DevOps for the Mainframe

Rosalind Radcliffe

IBM Distinguished Engineer, Enterprise Modernization Solution Architect

rradclif@us.ibm.com

Please note

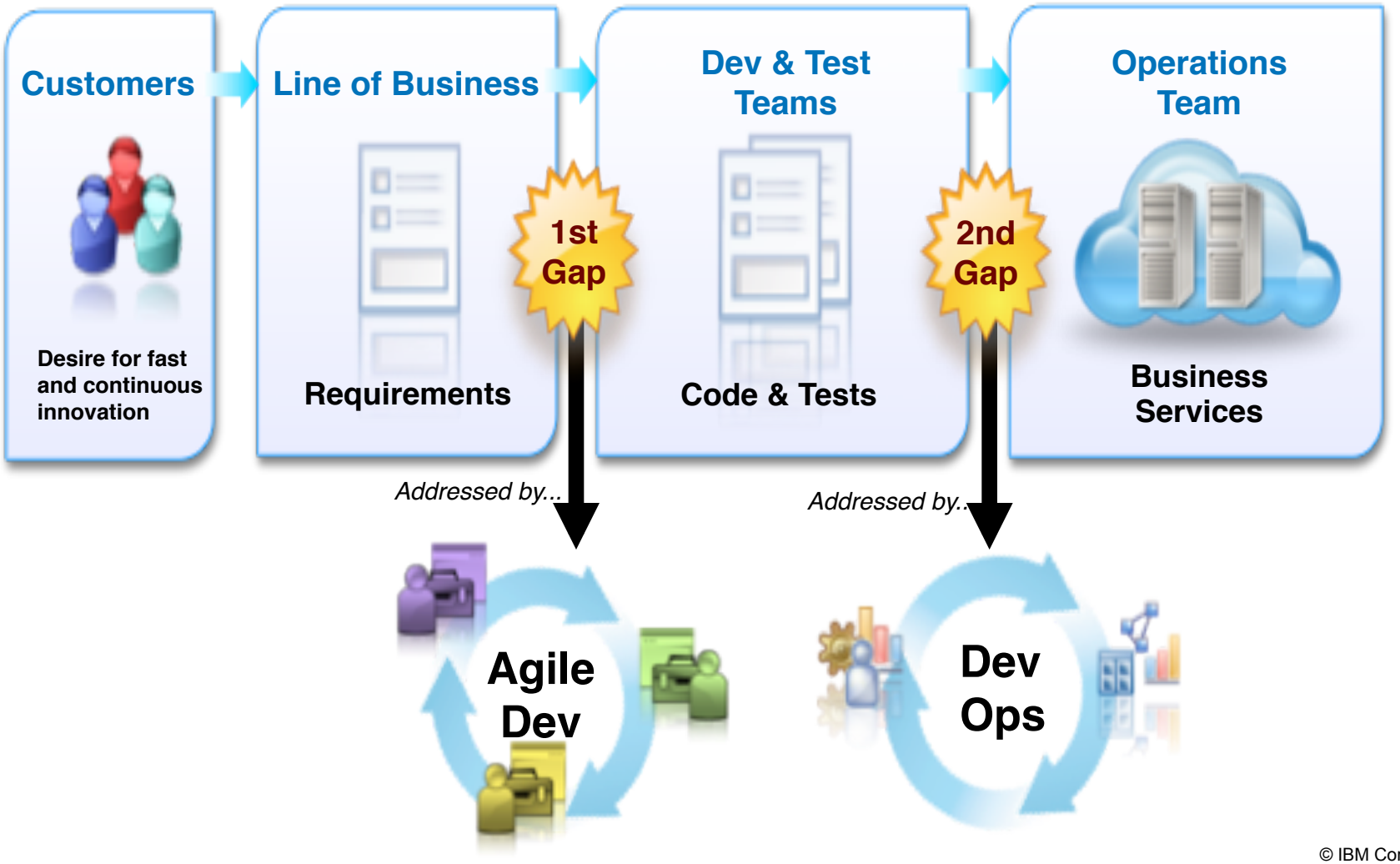
IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

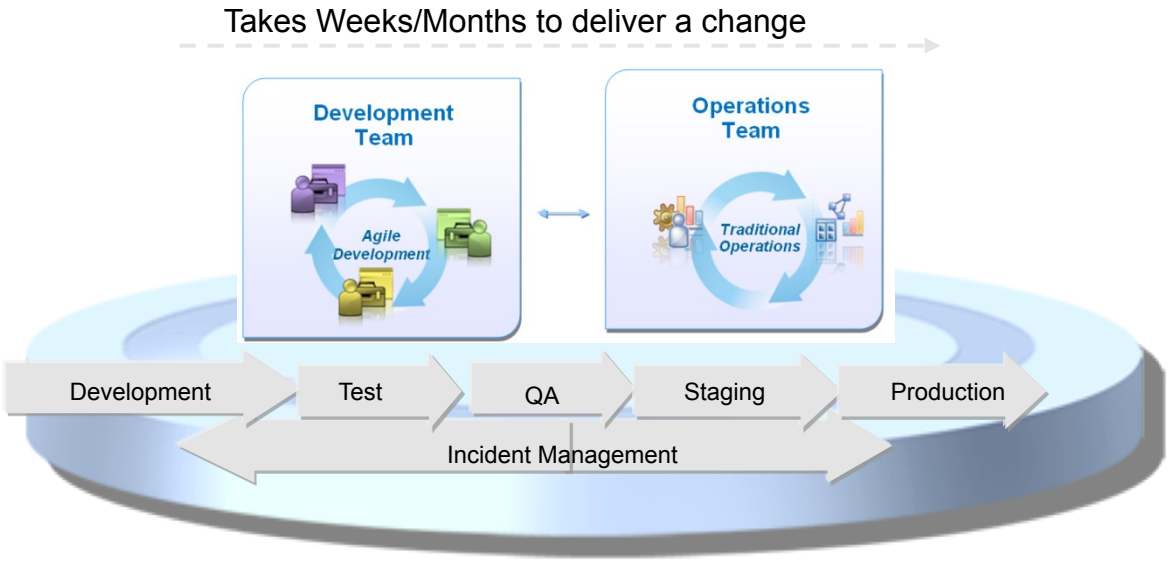
Challenges delaying delivery of functionality to the business



Software Delivery Challenges: what we hear from customers

Needs:

- Reduce cycle time and delays
- Improving software delivery efficiencies with standardization and automation
- Improving Quality of Delivery and reducing roll-backs



Quality Challenges

- Difficulties in reproducing production defects
- Long time to fix defects
- Poor Test coverage
- Lack of automated testing

Release Challenges

- Differences in Dev/Ops environments
- Siloed / Limited automation
- Long set up time

