@cassilup • JS (& others) • tim.js • 3PG

LONDON NODECONF 2016

Wed 11th May 2016 • The Barbican, London


barbican
art/theatre/music
dance/film/education
conferences/library
restaurants/bars


london.nodeconf.com

1
day

4
sections

11
talks

Start it

Build it

Ship it

Use it

Kassandra Perch
Robots, code and people
@nodebotanist

Nikhila Ravi
Server-less architecture in the wild
@nikhilaravi

James Halliday
The disintermediated web
@substack

Emily Rose
How I learned to stop worrying
@nexxylove

Andrew Martin
Avoiding release paralysis
@sublimino

Jan Lehnardt
Kill all humans
@janl

Christian Heilmann
Making ES6 happen with ChakraCore and Node
@codepo8

Colin Ihrig
State of Node.js Core
@cjihrig

Thomas Watson
It goes to eleven
@wa7son

Luca Maraschi
SWIMming in the microservices ocean
@lucamaraschi

Matt Clark
Node.js that's hugely reliable, fast, and scalable
@matthew1000

Start it

Build it

Ship it

Use it

Start it

Nikhila Ravi

Server-less architecture in the wild

@nikhilaravi

Serverless !== No Server

What it actually means:

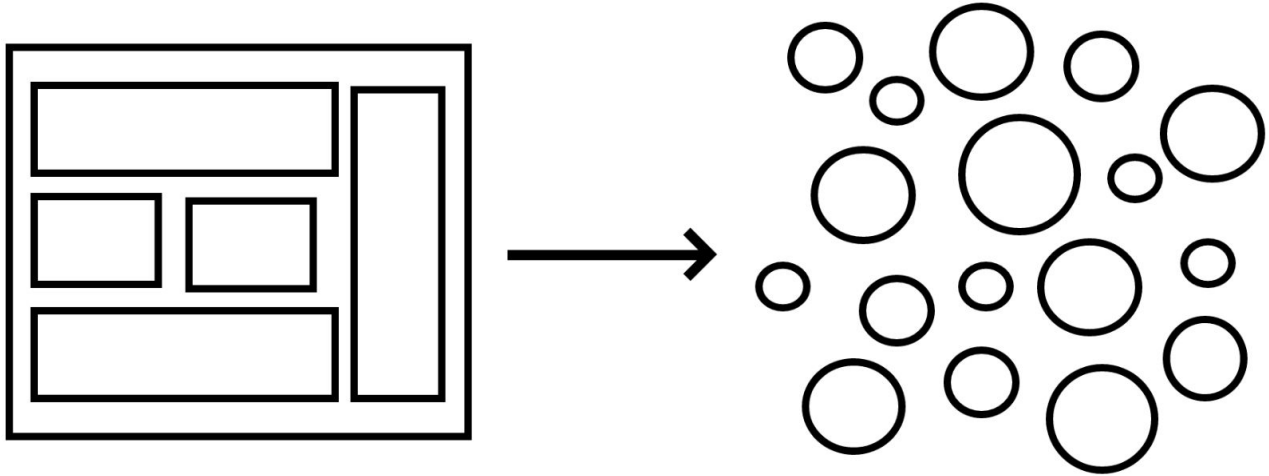
Event Driven

Stateless

Microservices

In the Cloud

Use a microservices architecture



Monolith

Microservices

CLOUD FUNCTIONS

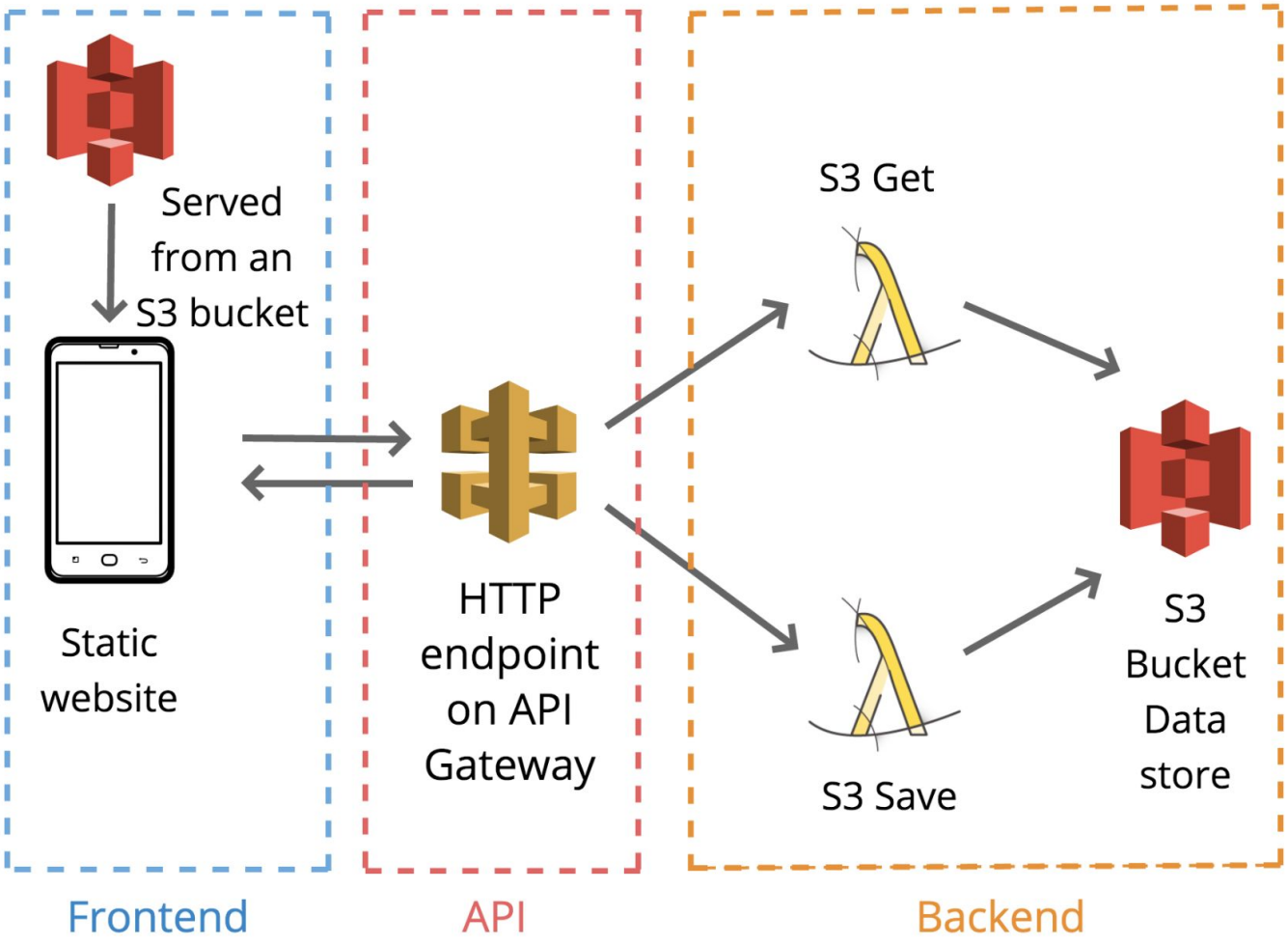
Create small, single-purpose functions that respond to events in the cloud

