



云开发 CloudBase

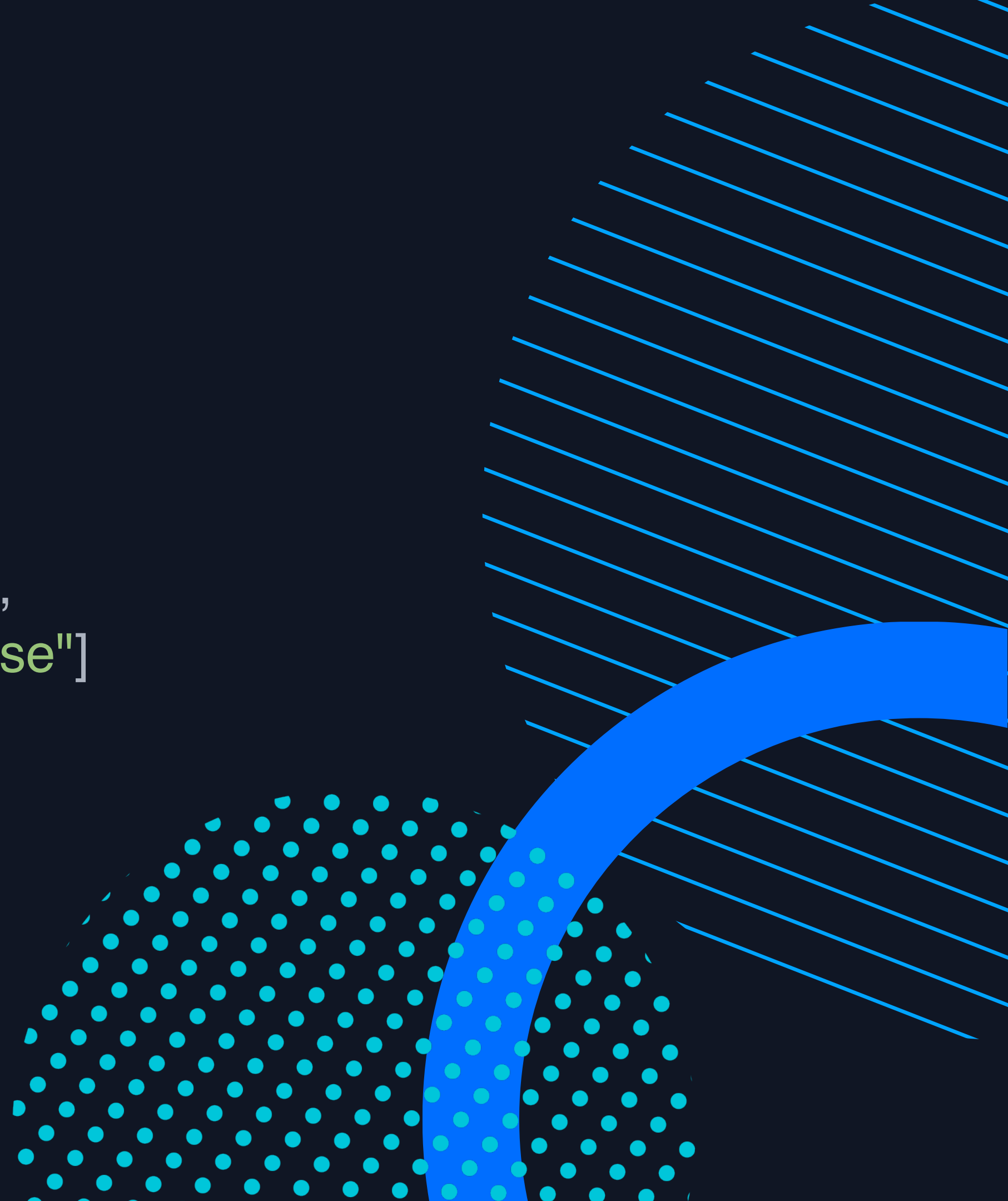
云原生一体化服务的设计与实践

王伟嘉 / Starkwang

Tencent Live Conference

2020.09 Shenzhen

```
{  
  name:      "王伟嘉",  
  nickname:   "Starkwang",  
  working_in: "Tencent Cloud",  
  tags:       ["Node.js Core Collaborator"],  
  products:   ["小程序·云开发", "CloudBase"]  
}
```





云时代的前端开发





前端工程师都在干什么？

画网页

网站

hybrid 页面

小程序

写 Node.js

接入层 / BFF

服务端渲染 / SSR

基础设施

日志、打点上报

监控告警

灰度

画网页

曾经

回源服务器

自建CDN

发布系统

域名解析

云时代



CDN



对象存储



静态托管

写 Node.js

曾经

自建服务器

服务发现

伸缩容

日志、监控

运行时

云时代

云主机

Docker

Kubernetes

Istio



多地容灾

数据库

CDN

集群灰度

日志

对象存储

HTTPS证书

Kubernetes

API网关

FaaS

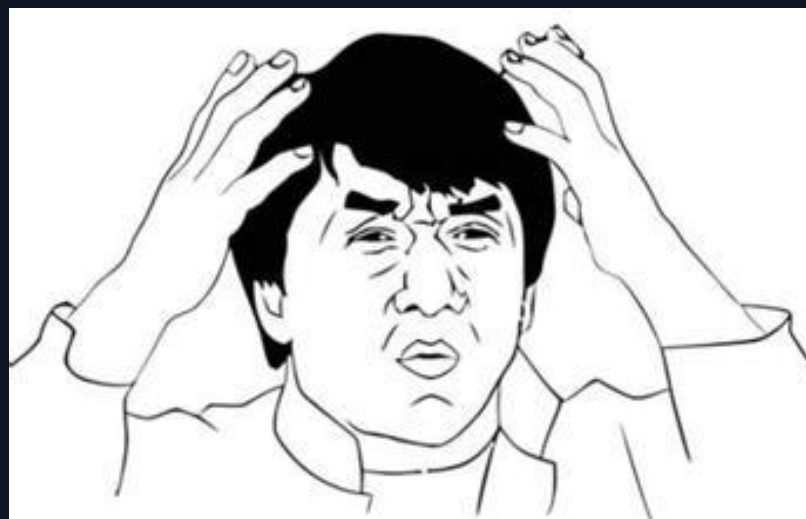
域名

负载均衡

监控告警

基础镜像优化

DevOps





云开发 CloudBase

为前端工程师提供云时代的基础设施



后端即服务

云数据库

云存储

云函数



```
const cloudbase = require('@cloudbase/js-sdk')
const app = cloudbase.init({ env: '...' })
const db = app.database()

const data = await db.collection('users').where({
  name: 'Tom'
}).get()
// => { _id: '...', name: 'Tom' }
```



