

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](mailto: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 it ?

---

Our approach for leveraging Git

---

Our journey

---

Lessons learnt

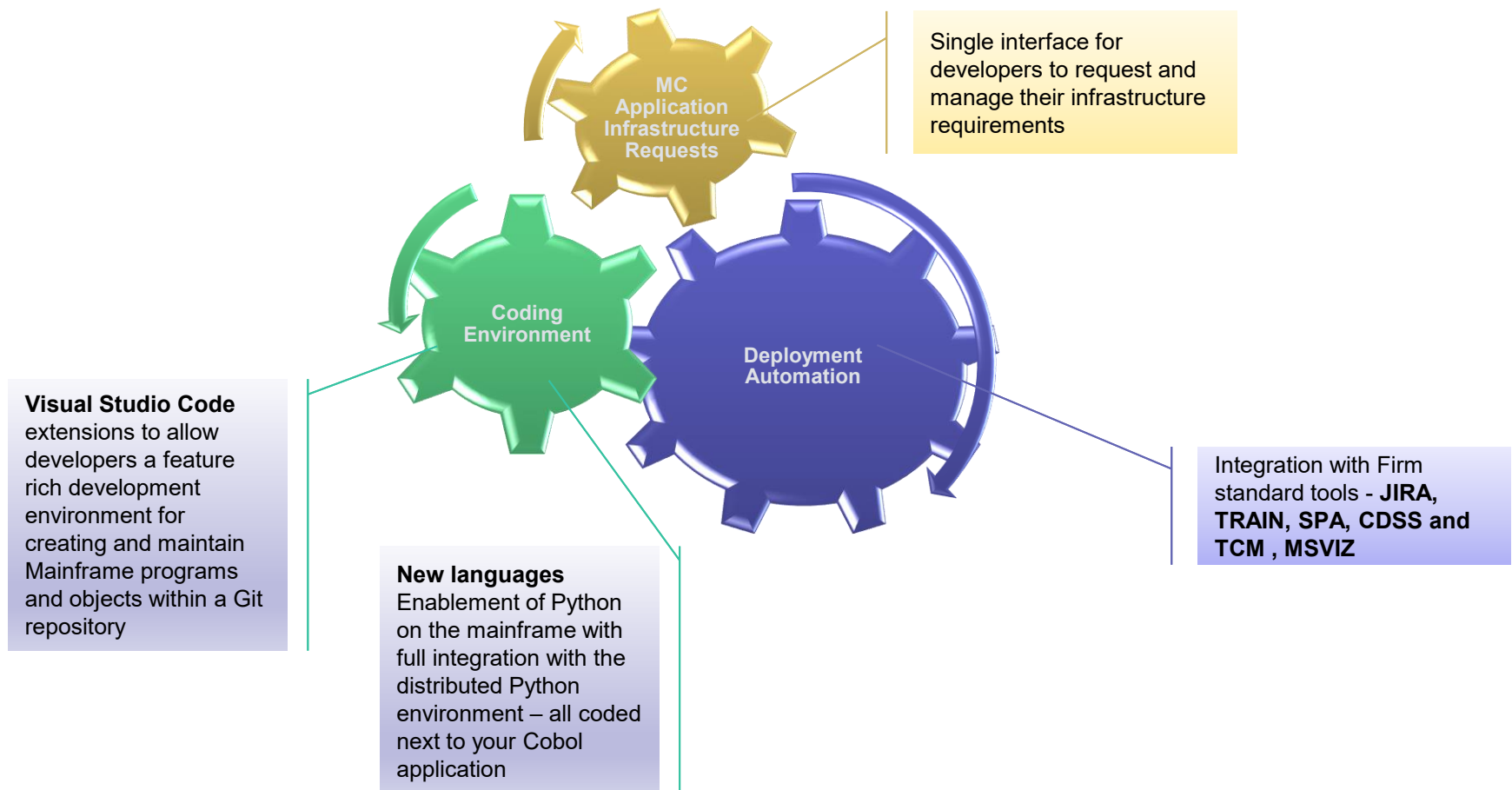
---

## 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



## 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

The screenshot displays the Visual Studio Code interface with two editor windows side-by-side, comparing two JCL files. The left window shows a file named `PMACADS1.jcl` and the right window shows `PMACADS3.jcl`. Both files contain similar JCL code, including comments and job definitions. Annotations highlight key features:

- Auto Complete and Real time Syntax checking are all part of the VS Code Plugins**: A green box points to the code in the right editor, where the IDE automatically suggests completions and highlights syntax errors in real-time.
- Side by side comparisons were never possible before on the mainframe**: A yellow box points to the split-view editor, enabling direct comparison of the two files.
- We can now do Audit rule checking as you code instead of only at compile time**: A blue box points to the terminal output, which shows the results of an audit rule check performed while coding.

