CLOUD COMPUTING APPLICATIONS

Serverless Cloud Computing: Landscape

Prof. Reza Farivar

Serverless Cloud Computing Landscape

- Different Categories of Serverless Computing
  - Compute
    - Platform as a Service (Apps)
    - Function as a Service (Functions)
    - Container as a Service (Containers)
  - Storage
    - Blobs (Binary Large Objects)
    - Key/Value Datastores
  - Analytics
  - AI and ML

Compute: PaaS

- The unit of compute is a full App
- You still select which server / platform you need, only that you don't need to manage it anymore

- Amazon Elastic BeanStalk
- Google AppEngine
- Microsoft Azure App Service
- IBM CloudFoundry
  - Based on Open Source CloudFoundry
    - Originally developed by VMware, transferred to Pivotal Software (a joint venture by EMC, VMware and General Electric), brought back into VMware at the end of 2019
- Oracle Java Cloud Service

Function as a Service

- This is what most people think of as “Serverless”
- Unit of compute is a function
  - Functions running when “events” are triggered

- Amazon AWS Lambda
- Microsoft Azure Functions
- Google Cloud Functions
- IBM Cloud Functions
  - Based on Apache Open Whisk
- Oracle Functions
  - Apache Fn

- Open Source Open Lambda

Container as a Service

- Function as a Service on HEAVY steroids!
  - Containers have to be Stateless
  - Unit of compute is a whole container
  - Container running when “events” are triggered

- Amazon Elastic Container Service (ECS), Elastic Kubernetes Service (EKS), Fargate
- Microsoft Azure Kubernetes Service (AKS)
  - Built on top of Open Source KEDA
- Google Cloud Run, Anthos
  - Built on top of Open Source Knative
    - Kubernetes-based platform to deploy and manage modern serverless workloads.
- IBM Cloud Kubernetes Service
- Oracle Container Engine for Kubernetes

- Open Source Kubernetes
  - Google Borg

Serverless Storage: Blobs

- Blob Storage

- Amazon Simple Storage Service (S3)
- Microsoft Azure Blob Storage
- IBM Object Storage
- Google Cloud Storage
- Oracle Object Storage

Serverless Storage: Key/Value Database

- Distributed NOSQL key/value storage service

- Amazon DynamoDB
- Microsoft Azure Cosmos DB
- Google Cloud Datastore, Firestore, Cloud BigTable
- IBM Cloudant