云开发 CloudBase

为前端工程师提供云时代的基础设施

托管

静态文件托管

函数托管

SSR 应用托管

Docker 托管



```
$ npm install -g @cloudbase/cli
```

```
$ echo "<p>Hello CloudBase!</p>" > index.html
```

```
$ cloudbase hosting:deploy index.html
```



云开发 CloudBase

为前端工程师提供云时代的基础设施

托管

静态文件托管

函数托管

Web应用托管

容器托管



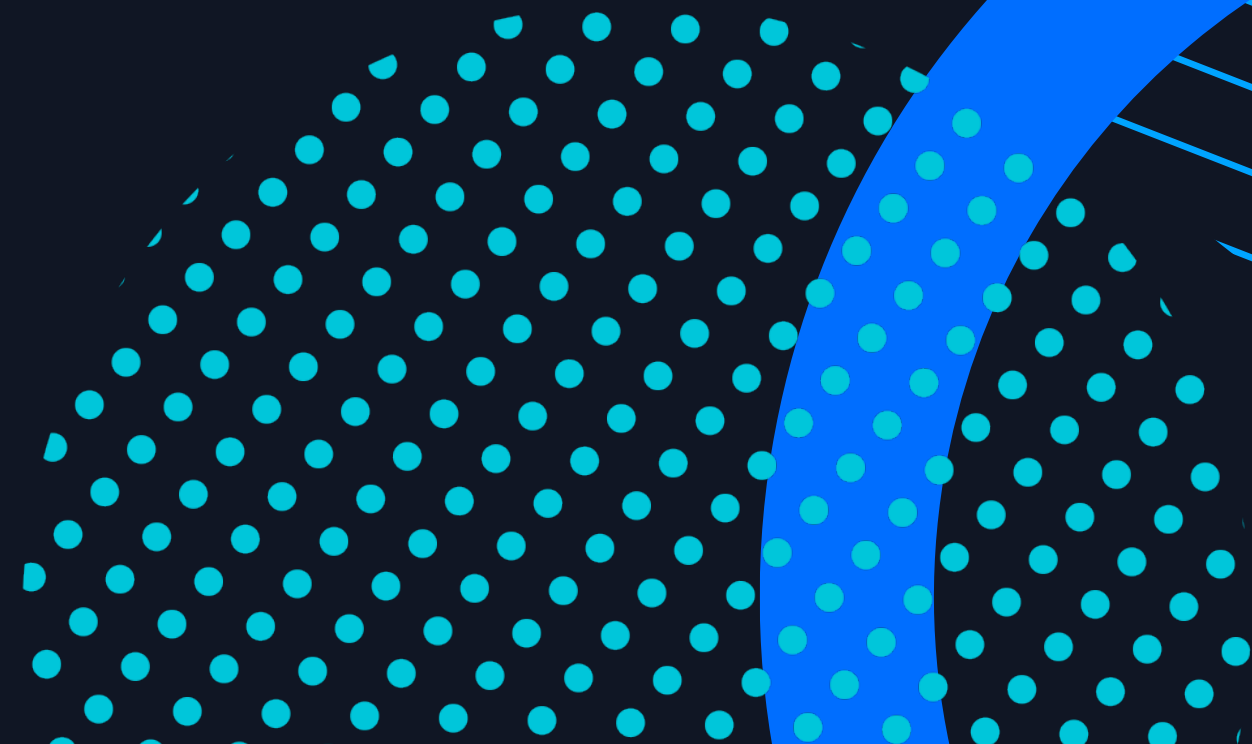
```
$ npm install -g @cloudbase/cli
```

```
$ echo "<p>Hello CloudBase!</p>" > index.html
```

```
$ cloudbase hosting:deploy index.html
```



BaaS 后台即服务





```
const cloudbase = require('@cloudbase/js-sdk')
const app = cloudbase.init({ env: '...' })
const db = app.database()

const data = await db.collection('users').where({
  name: 'Tom'
}).get()
// => { _id: '...', name: 'Tom' }
```

无论谁都可以使用云资源吗？

BaaS权限控制

你是谁？

用户端鉴权

你可以访问什么资源？

安全规则



```
{  
  // 所有登录用户可读  
  "read": "auth ≠ null",  
  
  // 所有登录用户可以发帖  
  "create": "auth ≠ null",  
  
  // 用户只能修改自己的帖子  
  "update": "doc.userID = auth.openid",  
  
  // 用户只能删除自己的帖子  
  "delete": "doc.userID = auth.openid"  
}
```

云数据库的架构


SDK → CloudBase → 底层数据库

如何传输数据？



序列化协议

JSON

Number	123
String	"hello"
Boolean	true
Null	null
...	
Date	
Float	
GEO Point	

JSON 存在局限性，无法承载某些数据类型

Extended JSON

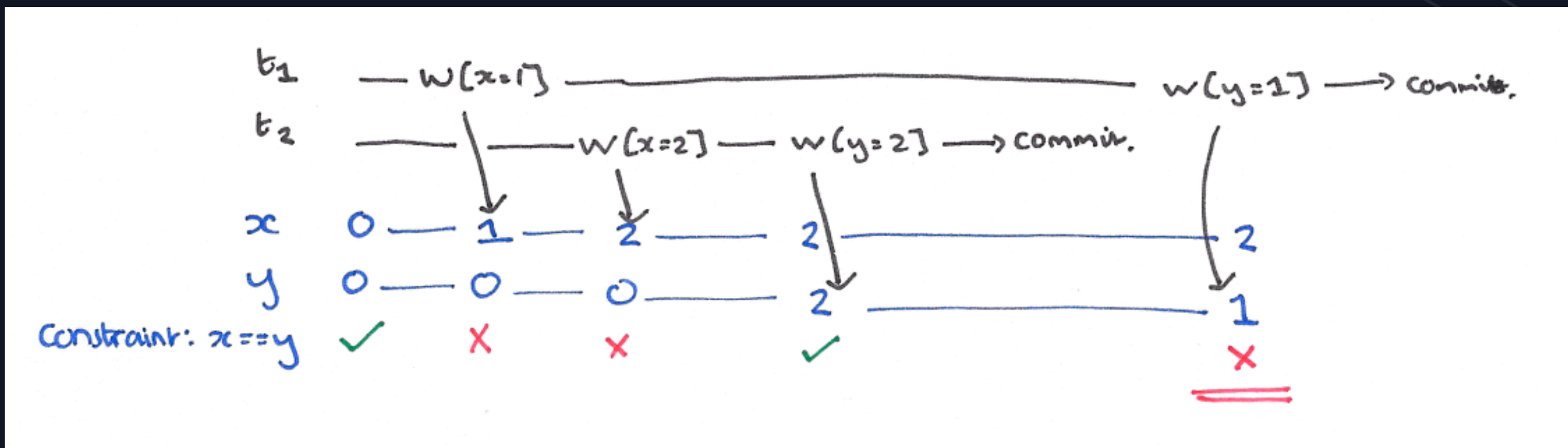


```
{  
  int: 123,  
  double: 123.4,  
  date: new Date(),  
  regexp: /abcd/  
}
```



