登录授权、TCP/IP、HTTPS

HenCoder Plus
扔物线

登录和授权

- Cookie
- Authorization

Cookie

Cookie

- Cookie 的起源：购物车

Cookie

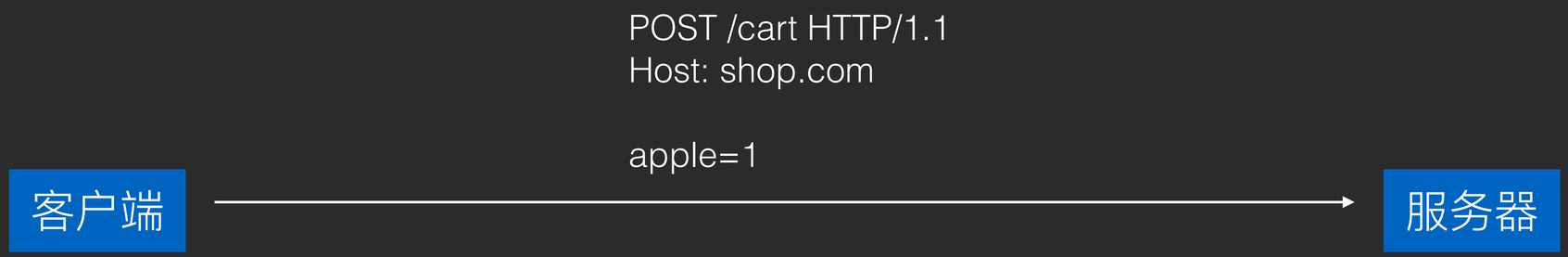
- Cookie 的起源：购物车
- Cookie 的工作机制

Cookie 的工作机制

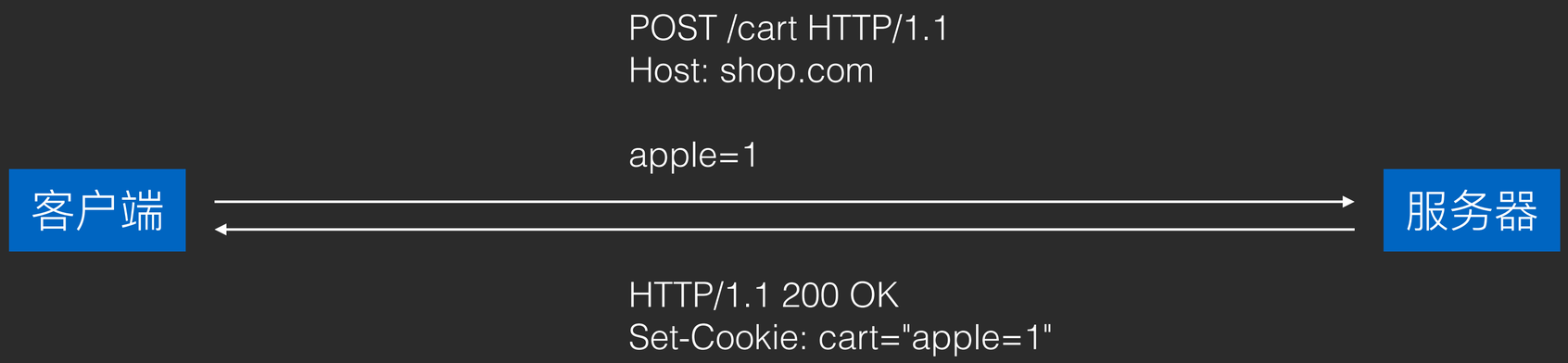
客户端

服务器

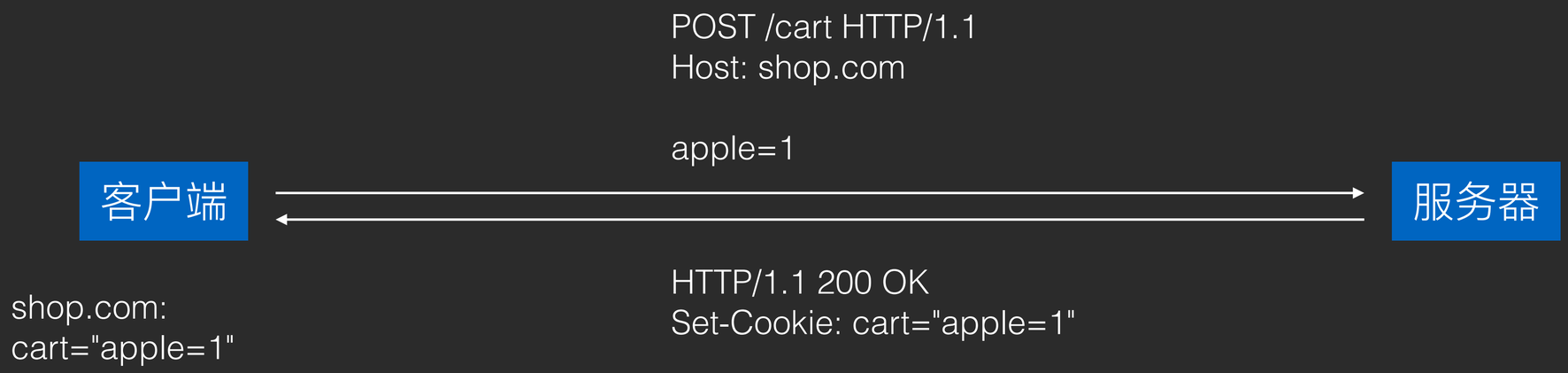
Cookie 的工作机制



Cookie 的工作机制



Cookie 的工作机制



