

# DevOps on z Proof of Technology

David Hawreluk

[dhawrel@us.ibm.com](mailto:dhawrel@us.ibm.com)

Vanessa Ramirez Callaghan

[vanessa.r.callaghan@ibm.com](mailto:vanessa.r.callaghan@ibm.com)

Carlos a Hirata

[chirata@us.ibm.com](mailto:chirata@us.ibm.com)

Wilbert Kho

[wilbert@us.ibm.com](mailto:wilbert@us.ibm.com)



# Audience

- This Proof of Technology is targeted to, but not limited to, architects, technical specialists or developers
- Non-technical attendees with an understanding of application lifecycle management are welcome

# Pre-requisites

- Familiarity with z/OS concepts
- Awareness (not necessarily proficiency) of basic z/OS System Programming tasks
- Basic familiarity with JCL concepts (JCL skills NOT required)
- No Java, COBOL or PL/I skills are required
- Developers are welcome

# Proof of Technology Objectives

The objective of this session is to demonstrate through hands-on workshops, the power of collaborative development and delivery enabled by the Enterprise DevOps solutions.

Application teams can learn how they can modernize their application portfolio using modern technology and tools. We will focus on code and build, automate testing and deploy.

Production teams can learn how they can extend and automate their existing build and deployment solutions for enterprise applications.

## Areas covered:

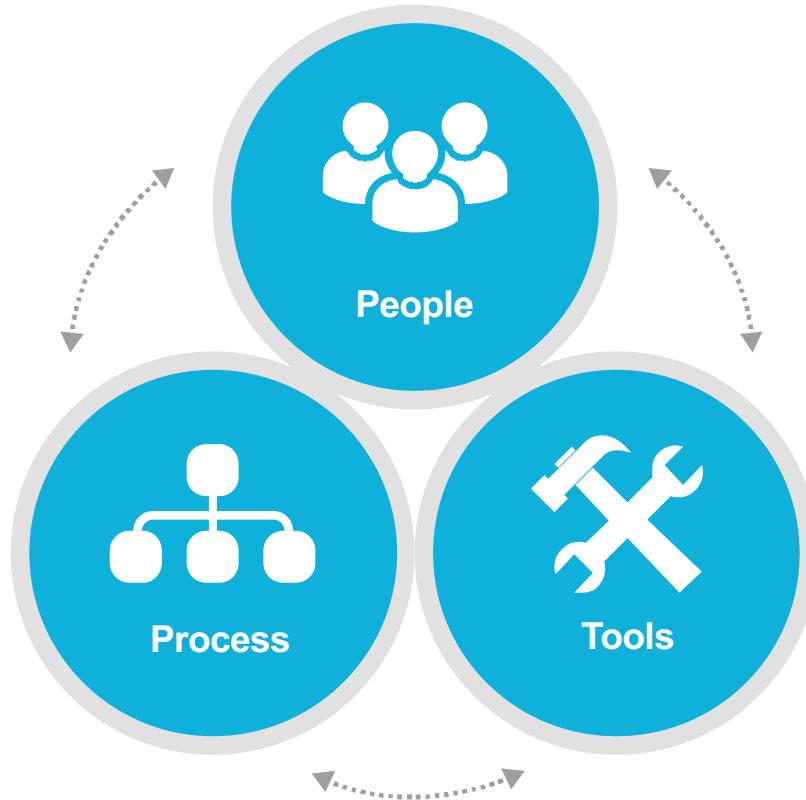
- Enterprise analytical tool for application understanding and modernization ([ADDI](#))
- Integrated application development and problem analysis ([ADFz](#))
- Managing your source code using modern technology (including [Git](#))
- Building automation ([DBB](#)) for COBOL without a specific source code manager but using a pipeline automation tool (including [Jenkins](#) )
- Using a unit testing framework ([zUnit](#)) for COBOL as part of the development and build environment.
- Using IBM Z Virtual Test Platform ([VTP](#)) for application Integration Testing of a COBOL/CICS/DB2 application
- Running Integration tests using [Galasa](#) framework
- Application deployments automation to many environments ([UCD](#))

# Introductions

- Introduce yourself and your team:
  - What's your role within your organization? (Architect, Developer, DBA, Manager, etc.)
  - Current development tools set in use? (IDz, TSO, SCLM, Changeman, Endevor)
- What do you hope to get out of this session?



# DevOps is not one of these things, it's *all* of them!



---

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the system development life cycle and provide continuous delivery with high software quality.

# Why DevOps for z/OS?

- Client Value through increased quality
  - Automated test (at any level) at code delivery and
  - Code reviews will include results of build, package and tests
- Modernizing to industry standard tools
  - Enables z/OS to leverage relevant open source tooling. E.g *Jenkins* and *Git*
  - Unlocks an ecosystem not currently available to z/OS dev \*
  - Decreased time to market
- Retention and Skills
  - New hires start out knowing some of the DevOps tools
  - Using homegrown, hard to maintain, proprietary tools is a problem
  - Lack of automation increases the time until a new team member is contributing fully
  - Current tools support not sustainable