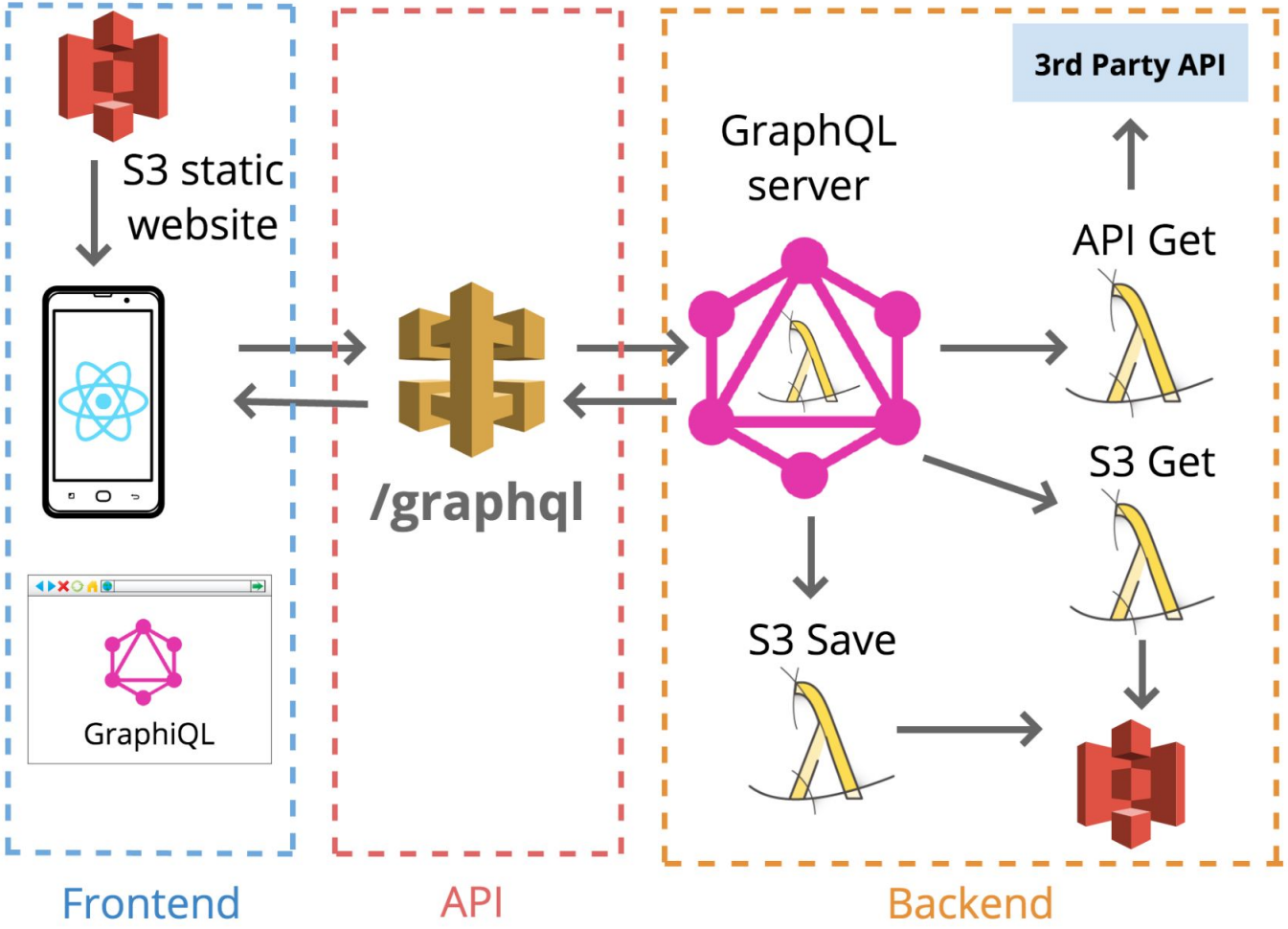
Implementations

- AWS Lambda
- IBM OpenWhisk
- Google Cloud Functions
- Iron.io
- Firebase

AWS Lambda

Run code without thinking about servers.
Pay for only the compute time you consume.


