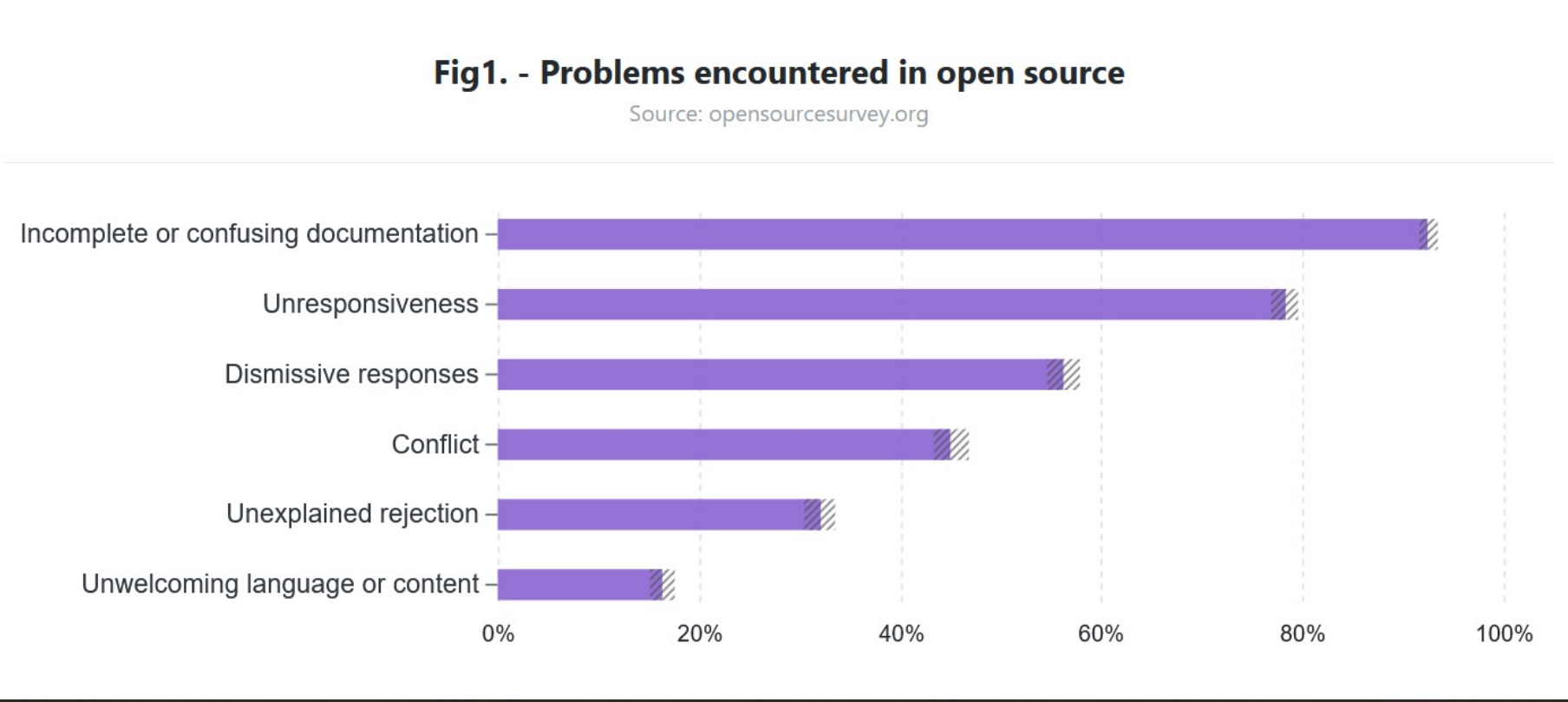
开源项目代码管理

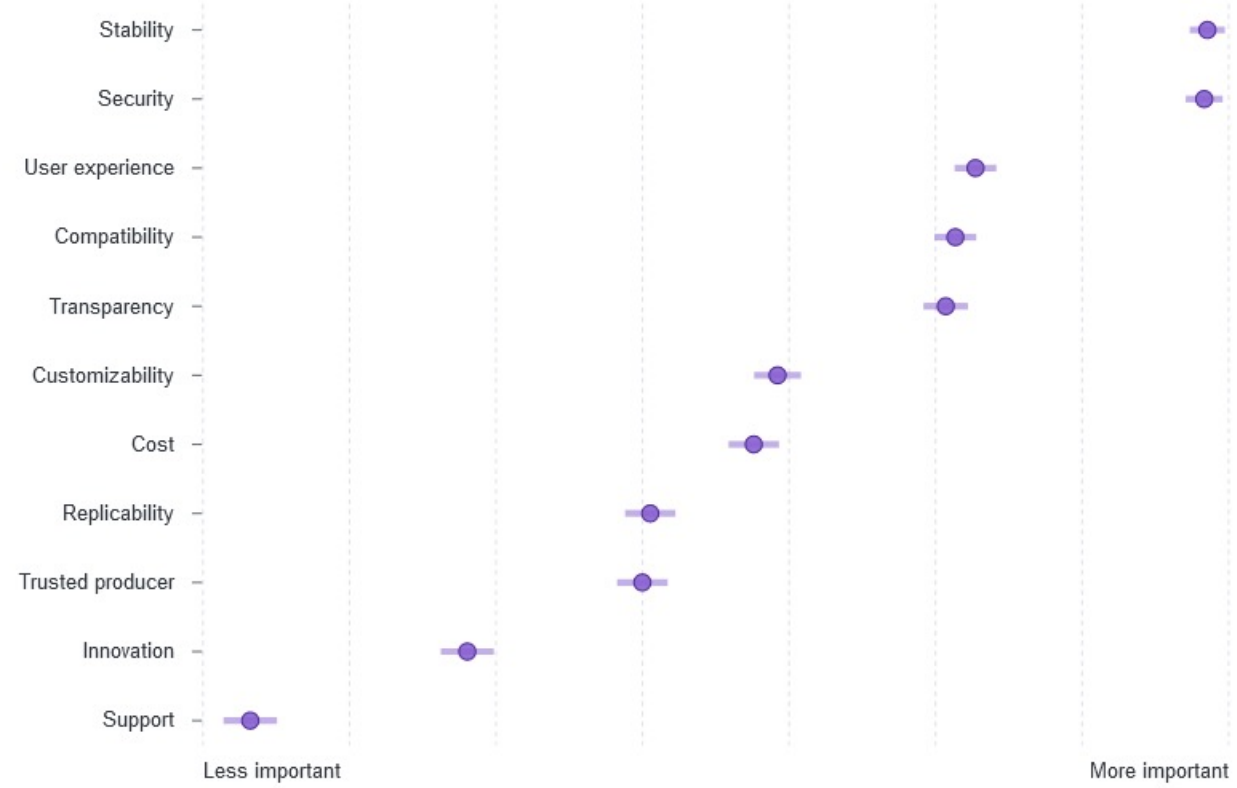
什么是一个好的开源项目



https://opensourcesurvey.org/2017/#overview

Fig5. - What open source users value in software

Source: opensourcesurvey.org



什么是一个好的开源项目

文档管理

- 清晰易懂文档，及时更新的文档

代码管理

- 高质量的代码管理
- 清晰的代码迭代管理

社区管理

- 积极的响应
- 活跃的社区建设
- 社区文化建设，License 选择

代码管理

- 高质量的代码管理
- 清晰的代码迭代管理

Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

开源项目代码平台

56+ million	3+ million	100+ million	72%
Developers	Organizations	Repositories	Fortune 50

开源项目面临的问题

Contributors 303

issues

issues

issues

Issues 130

7,260 commits

Pull requests

Pull requests

Pull requests

master 94 branches 257 tags

Used by 2k

如何解决这些问题和代码（**pull request**）的管理？

代码的持续集成测试管理体系