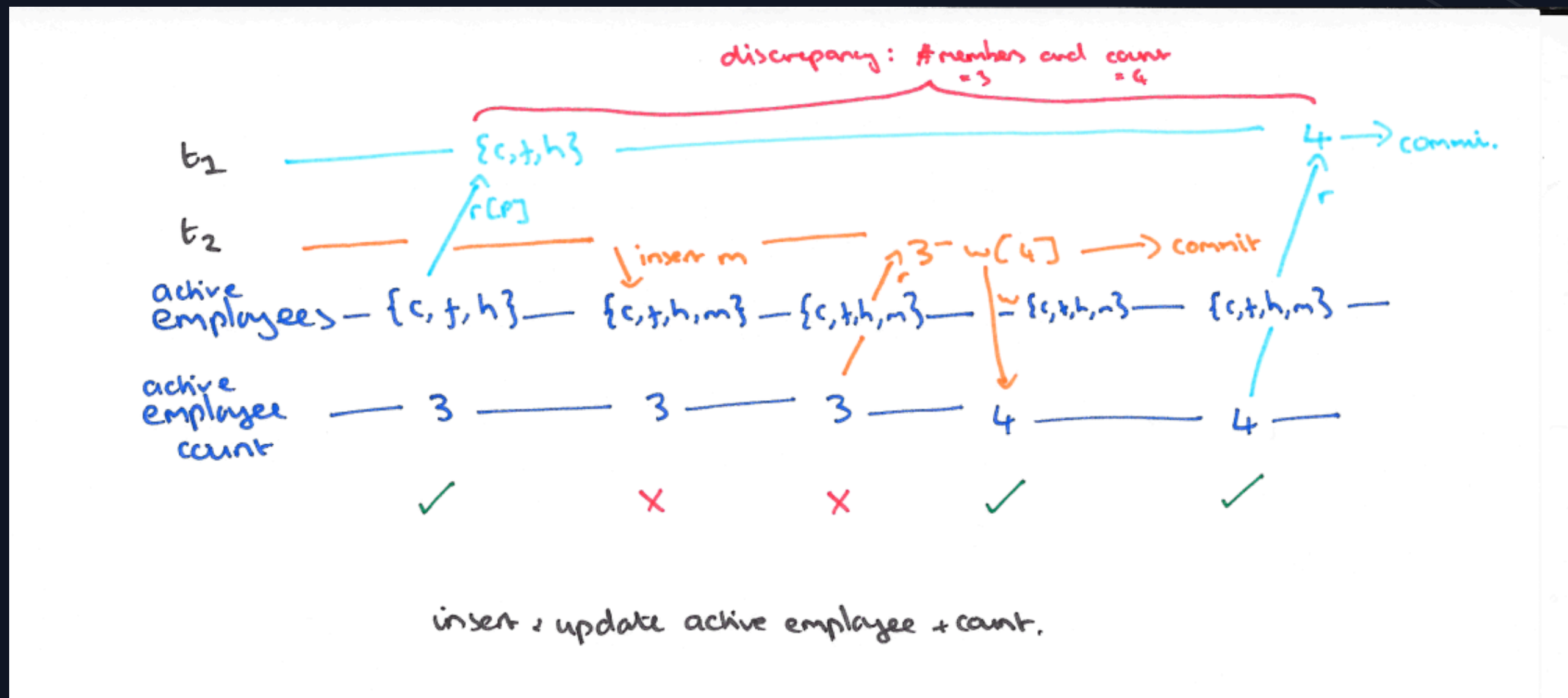
```
{  
  "int": {  
    "$numberInt": "123"  
  },  
  "double": {  
    "$numberDouble": "123.4"  
  },  
  "date": {  
    "$date": {  
      "$numberLong": "1594037126323"  
    }  
  },  
  "regexp": {  
    "$regularExpression": {  
      "pattern": "abcd",  
      "options": ""  
    }  
  }  
}
```

