greatlearning
Learning for Life

PGP in Cloud Computing

Managed Services on AWS

IaaS - key aspects

- Enterprise infrastructure
- Cloud hosting
- Virtual Data Centers (VDC)
- Scalability
- No investment in hardware
- Utility style costing
- Location independence
- Physical security of data center locations
- No single point of failure

PaaS - key aspects

greatlearning
Learning for Life

- Managed services, foundation, APIs
- Managed runtime, no need to start, stop
- Managed infrastructure, agnostic on the nature of hardware the application runs on, though the user can get some control of picking the type of machine
- Extensive monitoring & Dashboards, allows for monitoring and also identifying application bottlenecks and helps in debugging
- Auto scalability, can specify some limits to protect from malitious access
- Billing thresholds to keep a tab on the expenditure
- Alerts, always good to get informed "before" an outage
- Choice of programming language, customers have the ability to create teams as per organizational strengths

SaaS - key aspects

greatlearning
Learning for Life

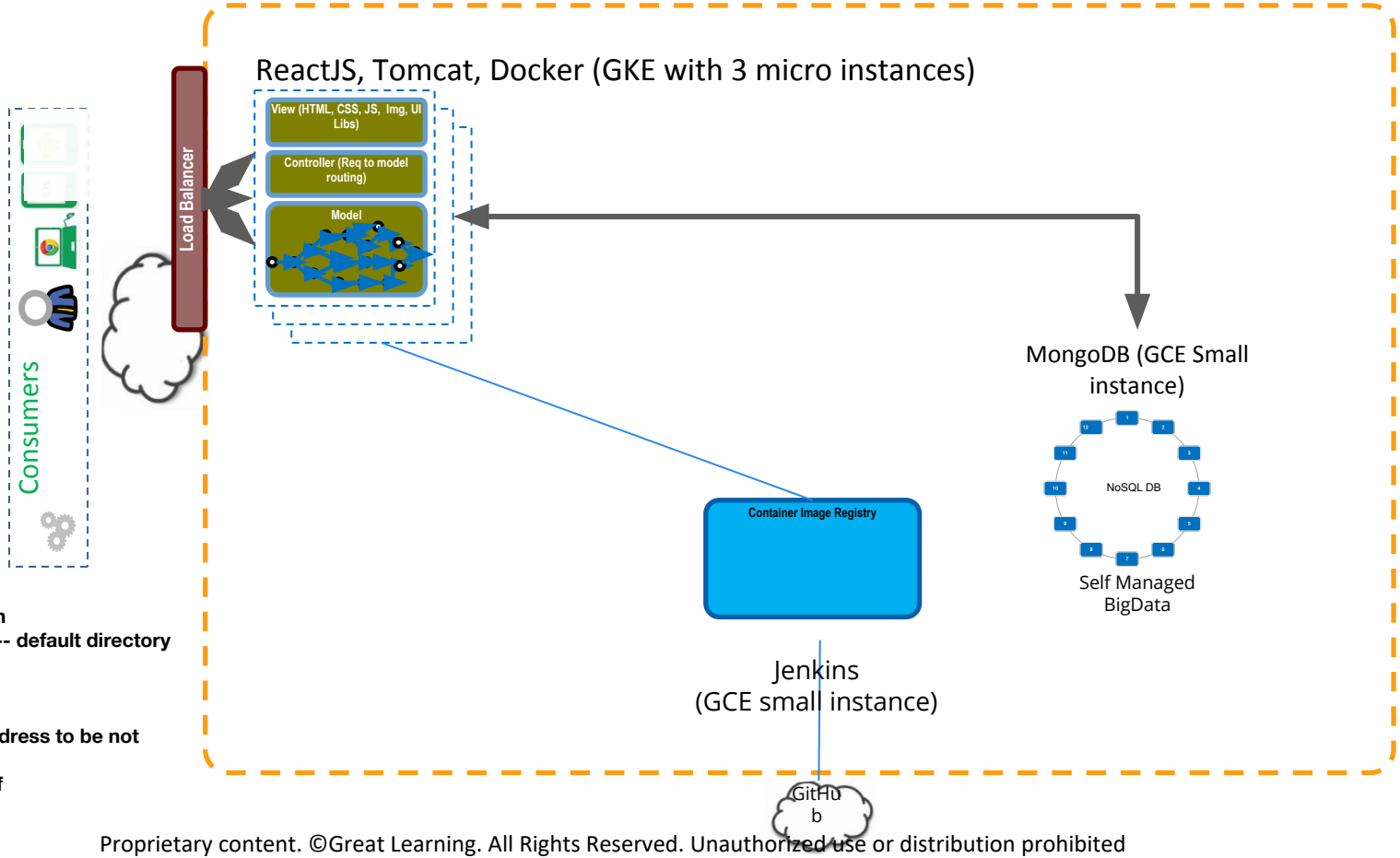
- Allows for hosting and management of applications to a 3rd party provider
- Usually a complete business process that can be used in isolation or in conjuntion with another business process to create composite macro business processes
- 3rd party provider is responsible for delivery SLA, L1 to L4 support etc
- Does not require any installation at the client location on any machine on any device
- Pricing can be per seat or fixed price depending on the provider
- The provider typically has a multitenant architecture for one instance to many client delivery models. This allows for economy of scale and reduces cost of usage
- These are usually online applications, however there can be offline applications too using contenporary browser capabilities

Cloud impact

What cloud features we can introduce in the mix to get take the advantages we have been discussing about?

