AWS re:Invent

CONSUMER DIGITAL TECH

ARC 308

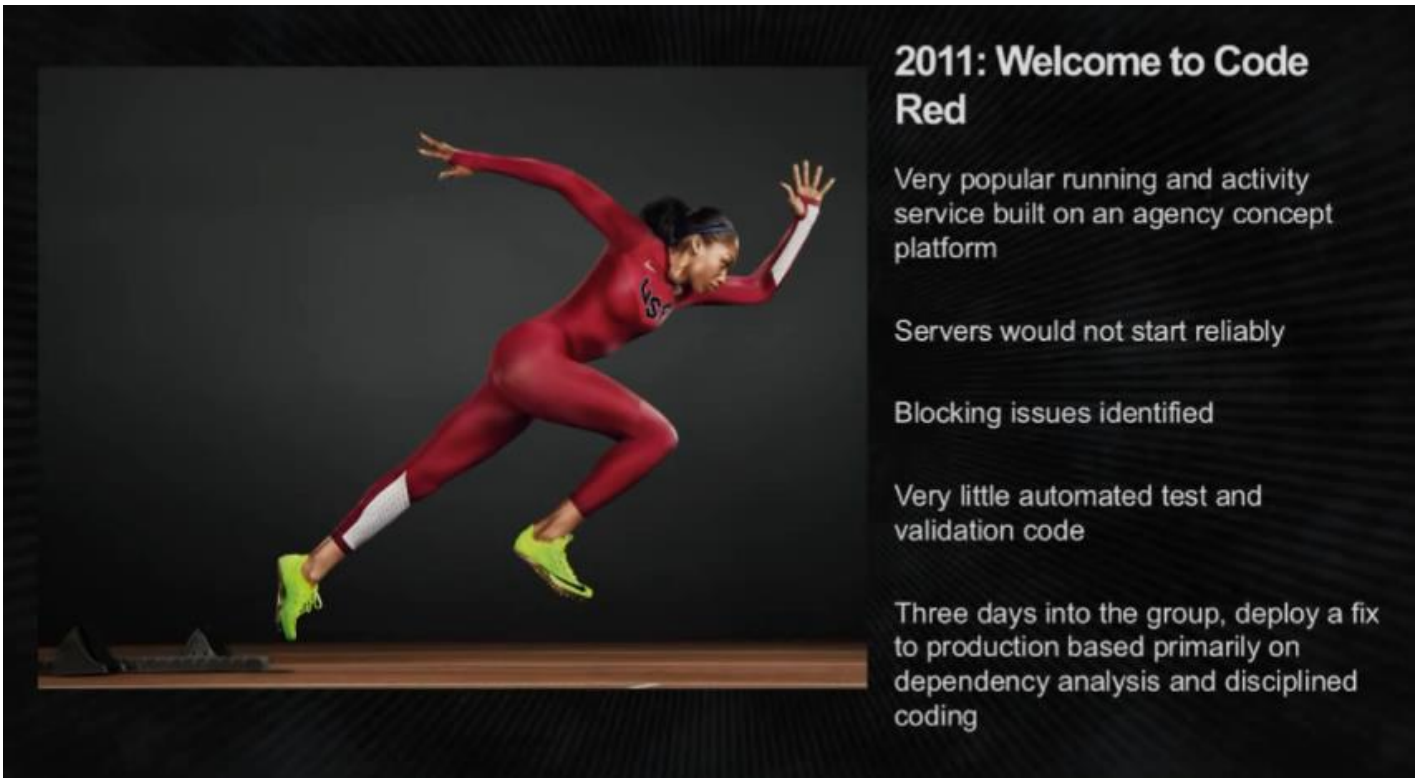
Nike's Journey to Microservices

Jason Robey, Consumer Digital Technology (CDT), Nike, Inc.

November 12, 2014 | Las Vegas, NV

amazon web services

Tightly coupled monolithic stacks can present challenges for companies looking to take full advantage of the cloud. In order to move to a 100 percent cloud-native architecture, the Nike team realized they would need to rewrite all of the Nike Digital sites (Commerce, Sport, and Brand) as microservices. This presentation will discuss this journey and the architecture decisions behind making this happen. Nike presenters will talk about adopting the Netflix operations support systems (OSS) stack for their deployment pipeline and application architecture, covering the problems this solved and the challenges this introduced.



