怎么样访问某一个全球网站：

DNS (ip)->

根据不同网络 DNS return当地 server 的ip 网站

网站在当地网络部署自己的服务

CDN

loading an page from the closet site(i.e. CDN site/point) instead of original site(source site/ origin site)

静态网站(static content)从CDN获取，动态内容（dyanamic content）只能在source site 网站获取。 动态内容在Geo-replication 同步更新database。

what questions do DevOps ask in frontend?

* how do we deploy the frontend code without customers notice?
* how do we rollout features safely to our customers?
* how to version control / rollout changes?
* what are the key metrics for measuring the frontend performance and reliabilities?
  + User flow
* what tools do you know to monitor/ logging the fronted events?
* what methods can you think of to monitor the frontend?
* what types of tests or monitoring can we do?

Backend

Why do we need a backend?

* Handle request from different users in parallel
* Make users access to the core data more secure e.g. Jira, Gmail
* Better performance and lower e.g. Uber Map

How to handle request form different users in parallel?

Load balance (LB)

LB rules

* Round robbin
* Health statues
* Pressure
* Traffic
* Latency

Use for multi-region

A historical transition

* Monolith
  + 需要一个功能则需要一整个app
* Microservices
  + 每个功能用一个软件
  + 功能简单，一个server承载一个功能。针对一个功能可能需要三四个server
* Scaling
  + Scale up/ scale down
    - Upgrade more hardware in the server.
  + Scale out / scale in
    - Add more units
    - 放置更多的服务器
    - 搭配LB
  + Monolith 可以使用scaling，但做完发现有大量资源浪费
    - 资源并没有百分百被利用起来
* 计算模型

|  |
| --- |
|  |
| Multiple Vms  (去把physical server没有办法完全利用的尽可能利用起来) |
| Physical resources |

Physical server

* + Vm对应一个软件（尽可能把physical server尽量百分百利用起来）
  + 每个vm必须得要有一个os+software。N个vm会有n+1 OS（including one OS in physical server）
  + 对于100% CPU利用率，N+1 OS已经沾满了60%。留给software剩下40%造成资源浪费
* Container
  + OS asnostic
  + Still can login to the instatnces
  + Need to monitor CPU, RAM, NETWORK.
  + Need to think about scaling
* Lambda

Message queue

Message Queue allow different parts of a system to communicate and process operations asynchronously. A message queue provides a light weight buffer which temporarily stores messages, and endpoints that allow software components to connect to the queue in order to send and receive messages.

Database:

Types of database:

1. Table database e.g. azure table storage; excel; google sheets
2. Relational database management systems e.g. mySQL, PostgreSQL
3. NoSQL and object-oriented databases e.g. dynamosDB, mongoDB
4. Time series databases e.g. influx DB;

**数据类型：格式要求，结构型、半结构型、 关系型数据、**

Cache memory:

for increased availability, increased performance (e.g.减少database的请求次数)

used for: scheduled/triggered task

Depends on what you choose:

* ﻿﻿Memcached is a high-performance distributed memory cache service
* Redis is an open-source key-value store.

Authorisation vs authentication

Authorisation授权: who you are

Authentication验证: what you are authorised to do?