OSCAR Tutorial

2025年2月

本次培训日程

2 月 22 日 上午		
Oscar general introduction	李腾	8:30 - 9:10
Oscar physics analysis example I: psi' -> pi pi J/psi	周杭	9:10 - 9:50
茶歇		9:50 - 10:20
Oscar physics analysis example II: J/psi->rho pi	秦小帅	10:20- 11:00
Oscar physics analysis example III: J/psi -> lambda lambdabar	于明玉	11:00 - 11:40
Oscar fast simulation example	秦小帅	11：40 - 12:00
2 月 22 日 下午		
Oscar development guide	李腾	14:00 - 15:00
Oscar hands-on session I		15:00- 15:30
茶歇		15:30 - 16:00
Oscar hands-on session II		16:00 - 17:30

2 月 23 日 上午		
Oscar hands-on session III		8:30 - 10:00
茶歇		10:00- 10:30
Oscar hands-on session IV		10:30- 12:00
2 月 23 日 下午		
Oscar geometry building example with MUD	刘昱林	14:00 - 14:40
Oscar simulation example with DTOF	齐斌斌	14:40 - 15:20
茶歇		15:20- 15:50
Oscar digi + reco example with ECAL	王博	15:50 - 17:00

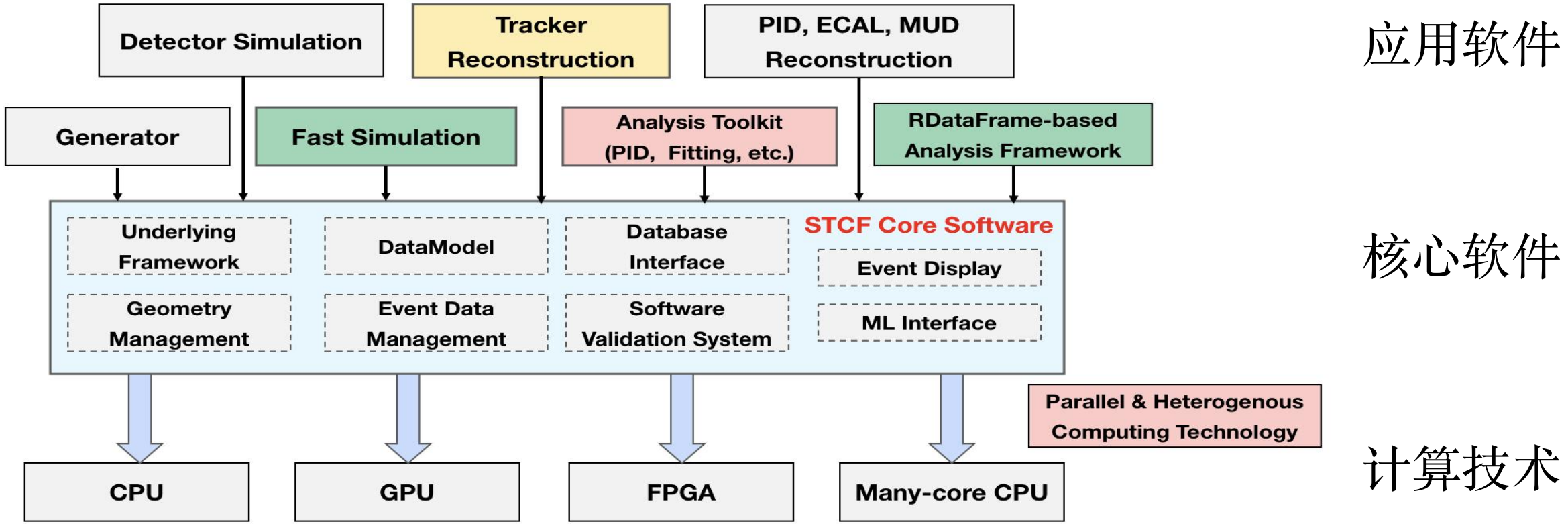
大纲

- 基本介绍
  - OSCAR系统介绍
  - Gitlab基本使用指南
  - OSCAR下载、编译和安装
  - 运行探测器模拟和重建软件
  - 数据分析
- 开发者指南
  - 创建和编译软件包
  - 算法开发
  - 数据模型和事例数据的管理
  - 服务调用
- 开发规范：Gitlab使用、代码规范、测试和文档等

第一部分
基本介绍

基本介绍

- OSCAR （The Offline Software of Super Tau-Charm Facility）
  - 主要用于离线数据处理（物理产生子、探测器模拟、数字化、刻度、重建和数据分析）
  - 对标BESIII-BOSS软件，但采取了更新的软件和计算技术

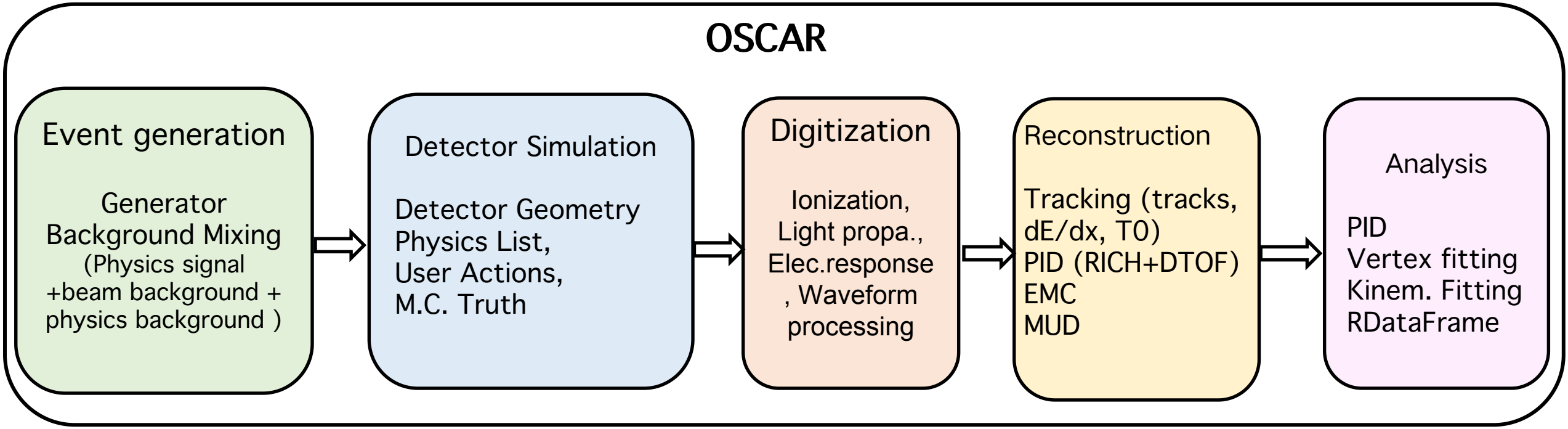


基本介绍

- 编写语言: C++, Python (and fortran)
  - C++： 主要代码
  - Python：作业配置
  - Fortran：部分产生子代码
- 编译、软件包管理: CMake
- 操作系统: SL7/CentOS7 (AlmaLinux9测试中)
  - G++ > 8.5.0 (11.2测试中) (C++17)
- 版本控制: GitLab
- 文档
  - Sphinx

基本介绍

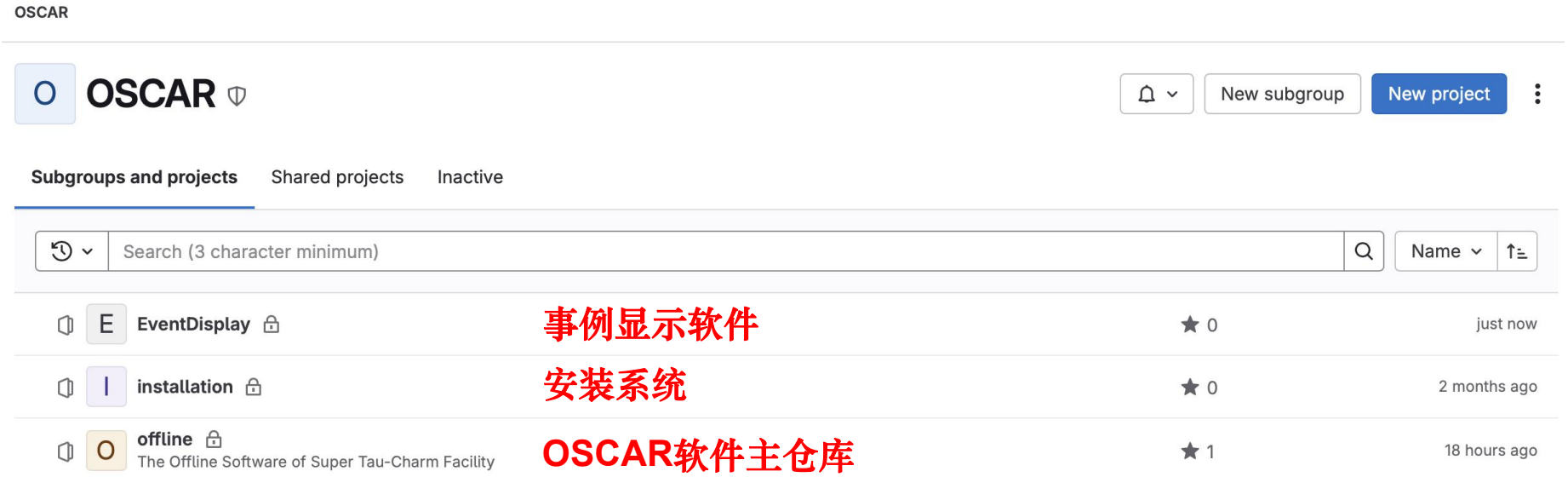
- OSCAR是一个正在快速开发中的项目
  - 最新稳定版本：2.6.0
  - 目前已经支持全离线链条，已具备进行数据分析的条件



第一步：设置Gitlab

OSCAR组和仓库

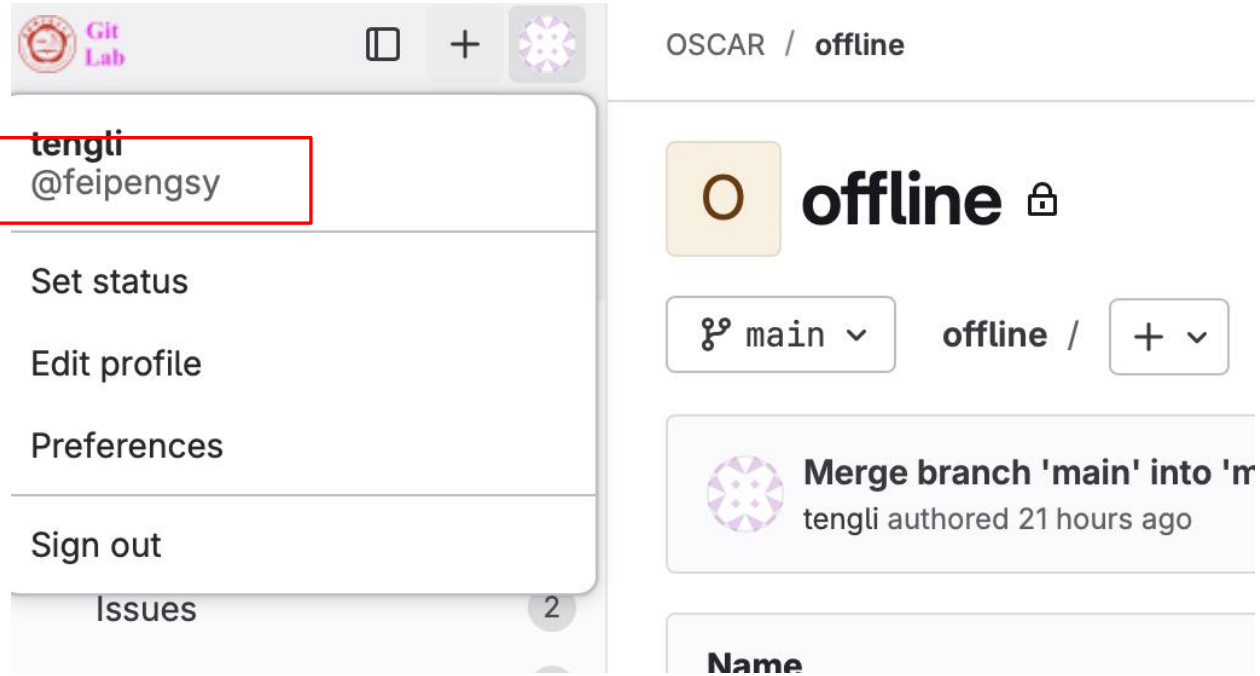
- OSCAR软件和文档源码通过Gitlab管理
- https://git.ustc.edu.cn/oscar1



