**1.)**[Provide static IP to docker containers via docker-compose](https://stackoverflow.com/questions/39493490/provide-static-ip-to-docker-containers-via-docker-compose)

|  |  |
| --- | --- |
|  | I'm trying to provide static IP address to containers. I understand that I have to create a custom network. I create it and the bridge interface is up on the host machine (Ubuntu 16.x). The containers get IP from this subnet but not the static I provided.  Here is my docker-compose.yml:  version: '2'  services:  mysql:  container\_name: mysql  image: mysql:latest  restart: always  environment:  - MYSQL\_ROOT\_PASSWORD=root  ports:  - "3306:3306"  networks:  - vpcbr  apigw-tomcat:  container\_name: apigw-tomcat  build: tomcat/.  ports:  - "8080:8080"  - "8009:8009"  networks:  - vpcbr  depends\_on:  - mysql    networks:  vpcbr:  driver: bridge  ipam:  config:  - subnet: 10.5.0.0/16  gateway: 10.5.0.1  aux\_addresses:  mysql: 10.5.0.5  apigw-tomcat: 10.5.0.6  The containers get 10.5.0.2 and 10.5.0.3, instead of 5 and 6. |

🡪Note that I don't recommend a fixed IP for containers in Docker unless you're doing something that allows routing from outside to the inside of your container network (e.g. macvlan).

DNS is already there for service discovery inside of the container network and supports container scaling. And outside the container network, you should use exposed ports on the host. With that disclaimer, here's the compose file you want:

version: '2'

services:

mysql:

container\_name: mysql

image: mysql:latest

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=root

ports:

- "3306:3306"

networks:

vpcbr:

ipv4\_address: 10.5.0.5

apigw-tomcat:

container\_name: apigw-tomcat

build: tomcat/.

ports:

- "8080:8080"

- "8009:8009"

networks:

vpcbr:

ipv4\_address: 10.5.0.6

depends\_on:

- mysql

networks:

vpcbr:

driver: bridge

ipam:

config:

- subnet: 10.5.0.0/16

gateway: 10.5.0.1

🡪Does not work for me: User specified IP address is supported only when connecting to networks with user-configured subnets

"Note: Additional IPAM configurations, such as gateway, are only honored for version 2 at the moment.

**2.)**[Docker-compose check if mysql connection is ready](https://stackoverflow.com/questions/42567475/docker-compose-check-if-mysql-connection-is-ready)

I am trying to make sure that my app container does not run migrations / start until the db container is started and READY TO accept connections.

So I decided to use the healthcheck and depends on option in docker compose file v2.

In the app, I have the following

app:

...

depends\_on:

db:

condition: service\_healthy

The db on the other hand has the following healthcheck

db:

...

healthcheck:

test: TEST\_GOES\_HERE

timeout: 20s

retries: 10

I have tried a couple of approaches like :

1. making sure the db DIR is created  test: ["CMD", "test -f var/lib/mysql/db"]
2. Getting the mysql version:  test: ["CMD", "echo 'SELECT version();'| mysql"]
3. Ping the admin (marks the db container as healthy but does not seem to be a valid test)  test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost"]

Does anyone have a solution to this?

**🡪**version: "2.1"

services:

api:

build: .

container\_name: api

ports:

- "8080:8080"

depends\_on:

db:

condition: service\_healthy

db:

container\_name: db

image: mysql

ports:

- "3306"

environment:

MYSQL\_ALLOW\_EMPTY\_PASSWORD: "yes"

MYSQL\_USER: "user"

MYSQL\_PASSWORD: "password"

MYSQL\_DATABASE: "database"

healthcheck:

test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost"]

timeout: 20s

retries: 10

The api container will not start until the db container is healthy (basically until mysqladmin is up and accepting connections.)

mysqladmin ping will return a false positive if the server is running but not yet accepting connections.

🡪

|  |  |
| --- | --- |
|  | If you can change the container to wait for mysql to be ready do it.  If you don't have the control of the container that you want to connect the database to, you can try to wait for the specific port.  For that purpose, I'm using a small script to wait for a specific port exposed by another container.  In this example, **myserver** will wait for port **3306** of **mydb** container to be reachable.  # Your database  mydb:  image: mysql  ports:  - "3306:3306"  volumes:  - yourDataDir:/var/lib/mysql  # Your server  myserver:  image: myserver  ports:  - "....:...."  entrypoint: ./wait-for-it.sh mydb:3306 -- ./yourEntryPoint.sh  You can find the script wait-for-it documentation [here](https://github.com/vishnubob/wait-for-it) |

I tried using wait-for-it.sh earlier but it overrides the default Dockerfile right? How does the entrypoint.sh look like

The entrypoint depends on your image. You can check it with docker inspect <image id>. This should wait for the service to be available and call your entry point.

🡪I modified the docker-compose.yml as per the following example and it worked.

mysql:

image: mysql:5.6

ports:

- "3306:3306"

volumes:

# Preload files for data

- ../schemaAndSeedData:/docker-entrypoint-initdb.d

environment:

MYSQL\_ROOT\_PASSWORD: rootPass

MYSQL\_DATABASE: DefaultDB

MYSQL\_USER: usr

MYSQL\_PASSWORD: usr

healthcheck:

test: mysql --user=root --password=rootPass -e 'Design your own check script ' LastSchema

In my case ../schemaAndSeedData contains multiple schema and data seeding sql files. Design your own check script can be similar to following 'select \* from LastSchema.LastDBInsert'.

While web dependent container code was

depends\_on:

mysql:

condition: service\_healthy

This may work for you but I am unsure whether or not this is supported in all MySQL engines.

mysql:5.6 supports it fully..

I'm talking about database engines like InnoDB, MyISAM etc. Is LastSchema.LastDBInsert a MySQL default or database engine specific?

No it is not a default in mysql either. It was just a sample. a dummy query.

3.)[Docker Compose wait for container X before starting Y](https://stackoverflow.com/questions/31746182/docker-compose-wait-for-container-x-before-starting-y)

|  |  |
| --- | --- |
|  | I am using rabbitmq and a simple python sample from [here](https://www.rabbitmq.com/tutorials/tutorial-one-python.html) together with docker-compose. My problem is that I need to wait for rabbitmq to fully started. From what I searched so far, I dont know how to wait with container x ( in my case worker ) until y (rabbitmq) is started.  I found this [blogpost](http://blog.chmouel.com/2014/11/04/avoiding-race-conditions-between-containers-with-docker-and-fig/) where he checks if the other host is online. I also found this [docker command](https://docs.docker.com/reference/commandline/wait/):  **wait**  Usage: docker wait CONTAINER [CONTAINER...]  Block until a container stops, then print its exit code.  Waiting for a container to stop is maybe not what I am looking for but if it is, is it possible to use that command inside the docker-compose.yml ? My solution so far is to wait some seconds and check the port, but is this the way to achieve this?. If I dont wait I get an error.  **docker-compose.yml**  worker:  build: myapp/.  volumes:  - myapp/.:/usr/src/app:ro  links:  - rabbitmq  rabbitmq:  image: rabbitmq:3-management  **python hello sample (rabbit.py):**  import pika  import time  import socket  pingcounter = 0  isreachable = False  while isreachable is False and pingcounter < 5:  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  try:  s.connect(('rabbitmq', 5672))  isreachable = True  except socket.error as e:  time.sleep(2)  pingcounter += 1  s.close()  if isreachable:  connection = pika.BlockingConnection(pika.ConnectionParameters(  host="rabbitmq"))  channel = connection.channel()  channel.queue\_declare(queue='hello')  channel.basic\_publish(exchange='',  routing\_key='hello',  body='Hello World!')  print (" [x] Sent 'Hello World!'")  connection.close()  **Dockerfile for worker:**  FROM python:2-onbuild  RUN ["pip", "install", "pika"]  CMD ["python","rabbit.py"]  **Update Nov 2015**:  A shell script or waiting inside your program is maybe a possible solution. But after seeing this [Issue](https://github.com/docker/compose/issues/374) I am looking for a command or feature of docker/docker-compose itself.  They mention a solution for implementing a health check, which may be the best option. A open tcp connection does not mean your service is ready or may remain ready. In addition to that I need to change my entrypoint in my dockerfile.  So I am hoping for an answer with docker-compose on board commands, which will hopefully the case if they finish this issue.  **Update March 2016**  There is a [proposal](https://github.com/docker/docker/issues/21142) for providing a built-in way to determine if a container is "alive". So docker-compose can maybe make use of it in near future.  **Update June 2016**  It seems that the healthcheck will be [integrated](https://github.com/docker/docker/pull/23218) into docker in Version 1.12.0  **Update Januar 2017**  I found a docker-compose solution see: [Docker Compose wait for container X before starting Y](https://stackoverflow.com/questions/31746182/docker-compose-wait-for-container-x-before-starting-y/41854997#41854997) |

**🡪**For container start ordering use

depends\_on:

For waiting previous container start use script

entrypoint: ./wait-for-it.sh db:5432

This article will help you <https://docs.docker.com/compose/startup-order/>

**4.)** [How do I configure docker compose to expose ports correctly?](https://stackoverflow.com/questions/37770718/how-do-i-configure-docker-compose-to-expose-ports-correctly)

I'm using docker and docker compose to run a clojure and a node app, alongside postgres.

The project is contained in the following folder structure.

project/

-- app/

-- -- Dockerfile

-- frontend/

-- -- /Dockerfile

-- docker-compose.yml

The app/Dockerfile looks like so...

FROM clojure:latest

COPY . /usr/src/app

WORKDIR /usr/src/app

EXPOSE 9000

CMD ["lein", "run", "migrate", "&&","lein", "run"]

The frontend/Dockerfile looks like so ...

FROM node:5

COPY . /usr/src/app

WORKDIR /usr/src/app

RUN npm install

EXPOSE 8080

CMD ["npm", "start"]

And lastly the docker-compose.yml looks like...

frontend:

image: bradcypert/node

volumes:

- ./frontend:/usr/src/frontend

ports:

- "8080:8080"

backend:

image: bradcypert/clojure

volumes:

- ./app:/usr/src/backend

ports:

- "9000:9000"

links:

- postgres

postgres:

image: postgres

ports:

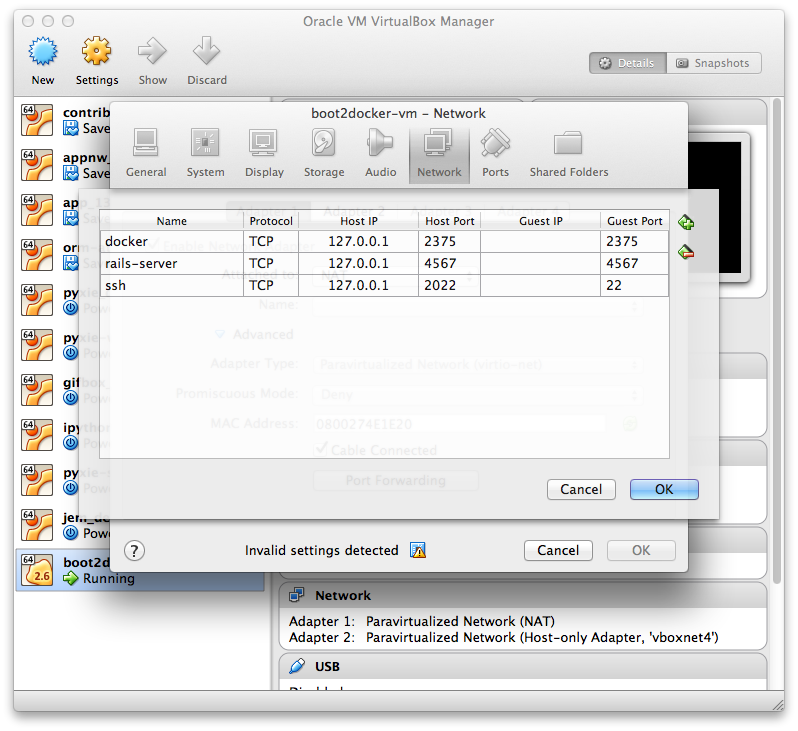
- "5432:5432"

backend is failing for a separate reason, but the frontend seems to be running successfully, that being said, I'm unable to hit localhost:8080 and see the app. What do I need to do make this happen?

Thanks in advance.

Just to clarify, the command being run is docker-compose up

**🡪**With boot2docker (on Mac or Windows), to access any port from localhost, you have to configure your VirtualBox VM in order to **port-forward** that port from the VM into the host.

Your port mappings are correct, but you still need to make visible to your host (Mac) the one port you want to access from localhost (your Mac). 

See for instance "[Using boot2docker to run Docker on a Mac or Windows](http://odewahn.github.io/docker-jumpstart/boot2docker.html)" from [**Andrew Odewahn**](https://github.com/odewahn):

That way, you don't have to find out what the IP of your machine is.  
(Which you can see with docker-machine ls followed by docker-machine ip <name>)

4.)[Docker Compose Mount Window Folder](https://stackoverflow.com/questions/35495639/docker-compose-mount-window-folder)

|  |  |  |
| --- | --- | --- |
|  | I am using docker toolbox in Windows and trying to mount a windows folder in a docker-compose.yml file like this:  nginx:  image: nginx:latest  container\_name: test\_server  ports:  - "80:80"  volumes:  - /sss:/c/data/www:ro  environment:  - VIRTUAL\_HOST=test.local  My objective is to Mount C:\data\www to boot2docker VM image which is already created by docker toolbox and then from there to the nginx container inside of it.  Unfortunately its not working, i get a folder sss inside boot2docker image but its empty without targeting to my windows data.  What I am doing wrong? Is there any better practice in order to use docker in windows while you are developing (so you need to share code between windows, docker vm (boot2docker) and docker containers)  🡪   |  | | --- | | Remember that with boot2docker, there are two layers of abstraction: a Linux virtual machine, then the Docker engine running in that VM. Docker Compose doesn't know anything about the world outside the VM where Docker is running, and can't, for example, configure VirtualBox shared folders. There is some discussion of this in the boot2docker documentation: [github.com/boot2docker/boot2docker#folder-sharing](https://github.com/boot2docker/boot2docker#folder-sharing) – [Kurt Raschke](https://stackoverflow.com/users/1068191/kurt-raschke) | |

🡪 My objective is to Mount C:\data\www to boot2docker VM image

From "[Manually sharing directory as docker volume mounting point](https://stackoverflow.com/a/35445801/6309)":

You need to:

* modify your VirtualBox VM (make sure it is stopped first):
* VBoxManage sharedfolder add <machine name/id> --name <mount\_name> --hostpath <host\_dir> --automount
* # in your case
* /c/Program\ Files/Oracle/VirtualBox/VBoxManage.exe sharedfolder add default --name www --hostpath 'C:\data\ww' --automount
* add an [automount to your boot2docker VM](https://stackoverflow.com/a/32030385/6309):
  + Edit/create (as root) /mnt/sda1/var/lib/boot2docker/bootlocal.sh, (sda1 may be different for you)
  + Add
  + mkdir -p <local\_dir>

mount -t vboxsf -o defaults,uid=`id -u docker`,gid=`id -g docker` <mount\_name> <local\_dir

🡪 in docker-compose.yml the volumes have nothing to do with windows but its to mount from linux vm to containers only?

Yes, they operate in the Linux world of boot2docker only.

5.) [Interactive shell using Docker Compose](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose)

Is there any way to start a interactive shell in a container using Docker Compose only? I've tried something like this, in my docker-compose.yml:

myapp:

image: alpine:latest

entrypoint: /bin/sh

When I start this container using docker-compose up it's exited immediately. Are there any flags I can add to the entrypoint command, or as and additional option to myapp, to start as interactive shell?

I know there are native docker command options to achieve this, just curious if it's possible using only Docker Compose, too.

🡪

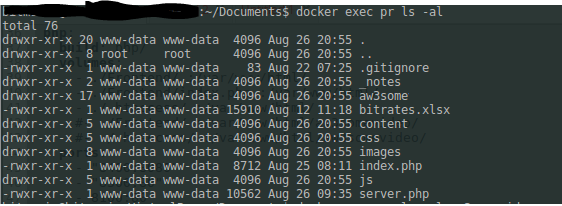
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | | |  |  | | --- | --- | |  |  | | Hmh, why not just start multiple shells? For example, a shell into a mysql container to work with mysql cli, and a shell into a backup container to run backup commands? – [drubb](https://stackoverflow.com/users/4042051/drubb) [Mar 27 '16 at 17:06](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment60129897_36249744) | | |  |  | | --- | --- | | 1 |  | | what about docker-compose run myapp ? – [ivoba](https://stackoverflow.com/users/541949/ivoba) [Mar 27 '16 at 17:07](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment60129930_36249744) | |

|  |  |
| --- | --- |
|  | Using docker-compose, I found the easiest way to do this is to do a docker ps -a (after starting my containers with docker-compose up) and get the ID of the container I want to have an interactive shell in (let's call it xyz123).  Then it's a simple matter to execute docker exec -ti xyz123 /bin/bash  and voila, an interactive shell. |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Not sure why this was down-voted - it's a great way to get debugging in case anything goes wrong, and I used this method with success, like, within a minute of reading this solution. – [ericmjl](https://stackoverflow.com/users/1274908/ericmjl) [Aug 31 at 17:30](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment78934242_43479221) | | |  |  | | --- | --- | |  |  | | @ericmjl Because it's a two step process where the question asked specifically about using docker-compose features, and was already stated in the other answer – [cricket\_007](https://stackoverflow.com/users/2308683/cricket-007) [Sep 2 at 4:34](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment78981471_43479221) | |
|  | The canonical way to get an interactive shell with docker-compose is to use: docker-compose run --rm myapp  You can set stdin\_open: true, tty: true, however that won't actually give you a proper shell with up, because logs are being streamed from all the containers.  You can also use  docker exec -ti <container name> /bin/bash  to get a shell on a running container. |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 4 |  | | Note you need to add --service-ports if you expose any ports(ie for a web server) – [epelc](https://stackoverflow.com/users/2252814/epelc) [Aug 21 '16 at 19:22](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment65484161_36265910) | | |  |  | | --- | --- | | 4 |  | | I've updated my answer to provide more information and add the --rm flag so that the container is removed. The answer by @lynx0123 is not correct. You will not get an interactive shell if you run docker-compose up. – [dnephin](https://stackoverflow.com/users/444646/dnephin) [Sep 1 '16 at 16:17](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment65887539_36265910) | | |  |  | | --- | --- | | 1 |  | | To get this to work using Docker for Windows, I needed docker-compose v. 1.9.0 (see Github [issue](https://github.com/docker/compose/issues/3194) and [PR](https://github.com/docker/compose/pull/3980)). As of 12/19/16, this only ships with beta versions of Docker for Windows. Then docker-compose runworks. You'll also want to add command: /bin/bash to docker-compose.yml. – [Joseph238](https://stackoverflow.com/users/5016547/joseph238) [Dec 20 '16 at 0:14](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment69665625_36265910) | | |  |  | | --- | --- | | 1 |  | | This should be the top answer. This is what I was looking for when I came here. – [mkasberg](https://stackoverflow.com/users/1263211/mkasberg) [Jan 25 at 18:38](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment70902075_36265910) | | |  |  | | --- | --- | | 1 |  | | docker-compose up -d && docker attach <container\_name> – [Aaron McMillin](https://stackoverflow.com/users/401636/aaron-mcmillin) [Apr 24 at 2:34](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment74207567_36265910) | |

|  |  |
| --- | --- |
|  | stdin\_open: true  tty: true  The first corresponds to -i in docker run and the second to -t. |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | that stdin\_open is the missing link, for simply providing me the expected behavior when I attach to one of my containers that is already running a shell. – [charneykaye](https://stackoverflow.com/users/1335245/charneykaye) [Dec 7 '16 at 20:50](https://stackoverflow.com/questions/36249744/interactive-shell-using-docker-compose#comment69261501_39150040) | |

5.) [Docker composer copy files](https://stackoverflow.com/questions/39176561/docker-composer-copy-files)

|  |  |
| --- | --- |
|  | I have a Dockerfile where i copy an existing directory(with content) to the container which works fine:  **Dockerfile**  FROM php:7.0-apache  COPY Frontend/ /var/www/html/aw3somevideo/  COPY Frontend/ /var/www/html/  RUN ls -al /var/www/html  RUN chown -R www-data:www-data /var/www/html  RUN chmod -R 755 /var/www/html |



But when i use a docker-compose.yml file there is only the directory àw3somevideo and inside àw3somevideo there is nothing.

**docker-compose.yml**:

php:

build: php/

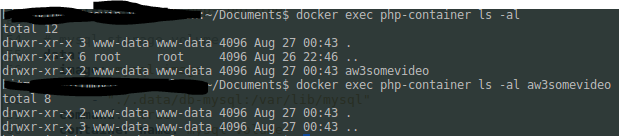
volumes:

- ./Frontend/ :/var/www/html/

- ./Frontend/index.php :/var/www/html/

ports:

- 8100:80



Maybe i do not understand the function of volumes and if that's the case please tell me how to copy my existing files to the container via a docker-compose.yml file.

-🡪The behavior has changed since I wrote the original answer. It is now consistent whether the right hand side specifies a named volume like myvolume or a path on the host like /var/lib/myapp. For instance

volumes:

- /dir/on/host:/var/www/html

if /dir/on/host doesn't exists, it is created on the host and the empty content is mounted in the container at /var/www/html. Whatever was in /var/www/html before is inaccessible.

---- old answer -----------

The volumes: section in your docker-compose overwrites whatever is in the /var/www/htmldirectory.

There are two mains situations:

1. The volume exists

In that case, the content of the volume overshadows whatever is in the dst directory.

Eg:

volumes:

- /dir/on/host:/var/www/html

1. The volume doesn't exist

If myvolume doesn't exist (a named volume for instance), the content of /var/www/html will be copied to volume the first time around

volumes:

- myvolume:/var/www/html

In case 2, if you try to mount the same volume again on some container, it will follow case 1.

volumes:

- myvolume:/var/www/html

In that case (assuming myvolume was already created), the content of /var/ww/html will be overwritten (shadowed) by whatever is in myvolume.

The official doc goes into more details <https://docs.docker.com/compose/compose-file/#/volumes-volume-driver>

🡪Is it possible, that with recent docker versions the content of var/www/html/ is not copied anymore to /dir/on/host even when it doesn't exist? Is there a way to force-copy the data from within the docker-container to the host when atarting a container the first time? – Tarator Apr 21 at 8:10

@Tarator yes indeed, the right hand side is not copied to the host anymore. I'll update the answer. As for a way to copy on container start, you can override the startup command with something like this docker run -v /dir/on/host:/hostdir php sh -c "cp -rp /var/www/html/\* /hostdir && exec myapp". Don't forget to use exec to invoke the final command so that it is assigned PID1. That will make sure that myapp receives termination signals (Ctrl-C for instance). – Alkaline Apr 24 at 10:07

7.) [chown docker volumes on host (possibly through docker-compose)](https://stackoverflow.com/questions/36312699/chown-docker-volumes-on-host-possibly-through-docker-compose)

8.) [docker-compose persistent data on host and container](https://stackoverflow.com/questions/44575369/docker-compose-persistent-data-on-host-and-container)

I have a problem with volumes in docker-compose yml 3.0+

So I know that a volume behaves like a mount.. But I have set up a wiki and when i set a volume in the docker-compose, the data on the container will be removed (hidden)

So how can I save data from my container to my host first and the next time I start the container, it will just overrides the data I saved.

So the current situation is: I start with "docker-compose up --build" and a volume is created (empty) and will be copied to the container.. Everything in that folder on the container is deleted as a result

docker-compose.yml

version: '3.1'

services:

doku-wiki:

build: .

ports:

- '4000:80'

Dockerfile

FROM php:7.1-apache

COPY dokuwiki-stable /var/www/html/

COPY entrypoint.sh /entrypoint.sh

RUN chmod 777 /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]

EXPOSE 80

----Can you clarify the question? Are you looking for a way to "copy existing container data to a volume on container start"? –

Yes, the first time i start the container via docker-compose up it should copy the existing container data to a volume. After that the container should use the date from the volume without overwriting everything. For example I want to backup the folder "/data/pages/" But on the first start the backup volume will be empty.. and it will copy nothing to /data/pages/ but deletes the existing content of the container – Bvz Jun 15 at 21:20

I do not see any volume definition in your docker-compose.yml file. Where does the volume come into play?

-🡪It sounds like you are using a host volume where you map a host directory into the container. When you do this, anything at that location inside your image will not be visible, only the files as they exist on the host.

If you want to have a copy of the files from inside your image to initialize the volume, you have two options:

1. Switched to a named volume. Docker will automatically initialize these to the contents of the image, including any permissions. If you don't require direct access to the files from outside of docker, this is the preferred solution.
2. Change your image entrypoint and the location where you store your files in the image.

On the second option, if you want /data to be a volume for your application, you could have an entrypoint.sh that does:

#!/bin/sh

if [ ! -d "/data" ]; then

ln -s /data\_save /data

elif [ -z "$(ls -A /data)" ]; then

cp -a /data\_save/. /data/

fi

exec "$@"

Your image would need to save all the initial files to /data\_save instead of /data. Then if the directory is empty it would do a copy of /data\_save to your volume /data. If the volume wasn't mapped at all, then it just creates a symlink from /data to /data\_save. The last line runs the CMDfrom your Dockerfile or docker run cli as if the entrypoint wasn't ever there. The added lines to your Dockerfile would look like:

COPY entrypoint.sh /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]

9.) [Docker-Compose not able to copy haproxy.cfg](https://stackoverflow.com/questions/45573831/docker-compose-not-able-to-copy-haproxy-cfg)

My problem is that I have a docker-compose.yml file and an haproxy.cfg file and I want docker-compose to copy the haproxy.cfg file to the docker container. As per the post [Docker composer copy files](https://stackoverflow.com/questions/39176561/docker-composer-copy-files) I can use volumes to do it but in my case I'm getting the below error. Can anybody help me achieve this.

Below is the code and everything

**docker-compose.yml**

version: "3.3"

services:

###After all services are up, we are initializing the gateway

gateway:

container\_name: gateway-haproxy

image: haproxy

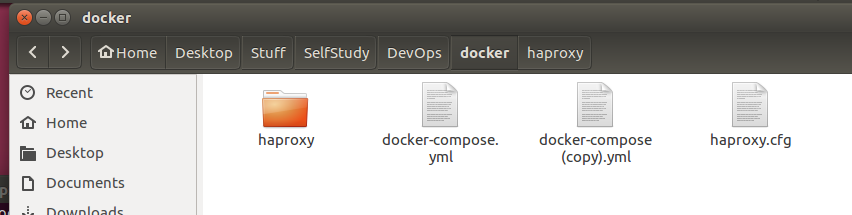
volumes:

- .:/usr/local/etc/haproxy

ports:

- 80:80

network\_mode: "host"

**Folder Structure**

**Command output**

root@ubuntu:/home/karunesh/Desktop/Stuff/SelfStudy/DevOps/docker# docker-compose up

Creating gateway-haproxy ...

Creating gateway-haproxy ... done

Attaching to gateway-haproxy

gateway-haproxy | <7>haproxy-systemd-wrapper: executing /usr/local/sbin/haproxy -p /run/haproxy.pid -f /usr/local/etc/haproxy/haproxy.cfg -Ds

gateway-haproxy | [ALERT] 219/163305 (6) : [/usr/local/sbin/haproxy.main()] No enabled listener found (check for 'bind' directives) ! Exiting.

gateway-haproxy | <5>haproxy-systemd-wrapper: exit, haproxy RC=1

gateway-haproxy exited with code 1

**-🡪**You can’t copy a file into a container using docker-compose. This has to be done inside of the Dockerfile itself – Serey Aug 8 at

I dont have the file...i am supposed to use official haproxy image...is there any way I can do it... – utkarsh31 Aug 8 at 16:55

@Serey I've done it before, see my answer below – MatTheWhale Aug 8 at 17:54

1

@MatTheWhale yeah, wasn't 100% sure of what he wanted because this is only useful when you mount the volume. If he ever wanted to work outside of his environment it would be different if he wanted to build. – Serey Aug 8 at 18:56

🡪

volumes:

- ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro

Instead of mounting the whole directory, this will only mount haproxy.cfg. The ro is an abbreviation for read-only, and its usage guarantees the container won't modify it after it gets mounted.

🡪In order to add additional files to the container, you have to build on top of the existing image from [haproxy](https://github.com/docker-library/haproxy).

For example, your Dockerfile should look like this:

FROM haproxy:latest

COPY haproxy.cfg /usr/local/etc/haproxy/haproxy.cfg

Then you can update your docker compose file accordingly.

If you plan on using this for local development, just mount the file(s), see @MatTheWhale's answer

See more at the [official haproxy Docker page](https://hub.docker.com/_/haproxy/)

-->Thanks a lot for this...but is there anyway that we can do it without using the docker file...let us assume I have the docker image present and I have made some changes in the haproxy.cfg. Instead of updating an image can I not directly copy the local haproxy.cfg to the location in the image... – utkarsh31 Aug 8 at 17:45

10.) [How to substitute variable value in “docker run” command](https://stackoverflow.com/questions/36876613/how-to-substitute-variable-value-in-docker-run-command)

I am using a bash script and trying to assign a fingerprint value as below

export FINGERPRINT=D0:19:C5:80:42:66:56:AC:6F

docker run --rm -i -v /var/run/docker.sock:/var/run/docker.sock --name ucp docker/ucp join --replica --fingerprint $FINGERPRINT

However, bash is not at all substituting the value of $FINGERPRINT

🡪Try this:

export FINGERPRINT=D0:19:C5:80:42:66:56:AC:6F

echo `docker run --rm -i -v /var/run/docker.sock:/var/run/docker.sock --name ucp docker/ucp join --replica --fingerprint $FINGERPRINT`

🡪stackoverflow.com/questions/36877186/… – Benjamin W. Apr 26 '16 at 23:00

What do you get if you do these after the first line: echo $FINGERPRINT, and echo docker run --rm -i -v /var/run/docker.sock:/var/run/docker.sock --name ucp docker/ucp join --replica --fingerprint $FINGERPRINT – webb Apr 26 '16 at 23:37

This should work in bash, however are you sure you are using bash? – ateles Apr 27 '16 at 7:03

11.)[Docker Machine: No space left on device](https://stackoverflow.com/questions/31909979/docker-machine-no-space-left-on-device)

I'm trying to set up Docker Machine with Docker Compose.

**Scenario 1 (without Docker Machine)**  
If I run docker-compose up -d without Docker Machine, it creates my 3 linked containers as intented (nginx+mongodb+nodejs).

**Scenario 2 (with Docker Machine)**  
Then I create a VM using Docker Machine and tell Docker to talk to that machine with eval $(docker-machine env streambacker-dev).

At this point, if I ssh to my docker machine and run df -h, I get:

If I then run docker-compose up -d, I get a ["no space left on device" error](https://imgur.com/ROS74NF) while downloading the last container.

"tmpfs" seems to be indeed a bit full after that:

Checking the **--virtualbox-disk-size** option shows that it defaults to 20000 MB, which I think is what we can see as "/dev/sda1" on both pictures. So why are containers filling up "tmpfs" n and what exactly is "tmpfs"? Is is a temporary download directory? **How can I create more space for my containers?**

Thanks!

For information, I'm using *Docker Machine 0.4.0-rc2* and *Docker Compose 1.3.2*.

