CICD with Jenkins

Module One: Introduction to CICD

Defining CICD

- CICD is not a methodology
  - It is not Agile or DevOps
  - Although both rely on CICD and use it extensively
- CICD is process automation applied to SE
  - Similar to other kinds of automation (including robotic process automation)
  - The goal is to improve process efficiency and effectiveness
  - CICD is process agnostic
  - It can be used anywhere a SE process is well defined
  - Using CICD with bad processes makes them worse

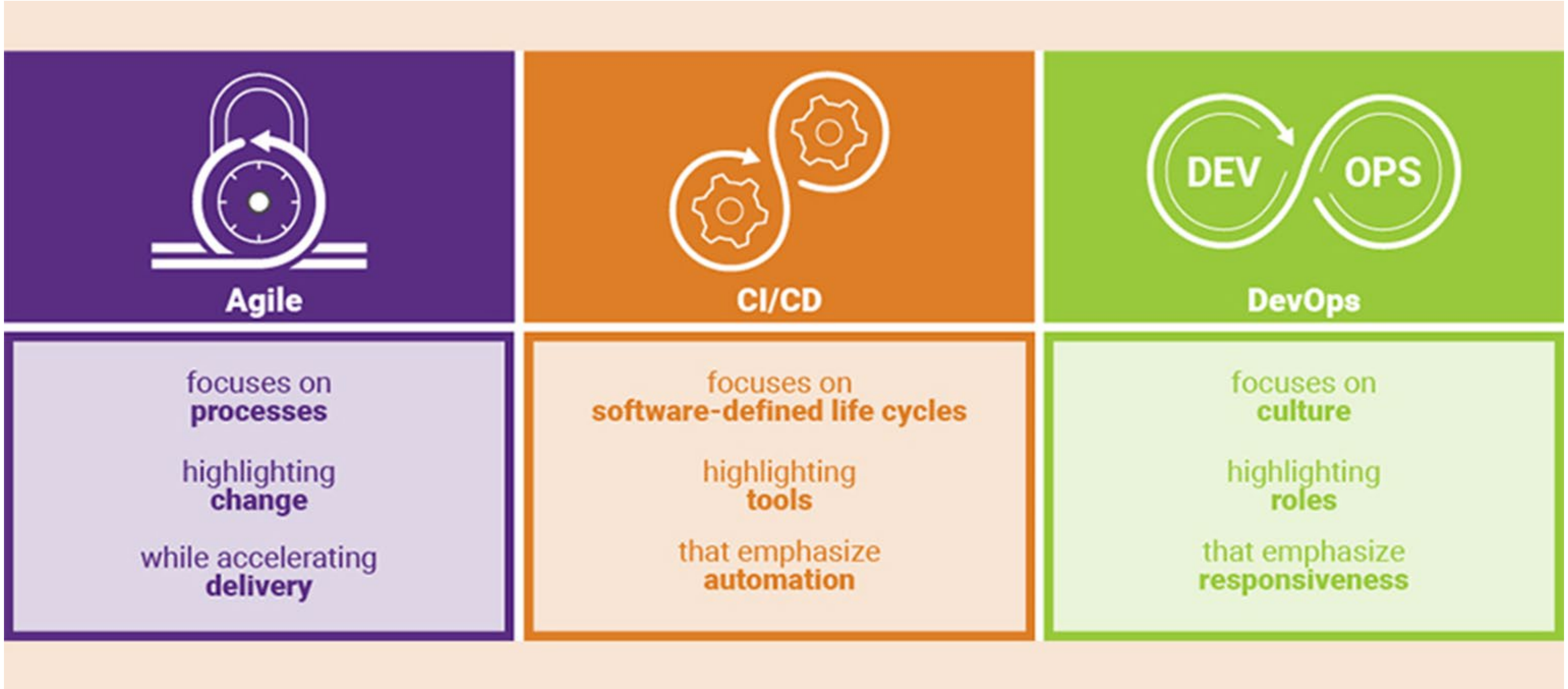
A fool with a tool is still a fool

Martin Fowler

A computer lets you make more mistakes faster than any invention in human history – with the possible exceptions of handguns and tequila

