Continuous Everything

Soumyak Bhattacharyya

Product Developer

DevOps Enthusiast

Simple ideas are easier to understand. Ideas that are easier to understand are repeated. Ideas that are repeated change the world.

Continuous Integration

Continuous Integration

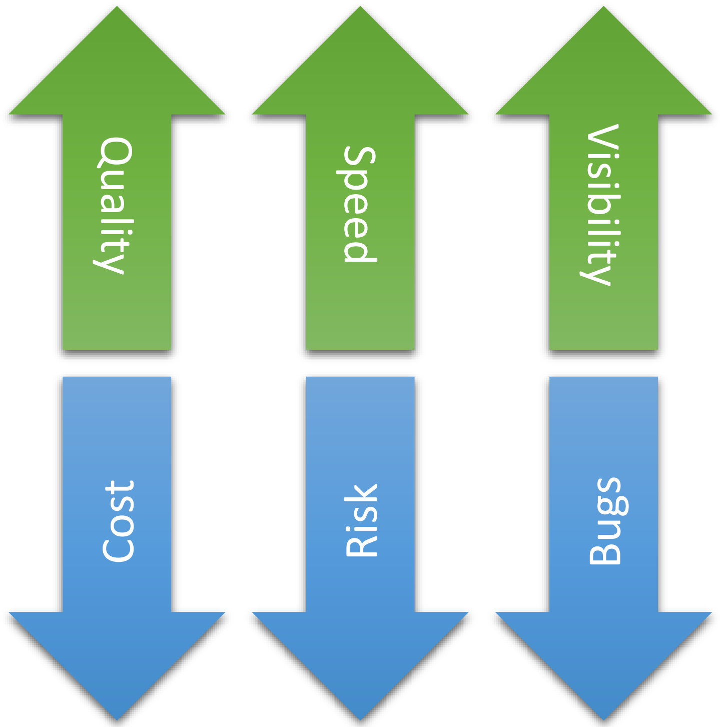
Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors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

Principles

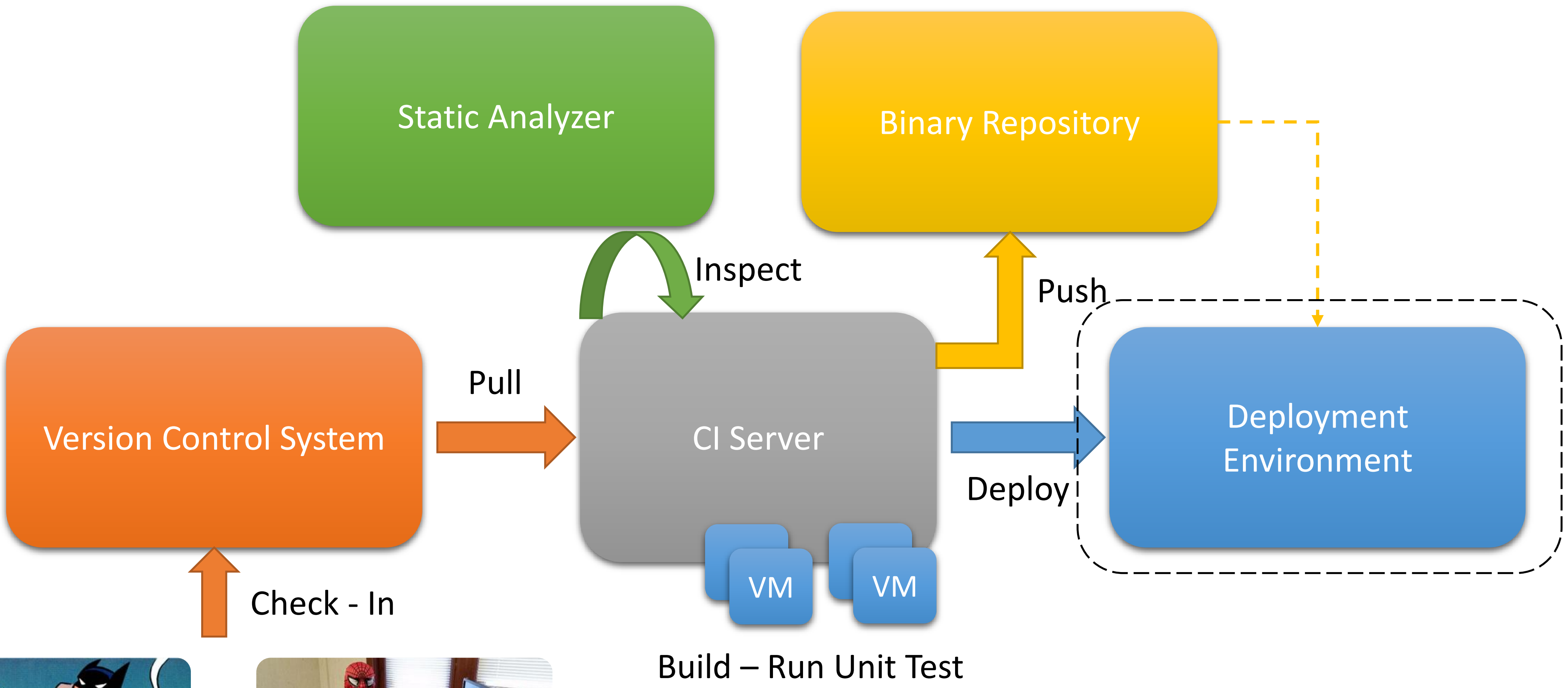
1. Maintain a Single Source Repository
2. Make Your Build Automated & Self-Testing
3. Everyone Commits To the Mainline Every Day
4. Every Commit Should Build the Mainline on an Integration Machine
5. Fix Broken Builds Immediately
6. Keep Build Fast
7. Test Environment Is A Clone Of Production Environment
8. Binary Lives In Binary Repository
9. Radiate Information
10. Automate Deployment

