#SHAREorg

SHARE

Stay Ahead with a Smarter Approach to DevOps

Rosalind Radcliffe (@rosalindrad)
IBM Distinguished Engineer
Rational Enterprise Modernization Solution Architect
rradclif@us.ibm.com

SHARE in Boston

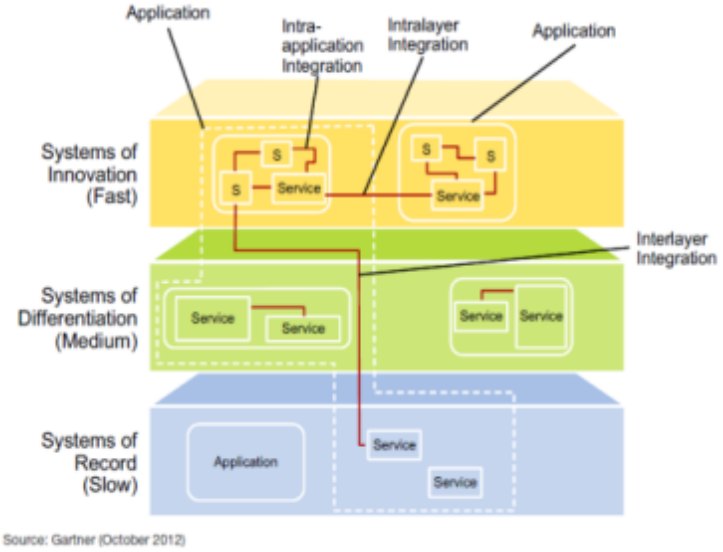
New era systems integrate existing operational systems with rapid delivery of new client-facing apps

Manage workloads and maintain security

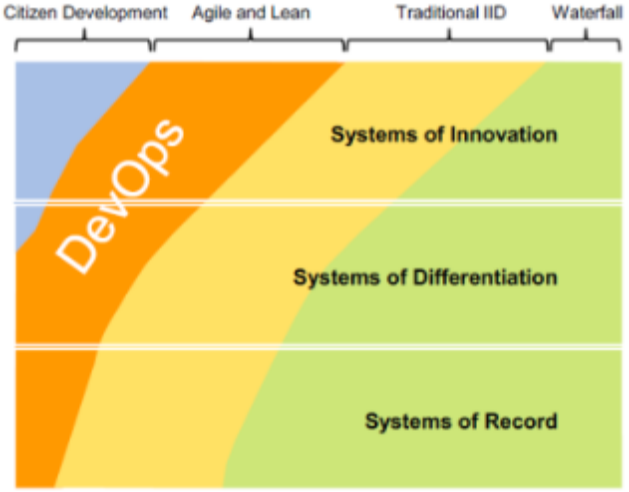
Rapid innovation in the cloud

User experience and mobile management

Evolving customer and market expectations



Source: Gartner (October 2012)



Source: Gartner (October 2012)

Capabilities and User Experience	Today	Emerging
Primary Workload Types	**Systems of Record** *Transactional*	**Systems of Engagement** (+ Record) *Big Data, Analytics, Mobile/Social Channels*
Time to Value	Planned	Opportunistic
Delivery Model	Planned	Incremental (DevOps)
Development and Operations Team Sizes	100s and Costly	10s with built-in DevOps automation
Release Frequency	**Months to Years**	**Hours to Days, based on business opportunity**
Integration Frequency	Weeks	Continuous
Service Sourcing	Develop	Consume and Assemble (Public and Private)
Operational Model	Systems Management	Built in to application, Recovery Oriented Computing, Continuous Availability
Infrastructure Deployment	Days	Minutes
Risk Profile	Big-Bang (High Risk)	Incremental

Agenda

- What is DevOps, Continuous Integration, and Continuous Testing?
- IBM DevOps Solution
- DevOps Challenges for System z
- Continuous Integration for System z
- Continuous Testing for System z

Agenda

- **What is DevOps Continuous Integration, and Continuous Testing?**
- IBM DevOps Solution
- DevOps Challenges for System z
- Continuous Integration for System z
- Continuous Testing for System z

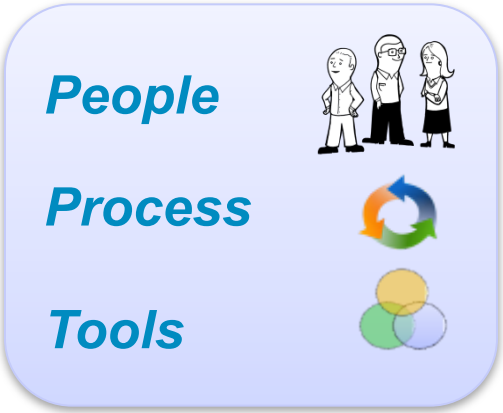
DevOps: A blueprint for continuous delivery of software-driven innovation

dev·ops *noun* \'dev-äps\
Enterprise capability for continuous software delivery that enables clients to seize market opportunities and reduce time to customer feedback.

