BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

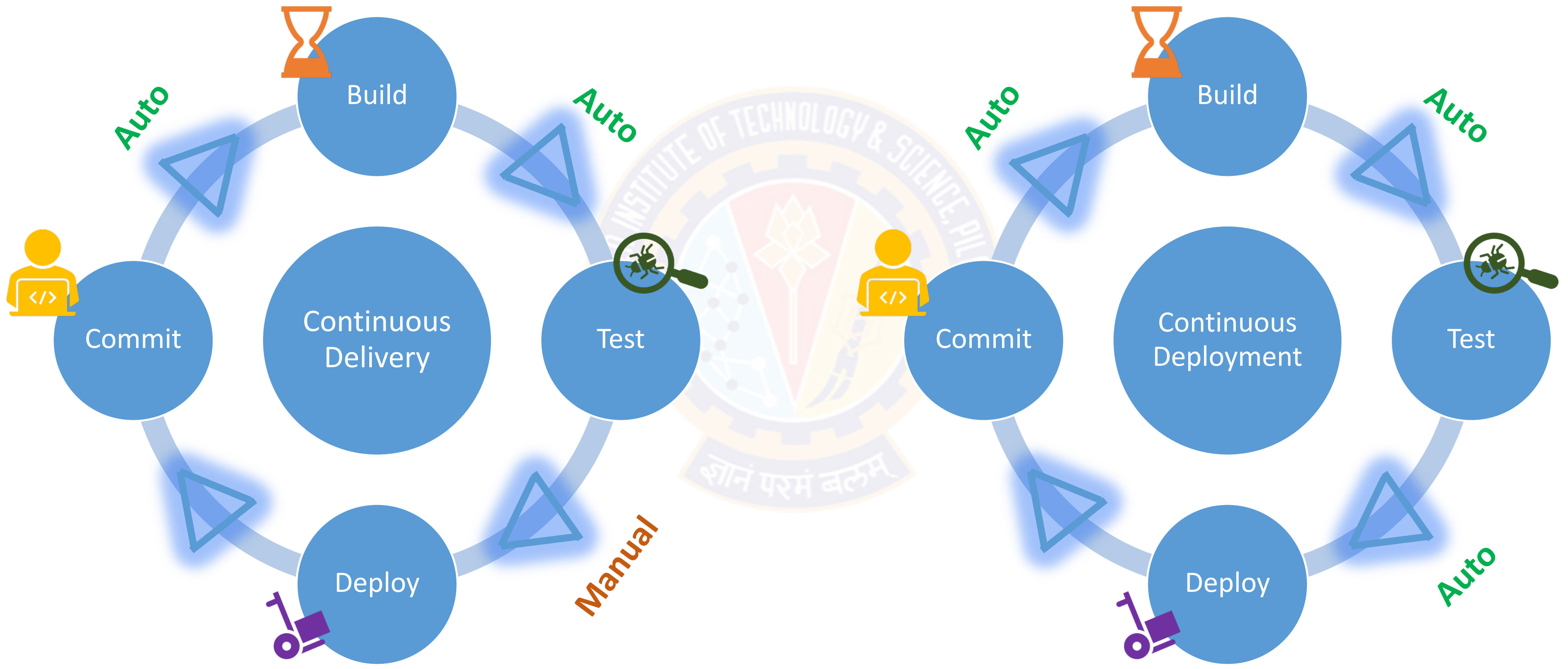
Agenda

Continuous Deployment

- Introduction to Deployment
- Deployment Consideration
- Challenges of Deployment
- Deployment pipeline
- Structure of Deployment Pipeline
- Basic Deployment Pipeline
- Stages of Deployment Pipeline

Continuous Deployment

Introduction to CD



Deployment

Considerations

Deployment Consideration

Compatibility

Backward

Forward

Canary Testing

Rollback

---> roll forward

Tools

Deployment

Challenges in Deployment

- Deployment Process Waste

  Build and operations teams waiting for documentation or fixes

  Testers waiting for "good" builds of the software

  Development teams receiving bug reports weeks after the team has moved on to new functionality