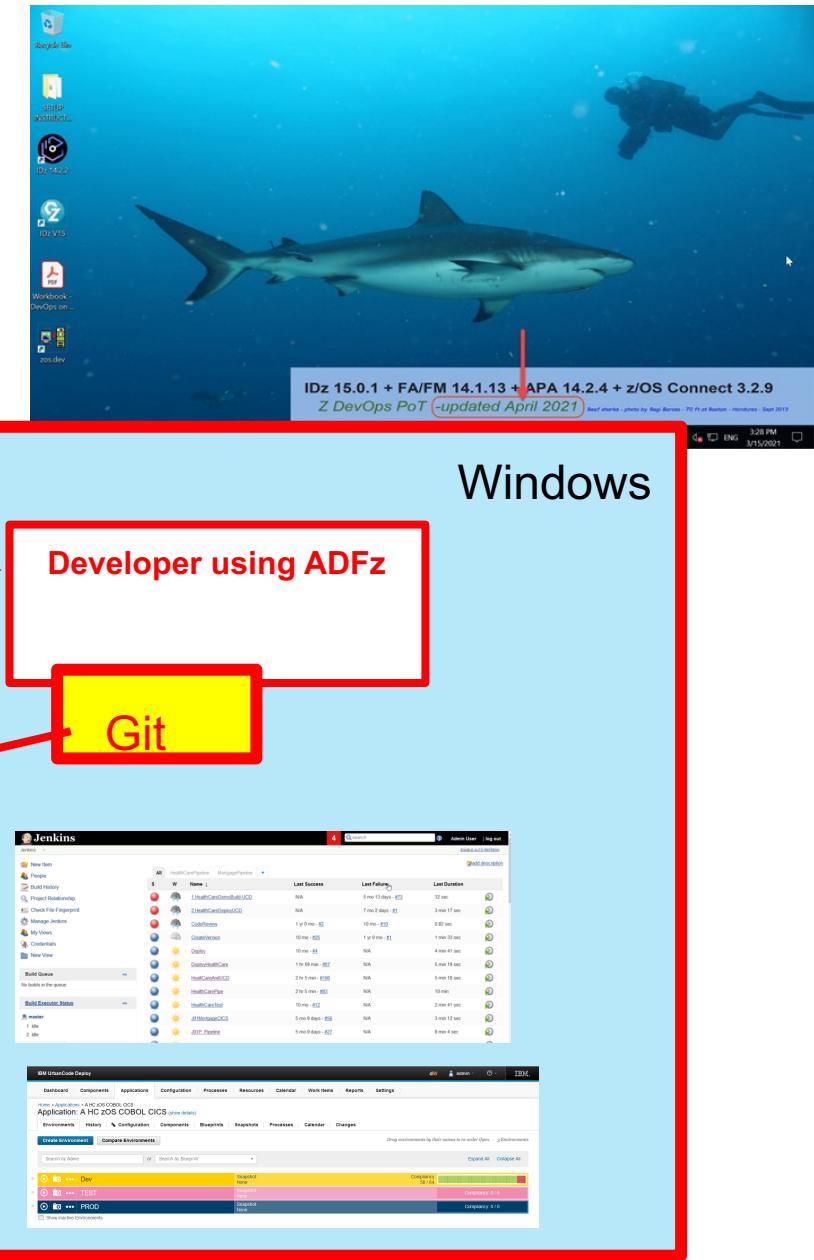
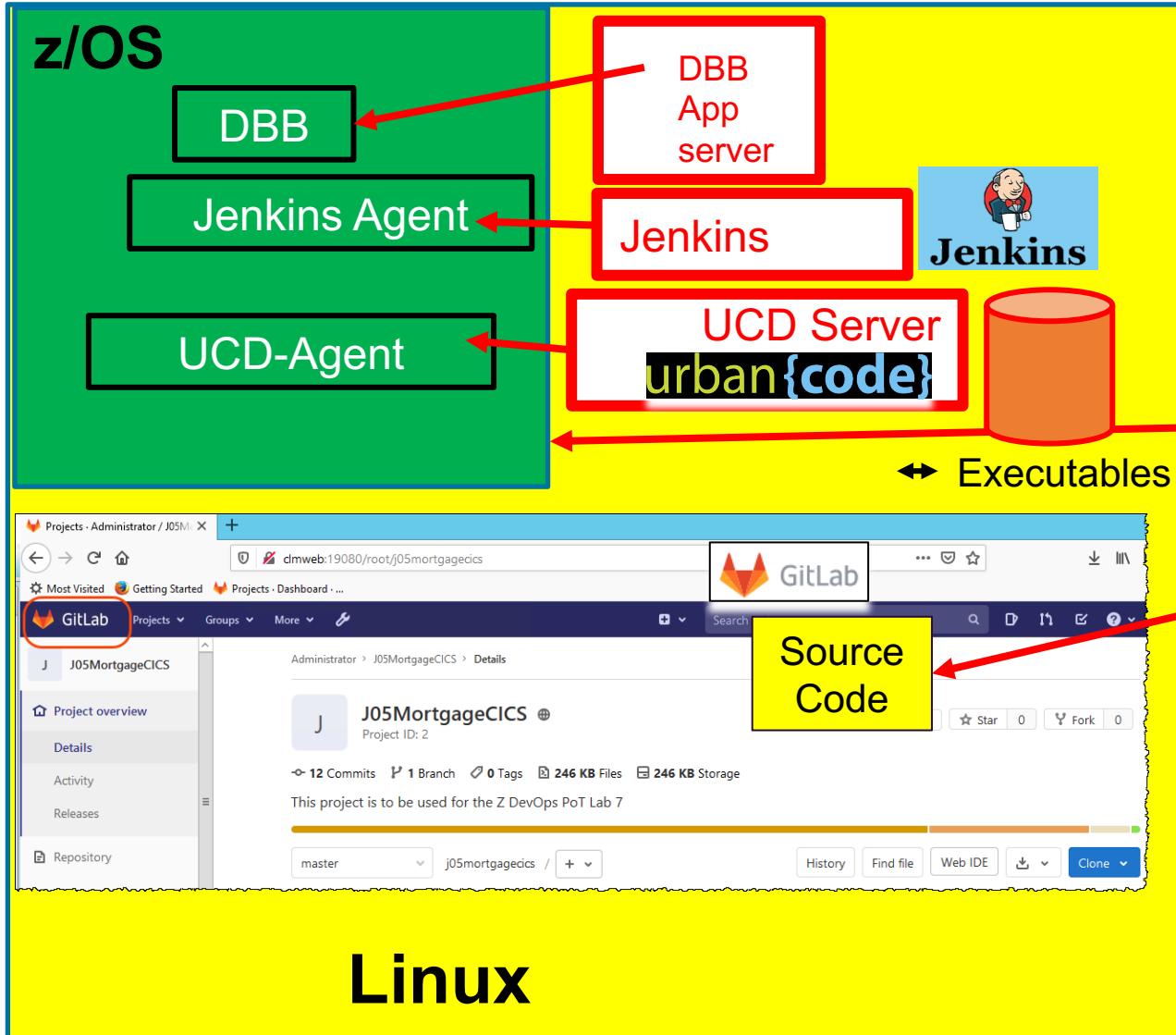
# Imagine a unified pipeline on z/OS