- Accelerated software delivery
- Reduced time to obtain and respond to customer feedback
- Improved governance across the lifecycle
- Balanced quality, cost and speed

DevOps Principles and Values

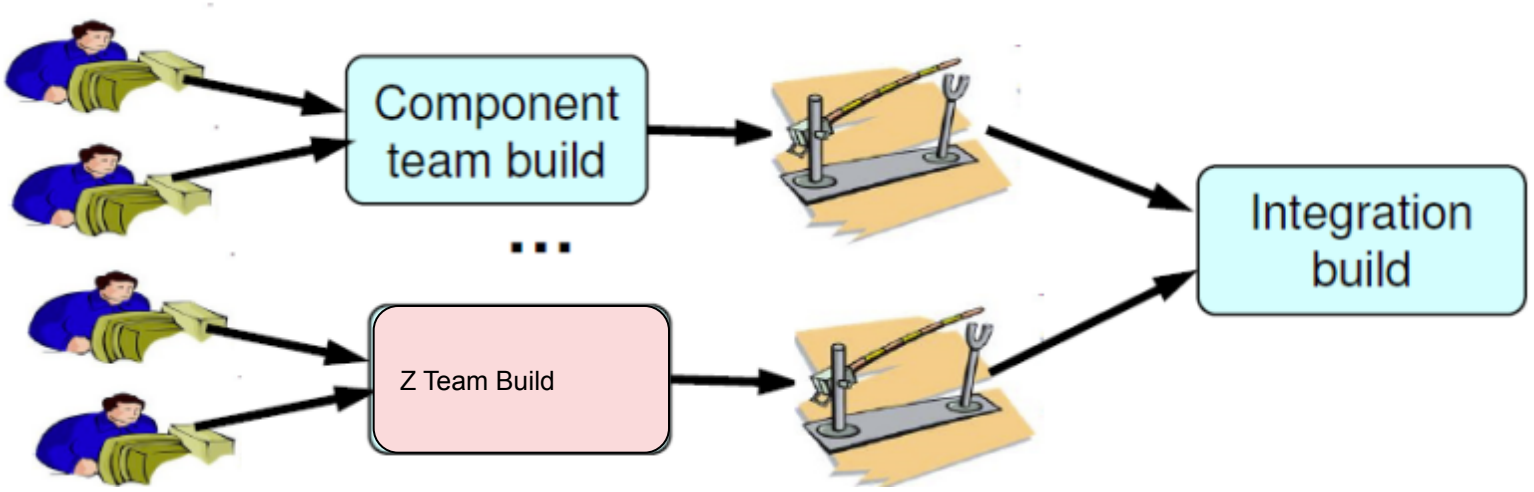
- Develop and test against a production-like system
- Iterative and frequent deployments using repeatable and reliable processes
- Continuously monitor and validate operational quality characteristics
- Amplify feedback loops



Key Concepts

The key concepts of DevOps

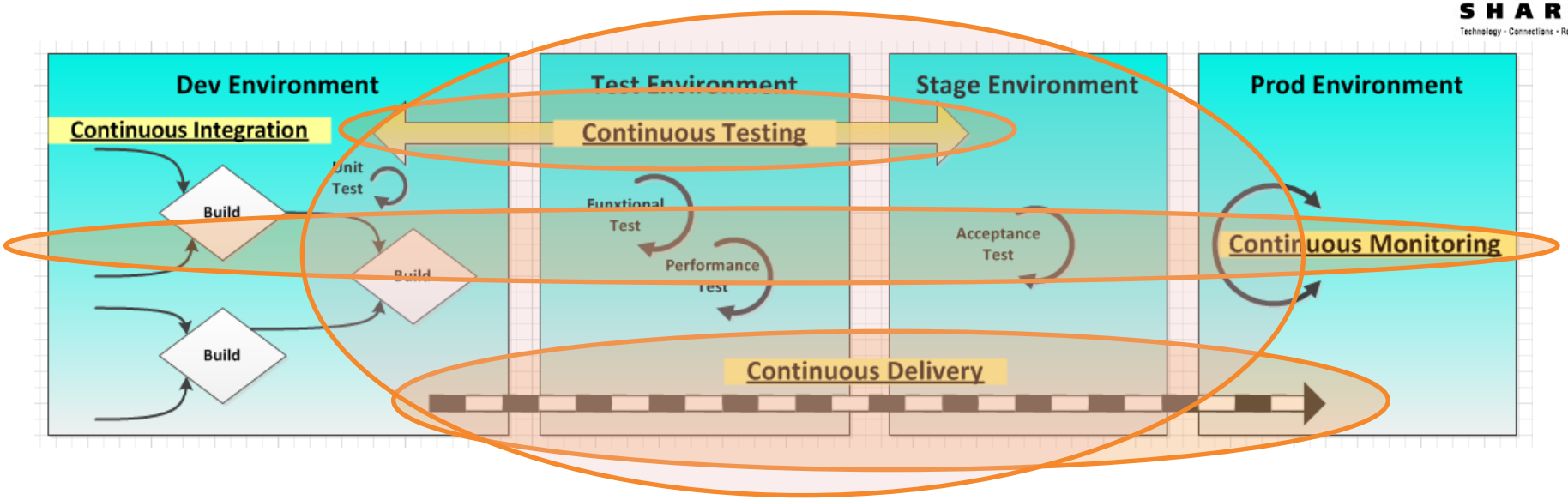
1. Continuous Integration
2. Continuous Delivery
3. Continuous Test
4. Continuous Monitoring
5. Build and Delivery Pipeline
6. Infrastructure as Code
7. Organizational Change



1. Continuous Integration

"Continuous Integration is a software development practice where members of a team integrate their work frequently; usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible."