事务



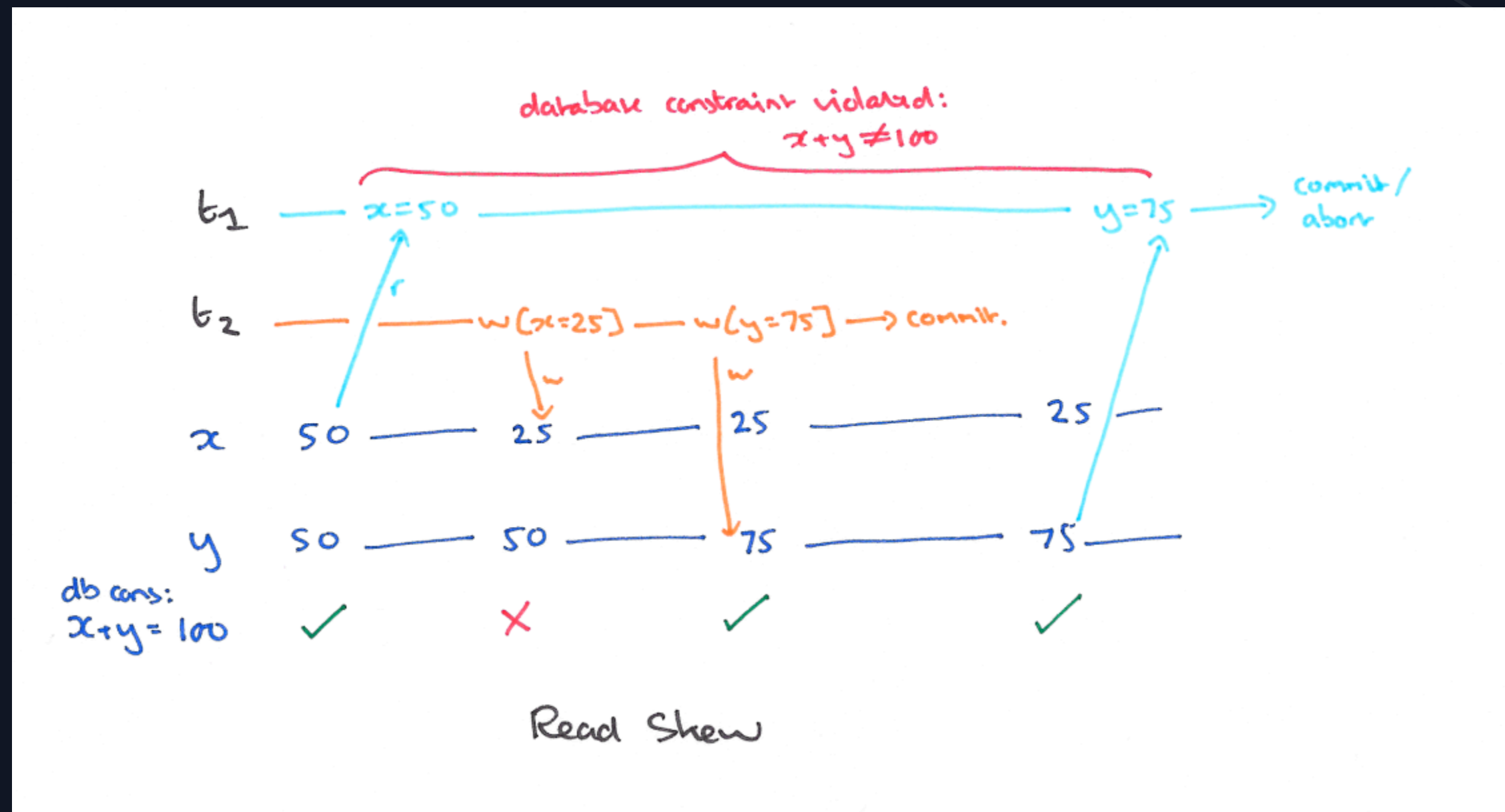
事务的隔离性

幻读 (Phantom Reads)

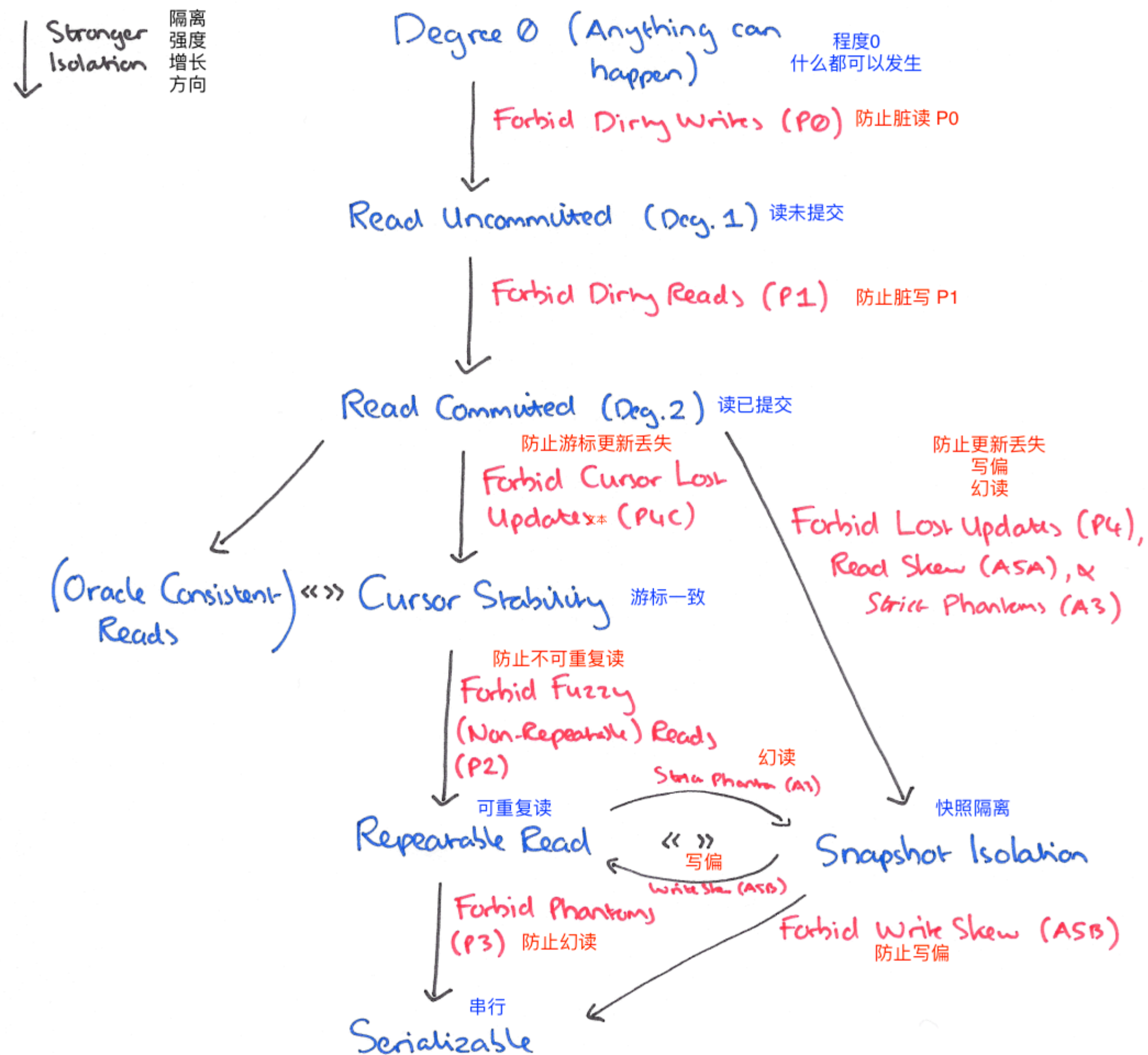


事务的隔离性

读倾斜 (Read Skew)



隔离性级别





```
const db = cloud.database()

db.runTransaction(async t => {
  const doc = t.collection('goods').doc('apple')

  const { data: price } = await doc.get()

  if (apple.amount === 0) {
    throw new Error('没有足够的苹果! ')
  }

  await doc.update({
    amount: apple.amount - 1
  })
})
```


实时推送



```
const db = wx.cloud.database()

db.collection('messages')
  .where({
    roomId: 1234
  })
  .watch({
    onSnapshot(docs) {
      console.log(docs)
    }
  })
```