什么是持续集成(Continuous Integration CI)

什么是持续集成(Continuous Integration CI)

Github: Continuous integration (CI) is a software practice that requires frequently committing code to a shared repository

“a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early” (Fowler, 2006) Martin Fowler
https://martinfowler.com/

频繁的多次的将本地的代码修改合并到公共代码仓库对应的分支

主代码仓库repo

Diff → Pull Request

Forked repo

Git clone

本地代码

修改代码

Git add
Git commit

提交pull request

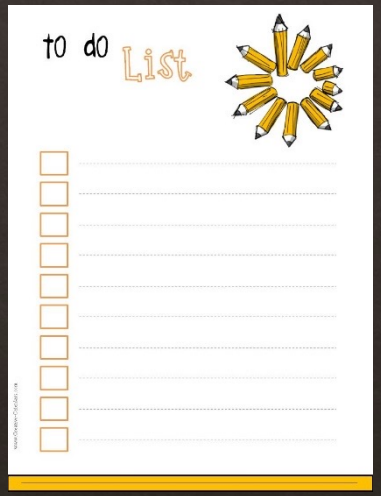
Git push

提交PR 到代码最终合并，中间发生了什么？

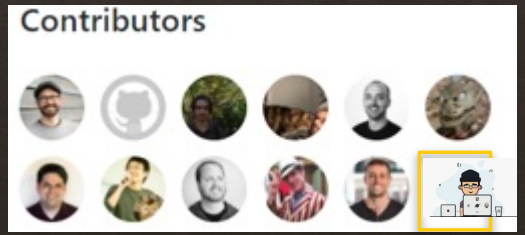
- 提交pull request

Pull requests

- 新的PR 产生



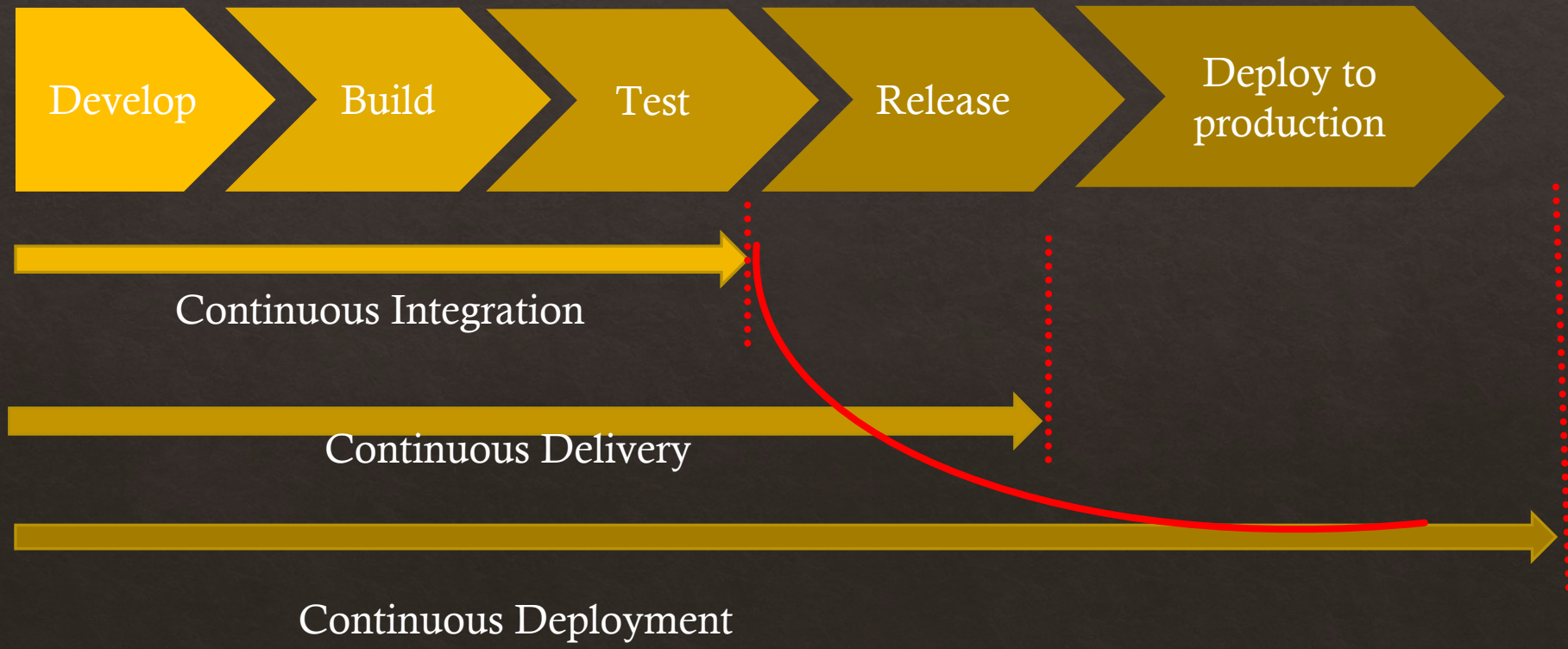
- ✓ 触发集成测试Continuous Integration Test(CI test)
- ✓ Comments
- ✓ Rebase/Update
- ✓ 再次提交
- ✓ CI test …
- ✓ Merge 合并代码