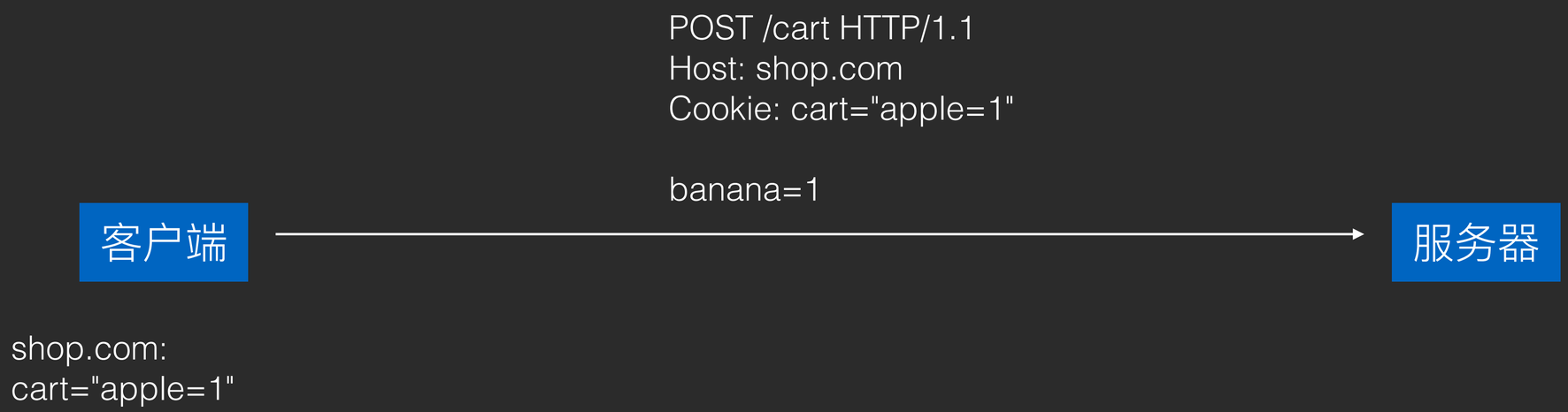
Cookie 的工作机制

客户端

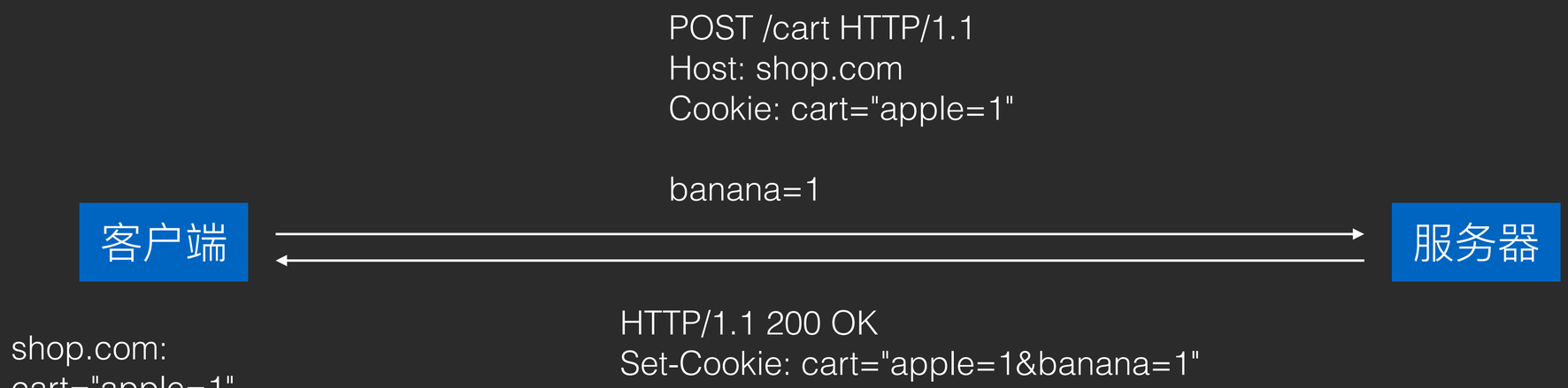
服务器

shop.com:
cart="apple=1"

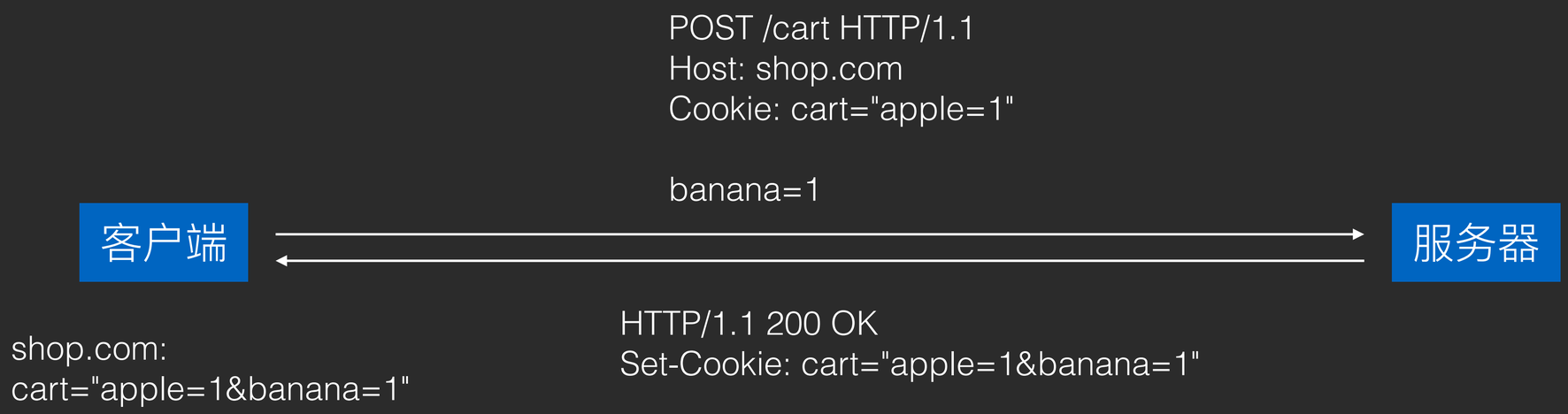
Cookie 的工作机制



Cookie 的工作机制



Cookie 的工作机制



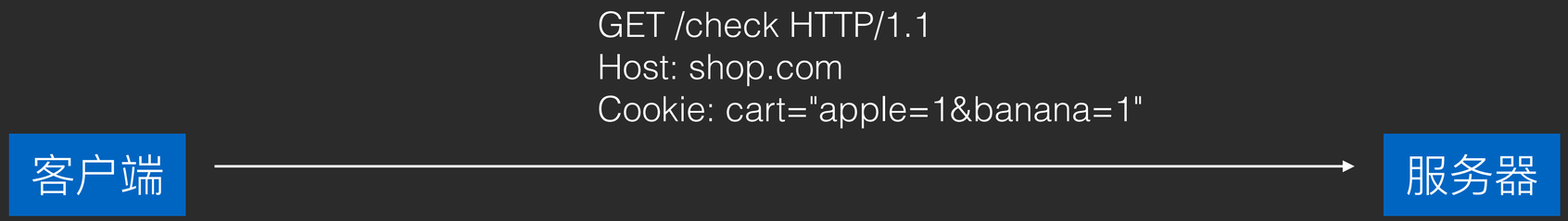
Cookie 的工作机制

客户端

服务器

shop.com:
cart="apple=1&banana=1"

Cookie 的工作机制



shop.com:
cart="apple=1&banana=1"

