Building Universal CI/CD* Pipelines

A vision of a universal, semantically correct,
and platform agnostic CI/CD pipeline

Lionel LONKAP TSAMBA
Sr. DevOps & Cloud Engineer

TWITTER
@lktslionel

DATE
2022.12.01

CONF42

Agenda

01. History

Where it all started

```
pipeline {
  agent any
  stages {
    [...]
    stage(”Build") {
      steps {
        sh "rake build VERSION=${env.BRANCH_NAME}"
      }
    }
    stage("package") {
      [...]
      steps {
        sh "rake package"
      }
    }
    stage("publish") {
      [...]
      steps {
        sh "rake publish VERSION=${env.BRANCH_NAME}"
      }
    }
```

Jenkinsfile

```
name: [...]
on: [...]
jobs:
  deliver:
    [...]
    steps:
    [...]
    - name: Build
      run: |
        make build VERSION=${{ env.REF_NAME }}
    [...]
    - name: Package
      run: |
        make package
    [...]
    - name: Publish
      [...]
      run: |
        make publish VERSION=${{ env.REF_NAME }}
```

.github/workflows/main.yml

Where it all started

```
pipeline {
  agent any
  stages {
    [...]
    stage("Build") {
      steps {
        sh "rake build VERSION=${env.BRANCH_NAME}"
      }
    }
    stage("package") {
      [...]
      steps {
        sh "rake package"
      }
    }
    stage("publish") {
      [...]
      steps {
        sh "rake publish VERSION=${env.BRANCH_NAME}"
      }
    }
```

Jenkinsfile

```
[...]
stages:
  - build
  - package
  - publish
[...]
build:
  stage: build
  [...]
  script:
    - make build VERSION=$CI_COMMIT_REF_NAME
package:
  stage: package
  [...]
  script:
    - make package
publish:
  stage: publish
  [...]
  script:
    - make publish VERSION=$CI_COMMIT_REF_NAME
```

.gitlab-ci.yml

DISCLAIMER

You’re encourage to steal, use or do whatever you want with the ideas I’m going to share as long as these enable you to build better* software

02. Definitions

"

It is my belief that many of the unsolved problems in the information technology (IT) field remain unsolved simply because our technical vocabulary is impairing our ability to speak and reason about these problems. Our contemporary technical vocabulary often redefines words from everyday use and from mathematics and logic in subtly different, often confusing, and sometimes mistaken ways."

— TED HILLS, NOSQL AND SQL DATA MODELING

CI/CD*

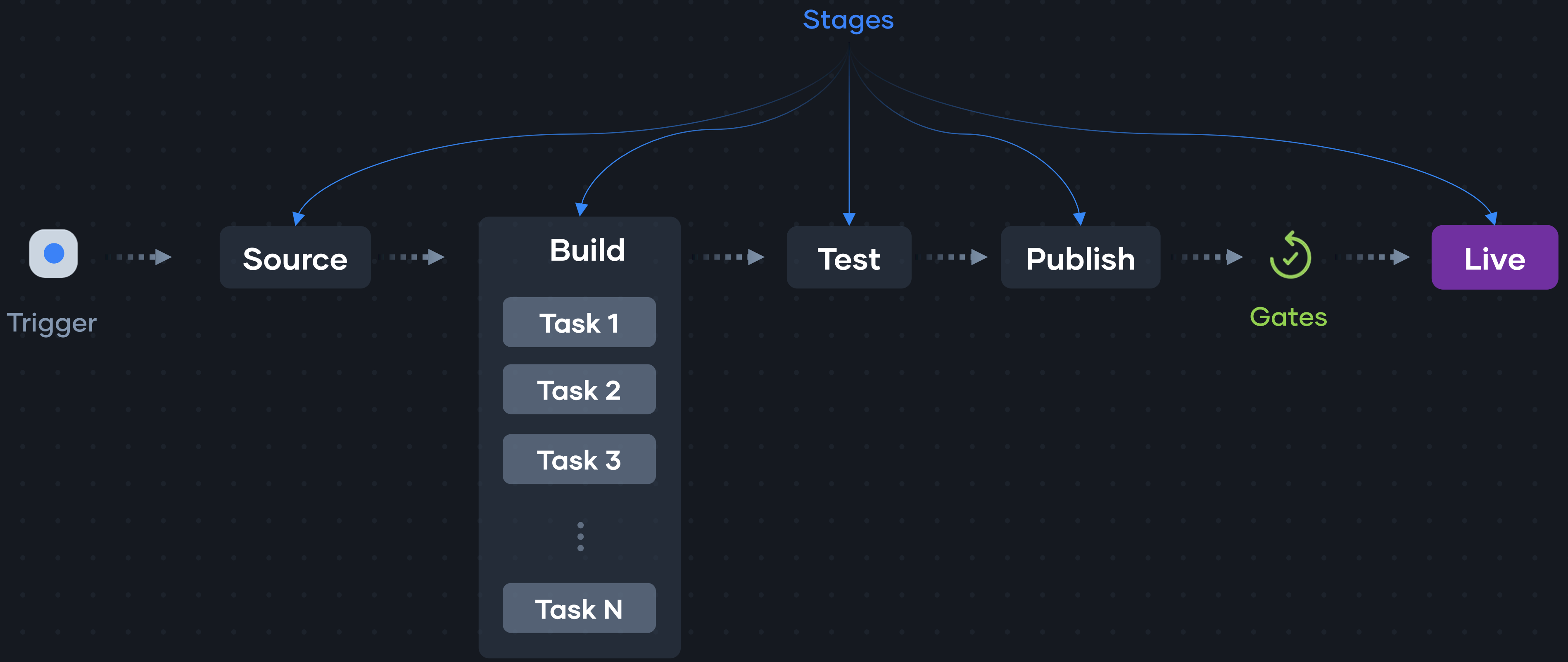
CI/CD* stands for Continuous Integration & Continuous Delivery

