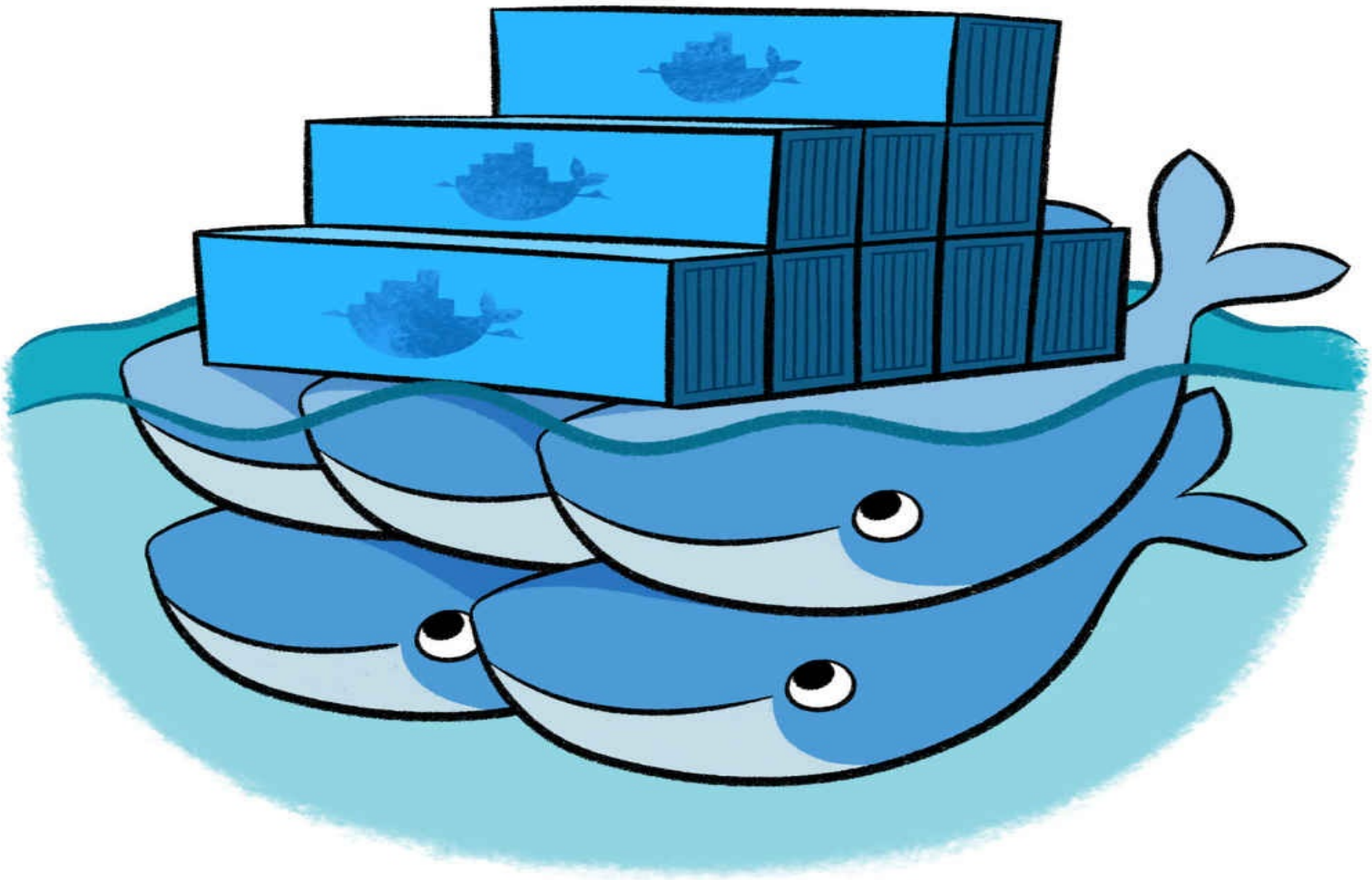
DOCKER

The Essential User Guide To Mastering Docker In No Time!



WALTER KING

Docker :

The Essential User Guide to Master Docker In No Time!

Bonus Gift For You!

Get **free** access to your complimentary book *"Amazon Book Bundle: Complete User Guides To 5 Amazon Products"* by clicking the link below.

>>>CLICK HERE TO DOWNLOAD<<<
(or go to: https://freebookpromo.leadpages.co/amazon-book-bundle/]

Table Of Contents

Introduction

Docker’s reliability and approach is the latest trend in technology. As technology is growing rapidly and world is adopting it so fast, you need to know how you are bring hosted through virtualizations. A lot of networks are connected to one single host where uncountable applications run all the time.

