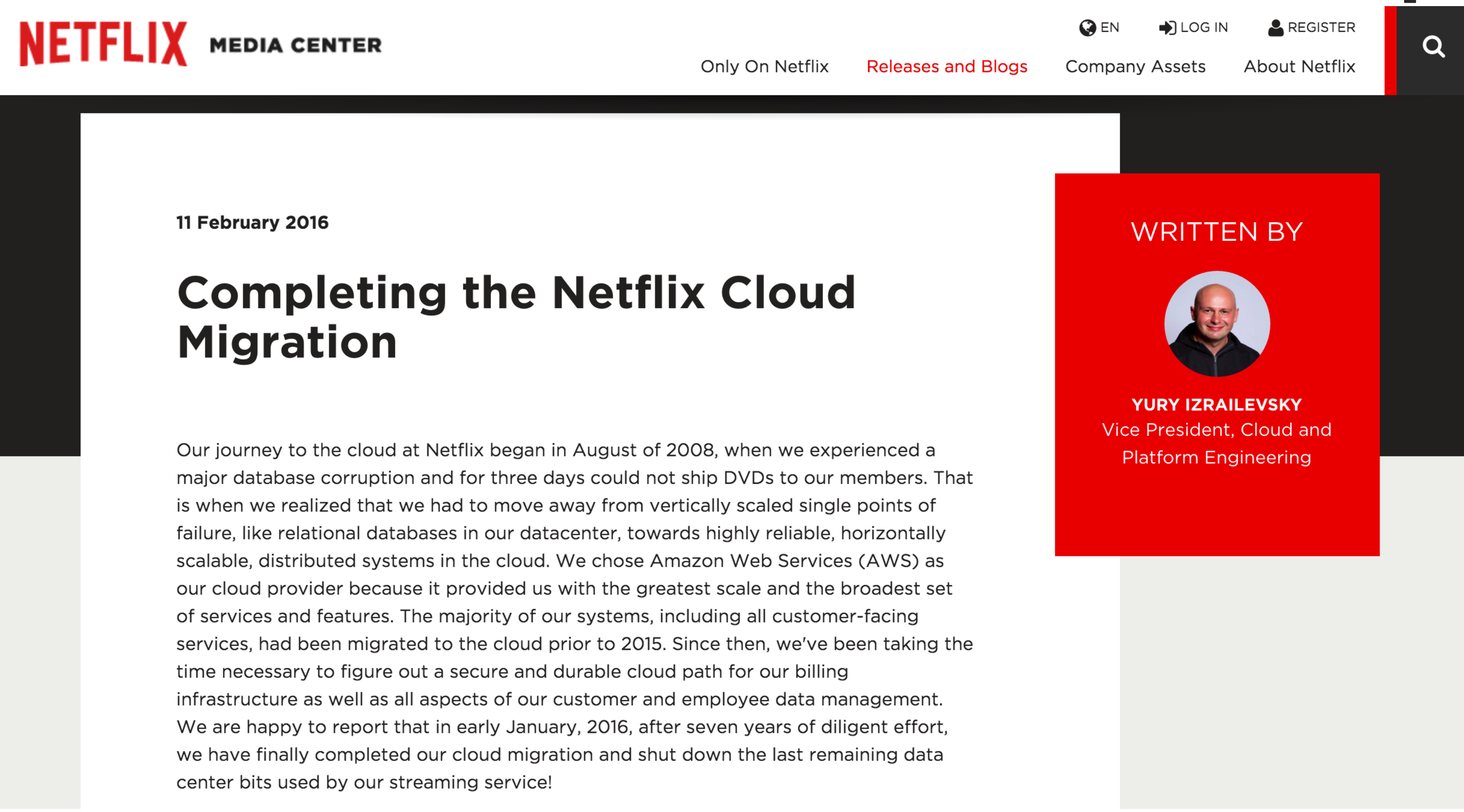
Cloud Computing

ECPE 276

Netflix Case Study



11 February 2016

Completing the Netflix Cloud Migration

WRITTEN BY

YURY IZRAILEVSKY
Vice President, Cloud and Platform Engineering

Our journey to the cloud at Netflix began in August of 2008, when we experienced a major database corruption and for three days could not ship DVDs to our members. That is when we realized that we had to move away from vertically scaled single points of failure, like relational databases in our datacenter, towards highly reliable, horizontally scalable, distributed systems in the cloud. We chose Amazon Web Services (AWS) as our cloud provider because it provided us with the greatest scale and the broadest set of services and features. The majority of our systems, including all customer-facing services, had been migrated to the cloud prior to 2015. Since then, we've been taking the time necessary to figure out a secure and durable cloud path for our billing infrastructure as well as all aspects of our customer and employee data management. We are happy to report that in early January, 2016, after seven years of diligent effort, we have finally completed our cloud migration and shut down the last remaining data center bits used by our streaming service!

