IBM Software

Innovate2012

The Premier Event for Software and Systems Innovation

Next NOW!

DevOps: Creating Value Thought Infrastructure Service Delivery at Aetna

Daniel Berg – IBM

Chief Architect for DevOps – Continuous Delivery

aetna SM

Quality health plans & benefits
Healthier living
Financial well-being
Intelligent solutions

Aetna, Inc.

Aetna (NYSE: AET) is one of the nation's leaders in health care, dental, pharmacy, group life, and disability insurance, and employee benefits. Dedicated to helping people achieve health and financial security, Aetna puts information and helpful resources to work for its members to help them make better-informed decisions about their health care.

Membership:

18.459 million medical members
13.670 million dental members
8.820 million pharmacy members

Health Care Networks:

More than 1 million health care professionals
More than 575,000 primary care doctors and specialists
More than 5,400 hospitals
A network of specialist physicians, recognized with Aexcel® designation, based on clinical performance and cost efficiency

Products and Programs:

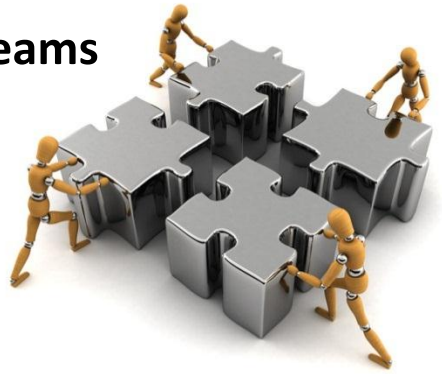
- Aetna offers a broad range of insurance and employee benefits products.
- The first national, full-service health insurer to offer a consumer-directed health plan, Aetna continues to lead the way with its Aetna HealthFund line of products, including HSA, HRA and RRA options.
- Aetna offers a wide array of programs and services that help control rising employee benefits costs while striving to improve the quality of health care, such as case management; disease management and patient safety programs; integrated medical, dental, pharmaceutical, behavioral health and disability information.
- Aetna provides members with access to convenient tools and easy-to-understand information that can help them make better-informed decisions about their health and financial wellbeing.

Agenda

1. Overview
2. Path to Maturity
3. Collaboration
4. Standardization
5. Automation

Growing Business Challenges

Fractured Relations Between Development and Operations Teams



- Lack of Trust between groups
- Conflicting Standards
- Poor communications
- Limited Quality Measurements

Rapid Solution Delivery

- Increased demand to deliver business value rapidly at lower cost while maintaining (or increasing) quality

Broadly Integrated Systems

- Solutions must span technologies, organizations and companies which can lead to an increase of one-off environments
- Solutions can no longer be developed in isolation.
- New technologies introduce new programming models and integration challenges. Standards and ownership models must reflect desired usage patterns to generate value

Subject Matter Expertise Limitations

- Proprietary expertise not documented or accessible to broad audience
- Brain Drain through staff reductions and retirements
- Random or non-existent processes and informal coordination between specialists

Compounded By Operational Issues

Complex - Not Customer Focused

- Service offerings not intuitive or aligned with customer needs
- Customers are required to know the unique organizational processes to request services, the order in which they must be requested and detailed infrastructure implementation details

Disparate Request Mechanisms

- Customers are required to know the unique processes to request services, the proper order these requests must be made, the format of these requests and detailed infrastructure integration requirements

Minimal Workflow Automation and SLA Management

- High levels of manual administration by Subject Matter Experts in many areas with little to no feedback loop
- Limited, cross-organizational coordination for delivery of infrastructure services and runtime stack

Highly Manual Governance

- Limited comprehensive standards make it difficult to ensure consistent, reproducible delivery and maintenance

Lack of Commitment to Develop an Integrated Strategy to Deliver Infrastructure Services

- Each IT operational organization had taken it upon themselves to invent their own solution to manage services
- Until very recently this was not seen as critical to our ability to survive

Vision

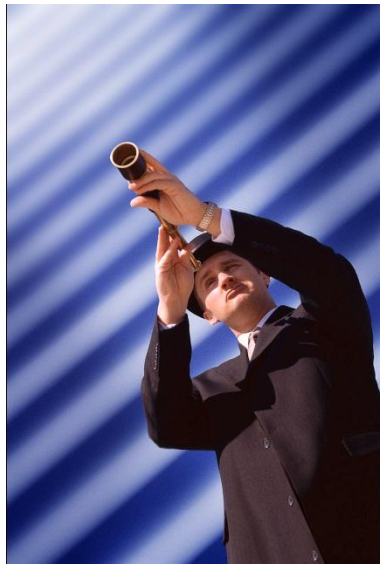
Increase Customer Satisfaction

Simplify interaction with IT operational areas and increase customer autonomy

Centralize request management through common portal and interfaces

Increase transparency with standard status reporting and traceability of requested services

