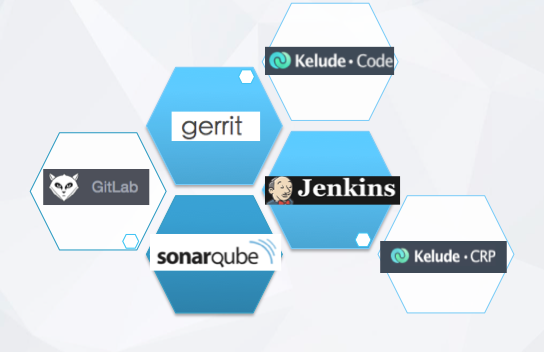
代码管理和持续集成

目前我们使用了如下的工具或者平台来实现代码管理和持续集成，满足我们团队不断发展的业务需要



这些工具各司其职，又彼此依赖，形成了一个整体的方案大图

# 图解

1.代码提交者从gerrit服务器下载或者拉取代码到本地机器

2.在本地进行编码，完成后通过 commit ->push到gerrit的代码仓库地址，gerrit上生成一条change，显示open状态，代码提交者填写代码评审人

3.gerrit自动发送邮件给代码评审人和代码提交者

4.代码评审人到gerrit服务器对代码进行评审，如果评分为+2，则显示出submit按钮，代码评审者提交代码merge到主干，如果评分-2，则收到邮件，重复1-4的过程

5.gerrit服务器将评审情况以邮件形式发送给代码提交者

6.jenkins监听gerrit服务器代码merge的情况

7.如果merge的是非master的分支，则发送邮件给代码提交者，提醒其不要忘记提交到主干;如果merge的是master的分支，则不做任何提醒

8.当merge成功以后，如果是手机端的代码，则gerrit将代码推送到gitlab

9.当merge成功以后，如果是服务端的代码，则gerrit将代码推送到alicode

10.如果submit不成功，则可按分析原因，参考解决方法<http://10.10.144.152:8090/pages/viewpage.action?pageId=21102975>，重新走评审流程

11.jenkins监听gitlab上的手机端的代码工程，如果存在merge成功事件，则自动打安装包

12.测试同学也可以上jenkins服务器，手动进行打包

13.打包成功后，将安装包发送到fir上；如果打包失败，则在jenkins会报错，不会进行安装包推送

14.推送安装到FIR成功后，jenkins发送包含安装包LOG，fir上传地址等信息的邮件给包括产品在内的全体成员

15.CRP监听alicode上的服务端的代码工程，如果存在merge成功事件，则自动进行工作流编译

16.编译成功后上传到ECS服务器；编译不成功，则在CRP上显示错误LOG，不会上传

17.jenkins监听到gerrit服务器代码merge，自动执行sonar扫描包含单元测试，如果扫描成功，则将扫描的结果发送到sonar服务器显示；如果不成功，则不会发送到sonar

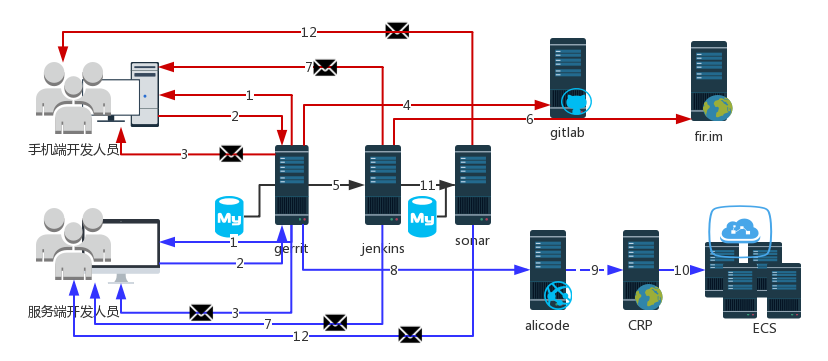
18.开发同学也可以上jenkins服务器，手动执行扫描JOB，如果扫描成功，则将扫描的结果发送到sonar服务器显示；如果不成功，则不会发送到sonar

19.sonar服务器扫描完成后，将扫描结果以邮件形式发送给代码最后提交者

20.代码最后提交者根据邮件内容，重新下载或者拉取代码到本地机器进行开发

# 以下分角色说明在整个流程中的职责和具体操作

## 开发人员



图解：

1.代码提交者从gerrit服务器下载或者拉取代码到本地机器

2.在本地进行编码，完成后通过 commit ->push到gerrit的代码仓库地址，gerrit上生成一条change，显示open状态，代码提交者填写代码评审人

3.gerrit自动发送邮件给代码评审人和代码提交者

4.当merge成功以后，如果是手机端的代码，则gerrit将代码推送到gitlab

5.jenkins监听gerrit服务器代码merge的情况

6.当jenkins监听到手机端android代码有merge成功操作，则执行打包JOB，成功后将安装包上传到FIR；当jenkins监听到手机端IOS的某个子模块工程有代码merge成功操作；则调用IOS主工程打包JOB进行编译打包，成功后将安装包上传到FIR；

