DevOps in action

Yuta Imai

Solutions Architect, Amazon Data Services Japan, K.K.

amazon web services

今日のテーマ

- いままで見てきたお客様や海外事例のなかで「これはおもしろいなー、すごいなー」と感じたDevOpsっぽい話を紹介していきます。

Agenda: Case Studies!

- NetFlix 1/2 : Service Oriented Architecture and Origanization
- A Global Gaming : Application integrated deploy
- MMO RPG : Application integrated deploy
- NetFlix 2/2 : Monitor everything

Case study: Netflix 1/2

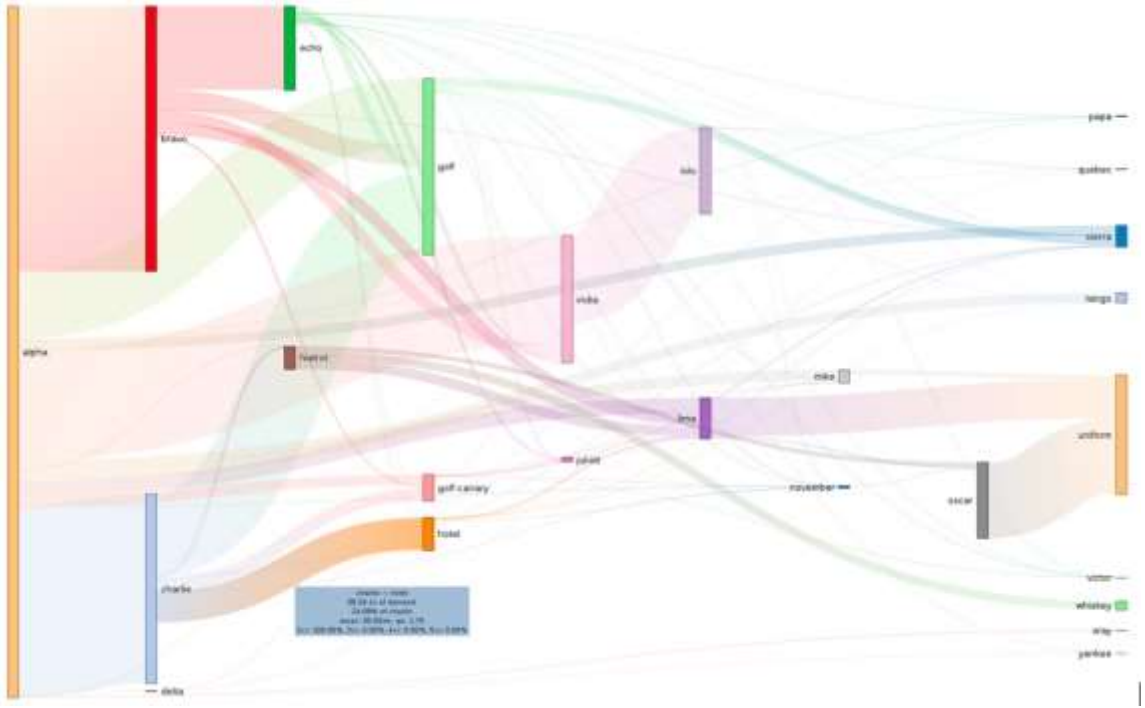
Service Oriented Architecture and Origanization

NetflixMeetup in Kyoto

http://connpass.com/event/9837/

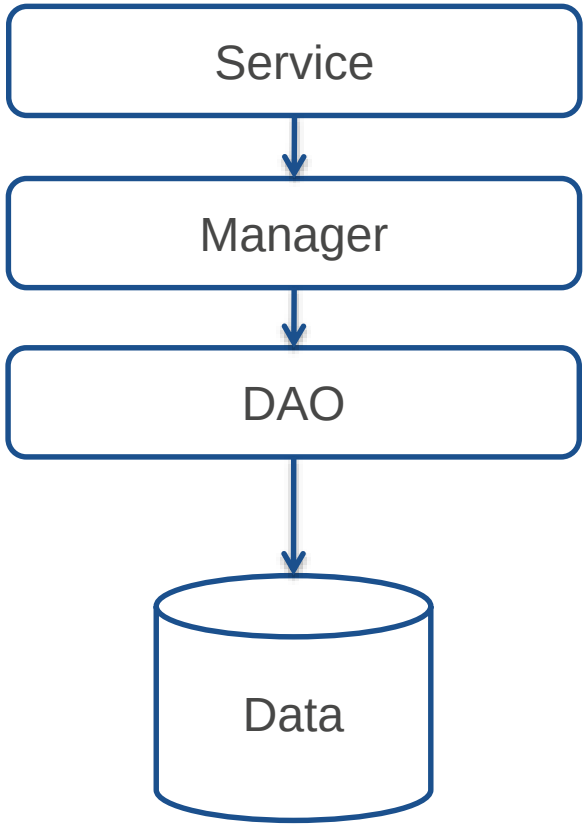
Netflix's way

- Service Oriented Architecture
- Appropriate delegation
- No dependency among teams

