# *Instagram: Unlocking the Secrets*

Jaideep Srivastava, Dhruv M. Dhokalia, Van Quach, Shyam S. Somasundaram, Song Liao
*Department of Computer Science & Engineering*
*College of Science & Engineering, University of Minnesota*
*Minneapolis, USA*
*srivasta@cs.umn.edu, dhoka002@umn.edu, somas009@umn.edu, quac0058@umn.edu, liaox125@umn.edu*

***Abstract*-Instagaram is a photo and video sharing social media platform, which also allows sharing of media onto other social platforms. It uses an intelligent, yet simple and reliable technology stack to achieve optimum performance.**

***Keywords* – photo, sharing, social media**

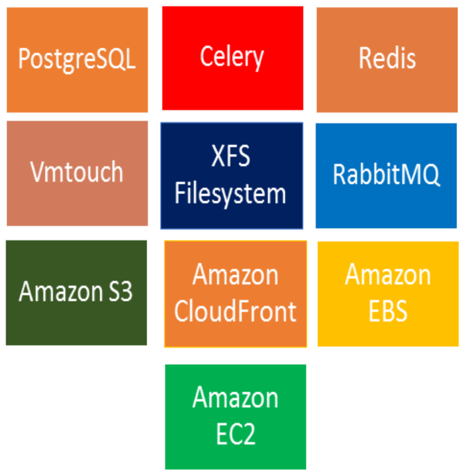
## I. INTRODUCTION

Instagram is a photo and video sharing social media application, which also facilitates posting media onto other platforms such as Facebook, Twitter, Tumblr, and Flickr. 'Kevin Systrom' and 'Mike Krieger' founded the company in October 2010. It has gain substantial amount of popularity since with the rise of mobile users among our contemporary civilization. Facebook bought Instagram for one billion dollars in April 2012 that signifies the importance of the underlying architecture behind Instagram. Hence we aim to derive some of the fundamental tools utilized by Instagram, and observe how it can be beyond superior, efficient and even more dynamic than traditional database management systems.

The main objective of this work is to gain an insight into the internal image storage and overall working architecture of Instagram. This involves an in-depth research of each of the underlying technologies, in order to better understand their architecture, functionalities and advantages. Lastly, based on our understanding of the technologies and other relevant data, we make a best guess as to how the technologies are brought together to work as a single cohesive application that we know as 'Instagram'.

## II. Technology Stack

Fig. 1. Technologies used by Instagram for



handling images.

## III. Amazon Web Services

Amazon web services provides a collection of cloud computing services that allow users to manage and operate assets on the virtual cloud instead of physical servers on a pay-as-you-go basis [1].

### *A. Amazon S3*

Amazon S3 is an online file storage web service. Instagram uses Amazon S3 to store terabytes of photos.

*Architecture*:
Users can create and access unlimited number of objects in 'buckets' (folder with a unique name). The objects are composed of two parts

- 'BLOB' (Binary large object), i.e. multimedia file up to 5GB in size.
- 'Metadata' which includes user-specified key (up to 2KB) that will be stored as key-value pair.

*Search* is limited on a single bucket and is based on object name only. Functionalities such as metadata or content-based search capabilities are not supported [2].

*Pricing*: Amazon provides three versions of storage that differ in performance. Glacier storage is designed to store infrequently accessed data [3].

TABLE 1
Amazon S3 Pricing

| /GB /month | Standard | Reduced Redundancy | Glacier Storage |
|---|---|---|---|
| First 1TB | $0.0300 | $0.024 | $0.01 |
| Next 49TB | $0.0295 | $0.0236 | $0.01 |
| Next 500TB | $0.0285 | $0.0228 | $0.01 |

*Case*: There are 60 million photos uploaded to Instagram daily. An image is cropped and compressed to estimate standard size of 100KB before being uploaded to S3. Therefore with 6000 GB of images uploaded daily, Instagram roughly spends $180 for image storage [4].

*Growth*: Since the launch of S3, Amazon S3 has been growing exponentially. Till this date, there are roughly 2 trillion objects stored in S3 [5].

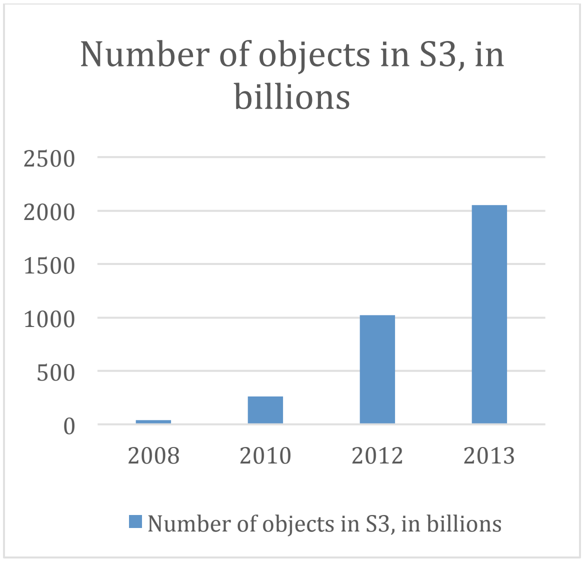


Fig. 2. S3 Growth

Data Access Protocols:
Amazon S3 [16] supports three data access protocols:

- Simple Object access protocol (SOAP)
- Representational State Transfer (REST)
- BitTorrent [2]

*B. Amazon EC2*

Amazon EC2 is a web hosting service that provides resizable computing capacity in the cloud, allowing user to quickly scale capacity as the computing requirement change [6].

*Case:* User can perform tasks in parallel through EC2 virtual servers. For example, media-sharing app *Mobli* uses EC2 to transcode the video format and gathers user feeds from different servers [7].

*Auto Scaling:* It allows user to scale EC2 capacity up or down automatically according to the traffic. When demands of the application increases, the auto-scaling feature will automatically launched new instances in the cloud. On the other hand, terminating instances is easy [6].

*Advantages*: Compare to traditional web hosting service where users have to go through a laborious and expensive process of adding more physical servers vertically and sync the computing together, EC2 provides a seamlessly horizontal scaling network that greatly reduce operation time and cost [16].

