镜像

```
docker pull gitlab/gitlab-ce:16.2.4-ce.0
docker pull gitlab/gitlab-runner:v16.2.1
```

gitlab

基于原有版本数据运行gitlab（用于版本升级）

1. 运行gitlab容器，映射原有gitlab容器的数据卷

```
docker run -d --hostname 192.168.239.142 \
-p 443:443 -p 80:80 --name gitlab-ce1 \
--restart always \
--volumes-from gitlab-ce \
--shm-size 256m \
gitlab/gitlab-ce:16.2.4-ce.0
```

安装新的gitlab

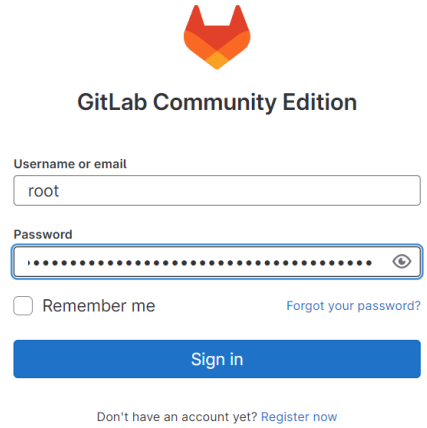
1. 运行gitlab 容器

```
docker run -d --hostname 192.168.239.142 \
-p 443:443 -p 80:80 --name gitlab-ce \
--restart always \
-v $HOME/work/gitlab/config:/etc/gitlab \
-v $HOME/work/gitlab/logs:/var/log/gitlab \
-v $HOME/work/gitlab/data:/var/opt/gitlab \
--shm-size 256m \
gitlab/gitlab-ce:16.2.4-ce.0
```

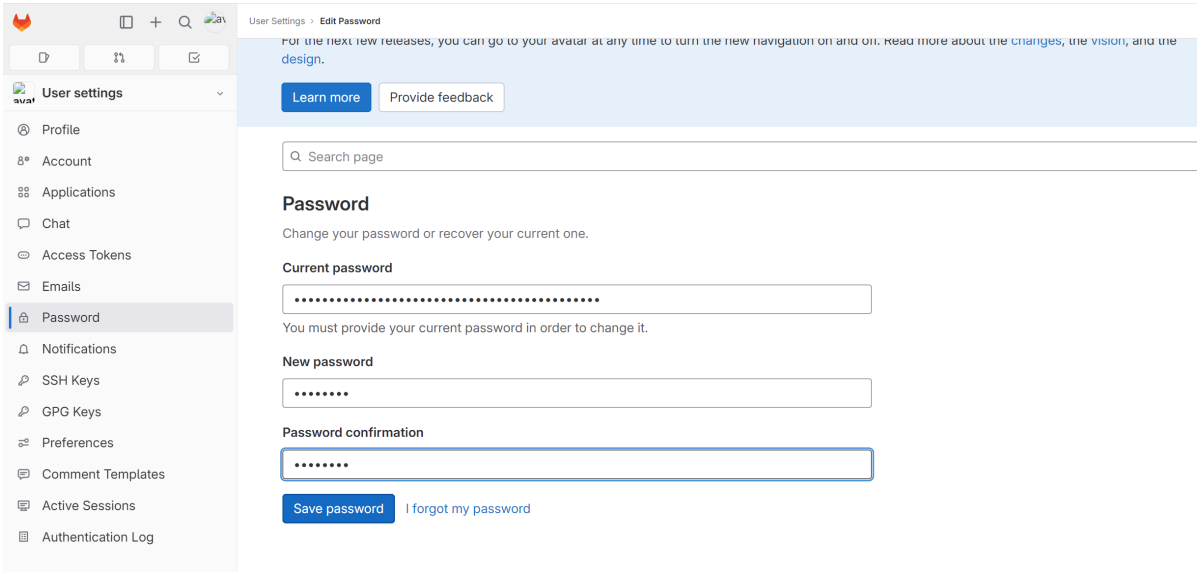
```
//查看容器初始密码
docker exec gitlab-ce grep 'Password:' /etc/gitlab/initial_root_password
// 查看gitlab 版本信息
docker exec gitlab-ce gitlab-rake gitlab:env:info
```

```
GitLab information
Version:         16.2.4
Revision:        deac0811459
Directory:       /opt/gitlab/embedded/service/gitlab-rails
DB Adapter:      PostgreSQL
DB Version:      13.11
URL:             http://192.168.239.142
HTTP Clone URL:  http://192.168.239.142/some-group/some-project.git
SSH Clone URL:   git@192.168.239.142:some-group/some-project.git
Using LDAP:      no
Using Omniauth:  yes
Omniauth Providers:
```

