Cloud overview

Portland State
Computer Science

"The illiterate of the 21st century will not be those who cannot read and write, but those who cannot learn, unlearn, and relearn."

- Alvin Toffler, "Future Shock" (1970)

From a recruiter (for a company with a 20% policy)

"We're getting better and better at finding the folks with the right attitude and the right history of learning… the world of technology changes so rapidly, that any amount of real experience and depth in a given area has value for about a *decade*."

Motivation

- Majority of jobs in CS in the next decade will involve the cloud
  - Companies moving to it
  - Software and services being migrated onto it
  - New services being built on top of it
- Jobs, skills, and software platforms transforming
  - Network administration jobs obsoleted by software-defined networking?
  - Virtualization software obsoleted by infrastructure-as-a-service?
  - Database administration jobs obsoleted by database-as-a-service?
  - IT and Ops jobs obsoleted by platform-as-a-service (NoOps)?
  - Cloud administration tools obsoleted by automated orchestration?
- In general, learn what's already been done to focus on what adds value

Abstraction and Composition

- At the heart of everything humans do
- Abstract an idea, then use it to compose
- e.g. Music
  - Use the abstractions of keys, notes, measures, and chords to form a symphony
- Also
  - Mathematics
  - Language
  - Physics
  - Architecture
  - Poetry
  - Fashion Design
  - Painting

Goal of course

- Abstraction and composition applied to computing systems
- Other courses cover..
  - What is inside the widget
    - Internet protocols
    - Operating systems
    - Database backends
    - Machine learning algorithms
  - How to make the widget
    - Functional programming
    - Object-oriented programming
    - Software engineering and testing
- This course
  - Abstractions that widgets provide
  - How to compose them to build complex systems
    - e.g. Scalable, fault-tolerant, globally distributed, cloud-based web application

Changing abstractions: communications

- Circa 1980s: program packets directly (IP)
- Circa 1990s: program via sockets as abstractions over TCP and UDP packets (BSD)
- Circa 2000s: program via URLs as abstraction over HTTP/TLS over sockets and packets

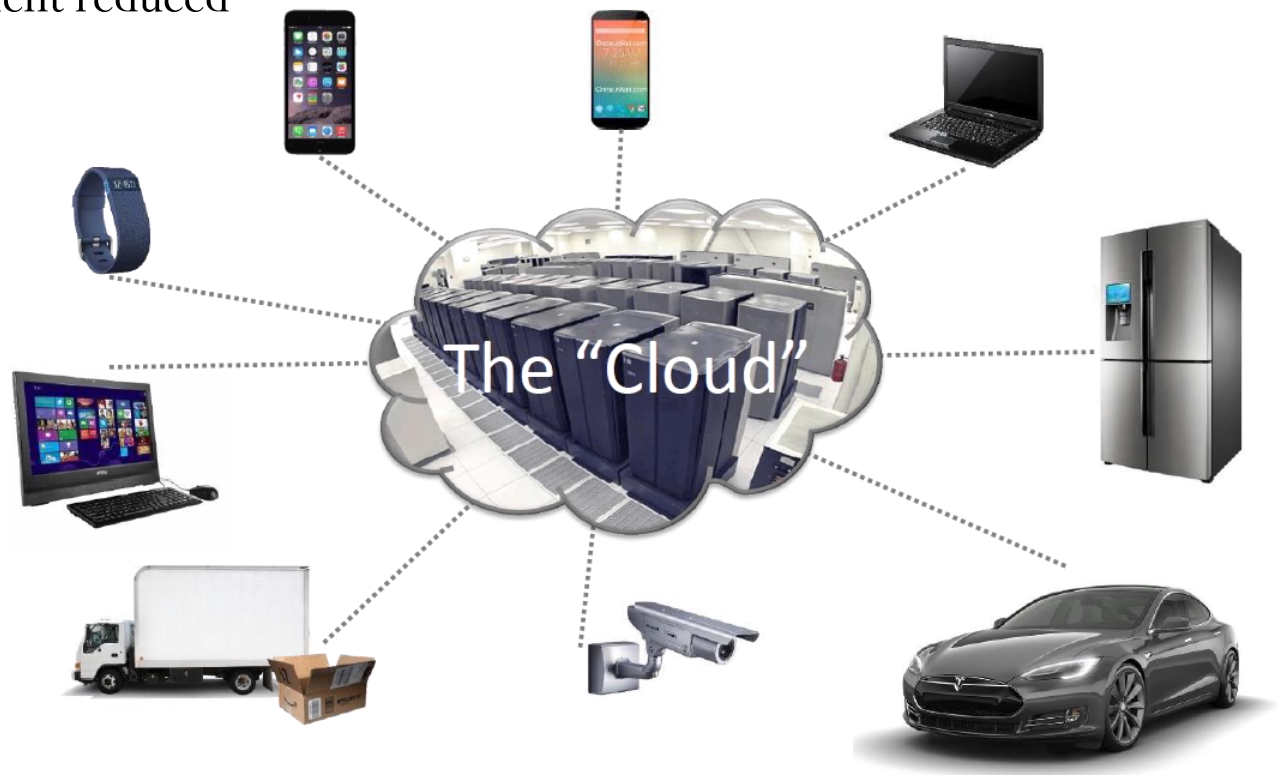
Changing abstractions: Software engineering