Align services with customer needs



Increase Productivity and Value

Improve Speed to Market: Formalize and Automate Service Delivery

Re-allocate skilled SME's to focus on higher-value activities

Develop reusable methodology for creating and deploying IT operational services

Optimize Infrastructure and Application delivery lifecycles

Standardize Cloud service offerings to align with service delivery best practices

Improved Governance

Standardize delivery pipelines and policy validation and reporting (a pre-requisite for automated services)

Increase consistency of infrastructure implementations

Increased audit-ability (exactly who, what, where, when, how much)

DevOps Context Example

Patterns

Patterns form the basis for both Dev and Ops

Pipelines

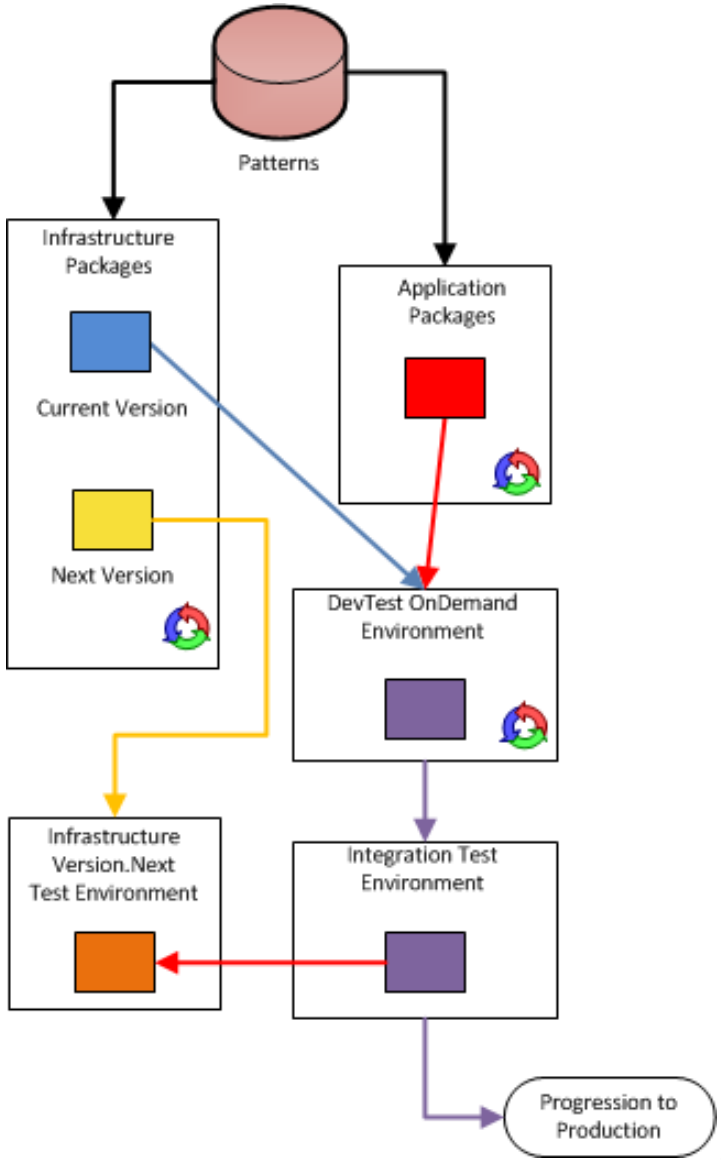
Standardized pipelines for delivery of infrastructure and application packages

Iterative Development

Good iterative development and testing practices for Development and Operations teams

Automation

Builds, delivery, testing automated wherever possible



How

Sponsorship

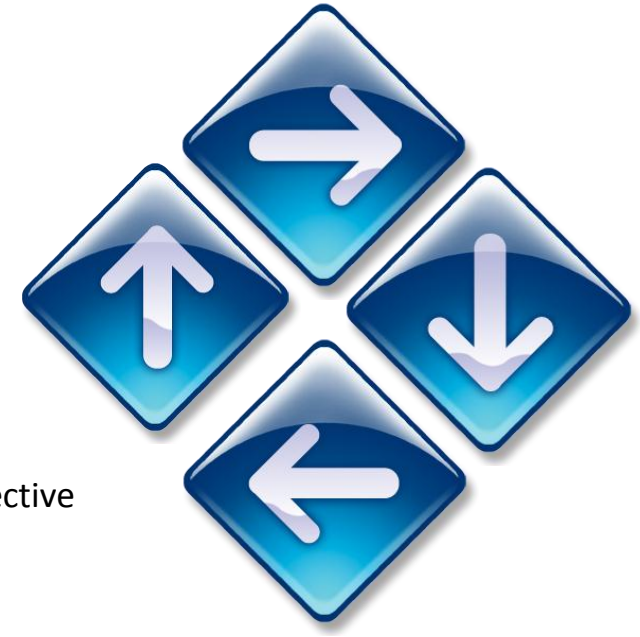
- Senior Executive support and backing

Collaborate

- Improve Operations and Development Collaboration

Standardize

- Common Language / Tools
- Formalize delivery processes of IT solutions to the business



Identify

- Determine IT operational service offerings from a customer perspective
- Decompose into domain specific service offerings

