**深 圳 大 学 实 验 报 告**

**课程名称： 软件工程**

**实验项目名称： 实验5 敏捷开发平台配置及使用**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 卢亚辉**

**报告人： 郑雨婷 学号： 2021150122 班级： 郑雨婷**

**实验时间：2023年11月10日（周五）~2023年11月26日（周日）**

**实验报告提交时间： 2023年11月20日（周日）**

**教务部制**

一、实验目的：

（1）了解DevOps的概念；

（2）了解DevOps的开发方法；

（3）了解如何使用Gitlab进行Devops开发；

二、实验内容：

1. 进入网址https://jihulab.com/users/sign\_up注册Gitlab账号，输入姓名、用户名、邮箱、密码等，注册成功。



1. 注册成功之后，进入https://jihulab.com/users/sign\_in登陆。登陆成功之后申请Saas的30天免费体验。

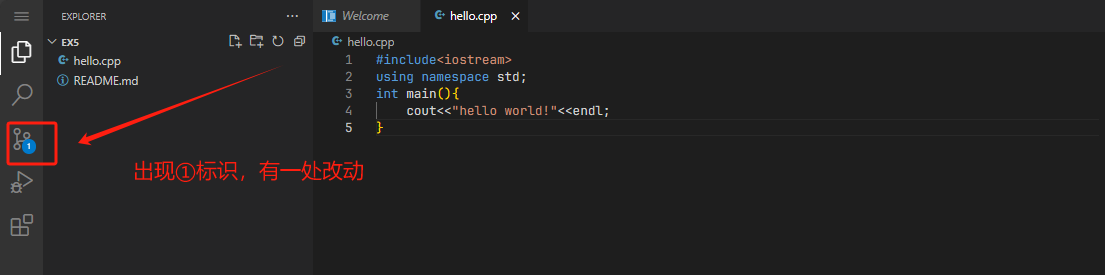


（3）新建一个空白项目，命名为ex5在左侧边栏中，找到“学习Gitlab”，按照要求一步步学习.

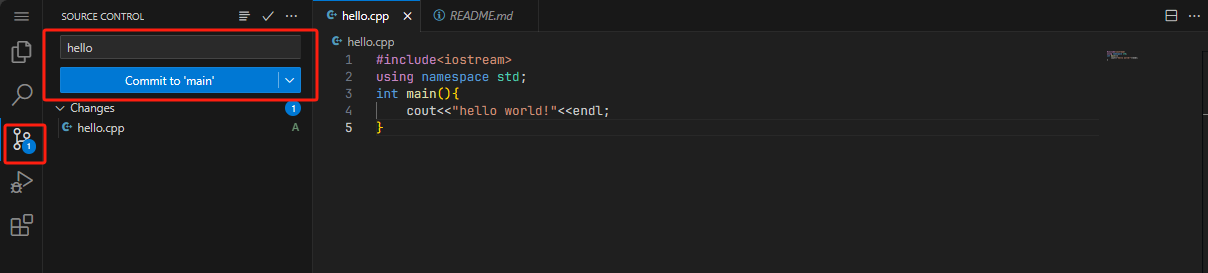


①将代码添加到项目，从使用 WebIDE 开始，使用内置编辑器创建或上传文件。

打开WebIDE,新建一个hello.cpp文件。



左侧的source control显示出一个改动，点击进入，在上方文本框中输入本次提交的注释信息，然后点击“commit to ‘main’”,即可提交至项目中。



②设置工作区

创建一个仓库，在步骤（3）中已经创建成功，创建了名为ex5的仓库。免费使用专业版也在步骤（2）中成功申请。

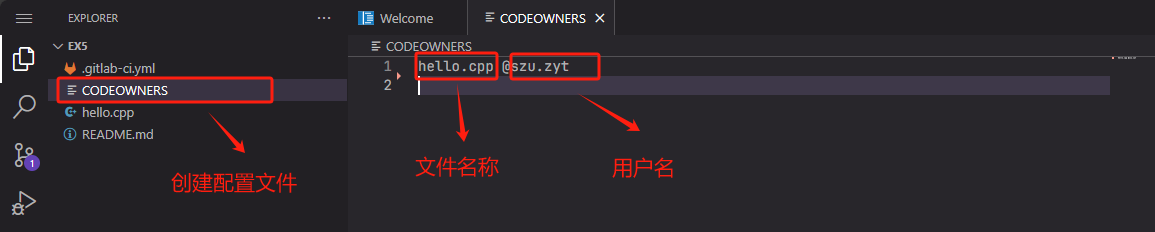
邀请同事：首先按照步骤（1）-（2）创建一个用户名为szuzyt的小号，在最上方填入小号的用户名，点击右下角“邀请”，即邀请成功。也可以选择用户的身份（Guest、Reporter、Developer、Maintainer、Owner）以及访问到期日期，我将小号设置为Guest且不设置访问到期日期。