The reason why Dockers are becoming popular is the portability. The containers are easy to run anytime anywhere. It is basically a cloud where you have everything organized. The coding takes the lead for the application, keep them in the container and it is distributed to all the ones who are on the network with that single host.

There are cons and pros of everything but the Docker server makes sure that you privacy is fully maintained where you have the options to provide best service to the clients as well as protect your privacy as well.

There is no access to any of your personal documents or files unknowingly from this server. It does not consume much disk space neither it harms your computer. If you love to create commands and your interest is in the servers, then you have come to the right place.

This eBook will help you learn about the basic commands and their behavior so that you are well aware of what you are typing. Each command has its own significance due to which it creates a while new idea for you. Once entered wrong, it will bring the errors for you. The commands are visible in the images for you to try them as an example in your learning phase.

Easy and simple to access with the coding you can do wonders with this server especially if you have a better understanding on the coding part because that’s what it all takes to get it all done. There are total of five chapters which will give you enough guidance in order to understand the server with running it successfully on your own.

You do not need any assistant after following this eBook as a virtual guide for yourself. All the operating systems are the target for applying the instructions accordingly.

Chapter 1: Basics of installation and configuration

Docker was initially used in Ubuntu software of the computers but now since the trend is increased, you are able to download it to MAC, Windows and Linux as well. Downloading Docker is not rocket science except for that you will have to follow certain instructions for the installation.

Starting a software can get frustrating if you are new to it but have patience and try out the steps accordingly in order to manage the time efficiently. Installation of Docker in Ubuntu is no different than windows or Linux. So you can apply these instructions on any system you are using.

Installation of Docker into Ubuntu

Make sure that you choose the 64-bit Ubuntu in order for the successful installation of Docker in your computer. Here are some of the steps which needs to be followed:

- Update the droplet
- Availability of AUFs Support
- Add the Key which is required for the package verification to be installed.
- Connect the Docker repository to Apt Sources
- Update new repository
- At the end, download and install the Docker software.

Now, by extracting each step let’s move forward in order chronologically.

- Update the Apt Sources
- Login to the software as a user with sudo or root privileges.
- Click and open the terminal window.

Make sure that the APT works with the https method and the certifications are already installed. This is where you update package if these two conditions are not met.

```
$ sudo apt-get update

$ sudo apt-get install apt-transport-https ca-certificates
```

Add the new GPG Key by this method :

```
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3
A912897C070ADBF76221572C52609D
```

Open the link and create it if it is not found on the system.
Make sure to delete the entries which are there already.
According to the operating system, Select the entry and type it in.

```
/etc/apt/sources.list.d/docker.list
```

On Ubuntu Precise 12.04 (LTS)

```
deb https://apt.dockerproject.org/repo ubuntu-precise main
```

On Ubuntu Trusty 14.04 (LTS)

```
deb https://apt.dockerproject.org/repo ubuntu-trusty main
```

Ubuntu Wily 15.10

```
deb https://apt.dockerproject.org/repo ubuntu-wily main
```

Ubuntu Xenial 16.04 (LTS)

```
deb https://apt.dockerproject.org/repo ubuntu-xenial main
```

- Save the settings and close the file.
- Update the APT package index.

```
$ sudo apt-get update
```

Verify

```
$ apt-cache policy docker-engine
```

Now the new repository has been installed.

After all the prerequisites are installed in Ubuntu the install docker by following method:

- login to Ubuntu as a user with sudo privileges.
- Update the APT package.

```
$ sudo apt-get update
```

1. Install the Docker.

```
$ sudo apt-get install docker-engine
```

2. Start the docker daemon.

```
$ sudo service docker start
```

3. Verify docker is installed correctly.

```
$ sudo docker run hello-worldO
```

Restart the system in order to get the appropriate changes in the system so it can configure better.

For configurations,

- Create a Docker group

This group would have users in it which makes it accessible to the docker deamon as it starts for the readable and writable option.

- Upgrade Docker

For the installation of latest docker server with apt-get:

```
$ sudo apt-get upgrade docker-engine
```

Configuring Docker

Run a command on Docker

```
DOCKER_OPTS="-D --tls=true --tlscert=/var/docker/server.pem --tlskey=/var/docker/server
key.pem -H tcp://192.168.59.3:2376"
```

These options are available:

Enable -D mode

Set tls to true with the server certificate and key specified using --tlscert and --tlskey respectively

Listen for connections on

tcp://192.168.59.3:2376

At the end. Save and close the files to restart the Docker server. It will verify the command and run as specified.

```
$ ps aux | grep docker | grep -v grep
```