* Continuous Deployment (automated)

CI/CD Pipelines

CI/CD Pipeline is a workflow triggered by any mean to automate every steps or activities involved in the delivery of fit for purpose (FRs) and fit for use (NFRs) software.

CI/CD* Pipelines



Meaning of “Universal”

- Applicable everywhere or in all cases; general
- Used or understood by all

— COLLINS DICTIONARY

03. Core principles

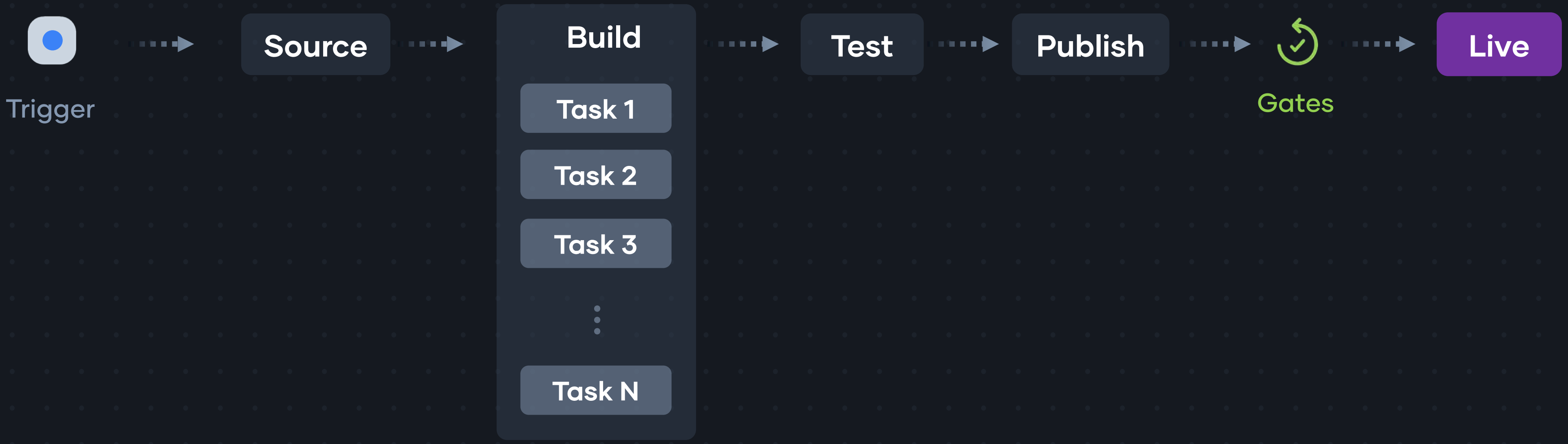
Principles of universal CI/CD pipelines

UNIVERSAL

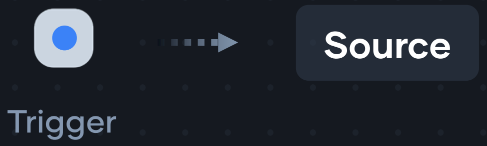
- Build a ubiquitous CI/CD pipeline *
- Used or understood by all *
- Applicable everywhere or in all cases; general
- Platform Agnostic

