GUIDE SHARE EUROPE UK REGION

GSE UK Conference 2019
Dock into the Dark Side

Making the Mainframe a Millennial Magnet

Gerald Pfeiffer
Gerald.Pfeiffer@Broadcom.com

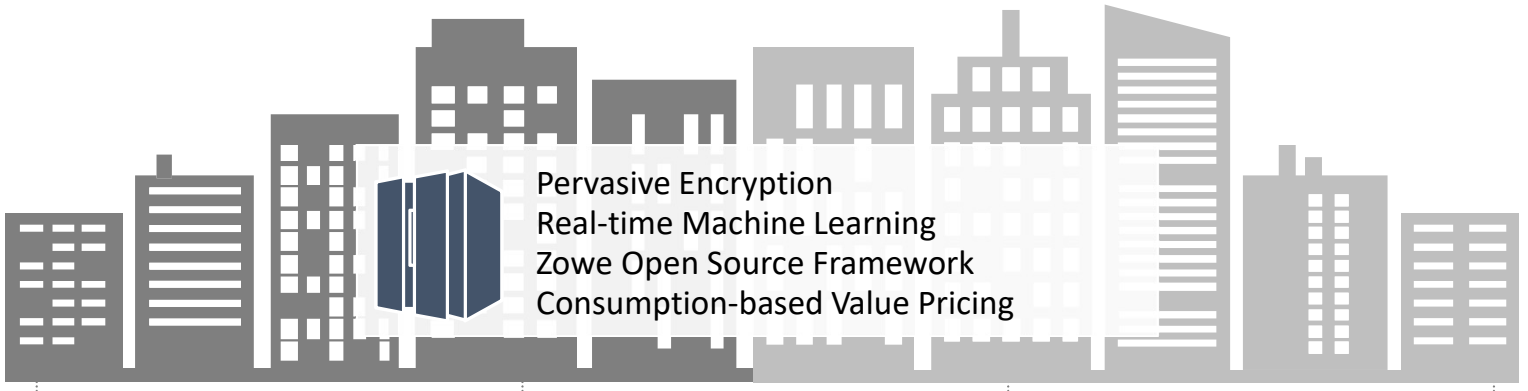
November 7th, 2019 at 11:45
Session AO

Agenda

- Making the Mainframe a Millennial Magnet:
  - attracting and retaining the next generation of top talent
  - open IBM Z to a new generation of developers
  - how can we hope to attract and retain the next generation of top talent to work on mainframe, a technology often unknown, ignored or misunderstood by millennials?
  - empowering development teams to work across siloes and use a common set of cross-enterprise tools across the software delivery lifecycle, from plan, build and test to deploy, operate and secure.

Mainframe – The Backbone of Business

Has Become a Platform for Innovation



84%

Of mainframe orgs experienced **MIPS growth** [2]

68%

Of the world's **production workloads** run on mainframes [1]

87%

Of all global **credit card transactions** supported by mainframes [1]

92

Of the world's **top 100 banks** rely on mainframe [3]

1. IBM Mainframe ushers in a new era of data protection, July 2017: https://www-03.ibm.com/press/uk/en/pressrelease/52824.wss
2. New: Arcati Mainframe Yearbook 2019: http://www.arcati.com/newyearbook19/newyearbook.pdf
3. SHARE, "Mainframe Matters: How Mainframes Keep the Financial Industry Up and Running", April 2018: https://www.share.org/blog/mainframe-matters-how-mainframes-keep-the-financial-industry-up-and-running

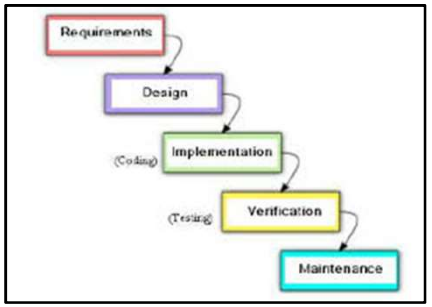
Mainframe Challenges (Existential)

People

As the mainframe workforce continues to age and retire, a skills shortage is looming

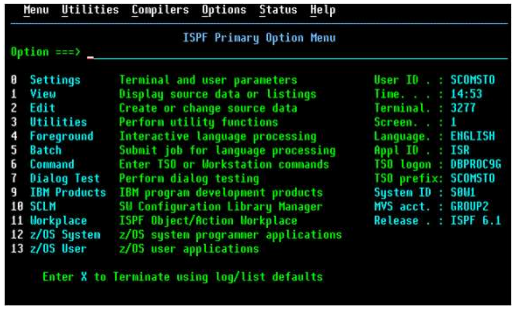
Process

Old ways of working, once state of the art, are now outdated



Technology

Traditional tools don't appeal to next gen developers nor provide high productivity



IDE trend in a Mainframe Developer's World

What is available for today's developers to use if they are working on the Mainframe part of their application?

Eclipse-based IDEs that are customized for mainframe development

Pro's	Con's
Rich functionality	Rich Client – taking HW/DASD
Easy to use	Difficult to maintain and deploy
Integrated with other tools	Slow response time

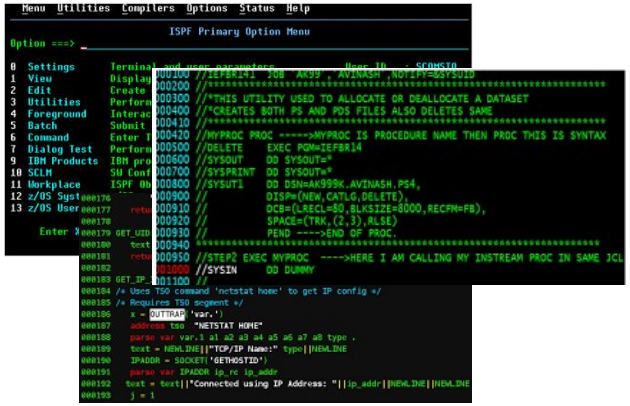
TSO/ISPF interfaces

JCL/Rexx for build and automated system testing

Pro's	Con's
Rich functionality	Limited graphical capabilities
Integrated with zOS	Not similar to modern sw developer tools
Quick response time	No integration with other platform tools

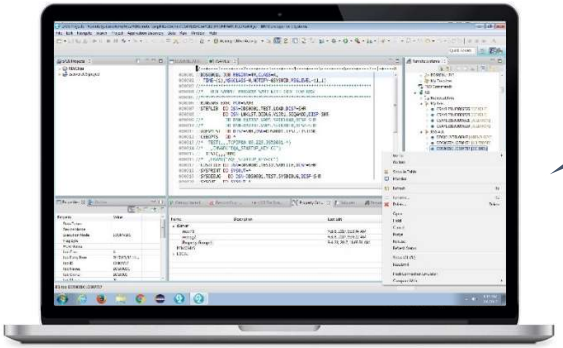
Mainframe Developer Experience needs an overhaul

Past



- TSO/ISPF interfaces
- JCL/Rexx for build and system testing
- Platform-dependent tools

Present



- Workstation-based Eclipse IDE
- Vendor plug-ins
- Proprietary tools

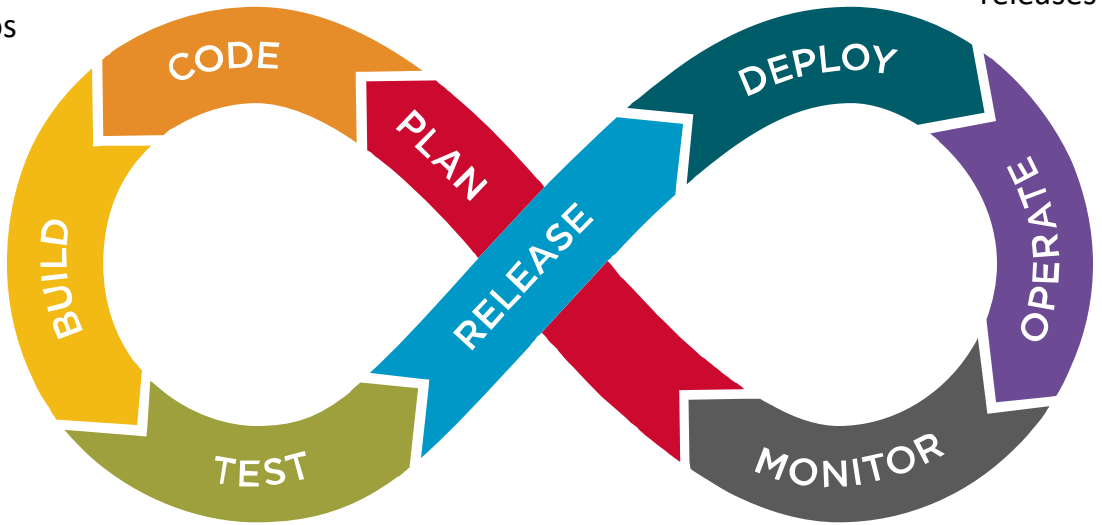
Primitive by today's standards

Customer Expectations

What is DevOps?

Developer tools deliver code insights and facilitate collaboration across silos

Code is **Continuously Integrated** and **Continuously Delivered** in a release pipeline for more frequent releases



Testing is automated to reduce cycle times and defect costs

Ops activities shift-left to bring in feedback loops and performance analytics earlier in the process

Mainframe / Enterprise DevOps Integration

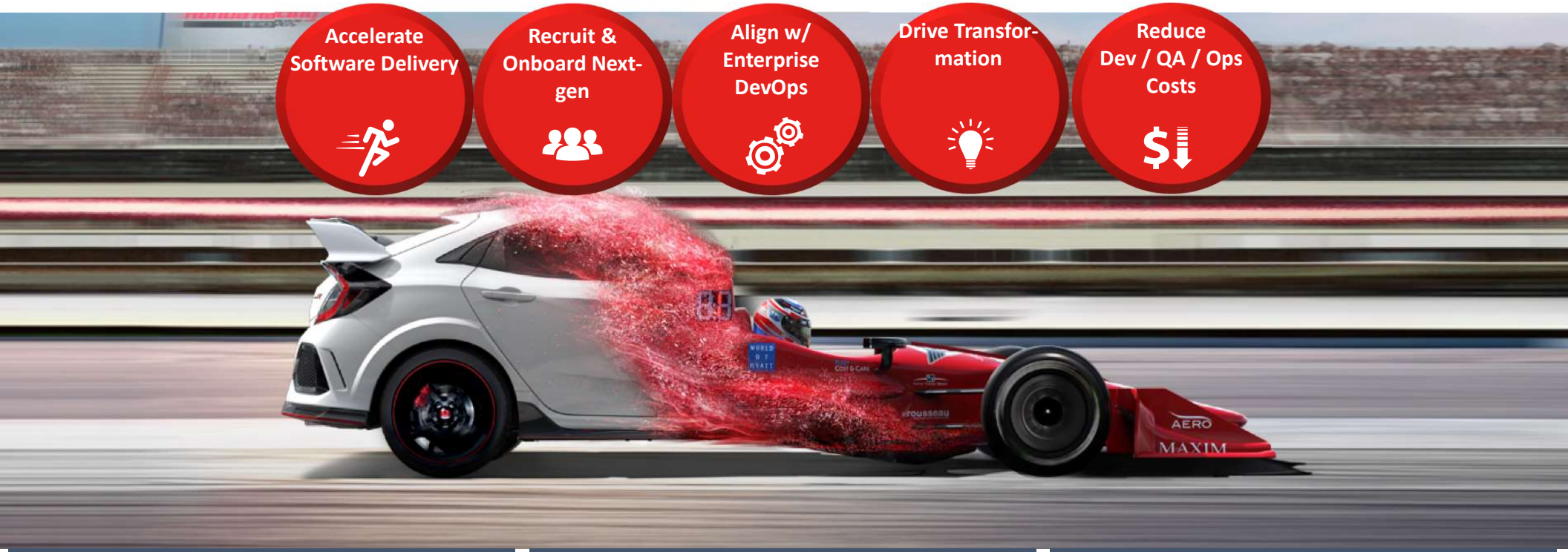
93% believe lack of enterprise DevOps integration is having an impact on the business

