Continuous Integration and Continuous Deployment tool for Control applications and frameworks

Շարունակական ինտեգրման և շարունակական տեղակայման գործիք հավելվածների և կառավարման հարթակների համար

L. Sargsyan

Continuous Integration

Definition: Continuous Integration (CI) is a software development practice where developers regularly merge their code changes into a central repository. Each integration triggers automated builds and tests, allowing early detection of integration issues.

Key Aspects:

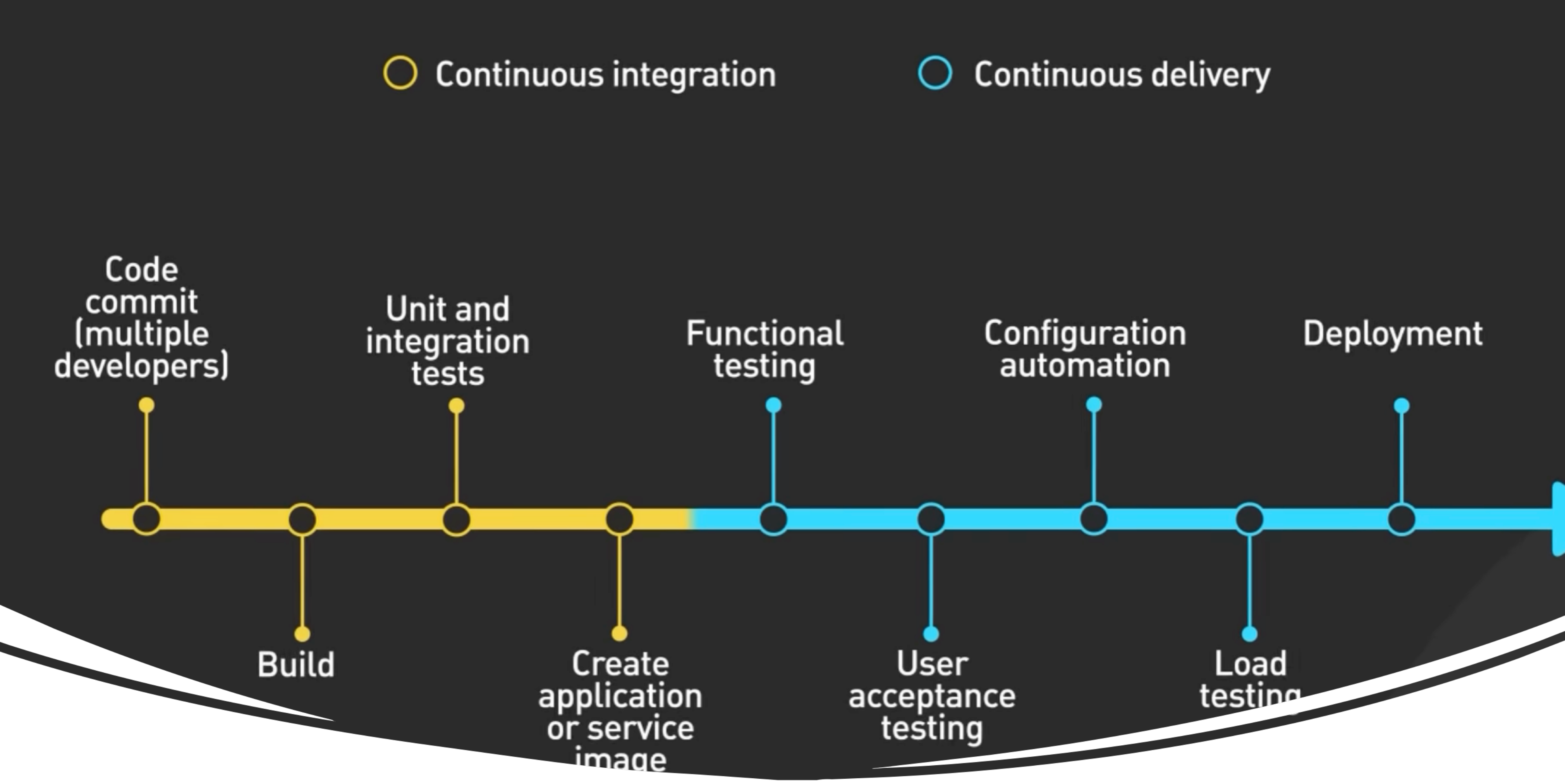
- Frequent Code Integration
- Automated Build and Test Processes
- Rapid Feedback Loop