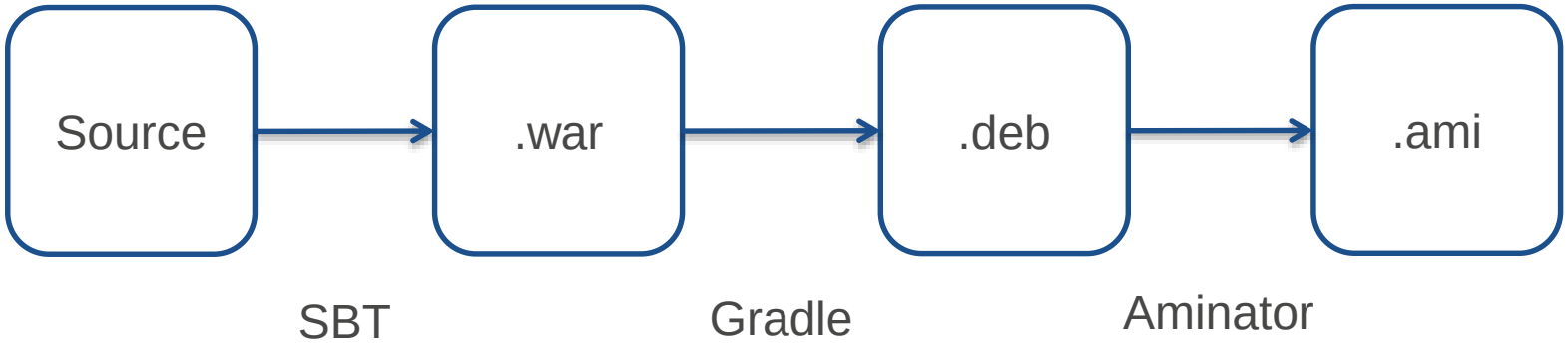


http://techblog.netflix.com/2015/02/a-microscope-on-microservices.html

Components in each Services



Build



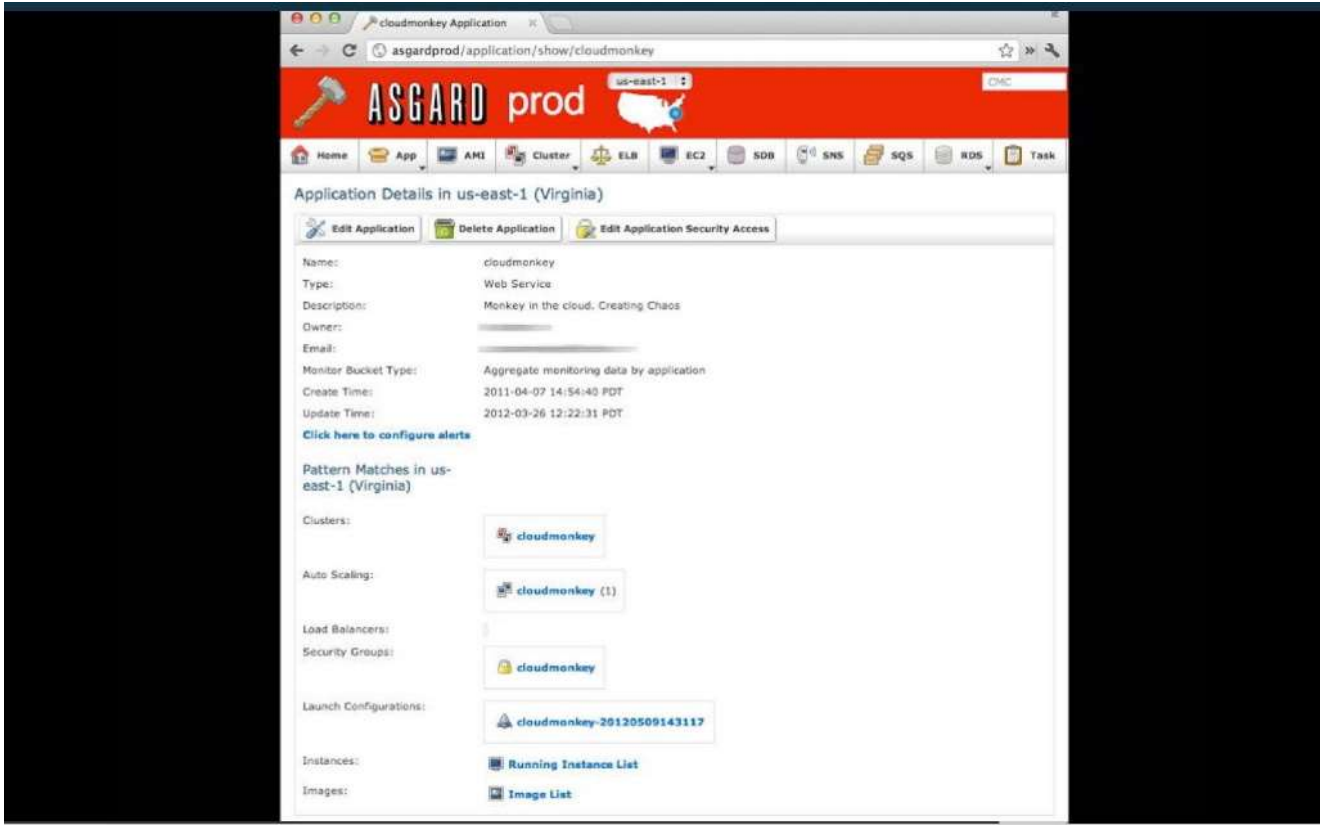
Deploy

1)  → 
dev

2) 
test
→ 
master

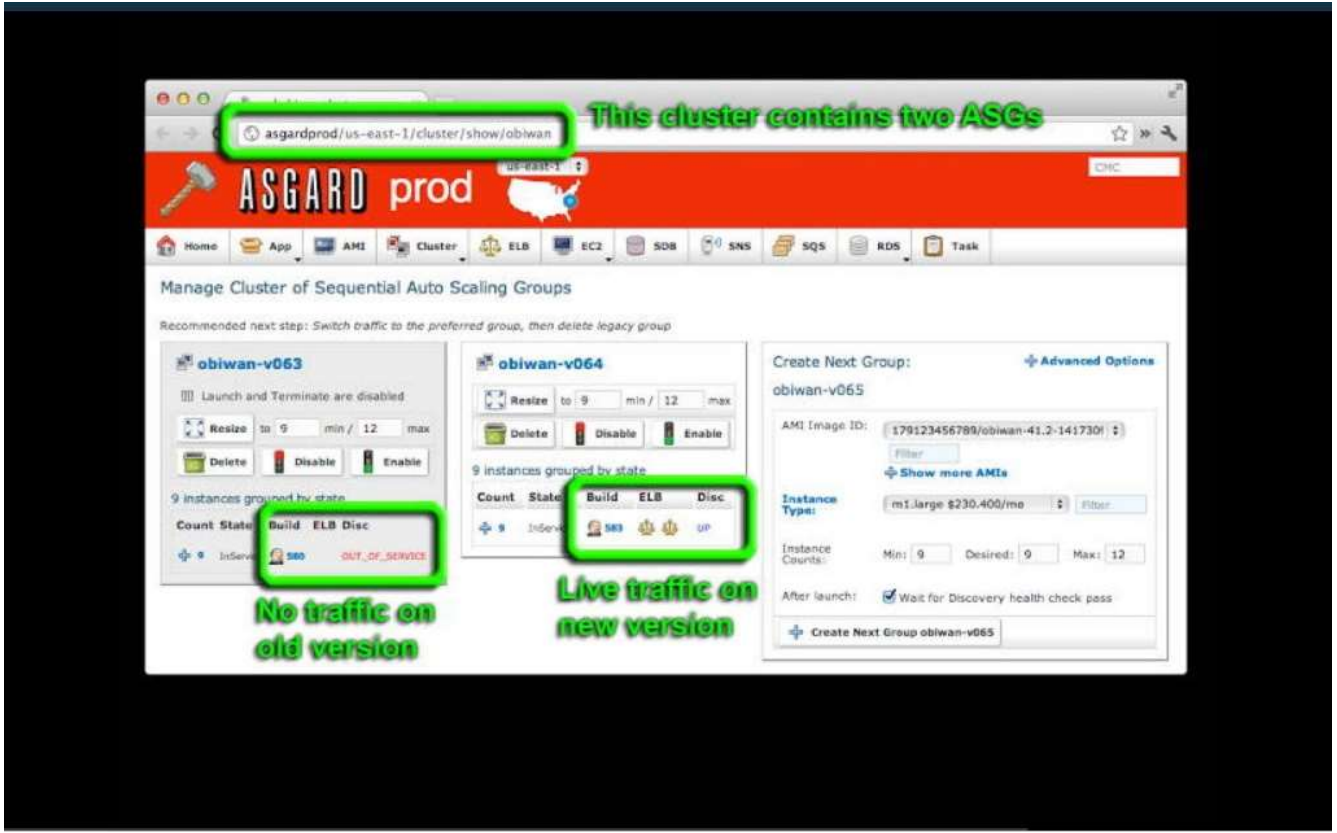
3) 
build
→ 
deploy

Deploy with ASGARD



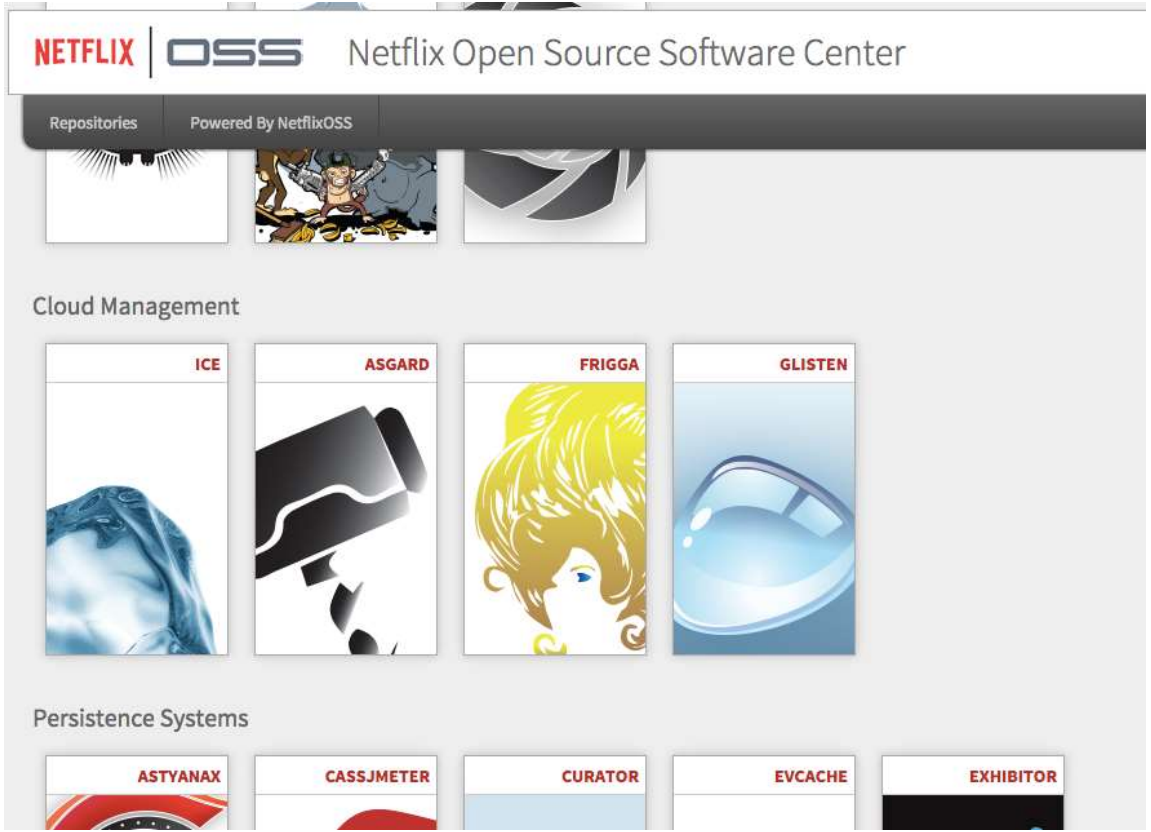
amazon web services

Deploy with ASGARD



More Netflix OSS..

http://netflix.github.io/



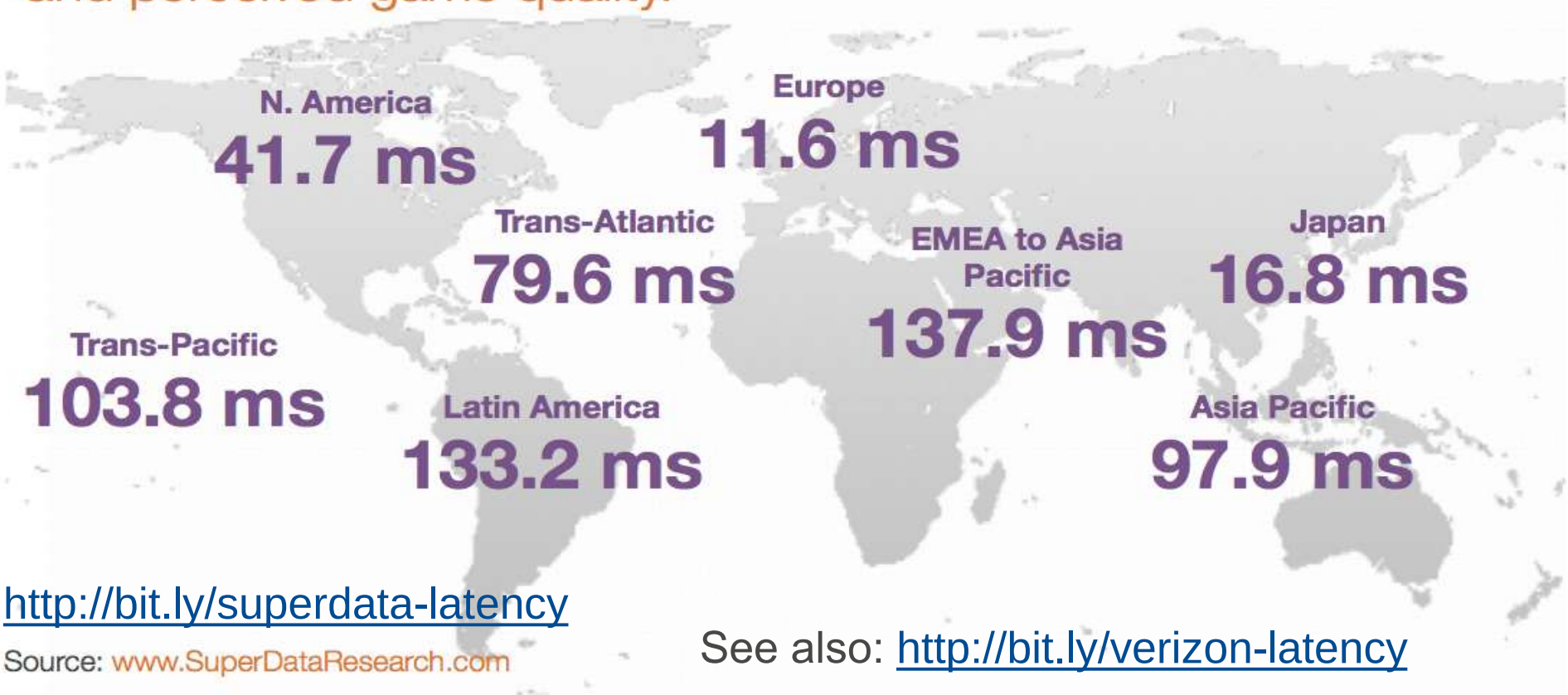
You can try with ready made docker containers.

http://techblog.netflix.com/2014/11/zerotodocker-easy-way-to-evaluate.html

Case study: A Global Gaming
Application integrated deploy

Domestic latency in selected regions:

"higher latency decreases game response time, user performance and perceived game quality."