-🡪

|  |
| --- |
| The [tmpfs](https://en.wikipedia.org/wiki/Tmpfs) has nothing to do with --virtualbox-disk-size. It is a filesystem (like a RAM disk) mounted in memory and nothing there is accessing your disk. – [h3nrik](https://stackoverflow.com/users/2706422/h3nrik) [Aug 10 '15 at 7:43](https://stackoverflow.com/questions/31909979/docker-machine-no-space-left-on-device#comment51741244_31909979) |

🡪Like said above, the tmpfs has nothing to do with --virtualbox-disk-size. It seems like boot2docker mounts tmpfs into memory, so you need to dedicate more memory to your virtualbox vm. You can do it by specifying the --virtualbox-memory parameter.

--virtualbox-memory "1024"

Size of memory for host in MB [$VIRTUALBOX\_MEMORY\_SIZE]

Defaults:

$ docker-machine create --driver virtualbox testA

Creating VirtualBox VM...

Creating SSH key...

Starting VirtualBox VM...

Starting VM...

$ docker-machine ssh testA

--Boot2Docker version 1.8.1, build master : 7f12e95 - Thu Aug 13 03:24:56 UTC 2015

Docker version 1.8.1, build d12ea79

docker@testA:~$ df -h /

Filesystem Size Used Available Use% Mounted on

tmpfs 896.6M 112.7M 783.9M 13% /

With --virtualbox-memory set to 8096

$ docker-machine create --driver virtualbox --virtualbox-memory 8096 testB

Creating VirtualBox VM...

Creating SSH key...

Starting VirtualBox VM...

Starting VM...

$ docker-machine ssh testB

Boot2Docker version 1.8.1, build master : 7f12e95 - Thu Aug 13 03:24:56 UTC 2015

Docker version 1.8.1, build d12ea79

docker@testB:~$ df -h /

Filesystem Size Used Available Use% Mounted on

Tmpfs 6.9G 112.4M 6.8G 2% /

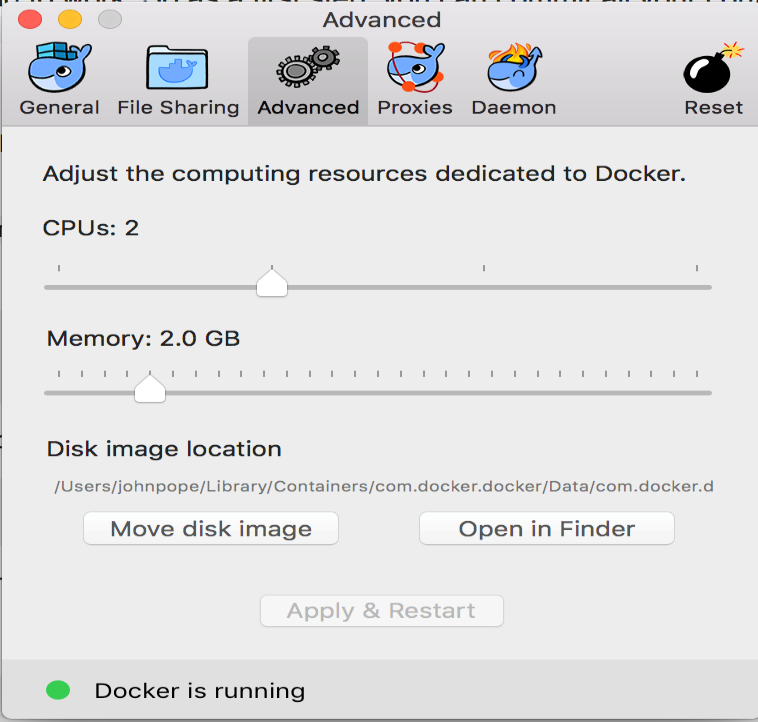
-🡪not very intuitive, but this worked.

Not intuitive indeed! However, this seems to be the problem with this and fixed it for me.

would seem that boot2docker mounts tmpfs in memory..

Fixed the problem for me too, I don't understand how but, thank you!

🡪 docker osx / I was able to press a button [**Move Disk Image**] and it successfully moved the Docker.qcow2 (presumably containing containers / images)

initially - when machines started - I was still getting a **No space left on device error** but it resolved shortly after.

🡪If you are using Docker Community Edition:

docker system prune

If you are using boot2docker (docker-machine) clear the volumes that are orphaned:

docker volume rm $(docker volume ls -qf dangling=true)

Clear unused images:

docker rmi $(docker images -q -f "dangling=true")

-🡪A. REMOVE UNUSED IMAGES

using the docker rm or docker rmi commands you can remove the images that you don't need. Actually exist an image that helps in this task (martin/docker-cleanup-volumes). The basis is to start selectig from your images and containers list:

docker ps -a -s

B. MODIFY THE DOCKER JSON DESCRIPTOR

it's mentioned in some forums. The idea is to increment the descriptor located in ~/.docker/machine/machines/default/config.json . The param seems to be DiskSize but i don't know if it works in other OSs (not in windows).

C. LINUX RESIZE:

in Windows OS, docker machine or boot2docker is in fact a virtualbox vm, then you can follow the procedure to resize the disk. Take care to backup the files. The general procedure is to make a resize in virtualbox and then use an utilitary called gpartd to modify the space perceived by linux in its partitions. There are some links to do this procedure referenced below:

* [resize vbox disk](http://derekmolloy.ie/resize-a-virtualbox-disk/)
* [move space](https://ask.fedoraproject.org/en/question/55867/how-can-i-move-space-from-one-partition-to-another/)
* [vbox forum](https://forums.virtualbox.org/viewtopic.php?f=35&t=50661)

D. RECREATE THE DOCKER-MACHINE / BOOT2DOCKER

The idea is recreate the default docker-machine. The following commands can illustrate you. Note that as you are re-creating the boot2docker, you will lost the previous downloaded docker images.

docker-machine rm default

docker-machine create --driver virtualbox --virtualbox-disk-size "100100" default

docker-machine env default

then you can go to virtual box and see the boot2docker space with the command "df -h"

Thanks. I was working on Windows. I made it by recreating the docker-machine. – KinoP Jun 20 '16 at 2:47

Helped the D: (recreating docker machine) – StanislavL Sep 4 at 13:37

Thank you, recreating the docker-machine worked for me (I'm on windows

-🡪

I had the same error ([ERROR] InnoDB: Error number 28 means 'No space left on device') and solve it this way:

1 . Delete the orphaned volumes in Docker, you can use the built-in docker volume command. The built-in command also deletes any directory in /var/lib/docker/volumes that is not a volume so make sure you didn't put anything in there you want to save.

Warning be very careful with this if you have some data you want to keep

**Cleanup:**

$ docker volume rm $(docker volume ls -qf dangling=true)

Additional commands:

List dangling volumes:

$ docker volume ls -qf dangling=true

List all volumes:

$ docker volume ls

2 . Also consider removing all the unused Images.

First get rid of the <none> images (those are sometimes generated while building an image and if for any reason the image building was interrupted, they stay there).

here's a nice script I use to remove them

docker rmi $(docker images | grep "^<none>" | awk "{print $3}")

Then if you are using Docker Compose to build Images locally for every project. You will end up with a lot of images usually named like your folder (example if your project folder named Hello, you will find images name Hello\_blablabla). so also consider removing all these images

you can edit the above script to remove them or remove them manually with

docker rmi {image-name}

12.) [Where is a log file with logs from a container?](https://stackoverflow.com/questions/33017329/where-is-a-log-file-with-logs-from-a-container)

|  |  |
| --- | --- |
|  | I am running several containers using docker-compose. I can see application logs with command docker-compose logs. However I would like to access raw log file to send it somewhere for example? Where is it located? I guess it's separate log per each container (inside container?) but where I can find it? |

🡪

|  |  |
| --- | --- |
|  | A container's logs can be found in :  /var/lib/docker/containers/<container id>/<container id>-json.log  (if you use the default log format which is json) |

-🡪You can [docker inspect](https://docs.docker.com/engine/reference/commandline/inspect/) each container to see where their logs are:

docker inspect --format='{{.LogPath}}' $INSTANCE\_ID

Of interest:

* [issue 1083](https://github.com/docker/compose/issues/1083#issuecomment-141936600):

Docker 1.8 and docker-compose 1.4 there is already exists a method to limit log size using [docker compose log driver](https://docs.docker.com/compose/yml/#log-driver) and log-opt max-size:

log\_driver: "json-file"

log\_opt:

max-size: "100k"

max-file: "20"

* [issue 1866](https://github.com/docker/compose/issues/1866): logs doesn't exit if the container is already stopped

13.) [Using docker-compose with CI - how to deal with exit codes and daemonized linked containers?](https://stackoverflow.com/questions/29568352/using-docker-compose-with-ci-how-to-deal-with-exit-codes-and-daemonized-linked)

Right now our Jenkins agents generate a docker-compose.yml for each of our Rails projects and then run docker-compose up. The docker-compose.yml has a main "web" container that has rbenv and all of our other Rails dependencies inside. It is linked to a DB container that contains the test Postgres DB.

|  |  |  |
| --- | --- | --- |
|  | The problem comes when we need to actually run the tests and generate exit codes. Our CI server will only deploy if the test script returns exit 0, but docker-compose always returns 0, even if one of the container commands fail.  The other issue is that the DB container runs indefinitely, even after the web container is done running the tests, so docker-compose up never returns.  Is there a way we can use docker-compose for this process? We would need to be able to run the containers, but exit after the web container is complete and return it's exit code. Right now we are stuck manually using docker to spin up the DB container and run the web container with the --link option.  -🡪--exit-code-from SERVICE and --abort-on-container-exit don't work in scenarios where you need to run all containers to completion, but fail if one of them exited early. An example might be if running 2 test suits in concurrently in different containers.  With @spenthil's suggestion, you can wrap docker-compose in a script that will fail if any containers do.  #!/bin/bash  set -e  # Wrap docker-compose and return a non-zero exit code if any containers failed.  docker-compose "$@"  exit $(docker-compose -f docker-compose.ci.build.yml ps -q | tr -d '[:space:]' |  xargs docker inspect -f '{{ .State.ExitCode }}' | grep -v 0 | wc -l | tr -d '[:space:]')  Then on your CI server simply change docker-compose up to ./docker-compose.sh up.  🡪Since version 1.12.0, you can use the --exit-code-from option.  From [documentation](https://docs.docker.com/compose/reference/up/):  --exit-code-from SERVICE  Return the exit code of the selected service container. Implies --abort-on-container-exit.  🡪That should be the right way of doing it if you are using docker-compose 1.12.0 and above. Maybe it is your case too. An example could be: docker-compose up --exit-code-from test-unit. Note that it didn't work for me until I added a set -e at the beginning of my script. – Adrian Antunez Jul 27 at 9:23  --exit-code-from doesn't work with -d though. It will throw these errors: using --exit-code-from implies --abort-on-container-exit and --abort-on-container-exit and -d cannot be combined. – ericat Aug 29 at 16:06    I was able to get this working on Travis CI: travis-ci.org/coyote-team/coyote/builds/274582053 here's the travis.yml: github.com/coyote-team/coyote/blob/master/.travis.yml#L12 – subelsky  🡪 Use docker wait to get the exit code:  $ docker-compose -p foo up -d  $ ret=$(docker wait foo\_bar\_1)  foo is the "project name". In the example above, I specified it explicitly, but if you don't supply it, it's the directory name. bar is the name you give to the system under test in your docker-compose.yml.  Note that docker logs -f does the right thing, too, exiting when the container stops. So you can put  $ docker logs -f foo\_bar\_1  between the docker-compose up and the docker wait so you can watch your tests run.  -🡪If you're willing to use docker-compose run to manually kick off your tests, adding the --rm flag, oddly enough, causes Compose to accurately reflect your command's exit status.  Here's my example:  $ docker-compose -v  docker-compose version 1.7.0, build 0d7bf73  $ (docker-compose run kpi false) || echo 'Test failed!' # False negative.  $ (docker-compose run --rm kpi false) || echo 'Test failed!' # True positive.  Test failed!  $ (docker-compose run --rm kpi true) || echo 'Test failed!' # True negative.  -🡪docker-compose run is the simple way to get the exit statuses you desire. For example:  $ cat docker-compose.yml  roit:  image: busybox  command: 'true'  naw:  image: busybox  command: 'false'  $ docker-compose run --rm roit; echo $?  Removing test\_roit\_run\_1...  0  $ docker-compose run --rm naw; echo $?  Removing test\_naw\_run\_1...  1  Alternatively, you do have the option to [inspect](http://docs.docker.com/reference/commandline/cli/#inspect) the dead containers. You can use the -f flag to get just the exit status.  $ docker-compose up  Creating test\_naw\_1...  Creating test\_roit\_1...  Attaching to test\_roit\_1  test\_roit\_1 exited with code 0  Gracefully stopping... (press Ctrl+C again to force)  $ docker-compose ps -q | xargs docker inspect -f '{{ .Name }} exited with status {{ .State.ExitCode }}'  /test\_naw\_1 exited with status 1  /test\_roit\_1 exited with status 0  As for the db container that never returns, if you use docker-compose up then you will need to sigkill that container; that's probably not what you want. Instead, you can use docker-compose up -d to run your containers daemonized, and manually kill the containers when your test is complete. docker-compose run *should* run linked containers for you, but I have heard chatter on SO about a bug preventing that from working as intended right now.  -🡪The problem with docker run is that it does not give any output when run with -T, and we want the output so we can inspect failed builds. – Logan Serman Apr 10 '15 at 19:45  @LoganSerman you can inspect the output with docker-compose logs – kojiro Apr 10 '15 at 19:47  Is there a way to constantly pipe those logs to STDOUT during the run so we can see it while the CI build is in progress? – Logan Serman Apr 10 '15 at 19:54  I guess I don't understand why you are running with -T – kojiro Apr 10 '15 at 20:21  Some of the commands we run inside of the container to run tests have the potential to ask for input, we want to run with -T to avoid this. Rbenv for example asks if you want to reinstall a Ruby version if it already exists. – Logan Serman Apr 15 '15 at 14:42  -🡪Building on kojiro's answer:  docker-compose ps -q | xargs docker inspect -f '{{ .State.ExitCode }}' | grep -v 0 | wc -l | tr -d ' '   1. get container IDs 2. get last runs exit code for each container ID 3. only non-0 status codes 4. count number of non-0 status codes 5. trim out white space   Returns how many non-0 exit codes were returned. Would be 0 if everything exited with code 0.  🡪   |  | | --- | | You can also use the non-quiet output from docker-compose ps, for example: docker-compose ps | grep -c "Exit 1" will give you the count where "Exit 1" is matched in the display from docker-compose ps (which provides a pretty-printed summary table of results). The exit codes are listed in the "State" column.– [eharik](https://stackoverflow.com/users/1721762/eharik) [Nov 3 '15 at 19:47](https://stackoverflow.com/questions/29568352/using-docker-compose-with-ci-how-to-deal-with-exit-codes-and-daemonized-linked#comment54799046_33291554) |   [docker-rails](https://github.com/alienfast/docker-rails) allows you to specify which container's error code is returned to the main process, so you CI server can determine the result. It is a great solution for CI and development for rails with docker.  For example  exit\_code: web  in your docker-rails.yml will yield the web containers exit code as a result of the command docker-rails ci test. docker-rails.yml is just a meta wrapper around the standard docker-compose.yml that gives you the potential to inherit/reuse the same base config for different environments i.e. development vs test vs parallel\_tests. |

14.) [Can't connect to docker from docker-compose](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)

|  |  |
| --- | --- |
|  | [I installed docker-machine 0.1.0 and docker-compose 1.1.0 on Mac OS 10.8.5. Docker-machine is running normally and able to connect by docker-machine ssh.](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [$ docker-machine ls](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [NAME ACTIVE DRIVER STATE URL SWARM](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [dev \* virtualbox Running tcp://192.168.99.100:2376](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [However can't connect from docker-compose.](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [$ docker-compose up](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [Couldn't connect to Docker daemon at http+unix://var/run/docker.sock - is it running?](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [If it's at a non-standard location, specify the URL with the DOCKER\_HOST environment variable.](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [My Dockerfile and docker-compose.yml is here.](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [Dockerfile](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [FROM centos:centos7](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [DOCKER\_HOST tcp://192.168.99.100:2376](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [docker-compose.yml](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [web:](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [build: .](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose)  [Why can't connect? Any ideas?](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose) |

-🡪

|  |
| --- |
| Does docker-machine run as root? I just came here with the same problem on Ubuntu and it was the permissions on the socket - sudo did the trick. – [SLD](https://stackoverflow.com/users/1284805/sld) [Mar 17 '15 at 14:49](https://stackoverflow.com/questions/29101043/cant-connect-to-docker-from-docker-compose#comment46434422_29101043) |
| For those on Linux, you might [just need to be added to the docker group](http://stackoverflow.com/a/33596140/434961). |

**-🡪**The Docker machine is running. But you need to export some environment to connect to the Docker machine. By default, the docker CLI client is trying to communicate to the daemon using http+unix://var/run/docker.sock (as shown in the error message).

Export the correct environment variables using eval $(docker-machine env dev) and then try again. You can also just run docker-machine env dev to see the environment variables it will export. Notice that one of them is DOCKER\_HOST, just as the error message suggests you may need to set.

-- My docker-machine name is dev, so I executed $(docker-machine env dev) command did the trick. Thanks for your help! – DIGITALSQUAD Mar

Yep, you are correct. I updated the answer to reflect this. – Andy Shinn Dec 11 '15 at 2:47

You can put that command (eval "$(docker-machine env default)") in your bin/postactivate to ensure it's always up-to-date. – Dannid

**-🡪**Simple solution for me: sudo docker-compose up

UPDATE 2016-3-14: At some point in the docker install process (or docker-compose ?) there is a suggestion and example to add your username to the "docker" group. This allows you to avoid needing "sudo" before all docker commands, like so:

~ > docker run -it ubuntu /bin/bash

root@665d1ea76b8d:/# date

Mon Mar 14 23:43:36 UTC 2016

root@665d1ea76b8d:/# exit

exit

~ >

Look carefully at the output of the install commands (both docker & the 2nd install for docker-compose) and you'll find the necessary step. It is also documented here:<https://subosito.com/posts/docker-tips/>

**Sudo? No!**

Tired of typing sudo docker everytime you issue a command? Yeah, there is a way for dealing with that. Although naturally docker is require a root user, we can give a root-equivalent group for docker operations.

You can create a group called docker, then add desired user to that group. After restarting docker service, the user will no need to type sudo each time do docker operations. How it looks like on a shell commands? as a root, here you go:

> sudo groupadd docker

> sudo gpasswd -a username docker

> sudo service docker restart

**-🡪**Anyone checked log ?

In my case error message in /var/log/upstart/docker.log was:

Listening for HTTP on unix (/var/run/docker.sock)

[graphdriver] using prior storage driver "aufs"

Running modprobe bridge nf\_nat failed with message: , error: exit status 1

Error starting daemon: Error initializing network controller: Error creating default "bridge" network: can't find an address range for interface "docker0"

Worth to mentioned I had vpn turned on, so:

$ sudo service openvpn stop

$ sudo service docker restart

$ docker-compose up|start

$ sudo service openvpn start was the solution.

# 15.) [Should I use docker-compose start up or run?](https://stackoverflow.com/questions/33066528/should-i-use-docker-compose-start-up-or-run)

Is there a reason to use run to start up a docker-compose.yml file or should you just use up?

I understand that run can start up a specific container, but I am referring to the case where you use it with out specifying a container so that it starts up all of your yml containers.

**-🡪**As mentioned in [docker-compose run](https://github.com/docker/compose/blob/17682c58db8986f1d1492559d72b3960b86f5f05/docs/reference/run.md):

**The command passed by run overrides the command defined in the service configuration**.  
For example, if the web service configuration is started with bash, then docker-compose run web python app.py overrides it with python app.py.

The second difference is the docker-compose run command **does not create any of the ports specified in the service configuration**.  
This prevents the port collisions with already open ports. If you do want the service's ports created and mapped to the host, specify the --service-ports flag:

$ docker-compose run --service-ports web python manage.py shell

So unless you have those specific needs (overriding a command or running only one container on different ports), docker-compose up (even for one container) is enough.

Can you help explain why or when you would not want the ports to be created? That is why or when they might conflict with already open ports

Simply because docker-compose run is made to run one-off commands for your services.  
That means that, if you already did a docker-compose up, all your containers are already running on their specified ports from docker-compose.yml.  
Doing a docker-compose run at this stage (to execute a one-off command), if it was respecting the same port, would fail immediately. Hence the default non-creation of those ports.

Another use case (in [Compose environment variables reference](https://github.com/docker/compose/blob/d577cd6877f9f8480f13cdbba3f2cf17e9c3d5eb/docs/env.md)):

To see what environment variables are available to a service, run docker-compose run SERVICE env.

**-🡪**I'd like to point out that if you're using Python with the pdb debugger:

import pdb; pdb.set\_trace()

It will not drop to the shell if you execute your script using:

docker-compose up

However if you use run, it will drop down to the debugger as expected:

docker-compose run

# 15.) [What are the differences between Kubernetes Pods and Docker Compose(s) (Composures?)](https://stackoverflow.com/questions/33946144/what-are-the-differences-between-kubernetes-pods-and-docker-composes-composur)

Both Kubernetes Pods and the results of Docker Compose scripts (henceforth: "Compositions") appear to result in clusters of virtual computers.

The computers in the clusters can all be configured to talk to each other so you can write a single script that mirrors your entire end-to-end production config. A single script allows you to deploy that cluster on any container-host.

Given the similarities between the two systems, I'm struggling to understand what the differences are between the two.

Why would I choose one over the other? Are they mutually exclusive systems or can I run compositions in kubernetes.

Are there any critical considerations that need to be accounted for when designing for a container system? If I am designing the architecture for a site *today* and would *like* to try and build a container-based system. What are the highest priority things I should design for? (as compared to building on a single machine system)

**-🡪**

|  |  |
| --- | --- |
|  | [docker compose](https://github.com/docker/compose) is just a way to declare the container you have to start: , unless it launches swarm master and swarm nodes, but that is [docker swarm](https://docs.docker.com/swarm/)) Update July 2016, 7 months later: docker 1.12 blurs the lines and [includes a "swarm mode"](https://docs.docker.com/engine/swarm/).  It is vastly different from [kubernetes](http://kubernetes.io/), a google tool to manage thousands of containers groups as Pod, over tens or hundreds of machines.  A [Kubernetes Pod](http://kubernetes.io/v1.0/docs/user-guide/pods.html) would [be closer from a docker swarm](http://googlecloudplatform.blogspot.fr/2015/01/everything-you-wanted-to-know-about-Kubernetes-but-were-afraid-to-ask.html):  Imagine individual Docker containers as packing boxes. The boxes that need to stay together because they need to go to the same location or have an affinity to each other are loaded into shipping containers. In this analogy, the packing boxes are Docker containers, and the shipping containers are Kubernetes pods.  As [commented below](https://stackoverflow.com/questions/33946144/what-are-the-differences-between-kubernetes-pods-and-docker-composes-composur/33946256#comment64413899_33946256) by [ealeon](https://stackoverflow.com/users/1686628/ealeon):  I think pod is equivalent to compose except that kubernetes can orchestrated pods, whereas there is nothing orchestrating compose unless it is used with swarm like you've mentioned.  You can [launch kubernetes commands with docker-compose by the way](http://sebgoa.blogspot.fr/2015/04/1-command-to-kubernetes-with-docker.html).    In terms of how Kubernetes differs from other container management systems out there, such as Swarm, Kubernetes is the third iteration of cluster managers that Google has developed.  You can hear more about kubernetes in the [episode #3 of Google Cloud Platform Podcast](https://www.gcppodcast.com/post/episode-3-kubernetes-and-google-container-engine/).  While it is true both can create a multi-container application, a Pod also serves as a unit of deployment and horizontal scaling/replication, which docker compose does not provide. Plus, you don't create a pod directly, but use controllers (like replication controllers).  POD lives within a larger platform which offers Co-location (co-scheduling), fate sharing, coordinated replication, resource sharing, and dependency management. Docker-compose lives... on its own, with its docker-compose.yml file |

# 16.) [Docker 1.10 access a container by it's hostname from a host machine](https://stackoverflow.com/questions/35828487/docker-1-10-access-a-container-by-its-hostname-from-a-host-machine)

|  |  |
| --- | --- |
|  | I have the Docker version 1.10 with embedded DNS service.  I have created two service containers in my docker-compose file. They are reachable each other by hostname and by IP, but when I would like reach one of them from the host machine, it doesn't work, it works only with IP but not with hostname.  So, is it possible to access a docker container from the host machine by it's hostname in the Docker 1.10, please?  **Update:**  docker-compose.yml  version: '2'  services:  service\_a:  image: nginx  container\_name: docker\_a  ports:  - 8080:80  service\_b:  image: nginx  container\_name: docker\_b  ports:  - 8081:80  then I start it by command: docker-compose up --force-recreate  when I run:   * docker exec -i -t docker\_a ping -c4 docker\_b - it works * docker exec -i -t docker\_b ping -c4 docker\_a - it works * ping 172.19.0.2 - it works (172.19.0.2 is docker\_b's ip) * ping docker\_a - **fails**   The result of the docker network inspect test\_default is  [  {  "Name": "test\_default",  "Id": "f6436ef4a2cd4c09ffdee82b0d0b47f96dd5aee3e1bde068376dd26f81e79712",  "Scope": "local",  "Driver": "bridge",  "IPAM": {  "Driver": "default",  "Options": null,  "Config": [  {  "Subnet": "172.19.0.0/16",  "Gateway": "172.19.0.1/16"  }  ]  },  "Containers": {  "a9f13f023761123115fcb2b454d3fd21666b8e1e0637f134026c44a7a84f1b0b": {  "Name": "docker\_a",  "EndpointID": "a5c8e08feda96d0de8f7c6203f2707dd3f9f6c3a64666126055b16a3908fafed",  "MacAddress": "02:42:ac:13:00:03",  "IPv4Address": "172.19.0.3/16",  "IPv6Address": ""  },  "c6532af99f691659b452c1cbf1693731a75cdfab9ea50428d9c99dd09c3e9a40": {  "Name": "docker\_b",  "EndpointID": "28a1877a0fdbaeb8d33a290e5a5768edc737d069d23ef9bbcc1d64cfe5fbe312",  "MacAddress": "02:42:ac:13:00:02",  "IPv4Address": "172.19.0.2/16",  "IPv6Address": ""  }  },  "Options": {}  }  ] |

**🡪**You might need to manually add it to your hosts file – Xiongbing Jin Mar 6 '16 at 17:01

@warmoverflow: Yes, it's possible to do it this way, but IP addresses are assigned dynamically to containers. Than is necessary to update the hosts file manually for each change :-(. So I would like to ask if is it possible to solve it without modification of the hosts file or using an extra discovery service? – Adam Bernau Mar 6 '16 at 17:32

I searched around and it does not seem possible without third party tool or some scripts. Why do you need to access containers directly from host without any port mapping?

-🡪 As [answered here](https://stackoverflow.com/a/45071126/2979435) there is a software solution for this, copying the anwser:

There is a opensource application that solves this issue, it's called [DNS Proxy Server](https://github.com/mageddo/dns-proxy-server)

It's a DNS server that solves containers hostnames, if could not found a hostname with that hostname then solve it from internet as well

Start the DNS Server

$ **docker run --hostname dns.mageddo --name dns-proxy-server -p 5380:5380 \**

**-v /var/run/docker.sock:/var/run/docker.sock \**

**-v /etc/resolv.conf:/etc/resolv.conf \**

**defreitas/dns-proxy-server**

**It will set as your default DNS automatically (and recover to the original when stops)**

Start your container for test

docker-compose up

docker-compose.yml

version: '2'

services:

redis:

container\_name: redis

image: redis:2.8

hostname: redis.dev.intranet

network\_mode: bridge # that way he can solve others containers names even inside, solve elasticsearch, for example

elasticsearch:

container\_name: elasticsearch

image: elasticsearch:2.2

hostname: elasticsearch.dev.intranet

Now solve your containers hostnames

**from host**

$ nslookup redis.dev.intranet

Server: 172.17.0.2

Address: 172.17.0.2#53

Non-authoritative answer:

Name: redis.dev.intranet

Address: 172.21.0.3

**from another container**

$ docker exec -it redis ping elasticsearch.dev.intranet

PING elasticsearch.dev.intranet (172.21.0.2): 56 data bytes

**As well it solves internet hostnames**

$ nslookup google.com

Server: 172.17.0.2

Address: 172.17.0.2#53

Non-authoritative answer:

Name: google.com

Address: 216.58.202.78

-🡪 The easiest way to do this is to add entries to your hosts file

* for linux: add 127.0.0.1 docker\_a docker\_b to /etc/hosts file
* for mac: similar to linux but use ip of virtual machine docker-machine ip default

-🡪 Here's what I do.

I wrote a Python script called [dnsthing](https://github.com/larsks/dnsthing/), which listens to the Docker events API for containers starting or stopping. It maintains a hosts-style file with the names and addresses of containers. Containers are named <container\_name>.<network>.docker, so for example if I run this:

docker run --rm --name mysql -e MYSQL\_ROOT\_PASSWORD=secret mysql

I get this:

172.17.0.2 mysql.bridge.docker

I then run a dnsmasq process pointing at this hosts file. Specifically, I run a dnsmasq instance using the following configuration:

listen-address=172.31.255.253

bind-interfaces

addn-hosts=/run/dnsmasq/docker.hosts

local=/docker/

no-hosts

no-resolv

And I run the dnsthing script like this:

dnsthing -c "systemctl restart dnsmasq\_docker" \

-H /run/dnsmasq/docker.hosts --verbose

So:

* dnsthing updates /run/dnsmasq/docker.hosts as containers stop/start
* After an update, dnsthing runs systemctl restart dnsmasq\_docker
* dnsmasq\_docker runs dnsmasq using the above configuration, bound to a local bridge interface with the address 172.31.255.253.
* The "main" dnsmasq process on my system, maintained by NetworkManager, uses this configuration from /etc/NetworkManager/dnsmasq.d/dockerdns:
* server=/docker/172.31.255.253

That tells dnsmasq to pass all requests for hosts in the .docker domain to the docker\_dnsmasq service.

This obviously requires a bit of setup to put everything together, but after that it seems to Just Work:

$ ping -c1 mysql.bridge.docker

PING mysql.bridge.docker (172.17.0.2) 56(84) bytes of data.

