

Why use Docker and Compose in your CI?

Carla Suárez | @carlast22

Why use Docker and Compose in your CI?

Carla Suárez | @carlast22

Hi! / Oi!



Ecuatoriana

Journalist | Developer

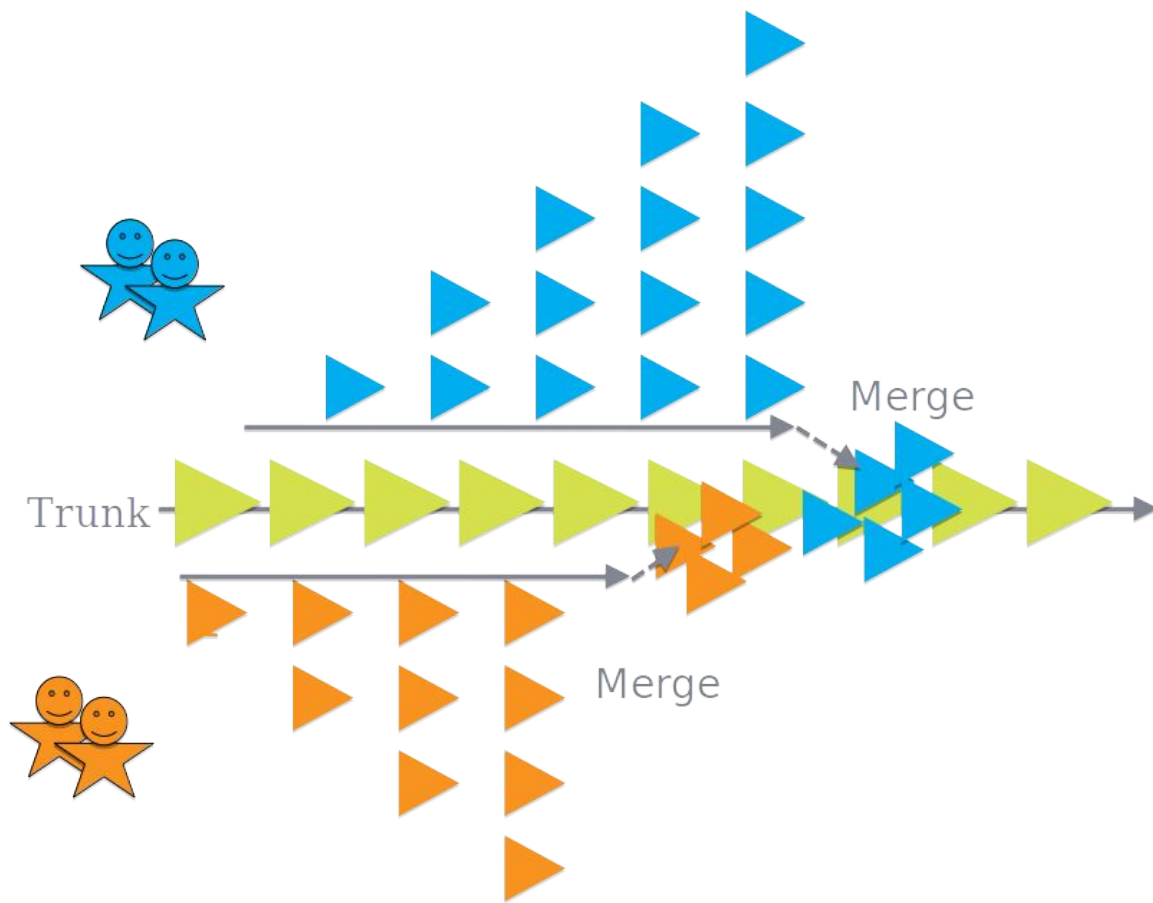
