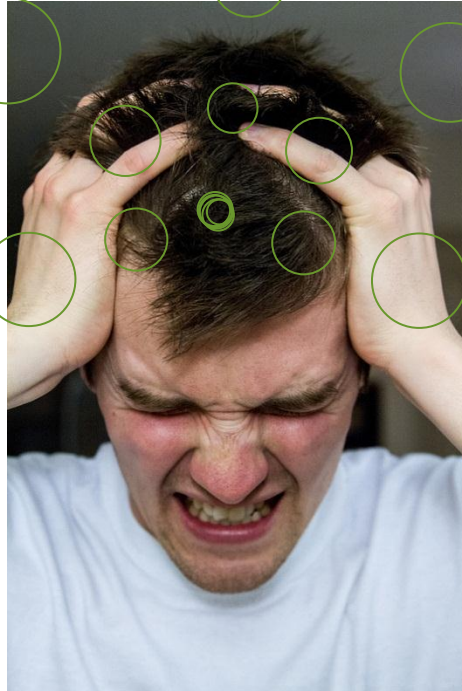# The DevOps Approach: System z Deployment Automation with UrbanCode

DevOps | IBM

Common deployment problems

Mistakes

Uncertainty

Confusion

Disconnects

Rework
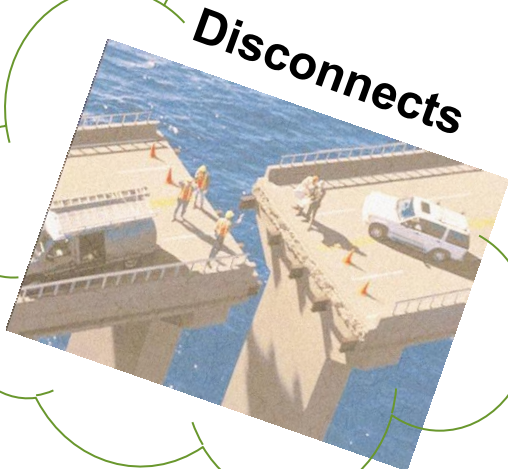
What's going wrong?

| Failures due to differences in dev and ops environments	Time to market pressure for more frequent releases	Manual (tribal) processes for release lack repeatability/speed	Major releases take days, 100 people and are managed by a spreadsheet
Dev Prod	Daily Build Release	Who did this last time? Dave… Dave's not here man…	I'll order breakfast

The Ideal of Service Delivery
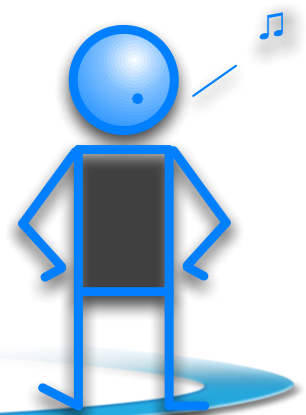
Line-of-business
"Need a feature"
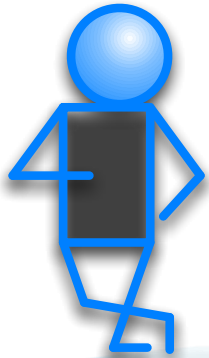
Development
"Create a feature"
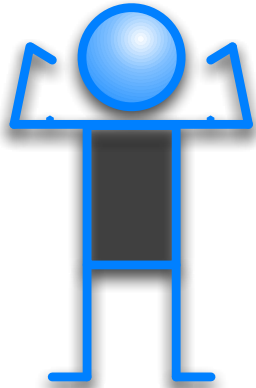
Operations
"Manage a feature"
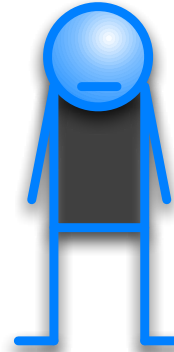
Customer
"Awesome feature"

