

openEuler社区成员管理

George

2022/04/14





- 01** / 社区成员管理现状和问题
- 02** / 希望的社区成员角色划分
- 03** / 成员管理优化方案
- 04** / 落地实际案例

● 社区成员管理现状和问题

1. 整个社区开发者只有两个角色：SIG组maintainer、代码开发者
2. Maintainer需要管理SIG组下所有仓库的相关审核
3. 部分SIG组有需要划分权限的需求，例如OpenStack, Ruby等
4. 版本管理层面希望能看到代码仓库的直接维护者
5. 社区成员管理需要有晋升机制

● 希望的社区成员角色划分



通过和多个sig组讨论，并报TC确认，openEuler社区开发者计划新划分社区成员角色：

- 1.Maintainer: 基本保持原Maintainer角色，定位SIG组的牵引者、规划者；
对SIG下所有仓库都有代码评审合入权限；
- 2.Committer: 新增角色，部分仓库的看护者；作为部分仓库的第一责任人；
对指定仓库有代码评审合入权限；
同时也是SIG组Maintainer的后备力量
在版本管理层面，通过committer确认版本问题的闭环责任人
- 3.Contributor: 新增角色，部分仓库的重要贡献者，也是这部分仓库Committer的后备力量；
在版本管理层面，通常是代码仓库问题主要修复者和代码开发者；
- 4.Developer: 已有角色，仓库的代码贡献者；

● 成员管理优化方案

1. 每个SIG组新增sig_info.yaml文件作为社区成员管理载体；
2. 只有当前SIG目录下OWNER文件被删除后， sig_info.yaml信息才会生效；
OWNER文件存在是继续保持原功能；
3. maintainer对sig组所有仓库都有合入权限（继承员逻辑）；——必选项
4. 可以在部分仓库下添加这些仓库特有的committer，仅有这一部分仓库的合入权限；——可选项
5. 部分仓库下可新增contributor字段，无代码合入权限；作为仓库的主要贡献者列出；——可选项
6. 新增admin字段，部分仓库需要有仓库管理员权限，直接push代码；——可选项；
7. 为缓解各sig组提交sig_info.yaml的困难，门禁组会按照当前OWNER文件信息和仓库信息，自动为sig组创建sig_info.yaml

● 落地实际案例

```
name: sig-ruby
description: Developed by Yukihiro "Matz" Matsumoto, ruby combines his favorite languages (Perl, Smalltalk, Eiffel, ADA and LISP) to create a new language with both functional and imperative programming features.
mailing_list: Ruby@openeuler.org
meeting_url: NA
mature_level: startup
maintainers:
- gitee_id: shinwell_hu
  name: xinwei Hu
  organization: Huawei
  email: huxinwei@huawei.com
- gitee_id: small_leek
  name: senlin Xia
  email: xiasenlin1@huawei.com
repositories:
- repo:
  - src-openeuler/ruby-augeas
  - src-openeuler/ruby-common
  ...
  - src-openeuler/rubygem-arel
  - src-openeuler/jruby
committers:
- gitee_id: ruebb
  name: yiru Wang
  organization: Huawei
  email: wangyirul@huawei.com
contributors:
- gitee_id: caodongxia
  name: dongxia Cao
  organization: Huawei
  email: caodongxia@h-partners.com
- gitee_id: lyn1001
  name: yanan Li
  organization: Huawei
  email: liyanan32@h-partners.com
- repo:
  - src-openeuler/rubygem-actionable
  - src-openeuler/rubygem-actionview
  ...
  - src-openeuler/rubygem-ZenTest
committers:
- gitee_id: jxy_git
  name: xinyu Jiang
  organization: KylinSoft
  email: jiangxinyu@kylinos.cn
contributors:
- gitee_id: liqiyu123
  name: qiyu Li
  organization: KylinSoft
  email: liqiyu@kylinos.cn
- gitee_id: yangzhao_kl
  name: zhao Yang
  organization: KylinSoft
  email: yangzhao1@kylinos.cn
```

