

# 一步一步完成GitLab Runner持续化自动部署

本文将以Ubuntu16.04.4+Docker自动化部署Dotnetcore项目

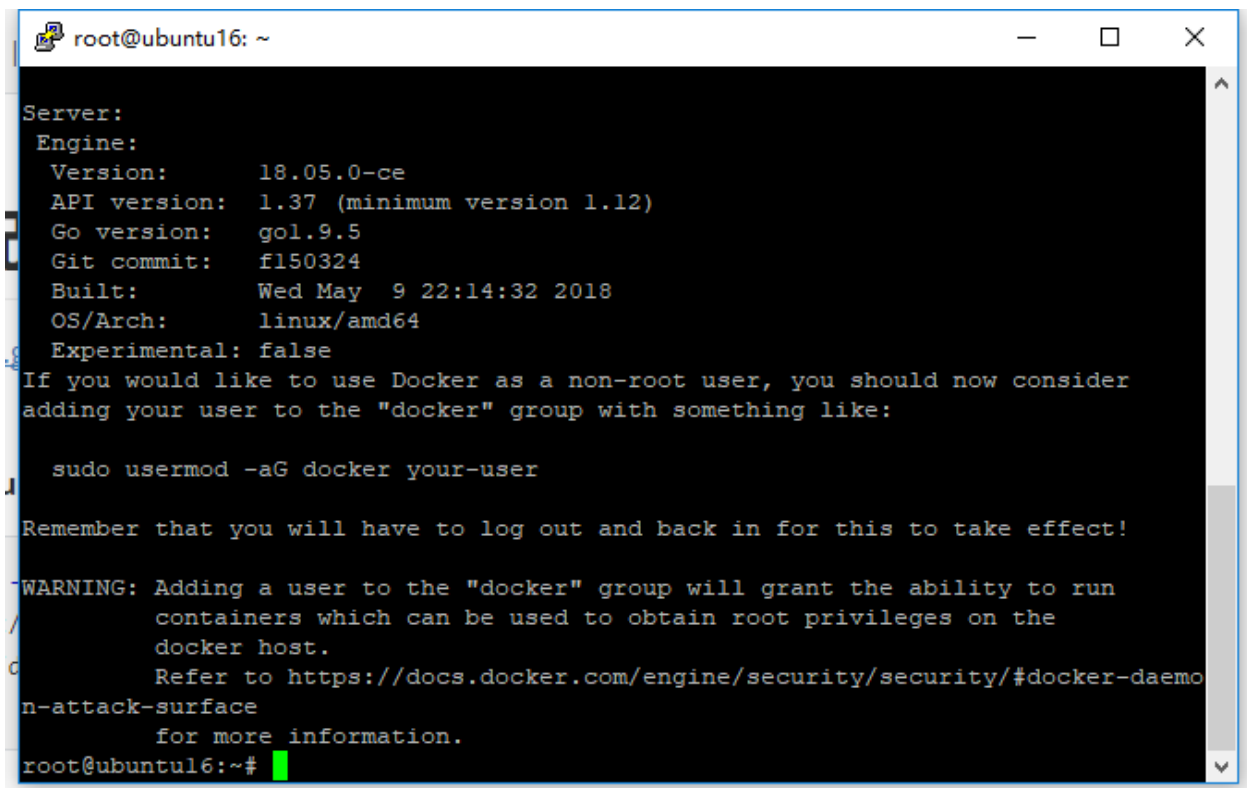
## 1.安装gitlab

- 安装使用本地离线安装,下载相应包后直接安装即可,然后安装配置web地址等等,这个在网上很多示例,这里不再赘述.

## 2.Ubuntu安装Docker

- 使用官方脚本安装Docker,安装源为阿里云

```
curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
```

A terminal window titled 'root@ubuntu16: ~' showing the output of the Docker installation script. The output includes the Docker Engine version (18.05.0-ce), API version (1.37), Go version (gol.9.5), Git commit (f150324), build date (Wed May 9 22:14:32 2018), OS/Arch (linux/amd64), and Experimental status (false). It also provides instructions for adding a user to the 'docker' group and a warning about root privileges.

```
root@ubuntu16: ~  
Server:  
Engine:  
  Version:      18.05.0-ce  
  API version:  1.37 (minimum version 1.12)  
  Go version:   gol.9.5  
  Git commit:   f150324  
  Built:        Wed May  9 22:14:32 2018  
  OS/Arch:      linux/amd64  
  Experimental: false  
If you would like to use Docker as a non-root user, you should now consider  
adding your user to the "docker" group with something like:  
  
    sudo usermod -aG docker your-user  
  
Remember that you will have to log out and back in for this to take effect!  
  
WARNING: Adding a user to the "docker" group will grant the ability to run  
containers which can be used to obtain root privileges on the  
docker host.  
Refer to https://docs.docker.com/engine/security/security/#docker-daemo  
n-attack-surface  
for more information.  
root@ubuntu16:~#
```

## 3.在Docker安装并配置GitLab Runner

- 参考官网地址Gitlab Runner <https://docs.gitlab.com/runner/install/>
- 在Docker安装GitLab Runner
  - 1.使用命令在Docker中安装Gitlab Runner

```
docker run -d --name gitlab-runner --restart always \
-v /srv/gitlab-runner/config:/etc/gitlab-runner \
-v /var/run/docker.sock:/var/run/docker.sock \
gitlab/gitlab-runner:latest
```

○ 使用[阿里云镜像加速](#),这样在部署的时候会快一点,按照阿里云官网添加即可,不赘述.

