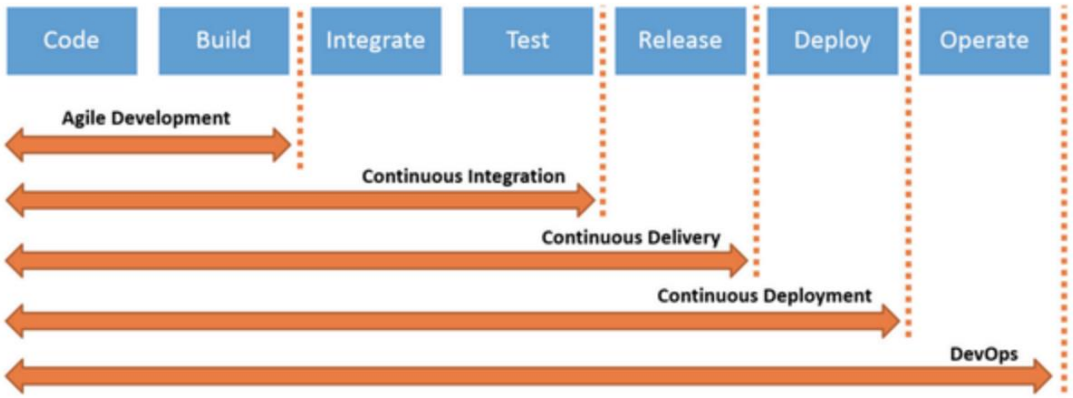
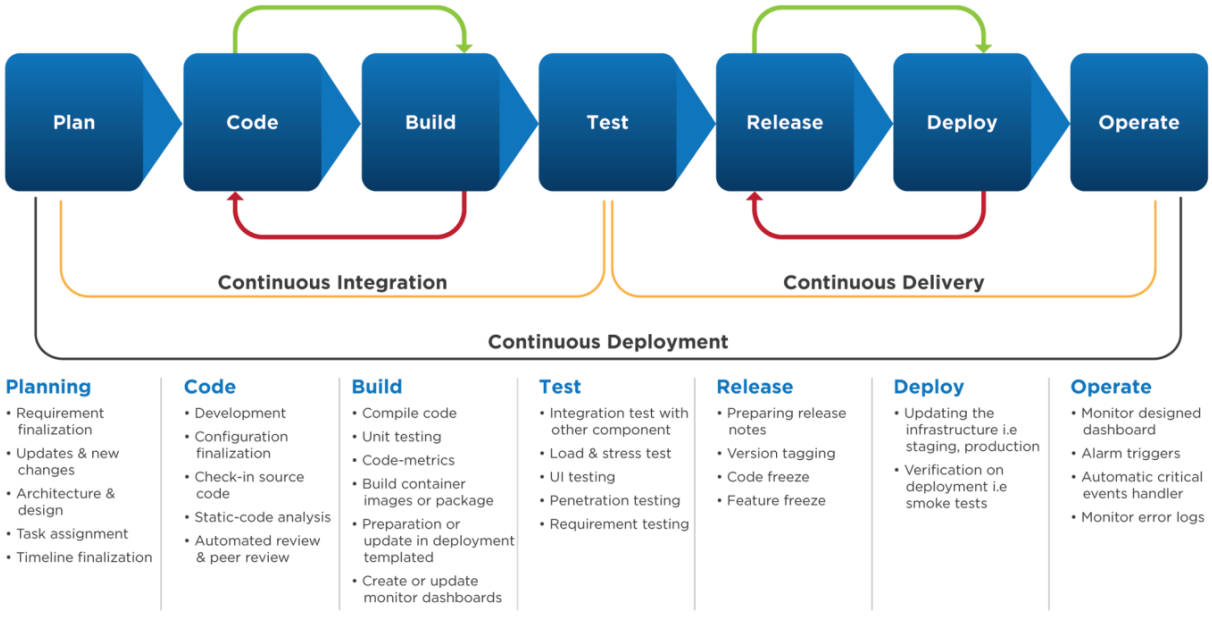
CICD Short Notes

what is Devops?



- DevOps is a software development approach which involves continuous integration, continuous testing, Continuous delivery, Continuous deployment, and continuous monitoring of the software throughout its development lifecycle.
- This is the process adopted by all the top companies to develop high-quality software and shorter development lifecycles, resulting in greater customer satisfaction, something that every company wants.

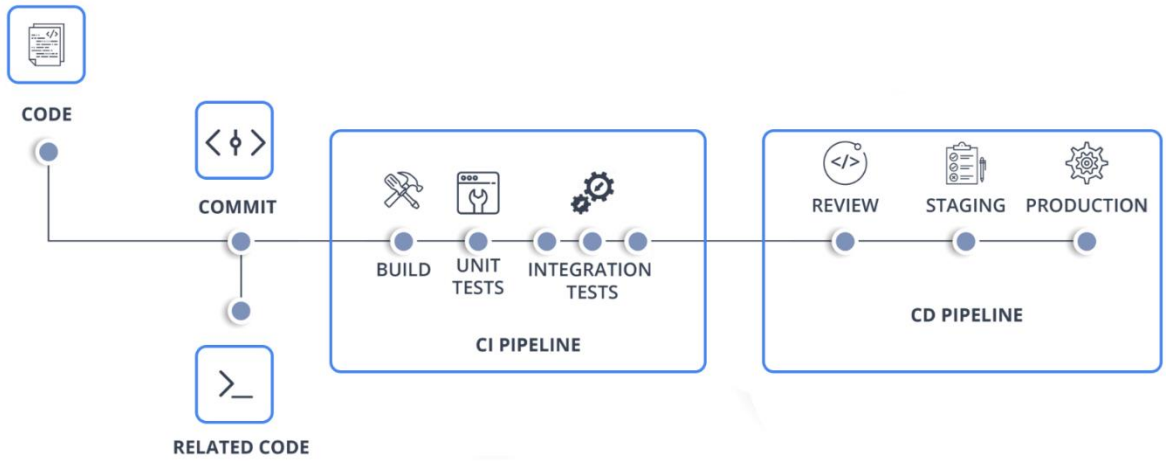
Various DevOps lifecycle stages.



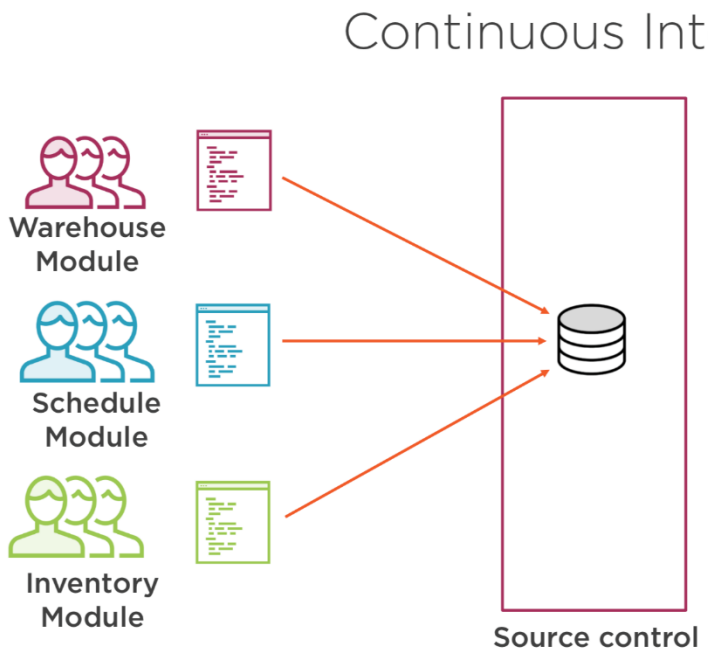
- **Plan.** This phase helps define business value and requirements. Sample tools include Jira or Git to help track known issues and perform project management.

