The rise of Docker, and the future of computing

Arnaud Porterie - @icecrime - 2015-09-29 - HPTS

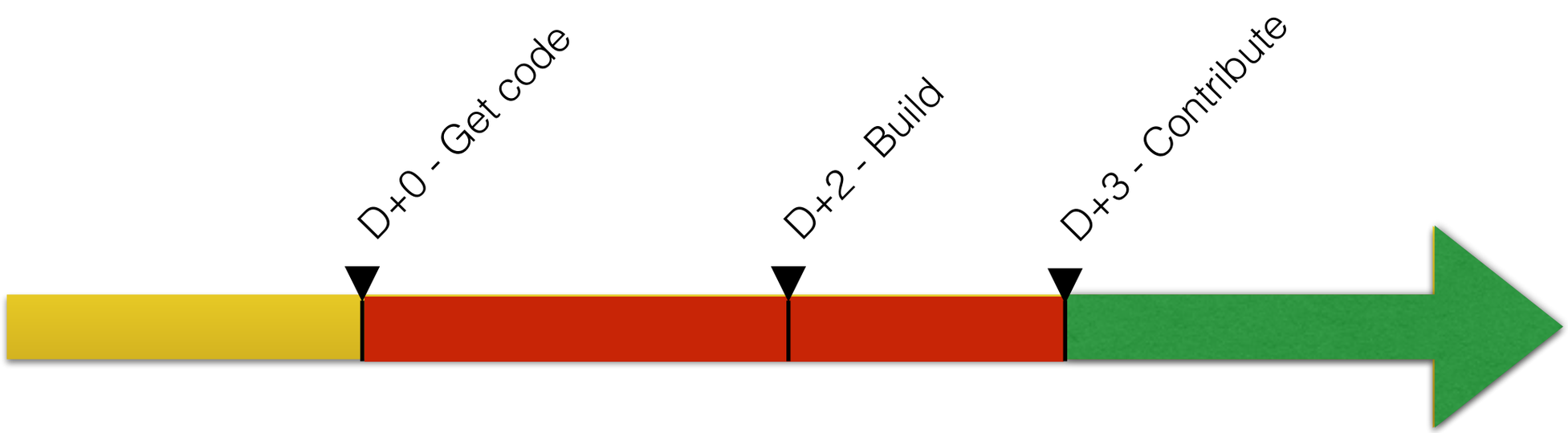
Who am I?

- Arnaud Porterie, @icecrime on  and 
- Core maintainer of the Docker open source project
- Engineering manager at Docker

From dev to prod, what is now different with Docker?

Developer onboarding

- Development dependencies described in a wiki page
- ~2 days process (when things go right)
- Team must keep the document up to date



Developer onboarding

- Docker value: **the builder**
- Development environment as code, with the `Dockerfile` as the source of truth
- **Universal**: get code, `docker build`, `docker run`
- *Docker own dev environment is in Docker*

Developer onboarding

14 lines (10 sloc) | 272 Bytes

```
FROM golang:1.4.2
MAINTAINER Arnaud Porterie <icecrime@docker.com>

# Install GB dependency manager
RUN go get github.com/constabulary/gb/...

# Build the project
ADD . /src
WORKDIR /src
RUN gb build all

# Set the entrypoint
ENTRYPOINT ["/src/bin/vossibility-collector"]
```

Continuous integration

- Traditionally: specific hosts test a specific app
- Managed by the individual dev teams
- “Works on my machine” syndrome

