Version : **2020.02**

Dernière mise-à-jour : 2020/11/17 13:18

DOF103 - Gérer les Images Docker

Contenu du Module

LAB #1 - Re-créer une image officielle docker

1.1 - Utilisation d'un Dockerfile

Bien que la compilation des images soient assuré par Docker Hub, il est tout à fait possible de compiler une image “officielle” à partir d'un Dockerfile :

```
root@debian9:~# mkdir mongodb
root@debian9:~# cd mongodb/
root@debian9:~/mongodb# touch Dockerfile docker-entrypoint.sh
```

Le Docker file contient les instructions nécessaires pour la contruction de l'image :

```
FROM ubuntu:bionic

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever
dependencies get added
RUN groupadd -r mongodb && useradd -r -g mongodb mongodb

RUN set -eux; \
    apt-get update; \
    apt-get install -y --no-install-recommends \
        ca-certificates \
        jq \
        numactl \
    ; \
    if ! command -v ps > /dev/null; then \
        apt-get install -y --no-install-recommends procps; \
    fi; \
    rm -rf /var/lib/apt/lists/*

# grab gosu for easy step-down from root (https://github.com/tianon/gosu/releases)
ENV GOSU_VERSION 1.11
# grab "js-yaml" for parsing mongod's YAML config files (https://github.com/nodeca/js-yaml/releases)
ENV JSYAML_VERSION 3.13.0

RUN set -ex; \
```

```
    \
    apt-get update; \
    apt-get install -y --no-install-recommends \
        wget \
    ; \
    if ! command -v gpg > /dev/null; then \
        apt-get install -y --no-install-recommends gnupg dirmngr; \
    fi; \
    rm -rf /var/lib/apt/lists/*; \
    \
    dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \
    wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch";
\
    wget -O /usr/local/bin/gosu.asc
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \
    export GNUPGHOME="$(mktemp -d)"; \
    gpg --batch --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4; \
    gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \
    command -v gpgconf && gpgconf --kill all || :; \
    rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc; \
    chmod +x /usr/local/bin/gosu; \
    gosu --version; \
    gosu nobody true; \
    \
    wget -O /js-yaml.js "https://github.com/nodeca/js-yaml/raw/${JSYAML_VERSION}/dist/js-yaml.js"; \
# TODO some sort of download verification here
    \
    apt-get purge -y --auto-remove wget

RUN mkdir /docker-entrypoint-initdb.d

ENV GPG_KEYS E162F504A20CDF15827F718D4B7C549A058F8B6B
RUN set -ex; \
    export GNUPGHOME="$(mktemp -d)"; \
```

```
    for key in $GPG_KEYS; do \
        gpg --batch --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \
    done; \
    gpg --batch --export $GPG_KEYS > /etc/apt/trusted.gpg.d/mongodb.gpg; \
    command -v gpgconf && gpgconf --kill all || :; \
    rm -r "$GNUPGHOME"; \
    apt-key list

# Allow build-time overrides (eg. to build image with MongoDB Enterprise version)
# Options for MONGO_PACKAGE: mongodb-org OR mongodb-enterprise
# Options for MONGO_REPO: repo.mongodb.org OR repo.mongodb.com
# Example: docker build --build-arg MONGO_PACKAGE=mongodb-enterprise --build-arg MONGO_REPO=repo.mongodb.com .
ARG MONGO_PACKAGE=mongodb-org-unstable
ARG MONGO_REPO=repo.mongodb.org
ENV MONGO_PACKAGE=${MONGO_PACKAGE} MONGO_REPO=${MONGO_REPO}

ENV MONGO_MAJOR 4.1
ENV MONGO_VERSION 4.1.9
# bashbrew-architectures:amd64 arm64v8 s390x
RUN echo "deb http://$MONGO_REPO/apt/ubuntu bionic/${MONGO_PACKAGE%-unstable}/$MONGO_MAJOR multiverse" | tee
"/etc/apt/sources.list.d/${MONGO_PACKAGE%-unstable}.list"

RUN set -x \
    && apt-get update \
    && apt-get install -y \
        ${MONGO_PACKAGE}=$MONGO_VERSION \
        ${MONGO_PACKAGE}-server=$MONGO_VERSION \
        ${MONGO_PACKAGE}-shell=$MONGO_VERSION \
        ${MONGO_PACKAGE}-mongos=$MONGO_VERSION \
        ${MONGO_PACKAGE}-tools=$MONGO_VERSION \
    && rm -rf /var/lib/apt/lists/* \
    && rm -rf /var/lib/mongodb \
    && mv /etc/mongod.conf /etc/mongod.conf.orig
```

```
RUN mkdir -p /data/db /data/configdb \
    && chown -R mongodb:mongodb /data/db /data/configdb
VOLUME /data/db /data/configdb

COPY docker-entrypoint.sh /usr/local/bin/
ENTRYPOINT ["docker-entrypoint.sh"]

EXPOSE 27017
CMD ["mongod"]
```

