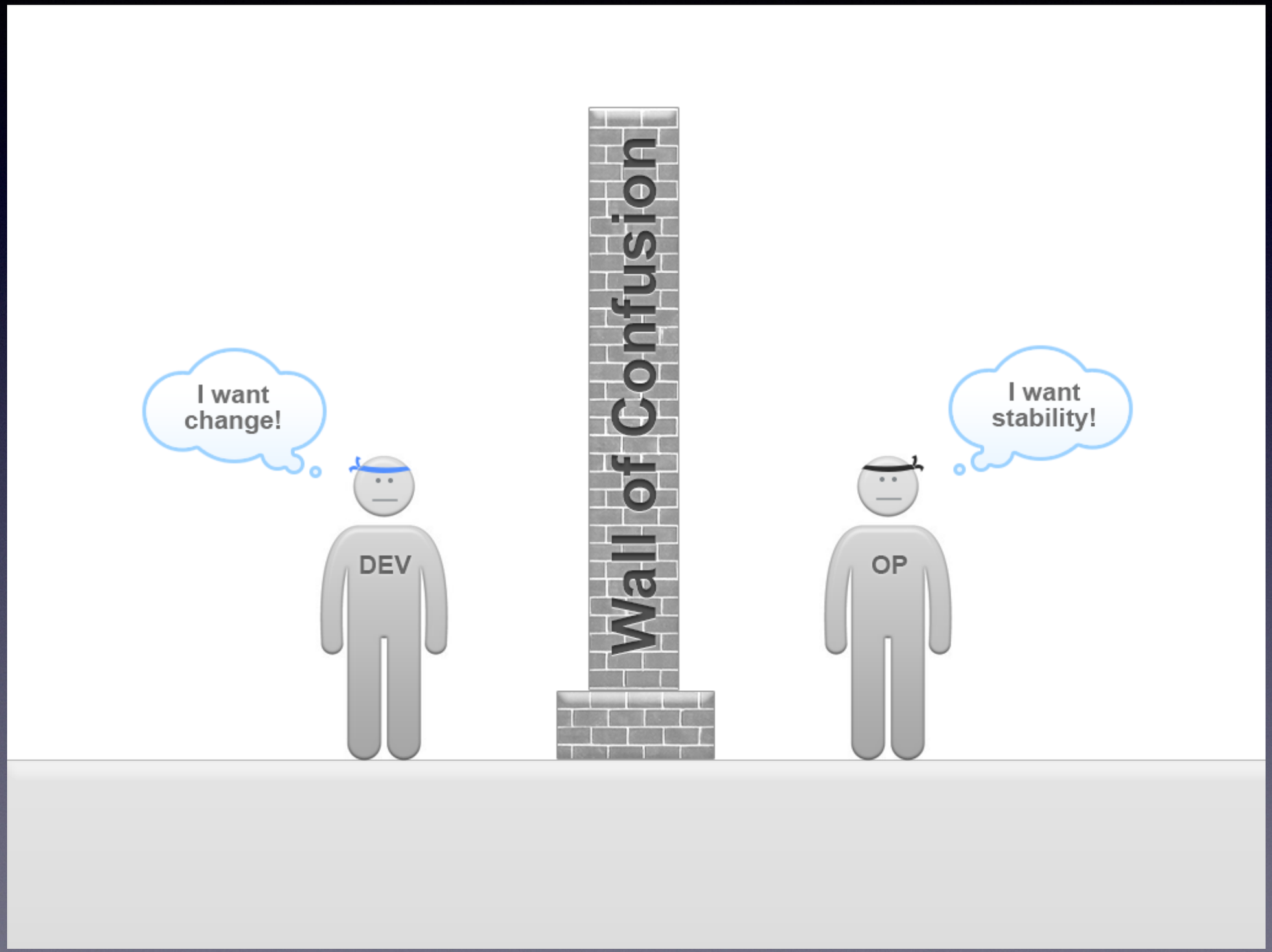
Thoughts about docker

Russell Clare
DevOps Engineer

A discussion about a tool



Docker around Production

- Key Challenges:
  - 1/ Logging
  - 2/ Scheduling
  - 3/ Security
- Key Benefits:
  - 1/ Vender neutrality
  - 2/ Deployments
  - 3/ Confidence

Dockerizing all the things!