2. root用户登录

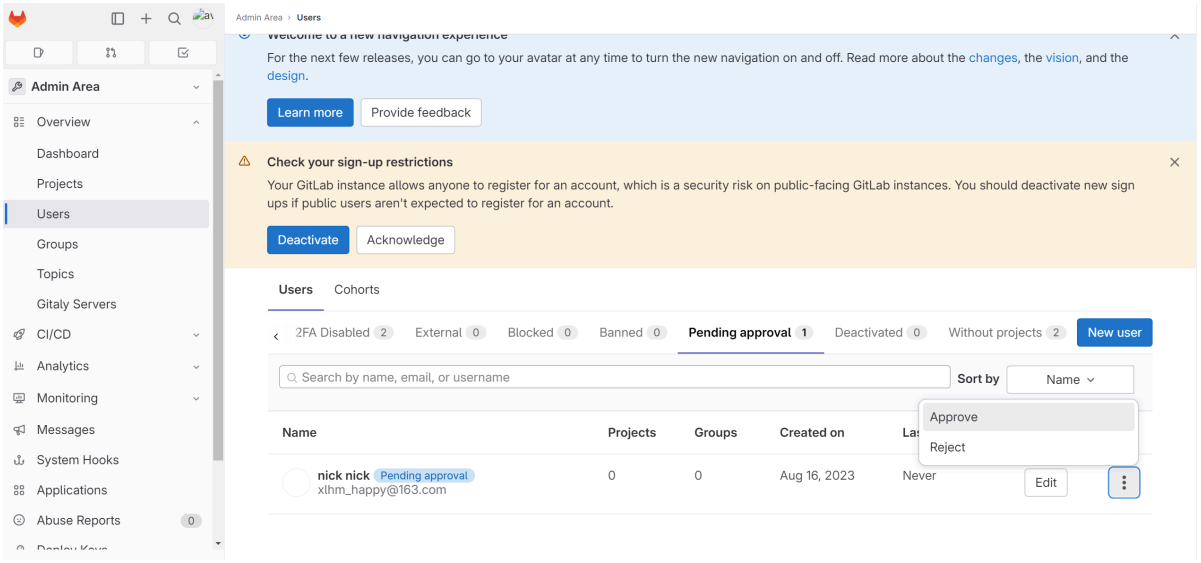


3. 修改root用户登录密码

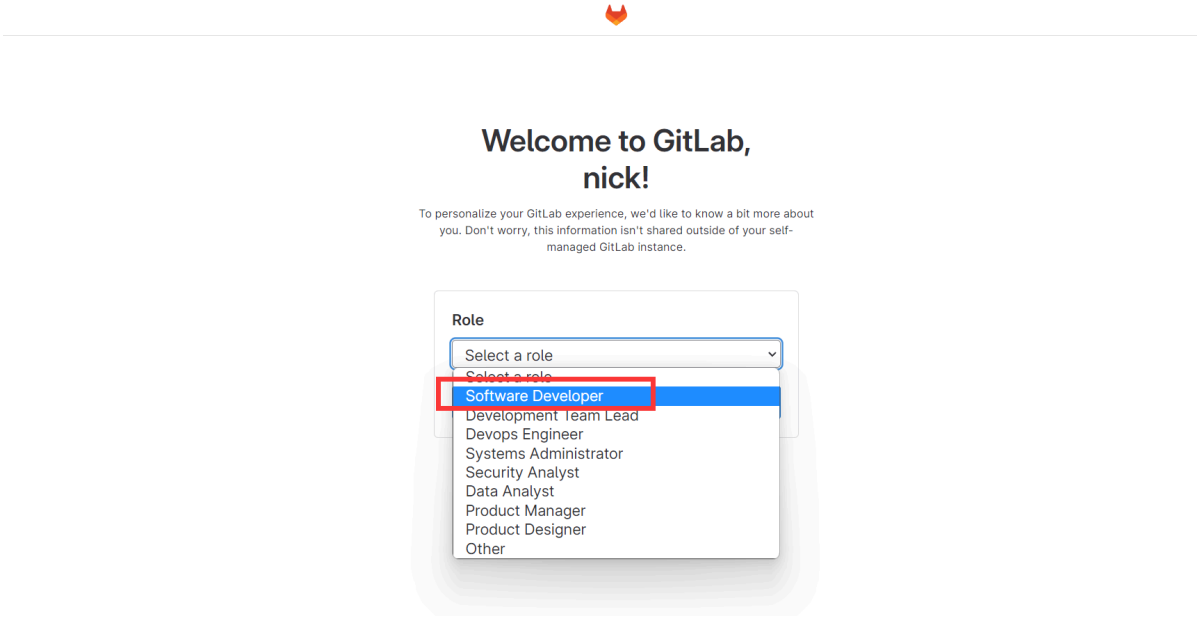


4. 注册开发者角色新用户区别于root用户

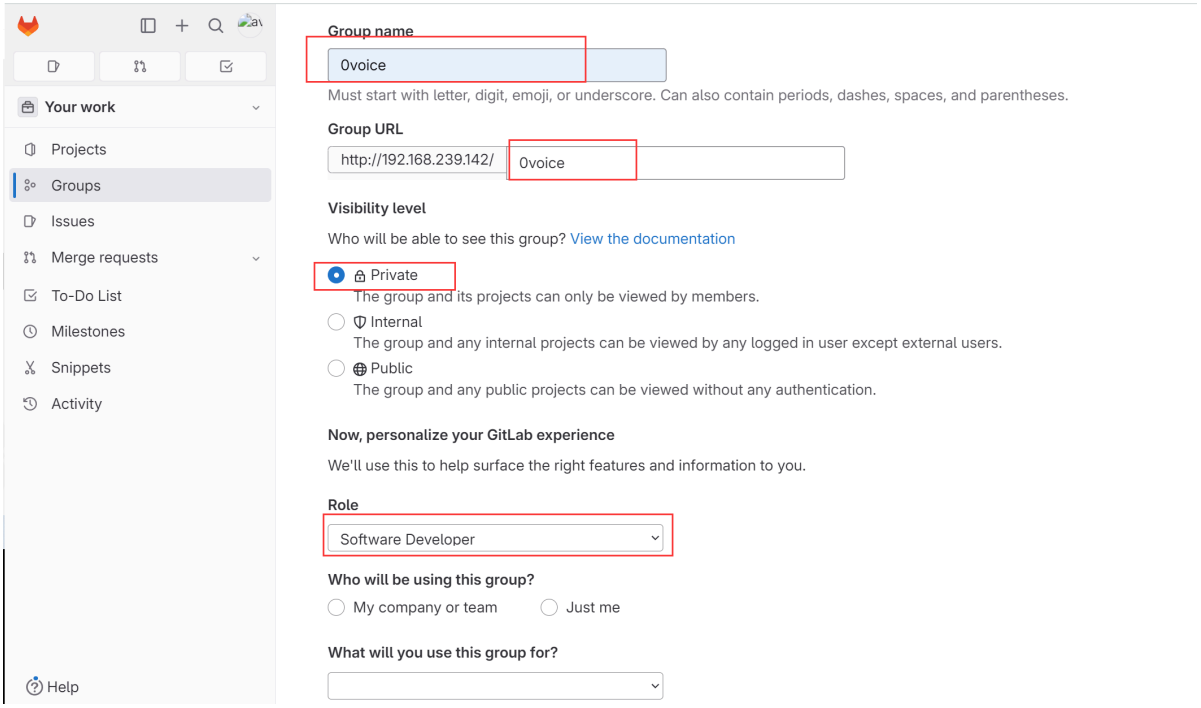
5. 管理员审核



6. 登录新用户选择开发者角色



7. 创建组