Served
from an
S3 bucket
Static
website
HTTP
endpoint
on API
Gateway
S3 Get
S3 Save
S3
Bucket
Data
store
Frontend
API
Backend


3rd Party API
GraphQL server
API Get
S3 static website
S3 Get
/graphql
S3 Save
GraphiQL
Frontend
API
Backend

The Good Parts

- Deploy each function individually
- Each micro-service is simple
- Piggyback off existing AWS Infrastructure
- Very attractive pricing model + generous free tier
- Lambdas and endpoints are easily disposable
- Versioning for CI/Prod

- More complex integration testing
- Debugging
- Many moving parts
- Latency issues if multiple lambdas executed in series

How to tame your microservices

- One repo per service
- Document interfaces very clearly
- Deployment scripts instead of manual deployments
- Architecture diagrams
- Local invoke scripts/tests so you don't have to deploy the service to test it
- Don't hop more than 3 services deep

http://slides.com/nikhilaravi/serverless-architecture-in-the-wild

Slides

Talk

https://www.youtube.com/watch?v=SwJUH3Le91s

James Halliday

The disintermediated web

@substack

the problem

too much technology depends on centrally controlled
ers:

* requires an internet connection
* high latency
* copyright/political/economic censorship

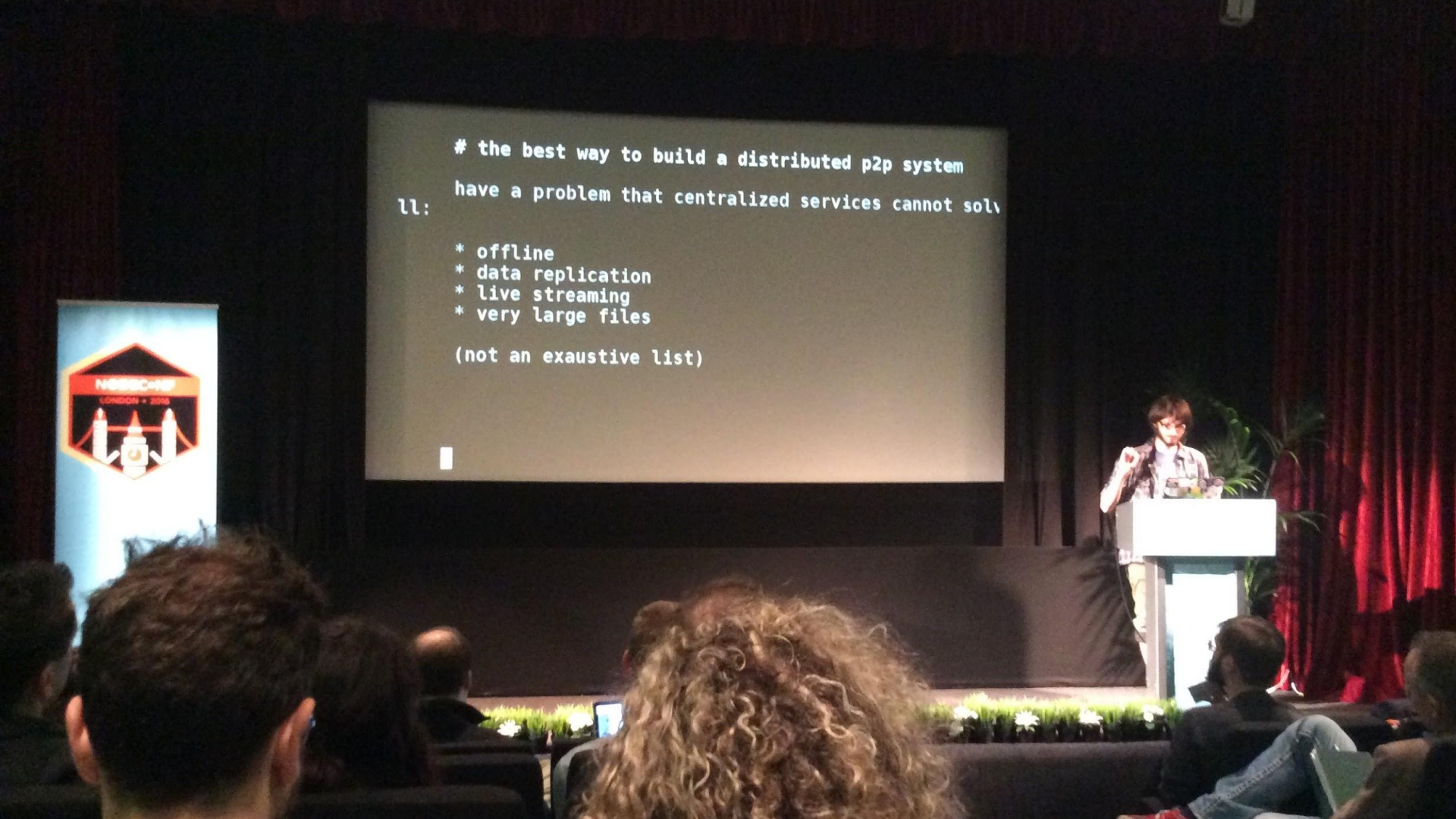
p2p

* datacenters are already distributed systems
* move the algorithms out to the clients
* then cut out the middlemen (servers)

things that are easy with p2p

and hard/impossible with centralized services:

* infinitely scalable file transfer
* log-based data replication
* offline/marginal networking



the best way to build a distributed p2p system
have a problem that centralized services cannot solv
ll:
* offline
* data replication
* live streaming
* very large files
(not an exaustive list)

https://github.com/substack/nodeconf-london-2016

Slides

Talk

https://www.youtube.com/watch?v=6jcQoSraHcw

Start it

Build it

Ship it

Use it

Christian Heilmann

Making ES6 happen with ChakraCore and Node

@codepo8

① The JavaScript promise and problem

② The Node revolution

③ Problems with the node approach

④ Making ES6 happen - or not

⑤ OMG, not a second engine!

