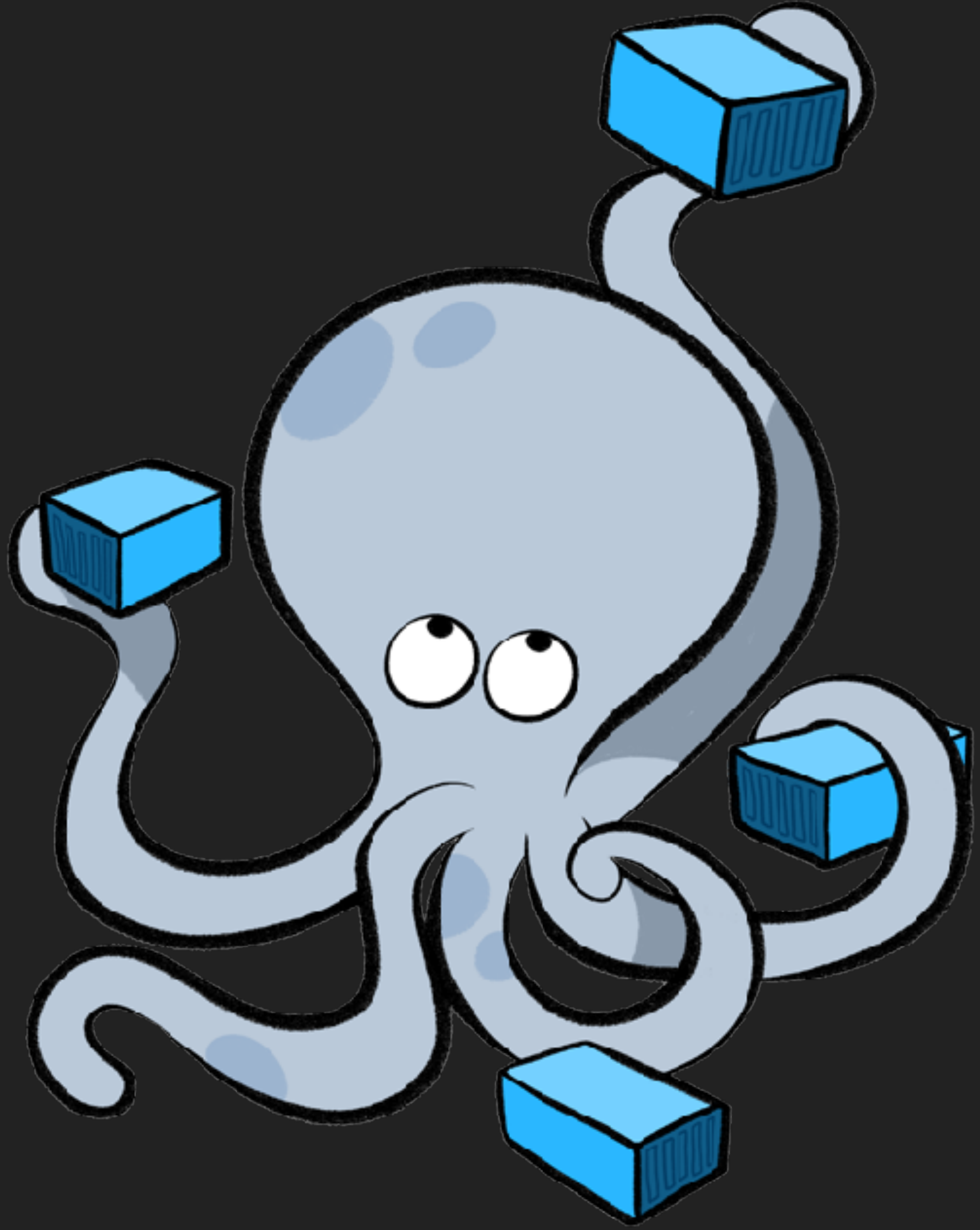
CREATE A DEVELOPMENT ENVIRONMENT WITH

# DOCKER COMPOSE



## WHAT IS A DOCKER CONTAINER?

- Is a way to package your application into a standardized unit for Software Development.
- Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries – anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment. —https://www.docker.com/what-docker

## WHAT IS DOCKER COMPOSE?

- Compose is a tool for defining and running multi-container Docker applications. https://docs.docker.com/compose/'
- Its about arranging containers into logical services.  Outlining dependencies and making sure that the appropriate containers are booted.