Mitch Ratcliffe

Agile, DevOps and CICD



Agile

focuses on
processes

highlighting
change

while accelerating
delivery

CI/CD

focuses on
software-defined life cycles

highlighting
tools

that emphasize
automation

DevOps

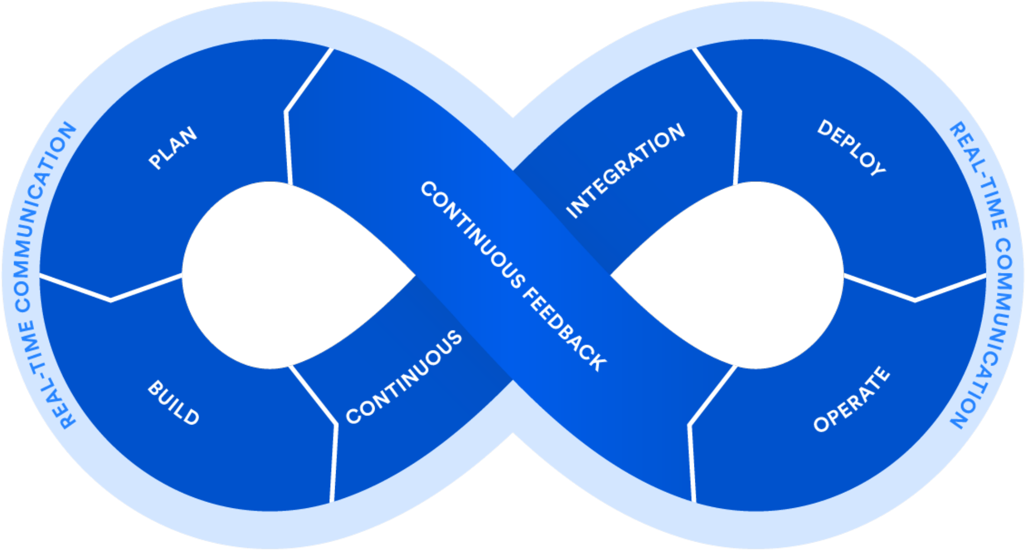
focuses on
culture

highlighting
roles

that emphasize
responsiveness

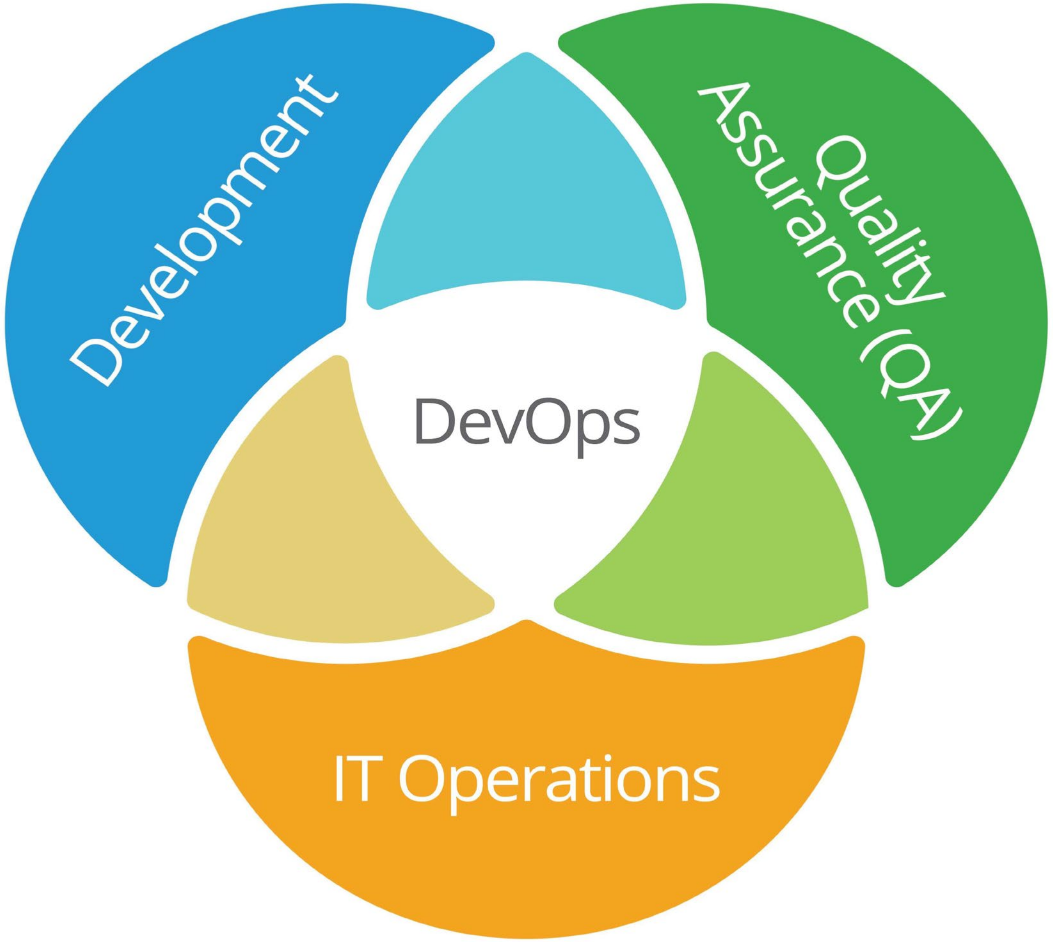
DevOps

- Driven by virtualization and Infrastructure as code
- Dev and Ops had been two separate worlds
  - Dev was sort of automated
  - Ops was manual and bare metal
- Virtualization turned it all into code
  - Now the same tools can be used in the entire life cycle of a software product
  - Opportunity for full process automation support