Continuous integration

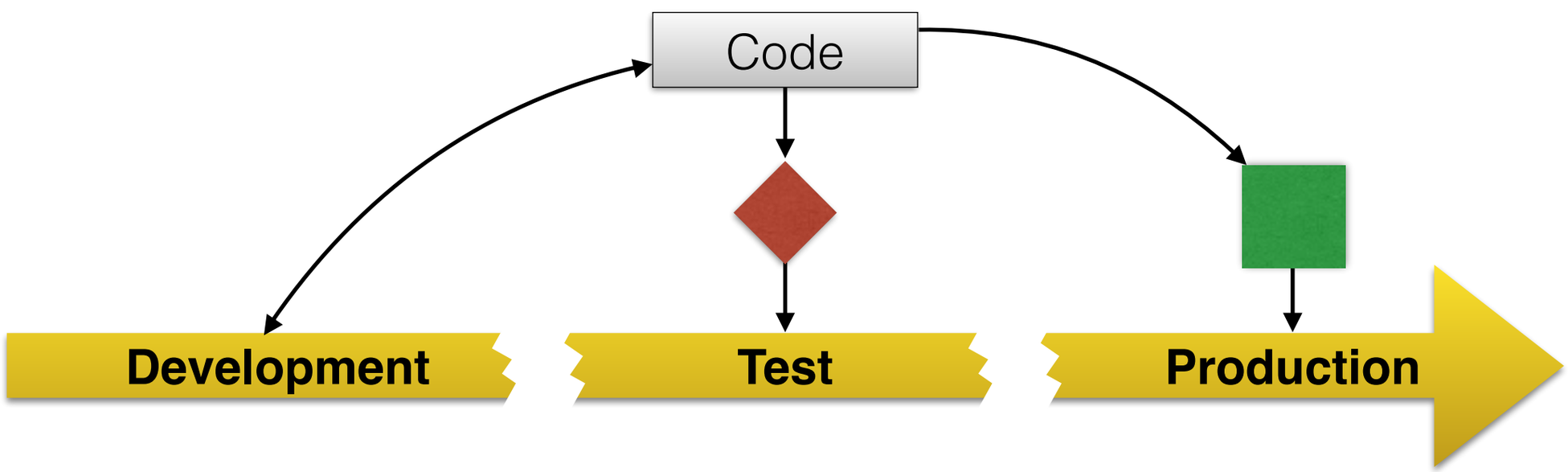
- Docker value: **repeatability through isolation**
- Running containers don’t share anything by default
- Tests don’t interfere with each other, no traces on host

Continuous integration

- Docker value: **universality of operations**
- Standard way to build & run across languages & stacks
- *Docker own tests are run in a Docker container*

Deployment

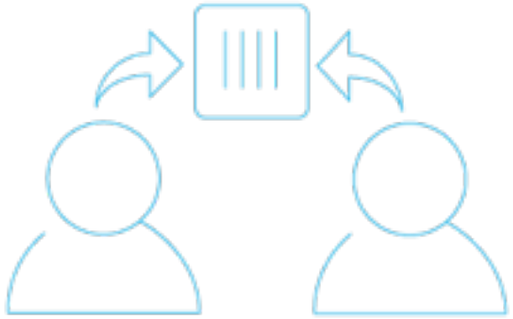
- Dependency management
- Specialized ops for a set of services



Deployment

- Docker value: **unified artifact**
- The Docker image is the artifact you **build** once, **test**, **ship** & **run** in production

Deployment



- Docker value: **collaboration**
- Docker Hub, on prem. Docker Registry
- Define deployment workflow, permissions