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等

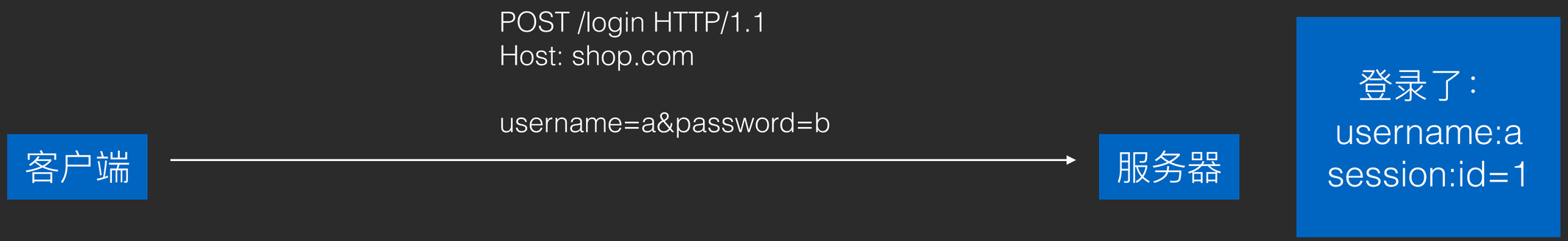
使用 Cookie 管理登录状态

使用 Cookie 管理登录状态

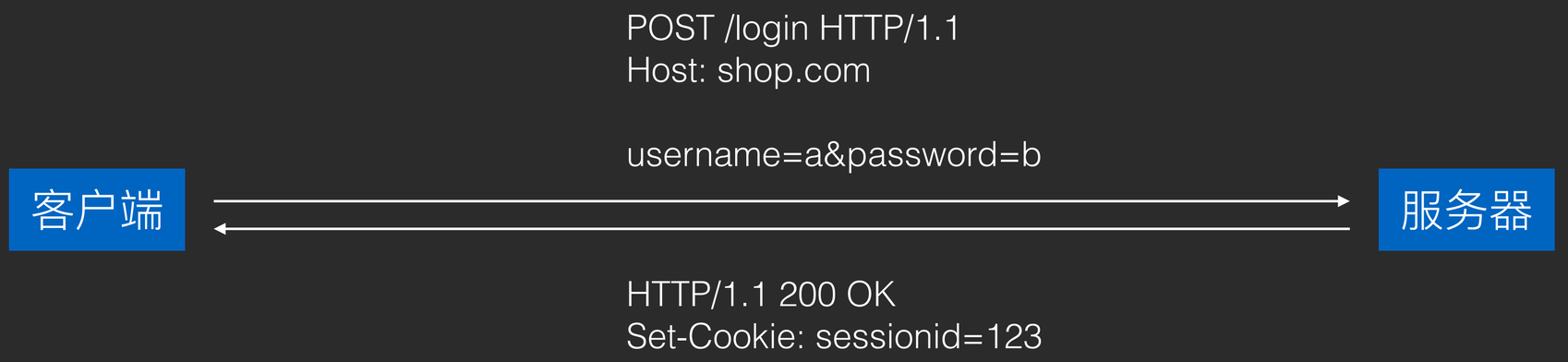
客户端

服务器

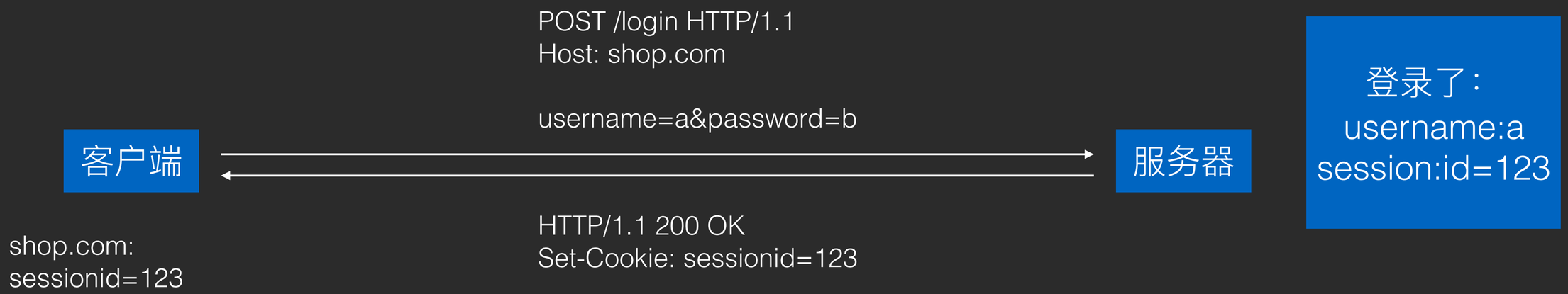
使用 Cookie 管理登录状态



使用 Cookie 管理登录状态



使用 Cookie 管理登录状态



使用 Cookie 管理登录状态

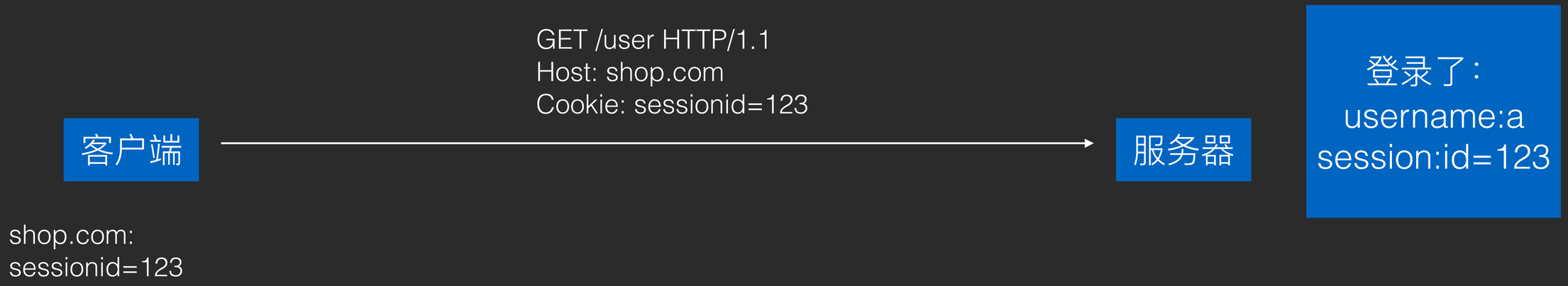
客户端

shop.com:
sessionid=123

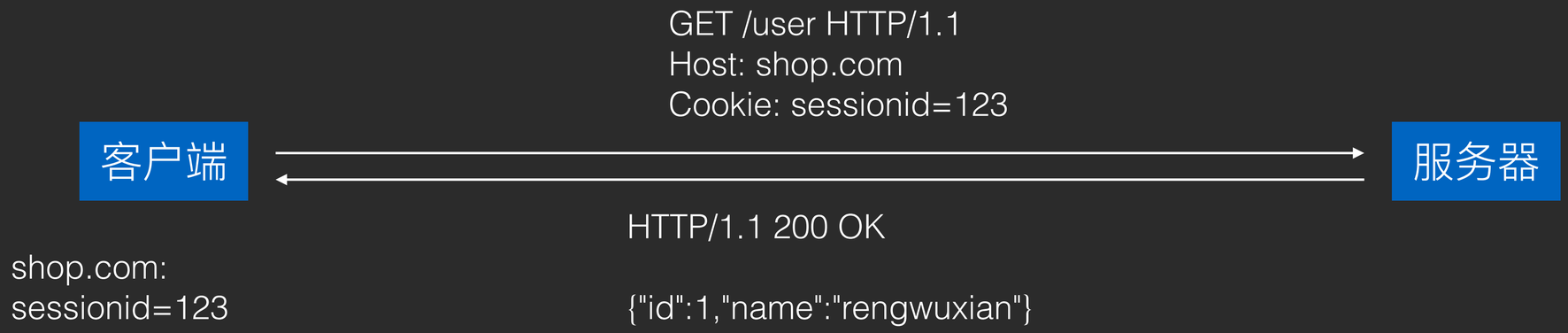
服务器