64 bytes from 172.17.0.2: icmp\_seq=1 ttl=64 time=0.087 ms

--- mysql.bridge.docker ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 0.087/0.087/0.087/0.000 ms

# 16.) [How to handle IP addresses when linking docker containers with each other using docker-compose?](https://stackoverflow.com/questions/33021084/how-to-handle-ip-addresses-when-linking-docker-containers-with-each-other-using)

|  |  |
| --- | --- |
|  | I am using docker-compose to build a complete development stack.  The application needs a mysql server to work.  The mysql server is an external container setup by docker-compose:  mysql:  image: mysql:5.6  volumes:  - /data/mysql:/var/lib/mysql  - ./docker/mysql.d:/etc/mysql/conf.d  ports:  - "3306:3306"  environment:  MYSQL\_ROOT\_PASSWORD: password  The application has its own docker-compose.yml and references the mysql container:  my-application:  build: . # the Dockerfile resides in the current folder  ports:  - "9180:80"  - "9543:443"  external\_links:  - mysql\_mysql\_1:mysql  environment:  DOCKER\_ENVIRONMENT: dev  DB\_NAME: local\_db  DB\_PASS: password  DB\_USER: root  DB\_HOST: # how to set the mysql's IP address?  I cannot pass them in the docker-compose as it is dynamic.  I know that the application is aware of the mysql IP address, as I have certain variables set:  application-container$ env|grep ADDR  MYSQL\_PORT\_3306\_TCP\_ADDR=172.17.0.241  Yet this is not my required DB\_HOST.  Can I map the variable somehow to DB\_HOST or set it differently? |

-🡪Why your mysql server is on an external docker-composer.yml file? Is this really required? Usually, if you have containerized applications that needs to run together you have to configure the in only one docker-composer.yml. – nessuno Oct 9 '15 at 9:06

@nessuno I want to use that mysql container for multiple projects. I don't need 12 mysql servers flying around. Furthermore, I map the mysql data to my a large HDD located at /data/mysql, as the mysql databse can become quite massive. If you have multiple mysql instances mapped to the same folder would lead to fileysystem lock issues. – k0pernikus Oct 9 '15 at 9:12

1

You can put the mysql container configuration together with your application, you just only have to run docker-compose with the --no-recreate flag, I guess. After that you have only one mysql container up, and you can reference that container directly in the docker-compose.yml file using his name. Thus make DB\_HOST: mysql. – nessuno Oct 9 '15 at 9:16

@nessuno Yet when I have another project also needing a mysql container than I would have two mysql containers if both were started. And if both were mapped to the same direcotory, they would lock each other out. I just want one mysql container for all applications I might develop on my machine. – k0pernikus Oct 9 '15 at 9:25

Sidenote: I don't consider my initial goal to have an external db container as a good solution anymore. Rather, each application stack should work on its own. – k0pernikus Apr 24 at 17:02

🡪

|  |  |
| --- | --- |
|  | You don't have to set the IP, but you can reference the container's virtual *hostname*, and this is the same value as you named your linked container.  This means you can indeed set the DB\_HOST from within the docker-compose.yml, either with links (recommended) or external\_links:  your\_application:  build: .  ports:  - "9180:80"  - "9543:443"  external\_links:  - mysql\_mysql\_1:docker-mysql  environment:  DB\_HOST: docker-mysql  As when you connect to your docker container, you could connect to your mysql container:  application-container $ mysql -h docker-mysql -uroot -ppassword -p 3360  It works the same when you link container's from the same docker-composer.yml as well.  This is also [documented](https://docs.docker.com/compose/compose-file/):  Link to containers in another service. Either specify both the service name and the link alias (SERVICE:ALIAS), or just the service name (which will also be used for the alias).  links:  - db  - db:database  - redis  An entry with the alias' name will be created in /etc/hosts inside containers for this service, e.g:  172.17.2.186 db  172.17.2.186 database  172.17.2.187 redis  Environment variables will also be created - see the [environment variable reference](https://docs.docker.com/compose/env/) for details. |

# 17.) [Docker Networking - nginx: [emerg] host not found in upstream](https://stackoverflow.com/questions/33639138/docker-networking-nginx-emerg-host-not-found-in-upstream)

I have recently started migrating to Docker 1.9 and Docker-Compose 1.5's networking features to replace using links.

So far with links there were no problems with nginx connecting to my php5-fpm fastcgi server located in a different server in one group via docker-compose. Newly though when I run docker-compose --x-networking up my php-fpm, mongo and nginx containers boot up, however nginx quits straight away with [emerg] 1#1: host not found in upstream "waapi\_php\_1" in /etc/nginx/conf.d/default.conf:16

However, if I run the docker-compose command again while the php and mongo containers are running (nginx exited), nginx starts and works fine from then on.

This is my docker-compose.yml file:

nginx:

image: nginx

ports:

- "42080:80"

volumes:

- ./config/docker/nginx/default.conf:/etc/nginx/conf.d/default.conf:ro

php:

build: config/docker/php

ports:

- "42022:22"

volumes:

- .:/var/www/html

env\_file: config/docker/php/.env.development

mongo:

image: mongo

ports:

- "42017:27017"

volumes:

- /var/mongodata/wa-api:/data/db

command: --smallfiles

This is my default.conf for nginx:

server {

listen 80;

root /var/www/test;

error\_log /dev/stdout debug;

access\_log /dev/stdout;

location / {

# try to serve file directly, fallback to app.php

try\_files $uri /index.php$is\_args$args;

}

location ~ ^/.+\.php(/|$) {

# Referencing the php service host (Docker)

fastcgi\_pass waapi\_php\_1:9000;

fastcgi\_split\_path\_info ^(.+\.php)(/.\*)$;

include fastcgi\_params;

# We must reference the document\_root of the external server ourselves here.

fastcgi\_param SCRIPT\_FILENAME /var/www/html/public$fastcgi\_script\_name;

fastcgi\_param HTTPS off;

}

}

How can I get nginx to work with only a single docker-compose call?

-🡪This can be solved with the mentioned depends\_on directive since it's implemented now (2016):

version: '2'

services:

nginx:

image: nginx

ports:

- "42080:80"

volumes:

- ./config/docker/nginx/default.conf:/etc/nginx/conf.d/default.conf:ro

depends\_on:

- php

php:

build: config/docker/php

ports:

- "42022:22"

volumes:

- .:/var/www/html

env\_file: config/docker/php/.env.development

depends\_on:

- mongo

mongo:

image: mongo

ports:

- "42017:27017"

volumes:

- /var/mongodata/wa-api:/data/db

command: --smallfiles

Successfully tested with:

$ docker-compose version

docker-compose version 1.8.0, build f3628c7

Find more details in the [documentation](https://docs.docker.com/compose/compose-file/#depends-on).

There is also a very interesting article dedicated to this topic: [Controlling startup order in Compose](https://docs.docker.com/compose/startup-order/)

-🡪I believe Nginx dont take in account Docker resolver (127.0.0.11), so please, can you try adding:

resolver 127.0.0.11

in your nginx configuration file?

# 17.) [Custom nginx container exits immediately when part of docker-compose](https://stackoverflow.com/questions/33724125/custom-nginx-container-exits-immediately-when-part-of-docker-compose)

I'm trying to learn how to use docker compose with a simple setup of an nginx container that reroutes requests to a ghost container. I'm using the standard ghost image but have a custom nginx image (that inherits from the standard image).

When I run the composition using "docker-compose up" it exits immediately with "docker\_nginx\_1 exited with code 0". However, when I build and run it manually, it runs fine and I can navigate my browser to the container and view the default nginx page. What am I misunderstanding about my compose file that causes it to behave differently than being custom built? What can I change to get it to stay running?

Disclaimer: I am also learning nginx as I go, so learning two things at once may be causing me undue problems.

EDIT: The original files were a bit more complex, but I've reduced the issue to simply: If I use the build command for a custom image that does nothing but inherit from the default nginx image, it exits immediately. If I use the default nginx image, it works. These are the now relevant files:

Compose file:

ghost:

expose:

- "2368"

image: ghost

nginx:

# image: nginx << If I use this instead of my custom build, it doesn't exit

build: ./nginx

ports:

- "80:80"

- "443:443"

links:

- ghost

nginx/Dockerfile:

FROM nginx

ORIGINAL FILES (with the same compose file as above):

nginx/Dockerfile:

FROM nginx

RUN rm /etc/nginx/nginx.conf

COPY conf/nginx.conf /etc/nginx/nginx.conf

COPY conf/sites-available/ghost /etc/nginx/sites-available/ghost

RUN mkdir /etc/nginx/sites-enabled

RUN ln -s /etc/nginx/sites-available/ghost /etc/nginx/sites-enabled/ghost

EXPOSE 80 443

# Is this even the right command I have no idea

CMD service nginx start

nginx/conf/nginx.conf:

daemon off;

user nginx;

# Let nginx figure out the processes I guess

worker\_processes auto;

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65;

#gzip on;

include /etc/nginx/conf.d/\*.conf;

}

nginx/conf/sites-available/ghost

server {

listen 80;

server\_name 127.0.0.1;

access\_log /var/log/nginx/localhost.log;

location / {

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header HOST $http\_host;

proxy\_set\_header X-NginX-Proxy true;

proxy\_pass http://0.0.0.0:2368;

proxy\_redirect off;

}

}

Running compose-up:

plays-MacBook-Pro:docker play$ docker-compose up

Creating docker\_ghost\_1...

Creating docker\_nginx\_1...

Attaching to docker\_ghost\_1, docker\_nginx\_1

docker\_nginx\_1 exited with code 0

Gracefully stopping... (press Ctrl+C again to force)

Stopping docker\_ghost\_1... done

Running manually:

plays-MacBook-Pro:nginx play$ docker build --no-cache -t nginx\_custom .

Sending build context to Docker daemon 8.704 kB

Step 0 : FROM nginx

---> 914c82c5a678

Step 1 : RUN rm /etc/nginx/nginx.conf

---> Running in 4ce9de96bb36

---> 98f97a9da4fc

Removing intermediate container 4ce9de96bb36

Step 2 : ADD conf/nginx.conf /etc/nginx/nginx.conf

---> dd3e089208a9

Removing intermediate container 36b9a47e0806

Step 3 : ADD conf/sites-available/ghost /etc/nginx/sites-available/ghost

---> 55fae53e5810

Removing intermediate container a82741d24af4

Step 4 : RUN mkdir /etc/nginx/sites-enabled

---> Running in 7659ead01b7b

---> 406be1c42394

Removing intermediate container 7659ead01b7b

Step 5 : RUN ln -s /etc/nginx/sites-available/ghost /etc/nginx/sites-enabled/ghost

---> Running in e9658a08affa

---> 021a84216e8a

Removing intermediate container e9658a08affa

Step 6 : EXPOSE 80 443

---> Running in 230e4523794c

---> 23d85e1a04cb

Removing intermediate container 230e4523794c

Step 7 : CMD service nginx start

---> Running in 209e129cae21

---> d7004d6fa223

Removing intermediate container 209e129cae21

Successfully built d7004d6fa223

plays-MacBook-Pro:nginx play$ docker run -t nginx\_custom

[It sits here on an empty line, running in the background]

# -🡪

Just ran into this same issue, and the initial fix was to change the name of the service in docker-compose.yml.

This worked, but the **reason** it worked is because Docker-compose caches the build & ties it to the service name. Every docker-compose up after the first one just uses what it built before, so any changes you make to the Dockerfile, or that section of the docker-compose.yml are basically ignored.

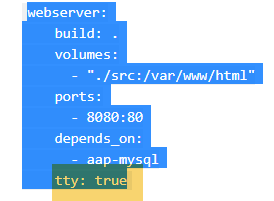
When you (and I) changed the service name, it triggered a new build since that service name hasn't been tagged before.

The **real** solution is to do a: [docker-compose build](https://docs.docker.com/v1.5/compose/cli/#build) to rebuild the image (followed by a docker-compose up). Their documentation doesn't really emphasize this issue.

-🡪You can also add a

tty: true

to the service in you docker-compose.yml .. eg

and it should stay running after docker-compose up

-🡪

|  |  |
| --- | --- |
|  | I figured out what it was. I needed to name the nginx part of my composition something other than 'nginx' . I'm not sure if it's because there is already an nginx image or if it is something else, but changing it made it work properly.  By changing my compose file to:  ghost:  expose:  - "2368"  image: ghost  mything:  # image: nginx  build: ./nginx  ports:  - "80:80"  - "443:443"  links:  - ghost  I was able to get it to work. An indicator was that when the name changed, I actually saw the build process output for my container. If anyone knows exactly why the naming needs to be that way, I'd love to know. |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | I just ran into this same issue...called my service "web". ...and nothing worked. ...hours of searching/tweaking for what I could have wrong in my setup eventually led me to this post. – [Gerrat](https://stackoverflow.com/users/429982/gerrat) [Mar 11 '16 at 21:39](https://stackoverflow.com/questions/33724125/custom-nginx-container-exits-immediately-when-part-of-docker-compose#comment59557055_33741351) | | |  |  | | --- | --- | | 1 |  | | @Merrillogic: see my answer for exactly why the naming needs to be that way – [Gerrat](https://stackoverflow.com/users/429982/gerrat) [Mar 11 '16 at 21:52](https://stackoverflow.com/questions/33724125/custom-nginx-container-exits-immediately-when-part-of-docker-compose#comment59557376_33741351) | |

# 17.) [How to replace volumes\_from in docker-composer v3](https://stackoverflow.com/questions/42244079/how-to-replace-volumes-from-in-docker-composer-v3)

 want to know the equivalent of the configuration below to suit version 3 of docker-composer.yml! volumes\_from is no longer valid so am I supposed to skip the data volume and replace it with top level volumes ?

version: '2'

services:

php:

build: ./docker-files/php-fpm/.

volumes\_from:

- data

working\_dir: /code

links:

- mysql

nginx:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf

volumes\_from:

- data

links:

- php

data:

image: tianon/true

volumes:

- .:/code

-🡪By default named volumes allow you to share data between containers. But it is some troubles with storing data in the same place on the host machine after restarting containers. But we can use [local-persist](https://github.com/CWSpear/local-persist) docker plugin for fix it.

For migration to version 3 you need

1) install [local-persist](https://github.com/CWSpear/local-persist) docker plugin (*if you want to store volumes data to the particular place on the host machine*)

2) modify docker-compose.yml

version: '3'

services:

php:

build: ./docker-files/php-fpm/.

volumes:

- data:/code

working\_dir: /code

links:

- mysql

nginx:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf

volumes:

- data:/code

links:

- php

data:

image: tianon/true

volumes:

- data:/code

# If you use local persist plugin

volumes:

data:

driver: local-persist

driver\_opts:

mountpoint: /path/on/host/machine/

# Or If you dont want using local persist plugin

volumes:

data:

Also you can store volumes data to the host machine with this volumes section:

volumes:

data:

external: true #< it means store my data to the host machine

But you can't specify path for this volume on host machine

-🡪Ok, but the data: service has no function any longer? And is there any downsides by just share the host volume directly? Like volumes: .:/code – prometheus Feb

you need data container only if this container encapsulate data inside himself. in this case you can simply backup this data, move container (with data) to another host. This is data only container pattern. But if you want use another approach: store data on the host machine then data container is unnecessary – Bukharov Sergey

installing third-party plugins for fundamental functionality (such us having a specific mountpoint on host) is unacceptable when you want to share your setup with other users. I am very disappointed by docker devs for leaving that basic functionality outside docker-compose version 3 – chefarov Jul 9 at 17:01

depends\_on can be used instead of links – Muzafar Ali Sep 12 at 20:50

# 18.) [How can I escape a $ dollar sign in a docker compose file?](https://stackoverflow.com/questions/40619582/how-can-i-escape-a-dollar-sign-in-a-docker-compose-file)

|  |  |
| --- | --- |
|  | I have a YAML scalar that is throwing the following error when I try to evaluate my docker-compose.yml file:  ERROR: Invalid interpolation format for "environment" option in service "time\_service": "${Time.now}"  YAML:  ---  version: '2'  services:  time\_service:  build: "."  environment:  TIME: "${Time.now}"  How can I maintain the same string output as written, but avoid having the docker-compose interpret it as faulty string interpolation? |

-🡪It is not the YAML parser that is interpreting that string. YAML doesn't know about ${}. Interpreting is done by docker-compose and that is written in Python, so the tag ruby was inappropriate as well. – Anthon Nov 15 '16 at 23:05

I removed the incorrect assumption that this is a YAML issue. It caused at least one other person to be misled and waste time – Anthon Jul 26 a

-🡪You are hitting the docker-compose variable substition, which is well documented [here](https://docs.docker.com/compose/compose-file/#variable-substitution):

Both $VARIABLE and ${VARIABLE} syntax are supported. Extended shell-style features, such as ${VARIABLE-default} and ${VARIABLE/foo/bar}, are not supported.

You can use a $$ (double-dollar sign) when your configuration needs a literal dollar sign. This also prevents Compose from interpolating a value, so a $$ allows you to refer to environment variables that you don’t want processed by Compose.

docker-compose is written in Python, as you see on [github](https://github.com/docker/compose), the doubling mechanism to get the original meaning of special characters can be found in many programs, I needed to use this myself, while programming, as far back in 1984

-🡪Found the answer by copying the suggestion for % characters in [this post](https://stackoverflow.com/questions/6051090/how-can-i-escape-characters-in-yaml)

It requires a double dollar sign $$.

So I needed "$${Time.now}", which evaluates to "${Time.now}"

# 18.) [Permission denied when mounting Docker volume in OSX](https://stackoverflow.com/questions/34442831/permission-denied-when-mounting-docker-volume-in-osx)

I'm at my wit's end with this, so hopefully you folks can help me. In OSX 10.11.2 with docker-machine, I've got a docker-compose file that should build a local Dockerfile and attach a MySQL container to it. The MySQL container should mount a local folder where I'm storing my database data, so if the container or VM comes down, I can just restart it without data loss. Problem is, when I run it, it throws a permissions error:

db\_1 | 2015-12-23 19:17:59 7facaa89b740 InnoDB: Operating system error number 13 in a file operation.

db\_1 | InnoDB: The error means mysqld does not have the access rights to

db\_1 | InnoDB: the directory.

I've tried every permutation I can think of to get this to work. I was reading around and it may have something to do with how docker-machine handles permissions with OSX, but the documentation for docker-machine says that it mounts the /Users folder, so that shouldn't be an issue.

Here's the docker-compose.yml:

web:

build: .

ports:

- "3000:3000"

links:

- db

db:

image: mysql:5.6

ports:

- "3306:3306"

volumes:

- /Users/me/Development/mysql-data:/var/lib/mysql

environment:

MYSQL\_ROOT\_PASSWORD: mypass

Any ideas? I can't help but think it's something really simple. Any help would be most appreciated!

**Edit:**

* Host - drwxr-xr-x 7 me staff 238 Dec 23 12:10 mysql-data/
* VM - drwxr-xr-x 1 docker staff 238 Dec 23 20:10 mysql-data/

As to the container, it won't run with the volume mounted. Without the -v mount, it is:

* Container - drwxr-xr-x 4 mysql mysql 4096 Dec 24 00:37 mysql

-🡪The issue this comes from is the userids used by Mac and Linux respectively. Mac does not like Linux wanting to use the 1 for the userID.

The way I worked around all the permissions craziness in my mac + docker-machine setup is to use this Dockerfile

FROM mysql:5.6

RUN usermod -u 1000 mysql

RUN mkdir -p /var/run/mysqld

RUN chmod -R 777 /var/run/mysqld

Instead of the plain MySQL 5.6 Image.

The last 2 lines are necessary, because changing the userid for the mysql user will mess up the build in permissions for that image. => you need the 777 permissions to make it run here :/

I know this is a little hacky, but so far the best solution I know to the permissions issue here.

---Okay, so that helped a lot! The image stays up at least. I'm not running it on a socket, though. Without the last two lines, though, I get this: [ERROR] Can't start server : Bind on unix socket: Permission denied [ERROR] Do you already have another mysqld server running on socket: /var/run/mysqld/mysqld.sock ? And then it does. Any ideas? – greggilbert Dec 24 '15 at 1:10

Oh sorry, my bad. I mixing thing sup a little here. If you switch user ids, this will mess with the permissions build into the base image. The userid 1000 will not have access to /var/run/mysqld => you need to adjust this manually. – Armin Braun Dec 24 '15 at 1:14

thanks for the note. Attempting to connect from the linked container says ERROR 1130 (HY000): Host '172.17.0.3' is not allowed to connect to this MySQL server, so I'm thinking that maybe I have to add my own my.cnf in there as well? – greggilbert Dec 24 '15 at 1:18

1

@alariva could you provide the logs for mysqld in your case? I'm sure we can figure it out from there. You could just add something like || cat /var/log/mysql\* to the end of your command for starting mysql. (sorry don't know the log path by heart here, but should be close to the above) – Armin Braun Nov 17 '16 at 14:35

1

@alariva ah now I see, sorry for missing the obvious. You cannot run /etc/init.d/mysql start from inside a RUN section (or during image build in general when using the debian master image)! Look at stackoverflow.com/questions/26938684/… and the first answer there for more details. Sorry, but you'll need to reorganise your build somehow :( – Armin Braun Nov 18 '16 at 19:42

# 19.) [Docker - PG::ConnectionBad](https://stackoverflow.com/questions/34340985/docker-pgconnectionbad)

Running Rails with Phusion Passenger, currently having trouble with postgres, whenever I run the build command I get this error:

PG::ConnectionBad (could not connect to server: No such file or directory

Is the server running locally and accepting

connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?

):

[My Dockerfile](http://pastebin.com/TF3VQkZT)

FROM phusion/passenger-customizable

MAINTAINER VodkaMD <support@nyvur.com>

ENV RACK\_ENV="production" RAILS\_ENV="production" SECRET\_KEY\_BASE="e09afa8b753cb175bcef7eb5f737accd02a4c16d9b6e5d475943605abd4277cdf47c488812d21d9c7117efd489d876f34be52f7ef7e88b21759a079339b198ce"

ENV HOME /root

CMD ["/sbin/my\_init"]

RUN /pd\_build/utilities.sh

RUN /pd\_build/ruby2.2.sh

RUN /pd\_build/python.sh

RUN /pd\_build/nodejs.sh

# Custom instructions

RUN apt-get update && apt-get install -y vim nano dialog net-tools build-essential wget

RUN apt-get install -y postgresql-client --no-install-recommends

RUN apt-get clean && rm -rf /var/lib/apt/lists/\* /tmp/\* /var/tmp/\*

# Generate SSL Certificate

RUN mkdir /etc/nginx/ssl

# RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt

# Enable Nginx and Passenger

RUN rm -f /etc/service/nginx/down

# Enable Redis

RUN rm -f /etc/service/redis/down

# Enable SSH

RUN rm -f /etc/service/sshd/down

# Enable Postgres

RUN rm -f /etc/service/postgres/down

# Install bundle of gems

WORKDIR /tmp

ADD Gemfile /tmp/

ADD Gemfile.lock /tmp/

RUN bundle install

# Copy application folder

RUN mkdir /home/app/Nyvur

ADD . /home/app/Nyvur

RUN chown -R app:app /home/app/Nyvur

WORKDIR /home/app/Nyvur

RUN bundle exec rake assets:precompile

# Add a virtual host entry

RUN rm /etc/nginx/sites-enabled/default

COPY config/nginx\_configs.conf /etc/nginx/sites-enabled/Nyvur.conf

ADD config/postgres-env.conf /etc/nginx/main.d/postgres-env.conf

RUN mkdir -p /var/www/Nyvur/tmp/pids

RUN mkdir -p /var/www/Nyvur/tmp/sockets

RUN mkdir -p /var/www/Nyvur/log

RUN apt-get clean && rm -rf /var/lib/apt/lists/\* /tmp/\* /var/tmp/\*

EXPOSE 80 443

RUN bundle exec passenger start -p 80 -e production

[My docker-compose](http://pastebin.com/eGtRBDr4)

postgres:

restart: always

image: postgres:9.4.4

ports:

- "5432:5432" # Bind host port 5432 to PostgreSQL port 5432

volumes:

- ./db/dumps:/app-db-dumps

- ./db/postgres-setup.sql/docker-entrypoint-initdb.d/setup.sql

environment:

LC\_ALL: C.UTF-8

POSTGRES\_PASSWORD: 3x1mpl3

mongo:

image: mongo:3.0.7

ports:

- "27017:27017"

command: --smallfiles --rest --auth

redis:

image: redis:3.0.4

ports:

- "6379:6379" # Bind host port 6379 to Redis port 6379

web: &app\_base

build: .

ports:

- "80:80"

volumes:

- .:/Nyvur

links: &app\_links

- postgres:postgres.local

- redis:redis.local

environment: &app\_environment

# PostgreSQL Development Database:

# DATABASE\_URL: postgres://postgres:3x1mpl3@postgres.local:5432/App\_production?pool=25&encoding=unicode&schema\_search\_path=public

REDIS\_URL: redis://redis.local:6379

SIDEKIQ\_CONCURRENCY: 5

SIDEKIQ\_TIMEOUT: 10

ENABLE\_DEBUG\_SERVER: true

RACK\_ENV: production

RAILS\_ENV: production

worker:

<<: \*app\_base

ports: []

command: sidekiq -c 25 -e production -q default

environment:

SECRET\_KEY\_BASE: e09afa8b753cb175bcef7eb5f737accd02a4c16d9b6e5d475943605abd4277cdf47c488812d21d9c7117efd489d876f34be52f7ef7e88b21759a079339b198ce

environment:

# SECRET\_KEY\_BASE:

RAILS\_ENV: production

[Full error log](http://pastebin.com/9UMZ4y03)

=============== Phusion Passenger Standalone web server started ===============

PID file: /home/app/Nyvur/passenger.80.pid

Log file: /home/app/Nyvur/log/passenger.80.log

Environment: production

Accessible via: http://0.0.0.0/

You can stop Phusion Passenger Standalone by pressing Ctrl-C.

Problems? Check https://www.phusionpassenger.com/library/admin/standalone/troubleshooting/

===============================================================================

App 107 stderr: stdin: is not a tty

App 107 stdout:

App 107 stdout: I, [2015-12-17T17:34:39.223372 #107] INFO -- : \*\* [Raven] Raven 0.12.3 ready to catch errors

I, [2015-12-17T17:34:40.413454 #127] INFO -- : Started HEAD "/" for 127.0.0.1 at 2015-12-17 17:34:40 +0000

App 127 stdout:

F, [2015-12-17T17:34:41.543651 #127] FATAL -- :

PG::ConnectionBad (could not connect to server: No such file or directory

Is the server running locally and accepting

connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?

):

activerecord (4.1.2) lib/active\_record/connection\_adapters/postgresql\_adapter.rb:888:in `initialize'

activerecord (4.1.2) lib/active\_record/connection\_adapters/postgresql\_adapter.rb:888:in `new'

activerecord (4.1.2) lib/active\_record/connection\_adapters/postgresql\_adapter.rb:888:in `connect'

activerecord (4.1.2) lib/active\_record/connection\_adapters/postgresql\_adapter.rb:568:in `initialize'

activerecord (4.1.2) lib/active\_record/connection\_adapters/postgresql\_adapter.rb:41:in `new'

activerecord (4.1.2) lib/active\_record/connection\_adapters/postgresql\_adapter.rb:41:in `postgresql\_connection'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:435:in `new\_connection'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:445:in `checkout\_new\_connection'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:416:in `acquire\_connection'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:351:in `block in checkout'

/usr/lib/ruby/2.2.0/monitor.rb:211:in `mon\_synchronize'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:350:in `checkout'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:265:in `block in connection'

/usr/lib/ruby/2.2.0/monitor.rb:211:in `mon\_synchronize'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:264:in `connection'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:541:in `retrieve\_connection'

activerecord (4.1.2) lib/active\_record/connection\_handling.rb:113:in `retrieve\_connection'

activerecord (4.1.2) lib/active\_record/connection\_handling.rb:87:in `connection'

activerecord (4.1.2) lib/active\_record/query\_cache.rb:51:in `restore\_query\_cache\_settings'

activerecord (4.1.2) lib/active\_record/query\_cache.rb:43:in `rescue in call'

activerecord (4.1.2) lib/active\_record/query\_cache.rb:32:in `call'

activerecord (4.1.2) lib/active\_record/connection\_adapters/abstract/connection\_pool.rb:621:in `call'

actionpack (4.1.2) lib/action\_dispatch/middleware/callbacks.rb:29:in `block in call'

activesupport (4.1.2) lib/active\_support/callbacks.rb:82:in `run\_callbacks'

actionpack (4.1.2) lib/action\_dispatch/middleware/callbacks.rb:27:in `call'

actionpack (4.1.2) lib/action\_dispatch/middleware/remote\_ip.rb:76:in `call'

actionpack (4.1.2) lib/action\_dispatch/middleware/debug\_exceptions.rb:17:in `call'

actionpack (4.1.2) lib/action\_dispatch/middleware/show\_exceptions.rb:30:in `call'

railties (4.1.2) lib/rails/rack/logger.rb:38:in `call\_app'

railties (4.1.2) lib/rails/rack/logger.rb:20:in `block in call'

activesupport (4.1.2) lib/active\_support/tagged\_logging.rb:68:in `block in tagged'

activesupport (4.1.2) lib/active\_support/tagged\_logging.rb:26:in `tagged'

activesupport (4.1.2) lib/active\_support/tagged\_logging.rb:68:in `tagged'

railties (4.1.2) lib/rails/rack/logger.rb:20:in `call'

request\_store (1.1.0) lib/request\_store/middleware.rb:8:in `call'

actionpack (4.1.2) lib/action\_dispatch/middleware/request\_id.rb:21:in `call'

rack (1.5.2) lib/rack/methodoverride.rb:21:in `call'

rack (1.5.2) lib/rack/runtime.rb:17:in `call'

activesupport (4.1.2) lib/active\_support/cache/strategy/local\_cache\_middleware.rb:26:in `call'

rack (1.5.2) lib/rack/sendfile.rb:112:in `call'

sentry-raven (0.12.3) lib/raven/integrations/rack.rb:61:in `call'

railties (4.1.2) lib/rails/engine.rb:514:in `call'

railties (4.1.2) lib/rails/application.rb:144:in `call'

passenger (5.0.22) src/ruby\_supportlib/phusion\_passenger/rack/thread\_handler\_extension.rb:97:in `process\_request'

passenger (5.0.22) src/ruby\_supportlib/phusion\_passenger/request\_handler/thread\_handler.rb:160:in `accept\_and\_process\_next\_request'

passenger (5.0.22) src/ruby\_supportlib/phusion\_passenger/request\_handler/thread\_handler.rb:113:in `main\_loop'

passenger (5.0.22) src/ruby\_supportlib/phusion\_passenger/request\_handler.rb:416:in `block (3 levels) in start\_threads'

passenger (5.0.22) src/ruby\_supportlib/phusion\_passenger/utils.rb:113:in `block in create\_thread\_and\_abort\_on\_exception'

I, [2015-12-17T17:34:41.559638 #127] INFO -- : Rendered /var/lib/gems/2.2.0/gems/actionpack-4.1.2/lib/action\_dispatch/middleware/templates/rescues/\_source.erb (0.6ms)

I, [2015-12-17T17:34:41.566011 #127] INFO -- : Rendered /var/lib/gems/2.2.0/gems/actionpack-4.1.2/lib/action\_dispatch/middleware/templates/rescues/\_trace.html.erb (1.1ms)

I, [2015-12-17T17:34:41.572608 #127] INFO -- : Rendered /var/lib/gems/2.2.0/gems/actionpack-4.1.2/lib/action\_dispatch/middleware/templates/rescues/\_request\_and\_response.html.erb (0.9ms)

I, [2015-12-17T17:34:41.572727 #127] INFO -- : Rendered /var/lib/gems/2.2.0/gems/actionpack-4.1.2/lib/action\_dispatch/middleware/templates/rescues/diagnostics.erb within rescues/layout (20.2ms)

^CERROR:

Aborting.

Any help is really appreciated, I'm still getting the hang of Docker and working on production

-🡪I think the application is assuming the database is available immediately. It can sometimes take a few seconds to start up. I would either have the application retry the connection a few times with a short delay, or create an entrypoint script to poll and wait for it to be available. – dnephin Dec 17 '15 at 22:56

That makes sense, I found a script for that purpose, I'll use it again with the new changes and see how that goes.

