Deliver Faster and Spend Less with Cloud Native Microservices

Adrian Cockcroft @adrianco
Technology Fellow - Battery Ventures

O'Reilly Software Architecture Workshop - March 2015

BV
Battery Ventures

Agenda

Workshop vs. Presentation

Introductions

Faster Development

Microservice Architectures

Cloud Native Cost Optimization

Workshop vs. Presentation

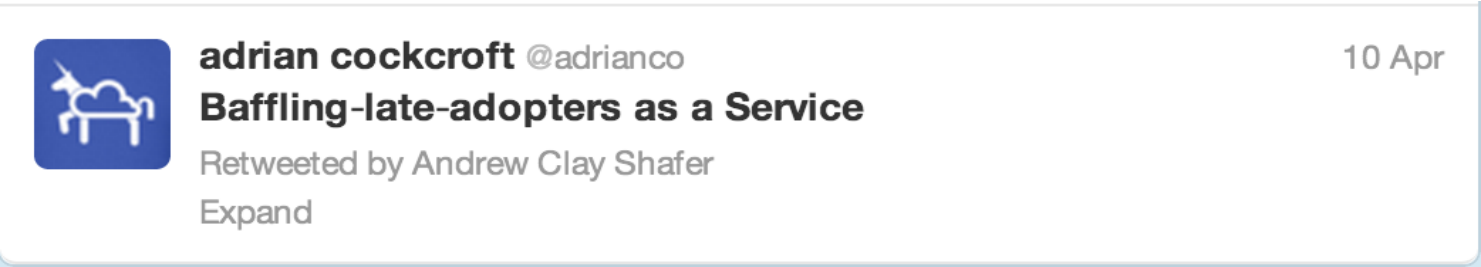
Questions at any time

Interactive discussions

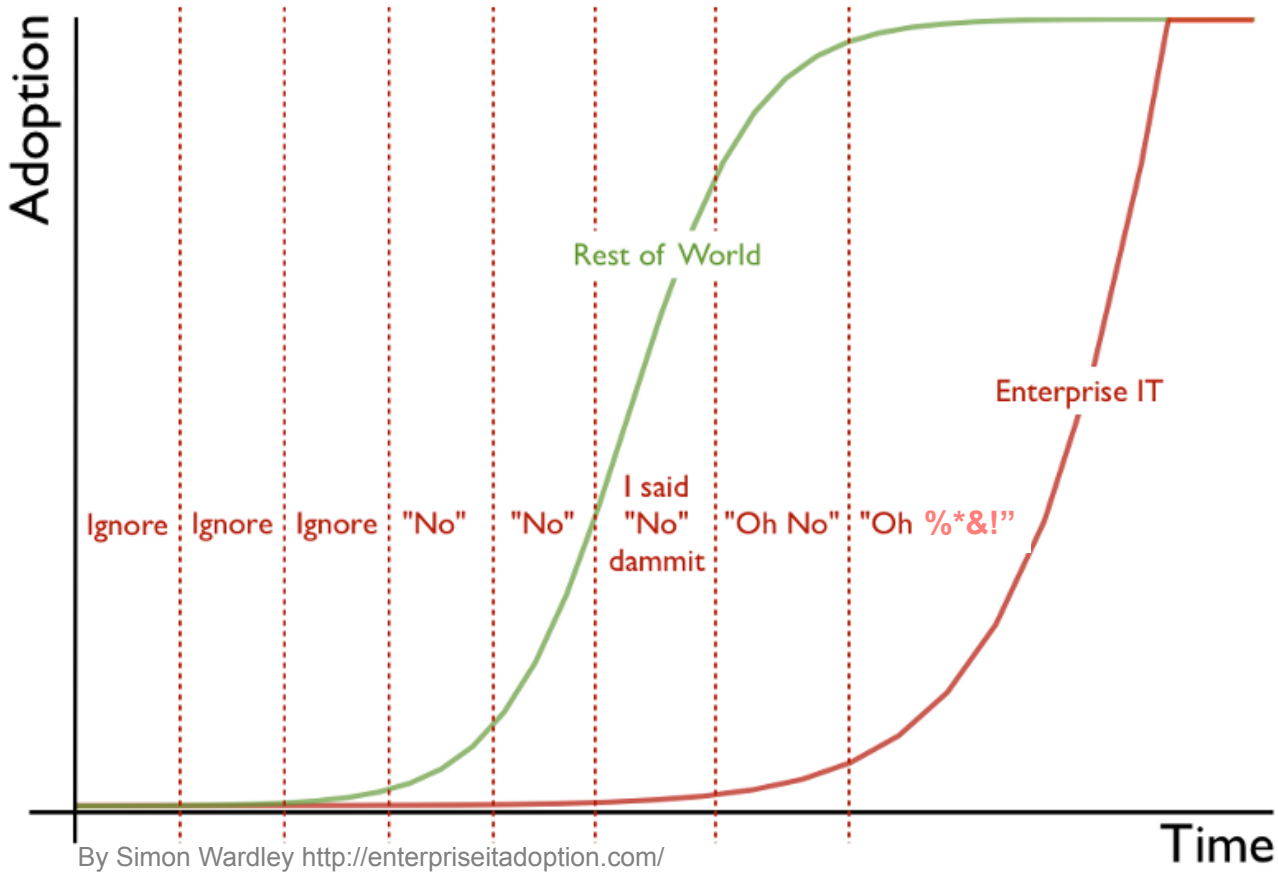
Share your experiences

Everyone's voice should be heard

This is me, who are you?

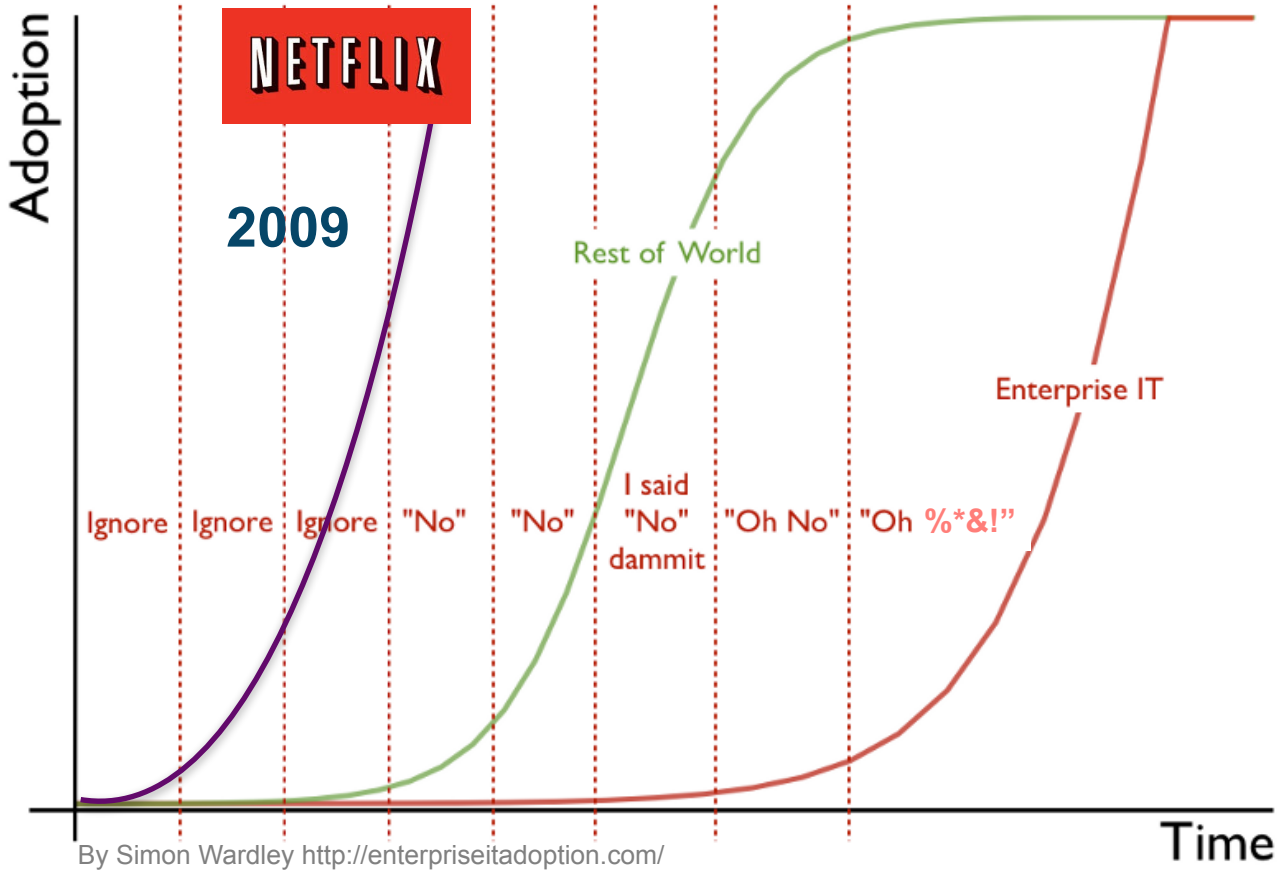


Why am I here?



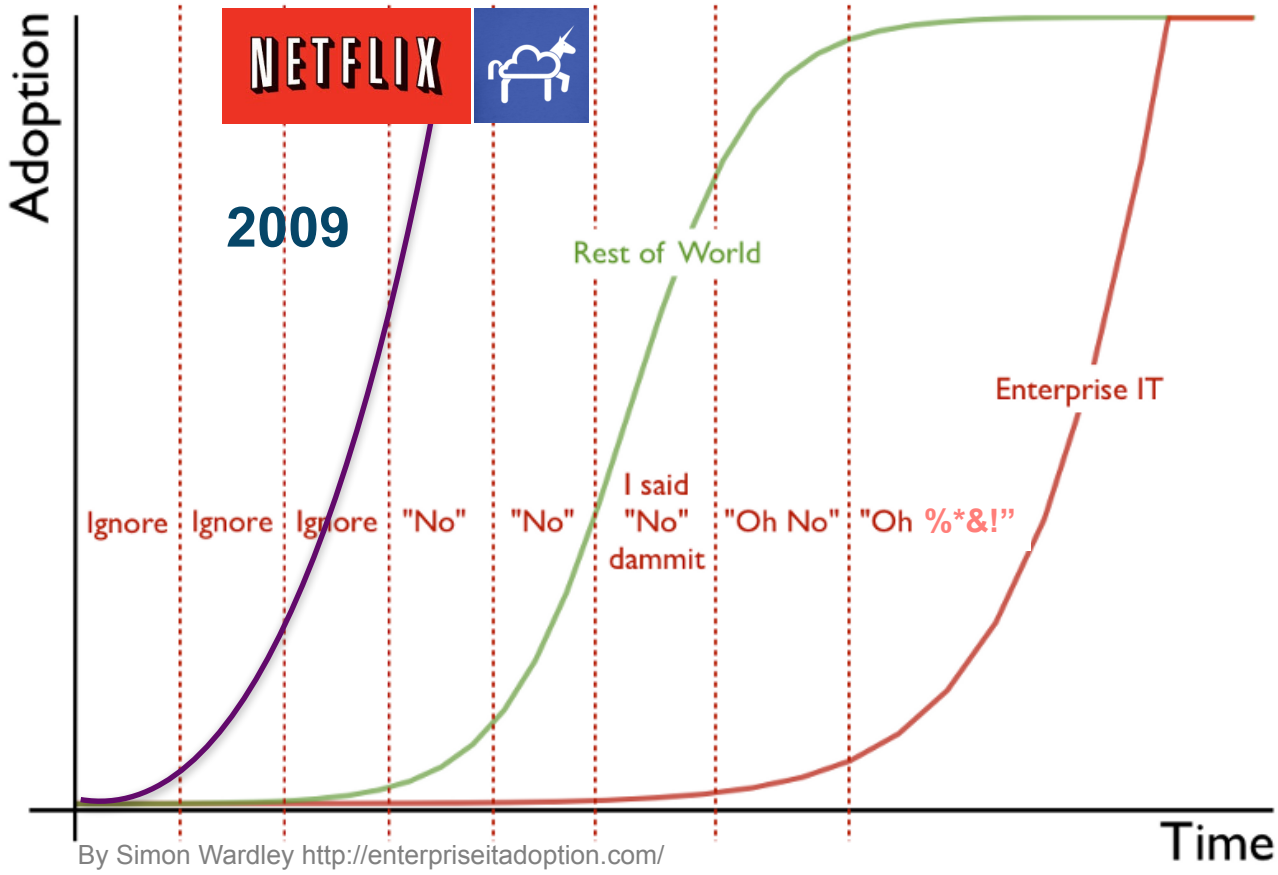
By Simon Wardley http://enterpriseitadoption.com/

Why am I here?



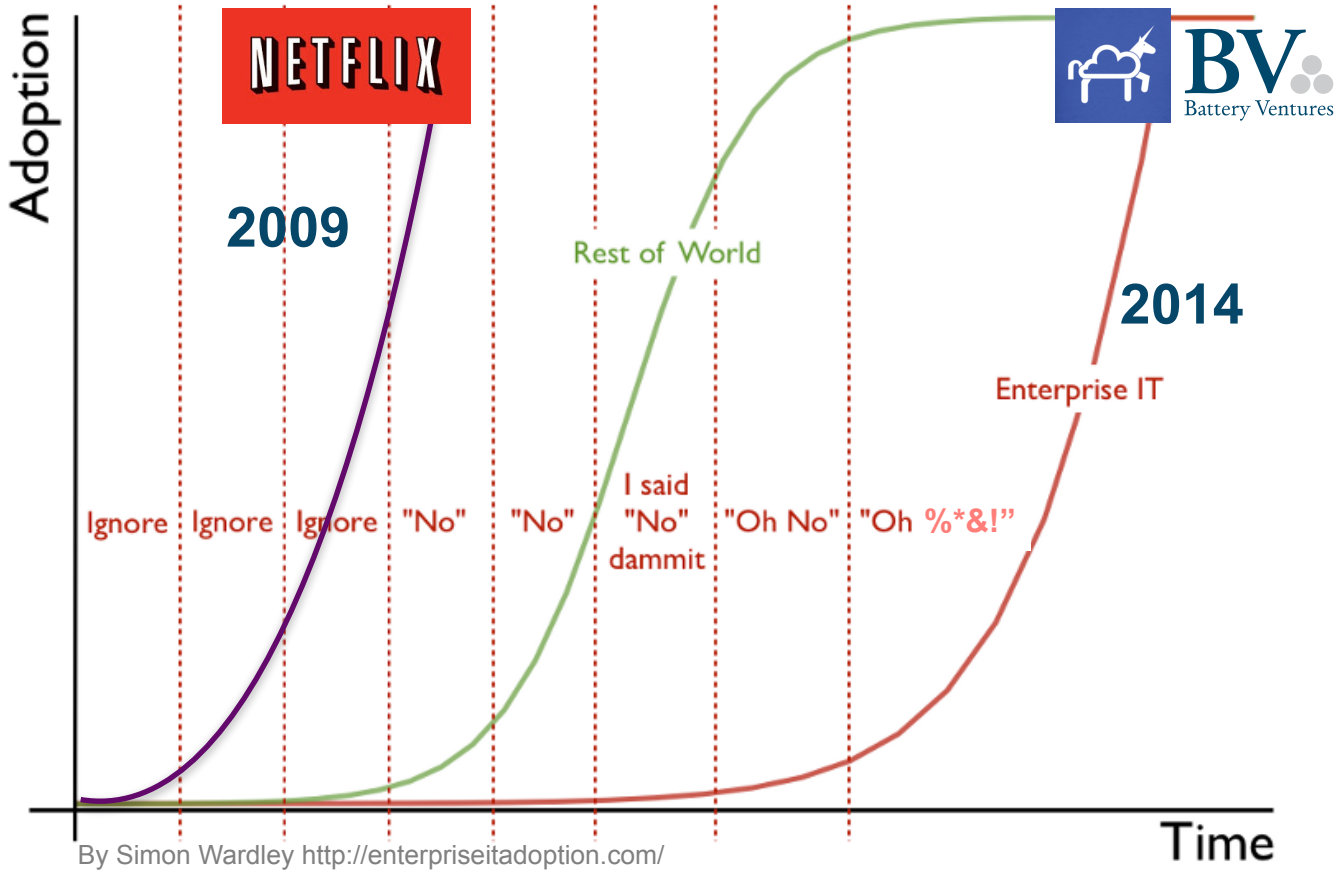
By Simon Wardley http://enterpriseitadoption.com/

Why am I here?



By Simon Wardley http://enterpriseitadoption.com/

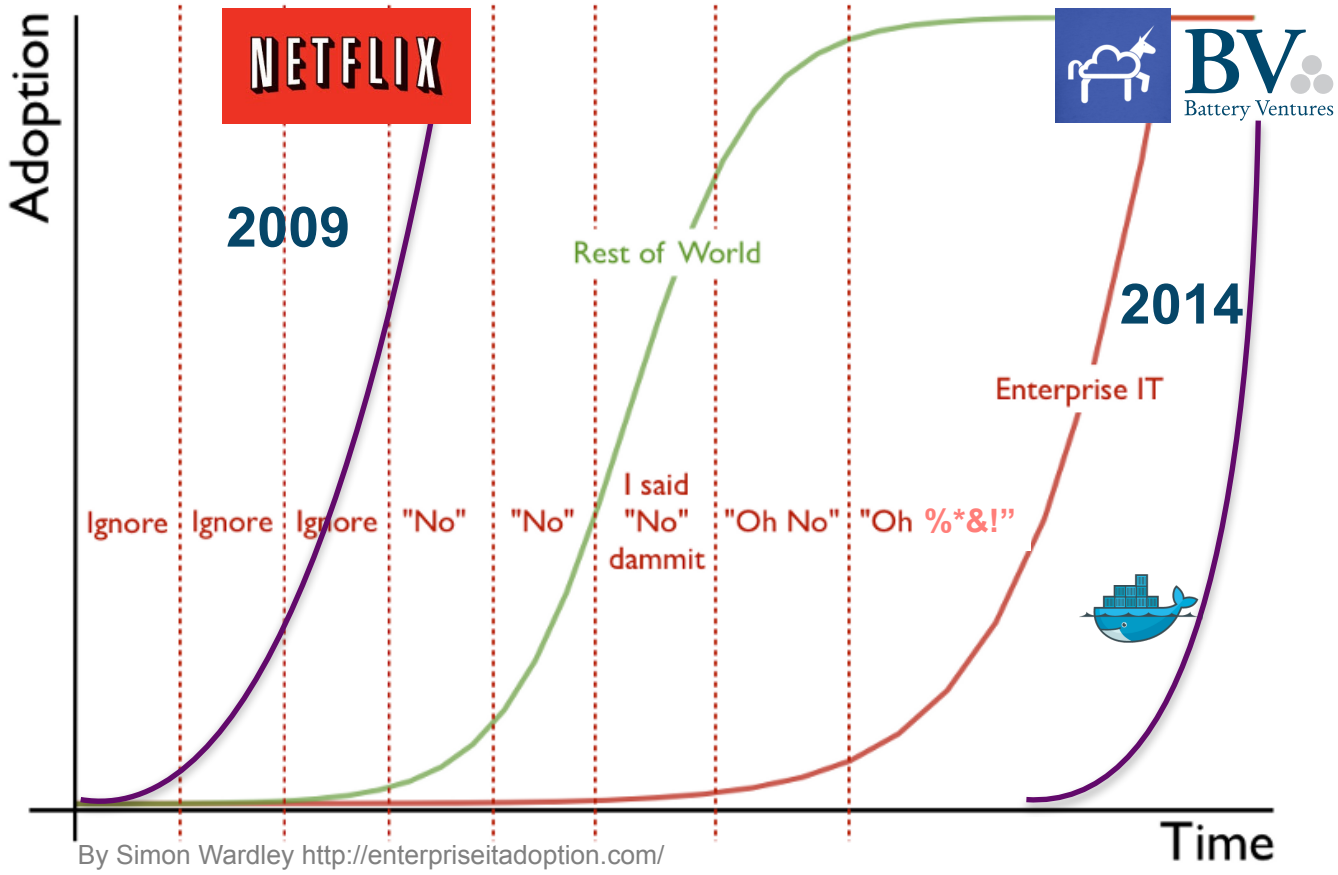
Why am I here?



BV Battery Ventures

@adrianco's job at the intersection of cloud and Enterprise IT, looking for disruption and opportunities.

Why am I here?



@adrianco's job at the intersection of cloud and Enterprise IT, looking for disruption and opportunities.

Example: Docker wasn't on anyone's roadmap for 2014. It's on everyone's roadmap for 2015.

What does @adrianco do?

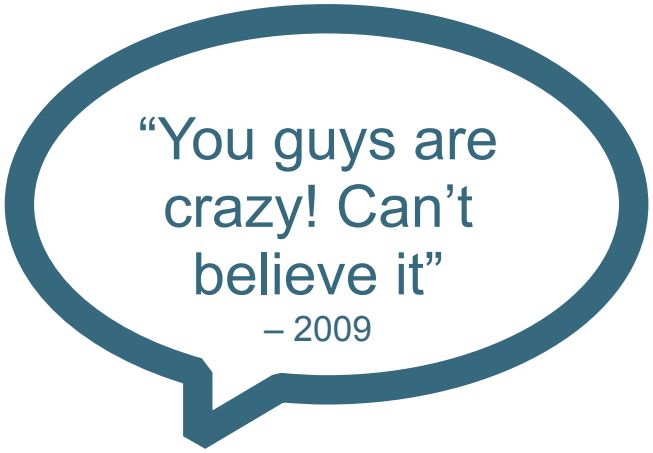
@adrianco

BV Battery Ventures

- Presentations at Conferences
- Maintain Relationship with Cloud Vendors
- Technology Due Diligence on Deals
- Presentations at Companies
- Technical Advice for Portfolio Companies
- Program Committee for Conferences
- Tinkering with Technologies
- Networking with Interesting People

Typical reactions to my Netflix talks…

Typical reactions to my Netflix talks…



Typical reactions to my Netflix talks…

"You guys are crazy! Can't believe it"
– 2009

"What Netflix is doing won't work"
– 2010

Typical reactions to my Netflix talks…

"You guys are crazy! Can't believe it"
– 2009

"What Netflix is doing won't work"
– 2010

It only works for 'Unicorns' like Netflix"
– 2011

Typical reactions to my Netflix talks…

"You guys are crazy! Can't believe it"
– 2009

"What Netflix is doing won't work"
– 2010

It only works for 'Unicorns' like Netflix"
– 2011

"We'd like to do that but can't"
– 2012

Typical reactions to my Netflix talks…

"You guys are crazy! Can't believe it"
– 2009

"What Netflix is doing won't work"
– 2010

It only works for 'Unicorns' like Netflix"
– 2011

"We'd like to do that but can't"
– 2012

"We're on our way using Netflix OSS code"
– 2013

What I learned from my time at Netflix

What I learned from my time at Netflix

- *Speed wins in the marketplace*

What I learned from my time at Netflix

- *Speed wins in the marketplace*
- *Remove friction from product development*

What I learned from my time at Netflix

- *Speed wins in the marketplace*
- *Remove friction from product development*
- *High trust, low process, no hand-offs between teams*

What I learned from my time at Netflix

- *Speed wins in the marketplace*
- *Remove friction from product development*
- *High trust, low process, no hand-offs between teams*
- *Freedom and responsibility culture*

What I learned from my time at Netflix

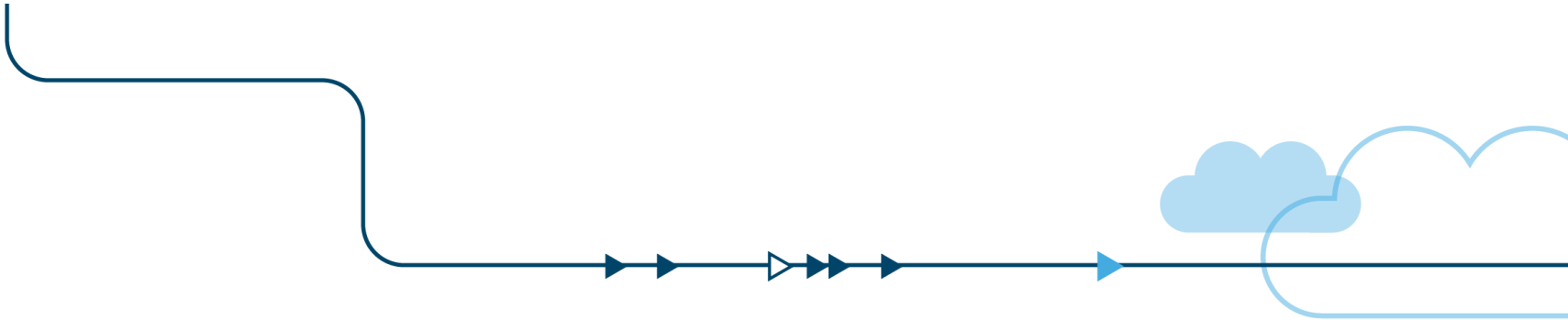
- *Speed wins in the marketplace*
- *Remove friction from product development*
- *High trust, low process, no hand-offs between teams*
- *Freedom and responsibility culture*
- *Don’t do your own undifferentiated heavy lifting*

What I learned from my time at Netflix

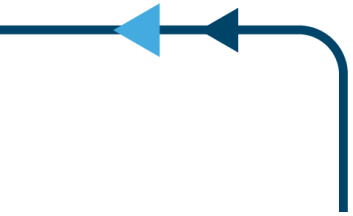
- *Speed wins in the marketplace*
- *Remove friction from product development*
- *High trust, low process, no hand-offs between teams*
- *Freedom and responsibility culture*
- *Don't do your own undifferentiated heavy lifting*
- *Use simple patterns automated by tooling*

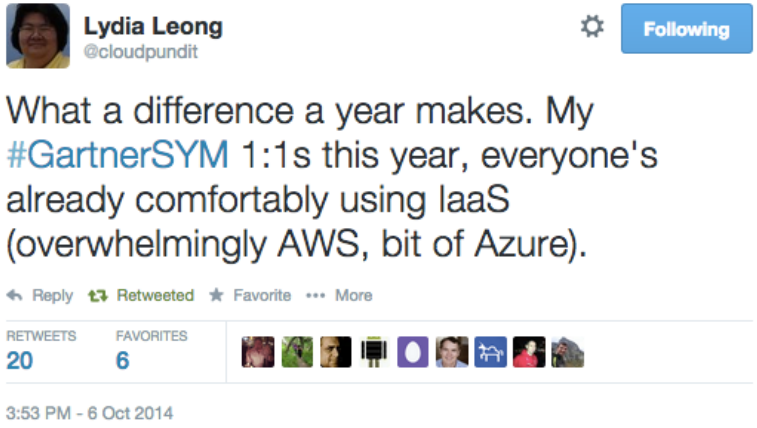
What I learned from my time at Netflix

- *Speed wins in the marketplace*
- *Remove friction from product development*
- *High trust, low process, no hand-offs between teams*
- *Freedom and responsibility culture*
- *Don't do your own undifferentiated heavy lifting*
- *Use simple patterns automated by tooling*
- *Self service cloud makes impossible things instant*

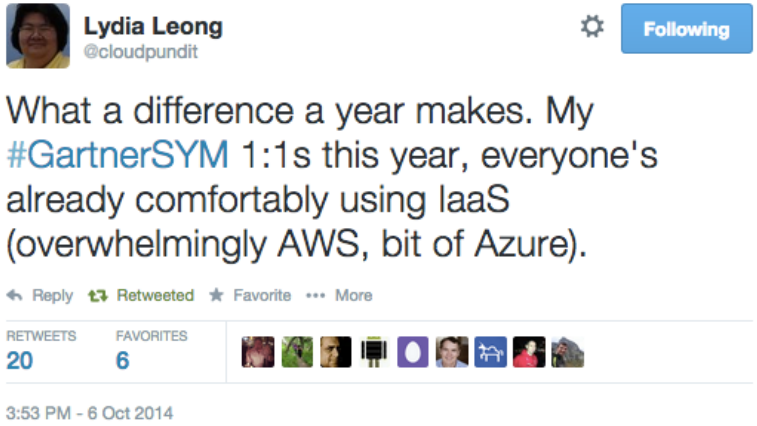


2014 was the year that Enterprises finally embraced cloud and DevOps.





2014 was the year that Enterprises finally embraced cloud and DevOps.



2014 was the year that Enterprises finally embraced cloud and DevOps.

Lydia Leong
@cloudpundit

Following

What a difference a year makes. My #GartnerSYM 1:1s this year, everyone's already comfortably using IaaS (overwhelmingly AWS, bit of Azure).

Reply · Retweeted · Favorite · More

RETWEETS 20 · FAVORITES 6

3:53 PM - 6 Oct 2014

2014 was the year that Enterprises finally embraced cloud and DevOps.

adrian cockcroft @adrianco · Oct 22

RT @devopscouts: Nordstrom went from optimizing for IT cost to optimizing for delivery speed @ladyhock #DevOps #DOES14 < this is key point

20 · 12

adrian cockcroft retweeted

Steve Brodie @stbrodie · Oct 22

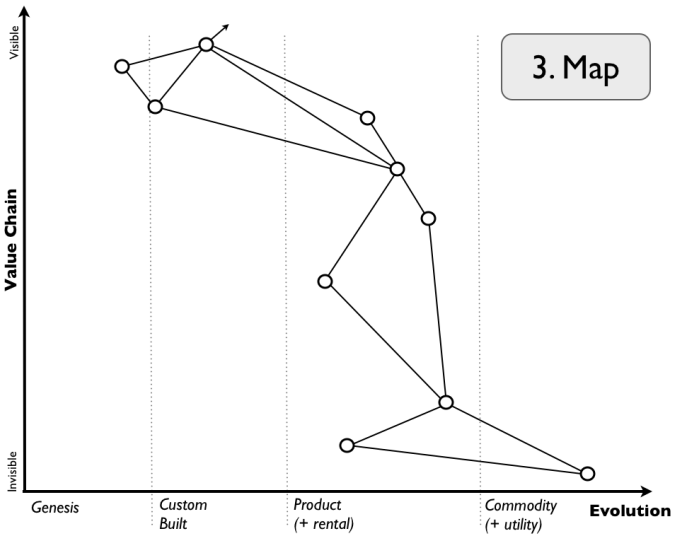
This may be the very best conference I have ever been to, @glennodonnell VP @Forrester on #DOES14

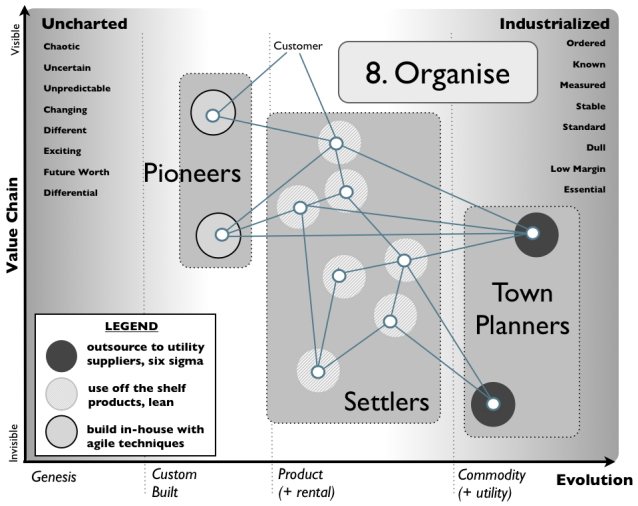
7 · 4

View more photos and videos

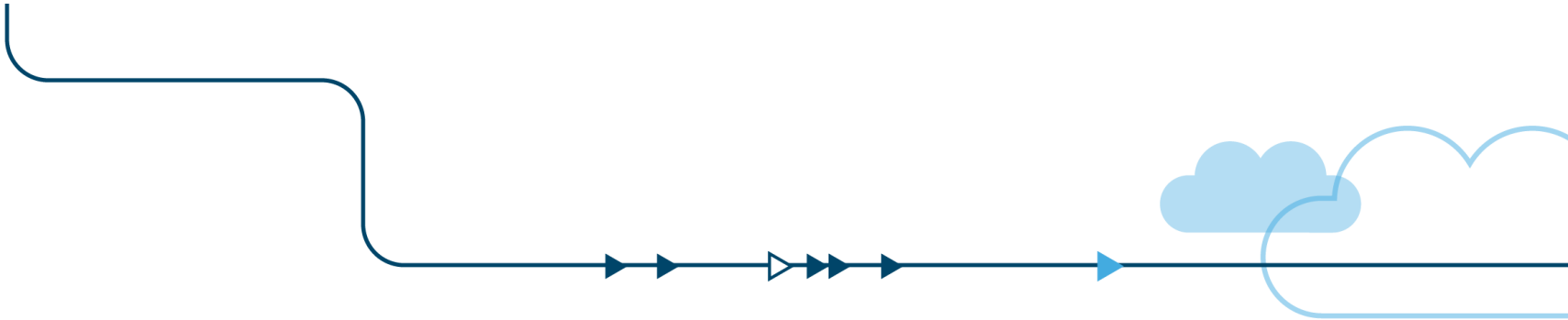
What separates incumbents from disruptors?

Strategy Mapping



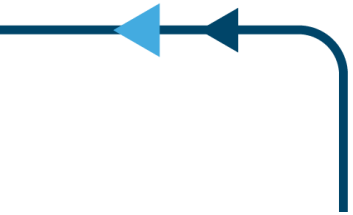


Simon Wardley http://blog.gardeviance.org/2014/11/how-to-get-to-strategy-in-ten-steps.html
Related tools and training http://www.wardleymaps.com/

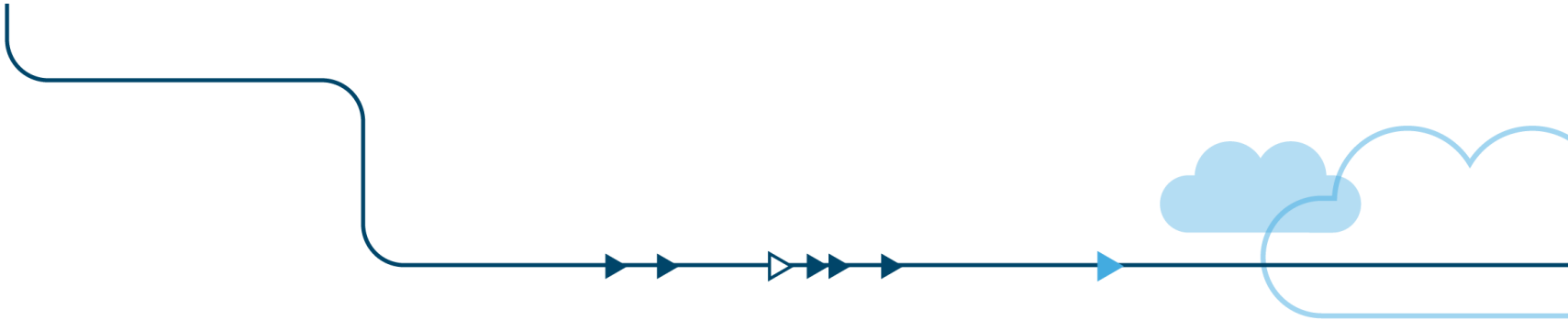


“It isn't what we don't know that gives us trouble, it's what we know that ain't so.”

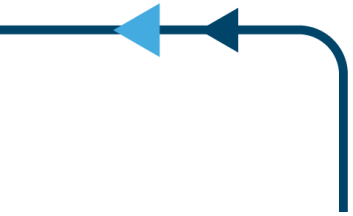
Will Rogers



Assumptions



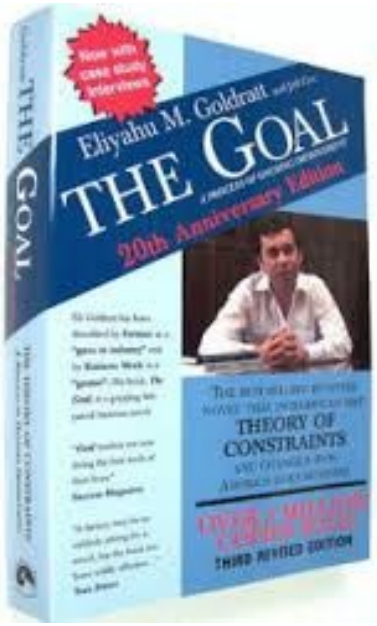
Optimizations



Assumption: Process prevents problems

Organizations build up slow complex "Scar tissue" processes

"This is the IT swamp draining manual for anyone who is neck deep in alligators."

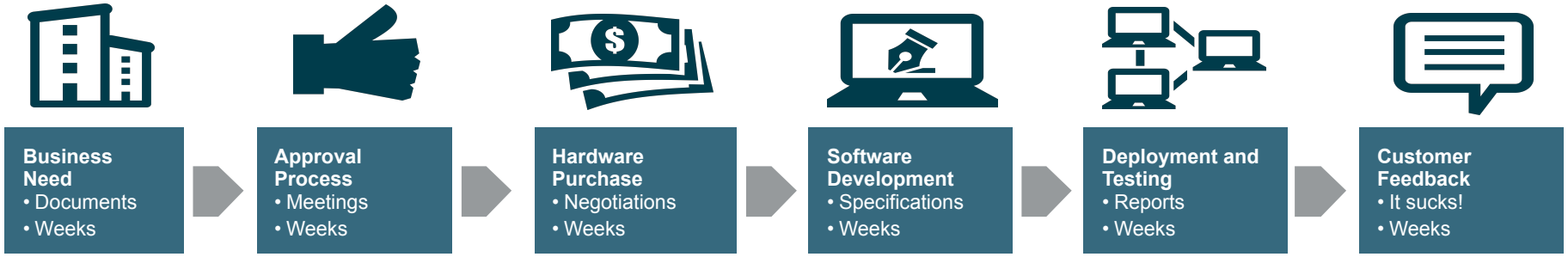


1984

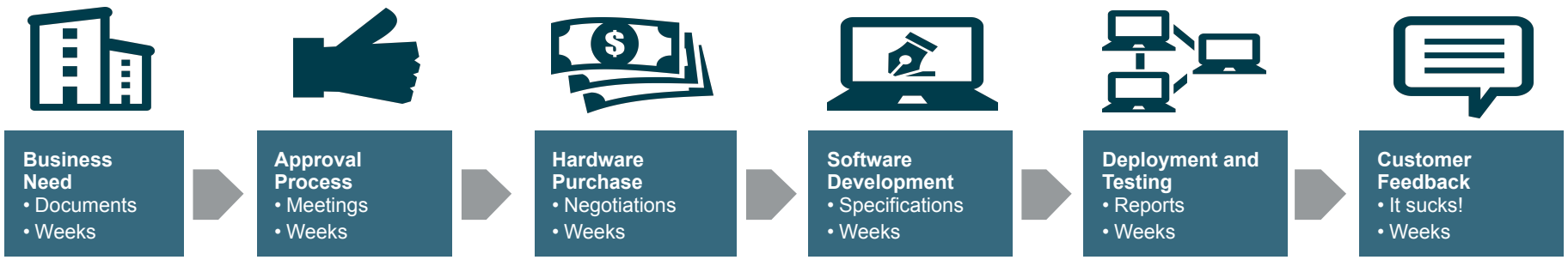
2014

Product Development Processes

Waterfall Product Development

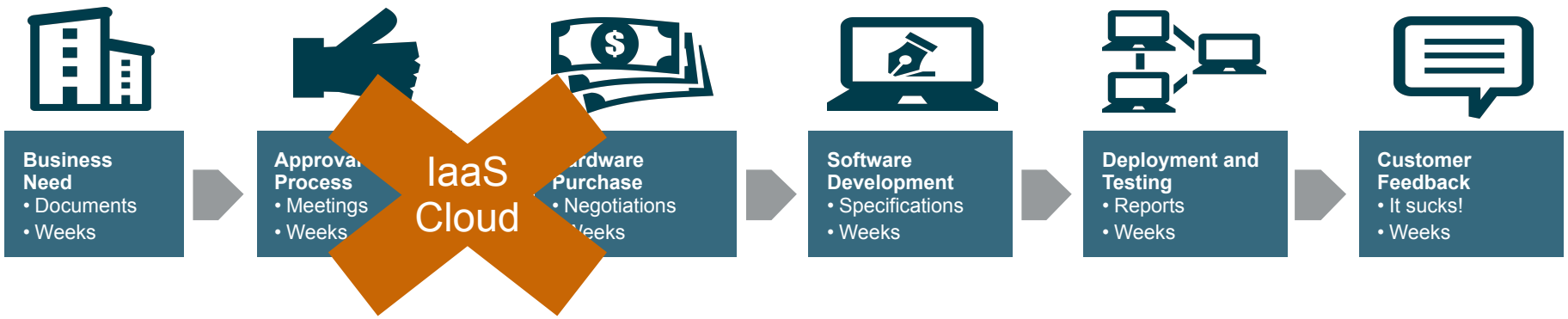


Waterfall Product Development



Hardware provisioning is undifferentiated heavy lifting – replace it with IaaS

Waterfall Product Development

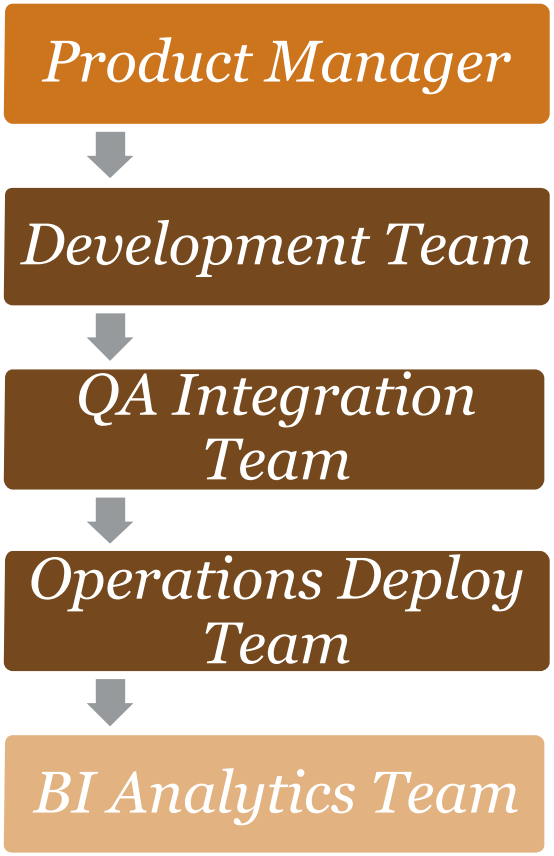


Hardware provisioning is undifferentiated heavy lifting – replace it with IaaS

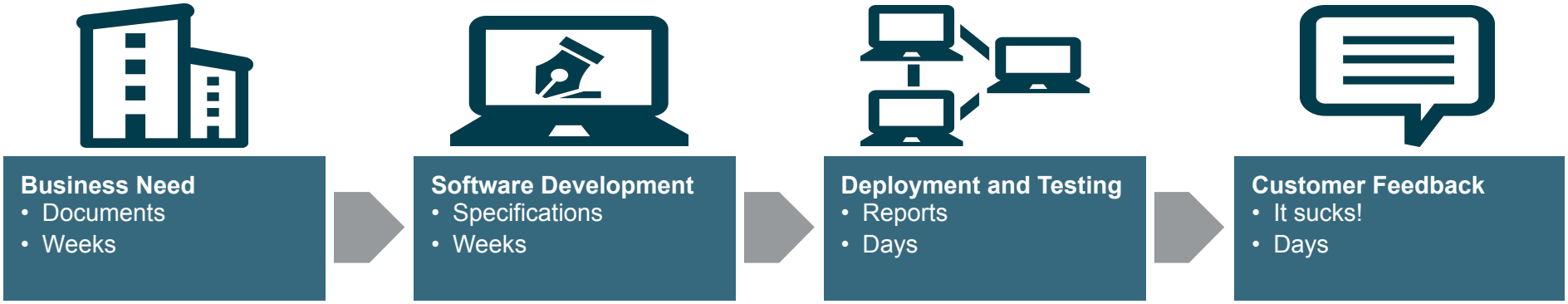
Waterfall Product Development

Hardware provisioning is undifferentiated heavy lifting – replace it with IaaS

Process Hand-Off Steps for Agile Development on IaaS



IaaS Agile Product Development



IaaS Agile Product Development

amazon web services™ | Windows Azure | SOFTLAYER an IBM Company | openstack CLOUD SOFTWARE | EUCALYPTUS | etc…