○ 2.注册并设置Gitlab Runner

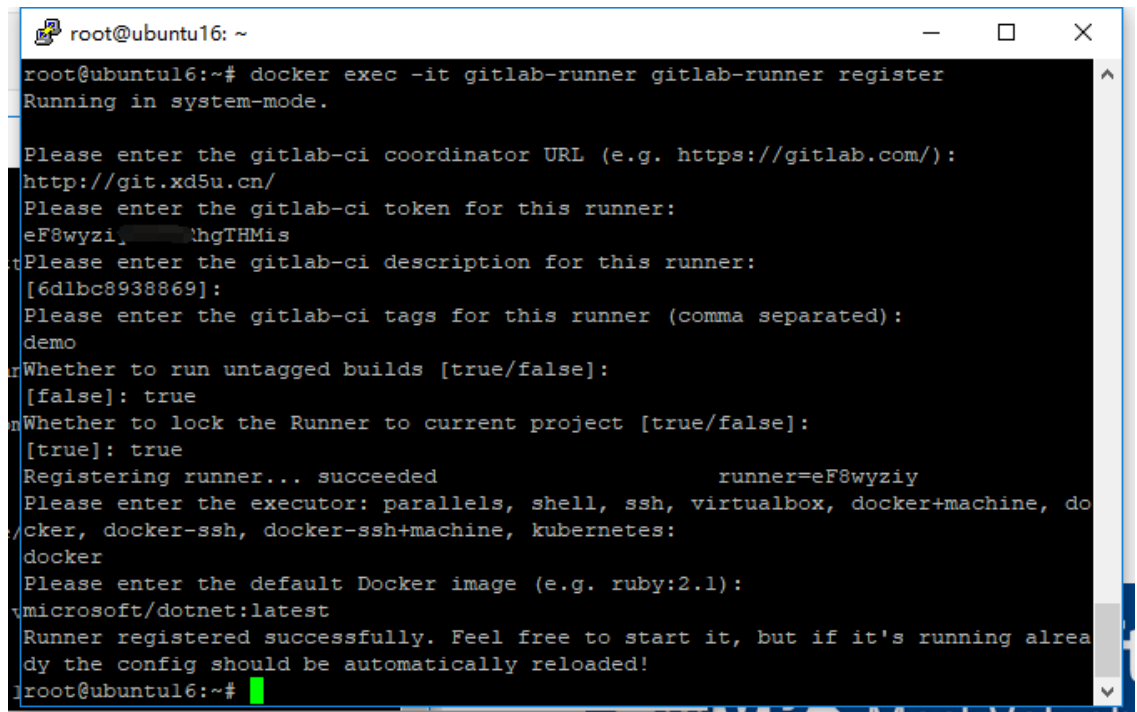
- 1.访问Gitlab获取 `http://你的gitlab地址/admin/runners`

### Setup a shared Runner manually

1. 安装一个与 GitLab CI 兼容的 Runner (如需了解更多的安装信息, 请查看 [GitLab Runner](#))
2. 在 Runner 设置时指定以下 URL: `http://git.xd5u.cn/`
3. 在安装过程中使用以下注册令牌: `eF8w...gTHMis`
4. 启动 Runner!

- 2.运行 `Gitlab Runner` 注册设置

```
docker exec -it gitlab-runner gitlab-runner register
```



```
root@ubuntu16: ~
root@ubuntu16:~# docker exec -it gitlab-runner gitlab-runner register
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://git.xd5u.cn/
Please enter the gitlab-ci token for this runner:
eF8wyziz...hgTHMis
Please enter the gitlab-ci description for this runner:
[6d1bc8938869]:
demo
Please enter the gitlab-ci tags for this runner (comma separated):
demo
Whether to run untagged builds [true/false]:
[false]: true
Whether to lock the Runner to current project [true/false]:
[true]: true
Registering runner... succeeded runner=eF8wyziz
Please enter the executor: parallels, shell, ssh, virtualbox, docker+machine, do
cker, docker-ssh, docker-ssh+machine, kubernetes:
docker
Please enter the default Docker image (e.g. ruby:2.1):
microsoft/dotnet:latest
Runner registered successfully. Feel free to start it, but if it's running alrea
dy the config should be automatically reloaded!
root@ubuntu16:~#
```

根据提示输入信息

```
Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://git.xd5u.cn/
Please enter the gitlab-ci token for this runner:
eF8wyziz****2RhgTHMis
Please enter the gitlab-ci description for this runner:
[6d1bc8938869]:
```

```
Please enter the gitlab-ci tags for this runner (comma separated):
demo
Whether to run untagged builds [true/false]:
[false]: true
Whether to lock the Runner to current project [true/false]:
[true]: true
Registering runner... succeeded runner=eF8wyziy
Please enter the executor: parallels, shell, ssh, virtualbox, docker+machine,
docker, docker-ssh, docker-ssh+machine, kubernetes:
docker
Please enter the default Docker image (e.g. ruby:2.1):
microsoft/dotnet:latest
Runner registered successfully. Feel free to start it, but if it's running already
the config should be automatically reloaded!
```