Automate

- Infrastructure as Code

Deliver

- Develop self-service delivery mechanisms for service request and execution
- Cross-domain Service Orchestration

Manage

- Create governance mechanisms
- Monitoring, metering, notification
- Incidence management

Service Management Maturity

Sponsorship

Collaborate

Senior Executive Sponsorship
Increased Ops and Dev Collaboration
Common collaboration tools

Increased Customer Collaboration

Standardize

Common Tools and Language
Formalized/ Documented processes
Identify customer-centric service opportunities

Formalize Tools and Processes

Automate

Create Well-structure Infrastructure as Code
Source code management
Continuous Unit and Integration Testing

Infrastructure As Code

Deliver

Autonomous Service Execution
Service Catalog
Composite Service Orchestration

Self-Service Execution

Manage

Service Lifecycle Management
Service and Process Reporting
Integrated Solution Delivery
Incident Management

Comprehensive Lifecycle Management

Value

Alignment with Customer Needs

Basis for Reuse
Comprehensive view of IT business

Rapid service delivery
Re-allocate SME resources
Service Execution

On Demand Delivery
Self-Service
Solution Delivery

Rapid Delivery
Metered Services
Utilization Reporting

Collaboration

Culture

Culture eats strategy for breakfast

- There are a million reasons to maintain the status quo
- If you don't address the core ones and get buy-in, you will fail

Collaboration is Key

- One of the more difficult tasks is identifying the services the customer needs
- Assuming you know is not the same thing as asking them what they need or want
- Your customer will answer in their terminology, not yours.

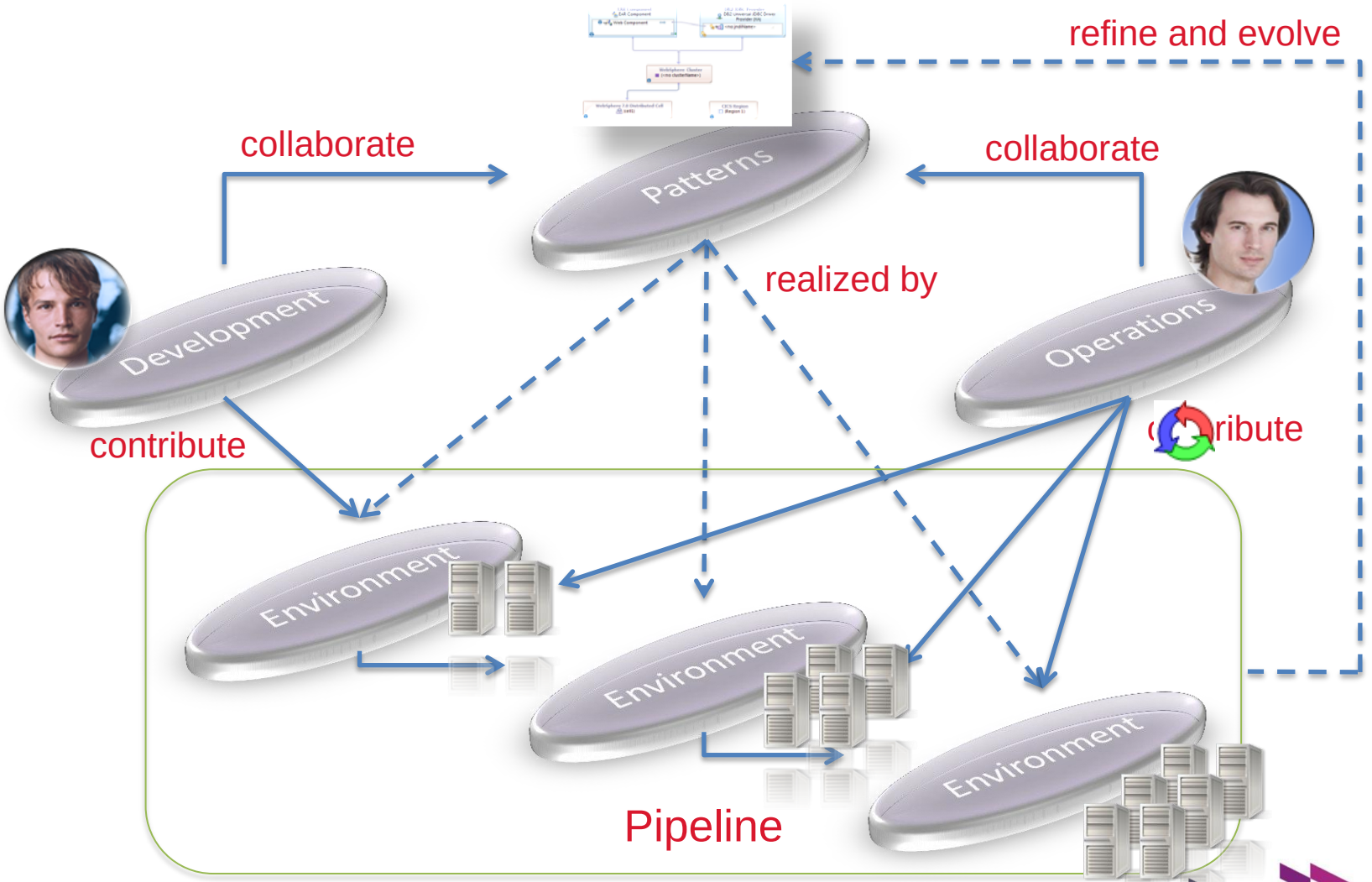
Requires treating IT Operations holistically