- 通过邮箱账号登录Gitlab（科大可直接注册，非科大单位需额外授权，请联系胡启鹏或李腾，后者联系科大Gitlab管理员授权注册）
- OSCAR是private group，只有授权过的用户才能访问
- 如果你没有访问权限，请通过网页申请加入，或联系组管理员加入群组
  - 李腾(tengli@sdu.edu.cn)

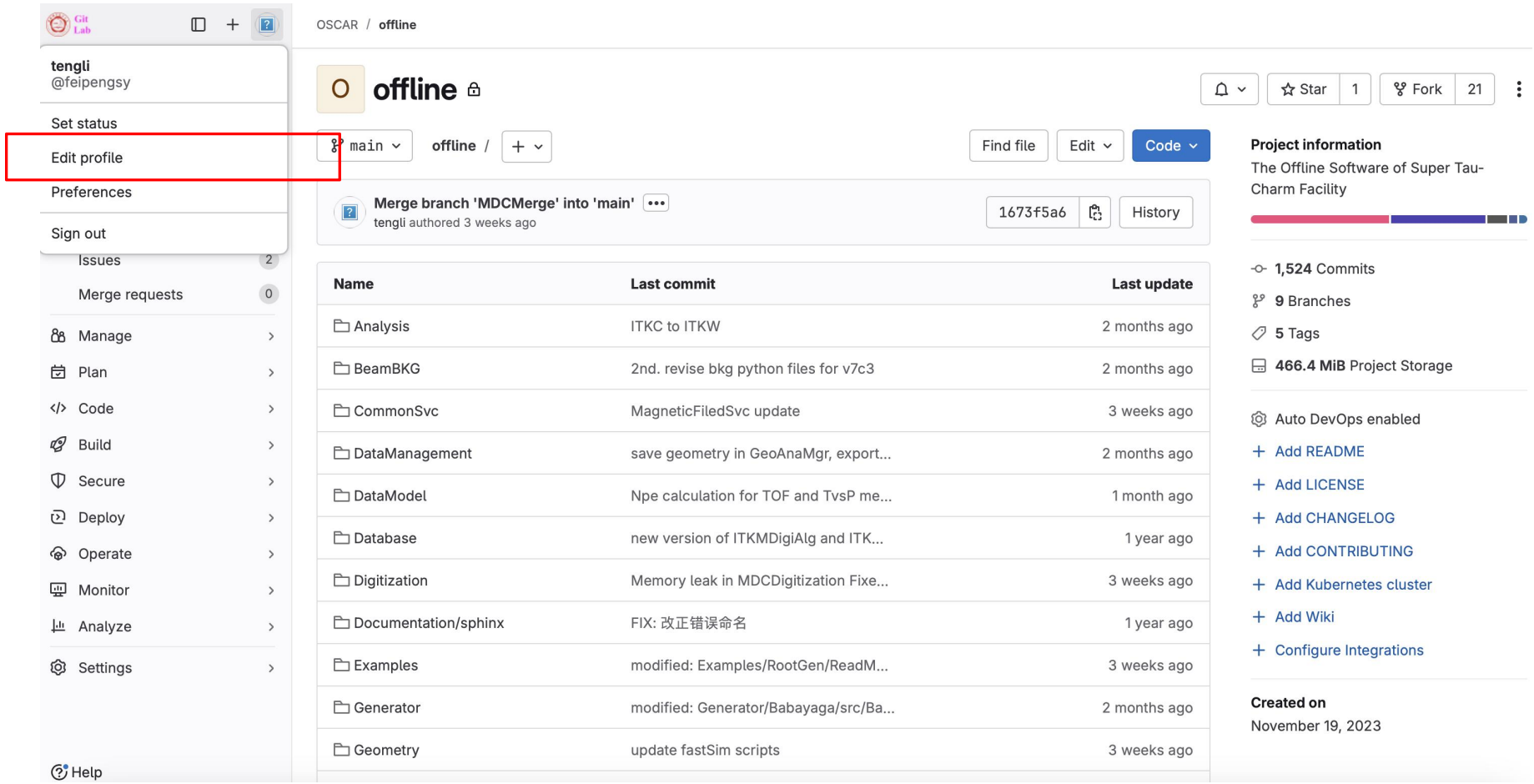
OSCAR组和仓库

- 通过用户名和密码下载OSCAR代码：
  - git clone https://git.ustc.edu.cn/oscar1/offline.git
  - Username & Password （登录网页时所用的密码）



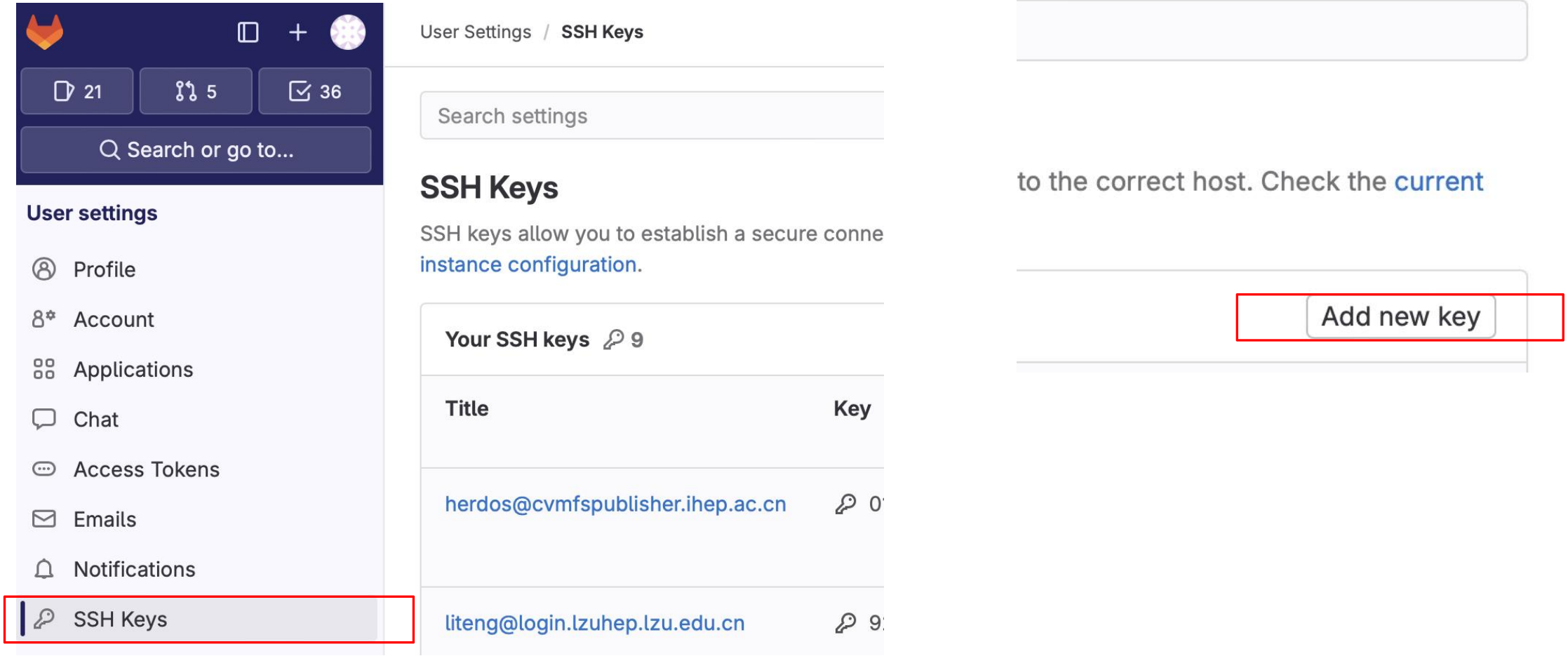
设置你的密钥

- Gitlab仓库的操作（clone、pull、push、fetch等）也可可使用ssh密钥进行认证
- 通过Gitlab网页设置你的密钥



设置你的密钥

- Gitlab仓库的操作（clone、pull、push、fetch等操作）使用ssh密钥进行认证
- 通过Gitlab网页设置你的密钥



设置你的密钥

- cat ~/.ssh/id_rsa.pub
- 如果你还没有ssh密钥，通过命令行创建

```
[tengli@lxlogin001 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/ihep.ac.cn/users/t/tengli/.ssh/id_rsa): /afs/ihep.ac.cn/users/t/tengli/.ssh/id_rsa_new
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/ihep.ac.cn/users/t/tengli/.ssh/id_rsa_new
Your public key has been saved in /afs/ihep.ac.cn/users/t/tengli/.ssh/id_rsa_new.pub
The key fingerprint is:
```

注意不要覆盖掉已有的！！！

```
[tengli@lxlogin001 ~]$ ls .ssh/
authorized_keys  config  id_rsa  id_rsa_new  id_rsa_new.pub  id_rsa.pub  known_hosts
```

私钥　　公钥

不要给任何人不要上传到任何地方

拷贝里面的内容

设置你的密钥

```
[tengli@lxlogin001 ~]$ cat .ssh/id_rsa_new.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCyvmfNGsBtjwdpPttHpJPo6BWOSAxPBYtCDbvg3uHmspPQ8aQVUFzbMV/cr/stIYyYwo77KwhYJbegVqmS4v9RxAQrkdwB/
Kemo7PZhOZEamCTJKCzJAnAU6Q5jjKzCugalQCER8YRIa039xcHfJgbVib99WRwRGcan3qtr3p8x4V/w4Ob3K4MYJLaCpZkhYoSb3oVgDla0GZSQwdcFRhQ6wAL/a78lGqpuh
xdg9jZOroGY2zZqsJPYeZwK9hX52HGGPSlgobuUW+9N6bWpqrXuNkG5/H37/e5RGSOayFZqpfb/tWkQAEnizHWZ6G+ryhq7Sn8tR8U6tVq79mygfvwfowVXwmZvGk6NfpTKXn
uB7iT+Yj8O9Kn2Hxl/6nOpl2dZGnxyQxII1M0v1nSu721+Z7HQnnjt1ox99nu0iFx20LawJPpW2SYdJXUYpHIwbEPS+9EaH3x4DsaiB4XPtw2osuY2FLJ/iDRw2dBs6OrtIhd
XZGdWqzpQZQB/yNbvI0= tengli@lxlogin001.ihep.ac.cn
```

设置你的密钥

- 建议使用默认密钥（~.ssh/id_rsa）
- 如果你不使用默认的密钥，需要在~/.ssh/config中设置

```
Host git.ustc.edu.cn
        Hostname git.ustc.edu.cn
        User git
        IdentifyFile ~/.ssh/id_rsa_new
```

测试你的密钥

- 设置git config

通过命令设置

```
git config --global user.name “tengli”
git config --global user.email “tengli@sdu.edu.cn”

git config --global pull.rebase true
git config --global push.default simple
```

cat ~/.gitconfig

```
# This is Git's per-user configuration file.
[user]
# Please adapt and uncomment the following lines:
        name = tengli
        email = tengli@sdu.edu.cn
[push]
        default = simple
[pull]
        rebase = true
```

- 测试下载OSCAR代码（使用默认密钥 (~/.ssh/id_rsa) ）

git clone git@git.ustc.edu.cn:oscar1/offline.git

第二步：下载和编译OSCAR源码

下载和编译OSCAR

- 使用集群上已安装的外部库环境编译OSCAR是最简单直接的方法
  - 兰大集群
    - /lzufs/user/STCF/oscar
  - 科大集群
    - ...
- 也可以自行安装外部库软件，或使用容器部署环境
  - 本次教程不包括这些内容，请使用集群上已经配置好的环境编译OSCAR

下载和编译OSCAR

- 下载OSCAR代码
  - git clone git@git.ustc.edu.cn:oscar1/offline.git
  - git checkout 2025-tutorial
- 设置外部库环境（使用兰大环境）
  - /cvmfs/container.ihep.ac.cn/bin/hep_container shell CentOS7
  - source /lzufs/user/STCF/oscar/centos7_amd64_gcc850/bashrc.sh
- 编译OSCAR
  - cd offline
  - ./build.sh (或运行build_Debug.sh，build_RelWithDebInfo.sh)
- 设置OSCAR运行环境
  - source install/setup.sh