实时推送可能遇到的问题



断连



超时

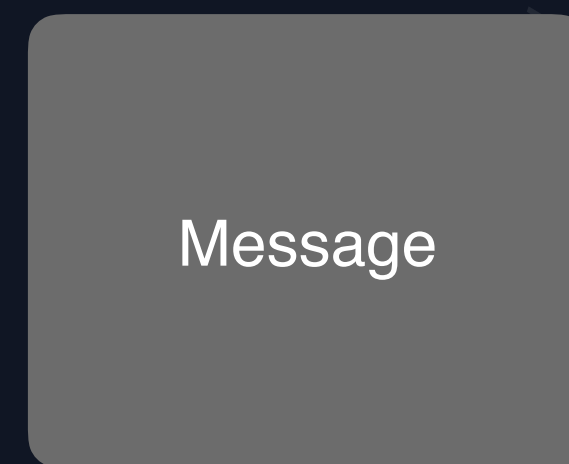
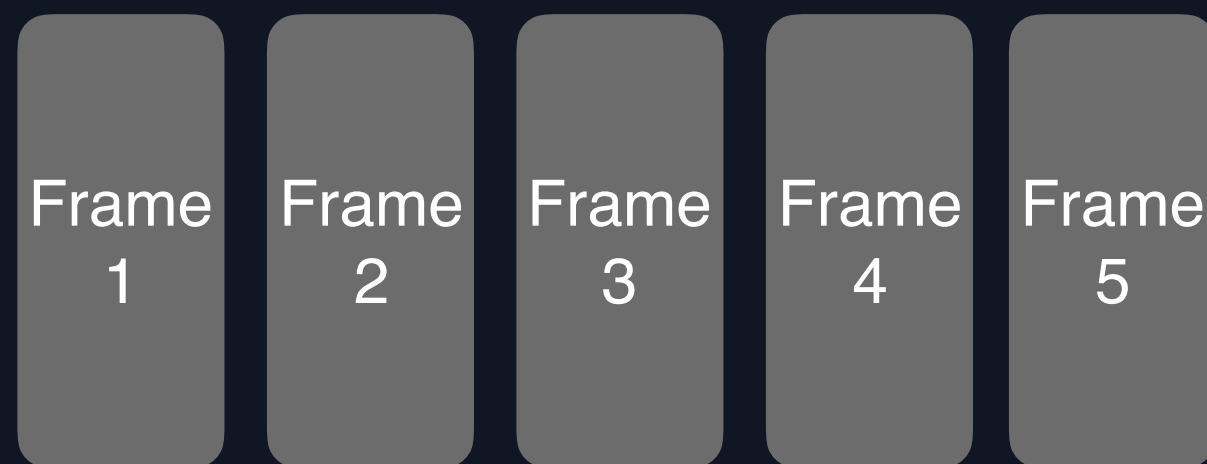


丢包

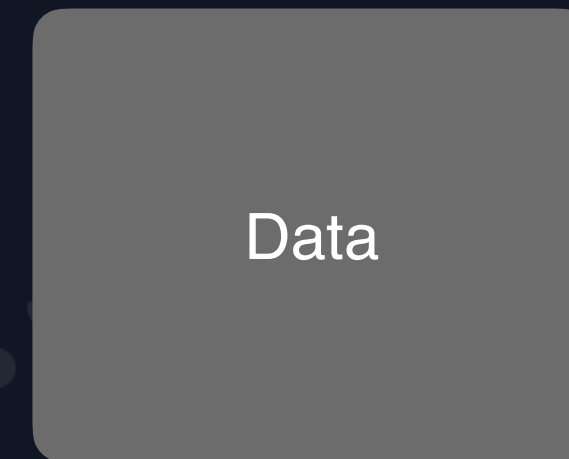
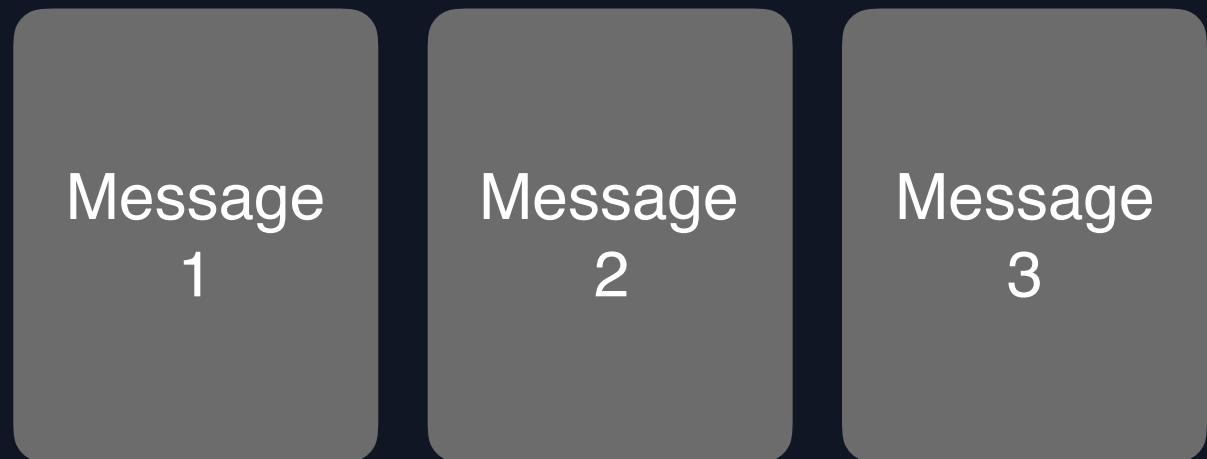


乱序

为什么会乱序



Websocket的最小单元



应用数据的最小单元

为什么会乱序

理想中的



实际上的

自增序号



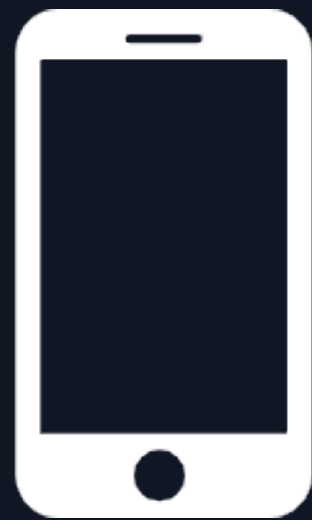
“这是1号”

“这是2号”

“这是3号”



心跳



“我在，你还在吗？”



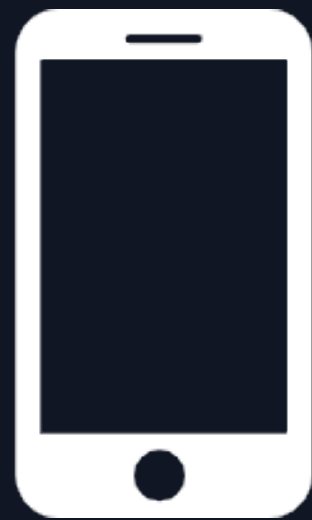
“我还在，你呢？”



.....