-🡪 The fact that it's trying to connect to a Unix domain socket points to a configuration issue in the Rails application. Your docker-compose.yml has DATABASE\_URL in it, but it's commented out -- uncommenting that should at least get closer (the application should at least attempt to start connecting to postgres.local instead of a local Unix domain socket which isn't going to exist since Postgres is running in a separate container).

---Thanks a lot, made me realize something, I shouldn't be running passenger from the Dockerfile, since it would before any docker-compose container even fires. All working well, only got to work on nginx and I should be good to go :) – whatAboutJohn Dec 18 '15 at 21:27

# 18.) [Dockerfile pass environments on docker compose build](https://stackoverflow.com/questions/45933083/dockerfile-pass-environments-on-docker-compose-build)

 have written a Dockerfile which uses two arguments:

FROM jessie

MAINTAINER Zeinab Abbasimazar

#Build Arguments

ARG REP\_USER

ARG REP\_PASS

# Build

RUN echo 'REP\_USER:'$REP\_USER', REP\_PASS:'$REP\_PASS

I wrote a docker-compose.yml for build:

version: "2"

services:

ui:

build:

context: .

dockerfile: Dockerfile

args:

REP\_USER: $REP\_USER

REP\_PASS: $REP\_PASS

I don't want to define these arguments directly in the compose file, so I tried to send them during docker compose build:

REP\_USER=myusername REP\_PASS=mypassword docker-compose build

Which didn't work. I changed my Dockerfile to use these arguments as environment variables; so I removed ARG lines:

FROM jessie

MAINTAINER Zeinab Abbasimazar

# Build

RUN echo 'REP\_USER:'$REP\_USER', REP\_PASS:'$REP\_PASS

And docker-compose.yml:

version: "2"

services:

ui:

build:

context: .

dockerfile: Dockerfile

And ran REP\_USER=myusername REP\_PASS=mypassword docker-compose build; still no result.

I also tried to save these information into an env file:

version: "2"

services:

ui:

build:

context: .

dockerfile: Dockerfile

env\_file:

- myenv.env

But it seems env files doesn't affect at build time; they are just take part into run time.

**EDIT 1:**

Docker version is 1.12.6 which doesn't support passing arguments with --build-arg.

**EDIT 2:**

I tried using .env file as described [here](https://docs.docker.com/compose/environment-variables/#the-env-file):

cat .env

REP\_USER=myusername

REP\_PASS=mypassword

I then called docker-compose config which returned:

networks: {}

services:

ui:

build:

args:

REP\_PASS: mypassword

REP\_USER: myusername

context: /home/zeinab/Workspace/ZiZi-Docker/Test/test-exec-1

dockerfile: Dockerfile

version: '2.0'

volumes: {}

Which means this resolved my issue.

**EDIT 3:**

I also tried third section of [docker-compose arg documentation](https://docs.docker.com/compose/compose-file/#args) in my docker-compose.yml file:

version: "2"

services:

ui:

build:

context: .

dockerfile: Dockerfile

args:

- REP\_USER

- REP\_PASS

And executed:

export REP\_USER=myusername;export REP\_PASS=mypassword;sudo docker-compose build --no-cache

Still not getting what I wanted.

-🡪I finally found the solution. I mentioned it in the question too. I first tried it with fail, then I found out that I had a typo naming .env file; it was .evn.

I tried using .env file as described [here](https://docs.docker.com/compose/environment-variables/#the-env-file):

cat .env

REP\_USER=myusername

REP\_PASS=mypassword

I then called docker-compose config which returned:

networks: {}

services:

ui:

build:

args:

REP\_PASS: mypassword

REP\_USER: myusername

context: /home/zeinab/Workspace/ZiZi-Docker/Test/test-exec-1

dockerfile: Dockerfile

version: '2.0'

volumes: {}

Which means this resolved my issue. I should mention that [this answer](https://stackoverflow.com/questions/39414373/docker-compose-using-environment-variables-to-set-extra-host?answertab=active#tab-top) was really helpful.

-🡪

|  |
| --- |
| You can set build arguments with docker compose as described [here](https://docs.docker.com/compose/reference/build/):  docker-compose build [--build-arg key=val...]  docker-compose build --build-arg REP\_USER=myusername --build-arg REP\_PASS=mypassword  Btw, AFAIK build arguments are a compromise between usability and deterministic building. Docker aims to build in a deterministic fashion. That is, wherever you execute the build the produced image should be the same. Therefore, it appears logical that the client ignores the environment (variables) it is executed in.  ---my docker version is 1.12.6 which doesn't support that. – Zeinab Abbasimazar Aug 29 at 7:34    To bad. Updating is no option? In a "normal" Linux you can just put a recent version of docker-compose into $HOME/bin and it should be executed instead of the system default. – fzgregor Aug 29 at 7:39    I'll check if updating is possible; not all work approaches are in my control at the office. – Zeinab Abbasimazar Aug 29 at 7:42    Yeah, that's what I meant with "normal". Some corporate security measures might forbid it. If $HOME/bin is in the output of ` echo "$PATH"` it should work. (In principle) – fzgregor Aug 29 at 7:45 |
|  |  |

-🡪The correct syntax for variable substitution in a docker-compose file is ${VARNAME}. Try with this one:

version: "2"

services:

ui:

build:

context: .

dockerfile: Dockerfile

args:

REP\_USER: ${REP\_USER}

REP\_PASS: ${REP\_PASS}

--it didn't work for me – [Zeinab Abbasimaza](https://stackoverflow.com/users/1626977/zeinab-abbasimazar)

# 19.) [Docker container cannot connect to host machine: No route to host](https://stackoverflow.com/questions/43879637/docker-container-cannot-connect-to-host-machine-no-route-to-host)

I've been trying to setup a docker environment using docker compose. One issue that has me stumped is that my docker containers cannot reach my host machine.

I setup a container using the following compose file:

version: '3'

services:

webapp:

image: ...

ports:

- "8080:8080"

When I enter the container, I am able to ping my host machine:

ping ${dockerHostIP}

However when I try and retrieve the home page using curl inside the container:

curl http://${dockerHostIP}:8080

I get:

curl: (7) Failed connect to ${dockerHostIP}:8080; No route to host

I cannot figure out what should be done to resolve this No route to host error. Unfortunately I need to be able to do this as the web application makes requests using its hostname internally.

Traceroute Results:

traceroute ${dockerHostIP}

traceroute to ${dockerHostIP} (${dockerHostIP}), 30 hops max, 60 byte packets

1 ${dockerHostName} (${dockerHostIP}) 0.039 ms !X 0.012 ms !X 0.007 ms !X

-🡪Really strange... what's result of traceroute ${dockerHostIP}? – Bor Laze May 9 at 21:01

Thanks for looking into this... I've edited the question with the traceroute results. – Brian DiCasa May 9 at 21:11

1

I guess, webapp is up and you can access http://${dockerHostIP}:8080 from docker host? I have only two ideas: 1) webapp doesn't accept connections from container or 2) on docker host firewall blocks incoming connections from container. BTW, can you connect to host via telnet ${dockerHostIP} 8080? – Bor Laze May 9 at 21:21

I can access http://${dockerHostIP}:8080 from the docker host. telnet ${dockerHostIP} 8080 returns Trying ${dockerHostIP}... telnet: Unable to connect to remote host: No route to host I've tried completely disabling the firewall on my host machine systemctl firewalld stop and then spinning up the containers using docker-compose up -d and I still have the same problem – Brian DiCasa May 9 at

2

**Well this is embarrassing... turns out it was a firewall issue. I thought I tried disabling the firewall and running, but docker-compose up wont run without firewalld up. When I opened up the port using "firewall-cmd --permanent --add-port=8080/tcp" on the docker host, this resolved the issue. Thanks for pointing me in the right direction! – Brian DiCasa May 9 at 21:44**

# 20.) [How to mount docker volume into my docker project using compose?](https://stackoverflow.com/questions/39977955/how-to-mount-docker-volume-into-my-docker-project-using-compose)

 have a Maven project. I'm running my Maven builds inside Docker. But the problem with that is it downloads all of the Maven dependencies every time I run it and it does not cache any of those Maven downloads.

I found some work arounds for that, where you mount your local .m2 folder into Docker container. But this will make the builds depend on local setup. What I would like to do is to create a volume (long live) and link/mount that volume to .m2 folder inside Docker. That way when I run the Docker build for the 2nd time, it will not download everything. And it will not be dependent on environment.

How can I do this with docker-compose?

-🡪

|  |  |
| --- | --- |
|  | Without knowing your exact configuration, I would use something like this...  version: "2"  services:  maven:  image: whatever  volumes:  - m2-repo:/home/foo/.m2/repository  volumes:  m2-repo:  This will create a data volume called m2-repo that is mapped to the /home/foo/.m2/repository(adjust path as necessary). The data volume will survive up/down/start/stop of the Docker Compose project.  You can delete the volume by running something like docker-compose down -v, which will destroy containers and volumes. |

---you can run mvn compile as part of your docker build and then you push that image to some registry, whenever you pull the image it has your .m2 folder cached in it and running builds should be significantly faster – bjhaid Oct 11 '16 at 13:16

tx, but then i have to manage another image. And things change, – dinesh707 Oct 11 '16 at 13:18

1

the .m2 folder has to live somewhere for it to move around environments conveniently and my suggestion is making it live in the image to dedup work

Yes, that would work if you want to have a "frozen" version of all of the dependencies included in the image. The dependencies would be downloaded and/or copied into the image at build time of the image. If you want to persist them at run time, please see my answer for a solution using a data volume. –

# 21.) [Docker run script in host on docker-compose up](https://stackoverflow.com/questions/43208979/docker-run-script-in-host-on-docker-compose-up)

thank you for watching this question.

So my question relates to best practices on how to run a script on a docker-compose up directive.

Currently i'm sharing a volume between host and container to allow for the script changes to be visible to both host and container. Similar to a watching script polling for changes on configuration file.

The script has to act on host on changes according to predefined rules.

So my question is:

How could I start this script on a docker-compose up directive or even from the dockerfile of the service, so that whenever the container goes up the "watcher" can find any changes being made and writing to.

The container in question will always run over a debian / ubuntu OS and should be architecture independent, meaning it should be able to run on ARM as well.

Thanks in advance

I would like to know as well, why was the question downvoted just because some people don't understand it.

I wish to run a script on the Host, not inside the container.. I need the Host to change its network interface configurations to easily adapt any environment The HOST needs to change I repeat.. This should be seamless to the user, and easily editable on a Web interface running Inside a CONTAINER to adapt to new environments.

I currently do this with a script running on the host based on crontab. I just wish to know the best practices and examples of how to run a script on HOST from INSIDE a CONTAINER, so that the deploy can be as easy for the installing operator to just run docker-compose up.

Thank you

**-🡪** just wish to know the best practices and examples of how to run a script on HOST from INSIDE a CONTAINER, so that the deploy can be as easy for the installing operator to just run docker-compose up

It seems that there is no best practice that can be applied to your case. A workaround proposed here: [How to run shell script on host from docker container?](https://stackoverflow.com/questions/32163955/how-to-run-shell-script-on-host-from-docker-container) is to use a client/server trick.

1. The host should run a small server (choose a port and specify a request type that you should be waiting for)
2. The container, after it starts, should send this request to that server
3. The host should then run the script / trigger the changes you want

This is something that might have serious security issues, so use at your own risk.

**-🡪**You can create an alias for docker-compose up. Put something like this in ~/.bash\_aliases (in Ubuntu):

alias up="docker-compose up; ~/your\_script.sh"

I'm not sure if running scripts on the host from a container is possible, but if it's possible, it's a severe security flaw. Containers should be isolated, that's the point of using containers.

**-🡪**The script needs to run continuously in the foreground.

In your Dockerfile use the [CMD](https://docs.docker.com/engine/reference/builder/#cmd) directive and define the script as the parameter.

When using the cli, use docker run -d IMAGE SCRIPT

# 22.) [docker executable file not found in $PATH](https://stackoverflow.com/questions/41167349/docker-executable-file-not-found-in-path)

Trying to run a rails migration on a running docker compose container throws this error:

$ docker-compose run webapp rails db:migrate

ERROR: Cannot start service webapp: invalid header field value "oci runtime error: container\_linux.go:247: starting container process caused \"exec: \\"rails\\": executable file not found in $PATH\"\n"

However, I can access rails from inside the container:

$ docker-compose run webapp bash

root@3fd3a87275a1:/home/app/webapp# which rails

/usr/local/rvm/gems/ruby-2.3.1/bin/rails

My container is already running and I can GET the page:

$ curl http://localhost -i

HTTP/1.1 200 OK

Content-Type: text/html; charset=utf-8

Transfer-Encoding: chunked

Connection: keep-alive

Status: 200 OK

Cache-Control: max-age=0, private, must-revalidate

ETag: W/"b62d4f67b7b823c017534cd9727752cd"

X-Frame-Options: SAMEORIGIN

X-XSS-Protection: 1; mode=block

X-Content-Type-Options: nosniff

X-Runtime: 0.019881

X-Request-Id: eb32ec21-3a8f-4caa-a27b-2e42f0b2bce9

Date: Thu, 15 Dec 2016 15:15:59 GMT

X-Powered-By: Phusion Passenger 5.0.29

Server: nginx/1.10.1 + Phusion Passenger 5.0.29

How do I run rails db:migrate in my container?

## **Dockerfile**

FROM phusion/passenger-ruby23

# Set correct environment variables.

ENV HOME /root

# Use baseimage-docker's init process.

CMD ["/sbin/my\_init"]

# additional packages

RUN apt-get update

# Active nginx

RUN rm -f /etc/service/nginx/down

# Install bundle of gems

WORKDIR /tmp

ADD Gemfile /tmp/

ADD Gemfile.lock /tmp/

RUN bundle install

# Copy the nginx template for configuration and preserve environment variables

RUN rm /etc/nginx/sites-enabled/default

# Add the nginx site and config

ADD webapp.conf /etc/nginx/sites-enabled/webapp.conf

RUN mkdir /home/app/webapp

COPY . /home/app/webapp

RUN usermod -u 1000 app

RUN chown -R app:app /home/app/webapp

WORKDIR /home/app/webapp

# Clean up APT when done.

RUN apt-get clean && rm -rf /var/lib/apt/lists/\* /tmp/\* /var/tmp/\*

EXPOSE 80

## **Docker Compose**

version: '2'

services:

webapp:

build: .

container\_name: myapp

working\_dir: /home/app/webapp

ports:

- "80:80"

environment:

- PASSENGER\_APP\_ENV=development

volumes:

- .:/home/app/webapp

networks:

- back-end

db:

container\_name: db

image: postgres:9.4

networks:

- back-end

networks:

back-end:

driver: bridge

**--🡪**

post your Dockerfile (ENTRYPOINT and CMD may be enough), and your docker run command – user2915097 Dec 15 '16 at 15:12

@user2915097 added. – amingilani Dec 15 '16 at 15:15

I suppose you have read docs.docker.com/compose/rails – user2915097 Dec 15 '16 at 15:46

1

I just faced the same error, and docker-compose run webapp rake db:migrate resolved it for me. – Drew Ogryzek Dec 16 '16 at 2:07

@DrewOgryzek it works! But that's just weird, I think I should file this as a bug report.

--🡪I don't know why this is happening, and I've filed an [issue](https://github.com/phusion/passenger-docker/issues/161), but in the meantime, here's a **workaround:**

docker-compose run webapp bundle exec rails db:migrate

# 23.) [Can't delete Docker Image from Registry](https://stackoverflow.com/questions/39918794/cant-delete-docker-image-from-registry)

Hi I want to delete a docker image from my private registry the steps that I did was:

I already did what the solution of [How can I use the Docker Registry API V2 to delete an image from a private registry?](https://stackoverflow.com/q/37033055/6638204) Recommended and it did not work

* I did a HEAD request to get the Docker-Content-Digest

curl --cacert ~/Documents/certificates//ca.pem --key ~/Documents/certificates//key.pem --cert ~/Documents/certificates/certificate.p12 --pass certpass -I https://myprivateregistry/v2/imagename/manifests/tag

* Then using the Dcker-content-digest from the previous step I did a delete request:

curl --cacert ~/Documents/certificates//ca.pem --key ~/Documents/certificates//key.pem --cert ~/Documents/certificates/certificate.p12 --pass certpass --header "Accept: application/vnd.docker.distribution.manifest.v2+json" -X DELETE https://myprivateregisty/v2/imagename/manifests/dockercontentdigestgotfrompreviousstep

* I got this error:

{"errors":[{"code":"MANIFEST\_UNKNOWN","message":"manifest unknown"}]}

-🡪In all likelihood, it means, that you have deleted the manifest, and this is right first step. To delete actual data from disk, you need to run docker registry garbage collector on registry host machine.

docker exec -it registry bin/registry garbage-collect /etc/docker/registry/config.yml

The info is from [that comment](https://github.com/docker/docker-registry/issues/988#issuecomment-224280919)

Also, as some adv, I want to propose you to check my [docker registry web UI](https://github.com/Evedel/bow) =) There is the possibility to delete an images from registry right with that UI.

# 24.) [Docker Compose - How reference many schemas in one mysql container](https://stackoverflow.com/questions/40336248/docker-compose-how-reference-many-schemas-in-one-mysql-container)

I'm trying to use two schemas into one mysql container. I have two flyway services that connect to two different schemas. The .yml file of Docker Compose looks like:

version: '2'

services:

mysqldb:

image: mysql:5.6.26

environment:

MYSQL\_USER: user

MYSQL\_PASSWORD: password

MYSQL\_ROOT\_PASSWORD: password

MYSQL\_DATABASE:

- my

- my\_post

ports:

- "3306:3306"

flyway-service1-i:

image: mik/flyway-service

volumes:

- "../resources/db/migration:/migrations/ro"

depends\_on:

- mysqldb

links:

- mysqldb

command: migrate -url=jdbc:mysql://mysqldb:3306/mi -user=user -password=password -baselineOnMigrate=true -locations='filesystem:/migrations'

flyway-service2-i:

image: mialk/flyway-post-service

volumes:

- "../../../service2/src/main/resources/db/migration:/migrations/ro"

depends\_on:

- mysqldb

links:

- mysqldb

command: migrate -url=jdbc:mysql://mysqldb:3306/mi\_post -user=user -password=password -baselineOnMigrate=true -locations='filesystem:/migrations'

But when I run the command sudo docker-compose up the terminal show this message:

**ERROR: The Compose file './docker-compose.yml' is invalid because: services.mysqldb.environment.MYSQL\_DATABASE contains ["mialquiler", "mialquiler\_post"], which is an invalid type, it should be a string, number, or a null**

I traid without specifying MYSQL\_DATABASE property, but it didn't work.

Is there any way to do that?

-🡪The MYSQL\_DATABASE variable allows a single database to be created, and permissions granted on the database to the MYSQL\_USER if specified.

You can use a single database to house multiple schema's.

If you need to create multiple databases you may need to run some custom SQL as flyway can't do database creation for you. The [flyway test resources include a mysql example](https://github.com/flyway/flyway/blob/master/flyway-core/src/test/resources/migration/dbsupport/mysql/createDatabase.sql).

----Thank you very much for your answer @Matt :) . So, I can use flyway in the .yml file in this way for example:**command: migrate -url=jdbc:mysql://mysqldb:3306/mi\_post -user=user -password=password -eschemas=my\_post\_schema1 -baselineOnMigrate=true -locations='filesystem:/migrations'** Thanks! :

# 25.) [Using Docker-Compose, how to execute multiple commands](https://stackoverflow.com/questions/30063907/using-docker-compose-how-to-execute-multiple-commands)

I want to do something like this where I can run multiple commands in order.

db:

image: postgres

web:

build: .

command: python manage.py migrate

command: python manage.py runserver 0.0.0.0:8000

volumes:

- .:/code

ports:

- "8000:8000"

links:

- db

-🡪As of Docker Compose version 2 (and incl. 3), you can specify a *list* of arguments as in a Dockerfile:

version: '3'

# version: '2'

services:

app:

image: foo:1-bar

command: ["foo", "bar", "baz"]

---

docker compose v2 also uses lists docs.docker.com/compose/compose-file/compose-file-v2/#comman‌​d – Stephen Cochran Jul 21 at

3

That list is an ARGV of a single command, not multiple commands to run in succession. – Chris Hillery Sep 12 at 22:07

-🡪

|  |  |
| --- | --- |
|  | I run pre-startup stuff like migrations in a separate ephemeral container, like so (note, compose file has to be of version '2' type):  db:  image: postgres  web:  image: app  command: python manage.py runserver 0.0.0.0:8000  volumes:  - .:/code  ports:  - "8000:8000"  links:  - db  depends\_on:  - migration  migration:  build: .  image: app  command: python manage.py migrate  volumes:  - .:/code  links:  - db  depends\_on:  - db  This helps things keeping clean and separate. Two things to consider:   1. You have to ensure the correct startup sequence (using depends\_on) 2. you want to avoid multiple builds which is achieved by tagging it the first time round using build and image; you can refer to image in other containers then |

---This seems like the best option to me, and I would like to use it. Can you elaborate on your tagging setup to avoid multiple builds? I would prefer to avoid extra steps, so if this needs some, I might go with bash -c above. – Stavros Korokithakis Apr 28 '16 at 12:21

In the yaml above, the build and tagging happens in the migration section. It's not really obvious at first sight, but docker-compose tags it when you specify the build AND image properties - whereby the image property specifies the tag for that build. That can then be used subsequently without triggering a new build (if you look at web, you see it has no build but only an image property). Here's some more details docs.docker.com/compose/compose-file) – Bjoern Stiel Apr 28 '16 at 12:35

That's very informative (and I completely missed that in the docs), thank you. – Stavros Korokithakis Apr 30 '16 at 10:35

4

While I like the idea of this, the problem is that depends\_on only ensures they start in that order, not that they are ready in that order. wait-for-it.sh may be the solution some people need. – traday Oct 26 '16 at 1:40

1

That is absolutely correct and a bit of a shame that docker-compose doesn't support any fine grained control like waiting for a container to exit or start listening on a port. But yes, a custom script does solve this, good point! – Bjoern Stiel Oct 26 '16 at 4:22

--🡪You can use entrypoint here. entrypoint in docker is executed before the command while command is the default command that should be run when container starts. So most of the applications generally carry setup procedure in entrypoint file and in the last they allow command to run.

make a shell script file may be as docker-entrypoint.sh (name does not matter) with following contents in it.

#!/bin/bash

python manage.py migrate

exec "$@"

in docker-compose.yml file use it with entrypoint: /docker-entrypoint.sh and register command as command: python manage.py runserver 0.0.0.0:8000 P.S : do not forget to copy docker-entrypoint.sh along with your code.

-🡪Use a tool such as wait-for-it or [dockerize](https://github.com/jwilder/dockerize). These are small wrapper scripts which you can include in your application’s image. Or write your own wrapper script to perform a more application-specific commands. according to: <https://docs.docker.com/compose/startup-order/>

-🡪

|  |  |  |  |
| --- | --- | --- | --- |
|  | If you need to run more than one daemon process, there's a [suggestion in the Docker documentation to use Supervisord](https://docs.docker.com/engine/admin/using_supervisord/) in an un-detached mode so all the sub-daemons will output to the stdout.  [From another SO question, I discovered you can redirect the child processes output to the stdout.](https://stackoverflow.com/a/26897796/918558)That way you can see all the output!  -🡪Another idea:  If, as in this case, you build the container just place a startup script in it and run this with command. Or mount the startup script as volume.  ---Yup, this too works! – [Bhargav Nanekalva](https://stackoverflow.com/users/1202145/bhargav-nanekalva)  -🡪Figured it out, use bash -c  example:  command: bash -c "python manage.py migrate && python manage.py runserver 0.0.0.0:8000"  ---didn't work for me. – Pedram Apr 5 '16 at 19:08  1  @ecoding5 Yes, I'm using version 2 – Pedram Apr 15 '16 at 19:20  3  @Pedram Make sure you are using an image that actually has bash installed. Some images may also require a direct path to bash e.g. /bin/bash – codemaven May 10 '16 at 20:57  2  When I do this the command gets interpreted like bash -c python instead of bash -c "python ...". Anybody else run into this? – Aaron Bruce Sep 20 '16 at 20:15  2  Alpine-based images actually seem to have no bash installed - do like @Chaoste recommends and use sh instead: [sh, -c, "cd /usr/src/app && npm start"] – Florian Loch Feb 20 at 23:02 26.) [How do links and scaling work together in docker compose?](https://stackoverflow.com/questions/29725955/how-do-links-and-scaling-work-together-in-docker-compose) Here's my understanding of docker compose:   * You can "scale" a service to run it in multiple containers at once. * If you link service A to service B, service A has access to a container running service B.   Is my understanding correct, and if so, where does a link connect if there are multiple containers running the service?  -🡪   |  |  | | --- | --- | |  | First of all I would clarify that, by default, with or without linking a container with other, all container has visibility to other containers running in the same host (using the container IP). You can change this behavior using the icc=true flag in docker daemon.  In respect of the links with docker-compose, these are generated when the container with the links are created. Let's see it with an example. Using this docker-compose.yml  web:  build: .  command: python app.py  ports:  - "5000:5000"  volumes:  - .:/code  links:  - redis  redis:  image: redis  After running docker-compose up -d the web\_1 container is linked with the container with redis\_1:  (...)  "Links": [  "/compose\_redis\_1:/compose\_web\_1/compose\_redis\_1",  "/compose\_redis\_1:/compose\_web\_1/redis",  "/compose\_redis\_1:/compose\_web\_1/redis\_1"  ], (...)  Now we want to scale the redis service using docker-compose scale redis=2. After running it (and create a new container redis\_2), the links in web\_1 keeps unchanged.  (...)  "Links": [  "/compose\_redis\_1:/compose\_web\_1/compose\_redis\_1",  "/compose\_redis\_1:/compose\_web\_1/redis",  "/compose\_redis\_1:/compose\_web\_1/redis\_1"  ], (...)  It is needed to stop, remove and run web\_1 to see these links created:  docker-compose stop web  docker-compose rm web  docker-compose run -d web  docker inspect compose\_web\_run\_2  (...) "Links": [  "/compose\_redis\_1:/compose\_web\_run\_2/compose\_redis\_1",  "/compose\_redis\_2:/compose\_web\_run\_2/compose\_redis\_2",  "/compose\_redis\_1:/compose\_web\_run\_2/redis",  "/compose\_redis\_1:/compose\_web\_run\_2/redis\_1",  "/compose\_redis\_2:/compose\_web\_run\_2/redis\_2"  ],(...)  And the /etc/hosts of web\_1 container:  172.17.0.24 7be2dabea910  127.0.0.1 localhost  ::1 localhost ip6-localhost ip6-loopback  fe00::0 ip6-localnet  ff00::0 ip6-mcastprefix  ff02::1 ip6-allnodes  ff02::2 ip6-allrouters  172.17.0.21 compose\_redis\_1 8a1297a5b3e4  172.17.0.23 compose\_redis\_2 069dd46836aa  172.17.0.21 redis 8a1297a5b3e4 compose\_redis\_1  172.17.0.21 redis\_1 8a1297a5b3e4 compose\_redis\_1  172.17.0.23 redis\_2 069dd46836aa compose\_redis\_2  **So to generate the new links, you is is needed to stop, remove, and run again the container.** |   -----So if service A is linked to service B, service A has access to ALL instances of service B? – Gaelan Apr 19 '15 at 16:38  Using the container's IPs has always access, but if you the access as with --link option (i.e.: with entries in /etc/hosts, global environment variables...) you would need to recreate service A container. – Javier Cortejoso Apr 19 '15 at 18:18  That's not my question. Is service A linked to ALL Instances of service B, or ONE instance of service B? – Gaelan Apr 19 '15 at  It's linked with ALL instances of service B, but each with different alias (e.g.: redis\_1 and redis\_2). Check my answer to see the /etc/hosts of the example – Javier Cortejoso Apr 19 '15 at 21:0  OK, that makes sense. Thanks! – Gaelan Apr 19 '15 at 21:02 27.) [Communication between multiple docker-compose projects](https://stackoverflow.com/questions/38088279/communication-between-multiple-docker-compose-projects) I have two separate docker-compose.yml files in two different folders:   * ~/front/docker-compose.yml * ~/api/docker-compose.yml   How can I make sure that a container in front can send requests to a container in api?  I know that --default-gateway option can be set using docker run for an individual container, so that a specific IP address can be assigned to this container, but it seems that this option is not available when using docker-compose.  Currently I end up doing a docker inspect my\_api\_container\_id and look at the gateway in the output. It works but the problem is that this IP is randomly attributed, so I can't rely on it.  Another form of this question might thus be:   * Can I attribute a fixed IP address to a particular container using docker-compose?   But in the end what I'm looking after is:   * How can two different docker-compose projects communicate with each other?   -🡪You just need to make sure that the containers you want to talk to each other are on the same network. Networks are a first-class docker construct, and not specific to compose.  # front/docker-compose.yml  version: '2'  services:  front:  ...  networks:  - some-net  networks:  some-net:  driver: bridge  ...  # api/docker-compose.yml  version: '2'  services:  api:  ...  networks:  - front\_some-net  networks:  front\_some-net:  external: true  They can then talk to each other using the service name. From front you can do ping api and vice versa.  -🡪Out of curiosity, wouldn't anything mapped to the host port (localhost), be visible to all the containers? – Robert Moskal Jun  And once they are ensured to be in the same network, as you suggest, what to do then? Can front container have a link to api container for instance? – Jivan Jun 29 '16 at 9:53  @JHarris well, if I do what you suggest, starting (up) ~/front triggers Some networks were defined but are not used by any service - while starting ~/api triggers Service "api" uses an undefined network – Jivan Jun 29 '16 at 12:25  Jivan that's a non-solution. Your containers should not need to know anything about the host or be manipulated like that. My answer was pretty short though, I have updated with more detail. – johnharris85 Jun 29 '16 at 17:10  Robert Moskal only if you hack around to get the ip of your docker host into the containers. Better to have them communicate on a common docker-defined network. – johnharris85 Jun 29 '16 at 17:15  -🡪The previous posts information is correct, but it does not have details on how to link containers, which should be connected as "external\_links".  Hope this example make more clear to you:   * Suppose you have app1/docker-compose.yml, with two services (svc11 and svc12), and app2/docker-compose.yml with two more services (svc21 and svc22) and suppose you need to connect in a crossed fashion: * svc11 needs to connect to svc22's container * svc21 needs to connect to svc11's container.   So the configuration should be like this:  **this is app1/docker-compose.yml:**  version: '2'  services:  svc11:  container\_name: container11  [..]  networks:  - default # this network  - app2\_default # external network  external\_links:  - container22:container22  [..]  svc12:  container\_name: container12  [..]  networks:  default: # this network (app1)  driver: bridge  app2\_default: # external network (app2)  external: true  **this is app2/docker-compose.yml:**  version: '2'  services:  svc21:  container\_name: container21  [..]  networks:  - default # this network (app2)  - app1\_default # external network (app1)  external\_links:  - container11:container11  [..]  svc22:  container\_name: container22  [..]  networks:  default: # this network (app2)  driver: bridge  app1\_default: # external network (app1)  external: true  -🡪All containers from api can join the front *default* network with following config:  # api/docker-compose.yml  ...  networks:  default:  external:  name: front\_default  See docker compose guide: [using a pre existing network](http://docs.docker.com/compose/networking/#using-a-pre-existing-network)  --🡪I would ensure all containers are docker-compose'd to the same network by composing them together at the same time, using:  docker compose --file ~/front/docker-compose.yml --file ~/api/docker-compose.yml up -d ---- Will that allow me, for instance, to make a link or depends\_on from one container of front to one container of api? – Jivan Jun 29 '16actually when I do what you suggest, docker-compose replies either build path ~/front/api either does not exist or is not accessible or with the other way around, build path ~/api/front either does not exist or is not accessible – Jivan Jun 29 '16 at 11:48If you're composing them at the same time you shouldn't need to. A network will be created with all your containers on it, they will all be able to communicate via the service name from the compose file (not the container name). – Nauraushaun Jun 29 '16 at 11:50It might be easier if the two compose files are in the same folder. But I don't think that's necessary - I think it should work either way. – Nauraushaun Jun 29 '16 at 11:52This solution does not work, see my comment on this thread: github.com/docker/compose/issues/3530#issuecomment-222490501 – johnharris85 |

-🡪Just a small adittion to @johnharris85's great answer, when you are running a docker compose file, a default network is created so you can just add it to the other compose file as an external network:

# front/docker-compose.yml

version: '2'

services:

front\_service:

...