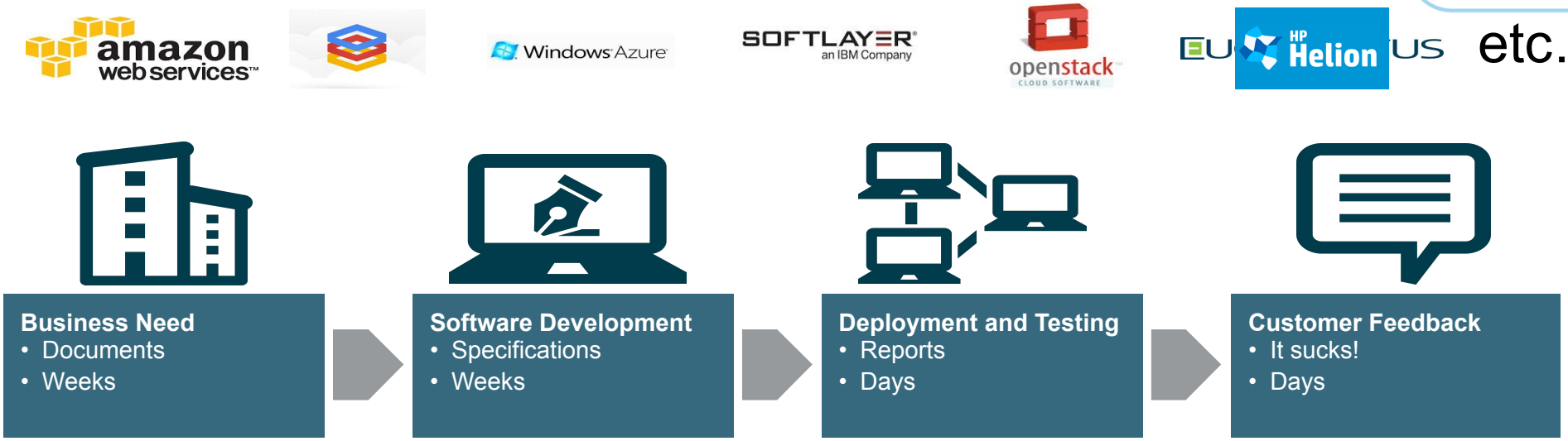
Business Need
- Documents
- Weeks

Software Development
- Specifications
- Weeks

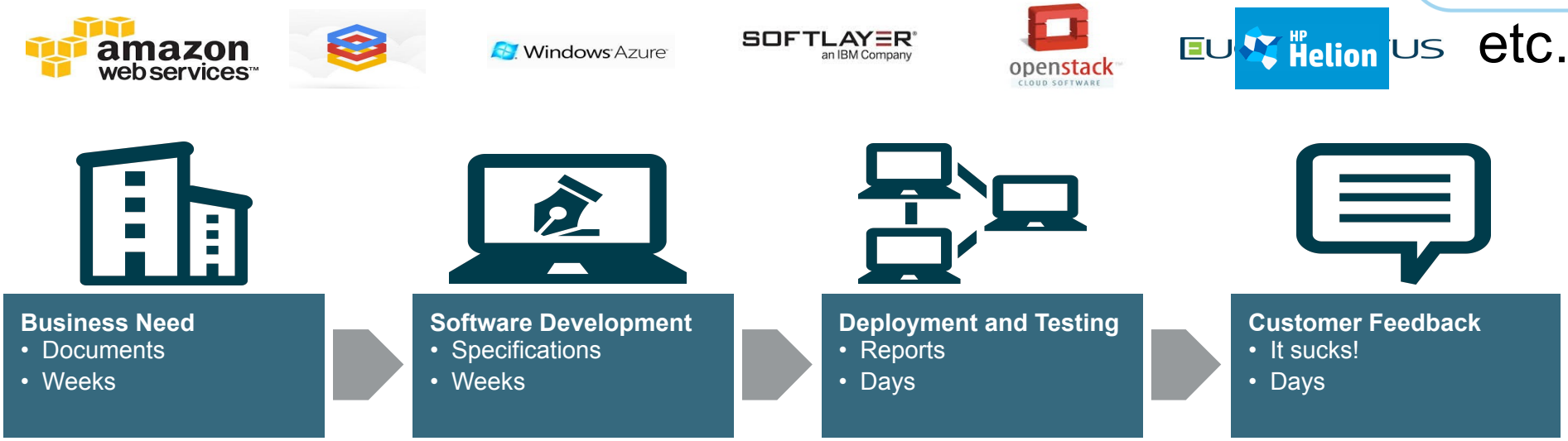
Deployment and Testing
- Reports
- Days

Customer Feedback
- It sucks!
- Days

IaaS Agile Product Development

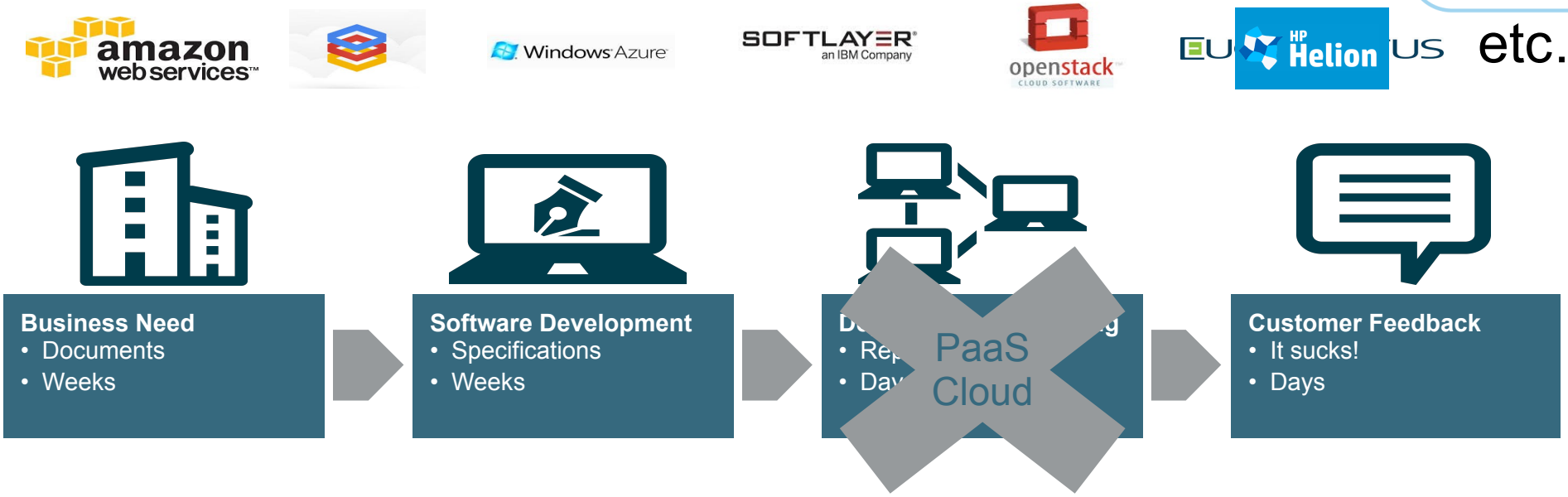


IaaS Agile Product Development



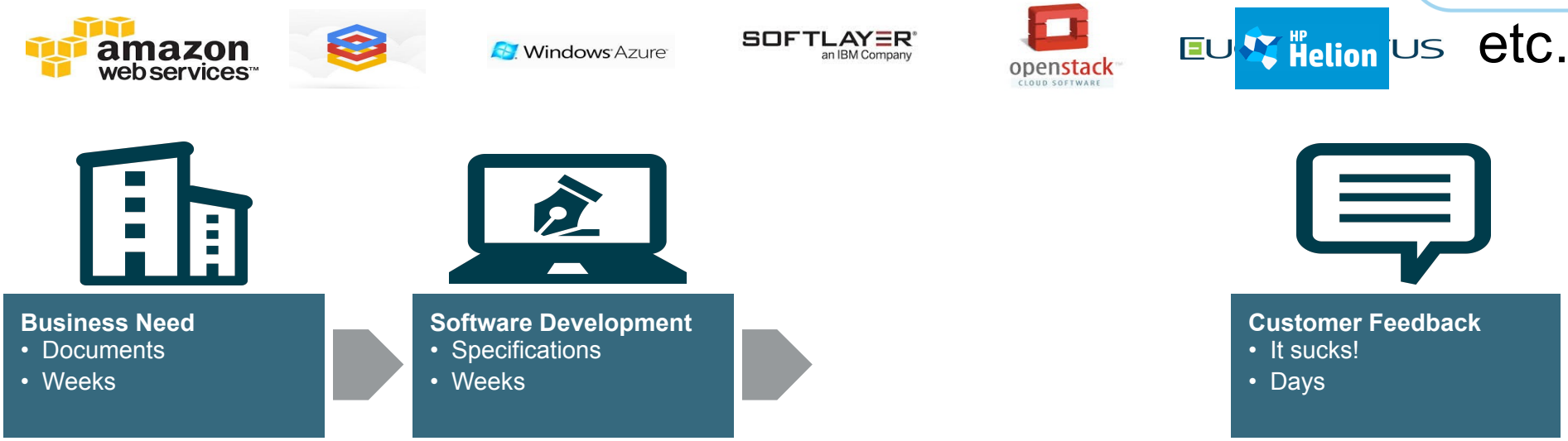
Software provisioning is undifferentiated heavy lifting – replace it with PaaS

IaaS Agile Product Development



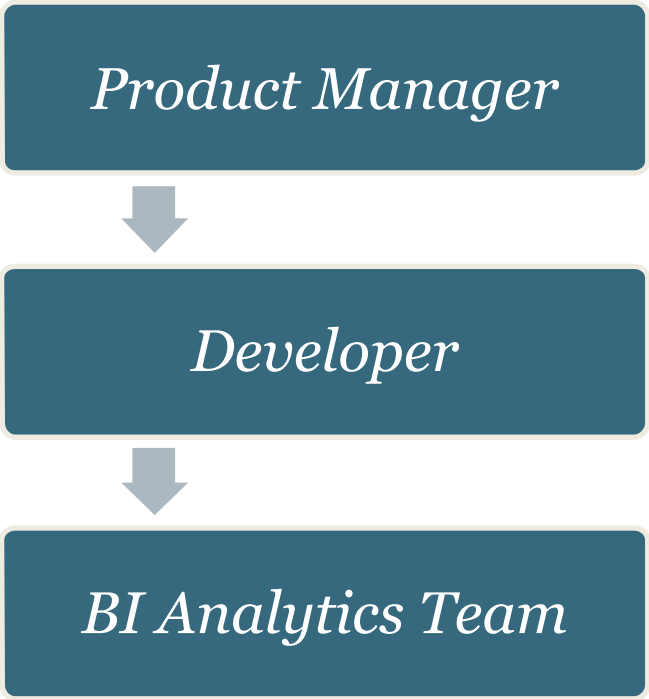
Software provisioning is undifferentiated heavy lifting – replace it with PaaS

IaaS Agile Product Development



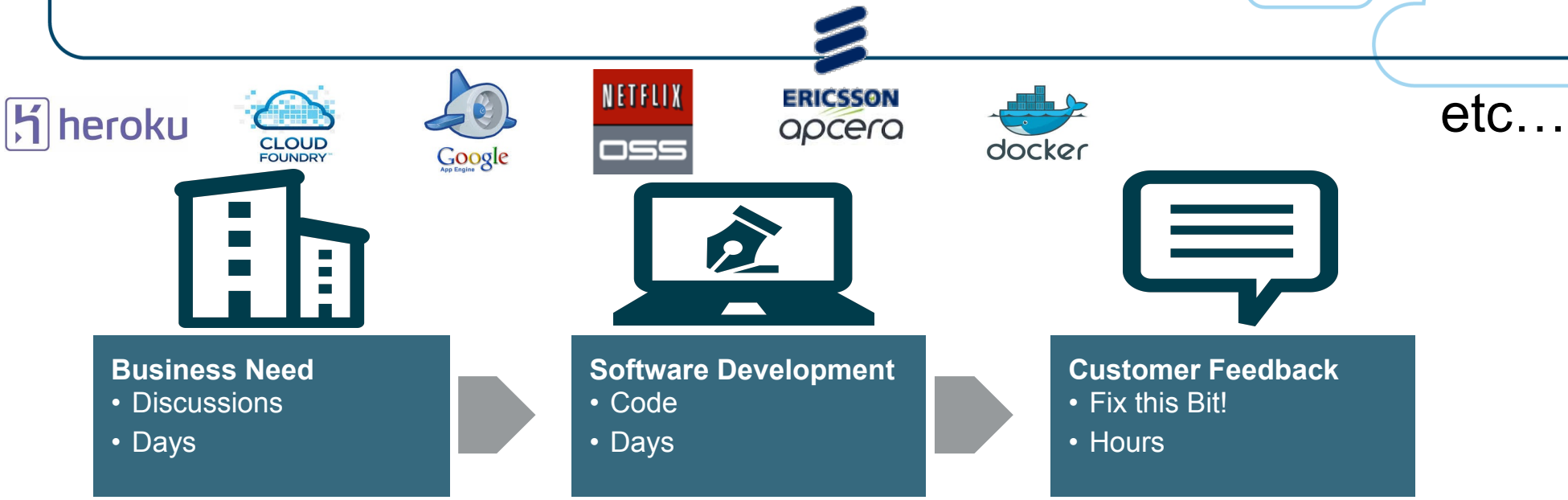
Software provisioning is undifferentiated heavy lifting – replace it with PaaS

Process for Continuous Delivery of Features on PaaS

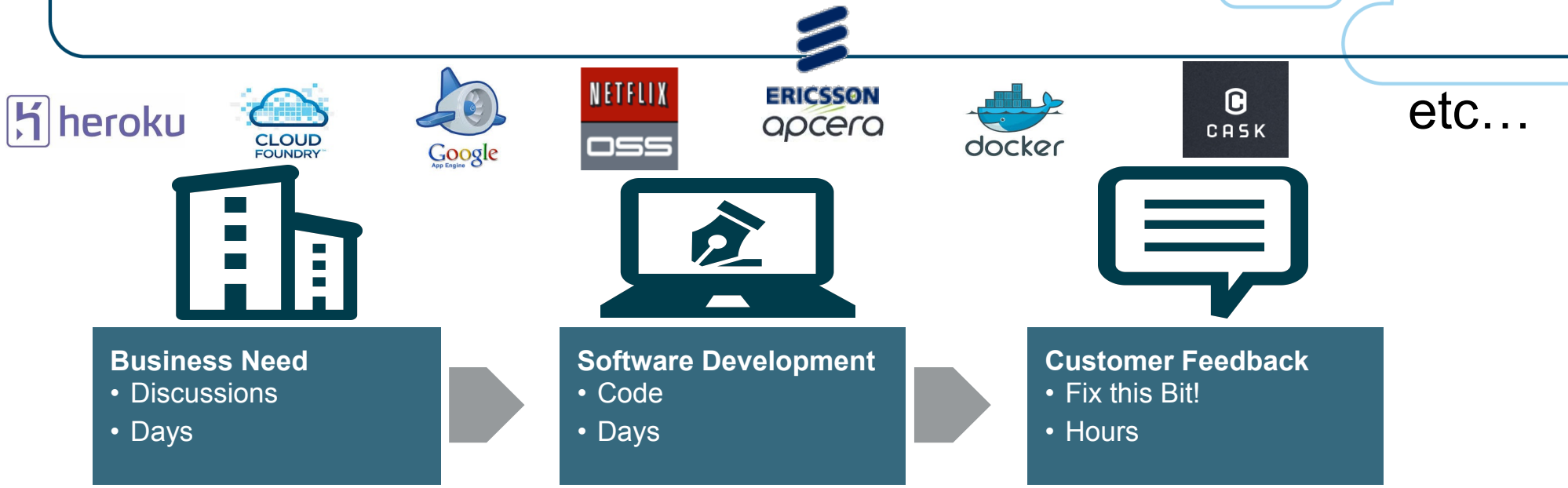


PaaS CD Feature Development

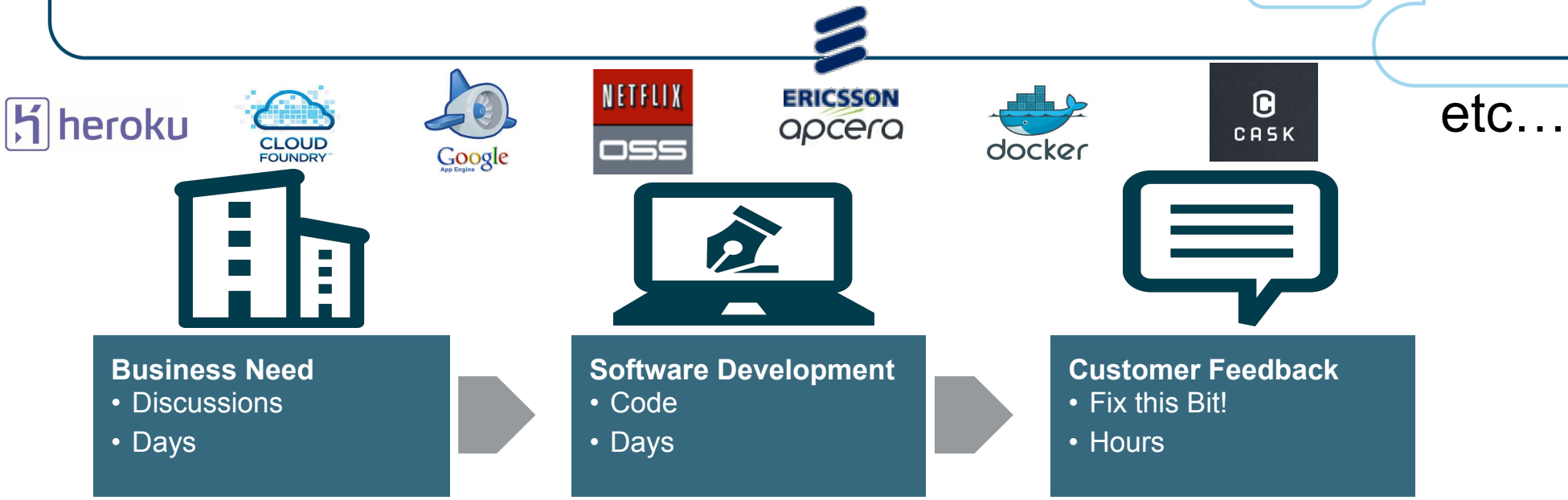
PaaS CD Feature Development



PaaS CD Feature Development

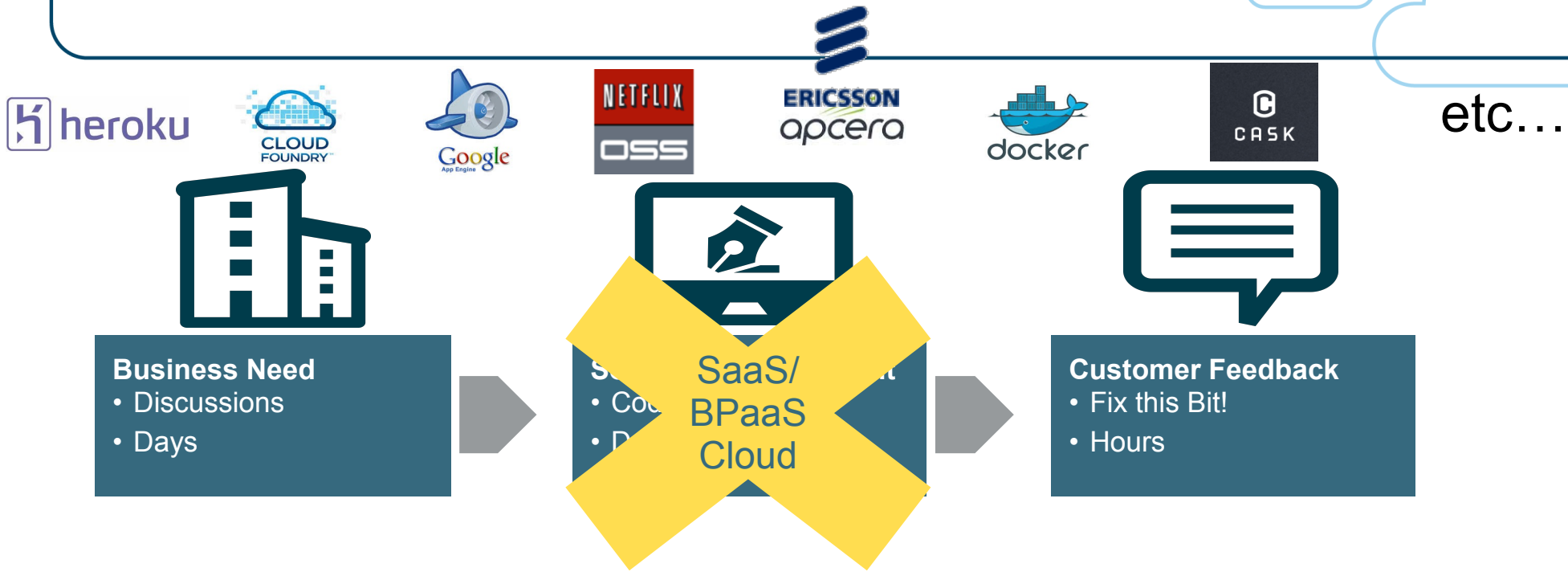


PaaS CD Feature Development



Building your own business apps is undifferentiated heavy lifting – use SaaS

PaaS CD Feature Development



Building your own business apps is undifferentiated heavy lifting – use SaaS

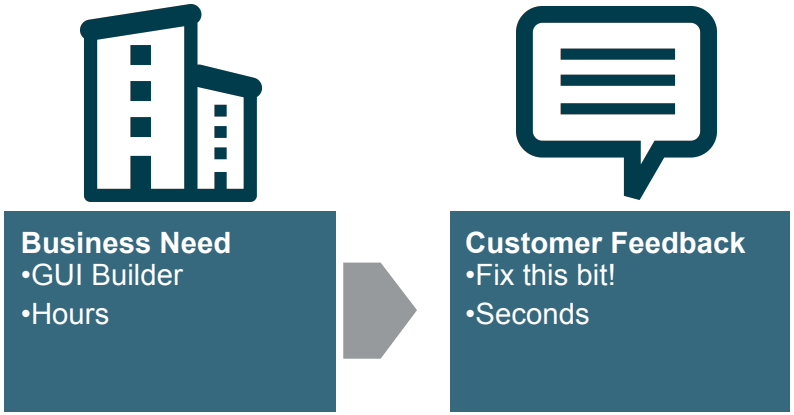
PaaS CD Feature Development

Business Need
- Discussions
- Days

Customer Feedback
- Fix this Bit!
- Hours

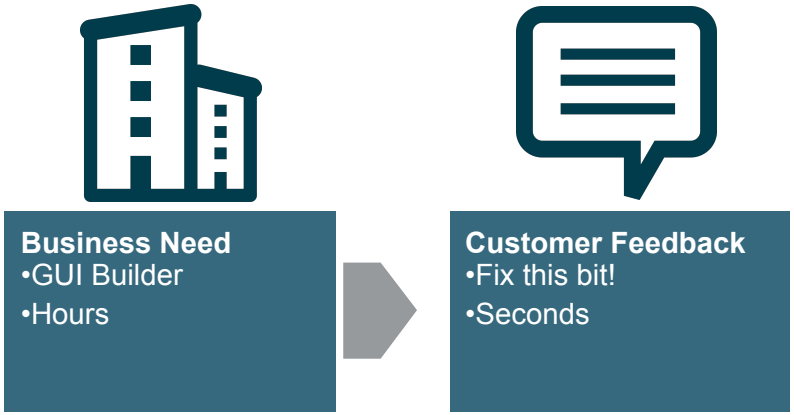
Building your own business apps is undifferentiated heavy lifting – use SaaS

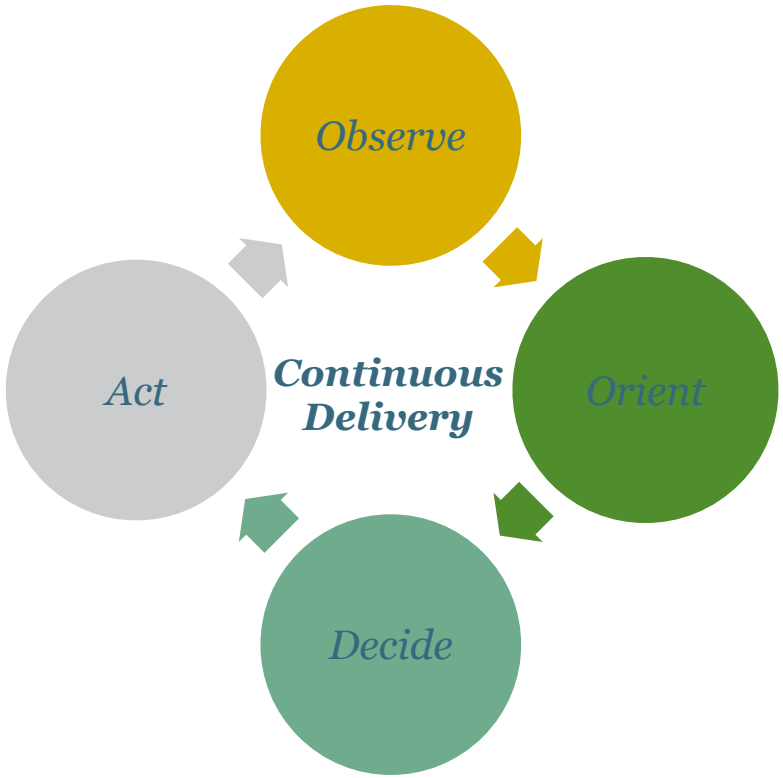
SaaS Based Business Application Development

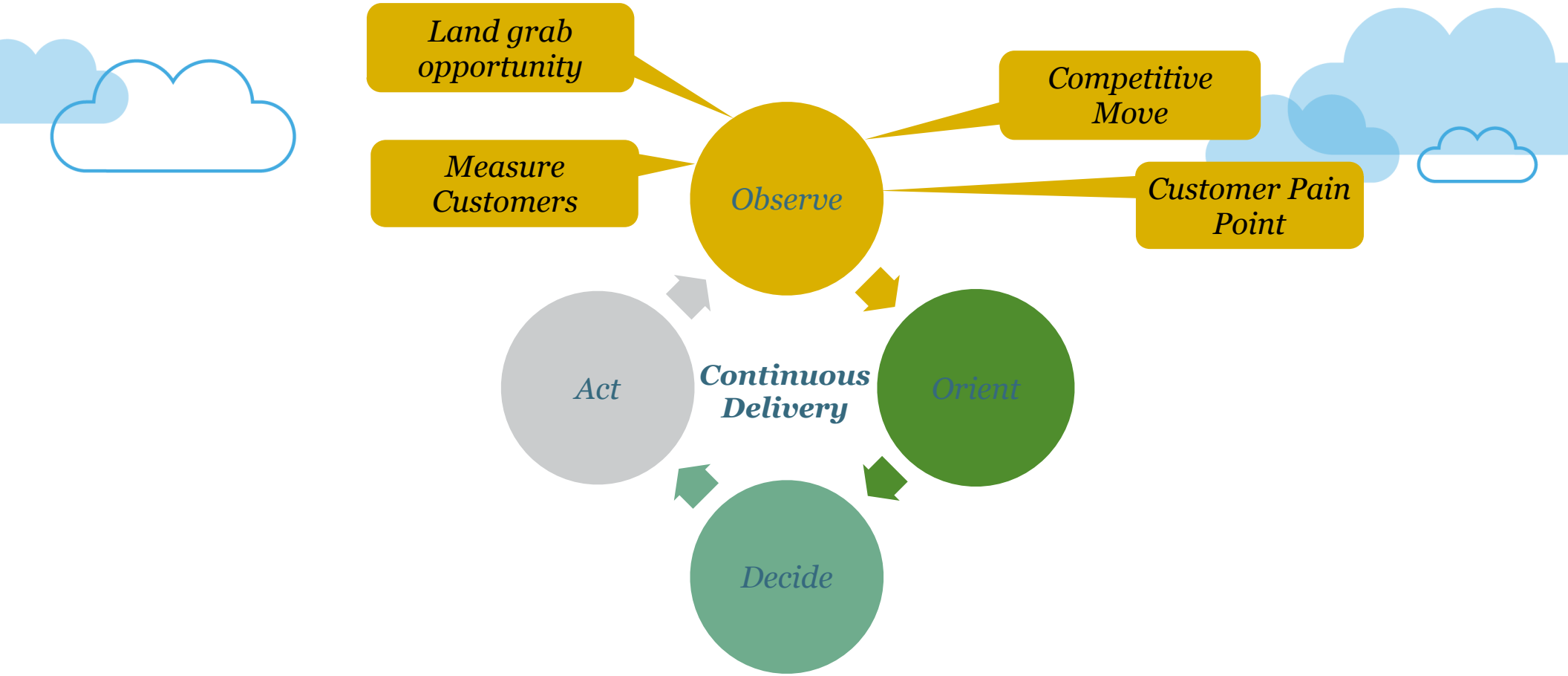


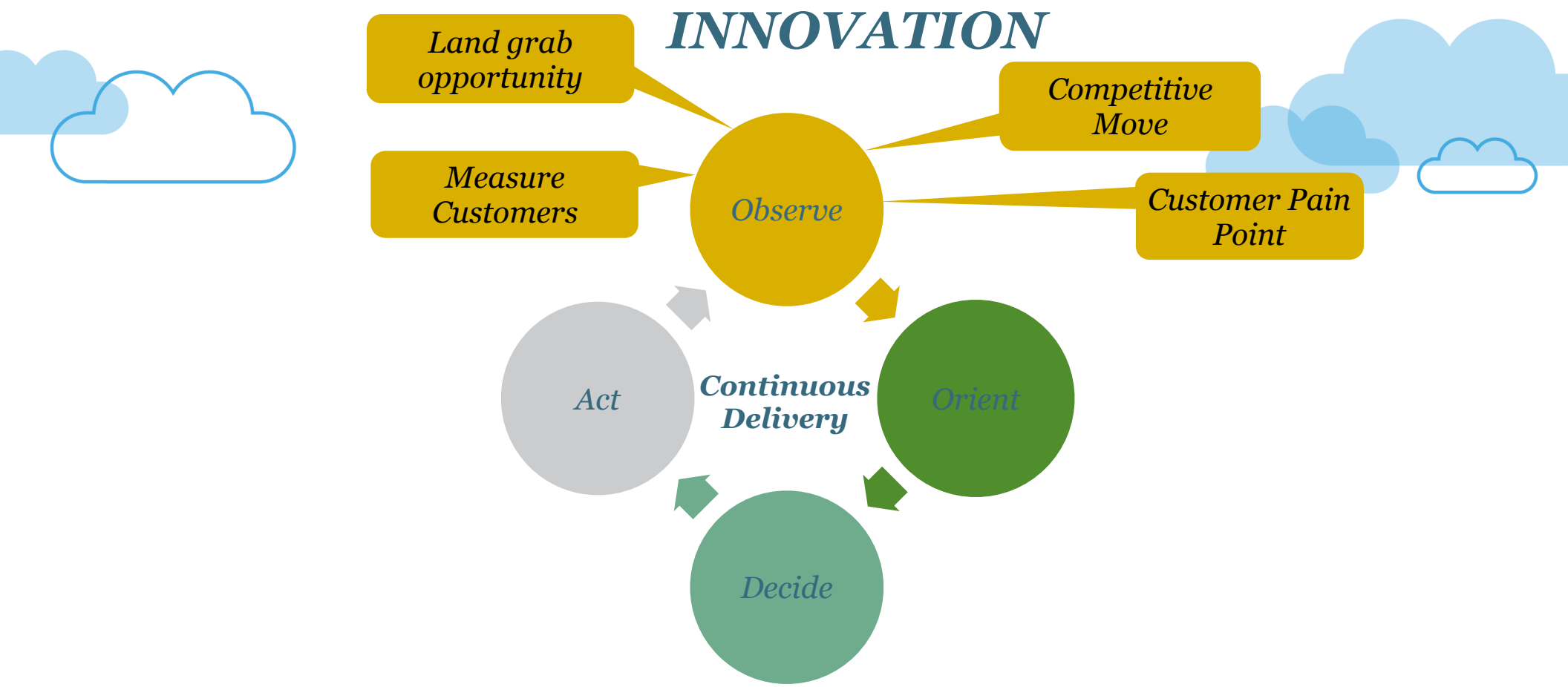
SaaS Based Business Application Development

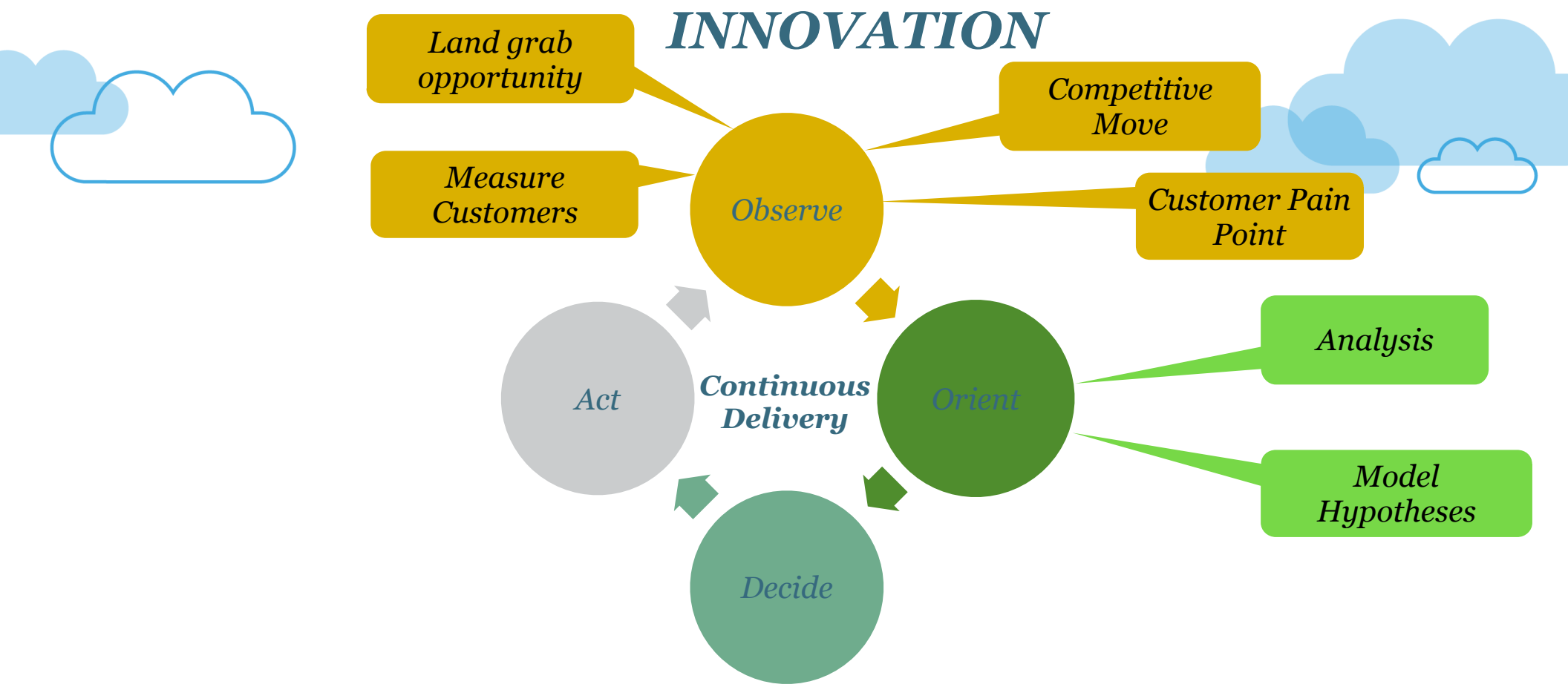
and thousands more…

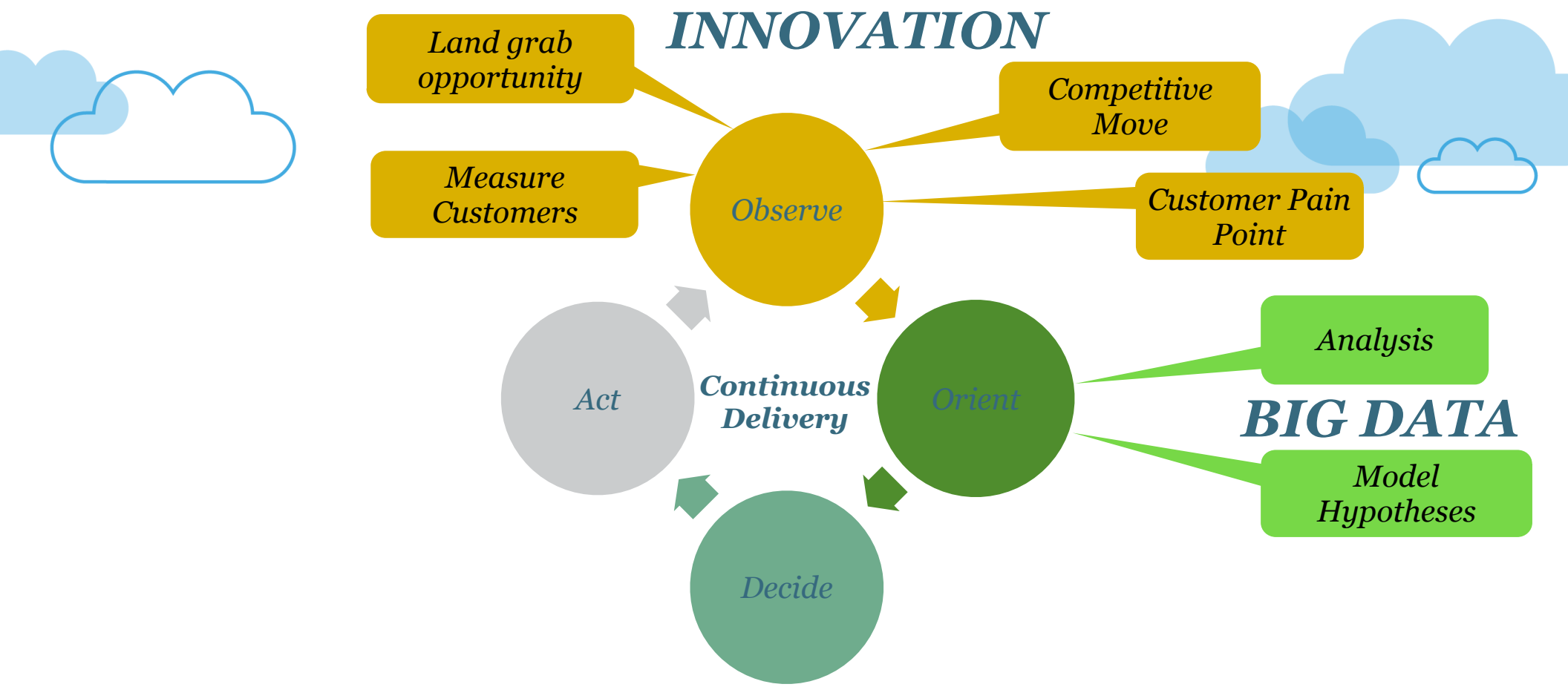


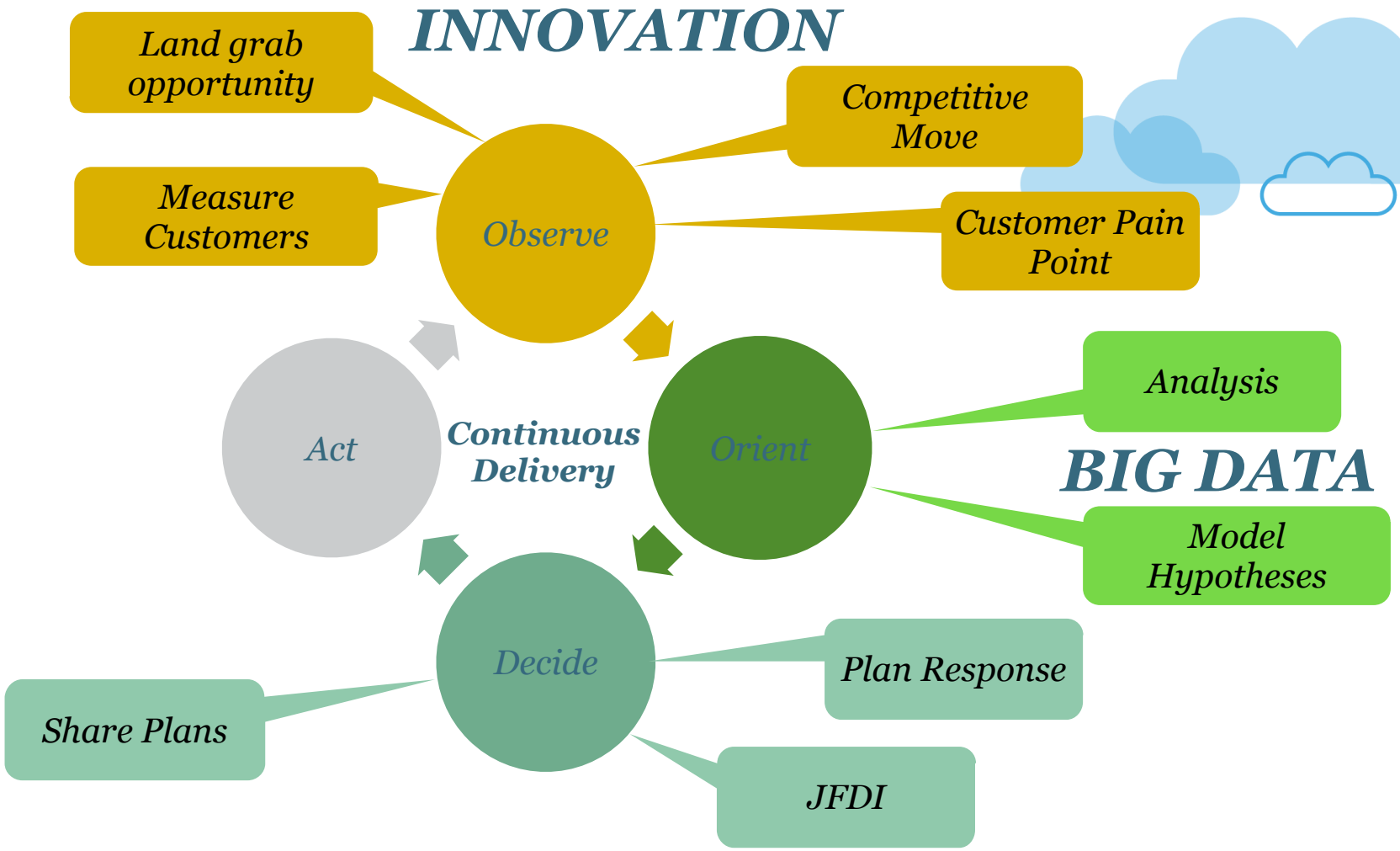

Observe
Orient
Decide
Act
Continuous Delivery

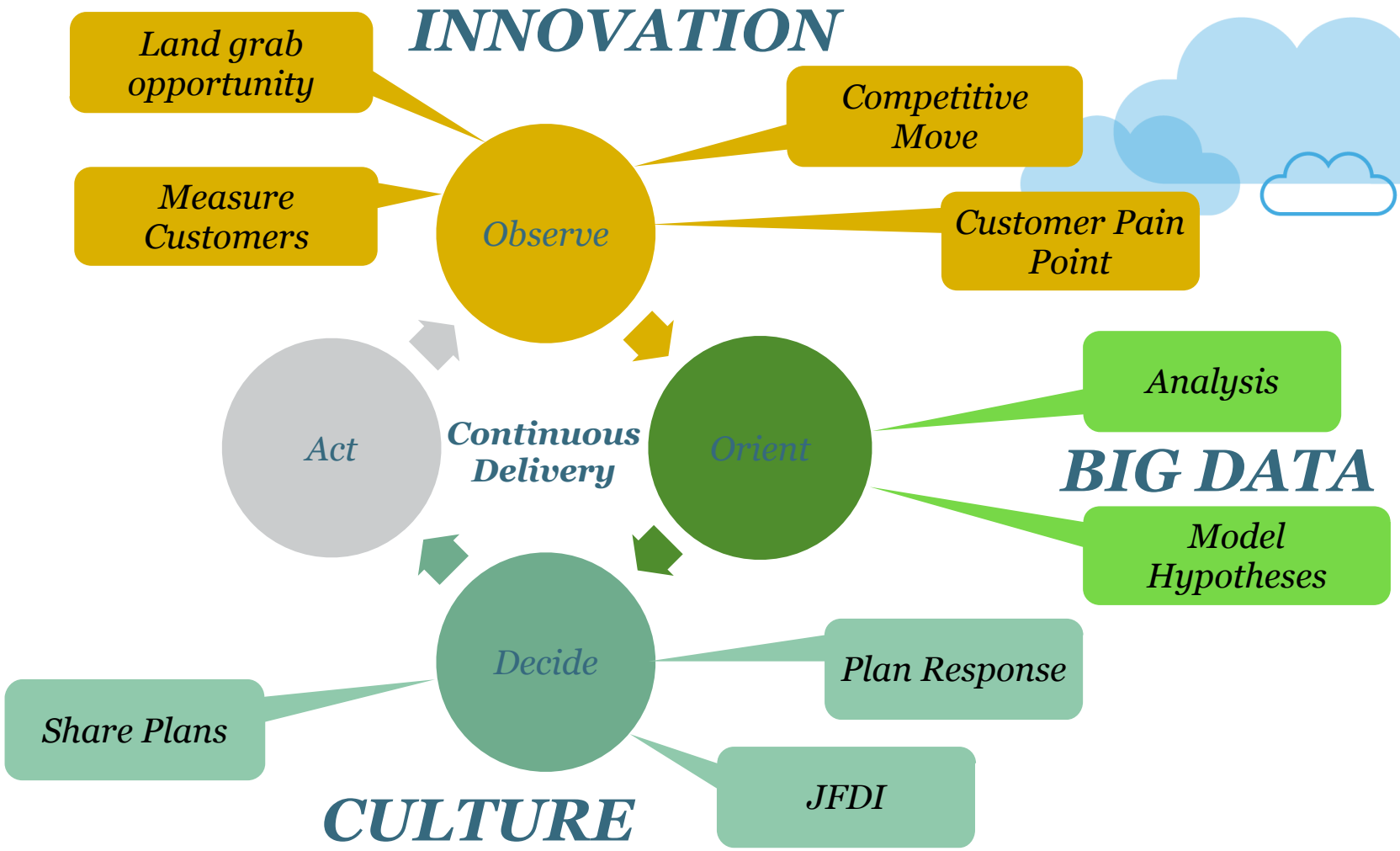

Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Act
Continuous Delivery
Orient
Decide

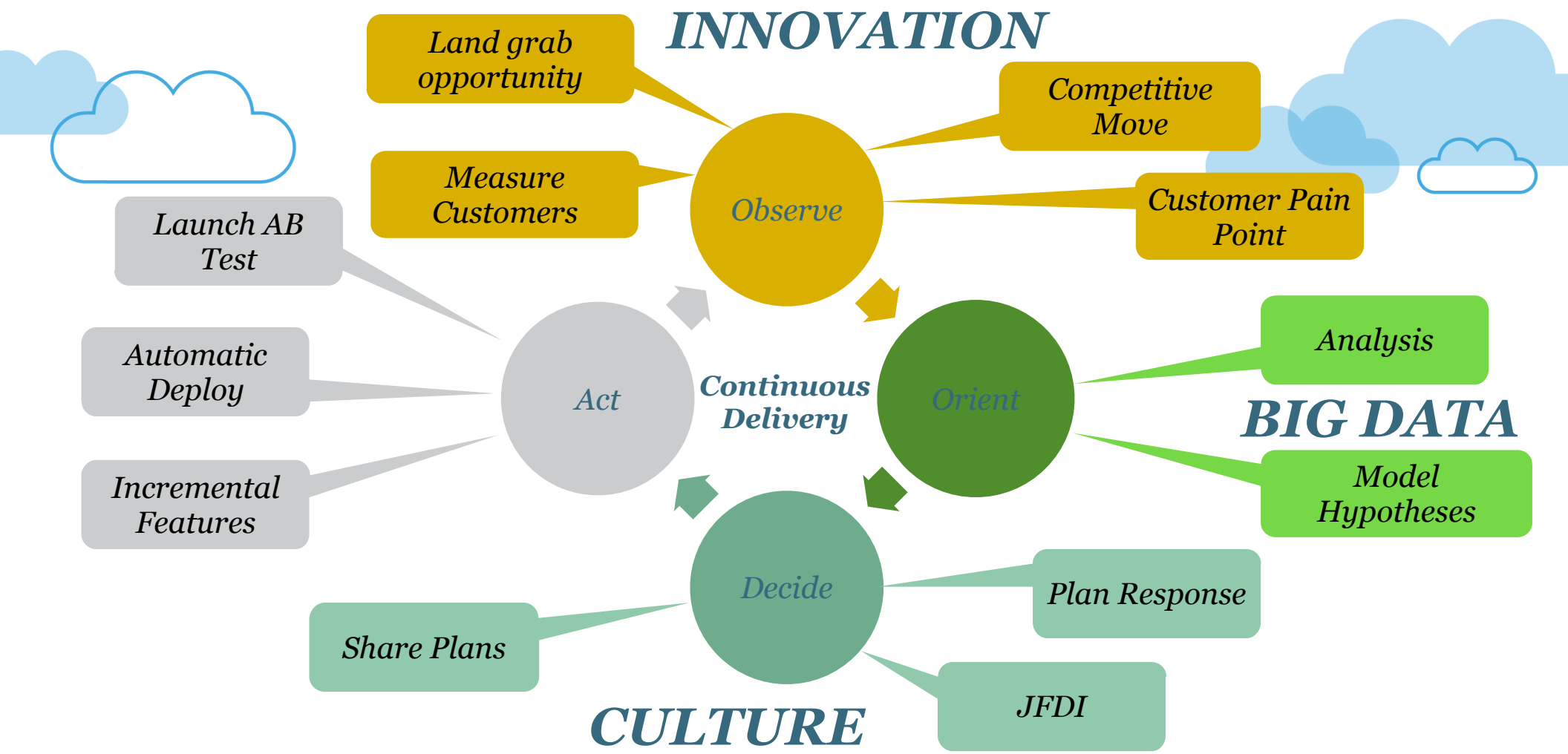

INNOVATION
Land grab opportunity
Measure Customers
Competitive Move
Customer Pain Point
Observe
Orient
Decide
Act
Continuous Delivery

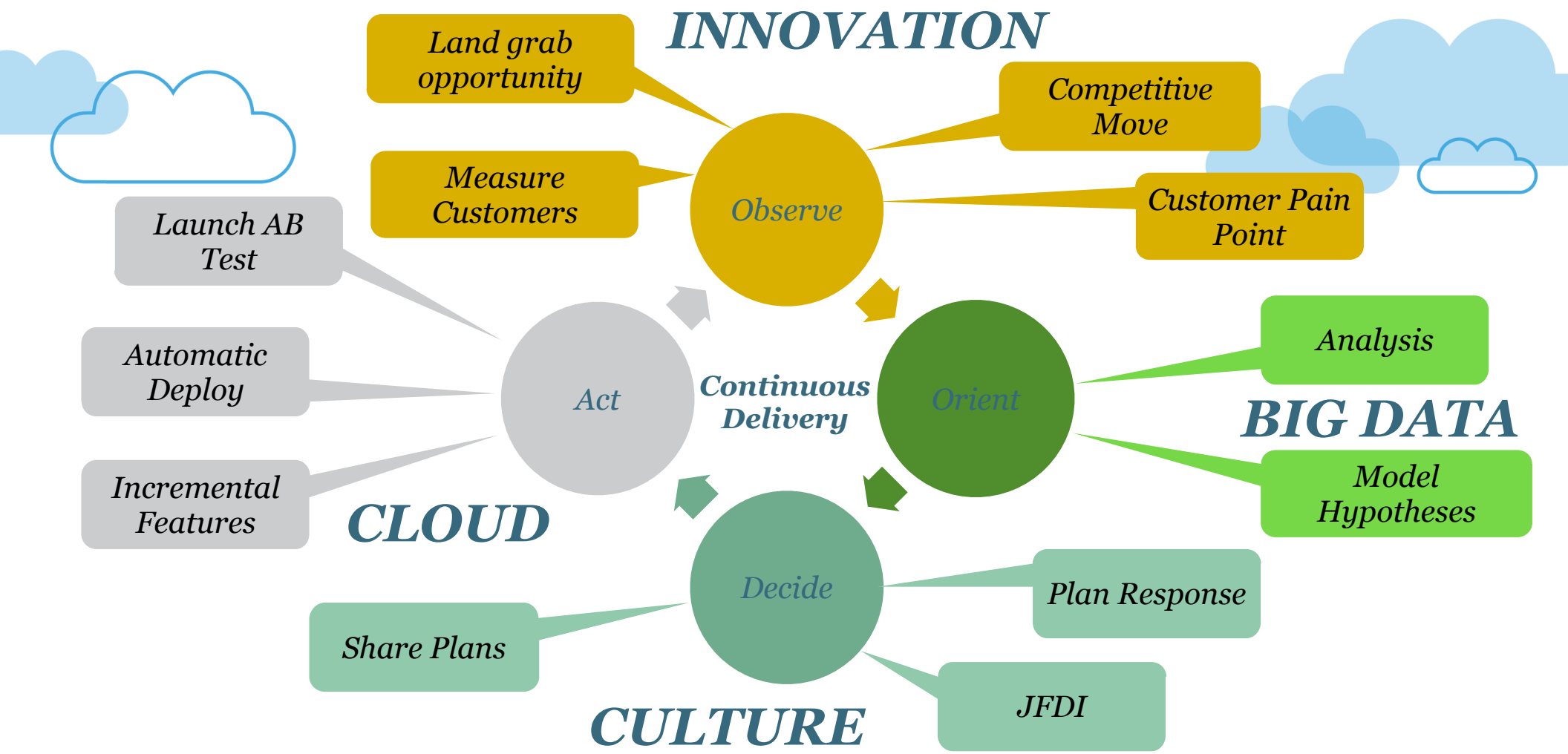

INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Act
Continuous Delivery
Orient
Analysis
Model Hypotheses
Decide