第二步：运行OSCAR产生数据（模拟-重建）

运行探测器模拟

- **模拟粒子枪电子：**
  - python $OSCAR_INSTALL/share/Tutorial/sim_particle_gun.py
  - 可以自定义$OSCAR_INSTALL/share/Tutorial/electron.mac修改粒子属性

```
/gps/source/clear

/gps/source/add 1 #idensity for particle source, you can add diffrent particles in one source
/gps/particle e- #common particles: e+, e-, mu+, mu-, pi+, pi-, kaon+, kaon-, proton, anti_proton, gamma, opticalphoton

/gps/pos/type Point
/gps/pos/centre 0. 0. 0. cm #vertex position

/gps/ang/type iso #iso for isotropic, cos for cosine-law
/gps/ang/mintheta 22 deg # min angle, reversed in simulation:180-theta
/gps/ang/maxtheta 90 deg # max angle to the beam (z azis)
/gps/ang/minphi 0 deg #min azimuthal angle
/gps/ang/maxphi 360 deg #max azimuthal angle

/gps/ene/type User #User for hist, other options: Mono, Lin, Pow, Exp, Gauss.....
/gps/ene/emspec false #false for momentum, true for kinetic energy
/gps/hist/type energy #define a hist for energy spectrum using points (energy in MeV, weight)
/gps/hist/point 1000 0. # cannot delete this line
/gps/hist/point 1000 1.

/tracking/verbose 0

/gps/source/list
```

运行探测器模拟

- **模拟e+e- -> jpsi pipi -> jpsi pipi mumu：**
  - python $OSCAR_INSTALL/share/Tutorial/sim_generator.py
  - 可以自定义$OSCAR_INSTALL/share/Tutorial/decay.dec修改粒子衰变表

```
# More details seen in Generator/StcfEvtGen/share/DECAY.DEC
##############################################################
# for e+e- -> pipiee or pipimumu
Decay psi(2S)
0.3504    J/psi  pi+ pi-           JPIPI;
Enddecay

Decay J/psi
#0.0594     e+     e-                PHOTOS  VLL;
0.0593     mu+    mu-               PHOTOS  VLL;
Enddecay
##############################################################
```

运行重建

- python $OSCAR_INSTALL/share/Tutorial/rec_Full.py

第三步：查看/分析数据

分析用数据文件: /lzufs/home1/liteng/offline/Examples/Tutorial/simulation.root
/lzufs/home1/liteng/offline/Examples/Tutorial/reconstruction.root

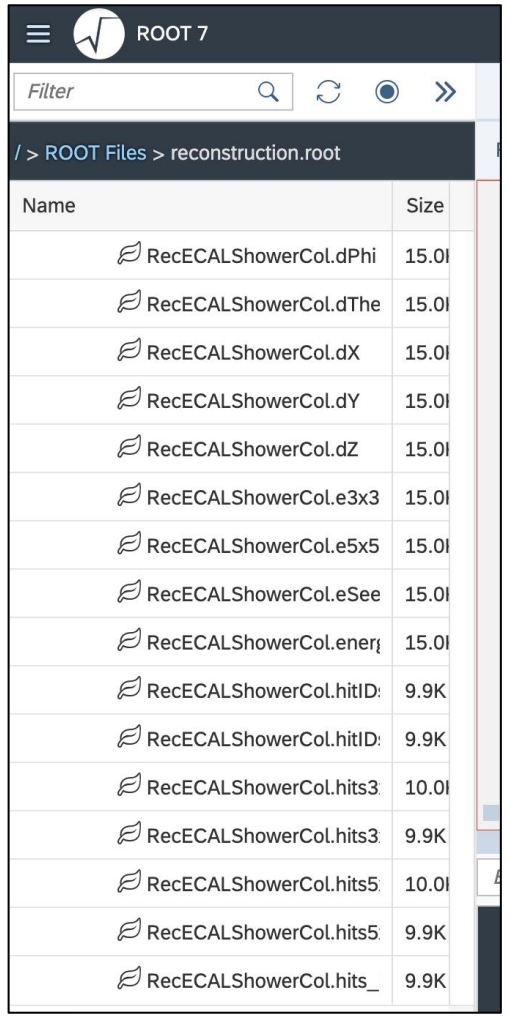
或自行产生：**python $OSCAR_INSTALL/share/Tutorial/sim_particle_gun.py**
python $OSCAR_INSTALL/share/Tutorial/rec_Full.py

分析数据

- OSCAR的事例数据（Event Data Model, EDM）是基于podio开发的。所有的事例数据以collection的方式存在
  - 文件中的collection内容，取决于运行的程序

simulation	digization	reconstruction
MCParticleCol	MDCDigiCol	DEDXRecTrackCol
MDCHitCol	ITKDigiCol	CanTrackCol
ITKHitCol	ECALDigiCol	BTOFLikelihoodCol
ECALHit3Col	MUDDigiCol	GlobalWLPidCol
DTOFBarHitCol	DTOFDigiCol	MUDTrackCol
DTOFPDHitCol	RICHDigiCol	RecECALShowerCol
RICHHitCol	...	ReconstructedParticleCol
...		...

```
244   MDCHit:
245     Description: "Data class for MDCHit"
246     Author: "SDU"
247     Members:
248       - int type //
249       - Vector3d dposition             // Error of position [cm]
250       - int detectorID                 // Detector unique identifier
251       - Vector3d position              // Position of hit [cm]
252
253       - Vector3d posIn // position when the particle enter the cell
254       - Vector3d momIn //
255       - Vector3d posOut // position when the particle leave the cell
256       - Vector3d momOut // test
257       - double pocaZ // z when the track is closest to the wire
258
259       - int pdgID          // pdgID
260       - int parentID       // mother ID
261       - int eventID        // event ID
262       - int trackID        // track ID
263       - int layerID        // layer ID
264       - int cellID         // cell ID
265       - int priElec       // number of primary electrons produced by ionization
266       - int totElec       // total number of electrons
267       - float charge       // charge
268       - float mass         // mass
269       - double time        // time
270       - double Qmeasurement    // measured Q(fC)
271       - double TrackLen        // track length in cell for tem use
272       - double TrackLenAll     // accumulate track length
273       - Vector3d initialMom    // init momentum
274       - Vector3d initialPos    // init position
275       - double driftDistance   // drift distance
```



- 模拟-数字化-重建作业产生的数据文件是ROOT格式（但以podio的方式组织，并非简单的TTree/TNtuple格式），分析这样的数据，需要一定的技巧
- 分析podio格式数据的几个方法：
  - 原始ROOT脚本
  - podio源码（ROOT脚本）
  - OSCAR算法

https://git.ustc.edu.cn/oscar1/offline/-/blob/main/DataModel/datalayout.yaml

方法一：使用标准ROOT脚本分析数据

- 最简单的方法，适用于快速简单的分析。不适用于复杂的分析（例如需要读取数据之间的关联关系，需要读取EDM中的vector member等内容）

```cpp
/*Using TTreeReader to read EDM data file
 * Get energy of reconstructed showers */
#include <iostream>
#include "TCanvas.h"
#include "TH1.h"
#include "TTree.h"
#include "TFile.h"
//TTreeReader related headers
#include <TTreeReader.h>
#include <TTreeReaderValue.h>
#include <TTreeReaderArray.h>

using namespace std;

void draw_shower(TString filename)
{
   //Define branches to read.The branch names are in the form of collection.member
   TTreeReader fReader; //the tree reader
   auto* shower_e = new TTreeReaderArray<double>(fReader,"RecECALShowerCol.energy");

   //Connect the reader with event tree
   TFile* f = new TFile(filename,"read");
   TTree* ReadTree = (TTree*)f->Get("events");
   fReader.SetTree(ReadTree);

   //Total entries
   int nentries = fReader.GetEntries(true);

   TH1F* h = new TH1F("shower energy distribution","shower energy distribution",100,0,1.5);
   for (int jentry=0;jentry<nentries;++jentry)
   {
      //Read the entry
      fReader.SetEntry(jentry);
      double energy=0;
      //Each element in shower_e is a reconstructed total energy deposition
      for (int i=0;i<shower_e->GetSize();++i)
      {
         energy = (*shower_e)[i];
         h->Fill(energy);
      }
   }
   TCanvas* c = new TCanvas("c","c",800,600);
   h->Draw();
   gPad->SetLogy();
   c->SaveAs("eshower.png");
}

#if !defined(__CINT__) && !defined(__CLING__)
int main(int argv,char* argc[])
{
   TString filename = argc[1];
   UseTTreeReader(filename);
}
#endif
```

https://git.ustc.edu.cn/oscar1/offline/-/blob/2025-tutorial/Examples/Tutorial/Analysis/draw_shower.C

方法一：使用标准ROOT脚本分析数据

```
1  /*Using TTreeReader to read EDM data file
2   * Get energy of reconstructed showers */
3  #include <iostream>
4  #include "TCanvas.h"
5  #include "TH1.h"
6  #include "TTree.h"
7  #include "TFile.h"
8  //TTreeReader related headers
9  #include <TTreeReader.h>
10 #include <TTreeReaderValue.h>
11 #include <TTreeReaderArray.h>
12
13 using namespace std;
14
15 void draw_shower(TString filename)
16 {
17    //Define branches to read.The branch names are in the form of collection.member
18    TTreeReader fReader; //the tree reader
19    auto* shower_e = new TTreeReaderArray<double>(fReader,"RecECALShowerCol.energy");
20
21    //Connect the reader with event tree
22    TFile* f = new TFile(filename,"read");
23    TTree* ReadTree = (TTree*)f->Get("events");
24    fReader.SetTree(ReadTree);
25
```

相关头文件

定义要读取的Branch名字
(collection.member)

打开数据文件，读取TTree

方法一：使用标准ROOT脚本分析数据

```
   //Total entries
   int nentries = fReader.GetEntries(true);

   TH1F* h = new TH1F("shower energy distribution","shower energy distribution",100,0,1.5);
   for (int jentry=0;jentry<nentries;++jentry)
   {
      //Read the entry
      fReader.SetEntry(jentry);
      double energy=0;
      //Each element in shower_e is a reconstructed total energy deposition
      for (int i=0;i<shower_e->GetSize();++i)
      {
         energy = (*shower_e)[i];
         h->Fill(energy);
      }
   }
   TCanvas* c = new TCanvas("c","c",800,600);
   h->Draw();
   gPad->SetLogy();
   c->SaveAs("eshower.png");
}

#if !defined(__CINT__) && !defined(__CLING__)
int main(int argv,char* argc[])
{
   TString filename = argc[1];
   UseTTreeReader(filename);
}
#endif
```

总entry数目

读取Entry
绘制直方图

用法：root draw_shower.C -- '"reconstruction.root"'

方法二：使用podio源码读取

```
/*Using podio to read EDM data file
 * Get energy of reconstructed showers.*/

#include <iostream>
#include "TH1.h"
#include "TCanvas.h"
//podio related headers
#include <podio/EventStore.h>
#include <podio/ROOTReader.h>
//EDM related headers
#include <DataModel/RecECALShowerCollection.h>

using namespace std;

void UsePodio(TString filename)
{
   //Open file in ROOTReader and then connect to EventStore
   podio::EventStore store;
   podio::ROOTReader reader;
   reader.openFile(string(filename));
   store.setReader(&reader);


   TH1F* h = new TH1F("shower energy distribution","shower energ
   //Findout the total number of entries
   for (int jentry=0;jentry<reader.getEntries();++jentry)
   {
      auto& showers = store.get<RecECALShowerCollection>("RecECALShowerCol");

      for (int i=0;i<showers.size();++i)
      {
         //showers is the collection of all reconstructed showers
         auto shower = showers.at(i);
         float energy=shower.getEnergy();
         h->Fill(energy);
      }

      //Prepare the next event.The next store.get() will then read from the n
      reader.endOfEvent();
      store.clear();
   }
   TCanvas* c = new TCanvas("c","c",800,600);
   h->Draw();
   gPad->SetLogy();
   c->SaveAs("energy.png");
}

#if !defined(__CINT__) && !defined(__CLING__)
int main(int argv,char* argc[])
{
   TString filename = argc[1];
   UsePodio(filename);
}
#endif
```

can be found at https://git.ustc.edu.cn/oscar1/offline/-/blob/2025-tutorial/Examples/Tutorial/Analysis/UsePodio.C

