

Git管理项目开发准备 和开发使用Git常规作业流程

2019/12/30 David.

目录

contents

PART 1

Gitlab管理
组别架构

PART 2

Git管理项目
开发准备

PART 3

开发使用Git
常规操作

PART

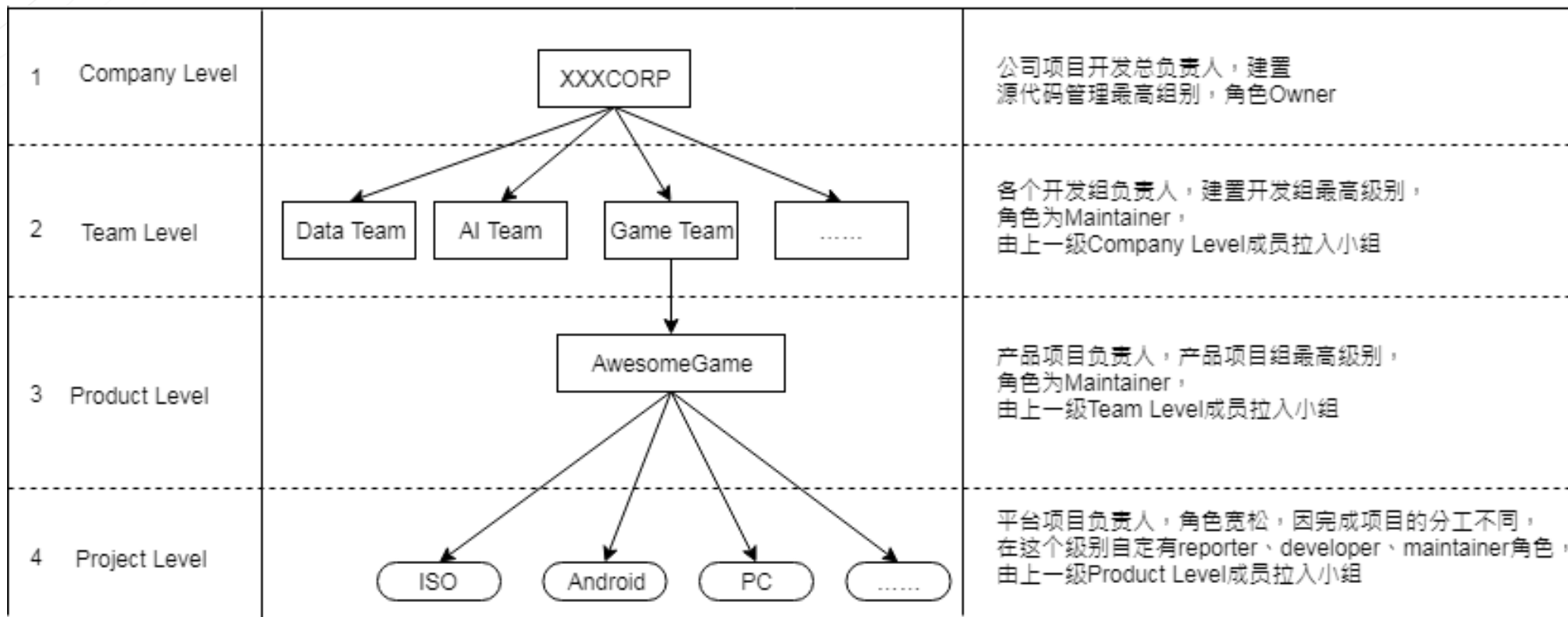
Gitlab管理组别架构

01

- Gitlab管理组别架构

Gitlab管理组别架构

- Gitlab管理组别架构
 - 创建组别(Group)层级如图



Gitlab管理组别架构

- 图示说明:

- 前提:

- 最重要的初始点, 需要给所有需要使用到的人自行注册一个gitlab账号
 - 项目开发总负责人需要管理所有该公司开发的项目
 - 所有的项目都放在该负责人创建的Group组织下, 自顶向下。上面的可以看到下面的, 下面的不能看到上面的。
 - 公司内部项目, 存放在内部的Gitlab server, 但是个人开发者如果把项目放到公司内部自己账号下, 并不影响整理项目代码管理。但政策面, 最好就规定, 不要把私有项目放到公司的服务器上, 也避免资源浪费。

Gitlab管理组别架构

- 第一层

- 项目开发总负责人创建公司群组，作为第一阶级Group，可以查看到下属SubGroup的所有内容，因为是创建者，角色为owner；
- 如果有助理等协助管理，则不会去异动到项目的内容，只可能会审查，然后回报，所以也需要在第一阶级Group中，可以查看所以下属subGroup的内容，但角色只需要到reporter。

- 第二层

- 项目开发总负责人计划按照目前的写代码的实际组别或者公司现有架构分组，分为几个第二阶级Group，如图示，并带入其group的实际负责人的账号，指定角色为Maintainer。
- 至此，项目开发总负责人要做的已经完成，第3、4阶级的创建，就有第二阶级的负责人（gitlab账号）去执行。

Gitlab管理组别架构

- 第三层

- 以Game Team为例子，如果开发组负责人认为， Game Team的所有项目，都是全部Game Team的成员都可以查看的，那么就可以直接在第二级Game Team拉入所有的Game Team的成员的gitlab账号。
- 如果他觉得他的所有项目其它组例如Data Team也可以看，也可以拉入Data Team相关人员，这需要按照实际作业规范执行
- 后续， Game Team有产品项目，例如一个叫Awesome Game的产品，可以专门创建一个AwesomeGame产品项目群组，拉入相关人员的gitlab账号。

Gitlab管理组别架构

- 第四层

- 上面3层，都是Group， AwesomeGame产品项目群组分为不同的例如平台，产品项目实现代码不同，这就是gitlab管理的最小单位的project，按照实际需求创建project。
- 如果在第二阶级或者第三阶级就拉入了所有Game Team组成员，这里就不用再拉成员了。如果没有，则需要再拉入。
- 如果是指定项目分来拉入，那么只有指定project的成员可以看到自己的project，而不能看到别人的project。
- 但是第一阶级、第二阶级或第三阶级还有其它gitlab账号，他们就可以此处所有的project。
- 如果AwesomeGame这个产品下属还需要分类为例如Server、Client等，按照实际情况可以继续往下分产品平台组，放不同模块project。
- 注意，不是分得越细越好，按照实际需求灵活处理。

