GitLab + Dokku で作る CI/CD 環境

Kazuhiro NISHIYAMA

第*78*回 *Ruby*関西 勉強会

2017/07/29

自己紹介

- 西山和広
- id:znz (github, twitter など)
- Ruby コミッター

GitLab + Dokku

- GitLab
- GitLab CI
- Dokku
- (+ Heroku)

GitLab とは?

- 簡単にいえば OSS の GitHub クローンのようなもの
- Git ホスティング
- Merge Request (GitHub の Pull Request)
- Issue 管理など色々
- GitHub にない機能もある
- https://about.gitlab.com/features/

GitLab CI とは?

- Continuous Integration/Continuous Delivery (CI/CD)
- Jenkins のジョブを実行しないマスターのようなものが GitLab に組み込み
- GitLab Runner (Jenkins の slave のようなもの) を動かすマシンが別途必要
- repository の .gitlab-ci.yml で設定 (.travis.yml などと同様)
- https://about.gitlab.com/features/gitlab-ci-cd/

GitLab Runner とは?

- マシン上で直接実行する (Shell executor)
- Docker の中で実行する (Docker executor)
  - これを使用
- その他
- https://docs.gitlab.com/runner/

Dokku とは?

- Docker を使った OSS のミニ Heroku (PaaS)
- ssh 経由の git で deploy できる
- http://dokku.viewdocs.io/dokku/

組み合わせた状態

- ブランチに push → Dokku に Review App を deploy
- Merge Request をマージ → Review App を停止
- master に push → Staging に deploy
- 確認後、クリックで Production に deploy
- https://about.gitlab.com/features/review-apps/

deploy 例

- https://docs.gitlab.com/ce/ci/examples/deployment/README.html では dpl gem で heroku に deploy する例がある
- Dokku との組み合わせは独自研究

組み合わせ方

- .gitlab-ci.yml で設定する
- deploy 用の ssh 秘密鍵などは secret variables に設定

.gitlab-ci.yml の設定例

使用する docker image を指定

```
image: ruby:2.3.3
```

Rails アプリなので https://hub.docker.com/r/_/ruby/ を使用

cache

per-branch caching:

```
cache:
  key: "$CI_COMMIT_REF_NAME"
  untracked: true
```

https://docs.gitlab.com/ce/ci/yaml/#cache-key

テスト用 variables

services で指定する postgres image で使用
(DATABASE_URL は Rails で使用)

```
variables:
  # for test
  POSTGRES_DB: dbname
  POSTGRES_USER: dbuser
  POSTGRES_PASSWORD: dbpass
  DATABASE_URL: "postgres://dbuser:dbpass@postgres:5432/dbname"
```

https://hub.docker.com/r/_/postgres/

deploy 用 variables

```
# for deploy
DOKKU: ssh dokku@$DOKKU_HOST
APP_NAME: $CI_ENVIRONMENT_SLUG
DB_NAME: $CI_ENVIRONMENT_SLUG-database
```

- DOKKU はあとで短く表記するため
- APP_NAME は Dokku でのアプリ名 (サブドメイン名)
- DB_NAME はデータベースコンテナ名 (内部用なので識別できれば何でも良い)

stages

```
stages:
  - test
  - review
  - staging
  - production
```

- ブランチに push → test → review
- master に push → test → staging → production (後述の例 (1))
- master に push → test → staging / タグを push → test → production (後述の例 (2))

before_script

```
before_script:
  - 'apt-get update -qq && apt-get -o dir::cache::archives="/cache/apt"
    install -y -qq sqlite3 libsqlite3-dev nodejs'
  - gem install bundler --no-ri --no-rdoc
  - bundle install --jobs $(nproc) --path=/cache/bundler
  - ln -nfs .test.env .env
```

- 開発環境に合わせてテスト環境でも sqlite3 が必要
- js runtime も必要なので nodejs
- インストール時に /cache を使用
- dotenv (dotenv-rails) でテスト用環境変数設定

.before_ssh

```
.before_ssh: &before_ssh
  # https://docs.gitlab.com/ce/ci/ssh_keys/README.html
  - 'which ssh-agent || ( apt-get update -y && apt-get -o
  dir::cache::archives="/cache/apt" install -y openssh-client )'
  - eval $(ssh-agent -s)
  - ssh-add <(echo "$SSH_PRIVATE_KEY")
  - mkdir -p ~/.ssh
  # Set `ssh-keyscan $DOKKU_HOST` to SSH_SERVER_HOSTKEYS
  - '[[ -f /.dockerenv ]] && echo "$SSH_SERVER_HOSTKEYS" > ~/.ssh/known_hosts'
  - '[[ -f /.dockerenv ]] && echo "$SSH_CONFIG" > ~/.ssh/config'
```