18% believe they have any integration between enterprise & mainframe DevOps

Source: DevOps.com, Bridging DevOps to the Mainframe, Mar 2019

Customer Challenge: Modernizing at Speed

Innovating while keeping the business running

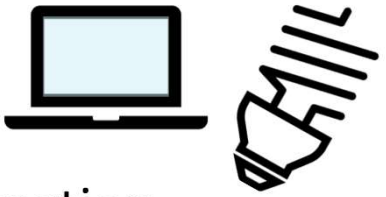


PEOPLE	PROCESS	TECHNOLOGY

A non Mainframe Developer's World

What is available for today's developers to use?

- Lightweight text-editors for quick editing with community plugins
- Powerful IDEs for specialized languages
- CLIs to interact with services
- Choice of powerful scripting languages for build and automation
- Continuous Integration and Delivery orchestration tools



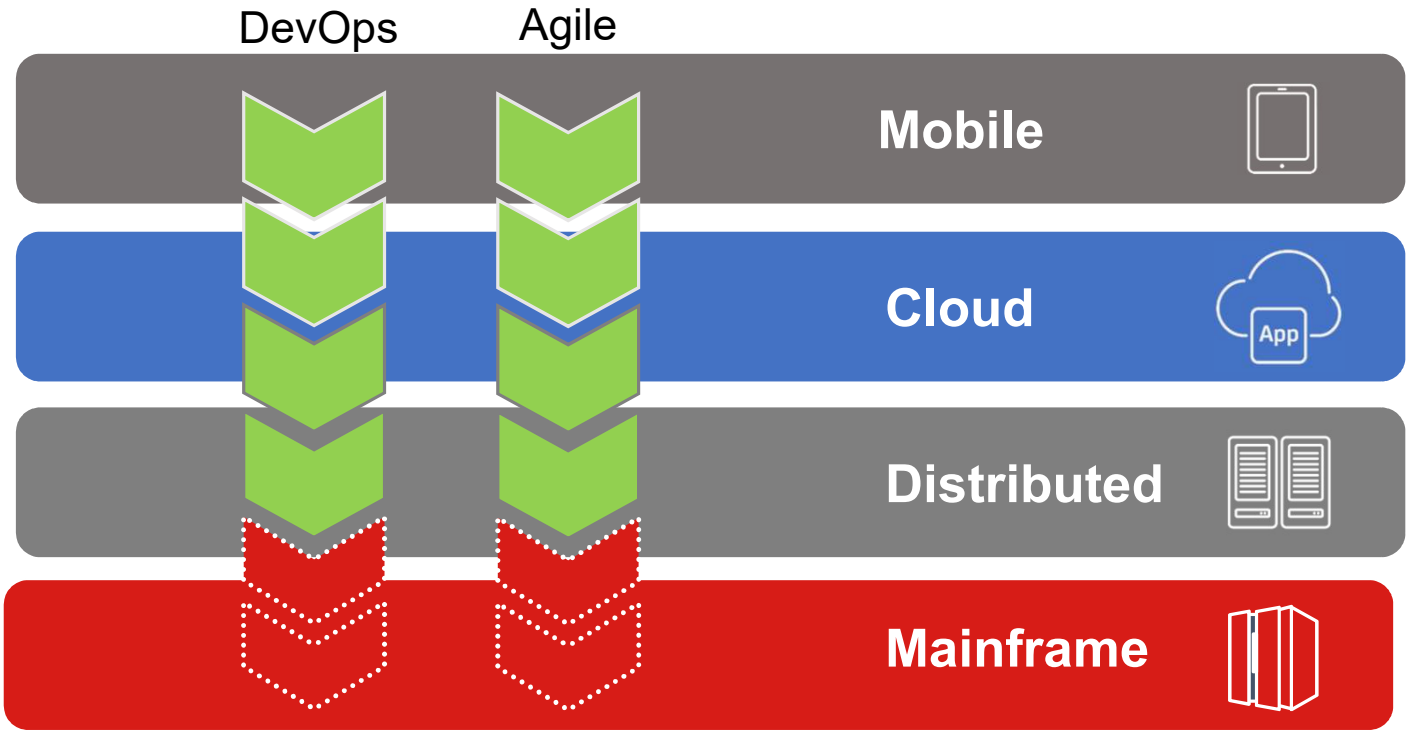
IntelliJ IDEA · Gradle · Gulp · CAST · python · sonarqube · Bitbucket · GitHub · Jenkins · JIRA · slack · git · Visual Studio · docker

Extending Modern DevOps Practices to Mainframe

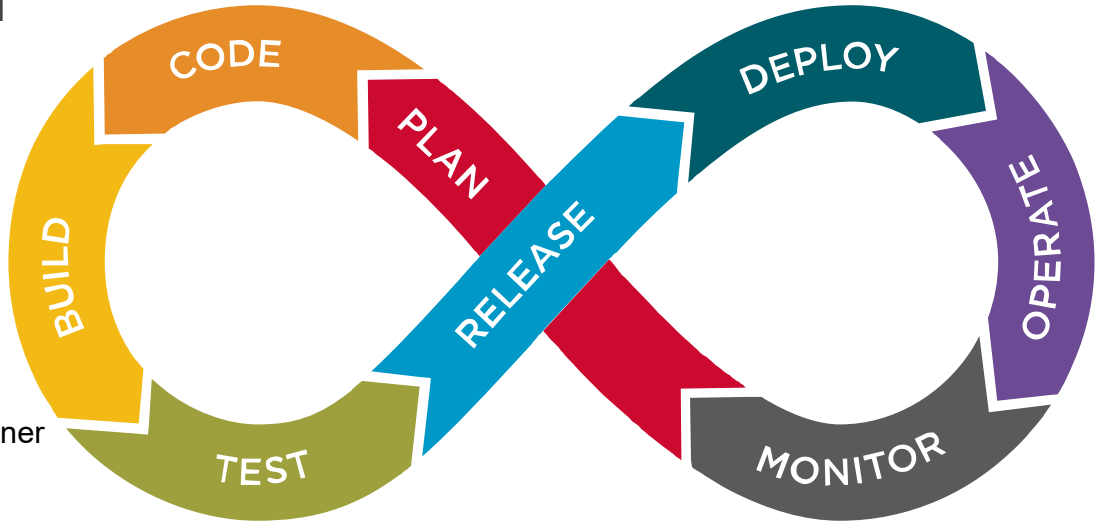
Our Guiding Principle: Mainframe as Easy as Cloud

Open Source DevOps Tools:

Jenkins, Ruby, node, Ansible, python, Gulp, JMeter, Visual Studio, Gradle, Selenium, git, chai, mocha, IntelliJ IDEA



Broadcom Mainframe DevOps Toolchain



- CA Endevor SCM
- CA Gen
- Agile Requirements Designer
- Rally (fka CA Agile Central)
- Automic Continuous Delivery Automation
- CA Mainframe Operational Intelligence
- CA OPS/MVS
- CA AIOps
- CA Database Management for Db2 for z/OS
- CA File Master Plus
- CA InterTest
- CA Mainframe Application Tuner
- CA SymDump
- CA Verify
- Blazemeter
- Test Data Manager
- Service Virtualization
- Automic Continuous Delivery Director
- CA SYSVIEW
- CA Mainframe Application Tuner
- Application Performance Management

Strategy: Open Source Driven

- ✓ **What the next-generation knows and wants**
- ✓ **Modern DevOps built on open source**
- ✓ **Proprietary software can't keep up**
- ✓ **Mainframe has much to gain**

VS Code git Jenkins MOCHA Gulp node Gradle python

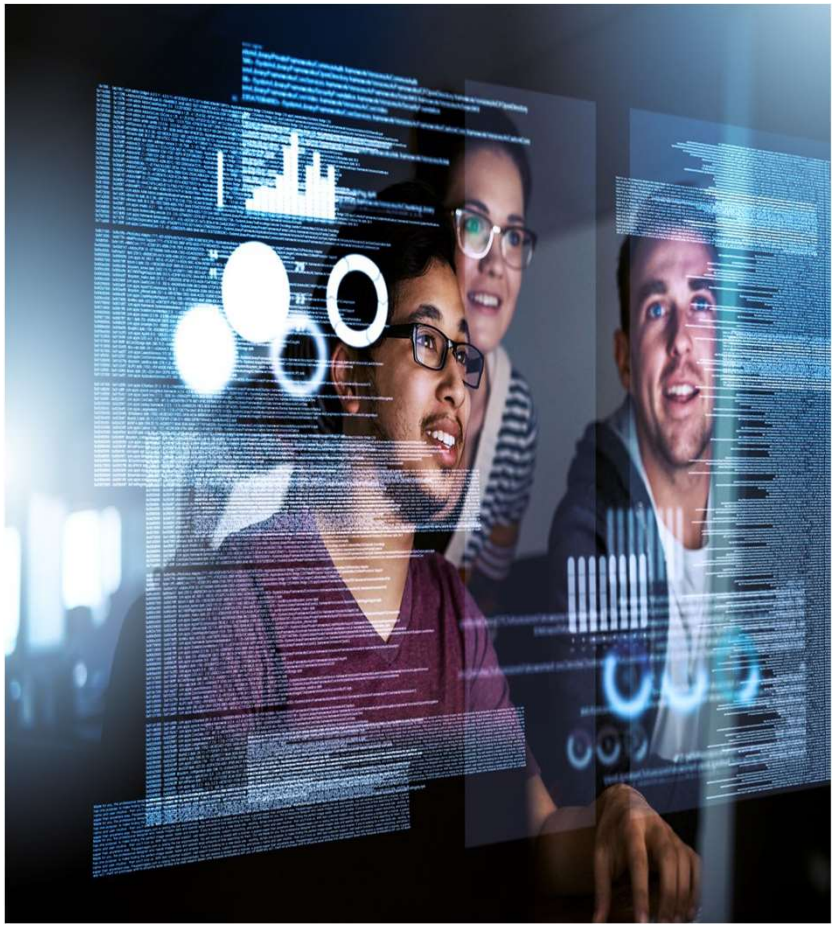
Zowe

Open source mainframe interface for the *next generation*

- ✓ Bridges enterprise DevOps to the mainframe
- ✓ Shifts the mainframe balance of power from vendors/proprietary to community/OSS
- ✓ Makes mainframe an exciting career choice