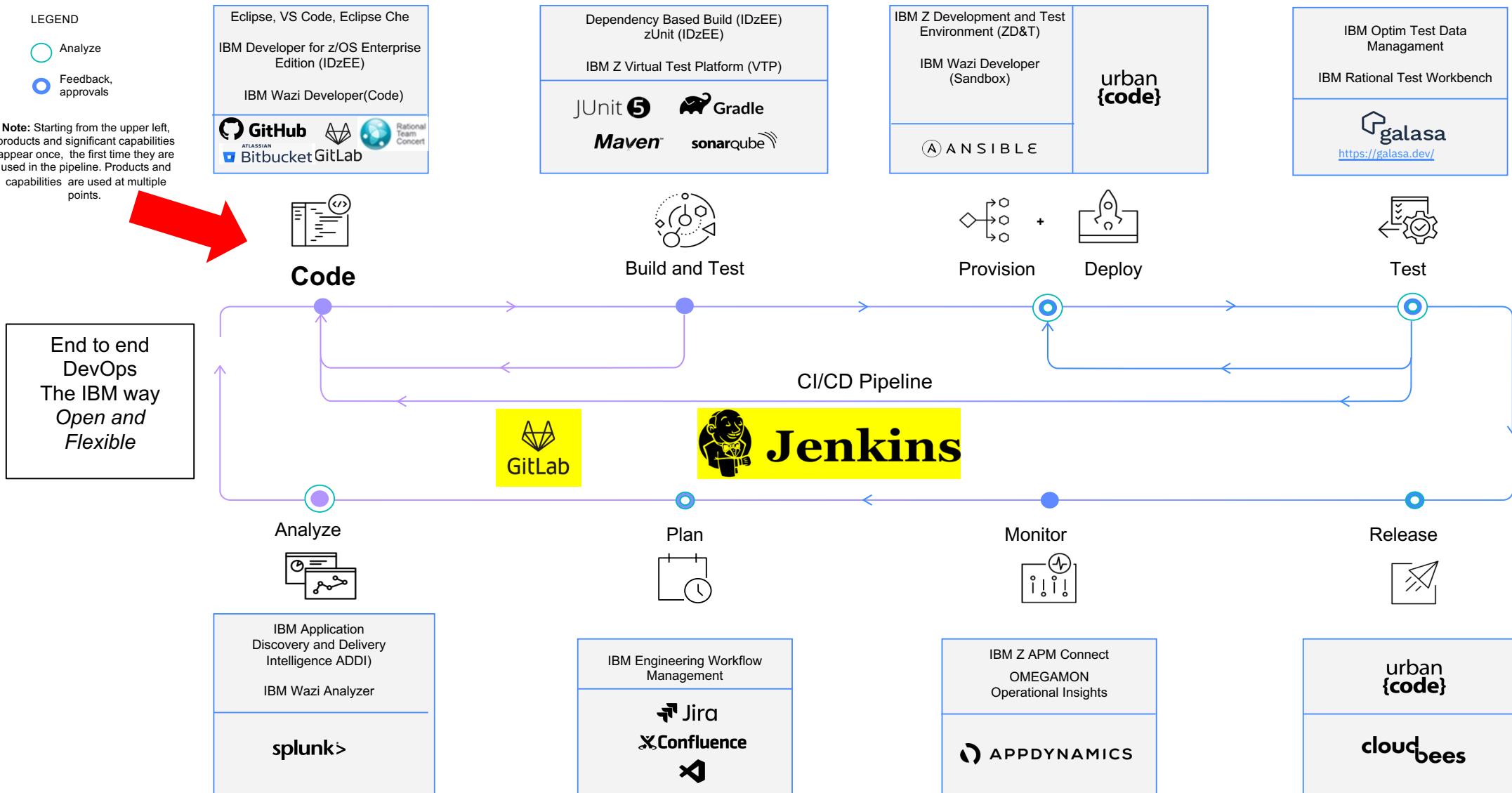
Ø → Mainframe did not use this concept

# Topology used on the labs

zD&T on Cloud



# Z DevOps Pipeline



# What is Jenkins ?

*“Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.”*

**Note:** Jenkins is not the only orchestrator. You can use Azure DevOps, TeamCity, Bambo, etc.



# Jenkins

# What is Git?

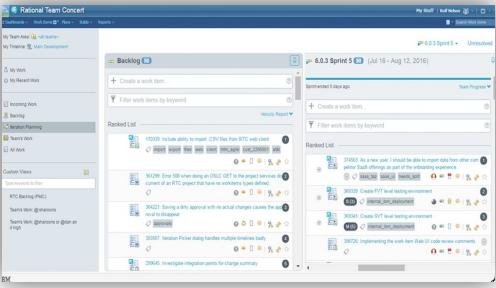
- A distributed, open-source version control system
  - Runs on all platforms (z/OS, Linux, Windows, Mac)
    - z/OS uses Rocket® open-source implementation on USS
  - Command line interface
  - Community supported
- The de-facto standard for software development
  - Integrated into many tools
    - Jenkins, Slack, Urban Code Deploy, Ansible, Artifactory, Jira, JUnit, Maven, Gradle, Make, DBB, Azure and many more.



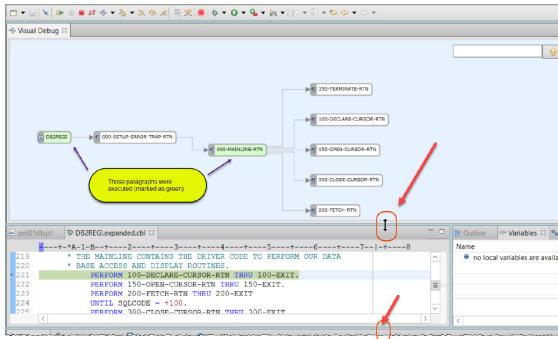
# z/OS and subsystems

## Build and SCM

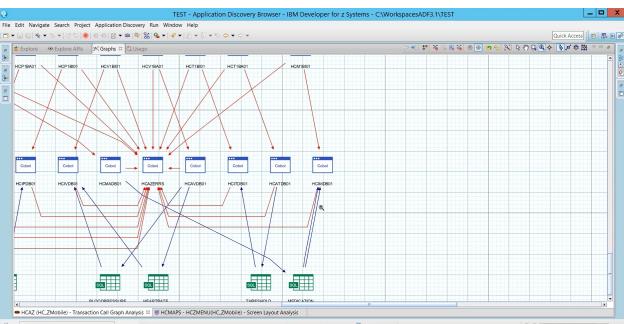
Integration with Git, RTC and Endevor for Lifecycle and Source Management



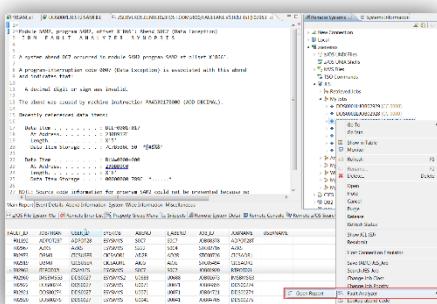
## Debug and code coverage



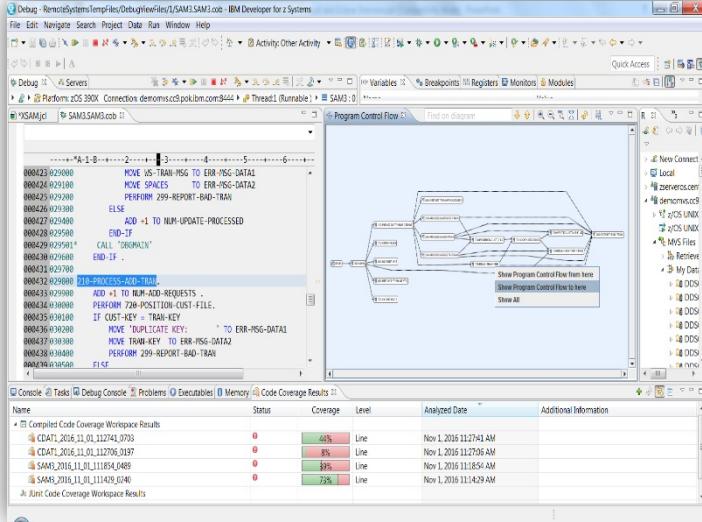
## Application Discovery



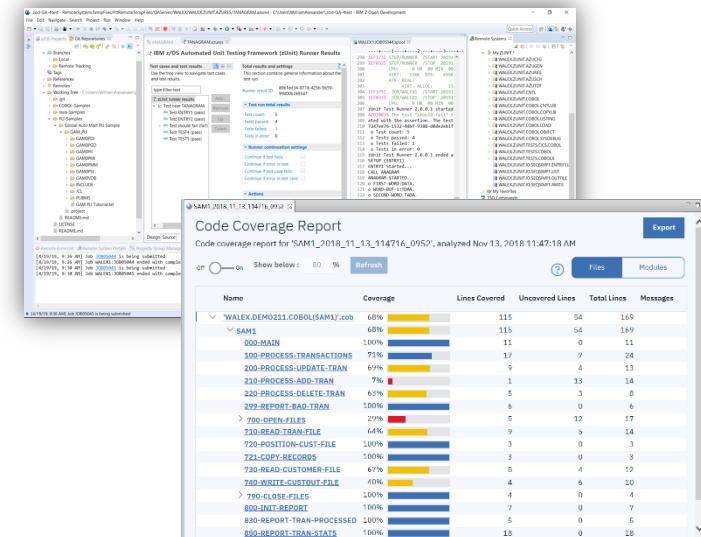
## Abend analysis



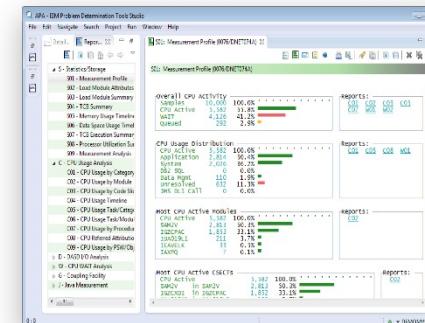
## IBM Developer for z/OS Enterprise Edition - the developer's cockpit!