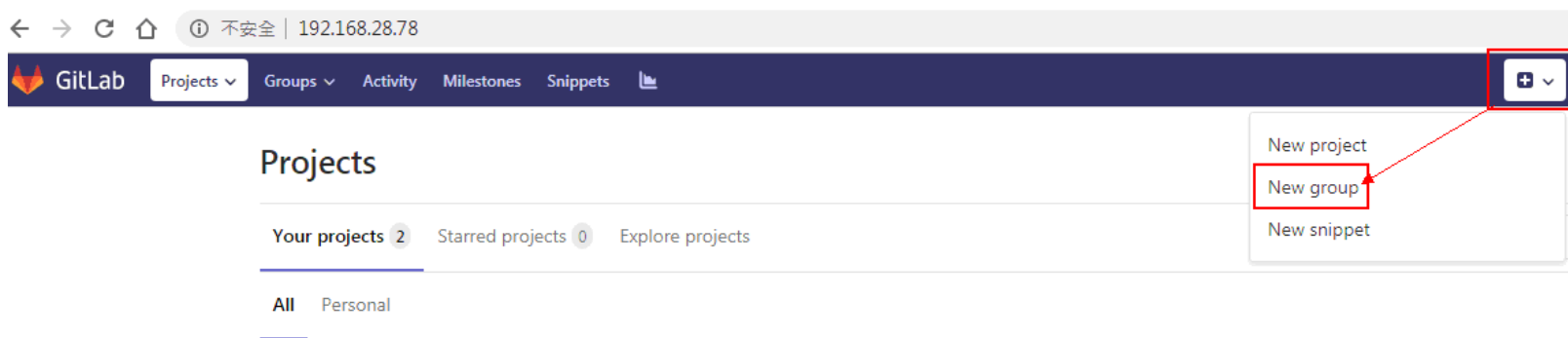
PART 02

Git管理项目开发准备

- Group准备
- 项目初始化准备
- 小组成员开发就绪
- 后续步骤
- Git补充的一些说明

Group准备

- 产品项目负责人（前述第三阶级或实际要上传代码的上一级群组）在gitlab上创建指定项目小组，并拉入项目成员git用户。
 - 此处示例是项目小组成员可查看该项目的所有内容
- 1、创建group:



Group准备

- 一般是私有group

New group

Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects.

Groups can also be nested by creating subgroups.

Projects that belong to a group are prefixed with the group namespace. Existing projects may be moved into a group.

Group name

Group URL

Group description (optional)

Group avatar

Choose File ... No file chosen

The maximum file size allowed is 200KB.

Visibility level

Who will be able to see this group? [View the documentation](#)

☒ Private

The group and its projects can only be viewed by members.

☐ Internal

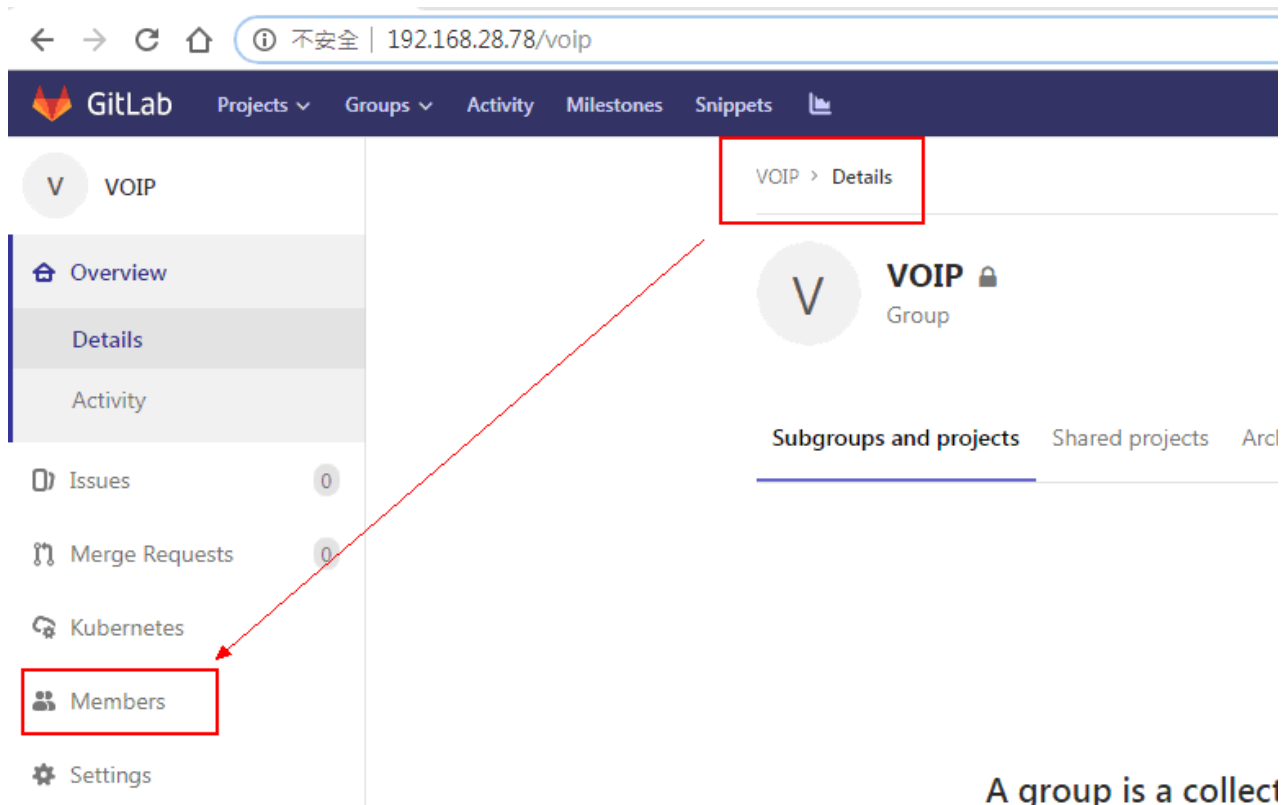
The group and any internal projects can be viewed by any logged in user.

☐ Public

The group and any public projects can be viewed without any authentication.

