

## 《devkit 测试平台安装使用以及与Jenkins集成部署指导手册》

### 一、lkp test 添加测试用例介绍

样例

概述

添加meta.yaml描述文件

添加job YAML

添加程序

添加依赖

示例-云测工具（compatibility-test）

### 二、Jenkins Pipeline 中集成lkp test (以云测工具(compatibility-test)为示例)

1 groovy 代码

2 创建流水线

3. 执行任务

4. 查看任务执行状态

5. 查看报告

6. lkp test报告内容(以云测工具(compatibility-test)为示例)

### 二、FAQ

#### 1. 联网安装指导

权限

安装教程

证书问题

make install 遇到的问题

lkp install 遇到的问题

#### 2. lkp test 任务创建指导

1. 文件介绍

2. 必须的文件

#### 3. 离线安装指导

1.yum源配置

2.gem 配置

3.环境变量 配置

4.测试是否安装成功

测试平台使用的是lkp test 工具，以下均已lkp test描述测试平台

## 一、lkp test 添加测试用例介绍

### 样例

如下目录中的文件，完整的添加了一个典型的测试用例mentier:

```
programs/mentier/jobs/mentier-dcpmm.yaml # 在job YAML里指定想跑的programs/params
programs/mentier/jobs/mentier.yaml # 可以预定义很多的jobs
programs/mentier/meta.yaml # mentier描述文件
programs/mentier/PKGBUILD # mentier下载编译
programs/mentier/run # mentier运行脚本
programs/mentier/parse # mentier结果解析
```

如果加的program type属于monitor/setup脚本，则需要放到对应的monitors/, setup/目录下，而非programs/目录。集中存放monitor/setup脚本，有利于他人查找和复用。

其中jobs/下的YAML文件，定义了mentier的各种常见运行参数、及与其它脚本的组合。用户要跑其中的一个测试组合，典型步骤如下

```
# 把job YAML从矩阵描述形式分解为一系列原子任务
$ lkp split mentier-dcpmm.yaml
jobs/mentier-dcpmm.yaml => ./mentier-dcpmm-1-cs-localhost-0-8-1-1-65535-never-never.yaml
jobs/mentier-dcpmm.yaml => ./mentier-dcpmm-1-cs-localhost-0-24-1-1-65535-never-never.yaml
# 安装依赖，包括安装meta.yaml里depends字段描述的软件包，以及调用PKGBUILD
$ lkp install ./mentier-dcpmm-1-cs-localhost-0-8-1-1-65535-never-never.yaml
# 运行任务，会调用其中指定的各run脚本，结果保存到/lkp/result/下一个新建的目录里
# 结束后自动运行各parse脚本，提取各结果指标并汇集到stats.json
$ lkp run ./mentier-dcpmm-1-cs-localhost-0-8-1-1-65535-never-never.yaml
```

### 概述

一个测试用例一般涉及如下部分

```
1) 基本信息说明 # meta.yaml metadata部分
2) 安装哪些依赖 # meta.yaml depends字段
3) 下载编译一些程序 # PKGBUILD脚本
4) 对所在环境做哪些设置 # run脚本 (type=setup)
5) 监控系统的一些状态 # run脚本 (type=monitor)
6) 运行哪些程序，以什么参数运行 # run脚本 (type=workload)
7) 怎么解析结果，抽取度量指标 # parse脚本
```

为了实现最大的灵活性、可复用性，我们以job-program-param三层模型来组织测试用例。一个job YAML的典型内容为

```
monitor_program1:
monitor_program2:
```

```
...
setup_program1:
  param1:
  param2:
setup_program2:
  param1:
...
workload_program1:
  param1:
workload_program2:
  param1:
  param2:
```

其中每个脚本只做一件事，这样组合起来会很灵活和强大。monitor/setup programs的可复用性就很好。

用户跑一个用例的入口是job，可以自己书写job，也可以使用jobs/目录下预定义的job。当运行一个job时，lkp会找到job中指定的各类programs，以指定的params key/val为环境变量，执行各program。确切的规则如下

```
# job YAML 内容
$program:
  param1: val1
  param2: val2
# lkp install job 执行的伪代码
find programs/$program/meta.yaml or
  programs/**/meta-$program.yaml
for each package in meta YAML's depends field:
  check install package RPM/DEB
  if OS has no such package:
    find programs/$package/PKGBUILD or
    programs/**/PKGBUILD-$package
    makepkg for the first found one
# lkp run job 执行的 shell 伪代码
# run
export param1=val1
export param2=val2
find programs/$program/run or
  programs/**/run-$program
run the first found one, redirecting stdout/stderr to $RESULT_ROOT/$program
# parse
run its parse script < $RESULT_ROOT/$program | dump-stat to $RESULT_ROOT/$program.json
unite all $RESULT_ROOT/$program.json to $RESULT_ROOT/stats.json
```