设置第一个项目的 CI/CD：因为我们之前提交的代码是hello.cpp，是一个简单的C++程序，我们选择C++的模板创建一个 .gitlab-ci.yml 文件。



添加代码所有者：创建一个CODEOWNERS文件，在文件中配置代码的所有者。我将自己添加为hello.cpp的代码所有者。点击hello.cpp查看详情，可以看到hello.cpp的代码所有者，添加成功！



启用需要合并批准：为要包含在整体覆盖率值中的所有作业设置 coverage 正则表达式。在项目中选择 设置 > 合并请求。在 合并请求批准 下，选择 Coverage-Check 批准规则旁边的 启用。

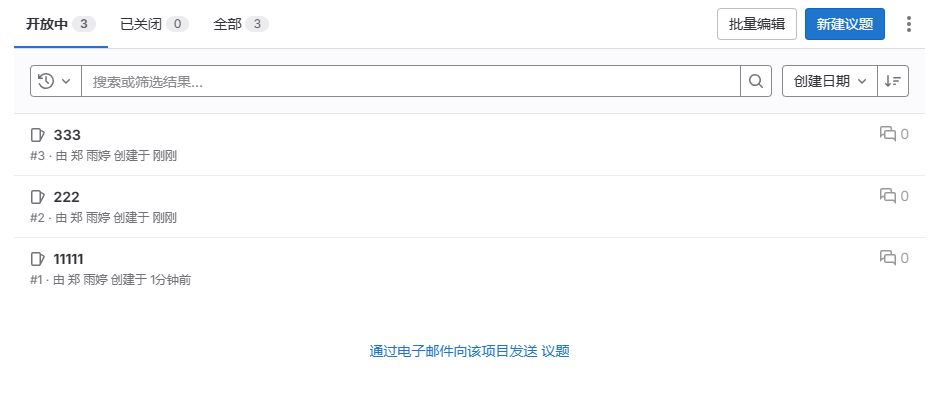


选择 目标分支。将 需要批准 的数量设置为大于零。选择要提供批准的用户或群组。选择 添加批准规则。

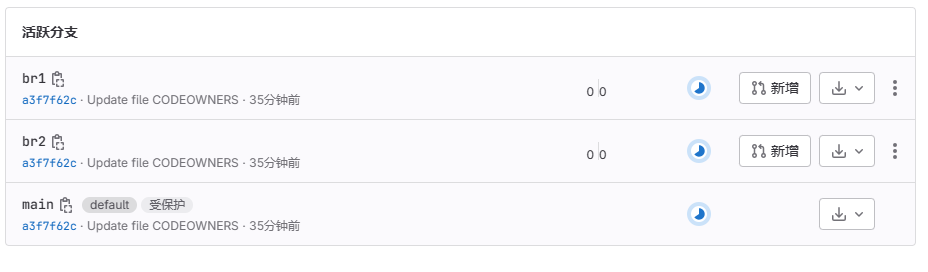


③计划和执行，为您的新工作区创建工作流，并了解 GitLab 功能如何协同工作：

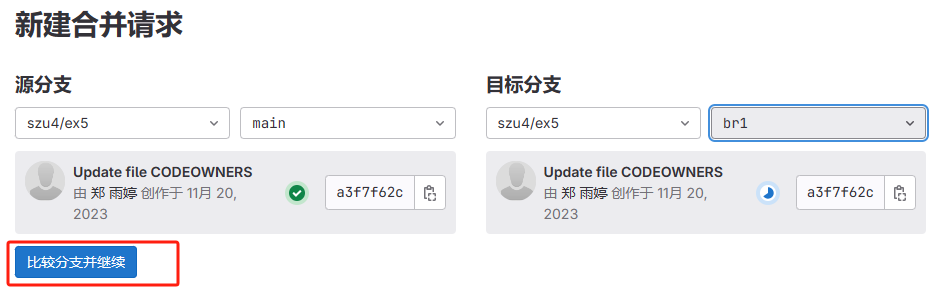
创建议题：在新建议题中填写标题和描述，在实际企业中会用到该项技术。新建议题后项目中的所有成员都可以看到，方便提前准备。