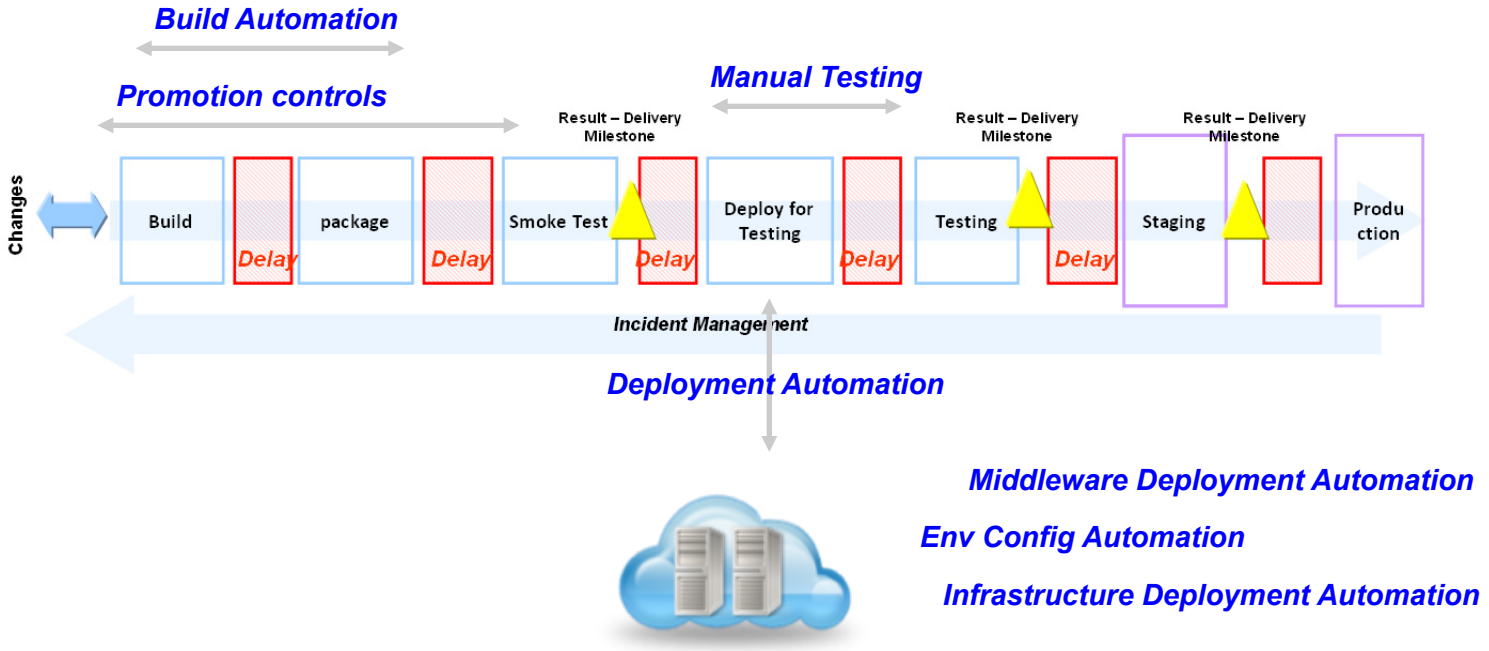
Process and Cultural challenges

- Point-Point, adhoc and Fragile integration of tools
- Poor visibility, stability and extensibility
- Cultural barriers limiting collaboration
- Heavy-handed control of dev environments

Simplified view of a single-release pipeline. In general, there are multiple projects, releases, and technologies at play

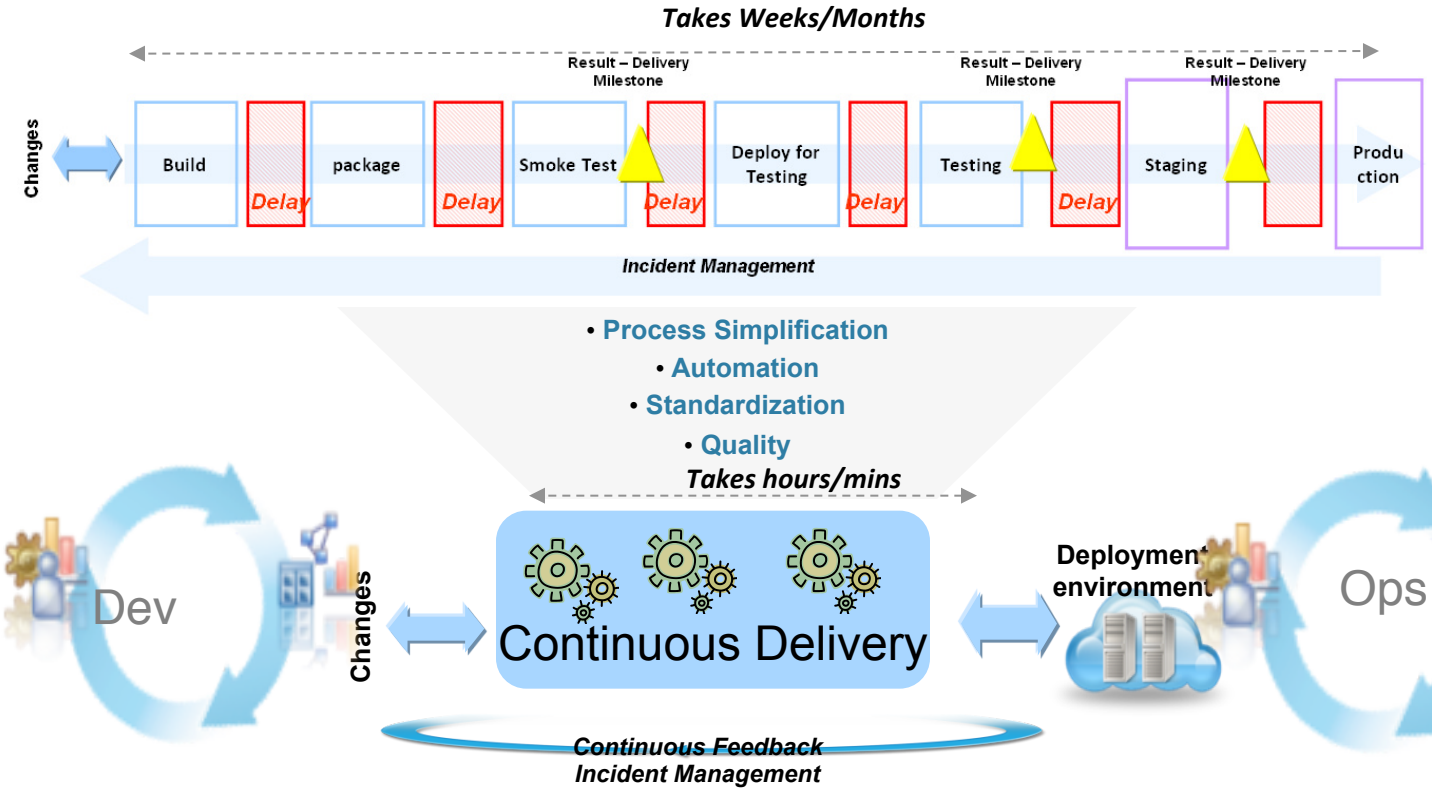
Current Customer approaches addressing these challenges..

- Selective & Siloed automation of the delivery process with limited benefits
- Poor visibility and control impacting cycle time
- Coordinated automated and manual processes