## 添加meta.yaml描述文件

一个meta.yaml文件描述一个program，其结构如下

```
metadata:
  name:          # 程序名
  summary:       # 单行描述
  description:   # 多行/多段详细描述
  homepage:      # 脚本所调用程序的上游项目的主页URL
  type:          # monitor|setup|daemon|workload
  monitorType:   # one-shot|no-stdout|plain
  depends:
    gem:         # ruby gem 依赖
    pip:         # python pip 依赖
    ubuntu@22.04: # ubuntu 22.04的DEB包依赖
    openeuler@22.03: # openeuler 22.03的RPM包依赖
  pkgmap: # 各OS之间的包名映射，这样我们可以在depends里指定一个OS的完整依赖列表，通过少量包名映射来支持其它OS
    archlinux..debian@10:
    debian@10..openeuler@22.03: # 以下为两个样例
    dnstools: bind-utils
    cron: cronie
  params: # run脚本可以接受的环境变量参数，以下为样例
    runtime:
      type: timedelta
      doc: length of time, with optional human readable time unit suffix
      example: 1d/1h/10m/600s
    ioengine:
      type: str
      values: sync libaio posixaio mmap rdma
  results: # parse脚本可以从结果中提取的metrics，以下为样例
    write_bw_MBps:
      doc: average write bandwidth
      kpi: 1 # weight for computing performance index; negative means the larger the worse
```

## 添加job YAML

一般我们需要主要跑一个type=workload的program，同时再跑一些type=monitor/setup/daemon的programs，加上它们的参数，构成一个完整的测试用例。我们用一个个的job YAML来描述这些测试用例。

所以预定义job YAML大体上可以按workload来组织，放在路径下

```
programs/$workload/jobs/xxx.yaml
```

当然也可以按更大粒度来组织，比如场景、测试类型等分类，此时可以放在路径下

```
jobs/$test_scene/xxx.yaml
jobs/$test_class/xxx.yaml
```

以上预定义jobs的搜索路径，lkp框架代码都支持。具体path glob pattern是

```
programs/*/jobs/*.yaml
jobs/**/*.*.yaml
```

## 添加程序

Job YAML中引用的programs，需要您预先写好，lkp会在如下路径搜索其文信息/脚本：

```
1st search path          2nd search path
programs/$program/meta.yaml  programs/**/meta-$program.yaml
programs/$program/{run,parse} programs/**/{run,parse}-$program
programs/$package/PKGBUILD   programs/**/PKGBUILD-$package
```

程序一般添加到 programs/\$program/ 目录下，具体添加以下几个脚本

```
programs/$program/meta.yaml # 描述文件
programs/$program/run        # 接收/转换环境变量传过来的参数，运行目标程序
programs/$program/parse      # 解析结果(一般是run的stdout)，输出metrics (YAML key=val)
programs/$program/PKGBUILD   # 下载编译安装run调用的目标程序
tests/$program => ../programs/$program/run # 创建符号链接 保持兼容
```

其中PKGBUILD仅必要时添加。parse一般在program type=monitor/workload时才需要。

一般一个program一个目录。但有时候client/server类型的测试，把workload+daemon programs放在一起比较方便。此时可以参照sockperf，把sockperf-server daemon以如下方式添加到sockperf workload目录下：

```
programs/sockperf/meta-sockperf-server.yaml
programs/sockperf/run-sockperf-server
```

## 添加依赖

一个program的依赖表述为

```
programs/$program/meta.yaml
  depends:
    - debian@10:
      - $package1
      - $package2
  pkgmap:
    debian@10..centos@8: # centos 8不自带$package2，映射为空
      $package2:
        programs/$program/PKGBUILD-$package1
        programs/$program/PKGBUILD-$package2
```