Group准备

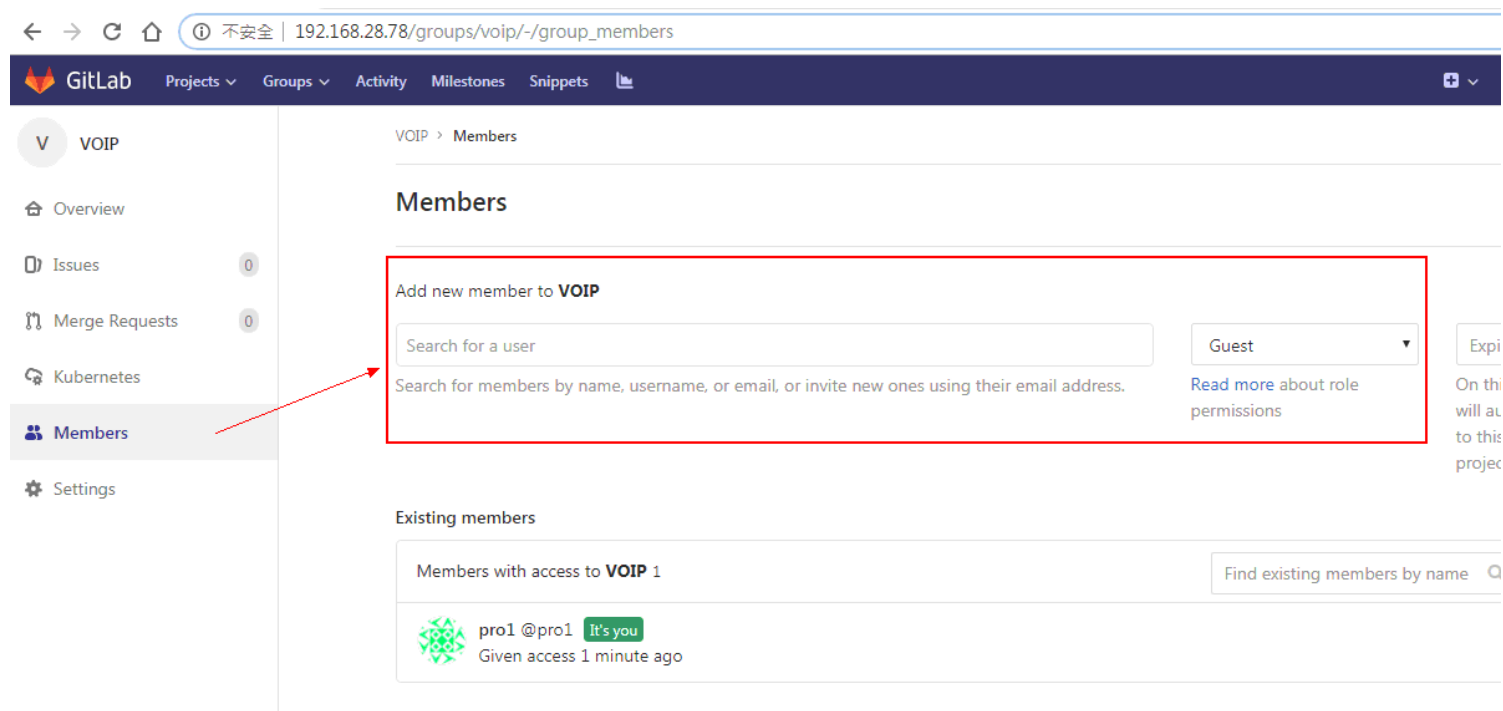
- 2、添加group成员



A group is a collect

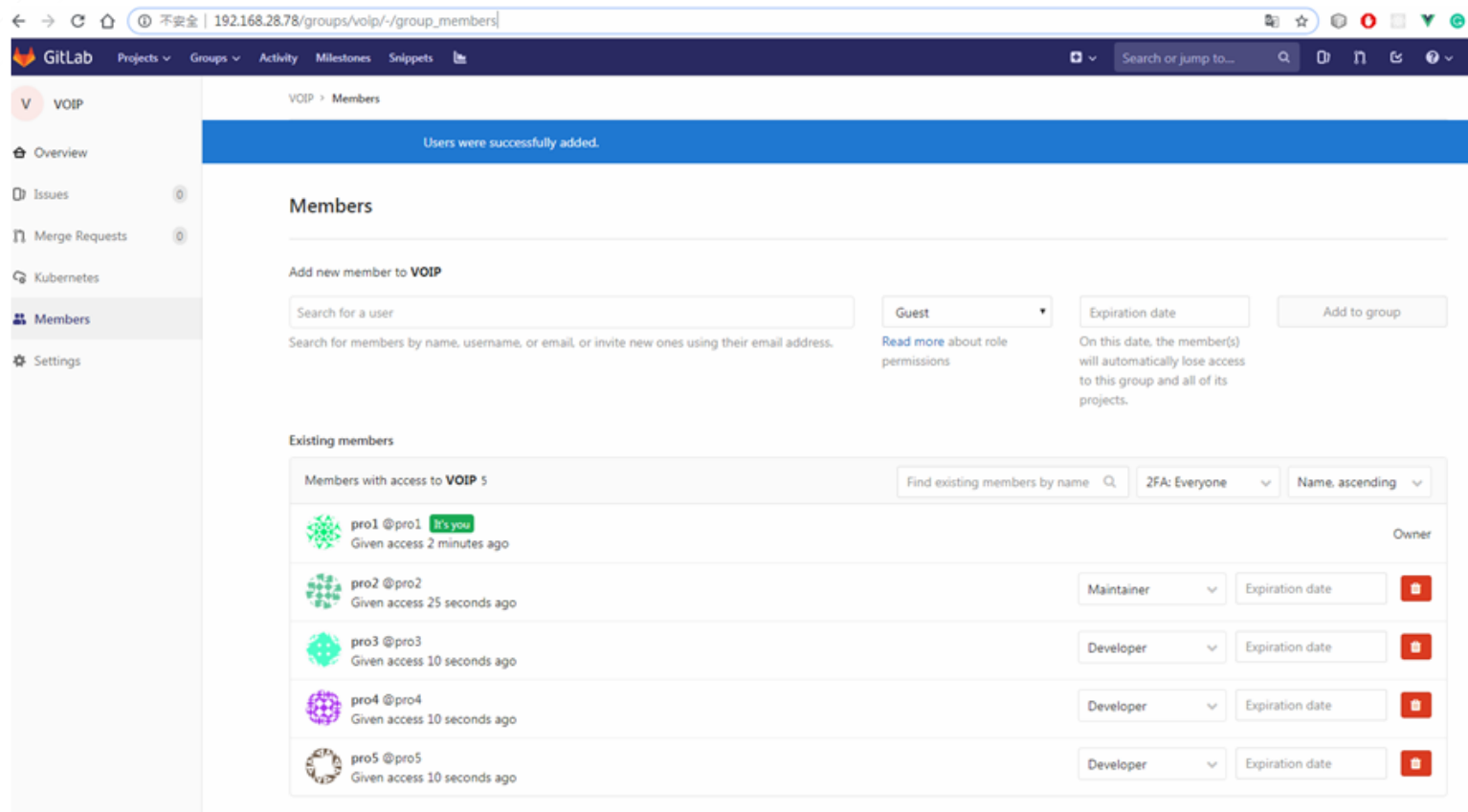
Group准备

- 添加成员注意分配指定角色（role）
 - 如果要限制项目成员不允许合并推送到被保护的分支，则角色只有到developer。



Group准备

- 添加成员后如下：



项目初始化准备

- 项目负责人在gitlab上创建指定项目。

VOIP > Details



VOIP
Group



New project



Subgroups and projects

Shared projects

Archived projects

Search by name

Last created



New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

Information about additional Pages templates and how to install them can be found in our [Pages getting started guide](#).

Tip: You can also create a project from the command line. [Show command](#)



Blank project

Create from template

Project name

VOIP-BackendSystem

Project URL

http://192.168.28.78/ voip

Project slug

voip-backendsystem

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Visibility Level ?