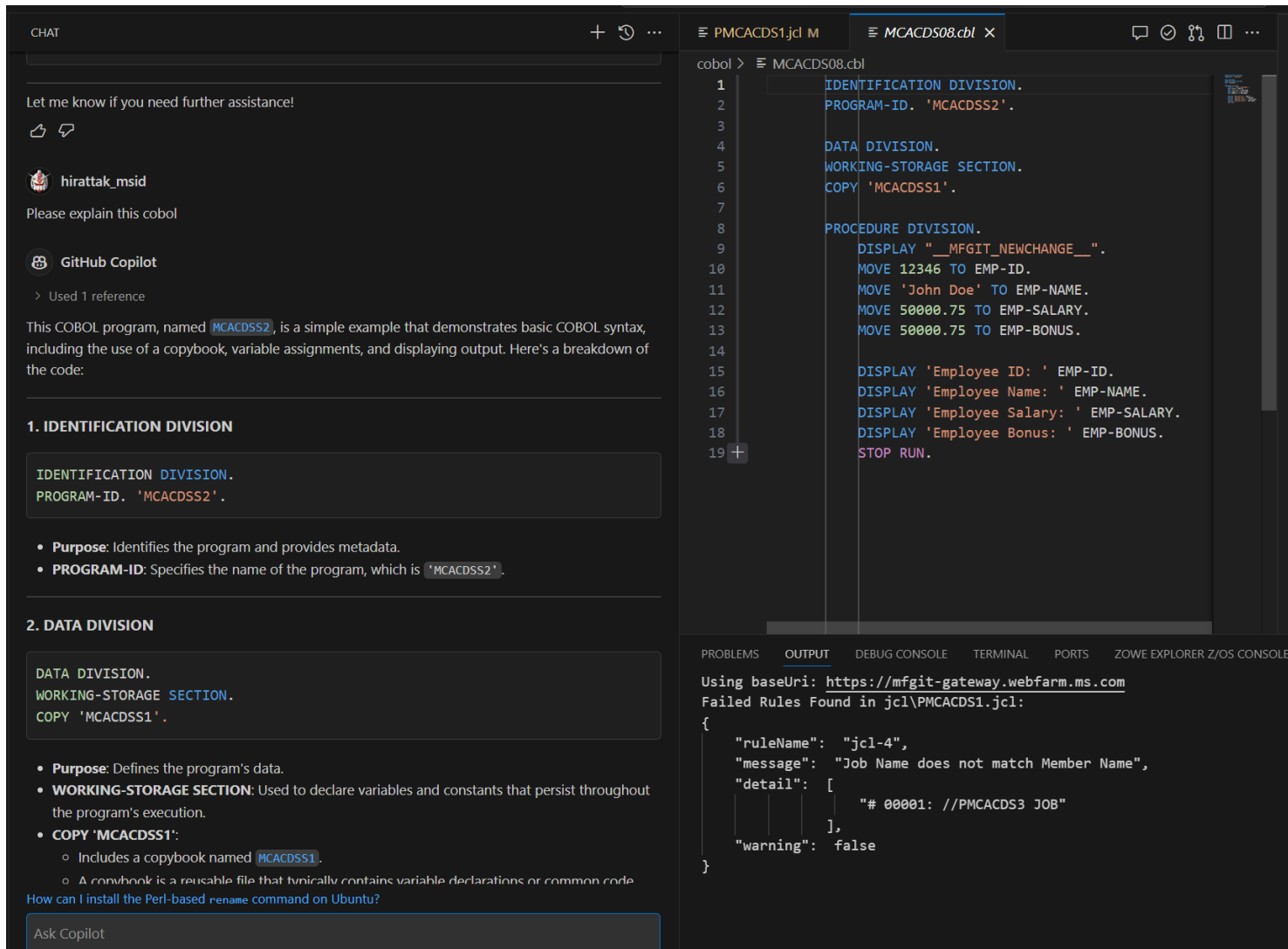
The terminal at the bottom shows the execution of a PowerShell task that runs an audit rule check. The output indicates a failed rule found in `jcl\PMACADS1.jcl`:

```

Executing task: powershell -ExecutionPolicy Bypass -File M:\dist\mcappdev\PROJ\mfgit-core-launcher\prod\mfgit-lint-invoke.ps1 -filePath jcl\PMACADS1.jcl

Using baseUri: https://mfgit-gateway.webfarm.ms.com
Failed Rules Found in jcl\PMACADS1.jcl:
{
  "ruleName": "jcl-4",
  "message": "Job Name does not match Member Name",
  "detail": [
    "# 00001: //PMACDSX JOB"
  ],
  "warning": false
}
Terminal will be reused by tasks, press any key to close it.
  
