what's new with DevOps on z

Agnes ten Brink

IBM Benelux - Technical Sales DevOps on Z
agnes.ten.brink@nl.ibm.com

Agenda

- What is DevOps
- Analyze
- Provision environment (Cloud)
- Check out code (SCM)
  - Develop/change
  - Build + Unit test + debug
  - Code review
- Deploy to next level in CICD Pipeline

So what is DevOps ?

- DevOps is a set of **practices**
- intended to **reduce the time** to move change to production
- while ensuring **high quality**

DevOps is about **Culture**...

It´s about

- **collaboration** across roles
- focus on **business**

  not departments
- **learning** by experimenting

It´s all about **people**

Why DevOps ?

- Shorten **time to value** – *“from concept to cash”*

- Increased **capacity to innovate**

- Enhanced **customer experience**

Why DevOps ?

Rise of Devops influenced by:

- Spread of **Agile/LEAN** development
- Introduction of (private) **Cloud**
- Move to a **Microservices** architecture

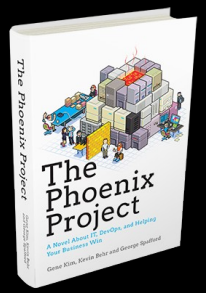
DevOps is about
“the three ways”

1st way **Systems Thinking**

2nd way amplify **Feedback Loops**

3rd way **Continuous Experimentation & Learning**

… this is explained in book “The Phoenix Project”



DevOps is about
“the three ways”

1st way **Systems Thinking**

- **Business** Value stream
  - **Not** the department or silo of work
- Begins with **Business requirements**

2th Way: **Feedback loops** (and Shift Left testing)

"80% of development costs are spent identifying and correcting defects" **

During the Coding or Unit Testing phases

$80/defect

During the BUILD phase

$240/defect

During Quality Assurance or the System Test phases

$960/defect

Once released into production

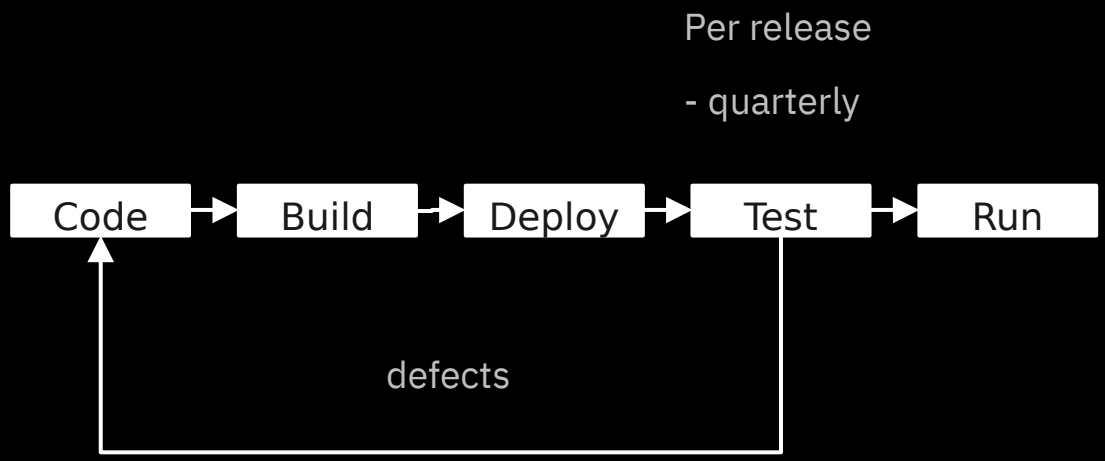
$7,600/defect
+
Law suits, loss of customer trust, damage to brand

**National Institute of Standards & Technology

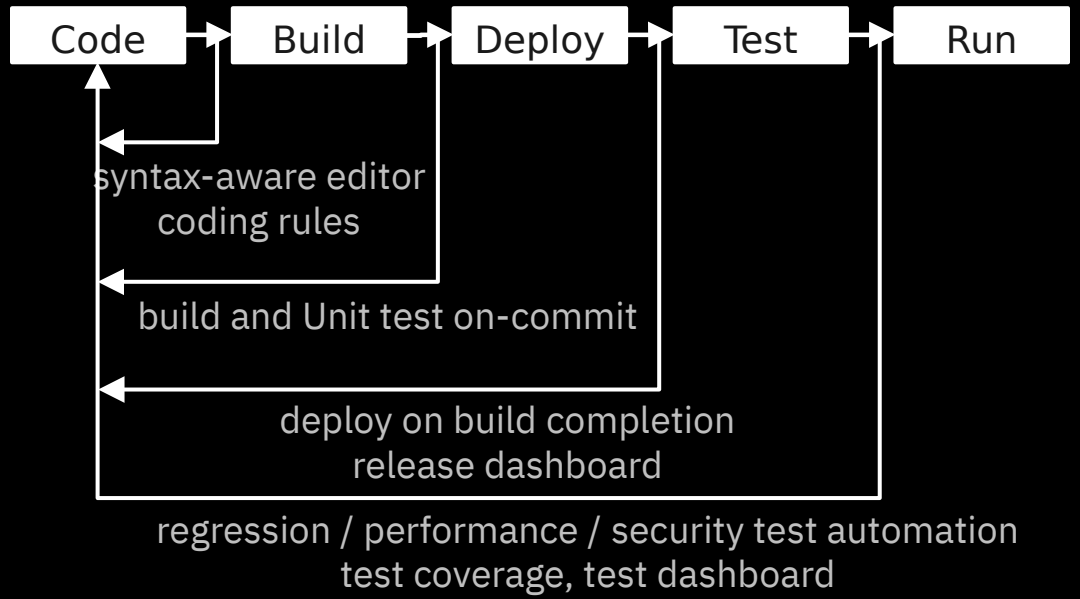
Source: GBS Industry standard study

Defect cost derived in assuming it takes 8 hours to find, fix and repair a defect when found in code and unit test.
Defect FFR cost for other phases calculated by using the multiplier on a blended rate of $80/hr.