Le fichier docker-entrypoint.sh sert à lancer le serveur mongodb dans le conteneur :

```
#!/bin/bash
set -Eeuo pipefail

if [ "${1:0:1}" = '-' ]; then
    set -- mongod "$@"
fi

originalArgOne="$1"

# allow the container to be started with `--user`
# all mongo* commands should be dropped to the correct user
if [[ "$originalArgOne" == mongo* ]] && [ "$(id -u)" = '0' ]; then
    if [ "$originalArgOne" = 'mongod' ]; then
        find /data/configdb /data/db \! -user mongodb -exec chown mongodb '{}' +
    fi

    # make sure we can write to stdout and stderr as "mongodb"
    # (for our "initdb" code later; see "--logpath" below)
    chown --dereference mongodb "/proc/$$/fd/1" "/proc/$$/fd/2" || :
    # ignore errors thanks to https://github.com/docker-library/mongo/issues/149
```

```
    exec gosu mongodb "$BASH_SOURCE" "$@"
fi

# you should use numactl to start your mongod instances, including the config servers, mongos instances, and any
clients.
# https://docs.mongodb.com/manual/administration/production-notes/#configuring-numa-on-linux
if [[ "$originalArgOne" == mongo* ]]; then
    numa='numactl --interleave=all'
    if $numa true &> /dev/null; then
        set -- $numa "$@"
    fi
fi

# usage: file_env VAR [DEFAULT]
#    ie: file_env 'XYZ_DB_PASSWORD' 'example'
# (will allow for "$XYZ_DB_PASSWORD_FILE" to fill in the value of
#  "$XYZ_DB_PASSWORD" from a file, especially for Docker's secrets feature)
file_env() {
    local var="$1"
    local fileVar="${var}_FILE"
    local def="${2:-}"
    if [ "${!var:-}" ] && [ "${!fileVar:-}" ]; then
        echo >&2 "error: both $var and $fileVar are set (but are exclusive)"
        exit 1
    fi
    local val="$def"
    if [ "${!var:-}" ]; then
        val="${!var}"
    elif [ "${!fileVar:-}" ]; then
        val="$(< "${!fileVar}")"
    fi
    export "$var"="$val"
    unset "$fileVar"
}
```

```
# see https://github.com/docker-library/mongo/issues/147 (mongod is picky about duplicated arguments)
_mongod_hack_have_arg() {
    local checkArg="$1"; shift
    local arg
    for arg; do
        case "$arg" in
            "$checkArg"|"$checkArg"=*)
                return 0
                ;;
        esac
    done
    return 1
}
# _mongod_hack_get_arg_val '--some-arg' "$@"
_mongod_hack_get_arg_val() {
    local checkArg="$1"; shift
    while [ "$#" -gt 0 ]; do
        local arg="$1"; shift
        case "$arg" in
            "$checkArg")
                echo "$1"
                return 0
                ;;
            "$checkArg"=*)
                echo "${arg#$checkArg=}"
                return 0
                ;;
        esac
    done
    return 1
}
declare -a mongodHackedArgs
# _mongod_hack_ensure_arg '--some-arg' "$@"
# set -- "${mongodHackedArgs[@]}"
```

```
_mongod_hack_ensure_arg() {
    local ensureArg="$1"; shift
    mongodHackedArgs=( "$@" )
    if ! _mongod_hack_have_arg "$ensureArg" "$@"; then
        mongodHackedArgs+=( "$ensureArg" )
    fi
}
# _mongod_hack_ensure_no_arg '--some-unwanted-arg' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_no_arg() {
    local ensureNoArg="$1"; shift
    mongodHackedArgs=()
    while [ "$#" -gt 0 ]; do
        local arg="$1"; shift
        if [ "$arg" = "$ensureNoArg" ]; then
            continue
        fi
        mongodHackedArgs+=( "$arg" )
    done
}
# _mongod_hack_ensure_no_arg '--some-unwanted-arg' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_no_arg_val() {
    local ensureNoArg="$1"; shift
    mongodHackedArgs=()
    while [ "$#" -gt 0 ]; do
        local arg="$1"; shift
        case "$arg" in
            "$ensureNoArg")
                shift # also skip the value
                continue
                ;;
            "$ensureNoArg"=*)
                # value is already included
```

```
                continue
                ;;
        esac
        mongodHackedArgs+=( "$arg" )
    done
}
# _mongod_hack_ensure_arg_val '--some-arg' 'some-val' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_arg_val() {
    local ensureArg="$1"; shift
    local ensureVal="$1"; shift
    _mongod_hack_ensure_no_arg_val "$ensureArg" "$@"
    mongodHackedArgs+=( "$ensureArg" "$ensureVal" )
}

# _js_escape 'some "string" value'
_js_escape() {
    jq --null-input --arg 'str' "$1" '$str'
}

jsonConfigFile="${TMPDIR:-/tmp}/docker-entrypoint-config.json"
tempConfigFile="${TMPDIR:-/tmp}/docker-entrypoint-temp-config.json"
_parse_config() {
    if [ -s "$tempConfigFile" ]; then
        return 0
    fi

    local configPath
    if configPath="$(_mongod_hack_get_arg_val --config "$@")"; then
        # if --config is specified, parse it into a JSON file so we can remove a few problematic keys (especially
SSL-related keys)
        # see https://docs.mongodb.com/manual/reference/configuration-options/
        mongo --norc --nodb --quiet --eval "load('/js-yaml.js'); printjson(jsyaml.load(cat($(_js_escape
"$configPath"))))" > "$jsonConfigFile"
```

```
        jq 'del(.systemLog, .processManagement, .net, .security)' "$jsonConfigFile" > "$tempConfigFile"
        return 0
    fi

    return 1
}
dbPath=
_dbPath() {
    if [ -n "$dbPath" ]; then
        echo "$dbPath"
        return
    fi

    if ! dbPath="$(_mongod_hack_get_arg_val --dbpath "$@")"; then
        if _parse_config "$@"; then
            dbPath="$(jq -r '.storage.dbPath // empty' "$jsonConfigFile")"
        fi
    fi

    if [ -z "$dbPath" ]; then
        if _mongod_hack_have_arg --configsvr "$@" || {
            _parse_config "$@" \
            && clusterRole="$(jq -r '.sharding.clusterRole // empty' "$jsonConfigFile")" \
            && [ "$clusterRole" = 'configsvr']
        }; then
            # if running as config server, then the default dbpath is /data/configdb
            # https://docs.mongodb.com/manual/reference/program/mongod/#cmdoption-mongod-configsvr
            dbPath=/data/configdb
        fi
    fi

    : "${dbPath:=/data/db}"

    echo "$dbPath"
```

```
}

if [ "$originalArgOne" = 'mongod' ]; then
    file_env 'MONGO_INITDB_ROOT_USERNAME'
    file_env 'MONGO_INITDB_ROOT_PASSWORD'
    # pre-check a few factors to see if it's even worth bothering with initdb
    shouldPerformInitdb=
    if [ "$MONGO_INITDB_ROOT_USERNAME" ] && [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
        # if we have a username/password, let's set "--auth"
        _mongod_hack_ensure_arg '--auth' "$@"
        set -- "${mongodHackedArgs[@]}"
        shouldPerformInitdb='true'
