

# **Ruyi sDK v0.2 测试策略和 自动化测试工具建设**

# 目 录

01

Ruyi SDK 简介

02

Ruyi SDK v0.2 测试策略

03

Ruyi SDK 自动化测试工具建设

04

Ruyi SDK v0.2 测试结果

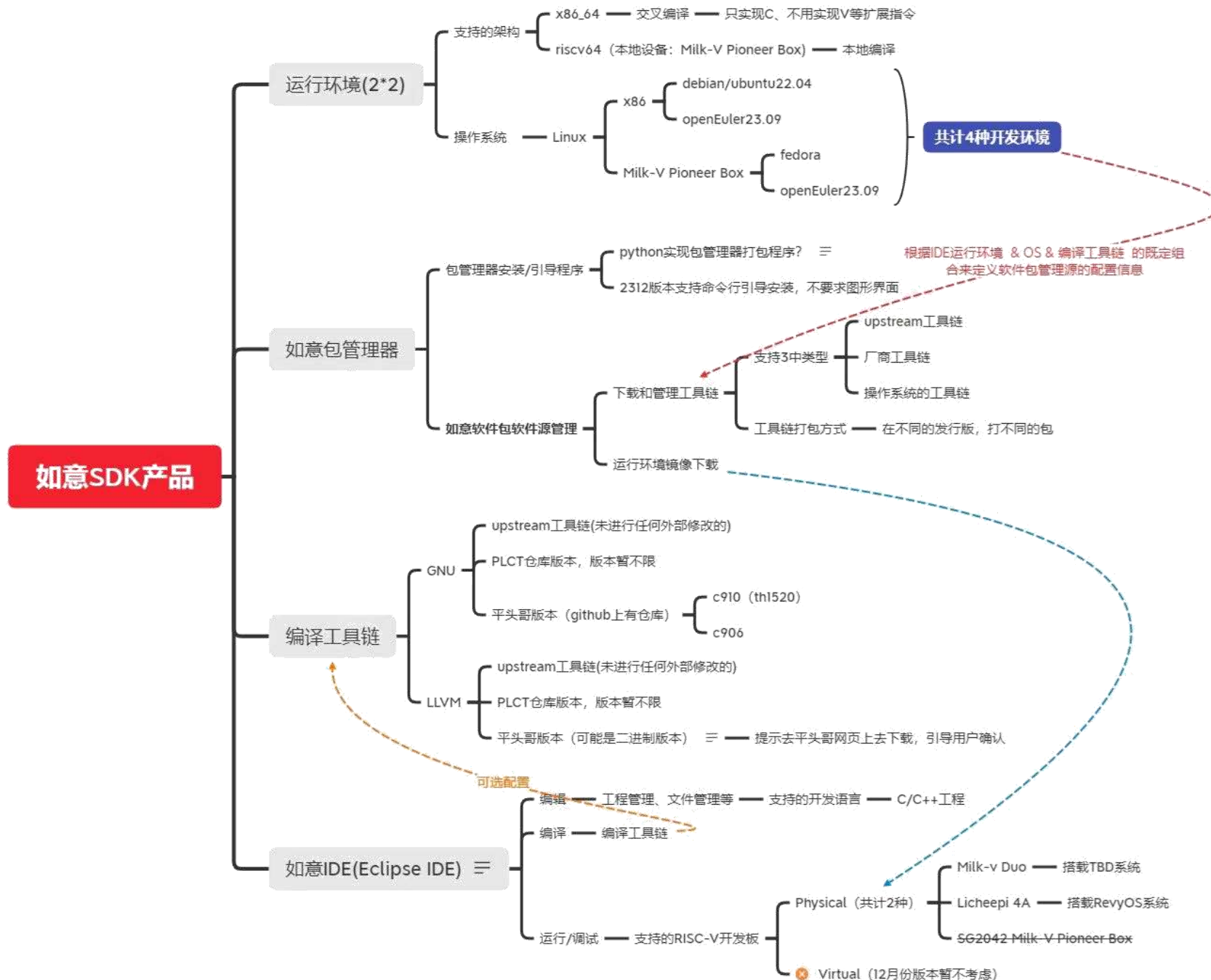
- 如意SDK旨在为 RISC-V 开发者提供一个一体化集成开发环境。
- 为普通 RISC-V 用户提供相同的开发流程，让用户可以在不同的开源工具和制造商定制的工具链之间轻松切换。
- 对于用户购买到的任何一款如意 SDK支持的 RISC-V 开发板或模组，都可通过如意 SDK 系统获得硬件资料说明、固件和软件更新、系统性开发环境支持、软硬件调试支持。

- 开发者可以轻松获取任何常用的 RISC-V 扩展指令集架构组合的系统软件支持，通过如意 SDK 系统便捷地生成客户所需的操作系统、工具链、语言执行环境(运行时或虚拟机)、计算库和应用框架。
- 如意SDK支持和持续维护 RISC-V Vector 0.7.1和 RVP 0.5.2 等已经大规模硅化的草案标准和一些厂商定制扩展。