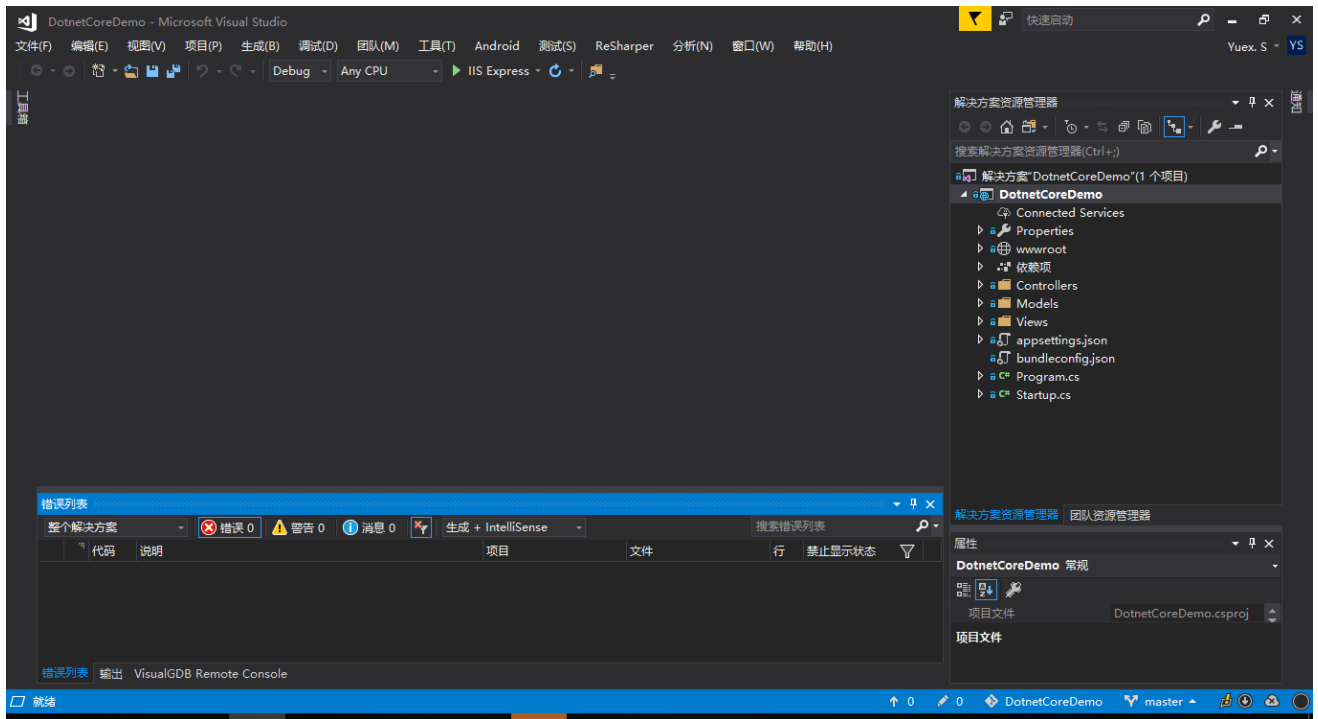
设置了默认源为 `microsoft/dotnet:latest` ,这时候Runners里面应该已经添加好了

类型	Runner 令牌	描述	版本	IP 地址	项目	作业	标签
<div>特定的</div> <div>锁定的</div>	10b197f5	6d1bc8938869	10.8.0	127.0.0.1	1	4	demo

## 4.添加一个dotnetcore测试项目

- 1.创建一个mvc示例





- 2.在gitlab创建一个项目,并且将刚创建项目提交上去

## DotnetCoreDemo

☆ 星标

0

🔗 派生

0

HTTP

http://git.xd5u.cn/Sample/Dotne

🔄

+

🔔 全局

文件(1.3 MB) 提交(2) 分支(1) 标签(0)

添加更新日志

添加许可证

添加贡献指南

添加 Kubernetes 集群

配置 CI/CD

master

DotnetCoreDemo /

+

历史

🔍 查找文件

Web IDE

🔄

 添加项目文件。  
由 YueX.S 提交于 2 分钟前

70b0fe95

名称	最后提交	最后更新
DotnetCoreDemo	添加项目文件。	2 分钟前
.gitattributes	添加 .gitignore 和 .gitattributes。	2 分钟前
.gitignore	添加 .gitignore 和 .gitattributes。	2 分钟前
DotnetCoreDemo.sln	添加项目文件。	2 分钟前

## 5.自动部署脚本.gitlab-ci.yml添加

- 1.添加.gitlab-ci.yml文件

DotnetCoreDemo

☆ 星标0

🔗 派生0

HTTP http://git.xd5u.cn/Sample/Dotne

📄

🔄

+

🔔 全局

文件(1.3 MB) 提交(2) 分支(1) 标签(0)

添加更新日志

添加许可证


添加贡献指南

添加 Kubernetes 集群

配置 CI/CD

master DotnetCoreDemo / +

历史 🔍 查找文件 Web IDE

 添加项目文件。  
由 Yuex.S 提交于 4 分钟前

70b0fe95

名称	最后提交	最后
DotnetCoreDemo	添加项目文件。	4 分
.gitattributes	添加 .gitignore 和 .gitattributes	4 分

还原包并且生成

master DotnetCoreDemo / .gitlab-ci.yml

 添加 .gitlab-ci.yml  
由 Yuex.S 提交于 不到 1 分钟前

✓ 这个 GitLab CI 配置是有效的。 [了解更多](#)

 .gitlab-ci.yml 144 Bytes

```
1 image: microsoft/aspnetcore-build
2 stages:
3   - build
4 build_job:
5   stage: build
6   only:
7     - master
8   script:
9     - dotnet restore
10    - dotnet build
```

```
# image: microsoft/aspnetcore-build
stages:
  - build
build_job:
  stage: build
  only:
    - master
  script:
    - dotnet restore
    - dotnet build
```