- High Yield benefits:
  - Application Servers
  - Proxy Servers
  - Log collectors
  - Load balancers

Embracing Agile - Docker ?

The Continues Delivery Maturity Model

Key Metrics Evaluated:

- Culture & Organisation
- Design & Architecture
- Build & Deploy
- Test & Verification
- Information & Reporting

Levels of Aptitude:

- Base
- Beginner
- Intermediate
- Advanced
- Expert

Embracing Agile - Docker ?

The Continuous Delivery Maturity Model

	Base	Beginner	Intermediate	Advanced	Expert
Culture & Organization	• Prioritized work • Defined and documented process • Frequent commits	• One backlog per team • Share the pain • Stable teams • Adopt basic Agile methods • Remove boundary dev & test	• Extended team collaboration • Component ownership • Act on metrics • Remove boundary dev & ops • Common process for all changes • Decentralize decissions	• Dedicated tools team • Team responsible all the way to prod • Deploy disconnected from Release • Continuous improvement (Kaizen)	• Cross functional teams • No rollbacks (always roll forward)
Design & Architecture	• Consolidated platform & technology	• Organize system into modules • API management • Library management • Version control DB changes	• No (or minimal) branching • Branch by abstraction • Configuration as code • Feature hiding • Making components out of modules	• Full component based architecture • Push business metrics	• Infrastructure as code
Build & Deploy	• Versioned code base • Scripted builds • Basic scheduled builds (CI) • Dedicated build server • Documented manual deploy • Some deployment scripts exsists	• Polling builds • Builds are stored • Manual tag & versioning • First step towards standardized deploys	• Auto triggered build (commit hooks) • Automated tag & versioning • Build once deploy anywhere • Automated bulk of DB changes • Basic pipeline with deploy to prod • Scripted config changes (e.g. app server) • Standard process for all environments	• Zero downtime deploys • Multiple build machines • Full automatic DB deploys	• Build bakery • Zero touch continuous deployments
Test & Verification	• Automatic unit tests • Separate test environment	• Automatic integration tests	• Automatic component tests (isolated) • Some automatic acceptance tests	• Full automatic acceptance tests • Automatic performance tests • Automatic security tests • Risk based manual testing	• Verify expected business value
Information & Reporting	• Baseline process metrics • Manual reporting	• Measure the process • Static code analysis • Scheduled quality reports	• Common information model • Traceability built into pipeline • Report history is available	• Graphing as a service • Dynamic test coverage analysis • Report trend analysis	• Dynamic graphing and dashboards • Cross silo analysis

Docker - Three use cases

Docker, as docker intended (dockerfiles)

Key Features:

- Easy to get going.
- Full Hierarchy of images.
- Free(ish) CI Services.
- Good community support.

‘Gotchas’:

- Images larger than optimal
- Changing build step invalidates chain.
- Update image (apt-get upgrade) difficult without code submission

Docker - Three use cases

Docker, as a mini vm (single logical service)

Key Features:

- Increased flexibility.
- Control over image hierarchy.
- Run under Private CI.
- Logging problems easy to solve.

‘Gotchas’:

- Need to manage init process (phusion/base-image)
- Easy to break Docker philosophy.
- Brings complexity
- Moving away from the ‘norm’

Docker - Three use cases

Docker, as tool for scheduled events ‘cron wrappers’

Key Features:

- Increasing levels of control.
- Same image for Application server & batch processing.
- Simplified deployment process.
- Logical extension of ‘mini vm’ style of doing things.