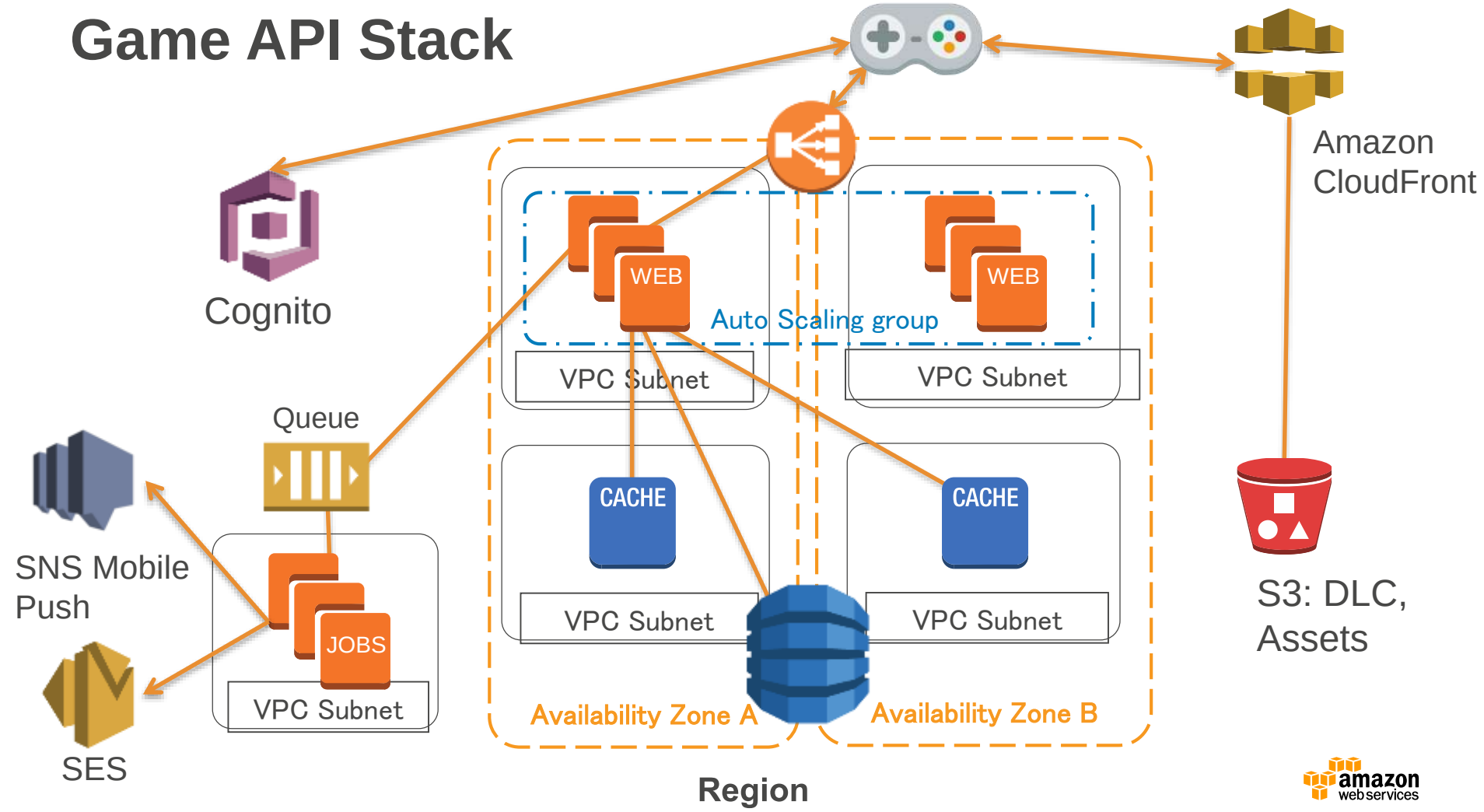


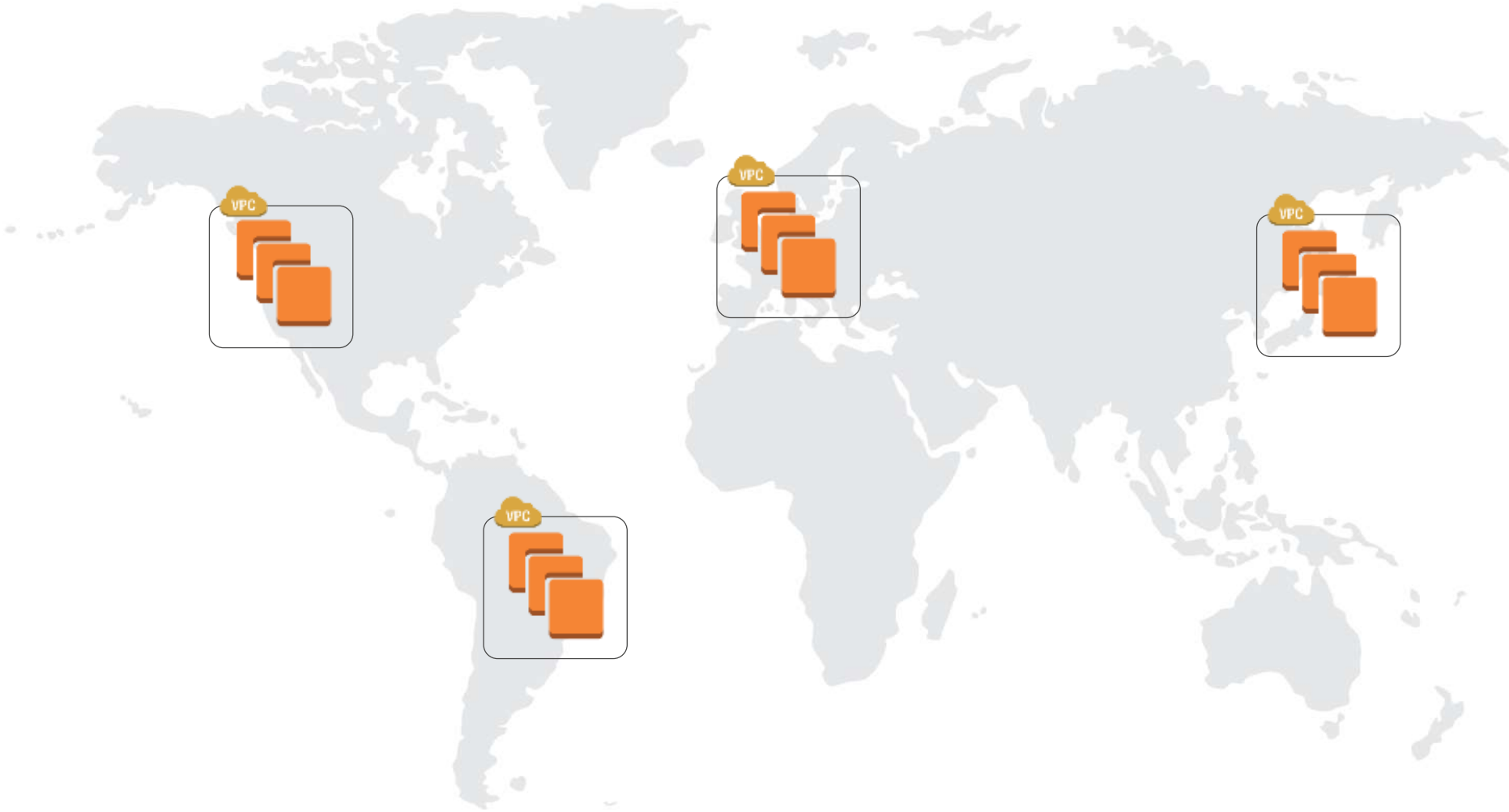
http://bit.ly/superdata-latency

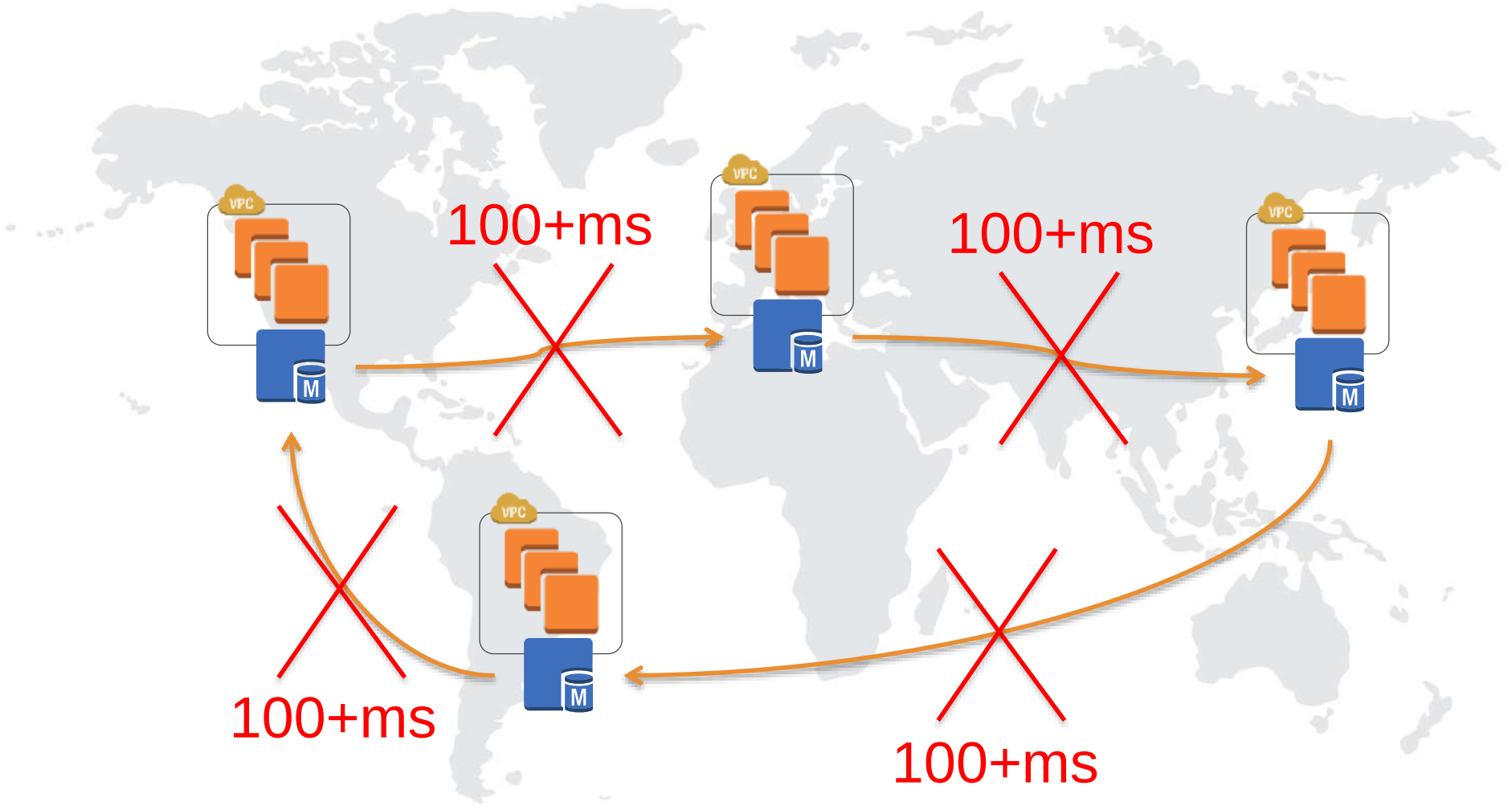
Source: www.SuperDataResearch.com

See also: http://bit.ly/verizon-latency

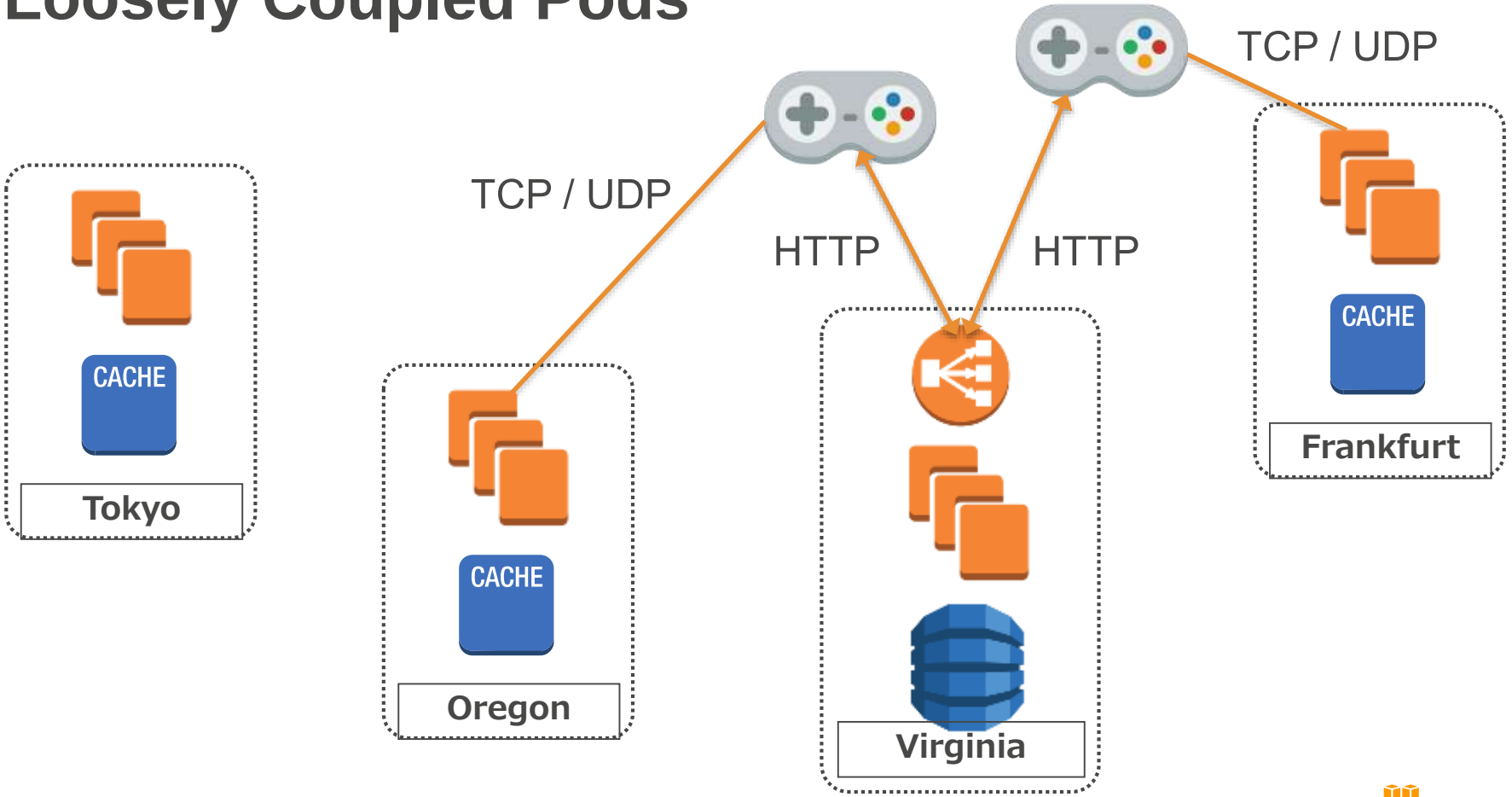
Game API Stack




VPC
VPC
VPC
VPC

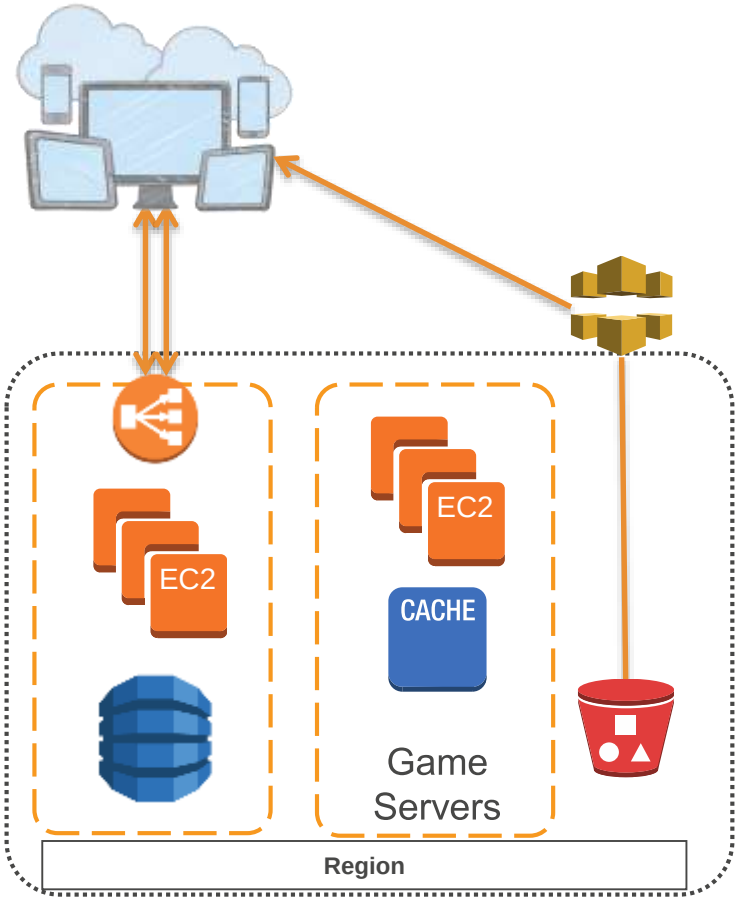

