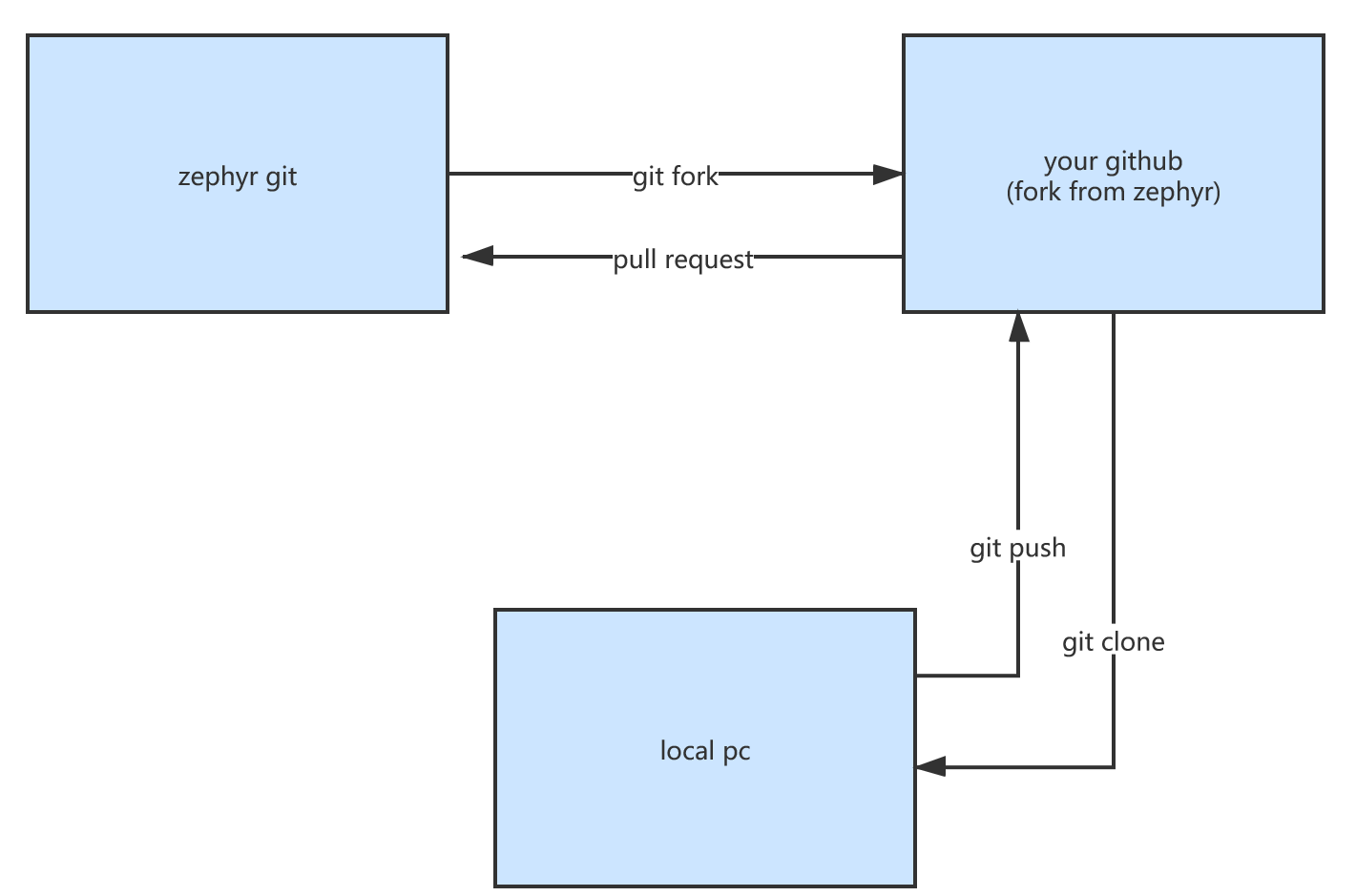
**Zephyr之github使用指南**

本文目的为首次做基于github开源项目的同事做参考，使其知道怎么上传代码到远端的zephyr 的github仓库。本文假设读者已经知道一些基本的git的操作命令。大致分为两个部分，第一部分是commit的格式需求；第二部分是关于push需要注意的地方。

下图大致描述了基于github的开发的工作流程：



可以看到，上图包括了三个模块，左上角的模块是zephyr的git仓库；右上交的模块是你自己的github的zephyr仓库；下面的模块是你本机的代码。在每次push代码之前需要运行gitlint做本地的pull request的检查；然后再运行相关sanitytest测试自己修改的模块。

工作流程如下：

1. 首先你会进入zephyr的git仓库，然后fork zephyr的代码到自己的github。
2. 你在本机用git clone从你自己的github的zephyr代码下载到本地。
3. 你在本地修改完相关代码之后，用git push 到你的github的zephyr的仓库。
4. 在你自己的github的zephyr的仓库create pull request。