## WHY USE IT?

- I use Homebrew.
- What about a regular VM?  Vagrant?
- Basically it comes down to one question.

## LETS WRITE A DOCKERFILE

```
FROM ruby:2.3.3
RUN apt-get update -qq
RUN apt-get install -y build-essential
RUN apt-get install -y nodejs
RUN apt-get install -y npm
RUN apt-get install -y nodejs-legacy
RUN apt-get install -y mysql-client libmysqlclient-dev
RUN mkdir /apps
RUN bundle config --global silence_root_warning 1
ADD Gemfile /apps/test/Gemfile
ADD Gemfile.lock /apps/test/Gemfile.lock
WORKDIR /apps/test
RUN bundle install
```

## WHAT HAPPENS WHEN WE RUN IT?

```
⚡ docker-compose build
Sending build context to Docker daemon 139.8 kB
Step 1/8 : FROM ruby:2.3.3
2.3.3: Pulling from library/ruby
5040bd298390: Pull complete
fce5728aad85: Pull complete
76610ec20bf5: Pull complete
52f3db4b5710: Pull complete
73c0dc2c700a: Pull complete
2be97eb0fc33: Pull complete
a385cee59a33: Pull complete
d6108f8be33c: Pull complete
Digest: sha256:523b6d221531ce9e3d418a3074fb879ec0c0b7435f731c7010b81e6c0b3e0c69
Status: Downloaded newer image for ruby:2.3.3
 ---> d089d4d3e81c
Step 2/8 : RUN apt-get update -qq
 ---> Running in cabad8452c76
 ---> 9faacddb076b
Removing intermediate container cabad8452c76
Step 3/8 : RUN apt-get install -y build-essential
 ---> Running in b685324572f4
```

## WHAT IF WE CHANGE IT?

```
⚡ docker-compose build
Sending build context to Docker daemon 140.3 kB
Step 1/9 : FROM ruby:2.3.3
 ---> d089d4d3e81c
Step 2/9 : RUN apt-get update -qq
 ---> Using cache
 ---> 9faacddb076b
Step 3/9 : RUN apt-get install -y build-essential
 ---> Using cache
 ---> 7c74058e1258
Step 4/9 : RUN apt-get install -y nodejs
 ---> Using cache
 ---> 1d2da1e228e1
Step 5/9 : RUN apt-get install -y npm
 ---> Using cache
 ---> eaa8567e0ffb
Step 6/9 : RUN apt-get install -y nodejs-legacy
 ---> Using cache
 ---> 98508d08b7f6
Step 7/9 : RUN apt-get install -y mysql-client libmysqlclient-dev
 ---> Using cache
 ---> 1b2f05a32929
Step 8/9 : RUN mkdir /apps
 ---> Using cache
 ---> f7297217e006
Step 9/9 : RUN bundle config --global silence_root_warning 1
 ---> Running in af292817c386
You have a bundler environment variable for silence_root_warning set to "1". This will take precedence over the global value you are setting
 ---> 586ed297429e
Removing intermediate container af292817c386
Successfully built 586ed297429e
```