这里定义了两类依赖 1) OS自带的包 2) 需要从源码下载编译的包 当OS包含package1/package2时，lkp框架可自动安装对应的rpm/deb; 如果没有，再使用PKGBUILD-xxx构建出包。

例如，在debian 10中，lkp install会执行

```
apt-get install $package1 $package2
```

在在centos 8中，lkp install会执行

```
yum install $package1
makepkg PKGBUILD-$package2 # 从源码下载编译
```

如您希望强制从源码编译下载，无论所在OS是否包含RPM/DEB包，那么可以通过指定PKGBUILD依赖

```
depends:
  PKGBUILD:
    - $package1
```

那么lkp install会无条件编译\$package1

注意，PKGBUILD语义上对应一个package，而不是对应program。这两者语义上不同，虽然很多时候两者内容是一样的。当内容一样时，比如

```
programs/$program/PKGBUILD-$package
```

也可以写为简化形式

```
programs/$program/PKGBUILD # when $package=$program
```

注意，PKGBUILD文件名及其内部depends/makedepends字段里的\$package使用的是archlinux包名。所以其它OS缺失此包，或者有此包，但是名字不一样的话，需要配置对应的pkgmap包名映射，或者加上OS后缀，比如

```
makedepends_debian_11=(lam4-dev libopenmpi-dev libmpich-dev pvm-dev)
```

## 示例-云测工具（compatibility-test）

1. 在programs 文件夹下创建compatibility-test文件夹，里面至少要包含以下几个文件，其余文件可以根据需求自行决定是否添加  
programs/compatibility-test/jobs/compatibility-test.yaml # 预定义compatibility-test的job，需要与文件夹名字一致  
programs/compatibility-test/meta.yaml # compatibility-test描述文件  
programs/compatibility-test/run # compatibility-test运行脚本

programs/compatibility-test/jobs/compatibility-test.yaml:

```
programs/compatibility-test/meta.yaml:
```

```
programs/compatibility-test/run:
```

### 3. 必要步骤

在完成此文件夹的创建后，依然还需要两步操作去让lbp命令找到指定的运行文件

## 二、Jenkins Pipeline 中集成lkp test (以云测工具(compatibility-test)为示例)

## 1 groovy 代码

```
stage('lkp test') {
    steps {
        script{
            echo '===== lkp test ====='
            sh '''
                cp -rf /home/lj/test/compatibility_testing/template.html.bak
/home/lj/test/compatibility_testing/template.html # 用于最终生成报告
                sudo /home/lj/lkp-tests-master/bin/lkp run /home/lj/lkp-tests-
master/programs/compatibility-test/compatibility-test-defaults.yaml
                cp -rf
/home/lj/test/compatibility_testing/compatibility_report.html ./compatibility_report.html
            '''
            sh(script: "sudo bash
/home/lj/test/compatibility_testing/Chinese/test_result.sh", returnStdout:true).trim() # 用来根据报告结
果设置流水线结果
        }
    }
    post {
        always {
            publishHTML(target: [allowMissing: false,
                                alwaysLinkToLastBuild: false,
                                keepAll: true,
                                reportDir: '.',
                                reportFiles:

```

```
'compatibility_report.html',
Report']
    }
}
}
```

reportName : 'compatibility test'

## 2 创建流水线

The screenshot shows the Jenkins dashboard. On the left sidebar, the 'New Item' button (labeled '新建任务') is highlighted with a red arrow. The main content area displays 'Welcome to Jenkins!' and 'Start building your software project' with a 'Create a Job' button. Below the dashboard, a modal window titled '输入一个任务名称' (Enter a task name) is shown. It contains a text input field with 'DevKitScanning' and a dropdown menu with '自定义任务名称' (Custom task name). Below the input field, there are several options with icons: '构建一个自由风格的软件项目' (Build a free-style software project), '流水线: 选择流水线(Pipeline)选项' (Pipeline: Select Pipeline option), '构建一个多配置项目' (Build a multi-configuration project), '(0) 文件夹' (0) Folder, '多分支流水线' (Multibranch pipeline), and '文件夹' (Folder). The 'Pipeline' option is highlighted with a red arrow. At the bottom of the modal, there is a '确定' (Confirm) button.