...

# api/docker-compose.yml

version: '2'

services:

api\_service:

...

networks:

- front\_default

networks:

front\_default:

external: true

For me this approach was more suited because I did not own the first docker-compose file and wanted to communicate with it.

# 28.) [docker-compose up leads to “client and server don't have same version (client : 1.14, server: 1.12)” error but client and server have the same version](https://stackoverflow.com/questions/29727171/docker-compose-up-leads-to-client-and-server-dont-have-same-version-client)

docker version prints:

Client version: 1.0.1

Client API version: 1.12

Go version (client): go1.2.1

Git commit (client): 990021a

Server version: 1.0.1

Server API version: 1.12

Go version (server): go1.2.1

Git commit (server): 990021a

docker-compose --version prints:

docker-compose 1.2.0

I installed docker with apt-get install docker.io and docker-compose with

curl -L https://github.com/docker/compose/releases/download/1.2.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

I use Ubuntu 14.04 x64.

-🡪Checkout the environment variable [COMPOSE\_API\_VERSION](https://docs.docker.com/compose/reference/envvars/#/composeapiversion).

I was getting ERROR: client and server don't have same version (client : 1.19, server: 1.18), then I did export COMPOSE\_API\_VERSION=1.18 and problem sovled!

---Thanks, saved a lot of my time – punov Jan 25 '16 at 10:26

that fixed all the issue, thank you. – kawashita86 Mar 7 '16 at 13:26

-🡪Since docker-compose 1.4 you can set the client API version, you don't have to downgrade the client or upgrade the server.

You do not have to set a specific API version either, you can set the environment variable COMPOSE\_API\_VERSION=auto to have it auto-detect the client version

# 29.) [How to create a DB for MongoDB container on start up?](https://stackoverflow.com/questions/42912755/how-to-create-a-db-for-mongodb-container-on-start-up)

I am working with Docker and I have a stack with PHP, MySQL, Apache and Redis. I need to add MongoDB now so I was checking the [Dockerfile](https://github.com/docker-library/mongo/blob/4a81205a13fefc418355248f750551e4f7c62361/3.4/Dockerfile) for the latest version and also [the docker-entrypoint.sh](https://github.com/docker-library/mongo/blob/4a81205a13fefc418355248f750551e4f7c62361/3.4/docker-entrypoint.sh) file from the [MongoDB Dockerhub](https://hub.docker.com/r/library/mongo/) but I couldn't find a way to setup a default DB, admin user/password and possibly auth method for the container from a docker-compose.yml file.

In MySQL you can setup some ENV variables as for example:

db:

image: mysql:5.7

env\_file: .env

environment:

MYSQL\_ROOT\_PASSWORD: ${MYSQL\_ROOT\_PASSWORD}

MYSQL\_DATABASE: ${MYSQL\_DATABASE}

MYSQL\_USER: ${MYSQL\_USER}

MYSQL\_PASSWORD: ${MYSQL\_PASSWORD}

And this will setup the DB and the user/password as the root password.

Is there any way to achieve the same with MongoDB? Anyone has some experience or workaround?

-🡪an you create a container based on mysql and set it up as you want and then use it? – Valentin Mar 20 at 21:17

@Valentin of course I can but what is your point? – ReynierPM Mar 20 at 22:43

My point is, that you can set up a default DB, admin user/password and possibly auth method in dockerfile using variables and then pass them in compose file – Valentin Mar 20 at 22:53

-🡪

|  |  |
| --- | --- |
|  | The [official mongo image](https://hub.docker.com/r/library/mongo/) has [merged a PR to include the functionality](https://github.com/docker-library/mongo/pull/145) to create users and databases.  The database initialisation will run when there is nothing populated in the /data/db directory.  Admin User Setup  The environment variables to control "root" user setup are   * MONGO\_INITDB\_ROOT\_USERNAME * MONGO\_INITDB\_ROOT\_PASSWORD   Example  docker run -d \  -e MONGO\_INITDB\_ROOT\_USERNAME=admin \  -e MONGO\_INITDB\_ROOT\_PASSWORD=password \  mongod  You don't need to/can't use --auth on the command line as the docker entrypoint.sh script adds this in when the environment variables exist.  Other Custom Setup  The container also provides a path to deploy custom .js or .sh setup scripts that will be run once on database initialisation. .js scripts will be run against test by default or MONGO\_INITDB\_DATABASE if defined in the environment.  COPY mysetup.sh /docker-entrypoint-initdb.d/  or  COPY mysetup.js /docker-entrypoint-initdb.d/ |

---This is excellent, I suppose set -e is telling the entrypoint.sh to read variables from ENV definition so I can use the same approach as the example from MySQL, I am right? As an addition to your answer I did found this PR where apparently someone is working in something similar to your proposal just coming from another point of view – ReynierPM Mar 21 at 11:27

The environment variables you get by default. set -e causes the whole script to exit when a command fails, so the script can't silently fail and startup mongo. – Matt Mar 21 at 21:04

Similar to all the Dockerfile build scripts you see that use && everywhere. – Matt Mar 21 at 21:04

That mongo PR is a much more thorough implementation! and the docker-entrypoint-initdb.d directory makes it extensible. I hope that get's merged – Matt Mar 21 at 21:25

It's merged (see here) – ReynierPM Mar

# 30.) [Confusion on using docker-compose volumes to serve django static files](https://stackoverflow.com/questions/36722569/confusion-on-using-docker-compose-volumes-to-serve-django-static-files)

Docker rookie here, trying to setup a simple Django project using Compose. I've had [success with this in the past](https://stackoverflow.com/questions/32030360/volume-changes-not-persistent-after-docker-compose-run-command-djangos-colle), but I'm trying a different setup this time, which I can't figure out why it doesn't work.

I have the following docker-compose.yml file:

data:

image: postgres:9.4

volumes:

- /var/lib/postgresql

command: /bin/true

db:

image: postgres:9.4

restart: always

expose:

- "5432"

volumes\_from:

- data

app:

build: .

restart: always

env\_file: .env

expose:

- "8000"

links:

- db:db

volumes:

- .static:/static\_files

web:

build: docker/web

restart: always

ports:

- "80:80"

links:

- app:app

volumes\_from:

- app

My /Dockerfile is:

FROM python:3.5

ENV PYTHONUNBUFFERED 1

ADD . /app

WORKDIR /app

RUN pip install -r requirements.txt

RUN SECRET\_KEY=tmpkey python manage.py collectstatic --noinput

CMD gunicorn mysite.wsgi:application -b 0.0.0.0:8000

My /docker/web/Dockerfile is:

FROM nginx:1.9

RUN rm /etc/nginx/conf.d/default.conf

ADD default.conf /etc/nginx/conf.d/

And my /docker/web/default.conf file is:

server {

listen 80 default\_server;

location /static/ {

autoindex on;

alias /static\_files/;

}

location / {

proxy\_pass http://app:8000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

}

The output from docker shows the static files installing into /static\_files, but nginx returns 404 for all the files under /static. If I look in my .static folder (in root of project), the directory is empty (aside from a .gitkeep file I have in there). If I run docker-compose app run ls -la /static\_files, the directory is empty, but docker-compose app run ls -la /app/.static has the .gitkeep file. Clearly I'm misunderstanding something with Docker and Compose. Any thoughts on what I'm doing wrong? My understanding is that the RUN SECRET\_KEY=tmpkey python manage.py collectstatic --noinput should be writing files to my local .static folder and that nginx should see these files; neither is happening.

Software versions: docker-compose version 1.7.0, build 0d7bf73 and Docker version 1.11.0, build 4dc5990 on OS X, with docker-machine connected to cloud instance.

-🡪I am still unclear of why my original code does not work, but switching my code to use Compose's v2 format works, where I have my volumes defined outside of services.

Here's my updated docker-compose.yml file:

version: '2'

services:

db:

image: postgres:9.4

restart: always

expose:

- "5432"

volumes:

- postgres-data:/var/lib/postgresql

app:

build: .

restart: always

env\_file: .env

expose:

- "8000"

links:

- db:db

volumes:

- static-data:/static\_files

web:

build: docker/web

restart: always

ports:

- "80:80"

links:

- app:app

volumes:

- static-data:/static\_files

volumes:

postgres-data:

driver: local

static-data:

driver: local

The rest of the config files remained the same.

(It may be worth noting that before I ran this new config, I deleted all existing Docker volumes listed in docker volume ls -- perhaps this was my actual fix?)

# 31.) [`data-container` with named or anonymous volumes - conceptual problems? (Discussion)](https://stackoverflow.com/questions/39013222/data-container-with-named-or-anonymous-volumes-conceptual-problems-discuss)

a) Anonymous volumes

When using data-containers, you can either use anonymous volumes like this

version '2'

services:

consumer:

volume\_from:

- data-container:rw

data-container:

image: cogniteev/echo

command: echo 'Data Container'

volume:

- /var/www

b) Name volumes

or you can use named volumes like this

version '2'

services:

consumer:

volume\_from:

- data-container:rw

data-container:

image: cogniteev/echo

command: echo 'Data Container'

volume:

- my-named-volume:/var/www

volumes:

my-named-volume:

driver: local

I usually go with b) and would like to discuss / get explained the conceptual issues / flaws of maybe both of those. So what are the pros and cons.

The aspects we can compare them against are / could be:

1. portability
2. upgradeability of the data-container (why would we ever upgrade the container ? )
3. Start/Stop (continue) compatibility?
4. multiple-stack issues?
5. efficiency ( reuse of the volume )

This question spwaned du to the discussion on this question <https://stackoverflow.com/a/38984689/3625317>

---Named volumes were created to replace data containers. I don't know of any use cases where data containers are preferred. See [this similar (dup?) question](http://stackoverflow.com/q/36011595/596285) –

🡪Short answer: named data volumes are preferred, data containers are no longer needed, so you should never use volumes-from on any new project.

Your version of named volumes is merging a named and data container, it should be:

version '2'

services:

web:

image: my-web-image

volumes:

- my-named-volume:/var/www

volumes:

my-named-volume:

driver: local

By merging the two, you've added an extra layer of indirection to reach your named volume, without any added benefits. Named volumes were created in 1.9 to [replace data containers](https://github.com/docker/docker/issues/17798) which were themselves a somewhat hacked method to provide persistent data. Advantages of named volumes over data containers include:

* Your data management is separate from your container management, you can remove all running containers and still have your data available
* Data can be stored in different locations using volume drivers, which means you can put it on nfs, a distributed file system, or even a local persistent directory
* You may start and stop any container in any order without dependencies between containers
* When first created, a named volume will receive a copy of the image filesystem it is first mounted on top of, identical to the behavior of data containers, which means it's a seamless transition (note that this is not the behavior of a host volume, aka bind mount)

See also [this question that also discusses named volumes vs data containers](https://stackoverflow.com/q/36011595/596285) and [this answer](https://serverfault.com/a/760244/351549) to another similar question. We also have a [blog post on this](https://boxboat.com/2016/06/18/docker-data-containers-and-named-volumes/) by a company that I work for.

----thank you very much, sir. Those are really valuable informations, esp. that it is actually intended to replace data-containers since 1.9. – Eugen Mayer Aug 18 '16 at 14:10

For further reference, also the docs got updated to remove the mention that data-containers are best-practice github.com/docker/docker/issues/20465 - so this all seconds your answer – Eugen Mayer Aug 18 '16 at 14:13

They need to be updated still, that issue is open. Might be time for me to check out the code and make a pull request. – BMitch Aug 18 '16 at 15:01

Yes, you are right, docs.docker.com/engine/tutorials/dockervolumes/#/… is still present and not yet updated .. created a comment github.com/docker/docker/issues/20465#issuecomment-240756653 but i guess, that will not help the progress. Seeing how much confusion this topic spreads, a best-practice guide should be really something to strive for – Eugen Mayer Aug 18 '16 at 15:19

# 32.) [connecting to a docker-compose mysql container denies access but docker running same image does not](https://stackoverflow.com/questions/37459031/connecting-to-a-docker-compose-mysql-container-denies-access-but-docker-running)

I am having some issues connecting to the mysql docker container that I have launched with docker-compose. This is a long post (sorry!).

Here is my docker-compose.yml file:

db:

image: mysql:5.7

ports:

- "3306:3306" # I have tried both ports and expose "3306". Still doesn't work

environment:

- MYSQL\_ROOT\_PASSWORD="secret"

- MYSQL\_USER="django"

- MYSQL\_PASSWORD="secret"

- MYSQL\_DATABASE="myAppDB"

Then:

$> docker-compose build

db uses an image, skipping #expected!

$> docker-compose up

<<LOTS OF OUTPUT>>

OK, so now I have an up and running docker container runner mysql:5.7. Great! Or is it? When testing in my django app, I get Operational errors saying that the user isn't allowed to connect the database. Ok, so maybe it's my django then?

$> docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

c7216f99ca0f mysql:5.7 "docker-entrypoint.sh" 3 minutes ago Up 3 minutes 0.0.0.0:3306->3306/tcp sharpfin\_db\_1

$> docker-machine ip dev

192.168.99.100

$> mysql -h 192.168.99.100 -P 3306 -u django -p

Enter password:

ERROR 1045 (28000): Access denied for user 'django'@'192.168.99.1' (using password: YES)

ok so maybe It's something to do with connecting to the docker-compose container? What if I try connecting from inside the docker container?

$> docker exec -it c7216f99ca0f /bin/bash

root@c7216f99ca0f:/#

root@c7216f99ca0f:/# mysql -u django -p

Enter password:

ERROR 1045 (28000): Access denied for user 'django'@'localhost' (using password: YES)

ok, so docker mysql won't let me connect, don't know why. Let's see what happens when I try do this without docker-compose:

$> docker run --name run-mysql -e MYSQL\_ROOT\_PASSWORD="secret" -e MYSQL\_USER="django" -e MYSQL\_PASSWORD="secret" -e MYSQL\_DATABASE="myAppDB" -p "3306:3306" mysql:5.7

<<LOTS OF OUTPUT SAME AS BEFORE>>

Ok, so now we have a container running the same image as before with the same settings. (I think this assertion is probably not true - docker-compose is doing something different to docker run).

$> docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

73071b929e82 mysql:5.7 "docker-entrypoint.sh" 3 minutes ago Up 3 minutes 0.0.0.0:3306->3306/tcp run-mysql

There's my container (called run-mysql). Let's connect!

$> mysql -h 192.168.99.100 -P 3306 -u django -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 2

Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| myAppDB |

+--------------------+

2 rows in set (0.01 sec)

mysql>

Alright. Can log in. That's weird... What about from inside the container?

$> docker exec -it 73071b929e82 /bin/bash

root@73071b929e82:/# mysql -u django -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 3

Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| myAppDB |

+--------------------+

2 rows in set (0.00 sec)

mysql>

Ok, I can log in from outside and inside the container when I launch with docker run, but not with docker-compose. What's going on? There must be something either docker-compose is doing behind the scenes that changes how the database is initialized.

All the above is the exact same if I try with the root user as well. So it's not a permissions issue with the django user.

Any ideas how to resolve this?

-🡪Environment variables in docker-compose.yml file should not have quotes when using array definition:

db:

image: mysql:5.7

ports:

- "3306:3306"

environment:

- MYSQL\_ROOT\_PASSWORD=secret

- MYSQL\_USER=django

- MYSQL\_PASSWORD=secret

- MYSQL\_DATABASE=myAppDB

If you use them in your docker-compose.yml file:

db:

image: mysql:5.7

ports:

- "3306:3306"

environment:

- MYSQL\_ROOT\_PASSWORD="secret"

- MYSQL\_USER="django"

- MYSQL\_PASSWORD="secret"

- MYSQL\_DATABASE="myAppDB"

and run:

$ docker-compose up -d

and enter running container:

$ docker-compose exec -it db /bin/bash

you will see the output:

root@979813643b0c:/# echo $MYSQL\_ROOT\_PASSWORD

"secret"

-🡪t looks like your problem is solved. I thought I'd discuss my problems similar to this.

I am running tomcat (web) and mysql (db) using docker-compose on a Synology NAS (DSM 6.0.2). It worked fine on an Ubuntu box I have but not on the NAS.

The problem was the firewall on the NAS - I had modified my firewall rules to allow certain ports open but then DENY ALL at end. When I added :3306 to the allowed ports it worked!

This is not a good solution and I don't know why DSM would require this since the docker-compose is running on a BRIDGE network. I've put in a support ticket about this.

This answer may help others with this blocked container issue.

-🡪

|  |  |
| --- | --- |
|  | I had a similar issue, and this helped me:  <https://github.com/docker-library/mysql/issues/51#issuecomment-76989402>  Have you changed the passwords since you first tried running the containers? docker-compose does extra work to preserve volumes between runs (thus preserving the database); you may want to try docker-compose rm -v to delete everything and try starting it up again. |

-🡪I am using the official mysql image with docker-compose and not having a problem. The only difference in my compose file is that I am using a dictionary instead of an array:

environment:

MYSQL\_ROOT\_PASSWORD: secret

MYSQL\_USER: django

MYSQL\_PASSWORD: secret

MYSQL\_DATABASE: myAppDB

I have noticed that the compose file documentation is still stuck in V1 in some places, so you could try this, if you're using V2. Otherwise, for debugging you can use docker-compose exec to interact with the container created by compose directly.

docker-compose exec db /bin/bash will get you a shell on the container that is giving you trouble and you can check things like SHOW GRANTS FOR django@'%' or whether the ports are being forwarded correctly. I hope this helps.

--🡪It was EXACTLY the problem of dictionary vs array. Thank you!!!!!! I have no idea why this is the case though. And silent erroring as well. – Davy Kavanagh May 26 '16 at 12:44

That's a good point, an invalid config should throw an error. The docs say that you can use either - I was assuming a bug in the documentation. – threeve May 26 '16 at 13:32

In general I think yaml isn't the most intuitive format to provide configuration options. Long live JSON!. I am now having problems with the links parameter (I think) probably for the same reason. – Davy Kavanagh May 26 '16 at 13:51

When using dict, I am getting: ERROR: yaml.scanner.ScannerError: mapping values are not allowed here – NeverEndingQueue Jan 29 at 17:51

3

@DavyKavanagh The issue is not array vs dictionary, but that in your array defintion, there must not be quotes as those become part of the secret. Hence: stackoverflow.com/a/41282871/457268 should be the accepted answer IMO – k0pernikus Feb 27 at 12:30

# 33.) [how do you manage secret values with docker-compose v3.1?](https://stackoverflow.com/questions/42139605/how-do-you-manage-secret-values-with-docker-compose-v3-1)

Version 3.1 of the docker-compose.yml specification introduces support for [secrets](https://docs.docker.com/engine/reference/commandline/secret/).

I tried this:

version: '3.1'

services:

a:

image: tutum/hello-world

secret:

password: the\_password

b:

image: tutum/hello-world

$ docker-compose up returns:

Unsupported config option for services.secret: 'password'

How can we use the secrets feature in practice?

--Are you sure docker-compose does already support secrets? Which docker-compose version are you running? – fzgregor Feb 9 at 14:54

$ docker-compose --version returns: docker-compose version 1.11.0, build 6de1806, so yes, it should support secrets according to the release notes. – Eric Feb 9 at 14:59

-🡪You can read the [corresponding section from the official documentation](https://docs.docker.com/compose/compose-file/#secrets-configuration-reference).

To use secrets you need to add two things into your docker-compose.yml file. First, a top-level secrets: block that defines all of the secrets. Then, another secrets: block under each service that specifies *which* secrets the service should receive.

As an example, create the two types of secrets that Docker will understand: **external** secrets and **file** secrets.

1. Create an 'external' secret using docker secret create

First thing: to use secrets with Docker, the node you are on must be part of a swarm.

$ docker swarm init

Next, create an 'external' secret:

$ echo "This is an external secret" | docker secret create my\_external\_secret -

(Make sure to include the final dash, -. It's easy to miss.)

2. Write another secret into a file

$ echo "This is a file secret." > my\_file\_secret.txt

3. Create a docker-compose.yml file that uses both secrets

Now that both types of secrets are created, here is the docker-compose.yml file that will read both of those and write them to the web service:

version: '3.1'

services:

web:

image: nginxdemos/hello

secrets: # secrets block only for 'web' service

- my\_external\_secret

- my\_file\_secret

secrets: # top level secrets block

my\_external\_secret:

external: true

my\_file\_secret:

file: my\_file\_secret.txt

Docker can read secrets either from its own database (e.g. secrets made with docker secret create) or from a file. The above shows both examples.

4. Deploy your test stack

Deploy the stack using:

$ docker stack deploy --compose-file=docker-compose.yml secret\_test

This will create one instance of the web service, named secret\_test\_web.

5. Verify that the container created by the service has both secrets

Use docker exec -ti [container] /bin/sh to verify that the secrets exist.

(Note: in the below docker exec command, the m2jgac... portion will be different on your machine. Run docker ps to find your container name.)

$ docker exec -ti secret\_test\_web.1.m2jgacogzsiaqhgq1z0yrwekd /bin/sh

# Now inside secret\_test\_web; secrets are contained in /run/secrets/

root@secret\_test\_web:~$ cd /run/secrets/

root@secret\_test\_web:/run/secrets$ ls

my\_external\_secret my\_file\_secret

root@secret\_test\_web:/run/secrets$ cat my\_external\_secret

This is an external secret

root@secret\_test\_web:/run/secrets$ cat my\_file\_secret

This is a file secret.

If all is well, the two secrets we created in steps 1 and 2 should be inside the web container that was created when we deployed our stack.

---I get this result when I run the first command you specify: Error response from daemon: This node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again. I'm confused! Do I need to start a swarm in order to use docker-compose with secrets? – Eric Feb 10 at

1

Oops, yes you do. The docker stack deploy command is part of the Swarm engine. I'll add a line in Step 1 to indicate that. – Mike Hearn Feb 10 at

I don't understand the reasoning behind that. Mine and many other use cases don't require a swarm, but do require secrets. Why make us use a swarm if we just want secrets? – Eric Feb 13 at 10:32

@Eric If you don't require swarm, you could just mount the secret file from your local machine. Why do you need docker secrets? – Vanuan Feb 14 at

@Vanuan because I need secrets to start containers on my remote machine, not just my local machine. for example, the official owncloud image has a docker-compose.yml that asks you to write down the MySQL password as an environment variable, which is bad practice. I thought docker secrets would solve that?

--🡪You can also specify secrets stored locally in a file using file: key in secrets object. Then you don't have to docker secret create them yourself, Compose / docker stack deploy will do it for you.

version: '3.1'

secrets:

password:

file: ./password

services:

password\_consumer:

image: alpine

secrets:

- password

Reference: [Compose file version 3 reference: Secrets](https://docs.docker.com/compose/compose-file/#secrets)

-🡪

|  |  |
| --- | --- |
| -🡪 | I You can also specify secrets stored locally in a file using file: key in secrets object. Then you don't have to docker secret create them yourself, Compose / docker stack deploy will do it for you.  version: '3.1'  secrets:  password:  file: ./password  services:  password\_consumer:  image: alpine  secrets:  - password  Reference: [Compose file version 3 reference: Secrets](https://docs.docker.com/compose/compose-file/#secrets)  guess the keyword is secrets not secret. That is at least what I understand from reading the [schema](https://github.com/docker/compose/blob/c9eb9380ed58992b1695f848fb68eba5f41bc784/compose/config/config_schema_v3.1.json). |

-🡪Is that the exact indentation of your docker-compose.yml file? I think ~~secret~~ secrets should be nested under a (i.e. one of the services), not directly under services section.

-🡪Given you have a service myapp and a secrets file secrets.yml:

Create a compose file:

version: '3.1'

services:

myapp:

build: .

secrets:

secrets\_yaml

Provision a secret using this command:

docker secret create secrets\_yaml secrets.yml

Deploy your service using this command:

docker deploy --compose-file docker-compose.yml myappstack

Now your app can access the secret file at /run/secrets/secrets\_yaml. You can either hardcode this path in your application or create a symbolic link.

**The different question**

This answer is probably to the question "how do you provision your secrets to your docker swarm cluster".

The original question "how do you manage secret values with docker compose" implies that the docker-compose file contains secret values. It doesn't.

There's a different question: "Where do you store the canonical source of the secrets.yml file". This is up to you. You can store it in your head, print on a sheet of paper, use a password manager, use a dedicated secrets application/database. Heck, you can even use a git repository if it's safely secured itself. Of course, never store it inside the system you're securing with it :)

I would recommend [vault](https://www.vaultproject.io/). To store a secret:

# create a temporary secret file

cat secrets.yml | vault write secret/myappsecrets -

To retrieve a secret and put it into your docker swarm:

vault read -field=value secret/myappsecrets | docker secret create secrets\_yaml -

Of course, you can use docker cluster itself as a single source of truth for you secrets, but if your docker cluster breaks, you'd lost your secrets. So make sure to have a backup elsewhere.

**The question nobody asked**

The third question (that nobody asked) is how to provision secrets to developers' machines. It might be needed when there's an external service which is impossible to mock locally or a large database which is impossible to copy.

Again, docker has nothing to do with it (yet). It doesn't have access control lists which specify which developers have access to which secrets. Nor does it have any authentication mechanism.

The ideal solution appears to be this:

* A developer opens some web application.
* Authenticates using some single sign on mechanism.
* Copies some long list of docker secret create commands and executes them in the terminal.

We have yet to see if such an application pops up.

----docker secret create seems to require that there be a pre-existing swarm? do I need to create one? – Eric Feb 10 at 13:33

@Eric So you're running this as a developer? I'm afraid Docker doesn't have support for that use case yet. But yeah, you could create a docker swarm consisting of only your developer machine. That's out of scope of this question. – Vanuan Feb 10 at 20:15

As of Feb 8th docker-compose 1.11 supports file-based secrets in compose files for local dev. See my comment above on chosen

@BretFisher but what's the point if it's essentially the same as specifying using volume's ./file\_based\_secret:/run/secrets/my\_secret ?

@Vanuan Right, there's no functional difference in container. It's about seamless workflow, and limiting the need for multiple compopse files. – Bret Fisher Mar 18 at 15:34

# 34.) [Using Docker I get the error: “SQLSTATE[HY000] [2002] No such file or directory”](https://stackoverflow.com/questions/40075065/using-docker-i-get-the-error-sqlstatehy000-2002-no-such-file-or-directory)

I'm using Docker to create a container to test my web app built on PHP and MySQL on my Mac. My PHP app is built using Fat-Free Framework for MVC and routing. I have two Dockerfiles, one for MySQL and one for PHP. I've used test Docker applications successfully, so I believe my images are installed correctly.