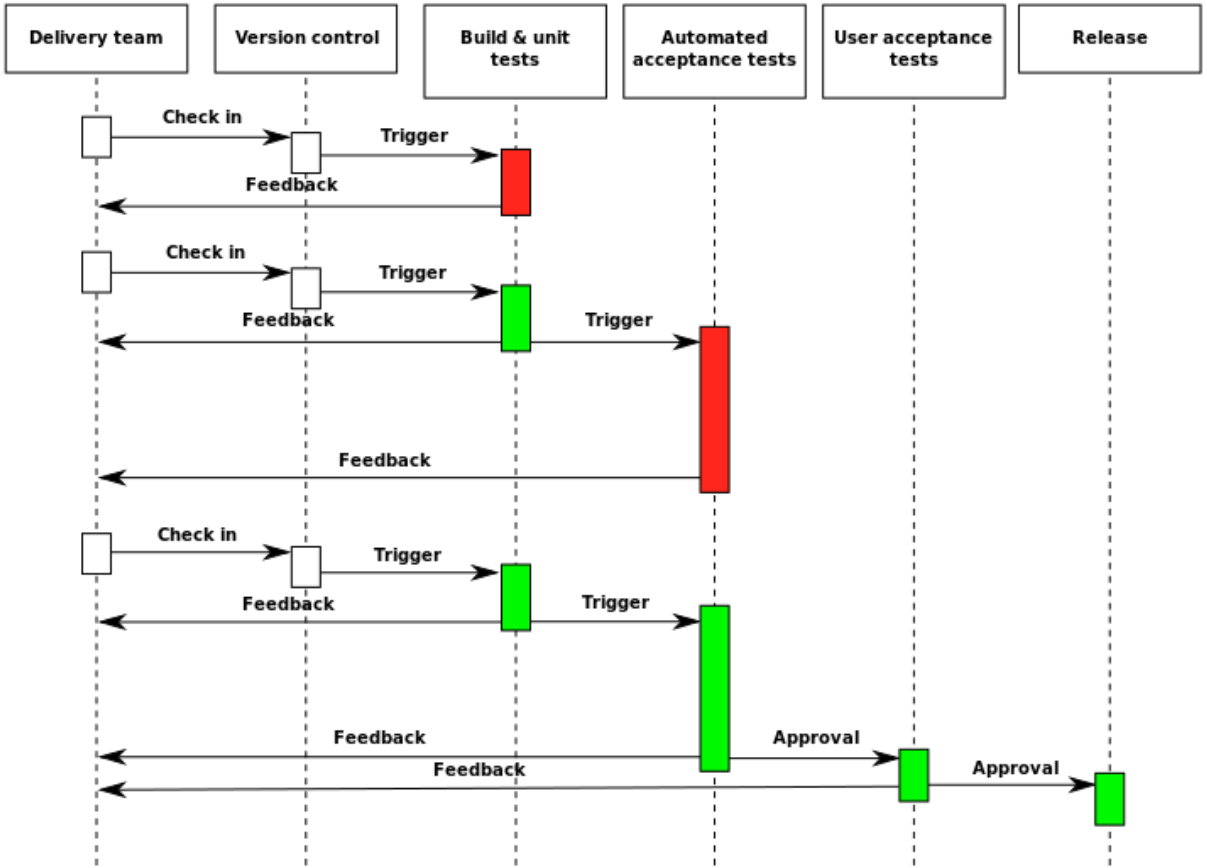


- 成为Contributors

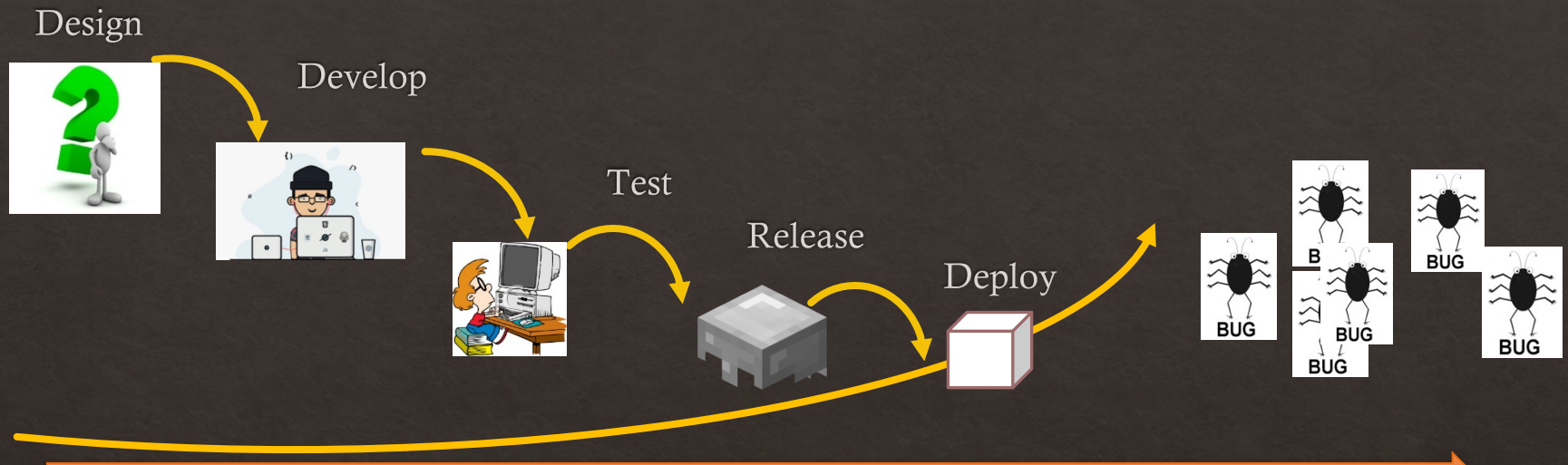
- ✓ 触发集成测试 (CI test)
- ✓ CD(持续集成部署）

CI(持续集成测试)， CD(持续交付）， CD(持续部署)