Chapter 2: Get started with Docker

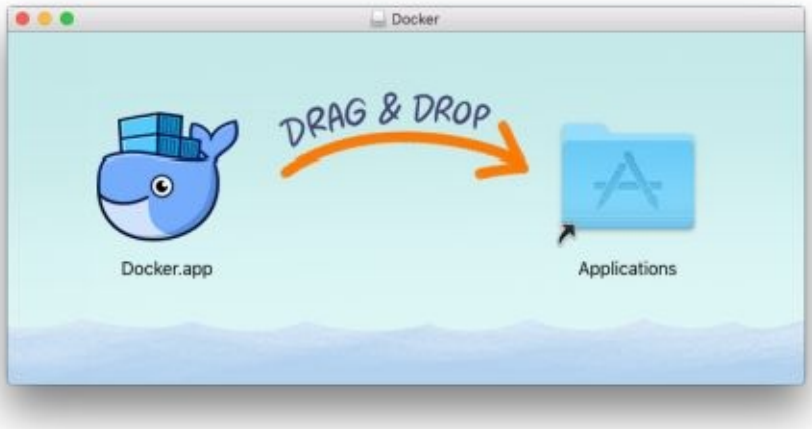
Once you have installed the docker into the computer, getting started with it comes in hand in hand. Make sure you follow the instructions whether you are using MAC, Windows, Linux or Ubuntu, all the instructions are same for every software.

There are some settings which you will have to input manually and they will remain there until you uninstall the server. Here are some of the steps which you can follow for the proper getting started session of the Docker.

Step 1:

Run the Mac Docker

- Click on the icon to which opens up the installer and after that take the whale by clicking once on it with dragging it to the folder of application as it is visible in the image below.



By dragging the whale, it might ask for the password of your system so make sure to add that without any worry. If not then it will automatically connect the links to the docker server.

Now since the docker is ready to use,

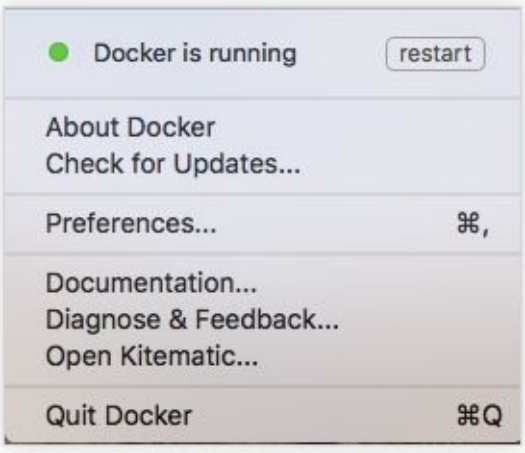
- Click twice on the docker.app to start the server in order to run it.

Check the top bar of your screen if you are using MAC and the bottom bar if you are using Windows. There will be a little icon of the whale shape which tells you that Docker server is running and you can access is quickly through the bar.

Most of the times when do not shut your computer after the installation of the program and start using it right after installation, it gives you a pop up message and also directs you to what you need to get done in order to get full access of the server. You can click on the whale shape icon on the bar to disable the pop up message as well.

When you are running the Docker server, you will be getting this message by informing that it is running on your computer right now. When you are done, click on the “got it” button to close the message.

If you wish to change the preferences, click on the whale shape icon on the bar and check out the other options as well. Check the image below for proper understanding.



Now, click on “About Docker” to know which version of server do you have. When you have to stop using the Docker completely, you can simply click on “Quit Docker” and it will close all the files or programs related to Docker.

Step 2:

Checking the Server Versions

In order to understand your server better, it is necessary that you know the version of certain components of Docker. You will have to run certain commands on Docker to check the versions for machine, compose and engine.

Here are the versions which you need to enter in order to get the versions.

`docker-compose`

`docker`

`docker-machine`

Make sure that these all are updated with the docker.app. Heres an image which you should be getting after testing the versions.

Note that these are the versions only for MAC operating system. It may vary upon the operation systems.

```
$ docker --version
Docker version 1.12.0, build 8eab29e

$ docker-compose --version
docker-compose version 1.8.0, build f3628c7

$ docker-machine --version
docker-machine version 0.8.0, build b85aac1
```

Step 3:

Surf the application software and run it with assumptions

- You have to make sure that the Docker server is working fine so you will need to run certain commands in order to check it accuracy. As an example you can try `docker run hello-world`

- To make sure that it is running fine.
- Another example can be to try a longer example which is to access the web on the browser by entering this :

Now you have this window opened in front of you

```
1.  CONTAINER ID        IMAGE               COMMAND                  CREATE
    D               STATUS              PORTS                           NAMES
```

```
1.  56f433965490        nginx               "nginx -g 'daemon off"   About
    a minute ago    Up About a minute   0.0.0.0:80->80/tcp, 443/tcp   webserver
```

Note that there could be latest BETA versions of this command.