## Unit test and code coverage



## Performance analysis



## File Management

|   | FILEID           | PATIENT_ID         | CARRIER_NAME | CARRIER_PHONE  | CARRIER_FAX    | TITLE | NAME              | ADDRESS                    | PHONE          | FAX            | EMAIL                 | WEBPAGE             | NOTES                 |
|---|------------------|--------------------|--------------|----------------|----------------|-------|-------------------|----------------------------|----------------|----------------|-----------------------|---------------------|-----------------------|
| 1 | 0000000000000000 | 123456789012345678 | ABC HOSPITAL | (555) 123-4567 | (555) 123-4568 | M     | Dr. John Smith    | 123 Main St, Anytown, USA  | (555) 123-4567 | (555) 123-4568 | jsmith@abc.com        | www.abchospital.org | Initial setup patient |
| 2 | 0000000000000001 | 987654321098765432 | DEF CLINIC   | (555) 987-6543 | (555) 987-6542 | F     | Dr. Jane Doe      | 456 Elm St, Anytown, USA   | (555) 987-6543 | (555) 987-6542 | jdoe@defclinic.com    | www.defclinic.org   | Initial setup patient |
| 3 | 0000000000000002 | 543210987654321098 | GHI LABS     | (555) 543-2109 | (555) 543-2108 | M     | Dr. Michael Green | 789 Pine St, Anytown, USA  | (555) 543-2109 | (555) 543-2108 | mgreen@ghilabs.com    | www.ghilabs.org     | Initial setup patient |
| 4 | 0000000000000003 | 098765432109876543 | JKL PHARMACY | (555) 098-7654 | (555) 098-7653 | F     | Dr. Linda King    | 234 Cedar St, Anytown, USA | (555) 098-7654 | (555) 098-7653 | lking@jklpharmacy.com | www.jklpharmacy.org | Initial setup patient |

# IDz - Db2 for z/OS Application Development



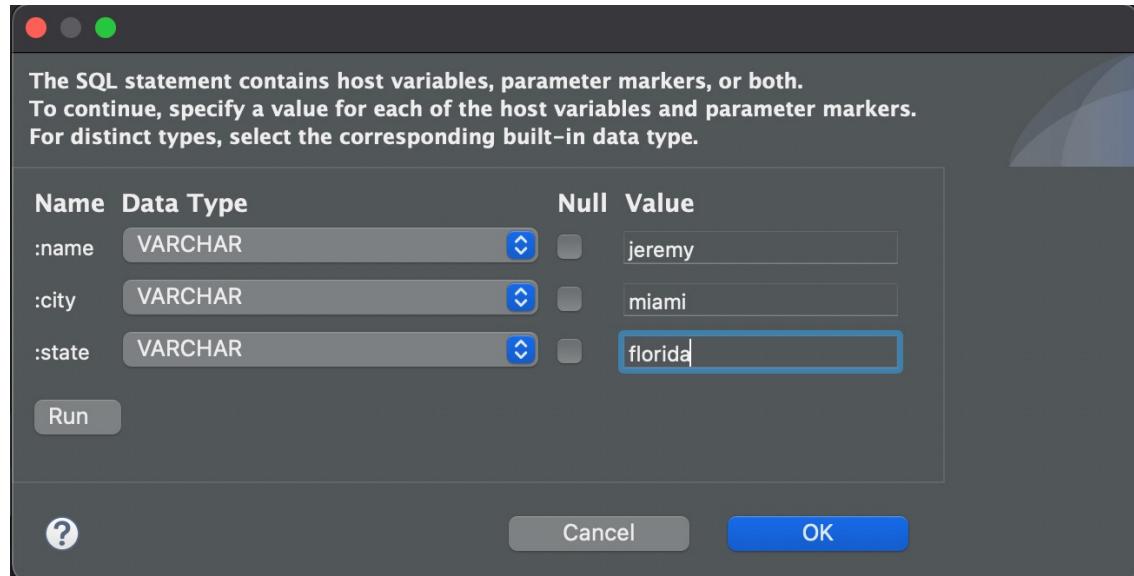
Db2 for z/OS connection integration with RSE view makes it easier to share credentials between RSE and Db2

Connections to Db2 Accessories Suite provides SQL Tuning Services such as Visual Explain

The screenshot shows the Eclipse-based Remote Systems Explorer (RSE) interface. The left sidebar lists connections to a local host and a remote z/OS system (IP address redacted). Under the remote system, there are nodes for z/OS UNIX Files, z/OS UNIX Shells, MVS Files, TSO Commands, JES, and DB2. The DB2 node is expanded, showing 'Db/2 for z/OS Connections' and a specific connection named 'STLEC1'. 'STLEC1' is selected and expanded, revealing 'Query History' containing two checked entries: 'SELECT \* FROM ADMF001' and 'SELECT NAME, CREATOR, ...'. Below 'Query History' are 'SQL Tuning Services servers' and 'Tuning Connection Profiles'.

While editing COBOL or PL/I source files users can highlight an SQL statement, run it and browse the results.

If the SQL statement contains host variables a prompt allows for input of values to use during execution.



The screenshot shows the 'Query History' view in the Db2 Accessories Suite. The query executed was:

```
SELECT NAME, CREATOR, DBNAME, TSNAME  
FROM SYSIBM.SYSTABLES WHERE CREATOR = 'ADMF001'  
AND TYPE = 'T'
```

The results are displayed in a table:

| Resource | NAME               | CREATOR | DBNAME | TSNAME   |
|----------|--------------------|---------|--------|----------|
| 0        | CBMT001_S_M_WORK   | ADMF001 | FORDDB | CBMT001R |
| 1        | CBMT002_USAGE_WRK  | ADMF001 | FORDDB | CBMT002R |
| 2        | CBMT004_PROD_GRP_P | ADMF001 | FORDDB | CBMT004R |
| 3        | CBMT005_WU_EI_WRK  | ADMF001 | FORDDB | CBMT005R |
| 4        | CBMT006_WU_AL_WRK  | ADMF001 | FORDDB | CBMT006R |
| 5        | CBMT007_STRUCT     | ADMF001 | FORDDB | CBMT007R |
| 6        | CBMT008_SUBSTUTE   | ADMF001 | FORDDB | CBMT008R |