Private

Project access must be granted explicitly to each user.

项目初始化准备

- 注意只是创建了一个项目，是空的，先不要做其它作业。

VOIP > VOIP-BackendSystem > Details

Project 'VOIP-BackendSystem' was successfully created.



VOIP-BackendSystem 🔒

Project ID: 13



Star

0



 Add license  0 Commits  0 Branches  0 Tags  0 Bytes Files

The repository for this project is empty

If you already have files you can push them using the [command line instructions](#) below.

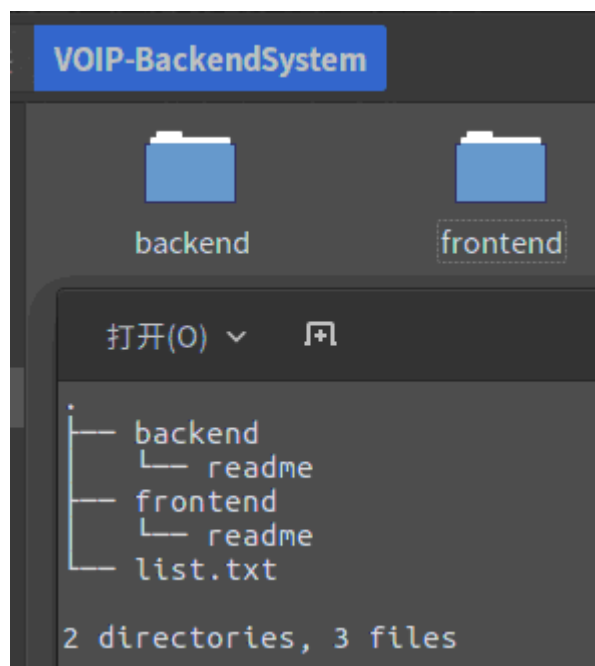
Note that the master branch is automatically protected. [Learn more about protected branches](#)

You can automatically build and test your application if you [enable Auto DevOps](#) for this project. You can automatically deploy it as well, if you [add a Kubernetes cluster](#).

Otherwise it is recommended you start with one of the options below.

项目初始化准备

- 项目负责人在本地构建一个项目初始化项目
 - 注意，空白活页夹不会被管理，可以添加一个readme



项目初始化准备

- 初始化项目推到gitlab server (常称远程origin)
 - 负责人创建好项目结构之后，按照gitlab上空项目下方的提示，将本地初始化项目git化，并推送到gitlab server

Existing folder

```
cd existing_folder
git init
git remote add origin http://192.168.28.78/voip/voip-backendsystem.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

项目初始化准备

- 过程如下图:

```
flos@flos-VirtualBox: ~/VOIP-BackendSystem
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
flos@flos-VirtualBox:~/VOIP-BackendSystem$ git init
初始化空的 Git 仓库于 /home/flos/VOIP-BackendSystem/.git/
flos@flos-VirtualBox:~/VOIP-BackendSystem$ git add .
flos@flos-VirtualBox:~/VOIP-BackendSystem$ git commit -m 'VOIP後臺系統初始化項目結構'
[master (根提交) 5f7d873] VOIP後臺系統初始化項目結構
 2 files changed, 4 insertions(+)
 create mode 100644 backend/readme
 create mode 100644 frontend/readme
flos@flos-VirtualBox:~/VOIP-BackendSystem$ git remote add origin http://192.168.28.78/voip/voip-backendsystem.git
flos@flos-VirtualBox:~/VOIP-BackendSystem$ git push -u origin master
Username for 'http://192.168.28.78': pro1
Password for 'http://pro1@192.168.28.78':
对象计数中: 6, 完成.
Delta compression using up to 4 threads.
压缩对象中: 100% (2/2), 完成.
写入对象中: 100% (6/6), 472 bytes | 0 bytes/s, 完成.
Total 6 (delta 0), reused 0 (delta 0)
To http://192.168.28.78/voip/voip-backendsystem.git
 * [new branch]      master -> master
分支 master 设置为跟踪来自 origin 的远程分支 master。
flos@flos-VirtualBox:~/VOIP-BackendSystem$
```

项目初始化准备

- 推送完成，在远程就可以看到推送的项目

VOIP > VOIP-BackendSystem > Details



VOIP-BackendSystem 🔒

Project ID: 13



Star

0



Fork

0

Clone ▾

Add license

1 Commit

1 Branch

0 Tags

113 KB Files

master ▾

voip-backendsystem /



History

Find file

Web IDE



VOIP後臺系統初始化項目結構

keeper of arcane lore authored 1 minute ago



5f7d873d



Add README

Add CHANGELOG

Add CONTRIBUTING

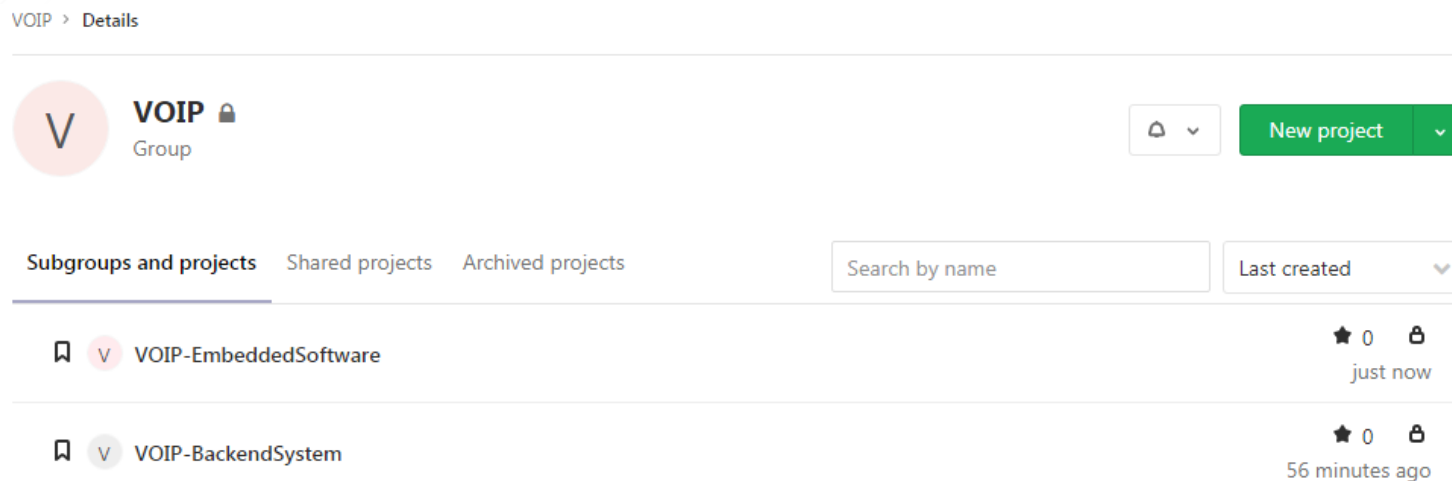
Auto DevOps enabled

Add Kubernetes cluster

Name	Last commit	Last update
backend	VOIP後臺系統初始化項目結構	1 minute ago
frontend	VOIP後臺系統初始化項目結構	1 minute ago

项目初始化准备

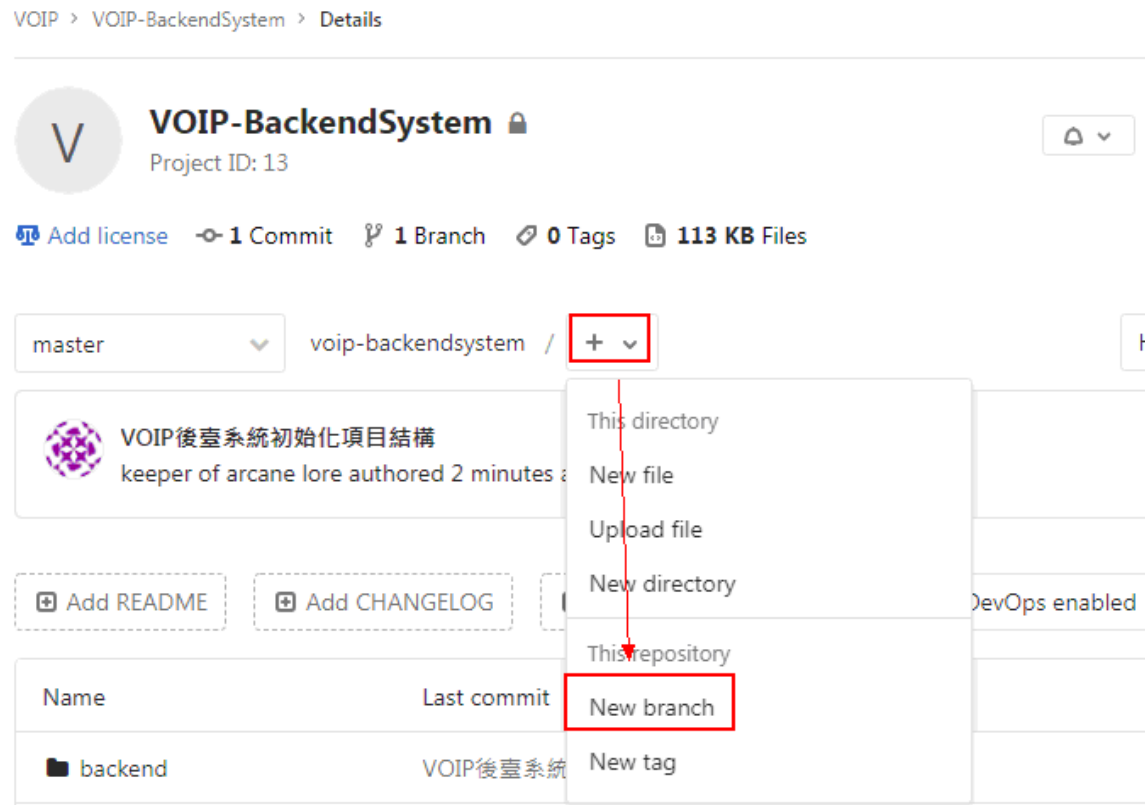
- 如果这个产品项目有很多个项目，再创建其它的即可



注意自顶向下，如果push了多个项目，在远程目前是无法合并成一个项目的，之前的文章有说明在本地合并多个git管理项目为一个，可再回顾。

项目初始化准备

- 创建分支保护
 - 创建分支



项目初始化准备

VOIP > VOIP-BackendSystem > New Branch

New Branch

Branch name

Create from

Existing branch name, tag, or commit SHA

Create branch



VOIP-BackendSystem

Project ID: 13

Add license 1 Commit 2 Branches 0 Tags 113 KB Files

master

voip-backendsystem /

+ -

Switch branch/tag

Search branches and tags



Branches