Need for a Simple approach to bringing agility across the lifecycle

Continuous, orchestrated, and automated delivery of changes leveraging Cloud

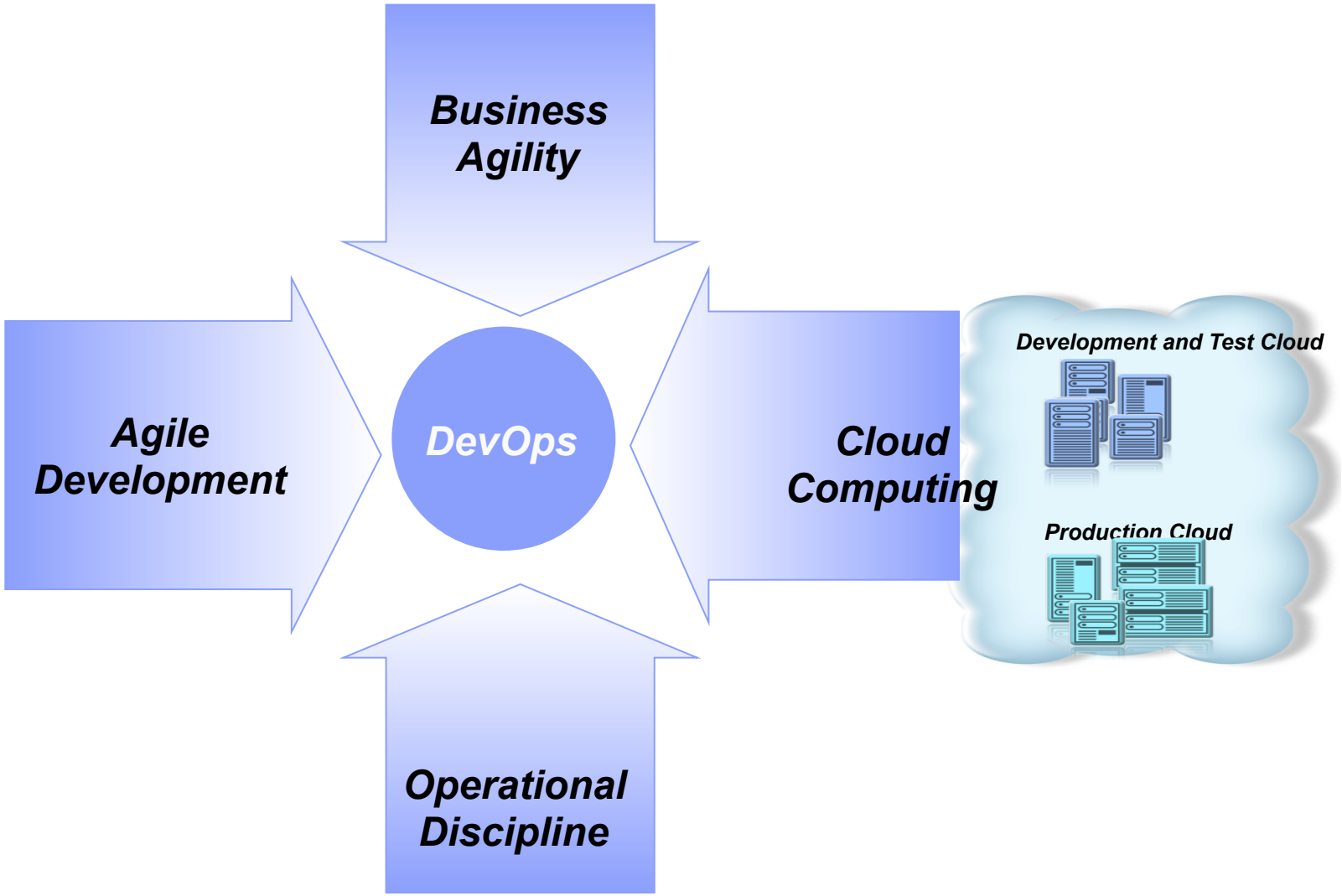


- Long Cycle Time
- Poor Visibility and Control

- Reduced cycle time
- Control, Agility and scalability

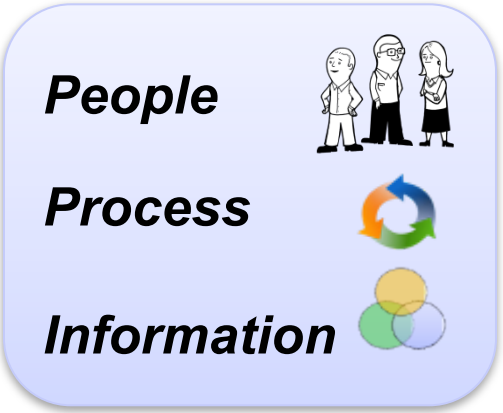
Time is now for DevOps

Trends accelerating the need for Continuous Delivery



DevOps: Principles & Values

- Collaborate across disciplines
- Develop and test against a production-like system
- Deploy frequently
- Continuously validate operational quality characteristics

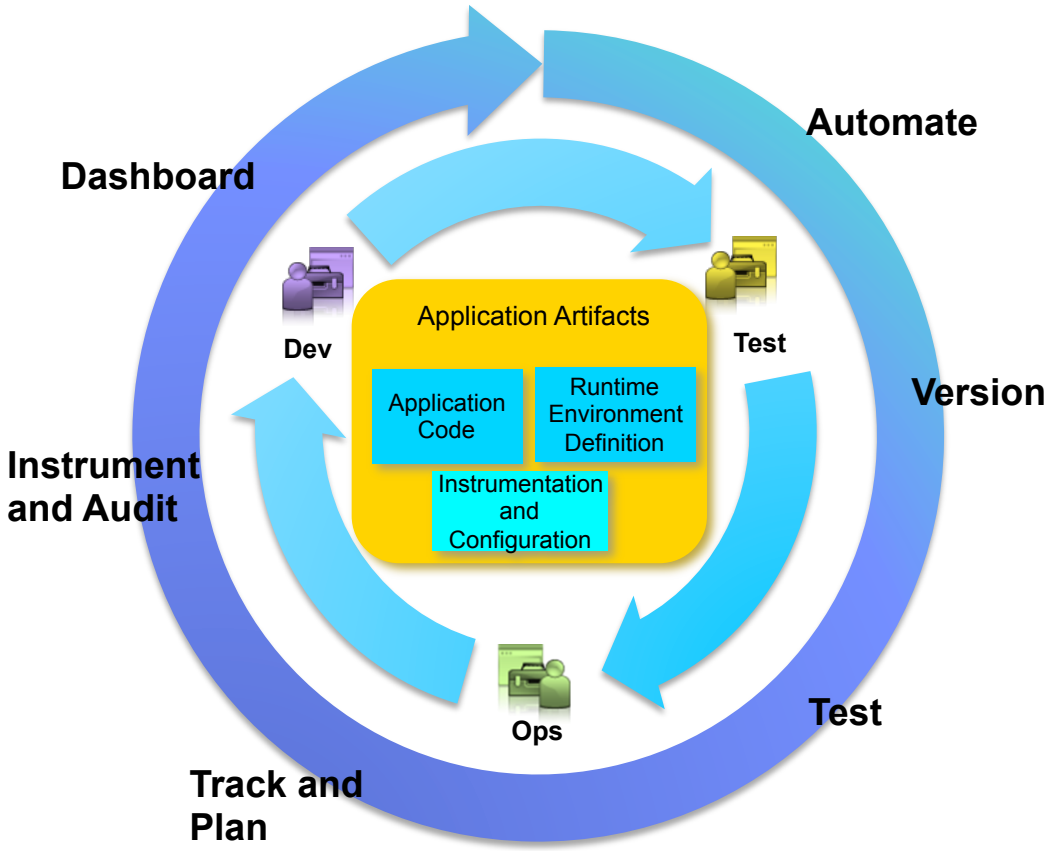


Results in:

- Rapid evolution of deployed business services
- Reduced risk
- Decreased cost
- Improved quality across the portfolio

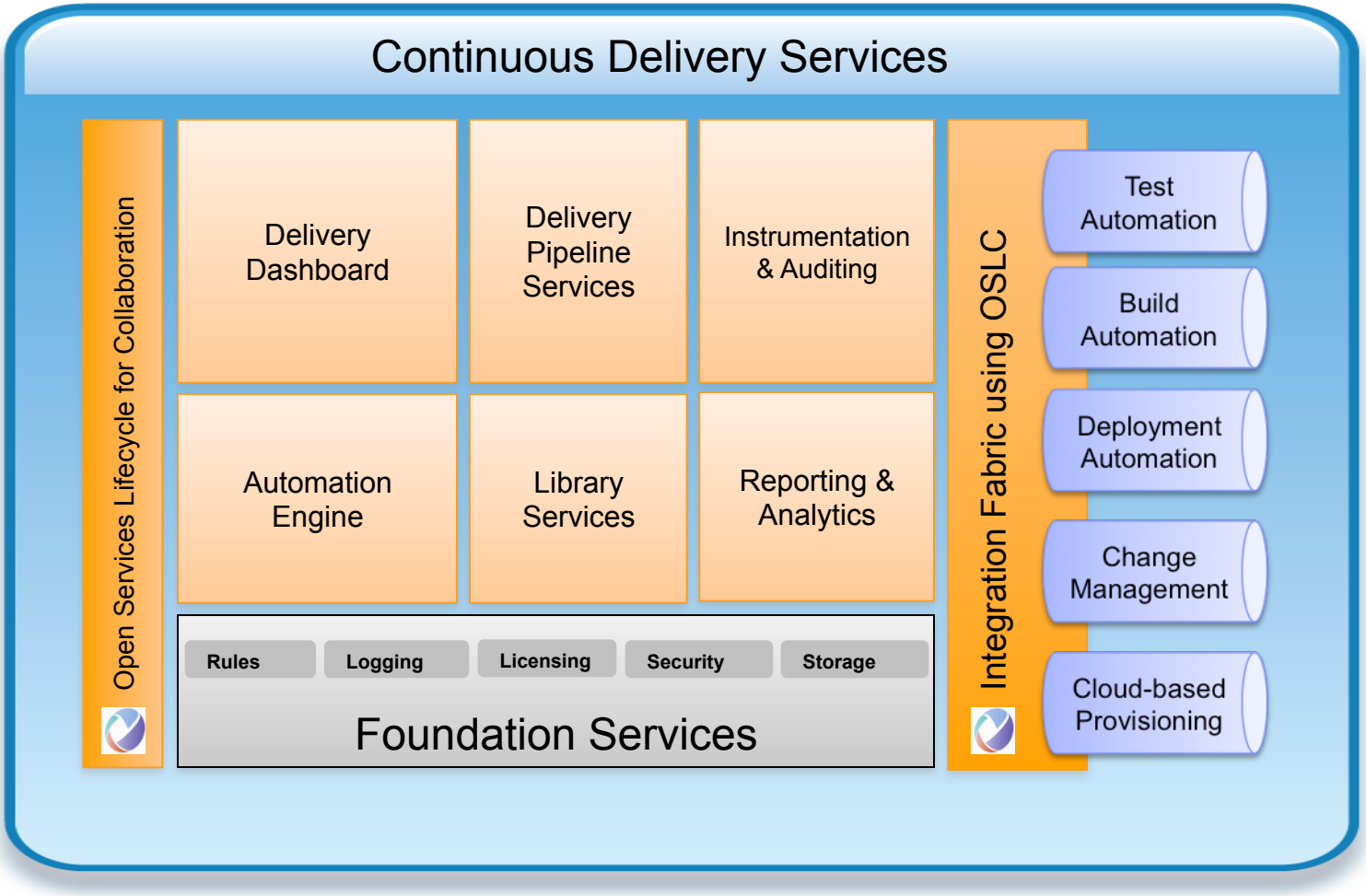
How do we make this happen?

