Why Does Performance Matter?
Latency Efficiency
Users: Response Latency
Developers: Release Latency
Operators: Efficiency
Less Time
Less Cost
Faster Delivery

See talks by @adrianco
Speed and Scale - QCon New York
Fast Delivery - GOTO Copenhagen
Cheaper

This talk: How to use Cloud Native architecture to reduce cost without slowing down releases
Speeding up Development
Cloud Native Applications
Cost Optimization
Why am I here?
Why am I here?

By Simon Wardley
http://enterpriseitadoption.com/
Why am I here?

By Simon Wardley http://enterpriseitadoption.com/

By 2009:

- Ignore
- Ignore
- Ignore
- "No"
- "No"
- "No, dammit"
- "Oh No"
- "Oh %&!"

Adoption:

Netflix

Rest of World

Enterprise IT

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

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

- Observe
- Orient
- Decide
- Act

- Land grab opportunity
- Competitive Move
- Customer Pain Point
- Measure Customers
- Continuous Delivery

INNOVATION
INNOVATION

Observe

Land grab opportunity
Measure Customers

Orient

Competitive Move
Customer Pain Point

Decide

Act

Continuous Delivery

Analysis
Model Hypotheses

Customers

Measure Customers
Continuous Delivery
Act

Observe

Competitive Move
Customer Pain Point

Orient

INNOVATION

Model Hypotheses

Analysis

Competitive Move

Customer Pain Point

Orient

INNOCATION

Observe

Land grab opportunity
Measure Customers
Observe

Innovation

Orient

Decide

Act

Land grab opportunity

Competitive Move

Customer Pain Point

Measure Customers

Continuous Delivery

Analysis

Model Hypotheses

BIG DATA

Customers

Continuous Delivery

Hypotheses
INNOVATION

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

BIG DATA

INNOVATION

Competitive Move

Customer Pain Point

Land grab opportunity

Measure Customers

Continuous Delivery

Decide

Plan Response

JFDI

Observe

Orient

Act

Share Plans

Analysis

Model Hypotheses

BIG DATA

INNOVATION

Competitive Move

Customer Pain Point

Land grab opportunity

Measure Customers
Observe

Orient

Decide

Act

Land grab opportunity

Competitive Move

Customer Pain Point

Measure Customers

INNOVATION

Continuous Delivery

CLOUD

Act

Incremental Features

Launch AB Test

Automatic Deploy

Continuous Delivery

CULTURE

CLOUD

Decide

Share Plans

Plan Response

JFDI

INNOVATION

Continuous Delivery

BIG DATA

Analysis

Model Hypotheses

CULTURE

INNOVATION
Observe

Orient

Decide

Act

Land grab opportunity

Competitive Move

Customer Pain Point

Measure Customers

Analysis

Model Hypotheses

Launch AB Test

Continuous Delivery

Plan Response

Incremental Features

Cloud

Hypotheses

Customer Pain Point

CULTURE

CLOUD

Big Data

INNOVATION

JFDI

Continuous Delivery

Automatic Deploy

Share Plans
Release Plan

QA Release Integration

Ops Replace Old With New Release

Monolithic service updates

Works well with a small number of developers and a single language like php, java or ruby
Monolithic service updates

Works well with a small number of developers and a single language like php, java or ruby
Monolithic service updates

Works well with a small number of developers and a single language like php, java or ruby
Breaking Down the SILOs
Breaking Down the SILOs

Prod Mgr  UX  Dev  QA  DBA  Sys Adm  Net Adm  SAN Adm
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Prod Mgr  UX  Dev  QA  DBA  Sys Adm  Net Adm  SAN Adm
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Microservices

Product Team Using Microservices

Product Team Using Microservices
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Monolithic Delivery

Prod Mgr  UX  Dev  QA  DBA

Product Team Using Microservices

Product Team Using Microservices

Product Team Using Microservices

Sys Adm  Net Adm  SAN Adm

Platform Team
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Monolithic Delivery

Product Team Using Microservices

Product Team Using Microservices

Product Team Using Microservices

Platform Team

Sys Adm

Net Adm

SAN Adm

Prod Mgr

UX

Dev

QA

DBA

API
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Monolithic Delivery

Product Team Using Microservices
Product Team Using Microservices
Product Team Using Microservices