✓ master

develop

项目初始化准备

- 设定分支保护

Protected Branches

Keep stable branches secure and force developers to use merge requests.

By default, protected branches are designed to:

- prevent their creation, if not already created, from everybody except Maintainers
- prevent pushes from everybody except Maintainers
- prevent **anyone** from force pushing to the branch
- prevent **anyone** from deleting the branch

Read more about [protected branches](#) and [project permissions](#).

Protect a branch

Branch:

Wildcards such as `*-stable` or `production/*` are supported

Allowed to merge:

Allowed to push:

Protect

Protected branch (1)	Last commit	Allowed to merge	Allowed to push	
master <small>default</small>	5f7d873d 5 minutes ago	<input type="text" value="Maintainers"/>	<input type="text" value="Maintainers"/>	Unprotect

项目初始化准备

- 如果按照一般原则，master分支不允许后续直接推送，则需要修改权限

Protected branch (2)	Last commit	Allowed to merge	Allowed to push	
develop	5f7d873d 5 minutes ago	Maintainers ▼	No one ▼	Unprotect
master default	5f7d873d 5 minutes ago	Maintainers ▼	Maintainers ▼	Unprotect

小组成员开发就绪

- 项目小组成员clone下负责人上传的初始化项目，开始各自功能的开发，并使用git管理此本地项目
 - 复制地址

VOIP > VOIP-BackendSystem > Details

You pushed to `develop` at `VOIP / VOIP-BackendSystem` 3 minutes ago

Create merge request



VOIP-BackendSystem 🔒

Project ID: 13

[Add license](#)

[1 Commit](#)

[2 Branches](#)

[0 Tags](#)

[113 KB Files](#)

master



voip-backendsystem



+



VOIP後臺系統初始化項目結構

keeper of arcane lore authored 7 minutes ago



Star

0



Fork

0

Clone



Clone with SSH

git@192.168.28.78:voip/voip-|



Clone with HTTP

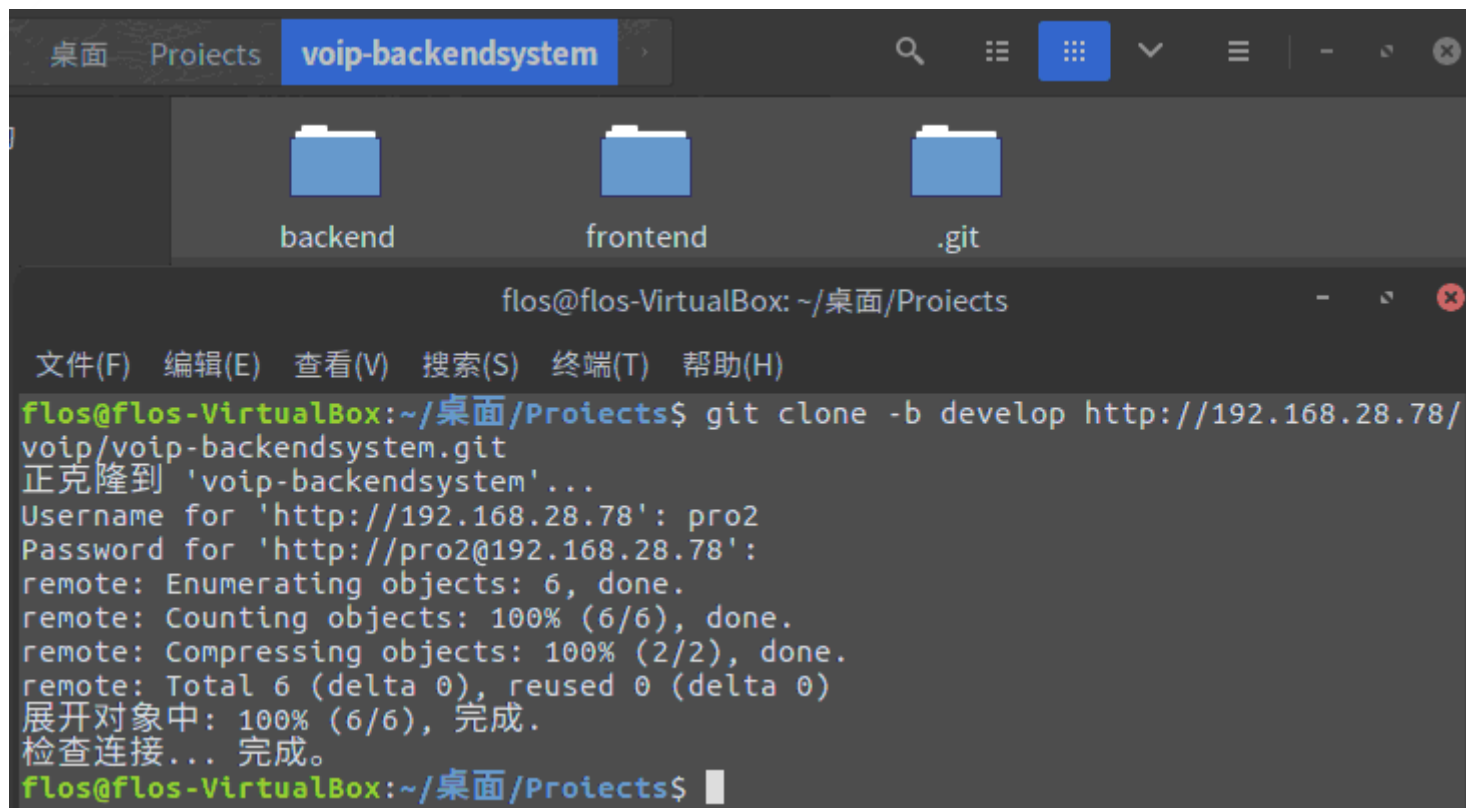
http://192.168.28.78/voip/vo



Copy URL to clipboard

小组成员开发就绪

- clone到本地



The screenshot shows a file manager window titled '桌面 Projects voip-backendsystem' with three folders: 'backend', 'frontend', and '.git'. Below it is a terminal window titled 'flos@flos-VirtualBox: ~/桌面/Proiects'. The terminal displays the command 'git clone -b develop http://192.168.28.78/voip/voip-backendsystem.git' and its output, including authentication prompts and progress bars for cloning and checking out the repository.