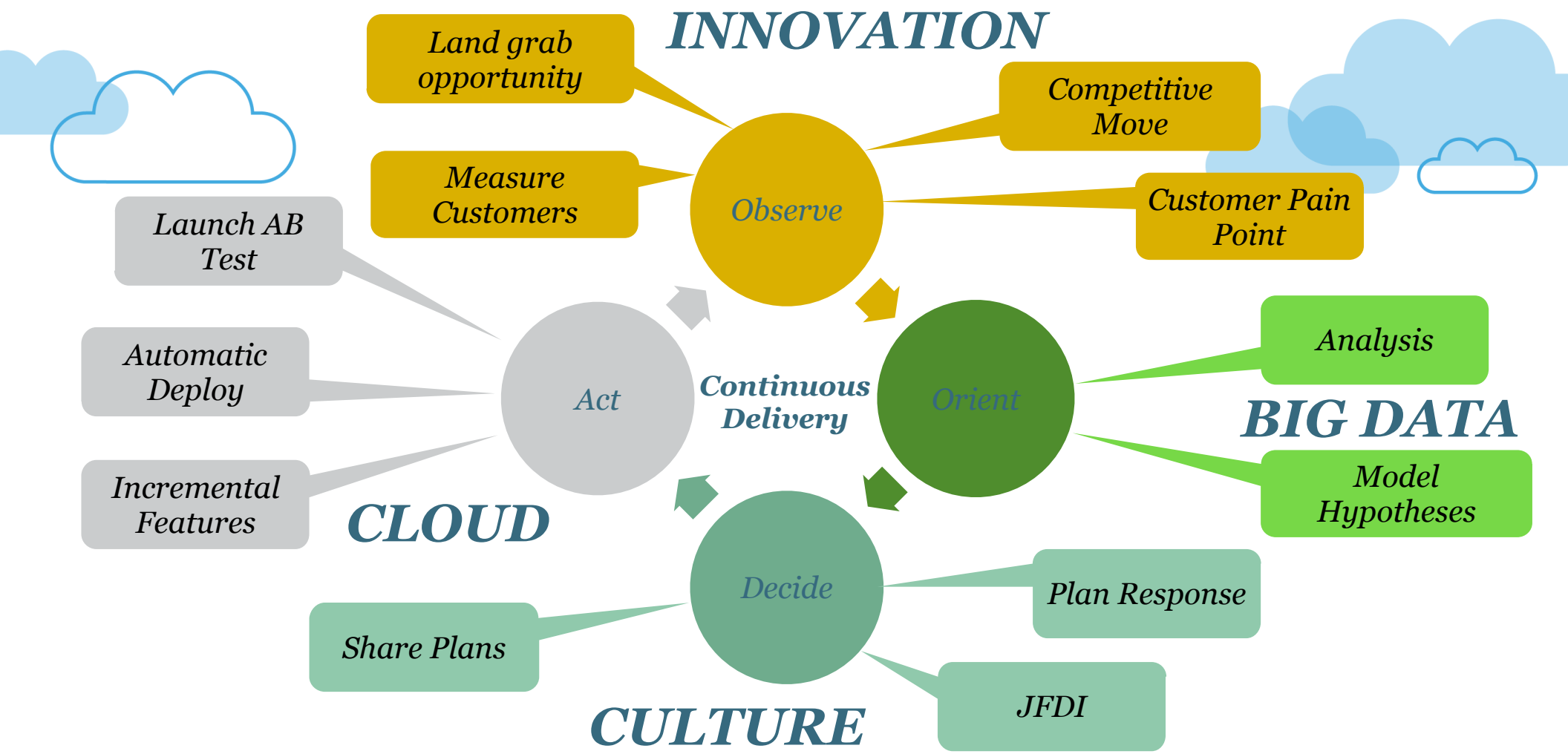

INNOVATION
Land grab opportunity
Measure Customers
Competitive Move
Customer Pain Point
Observe
Orient
Decide
Act
Continuous Delivery
Analysis
BIG DATA
Model Hypotheses

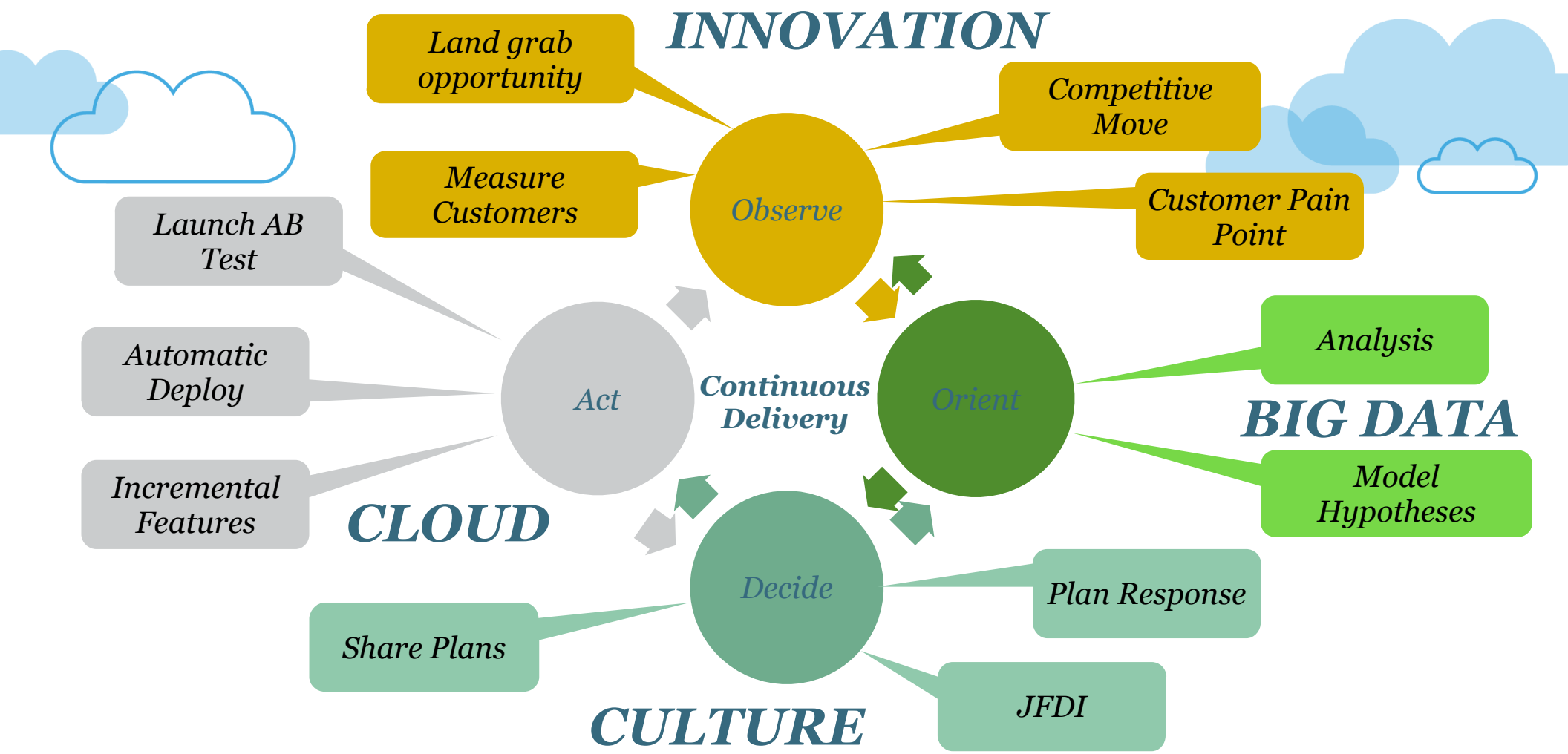

INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Act
Continuous Delivery
Orient
Analysis
BIG DATA
Model Hypotheses
Decide
Plan Response
Share Plans
JFDI


INNOVATION
Land grab opportunity
Measure Customers
Competitive Move
Customer Pain Point
Observe
Orient
Decide
Act
Continuous Delivery
Analysis
BIG DATA
Model Hypotheses
Plan Response
JFDI
Share Plans
CULTURE


INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Launch AB Test
Automatic Deploy
Incremental Features
Act
Continuous Delivery
Orient
Analysis
BIG DATA
Model Hypotheses
Decide
Plan Response
Share Plans
JFDI
CULTURE

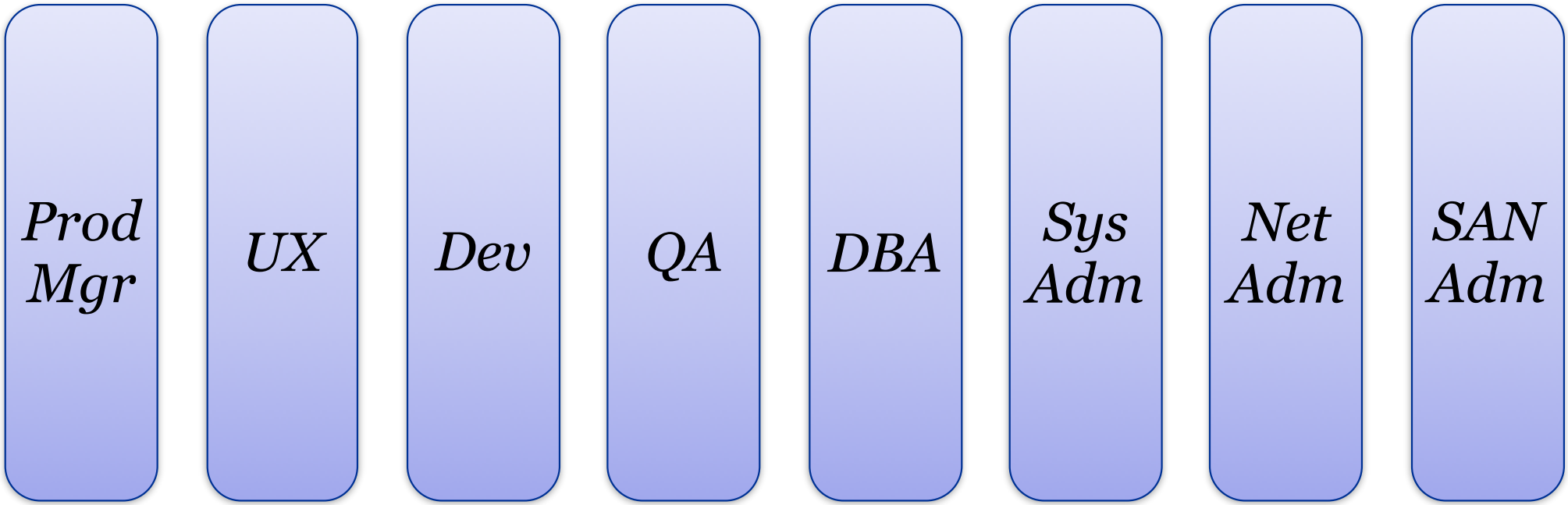

INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Launch AB Test
Automatic Deploy
Incremental Features
Act
CLOUD
Continuous Delivery
Orient
Analysis
BIG DATA
Model Hypotheses
Decide
Share Plans
Plan Response
JFDI
CULTURE


INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Launch AB Test
Automatic Deploy
Incremental Features
Act
CLOUD
Continuous Delivery
Orient
Analysis
BIG DATA
Model Hypotheses
Decide
Share Plans
Plan Response
JFDI
CULTURE

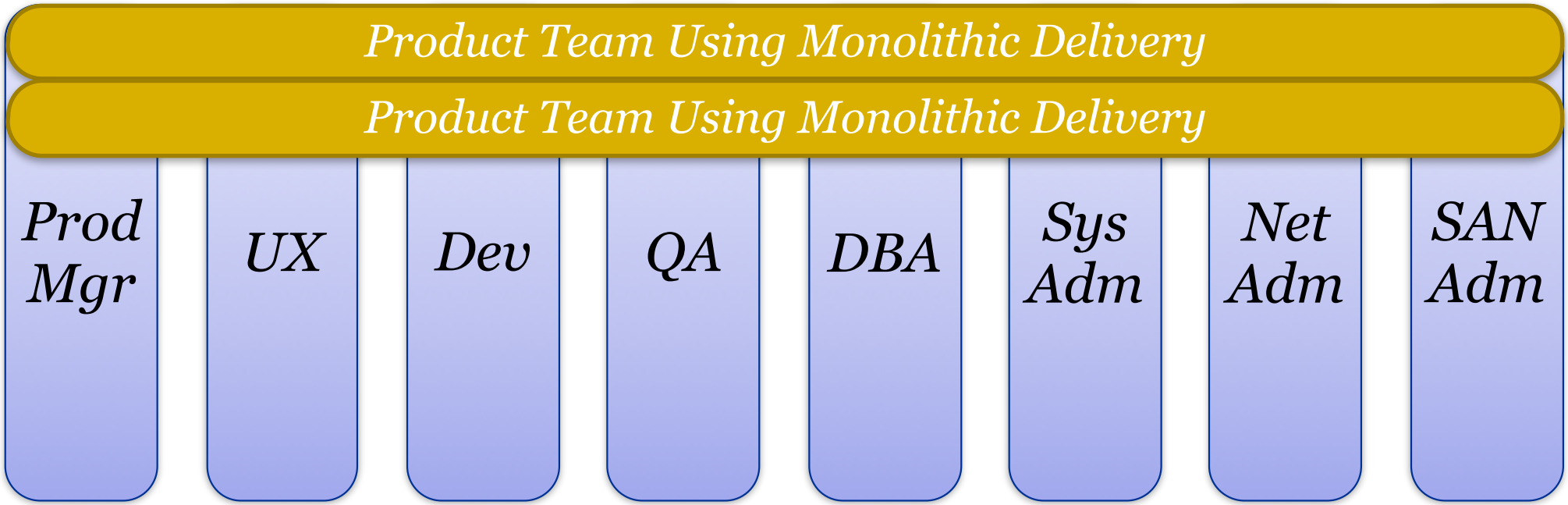

INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Launch AB Test
Automatic Deploy
Incremental Features
Act
CLOUD
Continuous Delivery
Orient
Analysis
BIG DATA
Model Hypotheses
Decide
Share Plans
Plan Response
JFDI
CULTURE

Breaking Down the SILOs

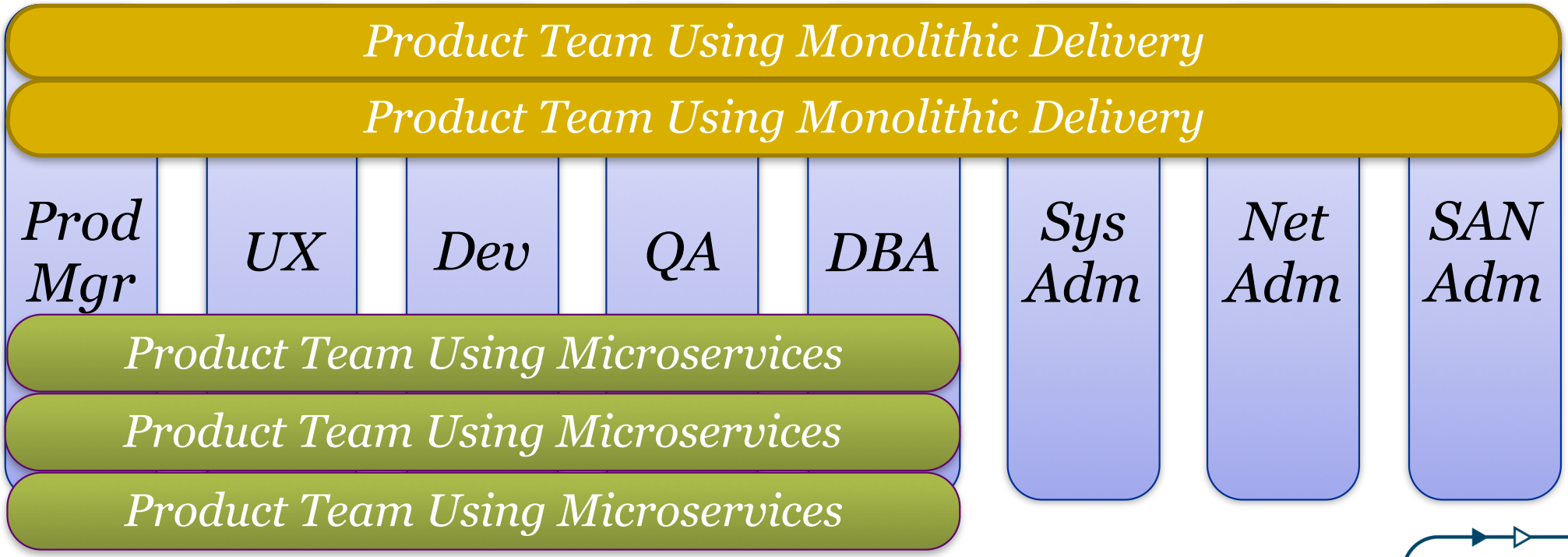
Breaking Down the SILOs



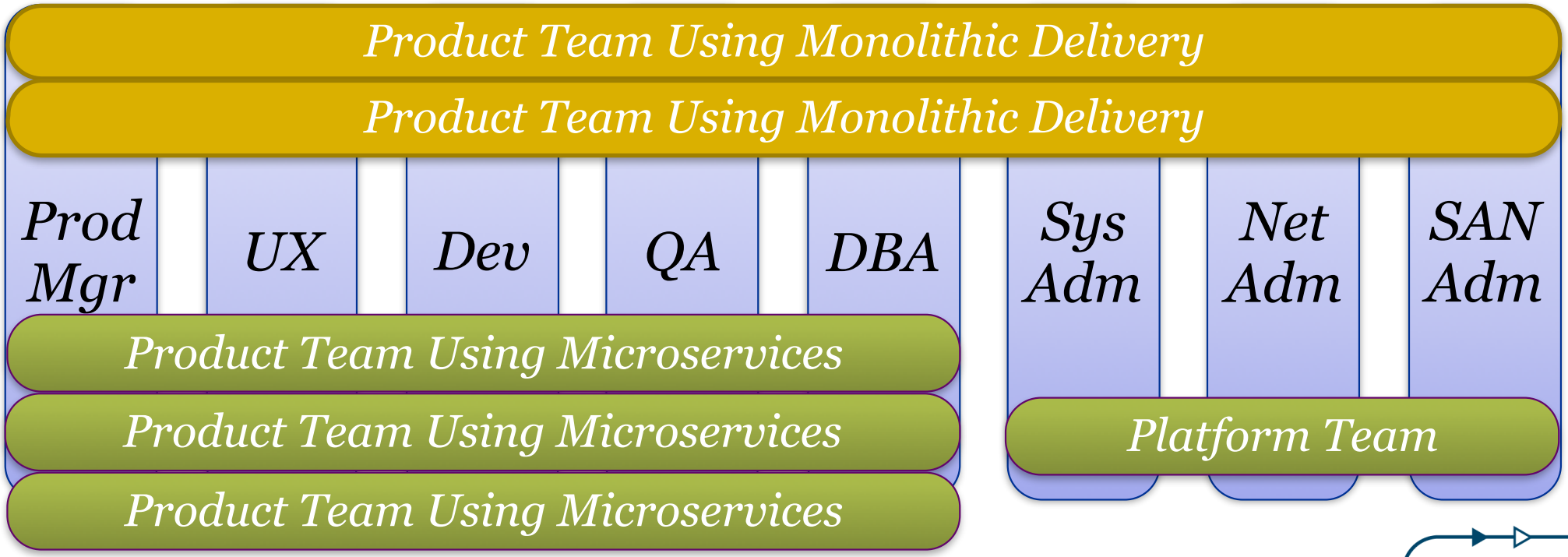
Breaking Down the SILOs



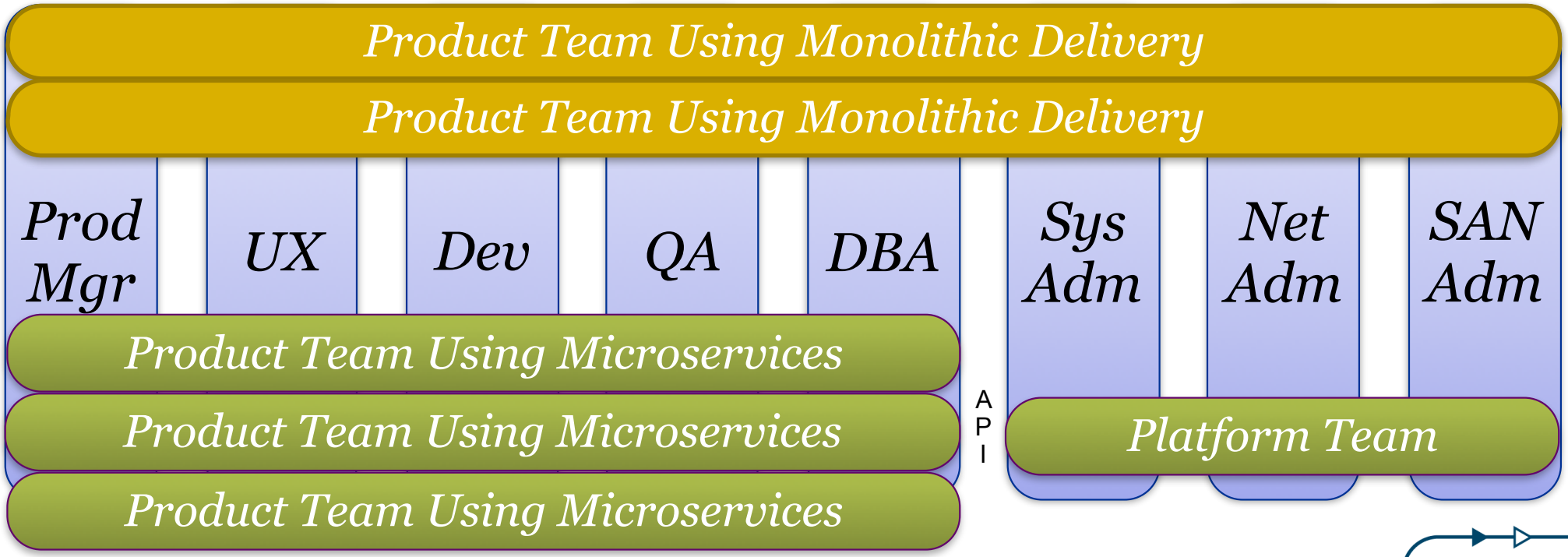
Breaking Down the SILOs



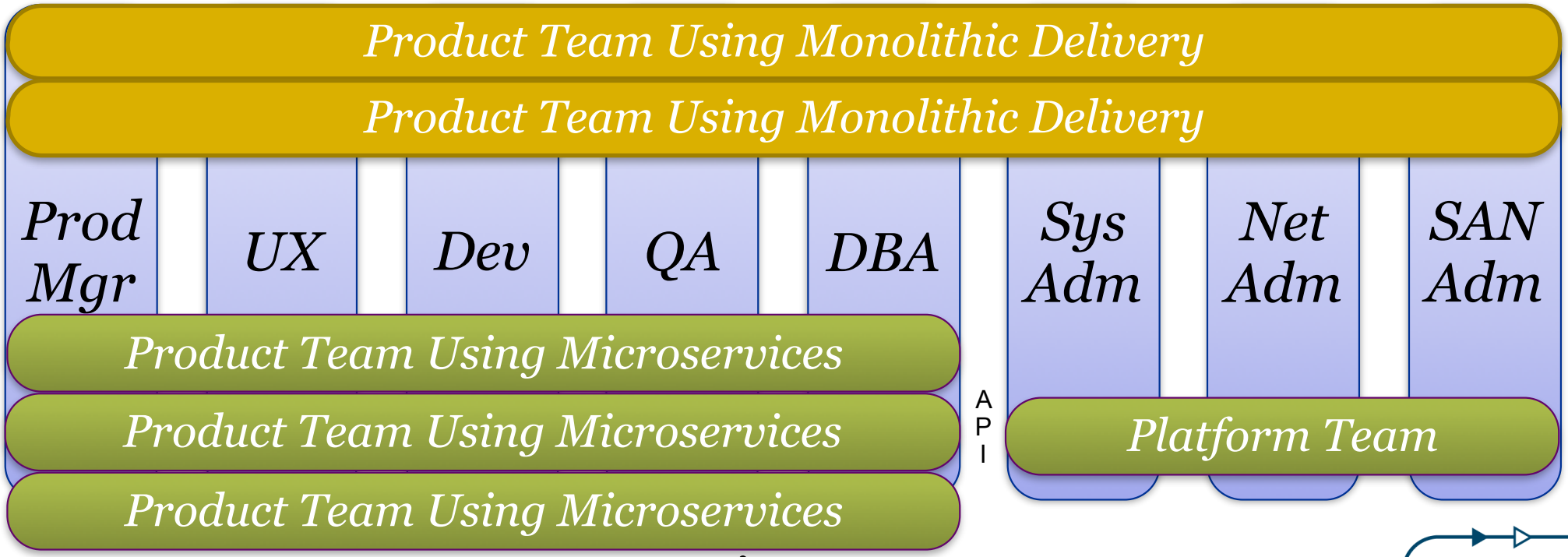
Breaking Down the SILOs



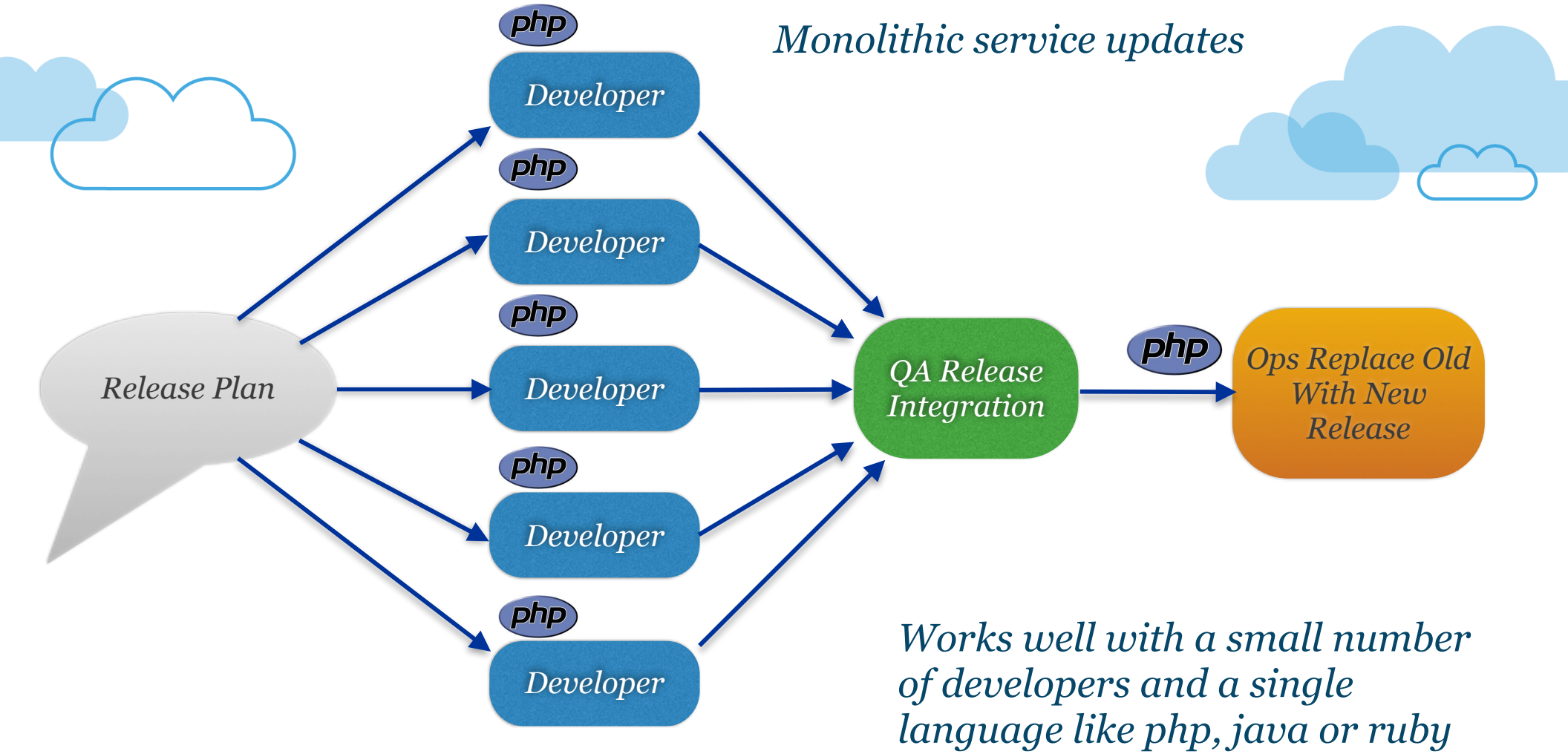
Breaking Down the SILOs

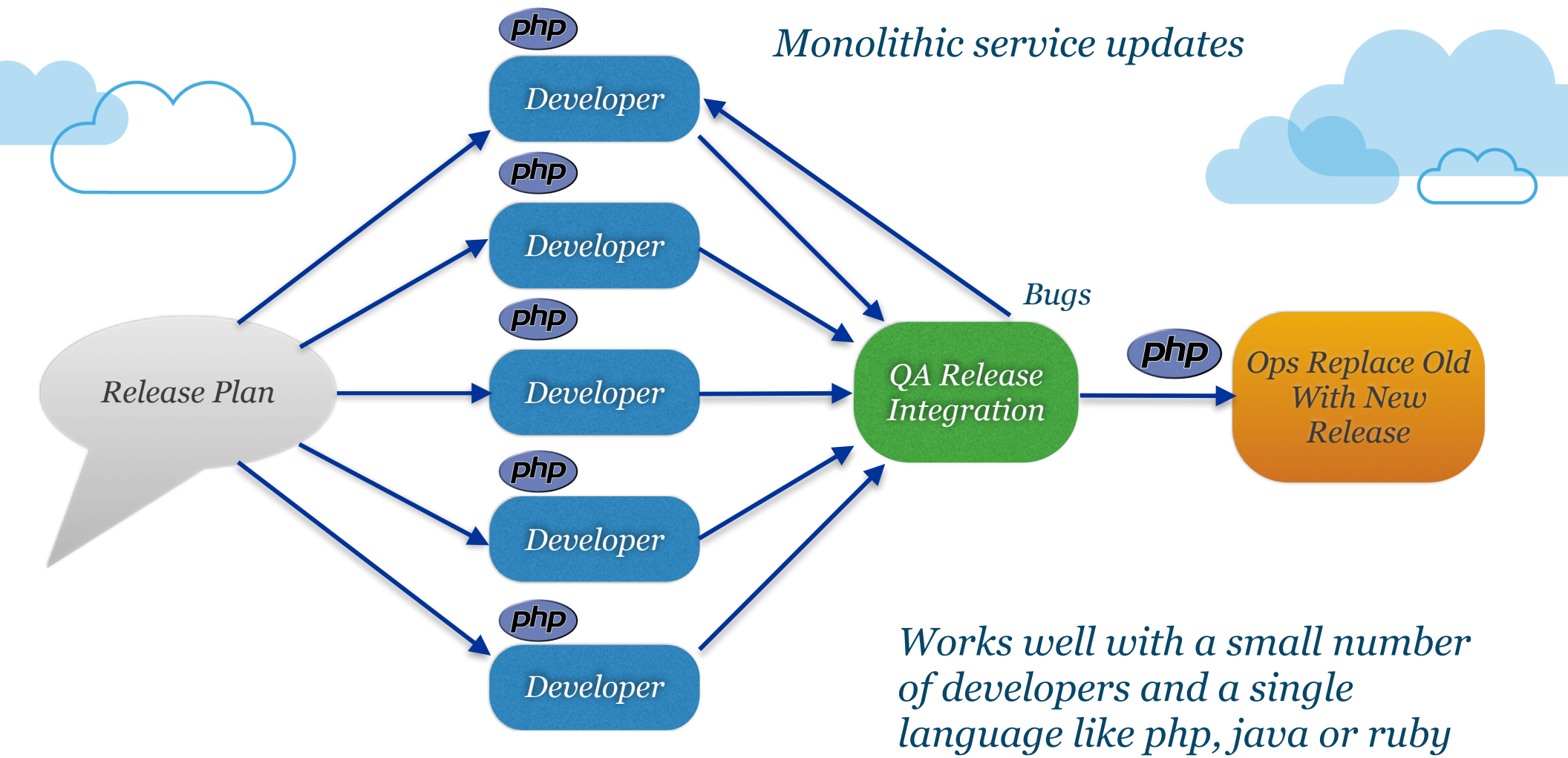


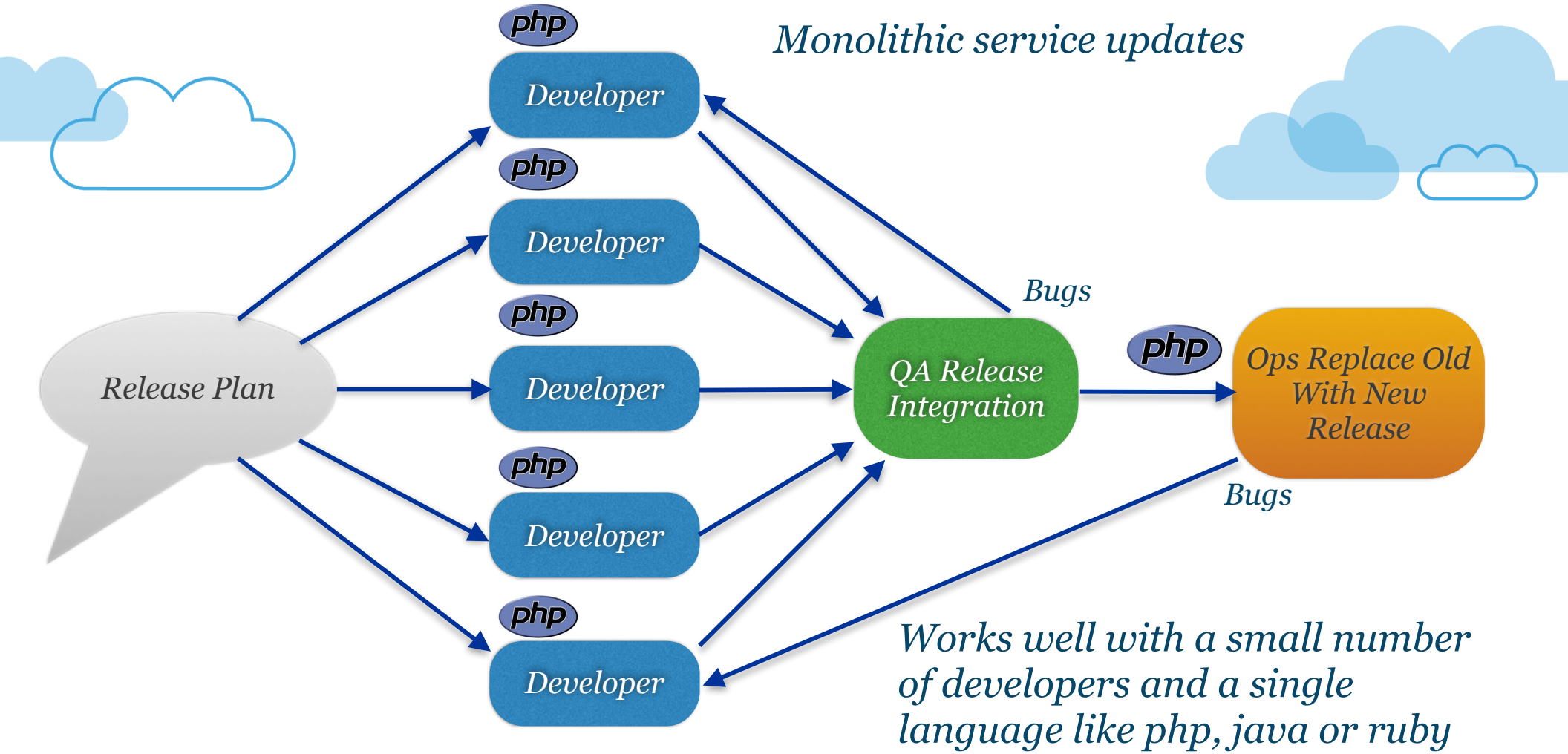
Breaking Down the SILOs



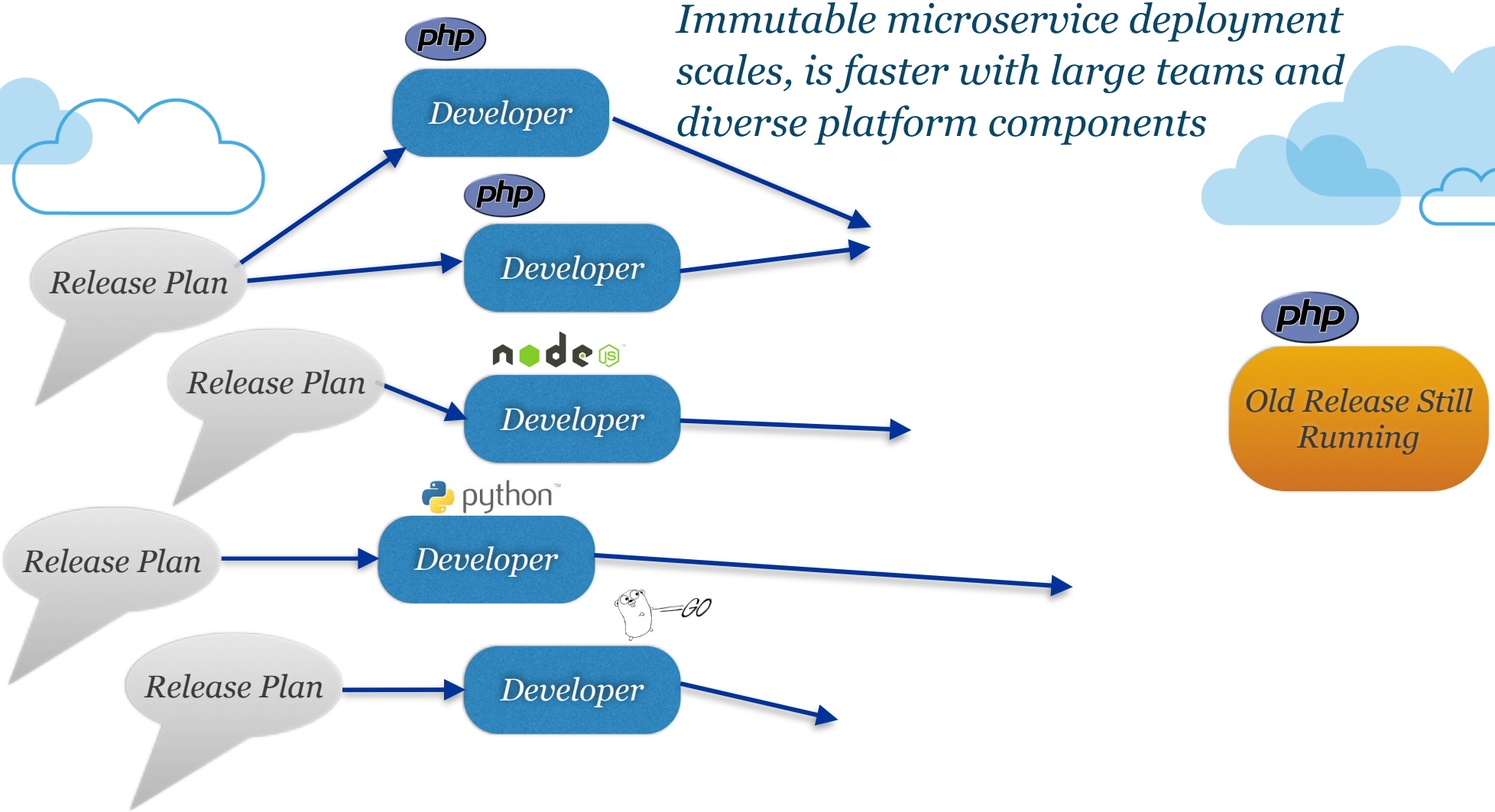
DevOps is a Re-Org!

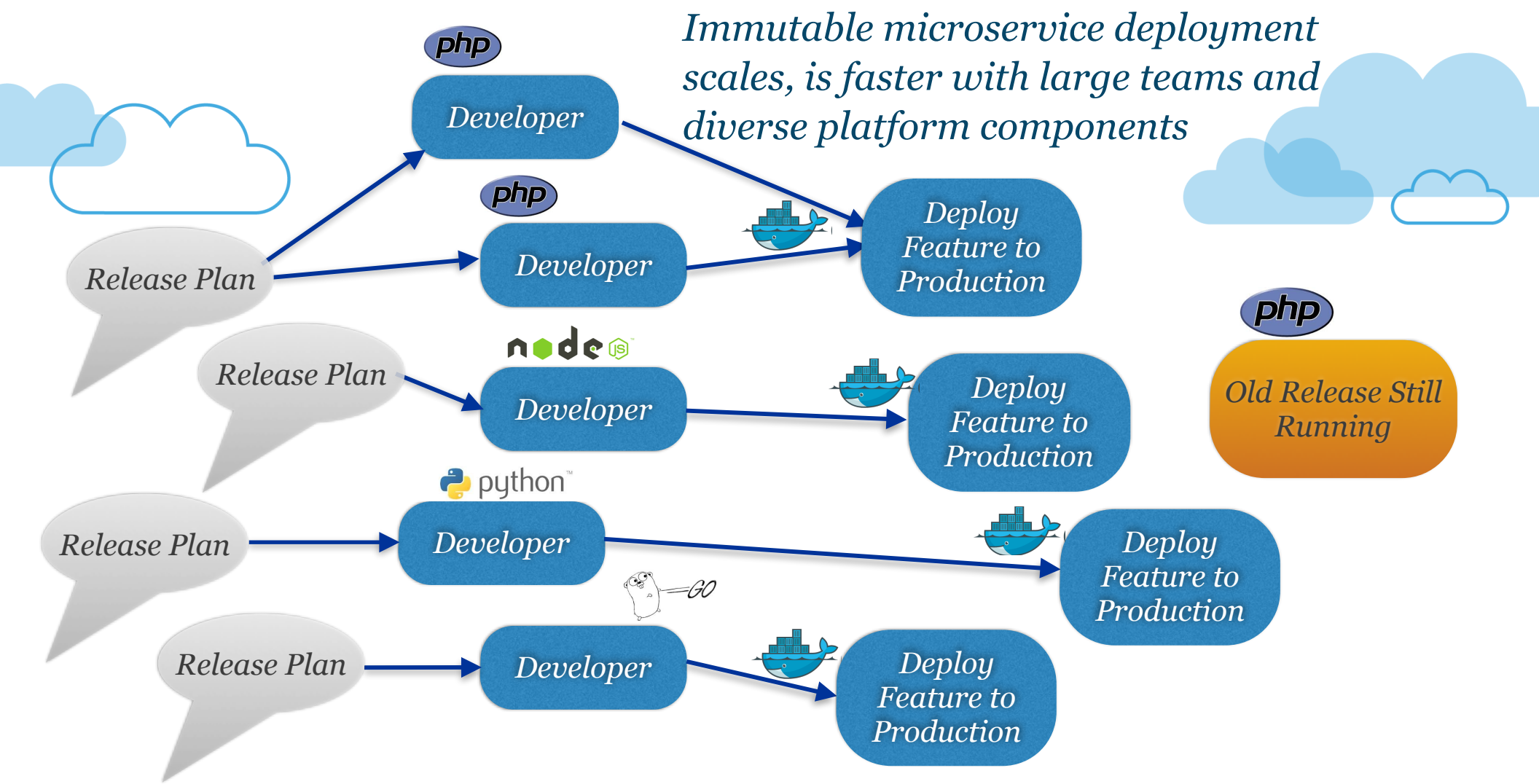

Monolithic service updates
Release Plan
php
Developer
php
Developer
php
Developer
php
Developer
php
Developer
QA Release Integration
php
Ops Replace Old With New Release
Works well with a small number of developers and a single language like php, java or ruby

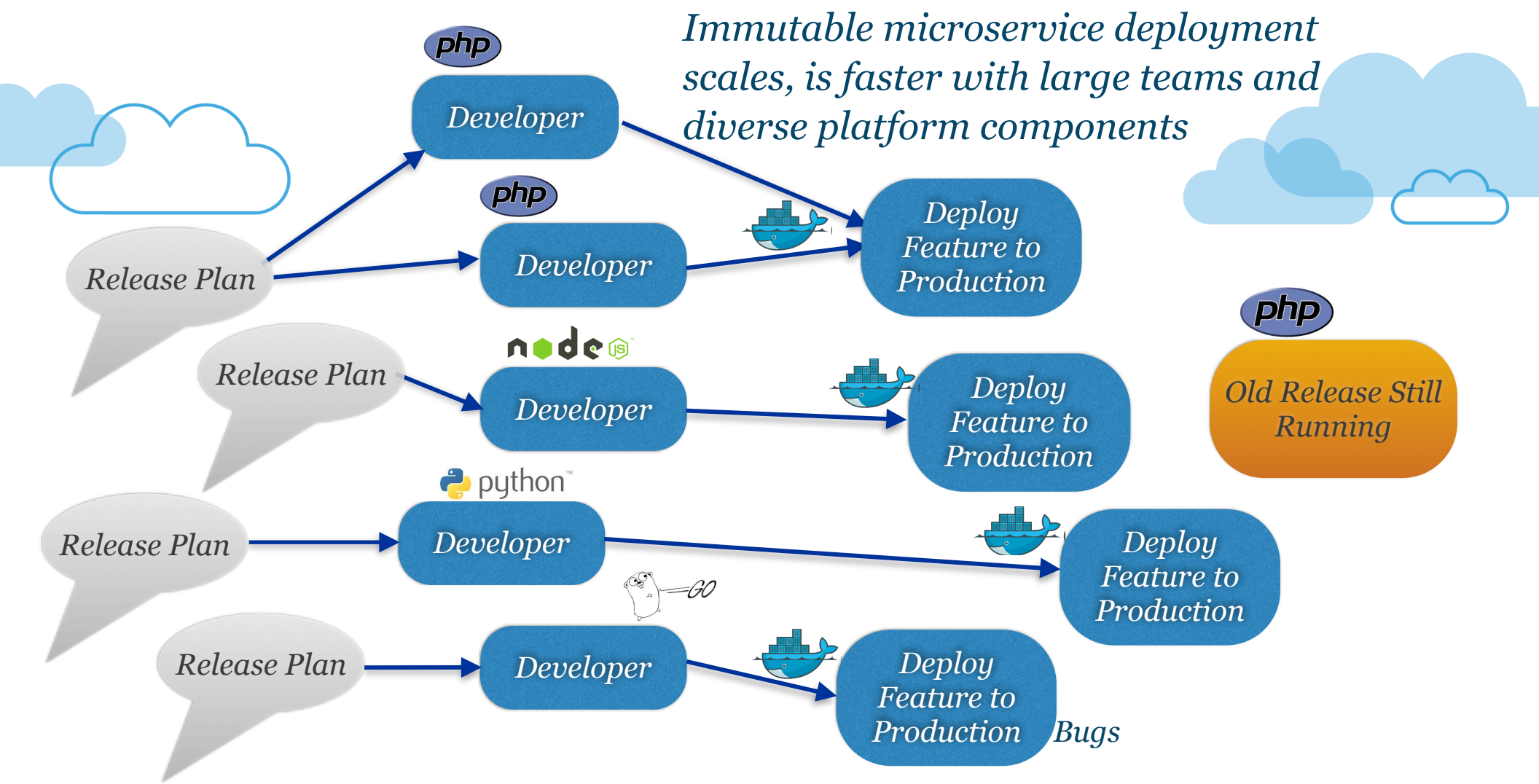

Monolithic service updates
Release Plan
Developer
Developer
Developer
Developer
Developer
QA Release Integration
Bugs
Ops Replace Old With New Release
Works well with a small number of developers and a single language like php, java or ruby