# Source Code Management

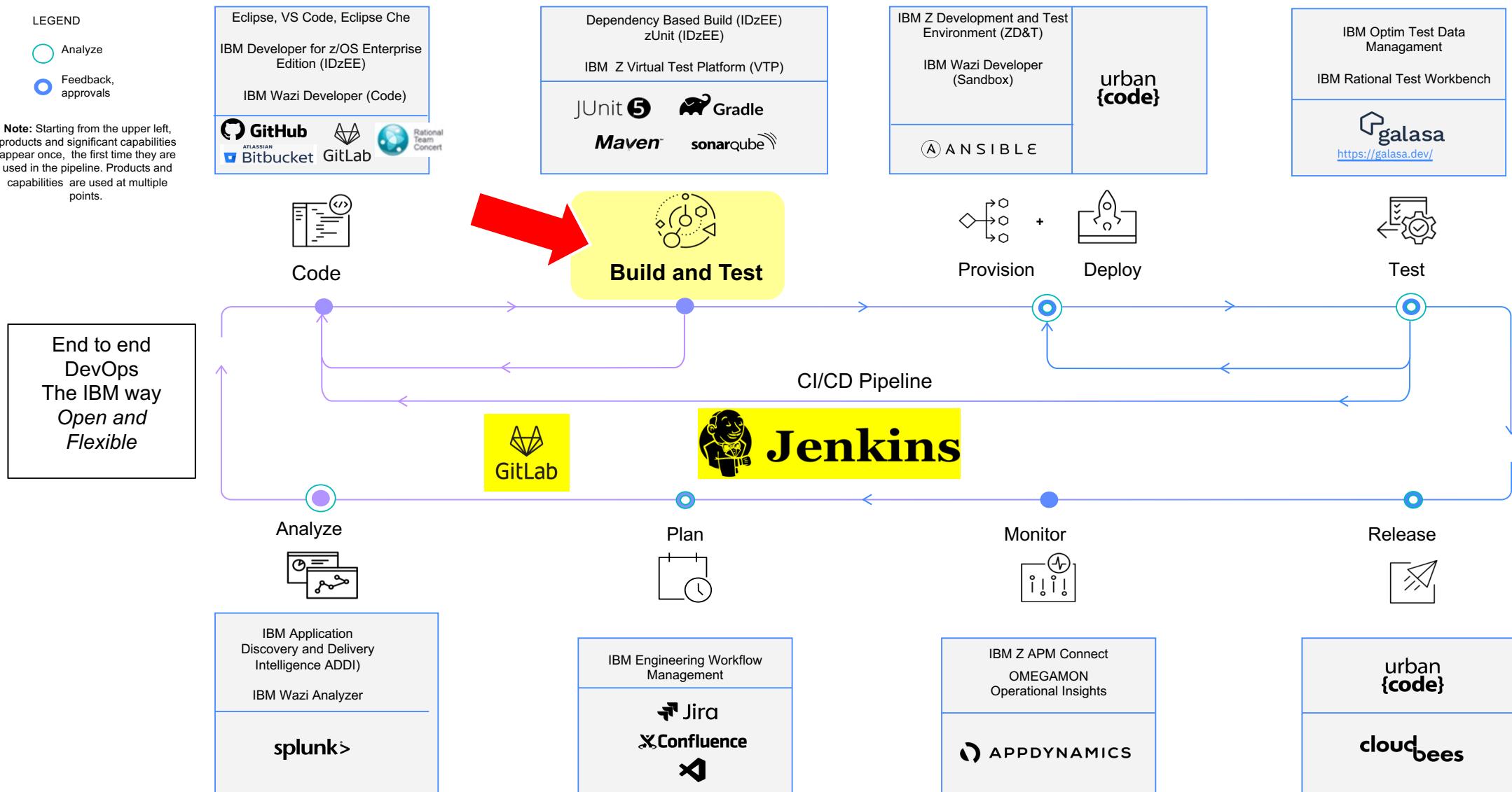


EGit included in IDz

IBM IDz offers out-of-the-box **integration** for the IBM Rational Team Concert, CA Endevor and IBM SCLM.

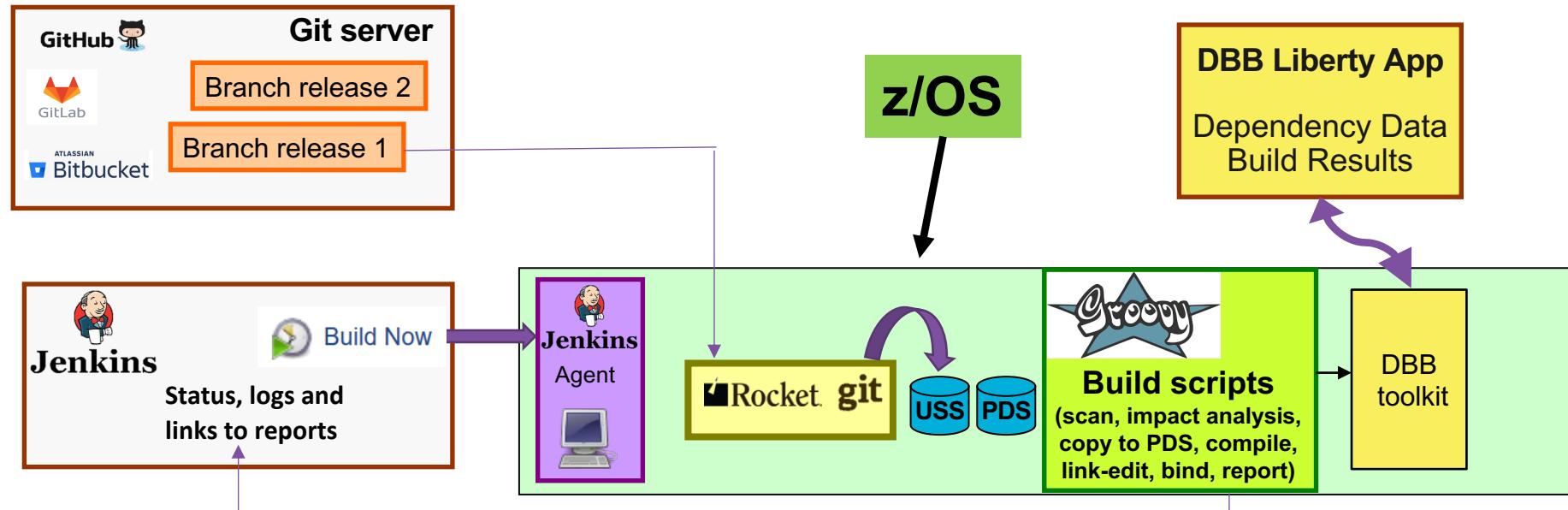
The screenshot illustrates the integration of GitHub within the IBM Integration Designer (IDz) environment. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. The main workspace shows a GitHub repository for 'RegiBrazil / DemoHealthCare'. A red box highlights the repository name. Below it, a tab bar indicates the current view: 'Code' (selected), Issues, Pull requests, Actions, and a 'sandbox' dropdown set to 'sandbox'. The central area displays a list of files: HCAPDB01.cbl, HCAPDB02.cbl, and HCATDB01.cbl, each with status indicators like 'Updated' or 'Baseline'. To the right, a large window titled 'Git Repositories' shows the file structure of the 'DemoHealthCare' repository, including '.git', 'DemoHealthCare', and various subfolders like '.settings', 'application-conf', 'bms', 'BuildOutput', 'cobol\_cics', and 'cobol\_cics\_db2'. A diff viewer is open for 'HCMADB02.cbl', comparing versions 59d1a2a and 59d1a2a. The commit message is 'Commit 59d1a2a816070038035ad43b56000d29bc5dc2b2'. The diff shows changes in the 'cobol\_cics\_db2/HCMADB02.cbl' file. Below the diff, a message states 'Pushed to DemoHealthCare - origin'. The bottom section shows a history of changes, including a push from 'sandbox' to 'origin' and a specific commit '59d1a2a: Changed (RegiBrazil on 2020-08-24 17:17:28)' that modified 'DemoHealthCare/cobol\_cics\_db2/HCMADB02.cbl'. A 'Message Details' section at the bottom right shows the repository URL: 'Repository git@github.com:RegiBrazil/DemoHealthCare.git'.

