Progress®

JumpStart to Continuous Integration & Continuous Delivery (CI/CD)

Manchester, UK

Radisson Blu Hotel

Thursday, June 15, 2023

Soumaya Eddahech

Speakers

Soumaya Eddahech

Principal Sales Engineer

Workshop’s Agenda

- 2:00 Workshop Overview
- 2:10 Introduction to the CI/CD
- 3:00 Lab1: Test & Build
- 3:50 Lab2: Use a CI Server
- 4:30 Deploy: Docker
- 4:45 Q&A

Progress®

Purpose

- Understand the elements of a CI/CD Pipeline and gain hands-on experience improving software delivery
- What steps are being covered?
  - ABLUnit Framework
  - OpenEdge DevOps Framework
  - Using a CI/CD Pipeline

Workshop Logistics

Wifi

- SSID: <Radisson_Guest>
- Password: <>

Windows Platform

- OpenEdge 12.2
- OpenEdge DevOps Framework 2.1
- Progress Developer Studio
- Git
- Jenkins

GitHub repositories

- https://github.com/SoumayaEddahech/cicd-starter
- https://github.com/SoumayaEddahech/cicd-workshop-2023

Introduction to the CI/CD

Before CI/CD

There is a BETTER way.

Progress®

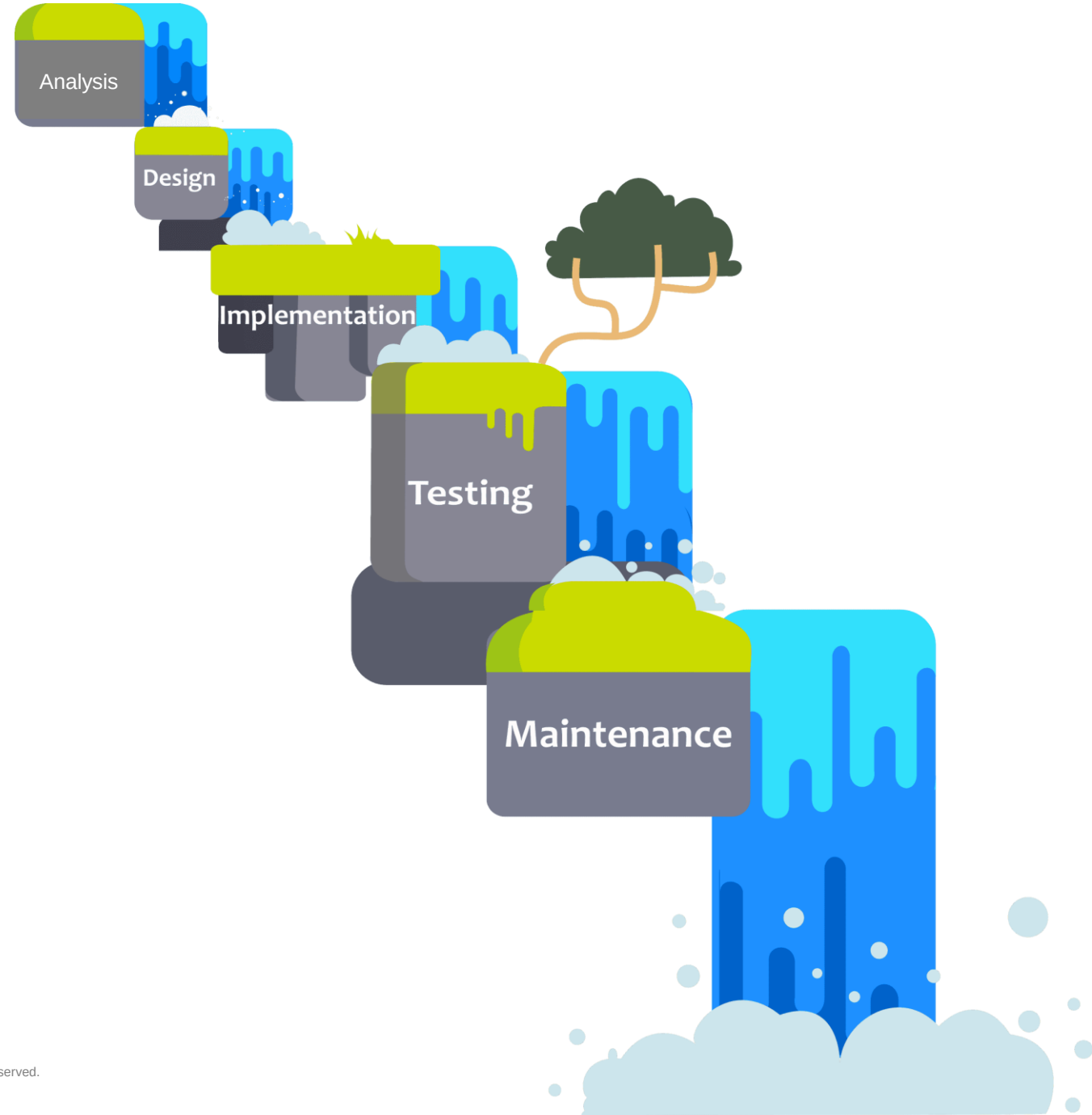
After CI/CD

Totally Automated and Low Risk!