    elif [ "$MONGO_INITDB_ROOT_USERNAME" ] || [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
        cat >&2 <<-'EOF'
            error: missing 'MONGO_INITDB_ROOT_USERNAME' or 'MONGO_INITDB_ROOT_PASSWORD'
                   both must be specified for a user to be created
        EOF
        exit 1
    fi

    if [ -z "$shouldPerformInitdb" ]; then
        # if we've got any /docker-entrypoint-initdb.d/* files to parse later, we should initdb
        for f in /docker-entrypoint-initdb.d/*; do
            case "$f" in
                *.sh|*.js) # this should match the set of files we check for below
                    shouldPerformInitdb="$f"
                    break
                    ;;
            esac
        done
    fi

    # check for a few known paths (to determine whether we've already initialized and should thus skip our initdb
scripts)
```

```
    if [ -n "$shouldPerformInitdb" ]; then
        dbPath="$(_dbPath "$@")"
        for path in \
            "$dbPath/WiredTiger" \
            "$dbPath/journal" \
            "$dbPath/local.0" \
            "$dbPath/storage.bson" \
        ; do
            if [ -e "$path" ]; then
                shouldPerformInitdb=
                break
            fi
        done
    fi

    if [ -n "$shouldPerformInitdb" ]; then
        mongodHackedArgs=( "$@" )
        if _parse_config "$@"; then
            _mongod_hack_ensure_arg_val --config "$tempConfigFile" "${mongodHackedArgs[@]}"
        fi
        _mongod_hack_ensure_arg_val --bind_ip 127.0.0.1 "${mongodHackedArgs[@]}"
        _mongod_hack_ensure_arg_val --port 27017 "${mongodHackedArgs[@]}"
        _mongod_hack_ensure_no_arg --bind_ip_all "${mongodHackedArgs[@]}"

        # remove "--auth" and "--replSet" for our initial startup (see
https://docs.mongodb.com/manual/tutorial/enable-authentication/#start-mongodb-without-access-control)
        # https://github.com/docker-library/mongo/issues/211
        _mongod_hack_ensure_no_arg --auth "${mongodHackedArgs[@]}"
        if [ "$MONGO_INITDB_ROOT_USERNAME" ] && [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
            _mongod_hack_ensure_no_arg_val --replSet "${mongodHackedArgs[@]}"
        fi

        sslMode="$(_mongod_hack_have_arg '--sslPEMKeyFile' "$@" && echo 'allowSSL' || echo 'disabled')" #
"BadValue: need sslPEMKeyFile when SSL is enabled" vs "BadValue: need to enable SSL via the sslMode flag when
```

```
using SSL configuration parameters"
        _mongod_hack_ensure_arg_val --sslMode "$sslMode" "${mongodHackedArgs[@]}"

        if stat "/proc/$$/fd/1" > /dev/null && [ -w "/proc/$$/fd/1" ]; then
            #
https://github.com/mongodb/mongo/blob/38c0eb538d0fd390c6cb9ce9ae9894153f6e8ef5/src/mongo/db/initialize_server_glo
bal_state.cpp#L237-L251
            # https://github.com/docker-library/mongo/issues/164#issuecomment-293965668
            _mongod_hack_ensure_arg_val --logpath "/proc/$$/fd/1" "${mongodHackedArgs[@]}"
        else
            initdbLogPath="$(_dbPath "$@")/docker-initdb.log"
            echo >&2 "warning: initdb logs cannot write to '/proc/$$/fd/1', so they are in '$initdbLogPath'
instead"
            _mongod_hack_ensure_arg_val --logpath "$initdbLogPath" "${mongodHackedArgs[@]}"
        fi
        _mongod_hack_ensure_arg --logappend "${mongodHackedArgs[@]}"

        pidfile="${TMPDIR:-/tmp}/docker-entrypoint-temp-mongod.pid"
        rm -f "$pidfile"
        _mongod_hack_ensure_arg_val --pidfilepath "$pidfile" "${mongodHackedArgs[@]}"

        "${mongodHackedArgs[@]}" --fork

        mongo=( mongo --host 127.0.0.1 --port 27017 --quiet )

        # check to see that our "mongod" actually did start up (catches "--help", "--version", MongoDB 3.2 being
silly, slow prealloc, etc)
        # https://jira.mongodb.org/browse/SERVER-16292
        tries=30
        while true; do
            if ! { [ -s "$pidfile" ] && ps "$(< "$pidfile")" &> /dev/null; }; then
                # bail ASAP if "mongod" isn't even running
                echo >&2
                echo >&2 "error: $originalArgOne does not appear to have stayed running -- perhaps it had an
```

```
error?"
                echo >&2
                exit 1
            fi
            if "${mongo[@]}" 'admin' --eval 'quit(0)' &> /dev/null; then
                # success!
                break
            fi
            (( tries-- ))
            if [ "$tries" -le 0 ]; then
                echo >&2
                echo >&2 "error: $originalArgOne does not appear to have accepted connections quickly enough --
perhaps it had an error?"
                echo >&2
                exit 1
            fi
            sleep 1
        done

        if [ "$MONGO_INITDB_ROOT_USERNAME" ] && [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
            rootAuthDatabase='admin'

            "${mongo[@]}" "$rootAuthDatabase" <<-EOJS
                db.createUser({
                    user: $(_js_escape "$MONGO_INITDB_ROOT_USERNAME"),
                    pwd: $(_js_escape "$MONGO_INITDB_ROOT_PASSWORD"),
                    roles: [ { role: 'root', db: $(_js_escape "$rootAuthDatabase") } ]
                })
            EOJS
        fi

        export MONGO_INITDB_DATABASE="${MONGO_INITDB_DATABASE:-test}"

        echo
```

```
        for f in /docker-entrypoint-initdb.d/*; do
            case "$f" in
                *.sh) echo "$0: running $f"; . "$f" ;;
                *.js) echo "$0: running $f"; "${mongo[@]}" "$MONGO_INITDB_DATABASE" "$f"; echo ;;
                *)    echo "$0: ignoring $f" ;;
            esac
            echo
        done

        "${mongodHackedArgs[@]}" --shutdown
        rm -f "$pidfile"

        echo
        echo 'MongoDB init process complete; ready for start up.'
        echo
    fi

    # MongoDB 3.6+ defaults to localhost-only binding
    if mongod --help 2>&1 | grep -q -- --bind_ip_all; then # TODO remove this conditional when 3.4 is no longer
supported
        haveBindIp=
        if _mongod_hack_have_arg --bind_ip "$@" || _mongod_hack_have_arg --bind_ip_all "$@"; then
            haveBindIp=1
        elif _parse_config "$@" && jq --exit-status '.net.bindIp // .net.bindIpAll' "$jsonConfigFile" >
/dev/null; then
            haveBindIp=1
        fi
        if [ -z "$haveBindIp" ]; then
            # so if no "--bind_ip" is specified, let's add "--bind_ip_all"
            set -- "$@" --bind_ip_all
        fi
    fi

    unset "${!MONGO_INITDB_@}"
```

```
fi

rm -f "$jsonConfigFile" "$tempConfigFile"

exec "$@"
```