⑥ Looking ahead

⑦ My hopes and wishes

① The JavaScript promise and problem

② ③ ④ ⑤ ⑥ ⑦

- Making the web **much more interactive**
- Easy to learn **without any tooling overhead** - use whatever you want.
- Doesn't need any compilation or conversion step - **works straight up in the browser**
- Forgiving and dynamic language allows for lot of **different and adventurous development** styles

Node to the rescue

- With Node we **liberated** ourselves from the **woes of the web**
- We could concentrate on **using JavaScript** and build all kind of things from micro services and bespoke servers up to huge applications
- We **control** the environment
- We can **innovate the approach to using JavaScript** and imitate, build upon or improve what other languages do

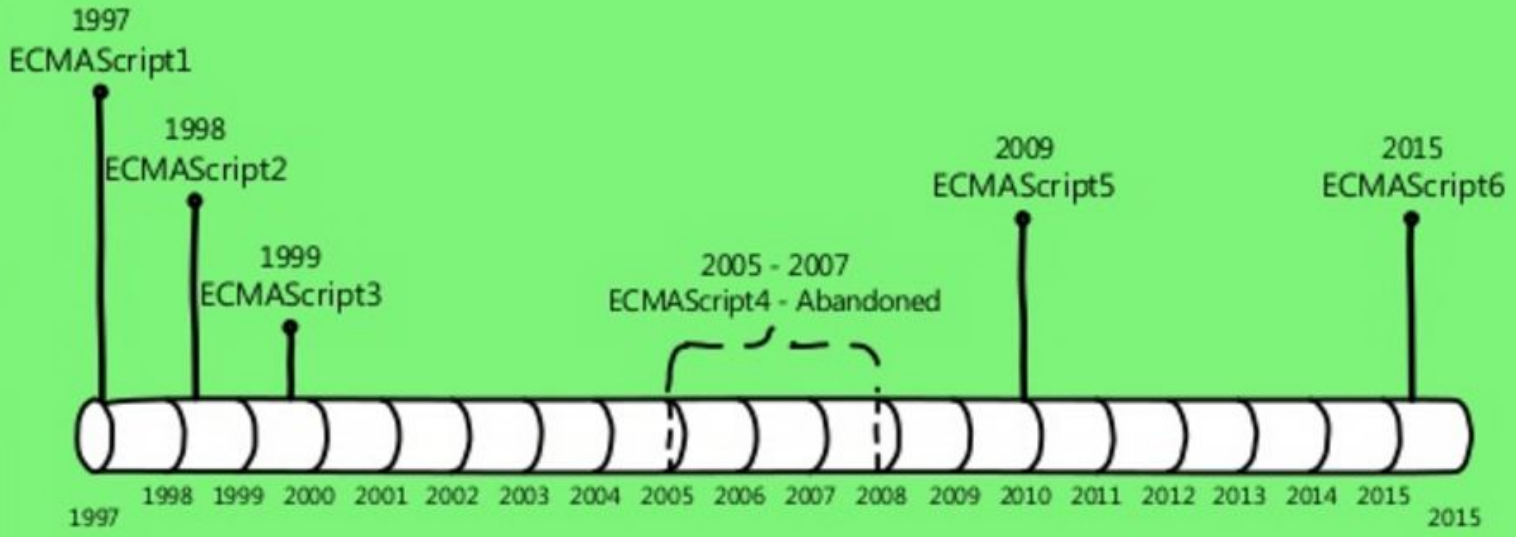
Problems with the node approach

- We limited ourselves to one JavaScript engine - **monoculture is never a good plan**
- We build towards the **capabilities of one engine** instead of a standard
- We run into the danger of **relying too much on dependencies** (see: left-pad)
- **Politics** and a harsh, very engineering, comp-sci driven approach
- "Hey, it works..."

Making ES6 happen - or not

- ES6 is a great opportunity to **clean up our act** as a JavaScript community
- We have a new, ratified **standard**
- This standard has truckloads of good features that **make library usage unnecessary**
- It is also a great opportunity to **re-vamp our educational materials** and fade out old, bad advice

Making ES6 happen - or not



OMG, not a second engine!

- ChakraCore is now **available** to power node
- Breaking the **monoculture**
- Highly optimised to **run in low end environments**
- **Brand new engine**, without overhead from old browser dependencies

Looking ahead

- Node should be the place where we **kick the tires of JavaScript**
- We **build the tools** we use to build things in Node, so let's not repeat the mistakes we did on the web
- We can **educate developers in our tools** rather than outside it
- If we want node to take off, we also need to embrace the **needs of people outside our community**.

My hopes and wishes

1. Less **drama** and more **documentary**
2. Embrace **constant change** and avoid **monoculture**
3. Don't forget to **keep things simple**
4. Stop making **computer science a religion** - the web was built **by everyone**
5. User choice and user experience **should always trump** developer convenience
6. Teaching **beats** preaching...

