Gitlab CI / CD

1 基本介绍

持续集成(CI)

将团队提供的代码集成到共享存储库中。开发人员在Merge(Pull)请求中共享新代码。在合并存储库的更改之前，该请求触发了构建，测试和验证新代码的管道，以确保没有集成问题并及早发现任何问题。

持续集成的工作原理是将小的代码块推送到Git存储库中托管的应用程序代码库中，并且每次推送时，都要运行脚本管道来构建，测试和验证代码更改，然后再将其合并到主分支中。

连续交付(CD)

可通过结构化的部署管道确保将经过CI验证的代码交付给您的应用程序。

CI可帮助在开发周期的早期发现并减少错误，并且CD可以将经过验证的代码更快地移至应用程序中。

参考链接：https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/

https://www.youtube.com/watch?v=1iXFbchozdY

- 为什么需要CI / CD工作流程
- GitLab CI / CD有哪些优势
- 特征

2 如何工作

参考链接：https://docs.gitlab.com/ee/ci/introduction/index.html#how-gitlab-cicd-works

3 管道架构

参考链接：https://docs.gitlab.com/ee/ci/pipelines/pipeline_architectures.html

- 基本管道：在构建过程中同时运行所有内容，一旦所有这些操作完成，它将以相同的方式在测试阶段运行所有内容。

```
stages:
  - build
  - test
  - deploy

image: alpine

build_a:
  stage: build
  script:
    - echo "This job builds something."

build_b:
  stage: build
  script:
    - echo "This job builds something else."

test_a:
  stage: test
  script:
    - echo "This job tests something. It will only run when all jobs in the"
    - echo "build stage are complete."

test_b:
  stage: test
  script:
    - echo "This job tests something else. It will only run when all jobs in the"
    - echo "build stage are complete too. It will start at about the same time as test_a."

deploy_a:
  stage: deploy
  script:
    - echo "This job deploys something. It will only run when all jobs in the"
    - echo "test stage complete."

deploy_b:
  stage: deploy
  script:
    - echo "This job deploys something else. It will only run when all jobs in the"
    - echo "test stage complete. It will start at about the same time as deploy_a."
```

- 有向无环图管道

```
stages:
  - build
  - test
  - deploy

image: alpine

build_a:
  stage: build
```

```yaml
  script:
    - echo "This job builds something quickly."

build_b:
  stage: build
  script:
    - echo "This job builds something else slowly."

test_a:
  stage: test
  needs: build_a
  script:
    - echo "This test job will start as soon as build_a finishes."
    - echo "It will not wait for build_b, or other jobs in the build stage, to finish."

test_b:
  stage: test
  needs: build_b
  script:
    - echo "This test job will start as soon as build_b finishes."
    - echo "It will not wait for other jobs in the build stage to finish."

deploy_a:
  stage: deploy
  needs: test_a
  script:
    - echo "Since build_a and test_a run quickly, this deploy job can run much earlier."
    - echo "It does not need to wait for build_b or test_b."

deploy_b:
  stage: deploy
  needs: test_b
  script:
    - echo "Since build_b and test_b run slowly, this deploy job will run much later."
```

- 子/父管道

```yaml
stages:
  - triggers

trigger_a:
  stage: triggers
  trigger:
    include: a/.gitlab-ci.yml
  rules:
    - changes:
      - a/*

trigger_b:
  stage: triggers
  trigger:
    include: b/.gitlab-ci.yml
  rules:
    - changes:
      - b/*
```

4 使用 CI / CD - 官网教程

参考链接：

http://101.200.51.192/help/ci/quick_start/README

5 使用 CI / CD - 实际操作

1 在服务器上安装gitlab-runner

第一步：添加GitLab的官方存储库

```
curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh | sudo bash
```