gitlab-runner

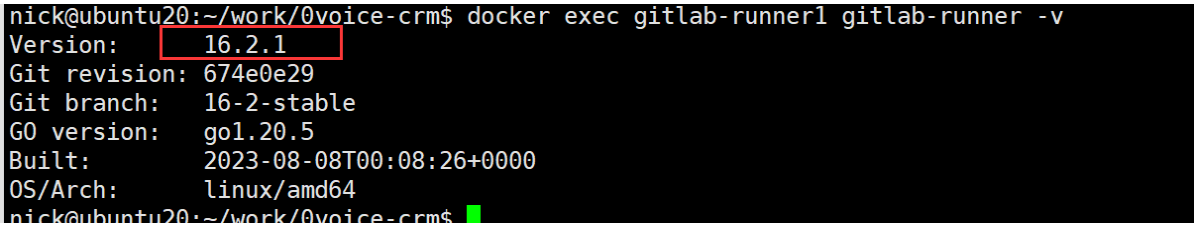
1. 拉取最新的gitlab-runner 镜像

```
docker pull gitlab/gitlab-runner:v16.2.1
```

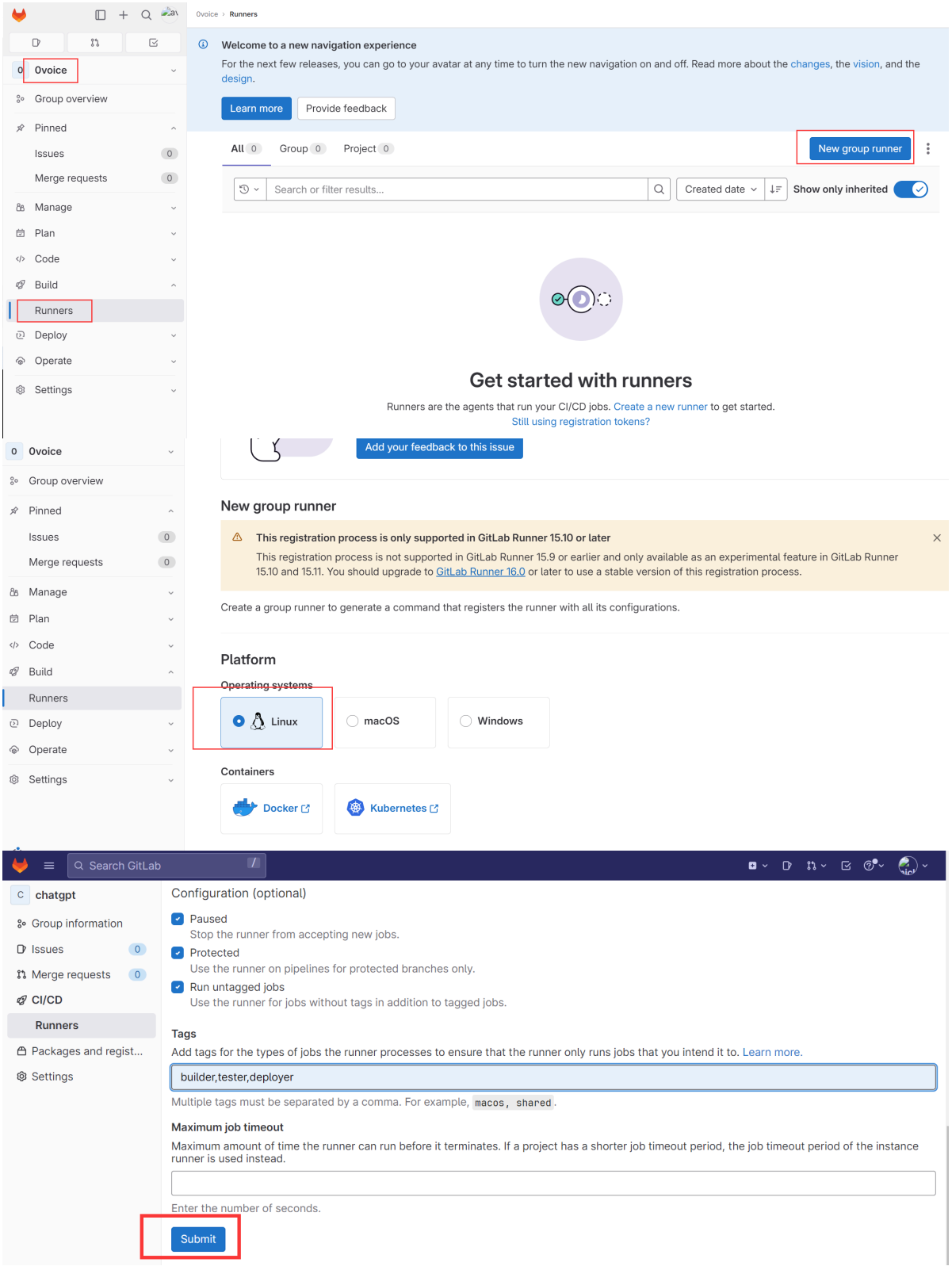
2. 运行gitlab-runner，注意：若需要将应用部署到swarm集群，那么需要再管理节点上部署gitlab-runner

```
# 运行runner
docker run -d --name gitlab-runner1 --restart always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v /srv/gitlab-runner/config:/etc/gitlab-runner \
  gitlab/gitlab-runner:v16.2.1
```