- Circa 1980s: Single program and machine/OS (C/Asm)
- Circa 1990s: Client-server apps, (Browsers/Servers)
- Now: Collections of machines and networks performing complex tasks (Distributed clusters, cloud, multi-cloud applications)

Changing abstractions: Systems deployment

- 1990s
  - Purchase your own building, hardware, software, network links/capacity
  - Hire staff to manage all of it
  - Pay for everything, even if not being used
- 2000s
  - Shared data-center
    - Co-locate with others to get economies of scale at the network and physical space level
    - Still requires companies to purchase servers, install software
    - Still requires companies to have their own IT/Operations staff

- Now: Cloud computing
  - Computing resources as a service (like electricity supplied by utility)
  - Economies of scale across machines, data-centers, networks, and their management
    - Statistical multiplexing between companies and users to reduce cost
      - Recall packet-switching vs. circuit-switching, but apply to all computing resources
    - Management reduced



Cloud computing advantages

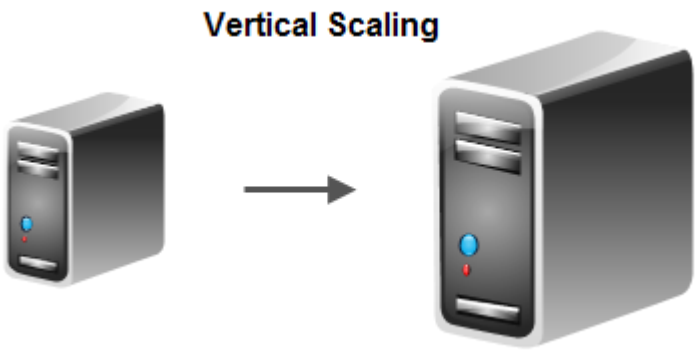
- Can go "assetless"
  - No hardware to purchase, no physical space to purchase
- Ability to pay only for what you use
  - Scale up and down based on demand
  - At per-second granularity with minimal up-front investment
- Can have minimal IT/Ops staff (comparatively)
  - Simplified, automated IT deployment and maintenance handled by provider
  - Exponential growth in machines and network without an exponential growth in employees (via increasing automation to manage)
- Access to value-added services
  - ML APIs (Vision, Video Intelligence, Natural Language Processing, Speech)
  - GIS APIs
  - Data warehousing
- Raises the levels of abstraction in systems development and deployment