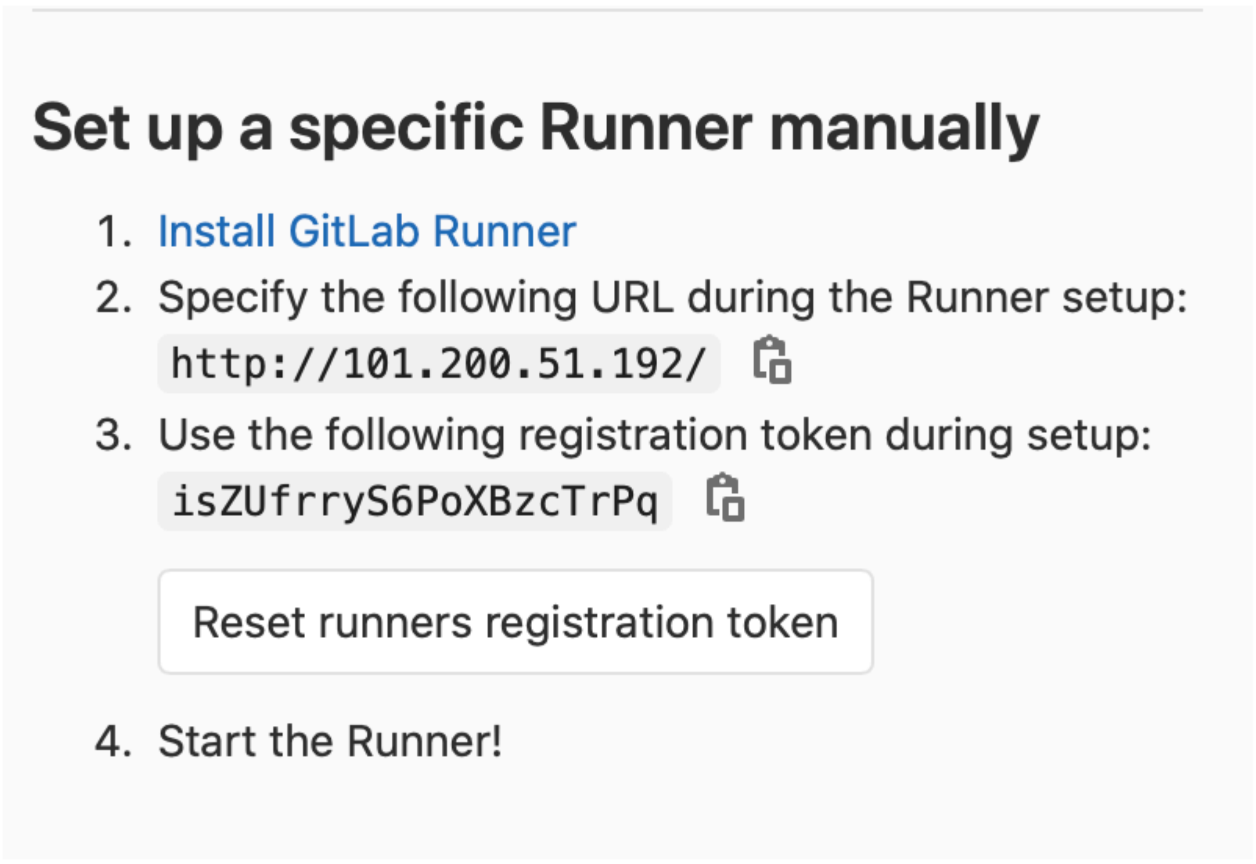
第二步：安装最新版本的GitLab Runner

```
apt-get install gitlab-runner
```

2 注册runner

下面开始实现Spring Boot的CI / CD：

将项目上传至gitlab，在gitlab项目的左侧菜单中的Settings下点击CI / CD，展开Runners，可以看到URL、token等信息。



Set up a specific Runner manually

1. Install GitLab Runner
2. Specify the following URL during the Runner setup:
   `http://101.200.51.192/`
3. Use the following registration token during setup:
   `isZUfrryS6PoXBzcTrPq`