- Martin Fowler



2. Continuous Delivery

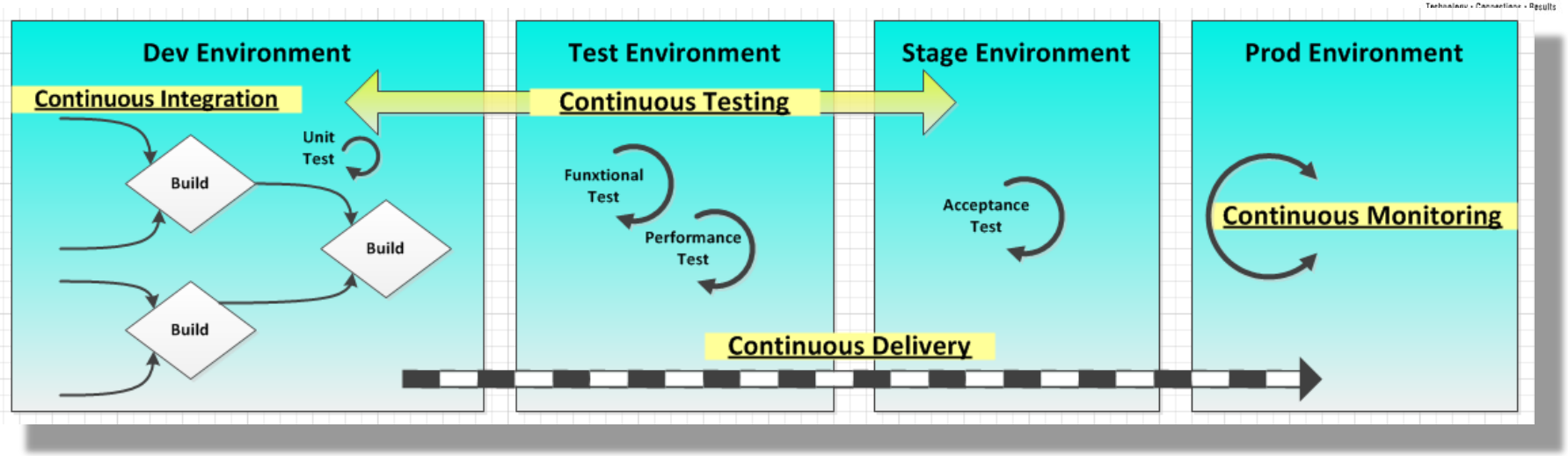
3. Continuous Testing

4. Continuous Monitoring

5. Build and Delivery Pipeline

```
/* REXX */
/* REXX BIND processor sample */
 trace o
 Arg PACKAGE DBRM

 rcode = 0

 /* Set BIND options */
 SYSTEM = 'DSN9'

 i = Pos('(', DBRM)
 len = Length(DBRM)
 LIBRARY = Substr(DBRM, 1, i - 1)
 MEMBER = Substr(DBRM, i + 1, len - i - 1)

 OWNER = 'DEVDBA'
 ACTION = 'REPLACE'
 VALIDATE = 'RUN'
 ISOLATION = 'CS'
 EXPLAIN = 'NO'
 QUALIFIER = 'DEVDBA'

 Call Bind_it

Exit rcode

Bind_it:

/* Create a bind control statement as a single long line. Then */
/* queue that into a FIFO stack */
 DB2_Line = "BIND PACKAGE("PACKAGE")" ||,
 " LIBRARY("'LIBRARY'")" ||,
 " MEMBER("MEMBER")" ||,
 " OWNER("OWNER")" ||,
 " ACTION("ACTION")" ||,
 " VALIDATE("VALIDATE")" ||,
 " ISOLATION("ISOLATION")" ||,
 " EXPLAIN("EXPLAIN")" ||,
 " QUALIFIER("QUALIFIER")"

 /* Write the bind control statement to the data queue and execute */
 /* DB2I to perform the bind. */

 queue DB2_Line
 queue "End"
 Address TSO "DSN SYSTEM("SYSTEM")"
 rcode = RC

Return
```

6. Infrastructure as Code / Software Defined Environment



'Shift Left' – Operational Concerns
Build 'Application aware' Environments
Environment Sprints

7. Organizational Change

10 Principles of Continuous Integration

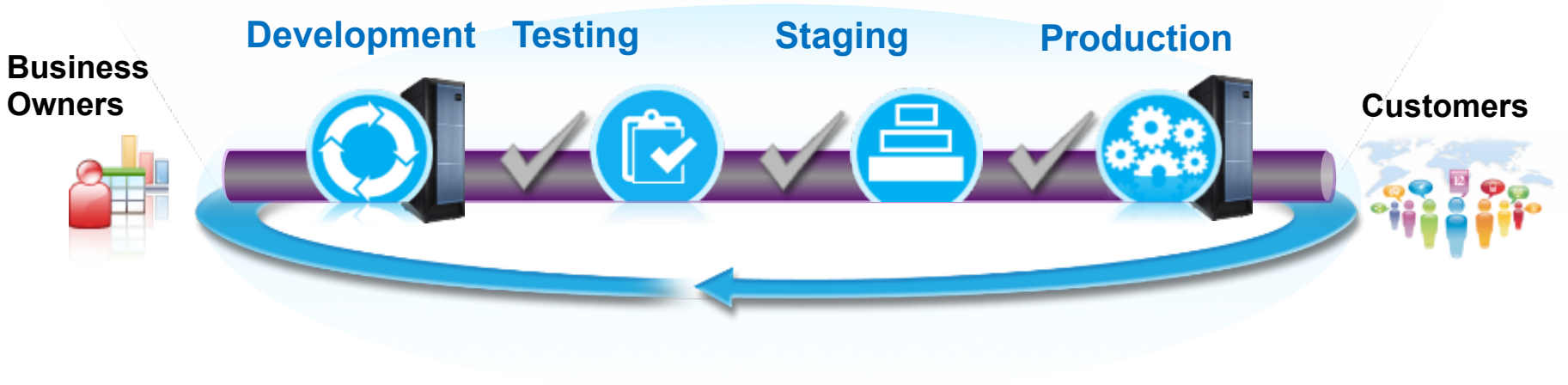
1. Maintain a Single Source Repository
2. Automate the Build
3. Make Your Build Self-Testing
4. Everyone Commits To the Mainline Every Day
5. Every Commit Should Build the Mainline on an Integration Machine
6. Keep the Build Fast
7. Test in a Clone of the Production Environment
8. Make it Easy for Anyone to Get the Latest Executable
9. Everyone can see what's happening
10. Automate Deployment

Agenda

- What is DevOps, Continuous Integration, and Continuous Testing?
- **IBM DevOps Solution**
- DevOps Challenges for System z
- Continuous Integration for System z
- Continuous Testing for System z

