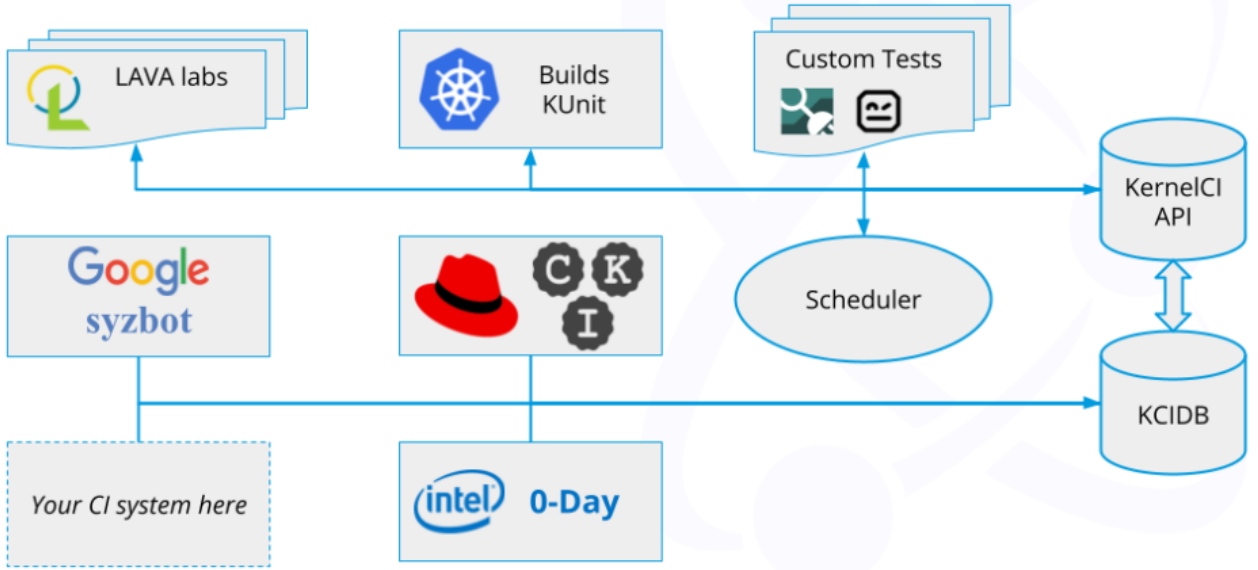
Kernel CI

Kernel CI结构

Documentation | KernelCI

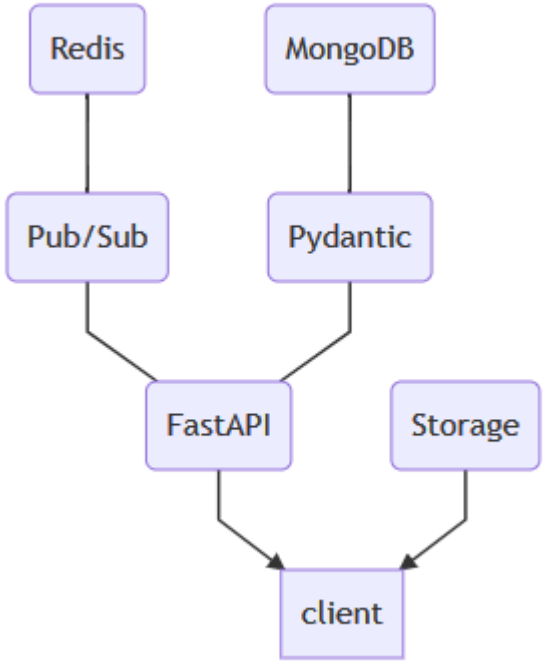
- Api & Pipeline
  几乎可以在任何地方运行

- KCIDB
  其他的完全自主的系统，生成自己的内核构建和测试报告。他们可以把报告上传到KCIDB
  KCIDB是BigQuery数据库（Google Cloud的一个产品）