- **Code.** This phase involves software design and the creation of software code. Sample tools include GitHub, GitLab, Bitbucket, or Stash.
- **Build.** In this phase, you manage software builds and versions, and use automated tools to help compile and package code for future release to production. You use source code repositories or package repositories that also "package" infrastructure needed for product release. Sample tools include Docker, Ansible, Puppet, Chef, Gradle, Maven, or JFrog Artifactory.
- **Test.** This phase involves continuous testing (manual or automated) to ensure optimal code quality. Sample tools include JUnit, Codeception, Selenium, Vagrant, TestNG, or Blaze Meter.
- **Deploy.** This phase can include tools that help manage, coordinate, schedule, and automate product releases into production. Sample tools include Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenStack, Docker, or Jira.
- **Operate.** This phase manages software during production. Sample tools include Ansible, Puppet, PowerShell, Chef, Salt, or Otter.
- **Monitor.** This phase involves identifying and collecting information about issues from a specific software release in production. Sample tools include New Relic, Datadog, Grafana, Wireshark, Splunk, Nagios, or Slack.

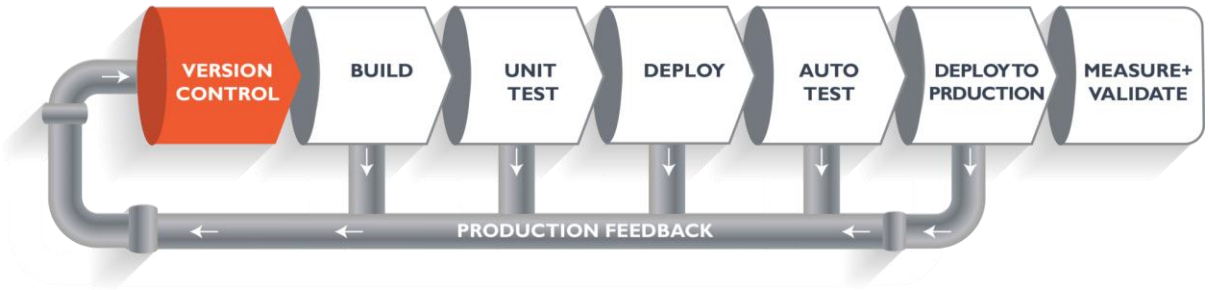
CI/CD Workflow Pipeline



SCM and the CI/CD Pipeline



SCM

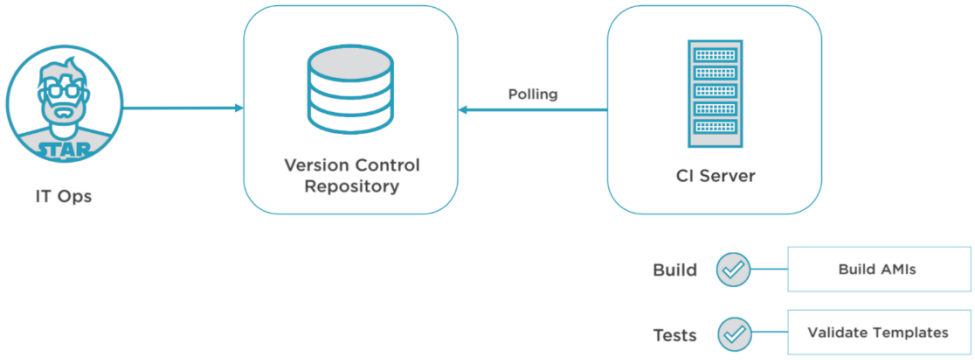


Source Control Management (SCM) is an important component of CI/CD Pipeline.

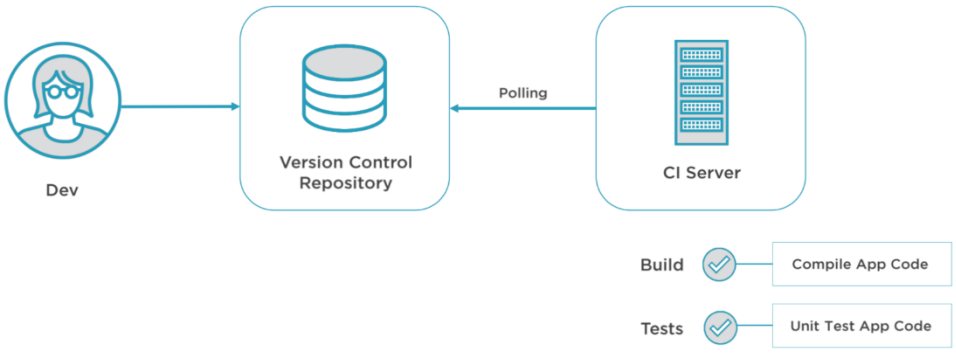
- It is a huge part of the daily developer's workflow:
- All Code changes made by developers will need to be tracked in SCM
- Developers will use SCM to track their changes separately and then merge them together
- All pieces of automation that need to interact with the source code will use SCM:
- Continuous Integration (CI) will get the code from SCM
- SCM will notify the CI server when code needs to be built

How CI Works

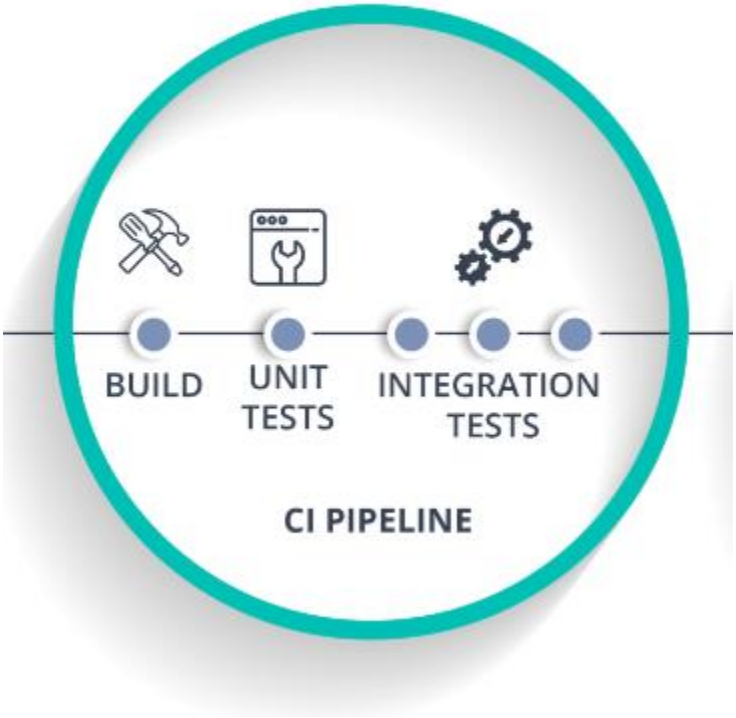
Continuous Integration (CI)