The Reality of Service Delivery
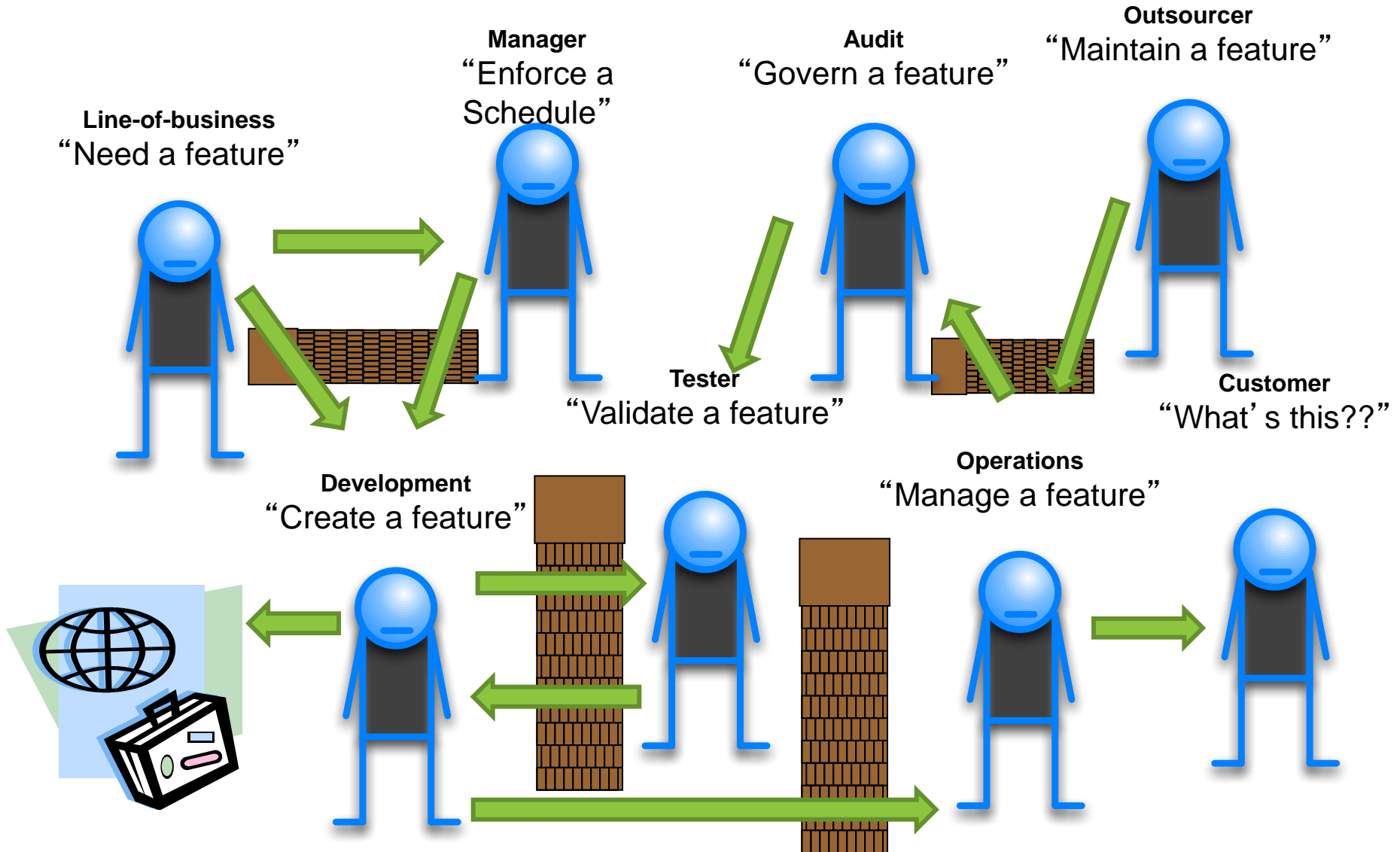
Large Releases Increase Risk

Service	Build	Func Test	Sys Test	Stage	Prod

A

B

C

D

E
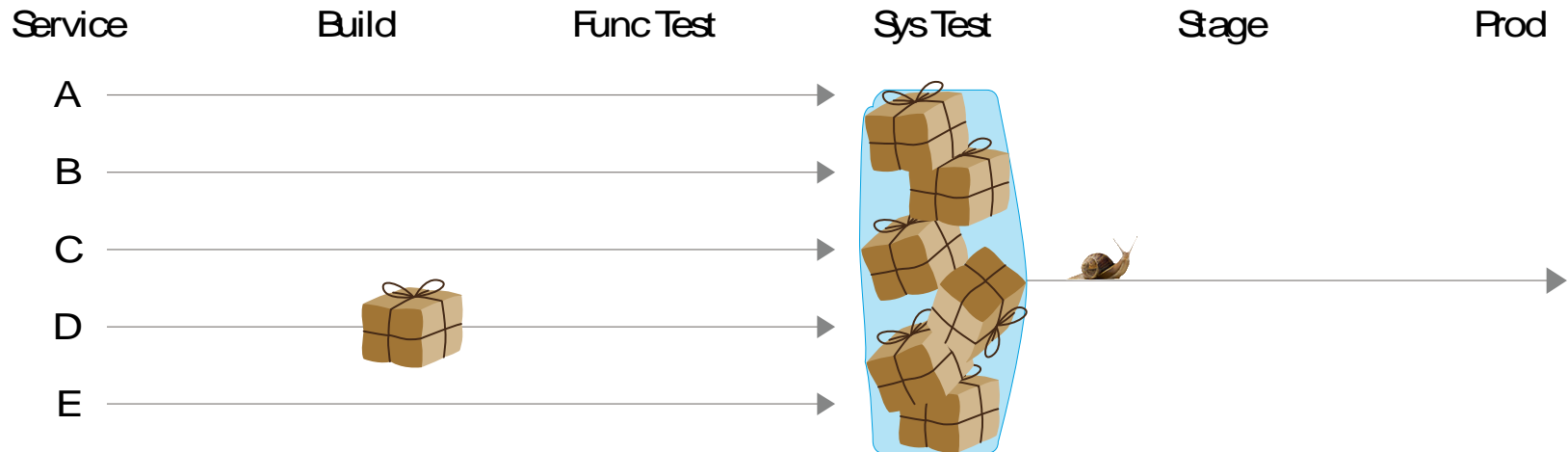
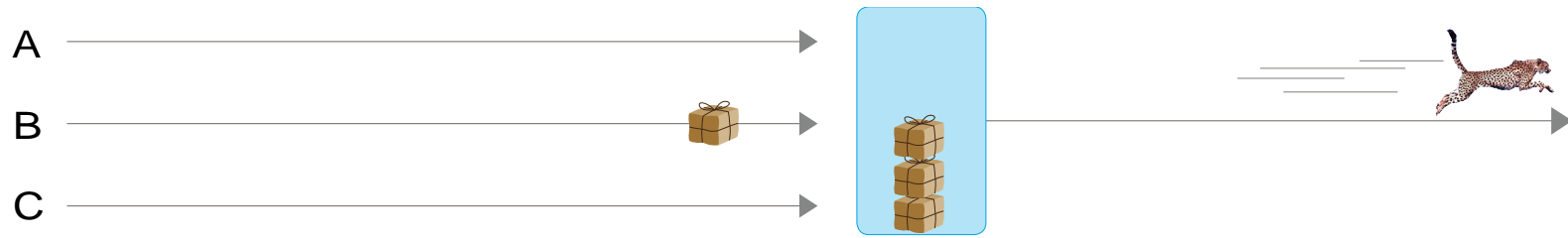## Large Batch Sizes