Examinons chaque commande dans le Dockerfile :

1.2 - FROM

```
FROM ubuntu:bionic
```

Cette ligne définit l'image à partir de laquelle sera construite notre image. Quand l'image n'est construite à partir d'une autre image, la valeur de **FROM** est **scratch**.

1.3 - RUN

```
...

RUN groupadd -r mongodb && useradd -r -g mongodb mongodb

RUN set -eux; \
    apt-get update; \
    apt-get install -y --no-install-recommends \
        ca-certificates \
        jq \
        numactl \
    ; \
```

```
    if ! command -v ps > /dev/null; then \
        apt-get install -y --no-install-recommends procps; \
    fi; \
    rm -rf /var/lib/apt/lists/*
...
RUN set -ex; \
    \
    apt-get update; \
    apt-get install -y --no-install-recommends \
        wget \
    ; \
    if ! command -v gpg > /dev/null; then \
        apt-get install -y --no-install-recommends gnupg dirmngr; \
    fi; \
    rm -rf /var/lib/apt/lists/*; \
    \
    dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \
    wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch";
\
    wget -O /usr/local/bin/gosu.asc
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \
    export GNUPGHOME="$(mktemp -d)"; \
    gpg --batch --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4; \
    gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \
    command -v gpgconf && gpgconf --kill all || :; \
    rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc; \
    chmod +x /usr/local/bin/gosu; \
    gosu --version; \
    gosu nobody true; \
    \
    wget -O /js-yaml.js "https://github.com/nodeca/js-yaml/raw/${JSYAML_VERSION}/dist/js-yaml.js"; \
# TODO some sort of download verification here
    \
    apt-get purge -y --auto-remove wget
```

```
RUN mkdir /docker-entrypoint-initdb.d
...

RUN set -ex; \
    export GNUPGHOME="$(mktemp -d)"; \
    for key in $GPG_KEYS; do \
        gpg --batch --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \
    done; \
    gpg --batch --export $GPG_KEYS > /etc/apt/trusted.gpg.d/mongodb.gpg; \
    command -v gpgconf && gpgconf --kill all || :; \
    rm -r "$GNUPGHOME"; \
    apt-key list
...
RUN set -x \
    && apt-get update \
    && apt-get install -y \
        ${MONGO_PACKAGE}=$MONGO_VERSION \
        ${MONGO_PACKAGE}-server=$MONGO_VERSION \
        ${MONGO_PACKAGE}-shell=$MONGO_VERSION \
        ${MONGO_PACKAGE}-mongos=$MONGO_VERSION \
        ${MONGO_PACKAGE}-tools=$MONGO_VERSION \
    && rm -rf /var/lib/apt/lists/* \
    && rm -rf /var/lib/mongodb \
    && mv /etc/mongod.conf /etc/mongod.conf.orig

RUN mkdir -p /data/db /data/configdb \
    && chown -R mongodb:mongodb /data/db /data/configdb
...
```

Cette commande lance un processus dans la construction de l'image. Dans les cas ci-dessus, chaque chaîne correspond à la commande passée au shell **/bin/sh**.

Il existe un autre syntaxe de la commande RUN appelé le format exec, à savoir :

```
RUN ["/bin/bash", "-c", "commande"]
```

Important : La commande RUN est utilisée pour exécuter une commande passée en argument lors de la compilation de l'image seulement. Cette commande ne doit pas donc être utilisée pour exécuter une commande lors du lancement du conteneur. La commande utilisée pour accomplir ce dernier est ENTRYPOINT.

1.4 - ENV

Cette commande permet de fixer la valeur d'une variable d'environnement disponible dans la suite du Dockerfile :

```
...
ENV GOSU_VERSION 1.11
# grab "js-yaml" for parsing mongod's YAML config files (https://github.com/nodeca/js-yaml/releases)
ENV JSYAML_VERSION 3.13.0
...

ENV GPG_KEYS E162F504A20CDF15827F718D4B7C549A058F8B6B
...

ENV MONGO_PACKAGE=${MONGO_PACKAGE} MONGO_REPO=${MONGO_REPO}

ENV MONGO_MAJOR 4.1
ENV MONGO_VERSION 4.1.95
...
```

et dans les conteneurs générés à partir de l'image construite.

1.5 - VOLUME

```
...
VOLUME /data/db /data/configdb
...
```

Cette commande expose les répertoires passés en argument afin qu'ils puissent être mappés vers des répertoires sur la machine hôte ou ailleurs, tel que nous avons vu avec l'exemple nginx.

1.6 - COPY

```
...
COPY docker-entrypoint.sh /usr/local/bin/
...
```

Cette commande permet de récupérer les fichiers dans le contexte et de les copier dans l'image.

Attention : tous les fichiers dans le contexte sont inclus dans l'image finale, même ceux qui sont inutiles.

Il est possible d'exclure des fichiers présents dans le contexte en les mettant dans un fichier appelé **.dockerignore** placé dans le contexte.

Important - Il existe une autre commande similaire à COPY : ADD. ADD est une commande qui n'est plus recommendé sauf dans le cas de cas spécifiques. Notez que dans le cas de l'utilisation de la commande ADD, si le fichier source est une archive de type TAR, son contenu sera désarchivé et copier vers la destination tandis que si le fichier source est référencé par un URL, le contenu sera téléchargé puis déposé dans la destination.

1.7 - ENTRYPOINT

```
...
ENTRYPOINT ["docker-entrypoint.sh"]
...
```

Cette commande stipule la commande qui sera exécutée lors du démarrage du conteneur.

Deux cas de figure se présentent :

- ENTRYPOINT suivi d'une chaîne - un shell est démarré pour exécuter la chaîne,
- ENTRYPOINT suivi d'une table JSON (comme ci-dessus) au format ENTRYPOINT [“commande à exécuter”, “paramètres de la commande”].

Dans le fichier **docker-entrypoint.sh** :

```
...
originalArgOne="$1"

# allow the container to be started with `--user`
# all mongo* commands should be dropped to the correct user
if [[ "$originalArgOne" == mongo* ]] && [ "$(id -u)" = '0' ]; then
    if [ "$originalArgOne" = 'mongod' ]; then
        find /data/configdb /data/db \! -user mongodb -exec chown mongodb '{}' +
    fi

    # make sure we can write to stdout and stderr as "mongodb"
    # (for our "initdb" code later; see "--logpath" below)
    chown --dereference mongodb "/proc/$$/fd/1" "/proc/$$/fd/2" || :
    # ignore errors thanks to https://github.com/docker-library/mongo/issues/149

    exec gosu mongodb "$BASH_SOURCE" "$@"
```

```
fi

# you should use numactl to start your mongod instances, including the config servers, mongos instances, and any
clients.
# https://docs.mongodb.com/manual/administration/production-notes/#configuring-numa-on-linux
if [[ "$originalArgOne" == mongo* ]]; then
    numa='numactl --interleave=all'
    if $numa true &> /dev/null; then
        set -- $numa "$@"
    fi
fi
...
exec "$@"
```

si la valeur du paramètre passé à entrypoint.sh est **mongod**, le script affecte l'utilisateur mongodb aux répertoires /data/configdb et /data/db puis lance mongo sous l'utilisateur mongodb avec des droits réduits (gosu).