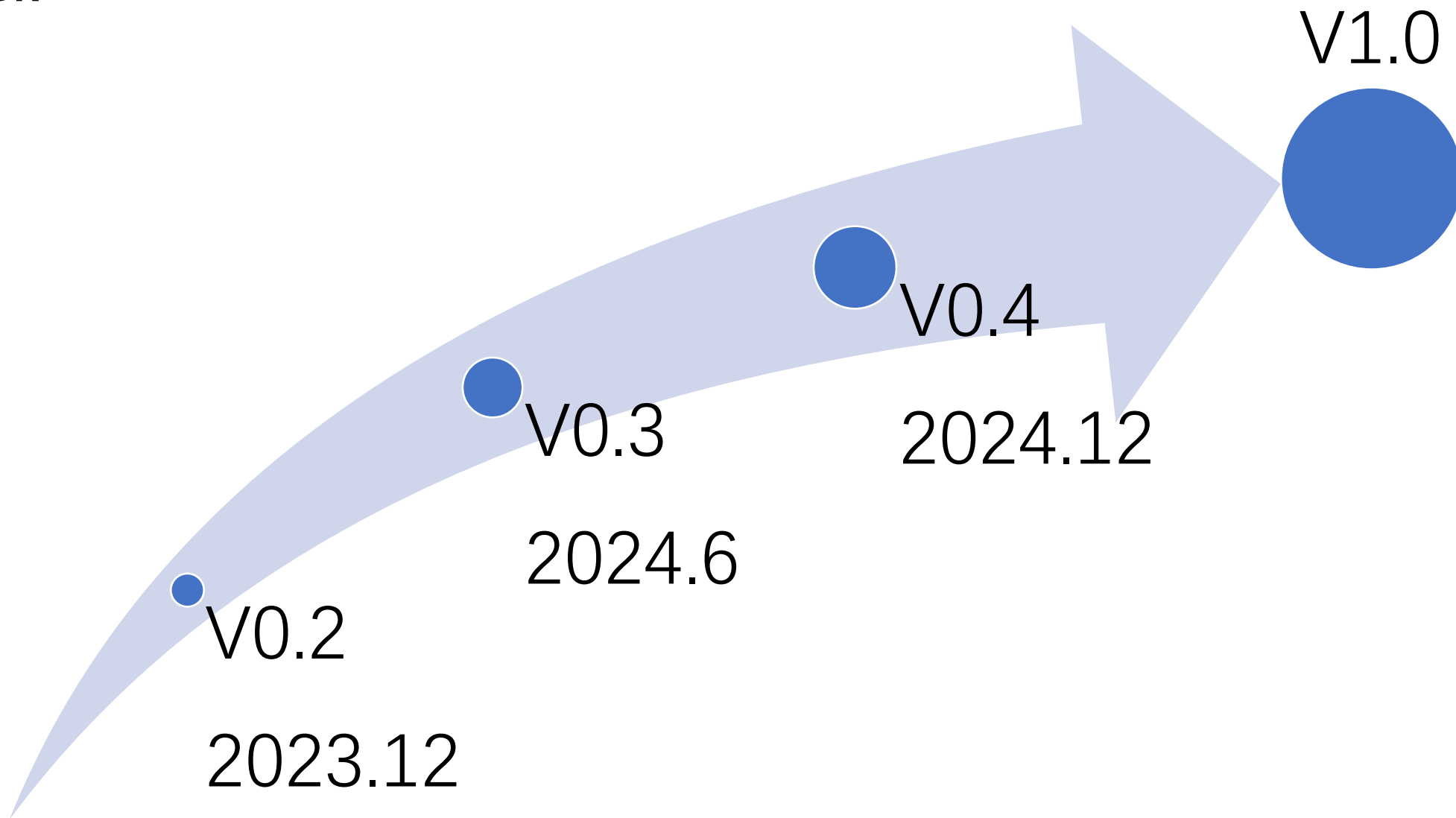
- 如意 SDK V1.0主要聚焦 RuyiSDK 基础框架的实现与主体环节的打通，发布内容主要包含6个方面
  1. Toolchain: GNU 工具链对 RISC-V 扩展指令集的多版本支持和系统适配
  2. Sysroot: openEuler, RevyOS 在内的2种操作系统的交叉环境和本地环境支持
  3. 模拟器: 整合了 Xuantie CPUs 的扩展实现，并模拟了 LicheeRV 和 LicheePi4A 的硬件环境

4. 硬件镜像：提供 Lichee RV, Lichee Pi 4A , Milk-V Pioneer 等多平台配套镜像
5. 辅助工具：提供了自动化构建与便捷交互使用工具
6. RuyiSDK 文档：介绍了工具链，模拟器，镜像和辅助工具的构建与使用方法

# Ruyi SDK



详细参考: <https://github.com/ruyisdk/> 和 <https://github.com/ruyisdk/ruyi>



RuyiSDK 发展路线图



# 目 录

01

Ruyi SDK 简介

02

Ruyi SDK v0.2 测试策略

03

Ruyi SDK 自动化测试工具建设

04

Ruyi SDK v0.2 测试结果

# RuyiSDK v0.2 测试计划

- 测试计划
  1. RuyiSDK v0.2版本测试按照RuyiSDK团队制定的版本发布计划规划相应的测试活动。
  2. RuyiSDK v0.2版本测试以用户的视角，基于RuyiSDK v0.2版本使用文档，测试检验每个功能是否能正常使用。

测试阶段	起始时间	结束时间	Days	备注
单元测试	2023.11.1	RuyiSDK v0.2开发完成 2023.12.04		单元测试启动
Round1	2023.12.04	2023.12.11	7	版本启动测试
缺陷修复	Round1测试结果同步提交 2023.12.05	2023.12.12	7	缺陷修复
Round2	2023.12.08 (1207 1211)	2023.12.15	7	发布测试

## RuyiSDK v0.2 测试重点

单元测试（RuyiSDK v0.2开发完成前）

1. 单一功能的测试（用于验证Ruyi工具的功能点）

- Mugen
- openQA

2. 问题单回归

## RuyiSDK v0.2 测试重点

Round1 (RuyiSDK v0.2开发完成后, ruyi命令行工具)

1. 问题单全量回归
2. 功能测试 (命令行—Mugen、IDE—openQA)
3. 硬件测试 (openQA)
4. 编译器测试 (用于验证安装成功的编译环境工具可运行, bin目录下的工具可运行, 不缺依赖)
5. 文档测试 (对安装文档和部分操作文档进行验证)

### Round2

1. 问题单全量回归
2. 功能测试（命令行—Mugen、IDE—openQA）
3. 硬件测试（openQA）
4. 编译器测试（用于验证安装成功的编译环境工具可运行，  
bin目录下的工具可运行，不缺依赖）
5. 文档测试（对安装文档和部分操作文档进行验证）