- Services: Many

- Dependencies: Many

- Changes: Many

- Complexity: Large

- Impact of Failure: Large

- Failure Analysis: Difficult
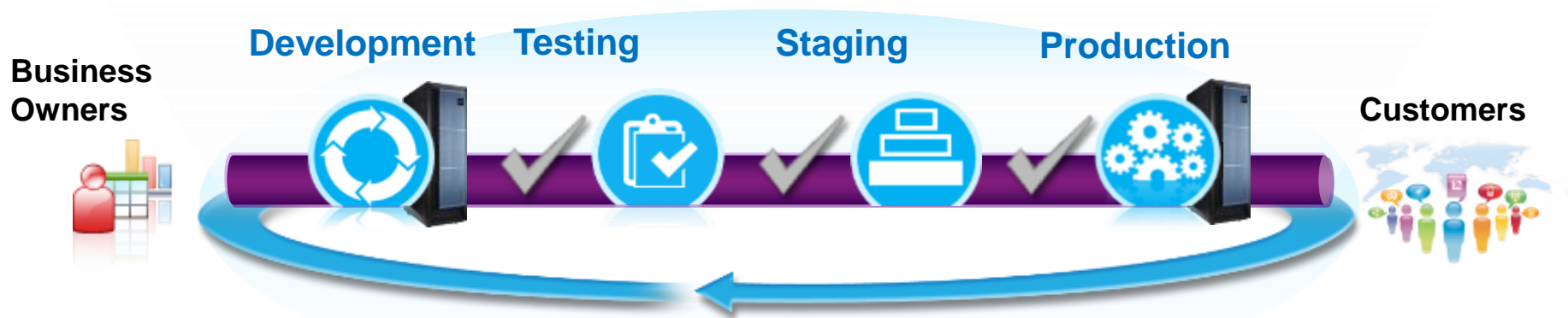
Scope and size reduction reduces risk

A ———————→

B ——————📦—→

C ———————→

🐆

Small Batch Size

- Services: Few

- Dependencies: Few

- Changes: Few

- Complexity: Small

- Impact of Failure: Small

- Failure Analysis: Simple

RISK

Solution: A Continuous Delivery Pipeline

Business Owners

Development **Testing** **Staging** **Production**

Customers

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

Investing in Release Automation … Introducing UrbanCode

Enabling clients to more rapidly deliver mobile, cloud, big data and traditional applications with high quality and low risk

Drive down cost

Remove manual effort and wasted resource time with push button deployment processes

Speed time to market

Simple, graphical process designer, with built-in actions to quickly create deployment automation

Reduce risk

Robust configuration management, coordinated release processes, audits, and traceability

IBM UrbanCode Deploy orchestrates and 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.

IBM UrbanCode Release is an intelligent collaboration release management solution that replaces error-prone manual spreadsheets and streamlines release activities for application and infrastructure changes.