```

# Using Copilot to explain Cobol code



CHAT

Let me know if you need further assistance!

hirattak\_msid

Please explain this cobol

GitHub Copilot

> Used 1 reference

This COBOL program, named `MCACDSS2`, is a simple example that demonstrates basic COBOL syntax, including the use of a copybook, variable assignments, and displaying output. Here's a breakdown of the code:

### 1. IDENTIFICATION DIVISION

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. 'MCACDSS2'.
```

- **Purpose:** Identifies the program and provides metadata.
- **PROGRAM-ID:** Specifies the name of the program, which is `'MCACDSS2'`.

### 2. DATA DIVISION

```
DATA DIVISION.  
WORKING-STORAGE SECTION.  
COPY 'MCACDSS1'.
```

- **Purpose:** Defines the program's data.
- **WORKING-STORAGE SECTION:** Used to declare variables and constants that persist throughout the program's execution.
- **COPY 'MCACDSS1':**
  - Includes a copybook named `MCACDSS1`.
  - A copybook is a reusable file that typically contains variable declarations or common code.

How can I install the Perl-based rename command on Ubuntu?

Ask Copilot

PMACDSS1.jcl M MCACDS08.cbl

```
1 IDENTIFICATION DIVISION.  
2 PROGRAM-ID. 'MCACDSS2'.  
3  
4 DATA DIVISION.  
5 WORKING-STORAGE SECTION.  
6 COPY 'MCACDSS1'.  
7  
8 PROCEDURE DIVISION.  
9 DISPLAY " __MFGIT_NEWCHANGE__".  
10 MOVE 12346 TO EMP-ID.  
11 MOVE 'John Doe' TO EMP-NAME.  
12 MOVE 50000.75 TO EMP-SALARY.  
13 MOVE 50000.75 TO EMP-BONUS.  
14  
15 DISPLAY 'Employee ID: ' EMP-ID.  
16 DISPLAY 'Employee Name: ' EMP-NAME.  
17 DISPLAY 'Employee Salary: ' EMP-SALARY.  
18 DISPLAY 'Employee Bonus: ' EMP-BONUS.  
19 STOP RUN.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ZOWE EXPLORER Z/OS CONSOLE

Using baseUri: <https://mfgit-gateway.webfarm.ms.com>

Failed Rules Found in jcl\PMACDSS1.jcl:

```
{  
  "ruleName": "jcl-4",  
  "message": "Job Name does not match Member Name",  
  "detail": [  
    "# 00001: //PMACDSS3 JOB"  
  ],  
  "warning": false  
}
```