ABOUT ZOWE

- Part of Linux Foundation's Open Mainframe Project (OMP)
- First open source project based on z/OS
- Four main components:
  - **Command line interface (CLI)**
  - **API Mediation Layer**
  - **Web UI**
  - **Microservices**
- Initial contributions from IBM, Broadcom, Rocket
- Over 2,500 downloads
- v1.0 released February 11th
- Zowe.org



Zowe Vision Statement

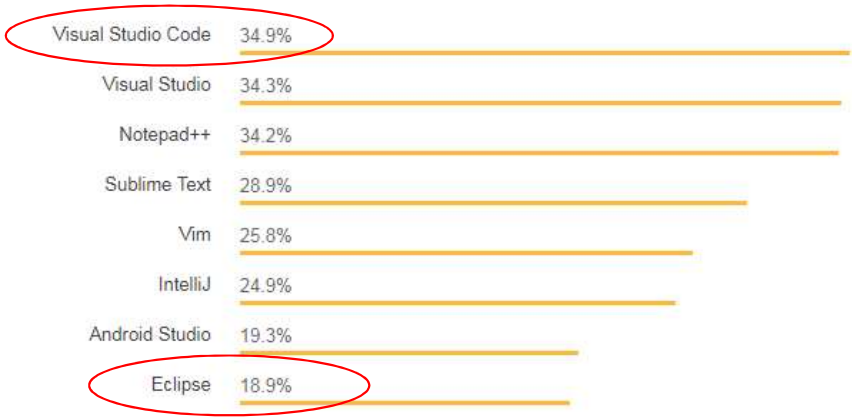
- Attract new people
  - ✓ Demystify the Z platform
  - ✓ Enhance integration and consumability
  - ✓ Promote Open community of practice
- Reduce learning curve
  - ✓ Improve productivity
  - ✓ Modern, platform-neutral interfaces
  - ✓ Cloud-like experience
- Simplify architecture
  - ✓ Reduce operational overhead
  - ✓ Improve co-existence
  - ✓ Enable rich ecosystem of free and commercial solutions

IDE Trend: VS Code Popularity

2018 Stack Overflow Survey

Most Popular Development Environments

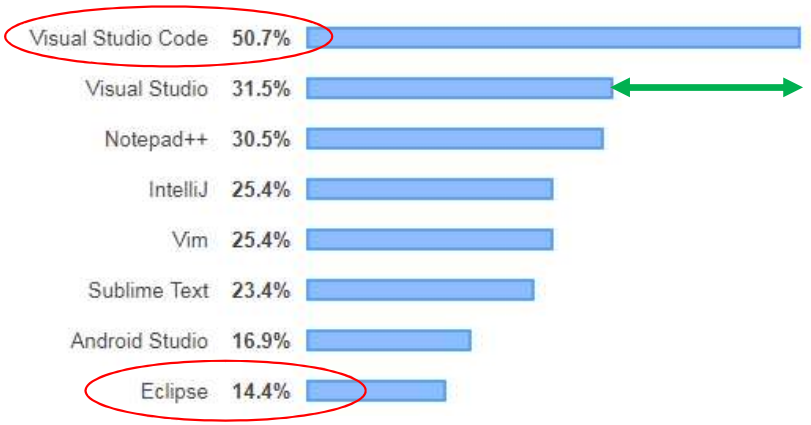
All Respondents | Web Developers | Mobile Developers | Sysadmin/DevOps



2019 Stack Overflow Survey

Most Popular Development Environments

All Respondents | Web Developers | Mobile Developers | SRE/DevOps



VS Code ↑ 45% YoY

eclipse ↓ 24% YoY

Open Source Extended

GUIDE SHARE EUROPE UK REGION

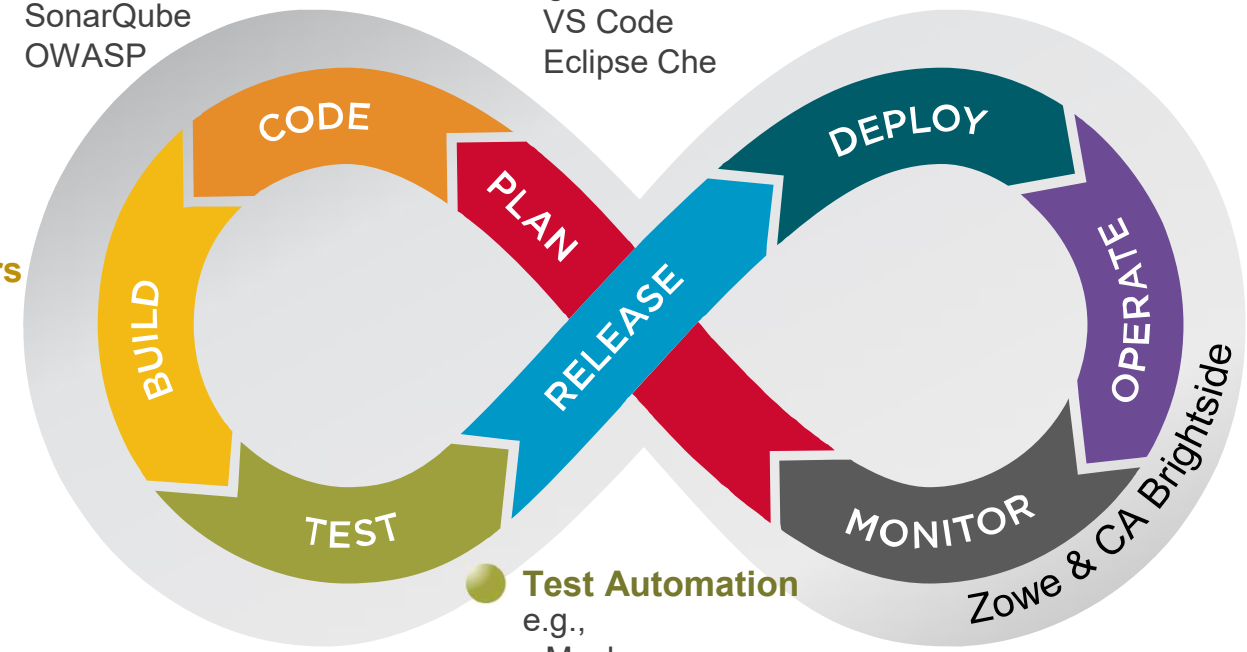
SCMs
e.g.,
Git
Github
Bitbucket
Gitlab

Code Quality
e.g.,
SonarQube
OWASP

IDEs
e.g.,
VS Code
Eclipse Che

CI/CD
e.g.,
Jenkins
Cloudbees
TravisCI
GoCD

Task Runners
e.g.,
NPM
Gradle
Gulp
Maven



Infrastructure Automation
e.g.,
SaltStack
Ansible

Scripting
e.g.,
Python
Ruby
JavaScript
Bash

Test Automation
e.g.,
Mocha
JUnit
Jest
Taurus

APM & Analytics
e.g.,
ELK Stack
InfluxData
Prometheus
Grafana

The Future …

- Flexibility and freedom to developers
  - Allow developers to work with CLI and/or their favorite IDE or Editors, to perform their development tasks with appropriate tooling
- Focus on developing common components that enable that flexibility and freedom
  - Implementation of standard protocols for editing and debugging
    - Develop once, use in any IDE/Editor that supports the protocol
  - Implementation of standard interfaces/plugins extensions (e.g for navigating resources etc.)
    - Develop once, use the extension in any IDE that supports that interface

The Future …

- Extend the open source cloud/web IDE ***Eclipse Che*** as a complete IDE solution for cross platform development
  - Delivered as containers to simplify the install and management
  - Centrally managed
  - Task oriented selection of the workspace with pre-built technology stacks
  - Leverage BOB and open source tooling
  - Access to z/OS resources and tools
- Provide VS Code extensions
- Software developers may program in multiple languages
  - Bring the right tools to the developer when they need them
  - Team sharing

Goal:

- Provide a Enterprise solution
  - for mainframe software and off-mainframe developers
  - Support Enterprise standard tool usage
  - Allow for developer flexibility
  - Leverage BOB and open source tooling

- Software developers may program in multiple languages
  - Bring the right tools to the developer when they need them
  - Team sharing

