CLOUD COMPUTING

REVIEW CLASS

MAIN TOPICS

BigData

Business Model

Definition of Cloud

Docker PART 1 & PART 2

Distributed Computing

RestFul APIs & Webservers

IAAS, PAAS, SAAS

Grid Computing

Cloud Storage

Databases in the Cloud

NoSQL-Part1 & Past 2

Security in the Cloud

CHALLENGES OF BIGDATA

Large Data will be measured in Exabytes not Gigabytes

Data won’t fit on a single device or datastore

Data needs to be processed to become useful

Data will naturally be more distributed

CLOUD COMPUTING (NIST, 2009)

CLOUD COMPUTING DEFINITION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models (NIST, 2009).

4 KEY CHARACTERISTICS OF CLOUD COMPUTING

- On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability1 at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.
- Resource pooling. The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

DEPLOYMENT MODELS OF CLOUD COMPUTING

DEPLOYMENT MODELS OF CLOUD COMPUTING

- Private cloud. The cloud infrastructure is provisioned for exclusive use by a single organisation comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organisation, a third party, or some combination of them, and it may exist on or off premises.
- Community cloud. The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organisations in the community, a third party, or some combination of them, and it may exist on or off premises.
- Public cloud. The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- Hybrid cloud. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

Business Model Advantages Of Cloud Computing Vs Traditional Hosting

ADVANTAGES OF CLOUD COMPUTING

- Simplified Customer acquisition
- Elastic demand
- Utility pricing
- Developer eco-system
- Businesses can make relatively low-risk bets to test market interest before committing significant spending.
- Strategic investment inflection points more visible

TEXT

VIRTUALISED COMPUTING.

CLOUD INFRASTRUCTURE SERVICES

Software as a Service (SaaS)

Platform as a Service (PaaS).

Infrastructure as a Service (IaaS).

SOFTWARE AS A SERVICE (SAAS).

The capability provided to the consumer is to use the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. For example, Gmail, Dropbox, Office 365

PLATFORM AS A SERVICE (PAAS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment. For example , Google App Engine, Nitrous, Cloud9, azure visual studio

INFRASTRUCTURE AS A SERVICE (IAAS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). Example of IAAS: Google Cloud, AWS, Azure, DigitalOcean

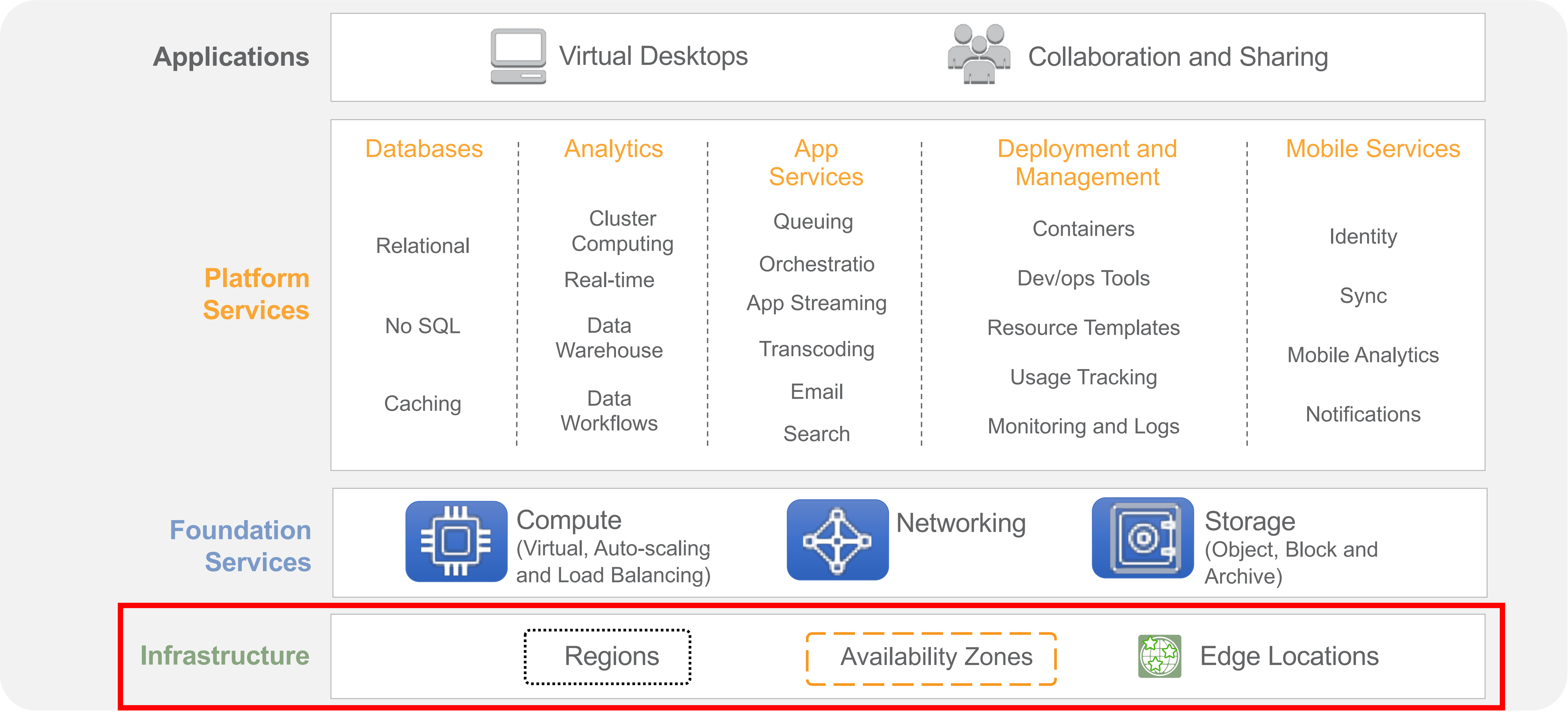
OVERVIEW OF AWS SERVICE

AWS

SIX ADVANTAGES OF CLOUD COMPUTING

- Trade capital expense for variable expense
- Benefit from massive economies of scale
- Stop guessing about capacity
- Increase speed and agility
- Stop spending money running and maintaining data centers
- Go global in minutes

AWS Global Infrastructure



Applications

Virtual Desktops

Collaboration and Sharing

Platform Services

Databases
- Relational
- No SQL
- Caching

Analytics
- Cluster Computing
- Real-time
- Data Warehouse
- Data Workflows

App Services
- Queuing
- Orchestratio
- App Streaming
- Transcoding
- Email
- Search

Deployment and Management
- Containers
- Dev/ops Tools
- Resource Templates
- Usage Tracking
- Monitoring and Logs

Mobile Services
- Identity
- Sync
- Mobile Analytics
- Notifications

Foundation Services

Compute (Virtual, Auto-scaling and Load Balancing)

Networking

Storage (Object, Block and Archive)

Infrastructure

Regions

Availability Zones

Edge Locations

CORE SERVICES

Compute:

Different AWS compute services in the cloud

AWS Lambda and serverless computing

AWS Elastic Beanstalk

CORE SERVICES

Storage:

Amazon Elastic Block Store (Amazon EBS)

Amazon Simple Storage Service (Amazon S3)

Amazon Elastic File System (Amazon EFS)

Amazon Glacier

CORE SERVICES

VPC:

Amazon Virtual Private Cloud (Amazon VPC)

Amazon VPC Security Groups

Amazon CloudFront

CORE SERVICES

Database:

Amazon Relational Database Service (Amazon RDS)

Amazon DynamoDB

Amazon Redshift

Amazon Aurora

CORE SERVICES

ELB & Autoscaling:

Elastic Load Balancing (ELB)

Amazon CloudWatch

Auto Scaling

SECURITY

AWS Shared Responsibility Model

IAM users, groups and roles

Different types of security credentials

WS Trusted Advisor checks

security compliance

Understand best practices on Day 1 with a new AWS account

CLOUD ARCHITECTING

Well-Architected Design Principles:

- Stop guessing your capacity needs.
- Test systems at production scale.
- Automate to make architectural experimentation easier.
- Allow for evolutionary architectures.
- Drive architectures using data.
- Improve through game days.

RELIABILITY AND HIGH AVAILABILITY

Understanding Reliability and High Availability

Reliability:

- Probability that entire system functions for a specified period of time
    - Includes hardware, firmware, and software
    - Measure of how long the item performs its intended function
- Two common measures of reliability:
    - Mean Time Between Failure (MTBF) – Total time in service/number of failures
    - Failure Rate – Number of failures/total time in service

DISTRIBUTED SYSTEM & ARCHITECTRUE

ADVANTAGES OF DISTRIBUTED ARCHITECTURE

- Cloud has enabled the creation of new businesses which may otherwise not exist
- Could be entire new start-ups or new business units within existing companies
- Cloud’s low friction, low cost acquisition model changes the game
- Facilitates a relatively risk-free trial-and-error approach to the marketplace – significant CAPEX commitment not necessarily required to validate an idea
- Published cloud tariffs allows rough cost-estimates to be made

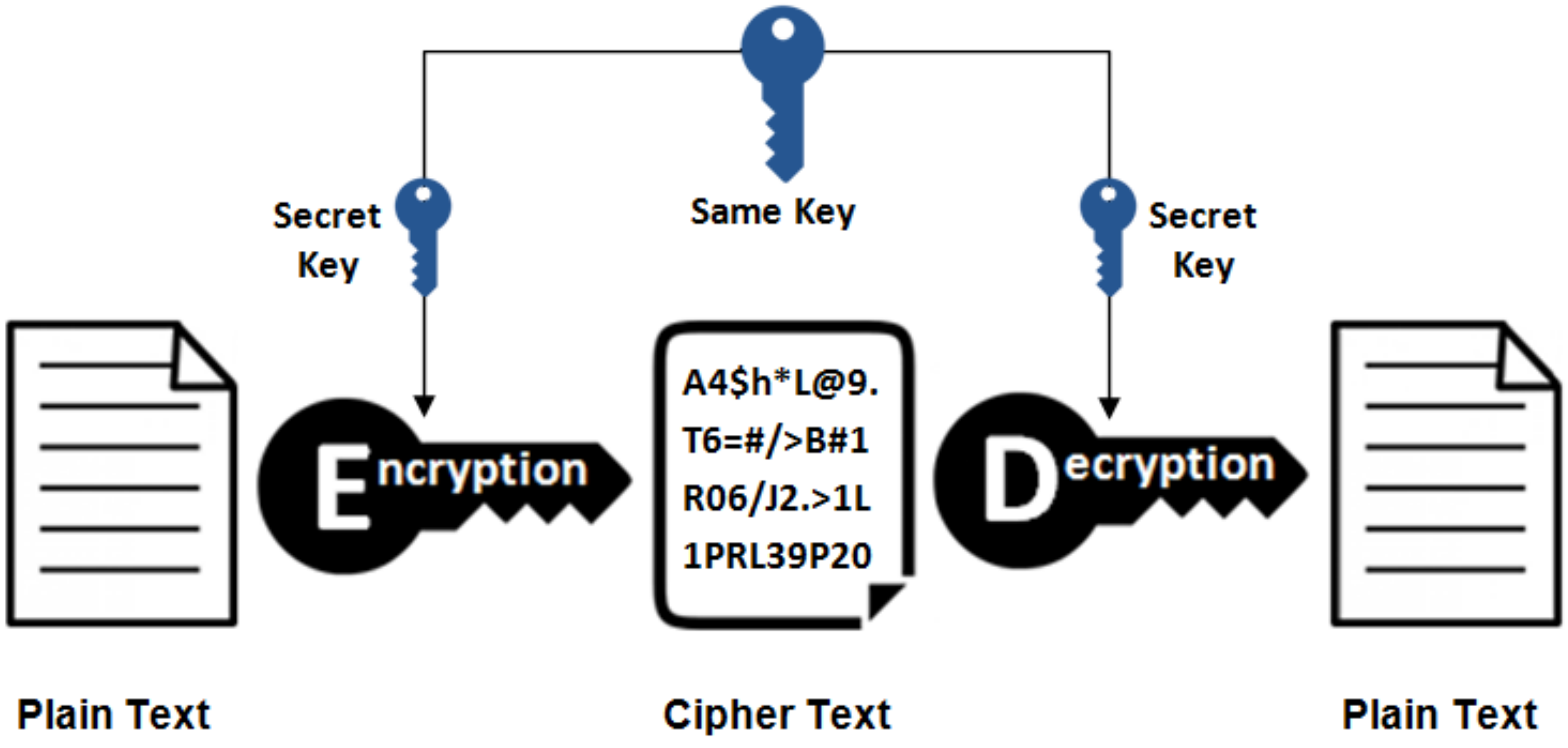
TEXT

TRUSTWORTHY ENCRYPTION SYSTEM
– SECURITY ON THE CLOUD

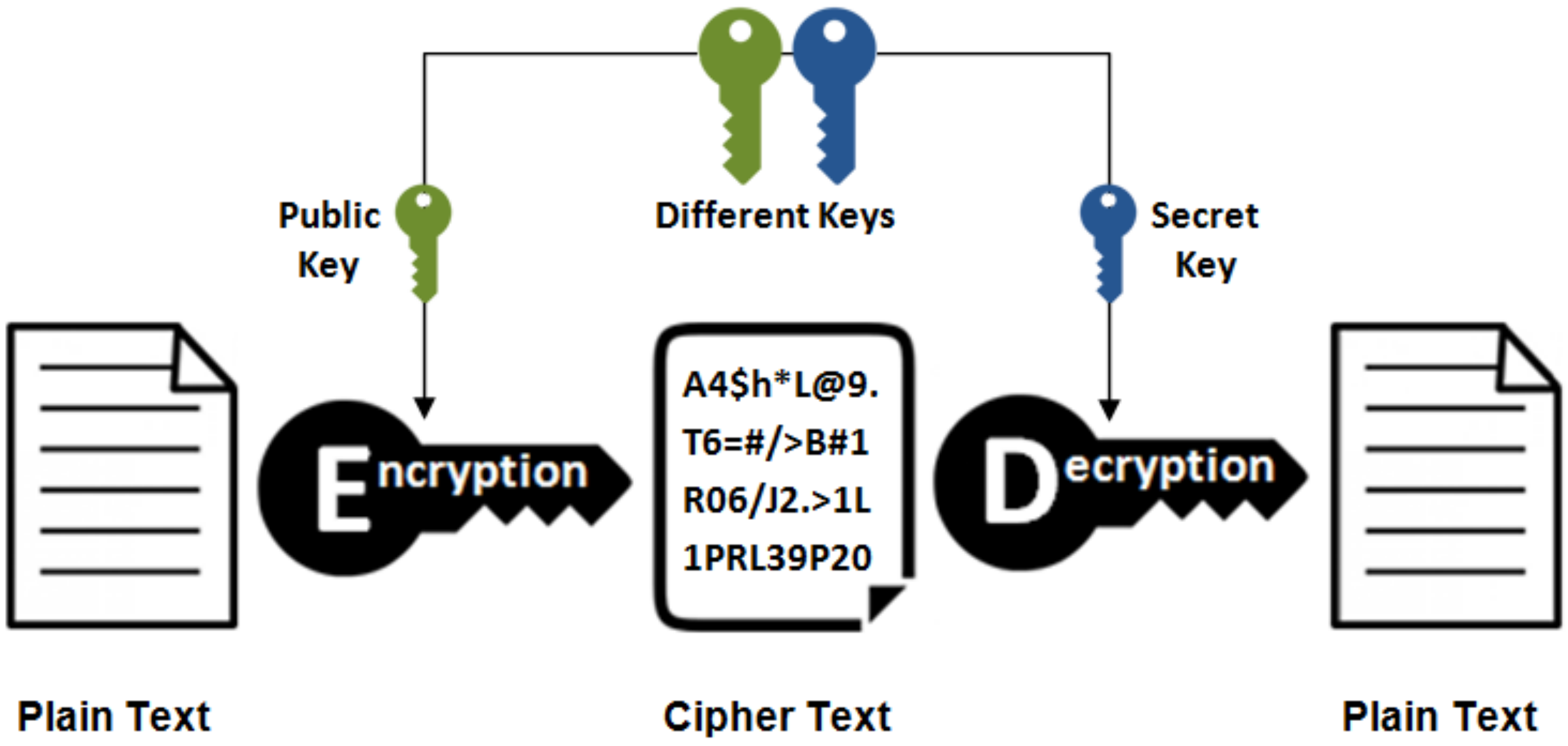
TRUSTWORTHY ENCRYPTION SYSTEM

- It is based on sound mathematics. Good cryptographic algorithms are not just invented, they are derived from solid principles.
- It has been analyzed by competent experts and found to be sound. Even the best cryptographic experts can think of only so many possible attacks, and the developers may become too convinced of the strength of their own algorithm. Thus, a review by critical outsiders is essential.
- It has stood the test of time. As a new algorithm gains popularity, people continue to review both its mathematical foundations and the way it builds upon those foundations. Although a long period of successful use and analysis is not a guarantee of a good algorithm, the flaws in may algorithms are discovered relatively soon after their release.

Symmetric Encryption



Asymmetric Encryption



AWS SQS SERVICE WHY & HOW WE USE IT !

SQS

The Amazon Simple Queuing Service (SQS) which is a distributed web based message delivery system.

The service defines itself as Reliable, Scalable, Simple, Secure and Inexpensive and is one of many similar distributed web messages services such as RabbitMQ.

A public message queue such as the AWS SQS system provides a transactional message system allowing for distributed processes to communicate.

A transactional system ensures that messages can be read, held exclusively by one process and removed from the queue as required. The basic lifecycle of a message is shown in the diagram.

TEXT

RDBMS ACID (ATOMICITY, CONSISTENCY, ISOLATION, DURABILITY)

ACID

ACID compliance my not be offered by a NoSQL DBMS and may need to be implemented at the application level

- Atomicity, Consistency, Isolation, Durability
  - Integrity checking may not exist, requiring application level implementations if needed
  - Security models tend to be less well development compared with relational counterparts
  - No stored procedures, triggers etc
  - Scalability and better performance of NoSQL is achieved by sacrificing ACID compatibility.
  - Atomic, Consistent, Isolated, Durable
  - NoSQL is having BASE compatibility instead.
  - Basically Available, Soft state, Eventual consistency

RDBMS– ACID PROPERTIES

–Atomic – All of the work in a transaction completes (commit) or none of it completes

–Consistent – A transaction transforms the database from one consistent state to another consistent state. Consistency is defined in terms of constraints.

–Isolated – The results of any changes made during a transaction are not visible until the transaction has committed.

–Durable – The results of a committed transaction survive failures

NOSQL BASE

- NOSQL BASE (Basically Available, Soft state, Eventual consistency) Transactions
  - Acronym contrived to be the opposite of ACID
    - Basically Available,
    - Soft state,
    - Eventually Consistent
  - Characteristics
    - Weak consistency – stale data OK
  - Availability first
  - Best effort
  - Approximate answers OK
  - Aggressive (optimistic)
  - Simpler and faster

TEXT

NOSQL DATABASE ARCHITECTURES
WHY WE NEED THEM !
EXAMPLE OF NOSQL DATABASE !
FEATURES OF NOSQL DATABASE

FEATURES OF NOSQL DATABASE

Independent projects designed to address different problems in data persistence

Main drivers include:

Web-scale storage and retrieval

Web application persistence

Data analysis and mining

There is no agreed definition so the term NoSQL can mean:

Definitely not relational (i.e. no SQL)

Not-only-SQL (i.e. heterogenous technologies including SQL)

THREE CLASSIFICATIONS OF NOSQL DATABASES

1. Key-value stores

- Store values and an index to find them, based on a user-defined key
- Extremely simple interface
- Data model: (key, value) pairs
- Operations: Insert(key,value), Fetch(key),
- Update(key), Delete(key)
- Implementation: efficiency, scalability, fault-tolerance
- Records distributed to nodes based on key
- Replication
- Single-record transactions, “eventual consistency

THREE CLASSIFICATIONS OF NOSQL DATABASES

2. Document databases

- Store documents, as just defined. The documents are indexed and a simple query mechanism is provided
- Like Key-Value Stores except value is document
- Data model: (key, document) pairs
- Document: JSON, XML, other semistructured formats
- Basic operations: Insert (key,document), Fetch(key),
- Update(key), Delete(key)
- Also Fetch based on document contents
- Example systems CouchDB, MongoDB, SimpleDB etc

THREE CLASSIFICATIONS OF NOSQL DATABASES

3. Graph databases

- Efficient distributed storage of documents as graph vertices, modeling relationships as edges between them
- Graph Databases are built with nodes, relationships between nodes (edges) and the properties of nodes.
- Nodes represent entities (e.g. "Bob" or "Alice").
- Similar in nature to the objects as in object-oriented programming.
- Properties are pertinent information related to nodes (e. g. age: 18).
- Edges connect nodes to nodes or nodes to properties.
- Represent the relationship between the two.
- Scaling graph DBs is problematic
- Interfaces and query languages vary

PAXOS ALGORITHM AND HOW IT WORK IN DISTRIBUTED COMPUTING

CONSENSUS IS THE PROCESS OF AGREEING ON ONE RESULT AMONG A GROUP OF PARTICIPANTS.

Consensus

PAXOS ALGORITHM

A consensus algorithm where by nodes in a distributed system can agree, by majority, on some value, e.g. who should be the master node

A proposer node (only one permitted at a time) prepares a proposal which it sends to acceptor nodes (there may be many of these)

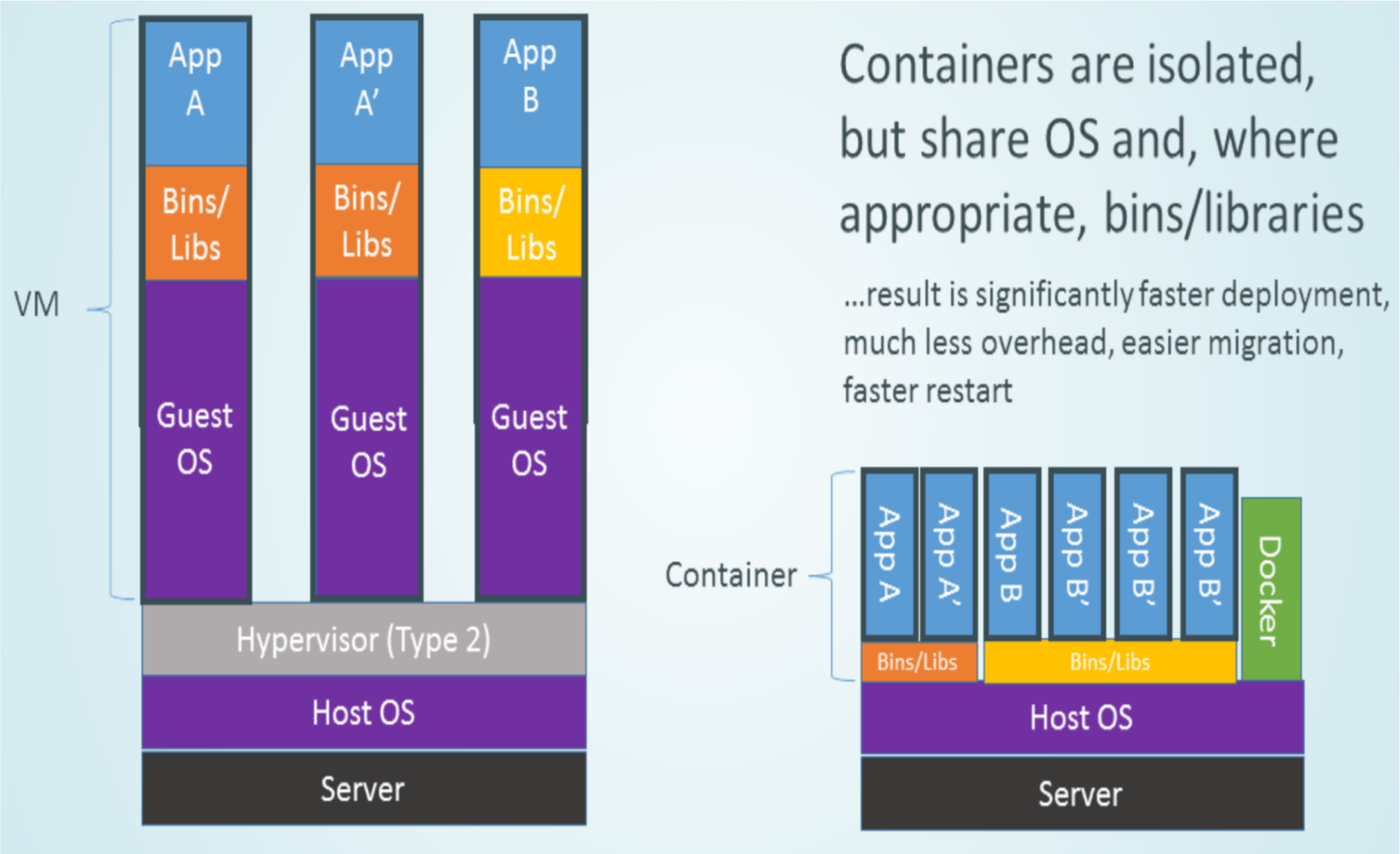
The proposals include a sequence number

Acceptors reply that they will either accept the proposal or reject it, because they have a proposal with a higher sequence number

When rejecting, the proposer is sent and now knows the higher sequence number allowing it re-issue a higher sequenced proposal

DOCKER
DOCKER COMMANDS

DOCKER



Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart

DOCKER

- Docker enables any application and its dependencies to be packaged up as a lightweight, portable, self-sufficient container. Containers have standard operations, thus enabling automation.
- Containers are designed to run on virtually any Linux server. The same container that that a developer builds and tests on a laptop will run at scale, in production, on VMs, bare-metal servers, OpenStack clusters, public instances, or combinations of the above.
- Developers can build their application once, and then know that it can run consistently anywhere. Operators can

DOCKER
DESCRIBE THE FUNCTION OF DOCKER COMMANDS

GRID COMPUTING VS. CLOUD COMPUTING

GRID COMPUTING VS. CLOUD COMPUTING

- Grids enable access to shared computing power and storage capacity from your desktop. Clouds enable access to leased computing power and storage capacity from your desktop
- GRID: Research institutes and universities federate their services around the world through projects such as EGI-InSPIRE and the European Grid Infrastructure
- Cloud: Large individual companies e.g. Amazon and Microsoft and at a smaller scale, institutes and organisations deploying open source software such as Open Slate, Eucalyptus and Open Nebula.
- GRID:Research collaborations, called "Virtual Organisations", which bring together researchers around the world working in the same field.
- Cloud: Commercial businesses or researchers with generic IT needs

GRID COMPUTING VS. CLOUD COMPUTING

GRID: Governments - providers and users are usually publicly funded research organisations, for example through National Grid Initiatives.

Cloud: The cloud provider pays for the computing resources; the user pays to use them

GRID: In computing centres distributed across different sites, countries and continents.

Cloud: The cloud providers private data centres which are often centralised in a few locations with excellent network connections and cheap electrical power.

GRID: You don`t need to buy or maintain your own large computer centre. You can complete more work more quickly and tackle more difficult problems. You can share data with your distributed team in a secure way..

Cloud: You don`t need to buy or maintain your own personal computer centre. You can quickly access extra resources during peak work periods

TEXT

EVOLUTION OF COMPUTER ARCHITECTURES

EVOLUTION OF COMPUTER ARCHITECTURES

Central Processing using Thin Clients (Mainframes)

Approach

Mainframe provides all resources and logic, Terminals for user interaction (thin clients)

Advantages

Simple deployment,

Problems and Limitations

Limited GUI capabilities, Every client allocates server resources , Limited scalability

Personal Applications using PCs

Client-Servers Architectures (Fat Clients)

Approach

Increased capabilities of PCs, Logic was (partially) transferred to clients thick clients, Central database server

Advantages

Better user experience (GUIs), Problems and Limitations, Every client has a permanent stateful connection, Every client holds server resources (db connections, …) , Limited scalability

EVOLUTION OF COMPUTER ARCHITECTURES

Middle Tier Architectures (Presentation/Processing/Data Store)

- Approach
- Logic bundled in Middle-tier
- Clients have no permanent connection to Middle-tier stateless connections

Advantages

1. Factoring of business logic
2. Middleware provides services (pooling, security, …)
3. Resources are shared between clients improved scalability
4. Problems
5. New programming model
6. Limited scalability

RESTFUL ARCHITECTURE AND THEIR CONSTRAINTS

RESTFUL ARCHITECTURE

It is a software architectural style for developing web services consisting of guidelines for web service which are scalable. REST web services communicate over the HTTP specification, using HTTP vocabulary:

Representational: Clients possess the information necessary to identify, modify, and/or delete a web resource.

State: All resource state information is stored on the client.

Transfer: Client state is passed from the client to the service through HTTP

RESTFUL ARCHITECTURE CONSTRAINTS

1. Uniform interface
2. Decoupled client-server interaction
    - Client-server: Clients are separated from servers by a uniform interface.
    - For portability, clients must not concern themselves with data storage.
    - For simplicity and scalability, servers must not concern themselves with the UI or user state.
    - Servers and clients may be replaced and developed independently, as long as the interface is not altered.
3. Stateless
    - No client context should be store on the server between requests.
    - Each request contains all of the information necessary to service the request and session state is held in the client.
    - The server can be stateful, but server-side state must be addressable by URL as a resource.
    - This makes servers: More scalable, More visible for monitoring. More reliable in the event of partial network failures

RESTFUL ARCHITECTURE CONSTRAINTS

4. Cacheable

    - Clients may cache responses, so responses must define whether they are cachable to prevent clients reusing stale or inappropriate data in response to further requests.Well-managed caching can eliminate repetitive client–server interactions, further improving scalability and performance.

5. Layered

    - A client's connection to a server may pass directly to the service or through several intermediaries, allowing:
    - Scalability by enabling load balancing and by providing shared caches
    - The enforcement of security policies.

6. Extensible through code on demand

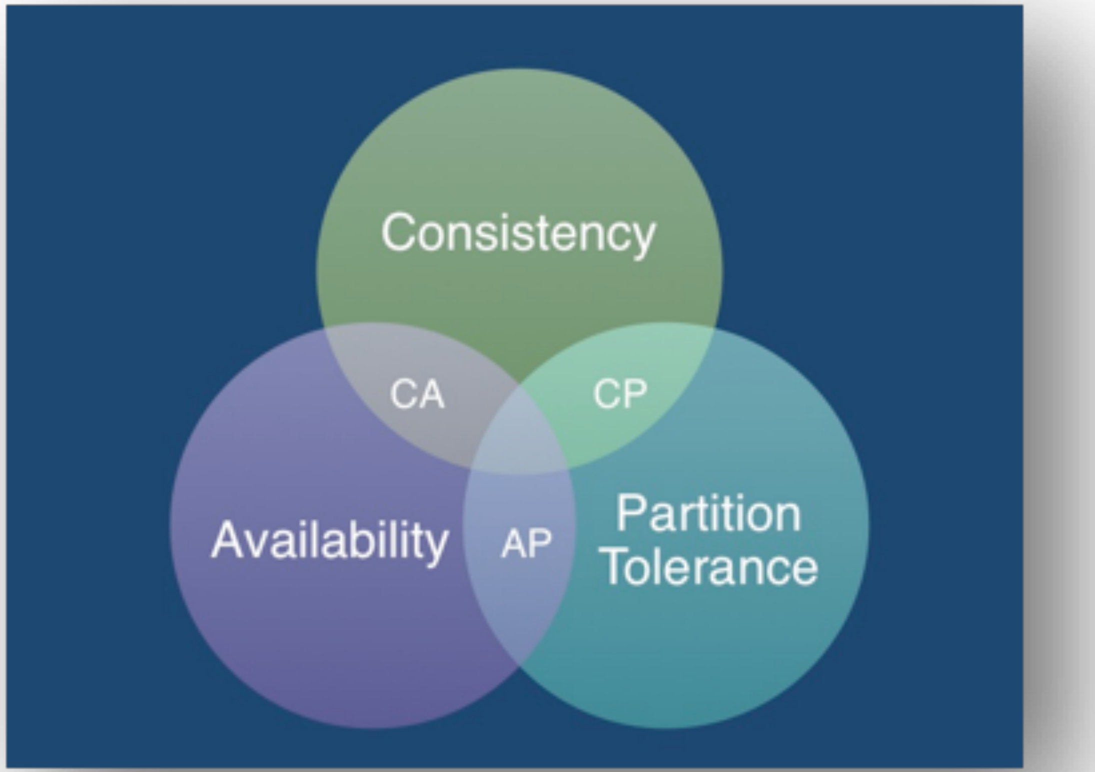
- Servers may temporarily extend or customise the functionality of a client by transferring logic to be executed. (e.g. client-side JavaScript)

TEXT

CAP THEOREM

CAP THEOREM

- It is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:
  - Consistency (all nodes see the same data at the same time)
  - Availability (a guarantee that every request receives a response about whether it was successful or failed)
  - Partition tolerance (the system continues to operate despite arbitrary message loss or failure of part of the system)
- A distributed system can satisfy any two of these guarantees at the same time, but not all three.



TEXT

SECURITY IMPLICATIONS OF CLOUD COMPUTING.

SECURITY IMPLICATIONS OF CLOUD COMPUTING

- **Lack of physical custody, Loss of Control**
    - As with many out-sourced solutions
    - Applications and data are hosted off-premises
    - Vendor has access to and is “responsible” for customer data but usually won’t take responsibility in their SLAs
    - Data may need to be encrypted “at rest”
    - Data and apps may still need to be on the cloud but diversify
    - Monitor systems and data
    - Focus on Access Control management
- **Subject to generic security settings offered by service vendor**
    - Chosen by vendor for vendor’s reasons
    - Vendor has to strike a balance between security and convenience for its customers
    - That balance, by definition, cannot suit all customers for example AWS Keys…

SECURITY IMPLICATIONS OF CLOUD COMPUTING

- **Lack of transparency** - the paradox that convenience and abstraction of details make clouds harder to understand and reason about, especially from a security perspective The convenience of on-demand, low-friction provisioning and payment for cloud services is offset by the lack of visibility into how applications and data are being managed by the service vendor
- **Lack of Trust?**
  - Understand the Technology used!
  - Perform Risk Assessment on service and consider risk/benefit
  - Understand what is regulated
- **Vendors unlikely to disclose the details of their security architecture**
  - **for security reasons**
  - **for reasons of intellectual property protection**
  - Means customers cannot do proper due diligence of vendors' solutions or carry out periodic audits of their systems
- **Data needs to be copied over insecure media**
  - From enterprise originators (the cloud service customers) to and from the cloud
  - From the enterprise's customers to and from the cloud
  - Within the cloud to provide geographical redundancy if necessary
  - Therefore may need to be encrypted "in motion

TEXT

THANK YOU !

Dr. Basel Magableh