https://media.netflix.com/en/company-blog/completing-the-netflix-cloud-migration

Netflix & The Cloud

- Cloud provider: AWS
- Services at AWS
  - Business logic, billing, payments
  - Distributed databases
  - Big data processing/analytics
  - Recommendations
  - Transcoding
- Video streaming *not* at AWS
  - Netflix *Open Connect* CDN
    (Netflix-owned servers co-located at/near customer ISPs)

Netflix Scale

- 2015 *Sandvine* report
- Streaming video and audio traffic now accounts for **over 70%** of North American downstream traffic
  - Peak evening hours, fixed access networks (not wireless)
- Top 3 sources of video traffic?
  - **Netflix - 37.1%**
  - YouTube - 17.9%
  - Amazon Video - 3.1%

Math:
70% * 37.1% =
25.9% of evening traffic is *Netflix*

https://www.sandvine.com/pr/2015/12/7/sandvine-over-70-of-north-american-traffic-is-now-streaming-video-and-audio.html

Cloud Migration Timeline

- **7 years migration** from private datacenter to cloud
- Migration strategy **option "A"**:
  Configure AWS to match your current datacenter
  - Equivalent EC2 nodes
  - Equivalent network topology
    ("Virtual Private Cloud")
  - Run the same applications you already do
  - **Pros? Cons?**

Cloud Migration Timeline

- **7 years migration** from private datacenter to cloud
- Migration strategy **option "B"**:
  Rebuild your applications and corporate processes to follow cloud norms
  - Monolithic app -> Micro applications (e.g. *Lambda*)
  - Relational DB -> NoSQL database (e.g. *DynamoDB*)
  - Budget approvals / procurement -> Self-service tools for new cloud nodes
  - **Pros? Cons?**

Cloud Migration Timeline

- Netflix chose option "B" (hence, the 7 years to migrate)

AWS re:Invent 2015 | (DVO203)