登录了:
username:a
session:id=123

使用 Cookie 管理登录状态



使用 Cookie 管理登录状态



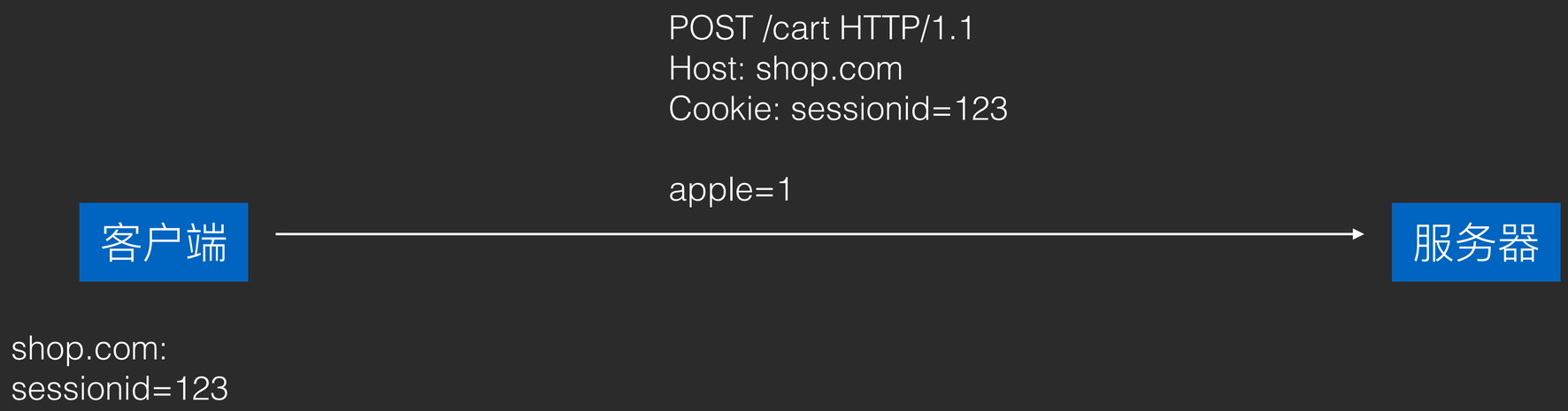
使用 Cookie 管理登录状态

客户端

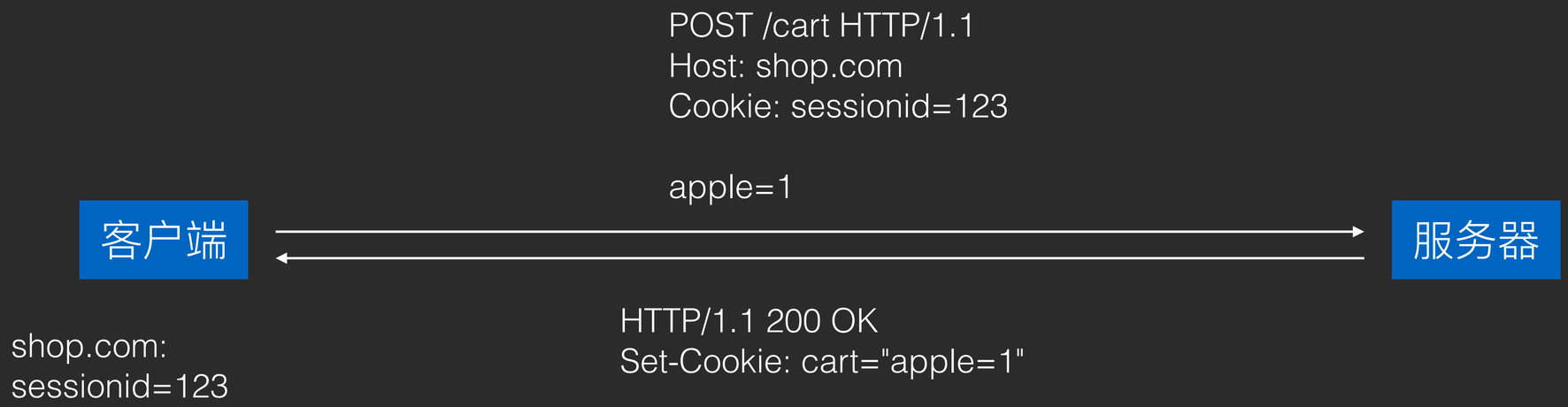
服务器

shop.com:
sessionid=123

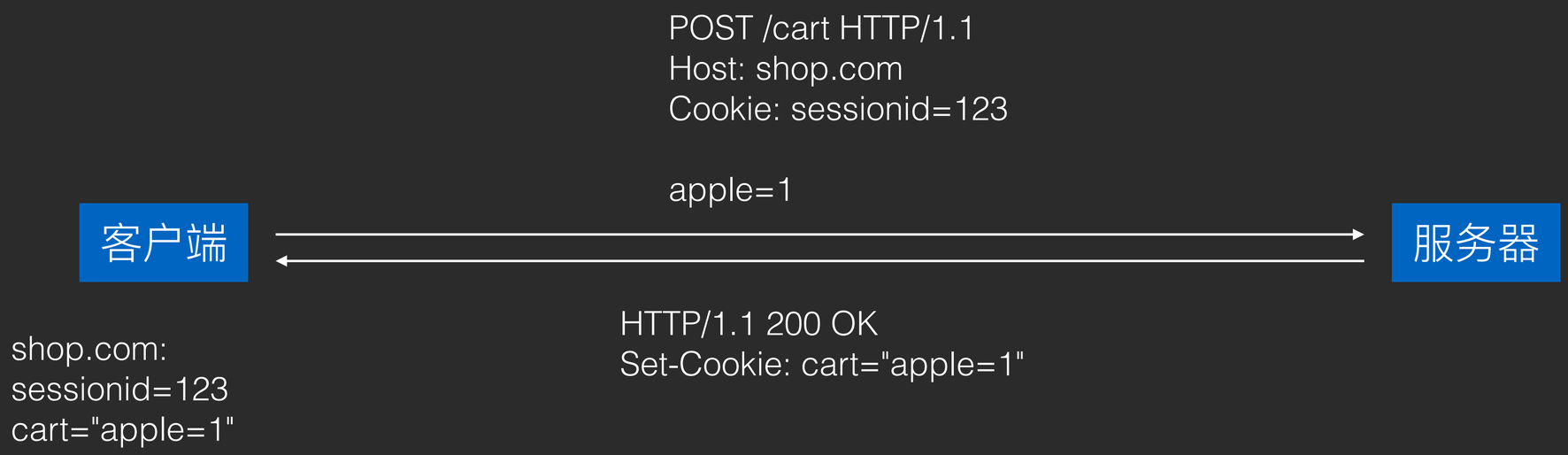
使用 Cookie 管理登录状态



使用 Cookie 管理登录状态



使用 Cookie 管理登录状态



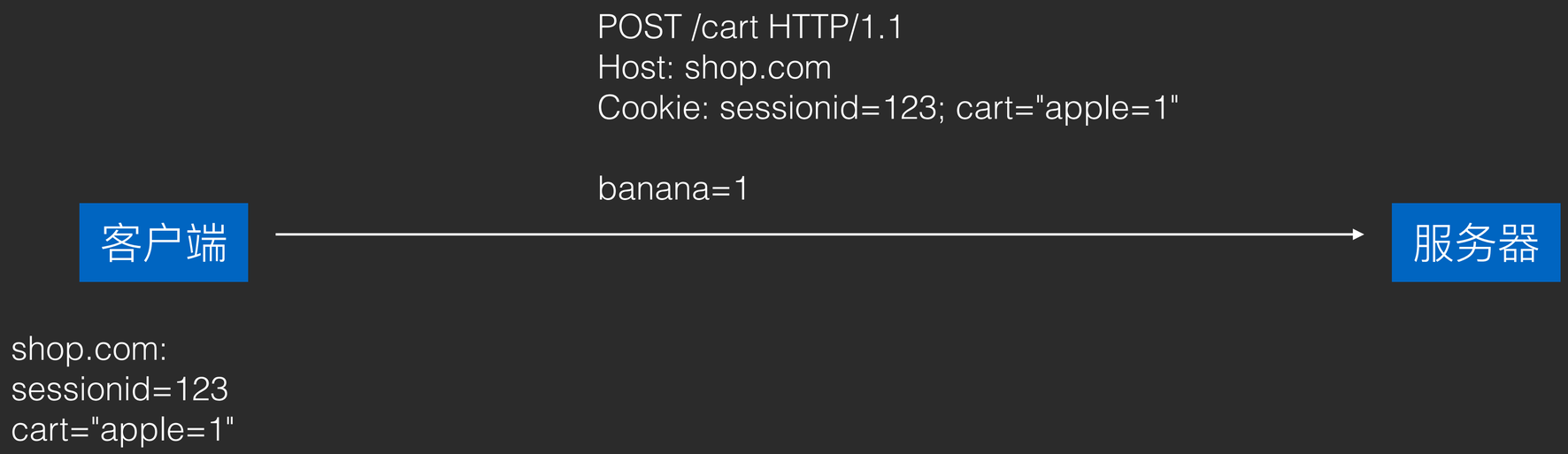
使用 Cookie 管理登录状态

客户端

服务器

shop.com:
sessionid=123
cart="apple=1"