- Open Source: Kassandra

CLOUD COMPUTING APPLICATIONS

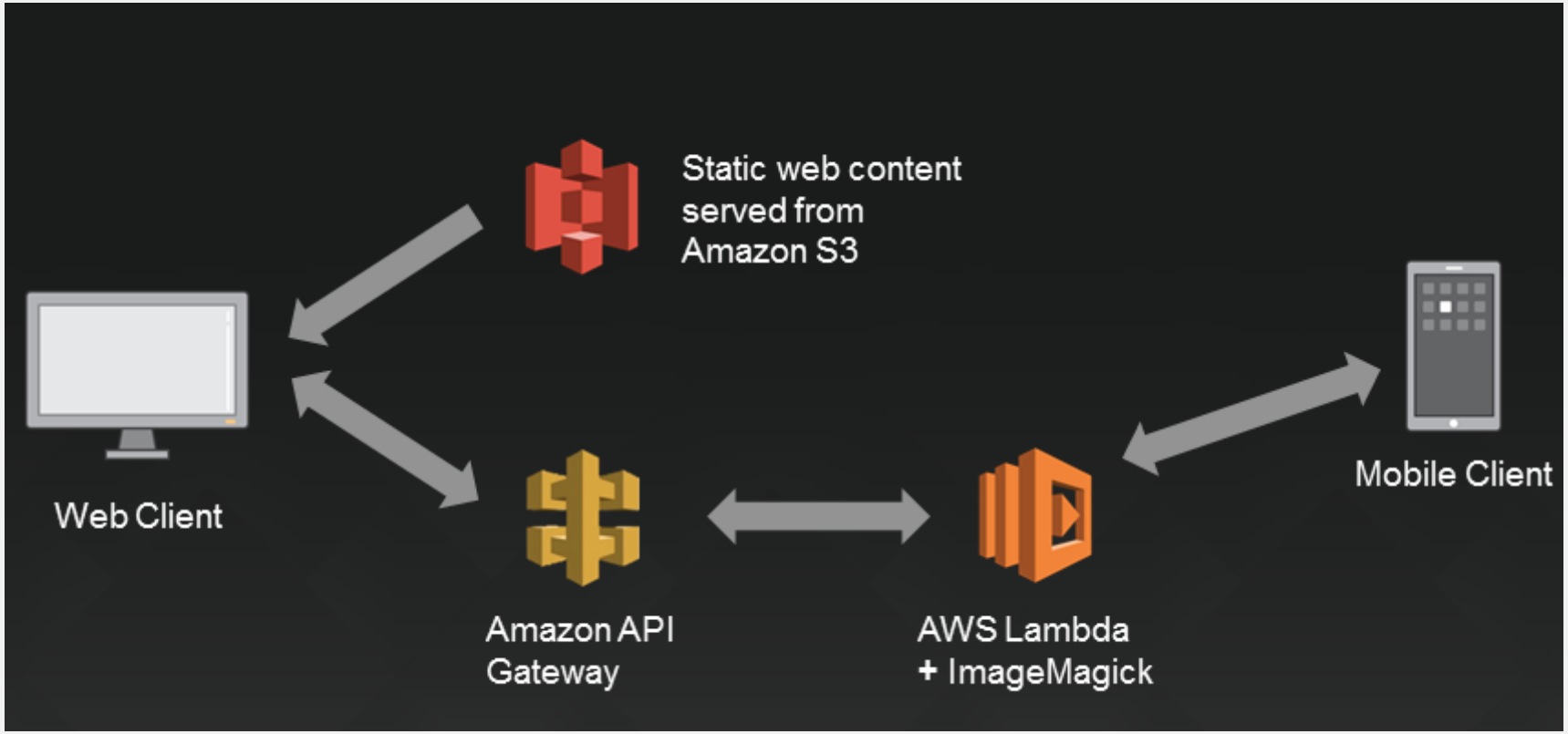
Serverless Architecture

Roy Campbell & Reza Farivar

Introduction to Serverless Architecture

- “Applications where some amount of server-side logic is still written by the application developer but unlike traditional architectures is run in stateless compute containers that are event-triggered, ephemeral (may only last for one invocation), and fully managed by a 3rd party”
- ‘Functions as a service / FaaS’
- AWS Lambda is one of the most popular implementations of FaaS at present, but there are others