保存后就应该已经在运行中了

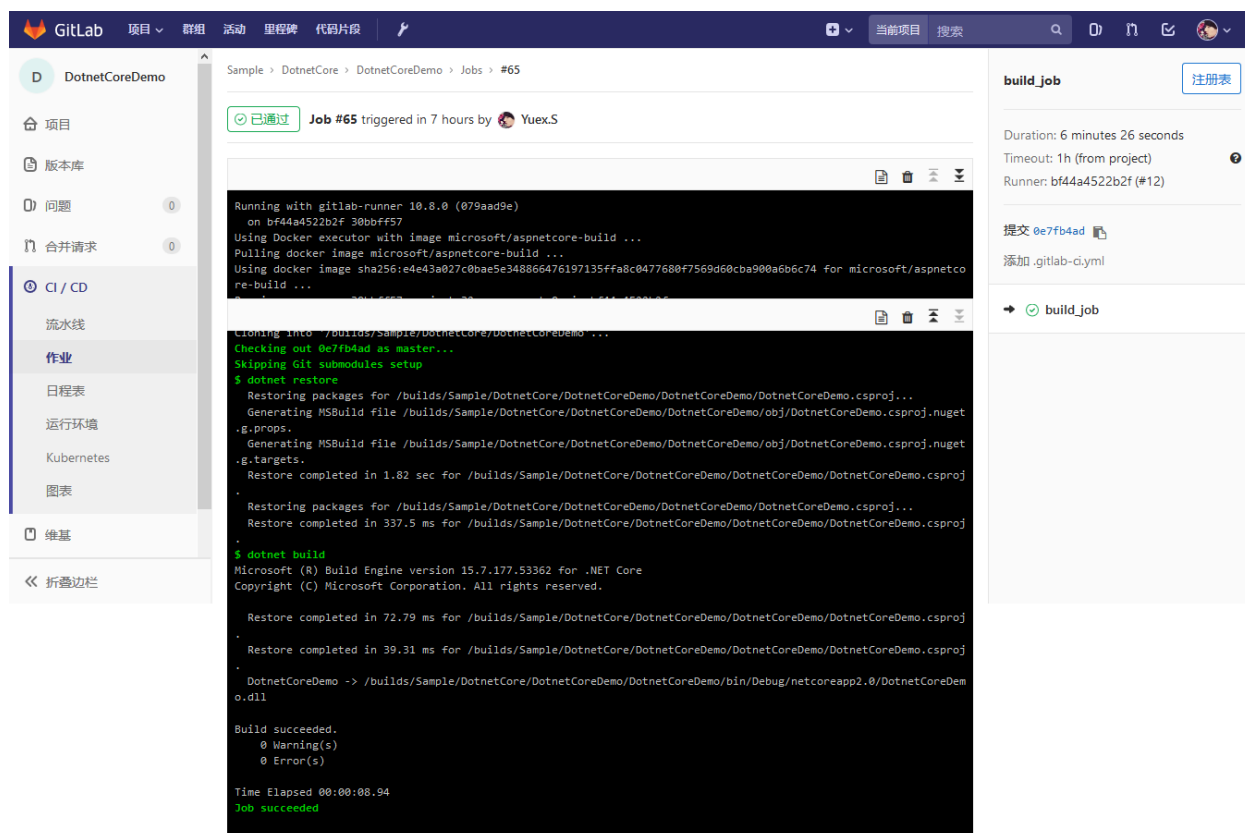
The screenshot displays the GitLab CI/CD interface for a project named 'DotnetCoreDemo'. On the left sidebar, the 'CI / CD' and '作业' (Jobs) sections are highlighted with red boxes. The main panel shows a table of jobs with the following data:

状态	作业	流水线	步骤	名称	时长
运行中	#65 于 master 分支 0e7fb4ad	#36 由 YueX.S	build	build_job	00:17

Below the table, a status bar indicates 'Job #65 triggered in 7 hours by YueX.S'. At the bottom, a terminal window shows the execution logs:

```
Running with gitlab-runner 10.8.0 (079aad9e)
on bf44a4522b2f 30bbff57
Using Docker executor with image microsoft/aspnetcore-build ...
Pulling docker image microsoft/aspnetcore-build ...
```

第一次部署,正在获取 `microsoft/aspnetcore-build` 镜像,这里如果很慢的话,可以使用阿里云镜像加速  
下面是完成后的截图



这里测试生成已经正常了,在这步可以还可以做部署测试等等,测试完成了再部署.

## 6.部署到web服务器

我这里使用阿里云作为部署服务器,再服务器安装好运行环境.我是ASP.NET Core+Nginx,下面开始搭建.

### 1.安装dotnet-sdk

参考官网,这里不赘述:<https://www.microsoft.com/net/learn/get-started/linux/ubuntu16-04>

```
root@iZhp3dp1181a3lwzy7vc24Z: ~  
Welcome to .NET Core!  
-----  
Learn more about .NET Core: https://aka.ms/dotnet-docs  
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli-docs  
  
Telemetry  
-----  
The .NET Core tools collect usage data in order to help us improve your experience. The data is anonymous and doesn't include command-line arguments. The data is collected by Microsoft and shared with the community. You can opt-out of telemetry by setting the DOTNET_CLI_TELEMETRY_OPTOUT environment variable to '1' or 'true' using your favorite shell.  
  
Read more about .NET Core CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry  
  
Configuring...  
-----  
A command is running to populate your local package cache to improve restore speed and enable offline access. This command takes up to one minute to complete and only runs once.  
Processing triggers for libc-bin (2.23-0ubuntu10) ...  
root@iZhp3dp1181a3lwzy7vc24Z:~#
```