### *C. Amazon Cloud-Front*

Amazon Cloud-Front is a content delivery web service. It integrates with other Amazon web services such as S3 and EC2 to give developers and business an easy way to distribute content to end user with low latency, high data transfer speed and no commitments [8].

*Methodology*: Amazon has physical servers called edge locations over 50 cities world-widely. Unlike a traditional delivery service that the user request has to route many times before reaching the destination, Cloud-Front will automatically find the nearest edge location to push the data with low latency.

*Case: How CloudFront delivers an image to a user in Austin, Texas*

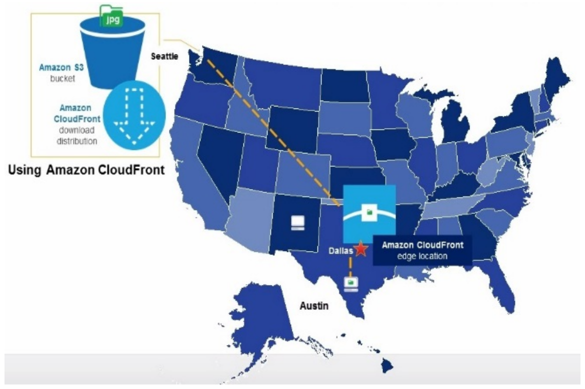

jpg
Amazon S3 bucket
Seattle
Amazon CloudFront download distribution
Using Amazon CloudFront
Dallas
Amazon CloudFront edge location
Austin

Fig. 3. CloudFront Example

1. A user in Austin, Texas accesses the application and request an image file.
2. DNS routes the request to the nearest CloudFront edge location in Dallas, which has the lowest latency.
3. In the edge location, CloudFront checks cache for requested files. If the object is in the cache, CloudFront returns them to user. If not, CloudFront does the following:
   i. Cloud Front forwards to request to original server in Seattle.
   ii. The original server sends back the file to the Cloud-Front location.
   iii. As soon as the first byte arrives, CloudFront begins forward the image to the user and stores the image in the cache for next time use. [8]

### *D. Amazon EBS*

Amazon Elastic Block Store provides persistant block level storage volumes to back up data. It works with S3, EC2, relational database and non-relational database.

Normally on a local machine, the data will be erased after the sever instance is terminated. This could be used to store temp files. Meanwhile in EBS, data will persist independently regardless of status of the server. [9]

Amazon EBS provides two types of storage.

*Storage Types:*

- Standard Volume: designed for applications with moderate I/O requirement.
- Provisioned IOPS Volume: offers storage with consistent and low-

latency performance, up to 10 times faster than standard volume. When attached to EC2 instances, the speed can be boosted up 10 times more. [9]

*Snapshotting:* It saves point-in-time of Amazon S3 volume. It saves only the change of the device, i.e. if a user has a device of 100 GB of data but only changed 5GB; a subsequent snapshot consumes only 5 additional GB.

### *E. How Amazon web services work together*

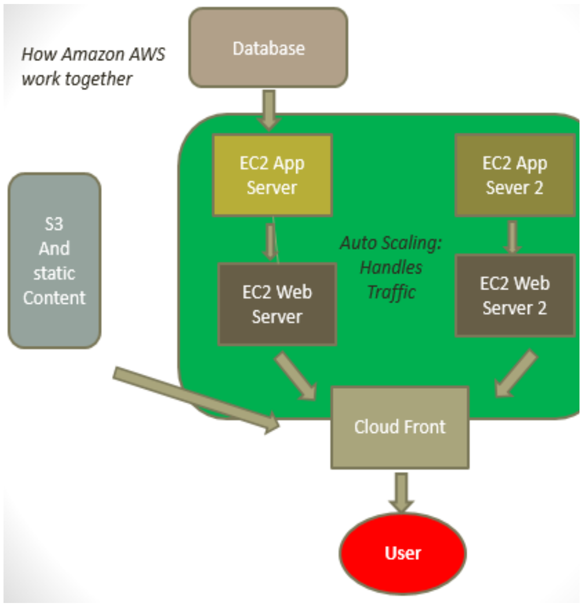


Fig. 4. AWS Architecture

In the above flowchart, there is a cloud of EC2 servers: two EC2 Application servers running the application (business logic) and two EC2 Web servers handle HTTP requests. The auto-scaling feature handles the number of servers in the cloud. S3 stores static contents such as image files. Everything is delivered using CloudFront to the user.

## IV. OTHER TECHNOLOGIES

Instagram uses certain technologies in addition to the amazon web services that play a crucial role in the working of Instagram architecture.

### *A. PostgreSQL*

PostgreSQL [10] is a powerful, open source object-relational database system. It supports foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL; 2008 data types, including ‘Integer’, ‘Numeric’, ‘Boolean’, and etc. It also supports storage of Binary Large Objects (BLOB), including pictures, sounds, or video.

It is an enterprise class database that boasts many sophisticated features such as ‘Point in Time Recovery’, ‘Online/Hot Backups’, and ‘Write Ahead Logging’ for fault tolerance. ‘Point in Time Recovery allows the user to view a database table and its data at a particular date in the past by maintaining a PITR log. Hot Backups are backups that do not require any downtime, i.e the system need not be shutdown while doing a backup. However any changes made to the system during the backup are not reflected in the backup. Hot Backups are very useful in multi user systems. Write Ahead Logging (WAL) is a technique in which all modifications to the system are written to a log before they are applied. If a system crash/ failure occurs the WAL helps to undo or complete any transaction that was started thus preserving the atomicity and durability properties of the database.

It is highly scalable. It can manage increasing amount of data and accommodate increasing number of concurrent users.

TABLE II
LIMITATIONS OF POSTGRESQL

| Limit | Value |
|---|---|
| Maximum Database Size | Unlimited |
| Maximum Table Size | 32TB |
| Maximum Row Size | 1.6TB |
| Maximum Field Size | 1GB |
| Maximum Rows/Table | Unlimited |
| Maximum Columns/Table | 250 -1600 |

Instagram uses PostgreSQL as its back-end database to store the user profile information. It runs on 12 Quadruple Extra-Large memory instances, each running in a different availability zone. To facilitate query results partial indexing is used which filters the query according to a certain condition and thus returns a subset of rows in the table.

