Managing Microservices Effectively

Daniel Hall (@smarthall)

About Me

- Systems Engineer at LIFX
- Making the 'Internet' in the Internet of Things

About This Talk

- This is how we do things at LIFX
- Feel free to ask questions as we go
- It works for us, it might not work for you
- Think about how each bit fits into your situation

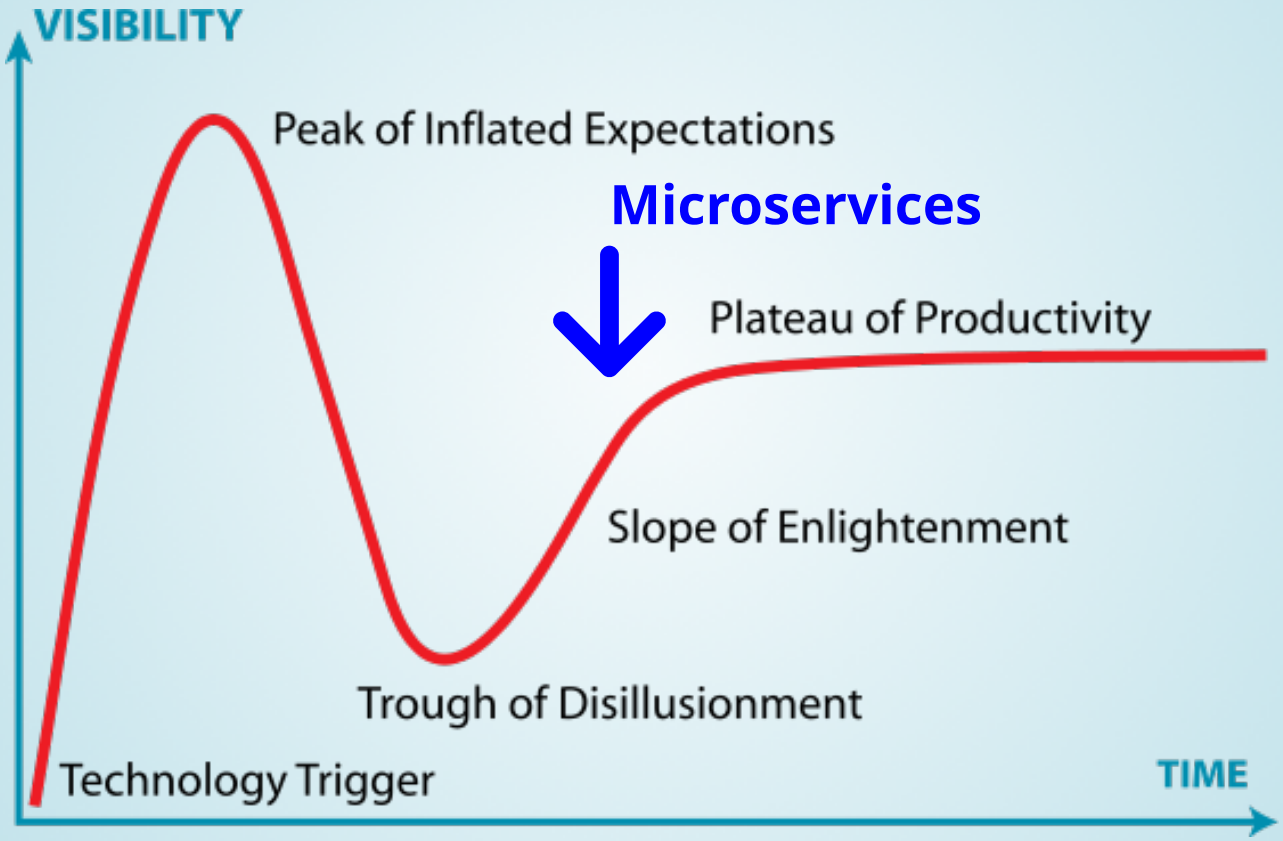
Step One: Write your apps

- You may not get input into this part
- Micro services are popular at the moment
- Design pattern that works with continuous delivery

Microservices

- Try to keep as much state outside your apps
- Don't make them too small, they're not nanoservices
- Don't make them too big, they're not milliservices
- Each service should be
  - Replacable
  - Independently Deployable
  - Have a single capability (billing, authentication)
- Think about information flow and circular dependencies

The Hype Curve



(http://commons.wikimedia.org/wiki/File:Gartner_Hype_Cycle.svg)