## 2.安装Nginx并配置

- 安装nginx

```
sudo apt-get install nginx
```

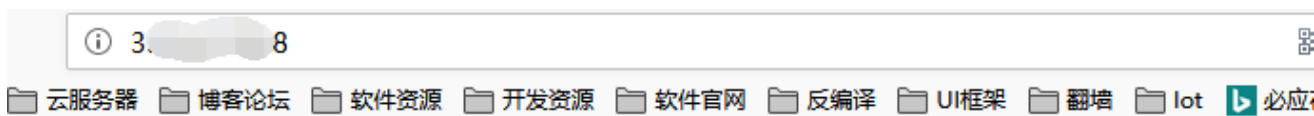
- 启动nginx

```
sudo service nginx start
```



```
root@iZhp3dp1181a3lw7vc24Z: ~  
Selecting previously unselected package nginx.  
Preparing to unpack .../nginx_1.10.3-0ubuntu0.16.04.2_all.deb ...  
Unpacking nginx (1.10.3-0ubuntu0.16.04.2) ...  
Processing triggers for man-db (2.7.5-1) ...  
Processing triggers for libc-bin (2.23-0ubuntu10) ...  
Processing triggers for ureadahead (0.100.0-19) ...  
Processing triggers for ufw (0.35-0ubuntu2) ...  
Processing triggers for systemd (229-4ubuntu21.2) ...  
Setting up fonts-dejavu-core (2.35-1) ...  
Setting up fontconfig-config (2.11.94-0ubuntu1.1) ...  
Setting up libfontconfig1:amd64 (2.11.94-0ubuntu1.1) ...  
Setting up libvpx3:amd64 (1.5.0-2ubuntu1) ...  
Setting up libxpm4:amd64 (1:3.5.11-1ubuntu0.16.04.1) ...  
Setting up libgd3:amd64 (2.1.1-4ubuntu0.16.04.8) ...  
Setting up libxslt1.1:amd64 (1.1.28-2.1ubuntu0.1) ...  
Setting up nginx-common (1.10.3-0ubuntu0.16.04.2) ...  
Setting up nginx-core (1.10.3-0ubuntu0.16.04.2) ...  
Setting up nginx (1.10.3-0ubuntu0.16.04.2) ...  
Processing triggers for libc-bin (2.23-0ubuntu10) ...  
Processing triggers for systemd (229-4ubuntu21.2) ...  
Processing triggers for ureadahead (0.100.0-19) ...  
Processing triggers for ufw (0.35-0ubuntu2) ...  
root@iZhp3dp1181a3lw7vc24Z:~# sudo service nginx start  
root@iZhp3dp1181a3lw7vc24Z:~#
```

- 启动后访问ip测试nginx是否已经正常



## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

3.现在环境已经搭建好了,继续配置nginx以及添加一个dotnetcore运行的服务和创建web路径

1.修改nginx的默认配置(实际可以添加配置文件,绑定域名,我这里就不分配域名了)

```
vim /etc/nginx/sites-available/default
修改为内容,主要是代理5000端口
server {
    listen      80;
    location / {
        proxy_pass      http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $http_host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

修改保存后重新加载配置文件

```
sudo nginx -t
sudo nginx -s reload
```

现在访问地址应该是502错误,不用管他,因为我们dotnetcore网站还没运行



## 2. 创建一个目录用于部署DotnetCoreDemo项目的目录

```
mkdir /var/www/DotnetCoreDemo
```

稍后网站以及服务会再这个目录执行

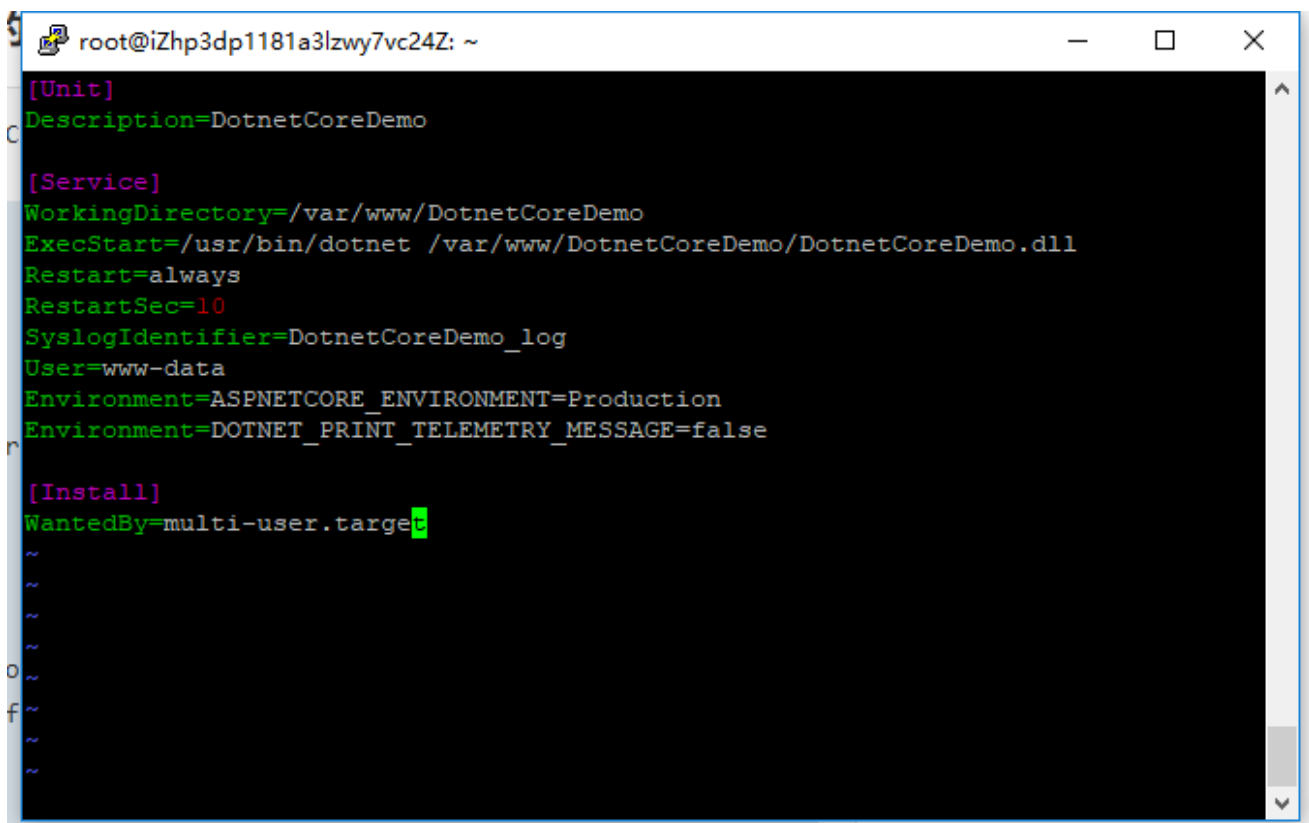
## 3. 创建一个运行DotnetCoreDemo网站的服务并启用