Python, Ruby, HTML, CSS

Learning about Devops

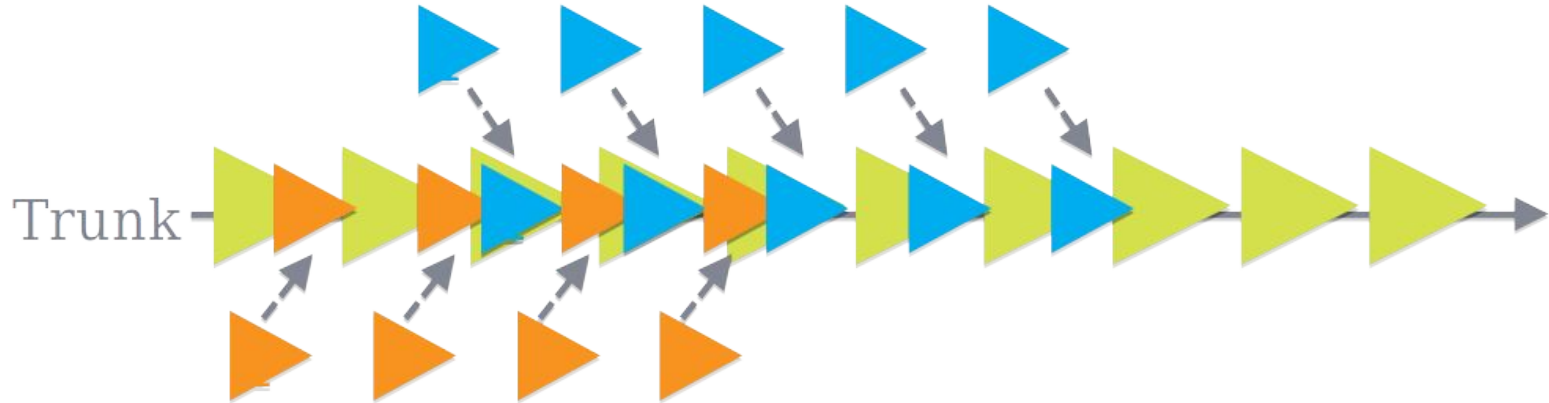
ThoughtWorks®

CI and CD?

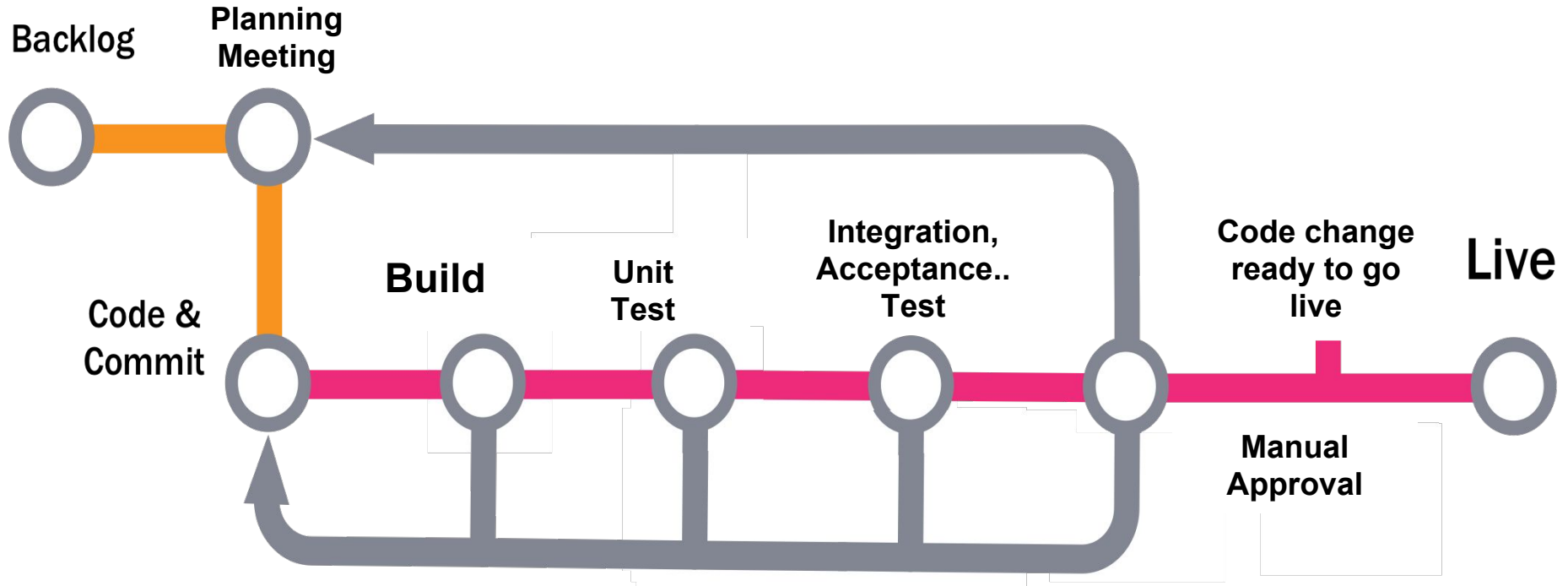
What is this? Concepts and importance in deployment