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

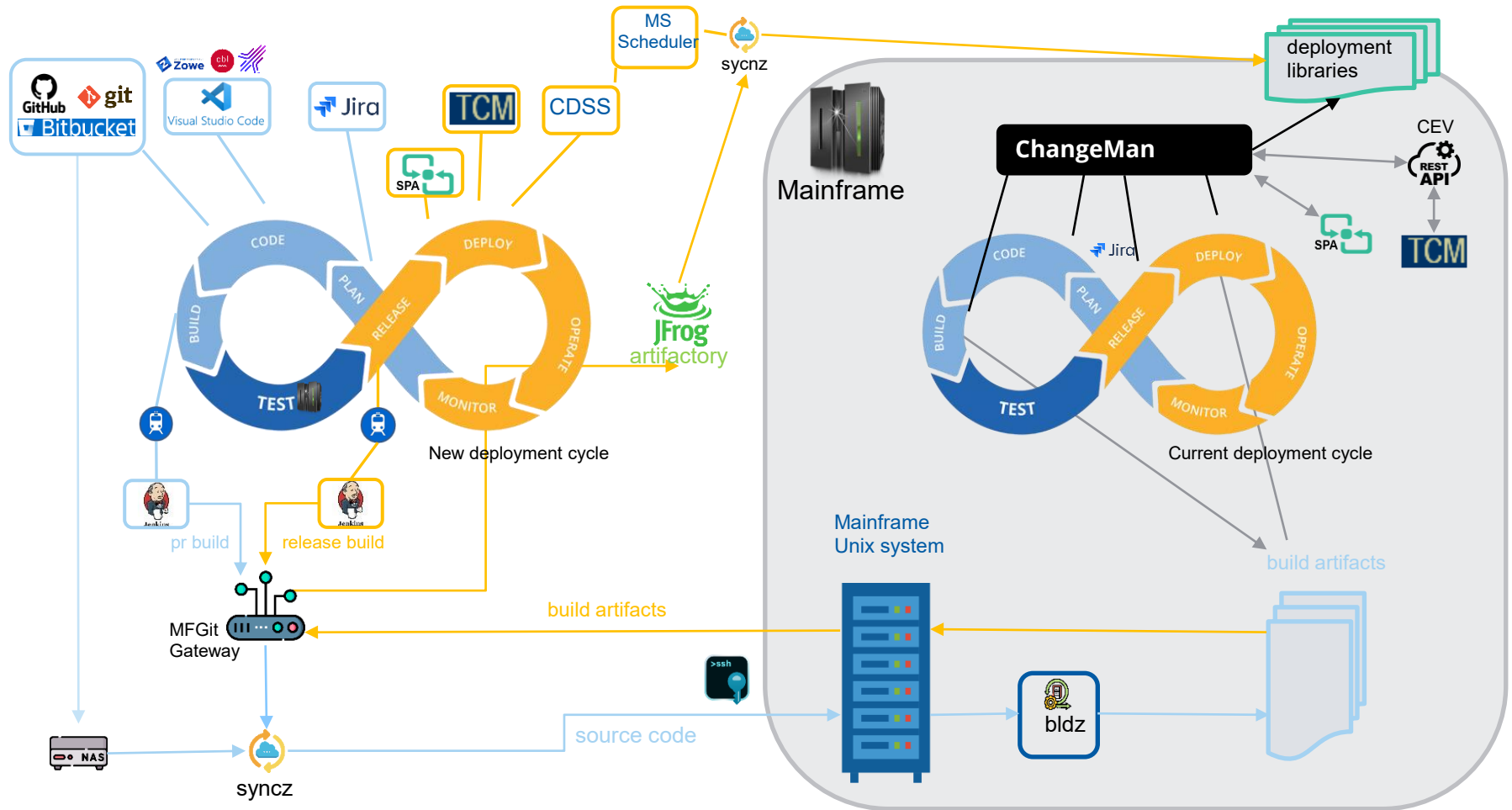
z/Python scripting

Getting Co-pilot help

The image displays a development environment with three main components:

- Left Pane (JCL Script):** Shows the original JCL script `P814FH51.jcl`. It includes comments like `//*JOB=ARM S=VSYS` and `//*ALLOCATE TEMP DATASET`, followed by `ALLOC` and `DD` statements for a temporary dataset.
- Right Pane (Python Script):** Shows the new Python script `cli.py` using `Click` for CLI handling. It defines a command `update_dataset_with_dd` and a function `update_dataset_with_dd_test`. Co-pilot suggestions are visible, such as "can you add the step to create/allocate temporary dataset" and "I can see you're working with a JCL file. Let me read the current content to add a step for creating/allocating a temporary dataset".
- Bottom Pane (Terminal):** Shows the execution of the new script: `(devcli-venv) C:\MSDE\hiraattak\workspace\teambuild\ezgit-zos-runner>git add . && git commit -m "MCJIRA-82610 adding test dd dataset update" && git push`. The output shows successful commit and push operations.

# The next step : Change the Mainframe Deployment Process



# The next step is to move your code to Git

Using Git allows the mainframe components to now become part of the eco system  
In our environment you can use GITHUB or Bitbucket

| Source       | Description                                                                             | Size    | Last Modified |
|--------------|-----------------------------------------------------------------------------------------|---------|---------------|
| ..           |                                                                                         |         |               |
| PICIRO03.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 1.84 KB | 30 Apr 2025   |
| PICPRO04.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 1.84 KB | 30 Apr 2025   |
| PIFSRO04.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 1.85 KB | 30 Apr 2025   |
| PIFSRO05.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 1.84 KB | 30 Apr 2025   |
| PIFSRO06.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 1.84 KB | 30 Apr 2025   |
| PMFSRVP1.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 5.1 KB  | 30 Apr 2025   |
| PMFSRVP2.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 5.1 KB  | 30 Apr 2025   |
| PMFSRVP3.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 2.76 KB | 30 Apr 2025   |
| PMFTRAX1.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 2.97 KB | 30 Apr 2025   |
| PMFTRVP1.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 5.1 KB  | 30 Apr 2025   |
| PMFTRVP2.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 5.89 KB | 30 Apr 2025   |
| PMFTRVP3.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 4.72 KB | 30 Apr 2025   |
| PSCSRVP1.jcl | TIPS-34325 Data Management - Mutual Funds - Table Data Purge - Phase 2 and VSAM Cleanup | 1.98 KB | 30 Apr 2025   |

Git repository management for enterprise teams powered by Atlassian Bitbucket  
Atlassian Bitbucket v8.19.16 · Documentation · Request a feature · About · Contact Atlassian

# Setup the Jenkins Pipelines to run

Jenkins

In job's deploy stage, if you use RHEL7 or RHEL8 as selector, please change it to linux

If you use RHEL7 as the selector in deploy stage, your job will fail. For detailed information, please see this link <http://forum.ms.com/train-announce/210411538>

Dashboard > AFS > mfgit > MIE\_MFInt > pipeline > mfgit-MIE\_MFInt-private\_pipeline-pr3 >

Status

Changes

Build with Parameters

View Configuration

Full Stage View

Train job config

Jenkinsfile link

Related CI jobs

Historical Train test results

Favorite

Job Config History

Stages

mfgit-MIE\_MFInt-private\_pipeline-pr3

Pipeline Job triggered by user.

Quick Links for mfgit/MIE\_MFInt [project page](#).

User is not allowed to edit the job configuration or replay a build. Should modify jenkinsfile instead in order to change the configuration.