在zephyr的仓库中，关于commit和push这两个操作有些地方需要注意。

* **Git commit -s的格式注意**

Commit的格式有三个地方需要注意，所谓的格式，是指在修改完代码，git add，然后运行git commit -s的时候，会出现一个窗口，需要填写一些相关信息，而zephyr对这些信息有一定的格式要求。在本地可以通过gitlint来检查这些格式是否符合要求。

1. **概要的格式**

概要在第一行，有两块构成：module：description

Module是说明你修改的是什么模块；description是用于大概描述此次修改的目的。比如我修改的是tests的模块，然后这次修改的目的是修复bug21812，信息如下：

tests: modify bug 21812

上面的提交的信息还是有问题，在开源项目中，代码的提交，不宜用“修改bugXX”这类 描述，应该是言简意赅的描述修改的目的。

1. **提交的详细描述**

详细描述在第三行，需要详细说明修改的理由，这里需要注意的是每行不能太长，每一行的长度不要超过窗口里面的其它文字，否则会 gitlint的时候会失败。比如：

In the function irq\_connect\_dynamic, it will check if the irq is

disable. If the irq is enable ,it will assert.So before invoking

the function irq\_connect\_dynamic, it must disable the irq.

1. **Signed-off-by的格式**

最后的需要注意的格式是：Signed-off-by: box zheng [box.zheng@intel.com](mailto:box.zheng@intel.com)

这行是默认生成的，是根据你通过

**git config --global user.name "David Developer"**

**git config --global user.email** [**david.developer@company.com**](mailto:david.developer@company.com)

需要注意的是user.name必须shi和email是对应的，比如说：

git config --global user.name "zhangsanx"

git config --global user.email [zhang.san@company.com](mailto:zhang.san@company.com)

如果是上面的格式的话，那gitlint就会检查失败。

* **Git push的相关问题**

1. **Github远程仓库的说明**

在这里，需要了解一些git远程仓库的配置信息。

在git里面是用git remote -v来查看远程仓库的信息。比如我自己的信息如下：

origin https://github.com/activebug202/zephyr (fetch)

origin https://github.com/activebug202/zephyr (push)

upstream https://github.com/zephyrproject-rtos/zephyr (fetch)

upstream https://github.com/zephyrproject-rtos/zephyr (push)

其中origin是指向了我的github的zephyr的仓库地址，而upstream是指向了zephyr的仓库地址。不管是origin和upstream都只是名字，你可以随意修改它的名字，zhangsan，lishi都可以。

在git里面有些默认的操作，比如如果你的代码是从你的github的地址clone下来的，那默认的origin就是指向你的github的地址；如果你的代码是通过west init zephyrproject下载的，那默认的origin就是指向zephyr的仓库地址。

这里需要注意的是，如果你是通过west下载的zephyr的代码，你需要新增加一个remote的名字，以指向你自己的github的zephyr项目地址。如下：

git remote add XXX https://github.com/<your github id>/zephyr

上面命令的XXX可以随意指定。并且可以通过git remote -v来查看。

这个XXX指向你自己的github的zephyr的地址很重要，因为如果你想将你的代码提交服务器的话，首先是将修改push到你自己的github的zephyr。

1. **Push代码的说明**

Git push的格式如下：

git push <远程主机名> <本地分支名>  <远程分支名>

远程主机名就是上面git remote里面提到的XXX，需要指向自己的github的zephyr的仓库。本地分支名，如果你没有git checkout -b，那默认就是master分支。远程分支名，你可以随意的命名，比如debug\_xxx。

Push成功之后，在你的github的zephyr的仓库可以看到新的debug\_xxx的分支。Push的github的密码输入，因为zephyr，github的账户都是双重认真，所以当你push代码到你自己的github的时候，不再是输入你自己的github的登录密码，具体输入什么密码，请参考：

<https://www.cnblogs.com/smh2015/p/11444415.html>

1. **如果需要修改已经提交的pull request怎么办**

Modify bug/comment之后，如果有修改代码，就git add，然后git commit --amend。

如果没有修改代码，只是想修改提交的comment，就直接git commit –amend。

然后使用git push 到原来创建pull request的时候对应的自己github的分支，然后github就自动把改动merge到原来的那个pull request

1. **如果在本地修改了文件，暂时还不想commit，怎么保存现场，切换到其它分支**

使用git stash 保持现场，如果有新增加的文件，需要git stash -u

然后在别的分支做完工作后，再切回到这个分支，需要恢复以前的现场：

首先 git stash list 查看存储了几个stash

如果是最新的一个stash需要恢复，就用git stash pop

如果不是，那就git stash apply stash@{XX} 。这个xx是stash的序列号。

1. **git repo gerrit之间的关系**
2. **git cherrypick怎么使用**
3. **git pull和git fetch的区别**