VPC
M
100+ms
VPC
M
100+ms
VPC
M
VPC
M
100+ms
100+ms

Loosely Coupled Pods



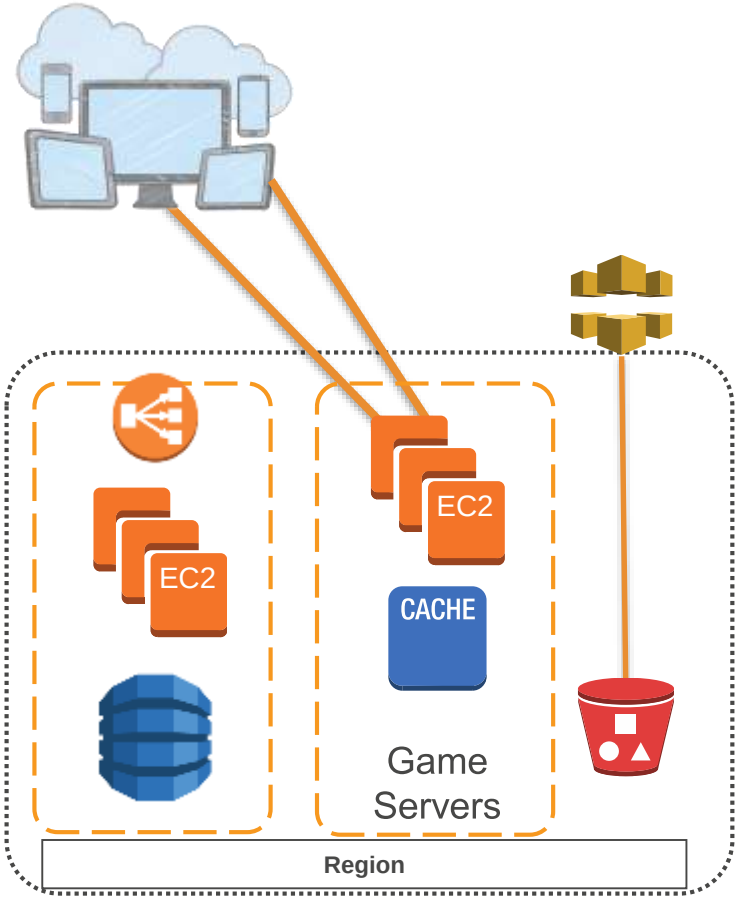
Game Flow

① Login via HTTP API
② Download Game Assets
③ Matchmaking to Game Server



Game Flow

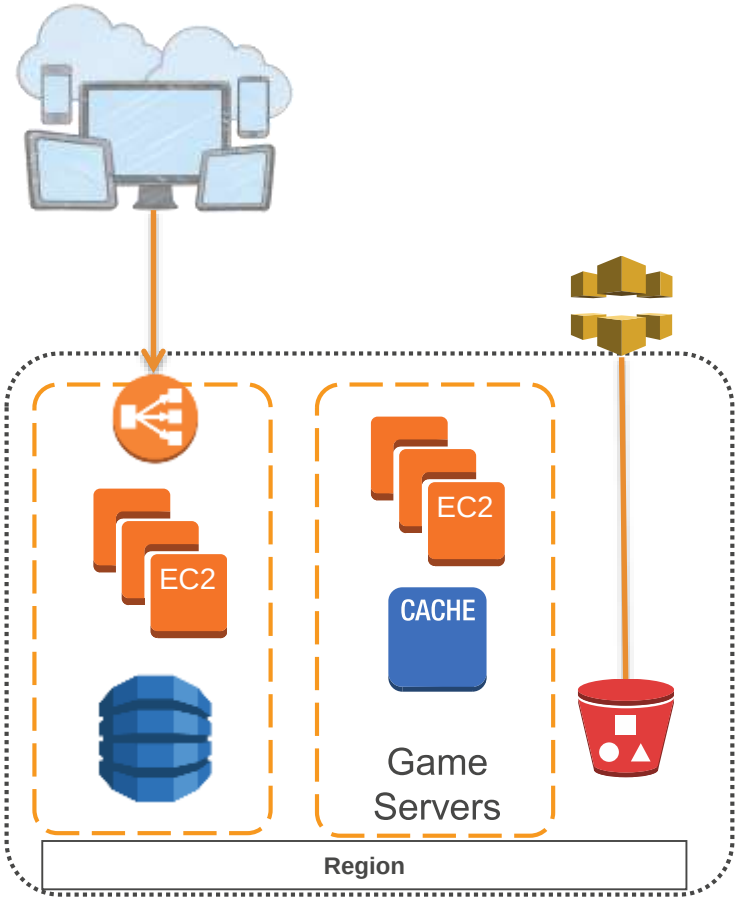
① Login via HTTP API
② Download Game Assets
③ Matchmaking to Game Server