Stage View

Average stage times:

(Average full run time: ~4min 57s)

|                                                                                       | Pre-Build | Build and Test | Declarative: Post Actions |
|---------------------------------------------------------------------------------------|-----------|----------------|---------------------------|
| <div>2024.10.28-6 =&gt; qa1</div> <div>Oct 28 14:37</div> <div>No Changes</div>       | 810ms     |                |                           |
| <div>origin/pull-requests/31/from</div> <div>Dec 05 16:21</div> <div>No Changes</div> | 1min 55s  | 3min 30s       | 3s                        |

This is where we call our gateway

mfgit-MIE\_MFInt-promotion\_pipeline-deploy3

Pipeline Job triggered by user.

Quick Links for mfgit/MIE\_MFInt [project page](#).

User is not allowed to edit the job configuration or replay a build. Should modify jenkinsfile instead in order to change the configuration.

## Stage View

|                                                                                 | Promotion workflow | Deploy | Declarative: Post Actions |
|---------------------------------------------------------------------------------|--------------------|--------|---------------------------|
| Average stage times:<br>(Average full run time: ~1min 23s)                      | 31s                | 44s    | 1s                        |
| <div>2024.10.28-6 =&gt; qa1</div> <div>Oct 28 14:37</div> <div>No Changes</div> | 23s                | 31s    | 1s                        |

mfgit-MIE\_MFInt-release\_pipeline-main3

Pipeline Job triggered by user.

Quick Links for mfgit/MIE\_MFInt [project page](#).

User is not allowed to edit the job configuration or replay a build. Should modify jenkinsfile instead in order to change the configuration.