方法二：使用podio源码读取

- 该方法需要额外运行loader.C加载所需库

```
/*use this file to run UsePodio.C by root directly
 * run the program by:
 * root loader.C 'UsePodio.C("input.root")'
 * */
{
   gInterpreter->AddIncludePath(gSystem->ExpandPathName("$OSCAR_EXTLIB_podio_HOME/include"));
   gSystem->Load("libpodio");
   gSystem->Load("libpodioRootIO");
   gSystem->Load("libDataModel");
   gSystem->Load("libDataModelDict");
}
```

- 使用方法：
  - 1） `root loader.C UsePodio.C -- '"reconstruction.root"'`
  - 2） `root loader.C UsePodio.C'("reconstruction.root")'`

https://git.ustc.edu.cn/oscar1/offline/-/blob/2025-tutorial/Examples/Tutorial/Analysis/loader.C
我们提供了makefile文件，也可以将其编译成可执行程序 (只需要在目录里运行make)

方法二：使用podio源码读取

```
4  #include <iostream>
5  #include "TH1.h"
6  #include "TCanvas.h"
7  //podio related headers
8  #include <podio/EventStore.h>
9  #include <podio/ROOTReader.h>
10 //EDM related headers
11 #include <DataModel/RecECALShowerCollection.h>
```

podio 头文件

EDM 头文件

```
//Open file in ROOTReader and then connect to EventStore
podio::EventStore store;
podio::ROOTReader reader;
reader.openFile(string(filename));
store.setReader(&reader);
```

打开文件，创建EventStore和ROOTReader （固定语法）

方法二：使用podio源码读取

```
24      TH1F* h = new TH1F("shower energy distribution","shower energy distributio
25      //Findout the total number of entries
26      for (int jentry=0;jentry<reader.getEntries();++jentry)
27      {
28          auto& showers = store.get<RecECALShowerCollection>("RecECALShowerCol");
29
30          for (int i=0;i<showers.size();++i)
31          {
32              //showers is the collection of all reconstructed showers
33              auto shower = showers.at(i);
34              float energy=shower.getEnergy();
35              h->Fill(energy);
36          }
37
38          //Prepare the next event The next store.get() will then read from the n
39          reader.endOfEvent();
40          store.clear();
41      }
42      TCanvas* c = new TCanvas("c","c",800,600);
43      h->Draw();
44      gPad->SetLogy();
45      c->SaveAs("energy.png");
```

获取需要的collection

遍历collection，获取每个shower的能量值

清理内存（固定语法）

提交作业

- **加载作业环境进入容器**
  - 可在个人bashrc中添加以下内容：
    export PATH=/cvmfs/common.ihep.ac.cn/software/hepjob/bin/:/cvmfs/container.ihep.ac.cn/bin/:$PATH
    alias key="hep_container shell CentOS79 -g STCF"
  - 另提交作业时需source oscar外部库环境和offline环境
- **作业提交**
  - **hep_sub** job.sh （默认提交CPU） -g STCF （指定组信息） （默认内存3G）
- **作业状态查询**
  - **hep_q** –u 查询用户所有作业 -hold （查询作业hold原因） -wt long （超过100小时，指定长作业）
- **作业删除**
  - **hep_rm** JOBID （删除指定作业） -a （删除全部作业）

站点使用手册参见：
http://afsapply.ihep.ac.cn/cchelp/zh/remote-sites/lzu/

第一部分小结

- 一、设置Gitlab
- 二、下载和编译OSCAR
- 三、运行模拟和重建
- 四、两种分析数据的方法
- 五、提交作业

Hands-on
Q&A

第二部分
算法开发

一、创建算法包

OSCAR目录结构

OSCAR以算法包的方式管理所有的源代码

Analysis	ITKC to ITKW
BeamBKG	2nd. revise bkg python files for v7c3
CommonSvc	MagneticFiledSvc update
DataManagement	save geometry in GeoAnaMgr, exported from gGeoM...
DataModel	Npe calculation for TOF and TvsP method separately
Database	new version of ITKMDigiAlg and ITKM geometry
Digitization	Memory leak in MDCDigitization Fixed. Save MCPare...
Documentation/sphinx	FIX: 改正错误命名
Examples	add Makefile
Generator	modified: Generator/Babayaga/src/Babayaga.cxx
Geometry	update fastSim scripts

Performance	change the eg of Lamvertexfit
Reconstruction	import cnn model onnx file
Simulation	update fastSim scripts
Utilities	optimize some automated tests
Validation	add automated kinetic fit test
cmake	Dtofcnnonnx
.gitignore	delete *.py in .gitignore
CMakeLists.txt	remove CMAKE warning
build.sh	change xvangle from 45 to 15 deg, change name ITK...
build_Debug.sh	new version of ITKMDigiAlg and ITKM geometry
build_RelWithDebInfo.sh	new version of ITKMDigiAlg and ITKM geometry
build_Test.sh	add automated kinetic fit test

可放置多级目录

OSCAR目录结构

OSCAR以算法包的方式管理所有的源代码

build: 编译期间产生的临时文件
install: 编译后安装的头文件、库和运行脚本等
cmake: CMake源码，规定OSCAR编译、安装、外部库管理等策略
CommonSvc：通用服务（如随机数服务等）
Database：数据库接口服务
DataModel：数据模型
DataManagement：事例数据管理（TES、输入、输出等）
Documentation：文档相关源码
Examples：各种示例
Utilities: 通用工具
Generator, Simulation, Digitization, Calibration, Reconstruction：产生子、模拟、数字化、刻度和重建算法
Analysis：分析工具、分析示例等
Geometry：探测器几何相关软件
Validation：软件测试和性能验证
Performance：物理性能测试

可放置多级目录，不允许将算法包直接放置在offline下

OSCAR目录结构

- OSCAR支持两种开发模式：
  1. OSCAR代码树中直接开发算法包
     - 开发的算法包将作为OSCAR的一部分，整体编译
  2. 独立于OSCAR代码树开发算法包
     - 开发的算法包独立于OSCAR存在
     - 本次教程不涉及这种开发模式
- 一般不允许开发者在顶级目录放置算法包
  - 请选择合适的二级目录，创建你的算法包
  - 本教程将在Example下面开发算法包

创建算法包

- 创建算法包
  - cd Example
  - addpackage ECALAna -d PodioDataSvc DataModel

算法包名　　算法包依赖的其它包

（使用该命令需要先编译OSCAR并设置运行环境）

一些常见的依赖包：
PodioDataSvc：读写数据
DataModel：数据模型
VertexFit：运动学/顶点拟合
GlobalPID：PID（机器学习）
GlobalWLPID：PID（传统）
GeometrySvc：几何

- 算法包目录结构

```
CMakeLists.txt  ECALAna  python  share  src
```

编译规则

放置需要开放的头文件

与算法包名对应

放置Python代码

配置脚本相关

放置C++源代码

创建算法包

CMakelist.txt

```
oscar_add_pkg(ECALAna
              DEPENDS PodioDataSvc DataModel
             )
```

算法包依赖PodioDataSvc和DataModel两个包

如果你的算法中会调用其它包的函数，或使用其头文件，需要加入依赖关系，保证相关的库在编译期间被连接

python/ECALAna/__init__.py

```
import Sniper
Sniper.loadDll("libECALAna.so")
del Sniper
```

确保Sniper可以“看到”这个算法包编译的库文件
库文件名和ECALAna文件夹名与算法包名对应！

如果手动创建算法包，上面这两个文件这些是必不可少的

二、开发算法

开发算法

- src/ECALAnaAlg.h　　一般来说算法的头文件无需开放给其它包，所以头文件也放置在/src里

```
#pragma once

#include <SniperKernel/AlgBase.h>

class ECALAnaAlg : public AlgBase
{
   public :
      // Default constructor
      ECALAnaAlg(const std::string& name);
      // Default destructor
      virtual ~ECALAnaAlg();

      virtual bool initialize();
      virtual bool execute();
      virtual bool finalize();

   private :
};
```

算法基类头文件

算法基类AlgBase

默认构造函数，接受字符串作为参数（对象名）

析构函数

initialize()：run开始时执行一次
execute()：每个事例执行一次
finalize()：run结束时执行一次
这三个函数是算法的主体

开发算法

- src/ECALAnaAlg.cc

```
#include "ECALAnaAlg.h"
#include "SniperKernel/AlgFactory.h"

// Declaration of the algorithm, by the name of the class.
DECLARE_ALGORITHM(ECALAnaAlg);

ECALAnaAlg::ECALAnaAlg(const std::string& name) : AlgBase(name)
{ }

// Destructor
ECALAnaAlg::~ECALAnaAlg()
{ }

bool ECALAnaAlg::initialize()
{
   return true;
}

bool ECALAnaAlg::execute()
{
   return true;
}

bool ECALAnaAlg::finalize()
{
   return true;
}
```

头文件

声明算法宏，与算法类名对应。固定语法

构造函数参数传往基类

主体函数的实现，暂时留空

三、配置和运行算法

配置和运行算法

- 首先将你的软件包加入OSCAR编译
- 修改Examples/CMakelist.txt

```
set(dirlist DemoSim  EDMtests  HelloWorld  MultiTask ECALAna)
foreach(dir ${dirlist})
  add_subdirectory(${dir})
endforeach()
```

回到offline目录
重新运行build.sh完成编译

配置和运行算法

- scripts/run.py　Sniper作业通过Python配置

```
#!/usr/bin/env python

import Sniper

task = Sniper.Task("task")
task.setLogLevel(3)

import ECALAna
alg = task.createAlg("ECALAnaAlg/calg")

task.setEvtMax(100)
task.show()
task.run()
```

python run.py 查看运行结果（需要完成编译，设置运行环境）

载入Sniper

创建Task (Sniper中的基本执行单元)

设置打印日志详细程度
0-5，数字越小，内容越详细

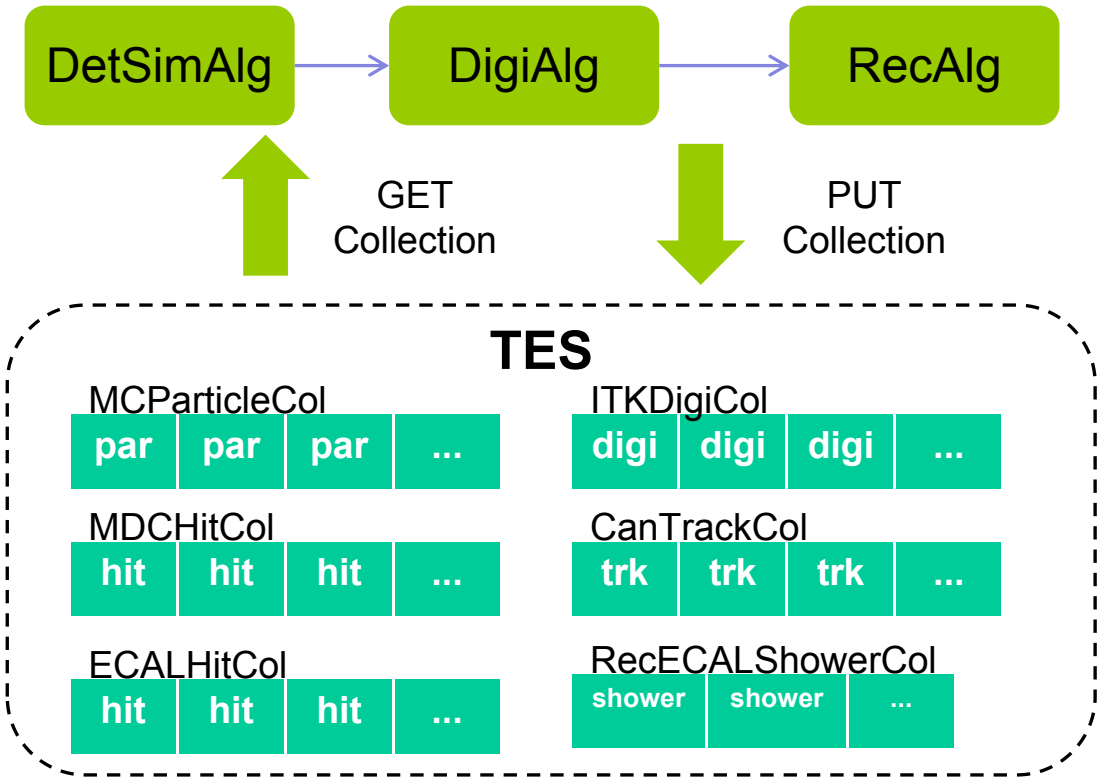
加载和创建刚刚开发的算法
import PackageName
createAlg(算法类名/算法实例名)
一个Task中可以有多个算法