```
flos@flos-VirtualBox: ~/桌面/Proiects

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

flos@flos-VirtualBox:~/桌面/Proiects$ git clone -b develop http://192.168.28.78/voip/voip-backendsystem.git
正克隆到 'voip-backendsystem'...
Username for 'http://192.168.28.78': pro2
Password for 'http://pro2@192.168.28.78':
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
展开对象中: 100% (6/6), 完成。
检查连接... 完成。
flos@flos-VirtualBox:~/桌面/Proiects$
```

后续步骤

- 本地git管理时注意一般规范（参看《git入门指南》）
 - 本地开发依然按照5大分支分类作业
 - 按时commit，并填写有意义的注释
 - 功能完成将该功能分支推送（push）到gitlab指定项目
- 推送到远程之后，在远程的作业：
 - 在gitlab创建合并请求
 - 项目负责人再将功能分支合并到develop分支
- 此外，合并之后，小组成员有需要，再拉取（pull）到本地仓库，已同步最新代码继续开发

后续步骤

- 对已有分散项目和合并和初始上传
 - 适用场景：
 - 已有进行的项目开发，还未使用版控；
 - 没有一份集中的程序，分散在各个主机；
 -
 - 建议做法：
 - 先项目负责人将目前已有的项目内容整合成一份，保留所有必要内容，并初始化为git仓库；
 - 在gitlab指定群组位置创建对应project，并上传文件整合的git仓库；
 - 各个开发人员clone该project继续开发，不再使用之前的材料。

Git补充的一些说明

- 1. gitlab不存在用于版本的用户，可以随意查看到所有gitlab服务器上的所有项目，所以有人员需要查看所有的项目只有2个办法：
 - 1、所有的项目成员中都拉上该成员。
 - 2、所有的项目都放在该成员创建的组或子组下面。
- 2. git管理的对象是项目，但其实就是一个个活页夹和活页夹下面的档案。所以一个项目可能有很多的项目，有很多平级的活页夹，所以项目管理最好建组管理。
- 3. gitlab是项目管理的服务器，代码托管的平台，管理的是person (group) 和project。每个人都可以创建project或者group，如果设为私有，那么只有指定人员 (group/project member) 才能查看，其它人查看不了。

PART

开发使用Git常规操作

03

- gitlab协作开发模式说明
- 开发者日常Git操作

gitlab协作开发模式说明

- 远程分支

- 指定开发人员创建项目的基本结构，并推送到origin的master分支
- 使用master分支创建一个新的分支dev，用于合并开发人员的code
- 开发人员(例如david/paul)以dev分支为基础，创建单独的功能分支download和分支upload，用于同步本地本机的code（非必要）
 - 本地仓库和远程仓库管理之后，如果本地推送的分支远程没有，远程会默认创建同名分支

gitlab协作开发模式说明

- 本地分支
 - 直接clone远程的dev分支作为本地主分支
 - 依据本地的dev分支创建本地开发的功能分支 (upload/download)
 - 在本地功能分支进行coding, 完成之后提交, 并直接push到远程的对应分支
 - 在远程将功能分支的code合并到dev分支,
 - 如果该功能分支已完成所有功能, 则删除
 - 如果该功能分支还有未完成内容, 则不删除 (不建议未完成功能就合并到dev分支)
 - 本地dev分支pull远程的dev分支
 - 将pull后的本地dev分支创建新的功能分支 (login), 继续开发

gitlab协作开发模式说明

- 操作说明

- 只需要日常重复如下操作

- 在功能分支进行coding
 - 完成某一部分功能之后或者到指定时间便commit到本地仓库
 - 根据需求将本地的code push到远程开发分支
 - 需求例如:
 - 1功能开发完成需要合并到主分支测试
 - 2 upload开发的功能download需要使用, 所以要将upload分支push到远程dev分支, 合并后供download分支 pull下来使用
 - 将远程功能分支合并到远程dev分支
 - 一般在本地分支推送到远程分支之后, 立刻就执行合并到dev分支的动作

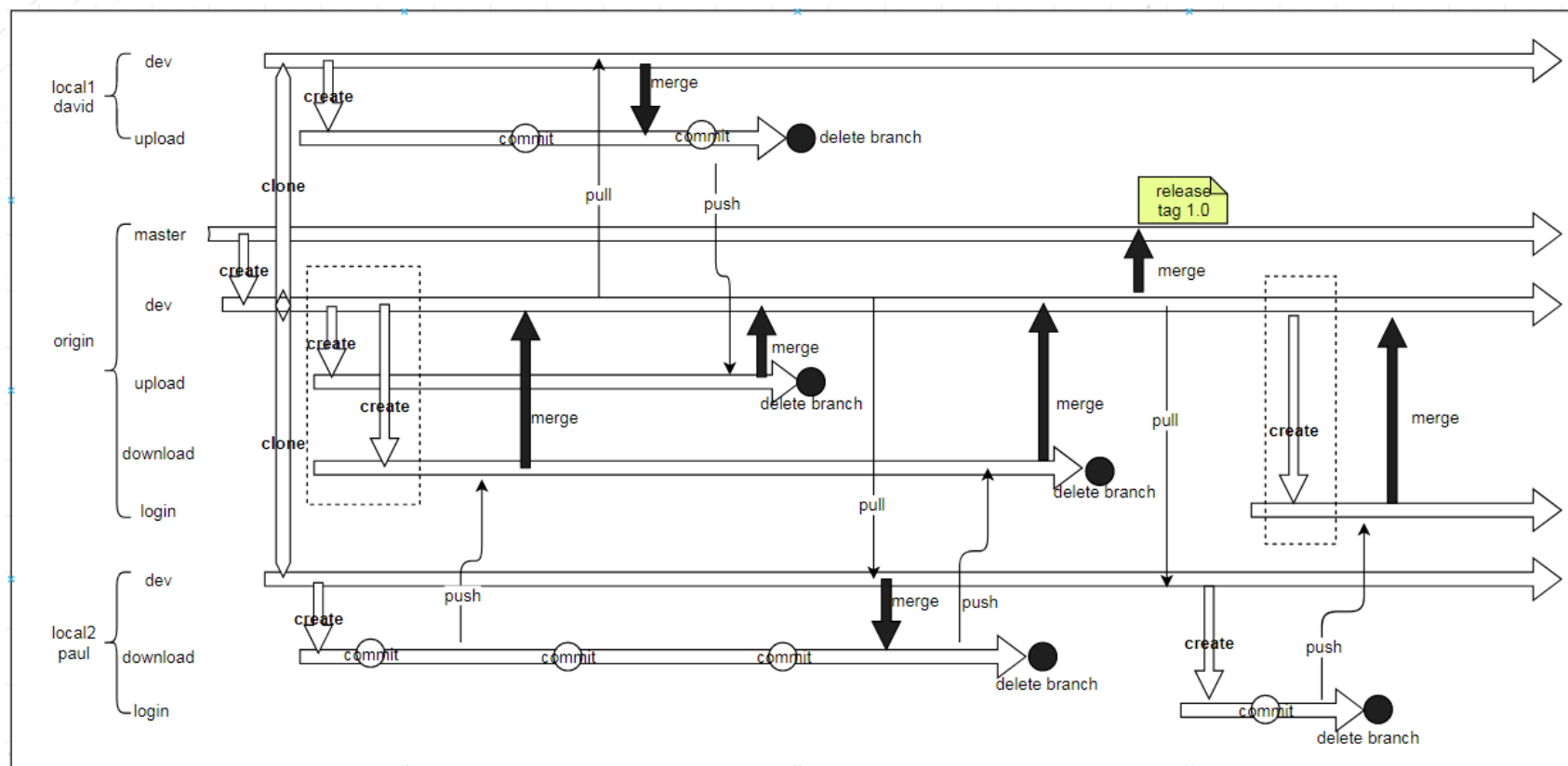
gitlab协作开发模式说明

- (续上一页)

- 本地dev分支同步pull远程dev分支
 - 一般待远程功能分支已经合并到了dev分支即可pull
- 旧功能完成，新功能开始，则在dev分支基础create新的功能分支
- 到一个阶段完成之后，可将dev分支merge到master分支，并标记tag
- 如果远程dev分支有被其它分支修改过，也可以先pull到本机dev分支，在合并到功能分支，如果遇到冲突，可以先解决。
- 合并完之后再push到远程dev分支，供其它分支同步

gitlab协作开发模式说明

- 图示参考



开发者日常Git操作

- 直接使用Git指令（假设Git已安装）
 - 1 本机提交修改

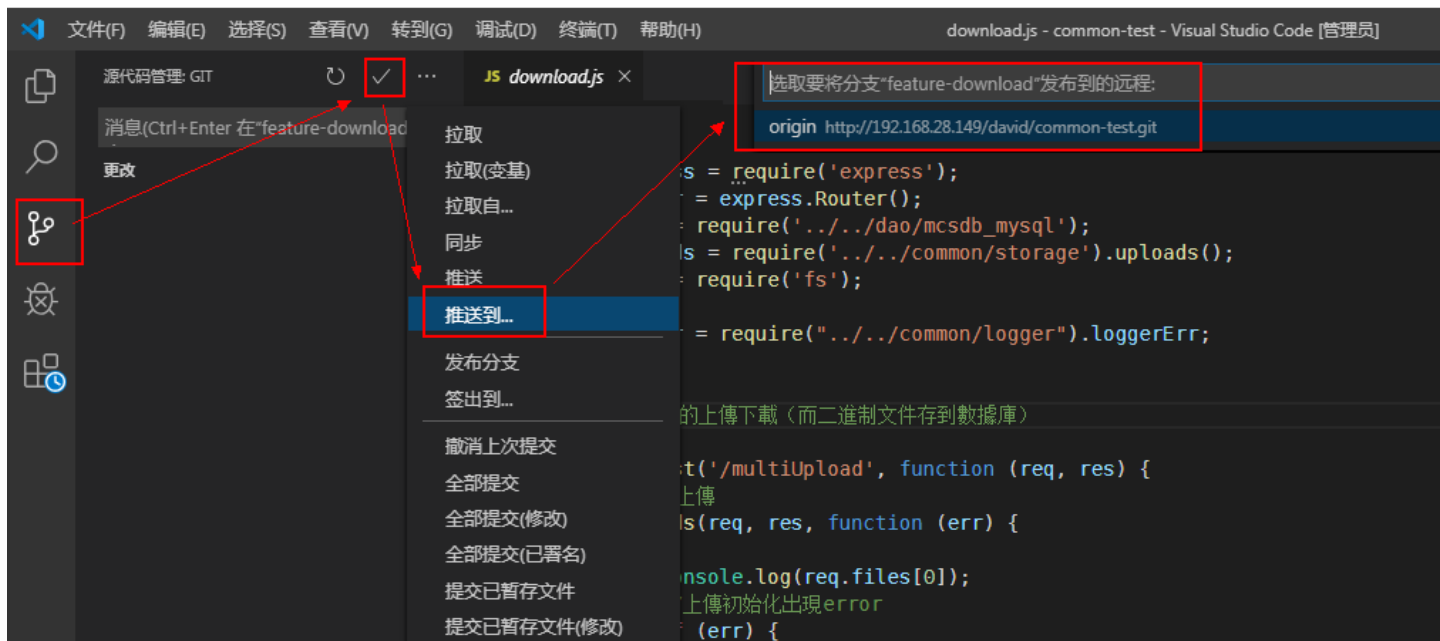
```
dav... MINGW64 /d/da.../Desktop/Gitlab測試/GitTest/common-test (feature-download)
$ git add .