### 1. 入口标准

- 上个阶段无 block 问题遗留。
- 转测版本的冒烟无阻塞性问题。
- 满足各阶段版本转测检查项。

### 2. 出口标准

- 策略规划的测试活动涉及的测试用例已执行完毕。
- 发布特性满足版本规划目标。
- 版本无阻塞问题遗留，严重问题有相应规避措施或说明。

# 目 录

01

Ruyi SDK 简介

02

Ruyi SDK v0.2 测试策略

03

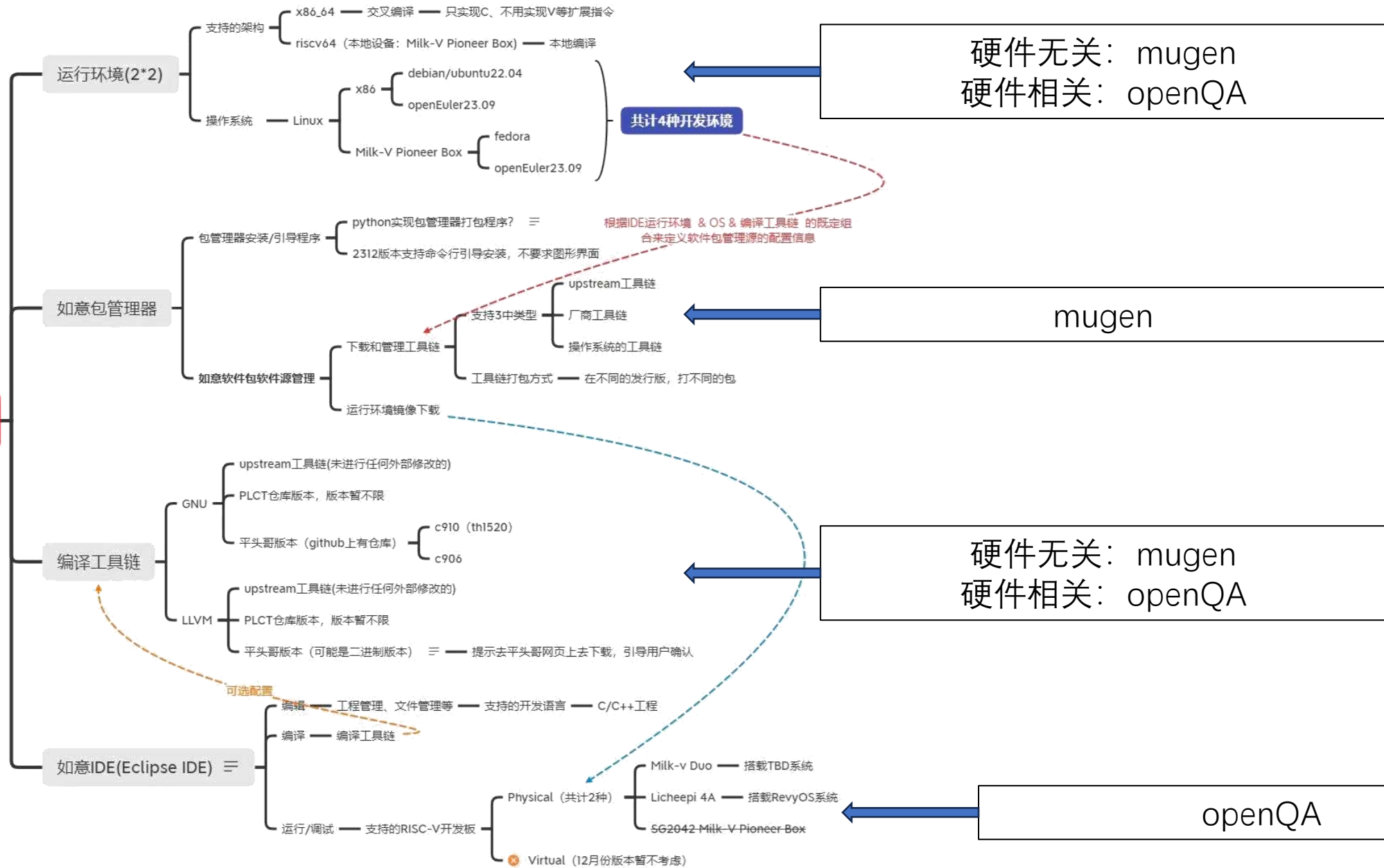
Ruyi SDK 自动化测试工具建设

04

Ruyi SDK v0.2 测试结果

# Ruyi SDK v0.2测试方法

如意SDK产品



硬件无关: mugen  
硬件相关: openQA

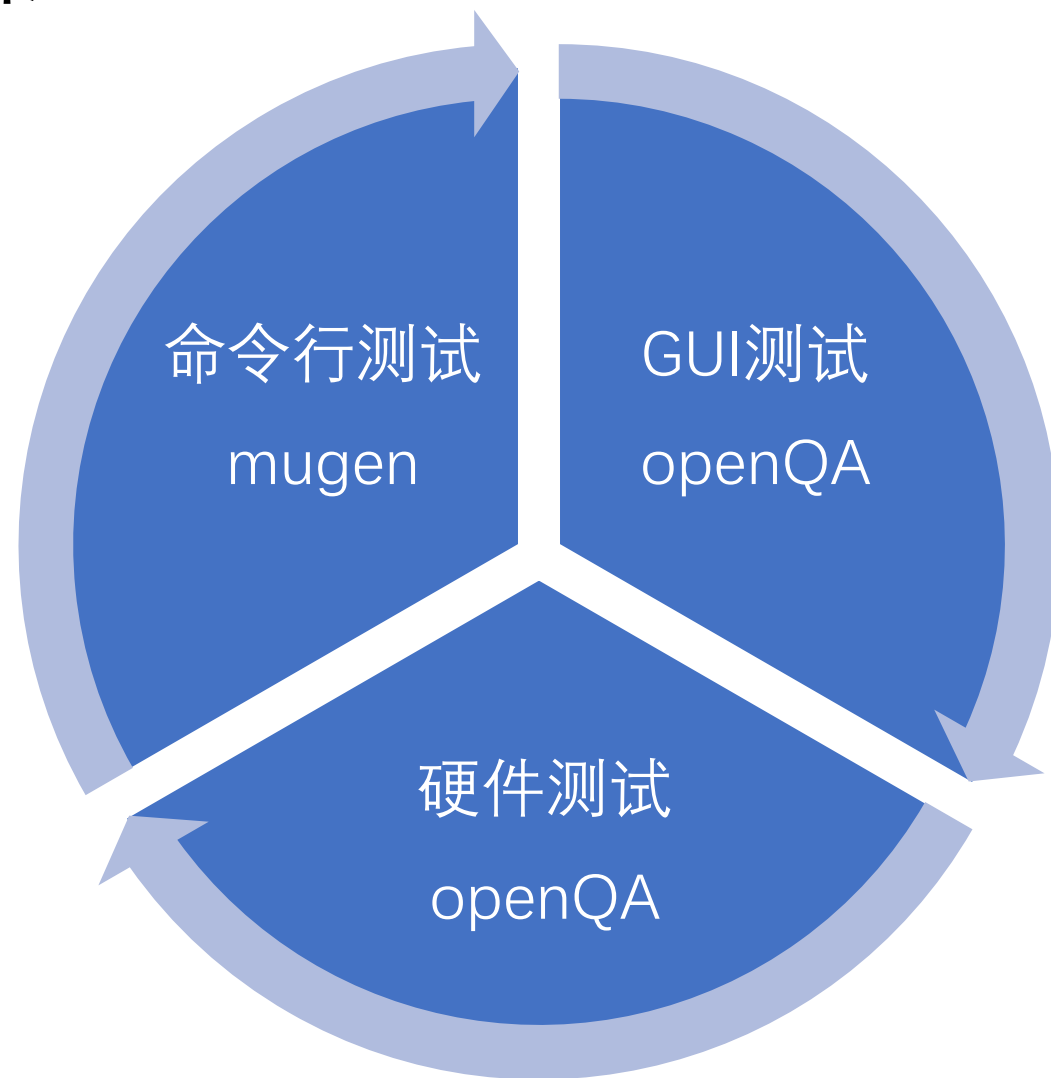
mugen

硬件无关: mugen  
硬件相关: openQA

openQA



## Ruyi SDK 自动化测试框架



## Ruyi SDK 自动化测试框架



PLCT云测试中心

命令行

GUI

硬件  
(启动和内核)

mugen 是 openEuler 社区开放的测试框架，提供公共配置和方法以便社区开发者进行测试代码的编写和执行。

mugen 适合纯命令程序的部署测试（配置测试）。

## 为什么使用Mugen

- Ruyi 是一个基于命令行的包管理程序