设置运行事例数目（-1为无限，耗尽输入事例位置）
打印task配置
运行task

四、访问数据

在算法中访问数据

- 如何在算法中访问数据？



算法

数据访问接口

框架管理的内存区域

- 算法可以访问内存中的数据（collection）
- 内存中的数据可由算法创建（一个算法创建了，其它算法即可访问），也可自输入文件输入
- 内存中的数据可输出到输出文件

在算法中访问数据

- 在算法中读数据

```
#include "ECALAnaAlg.h"
#include "SniperKernel/AlgFactory.h"
#include "PodioDataSvc/DataHandle.h"
#include "DataModel/RecECALShowerCollection.h"

// Declaration of the algorithm, by the name of the class.
DECLARE_ALGORITHM(ECALAnaAlg);

ECALAnaAlg::ECALAnaAlg(const std::string& name) : AlgBase(name)
{ }
```

DataHandle.hh (接口头文件)
相关的EDM collection头文件

在offline/DataModel/DataModel/中寻找全部列表

```
bool ECALAnaAlg::execute()
{
    auto showers = getROColl(RecECALShowerCollection, "RecECALShowerCol");
    if (showers) {
        for (size_t i=0; i<showers->size(); ++i)
        {
            auto shower = showers->at(i);
            double energy = shower.getEnergy();
            mHist->Fill(energy);
        }
    }
```

getROColl读取输入文件中的collection
两个参数：collection类型，collection名称
返回collection指针（失败则返回空指针）

读取当前事例的全部shower能量

在算法中访问数据

- 在算法中写数据

```
bool ECALAnaAlg::execute()
{
   auto showers = getROColl(RecECALShowerCollection, "RecECALShowerCol");
   if (showers) {
      for (size_t i=0; i<showers->size(); ++i)
      {
         auto shower = showers->at(i);
         double energy = shower.getEnergy();
         mHist->Fill(energy);
      }
   }

   auto showers1 = getRWColl(RecECALShowerCollection, "RecECALShowerCol1");
   auto sh = showers1->create();
   sh.setEnergy(1.0);

   auto sh1 = MutableRecECALShower();
   sh1.setEnergy(2.0);
   showers1->push_back(sh1);

   return true;
}
```

DataHandle.h (接口头文件)
相关的EDM collection头文件

在offline/DataModel中寻找全部列表

getRWColl读取输入文件中的collection
两个参数：collection类型，collection名称
返回collection指针

创建EDM对象，放入collection

在算法中访问数据

- getROColl和getRWColl的区别
  - getROColl首先尝试在内存中寻找collection，如果找不到，**则访问输入文件尝试读入**，如果输入文件中也没有，则返回NULL。getROColl返回的collection**只读**。
  - getRWColl首先尝试在内存中寻找collection，如果找不到，**则创建一个空的collection**。getRWColl返回的collection**可读写**。

getROColl多用来读文件中的collection

getRWColl多用来创建新的collection

如果getROColl和getRWColl都能满足你的需求，优先使用getROColl

getROColl和getRWColl要在execute()中使用，不能在initialize()中使用

在算法中访问数据

- Python配置文件

```
#!/usr/bin/env python

import Sniper

task = Sniper.Task("task")
task.setLogLevel(3)

import ECALAna
alg = task.createAlg("ECALAnaAlg/calg")

import PodioDataSvc
pSvc = task.createSvc("PodioDataSvc")

import PodioSvc
Isvc = task.createSvc("PodioInputSvc/InputSvc")
Isvc.property("InputFile").set("reconstruction.root")

Osvc = task.createSvc("PodioOutputSvc/OutputSvc")
Osvc.property("OutputFile").set("output.root")
Osvc.property("OutputCollections").set( [ "RecECALShowerCol1" ])


task.setEvtMax(100)
task.show()
task.run()
```

输入输出和内存服务由三个服务实现

- PodioDataSvc
- PodioInputSvc
- PodioOutputSvc

如果需要访问数据，PodioDataSvc是必须的

如果需要输入文件，配置PodioInputSvc和输入文件

如果需要输出，配置PodioOutputSvc和输出文件、输出的collection

五、向算法中传递参数

向算法中传递参数

- 在开发算法过程中，经常需要定义一些参数。将参数的值写死在C++代码中（硬编码），每次修改都要重新编译代码，是不合适的。
- 在OSCAR中，通常的做法是在Python脚本中配置这些参数，然后传递给算法。如果需要修改参数值，只需修改python脚本，或通过命令行（argparse）传入参数的值，无需重新编译软件。
- Sniper提供了这样的机制，可以自python向算法中传递变量：
  - Scalar type变量：整数、浮点数、布尔变量、字符串
  - 数组(vector)：scalar type变量的vector
  - 键值对(map)：scalar type变量的map

```
import PodioSvc
Isvc = task.createSvc("PodioInputSvc/InputSvc")
Isvc.property("InputFile").set("reconstruction.root")

Osvc = task.createSvc("PodioOutputSvc/OutputSvc")
Osvc.property("OutputFile").set("output.root")
Osvc.property("OutputCollections").set( [ "RecECALShowerCol1" ])
```

向算法中传递参数

在算法类中声明成员变量

```
#pragma once

#include <SniperKernel/AlgBase.h>
#include <vector>
#include <TH1.h>

class ECALAnaAlg : public AlgBase
{
   public :
      // Default constructor
      ECALAnaAlg(const std::string& name);
      // Default destructor
      virtual ~ECALAnaAlg();

      virtual bool initialize();
      virtual bool execute();
      virtual bool finalize();

   private :
      float mFloatValue;
      std::vector<float> mFloatVector;
      TH1F* mHist;
};
```

在算法类构造函数中声明Property

```
// Declaration of the algorithm, by the name of the class.
DECLARE_ALGORITHM(ECALAnaAlg);

ECALAnaAlg::ECALAnaAlg(const std::string& name) : AlgBase(name)
{
   declProp("SomeValue", mFloatValue=42.0);
   declProp("SomeVector", mFloatVector);
}

// Destructor
ECALAnaAlg::~ECALAnaAlg()
{ }

bool ECALAnaAlg::initialize()
{
   LogInfo << "mFloatValue: " << mFloatValue << std::endl;
   mHist = new TH1F("h","h",100,0,1.5);
   return true;
}
```

向算法中传递参数

Python配置

```python
#!/usr/bin/env python

import Sniper

task = Sniper.Task("task")
task.setLogLevel(3)

import ECALAna
alg = task.createAlg("ECALAnaAlg/calg")
alg.property("SomeValue").set(23.0)
alg.property("SomeVector").set([1.0, 1.0, 2.0, 3.0, 5.0, 8.0])

import PodioDataSvc
pSvc = task.createSvc("PodioDataSvc")

import PodioSvc
Isvc = task.createSvc("PodioInputSvc/InputSvc")
Isvc.property("InputFile").set("reconstruction.root")

Osvc = task.createSvc("PodioOutputSvc/OutputSvc")
Osvc.property("OutputFile").set("output.root")
Osvc.property("OutputCollections").set( [ "RecECALShowerCol1" ])


task.setEvtMax(100)
task.show()
task.run()
```

运行结果

```
      }
    },
    {
      "identifier": "PodioOutputSvc/OutputSvc",
      "properties": {
        "OutputCollections": [ "RecECALShowerCol1" ],
        "OutputFile": "output.root",
        "TransferAllBut": [],
        "TransferAllCollections": false,
        "TransferCollections": []
      }
    }
  ],
  "algorithms": [
    {
      "identifier": "ECALAnaAlg/calg",
      "properties": {
        "SomeValue": 23,
        "SomeVector": [ 1, 1, 2, 3, 5, 8 ]
      }
    }
  ]
}
```

六、访问服务

在算法中访问服务

- OSCAR中有一些公共服务，可以用来实现数据处理和分析中的各种功能
- 如何在算法中调用这些服务？
- 例一：调用ROOTWriter写出直方图

```
#pragma once

#include <SniperKernel/AlgBase.h>
#include <vector>
#include <TH1.h>

class ECALAnaAlg : public AlgBase
{
   public :
      // Default constructor
      ECALAnaAlg(const std::string& name);
      // Default destructor
      virtual ~ECALAnaAlg();

      virtual bool initialize();
      virtual bool execute();
      virtual bool finalize();

   private :
      float mFloatValue;
      std::vector<float> mFloatVector;
      TH1F* mHist;
};
```

```
#include <RootWriter/RootWriter.h>
#include <SniperKernel/SniperPtr.h>
```

```
bool ECALAnaAlg::initialize()
{
   LogInfo << "mFloatValue: " << mFloatValue << std::endl;
   mHist = new TH1F("h","h",100,0,1.5);

   SniperPtr<RootWriter> ws(getParent(), "RootWriter");
   if (ws.valid())
   {
      ws->attach("Fkey/histDir", mHist);
   }

   return true;
}
```

```
bool ECALAnaAlg::execute()
{
   auto showers = getROColl(RecECALShowerCollection, "RecECALShowerCol");
   if (showers) {
      for (size_t i=0; i<showers->size(); ++i)
      {
         auto shower = showers->at(i);
         double energy = shower.getEnergy();
         mHist->Fill(energy);
      }
   }
```

在算法中访问服务

- 例一：调用ROOTWriter写出直方图

```
#!/usr/bin/env python

import Sniper

task = Sniper.Task("task")
task.setLogLevel(3)

import ECALAna
alg = task.createAlg("ECALAnaAlg/calg")
alg.property("SomeValue").set(23.0)
alg.property("SomeVector").set([1.0, 1.0, 2.0, 3.0, 5.0, 8.0])

import RootWriter
svc = task.createSvc("RootWriter")
svc.property("Output").set({"Fkey" : "Hist.root"})

import PodioDataSvc
pSvc = task.createSvc("PodioDataSvc")

import PodioSvc
Isvc = task.createSvc("PodioInputSvc/InputSvc")
Isvc.property("InputFile").set("reconstruction.root")

Osvc = task.createSvc("PodioOutputSvc/OutputSvc")
Osvc.property("OutputFile").set("output.root")
Osvc.property("OutputCollections").set( [ "RecECALShowerCol1" ])


task.setEvtMax(100)
task.show()
task.run()
```

性能分析

- 使用jMonitor工具分析内存消耗、磁盘I/O、CPU占用等信息
  - jMonitor -h
  - jMonitor --gen-log -n sim_gen_dtof --enable-monitor --enable-io-profile “python run.py”
- 使用SniperProfiler统计算法耗时

  import SniperProfiling

  profiling = task.createSvc("SniperProfiling")
- 使用valgrind工具分析内存泄漏、野指针、悬垂指针等疑难问题
  - valgrind -h

```
########################## SniperProfiling ##########################
Name            Num of calls   Total(ms)      Mean(ms)       RMS(ms)
EventSamplingAlg12000            1307.31399     0.65366        6.44149
ITKWDigiAlg     2000           188257.38791   94.12869       184.83125
MDCDigiAlg      2000           72463.01697    36.23151       26.92424
ECALDigiAlg     2000           380141.55820   190.07078      182.28040
DTOFDigiAlg     2000           128.51900      0.06426        0.09076
RICHDigiAlg     2000           6501.41400     3.25071        0.72662
MUDDigiAlg      2000           26942.33102    13.47117       10.54209
ITKWClusterRecAlg2000            53.33800       0.02667        0.01226
MDCHitRecAlg1   2000           255.57100      0.12779        0.07784
HoughFinderAlg12000            38514.52100    19.25726       8.36381
FitTrackAlg1    2000           121359.68007   60.67984       10.51385
DeDxRecAlg      2000           1149.50400     0.57475        0.11380
TrackExtAlg1    2000           102161.25196   51.08063       8.21864
ECALRecAlg      2000           241.93300      0.12097        0.21304
DTOFRecAlg      2000           14715.03802    7.35752        21.11030
RICHRecAlg      2000           1264618.69365  632.30935      247.60813
MUDRecIdentifyAlg2000            2856.64799     1.42832        9.44662
GlobalWLPID     2000           63.75300       0.03188        0.01493
EventAssembler 2000            118.45600      0.05923        0.05952
TrackSummaryWriter2000            811.19100      0.40560        0.43217
EMCSummaryWriter2000            32.31200       0.01616        0.01421
DTOFSummaryWriter2000            477.84200      0.23892        0.50032
RICHSummaryWriter2000            200.44100      0.10022        0.25225
MUDSummaryWriter2000            1.30200        0.00065        0.00062
event           2000           2229988.24876  1114.99412     373.59251
#####################################################################
```

