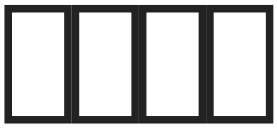


git.nju.edu.cn

- [redacted]
- Git[redacted]
- CI/CD[redacted] Docker[redacted]

Gravatar

Reply by email

Advanced Search

Container Registry

Shared Runners

- Linux Docker

Git[illegible]

NJU Git[illegible]

GitHub[illegible] GitLab[illegible] Gitee[illegible]

[illegible] https://mp.weixin.qq.com/s/EjAedt6A3PvuASGlFCfyWQ

CI/CD自动化构建 Docker镜像

> 一个通过 CI/CD自动构建 Docker镜像并用于 HPC集群计算的简单示例。

示例代码仓库: https://git.nju.edu.cn/escience/singularity-example

容器化在集群上使用时有很大的优势，主要体现在:

- 容器内模拟的 root权限，无需权限即可任意安装自己想要的程序包
- 版本可控，自定义性强
- 可复用性强，一处构建，多处部署

因此，对于一些编译特别复杂但早已有 Docker镜像的计算软件，或者自己写的配置环境麻烦的软件，考虑 Singularity是个不错的选择。

此外， eScience中心的多个服务也涵盖了 一整套工作流 。所以全程无需出校访问 在校园内网 即可实现。

本文将通过一个简单但常见的 Conda虚拟环境下 Python Numpy库的镜像构建来演示如何使用这一功能。

通过 Dockerfile 构建自己的镜像

通过 Dockerfile，可以事先构建一个 Docker镜像。

```
FROM continuumio/miniconda3:22.11.1

# 使用南大镜像站 conda源
COPY .condarc /root/.condarc

# 创建环境
RUN conda create -n my-env python=3.10 numpy

# 激活环境
SHELL ["/bin/bash", "--login", "-c"]
RUN conda init bash
RUN echo "source activate my-env" > ~/.bashrc
ENV PATH /opt/conda/envs/my-env/bin:$PATH
```

此处 `.condarc` 是修改为了南京大学镜像站的软件源以保证速度。如果要安装其他的包，把 `numpy` 改成你要的包即可，同时还可以控制 Python版本。

如若本地有 Docker，可通过 `docker` 本地尝试构建：

```
docker build -t escience/conda-numpy .
```

> Docker也可以使用 Docker缓存 `docker.nju.edu.cn` 。

构建完毕后，本地跑一下相关的脚本以测试：

```
docker run -i escience/conda-numpy python < test.py
```