IBM DevOps Reference Architecture

Solution: A Continuous Delivery Pipeline

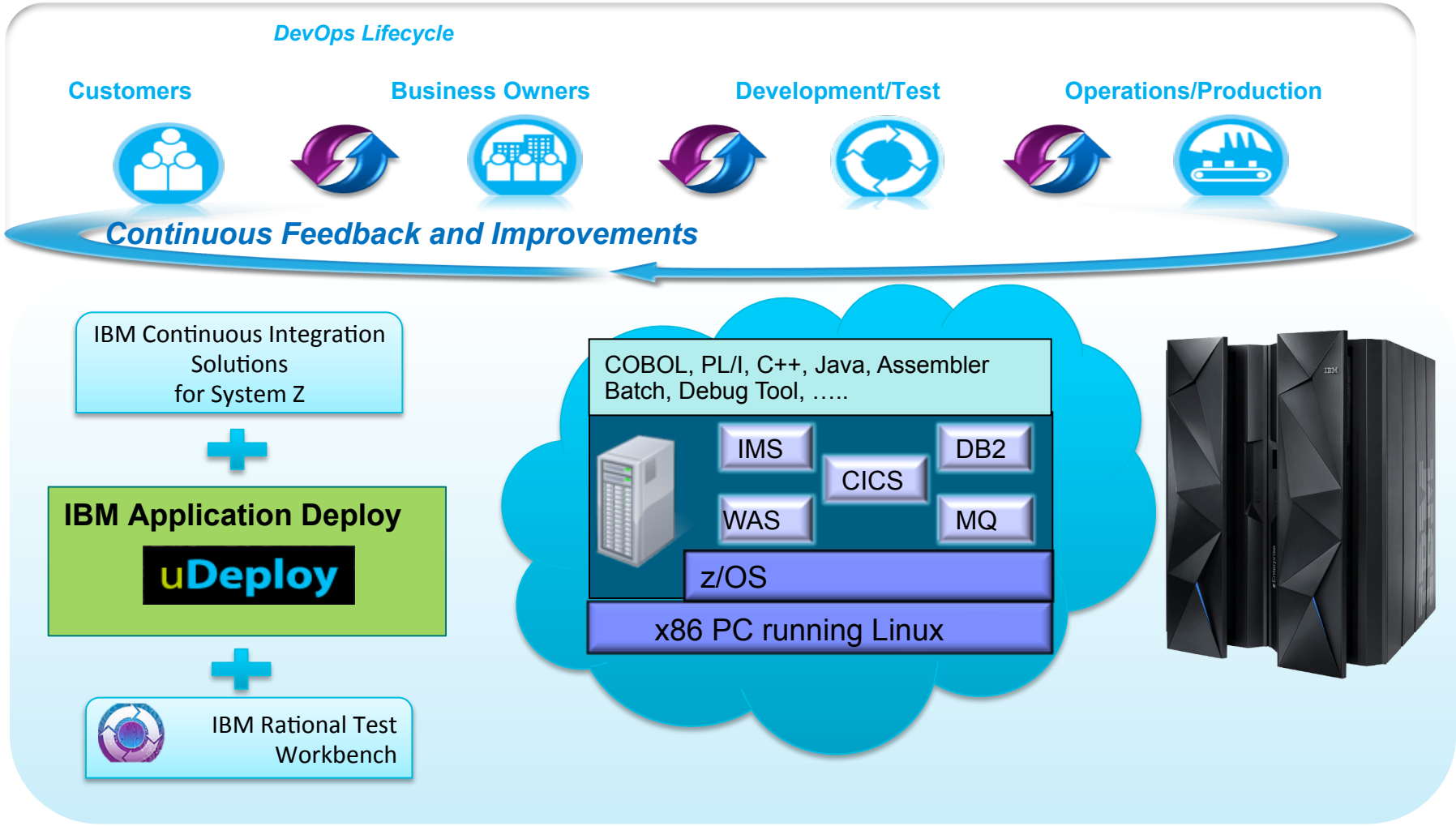


Ensure applications are production-ready throughout the lifecycle
and can be released at any time while minimizing rollback due to quality issues

- Validate on more production-like conditions earlier
- Automate hand-offs/promotions to increase velocity through the different stages
- Standardization on processes and assets between Dev and Ops
- Automated monitoring and dashboarding of quality and performance against service level agreements at multiple stages

Rational Development and Test Environment for System z

Continuous build and test of cross-platform systems



Note: This Program is licensed only for development and test of applications that run on IBM z/OS. The Program may not be used to run production workloads of any kind, nor more robust development workloads including without limitation production module builds, pre-production testing, stress testing, or performance testing.

Agenda

- What is DevOps, Continuous Integration, and Continuous Testing?
- IBM DevOps Solution
- **DevOps Challenges for System z**
- Continuous Integration for System z
- Continuous Testing for System z