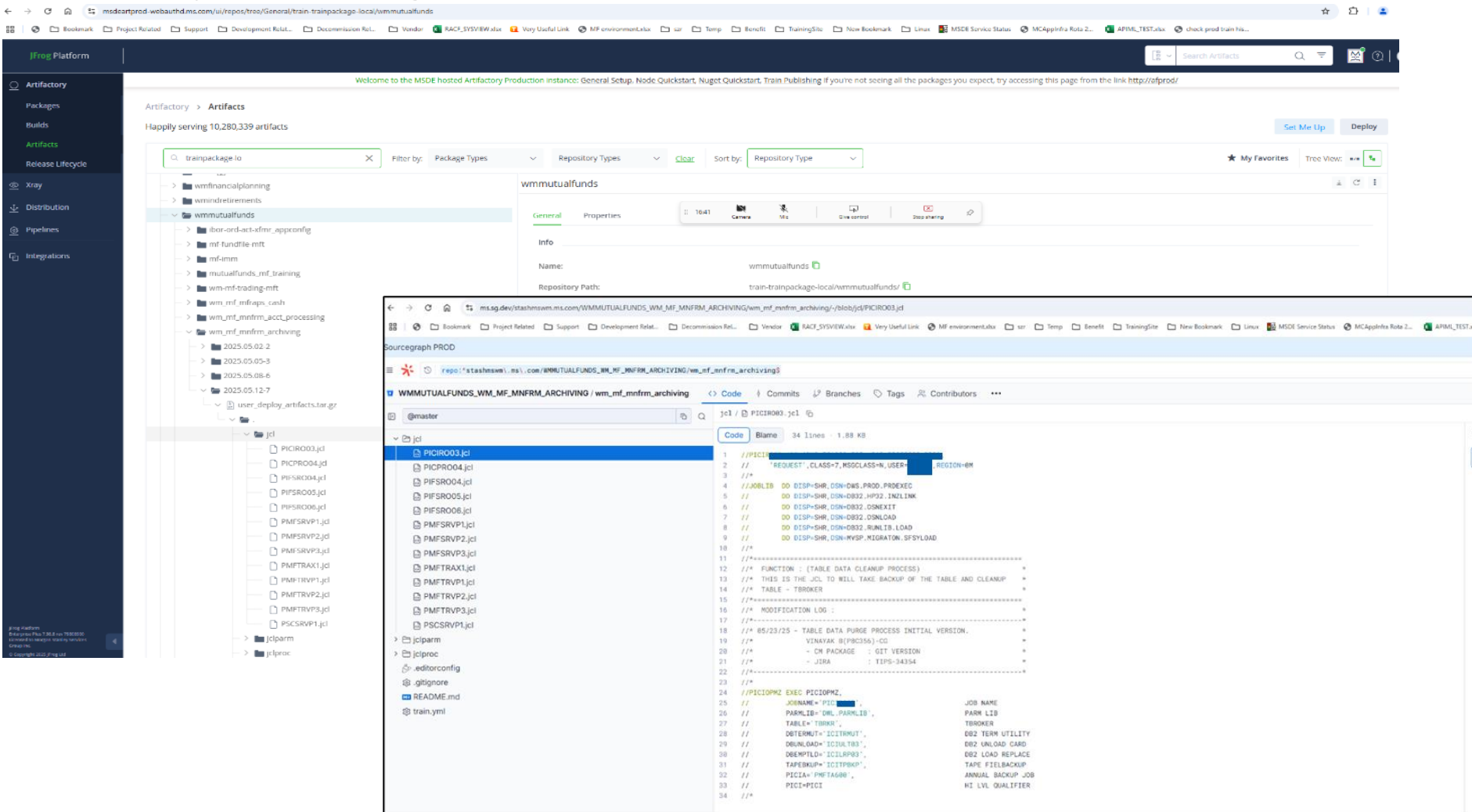
## Stage View

|                                                                       | Release Workflow | Pre-Build | Release Build and Test | Dist     | Declarative: Post Actions |
|-----------------------------------------------------------------------|------------------|-----------|------------------------|----------|---------------------------|
| Average stage times:<br>(Average full run time: ~8min 19s)            | 77ms             | 50s       | 5min 0s                | 2min 10s | 1s                        |
| <div>2024.12.05-7</div> <div>Dec 05 16:37</div> <div>10 commits</div> | 111ms            | 57s       | 5min 49s               | 2min 38s |                           |