<https://gitee.com/openeuler/community/blob/master/sig-sig-ruby/sig-info.yaml>



● 遗留

1. Sig 组成立时间加入到sig_info.yaml

THANKS

汇报人: xxx

汇报时间: 2021/02/19

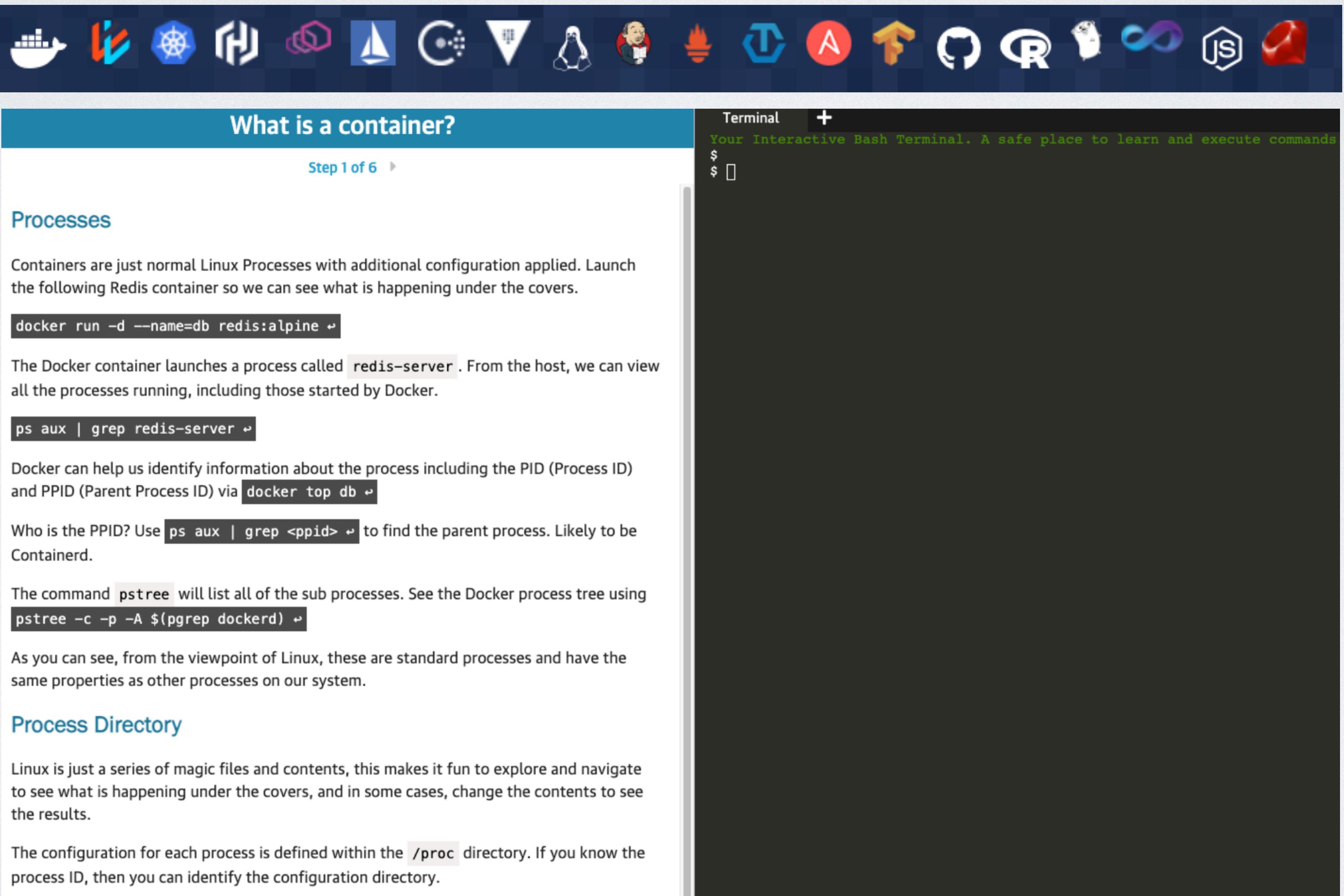


OPENEULER MOOCSTUDIO

集计算机课程编写和学习一体的教育平台

背景

<https://www.katacoda.com/>



The screenshot shows the Katacoda platform interface. At the top, there's a navigation bar with various icons representing different tools or services. Below the navigation bar, the main area has a title "What is a container?" and a subtitle "Step 1 of 6". The content is divided into sections: "Processes" and "Process Directory". In the "Processes" section, there's a terminal window titled "Terminal" with the message "Your Interactive Bash Terminal. A safe place to learn and execute commands". Inside the terminal, there are two command-line examples: "docker run -d --name=db redis:alpine" and "ps aux | grep redis-server". The text explains that Docker containers are just normal Linux processes and demonstrates how to view them from the host system. It also mentions using "ps aux | grep <ppid>" to find the parent process of a Docker process. In the "Process Directory" section, it's mentioned that Linux uses magic files and contents in the /proc directory to represent processes.

What is a container?

Step 1 of 6

Processes

Containers are just normal Linux Processes with additional configuration applied. Launch the following Redis container so we can see what is happening under the covers.

```
docker run -d --name=db redis:alpine
```

The Docker container launches a process called `redis-server`. From the host, we can view all the processes running, including those started by Docker.

```
ps aux | grep redis-server
```

Docker can help us identify information about the process including the PID (Process ID) and PPID (Parent Process ID) via `docker top db`

Who is the PPID? Use `ps aux | grep <ppid>` to find the parent process. Likely to be Containerd.

The command `pstree` will list all of the sub processes. See the Docker process tree using `pstree -c -p -A $(pgrep dockerd)`

As you can see, from the viewpoint of Linux, these are standard processes and have the same properties as other processes on our system.

