「GitLab CI」で簡単コンテナ開発

Victor Adossi

June, 2019

トピック

- コンテナとは？
- GitLab CI のコンテナフィーチャー
- コンテナビルド (GitLab + `dind`)
- コンテナ保存 (GitLab レジストリー)
- コンテナデプロイ (Gitlab CI)
- `docker` なしのコンテナ未来

コンテナとは？

Linux のコンテナは**プロセスを封じ込める**ツールです。

バーチャルマシン (“VM”) と違って、まったく新しいコンピューター作らず、プロセスの環境のコントロールで封じ込むんです。

本当は色々の Linux カーネルのフィーチャーを使ってコンテナと言うことが使えるようになりました。一番大事の二つは:

- ネームスペース (プロセスが見えるリソース)
- シーグループ (プロセスが使えるリソース)

一つのプロセスが見える世界 (ファイルシステム、他のプロセス、デバイス) をコントロール出来るから、封じ込むことが出来ます。

GitLab CI のコンテナフィーチャー

GitLab CI を使ってるなら、無料で色々のコンテナに関してのフィーチャーが使えます:

- コンテナ保存 (レジストリー)
- コンテナビルド出来る CI
- セキュリティスキャン
- Kubernetes やコンテナ専門のインフラシステムまでデプロイ
- ...

コンテナビルド

簡単に CI のパイプラインの中でコンテナがビルド出来ます。

```
build_container:
  stage: build
  image: docker # ドッカーのバイナリが持っているイメージ
  services:
    - docker:dind # "Docker in docker" (ドッカーの中にドッカー)
  only: # リリースタッグとかリリース前のブランチだけに実行
    - /v[0-9|\.]+/ # release tags
    - /pre-release-v[0-9|\.]+/ # pre-release branches
  except: # 他のブランチで実行しない
    - branches
  script:
    - apk add --update ca-certificates make nodejs nodejs-npm
    - npm install
    - make docker-image
```

メークファイルの中には:

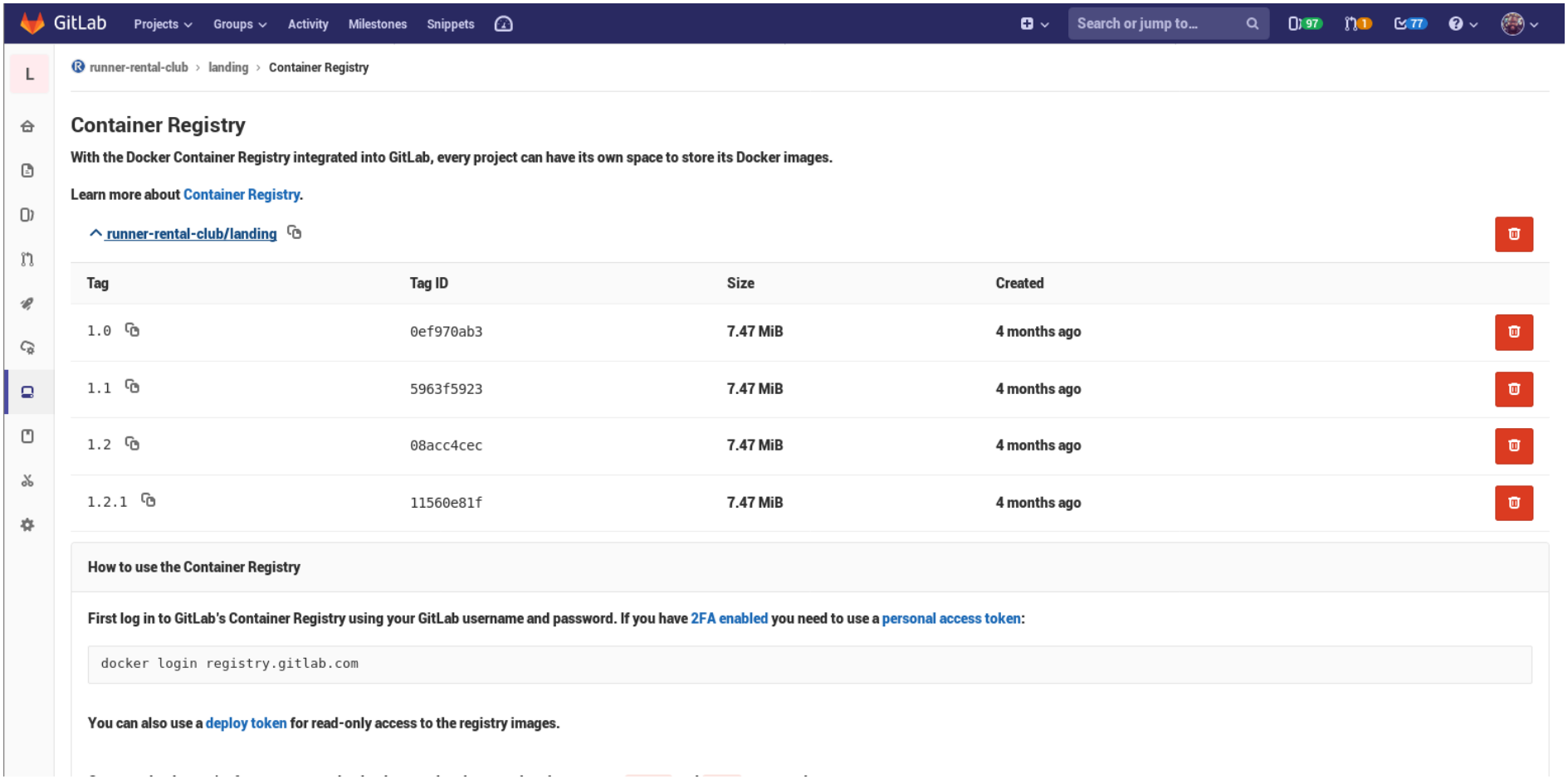
```
check-tool-docker:
ifndef DOCKER
    $(error "`docker` バイナリーがありません,`docker`をインストールしてください。(https://docs.docker.com)")
endif

docker-image: check-tool-docker
    docker build -f infra/docker/Dockerfile -t registry.gitlab.com/user/project:1.0.0 .
```