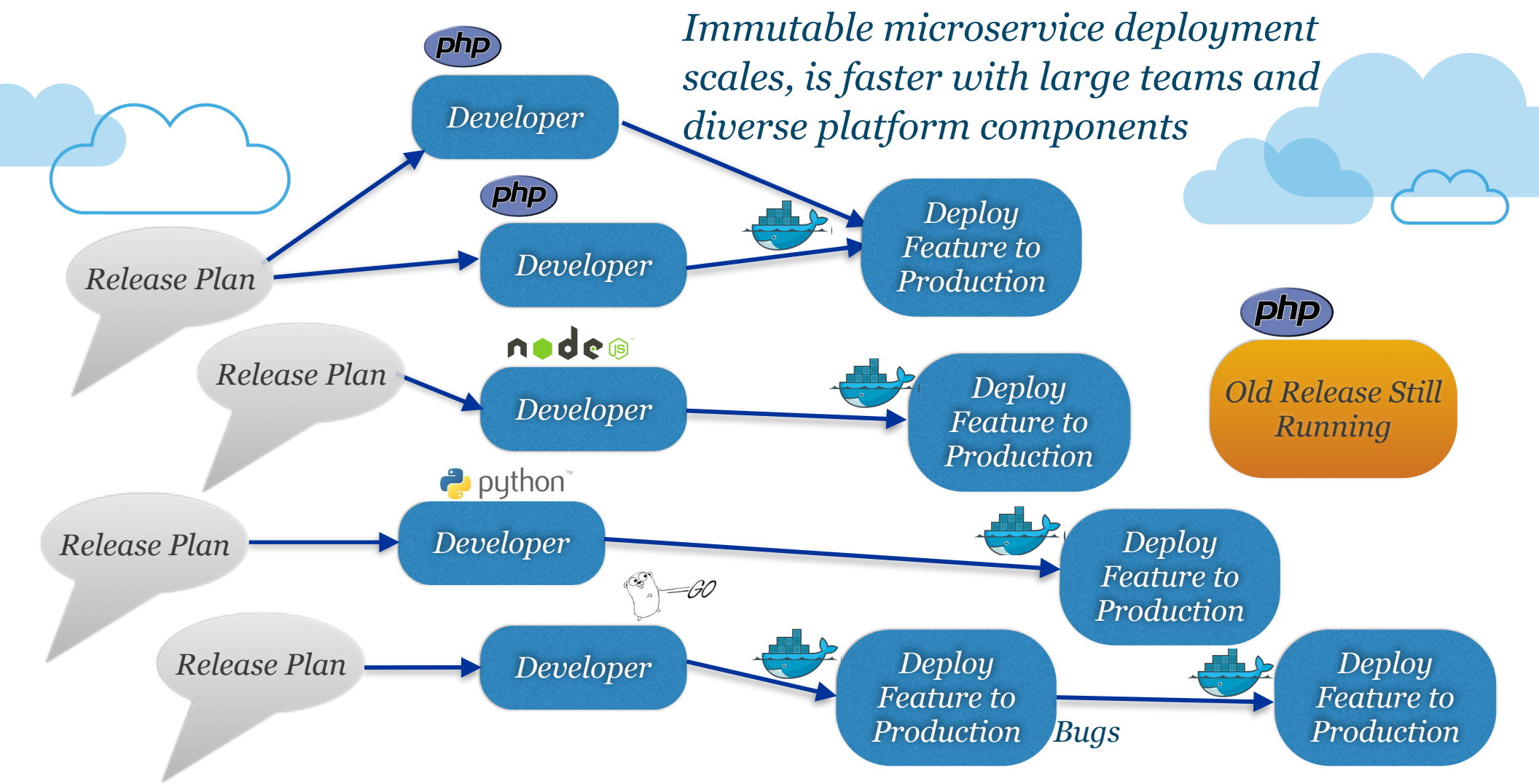

Monolithic service updates
Release Plan
php
Developer
Developer
Developer
Developer
Developer
QA Release Integration
Bugs
Ops Replace Old With New Release
Bugs
Works well with a small number of developers and a single language like php, java or ruby

Immutable microservice deployment scales, is faster with large teams and diverse platform components

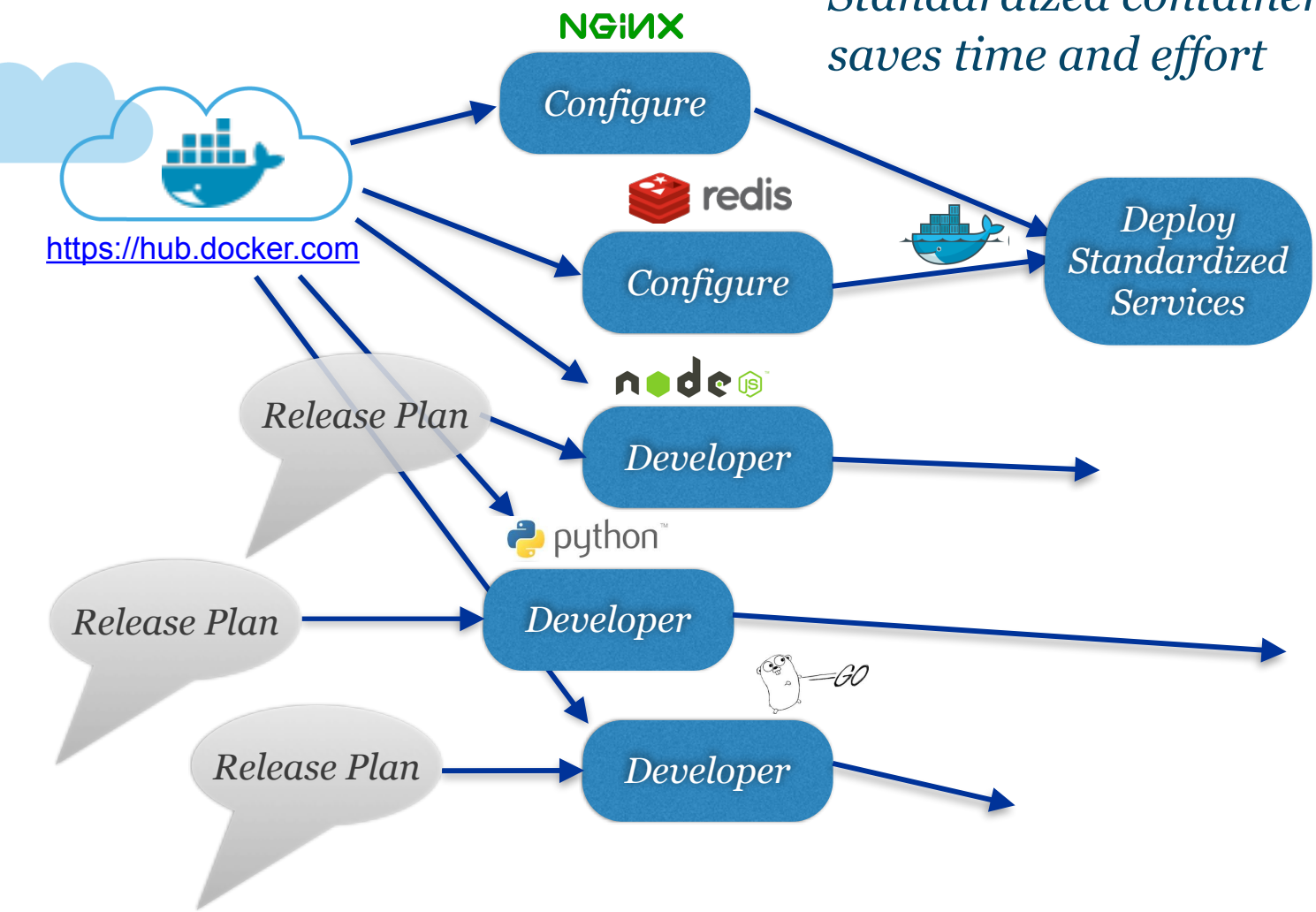


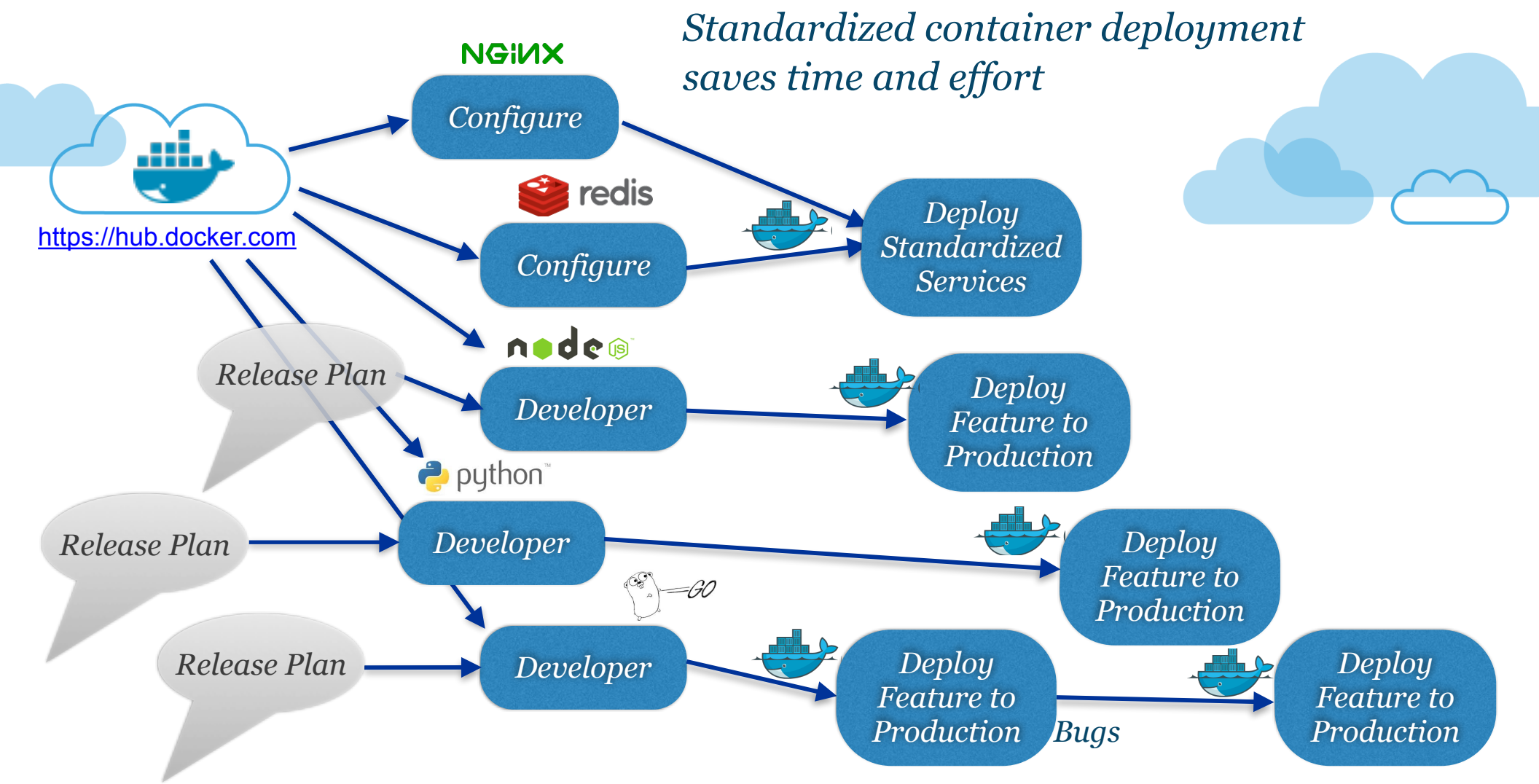

Immutable microservice deployment scales, is faster with large teams and diverse platform components
php
Developer
Release Plan
php
Developer
Deploy Feature to Production
Release Plan
node
Developer
Deploy Feature to Production
php
Old Release Still Running
Release Plan
python
Developer
Deploy Feature to Production
GO
Release Plan
Developer
Deploy Feature to Production


Immutable microservice deployment scales, is faster with large teams and diverse platform components
php
Developer
Release Plan
php
Developer
Deploy Feature to Production
Release Plan
node
Developer
Deploy Feature to Production
php
Old Release Still Running
Release Plan
python
Developer
Deploy Feature to Production
GO
Release Plan
Developer
Deploy Feature to Production
Bugs

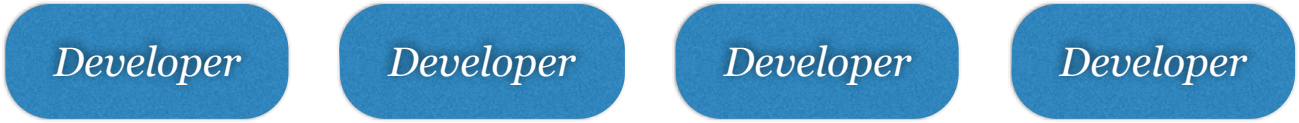

Immutable microservice deployment scales, is faster with large teams and diverse platform components
php
Developer
Release Plan
php
Developer
Deploy Feature to Production
Release Plan
node
Developer
Deploy Feature to Production
php
Old Release Still Running
Release Plan
python
Developer
Deploy Feature to Production
GO
Release Plan
Developer
Deploy Feature to Production
Bugs
Deploy Feature to Production

Standardized container deployment saves time and effort

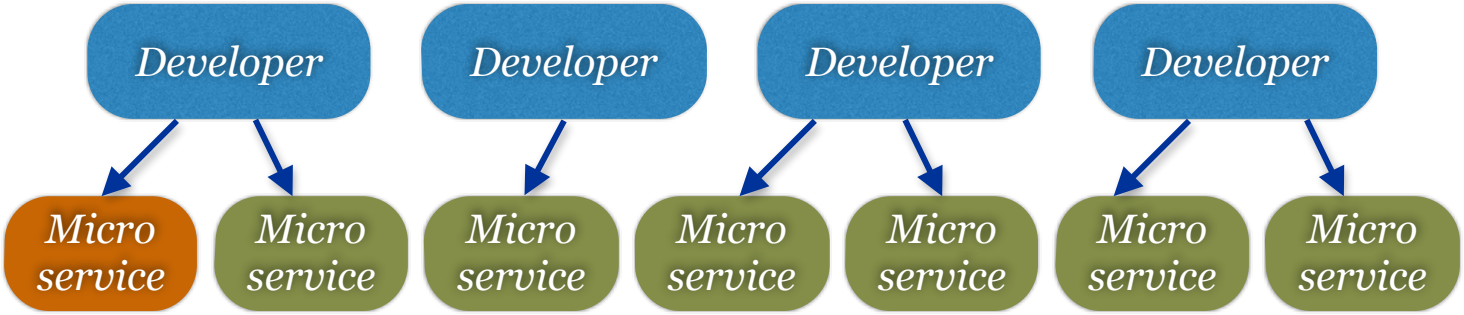



Standardized container deployment saves time and effort
NGINX
Configure
https://hub.docker.com
redis
Configure
Deploy Standardized Services
node
Release Plan
Developer
Deploy Feature to Production
python
Release Plan
Developer
Deploy Feature to Production
GO
Release Plan
Developer
Deploy Feature to Production
Bugs
Deploy Feature to Production

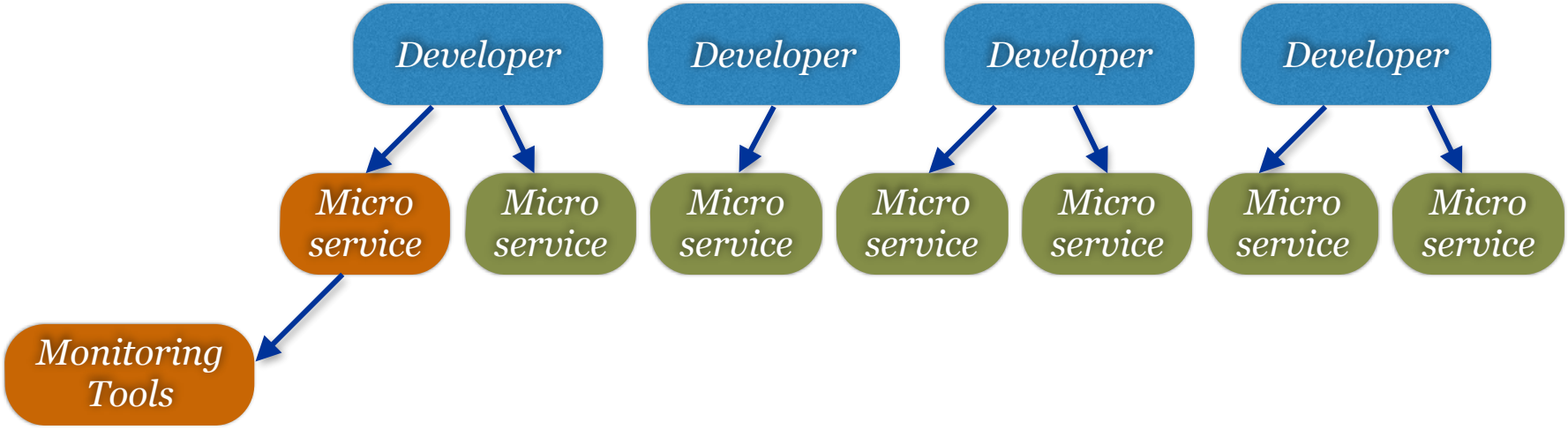
Run What You Wrote



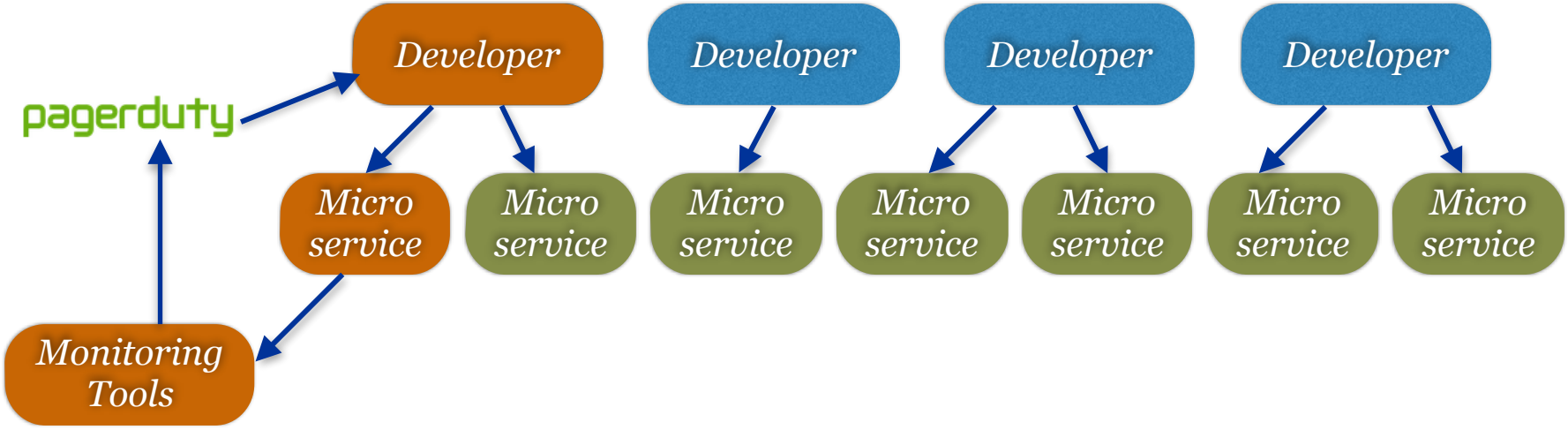
Run What You Wrote



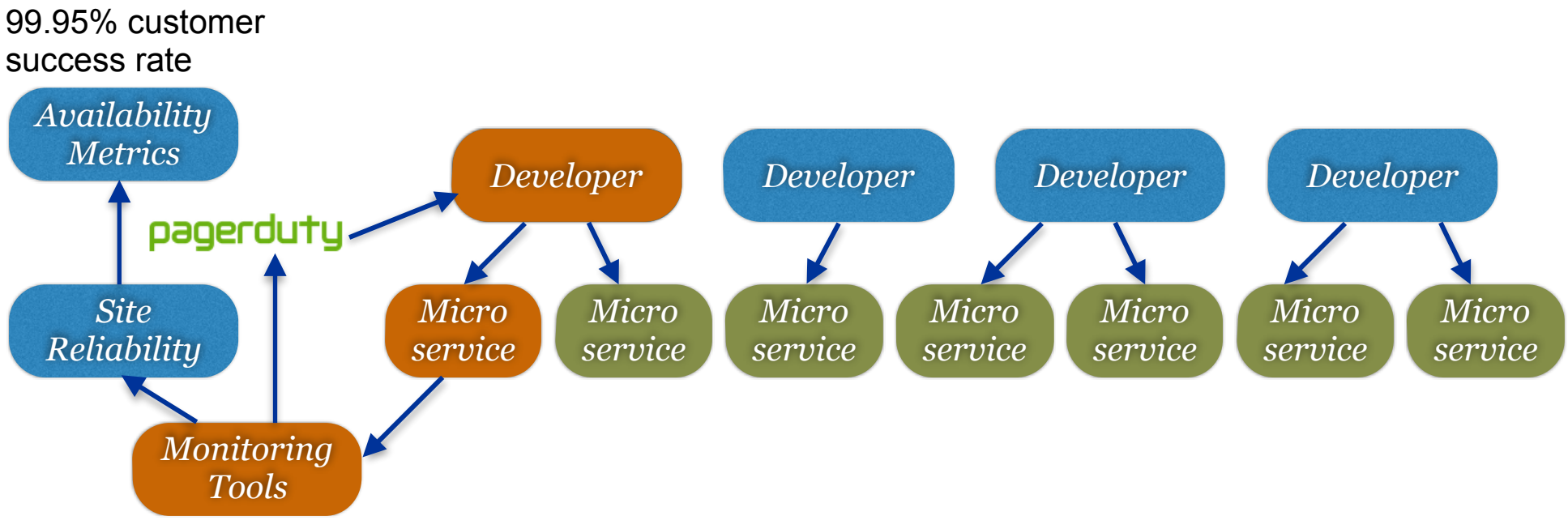
Run What You Wrote



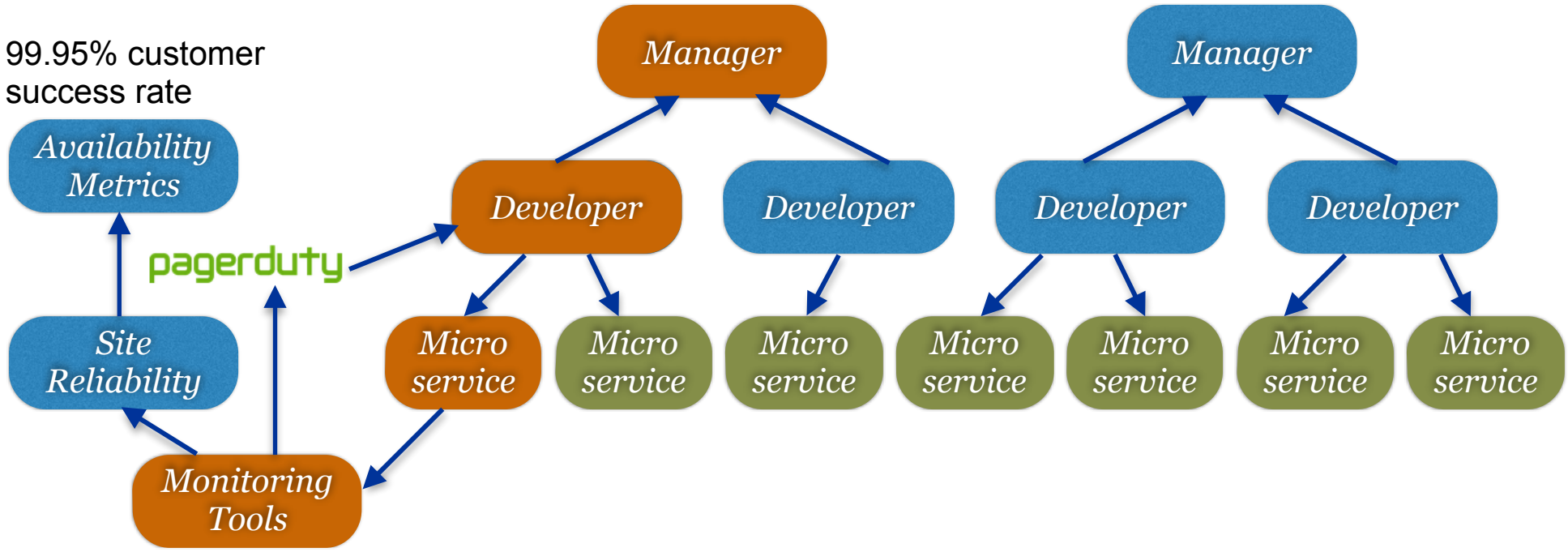
Run What You Wrote



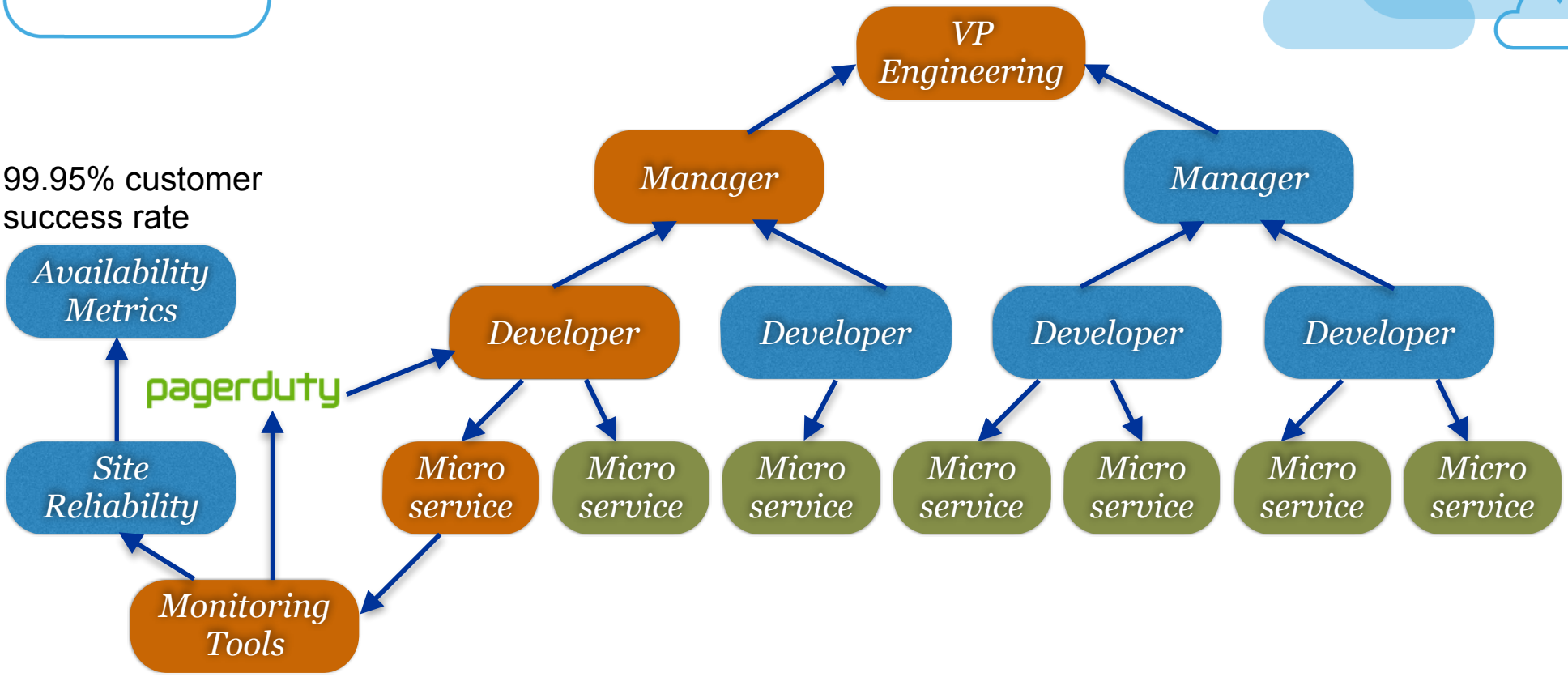
Run What You Wrote



Run What You Wrote



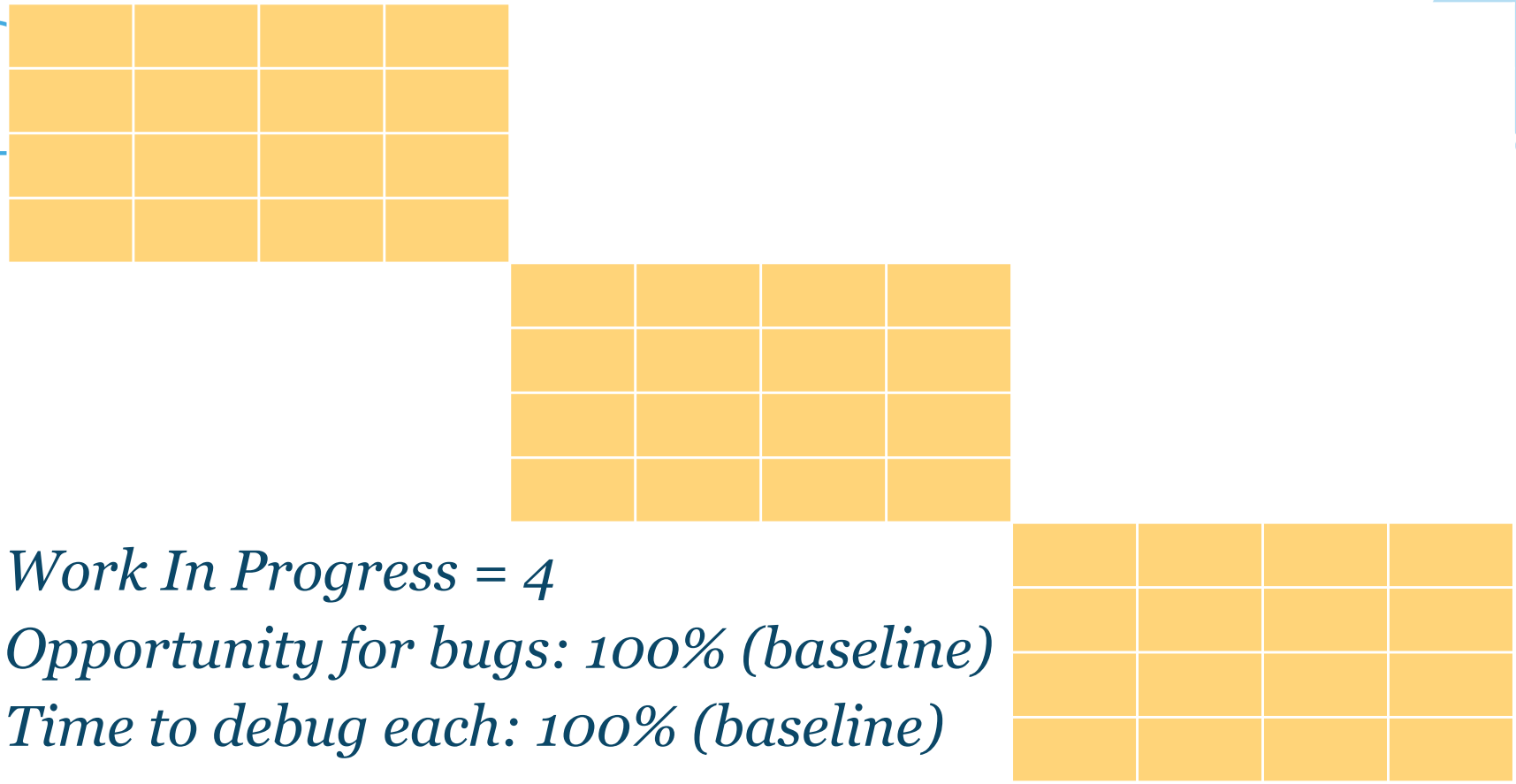
Run What You Wrote



Non-Destructive Production Updates

- *"Immutable Code" Service Pattern*
  - *Existing services are unchanged, old code remains in service*
  - *New code deploys as a new service group*
  - *No impact to production until traffic routing changes*
- *A|B Tests, Feature Flags and Version Routing control traffic*
  - *First users in the test cell are the developer and test engineers*
  - *A cohort of users is added looking for measurable improvement*

Deliver four features every four weeks



Work In Progress = 4

Opportunity for bugs: 100% (baseline)

Time to debug each: 100% (baseline)

Deliver four features every four weeks

Bugs! Which feature broke?
Need more time to test!
Extend release to six weeks?

Work In Progress = 4
Opportunity for bugs: 100% (baseline)
Time to debug each: 100% (baseline)

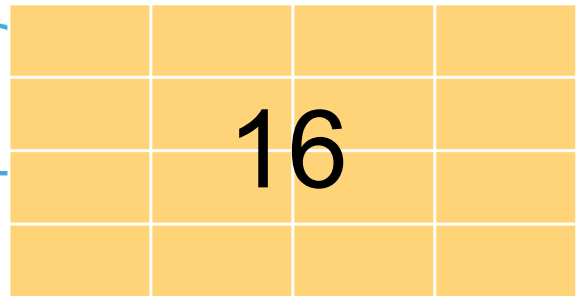
Deliver four features every four weeks

Bugs! Which feature broke?
Need more time to test!
Extend release to six weeks?

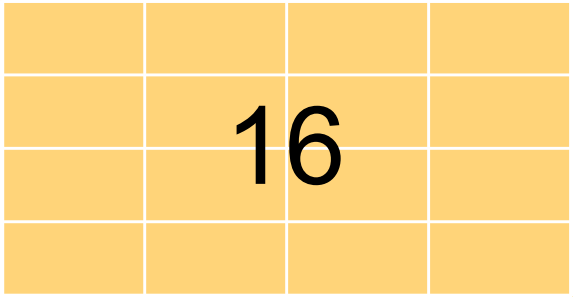
Work In Progress = 4
Opportunity for bugs: 100% (baseline)
Time to debug each: 100% (baseline)

But: risk of bugs in delivery increases with interactions!

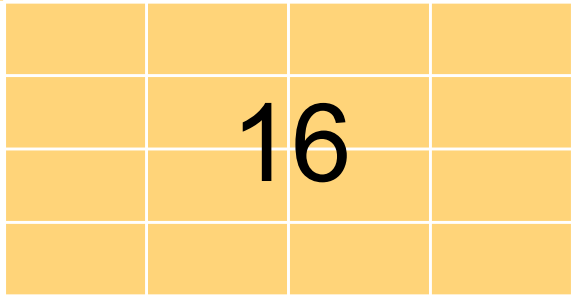
Deliver four features every four weeks



Bugs! Which feature broke?
Need more time to test!
Extend release to six weeks?

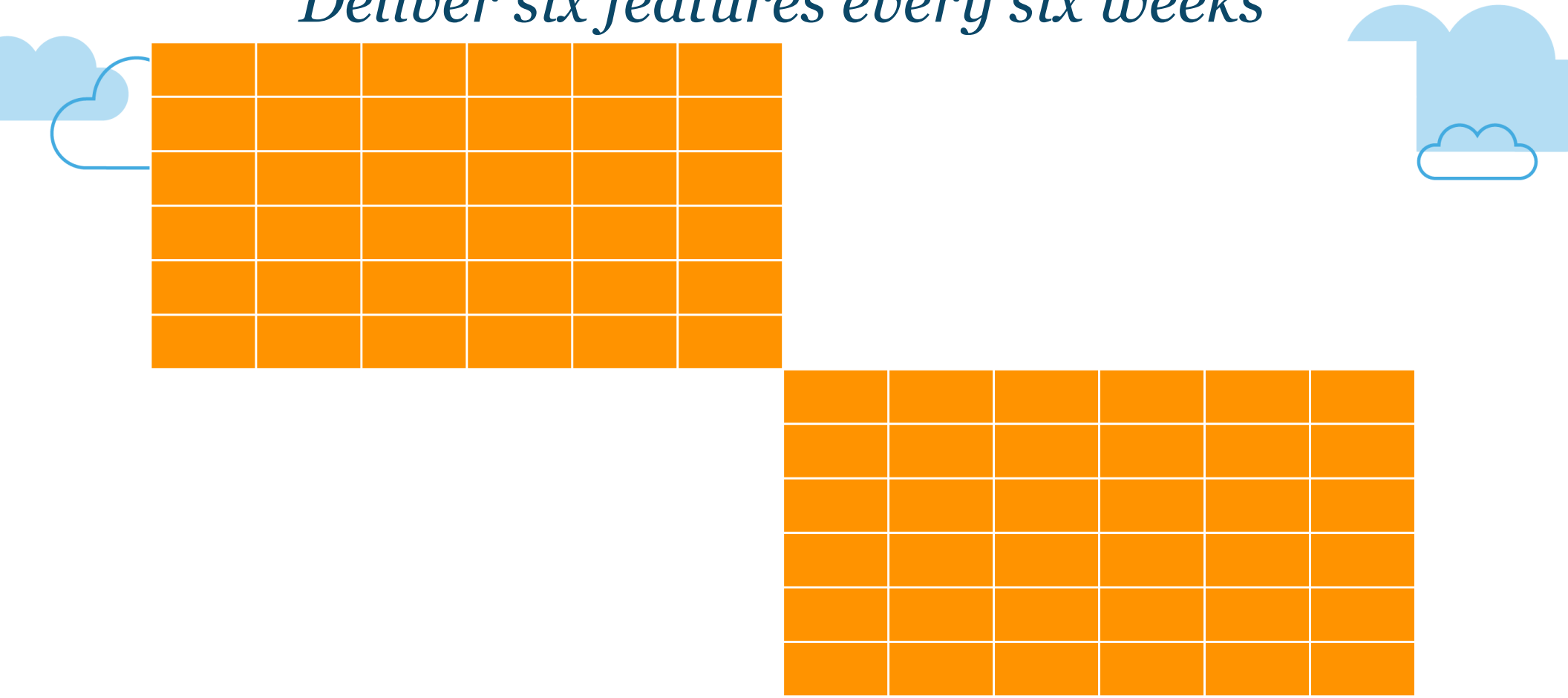


Work In Progress = 4
Opportunity for bugs: 100% (baseline)
Time to debug each: 100% (baseline)

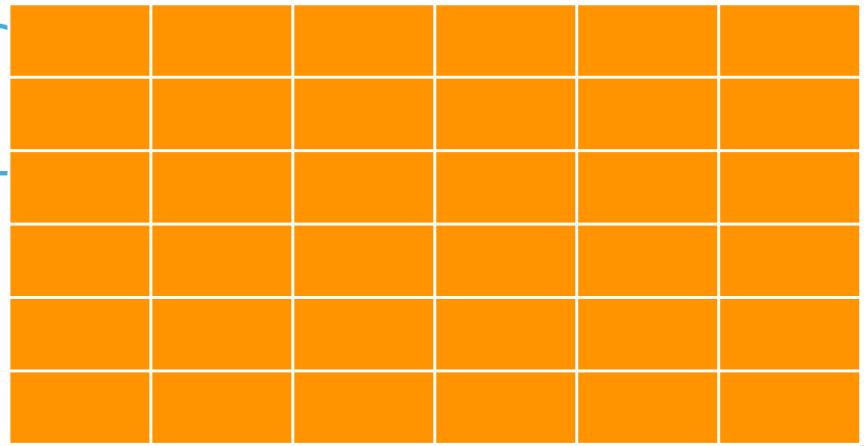


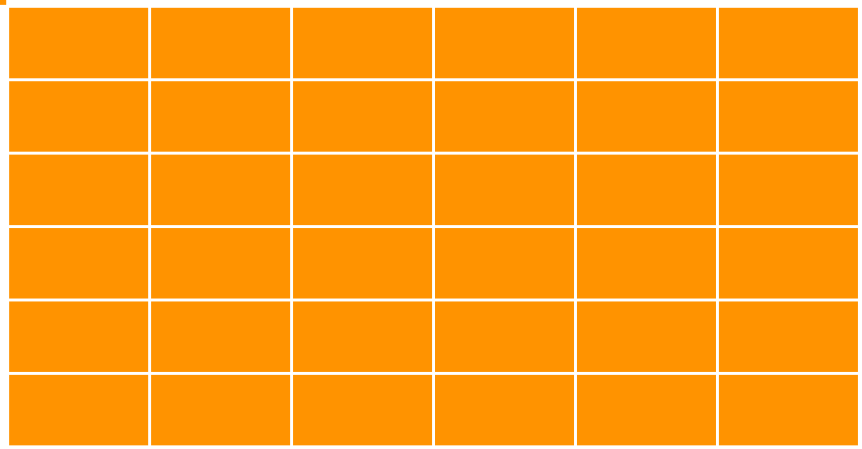
But: risk of bugs in delivery increases with interactions!

Deliver six features every six weeks



Deliver six features every six weeks



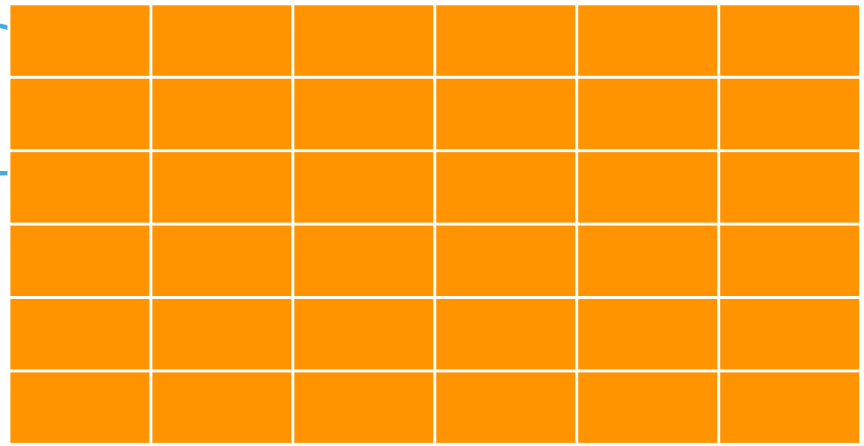


Work In Progress = 6

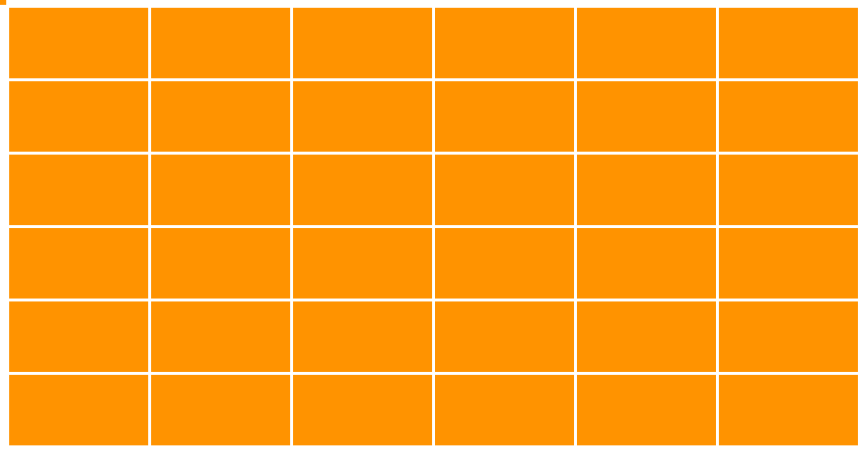
Individual bugs: 150%

Interactions: 150%?

Deliver six features every six weeks

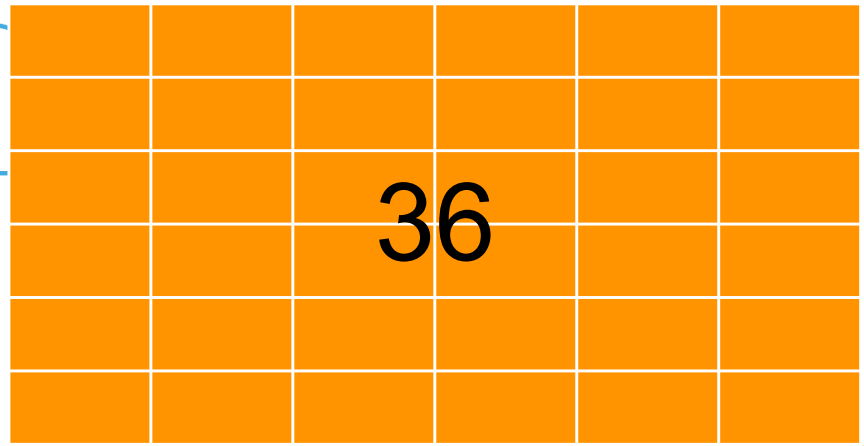


More features
What broke?
More interactions
Even more bugs!!



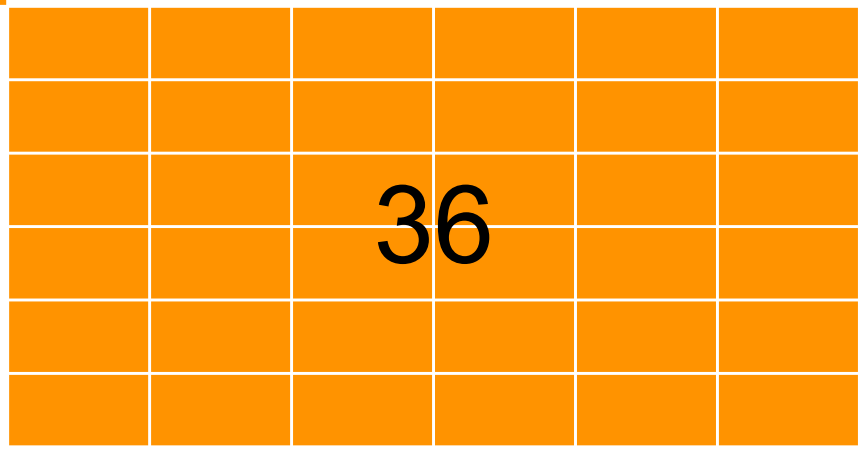
Work In Progress = 6
Individual bugs: 150%
Interactions: 150%?

Deliver six features every six weeks

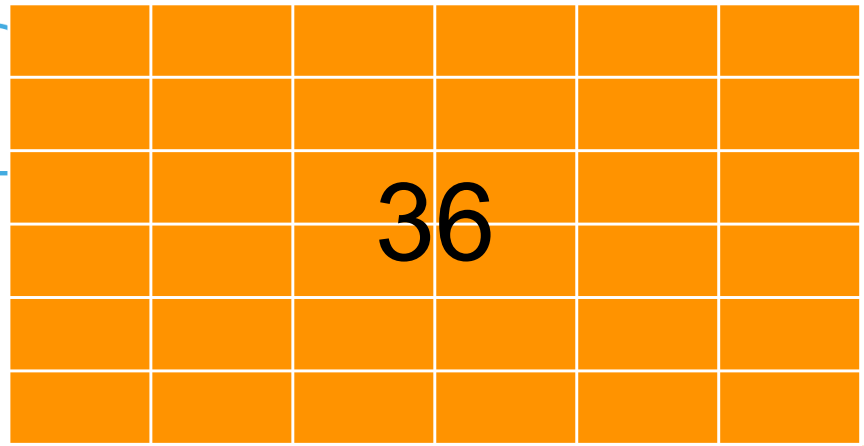


More features
What broke?
More interactions
Even more bugs!!

Work In Progress = 6
Individual bugs: 150%
Interactions: 150%?