Waterfall feedback loop

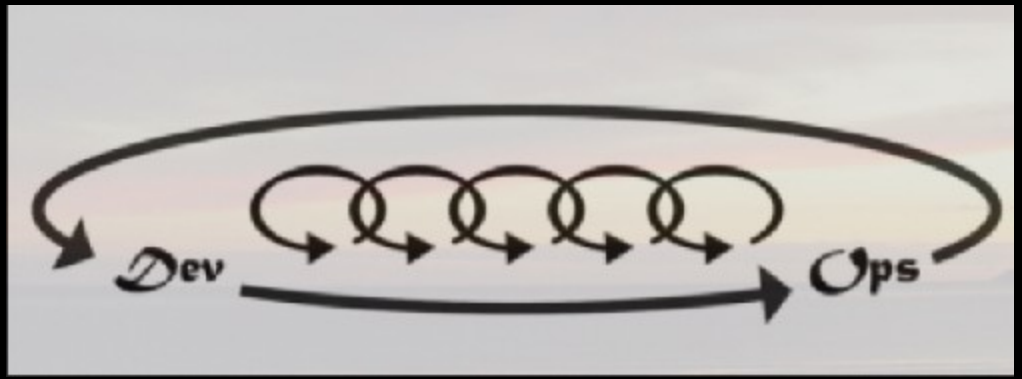


DevOps feedback loop

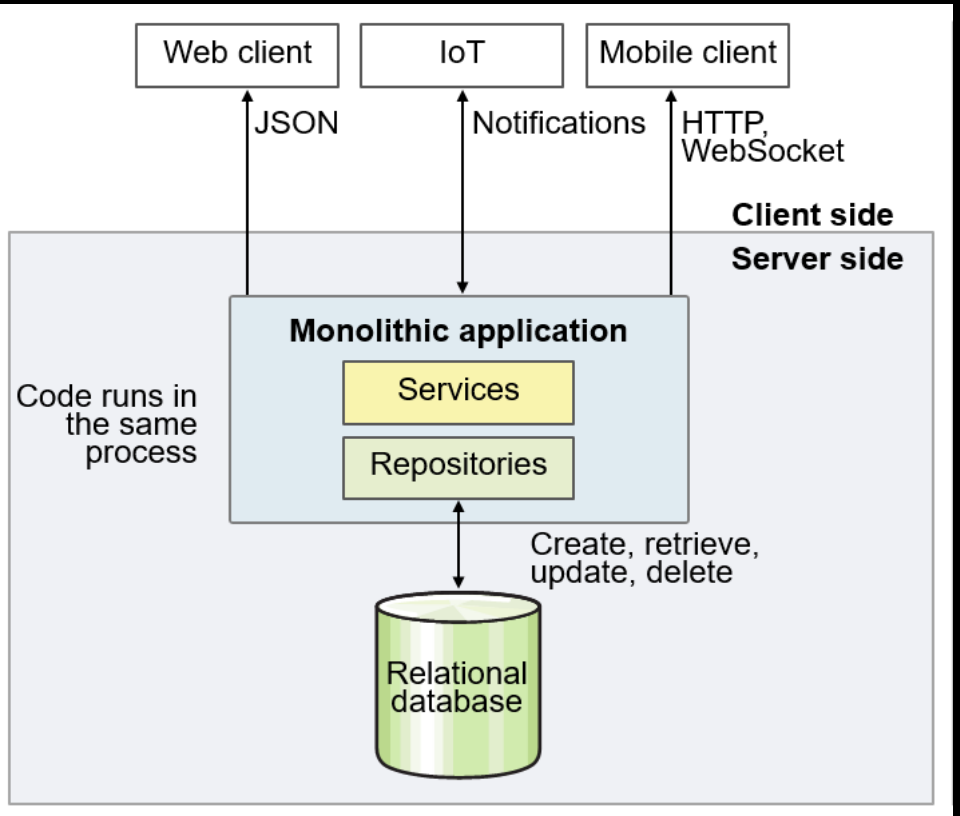


DevOps is about
“the three ways”

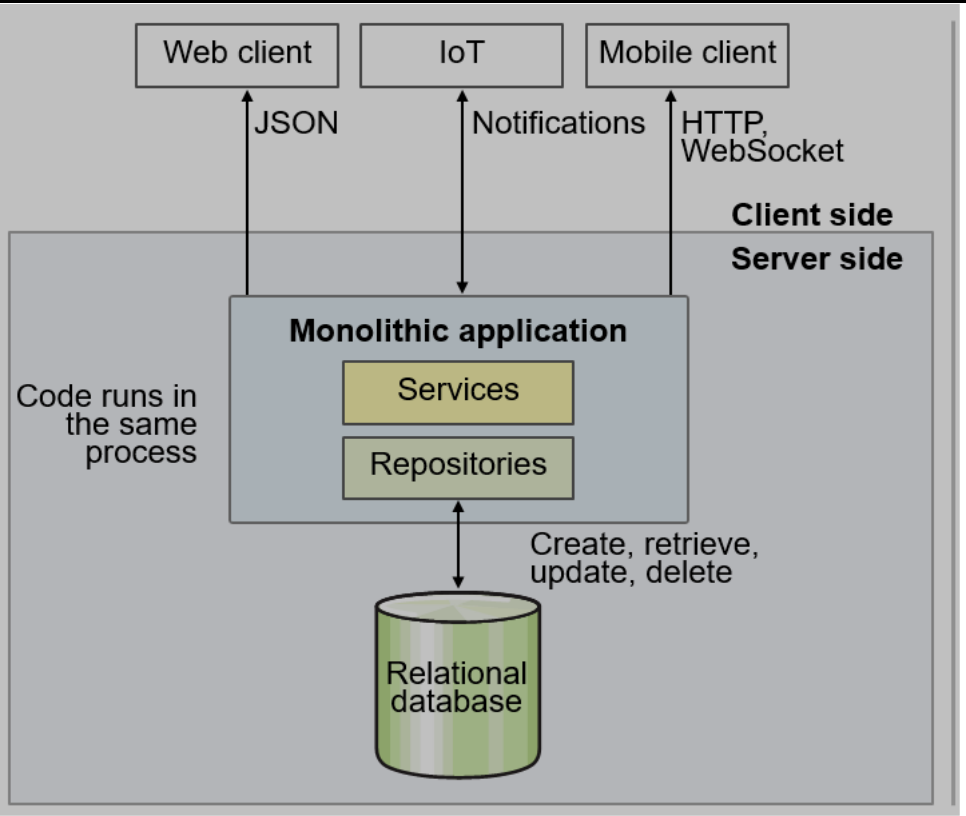
3rd way **Continuous Experimentation & Learning**