Architecture

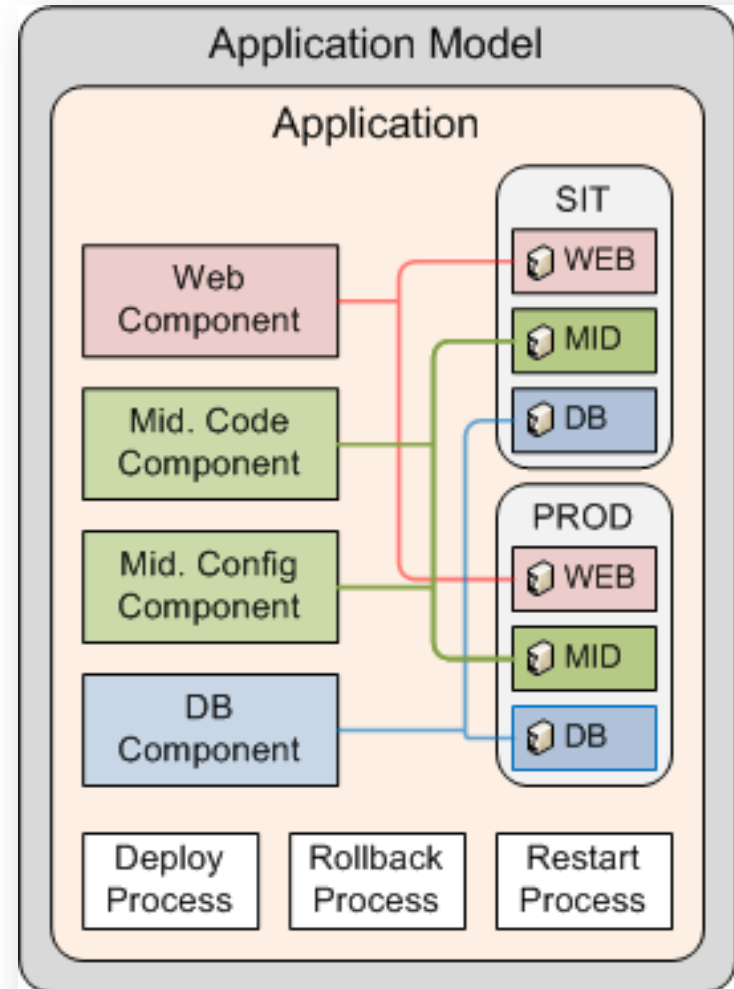# UrbanCode Deploy manages deployment of applications

Components

- Tiers or services

Environments

- Target servers
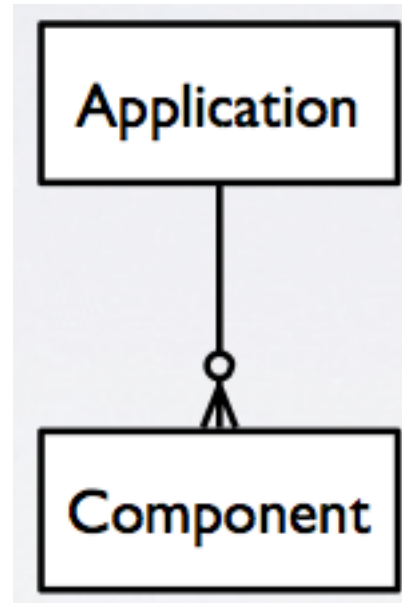- Environment specific configurations

Processes

- Coordinates component level processes
- Same processes used across environments

The Basics

■ Application

– A logical grouping of deployables

– Single unit of management

– Self-contained definitions of automation, target environments

– Create Snapshots for "good" sets of versioned Components

• Component

– Files or configuration to be deployed, steps to deploy them

– Generally mapped to something in an external system

– **Components are versioned and associated with artifacts**

– May contain binaries or scripts

Understanding the World

- Understand computing objects as "Resources"

- Resources map to Agents

- Every OS has an agent, and each agent corresponds to a resource

- Child resources can be created to represent logical containers/spaces

- Components are mapped onto Resources
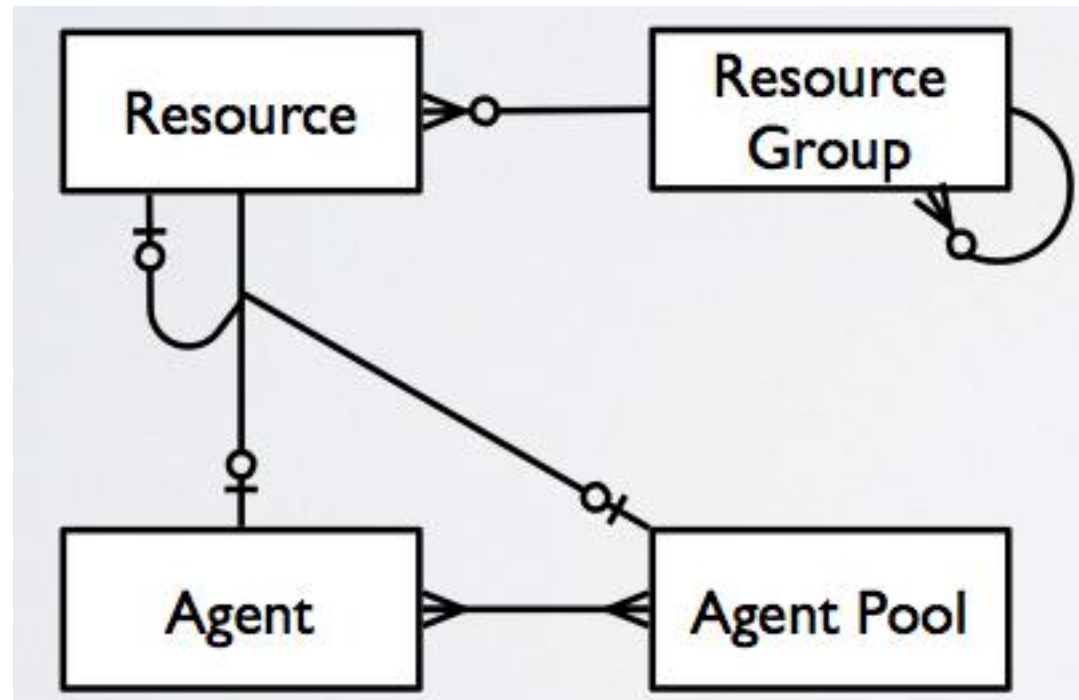
Bringing it together: Environment (v4.x)