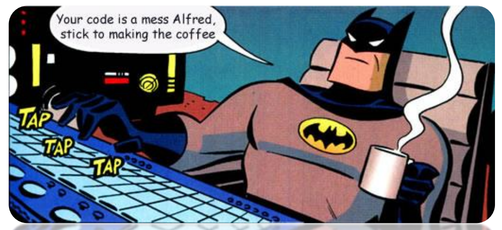
Reference : https://martinfowler.com/articles/continuousIntegration.html

Outcome



Continuous Integration

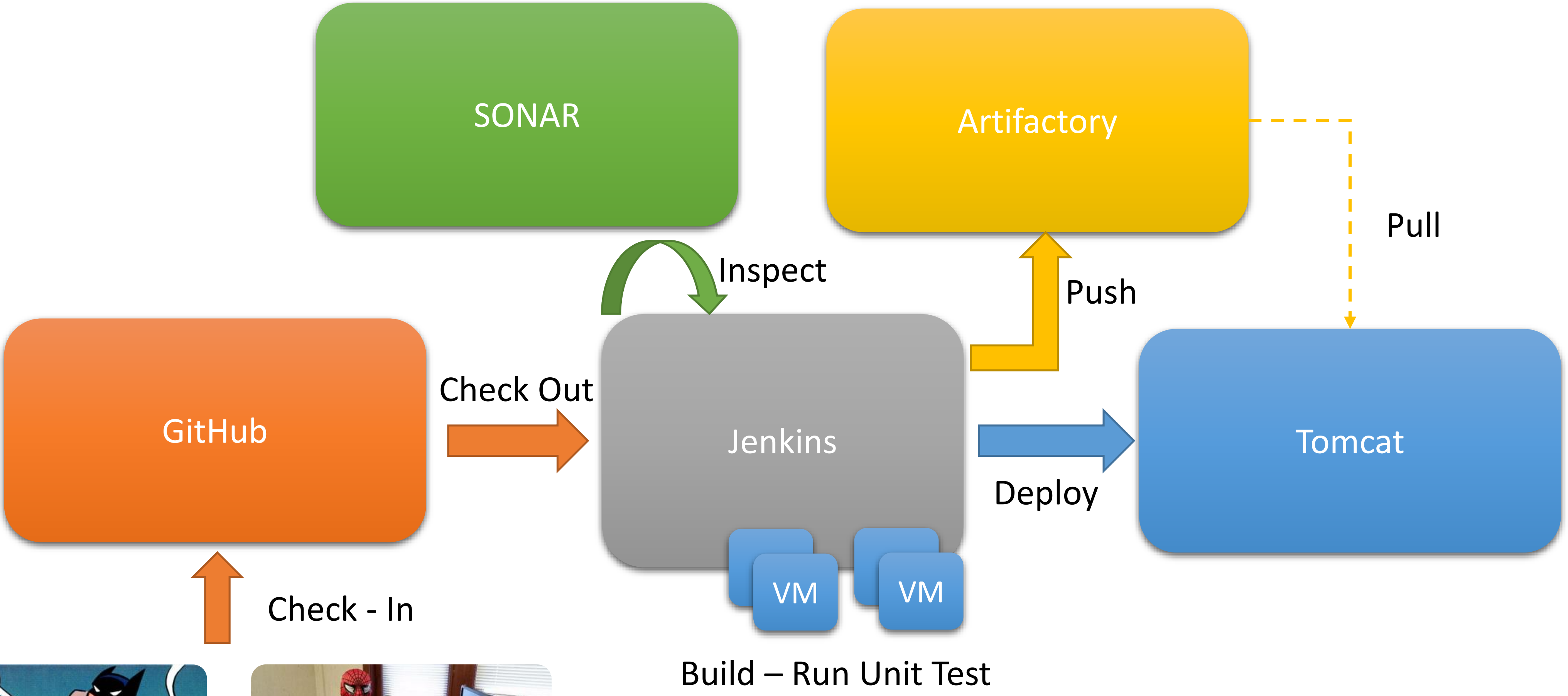




Developers

Conceptual Flow

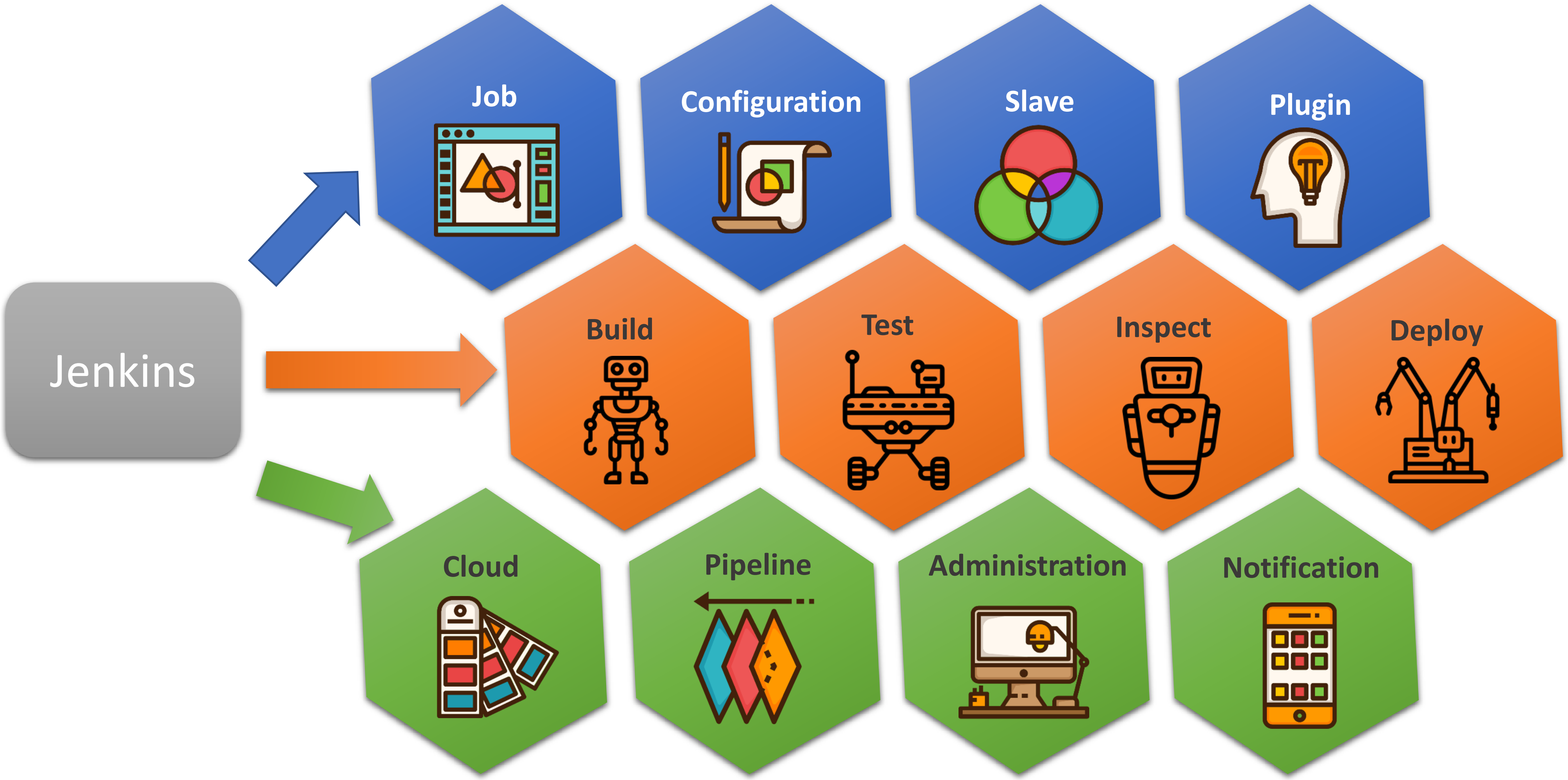
Continuous Integration



Developers

Typical Implementation

Continuous Integration



Continuous Inspection

sonar

https://www.sonarsource.com/

Inspect

6000 + Downloads per month