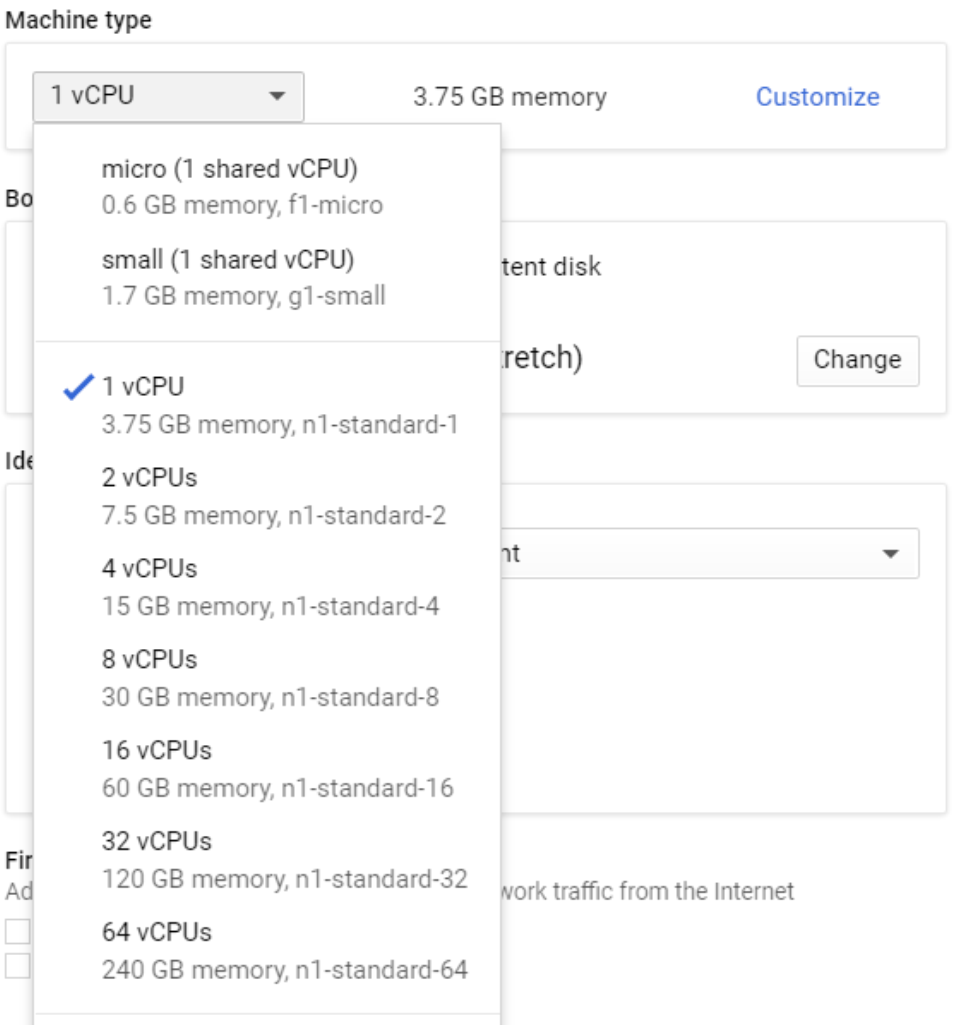
How?

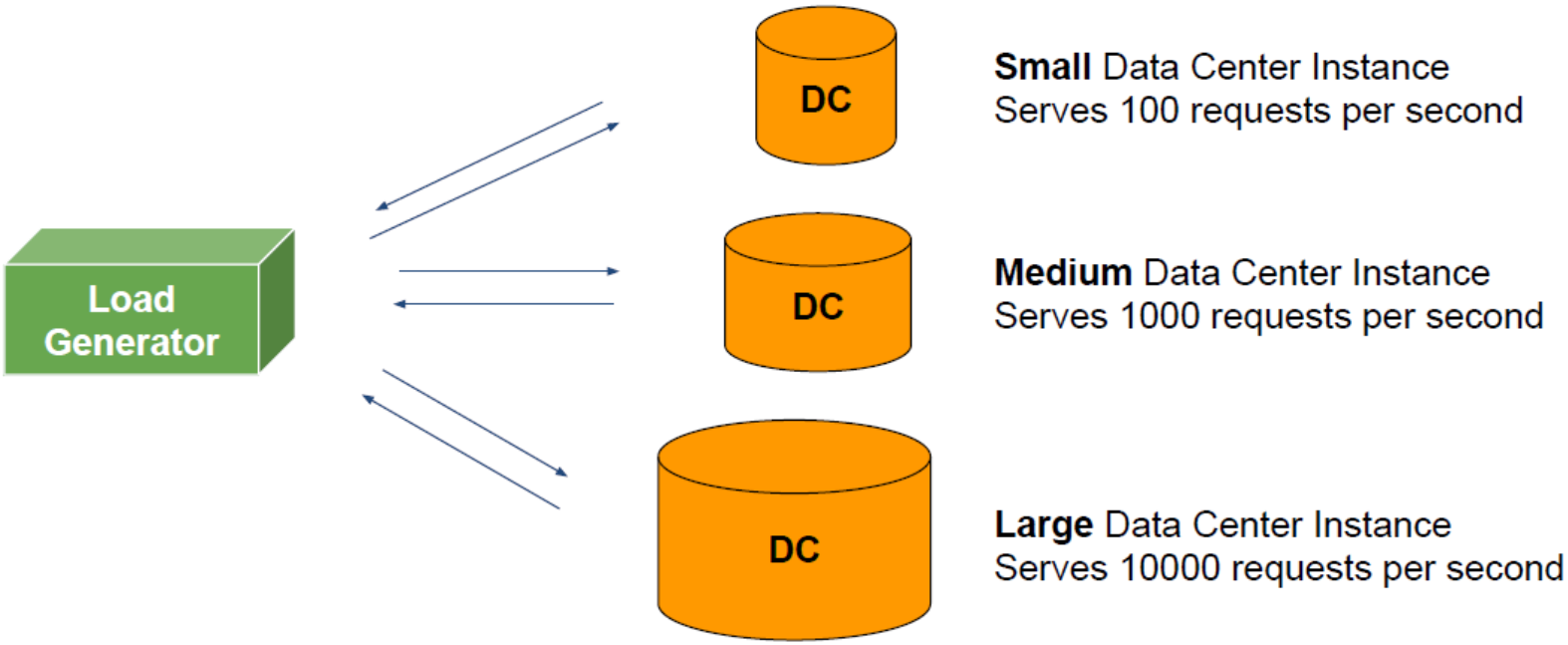
- Enabling technology
  - Full-featured web browsers
  - Fast and inexpensive servers
  - High speed Internet access
  - Large-scale distributed storage and file systems
  - Virtualization software, hardware, and networks
  - Open standards and software
    - TCP/IP, OpenFlow, Linux, Docker, nginx, apache2, python, MySQL, Hadoop, Kubernetes etc.