What does the new architecture look like?

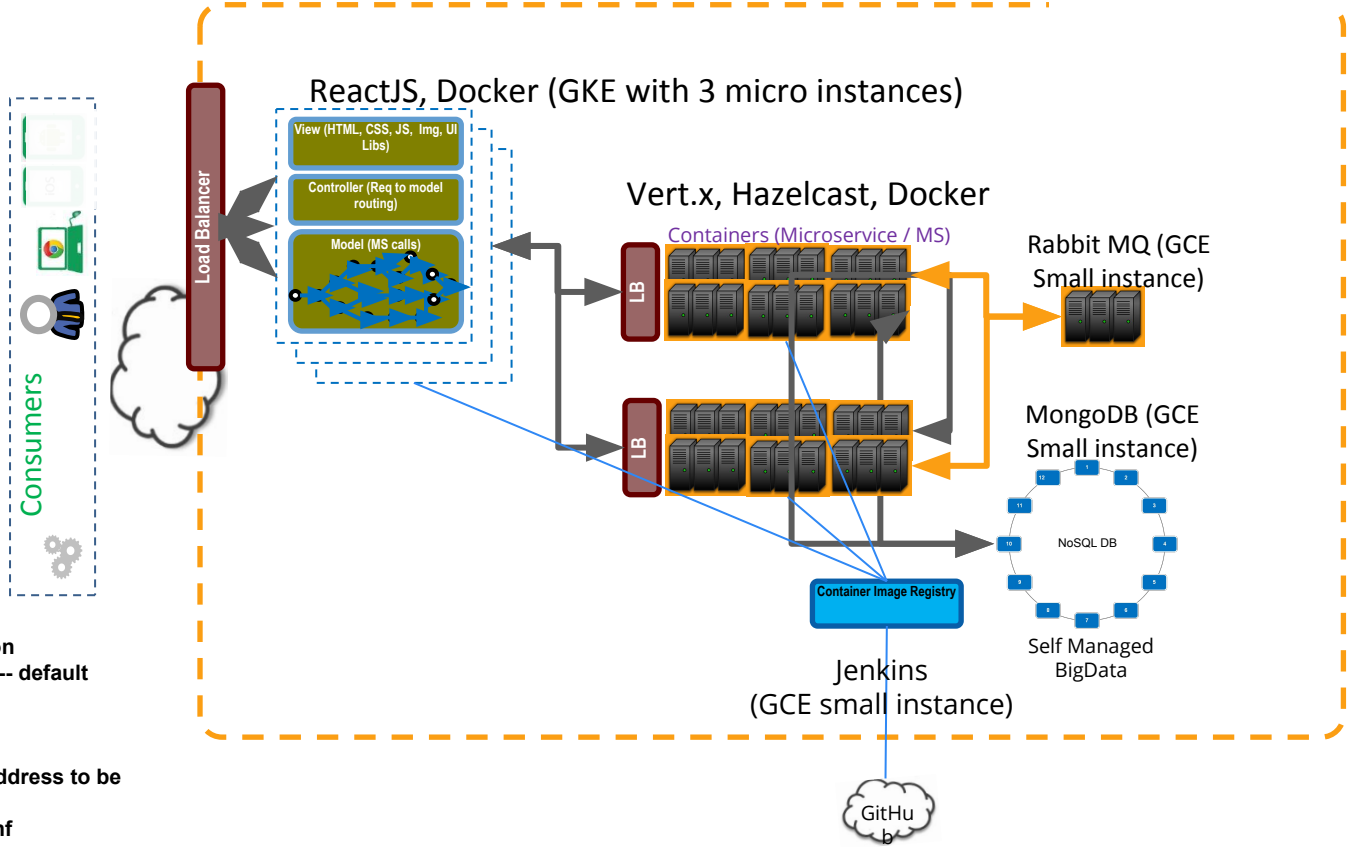
CMAD - Essentials



//Mongo DB configuration
//Create data directory --- default directory being /data/db
sudo mkdir -p /data/db

//configure the bind ip address to be not 127.0.0.1
sudo vi /etc/mongod.conf
//modify the bind ip with
bindIp: 0.0.0.0