* Empower ownership and shared responsibility/governance

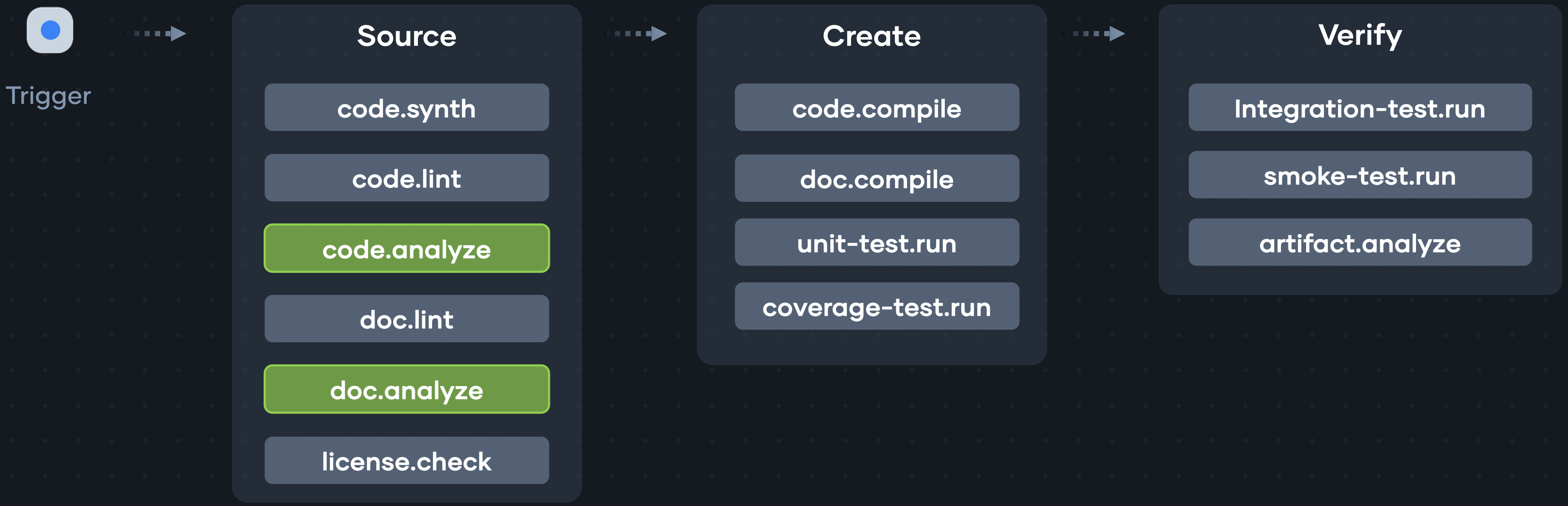
Build a ubiquitous CI/CD pipeline



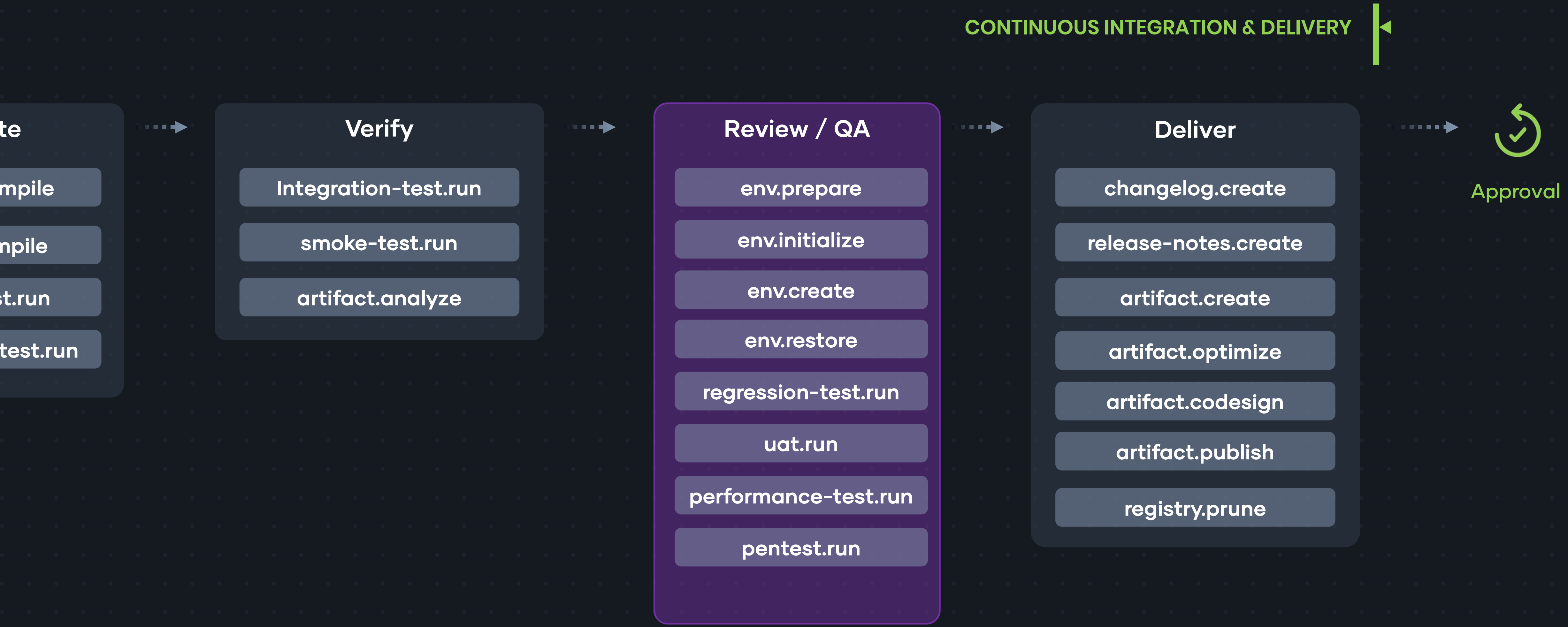
Build a ubiquitous CI/CD pipeline



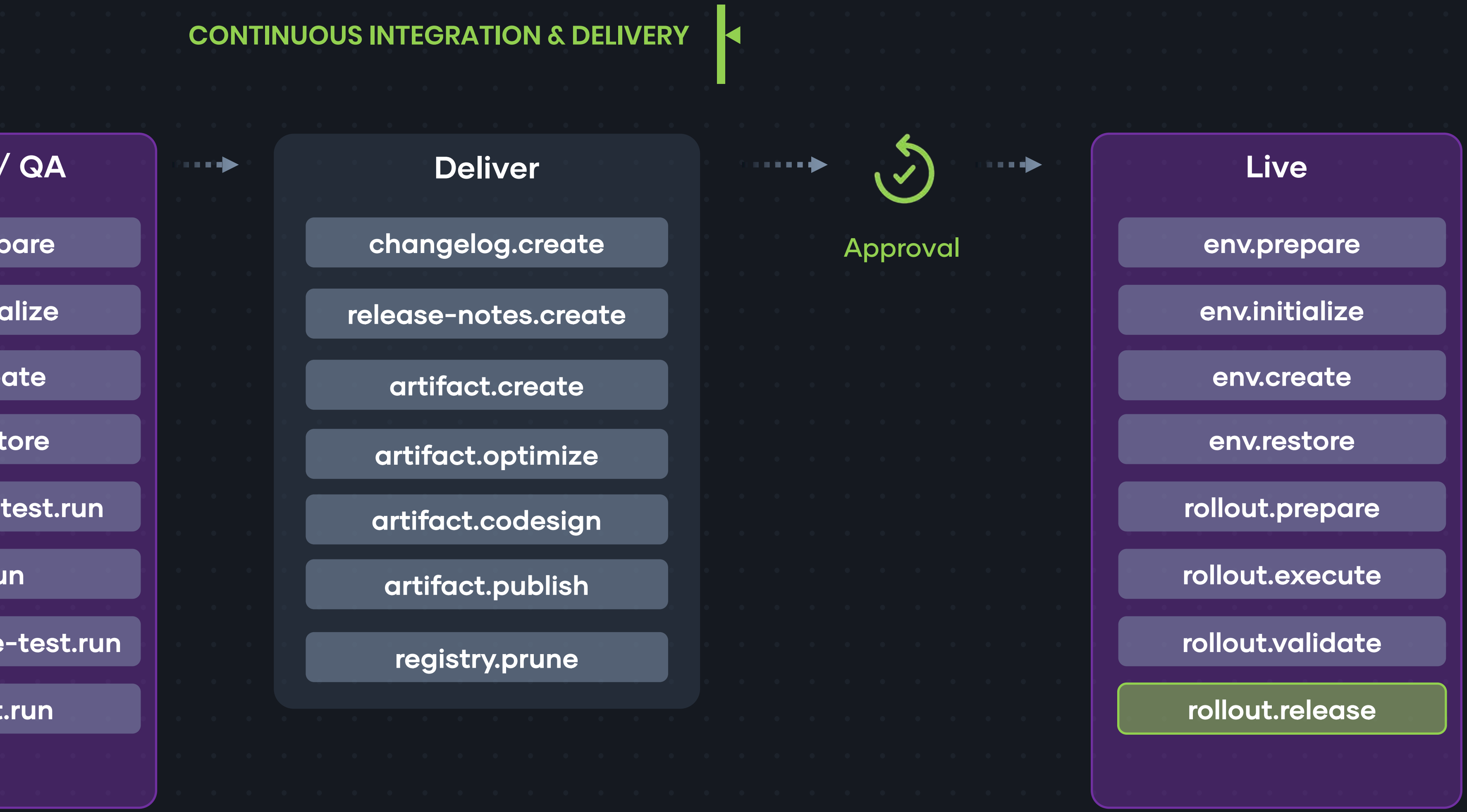
Build a ubiquitous CI/CD pipeline



Build a ubiquitous CI/CD pipeline



Build a ubiquitous CI/CD pipeline



Requirements

- Tasks managers: Make, Task, Rake, Batect
- Project generators : Projen, Hygen, Caz, Yeoman

Build a ubiquitous CI/CD pipeline ▶ Implementation

```
tasks
└── scripts
│   ├── code.synth
│   ├── code.lint
│   ├── code.analyze
│   └── license.check
├── code.synth.make
├── code.lint.make
├── code.analyze.make
└── license.check.make
Makefile
```