  Discovering, towards the end of the development process, that the application's architecture will not support the system's nonfunctional requirements

This leads to software that is undeployable because it has taken so long to get it into a production-like environment, and buggy because the feedback cycle between the development team and the testing and operations team is so long

Deployment

New Approach

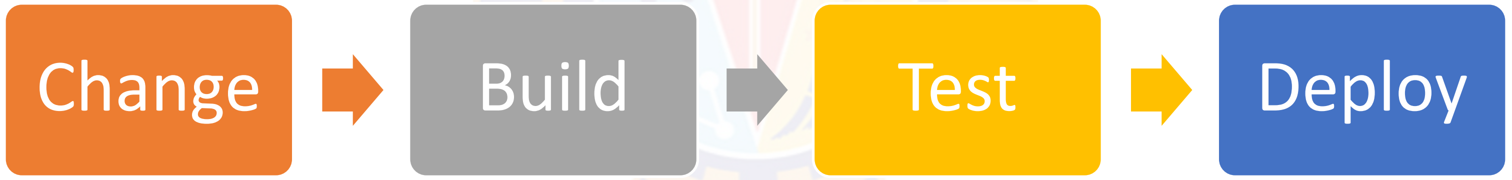
- End-to-End approach to delivering software
- Deployment of Application should be easy & one click to go
- A powerful feedback loop; rapid feedback on both the code and the deployment process
- Lowering the risk of a release and transferring knowledge of the deployment process to the development team
- Testing teams deploy builds into testing environments themselves, at the push of a button
- Operations can deploy builds into staging and production environments at the push of a button
- Developers can see which builds have been through which stages in the release process, and what problems were found
- Automate Build, Deploy, Test and Release System

Note: As a result, everybody in the delivery process gets two things: access to the things they need when they need them, and visibility into the release process to improve feedback so that bottlenecks can be identified, optimized, and removed.
This leads to a delivery process which is not only faster but also safer

Deployment Pipeline

What is Deployment Pipeline

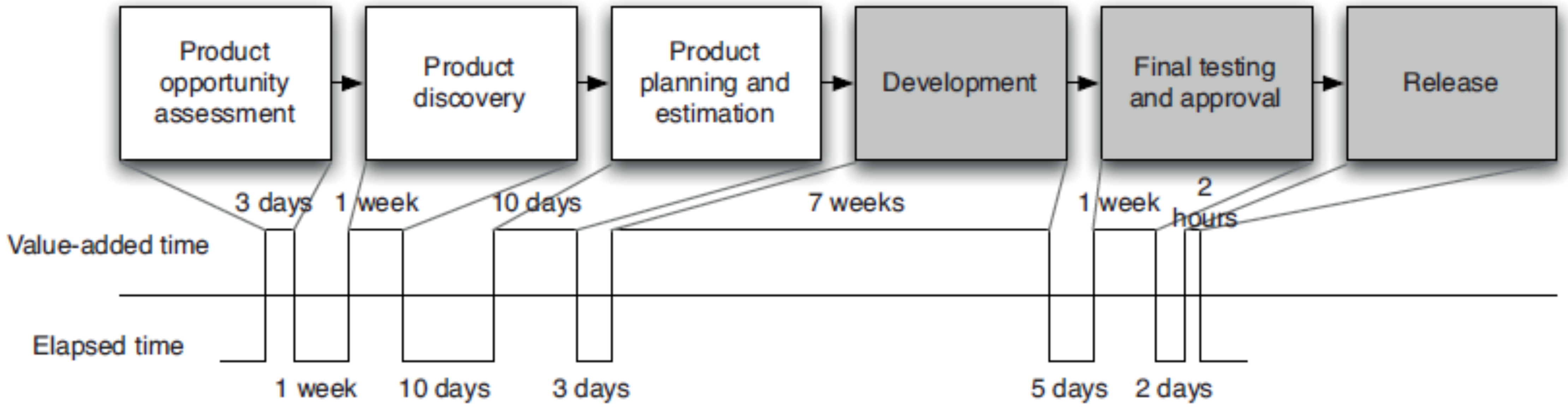
- An automated manifestation of your process for getting software from version control into the production