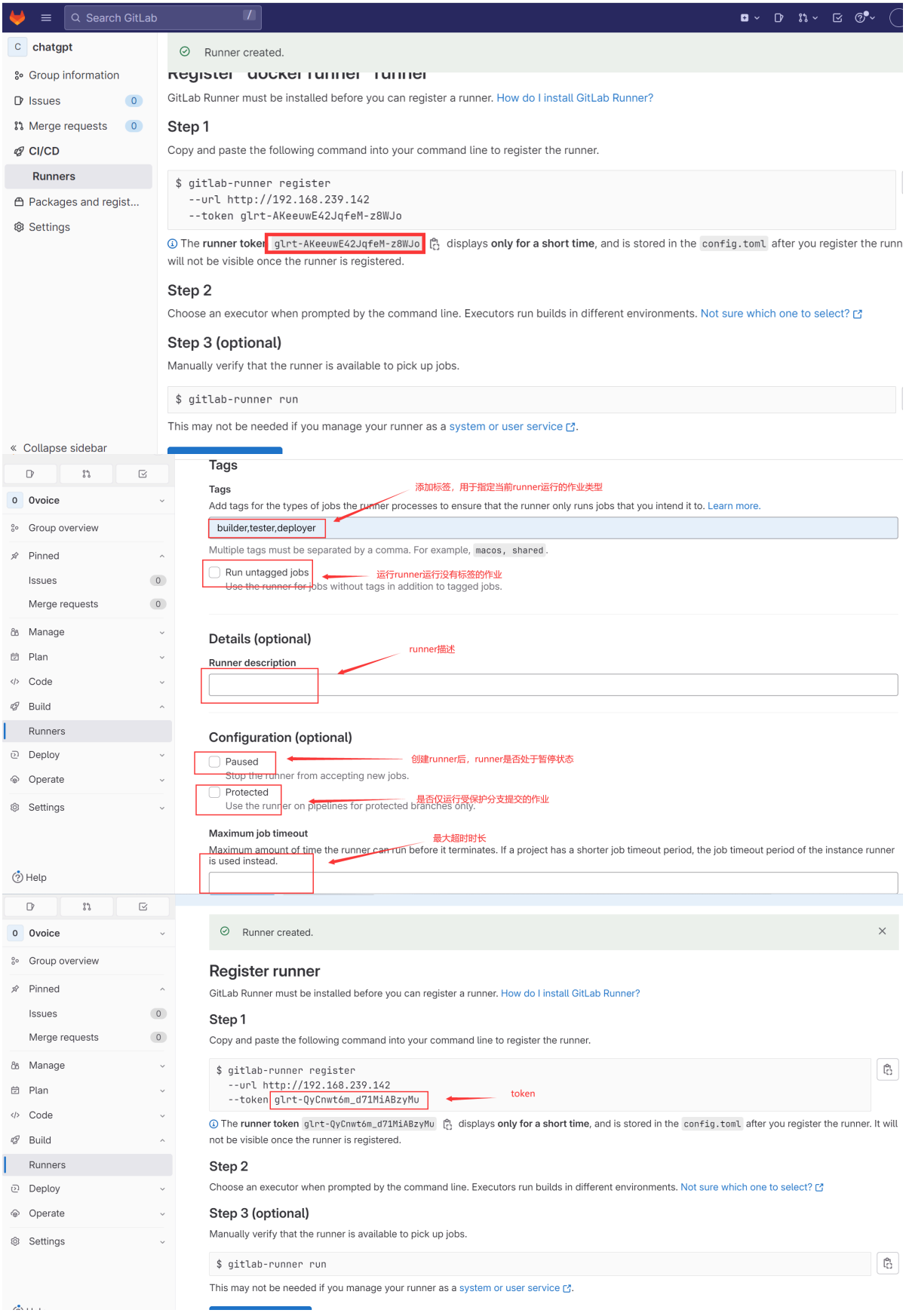
```
# 查看版本
docker exec gitlab-runner1 gitlab-runner -v
```



