



GitLab CI/CD 101



GitLab CI/CD 入门指南

(如无说明，本 Slide 提到的 GitLab 为 13.12.15 版本)

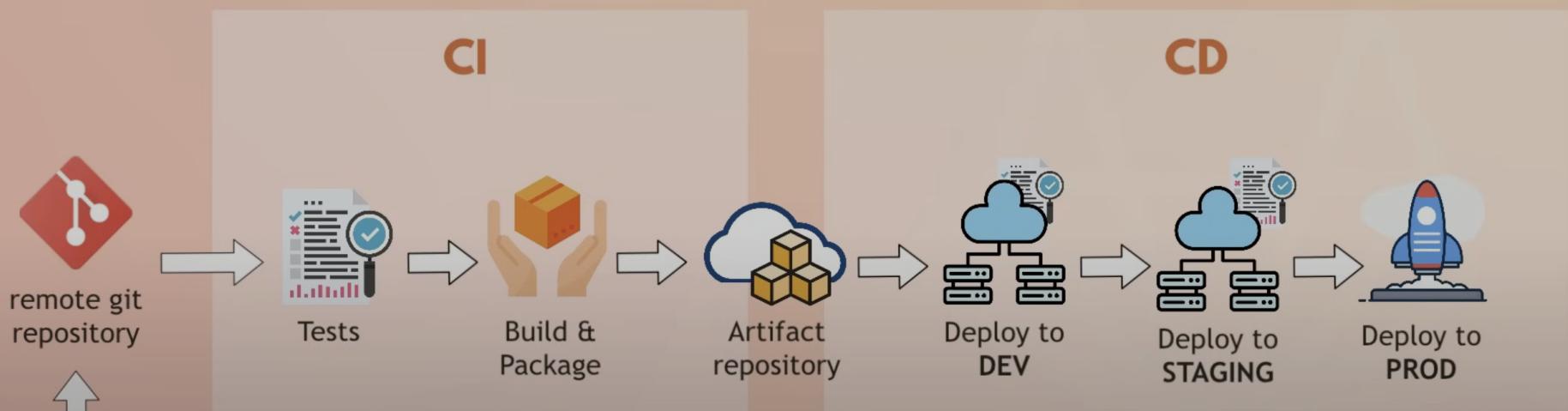
GitLab CI/CD 概述

GitLab CI/CD 是什么

- CI: Continuous Integration 持续集成
- CD: Continuous Delivery 持续交付
- CD: Continuous Deployment 持续部署