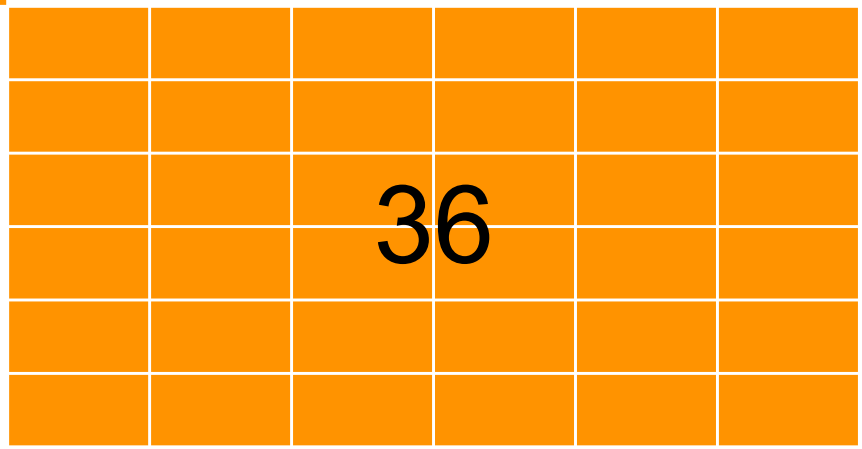


Deliver six features every six weeks



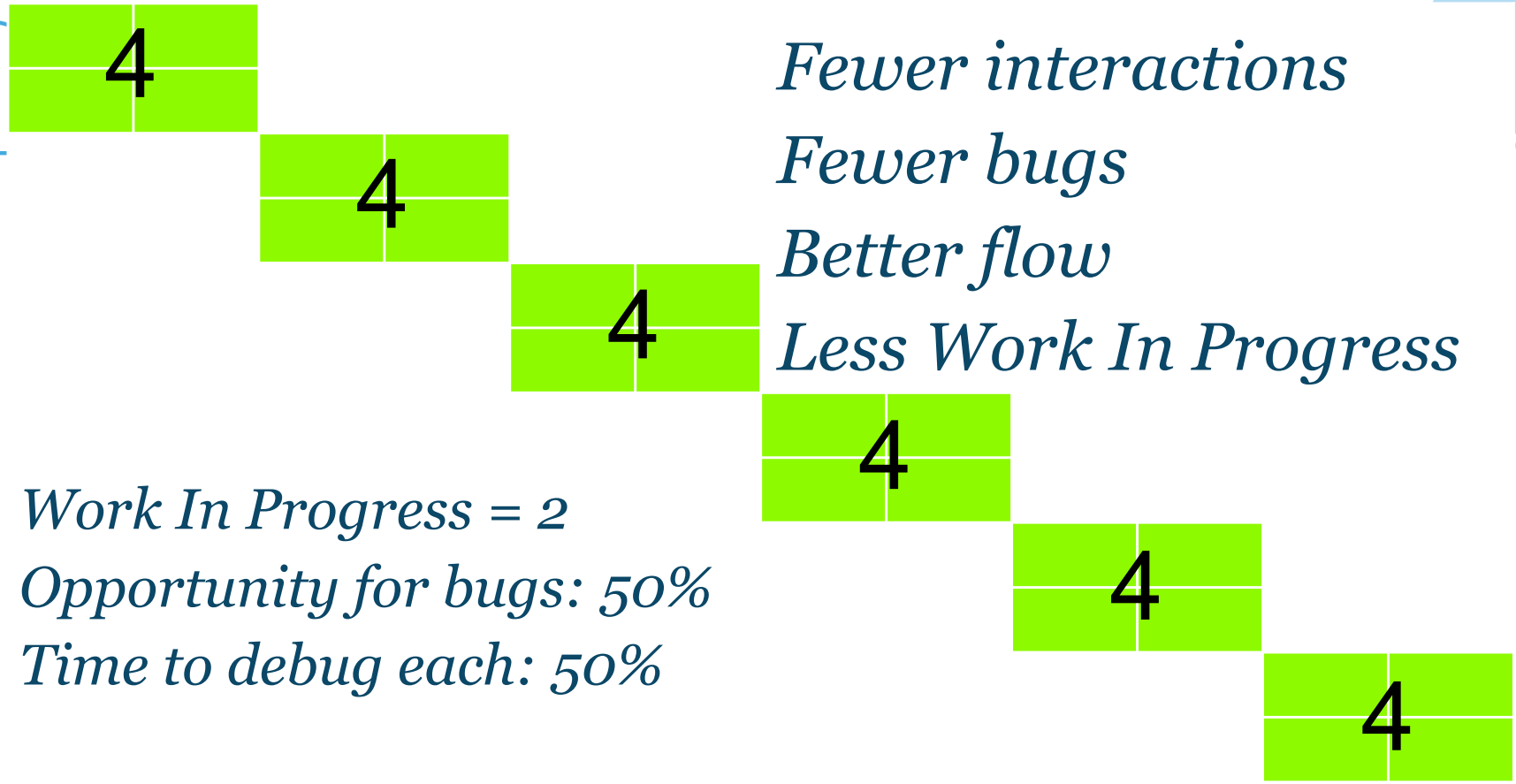
More features
What broke?
More interactions
Even more bugs!!

Work In Progress = 6
Individual bugs: 150%
Interactions: 150%?



Risk of bugs in delivery increased to 225% of original!

Deliver two features every two weeks



Fewer interactions

Fewer bugs

Better flow

Less Work In Progress

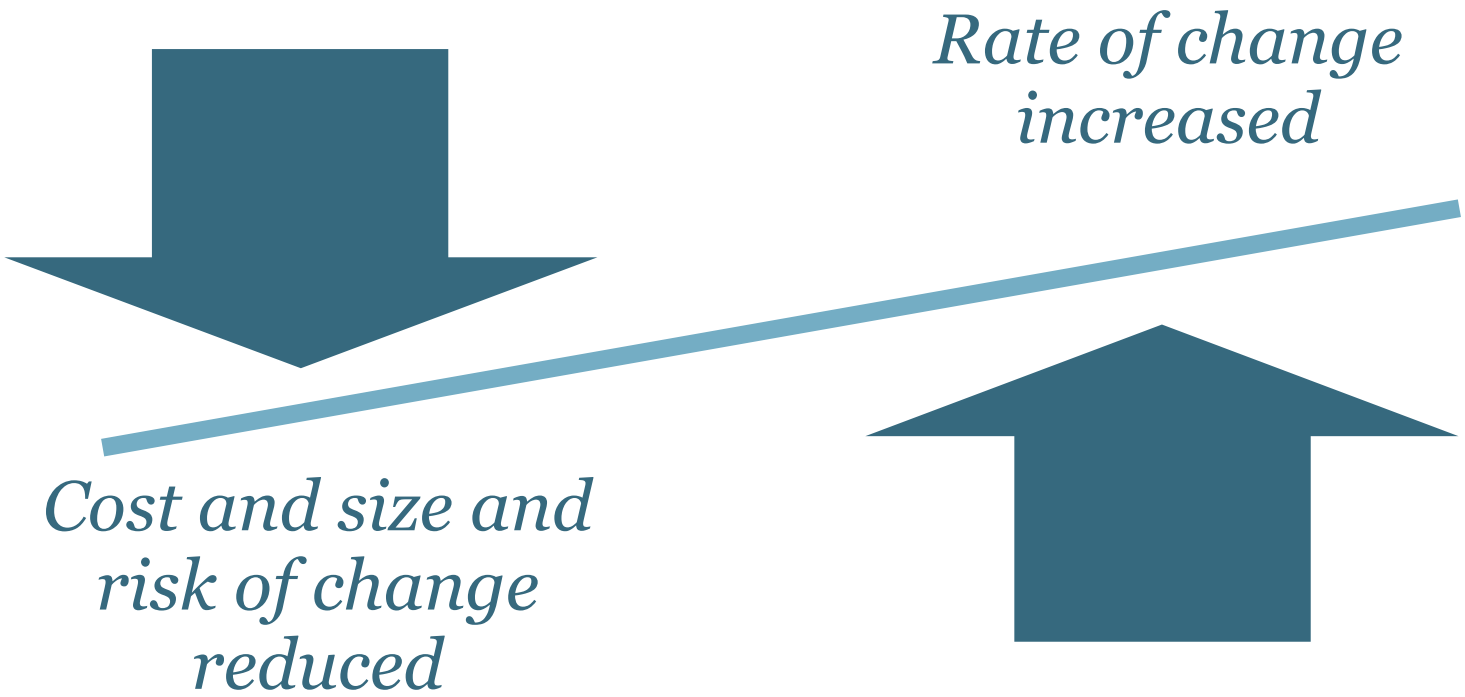
Work In Progress = 2

Opportunity for bugs: 50%

Time to debug each: 50%

Complexity of delivery decreased by 75% from original

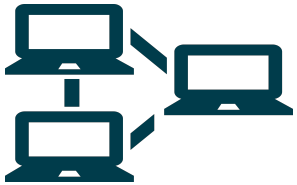
What Happened?



Developing at the Speed of Docker

Developers
- Compile/Build
- Seconds

Extend container
- Package dependencies
- Seconds

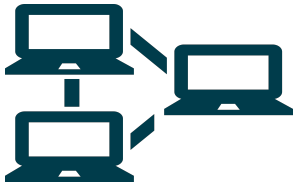


PaaS deploy Container
- Docker startup
- Seconds

Developing at the Speed of Docker

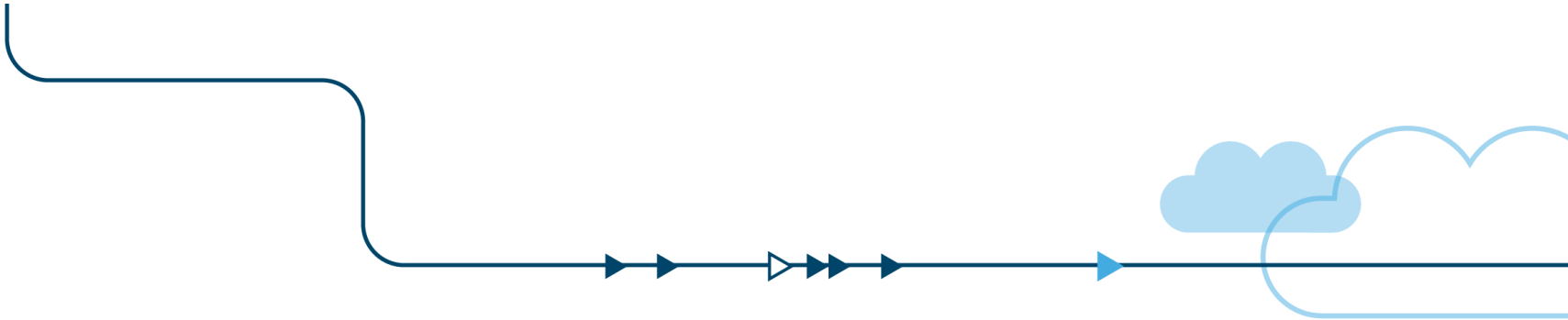
Developers
- Compile/Build
- Seconds

Extend container
- Package dependencies
- Seconds

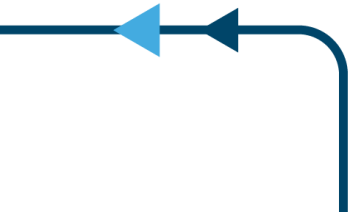


PaaS deploy Container
- Docker startup
- Seconds

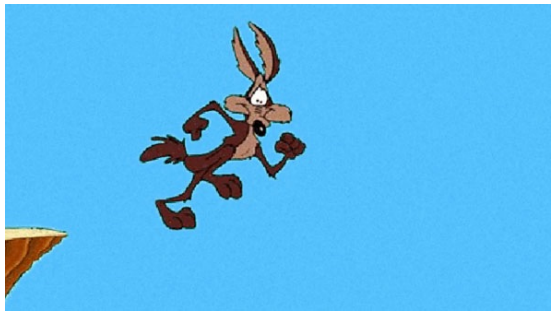
Speed is addictive, hard to go back to taking much longer to get things done



Disruptor: Continuous Delivery with Containerized Microservices

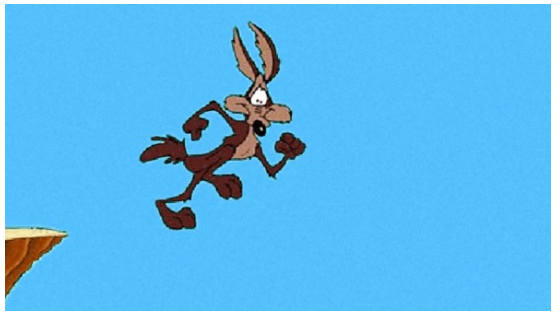


It’s what you know that isn’t so



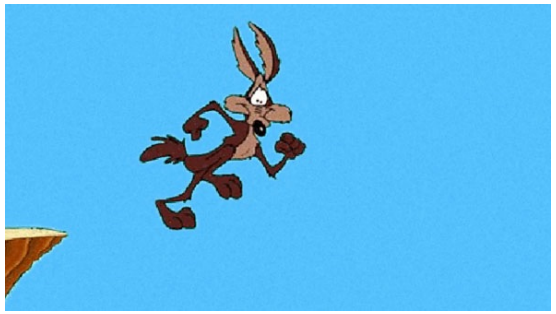
It’s what you know that isn’t so

- *Make your assumptions explicit*



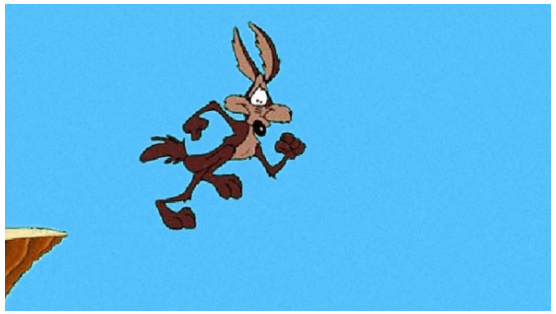
It’s what you know that isn’t so

- *Make your assumptions explicit*
- *Extrapolate trends to the limit*



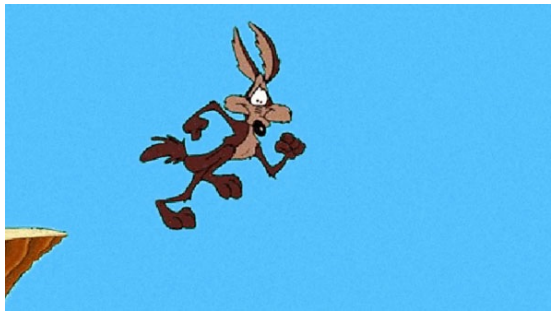
It’s what you know that isn’t so

- *Make your assumptions explicit*
- *Extrapolate trends to the limit*
- *Listen to non-customers*

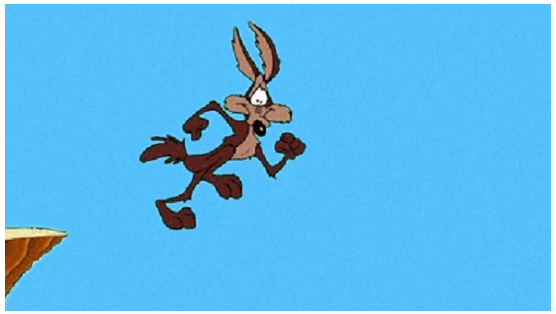


It’s what you know that isn’t so

- *Make your assumptions explicit*
- *Extrapolate trends to the limit*
- *Listen to non-customers*
- *Follow developer adoption, not IT spend*

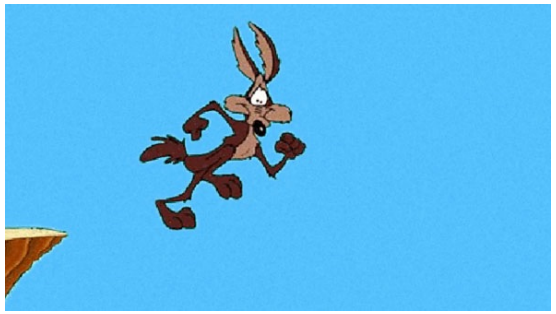


It’s what you know that isn’t so

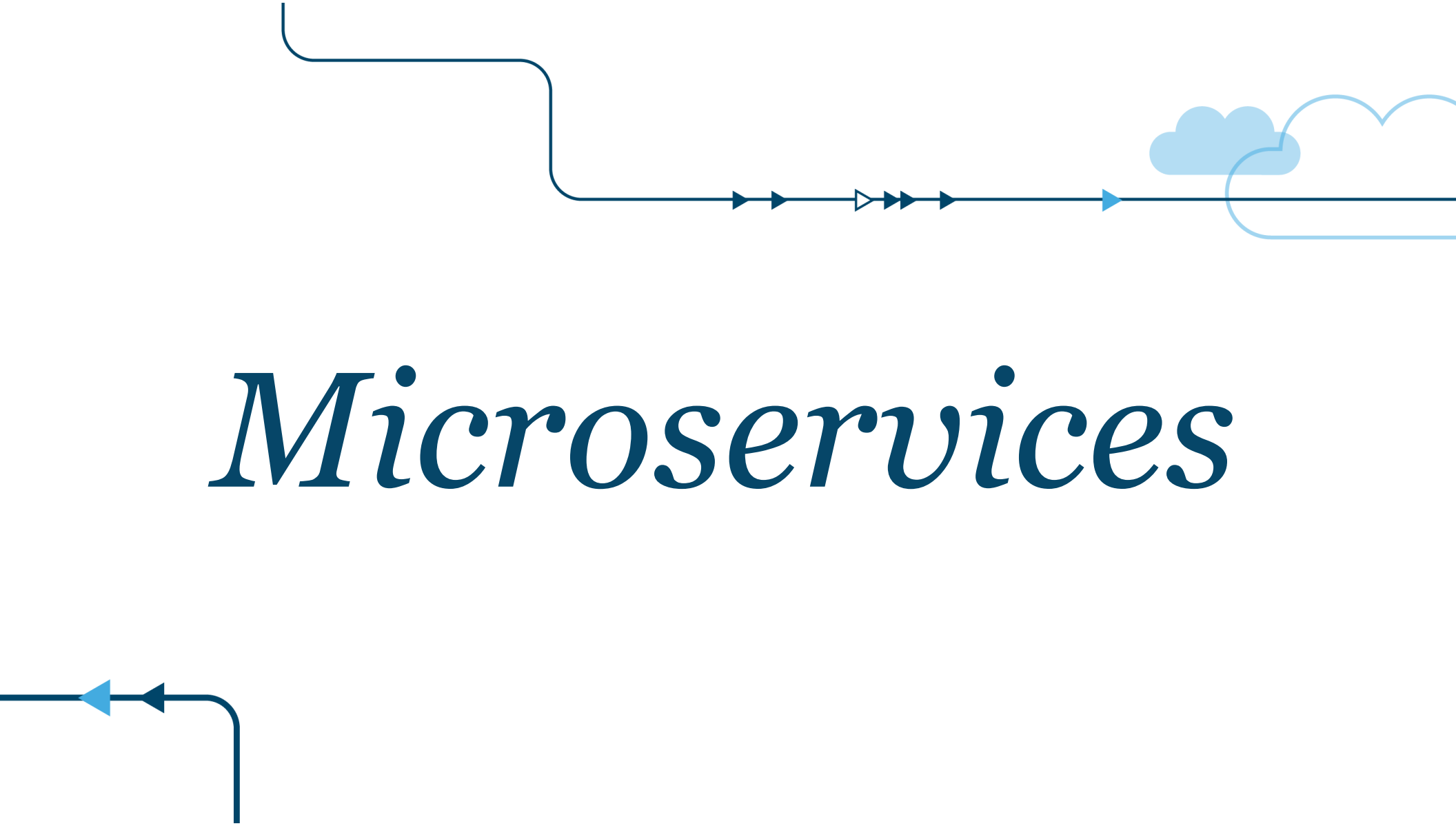


- *Make your assumptions explicit*
- *Extrapolate trends to the limit*
- *Listen to non-customers*
- *Follow developer adoption, not IT spend*
- *Map evolution of products to services to utilities*

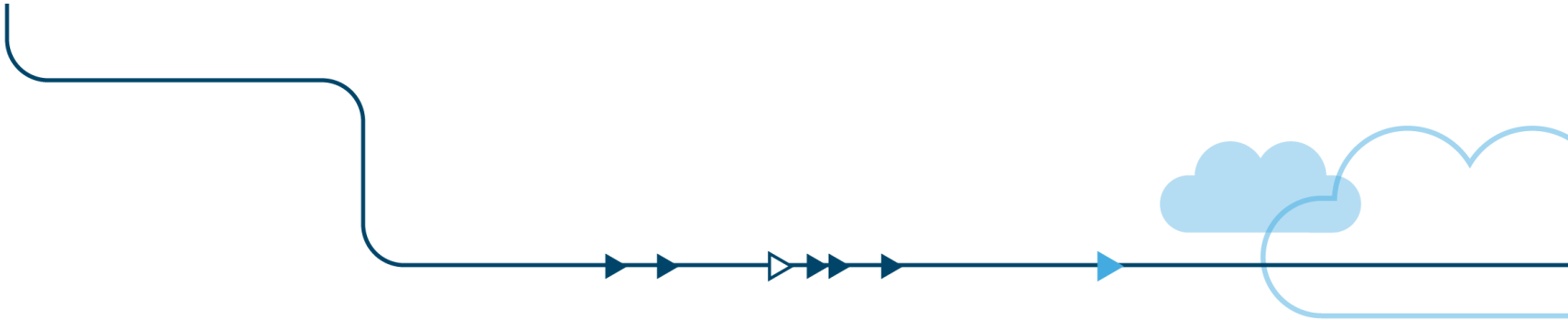
It’s what you know that isn’t so



- *Make your assumptions explicit*
- *Extrapolate trends to the limit*
- *Listen to non-customers*
- *Follow developer adoption, not IT spend*
- *Map evolution of products to services to utilities*
- *Re-organize your teams for speed of execution*

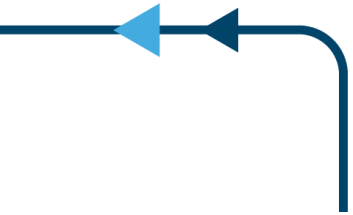


Microservices



A Microservice Definition

Loosely coupled service oriented architecture with bounded contexts



If every service has to be updated at the same time it's not loosely coupled

A Microservice Definition

Loosely coupled service oriented architecture with bounded contexts

If every service has to be updated at the same time it's not loosely coupled

A Microservice Definition

Loosely coupled service oriented architecture with bounded contexts

If you have to know too much about surrounding services you don't have a bounded context. See the Domain Driven Design book by Eric Evans.

Coupling Concerns

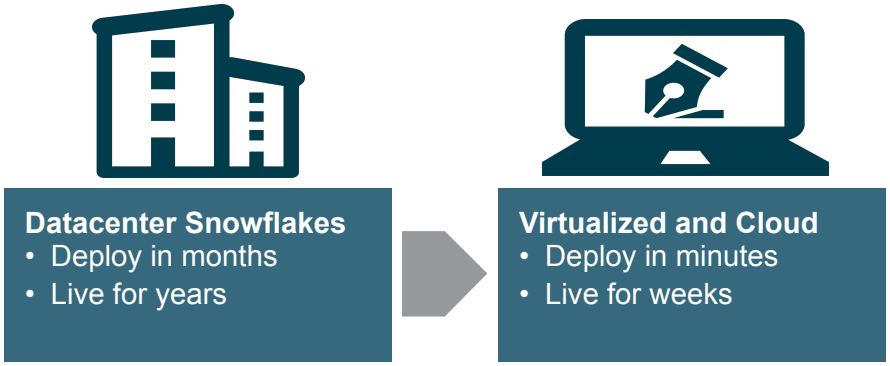
- *Conway's Law - organizational coupling*
- *Centralized Database Schemas*
- *Enterprise Service Bus - centralized message queues*
- *Inflexible Protocol Versioning*

http://en.wikipedia.org/wiki/Conway's_law

Speeding Up The Platform

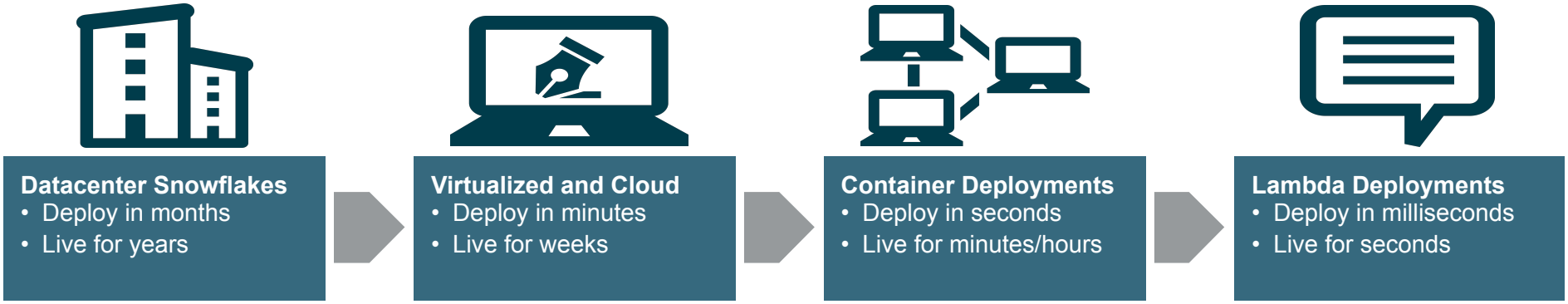
Datacenter Snowflakes
- Deploy in months
- Live for years

Speeding Up The Platform

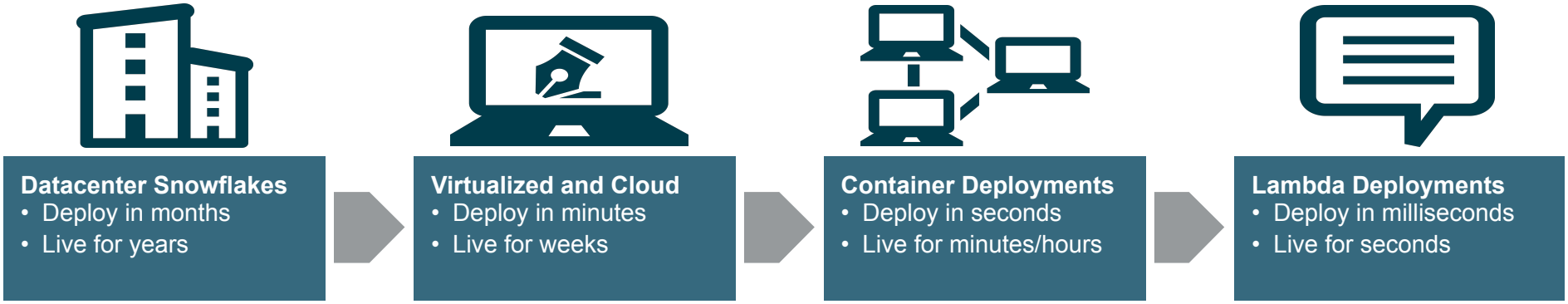


Speeding Up The Platform

Speeding Up The Platform



Speeding Up The Platform



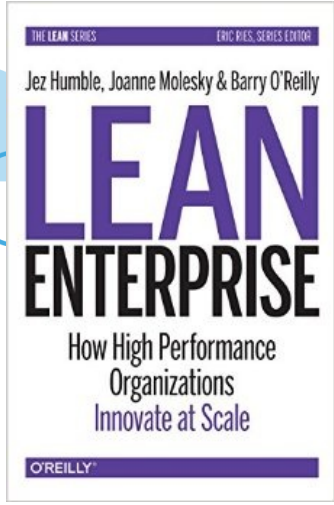
Speed enables and encourages microservice architectures

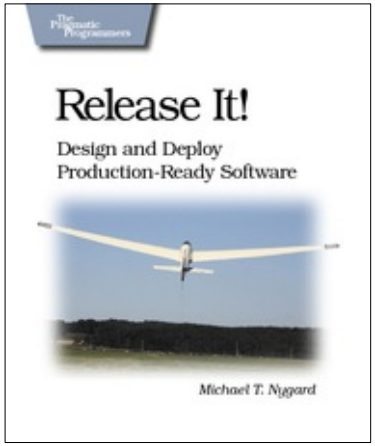
Separate Concerns with Microservices

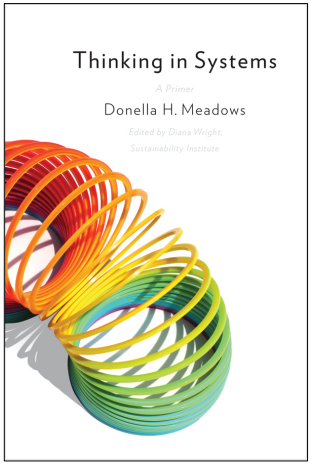
- *Invert Conway's Law – teams own service groups and backend stores*
- *One "verb" per single function micro-service, size doesn't matter*
- *One developer independently produces a micro-service*
- *Each micro-service is it's own build, avoids trunk conflicts*
- *Deploy in a container: Tomcat, AMI or Docker, whatever…*
- *Stateless business logic. Cattle, not pets.*
- *Stateful cached data access layer using replicated ephemeral instances*

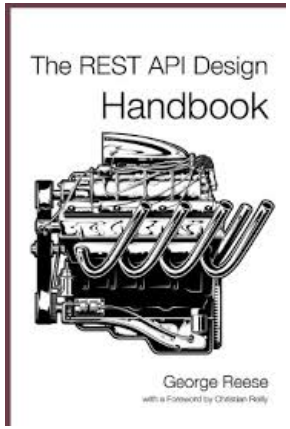
http://en.wikipedia.org/wiki/Conway's_law

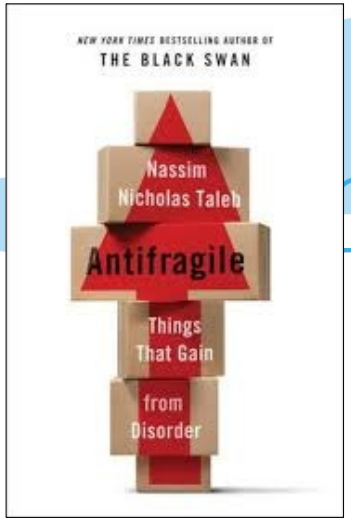
Inspiration

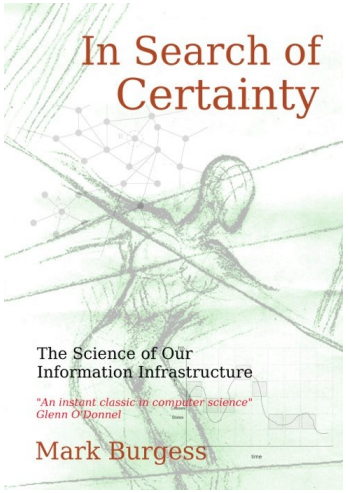


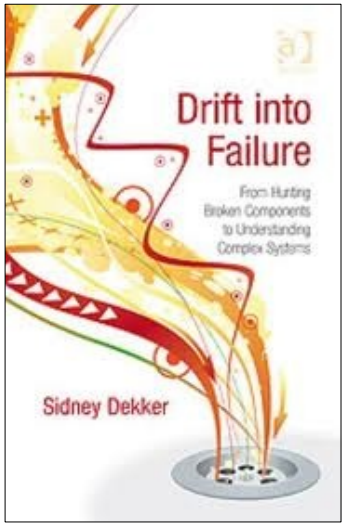


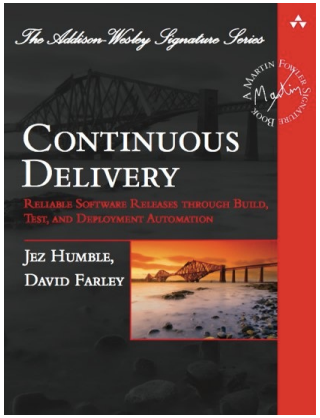


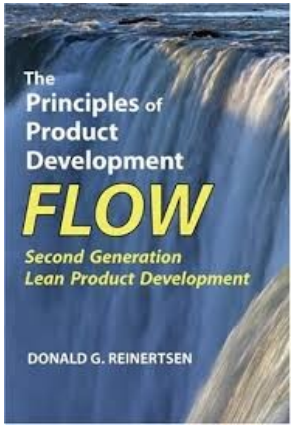


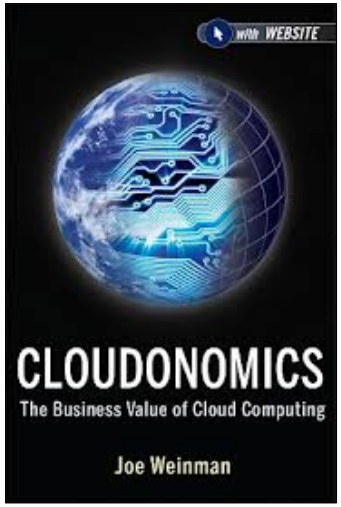












State of the Art in Web Scale Microservice Architectures

NETFLIX | OSS

AWS Re:Invent : Asgard to Zuul https://www.youtube.com/watch?v=p7ysHhs5hl0
Resiliency at Massive Scale https://www.youtube.com/watch?v=ZfYJHtVL1_w
Microservice Architecture https://www.youtube.com/watch?v=CriDUYtfrjs

GILT

http://www.infoq.com/presentations/scale-gilt

GROUPON

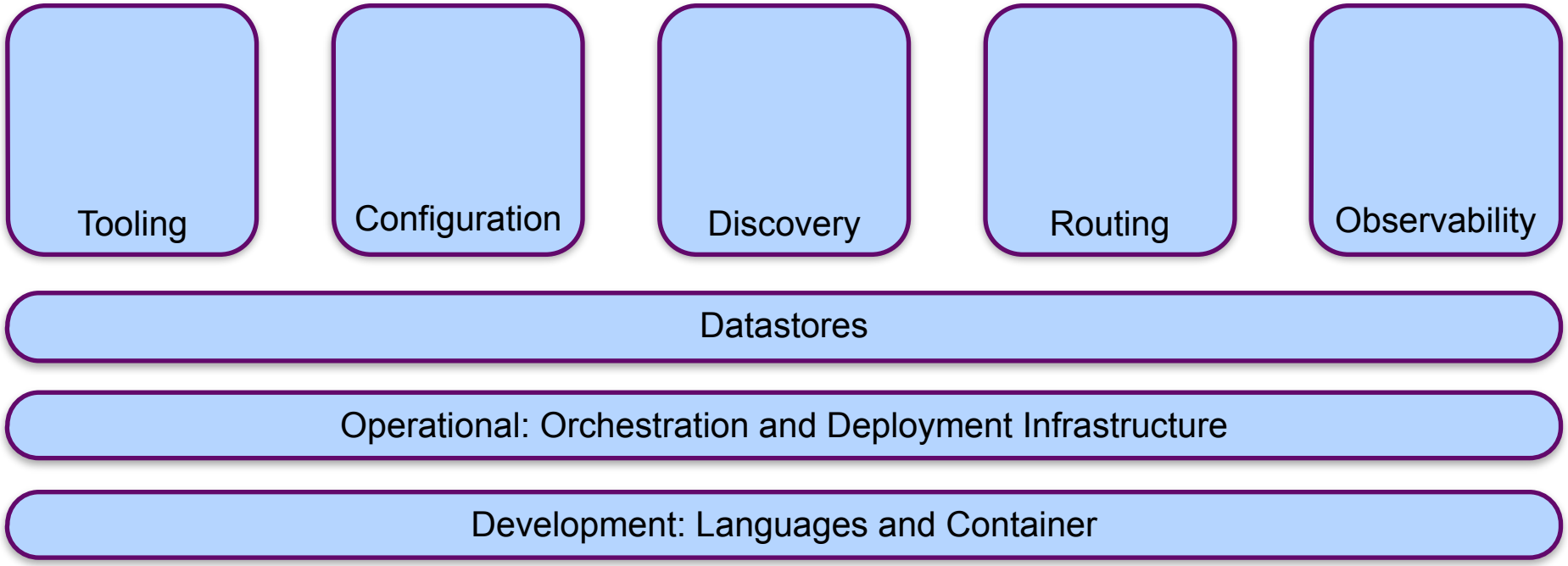
http://www.slideshare.net/mcculloughsean/itier-breaking-up-the-monolith-philly-ete

http://www.infoq.com/presentations/Twitter-Timeline-Scalability
http://www.infoq.com/presentations/twitter-soa
http://www.infoq.com/presentations/Zipkin

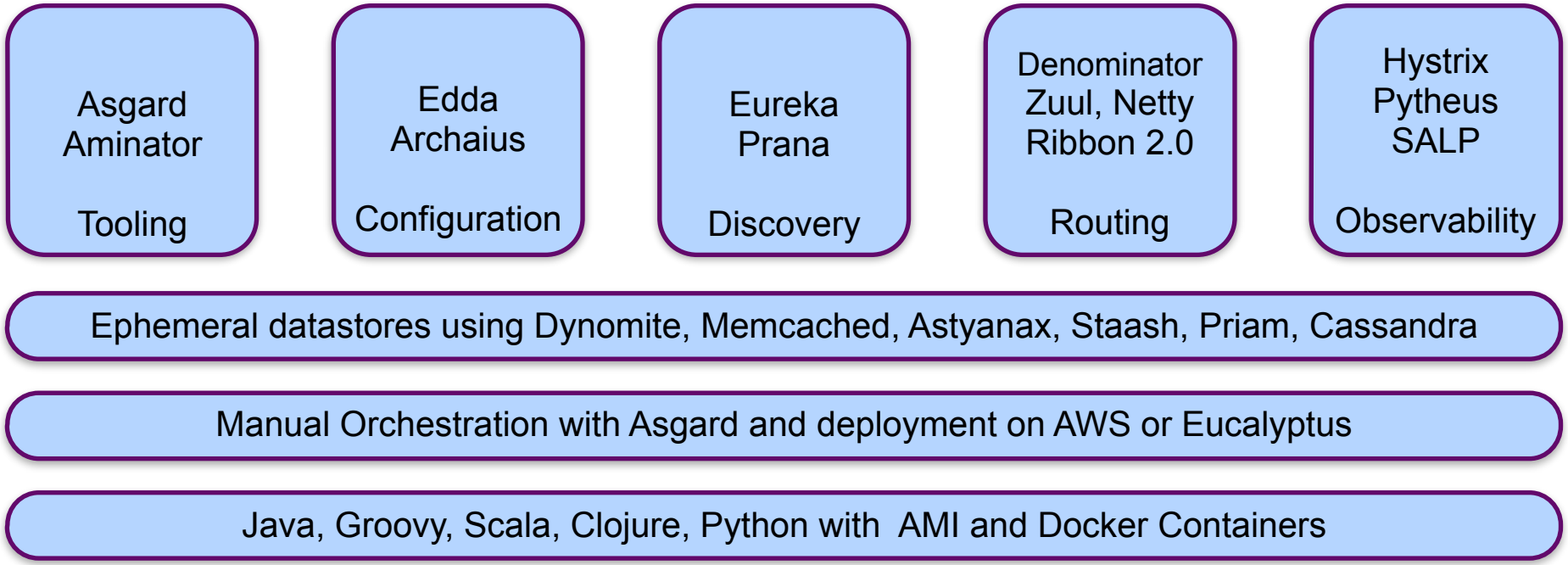
HAILO™

https://speakerdeck.com/mattheath/scaling-micro-services-in-go-highload-plus-plus-2014

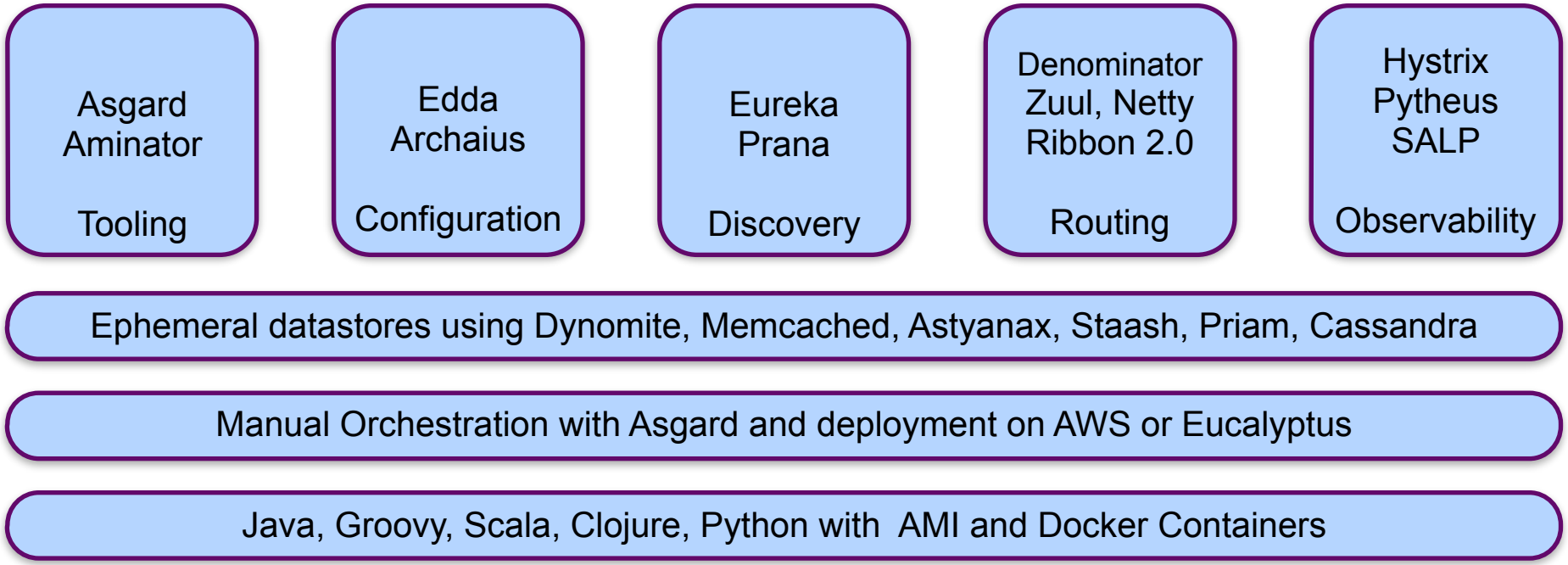
Microservice Concerns



NETFLIX | OSS Microservices



NETFLIX | OSS Microservices

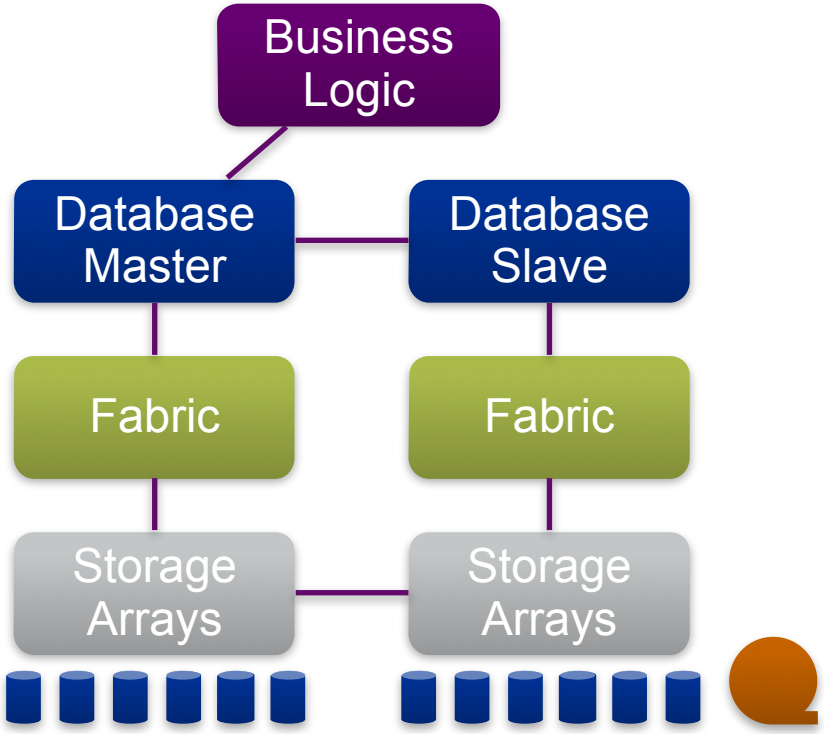


Focus on global distribution, high scale and availability

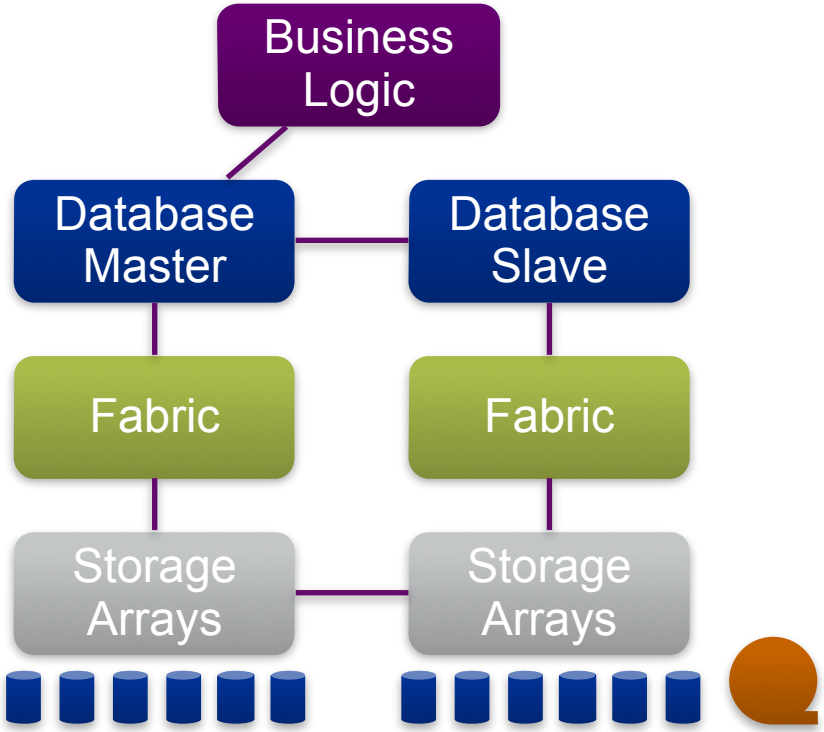
NETFLIX | OSS High Availability Patterns

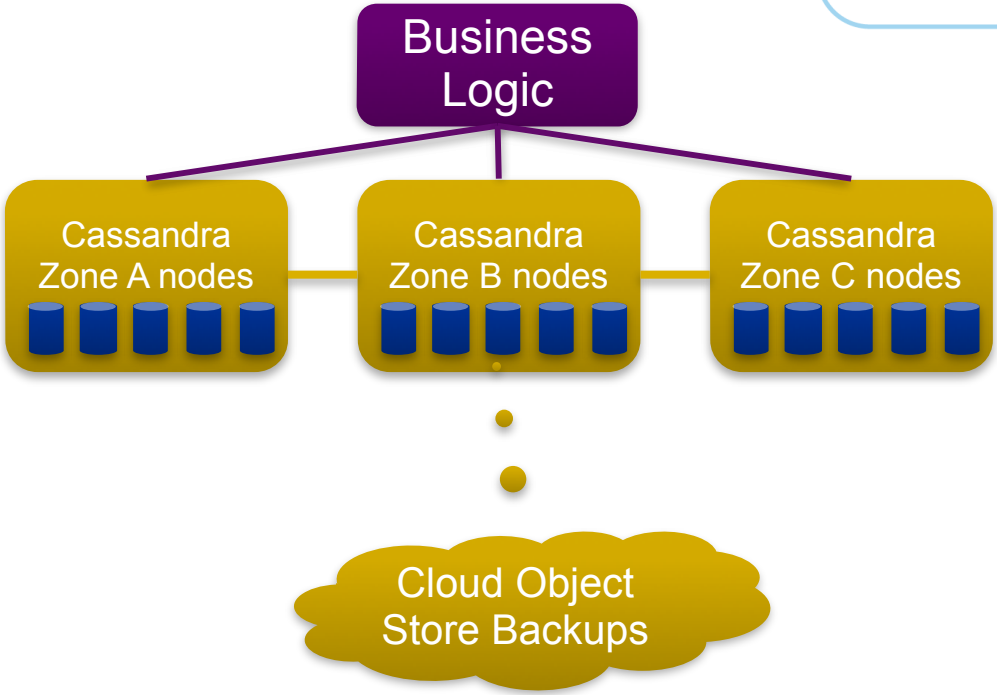
- *Business logic isolation in stateless micro-services*
- *Immutable code with instant rollback*
- *Auto-scaled capacity and deployment updates*
- *Distributed across availability zones and regions*
- *De-normalized single function NoSQL data stores*
- *See over 40 NetflixOSS projects at netflix.github.com*
- *Get "Technical Indigestion" trying to keep up with techblog.netflix.com*