3. gitlab 创建runner，并获取token


0voice
Runners
Welcome to a new navigation experience
For the next few releases, you can go to your avatar at any time to turn the new navigation on and off. Read more about the changes, the vision, and the design.
Learn more
Provide feedback
All 0
Group 0
Project 0
New group runner
Search or filter results...
Created date
Show only inherited
Get started with runners
Runners are the agents that run your CI/CD jobs. Create a new runner to get started.
Still using registration tokens?
Add your feedback to this issue
New group runner
This registration process is only supported in GitLab Runner 15.10 or later
This registration process is not supported in GitLab Runner 15.9 or earlier and only available as an experimental feature in GitLab Runner 15.10 and 15.11. You should upgrade to GitLab Runner 16.0 or later to use a stable version of this registration process.
Create a group runner to generate a command that registers the runner with all its configurations.
Platform
Operating systems
Linux
macOS
Windows
Containers
Docker
Kubernetes
Group overview
Pinned
Issues
Merge requests
Manage
Plan
Code
Build
Runners
Deploy
Operate
Settings

Search GitLab
chatgpt
Group information
Issues
Merge requests
CI/CD
Runners
Packages and regist...
Settings
Configuration (optional)
Paused
Stop the runner from accepting new jobs.
Protected
Use the runner on pipelines for protected branches only.
Run untagged jobs
Use the runner for jobs without tags in addition to tagged jobs.
Tags
Add tags for the types of jobs the runner processes to ensure that the runner only runs jobs that you intend it to. Learn more.
builder,tester,deployer
Multiple tags must be separated by a comma. For example, macos, shared.
Maximum job timeout
Maximum amount of time the runner can run before it terminates. If a project has a shorter job timeout period, the job timeout period of the instance runner is used instead.
Enter the number of seconds.
Submit