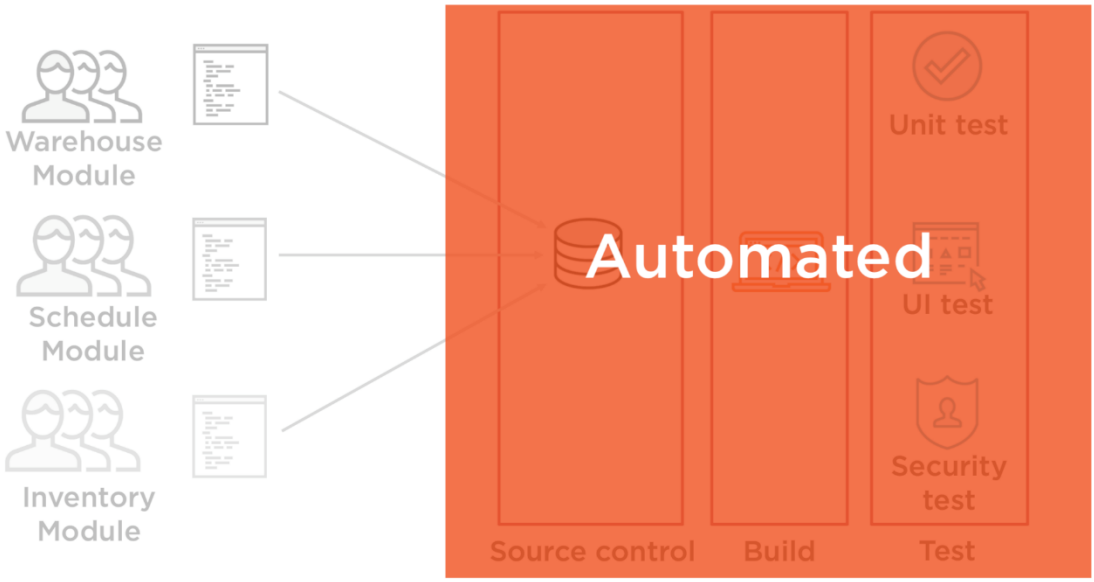
Continuous Integration (CI)



Build Pipeline / CI Pipeline



Continuous Integration



CI principles

Have a single place where all the code lives

Everyone commits to the mainline every day

Automate the build process

- Fix broken build immediately
- Make and keep the build fast

Every commit triggers a build

Automate the testing process

Everyone has access to the latest result

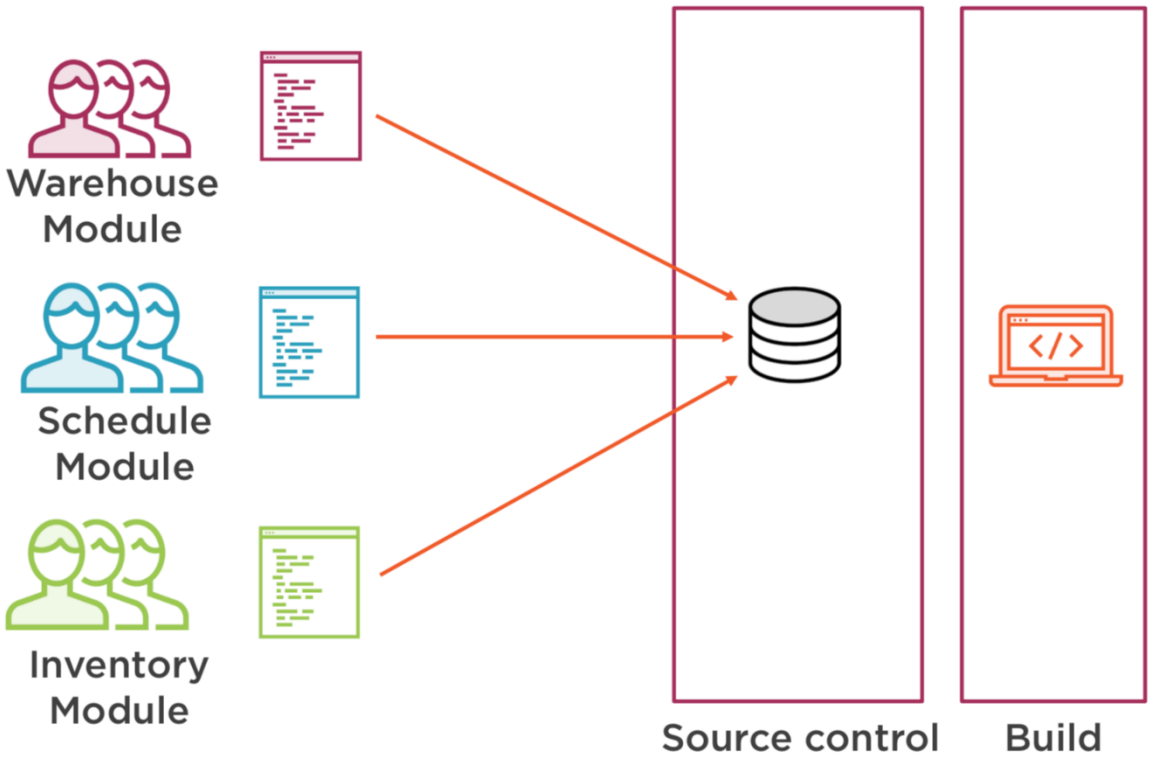
Everyone can see everything

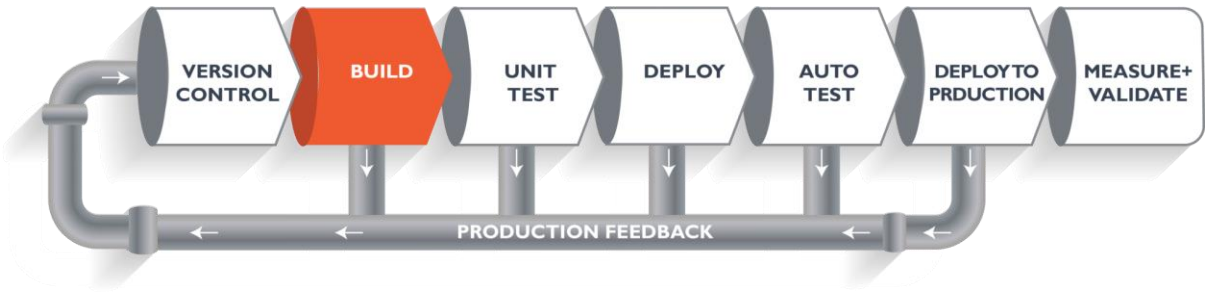
CI benefits

- Integration takes less effort
- Issues will come up more early
- Automation means less issues
- The process is more visible
- Improved team communication
- Short integration iterations means more flexibility
- The code is **ready to be delivered** more often

What Can CI Accomplish?