Trunk based development



Here is where CI go into scene



The practices for CI

- **Maintain a single source repository**
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- **Make your build self-testing**
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- **Every commit should build on an integration machine**
- Keep the build fast
- Test in a clone of the production environment
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- **Keep the build fast**
- Test in a clone of the production environment
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- **Test in a clone of the production environment**
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- **Everyone can see what is happening**
- Make it easy for anyone to get the latest executable
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Everyone can see what is happening
- **Make it easy for anyone to get the latest executable**
- Automate deployment

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Everyone can see what is happening
- Make it easy for anyone to get the latest executable
- **Automate deployment**

Continuous Deployment is closely related to *Continuous Integration* and refers to the release into production of software that passes the automated tests.

“Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time”

Martin Fowler

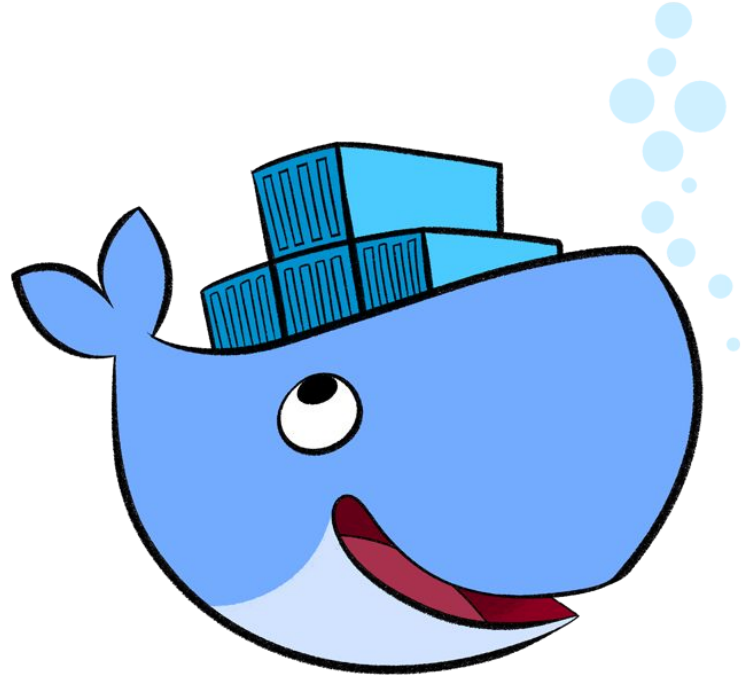
Main principles and practices

- Create a repeatable, reliable way to release software
- Everybody is responsible for the delivery process
- Automate almost everything

Docker and Compose

What are and how can it help you in CD?

Have you met
Docker yet?