code.synth.make

```
#
# @task: code.synth
#

.PHONY: code.synth
code.synth:
  $(MK_ENV_VARS) tasks/scripts/$@ $(filter-out $@,$(MAKECMDGOALS))
```

Makefile

```
[...]

#
# INCLUDES
#

include $(wildcard  tasks/*.make)
```

Build a ubiquitous CI/CD pipeline ▶ Implementation

```
name: [...]
on: [...]
jobs:
  source:
    [...]
    steps:
    [...]
 - name: Checking License
      run: |
        make license.check
    - name: Synth
      run: |
        make code.synth
    [...]
    - name: Lint
      run: |
        make code.lint
    [...]
    - name: Analyze
      [...]
      run: |
        make code.analyze
```

.github/workflows/main.yml

Platform Agnostic

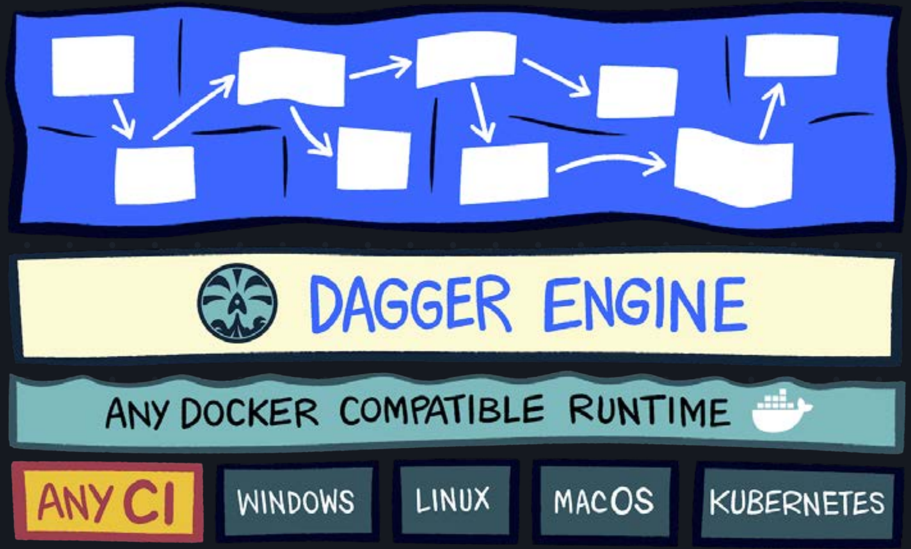
- Manifest file to define your workflows
- An Engine well integrated with CI/CD* Platforms

Dagger is a programmable CI/CD Engine

Cue	SDK
Go	SDK
Python	SDK
NodeJS	SDK

Technical Preview

+



Platform Agnostic ▶ Implementation

Demo

04. Learnings

- Focus more on semantics and less on technologies/tools
- Ease onboarding for new comers in IT
- SDKs enhance the Dev Ex and easy adoption
- Dagger makes tasks managers useless

05. What’s next

1. UNIVERSAL PROJECT SPECIFICATION — UPS

- Universal Project Folder Structure
- Tooling
- Patterns and guidelines

2. UNIVERSAL CODE DOCUMENTATION SPECIFICATION — UCDS

- Language specification
- Generators

Let’s connect

TWITTER

@lktslionel

LINKEDIN

linkedin.com/in/lktslionel

GITHUB

github.com/lktslionel