④ Connect to Server
⑤ Hack Apart Your Friends
⑥ Game Over



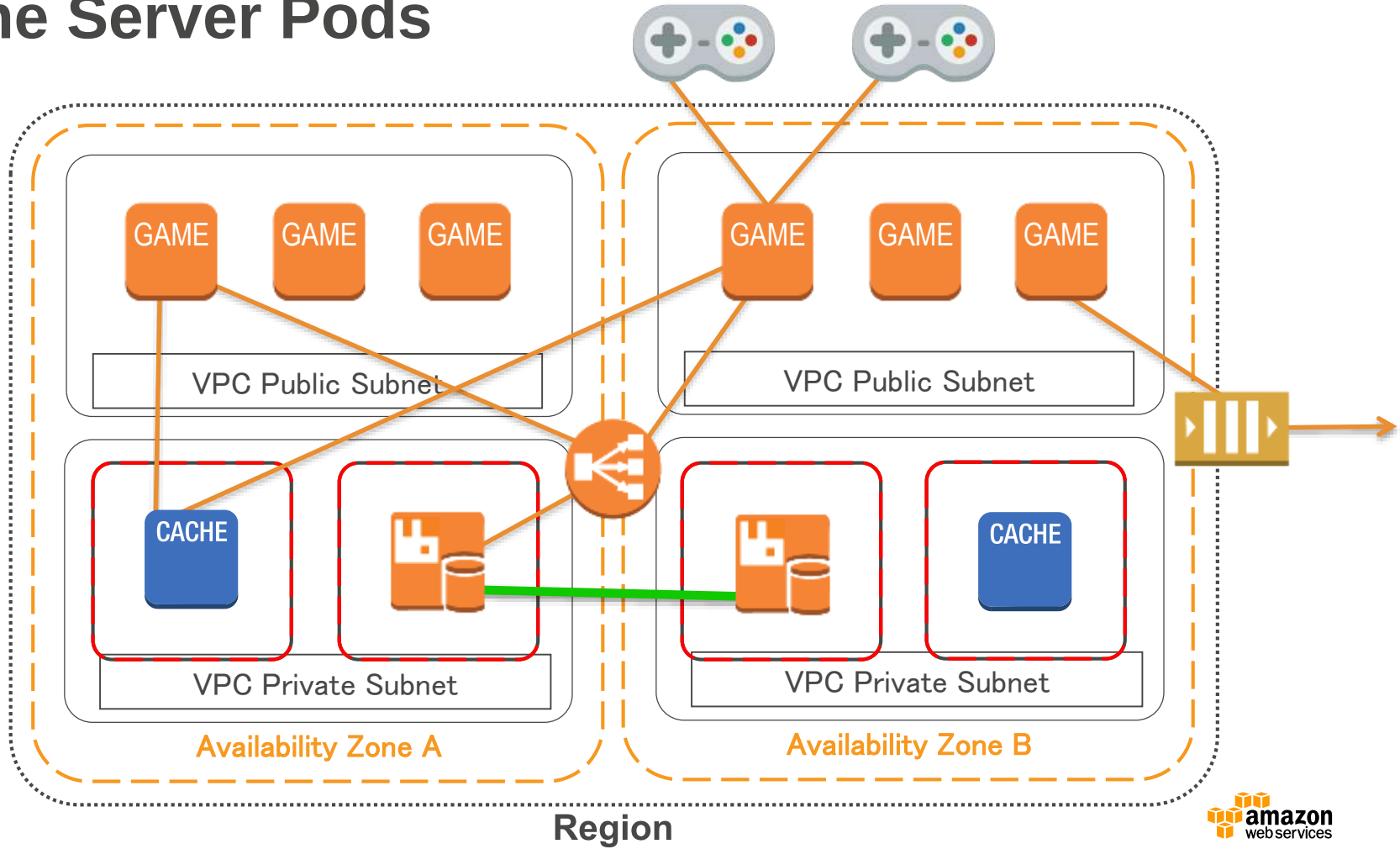
amazon web services

Game Flow

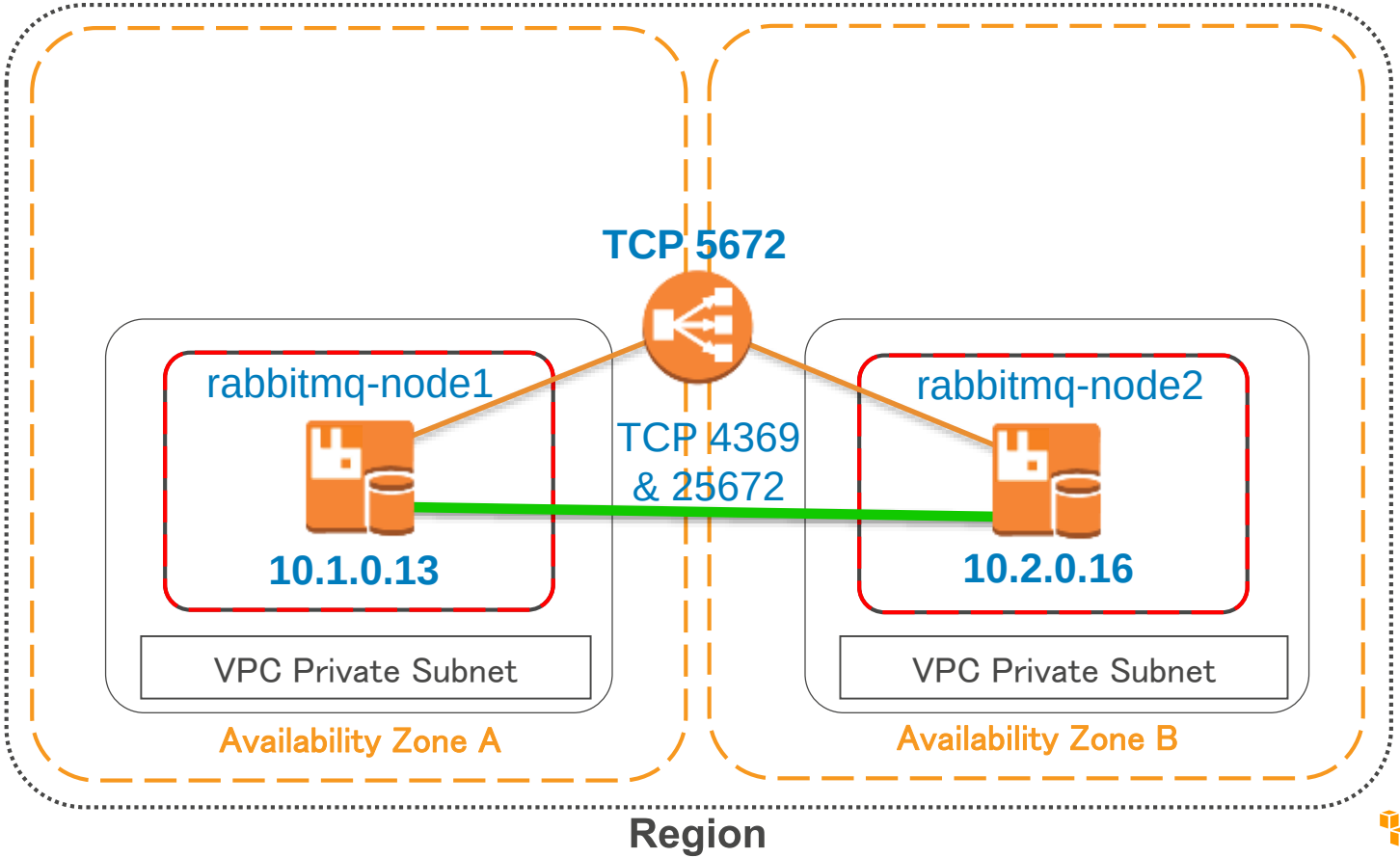
① Login via HTTP API
② Download Game Assets
③ Matchmaking to Game Server
④ Connect to Server
⑤ Hack Apart Your Friends
⑥ Game Over
⑦ Write via HTTP API