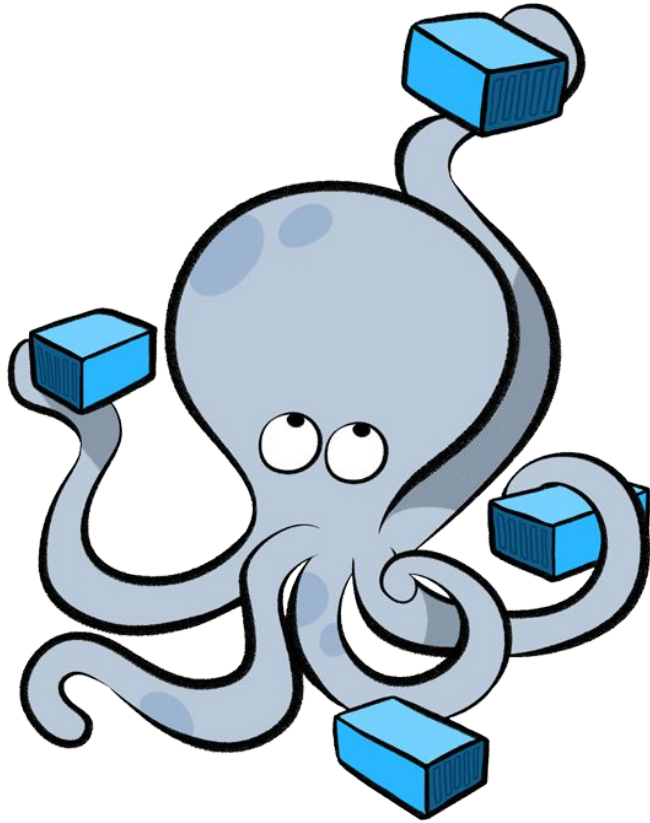


Dockerfile

```
1 FROM ruby:2.3-alpine
2
3 WORKDIR /app
4 COPY Gemfile /app
5 RUN bundle install
6 COPY . /app
7 EXPOSE 9292
8
9 CMD ["rackup", "--host", "0.0.0.0"]
10
```

Gemfile

```
1 source 'https://rubygems.org'
2
3 gem 'cuba', '~> 3.8'
4 gem 'tilt', '~> 2.0', '>= 2.0.5'
5 gem 'redis', '~>3.2'
```

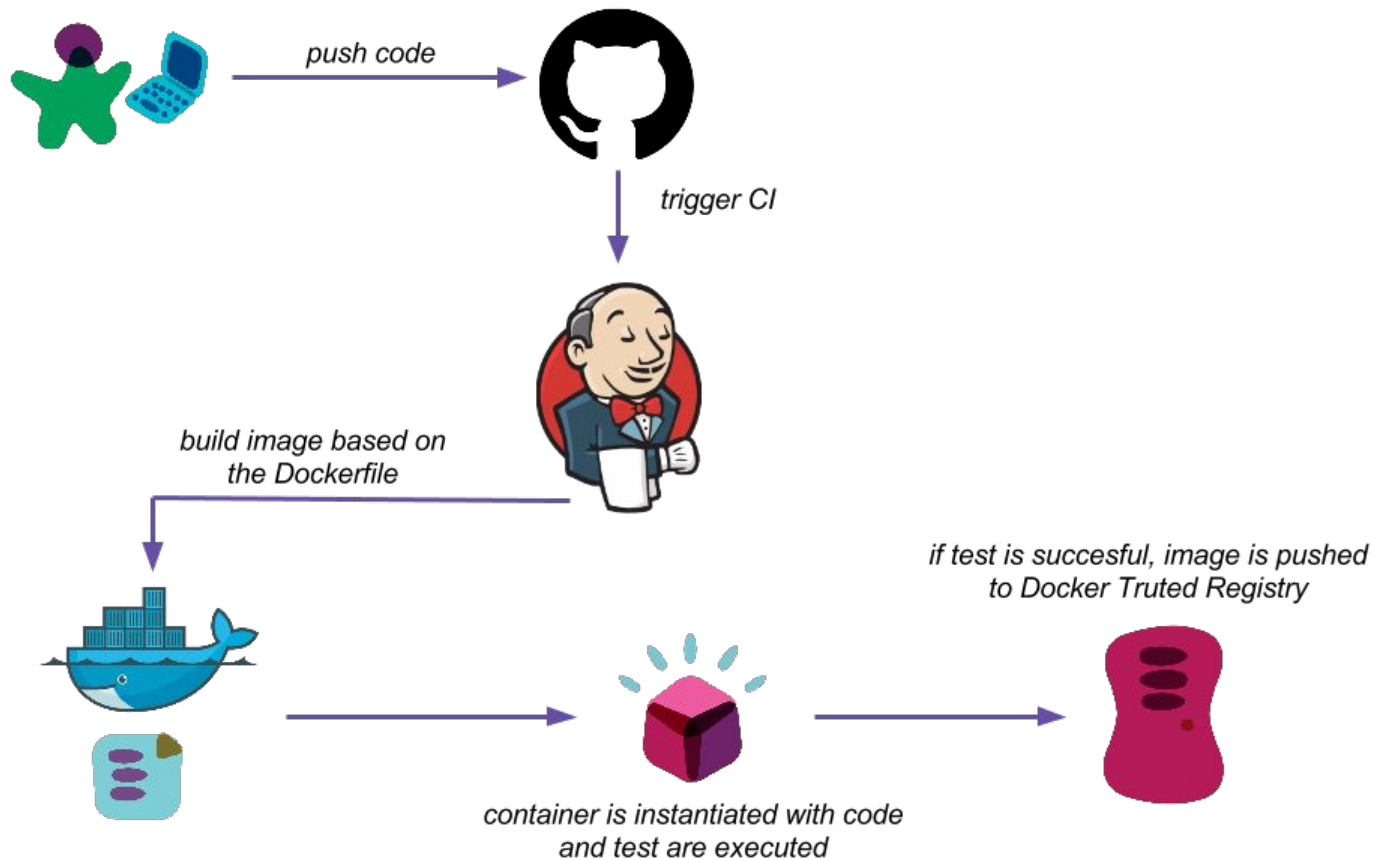
And Docker
Compose?