The main part of the error:

Internal Server Error

SQLSTATE[HY000] [2002] No such file or directory

[fatfree/lib/DB/SQL.php:466] PDO->\_\_construct('mysql:host=127.0.0.1;port=3306;dbname=robohome','root','password',array(1002=>'SET NAMES utf8;'))

[fatfree/app/Controllers/Controller.php:24] DB\SQL->\_\_construct('mysql:host=127.0.0.1;port=3306;dbname=robohome','root','password')

Note, if I connect using 127.0.0.1 instead of localhost I get a slightly different error that says: SQLSTATE[HY000] [2002] Connection refused

My PHP Dockerfile:

FROM php:5.6-apache

RUN docker-php-ext-install mysqli pdo pdo\_mysql

RUN a2enmod rewrite

My MySQL Dockerfile:

FROM mysql:5.7

ENV MYSQL\_ROOT\_PASSWORD password

ENV MYSQL\_DATABASE robohome

COPY ./schema.sql /docker-entrypoint-initdb.d/

My Controller.php file where the error mentions line 24:

<?php

namespace Controllers;

class Controller

{

protected $f3;

protected $db;

public function \_\_construct()

{

$f3 = \Base::instance();

$this->f3 = $f3;

$mysqlServerName = $f3->get("MYSQL\_SERVERNAME");

$mysqlDatabseName = $f3->get("MYSQL\_DBNAME");

//$container = \DI\ContainerBuilder::buildDevContainer(); <-Not used currently

//Below is line 24 referred to in the error

$db = new \DB\SQL(

"mysql:host={$mysqlServerName};port=3306;dbname={$mysqlDatabseName}",

$f3->get("MYSQL\_USERNAME"),

$f3->get("MYSQL\_PASSWORD")

);

$this->db = $db;

}

Those MYSQL\_\* values are pulled from an .ini file:

MYSQL\_SERVERNAME = "localhost" <-This is what I've tried changing to 127.0.0.1

MYSQL\_USERNAME = "root"

MYSQL\_PASSWORD = "password"

MYSQL\_DBNAME = "robohome"

My Docker compose file:

version: '2'

services:

web:

build: ./docker/php

ports:

- 80:80

volumes:

- .:/var/www/html/

links:

- db

db:

build: ./docker/mysql

ports:

- 3306

I run this by doing docker-compose up --build -d. The output I can then get from docker ps is:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f35066a16586 robohomeweb\_mysql "docker-entrypoint.sh" 3 minutes ago Up 2 seconds 0.0.0.0:32777->3306/tcp robohomeweb\_mysql\_1

86d34eb34583 robohomeweb\_php "apache2-foreground" 3 minutes ago Up 2 seconds 0.0.0.0:80->80/tcp robohomeweb\_php\_1

If I run in the foreground instead, I get the following output:

Building php

Step 1 : FROM php:5.6-apache

---> 8f9b7e57129a

Step 2 : RUN docker-php-ext-install mysqli pdo pdo\_mysql

---> Using cache

---> fadd8f9e7207

Step 3 : RUN a2enmod rewrite

---> Using cache

---> 9dfed7fdc60f

Successfully built 9dfed7fdc60f

Building mysql

Step 1 : FROM mysql:5.7

---> eda6a4884645

Step 2 : ENV MYSQL\_ROOT\_PASSWORD password

---> Using cache

---> 759895ac5772

Step 3 : ENV MYSQL\_DATABASE robohome

---> Using cache

---> e926c5ecc088

Step 4 : COPY ./schema.sql /docker-entrypoint-initdb.d/

---> Using cache

---> cf5d00aa8020

Successfully built cf5d00aa8020

Starting robohomeweb\_php\_1

Starting robohomeweb\_mysql\_1

Attaching to robohomeweb\_mysql\_1, robohomeweb\_php\_1

php\_1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive globally to suppress this message

php\_1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive globally to suppress this message

php\_1 | [Sun Oct 16 20:21:17.944575 2016] [mpm\_prefork:notice] [pid 1] AH00163: Apache/2.4.10 (Debian) PHP/5.6.26 configured -- resuming normal operations

php\_1 | [Sun Oct 16 20:21:17.946919 2016] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'

mysql\_1 | 2016-10-16T20:21:18.036272Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit\_defaults\_for\_timestamp server option (see documentation for more details).

mysql\_1 | 2016-10-16T20:21:18.038330Z 0 [Note] mysqld (mysqld 5.7.16) starting as process 1 ...

mysql\_1 | 2016-10-16T20:21:18.043331Z 0 [Note] InnoDB: PUNCH HOLE support available

mysql\_1 | 2016-10-16T20:21:18.043603Z 0 [Note] InnoDB: Mutexes and rw\_locks use GCC atomic builtins

mysql\_1 | 2016-10-16T20:21:18.043951Z 0 [Note] InnoDB: Uses event mutexes

mysql\_1 | 2016-10-16T20:21:18.044077Z 0 [Note] InnoDB: GCC builtin \_\_atomic\_thread\_fence() is used for memory barrier

mysql\_1 | 2016-10-16T20:21:18.044260Z 0 [Note] InnoDB: Compressed tables use zlib 1.2.3

mysql\_1 | 2016-10-16T20:21:18.044414Z 0 [Note] InnoDB: Using Linux native AIO

mysql\_1 | 2016-10-16T20:21:18.045150Z 0 [Note] InnoDB: Number of pools: 1

mysql\_1 | 2016-10-16T20:21:18.045620Z 0 [Note] InnoDB: Using CPU crc32 instructions

mysql\_1 | 2016-10-16T20:21:18.047629Z 0 [Note] InnoDB: Initializing buffer pool, total size = 128M, instances = 1, chunk size = 128M

mysql\_1 | 2016-10-16T20:21:18.057705Z 0 [Note] InnoDB: Completed initialization of buffer pool

mysql\_1 | 2016-10-16T20:21:18.059988Z 0 [Note] InnoDB: If the mysqld execution user is authorized, page cleaner thread priority can be changed. See the man page of setpriority().

mysql\_1 | 2016-10-16T20:21:18.074670Z 0 [Note] InnoDB: Highest supported file format is Barracuda.

mysql\_1 | 2016-10-16T20:21:18.101209Z 0 [Note] InnoDB: Creating shared tablespace for temporary tables

mysql\_1 | 2016-10-16T20:21:18.101433Z 0 [Note] InnoDB: Setting file './ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...

mysql\_1 | 2016-10-16T20:21:18.354806Z 0 [Note] InnoDB: File './ibtmp1' size is now 12 MB.

mysql\_1 | 2016-10-16T20:21:18.356928Z 0 [Note] InnoDB: 96 redo rollback segment(s) found. 96 redo rollback segment(s) are active.

mysql\_1 | 2016-10-16T20:21:18.357158Z 0 [Note] InnoDB: 32 non-redo rollback segment(s) are active.

mysql\_1 | 2016-10-16T20:21:18.358049Z 0 [Note] InnoDB: Waiting for purge to start

mysql\_1 | 2016-10-16T20:21:18.412987Z 0 [Note] InnoDB: 5.7.16 started; log sequence number 12179647

mysql\_1 | 2016-10-16T20:21:18.414470Z 0 [Note] Plugin 'FEDERATED' is disabled.

mysql\_1 | 2016-10-16T20:21:18.421833Z 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib\_buffer\_pool

mysql\_1 | 2016-10-16T20:21:18.424144Z 0 [Note] InnoDB: Buffer pool(s) load completed at 161016 20:21:18

mysql\_1 | 2016-10-16T20:21:18.425607Z 0 [Warning] Failed to set up SSL because of the following SSL library error: SSL context is not usable without certificate and private key

mysql\_1 | 2016-10-16T20:21:18.427018Z 0 [Note] Server hostname (bind-address): '\*'; port: 3306

mysql\_1 | 2016-10-16T20:21:18.427581Z 0 [Note] IPv6 is available.

mysql\_1 | 2016-10-16T20:21:18.427749Z 0 [Note] - '::' resolves to '::';

mysql\_1 | 2016-10-16T20:21:18.428019Z 0 [Note] Server socket created on IP: '::'.

mysql\_1 | 2016-10-16T20:21:18.456023Z 0 [Warning] 'db' entry 'sys mysql.sys@localhost' ignored in --skip-name-resolve mode.

mysql\_1 | 2016-10-16T20:21:18.456354Z 0 [Warning] 'proxies\_priv' entry '@ root@localhost' ignored in --skip-name-resolve mode.

mysql\_1 | 2016-10-16T20:21:18.480237Z 0 [Warning] 'tables\_priv' entry 'sys\_config mysql.sys@localhost' ignored in --skip-name-resolve mode.

mysql\_1 | 2016-10-16T20:21:18.488758Z 0 [Note] Event Scheduler: Loaded 0 events

mysql\_1 | 2016-10-16T20:21:18.490880Z 0 [Note] mysqld: ready for connections.

mysql\_1 | Version: '5.7.16' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server (GPL)

From my research, I've tried connecting using both localhost and 127.0.0.1 since they're technically treated differently. It could also be something related to trying to talk via sockets instead of TCP. Ideally I would like a solution that I can bake into my Dockerfiles so I don't have to worry about remembering commands or how I did something.

---->--

Well, what is mysql set up to listen on? Port 3306 (default for TCP) or a socket file? – Devon Oct 16 '16 at 20:25

Based on my comment above (you have to scroll right to see it) 0.0.0.0:32777->3306/tcp – roundtheworld Oct 16 '16 at 21:39

Edited to include my Docker compose file. – roundtheworld Oct 16 '16 at 22:43

-🡪As someone pointed out in the comments, the docker-compose file you provided is very relevant to your question.

The documentation for [links](https://docs.docker.com/compose/compose-file/#links) in docker-compose files says

Containers for the linked service will be reachable at a hostname identical to the alias, or the service name if no alias was specified.

In your case, the database container is named db, so resolving db host from the PHP container should point you at the MySQL container. Replacing localhost with db in your config file should allow the PHP container to connect to MySQL.

---That was the piece of information that I was missing/didn't understand; that the location of my database needs to match its name in my docker-compose file. I got it working! – roundtheworld Oct 17 '16 at 2:09

I also wanted to add that I don't need the links section. My problem was 100% that fact that the location of the database in my config file was localhost when it should have been the name of my service in my yml file db. – roundtheworld Oct 17 '16 at 22:34

1

@roundtheworld That's true, (fairly) recently, Docker gained support for basic DNS that allows containers to find each other by name. – R0MANARMY Oct 17 '16 at

--🡪php lives on a different docker image than mysql. thus localhost and 127.0.0.1 from php do not point to mysql. you should connect to the ip of the mysql docker instance.

also make sure mysql is listening on all interfaces. In mysql.ini you need to put listen 0.0.0.0 to listen on all available interfaces. By default it only allow connections from localhost (and the php docker container is a different host).

-----Don't publish unless you need it elsewhere, just another endpoint you need to secure. Link them together. docs.docker.com/compose/compose-file/#links – user2105103 Oct 16 '16 at 21:38

I understand that php and mysql live in different Docker images, but for sample's I found online that I built off of, I didn't see anything that did that. I was able to get this example to work which is very similar to mine: github.com/mikesir87/docker-compose-demo – roundtheworld Oct 16 '16 at 21:41

I updated the response. Linking the containers together is what you want, then reference by link name. – user2105103 Oct 16 '16 at 21:42

Also the docker-compose file is more relevant to your question than almost everything else supplied ;) But, that is where you want to spin them both up and link 'em. – user2105103 Oct 16 '16 at 21:44

Yeah, I've only been playing with Docker for about a week, so I'm pretty inexperienced. If I understood your comment with correctly in my .yml file I added links: - mysql to the end and it didn't work. I tried linking the php to the mysql as well, same result. I didn't think it was necessary to link things in .yml files. – roundtheworld Oct 16 '16 at 21:56

# 35.) [chown docker volumes on host (possibly through docker-compose)](https://stackoverflow.com/questions/36312699/chown-docker-volumes-on-host-possibly-through-docker-compose)

 have the following example

version: '2'

services:

proxy:

container\_name: proxy

hostname: proxy

image: nginx

ports:

- 80:80

- 443:443

volumes:

- proxy\_conf:/etc/nginx

- proxy\_htdocs:/usr/share/nginx/html

volumes:

proxy\_conf: {}

proxy\_htdocs: {}

which works fine. When I run docker-compose up it creates those named volumes in /var/lib/docker/volumes and all is good. However, from the host, I can only access /var/lib/docker as root, because it's root:root (makes sense). I was wondering if there is a way of chowning the host's directories to something more sensible/safe (like, my relatively unprivileged user that I use to do most things on the host) or if I just have to suck it up and chownthem manually. I'm starting to have a number of scripts already to work around other issues, so having an extra couple of lines won't be much of a problem, but I'd really like to keep my self-written automation minimal, if I can -- fewer chances for stupid mistakes.

By the way, no: if I mount host directories instead of creating volumes, they get overlaid, meaning that if they start empty, they stay empty, and I don't get the default configuration (or whatever) from inside the container.

Extra points: can I just move the volumes to a more convenient location? Say, /home/myuser/myserverstuff/volumes?

-🡪

|  |  |
| --- | --- |
|  | It's best to not try to access files inside /var/lib/docker directly. Those directories are meant to be managed by the docker daemon, and not to be messed with.  To access the data inside a volume, there's a number of options;   * use a bind-mounted directory (you considered that, but didn't fit your use case). * use a "service" container that uses the same volume and makes it accessible through that container, for example a container running ssh (to use scp) or a SAMBA container (such as [svendowideit/samba](https://hub.docker.com/r/svendowideit/samba/)) * use a [volume-driver plugin](https://docs.docker.com/engine/extend/plugins/). there's various plugins around that offer all kind of options. For example, the [local persist plugin](https://github.com/CWSpear/local-persist) is a really simple plug-in that allows you to specify wheredocker should store the volume data (so outside of /var/lib/docker) |

---Thanks, the local persist plugin might in fact be exactly what I need. I had thought of using a service container, and export volumes through nfs, but it seemed to be way too much convoluted for my needs. – Morpheu5 Mar 31 '16 at 11:10

1

Great! I hope it works out for you. There's no "one size fits all", so I thought "let's mention some options to pick from" :-) – thaJeztah Mar 31 '16 at

I also thought of the NFS/SMB export thing..I like the idea of mounting those volumes remotely on my laptop and using my favourite editor/file manager, but then NFS over SSH is painful, I have no use for Kerberos, and I'm not sure SMB offers any encryption (and access control is no less messy than Kerberos). – Morpheu5

# 36.) [Docker nginx-proxy : proxy between containers](https://stackoverflow.com/questions/40293757/docker-nginx-proxy-proxy-between-containers)

am currently running a development stack using Docker-Compose in my company, to provide to developers everything they need to code our applications.

It includes in particular:

* a Gitlab container ([sameersbn/gitlab](https://github.com/sameersbn/docker-gitlab)) to manage private GIT repositories,
* a Jenkins container ([library/jenkins](https://hub.docker.com/_/jenkins/)) for building and continuous integration,
* an Archiva container ([ninjaben/archiva-docker](https://hub.docker.com/r/ninjaben/archiva-docker/)) to manage Maven repositories.

In order to secure the services through HTTPS, and exposing them to the outside world, I installed the excellent nginx-proxy container ([jwilder/nginx-proxy](https://github.com/jwilder/nginx-proxy)) which allows automated nginx proxy configuration using environment variables on containers, and automated HTTP to HTTPS redirection.

DNS are configured to map each public URL of dockerized services to the IP of the host.

Finally, using Docker-Compose, my docker-compose.yml file looks like this :

version: '2'

services:

nginx-proxy:

image: jwilder/nginx-proxy

ports:

- "80:80"

- "443:443"

volumes:

- /var/run/docker.sock:/tmp/docker.sock:ro

- /var/config/nginx-proxy/certs:/etc/nginx/certs:ro

postgresql:

# Configuration of postgresql container ...

gitlab:

image: sameersbn/gitlab

ports:

- "10022:22"

volumes:

- /var/data/gitlab:/home/git/data

environment:

# Bunch of environment variables ...

- VIRTUAL\_HOST=gitlab.my-domain.com

- VIRTUAL\_PORT=80

- CERT\_NAME=star.my-domain.com

archiva:

image: ninjaben/archiva-docker

volumes:

- /var/data/archiva:/var/archiva

environment:

- VIRTUAL\_HOST=archiva.my-domain.com

- VIRTUAL\_PORT=8080

- CERT\_NAME=star.my-domain.com

jenkins:

image: jenkins

volumes:

- /var/data/jenkins:/var/jenkins\_home

environment:

- VIRTUAL\_HOST=jenkins.my-domain.com

- VIRTUAL\_PORT=8080

- CERT\_NAME=star.my-domain.com

For a developer workstation, everything works as expected. One can access the difference services through https://gitlab.my-domain.com, https://repo.my-domain.com and https://jenkins.my-domain.com.

The problem occurs when one of the dockerized service access another dockerized service. For instance, If I try to access https://archiva.my-domain.com from jenkins docker, I will get a timeout error from the proxy.

It seems that even if archiva.my-domain.com is resolved as the public host IP from the docker container, **requests coming from dockerized services are not proxied by nginx-proxy.**

As far as I understood, docker-nginx is handling requests coming from the host network, but does not care about the ones coming from the internal container network (\_dockerconfig\_default\_ for a Docker-Compose stack).

You could say, why would I need to use the proxy from a container ? Of course, I could use URL http://archiva:8080 from Jenkins container, and it would work. But this kind of configuration is not scalable.

For example, using a Gradle build to compile one application, the build.gradle needs to declare my private repository through https://archiva.my-domain.com. It will work if build is launched from a developer workstation, but not through the jenkins container ...

Another example is an authentication in Jenkins by OAuth GitLab service, where the same URL GitLab authentication needs to be **both** available from the outside, and inside the Jenkins container.

My question here is then : **How to configure nginx-proxy to proxy a request from a container to another container ?**

I did not see any topic discussing this problem, and I do not understand enough the problem to build a solution on nginx configuration.

Any help would be really appreciated.

-🡪BMitch, the odds were good, it was indeed a iptables rules problem, and not a misconfiguration of nginx-proxy.

The default policy of chain INPUT for the table filter was DROP, and no rules was made to ACCEPT requests from the container IPs (127.20.X.X).

So for the record, I give some details of the situation if other people face the same problem.

To access containers from the outside world, Docker put rules on PREROUTING and FORWARD rules to allow external IPs to be DNATed from the host IP to the container IPs. Theses default rules allow any external IPs, and that is why limiting access to containers requires some advanced iptables customizations.

See this link for an example : <http://rudijs.github.io/2015-07/docker-restricting-container-access-with-iptables/>

But if your containers need to access host resources (services runing on the host, or in my case, a nginx-proxy container listening to HTTP/HTTPS host port and proxying to containers), you need to take care about the iptables rules of the INPUT chain.

In fact, a request coming from the container and addressed to the host will be routed to the host network stack by the Docker daemon, **but** will then need to pass the INPUT chain (as the request src IP is the host one). So if you want to protect host resources and let containers access them, do not remember to add something like this :

iptables -A INPUT -s 127.20.X.X/24 -j ACCEPT

Where 127.20.X.X/24 is the virtual network on which your containers are running.

Thank you a lot for your help.

# 37.) [How to let different Docker containers talk to each other without exposing the ports to the whole world](https://stackoverflow.com/questions/35194761/how-to-let-different-docker-containers-talk-to-each-other-without-exposing-the-p)

I needed to test my kafka consumer and message triggers in a controlled environment. So I made an ansible project for creating some mock kafka servers: [mokafelk](https://github.com/xialingxiao/mokafelk).

It works fine except the security is shit. The playbook spins up a 3-node dockerized kafkacluster by default but the listening ports on the kafka servers are exposed to all. Here's the [jinja2template](https://github.com/xialingxiao/mokafelk/blob/master/templates/docker-compose.yml) of the Dockerfile used for creating the cluster.

Basically I want the containers to be able to talk to each other. I don't think container linking is an option because it seems to me linking is only one-way. But exposing a port using 127.0.0.1:{{ port }}:{{ port }} only exposes the port to the hosting machine and does not expose the port to the other containers if I am correct. 0.0.0.0:{{ port }}:{{ port }} exposes the port to the whole world. So how could I link two+ containers both ways? This must be a common problem but I don't seem to find a quick solution...

-🡪The docker container networking is explained in detail here: <https://docs.docker.com/engine/userguide/networking/dockernetworks/>

In short:

By default docker daemon adds a network adapter docker0 to the host system (it tries to guess an available IP, often uses 172.17.0.1). You can see this in $ ifconfig.

All containers are by default connected to this network in incremental IPs. You can examine the containers network settings via $ docker inspect <container name>.

So chances are good your docker cluster's IPs are as follows:  
kafka1 172.17.0.2

kafka2 172.17.0.3  
kafka3 172.17.0.4  
elasticsearch 172.17.0.5  
kibana 172.17.0.6

You can then access your kafkas at 172.17.0.2:9092, 172.17.0.3:9092, 172.17.0.4:9092 from the host system and from the containers alike.

----thank you mate for the answer i hv never tried this but making the ip address hard coded inside the scripts is bad practice as well no? it also assumes that no other containers r running at the start of play – Lingxiao Xia Feb 5 '16 at

Just read the link you posted. I wasn't aware of user defined networks of docker. it seems like a new feature. docker network create --driver bridge isolated\_nw creates a new network and creating all containers inside this user-defined network would allow them to communicate with each other by name automatically, which is not supported in the default bridge network. – Lingxiao Xia Feb 5 '16 at 2:56

# 38.) [Docker: where is docker volume located for this compose file](https://stackoverflow.com/questions/45271420/docker-where-is-docker-volume-located-for-this-compose-file)

 was setting up some materials for a trainning, when I came around this sample compose file:

<https://github.com/dockersamples/example-voting-app/blob/master/docker-compose.yml>

and I couldn't find out how this volume is mounted, on lines 48 and 49 of the file:

volumes:

db-data:

Can someone explain me where is this volume on the host? Couldn't find it and I wouldn't like to keep any postgresql data dangling around after the containers are gone. Similar thing happens to the networks:

networks:

front-tier:

back-tier:

Why docker compose accepts empty network definitions like this?

## --🡪**Finding the volumes**

Volumes like this are internal to Docker and stored in the Docker store (which is usually all under /var/lib/docker). You can get a list of volumes:

$ docker volume ls

DRIVER VOLUME NAME

local 1c59d5b7e90e9173ca30a7fcb6b9183c3f5a37bd2505ca78ad77cf4062bd0465

local 2f13b0cec834a0250845b9dcb2bce548f7c7f35ed9cdaa7d5990bf896e952d02

local a3d54ec4582c3c7ad5a6172e1d4eed38cfb3e7d97df6d524a3edd544dc455917

local e6c389d80768356cdefd6c04f6b384057e9fe2835d6e1d3792691b887d767724

You can find out exactly where the volume is stored on your system if you want to:

$ docker inspect 1c59d5b7e90e9173ca30a7fcb6b9183c3f5a37bd2505ca78ad77cf4062bd0465

[

{

"Driver": "local",

"Labels": null,

"Mountpoint": "/var/lib/docker/volumes/1c59d5b7e90e9173ca30a7fcb6b9183c3f5a37bd2505ca78ad77cf4062bd0465/\_data",

"Name": "1c59d5b7e90e9173ca30a7fcb6b9183c3f5a37bd2505ca78ad77cf4062bd0465",

"Options": {},

"Scope": "local"

}

]

## **Cleaning up unused volumes**

As far as just ensuring that things are not left dangling, you can use the prune commands, in this case docker volume prune. That will give you this output, and you choose whether to continue pruning or not.

$ docker volume prune

WARNING! This will remove all volumes not used by at least one container.

Are you sure you want to continue? [y/N]

## **"Empty" definitions in docker-compose.yml**

There is a tendency to accept these "empty" definitions for things like volumes and networks when you don't need to do anything other than define that a volume or network should exist. That is, if you want to create it, but are okay with the default settings, then there is no particular reason to specify the parameters.

---- Awesome answer! Thanks man – Vini.g.fer Jul 24 at 3:09

Thanx for this "prune" command! It is worth noting, that (as in my case) inspect actually won't show a correct mount location. and simple "volume rm" won't work either, saying, that the "volume is in use" Sometimes you just have to prune...

# 39.) [Docker-Compose persistent data MySQL](https://stackoverflow.com/questions/39175194/docker-compose-persistent-data-mysql)

I can't seem to get MySQL data to persist if I run $ docker-compose down with the following .yml

version: '2'

services:

# other services

data:

container\_name: flask\_data

image: mysql:latest

volumes:

- /var/lib/mysql

command: "true"

mysql:

container\_name: flask\_mysql

restart: always

image: mysql:latest

environment:

MYSQL\_ROOT\_PASSWORD: 'test\_pass' # TODO: Change this

MYSQL\_USER: 'test'

MYSQL\_PASS: 'pass'

volumes\_from:

- data

ports:

- "3306:3306"

My understanding is that in my data container using volumes: - /var/lib/mysql maps it to my local machines directory where mysql stores data to the container and because of this mapping the data should persist even if the containers are destroyed. And the mysql container is just a client interface into the db and can see the local directory because of volumes\_from: - data

Attempted this answer and it did not work. [Docker-Compose Persistent Data Trouble](https://stackoverflow.com/questions/34511336/docker-compose-persistent-data-trouble)

**EDIT**

Changed my .yml as shown below and created a the dir ./data but now when I run docker-compose up --build the mysql container wont start throws error saying

data:

container\_name: flask\_data

image: mysql:latest

volumes:

- ./data:/var/lib/mysql

command: "true"

mysql:

container\_name: flask\_mysql

restart: always

image: mysql:latest

environment:

MYSQL\_ROOT\_PASSWORD: 'test\_pass' # TODO: Change this

MYSQL\_USER: 'test'

MYSQL\_PASS: 'pass'

volumes\_from:

- data

ports:

- "3306:3306"

flask\_mysql | mysqld: Can't create/write to file '/var/lib/mysql/is\_writable' (Errcode: 13 - Permission denied)

flask\_mysql | 2016-08-26T22:29:21.182144Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit\_defaults\_for\_timestamp server option (see documentation for more details).

flask\_mysql | 2016-08-26T22:29:21.185392Z 0 [ERROR] --initialize specified but the data dir

-🡪The data container is a superfluous workaround. [Data-volumes](https://docs.docker.com/engine/tutorials/dockervolumes/) would do the trick for you. Alter your docker-compose.yml to:

version: '2'

services:

mysql:

container\_name: flask\_mysql

restart: always

image: mysql:latest

environment:

MYSQL\_ROOT\_PASSWORD: 'test\_pass' # TODO: Change this

MYSQL\_USER: 'test'

MYSQL\_PASS: 'pass'

volumes:

- my-datavolume:/var/lib/mysql

volumes:

my-datavolume:

Docker will create the volume for you in the /var/lib/docker/volumes folder. This volume persist as long as you are not typing docker-compose down -v

------ This appears to have fixed it. Thanks. – Adam Aug 29 '16 at 18:58

Starting with MySQL 5.7.6 there will (again) be permission problems with the mysql Docker image. You can instead use the mariadb Docker image, which works flawlessly with Docker volumes.

-🡪You have to create a separate volume for mysql data.

So it will look like this:

volumes\_from:

- data

volumes:

- ./mysql-data:/var/lib/mysql

And no, /var/lib/mysql is a path inside your mysql container and has nothing to do with a path on your host machine. Your host machine may even have no mysql at all. So the goal is to persist an internal folder from a mysql container.

-🡪There are 3 ways:

**First.** You need specify directory to store mysql data on your **host machine**. You can remove data container. Your mysql data will be saved on you local filesystem.

Mysql container definition must looks like this

mysql:

container\_name: flask\_mysql

restart: always

image: mysql:latest

environment:

MYSQL\_ROOT\_PASSWORD: 'test\_pass' # TODO: Change this

MYSQL\_USER: 'test'

MYSQL\_PASS: 'pass'

volumes:

- /opt/mysql\_data:/var/lib/mysql

ports:

- "3306:3306"

**Second way** is commit data container before typing docker-compose down:

docker commit my\_data\_container

docker-compose down

**Third way.** Also you can use docker-compose stop instead of docker-compose down (then you don't need commit container)

----Can't I just have volumes: - /var/lib/mysql because it maps HOST:CONTAINER and if you don't specify w/ a colon it maps the same dir? – Adam

Not. unfortunately in this case docker map this container directory to random host folder like /var/lib/docker/volumes/ec3c543bc92f114c2c568733541e89381881‌​e5a62996d7084e07793f‌​86280535 – Bukharov Sergey Aug 26 '16 at 22:27

Okay I thought volumes: - /var/lib/mysql was equivalent to volumes: - /var/lib/mysql:/var/lib/mysql – Adam Aug 26 '16 at 22:33

your thrid way will not work because docker does not commit data from volumes to the image. see github.com/moby/moby/issues/6999 – Ohmen May

# 40.) [Pull private docker images from Google Container Registry w/o gcloud](https://stackoverflow.com/questions/30926813/pull-private-docker-images-from-google-container-registry-w-o-gcloud)

I'm using shippable to push private docker images to the Google Container Registry that I then want to pull from either locally on a laptop, or inside an instance on the Google Compute Engine.

I know that the command gcloud preview docker pull gcr.io/projectID/image-name works, but I can't rely on gcloud being installed on every machine that someone may need to pull the image from.

If I run docker-compose up -d on my machine then I get the following error:

Pulling image gcr.io/projectID/image-name...

Pulling repository gcr.io/projectID/image-name

Traceback (most recent call last):

File "<string>", line 3, in <module>

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.cli.main", line 31, in main

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.cli.docopt\_command", line 21, in sys\_dispatch

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.cli.command", line 27, in dispatch

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.cli.docopt\_command", line 24, in dispatch

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.cli.command", line 59, in perform\_command

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.cli.main", line 464, in up

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.project", line 208, in up

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.service", line 214, in recreate\_containers

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.service", line 199, in create\_container

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.progress\_stream", line 37, in stream\_output

File "/compose/build/docker-compose/out00-PYZ.pyz/compose.progress\_stream", line 50, in print\_output\_event

compose.progress\_stream.StreamOutputError: Error: Status 403 trying to pull repository projectID/image-name: "Access denied."

Is there any way to authenticate or access the image with some form of OAuth or keys? I want to avoid having to install gcloud on every machine that will ever need to pull the image, and the images have to remain private.

I have tried gcloud preview docker -a but that is not the solution I'm looking for.

Thank you in advance for any help

-🡪If you want to work with the Google Container Registry on a machine not in the Google Compute Engine (i.e. local) using vanilla docker you can [follow Google's instructions](https://cloud.google.com/container-registry/docs/auth).

The two main methods are using an access token or a JSON key file.

*Note that \_token and \_json\_key are the actual values you provide for the username (-u)*

**Access Token**

$ docker login -e 1234@5678.com -u \_token -p "$(gcloud auth print-access-token)" https://gcr.io

**JSON Key File**

$ docker login -e 1234@5678.com -u \_json\_key -p "$(cat keyfile.json)" https://gcr.io

To create a key file you can follow these instructions:

1. Open the Credentials page.
2. To set up a new service account, do the following:
   * Click Add credentials > Service account.
   * Choose whether to download the service account's public/private key as a standard P12 file, or as a JSON file that can be loaded by a Google API client library.
   * Your new public/private key pair is generated and downloaded to your machine; it serves as the only copy of this key. You are responsible for storing it securely.

You can [view Google's documentation on generating a key file here](https://developers.google.com/console/help/new/#serviceaccounts).

# 41.) [How to pass arguments within docker-compose?](https://stackoverflow.com/questions/34322631/how-to-pass-arguments-within-docker-compose)

Docker 1.9 allows to pass arguments to a dockerfile. See link: <https://docs.docker.com/engine/reference/builder/#arg>

How can i pass the same arugments within docker-compose.yml??  
Please provide an example too, if possible.

-🡪This feature was added in Compose 1.6.

Reference: <https://docs.docker.com/compose/compose-file/#args>

services:

web:

build:

context: .

args:

FOO: foo

-🡪This can now be done as of docker-compose v2+ as part of the build object;

docker-compose.yml

version: '2'

services:

my\_image\_name:

build:

context: . #current dir as build context

args:

var1: 1

var2: c

See [the docker compose docs](https://docs.docker.com/compose/compose-file/#build).

In the above example "var1" and "var2" will be sent to the build environment.

Note: any env variables (specified by using the environment block) which have the same name as args variable(s) will override that variable.

-🡪Now docker-compose supports variable substitution.

Compose uses the variable values from the shell environment in which docker-compose is run. For example, suppose the shell contains POSTGRES\_VERSION=9.3 and you supply this configuration in your docker-compose.yml file:

db:

image: "postgres:${POSTGRES\_VERSION}"

When you run docker-compose up with this configuration, Compose looks for the POSTGRES\_VERSION environment variable in the shell and substitutes its value in. For this example, Compose resolves the image to postgres:9.3 before running the configuration.

# 42.) [Docker-compose and pdb](https://stackoverflow.com/questions/30854967/docker-compose-and-pdb)

I see that I'm not the first one to ask the question but there was no clear answer to this:

How to use pdb with docker-composer in Python development?

When you ask uncle Google about django docker you get awesome docker-composer examples and tutorials and I have an environment working - I can run docker-compose up and I have a neat developer environment **but the PDB is not working** (which is very sad).

I can settle with running docker-compose run my-awesome-app python app.py 0.0.0.0:8000 but then I can access my application over [http://127.0.0.1:8000](http://127.0.0.1:8000/) from the host (I can with docker-compose up) and it seems that each time I use run new containers are made like: dir\_app\_13 and dir\_db\_4 which I don't desire at all.

People of good will please aid me.

PS  
I'm using pdb++ for that example and a basic docker-compose.yml from [this django example](https://docs.docker.com/compose/django/). Also I experimented but nothing seems to help me. And I'm using docker-composer [1.3.0rc3](https://github.com/docker/compose/tree/1.3.0rc3) as it has Dockerfile pointing support.