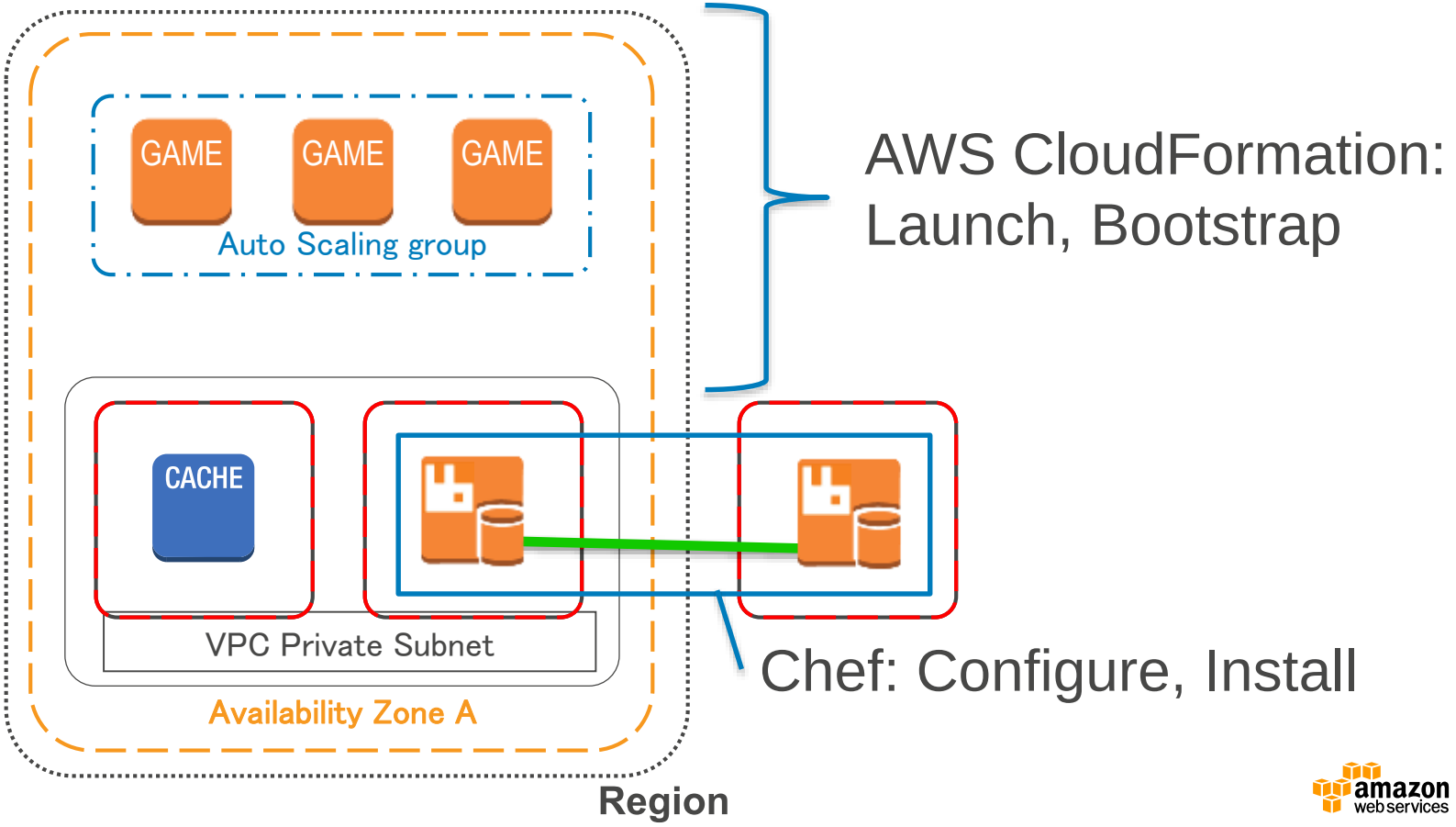
Game Server Pods



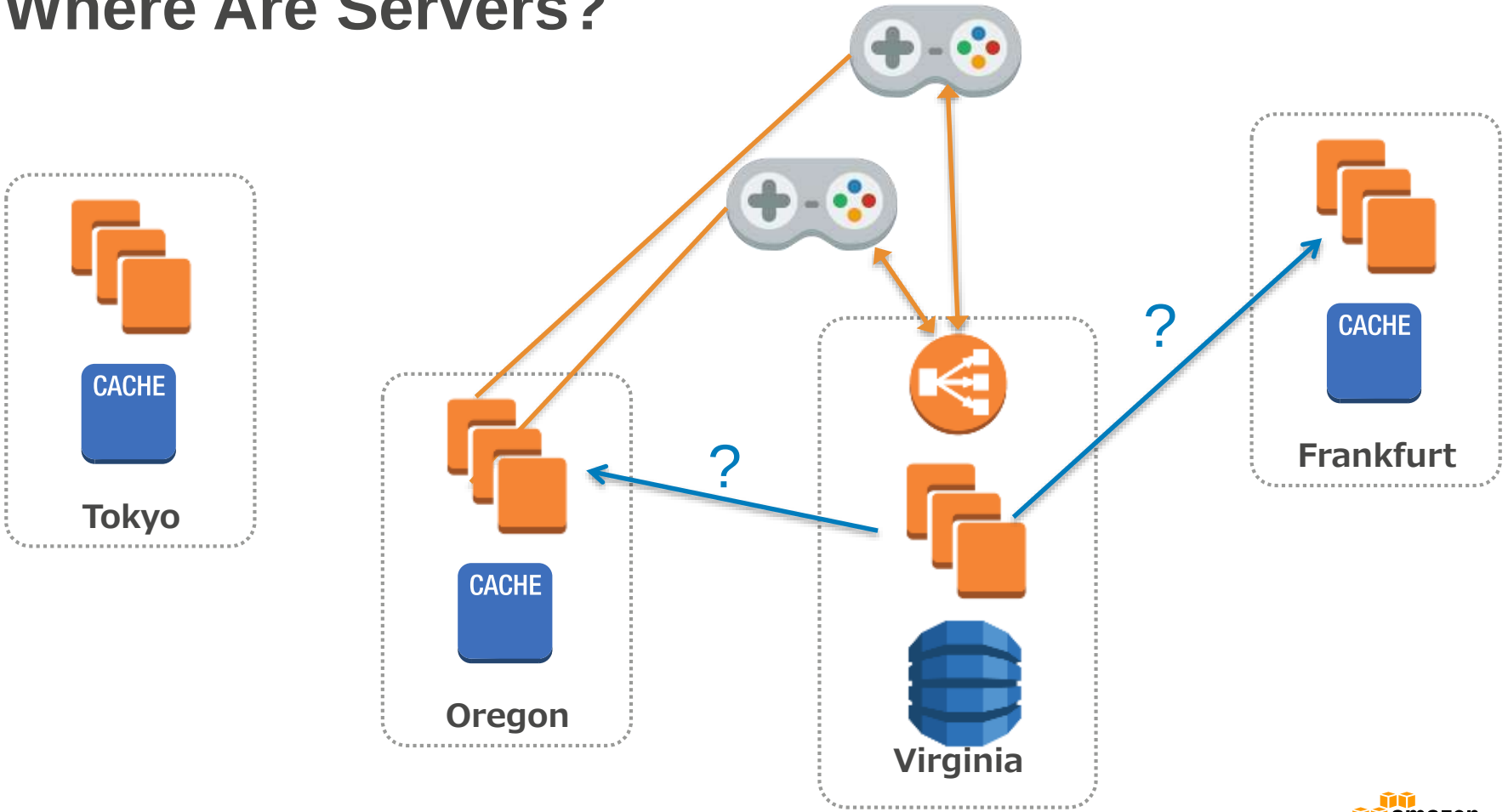
RabbitMQ + Elastic Load Balancing



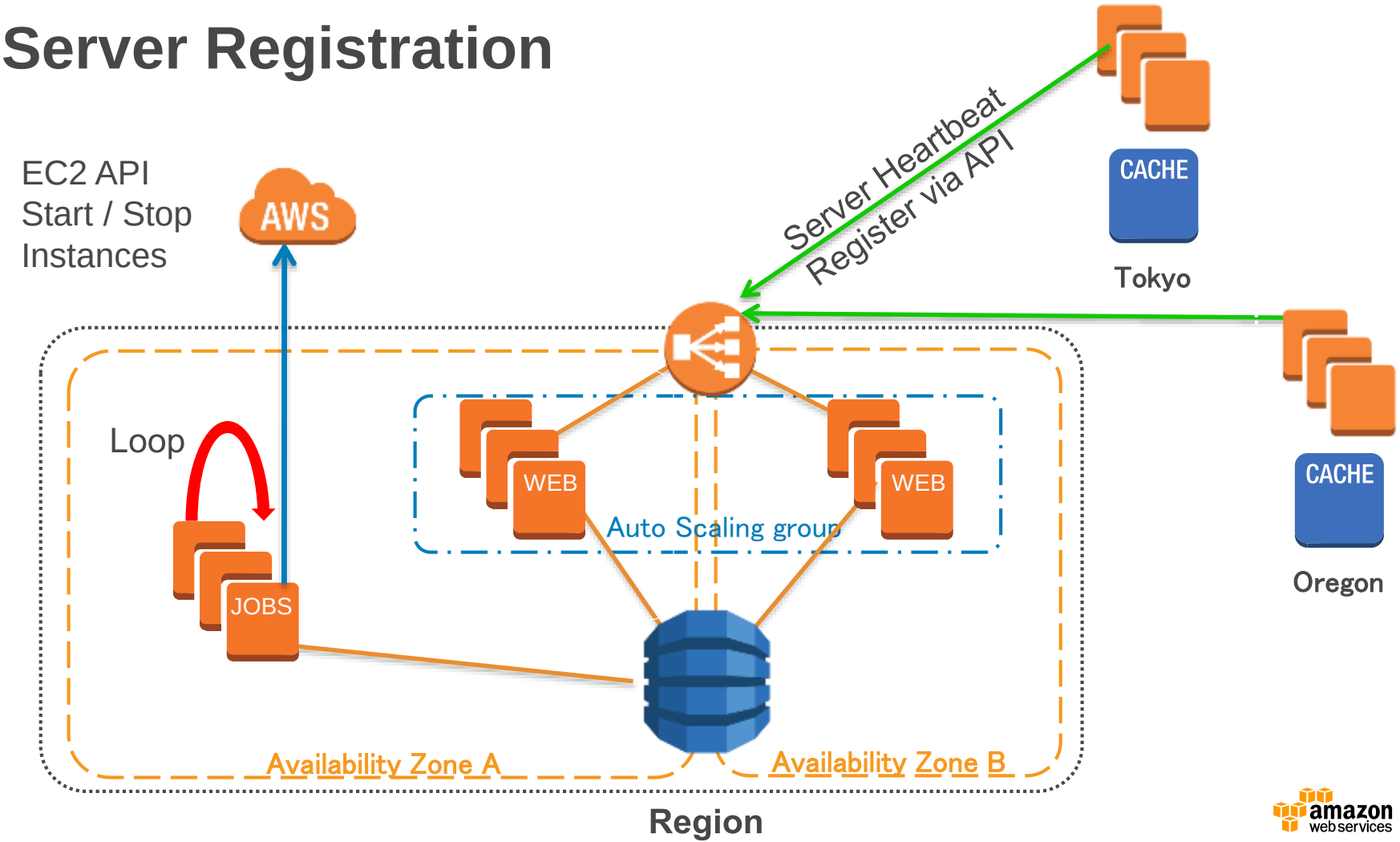
CloudFormation + Chef



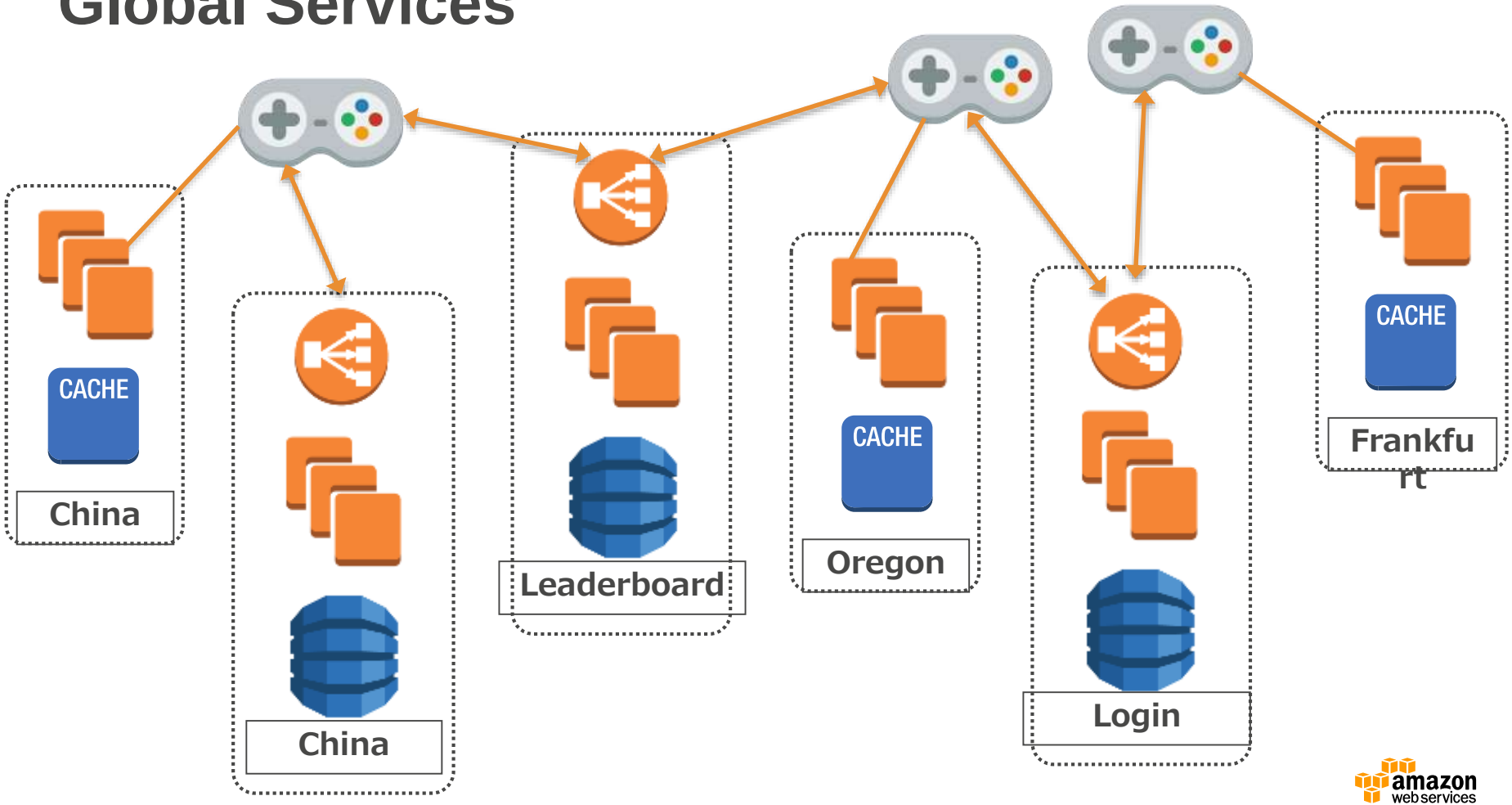
Where Are Servers?



