

# 基于qemu的大规模randconfig测试系统

余文斌



项目背景

项目方案

项目进展



# 项目背景

- 关键问题：Linux 内核有超过9000个编译选项，它们的随机组合是一个天文数字。
- 项目目标：有效组织测试，充分暴露、定位、报告在不同组合下的内核启动问题。



# 项目方案

- 每日批量randconfig内核构建
- 制作一个mini的rootfs(参考yocto)，并大规模qemu启动上述内核
- 在上述rootfs集成Trinity测试
- 批量找出qemu上可启动的kconfig最小配置集合
- 端到端完成和优化从linux上游内核更新到测试到bug报告
- 同时支持arm,x86,riscv
- 功能上线后，希望每周对内核社区报告1个以上kernel boot bug



# 项目方案

- 内核基线版本(Linux v5.10 2021-12-13)
- 内核目标版本(Linux v5.17 2022-3-20)
- 一共50000项左右commits
- Compass-CI平台
- qemu启动测试
- Fuzz测试



# 项目方案

- 实现每个randconfig:
  - 产生randconfig配置
  - 算法修复配置
  - 获取可启动配置
- 实现可启动配置:
  - 目标内核进行bisect定位
  - 找到关键commit
- 实现Fuzz测试、跟踪和报告



# 项目进展

```
boot failed
start make
make succeed
start qemu
boot succeed
start make
make succeed
start qemu
boot failed
start make
make succeed
start qemu
boot succeed
start make
make succeed
start qemu
boot failed
start make
make succeed
start qemu
boot failed
bootable randconfig-2022-04-09-03:01:45-fixed-2022-04-09-04:46:53
*****Congrats!*****
```

可启动randconfig

```
test commit 97e9c8eb4bb1dc57859acb1338dfddb967d7484 2022-03-19 11:04:10 -07:00
start make
make succeed
start qemu
boot succeed
test commit fe83f5eae432ccc8e90082d6ed506d5233547473 2022-03-20 14:55:46 +01:00
start make
make succeed
start qemu
boot succeed
test commit 7445b2dcd77ae8385bd08bb6c2db20ea0cfa6230 2022-03-20 09:46:52 -07:00
start make
make succeed
start qemu
boot failed
test commit 1e0e7a6a28f877312b93cd12a1448c8d53733b55 2022-03-20 09:27:52 -07:00
start make
make succeed
start qemu
boot failed
*****congrats!*****
Find ealiest non-bootalbe commit:
1e0e7a6a28f877312b93cd12a1448c8d53733b55 at 2022-03-20 09:27:52
```