Scaling approaches in the cloud

- Vertical vs. Horizontal
- Vertical scaling
  - "Scaling Up"
  - Upgrade machine type



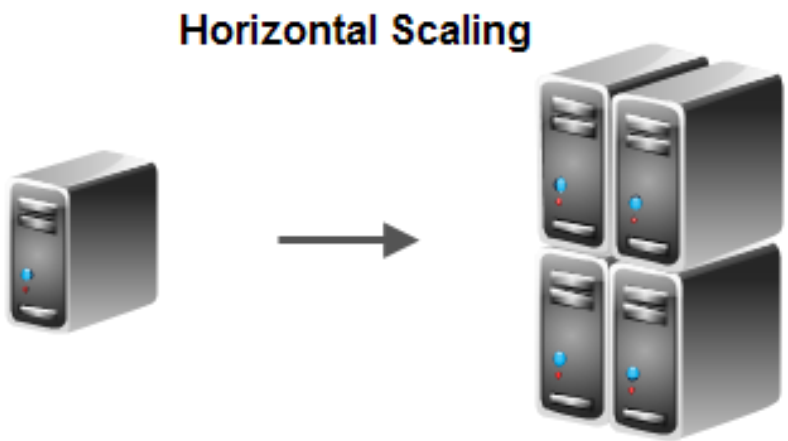


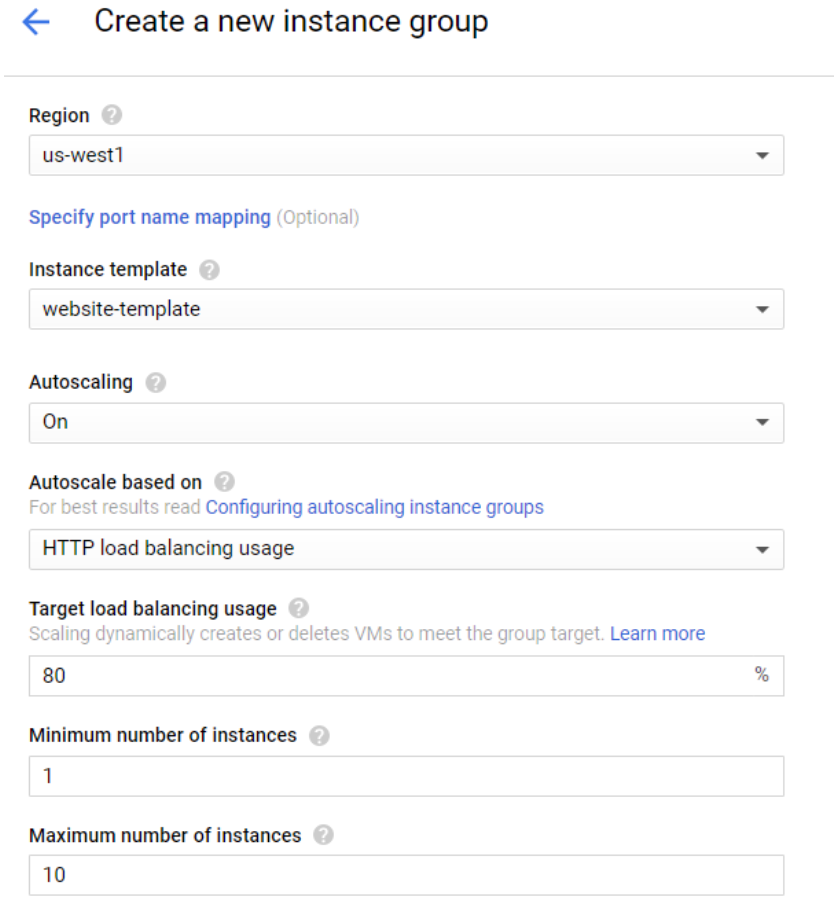


- But, beyond 10000 req/sec?
- What about downtime during failures?