Ce fichier finit par “$@” qui indique que si aucune condition n'ait été remplie, la commande est exécutée avec la valeur passée en argument.

Important - Notez que la compilation d'une image se fait à l'intérieur d'un **contexte**. Le **contexte** est le répertoire de build. Dernièrement, notez qu'il peut y avoir plusieurs ENTRYPOINT dans le fichier Dockerfile mais uniquement le dernier est pris en compte.

1.8 - EXPOSE

```
...
EXPOSE 27017
...
```

Cette commande permet d'exposer un port à l'extérieur du conteneur.

1.9 - CMD

```
...
CMD ["mongod"]
...
```

Ceci représente la valeur du paramètre par défaut si aucun paramètre n'est spécifié à la fin de la commande docker run.

1.10 - Autres Commandes

Le Dockerfile peut aussi contenir les commandes suivantes :

- **WORKDIR**,
  - Cette commande fixe le répertoire de travil lors de la compilation d'une image. Elle peut apparaître plusieurs fois dans le Dockerfile permettant ainsi l'évolution du répertoire de travail,
- **LABEL**,
  - Cette commande permet de définir des couples clef/valeur à inclure dans les méta-données décrivant l'image lors de sa distribution, par exemple, la **version**, la **description** ou un **readme**.

Lancez maintenant la compilation de l'image :

```
root@debian9:~/mongodb# docker build .
```

Consultez la liste de images :

```
root@debian9:~/mongodb# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
<none>              <none>              3bf216d921d6        About a minute ago   96.2MB
i2tch/mongodb       latest              eca7835d4fe6        11 minutes ago       1.03GB
nginx               latest              2bcb04bdb83f        13 days ago          109MB
centos              latest              9f38484d220f        3 weeks ago          202MB
ubuntu              bionic              94e814e2efa8        4 weeks ago          88.9MB
ubuntu              latest              94e814e2efa8        4 weeks ago          88.9MB
hello-world         latest              fce289e99eb9        3 months ago         1.84kB
```

Notez que l'image n'a ni REPOSITORY, ni TAG. Créez donc un TAG :

```
root@debian9:~/mongodb# docker tag 3bf2 i2tch/mongodb1
root@debian9:~/mongodb# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
i2tch/mongodb1      latest              3bf216d921d6        2 minutes ago        96.2MB
i2tch/mongodb       latest              eca7835d4fe6        11 minutes ago       1.03GB
nginx               latest              2bcb04bdb83f        13 days ago          109MB
centos              latest              9f38484d220f        3 weeks ago          202MB
ubuntu              bionic              94e814e2efa8        4 weeks ago          88.9MB
ubuntu              latest              94e814e2efa8        4 weeks ago          88.9MB
hello-world         latest              fce289e99eb9        3 months ago         1.84kB
```

Démarrez un conteneur à partir de l'image i2tch/mongodb1 :

```
root@debian9:~/mongodb# docker run -d --name mongo1 i2tch/mongodb1
bdb4bc0f81de8b5821f20d8609b9640abaaae7b4a7577c42b78d4bd34617d211
docker: Error response from daemon: oci runtime error: container_linux.go:262: starting container process caused
"exec: \"docker-entrypoint.sh\": executable file not found in $PATH".
root@debian9:~/mongodb# ls -l
total 16
```

```
-rw-r--r-- 1 root root 10971 avril  9 13:56 docker-entrypoint.sh
-rw-r--r-- 1 root root  3542 avril  9 13:55 Dockerfile
```

Important - Notez que le fichier docker-entrypoint.sh n'était pas exécutable !

Recompilez donc l'image :

```
root@debian9:~/mongodb# chmod +x docker-entrypoint.sh
root@debian9:~/mongodb# docker build .
Sending build context to Docker daemon   16.9kB
Step 1/22 : FROM ubuntu:bionic
 ---> 94e814e2efa8
Step 2/22 : RUN groupadd -r mongodb && useradd -r -g mongodb mongodb
 ---> Using cache
 ---> f40ac453fa97
Step 3/22 : RUN set -eux;   apt-get update;     apt-get install -y --no-install-recommends         ca-
certificates         jq         numactl ;     if ! command -v ps > /dev/null; then         apt-get install -y --
no-install-recommends procps;     fi;     rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> adc57da1b19f
Step 4/22 : ENV GOSU_VERSION 1.11
 ---> Using cache
 ---> 038e7de870b7
Step 5/22 : ENV JSYAML_VERSION 3.13.0
 ---> Using cache
 ---> 3bf216d921d6
...
Removing intermediate container a98ae692fe1f
 ---> 04c2e98927c3
Step 17/22 : RUN mkdir -p /data/db /data/configdb   && chown -R mongodb:mongodb /data/db /data/configdb
```

```
 ---> Running in d0f5bee34571
Removing intermediate container d0f5bee34571
 ---> d5b95e9e63e1
Step 18/22 : VOLUME /data/db /data/configdb
 ---> Running in c7626528a9b9
Removing intermediate container c7626528a9b9
 ---> 4250613adf6a
Step 19/22 : COPY docker-entrypoint.sh /usr/local/bin/
 ---> eedfd53da0f8
Step 20/22 : ENTRYPOINT ["docker-entrypoint.sh"]
 ---> Running in eff53d0213d1
Removing intermediate container eff53d0213d1
 ---> 716abf2faa87
Step 21/22 : EXPOSE 27017
 ---> Running in 5139fcf19d7f
Removing intermediate container 5139fcf19d7f
 ---> fc5896e08fd6
Step 22/22 : CMD ["mongod"]
 ---> Running in 458d6f15cdf2
Removing intermediate container 458d6f15cdf2
 ---> 12e00099ca8d
Successfully built 12e00099ca8d
root@debian9:~/mongodb#
```

Important - Notez ici les lignes **Using cache**. Il est cependant possible de ne pas utiliser le cache en stipulant **–no-cache**. Notez aussi l'utilisation de conteneurs temporaires par étape nouvelle avec un commit vers une image et une suppression dudit conteneur. Dernièrement, notez que la compilation d'une image se fait à l'intérieur d'un **contexte**. Le **contexte** est le répertoire de build. **Attention** : tous les fichiers dans le contexte sont inclus dans l'image finale, même ceux qui sont inutiles.