   Reset runners registration token

4. Start the Runner!

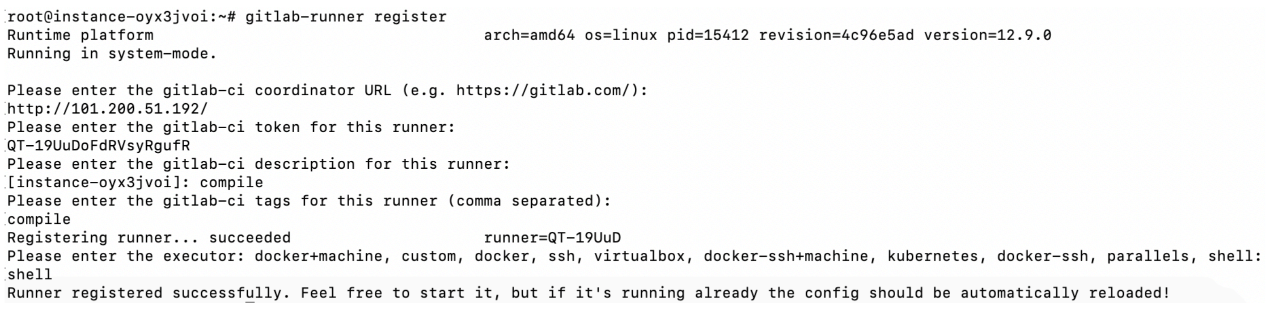
输入命令

```
gitlab-runner register
```

根据提示与上面的信息，完成Runner注册。

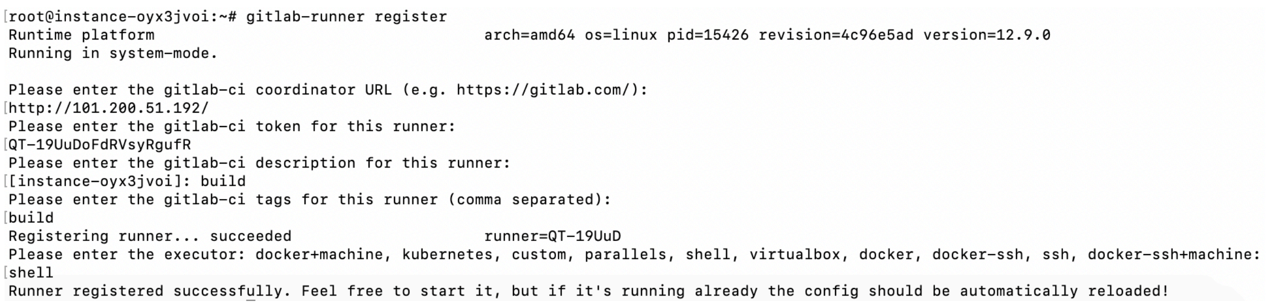
为compile、build、deploy分别注册Runner，用输入对应的tag，使用shell。

compile



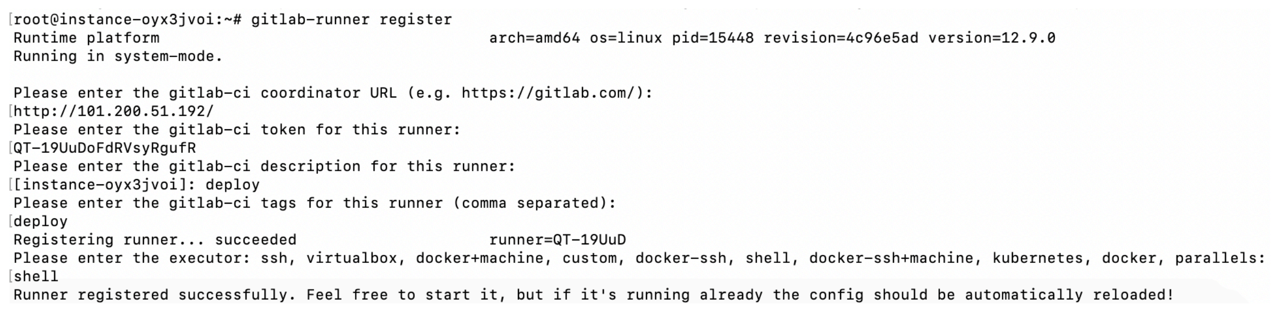
```
root@instance-oyx3jvoi:~# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=15412 revision=4c96e5ad version=12.9.0
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://101.200.51.192/
Please enter the gitlab-ci token for this runner:
QT-19UuDoFdRVsyRgufR
Please enter the gitlab-ci description for this runner:
[instance-oyx3jvoi]: compile
Please enter the gitlab-ci tags for this runner (comma separated):
compile
Registering runner... succeeded                     runner=QT-19UuD
Please enter the executor: docker+machine, custom, docker, ssh, virtualbox, docker-ssh+machine, kubernetes, docker-ssh, parallels, shell:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

build



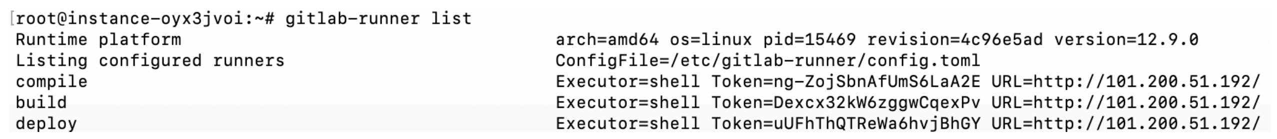
```
root@instance-oyx3jvoi:~# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=15426 revision=4c96e5ad version=12.9.0
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://101.200.51.192/
Please enter the gitlab-ci token for this runner:
QT-19UuDoFdRVsyRgufR
Please enter the gitlab-ci description for this runner:
[instance-oyx3jvoi]: build
Please enter the gitlab-ci tags for this runner (comma separated):
build
Registering runner... succeeded                     runner=QT-19UuD
Please enter the executor: docker+machine, kubernetes, custom, parallels, shell, virtualbox, docker, docker-ssh, ssh, docker-ssh+machine:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