Build Phase / Build Automation


Warehouse Module
Schedule Module
Inventory Module
Source control
Build


VERSION CONTROL
BUILD
UNIT TEST
DEPLOY
AUTO TEST
DEPLOY TO PRDUCTION
MEASURE+ VALIDATE
PRODUCTION FEEDBACK

Build Stage

Build Application Artifacts

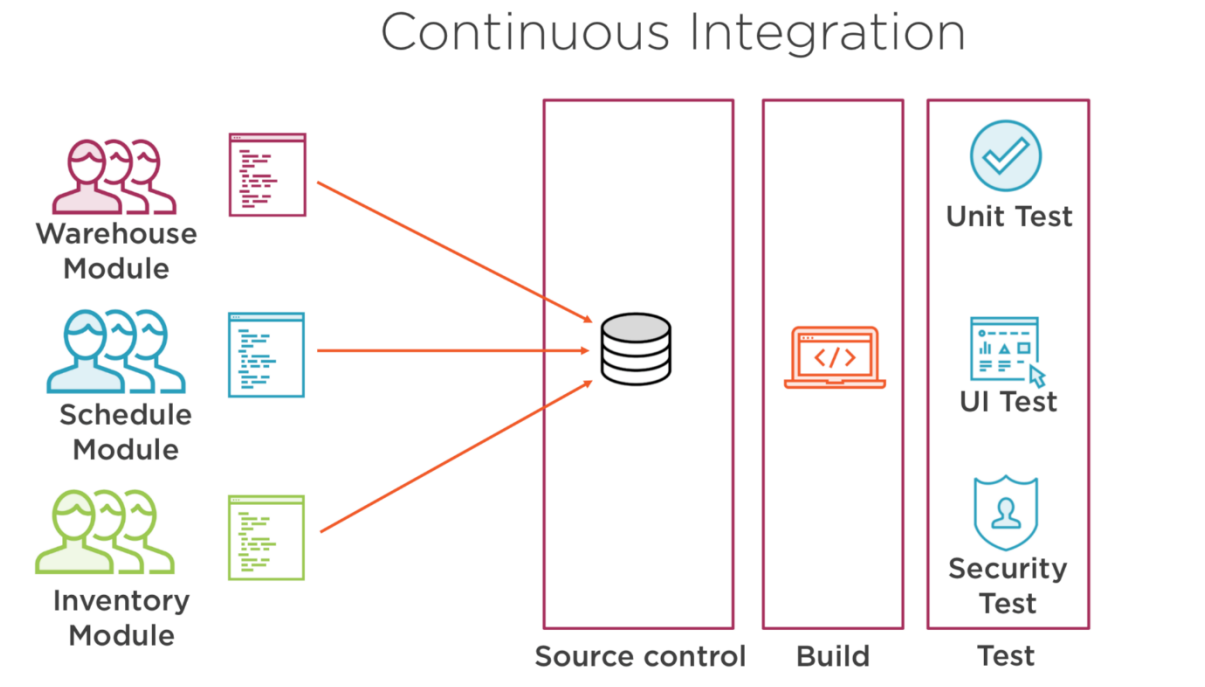
- e.g. Python Wheels
- e.g. Java JAR files

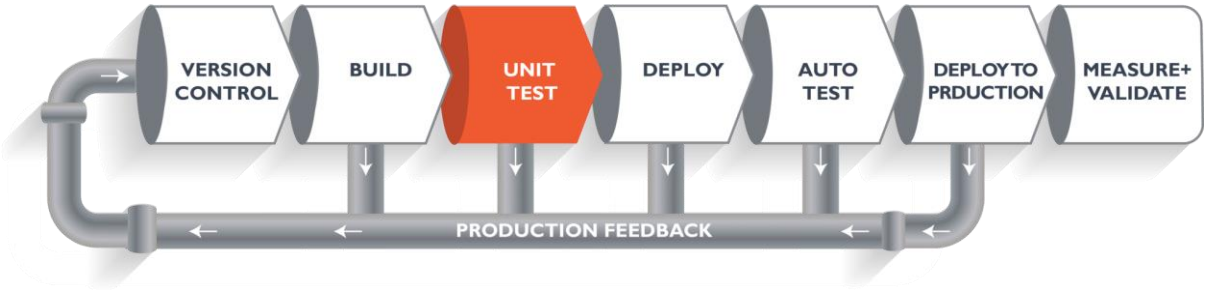
Must represent as tested application state

Creates a Deployable Artifact

- i.e. a built distribution
- Pre-compiled, pre-built
- Installation requires no development dependencies

Testing Phase / Automated Testing

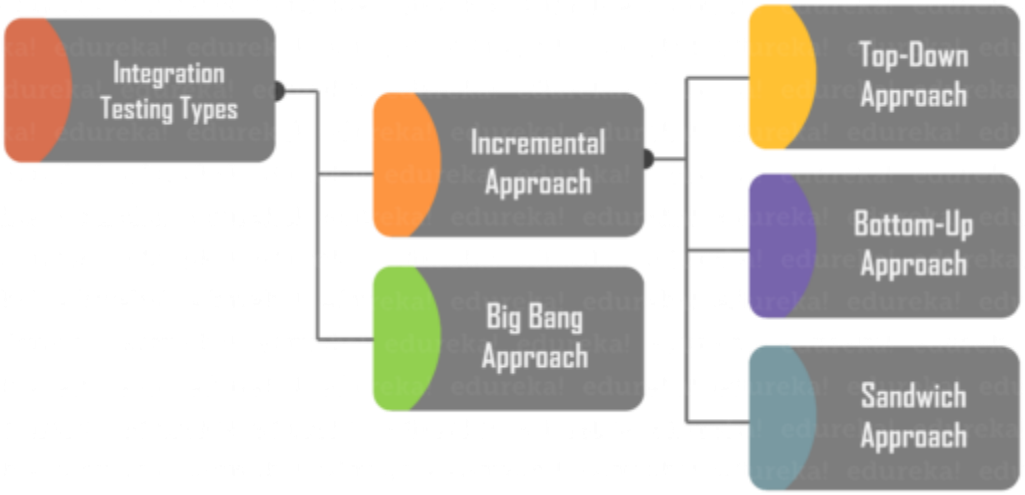


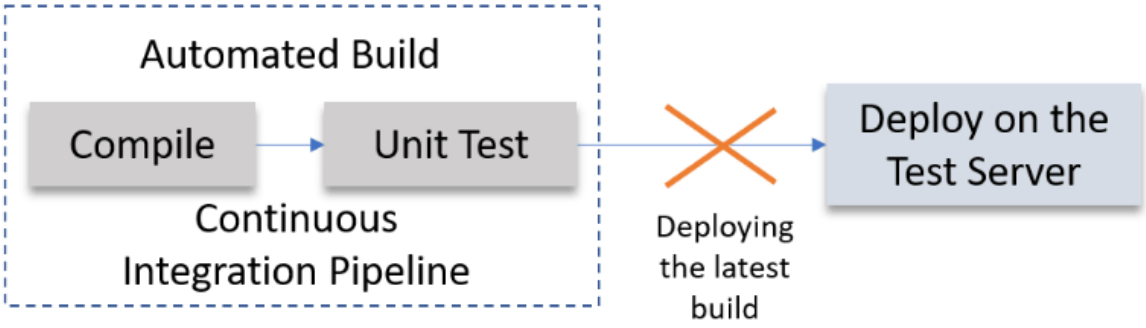


Testing types

1. Unit Testing
2. Integration Testing

- **Unit Testing:** It is the testing of an individual unit or group of related units. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.
- **Integration Testing:** It is a type of testing in which a group of components are combined to produce the output. Also, the interaction between software and hardware is tested if software and hardware components have any relation. It may fall under both white box testing and black box testing.