Consultez la liste des images de nouveau et renommez votre dernière image :

```
root@debian9:~/mongodb# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              12e00099ca8d        42 seconds ago      377MB
i2tch/mongodb1      latest              3bf216d921d6        10 minutes ago      96.2MB
i2tch/mongodb       latest              eca7835d4fe6        19 minutes ago      1.03GB
nginx               latest              2bcb04bdb83f        13 days ago         109MB
centos              latest              9f38484d220f        3 weeks ago         202MB
ubuntu              bionic              94e814e2efa8        4 weeks ago         88.9MB
ubuntu              latest              94e814e2efa8        4 weeks ago         88.9MB
hello-world         latest              fce289e99eb9        3 months ago        1.84kB
root@debian9:~/mongodb# docker tag 12e0 i2tch/mongodb2
root@debian9:~/mongodb# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
i2tch/mongodb2      latest              12e00099ca8d        About a minute ago   377MB
i2tch/mongodb1      latest              3bf216d921d6        11 minutes ago       96.2MB
i2tch/mongodb       latest              eca7835d4fe6        20 minutes ago       1.03GB
nginx               latest              2bcb04bdb83f        13 days ago          109MB
centos              latest              9f38484d220f        3 weeks ago          202MB
ubuntu              bionic              94e814e2efa8        4 weeks ago          88.9MB
ubuntu              latest              94e814e2efa8        4 weeks ago          88.9MB
hello-world         latest              fce289e99eb9        3 months ago         1.84kB
```

Lancez un conteneur à partir de la dernière image :

```
root@debian9:~/mongodb# docker run -d --name mongo2 i2tch/mongodb2
e91a055283f4d67cbd91d11bb3faa6f67925893cb18f9cc25023e72e0f7ed85a
```

Utilisez la commande **docker ps** pour visualiser si le processus mongodb est bien démarré :

```
root@debian9:~/mongodb# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
```

```
NAMES
e91a055283f4        i2tch/mongodb2      "docker-entrypoint.s…"   28 seconds ago      Up 27 seconds
27017/tcp           mongo2
d2ddb4f8ca8a        i2tch/mongodb       "bash"                   21 minutes ago      Up 19 minutes
mongo
c080793965de        nginx               "nginx -g 'daemon of…"   About an hour ago   Up About an hour
0.0.0.0:81->80/tcp   suspicious_sanderson
```

Connectez-vous à mongodb à partir de votre machine hôte :

```
root@debian9:~/mongodb# docker inspect mongo2 | grep IP
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.4",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.4",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
root@debian9:~/mongodb#
root@debian9:~/mongodb# mongo --host 172.17.0.4
MongoDB shell version v4.0.8
connecting to: mongodb://172.17.0.4:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("3feff8c0-5460-473b-b036-4aee64a314f7") }
MongoDB server version: 4.1.9
```

```
WARNING: shell and server versions do not match
Server has startup warnings:
2019-04-09T17:50:12.635+0000 I STORAGE  [initandlisten]
2019-04-09T17:50:12.636+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly
recommended with the WiredTiger storage engine
2019-04-09T17:50:12.636+0000 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2019-04-09T17:50:13.458+0000 I CONTROL  [initandlisten]
2019-04-09T17:50:13.459+0000 I CONTROL  [initandlisten] ** NOTE: This is a development version (4.1.9) of
MongoDB.
2019-04-09T17:50:13.459+0000 I CONTROL  [initandlisten] **       Not recommended for production.
2019-04-09T17:50:13.459+0000 I CONTROL  [initandlisten]
2019-04-09T17:50:13.459+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the
database.
2019-04-09T17:50:13.459+0000 I CONTROL  [initandlisten] **          Read and write access to data and
configuration is unrestricted.
2019-04-09T17:50:13.460+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> exit
bye
root@debian9:~/mongodb#
```

Notez que lors de la compilation de l'image finale, une image a été créée lors de chaque instruction dans le fichier Dockerfile sauf en cas d'utilisation d'une image en cache :

```
root@debian9:~/mongodb# docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
i2tch/mongodb2      latest              12e00099ca8d        5 minutes ago       377MB
<none>              <none>              d5b95e9e63e1        5 minutes ago       377MB
<none>              <none>              4250613adf6a        5 minutes ago       377MB
<none>              <none>              eedfd53da0f8        5 minutes ago       377MB
<none>              <none>              04c2e98927c3        5 minutes ago       377MB
<none>              <none>              c6eae79e3d22        7 minutes ago       110MB
<none>              <none>              c205179d538c        7 minutes ago       110MB
<none>              <none>              b70835bebe35        7 minutes ago       110MB
<none>              <none>              5b2827910929        7 minutes ago       110MB
<none>              <none>              5b1f6df94d98        7 minutes ago       110MB
<none>              <none>              a950a5d04b68        7 minutes ago       110MB
<none>              <none>              c183cfecc5f0        7 minutes ago       110MB
<none>              <none>              aadb5806f1b8        8 minutes ago       110MB
<none>              <none>              8d538d38407e        8 minutes ago       110MB
<none>              <none>              32d59bf23987        8 minutes ago       110MB
i2tch/mongodb1      latest              3bf216d921d6        15 minutes ago      96.2MB
<none>              <none>              038e7de870b7        15 minutes ago      96.2MB
<none>              <none>              adc57da1b19f        15 minutes ago      96.2MB
<none>              <none>              f40ac453fa97        15 minutes ago      89.3MB
i2tch/mongodb       latest              eca7835d4fe6        24 minutes ago      1.03GB
<none>              <none>              620057baa411        27 minutes ago      816MB
<none>              <none>              67afc80e1424        33 minutes ago      816MB
nginx               latest              2bcb04bdb83f        13 days ago         109MB
centos              latest              9f38484d220f        3 weeks ago         202MB
ubuntu              bionic              94e814e2efa8        4 weeks ago         88.9MB
ubuntu              latest              94e814e2efa8        4 weeks ago         88.9MB
hello-world         latest              fce289e99eb9        3 months ago        1.84kB
```