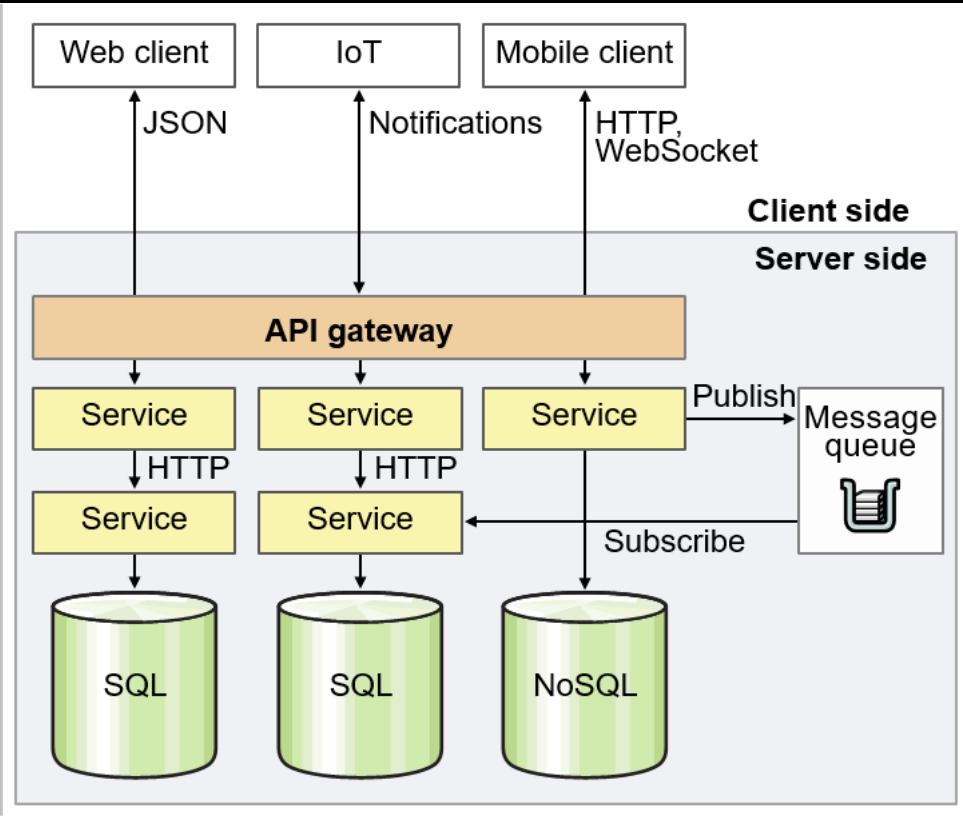


MicroServices break up applications in manageable parts to improve business agility, and leading to many smaller parts



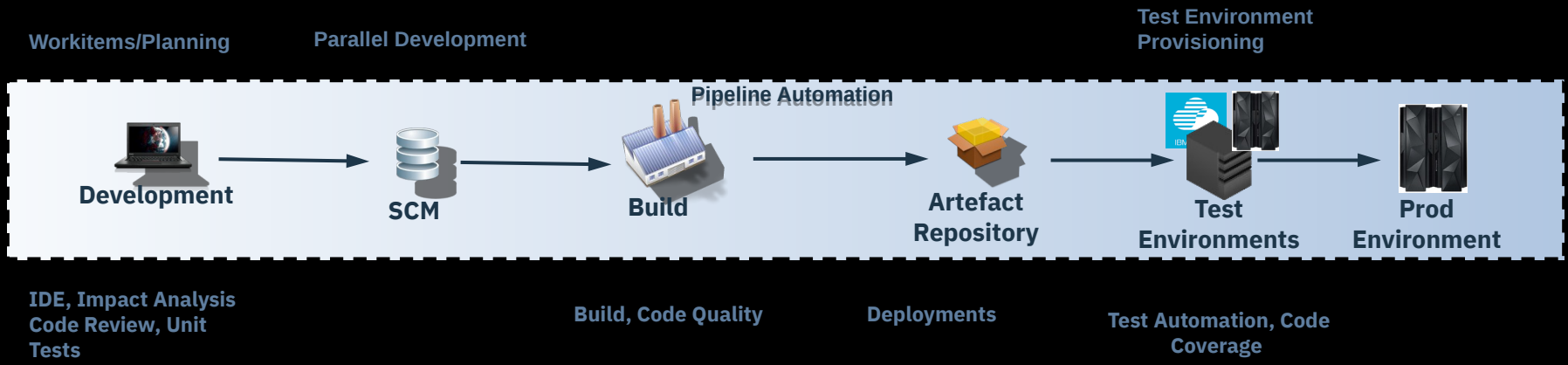
MicroServices break up applications in manageable parts to improve business agility, and leading to many smaller parts





CI/CD Pipeline

DevOps Engineer: what could his day look like in 2020

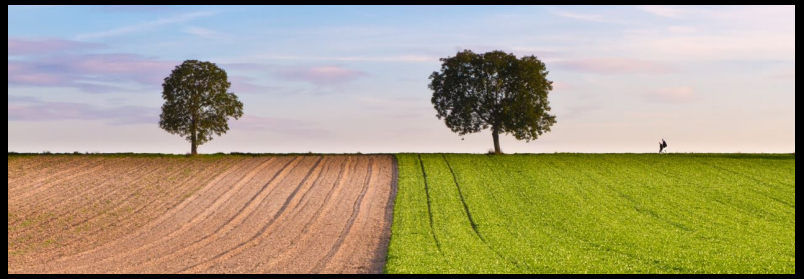


To design a toolchain

- Clarify **requirements**
- Make **design decisions**

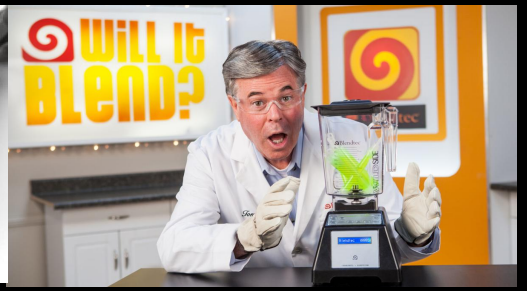
#architecture

Build for **brownfield** or **greenfield** ?

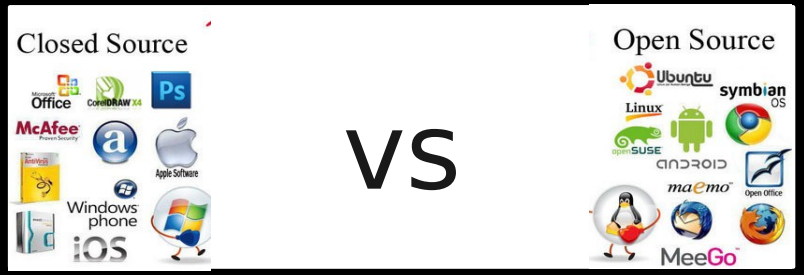


What **existing tools** do you have ?

Will the toolchain **integrate** ?



Do we prefer **open source** or **Vendor tools** ?