设置 CI/CD

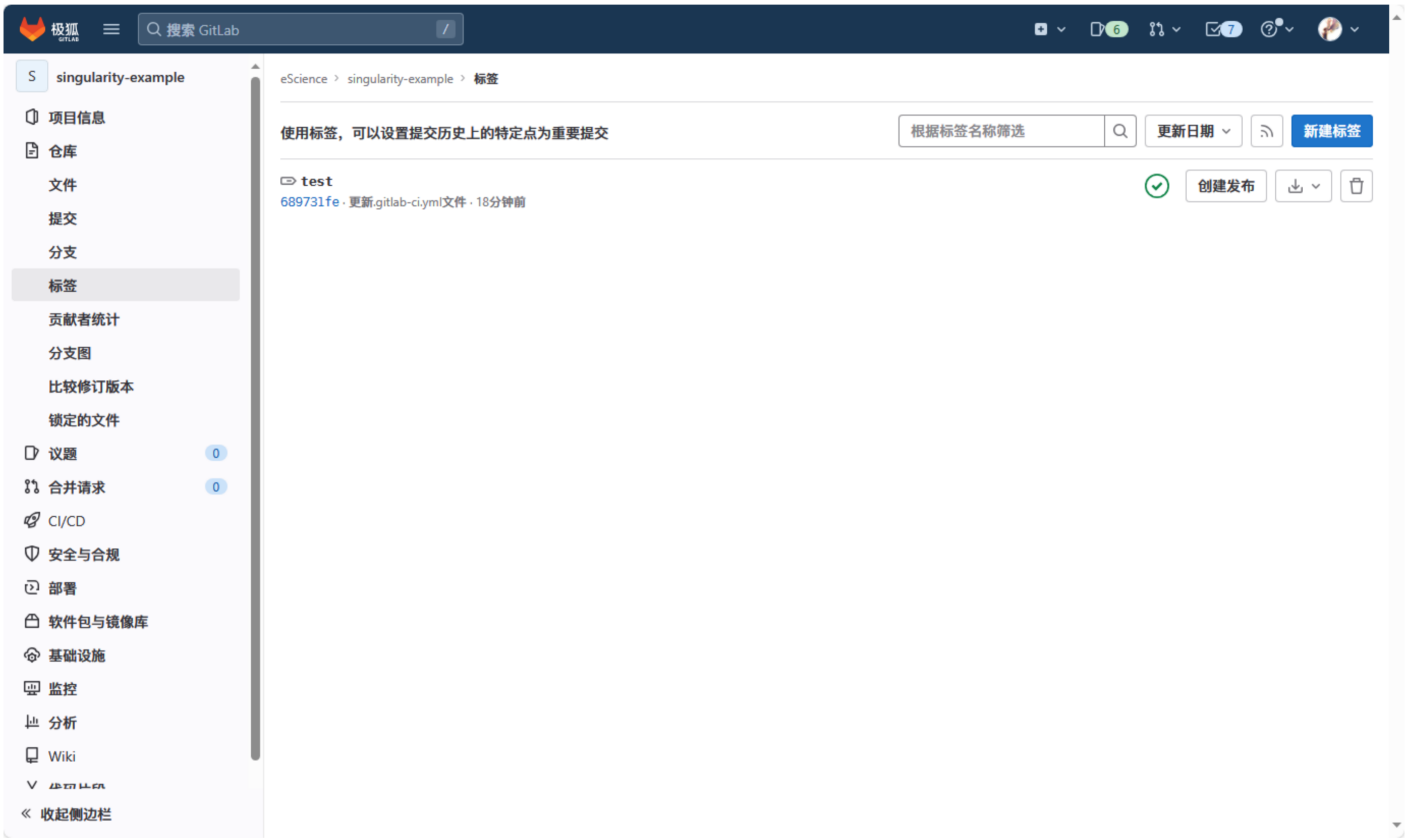
CI/CD是代码托管服务 `git.nju.edu.cn` 的一个自动化工具，在代码仓库下的 `.gitlab-ci.yml` 文件中配置 CI/CD，可以让服务器按照设定执行自动构建、编译、测试、集成、部署等等。

CI/CD需要一个基础镜像来运行，这个镜像 `gcr.nju.edu.cn` 也有缓存服务。文件内容如下：

```
# 执行顺序
stages:
  - build
  - test
# 自动构建镜像
build:
  stage: build
  image:
    name: gcr.nju.edu.cn/kaniko-project/executor:debug # 南大 gcr缓存加速
    entrypoint: [""]
  script:
    - /kaniko/executor
      --context "${CI_PROJECT_DIR}"
      --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
      --destination "${CI_REGISTRY_IMAGE}:${CI_COMMIT_TAG}"
  rules: # 只在推送标签时触发
    - if: $CI_COMMIT_TAG
# 使用文件来测试
```