1500 + Subscriber to mailing list

60 + Open Source Plugin

150,000 + Downloads

- Code Duplication
- Bad Distribution of Complexity
- Spaghetti Design
- Lack of Unit Test
- Insufficient Coding Standard
- Potential Bugs
- Inadequate Comment … and so on

Continuous Integration (Produce binary)

JFrog Artifactory

Deploy

Repository Manager

1. To facilitate artifact storage & proxy
2. To avoid hitting public remote repository
3. To avoid being inefficient, unreliable and non – secure
4. To deploy, manage and share local artifacts
5. To establish full control on artifact resolution
6. To build once and deploy many times

Features

1. Integrate with CI Engines, Build Tools (Maven / Gradle)

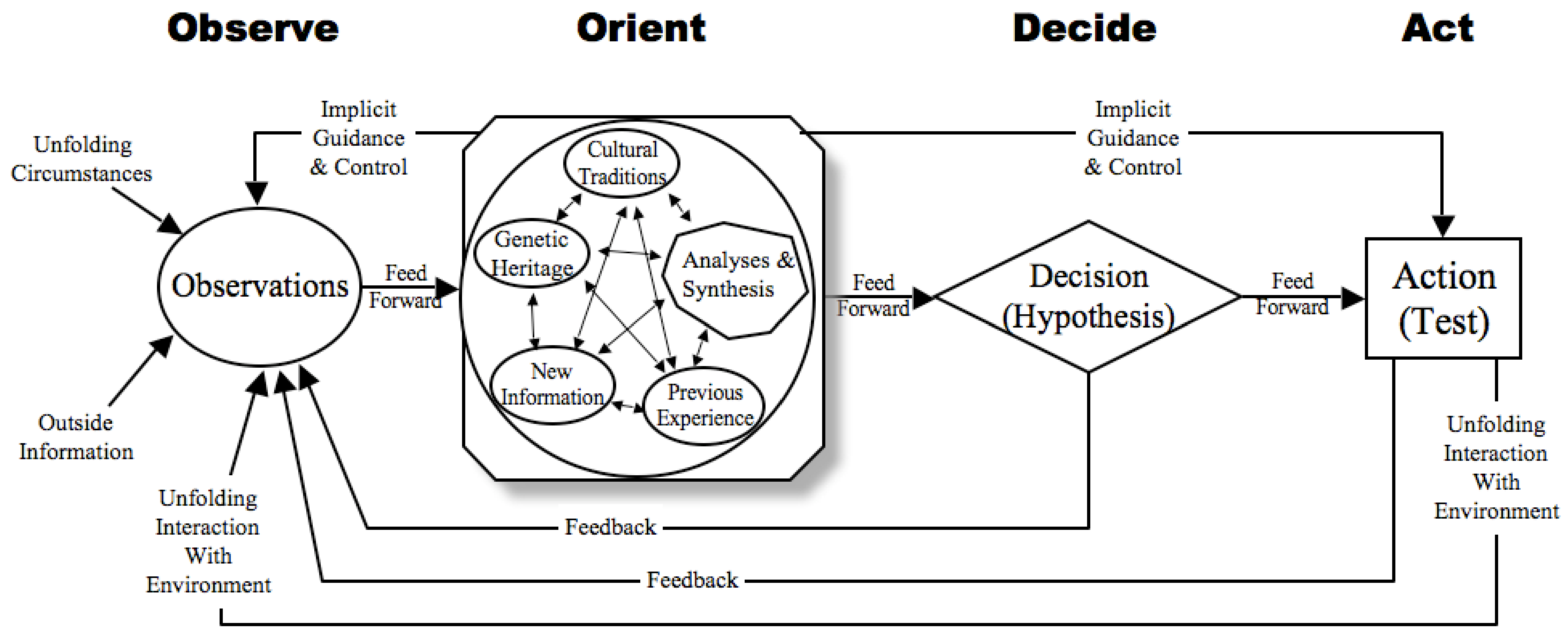
2. Host and proxy
    1. Maven Dependencies
    2. Docker Images
    3. NuGet packages
    4. node.js packages
    5. Bower registry
    6. PyPI distributions
    7. Microsoft .NET ecosystem