```
sudo vim /etc/systemd/system/kestrel-DotnetCoreDemo.service
内容为
```

```
[Unit]
Description=DotnetCoreDemo

[Service]
WorkingDirectory=/var/www/DotnetCoreDemo
ExecStart=/usr/bin/dotnet /var/www/DotnetCoreDemo/DotnetCoreDemo.dll # 路径根据自己项目来设置
Restart=always
RestartSec=10
SyslogIdentifier=DotnetCoreDemo_log
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```



```
root@iZhp3dp1181a3lzw7vc24Z: ~
[Unit]
Description=DotnetCoreDemo

[Service]
WorkingDirectory=/var/www/DotnetCoreDemo
ExecStart=/usr/bin/dotnet /var/www/DotnetCoreDemo/DotnetCoreDemo.dll
Restart=always
RestartSec=10
SyslogIdentifier=DotnetCoreDemo_log
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

```
# 启用服务(这里只是启用了,并没有启动)
systemctl enable kestrel-DotnetCoreDemo.service
```

## 4.配置服务器ssh免密码登陆,在脚本中会使用到

在本地服务器创建RSA无密码密钥,并添加远程授权

```
ssh-keygen -t rsa -P ''
ssh-copy-id root@阿里云部署服务器IP
```

复制私钥添加到项目变量SSH\_PRIVATE\_KEY\_DEV