- **Automate** *everything*
- **Version** *everything*
- **Test** *everything*
- **Track and Plan** *everything*
- **Instrument and Audit** *everything*
- **Dashboard** *everything*



Continuous Delivery Reference Architecture

Built on open standards allowing plug-in components from IBM products, open source, and third party



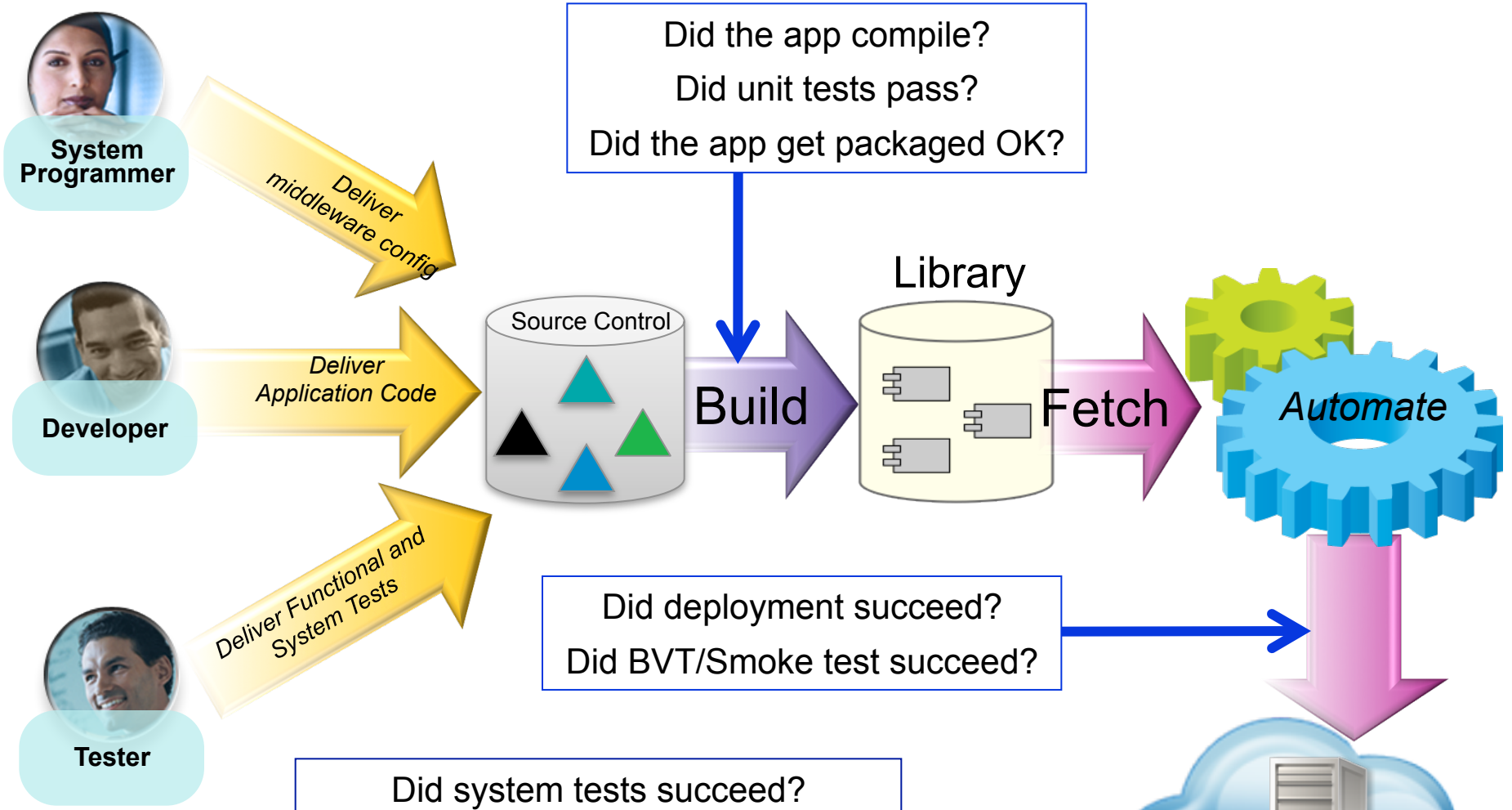
Mainframe DevOps Adoption Strategy

1. Adopt a test-everything strategy using continuous integration build approach with (automated) self-validating builds [Automate and Test]
2. Unify mainframe application asset change management to ease orchestration [Version, Track, and Plan]

Available Today from Rational

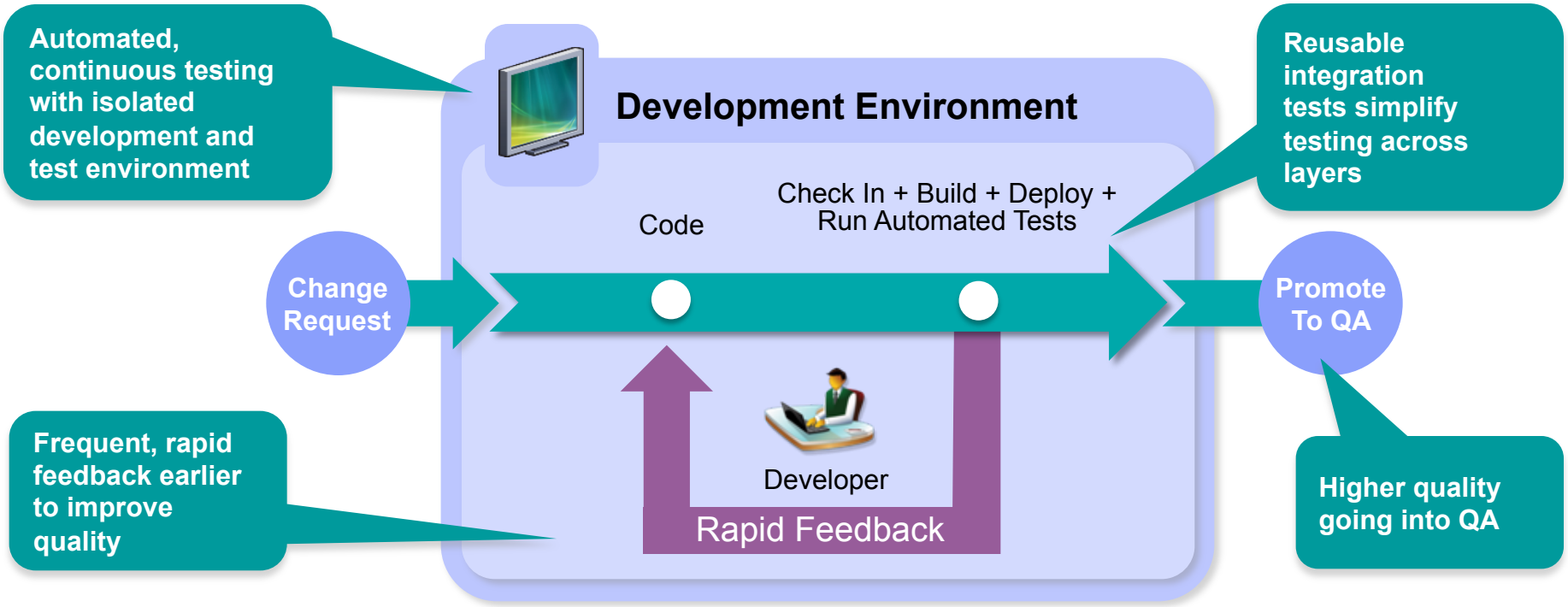
3. Document standardized environments and drive cloud provisioning of isolated mainframe images and applications based on standards [Automate, Instrument, and Audit]
4. Optimize provisioning orchestration to include dependency virtualization and test data conditioning