3. Watch / Filter / Search for artifacts

Demo Time

Continuous Delivery

Why Do We Need Continuous Delivery

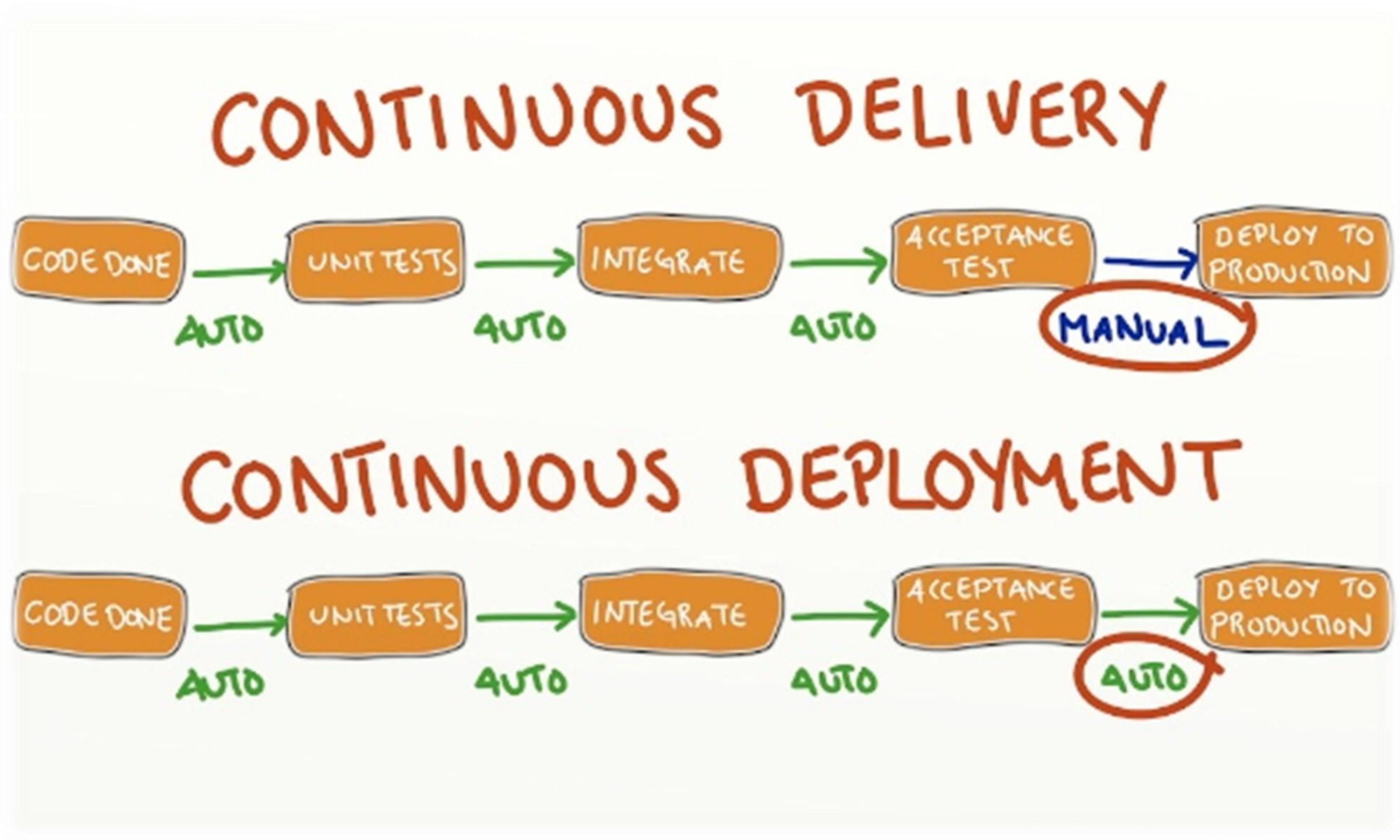


OODA loop By John Boyd