- Combines resources into a single logical deployment target

- Specifies 0+ resources or groups for each component

- Good place for environment configuration (database URL, etc)
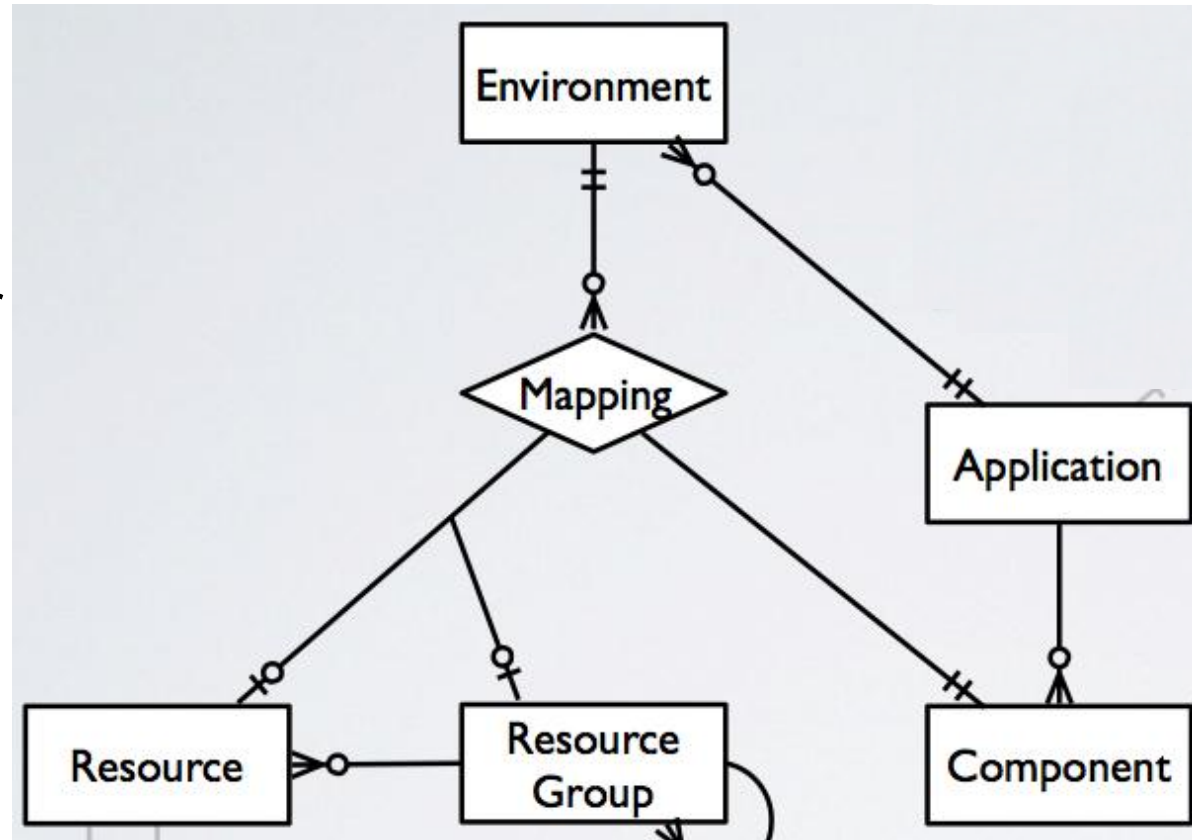
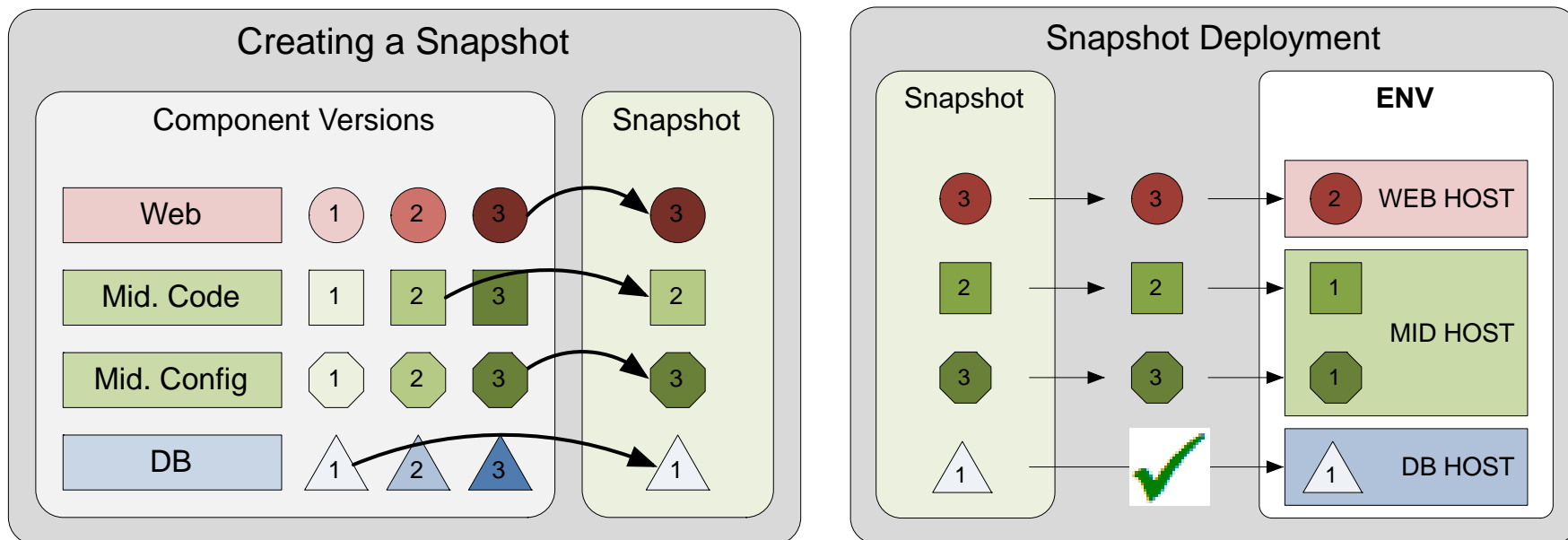# Understanding Environments in UrbanCode Deploy