- Flexibility
  - Provide developers of choice

What about faster access to the tools I need

Saving & Sharing my work

Changing languages

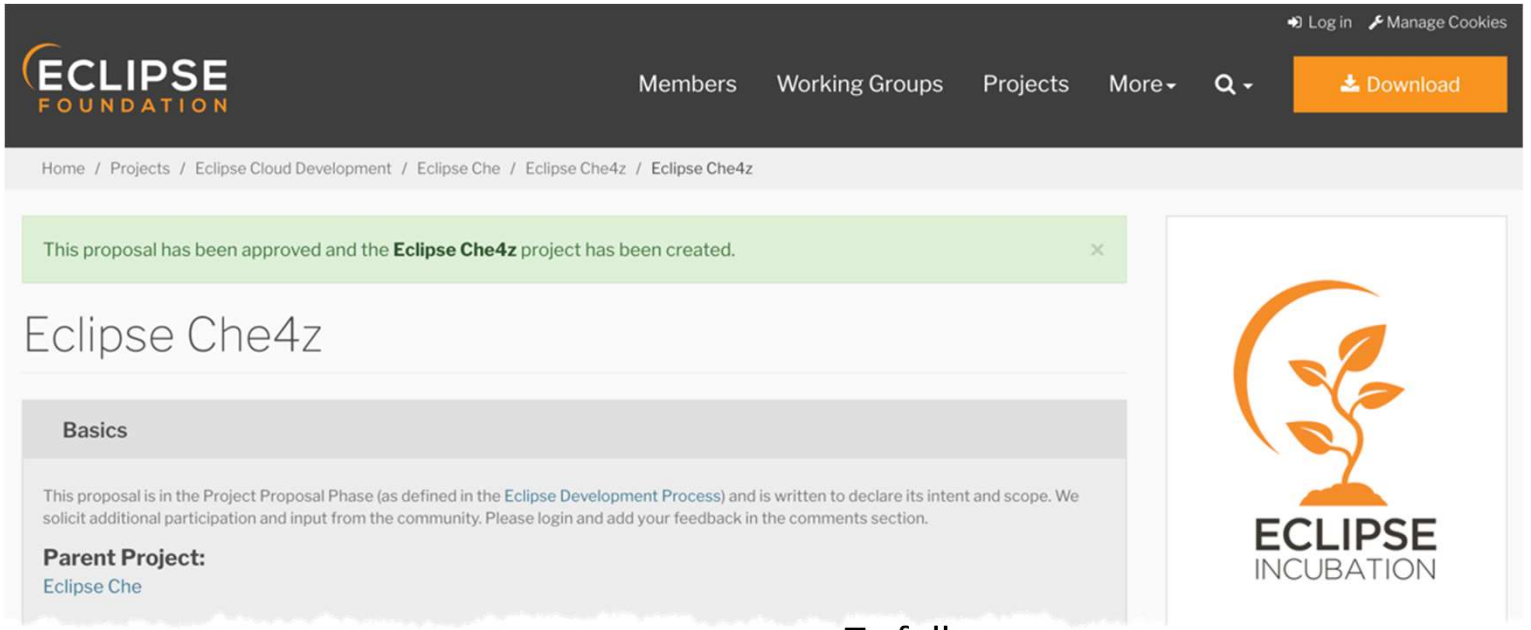
Access to the latest tooling with updates

Tool integration

Quickly on-board new developers

Announcing Eclipse Che4z subproject

Che extensions for IBM z/OS platform



Starting with:

- Access to mainframe resources
- COBOL LSP implementation

To follow:

- LSP implementations for other mainframe languages (PL/I, Assembler)
- Community requests

Eclipse Che 7

- VS Code compatible, collaboration rich

Cloud/Web IDE framework

- Provides flexibility for using any IDE
- Using Theia based Editor
- VS Code user experience/extensions support
- Standard integration with Git
- Support of Language Server Protocol and Debug Adapter Protocol
- Supports “Bring your own device”

Workspace

- Accelerate project and developer on boarding: zero install development environment
- Remove inconsistencies between developer environments
- Built-in security and enterprise readiness

Eclipse Che workflow

Eclipse Che 7

Next-Generation IDE

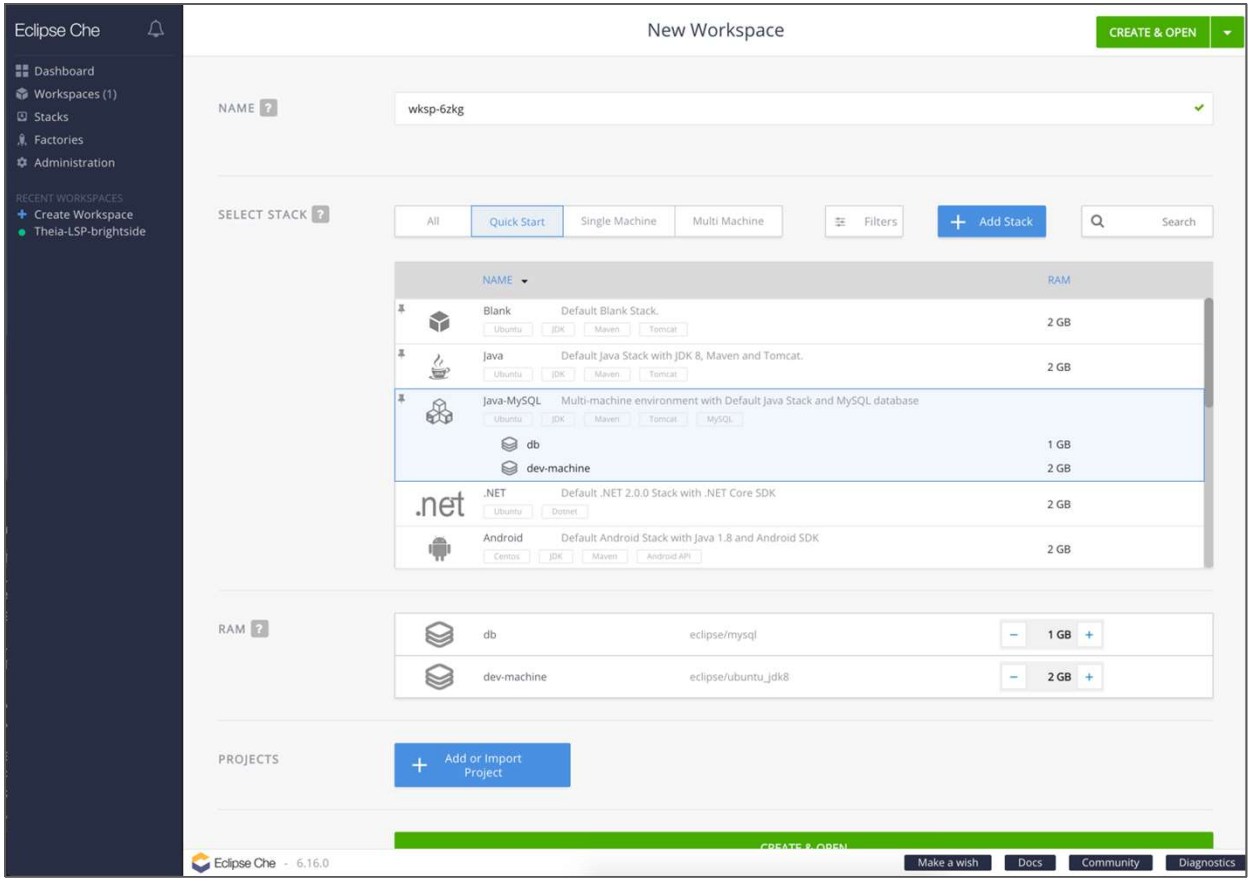
Eclipse Che supports

- Multi-user and multi-tenancy
- Team Workspaces and organization
- Container support: Runs on OpenShift, Kubernetes, or Docker
- Your project and key dependency state persists between runs

Simply create new workspaces

Team-Based Development

- Create custom stacks –
  - runtimes based on your production images, with the tools your developers need
  - Build\on a team stack, or duplicating a workspace



Quickly get started

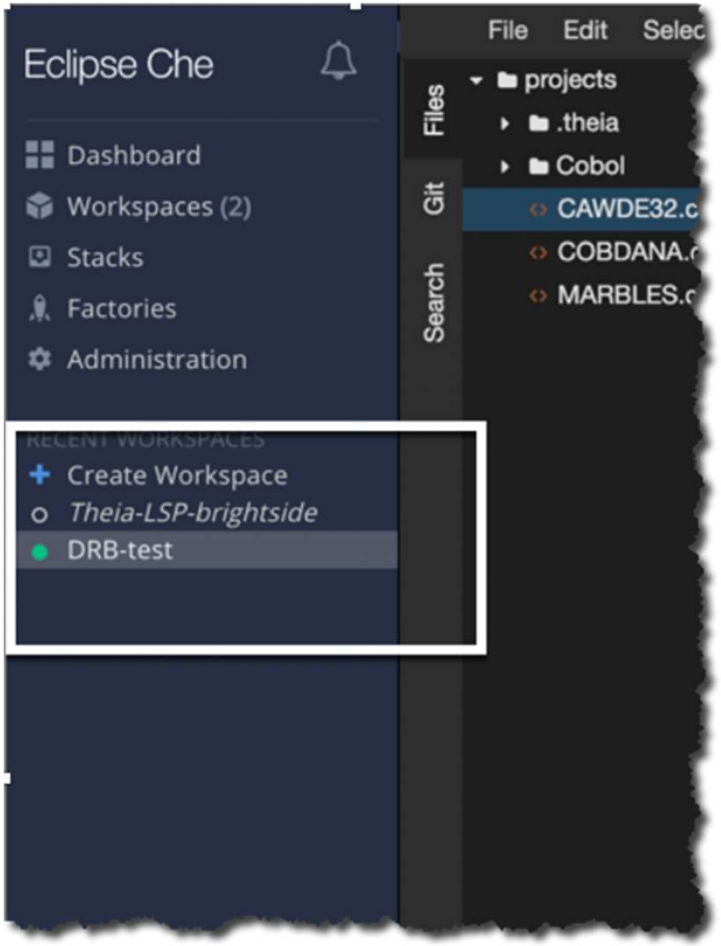
Instant Project Onboarding

Onboard teams with powerful collaboration workspace automation, and permissions. Developers in a team can use their local IDE or the Che browser IDE.