-🡪Try running your web container with the --service-ports option: docker-compose run --service-ports web

---That did it! Thank you kind sir. I'm now using docker-compose to create my environment and running my service or it's tests with docker-compose run --rm --service-ports my-awesome-app python app.py 0.0.0.0:8000. Goodbye Vagrant! – McAbra Jun 17 '15 at 19:51

I am having some issue with this method. My terminal is blank when I am typing something in pdb – Jitu Jan 3 '16 at 12:16

You get to the pdb prompt, though? I don't know how to help without more information, like what framework you're using, what services, etc. – Jamey Jan 4 '16 at 19:30

I think you also need something like -it: check the answer below – jpic

# -🡪

|  |  |  |  |
| --- | --- | --- | --- |
|  | Use the following steps to attach pdb on any python script.  Step 1. Add the following in your yml file  stdin\_open: true  tty: true  This will enable interactive mode and will attach stdin. This is equivalent for -it mode.  Step 2.  docker attach <generated\_instanceid>  You'll now get the pdb shell 43.) [Docker Machine on Mac: Cannot see mounted Volumes on docker host/docker-machine? Where are volumes physically stored?](https://stackoverflow.com/questions/33575351/docker-machine-on-mac-cannot-see-mounted-volumes-on-docker-host-docker-machine)  |  |  | | --- | --- | |  | Am on a Macbook Pro laptop and running docker-machine (0.5.0) and docker-compose (1.5.0) to get my containers going.  This means I'm using docker-machine to create my virtualbox boot2docker driven HOST machines, which will run my docker daemon and host all my containers.  I think I'm missing something critical with the concept of HOSTS and VOLUME, as they refer to Docker and the documentation.  This is my docker-compose.yml file (web simply builds the php:5.6-apache image):  web:  restart: "always"  build: ./docker-containers/web  ports:  - "8080:80"  volumes:  - ./src:/var/www/html  links:  - mysql:mysql  mysql:  restart: "always"  image: mysql:5.7  volumes\_from:  - data  ports:  - "3306:3306"  environment:  - MYSQL\_ROOT\_PASSWORD=XXX  data:  restart: "no"  image: mysql:5.7  volumes:  - /var/lib/mysql  command: "true"  Docker Compose file documention for volumes is here: <http://docs.docker.com/compose/compose-file/>  It states for volumes - Mount paths as volumes, optionally specifying a path on the host machine (HOST:CONTAINER), or an access mode (HOST:CONTAINER:ro).  HOST in this case refers to my VM created by docker-machine, correct? Or my local macbook file system? Mounting a path on my VM to a container?  Under web I declare:  volumes:  - ./src:/var/www/html  and this is mapping my local macbook file system ./src folder on my macbook pro to my web container. If my understanding is correct though, shouldn't it be mapping the ./src folder on my **VM**to /var/www/html within the web container?! In theory I think I should be required to COPY my local mac file system folder ./src to my VM first, and then I do this volume declaration. It seems docker-compose is magically doing it all at once though? confused  Lastly, we can see that I'm creating a data-only container to persist my mysql data. I've declared:  volumes:  - /var/lib/mysql  Shouldn't this create a /var/lib/mysql folder on my HOST boot2docker VM and I could then navigate to this folder on the VM, yes/no? When I use docker-machine to ssh into my machine, and then navigate to /var/lib, there is **NO** mysql folder at all?! Why is it not being created? Is there something wrong with my configuration? :/  Thanks in advance! Any explanations as to what I'm doing wrong here would be greatly appreciated! | |

---🡪Still some apps ( Apache for example ) will still give you a hard time. The fact that the user id of whatever runs in the container differs from your Mac user id will make your life hell. In order to get around this, you need to adjust the user id as well as the user group in a way that doesn't conflict with your Mac's permissions. The group you want on a Mac is staff, a UID that works would be for example 1000. Hence you could put this at the end of your Dockerfile:

RUN usermod -u 1000 www-data

RUN usermod -G staff www-data

or

RUN usermod -u 1000 mysql

RUN usermod -G staff mysql

So as you have now learnt:

In theory I think I should be required to COPY my local mac file system folder ./src to my VM first, and then I do this volume declaration. It seems docker-compose is magically doing it all at once though?

Right on, it does that :)

# 43.) [NGINX Reverse Proxy failing with Linked Docker Containers](https://stackoverflow.com/questions/31331760/nginx-reverse-proxy-failing-with-linked-docker-containers)

I have the following docker-compose.yml:

node1:

build: ./node

links:

- redis

ports:

- "8080"

node2:

build: ./node

links:

- redis

ports:

- "8080"

service1:

build: ./service

links:

- redis

ports:

- "8383"

redis:

image: redis

ports:

- "6379"

nginx:

build: ./nginx

links:

- node1:node1

- node2:node2

- service1:service1

ports:

- "80:80"

After executing this and running docker ps I get the following:

080d9d7dc2e0 dockerworkflow\_nginx:latest "nginx -g 'daemon of 5 minutes ago Up 5 minutes 0.0.0.0:80->80/tcp, 443/tcp dockerworkflow\_nginx\_1

8c25bfdb9d00 dockerworkflow\_node1:latest "nodemon /src/index. 6 minutes ago Up 6 minutes 0.0.0.0:33023->8080/tcp dockerworkflow\_node1\_1

4ae817be2a63 dockerworkflow\_service1:latest "nodemon /src/index. 6 minutes ago Up 6 minutes 0.0.0.0:33022->8383/tcp dockerworkflow\_service1\_1

91ff238fe3f6 dockerworkflow\_node2:latest "nodemon /src/index. 6 minutes ago Up 6 minutes 0.0.0.0:33021->8080/tcp dockerworkflow\_node2\_1

fe0c7e02c860 redis:latest "/entrypoint.sh redi 6 minutes ago Up 6 minutes 0.0.0.0:33020->6379/tcp dockerworkflow\_redis\_1

Everything seems to be good so far.

The nginx.conf I am using looks like the following:

worker\_processes 4;

events { worker\_connections 1024; }

http {

server {

listen 80;

location / {

proxy\_pass http://node1;

}

location /a/ {

proxy\_pass http://node2;

}

location /b/ {

proxy\_pass http://service1;

}

}

}

All this should really be doing is the following:

If I enter http://{host-ip}/ then the node1 container is forwarded the request.

If I enter http://{host-ip}/a/ then the node2 container is forwarded the request.

If I enter http://{host-ip}/b/ then the service1 container is forwarded the request.

Right now, I am getting 502 Bad Gateway if I try anything.

--🡪I was able to figure out the solution and it turned out to be something stupid that didn't show up in any logs and was difficult for me to come across.

Below is the updated nginx.conf file.

worker\_processes 4;

events { worker\_connections 1024; }

http {

upstream node\_app {

server node1:8080;

}

upstream service\_app {

server service1:8383;

}

server {

listen 80;

location / {

proxy\_pass http://node\_app/;

include /etc/nginx/proxy\_params;

}

location /a/ {

proxy\_pass http://node\_app/;

include /etc/nginx/proxy\_params;

}

location /b/ {

proxy\_pass http://service\_app/;

include /etc/nginx/proxy\_params;

}

}

}

Not sure if the include is necessary at this point, but the trailing / at the end of the proxy\_passdirective seem to do the trick at the end of the day.

----  
Out of curiosity, did the upstream {} block do anything to resolve your issue? Or was it just having a trailing / that fixed it .

# 44.) [Running docker on Ubuntu: mounted host volume is not writable from container](https://stackoverflow.com/questions/34031397/running-docker-on-ubuntu-mounted-host-volume-is-not-writable-from-container)

Docker works great on a Mac for me, but I have to run docker host inside of a VirtualBox (or Parallels, or VMWare Fusion), since Mac's kernel doesn't support docker.

So I tried to setup my application and a docker-compose on an Ubuntu Desktop - natively, where both docker client and docker host run physically on the same system. This worked, but my running docker containers can't write into a mounted host volume.

I use docker-compose with the following settings:

volumes:

- ./api:/usr/src/app

So I'm mounting the "api" directory of the host Ubuntu OS into docker container under /usr/src/app.

docker inspect <container ID> shows that the volume is writable

"Destination": "/usr/src/app",

"Mode": "rw",

"RW": true

However it is not: I get permission denied when I try to create a directory or edit a file from within the docker container.

I googled for this issue, of course, and I came across a few SELinux issues of CentOS/RHEL, but **I'm running Ubuntu 15.10, 64 bit edition**, not CentOS.

-🡪If your uid on the host (id -u) isn't the same as the uid of the user in the docker container (often "docker") then you can have this problem. You can try:

1. Making the UIDs the same between your user and the user in the docker container.
2. Setting the group permissions on the directory to be writable for a group that both you and docker belong to.
3. You could also use the nuclear option:

chmod a+rwx -R project-dir/

The nuclear option will make your git workspace filthy, which will annoy you greatly, so isn't the best long-term solution. It stops the bleeding tho.

For further understanding the problem, you might find these useful:

1. <https://github.com/docker/docker/issues/7906>
2. <https://github.com/docker/docker/issues/7198>

---Both "nuclear option" and uid fix worked. For uid fix I just added "usermod -u 1000 docker" to the Dockerfile

# 45.) [Docker, Django and Selenium - Selenium unable to connect](https://stackoverflow.com/questions/35143927/docker-django-and-selenium-selenium-unable-to-connect)

I have Docker configured to run Postgres and Django using docker-compose.yml and it is working fine.

The trouble I am having is with Selenium not being able to connect to the Django liveserver.

Now it makes sense (to me at least) that django has to access selenium to control the browser and selenium has to access django to access the server.

I have tried using the docker 'ambassador' pattern using the following configuration for docker-compose.yml from here: <https://github.com/docker/compose/issues/666>

postgis:

dockerfile: ./docker/postgis/Dockerfile

build: .

container\_name: postgis

django-ambassador:

container\_name: django-ambassador

image: cpuguy83/docker-grand-ambassador

volumes:

- "/var/run/docker.sock:/var/run/docker.sock"

command: "-name django -name selenium"

django:

dockerfile: ./docker/Dockerfile-dev

build: .

command: python /app/project/manage.py test my-app

container\_name: django

volumes:

- .:/app

ports:

- "8000:8000"

- "8081:8081"

links:

- postgis

- "django-ambassador:selenium"

environment:

- SELENIUM\_HOST=http://selenium:4444/wd/hub

selenium:

container\_name: selenium

image: selenium/standalone-firefox-debug

ports:

- "4444:4444"

- "5900:5900"

links:

- "django-ambassador:django"

When I check [http://DOCKER-MACHINE-IP:4444/wd/hub/static/resource/hub.html](http://docker-machine-ip:4444/wd/hub/static/resource/hub.html) I can see that firefox starts, but all the tests fail as firefox is unable to connect to django

'Firefox can't establish a connection to the server at localhost:8081'

I also tried this solution here <https://github.com/docker/compose/issues/1991> however this is not working cause I can't get django to connect to postgis and selenium at the same time

'django.db.utils.OperationalError: could not translate host name "postgis" to address: Name or service not known'

I tried using the networking feature as listed below

postgis:

dockerfile: ./docker/postgis/Dockerfile

build: .

container\_name: postgis

net: appnet

django:

dockerfile: ./docker/Dockerfile-dev

build: .

command: python /app/project/manage.py test foo

container\_name: django

volumes:

- .:/app

ports:

- "8000:8000"

- "8081:8081"

net: appnet

environment:

- SELENIUM\_HOST=http://selenium:4444/wd/hub

selenium:

container\_name: selenium

image: selenium/standalone-firefox-debug

ports:

- "4444:4444"

- "5900:5900"

net: appnet

but the result is the same

'Firefox can't establish a connection to the server at localhost:8081'

So how can I get selenium to connect to django?

I have been playing around with this for days - would really appreciate any help.

**More Info**

Another weird thing is that when the testserver is running **not** using docker (using my old config of virtualenv etc.) if I run ./manage.py test foo I can access the server through any browser at [http://localhost:8081](http://localhost:8081/) and get served up webpages, but I can't access the test server when I run the equivalent command if I run it under docker. This is weird cause I am mapping port 8081:8081 - is this related?

Note: I am using OSX and Docker v1.9.1

-🡪I've been struggling with this as well, and I finally found a solution that worked for me. You can try something like this:

postgis:

dockerfile: ./docker/postgis/Dockerfile

build: .

django:

dockerfile: ./docker/Dockerfile-dev

build: .

command: python /app/project/manage.py test my-app

volumes:

- .:/app

ports:

- "8000:8000"

links:

- postgis

- selenium # django can access selenium:4444, selenium can access django:8081-8100

environment:

- SELENIUM\_HOST=http://selenium:4444/wd/hub

- DJANGO\_LIVE\_TEST\_SERVER\_ADDRESS=django:8081-8100 # this gives selenium the correct address

selenium:

image: selenium/standalone-firefox-debug

ports:

- "5900:5900"

I don't think you need to include port 4444 in the selenium config. That port is exposed by default, and there's no need to map it to the host machine, since the django container can access it directly via its link to the selenium container.

[Edit] I've found you don't need to explicitly expose the 8081 port of the django container either. Also, I used a range of ports for the test server, because if tests are run in parallel, you can get an "Address already in use" error, as discussed [here](https://docs.djangoproject.com/en/1.10/topics/testing/tools/#liveservertestcase).

--🡪I ended up coming up with a better solution that didn't require me to hardcode the IP Address. Below is the configuration I used to run tests in django with docker.

### **Docker-compose file**

# docker-compose base file for everything

version: '2'

services:

postgis:

build:

context: .

dockerfile: ./docker/postgis/Dockerfile

container\_name: postgis

volumes:

# If you are using boot2docker, postgres data has to live in the VM for now until #581 fixed

# for more info see here: https://github.com/boot2docker/boot2docker/issues/581

- /data/dev/docker\_cookiecutter/postgres:/var/lib/postgresql/data

django:

build:

context: .

dockerfile: ./docker/django/Dockerfile

container\_name: django

volumes:

- .:/app

depends\_on:

- selenium

- postgis

environment:

- SITE\_DOMAIN=django

- DJANGO\_SETTINGS\_MODULE=settings.my\_dev\_settings

links:

- postgis

- mailcatcher

selenium:

container\_name: selenium

image: selenium/standalone-firefox-debug:2.52.0

ports:

- "4444:4444"

- "5900:5900"

### **Dockerfile (for Django)**

ENTRYPOINT ["/docker/django/entrypoint.sh"]

### **In Entrypoint file**

#!/bin/bash

set -e

# Now we need to get the ip address of this container so we can supply it as an environmental

# variable for django so that selenium knows what url the test server is on

# Use below or alternatively you could have used

# something like "$@ --liveserver=$THIS\_DOCKER\_CONTAINER\_TEST\_SERVER"

if [[ "'$\*'" == \*"manage.py test"\* ]] # only add if 'manage.py test' in the args

then

# get the container id

THIS\_CONTAINER\_ID\_LONG=`cat /proc/self/cgroup | grep 'docker' | sed 's/^.\*\///' | tail -n1`

# take the first 12 characters - that is the format used in /etc/hosts

THIS\_CONTAINER\_ID\_SHORT=${THIS\_CONTAINER\_ID\_LONG:0:12}

# search /etc/hosts for the line with the ip address which will look like this:

# 172.18.0.4 8886629d38e6

THIS\_DOCKER\_CONTAINER\_IP\_LINE=`cat /etc/hosts | grep $THIS\_CONTAINER\_ID\_SHORT`

# take the ip address from this

THIS\_DOCKER\_CONTAINER\_IP=`(echo $THIS\_DOCKER\_CONTAINER\_IP\_LINE | grep -o '[0-9]\+[.][0-9]\+[.][0-9]\+[.][0-9]\+')`

# add the port you want on the end

# Issues here include: django changing port if in use (I think)

# and parallel tests needing multiple ports etc.

THIS\_DOCKER\_CONTAINER\_TEST\_SERVER="$THIS\_DOCKER\_CONTAINER\_IP:8081"

echo "this docker container test server = $THIS\_DOCKER\_CONTAINER\_TEST\_SERVER"

export DJANGO\_LIVE\_TEST\_SERVER\_ADDRESS=$THIS\_DOCKER\_CONTAINER\_TEST\_SERVER

fi

eval "$@"

### **In your django settings file**

SITE\_DOMAIN = 'django'

### **Then to run your tests**

docker-compose run django ./manage.py test

# 44.) [Docker named volumes vs DOC (data-only-containers)](https://stackoverflow.com/questions/36011595/docker-named-volumes-vs-doc-data-only-containers)

Up to recent version of Docker (v1.10), we were thought that we can use DOC: *data-only containers*. So I would create such DOC (based on e.g. busybox) and use --volumes-from to link it to my container. You can still read about this in [Docker documentation](https://docs.docker.com/engine/userguide/containers/dockervolumes/#creating-and-mounting-a-data-volume-container).

With new version of docker, it is said that instead of DOC we should use named volumes. Here is an example of docker-compose.yml:

version: '2'

services:

elasticsearch:

image: elasticsearch:2.2.0

command: elasticsearch -Des.network.host=0.0.0.0

ports:

- "9201:9200"

volumes:

- "es-data:/usr/share/elasticsearch/data"

volumes:

es-data:

Here we created and use named volume es-data.

There is still not much documentation on this new feature. I am asking:

* Can we replace DOC with named containers? How long volume is persisted? What if I remove the container that is using it?
* How can we e.g. backup now? Previously, I could docker run --rm --volumes-from es-data ... and then tar it.

-🡪Can we replace DOC with named containers?

In many cases, yes, named containers will be a better option.

How long volume is persisted? What if I remove the container that is using it?

If you remove the container, the volume will still be there. The only way to remove the volume is to use docker-compose down -v or docker volume rm <volume name>.

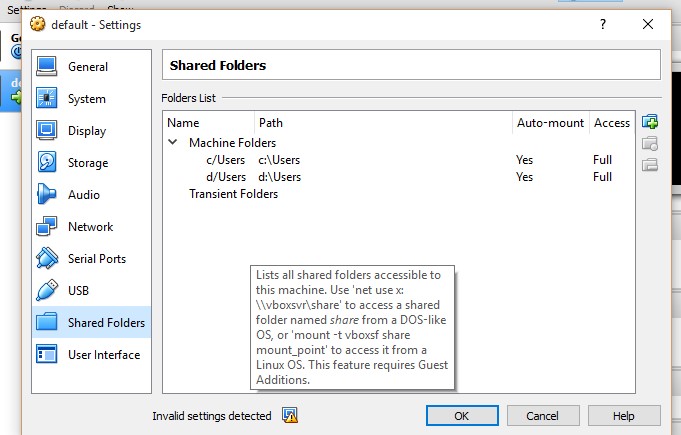
How can we e.g. backup now? Previously, I could docker run --rm --volumes-from es-data ... and then tar it.

Instead of --volumes-from, you can use --volume=<volume name>.

Note that volumes created by docker-compose are always prefixed with the project name, so if you use it with a docker command the full name is actually <project\_name>\_es-data.

# 45.) [docker with shared folder d drive](https://stackoverflow.com/questions/38046051/docker-with-shared-folder-d-drive)

I'm trying to get docker to mount my d drive. Have been struggling with a lot of stack and git, and none of them seem to work for me.

I already added my d drive to the shared folder.

then I mount in my docker-compose.yml



running docker-compose up works when the folder is in C drive, but not when in D drive.

I will get the error package.json not found, meaning that the D drive is not mounted.

I tried to inspect my container, and this is what I get

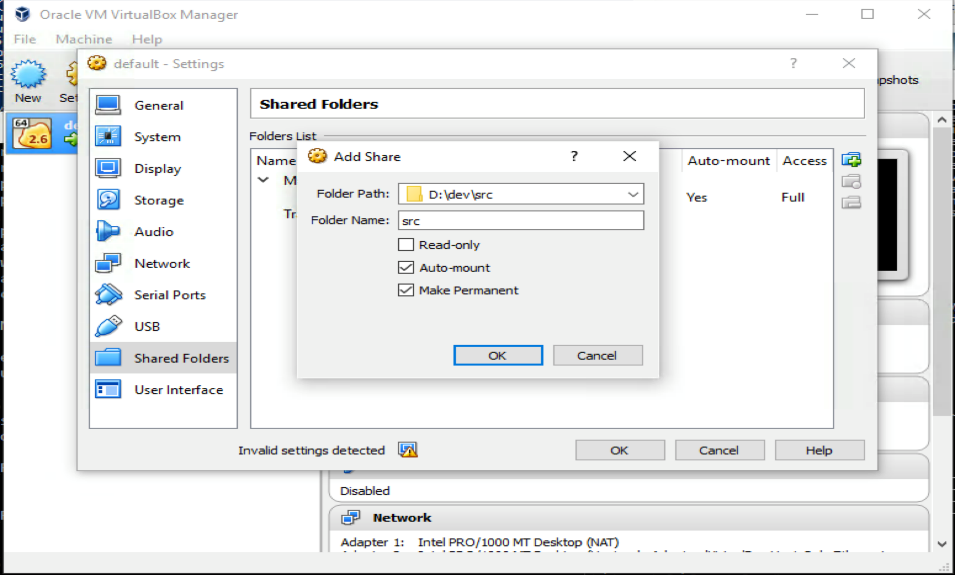
can see that source is the right path, but it doesn't work.

Is it something wrong with my setting, or that docker doesn't support this yet? Any workaround? like move everything to D drive?

I did tried with setting the MACHINE\_STORAGE\_PATH in environment, turn out didn't work and screw up everything. Must be I did something wrong.

Please help. My new laptop only have 128GB in C drive, is not possible have everything in the C drive.

-🡪You could check "[Docker: Permanently Mount a VirtualBox Shared Folder](http://www.developmentalmadness.com/2016/03/05/docker-permanently-mount-a-virtualbox-shared-folder/)"

Creating a shared folder in VirtualBox (which you did) is only the first step

You still need to mount it permanently in your boot2docker ssh session:

sudo touch /mnt/sda1/var/lib/boot2docker/bootlocal.sh

Add to that file:

mkdir -p /mnt/src

mount -t vboxsf -o defaults,uid=`id -u docker`,gid=`id -g docker` src /mnt/src

# 46.) [Difference between links and depends\_on in docker\_compose.yml](https://stackoverflow.com/questions/35832095/difference-between-links-and-depends-on-in-docker-compose-yml)

This answer is for docker-compose **version 2** and it also works on **version 3**

**You can still access the data when you use depends\_on.**

If you look at docker docs [Docker Compose and Django](https://docs.docker.com/compose/django/#/define-the-project-components), you still can access the database like this:

version: '2'

services:

db:

image: postgres

web:

build: .

command: python manage.py runserver 0.0.0.0:8000

volumes:

- .:/code

ports:

- "8000:8000"

depends\_on:

- db

What is the difference between links and depends\_on?

**links:**

When you create a container for a database, for example:

docker run -d --name=test-mysql --env="MYSQL\_ROOT\_PASSWORD=mypassword" -P mysql

docker inspect d54cf8a0fb98 |grep HostPort

And you may find

"HostPort": "32777"

This means you can connect the database from your localhost port 32777 (3306 in container) but this port will change every time you restart or remove the container. So you can use links to make sure you will always connect to the database and don't have to know which port it is.

web:

links:

- db

**depends\_on:**

I found a nice blog from Giorgio Ferraris [Docker-compose.yml: from V1 to V2](https://medium.com/@giorgioto/docker-compose-yml-from-v1-to-v2-3c0f8bb7a48e#.ukh8ajps0)

When docker-compose executes V2 files, it will automatically build a network between all of the containers defined in the file, and every container will be immediately able to refer to the others just using the names defined in the docker-compose.yml file.

And

So we don’t need links anymore; links were used to start a network communication between our db container and our web-server container, but this is already done by docker-compose

Update

[depends\_on](https://docs.docker.com/compose/compose-file/#/dependson)

Express dependency between services, which has two effects:

* docker-compose up will start services in dependency order. In the following example, db and redis will be started before web.
* docker-compose up SERVICE will automatically include SERVICE’s dependencies. In the following example, docker-compose up web will also create and start db and redis.

Simple example:

version: '2'

services:

web:

build: .

depends\_on:

- db

- redis

redis:

image: redis

db:

image: postgres

Note: depends\_on will not wait for db and redis to be “ready” before starting web - only until they have been started. If you need to wait for a service to be ready, see Controlling startup order for more on this problem and strategies for solving it.

# 47.) [How to restart a single container with docker-compose](https://stackoverflow.com/questions/31466428/how-to-restart-a-single-container-with-docker-compose)

I have a docker-compose.yml file that contains 4 containers: redis, postgres, api, worker

During the development of worker, I often need to restart it in order to apply changes. Is there any good way to restart a container (e.g. worker) without restarting the other containers?

-🡪docker-compose restart worker

You can set the time to wait for stop before killing the container (in seconds)

docker-compose restart -t 30 worker

---for me it worked, but a general question if allowed here: does 'restart' take care of linked containers and update the /etc/hosts or doesn't a 'restart' change any IPs at all? – michabbb Jan 11 '16 at 17:04

The containers are linked by name and typically the only IP you need to worry about is the external docker host IP (typically 192.168.99.100). Where there can be some trouble is if you, say, restart a database container that other containers are connected to. The dependent containers will have to be resilient enough to reconnect. – Ryan Kimber Apr 2 '16 at 16:34

-🡪To restart a service with changes here are the steps that I performed:

docker-compose stop -t 1 worker

docker-compose build worker

docker-compose create worker

docker-compose start worker

---If you need changes to apply with a build, you can easily do a docker-compose up -d --build and it will rebuild everything and restart any changed containers. No need for the stop first, with downtime, and separate create and start commands. – BMitch Sep 15 '16 at 0:05

Yes, if you want to restart all services, but the OP only wants to restart a single service and not restart the others – Jeff Sep 16 '16 at 16:49

See the answer I posted, in the example, the up will only recreated the container that had been changed and therefore needed a restart. –

-🡪The other answers to restarting a single node are on target, docker-compose restart worker. That will bounce that container, but not include any changes, even if you rebuilt it separately. You can manually stop, rm, create, and start, but there are much easier methods.

If you've updated your code, you can do the build and reload in a single step with:

docker-compose up -d --build

That will first rebuild your images from any changed code, which is fast if there are no changes since the cache is reused. And then it only replaces the changed containers. If your downloaded images are stale, you can precede the above command with:

docker-compose pull

To download any changed images first (the containers won't be restarted until you run a command like the up above). Doing an initial stop is unnecessary.

And to only do this for a single service, follow the up or pull command with the services you want to specify, e.g.:

docker-compose up -d --build worker

Here's a quick example of the first option, the Dockerfile is structured to keep the frequently changing parts of the code near the end. In fact the requirements are pulled in separately for the pip install since that file rarely changes. And since the nginx and redis containers were up-to-date, they weren't restarted. Total time for the entire process was under 6 seconds:

$ time docker-compose -f docker-compose.nginx-proxy.yml up -d --build

Building counter

Step 1 : FROM python:2.7-alpine

---> fc479af56697

Step 2 : WORKDIR /app

---> Using cache

---> d04d0d6d98f1

Step 3 : ADD requirements.txt /app/requirements.txt

---> Using cache

---> 9c4e311f3f0c

Step 4 : RUN pip install -r requirements.txt

---> Using cache

---> 85b878795479

Step 5 : ADD . /app

---> 63e3d4e6b539

Removing intermediate container 9af53c35d8fe

Step 6 : EXPOSE 80

---> Running in a5b3d3f80cd4

---> 4ce3750610a9

Removing intermediate container a5b3d3f80cd4

Step 7 : CMD gunicorn app:app -b 0.0.0.0:80 --log-file - --access-logfile - --workers 4 --keep-alive 0

---> Running in 0d69957bda4c

---> d41ff1635cb7

Removing intermediate container 0d69957bda4c

Successfully built d41ff1635cb7

counter\_nginx\_1 is up-to-date

counter\_redis\_1 is up-to-date

Recreating counter\_counter\_1

real 0m5.959s

user 0m0.508s

sys 0m0.076s

# 48.) [Docker - Error response from daemon: client is newer than server](https://stackoverflow.com/questions/34015271/docker-error-response-from-daemon-client-is-newer-than-server)

After creating a new machine with Docker Machine, I'm getting the following error:

$ docker ps

Error response from daemon: client is newer than server(client API version 1.21, server API version: 1.19)

How can I fix this?

-🡪 If someone happens to get this error, but is not using docker-machine, there is another way to resolve the issue by specifying an older API version in an environment variable on the client side:

export DOCKER\_API\_VERSION=<version>

for example:

export DOCKER\_API\_VERSION=1.19

and retrying the docker command.

# 49.) [Cannot link to a running container started by docker-compose](https://stackoverflow.com/questions/36489696/cannot-link-to-a-running-container-started-by-docker-compose)

 am setting up my local development environment with docker containers. The docker-compose.yml is like following

version: '2'

services:

db:

image: mongo:3

mq:

image: rabbitmq:3

api:

build: .

image: my\_app/api

ports:

- "3000:3000"

links:

- db

- mq

environment:

- NODE\_ENV=development

It starts without error. And docker lists 3 running containers

docker-compose up -d

docker ps

e90e5a8b5d33 my\_app/api "/usr/local/bin/node " 0.0.0.0:3000->3000/tcp my\_app\_api\_1

42bfcd971b16 mongo:3 "/entrypoint.sh mongo" 27017/tcp my\_app\_db\_1

a0685a816c47 rabbitmq:3 "/docker-entrypoint.s" 4369/tcp, 5671-5672/tcp, 25672/tcp my\_app\_mq\_1

However when I try to link to those running containers from another container

docker run --link my\_app\_mq\_1:mq --link my\_app\_db\_1:db -it worker

I get error

docker: Error response from daemon: Cannot link to /my\_app\_mq\_1, as it does not belong to the default network.

I have also tried

docker run --link my\_app\_mq\_1:mq --link my\_app\_db\_1:db -it --net default worker

Same error.

So how can I link to a running container started by docker-compose?

-🡪Ok, found the answer to it. In case anyone else comes across the same problem, just do

docker network ls

This command lists all the docker networks. docker-compose will create a new network when you run docker-compose up. In my case, the network is named as myapp\_default.

Note: Your app’s network is given a name based on the “project name”, which is based on the name of the directory it lives in. You can override the project name with either the --project-name flag or the COMPOSE\_PROJECT\_NAME environment variable. [Networking in Compose](https://docs.docker.com/compose/networking/)

So the correct way to link to those containers is

docker run --link my\_app\_mq\_1:mq --link my\_app\_db\_1:db -it --net myapp\_default worker

-----For those who didn't notice you have to put the --net before the command. I encountered this problem when trying to run the official redis docker image. – mistertee Jan 18 at 17:56

I had to use the --network flag, instead of the --net. Use it at the start of the run command. I hope this helps. Reference is available here: docs.docker.com/engine/reference/run/#network-settings.

-🡪When you use service definition version 2 and more, docker-compose creates user-defined network. Name resolution in a user-defined network works via Docker embedded DNS server. Here's related quote from [documentation](https://docs.docker.com/engine/userguide/networking/work-with-networks/):

The Docker embedded DNS server enables name resolution for containers connected to a given [user-defined] network. This means that any connected container can ping another container on the same network by its container name.

Containers are also available by network aliases that docker-compose creates. It can be verified by command like:

docker inspect \

-f '{{json .NetworkSettings.Networks.myapp\_default.Aliases}}' my\_app\_db\_1

It prints ["db","$CONTAINER\_ID"].

Providing links with --link will not have any effect in case of existing user-defined network. You can make sure and look at /etc/hosts, which will not have the corresponding lines.

Thus the following command is sufficient:

docker run -it --net myapp\_default worker

# 50.) [Auto-reloading of code changes with Django development in Docker with Gunicorn](https://stackoverflow.com/questions/32022014/auto-reloading-of-code-changes-with-django-development-in-docker-with-gunicorn)

I'm using a Docker container for Django development, and the container runs Gunicorn with Nginx. I'd like code changes to auto-load, but the only way I can get them to load is by rebuilding with docker-compose (docker-compose build). The problem with "build" is that it re-runs all my pip installs.