- Traditionally very silo'd with very little interaction between operational teams
- Rather than "What do I do?", the question is more involved: "What does the customer need, how can be standardize that need and how should we deliver it?"
- DevOps is focused on the service delivery approach as well as the content

Infrastructure Service Delivery

- Formalization of processes, topologies, patterns
- Incorporated degree of agility to allow it to evolve

DevOps Collaboration Framework



Topology Patterns

Patterns are one of the core mechanisms for Operations and Development teams to reach a **common understanding for delivery** of solutions

Pattern creation is a **collaborative effort** between stakeholders and encourages standardization by defining reusable architectures that adhere to the governance policies

Reduce cost through consistent implementation and administration, problem determination and maintenance

Two Types:

- Logical Design Patterns: More generalized - Show relationships, constraints for common scenarios
- Physical Patterns: More detailed – Defines physical, vetted details for component usage

Implementation:

- **Communicate:** Get buy-in between stakeholders
- **Collaborate:** Setup pattern workgroups to develop, collaborate and refine patterns
- **Share:** Establish a catalog of standard, supported patterns

Avoid:

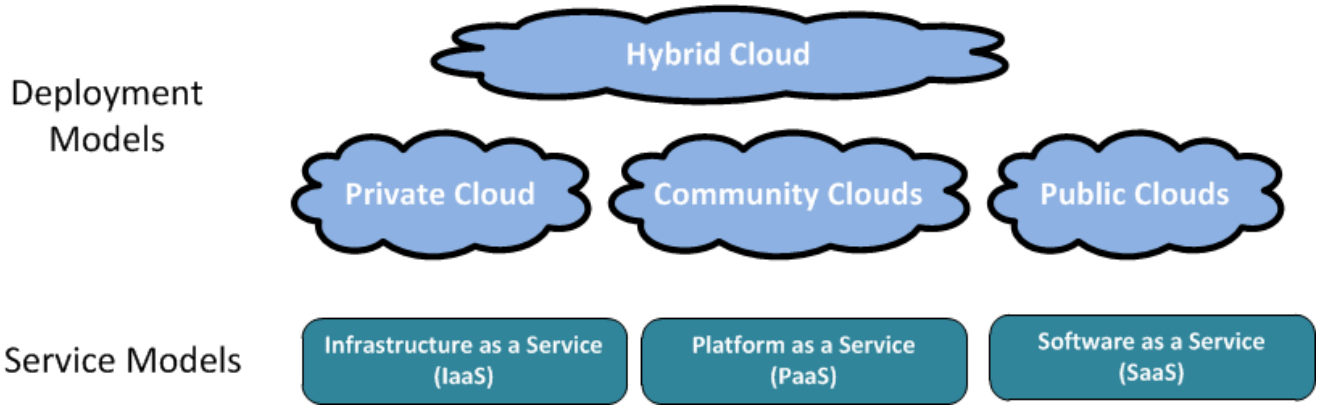
- Lack of consensus on architectural patterns
- One size fits all (doesn't fit anyone well) or Custom Everything (increased cost to manage and maintain)

Standardization

Standardization Focus Areas

Service Description

- What constitutes a service?
- Dependent upon whether services are domain specific or exposed as an enterprise service to the customer
- Service delivery not limited to Cloud, but provides a good context for discussion



Service Lifecycles and Pipelines

- Describes the processes used for the creation, deployment, maintenance and retirement of services and their actions

Solution Lifecycles

Represent the progressive states solutions traverse and the relationships and dependencies on other activities

Defines how new solutions are created, deployed, maintained and retired

Importance:

- Communicate the macro flows involved in the delivery of a solution, ensuring stakeholders are in agreement
- Provide means to measure the operational efficiency
- Consolidate consistent, reproducible actions for reuse
- Allows processes to evolve over time

Lifecycle creation is a collaborative effort between the stakeholders

- Create well defined processes and hand-offs
- Define consistent logical architectures for each application and environment
- Define physical architectures for each environment which supports the logical architecture
- Ensure that progress can be measured and reported