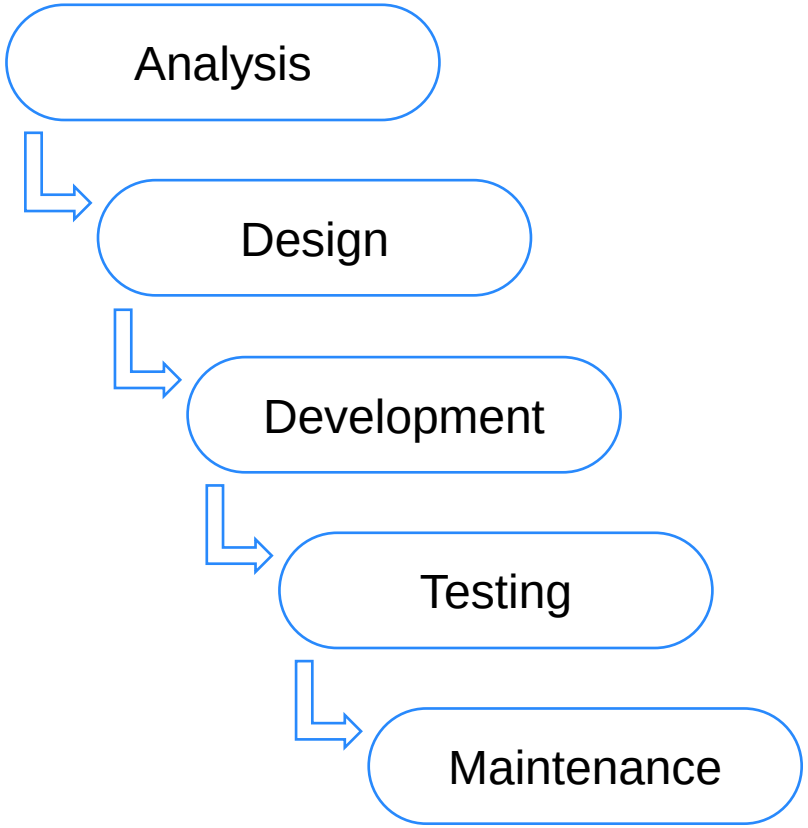
Progress®

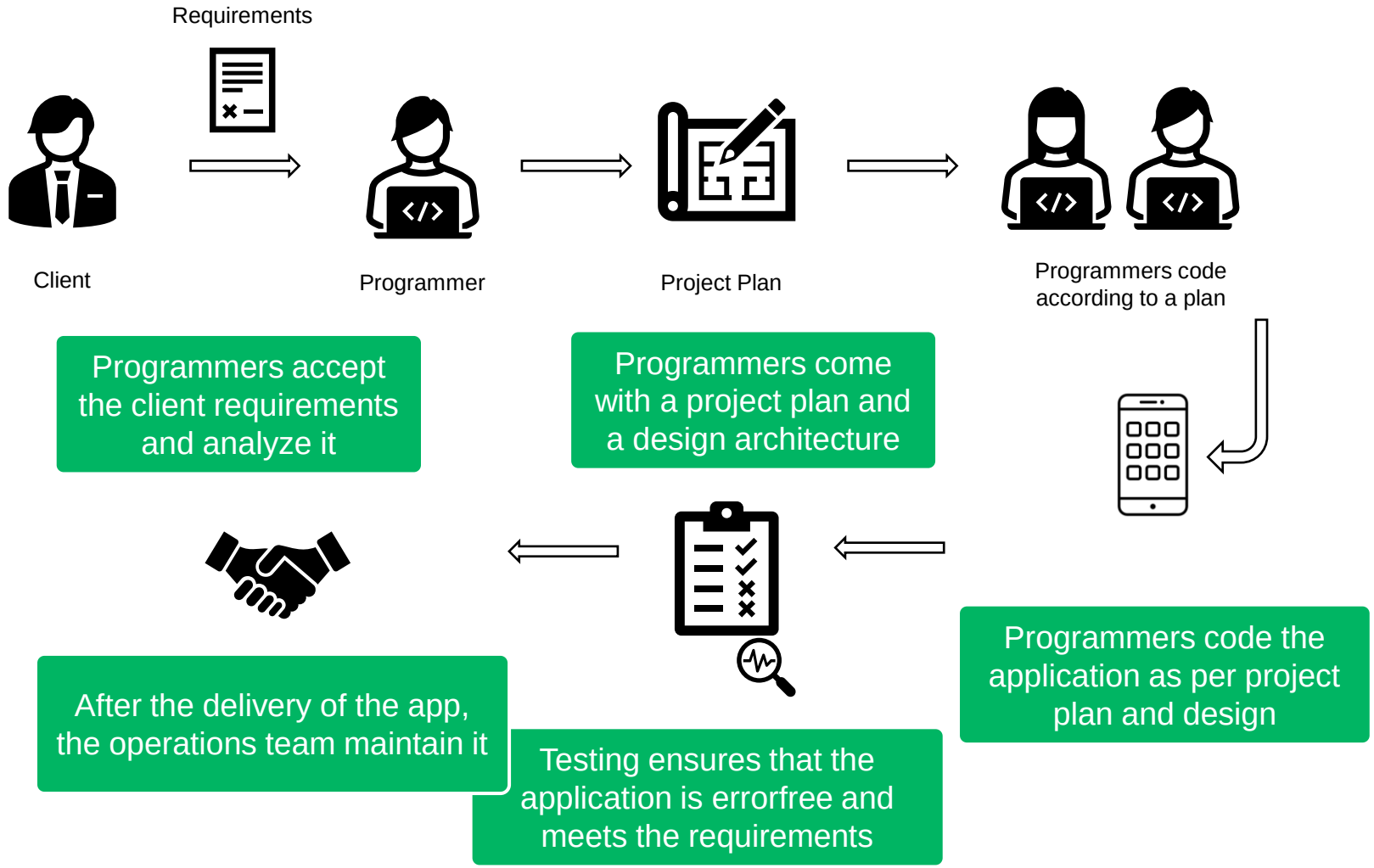
Waterfall Model

- Traditional approach of Software development
- Development happens in a step by step manner



Waterfall Model

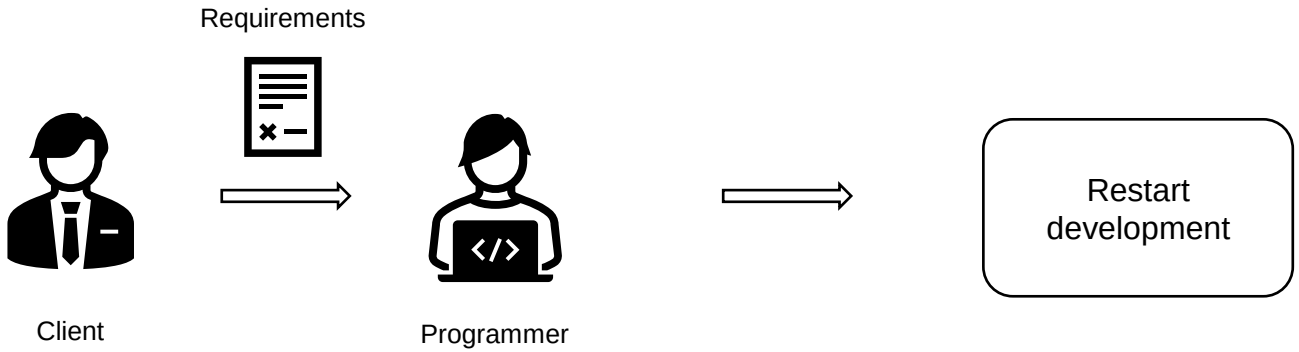




Waterfall Model

Disadvantages of the Waterfall

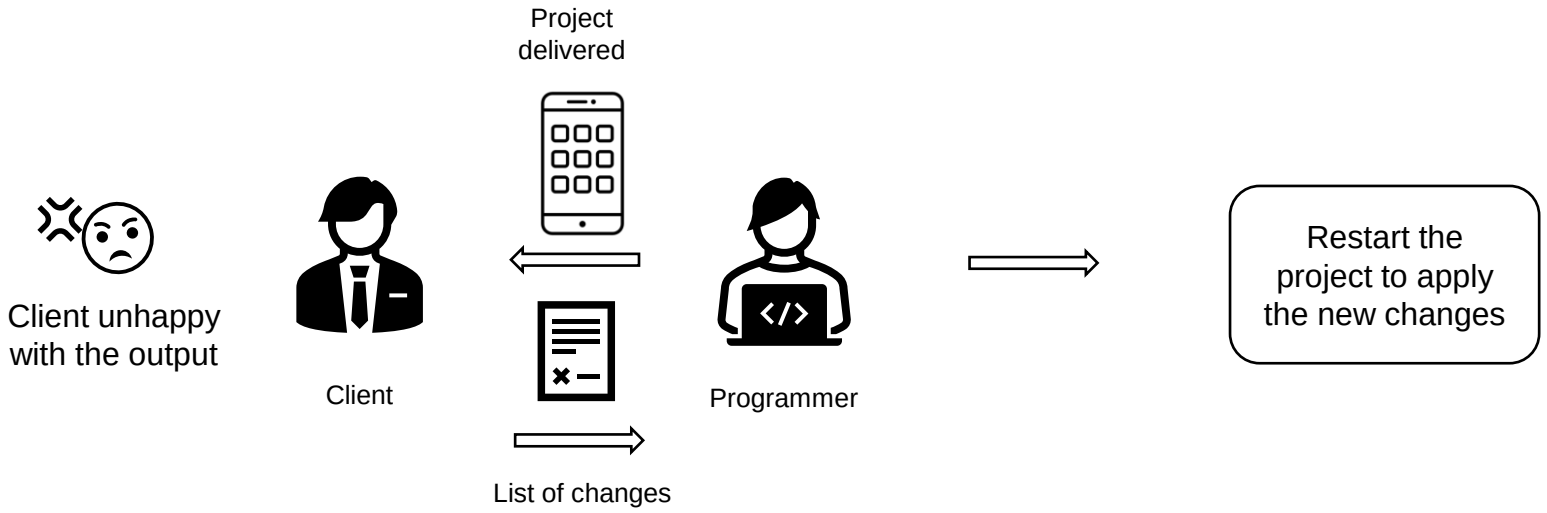
Any new customer requirement restarts the development cycle.