提交合并请求 (MR)：合并请求用于提出对项目的更改并与他人进行讨论。用户可以通过推送提交来对项目作出贡献。在合并前需要有不止一个分支，因此先创先两个新分支br1和br2。此时共有一个mainbr1、br2这三个分支。



新建合并分支时源分支选择main,目标分支选择br1。



（4）通过CI/CD开发部署系统

更改.gitlab-ci.yml 文件的内容并推送到GitLab，这份 .gitlab-ci.yml 文件定义了一个包括三个阶段的CI/CD工作流。首先是构建阶段（build），其中通过g++编译hello.cpp文件，并将生成的可执行文件命名为 artifact1，然后将其设置为Artifact，以便在后续阶段中使用。接下来是测试阶段（test），在该阶段执行 ./artifact1命令，测试生成的可执行文件。最后是部署阶段（deploy），在此阶段执行一个简单的命令，输出 "Deploying artifact1"，并且该阶段依赖于构建阶段的完成。这个CI/CD流水线的目标是通过自动化流程完成代码的构建、测试和部署，确保代码在不同阶段的正确性和一致性。

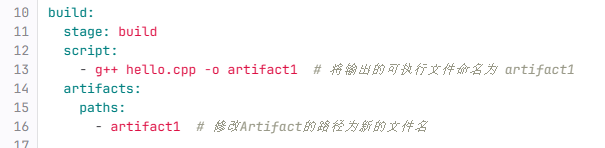


之后就可以查看最近的提交的状态：

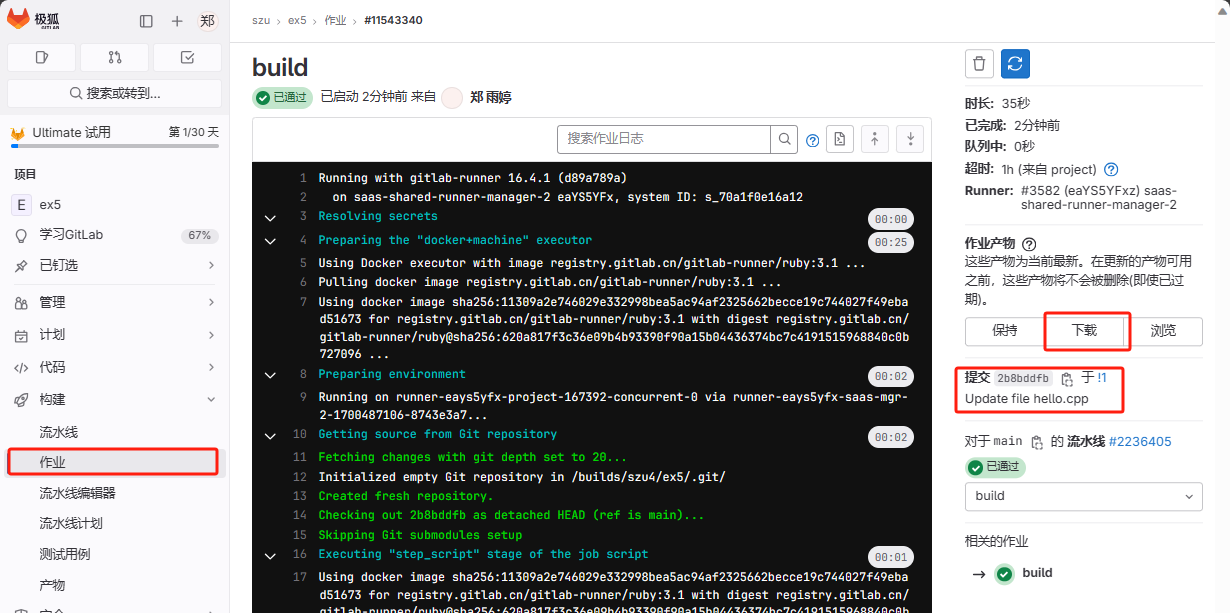


(5) 执行Build app，得到build的结果，提交artifact1。

这段代码定义了一个构建阶段（build stage）的配置，其中使用了`g++`编译`hello.cpp`文件，并将生成的可执行文件命名为`artifact1`。此阶段的目标是生成一个Artifact（构建产物），即`artifact1`文件，以便在后续阶段中引用和使用。Artifact是构建过程中产生的文件，可以在CI/CD流水线的不同阶段之间传递和共享。