Summing it up

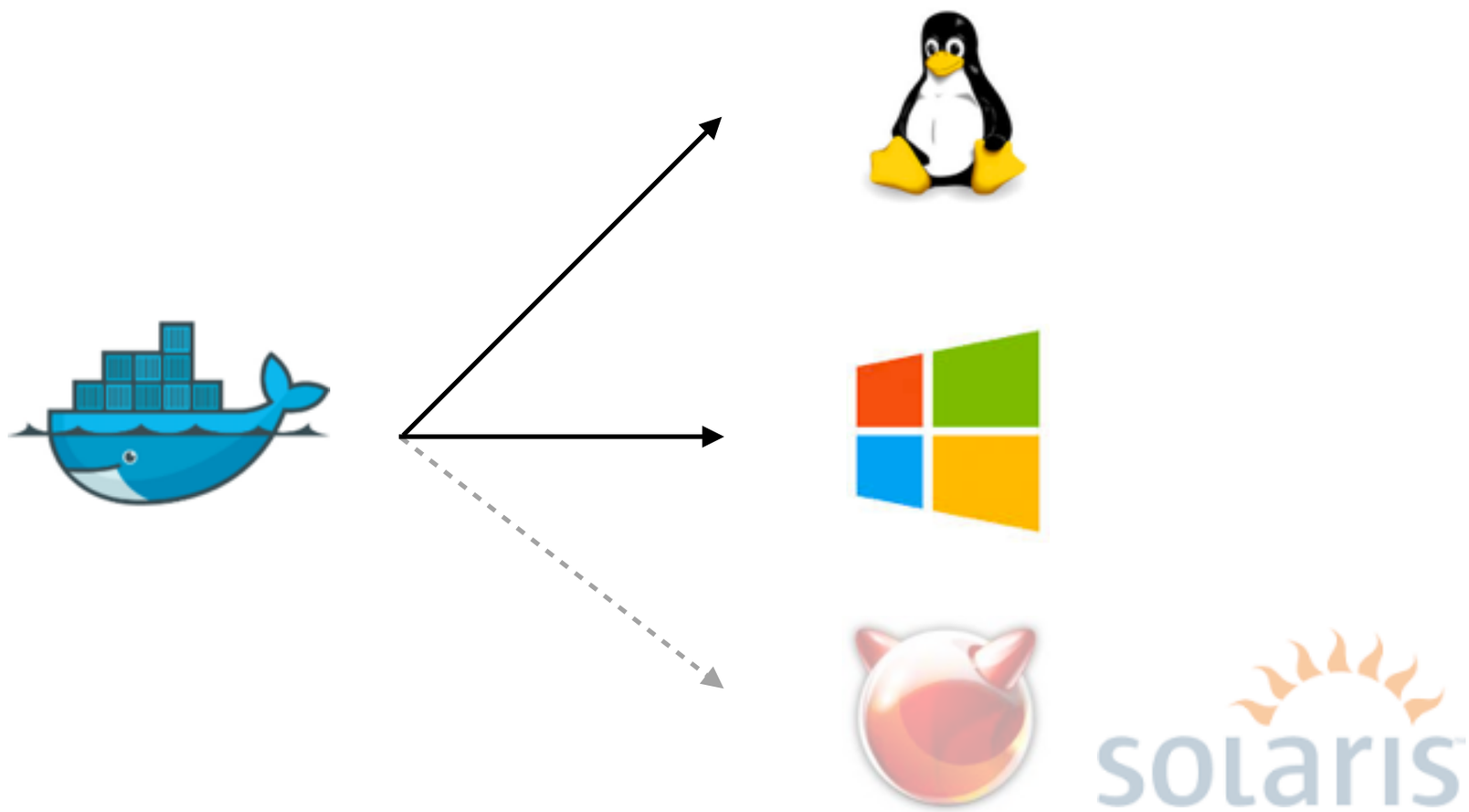
- Docker elegantly solves very common problems
- Makes modern best practices natural
  - Repeatability (e.g., host independence)
  - Immutability (the Docker image is the unit)
  - Single responsibility principle (microservices)

Docker is established as an essential building block for distributed applications

What’s next?

Operating system

- Docker abstracts underlying OS differences
- Defines a unified API for processes management



Operating system

- Race for the “best OS for running containers”

RANCHEROS

Nano Server

Linux kernel

- Linux containers have existed for ~ a decade, but never had as much attention than in the past 2 years
- Downside: aggressive kernel version requirements can contribute to slowing down adoption

Container runtime

- The Open Container Initiative
- Driven by a group of technical experts (Docker, Google, RedHat, CoreOS, IBM, …)

https://www.opencontainers.org

THE LINUX FOUNDATION

User experience

- Docker succeeded at the frontier of dev & ops
- Containers are now in everybody's hands
- Can one size fit all?