Waterfall Model

Disadvantages of the Waterfall

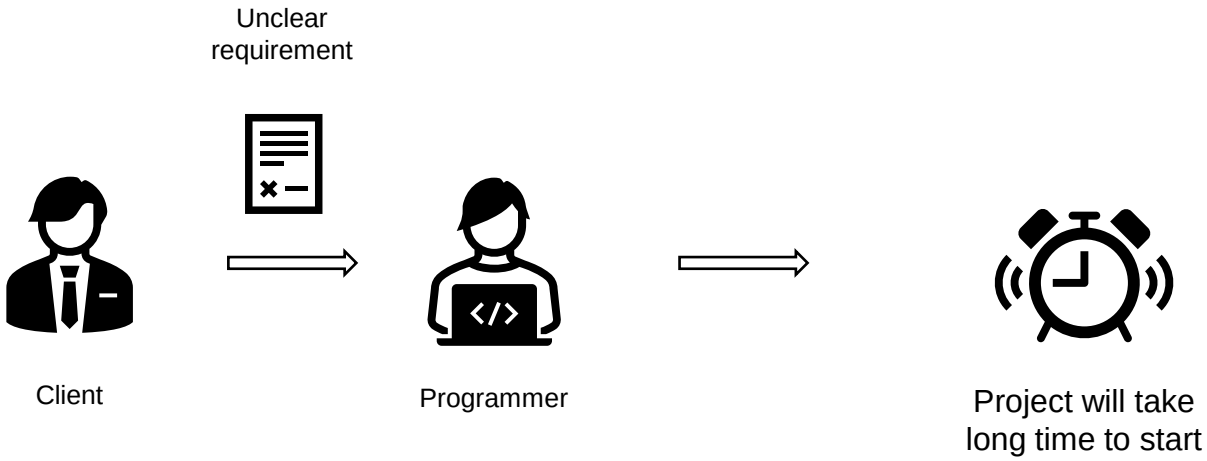
If the customer is not satisfied with the product, the entire project cycle is restarted.



Waterfall Model

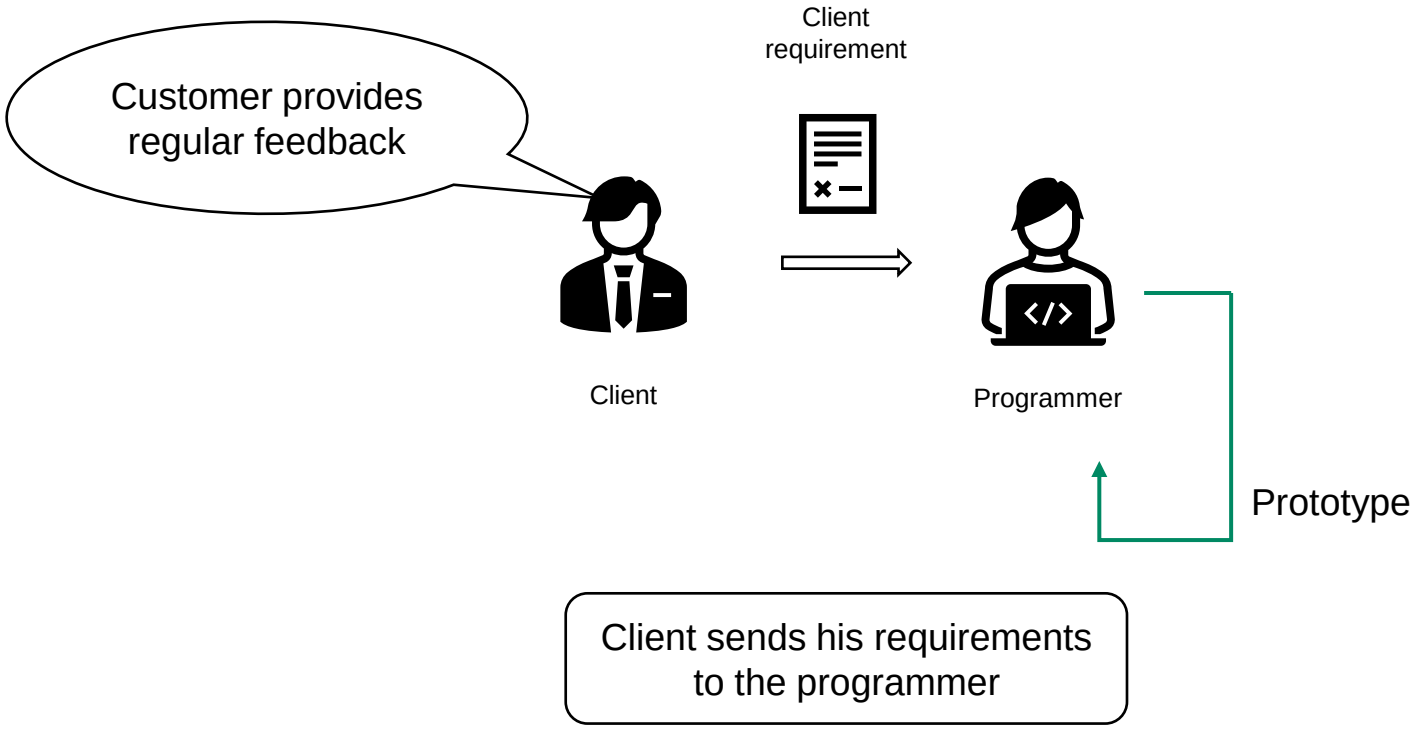
Disadvantages of the Waterfall

Until the requirements are clear, the project can't be started and ends up being delayed.