Api & Pipeline

- API架构



MongoDB 存储各种层次的测试数据，比如内核的修订版、构建和静态测试的结果、运行时功能测试、regression等（regression 是指在旧版本下某些应用或例程（some application or practical use case)工作正常，但是在相似设置的新版本下效率变低）
MongoDB里面存储的是Node对象（kernelci自己定义的对象），对象中有一些标志位用来在流水线中规划顺序
Node对象定义

Pydantic是python的一个库，用来做数据模型验证（schema validation）

Pub/Sub使用Redis实现，然后Redis还作为一个内存数据库（in-memory database）给其他的各种功能使用

FastAPI作为Web框架，处理异步请求。OpenAPI生成相应的文档

Storage上传用ssh，下载用http。用docker-compose搭建Storage、Redis、MongoDB、ssh服务
docker-compose.yaml

Pipeline

流水线里一共有6个服务：

1. Trigger：每隔一段时间检查一次内核是否有新的修订版，如果有进入Tarball服务
2. Tarball：创建‘checkout’节点，设置state=running和result=None。发出事件。创建一个tarball并且上传到storage。上传完成之后进入Runner Service
3. Runner：修改节点state=available，设置超时时间（holdoff）。更新描述和工件（artifacts）。发出事件之后准备Run build/test
4. Runtime environment：进Runtime env之前创建build/test node，设置state=running，result=None，holdoff=None。Runtime env可以是shell，Kubernetes或者LAVA lab。任务跑完之后会更新node的state=done，result=pass/skip/fail
5. Timeout：用来处理holdoff超时，超时的节点连同其父节点和子节点都会被设置为done，发出事件，然后进入Test Report。假如有子节点没有跑完jobs，会设置成closing状态
6. Test Report：检查是不是收到checkout节点，如果是生成测试报告邮件