Integration Testing Types
Incremental Approach
Big Bang Approach
Top-Down Approach
Bottom-Up Approach
Sandwich Approach


Automated Build
Compile
Unit Test
Continuous Integration Pipeline
Deploying the latest build
Deploy on the Test Server

What is CD?

Continuous Delivery

Release Often

Release Faster

Greater Reliability

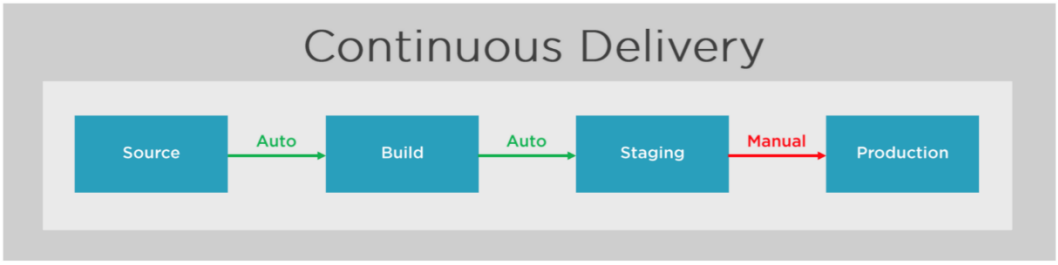
Difference Between Delivery and Deployment

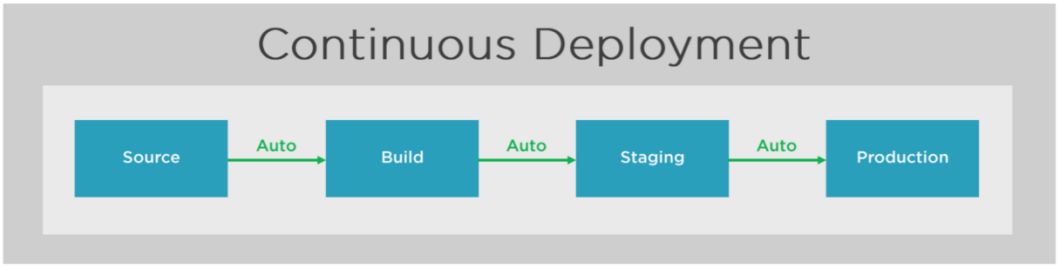
Continuous Delivery

Software **can** be deployed to production at any time

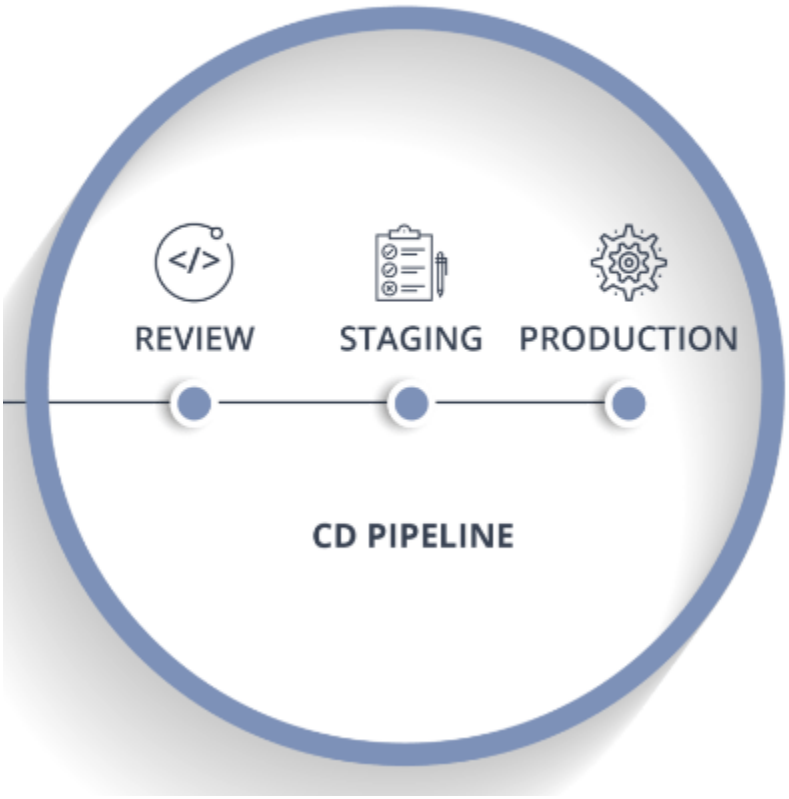
Continuous Deployment

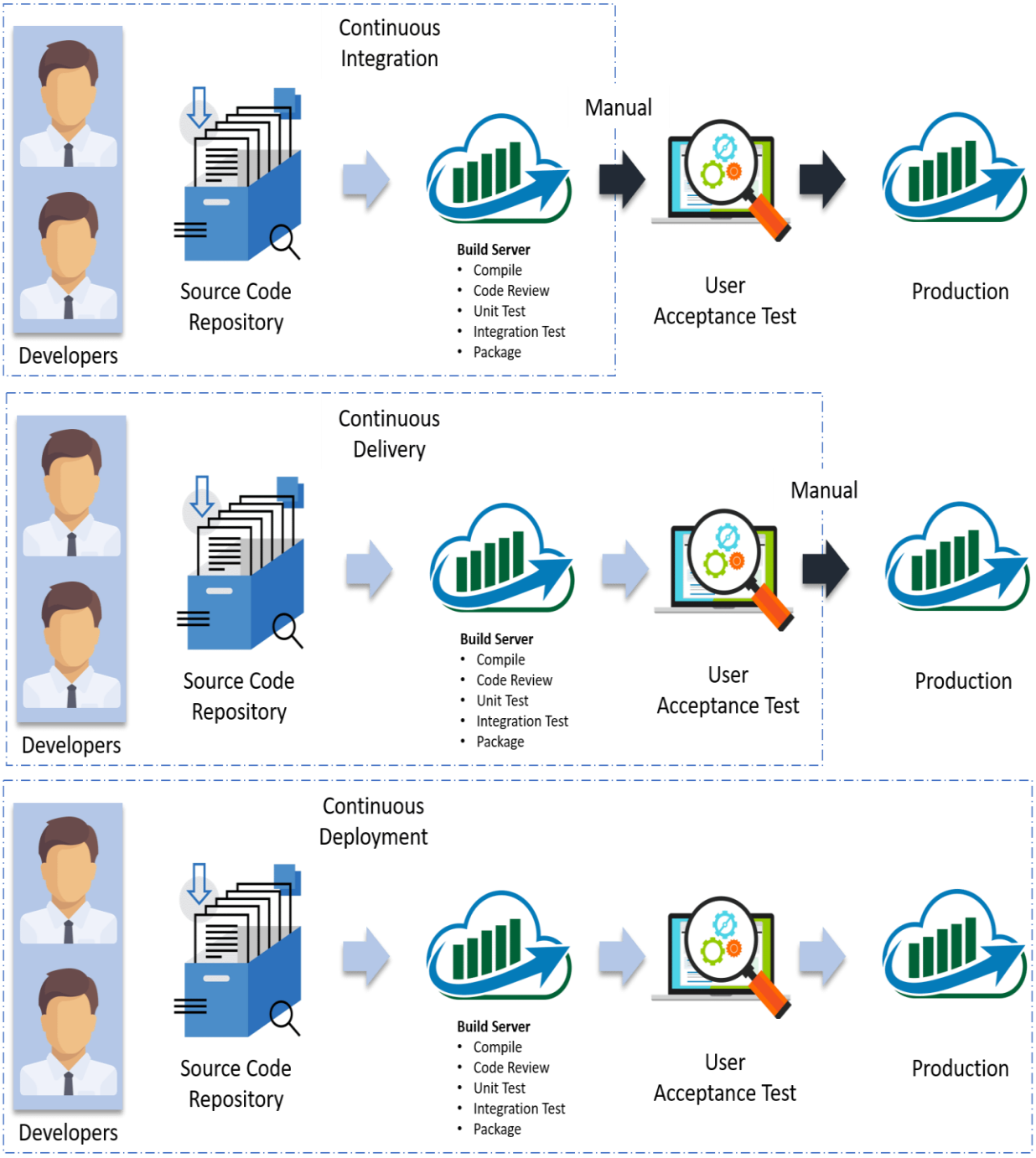
Software **is automatically** deployed to production all the time



