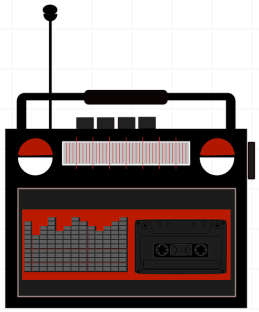
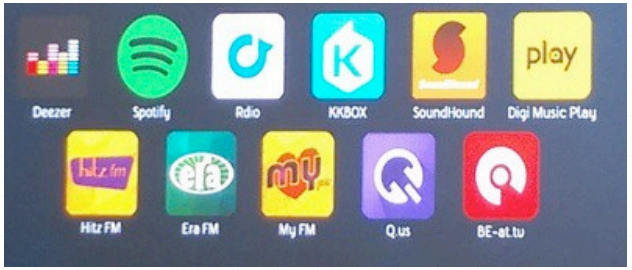
Serverless? are there any Cons?

Ali Kanso, PhD
Senior Cloud SW Engineer
IBM Research
T.j. Waston, NY
USA

The New Paradigm

Netflix: 125 million streaming hours a day



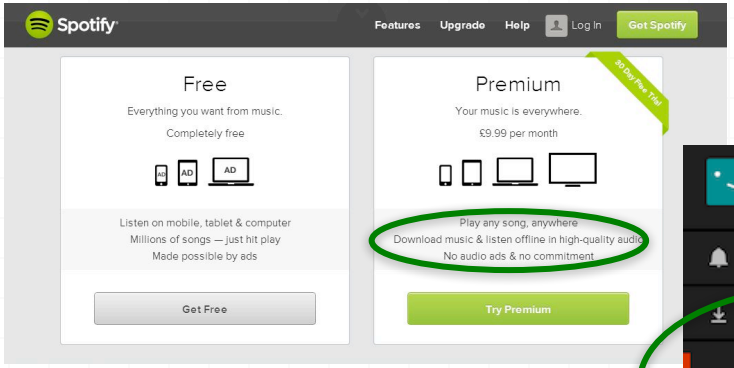


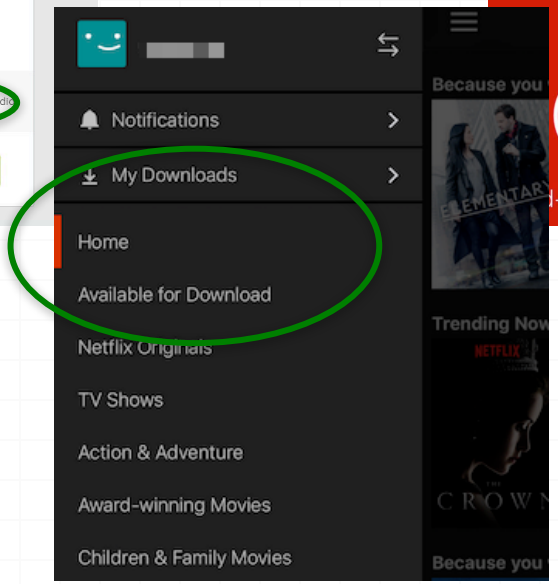
Millennials spend 25 hours a week streaming music

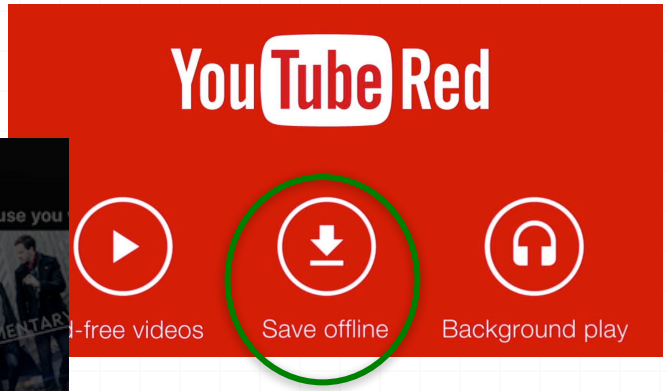
✓From **Passive** Delivery, to **Active** Delivery