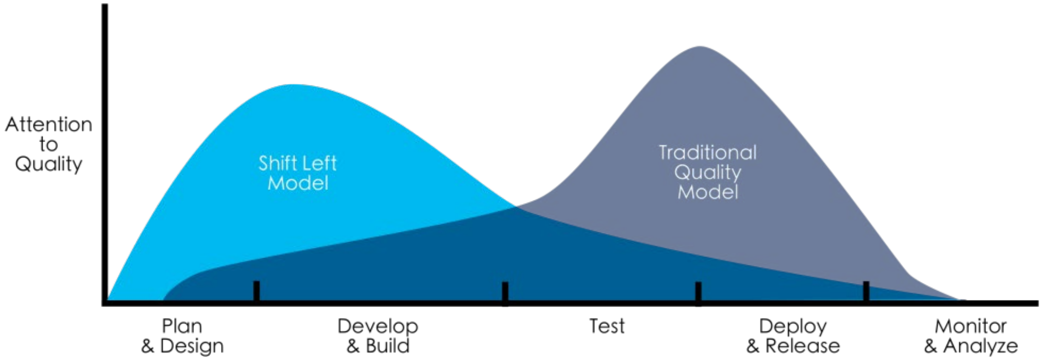
The Goal of DevOps

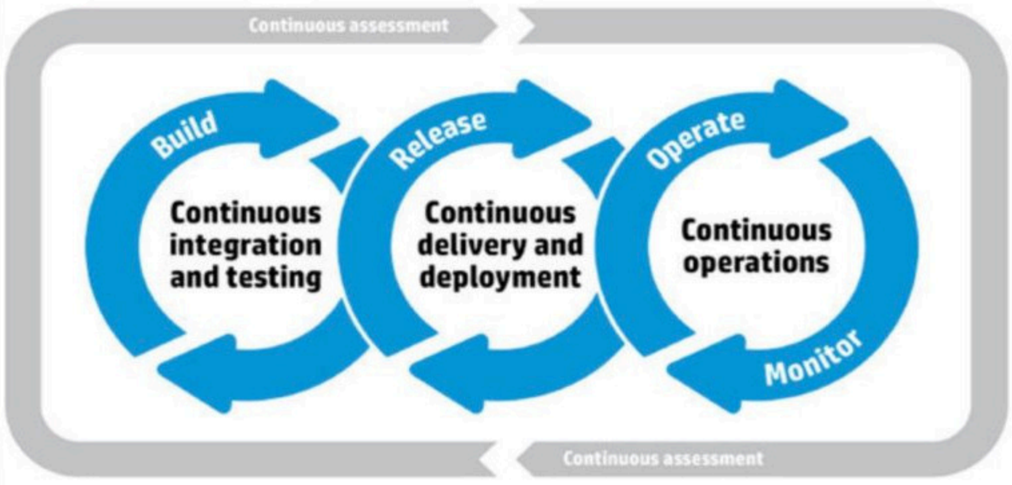
- Desilo-ize the three areas in software development
- Get everyone using the same sorts of tools, practices and automations
- CICD
  - **Continuous Integration**: continuous integration of multiple development activities
  - **Continuous Delivery**: build artifact made available for delivery
  - **Continuous Deployment**: Delivered artifact is also pushed into operational environment



Continuous Testing

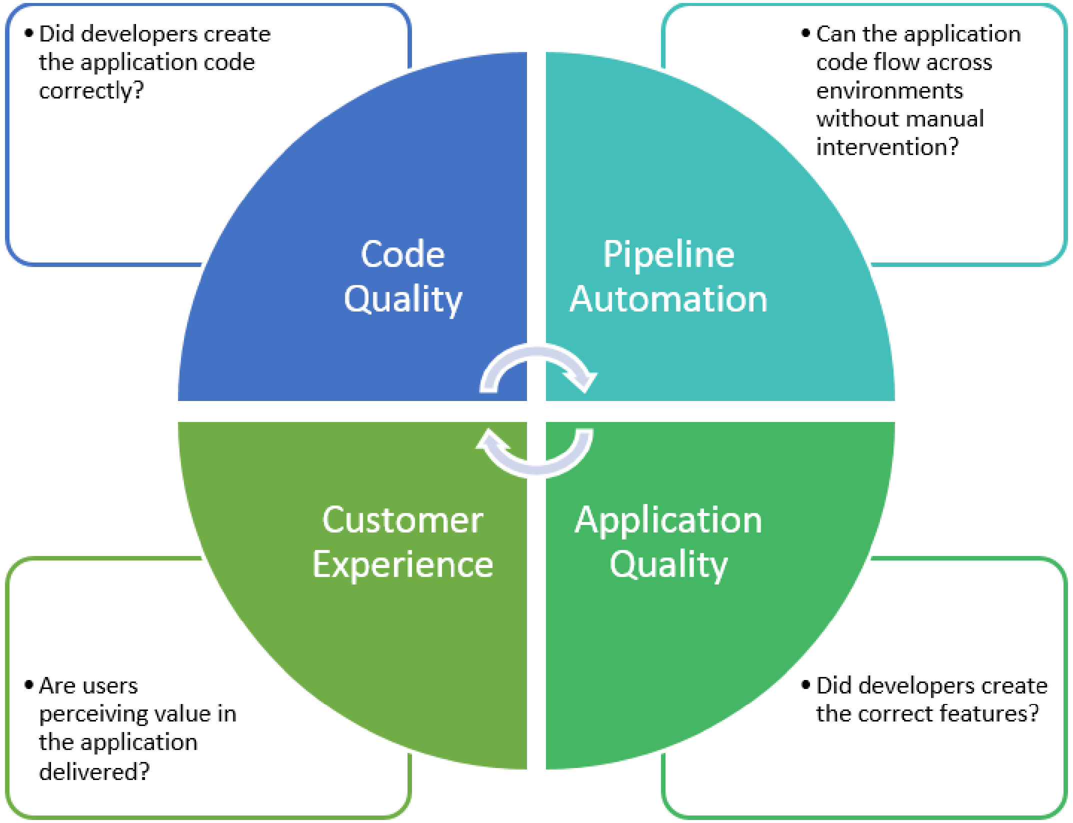
- Continuous Testing
  - Every artifact is tested as it is created
- Shift Left Model
  - Test early, test often
- CICD also adds
  - Automated testing at every stage
- CT is triggered by events in the CICD process
  - Checking in code => automated unit testing
  - Build => integration testing





Continuous Testing

- Does not replace human based testing
  - Like pair programming and code reviews
- Creates “quality gates”
  - Development pipelines abort when tests fail
- Adding continuous security testing and security planning is called DevSecOps



Did developers create the application code correctly?
Code Quality
Pipeline Automation
Can the application code flow across environments without manual intervention?
Customer Experience
Application Quality
Are users perceiving value in the application delivered?
Did developers create the correct features?

