**DB组CI环境治理**

【后续有变更，请继续更新该文档】

# CI总体流程

DB内核代码的CI当前构建如下表所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| master | kw | coverity | st | coverage | funcheck | lizard | 构建版本 |
| VerifyCI | √ | √ | × | × | √ | √ | × |
| MergeCI | × | × | × | × | × | √ | √ |
| DailyCI | × | × | √ | √ | × | √ | × |
| ManualCI | 可选 | 可选 | 可选 | 可选 | × | √ | 可选 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| allmaster | kw | coverity | st | coverage | funcheck | lizard | 构建版本 |
| VerifyCI | √ | √ | × | × | √ | √ | × |
| MergeCI | × | × | × | × | × | √ | √ |
| DailyCI | × | × | √ | √ | × | √ | × |
| ManualCI | 可选 | 可选 | 可选 | 可选 | × | √ | 可选 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 各个branch | kw | coverity | st | coverage | funcheck | lizard | 构建版本 |
| VerifyCI | **√** | **√** | × | × | √ | √ | × |
| MergeCI | × | × | × | × | × | √ | √ |
| DailyCI | × | × | √ | × | × | √ | × |
| ManualCI | 可选 | 可选 | 可选 | 可选 | × | √ | 可选 |

（其中，标红是后续新拉branch分支需要注意，处理的。）

DailyCI定时触发机制在Jenkinsfile脚本的set\_default\_properties函数中，表示的是每天的定时触发的时间点，master是每天0点到0:29分之间随机触发，allmaster是在每天4点到4:29分之间随机触发，branch分支是在每天的0点到3点之间随机触发。

针对branch分支，新拉一个branch分支时需要根据实际情况指定branch分支执行区间，每一个活跃的分支间隔2小时。时规划为0点跑一个分支，2点跑一个分支，4点跑一个分支，6点跑一个分支。已经废弃的分支时间可以覆盖。

|  |
| --- |
| def set\_default\_properties() {  ......  /\* 根据分支JOB dailyci定时设置\*/  if(plll.checkBranchName('master')) {  cron = 'H(0-29) 0 \* \* \*'  }else if(plll.checkBranchName('allmaster')){  cron = 'H(0-29) 4 \* \* \*'  }else if("${plll.getUserName()}" != "branchindexing"){  cron = 'H H(0-3) \* \* \*'  }  ......  } |

当前master分支的所有CI脚本全部规整到DB代码代码目录DB/CI/目录下，后续有需要新增或者修改，请再该目录下进行。

对于老的branch分支，继续沿用以前的，代码不做修改。对于后续新增的branch分支，需要依据master或者allmaster分支进行CI脚本修改。

# 禁止某一个分支执行某一个流程

禁止某一个分支执行一个具体的功能检查流程，则需要在具体的功能函数中进行限制。

示例：如果需要禁止ZXCLOUD-GOLDENDB-ALL-DBV5.1.01版本跑kw功能检查，则在该分支上的Jenkinsfile代码中加入如下限制：

|  |
| --- |
| def run\_kw(){  **if(plll.checkBranchName('ZXCLOUD-GOLDENDB-ALL-DBV5.1.01'))**  **{**  **print 'skip ALL-DBV5.1.01 kw check'**  **return**  **}**  ......  } |

# 新建branch分支CI流程

1. 修改合入版本号：

VERSION

mysql-test\r\select\_goldendb\_version.result

autoupdate\upgrade\_db.sh

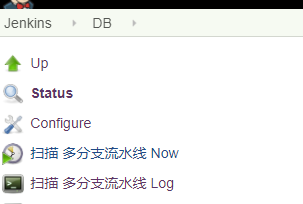
autoupdate\upgrade\_db\_new.sh

autoupdate\modify\_cnf.ini

1. 修改合入ST脚本，构建ST测试环境；
2. 修改合入Jenkinsfile中定时时间，以及其它需要修改的地方；
3. 在configure中增加branch分支名称，拉出该branch分支；
4. 构建kw全量检测；
5. 构建coverity全量检测；

# 从CI上摘除一个branch流程

1）如下图所示，从configure配置中去除相应的branch分支名称。



2）如果该分支跑ST的话，则去相应的ST服务器，删除相应的branch分支的ST测试目录。

# Coverity

## coverity检查逻辑

coverity部分的jenkinsfile中代码主要调用见run\_coverity()函数。增量的coverity检查耗时约10分钟左右，可能更长，而全量的coverity检查耗时约9小时。其主要逻辑如下：