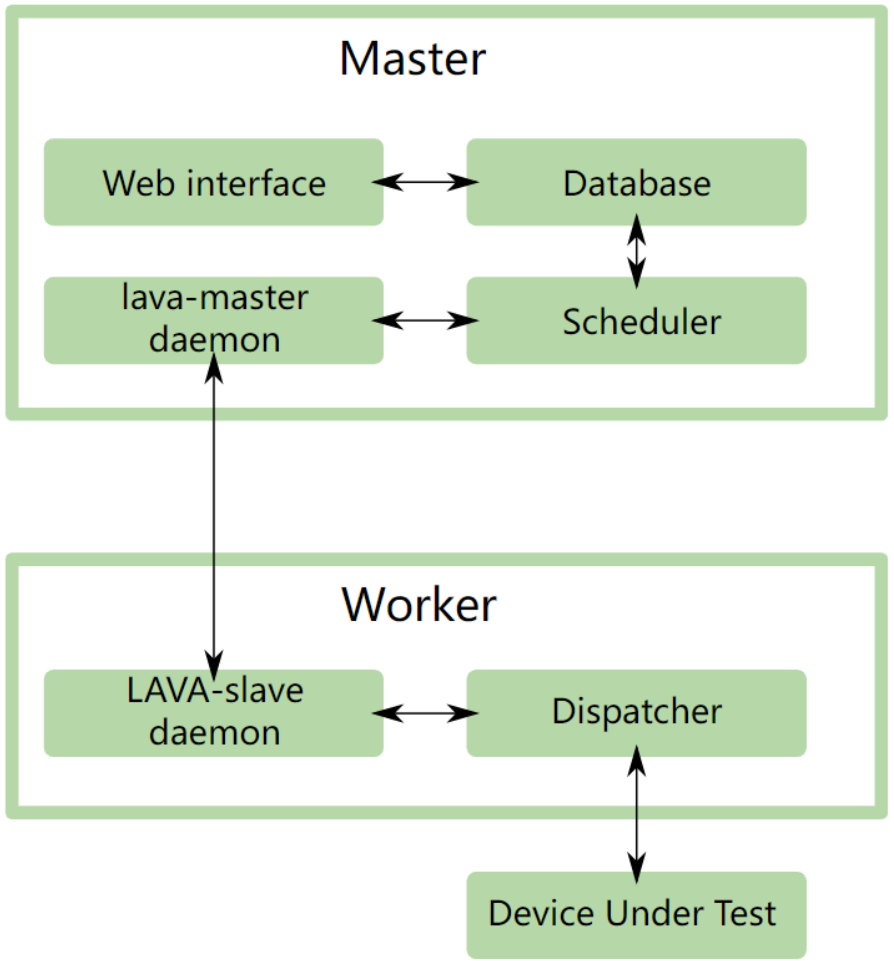
这些服务的搭建也是用docker-compose

详细说明：Pipeline details | KernelCI

Source code: kernelci-pipeline

LAVA

架构



主要分为两个部分，master和worker

Master：

- Web interface：Apache作为web server，使用uWSGI应用服务器和Django web框架。支持Restful API和XML-RPC
- Database：使用运行在Master本地的PostgreSQL
- Scheduler：定时检查数据库，核对排好队的测试任务和可用的测试设备。如果资源都可用，开始测试任务。
- Lava-master daemon：用gunicorn创建的daemon程序，和worker用http通信

Worker:

- Lava-slave daemon：接受Master发来的控制信息