A Day in the Life of a Netflix Engineer using 37% of the Internet

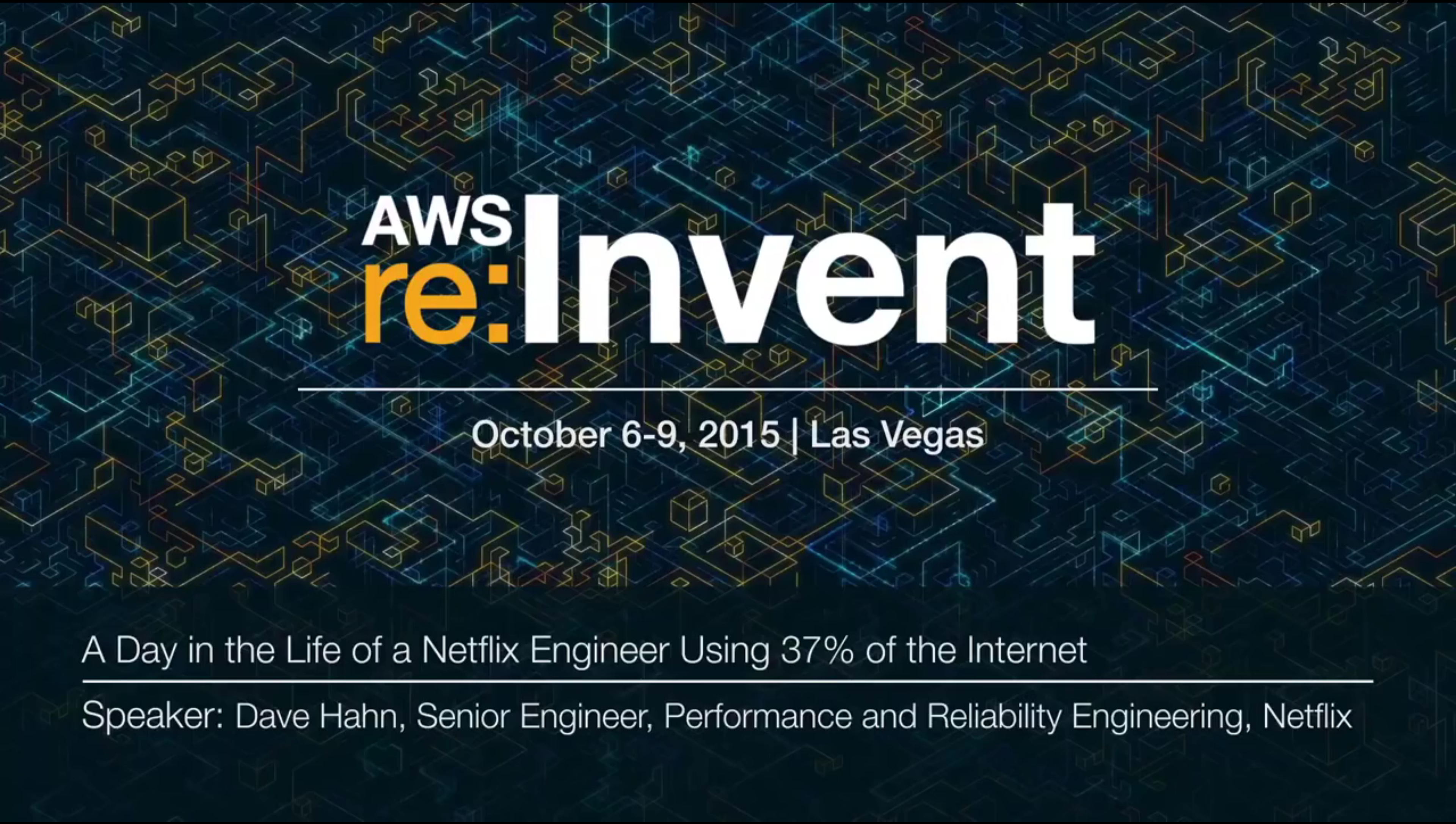
AWS re:Invent

October 6-9, 2015 | Las Vegas

A Day in the Life of a Netflix Engineer Using 37% of the Internet

Speaker: Dave Hahn, Senior Engineer, Performance and Reliability Engineering, Netflix

https://www.youtube.com/watch?v=-mL3zT1iIKw



AWS re:Invent

October 6-9, 2015 | Las Vegas

A Day in the Life of a Netflix Engineer Using 37% of the Internet

Speaker: Dave Hahn, Senior Engineer, Performance and Reliability Engineering, Netflix

NETFLIX

- Chaos Monkey
- Chaos Gorilla
- Chaos Kong
- Armageddon Monkey (disable AMAZON!)
- Janitor Monkey
- Conformity Monkey

http://techblog.netflix.com/2011/07/netflix-simian-army.html

Chaos Monkey

- Identify group of systems, and **randomly terminate one server**
- Runs during off-peak, normal business hours while engineers are working
- Idea: Unleash a wild monkey with a weapon in your data center to randomly shoot down instances and chew through cables -- all the while continuing to serve customers without interruption
- **Why would you do this?**

Chaos Monkey Motivation

“Failures happen, and they inevitably happen when least desired.”

“If your application can't tolerate a system failure would you rather find out by being paged at 3am or after you are in the office having already had your morning coffee?”

“Even if you are confident that your architecture can tolerate a system failure, are you sure it will still be able to next week or next month? Software is complex and dynamic, that "simple fix" you put in place last week could have undesired consequences.”

https://github.com/Netflix/SimianArmy/wiki/Chaos-Monkey

Chaos Monkey Motivation

- With *Chaos Monkey*, the **engineers know** which servers/services were killed (via logging), so they can diagnose & correct any unanticipated failures

Chaos Gorilla

- Simulates outage of entire **AWS availability zone**

Chaos Kong

Simulates outage of entire **AWS region**

Real life failure:

- DynamoDB (and dependent services) failed in US-EAST-1 region on Sept 20th 2015 for 6-8 hours

Simulating this in advance allowed Netflix to **survive failure**

http://techblog.netflix.com/2015/09/chaos-engineering-upgraded.html

Latency Monkey

- Introduce *artificial delays* in RESTful client-server communication layer
- **Small delays** simulate service *degradation*
  - Measure if upstream services respond appropriately
- **Large delays** simulate failure of node or service (without physically bringing instances down)
  - Test fault-tolerance of a service by simulating the failure of its dependencies, *without* making these dependencies unavailable to the rest of the system

Open Source Monkeys

- https://github.com/Netflix/SimianArmy/wiki
- **Chaos Monkey**
- **Janitor Monkey**
  - *Locate unused resources and terminate them*
- **Conformity Monkey**
  - *Locate nodes not following best practices and terminate them*
  - *Example: Nodes not in auto-scaling group*