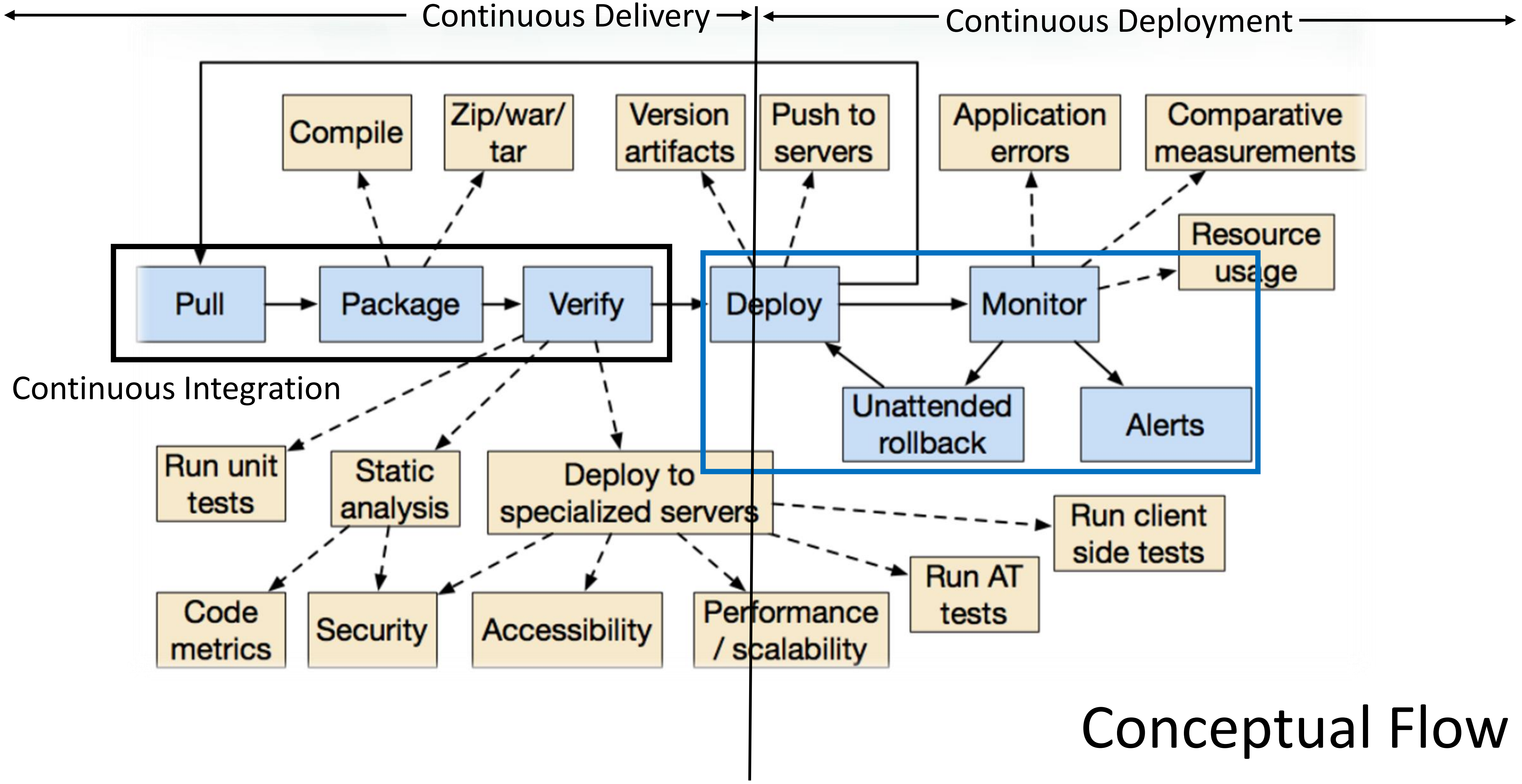
Cycle Time : Act phase of OODA loop

Cycle Time is the reaction time of Organization

Continuous Delivery (vs. Deployment)