2011: The early days

- Validation: Haphazard and manual based on specific features in a shared environment
- In-house product understanding: very low
- Deployment quality: poor, nearly every release contained escaped defects
- Engineer work cycle: near 24/7
- Time to deploy a new feature: months

2012: We have our own back end

Deep vertical tech stack

Universal mentality to supporting multiple apps

Expensive technology

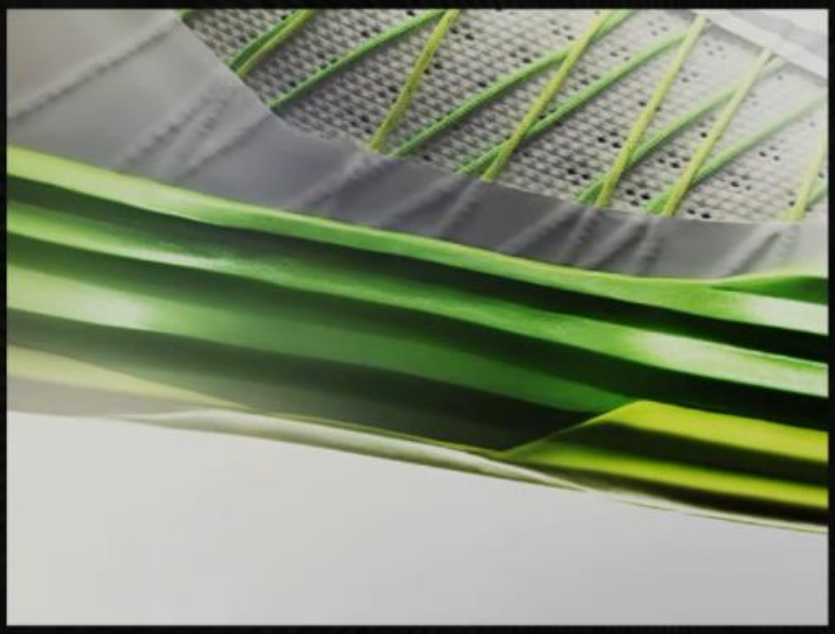
Well-known public standards

Nine development teams on the same code base for 14 products

2 days developing to 15 days "stabilizing"

2012: Start-up development

- Validation: JUnit/JMeter with coverage requirements but inconsistent results
- In-house product understanding: high
- Deployment quality: okay; intense manual validation required with many date misses
- Engineer work cycle: near 24/7
- Time to deploy a new feature: months to quarters
- Product perspective: fast and chaotic



2013: Get to green

400,000 lines of code, 1.5 million lines of test code

Pairwise development and TDD

Pull requests enable 100% peer reviewed code

Development cycle gets 5–10 days to develop for every 15

Developed strict deployment process and change control

Reduced to 3 development teams

2013: Tech debt repayment development

- Validation: JUnit/JMeter automated and mostly consistent
- In-house product understanding: very high
- Deployment quality: stable with intended feature sets and bug fixes
- Engineer work cycle: sane
- Time to deploy a new feature: weeks to months
- Product perspective: slow and stable

Key observations

- Configuration management still delays our software cycle
- Database deployments still too manual
- Even small changes have unintended consequences (now caught in testing)
- Still moving too slowly
- Normalization is the root of all evil
- Huge organizational approvals for everything