关键commit



# 项目进展

- Trinity采用linux system call fuzz测试
- 根据bootable randconfig在Compass-CI上提交job
- 返回测试结果

```
[child1:43174] [262] clock_adjtime(which_clock=/, utx=0x+++c262c0000) = -1 (Operation not supported)
[child1:43174] [263] msgget(key=-37, msgflg=0x5a61f0a9fb) = -1 (No such file or directory)
[child1:43174] [264] mmap(addr=0xffffc262a0000, old_len=0, new_len=0x10000, flags=0x0, new_addr=0) = -1 (Cannot allocate memory)
[child1:43174] [265] unshare(unshare_flags=0x20010b00) = -1 (Operation not permitted)
[child1:43174] [266] mlock(addr=0xffffc266c0000, len=0x140000) = -1 (Cannot allocate memory)
[child1:43174] [267] inotify_init1(flags=0x800) = 516
[child1:43174] inotify fd:516 flags:0 global:0
[child1:43174] [268] renameat2(oldfd=28, oldname="/proc/5266/task/5266/comm", newfd=28, newname="/proc/288/task/288/net/snmp", flags=0x3) = -1 (Invalid argument)
[child1:43174] [269] unlinkat(dfd=28, pathname="/sys/devices/virtual/bdi/253:0/power/runtime_statuses", flag=0x20000000) = -1 (Invalid argument)
[child1:43174] [270] quotactl(cmd=0xffffe, special=0xffffc262c0000, id=0xb021232042b, addr=0xffffc264c0000) = -1 (No such file or directory)
[child1:43174] [271] inotify_add_watch(fd=28, pathname="/proc/35/net/netfilter", mask=0x84008493) = -1 (Invalid argument)
[child1:43174] [272] ioprio_set(which=0x6c6c, who=0xffffffffff, ioprio=0) = -1 (Invalid argument)
[child1:43174] [273] futex(uaddr=0xffffc26ac0000, op=0x89, val=0xffffffffffffbf1a, utime=0xffffc26bd0000, uaddr2=0xffffc269c0000, val3=512) = -1 (Invalid argument)
[child1:43174] [274] execve(name="/proc/5471/task/5471/net/rt_acct", argv=0x30516620, envp=0x305163e0) = -1 (No such file or directory)
[child1:43174] [275] set_mempolicy(mode=0xffffe, nmask=0xffffc266c0000, maxnode=1554) = -1 (Invalid argument)
channel_input_data: channel 0: append data: memory allocation failed
```

```
[child16:14681] [529] vmsplce(fd=315, iov=0x559f8d2c5d40, nr_segs=819, flags=0x2) = -1 (Bad address)
[child16:14681] [530] [32BIT] mprotect(start=0x7f089e996000, len=0x72000, prot=0x3000004) = -1 (Invalid argument)
[child16:14681] [531] statx(dfd=389, filename="/proc/2279/mem", flags=0x2000, mask=0xf7f6257fa1ae, buffer=0x0) = -1 (Bad address)
[child16:14681] [532] quotactl(cmd=0xffffd, special=0x7f089e996000, id=0xffffffffc0000000, addr=0x7f08a0873000) = -1 (No such file or directory)
[child16:14681] [533] mq_unlink(u_name=0x7f08a0874000) = -1 (No such file or directory)
[child16:14681] [534] [32BIT] utimensat(dfd=389, filename="/proc/266/limits", utimes=0x0, flags=0x100) = -1 (Invalid argument)
[child16:14681] [535] inotify_init() = 485
[child16:14681] inotify fd:485 flags:fffffffe global:0
[child16:14681] [536] timerfd_create(clockid=0x0, flags=0x80800) = 486
[child16:14681] [537] mlock(addr=0x7f089e296000, len=0x189000) = -1 (Cannot allocate memory)
[child16:14681] [538] waitid(which=33, upid=1, infop=0x7f089e496000, options=0x40000, ru=0x7f089e496000) = -1 (Invalid argument)
[child16:14681] [539] getcpu(cpu=0x0, nodep=0x4, unused=0xc) = -1 (Bad address)
[child16:14681] [540] swapoff(path="/proc/367/task") = -1 (Operation not permitted)
[child16:14681] [541] [32BIT] chroot(filename="/proc/1978/task/2007/fdinfo/20") = -1 (Bad address)
[child16:14681] [542] [32BIT] getrlimit(resource=0x1, rlim=0x7f089e896000) = -1 (Bad address)
[child16:14681] [543] userfaultfd(flags=0x80000) = 487
[child16:14681] [544] move_mount(from_dfd=389, from_pathname="/proc/318/net/raw", to_dfd=335, to_pathname="/proc/1574/task/1579/net/sockstat6", flags=0x2) = -1 (Invalid argument)
[child16:14681] [545] setattr(pathname="/sys/devices/LNXSYSTM:00/LNXSYBUS:00/PNP0A05:2e/PNP0A05:31/PNP0C80:154/uevent", name=0x1, value=0x7f089e996000, size=0, flags=0x3) = -1 (Bad address)
[child16:14681] [546] writev(fd=389, vec=0x559f8d31ce30, vlen=195) [main] kernel became tainted! (512/0) Last seed was 600846529
trinity: Detected kernel tainting. Last seed was 600846529
[main] exit_reason=7, but 22 children still running.
[main] Bailing main loop because kernel became tainted..
[main] Ran 260585 syscalls. Successes: 60605 Failures: 198881
```