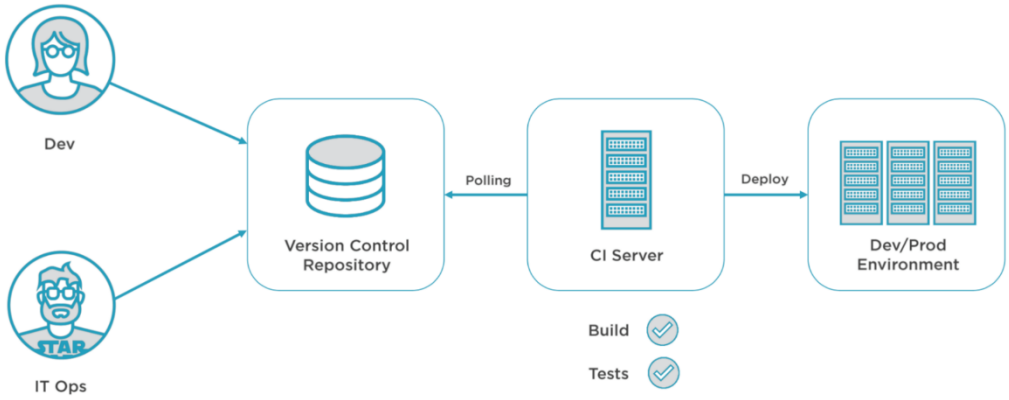


Release pipeline /CD Pipeline

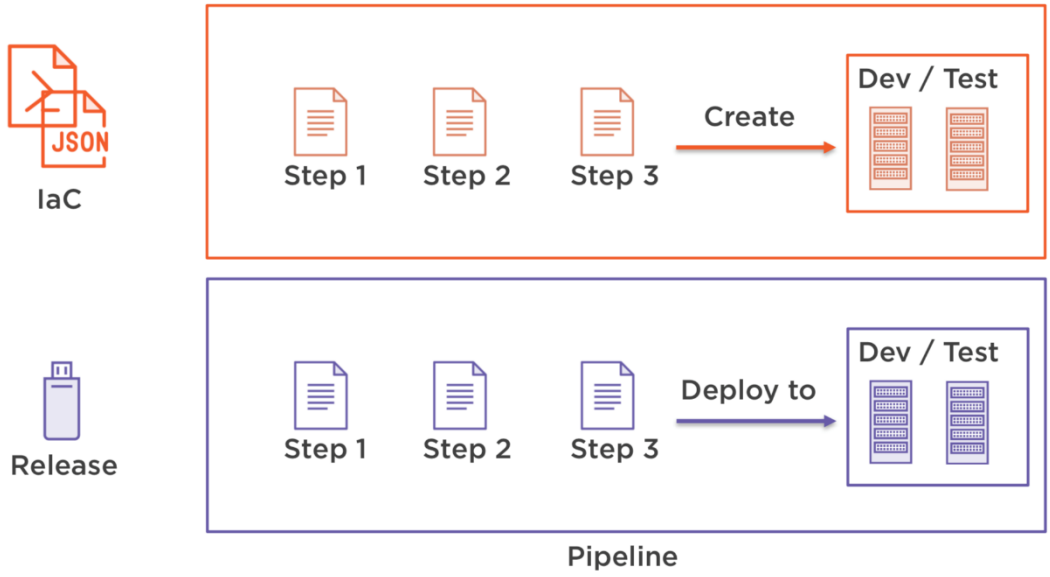


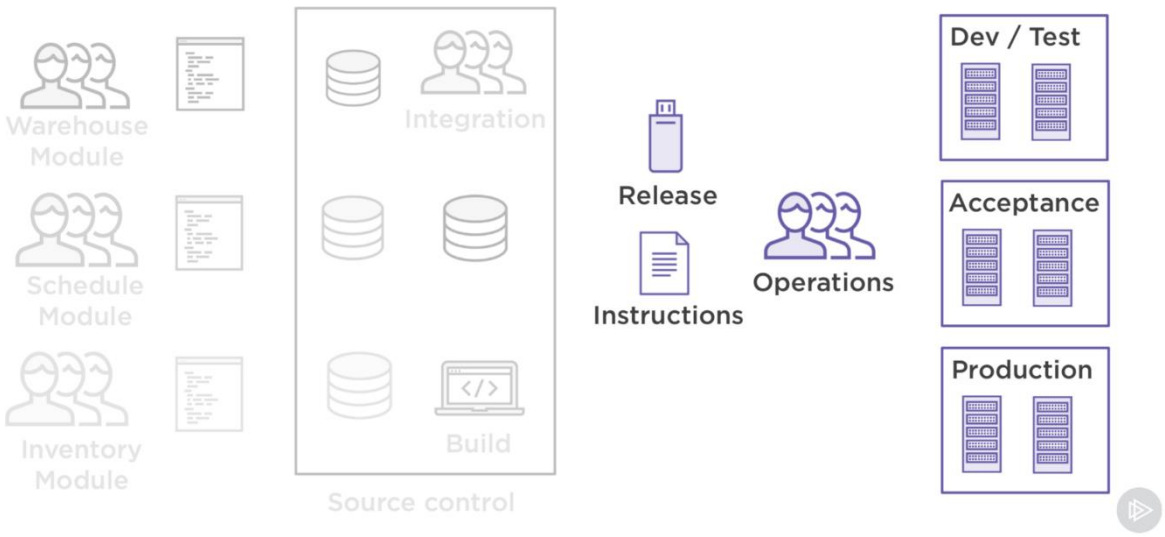

Continuous Integration
Developers
Source Code Repository
Build Server
• Compile
• Code Review
• Unit Test
• Integration Test
• Package
Manual
User Acceptance Test
Production
Continuous Delivery
Developers
Source Code Repository
Build Server
• Compile
• Code Review
• Unit Test
• Integration Test
• Package
Manual
User Acceptance Test
Production
Continuous Deployment
Developers
Source Code Repository
Build Server
• Compile
• Code Review
• Unit Test
• Integration Test
• Package
User Acceptance Test
Production