# Z DevOps Pipeline



# What is IBM Dependency Based Build (DBB)?

- Provides a modern scripting languagebased automation capability that can be used on z/OS.
  - The DBB API can be called by Java based scripting languages such as **Groovy**, JRuby, Jython, Ant, Maven, Python, etc.
- Consists of a **build toolkit** (Java API, Groovy Installation) installed on **USS (z/OS)** and a **Liberty application server** that hosts build metadata (dependency data, build results) installed on **Linux**.



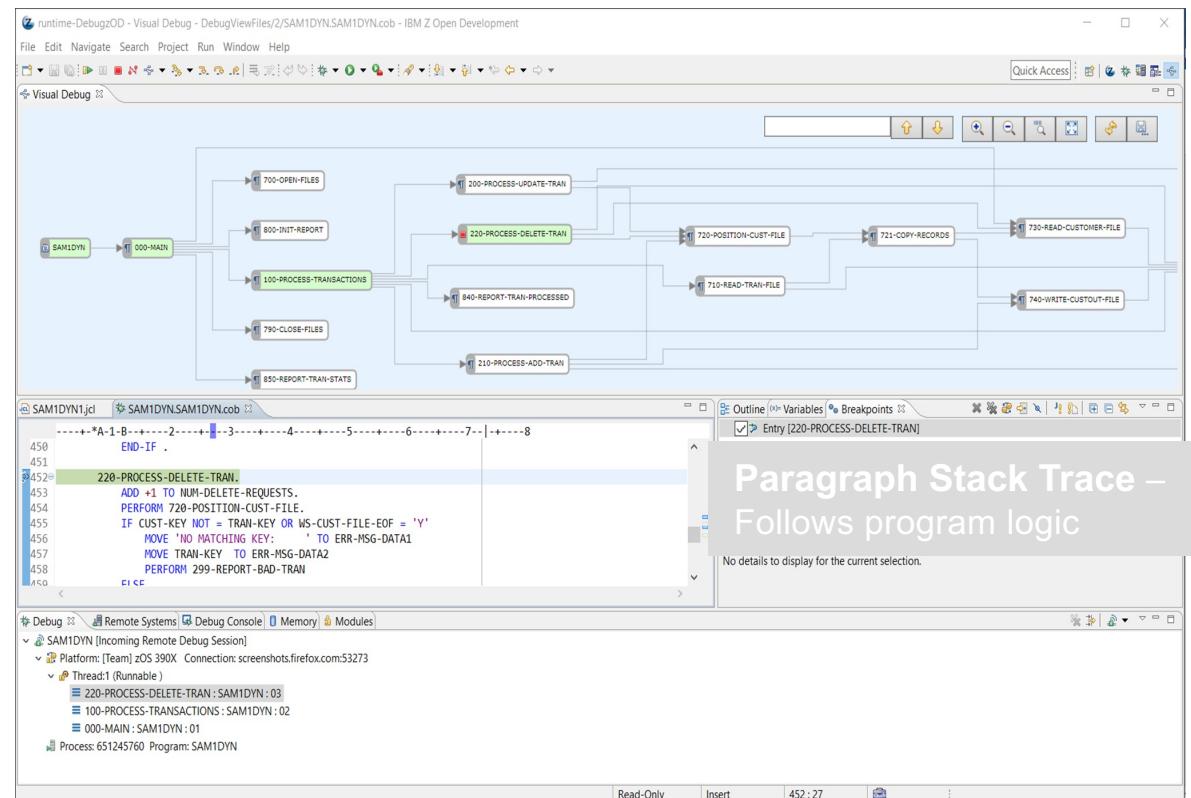
# Diagnose (z/OS Debugger)

Program testing and analysis

Helps you examine, monitor, and control the execution of application programs on z/OS

With:

- Visual debug
- Code coverage facilities
- Eclipse GUI
- 3270 UI  
(Debug for z/OS and IDZ-EE)



This screenshot shows a 3270 terminal window displaying a COBOL monitor session. The monitor is showing input data for a birthdate. The data includes fields for year ('1967'), month ('12'), and day ('01'). The monitor also displays a verification routine starting at line 436, which checks if the birthdate is numeric and less than 1582. The terminal has a green background with colored text for different levels of the monitor. At the bottom, there's a command line with PF keys and function keys.

```
COBOL LOCATION: CDAT1 :> 436.1
Command ==>
MONITOR +---+1---+---+2---+---+3---+---+4---+---+5---+---+6---+---+7---+---+8---+
***** TOP OF MONITOR ***** +---+1---+---+2---+---+3---+---+4---+
0001 1 03 W-COM-INPUT-BIRTHDATE
0002 04 W-COM-INPUT-DATE-CCYY
0003
0004 04 W-COM-INPUT-DATE-MM
0005 04 W-COM-INPUT-DATE-DD
0006 2 03 COM-INPUT-DATE
0007 04 COM-INPUT-DATE-CCYY
0008 04 COM-INPUT-DATE-MM
0009 04 COM-INPUT-DATE-DD
SOURCE CDAT1 +---+1---+---+2---+---+3---+---+4---+---+5---+---+6---+---+7---+---+8---+
LINE: 1 OF 9
434
435 0500-VERIFY-INPUT-DATE.
436 IF W-COM-INPUT-BIRTHDATE NUMERIC
437 MOVE W-COM-INPUT-BIRTHDATE TO COM-INPUT-DATE, L-INPUT-
438 MOVE 'OK' TO W-DATE-VALID-SW
439 EVALUATE TRUE
440 WHEN W-COM-INPUT-DATE-CCYY < 1582
PF 1:?: 2:STEP 3:QUIT 4:LIST 5:FIND 6:AT/CLEAR
PF 7:UP 8:DOWN 9:GO 10:ZOOM 11:ZOOM LOG 12:RETRIEVE
MA a
LINE: 434 OF 505
02/015
```