Scaling approaches in the cloud

- Horizontal scaling
  - "Scale out"
  - Replicate and load balance
    - As seen in CDN lab…
    - Automatically add more servers to meet demand
  - Load balancer distributes based on policy
    - Round-robin, server load, least connections, URL



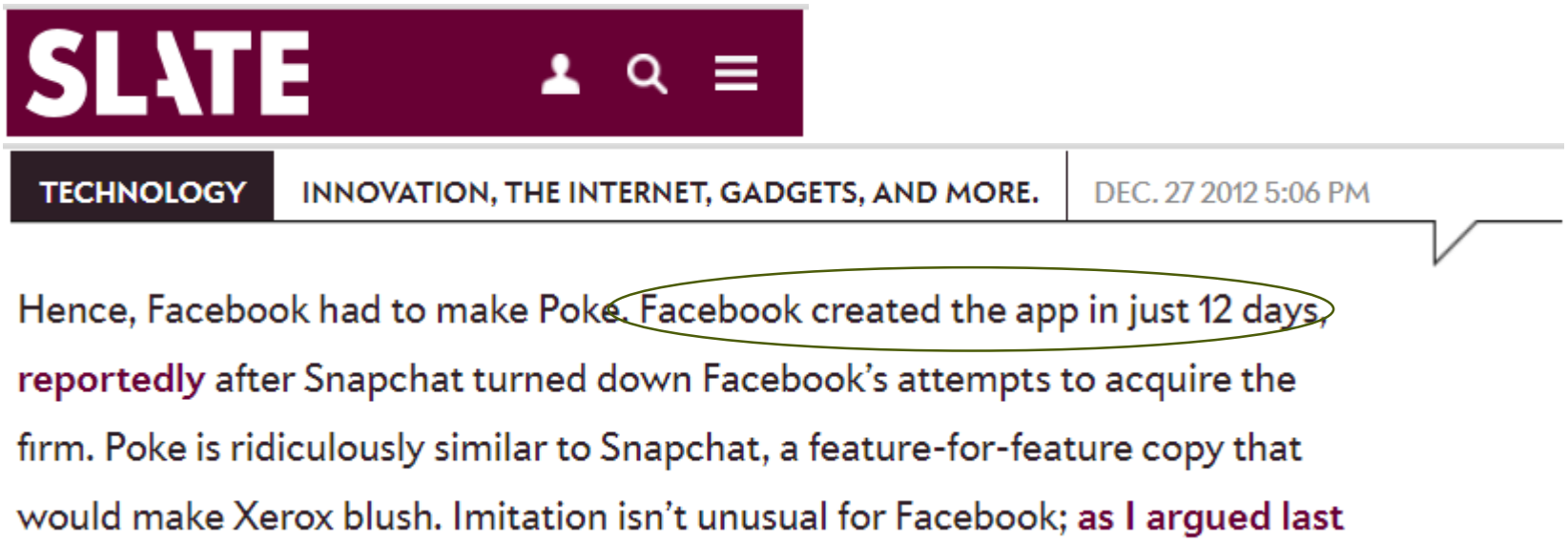


Migration models

- Private Clouds
  - Recreate cloud software on-premises
  - Networks, infrastructures, data centers owned by the organization
- Public Clouds
  - Hosted, operated and managed by third party vendor (this class)
  - Security and day to day management by the vendor
- Hybrid Clouds
  - Sensitive applications in a private cloud and non sensitive applications in a public cloud
  - Why?
    - Regulatory issues
    - Bandwidth issues (e.g. lack of nearby GCP)
    - Location of data (China)
    - Sunk costs in data centers