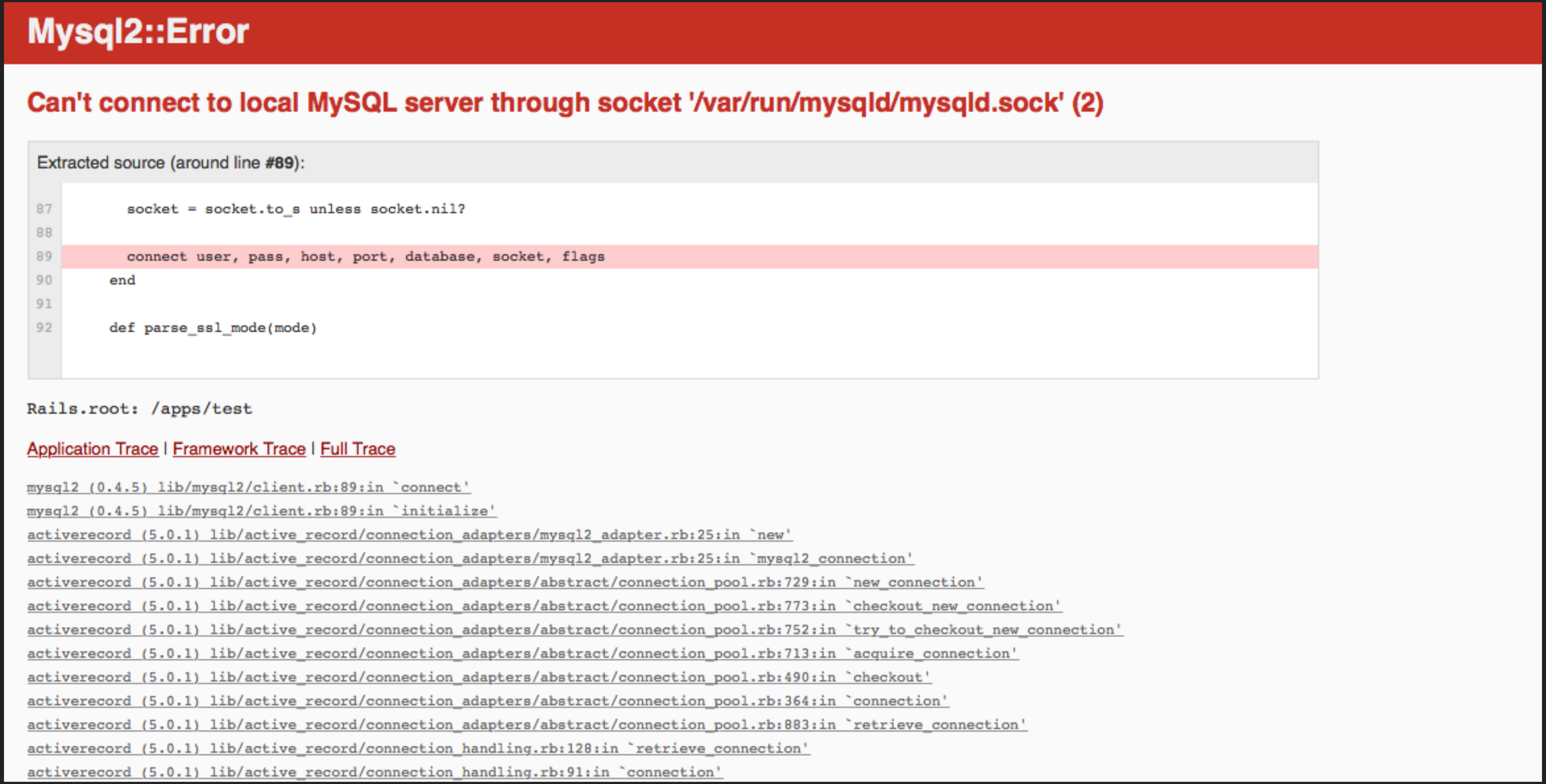
## WHAT IF WE CHANGE IT?

```
⚡ docker-compose build
Sending build context to Docker daemon 140.3 kB
Step 1/9 : FROM ruby:2.3.3
 ---> d089d4d3e81c
Step 2/9 : RUN apt-get update -qq
 ---> Using cache
 ---> 9faacddb076b
Step 3/9 : RUN apt-get install -y build-essential
 ---> Using cache
 ---> 7c74058e1258
Step 4/9 : RUN apt-get install -y nodejs
 ---> Using cache
 ---> 1d2da1e228e1
Step 5/9 : RUN apt-get install -y npm
 ---> Using cache
 ---> eaa8567e0ffb
Step 6/9 : RUN apt-get install -y nodejs-legacy
 ---> Using cache
 ---> 98508d08b7f6
Step 7/9 : RUN apt-get install -y mysql-client libmysqlclient-dev
 ---> Using cache
 ---> 1b2f05a32929
Step 8/9 : RUN mkdir /apps
 ---> Using cache
 ---> f7297217e006
Step 9/9 : RUN bundle config --global silence_root_warning 1
 ---> Running in af292817c386
You have a bundler environment variable for silence_root_warning set to "1". This will take precedence over the global value you are setting
 ---> 586ed297429e
Removing intermediate container af292817c386
Successfully built 586ed297429e
```