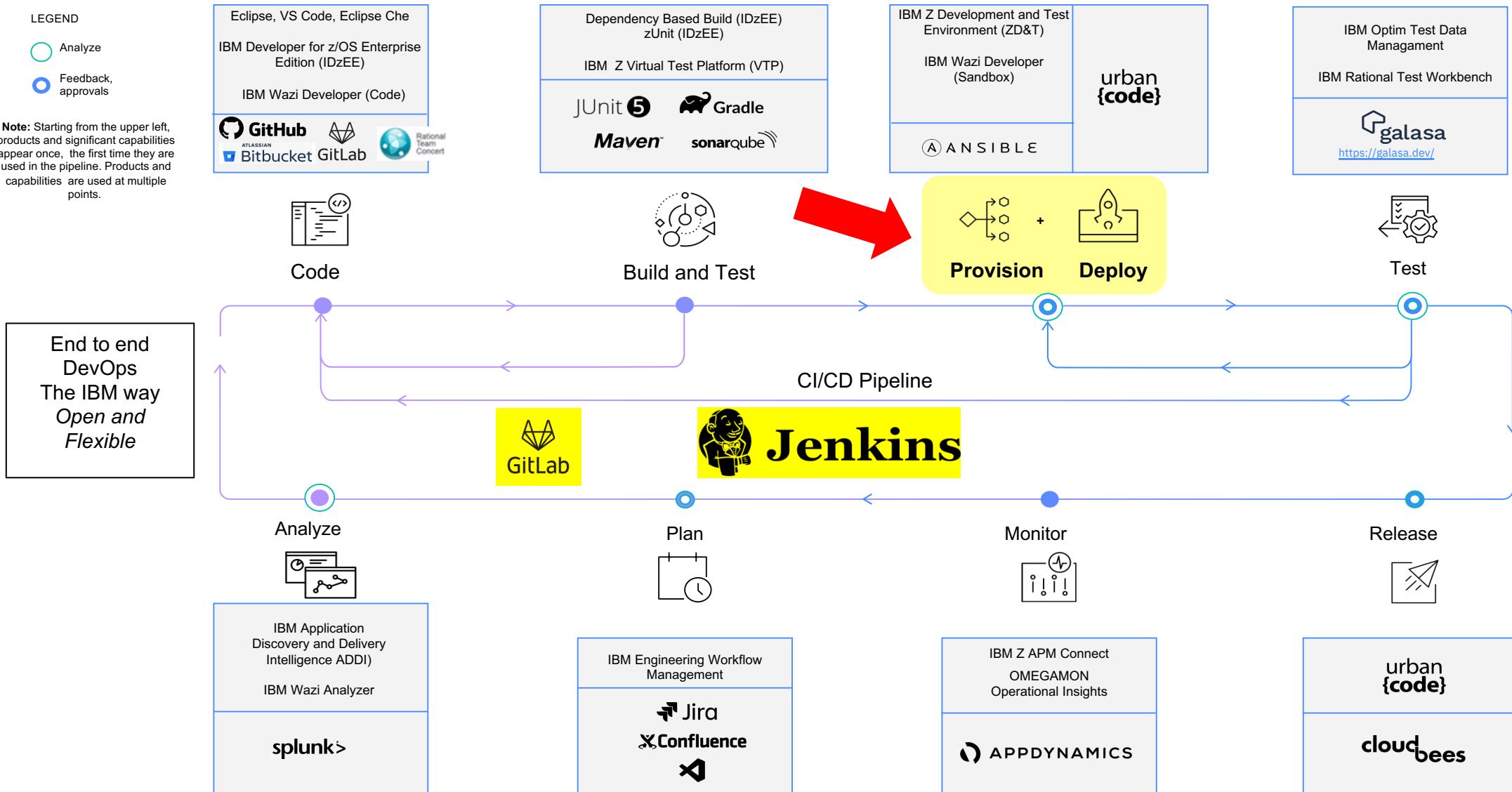
# Test (code coverage)

- Tracks lines of code that have been executed during test
- Improves application testing quality
- Focuses testing resource usage
- Numerous reports and options to assess testing and trends
  - Source view
  - Summary
  - Comparisons
  - Merging
- Supports: Batch, CICS and IMS
- Integration with ADDI and SonarQube
- Run in batch as part of your DevOps pipeline, with RESTful APIs for Debug profile creation

The screenshot displays the 'Code Coverage Results' interface. A context menu is open over a list of coverage workspace entries, with the 'Compare' option highlighted. Below this, a 'Compare results' dialog is shown with the title 'Coverage'. The dialog contains a code editor window displaying COBOL source code for 'WALEX.DEMO211.COBOL(SAM1).cob'. The code includes various PERFORM and IF statements. To the right of the dialog is a 'Code Coverage Comparison Report' table. The table compares two workspaces: 'SAM1\_2018\_12\_01\_144006\_0842' and 'SAM1\_2018\_11\_13\_114716\_0952'. The columns include Name, Coverage, Lines Covered, Uncovered Lines, Total Lines, and Messages. Coverage percentages range from 29% to 100%, with some sections showing significant differences or no coverage at all.

| Name                            | Coverage    | Lines Covered | Uncovered Lines | Total Lines | Messages |
|---------------------------------|-------------|---------------|-----------------|-------------|----------|
| 'WALEX.DEMO211.COBOL(SAM1).cob' | 75% (+7) ↑  | 115 → 127     | 54 → 42         | 169         |          |
| SAM1                            | 75% (+7) ↑  | 115 → 127     | 54 → 42         | 169         |          |
| 000-MAIN                        | 100% (0) ○  | 11            | 0               | 11          |          |
| 100-PROCESS-TRANSACTIONS        | 75% (+4) ↑  | 17 → 18       | 7 → 6           | 24          |          |
| 200-PROCESS-UPDATE-TRAN         | 77% (+8) ↑  | 9 → 10        | 4 → 3           | 13          |          |
| 210-PROCESS-ADD-TRAN            | 79% (+72) ↑ | 1 → 11        | 13 → 3          | 14          |          |
| 220-PROCESS-DELETE-TRAN         | 63% (0) ○   | 5             | 3               | 8           |          |
| 299-REPORT-BAD-TRAN             | 100% (0) ○  | 6             | 0               | 6           |          |
| 700-OPEN-FILES                  | 29% (0) ○   | 5             | 12              | 17          |          |
| 710-READ-TRAN-FILE              | 64% (0) ○   | 9             | 5               | 14          |          |
| 720-POSITION-CUST-FILE          | 100% (0) ○  | 3             | 0               | 3           |          |
| 721-COPY-RECORDS                | 100% (0) ○  | 3             | 0               | 3           |          |
| 730-READ-CUSTOMER-FILE          | 67% (0) ○   | 8             | 4               | 12          |          |
| 740-WRITE-CUSTOUT-FILE          | 40% (0) ○   | 4             | 6               | 10          |          |
| 790-CLOSE-FILES                 | 100% (0) ○  | 4             | 0               | 4           |          |
| 800-INIT-REPORT                 | 100% (0) ○  | 7             | 0               | 7           |          |
| 830-REPORT-TRAN-PROCESSED       | 100% (0) ○  | 5             | 0               | 5           |          |