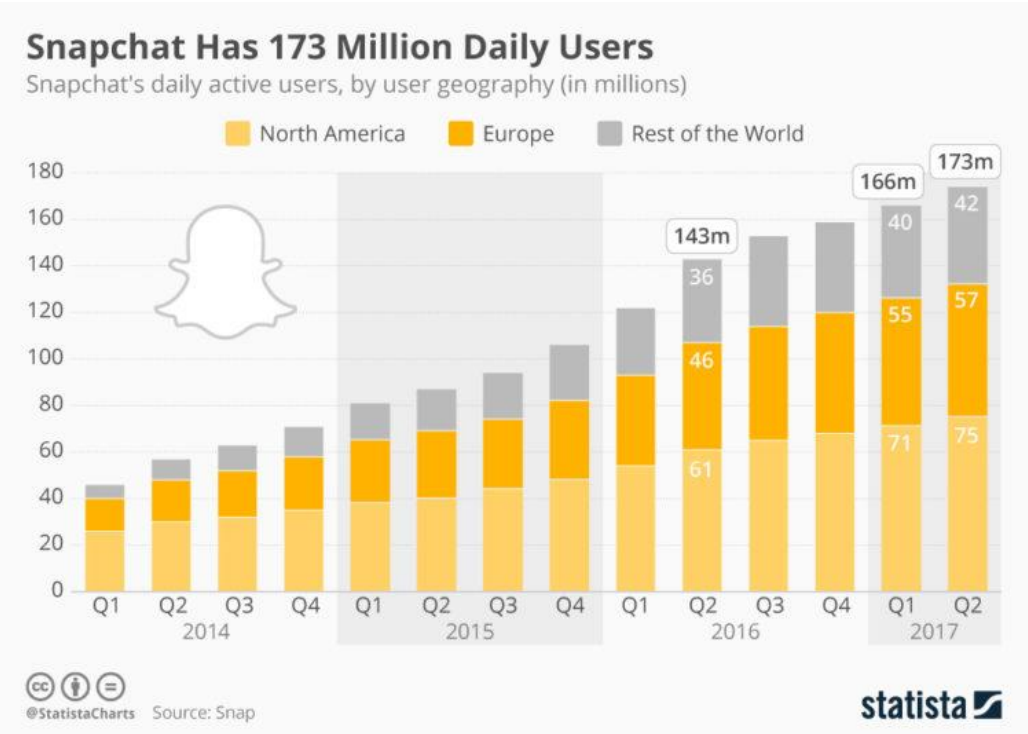
Case study: Snapchat (2011)

- Small startup with no data centers, no operations team
- Two developers with a very simple app
- So simple that…