在左侧创建中“作业”中选择hello.cpp的build，点击右侧的作业产物，即可以将生成的artifact1保存至本地。



（6）创建新的项目

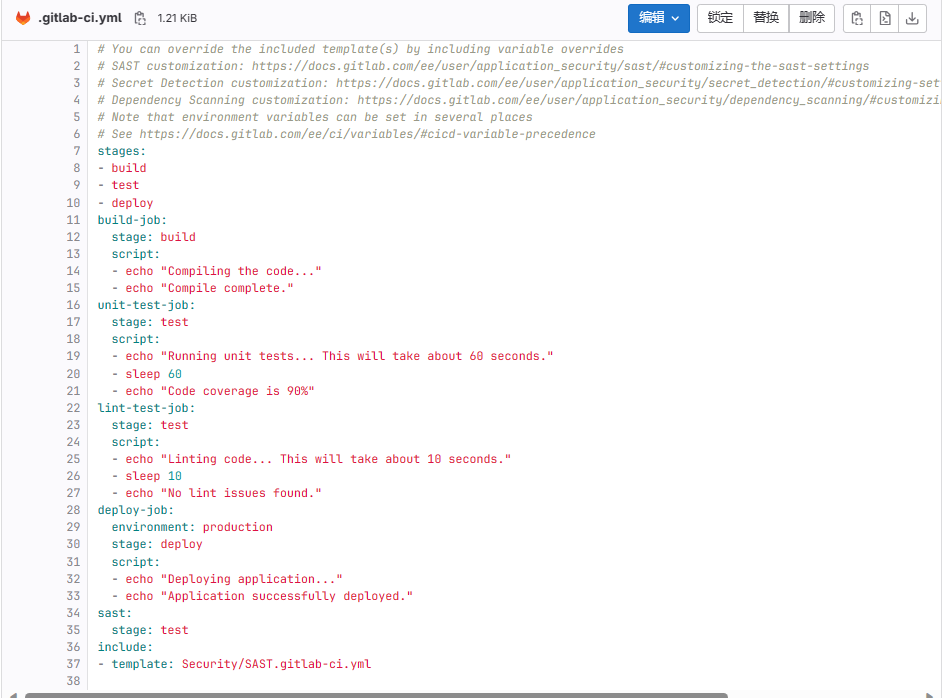
Auto DevOps 提供了预定义的CI/CD配置，使你可以自动检测，构建，测试，部署和监视应用程序。借助CI/CD最佳实践和工具，Auto DevOps旨在简化成熟和现代软件开发生命周期的设置和执行。只需推送你的代码，GitLab就会处理其他所有事情。这使得启动新项目更加容易，并使整个公司的应用程序设置方式保持一致。

①选择从模板中创建，选择**Ruby on Rails, 命名为new\_project.**



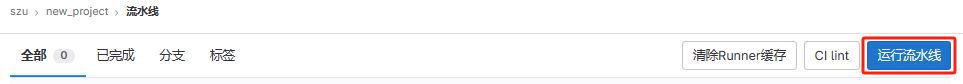
②**查看其.gitlab-ci.yml文件**

这段 `.gitlab-ci.yml` 文件定义了一个包含四个阶段的CI/CD工作流。首先是构建阶段（`build`），其中通过两个 `echo` 命令模拟编译代码的过程。接着是两个测试阶段（`test`），分别是单元测试（`unit-test-job`）和代码风格检查（`lint-test-job`）。在单元测试阶段，通过 `sleep` 模拟运行单元测试需要约60秒的时间，并输出测试覆盖率为90%。而在代码风格检查阶段，同样通过 `sleep` 模拟检查需要约10秒的时间，并输出未发现代码风格问题。最后是部署阶段（`deploy`），在该阶段模拟部署应用程序的过程，并将该阶段设置为生产环境。此外，还包含了一个名为 `sast` 的特殊阶段，它通过 `include` 引入了GitLab提供的安全性分析模板（`Security/SAST.gitlab-ci.yml`），用于执行静态应用程序安全性测试（SAST）。整体而言，这个CI/CD流水线通过模拟各个阶段的基本操作，展示了从构建到测试再到部署的完整流程，并集成了静态应用程序安全性测试。