第二部分小结

- 一、创建算法包
- 二、开发算法
- 三、配置和运行算法
- 四、访问数据
- 五、向算法中传递参数
- 六、访问服务
- 七、性能分析
- 所有教程中的代码可以在offline/Example/ECALAna算法包中找到
  - 建议按顺序练习

练习
Q&A

一、基于Gitlab的标准开发流程

OSCAR开发规则

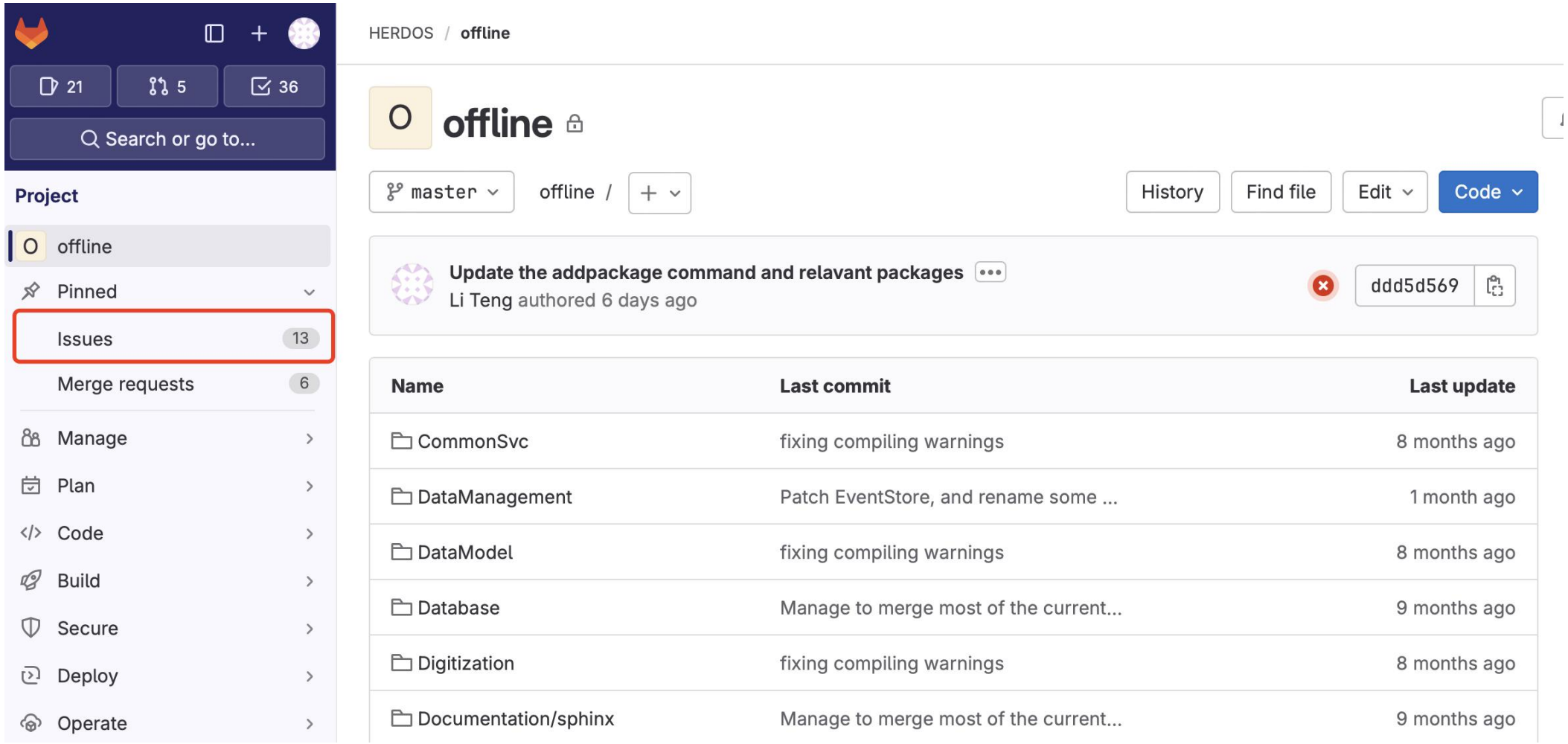
- 开发流程
  1. 计划
  2. 开发
  3. 合并
  4. 发布

OSCAR开发规则

- 1. 计划
  - 当开发者准备在OSCAR中开发，或建议新功能/算法/工具，或发现代码中的问题，请在gitlab系统中创建一个issue，并通知相关的子系统负责人（设置为assignee，gitlab系统会自动发邮件通知）。Issue中的文字描述要详细。如果汇报bug，请提供复现bug的环境和脚本。
  - 相关子系统负责人视情况拆分issue，分配任务。
  - 发起issue的人在issue中与开发者充分交流，并在开发结束后提供反馈。相关子系统负责人及时跟进开发进展，并决定什么时候可以关闭issue
  - 相关的代码分支以issue的编号开头（例如：53-fix-geometry）

创建Issue

- 创建issue



New Issue

Title (required)

Type

Issue

Description

Preview

Write a description or drag your files here…

Switch to rich text editing

Add description templates to help your contributors to communicate effectively!