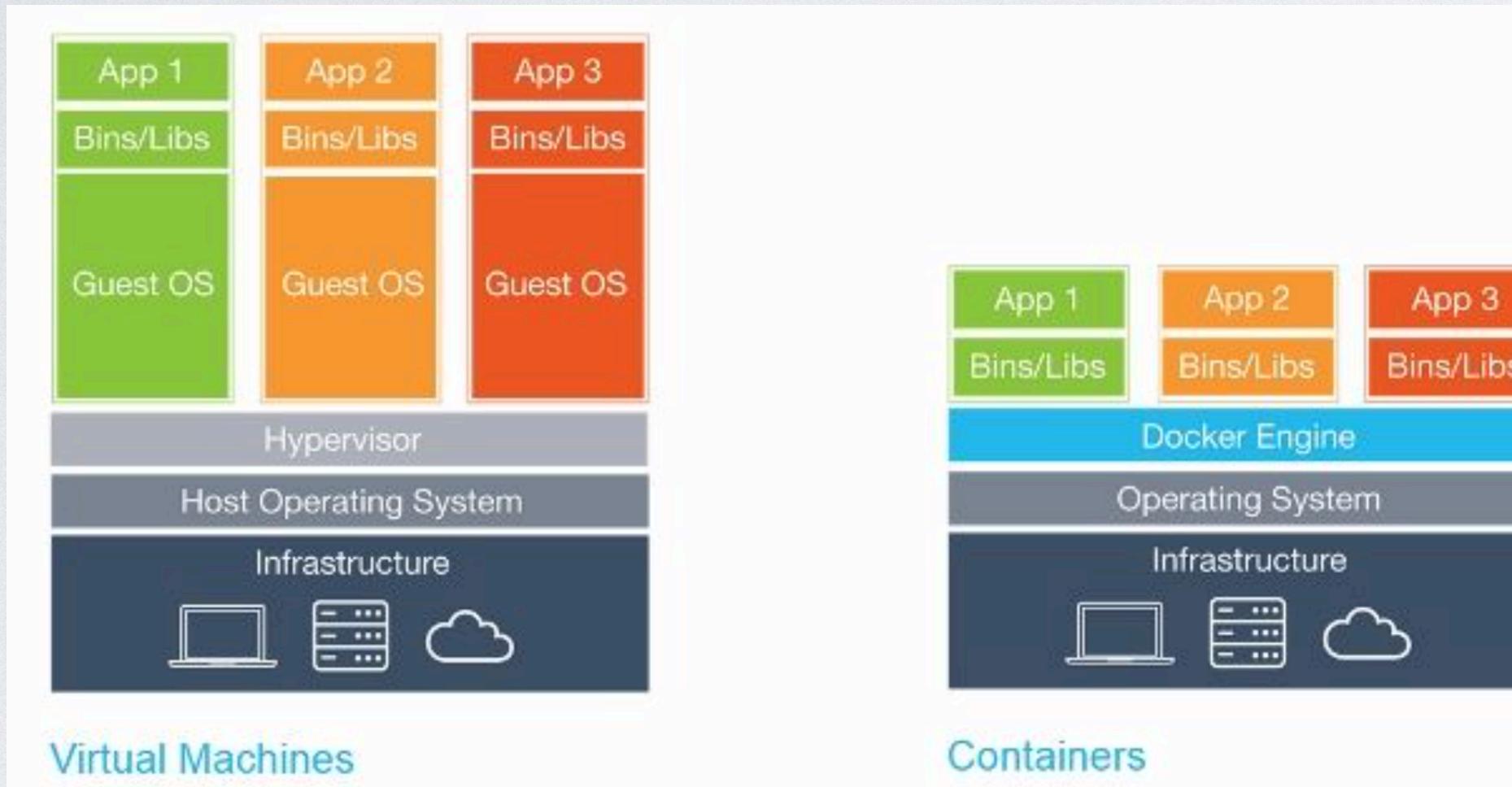
Process Directory

Linux is just a series of magic files and contents, this makes it fun to explore and navigate to see what is happening under the covers, and in some cases, change the contents to see the results.

The configuration for each process is defined within the `/proc` directory. If you know the process ID, then you can identify the configuration directory.

MOOCSTUDIO 技术路线

底层环境



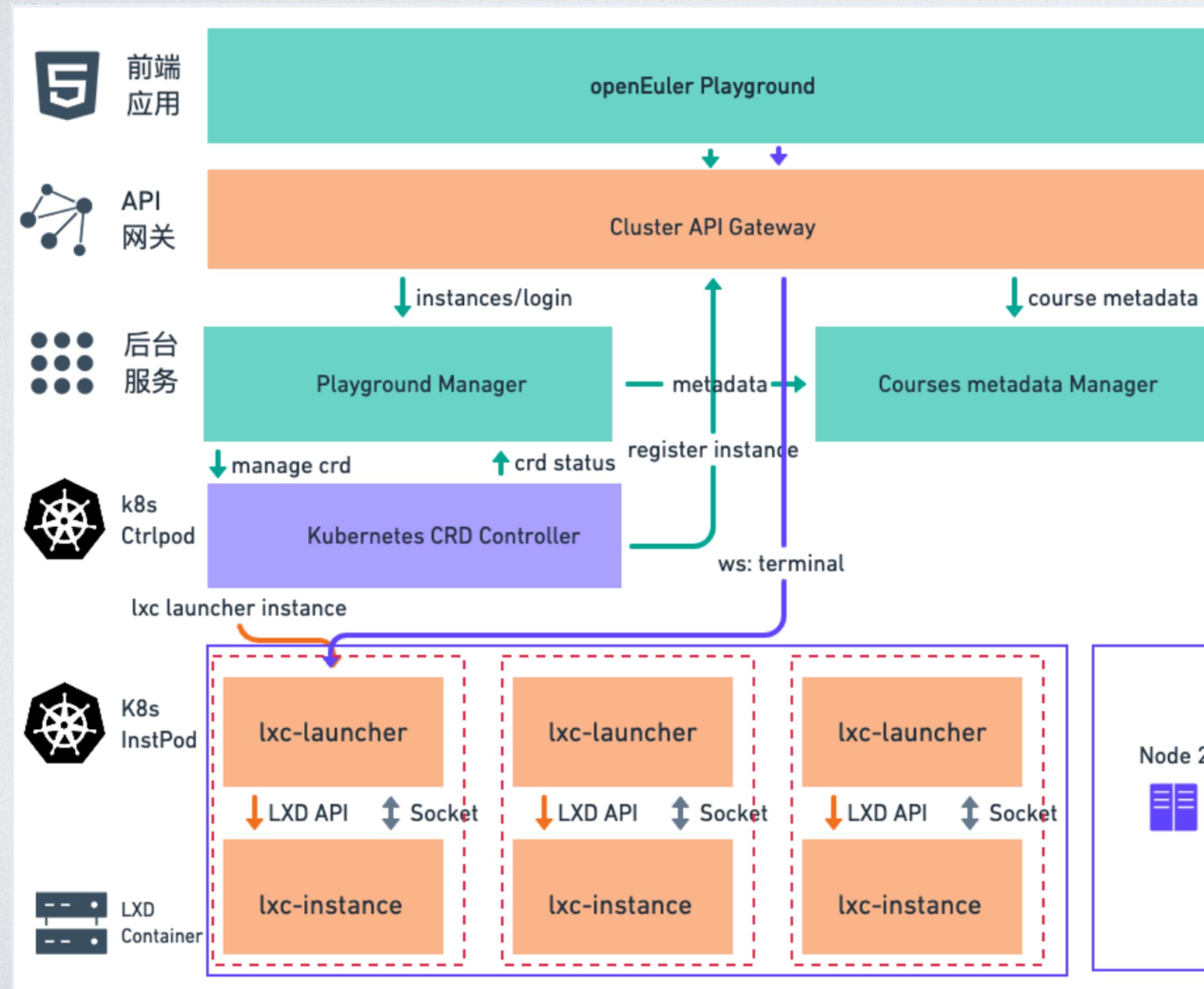
虚拟机VS容器



应用容器VS系统容器

MOOCSTUDIO 技术路线

架构



特点

- 课程自定义，支持用户在社区提交自己的课程和环境。
- 环境多类型支持，支持应用容器，系统容器以及虚拟机支持，支持X86&Arm。
- 支持用户端口，支持环境资源分级销毁。