Cloud Native Storage

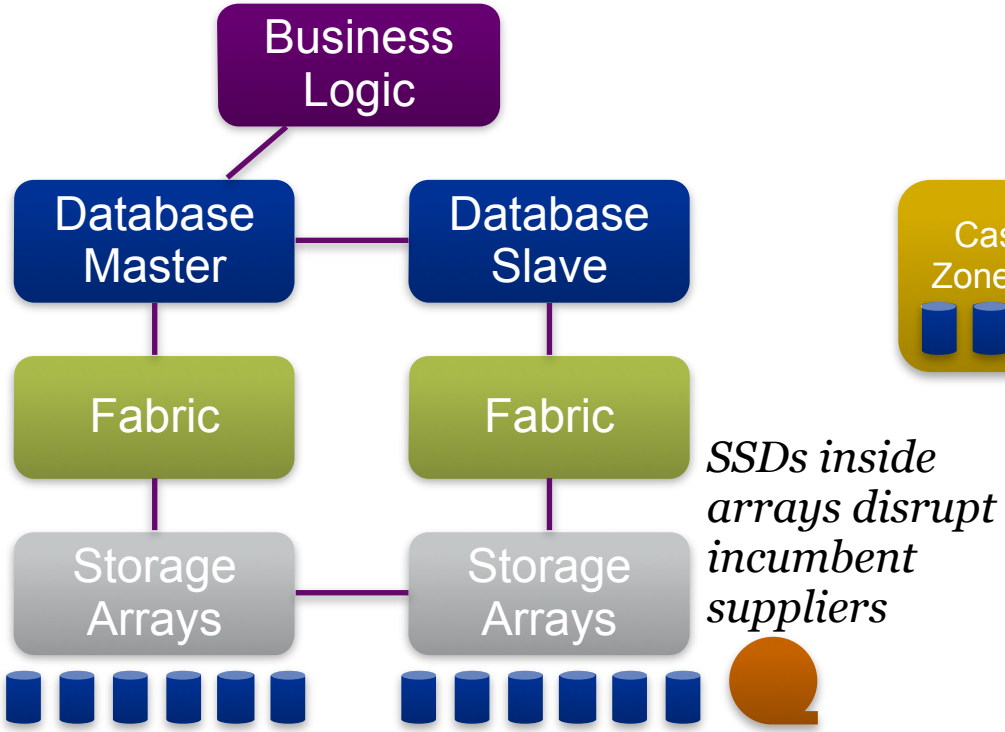


Cloud Native Storage

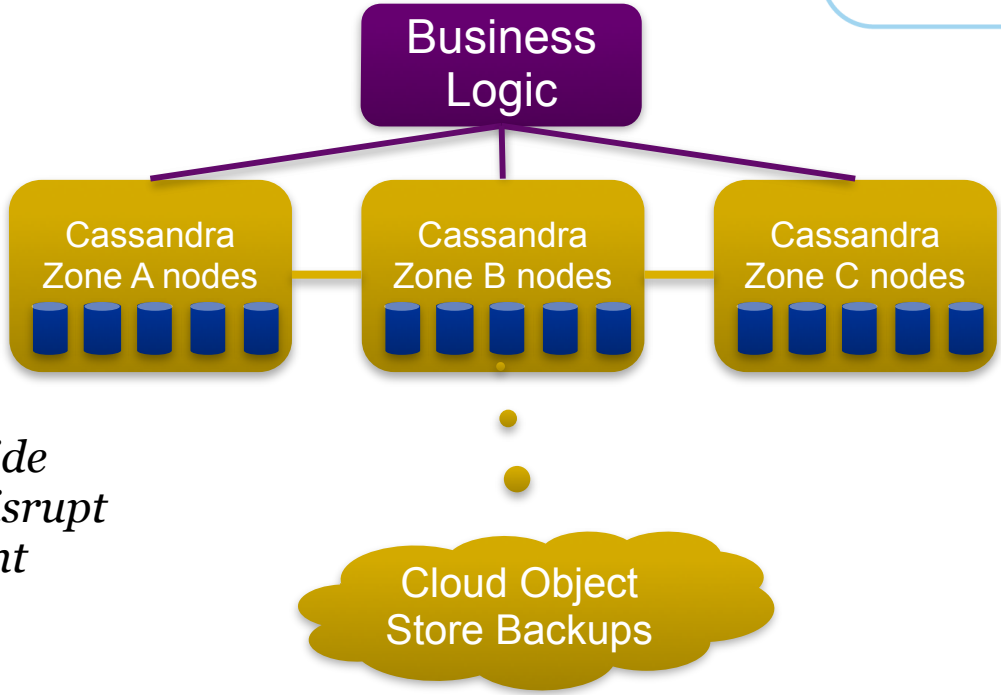




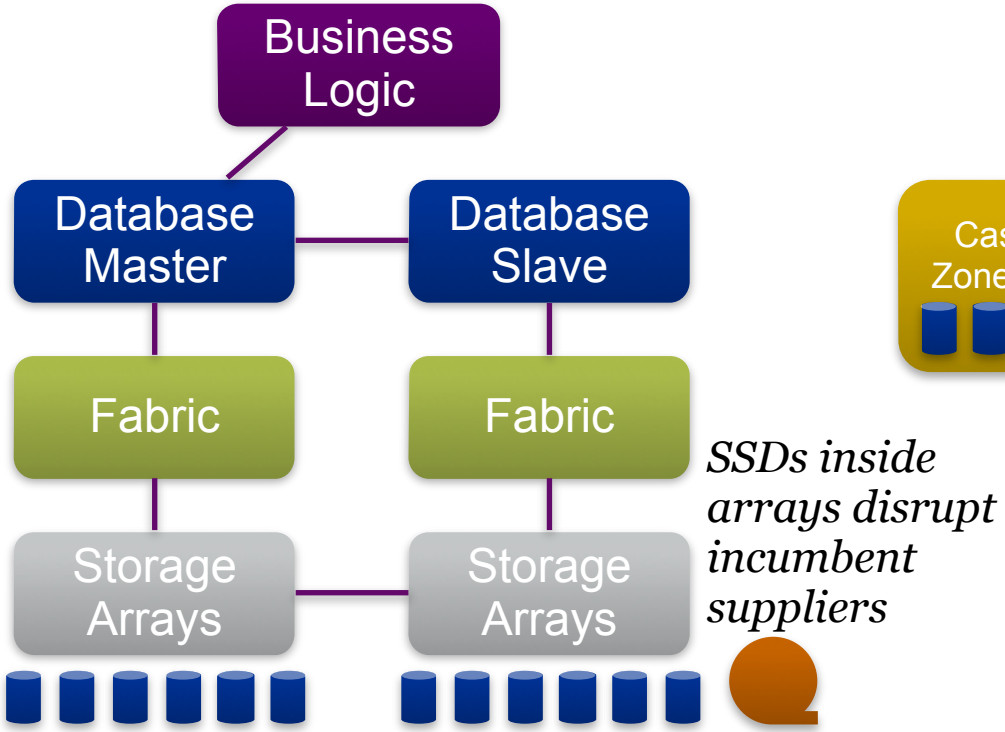
Cloud Native Storage



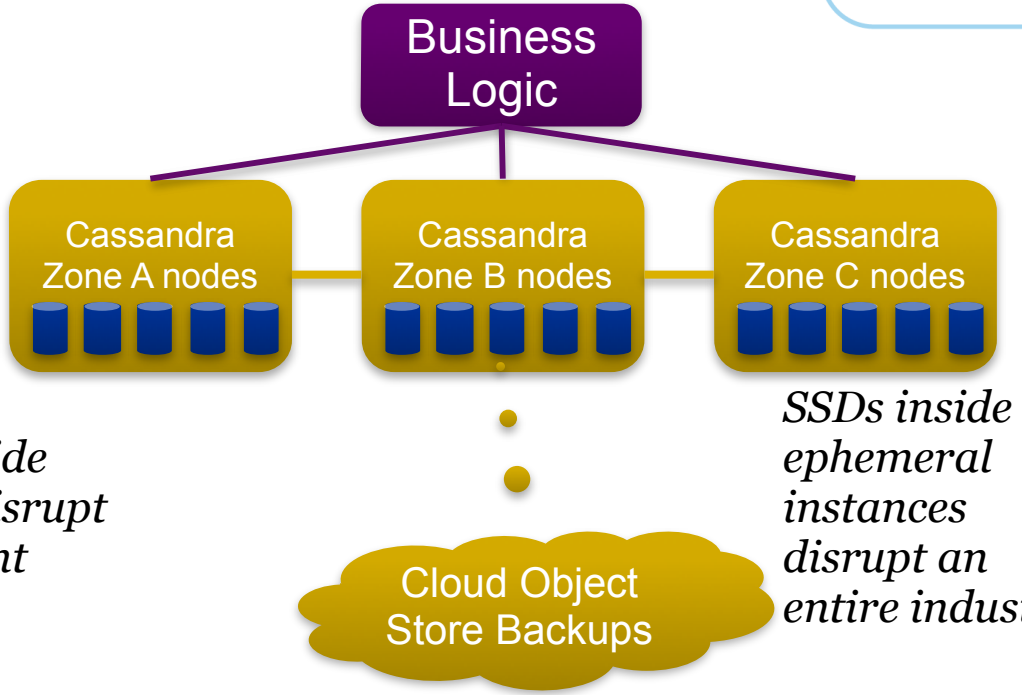
SSDs inside arrays disrupt incumbent suppliers



Cloud Native Storage

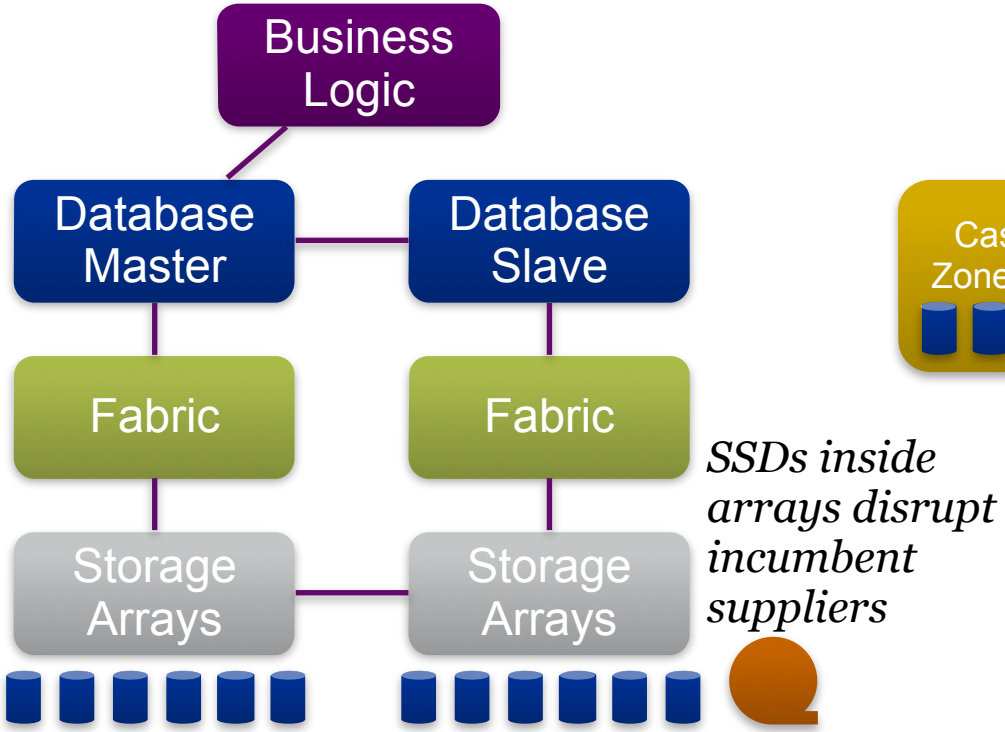


SSDs inside arrays disrupt incumbent suppliers

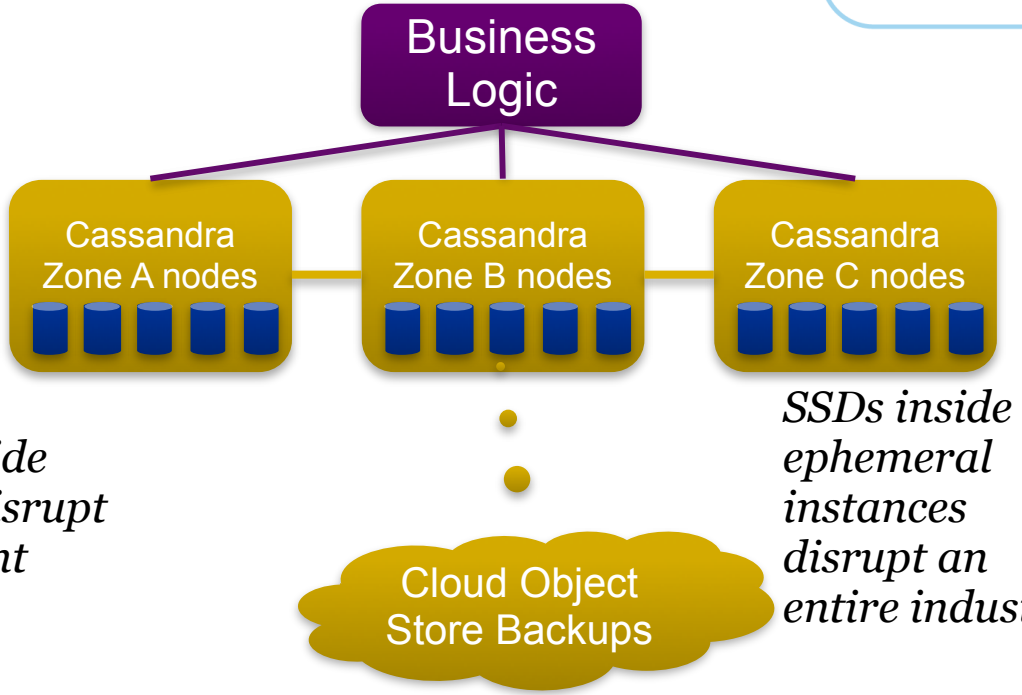


SSDs inside ephemeral instances disrupt an entire industry

Cloud Native Storage



SSDs inside arrays disrupt incumbent suppliers



SSDs inside ephemeral instances disrupt an entire industry

NetflixOSS Uses Priam to create Cassandra clusters in minutes

NETFLIX | OSS Trust with Verification

- *Edda - the "black box flight recorder" for configuration state*
- *Chaos Monkey - enforcing stateless business logic*
- *Chaos Gorilla - enforcing zone isolation/replication*
- *Chaos Kong - enforcing region isolation/replication*
- *Security Monkey - watching for insecure configuration settings*
- *See over 40 NetflixOSS projects at netflix.github.com*
- *Get "Technical Indigestion" trying to keep up with techblog.netflix.com*

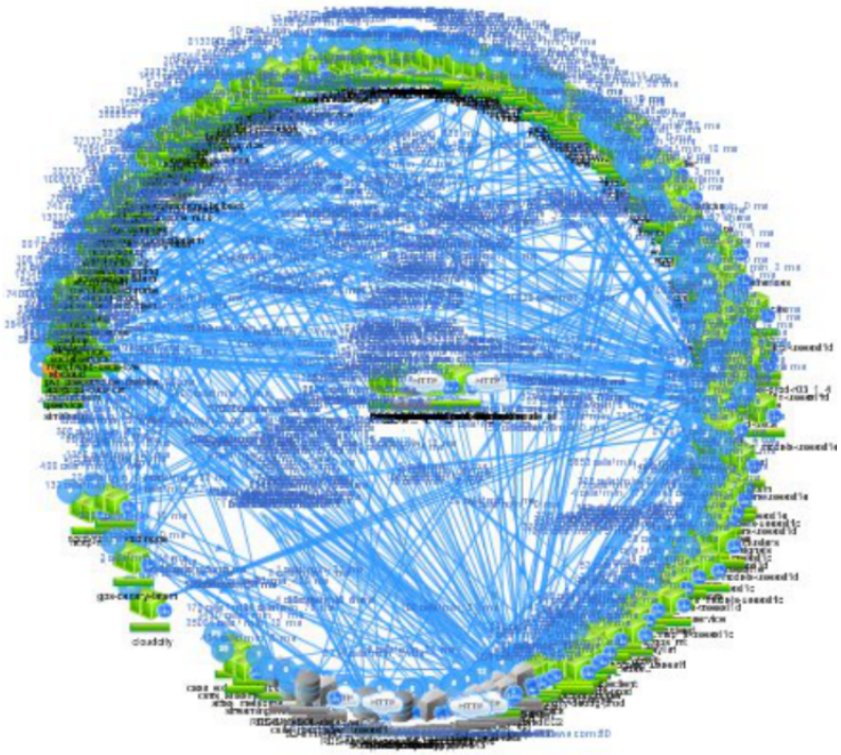
NETFLIX | OSS Netflix Open Source Software Center

Repositories | Powered By NetflixOSS

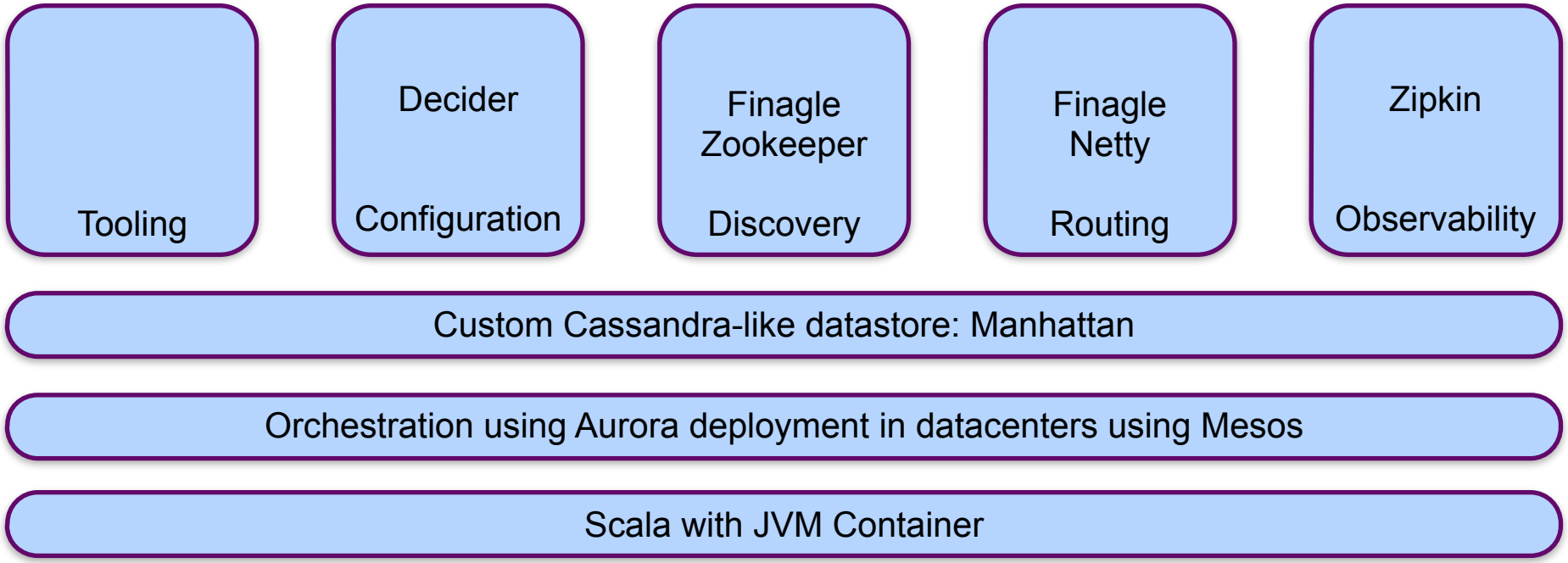
These companies are using and contributing to Netflix OSS Components

Email netflixoss@netflix.com to have your logo here.

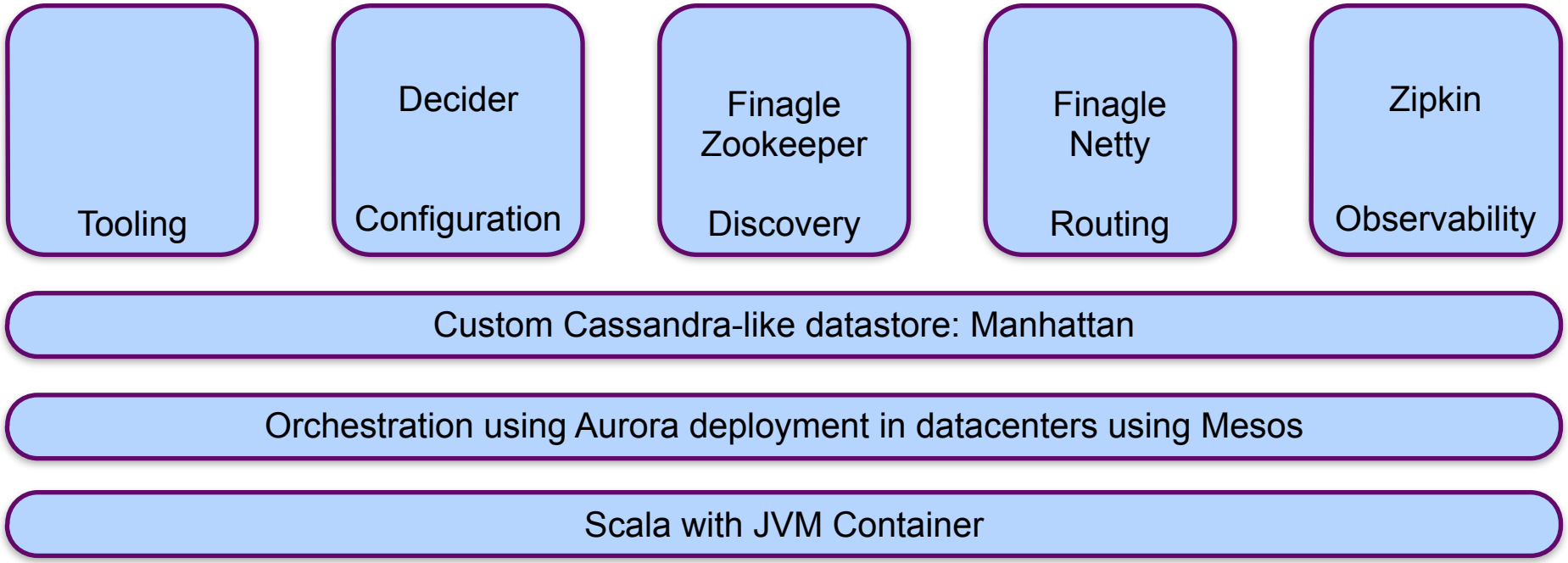
NETFLIX



Twitter Microservices



Twitter Microservices



Focus on efficient datacenter deployment at scale

Gilt Microservices

Ion Cannon
SBT
Rake

Tooling

Decider

Configuration

Finagle
Zookeeper

Discovery

Akka
Finagle
Netty

Routing

Zipkin

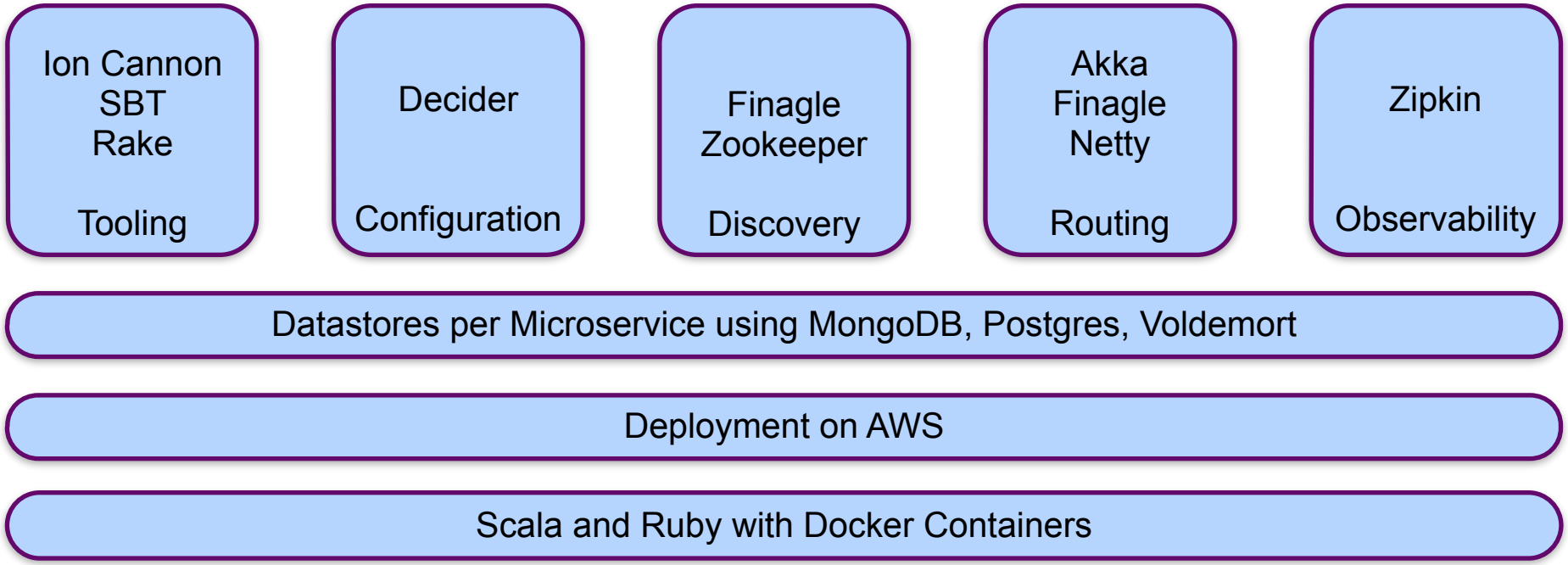
Observability

Datastores per Microservice using MongoDB, Postgres, Voldemort

Deployment on AWS

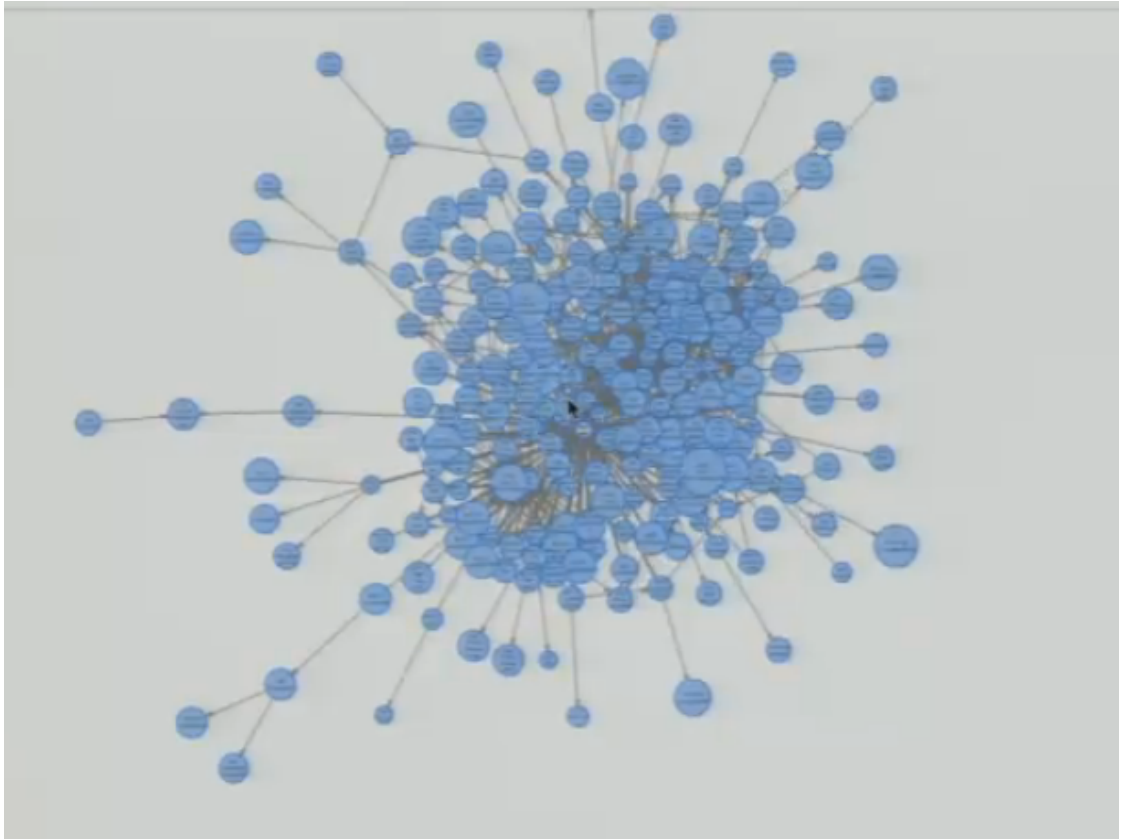
Scala and Ruby with Docker Containers

Gilt Microservices



Focus on fast development with Scala and Docker

GILT



Hailo Microservices

Hubot
Janky
Jenkins

Tooling

Configuration

go-platform

Discovery

go-platform
RabbitMQ

Routing

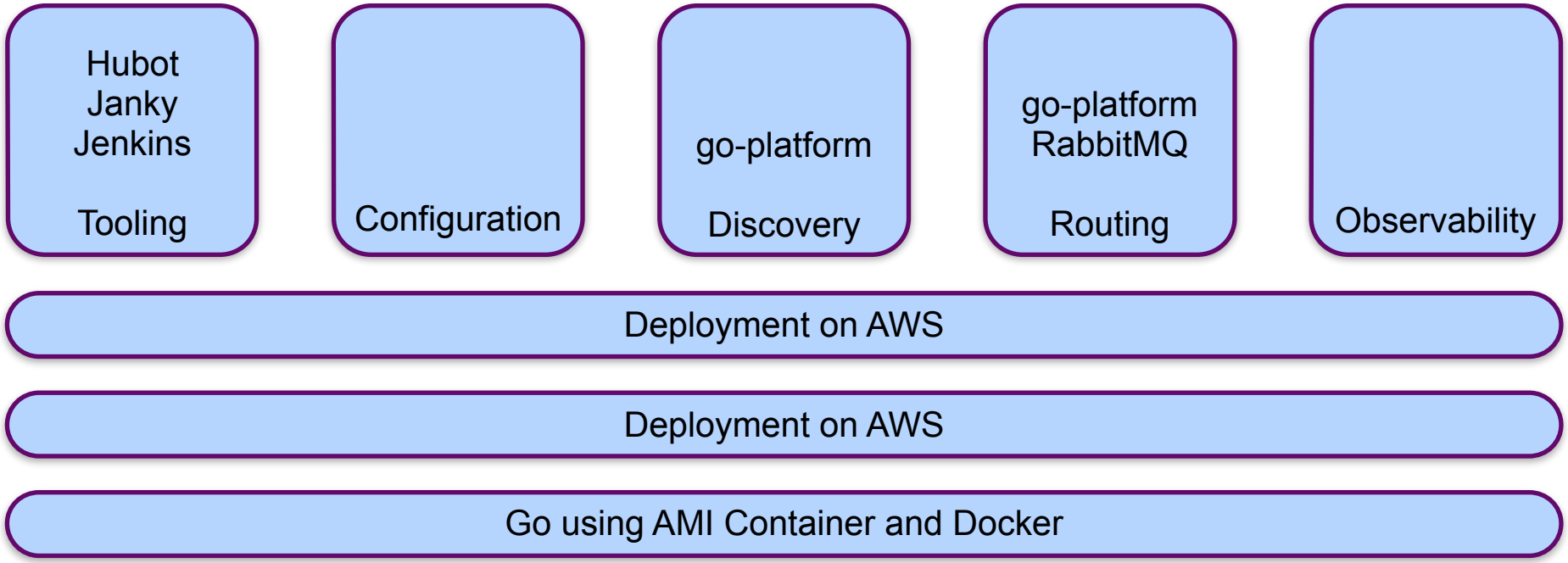
Observability

Deployment on AWS

Deployment on AWS

Go using AMI Container and Docker

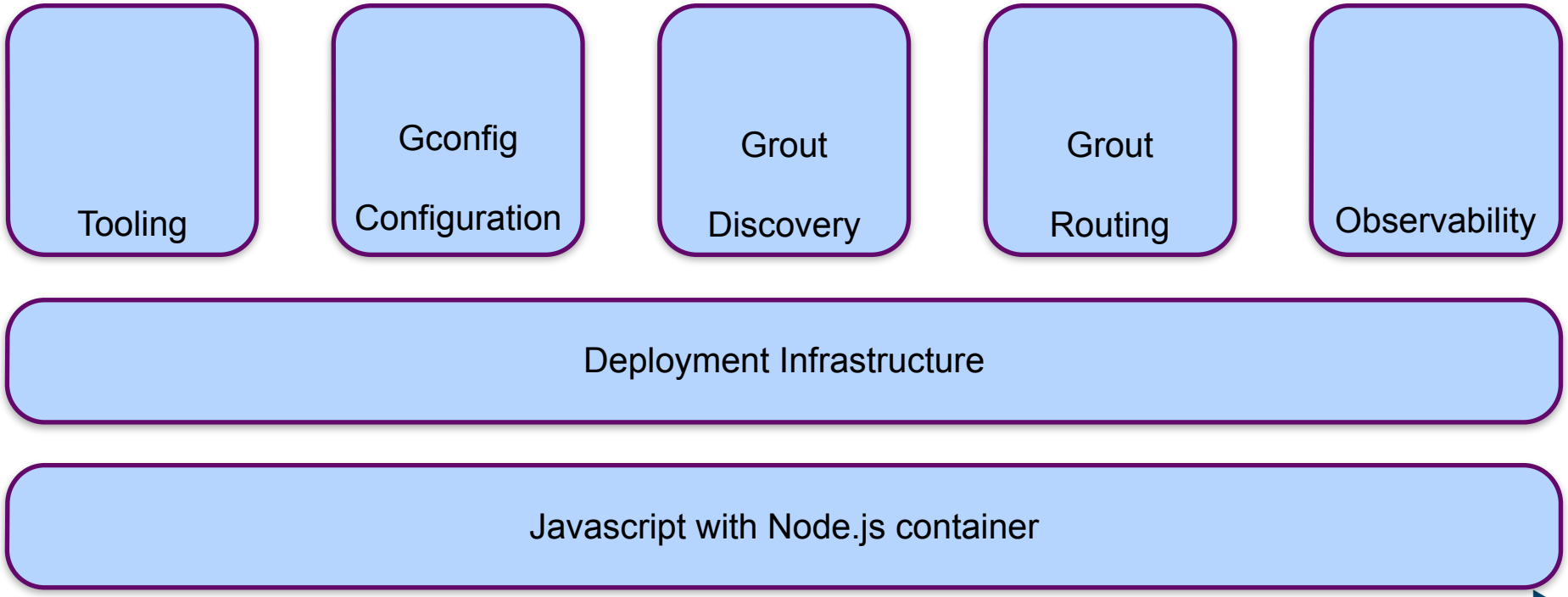
Hailo Microservices



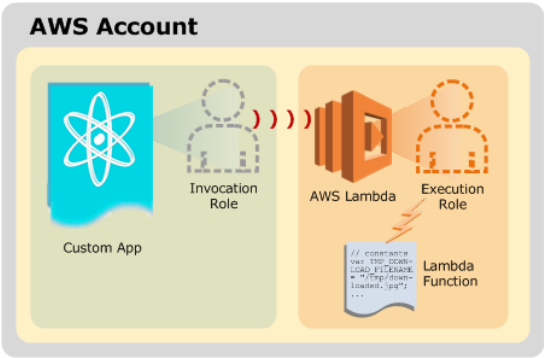
Also watching: https://github.com/peterbourgon/gokit

 api.v1.break
api.v1.city
api.v1.customer
api.v1.driver
api.v1.event
api.v1.experiment
api.v1.features
api.v1.feedback
api.v1.gamification
api.v1.geocode
api.v1.heatmap
api.v1.job
api.v1.point
api.v1.pwh
api.v1.quote
api.v1.rank
api.v1.shift
api.v1.stat.driver
api.v1.system
api.v1.track
api.v1.version
api.v2.sendgrid
hailo-2-api
infrastructure.compute
infrastructure.whisper
kernel.binding
kernel.discovery
kernel.provisioning
service.account
service.adhoc-payment
service.allocation
service.app-feedback
service.app-version
service.archived-job
service.b2c-events
service.b2c-profile
service.banning
service.beta-customer
service.billing-shim
service.cancellation
service.card-gateway
service.charging
service.city
service.comms
service.config
service.config-updater
service.corporate-account
service.credit
service.crm-preference
service.customer
service.developer
service.dodgy-job
service.driver
service.driver-augmenter
service.driver-license
service.driver-migration
service.driver-organization
service.driver-scoring
service.email
service.eta
service.experiment
service.fare
service.feature-flags
service.hob
service.hob-organization
service.hob-reporting
service.hob-transaction
service.hms
service.hob-activity
service.hob-editor
service.login
service.job
service.job-events
service.job-points
service.job-system-cancel
service.lead
service.log
service.login
service.loyalty
service.merchant
service.merchant-payment
service.message-federation
service.metrics
service.monitoring
service.nearest-driver
service.note
service.nsq-to-halloengine
service.organization
service.outage-impact
service.payment-validation
service.phone
service.platform-event-correlation
service.pricing
service.profile
service.profile-flow
service.provisioning-manager
service.public-routes
service.pwh
service.radar
service.redshift-import
service.routing
service.zoning

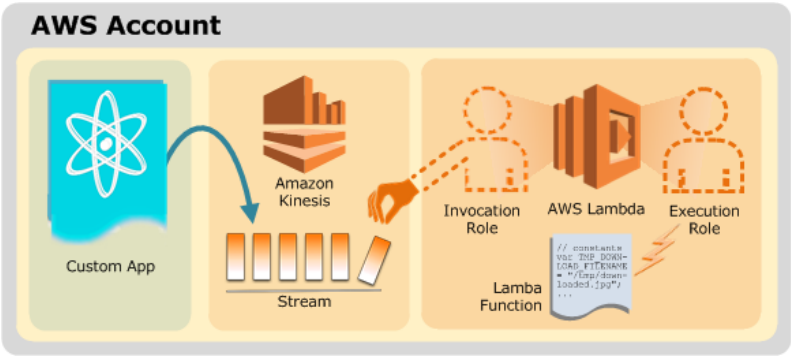
GROUPON Microservices

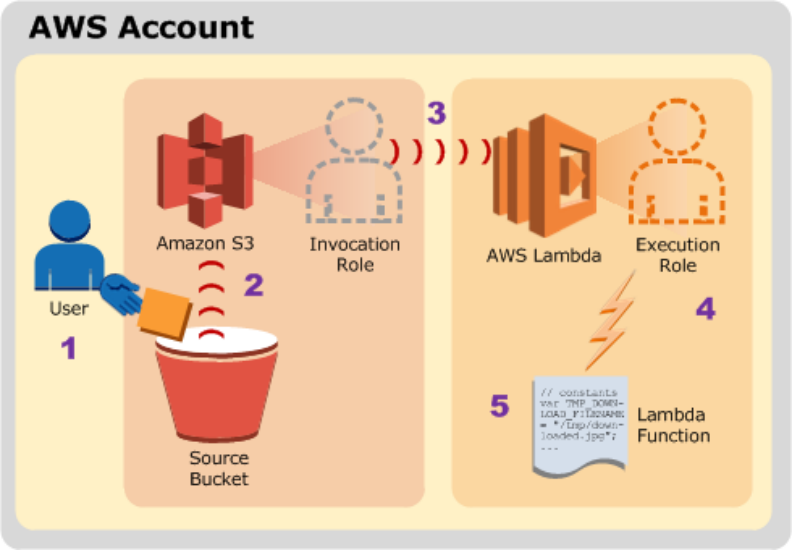


With AWS Lambda compute resources are charged by the 100ms, not the hour

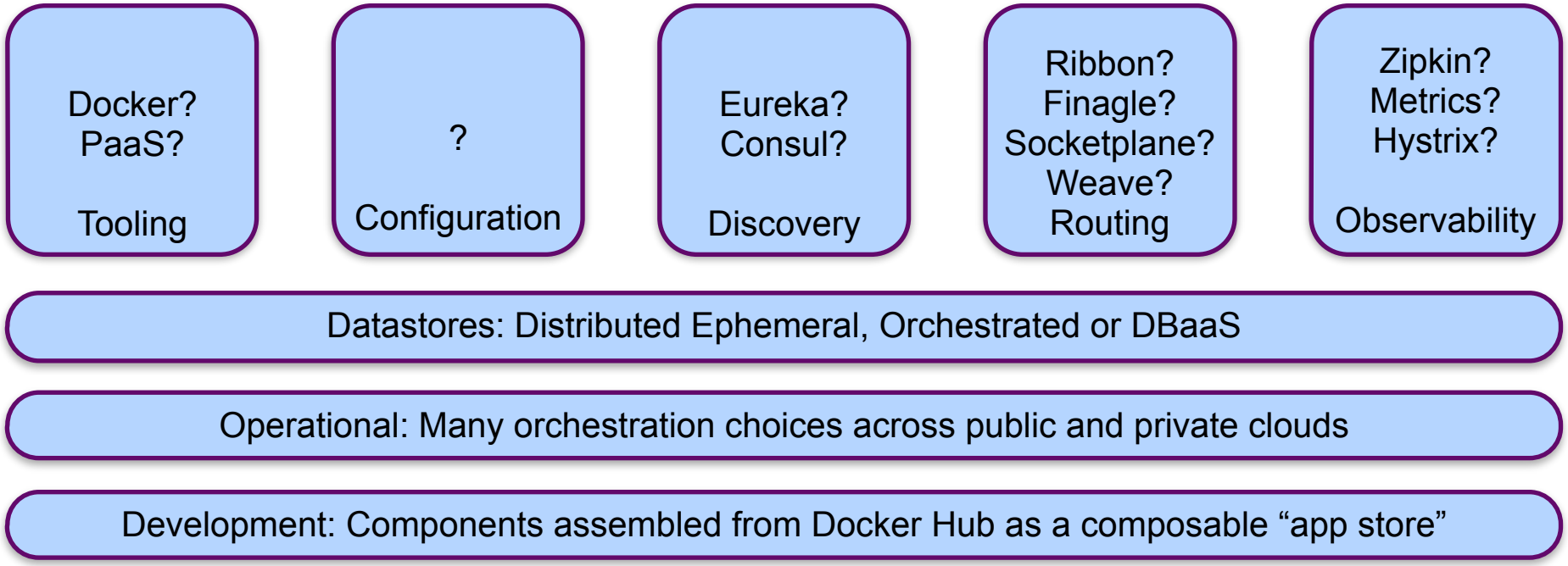


First 1M node.js executions/month are free

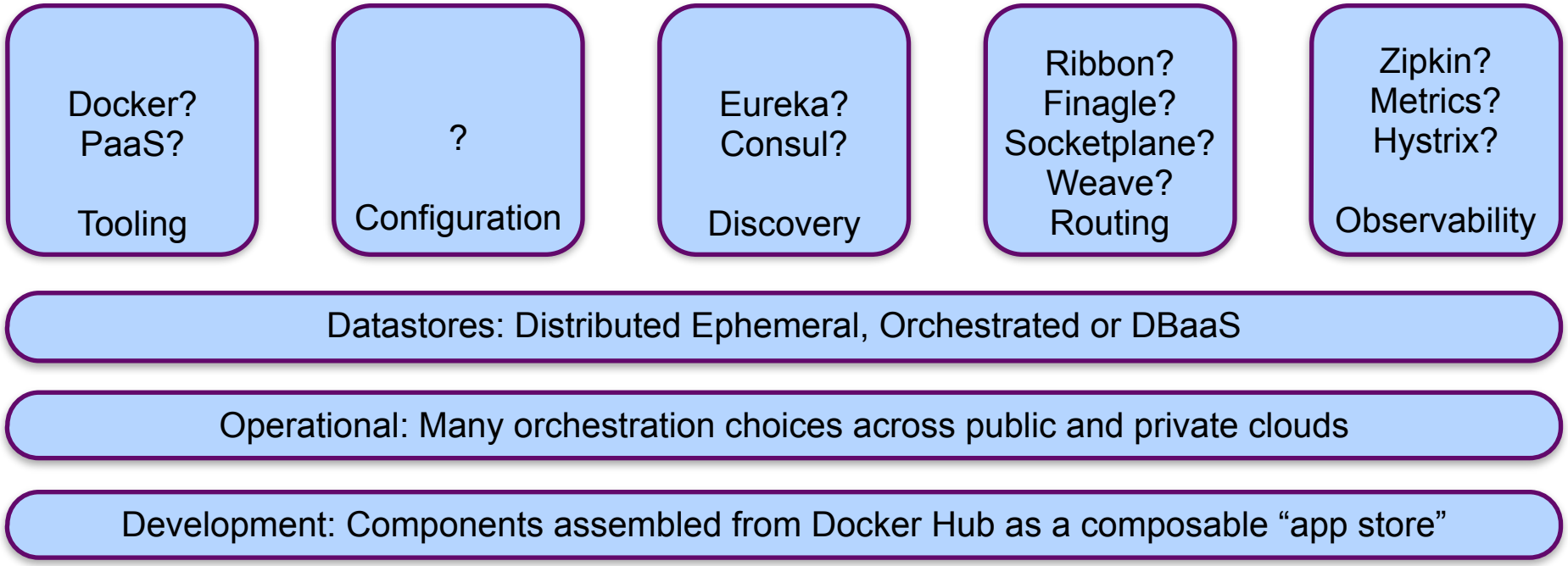




Next Generation Applications



Next Generation Applications



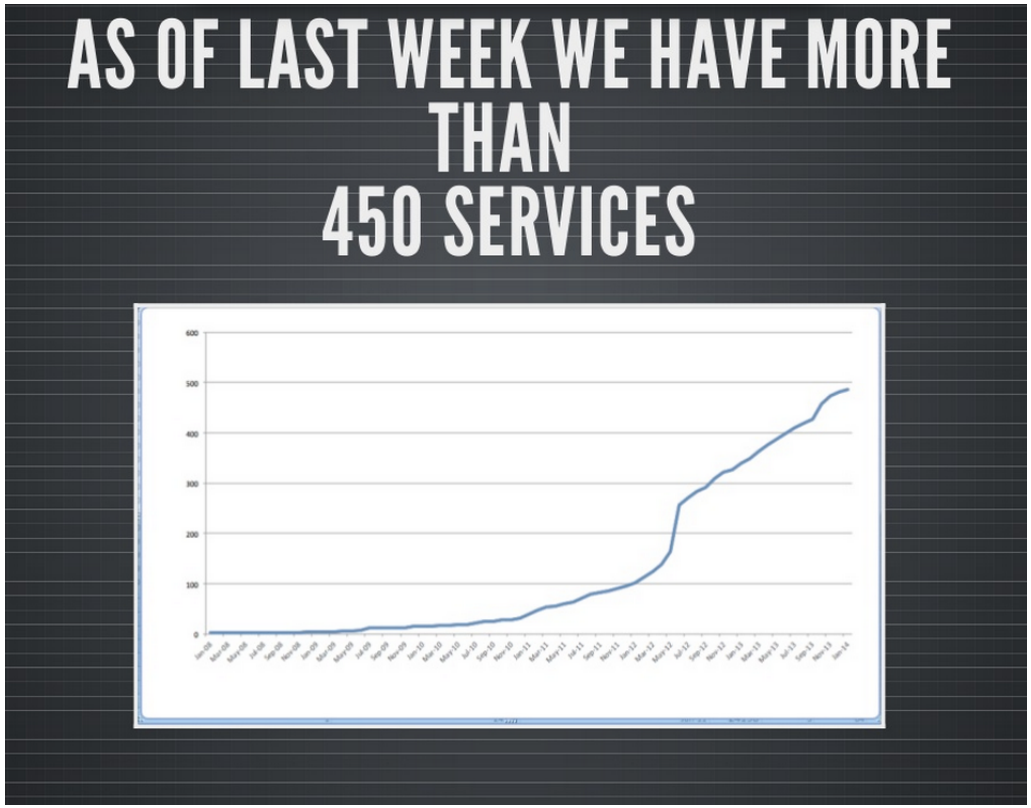
Fill in the gaps, rapidly evolving ecosystem choices

Cloud Native Monitoring and Microservices

Cloud Native Microservices

- *High rate of change*
  - *Code pushes can cause floods of new instances and metrics*
  - *Short baseline for alert threshold analysis – everything looks unusual*
- *Ephemeral Configurations*
  - *Short lifetimes make it hard to aggregate historical views*
  - *Hand tweaked monitoring tools take too much work to keep running*
- *Microservices with complex calling patterns*
  - *End-to-end request flow measurements are very important*
  - *Request flow visualizations get overwhelmed*

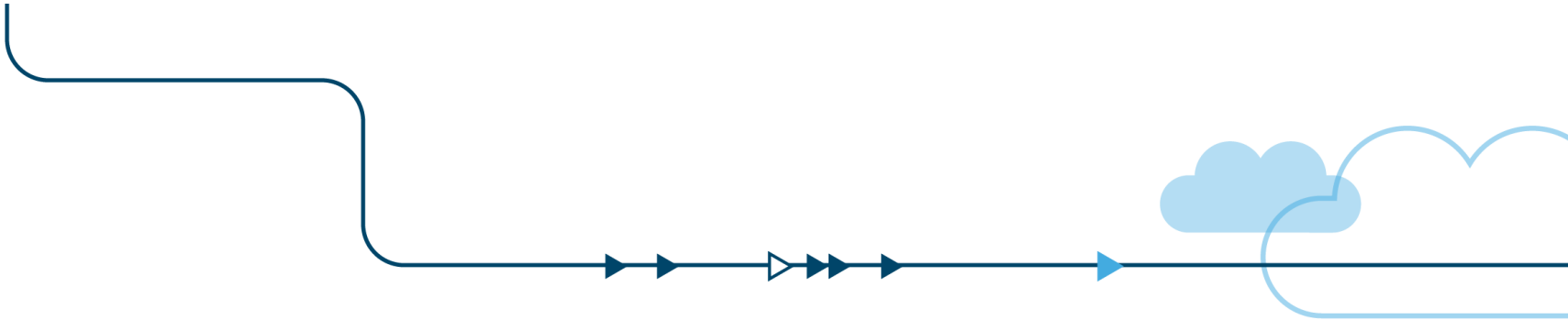
Microservice Based Architectures



See http://www.slideshare.net/LappleApple/gilt-from-monolith-ruby-app-to-micro-service-scala-service-architecture

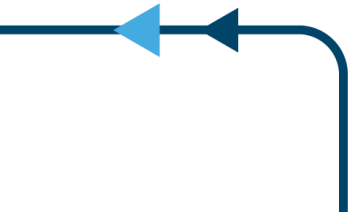
Continuous Delivery and DevOps

- *Changes are smaller but more frequent*
- *Individual changes are more likely to be broken*
- *Changes are normally deployed by developers*
- *Feature flags are used to enable new code*
- *Instant detection and rollback matters much more*