LAB #2 - Créer un Dockerfile

2.1 - Création et test du script

Créez un répertoire nommé myDocker :

```
root@debian9:~/mongodb# mkdir ~/myDocker
root@debian9:~/mongodb# cd ~/myDocker
root@debian9:~/myDocker#
```

Créez le fichier myEntrypoint.sh :

```
root@debian9:~/myDocker# vi myEntrypoint.sh
root@debian9:~/myDocker# cat myEntrypoint.sh
#!/bin/bash
if [ -z "$myVariable" ]; then
    echo "La variable myVariable doit être renseignée"
    return 1
fi

while true;
do
    echo $1 \($(date +%H:%M:%S)\);
    sleep "$myVariable";
done
```

Testez ce script :

```
root@debian9:~/myDocker# myVariable=3 . ./myEntrypoint.sh salut
salut (20:04:39)
salut (20:04:42)
salut (20:04:45)
salut (20:04:48)
salut (20:04:51)
^C
root@debian9:~/myDocker#
```

Rendez ce script exécutable :

```
root@debian9:~/myDocker# chmod u+x myEntrypoint.sh
```

Créez maintenant le fichier **Dockerfile** dans le répertoire **~/myDocker** :

```
root@debian9:~/myDocker# vi Dockerfile
root@debian9:~/myDocker# cat Dockerfile
FROM centos:latest
MAINTAINER i2tch "infos@i2tch.eu"
COPY myEntrypoint.sh /entrypoint.sh
ENV myVariable 3
ENTRYPOINT ["/entrypoint.sh"]
CMD ["mycommand"]
```

Générez maintenant l'image :

```
root@debian9:~/myDocker# docker build -t i2tch/mydocker .
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM centos:latest
```

```
 ---> 9f38484d220f
Step 2/6 : MAINTAINER i2tch "infos@i2tch.eu"
 ---> Running in 02c700ed04da
Removing intermediate container 02c700ed04da
 ---> 4274107d52e2
Step 3/6 : COPY myEntrypoint.sh /entrypoint.sh
 ---> 7a3923372768
Step 4/6 : ENV myVariable 3
 ---> Running in 3288bf6291ad
Removing intermediate container 3288bf6291ad
 ---> 3edb630c1511
Step 5/6 : ENTRYPOINT ["/entrypoint.sh"]
 ---> Running in 8dcba2c41520
Removing intermediate container 8dcba2c41520
 ---> 11962052539c
Step 6/6 : CMD ["mycommand"]
 ---> Running in f891fbcfaad0
Removing intermediate container f891fbcfaad0
 ---> 7925ba23abb2
Successfully built 7925ba23abb2
Successfully tagged i2tch/mydocker:latest
```

Lancez le conteneur :

```
root@debian9:~/myDocker# docker run -it --name myDocker i2tch/mydocker
mycommand (18:07:12)
mycommand (18:07:15)
mycommand (18:07:18)
mycommand (18:07:21)
^Cmycommand (18:07:22)
mycommand (18:07:25)
mycommand (18:07:28)
```

```
^P^Q
root@debian9:~/myDocker#
```

Constatez que le conteneur est toujours en cours de fonctionnement :

```
root@debian9:~/myDocker# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
NAMES
140ecfdd80b7        i2tch/mydocker      "/entrypoint.sh myco…"   About a minute ago   Up About a minute
myDocker
b3380889eb75        i2tch/mongodb2      "docker-entrypoint.s…"   7 minutes ago        Up 7 minutes
27017/tcp           mongo2
d2ddb4f8ca8a        i2tch/mongodb       "bash"                   38 minutes ago       Up 36 minutes
mongo
c080793965de        nginx               "nginx -g 'daemon of…"   About an hour ago    Up About an hour
0.0.0.0:81->80/tcp   suspicious_sanderson
root@debian9:~/myDocker#
root@debian9:~/myDocker# docker logs myDocker | tail
mycommand (18:08:25)
mycommand (18:08:28)
mycommand (18:08:31)
mycommand (18:08:34)
mycommand (18:08:37)
mycommand (18:08:40)
mycommand (18:08:43)
mycommand (18:08:46)
mycommand (18:08:49)
mycommand (18:08:52)
```

Arrêtez le conteneur :

```
root@debian9:~/myDocker# docker stop -t 1 myDocker
myDocker
root@debian9:~/myDocker# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
b3380889eb75        i2tch/mongodb2      "docker-entrypoint.s…"   9 minutes ago       Up 9 minutes
27017/tcp           mongo2
d2ddb4f8ca8a        i2tch/mongodb       "bash"                   40 minutes ago      Up 38 minutes
mongo
c080793965de        nginx               "nginx -g 'daemon of…"   About an hour ago   Up About an hour
0.0.0.0:81->80/tcp   suspicious_sanderson
```

Démarrez le conteneur :

```
root@debian9:~/myDocker# docker start myDocker
myDocker
root@debian9:~/myDocker# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
140ecfdd80b7        i2tch/mydocker      "/entrypoint.sh myco…"   3 minutes ago       Up 10 seconds
myDocker
b3380889eb75        i2tch/mongodb2      "docker-entrypoint.s…"   10 minutes ago      Up 10 minutes
27017/tcp           mongo2
d2ddb4f8ca8a        i2tch/mongodb       "bash"                   40 minutes ago      Up 38 minutes
mongo
c080793965de        nginx               "nginx -g 'daemon of…"   About an hour ago   Up About an hour
0.0.0.0:81->80/tcp   suspicious_sanderson
```

Mettez le conteneur en pause :

```
root@debian9:~/myDocker# docker pause myDocker
```

```
myDocker
root@debian9:~/myDocker# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                NAMES
140ecfdd80b7        i2tch/mydocker      "/entrypoint.sh myco…"   3 minutes ago       Up 51 seconds (Paused)
myDocker
b3380889eb75        i2tch/mongodb2      "docker-entrypoint.s…"   10 minutes ago      Up 10 minutes
27017/tcp            mongo2
d2ddb4f8ca8a        i2tch/mongodb       "bash"                   41 minutes ago      Up 39 minutes
mongo
c080793965de        nginx               "nginx -g 'daemon of…"   About an hour ago   Up About an hour
0.0.0.0:81->80/tcp   suspicious_sanderson
```