Increase the collaboration between many individuals

Deployment Pipeline

Value Stream Map of a Product Creation



A high-level value stream map for the creation of a new product

Cycle Time	: 11 Weeks,3 Days & 2 hours
Waste Time	: 5 Weeks
Lead Time	: 16 Weeks,3 Days & 2 hours

Structure of Deployment Pipeline

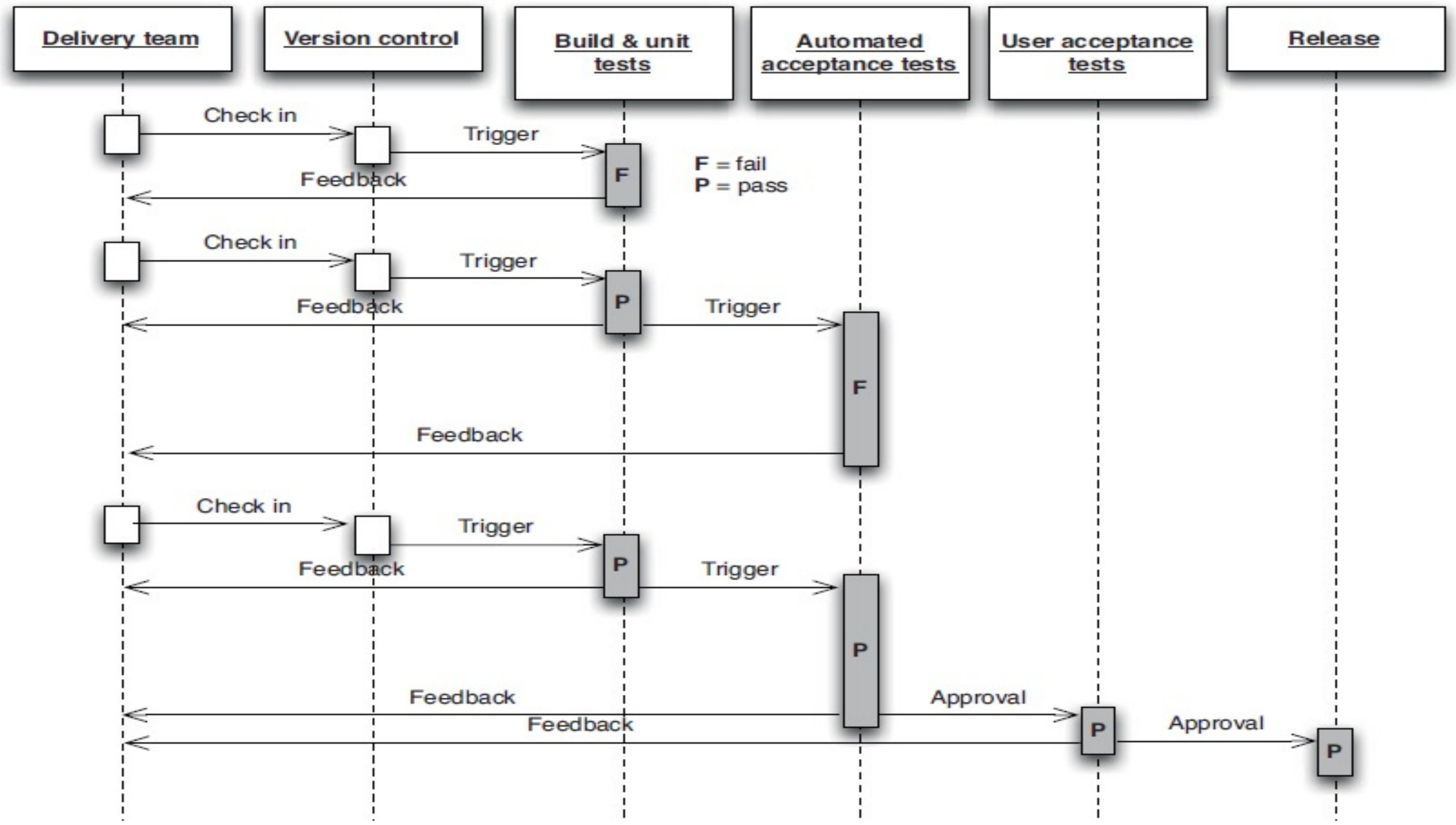
Expected steps to be followed

1. The input to the pipeline is a particular revision in version control
2. Every change creates a build
3. It pass through a sequence of tests and challenges to, its viability as a production release

- As the build passes each test of its fitness, confidence in it increases
- The objective is to eliminate unfit release candidates as early in the process as we can and get feedback on the root cause of failure to the team as rapidly as possible
- Any build that fails a stage in the process will not generally be promoted to the next