- Share workspaces with anyone
- Control workspace permissions

Integrate developer services into a workspace

- Language Servers
- IntelliSense and Refactoring
- Debuggers
- Command line



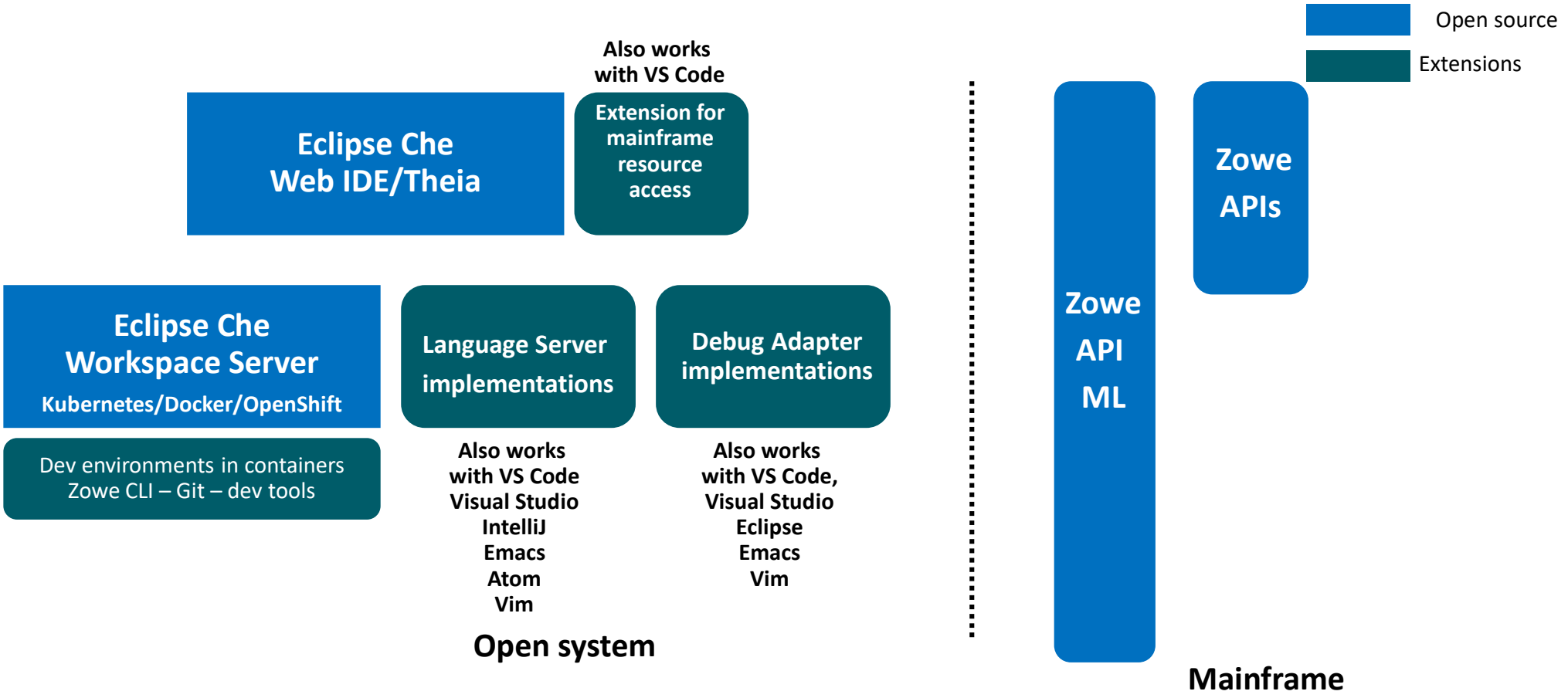
Enterprise management

DevOps Workspace Platform

Manage workspaces at scale with programmable and customizable infrastructure that lets you control system performance, availability, and functionality.

- Use in the cloud or install locally
- Scale horizontally or vertically
- Keep source code off devices
- Enterprise security solutions

Eclipse Che & Zowe



I prefer to use the editor capabilities I like

I like syntax highlighting and syntax check

I like having content assist

I need to work in multiple languages without learning a new tool/editor

I like having code formatting capabilities

*What is Language Server Protocol ?

It was originally developed for Microsoft's Visual Studio Code and is now an open standard

- Adding features like **auto complete, go to definition, or documentation on hover** for a programming language takes significant effort. Traditionally this work had to be repeated for each development tool, as each tool provides different APIs for implementing the same feature.
- A ***Language Server*** is meant to provide the language-specific smarts and communicate with development tools over a protocol that enables inter-process communication.
- The idea behind the *Language Server Protocol (LSP)* is to **standardize the protocol** for how such servers and development tools communicate. This way, a single *Language Server* can be re-used in multiple development tools, which in turn can support multiple languages with minimal effort.
- LSP is a win for both language providers and tooling vendors!

*From Official Language Server Protocol page: https://microsoft.github.io/language-server-protocol/

Language Server Protocol (LSP)

Using Common language protocol enables language plugins like COBOL and JCL to work with Eclipse Che, VS Code and other IDE's

https://langserver.org

The flow illustrates

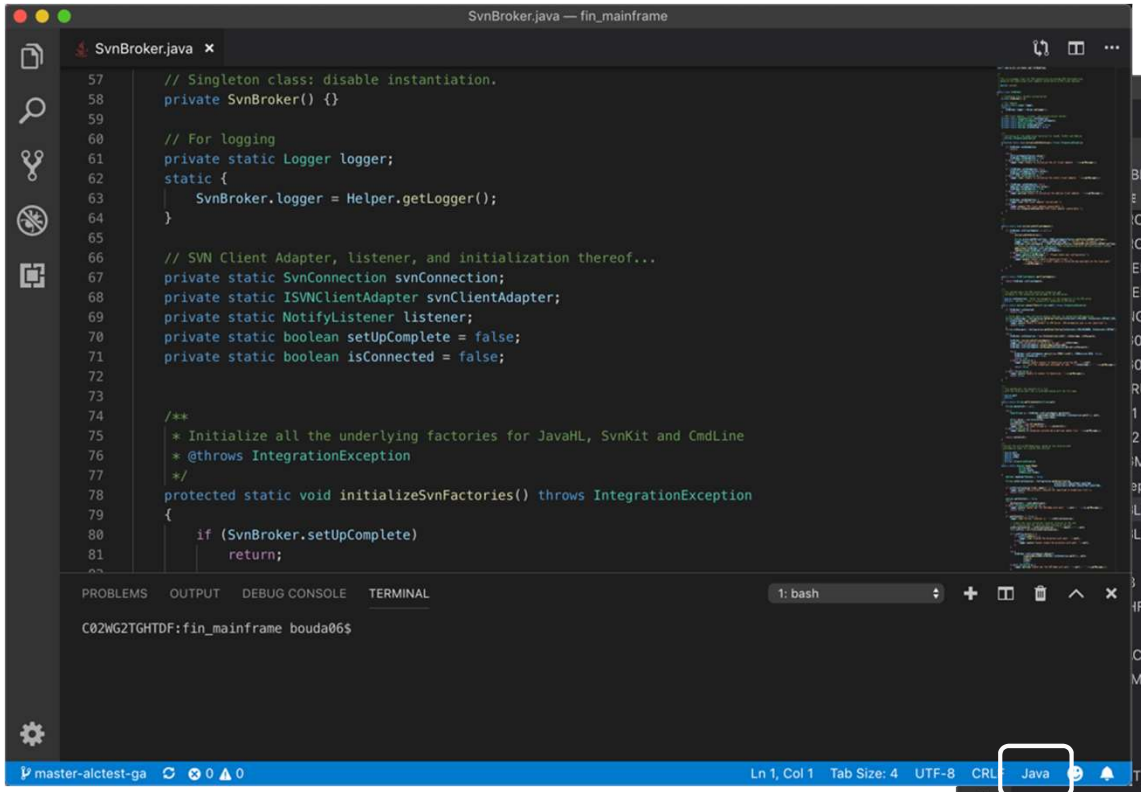
- How the **protocol communicates with the language server** at the level of document references (URIs) and document positions.
- These **data types are programming language neutral** and apply to all programming languages.
- The data types are not at the level of a programming language domain model which would usually provide abstract syntax trees and compiler symbols (for example, resolved types, namespaces, …).
- The fact, that the **data types are simple and programming language neutral** simplifies the protocol significantly.
- It is much **simpler to standardize a text document URI or a cursor position** compared with standardizing an abstract syntax tree and compiler symbols across different programming languages.

Capabilities

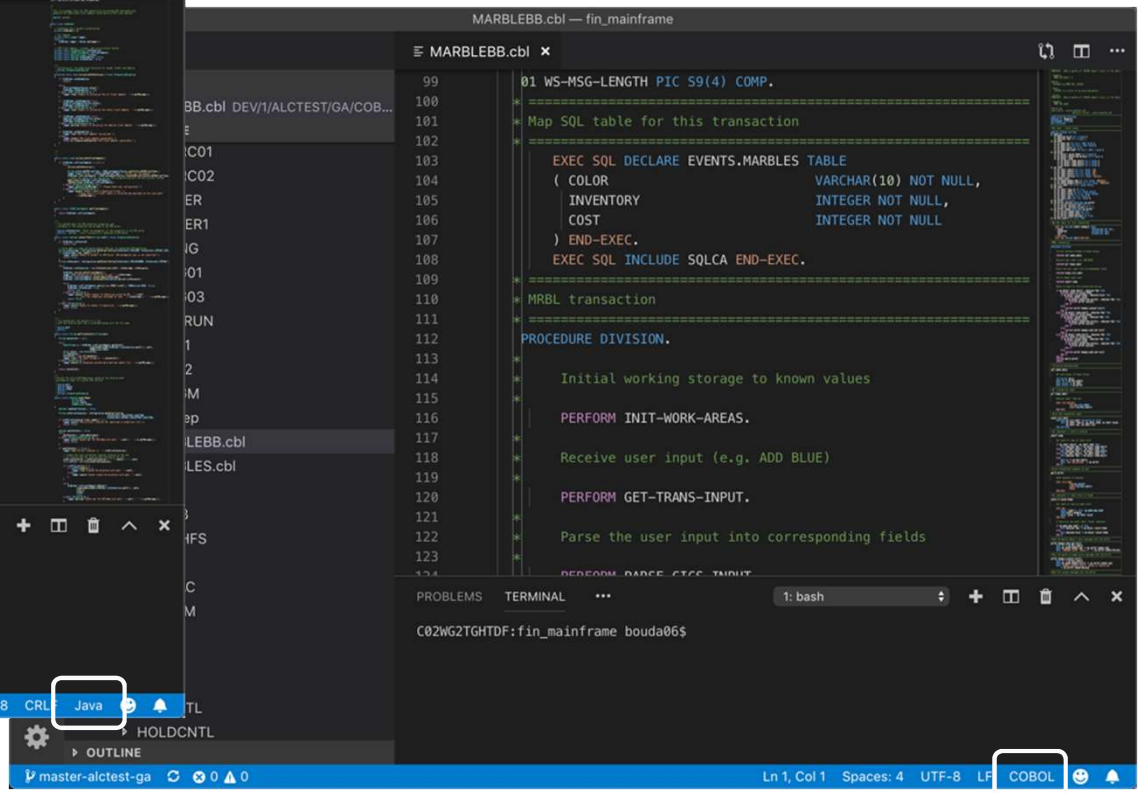
- **Not every language server can support all features** defined by the protocol. LSP therefore provides ‘capabilities’.
- A **capability groups a set of language features**. A development tool and the language server announce their supported features using capabilities.
- As an example, a server announces that it can handle the ‘textDocument/definition’ request, but it might not handle the ‘workspace/symbol’ request. Similarly, a development tool announces its ability to provide ‘about to save’ notifications before a document is saved, so that a server can compute textual edits to format the edited document before it is saved.

What that means for VS Code?