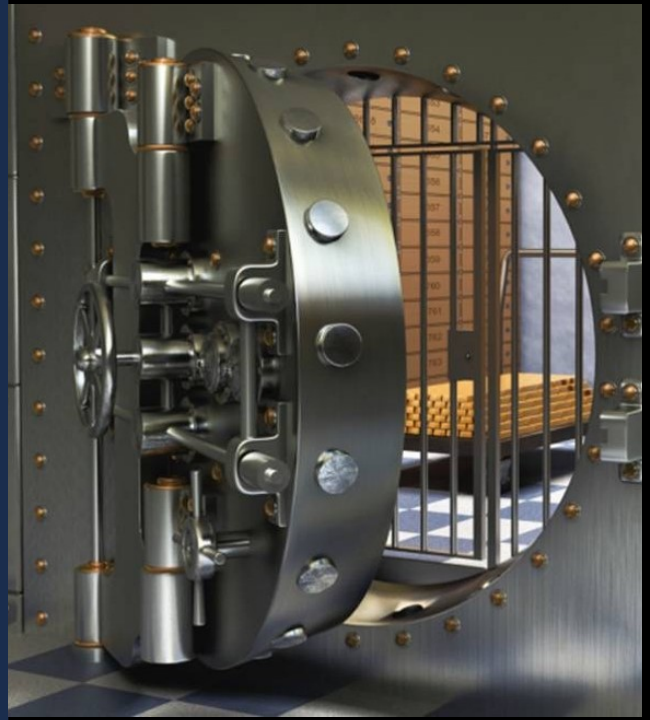


What do **other companies** do ?

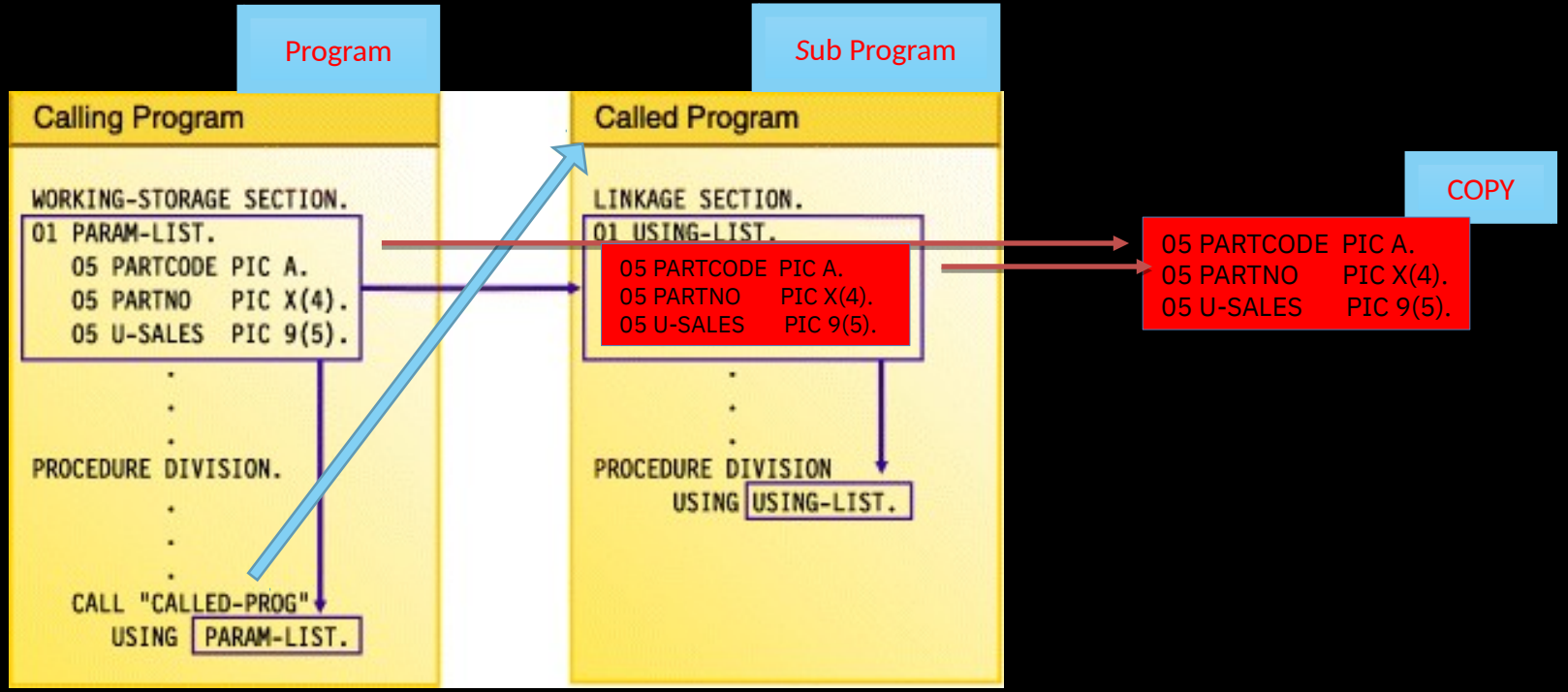
Do we have the right **skills** ?