使用 Cookie 管理登录状态



Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等
  - 个性化：用户偏好、主题

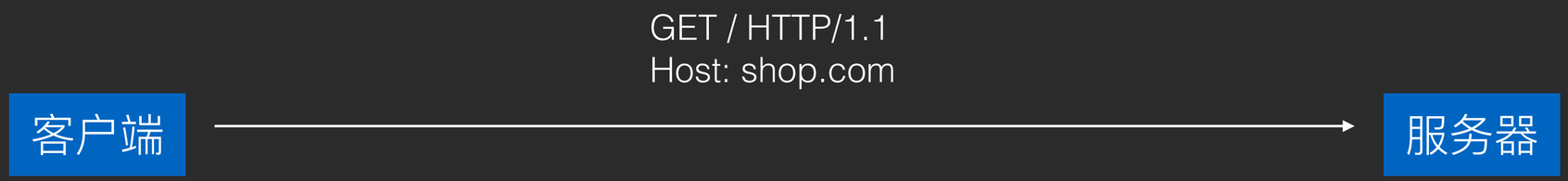
用 Cookie 管理用户偏好

用 Cookie 管理用户偏好

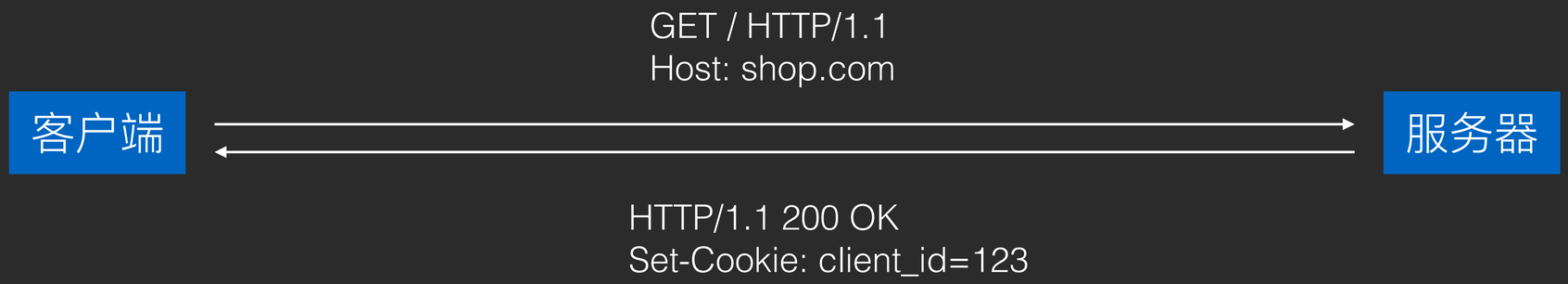
客户端

服务器

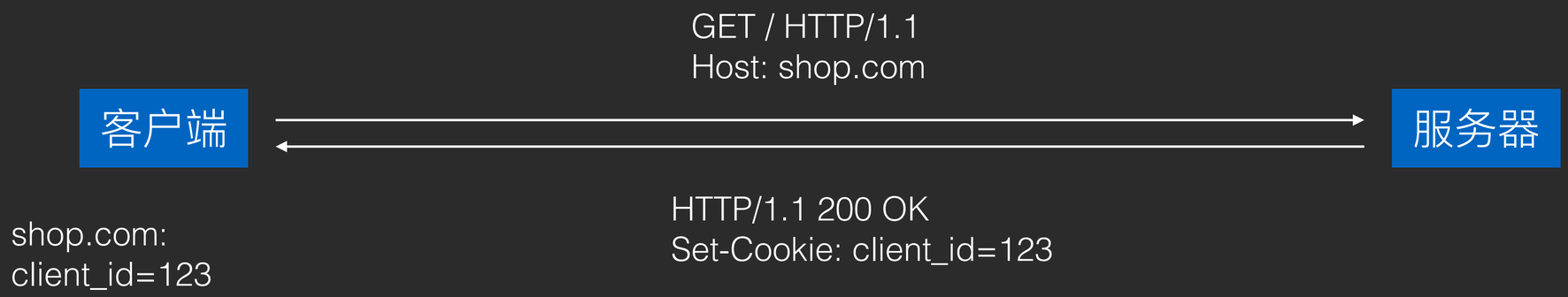
用 Cookie 管理用户偏好



用 Cookie 管理用户偏好



用 Cookie 管理用户偏好



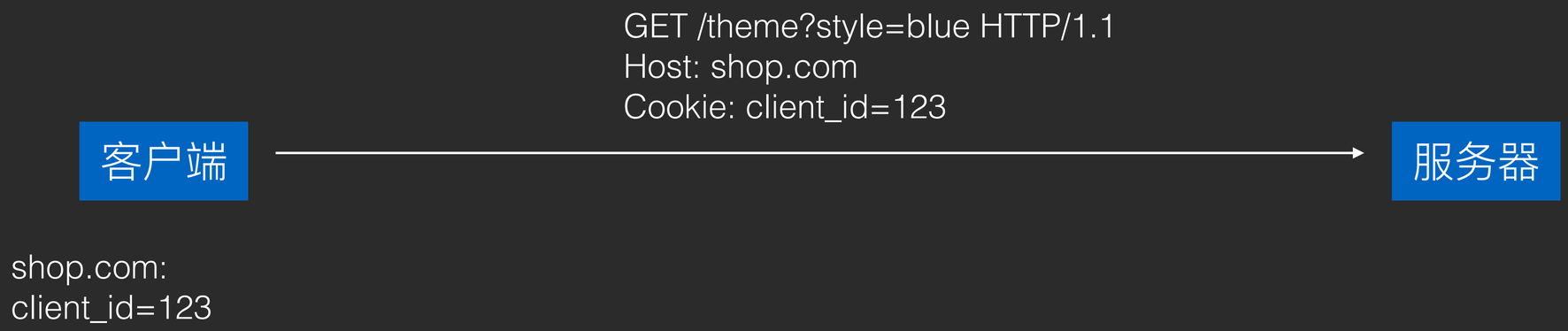
用 Cookie 管理用户偏好

客户端

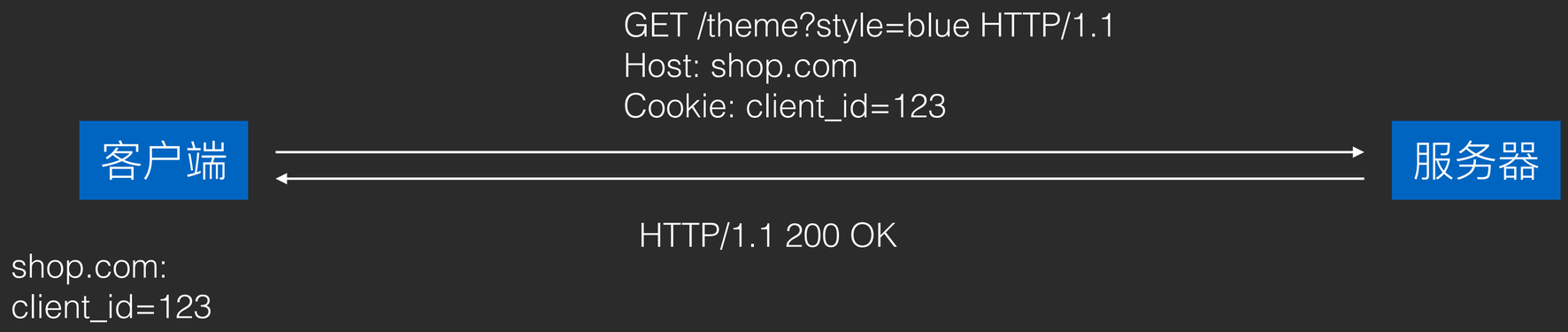
服务器

shop.com:
client_id=123

用 Cookie 管理用户偏好



用 Cookie 管理用户偏好



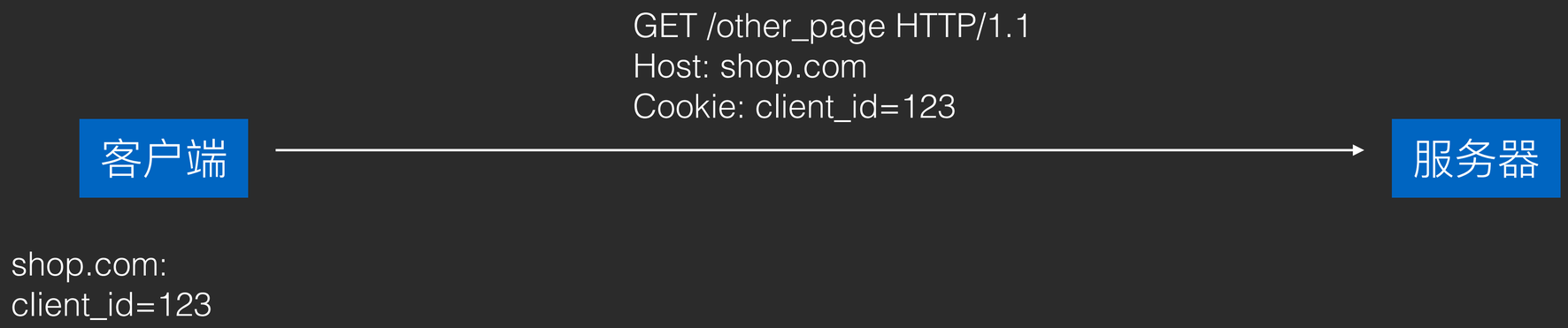
用 Cookie 管理用户偏好

客户端

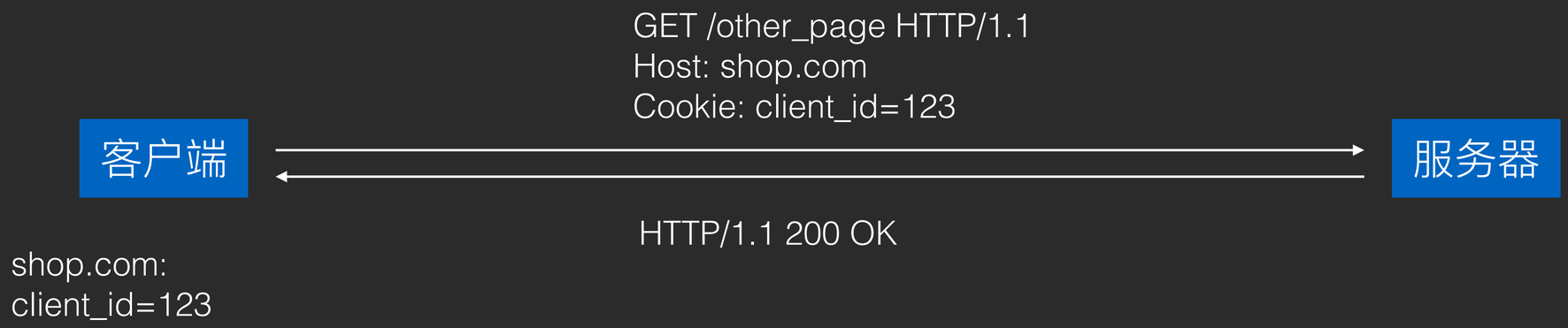
服务器

shop.com:
client_id=123

用 Cookie 管理用户偏好



用 Cookie 管理用户偏好



Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等
  - 个性化：用户偏好、主题