deploy

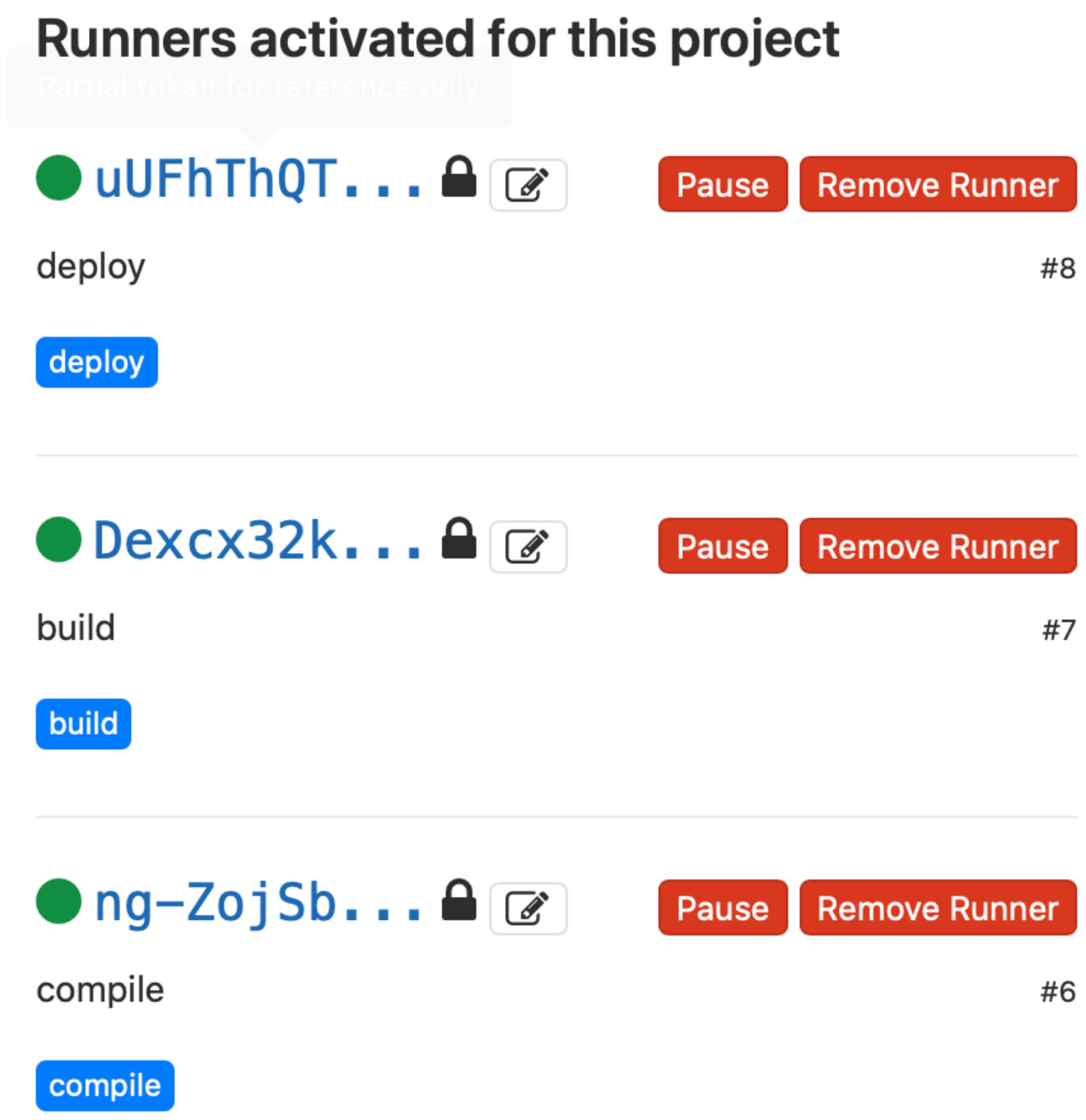


```
root@instance-oyx3jvoi:~# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=15448 revision=4c96e5ad version=12.9.0
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://101.200.51.192/
Please enter the gitlab-ci token for this runner:
QT-19UuDoFdRVsyRgufR
Please enter the gitlab-ci description for this runner:
[instance-oyx3jvoi]: deploy
Please enter the gitlab-ci tags for this runner (comma separated):
deploy
Registering runner... succeeded                     runner=QT-19UuD
Please enter the executor: ssh, virtualbox, docker+machine, custom, docker-ssh, shell, docker-ssh+machine, kubernetes, docker, parallels:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

输入gitlab-runner list命令可以获得当前注册的runners，在gitlab项目的Settings菜单中的CI / CD中Runners的展开页面也可以看到新增runner：



```
root@instance-oyx3jvoi:~# gitlab-runner list
Runtime platform                                    arch=amd64 os=linux pid=15469 revision=4c96e5ad version=12.9.0
Listing configured runners                          ConfigFile=/etc/gitlab-runner/config.toml
compile                                             Executor=shell Token=ng-ZojSbnAfUmS6LaA2E URL=http://101.200.51.192/
build                                               Executor=shell Token=Dexcx32kW6zggwCqexPv URL=http://101.200.51.192/
deploy                                              Executor=shell Token=uUFhThQTReWa6hvjBhGY URL=http://101.200.51.192/
```



Runners activated for this project

● uUFhThQT... Pause Remove Runner

deploy #8

deploy

● Dexcx32k... Pause Remove Runner

build #7

build

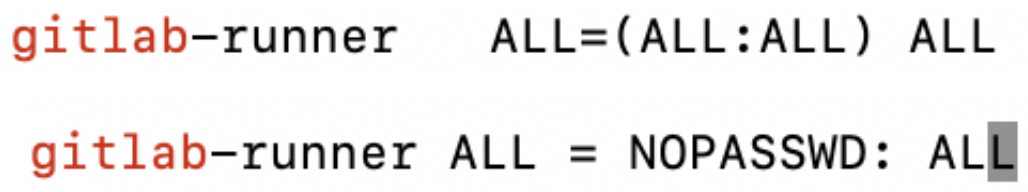
● ng-ZojSb... Pause Remove Runner

compile #6

compile

3 服务器端准备

首先先给gitlab-runner用户管理员权限并可以免密码使用sudo，修改/etc/sudoers增加下面两行。