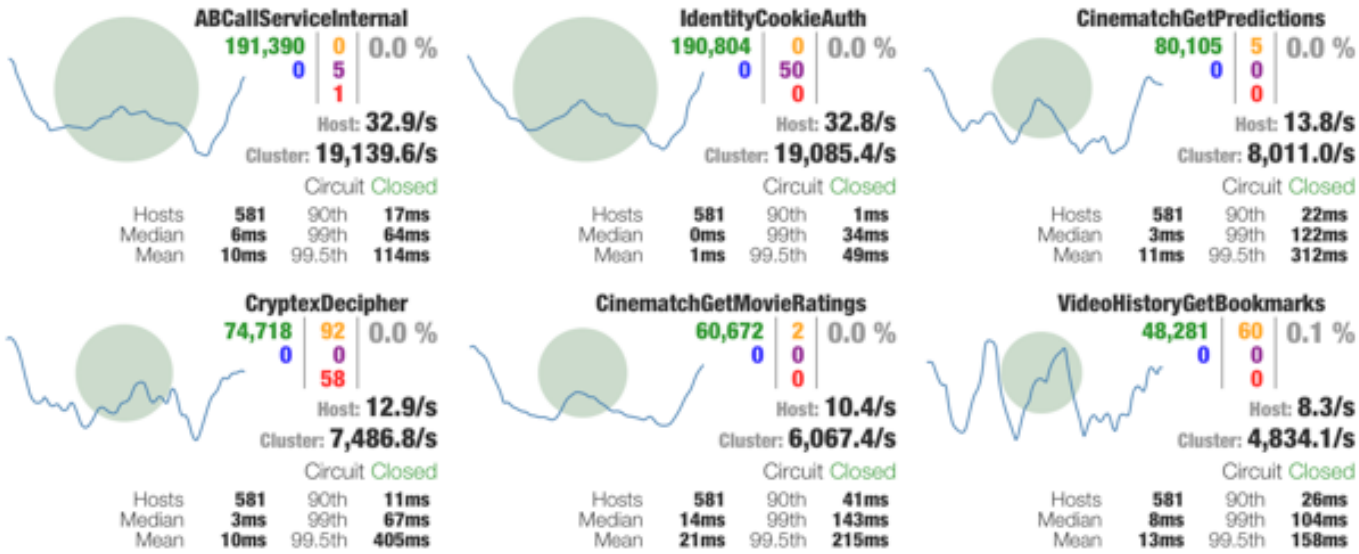


Whoops! I didn't mean that! Reverting…

Not cool if it takes 5 minutes to see it failed and 5 more to see a fix
No-one notices if it only takes 5 seconds to detect and 5 to see a fix

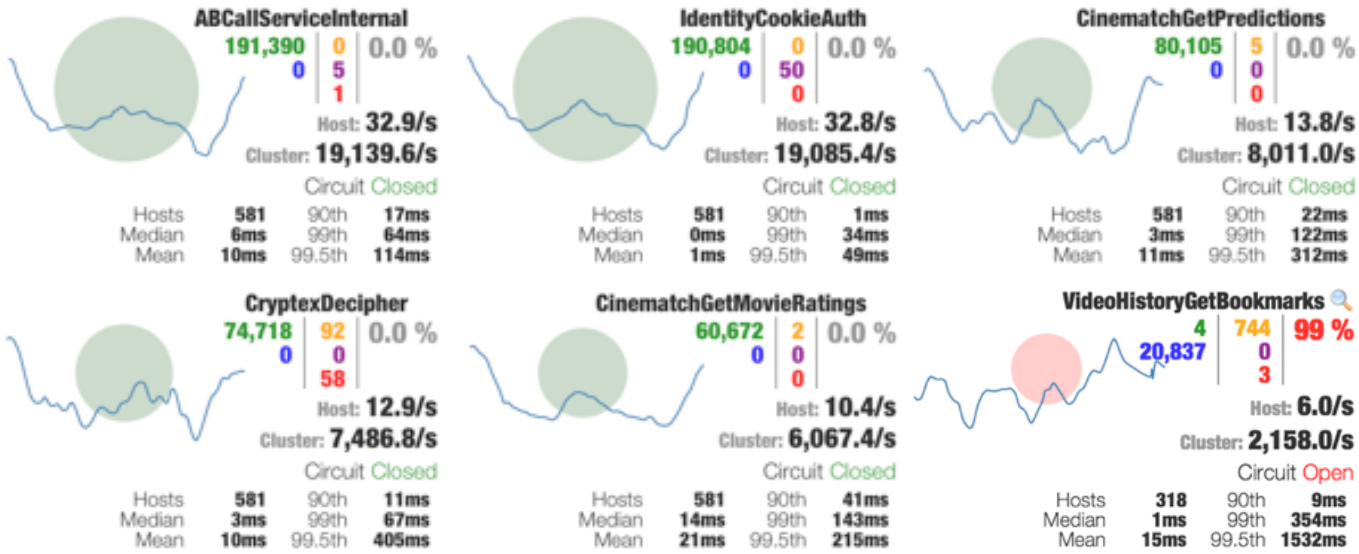


NetflixOSS Hystrix/Turbine Circuit Breaker



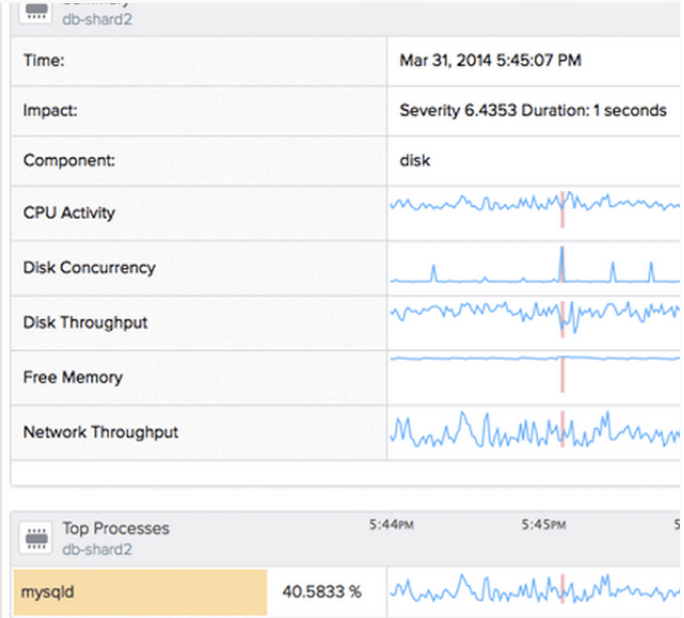
http://techblog.netflix.com/2012/12/hystrix-dashboard-and-turbine.html

NetflixOSS Hystrix/Turbine Circuit Breaker



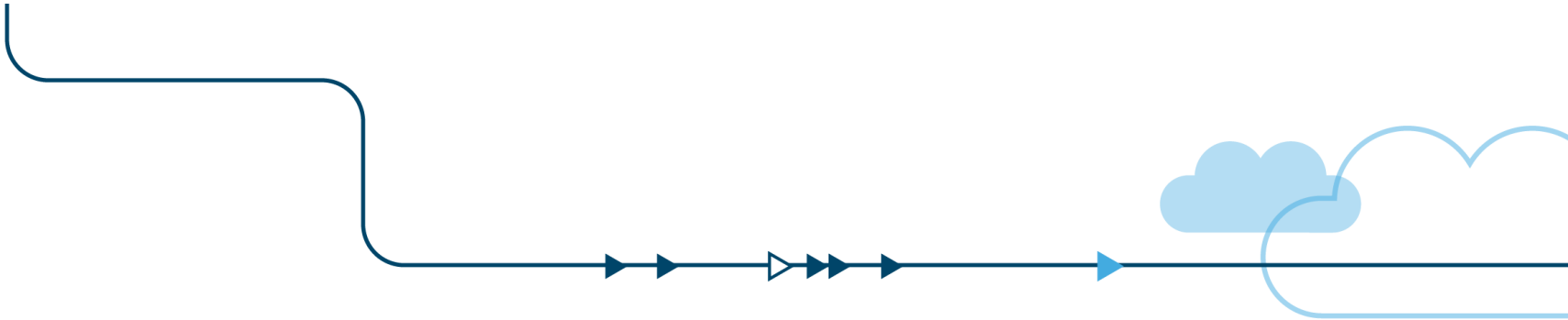
http://techblog.netflix.com/2012/12/hystrix-dashboard-and-turbine.html

Low Latency SaaS Based Monitors

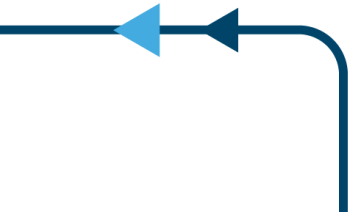


1-second data collection and real-time streaming processing on all components of the application stack

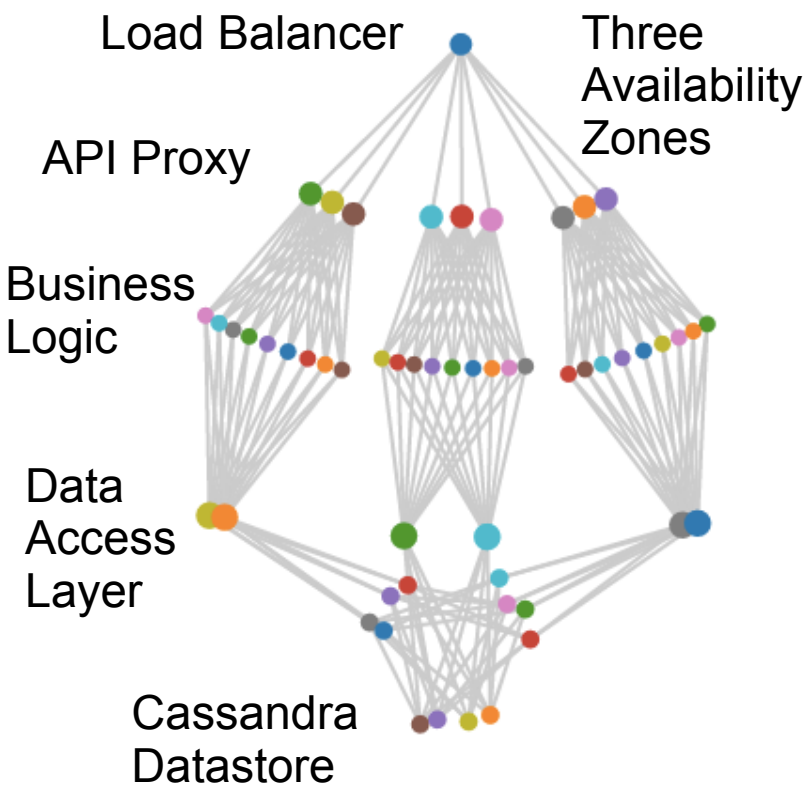
www.vividcortex.com and www.boundary.com



Metric to display latency needs to be less than human attention span (~10s)



Adrian's Tinkering Projects

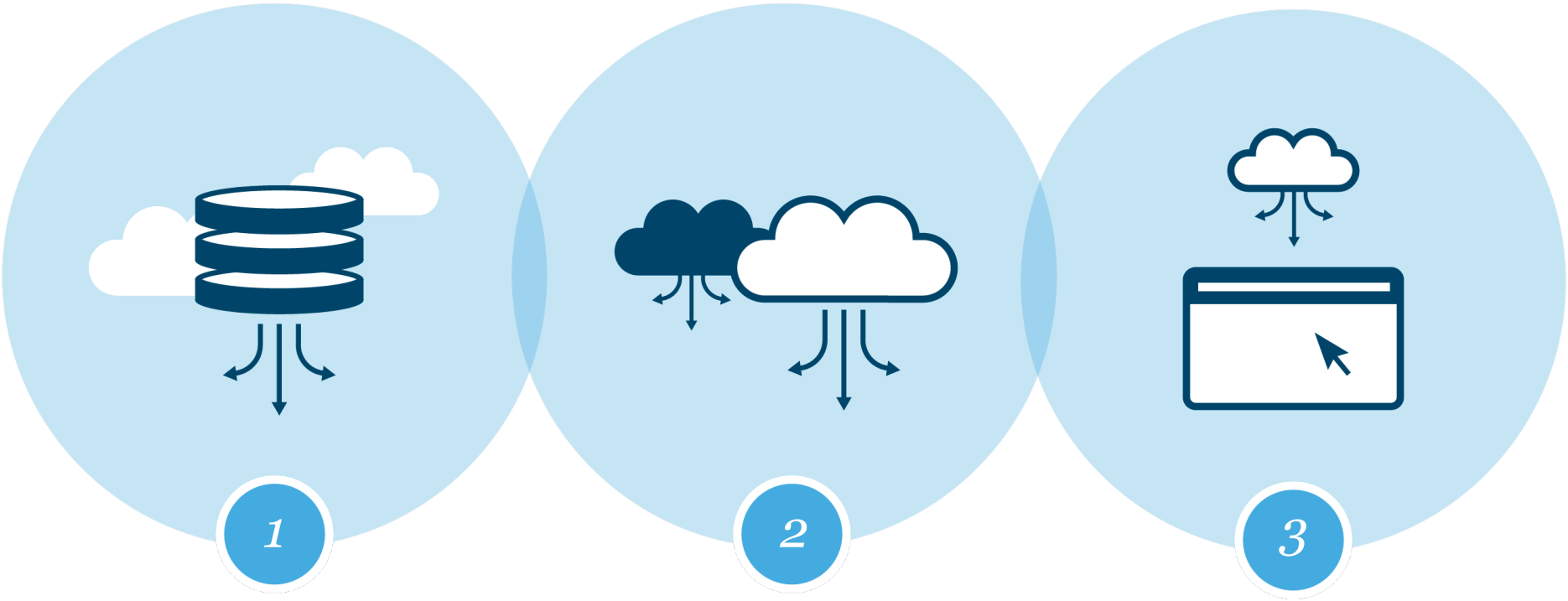


Model and visualize microservices
Simulate interesting architectures
Generate large scale configurations
Eventually stress test real tools

See github.com/adrianco/spigo
Simulate Protocol Interactions in Go

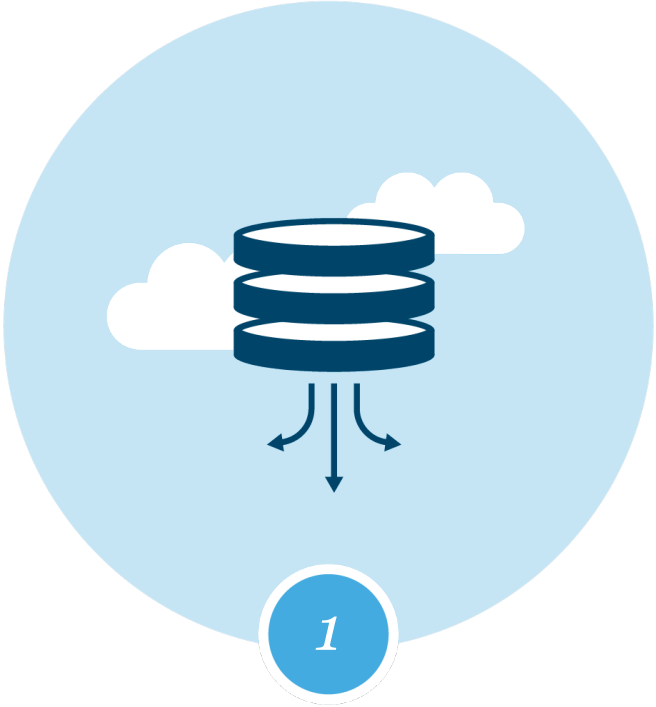
Cost Optimization

How is Cost Measured?



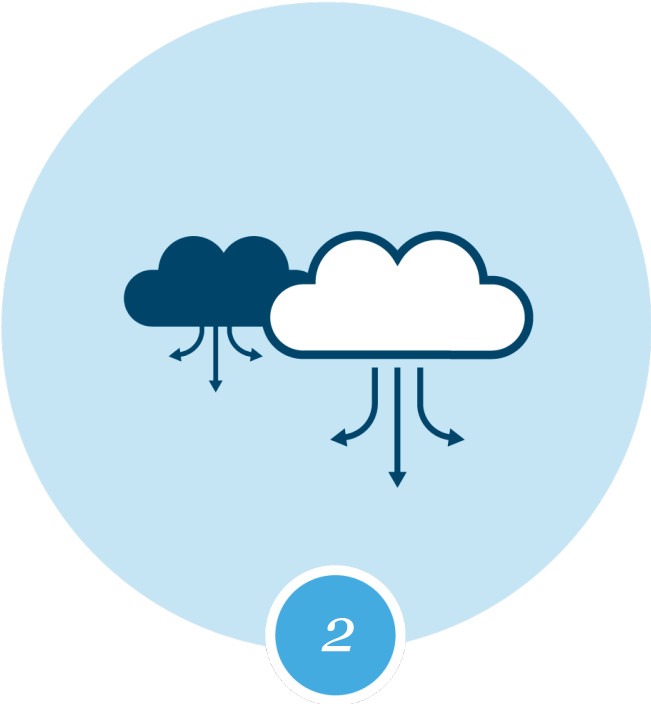
Bottom Up **Product** **Top Down**

Bottom Up Costs



Add up the cost to buy and operate every component

Product Cost

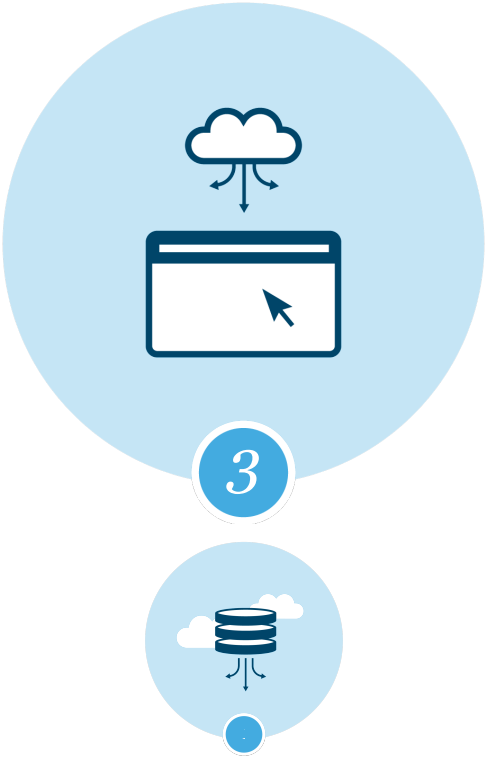


Cost of delivering and maintaining each product

Top Down Costs

Divide total budget by the number of components

Top Down vs. Bottom Up



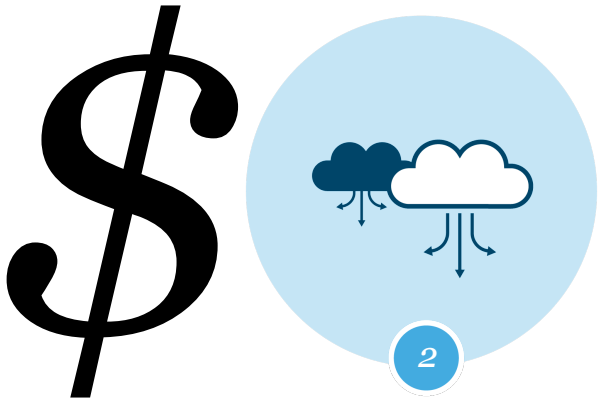
Will never match!

Hidden Subsidies vs. Hidden Costs

@adrianco

BV Battery Ventures

Agile Team Product Profit



Value minus costs

Time to value

ROI, NPV, MMF

Profit center

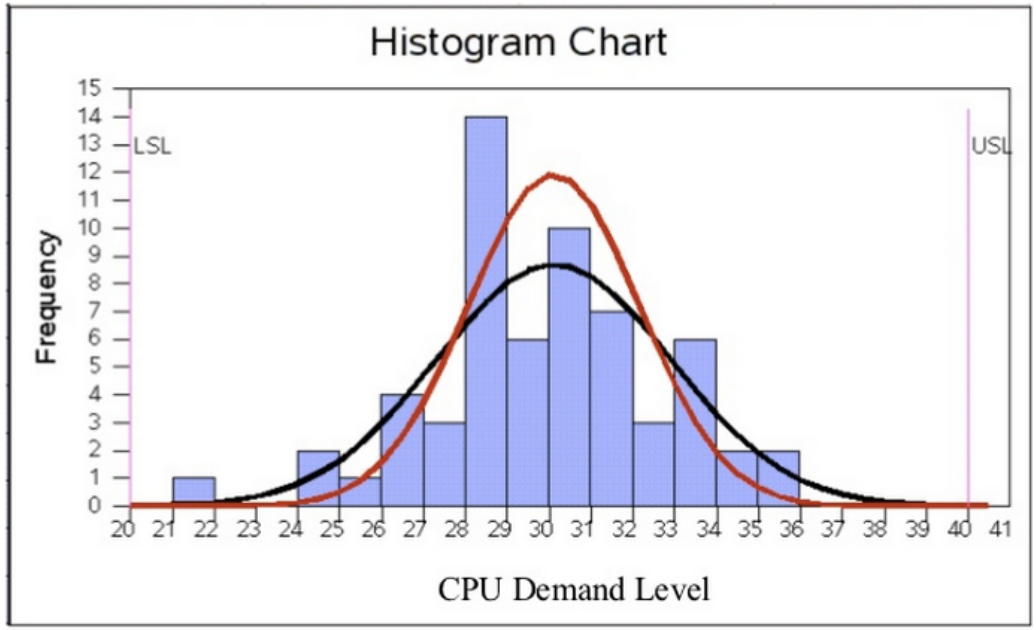
@adrianco

BV Battery Ventures

Capacity Optimization for a Single System Bottleneck

Lower Spec Limit

When demand probability is below USL by 3.0 sigma scale down resource to save money



Upper Spec Limit

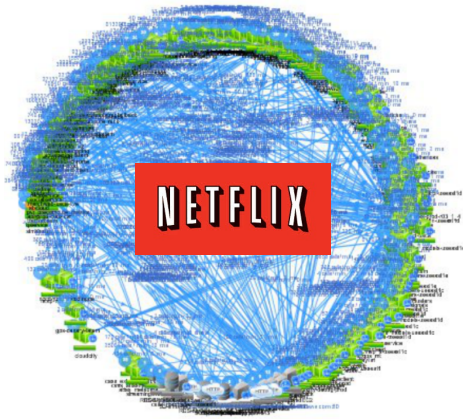
When demand probability exceeds USL by 4.0 sigma scale up resource to maintain low latency

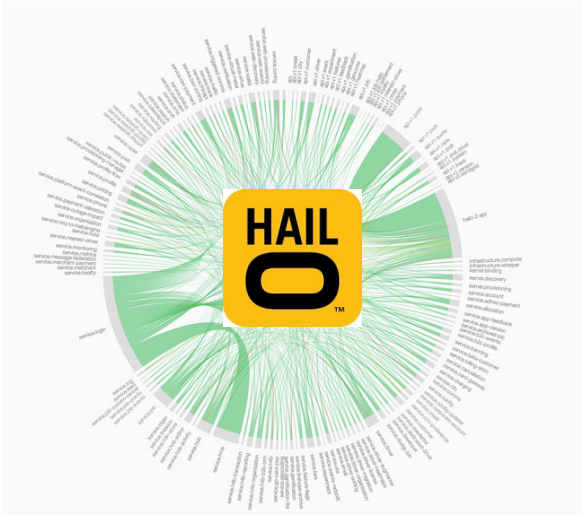
Documentation on Capability Plots

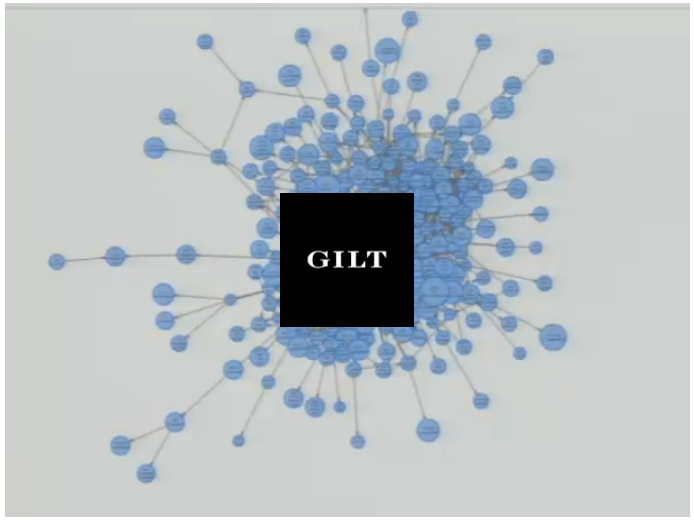
To get accurate high dynamic range histograms see http://hdrhistogram.org/

Slideshare: 2003 Presentation on Capacity Planning Methods See US Patent: 7467291

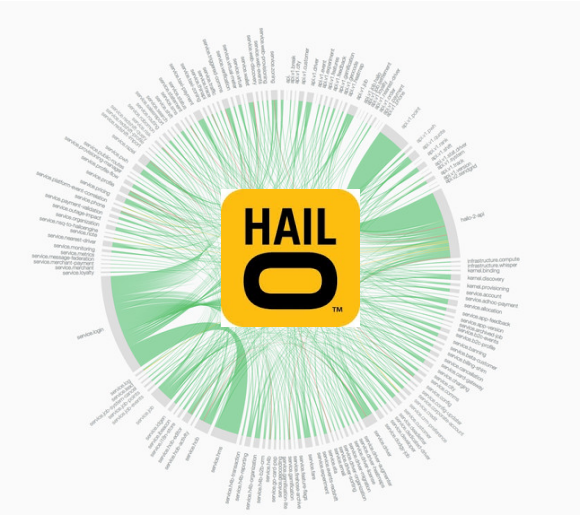
But interesting systems don't have a single bottleneck nowadays...

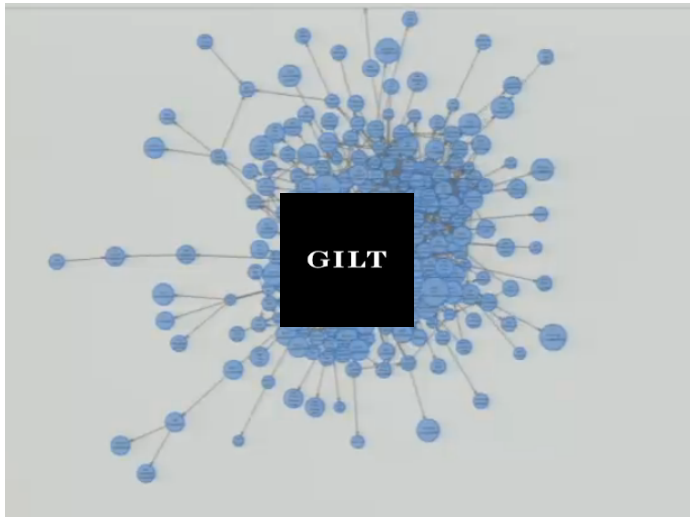


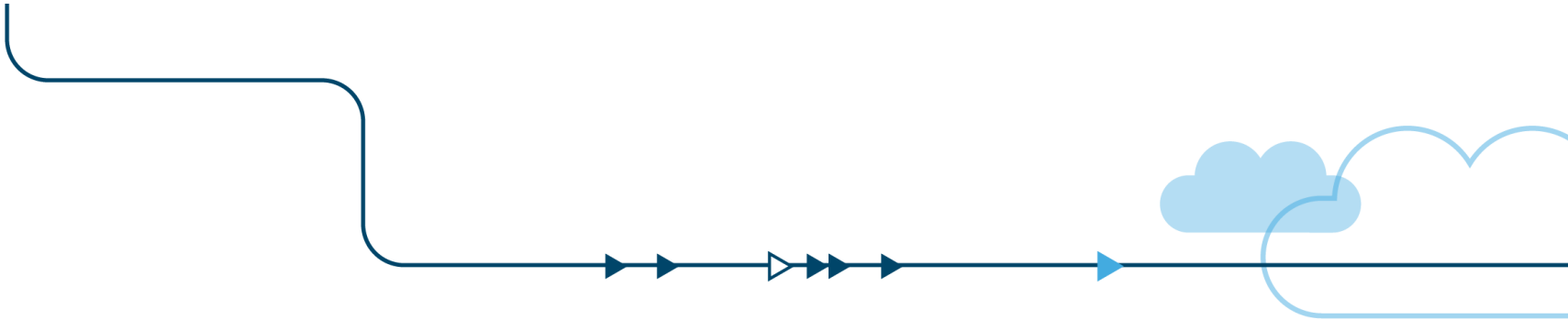




But interesting systems don't have a single bottleneck nowadays...

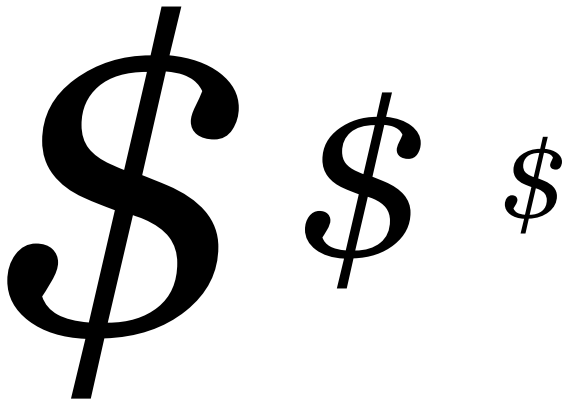






What about cloud costs?

Cloud Native Cost Optimization



Optimize for speed first

Turn it off!

Capacity on demand

Consolidate and Reserve

Plan for price cuts

FOSS tooling

@adrianco

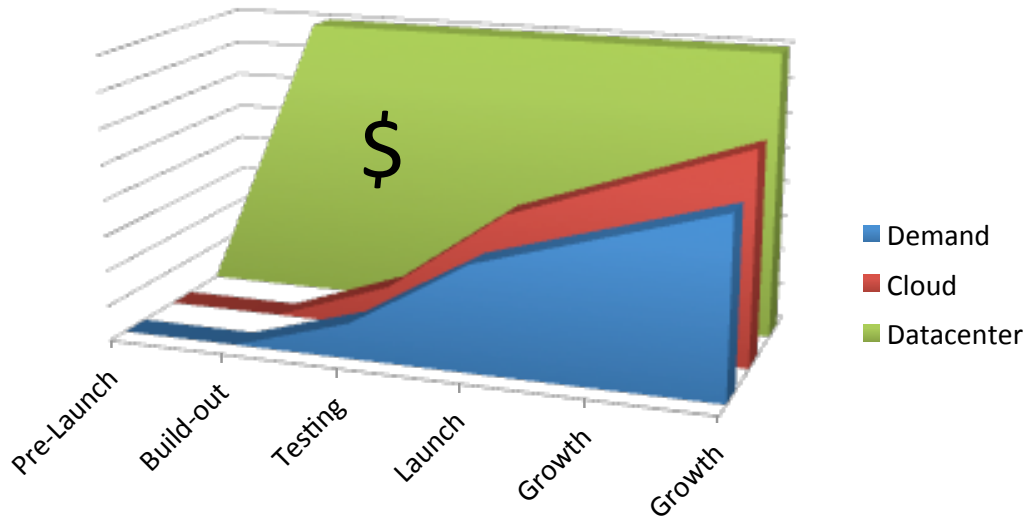
BV Battery Ventures

The Capacity Planning Problem

@adrianco
BV
Battery Ventures

Best Case Waste

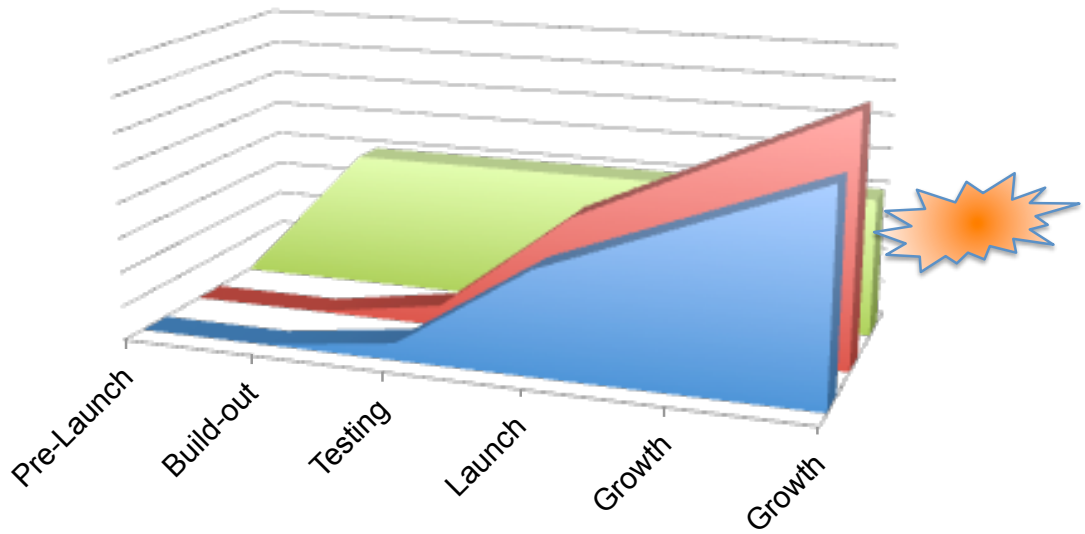
Product Launch Agility - Rightsized



Cloud capacity used is maybe half average DC capacity

Failure to Launch

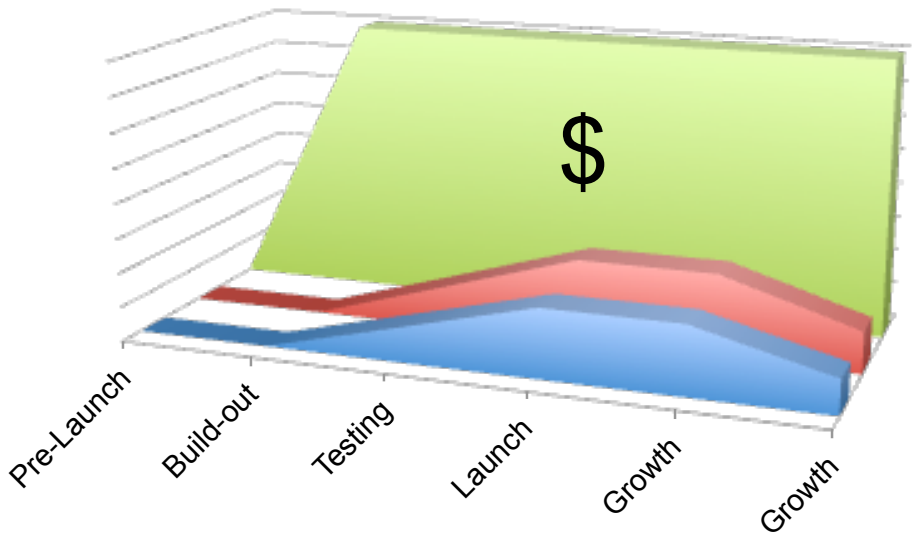
Product Launch - Under-estimated



Mad scramble to add more DC capacity during launch phase outages

Over the Top Losses

Product Launch Agility – Over-estimated



Capacity wasted on failed launch magnifies the losses

Turning off Capacity

Off-peak production
Test environments
Dev out of hours
Dormant Data Science

Containerize Test Environments

Snapshot or freeze

Fast restart needed

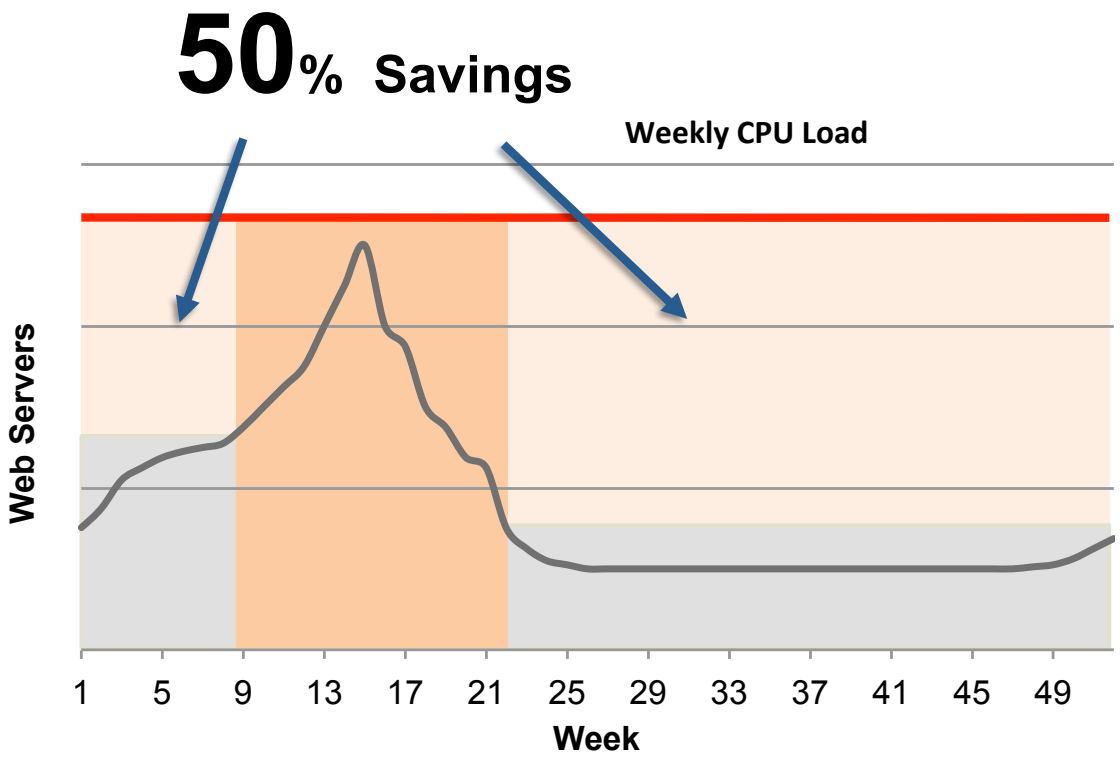
Persistent storage

40 of 168 hrs/wk

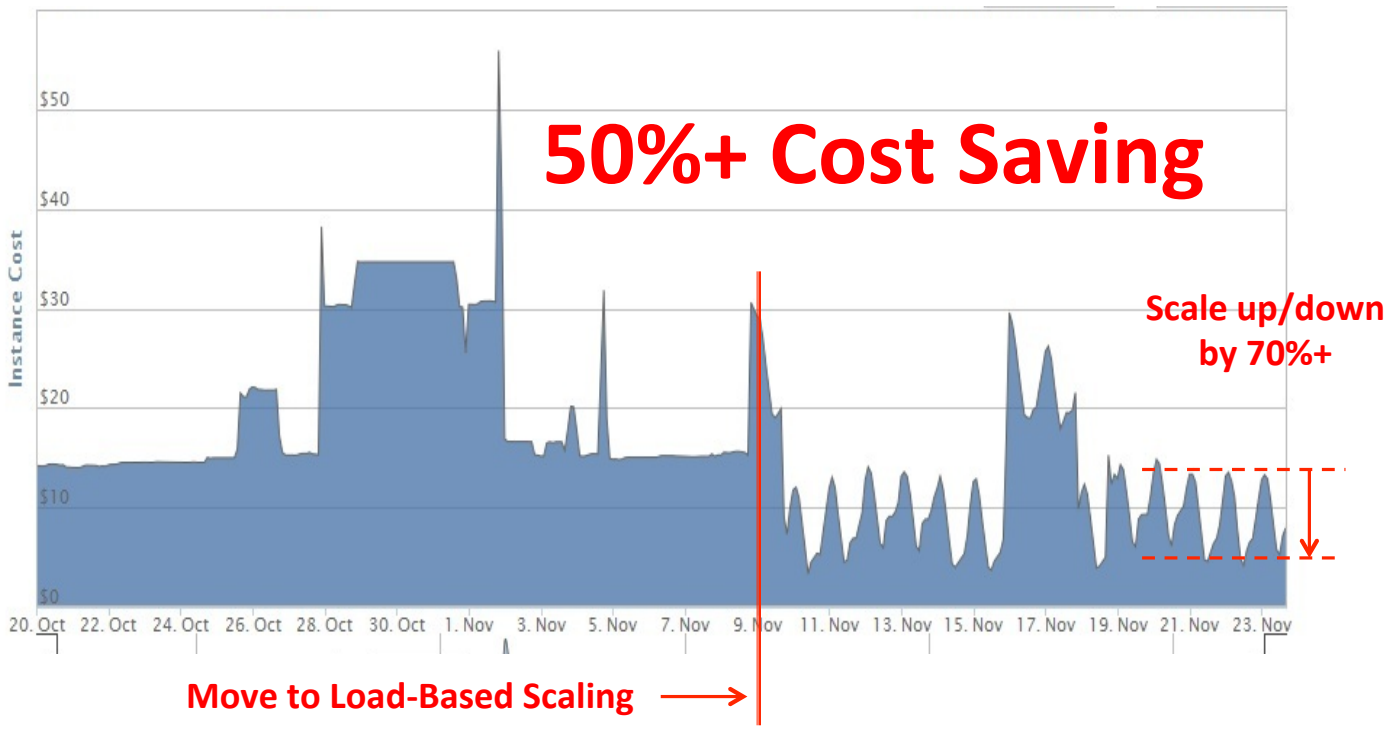
Bin-packed containers

shippable.com saved 70%

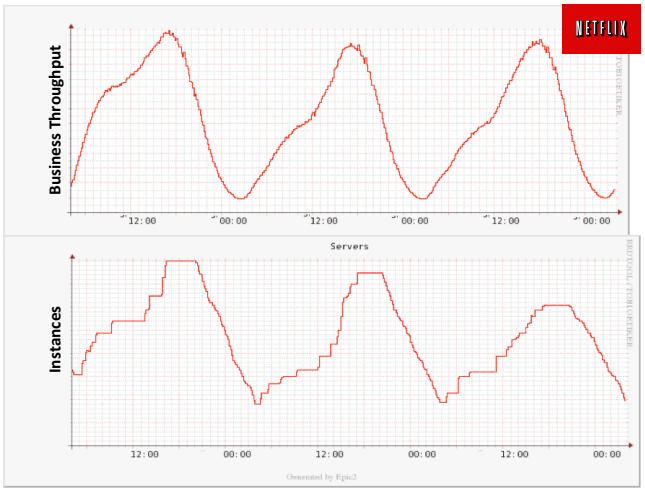
Seasonal Savings



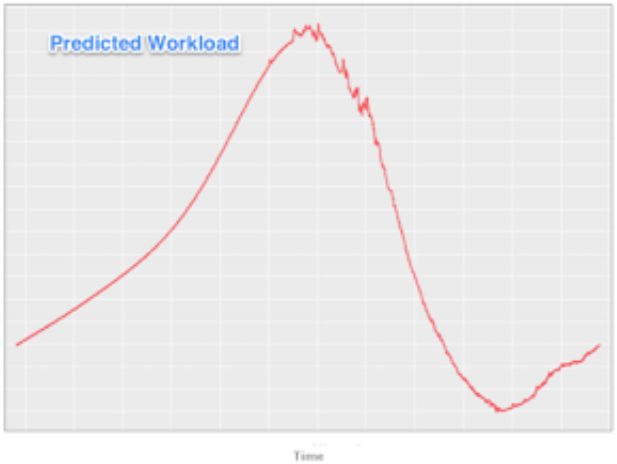
Autoscale the Costs Away

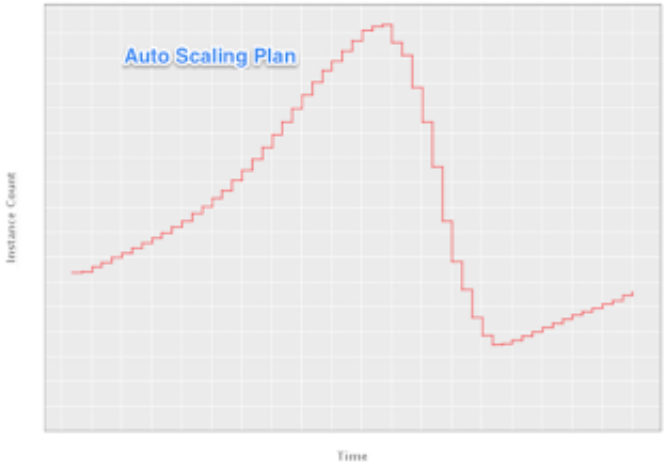


Daily Duty Cycle



Reactive Autoscaling saves around 50%





Predictive Autoscaling saves around 70%
See Scryer on Netflix Tech Blog

Underutilized and Unused

AWS Support – Trusted Advisor – Your personal cloud assistant

Trusted Advisor Beta

Expand All | Download Excel | Refresh All | Contact Support

The AWS Trusted Advisor program monitors AWS infrastructure services, identifies customer configurations, compares them to known best practices, and then notifies customers when opportunities may exist to save money, improve system performance, or close security gaps.

No issue detected | Investigation Recommended | Action Recommended

Cost Optimizing Checks

Unused Elastic IPs — Updated: **2012-06-14 00:00 PDT**

Summary: **0 of 6 Elastic IPs are not in use**

Underutilized EC2 Instances — Updated: **2012-06-13 22:27 PDT**

Summary: **27 EC2 instances are potentially underutilized**

Clean Up the Crud

Other simple optimization tips

- **Don't forget to…**
  - Disassociate unused EIPs
  - Delete unassociated Amazon EBS volumes
  - Delete older Amazon EBS snapshots
  - Leverage Amazon S3 Object Expiration

NETFLIX OSS Janitor Monkey cleans up unused resources

Total Cost of Oranges

When Comparing TCO…

Make sure that you are including all the cost factors into consideration

Place
Power
Pipes
People
Patterns

Total Cost of Oranges

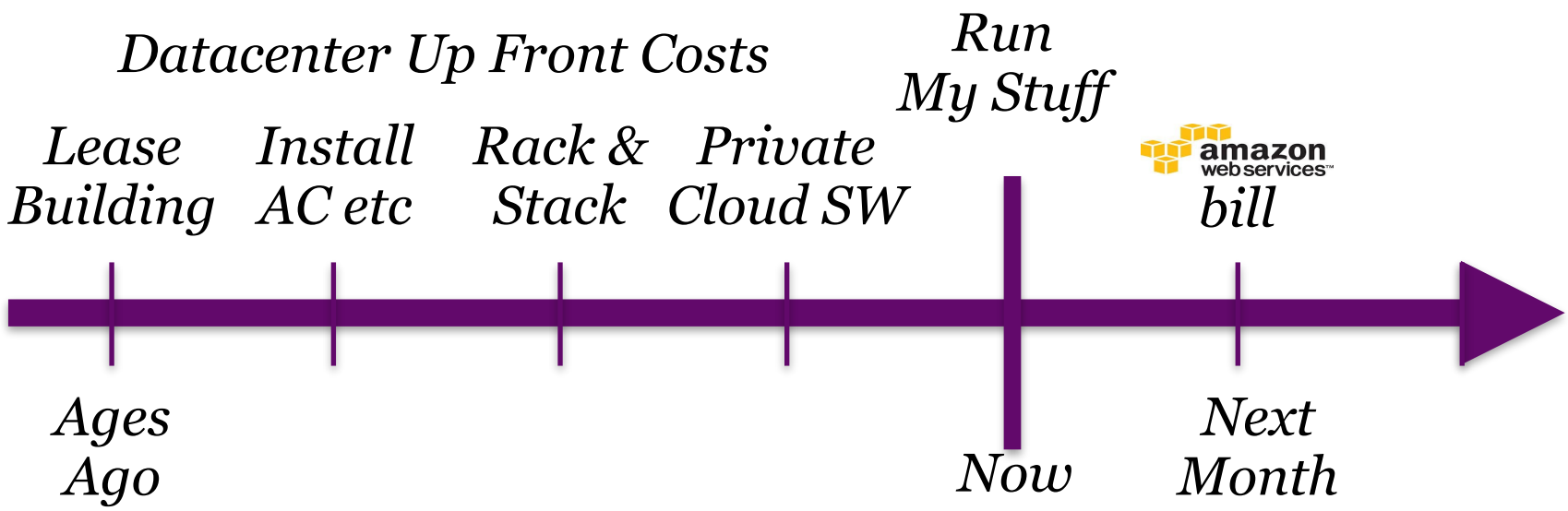
When Comparing TCO…

Make sure that you are including all the cost factors into consideration

Place
Power
Pipes
People
Patterns

How much does datacenter automation software and support cost per instance?

When Do You Pay?