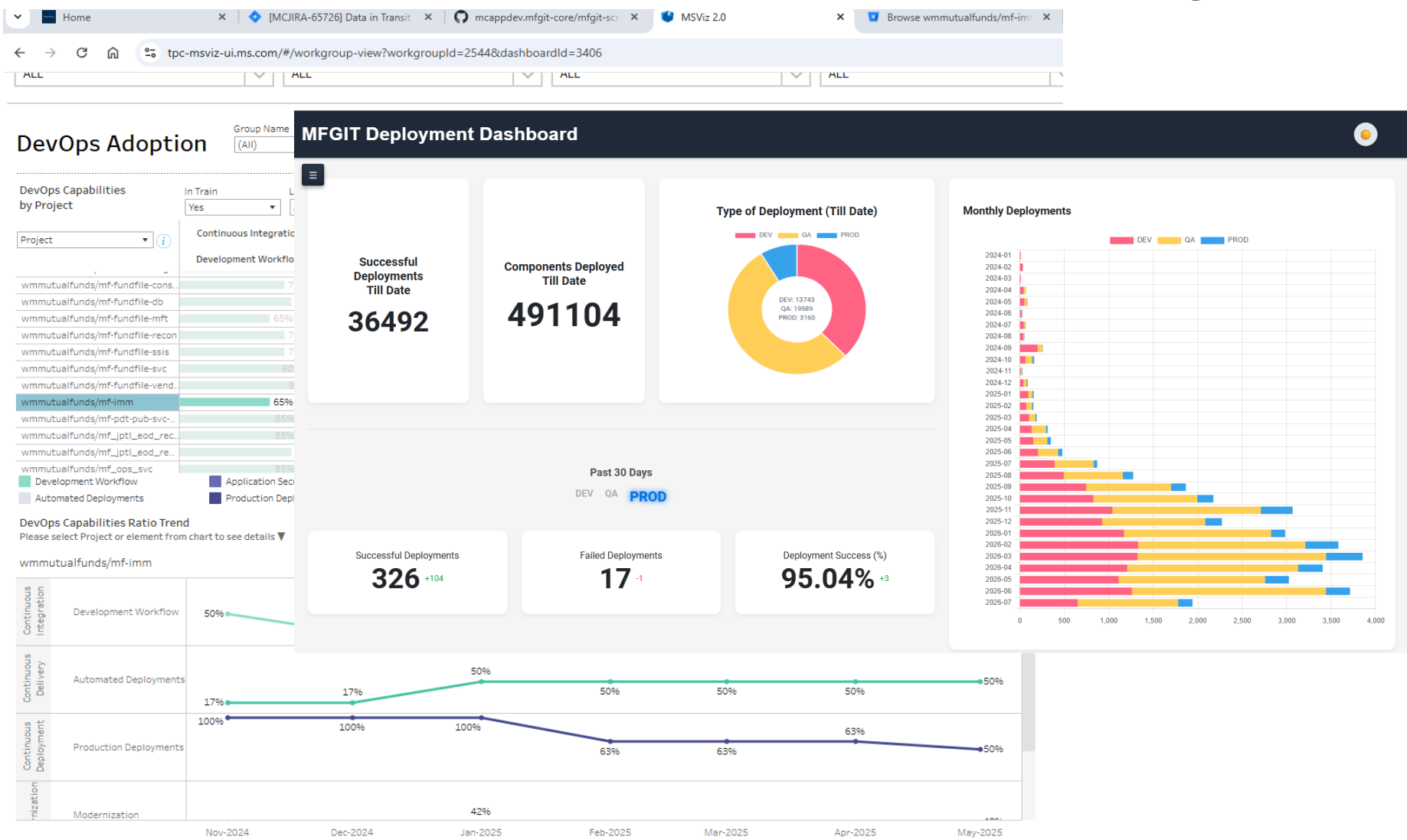
GIT AND DEVOPS PIPELINES FOR MAINFRAME APPLICATIONS

13

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

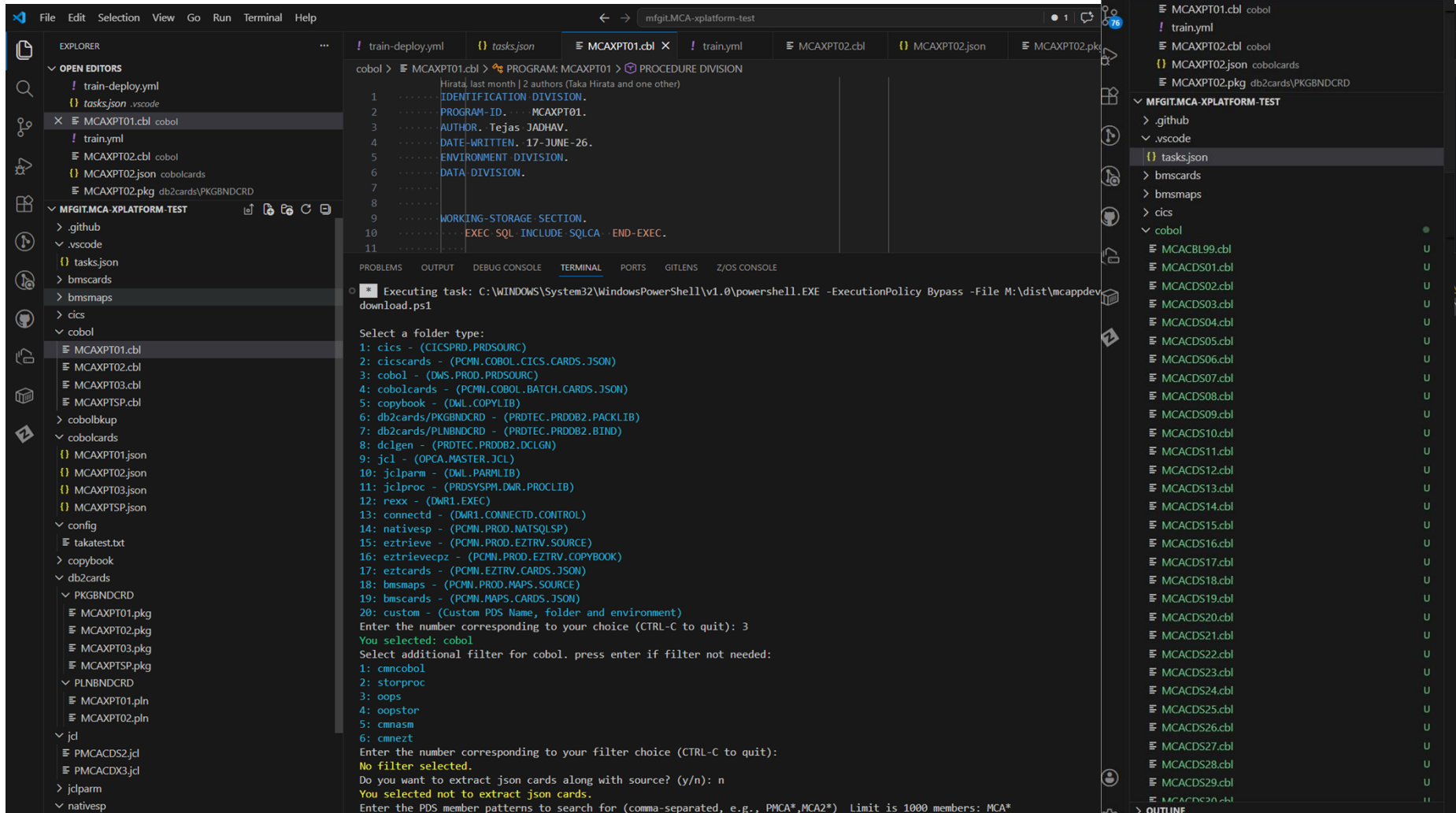


# Mainframe code can now be measured in standard reporting tools



# 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



## EZ GIT Community

[Following in 1 stream](#)
[Leave group](#)





[Overview](#)
[Content](#)
[People](#)
[Projects](#)
[Calendar](#)
[Actions](#)
[About](#)
[Share](#)

**ANNOUNCEMENT:** [MFGIT OUTAGE Window: January 23 9:15PM NYT - January 25, 2026 at 5:00AM NYT](#) [Show Details](#)

1/2

### ADOPTION DASHBOARDS



1. Training & Adoption Tracker
2. Consolidated Deployment Dashboard
3. Adoption Trendlines: Migration Tracker
4. **\*NEW:** Change2Git Dashboard
5. WM Deployment Dashboard: WMTDevCentral
6. WM GIT Repos Stats: WMTDevCentral
7. Training instructor roster: MFGIT Adoption Training Roster.xlsx
8. Adoption Directory: Deprecated. Please refer Training Tracker instead.