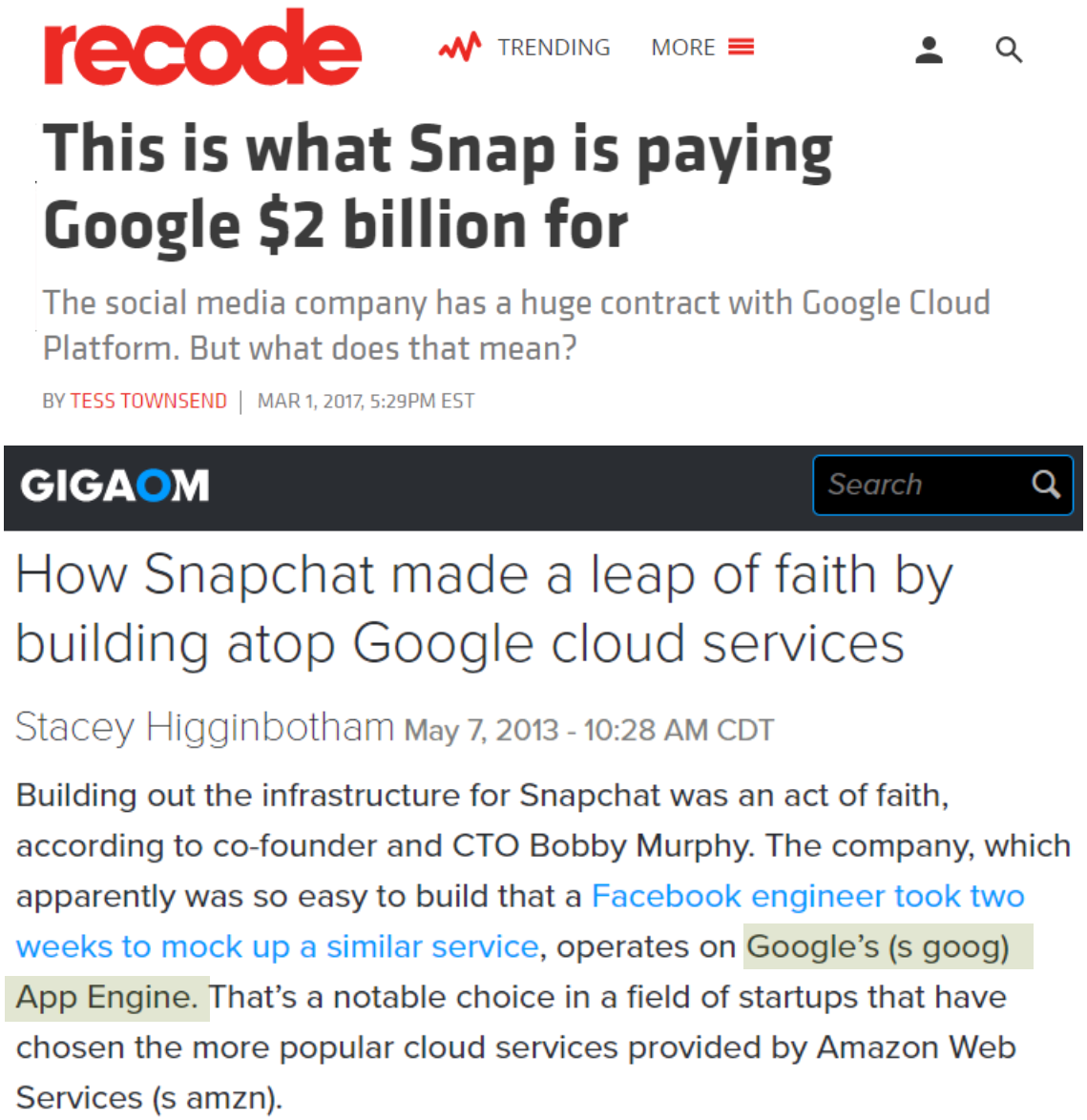
Case study: Snapchat

- By 2013, 10s of millions of users, hundreds of millions of messages
- 2017
  - > 170 million daily users
  - 3 billion photos and videos per day
  - How?



http://www.businessofapps.com/data/snapchat-statistics/ (2017)

Case study: Snapchat



recode TRENDING MORE

This is what Snap is paying Google $2 billion for

The social media company has a huge contract with Google Cloud Platform. But what does that mean?

BY TESS TOWNSEND | MAR 1, 2017, 5:29PM EST

GIGAOM Search

How Snapchat made a leap of faith by building atop Google cloud services

Stacey Higginbotham May 7, 2013 - 10:28 AM CDT

Building out the infrastructure for Snapchat was an act of faith, according to co-founder and CTO Bobby Murphy. The company, which apparently was so easy to build that a Facebook engineer took two weeks to mock up a similar service, operates on Google's (s goog) App Engine. That's a notable choice in a field of startups that have chosen the more popular cloud services provided by Amazon Web Services (s amzn).

Case study: Snapchat

- App Engine (Platform-as-a-Service)
  - Allows startup company focus on core competency (the app itself), not on the data center or infrastructure
  - Allows fast deployment of new versions
  - Allows app to leverage reliability of Google infrastructure
  - Allows fast scale up to massive client base

- Key: ability to horizontally scale application to accommodate additional users

Google Cloud Platform Intro

Portland State
Computer Science

Most labs on GCP

- Student-friendly
  - Credits without credit-cards
  - Ability to use pdx.edu accounts for credits
  - Per-second billing
- Supports open-source APIs and tools to avoid vendor lock-in
  - Go
  - Kubernetes
  - TensorFlow*
- Carbon-neutral since 2007

- Abstractions the same across cloud providers
  - AWS labs

Generous free-tier

- App Engine
  - 28 instance-hours per day
- Cloud Datastore
  - 1GB storage, 50k reads, 20k writes, 20k deletes
- VisionAPI
  - 1k units/month
  - Unit == feature (e.g. facial detection)
- BigQuery
  - Arbitrary loading, copying, exporting
  - First TB of processed data in queries free
  - But, $0.02 per GB per month storage