Delivery team
Version control
Build & unit tests
Automated acceptance tests
User acceptance tests
Release
Check in
Trigger
Feedback
Check in
Trigger
Feedback
Trigger
Feedback
Check in
Trigger
Feedback
Trigger
Feedback
Approval
Feedback
Approval

瀑布WaterFall模型



CD(持续部署)

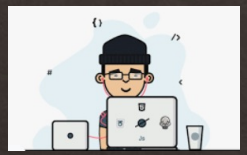
持续集成带来的优点

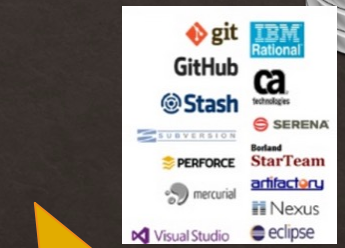
Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove.
Martin Fowler, Chief Scientist, ThoughtWorks

快速发现错误： 每完成一点更新，就集成到对应的分支，可以快速发现错误，定位错误也比较容易。

防止分支大幅偏离主干： 如果不是经常集成，分支或者主干又在不断更新，会导致以后集成的难度变大，甚至难以集成。

持续集成发布流程-CI/CD





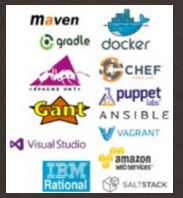
1. 检测代码变动
2. 自动构建编译
3. 自动测试
4. 自动 打包

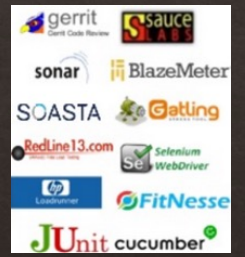
持续集成服务器

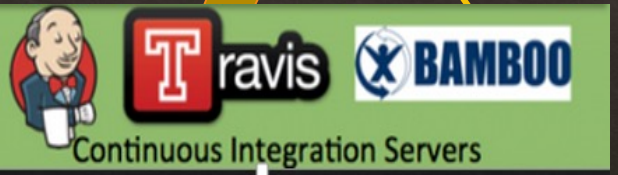
编译服务器

测试服务器

打包发布







通知

持续集成测试流程管理工具平台

Actions -- Github

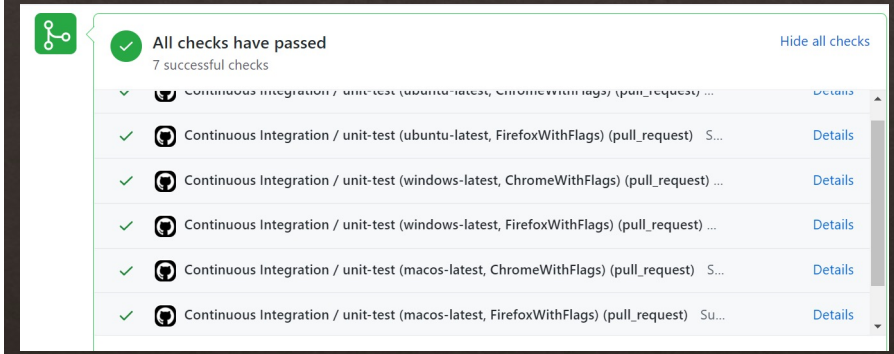
Travis CI – 3rd party public server

Jenkins -- Self host CI server

GitHub Actions

GitHub Actions

Automate, customize, and execute your software development workflows right in your repository with GitHub Actions. You can discover, create, and share actions to perform any job you'd like, including CI/CD, and combine actions in a completely customized workflow.



https://lab.github.com/githubtraining/github-actions:-hello-world?overlay=register-box-overlay

GitHub Actions

通过*.yml 文件描述整个CI/CD 的workflow, 在GitHub 提供的服务器上或者自己维护的服务器上进行CI/CD 的流程自动构建

```yaml
name: Python package

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: [2.7]

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        python -m pip install flake8  pytest
        if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
    - name: Lint with flake8
      run: |
        # stop the build if there are Python syntax errors or undefined names
        flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
```

名字

监听事件

编译环境设置

编译

运行

- name
- on
- on.<event_name>.types
- on.<push|pull_request>.<branches|tags>
- on.<push|pull_request>.paths
- on.schedule
- permissions
- env
- defaults
- defaults.run
- concurrency
- jobs
- jobs.<job_id>
- jobs.<job_id>.name
- jobs.<job_id>.needs
- jobs.<job_id>.runs-on
- jobs.<job_id>.permissions
- jobs.<job_id>.environment
- jobs.<job_id>.concurrency
- jobs.<job_id>.outputs
- jobs.<job_id>.env

Demo

Travis CI –free for open source project

Travis CI is a hosted[2] continuous integration service used to build and test software projects hosted on GitHub[3] and Bitbucket.[4]

Travis CI was the first CI service which provided services to open-source projects for free, however free open-source plans were removed at the end of 2020.[5][6]

Travis CI

通过*.travis.yml 文件描述整个CI/CD 的workflow,在Travis 提供的服务器上进行CI/CD 的流程自动构建

```yaml
language: node_js

node_js:
- node
- 'lts/*'
```