Search GitLab
chatgpt
Group information
Issues 0
Merge requests 0
CI/CD
Runners
Packages and regist...
Settings
Collapse sidebar
Runner created.
GitLab Runner must be installed before you can register a runner. How do I install GitLab Runner?
Step 1
Copy and paste the following command into your command line to register the runner.
$ gitlab-runner register
--url http://192.168.239.142
--token glrt-AKeeuwE42JqfeM-z8WJo
The runner token glrt-AKeeuwE42JqfeM-z8WJo displays only for a short time, and is stored in the config.toml after you register the runn
will not be visible once the runner is registered.
Step 2
Choose an executor when prompted by the command line. Executors run builds in different environments. Not sure which one to select?
Step 3 (optional)
Manually verify that the runner is available to pick up jobs.
$ gitlab-runner run
This may not be needed if you manage your runner as a system or user service.
Tags
Tags
添加标签，用于指定当前runner运行的作业类型
Add tags for the types of jobs the runner processes to ensure that the runner only runs jobs that you intend it to. Learn more.
builder,tester,deployer
Multiple tags must be separated by a comma. For example, macos, shared.
Run untagged jobs
运行runner运行没有标签的作业
Use the runner for jobs without tags in addition to tagged jobs.
Details (optional)
Runner description
runner描述
Configuration (optional)
Paused
创建runner后，runner是否处于暂停状态
Stop the runner from accepting new jobs.
Protected
是否仅运行受保护分支提交的作业
Use the runner on pipelines for protected branches only.
Maximum job timeout
最大超时时长
Maximum amount of time the runner can run before it terminates. If a project has a shorter job timeout period, the job timeout period of the instance runner is used instead.
Ovoice
Group overview
Pinned
Issues 0
Merge requests 0
Manage
Plan
Code
Build
Runners
Deploy
Operate
Settings
Help
Runner created.
Register runner
GitLab Runner must be installed before you can register a runner. How do I install GitLab Runner?
Step 1
Copy and paste the following command into your command line to register the runner.
$ gitlab-runner register
--url http://192.168.239.142
--token glrt-QyCnwt6m_d71MiABzyMu
token
The runner token glrt-QyCnwt6m_d71MiABzyMu displays only for a short time, and is stored in the config.toml after you register the runner. It will not be visible once the runner is registered.
Step 2
Choose an executor when prompted by the command line. Executors run builds in different environments. Not sure which one to select?
Step 3 (optional)
Manually verify that the runner is available to pick up jobs.
$ gitlab-runner run
This may not be needed if you manage your runner as a system or user service.

4. 注册runner

```
//非交互方式注册
docker run --rm -v /srv/gitlab-runner/config:/etc/gitlab-runner \
gitlab/gitlab-runner:v16.2.1 register \
  -n \
  --url "http://192.168.239.142" \
  --token "glrt-QyCnwt6m_d71MiABzyMu" \
  --name "gitlab-runner1" \
  --executor "docker" \
  --docker-image docker:24.0.5 \
  --docker-volumes /var/run/docker.sock:/var/run/docker.sock \
  --docker-pull-policy if-not-present
```

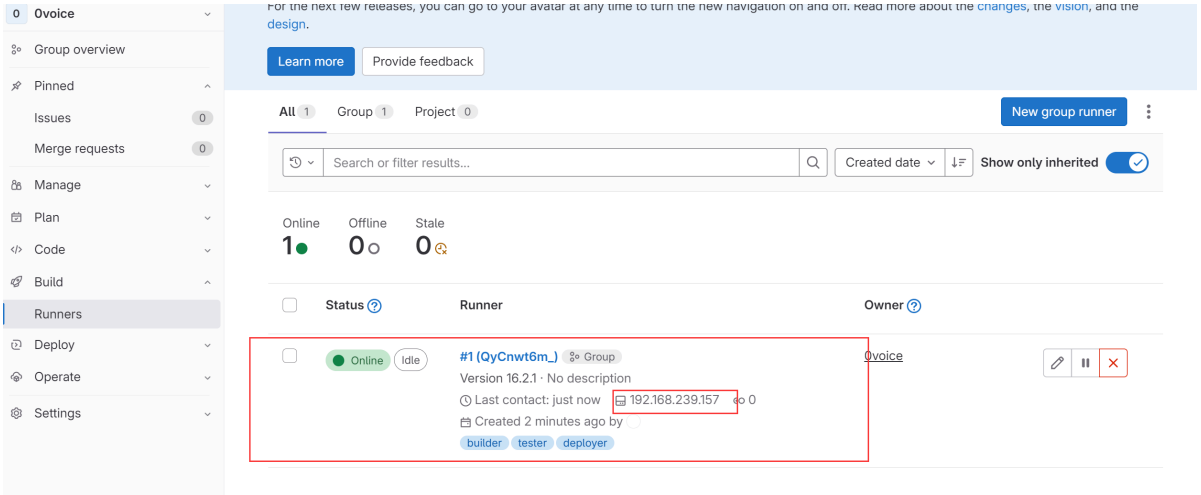
```
nick@ubuntu20:~/work/0voice-crm$ docker run --rm -v /srv/gitlab-runner/config:/etc/gitlab-runner \
> gitlab/gitlab-runner register \
>   -n \
>   --url "http://192.168.239.142" \
>   --token "glrt-QyCnwt6m_d71MiABzyMu" \
>   --name "gitlab-runner1" \
>   --executor "docker" \
>   --docker-image docker:24.0.5 \
>   --docker-volumes /var/run/docker.sock:/var/run/docker.sock \
>   --docker-pull-policy if-not-present
Runtime platform                                    arch=amd64 os=linux pid=6 revision=674e0e29 version=16.2.1
Running in system-mode.

Verifying runner... is valid                        runner=QyCnwt6m_
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.toml"
nick@ubuntu20:~/work/0voice-crm$
```

```
# 重启容器
docker restart gitlab-runner1
```