*B. Redis*

Redis [11] is an open source, advanced key-value store with optional durability often referred to as a '*Data Structure Server*', since the keys can contain strings, hashes, lists, sets and sorted sets.

Redis typically holds the whole dataset in memory. By default, it synchronizes data to the disk at least every 2 seconds, with more or less robust options available if needed. In the case of a complete system failure on default settings, only a few seconds of data would be lost. It supports 'Master-Slave Replication' i.e. Data from any Redis server can replicate to any number of slaves. Replication is useful for read scalability or data redundancy.

Redis can also be deployed on Amazon Web Services platform.

Redis essentially stores a user's feed that they could be fetched at any given time. Each user is assigned a unique media ID. When a user uploads a photo, the system finds out all of the user's followers and assigns individual tasks to place the photo into each followers feed. This data strategy is called a 'Fanout-On-Write approach'.

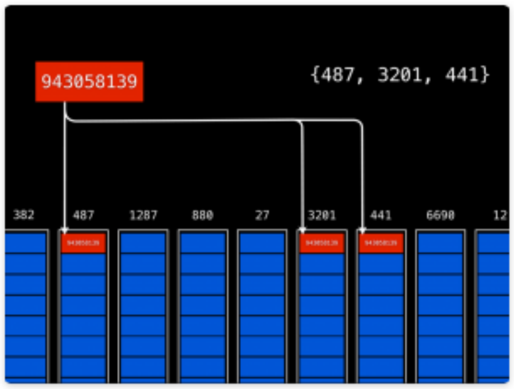


Fig. 5. Fanout-On-Write Approach

*C. Celery*

Celery [13] is an open source distributed task framework based on distributed message passing. It focuses on real time operation but supports scheduling as well. The tasks are executed asynchronously in the background or synchronously using wait-until-ready policy in a single server or multiple worker servers using multiprocessing. Instagram primarily uses celery as task manager.

*D. RabbitMQ*

RabbitMQ [12] is an open source message broker software that is primarily used for message routing.

When a web request pushes the post to the RabbitMQ broker, messages are distributed out to workers in a round robin style fashion. If a worker fails, the task is redistributed to the next worker.

It is based on Advanced Message Queuing Protocol (AMQP) that consists of rules and standards to pass messages like *atlease-once, atmost-once and exactly-once.*

Instagram uses RabbitMQ to display notifications in the follower's news feeds. However, having hit their high of over 10,000 connections of users simultaneously posting pictures, overall, 'Instagram' uses RabbitMQ for monitoring availability, scaling, and fault-tolerance.

## V. Technology Integration

The founders aimed at adopting simple and tried technologies, rather than experimenting and innovating.

### *A. Operating System Hosting*

The instagram application runs its central server upon the Amazon web services EC-2 (Electronic Cloud Compute). After much experimenting, the people at instagram realized that the Ubuntu Linux version 11.04 works best for them, since it is the most stable version, and gives continuous uninterrupted performance even under heavy user traffic conditions, which was a problem with the previous versions.

### *B. Application Servers*

Instagram makes use of the 'Django' application servers, which run on the Amazon high CPU- extra large machines. Instagram uses a total of about 25 such servers.

Application servers are similar to web servers which return http pages as the result to a request, but instead application servers provide the business logic to the application programs on user end, via multiple protocols.

Instagram experiences a great advantage in using these servers since majority of its work if CPU bound and not Memory bound, i.e. the time to complete the task is dependent upon the central processor's capacity and speed rather than the maximum mount of data that can be fit into the memory.

### *C. Data Storage*

Instagram being such a popular and widely used media platform has a huge amount of data to store. For this purpose it needs an efficient and dynamic storing mechanism, which is always in a consistent state and available. [14] For this reason, after careful research, they decided to go in favor of 'postgreSQL'. They use it as the backend database, to store all of the user data, photo metadata, tags, likes, etc.

Instagram gets 25 new photo uploads and more than 90 likes per second. At this rate the existing data, and the new data being getting added on in real time in really vast to store in an efficient manner. To help facilitate easy and convenient storage and access to the stored data, Instagram makes use of the 'Sharding' technique.

The sharding technique in postgreSQl helps ensure that the entire data fits into the memory, and is available to the user without any kind of delay or down time. The data is broken into smaller parts, each of which is then stored into a separate database, from amongst the multitude of databases [18] maintained by Instagram.

The major challenge faced while this technique was about how to assign a unique identifier to every piece of data in the database. [15] The traditional approach is using the primary key, as the unique identifier does not work, since the data is to be inserted across multiple databases simultaneously.

This issue was overcome by making use of the inbuilt 'schema' feature in postgreSQL (different from SQL schema). The schemas, act as the logical grouping feature. Each postgreSQL database can have several schemas, each with one or more tables. The table names need to be unique only within a schema, not within a database.

Several thousand of the created logical shards are mapped to a lesser number of

physical shards. Each of the logical shards is a schema, and each schema holds every sharded table within it. The unique IDs are created inside every table in the schema using the PL/PGSQL, (which is postgreSQl's inbuilt programming language), in an incremental manner.
Each of the created IDs has 3parts:
1. 41 bits of time; in msec.
2. 13 bits of logical shard ID.
3. 10 bits representing an auto increment sequence, modulus 1024.

*Example [14]*: lets say its September 9th 2011 5:00pm, and our epoch starts from January 1st 2011. There have been 1387263000illiseconds since the beginning of our epoch. So to calculate the ID, we fill the left most 41bits with this value with a left shift:

Id 1 =1387263000 <<(64-41)

Next, we take the shard ID for this particular piece of data we're trying to insert. Let's say we are sharding by user ID, and there are 2000 logical shards; if our user ID is 31341, then the shard ID is 31341 % 2000 >> 1341. We fill the next 13 bits with this value:

Id 1 = 1341<< (64-41-13)

Finally, we take whatever the next value of our auto-increment sequence (this sequence is unique to each table in each schema) and fill out the remaining bits. Let's say we'd generated 5,000 IDs for this table already; our next value is 5,001, which we take and mod by 1024 (so it fits in 10 bits) and include it too:

Id 1 = (5001 % 1024)

We now have our ID, which we can return to the application server using the RETURNING keyword.