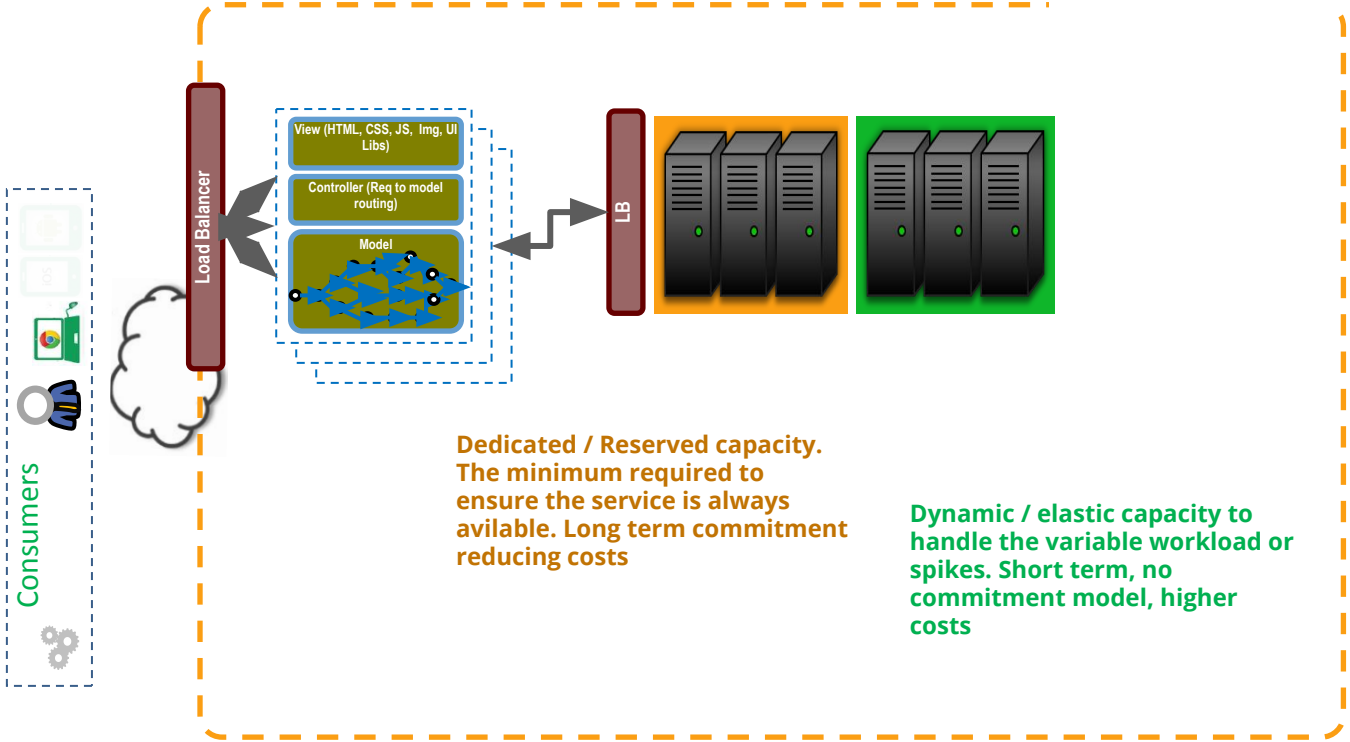
CMAD - Advanced



//Mongo DB configuration
//Create data directory --- default directory being /data/db
sudo mkdir -p /data/db