Now you have to follow certain steps in order to access the web browser.

Check out the images below and do it accordingly.

- Run
  `docker ps`

```
docker run -d -p 80:80 --name webserver nginx
```

If you wish to you can remove the images and the container by stopping it. Until you do not stop, the web server will keep on running. You can stop with this entry, `docker stop webserver`

And restart it with this entry anytime with replacing the stop word with start in the above entry. To remove an image fully on the web server, input this entry `docker rmi <imageID>|<imageName>`.

You can check out more examples by walking through the data base on the website as well as on the server where it will give you the option for examples to make it easier and simple for you to use it.

If you wish to check the running time, then click on preferences by accessing the whale shape icon on the bar.

Here are some of the tips which you can use for the settings of the Docker server.

- The General tab will give you the option to set the automatic time when you log in, if you uncheck it, you will have to manually start the server whenever you open your computer.
- If you do not want the time machine for backups, you can leave the option unchecked.
- Make sure to adjust the settings according to the image above in order for the proper use of the Docker server. If it is like that by default, then you are good to go.

The next tab is Advanced options.

- This option is suitable if you wish to save the public or private images with the registry. You can add your own URLs and register with the hosts.
- Rest of the settings are automatically done. If you wish to add any proxy, you can do it in the HTTP bar.

Next tab is File sharing

Here you are free to decide whether you want to share the files with the containers.

If you wish to add then click on the plus sign and add any directory you want.

Last tab is Privacy

The most concerned area of the users. You need to check the boxes for the options of “send usage information” and also “send crash reports” so you know that it is performing better for the clients or not. If you do not wish to receive the automatic report then you can disable the options anytime you want.

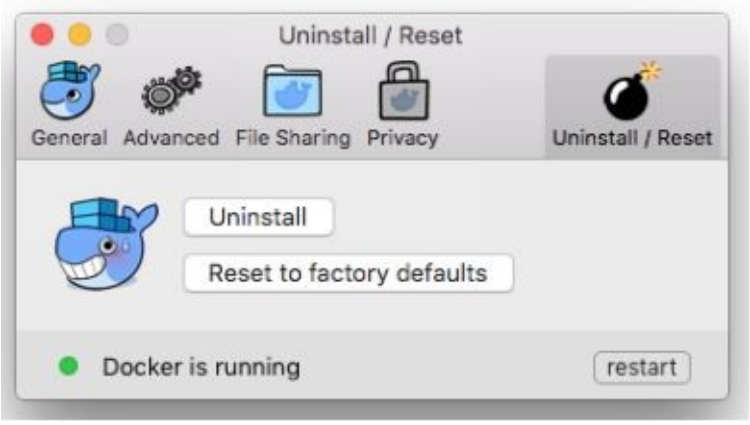
```
$ /Applications/Docker.app/Contents/MacOS/Docker --uninstall
Docker is running, exiting...
Docker uninstalled successfully. You can move the Docker application to the trash.
```

On the corner right, you will be able to locate the reset/uninstall option. You can easily click on that and it will reset all the settings you have made or uninstall it completely from your computer data.

You will be needing the command in order to uninstall the server which is

```
<DockerforMacPath> --uninstall
```

If there is any problem occurring in the server, you can uninstall and reinstall it later with the same directions.




Welcome to nginx!
localhost
Vicky
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

Chapter 3: Learn to build images with DockerFile

You can make images with the DockerFile of you own with certain commands which you need to know. It is simple understanding of commands which help you get through any software. It is quick and simple to use so let's get started with it. There are 4 simple steps which you can follow for the positive results out of Dockerfile.

Step 1:

Write up

- Access the window where you can enter commands
- Type this entry

```
mkdir mydockerbuild
```

- By pressing Enter, it will build a context for you which means that everything will be built automatically needed for the image to be created.
- Now, type in the new entry for directory.

```
1. FROM docker/whalesay:latest
```

- You will have to make the Dockerfile by typing the image below and pressing enter

```
touch Dockerfile
```

- Now by entering the data below, you will be able to recognize whether it is working or not.

```
1. $ cd mydockerbuild
```

Now once you are done with that, open the Dockerfile and enter this text

```
1.      RUN apt-get -y update && apt-get install -y fortunes
```

Insert the next text in order to get the program for the image.

```
$ ls Dockerfile

Dockerfile
```

After this command is entered, the fortune program gets added into the software. Now the image is available with the software so you can access it and run it through which the image is available.

```
docker build -t docker-whale .
```

At the end, it is important to recheck the work in order to know whether you have done it right or not. Check out the image below and if your file looks like this then you are on the right track.