## OK NOW WHAT?

## DOCKER-COMPOSE.YML

```
version: "2"
services:
  web:
    build: .
    volumes:
      - .:/apps/test
    working_dir: /apps/test
    command: bundle exec puma -b 0.0.0.0
    expose:
      - 3000
    ports:
      - 3000:3000
```


Mysql2::Error

Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2)

Extracted source (around line **#89**):

```
      socket = socket.to_s unless socket.nil?

      connect user, pass, host, port, database, socket, flags
    end

    def parse_ssl_mode(mode)
```

```
Rails.root: /apps/test
```

Application Trace | Framework Trace | Full Trace

```
mysql2 (0.4.5) lib/mysql2/client.rb:89:in `connect'
mysql2 (0.4.5) lib/mysql2/client.rb:89:in `initialize'
activerecord (5.0.1) lib/active_record/connection_adapters/mysql2_adapter.rb:25:in `new'
activerecord (5.0.1) lib/active_record/connection_adapters/mysql2_adapter.rb:25:in `mysql2_connection'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:729:in `new_connection'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:773:in `checkout_new_connection'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:752:in `try_to_checkout_new_connection'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:713:in `acquire_connection'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:490:in `checkout'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:364:in `connection'
activerecord (5.0.1) lib/active_record/connection_adapters/abstract/connection_pool.rb:883:in `retrieve_connection'
activerecord (5.0.1) lib/active_record/connection_handling.rb:128:in `retrieve_connection'
activerecord (5.0.1) lib/active_record/connection_handling.rb:91:in `connection'
```

## LETS ADD SOME ADDITIONAL SERVICES

```
version: "2"
services:
  web:
    build: .
    volumes:
      - .:/apps/test
    working_dir: /apps/test
    command: bundle exec puma -b 0.0.0.0
    ports:
      - 3000:3000
    depends_on:
      - db
  db:
    image: mysql:5.7
    ports:
      - 3306:3306
    environment:
      MYSQL_ALLOW_EMPTY_PASSWORD: 'yes'
```

## NETWORKING

```
⚡ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS                     NAMES
c35e7666439e        test_web            "bundle exec puma ..."   About a minute ago   Up 2 seconds        0.0.0.0:3000->3000/tcp    test_web_1

2bf299b1d180        mysql:5.7           "docker-entrypoint..."   About a minute ago   Up 2 seconds        0.0.0.0:32769->3306/tcp   test_db_1
```

