# DOCKERFILES : BUILDING DOCKER IMAGES AUTOMATICALLY III - RUN



Ph.D. / Golden Gate Ave, San Francisco / Seoul National Univ / Carnegie Mellon / UC Berkeley / DevOps / Deep Learning / Visualization





## 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/ (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 `RUN` commands 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`:

(/DevOps/Docker/Docker_Commar

More on docker run command (docker run -it, docker run --rm, etc.) (/DevOps/Docker/Docker_Run_Cor

Docker Networks - Bridge Driver Network (/DevOps/Docker/Docker-Bridge-Driver-Networks.php)

Docker Persistent Storage (/DevOps/Docker/Docker_Containe

File sharing between host and container (docker run -d -p -v) (/DevOps/Docker/Docker_File_Sha

Linking containers and volume for datastore (/DevOps/Docker/Docker_Containe

Dockerfile - Build Docker images automatically I - FROM, MAINTAINER, and build context (/DevOps/Docker/Docker_Dockerfi

Dockerfile - Build Docker images automatically II - revisiting FROM, MAINTAINER, build context, and caching (/DevOps/Docker/Docker_Dockerfi

Dockerfile - Build Docker images automatically III - RUN (/DevOps/Docker/Docker_Dockerfi

Dockerfile - Build Docker images automatically IV - CMD (/DevOps/Docker/Docker_Dockerfi

Dockerfile - Build Docker images automatically V - WORKDIR, ENV, ADD, and ENTRYPOINT (/DevOps/Docker/Docker_Dockerfi

Docker - Apache Tomcat (/DevOps/Docker/Docker_Apache_

Docker - NodeJS (/DevOps/Docker/Docker-NodeJS.php)

Docker - NodeJS with hostname (/DevOps/Docker/Docker-NodeJS-with-hostname.php)

Docker Compose - NodeJS with MongoDB (/DevOps/Docker/Docker-Compose-Node-MongoDB.php)

```
$ docker image 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:

```
$ 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:

```
$ docker image 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 - Prometheus and Grafana with Docker-compose (/DevOps/Docker/Docker_Prometh

Docker - StatsD/Graphite/Grafana (/DevOps/Docker/Docker_StatsD_C

Docker - Deploying a Java EE JBoss/WildFly Application on AWS Elastic Beanstalk Using Docker Containers (/DevOps/Docker/Docker_Containe

Docker : NodeJS with GCP Kubernetes Engine (/DevOps/Docker/Docker-NodeJS-GCP-Kubernetes-Engine.php)

Docker : Jenkins Multibranch Pipeline with Jenkinsfile and Github (/DevOps/Docker/Docker-Jenkins-Multibranch-Pipeline-with-Jenkinsfile-and-Github.php)

Docker : Jenkins Master and Slave (/DevOps/Docker/Docker-Jenkins-Master-Slave-Agent-ssh.php)

Docker - ELK : ElasticSearch, Logstash, and Kibana (/DevOps/Docker/Docker_ELK_Elas

Docker - ELK 7.6 : Elasticsearch on Centos 7 (/DevOps/Docker/Docker_ELK_7_6_
Docker - ELK 7.6 : Filebeat on Centos 7 (/DevOps/Docker/Docker_ELK_7_6_

Docker - ELK 7.6 : Logstash on Centos 7 (/DevOps/Docker/Docker_ELK_7_6_

Docker - ELK 7.6 : Kibana on Centos 7 Part 1 (/DevOps/Docker/Docker_ELK_7_6_

Docker - ELK 7.6 : Kibana on Centos 7 Part 2 (/DevOps/Docker/Docker_ELK_7_6_

Docker - ELK 7.6 : Elastic Stack with Docker Compose (/DevOps/Docker/Docker_ELK_7_6_

Docker - Deploy Elastic Cloud on Kubernetes (ECK) via Elasticsearch operator on minikube (/DevOps/Docker/Docker_Kuberne

# docker run - launching container

```
$ docker container run -it --rm bogodevops/demo:v2 /bin/bash
root@cf6430ffba1b:/# exit
exit
```

If we drop the `:v2` tag in the command:

```
$ docker container 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:

```
$ docker image build -t bogodevops/demo .
```

Now, if look at the images:

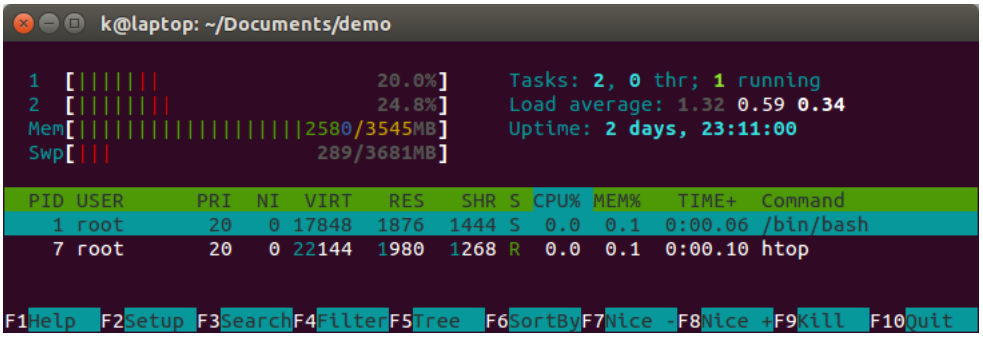
```
$ docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
bogodevops/demo     v2                  327d400a953c        32 minutes ago      96.16 MB
bogodevops/demo     latest              327d400a953c        32 minutes ago      96.16 MB
```

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:

```
$ docker container 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
```