（1）VerifyCI触发增量coverity检查；

（2）MergeCI即不触发增量coverity检查，也不触发全量coverity检查；

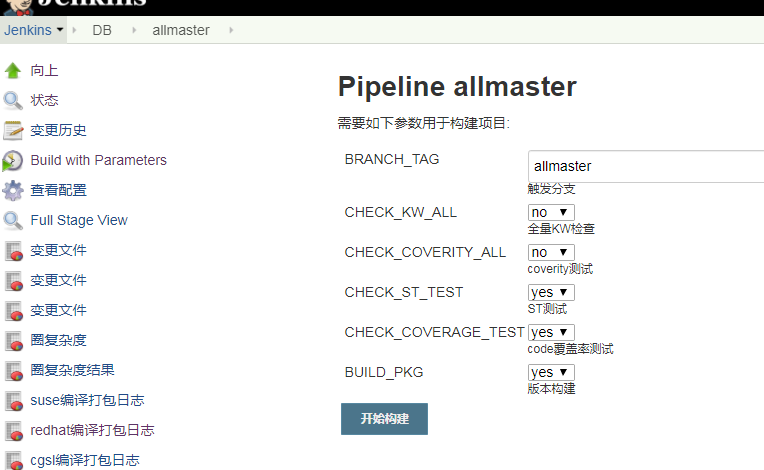
（3）DailyCI不触发全量coverity检查；

（4）manualCI默认不触发全量coverity检查。

（5）触发全量coverity检查的条件为：manualCI+设置CHECK\_COVERITY\_ALL参数为yes。

（6）针对master，allmaster和所有在研branch分支都进行检查。

manualCI设置参数如下，按照需要选择对应的检查项即可，以allmaster为例：



coverity脚本在DB代码目录，脚本为CI/coverity/coverity.sh。

coverity检查工具的使用问题，可以从项目icenter空间获取，搜索关键词“coverity”。

[http://i.zte.com.cn/#/shared/0a69b28973c94d639cd039915ee6feaa/wiki/page/bf141b76efa64f12ac6e63da79ba71f3/view?track\_id=7f181ad046284d578d84e2497528cc5d](http://i.zte.com.cn/" \l "/shared/0a69b28973c94d639cd039915ee6feaa/wiki/page/bf141b76efa64f12ac6e63da79ba71f3/view?track_id=7f181ad046284d578d84e2497528cc5d)

## Coverity环境信息

目前最新的coverity服务器信息为：

http://10.46.150.158:8080 goldendb/goldendb-ci （新服务器）

master支持的流名称为：ZXCLOUD-GOLDENDB-DBVTRUNK

allmaster支持的流名称为：ZXCLOUD-GOLDENDB-ALL-DBVTRUNK

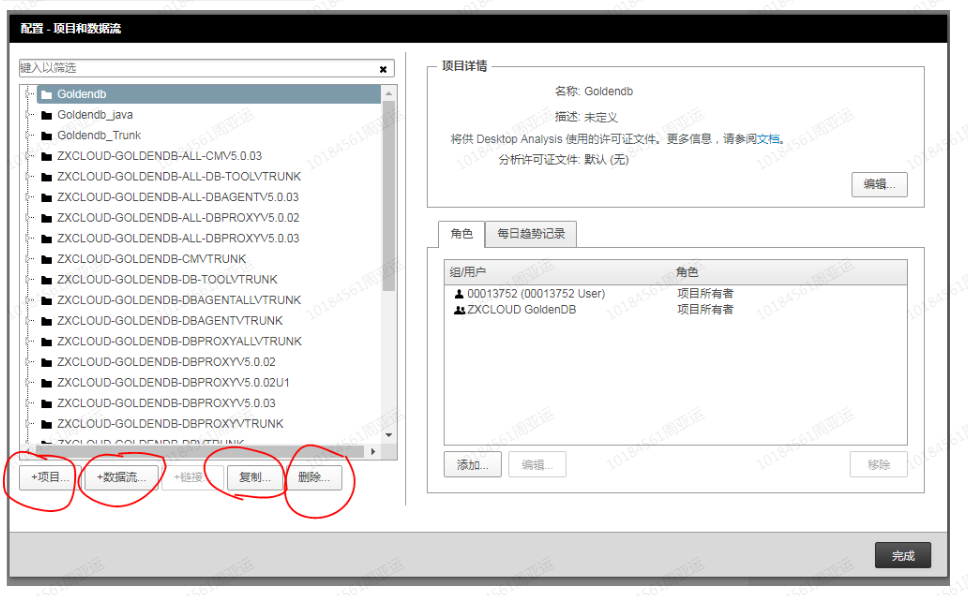
（PS:如果服务器信息更新，则需要更新coverity脚本中相应的服务器信息，并且根据新服务器中是否有全量数据来判断是否要做一次全量KW检查。）