Dev

Ops

Docker

Early adopters

Skeptics

Want Docker end-to-end

Evaluating for production

Well established

Rapidly growing


experience
fast
iteration
one-click deploy
magic
share
in code
one-click deploy
one-click deploy
fast
IDE
fast
experience
share
experience
in code
IDE
experience
productivity
sync code
experience
sync code
productivity
sync code
integration
integration
fast
productivity
iteration
iteration
integration
build
magic
productivity
productivity
feedback
IDE
magic
fast
integration
build
in code
share
magic
one-click deploy


magic
feedback
feedback
integration
one-click deploy
IDE
productivity
share
one-click deploy
in code
feedback
experience
magic
sync code
build
experience
productivity
magic
in code
magic
sync code
sync code
experience
sync code
build
integration


sync code
fast
feedback
magic
iteration
feedback
feedback
share
iteration
in code
in code
experience
integration
fast
build
integration
in code
build
share
iteration
share
IDE
build
one-click deploy
productivity
one-click deploy
build
IDE
iteration
productivity
share
one-click deploy
IDE
IDE
feedback
build
iteration
fast
fast
IDE
integration
sync code
in code
build

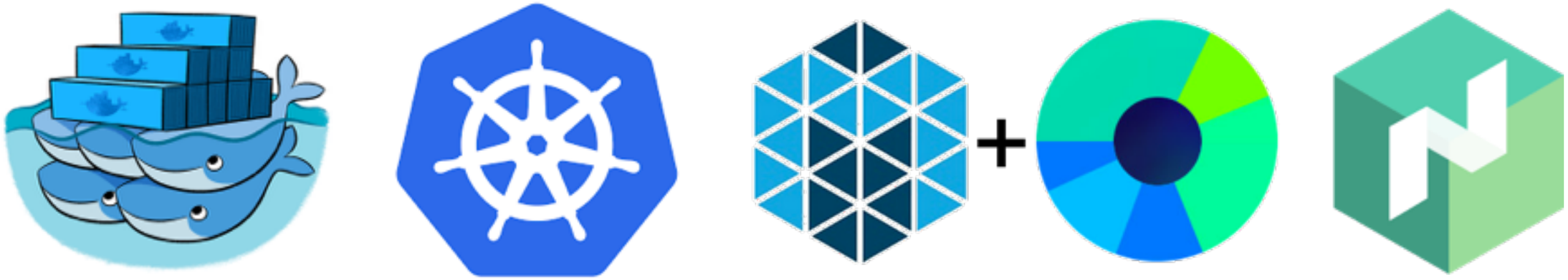

networking
stability
performance
plugins
control
control
storage
networking
monitoring
lightweight
security
modularity
monitoring
lightweight
stability
orchestration
plugins
auditability
auditability
lightweight
security
customization
orchestration
control
lightweight
UNIX principles
modularity
stability
customization
security
customization
orchestration
customization
control
security
storage
no magic
customization
plugins
performance
orchestration


UNIX principles
auditability
storage
storage
networking
stability
plugins
no magic
stability
customization
performance
performance
customization
orchestration
UNIX principles
lightweight
monitoring
performance
monitoring
modularity
plugins
monitoring
performance
lightweight
no magic
auditability
no magic
stability
control
control
auditability
control
storage
UNIX principles
UNIX principles
security
orchestration
plugins
auditability
stability
no magic


security
lightweight
no magic
auditability
auditability
performance
orchestration
UNIX principles
monitoring
plugins
modularity
no magic
networking
storage
no magic
security
UNIX principles
networking
lightweight
stability
UNIX principles
plugins
security
storage
performance
networking
no magic
networking
customization
monitoring
modularity
security
stability
networking
orchestration
storage
lightweight
UNIX principles
modularity
modularity
control
control
performance
customization
storage
modularity
monitoring
modularity
orchestration

Orchestration

- Currently the biggest area of innovation



- Challenges: networking, storage, service discovery, monitoring, log aggregation, …

Orchestration

Will a standard for orchestration emerge?

Thank you!
Questions?

arnaud@docker.com