Analyze

Task: make a change to this 20 year old program but don’t break everything else

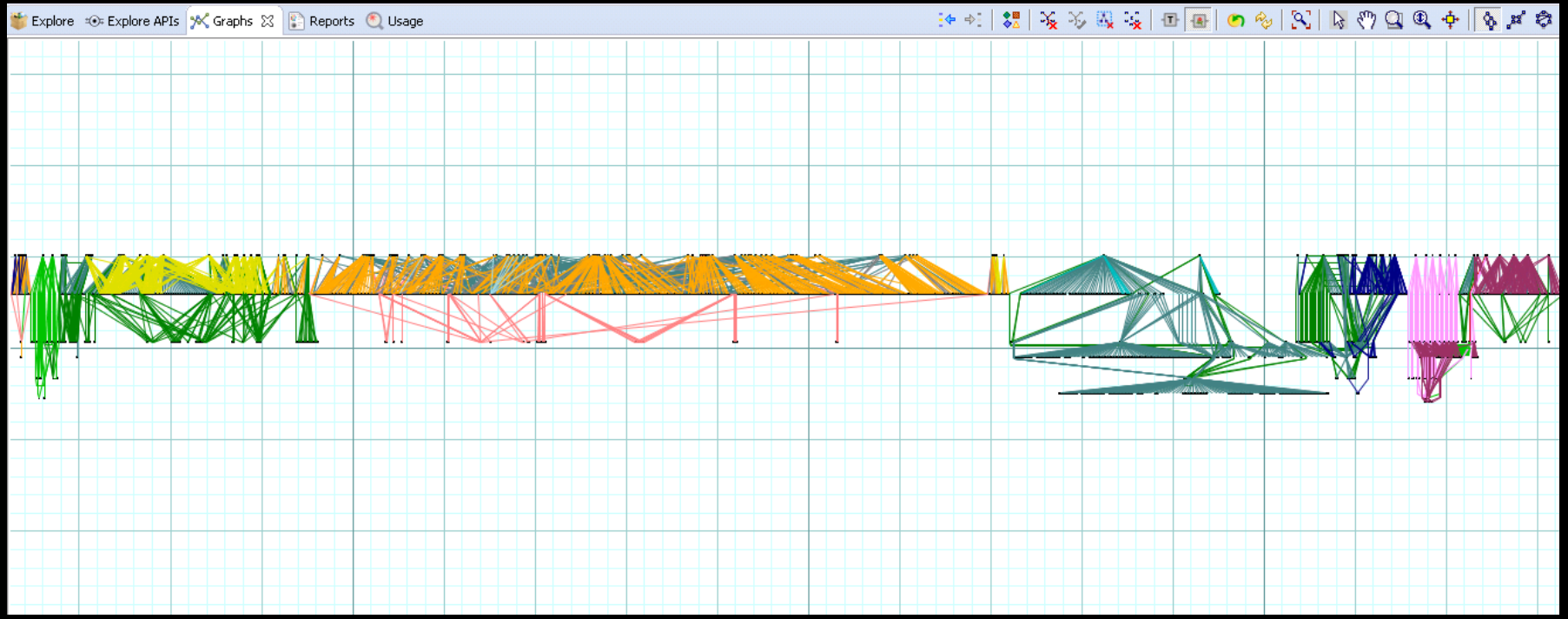


The challenge of managing dependencies between applications



Breaking down the application:

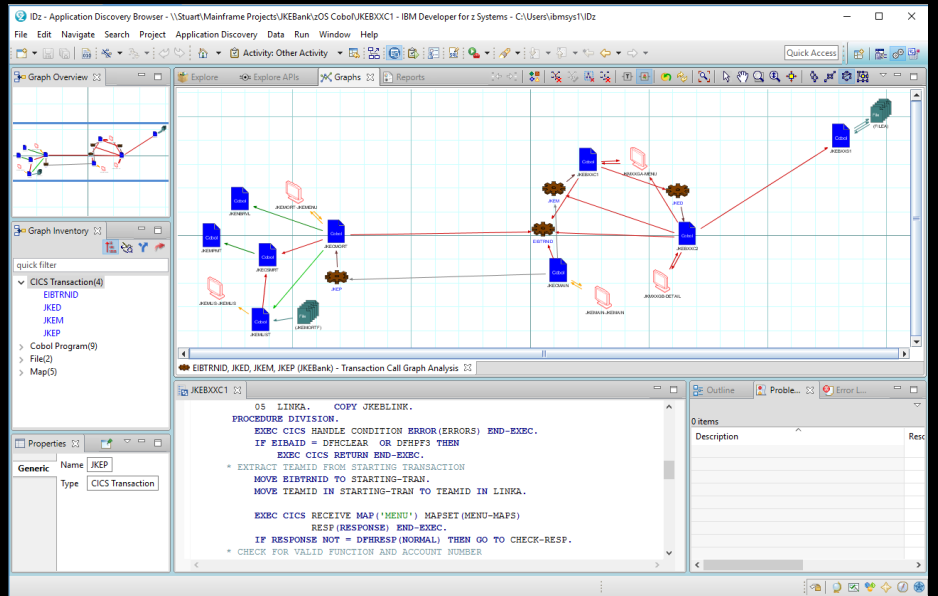
Complex source code presented in call graph layout