  - Tracking：分析用户行为

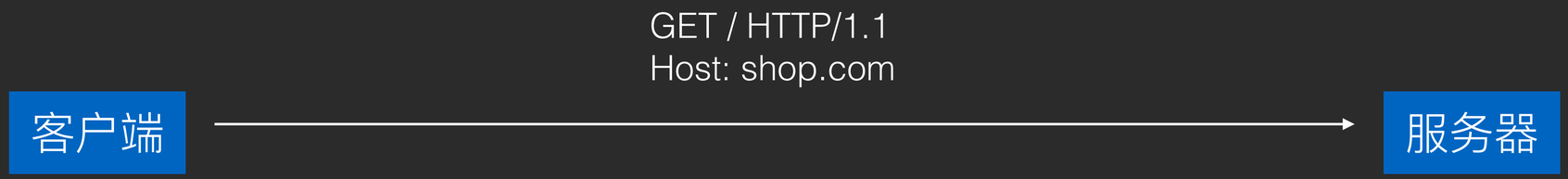
使用 Cookie 追踪用户行为

使用 Cookie 追踪用户行为

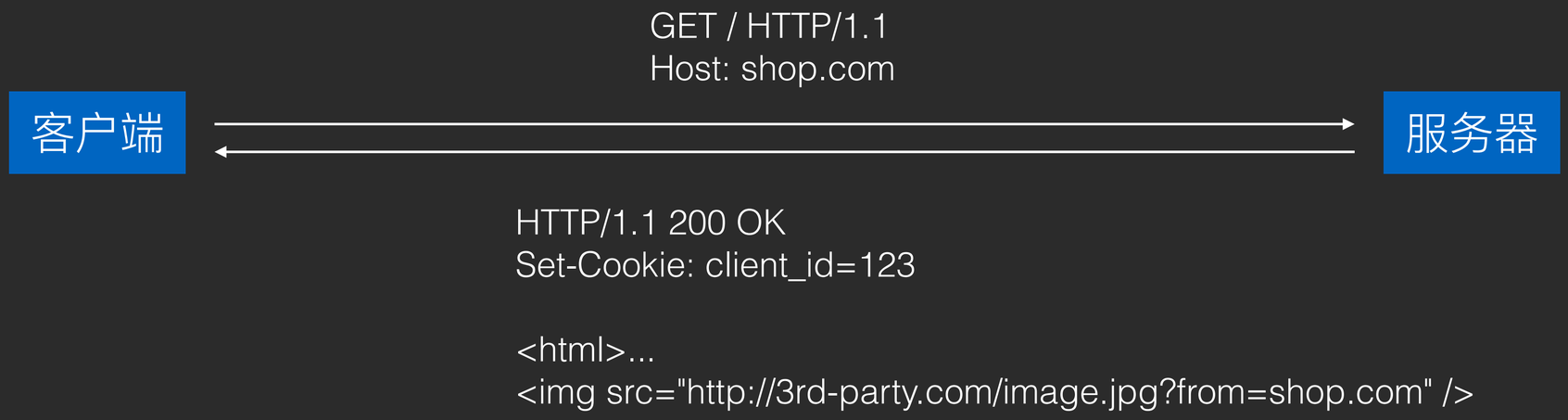
客户端

服务器

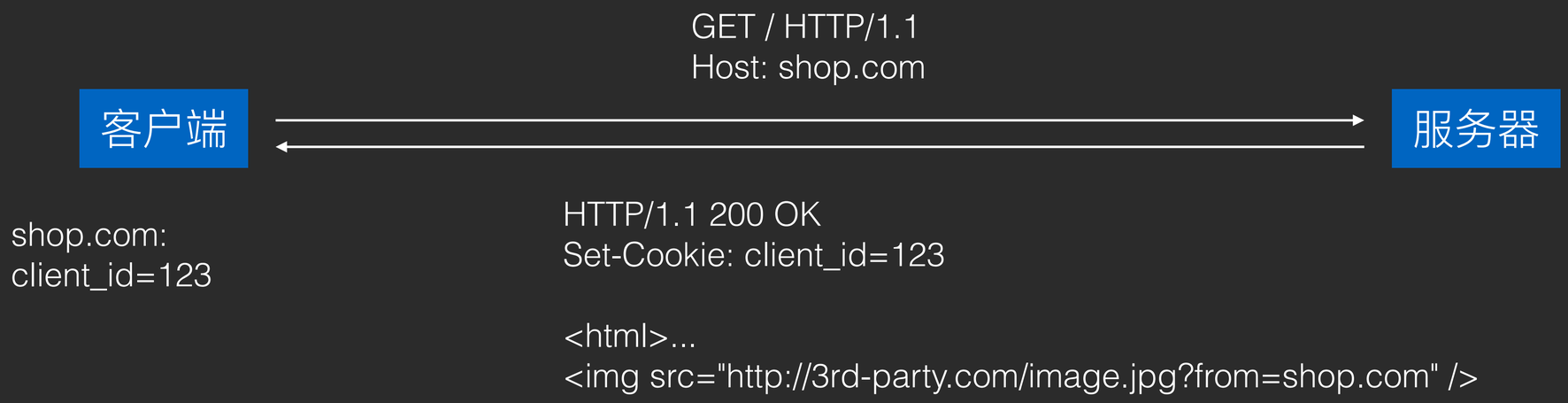
使用 Cookie 追踪用户行为



使用 Cookie 追踪用户行为



使用 Cookie 追踪用户行为



使用 Cookie 追踪用户行为

使用 Cookie 追踪用户行为

使用 Cookie 追踪用户行为

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等
  - 个性化：用户偏好、主题
  - Tracking：分析用户行为

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等
  - 个性化：用户偏好、主题
  - Tracking：分析用户行为
- XSS (Cross-site scripting): HttpOnly

Cookie

- Cookie 的起源：购物车
- Cookie 的工作机制
- Cookie 的作用
  - 会话管理：登录状态、购物车等
  - 个性化：用户偏好、主题
  - Tracking：分析用户行为
- XSS (Cross-site scripting): HttpOnly
- XSRF (Cross-site request forgery): Referer