## CHANGES TO DATABASE.YML

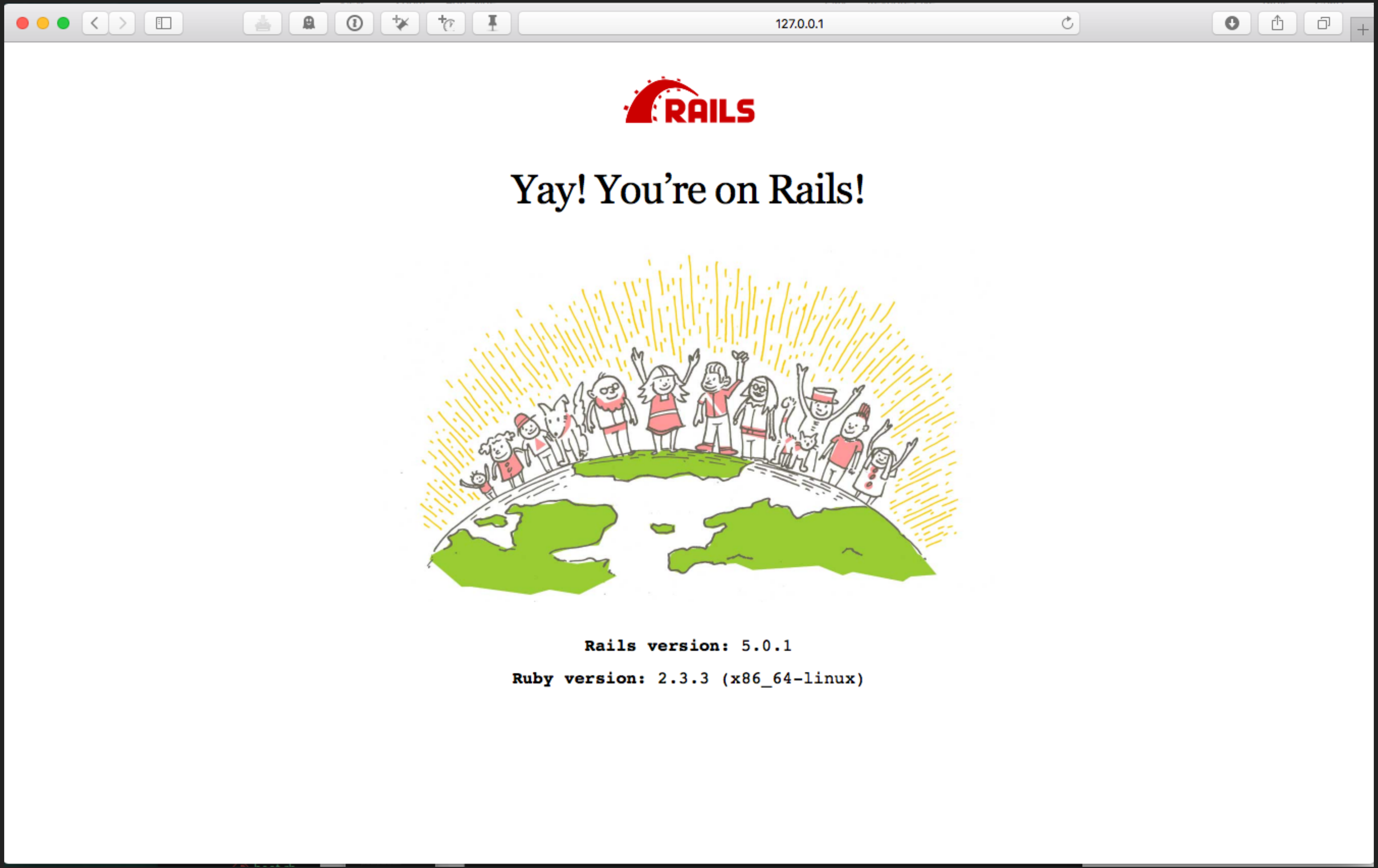
```
default: &default
  adapter: mysql2
  encoding: utf8
  pool: 5
  username: root
  password:
  host: db

⚡ docker-compose run web rails db:create
Created database 'Test_development'
Created database 'Test_test'
```

## CHANGES TO DATABASE.YML

```
default: &default
  adapter: mysql2
  encoding: utf8
  pool: 5
  username: root
  password:
  host: db

⚡ docker-compose run web rails db:create
Created database 'Test_development'
Created database 'Test_test'
```



Yay! You're on Rails!

```
Rails version: 5.0.1
Ruby version: 2.3.3 (x86_64-linux)
```