7

http://www.slideshare.net/cheilmann/nodeconflondon-making-es6-happen-with-chakracore-and-node

Slides

Talk

https://www.youtube.com/watch?v=jHTSCXQ3-hU

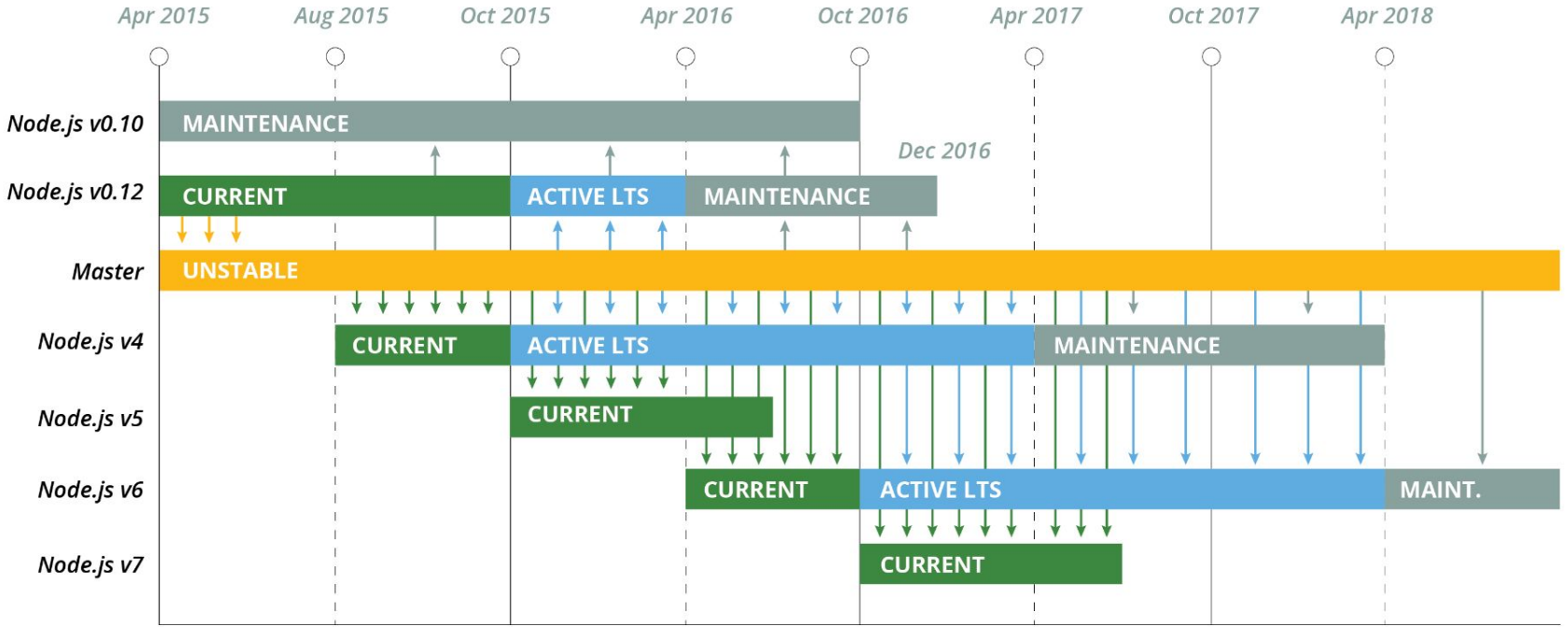
Build it

Colin Ihrig

State of Node.js Core

@cjihrig

Node.js Long Term Support Release Schedule

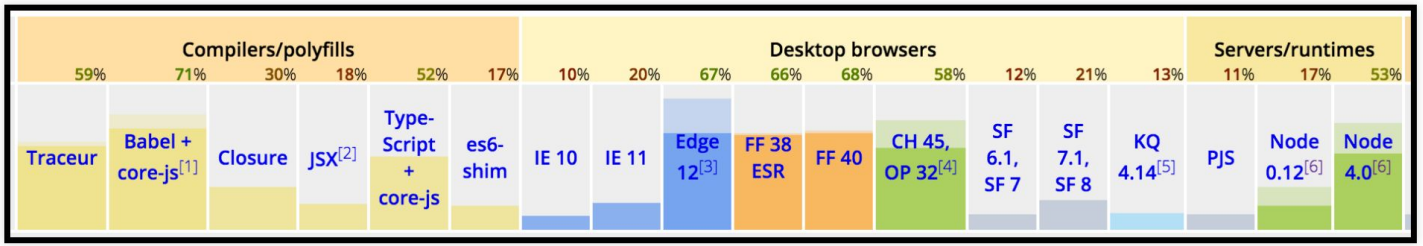


Details: **https://github.com/nodejs/LTS**

ES6 Features in LTS

- Block scoping (let / const)
- Arrow functions
- Symbols
- Object literal extensions
- Binary and Octal literals
- Template literals
- Classes
- Collections (Map, Set, etc.)
- Promises
- More!

https://nodejs.org/en/docs/es6/



Node.js v6.0.0

- Release on April 26, 2016
- Hey look, a slightly different logo!
- Includes V8 5.0, same as Chrome
- Drops support for Windows XP and Vista
- Drops support for OS X < 10.7
- Improved IBM support and PPC binaries
- v6.x.x to follow Argon in LTS in October 2016