d... MINGW64 /d/da.../Desktop/Gitlab測試/GitTest/common-test (feature-download)
$ git commit -m '新增download功能結構'
[feature-download f35a4cd] 新增download功能結構
1 file changed, 90 insertions(+)
create mode 100644 download.js

dav... MINGW64 /d/da.../Desktop/Gitlab測試/GitTest/common-test (feature-download)
$
```

开发者日常Git操作

- 2功能分支完功能后合并到dev分支
 - 推送到远程





开发者日常Git操作




- 创建一个新的合并请求






david > common-test > 详情

您推送了 `feature-download` 于 `david / common-test` 51 秒前

[创建合并请求](#)




 **common-test** 
项目ID : 69

  星标 0  派生 0 [Clone](#)


 添加许可证  1 Commit  2 Branch  0 Tag  92 KB Files

master common-test / +

历史 查找文件 Web IDE

 Initial commit
由 david 提交于 13 分钟前  3b0c583a 

[Add README](#) [Add CHANGELOG](#) [Add CONTRIBUTING](#) [启用Auto DevOps](#) [添加 Kubernetes 集群](#)

名称	最后提交	最后更新
 index.js	Initial commit	13 分钟前

开发者日常Git操作

- 注意是将功能分支合并到dev分支
 - 功能完成，合并后建议都删掉原功能分支

From **feature-download** into **dev** [Change branches](#)

标题

Start the title with **WIP:** to prevent a **Work In Progress** merge request from being merged before it's ready.
Add [description templates](#) to help your contributors communicate effectively!

Description

Write Preview

新增download功能结构
完成download功能

[Attach a file](#)

Markdown and quick actions are supported

Assignee [Assign to me](#)

Milestone

Labels

Source branch

目标分支 [Change branches](#)

☒ Delete source branch when merge request is accepted.

☐ Squash commits when merge request is accepted. [?](#)

[Submit 合并请求](#) [Cancel](#)

开发者日常Git操作

- 3执行分支合并请求
 - 没有冲突的话可以直接点击Merge按钮，或者自行修改merge commit信息后merge

david > common-test > Merge Requests > 12

[Open](#) Opened 13 秒前 by david [Edit](#) [Close 合并请求](#)

完成download功能

新增download功能結構 完成download功能

请求合并 feature-download 入 dev [在Web IDE中打开](#) [检出分支](#)

Pipeline #33 已取消 for f35a4cd2 on feature-download

Merge ☒ Delete source branch

> 1 次提交 and 1 merge commit will be added to dev. [Modify commit messages](#)

此合并请求可以手动合并，请使用以下命令行


0 0



开发者日常Git操作

- 4合并之后可在dev分支查看提交信息，且不再有feature-download分支

david > common-test > 仓库

dev common-test / +

 完成download功能
由 david 提交于 31 秒前

名称	最后提交
 download.js	完成download功能
 index.js	Initial commit

切换分支/标签

搜索分支和标签

Branches

- ✓ dev
- master

开发者日常Git操作

- 5合并之后删除本地的功能分支
 - 切换到dev分支，并删除feature-download分支
 - 在删除本地功能分支之前，请确认已经上传到远程分支或已合并完成，避免删除之后代码遗失，找回麻烦。

```
dav ~ MINGW64 /d/dav/Desktop/Gitlab測試/GitTest/common-test (feature-download)
$ git checkout dev
Switched to branch 'dev'

dav ~ MINGW64 /d/dav/Desktop/Gitlab測試/GitTest/common-test (dev)
$ git branch -D feature-download
Deleted branch feature-download (was f35a4cd).

dav ~ MINGW64 /d/dav/Desktop/Gitlab測試/GitTest/common-test (dev)
```

开发者日常Git操作

- 6 pull 远程dev分支到本地

```
david@DESKTOP-XXXXX: MINGW64 /d/david/Desktop/Gitlab测试/GitTest/common-test (dev)
$ git pull origin dev
From http://192.168.28.149/david/common-test
 * branch            dev              -> FETCH_HEAD
Updating 3b0c583..86fb15c
Fast-forward
 download.js | 90 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 90 insertions(+)
 create mode 100644 download.js

david@DESKTOP-XXXXX: MINGW64 /d/david/Desktop/Gitlab测试/GitTest/common-test (dev)
$
```

- 7 如有新功能，继续使用最新的dev分支创建功能分支

```
david@DESKTOP-XXXXX: MINGW64 /d/david/Desktop/Gitlab测试/GitTest/common-test (dev)
$ git checkout -b feature-login
Switched to a new branch 'feature-login'

david@DESKTOP-XXXXX: MINGW64 /d/david/Desktop/Gitlab测试/GitTest/common-test (feature-login)
$ |
```

开发者日常Git操作

- 8 release 发行

- 一般是版本功能完成之后，测试无误，合并到master，再定版本号发行，做标签(tag)



开发者日常Git操作

- 新建tag，并输入对应发行说明

david > common-test > 新建标签

新建标签

Tag name

1.0

Create from

master

已存在分支名称，标记或提交SHA

Message

發行1.0版本

(可选)添加一条消息到标签。

发行说明

Write Preview

完成common-test專案1.0版本的所有功能
1、完成download
2、.....

Markdown is supported

Attach a file

(可选)将发行说明添加到标签。它们将被存储在GitLab数据库中并显示在标签页上。

创建标签

取消

开发者日常Git操作

- 完成之后可以在 仓库→标签, 和 项目→Releases中可以看到。



总结

主要说明一般企业使用gitlab配合git管理公司内部产品项目代码的规范规划，以及开发者日常一般操作流程规范示例。

具体公司推行，应该按照实际状况进行。。