Server Registration



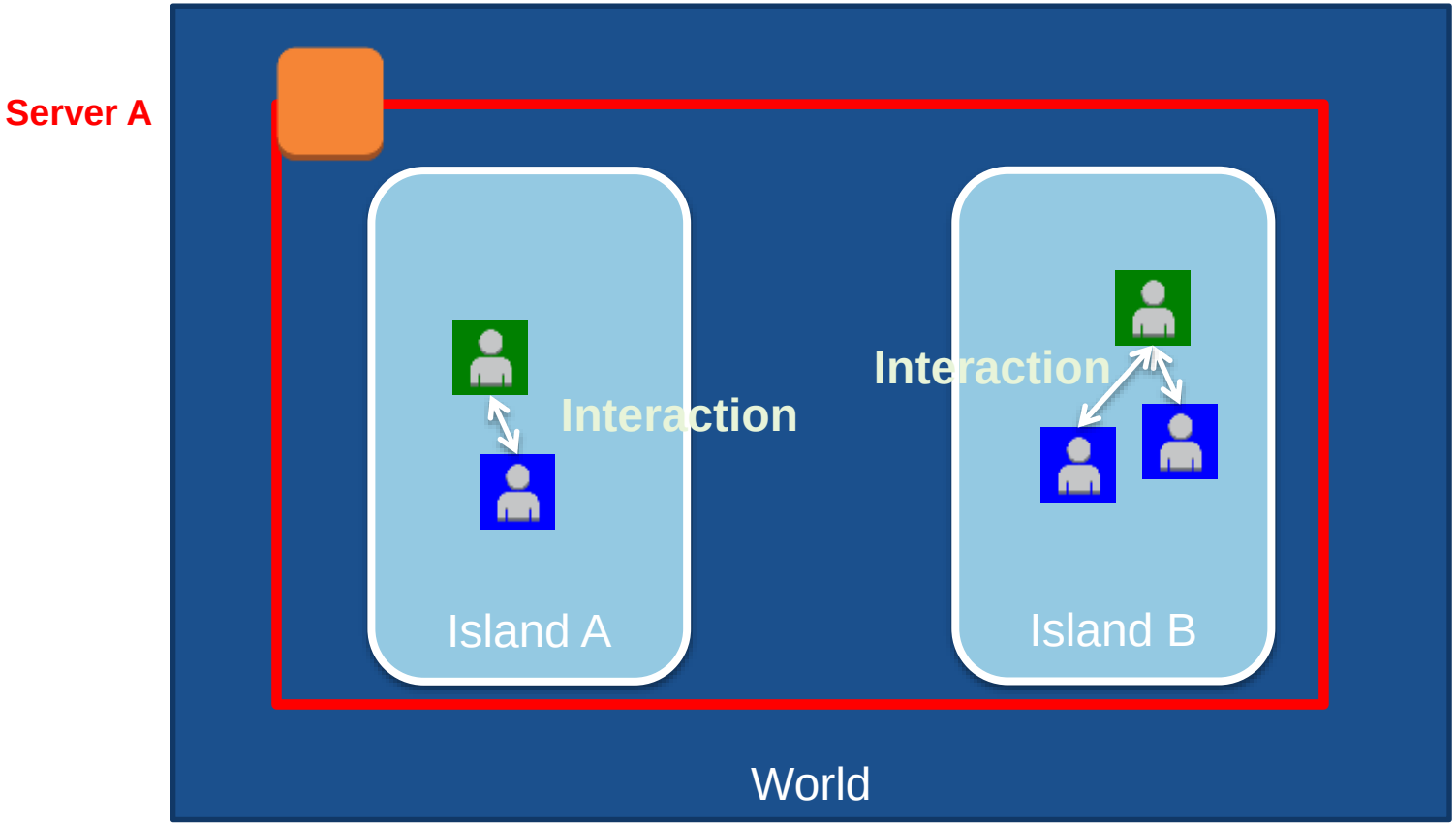
Global Services



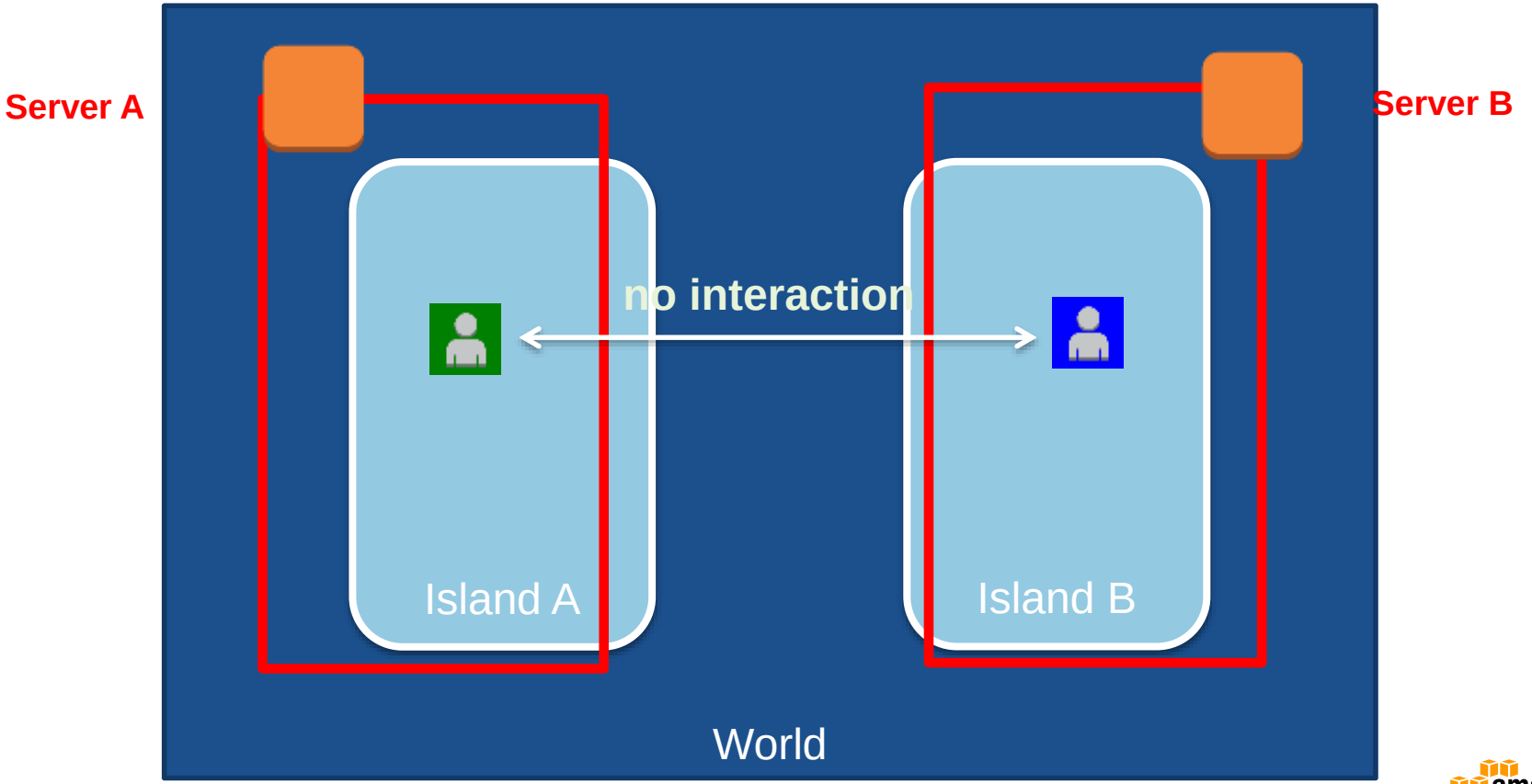
amazon web services

Case study: MMO RPG
Application integrated deploy

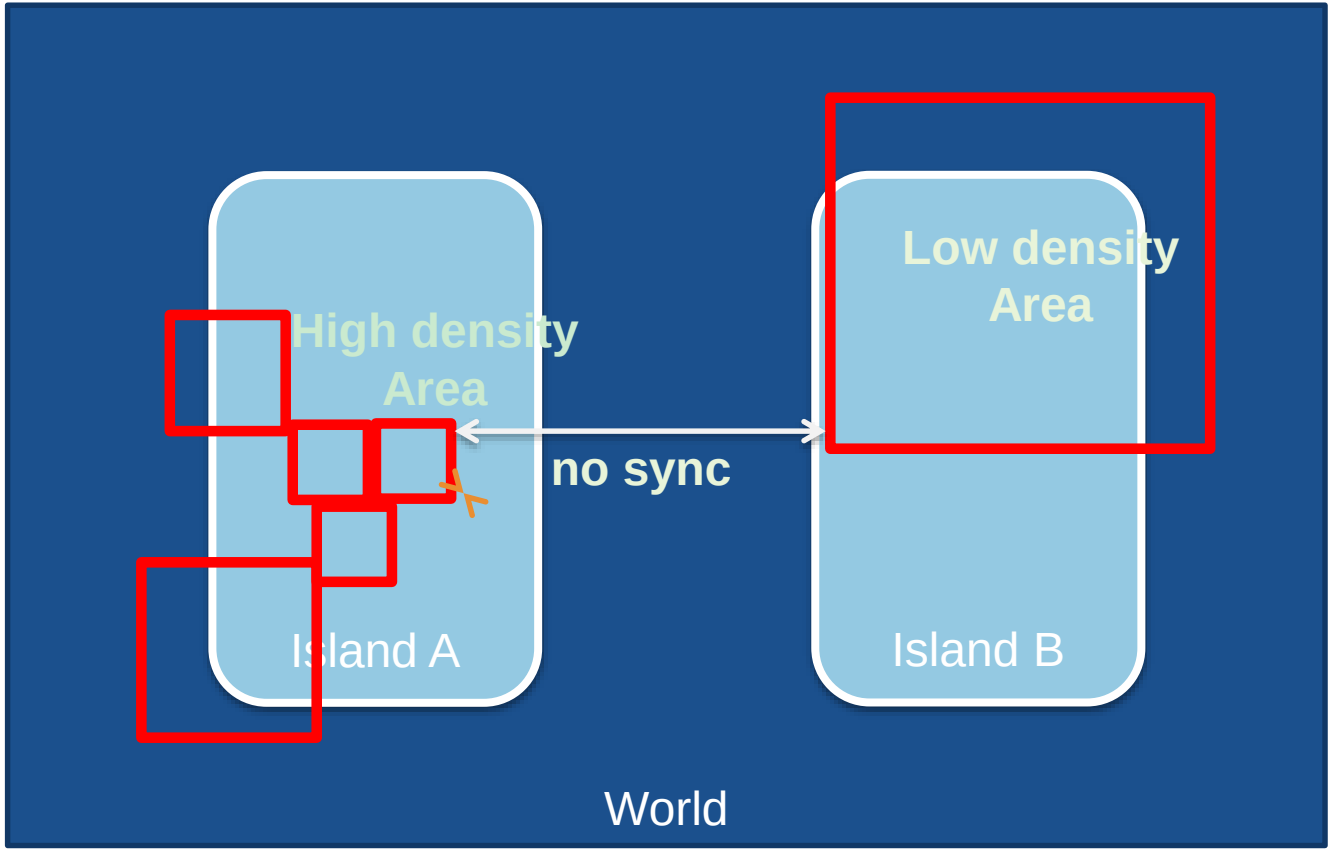
Usual way of MMO implementation: Single big server



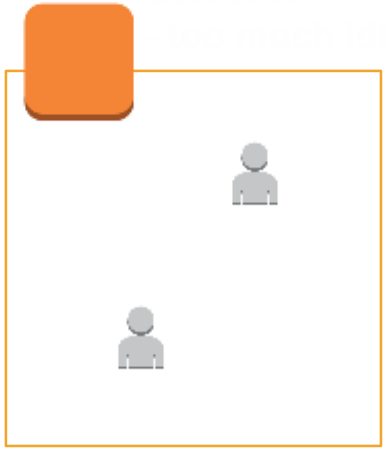
Distribute users by geo location in game map



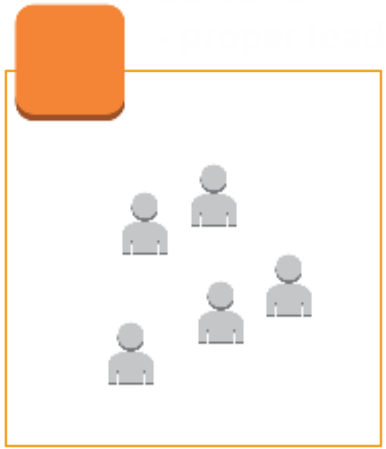
Distribute hot area with more game servers



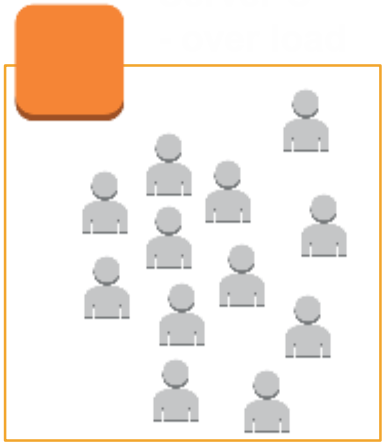
User density differentiated by map's popularity



Area A
less monsters
low experience points
poor item drop rate

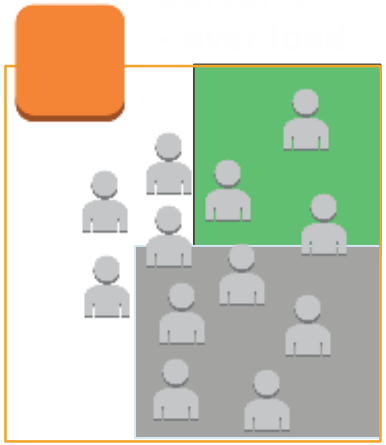


Area B
medium experience points
enough gold drops
avg. item drop rate



Area C
boss monsters
high chance of get items

When an area's population get higher



Area C
boss monsters
high chance of get items

Geo size
3

Server C-1
- newly created

Server C-2
- newly created

Server C
- already existed

Case study: Netflix 2/2

Monitor everything

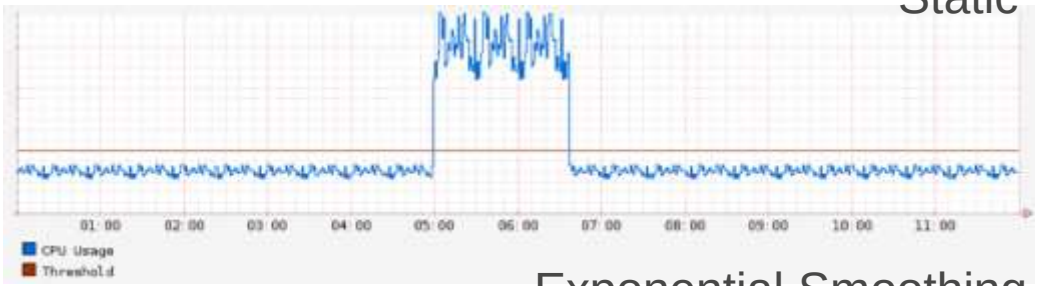
Monitor everything

- Let’s monitor everything
  - User behavior
  - Application log
  - Access log
- Then make it available!

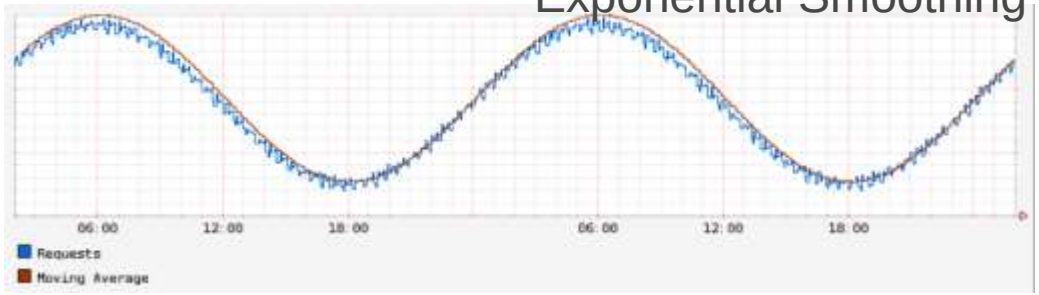
SPS : the Pulse of Netflix Streaming

- Starts Per Second
- They use SPS to detect issues
- Deviation modeling is important.

