Microservices

Pros & Cons

James Allen (DevOps Engineer, Clear Measure)
Miguel Gonzalez (Chief Architect, BoonGroup)
Gabriel Schenker (Distinguished Architect, Clear Measure)
Andrew Siemer (Director of Engineering, Volusion)

What is a microservice?

- Small piece of software that does one thing really well
- Loosely coupled
- Separate data store
- Just enough to solve a problem
- Right technology for the job
- Autonomous
- Can update as often as is needed
- Intelligence in the service, not the routing/infrastructure/bus
- Immutable infrastructure

Should I use microservices?

- It depends!
- Likely no
- Fallacies of distributed computing:
  - The network is reliable
  - Latency is zero
  - Bandwidth is infinite
  - The network is secure
  - **Topology doesn’t change**
  - There is one administrator
  - Transport cost is zero
  - The network is homogeneous

When Should I start thinking about Micro Services?

- Many teams work on the same code base
- Merge hell, cross team dependencies, ...
- Huge monolithic application which is difficult to deploy
- Monolith cannot be scaled horizontally
- Different parts of the application have totally different requirements
  - CPU bound
  - I/O bound
  - Memory bound
  - etc.
- Some but not all areas of the application change frequently
- Development stack is outdated. New tools and patterns are hard if not impossible to embrace

How big should a microservice be?

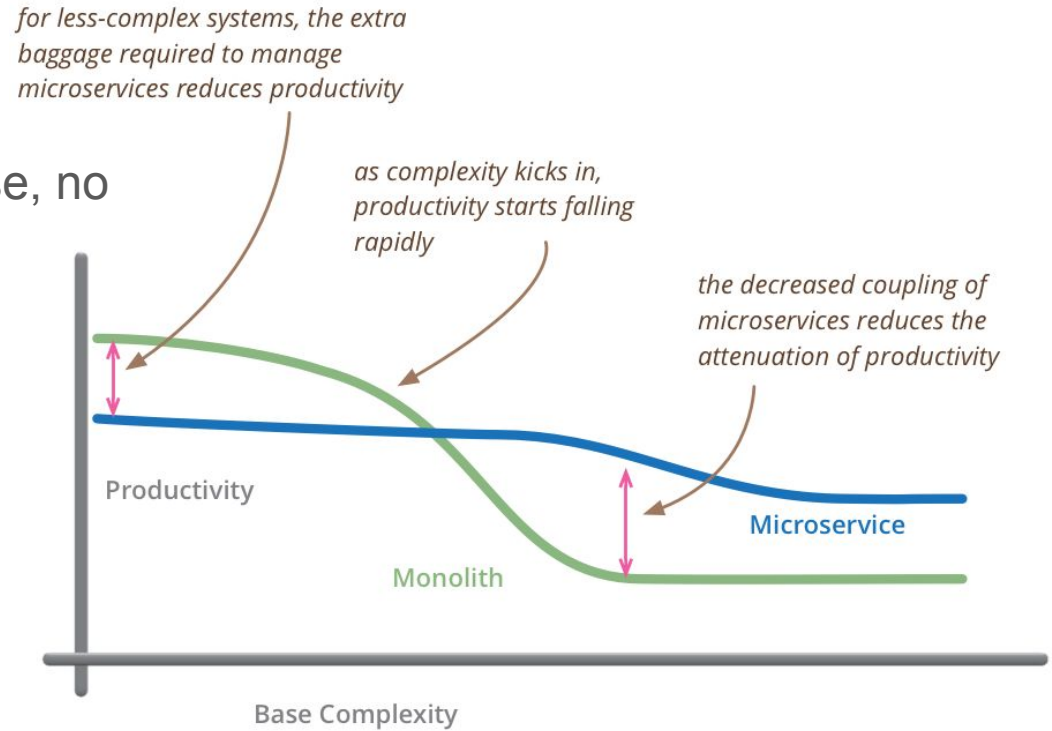
- Small enough to fit full context in your head
- Big enough to solve a problem
- Owned by one team

How should a microservice communicate?