Mainframe Delivery Pains…

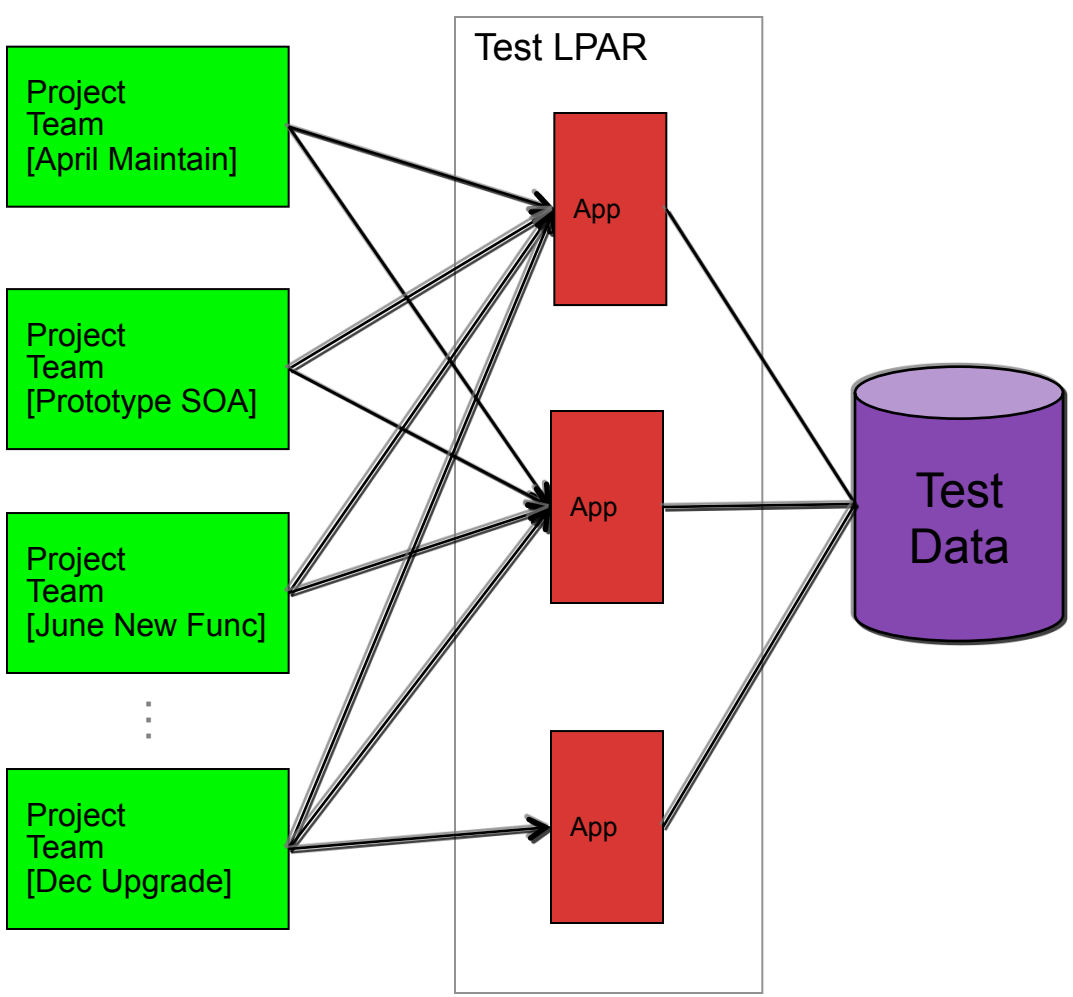
- Multiple teams working across restricted dev and test capacity lead to conflict, delays, or bad test results in shared environments
- Too much bad code going into test and production causes crit sits and emergency fixes
- Complex and manual management and configuration tasks result in errors and delays
- Bottlenecks due to inefficient communications between disparate platforms and teams (Dev/Test - System Programmers; mobile – distributed - mainframe)

…solutions from IBM

- Provide cheap, isolated, development and test environments for project teams
  - Rational Development and Test Environment
  - Rational Test Virtualization Server
  - SmartCloud Provisioning
  - Cloud Ready for Linux on System z
- Enforce base quality standards automatically prior to promotion
  - Rational Test Workbench
  - Rational Quality Manager
  - SmartCloud Application Monitoring
  - OMEGAMON
- Automate consistent build, configure, and deploy processes across all stages
  - Rational Team Concert
  - uDeploy
  - SmartCloud Orchestrator
- Improve communication and collaboration with cross-platform release planning
  - IBM Collaborative Lifecycle Management
  - Smart Cloud Control Desk

Typical z/OS Testing Architecture

Organized by project team, vertically scaled, sharing resources, limited automation