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee

Unassigned Assign to me

Due date

Select due date

Milestone

Select milestone

Labels

Select label

Create issue Cancel

Project

offline

Pinned

Issues

Merge r

Manage

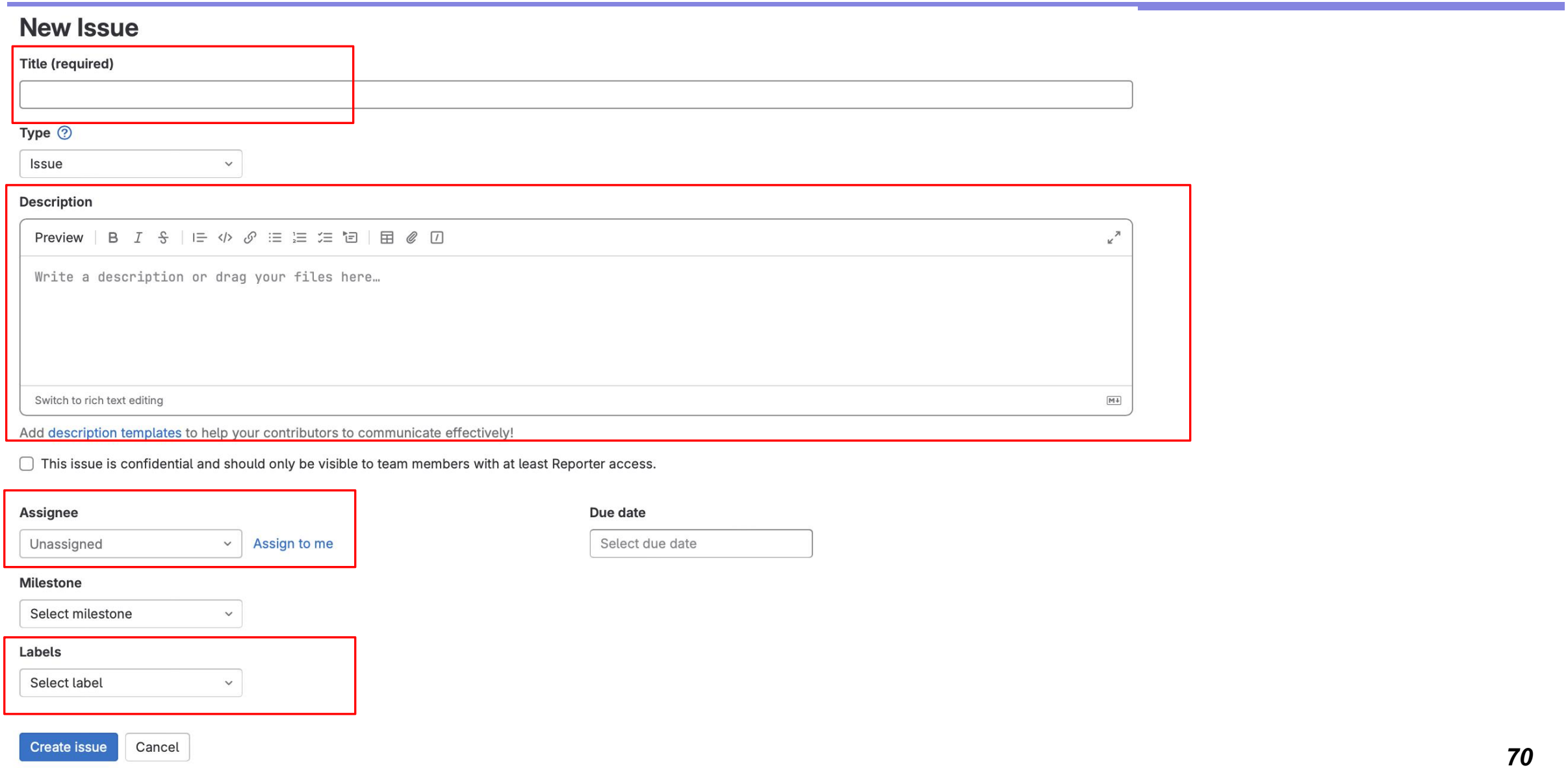
Plan

Code

Build

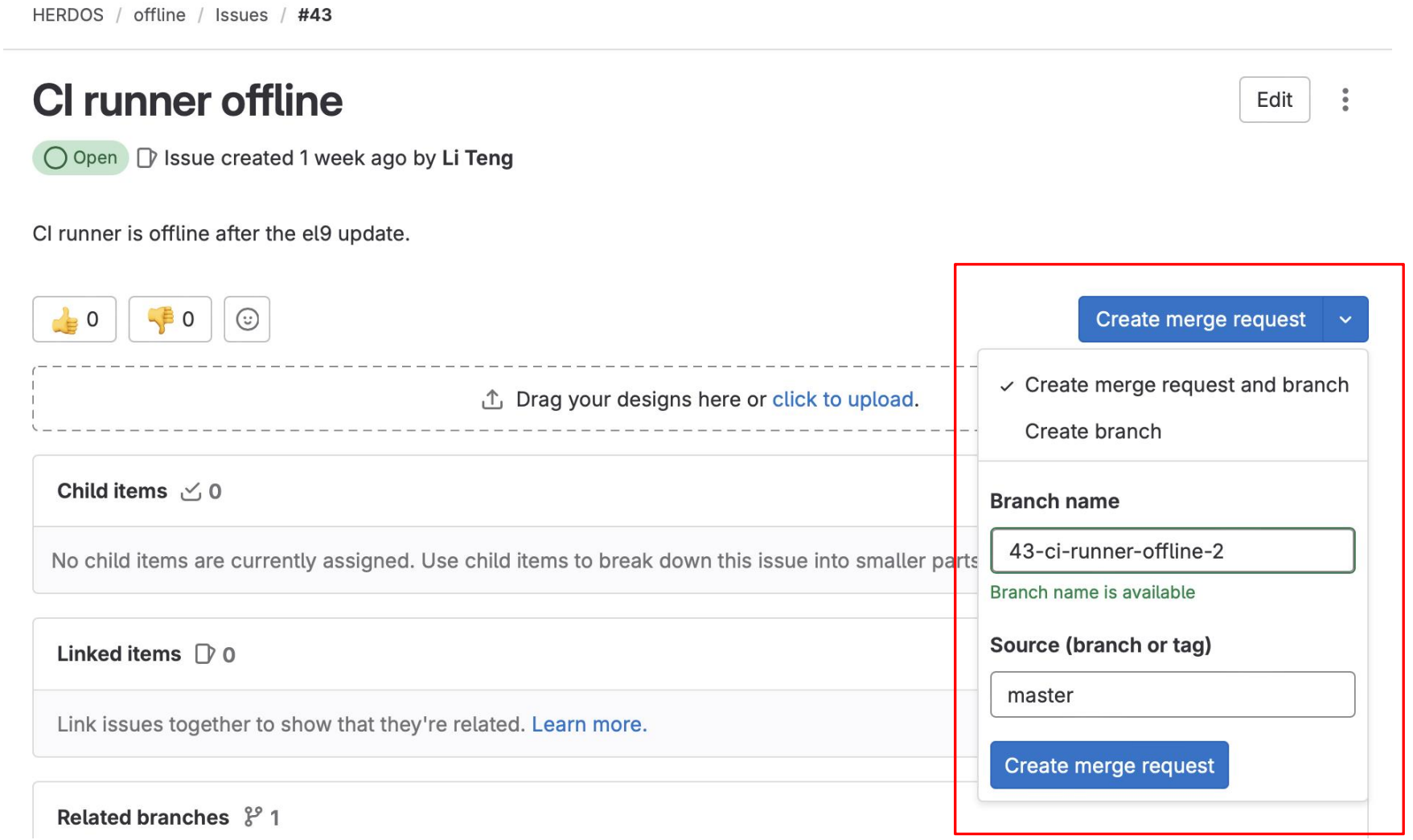
Secure

创建Issue

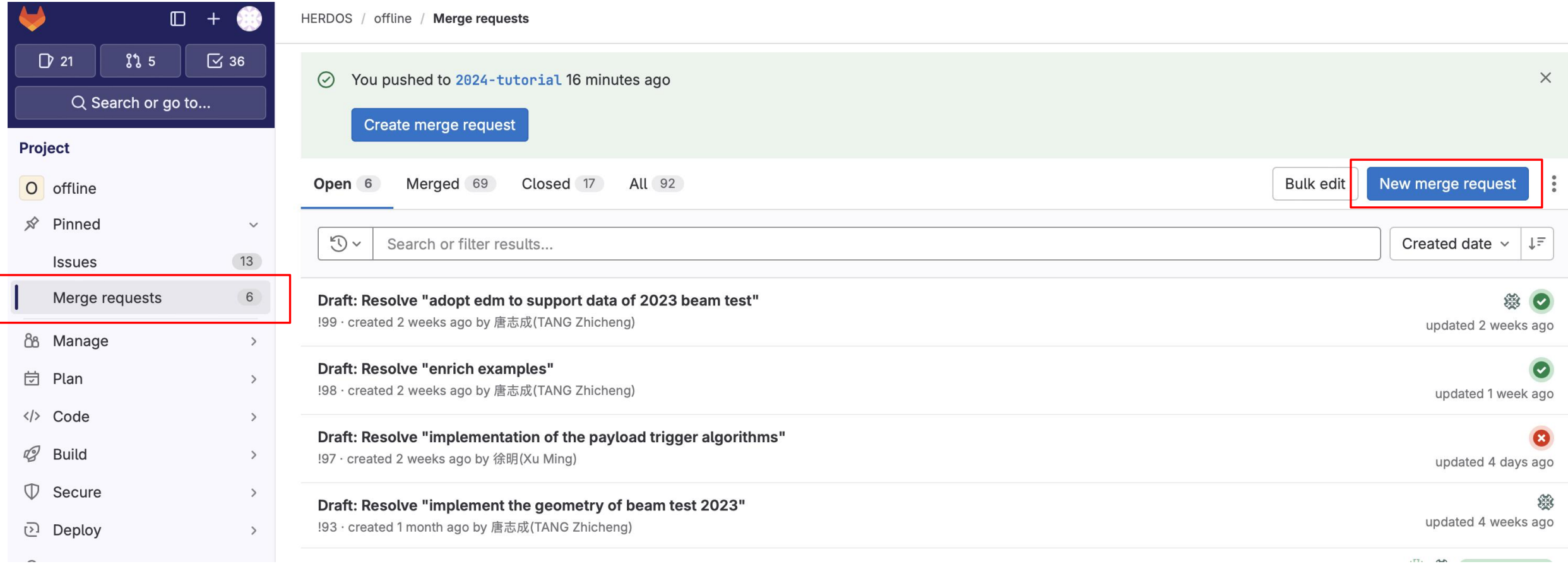

New Issue
Title (required)
Type
Issue
Description
Preview
Write a description or drag your files here…
Switch to rich text editing
Add description templates to help your contributors to communicate effectively!
This issue is confidential and should only be visible to team members with at least Reporter access.
Assignee
Unassigned
Assign to me
Due date
Select due date
Milestone
Select milestone
Labels
Select label
Create issue
Cancel

创建分支和MergeRequest

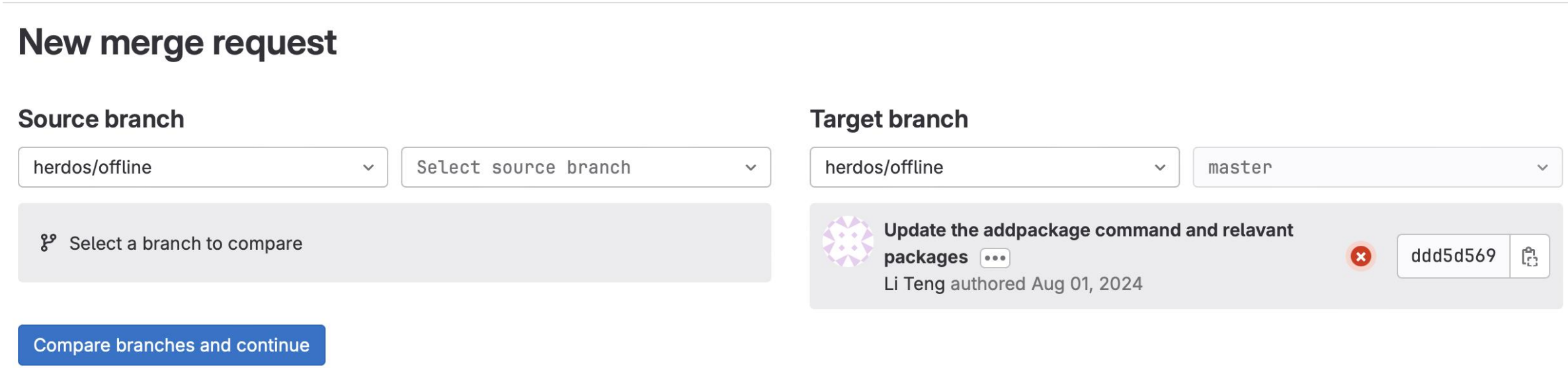
- 打开刚刚创建的issue



创建MergeRequest



创建MergeRequest



New merge request

From `2024-tutorial` into `master` Change branches

Title (required)

Mark as draft
Drafts cannot be merged until marked ready.

Description

Preview | B I S | </> | Describe the goal of the changes and what reviewers should be aware of.

Switch to rich text editing

Add description templates to help your contributors to communicate effectively!

Assignee

Unassigned Assign to me

Reviewer

Unassigned

Milestone

Select milestone

Labels

Select label

Merge options

- [x] Delete source branch when merge request is accepted.
- [x] Squash commits when merge request is accepted.