Test *Everything:*
Continuous, automatic testing across the lifecycle



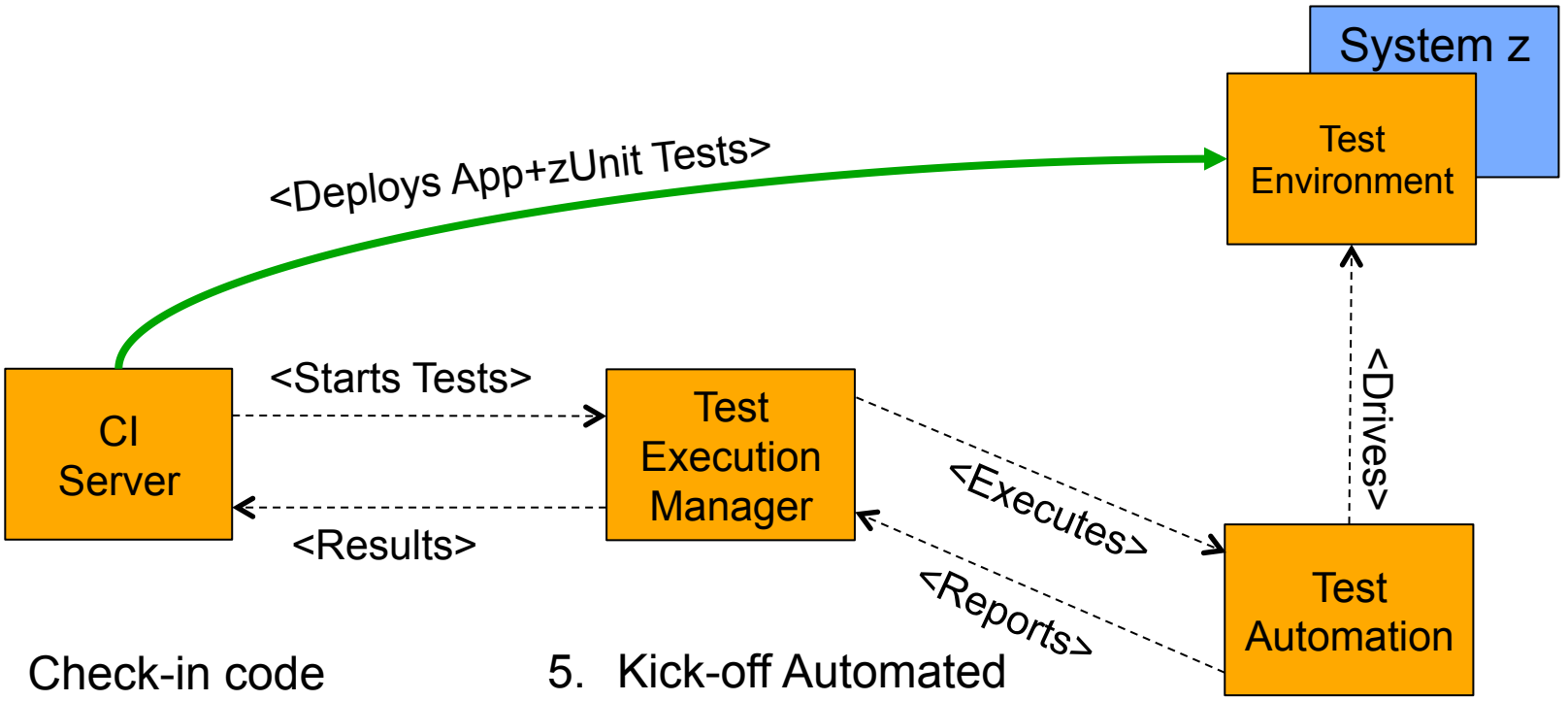
Solution: Continuous Integration

Reduced delivery time, end-to-end visibility of test activities, safer and faster upgrades (V2V)



- 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

Detailed Continuous Integration for System z Scenario



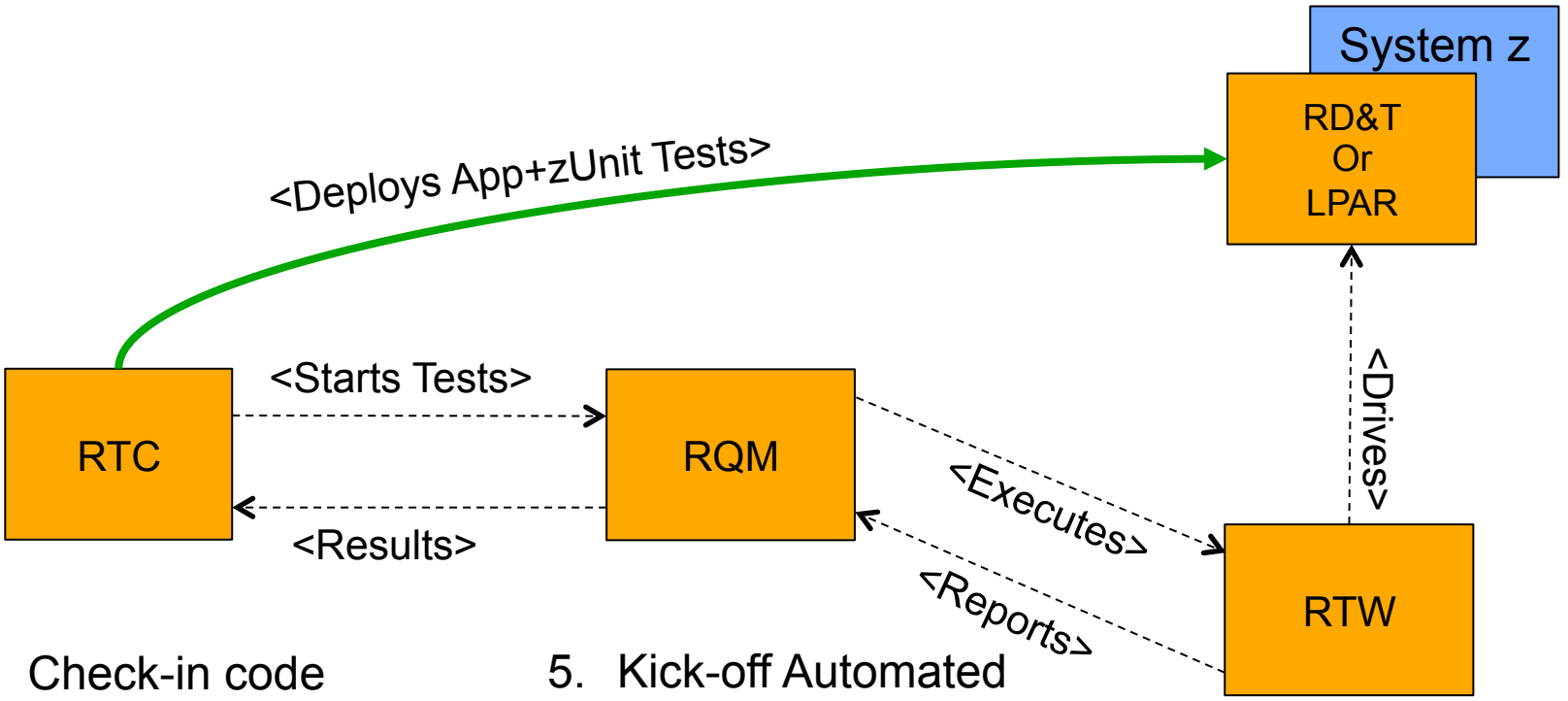
1. Check-in code
2. Build code and zUnit tests
3. Deploy build results to Test Environment
4. Execute zUnit Tests