- UrbanCode Deploy v6.0 introduces a global Resource hierarchy

- Resources can be organized into groups for your Lines of Business, Applications, Environment classes (Dev, QA, Cert, Prod, etc)

- Component Resources exist in the tree and Component versions are matched to the appropriate resource when deployed

| | Resources | Resource Templates | Agents | Agent Pools | Cloud Connections |

Create Top-Level Group   **Select All...** ▾   **Actions...** ▾

| | | Name | | Inventory | Status |

Entire resource tree is available, organized into top-level groups

Agent resources represent running agents on live hosts

▾ 📁 US-EAST
  ▾ 📁 Development
    ▾ 📁 eCommerce
      ▾ 📁 JKE
        ▾ 🖥 fit-vm13-219.rtp.raleigh.ibm.com (View Agent) — Online
          🐞 jke.db   20130910_171755
          🐞 jke.tomcat   20130910_171755
          🐞 jke.war   20130910_171755
        ▸ 🖥 fit-vm13-76.rtp.raleigh.ibm.com (View — Online
  ▾ 📁 Sandboxes
    ▾ 📁 eCommerce
      ▾ 📁 JKE
        ▾ 📁 GSS4-SANDBOX-1
          ▾ 🖥 os_part (View Agent) — Online
            ▾ 🐞 jke.tomcat
              🐞 jke.war
          ▸ 🖥 os_part_0 (View Agent) — Online
  ▸ 📁 System Test

Component resources represent where Components are deployed

Multiple application components can be placed on the same node in the resource tree

© 2013 IBM Corporation

Application versioning is critical

- Version components, configurations, and processes
- Manage a set of tested component versions as a single unit
- Promote snapshots between environments
- Ensures higher quality by guaranteeing the correct versions of related components are deployed together