```
cat /root/.ssh/id_rsa

root@iZhp3dp1181a3lzw7vc24Z: ~
1yaCxCCRCPBVO4U9L5AdKif8zEcs+sZrfPEpyRpk+sNulRtD2Y5Jl8xwNCL2NGOL
Zxjaqdelcn/sXzO5PRNhXXJlq239lRNnpOghnbzxnIalXGFT3rdXcNSsvzEq3lX
Z6cl2LaXlPduYa105YOwmPQp3rHUozO8Fy0sLwIDAQABAoIBABFWtAse7Zlacb0D
xCoPFZV2h2SxrKCUIDKt/Rvvr6nrWMV85nI39mbkQTOgg5AtbFcCMpHFRmu0hS4c
BMYVmFULAO6KYRwNieXXFaDdElRH6kkTiFlOTy1L3riLKdt/XgsjMT5097i6zNpC
J0eJVjHTkPjAxYOj8bTwmfL64wfw3RzMyhZG9930ZkjaWu2bKcwMBjzP2s8PUKCH
LxuicklbRtOoHOiBNLLmAku6QjtvNH46x6Cqn7l2DAKVC+PAnZ4QtwrFpOraDMJl
tDTJ6y8Aw98zWf6Lrrlo8iHlthillUCJNIMXdzK4HggT6OIwYp+H34oXfiiqRGfs
rXTBDkOkCgYEA/RtLlvyvr/bZK0PjVwMSS56fgMvwTdvu/6QHMeOEpmPz03RT8sdq
sOorSqYlqztn3Plmb/mYqfHEBMzj7XANRwgHWgSn4xeKL9FVAu8ViSy4+aTeHbFJ
Sx9pdtojclZl2mWZWejtTN6k3Lj9S5c/jAzD2luD9Z3GDx6fCdfexnUCgYEA7KF0
W9OVme9JdCJqI3yYSrxulIKUreOiq1Fhekfosf+2B0btG/zsOmGQ/80IkR+omhlA
yn+Q7jmgaRziH+MNTPOemjd+/SNceLh+G6NXiKz4RzD09ZlAlzLGAA5OV2ZkwlBJ
dIuj9W3Z0iiTpflqIVp9KISk83fEV7fgls5re5MCgYEAst/TnUBBWDJwEC+OPzQg
fejAeNmoHp5MLcbSfuN4H8+Lxej0WyOrtxQHPF5yl20IXhmHh6MxzQlzpKDYhDko
aQa3ilq2+7rrIhgiqEwnQCanI2bswrdeQR8V/bT0TvfYeaFb2zrOWgEPKh5ihEWD
l4o979elTqQOESlm5CH3gvECgYAy7oV3B1Ga8nAi+QlTWgwFegl43CpYSzjGGxwa
D8Q8G8f4RQHgFk5aOonVJTKE2gWWNlmtvEtbmzelqCvbpSlcGlul4eAJaEUNDtjF
5iB/IIvH9veiq6xWpW0MsgeWtElBTyqgVTYTK3PI0kcFYGvBhhKOWA6RAORX5RII
I6HqlQKBgB9xNMVL05sBp9FImNMDt8IbMCnxUF9YNS4FXysQsSm3U85BWKoi3w7
nMPHoVTg+/PjsrILUu5RTc4urBdtxnoYhbHRjFqCQwu5/VQcRP6MPEu0I4Qe2P2p
QnCh/2Nrm4a2YDq9l+jJW9/uwUQ53RKXKoVsu8U4a8734Wj0+TGf
-----END RSA PRIVATE KEY-----
root@iZhp3dp1181a3lzw7vc24Z:~#
```

GitLab

项目

群组

活动

里程碑

代码片段

当前项目

搜索

DotnetCoreDemo

项目

版本库

问题

合并请求

CI / CD

维基

代码片段

设置

常规

成员

Badges

集成

Sample > DotnetCore > DotnetCoreDemo > CI / CD 设置

常规流水线设置

Access your runner token, customize your pipeline configuration, and view your pipeline status and coverage report.

展开

Auto DevOps (Beta)

Auto DevOps will automatically build, test, and deploy your application based on a predefined Continuous Integration and Delivery configuration. [Learn more about Auto DevOps](#)

展开

Runners 设置

注册和查看本项目的 runner

展开

加密变量

变量通过runner作用于环境中。可将变量限制为仅受保护的分支或标签可以访问。可以使用变量来保存密码、密钥或任何其他内容。

SSH\_PRIVATE\_KEY\_DEV

-----END RSA PRIVATE KEY-----

受保护

输入变量的名称

输入变量的值

受保护

保存变量

隐藏值

继续在Gitlab添加变量

- DEPLOY\_SERVER\_DEV ,服务器部署的地址
- KESTREL\_SERVICENAME ,服务器部署的dotnet网站的服务名称
- WEB\_DIR ,服务器网站部署的路径

## 加密变量

变量通过runner作用于环境中。可将变量限制为仅受保护的分支或标签可以访问。可以使用变量来保存密码、密钥或

SSH_PRIVATE_KEY_DEV	-----END RSA PRIVATE KEY-----	受保护
DEPLOY_SERVER_DEV	30.1.1.8	受保护
KESTREL_SERVICENAME	kestrel-DotnetCoreDemo.service	受保护
WEB_DIR	/var/www/DotnetCoreDemo	受保护
输入变量的名称	输入变量的值	受保护

保存变量

隐藏值

修改脚本.gitlab-ci.yml文件

```
# 指定镜像 microsoft/aspnetcore-build暂时没有sdk2.1的编译环境,暂时不需要
# image: microsoft/aspnetcore-build
stages:
  - build
  - deploy_dev

before_script:
  # Install ssh-agent if not already installed, it is required by Docker.
  # (change apt-get to yum if you use a CentOS-based image)
  - 'which ssh-agent || ( apt-get update -y && apt-get install openssh-client -y )'

  # Run ssh-agent (inside the build environment)
  - eval $(ssh-agent -s)

  # Add the SSH key stored in SSH_PRIVATE_KEY variable to the agent store
  # error: https://gitlab.com/gitlab-examples/ssh-private-key/issues/1
  # - echo "$SSH_PRIVATE_KEY_DEV"
  - ssh-add <(echo "$SSH_PRIVATE_KEY_DEV")

  # For Docker builds disable host key checking. Be aware that by adding that
  # you are susceptible to man-in-the-middle attacks.
  # WARNING: Use this only with the Docker executor, if you use it with shell
  # you will overwrite your user's SSH config.
  - mkdir -p ~/.ssh
  - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
build_job:
```

```

stage: build
only:
  - master
script:
  - dotnet restore
  - dotnet build
deploy_dev_job:
  stage: deploy_dev
  environment:
    name: development
  only:
    - master
  script:
    # 发布程序
    - dotnet publish -c Release --output /publish
    # 停止服务器网站的服务
    - ssh root@$DEPLOY_SERVER_DEV "systemctl stop $KESTREL_SERVICENAME"
    # scp复制发布文件到服务器
    - scp -r /publish/* root@$DEPLOY_SERVER_DEV:$WEB_DIR
    # 启动服务器的服务
    - ssh root@$DEPLOY_SERVER_DEV "systemctl start $KESTREL_SERVICENAME"

```

状态	作业	流水线	步骤	名称	
🟢 已通过	#82 🟢 master → 2070d05c	#42 by	deploy_dev	deploy_dev_job	🕒 00:44 📅 不到 1 分钟前
🟢 已通过	#81 🟢 master → 2070d05c	#42 by	build	build_job	🕒 05:38 📅 不到 1 分钟前