谢谢！！





# linux kernel 性能测试

Sig成员



# 为什么需要自动化测试

1. 敏捷开发与CI/CD
2. Linux内核测试现状
3. Linux内核的自动化测试



# 敏捷开发与CI/CD

**CD** Continuous delivery 持续性交付

完成 CI 中构建及单元测试和集成测试的自动化流程后，持续交付可自动将已验证的代码发布到存储库（repository）。

**CD** Continuous delivery  
持续性交付

Automated acceptance tests / user acceptance tests

持续交付指的是，频繁地将软件的新版本，交付给质量团队或者用户，以供评审。如果评审通过，代码就进入生产阶段。



# Linux内核测试现状

1. 测试人员难以满足快速活跃的开发
2. 快速的版本发布会引进大量的代码更新
3. 需要覆盖多种平台
4. ....

# 适配OS(以openEuler;Centos;Debian为例)

1. 解决aarch64架构下测试套与os的依赖问题，指定或者打出相应的依赖包
  - 对于缺失依赖包的情况，利用cci-depends和cci-makepkg查找并构建相关依赖包文件
  - 登录机器修复jobs(顺次执行任务提交文件断点执行定位问题)
  - 根据测试日志修改脚本。
1. 根据OS,OS\_version,jobs等进行任务分组（提高总体效率）

jobs地址:[https://gitee.com/wu\\_fengguang/lkp-tests/tree/master/jobs](https://gitee.com/wu_fengguang/lkp-tests/tree/master/jobs)

主要技术文档:[https://gitee.com/wu\\_fengguang/compass-ci/blob/master/doc/README.zh.md](https://gitee.com/wu_fengguang/compass-ci/blob/master/doc/README.zh.md)

执行机位置:[https://gitee.com/wu\\_fengguang/lab-z9/tree/master/hosts](https://gitee.com/wu_fengguang/lab-z9/tree/master/hosts)



# 适配OS (openEuler; centos; Debian)



1. 对于缺失依赖包的情况，利用cci-depends和cci-makepkg查找并构建相关依赖包文件
2. 登录机器修复jobs(顺次执行任务提交文件断点执行定位问题)
3. 根据测试日志修改脚本
4. 对jobs进行分组（提高使能总体效率）

利用compass-ci中nr\_run=n(一般可以为3)寻找标准差比较大的jobs并持续优化对于性能测试类型的任务，提供一个工具能够寻找到stats的方差比较大的任务，对这个工具找出来的大方差任务进行优化，使方差足够小

开发能够找出经常跑失败的功能测试任务的工具，并对这个工具找出的jobs进行优化，降低任务的失败率。







# Compass CI 易用性提升

CICD sig

鲁伟涛



1. Compas CI Web 提交job
2. Compass CI 用户个人数据看板

# Compass CI 当前提交job方式

- 命令行 提交job命令:

```
lkp-tests/sbin/submit job.yaml
```

- 在哪里运行命令?
  - Option1: 邮件申请Compass CI 账号, account-vm中运行;
  - Option2: 邮件申请Compass CI 账号, 个人linux机器运行, 必要的环境准备:
    - 下载lkp-tests代码仓
    - 安装Ruby-dev及submit运行依赖库
    - 配置~/.config/compass-ci/\*



# Compass CI Web页面提交job的好处



- 账号申请简单， gitee账号登录Compass CI网站， 绑定Compass CI账号即可
- 不局限于account-vm或个人linux机器， 登录Compass CI网站就可以提交job
- 随时随地提交， 甚至手机上都可以提交job
- 申请账号 → 提交job， 甚至不需要文档

# Compass CI Web页面提交job

- Web 页面如何提交job
  - 账号登录Compass CI网站, 在线编辑job.yaml, 并提交job
  - 测试套的说明

unixbench.yaml ▼

编辑

```
suite: unixbench
category: benchmark
runtime: 300s
nr_task:
- 1
- 100%

unixbench:
test:
- whetstone-double
- shell1
- shell8
```

提交

取消



# Compass CI Web页面提交job 依赖功能



- 支持gitee等第三方账号登录Compass CI网站
- 支持gitee等第三方账号绑定Compass CI账号
- Compass CI 调度器对应接口submit\_job支持用户动态token鉴权