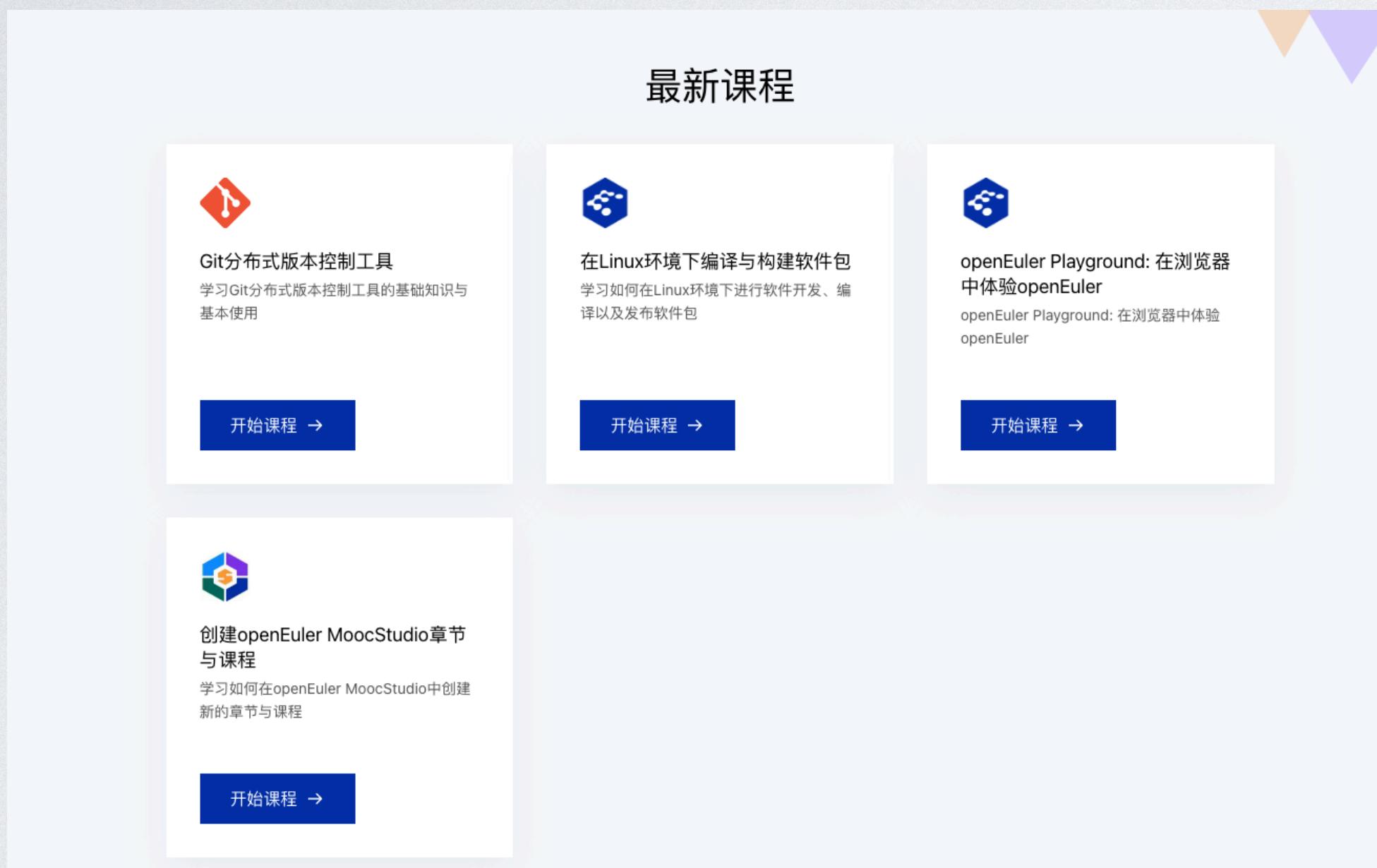
核心组件

- Playground App: 纯前端应用，基于Vue.js，负责课程展示和用户交互。
- Playground Manager: 面向用户和前端的应用，登录，用户管理，课程管理，环境管理等。
- Courses Metadata Manager: 负责课程和环境信息同步，运营人员负责在Gitee上维护课程信息，服务会实时更新和转换数据。
- Courses Operator: 负责CRD环境相关资源(ingress/configmap/volume/pod)的准备和清理。
- Lxc Launcher: LXD Instance和容器环境的一对一维护，同时也负责LXD Instance的网络代理。
- Lxc Manager: LXD镜像的自动同步，异常资源清理。
- Distrobuilder: 用户自定义环境的LXD镜像构建。
- Gotty: Web terminal

MOOCSTUDIO DEMO

课程列表

最新课程



Git分布式版本控制工具
学习Git分布式版本控制工具的基础知识与基本使用
[开始课程 →](#)