```

Installing Microsoft.Build.Runtime 15.3.409.
Installing NETStandard.Library 2.0.1.
Installing Microsoft.NETCore.DotNetHostPolicy 2.0.7.
Installing Microsoft.Extensions.FileSystemGlobbing 2.0.1.
Installing Microsoft.Extensions.FileProviders.Abstractions 2.0.1.
Installing System.Text.Encoding.CodePages 4.0.1.
Installing System.Reflection.Metadata 1.3.0.
Installing Microsoft.Build.Framework 15.3.409.
Installing Microsoft.Build.Utilities.Core 15.3.409.
Installing Microsoft.Build.Tasks.Core 15.3.409.
Installing Microsoft.Build 15.3.409.
Installing Microsoft.NETCore.DotNetHostResolver 2.0.7.
Installing Microsoft.Extensions.Primitives 2.0.0.
Installing System.Diagnostics.Process 4.1.0.
Installing System.Threading.Thread 4.0.0.
Installing System.Diagnostics.TraceSource 4.0.0.
Installing System.Collections.NonGeneric 4.0.1.
Installing System.Runtime.Serialization.Xml 4.1.1.
Installing System.Resources.Reader 4.0.0.
Restore completed in 6.58 sec for /builds/Sample/DotnetCore/DotnetCoreDemo/DotnetCoreDemo/DotnetCoreDemo.csproj
.
DotnetCoreDemo -> /builds/Sample/DotnetCore/DotnetCoreDemo/DotnetCoreDemo/bin/Release/netcoreapp2.1/DotnetCoreD
emo.dll
DotnetCoreDemo -> /builds/Sample/DotnetCore/DotnetCoreDemo/DotnetCoreDemo/bin/Release/netcoreapp2.1/DotnetCoreD
emo.Views.dll
DotnetCoreDemo -> /publish/
$ ssh root@$DEPLOY_SERVER_DEV "systemctl stop $KESTREL_SERVICENAME"
Warning: Permanently added '39.104.122.8' (ECDSA) to the list of known hosts.
$ scp -r /publish/* root@$DEPLOY_SERVER_DEV:$WEB_DIR
$ ssh root@$DEPLOY_SERVER_DEV "systemctl start $KESTREL_SERVICENAME"
Job succeeded

```

#### deploy\_dev\_job

Duration: 44 seconds  
Timeout: 1h (from project)  
Runner: bf44a4522b2f (#12)

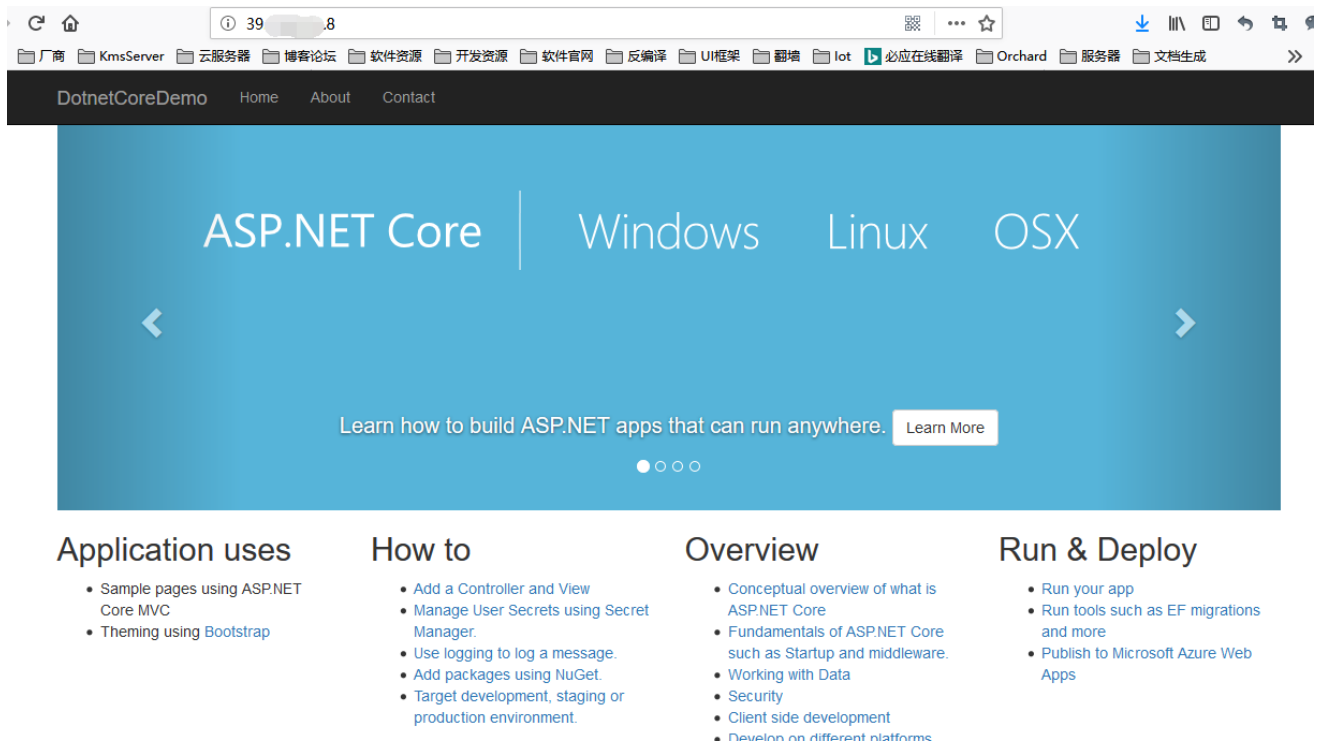
提交 2070d05c

更新 .gitlab-ci.yml

🟢 流水线 #42 来自 master

deploy\_dev

➔ 🟢 deploy\_dev\_job



到这里,自动化部署全部完成.其他方式类似操作.

2018年6月4日 Devil月哥