DevOps is a Re-Org!
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.
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
Standardized portable container deployment saves time and effort

https://hub.docker.com
Configure

Developer

Release Plan

Deploy

Standardized Services

Deploy Feature to Production

Bugs

Deploy Feature to Production

Deploy Feature to Production

https://hub.docker.com

Standardized portable container deployment saves time and effort
Developing at the Speed of Docker

Developers
- Compile/Build
- Seconds

Extend container
- Package dependencies
- Seconds

PaaS deploy Containers
- Docker startup
- Seconds
Developing at the Speed of Docker

- **Developers**
  - Compile/Build
  - Seconds

- **Extend container**
  - Package dependencies
  - Seconds

- **PaaS deploy Containers**
  - Docker startup
  - Seconds

*Speed is addictive, hard to go back to taking much longer to get things done*
What Happened?

Rate of change increased

Cost and size and risk of change reduced
Cloud Native Applications
Cloud Native
A new engineering challenge

Construct a highly agile and highly available service from ephemeral and assumed broken components
Inspiration
Inspiration
State of the Art in Cloud Native Microservice Architectures

AWS Re:Invent : Asgard to Zuul [https://www.youtube.com/watch?v=p7ysHhs5hl0](https://www.youtube.com/watch?v=p7ysHhs5hl0)
Resiliency at Massive Scale [https://www.youtube.com/watch?v=ZfYJHtVL1_w](https://www.youtube.com/watch?v=ZfYJHtVL1_w)

Microservice Architecture [https://www.youtube.com/watch?v=CriDUYtfjrj](https://www.youtube.com/watch?v=CriDUYtfjrj)

[http://www.infoq.com/presentations/scale-gilt](http://www.infoq.com/presentations/scale-gilt)

[http://www.slideshare.net/mcculloughsean/itier-breaking-up-the-monolith-philly-ete](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-Timeline-Scalability)

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
Autoscaled Ephemeral Instances at Netflix

Largest services use autoscaled red/black code pushes

Average lifetime of an instance is 36 hours
Netflix Automatic Code Deployment Canary Bad Signature

Implemented by Simon Tuffs
Netflix Automatic Code Deployment Canary Bad Signature

Implemented by Simon Tuffs
Speeding Up The Platform

Datacenter Snowflakes
• Deploy in months
• Live for years
Speeding Up The Platform

Datacenter Snowflakes
- Deploy in months
- Live for years

Virtualized and Cloud
- Deploy in minutes
- Live for weeks
Speeding Up The Platform

Datacenter Snowflakes
• Deploy in months
• Live for years

Virtualized and Cloud
• Deploy in minutes
• Live for weeks

Docker Containers
• Deploy in seconds
• Live for minutes/hours
Speeding Up The Platform

- Datacenter Snowflakes
  - Deploy in months
  - Live for years

- Virtualized and Cloud
  - Deploy in minutes
  - Live for weeks

- Docker Containers
  - Deploy in seconds
  - Live for minutes/hours

- AWS Lambda
  - Deploy in milliseconds
  - Live for seconds
Speeding Up The Platform

Datacenter Snowflakes
- Deploy in months
- Live for years

Virtualized and Cloud
- Deploy in minutes
- Live for weeks

Docker Containers
- Deploy in seconds
- Live for minutes/hours

AWS Lambda
- Deploy in milliseconds
- Live for seconds

† Speed enables and encourages new microservice architectures
With AWS Lambda compute resources are charged by the 100ms, not the hour

First 1,000,000 node.js executions/month are free
First 400,000 GB-seconds of RAM-CPU are free
Monitoring Requirements

Metric resolution microseconds
Metric update rate 1 second
Metric to display latency less than human attention span (<10s)
Low Latency SaaS Based Monitors

www.vividcortex.com and www.boundary.com
Adrian’s Tinkering Projects

Model and visualize microservices
Simulate interesting architectures

See [github.com/adrianco/spigo](https://github.com/adrianco/spigo)
Simulate Protocol Interactions in Go

See [github.com/adrianco/d3grow](https://github.com/adrianco/d3grow)
Dynamic visualization
Cost Optimization
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/](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
The Capacity Planning Problem
Best Case Waste

Product Launch Agility - Rightsized

Cloud capacity used is maybe half average DC capacity
Failure to Launch

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

When you turn off your cloud resources, you actually stop paying for them
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