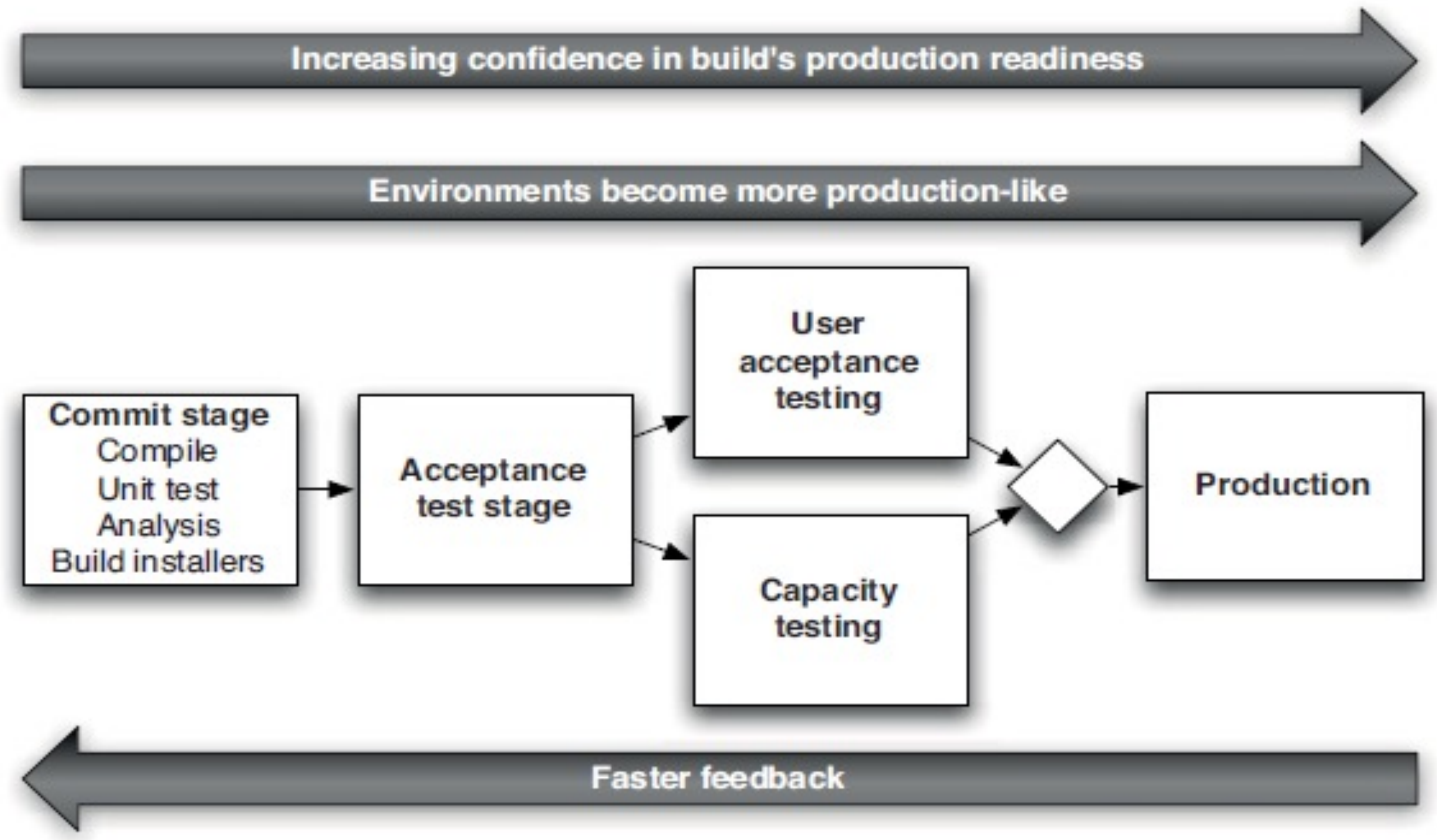
Structure of Deployment Pipeline

Example



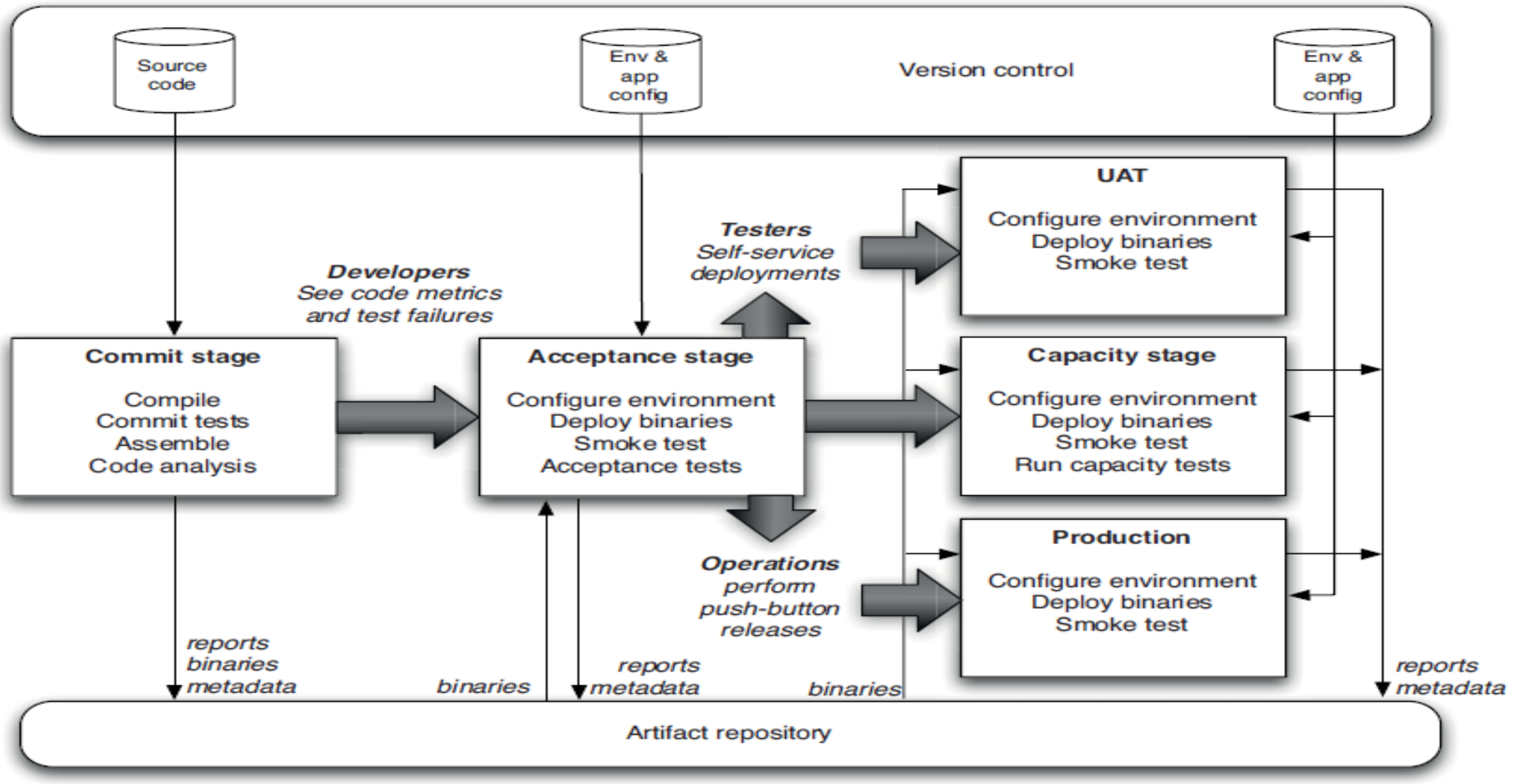
Structure of Deployment Pipeline

Broad Overview



Deployment Pipeline

Basic Deployment Pipeline



Basic Deployment Pipeline

Outcome

- The ultimate purpose of all this is to get feedback as fast as possible
- To make the feedback cycle fast
- To be able to see which build is deployed into which environment and
- Which stages in your pipeline each build has passed

Note: It means that if you see a problem in the acceptance tests (for example), you can immediately find out which changes were checked into version control that resulted in the acceptance tests failing

Deployment Pipeline

Antipatterns of dealing with Binaries

- The source code will be compiled repeatedly in different contexts: during the commit process, again at acceptance test time, again for capacity testing, and often once for each separate deployment target

- Every time you compile the code, you run the risk of introducing some difference
- The version of the compiler installed in the later stages may be different from the version that you used for your commit tests
- You may pick up a different version of some third-party library that you didn't intend
- Even the configuration of the compiler may change the behavior of the application

Antipatterns of dealing with Binaries

Violates two Principles

Efficiency of Deployment pipeline

- Recompiling violates this principle because it takes time, especially in large systems
- Delayed feedback

Always build upon foundations

- The binaries that get deployed into production should be exactly the same as those that went through the acceptance test process
- Recreation of binaries violates this principle