登录和授权

- Cookie
- Authorization

Authorization

Authorization

- Basic

Basic

- Authorization: Basic <username:password(Base64ed)>

Authorization

- Basic

Authorization

- Basic
- Bearer

Bearer

- Authorization: Bearer <bearer token>

Authorization

- Basic
- Bearer

Authorization

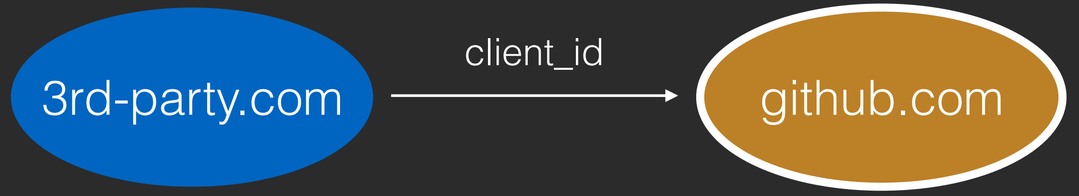
- Basic
- Bearer
  - OAuth2

Authorization

- Basic
- Bearer
  - OAuth2
    - OAuth2 流程

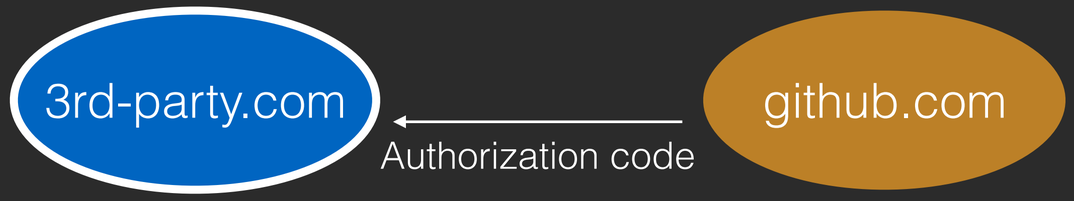
OAuth2 流程

OAuth2 流程