# Mainframe Deployment Automation with IBM UrbanCode Deploy
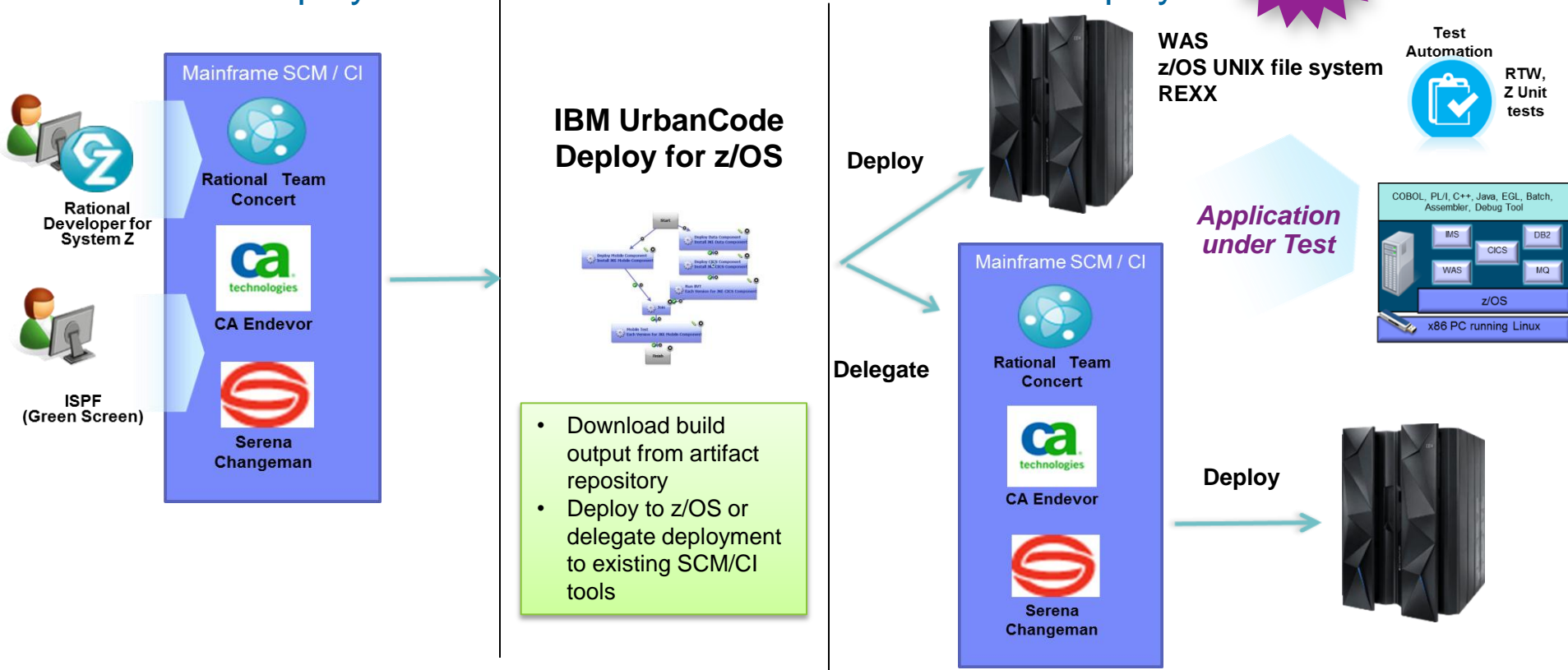
- Support for UrbanCode Deploy Server and Agent on zLinux – *Available in v6.0*
- Initial deployment automation support of applications on z/OS – *Available in v6.0.1*
    - z/OS UCD agent will run deployment activities directly (e.g. file operations, deploying to application servers, running shell scripts including REXX)
    - z/OS UCD agent interacts directly with an existing mainframe deployment tools to delegate deployment and promotion