‘Gotchas’:

- Adding further complexity.
- Increasing reliance on docker images.
- Alerting for schedule tasks needs additional attention.

Docker - Three use cases

```
vars:
  - book:
      name: 'confluence'
      entrypoint: true
      command: '/docker/start.sh'
      port:
        - { dst: '8090', src: "{{ confluence_port }}" }
      maps:
        - { dst: '/docker', src: '/home/ubuntu/confluence' }
      home: '/home/ubuntu/confluence'

tasks:

- name: tag new image for production
  action: command
    docker tag
    {{ book.name }}:latest
    {{ book.name }}:run
  sudo: yes
  ignore_errors: yes
```

Docker - Three use cases

```ruby
#!/usr/bin/env ruby
require 'rubygems'
require 'time'
require 'docker'
require 'json'

__ROOT = '{{ book.home }}'
Docker.url = 'unix:///var/run/docker.sock'

#puts Docker.info
# Fetch the container information
container = Docker::Container.get('{{ book.name }}-production')

# exit if the container is currently running
if container.info['State']['Running'] == true
  puts 'Exiting.. container is already running...'
  exit 0
end

# Fetch the backoff information
backoff = File.read("#{__ROOT}/backoff")

# exit if a backoff condition has been set
if backoff.strip == "true"
  puts 'Exiting.. Backoff condition has been specified...'
  exit 0
end

# check exit code of container
if container.info['State']['ExitCode'] == 0
   container.delete(:force => false)
else
   puts 'failed - boo'
   output = container.attach(:stream => false, :stdin => nil, :stdout => true, :stderr => true, :logs => true, :tty => true)
   puts output
   filename = "#{Time.now.utc.iso8601}-cron-failure-{{ book.name }}-exit-#{container.info['State']['ExitCode']}-#{container.info['Config']['Hostname']}"
   file_out = File.open( "#{__ROOT}/logs/#{filename}","w" )
   file_out << "#{output}"
   file_out.close
   container.delete(:force => false)
end
```

Docker - Three use cases

```
#!/bin/bash

    docker run -t -d -i \
    --name {{ book.name }}-production   \
{% for port in book.port %}
    -p {{ port.src }}:{{ port.dst}} \
{% endfor %}
{% for map in book.maps %}
    -v {{ map.src }}:{{ map.dst }} \
{% endfor %}
{% if book.entrypoint == true %}
    --entrypoint {{ book.command }} \
{% endif %}
    silverfloat-{{ book.name }}:run
```

Docker - Three use cases

```
#!/bin/bash
set -ev

__BACKOFF=/docker/backoff

# put server.xml into the correct place
ln -sf /docker/server.xml /opt/atlassian/confluence/conf/server.xml

# setup path
export PATH=/usr/sbin:/usr/bin:/sbin:/bin

# start up confluence
/opt/atlassian/confluence/bin/start-confluence.sh

# warm up cache
curl http://localhost:8090/confluence || true

# run the job and capture the output
while true
        do
             echo 'running' > /docker/status
             _BACKOFF=$(cat ${__BACKOFF})
             if [ ${_BACKOFF} = 'true' ]
             then
             /opt/atlassian/confluence/bin/stop-confluence.sh
             CONFLUENCE_EXIT=$?
               echo 'backedoff' > /docker/status
               exit ${CONFLUENCE_EXIT}
             fi
             echo 'sleeping' > /docker/status
             sleep 30
    done
```

Obligatory ‘we are recruiting’

- Global Leader in facilitating Intelligent Delivery. (big digital franking machine)
- Used by ~ 80% of the top online retailers in the UK. (ASOS, John Lewis, M&S..)
- Shipping > 350 Million packages a year.
- Nice modern offices with natural light :) —>

200, Grays Inn Road, WC1

Questions ?

Sources:
http://www.infoq.com/articles/Continuous-Delivery-Maturity-Model
https://github.com/phusion/baseimage-docker