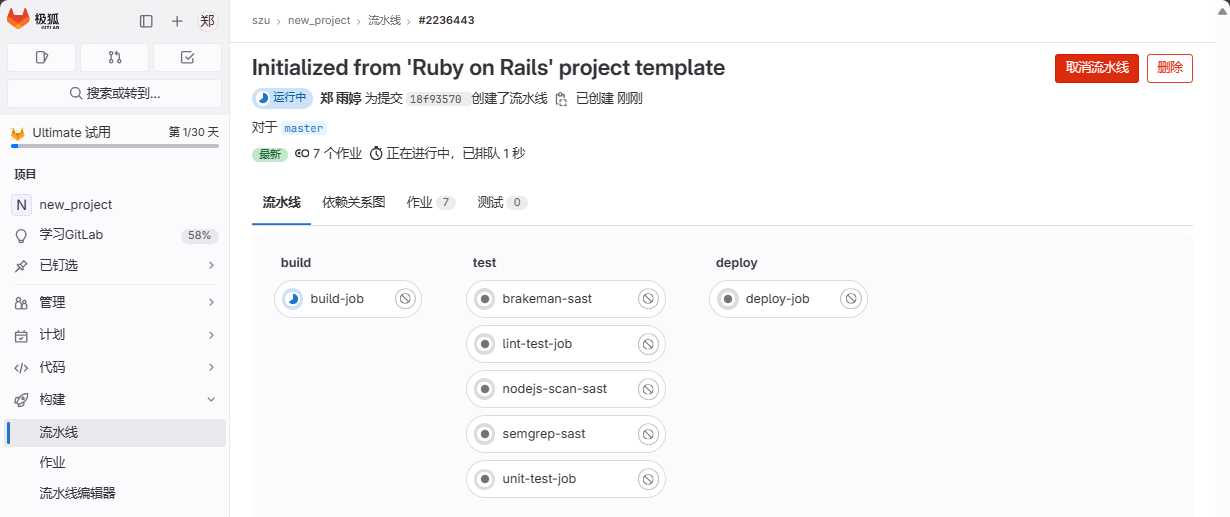


③**运行流水线构建**

通过在GitLab左侧构建—>流水线中，点击运行流水线，所有设置为默认设置。点击“运行流水线”我们手动触发了CI/CD流水线的构建过程。

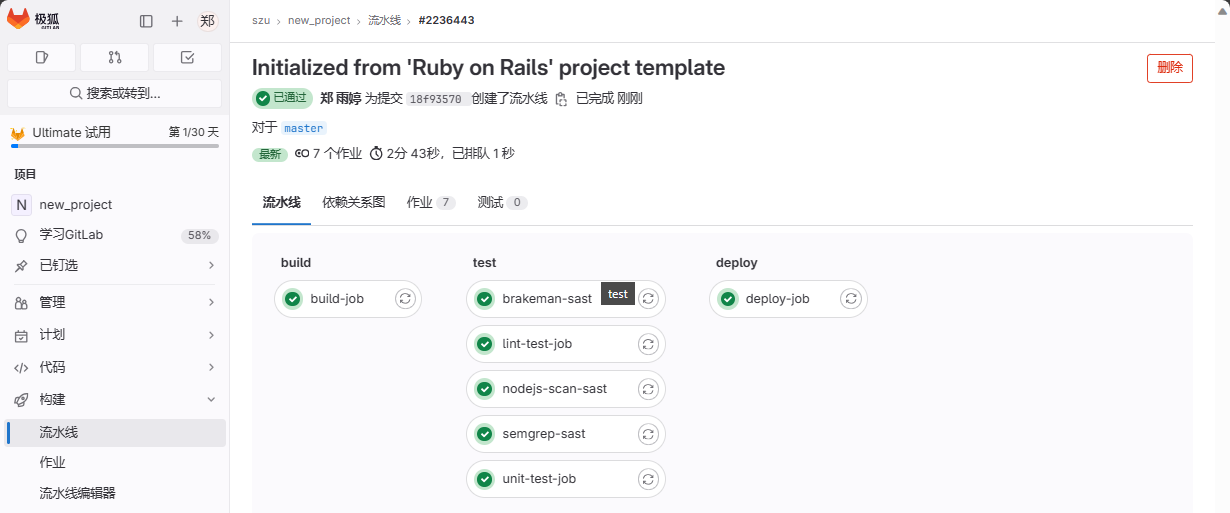


管道内部分为3个阶段，我们可以查看每个阶段有几个作业在运行，如下图：构建 -> 测试 -> 部署, 其中构建阶段有一个作业、测试阶段有五个作业、部署阶段有一个作业。通过监视每个作业的执行状态，我们确认了整个流水线的成功完成。这个实验演示了GitLab CI/CD的能力，帮助我们自动化软件开发过程，提高效率和代码质量。