Dashboard > lkp-test > Configuration

**Configure**

**General**

高级项目选项

流水线

高级项目选项

高级

流水线

定义

Pipeline script

脚本

将上文提到的流水线代码写入

```
1 pipeline {
2   agent any
3   options {
4     timeout(time: 1, unit: 'HOURS')
5   }
6   stages {
7     stage 'lcp test' {
8       steps {
9         script {
10           echo '===== lcp test ====='
11           sh ""
12           sudo /home/j2/lkp-test-master/bin/lcp_run /home/j2/lkp-test-master/program/compatibility-test/compatibility-test-defaults.yml
13           ... /home/j2/test/compatibility_testing/compatibility_report.html /compatibility_report.html
14           sh(script: "sudo bash /home/j2/test/compatibility_testing/chinese/test_result.sh", returnStdout:true).trim()
15         }
16       }
17     }
18     post {
19       always {
20         publishJUnitTarget: [skipPublishing: false,
21                               alwaysPublishTestResults: false,
22                               keepAll: true,
23                               reportFile: 'compatibility_report.html',
24                               reportFile: 'compatibility_report.html',
25                               reportName: 'compatibility Test Report']
26       }
27     }
28   }
29 }
30 }
```

使用 Groovy 沙盒

流水线描述

保存 应用

### 3. 执行任务

Dashboard > lkp-test >

**状态**

变更历史

立即构建

配置

删除 Pipeline

完整阶段视图

compatibility test Report

收藏夹

打开 Blue Ocean

重命名

流水线语法

**阶段视图**

Average stage times:  
(Average full run time: ~12min  
3s)

Stage	Time	Status
#43	3月 17 日 13:05	No Changes
#42	3月 17 日 12:01	No Changes
#41	3月 17 日 11:58	No Changes

**lkp test**

9min 24s

18min 16s

18min 1s


14s

aborted

构建历史

趋势

### 4. 查看任务执行状态

**Jenkins**

Dashboard > lkp-test > #43

状态

</> 变更历史

Console Output

View as plain text

编辑构建信息

删除构建 '#43'

compatibility test Report

打开 Blue Ocean

从指定阶段重新运行

回放

流水线步骤

Workspaces

← 上一次构建

Build #43 (2024年3月17日 下午1:05:49)

启动用户AntsCode

表示当前任务执行成功

## 5. 查看报告

Dashboard > lkp-test > #43

状态

</> 变更历史

Console Output

View as plain text

编辑构建信息

删除构建 '#43'

compatibility test Report

打开 Blue Ocean

从指定阶段重新运行

回放

流水线步骤

Workspaces

← 上一次构建

Build #43 (2024年3月17日 下午1:05:49)

启动用户AntsCode

对应代码中的报告名称

## 6. lkp test报告内容(以云测工具(compatibility-test)为示例)

鲲鹏测试报告

Search:

id	result	reason	evidence
Compatibility_Application_Start	passed		#2024-03-23 23:08:45#info#业务应用test1自动启动
Compatibility_Application_Stop	passed		#2024-03-23 23:03:24#info#业务应用test1正常存在
Compatibility_Hardware_Server	passed	硬件包含鲲鹏芯片，符合要求	
Compatibility_Idle_Cpu	passed		Average: all 0.51 0.00 0.36 0.00 0.00 99.13,Average: all 0.52 0.00 0.38 0.00 0.00 99.09,被测软件中启动前采集CPU资源利用率与被测软件中停止后CPU利用率之间的差值为0.04 %,小于 1.00 %
Compatibility_Idle_Disk	passed		Average: 3.88 0.00 15.53 0.00 4.00 0.00 0.43 0.17 ,openuler-root,Average: 3.65 0.00 14.60 0.00 4.00 0.00 0.04 0.23 ,openuler-root,被测软件启动前采集硬盘资源利用率与被测软件停止后硬盘利用率之间的差值为0.06 %,小于 1.00 %
Compatibility_Idle_Memory	passed		Average: 2331054 5771318 707997 10.38 757112 2535802 1544980 10.24 2225948 1701718 98,Average: 2330568 5771117 708112 10.38 757116 2536093 1545496 10.24 2225948 1702168 135,被测软件启动前采集内存资源利用率与被测软件停止后内存利用率之间的差值为0.00 %,小于 1.00 %
Compatibility_Idle_Network	passed		Average: emp1s0 6.10 3.00 0.54 0.30 0.00 0.00 0.00 0.00,Average: emp1s0 5.65 3.05 0.43 0.30 0.00 0.00 0.00 0.00,被测软件启动前采集网卡资源利用率与被测软件停止后采集网卡资源利用率之间的差值为0.00 %,被测软件启动前的1.00%,Average: docker0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00,Average: docker0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00,被测软件启动前采集网卡资源发送数据与被测软件停止后采集网卡资源发送数据之间的差值为0.00 %,发送测试前成功的1.00%
Compatibility_Software_Name	passed		
Reliability_Exception_Kill			
Reliability_Pressure_Disk	passed		11:12:31 PM 5.80 0.00 23.20 0.00 4.00 0.00 0.00 0.24 ,openuler-root,11:12:56 PM 2.80 0.00 11.20 0.00 4.00 0.00 0.00 0.08 ,openuler-root,压力测试期间磁盘 资源波动值为0.16 %,小于 5.00%