```
gitlab-runner   ALL=(ALL:ALL) ALL

gitlab-runner ALL = NOPASSWD: ALL
```

服务器进入gitlab-runner用户。

进入～目录，git clone下需要部署的项目。

进入～目录，创建CICD目录，在CICD目录创建Dockerfile、main.sh，Dockerfile、main.sh编写指南详见“项目部署指南.pdf”。

Dockerfile

```
# Docker image for springboot file run
# VERSION 0.0.1
# Author: eangulee
# 基础镜像使用java
FROM java:8
# VOLUME 指定了临时文件目录为/tmp。
# 其效果是在主机 /var/lib/docker 目录下创建了一个临时文件，并链接到容器的/tmp
VOLUME /tmp
# 将jar包添加到容器中并更名为app.jar
ADD shop-1.0.jar app.jar
# 运行jar包
RUN bash -c 'touch /app.jar'
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

mian.sh

```
docker stop test
docker rm test
docker rmi test
docker build -t test .
docker run -p 8080:8080 -d --name test test
```

4 创建.gitlab-ci.yml

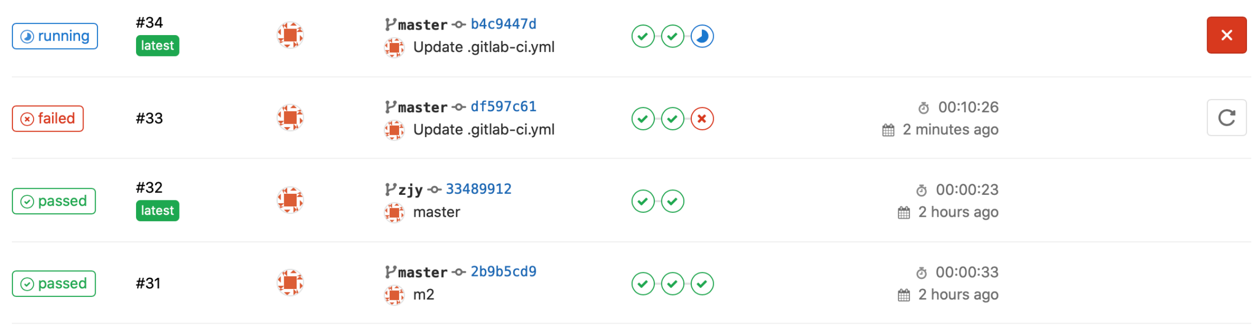
在gitlab项目的根目录创建.gitlab-ci.yml，随项目一起上传到gitlab仓库中。

```
stages:
  - compile
  - build
  - deploy

```

```
compile:
  stage: compile
  tags:
    - compile
  script:
    - cd ~/test #test是从gitlab拉下来的项目
    - git pull origin master
    - mvn clean compile #编译

build:
  stage: build
  tags:
    - build
  script:
    - mvn package spring-boot:repackage #打包
    - mv ./target/shop-1.0.jar ~/CICD

deploy:
  stage: deploy
  tags:
    - deploy
  script:
    - cd ~/CICD
    - sudo sh main.sh #见项目部署指南
  only:
    - master
```

注意：tags用来指示使用的runner，和注册时输入的runner tag相对应，only: - master表示只在master分支上启效果，script下的命令时shell命令。

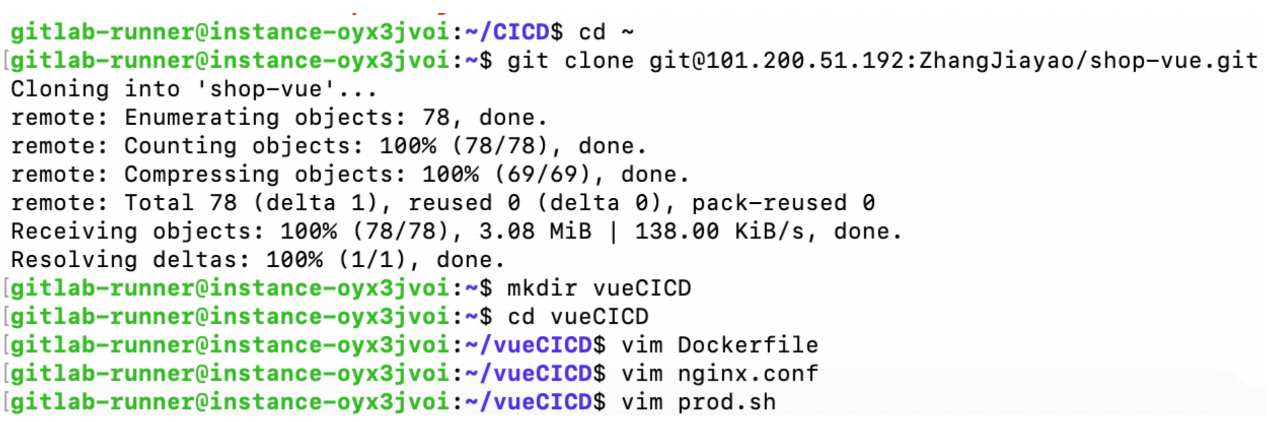
上传完毕后，点击左侧菜单栏中CI / CD，可以看到当前pipelines的状态等。