## NOW THAT IT IS RUNNING, WHAT ARE SOME USEFUL COMMANDS

- `docker-compose build`
- `docker-compose rm`
- `docker-compose (up | down)`
- `docker-compose (start | stop)`
- `docker-compose run`
- `docker-compose exec`
- `docker-compose logs (-f)`

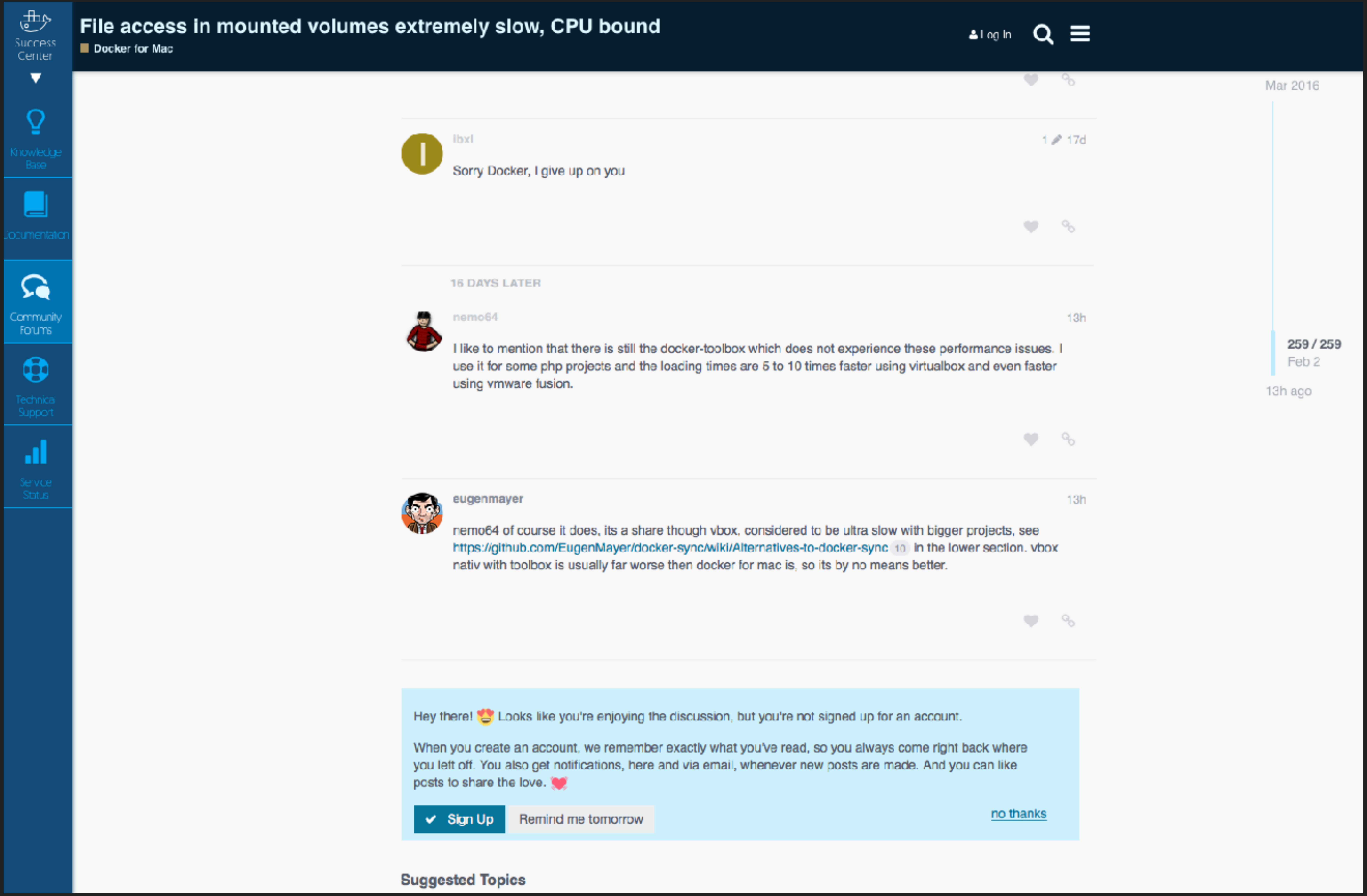
## TIPS FROM REAL WORLD USAGE

- Make your base image specific to your app not its environment.
- Use environment variables to set up environment specific aspects.
- Make individual containers disposable.
- Utilize community containers, but feel comfortable creating your own when you move to production.
- If your docker-compose.yml is getting really complicated use the extends command to break it up.