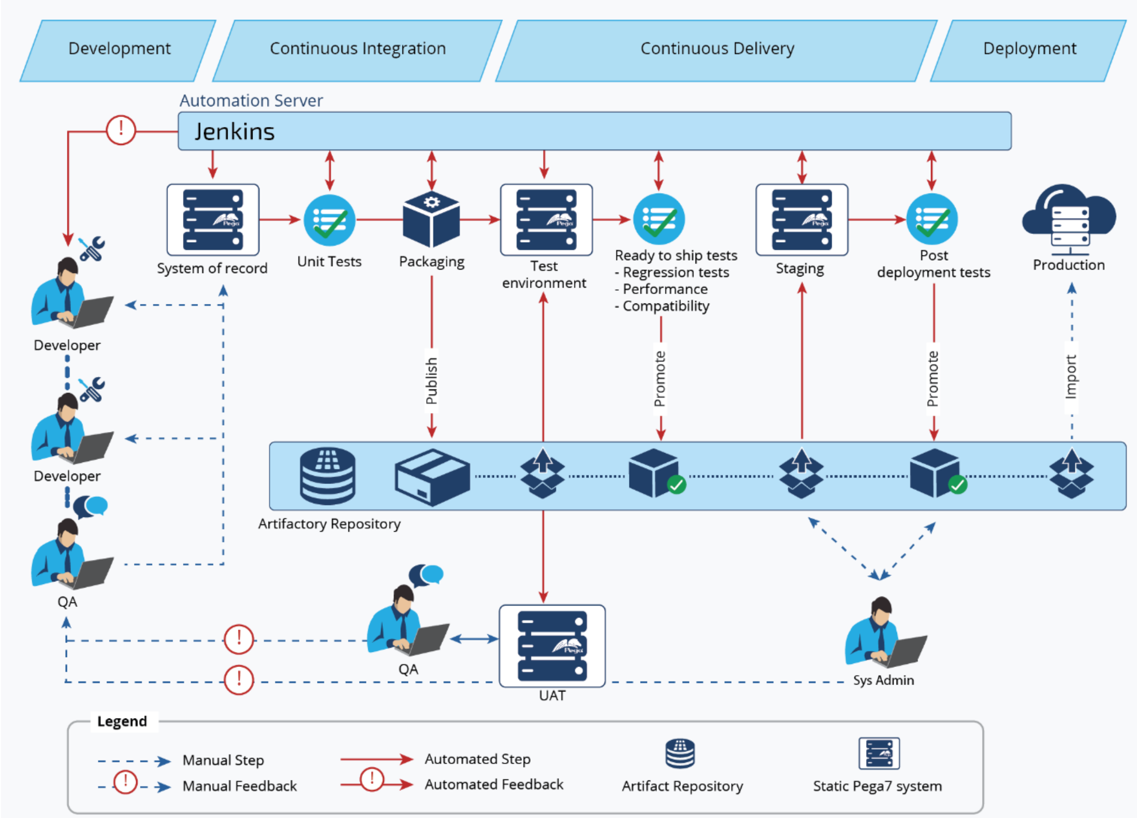
More than the act of testing, the act of designing tests is one of the best bug preventers known. The thinking that must be done to create a useful test can discover and eliminate bugs at every stage in the creation of software, from conception to specification, to design, coding and the rest.

If you can't test it, don't build it.
If you don't test it, rip it out.

Boris Beizer

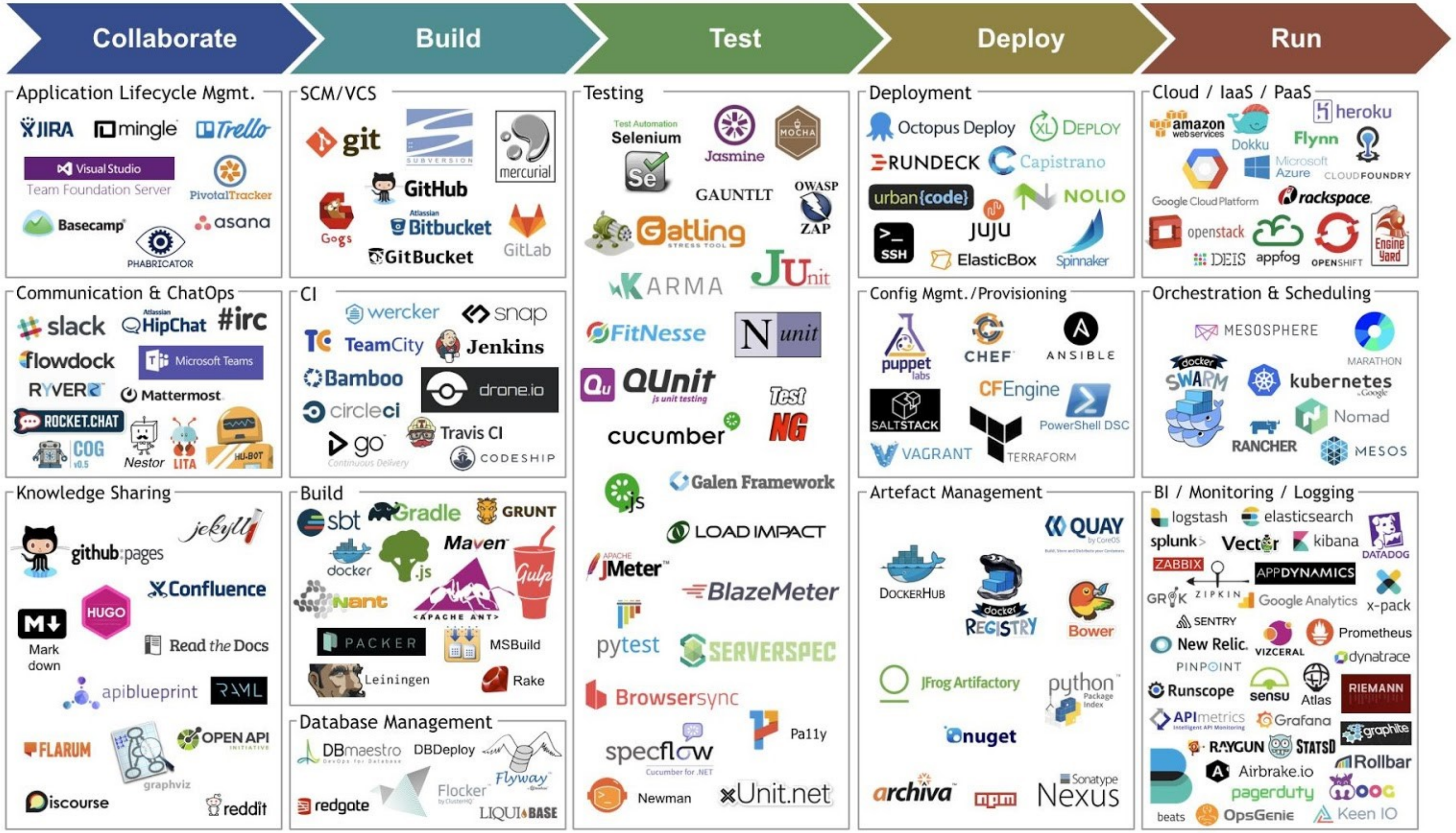
Pipelines

- Series of automated tasks
  - Implements a CICD flow
- Managed by an orchestration tool
  - Jenkins is the used in the diagram