以上功能已发布学生任务：

<https://gitee.com/openeuler/compass-ci/issues/I4VM09?from=project-issue>

1. Compas CI Web 提交job

2. Compass CI 用户个人数据看板



# 当前Compass CI 查看测试结果



Compass CI 查看测试结果：  
<https://compass-ci.openeuler.org/jobs>

开源软件的测试结果如何，这里告诉你答案

reports per page 10

序号	upstream_repo	os	os_version	os_arch	start_time	suite	category	testbox	job_state	id	error_ids
1		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432076	
2		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432079	
3		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g.taishan...	incomplete	z9.14432038	
4		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432072	
5		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432030	
6		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432071	
7		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432066	
8		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432065	
9		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432027	
10		openeuler	20.03-LTS-SP3	aarch64	2022-04-13T1...	rpmbuild	functional	dc-16g	incomplete	z9.14432064	

# 当前Compass CI 查看测试汇总



Compass CI 查看测试汇总：  
<https://compass-ci.openeuler.org/job-summary?>

CI

Compass CI

Job Summary

Filter

suite=openeuler\_docker group\_id=test\_fail

Group By

os

确定

Example

os	nr_all	nr_pass	nr_fail
openeuler	1944	0	<a href="#">1944</a>
archlinux	0	0	<a href="#">0</a>
centos	0	0	<a href="#">0</a>
debian	0	0	<a href="#">0</a>
unikylin	0	0	<a href="#">0</a>
delimiter	0	0	<a href="#">0</a>

# 当前Compass CI 查看性能测试图表



Compass CI 查看性能测试图表：  
<https://compass-ci.openeuler.org/job-summary?>

## Performance Result

Filter

suite:netperf,pp.netperf.test:TCP\_STREAM;tbx\_group:vm-2p8g;

series

os:centos;os:openeuler;

metrics

netperf.workload

x\_params

send\_size

确定

Example

netperf-workload	1	64	128	256	512	1024	1500	2048	4096	9000	16384	32768	65536
centos	214125000	223593750	317748046.875	0	224260253.9062	0	236305250	179182250.9766	162605895.9961	136855833.333	0	39634494.7815	11828041.0767
openeuler	165187500	180896484.375	298291992.1875	357159667.9688	325276611.3281	275883911.1328	215130750	211813293.457	147635742.1875	65134962.7597	79340950.0122	42881687.1643	0
openeuler vs centos	-22.9	-19.1	-6.1	0	45	0	-9	18.2	-9.2	-52.4	0	8.2	0

netperf-workload

centos

openeuler



# 用户个人测试数据看板的价值

- Compass CI当前数据可视化的弊端：
  - 用户不能直观的看到自己的测试数据，需要去job-list中筛选
  - 查看测试汇总、性能图表，需要用户手动输入复杂的查询条件
  - 页面分布零散，新用户甚至不会关注测试汇总、性能图表
- 用户个人性能看板的好处
  - 用户能直接看到自己的测试数据
  - 测试汇总、性能图表数据自动展示在看板，用户容易上手使用

# 用户个人性能看板

用户能够比较直接的看到自己的测试数据  
测试汇总、性能图表数据自动展示在看板

Dashboard

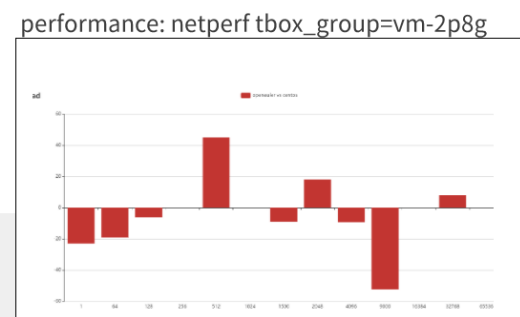
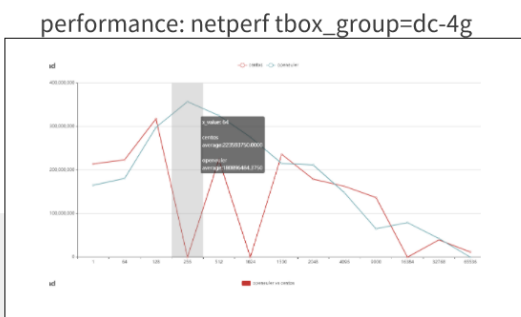
job list