Docker's friend

docker-compose.yml

```
1 version: '2'
2
3 services:
4   app:
5     build: .
6     ports:
7       - "9292:9292"
8     links:
9       - redis
10  db:
11    image: redis
12
```

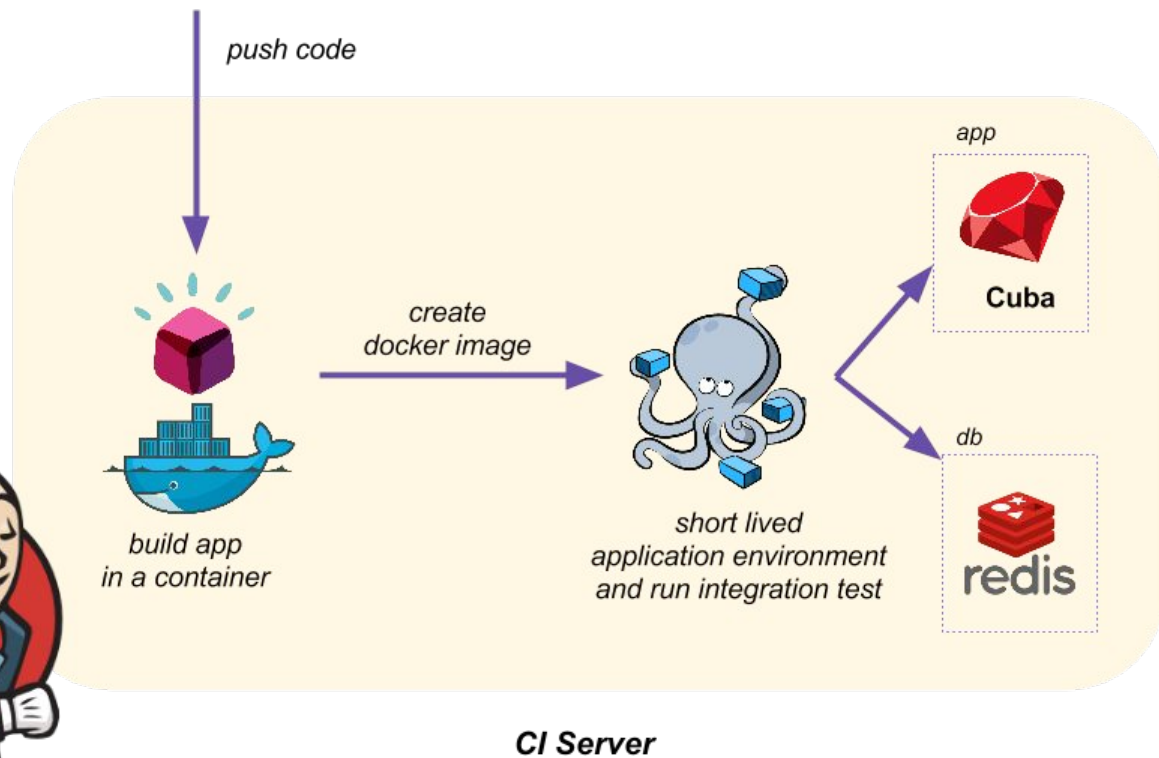
CI Workflow



What is
Registry?