running	#34 latest	master b4c9447d Update .gitlab-ci.yml	✓ ✓ ◔	
failed	#33	master df597c61 Update .gitlab-ci.yml	✓ ✓ ✗	00:10:26 2 minutes ago
passed	#32 latest	zjy 33489912 master	✓ ✓	00:00:23 2 hours ago
passed	#31	master 2b9b5cd9 m2	✓ ✓ ✓	00:00:33 2 hours ago

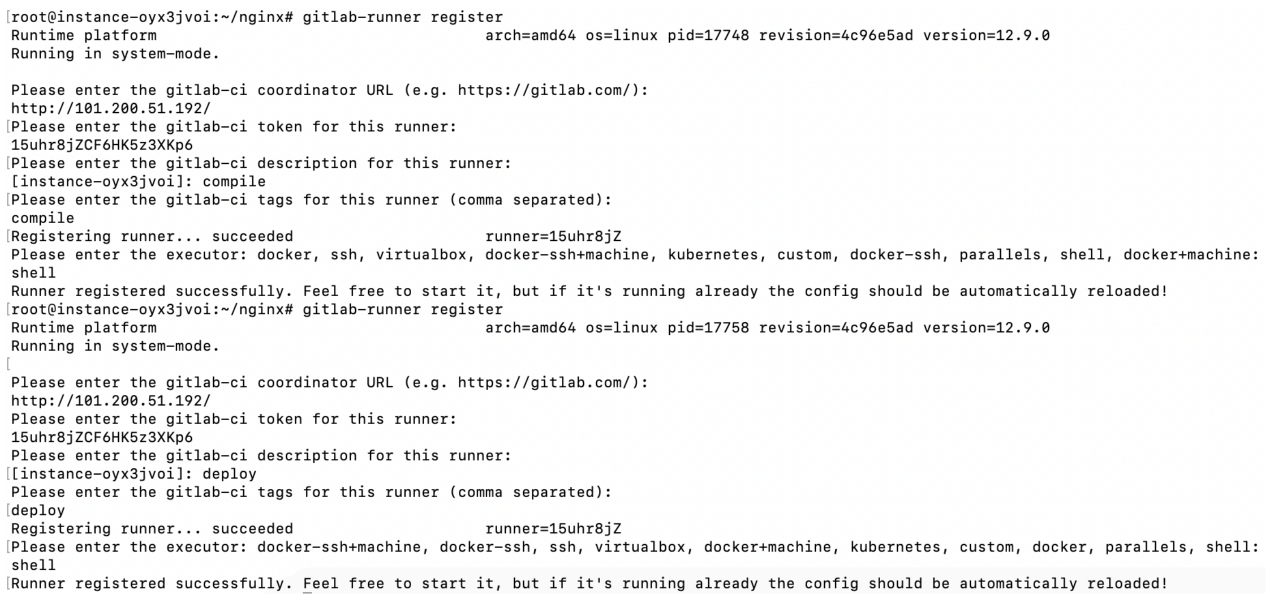
vue项目

vue项目实现CI / CD过程与上述Spring Boot项目大同小异，相应的文件准备见“项目部署指南.pdf”。



```
gitlab-runner@instance-oyx3jvoi:~/CICD$ cd ~
gitlab-runner@instance-oyx3jvoi:~$ git clone git@101.200.51.192:ZhangJiayao/shop-vue.git
Cloning into 'shop-vue'...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 78 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (78/78), 3.08 MiB | 138.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
gitlab-runner@instance-oyx3jvoi:~$ mkdir vueCICD
gitlab-runner@instance-oyx3jvoi:~$ cd vueCICD
gitlab-runner@instance-oyx3jvoi:~/vueCICD$ vim Dockerfile
gitlab-runner@instance-oyx3jvoi:~/vueCICD$ vim nginx.conf
gitlab-runner@instance-oyx3jvoi:~/vueCICD$ vim prod.sh
```