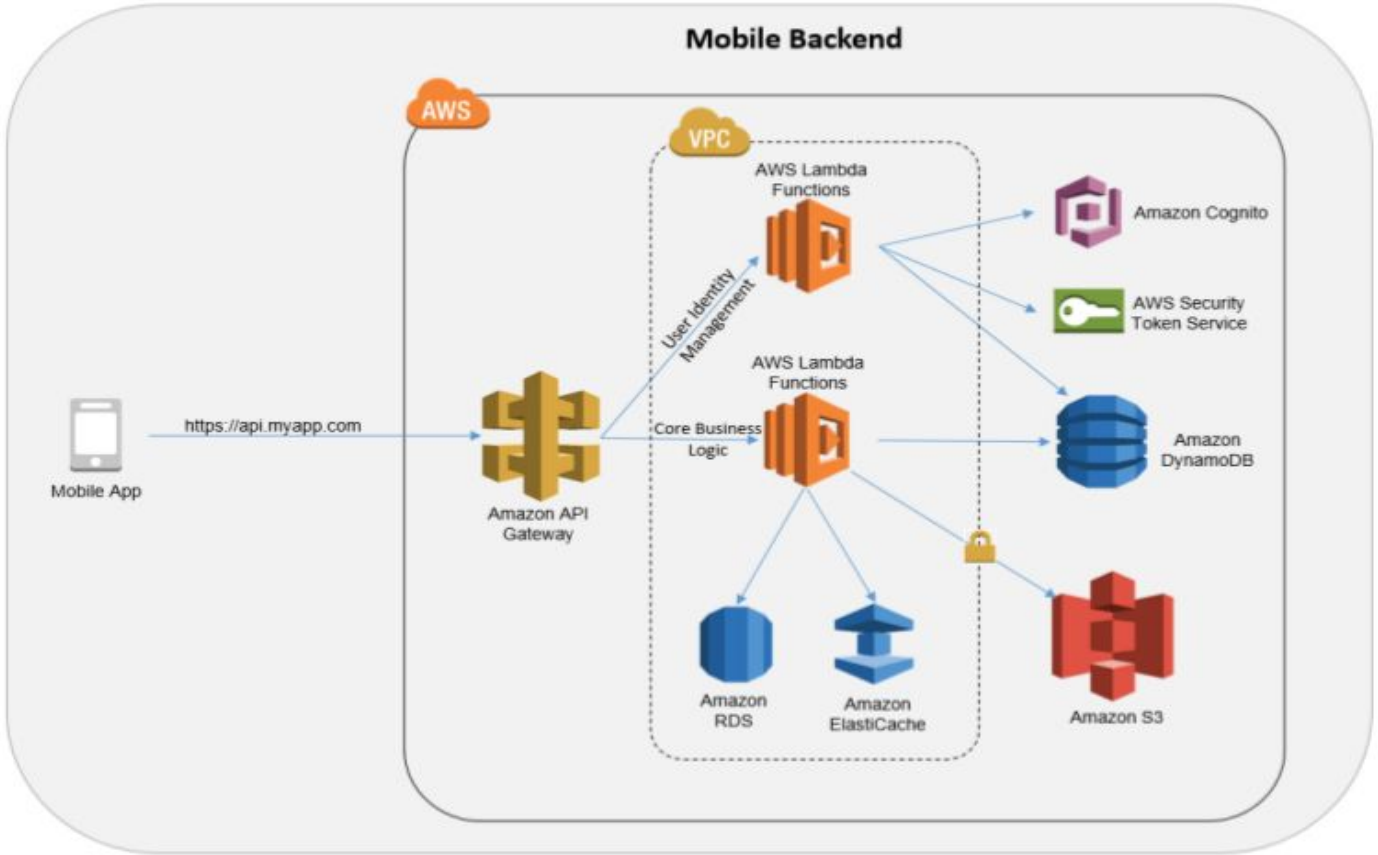
//configure the bind ip address to be not 127.0.0.1
sudo vi /etc/mongod.conf
//modify the bind ip with
bindIp: 0.0.0.0

Factor in cost

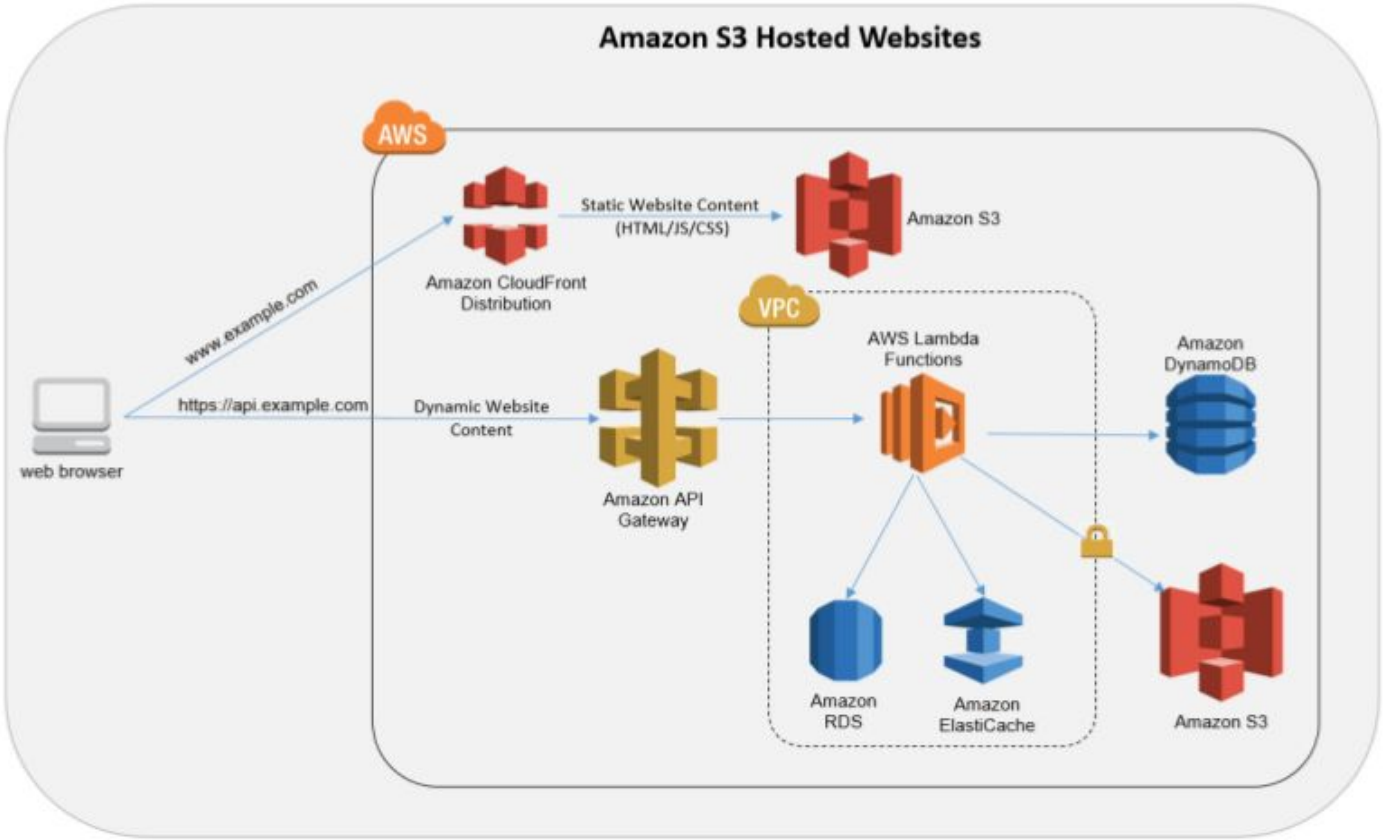


Split your infrastructure in "Reserved" and "Dynamic" buckets to control costs. More reserved instances mean lower costs but lower utilization if not planned properly! Weigh your options and understand your workload well.

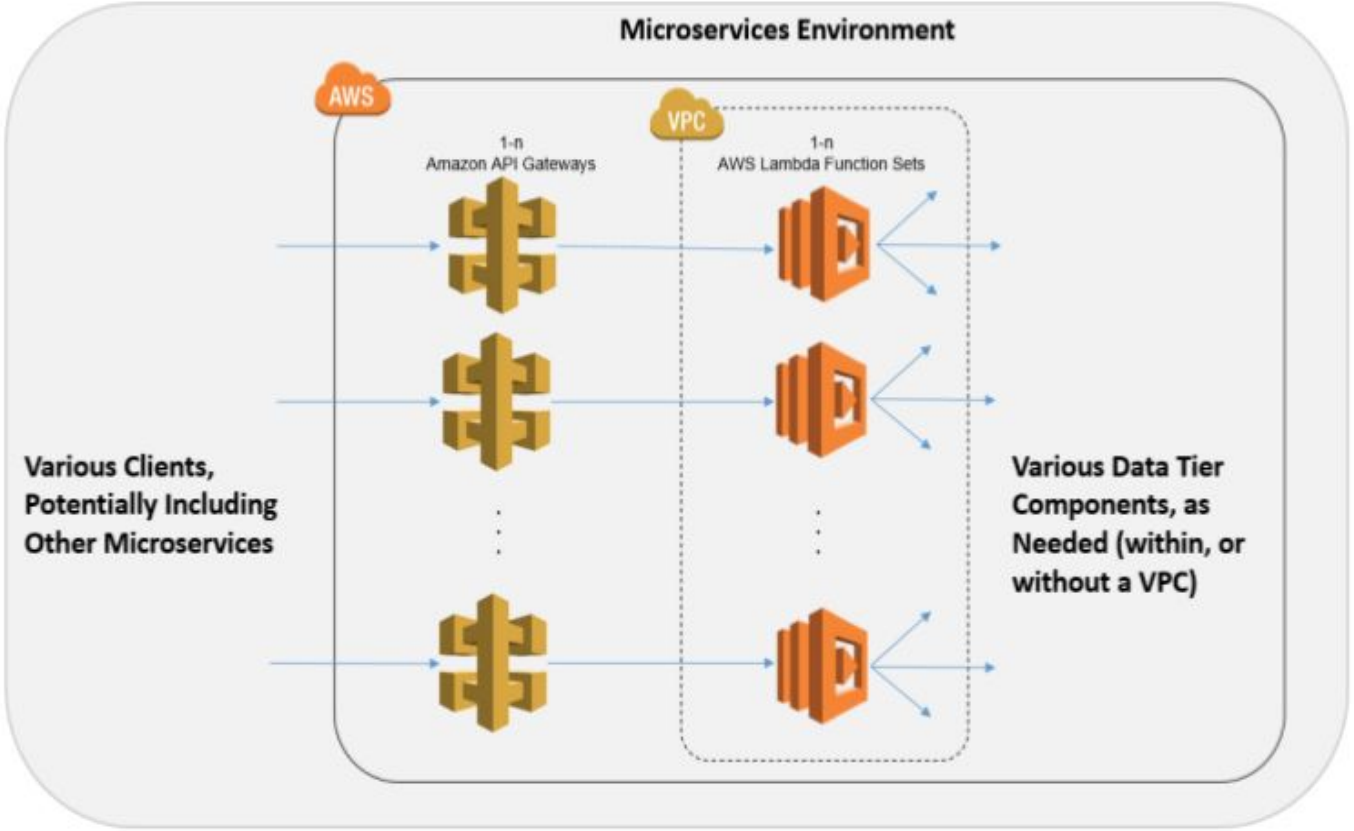
Cloud architecture - AWS example



Cloud architecture - AWS example

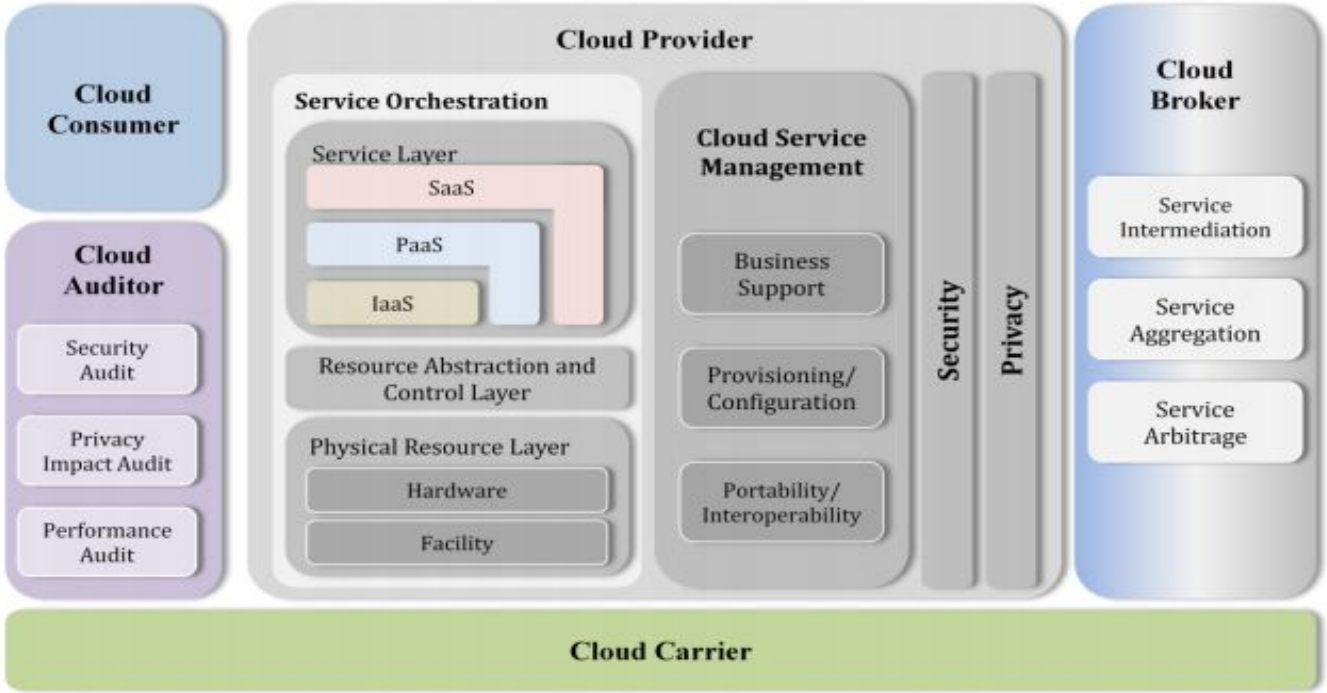


Cloud architecture - AWS example



Reference architecture

Figure 1 presents an overview of the NIST cloud computing reference architecture, which identifies the major actors, their activities and functions in cloud computing. The diagram depicts a generic high-level architecture and is intended to facilitate the understanding of the requirements, uses, characteristics and standards of cloud computing.



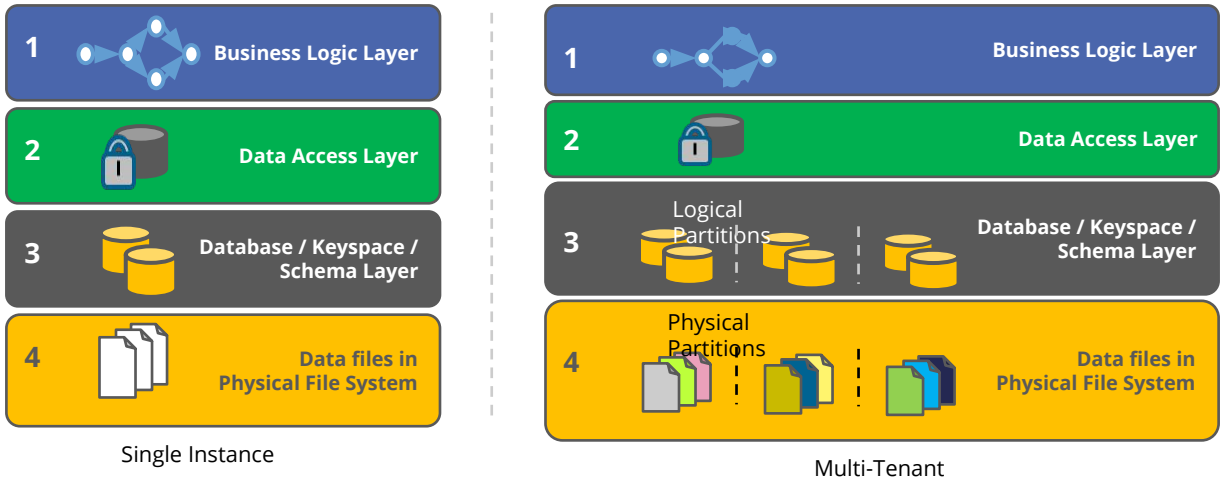
greatlearning
Learning for Life

It's all about data

Remember multi tenancy? How can we achieve that on the cloud? In addition what about modeling?

Data architecture

- Applications on the cloud can be classified as single tenant or multi tenant. While single tenant applications have a natural partition of data from one customer to another; multi tenant applications require a deeper architectural consideration



- Applications that consume data from external sources need to consider appropriate security and customer partitioning handshake protocols.
- The application on the cloud needs to consider various country specific regulations that restrict data movements beyond the physical boundaries of that country. This is an important aspect to consider for applications which may have an impact to the deployment model or even the overall ROI.

Data storage compliance

- Driven by compliance regulations such as
  - HIPAA in USA
  - Data Protection Directive in EU
- Data sensitivity - Financial / Medical
- Driven by a sense of "Complete control" as realized by on-premise
- Legal issues with data residency depending on where the data is physically stored off-premise
- Data archival policy impact

Denormalization

- Unlike RDBMS you cannot use any foreign key relationships
- FK and joins are not supported anyway!
- Embed details, it will cause duplication but that is alright
- Helps in getting the complete data in a single read
- More efficient, less random I/O
- May have to create more than one denormalized view to serve different queries

Sharding

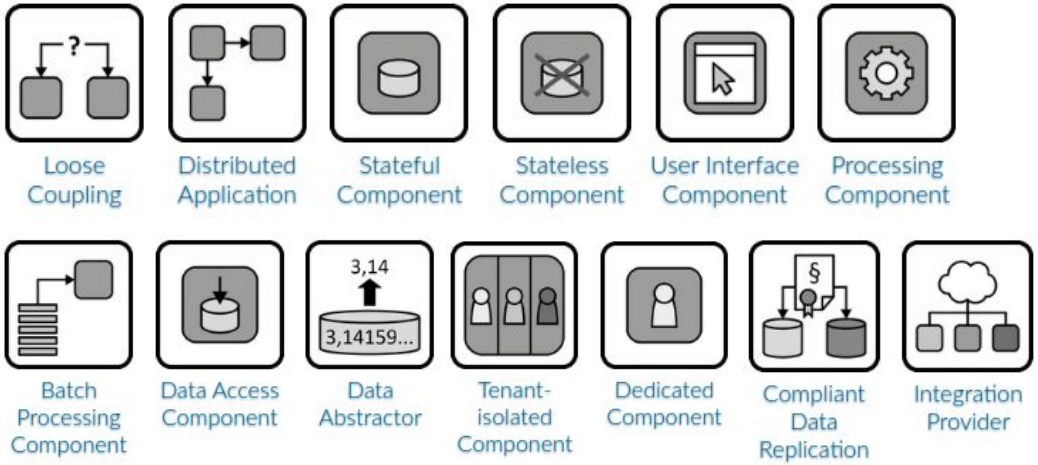
- Horizontal partitioning splits one or more tables by row
- Splits large partitionable tables across the servers
- Smaller tables are replicated as complete units to allow joining with the sharded large table
- This is also why sharding is related to a shared nothing architecture—once sharded, each shard can live in a totally separate logical schema instance / physical database server

greatlearning
Learning for Life

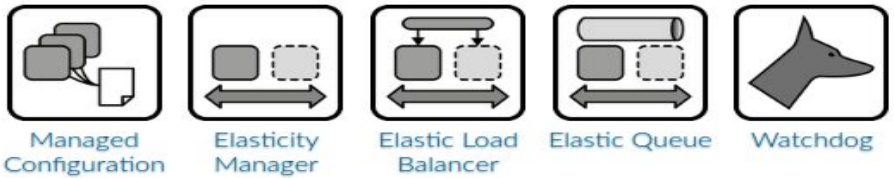
Architectural styles

Cloud architectural styles & components

- Foundation, common service layer & these patterns



- Define application zones & scalable resilient deployments



- Serverless, CQRS, Event sourcing patterns
- Some other techniques such as vector clocks, mutex models needed in distributed computing

Stateless / singleton design

greatlearning
Learning for Life

- What is stateless design?
- Is too much stateless good?
- What's the optimum balance?
  - Split between framework components and business components
  - Biz components are BO and Helpers
  - Data access layer
- File system
  - Using local file system for storage. Really?
- Http Session
  - In distributed systems where to create the session?
  - Session stickiness and impact

Thinking adapters

greatlearning
Learning for Life

- Applications in the cloud can potentially leverage services/APIs from multiple providers
- Each provider will have their own
  - release cycles
  - API service versioning
  - feature enhancements
  - payload data structure upgrades
- How do we protect our application from such changes?
- Abstraction techniques - data format transformations
- Caching
- API calling techniques - Web services or REST?

Serverless approach

greatlearning
Learning for Life

- Running code in the cloud without provisioning or managing any server
- Automatically scales application based on the trigger/call to the application by the client
- No need to pay anything when code is not running
- Examples
  - AWS Lambda
  - Google cloud functions

Business service layer pattern

greatlearning
Learning for Life

- Can be looked at as an orchestration layer = Biz Process layer
- Each step of the orchestration can be easily powered by Microservices
- Access
  - Consider using REST / URL APIs to access business features from outside
  - Within an application consider using direct API calls and object exchange
- Service versioning is critical
- Data exchange
  - JSON (compact)
  - XML (verbose)
- This allows for the capability of "Integration ready" where other applications can leverage various features
  - Other applications can be inside OR outside the enterprise
  - Since the services are atomic the usage can be monitored very closely
  - Usage monitoring can lead to very interesting and flexible pricing models in case these are exposed to outside the enterprise
  - Example http://api.acme.com/sc/customer/update/123

Define application zones

greatlearning
Learning for Life

- Various components of the application tiers require different attention
  - Choice of programming language
  - Choice of hardware
  - Specialized libraries
  - Scaling requirements
  - Backup and recovery mechanisms
  - Monitoring requirements

- Can "One shoe fit all?" How do we ensure "Availability of customer facing services?"

- Application zones are an effective way to isolate different areas of the application and provide an environment that is optimized and tuned for that function
  - Distributed data layer
  - Microservice layer
  - Analytics layer
  - Online web application layer

Debate - Mobile apps!

- "Download our mobile app and your next ride will be Rs 100 only"
- "Buy using our mobile app and you will get an additional 10% off"
- "This discount coupon is available on purchases from our mobile app only"

Q: What's going on? What are your thoughts?

Assemble UI on devices

- There are two models of rendering the user interface
  - Assemble on server side
  - Assemble on user's device
- Server side
  - Fully created HTML is served
  - Server needs to be aware of user's device
  - Server utilization is more
  - Not possible to do responsive design
  - Page refresh may be time consuming
- User's device
  - Server serves the page skeleton
  - UI calls various services and builds the UI depending on the device characteristics
  - Server utilization is low
  - Responsive design
  - Partial refresh creates engaging UI and gives a sense of speed
- Assembly on user's device should be the choice because
  - Reduces OPEX
  - Leverage the capabilities of very powerful handheld devices

Native device applications or ...

greatlearning
Learning for Life

- There are two kinds of applications that can run on a mobile device (iPads included)
  - Native applications
  - Browser based applications
- Let's look at the Native application properties ...

- Is a different app for each type of device - iOS, Android, WinOS etc
- Since these are installed on the device, they have access to the device a lot more (phone book, camera, local disk etc)
- Requires specialized development skills
- Each app is like a project that needs to exist and get development by a team
- Hard to synchronize app features across the board
- Push updates are required to ensure latest version is available
- Can work in offline mode if designed
- Degree of customization and application features are limited to whatever is supported by the device OS

... browser based?

greatlearning
Learning for Life

- Let's look at the browser based app properties ...

- As long as there is a compatible browser with a JS engine, the site will run
- No installation required but may not have access to device features (address book etc)
- It is a web applications and the skills are confined to web UI development
- There is one web app that renders on a multitude of devices
- No feature synchronization issues
- If the site is updated then all devices will serve the latest
- Cannot work offline very effectively (though there are offline embedded DB in JS)
- Degree of customization is limited to the browser and JS engine
- Regardless of the type chosen the server side services should remain the same