Problems Encountered

1. Shared resources combined with overlapping schedules can elicit conflicts, impede innovation and slow code delivery

2. Coordination of environmental changes and releases cause bottlenecks, delays and additional overhead

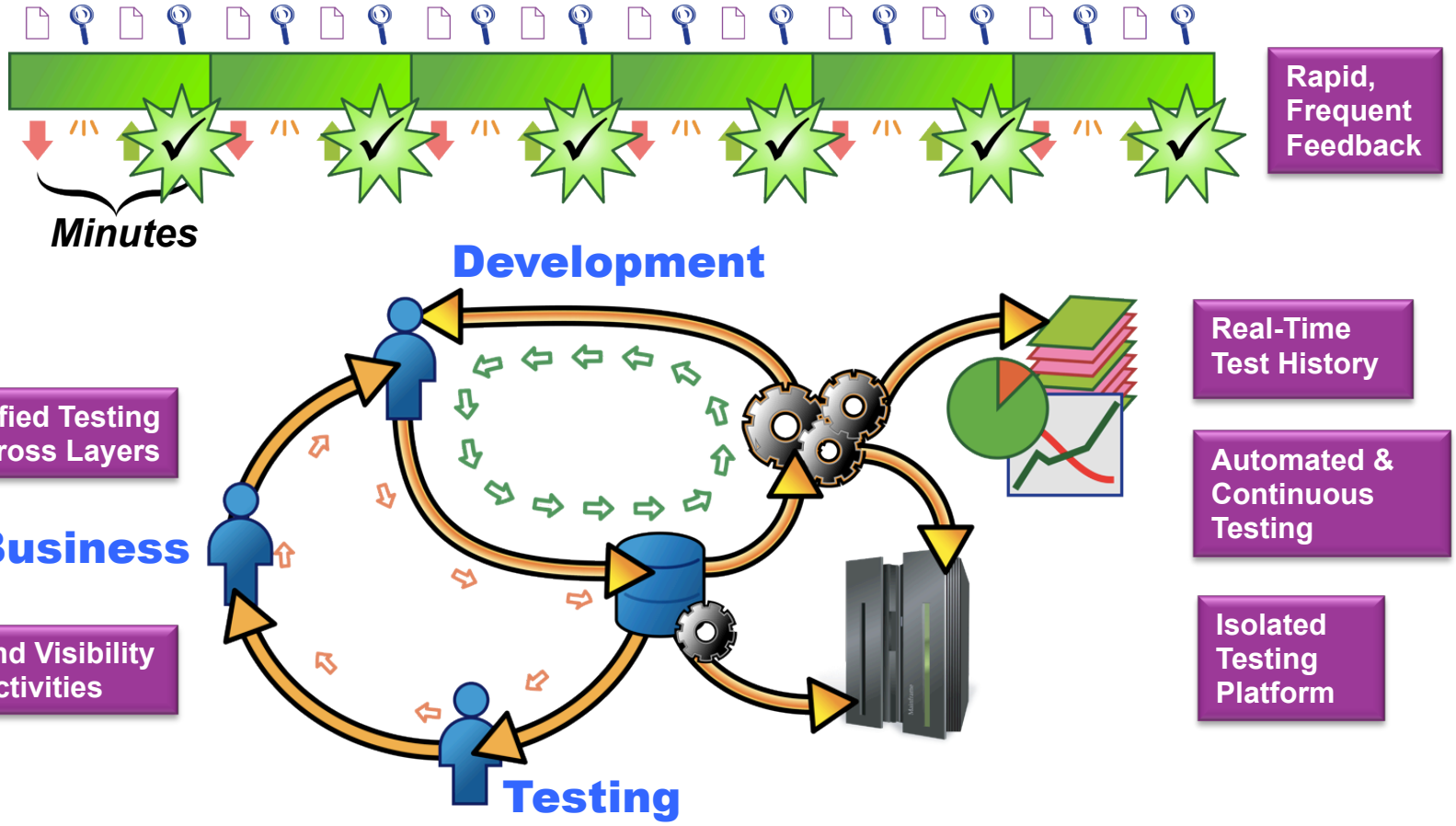
3. Shared test data is difficult to manage and can lead to over testing or incorrect test results

Agenda

- What is DevOps, Continuous Integration, and Continuous Testing?
- IBM DevOps Solution
- DevOps Challenges for System z
- **Continuous Integration for System z**
- Continuous Testing for System z

Continuous Integration

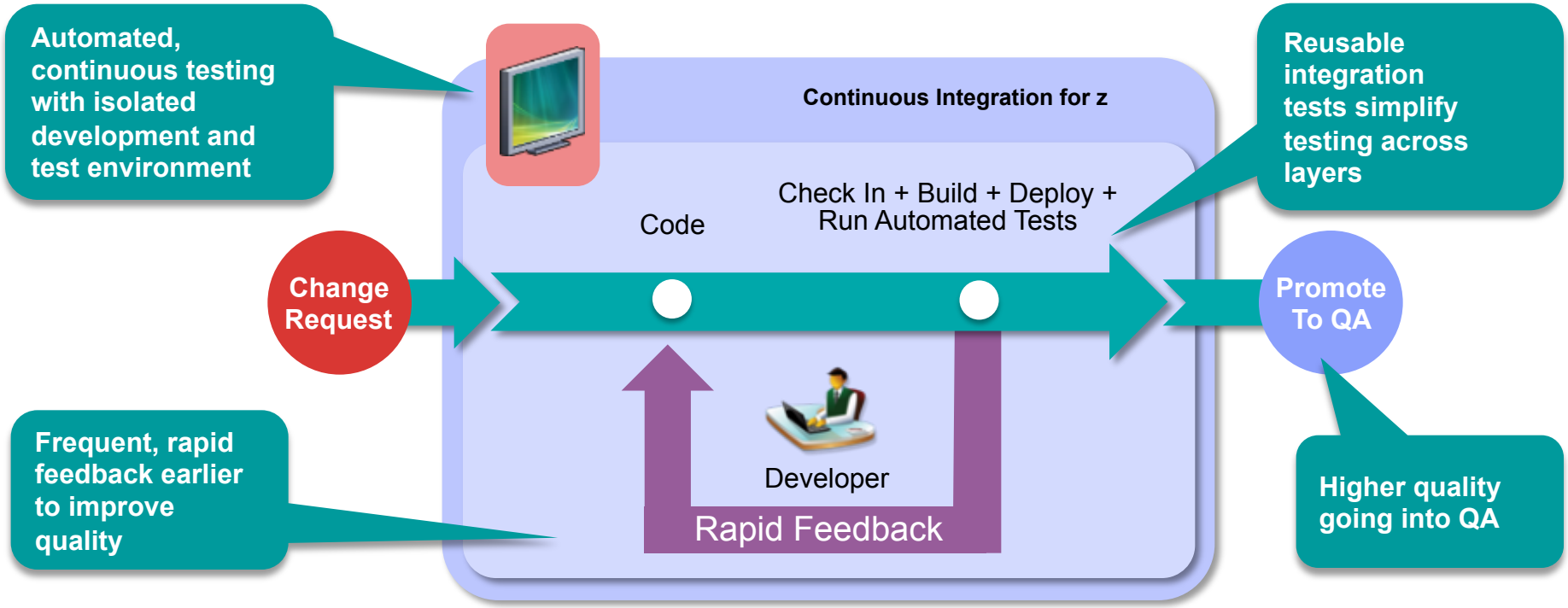
Reduced delivery time, end-to-end visibility of test activities, safer and faster V2V migrations



- ✓ Accelerates: Fast, dependable feedback speeds Time to Market
- ✓ Additive: Embraces existing tools, practices and momentum yet provides value immediately
- ✓ Strengthens: Enables confidence by automatically tracking and promoting code health

Fail fast and drive better quality downstream

Reduced delivery time, end-to-end visibility of test activities, safer and faster upgrades