ES6 Features in v6.0.0

- Spread operator
- Default parameters
- @@toStringTag, etc.
- Regex sticky flag
- Rest parameters
- Destructuring
- Proxies
- new.target

96% ES6 support. 93% without flags

ES6 Modules

- Node: Hey, I made the largest module system in the world.
- TC39: I'm going to invent JavaScript modules.
- Node: But... check out what I did!
- TC39: Hey look, I invented JavaScript modules!

https://github.com/nodejs/node-eps/pull/3

Integration of DevTools

- Proof of concept JS DevTools integration with Node
- Provides step debugging, sourcemaps, profiling, and more
- Requires WebSockets support
- Full discussion: nodejs/node#2546
- Try it out: repenaxa/node v8_inspector branch

https://talks.continuation.io/nodeconf-london-5-16

Slides

Talk

https://www.youtube.com/watch?v=kn0Trjl7ZRE

Build it

Thomas Watson

It goes to eleven

@wa7son

An oversimplified example

```
function doMath (a, b) {
  // ...do stuff with a...
  return a + b
}

function prepareMath (a) {
  // ...do stuff with a...
  return function doMath (b) {
    return a + b
  }
}

function prepareMath (a) {
  // ...do stuff with a...
  var src = 'return ' + a + ' + b'
  return new Function ('b', src)
}
```

Performance
Boost

https://github.com/watson/talks/tree/master/2016/05%20NodeConf%20London

Slides

Talk

https://www.youtube.com/watch?v=ayvAx6yuMYc

Start it

Build it

Ship it

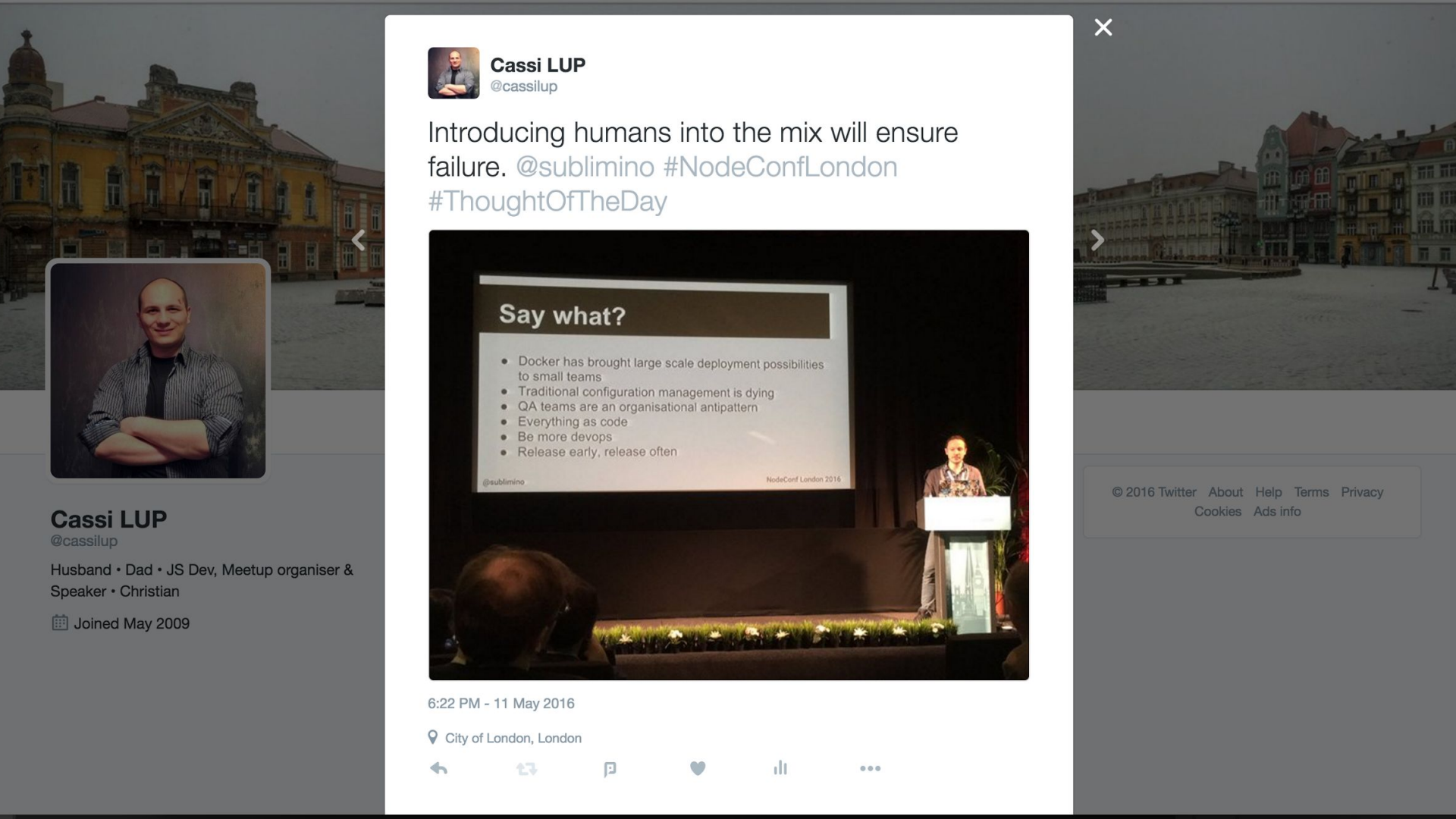
Use it

Ship it

Andrew Martin

Avoiding release paralysis

@sublimino



Cassi LUP
@cassilup

Introducing humans into the mix will ensure failure. @sublimino #NodeConfLondon #ThoughtOfTheDay