Lifecycles

Service Creation

- Development lifecycle of a service
- Modeling thru Publish to Deprecate

Service Request

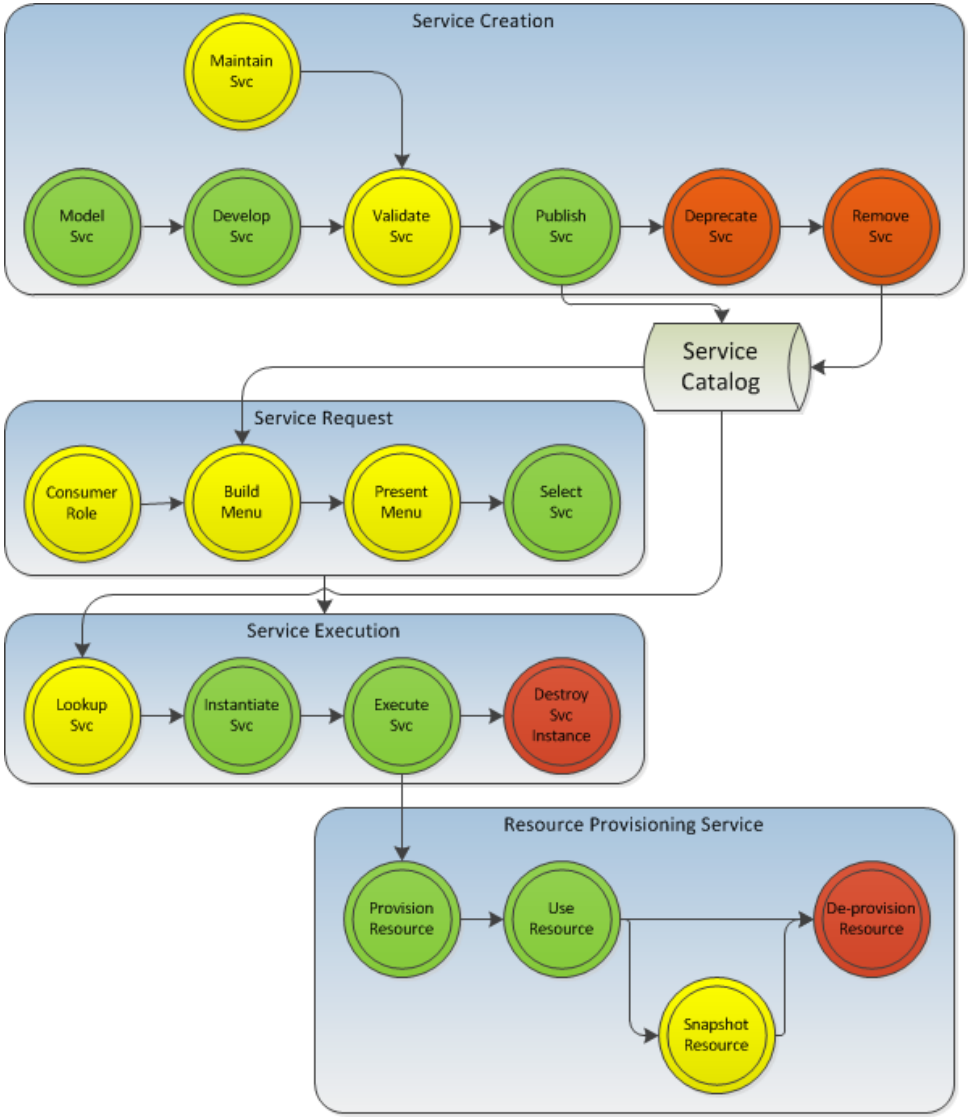
- Presentation of a service catalog to the customer and associated validations

Service Execution

- Lifecycle of a service instance

Resource provisioning

- Cradle to grave management of the instance



Pipelines

There are many ways to deliver enterprise assets (code and technology) through the testing stages and into a production environment

Delivery pipelines formalize an end-to-end process to provide common and consistent mechanisms to manage asset migrations

- Each stage provides a validation feedback loop of the application along some axis (functionality, performance, user acceptance)
- Each stage should define proper entry and exit criteria and the governance processes around when to promote
- Artifacts should be managed in a well known repository (more on that later)

Implementation:

- Well defined processes and hand-offs
- Common, automated mechanisms to ensure consistent build, test and promotion.
- Consistent flow from Unit Test through Production
- Well defined interfaces for interaction and integration
- Standardized reporting mechanisms for pipeline activity health

Avoid:

Each team with their own, unique process

Example Delivery Pipeline

Deploy/Debug/Test/ Validate Development stage

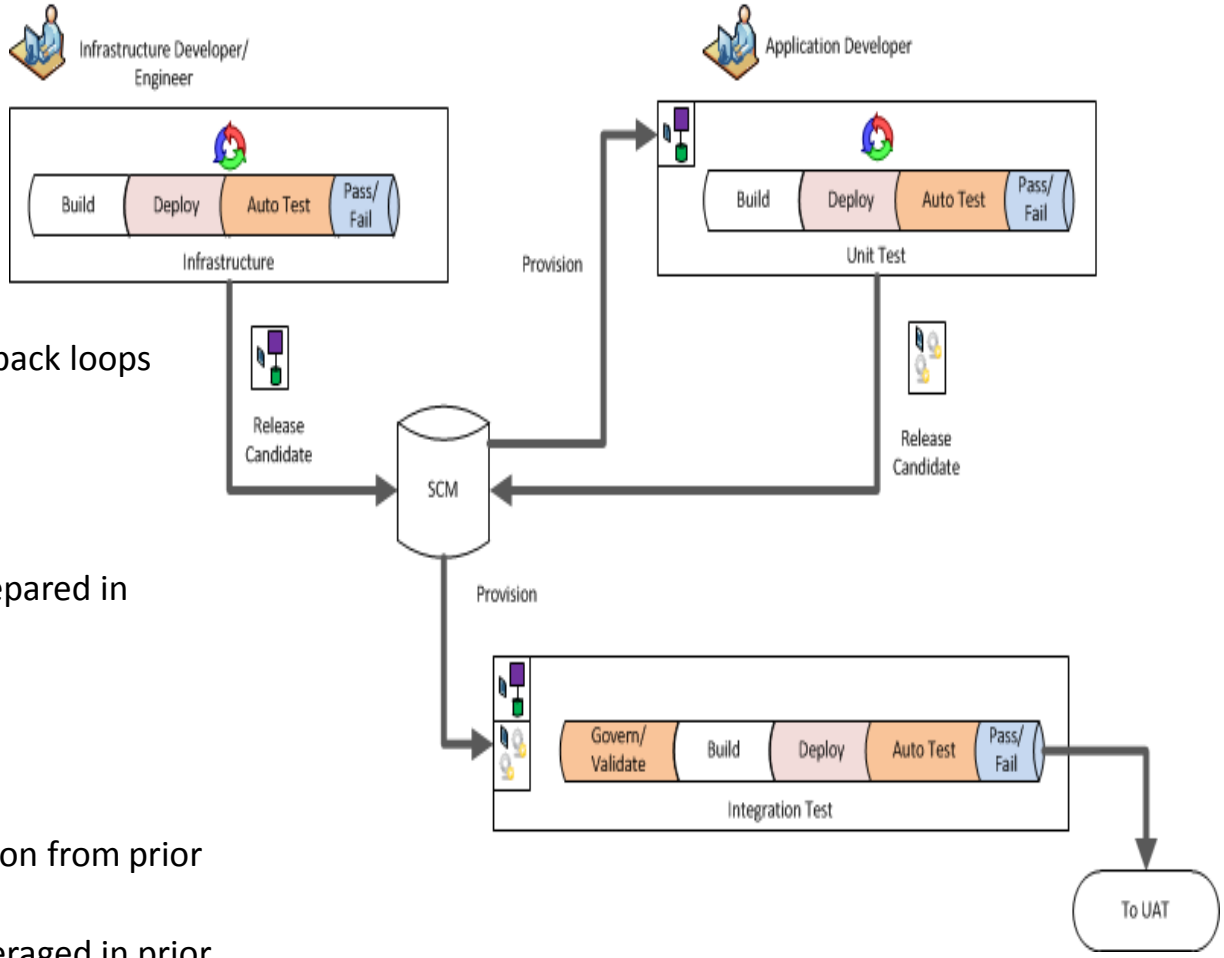
- Continuous Delivery for short feedback loops
- Define automation for deployment

Promote to next staging area

- Verify deployment automation, prepared in Development,
  also works in Staging

Promote to production

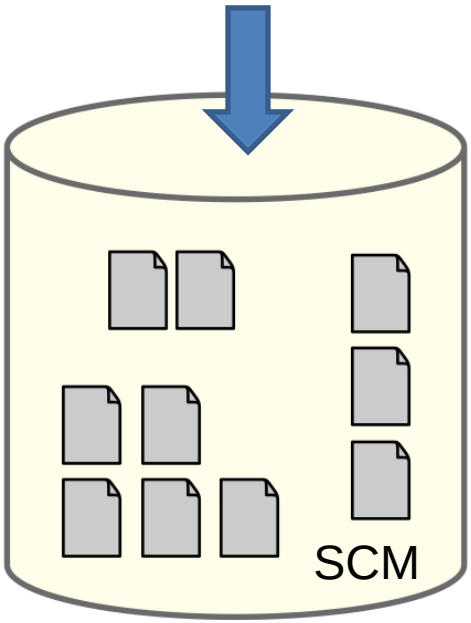
- Re-use same deployment automation from prior environments
- Use same automatic validation leveraged in prior milestones



Automation

Infrastructure As Code (IasC)

```
#!/usr/bin/env ruby

class DevopsDeployer
  def initialize(build_url, build_id)
    @log = Logger.new(LOG_FILE)
    @log.level = LOG_LEVEL

    @iaas_gateway = IaasGateway.new(HsltProvider.new(),
LOG_FILE, LOG_LEVEL)
    @server_instance = nil

    rtc_build_system_provider = RtcBuildSystemProvider.new(
RTC_REPOSITORY_URL, RTC_USER_ID, RTC_PASSWORD_FILE)
    @build = rtc_build_system_provider.resolve_build(
build_url, ENV['buildResultUUID'], build_id)
    @build_system_gateway = BuildSystemGateway.new(
rtc_build_system_provider, LOG_FILE, LOG_LEVEL)
  end

  def add_build_stamp
    template_file = WEB_APP_ROOT +
"/app/templates/pages/page.html"
    @log.info "Adding build ID stamp #{@build.id} to \
#{template_file}"

    # Read in the file's contents as a string, replace
    # the build_id, then overwrite the original contents
    # of the file
    text = File.read(template_file)
    new_text = text.gsub(/\{\{ build_id \}\}/,
"<a href=\"#{@build.uri}\">#{@build.id}</a>")
    File.open(template_file, "w") { |file|
      file.puts new_text
    }
  end

# ...
```