Analyze

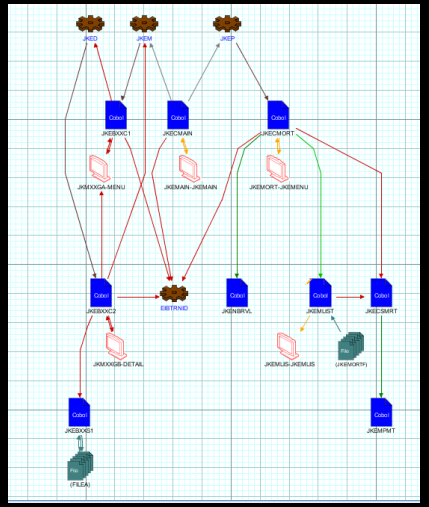
Using Application Discovery

- Eclipse based
- Integrates with Idz, zD&T

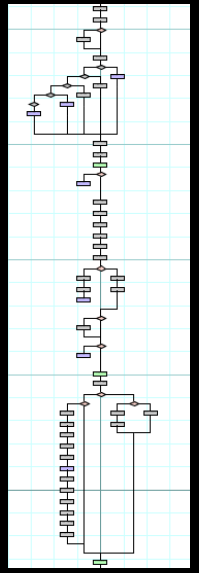


Analyze

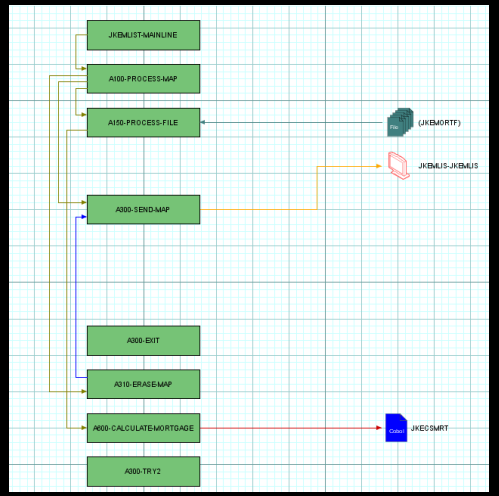
Using Application Discovery



Transaction Call Graph



Program flowchart



Program flow

IBM ADDI Supported Environments

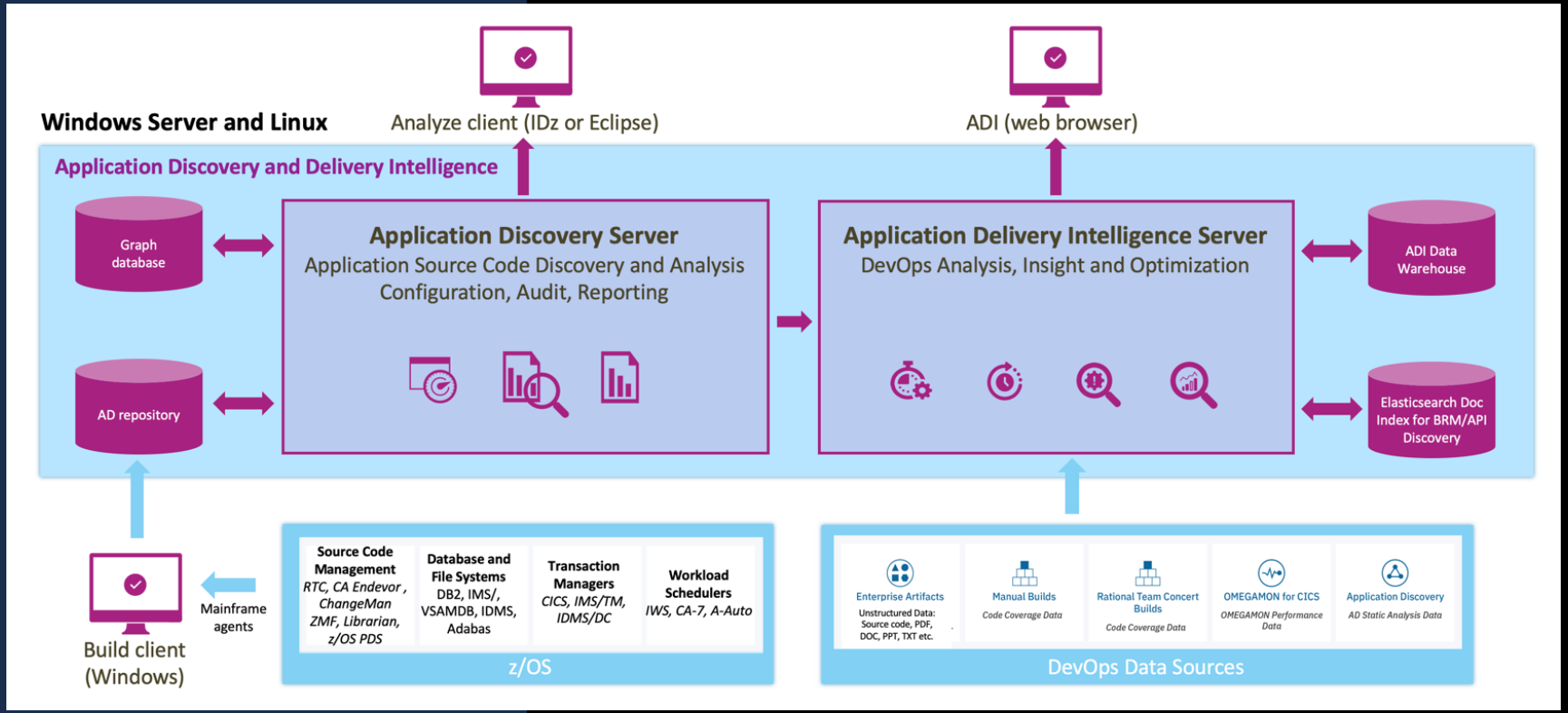
Mainframe – z/OS

- Languages – COBOL, PL/I, Assembler, Natural, CA ADS/Online
- Databases - VSAM, DB2, IMS/DB, Datacom, Adabas, IDMS
- Batch – JCLs, Procs, Ctrl
- TP monitors – CICS, IMS/TM, IDMS/DC
- Schedulers – IWS, A-Auto, CA7
- Messaging – MQ
- SMF analysis – Jobs, CICS Transactions
- SCM – RTC EE, Broadcom Endevor, Librarian, Serena ChangeMan ZMF, z/OS PDS, ITC
- AD tools – ADFz, IDz
- Monitoring – OMEGAMON for CICS
- API connectivity – z/OS Connect

Distributed

- Languages – Java

IBM ADDI Architecture



Provision: zD&T

- Run z/OS local on any Intel server
  - For Development and Test
  - Everyone their own environment
  - Adaptable to the developer’s needs

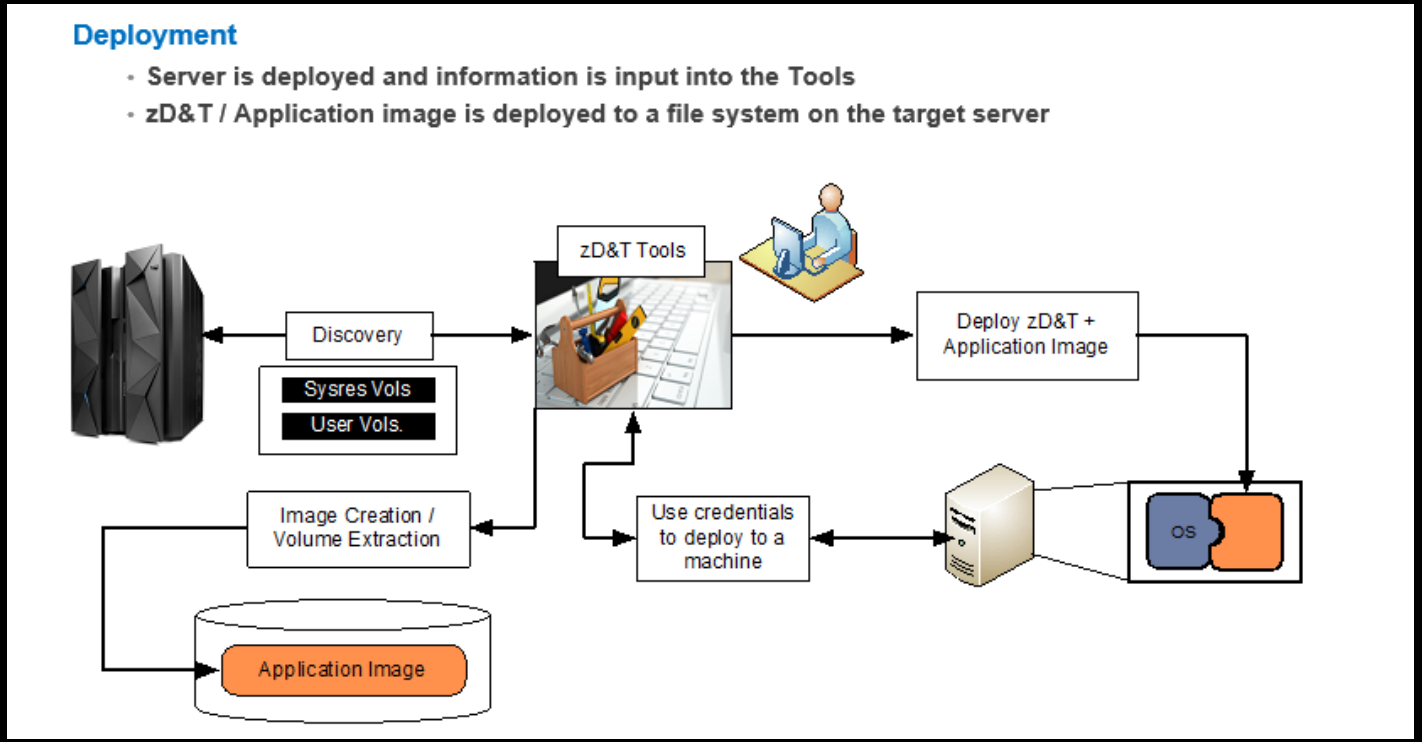
Provision: zD&T

Two main options:

- Use the provided ADCD Volumes
  - Needs tweaking to get a production-like environment
  - Need to add user data and sources
- Use Volume cloning to create Volumes
  - Needs a real LPAR to clone from

In both cases: Create a “golden copy” to spin off zD&T instances

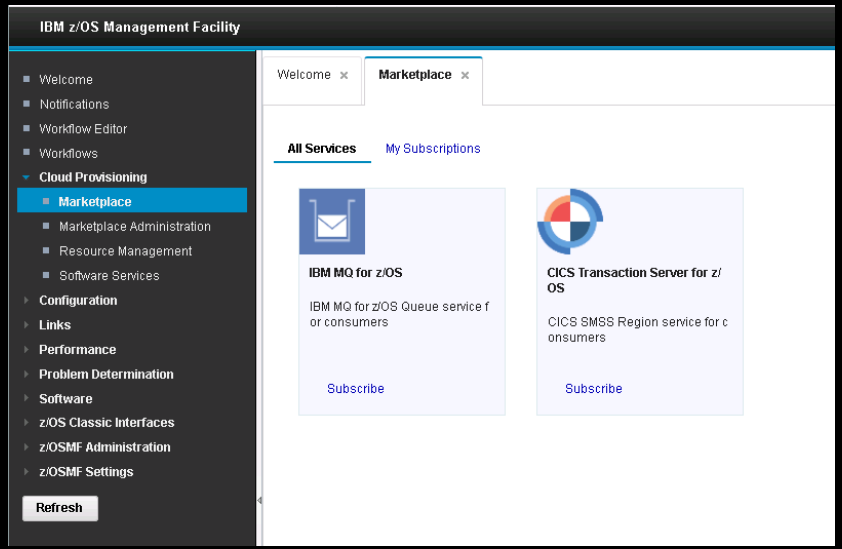
ZD&T EE – Provisioning with Application Discovery



- Integration with Application Discovery; used to identify data sets and CICS resources used by an application. The identified data sets can then be automatically provisioned with other components to a new ZD&T instance

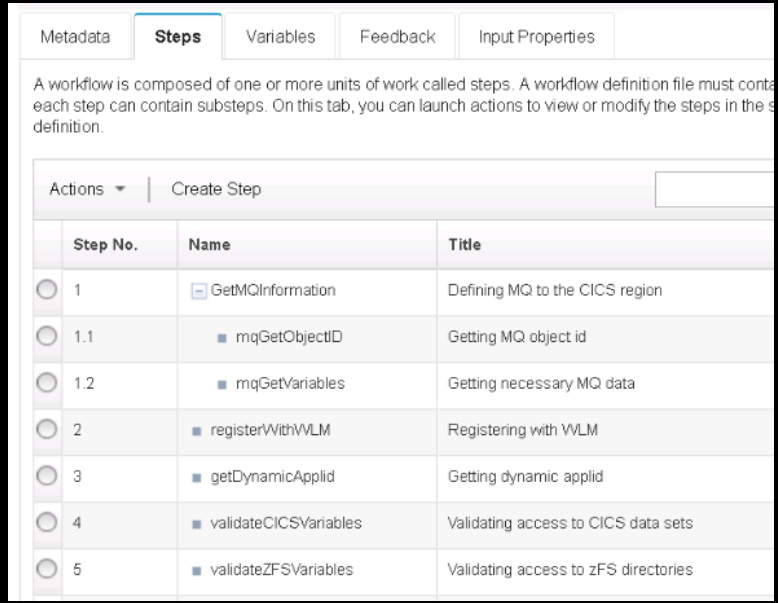
Provision: z/OS Subsystems

- Z/OSMF Provisioning Toolkit
- IBM templates for CICS, MQ, DB2, …
- “Home grown” z/OSMF flows
- Dashboard

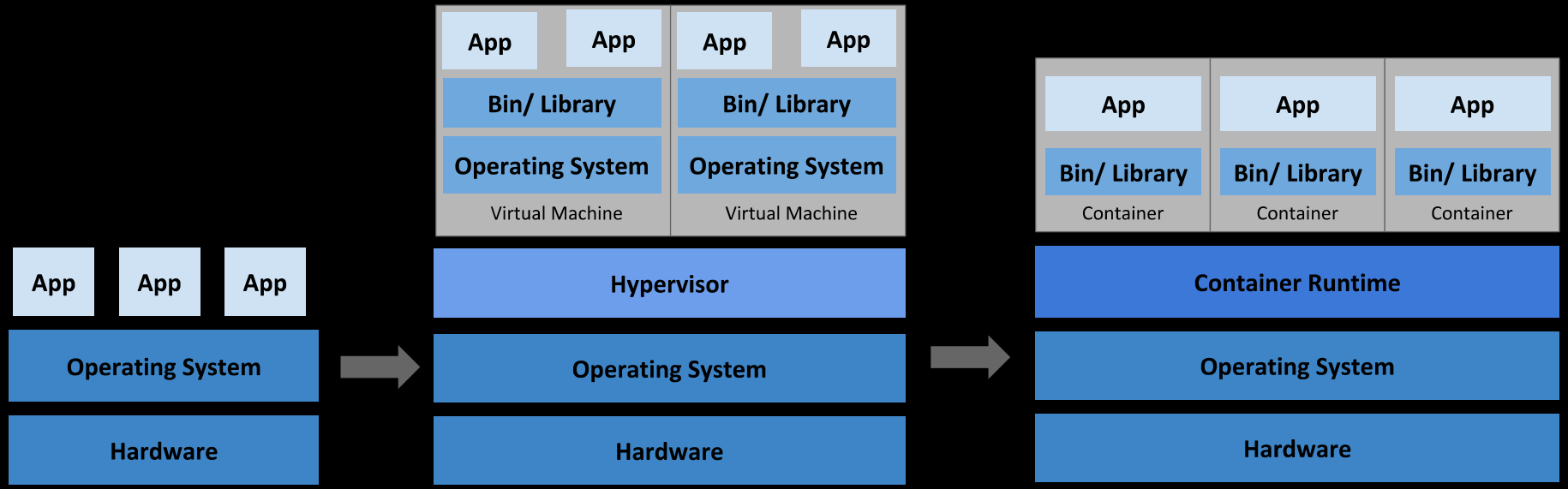


Provision: z/OS Subsystems

- A Sysprog will provide the templates
  - Defines the definition
  - Decides what parameters, if any the developer will provide
- Developers can select a template