- mugen 对测试环境要求低
- mugen 与发行版相关的部分只有  
DNF\_INSTALL/DNF\_REMOVE

## Mugen如何进行测试

一个 mugen 测试用例通常有三个部分

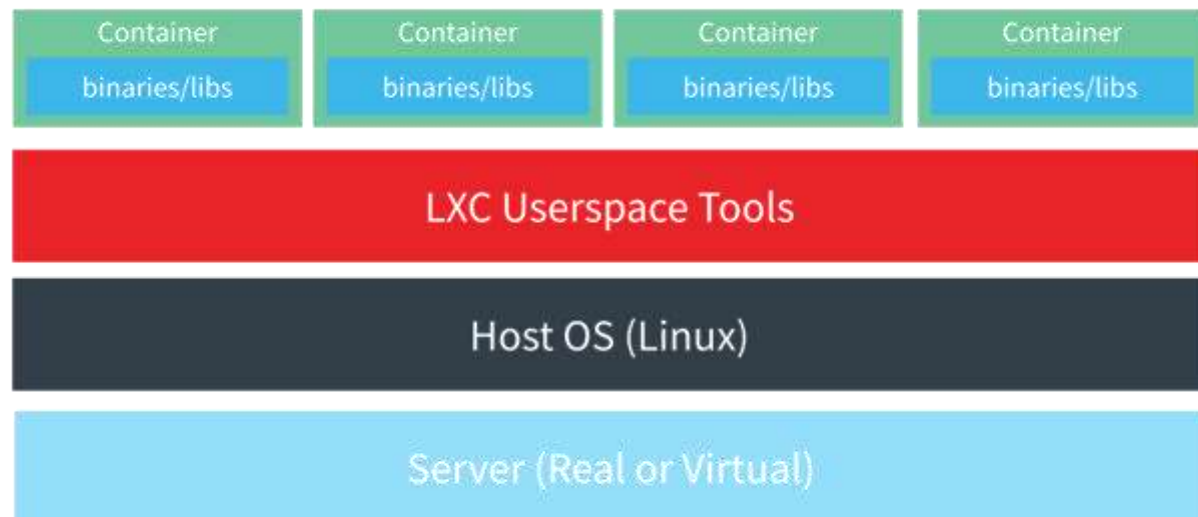
1. pre\_test
2. run\_test
3. post\_test

其中 run\_test 同时放整个测试测试过程，并在整个过程中插入检查点，判断某个变量值/命令返回值是否符合预期，并可以选择是否提前中断测试

# RuyiSDK v0.2 Mugen测试

## QEMU

QEMU openEuler 2309 x86\_64  
QEMU Ubuntu 22.04 LTS x86\_64  
QEMU Fedora 38 x86\_64  
QEMU openEuler 2309 riscv64



## 硬件开发板

LicheePi 4A openEuler 2309 riscv64  
LicheePi 4A RevyOS 20231210 riscv64  
SG2042 Milkv pioneer box v1.1 openEuler 2309 riscv64  
SG2042 Milkv pioneer box v1.1 Fedora 39 riscv64



Host OS (Linux)