- Dispatcher：将任务分发给不同的硬件

KCIDB

其他独立的CI系统有

1. 红帽的CKI
   cki-project · GitLab
   CKI Project (cki-project.org)
2. 谷歌的syzbot
   syzkaller/docs/syzbot.md at master · google/syzkaller (github.com)
3. Intel的0-Day
   或许是这个intel/lkp-tests: Linux Kernel Performance tests (github.com)
4. Huawei Compass CI（暂时没有给KCIDB提供测试报告）
   compass-ci: Compass CI (gitee.com)

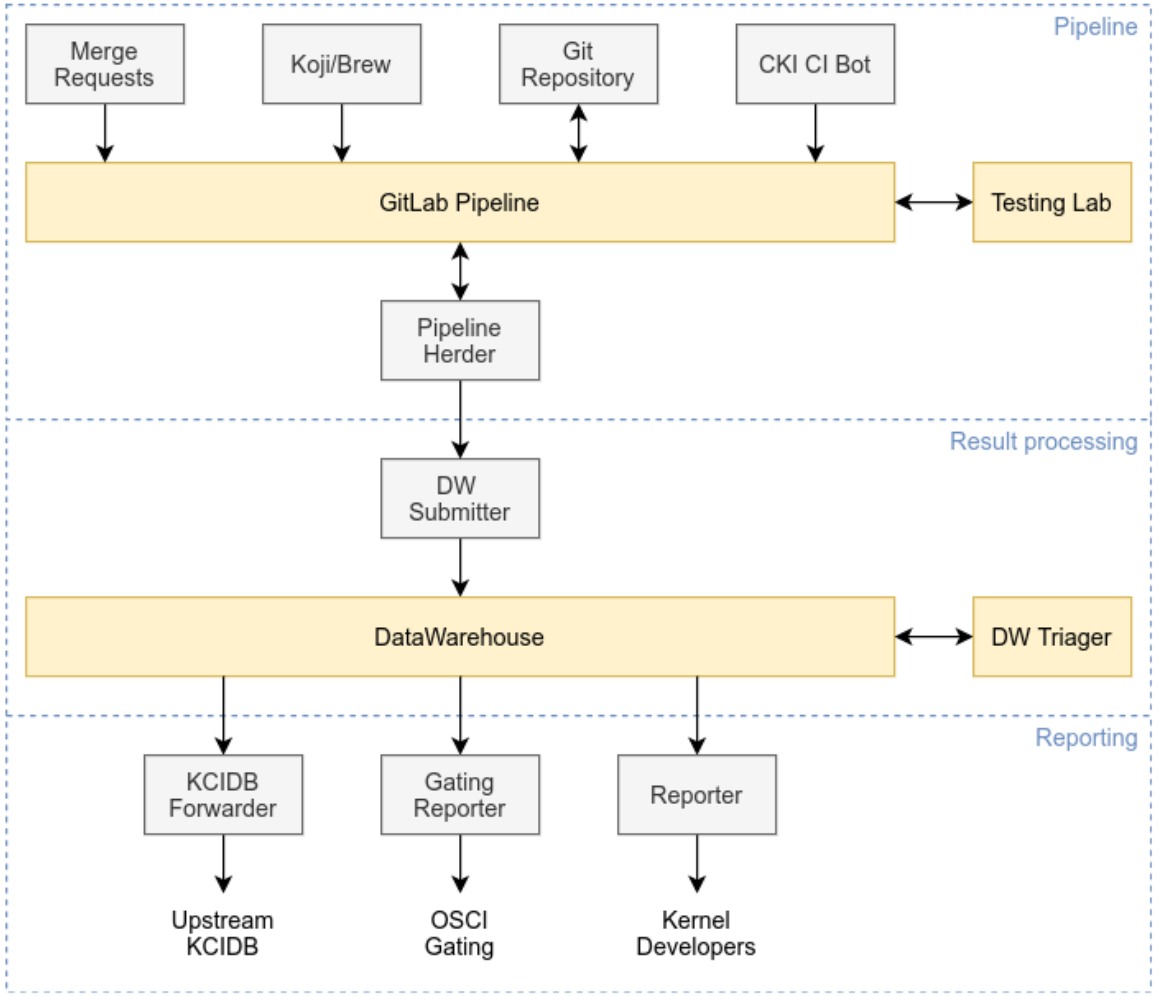
注：intel/lkp-tests和Huawei Compass CI的作者都是大牛吴峰光