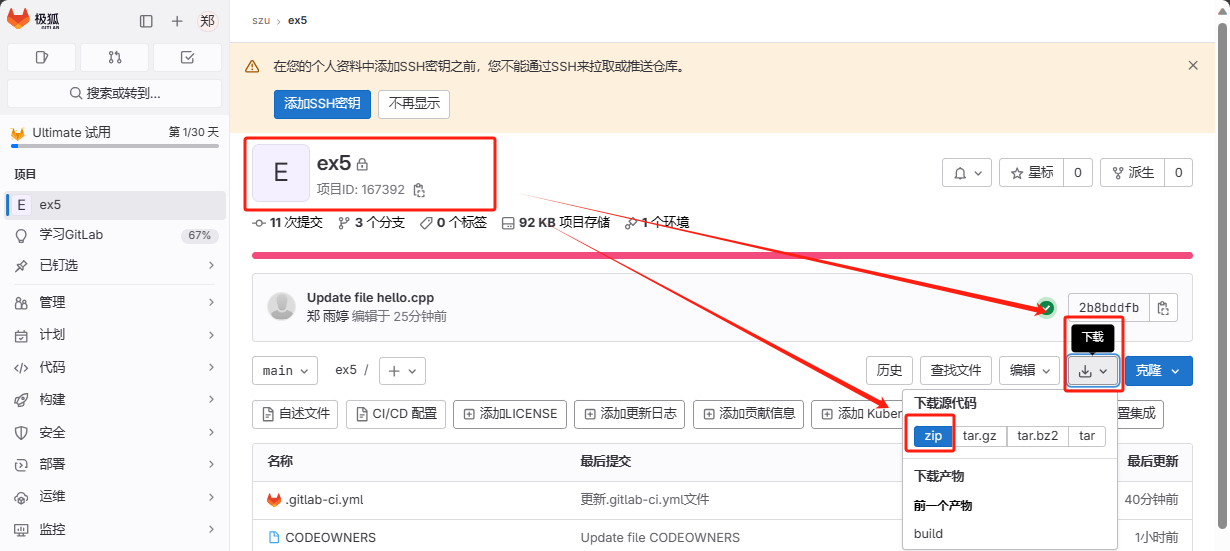


在大约三分钟的等待后，我们成功地完成了包含七个作业的流水线。

三、提交内容



1. 实验报告
2. build app得到的artifact
3. [learn-gitlab-ultimate-trial](https://jihulab.com/szu1/learn-gitlab-ultimate-trial/-/tree/master/)的文件打包下载，我的项目名称为ex5，因此下载的文件为ex5-main.zip.



四、实验心得

在完成这次实验的过程中，我通过一系列步骤全面了解了GitLab平台的基本使用以及如何在其中进行DevOps开发。首先，通过注册GitLab账号并登录成功，我顺利申请了Saas的30天免费体验，为后续的实验提供了一个稳固的基础。在新建空白项目的过程中，我通过WebIDE创建了一个简单的C++程序，并成功提交到项目中。

在设置工作区的步骤中，我创建了一个名为ex5的仓库，并成功邀请了一个小号作为协作者。此外，通过创建CODEOWNERS文件，我成功配置了代码的所有者，并在合并请求中启用了合并批准功能，以确保代码质量和审查。

在学习GitLab的过程中，我通过创建议题和提交合并请求，深刻理解了这些功能在实际企业中的应用场景。通过这些工具，团队成员能够更好地协同工作、提前准备，为项目的进展提供了可视化的管理方式。

进入CI/CD开发部署系统的阶段，我修改了.gitlab-ci.yml文件，定义了一个包含三个阶段的CI/CD工作流。这个流水线通过自动化构建、测试和部署过程，确保了代码的正确性和一致性。通过执行Build app，得到build的结果，并成功提交artifact1，我更深入地体验了CI/CD流水线的实际应用。

最后，通过创建新的项目并查看其.gitlab-ci.yml文件，我了解了Auto DevOps提供的预定义CI/CD配置，以及如何借助这些配置实现从检测到部署的全自动化流程。通过手动运行流水线构建，我亲身经历了整个流程，验证了这种DevOps方法对于项目启动和持续交付的便捷性和高效性。

总体而言，这次实验不仅让我熟悉了GitLab平台的基本操作，还深入理解了DevOps的核心思想和CI/CD流水线的实际应用。通过这些实践，我对于如何在开发过程中利用GitLab实现高效协同和持续交付有了更清晰的认识。

深圳大学学生实验报告用纸

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2021年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。