Docker - Deploy Elastic Stack via Helm on minikube (/DevOps/Docker/Docker_Kuberne

Docker Compose - A gentle introduction with WordPress (/DevOps/Docker/Docker-Compose.php)

Docker Compose - MySQL (/DevOps/Docker/Docker-Compose-MySQL.php)

MEAN Stack app on Docker containers : micro services (/MEAN-Stack/MEAN-Stack-NodeJS-Angular-Docker.php)

Docker Compose - Hashicorp's Vault and Consul Part A (install vault, unsealing, static secrets, and policies) (/DevOps/Docker/Docker-Vault-Consul.php)

Docker Compose - Hashicorp's Vault and Consul Part B (EaaS, dynamic secrets, leases, and revocation) (/DevOps/Docker/Docker-Vault-Consul-B.php)

Docker Compose - Hashicorp's Vault and Consul Part C (Consul) (/DevOps/Docker/Docker-Vault-Consul-C.php)

Docker Compose with two containers - Flask REST API service container and an Apache server container (/DevOps/Docker/Docker-Compose-FlaskREST-Service-Container-and-Apache-Container.php)

Docker compose : Nginx reverse proxy with multiple containers (/DevOps/Docker/Docker-Compose-Nginx-Reverse-Proxy-Multiple-Containers.php)

Docker compose : Nginx reverse proxy with multiple containers (/DevOps/Docker/Docker-Compose-Nginx-Reverse-Proxy-Multiple-Containers.php)

Docker & Kubernetes : Envoy - Getting started (/DevOps/Docker/Docker-Envoy-Getting-Started.php)

```
root@88d48b65ebd7:/# exit
exit
```

We should not see any container:

```
$ docker container ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

No containers are hanging around!

## docker run - CMD

Continued in Dockerfile - Build Docker images automatically IV - CMD (http://www.bogotobogo.com/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically_4_CMD.php).

# Docker & K8s

1. Docker install on Amazon Linux AMI (/DevOps/Docker/Docker_Install_On_Amazon_Linux_AMI.php)
2. Docker install on EC2 Ubuntu 14.04 (/DevOps/Docker/Docker_Install_On_EC2_Ubuntu.php)
3. Docker container vs Virtual Machine (/DevOps/Docker/Docker_Container_vs_Virtual_Machine.php)
4. Docker install on Ubuntu 14.04 (/DevOps/Docker/Docker_Install_On_Ubuntu_14.php)
5. Docker Hello World Application (/DevOps/Docker/Docker_Hello_World_Application.php)
6. Nginx image - share/copy files, Dockerfile (/DevOps/Docker/Docker_Nginx_WebServer.php)
7. Working with Docker images : brief introduction (/DevOps/Docker/Docker_Working_with_images.php)
8. Docker image and container via docker commands (search, pull, run, ps, restart, attach, and rm) (/DevOps/Docker/Docker_Commands_for_Images_Container.php)
9. More on docker run command (docker run -it, docker run --rm, etc.) (/DevOps/Docker/Docker_Run_Command.php)
10. Docker Networks - Bridge Driver Network (/DevOps/Docker/Docker-Bridge-Driver-Networks.php)
11. Docker Persistent Storage (/DevOps/Docker/Docker_Container_Persistent_Storage_Data_Share.php)
12. File sharing between host and container (docker run -d -p -v) (/DevOps/Docker/Docker_File_Share_between_Host_and_Container.php)
13. Linking containers and volume for datastore (/DevOps/Docker/Docker_Container_Linking_Connect_with_linking_system_Communication_across_links_Environment_variables.php)
14. Dockerfile - Build Docker images automatically I - FROM, MAINTAINER, and build context (/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically.php)
15. Dockerfile - Build Docker images automatically II - revisiting FROM, MAINTAINER, build context, and caching (/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically_2.php)
16. Dockerfile - Build Docker images automatically III - RUN (/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically_3.php)
17. Dockerfile - Build Docker images automatically IV - CMD (/DevOps/Docker/Docker_Dockerfile_to_build_images_automatically_4_CMD.php)


Docker & Kubernetes : Envoy - Front Proxy (/DevOps/Docker/Docker-Envoy-Front-Proxy.php)

Docker & Kubernetes : Ambassador - Envoy API Gateway on Kubernetes (/DevOps/Docker/Docker-Envoy-Ambassador-API-Gateway-for-Kubernetes.php)

Docker Packer (/DevOps/Docker/Docker-Packer.php)

Docker Cheat Sheet (/DevOps/Docker/Docker-Cheat-Sheet.php)

Docker Q & A (/DevOps/Docker/Docker_Q_and_A

Kubernetes Q & A - Part I (/DevOps/Docker/Docker_Kuberne

Kubernetes Q & A - Part II (/DevOps/Docker/Docker_Kuberne

Docker - Run a React app in a docker (/DevOps/Docker/Docker-React-App.php)

Docker - Run a React app in a docker II (snapshot app with nginx) (/DevOps/Docker/Docker-React-App-2-SnapShot.php)

Docker - NodeJS and MySQL app with React in a docker (/DevOps/Docker/Docker-React-Node-MySQL-App.php)

Docker - Step by Step NodeJS and MySQL app with React - I (/DevOps/Docker/Step-by-Step-React-Node-MySQL-App.php)

Installing LAMP via puppet on Docker (/DevOps/Docker/Installing-LAMP-with-puppet-on-Docker.php)

Docker install via Puppet (/DevOps/Docker/Docker_puppet.p

Nginx Docker install via Ansible (/DevOps/Ansible/Ansible-Deploy-Nginx-to-Docker.php)

Apache Hadoop CDH 5.8 Install