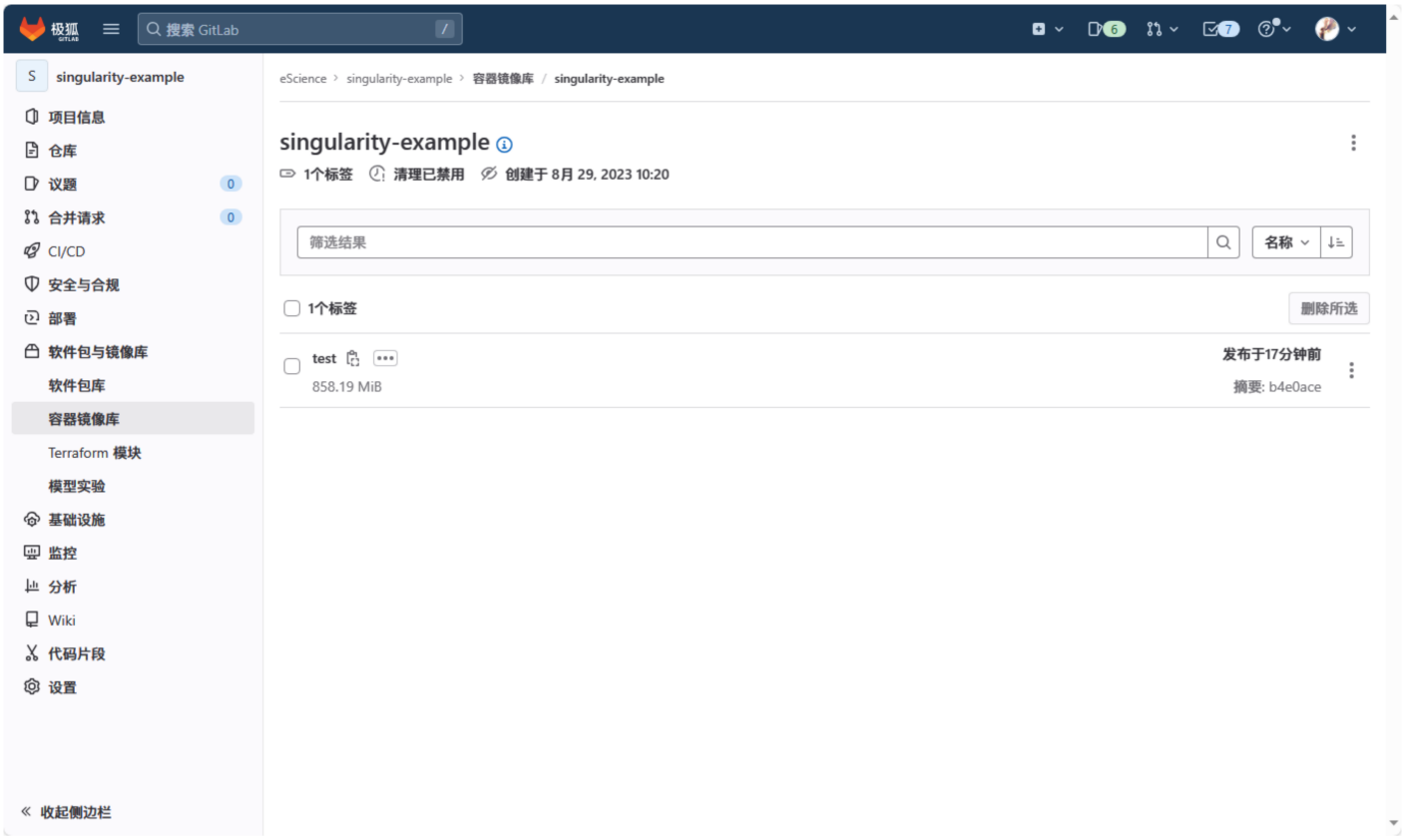
```
test:
  stage: test
  image:
    name: ${CI_REGISTRY_IMAGE}:${CI_COMMIT_TAG}
  script:
    - python "${CI_PROJECT_DIR}/test.py"
  rules:
    - if: $CI_COMMIT_TAG
```

