DockerでLLVMビルド

方針

Dockerfileを利用して、LLVMをビルドする。

- Dockerコマンドの簡単な使用方法を知る。
- DockerイメージはUbuntu 18.04 と CentOS 8 を扱う。
  それぞれのDockerイメージに合わせたDockerfileを用意しているので、それを使用する。

やらないこと

- Dockerの解説
  Dockerとは何かの詳細な解説や、使用することの具体的なメリットなどには触れません。
- Dockerfileの解説
  Dockerfileに関する記述方法などには触れません。
  (※ 後日、今回使用するDockerfileの内容説明だけは補足されるかも。。。)

概要

Dockerfileの役割とLLVMビルドの大まかな作業フロー・動作内容を示す。

Dockerfileの役割
そもそもDockerを利用する上で必ずしもDockerfileを使用する必要はない。

Dockerを利用した「コンテナ起動 → コンテナ内で任意の開発環境構築などを行う」といった一連の流れが存在するが、これを省略化するためにDockerイメージをビルドする必要がある。

OS単位でのDockerイメージ(例:Ubuntu)はデフォルトで提供されており、そこから用途に合わせた独自環境を構築する場合にデフォルトのイメージから新たなDockerイメージをビルドする際に利用できるのがDockerfileである。

Dockerfileには環境構築の手数を記述しておくだけで自動的に実行され、Dockerイメージがビルドされる。

LLVMビルド

1. Ubuntu 18.04 或いは CentOS 8 に合わせたDockerfileを用意する。
2. Dockerfileから新たなDockerイメージのビルドを開始する。
   2.1. Ubuntu 18.04 或いは CentOS 8 のイメージを基にコンテナが起動する。
   2.2. コンテナ内でDockerfile内に記述されたコマンドが実行される。
   2.3. コマンドが実行されると、コンテナが終了し新たなイメージが生成される。
3. LLVMビルド環境が整備された新たなDockerイメージがビルドされる。

以降は、ビルドされたDockerイメージからコンテナを起動させ、LLVM環境で開発できたり、
更に変化を加えて、豊富な環境のDockerイメージをつくりだすことができる。

Dockerfileを利用してLLVMビルド

- Docker コマンドについて
- 手順

Docker コマンドについて

はじめに、Dockerfileを利用してLLVMをビルドする為にしっておきたい一般的なDockerコマンドを簡易的に示す。
※ 筆者の主観で選別したため、今回のビルド手順には含まれないものもある。

- 動いているコンテナの確認

```
docker ps
```

- 停止したものも含めたコンテナの確認

```
docker ps -a
```

- コンテナの削除

```
docker rm [コンテナID]
```

- 既存イメージ/コンテナ実行
  - コマンド(起動→新コンテナ分岐作成) :

```
docker run -it [イメージ名]
```

  - コマンド(上書起動)：

```
docker exec -it [コンテナID] /bin/bash
```

- コンテナからイメージ作成

```
docker commit [コンテナID] [イメージ]
```
(推奨)イメージは 「所有者/イメージ名」とすること。所有者は4文字以上。

- (Tips)コンテナ実行後、即削除

```
docker build --force-rm=true|false
```
https://qiita.com/Gin/items/dde3c3085f13f0a45c40#--rmtruefalse
※ デフォルトは`false`

- 現状のイメージの確認

```
docker images
```

- イメージの削除

```
docker rmi [イメージID]
```

手順

1. Dockerのinstall
2. Dockerfileのclone
3. Ubuntu 18.04イメージでのLLVMビルド
4. CentOS 8イメージでのLLVMビルド
5. ビルド後

1. Dockerのinstall

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo apt-key fingerprint 0EBFCD88
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
$ sudo docker run hello-world
$ docker -v
```

Dockerのバージョン情報が確認できればクリア。

2. Dockerfileのclone

(諸事情により、一時的に個人的なGithubアカウントを試作したところに置きました。)
cloneに必要な入力情報は以下の通りである。

```
$ git clone https://github.com/arcinoue/Docker.git
Username for 'https://github.com': inoue@architek.co.jp
Password for 'https://arcinoue@github.com': zxNC9Uk6davf9k5
```

3. Ubuntu 18.04イメージでのLLVMビルド

```
$ cd ~/Docker/llvm-build/ubuntu/Dockerfile
$ docker build -t [イメージ]
```

※ [イメージ]は任意の名前を入力すること。ただし、構文エラーに注意すること。

4. CentOS 8イメージでのLLVMビルド

```
$ cd ~/Docker/llvm-build/centos/Dockerfile
$ docker build -t [イメージ]
```

※ [イメージ]は任意の名前を入力すること。ただし、構文エラーに注意すること。

5. ビルド後

それぞれビルドしたイメージで、LLVMを使った開発環境がしたい場合は、以下のコマンドでコンテナを立ち上げる。

```
$ docker run -it [イメージ]
```

コンソール上で、シェルの起動を確認できればクリア。

備考

PC環境
OS : Linux (Ubuntu 18.04)
CPU : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
CPU CORES : 6
RAM : 32GB

※ Dockerfile実行におよそ1時間かかりました。

変更履歴

2020/03/25. inoue, 新規作成、いつか内容拡充・再構成予定。