Automation Tools

- A jungle of CICD tools
  - Each automates some part of the pipeline
  - Lots of overlap
  - Not all are compatible
  - Wide range in quality
- Developing a toolset for CICD
  - Can be very problematic
  - Especially if some tools become obsolete



CICD Benefits

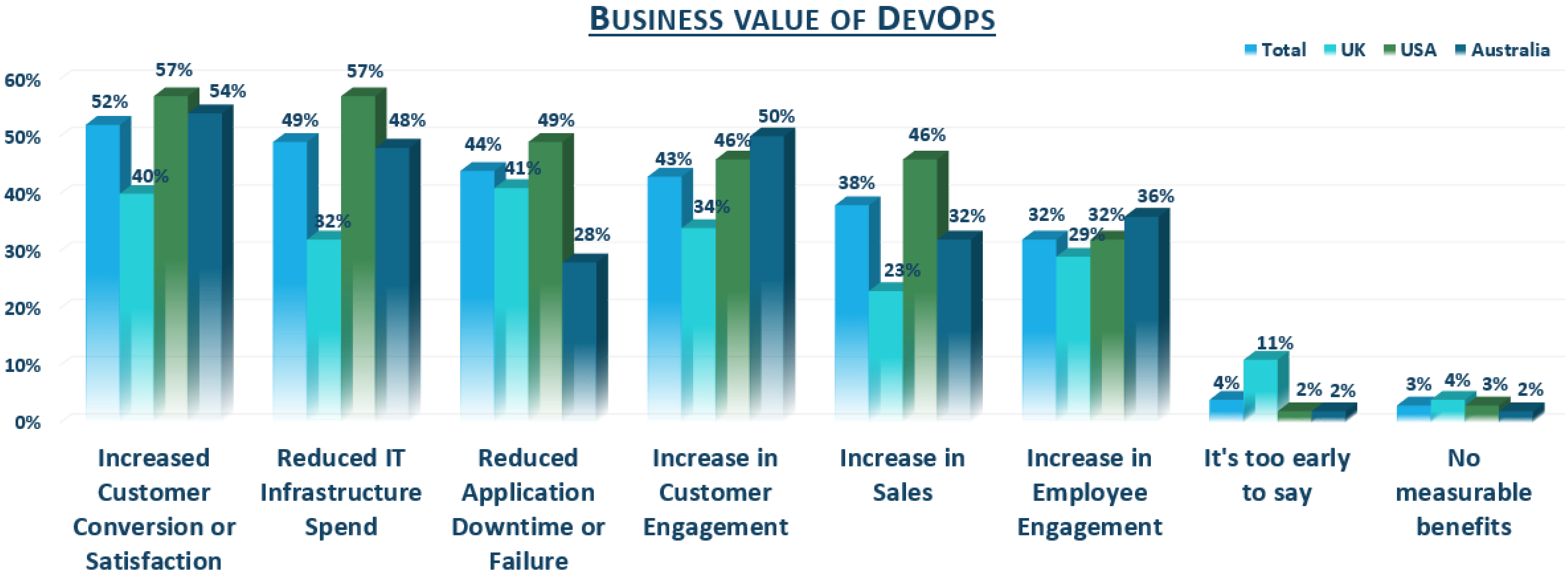
1. Smaller code changes are simpler (more atomic) and have fewer unintended consequences.
2. Fault isolation is simpler and quicker.
3. Mean time to resolution (MTTR) is shorter because of the smaller code changes and quicker fault isolation.
4. Testability improves due to smaller, specific changes. These smaller changes allow more accurate positive and negative tests.
5. Elapsed time to detect and correct production escapes is shorter with a faster rate of release.
6. The backlog of non-critical defects is lower because defects are often fixed before other feature pressures arise.
7. The product improves rapidly through fast feature introduction and fast turn-around on feature changes.
8. Upgrades introduce smaller units of change and are less disruptive.

CICD Benefits

9. CI-CD product feature velocity is high. The high velocity improves the time spent investigating and patching defects.
10. Feature toggles and blue-green deploys enable seamless, targeted introduction of new production features.
11. You can introduce critical changes during non-critical (regional) hours. This non-critical hour change introduction limits the potential impact of a deployment problem.
12. Release cycles are shorter with targeted releases and this blocks fewer features that aren’t ready for release.
    - End-user involvement and feedback during continuous development leads to usability improvements. You can add new requirements based on customer’s needs on a daily basis.