Continuous Deployment

Definition: Continuous Deployment (CD) is an extension of CI where the code, after passing automated tests, is automatically deployed to production. It ensures a streamlined and automated release process.

Key Aspects:

Automated Deployment	Release Automation	Minimal Manual Intervention



- **CI/CD automates software development from code commit to production deployment, eliminating manual steps**

Cloud infrastructure

Interconnected hardware and software resources that provide on-demand computing services to the users.

Key components of cloud infrastructure:

Computing

- **Virtual machines (VMs):** These are isolated, virtualized computers that run on the cloud provider's physical hardware.
- **Containers:** These are lightweight, portable units of software that package code and its dependencies.

Storage

Networking

Security: Cloud providers offer a range of security features, including firewalls, intrusion detection systems, and encryption, to protect your data and applications.

Management tools: Cloud providers offer tools to manage your infrastructure resources, monitor performance, and automate tasks.

CI/CD in the cloud infrastructure

Continuous Integration: Cloud platforms offer containerization technologies like Docker, enabling you to build and test your code in isolated, ephemeral environments. This ensures consistent builds and deployments regardless of the underlying infrastructure.

↓

Continuous Delivery: Cloud providers offer various deployment tools and services that automate the deployment process, enabling you to push your code to production with minimal manual intervention.

Motivation to use CI/CD Infrastructure

- The CERN Industrial Control Systems Group provides support and software solutions for WinCC OA applications used in general infrastructure, experiments, and associated institutes. To manage the complexity of these projects, we previously relied on manual and error-prone release procedures.

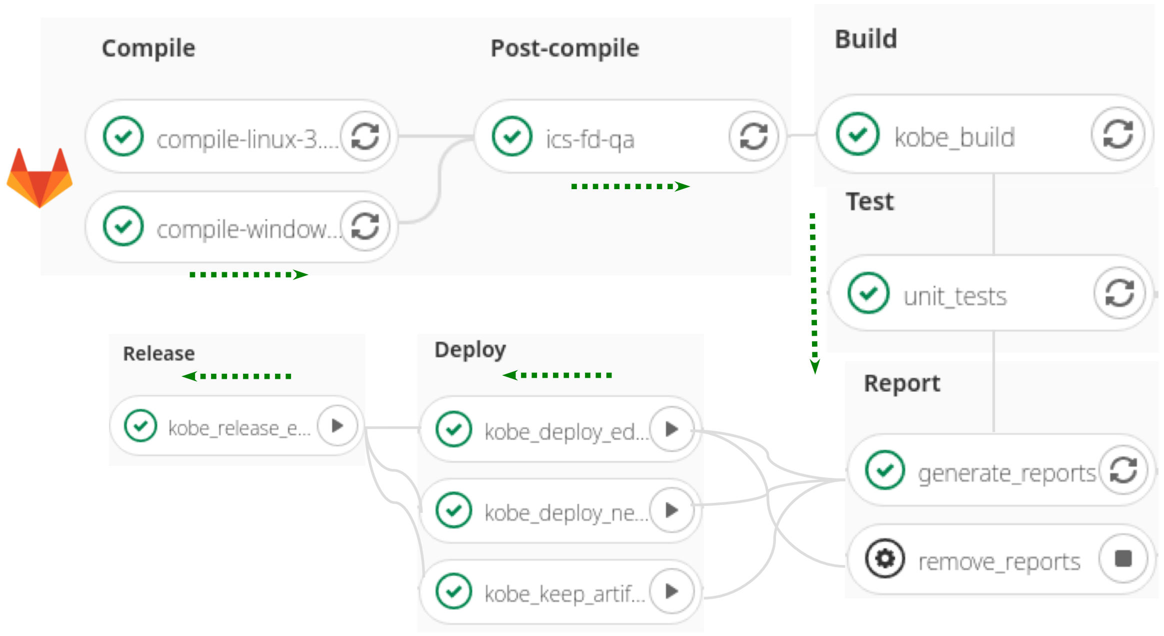
Challenges:

- Manual build and deployment processes leading to inconsistencies and errors.
- Lack of automation for common tasks, requiring expert intervention.
- Disconnected development and release processes, hindering efficiency.
- Difficulty in ensuring quality and security for software with millions of lines of code.

CI/CD Infrastructure for control applications

Example of the workflow. RELEASE GENERATION

The pipelines generate deployment-ready software releases, which pass through both static code analysis and unit tests before automatically being deployed to short and long-term repositories.

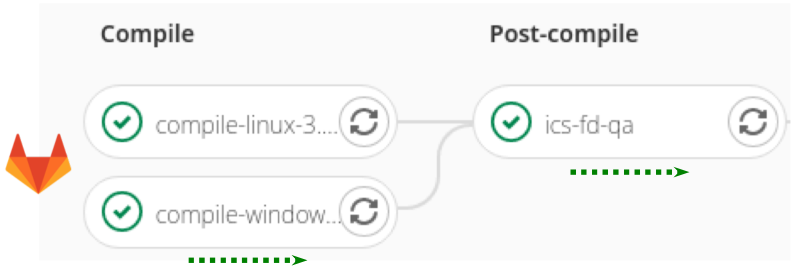


The tool chain leverages industry standard technologies, such as GitLab, Docker and Nexus.

CI/CD pipeline

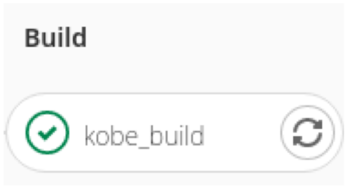
The automation of the processes is achieved by use of GitLab Pipelines, executing jobs on private hosts with Docker executors, which run custom Linux images loaded with WinCC OA application and a set of custom tooling.

Service repackages WinCC OA for Windows and Linux.

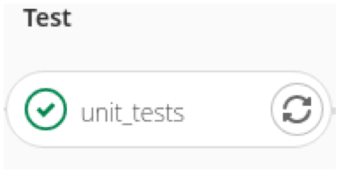


Each release is validated to meet the highest QA standards. The validation has been completely automated for Linux with the use of containers.

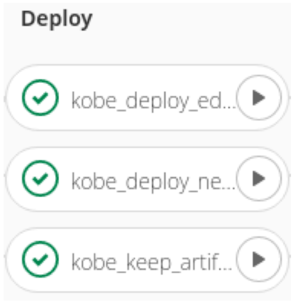
CI/CD pipeline



- After compilation, the release pipeline passes on to the build stage, where the tooling generates documentation and manuals, updates the component XML specification used for installing them, and packages it all into a zip file.



- Passing into the test stage, several QA jobs are run to find issues earlier in the development cycle, producing a stable and well tested set of components for deployment.



- Once packaged and tested, components and frameworks are deployed to repositories. Each repository serves its own purpose, and depending on the type of release, it is routed to a different one, completely automatically.