Continuous Delivery (CD)



Continuous Delivery





Release Stage

Build Docker Release Image

- Includes minimal runtime environment
- Installs application artifacts

Release Environment

- A production-like environment
- Use an external test runner to run acceptance tests

Tag and Publish

- Only if acceptance tests pass
- Docker Hub

Release

High chance of errors

Lots of effort in deployment

Difficult to reproduce deployment

Inconsistency in environments

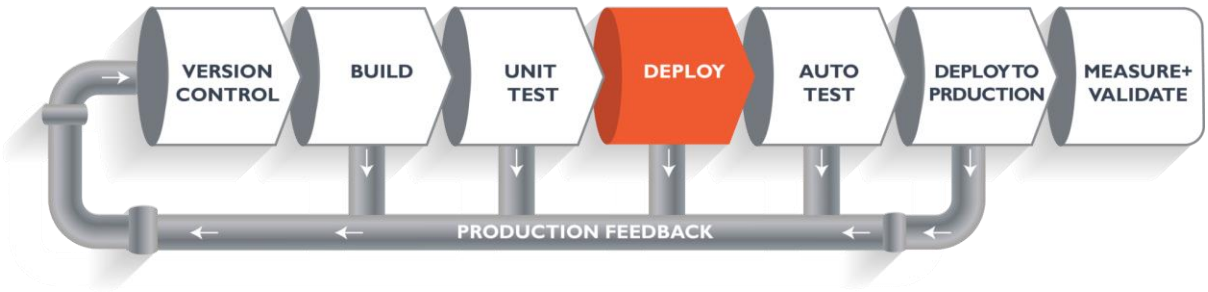
Deployments are scary

Slow delivery of functionality

Release

- Preparing release notes
- Version tagging
- Code freeze
- Feature freeze

Deploy Phase



Deploy Stage

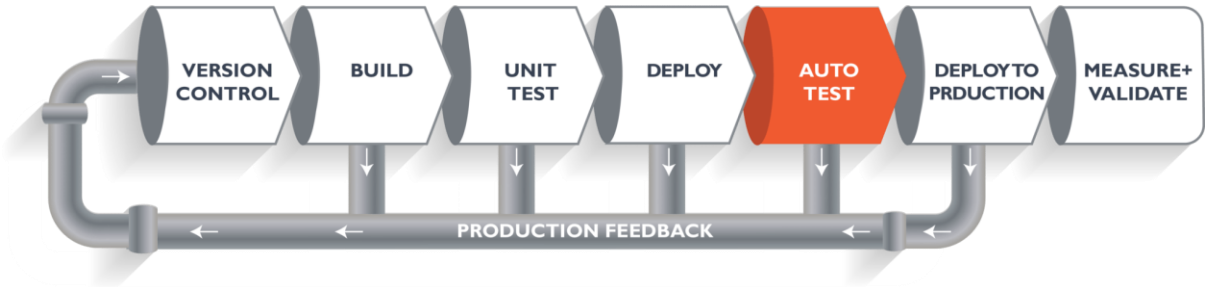
Deploy release image to:

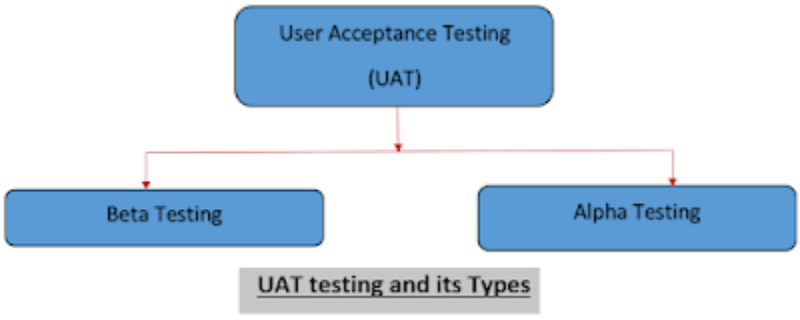
- At least one environment
- e.g. Development Environment
- e.g. QA or Staging Environments
- Perhaps even production

Fully automated deployments

- Using orchestration tools such as Ansible
- Leverage Infrastructure automation tools such as AWS CloudFormation

Auto Test Phase:
Once the code is deployed successfully, you can run another series of Acceptance Tests or Sanity test for the final check. If everything is accepted, then it can be deployed to production.





When to perform UAT Testing?

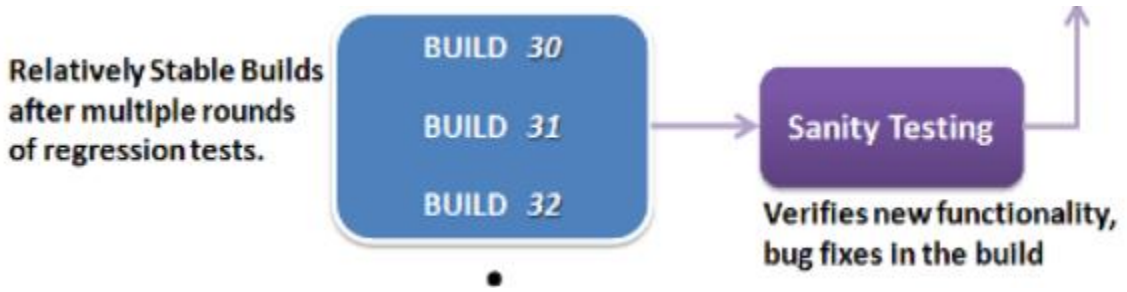
At the last stage before the software goes live or made available for users

UAT performs after Unit, integration, System testing.

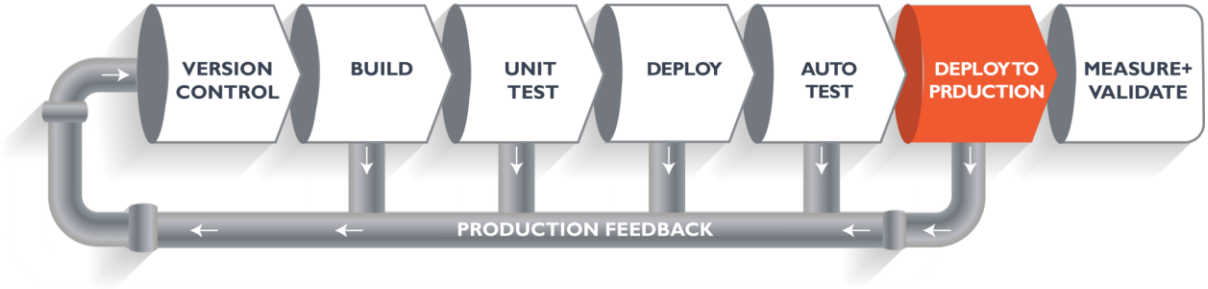
UAT testing is performed when all the measure bugs are fixed

UAT Testing is performed when fully integrated and bug fixed software is available.