语言

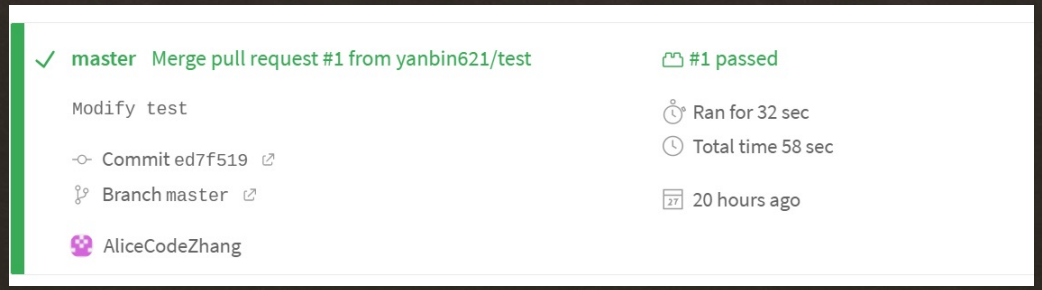
版本和环境参数

Default Build Script

The default build script for projects using nodejs is:

```
npm test
```

Bash



Demo

持续集成（CI）管理工具

Availability: Free
Platform: Cross-platform

https://jenkins.io/

Jenkins

Build great things at any scale

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

Jenkins 的优势

Source Control Management or the Version Control Management

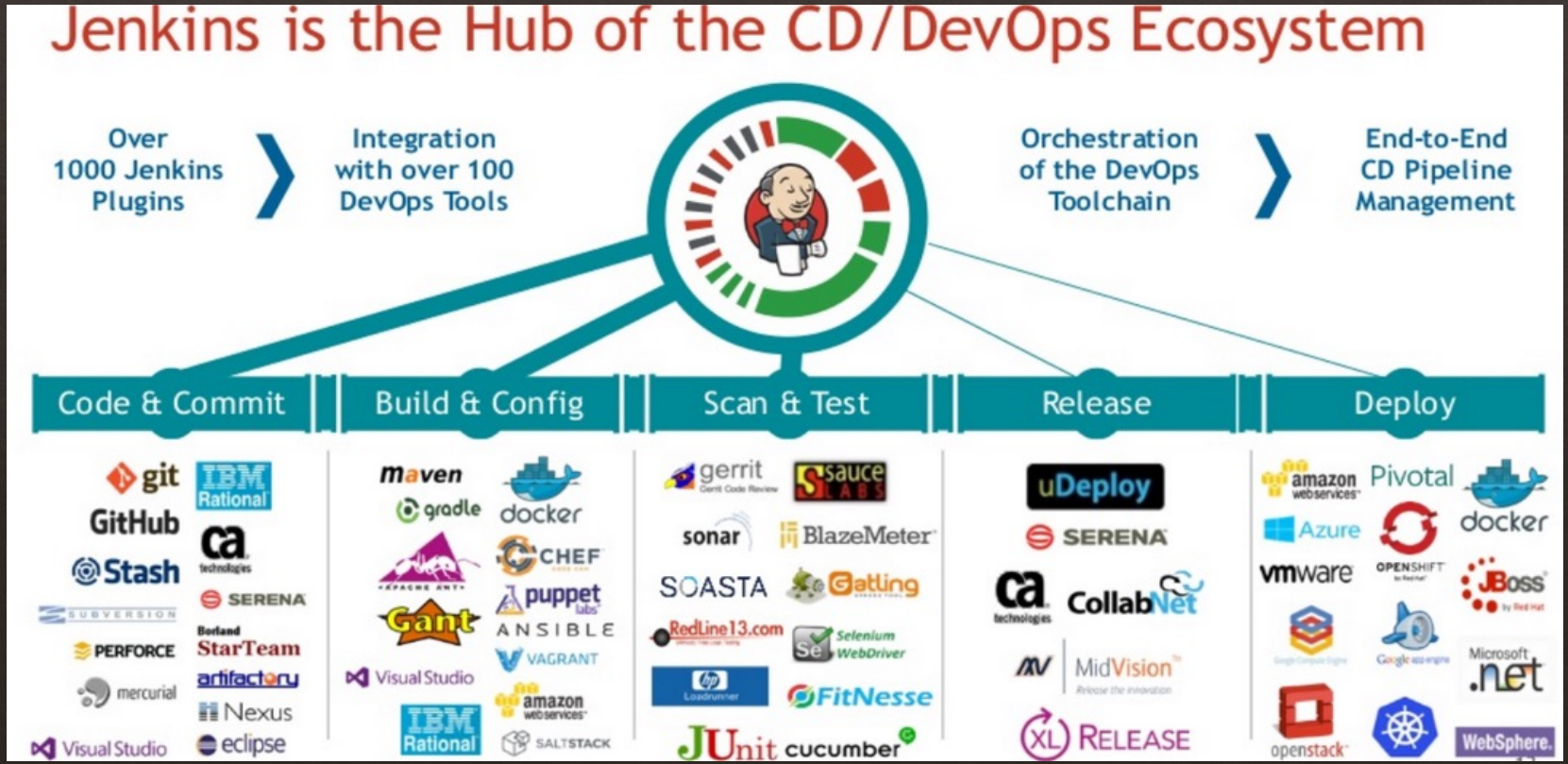
https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

Name	AccuRev	BitKeeper	CA Harvest	ClearCase	CVS	Darcs	Git	GNU Bazaar	Integrity	Mercurial	Perforce	Plastic	PVCS	StarTeam	Subversion	Surround	Synergy	Team Concert	Team Foundation Server
Vexor	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No
TeamCity	Yes	No	No	Yes	Yes	No	Yes	Yes[24]	No	Yes	Yes	No	No	Yes	Yes	No	No	No	Yes
Team Foundation Server	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No	Yes	No	No	No	Yes
OpenMake Software Meister	Yes	No	Yes	Yes	Yes	No	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Jenkins-Hudson	Yes	Yes	Yes	Yes	Yes	Yes[17]	Yes	Yes	Yes[18]	Yes	Yes	Yes[19]	Yes	Yes	Yes	Yes[20]	Yes[21]	Yes[22]	Yes
Distelli	No	No	No	No	No	No	Yes	No	No	Yes	No	No	No	No	No	No	No	No	No
CruiseControl.NET	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
CruiseControl	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No	Yes
CABIE	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	No	No	No	No
BuildMaster	Yes	No	No	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes
BuildBot	No	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	No	Yes	No	No	No	No
Buddy	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No
Bamboo	Yes[15]	No	No	Yes	Yes	No	Yes	No	No	Yes	Yes	No	No	No	Yes	No	No	No	Yes[16]
AppVeyor	No	No	No	No	No	No	Yes	No	No	Yes	No	No	No	No	Yes[14]	No	No	No	No
Apache Gump	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
Apache Continuum	No	No	No	Yes[2]	Yes[2]	No	Yes[2]	Yes[2]	No	Yes[2]	Yes[2]	No	No	Yes[2]	Yes[2]	No	Yes[2]	No	No
AnthillPro	Yes	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes

Jenkins 的优势

- Open Source 10+ years millions of users
- 超大量的安装数目（>23000) (Jul 2010)
- 超丰富的插件支持（>上千 Plugins）
- 支持几乎所有的source control management 和 Version control management tools
- 可定制化部署