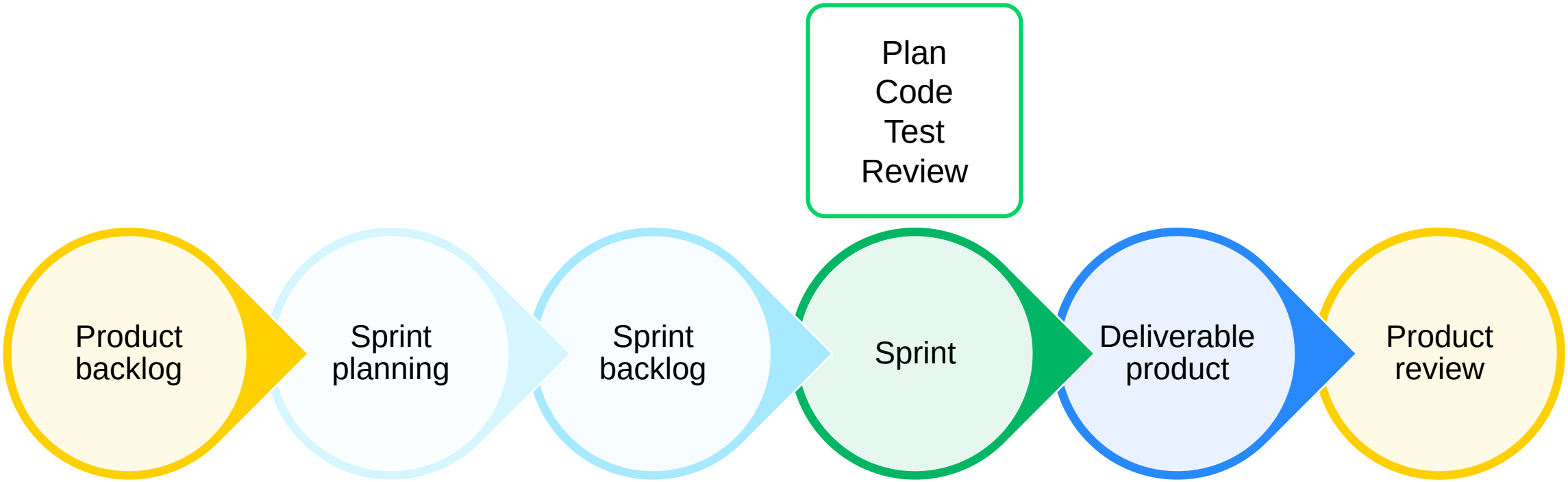


Agile Model

Following the Agile model, programmers create prototypes to understand client requirements



Agile Model



DevOps Model

DevOps is an evolution from Agile model of software development

Agile addressed the gap between clients and developers

DevOps addressed the gap between developers and operations

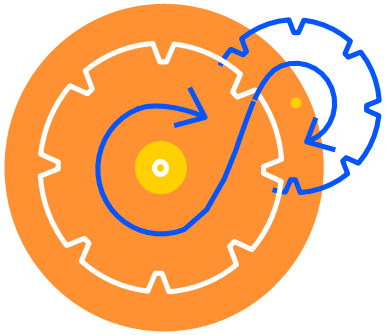
DevOps

DevOps is the combination of philosophies, practices, teams, and tools that **increase** an organization's **ability to deliver** applications and services at **high velocity and scale**

Progress®

What is CI/CD?

CI/CD



- Stands for **Continuous Integration (CI)** and **Continuous Delivery or Continuous Deployment (CD)**
- Continuous Integration (CI) uses automation tooling that empowers development teams to build, test and merge code as seamlessly as possible.
- Continuous Delivery (CD) is a means of releasing code incrementally to a platform through automation i.e., Staging, UAT, and QA. IOW it is the practice of ensuring that software is always ready to be deployed. This eliminates a risky, big-bang approach.
- Continuous Deployment (CD) provides the ability to push new software releases into production in an automated way based on a schedule or on-demand. “Push button” deployment.

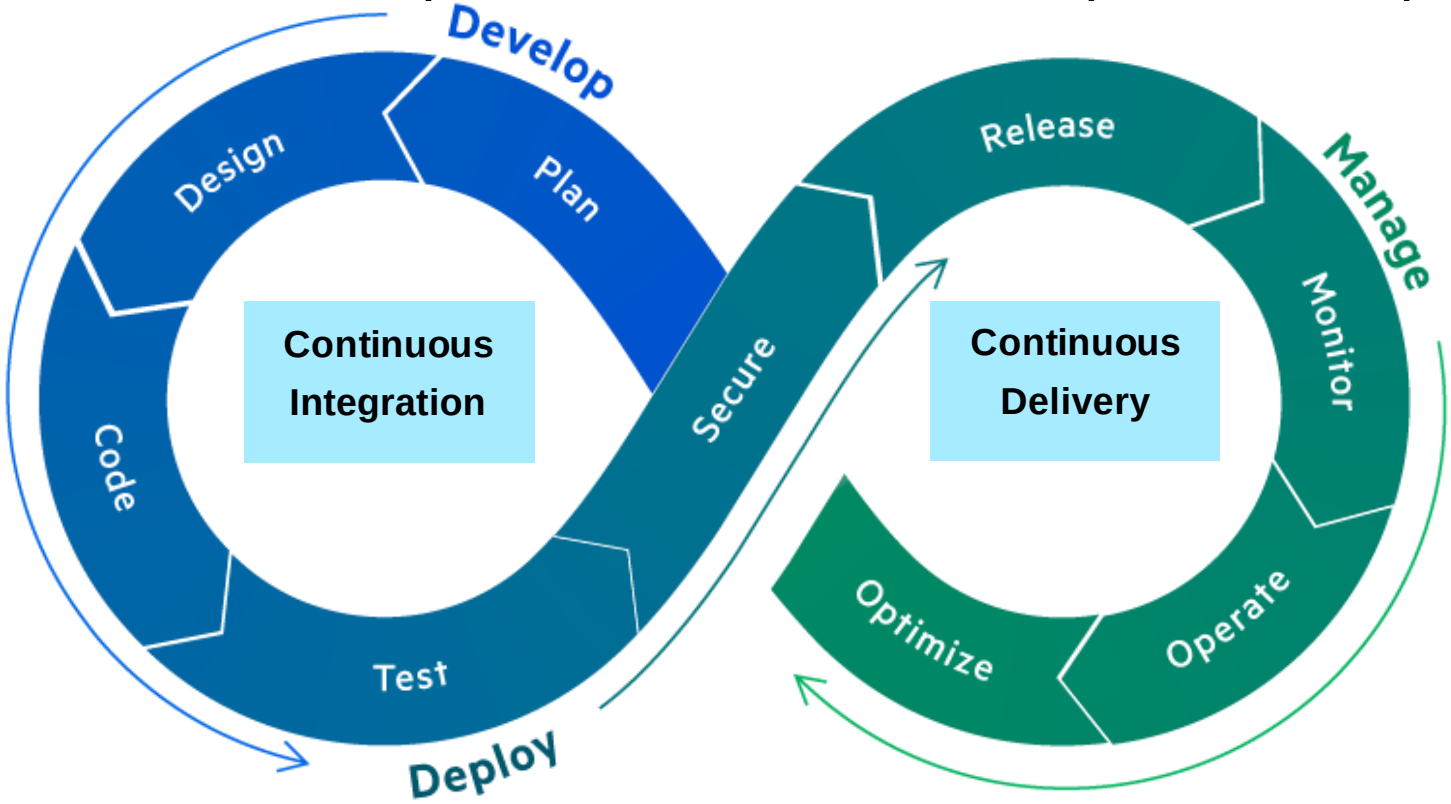
The Goal of a Mature CI/CD Strategy

To **deliver/deploy** a quality software release with **high confidence**, in a **low-risk**, **repeatable**, **non-disruptive manner** based on a schedule or on-demand.