5. Kick-off Automated Test Plan

8. Report test results in dashboard/build results/defect records in CI server.

6. Run automated interface tests against Test Environment
7. Mark execution records Pass/Fail in Test Execution Manager

Detailed Continuous Integration for System z Scenario



1. Check-in code
2. Build code and zUnit tests
3. Deploy build results to Test Environment
4. Execute zUnit Tests

5. Kick-off Automated Test Plan

8. Report test results in dashboard/build results/defect records in CI server.

6. Run automated interface tests against Test Environment
7. Mark execution records Pass/Fail in Test Execution Manager

Building and Evolving automated test execution

- Look at what is possible based on current codebase
- Start small and quick, build to more complicated test setups
- Validate core functionality first
- Progress to edge cases and error conditions
- Refactor codebase to improve testability

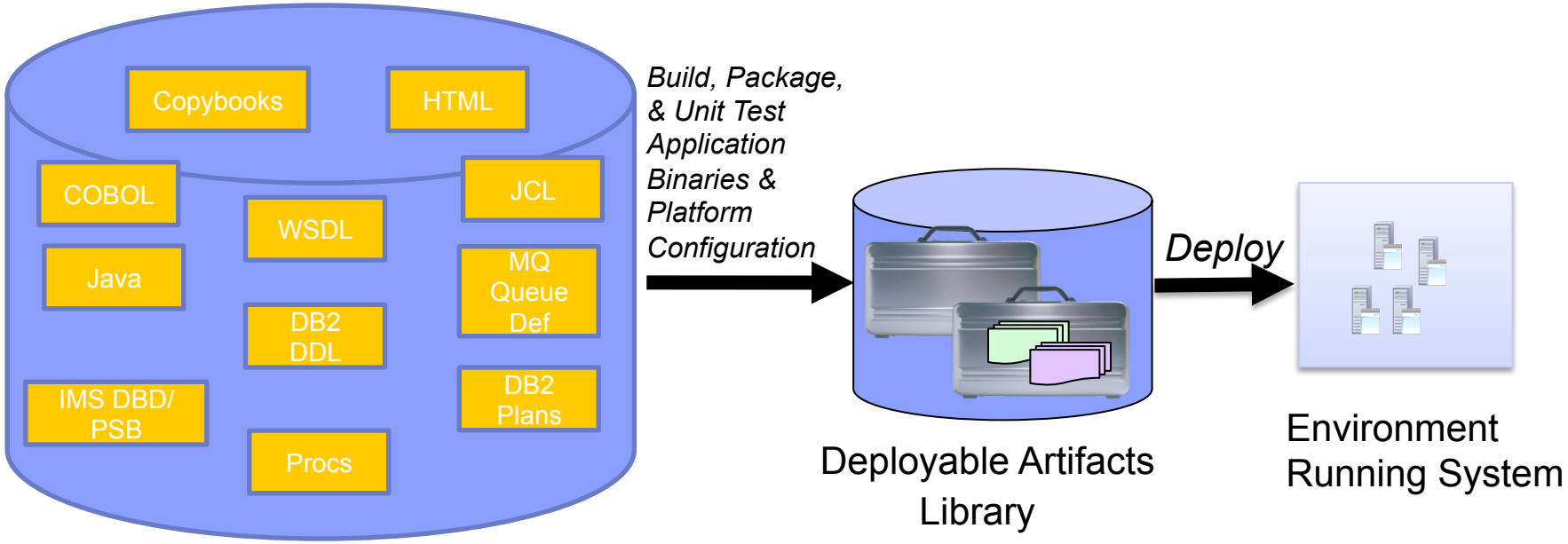
Test phase	Test Type	Focus	Rational Tool
1	Unit	Module	zUnit (RDz)
2	Interface	APIs, Service interfaces, External calls	RTW or RIT
3	Functional	User interface	RTW or RFT
4	Performance	Response time from system	RTW or RPT
5	Exploratory	Manual testing (“poking around”)	RQM
6	Acceptance	User validation / pre-production	RQM

DevOps: Version everything, package everything, define a pipeline

Enterprise Applications have thousands of disparate parts

- Currently maintained in separate systems
- Limited linkage between systems for application dependencies
- Missing assets and information is rampant

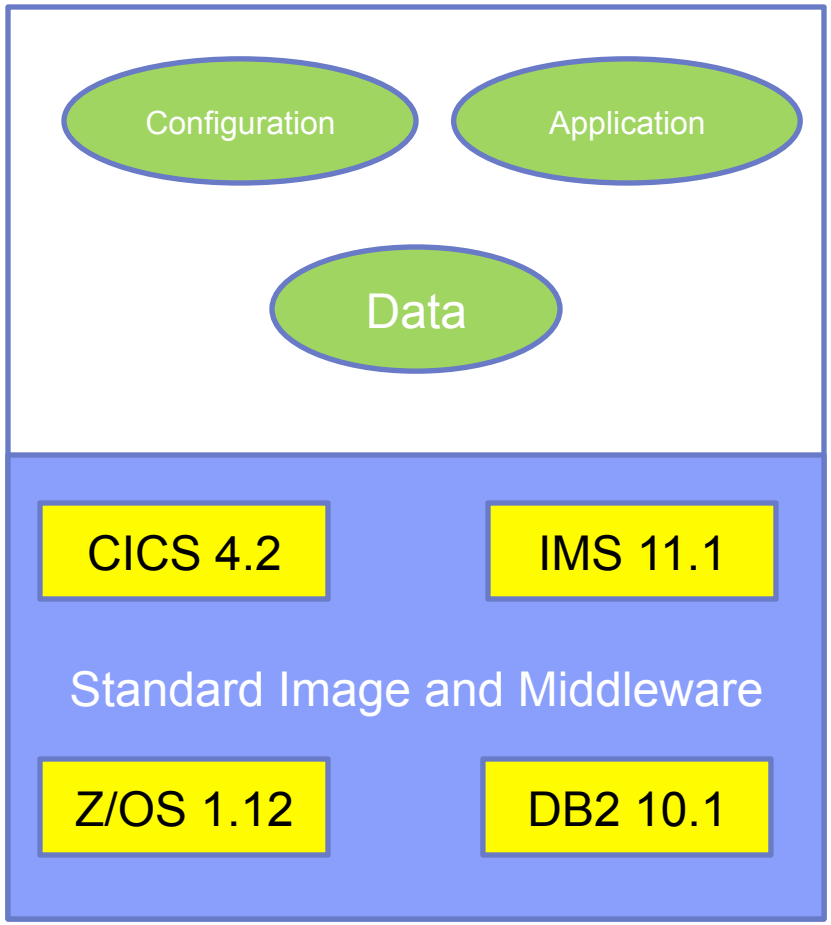
DevOps practices force linkages, automation, and standardized packaging…