Continuously release code changes to the end environment



实现 CI/CD 的工具



Jenkins



Travis



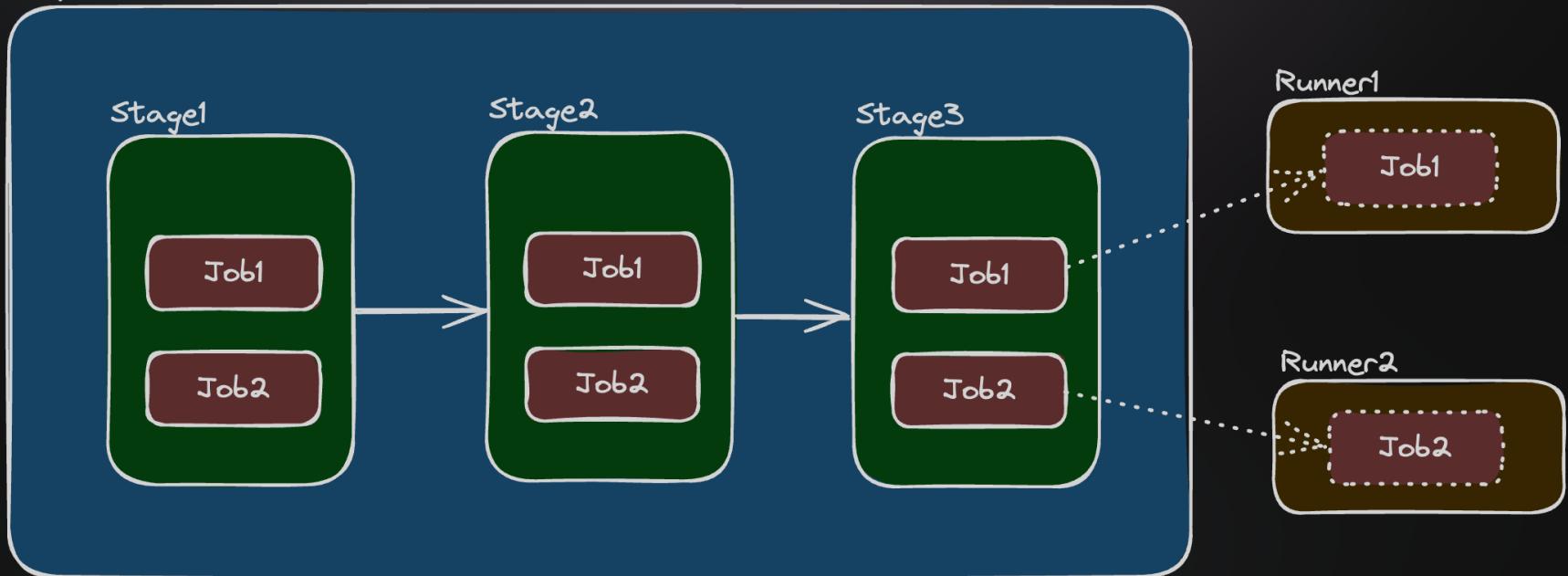
GitHub Actions



GitLab CI

GitLab CI/CD 基本概念

Pipeline



配置文件 .gitlab-ci.yml

```
stages:
  - test
  - build
  - deploy

job_name:
  stage: test # 所属 stage
  tags: # 用于选择合适的 Runner 执行 Job
    - k8s
  image: harbor-registry.inner.youdao.com/hikari/node-lite:18
  rules: # 规则, 用于控制 Job 是否应该创建或执行
    - if: '$CI_COMMIT_TAG =~ /^v\d+\.\d+\.\d+$/' # 线上环境: master 分支打 tag 触发, tag 格式为 v1.0.0
      when: manual # 手动触发
  script: # Job 执行的命令
    - npm run test
```

高级功能

变量 Variables

预定义 CI/CD 变量

- CI_COMMIT_REF_NAME
- CI_COMMIT_TAG
- CI_SERVER_HOST
- CI_PROJECT_PATH

.gitlab-ci.yml 中定义变量

```
variables: # 全局变量
  RANCHER_CLUSTER_PROD: k8s-prod-common1
  RANCHER_CLUSTER_PRE: k8s-dev-common1

job_name: # job 变量
variables:
  RANCHER_NAMESPACE: default
  RANCHER_APP_NAME: app-name
```

Web 界面中定义变量

Variables

Collapse

Variables store information, like passwords and secret keys, that you can use in job scripts. [Learn more.](#)

Variables can be:

- **Protected:** Only exposed to protected branches or tags.
- **Masked:** Hidden in job logs. Must match masking requirements. [Learn more.](#)

Type	↑ Key	Value	Protected	Masked	Environments
------	-------	-------	-----------	--------	--------------

There are no variables yet.

缓存 Cache

在 Jobs 及 Pipeline 之间共享文件，常用于缓存依赖包，提高构建速度。

```
build-app:  
  script:  
    - pnpm install  
    - pnpm build  
  cache:  
    key:  
      files:  
        - pnpm-lock.yaml  
  paths:  
    - .pnpm-store
```

工件 Artifacts

Artifacts 是 Jobs 生成的文件，可以在 Pipeline 的后续 Jobs 中使用。常用于传递构建结果、测试报告等。

```
build-app:  
  script:  
    - pnpm install  
    - pnpm build  
  artifacts:  
    name: $CI_COMMIT_REF_SLUG  
    paths:  
      - dist
```

代码复用-include

将公共的 CI 配置放在单独的文件中，然后在 .gitlab-ci.yml 文件中使用 include 指令引入这些公共配置文件。

```
include:
  - project: "hikari/f2e/common-components/common-ci-cd"
    ref: master
    file: "/templates/web-docker.yml"
```