Progress®

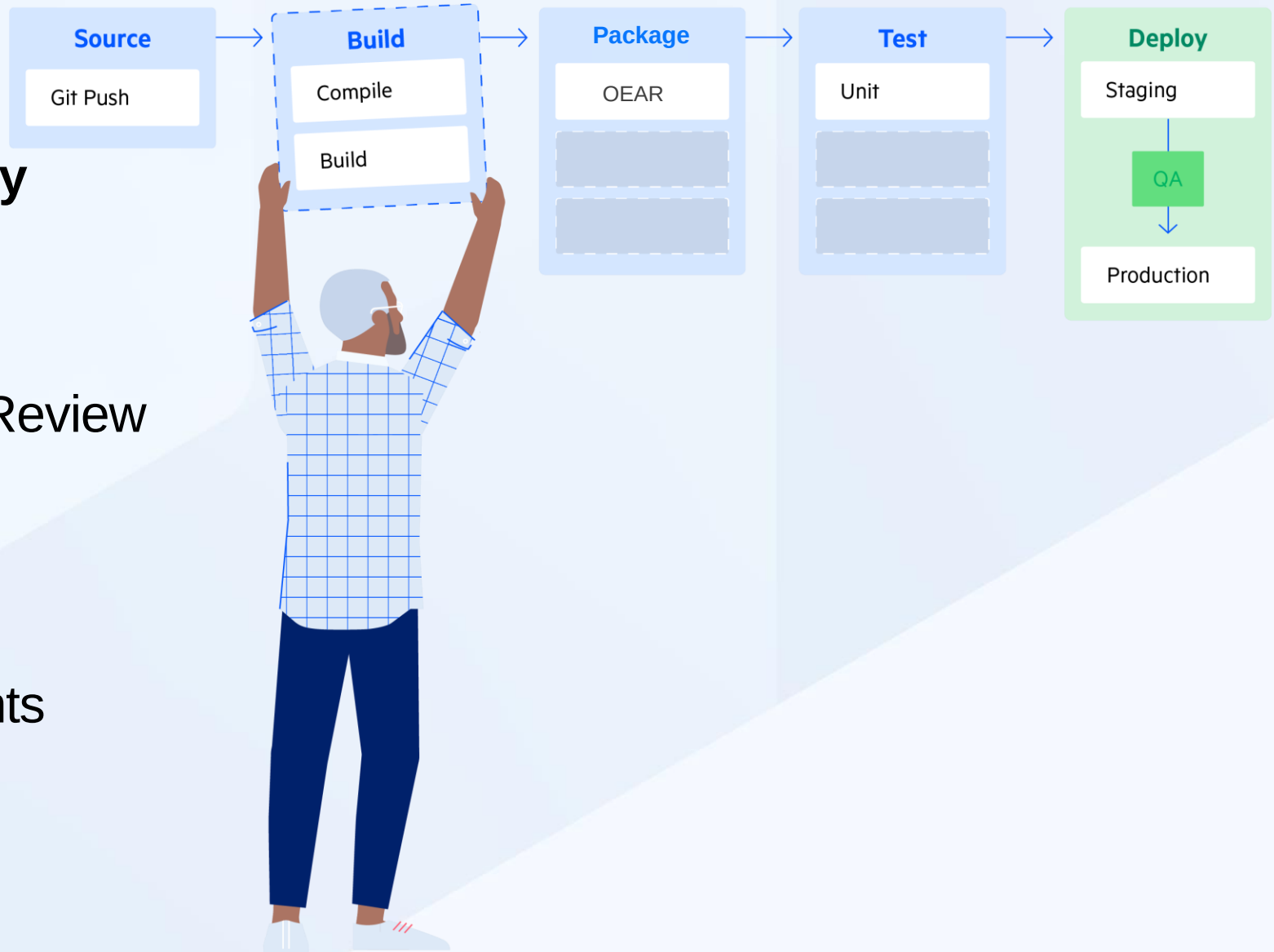
CI/CD Pipeline

- An important aspect of the pipeline is the ability to iterate, test, and validate, provide feedback at critical points, and to fail back (undo tasks) if required.



How to get started with CI/CD...

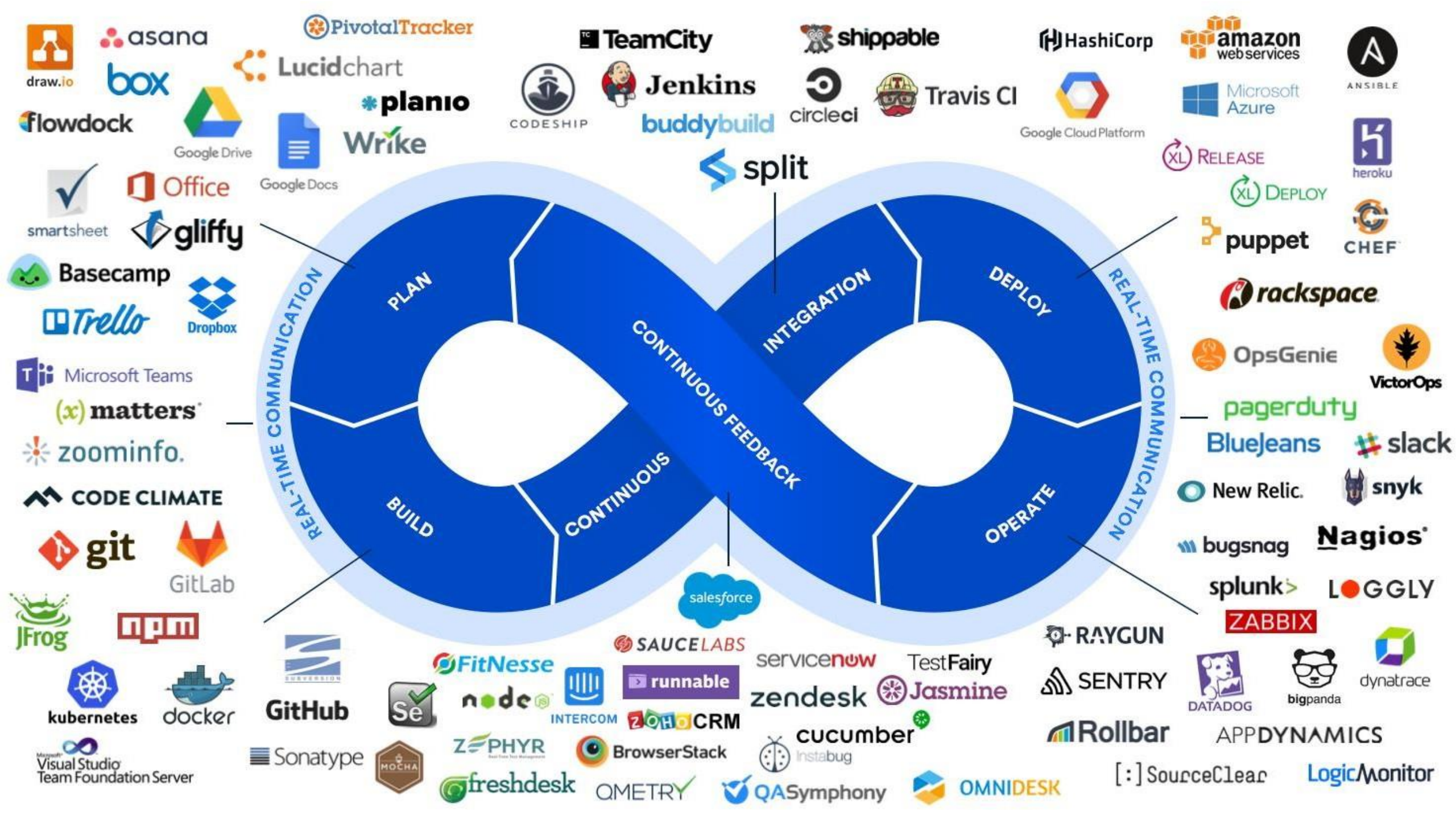
Getting Started



Or what if I had to rebuild my production platform from scratch?