注册runners：



```
root@instance-oyx3jvoi:~/nginx# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=17748 revision=4c96e5ad version=12.9.0
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://101.200.51.192/
Please enter the gitlab-ci token for this runner:
15uhr8jZCF6HK5z3XKp6
Please enter the gitlab-ci description for this runner:
[instance-oyx3jvoi]: compile
Please enter the gitlab-ci tags for this runner (comma separated):
compile
Registering runner... succeeded                     runner=15uhr8jZ
Please enter the executor: docker, ssh, virtualbox, docker-ssh+machine, kubernetes, custom, docker-ssh, parallels, shell, docker+machine:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
root@instance-oyx3jvoi:~/nginx# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=17758 revision=4c96e5ad version=12.9.0
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://101.200.51.192/
Please enter the gitlab-ci token for this runner:
15uhr8jZCF6HK5z3XKp6
Please enter the gitlab-ci description for this runner:
[instance-oyx3jvoi]: deploy
Please enter the gitlab-ci tags for this runner (comma separated):
deploy
Registering runner... succeeded                     runner=15uhr8jZ
Please enter the executor: docker-ssh+machine, docker-ssh, ssh, virtualbox, docker+machine, kubernetes, custom, docker, parallels, shell:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

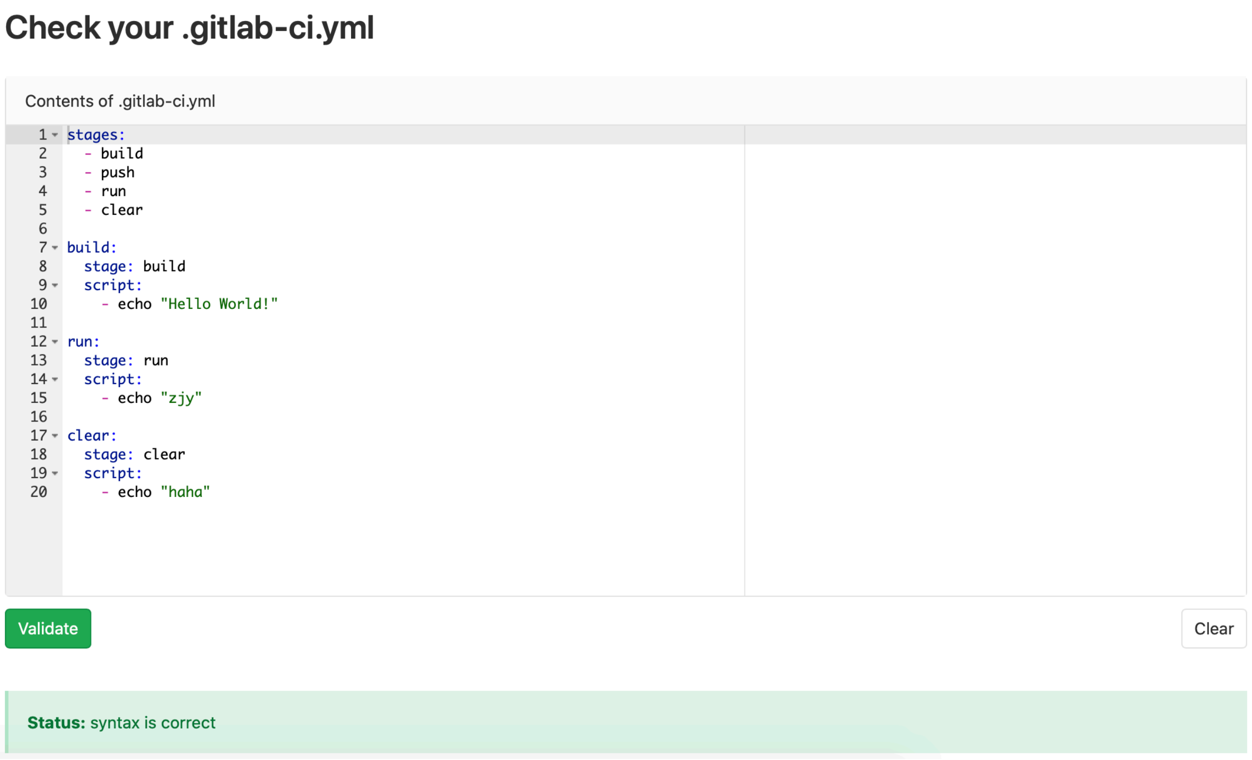
vue项目的.gitlab-ci.yml如下：