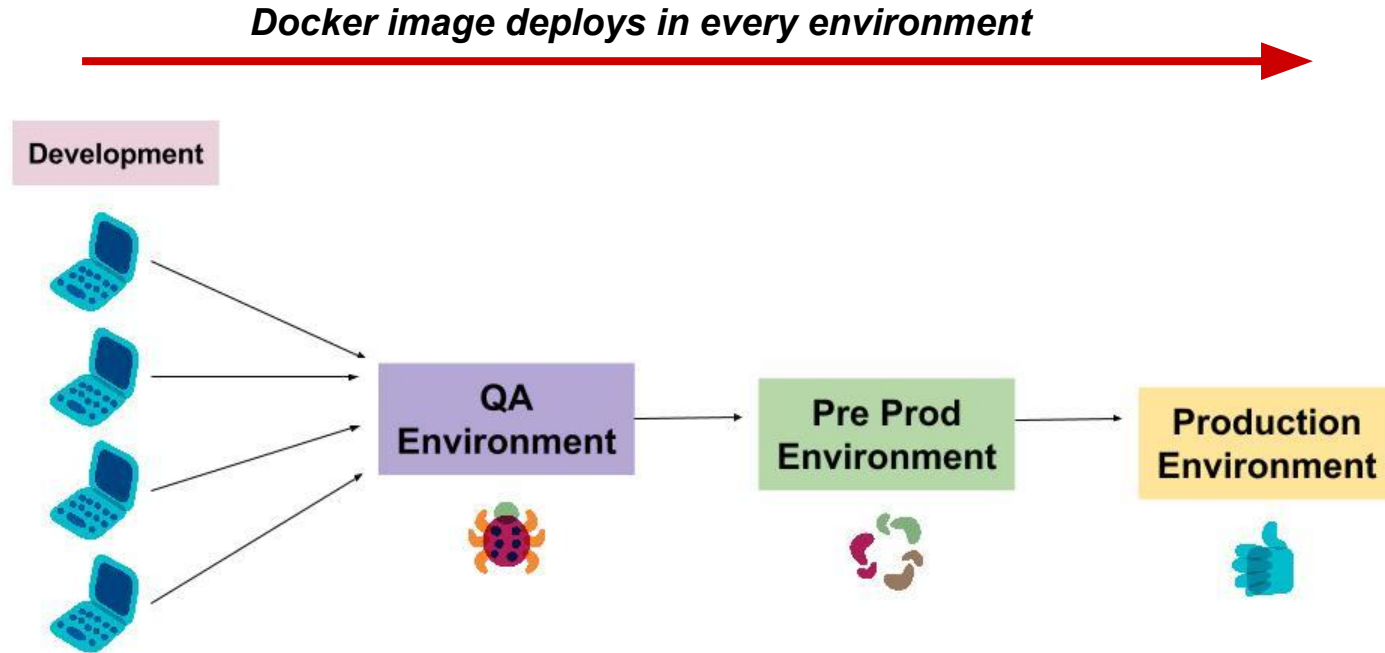


How about Docker Compose?



Using Docker and Compose

The same in different environments



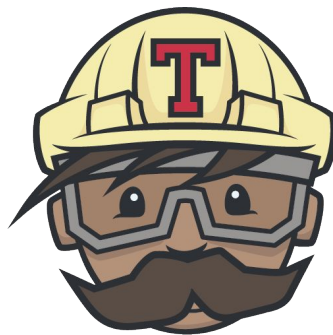
Deploy all your application needs

Automated testing

Dedicated container for each application

Reduce time and cost

Stability and resilience



VM's vs. Containers

- **VM** is a completely virtualized environment that only abstracts the physical hardware.
- **VM** comes with its own BIOS, virtualized network adapters, disk storage, CPU and a complete operating system.

- **Container** abstraction happens at the operating system level.
- Each **container** user shares the same operating system, kernel instance, network connection and base file system, each instance of the application will run within a separate user space.

Let's use Docker!

Let's use Docker!

Is it the best for your project requirements?

Spike before choosing

Do CD, if you are not doing it yet

Let's use Docker!

Is it the best for your project requirements?

Spike before choosing

Do CD, if you are not doing it yet

Let's use Docker!

Is it the best for your project requirements?

Spike before choosing

Do CD, if you are not doing it yet

Obrigada!

csuarez@thoughtworks.com

[@carlast22](#)