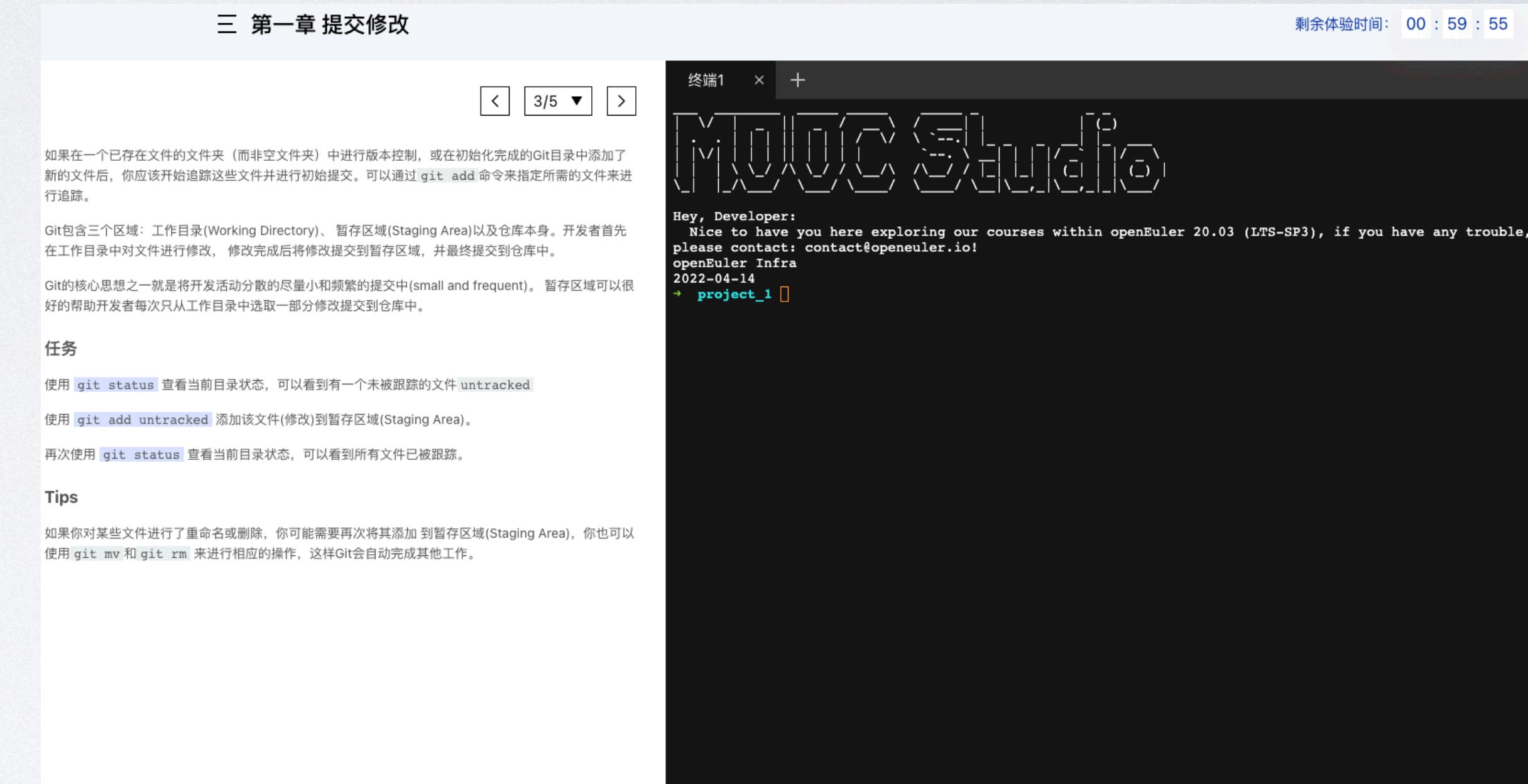
在Linux环境下编译与构建软件包
学习如何在Linux环境下进行软件开发、编译以及发布软件包
[开始课程 →](#)

openEuler Playground: 在浏览器中体验openEuler
openEuler Playground: 在浏览器中体验openEuler
[开始课程 →](#)

创建openEuler MoocStudio章节与课程
学习如何在openEuler MoocStudio中创建新的章节与课程
[开始课程 →](#)

课程演示

三 第一章 提交修改



剩余体验时间: 00 : 59 : 55

终端1 × +

```
Hey, Developer:  
Nice to have you here exploring our courses within openEuler 20.03 (LTS-SP3), if you have any trouble,  
please contact: contact@openeuler.io!  
openEuler Infra  
2022-04-14  
→ project_1
```

如果在一个已存在文件的文件夹（而非空文件夹）中进行版本控制，或在初始化完成的Git目录中添加了新的文件后，你应该开始追踪这些文件并进行初始提交。可以通过`git add`命令来指定所需的文件来进行追踪。

Git包含三个区域：工作目录(Working Directory)、暂存区域(Staging Area)以及仓库本身。开发者首先在工作目录中对文件进行修改，修改完成后将修改提交到暂存区域，并最终提交到仓库中。

Git的核心思想之一就是将开发活动分散的尽量小和频繁的提交(small and frequent)。暂存区域可以很好的帮助开发者每次只从工作目录中选取一部分修改提交到仓库中。

任务

使用`git status`查看当前目录状态，可以看到有一个未被跟踪的文件`untracked`。

使用`git add untracked`添加该文件(修改)到暂存区域(Staging Area)。

再次使用`git status`查看当前目录状态，可以看到所有文件已被跟踪。

Tips

如果你对某些文件进行了重命名或删除，你可能需要再次将其添加到暂存区域(Staging Area)，你也可以使用`git mv`和`git rm`来进行相应的操作，这样Git会自动完成其他工作。