推送代码到仓库后，为某个提交创建标签，例如此处我们创建为 `test`，这也会成为镜像的标签。



触发流水线任务后，如果通过，在 **“CI/CD”-“流水线”** 可以看到作业情况：

在自动构建成功后，即可在 **“软件包与镜像库”-“容器镜像库”** 中查看到此镜像。



> 当然，这里你也不一定必须用 Kaniko 的构建方案，此处是为了实现自动化。你也可以本地构建完毕后通过 `docker push` 命令直接推送至 `reg.nju.edu.cn` 来托管。此外，Singularity有其自己的 定义文件 (类似 Dockerfile)，可以用类似的方式一步到位而不需要单独构建一个 Docker镜像。

在集群上使用

`hpc.nju.edu.cn` 唐楼集群的简单基本使用见 《校内用户超算集群申请与基本使用简明指南》一文。

登录集群通过此镜像来构建 Singularity镜像:

```
singularity build conda-numpy.sif docker://reg.nju.edu.cn/escience/singularity-example:test
```

```
INFO:    Starting build...
Getting image source signatures
Copying blob 3389e9eb8624 done
Copying blob 3f4ca61aafcd done
Copying blob baee49be4542 done
Copying blob 7b4354700ca4 done
Copying blob 69a5d9e1ecd6 done
Copying blob 8a7a15cee421 done
Copying blob 5f4e24b7e321 done
Copying config 51871566f8 done
Writing manifest to image destination
Storing signatures
2023/08/29 21:46:55  info unpack layer: sha256:3f4ca61aafcd4fc07267a105067db35c0f0ac630e1970f3cd0c7bf552780e985
2023/08/29 21:46:57  info unpack layer: sha256:69a5d9e1ecd6566da53d0978004bdf37dddfaba1d8a6117966f397b41cbbc529
2023/08/29 21:46:59  info unpack layer: sha256:7b4354700ca480732ead22a553cc45916dc5466709ca64d964c4647b5b9343e9
2023/08/29 21:47:02  info unpack layer: sha256:baee49be454261f20f9770566da694b7e7845cf7d279cc2421c6b3eed68c012c
2023/08/29 21:47:02  info unpack layer: sha256:8a7a15cee4219b244df17b401664119dfd5b7e52a5659107f1ee3ec210722373
2023/08/29 21:47:20  info unpack layer: sha256:3389e9eb8624f21330c272ba23defc59ba31d46273e38cde475e7256edea80cb
2023/08/29 21:47:20  info unpack layer: sha256:5f4e24b7e32113995e0417eb5c4de5cbef9be70a4a3841a19cdb2fa1f3f12a34
INFO:    Creating SIF file...
INFO:    Build complete: conda-numpy.sif
```

虽然镜像较大，但是可以看见速度仍然很快，这正是开头 全程校园内网 带来的好处。

由于示例是一个 `numpy` 的环境，后续我们对应地使用作业脚本即可。创建一个 `job.lsf` 文件：

```
#BSUB -q 6140ib
#BSUB -n 1

module load singularity/latest

SINGULARITY="singularity run --env MKL_NUM_THREADS=$LSB_DJOB_NUMPROC conda-numpy.sif"
${SINGULARITY} python test.py
```

然后执行

```
bsub < job.lsf
```

通过 `bjobs` 查看任务运行情况，完成后通过 `bpeek` 指令可以查看输出，可以看到，任务成功提交并正确执行了。

总结

通过 CI/CD自动构建 Docker镜像并用于 HPC集群计算的简单示例，我们尝试将 eScience 的下列服务进行了结合：

- 缓存 /镜像加速: `docker.nju.edu.cn`、`gcr.nju.edu.cn`、`mirror.nju.edu.cn`
- CI/CD自动构建: `git.nju.edu.cn`、`reg.nju.edu.cn`
- 计算集群: `hpc.nju.edu.cn`

在这个过程中，这些服务的灵活使用有助于从版本控制、环境搭建、迁移部署等诸多方面建立可靠的工作流程，最终得以节省时间、便捷开发，提升创新协同的质量。