مستندات

طراحی پایپلاین CI/CD

تاریخ تنظیم	۱۳۹۹/۰۷/۲۰
نسخه	۰.۱

به نام خدا

فهرست مطالب

۱ مقدمه

فرآیند CI/CD یا Continuous Integration و Continuous Delivery با هدف خودکارسازی فرایند چرخه سورس‌کد به محصول پیاده‌سازی می‌شود. Continuous Integration می‌تواند مراحل مختلفی را مانند Build ، Test ، بررسی کیفیت و ... شامل شود. در مورد Continuous Delivery می‌توان گفت که پس از مراحل CI یک نسخه پایدار به تیم اجرا تحویل داده می‌شود. از طرف دیگر Continuous Deployment زمانی اطلاق می‌شود که همه مراحل ذکرشده انجام شود و سپس در محیط عملیاتی (Production) «مستقر» شود.

در این نوشتار سعی شده است پایپلاین CI/CD برای یک سورس ساده PHP با استفاده از Docker و Gitlab CI/CD انجام شود.

طراحی با این پیش‌فرض انجام شده است که کانفیگ‌های سرور Build و Gitlab Runner وجود دارد.

۲ سورس‌کد

ابتدایی‌ترین نیاز برای طراحی پایپلاین، استفاده از کنترل ورژن است. در این مستند از ورژن کنترل git و زیرساخت Gitlab استفاده می‌شود.

۳ داکرایز

برای Dockerize کردن پروژه، `Dockerfile` بشرح زیر تعریف می‌شود:

```
FROM php:7.4-apache

COPY . /var/www/html

VOLUME [ "/var/www/html/config" ]

VOLUME [ "/var/www/html/uploads" ]

RUN chown -R www-data:www-data /var/www/html
```

۴ کانفیگ Gitlab CI/CD

برای کانفیگ Gitlab CI/CD لازم است فایلی با عنوان[۱] `gitlab-ci.` در پروژه قرارگیرد. همچنین می‌توان کانفیگ‌های مختلفی را به ازای هر برنچ استفاده کرد.

فایل `gitlab-ci.` در فرمت YAML تعریف می‌شود.

```
stages:
  - test
  - build
  - deploy
test:
  image: php:7.4
  stage: test
  before_script:
  - curl --location --output /usr/local/bin/phpunit
      https://phar.phpunit.de/phpunit.phar
  - chmod +x /usr/local/bin/phpunit
  script:
    - phpunit --configuration phpunit.xml
build:
  image: docker
  stage: build
  before_script:
    - docker login $CI_REGISTRY -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
  script:
    - docker build -t $CI_REGISTRY_IMAGE .
    - docker push $CI_REGISTRY_IMAGE
```

[۱] امکان تغییر نام در تنظیمات پروژه وجود دارد.

```
# Ref: https://docs.gitlab.com/ee/ci/ssh_keys/

deploy:
  image: alpine
  stage: deploy
  before_script:
    - apk add --no-cache openssh
    - echo "$PROD_SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - ssh-keyscan $PROD_SSH_ADDRESS >> ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts
    - docker login $CI_REGISTRY -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
  script:
    - ssh PROD_SSH_USER@$PROD_SSH_ADDRESS "docker login $CI_REGISTRY -u
        $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD"
    - ssh PROD_SSH_USER@$PROD_SSH_ADDRESS "docker pull $CI_REGISTRY_IMAGE"
    - ssh PROD_SSH_USER@$PROD_SSH_ADDRESS "docker stop web-app && docker rm
        web-app"
    - ssh PROD_SSH_USER@$PROD_SSH_ADDRESS "docker run -d -v
        /data/web-app/config:/var/www/html/config -v
        /data/web-app/uploads:/var/www/html/uploads --restart=always -p 8080:80
        $CI_REGISTRY_IMAGE --name web-app"
```

برای پیاده‌سازی از سه مرحله استفاده می‌شود:

- Test
- Build
- Deploy

۱.۴ Continuous Integration

۱.۱.۴ پیاده‌سازی Test

برای پیاده‌سازی تست، از ابزار PHPUnit استفاده شده است. اجرای PHPUnit در محیط داکر صورت می‌گیرد، بدینصورت که ایمیج `php:7.4` گرفته شده و دستورات زیر برای دانلود phpunit استفاده می‌شود:

```
curl --location --output /usr/local/bin/phpunit
    https://phar.phpunit.de/phpunit.phar
chmod +x /usr/local/bin/phpunit
```

سپس تست‌ها انجام می‌شود:

```
phpunit --configuration phpunit.xml
```

امکان بررسی کیفیت کد (Code Quality) نیز با ابزارهایی مانند SonarCube امکان‌پذیر است.

۲.۴ Continuous Delivery

۱.۲.۴ پیاده‌سازی Build

در صورت موفقیت‌آمیز بودن مرحله قبل، در اینجا باید عملیات Build ایمیج انجام شود. در ابتدا شل در رجیستری که مدنظر است لاگین می‌کند. درحال حاضر امکان استفاده از رجیستری گیت‌لب (اصلی) و یا پیاده‌سازی رجیستری در گیت‌لب لوکال نیز امکان‌پذیر است. آدرس رجیستری در متغیر محیطی[۲] `$CI_REGISTRY` ثبت شده است. احرازهویت بصورت خودکار و با پارامترهای `$CI_ REGISTRY_USER` و `$CI_REGISTRY_PASSWORD` انجام می‌شود.

[2] Environment Variable

پس از لاگین، ایمیج با استفاده از `Dockerfile` بیلد شده و در مرحله بعد داخل رجیستری Push می‌شود.

۳.۴ Continuous Deployment

۱.۳.۴ پیاده‌سازی Deploy

این مرحله با استفاده از SSH انجام می‌شود لذا ضروری است پارامترهای زیر در کانفیگ پروژه اضافه شوند:

- آدرس سرور پروداکشن: `PROD_SSH_ADDRESS`
- یوزرنیم: `PROD_SSH_USER`
- کلیدخصوصی برای اتصال سرور پروداکشن: `PROD_SSH_PRIVATE_KEY`

برای این مرحله از ایمیج Alpine استفاده می‌شود، البته بسته به کانفیگ runner می‌توان بدون ایمیج هم فرآیند را انجام داد، اما بدلیل تاثیرات جانبی توصیه نمی‌شود.

زمانی که این مرحله شروع بکار می‌کند، پکیج `openssh` نصب شده و کلید خصوصی اتصال به پروداکشن کپی و کلید عمومی سرور با استفاده از `ssh-keyscan` ذخیره می‌شود.

برای مراحل بعدی، هربار یک کانکشن SSH به سرور پروداکشن زده و دستورات زیر اجرا می‌شود:

- احراز هویت در رجیستری
- دریافت ایمیج جدید
- متوقف‌سازی و حذف کانتینر قبلی
- شروع کانتینر با ایمیج جدید

لازم به ذکر برای سادگی می‌توان از docker-compose استفاده کرد. همچنین درصورت استفاده از کلاستر کوبرنتیر مراحل دپلوی ساده‌تر خواهدشد.