GCP projects

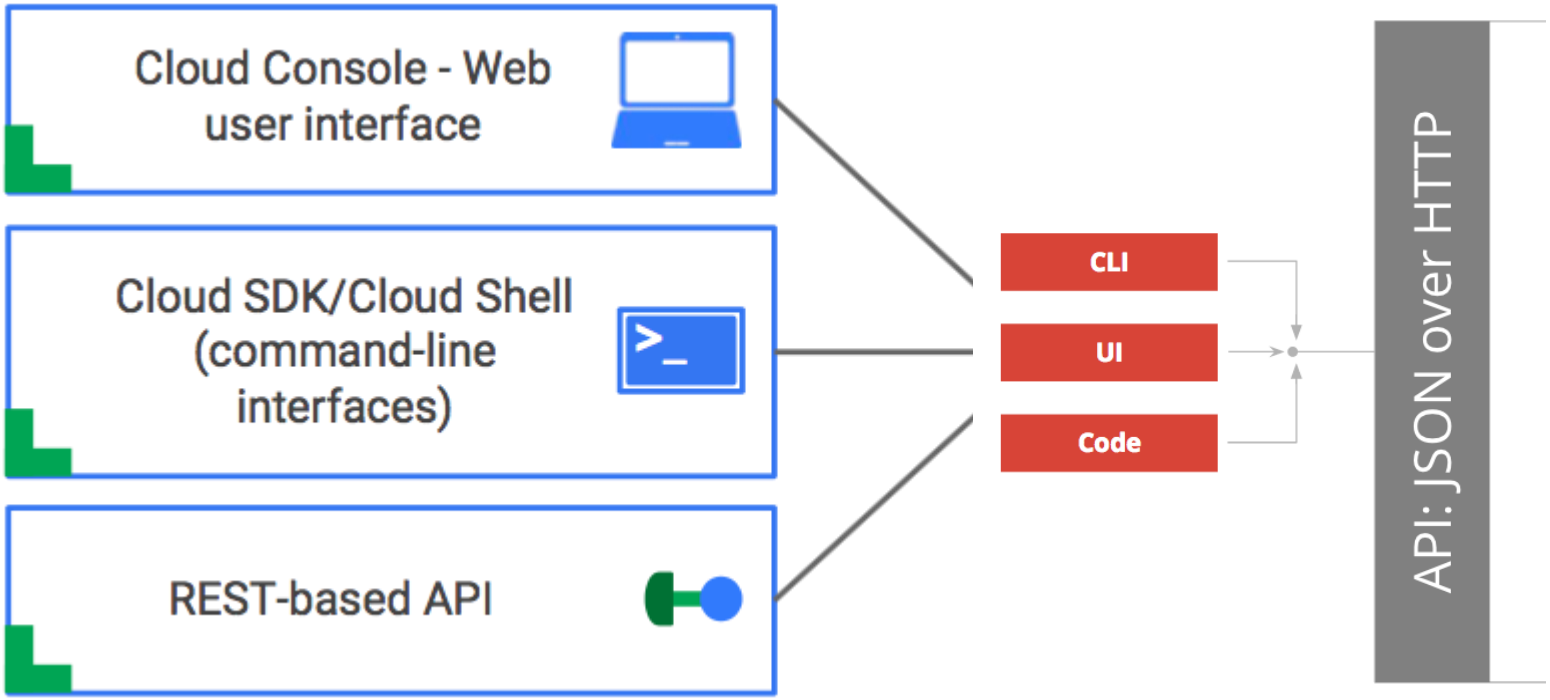
- Many companies with multiple sites
- Each site needs its own
  - Security/access control policies, permissions, and credentials
  - Billing account with separate credit-card/bank accounts
  - Resource and quota tracking
  - Set of enabled services and APIs (most are default OFF and turn on once first used)
- Project abstraction encapsulates this collection
  - Google has 100,000+ projects on GCP to run its sites
  - Contains all resources associated with site and the ability to set permissions on them

Regions and zones in GCP

- Regions: geographic areas where data centers reside
  - us-west, us-east, us-central
  - Consist of collections of zones
- Zones: isolated location within region
  - https://cloud.google.com/compute/docs/regions-zones/

us-west1	Oregon	The Dalles, Oregon, USA	**us-west1-a**	• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform (default) • 2.0 GHz Intel Xeon (Skylake) platform • Up to 96 vCPU machine types when using the Skylake platform • Local SSDs
			us-west1-b	• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform (default) • 2.0 GHz Intel Xeon (Skylake) platform • Up to 96 vCPU machine types when using the Skylake platform • Local SSDs • GPUs
			us-west1-c	• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform (default) • Up to 64 vCPU machine types • Local SSDs

Access to resources



- Also programmatic access in many languages (JavaScript, Python, Go, Java, Ruby)

Command-line GCP

- Install SDK on your local VM (`google-cloud-sdk`) to get commands
  - https://cloud.google.com/sdk/docs/quickstart-debian-ubuntu
  - `gcloud`
  - `gsutil` (Cloud Storage)
  - `bq` (Big Query)
- Docker image

```
docker pull google/cloud-sdk
```

Command-line GCP

- Google Cloud Shell
  - Command-line access via web browser
  - Containerized version of Linux with the latest gcloud SDK running on a ComputeEngine instance
  - Has `nano`, `vim`, `emacs`, `python3`, `virtualenv`, etc.