乱序重传



“这是1号”



“这是2号”



“这是4号”



“乱了！我最新的是2号”



“好的，重新同步，这是3、4号数据”



TODO：重传 -> 选择重传

断线重连



“这是1号”

“这是2号”

⊗ 通信中断

“复活了吗？”

.....

“你好，我是新的服务进程”

“从2号开始同步”

“这是3号”



超时确认



“这是1号”

“这是2号”

 沉默.....

“还在吗？我最新是2号”

“这是3号”



你为什么不用 Socket.io ？



😊 有用的部分

双向通信

自动重连

心跳

😐 不太有用的部分

降级到HTTP

广播

😞 没用/不好的部分

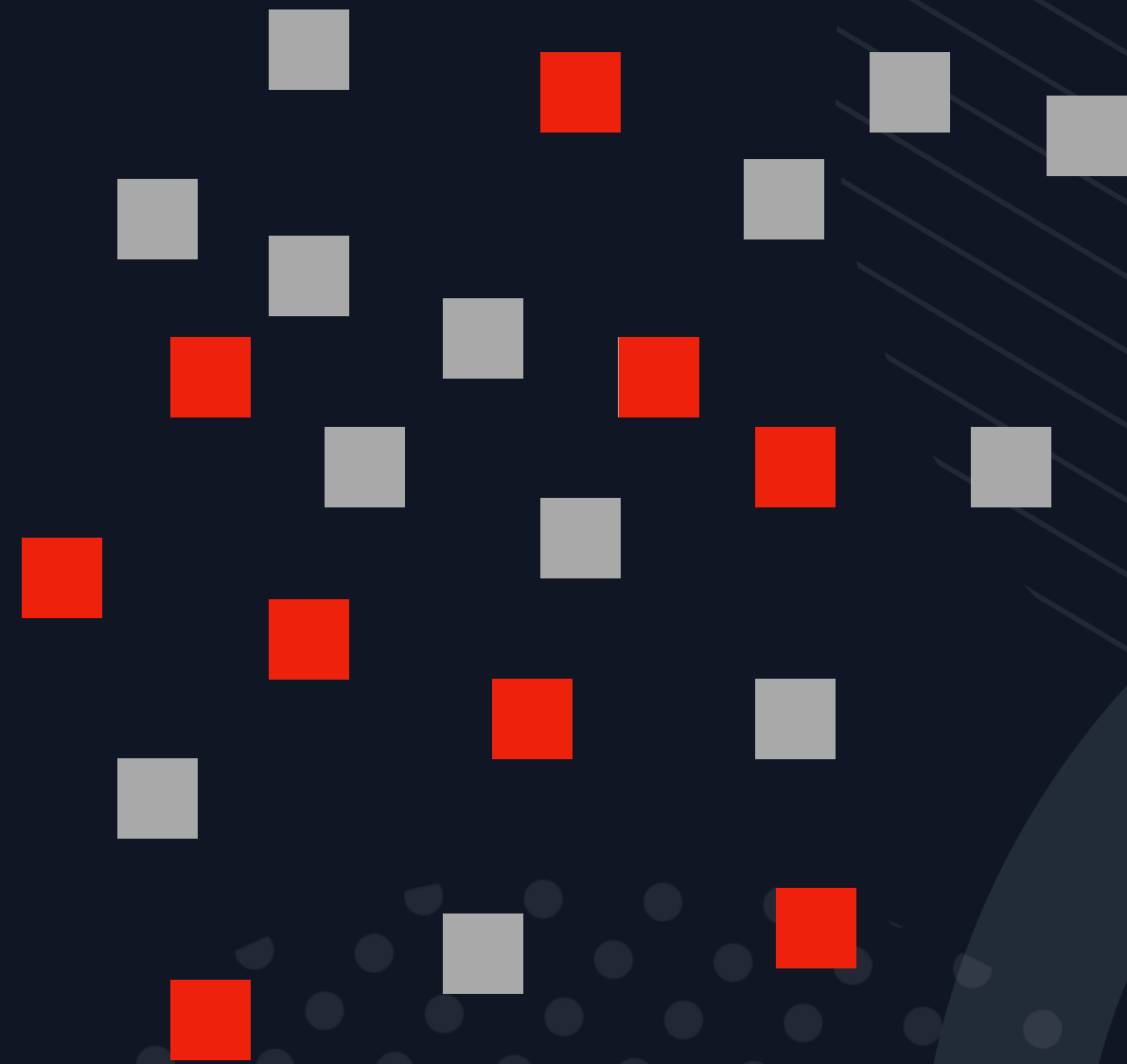
握手协议

Room/NameSpace

横向扩容不友好

如何同步数据库

Watcher





文档的变化

(DataType)

增

删

改 (Update)

改 (Replace)



队列的变化

(QueueType)

入队

出队

内部变化

ChangeEvent 说明

字段	类型	说明
id	number	更新事件 id
queueType	string	列表更新类型，表示更新事件对监听列表的影响，枚举值，定义见 QueueType
dataType	string	数据更新类型，表示记录的具体更新类型，枚举值，定义见 DataType
docId	string	更新的记录 id
doc	object	更新的完整记录
updatedFields	object	所有更新的字段及字段更新后的值，key 为更新的字段路径，value 为字段更新后的值，仅在 <code>update</code> 操作时有此信息
removedFields	string[]	所有被删除的字段，仅在 <code>update</code> 操作时有此信息