CKI

架构

1. GitLab的流水线和Testing Lab
2. DataWarehouse做结果的可视化和审计
3. 调解DataWarehouse周围数据流的各种微服务

架构详情：[CKI architecture | CKI Project (cki-project.org)]



CKI

- 测试用例
  测试包含LTP（Linux Test Project)和kselftest
- 自定义测试
  需要编写测试之后提交PR到红帽的public tests repo（repo里面含有所有的测试）
  编写测试的例子： example

syzbot

作用：持续模糊测试kernel，自动汇报bugs到邮箱列表。通过系统调用的方式测试驱动，主要测试open、write、mmap、ioctl、read

流程：dashboard-app从mailing lists中获取patches，之后将patches和program传给syz-ci，syz-ci将patched kernel和program传给syz-manager，syz-manager创建patches VMs测试。测试之后将logs，coverage，programs返回给syz-manager，syz-manager把数据传给dashboard-app，dashboard-app再把内容给开发者和mailing lists

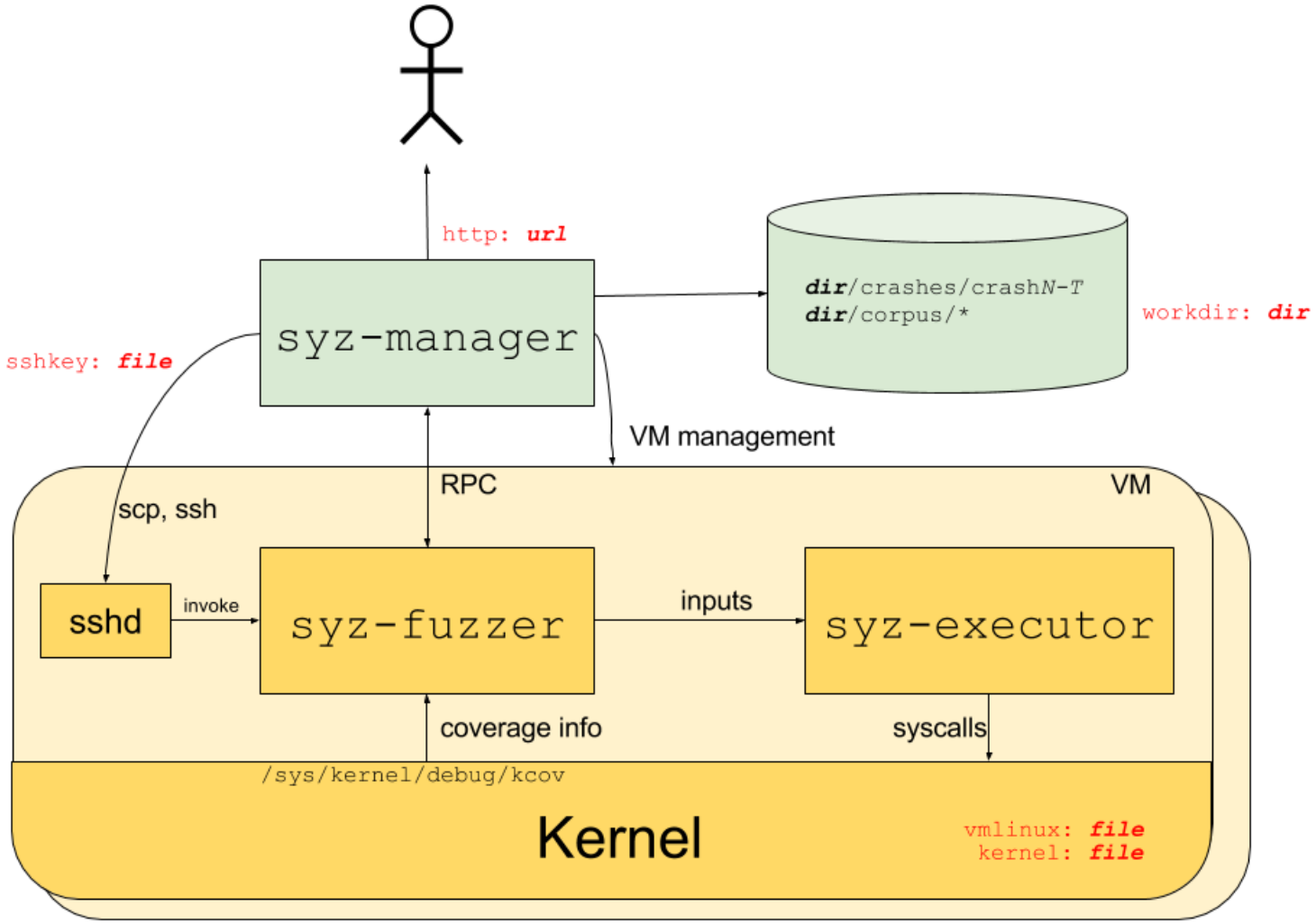
ci由syz-ci执行，Syz-ci向下控制syz-manager去做各种测试。syz-ci不仅获取patches，还会从kernel.org和github中获取kernel和syzkaller binaries

如果需要多个syz-manager，使用syz-hub管理，需要配置hub.cfg

syzbot架构图：[syzkaller/docs/syzbot_architecture.png at master · google/syzkaller (github.com)]