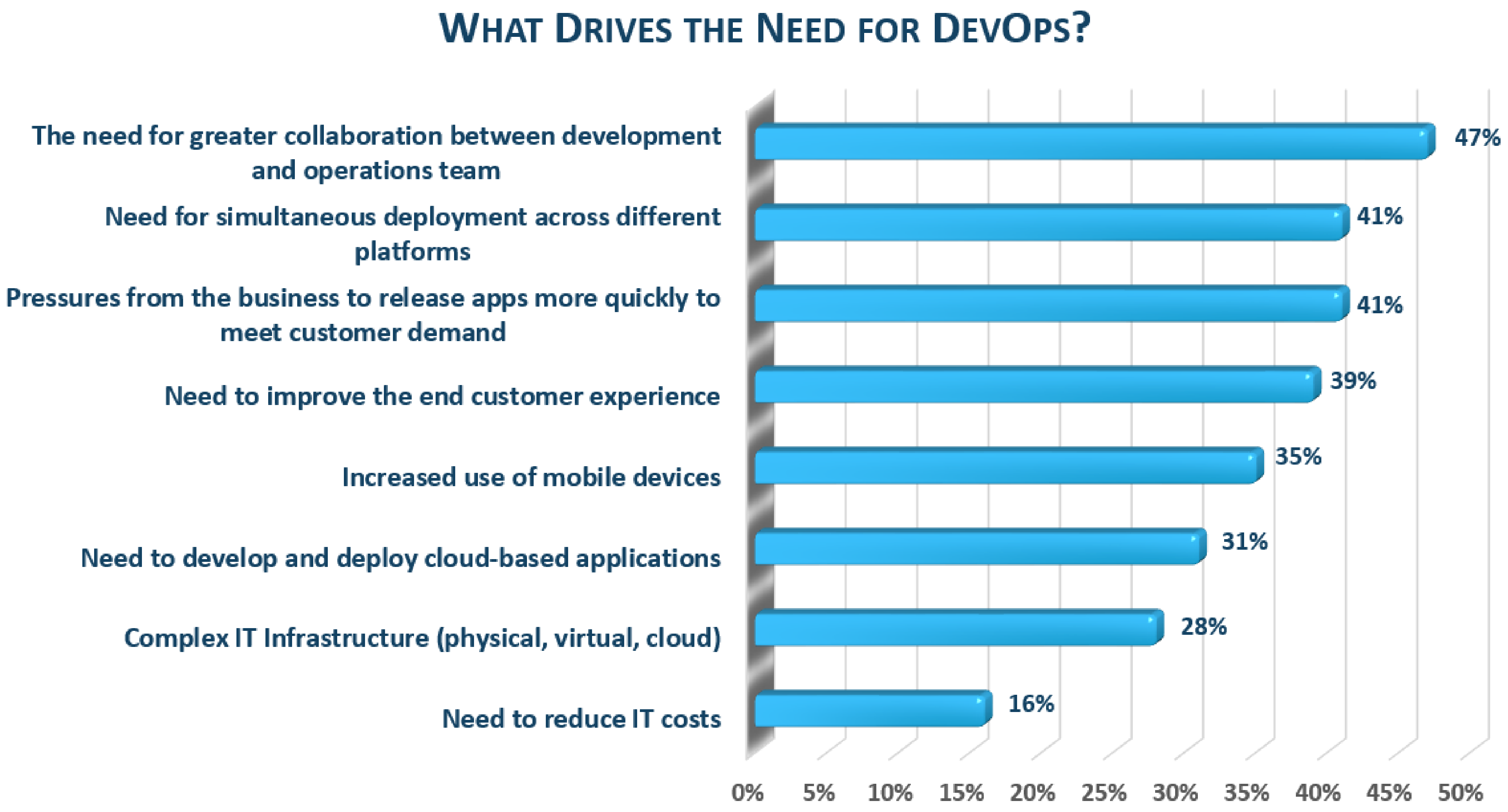
https://help.mypurecloud.com/articles/benefits-continuous-integration-continuous-deployment-ci-cd/

CICD Benefits



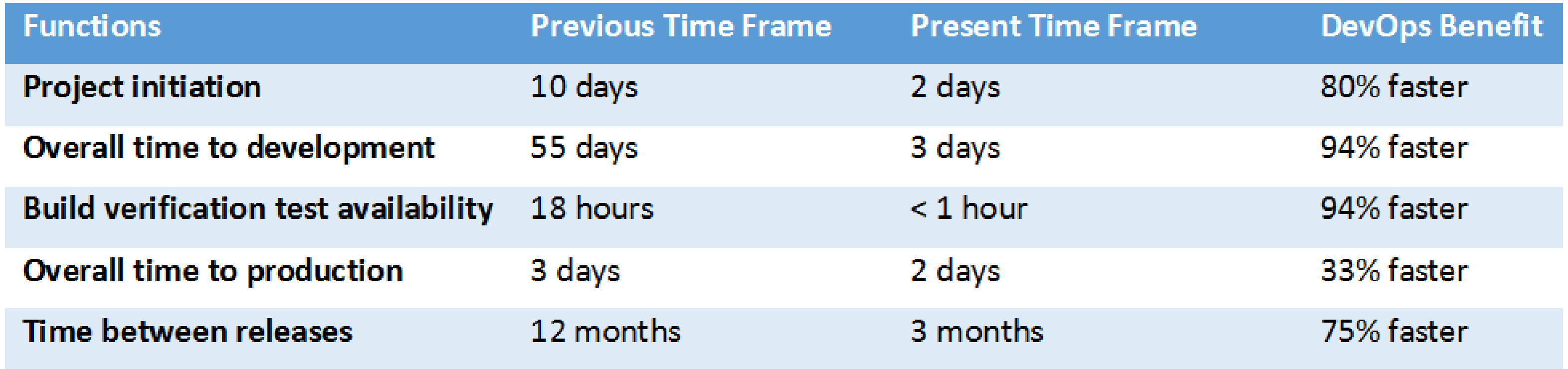
https://www.compaid.co.in/Articles/DevOps-Value-and-Cost-Savings

CICD Drivers



https://www.compaid.co.in/Articles/DevOps-Value-and-Cost-Savings

IBM Internal DevOps



Functions	Previous Time Frame	Present Time Frame	DevOps Benefit
Project initiation	10 days	2 days	80% faster
Overall time to development	55 days	3 days	94% faster
Build verification test availability	18 hours	< 1 hour	94% faster
Overall time to production	3 days	2 days	33% faster
Time between releases	12 months	3 months	75% faster

DevOps, clearly an extension of lean and agile principles, was as much, in IBM, born of necessity to respond to a pervasive industry mandate to “do more with less” and has evolved to “quality software faster.”