- .で始まるキーは後で参照する用途に使える
- Dokku や Heroku に ssh で git push するときの前処理
- secret variables から ssh-agent に秘密鍵と known_hosts と ssh config を設定

.deploy_script

```
.deploy_script: &deploy_script
  - $DOKKU apps:create $APP_NAME || echo $?
  # require `sudo dokku plugin:install https://github.com/dokku/dokku-postgres`
  - $DOKKU postgres:create $DB_NAME || echo $?
  - $DOKKU postgres:link $DB_NAME $APP_NAME || echo $?
  - $DOKKU config:set --no-restart $APP_NAME TZ=Asia/Tokyo
    RAILS_SERVE_STATIC_FILES=1 NO_FORCE_SSL=1 RACK_DEV_MARK_ENV=review
  - git push dokku@$DOKKU_HOST:$APP_NAME HEAD:refs/heads/master
  - $DOKKU -tt run $APP_NAME bundle exec rake db:seed
```

- app と db がなければ作成
- TZ などの環境変数設定
- HEAD:refs/heads/master という指定で push
- -tt で強制的に tty を確保して rake db:seed

rake test

```
rake:
  stage: test
  services:
    - postgres:latest
  script:
    - bundle exec rake db:setup RAILS_ENV=test
    - bundle exec rake
```

- services に指定した postgres image とリンクした状態で実行
- データベースの初期設定をしてテスト実行

staging deploy (1)

```
staging:
  stage: staging
  variables:
    APP_NAME: hello-app-staging.example.jp
  before_script: *before_ssh
  script:
  - git push dokku@$PRODUCTION_DOKKU_HOST:$APP_NAME HEAD:refs/heads/master
  environment:
    name: staging
    url: https://hello-app-staging.example.jp/
  only:
    - master
```

- before_script は sqlite3 のインストールなどの代わりに ssh 設定
- Pipelines の Environments からリンク
- master に push したときのみ

production deploy (1)

```
production:
  stage: production
  variables:
    APP_NAME: hello-app.example.jp
  before_script: *before_ssh
  script:
  - git push dokku@$PRODUCTION_DOKKU_HOST:$APP_NAME HEAD:refs/heads/master
  environment:
    name: production
    url: https://hello-app.example.jp/
  when: manual
  only:
    - master
```

- staging の後に手動実行
- master に push したときのみ

staging deploy (2)

```
staging:
  stage: staging
  variables:
    APP_NAME: hello-app-staging
  script:
  - gem install dpl
  - dpl --provider=heroku --app=$APP_NAME --api-key=$HEROKU_STAGING_API_KEY
  environment:
    name: staging
    url: https://$APP_NAME.herokuapp.com/
  only:
  - master
```

- master に push したときのみ
- dpl で heroku に deploy

production deploy (2)

```
production:
  stage: production
  variables:
    APP_NAME: hello-app
  script:
  - gem install dpl
  - dpl --provider=heroku --app=$APP_NAME --api-key=$HEROKU_PRODUCTION_API_KEY
  environment:
    name: production
    url: https://$APP_NAME.herokuapp.com/
  only:
  - tags
```

- タグを push したときのみ
- dpl で heroku に deploy

review deploy

```
review:
  stage: review
  before_script: *before_ssh
  script: *deploy_script
  environment:
    name: review/$CI_COMMIT_REF_NAME
    url: http://$CI_ENVIRONMENT_SLUG.$DOKKU_DOMAIN
    on_stop: stop_review
  only:
    - branches
  except:
    - master
```

- review 用の Dokku アプリを deploy
- master 以外のブランチに push したときのみ

stop review

```
stop_review:
  stage: review
  variables:
    GIT_STRATEGY: none
  before_script: *before_ssh
  script:
    - $DOKKU apps:destroy $CI_ENVIRONMENT_SLUG --force || echo $?
    - $DOKKU postgres:destroy $CI_ENVIRONMENT_SLUG-database --force || echo $?
  environment:
    name: review/$CI_COMMIT_REF_NAME
    action: stop
  when: manual
  only:
    - branches
  except:
    - master
```

- postgres は使用中だとエラーになるので apps から停止
- リンクも消える

まとめ

- GitLab と Dokku を組み合わせて CI/CD 環境を作成する例を紹介
- 環境構築に使っている Ansible Playbook は https://github.com/znz/ansible-playbook-gitlab-dokku
- ブログ記事は http://blog.n-z.jp/blog/categories/gitlab/