*D. Backup*

With such a huge amount of data at hand to be handled, keeping the database in a consistent state continuously is extremely important. For this reason, the data is continuously backed up to the Amazon EBS (Elastic Block Storage), by setting up the EBS drives in software RAID.
EBS snapshotting is used to take frequent backups. The EC2 snapshotting is replaced by the XFS file system, in order to freeze and unfreeze the RAID system, to guarantee consistent snapshots.

*E. Connections*

Having such a large backend database, and a large number of application servers, there is a critical need to connect the two, to enable smooth working of the application. This connection between the two individual units is established by using the 'PgBouncer' connection pooler. It links the various application servers to the large database.

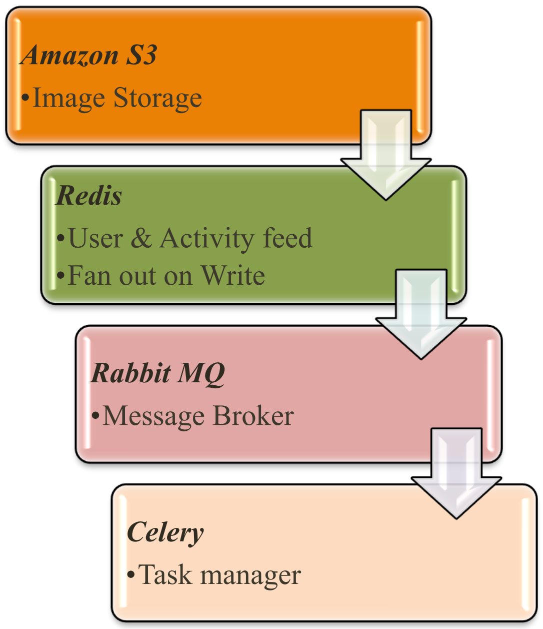


Fig. 6. Step-by-Step walk through of uploading an image in Instagram.

*F. Image Storage and Loading*

The images which a user uploads are immediately stored into the Amazon S3 storage system. Currently this data ranges into terabytes.

The Amazon CDN (Cloud-front) is used along with the Amazon S3 storage to boost the image load time, and make the user experience fast and smooth.

### *G. User Feed and Activity Feed*

Redis is an open source, BSD licensed, advanced key-value store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets.

Incase of 'Instagram' the 'key' is the ID of the user, and the 'value' is the list of the friends and followers.

### *H. Task Queue & Push Notifications*

When a user decides to share a photo on twitter or Facebook, or when a real time subscriber needs to be notified about a new photo posted, the built in system in Redis is used to fetch the user feed at any point of time.

The Redis system pulls out all of the user's followers and assigns individual tasks to place the photo in every followers feed. This is the 'Fan out on Write' approach.

To write out these posts onto follower feeds, a message broker and task manager are used.

RabbitMQ buffers the tasks and distributes them onto the various work queues, among the workers.

Celery executes the individual tasks of updating all of the follower and user feeds. It runs in the background, and completes all of the individual tasks, without affecting the user facing code in any manner.

### *I. Posting to Other Media Platforms*

Instagram makes use of the 'Pubsubhubuh' protocol [14], an open source protocol that is used for distributed publishing or communication over the Internet.

It is primarily used to publish photos from Instagram, onto follower feeds and other social media platforms.

It provides real time notifications of changes made to the user feed page, and it is applicable for any data type- text, picture, audio, video or anything else that can be shared via http.

### *J. System Monitoring*

Now that we have explored and discussed about the entire Instagram system, the need for system monitoring is clearly evident.

For this purpose Instagram uses 'Munin' [14], a networked resource-monitoring tool, used to graph metrics across all of the systems, and throw alerts whenever anything outside the normal range is observed.

## VI. CONCLUSION AND FUTURE WORK

In conclusion, our work has come to show that Instagram has a very unique and well-integrated architecture, which reflects on why it has become successful as it is today.

Despite the fact that we have come short in discovering more about the technologies used behind Instagram, we can further go into depth for future references or for our own personal interests. As a group, we learned a lot on how Instagram strategically modeled its data structure and absorbed a lot of new topics in relation to the procedures used to craft Instagram along the way. Yet, opposed to how we first picked up the project mindlessly, we have peaked over many new obstacles and tackled upon new designs along the process.

## REFERENCES

[1] Amazon Web Services:
http://en.wikipedia.org/wiki/Amazon_Web_Services

[2] M.Palankar, A.Onibokun, A.Iamnitchi and M.Ripeanu, "*Amazon S3 for Science Grids: a viable solution?"*

[3] Amazon S3 Pricing: https://aws.amazon.com/s3/pricing

[4] 70 Interesting Instagram Statistics: http://expandedramblings.com/index.php/important-instagram-stats/

[5] Amazon S3 Growth https://gigaom.com/2013/04/18/amazon-s3-goes-exponential-now-stores-2-trillion-objects/

[6] Amazon EC2 Official: https://aws.amazon.com/ec2/

[7] AWS Case Study: mobli https://aws.amazon.com/solutions/case-studies/mobli/

[8] Amazon CloudFront Documentation: https://aws.amazon.com/documentation/cloudfront/

[9] Amazon EBS Product Details: http://aws.amazon.com/ebs/details/#piops

[10] PostgreSQL About https://www.psotgresql.org/about/

[11] Introduction to Redis https://www.redis.io/topic/introduction/

[12] RabbitMQ Features https://www.rabbitmq.com/features.html/

[13] Celery: Distributed Task Queue https://www.celeryproject.org

[14] Instagram Engineering https://www.instagram-engineering.tumblr.com

[15] Thiago H. Silva, Pedro Vaz de Melo, Jussara M. Almeida, Juliana Salles and Antonio A.F. Louriero, "*A picture of Instagram is worth more than a thousand words: Workload characterization and application*", Microsoft Research, Redmond, WA, USA

[16] Garfinkel S.L, *"An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS"*

[17] Hibler J.N.D, Lueng C.H.C, Mannock K.L, *"System for content based storage and retrieval in an image database"*, PROC.SPIE1662, Image Storage and Retrieval Systems, April 1992.