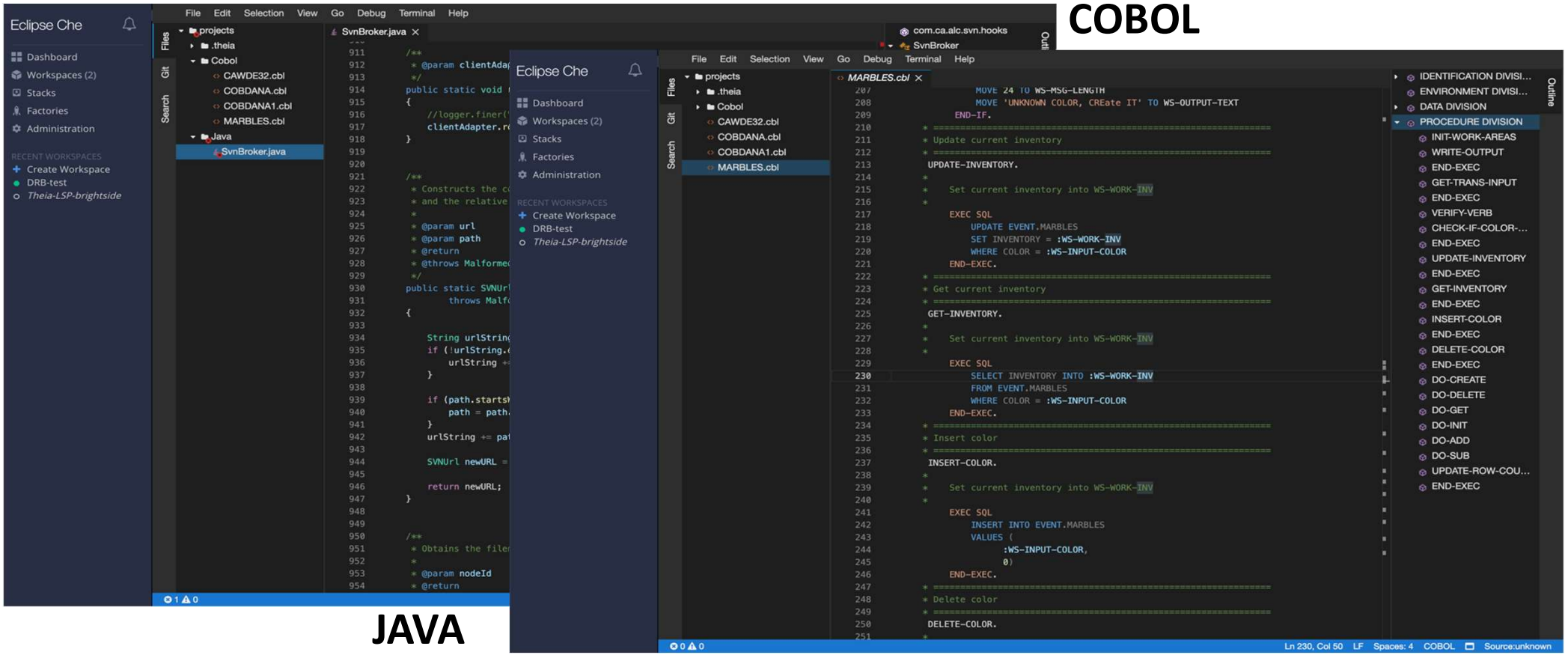
JAVA



COBOL



What that means for Eclipse Che ?



GUIDE SHARE EUROPE UK REGION

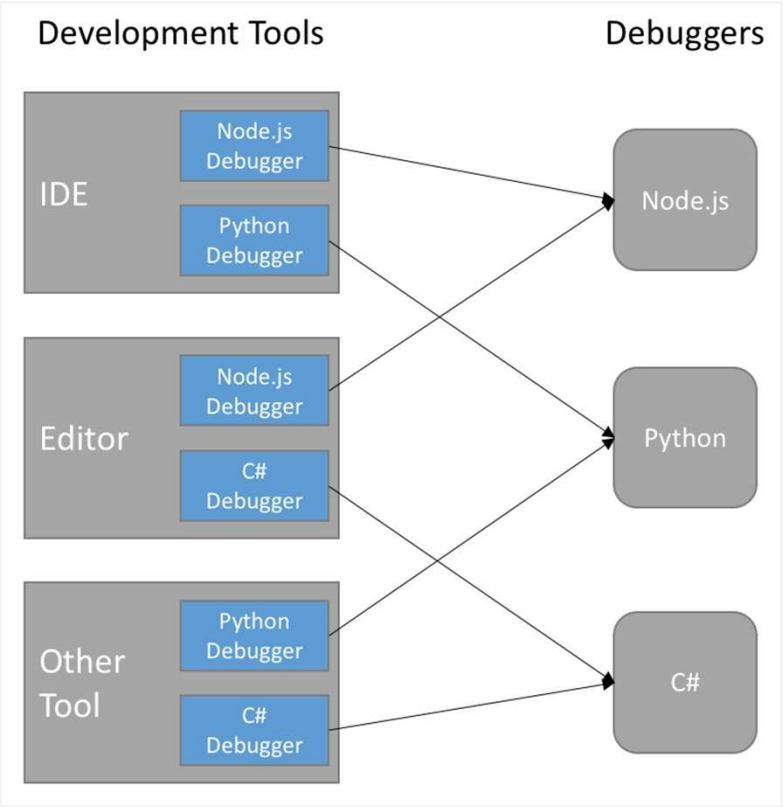
I prefer to use the debuggers I like

I want consistent debugging experience

I want to leverage my remote debugger

I need to work in multiple languages

Current state of Debuggers…



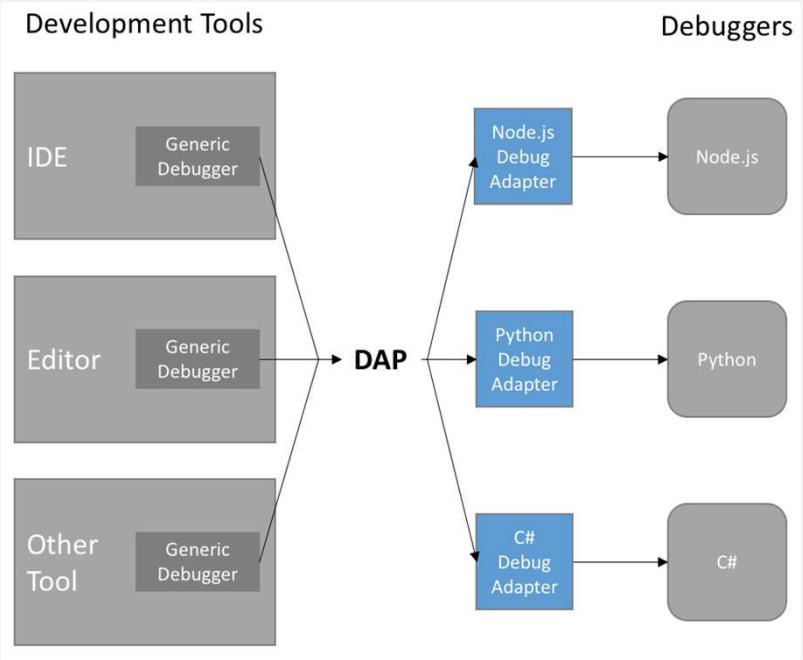
It takes a significant effort to implement the UI for a debugger for features like

- source-, function-, conditional-, and inline breakpoints,
- variable values shown in hovers or inlined in the source,
- multi-process and multi-thread support,
- Multiple runtimes,
- navigating through complex data structures,
- watch expressions

Typically this work must be **repeated for each development tool**

- Each tool uses different UI APIs
- Creating duplicated functionality (and implementation)

What is the Debug Adapter Protocol?



Similar to Language Server Protocol, the idea behind the ***Debug Adapter Protocol*** **(DAP)** is to abstract the way how the debugging support of development tools communicates with debuggers or runtimes into a protocol.

Since it is unrealistic to assume that existing debuggers or runtimes adopt this protocol any time soon, we rather assume that an intermediary component - a so called *Debug Adapter* - adapts an existing debugger or runtime API to the Debug Adapter Protocol providing “language smartness.”

This intermediary becomes the *Debug Adapter* which explains the name of the protocol: *Debug Adapter Protocol*.