Enter microservices…

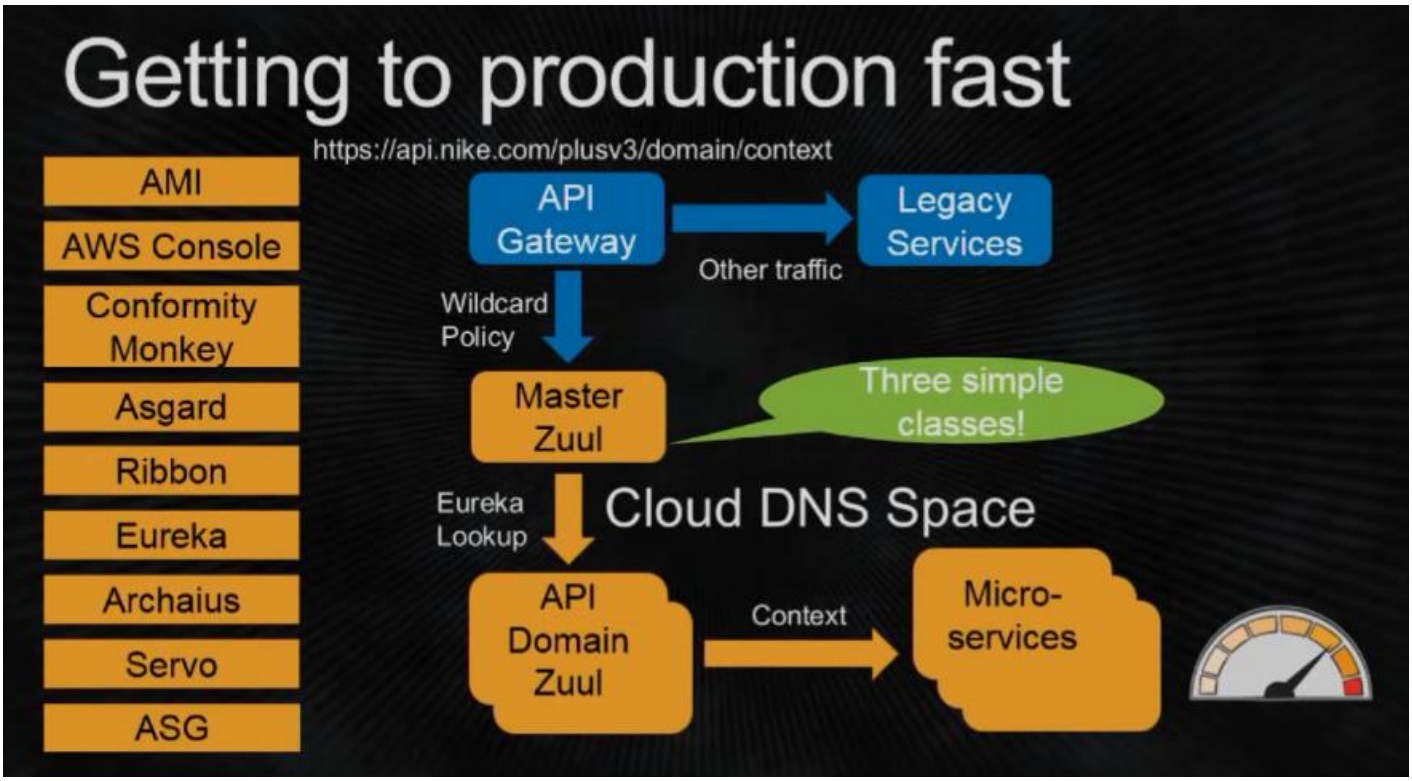
- Phoenix pattern: immutable deployment units
- Share nothing: Cassandra
- Consumer experience flows over end-to-end implementations
- Continuous delivery: reduce the risk profile of any deploy/trust your engineers

Life is not perfect

- Integrate with enough of the legacy systems
- Work within legacy process yet enable continuous delivery
- Notification pattern when still required
- Break all that is shared