@adrianco
BV
Battery Ventures

Cost Model Comparisons

AWS has most complex model

- *Both highest and lowest cost options!*

CPU/Memory Ratios Vary

- *Can't get same config everywhere*

Features Vary

- *Local SSD included on some vendors, not others*
- *Network and storage charges also vary*

Digital Ocean Flat Pricing

Hourly Price ($0.06/hr)	*Monthly Price ($40/mo)*
$ No Upfront	***$ No Upfront***
$0.060/hr	***$0.056/hr***
$1555/36mo	***$1440/36mo***
Savings	***7%***

Prices on Dec 7th, for 2 Core, 4G RAM, SSD, purely to show typical savings

Google Sustained Usage

Full Price Without Sustained Usage	*Typical Sustained Usage Each Month*	*Full Sustained Usage Each Month*
$ No Upfront	***$ No Upfront***	***$ No Upfront***
$0.063/hr	***$0.049/hr***	***$0.045/hr***
$1633/36mo	***$1270/36mo***	***$1166/36mo***
Savings	***22%***	***29%***

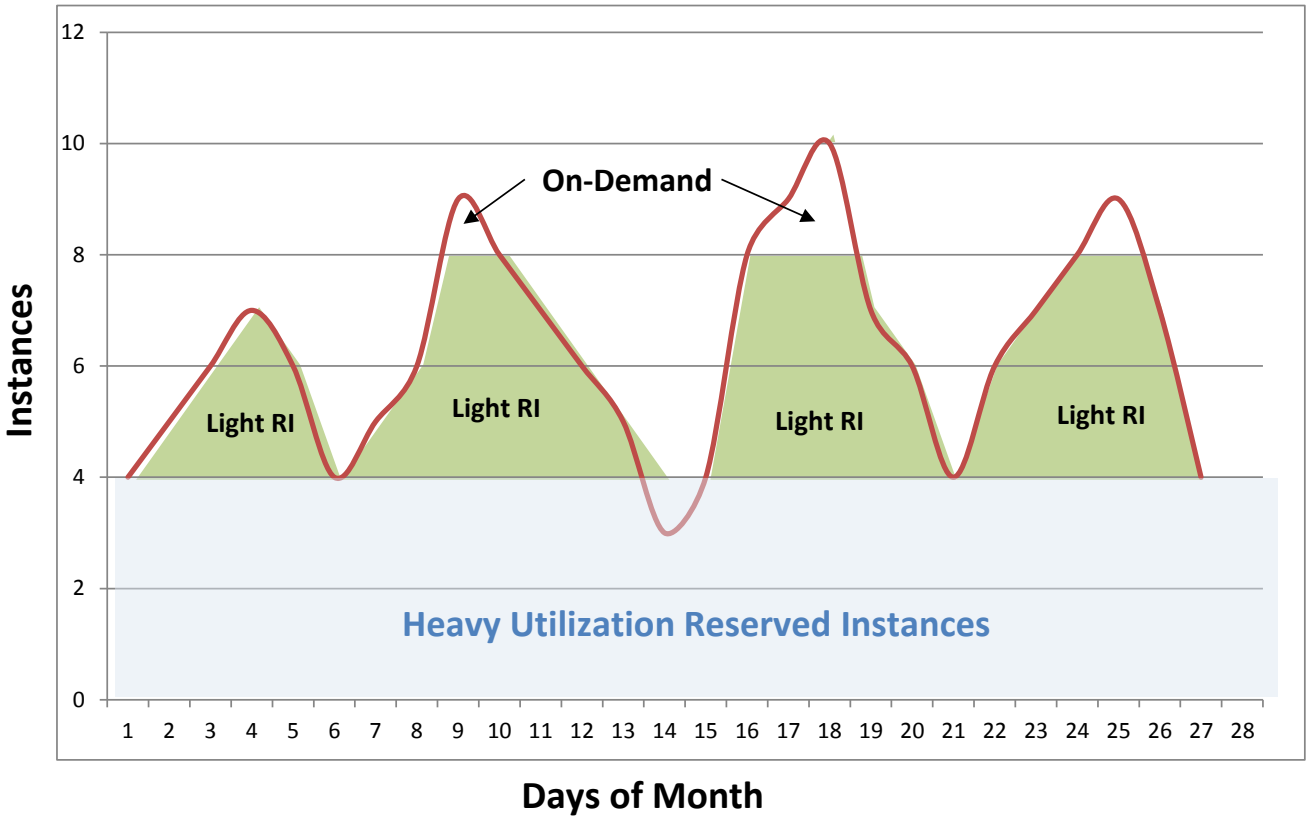
Prices on Dec 7th, for n1.standard-1 (1 vCPU, 3.75G RAM, no disk) purely to show typical savings

AWS Reservations

On Demand	*No Upfront 1 year*	*Partial Upfront 3 year*	*All Upfront 3 year*
$ No Upfront	*$No Upfront*	*$337 Upfront*	*$687 Upfront*
$0.070/hr	*$0.050/hr*	*$0.0278/hr*	*$0.00/hr*
$1840/36mo	*$1314/36mo*	*$731/36mo*	*$687/36mo*
Savings	*29%*	*60%*	*63%*

Prices on Dec 7th, for m3.medium (1 vCPU, 3.75G RAM, SSD) purely to show typical savings

Blended Benefits



On Demand

Partial Upfront

All Upfront

Consolidated Reservations

Burst capacity guarantee

Higher availability with lower cost

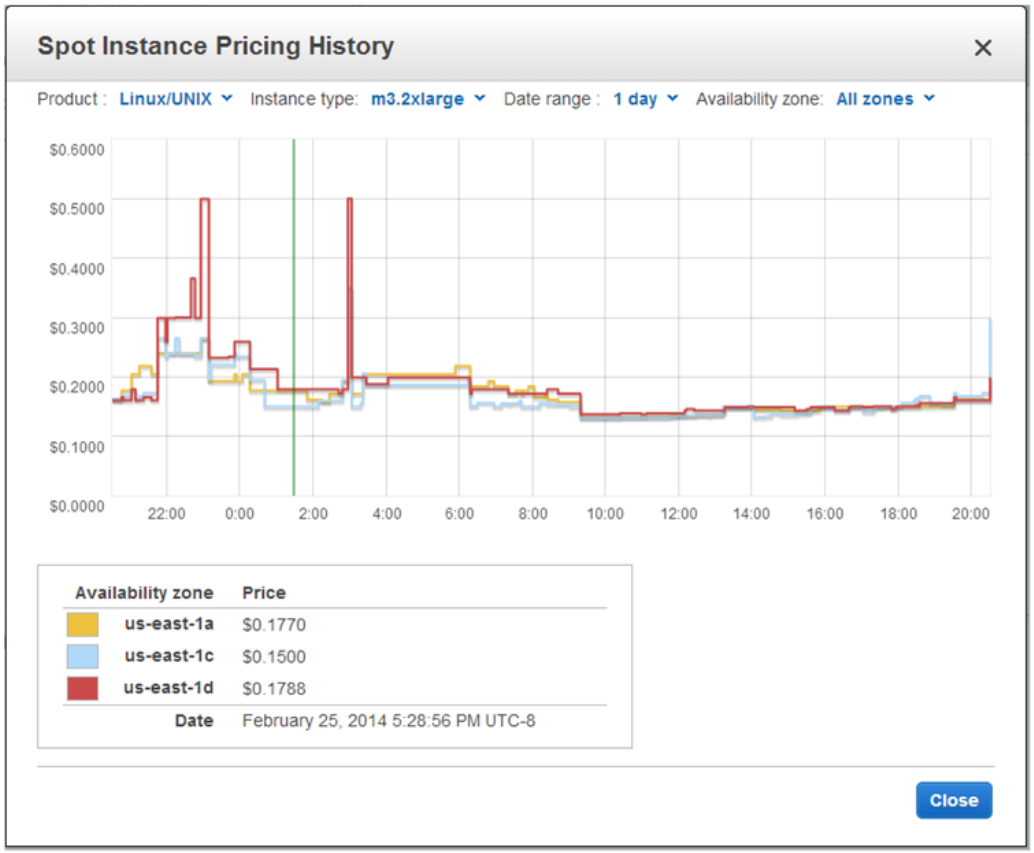
Other accounts soak up any extra

Monthly billing roll-up

<u>Capitalize</u> upfront charges!

But: Fixed location and instance type

Use EC2 Spot Instances

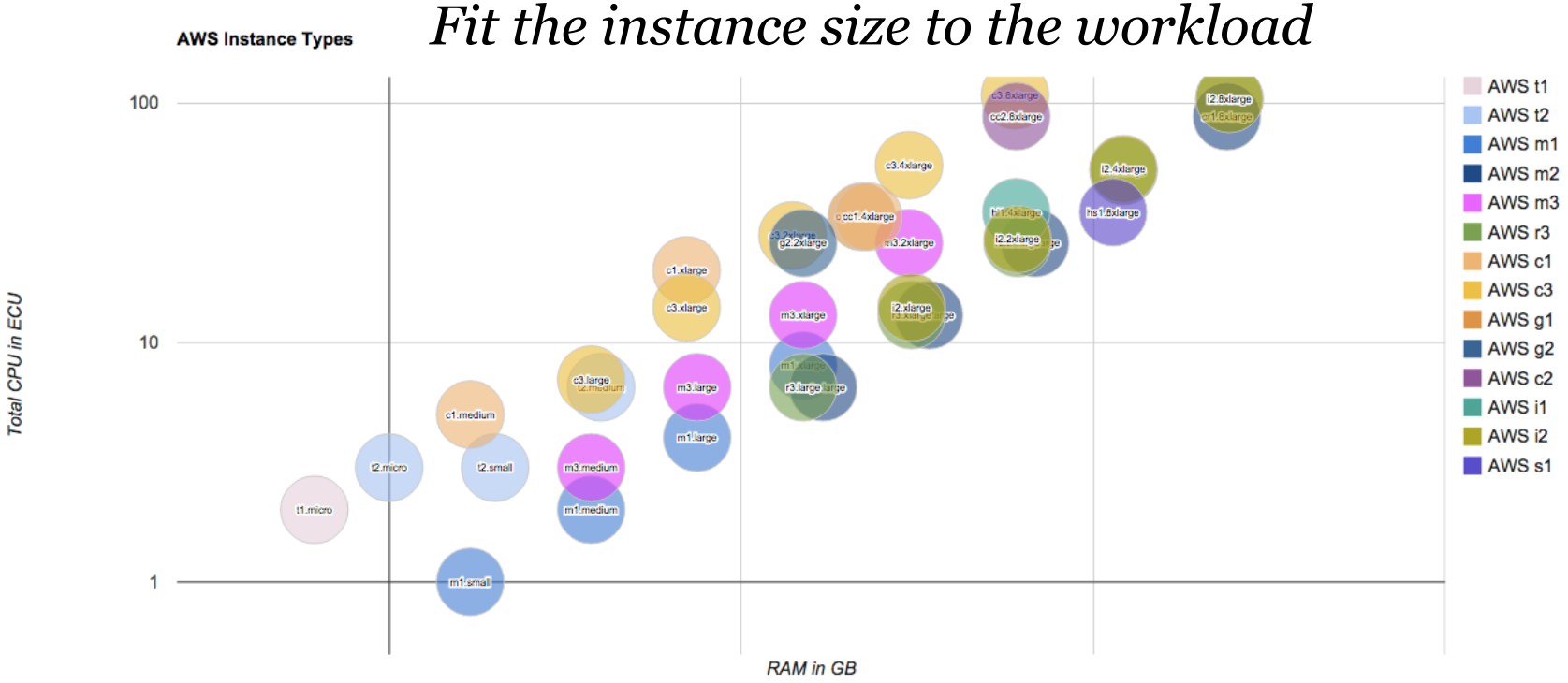


Cloud native dynamic autoscaled spot instances

Real world total savings up to 50%

Right Sizing Instances

Fit the instance size to the workload



Six Ways to Cut Costs

#1 Business Agility by Rapid Experimentation = Profit

#2 Business-driven Auto Scaling Architectures = Savings

#3 Mix and Match Reserved Instances with On-Demand = Savings

#4 Consolidated Billing and Shared Reservations = Savings

#5 Always-on Instance Type Optimization = Recurring Savings

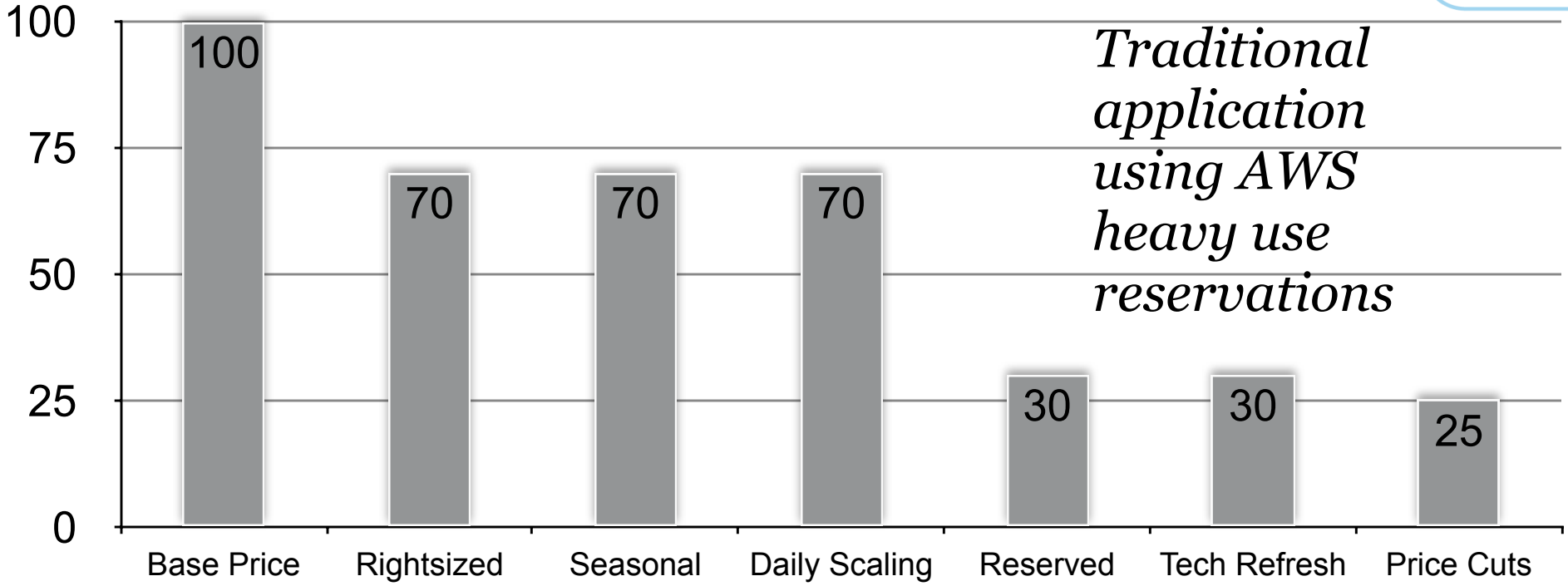
#6 Follow the Customer (Run web servers) during the day
Follow the Money (Run Hadoop clusters) at night

Credit to Jinesh Varia of AWS for this summary

Compounded Savings

@adrianco
BV
Battery Ventures

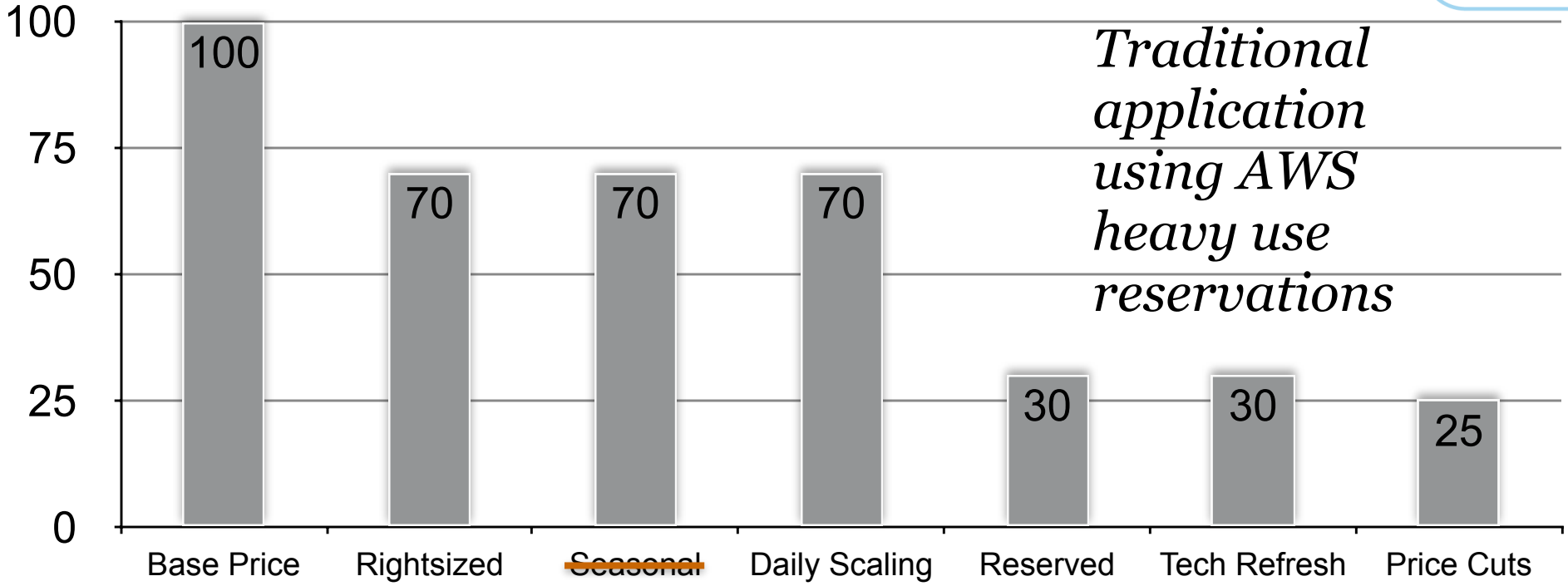
Lift and Shift Compounding



Traditional application using AWS heavy use reservations

Base price is for capacity bought up-front

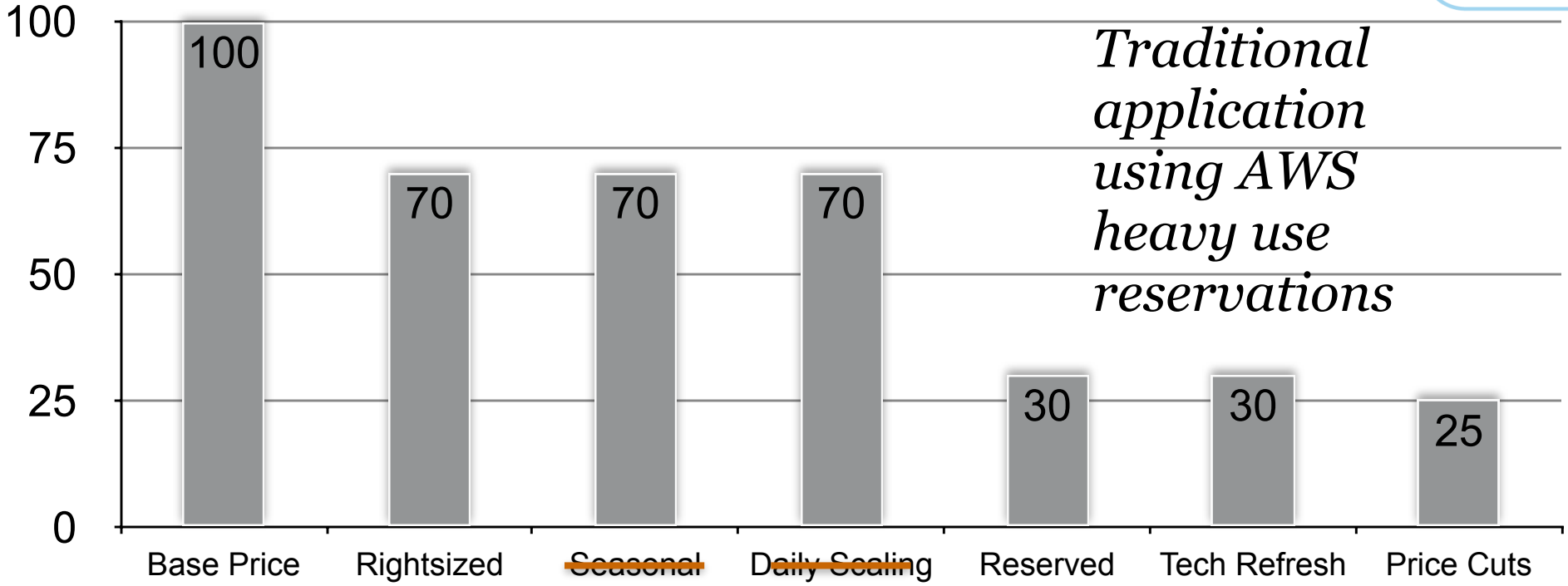
Lift and Shift Compounding



Traditional application using AWS heavy use reservations

Base price is for capacity bought up-front

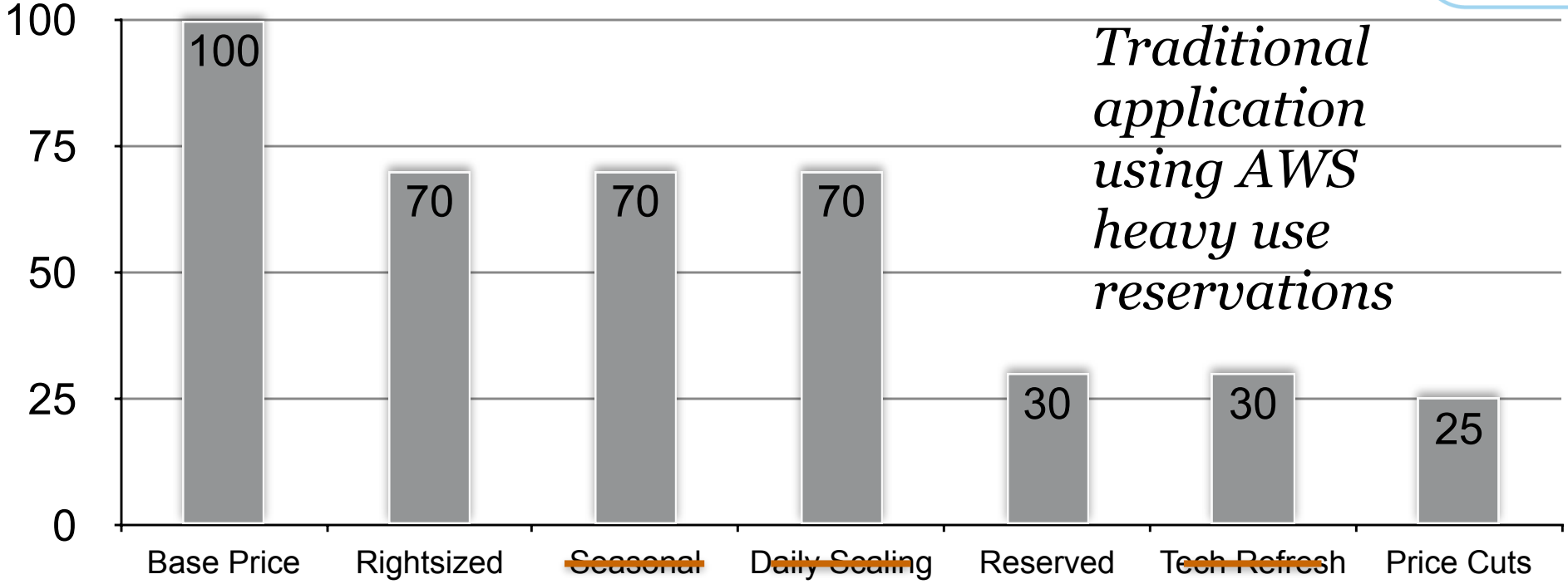
Lift and Shift Compounding



Traditional application using AWS heavy use reservations

Base price is for capacity bought up-front

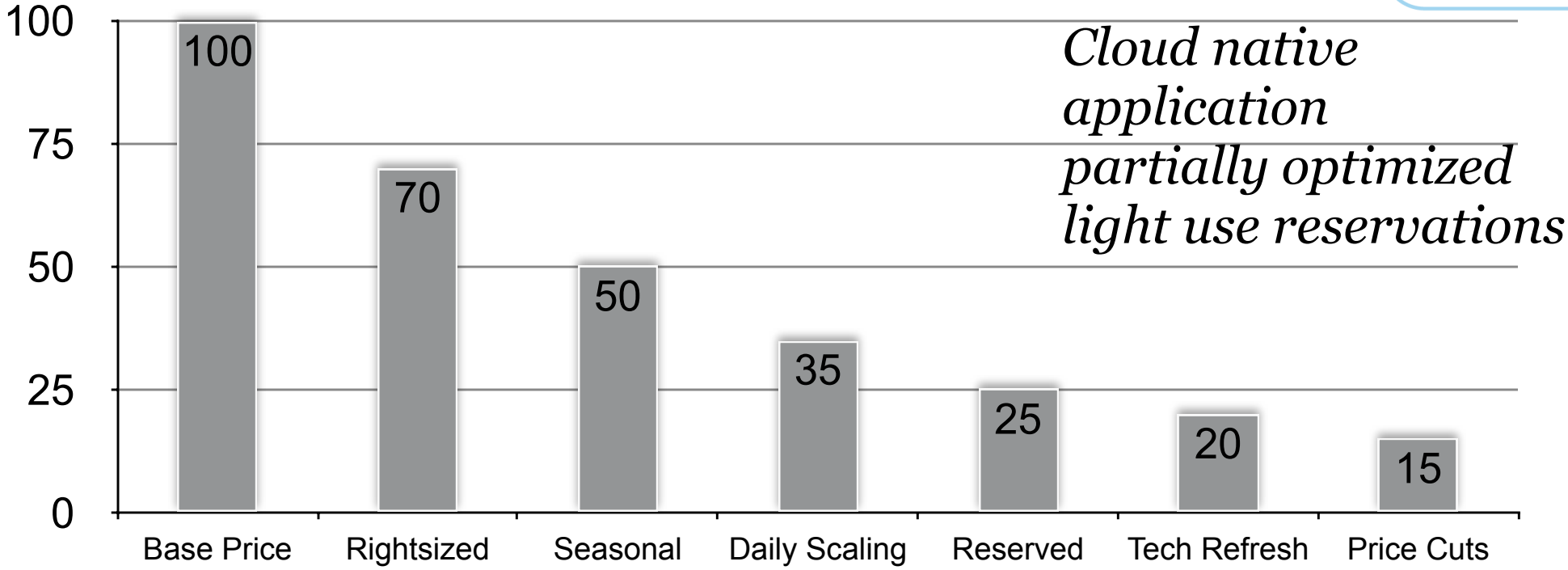
Lift and Shift Compounding



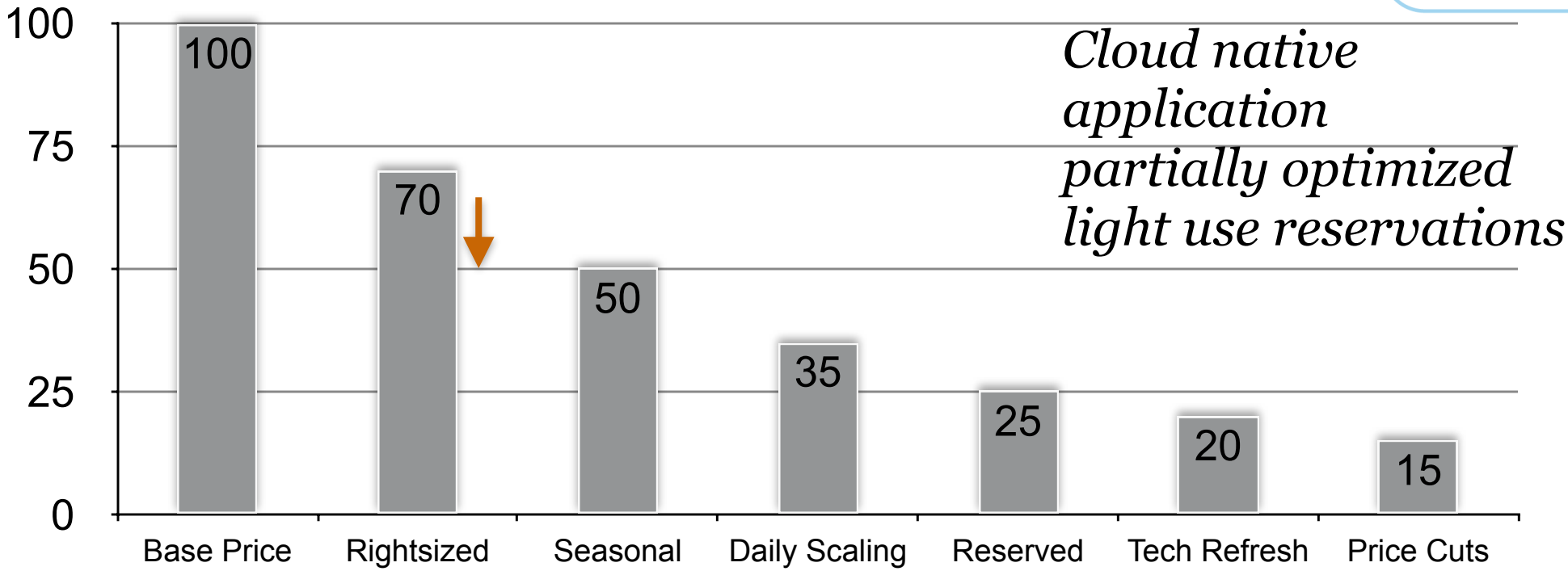
Traditional application using AWS heavy use reservations

Base price is for capacity bought up-front

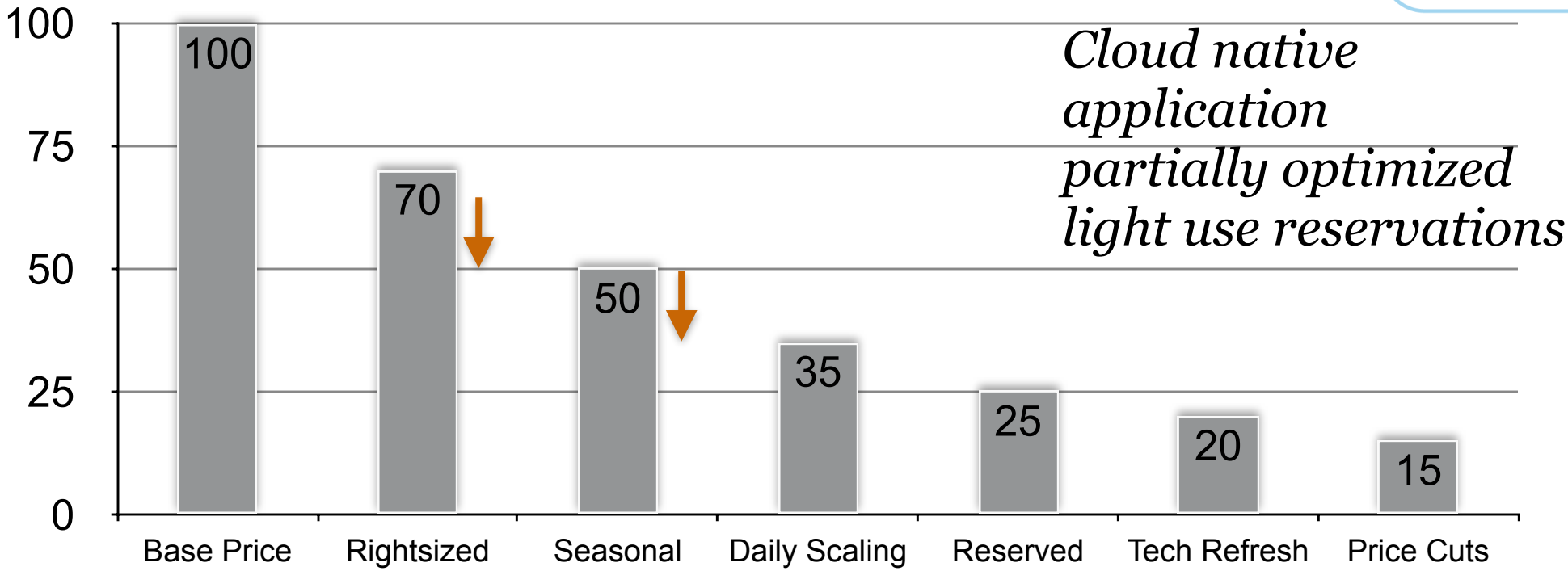
Conservative Compounding



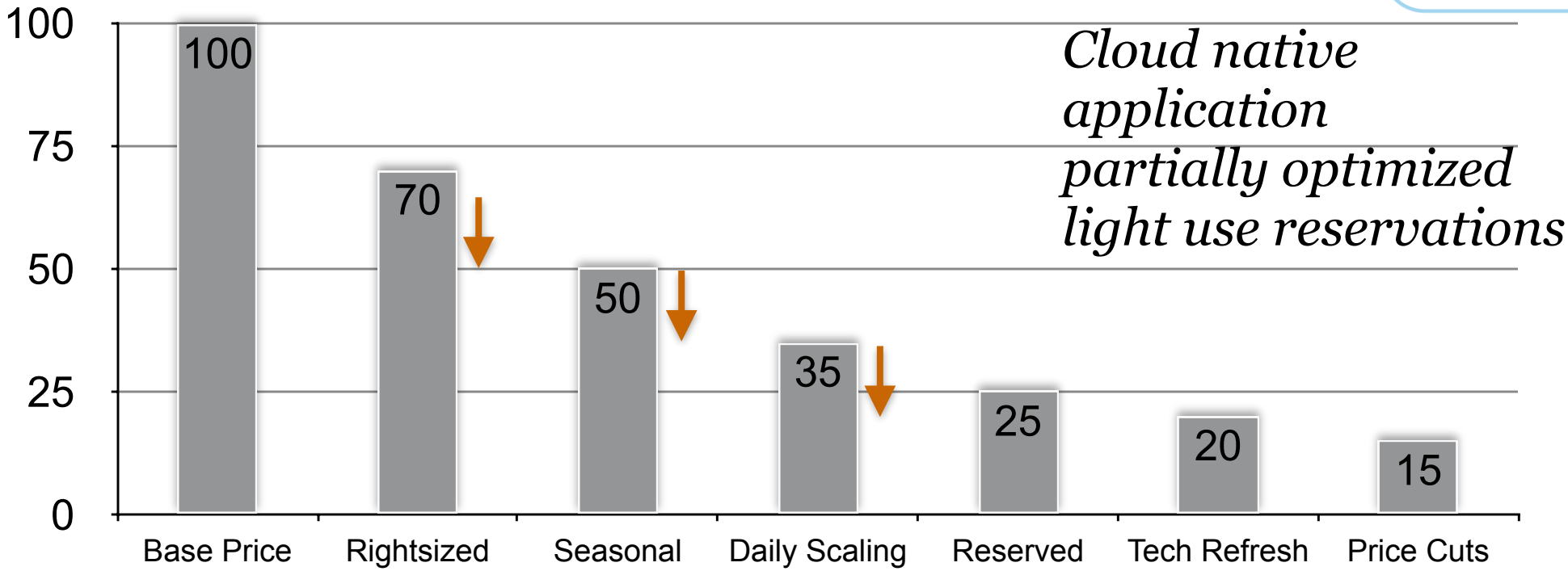
Conservative Compounding



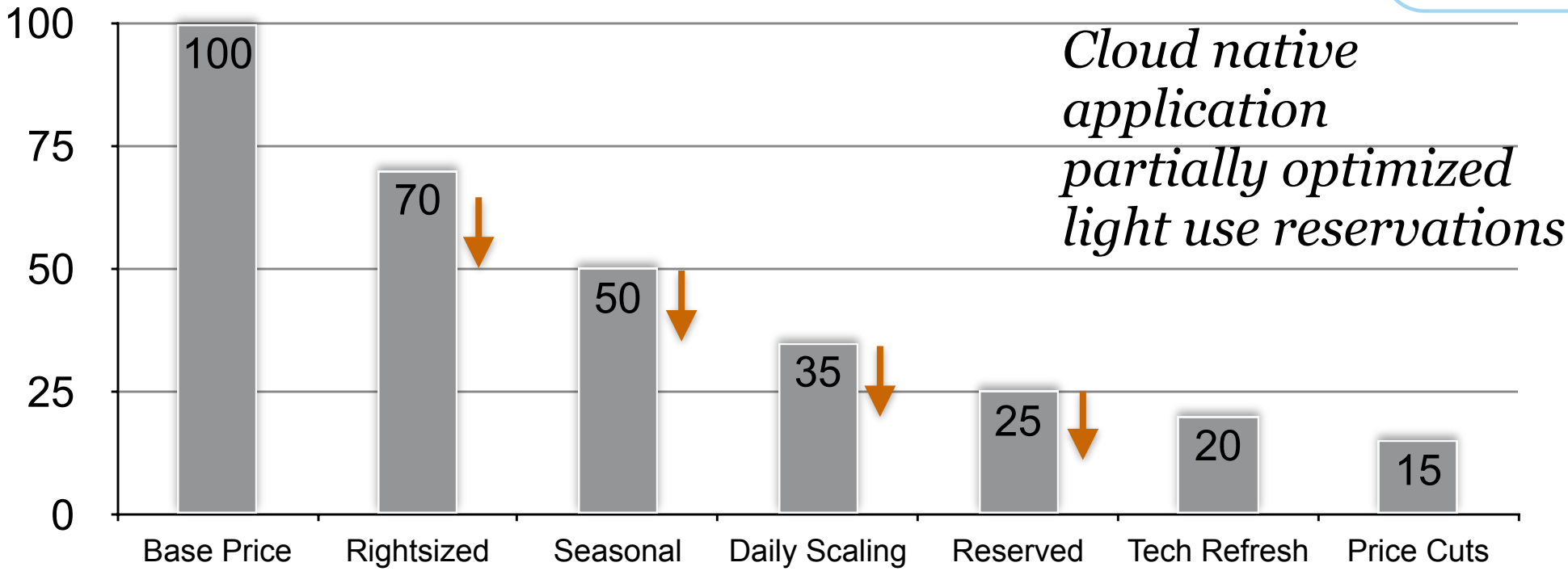
Conservative Compounding



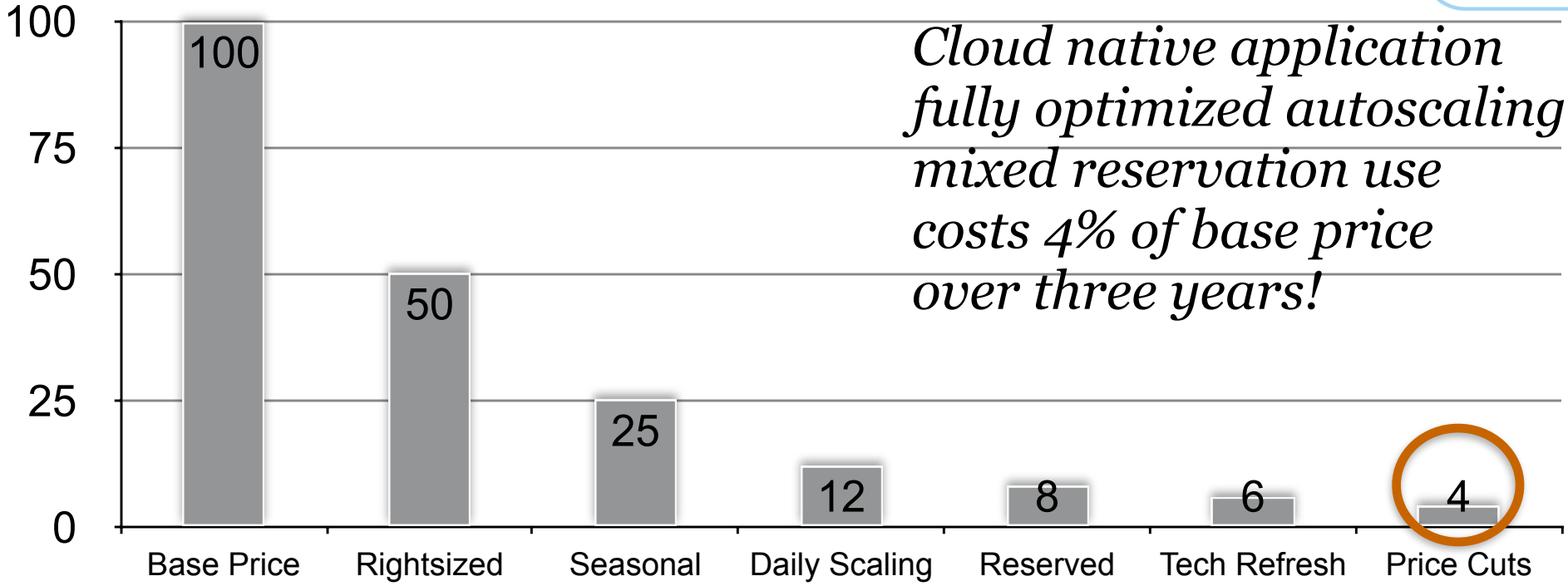
Conservative Compounding



Conservative Compounding

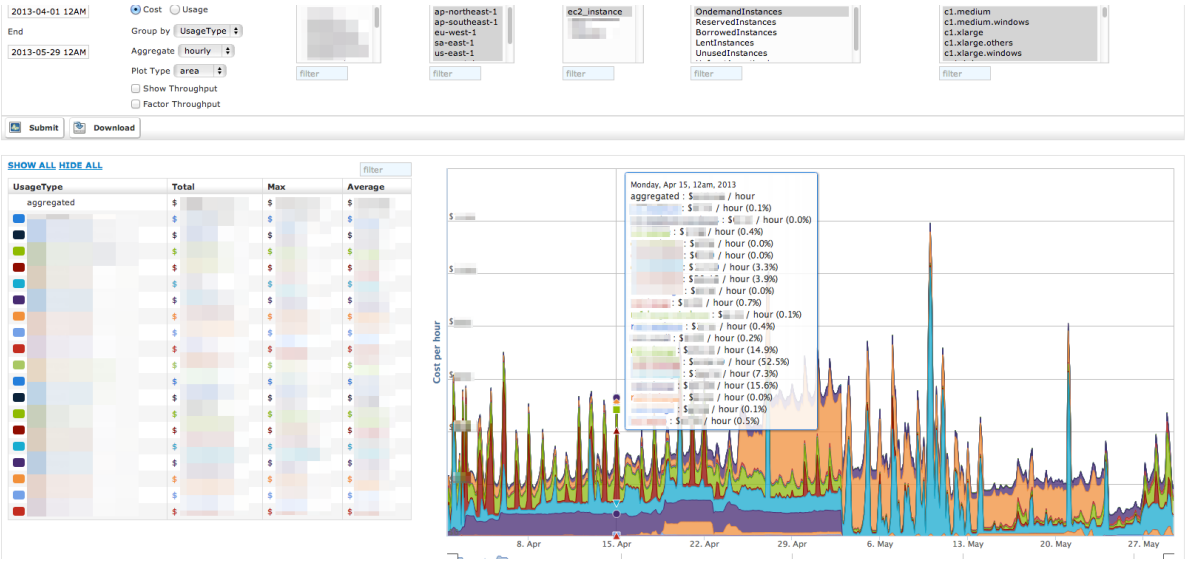


Agressive Compounding



Cloud native application fully optimized autoscaling mixed reservation use costs 4% of base price over three years!

Cost Monitoring and Optimization



ICE

NETFLIX | OSS

Cloud Options™

cloudability

krystallize TECHNOLOGIES

cedexis

Final Thoughts

Turn off idle instances

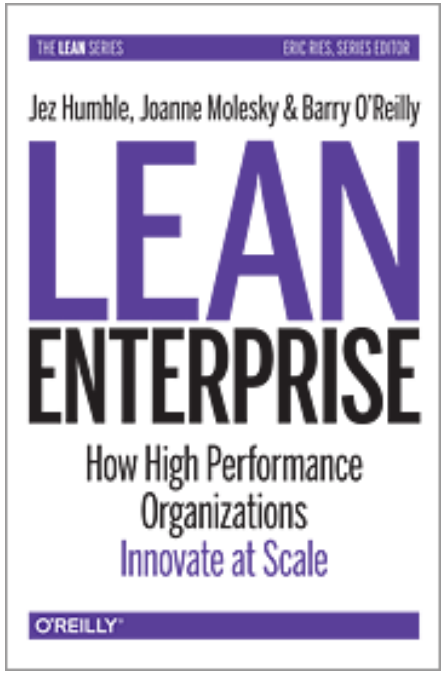
Clean up unused stuff

Optimize for pricing model

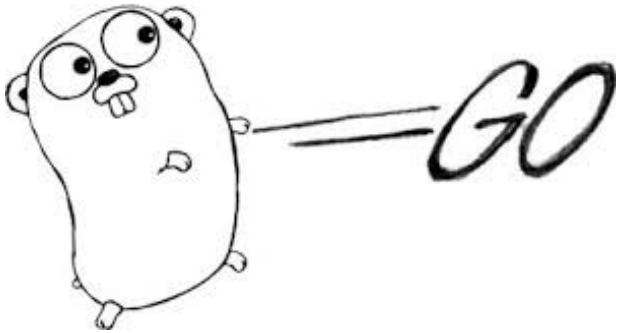
Assume prices will go down

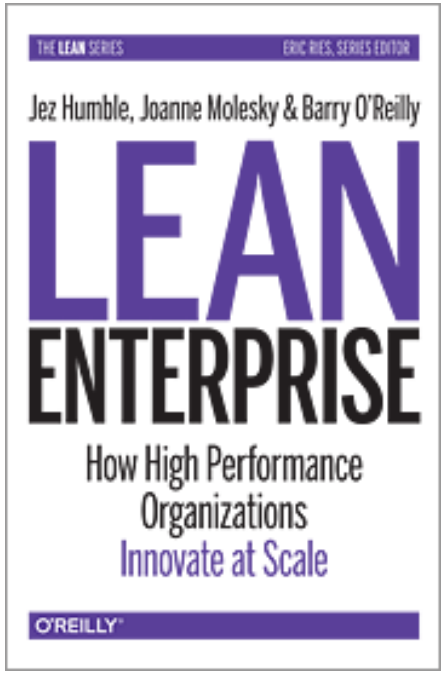
Go cloud native to be fast and save

Forward Thinking

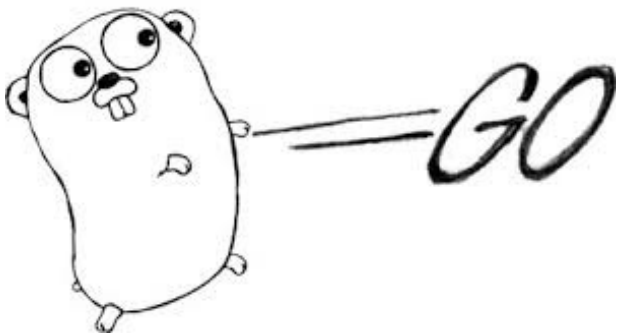


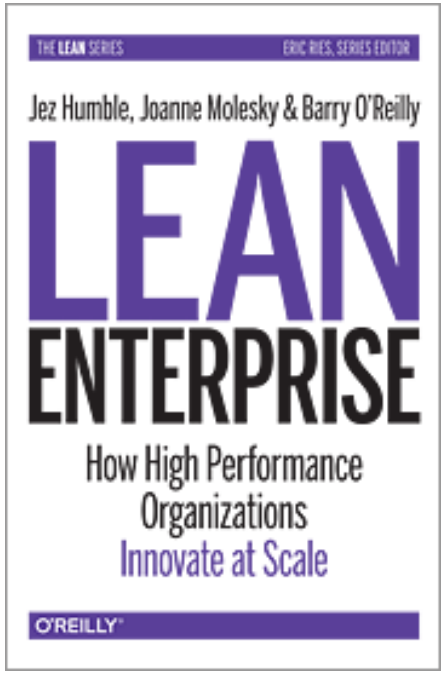
Forward Thinking

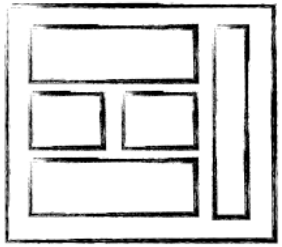




Forward Thinking







MONOLITHIC/LAYERED

MICRO SERVICES

http://eugenedvorkin.com/seven-micro-services-architecture-advantages/

Any Questions?

- *Battery Ventures http://www.battery.com*
- *Adrian's Tweets @**adrianco** and Blog http://perfcap.blogspot.com*
- *Slideshare http://slideshare.com/adriancockcroft*

- *Monitorama Opening Keynote Portland OR - May 7th, 2014*
- *GOTO Chicago Opening Keynote May 20th, 2014*
- *Qcon New York – Speed and Scale - June 11th, 2014*
- *Structure - Cloud Trends - San Francisco - June 19th, 2014*
- *GOTO Copenhagen/Aarhus – Fast Delivery - Denmark – Sept 25th, 2014*
- *DevOps Enterprise Summit - San Francisco - Oct 21-23rd, 2014 #DOES14*
- *GOTO Berlin - Migrating to Microservices - Germany - Nov 6th, 2014*
- *AWS Re:Invent - Cloud Native Cost Optimization - Las Vegas - November 14th, 2014*
- *O'Reilly Software Architecture Conference - Boston March 16th 2015*