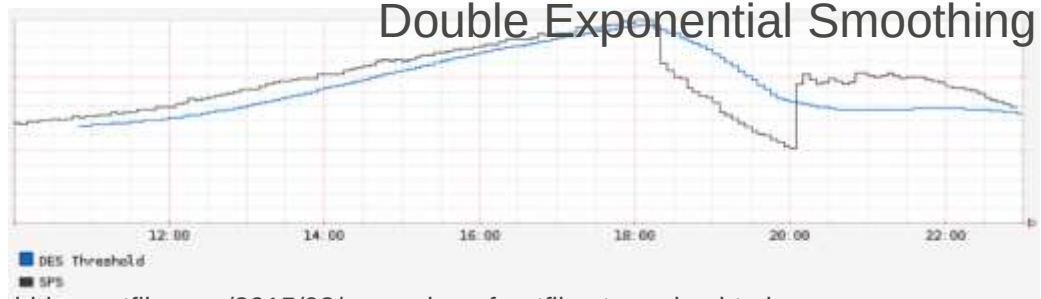
Static



Exponential Smoothing

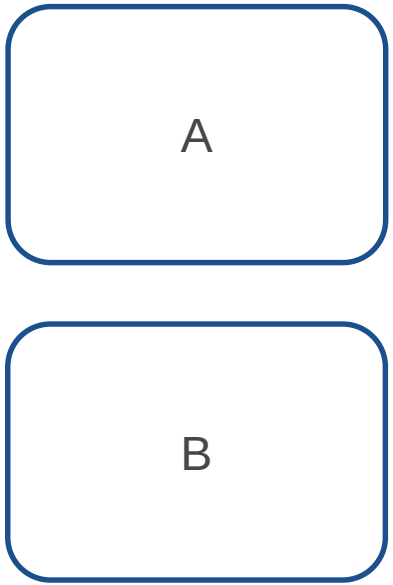


Double Exponential Smoothing

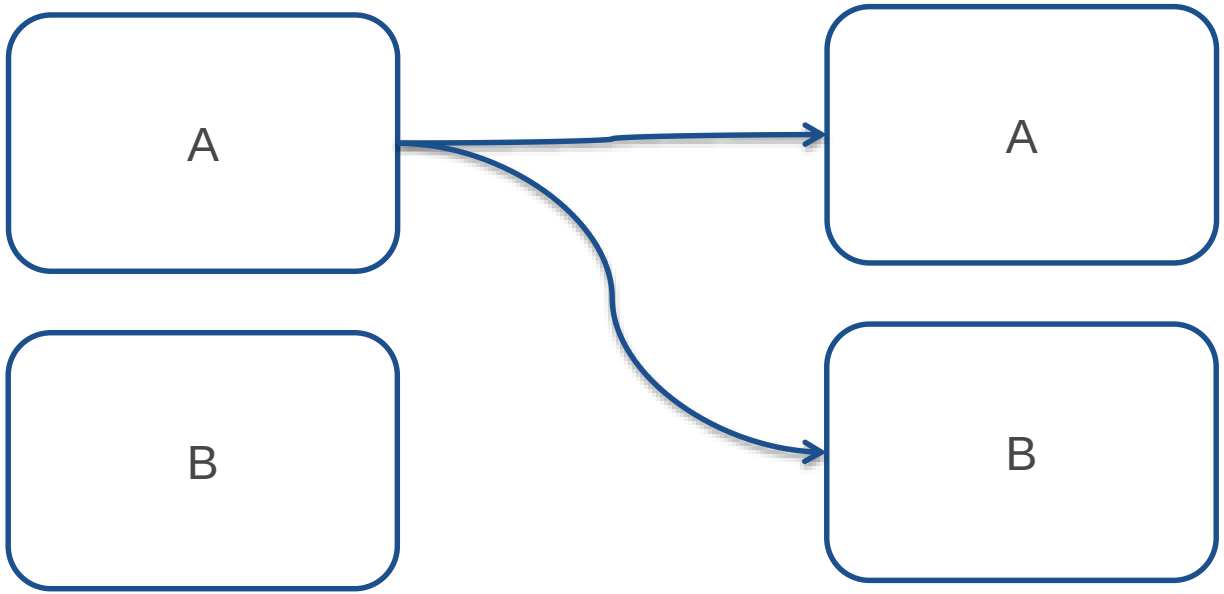


http://techblog.netflix.com/2015/02/sps-pulse-of-netflix-streaming.html

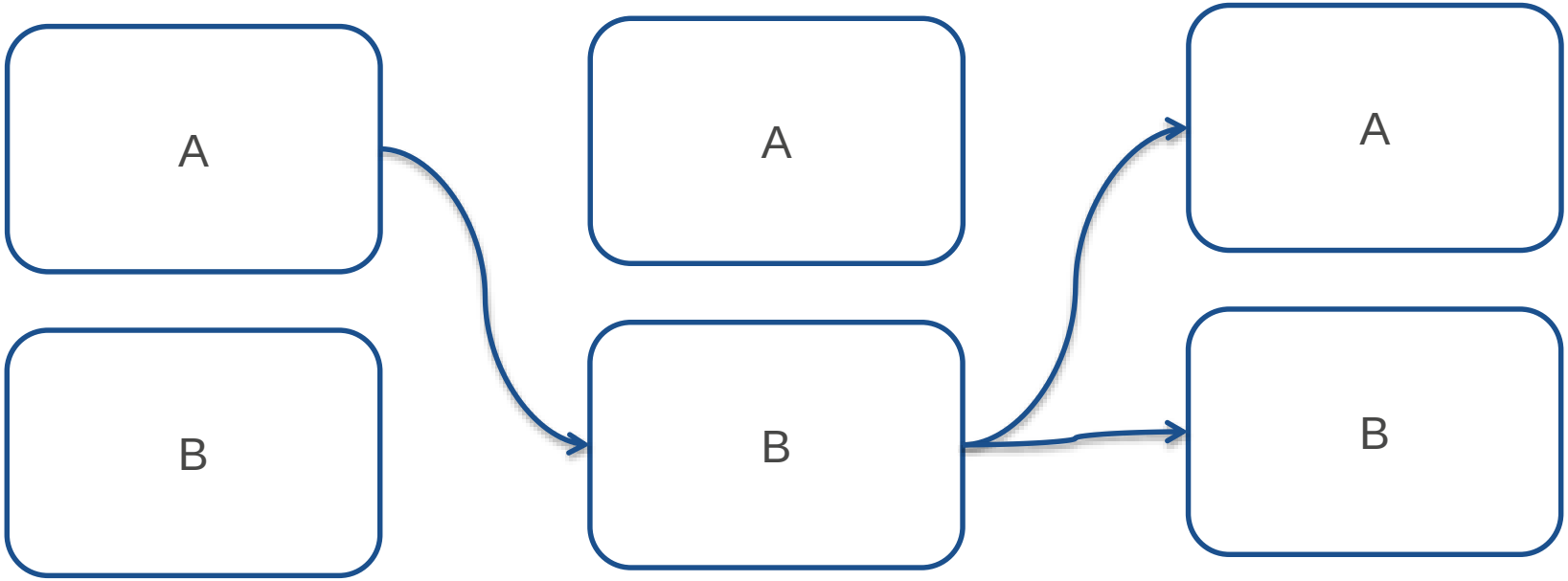
A/B Testing



A/B Testing



A/B Testing



A/B Testing

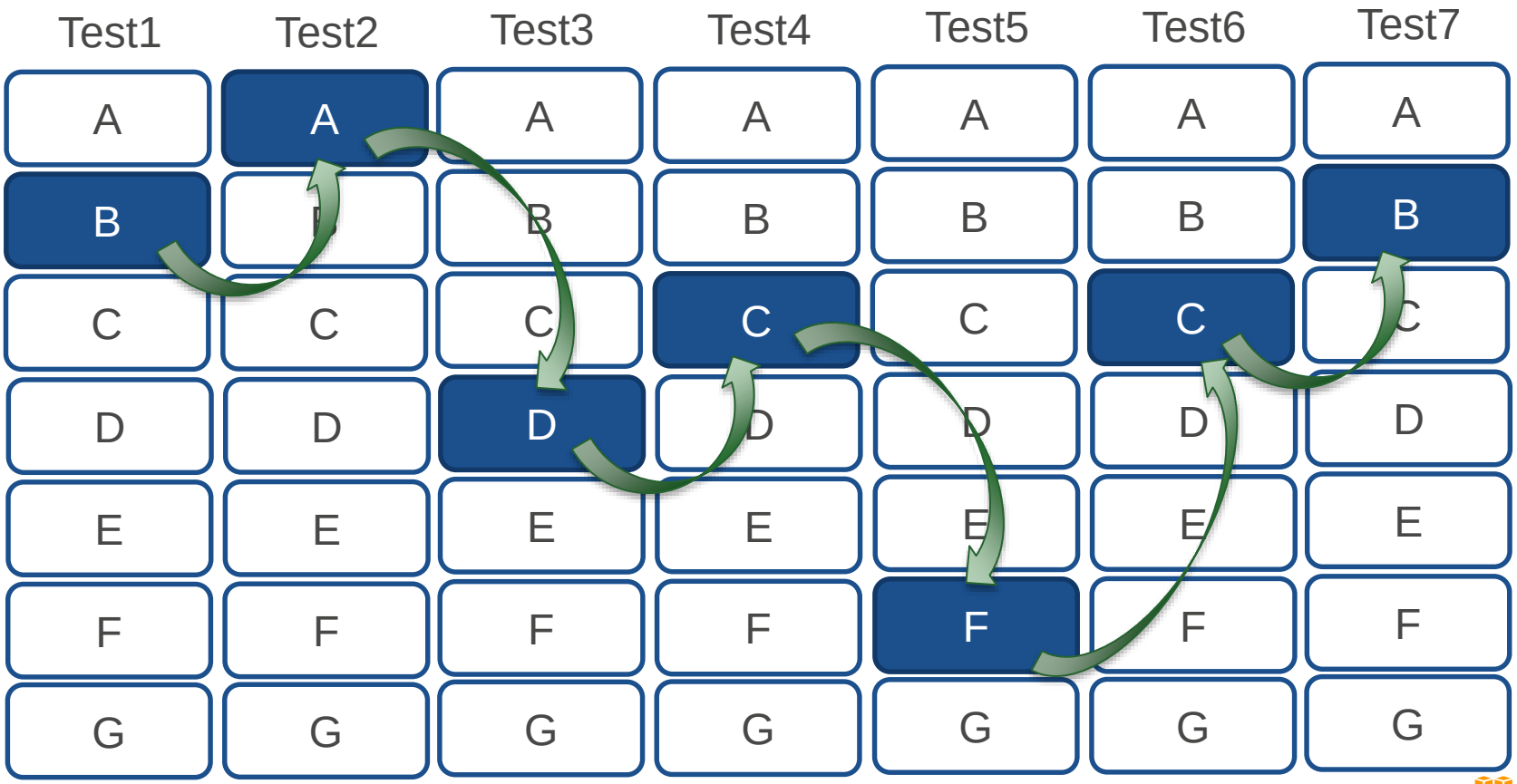
Test1	Test2	Test3	Test4	Test5	Test6	Test7
A	A	A	A	A	A	A
B	B	B	B	B	B	B
C	C	C	C	C	C	C
D	D	D	D	D	D	D
E	E	E	E	E	E	E
F	F	F	F	F	F	F
G	G	G	G	G	G	G

Huge numbers in A/B testing

- 2,097,152 unique experience across 7 tests
- Hundreds of new A/B tests per year
- 2,105 Templates, 566 CSS, 685 JS
- 2.5 million unique css/js packages per push cycle

This cannot be handled manually!

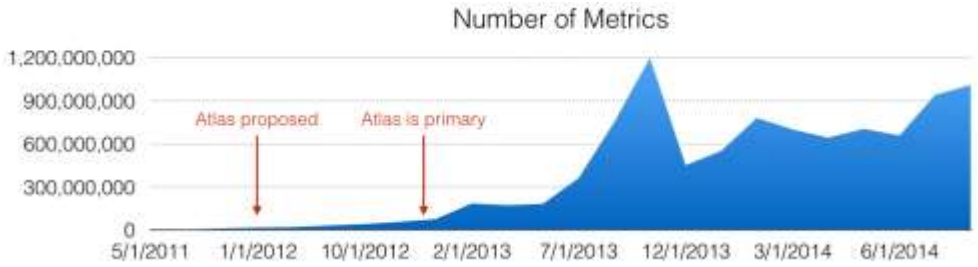
A/B tests should be automatically convergent

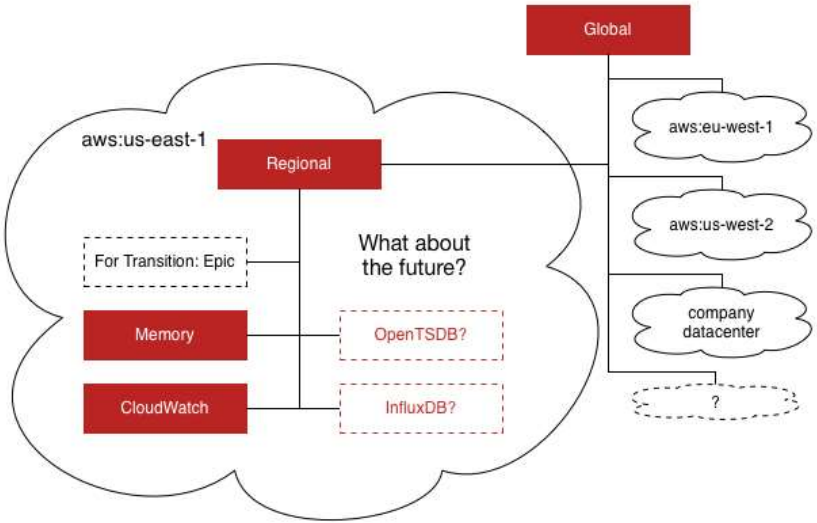


A/B tests should be automatically convergent: How?

- Monitor metrics
- Then
  - Deploy multiple versions of application
  - Configuration

Atlas: Telemetry Platform

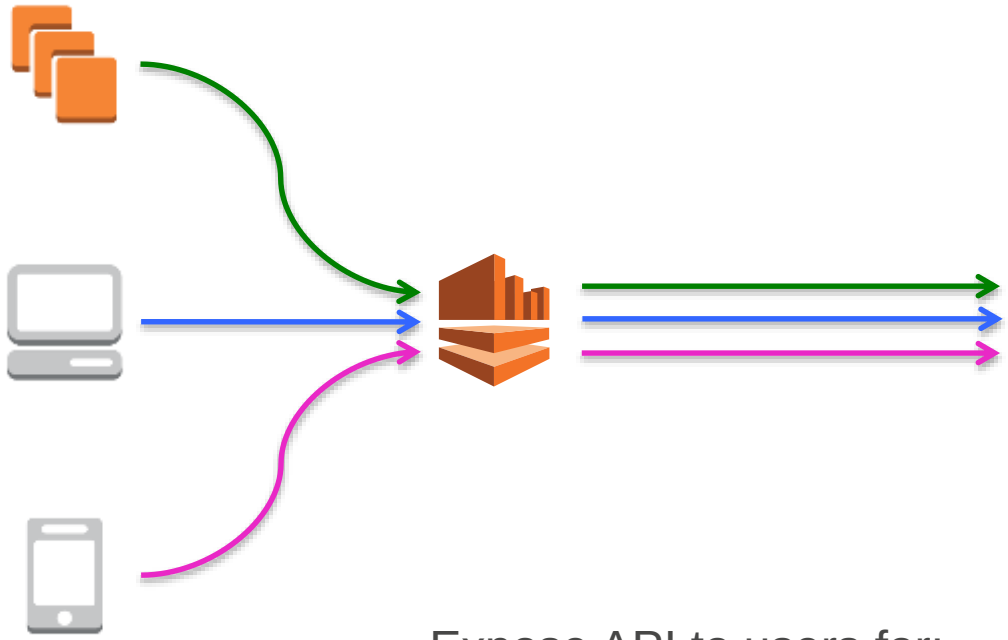




http://techblog.netflix.com/2014/12/introducing-atlas-netflixs-primary.html

amazon web services

Let’s build data stream/pipeline



Expose API to users for:

- Adding new data source
- Adding new application

Conclusion

Conclusion

- Infra team should become ‘Tool team’
- Many things to go other than Test, Build, Deploy
- DevOps affect software architecture