- Now for some patterns

Getting to production fast



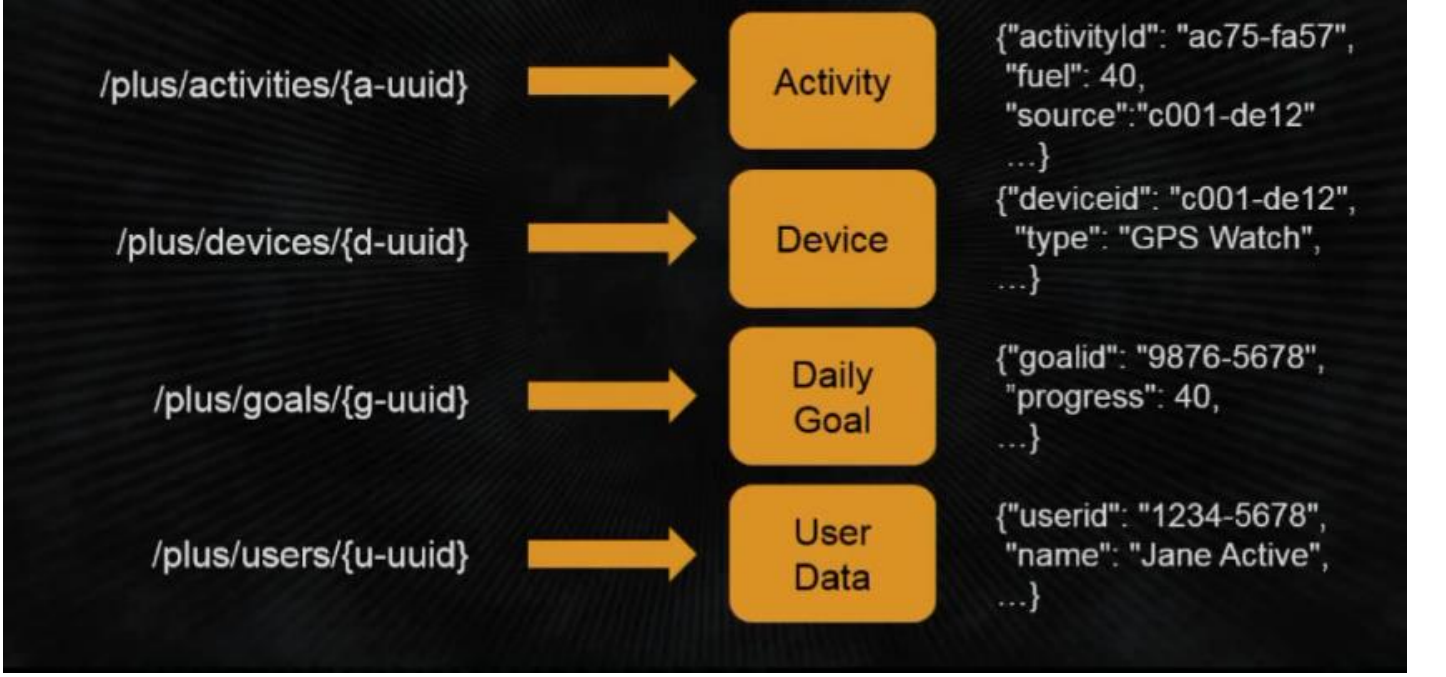
Notification pattern

Legacy service

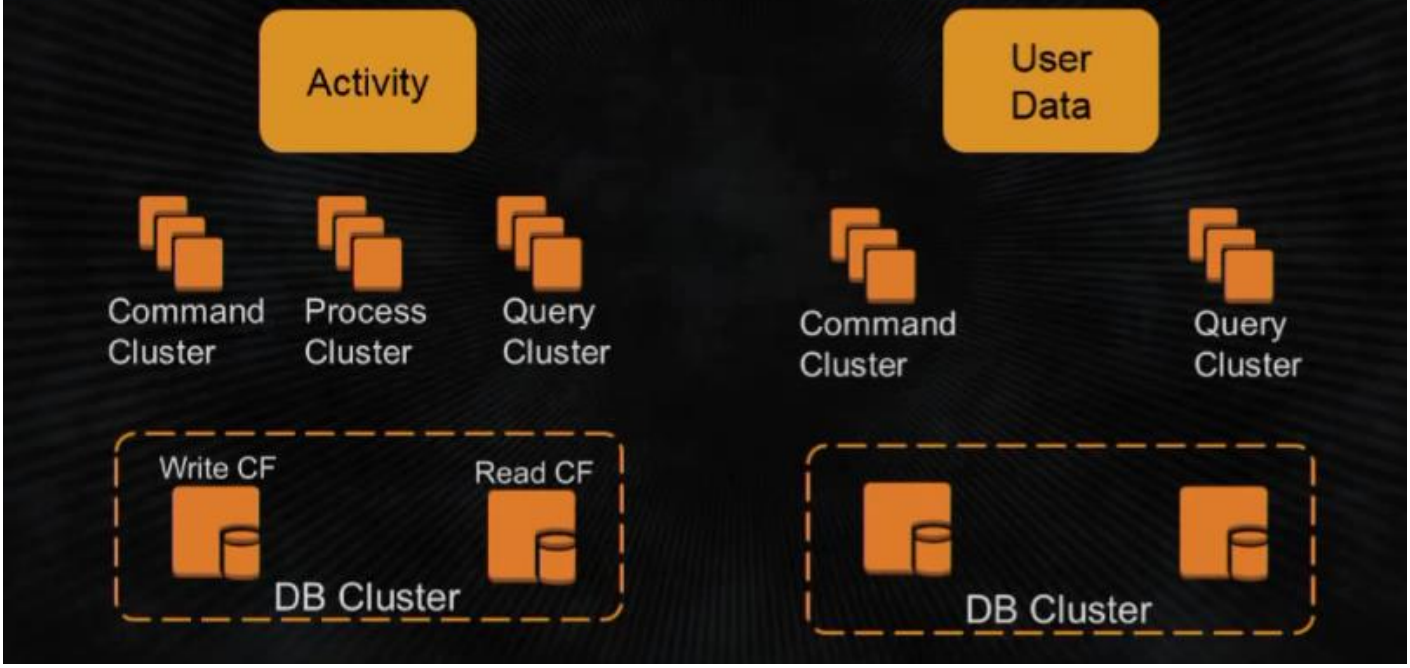
/plus/v1.0/me/activities/id/{uuid} → Produces

```
{"activity": {
      "activityId": "ac75-fa57",
      "fuel": 40,
      …
      "dailyGoal": {
         "progress": 40,
         "targetValue": 8000
      }
   },
   "userData": {
      "name": "Jane Active",
      "height": 68,
      …
   },
   "device":{
      "type":"GPS Watch",…
}
```

