Choose your own Adventure
Finding the Right Path to Containerization
How BS is all this containerization stuff?

A guide to overhauling major parts of your system while you are overloaded, under-resourced, and (probably) burnt out.
About Me

Erica von Buelow
Software Engineer, CoreOS

@evonbuelow
evb@coreos.com
https://coreos.com
My Containerization Journey
Clever
2013-2016
Mohit
mohitinspace.com
Rafael Garcia
Erica von Buelow starting to outline things in this task.
9:14AM on October 15

Rafael Garcia added to Dev Ops
9:11AM on October 15

Docker

Description
Motivation: our infrastructure is quickly becoming more micro in the size of individual components. Whole machines running these services is overkill and also cumbersome to deploy. Docker allows us to run many isolated services on one machine via linux containers.

Subtasks
- Easy way to launch docker vm in dev environment
- Easy way to launch docker instance for prod

Projects
- Dev Ops
- experiments

Followers
- Erica von Buelow
- Rafael Garcia
All day, everyday
My motivations for this talk
So much advice...
Vendor

“Download our whitepaper!”
Expert

“You should be doing X. That's how I do it for my personal project and it works fine.”
Peer

“It depends.”
– How I feel about getting that kind of advice
It's cool that your home automation system runs on Kubernetes but I don't think that's gonna help me right now.
Based on my opinionated, biased, ignorant point of view,

My goal is for you to come away with:

1. A decision making framework for those “it depends” questions
2. A heads up on some common issues to look out for and avoid
3. Confidence that you’re gonna do just fine
4. Convince you to move to NYC and join my team at CoreOS
The Heads Ups
The Novice Mistakes

1. Network bottleneck
2. Disks filling up
3. GC'ing the evidence
4. Under-provisioning
5. Infra cost explosion
Network Bottleneck

The Problem:

• Container images are big! And they can get REALLY big.

• VPN gateway? Single router on top-level domain? Hitting rate limits on your registry’s IP?

• And more services deployed more often means even more images pushed and pulled around
Network Bottleneck

Solutions:

- Slim images - get in good habits now
- Stagger deploys, exponential backoff on deploy failures
- Separate networks for dev/prod/other business functions
Disks filling up

The Problem:

• dangling containers
• log spam
• volumes not GC'd
Disks filling up

Solutions:

• Alert on disk space! Set the alerts up now.
• Cron job to clean up after docker (this is only half a joke)
• Reduce application’s reliance on disk storage
GC’ing the Evidence

The Problem:

• Container fails, or application has a failure and exits.

• Container gets removed.

• Information needed to debug the failure got removed with it.
GC’ing the Evidence

Solutions:

• Make sure your logging and alerting systems are solid

• Put as much context as possible into the error messages and events emitted

• Store state at every stage of computation in remote durable storage for auditing

• Distinguish between known and unknown failures, then don't GC right away on the unknown
Under-provisioning

The Problem:

• Utilization curve not linear
• Unknown unknowns
• Application variability constrained for scheduling/system variability
Under-provisioning

After 70/80%, you run out of time to scale up.
Under-provisioning

Solutions:

• Never push past 70% utilization

• Don't be over-eager and tighten resources immediately

• Artificial load

• Autoscaling (somewhat effective, less than you think)
Infra cost explosion

The Problem:

• When it's easy to spin up a stack, developers will!

• and then forget about that one cluster or app or experiment

• These resources add up quickly
Infra cost explosion

Solutions:

• Associate resource costs with owner, then push cost responsibility down to them

• via shaming or literally charging them

• Aggressively garbage collect to get developers in the habit of being responsible early
The Constant Challenges

• End of life application lifecycle
• Build system
• Local development cycle
• Metrics and Logs
• Networking
• Configuration and secret management
Shifting the way things are done in an organization requires leadership.
Leadership-y Things

- Motivate
- Teach
- Communicate
1. Get your real goals down
Some real goals behind containerization efforts:

1. Reduce operational burden
2. Move faster
3. Reduce risk and fragility
4. Attract talent
Reduce Operational Burden

What they say:
• “Move to the cloud”
• “Get off of (operationally expensive tech here)”
• “Autoscaling”, “self-healing”

What they mean:
• Get costs under control
• Stop burning out operations people
• Clear the operations bottleneck for development
• Concentrate on core competencies, not infrastructure
Move Faster

What they say:

• “Microservices”
• “Continuous Integration”
• “Continuous Deployment”
• “Agile”

What they mean:

• Shorter release cycles
• Tighter feedback loops between development and production
• More efficient development workflows
• Easier prototyping and user testing
Reduce risk and fragility

What they say:

• “Observability”
• “Rollbacks”, “Canaries”
• “Autoscaling”, “self-healing”
• “Blue/Green Deploys”

What they mean:

• Legacy system error prone and not scaling to current needs
• Decouple components
• Automate to reduce human interaction points
• Consistent, repeatable structures
Attract talent

What they say:

• “Docker”
• “Kubernetes”
• “Unlimited Vacations”

What they mean:

• We’re worried we’re not cool enough to hire young, hip developers
• Maybe with new tech to talk about, our Medium posts will make it on HN
Or: reduce costs, reduce risk, reduce time to value
Make goals explicit

- Write them down
- Order them by importance
- Get sign off from the top
- Publish and share
2. Accept you're going to make things worse before they get better
• Going to be painful; there will be outages

• You can't predict all the ways the new technology will interact with the systems

• There's a lot you don't know you don't know
• You're probably underestimating the risks
• Your preparations are probably insufficient
• Quickly reacting and adjusting as these come up more important than whatever testing or pre-planning you can do

• Humility is key
3. Communicate and over-communicate
• Track progress
• Celebrate success
• Set the tone - don't sh*t on the old or new technology
• UX matters
Super helpful emails

That's great, but what does this mean? What should I do?

This means discovery is easier than ever. Because dependency env-var are auto-generated, no pillar changes are required to use discovery-go/node. Just use it:

```go
c := discovery.URL("stoked", "thrift")
```

Please don't hesitate to take advantage of this. We're shooting for 100% Discovery adoption by April 1st. Infra will do some of the heavy lifting, but we'd appreciate all the help we can get.

So until then, what should you do?

1. Start using discovery-go/node
2. Don't add anymore discovery-style environment variables to pillar.
3. When deploying to production, keep an eye for errors like this:
   ```
   Mismatch between generated env-vars and pillar values:
   Repo: json-processor
   Env-var: SERVICE_GEARMAND_TCP_PROTO
   generated: "http"
   pillar: "tcp"
   Tell infra in #eng-infra
   Let's make Discovery great again!
   ```
   and tell infra about it.
4. If you create a new service, tell infra and add it to this file:
   ```
   https://github.com/Clever/fabulous/blob/master/discovery-mappings/production.yaml
   ```
5. Most importantly, list all your dependencies in your launch yaml.

Why are we doing this?

One day (April 1st) our journey will be complete and so much cruft will be removed you're pillar files and launch yamls. Soon after we'll be able to render accurate dependency graphs and our discovery libraries will become more usable.

Ultimately, we're doing this make our transition from Marathon/Mesos to ECS safer. After April 1st, we'll be one step closer to a unified deployment process. No more standalone workers.

The process of Discovery started just now when Nate/Ryan did his incantation. Once this is done we'll...
About Me

Erica von Buelow
Software Engineer, CoreOS

@evonbuelow
evb@coreos.com
https://coreos.com
Thank you!

Erica von Buelow

@evonbuelow

evb@coreos.com

https://coreos.com