[18] Wong, Stephen T. C.; Patwardhan, Anil Yeung, Minerva M.; Li, Chung-Sheng ;Lienhart, Rainer W, "*Storage and Retrieval for Media Databases"* 2002, Wednesday 23 January 2002, Vol.4676 (1), pp.248-263

# APPENDIX

1.

Instagram

Fast beautiful photo sharing

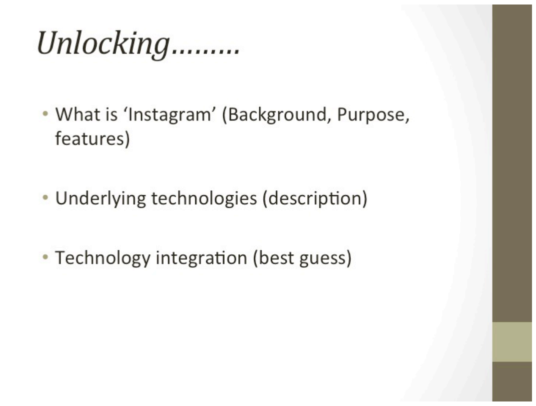
Unlocking the Secret

CSCI 5708

Group: Dataminators

Members: Van, Shyam, Dhruv, Song

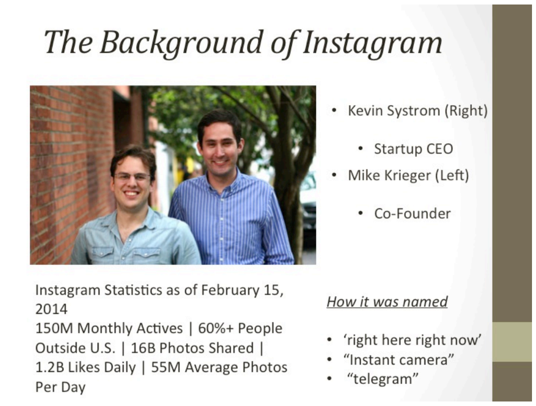
2.



## *Unlocking.........*

- What is 'Instagram' (Background, Purpose, features)
- Underlying technologies (description)
- Technology integration (best guess)

3.



## *The Background of Instagram*

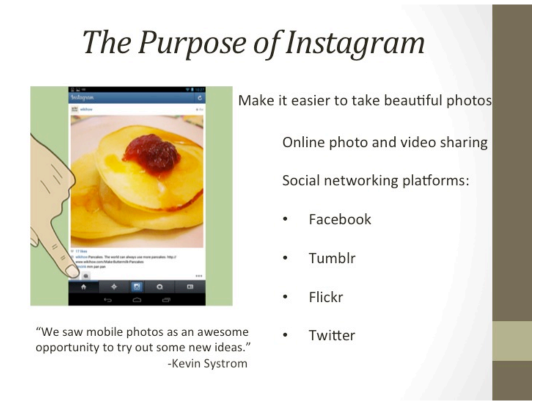
- Kevin Systrom (Right)
  - Startup CEO
- Mike Krieger (Left)
  - Co-Founder

Instagram Statistics as of February 15, 2014
150M Monthly Actives | 60%+ People Outside U.S. | 16B Photos Shared | 1.2B Likes Daily | 55M Average Photos Per Day

*How it was named*

- 'right here right now'
- "Instant camera"
- "telegram"

4.



## *The Purpose of Instagram*

Make it easier to take beautiful photos

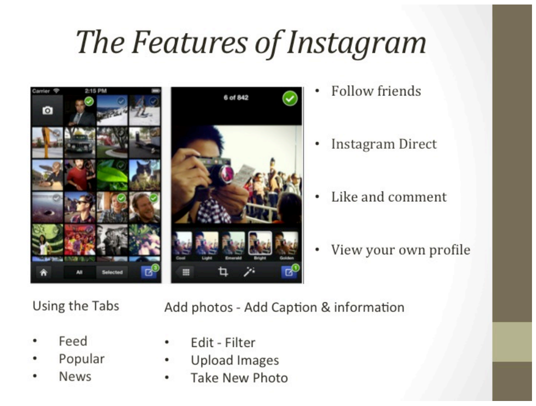
Online photo and video sharing

Social networking platforms:

- Facebook
- Tumblr
- Flickr
- Twitter

"We saw mobile photos as an awesome opportunity to try out some new ideas."
-Kevin Systrom

5.



## *The Features of Instagram*

- Follow friends
- Instagram Direct
- Like and comment
- View your own profile

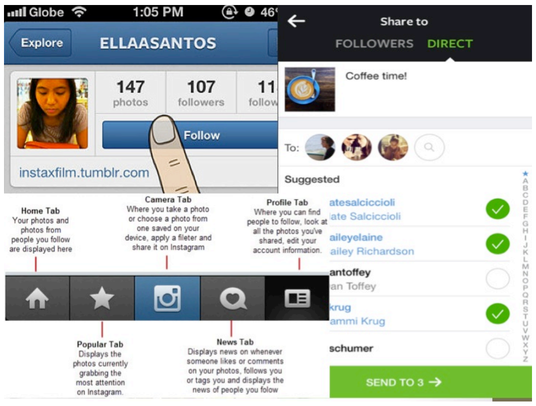
Using the Tabs

- Feed
- Popular
- News

Add photos - Add Caption & information

- Edit - Filter
- Upload Images
- Take New Photo

6.



**Home Tab**
Your photos and photos from people you follow are displayed here

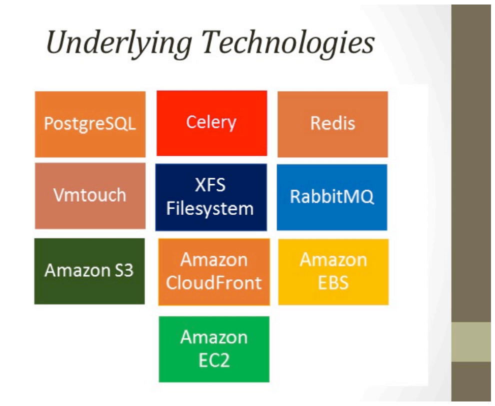
**Camera Tab**
Where you take a photo or choose a photo from one saved on your device, apply a fileter and share it on Instagram

**Profile Tab**
Where you can find people to follow, look at all the photos you've shared, edit your account information.