Event-based

From Streaming to File Transfer

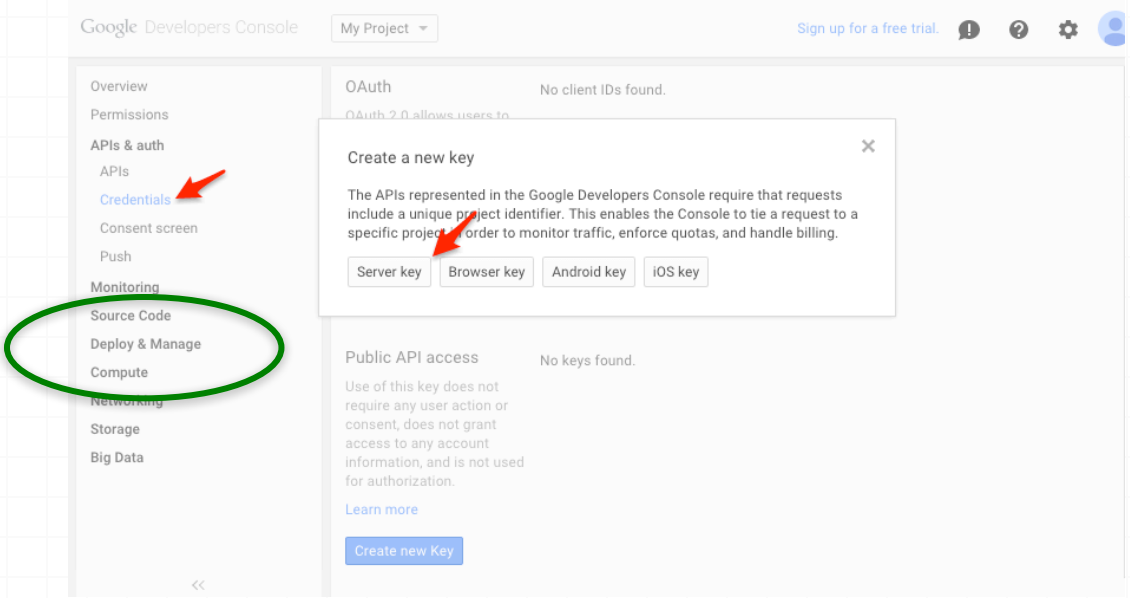






- ✓ Subscription pays more than adds!
- ✓ Streaming is costly!
- ✓ Solution: File Transfer with **Smart-Client**
- ✓ **Transfer from long-lived event, to short lived event!**

Why not simply PaaS?



- Manage min # of instances
  - With microservice this can grow
- Manage the workflow
- **Scaling is reactive**, not deterministic

Pricing is a trap!

A function created with 1 GB of memory will cost $0.000001667 per 100ms of execution time

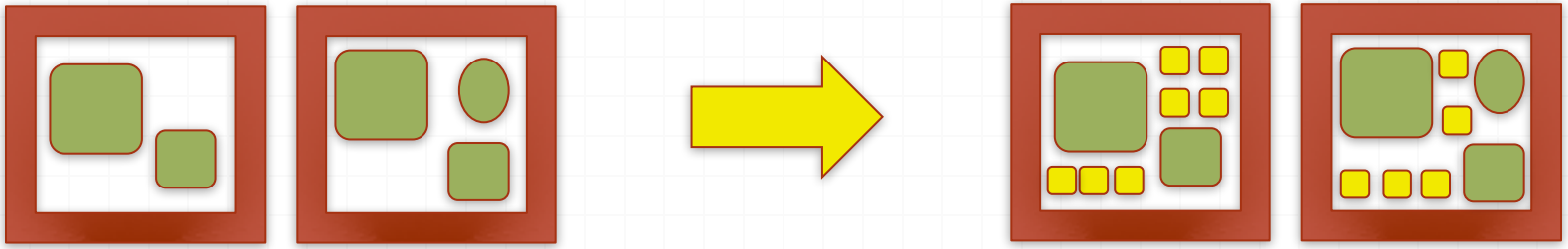
Amazon provides a perpetual free tier with 1 million free requests and **400,000 GB-seconds** of compute time per month

5M requests, 100GB data moving, 10M GB-seconds = $62.307 per month

ACTION EXECUTION TIME	ACTION MEMORY	EXECUTIONS PER MONTH	COST PER MONTH
500ms	128MB	5,000,000	Free
500ms	256MB	5,000,000	$3.83

Making Sense Of Pricing

- Cloud Providers know that:
  - serverless cannot work independently
  - You will need a backends that the cloud provider is offering (less latency), otherwise you pay for egress
  - As well as other services, and notifications …
- The model allows Cloud Providers to fill unused resources in small portions



Any Cons?

- When **Public:**
  - It may not be suitable for privacy-sensitive apps
    - Government, healthcare, Banking, …
  - It may not be suitable for latency-sensitive applications
  - You cannot customize your infrastructure to better suit your needs (e.g. lighter OS)
  - Vendor lock-in with the APIs and ther services eco-system
- When **Private:**
  - You have to manage and maintain the serverless framework
    - Typically it is a multicomponent system (nginx, kafka, CouchDB, …)
      - Deployment, Availability, scaling, …

Any Cons?

- You need to consider building thicker smarter front ends
  - E.g. make Third Party Calls and embed them in the request action
- Smarter front-ends can help reducing the number of invocations, and making sense of the results

Remaining Question

Is it still worth it??