- Synchronous
- Asynchronous
- Messaging
- Fan out
- HTTP/REST
- TCP/IP
- Pub/sub

Are microservices less complex?

- Code wise, yes
- Deployment wise, yes
- Infrastructure configuration wise, no



but remember the skill of the team will outweigh any monolith/microservice choice

How to deploy microservices?

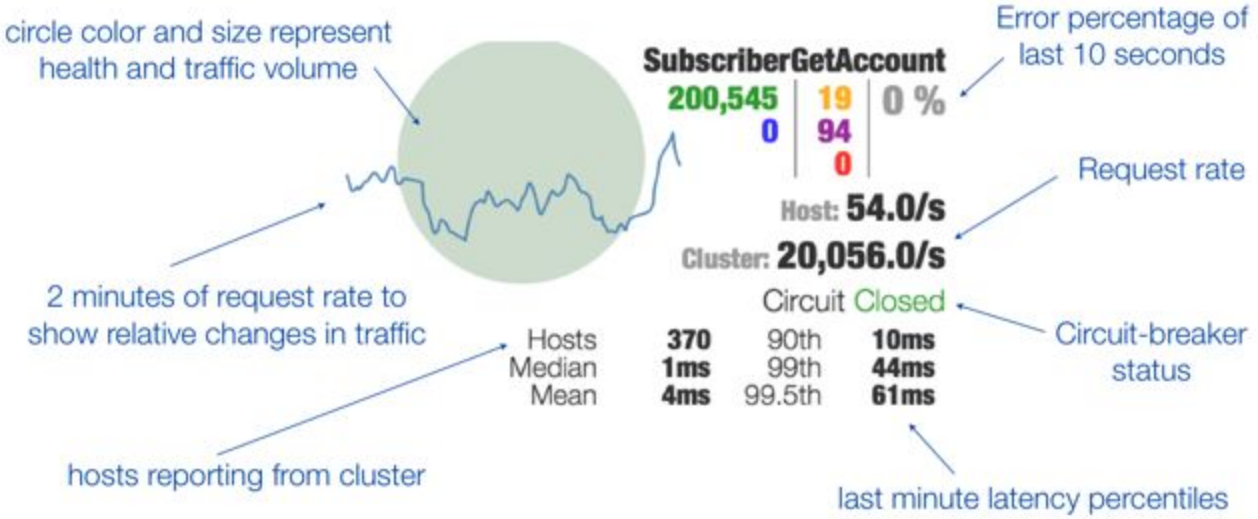
- AUTOMATE EVERYTHING
- Jenkins / TeamCity / Ansible / Chef / Puppet / Capistrano / StackStorm
- Consul / Consul-Templates / Registrator
- Kubernetes / Mesos / Docker Swarm
- Docker / Compose
- Nginx / HAProxy

Release strategy for microservices?

- Continuous Deployment
- Continuous Delivery
- Versioned
- Blue / Green
- A-B Testing
- Net new, add to routing

How to monitor microservices?

- Measure anything, measure everything
  - https://codeascraft.com/2011/02/15/measure-anything-measure-everything/
- Performance monitoring, exception monitoring, logs, metrics
- NewRelic, nagios
- statsd / graphite (hosted graphite) / kibana / grafana
- Centralized logging (logentries, logstash)
- Circuit Breaker (hystrix)



Rolling 10 second counters
with 1 second granularity

Successes	200,545	19	Thread timeouts
Short-circuited (rejected)	0	94	Thread-pool Rejections
		0	Failures/Exceptions

How do you handle failing services?

It’s not IF it will fail but WHEN it will fail!

- Auto healing
- Automated Remediation
- Circuit breaker
- Fallbacks
- Graceful degradation
- Don’t cascade failures
- Docker
- API Gateways (kong, azure, aws)