You are all good to start building your own personalized image.

Step 2:

```
CMD /usr/games/fortune -a | cowsay
```

Creating your own Image at Dockerfile

- First type the following image in the command box.

Add:

```
1.$ cat Dockerfile
2.FROM docker/whalesay:latest
3.
4.RUN apt-get -y update && apt-get install -y fortunes
5.
6.      CMD /usr/games/fortune -a | cowsay
```

Now by creating a new command, type

```
1.$ docker build -t docker-whale .
2.Sending build context to Docker daemon 158.8 MB
3....snip...
4.Removing intermediate container a8e6faa88df3
5.Successfully built 7d9495d03763
```

```
1.FROM docker/whalesay:latest
2.RUN apt-get -y update && apt-get install -y fortunes
3.      CMD /usr/games/fortune -a | cowsay
```

It will consume some seconds to settle and then run by giving you the best results so you will have be patient in that time. But before proceeding, you need to know about the Dockerfile build procedure which is in next step.

Step 3:

The build procedure of Dockerfile

When you create the image, it saves into the machine of Dockerfile. There are certain messages which you need to understand in order to carry on forward with better results.

- Type in this text in the command box

```
Sending build context to Docker daemon 158.8 MB
```

It will pop up this command, which means it has your image in the record already which you created in step 2.

```
Step 1 : FROM docker/whalesay:latest
 ---> fb434121fc77
```

Next you need to get the package of apt - get
This is how you entry the data for it :

```
Step 2 : RUN apt-get -y update && apt-get install -y fortunes
 ---> Running in 27d224dfa5b2
Ign http://archive.ubuntu.com trusty InRelease
Ign http://archive.ubuntu.com trusty-updates InRelease
Ign http://archive.ubuntu.com trusty-security InRelease
Hit http://archive.ubuntu.com trusty Release.gpg
....snip...
Get:15 http://archive.ubuntu.com trusty-security/restricted amd64 Packages [14.8 k
B]
Get:16 http://archive.ubuntu.com trusty-security/universe amd64 Packages [134 kB]
Reading package lists...
 ---> eb06e47a01d2
```

Next again, build it for the Fortunes program:

```
Removing intermediate container e2a84b5f390f
Step 3 : RUN apt-get install -y fortunes
 ---> Running in 23aa52c1897c
Reading package lists...
Building dependency tree...
Reading state information...
The following extra packages will be installed:
  fortune-mod fortunes-min librecode0
Suggested packages:
  x11-utils bsdmainutils
The following NEW packages will be installed:
  fortune-mod fortunes fortunes-min librecode0
0 upgraded, 4 newly installed, 0 to remove and 3 not upgraded.
Need to get 1961 kB of archives.
After this operation, 4817 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ trusty/main librecode0 amd64 3.6-21 [771 k
B]
...snip......
Setting up fortunes (1:1.99.1-7) ...
Processing triggers for libc-bin (2.19-0ubuntu6.6) ...
 ---> c81071adeeb5
Removing intermediate container 23aa52c1897c
```

At the end, when it will finish building the Dockerfile, you will be given the output of it in a form of reports.

```
Step 4 : CMD /usr/games/fortune -a | cowsay
 ---> Running in a8e6faa88df3
 ---> 7d9495d03763
Removing intermediate container a8e6faa88df3
Successfully built 7d9495d03763
```

Step 4:

Run the Image you created

- Type in the image below in the command box

```
docker images
```

- Next, you will be able to see the results something like this in the image below :

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker-whale        latest              7d9495d03763        4 minutes ago       273.
7 MB
docker/whalesay     latest              fb434121fc77        4 hours ago         247
MB
hello-world         latest              91c95931e552        5 weeks ago         910
B
```

- Type in the command box docker run docker – whale

```
- $ docker run docker-whale
```

```
 _______________________________________
/ "He was a modest, good-humored boy. It  \
\ was Oxford that made him insufferable." /
 ---------------------------------------
        \
         \
          \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \_______ o          __/
        \    \        __/
          \____\______/
```

Now the image is available for you, your own image which is original and not taken from anywhere but by following certain steps of commands, you were able to make your own unique image.

Chapter 4: Learn how to Work with the DockerContext

The Docker server can make the images on its own by reading the commands on the server. Basically the Dockerfile is a document which has all the text which a user inputs. It makes sure to combine all the commands to make an image for you within no time. The docker build helps you make and intact several commands which makes the images.

The Use of DockerContext

The Docker build takes the input of the text path which you have typed in the command box and makes the DockerContext out of it automatically.