# Z DevOps Pipeline

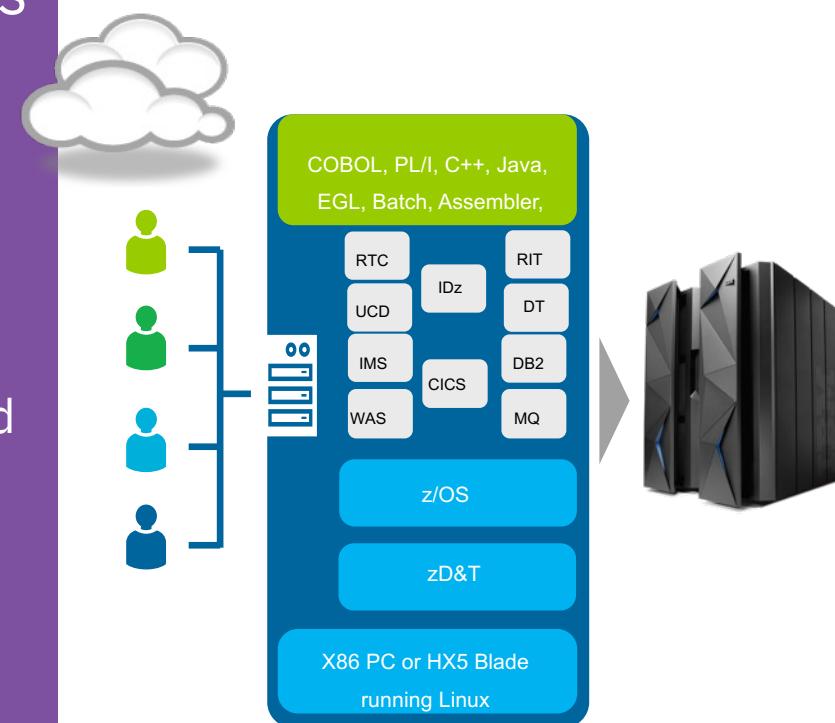


# Provisioning

## —Shift left with **Z Development and Test Environment**

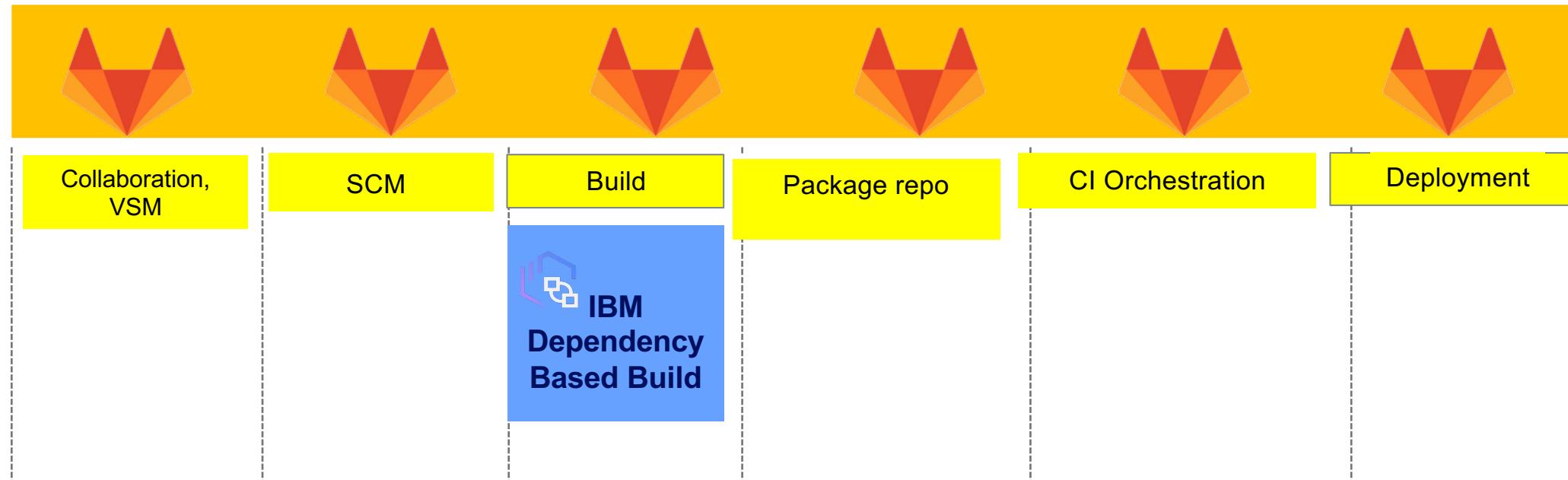
Develop and test IBM Z applications anywhere, anytime with z/OS optimized for x86 hardware

- **zD&T tools to accelerate provisioning and image management**
- Cloud friendly, software-based licensing for enterprise customers (managed service on IBM Cloud)
- Latest z/OS 2.4 software and middleware
- ***Developer autonomy just like distributed, web and mobile developers!***



# DevOps mainframe pipeline with GitLab Ultimate for z/OS

One solution provides: Project Management, SCM, CI/CD, issue tracking, container registries, logging, dependency scanning and license management. Many IBM, 3rd party and open source tools integrate.



## Extend GitLab with Specialists



Jenkins



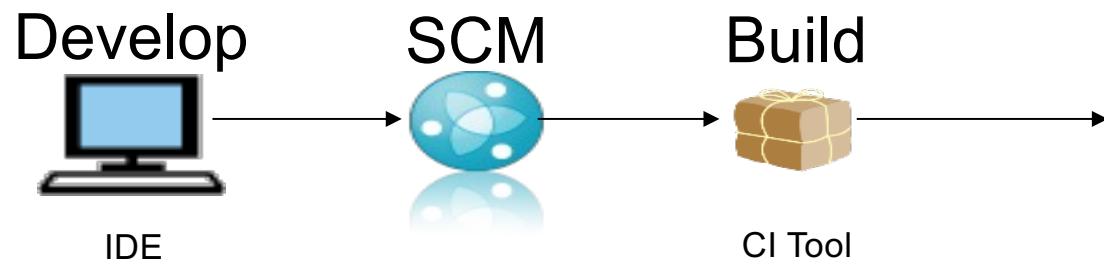
IBM  
Rational  
Test

urban{code}

Velocity (VSM, data,  
pipeline orchestration)

# UrbanCode for Deployment automation (UCD)

FROM MAINFRAME TO I-SERIES TO DISTRIBUTED TO CLOUD-NATIVE



**IBM UrbanCode Deploy** automates the deployment of applications, databases and configurations into development, test and production environments, helping to drive down cost, speed time to market with reduced risk.

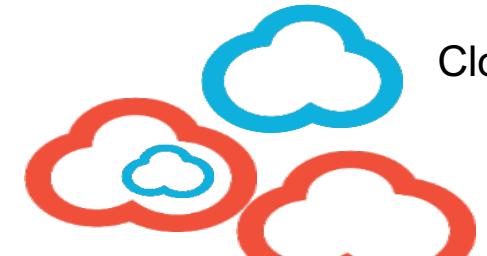
Deployment orchestration for Distributed and z/OS Applications – including CICS, IMS, Db2, MQ, across multiple environments

Integrates with Jenkins

Manages artifact versions via an in-built repository called codestation



Mobile Device



Cloud: Public /  
Private /  
Hybrid



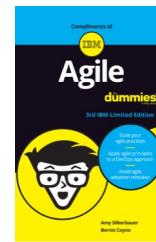
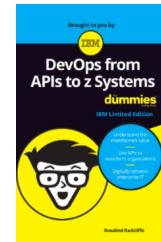
Traditional



Mainframe

# Additional Resources

- Mainframe CI/CD with an open toolchain:
  - <https://www.linkedin.com/pulse/mainframe-cicd-open-toolchain-its-real-spectacular-minaz-merali/>
- Mainframe Dev Center:
  - [Downloads - Mainframe DEV \(ibm.github.io\)](https://ibm.github.io/Mainframe-DEV/)
- IBM DevOps for Enterprise Systems:
  - <https://www.ibm.com/it-infrastructure/z/capabilities/enterprise-devops>
- For Dummies books:
  - <https://www.ibm.com/ibm/devops/us/en/resources/dummiesbooks/>
- IBM Z Trial:
  - <https://ibm.biz/z-trial>



# QUESTIONS?