## openQA简介

openQA是一个测试框架，它可以用来测试 **GUI** 应用程序，同时可以测试**引导加载程序和内核**，适用于基于RISC-V开发板的Linux发新版操作系统测试。

openQA 运行虚拟机, 监视其状态, 运行测试。

测试框架可以分为两部分。该存储库中托管的包含 Web 前端和管理逻辑（测试调度、管理、高级 API，...）

- (1) X86\_64 & ubuntu22.04 (GUI 交叉编译)
- (2) X86\_64 & openEuler23.09 (GUI 交叉编译)
- (3) Milk-V Pioneer Box & Fedora
- (4) Milk-V Pioneer Box & oerv23.09

测试基于 openQA 框架运行

- 针对 x86 (GUI 交叉编译) 两个测试, 使用基于QEMU的方式测试
- 针对 Milk-V 两个测试, 使用真机的方式测试



## 准备测试环境

- 下载系统镜像，并进行必要的初始化（例如设置密码等）
- 对于 ubuntu 采用 cloud img 最小镜像进行测试

 ubuntu-22.04-server-cloudimg-amd64.img 2023-10-26 09:11 641M Ubuntu Server 22.04 LTS (Jammy Jellyfish) released builds

- 对于 openeuler 采用 cloud 和 标准两个镜像测试

qcow2.xz ⓘ

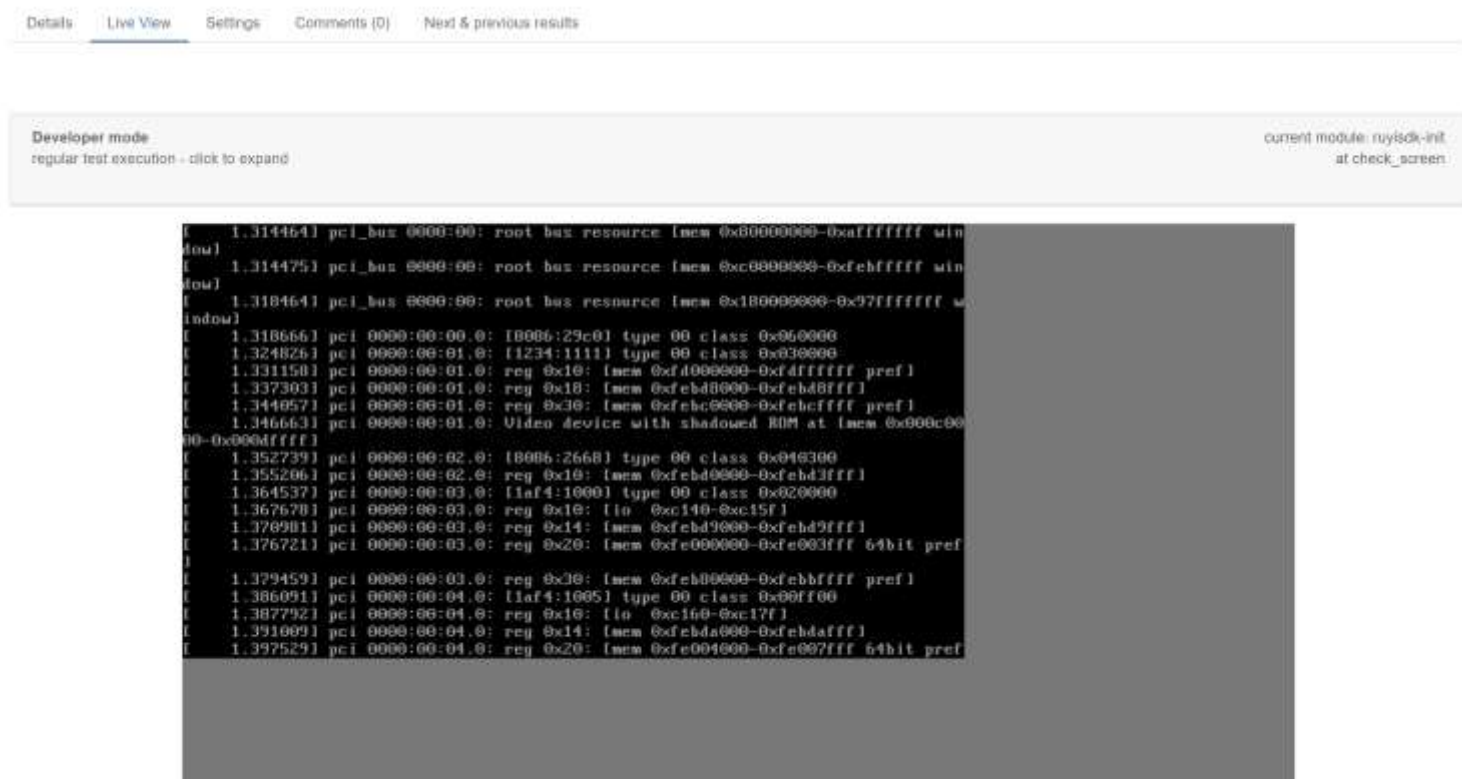
467.4 MiB

Offline Standard ISO ⓘ

4.0 GiB

# 运行测试

- openQA 会根据配置文件找到先前下载的镜像，通过 QEMU 启动虚拟机
- 测试运行过程中可以通过 vnc 查看系统情况



The screenshot displays the 'Live View' tab of an openQA test result. At the top, there are navigation links: 'Details', 'Live View' (active), 'Settings', 'Comments (0)', and 'Next & previous results'. Below the navigation bar, a status bar indicates 'Developer mode' and 'regular test execution - click to expand'. On the right side of the status bar, it says 'current module: ruyisdg-init at check\_screen'. The main content area shows a log of system boot messages, primarily related to PCI bus resources and device initialization. The log entries include timestamps and detailed information about PCI buses, resources, and devices, such as 'pci 0000:00:00:00: root bus resource', 'pci 0000:00:01:0: reg 0x10: [mem 0xfbd00000-0xfbd00000] pref 1', and 'pci 0000:00:01:0: Video device with shadowed ROM at [mem 0x00c00000-0x00c00000]'. The log continues with various PCI device initialization messages, including 'pci 0000:00:02:0: reg 0x10: [mem 0xfbd00000-0xfbd00000] pref 1', 'pci 0000:00:03:0: reg 0x10: [io 0xc140-0xc15f] pref 1', and 'pci 0000:00:04:0: reg 0x10: [io 0xc160-0xc17f] pref 1'.

```
[ 1.314464] pci_bus 0000:00: root bus resource [mem 0x00000000-0xffffffff] window
[ 1.314475] pci_bus 0000:00: root bus resource [mem 0xc0000000-0xfefeffff] window
[ 1.318464] pci_bus 0000:00: root bus resource [mem 0x180000000-0x97ffffff] window
[ 1.318666] pci 0000:00:00:0: [8086:29c0] type 00 class 0x060000
[ 1.324826] pci 0000:00:01:0: [1234:1111] type 00 class 0x030000
[ 1.331158] pci 0000:00:01:0: reg 0x10: [mem 0xfbd00000-0xfbd00000] pref 1
[ 1.337303] pci 0000:00:01:0: reg 0x18: [mem 0xfbd00000-0xfbd00000]
[ 1.344057] pci 0000:00:01:0: reg 0x30: [mem 0xfbd00000-0xfbd00000] pref 1