UrbanCode Release
Release planning and coordination

- Plan the release day

- Execute the release

- Communicate what's going on

- Allocate environments to releases
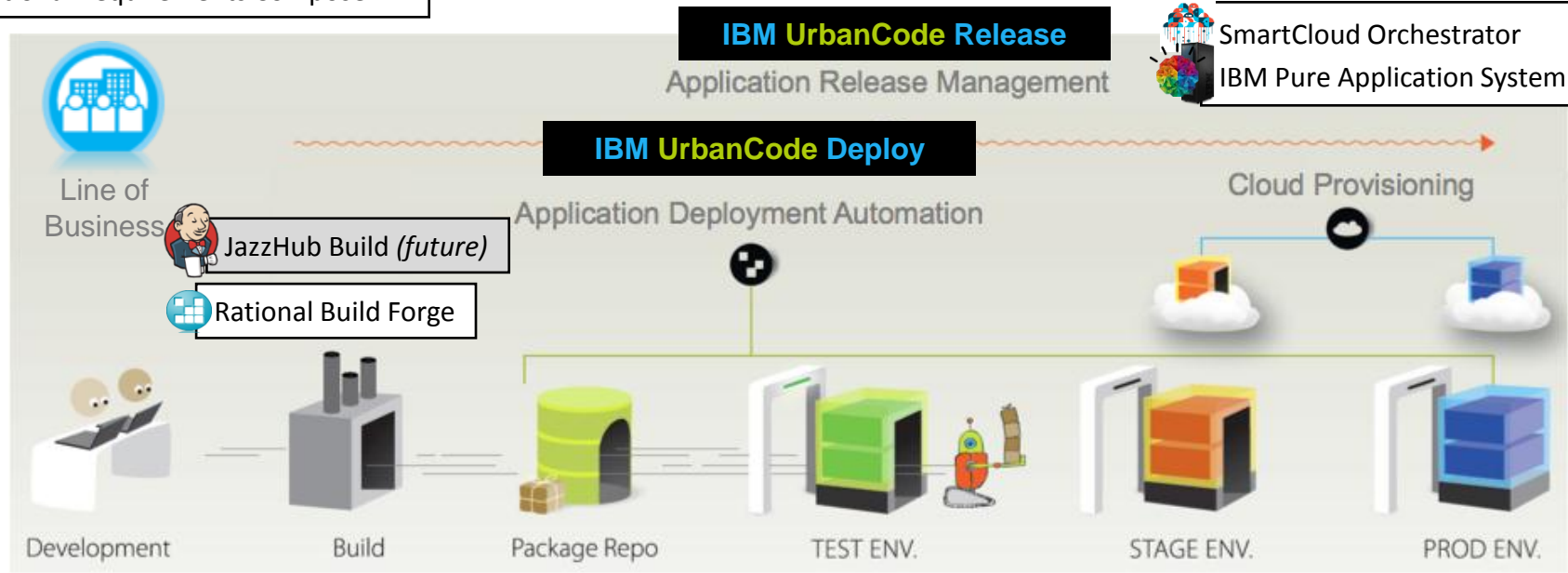
- Tie release back to development
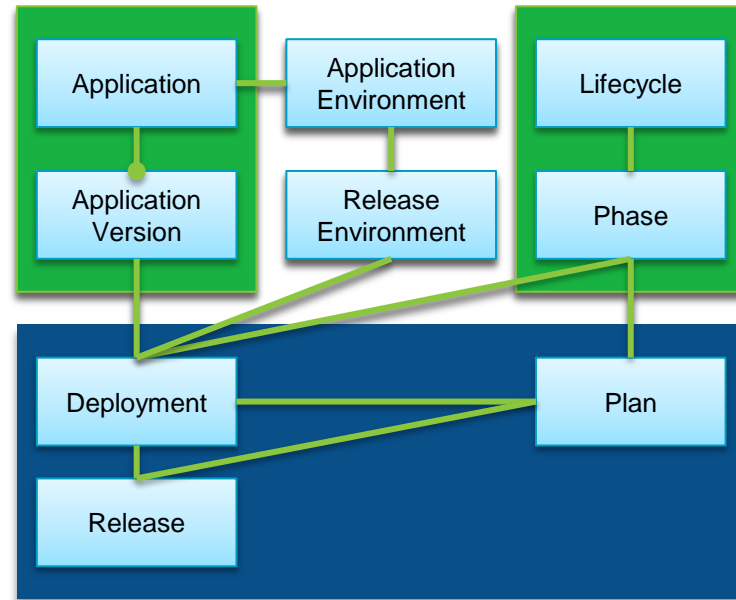
Delivery Pipeline

Rational Focal Point
Rational Requirements Composer

UrbanCode Environment
Development Toolkit *(future)*

IBM UrbanCode Release

Application Release Management

SmartCloud Orchestrator

IBM Pure Application System

Line of
Business

IBM UrbanCode Deploy

Cloud Provisioning

Application Deployment Automation

JazzHub Build *(future)*

Rational Build Forge

Development   Build   Package Repo   TEST ENV.   STAGE ENV.   PROD ENV.

Rational Developer for System z
Rational Team Concert

Rational Quality Manager
Rational Development and Test
    Environment
Rational Test Workbench
Rational Test Virtualization Server

InfoSphere Optim Test Data Management

SmartCloud Application
Performance
Management

tealeaf

Deploy and Release Positioning

	Deploy	Release
Application Deployment Automation	✓	
Automate deployment of applications and components	✓	
Visual process designer	✓	
Plug-in steps for application containers, web servers, network devices, database deployment, etc.	✓	
Model deployment targets with properties.	✓	
Release Management		✓
Orchestrate changes across multiple applications.		✓
Coordinate changes outside the scope of application deployment automation.		✓
Real-time status for long running production deployments.		✓
Visibility and control of quality process.		✓

UrbanCode solutions deliver measureable business results

International Investment Firm
Driving Down Costs

Release process required considerable effort and delayed by manually-introduced errors

- Solution: Automated release process

- Results: Cost avoidance of over $2.3M/year, reduced release time from 2-3 days to 1-2 hours and virtually eliminated test team down-time

Online Retailer
Speeding Time-to-Market

Significant delays getting application changes to production

- Solution: Scaled up continuous deployment

- Results: Deployment time reduced by over 95% with easy scale and deploying to over 250 servers within 2 months of implementation

Higher Education
Speeding Time-to-Market

Agile development teams constrained by slow deployment to dev and test environments

- Solution: Accelerate deployment by enabling development teams to self deploy with automation

- Results: Deployments cut from hours to minutes and a greater number of servers with fewer resources

SaaS Software Provider
Reducing Risk

Difficulty managing multiple customer configurations and versions of software deployed across servers

- Solution: Automate managing configuration and version deployment

- Results: Execute customer specific releases, reduced deployment outages by over 90%

www.ibm.com/software