托管

静态托管不是很简单吗？



```
$ echo "hello world" > index.html  
$ npx serve
```

实现路由规则

/
/index.html



index.html

/xxx
/xxx.html



xxx.html

/foo
/foo/



foo.html
foo/index.html

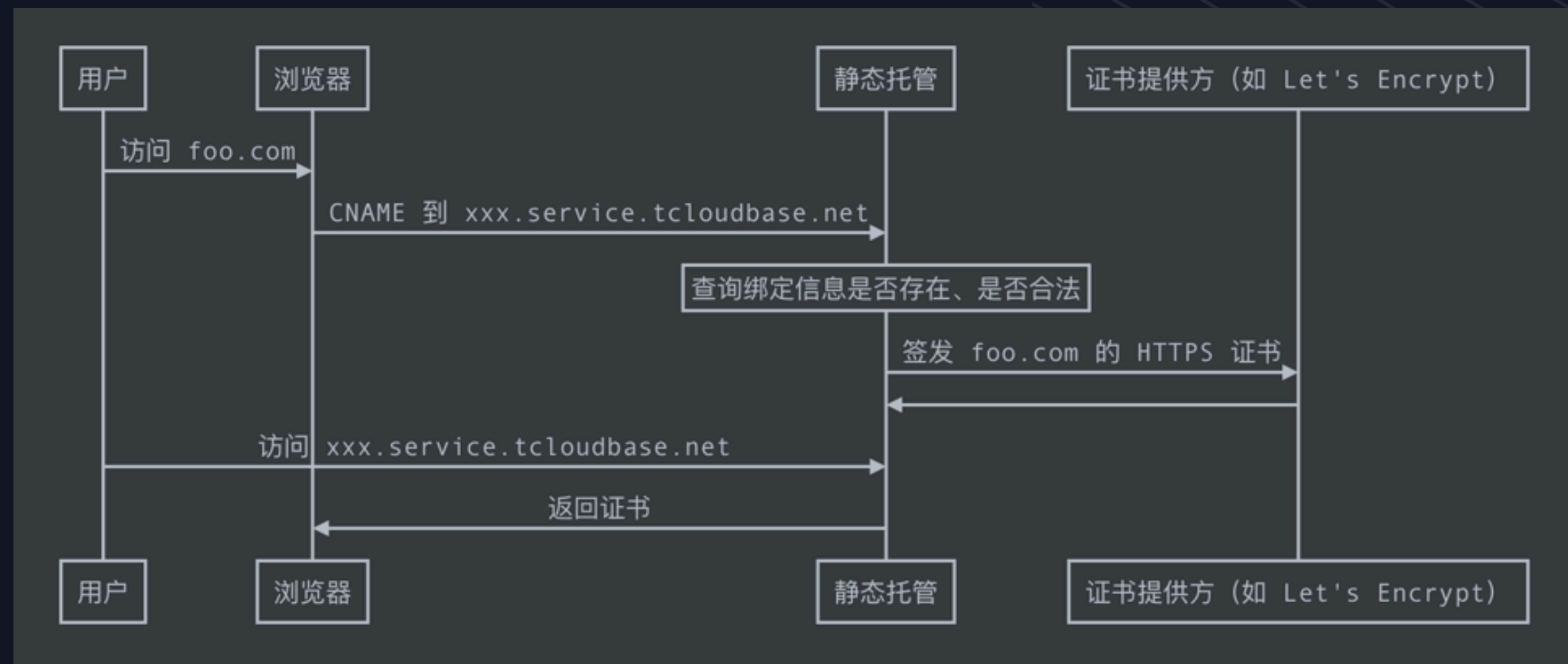
HTTPS与自定义域名

买一个域名

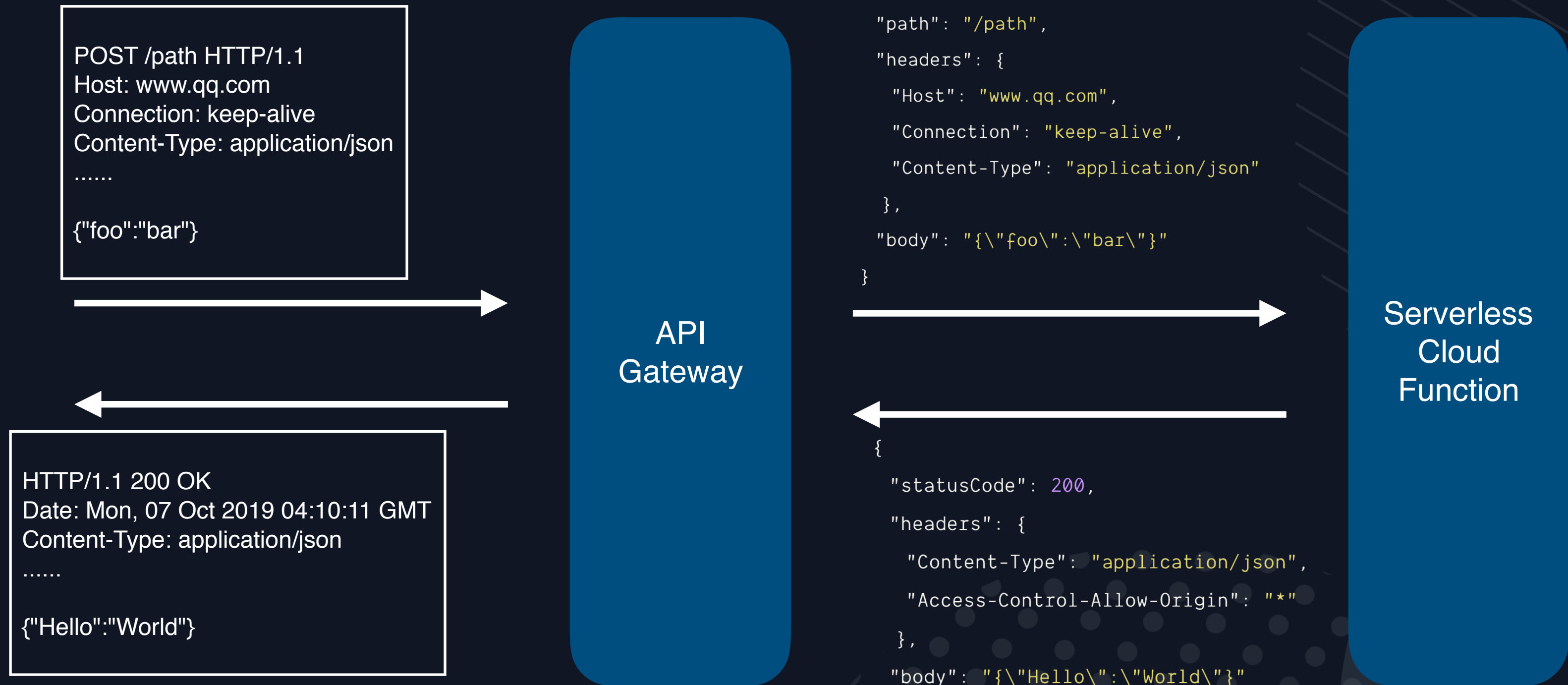
把域名 CNAME 到 xxx.github.io

项目根目录下放一个 CNAME 文件

HTTPS与自定义域名



Serverless动态托管



将 Node Server 转换为函数



```
const serverless = require('serverless-http')
const express = require('express')

// 初始化 server
const app = express()
app.get('/', (req, res) => res.send('Hello World!'))

// 将 server 转为云函数
module.exports.handler = serverless(app)
```

Node.js实现高性能网关

完整的HTTP请求

```
POST /path HTTP/1.1
Host: www.qq.com
Connection: keep-alive
Content-Type: application/json

{
  "foo": "bar",
  .....
}
```

Node.js Core 收到的 Chunk

Chunk 1 POST /pa

Chunk 2 th HTTP/1.1
Host: www.q

Chunk 3 q.com
Connection: keep-alive
Content-Type: appli

Chunk 4 cation/json
{
 "foo": "bar

Chunk 5 bar"},
.....