7.fir上传地址等信息的邮件给包括产品在内的全体成员

8.服务端代码评审通过，在gerrit上merge成功后，代码推送到alicode

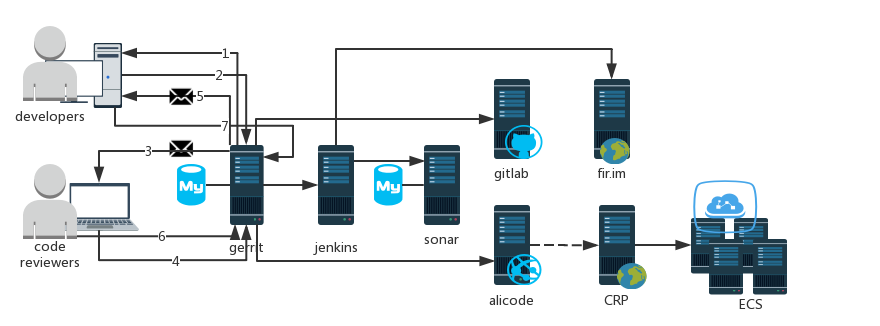
9.CRP监听到alicode上的push event，对应的工作流进行编译

10.编译成功后上传到ECS上部署；编译失败则CRP上显示错误LOG

11.Jenkins监听到gerrit服务器代码merge，自动执行sonar扫描包含单元测试，如果扫描成功，则将扫描的结果发送到sonar服务器显示；如果不成功，则不会发送到sonar

12.sonar服务器扫描完成后，将扫描结果以邮件形式发送给代码最后提交者

## 2.评审人员



图解：

1.代码提交者从gerrit服务器下载或者拉取代码到本地机器

2.在本地进行编码，完成后通过 commit ->push到gerrit的代码仓库地址，gerrit上生成一条change，显示open状态，代码提交者填写代码评审人

3.gerrit根据填写的代码评审人发送提醒评审邮件给代码评审人

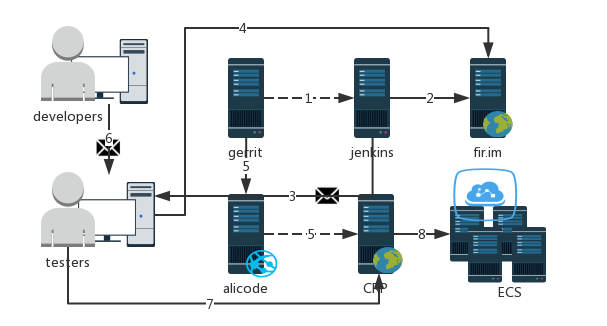
4.代码评审人到gerrit对change进行评审，分布评论，打分等

5.根据评分进行不同的操作，如果评-2将gerrit发送邮件，代码提交者重复1，2动作，再次从3开始执行

6.如果评+2将gerrit发送邮件，代码评审者可以点击submit按键，代码提交到gerrit

7.如果评论为-1，0，+1的话，不足以产生决定性的评审结果，重复4或者代码提交者重复1，2加补丁集再进行评审

## 3.测试人员



图解：

1.手机端代码评审通过，在gerrit上merge成功后，jenkins监听到该事件，调用jenkins编译打包job

2.打包成功后，会将打好的包会发送到FIR

 3.打包成功后，会发邮件给测试人员；打包失败则jenkins显示错误信息，不发送邮件

4.测试根据邮件选择要测试的安装包，到FIR上下载安装后进行测试

5.CRP监听到alicode上的push event，对应的工作流进行编译

6.开发人员发邮件申请部署服务端的代码

7.测试人员根据开发发送的部署申请邮件，可以到CRP上进行手动编译和部署

8.编译成功后上传到ECS上部署；编译失败则CRP上显示错误LOG

## 4.配管人员

图解：

1.配置人员到gerrit上进行代码工程和权限的初始化操作

2.每个迭代的第二个周周三下午全员发邮件，提醒本次迭代封板和写权限锁定时间，请尽快提交代码。在每个迭代的第二个周周五下午二点，关闭gerrit代码提交权限，并基于TAG结点拉分支

3.当merge成功以后，如果是手机端的代码，则gerrit将代码推送到gitlab

4.当merge成功以后，如果是服务端的代码，则gerrit将代码推送到alicode

5.在每个迭代的第二个周周五下午二点，到gitlab上基于Master打Tag，升级版本的第二位数字，第三位数字置0

6.在每个迭代的第二个周周五下午二点，到alicode上基于Master打Tag，升级版本的第二位数字，第三位数字置0

7.jenkins监听gerrit服务器代码merge的情况

8.配置人员到jenkins上进行手机端的打包JOB配置和代码静态扫描JOB配置

9.JOB执行后发送邮件，抄送配置人员

10.jenkins上JOB执行后将结果推送到sonar展现

11.配置人员到sonar上对工程的展现维度，规则集，阈值等进行设置