Capture system configuration as automation which can be applied to a new or running system in such a way that it can be version controlled, rolled back, and managed like source code.

As more automated routines are developed within the infrastructure areas to perform provisioning, resource management and administrative activities, they become more important as enterprise assets.

Loss of these routines can be catastrophic to the operations of the business and need to be managed like other code assets

Well defined SCM processes for check-in/checkout, testing and migration of assets is required

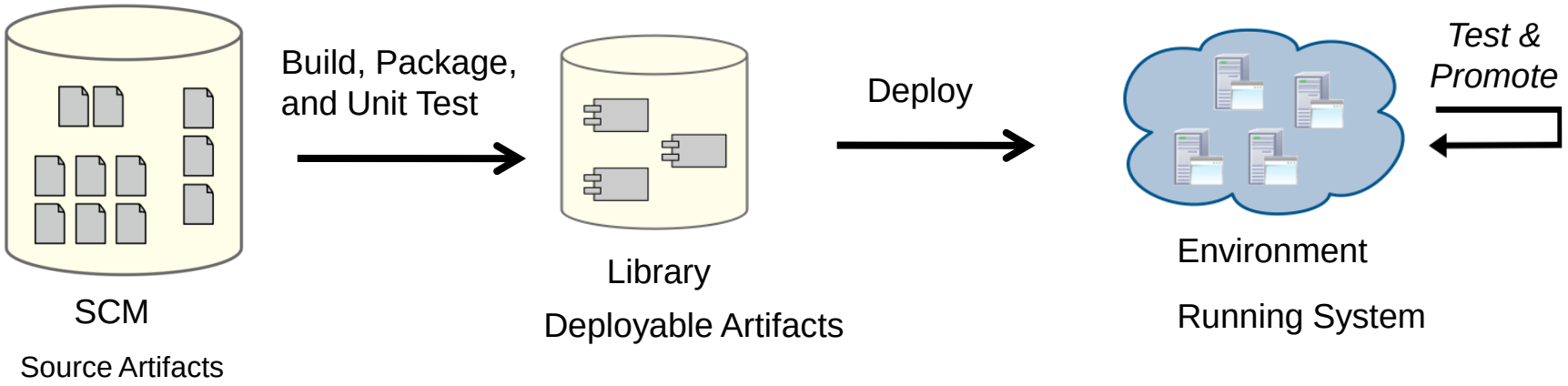
Continuous Delivery

Short feedback loops to detect problems and ensure quality

- Test Early and Often

Automation (IasC) becomes part of the what is being tested

- Results in automation logic undergoing testing 1000's of times prior to deployment in production



Goal:

- Don't fix problem through administrative consoles, fix once in the automation logic

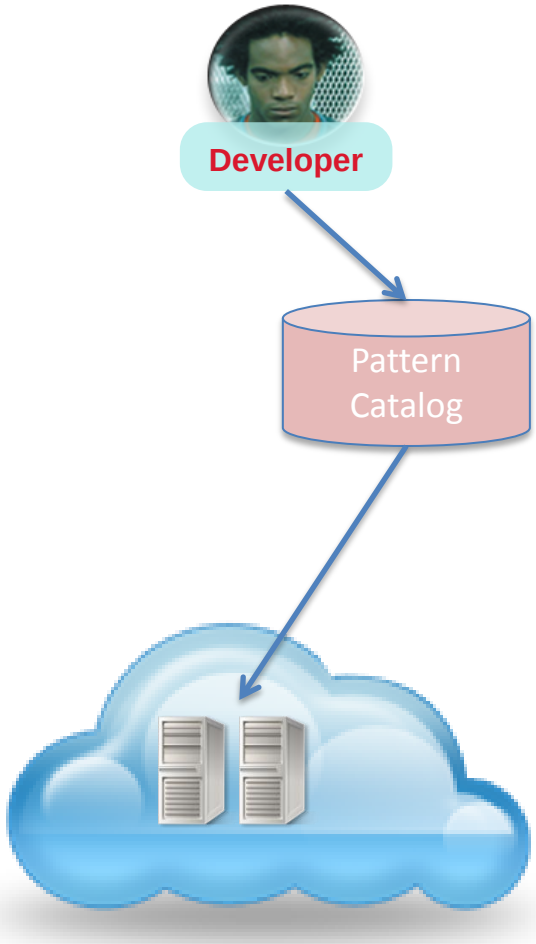
Representative Environments

Key goal in DevOps is ability to provide developers ready access to representative environments for their target application architectures