6:22 PM - 11 May 2016

City of London, London

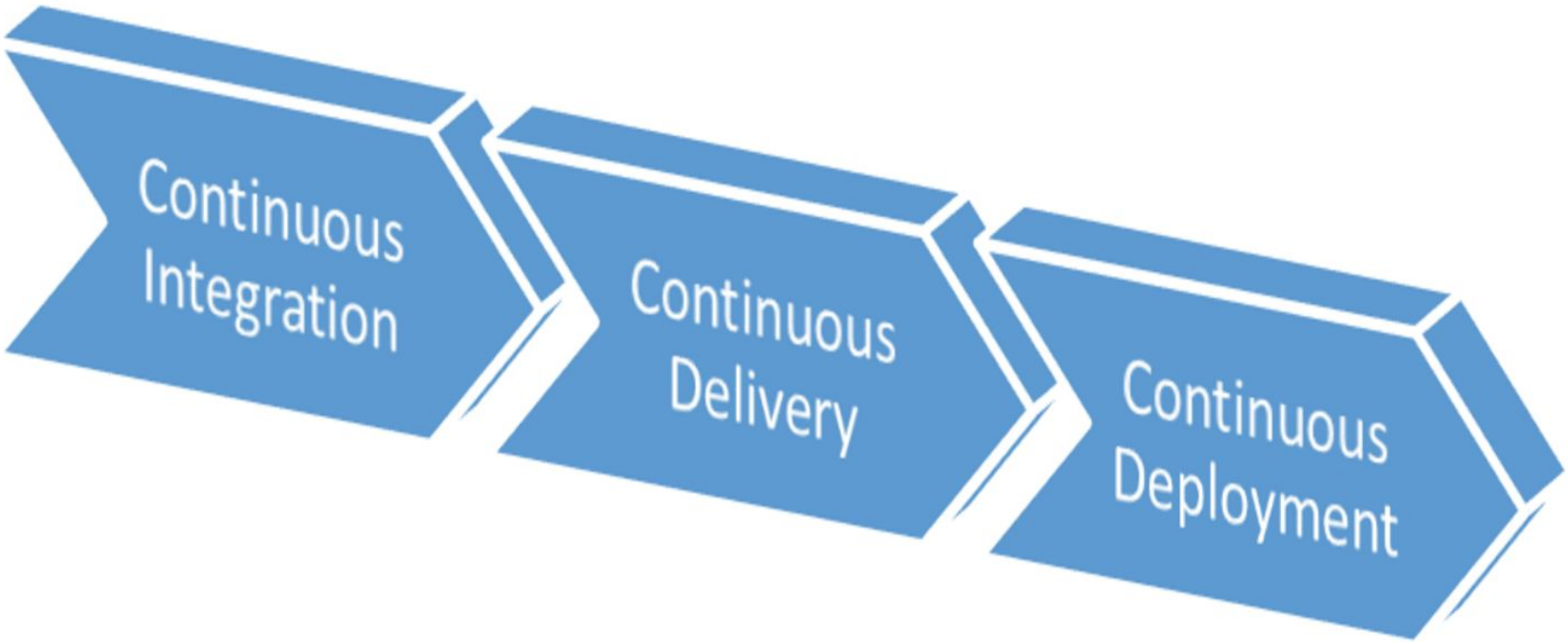
Cassi LUP
@cassilup

Husband • Dad • JS Dev, Meetup organiser & Speaker • Christian

Joined May 2009

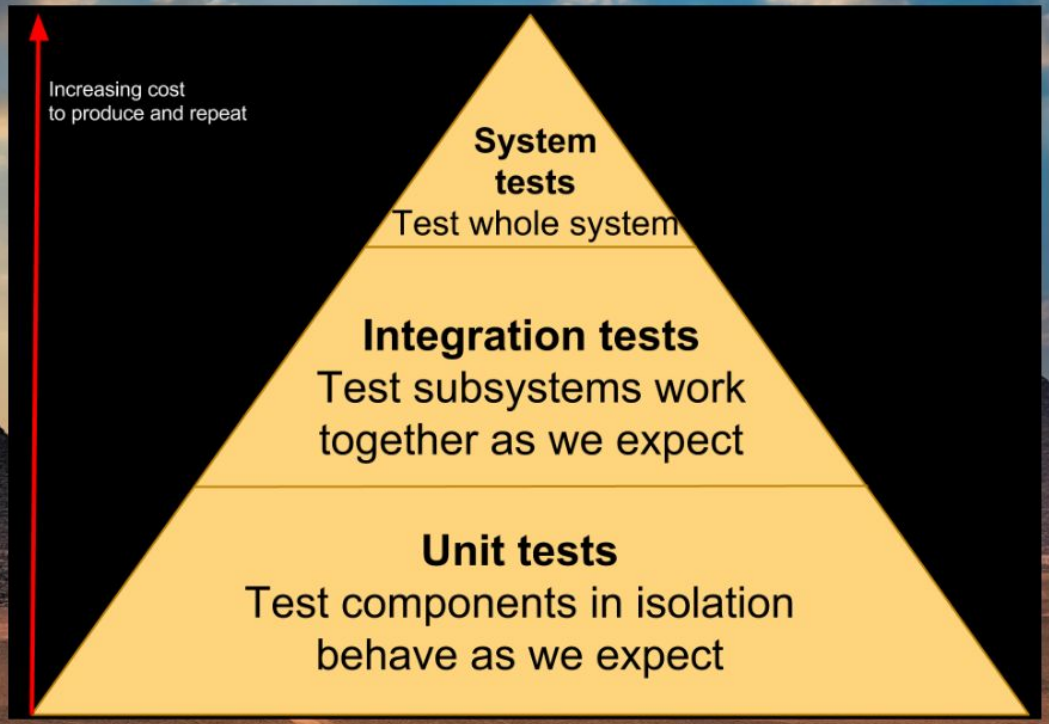
Say what?