EVENT CALENDAR

< January 2026 >

None

RECENTLY JOINED



**\*New: Changeman to MFGit Component Bulk Copy Utility**

MFGIT ADOPTION: START HERE

Modern DevOps practices can now be adopted by Mainframe users too. GIT for Change management is being implemented across the firm. GIT on Mainframe will replace Changeman as the source control tool, and teams can now get onboard the central Train for deployment of code. We have tons of resources to help you adopt these modern practices, and this Jive page will provide helpful links and guidance for anyone ready for the change. Below are a few important links to get you started:

|                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <ul style="list-style-type: none"> <li>Register and take this <b>Video based classroom</b> session if you are ready to implement MFGIT (Select based on whether you are a BitBucket user or Github user)               <ul style="list-style-type: none"> <li><a href="#">MFGIT Adoption Training for Github users</a> [Github is the firm standard]</li> <li><a href="#">MFGIT Adoption Training for BitBucket users</a> [Please check with your ITSO if you really need to use BitBucket. Otherwise, go in for the firm-standard Github]</li> </ul> </li> </ul> |
|  | <ul style="list-style-type: none"> <li><a href="#">MFGit</a> - A one-stop page with <b>step by step instructions</b> for adoption of MFGIT process</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                     |
|  | <ul style="list-style-type: none"> <li>Join the <a href="#">MFGIT Community Teams Channel</a></li> <li>ONLY FOR THE Enterprise Z (MC) teams &gt;&gt; Please join this <a href="#">Teams Channel</a> for posting your questions during training</li> </ul>                                                                                                                                                                                                                                                                                                         |
|  | <ul style="list-style-type: none"> <li>Please find the recording of the KYI(Know Your Infrastructure) session on <a href="#">MFGIT (Mainframe Modernisation using DevOps)</a>, presented by Tejas Jadhav.</li> </ul>                                                                                                                                                                                                                                                                                                                                              |
|  | <ul style="list-style-type: none"> <li><a href="#">Frequently faced issues (FAQ)</a> while implementing MFGIT</li> <li><b>NEW:</b> <a href="#">Component ownership issues</a> while migrating to MFGIT? Raise a ServiceNow Ticket by going to the Support page on <a href="#">Change2Git</a>. Also update the ownership of those components in MRC/</li> </ul>                                                                                                                                                                                                    |
|  | <ul style="list-style-type: none"> <li>Distribution list to reach out to all 580+ Mainframe developers: <a href="mailto:mfgit_forum@morganstanley.com">mfgit_forum@morganstanley.com</a></li> <li>Have <b>feedback</b> to share about our training? Please take the survey: <a href="#">MFGit Training Survey</a></li> </ul>                                                                                                                                                                                                                                      |



**Changeman Block: Sep 15**

Starting Sep 15, Changeman will be blocked. [Read more ...](#)

**MFGIT Champions (Wealth Mgmt.)**

**MFGIT Internals**

LATEST IN BLOGS!

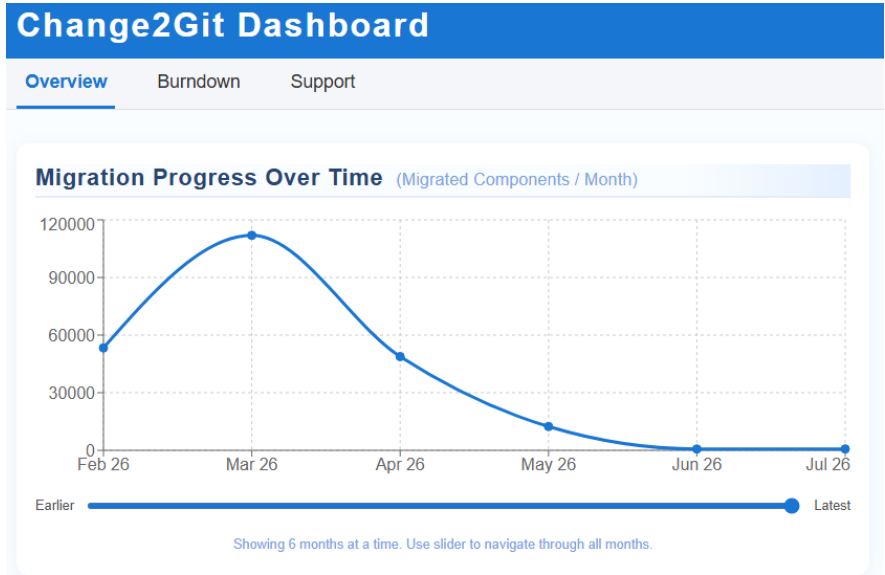
- [MFGIT AUDIT/LINT: Notes - Taka Hirata](#)
- [Leveraging Git for mainframe code - Calder Sacks](#)

USEFUL DOCS



- [SDLC Jargons](#)

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



Total Components Left To be Migrated

**14060** /424526 (3%)

Total Components Migrated

**410466**

## 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 ?