Break all that is shared

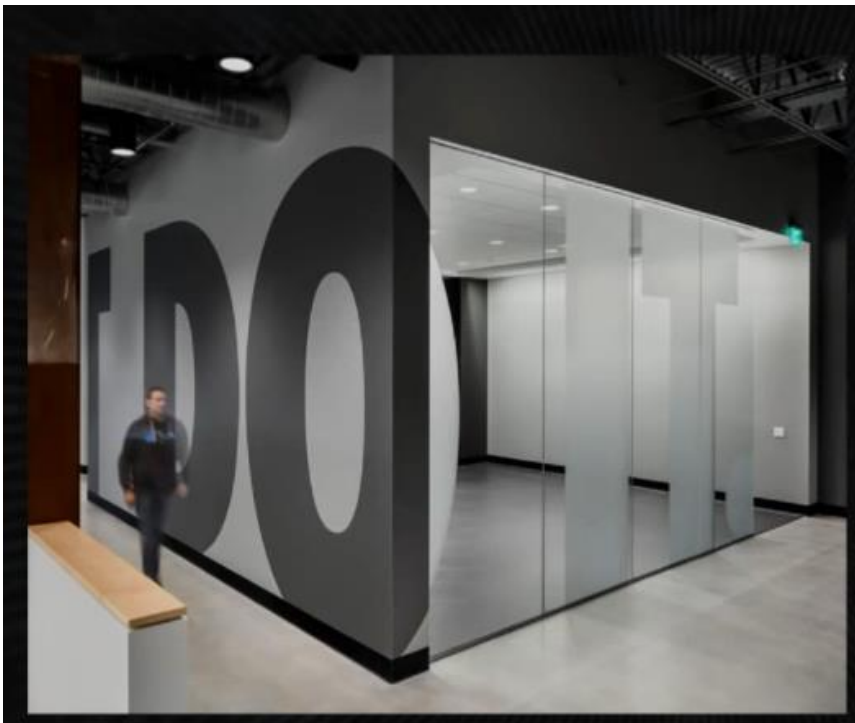


Share nothing design



Client benefit

- Clients now ask for the information they need
  - Activity data, not user and goal
- Can still create custom, no-logic composites
- Clients able to effectively cache or just use the really fast service—experience is now composable
- Scaling concerns to not cross functional boundaries
- Simplified domains do not require one-off business logic



2014: Continuous Delivery

15 projects

700–2000 lines of code in a project

Developers plus quality engineer take changes from sprint prioritization to production multiple times a day

Able to work on and complete the priority of the day

Open source feel—pick up a project and improve it

0 downtime, high scale

2014: Continuous Delivery

- Validation: 100% automated per project
- In-house product understanding: Very High
- Deployment Quality: 0-downtime, well understood
- Engineer work cycle: sane and rewarding
- Time to deploy a new feature: hours
- Product perspective: Free your mind!