[ 1.346663] pci 0000:00:01:0: Video device with shadowed ROM at [mem 0x00c00000-0x00c00000]
[ 1.352739] pci 0000:00:02:0: [8086:2668] type 00 class 0x040300
[ 1.355206] pci 0000:00:02:0: reg 0x10: [mem 0xfbd00000-0xfbd00000]
[ 1.364537] pci 0000:00:03:0: [1af4:1000] type 00 class 0x020000
[ 1.367670] pci 0000:00:03:0: reg 0x10: [io 0xc140-0xc15f]
[ 1.370981] pci 0000:00:03:0: reg 0x14: [mem 0xfbd00000-0xfbd00000]
[ 1.376721] pci 0000:00:03:0: reg 0x20: [mem 0xfbd00000-0xfbd00000] 64bit pref
[ 1.379459] pci 0000:00:03:0: reg 0x30: [mem 0xfbd00000-0xfbd00000] pref 1
[ 1.386091] pci 0000:00:04:0: [1af4:1005] type 00 class 0x00ff00
[ 1.387792] pci 0000:00:04:0: reg 0x10: [io 0xc160-0xc17f]
[ 1.391099] pci 0000:00:04:0: reg 0x14: [mem 0xfbd00000-0xfbd00000]
[ 1.397529] pci 0000:00:04:0: reg 0x20: [mem 0xfbd00000-0xfbd00000] 64bit pref
```

## 测试脚本编写及效果

- 用户可以编写代码利用 API 模拟命令输入

```
assert_script_run 'rui -h';  
assert_script_run 'rui update';  
assert_script_run 'rui list';  
assert_script_run 'rui install plct';
```

### 测试脚本

```
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=255 time=13.4 ms  
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=255 time=12.4 ms  
  
--- baidu.com ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2014ms  
rtt min/avg/max/mdev = 12.436/12.895/13.414/0.401 ms  
root@ubuntu:~# curl https://mirror.iscas.ac.cn/rui/sdk/rui/testing/rui.2023103  
0 --output rui; echo IVTtC-$?- > /dev/ttyS0  
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current  
           Dload  Upload   Total   Spent    Left   Speed  
100 9727k  100 9727k    0     0  7973k      0  0:00:01  0:00:01 --:--:-- 7979k  
root@ubuntu:~# chmod +x ./rui; echo tqGJs-$?- > /dev/ttyS0  
root@ubuntu:~# export PATH=$PATH:$(pwd); echo sgk2B-$?- > /dev/ttyS0  
root@ubuntu:~# rui -h | grep "usage: rui"; echo GazMu-$?- > /dev/ttyS0  
usage: rui [-h] {install,i,list,toolchain,update} ...  
root@ubuntu:~# rui update | grep ""; echo wkawd-$?- > /dev/ttyS0  
root@ubuntu:~# rui list | grep ""; echo VdxB5-$?- > /dev/ttyS0  
root@ubuntu:~# rui install plct | grep ""; echo PES_K-$?- > /dev/ttyS0  
fatal error: {'plct'} not found in the repository  
root@ubuntu:~# rui install name:plct | grep ""; echo ECjaK-$?- > /dev/ttyS0  
fatal error: {'name:plct'} not found in the repository  
root@ubuntu:~# rui install slug:plct-20231026 | grep ""; echo K90rf-$?- > /dev/  
ttyS0  
fatal error: {'slug:plct-20231026'} not found in the repository  
root@ubuntu:~# _
```

