[Dockerfile - Build Docker images automatically III - RUN](http://www.bogotobogo.com/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically_3.php)

Continued from ...

Continued from [Dockerfile - Build Docker images automatically II - revisiting FROM, MAINTAINER, build context, and caching](http://www.bogotobogo.com/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically_2.php)

In this chapter, we're going to learn more on how to automate this process via instructions in Dockerfiles, especially, the **RUN** instruction.

Dockerfie - RUN

This section is from <http://docs.docker.com/reference/builder/>.

RUN has 2 forms:

RUN <command> (the command is run in a shell - /bin/sh -c - shell form)

RUN ["executable", "param1", "param2"] (exec form)

The RUN instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile.

Layering RUN instructions and generating commits conforms to the core concepts of Docker where commits are cheap and containers can be created from any point in an image's history, much like source control.

The **exec form** makes it possible to avoid shell string munging, and to RUNcommands using a base image that does not contain /bin/sh.

1. **Note**: To use a different shell, other than '/bin/sh', use the exec form passing in the desired shell. For example, RUN ["/bin/bash", "-c", "echo hello"].
2. **Note**: The exec form is parsed as a JSON array, which means that you must use double-quotes (") around words not single-quotes (').
3. **Note**: Unlike the shell form, the exec form does not invoke a command shell. This means that normal shell processing does not happen. For example, RUN [ "echo", "$HOME" ] will not do variable substitution on $HOME. If you want shell processing then either use the shell form or execute a shell directly, for example: RUN [ "sh", "-c", "echo", "$HOME" ].

Dockerfile 'RUN' sample

Here is our Dockerfile we're going to playing with in this chapter. We'll run instructions from this file step by step by uncommenting and commenting each line.

FROM debian:latest

MAINTAINER devops@bogotobogo.com

# 1 - RUN

RUN apt-get update && DEBIAN\_FRONTEND=noninteractive apt-get install -yq apt-utils

RUN DEBIAN\_FRONTEND=noninteractive apt-get install -yq htop

RUN apt-get clean

We have three instructions for RUN, and each of these instruction will create a new container, and at the completion of each instruction, it will become an image.

The following enviroment setting is to block any terminal output caused by some errors:

DEBIAN\_FRONTEND=noninteractive

The 2nd instruction, htop is to monitor processes in linux system. Then, we removes all packages from the package cache using apt-get clean.

Let's run docker build with v2 instead of v1:

$ docker build -t bogodevops/demo:v2 .

Sending build context to Docker daemon 33.56 MB

Sending build context to Docker daemon

Step 0 : FROM debian:latest

---> f6fab3b798be

Step 1 : MAINTAINER k@bogotobogo.com

---> Using cache

---> 511bcbdd59ba

Step 2 : RUN apt-get update && DEBIAN\_FRONTEND=noninteractive apt-get install -yq apt-utils

---> Running in 10ffa5b21a27

...

Setting up apt-utils (0.9.7.9+deb7u6) ...

---> e6e2c03b8efc

Removing intermediate container 10ffa5b21a27

Step 3 : RUN DEBIAN\_FRONTEND=noninteractive apt-get install -yq htop

---> Running in 2fe900ff207c

...

Setting up htop (1.0.1-1) ...

---> fac6e3168cfe

Removing intermediate container 2fe900ff207c

Step 4 : RUN apt-get clean

---> Running in 990373d72cc9

---> 327d400a953c

Removing intermediate container 990373d72cc9

Successfully built 327d400a953c

Listing images:

k@laptop:~/Documents/demo$ docker images -a

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

bogodevops/demo v2 327d400a953c 7 minutes ago 96.16 MB

<none> <none> fac6e3168cfe 7 minutes ago 96.16 MB

<none> <none> e6e2c03b8efc 7 minutes ago 95.12 MB

bogodevops/demo v1 511bcbdd59ba 2 hours ago 85.1 MB

debian latest f6fab3b798be 2 weeks ago 85.1 MB

<none> <none> f10807909bc5 2 weeks ago 85.1 MB

<none> <none> 511136ea3c5a 17 months ago 0 B

As we discussed in the previous chapter, if we run this again, it will be completed much faster thanks to caching:

k@laptop:~/Documents/demo$ docker build -t bogodevops/demo:v2 .

Sending build context to Docker daemon 33.56 MB

Sending build context to Docker daemon

Step 0 : FROM debian:latest

---> f6fab3b798be

Step 1 : MAINTAINER k@bogotobogo.com

---> Using cache

---> 511bcbdd59ba

Step 2 : RUN apt-get update && DEBIAN\_FRONTEND=noninteractive apt-get install -yq apt-utils

---> Using cache

---> e6e2c03b8efc

Step 3 : RUN DEBIAN\_FRONTEND=noninteractive apt-get install -yq htop

---> Using cache

---> fac6e3168cfe

Step 4 : RUN apt-get clean

---> Using cache

---> 327d400a953c

Successfully built 327d400a953c

docker run - launching container

k@laptop:~/Documents/demo$ docker run -it --rm bogodevops/demo:v2 /bin/bash

root@cf6430ffba1b:/# exit

exit

If we drop the :v2 tag in the command:

k@laptop:~/Documents/demo$ docker run -it --rm bogodevops/demo /bin/bash

Unable to find image 'bogodevops/demo' locally

Pulling repository bogodevops/demo

2014/11/24 18:55:36 Error: image bogodevops/demo not found

So, to make it work, we need to build default as latest:

k@laptop:~/Documents/demo$ docker build -t bogodevops/demo .

Now, if look at the images:

k@laptop:~/Documents/demo$ docker images -a

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

bogodevops/demo v2 327d400a953c 32 minutes ago 96.16 MB

bogodevops/demo latest 327d400a953c 32 minutes ago 96.16 MB

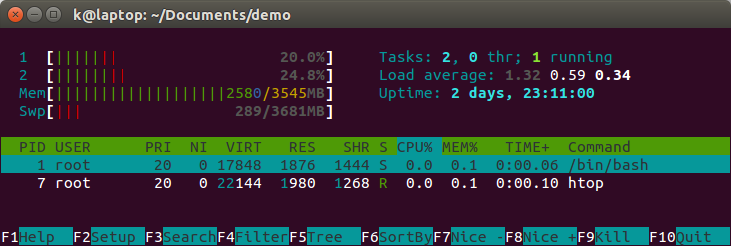
we have a new bogodevops/demo image tagged as 'latest'. So, from now on, we can execute docker run without the 'tag' since it'll look for 'latest' tag by default:

k@laptop:~/Documents/demo$ docker run -it --rm bogodevops/demo /bin/bash

root@88d48b65ebd7:/#

Now, we're in our Docker container for Debian, and htop has been installed.

root@88d48b65ebd7:/# htop



root@88d48b65ebd7:/# exit

exit

We should not see any container:

k@laptop:~/Documents/demo$ docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

No containers are hanging around!