Jenkins 与CD 和DevOps 关系



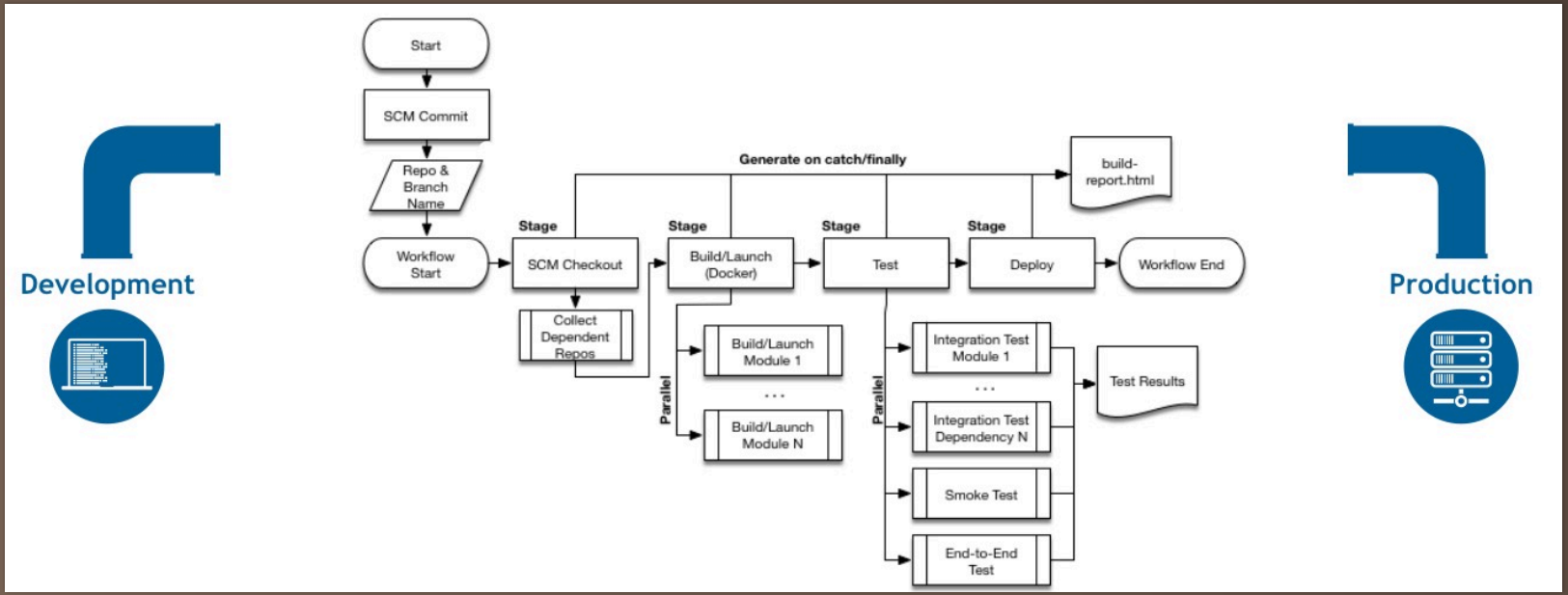
Jenkins Pipeline 机制

用解释性代码Jenkinsfile来描述，通过Jenkins服务器和一系列Jenkins 插件将整个CD(持续交付流程

Jenkins Pipeline 机制

Pipeline as code

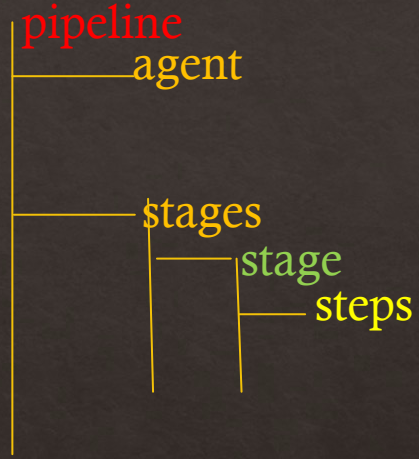
Design a whole pipeline with Jenkinsfile



http://Jenkins.io

Declarative Pipeline scripts 语法

```
pipeline {
    agent{
        label 'mac'
    }
    environment{
        mvnHome = tool 'maven'
    }
    stages{
      stage('Preparation') { // for display purposes
        // Get some code from a GitHub repository
        steps{
          git 'https://github.com/AliceCodeZhang/androidSample.git'
        }
      }
      stage('Build') {
        // Run the maven build
        steps{
          sh "pwd ; cd ./SeleniumTest; '${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore clean package"
        }
      }
      stage('Results') {
        steps{
          junit '**/target/surefire-reports/*.xml'
          archive 'target/*.jar'
        }
      }
    }
}
```



Declarative Pipeline scripts 语法

DECLARATIVE PIPELINE SYNTAX - REQUIRED

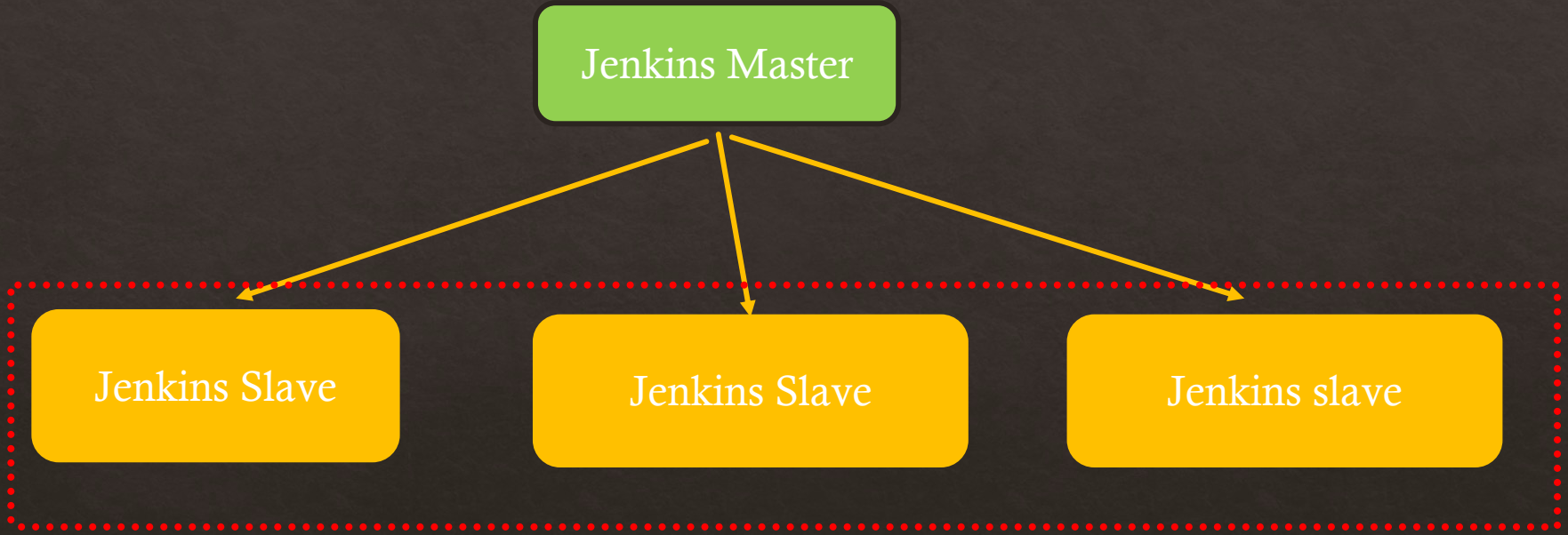
- **pipeline -** Contains the entire Jenkins Pipeline definition
- **agent -** Defines the agent used for the entire Pipeline or a stage
  - **label -** Existing Jenkins node label
  - **docker -** Requires Docker-enabled node
    - **image -** Run inside specified Docker image
    - **label -** Existing Jenkins node label
    - **args -** Arguments for Docker container
  - **dockerfile -** Use a local Dockerfile
    - **filename -** Name of local Dockerfile
    - **label -** Existing Jenkins node label
    - **args -** Arguments for Docker container
- **stages -** Contains Pipeline stages and steps
- **stage -** A specific named "stage" of the Pipeline
  - **steps -** One or more build steps that define the actions in the stage. Contains one or more of the following:
    - Any build step or build wrapper defined in Pipeline e.g. **sh, bat, timeout, echo, archive, junit,** etc.
    - **parallel (optional) -** Execute steps in parallel. May not be used with other steps
    - **script (optional) -** Execute Scripted Pipeline block
  - **when (optional) -** Runs stage conditionally
    - **branch -** Stage runs when branch name matches
    - **expression -** Boolean expression
  - **agent, environment, tools** and **post** may also optionally be defined in **stage**