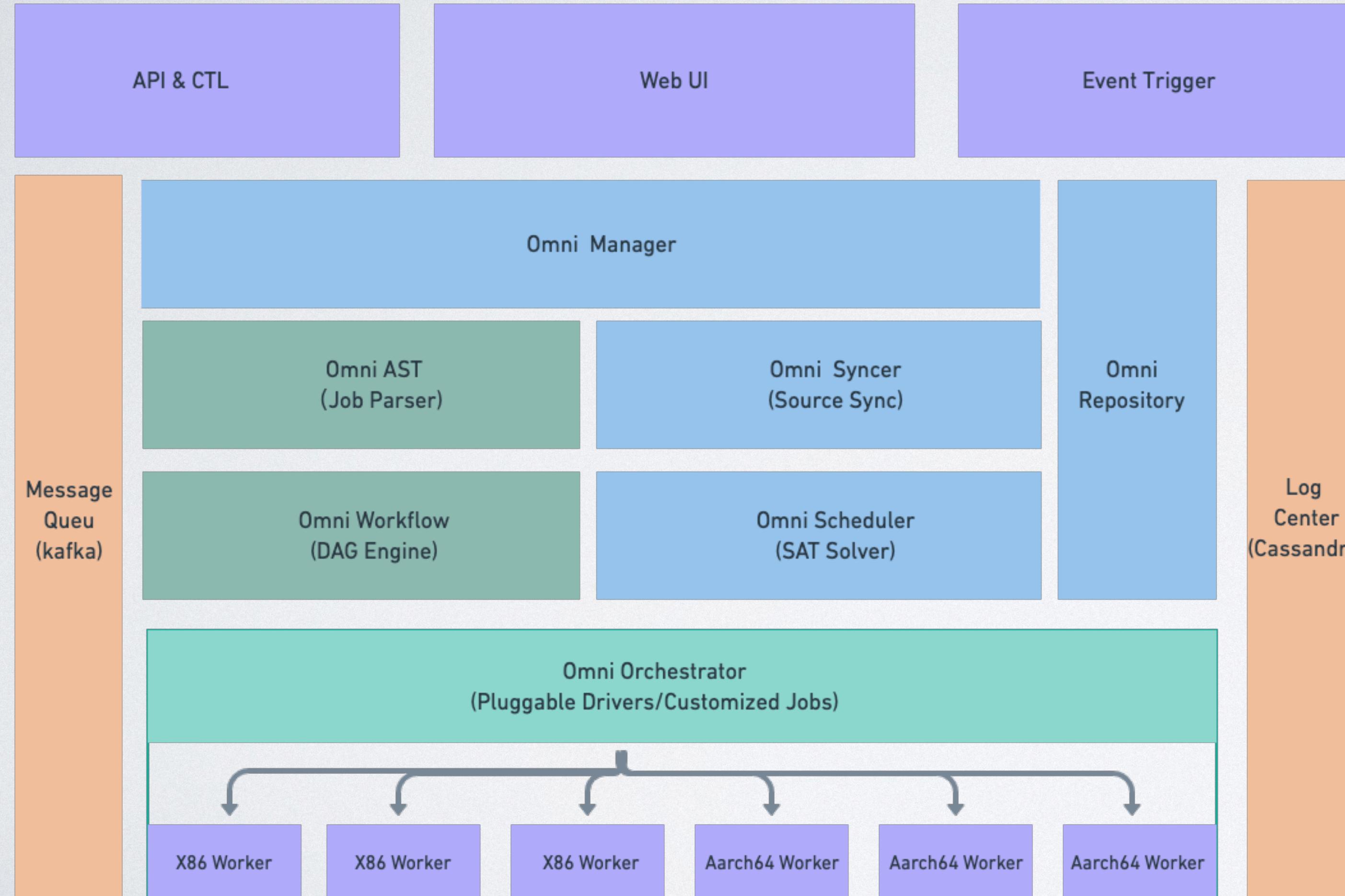
MOOCSTUDIO 未来规划

TODOS

1. 多社区支持...
2. 课程打分...
3. 成长和荣誉系统
4. 视频, 编码, 等多资源支持
5. 虚拟机, Arm支持

Omni Build Platform

OBP是基础设施Sig主导的openEuler社区统一构建平台，旨在提供软件包持续构建，操作系统镜像自定义构建，自定义CICD任务等多种构建任务。



背景:

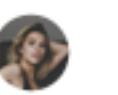
1. 社区构建系统臃肿，代码可维护性差，不方便新功能迭代演进。
2. 社区内缺少开放可定制的CICD系统。

核心组件:

1. **Omni Scheduler:** 基于openSUSE的LibSolv完成包的依赖分析，调度排序，后续重点优化算法和调度性能瓶颈。
2. **Omni Imager:** 基于Calamares提供高度可定制的ISO Installer制作能力。
3. **Omni Orchestrator:** 基于Kubernetes/cadence/CompassCI提供可扩展且支持异构环境的任务执行引擎。
4. **Omni Workflow:** 支持用户配置自定义CICD任务，兼容Github Action。

计划:

1. 2022 Q2完成操作系统镜像构建，可定制，可自由添加包，修改。
2. 2022 Q4完成操作系统包持续构建，支X86&Aarch64。
3. 2023 支持自定义CICD任务。



概览

Build Image

Job List

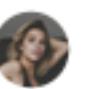


新建任务

FAQ

最近任务

Status	类型	架构	ID	Name	Description	Last Duration	操作	Download	查看详情
✓	iso	X86	009892	xiaotiancai	Engine	3h45min	rebuild delete	下载	查看详情
✗	vhd	ARM	009892	xiaotiancai	Engine	50min	rebuild delete	下载 <small>②</small>	查看详情
✓	qcow2	MIPS	009892	xiaotiancai	Engine	1min	rebuild delete	下载	查看详情
✗	raw	POWER	009892	xiaotiancai	Engine	8.8sec	rebuild delete	下载 <small>②</small>	查看详情
✓	iso	ALPHA	009892	xiaotiancai	Engine	50min	rebuild delete	下载 <small>②</small>	查看详情
✗	iso	Z80	009892	xiaotiancai	Engine	20min	rebuild delete	下载	查看详情



概览

Build Image

Job List

Search for...