<

1

2

>

## 二、FAQ

### 1. 联网安装指导

#### 权限

lkp test 运行需要root用户运行，所以在自动化的过程中有一个切root的操作

#### 安装教程

<https://docs.openeuler.org/zh/docs/22.09/docs/certification/%E6%B5%8B%E8%AF%95%E6%A0%87%E5%87%86%E5%92%8C%E6%B5%8B%E8%AF%95%E5%B7%A5%>

#### 证书问题

```
https://rubygems.org/ removed from sources
[root@server2 lkp-tests]# gem sources -a https://gems.ruby-china.com/

Error fetching https://gems.ruby-china.com/:
SSL_connect returned=1 errno=0 state=error: wrong version number (https://gems.ruby-china.com/specs.4.8.gz)
```

[解决方式]: <https://developer.baidu.com/article/details/2821747>

#### make install 遇到的问题

1. 未从更新后的镜像源拉取资源

```
bash sbin/install-dependencies.sh
Installing packages ruby rubygems make gcc diffutils util-linux lftp hostname sudo
gzip git rubygem-json rubygem-bundler gcc-c++ ruby-devel rubygem-rake rpm-build
Installing ruby gems...
source https://rubygems.org/ not present in cache
```

[解决方式]:

```
1 #!/bin/sh
2
3 . $LKP_SRC/lib/detect-system.sh
4
5 setup_gem_china()
6 {
7     # https://mirrors.ustc.edu.cn/help/rubygems.html
8     gem sources -a http://mirrors.aliyun.com/rubygems/
9     # Note: centos7 does not work with tuna/ruby-china
10    gem sources -a https://mirrors.tuna.tsinghua.edu.cn/rubygems/
11    gem sources -a https://mirrors.ustc.edu.cn/rubygems/      #添加科大源
12    gem sources -a https://gems.ruby-china.com/
13    # Note: ustc looks not reliable
14    # bundle config mirror.https://rubygems.org https://gems.ruby-china.com
15 }
```

将 lib/install.sh里第8行替换为可用的镜像源，以及可以屏蔽掉10-12行中不可用的镜像源

2. 卡死报错

```
[root@master01 lkp-tests]# make install
bash sbin/install-dependencies.sh
Installing packages ruby rubygems make gcc diffutils util-linux lftp hostname sudo gzip git rubygem-json rubygem-bundler gcc-c++ ruby-devel rubygem-rake rpm-build
Installing ruby gems...
source http://mirrors.aliyun.com/rubygems/ already present in the cache
```

```
Error fetching https://mirrors.tuna.tsinghua.edu.cn/rubygems/:
too many connection resets (https://mirrors.tuna.tsinghua.edu.cn/rubygems/specs.4.8.gz)
https://mirrors.ustc.edu.cn/rubygems/ added to sources
Error fetching https://gems.ruby-china.com/:
SSL_connect returned=1 errno=0 state=error: wrong version number (https://gems.ruby-china.com/specs.4.8.gz)
Traceback (most recent call last):
  2: from /usr/bin/bundle:23:in `<main>'
  1: from /usr/share/rubygems/rubygems.rb:308:in `activate_bin_path'
/usr/share/rubygems/rubygems.rb:289:in `find_spec_for_exe': can't find gem bundler (>= 0.a) with executable bundle (Gem::GemNotFoundException)
```