50% Savings

Weekly CPU Load

Web Servers

Week

1 5 9 13 17 21 25 29 33 37 41 45 49
Autoscale the Costs Away

50%+ Cost Saving

Scale up/down by 70%+

Move to Load-Based Scaling
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

<table>
<thead>
<tr>
<th>Trusted Advisor</th>
<th>Beta</th>
<th>Expand All</th>
<th>Download Excel</th>
<th>Refresh All</th>
<th>Contact Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cost Optimizing Checks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Unused Elastic IPs</td>
<td></td>
</tr>
<tr>
<td>✔️ Summary: 0 of 5 Elastic IPs are not in use</td>
<td>✔️ Summary: 27 EC2 instances are potentially underutilized</td>
</tr>
<tr>
<td>Updated: 2012-06-14 00:00 PDT</td>
<td>Updated: 2012-06-13 22:27 PDT</td>
</tr>
</tbody>
</table>
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

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?

Datacenter Up Front Costs
- Lease Building
- Install AC etc
- Rack & Stack
- Private Cloud SW

Run My Stuff

bill

Ages Ago
Now
Next Month
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

<table>
<thead>
<tr>
<th>Hourly Price ($0.06/hr)</th>
<th>Monthly Price ($40/mo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ No Upfront</td>
<td>$ No Upfront</td>
</tr>
<tr>
<td>$0.060/hr</td>
<td>$0.056/hr</td>
</tr>
<tr>
<td>$1555/36mo</td>
<td>$1440/36mo</td>
</tr>
<tr>
<td>Savings</td>
<td>7%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Full Price Without Sustained Usage</th>
<th>Typical Sustained Usage Each Month</th>
<th>Full Sustained Usage Each Month</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$ No Upfront</strong></td>
<td>$ No Upfront</td>
<td>$ No Upfront</td>
<td>$ No Upfront</td>
</tr>
<tr>
<td><strong>$0.063/hr</strong></td>
<td>$0.049/hr</td>
<td>$0.045/hr</td>
<td></td>
</tr>
<tr>
<td><strong>$1633/36mo</strong></td>
<td>$1270/36mo</td>
<td>$1166/36mo</td>
<td></td>
</tr>
<tr>
<td><strong>Savings</strong></td>
<td>22%</td>
<td>29%</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>On Demand</th>
<th>No Upfront 1 year</th>
<th>Partial Upfront 3 year</th>
<th>All Upfront 3 year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>$ No Upfront</td>
<td>$No Upfront</td>
<td>$337 Upfront</td>
<td>$687 Upfront</td>
</tr>
<tr>
<td>Cost/heure</td>
<td>$0.070/hr</td>
<td>$0.050/hr</td>
<td>$0.0278/hr</td>
<td>$0.00/hr</td>
</tr>
<tr>
<td>Cost/mois</td>
<td>$1840/36mo</td>
<td>$1314/36mo</td>
<td>$731/36mo</td>
<td>$687/36mo</td>
</tr>
<tr>
<td>Savings</td>
<td>29%</td>
<td>60%</td>
<td>63%</td>
<td></td>
</tr>
</tbody>
</table>

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
Capitalize 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
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

Base price is for capacity bought up-front

Traditional application using AWS heavy use reservations
Conservative Compounding

Cloud native application
partially optimized
light use reservations

Base Price  Rightsized  Seasonal  Daily Scaling  Reserved  Tech Refresh  Price Cuts

100  70  50  35  25  20  15
Conservative Compounding

Cloud native application

partially optimized
light use reservations
Conservative Compounding

Cloud native application
partially optimized light use reservations
Conservative Compounding

Cloud native application
partially optimized
light use reservations
Conservative Compounding

Cloud native application partially optimized light use reservations
Agressive Compounding

Cloud native application fully optimized autoscaling mixed reservation use costs 4% of base price over three years!
Cost Monitoring and Optimization
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
Complex dynamic control issues!
Any Questions?

- Battery Ventures [http://www.battery.com](http://www.battery.com)
- Adrian’s Tweets [@adrianco](https://twitter.com/adrianco) and Blog [http://perfcap.blogspot.com](http://perfcap.blogspot.com)
- Slideshare [http://slideshare.com/adriancockcroft](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 - Fast Delivery - Boston March 16th 2015

Disclosure: some of the companies mentioned may be Battery Ventures Portfolio Companies
See [www.battery.com](http://www.battery.com) for a list of portfolio investments