ge 10 search jobs for rpm-build package

id	name	status	start time	end time	category	testcase	job name	id
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442111
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442112
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442113
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442114
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442115
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442116
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442117
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442118
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442119
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10	rpm-build	rpm-build	rpm-build	215442120

job summary: rpmbuild

id	name	status	start time	end time
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10
rpm-build	rpm-build	success	2023-04-10 10:10:10	2023-04-10 10:10:10



# Compass CI 用户个人数据看板 依赖功能



- 支持gitee等第三方账号登录Compass CI网站
- 支持gitee等第三方账号绑定Compass CI账号
- Compass CI web-bakend支持用户动态token鉴权

以上功能已发布学生任务：

<https://gitee.com/openeuler/compass-ci/issues/I4VM09?from=project-issue>







# PR触发构建测试矩阵

CICD sig

汪林

# 特性

- 监测软件的PR提交，并基于PR代码进行包构建；
- 支持软件包直接反向依赖构建；
- 支持多OS、多架构组合测试；
- 支持构建测试、兼容性测试；
- 支持可重复构建；

# 价值

- ☆ 基于PR代码进行构建，测试时间节点提前，代码合入前屏蔽问题
- ☆ 直接反向依赖包构建，避免引入兼容性问题
- ☆ 可重复构建，避免潜在安全风险，二进制质量保障
- ☆ 多OS、多架构矩阵测试，可视化报表对比测试结果，提供jobs测试日志




# 场景

## ◆ OpenEuler官方仓

为src-openeuler官方仓提供门禁测试，当软件包有PR提交，触发门禁检查，自动进行包构建测试并反馈测试结果。

## ◆ 个人仓

开发者可以通过仓库注册，自定义测试组合，对软件进行构建测试、兼容性测试等，提供测试报告和可视化的测试报表。

 **openeuler-ci-bot** 拥有者 3 小时前

Check Name		Build Result	Build Details
check_binary_file		✓ SUCCESS	#9
check_package_license		✓ SUCCESS	
check_package_yaml_file		✓ SUCCESS	
check_spec_file		✓ SUCCESS	
x86_64	check_build	✓ SUCCESS	#9
	check_install	✓ SUCCESS	
aarch64	check_build	✓ SUCCESS	#9
	check_install	✓ SUCCESS	

😊 表态    💬 回复

# 准备工作

## 1、webhook配置

首先，要为监测仓库配置webhook，以gitee webhook配置为例：  
用户可以通过 gitee 「仓库主页」 -> 「管理页面」 -> 「WebHooks」 添加 WebHook。

### 添加 WebHook

URL: WebHook 被触发后，发送 HTTP / HTTPS 的目标通知地址。

WebHook 密码/签名密钥: 用于 WebHook 鉴权的方式，可通过 **WebHook 密码** 进行鉴权，或通过 **签名密钥** 生成请求签名进行鉴权，防止 URL 被恶意请求。签名文档可查阅 [《WebHook 推送数据格式说明》](#)。

更多文档可查阅 [《Gitee WebHook 文档》](#)。

URL:

WebHook 密码/签名密钥:

WebHook 密码

选择事件:

<input type="checkbox"/>	Push	仓库推送代码、推送、删除分支
<input type="checkbox"/>	Tag Push	新建、删除 tag
<input type="checkbox"/>	Issue	新建任务、删除任务、变更任务状态、更改任务指派
<input checked="" type="checkbox"/>	Pull Request	新建、更新、合并、关闭 Pull Request, 新建、更新、删除 Pull Request 下标签, 关联、取消关联 Issue
<input type="checkbox"/>	评论	评论仓库、任务、Pull Request、Commit

☒ 激活 (激活后事件触发时将发送请求)

添加

参数说明:

**URL(\*):** 接收 WebHook 数据的 http 地址。

Gitee 将发送 Post 请求到这个地址

**密码:** 为了保证安全以及识别数据来源，建议设置一个密码。Gitee 将会在 Post 数据中携带这个密码。注意，密码是明文

**钩子:** 用户在 Gitee 上可触发的 WebHook，支持添加多个钩子

## 2、仓库注册

1) fork仓库 ([https://gitee.com/wu\\_fengguang/upstream-repos](https://gitee.com/wu_fengguang/upstream-repos)) , 以便通过pull request的方式提交文件

### 2) 创建目录及注册文件

第一层目录的名称以测试仓库名称首个字符命名, 第二层目录的名称以测试仓库名称命名, 第二层目录下的文件以上游仓库名称命名

以mongodb/mongo项目为例:

```
mkdir -p m/mongo
```

```
echo 'url: https://github.com/mongodb/mongo.git' > m/mongo/mongo
```

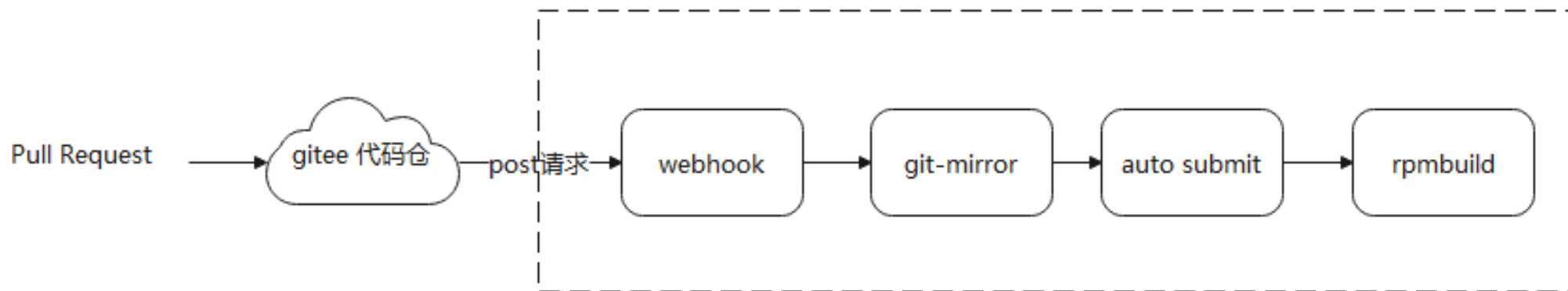
### 3) DEFAULTS文件指定测试组合

在第二层目录创建DEFAULTS文件来指定测试组合, 使用os、os\_version、os\_arch、docker\_image等字段来指定测试的os、os版本、架构、容器等, 内容如下:

submit:

- command:testbox=dc-16g os=openeuler os\_version=20.09 os\_arch=aarch64 docker\_image=openeuler:20.09-pre rpmbuild-without-arch.yaml
- command:testbox=dc-16g os=openeuler os\_version=21.09 os\_arch=aarch64 docker\_image=openeuler:21.09-pre rpmbuild-without-arch.yaml

# 流程



1. 当仓库有PR提交后，gitee根据webhook配置回调Compass CI API接口
2. git-mirror服务接收webhook请求，获取PR最新代码
3. 根据仓库注册时提供的DEFAULTS文件，提交多OS、多架构下的包构建任务
4. 构建成功后，提交兼容性测试任务
5. 提供测试矩阵报表，可视化对比测试结果



# 测试矩阵



Compass CI

## Test Matrix

多架构多系统测试结果对比



包名	os	架构	构建测试	兼容性测试	功能测试
zstd	openeuler 20.03	aarch64	✓	✓	✓
zstd	openeuler 20.03-LTS-SP1	aarch64	✓	✓	✓
zstd	openeuler 20.03-LTS-SP2	aarch64	✓	✓	✓
zstd	kylin 10	aarch64	✓	✓	✓

Join Compass CI  
让社区开发更简单

[隐私政策](#) | [法律声明](#) | [免责声明](#)  
版权所有 © 2020 openEuler 保留一切权利