- Development Process Review or Assessment
  - Source Code
  - Services
  - Application Touchpoints
  - Integrations

CI/CD Tooling


PLAN
CONTINUOUS
BUILD
CONTINUOUS
INTEGRATION
DEPLOY
OPERATE
CONTINUOUS FEEDBACK
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
TeamCity
shippable
HashiCorp
amazon web services
ANSIBLE
box
Lucidchart
Jenkins
Microsoft Azure
planio
CODESHIP
circleci
Travis CI
flowdock
Google Drive
Google Cloud Platform
Wrike
buddybuild
Google Docs
split
RELEASE
heroku
Office
DEPLOY
smartsheet
gliffy
puppet
CHEF
Basecamp
rackspace
Trello
Dropbox
OpsGenie
VictorOps
Microsoft Teams
pagerduty
(x) matters
BlueJeans
slack
zoominfo.
New Relic.
snyk
CODE CLIMATE
bugsnag
Nagios
git
GitLab
splunk>
LOGGLY
JFrog
npm
ZABBIX
salesforce
RAYGUN
SAUCELABS
servicenow
TestFairy
FitNesse
SENTRY
dynatrace
kubernetes
docker
runnable
zendesk
Jasmine
DATADOG
bigpanda
GitHub
node
INTERCOM
ZOHO CRM
Rollbar
APPDYNAMICS
ZEPHYR
cucumber
BrowserStack
Sonatype
Instabug
Visual Studio Team Foundation Server
freshdesk
QMETRY
QASymphony
OMNIDESK
[:] SourceClear
LogicMonitor

What Tools can I use with OpenEdge?

Development and Source Code Quality

SonarLint

SonarQube

PDSOE

Eclipse

What Tools can I use with OpenEdge?

Source Code and Asset Management

GitHub

RoundTable

Subversion

Mercurial

What Tools can I use with OpenEdge?

Functional and Non-functional testing

ABLUnit

Postman

jMeter

Test Studio

What Tools can I use with OpenEdge?

Build and Deploy

Jenkins

Team City

Bamboo

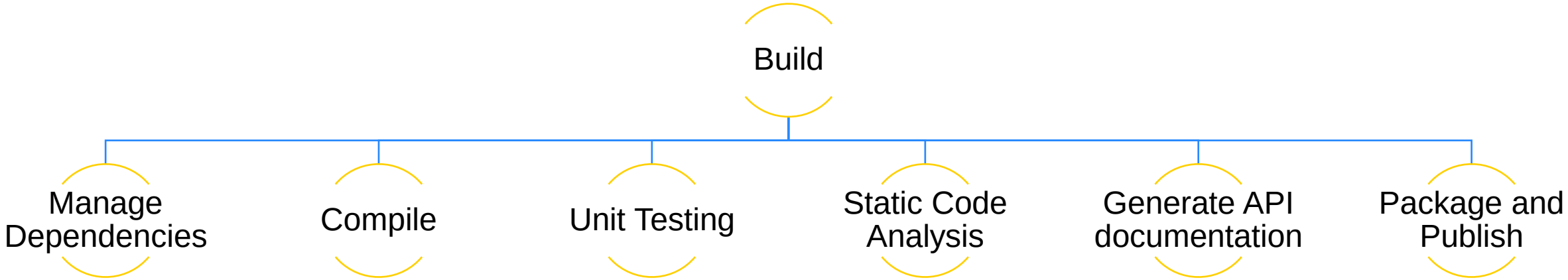
Bamboo

Ant

Gradle

CI, The Build Phase

Understand build process of an ABL project



Coding practices

- SonarLint
  - Code Analyzer for ABL (CABL)
  - Installed in Progress Developer Studio for OpenEdge
  - Checks code quality basing on predefined rules.
- SonarQube
  - Open-source code quality management platform
  - Extensible (Language support and Additional rules can be extended by plugins)
  - Over 20 languages

OpenEdge DevOps Framework Gradle plugin

- Help with implementing an efficient CI pipeline that handles:
  - Compilation
  - Repository integration
  - Testing
  - Packaging
- OpenEdge DevOps Framework includes two ABL gradle plugins available:
  - ABL base plugin (progress.openedge.abl.base)
  - ABL plugin (progress.openedge.abl)

Build with Gradle

- Open-source build automation tool
  - Takes your code and packages it into deployable unit.
  - Applies to small or large projects
- Uses domain-specific language based on Groovy for project configuration
- Determines which parts of a code base have not changed, builds and executes only the changed parts.

Key Gradle Concepts

- Plugins
  - Packages up reusable pieces of build logic
  - Can be used across many different projects and builds
  - Gradle can run custom plugins

Key Gradle Concepts

build.gradle

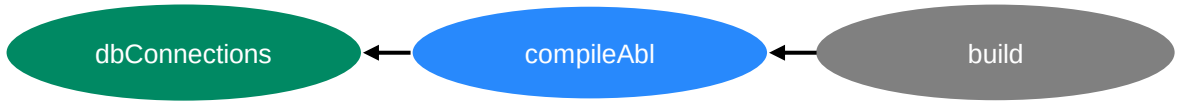
- is the Gradle build script file
  - written in Groovy DSL
  - lives at the top level of your project

*build.gradle

```
12
13 import com.progress.gradle.abl.tasks.*
14
15 plugins {
16     id 'base'                // Gives base ta
17     id 'distribution'        // for zip creat
18     id "com.dorongold.task-tree" // provide
19     id "progress.openedge.abl-base" // base
20     id "progress.openedge.abl"      // link
21 }
```

Task

- define a unit of work
- invoked from the command line ./gradelw build
- see available tasks by running ./gradelw tasks
- tasks have dependencies on other tasks