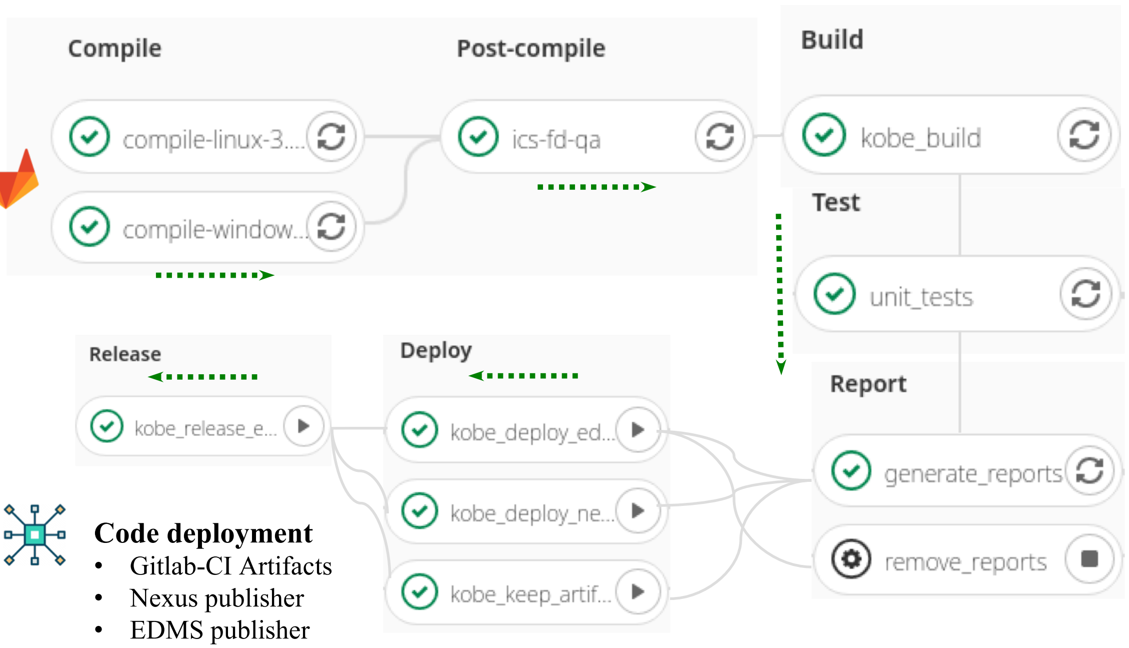
CI/CD Infrastructure for control applications

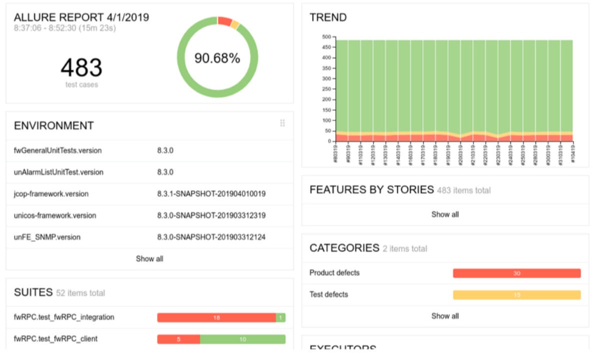
- Used by hundreds of projects with millions of lines of code
- Collaboration across the group, the department and with experiments
- Consolidation of tools and procedures (e.g. wccadm and deployments)

Kobe build

- Version automatically generated
- Up to date Qt documentation with each build, available in project help
- Subcomponents included in every snapshot

- Automated QA integration in the early development process
- Successful component build triggers build (and tests) of the framework snapshot





ALLURE REPORT 4/1/2019
8:37:06 - 8:52:30 (15m 23s)

483
test cases

90.68%

ENVIRONMENT

fwGeneralUnitTests.version	8.3.0
unAlarmListUnitTest.version	8.3.0
jcop-framework.version	8.3.1-SNAPSHOT-201904010019
unicos-framework.version	8.3.0-SNAPSHOT-201903312319
unFE_SNMP.version	8.3.0-SNAPSHOT-201903312124

Show all

SUITES 52 items total

fwRPC.test_fwRPC_integration	18 1
fwRPC.test_fwRPC_client	5 10

TREND

FEATURES BY STORIES 483 items total

Show all

CATEGORIES 2 items total

Product defects	30
Test defects	15

Show all

EXECUTORS

Code deployment

- Gitlab-CI Artifacts
- Nexus publisher
- EDMS publisher

Tests

- Daily Unit Tests with full framework
- Daily Integration tests
- Allure reports

Development and validation of the wccadm gitlab component source in collaboration with BE-ICS-CE

Conclusion

Benefits of CI/CD approach

- release frequency
- improved software quality
- reduced development effort
- early warning of issues