Supprimez la pause :

```
root@debian9:~/myDocker# docker unpause myDocker
myDocker
root@debian9:~/myDocker# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
140ecfdd80b7        i2tch/mydocker      "/entrypoint.sh myco…"   4 minutes ago       Up About a minute
myDocker
b3380889eb75        i2tch/mongodb2      "docker-entrypoint.s…"   11 minutes ago      Up 11 minutes
27017/tcp            mongo2
d2ddb4f8ca8a        i2tch/mongodb       "bash"                   42 minutes ago      Up 40 minutes
mongo
c080793965de        nginx               "nginx -g 'daemon of…"   About an hour ago   Up About an hour
0.0.0.0:81->80/tcp   suspicious_sanderson
```

Lancez maintenant le conteneur avec un paramètre :

```
root@debian9:~/myDocker# docker rm -fv myDocker
myDocker
root@debian9:~/myDocker# docker run -d --name myDocker i2tch/mydocker "Up and Running"
0cf8c8c1bdf4cb05d9852900ecdf171ad9abad0fce29a9f040d5d8436285db65
root@debian9:~/myDocker# docker logs myDocker
Up and Running (18:13:33)
Up and Running (18:13:36)
Up and Running (18:13:39)
Up and Running (18:13:42)
root@debian9:~/myDocker#
```

Changez la valeur de la variable d'environnement **myVariable** :

```
root@debian9:~/myDocker# docker rm -fv myDocker
myDocker
root@debian9:~/myDocker# docker run -d --name myDocker --env myVariable=1 i2tch/mydocker
fbbe3b48c63310e37a3bad5fc962361c39c045a107f47980614efd6b2e8d3981
root@debian9:~/myDocker# docker logs myDocker
mycommand (18:14:47)
mycommand (18:14:48)
mycommand (18:14:49)
mycommand (18:14:50)
mycommand (18:14:51)
mycommand (18:14:52)
mycommand (18:14:53)
mycommand (18:14:54)
mycommand (18:14:55)
mycommand (18:14:56)
mycommand (18:14:57)
root@debian9:~/myDocker#
```

2.2 - Bonnes Pratiques liées au Cache

Opérations Non-Idempotentes

Créez un répertoire **bestp** ainsi que le fichier Dockerfile suivant :

```
root@debian9:~/myDocker# cd ..
root@debian9:~# mkdir bestp
root@debian9:~# cd bestp
root@debian9:~/bestp# vi Dockerfile
root@debian9:~/bestp# cat Dockerfile
FROM ubuntu:latest
RUN date +%N > /tmp/moment
ENTRYPOINT ["more"]
CMD ["/tmp/moment"]
```

Le fichier Dokerfile contient une opération non idempotente.

Important : Une opération idempotente est une opération qui aboutit systématiquement au même résultat quand elle est lancée dans le même contexte.

Compilez l'image :

```
root@debian9:~/bestp# docker build -t testcache .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:latest
 ---> 94e814e2efa8
```

```
Step 2/4 : RUN date +%N > /tmp/moment
 ---> Running in 6c8c677c1549
Removing intermediate container 6c8c677c1549
 ---> 66c3c88c57bb
Step 3/4 : ENTRYPOINT ["more"]
 ---> Running in e9658e591172
Removing intermediate container e9658e591172
 ---> 81cb68241ec9
Step 4/4 : CMD ["/tmp/moment"]
 ---> Running in 48974dc12faa
Removing intermediate container 48974dc12faa
 ---> c55a42a18572
Successfully built c55a42a18572
Successfully tagged testcache:latest
root@debian9:~/bestp#
```

Exécuter maintenant un premier conteneur à partir de l'image compilée :

```
root@debian9:~/bestp# docker run --name test1 -it testcache
369009216
```

Supprimez maintenant le conteneur et relancez la compilation de l'image :

```
root@debian9:~/bestp# docker rm test1
test1
root@debian9:~/bestp# docker build -t testcache .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:latest
 ---> 94e814e2efa8
Step 2/4 : RUN date +%N > /tmp/moment
 ---> Using cache
```

```
 ---> 66c3c88c57bb
Step 3/4 : ENTRYPOINT ["more"]
 ---> Using cache
 ---> 81cb68241ec9
Step 4/4 : CMD ["/tmp/moment"]
 ---> Using cache
 ---> c55a42a18572
Successfully built c55a42a18572
Successfully tagged testcache:latest
root@debian9:~/bestp#
```

Lancez un conteneur à partir de l'image re-compilée :

```
root@debian9:~/bestp# docker run --name test1 -it testcache
369009216
```

Important - Notez que les deux sorties des conteneurs sont identiques malgré le fait que la valeur de la commande date aurait du modifier le résultat obtenu lors de l'exécution du deuxième conteneur. La raison que ceci n'est pas le cas est l'utilisation dans la deuxième compilation du cache. Si cette commande avait été quelque chose de plus importante telle apt-get upgrade, le résultat pourrait être génant !

Pour contourner ce problème, il est possible d'utiliser l'option **–no-cache**. Malheureusement ceci produirait une compilation complète à chaque fois, même pour les opérations idempotentes. Il est donc conseillé de combiner les opérations non-idempotentes avec des opérations idempotentes dans la même ligne de commande afin d'invalider le cache pour cette ligne de commande seulement :

```
root@debian9:~/bestp# vi Dockerfile
root@debian9:~/bestp# cat Dockerfile
FROM ubuntu:latest
RUN date +%N > /tmp/moment \
```

```
    && echo "V1.1" > /tmp/version
ENTRYPOINT ["more"]
CMD ["/tmp/moment"]
```

Supprimez maintenant le conteneur et relancez la compilation de l'image :

```
root@debian9:~/bestp# docker rm test1
test1
root@debian9:~/bestp# docker build -t testcache .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:latest
 ---> 94e814e2efa8
Step 2/4 : RUN date +%N > /tmp/moment     && echo "V1.1" > /tmp/version
 ---> Running in 3d2a5cee6ac8
Removing intermediate container 3d2a5cee6ac8
 ---> 75d0498a9676
Step 3/4 : ENTRYPOINT ["more"]
 ---> Running in 88c0cec68659
Removing intermediate container 88c0cec68659
 ---> 2aee524c8da4
Step 4/4 : CMD ["/tmp/moment"]
 ---> Running in 82d2162bb701
Removing intermediate container 82d2162bb701
 ---> a54c4af89994
Successfully built a54c4af89994
Successfully tagged testcache:latest
```

Lancez un conteneur à partir de l'image re-compilée :

```
root@debian9:~/bestp# docker run --name test1 -it testcache
```

```
746997174
```