VNC 运行截图

## 测试结果

- 通过命令的返回值，或者命令输出判断是否成功
- 当命令返回值为非 0 是测试出错，可以在测试结果中观察到

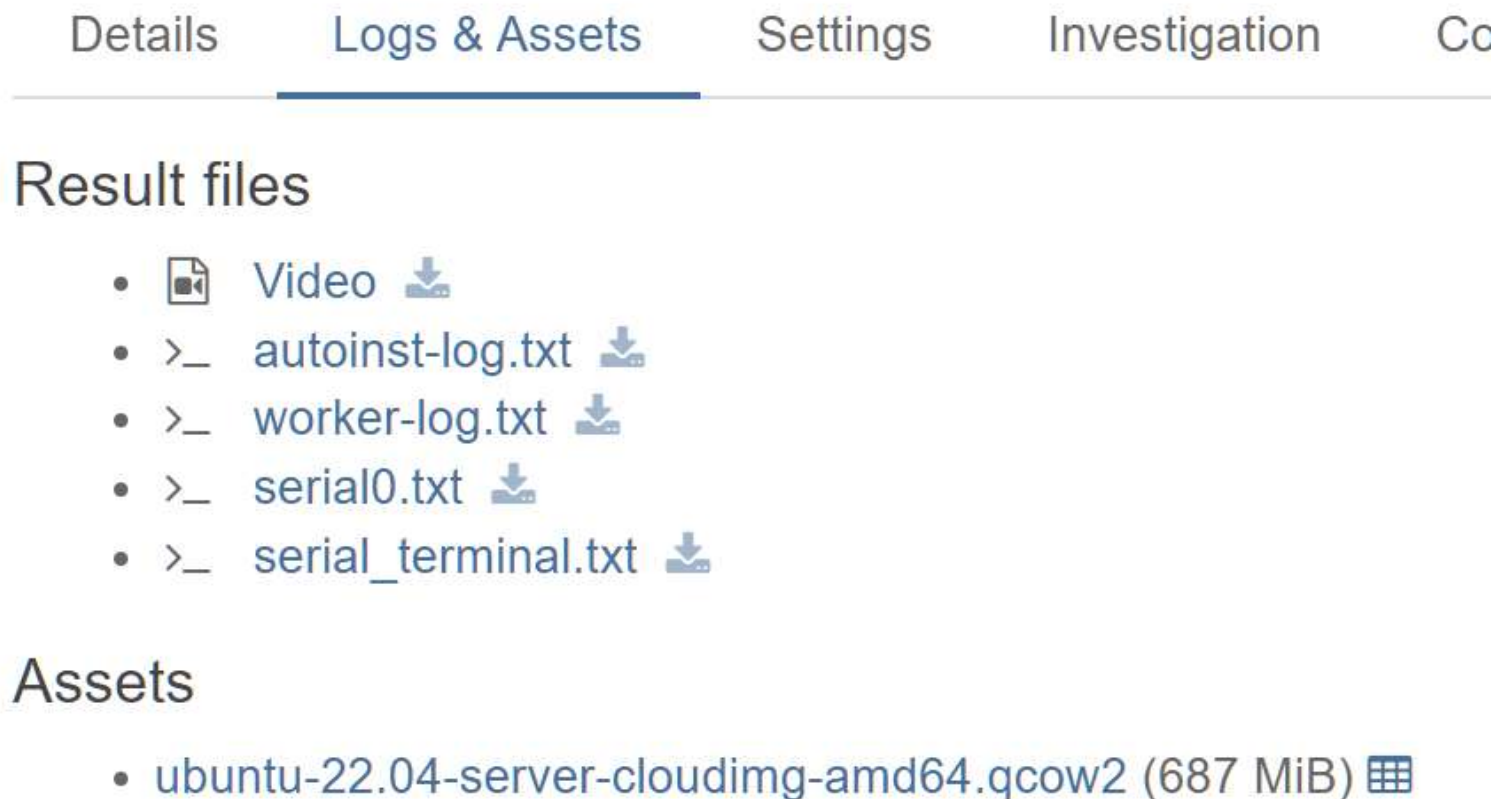


示例：curl 命令出错，测试用例不通过

## 测试结果

openQA 会在测试过后自动上传日志，包括：







- 虚拟机运行视频
- 串口输出
- 各类日志...




The screenshot displays the 'Logs & Assets' tab of the openQA web interface. At the top, there are navigation tabs: 'Details', 'Logs & Assets' (which is active and underlined), 'Settings', 'Investigation', and 'Cc'. Below the tabs, the section 'Result files' lists five items, each with a download icon: 'Video', 'autoinst-log.txt', 'worker-log.txt', 'serial0.txt', and 'serial\_terminal.txt'. Below this, the 'Assets' section lists one item: 'ubuntu-22.04-server-cloudimg-amd64.qcow2 (687 MiB)' with a file icon.

Details Logs & Assets Settings Investigation Cc

Result files

-  Video 
- >\_ autoinst-log.txt 
- >\_ worker-log.txt 
- >\_ serial0.txt 
- >\_ serial\_terminal.txt 

Assets

- ubuntu-22.04-server-cloudimg-amd64.qcow2 (687 MiB) 

## openQA 硬件环境测试方法

硬件环境下的测试需要进行一系列配置 可参考 [openQA + unmached](#)

- 修改配置文件，更换测试后端换为真机
- 配置测试机器
  - 设置好防火墙
  - 配置 ssh 访问
  - 安装开启 VNC

剩余测试步骤与 QEMU 相同

## 图形化测试

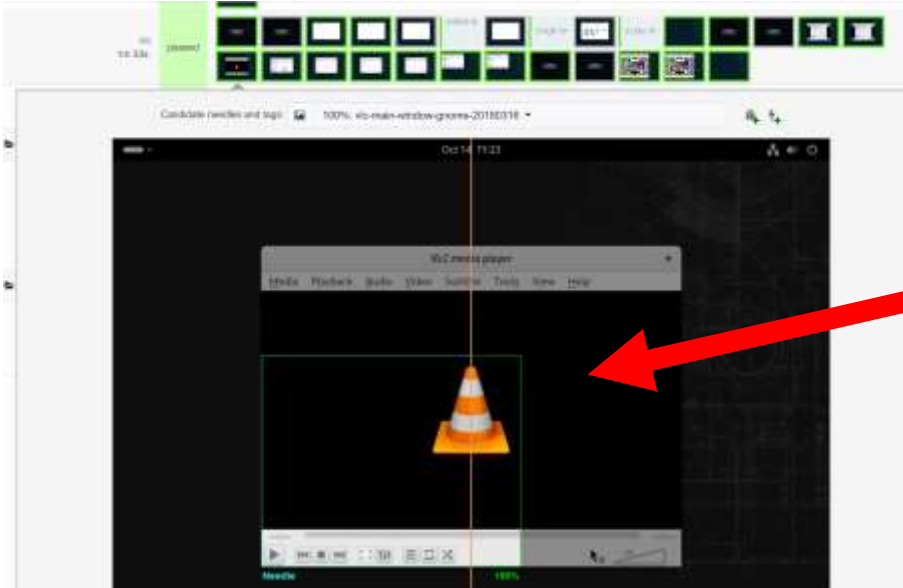
- 用户可以使用 openQA 提供了 type\_key, mouse\_click 等 API 直接操作虚拟机
- 在操作结束后可以利用 assert\_screen 相关 api 比较当前 VNC 显示是否符合预期

```
assert_screen 'winget';  
assert_and_click 'winget-install';
```

```
{  
  "area": [  
    {  
      "xpos": 7,  
      "ypos": 30,  
      "width": 400,  
      "height": 427,  
      "type": "match",  
      "click_point": {  
        "xpos": 55,  
        "ypos": 404.5  
      }  
    }  
  ],  
  "properties": [],  
  "tags": [  
    "application-finder-display",  
    "application-finder-preferences"  
  ]  
}
```