- Kristof Kloeckner,
General Manager,
IBM Software Group – Rational

https://www.compaid.co.in/Articles/DevOps-Value-and-Cost-Savings

Challenges for CICD

1. Organization silos and corporate culture
   - Lack of communication between development, QA and operations
2. Failure to automate testing or do continuous testing
   - QA starts lagging behind development requiring rework to fix buggy code
3. Legacy systems integration
   - Automated tools may not be available for legacy systems
   - E.g. Unit testing frameworks for COBOL code
4. Complexity and size of applications
   - Trying to apply CICD to too big a “chunk” of development
   - Especially when introducing CICD

CI and CD in Agile Development

- Continuous Integration and Continuous Delivery become essential ingredients for teams doing iterative and incremental software delivery in Agile Development
  - Developers share a common source code repository
  - Dedicated Continuous Integration environment
  - All code must pass unit tests
  - Integrate often
  - Regression tests run often
  - Code metrics are published
  - Every change to the system is releasable to production
  - **Automation is the key**

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)
- Goal is to merge and test the code **continuously** to catch issues early by **automating integration process**
  - Your project must have a reliable, repeatable, and automated build process involving no human intervention
- Continuous Integration Server (Jenkins) is responsible for performing the integration tasks
- Concepts of unit testing, static analysis, failing fast and automated testing are core to Continuous Integration

Continuous Integrations Practices

- Single source repository for all developers
- Build automation
- Every change to VCS should make a new build
- Keep the builds fast and trackable
- Make the builds self-testing
- Test the builds in production-like environment
- Keep all verified releases in artifacts repository and available to everyone
- Publish coding metrics

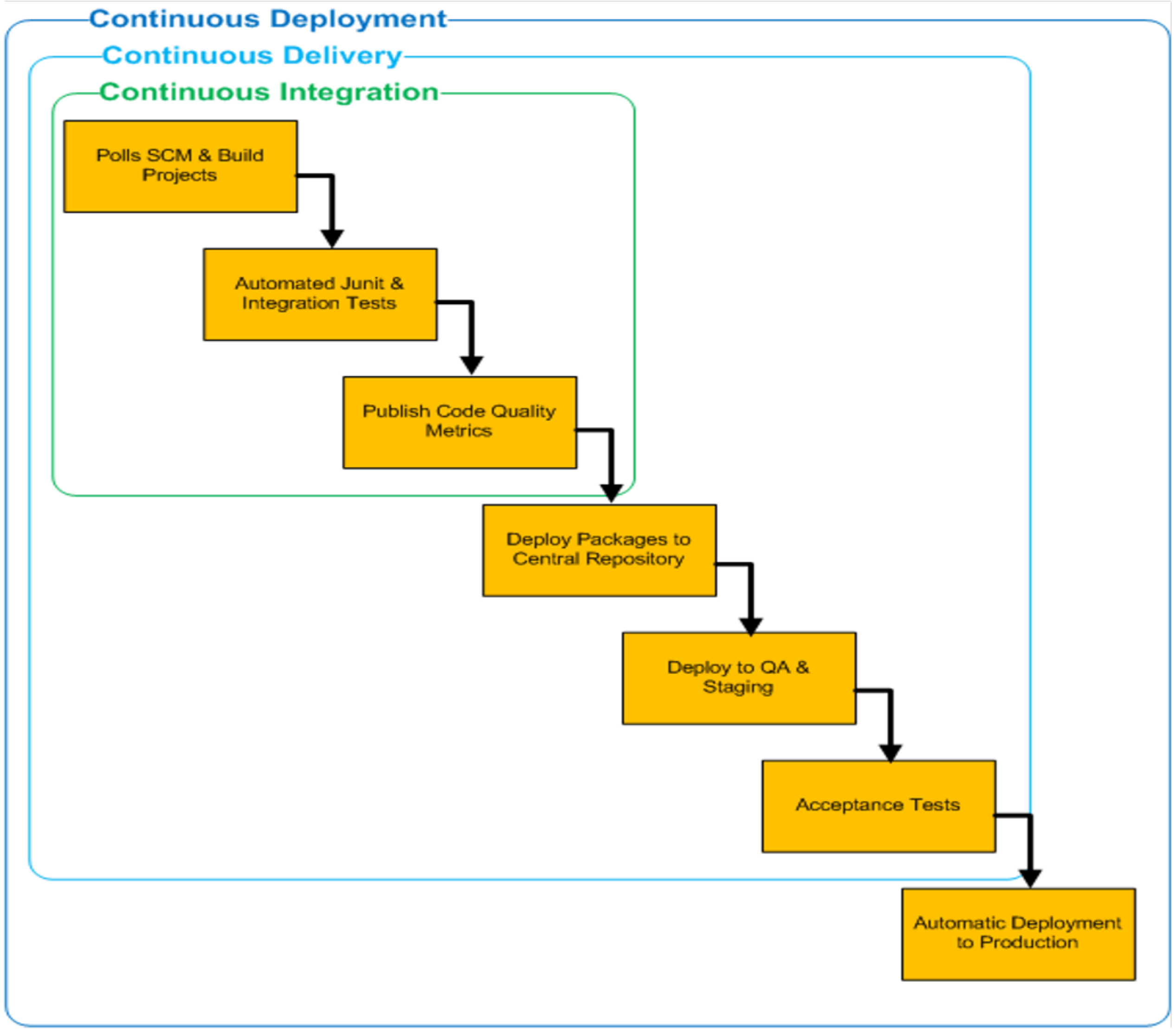
Continuous Delivery

- Continuous Delivery is a **natural extension** of Continuous Integration
  - Every change to the system has passed all the relevant automated tests and is ready to deploy in production
  - Team can release any version at the push of a button
  - But the deployment to production is **not automatic**
- The goal of CD is to put business owners in the control of scheduling of the software releases
- Continuous Delivery is a core principle of **DevOps**