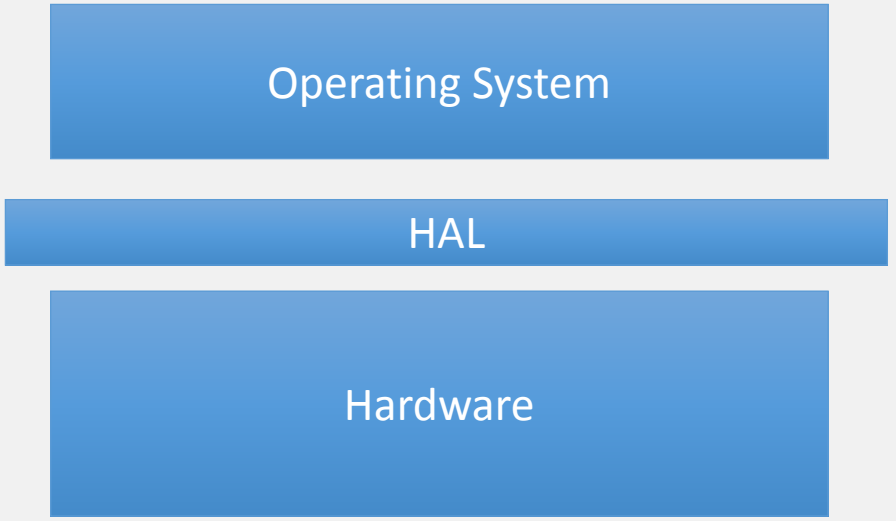
Introduction to Serverless Architecture



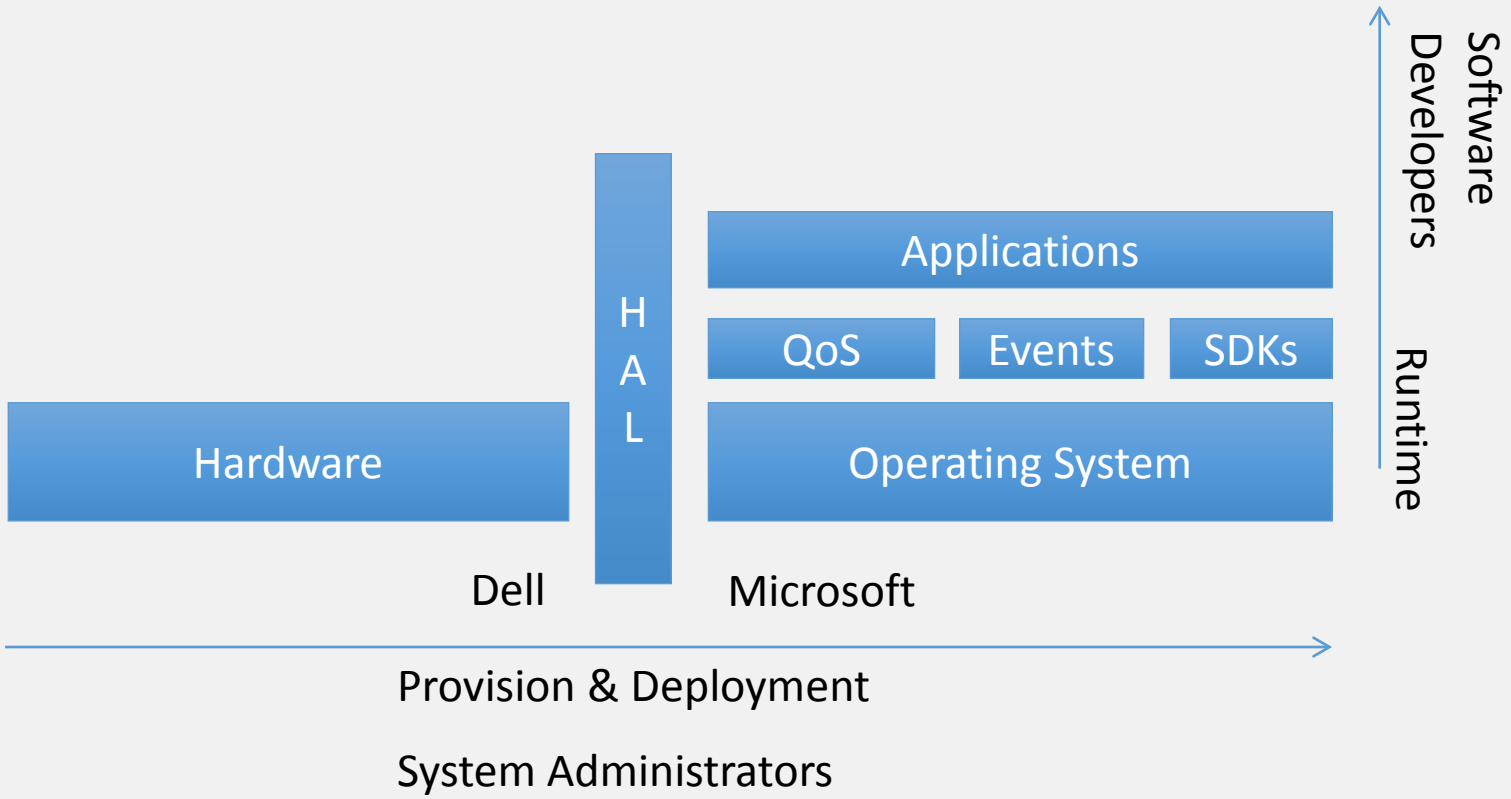
Desktop Platform

Microsoft

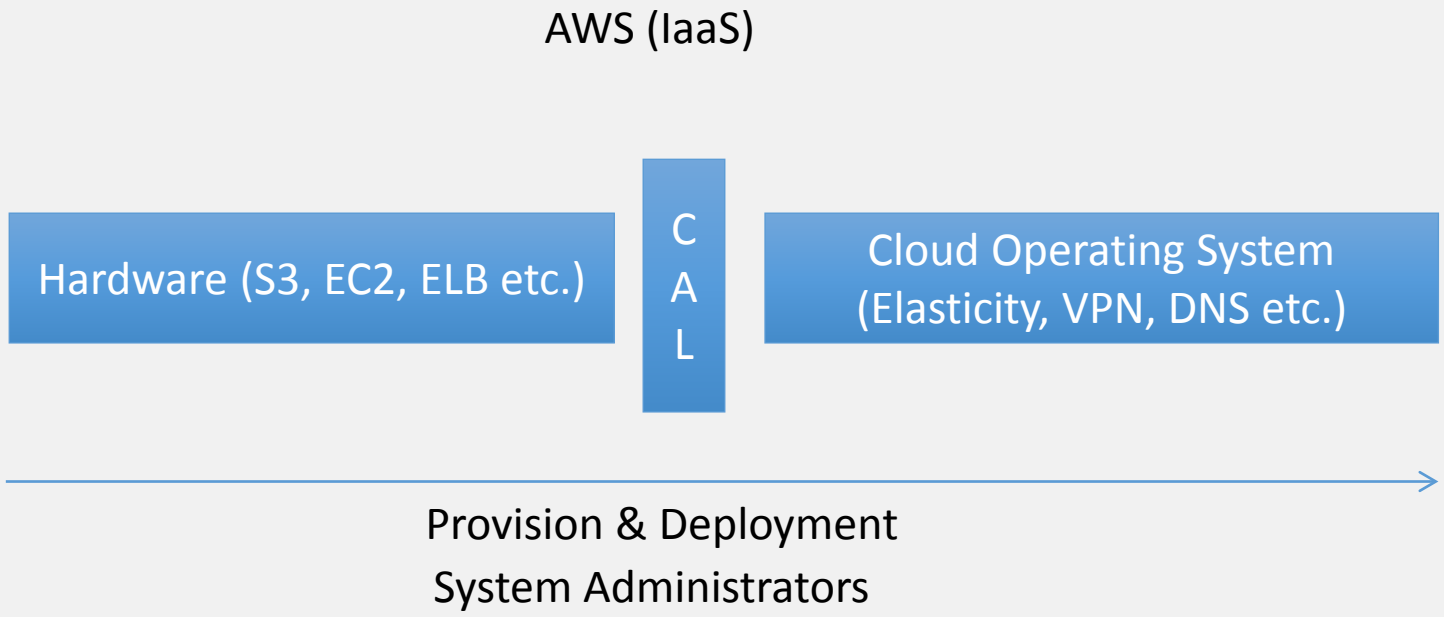
Dell