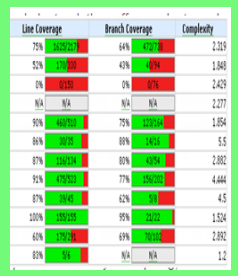


- Fast, dependable, automatic feedback speeds time to market
- Lower cost of application testing using off-mainframe z/OS test environment
- Enables confidence by automatically tracking and promoting code health

Testing and Delivery – moving one step forward

Java / .Net teams

Compile → Automated Unit Test → Install & Configure → Run Automated Tests → Run Manual Tests → Promote Results



Hours to days

Mainframe teams after CIz

Compile → Promote & Configure → Run Automated Tests → Run Manual Tests → Promote Results

Days to Weeks

Agenda

- What is DevOps, Continuous Integration, and Continuous Testing?
- IBM DevOps Solution
- DevOps Challenges for System z
- Continuous Integration for System z
- **Continuous Testing for System z**

Decouple subsystem teams using Green Hat

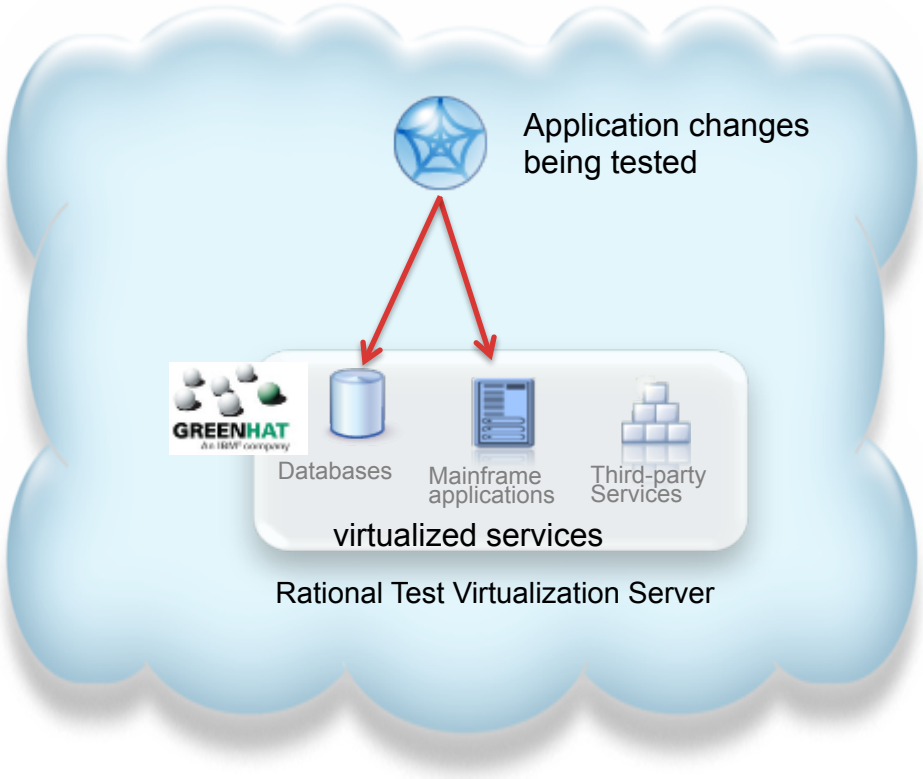
Improve testing with Green Hat service simulation

Simulate subsystem dependencies

- Test impact of latency
- Test application response to unresponsive services

Create a stable test environment

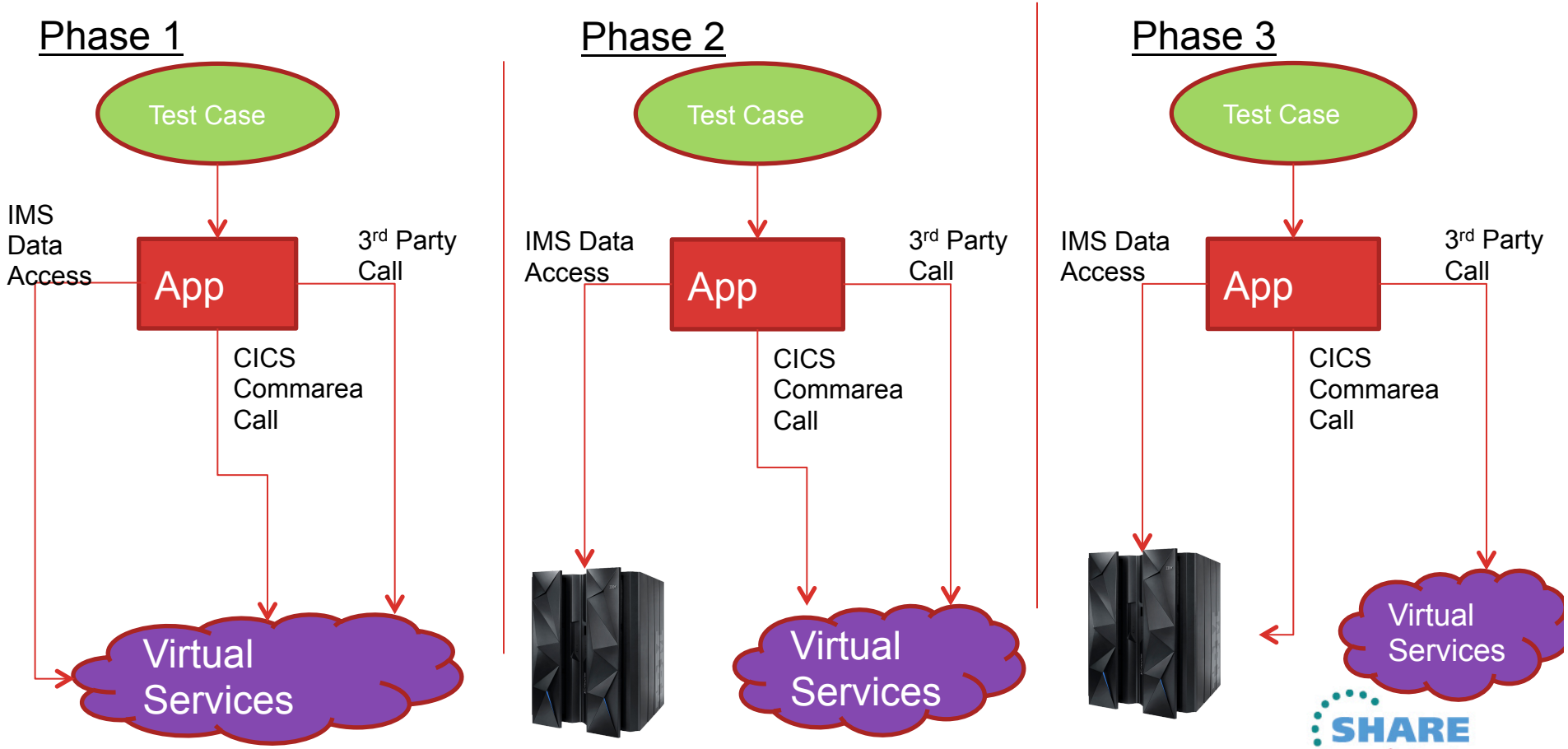
- Simulate “public” services
- Reduce capacity requirements on infrastructure
- Improve security by reducing access from external services