- Docker has brought large scale deployment possibilities to small teams
- Traditional configuration management is dying
- QA teams are an organisational antipattern
- Everything as code
- Be more devops
- Release early, release often


Continuous Integration
Continuous Delivery
Continuous Deployment

CX Best Practices

The Test Pyramid



Pragmatic Unit Testing – CheatSheet 1.1

Right-BICEP		
Right	Are the results **right**?	Use data files, golden bits, oracle…
B	Are all **boundary** conditions CORRECT	Conformance, Ordering, Range, Reference, Existence, Cardinality, Time.
I	Can you check **inverse** relationships?	Apply logical inverse (e.g. sqrt vs pow).
C	Can you **cross-check** results using other means?	Use alternative way of achieving result.
E	Can you enforce **error conditions** to happen?	What errors could occur, e.g. environmental constraints.
P	Are **performance** characteristics within bounds?	Quick regression test of performance characteristics.

CORRECT		
C	Conformance	Any specific data format? What if the format is different?
O	Ordering	Order of data or position of an element? Reverse order?
R	Range	Start and end of indices, first is greater than last, index is negative, index is greater than allowed, count doesn't match number of actual number of items…
R	Reference	Any external dependencies? Any preconditions? Do we guarantee post-conditions?
E	Existence	Does some given thing exist? Null, blank, empty, 0 (zero)?
C	Cardinality	12 feet lawn, each section 3 feet, how many poles required? 4? Actually 5! Test for how many things there might be: Zero, one or more than one: 0-1-n-Rule.
T	Time	Relative time (ordering in time), absolute time (elapsed time), concurrency issues?

A-TRIP (Good Tests)		
A	Automatic	Running the test and checking the results should be automatic.
T	Thorough	Well-tested methods may have 4-5 asserts.
R	Repeatable	Run over and over again, in any order produce the same results.
I	Independent	Keep tests tight, focused, test only one thing at once.
P	Professional	Write real code, refactor.

https://t.co/jaYQUKqnFi

Slides

Talk

https://www.youtube.com/watch?v=vOMcJ3kRMVo

Jan Lehnardt

Kill all humans

@janl

npm is Lego
for your code

npm only works if you can trust people

Version numbers are for computers, release names are for humans.

```
$ npm install -g
semantic-release-cli

$ semantic-release-
cli setup
```

This is what happens in series:

1. `git push`	2. `semantic-release pre`	3. `npm publish`	4. `semantic-release post`
New code is pushed and triggers a CI build.	Based on all commits that happened since the last release, the new version number gets written to the `package.json`.	The new version gets published to `npm`.	A changelog gets generated and a release (including a git tag) on GitHub gets created.

```
$ npm install -g
greenkeeper
```

Greenkeeper will do its job automatically. If your dependencies are already outdated the first thing you are going to notice is a Pull Request where we update all your dependencies in your repository's package.json to their respective latest versions. Then, whenever one of your dependencies is updated on npm, you will receive a Pull Request to update your repository accordingly.

http://jan.prima.de/u/Kill-All-Humans.pdf

Slides

Talk

https://www.youtube.com/watch?v=ZXyx_1kN1L8

Start it

Build it

Ship it

Use it

Use it

Luca Maraschi

SWIMming in the microservices ocean

@lucamaraschi

- first came Java, then came microservices
- Node.js embraced µservices
- Success === More Users
- Amazon started with books, moved towards cloud + hardware (EC)

DOCKER...

DOES NOT SOLVE THE PROBLEM!

NO RESILIENCY

NO FAULT TOLERANCE

SWIM: *S*calable *W*eakly-consistent *I*nfection-style Process Group *M*embership Protocol

Abhinandan Das, Indranil Gupta, Ashish Motivala*
Dept. of Computer Science, Cornell University
Ithaca NY 14853 USA
{asdas,gupta,ashish}@cs.cornell.edu

Abstract

1. Introduction

SCALABLE WEAKLY CONSISTENT INFECTION STYLE MEMBERSHIP PROTOCOL

WEAKLY CONSISTENT

VS. STRONGLY CONSISTENT

PREDICTABLE ELASTICITY

https://github.com/lucamaraschi/presentations/blob/master/20160511_nodeconf_london

Slides

Talk

https://www.youtube.com/watch?v=TK7eeL3RQ4M

Matt Clark

Node.js that's hugely reliable, fast, and scalable

@matthew1000

BBC homepage is
entirely written in Node.

Sports page:
60k requests/second,
1.4M open connections.

Our 8 principles for fast, reliable and scalable Node apps

1. Design as micro services, embrace the elasticity of the cloud
2. Scale up like a rocket, and down like a feather
3. Understand your load, in theory and for real
4. An event driven approach helps scaling and load
5. Beware the cost of micro services
6. It's fine to test on production
7. Be two steps away from catastrophe
8. Become great at running Node infrastructure

Thanks for listening,
keep in touch

matthew.clark@bbc.co.uk
@matthew1000

johnathan.ishmael@bbc.co.uk

https://t.co/trCjlIA9yE

Slides

Talk

https://www.youtube.com/watch?v=pxmXiKlh5OU

THANK YOU

@cassilup