+

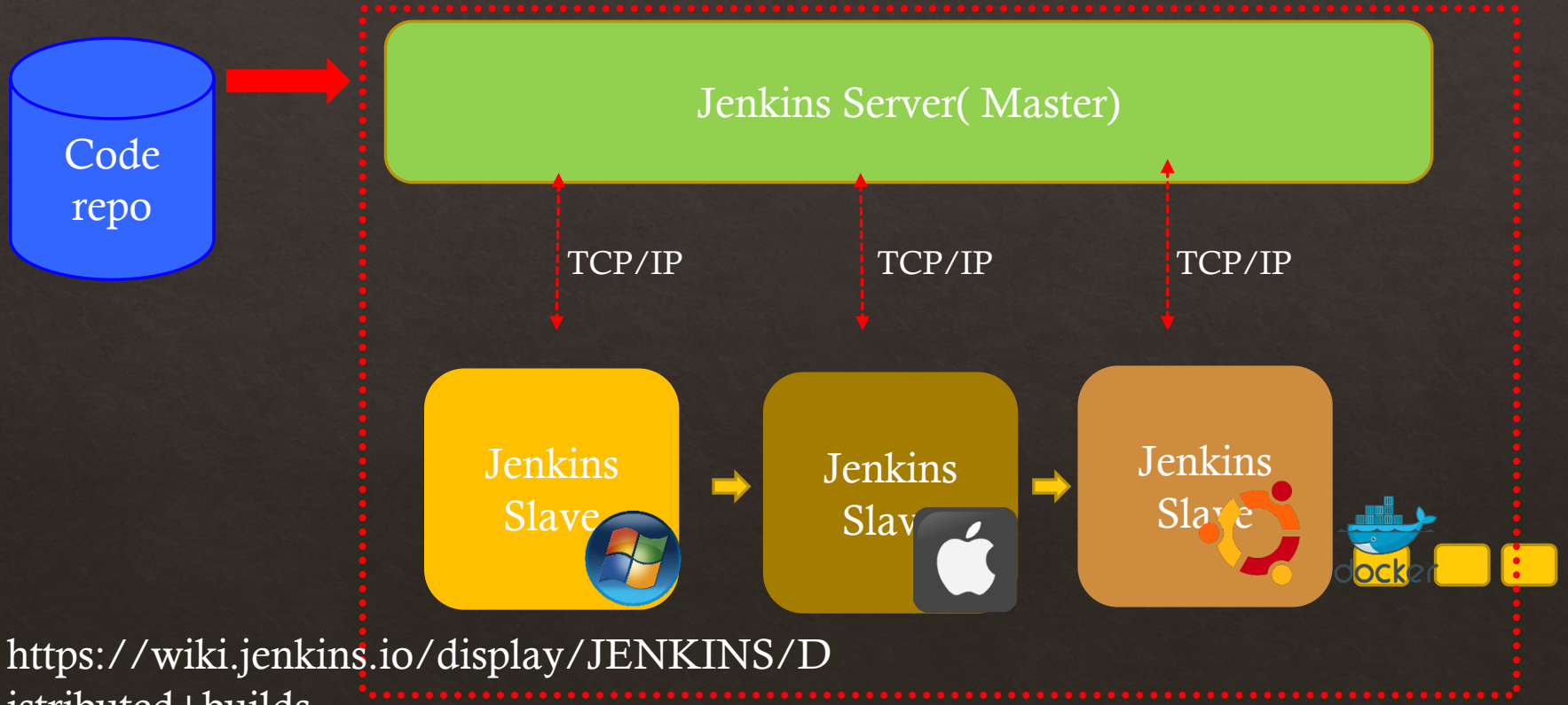
DECLARATIVE PIPELINE SYNTAX - OPTIONAL

- **environment -** A sequence of "key = value" pairs to define environment variables
  - **credentials('<id>') -** Bind credentials to variable
- **options -** Options for the entire Pipeline. For example:
  - **skipDefaultCheckout -** Disable auto checkout SCM
  - **timeout -** Sets timeout for entire Pipeline
  - **buildDiscarder -** Discard old builds
  - **disableConcurrentBuilds -** Disable concurrent Pipeline runs
- **tools -** Installs pre-defined tools to be available on PATH
- **triggers -** Triggers for Pipeline based on schedule, polling, etc.
- **parameters -** Pipeline parameters that are prompted for at run time
- **post -** Defines actions to be taken when pipeline or stage completes based on outcome. Conditions execute in order:
  - **always -** Run regardless of Pipeline status
  - **changed -** Run if the result of Pipeline has changed from last run
  - **success -** Run if Pipeline is successful
  - **unstable -** Run if Pipeline result is unstable
  - **failure -** Run if the Pipeline has failed