We've only just begun…

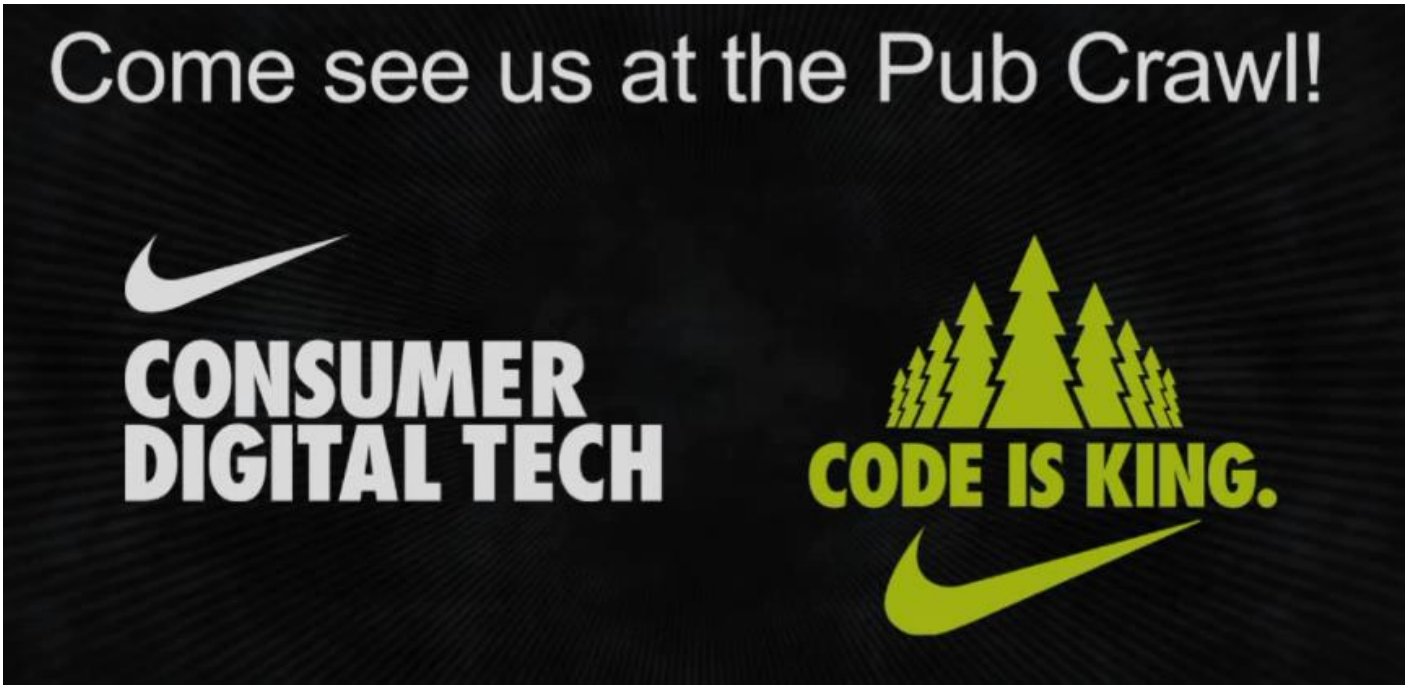
- Share nothing has enabled us to move forward quickly
- AWS enabled our exploration and production hardening
- Netflix OSS really provided us a jumpstart
- We are on the journey to rebuild our services to be horizontally scaling micro-service based, with strong API contracts

Want to help?

CONSUMER DIGITAL TECH

CODE IS KING.

- Technology is a vital enabler of Nike's goal to bring inspiration to all athletes. Our consumers expect us to be digitally connected as they are, making digital a top priority for the company.
- We employ top industry talent working in a dynamic, cutting-edge environment, at one of the world's most innovative brands. We are looking for talent across all disciplines: PM, Dev, QM, Operations, Dev Ops



AWS re:Invent

Please give us your feedback on this session.
Complete session evaluations and earn re:Invent swag.

http://bit.ly/awsevals

ARC308

amazon web services

Join the conversation on Twitter with #reinvent