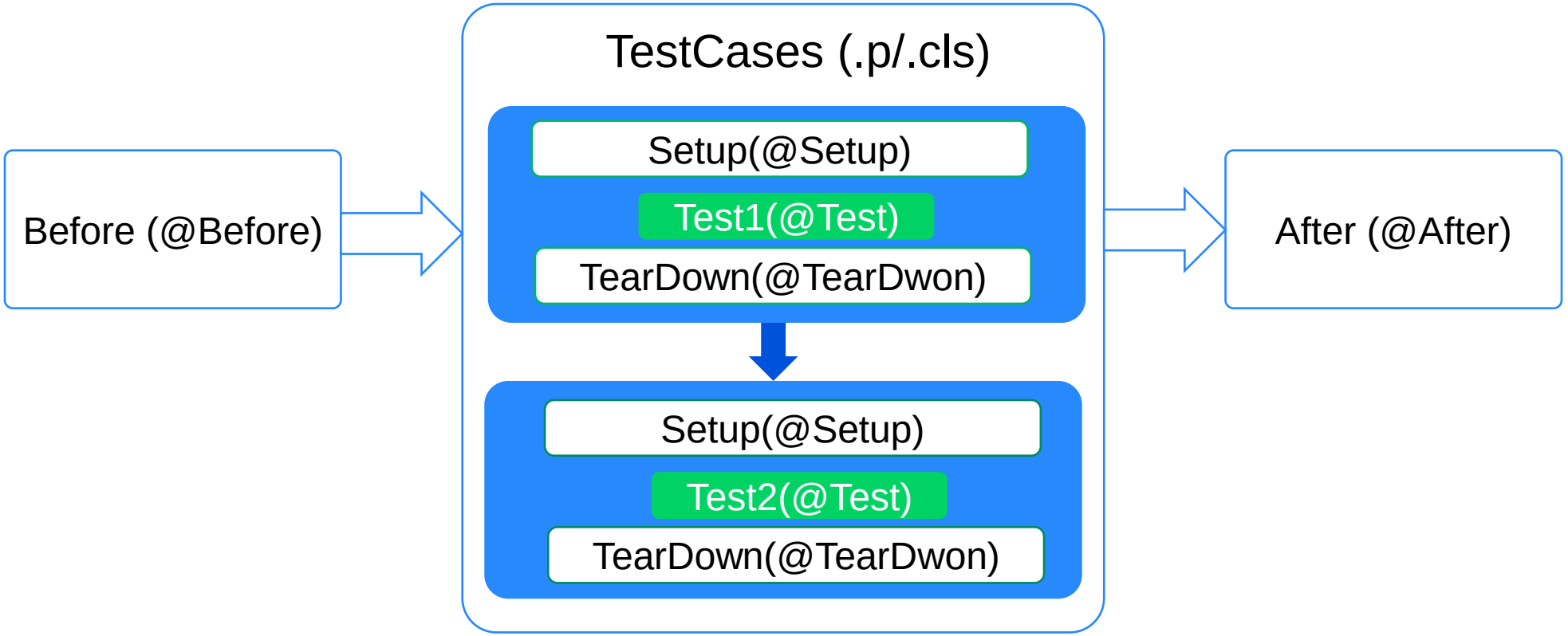
Key Gradle Concepts

- Wrapper
  - script used to invoke Gradle and run task
  - committed into version control
  - contains a specific version of Gradle for your project
- Properties
- Settings

CI, The Test Phase

ABLUnit

- Unit tests are the tests written to verify a small unit of functionality in a software.
- ABLUnit is a unit testing framework for testing in ABL.

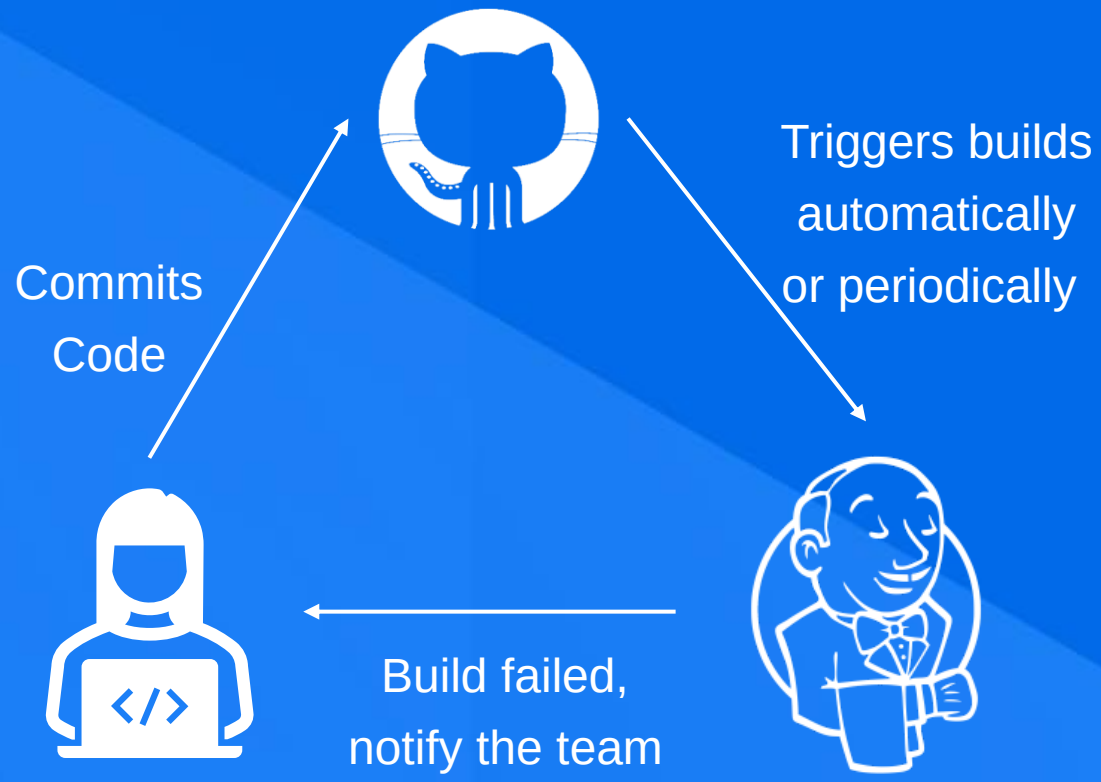


Lab1, Test & Build

Lab2, Use a CI Server

What is Jenkins?

- Jenkins is a continuous integration and build server.
- It is used to manually, periodically, or automatically build software development projects.
- It is an open-source Continuous Integration tool written in Java.
- Jenkins is used by teams of all different sizes, for projects with various languages.



CD, Deploy Docker Images

Why use Docker with OpenEdge?

- Give me the ability to “break the monolith”
- Modernize architecture
- Step towards high availability deployments
- Faster feedback in development loop

When should I use Docker?

- Docker is a basic tool, like git or java, that you should start incorporating into your daily development and ops practices.
  - Use Docker as version control system for your entire app's operating system
  - Use Docker when you want to distribute/collaborate on your app's operating system with a team
  - Use Docker to run your code on your laptop in the same environment as you have on your
  - Use Docker whenever your app needs to go through multiple phases of development (dev/test/qa/uat/prod, and Docker CI/CD)
  - Use Docker with your Chef Cookbooks (Docker doesn't do configuration management)

When should I use Docker with OpenEdge?

- At the beginning of the lifecycle … development
- During your CI/CD process
- In any high-availability high-scale load balanced deployment

The Benefits and Importance of CI/CD

Benefits of CI/CD

Automation

Build automation insures efficiency, faster build and less human intervention

Code Stability

Automation enables quicker feedback loops in cases of code defects. Identification of defects and logging results in quicker resolution

Metrics and Analytics

Quality dashboards (e.g., SonarQube) provide complete visibility and complete view into build results. Software such as SonarQube can help with validating coding standards, best-practices, security vulnerabilities and performance pitfalls, using automated static Code Analysis rules.