Debug Adapter

Start Debug adapter

Initialize request

Response: capabilities

Start Debug monitoring

Set Breakpoints request

Break: Set Breakpoints

Response: breakpoints

Debug

Continue

Step Over

Step Into

Step Out

Continue

Response: status

Stopped

Stopped event

Variables request

Find Variable

Get Variable details

Response: variables

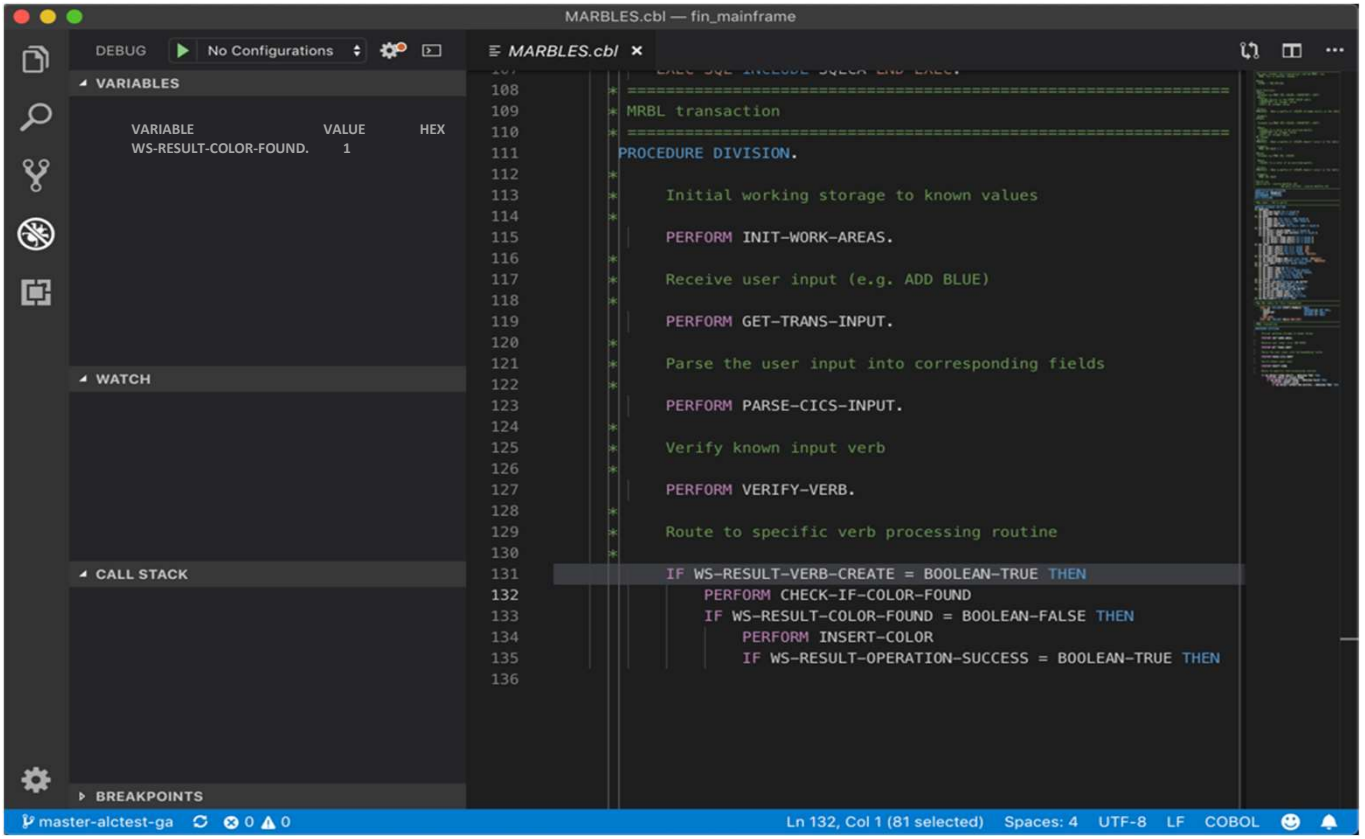
Exit

Exit

The flow illustrates

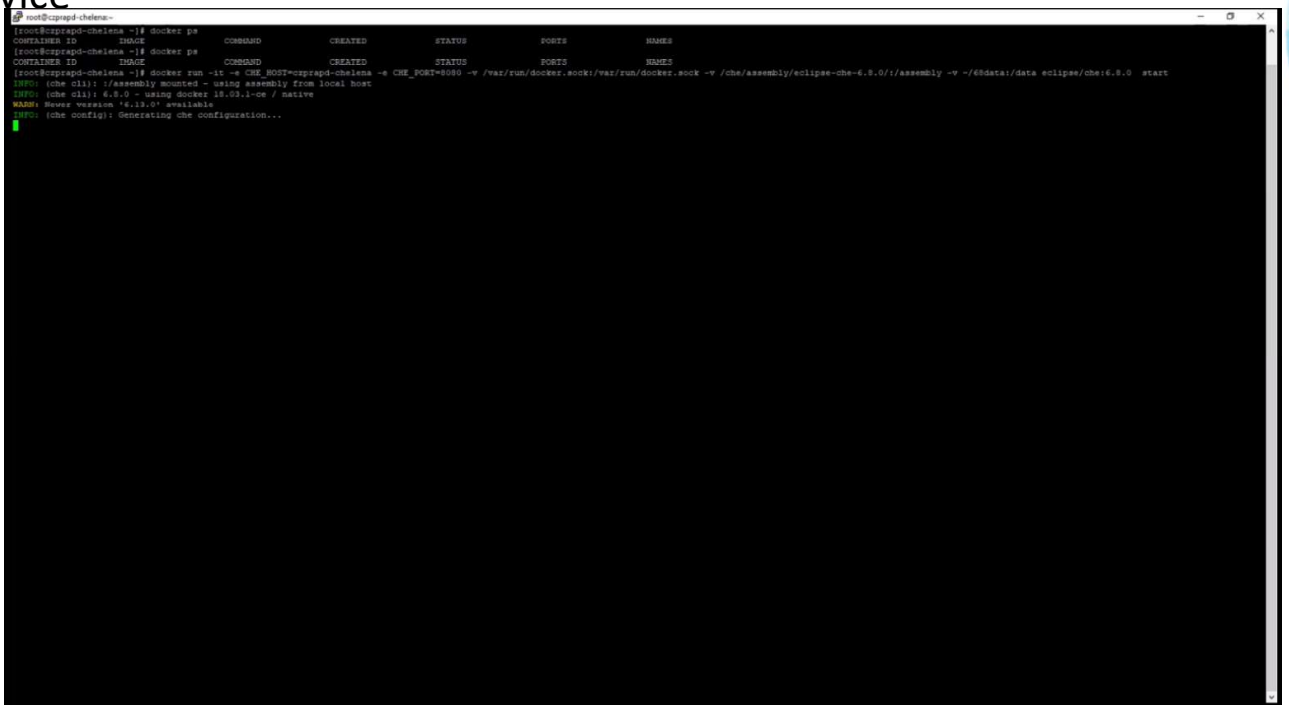
- The Debug Adapter Protocol makes it possible to implement a single generic debugger UI per development tool and that **Debug Adapters can be re-used** across these tools. This reduces the effort to support a new debugger considerably.
- **Standardizing on a wire-protocol instead of an API and a client library** has the advantage that a debug adapter can be implemented in the language most suitable for the given debugger or runtime.
- Since the *Debug Adapter Protocol* was designed for supporting the debugging UI in a **language agnostic way**, it is fairly high-level and does not have to surface all the fine details of the underlying language and low-level debugger API. The most important data type used in the protocol are strings, because that's what the end user will see in the UI. So Debug Adapters typically aggregate information received via debugger APIs into high-level, string-based data-structures that are directly consumed in the UI of the development tool. Since this mapping is mostly straightforward and has little complexity, Debug adapters can be developed with minimal effort.

Debugging COBOL in VS Code



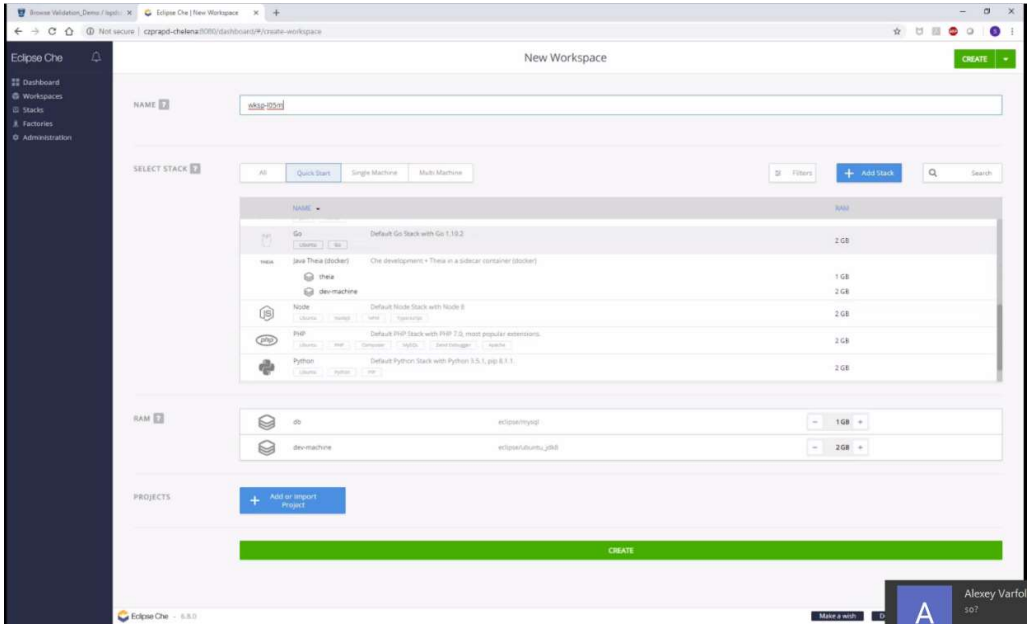
Deploy the solution

- Todd prepare the server
- Deploying the Che
- Deploy LSP service



Developer getting started

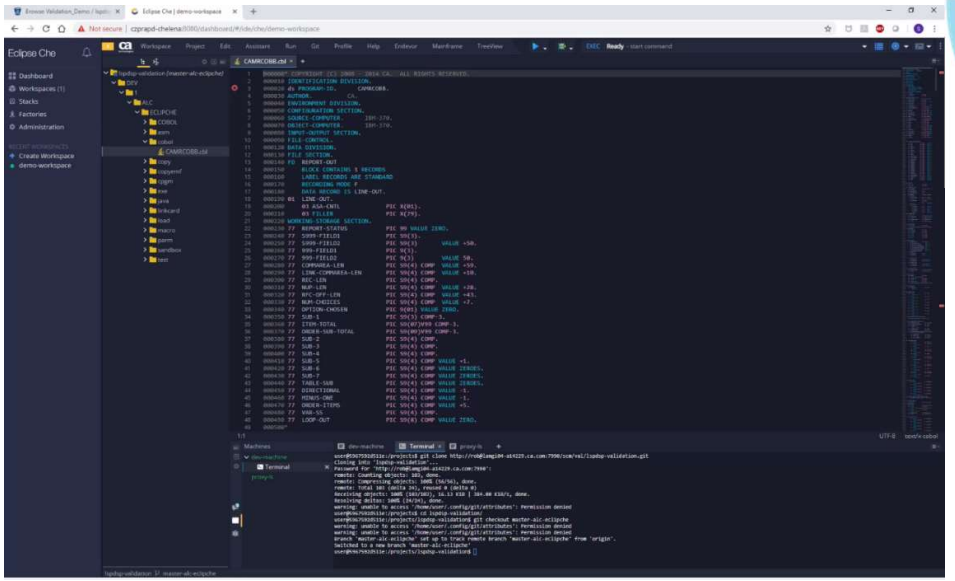
- Use URL from admin
- Choose proper stack
- Create workspace
- Configure workspace



Michelle
Application developer

Developer – working with code

- Open workspace
- Access the code
  - Using Git to Endevor
  - Using File Explorer for z/OS
- Make changes in code
  - Using COBOL smart editors leveraging LSP support
  - Using HLASM smart editor edit the code
- Debug the code



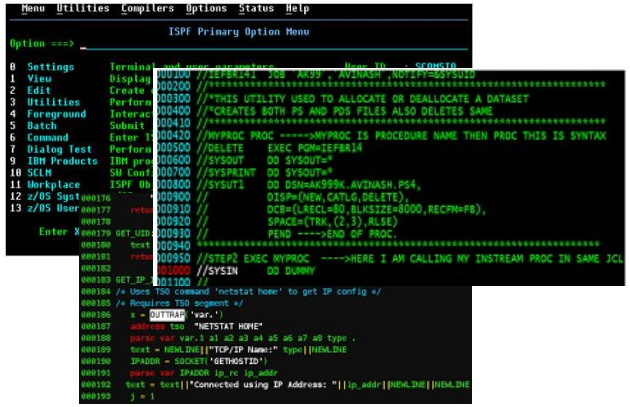
Michelle
Application developer

Modern Developer Experience

Eclipse Che

Past

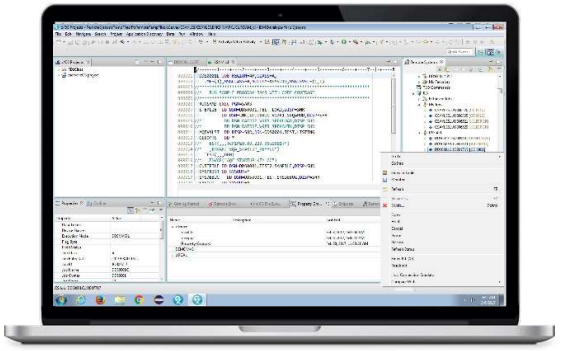
Mainframe only



- TSO/ISPF interfaces
- JCL/Rexx for build and system testing
- Waterfall aligned
- Platform-dependent tools

