devops with docker

@beshippable

APACHE CON
NORTH AMERICA
avi cavale
co-founder & ceo
@avinci || avi@shippable.com

shippable
continuous app delivery pipelines
the most frequently asked question during an outage
the most frequently asked question during an outage

what changed?
the immediate next question is....
the immediate next question is....

who made the change?
both questions are bad for your business
both questions are bad for your business

slow down
both questions are bad for your business

slowing down

finger pointing

courtesy: dilbert comics
change is the root cause of almost all outages, but...
change is the root cause of almost all outages, but...

“business requires change”
“need to embrace change”
“need to embrace change”

devops: a CULTURE of embracing change
we embraced it tightly
traditional

change required

change complete

application delivery pipeline
traditional

development

application delivery pipeline
traditional

application delivery pipeline
traditional

change required

development

unit testing

functional testing

change complete

application delivery pipeline
traditional

application delivery pipeline
traditional

application delivery pipeline
devops culture embraced

application delivery pipeline
devops culture embraced

application delivery pipeline
devops culture embraced

1 org silo friction

15-20% improvement

change required
development
unit testing
functional testing
deployment
running
change complete

application delivery pipeline
devops culture embraced

1. org silo friction
   - change required

2. tool chain vertically optimized
   - application delivery pipeline

- development
- unit testing
- functional testing
- deployment
- running

change complete

15-20% improvement
devops culture embraced

1. org silo friction
2. tool chain vertically optimized
3. deployable image built last

change required

application delivery pipeline

development | unit testing | functional testing | deployment | running

change complete

15-20% improvement
20% better is not a REVOLUTION!

It's hard to fight a revolution on an empty stomach.

courtesy : V for Vendetta
why not build deployable image first?
devops is culture, so to make it a
devops is culture, so to make it a

F - frictionless
A - agile
C - continuous
T - transparent

application delivery pipelines
if you want to do just 1 thing this year

Transparency

every action

• starts in a chat room

• tracked as a github issue

• ends in a chat room
if you want to do just 1 thing this year

Transparency

every action

• starts in a chat room

• tracked as a github issue

• ends in a chat room

business req

#27 Post CI Docker build
Executive Summary
We released Docker build as a pre-CI feature, i.e. customers have a Dockerfile in their repo, we build the image, layer the code, test it, and ship the image to Docker Hub.
However, several customers have asked us for a different workflow. They want to do CI on a container that contains their test tools and tests, but after the CI passes, they want to run Docker build on their actual prod container that is super lightweight and does not contain tools/tests that they do not

support

#1294 Building branches on fork not using correct organization
We have an organization with multiple private repos that we build in Shippable
A developer has forked the private repo and is working on branch
Shippable has started building the private repo on commits to the branch in the developers context instead of organization.

prod errors

https://github.com/.../issues/2692
error: _getProjectPermission No projectPermission found.
2015-03-19T23:32:28.233Z - error: Github returned status 404 for request /user/emails using token token 7e63f4851c3635b88fd4e9901d59757e4d9430bf
2015-03-19T23:32:38.993Z - error: _getProjectPermission No projectPermission found.
Continuous

- continuous integration for every pull-request
- 1 step docker build image for every merge to `master`
- 1 step auto deploy & continuous functional testing
- deployment to prod is just changing an ENV var
Agile

- no SSH!
  - bug fixes, new changes, ops scripts all go the same way

- always deployable `master`
  - no feature branching

- feature flags
  - in-code branching

- small frequent changes
  - beta program
Frictionless

app delivery pipelines

container fab

dev
- r12 api
- r21 fe

beta
- r11 api
- r20 fe

prod
- r10 api
- r20 fe
noticeable FACTs

ship images 40-60 times a day

15 devs in 5 timezones

2hrs mean time to resolve bugs
where are we going from here?

micro-services baby!

- throughput creates merge hell
- rapid changes on 1 repo creates images chaos
25,000 build containers/week

45,000 devs on Shippable

signup for our beta program

email me avi@shippable.com
thank you very much

questions?

@beshippable