```
$ docker build .

Sending build context to Docker daemon  6.51 MB

...
```

First of all, you need to have the directory completely spacious which means there should not be any file or directory in it. The Docker server sends all the build in the form of context repeatedly. All the builds are commanded by the server. Make sure to keep the files which are only needed otherwise it will be confusing for you. You have to pay proper attention while doing this.

```
Warning: Do not use your root directory, /, as the PATH as it causes the build to transfer the
entire contents of your hard drive to the Docker daemon.
```

There are proper instructions available with the software to build the context for whichever file you need. If you want good output from the context file, then make sure to choose the ones which are necessary only. Now add the command if you wish to remove any file which is not needed there.

```
.dockerignore
```

You can also find the file by en`-f` g I in order to locate the file on the Docker system.

For the new image, you will be able to see this message which will tell you for the success of the entry.

```
$ docker build -t shykes/myapp .
```

If you wish to tag the images into different places then you will have to add for the command to provide you desirable results.

```
$ docker build -t svendowideit/ambassador .
Sending build context to Docker daemon 15.36 kB
Step 1 : FROM alpine:3.2
 ---> 31f630c65071
Step 2 : MAINTAINER SvenDowideit@home.org.au
 ---> Using cache
 ---> 2a1c91448f5f
Step 3 : RUN apk update &&      apk add socat &&        rm -r /var/cache/
 ---> Using cache
 ---> 21ed6e7fbb73
Step 4 : CMD env | grep _TCP= | (sed 's/.*_PORT_\([0-9]*\)_TCP=tcp:\/\/\(.*\):\(.*
\)/socat -t 100000000 TCP4-LISTEN:\1,fork,reuseaddr TCP4:\2:\3 \&/' && echo wait)
| sh
 ---> Using cache
 ---> 7ea8aef582cc
Successfully built 7ea8aef582cc
```

```
$ docker build -t shykes/myapp:1.0.2 -t shykes/myapp:latest .
```

All the step by step approach is taken into consideration by the Docker server. If you try to do all at once, it will mix the stuff for you. It is important that you follow the step by step instructions by seeing the result of every image at the end. When you are done, the server will clear out all the context on its own which you have sent to it.

All the commands inputs are done separately. They perform individually as well in order to build the new image. If the server wants, it will take the data again from the history to re build the image for you. here is an example of the cache which you may see when you are done with the DockerContext usage.

`Dockerfile`

```
# Comment
INSTRUCTION arguments
```

This is how it looks when you are done with it. If you see the same on your screen similar to this image above, then you are on the right track.

The Formatting

This is how you can format the

```
# Comment
RUN echo 'we are running some # of cool things'
```

The word instruction on the image is not large letters so make sure to type it in lower case. That is just to differentiate it to make it visible enough. The server creates the images from the build and make sure to deliver it as it is. It starts with the comment with the hashtag on its left. This is how the command looks like:

```
FROM windowsservercore
COPY testfile.txt c:\\
RUN dir c:\
```

The next is the escape mode of formatting which should look like this to you

```
# escape=\ (backslash)
```

Or might look this way :

```
# escape=` (backtick)
```

These need to be typed in the command box to instruct the build in order to multiply the Dockerfile. You need to know that the escape mode is not to run the command but done as to end the procedure. Basically when you enter a command, it is like an instruction for the server to perform. When you give the right directions, it will head towards something you expect of.

Then it will give you this output. Check the image below:

Some of the signs could be confusing but make sure to add it exactly the way it is shown on the image. If you do a mistake then it will ruin it and consider it as an error. The commands easily catch the errors and you will have to find out a different PATH or URL to consider your escape mode. Heres how it looks what you should be getting in your screen. Note: This is how it will show on the Windows operating system. It may differ on the other operating systems but slightly. By entering this :

```
# escape=`

FROM windowsservercore
COPY testfile.txt c:\
RUN dir c:\
```

And as the output, you get this on your screen. Check the image below:

```
PS C:\John> docker build -t succeeds --no-cache=true .
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM windowsservercore
 ---> dbfee88ee9fd
Step 2 : COPY testfile.txt c:\
 ---> 99ceb62e90df
Removing intermediate container 62afbe726221
Step 3 : RUN dir c:\
 ---> Running in a5ff53ad6323
 Volume in drive C has no label.
 Volume Serial Number is 1440-27FA