dind サービスと言う GitLab のフィーチャーを使って、CI のステップの中で docker が実行出来ます。

コンテナ保存

無料で使えられるコンテナレジストリー



コンテナ保存 (続)

ビルド後レジストリーまでパブリッシュも簡単です:

```
publish:
  stage: publish
  image: docker
  services:
    - docker:dind
  only:
    - /v[0-9|\.]+/ # release tags
    - /pre-release-v[0-9|\.]+/ # pre-release branches
  except:
    - branches
  script:
    - apk add --update ca-certificates make nodejs nodejs-npm
    - npm install
    - docker login -u gitlab-ci-token --password $CI_BUILD_TOKEN registry.gitlab.com
    - make docker-image publish-docker-image
```

CI_BUILD_TOKEN は GitLab CI が毎回作ってくれるレジストリーまでプッシュのために使えるトークン。

メークファイルの中のターゲット:

```
docker-image: ... # 変わりなし

publish-docker-image: check-tool-docker
    docker push registry.gitlab.com/user/project:1.0.0
```

コンテナデプロイ

色々のコンテナのデプロイ方があります:

- ベアメタルサーバー/インスタンス
- クラウド (AWS ElasticBeanstalk, Elastic Container Service, Google Cloud Platform, Azure Container Service)
- Kubernetes, Mesos, Docker Swarm

どっちにしても効率がいいデプロイプロセス欲しいなら:

- どこかでコンテナをビルド
- どこかのレジストリーにコンテナをプッシュ
- どこかの CI を使って選んだプラットフォームを実行させる

その三つの「どこか」が全部「GitLab」に出来ます!

コンテナデプロイ (続)

ビルドとパブリッシュする後デプロイはチーム次第:

```
deploy_staging:
  stage: deploy
  image: alpine
  only:
    - web # ウェブだけから実行出来る
  script:
    - apk add --update ca-certificates make openssh ...
    - make deploy
    # - ssh ....
    # - kubectl apply ....
    # - awscli ....
    - make deploy-check
    - make deploy-alert
```

BONUS: CI のステップの中で使える「サービス」とは?

Docker のコンテナ何でも使えますシステムです。

```yaml
e2e:
  stage: test
  only:
    - merge_requests # マージリクエストだけ実行する
  services:
    - name: postgres:10.4-alpine # データベース
    - name: jeanberu/mailcatcher:latest # メール
      alias: mailcatcher
    - name: minio/minio:latest # S3 見たいのファイルストア
      alias: minio
      command: ["server", "/data"] # 特別のコマンド
  variables:
    # Postgres の設定
    POSTGRES_DB: db_name
    POSTGRES_USER: db_user
    POSTGRES_PASSWORD: db_password
    DB_HOST: postgres
    # Minio の設定
    MINIO_ACCESS_KEY: badkey
    MINIO_SECRET_KEY: badsecret
    # UGC
    S3_HOST: minio
    # Mailer の設定
    MAILER_SMTP_HOST: mailcatcher
    MAILER_SMTP_PORT: 1025
  script:
    - apk add --update make
    - make setup build test-e2e # 実行する時に全部の上に書いてあるサービスを走ってる間 E2E テストを実行
```

終わり

今日の皆さん時間頂いてありがとうございます。

whoami

アドッシー　ビクトル (vados@gaisma.co.jp, vados@vadosware.io)

GPG: ED874DE957CFB552

日本で「ガイスマ」と言うコンサルティング会社をやっています、

プレゼンについて補正とか質問がある方ぜひメールでメッセージしてください。

GitLab とか他の件についてのご相談がした方はぜひメールしてください。

こう言う技術的のコンテントが好きなら、個人的のブログもやっています
(https://vadosware.io)