删除

<input type="checkbox"/>	Status	类型	架构	ID	Name	Description	Last Duration	操作	Download	查看详情
<input checked="" type="checkbox"/>		<input type="checkbox"/> iso <input type="checkbox"/> vhd <input type="checkbox"/> qcow2 <input type="checkbox"/> raw	X86	009892	xiaotiancai	Engine	3h45min	rebuild delete	下载	查看详情
<input checked="" type="checkbox"/>		<input type="checkbox"/> raw	ARM	009892	xiaotiancai	Engine	50min	rebuild delete	下载 ?	查看详情
<input checked="" type="checkbox"/>		<input type="checkbox"/> iso <input type="checkbox"/> vhd <input type="checkbox"/> qcow2 <input type="checkbox"/> raw	MIPS	009892	xiaotiancai	Engine	1min	rebuild delete	下载	查看详情
<input type="checkbox"/>		<input type="checkbox"/> raw	POWER	009892	xiaotiancai	Engine	8.8sec	rebuild delete	下载 ?	查看详情
<input type="checkbox"/>		<input type="checkbox"/> iso	ALPHA	009892	xiaotiancai	Engine	50min	rebuild delete	下载 ?	查看详情
<input type="checkbox"/>		<input type="checkbox"/> iso	Z80	009892	xiaotiancai	Engine	20min	rebuild delete	下载	查看详情
<input checked="" type="checkbox"/>		<input type="checkbox"/> iso	X86	009892	xiaotiancai	Engine	3h45min	rebuild delete	下载	查看详情
<input checked="" type="checkbox"/>		<input type="checkbox"/> vhd	ARM	009892	xiaotiancai	Engine	50min	rebuild delete	下载 ?	查看详情
<input checked="" type="checkbox"/>		<input type="checkbox"/> qcow2	MIPS	009892	xiaotiancai	Engine	1min	rebuild delete	下载	查看详情
<input type="checkbox"/>		<input type="checkbox"/> raw	POWER	009892	xiaotiancai	Engine	8.8sec	rebuild delete	下载 ?	查看详情
<input type="checkbox"/>		<input type="checkbox"/> iso	ALPHA	009892	xiaotiancai	Engine	50min	rebuild delete	下载 ?	查看详情
<input type="checkbox"/>		<input type="checkbox"/> iso	Z80	009892	xiaotiancai	Engine	20min	rebuild delete	下载	查看详情



概览

Name

Search for...

Build Image

Job List

Description

Search for...

Build

Stop

Download

Configure

Architecture

X86_64



Release Version

openEuler 22.03 LTS



Output Format

Calamares Installer ISO



Group:

Development



Custom:

Search for...

- Package xxxxxxxxxxxx

加入 →

← 减去

Search for...

- Package xxxxxxxxxxxx

Packages

Default:

WARNING: The packages in the below list is the default package list to build up a basic usable openEuler dotnbutilon, please do nor modify the below list unless you are aware of what you are doing

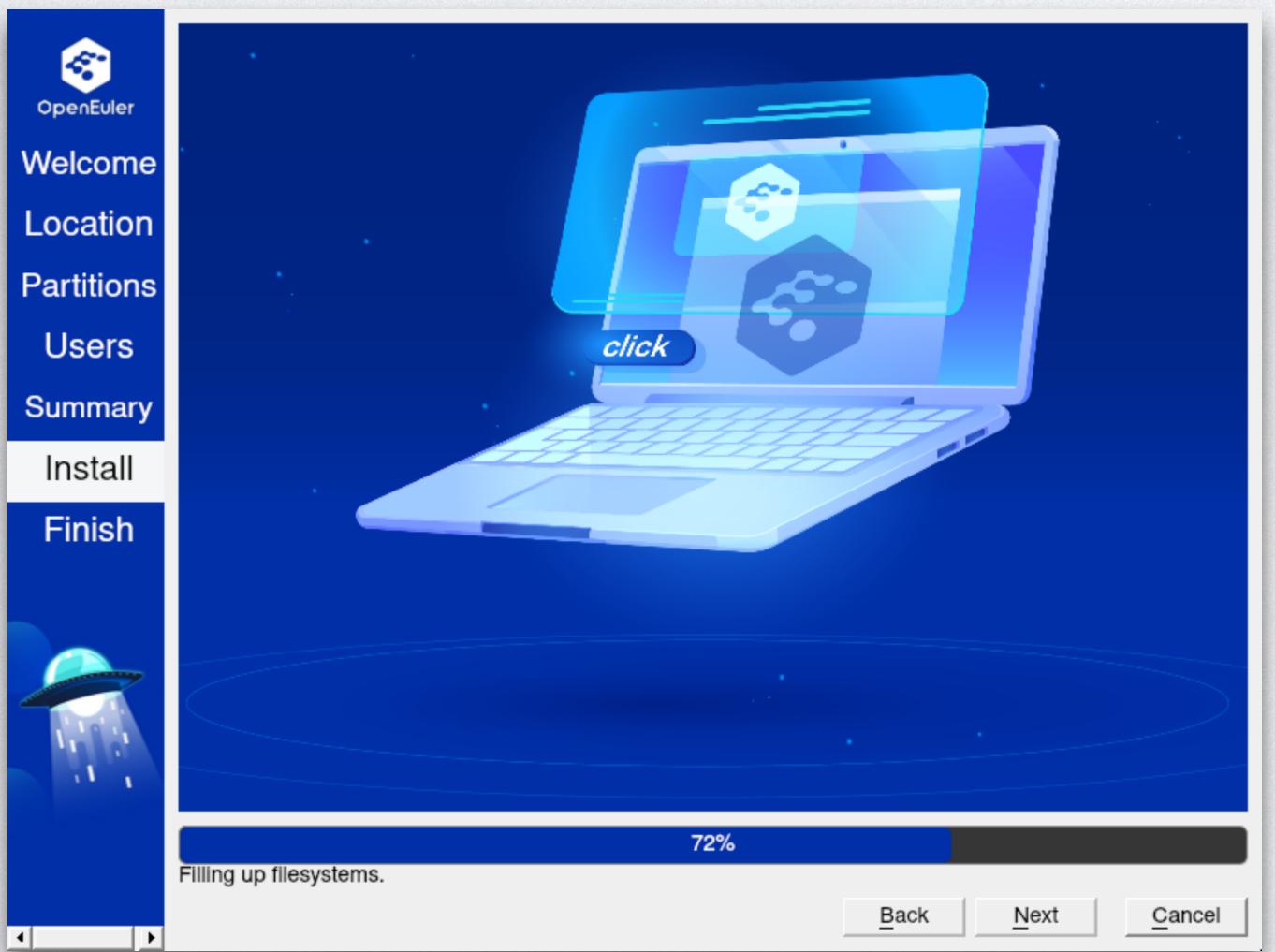
filesystem
glibc
systemd
kernel
...

filesystem
glibc
systemd
kernel
...

filesystem
glibc
systemd
kernel
...