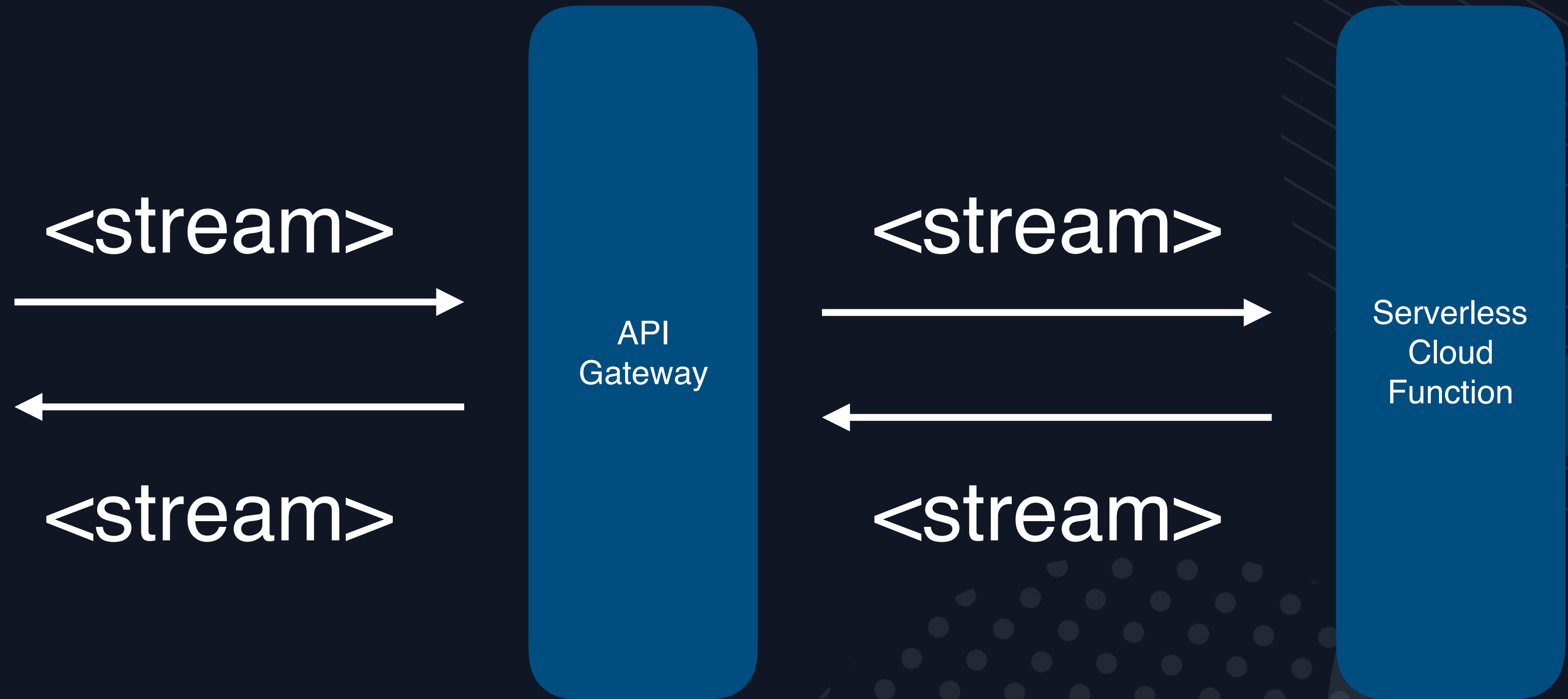
```
const http = require('http');

const server = http.createServer((req, res) => {
  let body = '';
  req.setEncoding('utf8');

  req.on('data', (chunk) => {
    body += chunk;
  });

  req.on('end', () => {
    console.log(body)
  });
});
```

Node.js实现高性能网关



如何托管 SSR 应用?

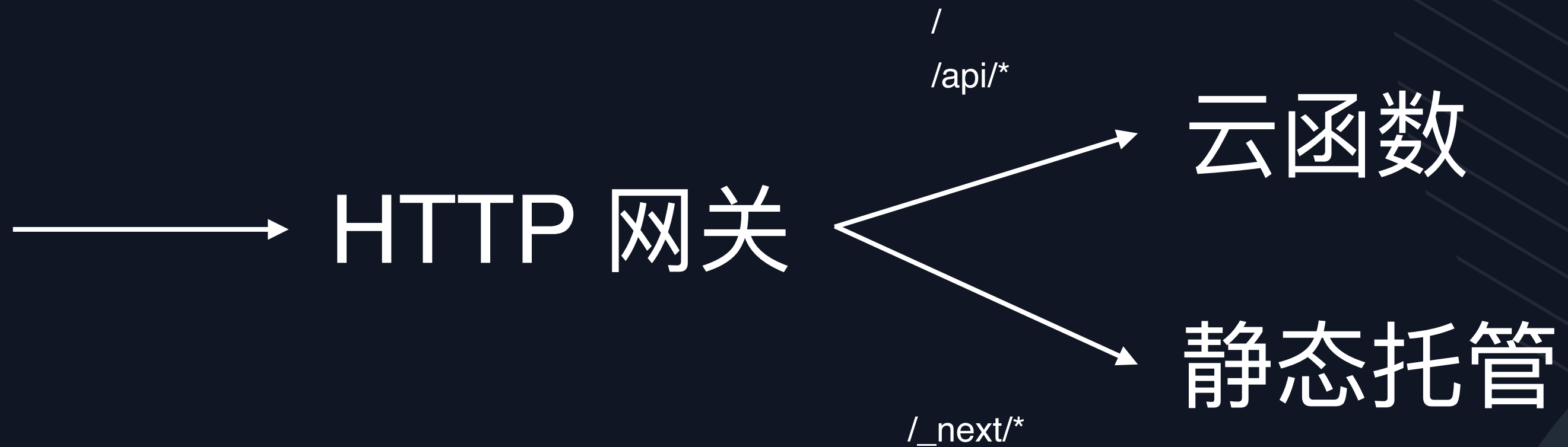
NEXT.js

 NUXTJS

部署 Next.js



访问 Next.js





```
$ npm install -g @cloudbase/cli
```

```
$ echo "<p>Hello CloudBase!</p>" > index.html
```

```
$ cloudbase hosting:deploy index.html
```

总结



Q / A

Github@starkwang

知乎@Starkwang