## 新建branch分支

**步骤一**：登录coverity服务器：<http://10.46.150.158:8080>

**步骤二**：根据icenter上（2.1节内容）教程在该coverity中新建一个project和stream。

注意新建的项目名称和stream名称都必须和branch分支名称一样。



（PS：新建的项目stream如果没有数据，搜索时将找不到，只能在配置这个界面中看到。）

**步骤三**：拷贝Jenkinsfile文件

Master分支的branch版本从master分支拷贝对应的Jenkinsfile文件；

ALLMASTER分支的branch版本从allmaster分支拷贝对应的Jenkinsfile文件；

（PS:此处结合根据实际情况，目的是避免branch分支继承了较老的Jenkinsfile脚本。例如branch分支的CI脚本没有完成规整，仍然分散在各个地方。）

**步骤四**：手动构建一次coverity全量工程，待完成后去对应服务器检查是否存在数据。

**注：**可利用已有branch分支已经搭建好的coverity环境来构建新branch分支的coverity环境，只需登录branch分支的coverity服务器，修改已有branch分支的项目名和数据流名为新branch分支的名称。

# KW

## Kw检查逻辑

Kw部分的jenkinsfile中代码主要调用见run\_kw()函数。增量的kw检查耗时约5分钟以内，而全量的kw检查耗时约3小时。其主要逻辑如下：

1. VerifyCI触发增量KW检查；
2. MergeCI即不触发增量KW检查，也不触发全量KW检查；
3. DailyCI不触发全量KW检查；
4. manualCI默认不触发全量KW检查。
5. 触发全量kw检查的条件为：manualCI+设置CHECK\_KW\_ALL参数为yes。
6. 针对master，allmaster和所有在研branch分支都进行检查。

Kw脚本在DB代码目录，增量脚本为CI/kw/kw\_inc.sh，全量脚本为CI/kw/make\_kw\_pipe.sh。

Kw检查工具的使用问题，可以从项目icenter空间获取，搜索关键词“kw”。

[http://i.zte.com.cn/#/shared/0a69b28973c94d639cd039915ee6feaa/wiki/page/f39939f3049542ed943818782eb7516a/view?track\_id=c69e1820f34b42fa8c1cf153e0325b1e](http://i.zte.com.cn/" \l "/shared/0a69b28973c94d639cd039915ee6feaa/wiki/page/f39939f3049542ed943818782eb7516a/view?track_id=c69e1820f34b42fa8c1cf153e0325b1e)

## Kw环境信息

**账户信息**：

kw管理员账号密码 （仅用于新建JOB时候使用）

账号： klocwork

密码： zte@123

普通用户和密码 （用于解决kw问题及其它场景使用）

用户：root

密码：（空）

**服务器信息**：

http://10.46.150.182:8080/ (用于master分支和allmaster分支 )

