and distributed
    - Interactions with many of the teams outside of the core team requires “translation” and training

Questions ?