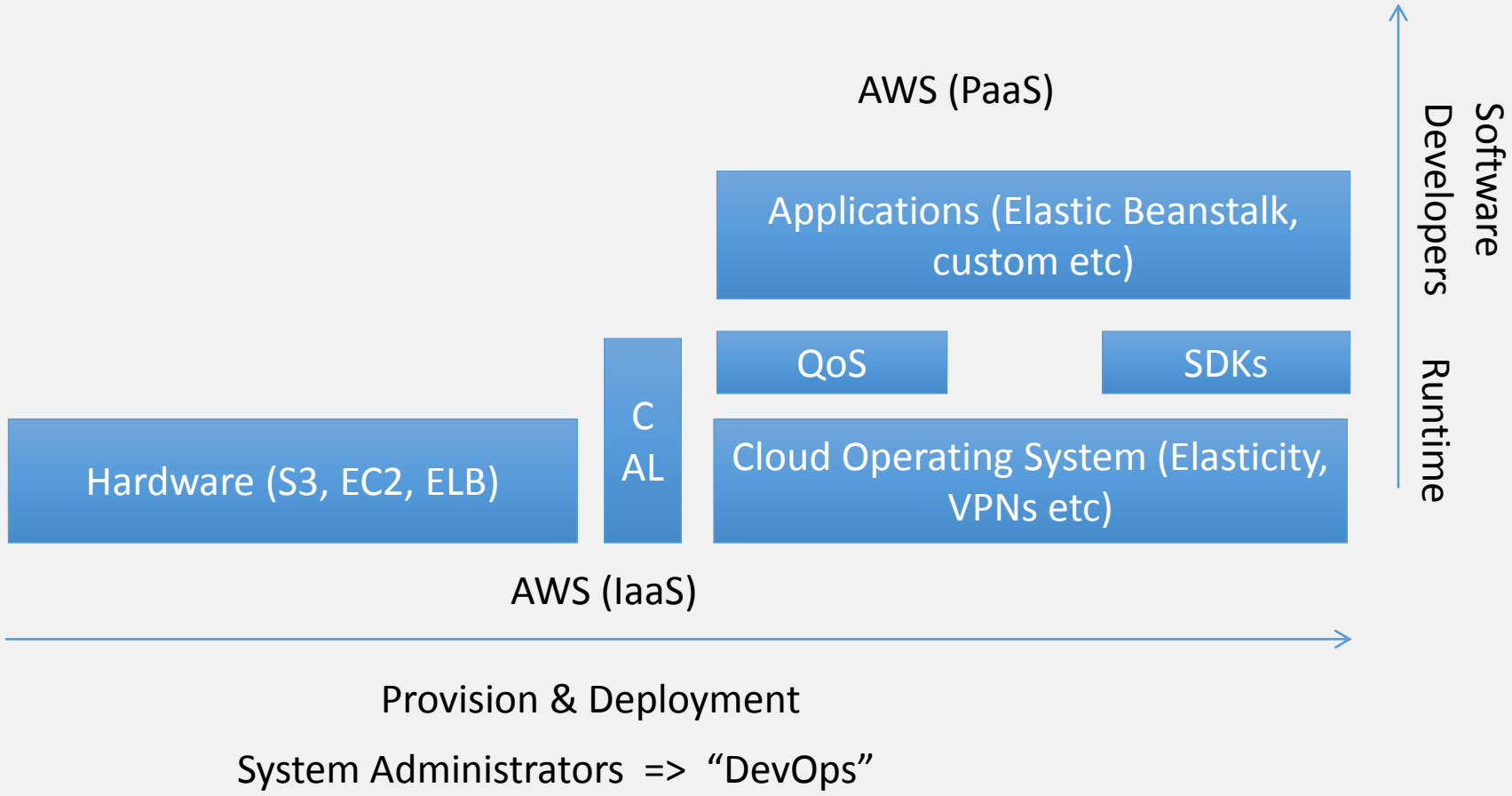
Desktop Platform



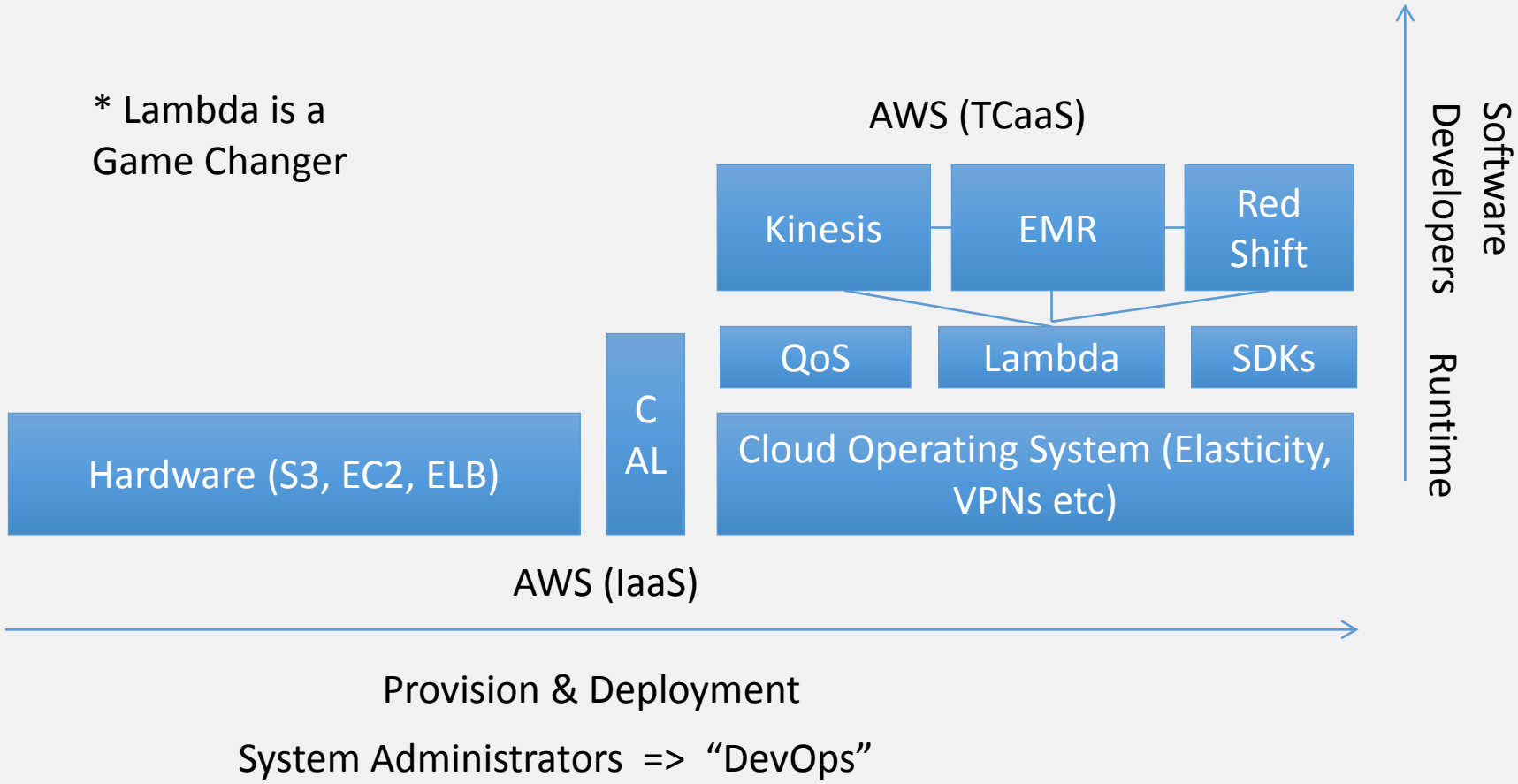
AWS Cloud Platform, 2010



AWS Cloud platform, 2014



AWS Cloud Platform 2016



AWS Elastic BeanStalk

- Deploy and scale web applications easily
- Languages: Java, .NET, PHP, Node.js, Python, Ruby, Docker
- Servers: Apache, Nginx, Phusion Passenger, IIS
- Simply upload your code; AWS handles:

Deployment

Capacity Provisioning

Load balancing

Auto scaling

Health Monitoring

AWS Lambda Event-driven Compute

- Runs stateless, request-driven code called **Lambda functions in Java, NodeJS & Python**
- Triggered by events (state transitions) in other AWS services
- Pay only for the requests served and the compute time
- Focus on business logic, not infrastructure.
- Just upload your code; AWS Lambda handles:

Capacity	**Monitoring**	**Fault Tolerance**
Scaling	**Logging**	**Security Patching**
Deployment	**Web service front end**	

AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- Scheduled Events (powered by Amazon CloudWatch Events)
- AWS Config
- Amazon Echo
- Amazon API Gateway
- Other Event Sources: Invoking a Lambda Function On Demand
- Sample Events Published by Event Sources

AWS Lambda Execution Environment

- State-less functions
- You can use multi-threading, etc.
- 500 MB of /tmp storage space
- You set how much memory you need:
  - From 128 MB to 1.5GB
  - 64GB increments
  - CPU scales accordingly
- Function should finish in a certain time
  - Default 3 seconds, up to 300 seconds

AWS Lambda Pricing

- You pay per use of your function
- $0.20 per 1 million function call
- Also, $0.00001667 for every GB-second used

CLOUD COMPUTING APPLICATIONS

Amazon S3 BLOB Storage

Roy Campbell & Reza Farivar

Definition

Online file storage web service offered by Amazon Web Services. Amazon S3 provides storage through web service interfaces REST, SOAP, BitTorrent. (wikipedia)

https://aws.amazon.com/s3/

Use case

- Scalability, high availability, low latency – 99.99% availability
- Files up to 5 terabytes
- Objects stored in buckets owned by users
- User assigned keys refer to objects

- Amazon Machine Images (exported as a bundle of objects)
- SmugMug, Hadoop file store, Netflix, reddit, Dropbox, Tumbler

Simple Storage Service (S3)

- A **bucket** is a container for objects and describes location, logging, accounting, and access control.
  - A bucket has a name that must be **globally unique**.
  - http://*bucket*.s3.amazonaws.com
  - http://*bucket*.s3-*aws-region*.amazonaws.com.

- A bucket can hold any number of **objects**, which are files of up to 5TB.
  - http://bucket.s3.amazonaws.com/object
  - http://johnsmith.s3.amazonaws.com/photos/puppy.jpg

Fundamental operations corresponding to HTTP actions:

http://**bucket**.s3.amazonaws.com/**object**

- POST a new object or update an existing object.
- GET an existing object from a bucket.
- DELETE an object from the bucket
- LIST keys present in a bucket, with a filter.

A bucket has a **flat directory structure**

S3 Weak Consistency Model

“Updates to a single key are **atomic**….”

“Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However:

- A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might report "key does not exist."
- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.”

S3 Command Line Interface

```
aws    s3   mb    s3://bucket
. . .       cp    localfile   s3://bucket/key
            mv    s3://bucket/key    s3://bucket/newname
            ls    s3://bucket
            rm    s3://bucket/key
            rb    s3://bucket

aws   s3    help
aws   s3    ls help
```

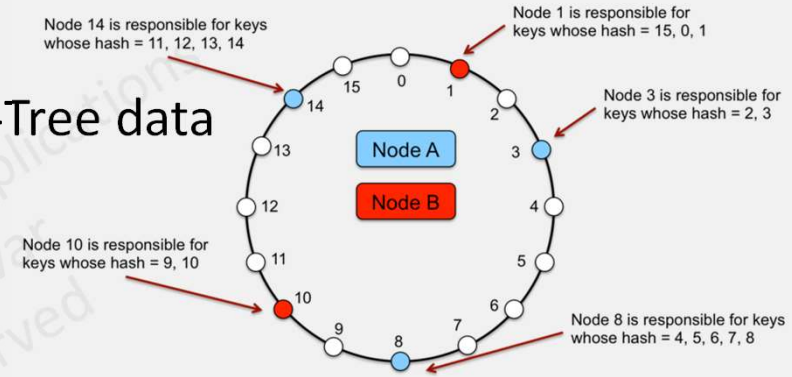
CLOUD COMPUTING APPLICATIONS

Serverless Storage: DynamoDB

Prof. Reza Farivar

DynamoDB

- DynamoDB is a fully managed NoSQL database provided by Amazon AWS
- Think of it as a massive distributed B-Tree data structure in the cloud
  - Accessing specific items is blazingly fast
- Distributed system
  - Using the consistent hashing algorithm in a ring



Usage model

- First create a table
  - Remember that it's a managed service, so just create a table using the console (or CLI, API)
    - We will use the Python Boto3 package in this lesson
- While creating the table, define the primary key
  - This key will be used by DynamoDB to distribute key/values in different partitions
- Optionally, identify a sort key
  - The sort key is used to keep the items in a partition sorted
  - Will be useful for query and scan later on

Using the table: Put

- Having defined the table, we can now put values into it.
  - Note: DynamoDB items are limited to 400KB size

```
import boto3
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('users')

table.put_item(
      Item={
            'username': 'janedoe',
            'first_name': 'Jane',
            'last_name': 'Doe',
            'age': 25,
            'account_type': 'standard_user',
      }
)
```

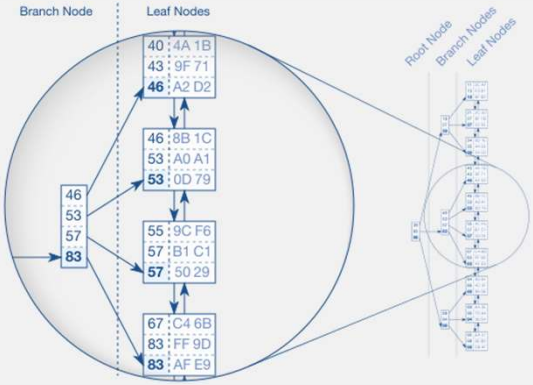
Using the table: Get

- Retrieve an item

```
response = table.get_item(
    Key={
        'username': 'janedoe',
        'last_name': 'Doe'
    }
)
item = response['Item']
print(item)



#Expected output:

{u'username': u'janedoe',
 u'first_name': u'Jane',
 u'last_name': u'Doe',
 u'account_type': u'standard_user',
 u'age': Decimal('25')}
```

Query and Scanning

- In RDBMS world*, query is usually defined as an operation where there is a usable index available, and we can quickly retrieve the item in Log(n) time
- In comparison, scan happens where there is no usable index, and the engine has to read every record and test for a condition
- An RDBMS engine parses a SELECT statement, and performs query optimization, all behind the curtain
- DynamoDB just allows you to be your own database engine!



*See https://use-the-index-luke.com/sql/where-clause/searching-for-ranges/greater-less-between-tuning-sql-access-filter-predicates

Query

- Query only works on the primary key already defined for the table
- Or any other attribute that we have explicitly made a secondary index for it
- If a composite primary key was used (hash key + sort key), we can ask query to return a conditional range of value

```
response = table.query(
    KeyConditionExpression=Key('username').eq('johndoe')
)
items = response['Items']
print(items)


#Expected output:

[{u'username': u'johndoe',
  u'first_name': u'John',
  u'last_name': u'Doe',
  u'account_type': u'standard_user',
  u'age': Decimal('25'),
  u'address': {u'city': u'Los Angeles',
               u'state': u'CA',
               u'zipcode': Decimal('90001'),
               u'road': u'1 Jefferson Street'}}]
```

Scan

- What if we want to perform a query conditioned on attributes that there is no index for them?
- Scan will return everything!
  - It allows to filter based on any arbitrary condition

```
response = table.scan(
    FilterExpression=Attr('age').lt(27)
)
items = response['Items']
print(items)
```

Secondary Index

- Similar to the main index, requires a partition key and a sort key
- Local (LSI)
  - First released by Amazon in 2013
  - Immediately consistent
  - Once created, the table cannot grow any more
    - All the records tha share the same partition key need to fit in 10GB
    - Once the allocation is full, writes fail ☹
- Global (GSI)
  - Came a few months after local indexes
    - Eventual consistency model
    - Do not constrain table size ☺

CLOUD COMPUTING APPLICATIONS

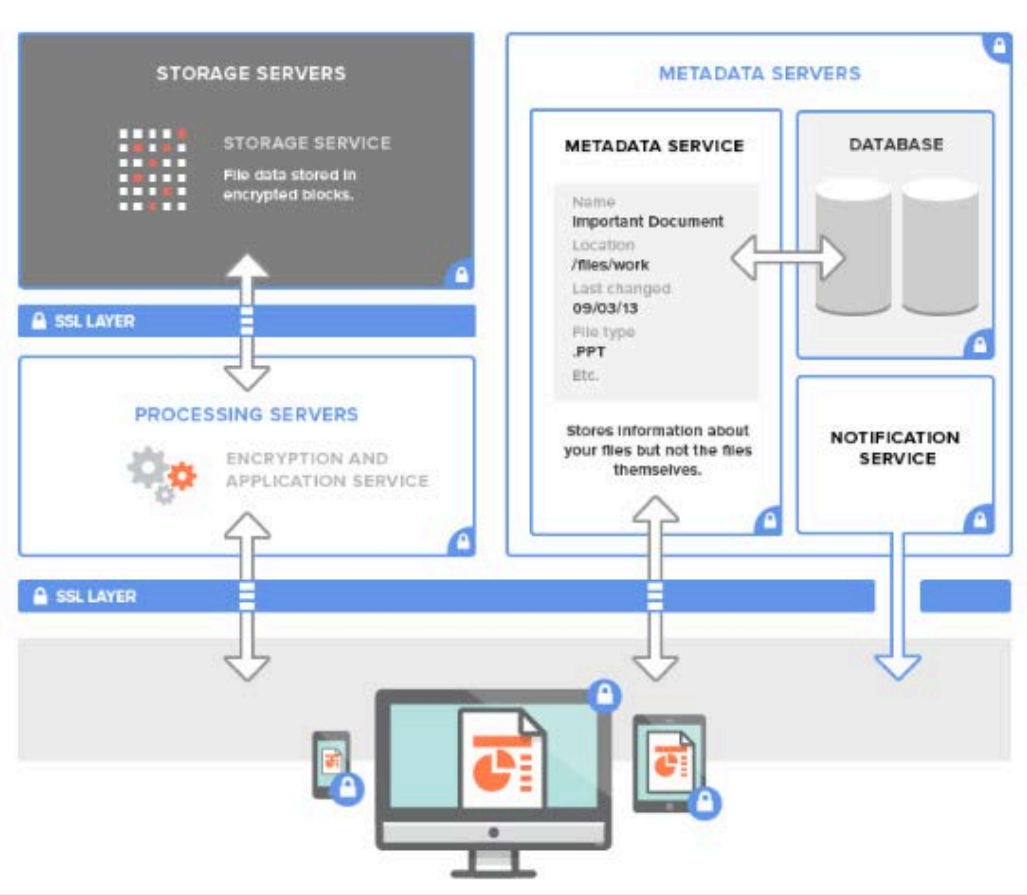
Dropbox Cloud API

Roy Campbell & Reza Farivar

Cloud Storage

- One interesting case study is Dropbox
- Dropbox offers cloud file storage
  - Easily synced across multiple devices
  - Accessible through web interface, mobile apps, and directly integrated with the file system on PCs
- Dropbox itself uses clouds!
  - Metadata stored in Dropbox servers
  - Actual files stored in Amazon S3
  - Amazon EC2 instances run the logic

Dropbox Architecture

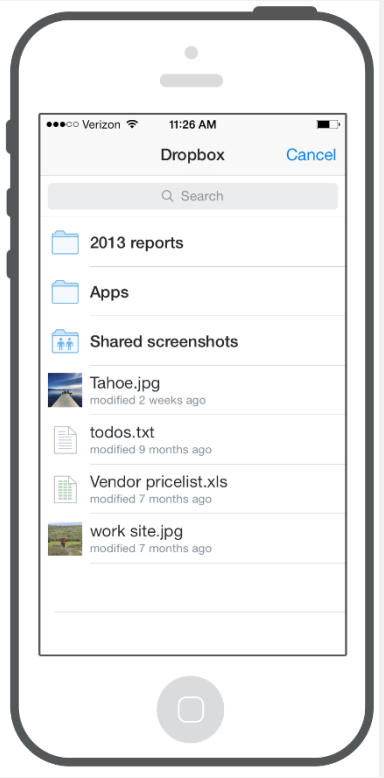


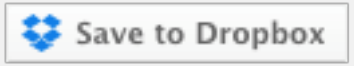
Dropbox API

- Two levels of API access to Dropbox
  - Drop-ins
    - Cross-platform UI components that can be integrated in minutes
    - *Chooser* allows instant access to files in Dropbox
    - *Saver* makes saving files to Dropbox easy
  - Core API
    - Support for advanced functionality like search, revisions, and restoring file
    - Better fit for deeper integration

Drop-In API

- Simple objects
  - *Chooser* available for JavaScript, Android and iOS
  - *Saver* on web and mobile web
- Handles all the authentication (OAuth), file browsing
- Chooser object returns the following:
  - Link: URL to access the file
  - File name
  - File Size
  - Icon
  - Thumbnails
- Saver
  - Pass in URL, filename and options





Core API

- Many languages and environments
  - Python, Ruby, PHP, Java, Android, iOS, OS X, HTTP
- Based on HTTP and OAuth
  - OAuth v1, OAuth v2
- Low-level calls to access and manipulate a user's Dropbox account
  - Create URL schemes
  - Upload files
  - Download files
  - List files and folders
  - Delta
  - Metadata access
  - Create and manage file sharing