2. runner注册成功



.gitlab-ci.yaml

关键词参考文档：https://docs.gitlab.com/ee/ci/yaml/gitlab_ci_yaml.html
CI/CD预定义变量参考文档：https://docs.gitlab.com/ee/ci/variables/

关键词	描述
全局关键词：	
default	为某些关键词设置全局的默认值
include	将外部的yaml文件包括到当前配置，例如：将一个长的.gitlab-ci.yml文件分为多个文件，通过include来引入各个小文件，用于提高可读性
stages	阶段，用于定义作业组的阶段，流水线默认阶段包括：.pre,build,test,deploy,.post
workflow	通过规则控制流水线的行为

Job关键词：

after_script 使用after_script定义在每个作业（包括失败的作业）之后运行的命令数组
allow_failure 使用allow_failure确定作业失败时管道是否应继续运行
artifacts 使用artifacts指定要保存为作业工件的文件。作业工件是作业成功、失败或始终附加到作业的文件和目录的列表
before_script 使用before_script定义一组命令，这些命令应在每个作业的脚本命令之前运行，但应在工件恢复之后运行
cache 使用缓存可以指定要在作业之间缓存的文件和目录的列表。只能使用本地工作副本中的路径
coverage 将覆盖率与自定义正则表达式一起使用，可以配置如何从作业输出中提取代码覆盖率
dast_configuration 使用dast_configuration关键字可以指定要在CI/CD配置中使用的站点配置文件和扫描仪配置文件
dependencies 使用dependencies关键字可以定义从中获取工件的作业列表。您还可以设置一个作业，以完全不下载工件
environment 定义环境变量或指定作业部署的环境
extends 使用extends可重用配置节。它是YAML锚的一种替代方案，并且更加灵活和可读
hooks 使用钩子指定在作业执行的某些阶段（如检索Git存储库之前）在运行程序上执行的命令列表
id_tokens 使用id_tokens创建JSON web令牌（JWT），以通过第三方服务进行身份验证
image 使用image指定作业运行的Docker映像
inherit 使用inherit控制默认关键字和变量的继承
interruptible 如果在作业完成之前新管道启动时应取消作业，请使用可中断
needs 使用needs无序执行作业
only / except only和except来控制何时将作业添加到管道
pages 使用pages定义将静态内容上载到GitLab的GitLab页面作业。然后将内容发布为网站
parallel 使用parallel在单个管道中并行运行作业多次
release 创建一个发布版本
resource_group 使用resource_group可以创建一个资源组，以确保作业在同一项目的不同管道之间互斥
retry 使用“重试”配置作业失败时重试的次数。如果未定义，则默认为0，并且作业不会重试
rules 使用rules包括或排除管道中的作业
script 使用script为运行程序指定要执行的命令
secrets 使用secrets将CI/CD中的机密信息指定为：1.外部secrets检索提供程序；2.在作业中作为ci/cd的变量
services 使用services指定脚本成功运行所需的任何其他Docker映像
stage 使用阶段定义作业在哪个阶段运行。同一阶段中的作业可以并行执行
tags 使用标记从项目可用的所有runner列表中选择特定runner
timeout 使用timeout为特定作业配置超时。如果作业运行的时间超过超时时间，则作业将失败
trigger 使用触发器声明作业是启动下游管道的“触发器作业”
variables 使用变量定义作业的CI/CD变量
when 使用wen配置作业运行的条件。如果未在作业中定义，则默认值为：on_ccess