# 图形化测试原理



图中框选的部分表示要匹配的部分

```
[2023-11-07T10:22:50.970060+08:00] [debug] [pid:345] no match 7072.1s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:51.978835+08:00] [debug] [pid:345] no match 7071.1s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:53.129131+08:00] [debug] [pid:345] no match 7070.1s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:53.984658+08:00] [debug] [pid:345] no match 7069.0s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:54.986733+08:00] [debug] [pid:345] no match 7068.0s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:55.987119+08:00] [debug] [pid:345] no match 7067.0s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:56.990123+08:00] [debug] [pid:345] no match 7066.0s, best candidate: fedora-text-logged-in-loginprompt-20231106 (0.00)
[2023-11-07T10:22:58.202404+08:00] [debug] [pid:342] >>> testapi::_handle_found_needle: found ubuntu-text-login-prompt-20231106, similarity 1.00 @ 65/47
```



# 目 录

01

Ruyi SDK 简介

02

Ruyi SDK v0.2 测试策略

03

Ruyi SDK 自动化测试工具建设

04

Ruyi SDK v0.2 测试结果

# RuyiSDK v0.2 功能测试

测试阶段	测试报告
Pre_round	<a href="#">pre_round1测试报告</a> <a href="#">pre_round2测试报告</a> <a href="#">pre_round3测试报告</a> <a href="#">pre_round4测试报告</a> <a href="#">pre_round5测试报告</a>
Round1	<a href="#">SG2042 测试报告</a> <a href="#">Lichee Pi 4A openEuler 测试报告</a> <a href="#">LicheePi 4A RevyOS 测试报告</a> <a href="#">QEMU openEuler23.09 x86_64 测试报告</a> <a href="#">QEMU Ubuntu22.04 x86_64 测试报告</a> <a href="#">QEMU Fedora38 x86_64 测试报告</a> <a href="#">QEMU openEuler23.09 riscv64 测试报告</a>
Round2	<a href="#">RUYI 包管理 20231211 版本工具链可用性测试结果</a> <a href="#">SG2042 测试报告</a> <a href="#">Lichee Pi 4A openEuler 测试报告</a> <a href="#">RevyOS 20231210 riscv64 容器测试报告</a> <a href="#">QEMU openEuler23.09 x86_64 测试报告</a> <a href="#">QEMU Ubuntu22.04 x86_64 测试报告</a> <a href="#">QEMU Fedora38 x86_64 测试报告</a> <a href="#">QEMU openEuler23.09 riscv64 测试报告</a>

- 用户使用手册（参与编写和主导验证）

<https://ruyisdk.github.io/docs/zh/>

## 工具链与预置配置组合

- Ruyi 包管理在建立编译环境之前会检查该环境是否合法，但是并不保证建立成功的环境一定可用于构建。
- 经过测试可用的配置组合如右图

工具链	sysroot	预置配置
gnu-upstream	自带	generic
gnu-plct	自带	generic
gnu-plct	自带	milkv-duo
gnu-plct-xthead	自带	sipeed-lpi4a
llvm-upstream	gnu-upstream	generic
llvm-upstream	gnu-plct	generic

## 常见问题

### 1. clang-cl 命令出现有未知参数警告。

```
$ ruyi venv -t llvm-upstream --sysroot-from gnu-plct generic venv
$ . venv/bin/ruyi-activate
«Ruyi venv» $ clang-cl /help
clang-cl: warning: unknown argument ignored in clang-cl: '--gcc-install-dir=/home/hachi/.local/share/ruyi/bin/
clang-cl: warning: unknown argument ignored in clang-cl: '-mabi=lp64d' [-Wunknown-argument]
clang-cl: warning: unknown argument ignored in clang-cl: '--sysroot' [-Wunknown-argument]
OVERVIEW: clang LLVM compiler

USAGE: clang-17 [options] file...
```

解答：此 clang 别名是用来兼容 Windows MSVC 编译器的，而目前 Windows 没有 RISC-V 支持，因此预计的用户使用场景不会涉及该命令。

## 常见问题

2. xthead 工具链构建产物无法在 RevyOS 运行。

```
$ file xx
xx: ELF 64-bit LSB executable, UCB RISC-V, RVC, double-float ABI, version 1 (SYSV), dynamically linked, interp
```

解答：multilib布局问题是为了兼容rv32/rv64各种玄铁组合 revyos无此需求 所以没有做支持。

## 常见问题

3. `ruyi install slug:llvm-upstream-20231121` 找不到软件包。

解答：slug 的设计是最早没有完善的 repo、包版本匹配机制时，为了仍然能方便定位到唯一包，而投机取巧的产物。后续 slug 只会用于为超常用的软件包特定版本提供短名称，其他包默认都不会有 slug。

## 常见问题

4. `ruyi venv` 命令的输出在硬盘空间不足时会导致产生大量日志。

解答：不是功能缺陷，计划在v0.3版本进行完善。



## 测试结论

正确判定标准：按照RuyiSDK v0.2版本用户手册，实际输出和预期输出的一致。

测试结论：ruiyiSDK v0.2 功能与文档功能描述一致。

**感谢大家！**

**Q&A**