Present

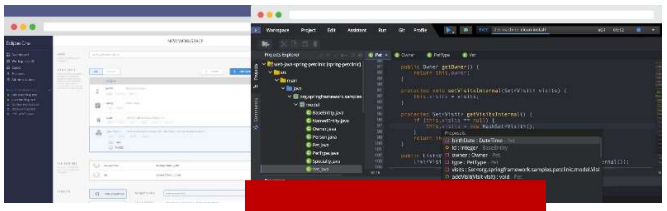
Similar to non-mainframe



- Workstation-based IDE (Eclipse, VS Code,…)
- Vendor plug-ins
- Agile aligned
- Proprietary tools

Coming Soon

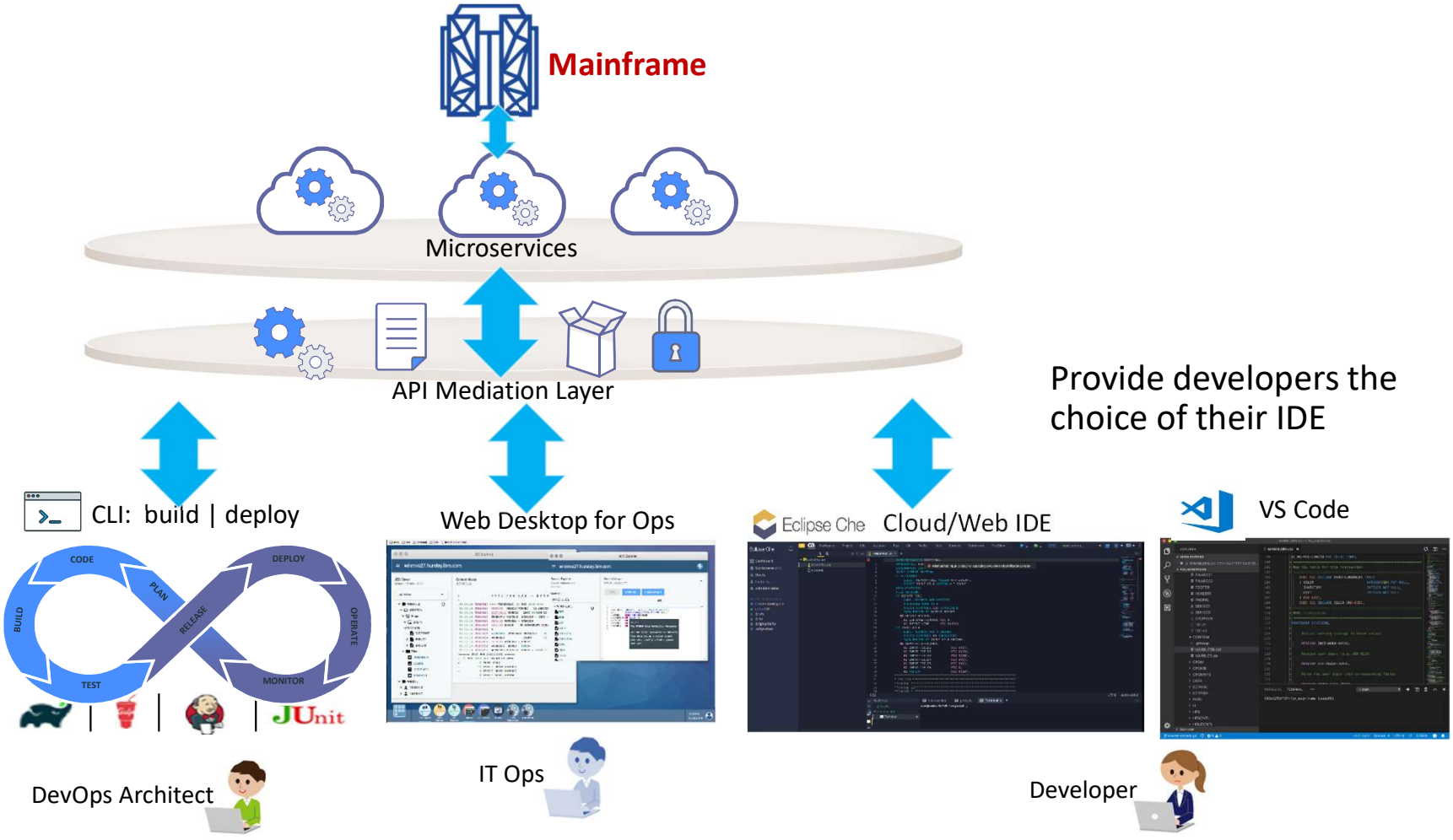
Same as non-mainframe



Maintenance costs eliminated

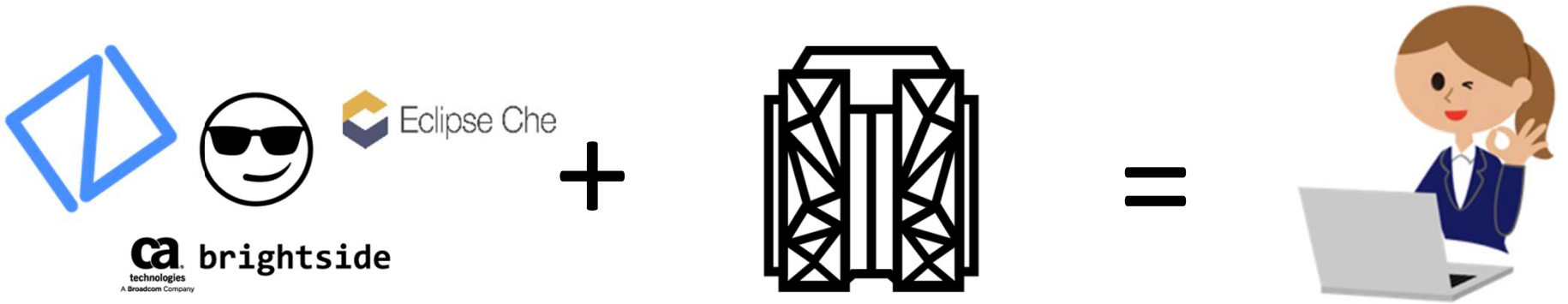
- Cloud-based Eclipse Che IDE
- Workspaces, CLI's, REST APIs
- DevOps aligned
- Open source tools

In conclusion…



Modern Developer Summary

1. Developers get to use the tools and languages of their choice
2. Remove technical and cultural barriers to automating mainframe testing
3. Remove technical and culture barriers to mainframe CI/CD
4. Enable cross-platform automated delivery



Want to learn more about Zowe?

What is Zowe?

Zowe is an open source project created to host technologies that benefit the Z platform from all members of the Z community (Integrated Software Vendors, System Integrators and z/OS consumers). Zowe, like Mac OS or Windows, comes with a set of APIs and OS capabilities that applications build on and also includes some applications out of the box.

Zowe offers modern interfaces to interact with z/OS and allows you to work with z/OS in a way that is similar to what you experience on cloud platforms today. You can use these interfaces as delivered or through plug-ins and extensions that are created by clients or third-party vendors.

Zowe consists of the following main components.

Zowe Application Framework: A web user interface (UI) that provides a virtual desktop containing a number of apps allowing access to z/OS function. Base Zowe includes apps for traditional access such as a 3270 terminal and a VT Terminal, as well as an editor and explorers for working with JES, MVS Data Sets and Unix System Services.

API Mediation Layer: Provides a gateway that acts as a reverse proxy for z/OS services, together with a catalog of REST APIs and a dynamic discovery capability. Base Zowe provides core services for working with MVS Data Sets, JES, as well as working with z/OSMF REST APIs. The API Mediation Layer also provides a framework for Single Sign On (SSO).

Zowe CLI: Provides a command-line interface that lets you interact with the mainframe remotely and use common tools such as Integrated Development Environments (IDEs), shell commands, bash scripts, and build tools for mainframe development. It provides a set of utilities and services for application developers that want to become efficient in supporting and building z/OS applications quickly. The CLI provides a core set of commands for working with data sets, USS, JES, as well as issuing TSO and console commands.

Download

The easiest way to get started with Zowe is by downloading the convenience build. You can also go to the GitHub repository to build Zowe on your own.

Zowe 1.5.0 z/OS Components | **Zowe 1.5.0 Command Line Interface** | **Zowe GitHub repositories** | **Zowe SMP/E Alpha**

▸ Past Releases

** Please note the Zowe binaries are made available to you by Zowe Binary Projects a Series of LF Projects, LLC, and not by The Linux Foundation or the Open Mainframe Project.*

Navigating your DevOps journey

How to get started

- Prioritize outcomes and take actions that move toward those goals
- Alignment w/ Enterprise DevOps is critical
- Start making progress on automated testing
- Modernizing the developer experience with Git / Endevor bridge and Zowe drives multiple outcomes
- Empower your change agents to automate
  - w/ Zowe and open source tools
  - Bottom-up adoption
  - Change begins w/ culture

Navigating *your* DevOps journey

GUIDE SHARE EUROPE UK REGION

Broadcom DevOps Center of Excellence

Broadcom offerings (no fee)

Use Case Exploration

Format: 1/2 day facilitated session using Design Thinking practices

Deliverable: Prioritized set of DevOps-related use cases by persona presented in a read-out session

EXPLORE

DevOps Assessment

Format: 1-2 day agenda for interviews with Leadership, Development, & Operations teams

Deliverable: Custom DevOps Assessment with maturity ratings (1-5), detailed findings, recommendations & next steps (workshops, health checks, integrations, tooling)

ASSESS

CA Brightside Workshop

Format: 2-day, on-site workshop, facilitated by a DevOps expert

Deliverable: Participants will learn how to automate mainframe AppDev using Zowe with modern tools like Jenkins, Gulp, Jest, and Visual Studio Code

LEARN

DevOps Proof of Concept

Format: Access to a DevOps/Zowe expert for mutually-defined PoC

Deliverable: Successful application of Zowe and complementary tooling in your environment designed to demonstrate feasibility and ROI

PROVE

Q&A

Broadcom DevOps Center of Excellence

Contact:
Gerald.Pfeiffer@Broadcom.com
linkedin.com/in/gerald-pfeiffer-6911176/
twitter @GeraldMainframe

Please submit your session feedback!

- Do it online at http://conferences.gse.org.uk/2019/feedback/ao
- This session is AO