**Popular Tab**
Displays the photos currently grabbing the most attention on Instagram.

**News Tab**
Displays news on whenever someone likes or comments on your photos, follows you or tags you and displays the news of people you folow

7.



## Underlying Technologies

| | | |
|---|---|---|
| PostgreSQL | Celery | Redis |
| Vmtouch | XFS Filesystem | RabbitMQ |
| Amazon S3 | Amazon CloudFront | Amazon EBS |
| | Amazon EC2 | |

8.

amazon webservices™

- S3 — Scalable Storage in the Cloud
- EC2 — Virtual Servers in the Cloud
- CloudFront — Global Content Delivery Network
- CloudWatch — Resource and Application Monitoring
- Elastic MapReduce — Managed Hadoop Framework
- RDS — Managed Relational Database Service
- DynamoDB — Predictable and Scalable NoSQL Data Store

*And more..*

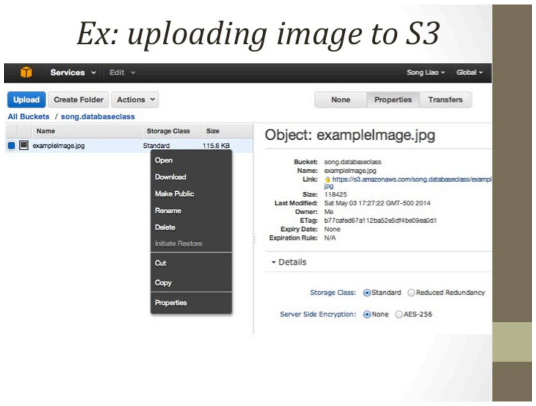
9.

## Amazon S3: Data Storage

- Store data objects(photos) up to 5GB in size
- Metadata in key-value pairs for each
- Search is limited in a single bucket
- Metadata or content-based search capabilities are not provided.

amazon webservices™
Amazon Simple Storage Service (S3)

10.



## Ex: uploading image to S3

All Buckets / song.databaseclass

| Name | Storage Class | Size |
|---|---|---|
| exampleImage.jpg | Standard | 115.6 KB |

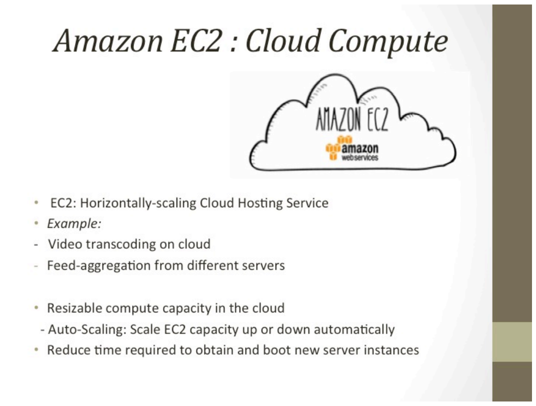
Object: exampleImage.jpg

Bucket: song.databaseclass
Name: exampleImage.jpg
Link: https://s3.amazonaws.com/song.databaseclass/exampl jpg
Size: 118425
Last Modified: Sat May 03 17:27:22 GMT-500 2014
Owner: Me
ETag: b77cafed67a112ba52e5df4be09ea0d1
Expiry Date: None
Expiration Rule: N/A

▾ Details

Storage Class: [✓ Standard] Reduced Redundancy
Server Side Encryption: [✓ None] AES-256

11.



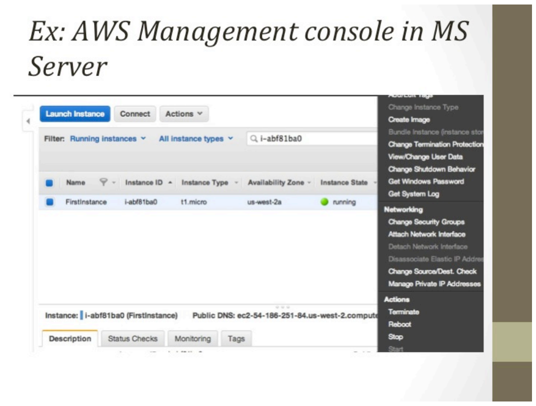
## Amazon EC2 : Cloud Compute

AMAZON EC2
amazon webservices

- EC2: Horizontally-scaling Cloud Hosting Service
- *Example:*
- Video transcoding on cloud
- Feed-aggregation from different servers

- Resizable compute capacity in the cloud
- Auto-Scaling: Scale EC2 capacity up or down automatically
- Reduce time required to obtain and boot new server instances

12.



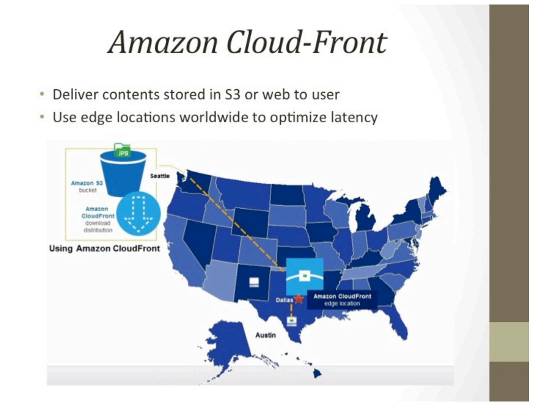
## Ex: AWS Management console in MS Server

Filter: Running instances All instance types i-abf81ba0

| Name | Instance ID | Instance Type | Availability Zone | Instance State |
|---|---|---|---|---|
| FirstInstance | i-abf81ba0 | t1.micro | us-west-2a | running |

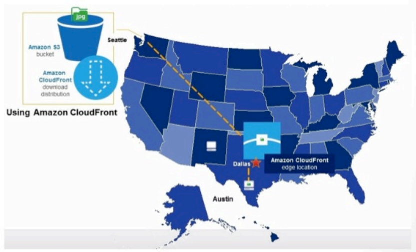
Instance: i-abf81ba0 (FirstInstance) Public DNS: ec2-54-186-251-84.us-west-2.compute

13.