Create merge request Cancel

Commits 1 Pipelines 1 Changes 9

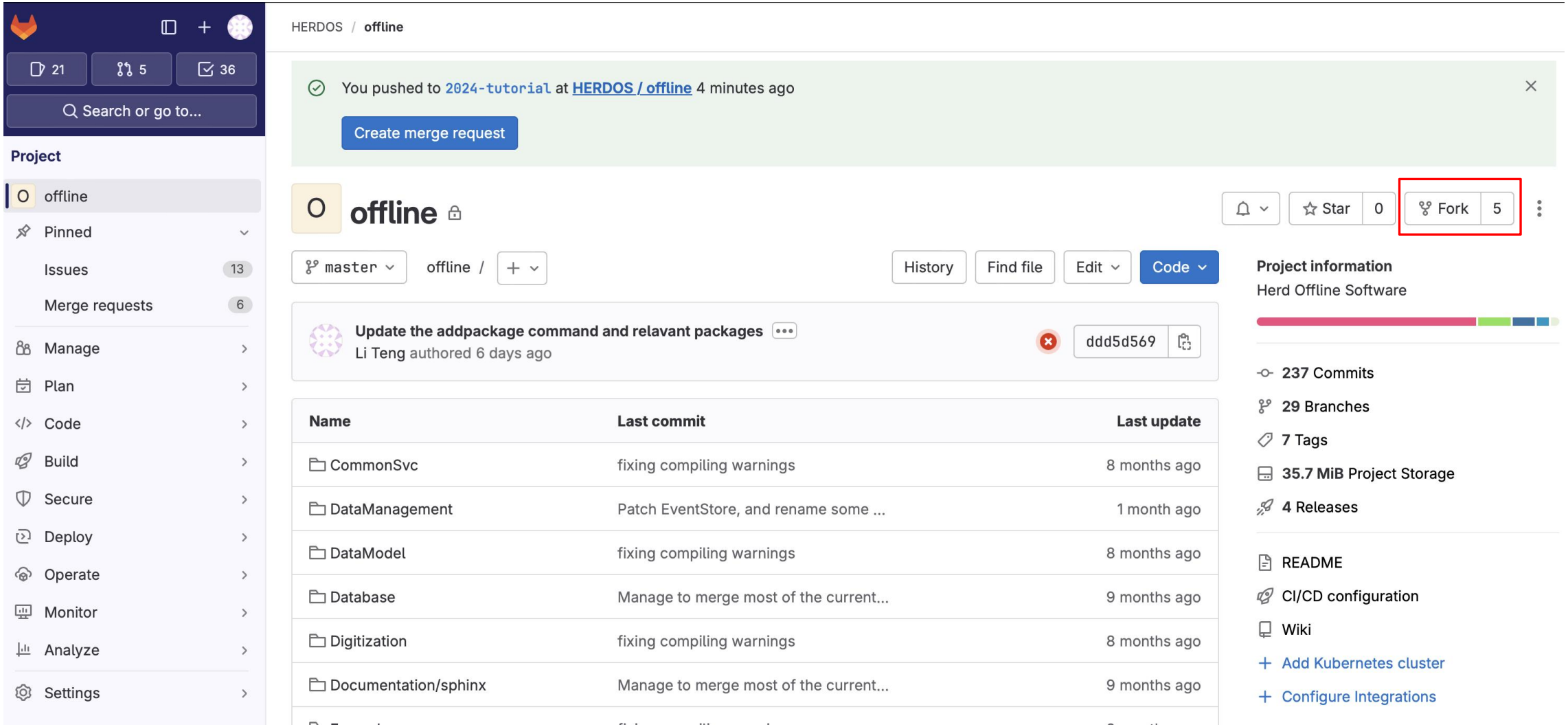
Aug 07, 2024

prepare code for tutorial
Li Teng authored 18 minutes ago dedf3c3c

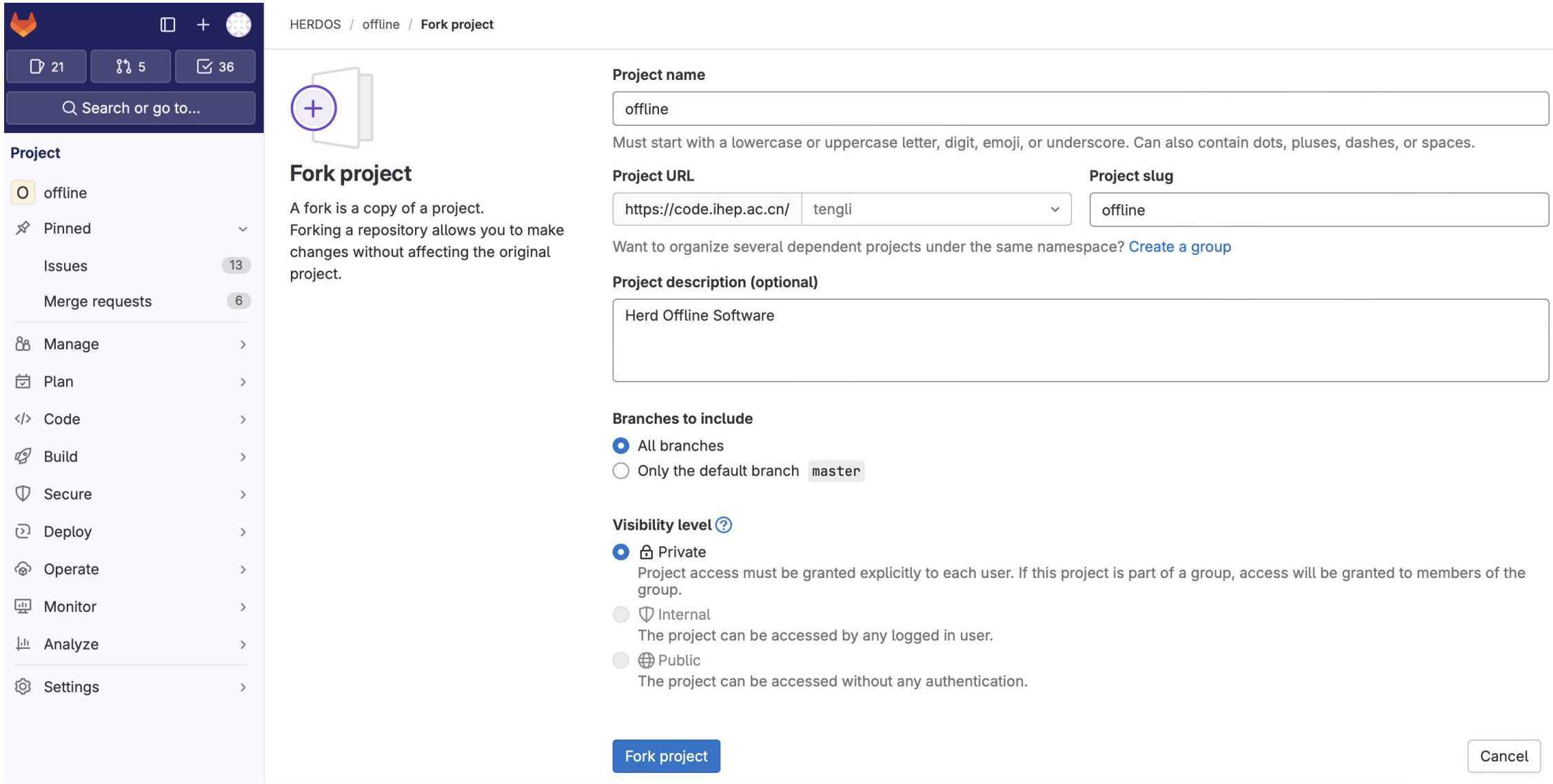
OSCAR开发规则

- 2. 开发
  - Master（主分支）不接受直接commit. 支持branch-mergeRequest和fork-branch-mergeRequest两种开发模式
  - branch-mergeRequest指的是先从master创建分支，在分支上开发代码，开发完毕后合并。这是建议的开发模式。
  - fork-branch-mergeRequest指的是先fork私人仓库，在私人仓库开发代码，开发完毕后合并到主仓库分支，检查审核完毕后再合并到master的开发模式。对整个OSCAR有较大影响的开发、大规模、长期的开发不要使用这种模式。
  - commit信息、issue中的讨论、代码注释等一律使用英文。commit信息要简要、精确。
  - 代码开发必须遵守代码规范
  - 新开发的功能、算法、工具等需要提供测试
  - 文档需要随代码一并更新、补充

创建fork仓库


HERDOS / offline
21
5
36
Search or go to...
Project
offline
Pinned
Issues 13
Merge requests 6
Manage
Plan
Code
Build
Secure
Deploy
Operate
Monitor
Analyze
Settings
You pushed to 2024-tutorial at HERDOS / offline 4 minutes ago
Create merge request
offline
Star 0
Fork 5
master
offline /
History
Find file
Edit
Code
Update the addpackage command and relavant packages
Li Teng authored 6 days ago
ddd5d569
Name
Last commit
Last update
CommonSvc
fixing compiling warnings
8 months ago
DataManagement
Patch EventStore, and rename some ...
1 month ago
DataModel
fixing compiling warnings
8 months ago
Database
Manage to merge most of the current...
9 months ago
Digitization
fixing compiling warnings
8 months ago
Documentation/sphinx
Manage to merge most of the current...
9 months ago
Project information
Herd Offline Software
237 Commits
29 Branches
7 Tags
35.7 MiB Project Storage
4 Releases
README
CI/CD configuration
Wiki
Add Kubernetes cluster
Configure Integrations

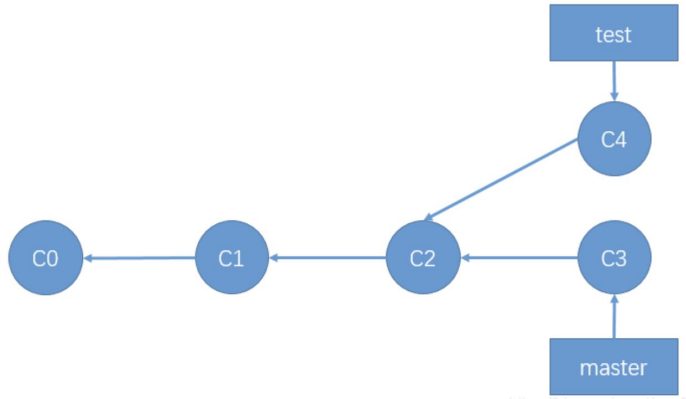
创建fork仓库


21
5
36
Search or go to...
Project
offline
Pinned
Issues 13
Merge requests 6
Manage
Plan
Code
Build
Secure
Deploy
Operate
Monitor
Analyze
Settings
HERDOS / offline / Fork project
Fork project
A fork is a copy of a project.
Forking a repository allows you to make changes without affecting the original project.
Project name
offline
Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.
Project URL
https://code.ihep.ac.cn/
tengli
Project slug
offline
Want to organize several dependent projects under the same namespace? Create a group
Project description (optional)
Herd Offline Software
Branches to include
All branches
Only the default branch master
Visibility level
Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.
Internal
The project can be accessed by any logged in user.
Public
The project can be accessed without any authentication.
Fork project
Cancel

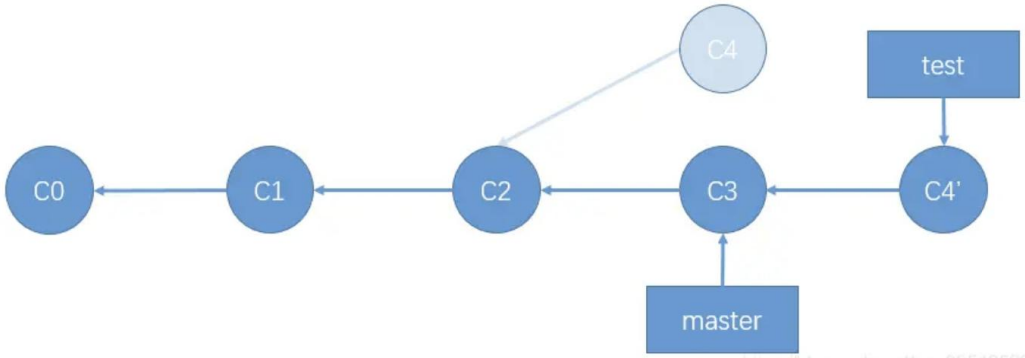
OSCAR开发规则

- 3. 合并
  - 在代码合并到master前，请保证1）所有的讨论thread已经结束；2）所有的代码冲突已经解决；3）所有的CI测试已经通过；4）相关子系统负责人同意。
  - 如果分支和master的代码已经互相偏离，并产生冲突，请使用变基（rebase）的方法解决冲突
  - 为了保证主干的历史简洁，合并代码时请勾选squash-commits，并使用fastforward（默认）方法避免产生merge commit
  - 分支之间的合并应尽量避免，除非确实需要
  - 每个MergeRequest需要指定负责人和审核人，负责人一般是相关开发者，审核人一般是子系统负责人。原则上负责人和审核人不是一个人。

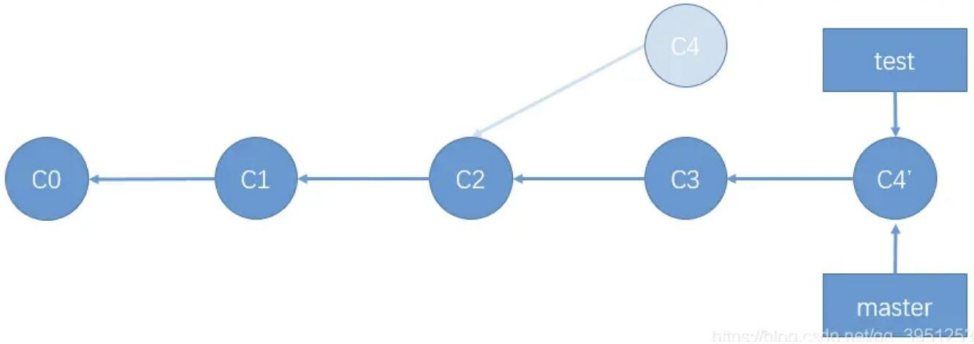
代码冲突和变基



变基前



变基后



合并后

代码冲突和变基

1. Open a terminal and change to your project.
2. Ensure you have the latest contents of the target branch. In this example, the target branch is `main`:

```
git fetch origin main
```

3. Check out your branch:

```
git checkout my-branch
```

4. Optional. Create a backup of your branch:

```
git branch my-branch-backup
```

   Changes added to `my-branch` after this point are lost if you restore from the backup branch.

5. Rebase against the main branch:

```
git rebase origin/main
```

6. If merge conflicts exist:
   a. Fix the conflicts in your editor.
   b. Add the files:

```
git add .
```

   c. Continue the rebase:

```
git rebase --continue
```

7. Force push your changes to the target branch, while protecting others' commits:

```
git push origin my-branch --force-with-lease
```

详细解释请见：
https://docs.gitlab.com/ee/topics/git/git_rebase.html

常用的Git命令

- git clone：下载仓库
- git checkout: 切换分支
- git add：将更改过的文件加入工作区
- git mv：重命名、移动文件
- git rm：删除文件
- git restore：恢复
- git status：查看工作区状态
- git commit：提交更改
- git pull：更新本地仓库
- git push：推送本底代码到远程仓库

运行git -h 查看全部子命令
运行git 子命令 -h查看详细说明

一般的Git工作流程

- 第一步：创建issue和Branch
- 第二步：下载代码，选择分支
  - git clone https://git.ustc.edu.cn/oscar1/offline.git
  - git checkout branch_name
- 第三步：开发代码
- 第四步：Stage the changes
  - git add
  - git rm
  - git mv
- 第五步：提交代码
  - git commit -m “I have changed xxxxxx”
- 第六步：推送远程仓库
  - git push
- 第七步：提醒、等待审核者审核和反馈；修改代码或合并分支。

二、代码规范

代码规范

- https://git.ustc.edu.cn/oscar1/offline/-/blob/main/Documentation/sphinx/source/appendix/appendix.md
- 算法包规范
  - 算法包命名使用Pascal Case命名方法。例如：DetectorSimulation, DetSim, ECALAna, CaloPCA
  - 算法包目录结构：
    - CMakeList.txt：编译规则
    - <PackageName>/: 需要开放给其它包的头文件: *.h
    - src/: 源文件: *.cc；不需要开放的头文件。
    - python/: python代码
    - share/: 运行脚本、测试脚本
    - compact/: DD4hep定义的几何
    - g4macro/: GEANT4 macro文件 *.g4mac

代码规范

- Namespace: 小写的短字符串
- 类名：Pascal Case命名方法，如ECALAnaAlg
- 变量名：
  - 按照public, protected和private的顺序排列
  - 普通类成员以m_开头，如m_energy
  - 静态变量以s_开头，如s_count
  - 常量以k开头，如kGainRatio
  - 全局变量以g开头，如gEnv
- 函数名：
  - 函数名：Camel Case: setEnergy()
  - 局部变量和函数参数：Camel Case：nbytes,recoEnergy

代码规范

- 其它：
  - 一个header内通常只定义一个类，处类内部成员所需额外的类/结构体
  - header需要防止重复编译：
    - #pragma once
    - 或者

```
#ifndef NAMESPACE_CLASSNAME_H
#define NAMESPACE_CLASSNAME_H
...
#endif//NAMESPACE_CLASSNAME_H
```

  - inline函数定义在头文件中
  - 缩进：建议使用四个空格，不要使用tab；花括号{ }单独占一行
  - 成员变量需在构造函数赋初始值

代码规范

- 代码注释：
  - 单行注释：//
  - 多行注释：

```
/**
 *
 *
 */
```

  - 每个头文件顶部须注释类的描述（多行注释，包括简要功能介绍、作者和email）
  - 每个成员函数和变量声明时须用单行注释简要描述
  - 每个函数在定义时用注释简要描述其功能、返回值和参数。建议根据Doxygen规则书写注释，以自动产生reference documentation
    - https://www.doxygen.nl/manual/index.html