## TIPS FROM REAL WORLD USAGE CONT…

- If you are using Docker for Mac be prepared for the slow performance of OSXFS if you have a lot of small files (RubyGems, NPM)

## HOW ANGRY? . . .😡



Success Center

Knowledge Base

Documentation

Community Forums

Technical Support

Service Status

File access in mounted volumes extremely slow, CPU bound

Docker for Mac

Log In

Mar 2016

lbxl 1 17d

Sorry Docker, I give up on you

16 DAYS LATER

nemo64 13h

I like to mention that there is still the docker-toolbox which does not experience these performance issues. I use it for some php projects and the loading times are 5 to 10 times faster using virtualbox and even faster using vmware fusion.

eugenmayer 13h

nemo64 of course it does, its a share though vbox, considered to be ultra slow with bigger projects, see https://github.com/EugenMayer/docker-sync/wiki/Alternatives-to-docker-sync 10 in the lower section. vbox nativ with toolbox is usually far worse then docker for mac is, so its by no means better.

259 / 259
Feb 2

13h ago

Hey there! 😍 Looks like you're enjoying the discussion, but you're not signed up for an account.

When you create an account, we remember exactly what you've read, so you always come right back where you left off. You also get notifications, here and via email, whenever new posts are made. And you can like posts to share the love. 💗

Sign Up | Remind me tomorrow | no thanks

Suggested Topics

## THERE ARE SOLUTIONS THOUGH

- Use NFS available with alternate docker-machines such as dinghy (codekitchen / **dinghy**) https://github.com/codekitchen/dinghy
- Use the local driver for volumes that can live on a container

```
volumes:
  mysql-data:
    driver: local
```

```yaml
version: "2"
services:
  web:
    build: .
    volumes:
      - .:/apps/test
    working_dir: /apps/test
    command: bundle exec puma -b 0.0.0.0
    ports:
      - 3000:3000
    depends_on:
      - db
  db:
    image: mysql:5.7
    ports:
      - 3306
    environment:
      MYSQL_ALLOW_EMPTY_PASSWORD: 'yes'
    volumes:
      - mysql-data:/var/lib/mysql

volumes:
  mysql-data:
    driver: local
```

```yaml
version: "2"
services:
  web:
    build: .
    volumes:
      - .:/apps/test
    working_dir: /apps/test
    command: bundle exec puma -b 0.0.0.0
    ports:
      - 3000:3000
    depends_on:
      - db
  db:
    image: mysql:5.7
    ports:
      - 3306
    environment:
      MYSQL_ALLOW_EMPTY_PASSWORD: 'yes'
    volumes:
      - mysql-data:/var/lib/mysql

volumes:
  mysql-data:
    driver: local
```

## THERE ARE SOLUTIONS THOUGH

- Use NFS available with alternate docker-machines such as dinghy (codekitchen / **dinghy**) https://github.com/codekitchen/dinghy
- Use the local driver for volumes that can live on a container

```
volumes:
  mysql-data:
    driver: local
```

- docker-sync http://docker-sync.io

## DOCKER-SYNC

```yaml
# docker-sync.yml
options:
  verbose: true
  compose-file-path: './base.yml'
  compose-dev-file-path: './docker-compose.yml'
  sync_user: '1000'
syncs:
  accounts-sync:
    src: '.'
    dest: '/apps/test'
    sync_strategy: 'unison'
    sync_excludes: ['.git', 'tmp', 'log']
    sync_excludes_type: 'Name'
```

THANK YOU.
QUESTIONS?