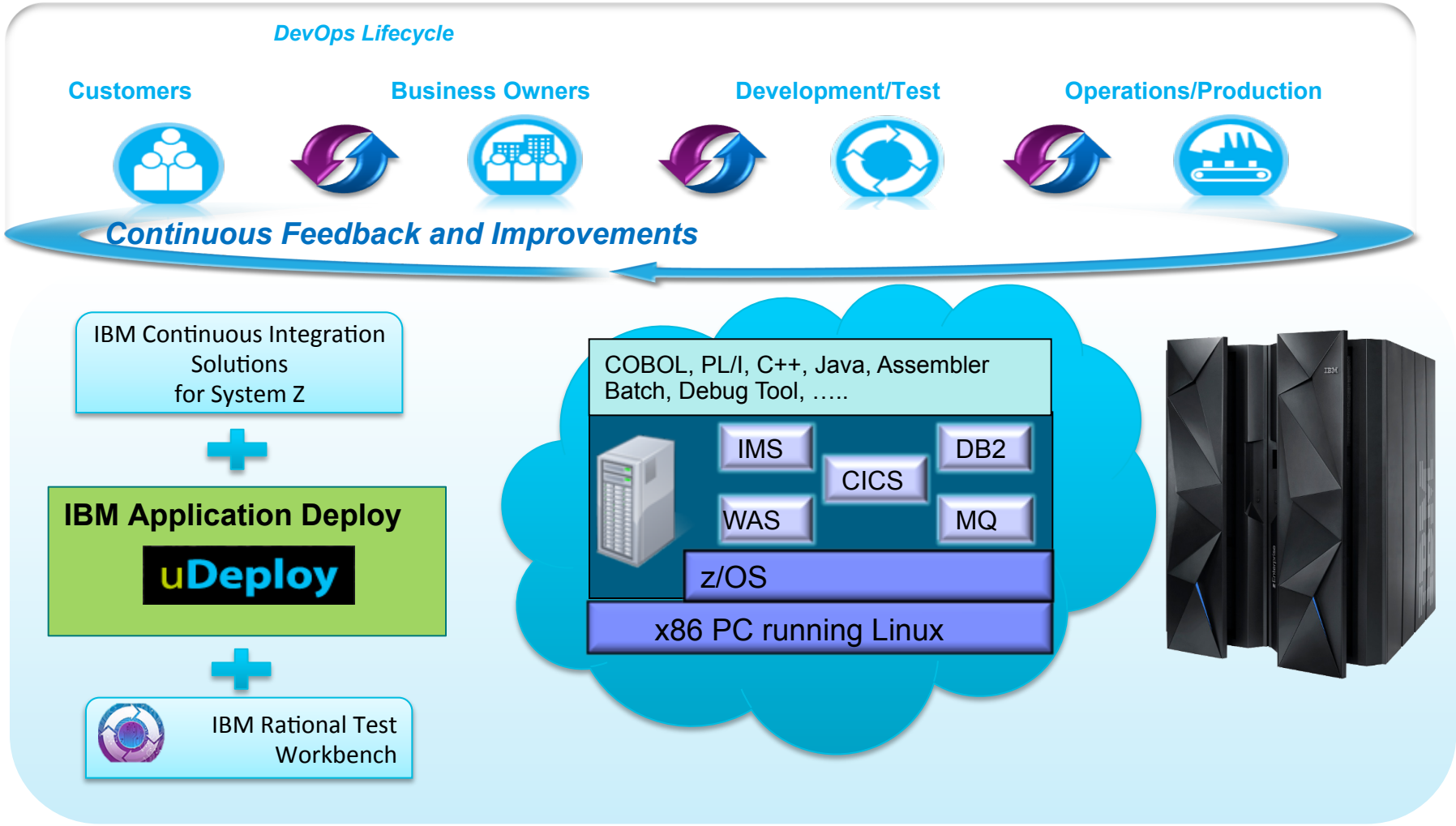
Testing with dependency virtualization

Controlled large system testing by isolating components under test

- Easier problem determination
- Lower test environment capacity requirements
- Improved component quality



DevOps for System z



Note: This Program is licensed only for development and test of applications that run on IBM z/OS. The Program may not be used to run production workloads of any kind, nor more robust development workloads including without limitation production module builds, pre-production testing, stress testing, or performance testing.

Questions

Acknowledgements and disclaimers

Availability: References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Thank You