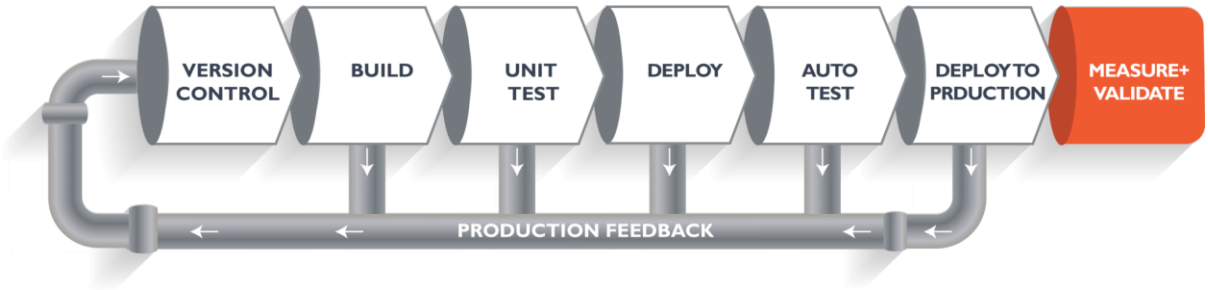
Sanity Test:



Deploy to Production:

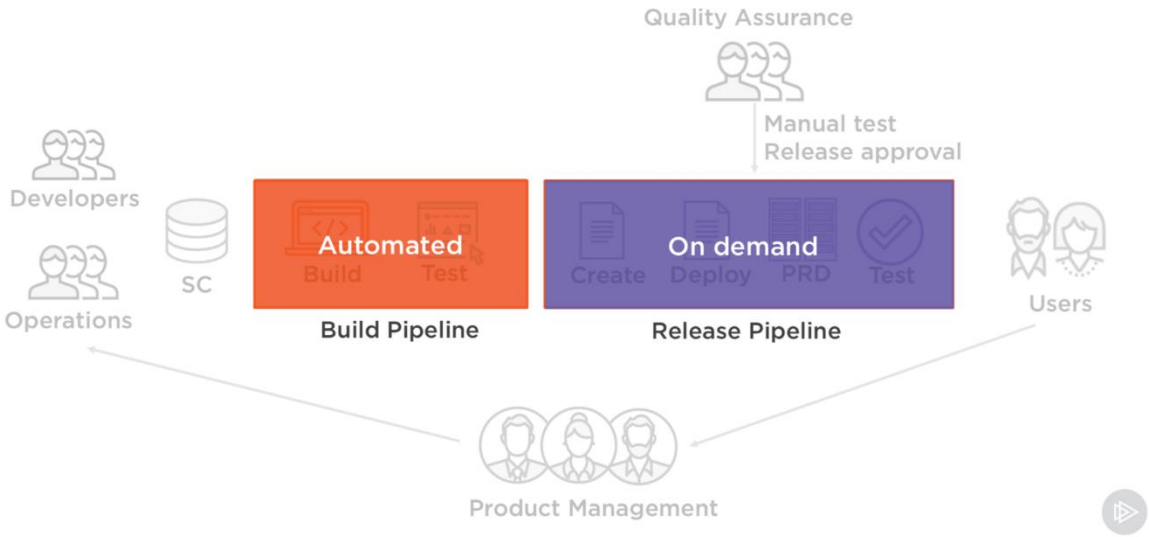


Measure & Validate:
The pipeline continues until we get a product or application which can be deployed in the production server where we measure and validate the code. Splunk, ELK Stack, Nagios etc. are some of the popular tools for monitoring.



This lifecycle continues until we get code/a product which can be deployed to the production server where we measure and validate the code.

Continuous Delivery



CD principles

- **Have continuous integration in place**
- **Development and Operations should work well together**
- **Treat infrastructure as a code artifact**
- **Automate the environment creation process**
- **Automate the release process**
  - Automate acceptance tests
- **Include release to definition of done**
- **Releasing should be on-demand**
- **Everyone has access to the latest result**
- **Everyone can see everything**

CD benefits

Releasing takes less effort

Releasing is more

- Reliable
- Repeatable

Put control of release in the hands of business

Release more often

Get feedback earlier

What Can CD Accomplish?

Higher Quality

Faster Delivery

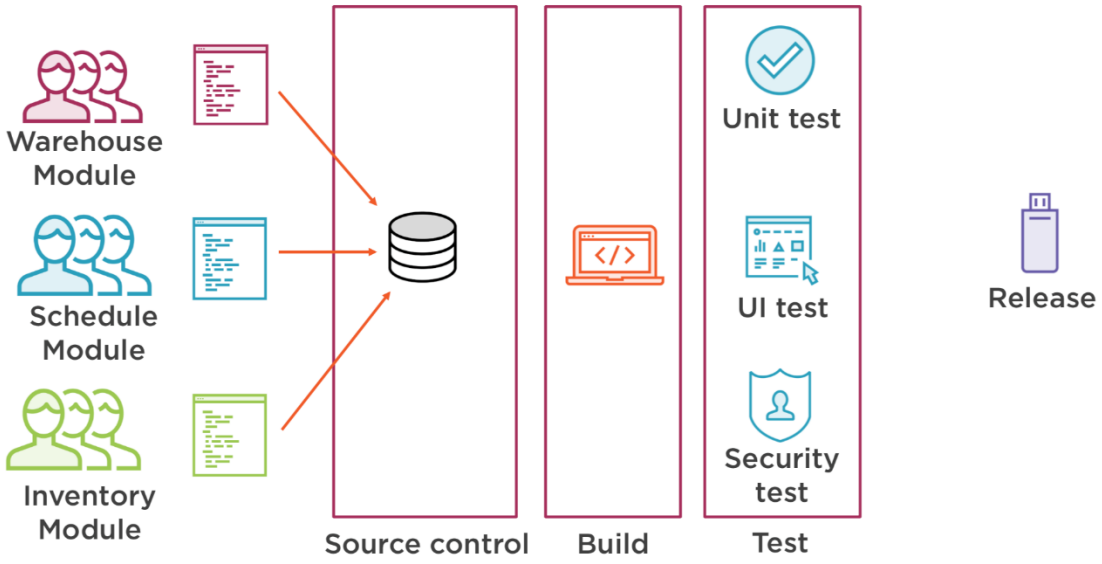
Lower Costs

More Flexibility

Continuous Integration & Continuous Deployment Principles

- Automate everything
- Define infrastructure as code
- Store application and infrastructure code in version control
- Unify the application and the infrastructure
- Perform end-to-end automated testing

Continuous Delivery



Continuous Delivery