```yaml
before_script:
  - cd ~/shop-vue
  - npm install

stages:
  - compile
  - deploy

compile:
  stage: compile
  tags:
    - compile
  script:
    - git pull origin master
    - npm run build

deploy:
  stage: deploy
  tags:
    - deploy
  script:
    - mv ~/shop-vue/dist ~/vueCICD
    - cd ~/vueCICD
    - sudo sh prod.sh
  only:
    - master
```

6 check .gitlab-ci.yml

可以点击左侧菜单栏中CI / CD，点击右上方的

 按钮Check .gitlab-ci.yml 文件：

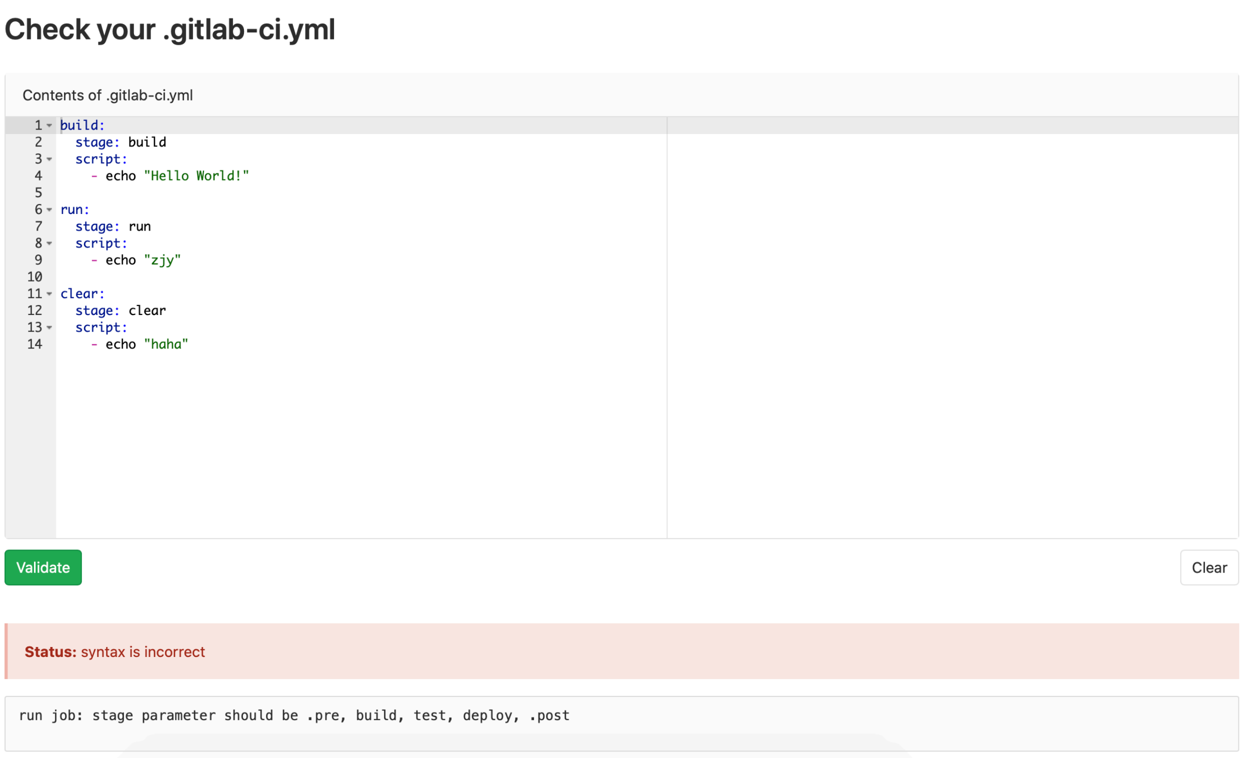


Check your .gitlab-ci.yml

Contents of .gitlab-ci.yml

```
stages:
  - build
  - push
  - run
  - clear

build:
  stage: build
  script:
    - echo "Hello World!"

run:
  stage: run
  script:
    - echo "zjy"

clear:
  stage: clear
  script:
    - echo "haha"
```

Status: syntax is correct


Check your .gitlab-ci.yml

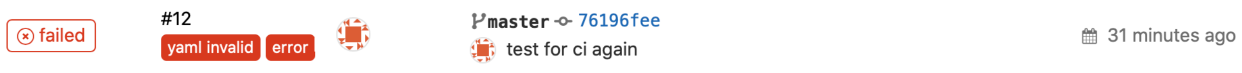
Contents of .gitlab-ci.yml

```
build:
  stage: build
  script:
    - echo "Hello World!"

run:
  stage: run
  script:
    - echo "zjy"

clear:
  stage: clear
  script:
    - echo "haha"
```

Status: syntax is incorrect

```
run job: stage parameter should be .pre, build, test, deploy, .post
```

如果.gitlab-ci.yml错误，会导致piplelines出现error。


failed #12 yaml invalid error master 76196fee test for ci again 31 minutes ago

如果pipeline一直处于pending状态，可以尝试gitlab-ci-multi-runner restart命令重启runner，但不一定有用。

7 其他参考资料

https://lusyoe.github.io/2016/08/29/Gitlab-CI-Multi-Runner%E6%90%AD%E5%BB%BACI%E6%8C%81%E7%BB%AD%E9%9B%86%E6%88%90%E7%8E%AF%E5%A2%83/