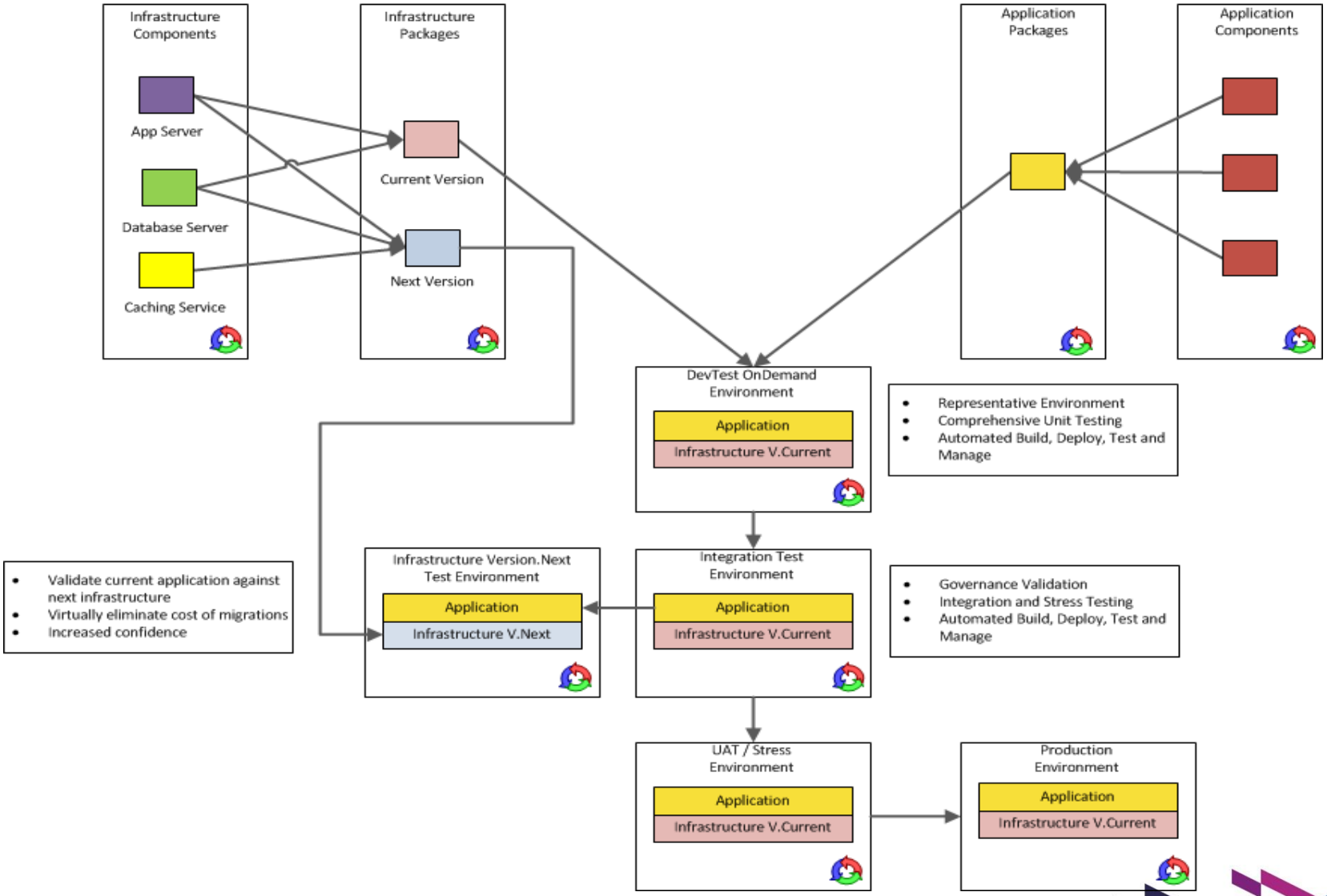
Ability to validate application (and infrastructure) code early and often

- Includes provisioning, administrative and operational functionality

Cloud-based environments allow greater ability to provide realism and augmentation that a standard development environment doesn't have



DevOps Example



Adjusting Operations Mindset

Infrastructure Developer vs. Engineer/Administrator

- Need to bring a software development mindset to the operational areas
- Replicate, where appropriate, standard architecture/development tools and methodologies

Use an Agile approach to delivery of routines

- Continuous, incremental improvements and delivery of new functionality
- Automated unit and integration testing improves operational runtimes

Source Control Management

- Automation routines and scripts are fundamental to Operations
- Managing Operations routines like source code offers several benefits:
  - Central point of truth as routines and environments change
  - Backup in case of loss
  - Identify possible regressions by comparing with prior versions

Example Managed Assets:

- Perl, Jython, WSADMIN, ANT scripts, Service orchestration routines (opsware, buildforge, etc), Infrastructure Gold copies components

Thank you