I'm using the Gunicorn --reload flag, which is apparently supposed to do what I want. Here are my Docker config files:

## Dockerfile:

FROM python:3.4.3

RUN mkdir /code

WORKDIR /code

ADD . /code/

RUN pip install -r /code/requirements/docker.txt

## docker-compose.yml:

web:

restart: always

build: .

expose:

- "8000"

links:

- postgres:postgres

volumes:

- /usr/src/app/static

env\_file: .env

command: /usr/local/bin/gunicorn myapp.wsgi:application -w 2 -b :8000 --reload

nginx:

restart: always

build: ./config/nginx

ports:

- "80:80"

volumes:

- /www/static

volumes\_from:

- web

links:

- web:web

postgres:

restart: always

image: postgres:latest

volumes:

- /var/lib/postgresql

ports:

- "5432:5432"

I've tried some of the other Docker commands (docker-compose restart, docker-compose up), but the code won't refresh.

What am I missing?

-🡪I looked into having a volume for my code, and after looking at the [Docker Compose recommended Django setup](https://docs.docker.com/compose/django/), I added volumes: - .:/code to my web container in docker-compose.yml, and now any code changes I make automatically apply.

## docker-compose.yml:

web:

restart: always

build: .

expose:

- "8000"

links:

- postgres:postgres

volumes:

- /usr/src/app/static

- .:/code

env\_file: .env

command: /usr/local/bin/gunicorn myapp.wsgi:application -w 2 -b :8000 --reload

*Update:* for a thorough example of using Gunicorn and Django with Docker, checkout this [example project from Rackspace](https://github.com/rackerlabs/guestbook), which also shows how to use docker-machine to launch the setup on remote servers like Rackspace Cloud.

**Caveat:** currently, this method does not work when your code is locally and the docker host is remote (e.g., on a cloud provider like Digital Ocean or Rackspace). This also applies to virtual machines if your local file system is not mounted on the VM. Note that there are separate volume drivers (e.g., flocker), and there *might* be something out there to address this need. ~~For now, the "fix" is to rsync/scp your files up to a directory on the remote docker host. Then, the --reload flag will auto-reload gunicorn after any scp/rsync.~~ **Update:** If pushing code to remove docker host, I find it far easier to just rebuild the docker container (e.g., docker-compose build web && docker-compose up -d). This can be slower though than the rsync approach if your src folder is large.

-🡪You have another problem- Docker caches each layer that it builds. You shouldn't have to re-run pip install every time!

ADD . /code/

RUN pip install -r /code/requirements/docker.txt

This is your problem- Docker checks every ADD statement to see if any files have changed and invalidates the cache for it and every later step if it has. The correct way to do this is...

ADD ./requirements/docker.txt /code/requirements/

RUN pip install -r /code/requirements/docker.txt

ADD ./code/

Which will only invalidate your pip install line if your requirements file changes!

# 51.) [Shared folder in Docker. With Windows. Not only “C/user/” path](https://stackoverflow.com/questions/33966225/shared-folder-in-docker-with-windows-not-only-c-user-path)

I'm new to Docker, I come from Vagrant.

I'm using Docker (1.9.1) inside my "D:/Works/something/DockerFirstTime" folder.

Now I create the machine with

docker-machine create first

and simple **Dockerfile**:

FROM ruby:2.2-onbuild

and simple **Gemfile**:

source 'https://rubygems.org'

gem 'rails'

Now with this command I want to use a shared folder like in Vagrant in the same hard drive of my Dockerfile:

docker run -it -v //d/Works/something/DockerFirstTime:/usr/src/app -w /usr/src/app ruby:2.2 bundle install

But it doesn't works.

How to do this?

I know that Docker only shares the **/c/User/folder**, is that right?

How can I use the folder with the files and modify my files with editor in Windows and then restart server like in a normal shell on a single PC or like in Vagrant?

-🡪 This question and [this question](https://stackoverflow.com/questions/33807891/windows-boot2docker-how-to-add-d-drive-to-be-accessible-from-within-docker/33988307) have a similar root problem, mounting a non C:/ drive folder in boot2docker. I wrote an in-depth answer to the other question that provide the same information that is in the first half of @VonC's answer.

From [Docker Docs](https://docs.docker.com/engine/userguide/dockervolumes/#mount-a-host-directory-as-a-data-volume):

All other paths come from your virtual machine’s filesystem. [...] In the case of VirtualBox you need to make the host folder available as a shared folder in VirtualBox. Then, you can mount it using the Docker -v flag.

**To get your folder mounted in a container:**

This mounts your entire D:\ drive, you can simply change the file paths to be more granular and specific.

Share the directory with VBox:

This only needs to be done once.

In windows CMD:

VBoxManage sharedfolder add "boot2docker-vm" --name "d-share" --hostpath "D:\"

Mount the shared directory in your VM:

This will need to be done each time you restart the VM.

# [Docker Compose wait for container X before starting Y](https://stackoverflow.com/questions/31746182/docker-compose-wait-for-container-x-before-starting-y)

mount -t vboxsf -o uid=1000,gid=50 d-share /d

To see sources and explanation for how this works see [my full answer to the other similar question](https://stackoverflow.com/questions/33807891/windows-boot2docker-how-to-add-d-drive-to-be-accessible-from-within-docker/33988307#33988307)

After this you can use the -v/--volume flag in Docker to mount this folder or any sub-folders or files into containers. If you mounted your whole D:\ drive you can use that exact docker run command from your question and it should now work. If you mounted a specific part of you drive you will have to change the paths to match.

To edit in windows, run in docker:

Also from [Docker Docs](https://docs.docker.com/engine/userguide/dockervolumes/#mount-a-host-directory-as-a-data-volume):

Mounting a host directory can be useful for testing. For example, you can mount source code inside a container. Then, change the source code and see its effect on the application in real time.

As a VBox shared directory you should be able to see changes made from the Windows side reflected in the boot2docker vm.

You may need to restart containers to see the changes actually appear, this depends on how the program running inside the container, in your case ruby, uses the files. If the files are compiled into an app when the container starts, for example, you will definitely need to restart the container to see the changes.

Note:

Beware the CR LF vs. LF line ending difference when writing files in Windows and reading them in Linux. Make sure your text editor is saving files with Unix line endings or else you may start to see errors caused by '^M' appended to the end of all your lines.

# 52.) [How to mount local volumes in docker machine](https://stackoverflow.com/questions/30040708/how-to-mount-local-volumes-in-docker-machine)

I am trying to use docker-machine with docker-compose. The file docker-compose.yml has definitions as follows:

web:

build: .

command: ./run\_web.sh

volumes:

- .:/app

ports:

- "8000:8000"

links:

- db:db

- rabbitmq:rabbit

- redis:redis

When running docker-compose up -d all goes well until trying to execute the command and an error is produced:

Cannot start container b58e2dfa503b696417c1c3f49e2714086d4e9999bd71915a53502cb6ef43936d: [8] System error: exec: "./run\_web.sh": stat ./run\_web.sh: no such file or directory

Local volumes are not mounted to the remote machine. Whats the recommended strategy to mount the local volumes with the webapps' code?

-🡪 Also ran into this issue and it looks like local volumes are not mounted when using docker-machine. A hack solution is to

1. get the current working directory of the docker-machine instance docker-machine ssh <name> pwd
2. use a command line tool like rsync to copy folder to remote system
3. rsync -avzhe ssh --progress <name\_of\_folder> username@remote\_ip:<result \_of\_pwd\_from\_1>.

The default pwd is /root so the command above would be rsync -avzhe ssh --progress <name\_of\_folder> username@remote\_ip:/root

NB: you would need to supply the password for the remote system. You can quickly create one by ssh into the remote system and creating a password.

1. change the volume mount point in your docker-compose.yml file from .:/app to /root/<name\_of\_folder>:/app
2. run docker-compose up -d

NB when changes are made locally, don't forget to rerun rsync to push the changes to the remote system.

Its not perfect but it works. An issue is ongoing <https://github.com/docker/machine/issues/179>

Other project that attempt to solve this include [docker-rsync](https://github.com/synack/docker-rsync)

---- Created a script for the solution mentioned here. Works on latest docker 1.10 and docker-machine 0.6.0 gist.github.com/cristobal/fcb0987871d7e1f7449e

@cristobal it looks like you scripted the mount solution, not the rsync solution? – Andy Feb 17 '16 at 14:18

# 53.) [Docker-compose: node\_modules not present in a volume after npm install succeeds](https://stackoverflow.com/questions/30043872/docker-compose-node-modules-not-present-in-a-volume-after-npm-install-succeeds)

I have an app with the following services:

* web/ - holds and runs a python 3 flask web server on port 5000. Uses sqlite3.
* worker/ - has an index.js file which is a worker for a queue. the web server interacts with this queue using a json API over port 9730. The worker uses redis for storage. The worker also stores data locally in the folder worker/images/

Now this question only concerns the worker.

worker/Dockerfile

FROM node:0.12

WORKDIR /worker

COPY package.json /worker/

RUN npm install

COPY . /worker/

docker-compose.yml

redis:

image: redis

worker:

build: ./worker

command: npm start

ports:

- "9730:9730"

volumes:

- worker/:/worker/

links:

- redis

When I run docker-compose build, everything works as expected and all npm modules are installed in /worker/node\_modules as I'd expect.

npm WARN package.json unfold@1.0.0 No README data

> phantomjs@1.9.2-6 install /worker/node\_modules/pageres/node\_modules/screenshot-stream/node\_modules/phantom-bridge/node\_modules/phantomjs

> node install.js

<snip>

But when I do docker-compose up, I see this error:

worker\_1 | Error: Cannot find module 'async'

worker\_1 | at Function.Module.\_resolveFilename (module.js:336:15)

worker\_1 | at Function.Module.\_load (module.js:278:25)

worker\_1 | at Module.require (module.js:365:17)

worker\_1 | at require (module.js:384:17)

worker\_1 | at Object.<anonymous> (/worker/index.js:1:75)

worker\_1 | at Module.\_compile (module.js:460:26)

worker\_1 | at Object.Module.\_extensions..js (module.js:478:10)

worker\_1 | at Module.load (module.js:355:32)

worker\_1 | at Function.Module.\_load (module.js:310:12)

worker\_1 | at Function.Module.runMain (module.js:501:10)

Turns out none of the modules are present in /worker/node\_modules (on host or in the container).

If on the host, I npm install, then everything works just fine. But I don't want to do that. I want the container to handle dependencies.

What's going wrong here?

(Needless to say, all packages are in package.json.)

--🡪 This happens because you have added your worker directory as a volume to your docker-compose.yml, as the volume is not mounted during the build.

When docker builds the image, the node\_modules directory is created within the worker directory, and all the dependencies are installed there. Then on runtime the worker directory from outside docker is mounted into the docker instance (which does not have the installed node\_modules), hiding the node\_modules you just installed. You can verify this by removing the mounted volume from your docker-compose.yml.

A workaround is to use a data volume to store all the node\_modules, as data volumes copy in the data from the built docker image before the worker directory is mounted. This can be done in the docker-compose.yml like this:

redis:

image: redis

worker:

build: ./worker

command: npm start

ports:

- "9730:9730"

volumes:

- worker/:/worker/

- /worker/node\_modules

links:

- redis

I'm not entirely certain whether this imposes any issues for the portability of the image, but as it seems you are primarily using docker to provide a runtime environment, this should not be an issue.

If you want to read more about volumes, there is a nice user guide available here: <https://docs.docker.com/userguide/dockervolumes/>

-🡪 I recently had a similar problem. You can install node\_modules elsewhere and set the NODE\_PATHenvironment variable.

In the example below I installed node\_modules into /install

worker/Dockerfile

FROM node:0.12

RUN ["mkdir", "/install"]

ADD ["./package.json", "/install"]

WORKDIR /install

RUN npm install --verbose

ENV NODE\_PATH=/install/node\_modules

WORKDIR /worker

COPY . /worker/

docker-compose.yml

redis:

image: redis

worker:

build: ./worker

command: npm start

ports:

- "9730:9730"

volumes:

- worker/:/worker/

links:

- redis

# 55.) [How do I seed a mongo database using docker-compose?](https://stackoverflow.com/questions/31210973/how-do-i-seed-a-mongo-database-using-docker-compose)

I am trying to distribute a set of connected applications running in several linked containers that includes a mongo database that is required to:

* be distributed containing some seed data;
* allow users to add additional data.

Ideally the data will also be persisted in a linked data volume container.

I can get the data into the mongo container using a mongo base instance that doesn't mount any volumes (dockerhub image: psychemedia/mongo\_nomount - this is essentially the base mongo Dockerfile without the VOLUME /data/db statement) and a Dockerfile config along the lines of:

ADD . /files

WORKDIR /files

RUN mkdir -p /data/db && mongod --fork --logpath=/tmp/mongodb.log && sleep 20 && \

mongoimport --db testdb --collection testcoll --type csv --headerline --file ./testdata.csv #&& mongod --shutdown

where ./testdata.csv is in the same directory (./mongo-with-data) as the Dockerfile.

My docker-compose config file includes the following:

mongo:

#image: mongo

build: ./mongo-with-data

ports:

- "27017:27017"

#Ideally we should be able to mount this against a host directory

#volumes:

# - ./db/mongo/:/data/db

#volumes\_from:

# - devmongodata

#devmongodata:

# command: echo created

# image: busybox

# volumes:

# - /data/db

Whenever I try to mount a VOLUME it seems as if the original seeded data - which is stored in /data/db - is deleted. I guess that when a volume is mounted to /data/db it replaces whatever is there currently.

That said, the [docker userguide](https://docs.docker.com/userguide/dockervolumes/) suggests that: *Volumes are initialized when a container is created. If the container’s base image contains data at the specified mount point, that existing data is copied into the new volume upon volume initialization*? So I expected the data to persist if I placed the VOLUME command after the seeding RUN command?

So what am I doing wrong?

The long view is that I want to automate the build of several linked containers, and then distribute a Vagrantfile/docker-compose YAML file that will fire up a set of linked apps, that includes a pre-seeded mongo database with a (partially pre-populated) persistent data container.

--🡪 I do this using another docker container whose only purpose is to seed mongo, then exit. I suspect this is the same idea as [ebaxt](https://stackoverflow.com/a/32666074/317951)'s, but when I was looking for an answer to this, I just wanted to see a quick-and-dirty, yet straightforward, example. So here is mine:

**docker-compose.yml**

mongodb:

image: mongo

ports:

- "27017:27017"

mongo-seed:

build: ./mongo-seed

links:

- mongodb

# my webserver which uses mongo (not shown in example)

webserver:

build: ./webserver

ports:

- "80:80"

links:

- mongodb

**mongo-seed/Dockerfile**

FROM mongo

COPY init.json /init.json

CMD mongoimport --host mongodb --db reach-engine --collection MyDummyCollection --type json --file /init.json --jsonArray

**mongo-seed/init.json**

[

{

"name": "Joe Smith",

"email": "jsmith@gmail.com",

"age": 40,

"admin": false

},

{

"name": "Jen Ford",

"email": "jford@gmail.com",

"age": 45,

"admin": true

}

]

# 56.) [How to get docker-compose to always re-create containers from fresh images?](https://stackoverflow.com/questions/32612650/how-to-get-docker-compose-to-always-re-create-containers-from-fresh-images)

My docker images are built on a Jenkins CI server and are pushed to our private Docker Registry. My goal is to provision environments with docker-compose which always start the originally built state of the images.

I am currently using docker-compose 1.3.2 as well as 1.4.0 on different machines but we also used older versions previously.

I always used the docker-compose pull && docker-compose up -d commands to fetch the fresh images from the registry and start them up. I believe my preferred behaviour was working as expected up to a certain point in time, but since then docker-compose up started to re-run previously stopped containers instead of starting the originally built images every time.

Is there a way to get rid of this behaviour? Could that way be one which is wired in the docker-compose.yml configuration file to not depend "not forgetting" something on the command line upon every invocation?

ps. Besides finding a way to achieve my goal, I would also love to know a bit more about the background of this behaviour. I think the basic idea of Docker is to build an immutable infrastructure. The current behaviour of docker-compose just seem to plain clash with this approach.. or do I miss some points here?

-🡪 docker-compose up --force-recreate is one option, but if you're using it for CI, I would start the build with docker-compose rm -f to stop and remove the containers and volumes (then follow it with pull and up).

This is what I use:

docker-compose rm -f

docker-compose pull

docker-compose up --build -d

# Run some tests

./tests

docker-compose stop -t 1

The reason containers are recreated is to preserve any data volumes that might be used (and it also happens to make up a lot faster).

If you're doing CI you don't want that, so just removing everything should get you want you want.

Update: use up --build which was added in docker-compose 1.7

--🡪 By current official [documentation](https://docs.docker.com/compose/reference/down/) there is a short cut that stops and removes containers, networks, volumes, and images created by up, if they are already stopped or partially removed and so on, then it will do the trick too:

docker-compose down

Then if you have new changes on your images or Dockerfiles use:

docker-compose build --no-cache

Finally:docker-compose up

In one command: docker-compose down && docker-compose build --no-cache && docker-compose up

-🡪 he only solution that worked for me was this command :

docker-compose build --no-cache

This will automatically pull fresh image from repo and won't use the cache version that is prebuild with any parameters you've been using before

-🡪 You can pass --force-recreate to docker compose up, which should use fresh containers.

I think the reasoning behind reusing containers is to preserve any changes during development. Note that Compose does something similar with volumes, which will also persist between container recreation (a recreated container will attach to its predecessor's volumes). This can be helpful, for example, if you have a Redis container used as a cache and you don't want to lose the cache each time you make a small change. At other times it's just confusing.

I don't believe there is any way you can force this from the Compose file.

Arguably it does clash with immutable infrastructure principles. The counter-argument is probably that you don't use Compose in production (yet). Also, I'm not sure I agree that immutable infra is the basic idea of Docker, although it's certainly a good use case/selling point.

# 57.) [How to install docker-compose on Windows](https://stackoverflow.com/questions/29289785/how-to-install-docker-compose-on-windows)

* install [docker-machine](https://docs.docker.com/installation/windows/);
* install [python](https://www.python.org/downloads) (3.4.3 worked fine)
* install [pip](https://pypi.python.org/pypi/pip);
* after pip... run this command to install docker-compose:

`pip install git+git://github.com/docker/compose.git`

# 58.) [How to use environment variables in docker compose](https://stackoverflow.com/questions/29377853/how-to-use-environment-variables-in-docker-compose)

I would like to be able to use env variables inside docker-compose.yml, with values passed in at the time of docker-compose up. This is the example. I am doing this today with basic docker run command, which is wrapped around my own script. Is there a way to achieve it with compose, without any such bash wrappers?

proxy:

hostname: $hostname

volumes:

- /mnt/data/logs/$hostname:/logs

- /mnt/data/$hostname:/data

1. -🡪 Create a template.yml, which is your docker-compose.yml with environment variable.
2. Suppose your environment variables are in a file 'env.sh'
3. Put the below piece of code in a sh file and run it.

source env.sh; rm -rf docker-compose.yml; envsubst < "template.yml" > "docker-compose.yml";

A new file docker-compose.yml will be generated with the correct values of environment variables.

Sample template.yml file:

oracledb:

image: ${ORACLE\_DB\_IMAGE}

privileged: true

cpuset: "0"

ports:

- "${ORACLE\_DB\_PORT}:${ORACLE\_DB\_PORT}"

command: /bin/sh -c "chmod 777 /tmp/start; /tmp/start"

container\_name: ${ORACLE\_DB\_CONTAINER\_NAME}

Sample env.sh file:

#!/bin/bash

export ORACLE\_DB\_IMAGE=<image-name>

export ORACLE\_DB\_PORT=<port to be exposed>

export ORACLE\_DB\_CONTAINER\_NAME=ORACLE\_DB\_SERVER

# --🡪 The DOCKER solution:

It looks like docker-compose 1.5+ has enabled variables substitution: <https://github.com/docker/compose/releases>

The latest Docker Compose allows you to access environment variables from your compose file. So you can source your environment variables, then run Compose like so:

set -a

source .my-env

docker-compose up -d

Then you can reference the variables in docker-compose.yml using ${VARIABLE}, like so:

db:

image: "postgres:${POSTGRES\_VERSION}"

And here is more info from the docs, taken here: <https://docs.docker.com/compose/compose-file/#variable-substitution>

When you run docker-compose up with this configuration, Compose looks for the POSTGRES\_VERSION environment variable in the shell and substitutes its value in. For this example, Compose resolves the image to postgres:9.3 before running the configuration.

If an environment variable is not set, Compose substitutes with an empty string. In the example above, if POSTGRES\_VERSION is not set, the value for the image option is postgres:.

Both $VARIABLE and ${VARIABLE} syntax are supported. Extended shell-style features, such as ${VARIABLE-default} and ${VARIABLE/foo/bar}, are not supported.

If you need to put a literal dollar sign in a configuration value, use a double dollar sign ($$).

And I believe this feature was added in this pull request: <https://github.com/docker/compose/pull/1765>

**The BASH solution:**

I notice folks have issues with Docker's environment variables support. Instead of dealing with environment variables in Docker, let's go back to basics, like bash! Here is a more flexible method using a bash script and a .env file.

An example .env file:

EXAMPLE\_URL=http://example.com

# Note that the variable below is commented out and will not be used:

# EXAMPLE\_URL=http://example2.com

SECRET\_KEY=ABDFWEDFSADFWWEFSFSDFM

# You can even define the compose file in an env variable like so:

COMPOSE\_CONFIG=my-compose-file.yml

# You can define other compose files, and just comment them out

# when not needed:

# COMPOSE\_CONFIG=another-compose-file.yml

then run this bash script in the same directory, which should deploy everything properly:

#!/bin/bash

docker rm -f `docker ps -aq -f name=myproject\_\*`

set -a

source .env

cat ${COMPOSE\_CONFIG} | envsubst | docker-compose -f - -p "myproject" up -d

Just reference your env variables in your compose file with the usual bash syntax (ie ${SECRET\_KEY} to insert the SECRET\_KEY from the .env file).

Note the COMPOSE\_CONFIG is defined in my .env file and used in my bash script, but you can easily just replace {$COMPOSE\_CONFIG} with the my-compose-file.yml in the bash script.

Also note that I labeled this deployment by naming all of my containers with the "myproject" prefix. You can use any name you want, but it helps identify your containers so you can easily reference them later. Assuming that your containers are stateless, as they should be, this script will quickly remove and redeploy your containers according to your .env file params and your compose YAML file.

**Update** Since this answer seems pretty popular, I wrote a blog post that describes my Docker deployment workflow in more depth: <http://lukeswart.net/2016/03/lets-deploy-part-1/> This might be helpful when you add more complexity to a deployment configuration, like nginx configs, LetsEncrypt certs, and linked containers.

# 59.) [How to correctly link php-fpm and Nginx Docker containers?](https://stackoverflow.com/questions/29905953/how-to-correctly-link-php-fpm-and-nginx-docker-containers)

I am trying to link 2 separate containers:

* [nginx:latest](https://registry.hub.docker.com/_/nginx/)
* [php:fpm](https://registry.hub.docker.com/_/php/)

The problem is that php scripts do not work. Perhaps the php-fpm configuration is incorrect. Here is the source code, which is in my [repository](https://github.com/bocharsky-bw/docker). Here is the file docker-compose.yml:

nginx:

build: .

ports:

- "80:80"

- "443:443"

volumes:

- ./:/var/www/test/

links:

- fpm

fpm:

image: php:fpm

ports:

- "9000:9000"

and Dockerfile which I used to build a custom image based on the nginx one:

FROM nginx

# Change Nginx config here...

RUN rm /etc/nginx/conf.d/default.conf

ADD ./default.conf /etc/nginx/conf.d/

Lastly, here is my custom Nginx virtual host config:

server {

listen 80;

server\_name localhost;

root /var/www/test;

error\_log /var/log/nginx/localhost.error.log;

access\_log /var/log/nginx/localhost.access.log;

location / {

# try to serve file directly, fallback to app.php

try\_files $uri /index.php$is\_args$args;

}

location ~ ^/.+\.php(/|$) {

fastcgi\_pass 192.168.59.103:9000;

fastcgi\_split\_path\_info ^(.+\.php)(/.\*)$;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_param HTTPS off;

}

}

Could anybody help me configure these containers correctly to execute php scripts?

**P.S.** I run containers via docker-composer like this:

docker-compose up

from the project root directory.

--🡪 Don't hardcode ip of containers in nginx config, docker link adds the hostname of the linked machine to the hosts file of the container and you should be able to ping by hostname.

EDIT: Docker 1.9 Networking no longer requires you to link containers, when multiple containers are connected to the same network, their hosts file will be updated so they can reach each other by hostname.

Every time a docker container spins up from an image (even stop/start-ing an existing container) the containers get new ip's assigned by the docker host. These ip's are not in the same subnet as your actual machines.

[see docker linking docs](http://docs.docker.com/userguide/dockerlinks/#container-linking) (this is what compose uses in the background)

[but more clearly explained in the docker-compose docs on links & expose](http://docs.docker.com/compose/yml/#links)

**links**

links:

- db

- db:database

- redis

An entry with the alias' name will be created in /etc/hosts inside containers for this service, e.g:

172.17.2.186 db

172.17.2.186 database

172.17.2.187 redis

**expose**

Expose ports **without publishing them to the host machine** - they'll only be **accessible to linked services**. Only the internal port can be specified.

and if you set up your project to get the ports + other credentials through environment variables, [links automatically set a bunch of system variables](http://docs.docker.com/compose/env/):

To see what environment variables are available to a service, run docker-compose run SERVICE env.

name\_PORT

Full URL, e.g. DB\_PORT=tcp://172.17.0.5:5432

name\_PORT\_num\_protocol

Full URL, e.g. DB\_PORT\_5432\_TCP=tcp://172.17.0.5:5432

name\_PORT\_num\_protocol\_ADDR

Container's IP address, e.g. DB\_PORT\_5432\_TCP\_ADDR=172.17.0.5

name\_PORT\_num\_protocol\_PORT

Exposed port number, e.g. DB\_PORT\_5432\_TCP\_PORT=5432

name\_PORT\_num\_protocol\_PROTO

Protocol (tcp or udp), e.g. DB\_PORT\_5432\_TCP\_PROTO=tcp

name\_NAME

Fully qualified container name, e.g. DB\_1\_NAME=/myapp\_web\_1/myapp\_db\_1

## --🡪 New Answer

Docker Compose has been updated. They now have a [version 2 file format](https://github.com/docker/compose/blob/master/docs/compose-file.md#versioning).

Version 2 files are supported by Compose 1.6.0+ and require a Docker Engine of version 1.10.0+.

They now support the networking feature of Docker which when run sets up a default network called *myapp\_default*

From [their documentation](https://github.com/docker/compose/blob/master/docs/networking.md) your file would look something like the below:

version: '2'

services:

web:

build: .

ports:

- "8000:8000"

fpm:

image: phpfpm

nginx

image: nginx

As these containers are automatically added to the default *myapp\_default* network they would be able to talk to each other. You would then have in the Nginx config:

fastcgi\_pass fpm:9000;

Also as mentioned by @treeface in the comments remember to ensure PHP-FPM is listening on port 9000, this can be done by editing /etc/php5/fpm/pool.d/www.conf where you will need listen = 9000.

Old Answer

I have kept the below here for those using older version of Docker/Docker compose and would like the information.

I kept stumbling upon this question on google when trying to find an answer to this question but it was not quite what I was looking for due to the Q/A emphasis on docker-compose (which at the time of writing only has experimental support for docker networking features). So here is my take on what I have learnt.

Docker has recently [deprecated its link](http://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/) feature in favour of its [networks feature](http://docs.docker.com/engine/userguide/networking/dockernetworks/)

Therefore using the Docker Networks feature you can link containers by following these steps. For full explanations on options read up on the docs linked previously.

First create your network

docker network create --driver bridge mynetwork

Next run your PHP-FPM container ensuring you open up port 9000 and assign to your new network (mynetwork).

docker run -d -p 9000 --net mynetwork --name php-fpm php:fpm

The important bit here is the --name php-fpm at the end of the command which is the name, we will need this later.

Next run your Nginx container again assign to the network you created.

docker run --net mynetwork --name nginx -d -p 80:80 nginx:latest

For the PHP and Nginx containers you can also add in --volumes-from commands etc as required.

Now comes the Nginx configuration. Which should look something similar to this:

server {

listen 80;

server\_name localhost;

root /path/to/my/webroot;

index index.html index.htm index.php;

location / {

try\_files $uri $uri/ /index.php?$query\_string;

}

location ~ \.php$ {

fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

fastcgi\_pass php-fpm:9000;

fastcgi\_index index.php;

include fastcgi\_params;

}

}

Notice the fastcgi\_pass php-fpm:9000; in the location block. Thats saying contact container php-fpm on port 9000. When you add containers to a Docker bridge network they all automatically get a hosts file update which puts in their container name against their IP address. So when Nginx sees that it will know to contact the PHP-FPM container you named php-fpm earlier and assigned to your mynetwork Docker network.

You can add that Nginx config either during the build process of your Docker container or afterwards its up to you.

# 60.) [how to link docker container to each other with docker-compose](https://stackoverflow.com/questions/29307645/how-to-link-docker-container-to-each-other-with-docker-compose)

|  |  |
| --- | --- |
|  | I have to setup a mongo replica set with docker-compose. For the replica set the containers have to know each other.  I tried in docker-compose.yml  dbreplicasetpart1:  image: mongo:2.6.8  expose:  - '27018'  links:  - replicasetpart2  - replicasetpart3  cap\_add:  - NET\_ADMIN  dbreplicasetpart2:  image: mongo:2.6.8  links:  - replicasetpart1  - replicasetpart3  expose:  - '27019'  cap\_add:  - NET\_ADMIN  ...  I get an circular import message. But if I remove the back-link to dbreplicasetpart1 I can't ping from dbreplicasetpart2 to dbreplicasetpart1. What is the solution?  --🡪 **Updated for Docker 1.10**  Docker 1.10 allows the definition of networks within the compose file. Here's the updated code  version: "2"  services:  replica1:  image: mongo:2.6.8  container\_name: replica1  networks:  - my-net  ports:  - "27018"  environment:  REPLICA2\_URL: "http://replica2:27019"  replica2:  image: mongo:2.6.8  container\_name: replica2  networks:  - my-net  ports:  - "27019"  environment:  REPLICA1\_URL: "http://replica1:27018"  networks:  my-net:  driver: bridge  **Previous answer for Docker 1.9**  As of Docker 1.9, the solution to this is to create a custom network and pass it to the docker-compose up command.   1. Create a network  docker network create --driver bridge my-net 2. Reference that network as an environment variable (${NETWORK})in the docker-compose.yml files. Eg:   ```  replica1:  image: mongo:2.6.8  container\_name: replica1  net: ${NETWORK}  ports:  - "27018"  environment:  REPLICA2\_URL: "http://replica2:27019"  replica2:  image: mongo:2.6.8  container\_name: replica2  net: ${NETWORK}  ports:  - "27019"  environment:  REPLICA1\_URL: "http://replica1:27018"  ```  Note that replica1 in http://replica1:27018 will resolve to the ip address of the replica1 service (container). No need to hardcode ip addresses; An entry for replica1 is automatically added to the /etc/host of the replica2 container. Same goes for the replica1 container. Docker will add an entry for replica2 in its /etc/host file.   1. Call docker-compose, passing it the network you created  NETWORK=my-net docker-compose up -d -f docker-compose.yml   I've created a [**bridge network**](https://docs.docker.com/engine/userguide/networking/dockernetworks/#a-bridge-network) above which only works within one node (host). Good for dev. If you need to get two nodes to talk to each other, you need to create an [**overlay network**](https://docs.docker.com/engine/userguide/networking/get-started-overlay/). Same principle though. You pass the network name to the docker-compose up command. |