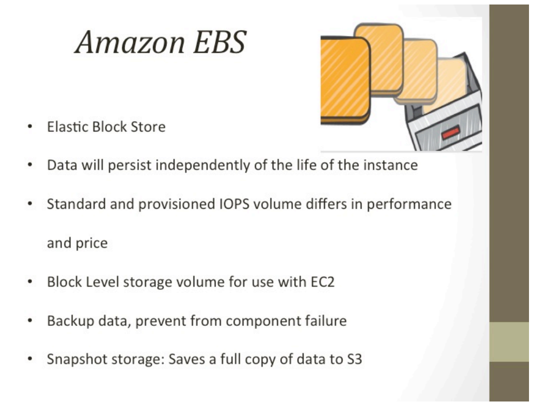


## Amazon Cloud-Front

- Deliver contents stored in S3 or web to user
- Use edge locations worldwide to optimize latency


Amazon S3 bucket
Amazon CloudFront download distribution
Using Amazon CloudFront
Seattle
Dallas
Amazon CloudFront edge location
Austin

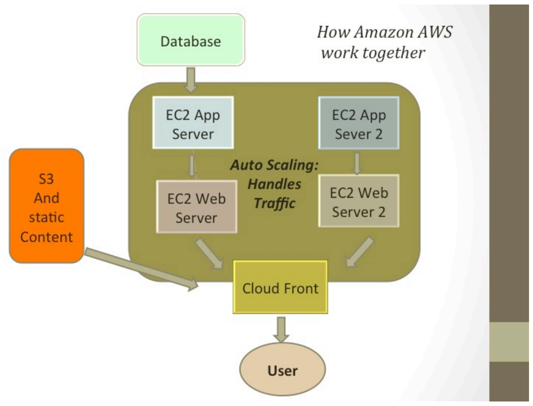
14.



## Amazon EBS

- Elastic Block Store
- Data will persist independently of the life of the instance
- Standard and provisioned IOPS volume differs in performance and price
- Block Level storage volume for use with EC2
- Backup data, prevent from component failure
- Snapshot storage: Saves a full copy of data to S3

15.

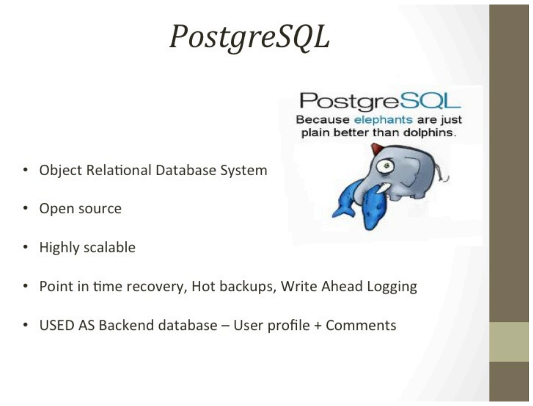


16.

## Other Technologies

PostgreSQL
Celery
redis
RabbitMQ

17.

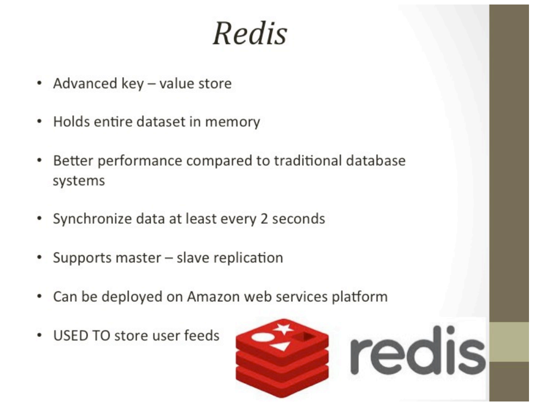


## PostgreSQL

PostgreSQL
Because elephants are just plain better than dolphins.

- Object Relational Database System
- Open source
- Highly scalable
- Point in time recovery, Hot backups, Write Ahead Logging
- USED AS Backend database – User profile + Comments

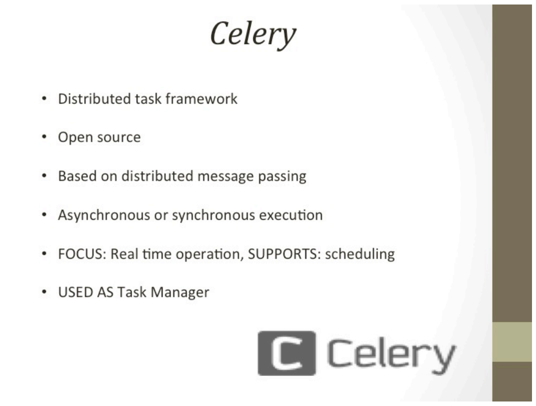
18.



## Redis

- Advanced key – value store
- Holds entire dataset in memory
- Better performance compared to traditional database systems
- Synchronize data at least every 2 seconds
- Supports master – slave replication
- Can be deployed on Amazon web services platform
- USED TO store user feeds

19.

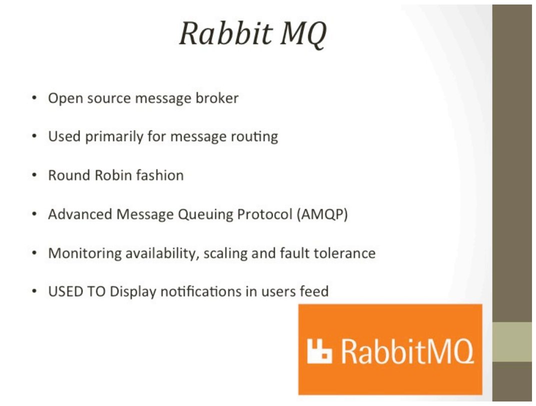


## *Celery*

- Distributed task framework
- Open source
- Based on distributed message passing
- Asynchronous or synchronous execution
- FOCUS: Real time operation, SUPPORTS: scheduling
- USED AS Task Manager

Celery

20.