FAQ

Demo/Screen

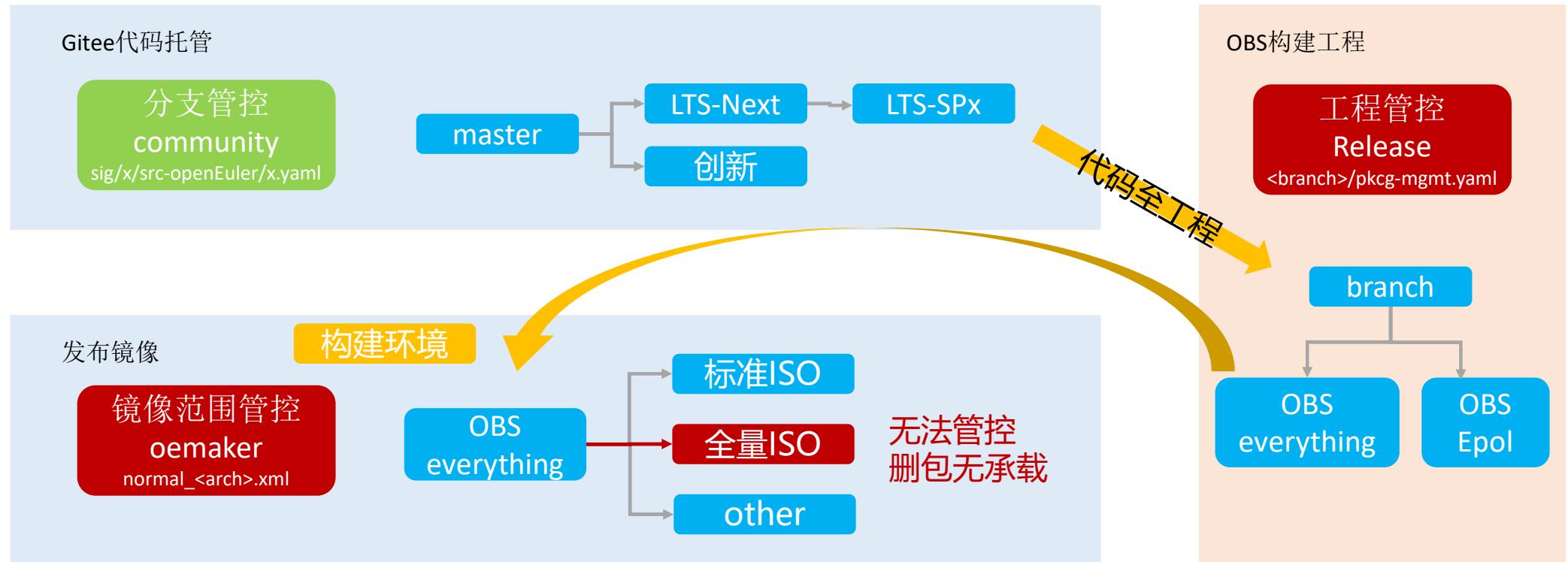


openEuler Installer ISO

The screenshot shows the OmniBuildPlatform interface for ISO Customized Build. At the top, there are dropdown menus for Architecture (X86_64), Release Version (openEuler 22.03 LTS), and Output Format (Calamares Installer ISO). Below these are three main sections: 'Packages' (Default: filesystem, glibc, systemd, kernel...), 'Custom' (PKG1, PKG2, PKG3...), and 'Group' (Development). The 'Build' section at the bottom shows a progress bar and a log output for 'Set up job' tasks, with a timestamp of 33s.

ISO Customized Build

转测入口问题



问题:

1. 全量ISO范围无承载/强管控
2. 转测软件范围变更无承载 (每轮转测范围)
3. 未发布软件包无记录
4. 软件包维护级别
5. 直至rc4仍有需求合入
6. 构建环境和运行环境不一致

措施建议:

1. 增加全量ISO的黑名单或白名单
2. 在社区承载每轮转测范围变更清单 (明确需求和软件对应关系/发布方式)
3. 社区公示软件包维护级别 (包含新增需求的维护级别)
4. 落实需求合入管控机制
5. 构建环境和运行环境保持一致

测试过程数据 - 社区质量看板

	测试策略	测试用例	测试主体	问题管理	测试分析
现状	1. 测试策略high-level 无法体现实际测试内容	1. 开源测试用例少 2. 用例与策略无对应关系 3. AT测试覆盖不够完整	1. 测试过程不可追溯 2. 测试主体分散 3. 无版本前感知问题的措施	1. 多里程碑同事查看困难 2. 社区新增issue不可可控 3. 社区issue未挂载里程碑	
改进	1. 测试策略与测试用例 对应管理	1. 单包集成用例增加 2. 特性测试用例开源 (各sig) 3. AT测试用例补齐	1. 平台管理测试任务 2. 各特性提供测试结论 3. 每周测试防护网	1. 版本问题总体看板 2. 版本质量策略明确发 布前issue的处理策略 3. 增加社区版本里程碑	
功能诉求	1. 平台承载策略与测试 用例对应关系 2. 版本基线测试策略 (策略模板)	1. 基于权限的用例管理 2. GUI测试框架 (合并 openqa能力) - 短期 获取openqa结果	1. 测试任务管理 (执行 /统计看板) 2. 测试任务自动触发能 力 (可结合compass-ci)	1. 问题自动提单 2. 问题辅助分析能 力 (日志分析, 可结合A- Ops) 3. 问题看板	1. 测试报告 自动生成 2. 基于模块的测试活动 管理能力

测试过程数据 - 缺陷流出

当前rpm包对外接口/功能定义（后续承载在社区）：

1. src.rpm 编译功能
2. rpm 安装/卸载/升级/降级
3. /usr/bin /usr/sbin下提供的命令 help 提供的所有使用方式
4. service的启停