Moving toward the future

Completing the DevOps solution…

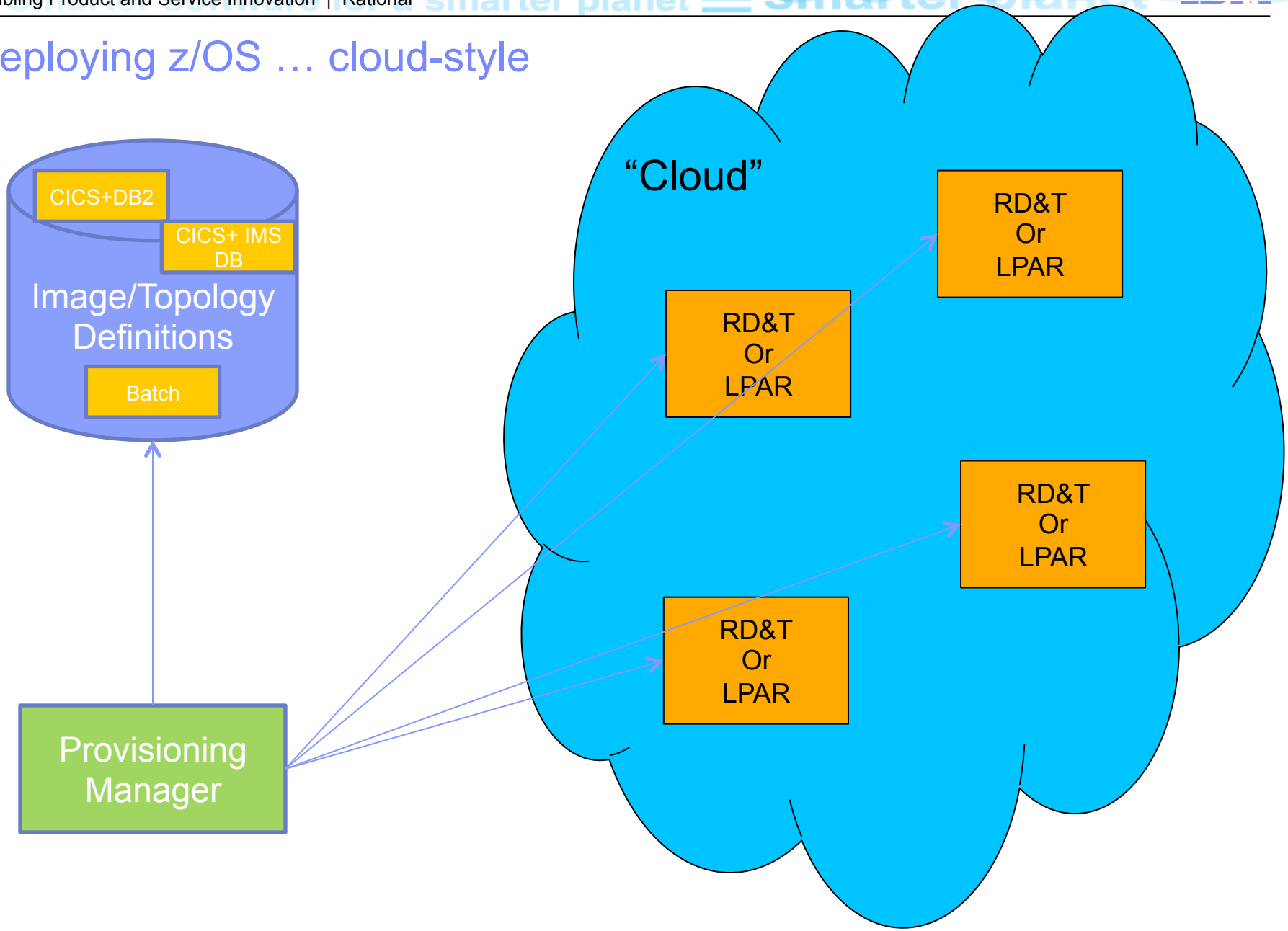
Standardize z/OS and distributed images to ease deployment



- Standard topologies exist today (production LPARs)
- Standardized images few and far between
- Standardized/Automated deployment of COMPLETE system is spotty

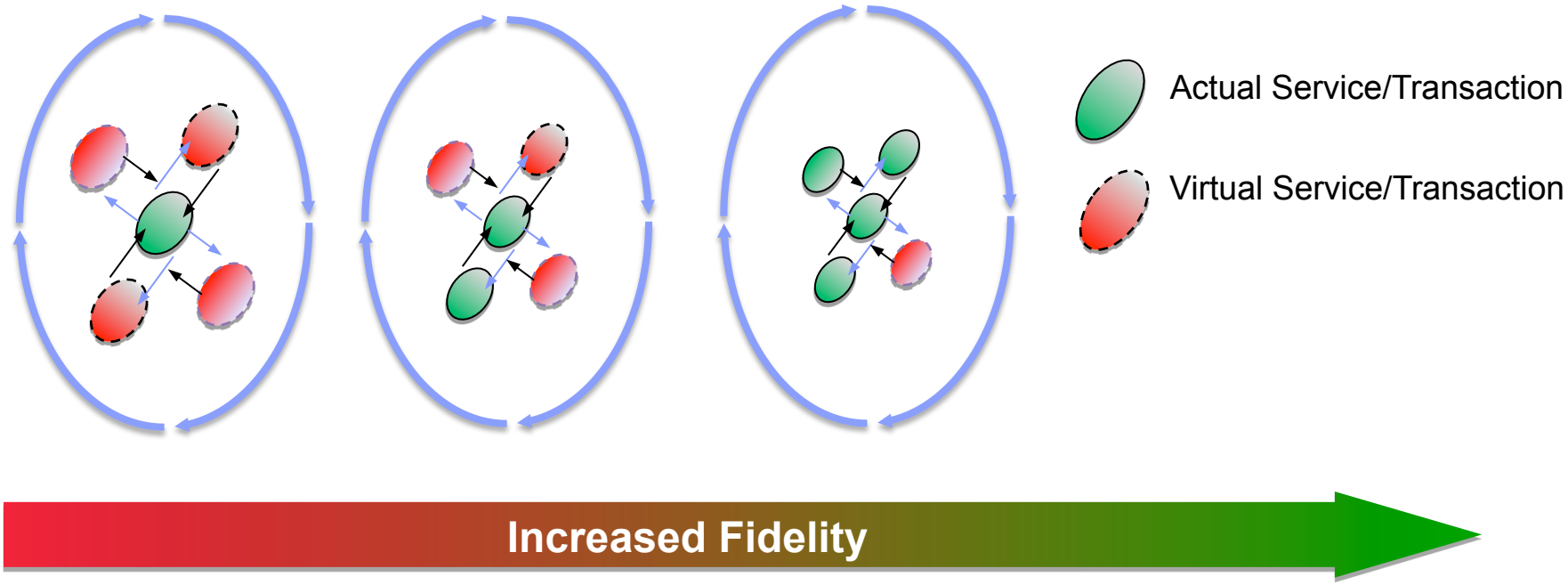
Standardize and automate provisioning of everything …

Deploying z/OS … cloud-style



Focusing testing via dependency virtualization (using Green Hat)