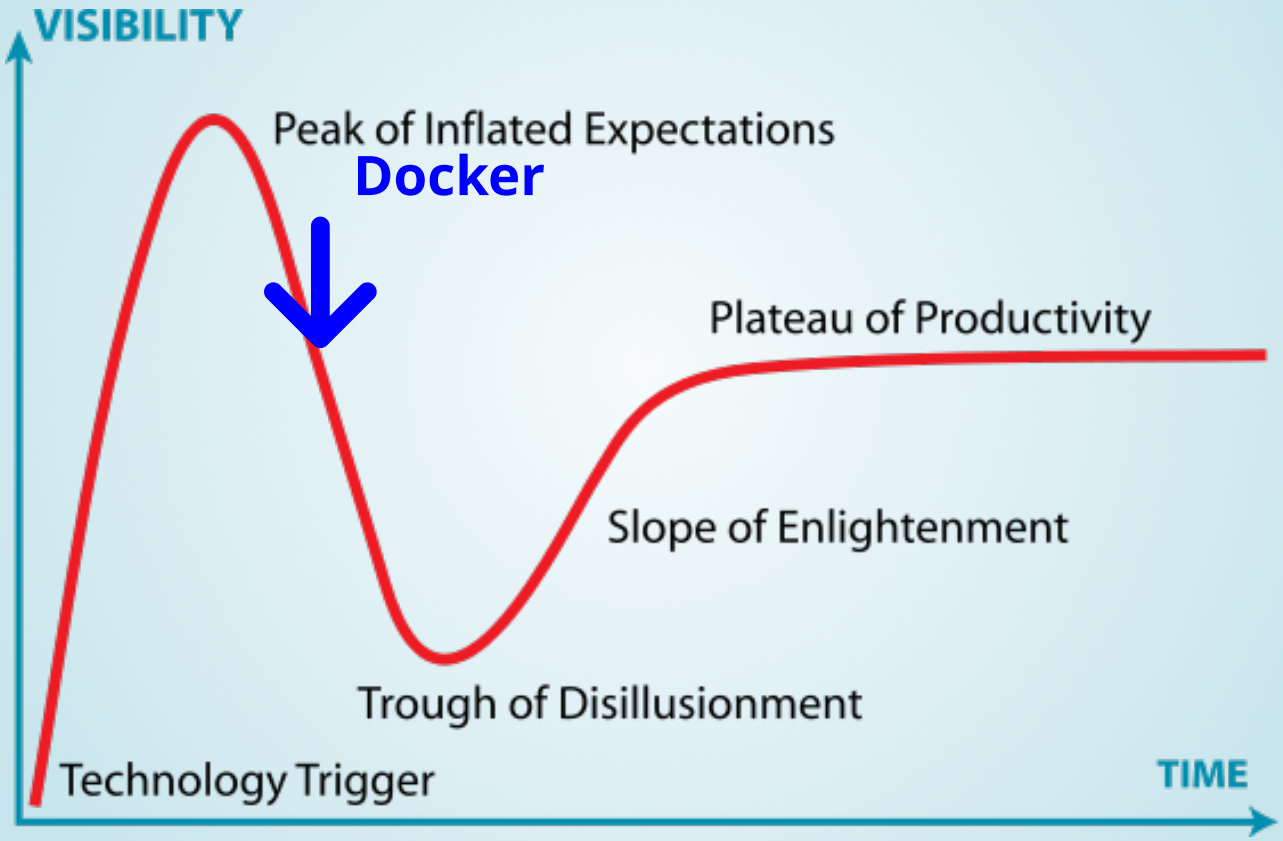
Step Two: Packaging

- All dependencies need to be available
- Needs to be small or cachable
  - Faster install means faster deployments
- You might want multiple versions on the same machine
- Preferably it works in several environments

Docker

- Filesystem layers stacked on top of each other
- Uses Linux containers to isolate applications
- You can run a local Docker registry
  - Security
  - Speed
- You can run it locally in dev and on your servers
- Less of 'it works on my laptop'
- Minuscule performance hit compared to VMs

The Hype Curve



(http://commons.wikimedia.org/wiki/File:Gartner_Hype_Cycle.svg)