[解决方式]:

安装bundler

首先运行

```
cat Gemfile.lock | grep -A 1 "BUNDLED WITH"
```



```
[root@master01 lkp-tests]# cat Gemfile.lock | grep -A 1 "BUNDLED WITH"
BUNDLED WITH
2.2.33
[root@master01 lkp-tests]# yum install bundler -v '2.2.33'
Loaded plugins: builddep, changelog, config-manager, copr, debug, debuginfo-install, download, generate_completion_cache, needs-restarting, playground, repoclosure, repodiff, repograph, repo
manage, reposync
YUM version: 4.2.23
```

安装相同版本

```
gem install bundler -v 2.2.33
```

### 3. 为更新gem缓存导致更新的镜像源未生效

```
[root@master01 lkp-tests]# make install
bash sbin/install-dependencies.sh
Installing packages ruby rubygems make gcc diffutils util-linux lftp hostname sudo gzip git rubygem-json rubygem-bundler gcc-c++ ruby-devel rubygem-rake rpm-build
Installing ruby gems...
source http://mirrors.aliyun.com/rubygems/ already present in the cache
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and installing your bundle as root will break this application for all non-root users on this machine.
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:232: warning: already initialized constant Psych::LIBYAML_VERSION
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:234: warning: already initialized constant Psych::LIBYAML_VERSION was here
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:234: warning: already initialized constant Psych::FALLBACK
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:234: warning: already initialized constant Psych::FALLBACK was here
Fetching source index from https://rubygems.org/

Retrying fetcher due to error (2/4): Bundler::HTTPError Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version nu
mber (https://rubygems.org/specs.4.8.gz)>

Retrying fetcher due to error (3/4): Bundler::HTTPError Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version nu
mber (https://rubygems.org/specs.4.8.gz)>

Retrying fetcher due to error (4/4): Bundler::HTTPError Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version nu
mber (https://rubygems.org/specs.4.8.gz)>

Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version number (https://rubygems.org/specs.4.8.gz)>
make: *** [Makefile:7: install] Error 17
```

[解决方式]:

```
# 运行命令，删除无效镜像源
gem source
# 在更新镜像源后需要更新gem缓存，让更新的镜像源生效
gem source -u
```

### 4. bundle镜像源配置问题

```
[root@master01 lkp-tests]# make install
bash sbin/install-dependencies.sh
Installing packages ruby rubygems make gcc diffutils util-linux lftp hostname sudo gzip git rubygem-json rubygem-bundler gcc-c++ ruby-devel rubygem-rake rpm-build
Installing ruby gems...
source http://mirrors.aliyun.com/rubygems/ already present in the cache
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and installing your bundle as root will break this application for all non-root users on this machine.
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:232: warning: already initialized constant Psych::LIBYAML_VERSION
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:232: warning: already initialized constant Psych::LIBYAML_VERSION was here
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:234: warning: already initialized constant Psych::FALLBACK
/usr/share/gems/gems/psych-3.0.2/lib/psych.rb:234: warning: already initialized constant Psych::FALLBACK was here
Fetching source index from https://rubygems.org/

Retrying fetcher due to error (2/4): Bundler::HTTPError Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version nu
mber (https://rubygems.org/specs.4.8.gz)>

Retrying fetcher due to error (3/4): Bundler::HTTPError Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version nu
mber (https://rubygems.org/specs.4.8.gz)>

Retrying fetcher due to error (4/4): Bundler::HTTPError Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version nu
mber (https://rubygems.org/specs.4.8.gz)>

Could not fetch specs from https://rubygems.org/ due to underlying error <SSL_connect returned=1 errno=0 state=error: wrong version number (https://rubygems.org/specs.4.8.gz)>
make: *** [Makefile:7: install] Error 17
```

[解决方式]:

```
# 替换bundle镜像源
bundle config mirror.https://rubygems.org https://mirrors.aliyun.com/rubygems
```