目前已经有了2个分支如下：

goldendb-DB-trunk 对应master分支

goldendb\_all\_master\_db 对应allmaster分支

http://10.46.150.183:8080/ （用于所有branch分支）

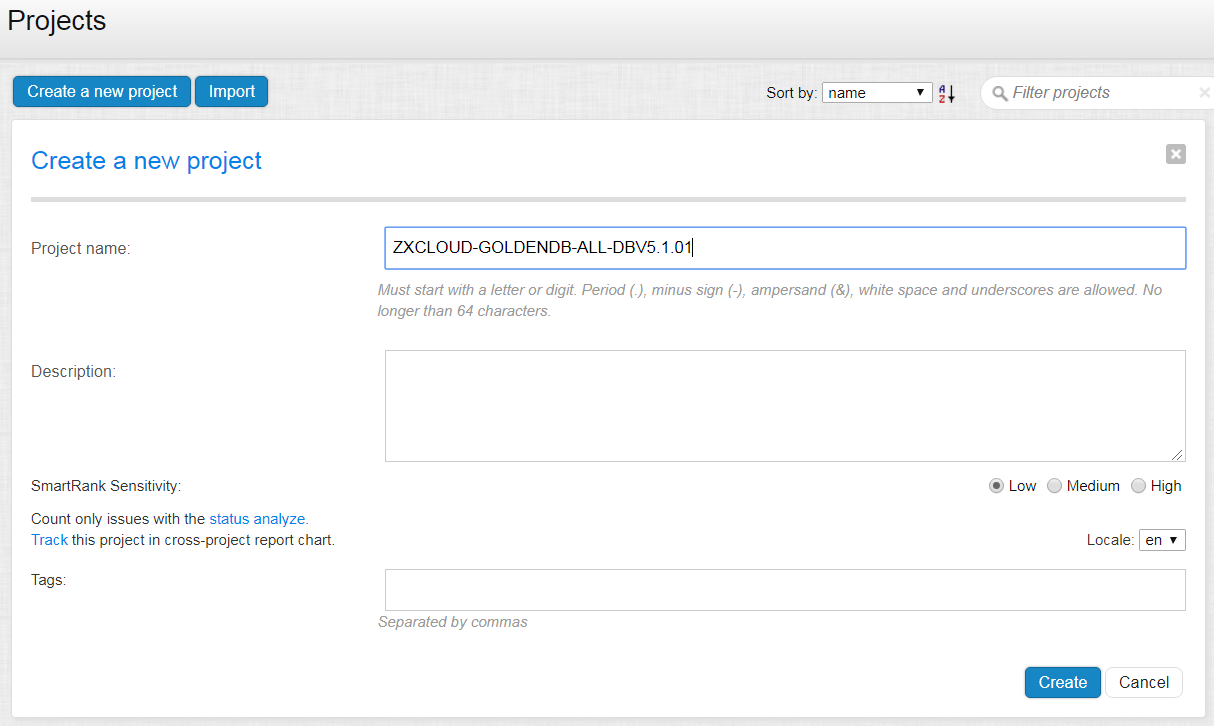
（PS:如果服务器信息更新，则需要更新Jenkinsifle中相应的服务器信息，并且根据新服务器中是否有全量数据来判断是否要做一次全量KW检查。）

## 新建branch分支

**步骤一**：使用管理员账号登陆服务器：<http://10.46.150.183:8080/>

**步骤二**：新建一个和branch名称一样的JOB工程，

如ZXCLOUD-GOLDENDB-ALL-DBV5.1.01版本。



**步骤三**：拷贝Jenkinsfile文件

Master分支的branch版本从master分支拷贝对应的Jenkinsfile文件；

ALLMASTER分支的branch版本从allmaster分支拷贝对应的Jenkinsfile文件；

（PS:此处结合根据实际情况，目的是避免branch分支继承了较老的Jenkinsfile脚本。例如branch分支的CI脚本没有完成规整，仍然分散在各个地方。）

**步骤四**：手动构建一次kw全量工程，待完成后去对应服务器检查是否存在数据。

**注：**可利用已有branch分支已经搭建好的kw环境来构建新branch分支的kw环境，只需登录branch分支的kw服务器，修改已有branch分支的JOB工程名为新branch分支的名称。

# ST

## ST检查逻辑

ST部分的jenkinsfile中代码主要调用见run\_st()函数，其检查的逻辑为：

1. 在DailyCI中跑ST测试用例；
2. 可以通过ManualCI+设置CHECK\_ST\_TEST手动触发跑全量的自动化ST测试；
3. 所有master分支，allmaster分支，banch分支均需要跑ST。

如果新增branch分支不跑ST的话，需要在在run\_st函数中增加skip代码，详见第8章类容。

其ST测试包含4个文件：入口调用脚本update\_and\_execute\_it.sh，deploy\_it.sh脚本，execute\_it.sh脚本，以及mysql-test\_install\_help.txt说明文档。

## ST环境信息

目前ST测试环境是我们DB组自己提供的，

使用2个高核环境作为ST测试环境，分别如下：

10.46.178.97 root/db1x@NJ+1 用于测试master分支和allmaster分支ST用例；

10.46.178.124 root/db1x@NJ+1 用于测试branch分支的ST用例；

（PS：目前已经在97环境跑的ST用例不用变更，后续需要新增branch的ST时，需要在135机器上部署，在124上部署的测试用例全都位于/home/ST\_branch目录下，每一个branch分支均以”ST\_分支名”的格式命名，如ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01）

## 新建branch-ST流程

对于新建branch分支来说，以ZXCLOUD-GOLDENDB-ALL-DBV5.1.01为例来说其步骤如下（参考mysql-test\_install\_help.txt说明文档）：

mysql-test自动化测试用例部署环境：

1、登录到ST\_branch对应的机器环境（10.46.178.124）

2、建立一个branch分支测试目录

/home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01

3、建立一份代码克隆路径

进去branch分支目录；

git clone ...