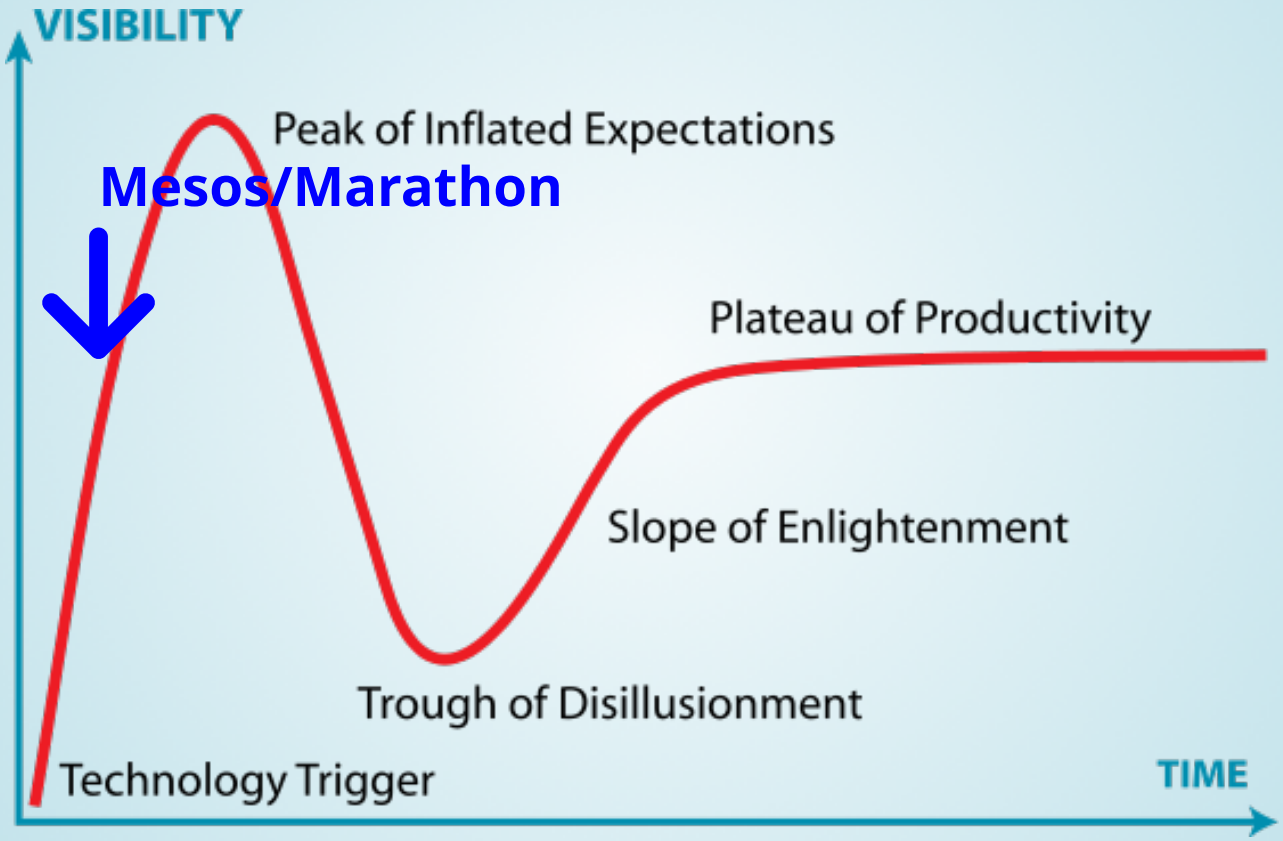
Step Three: Deployment

- As fast as possible
- Preferably minimal interaction
- Recovery from failures

Mesos/Marathon

- Mesos manages tasks running on a cluster
- Marathon coordinates long running jobs
- You submit a JSON job description to Marathon
- Marathon handles switching from the old app to new
- Marathon will also handle task failure and recover
- Health checks ensure broken tasks get replaced

The Hype Curve



(http://commons.wikimedia.org/wiki/File:Gartner_Hype_Cycle.svg)

Extra Credit: Sheduling

- Some things need to run repeatedly
- Cron works, but its not really HA
- HA Crons exist but can be complex
- Your cluster probably has spare capacity

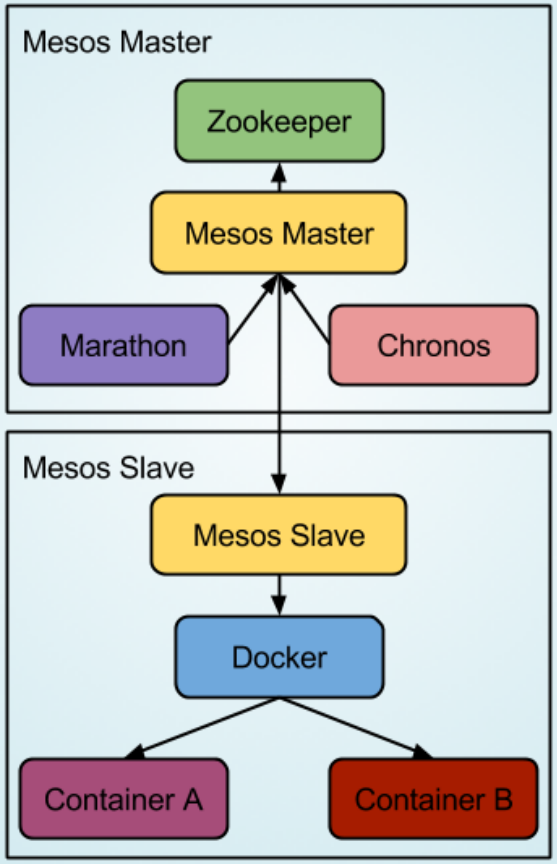
Chronos

- Chronos runs your scheduled tasks in Mesos
- Uses ISO8601 intervals to specify schedules
- Use your spare capacity for repeating tasks
- Can rerun failing jobs
- Can handle job dependencies
- Records stats on run times for jobs

The Hype Curve

(http://commons.wikimedia.org/wiki/File:Gartner_Hype_Cycle.svg)

Summary


Mesos Master
Zookeeper
Mesos Master
Marathon
Chronos
Mesos Slave
Mesos Slave
Docker
Container A
Container B

Demo

https://github.com/smarthall/ansible-mesos

Demo Time!

- All the code is on Github
  - https://github.com/smarthall/ansible-mesos
- 'vagrant up' will give you a development cluster
- './init-cluster.sh' will add some sample apps