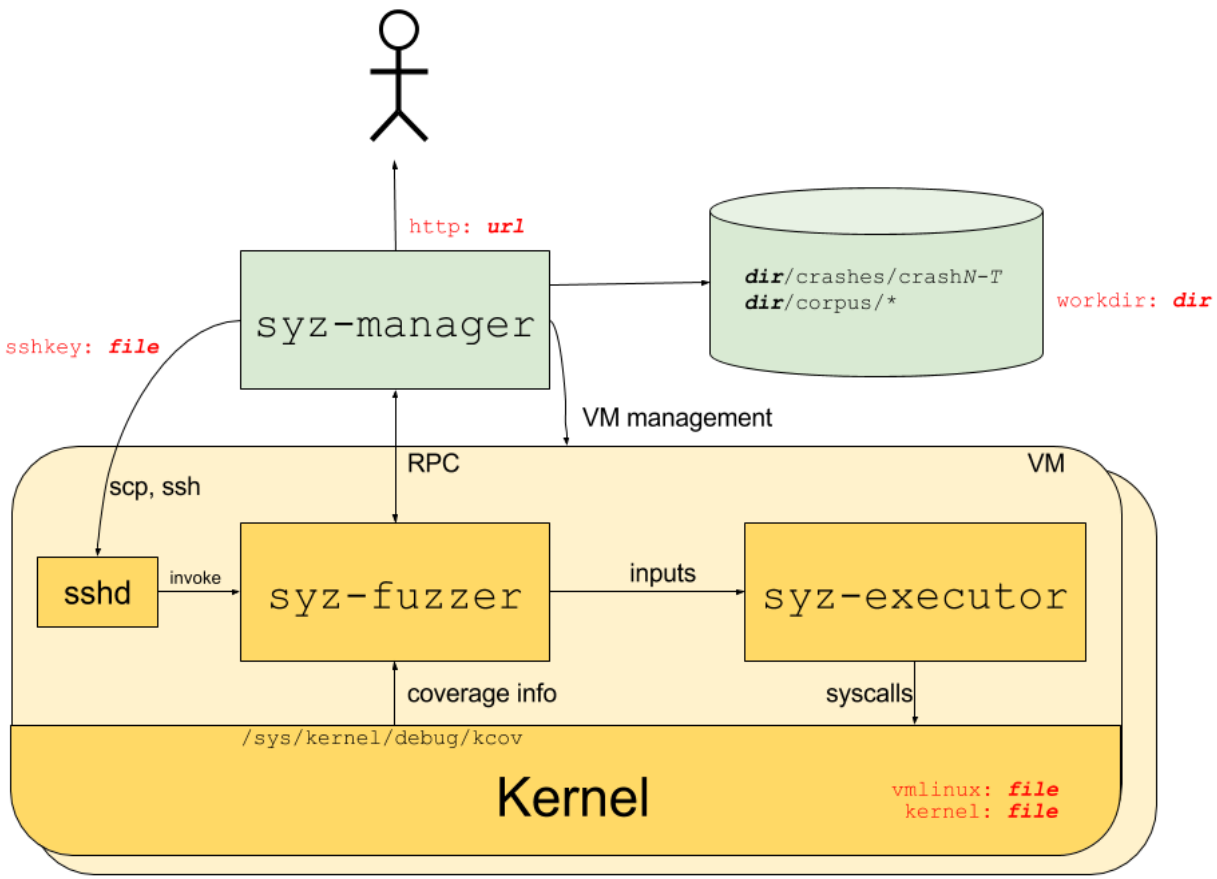
syz-manager结构

启动时可以设定参数--config=xxx.cfg，这个文件内容是json格式文本。
sshkey是宿主机运行ssh-keygen生成的公钥（例如id_rsa.pub），用来连接到sshd
http是syz-manager的web页面url
workdir定义存放测试结果的目录
kernel定义内核编译之后生成的bzImage
详细参数：config.go



Syz-manager

- Syzkaller（泛指syz-manager和其向下创建的整个系统）是基于覆盖引导的fuzzer（向目标程序插桩统计覆盖率，结果反馈给fuzzer，生成更有价值的输入）
- Syz-manager会创建多个VM实例，syz-manager还作为持久语料库，持续给syz-fuzzer发送有价值的输入。Syz-fuzzer负责创建短暂运行的syz-executor。syz-fuzzer把输入给syz-executor让其运行一系列系统调用，也会把引起覆盖的输入返回给syz-manager
- Syzkaller测试是白盒测试，必须有所有源码，并且编译时开启KASAN(Kernel Address SANitizer)和KCOV(KCOV collects and exposes kernel code coverage information in a form suitable for coverage-guided fuzzing.)



0-Day（lkp-tests)

lkp-tests (Linux Kernel Performance tests)

可以运行在：

- Debian sid
- Archlinux
- CentOS7

只需要安装测试需要的所有依赖，即可运行lkp-tests

lkp-tests需要ruby2.x，而Ubuntu22.04软件源中的ruby是3.x版本。需要手动安装ruby2.x。(安装时会有一些依赖问题需要解决)

lkp-tests在Ubuntu20.04上使用更方便，软件源中能直接下载ruby2.x，然后开始测试（不清楚2025.04之后lkp-tests的ruby依赖版本会不会更新）

lkp-tests

测试包含的部分:

- 虚拟内存管理
- I/O子系统
- 进程调度
- 文件系统
- 网络
- 设备驱动

通过yaml文件定义具体的测试

详细测试用例:

lkp-tests/jobs at master · intel/lkp-tests (github.com)

Huawei Compass CI

- Huawei Compass CI在Linaro Connect 2021发布时的文档: Compass CI (linaro.org)
- Huawei Compass CI的测试用例依赖lkp-tests（毕竟都是一个作者）
- 使用ssh登录调测的环境
- 使用web接口分析测试结果
- 通过提交相同的job.yaml实现测试结果的复现
- Huawei Compass CI做了bisect（类似git bisect的作用），出现error id的时候会定位引入该error 的 commit