### 5. 安装超时报错

```
Bundler::HTTPError: Could not download gem from https://mirrors.aliyun.com/rubygems/ due to underlying error <Errno::ECONNREFUSED: Failed to open TCP connection to 90.253.6.207:8080
(Connection refused - connect(2) for "90.253.6.207" port 8080) (https://mirrors.aliyun.com/rubygems/gems/zeitwerk-2.6.5.gem)>
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/rubygems_integration.rb:549:in 'rescue in download_gem'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/rubygems_integration.rb:521:in 'download_gem'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/source/rubygems.rb:527:in 'download_gem'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/source/rubygems.rb:479:in 'fetch_gem'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/source/rubygems.rb:165:in 'install'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/installer/gem_installer.rb:54:in 'install'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/installer/gem_installer.rb:16:in 'install_from_spec'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/installer/parallel_installer.rb:186:in 'do_install'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/installer/parallel_installer.rb:177:in 'block in worker_pool'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/worker.rb:62:in 'apply_func'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/worker.rb:57:in 'block in process_queue'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/worker.rb:54:in 'loop'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/worker.rb:54:in 'process_queue'
/usr/local/share/gems/gems/bundler-2.2.33/lib/bundler/worker.rb:91:in 'block (2 levels) in create_threads'

An error occurred while installing zeitwerk (2.6.5), and Bundler cannot continue.

In Gemfile:2
  activerecord was resolved to 6.1.7, which depends on
    zeitwerk
```

[解决方式]:

安装时间过长，连接中断，重新运行make install即可解决

### lkp install 遇到的问题

#### 1. 报错，系统不支持

```
[root@master01 lkp-tests]# lkp install
Not a supported system, cannot install packages.
```

[解决方式]:

环境变量中增加 LKP\_SRC，路径和LKPPATH一样export PATH=LKP\_PATH 一样 export PATH=LKP\_PATH一样export PATH=PATH/home/ij/lkp-tests/sbin/home/ij/lkp-tests/bin/home/ij/lkp-tests/sbin/home/ij/lkp-tests/bin  
export LKP\_PATH=/home/ij/lkp-tests  
export LKP\_SRC=/home/ij/lkp-tests

## 2. lkp test 任务创建指导

### 1. 文件介绍

doc/add-testcase.zh.md · Fengguang/lkp-tests - Gitee.com

### 2. 必须的文件

run（可执行脚本）

meta.yaml (介绍项目的详细信息)

jobs 文件夹以及文件夹内需要包含一个与program同名的yaml文件

```
lkp split xxx.yaml # 这个yaml是jobs文件夹里的，在哪里执行这个命令，分割出来的任务就会在哪  
lkp run xxxx.yaml # 这个yaml是上一步分割完后生成的yaml
```

### 3. 离线安装指导

#### 1.yum源配置

请配置everything的yum源

<https://repo.huaweicloud.com/openEuler-20.03-LTS/ISO/aarch64/>

#### 2.gem 配置

去<https://gems.ruby-china.com/> 下载以下gem依赖

bundler 2.2.33, diff-lcs 1.5.0, minitest 5.15.0 concurrent-ruby 1.1.10, docile 1.4.0, rchardet 1.8.0,  
gnuplot 2.6.2, parallel 1.22.1, public\_suffix 4.0.7, regexp\_parser 2.6.0, rexml 3.2.5, ast 2.4.2,  
rainbow 3.1.1, rspec-support 3.12.0, ruby-progressbar 1.11.0, unicode-display\_width 2.3.0,  
git 1.7.0, simplecov\_json\_formatter 0.1.4, simplecov-html 0.12.3, rspec-core 3.12.0, rspec-expectations 3.12.0,  
rspec-mocks 3.12.0, il8n 1.12.0, builder 3.2.4, sync 0.5.0, tzinfo 2.0.5, rspec 3.12.0, ci\_reporter 2.0.0,  
ci\_reporter\_rspec 1.0.0, parser 3.1.2.1, tins 1.31.1, term-ansicolor 1.7.1, rubocop 1.12.1, simplecov 0.21.2,  
simplecov-rcov 0.3.1

将以上依赖放到 /usr/share/gems/gems

并执行gem install --local 安装以上依赖

#### 3.环境变量 配置

export PATH=\$PATH:lkptest路径/lkp-tests/sbin:lkptest路径/lkp-tests/bin:lkptest路径/lkp-tests/bin

export LKP\_PATH=lkptest路径/lkp-tests

export LKP\_SRC=lkptest路径/lkp-tests

#### 4.测试是否安装成功

```
lkp help  
lkp install
```