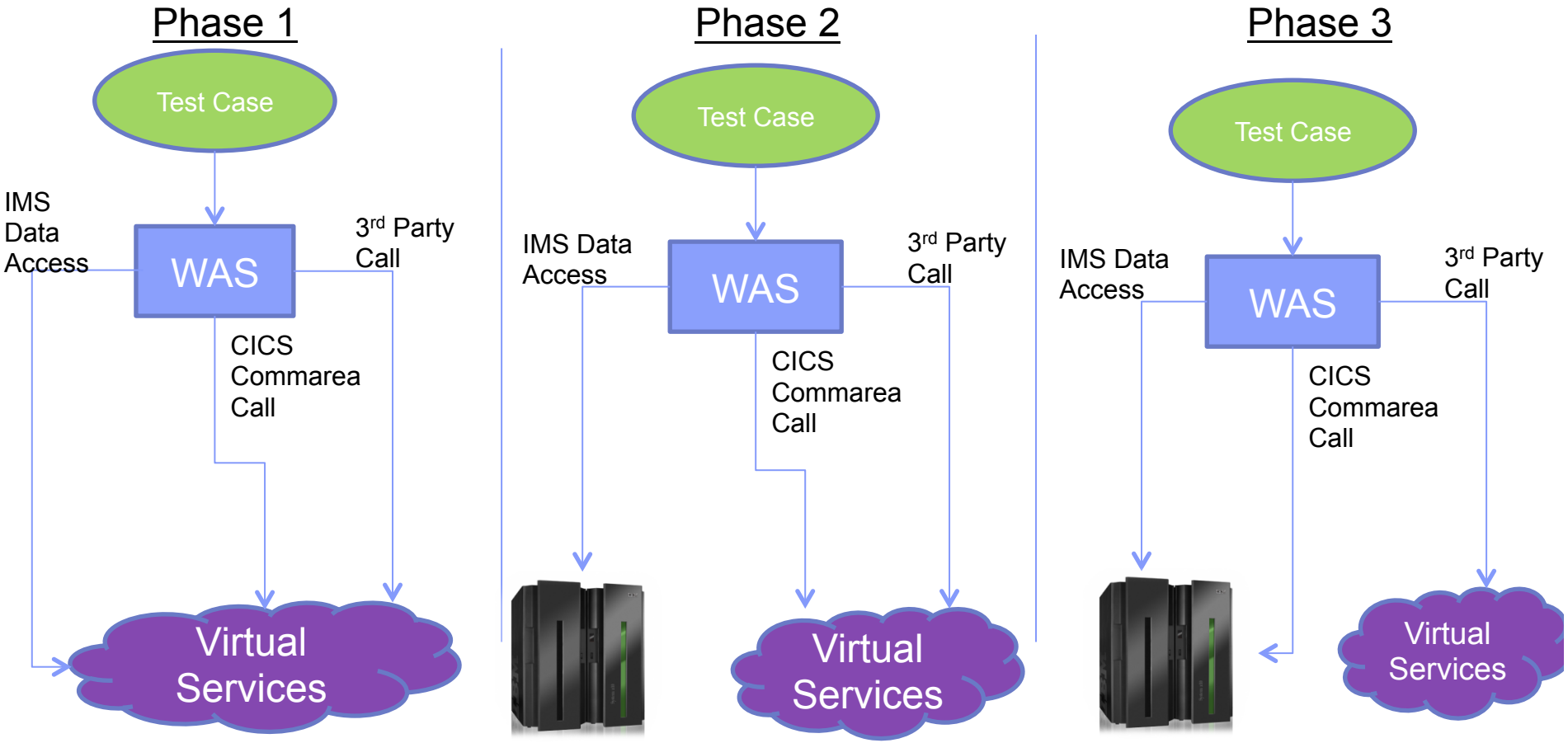
Integration test complex systems



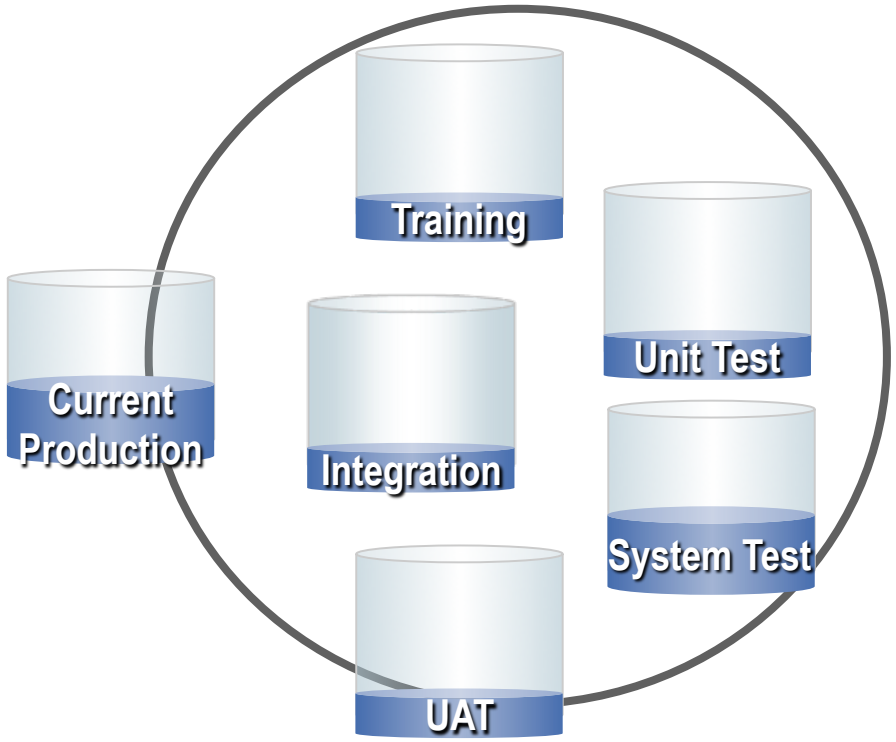
Pipeline testing with dependency virtualization

Controlled large system testing by isolating components under test

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



Test Data Management scales down testing to the essentials



Production	200GB
Training	25GB
Unit Test	25GB
System Test	200GB
UAT	25GB
Integration	25GB
Total	500GB

Infrastructure reduced by 83%

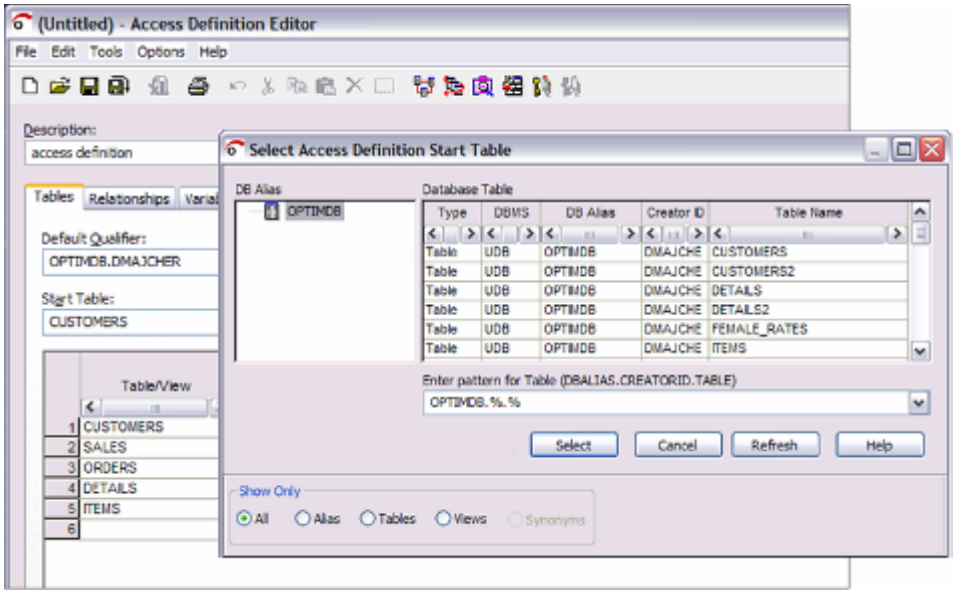
- Create targeted, “right-sized” subsets faster and more efficiently than cloning
- Compare to pinpoint and resolve application defects faster
- Improve development efficiencies

Creating right-sized targeted test environments saves storage costs & speeds testing

Optim Test Data Management Solution

Test Data Management

Create "right-size" production-like environments for application testing



Requirements

- Create referentially intact, "right-sized" test databases
- Automate test result comparisons to identify hidden errors
- Shorten iterative testing cycles and accelerate time to market

Benefits

- Deploy new functionality more quickly and with improved quality
- Easily refresh & maintain test environments
- Reduce storage and operational costs

Where should I start?

1. Identify a well contained project for adoption
2. Define infrastructure code for platforms and the project application
3. Define automated tests for the infrastructure and application, including conditioned data for testing
4. Adopt a single-stage continuous delivery process to support continuous build, deploy, and test for dev and test virtual environments
5. Inject monitoring as part of the standard pattern and use the data in the delivery process to improve feedback
6. Adopt a multi-stage delivery process that supports promotion of changes from one stage to the next (e.g., Dev to QAT).
7. Adopt a delivery process with promotion to production
8. Track and manage incidents in production linked to work/tasks in development

www.ibm.com/software/rational

www.ibm.com/software/rational