Huawei Compass CI

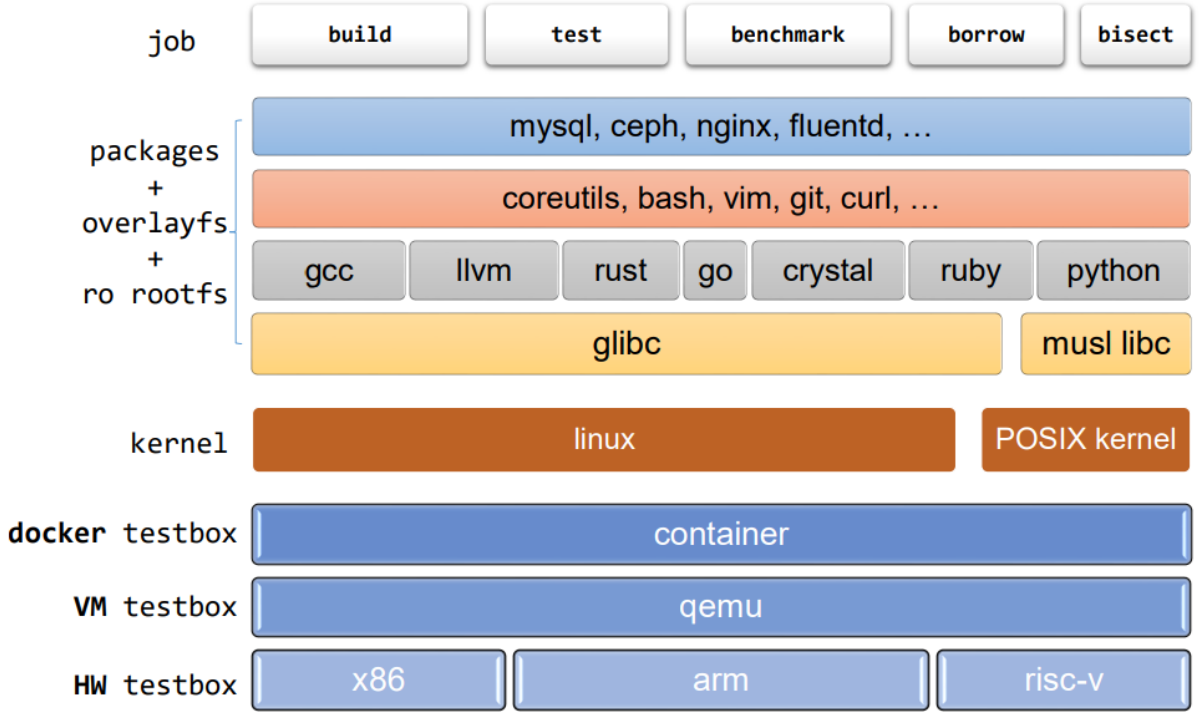
测试矩阵

workloads
X
OS
X
machine

job.yaml

```
script_name:
   param1: ……
   param2: ……

os: ……
kernel: ……
testbox: ……
```



支持的操作系统

os	os_arch	os_version
openeuler	aarch64	1.0
		20.03
		20.09
		daily
	x86_64	20.03
		20.09
centos	aarch64	7.6
		7.8
		8.1
	x86_64	7.6
		7.8
debian	aarch64	sid
	x86_64	sid
archlinux	aarch64	2020-11-12

系统之间的异同

- 0-day（aka ‘lkp-tests’）和Huawei Compass CI的测试用例都是基于lkp-tests的测试用例。lkp-tests的测试用例构建起来很方便
  how to add test case
  Add Test Suite · intel/lkp-tests Wiki (github.com)
  但是syzkaller的自定义测试用例比较麻烦
  syzkaller/docs/syscall_descriptions.md at master · google/syzkaller (github.com)
- Syzkaller是fuzzer，可能会发现内核中比较难发现的bug。Lkp-tests的错误定位能力没有syzkaller那么强

Thanks