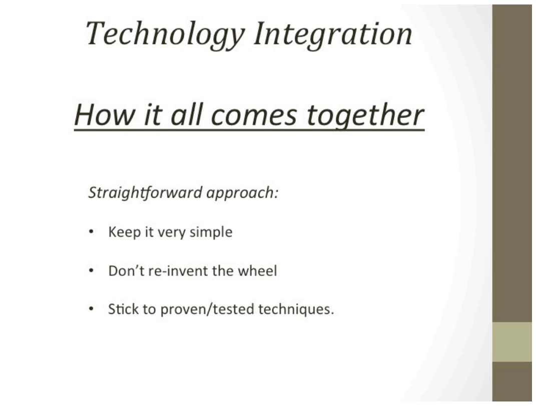


## *Rabbit MQ*

- Open source message broker
- Used primarily for message routing
- Round Robin fashion
- Advanced Message Queuing Protocol (AMQP)
- Monitoring availability, scaling and fault tolerance
- USED TO Display notifications in users feed

RabbitMQ

21.



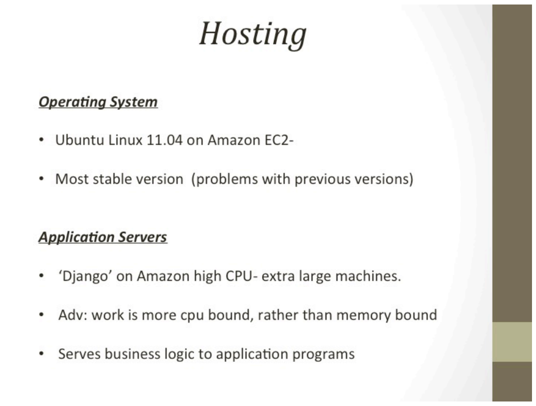
## *Technology Integration*

## *How it all comes together*

*Straightforward approach:*

- Keep it very simple
- Don't re-invent the wheel
- Stick to proven/tested techniques.

22.



## *Hosting*

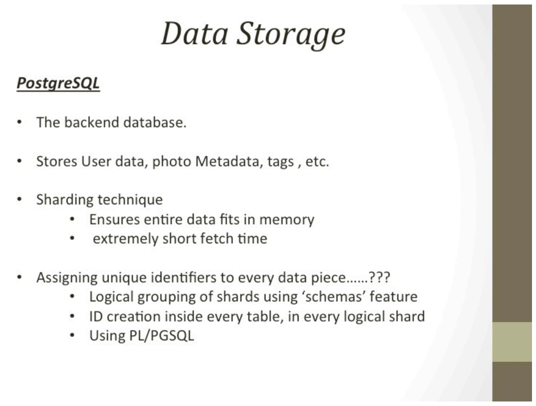
***Operating System***

- Ubuntu Linux 11.04 on Amazon EC2-
- Most stable version  (problems with previous versions)

***Application Servers***

- 'Django' on Amazon high CPU- extra large machines.
- Adv: work is more cpu bound, rather than memory bound
- Serves business logic to application programs

23.

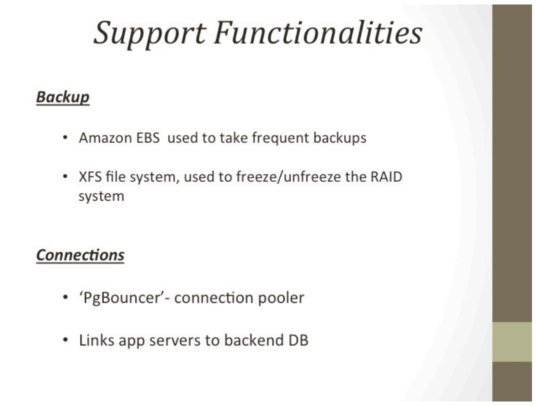


## *Data Storage*

***PostgreSQL***

- The backend database.
- Stores User data, photo Metadata, tags , etc.
- Sharding technique
  - Ensures entire data fits in memory
  - extremely short fetch time
- Assigning unique identifiers to every data piece……???
  - Logical grouping of shards using 'schemas' feature
  - ID creation inside every table, in every logical shard
  - Using PL/PGSQL

24.



## *Support Functionalities*

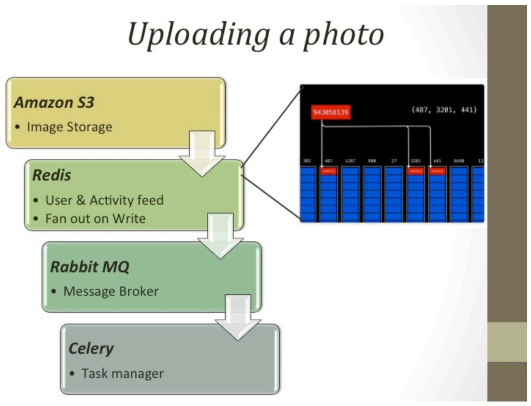
***Backup***

- Amazon EBS  used to take frequent backups
- XFS file system, used to freeze/unfreeze the RAID system

***Connections***

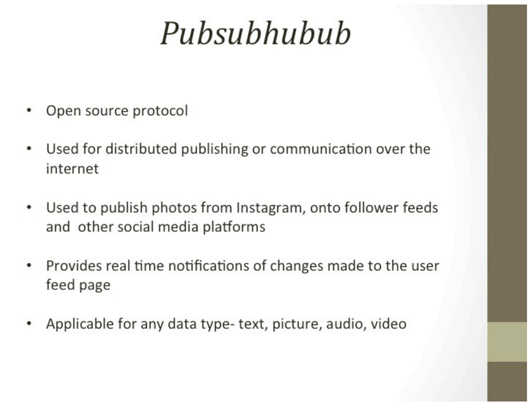
- 'PgBouncer'- connection pooler
- Links app servers to backend DB

25.



## *Uploading a photo*

26.



## *Pubsubhubub*

- Open source protocol
- Used for distributed publishing or communication over the internet
- Used to publish photos from Instagram, onto follower feeds and other social media platforms
- Provides real time notifications of changes made to the user feed page
- Applicable for any data type- text, picture, audio, video

27.

*Questions..?????*