Continuous Deployment

- Continuous Development adds **automatic deployment to end users** in the Continuous Delivery process
- Continuous Deployment automatically deploys every successful build directly into production
  - Deploying the build to production as soon it passes the automated and UAT tests
- Continuous Deployment is not appropriate for many business scenarios
  - Business Owners prefer more predictable release cycles as opposed to arbitrary deployments

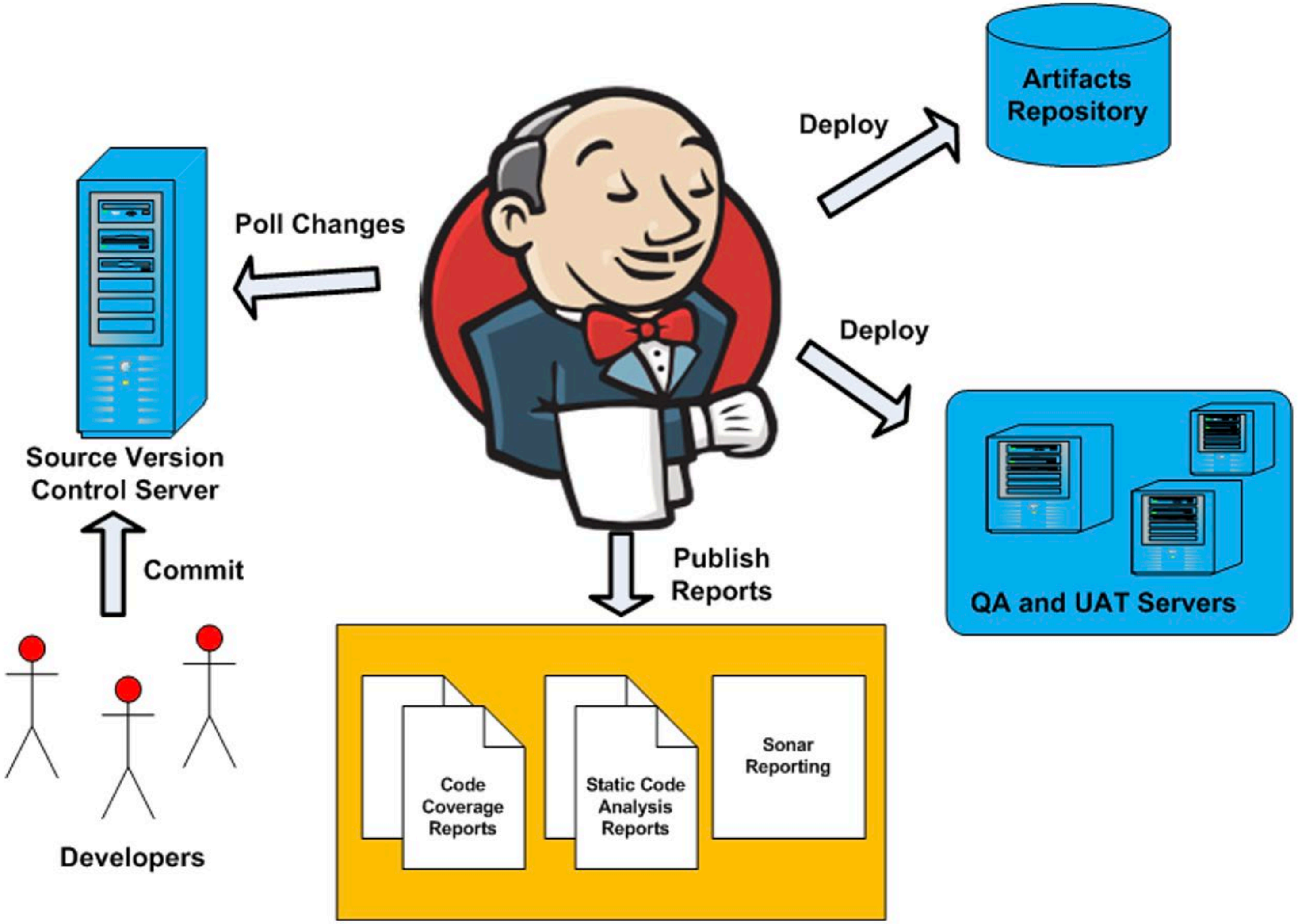
Continuous Integration, Delivery and Deployment


Continuous Deployment
Continuous Delivery
Continuous Integration
Polls SCM & Build Projects
Automated Junit & Integration Tests
Publish Code Quality Metrics
Deploy Packages to Central Repository
Deploy to QA & Staging
Acceptance Tests
Automatic Deployment to Production

Jenkins

- **Continuous Integration (CI) Server**
- Open Source, free and written in Java
- Large and dynamic community with massive adoption
- Easy to install on many different platforms
- Easy to use with a user-friendly interface
- Significant amounts of resources and tutorials available
- More than 1000 plugins
- New plugins released every week
- Extensible architecture – easy to extend and customize
- Distributed builds
- Many more features...

How Jenkins Fits in CI and CD



Jenkins - History

- Jenkins is derived from **Hudson**
- Hudson was first released by **Kohsuke Kawaguchi** of Sun Microsystems in 2005
- Initially it was only used within Sun but by 2010 Hudson captured 70% of CI market share
- Oracle bought Sun Microsystems in 2010
- Due to naming and an open-source dispute, original Hudson team forked Jenkins forked from Hudson as a new project
- Oracle continued the development of original Hudson
- Majority of Hudson users **migrated to Jenkins** within few months of the initial Jenkins release

State of the Jenkins Community

- Largest installed base of any open-source continuous integration and delivery platform
- More than **100K active users** in open-source Jenkins CI project
- Community contributed with more than **1000 plugins**
- Over 1000+ public repositories on GitHub and strong commit activity
- Quick feedback with addressing bugs and issues
- Get answer on any questions from Jenkins user mailing list and StackOverflow
  - Chances are other people have had your question and may have a solution
- Learn more about Jenkins at http://jenkins-ci.org/

Jenkins Setup

Lab One

End of Module