Benefits of CI/CD

Enables CD

Through a CI Server (e.g., Jenkins), resulting builds can be auto deployed to the target server as desired

Productivity

Developers will be able to focus on developing and spend far less energy on the CI process

Quality

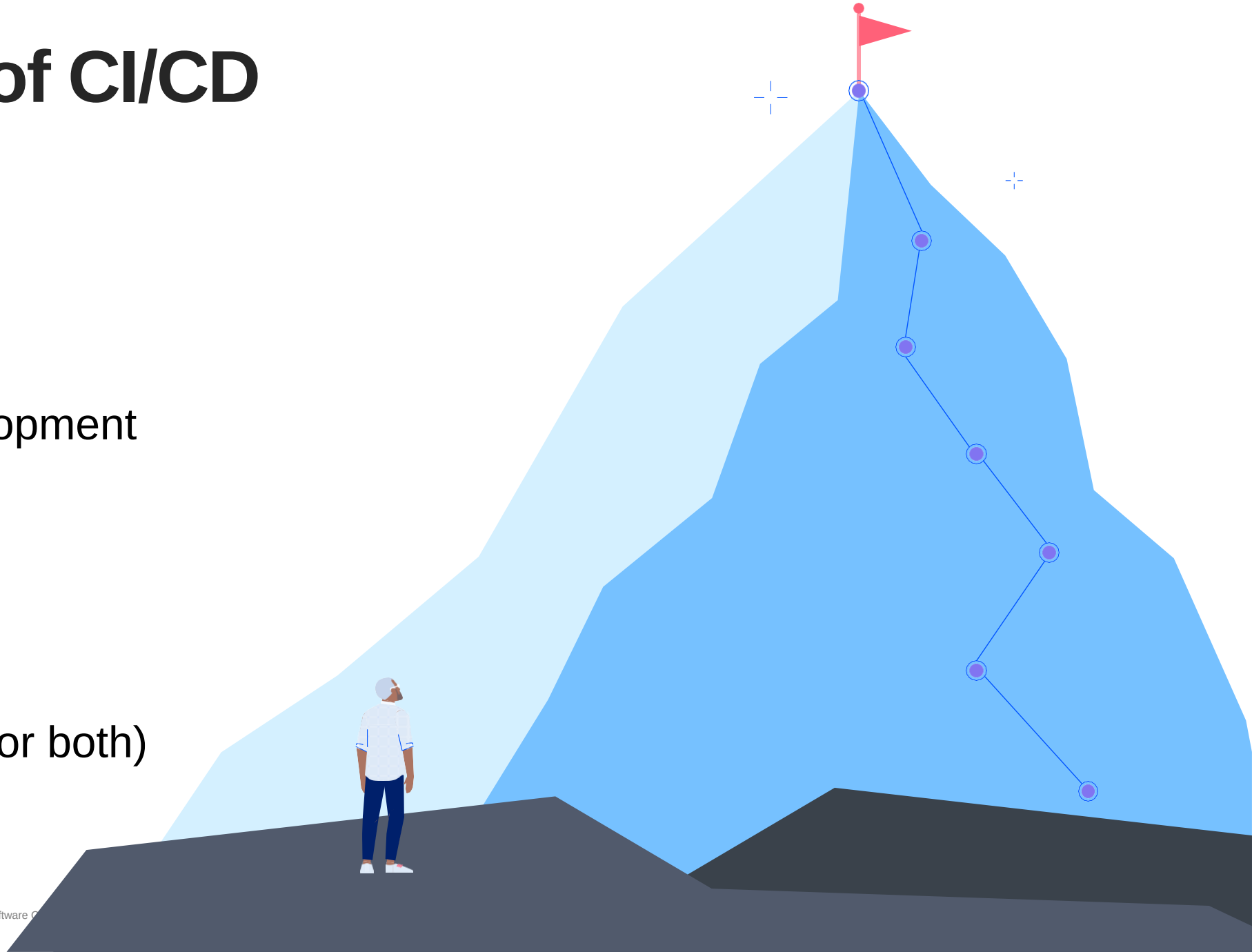
The combination of test automation and other CI benefits allows for more development focus resulting in better quality

Faster Updates/Releases

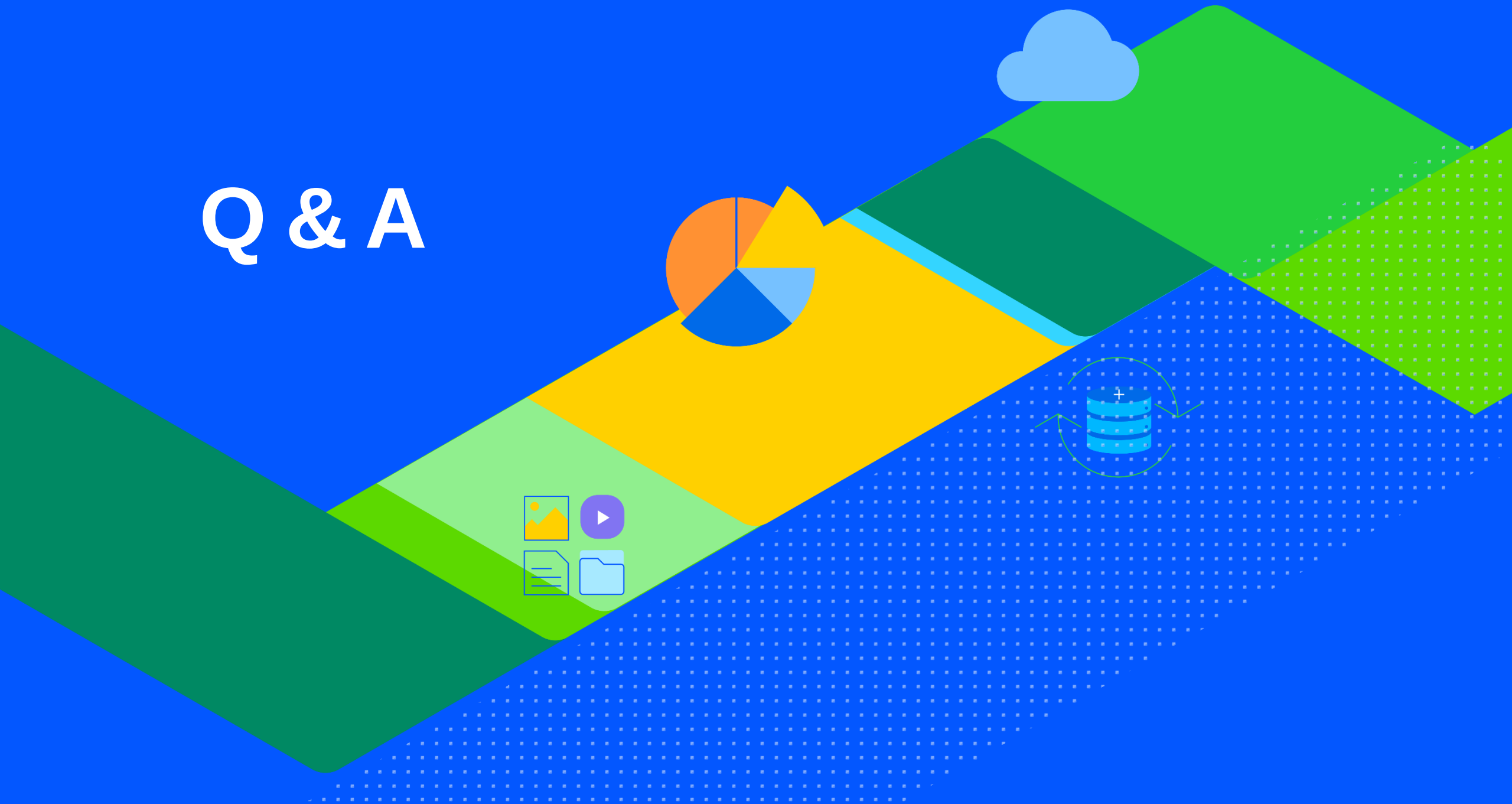
Confidence in code quality and repeatable processes enables faster release cycles

Challenges of CI/CD

- Technical debt
- Legacy Systems
- Large Team Development
- Platform
- Operating systems
- Versions
- Cloud v On-prem (or both)
- Customizations



Q & A




Progress®