Jenkins 的Master 和 Slaves 架构



Jenkins 的Master 和 Slaves 架构



https://wiki.jenkins.io/display/JENKINS/Distributed+builds

Demo

测试用例类型

测试体系

静态代码扫描	功能测试	其他测试
• 安全性扫描 • 代码格式检测 • 3rd party 依赖安全扫描	• 单元测试 • API 接口测试 • 集成测试 • 端对端测试 • 系统测试	• 性能测试 • 压力测试 • 兼容性测试 • 稳定性测试 • …

测试和CI /CD

Commit code

Pull request

Build

静态扫描
单元测试

API 接口测试
端对端测试
集成测试

性能测试
压力测试
兼容性测试
稳定性测试

CI

CI

CD

各个构建工具对比

- Github Actions
- 配置简单
- 无需第三方工具的集成

- Jenkins
- 插件灵活
- 自定义，自维护，分布式部署

- Travis CI
- 第三方CI/CD 平台，构建相对简单，可能会取消免费支持

如何选择合适的CI/CD 构建工具？

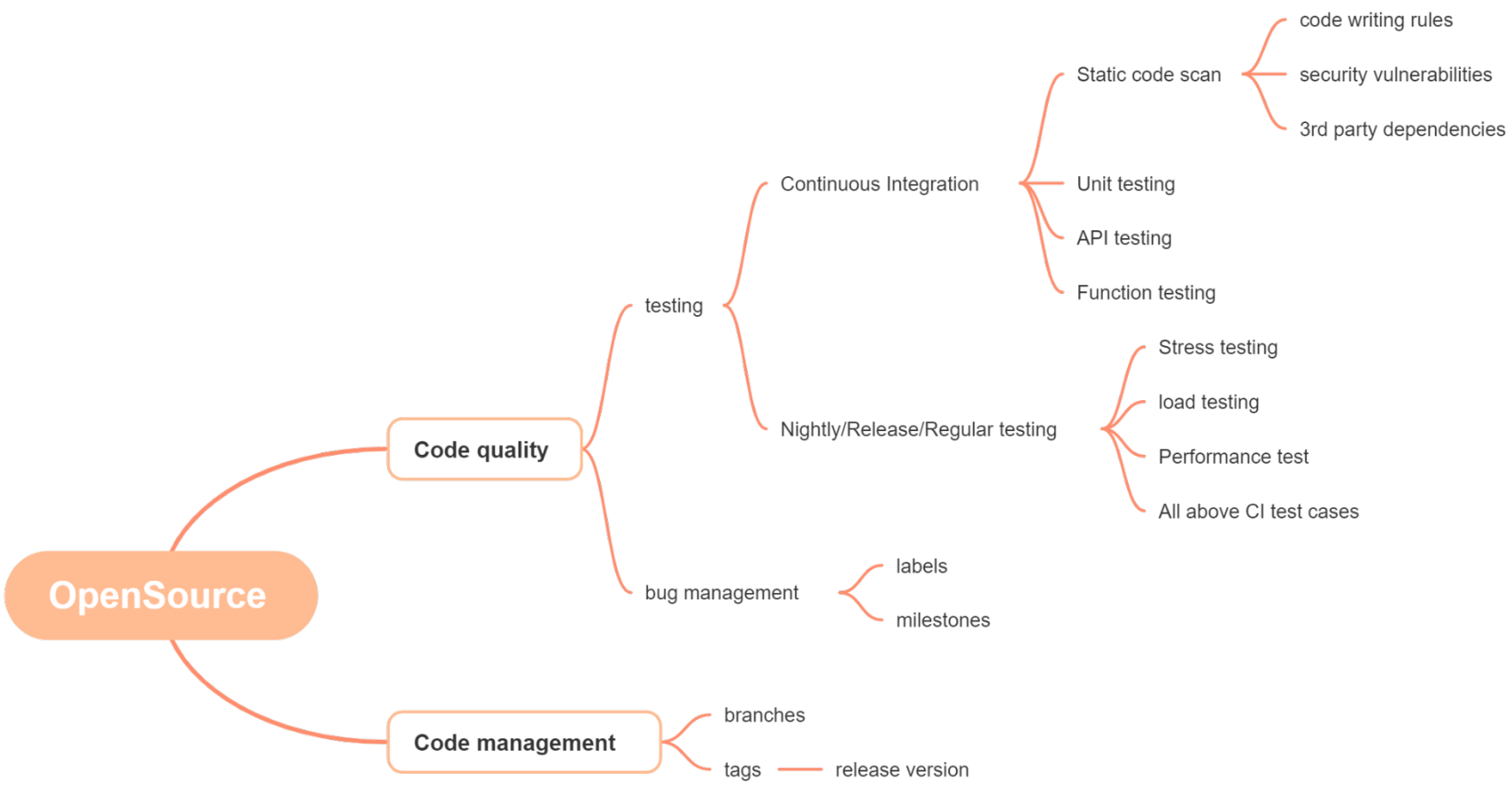
项目一	项目二	项目三
• 单一语言，多平台 • 对编译机器性能要求低，对第三方依赖要求低	• 多语言，单一平台 • 对编译机器性能要求高，对第三方依赖要求高	• C/S mode • 多语言，多平台，多模块 • 第三方依赖重，对编译机器性能要求高，定制化要求高

Back up

Bug /Issues 管理

- 1. 标题简洁规范
- 2. 内容清楚
  - 1. 问题的大概原因
  - 2. 对应的软件版本，硬件环境描述
  - 3. 重现的步骤
  - 4. 问题出现时候的log , 截图信息等

- 1. 及时查看Issues 内容，使用Tag进行分类
- 2. 分配到合理的模块管理人员
- 3. 设置milestones 设置解决期限
- 4. 对应的PR 如果解决之后要及时关闭bug


OpenSource
Code quality
testing
Continuous Integration
Static code scan
code writing rules
security vulnerabilities
3rd party dependencies
Unit testing
API testing
Function testing
Nightly/Release/Regular testing
Stress testing
load testing
Performance test
All above CI test cases
bug management
labels
milestones
Code management
branches
tags
release version