 Directory of c:\

03/25/2016  05:28 AM    <DIR>          inetpub
03/25/2016  04:22 AM    <DIR>          PerfLogs
04/22/2016  10:59 PM    <DIR>          Program Files
03/25/2016  04:22 AM    <DIR>          Program Files (x86)
04/18/2016  09:26 AM                 4 testfile.txt
04/22/2016  10:59 PM    <DIR>          Users
04/22/2016  10:59 PM    <DIR>          Windows
               1 File(s)              4 bytes
               6 Dir(s)  21,252,689,920 bytes free
 ---> 2569aa19abef
Removing intermediate container a5ff53ad6323
Successfully built 2569aa19abef
PS C:\John>
```

The Ignore File

Since we have mentioned before that to empty the directory you have to use the `.dockerignore`

It extracts the files from the bottom to bring it up on the server. If the combination of both matches then it pops up the files for you. It stops the server from sending the unnecessary data to the host.

Sometimes, you tend to have files which are password protected, those files are not touched at all. They are protected and if you enter this ignore command then you would not have to worry about any sharing at all. It will be all limited however you wish to format it.

Even if you type in command for a word which you do not remember fully, it will pull out all the files or directories which consist that word so you can locate it easily. If you remember the exact word, then you will be getting the exact same file which you would be looking for in the server. Finding anything in the server is made easier by adding the add or stop command. Heres how the docker ignore file would should look as you enter the command.

```
# comment
*/temp*
*/*/temp*
temp?
```

```
*.md
!README.md
```

The meaning of each command is as followings to give you a clear understanding of it. If you wish to learn properly from this eBook then proper guidance is necessary for each word which you enter as a command so you know what actions it performs and how it effects the server.

Rule	Behavior
`# comment`	Ignored.
`*/temp*`	Exclude files and directories whose names start with `temp` in any immediate subdirectory of the root. For example, the plain file `/somedir/temporary.txt` is excluded, as is the directory `/somedir/temp`.
`*/*/temp*`	Exclude files and directories starting with `temp` from any subdirectory that is two levels below the root. For example, `/somedir/subdir/temporary.txt` is excluded.
`temp?`	Exclude files and directories in the root directory whose names are a one-character extension of `temp`. For example, `/tempa` and `/tempb` are excluded.

The matching of the files are done automatically by the server so you do not have to perform any command over that. There are elements which are removed automatically as well, those which are not needed.

There are some lines which would be left empty that means that the server is not accepting them but it is valid to leave them blank. There are some phrases which would be with the exclamation marks which means that they are exceptions for you to leave them. Such as here is an example:

The readme would be in lowercase but just to differentiate it, it is written as uppercase.

The From Message

The message from would be creating the image but you have to enter form in the command before typing in the image in brackets. Here is an example which you should follow:

```
FROM <image>
```

Or

```
FROM <image>:<tag>
```

Or

```
FROM <image>@<digest>
```

It is like an instruction for the image to appear for you at the end. It is to use for the valid images only. When you want an image, you can pull it out of the server by using this technique. It is short and quick to handle.

To generate the images you have to use the maintainer:

```
MAINTAINER <name>
```

Now comes the main part:

The Run

It has two important commands which are as followings:

- `RUN <command>` (*shell* form, the command is run in a shell, which by default is `/bin/sh -c` on Linux or `cmd /S /C` on Windows)
- `RUN ["executable", "param1", "param2"]` (*exec* form)

It starts with creating a brand new layer for you for the image which is open for you. It gives you the output for which you have done all the work. It will be moved to the Dockerfile automatically. When you create a layer, it simply makes the concepts clear for the image to be saved at one place. When you have to run the image again, enter this command:

```
RUN /bin/bash -c 'source $HOME/.bashrc ;\
echo $HOME'
RUN /bin/bash -c 'source $HOME/.bashrc ; echo $HOME'
```

Remove the Volumes

Even if you delete the things on the data of server, it is still there which would be compiling over another. You need to remove it completely once you think you do not need it anymore.

There is data available from the hosts unknowingly so it is necessary to empty it once in a while. The serve should be cleaned inside for it to keep on running smoothly. Here is the image which you can enter in order for the complete cleaning of volumes:

```
$ docker run --rm -v /foo -v awesome:/bar busybox top
```

Since the server is shared with the hosts, there could be multiple things approaching your way. Apply your settings in a way which can completely remove the tools which are not needed. It does not corrupt the files and keep the system running smoothly as well. The application should be made for a complete write up with the space.

Backup Data

It is important to backup the data in the server so that if your computer crashes or any mis-happenings occurs, you can recover it back by installing the server again on the other computer. It is always better to be safe rather than doing it all again.

There are volumes container which help you keep all the data in it and restore it anywhere anytime but with your authentication. Check out the image below for the commands to be entered in order to save the data from crashing.