Kubernetes

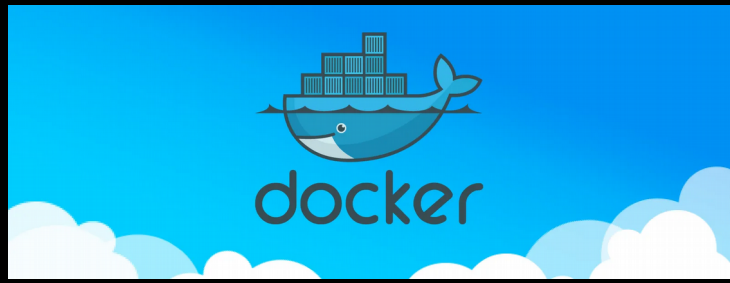

App
App
App
Operating System
Hardware
Traditional Deployment
App
App
App
App
Bin/ Library
Bin/ Library
Operating System
Operating System
Virtual Machine
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware
Containerized Deployment

Provision: z/OS Cloud Broker

- Bridge between z/OS services and Red Hat OpenShift (with Kubernetes)
  - Services can be built in z/OSMF
  - z/OS Cloud Broker runs in a Docker pod in OpenShift

Provision: zCX

- Z Container eXtensions
  - Linux workloads on z/OS
  - As close as you can get to your back-end

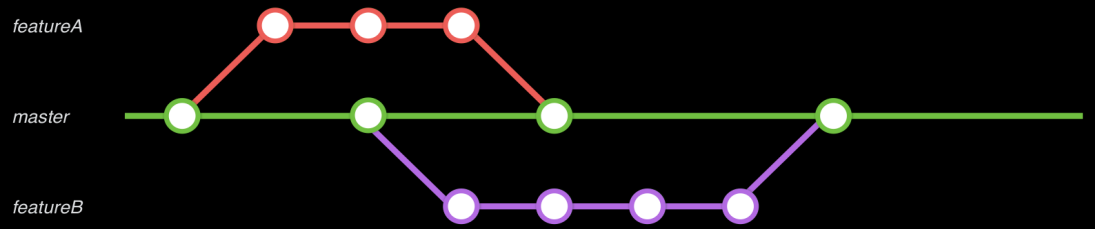


Checks out the code he will work on

RTC EE, Git, Azure Devops, ...

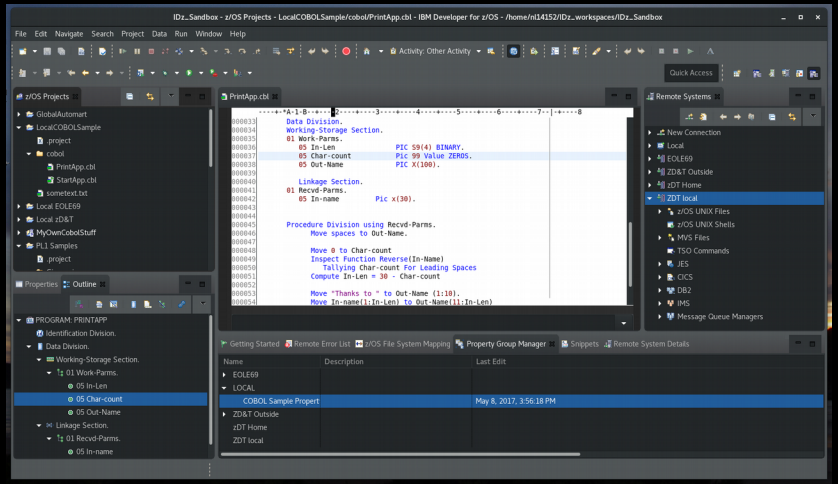
- Modern SCMs
  - Works with z/OS
    - And distributed systems
  - Source code repository does not have to reside on z/OS
- In combination with Dependency Based Build (DBB)

Parallel development with modern SCMs



Develop: IDEs

- ISPF vs modern graphical IDE
  - Debug
  - Unit test
  - SCM Integration
  - Coding rules



Unit testing: zUnit

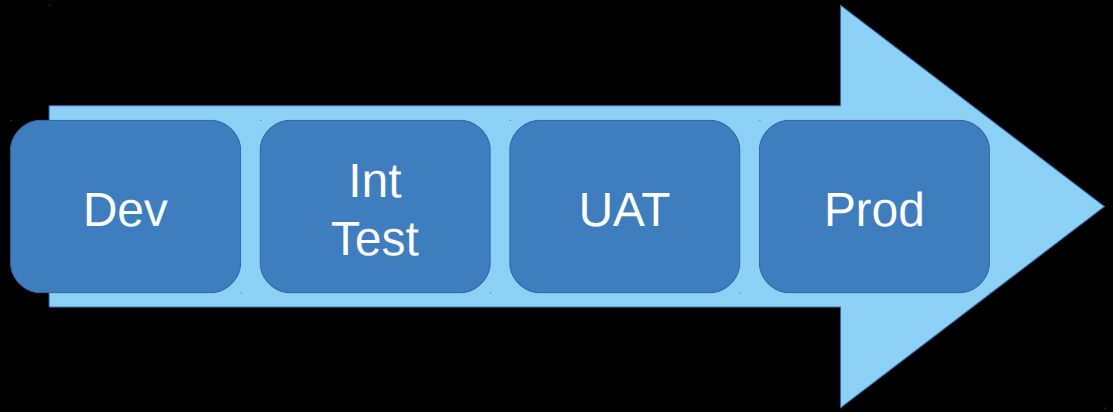
- Record CICS or IMS transactions
- Replay
- Generated test cases in your SCM with the code
- Run it automated after check in

Unit testing: zUnit

- Replay stubs out all the calls
- Makes it possible to do a unit test without having an actual CICS or IMS region available
  - Means zD&T can be simplified when using it just for unit testing

Deploy

- Check in your branch to RTC EE or Git or Azure Devops or …
- Perform an (automated) build
  - DBB, RTC EE, Maven, …
- Deploy to next level
  - UrbanCode Deploy, Jenkins, …



Questions?

Agnes ten Brink

DevOps for IBM Z Technical Sales Benelux

+31 6 5142 1673
agnes.ten.brink@nl.ibm.com
www.linkedin.com/in/agnestenbrink/
www.twitter.com/AgnesTB

Links

zTrials:
https://www.ibm.com/it-infrastructure/z/resources/trial