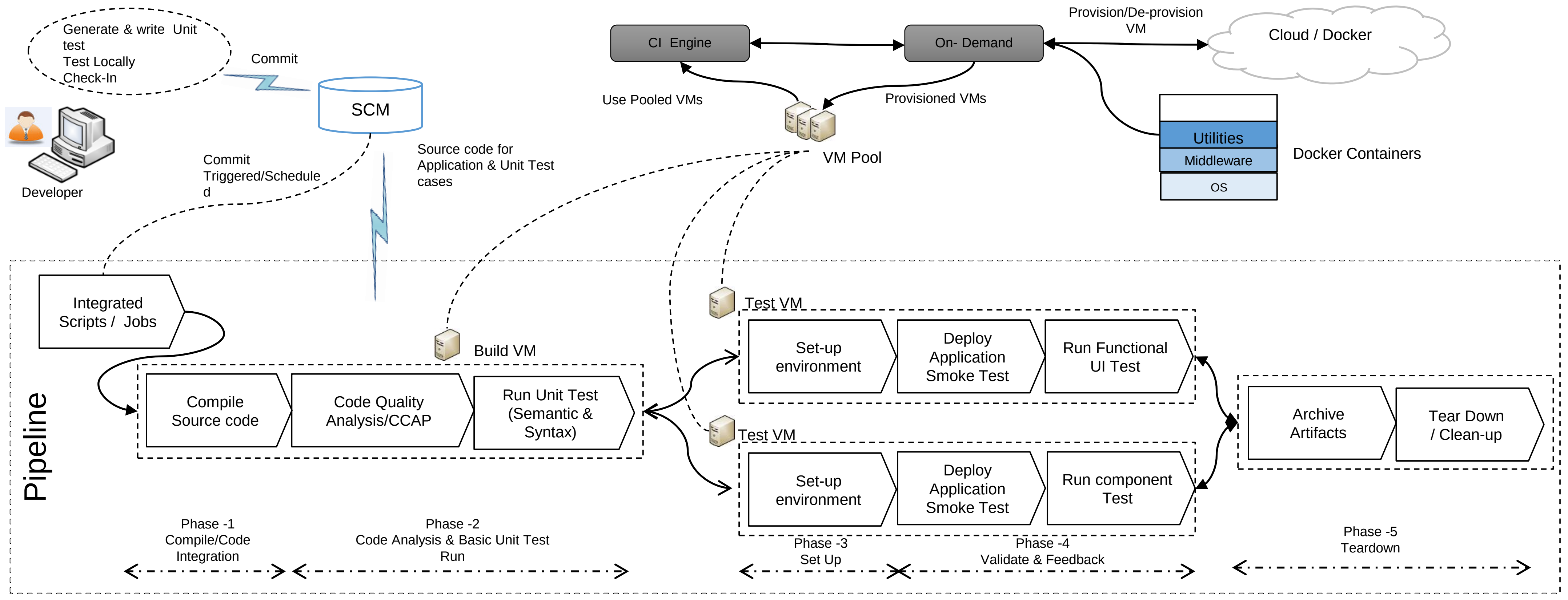


Continuous Delivery



Conceptual Flow

Continuous Delivery

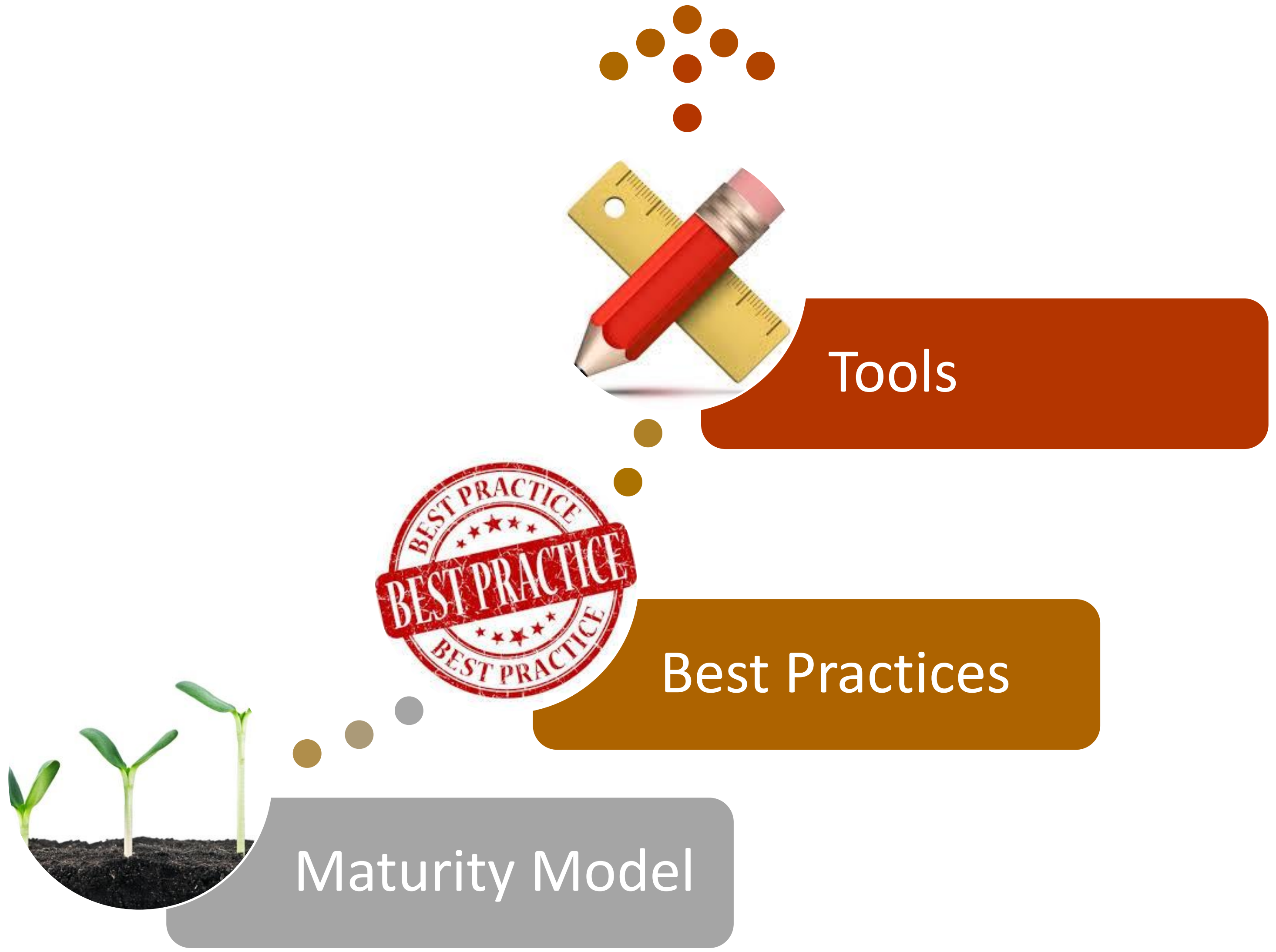

Generate & write Unit test
Test Locally
Check-In
Commit
SCM
Developer
Commit Triggered/Scheduled
Source code for Application & Unit Test cases
CI Engine
On- Demand
Provision/De-provision VM
Cloud / Docker
Use Pooled VMs
Provisioned VMs
VM Pool
Utilities
Middleware
OS
Docker Containers
Pipeline
Integrated Scripts / Jobs
Build VM
Compile Source code
Code Quality Analysis/CCAP
Run Unit Test (Semantic & Syntax)
Test VM
Set-up environment
Deploy Application Smoke Test
Run Functional UI Test
Test VM
Set-up environment
Deploy Application Smoke Test
Run component Test
Archive Artifacts
Tear Down / Clean-up
Phase -1 Compile/Code Integration
Phase -2 Code Analysis & Basic Unit Test Run
Phase -3 Set Up
Phase -4 Validate & Feedback
Phase -5 Teardown

Release Management Platform

- Release Dashboard
- Automated Rollback
- Agentless Architecture
- Provisioning
- Workflow
- Audit
- Extensibility
- Self Service

Demo Time

Parting Note !!!



Q & A

Thanks !!!