代码复用-extends

`extends` 指令允许一个 job 继承另一个 job 的配置。可以定义一个基础 job，然后让其他 job 继承它。

```
rspec:  
  script: rake rspec  
  stage: test
```

代码复用-anchor

锚点（anchors）是用来简化和复用配置的一种 YAML 语法特性。它们允许你定义一组配置片段，并在同一个文件中多次引用这些片段。

```
test1:  
  image: ruby:2.6  
  services:  
    - postgres  
    - redis  
  script:  
    - test1 project
```

代码复用-CI/CD Components

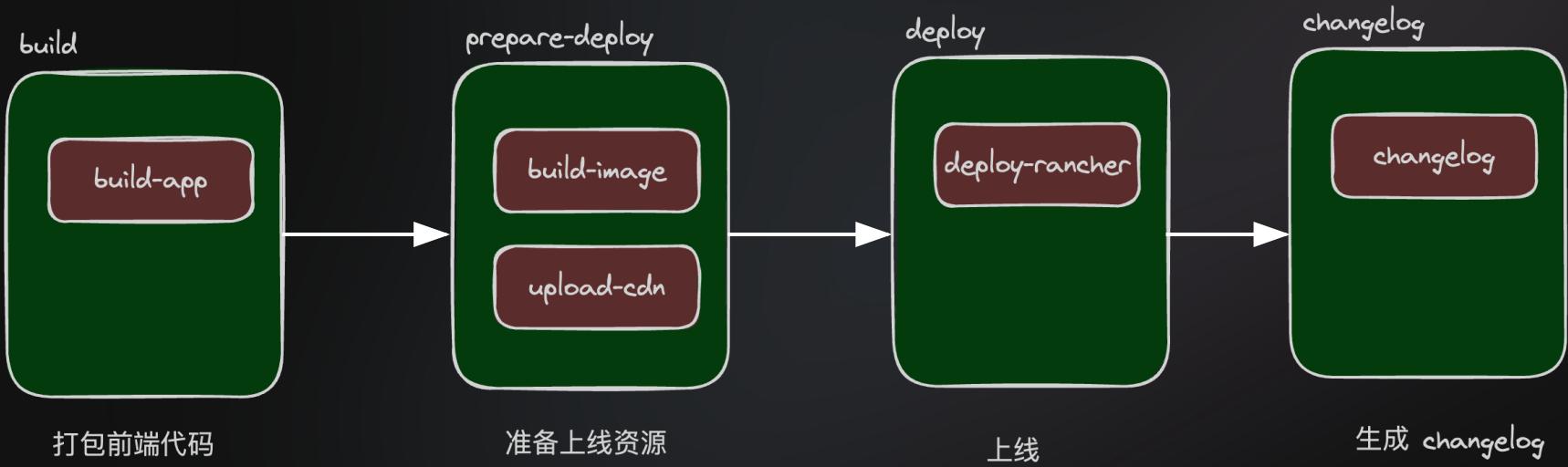
GitLab 16 引入

GitLab CI/CD Components 是可重用的 CI/CD 配置片段，可以在不同的项目中共享和重用。

```
# 组件
spec:
  inputs:
    stage:
      default: test
---
unit-test:
  stage: $[[ inputs.stage ]]
  script: echo unit tests
```

```
# 引用组件
include:
- component: $CI_SERVER_FQDN/myorg/ruby/test@1.0.0
  inputs:
    stage: verify
```

公共 CI 配置



遇到的问题

- DEBUG 变量导致 cache 失效
- cache 无法使用 job 变量
- before_script 覆盖

如何使用

<https://docs.popo.netease.com/lingxi/a89c80332d954928bfd4a891ec8e5f92>

参考资料

- [GitLab CI CD Tutorial for Beginners](#)
- [Get started with GitLab CI/CD](#)
- [使用公共 CI 部署网页](#)

谢谢！

Slides on slides.haydenhayden.com