增加了一个目录为：

/home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01/DB

4、建立1个测试用户目录：如mysql\_it1或者mysql\_it2或者mysql\_it3......

useradd mysql\_it1 -d /home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01/mysql\_it1 -m

5、建立一个输出结果目录：output

/home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01/output

6、jfrog配置及安装（可执行文件存放在10.46.178.97 /home/zyy\_work/目录下）

1）将jfrog进程传到/home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01目录下

2）vi /etc/hosts 在文件最后追加以下内容： 10.41.103.97 artnj.zte.com.cn

3）在/home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01目录下执行：

./jfrog rt c goldendb-snapshot-generic/ --url=https://artnj.zte.com.cn/artifactory/

Artifactory server ID [goldendb-snapshot-generic/]: 回车

API key (leave empty for basic authentication): 回车

User [10184561]: 10184561

Password: [Info] Encrypting password...

[Info] Done encrypting password.

4）验证是否可以下载：

./jfrog rt dl --flat=true goldendb-snapshot-generic/DB/ZXCLOUD-GOLDENDB-ALL-DBV5.1.01/last.txt

7、修改脚本update\_and\_execute\_it.sh并提交代码

1）如果步骤4的用户名不一致，则需要更改update\_and\_execute\_it.sh脚本中test\_user环境变量的名称；

2）根据branch分支名称更改update\_and\_execute\_it.sh脚本中file\_name和file\_prefix环境变量的名称；

举例如下：

file\_name = ZXCLOUD-GOLDENDB-ALL-DBV5.1.01.zip

file\_prefix = ZXCLOUD-GOLDENDB-ALL-DBV5.1.01

8、上传代码目录中的执行脚本CI/ST/update\_and\_execute\_it.sh到/home/ST\_branch/ST\_ZXCLOUD-GOLDENDB-ALL-DBV5.1.01目录下；

## 优化方法---目标实现在机器上一键测试任意版本

1. 在/home/data\_sdb/st/目录下创建“st\_时间戳”的目录。这样可以避免重复。

2）在第一步的目录中创建用户，使用固定的命名方式如：it\_时间戳。创建output目录等。

3）Jenkinsfile中调用命令，首先git克隆代码，然后切换分支，调用分支中update\_and\_execute\_it.sh脚本触发该分支的测试

4）为了防止累计太多，可以在在脚本中加上过期清理目录机制，比如清理2天前的st测试用例目录。

## 注意事项

新增的ST测试环境，除了要做好不同分支之间的时间上的隔离，也要做好mysql-test端口之间的隔离，端口的设置在ST的测试用脚本中。

# ST-COVERAGE

## coverage检查逻辑

coverage部分的jenkinsfile中代码主要调用见run\_stcodecov()函数，其检查的逻辑为：

1. 在DailyCI中跑ST测试用例；
2. 可以通过ManualCI+设置CHECK\_COVERAGE\_TEST手动触发跑全量的自动化coverage测试；
3. 所有master分支，allmaster分支均需要跑coverage。（但是目前只支持master分支，allmaster分支待支持）

Coverage的检查脚本为CI/st-coverage-test/cov\_test.sh脚本，还存在一个用于coverage

的编译脚本，本质上编译脚本的区别就是增加了2个编译选项，因此后续优化时，可以删除该编译脚本，在coverage脚本中增加对原始编译脚本的修改即可。

## Coverage环境信息

目前master分支的coverage环境部署在10.46.178.97环境上，需要跑80分钟左右。

目录为：/home/data\_sdb/coverage

后续增加allmaster分支的coverage时，可以将/home/data\_sdb/coverage目录结构重新设计为：/home/data\_sdb/coverage/master用于master分支，/home/data\_sdb/coverage/allmaster用于allmaster分支。

## 新建Coverage检查

步骤一：登录测试环境：10.46.178.97

步骤二：搭建测试目录，其中包含DB子目录存放代码，包含output子目录存放结果文件等。具体需要结合master的测试环境及coverage脚本。

步骤三：上传测试脚本；

步骤四：手动触发coverage测试；

## 注意事项

新增的coverage测试环境，除了要做好不同分支之间的时间上的隔离，也要做好mysql-test端口之间的隔离，端口的设置在ST的测试用脚本中。

# FUNCHECK

funcheck部分的jenkinsfile中代码主要调用见run\_funcheck()函数，其检查逻辑为只在VerifyCI中检查，且所有的版本分支都会检查。

该检查包含了一个检查脚本为CI/funcheck/rulecheck.sh，以及一个检查规则文件CI/funcheck/regular.txt。

# LIZARD

lizard部分的jenkinsfile中代码主要调用见run\_lizard()函数，其检查的逻辑为：

1. VerifyCI跑增量的圈复杂度检查；
2. 其余CI场景均跑全量的圈复杂度检查；
3. 所有的版本分支都会检查；

该检查包含了一个检查python脚本为CI/lizard/complexCompare.py。