Deployment Pipeline

Deployment Pipeline Practices

1. Only Build Your Binaries Once
2. Deploy the Same Way to Every Environment

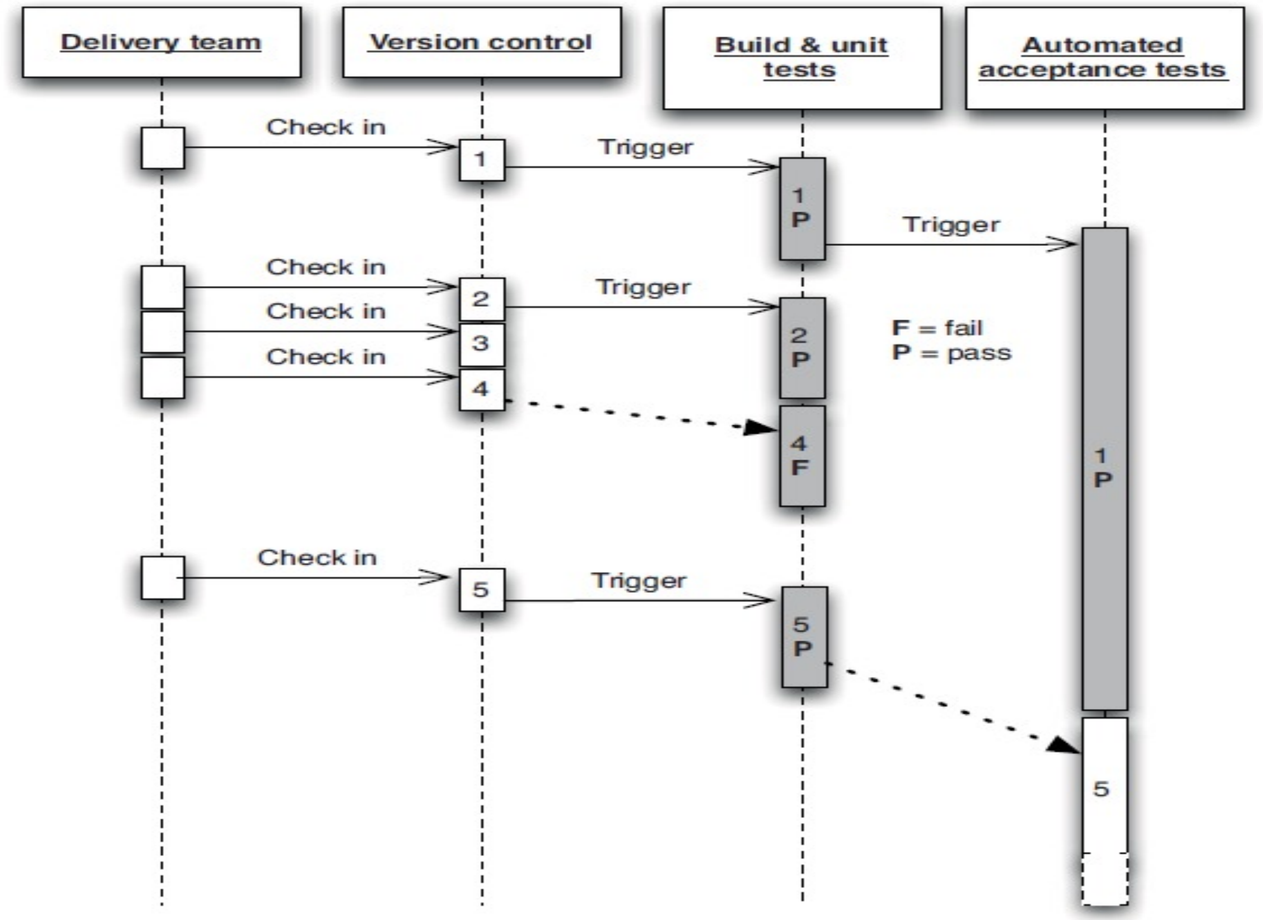
   Note: Using the same script to deploy to production that you use to deploy to development environments is a fantastic way to prevent the "it works on my machine" syndrome

3. Smoke-Test Your Deployments
4. Deploy into a Copy of Production
5. Each Change Should Propagate through the Pipeline Instantly
6. Intelligent pipeline scheduling
7. If Any Part of the Pipeline Fails, Stop the Line

Deployment Pipeline Practices

Intelligent scheduling

- Intelligent scheduling is crucial to implementing a deployment pipeline



Deployment Pipeline

Stages in Deployment Pipeline

- The Commit Stage
- The Automated Acceptance Gate
- Subsequent Test Stages
- Preparing to Release

Preparing to Release

Release Plan

- Have a release plan that is created and maintained by everybody involved in delivering the software, including developers and testers, as well as operations, infrastructure, and support personnel
- Minimize the effect of people making mistakes by automating as much of the process as possible
- Practice the procedure often in production-like environments, so you can debug the process and the technology supporting it
- Have the ability to back out a release if things don't go according to plan
- Have a strategy for migrating configuration and production data as part of the upgrade and rollback processes

Note: Goal is a completely automated release process, Releasing should be as simple as choosing a version of the application to release and pressing a button and Backing out should be just as simple

Q&A

Thank You!

In our next session: