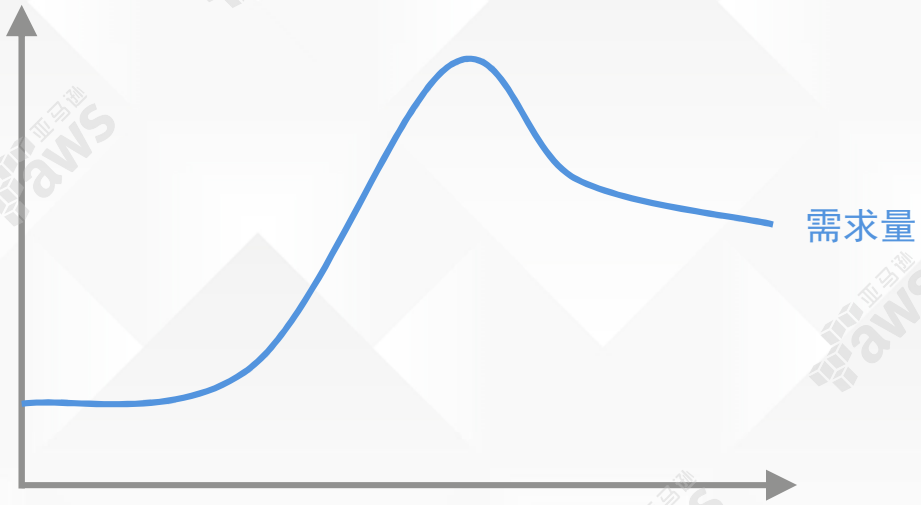

AWS Summit
AWS技术峰会2015 · 上海
亚马逊
aws

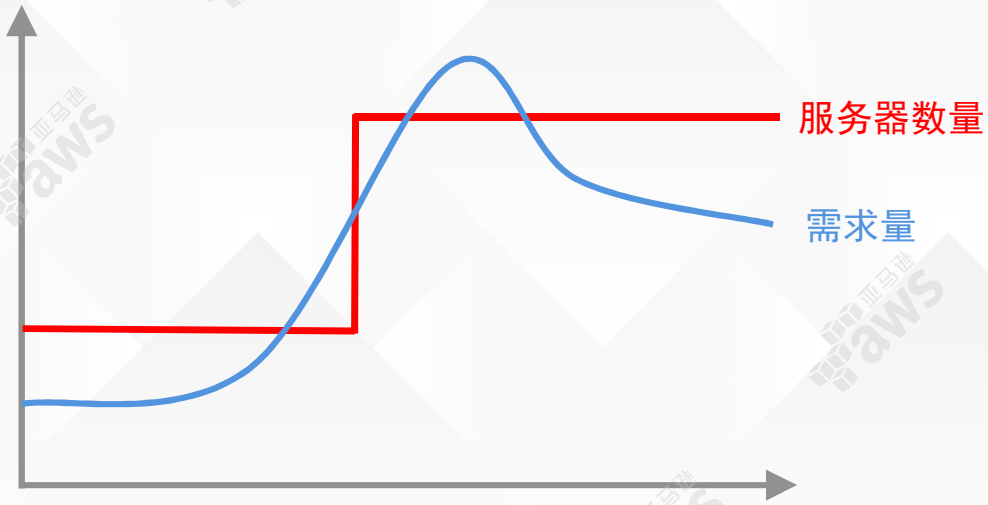
游戏行业解决方案

姜可舒

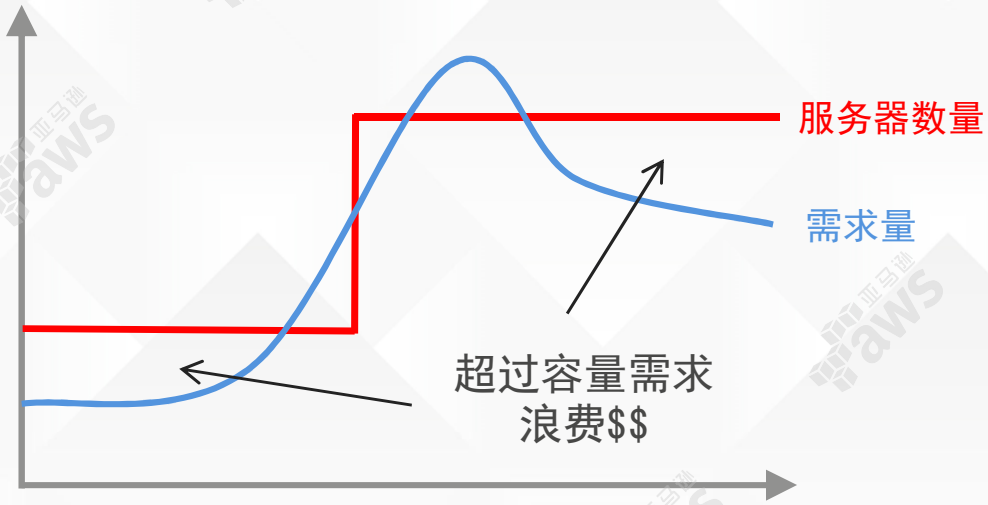
按需扩展，按用量付费



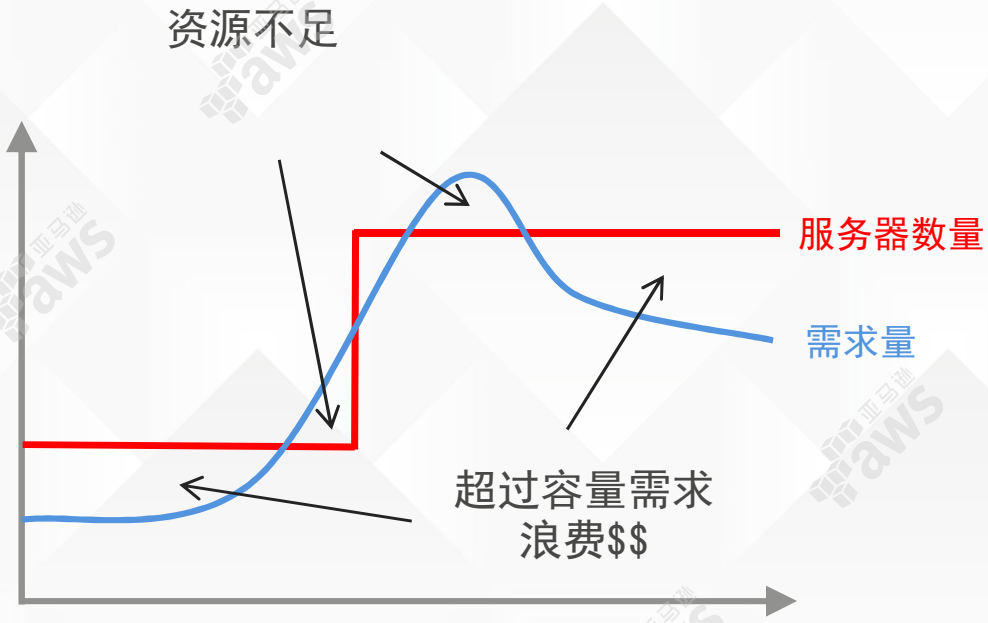
按需扩展，按用量付费



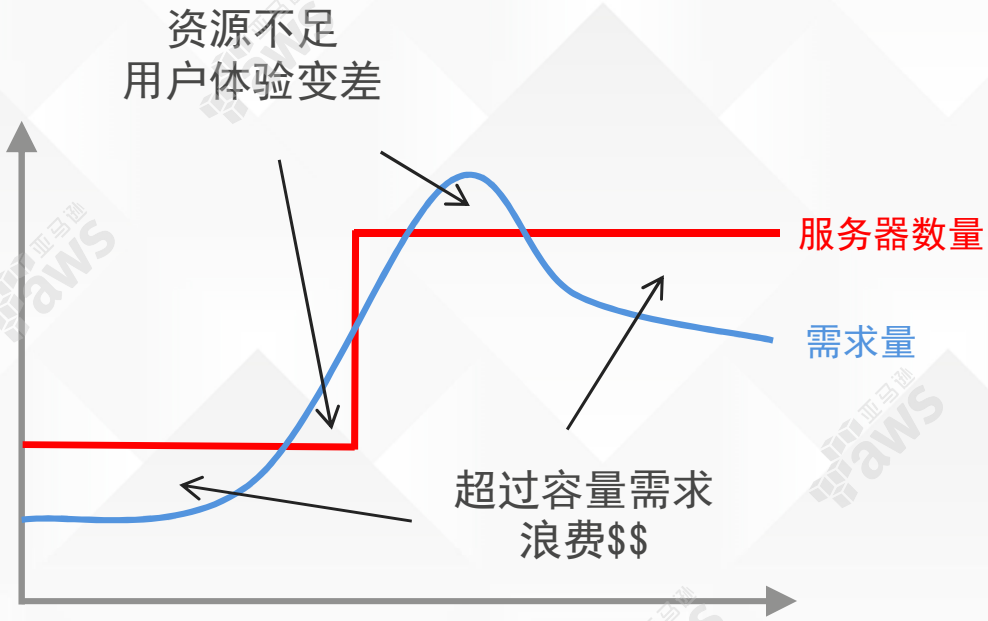
按需扩展，按用量付费



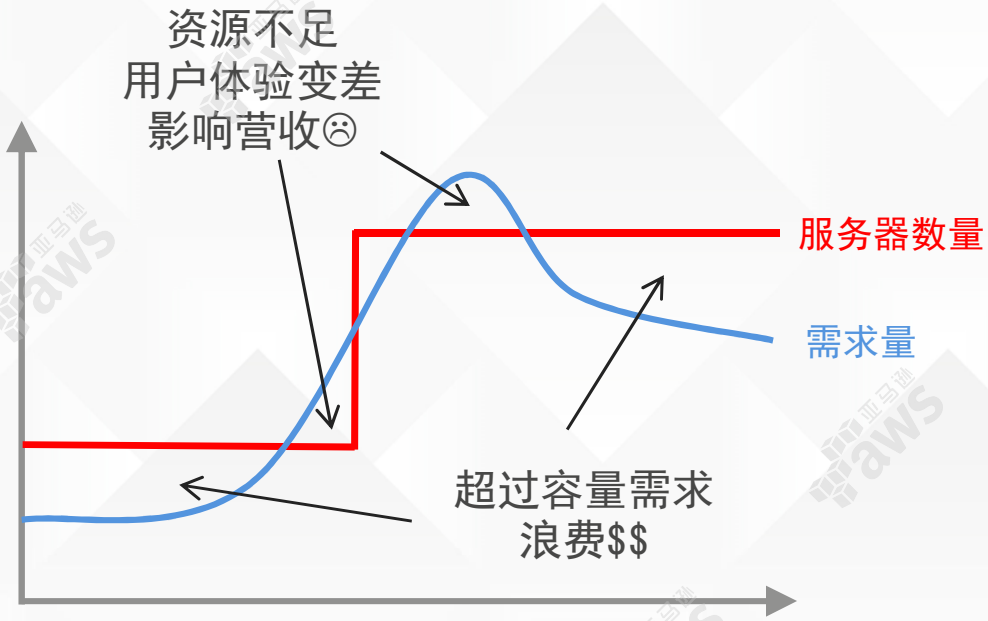
按需扩展，按用量付费



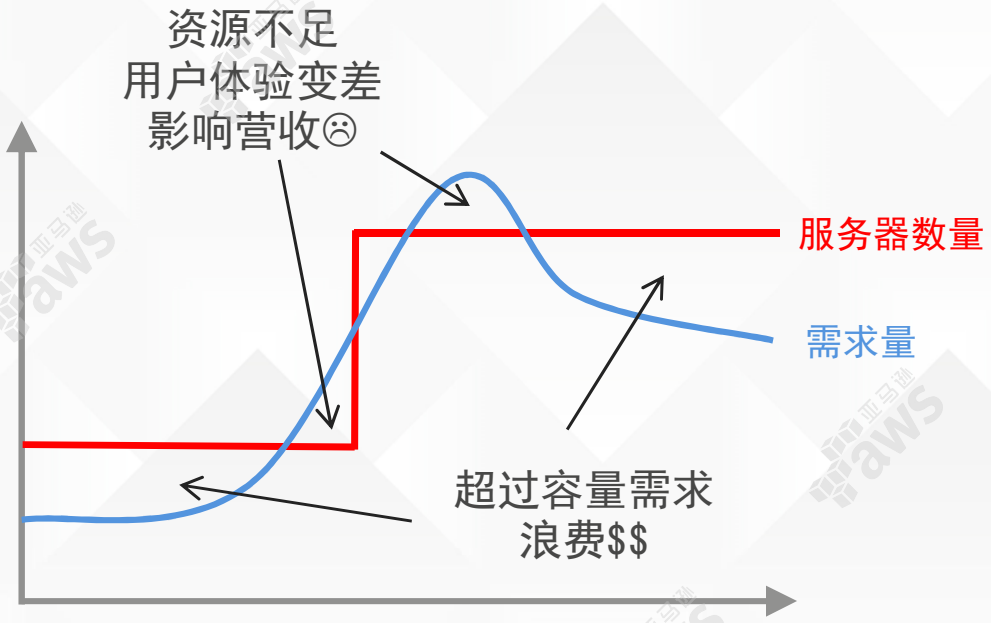
按需扩展，按用量付费



按需扩展，按用量付费

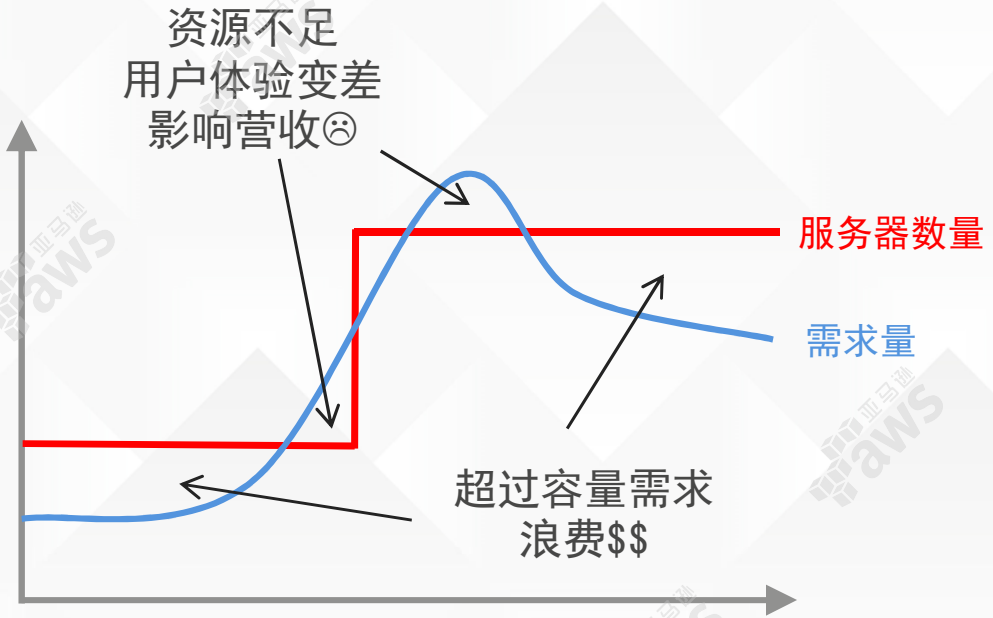


按需扩展，按用量付费

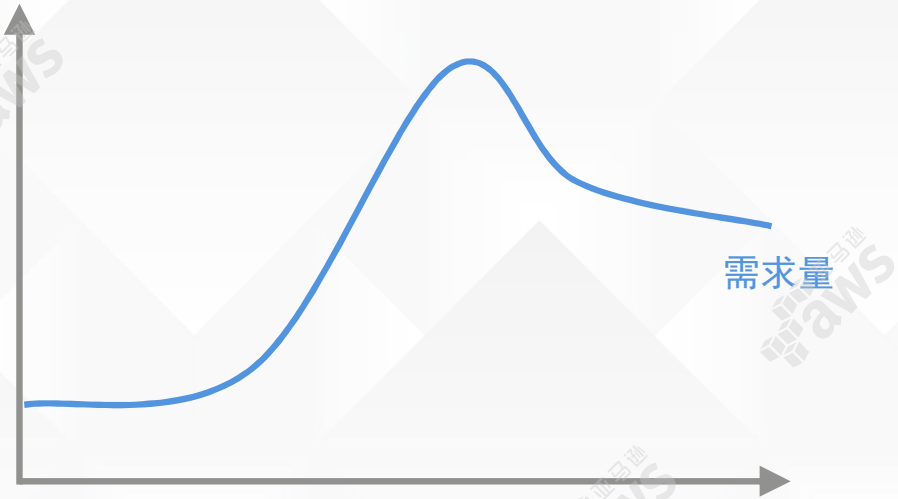


传统方式: 不灵活

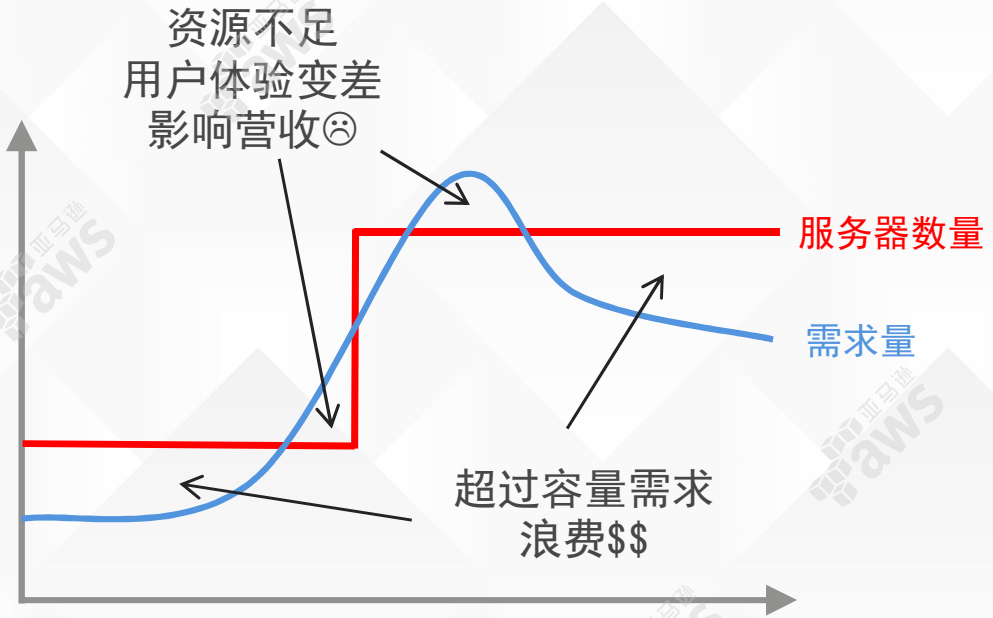
按需扩展，按用量付费



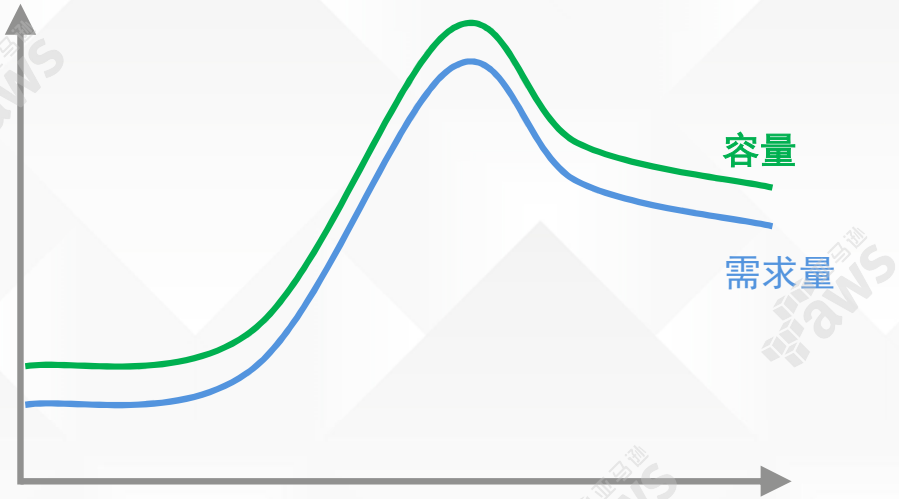
传统方式: 不灵活



按需扩展，按用量付费



传统方式: 不灵活



AWS: 弹性

全球部署

11个区域

53个边缘节点

仍在持续扩展

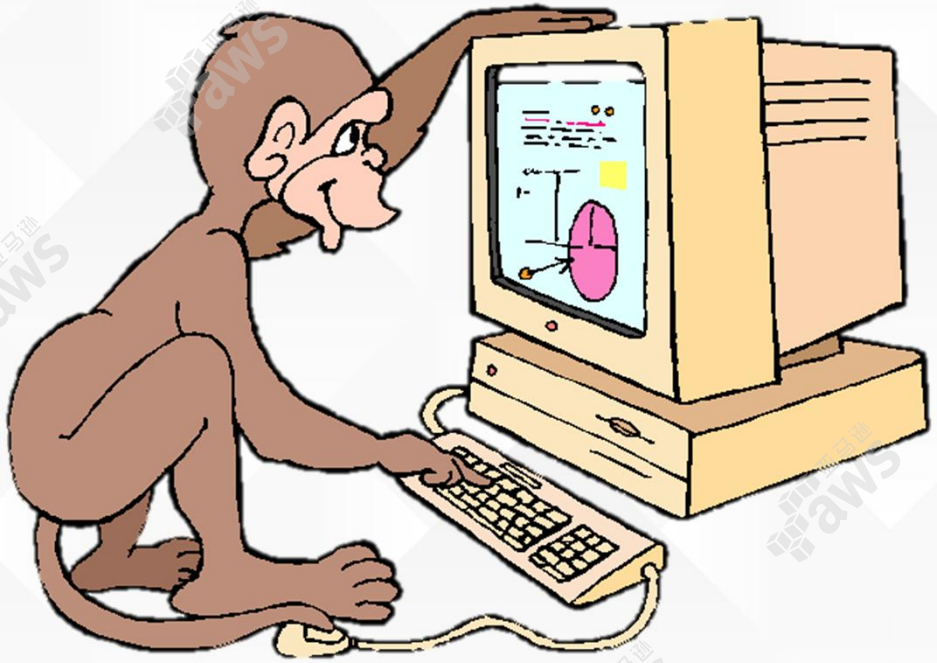
全球部署

11个区域

53个边缘节点

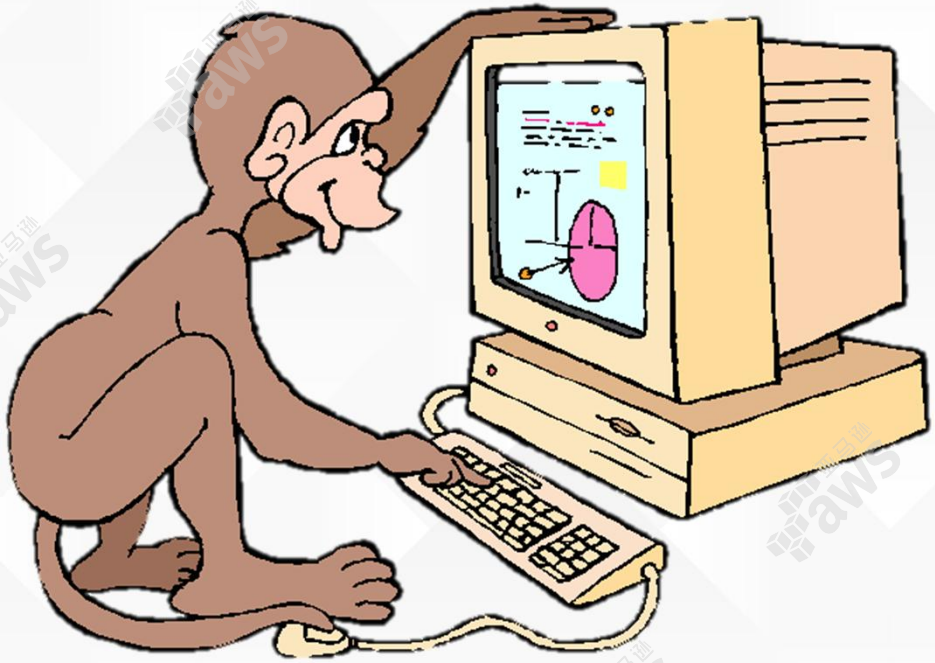
仍在持续扩展

一些经常提到的游戏后端设计理念



一些经常提到的游戏后端设计理念

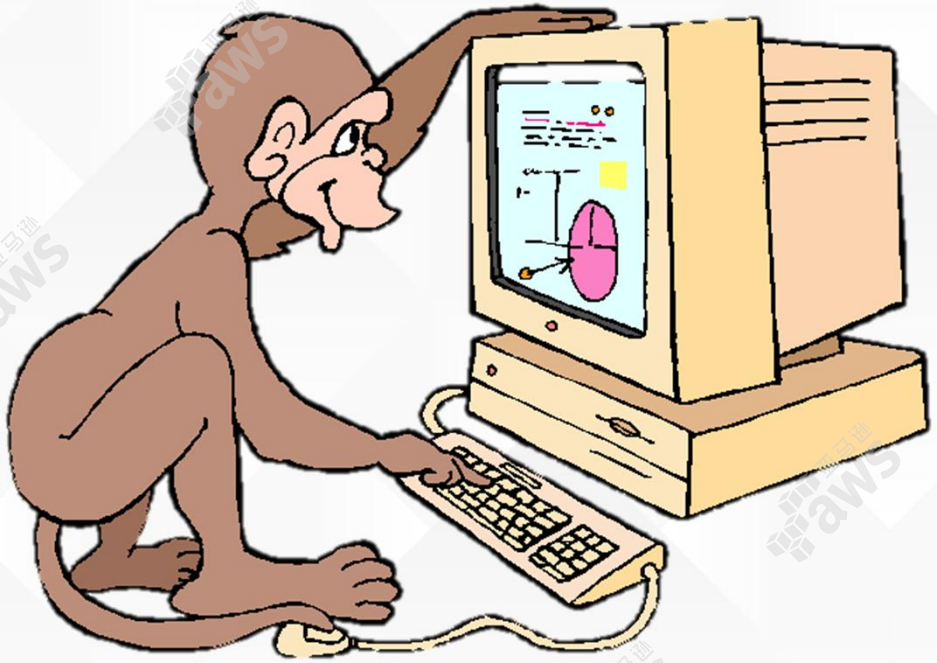
API设计模式



一些经常提到的游戏后端设计理念

API设计模式

HTTP + JSON

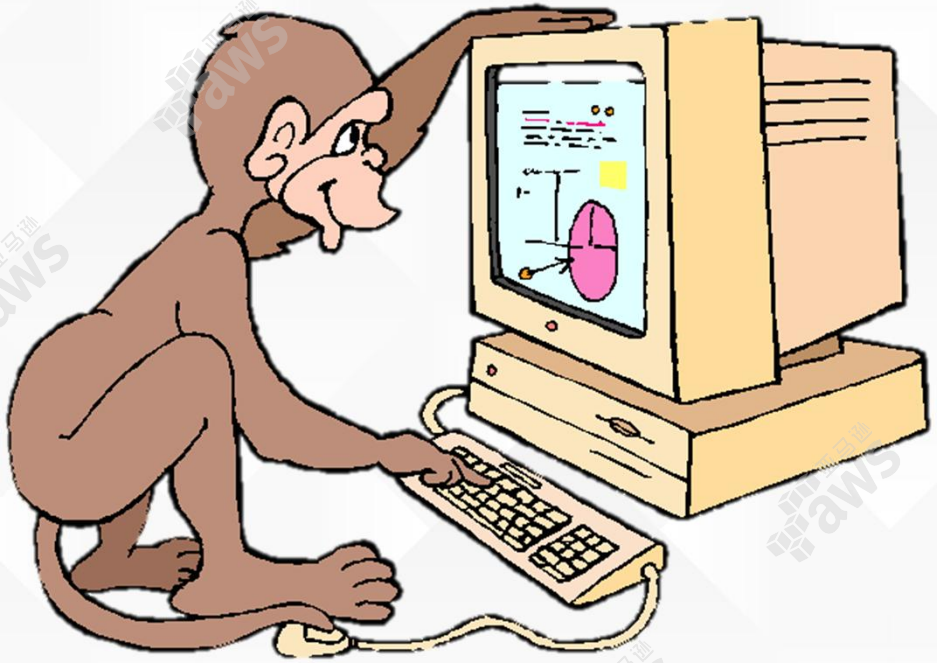


一些经常提到的游戏后端设计理念

API设计模式

HTTP + JSON

交友，积分榜



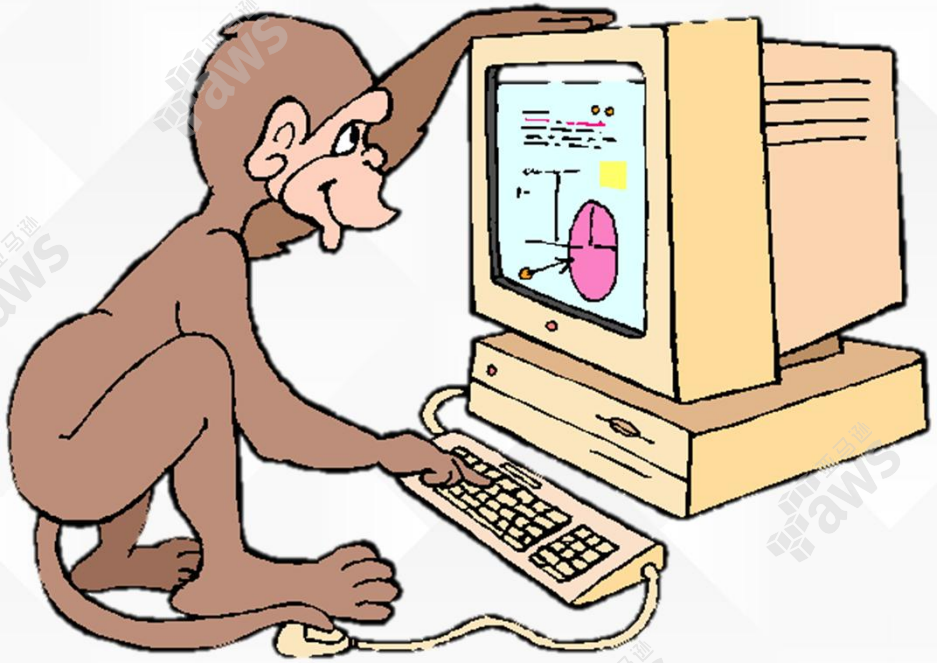
一些经常提到的游戏后端设计理念

API设计模式

HTTP + JSON

交友，积分榜

打包，资源，数据



一些经常提到的游戏后端设计理念

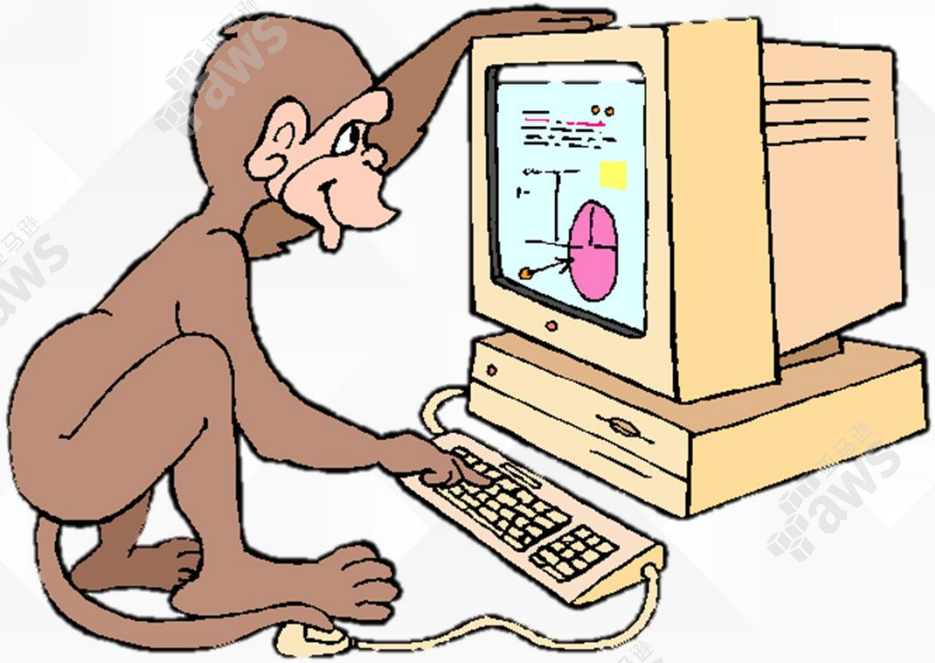
API设计模式

HTTP + JSON

交友，积分榜

打包，资源，数据

多玩家服务



一些经常提到的游戏后端设计理念

API设计模式

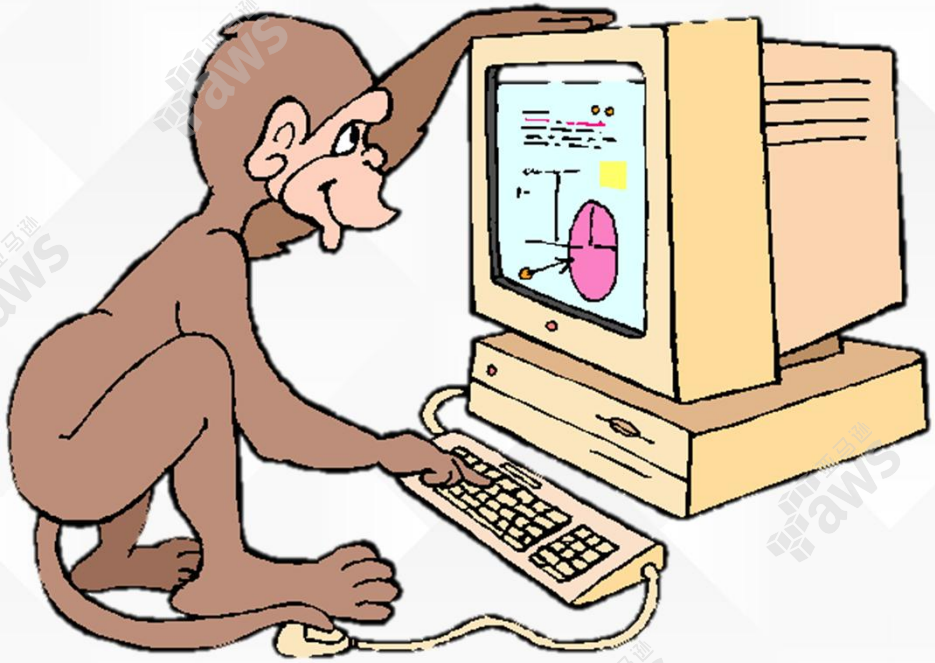
HTTP + JSON

交友，积分榜

打包，资源，数据

多玩家服务

高可用



一些经常提到的游戏后端设计理念

API设计模式

HTTP + JSON

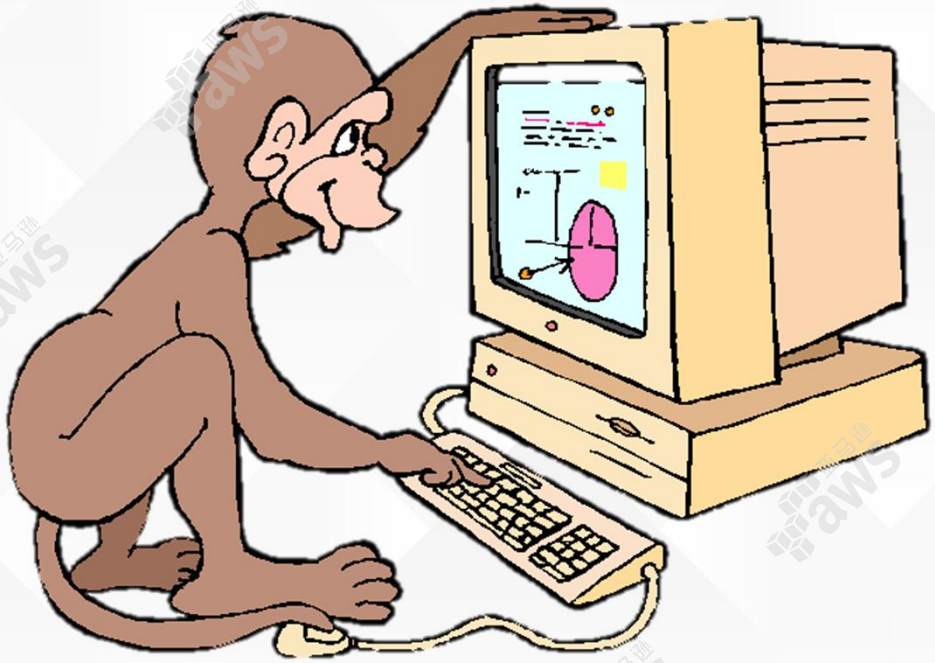
交友，积分榜

打包，资源，数据

多玩家服务

高可用

扩展性



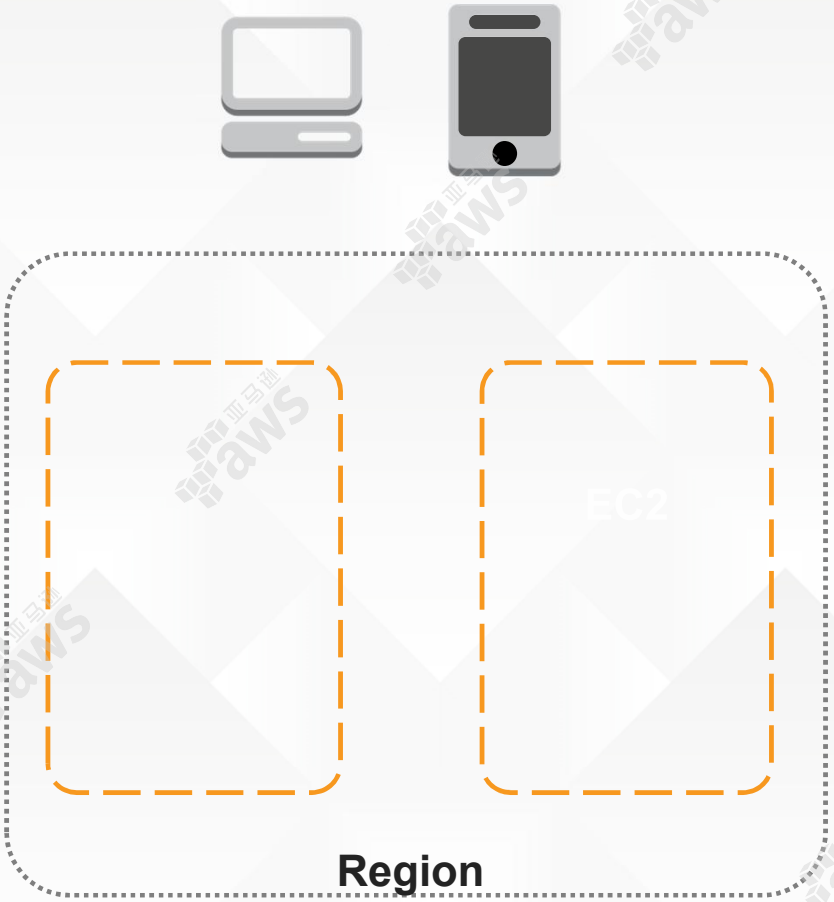
高可用游戏核心后端架构

高可用游戏核心后端架构

- 选择区域

高可用游戏核心后端架构

- 选择区域
- >=2 可用区

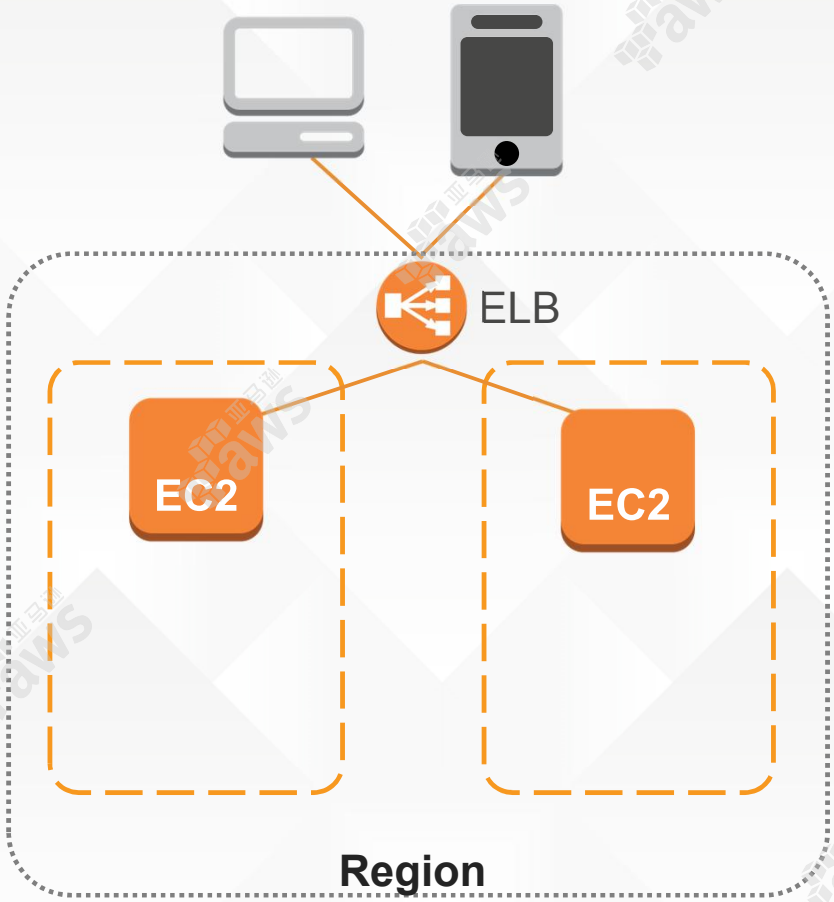


高可用游戏核心后端架构

- 选择区域
- >=2 可用区
- App使用EC2

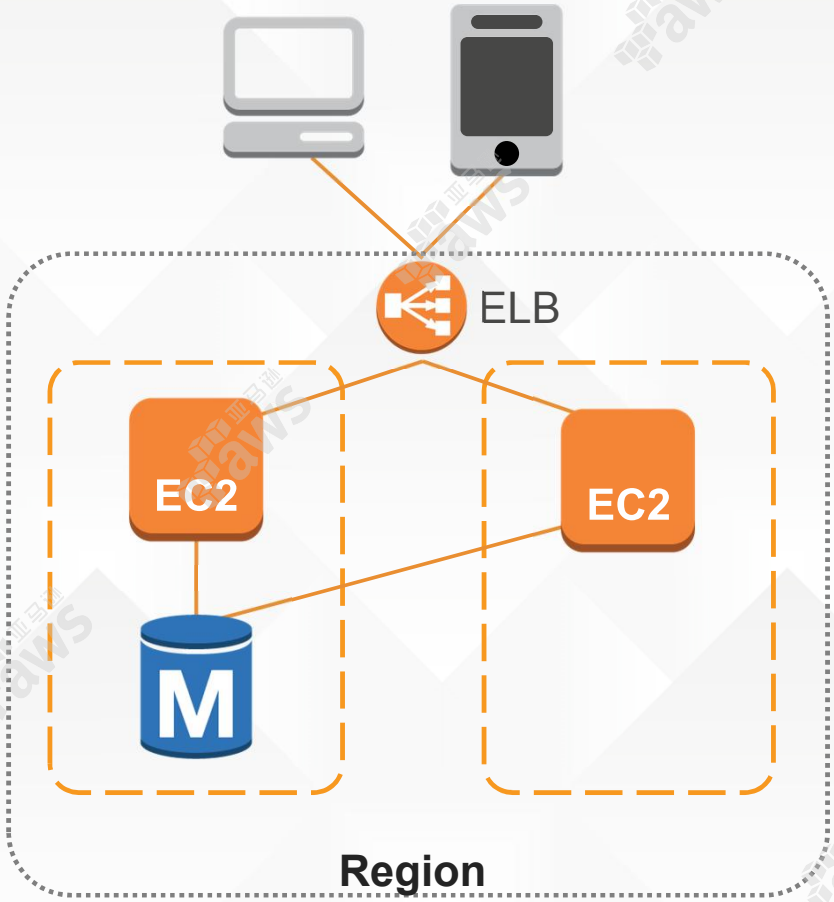
高可用游戏核心后端架构

- 选择区域
- >=2 可用区
- App使用EC2
- 弹性负载均衡



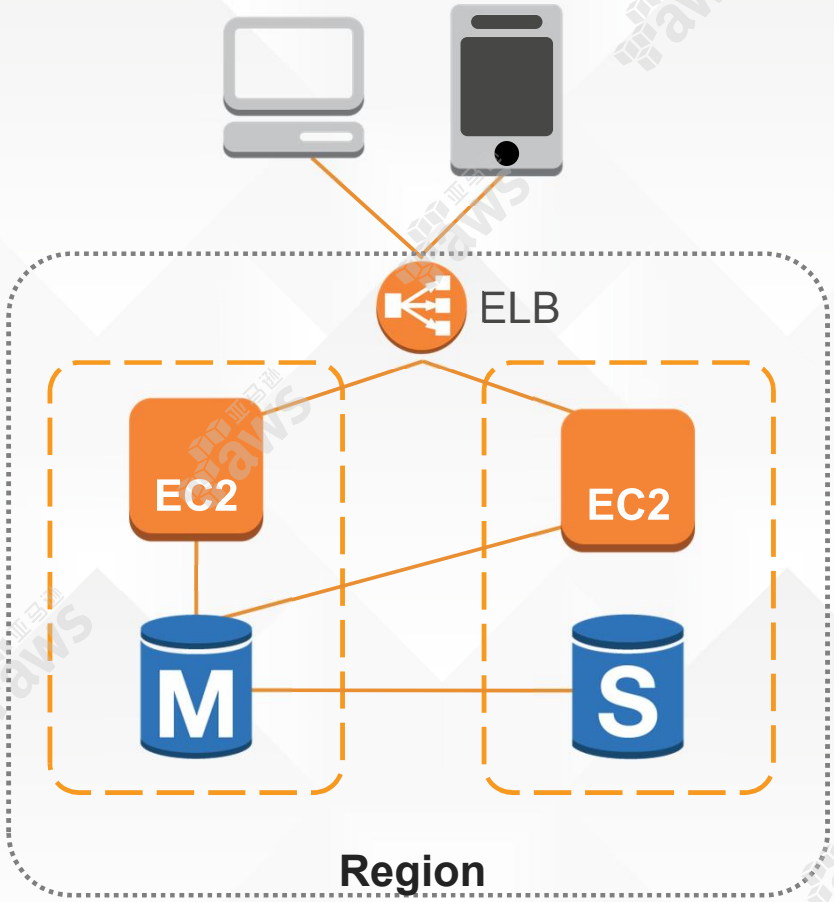
高可用游戏核心后端架构

- 选择区域
- >=2 可用区
- App使用EC2
- 弹性负载均衡
- RDS



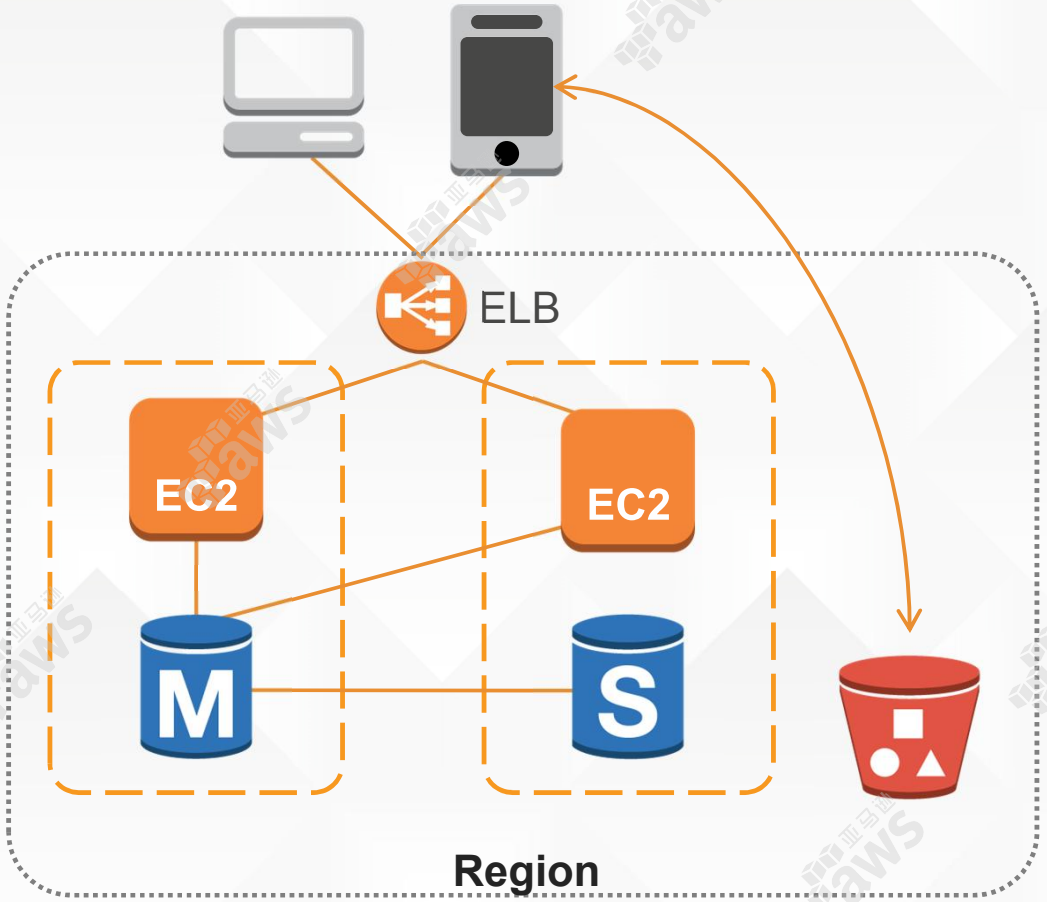
高可用游戏核心后端架构

- 选择区域
- >=2 可用区
- App使用EC2
- 弹性负载均衡
- RDS
  - 多可用区部署



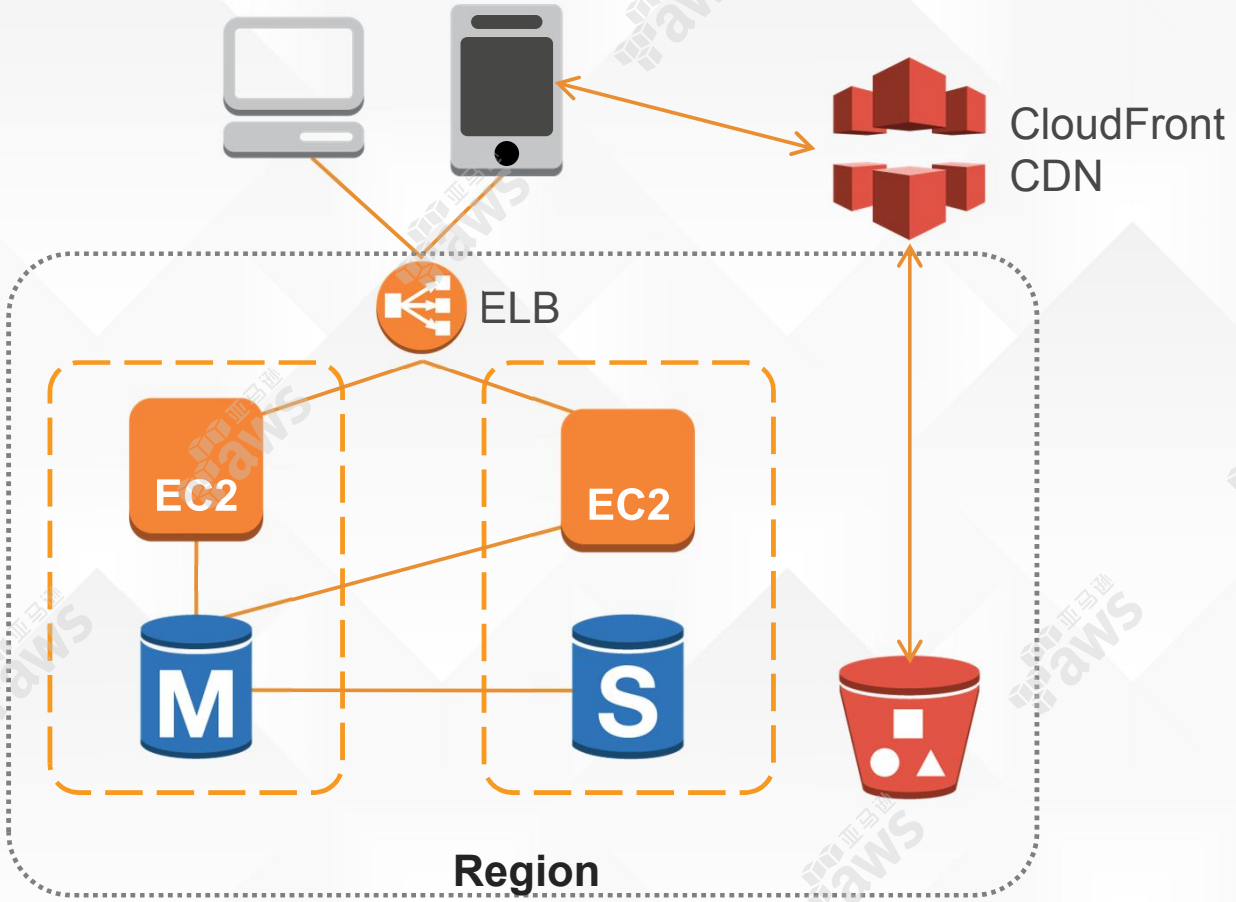
继续扩展

- 使用S3存储游戏数据
  - 资源文件
  - 用户创建内容
  - 分析数据



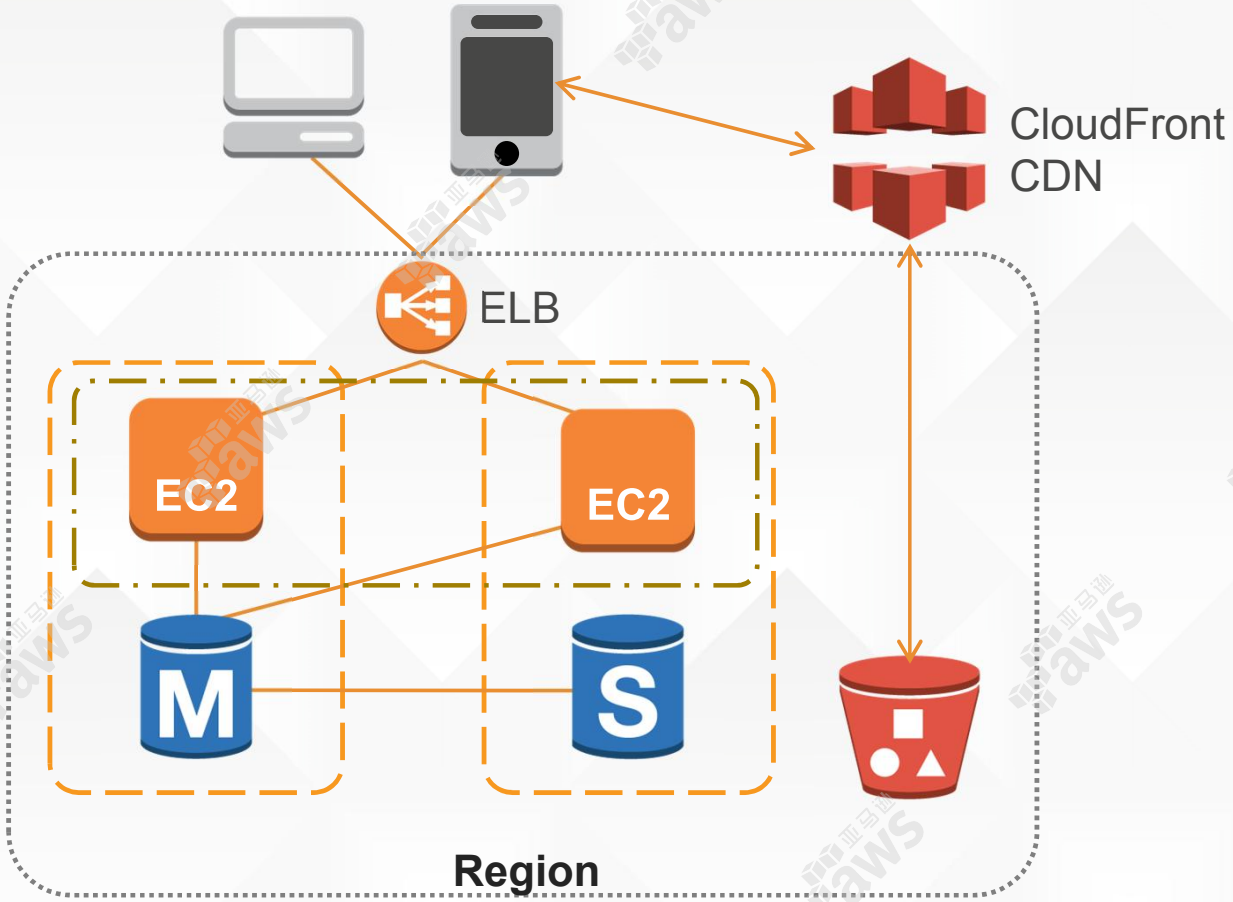
继续扩展

- 使用S3存储游戏数据
  - 资源文件
  - 用户创建内容
  - 分析数据
  - …利用CloudFront



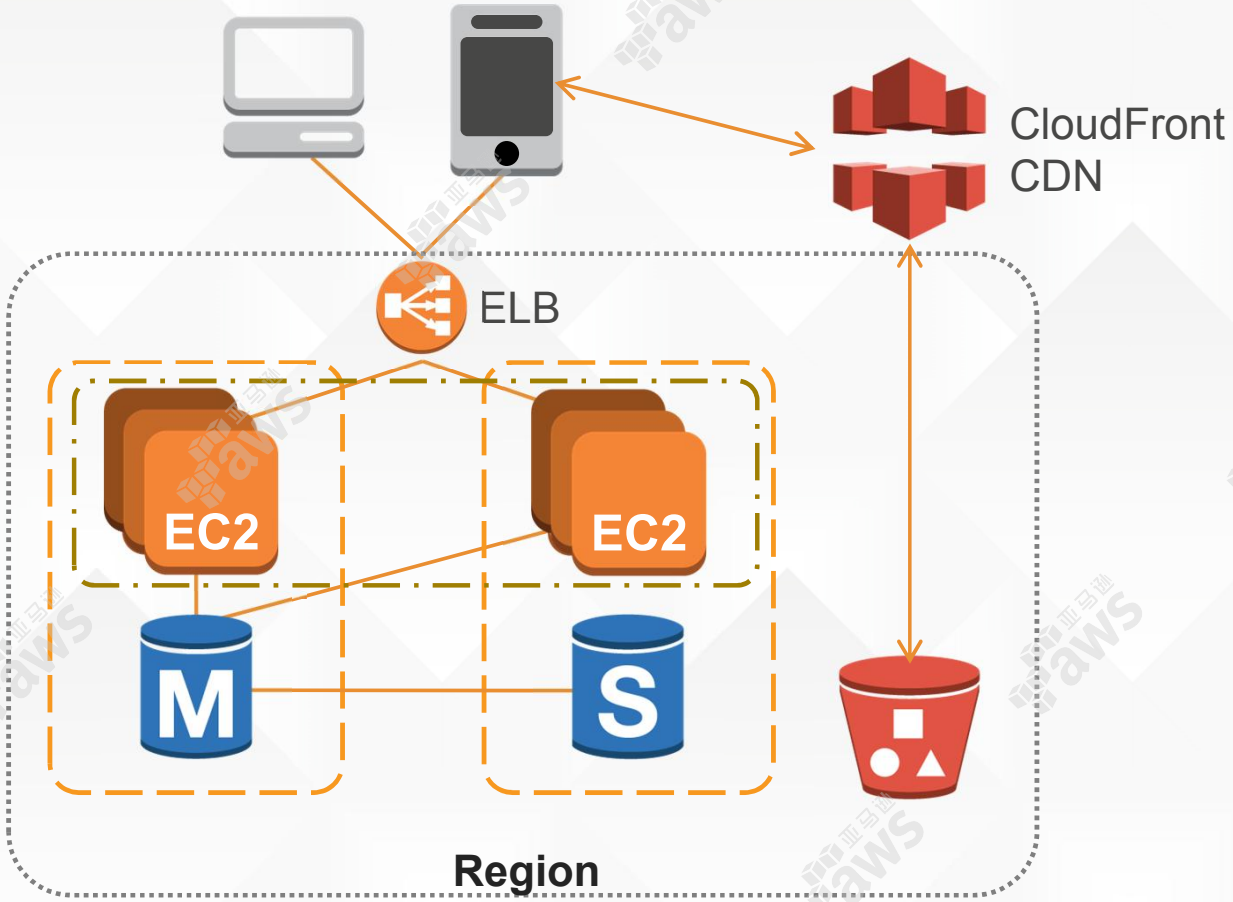
继续扩展

- 使用S3存储游戏数据
  - 资源文件
  - 用户创建内容
  - 分析数据
  - …利用CloudFront
- Auto Scaling Group
  - 按需调整容量
  - 根据用户量调整
  - 自愈



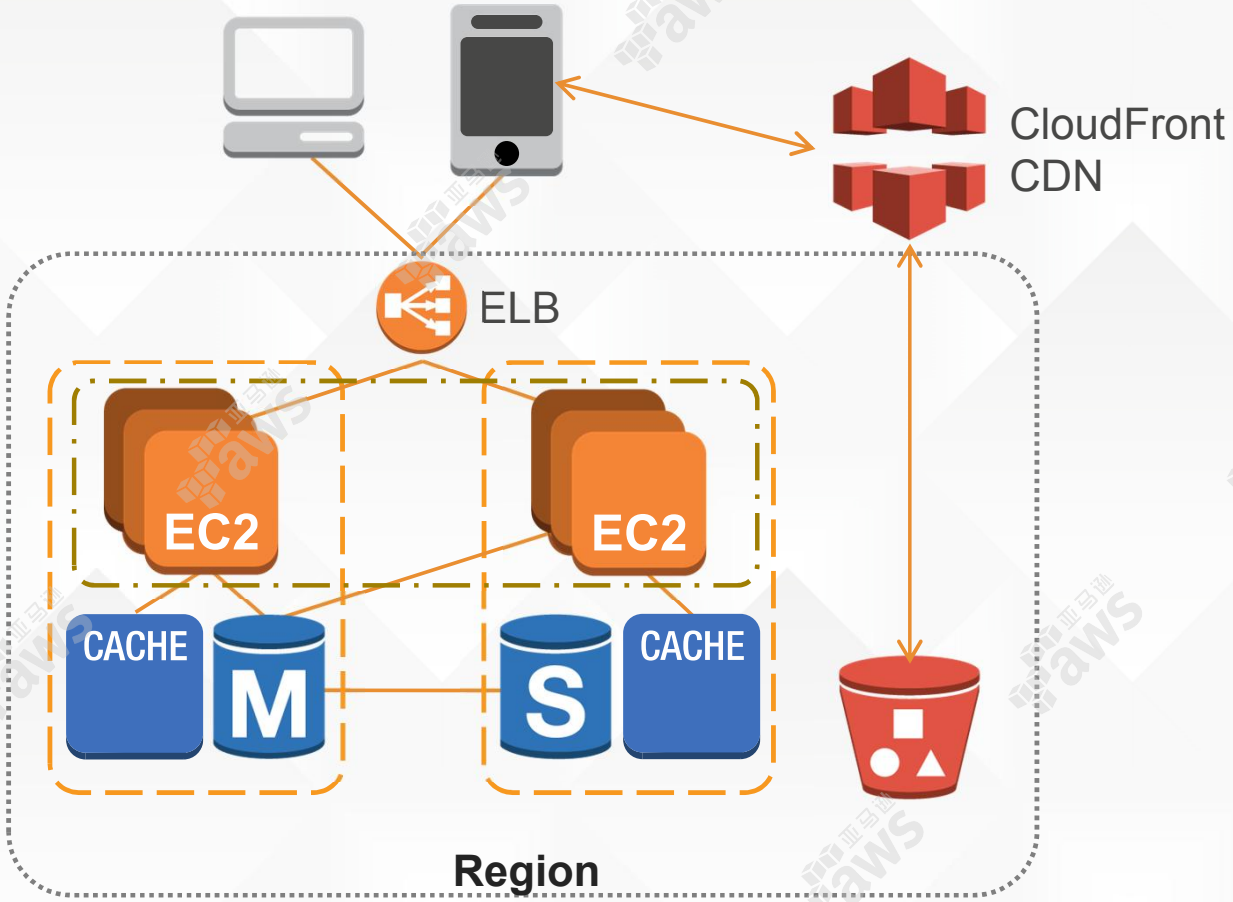
继续扩展

- 使用S3存储游戏数据
  - 资源文件
  - 用户创建内容
  - 分析数据
  - …利用CloudFront
- Auto Scaling Group
  - 按需调整容量
  - 根据用户量调整
  - 自愈

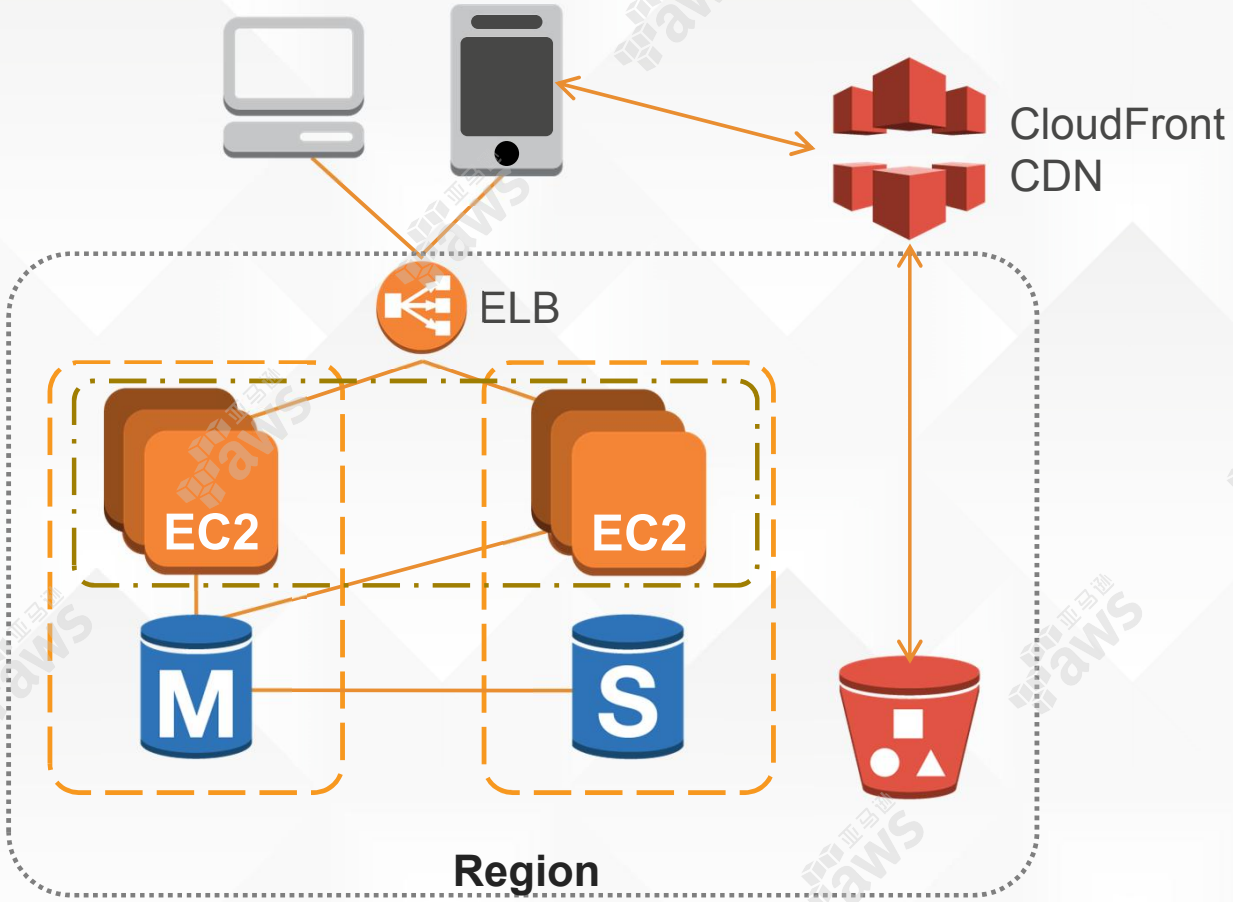


继续扩展

- 使用S3存储游戏数据
  - 资源文件
  - 用户创建内容
  - 分析数据
  - …利用CloudFront
- Auto Scaling Group
  - 按需调整容量
  - 根据用户量调整
  - 自愈
- ElastiCache
  - Memcached
  - Redis

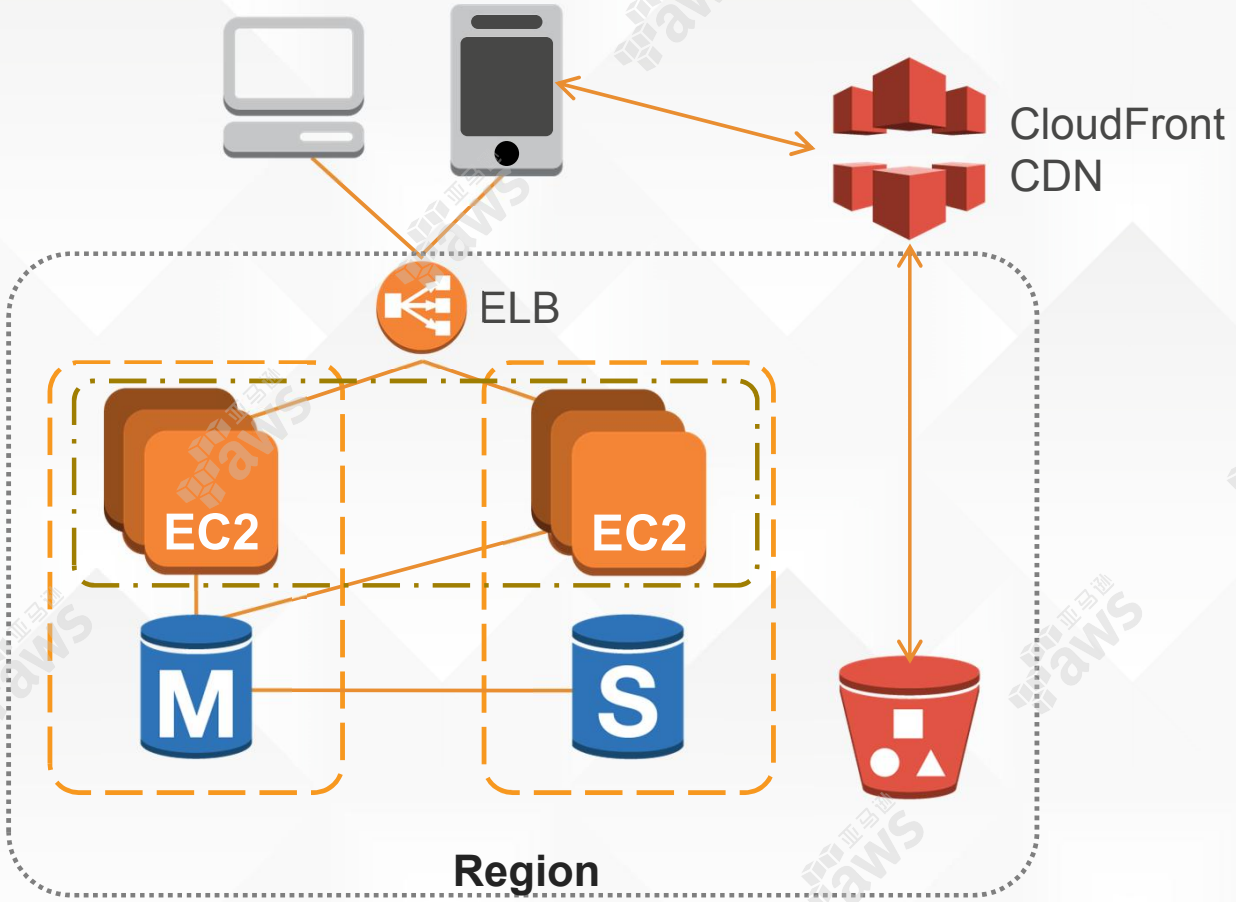


继续扩展



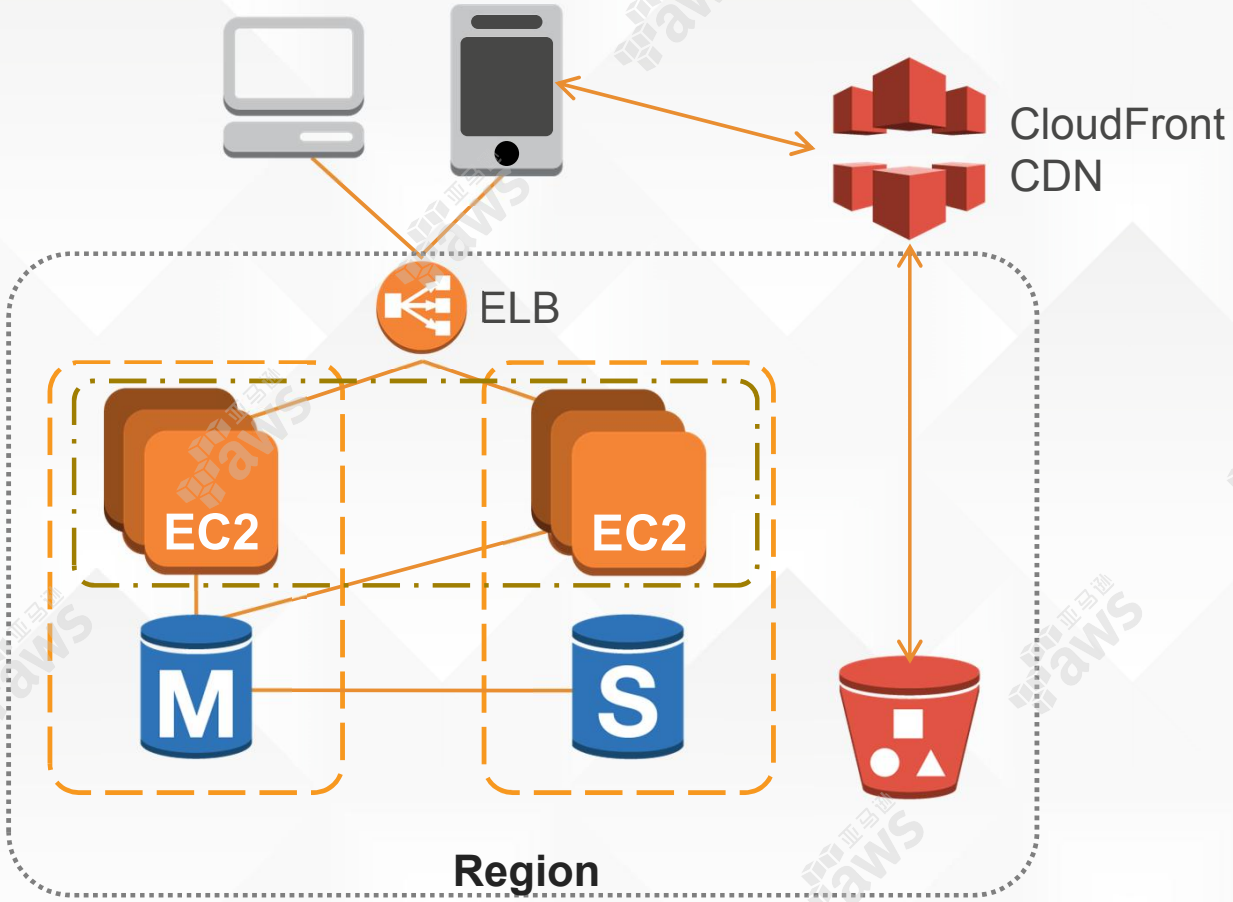
继续扩展

- 通常情况下写负载会更高



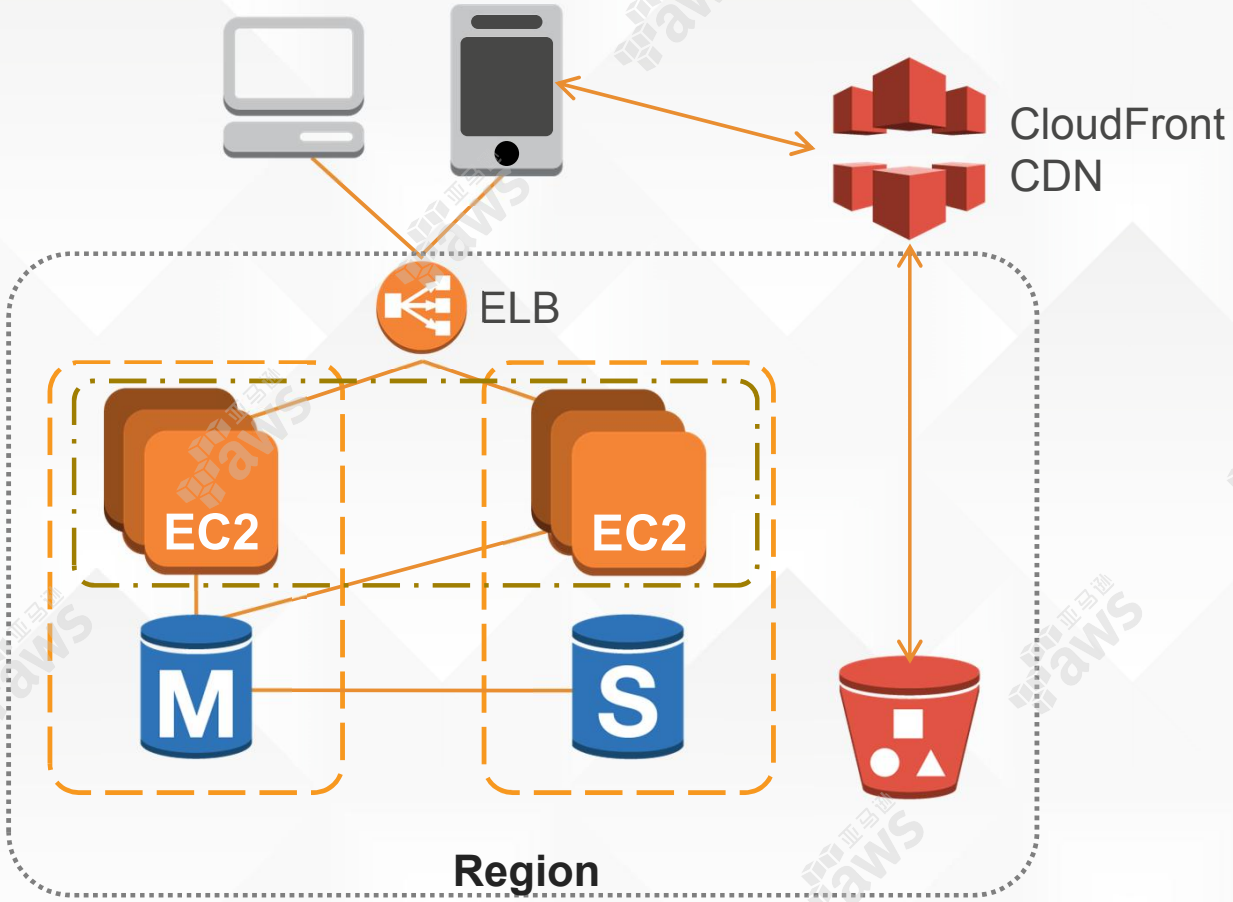
继续扩展

- 通常情况下写负载会更高
- 缓存能提供的帮助有限



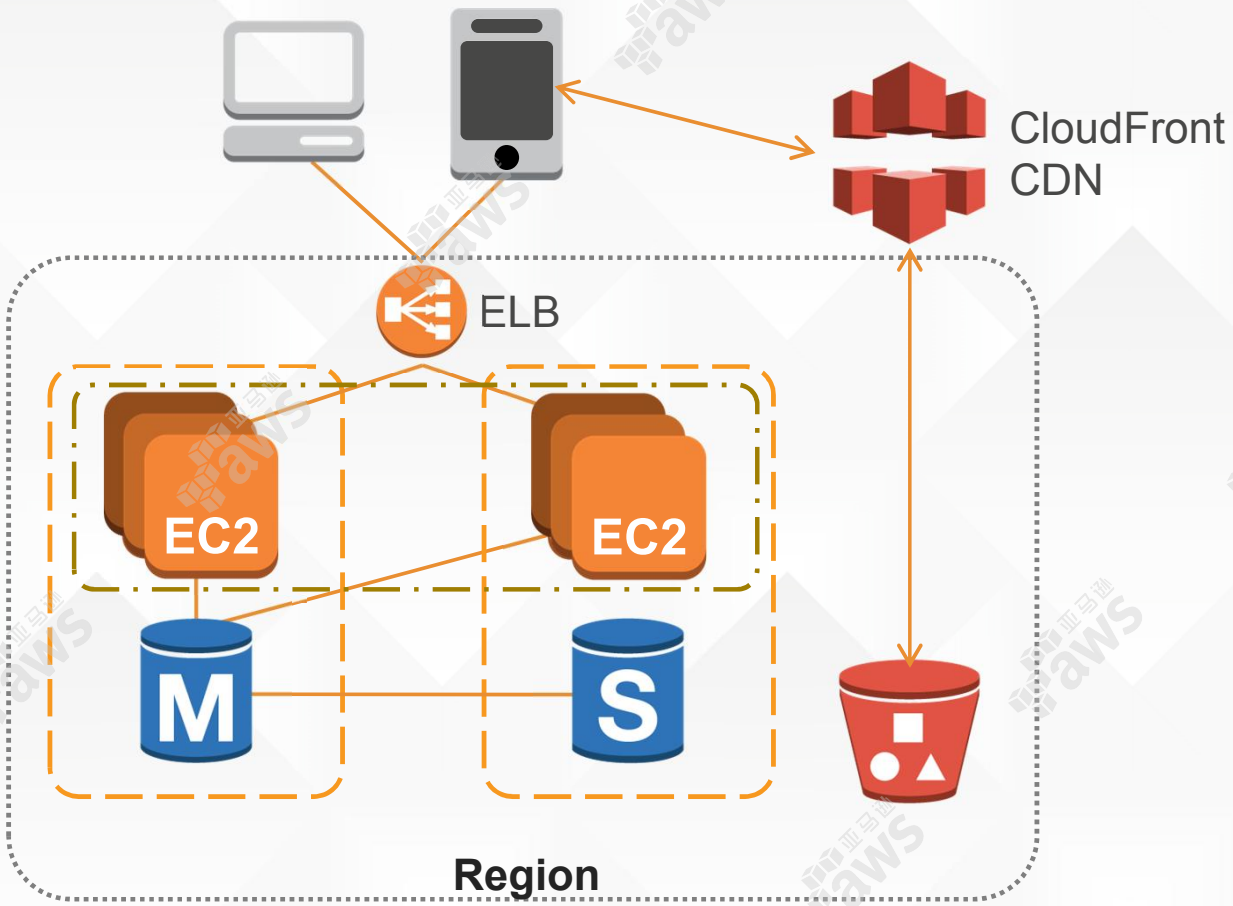
继续扩展

- 通常情况下写负载会更高
- 缓存能提供的帮助有限
- Key Value



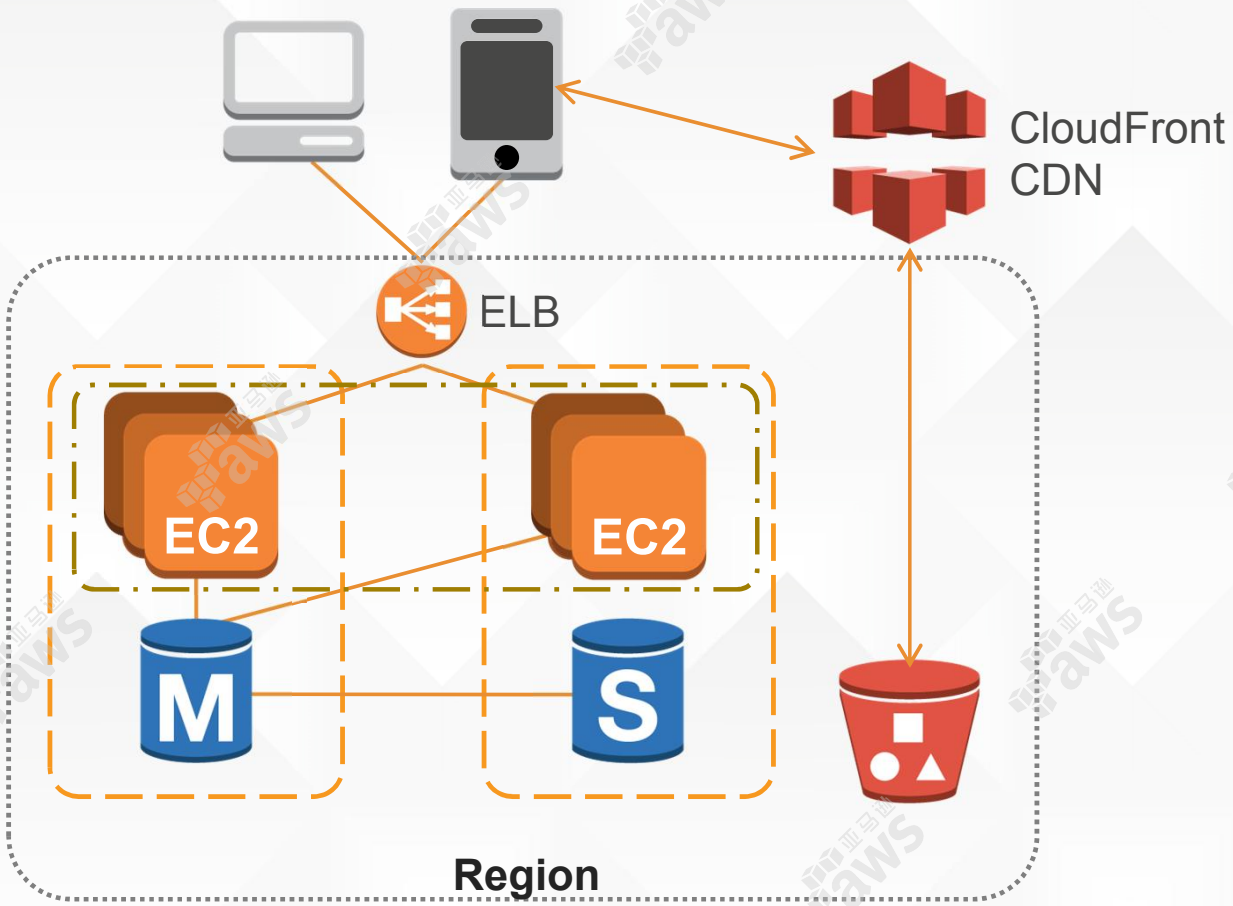
继续扩展

- 通常情况下写负载会更高
- 缓存能提供的帮助有限
- Key Value
- 二进制结构



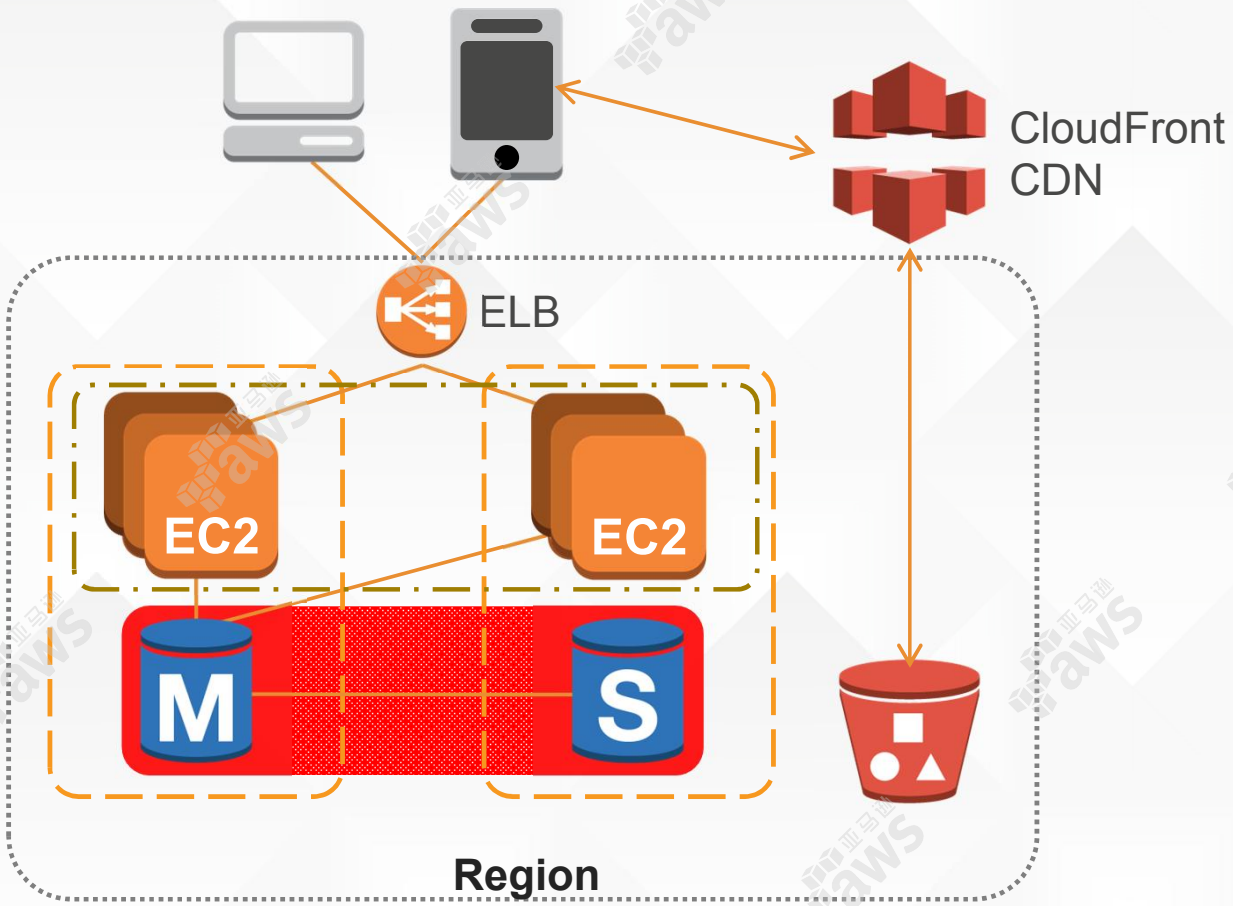
继续扩展

- 通常情况下写负载会更高
- 缓存能提供的帮助有限
- Key Value
- 二进制结构
- 数据库 = 瓶颈

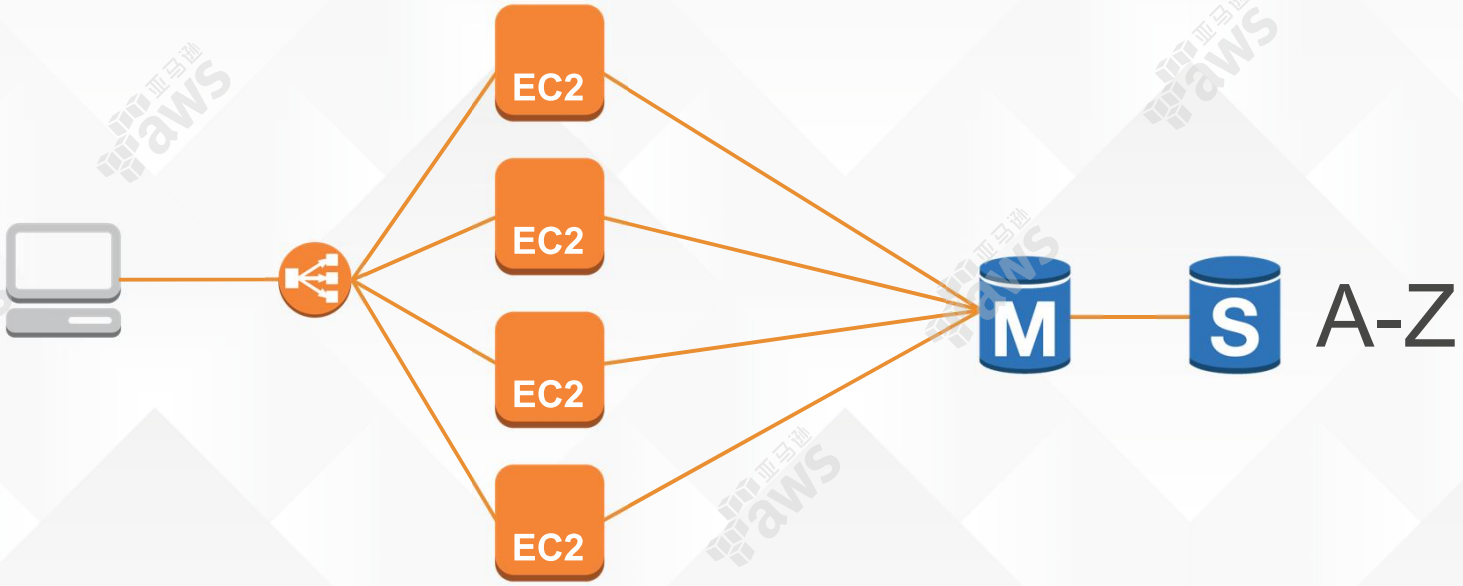


继续扩展

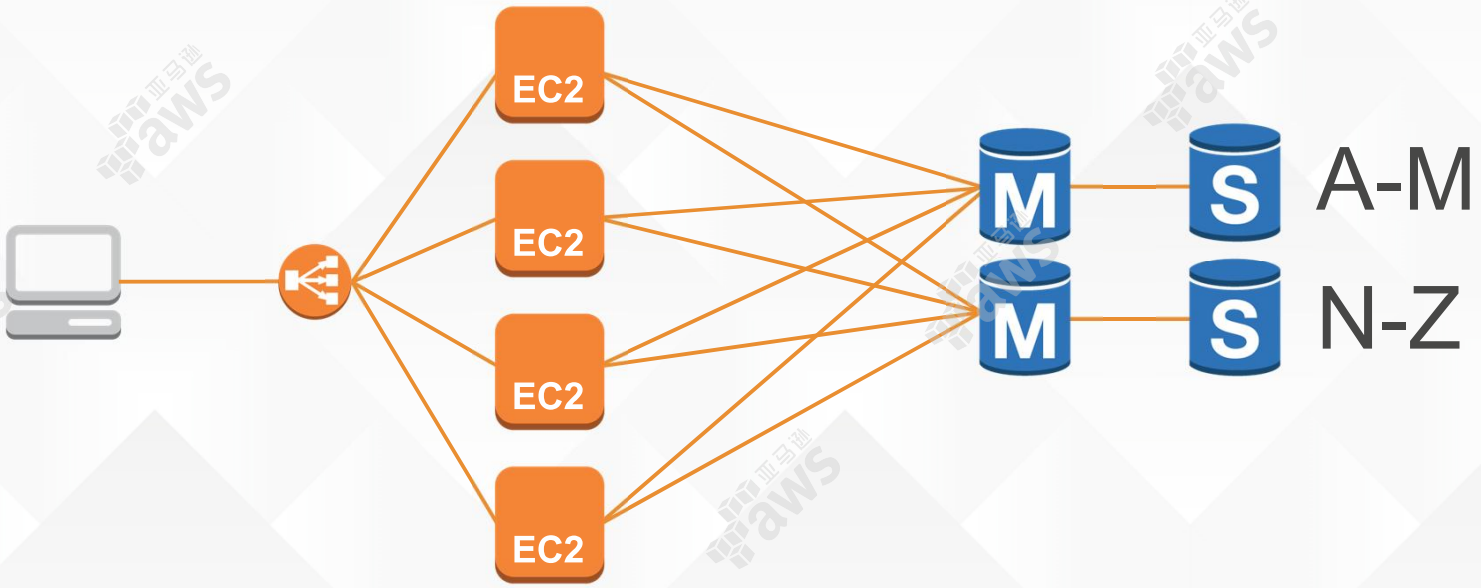
- 通常情况下写负载会更高
- 缓存能提供的帮助有限
- Key Value
- 二进制结构
- 数据库 = 瓶颈



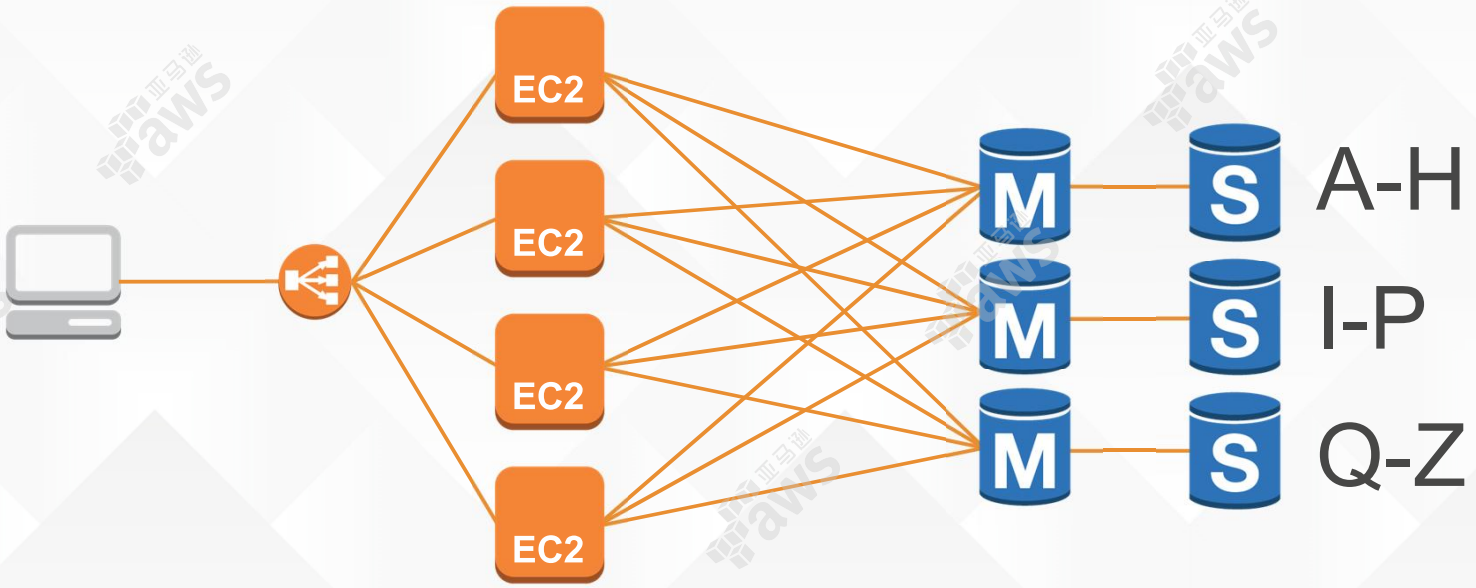
数据库分片



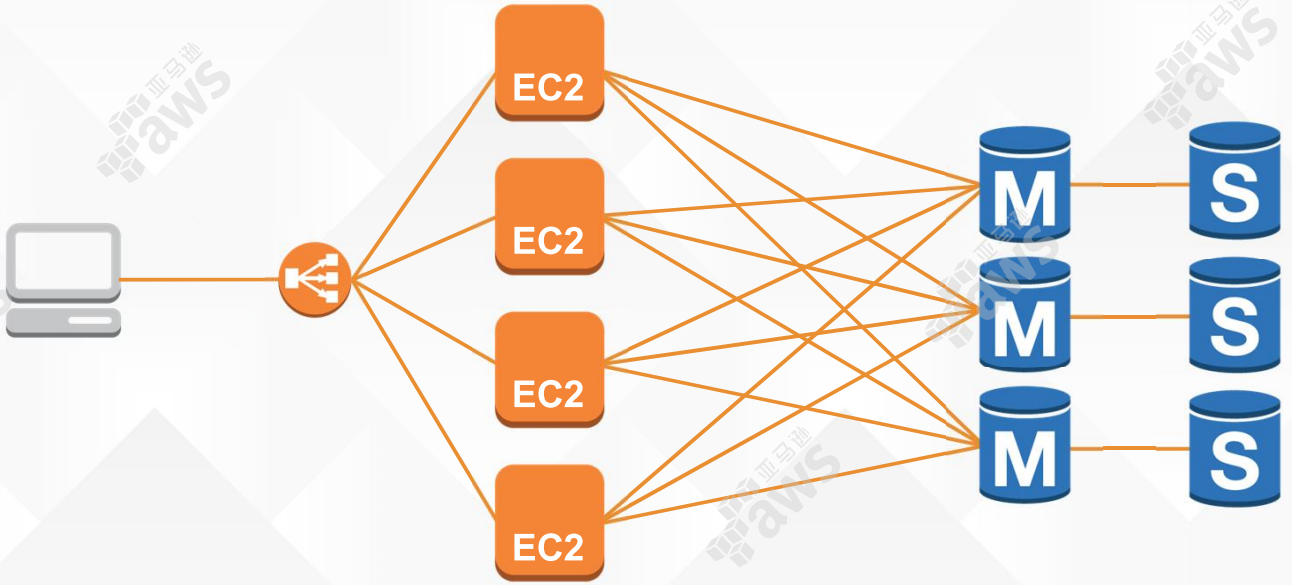
数据库分片

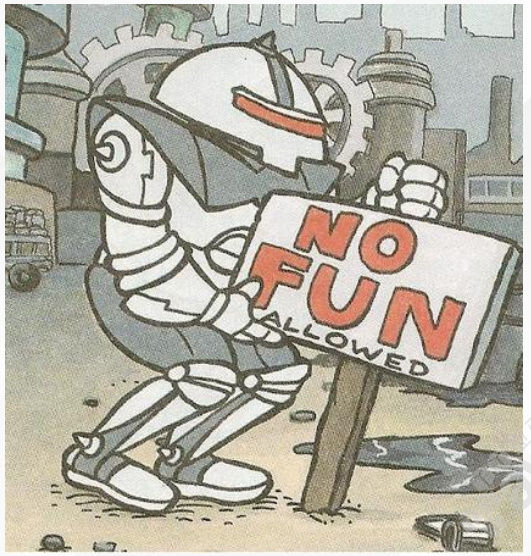


数据库分片

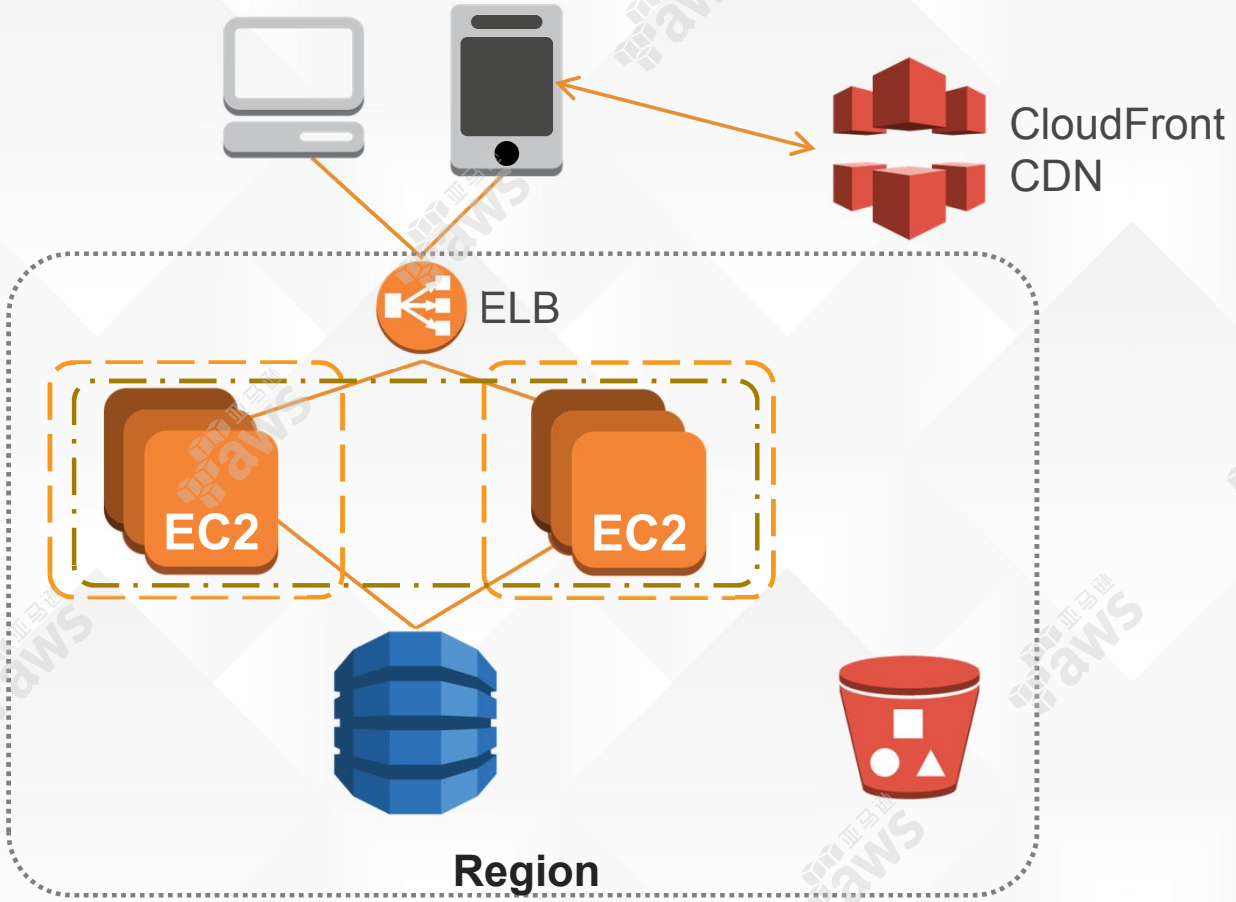


数据库分片



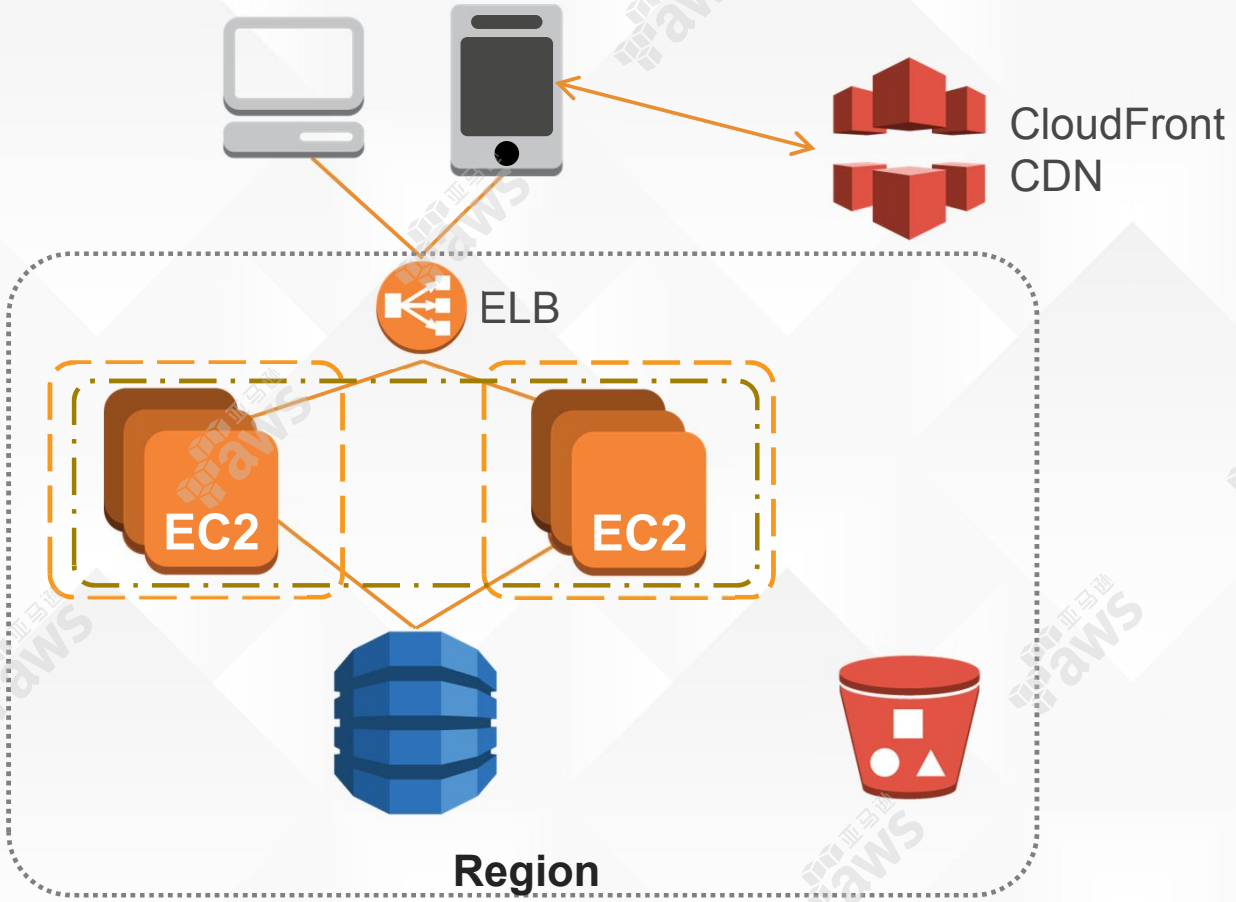


Amazon DynamoDB



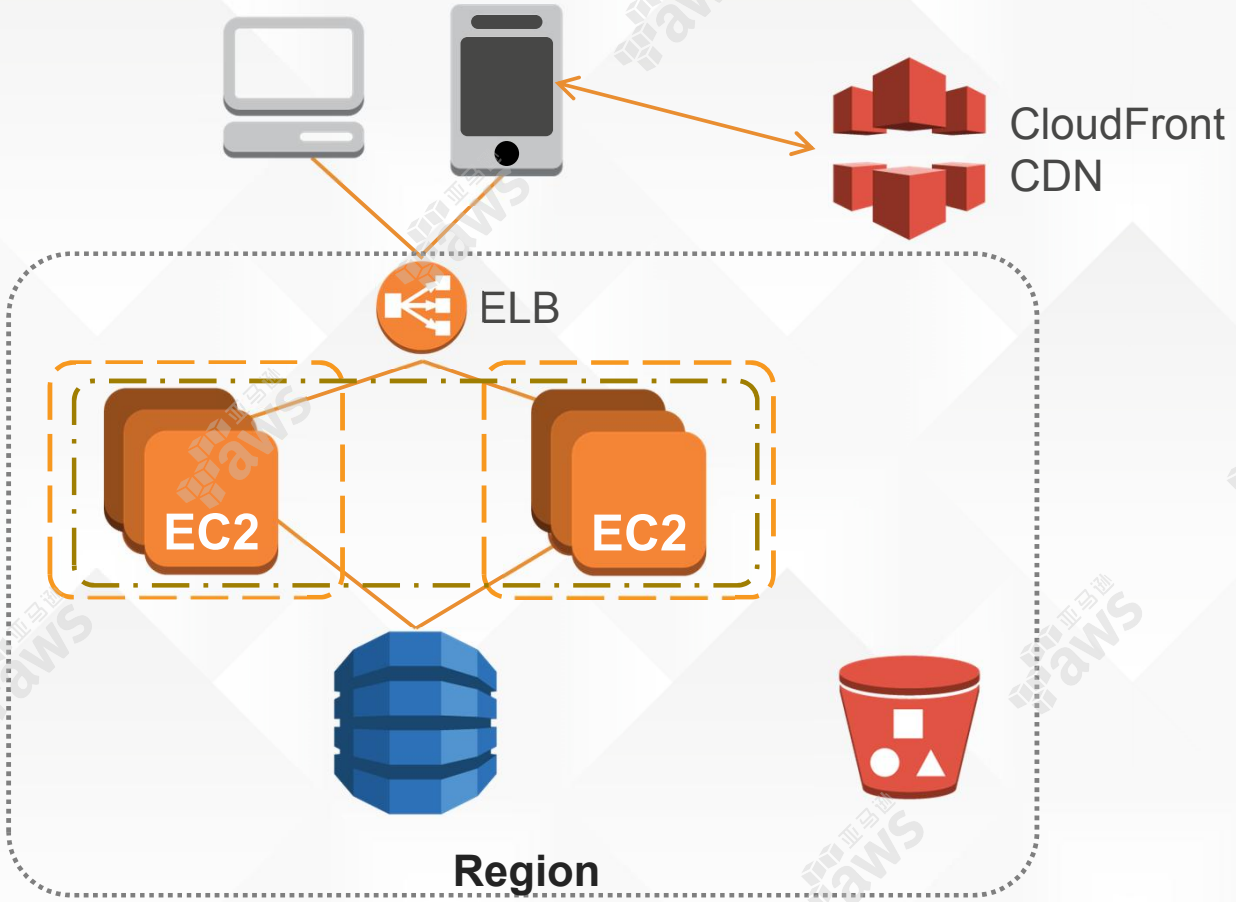
Amazon DynamoDB

- 完全托管



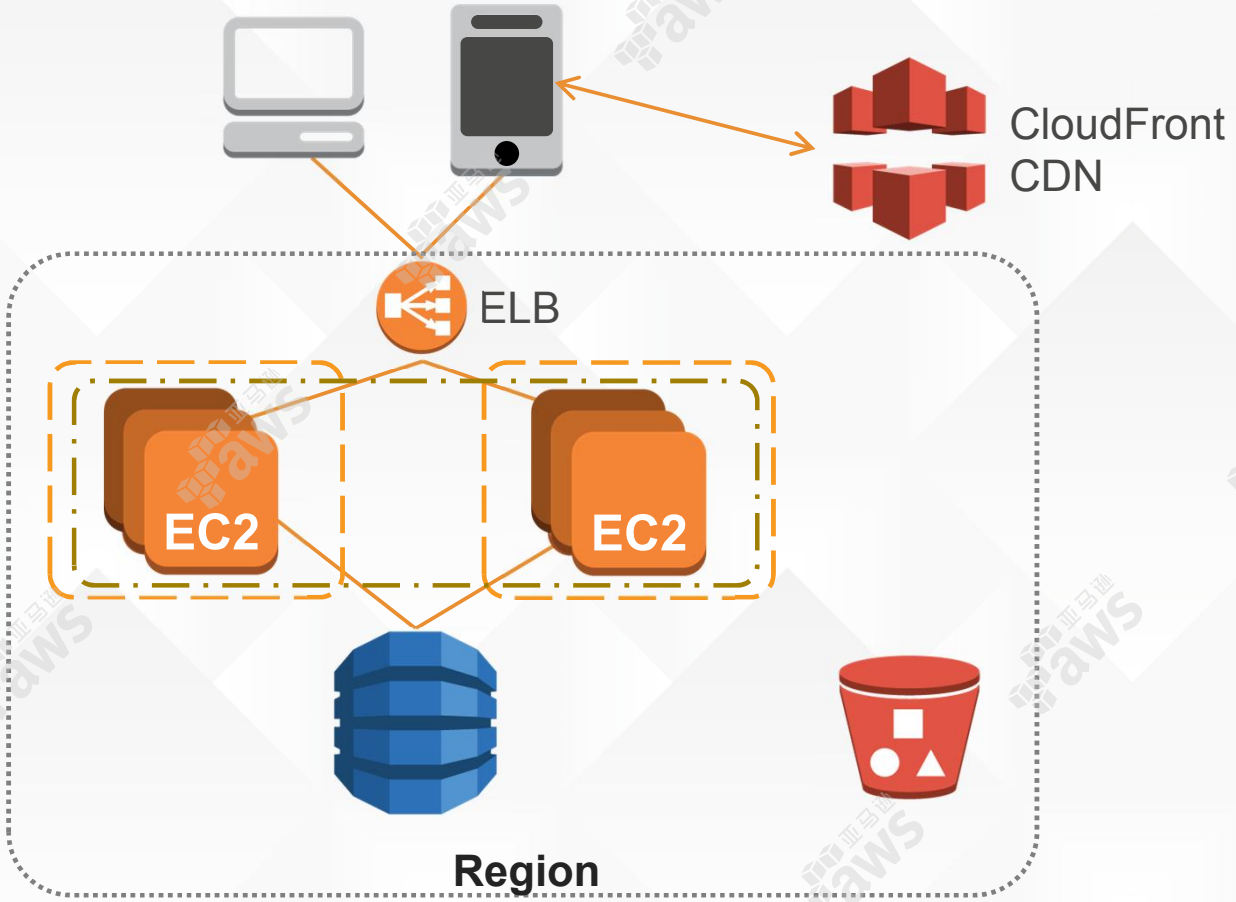
Amazon DynamoDB

- 完全托管
- NoSQL数据存储



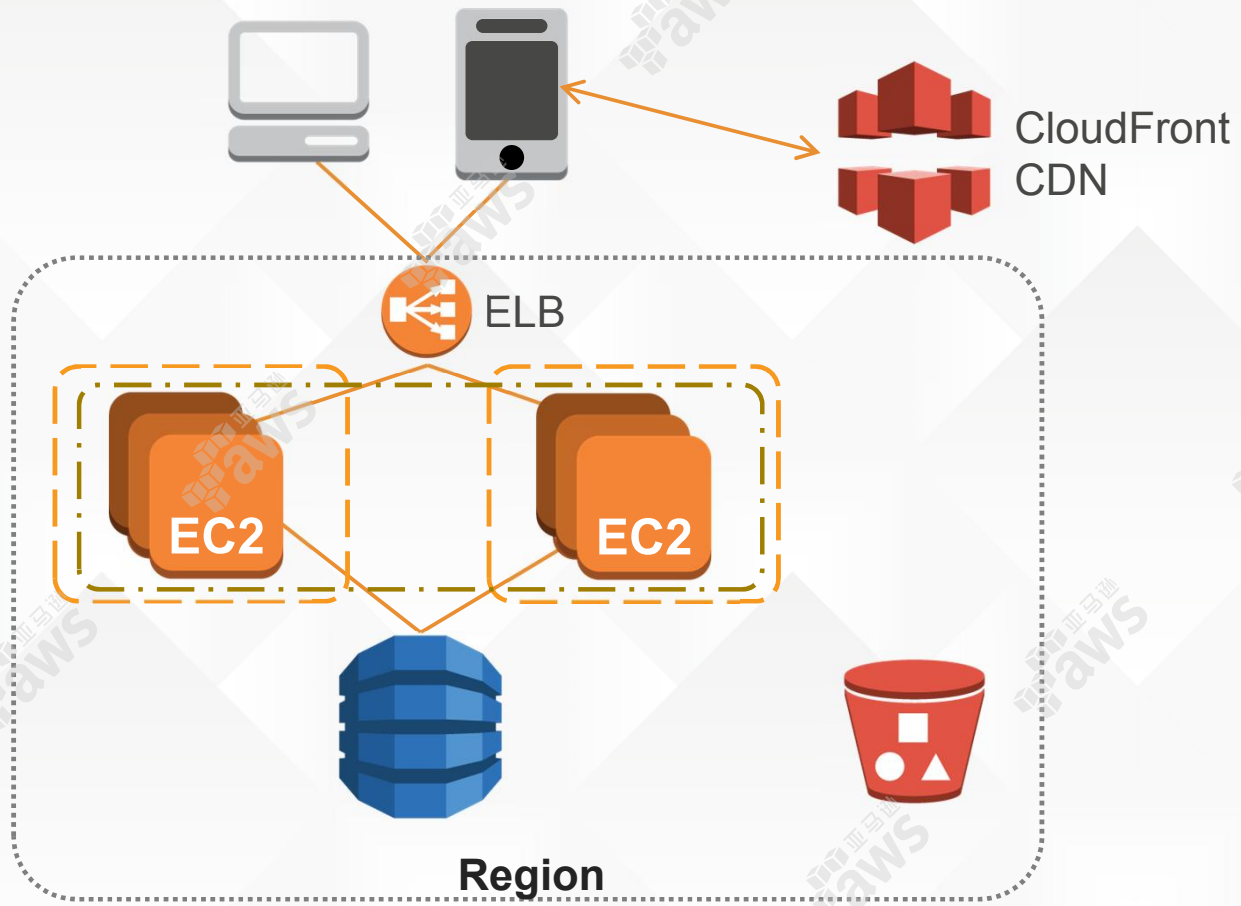
Amazon DynamoDB

- 完全托管
- NoSQL数据存储
- 预定义吞吐量



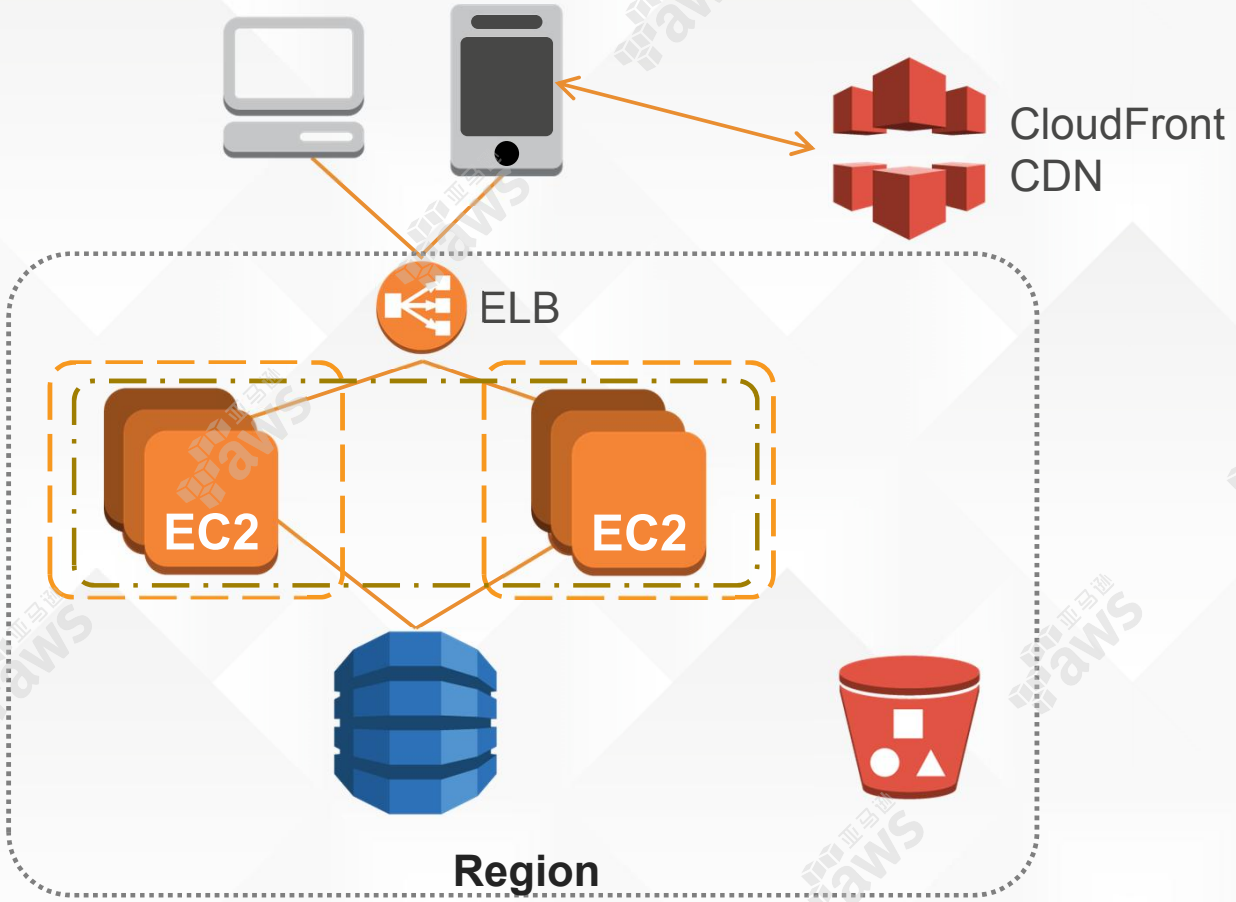
Amazon DynamoDB

- 完全托管
- NoSQL数据存储
- 预定义吞吐量
- Secondary Indexes



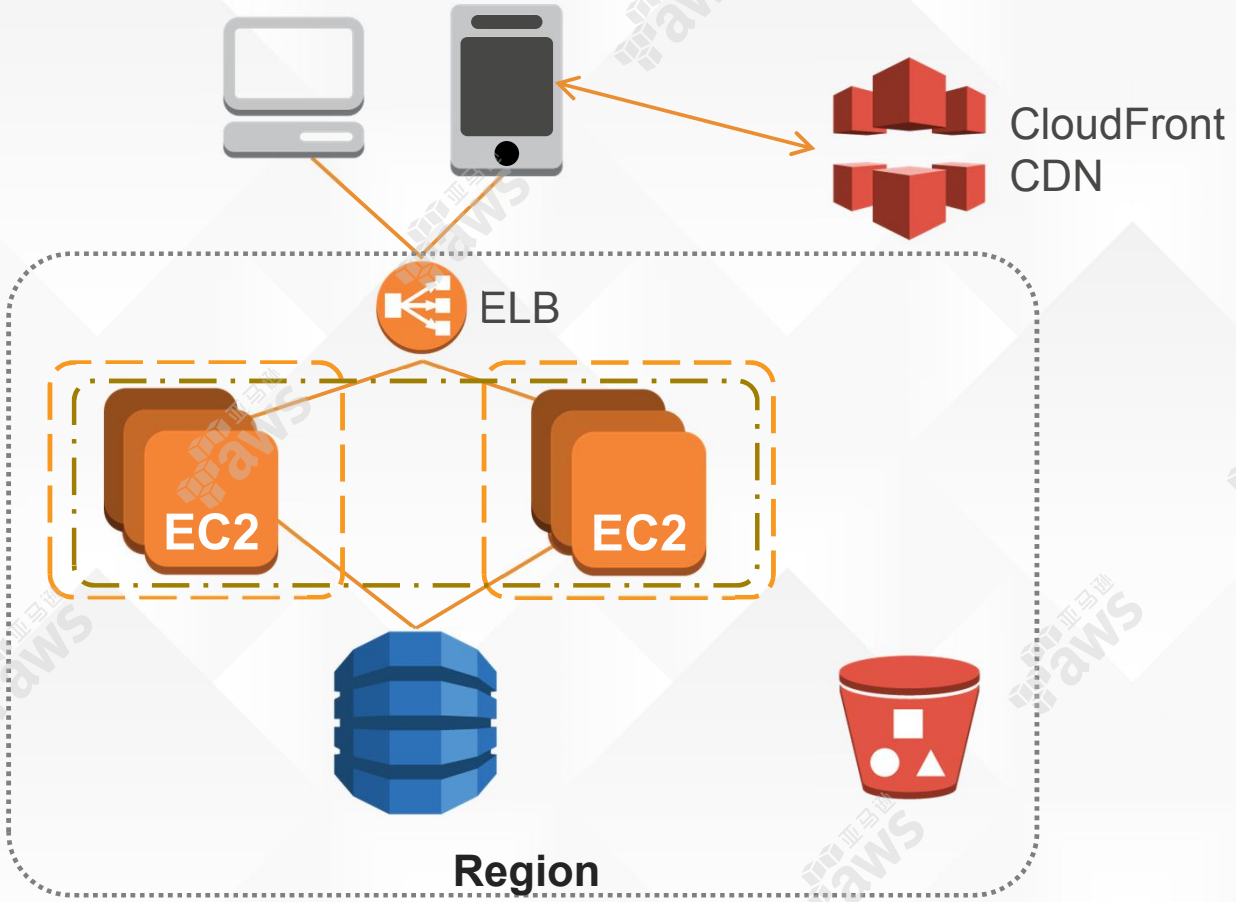
Amazon DynamoDB

- 完全托管
- NoSQL数据存储
- 预定义吞吐量
- Secondary Indexes
- PUT/GET Keys



Amazon DynamoDB

- 完全托管
- NoSQL数据存储
- 预定义吞吐量
- Secondary Indexes
- PUT/GET Keys
- 支持文档



示例：基于Amazon DynamoDB的排行榜

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"
"101"	"Meteor Blasters"	1000	"2014-10-22T23:18:01"
"101"	"Starship X"	24	"2014-08-31T13:14:21"
"102"	"Alien Adventure"	192	"2014-07-12T11:07:56"
"102"	"Galaxy Invaders"	0	"2014-09-18T07:33:42"
"103"	"Attack Ships"	3	"2014-10-19T01:13:24"
"103"	"Galaxy Invaders"	2317	"2014-09-11T06:53:00"
"103"	"Meteor Blasters"	723	"2014-10-19T01:14:24"
"103"	"Starship X"	42	"2014-07-11T06:53:03"

示例：基于Amazon DynamoDB的排行榜

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"
"101"	"Meteor Blasters"	1000	"2014-10-22T23:18:01"
"101"	"Starship X"	24	"2014-08-31T13:14:21"
"102"	"Alien Adventure"	192	"2014-07-12T11:07:56"
"102"	"Galaxy Invaders"	0	"2014-09-18T07:33:42"
"103"	"Attack Ships"	3	"2014-10-19T01:13:24"
"103"	"Galaxy Invaders"	2317	"2014-09-11T06:53:00"
"103"	"Meteor Blasters"	723	"2014-10-19T01:14:24"
"103"	"Starship X"	42	"2014-07-11T06:53:03"

- Hash Key = Primary Key

示例：基于Amazon DynamoDB的排行榜

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"
"101"	"Meteor Blasters"	1000	"2014-10-22T23:18:01"
"101"	"Starship X"	24	"2014-08-31T13:14:21"
"102"	"Alien Adventure"	192	"2014-07-12T11:07:56"
"102"	"Galaxy Invaders"	0	"2014-09-18T07:33:42"
"103"	"Attack Ships"	3	"2014-10-19T01:13:24"
"103"	"Galaxy Invaders"	2317	"2014-09-11T06:53:00"
"103"	"Meteor Blasters"	723	"2014-10-19T01:14:24"
"103"	"Starship X"	42	"2014-07-11T06:53:03"

- Hash Key = Primary Key
- Range Key = Sub Key
- Range Key = Sort Key

示例：基于Amazon DynamoDB的排行榜

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"
"101"	"Meteor Blasters"	1000	"2014-10-22T23:18:01"
"101"	"Starship X"	24	"2014-08-31T13:14:21"
"102"	"Alien Adventure"	192	"2014-07-12T11:07:56"
"102"	"Galaxy Invaders"	0	"2014-09-18T07:33:42"
"103"	"Attack Ships"	3	"2014-10-19T01:13:24"
"103"	"Galaxy Invaders"	2317	"2014-09-11T06:53:00"
"103"	"Meteor Blasters"	723	"2014-10-19T01:14:24"
"103"	"Starship X"	42	"2014-07-11T06:53:03"

- Hash Key = Primary Key
- Range Key = Sub Key
- Range Key = Sort Key
- 其他属性不需要定义

示例：基于Amazon DynamoDB的排行榜

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"
"101"	"Meteor Blasters"	1000	"2014-10-22T23:18:01"
"101"	"Starship X"	24	"2014-08-31T13:14:21"
"102"	"Alien Adventure"	192	"2014-07-12T11:07:56"
"102"	"Galaxy Invaders"	0	"2014-09-18T07:33:42"
"103"	"Attack Ships"	3	"2014-10-19T01:13:24"
"103"	"Galaxy Invaders"	2317	"2014-09-11T06:53:00"
"103"	"Meteor Blasters"	723	"2014-10-19T01:14:24"
"103"	"Starship X"	42	"2014-07-11T06:53:03"

- Hash Key = Primary Key
- Range Key = Sub Key
- Range Key = Sort Key
- 其他属性不需要定义

示例：基于Amazon DynamoDB的排行榜

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"
"101"	"Meteor Blasters"	1000	"2014-10-22T23:18:01"
"101"	"Starship X"	24	"2014-08-31T13:14:21"
"102"	"Alien Adventure"	192	"2014-07-12T11:07:56"
"102"	"Galaxy Invaders"	0	"2014-09-18T07:33:42"
"103"	"Attack Ships"	3	"2014-10-19T01:13:24"
"103"	"Galaxy Invaders"	2317	"2014-09-11T06:53:00"
"103"	"Meteor Blasters"	723	"2014-10-19T01:14:24"
"103"	"Starship X"	42	"2014-07-11T06:53:03"

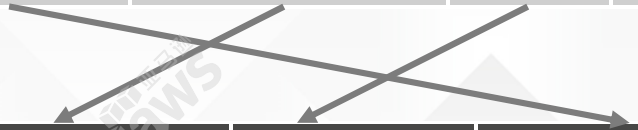
- Hash Key = Primary Key
- Range Key = Sub Key
- Range Key = Sort Key
- 其他属性不需要定义
- 那么，如何按照最高分进行全局排序呢？

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"

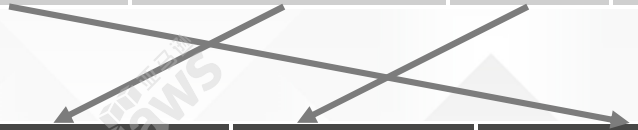


BoardName (hash key)	TopScore (range key)	UserID
"Alien Adventure"	192	"101"
"Attack Ships"	3	"103"
"Galaxy Invaders"	0	"102"
"Galaxy Invaders"	2317	"103"
"Galaxy Invaders"	5842	"101"
"Meteor Blasters"	723	"103"
"Meteor Blasters"	1000	"101"
"Starship X"	24	"101"
"Starship X"	42	"103"

- 创建第二索引

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"

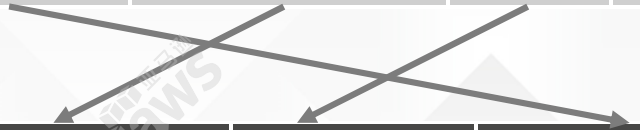


BoardName (hash key)	TopScore (range key)	UserID
"Alien Adventure"	192	"101"
"Attack Ships"	3	"103"
"Galaxy Invaders"	0	"102"
"Galaxy Invaders"	2317	"103"
"Galaxy Invaders"	5842	"101"
"Meteor Blasters"	723	"103"
"Meteor Blasters"	1000	"101"
"Starship X"	24	"101"
"Starship X"	42	"103"

- 创建第二索引
- 选择BoardName为Hash Key

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"

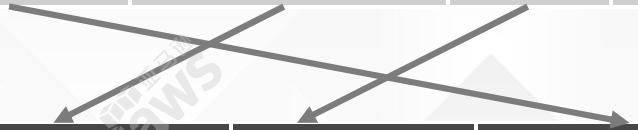


BoardName (hash key)	TopScore (range key)	UserID
"Alien Adventure"	192	"101"
"Attack Ships"	3	"103"
"Galaxy Invaders"	0	"102"
"Galaxy Invaders"	2317	"103"
"Galaxy Invaders"	5842	"101"
"Meteor Blasters"	723	"103"
"Meteor Blasters"	1000	"101"
"Starship X"	24	"101"
"Starship X"	42	"103"

- 创建第二索引
- 选择BoardName为Hash Key
- 选择TopScore为Range Key

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"

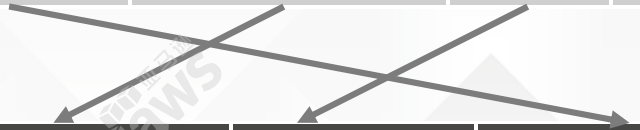


BoardName (hash key)	TopScore (range key)	UserID
"Alien Adventure"	192	"101"
"Attack Ships"	3	"103"
"Galaxy Invaders"	0	"102"
"Galaxy Invaders"	2317	"103"
"Galaxy Invaders"	5842	"101"
"Meteor Blasters"	723	"103"
"Meteor Blasters"	1000	"101"
"Starship X"	24	"101"
"Starship X"	42	"103"

- 创建第二索引
- 选择BoardName为Hash Key
- 选择TopScore为Range Key
- 如果有需要，添加其他的属性

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"

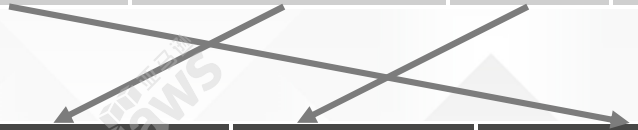


BoardName (hash key)	TopScore (range key)	UserID
"Alien Adventure"	192	"101"
"Attack Ships"	3	"103"
"Galaxy Invaders"	0	"102"
"Galaxy Invaders"	2317	"103"
"Galaxy Invaders"	5842	"101"
"Meteor Blasters"	723	"103"
"Meteor Blasters"	1000	"101"
"Starship X"	24	"101"
"Starship X"	42	"103"

- 创建第二索引
- 选择BoardName为Hash Key
- 选择TopScore为Range Key
- 如果有需要，添加其他的属性
- 现在就可以通过BoardName查询，并按照TopScore进行排序了

利用第二索引

UserID (hash key)	BoardName (range key)	TopScore	TopScoreDate
"101"	"Galaxy Invaders"	5842	"2014-09-15T17:24:31"



BoardName (hash key)	TopScore (range key)	UserID
"Alien Adventure"	192	"101"
"Attack Ships"	3	"103"
"Galaxy Invaders"	0	"102"
"Galaxy Invaders"	2317	"103"
"Galaxy Invaders"	5842	"101"
"Meteor Blasters"	723	"103"
"Meteor Blasters"	1000	"101"
"Starship X"	24	"101"
"Starship X"	42	"103"

- 创建第二索引
- 选择BoardName为Hash Key
- 选择TopScore为Range Key
- 如果有需要，添加其他的属性
- 现在就可以通过BoardName查询，并按照TopScore进行排序了
- 适用于很多常见的游戏场景

多玩家游戏服务端架构

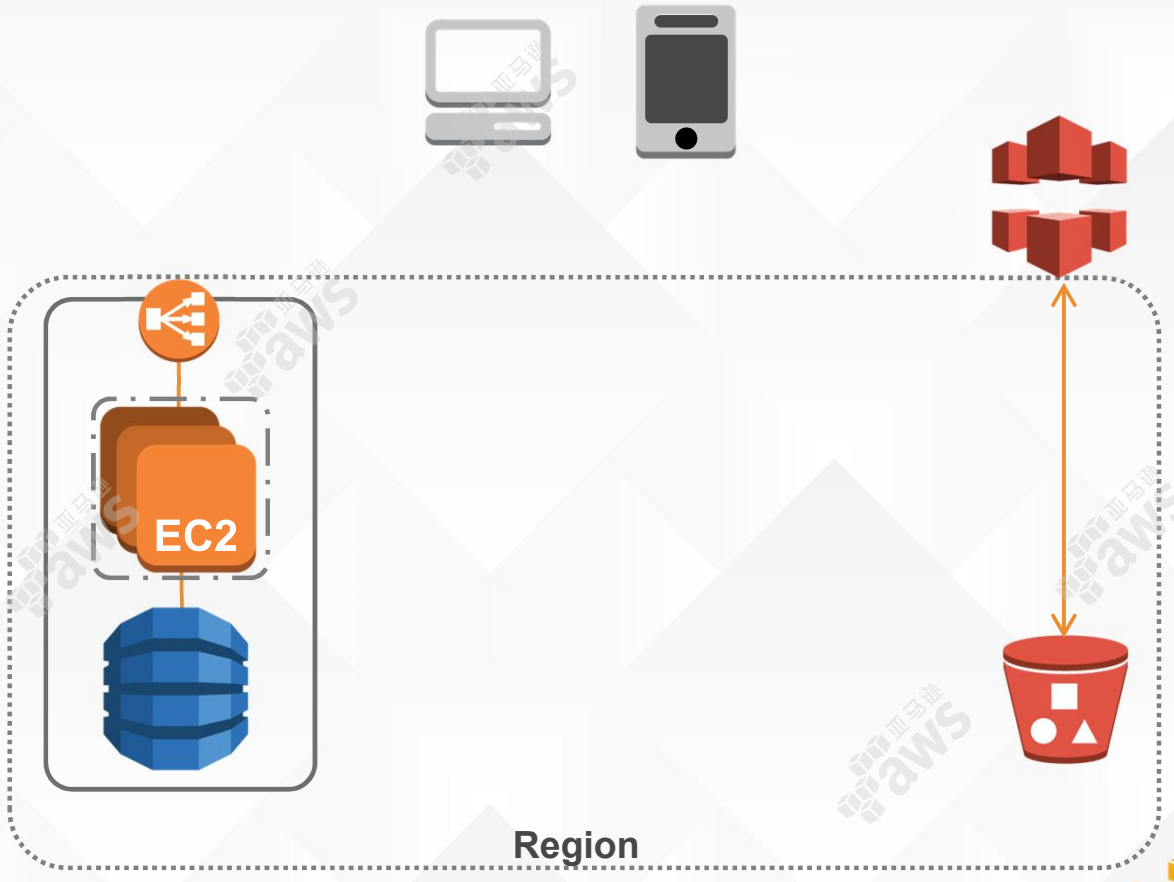
Region

多玩家游戏服务端架构

- 后端API服务
  - 核心Session
  - 匹配

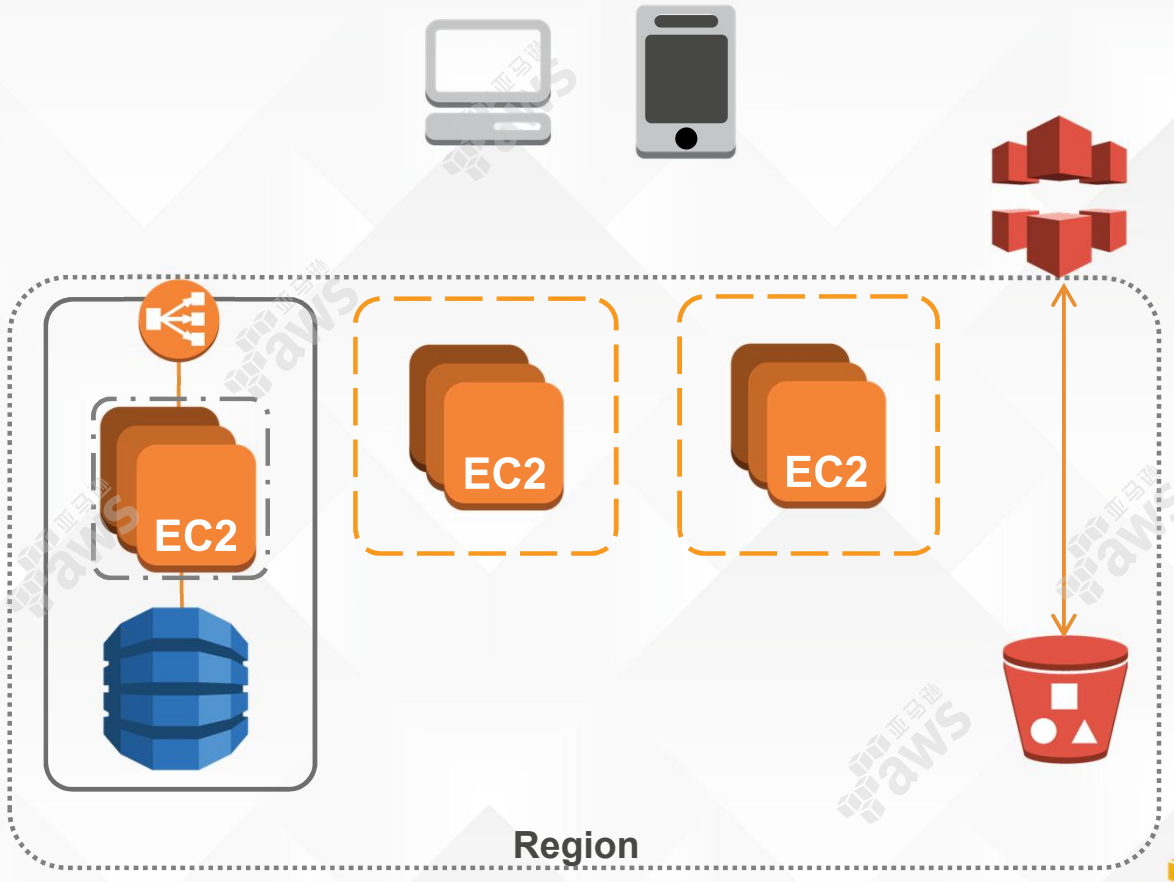
多玩家游戏服务端架构

- 后端API服务
  - 核心Session
  - 匹配
- S3 + CloudFront
  - DLC, assets
  - 游戏存档
  - 用户创建内容

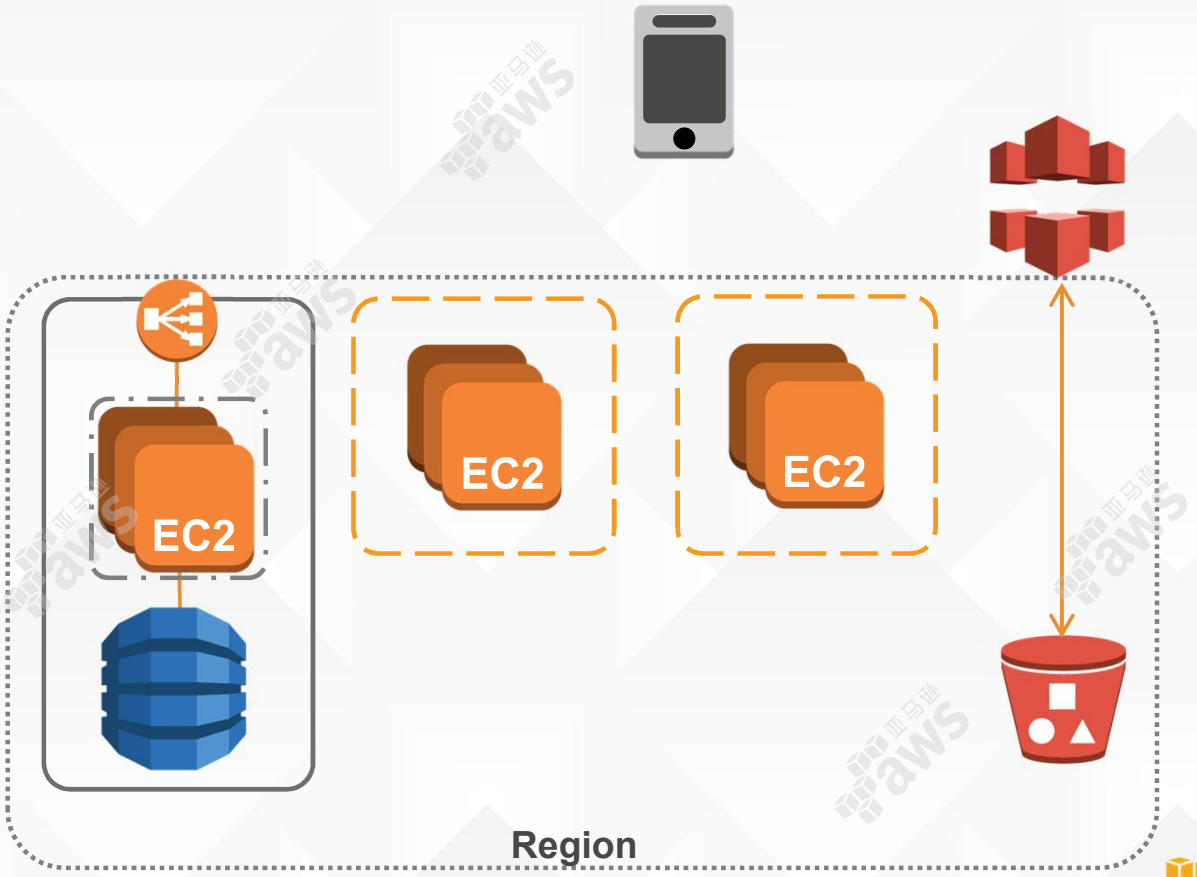


多玩家游戏服务端架构

- 后端API服务
  - 核心Session
  - 匹配
- S3 + CloudFront
  - DLC, assets
  - 游戏存档
  - 用户创建内容
- 游戏服务器
  - 客户端Socket
  - 根据玩家数量扩展

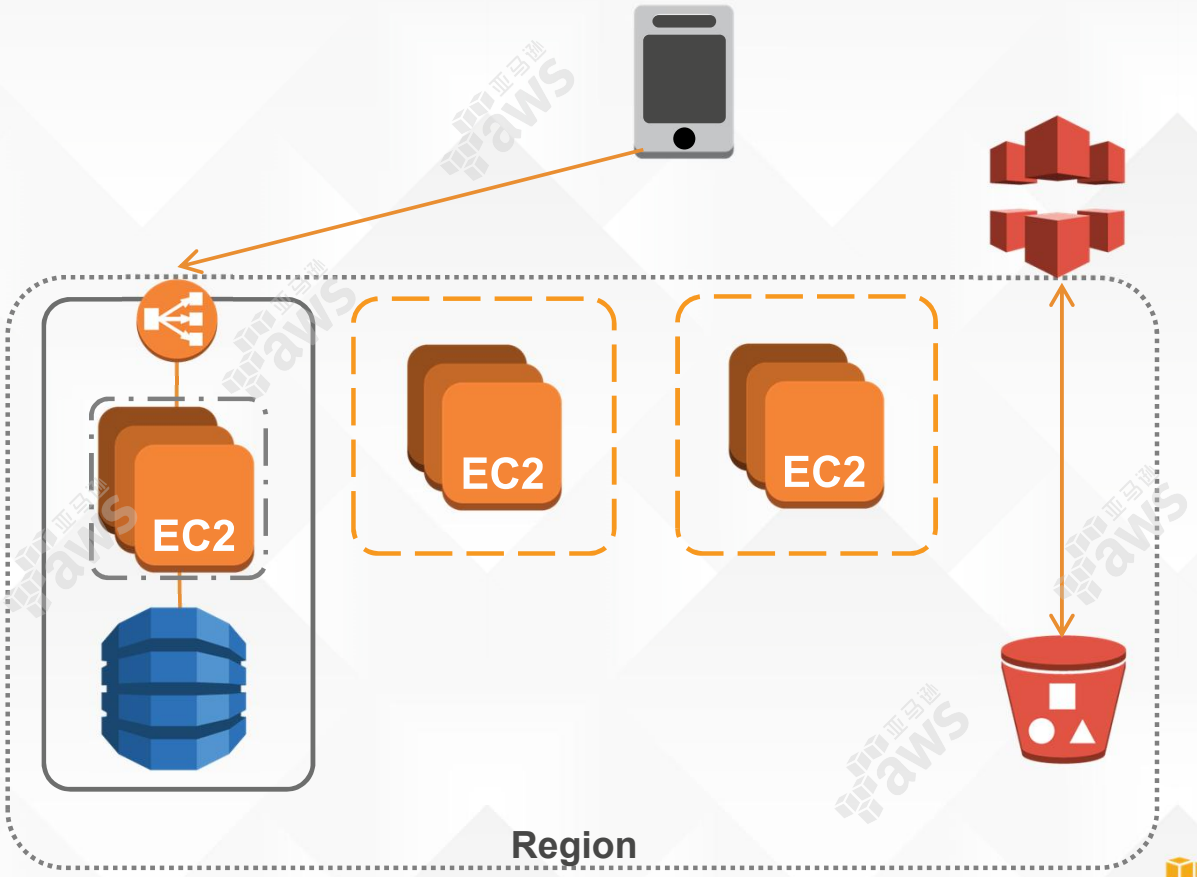


多玩家游戏服务端架构



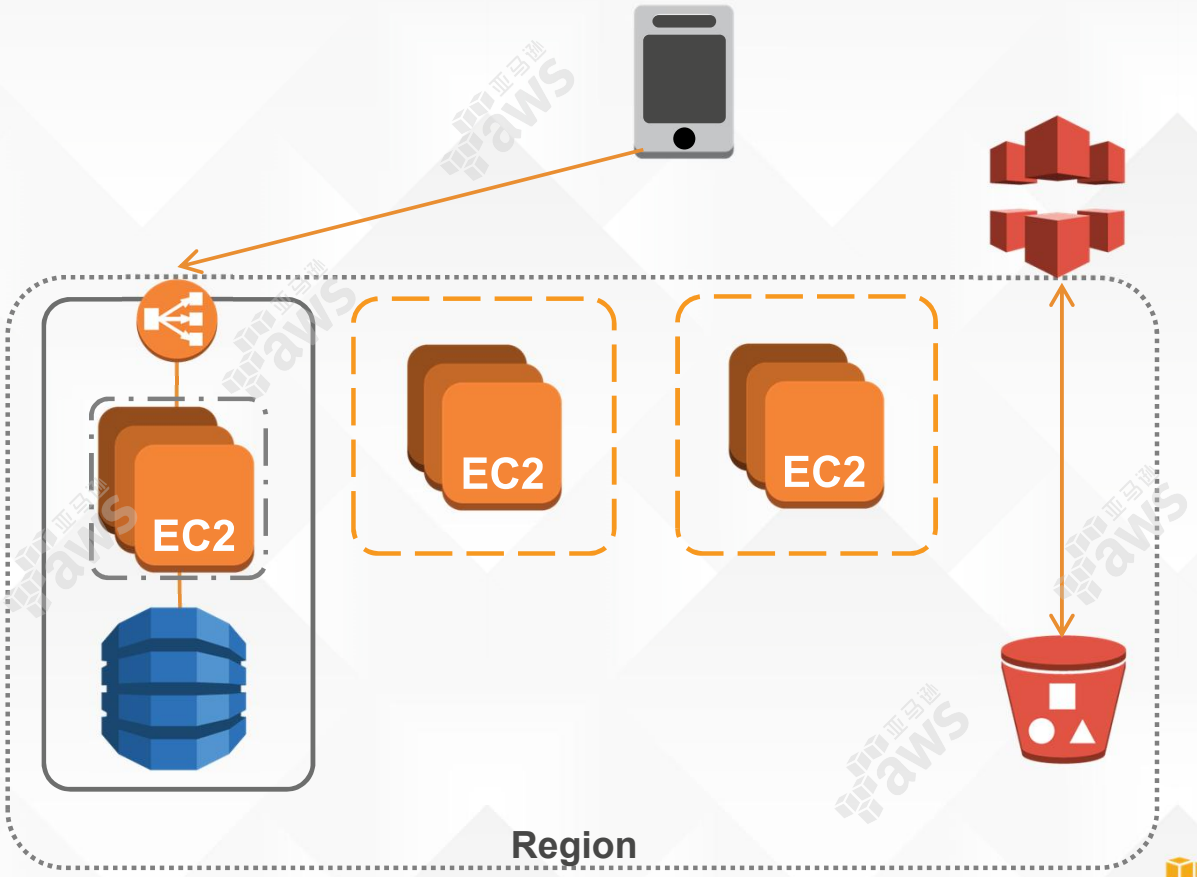
多玩家游戏服务端架构

1) 基于API登录



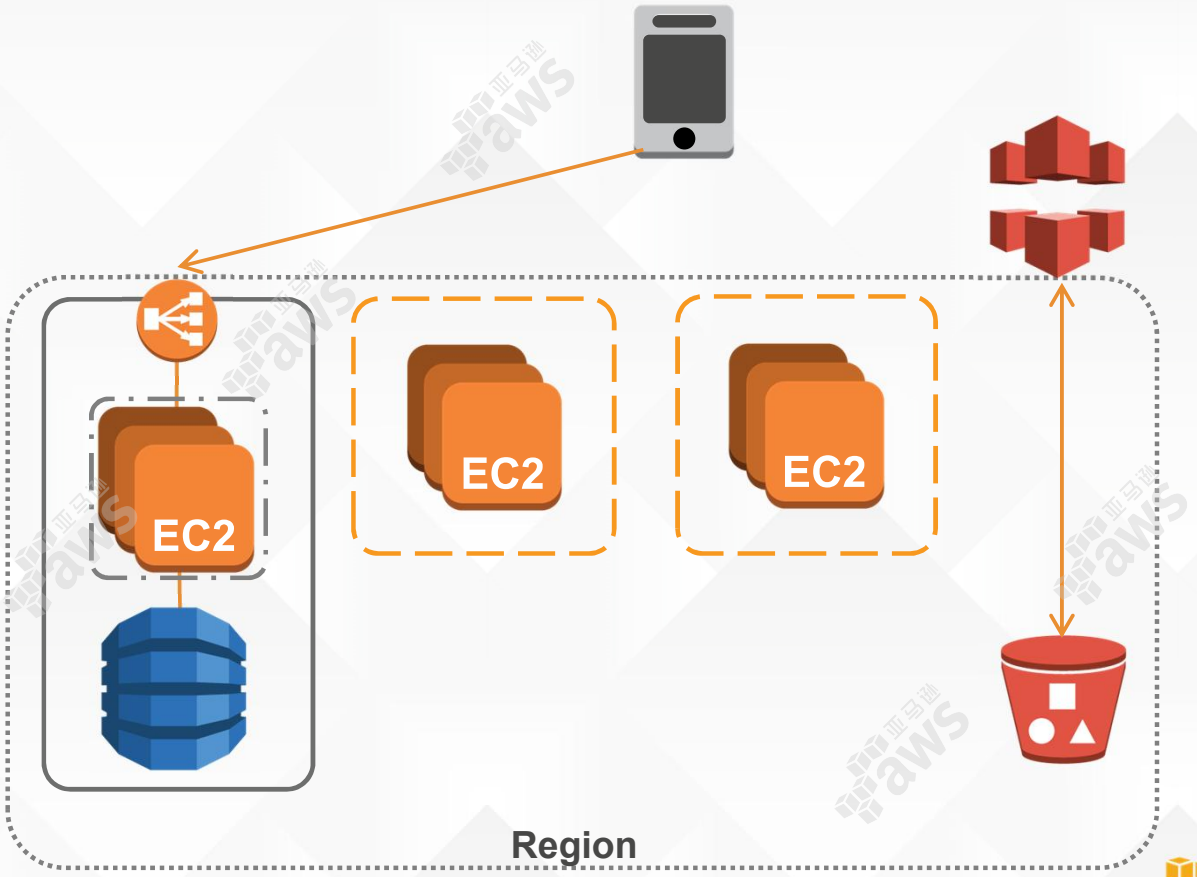
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配



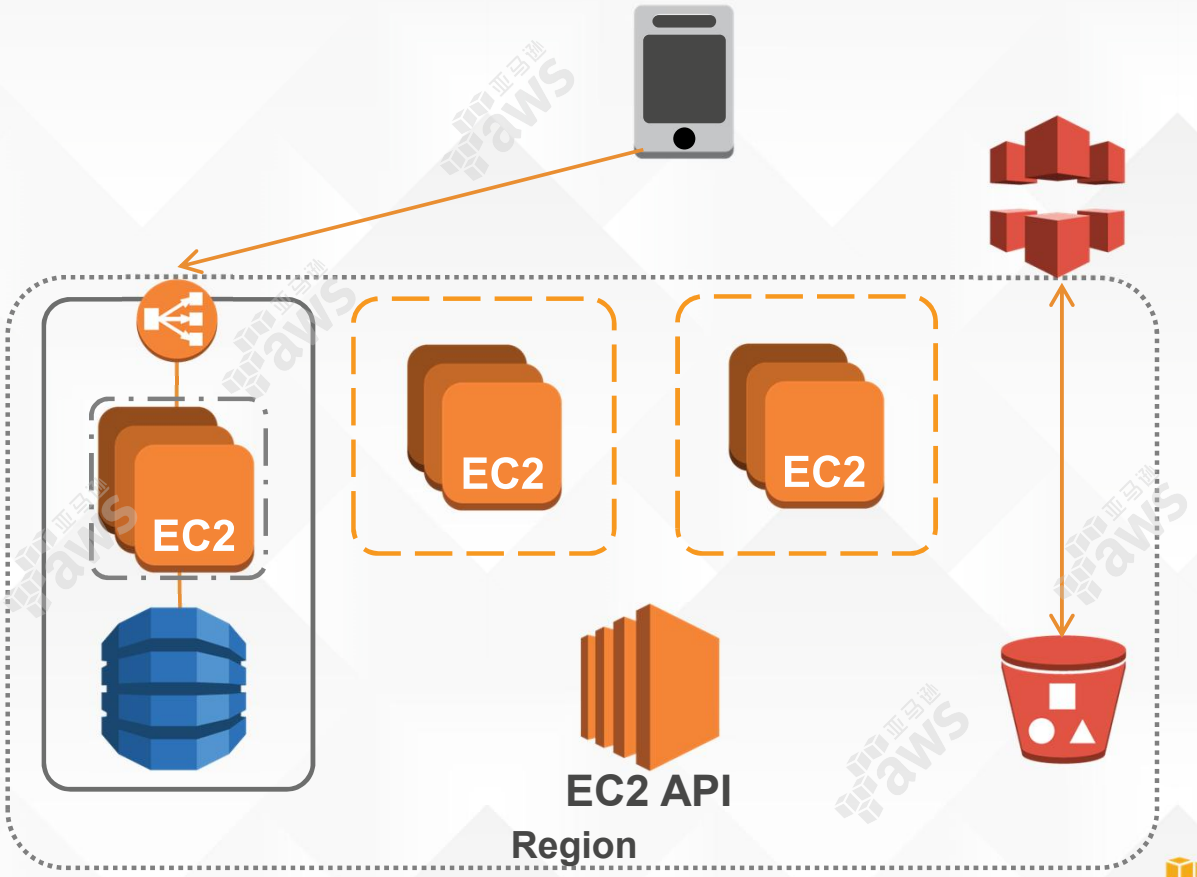
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP



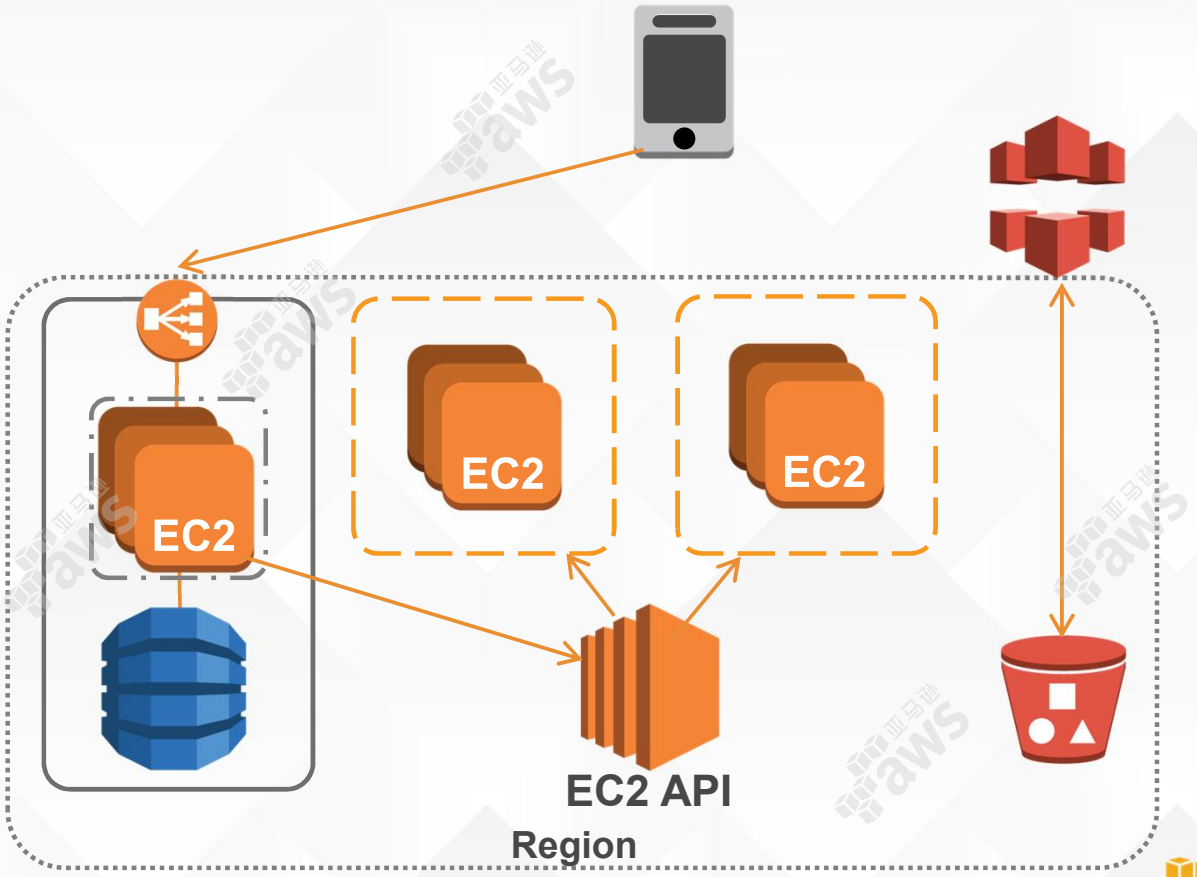
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP



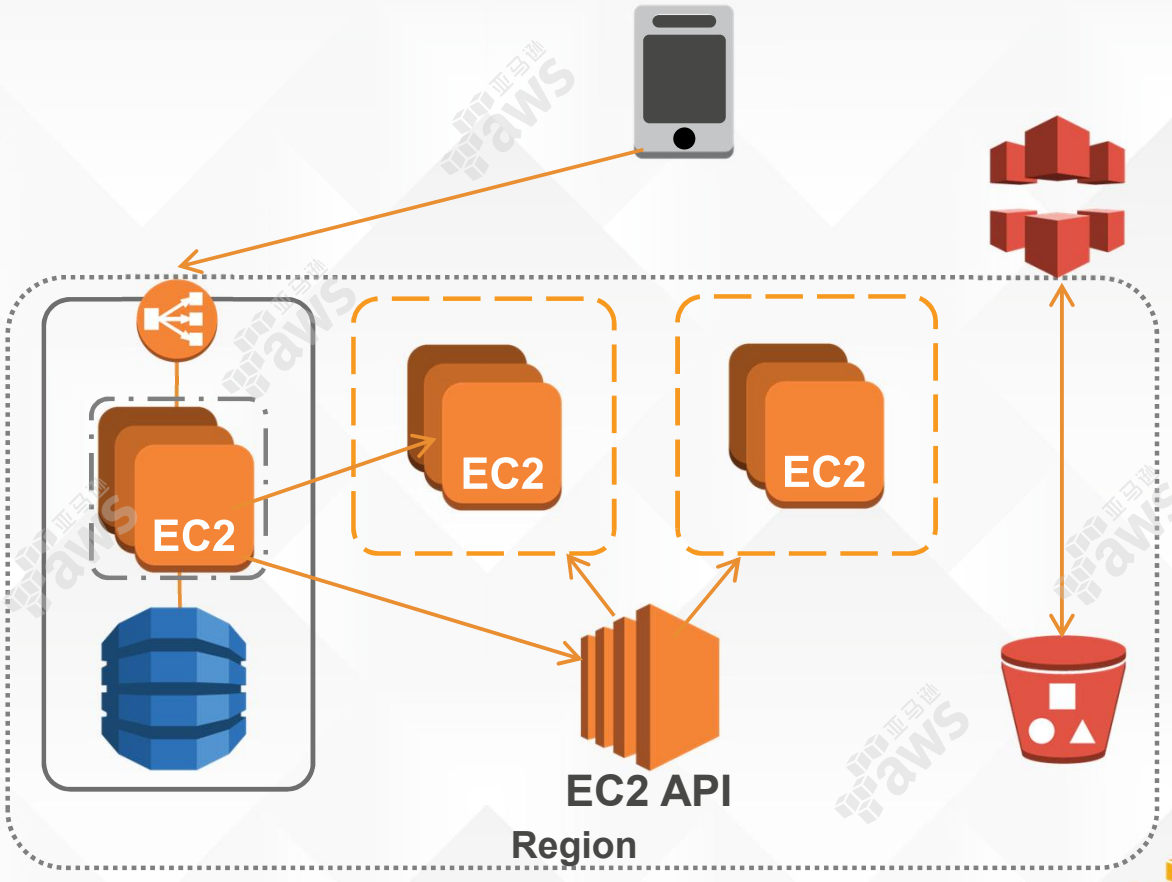
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP



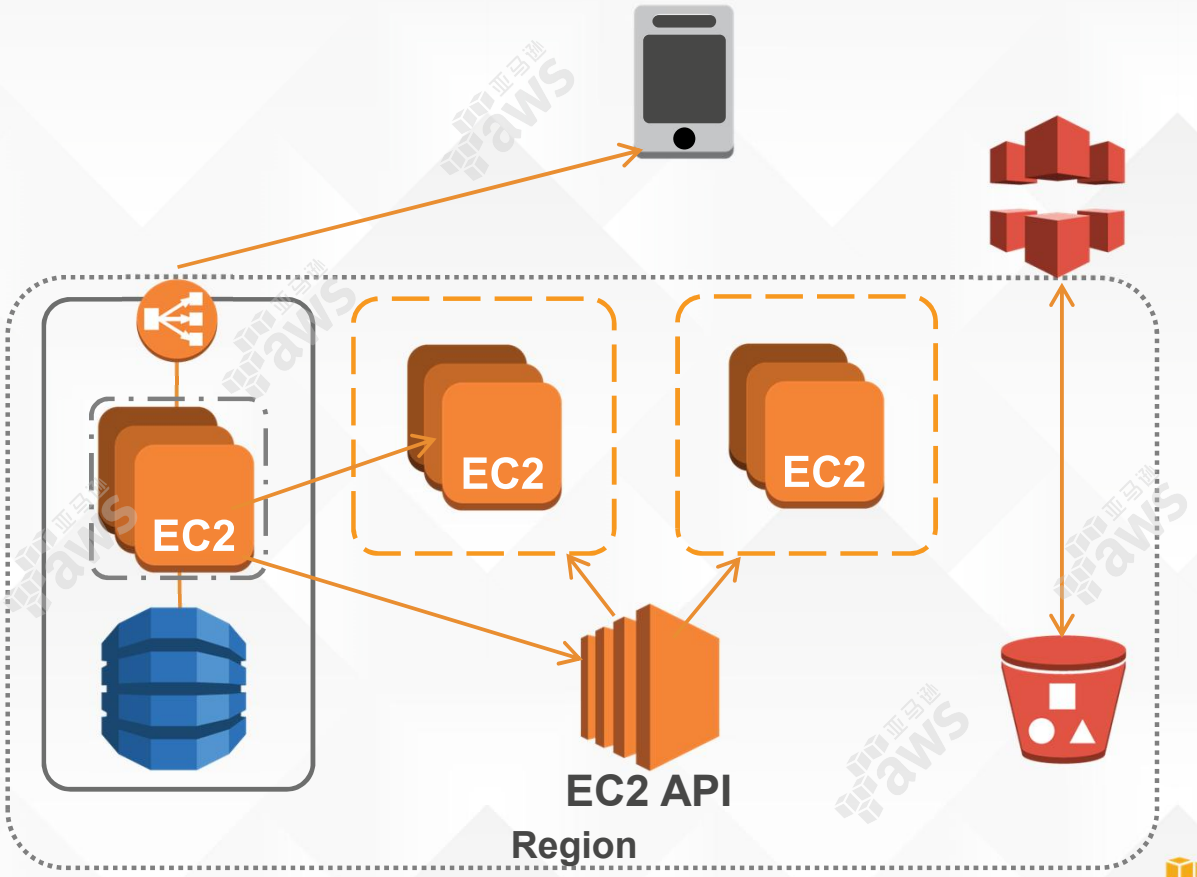
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP



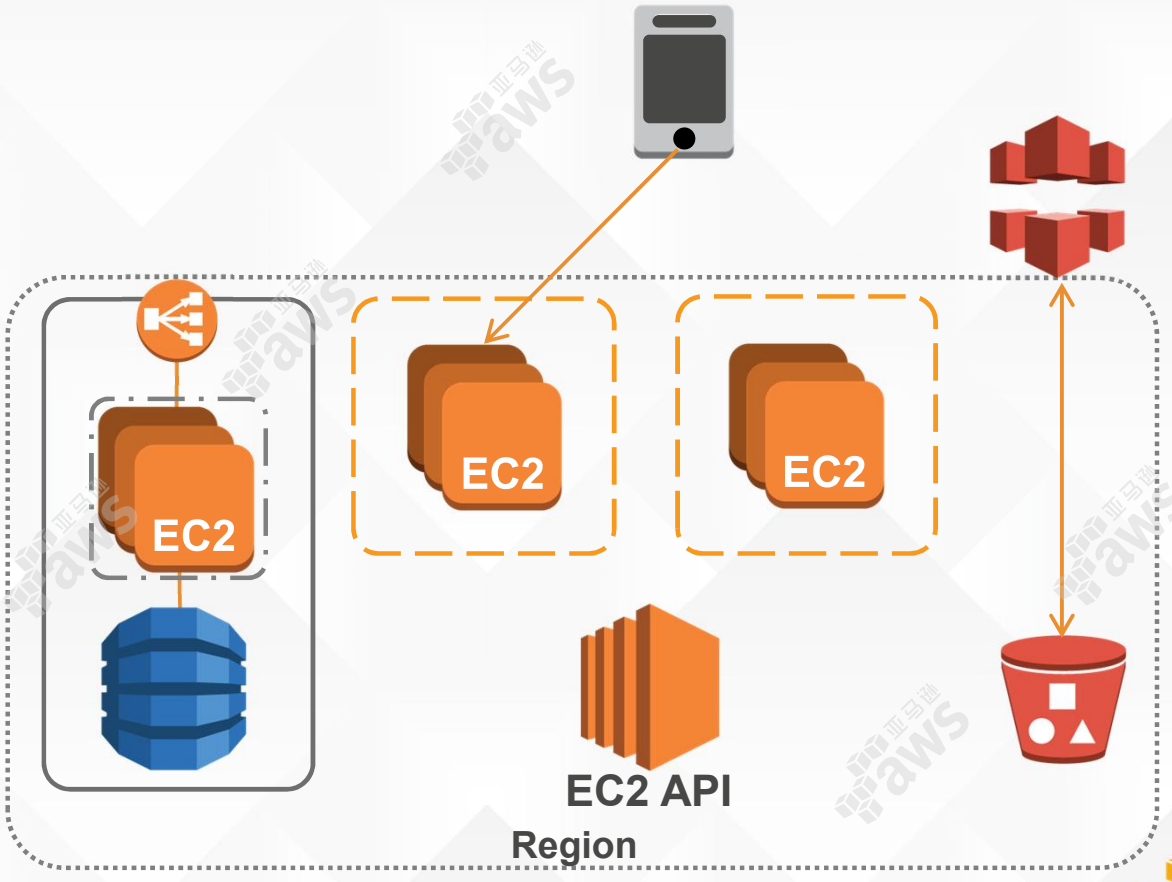
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP



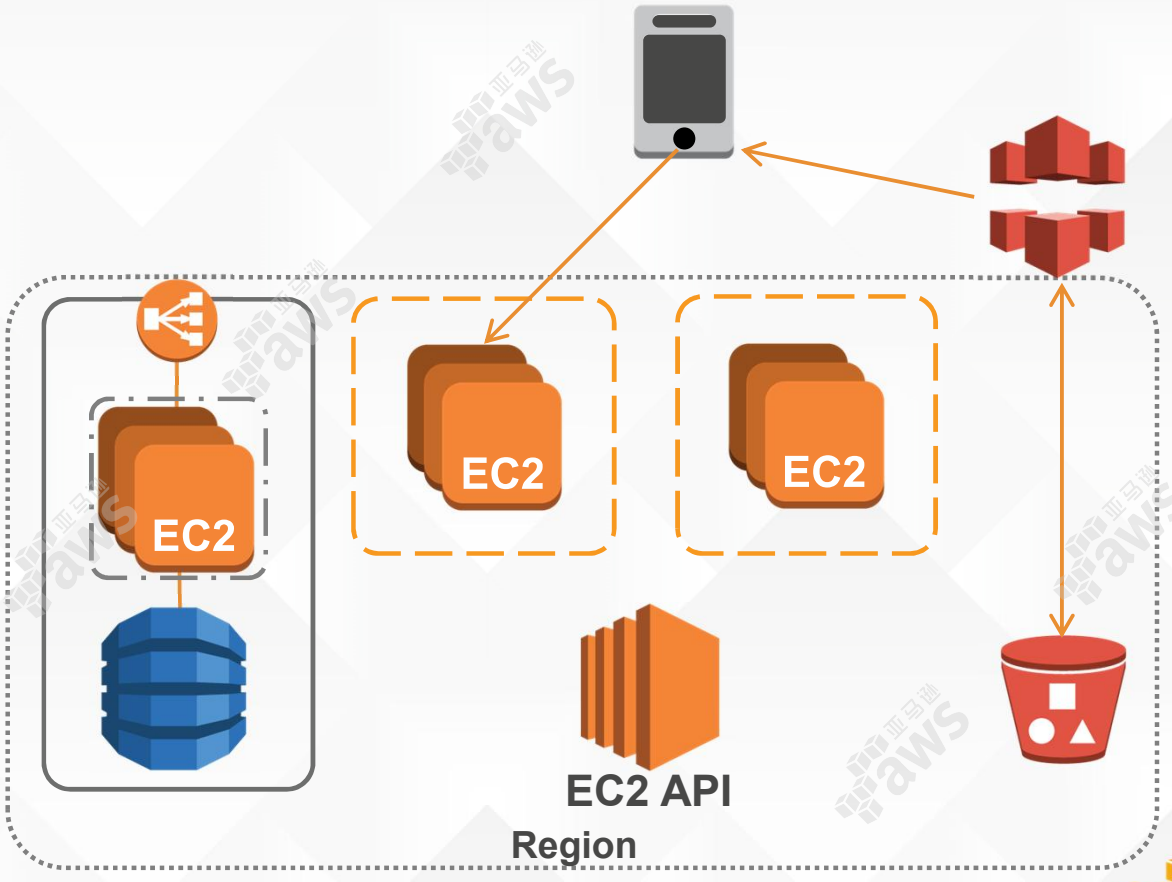
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP
4) 连接到游戏服务器



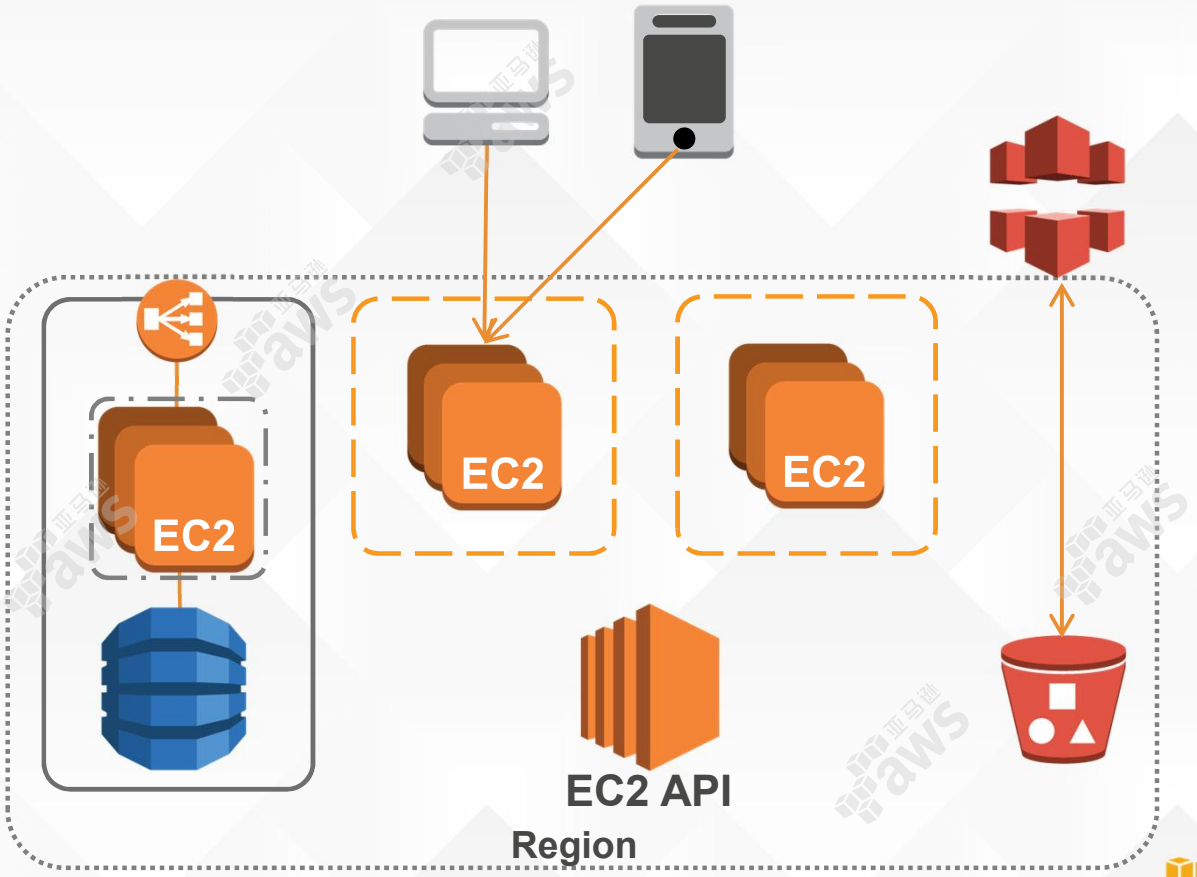
多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP
4) 连接到游戏服务器
5) 获取静态资源

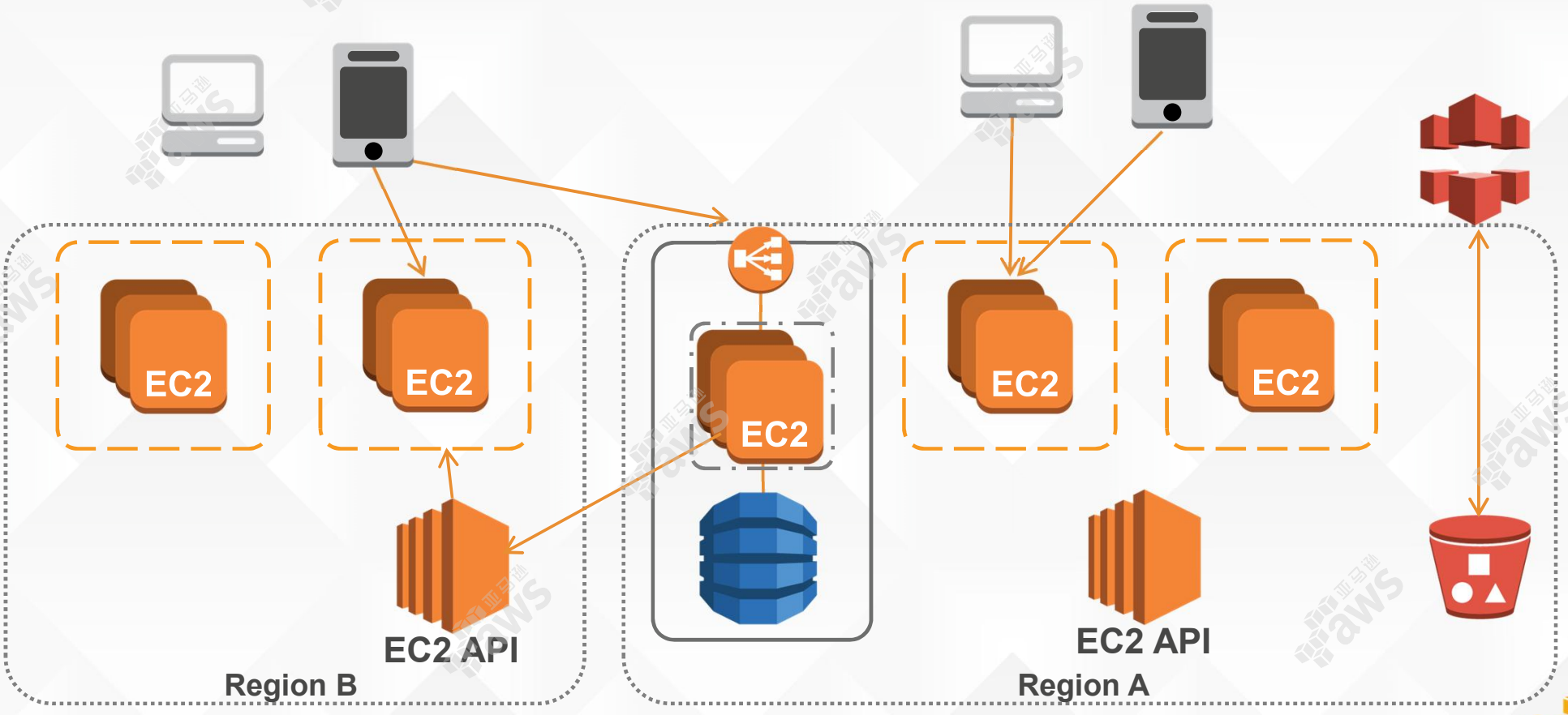


多玩家游戏服务端架构

1) 基于API登录
2) 请求匹配
3) 获取游戏服务器IP
4) 连接到游戏服务器
5) 获取静态资源
6) 其他玩家加入



多玩家游戏服务端架构



Thank You