```
$ docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu tar cvf /backup/
backup.tar /dbdata
$ docker run -v /dbdata --name dbstore2 ubuntu /bin/bash
$ docker run --rm --volumes-from dbstore2 -v $(pwd):/backup ubuntu bash -c "cd /d
bdata && tar xvf /backup/backup.tar --strip 1"
```

This involves the backup and the restoration commands which can help you in the Ubuntu operating system but they do not vary on any other operating system. By creating this new container, it helps you save from a lot of hassle later on. These are machines which can cause you trouble anytime so keep a caution with you all the time by accessing your own backup database.

Chapter 5: Enhanced functionality and shortcut for Docker

The Docker file has the commands which keeps on repeating but with few different changes which can make a whole new thing for you in the server. There are different combinations which can help you make new image and build it directly saved to the server.

The shortcuts allow you to use the server in a faster way with formatting it. Three areas which we are going to cover here is Users and groups, Files and directories, and Miscellaneous. To make it simpler for you, let's go through each to explain.

Users and Groups shortcuts

There are no names for the users on the Docker server. You have to depend on the images, IDs and so on. It may not be so clear to you but what you can do is to run the command these need to be consistent in order for the best results.

Here is a good way to add the user by creating the user id in the shortcut. It is not necessary to do so but you can access it and make it feasible for you to create one for yourself to avoid any confusion. It is to run the programs but for login, you do not need any user name. Here are some of the commands which you can use to create the usernames:

```
RUN adduser username ...
```

```
system
```

```
no_login
```

```
no_password
```

```
group
```

```
gecos
```

You can create the username with any name you want and add into the group by

adding the command such as addgroup or addgroupuser.

Here is another example which you can apply as a shortcut.

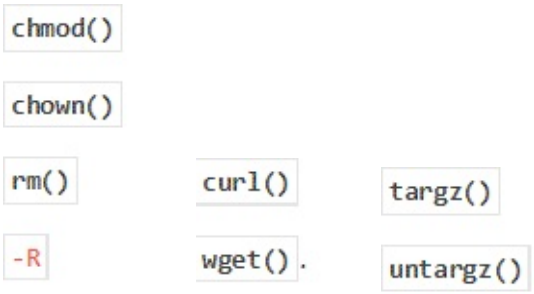
```
df = DockerFile(...)
... # Additional build commands
df.run(addgroupuser('nginx', 2000))

df = DockerFile(...)
... # Additional build commands
df.run_all(
    addgroup('apps', 2001),
    addgroupuser('nginx', 2000, ['apps']),
```

Files and Directories shortcuts

If you want to make the directories again then you have to use this command

Automatically the directories are created in the server but if you wish to create a separate one then you can enter that command which will make the separate directory for you. These commands can be used for the changes repeatedly on the server. Here is the image which you can use as reference:



This is how the directory looks when it is added by the commands:

```
df = DockerFile(...)
... # Additional build commands
df.run(mkdir_chown('/var/lib/app', 2001, 'u=rwX,go='))
```

Miscellaneous

Know that you can create any dockerfile by “add” command. You can also build the dockerfile with the same command. There are two main functions of miscellaneous.

These two functions can find out the directories for you in the file and locate the

exact thing which you are looking for. You can reach the archives without any complications with the commands. Here are the commands which you can enter and reach at the designated directory.

Conclusion

This guide is helpful and useful if you are not aware of Docker server. Mostly the people who are aware of the server, knows how to run it thoroughly because they are engineers.

They are taught to do the coding and commands which helps them with every software. This software is simple to use with slightly different commands which will help you create images also. There are minor functionalities and shortcuts which help you to get through the server quickly instead of taking the longer route.

The more you use it, the more you will be comfortable with the commands. All it requires is the practice. You need to make sure that the client privacy is not disturbed also and they are satisfied as well which is why it is important to be expert in this and understand the little things also.

There are many tutorials available for the Docker server with examples as well where you can see exactly what you need to do with the basic start. Once you are aware of the server and knows the in and out of it, you will enjoy it with providing excellent results for yourself and for the client on the other end.

It is an era of technology and there can’t be anything better than learning about the servers because it all starts from the server where everyone can’t access except the ones who are professionals in it.

Technology is rapidly changing and learning the basics would help you create something big as you keep on learning. This eBook makes sure to cover the main aspects of Docker server from the installation process till the shortcuts of the functions which may be useful for the users so that they do not have to waste much time through the longer way.

The docker build and docker image is the main things which helps you serve on this Dockerfile. It increases the efficiency of this server and makes sure it is effective for the users.

Hopefully this eBook comes handful for the ones who are eager to learn about Docker server!