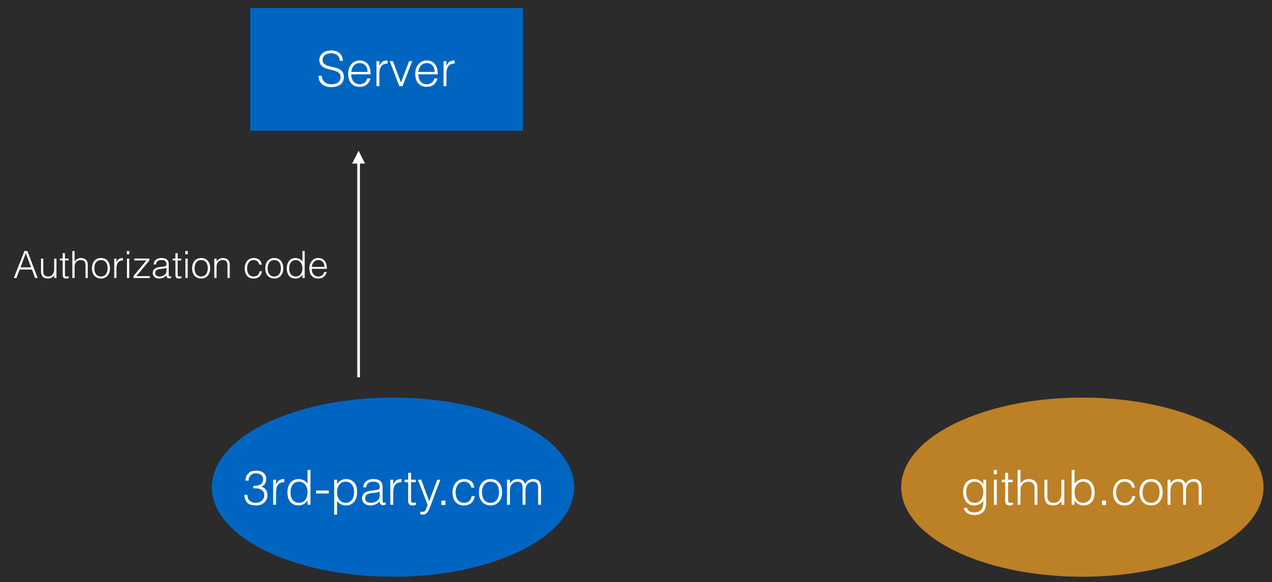
OAuth2 流程

OAuth2 流程

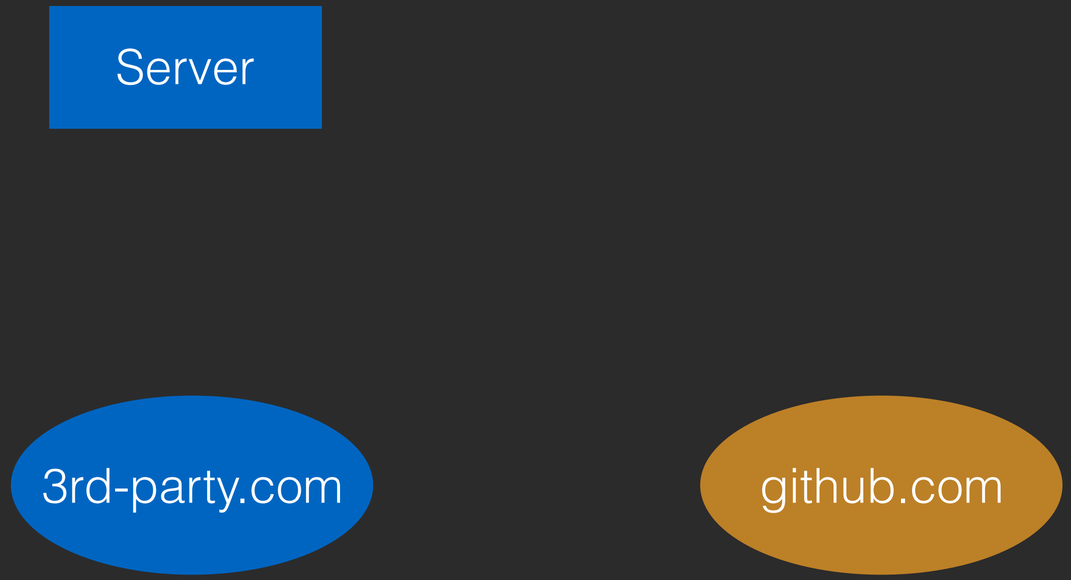


OAuth2 流程

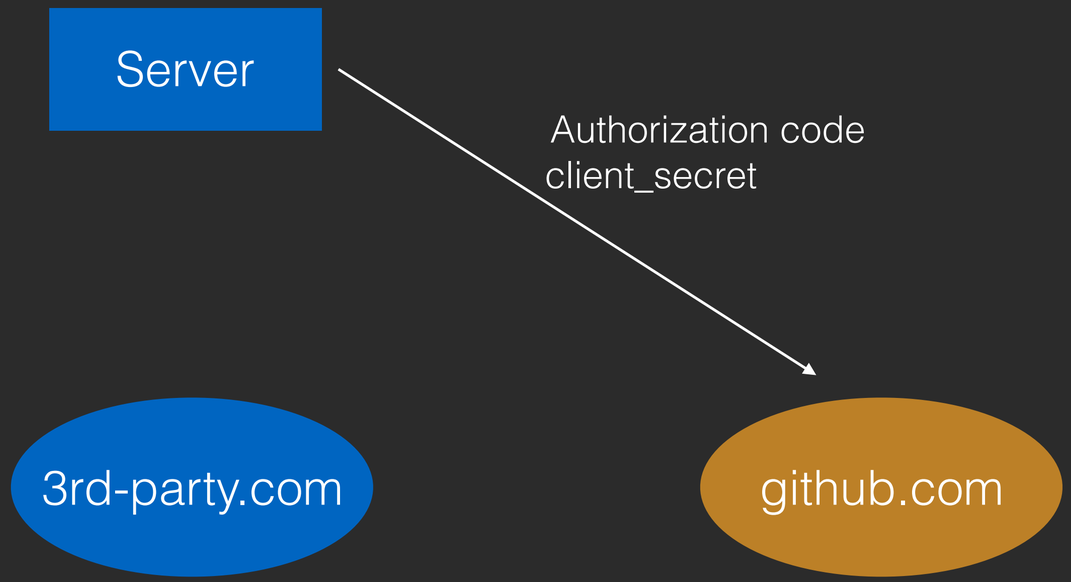
OAuth2 流程



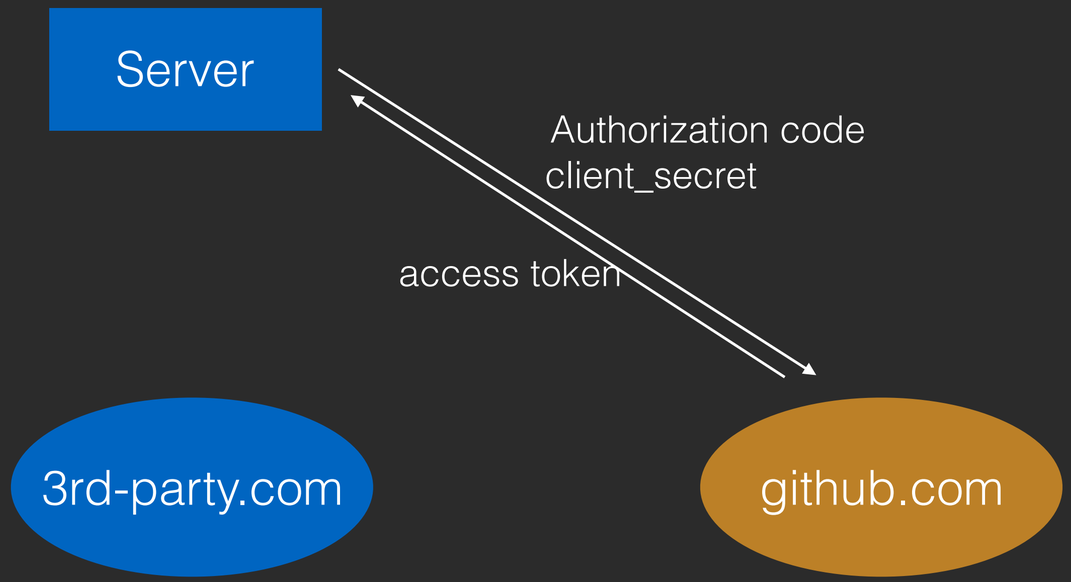
OAuth2 流程



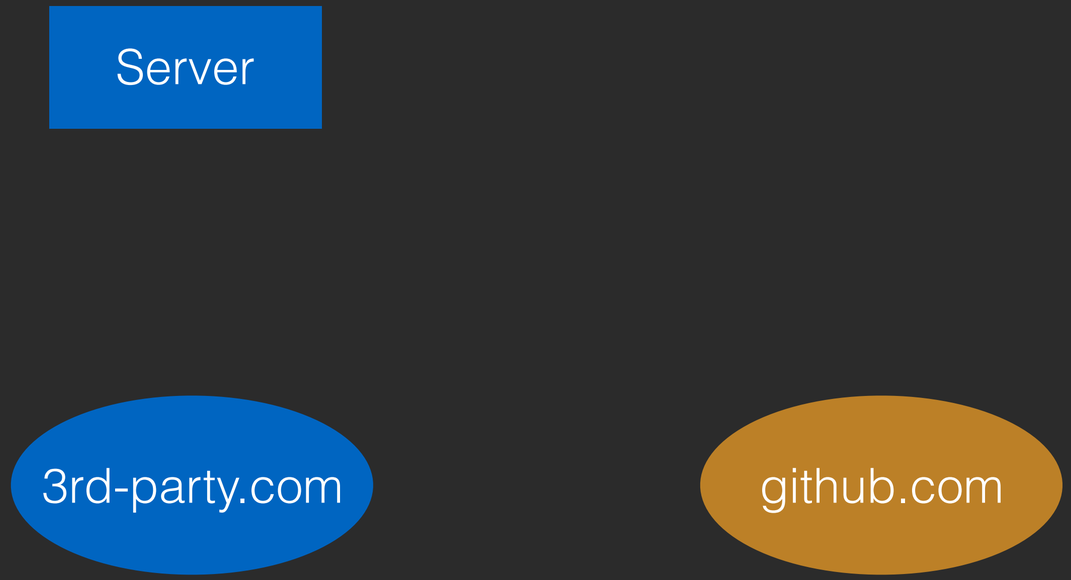
OAuth2 流程


Server
Authorization code
client_secret
3rd-party.com
github.com

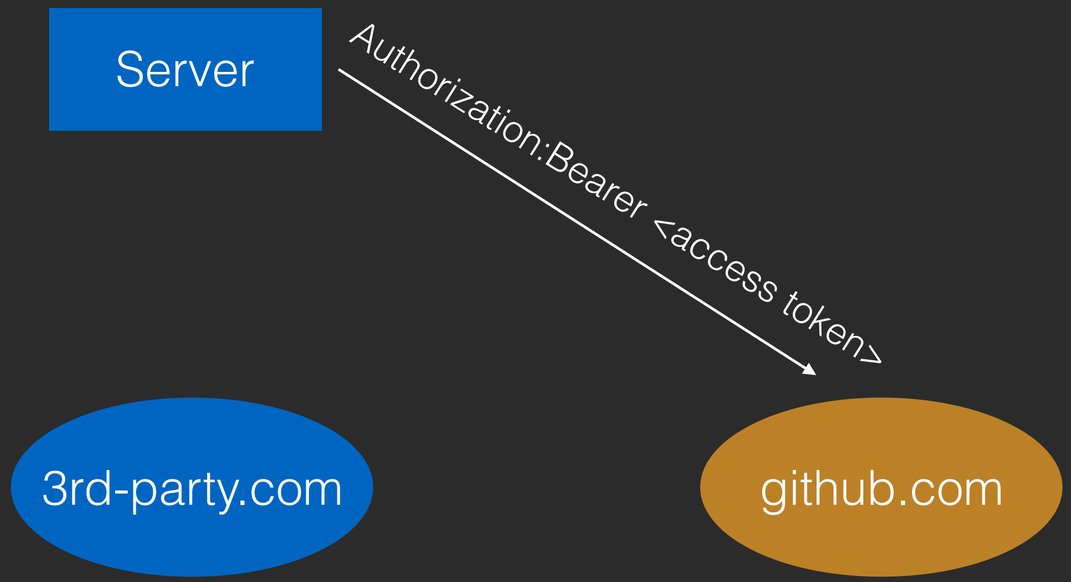
OAuth2 流程


Server
Authorization code
client_secret
access token
3rd-party.com
github.com

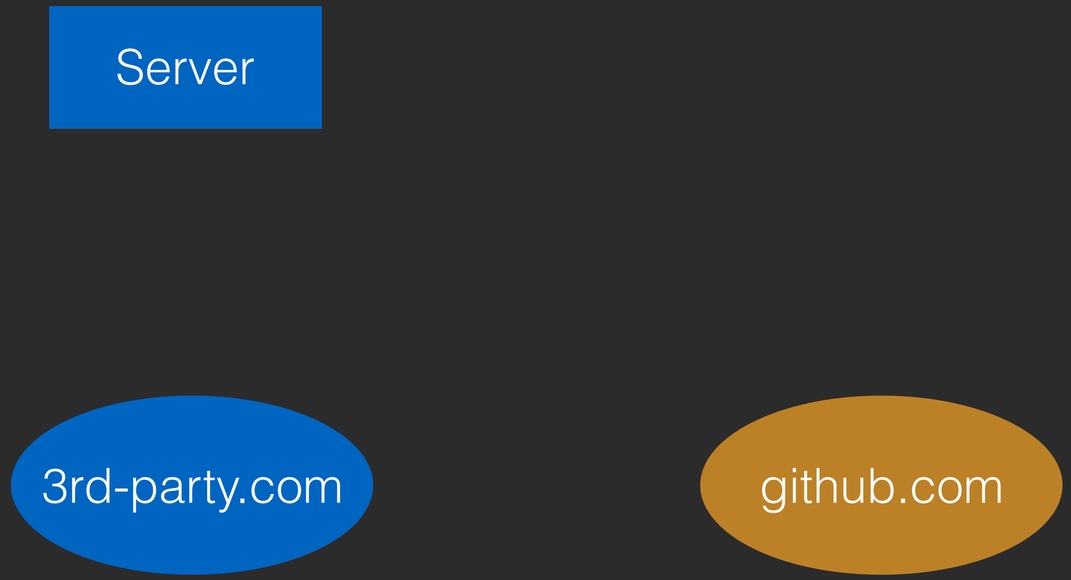
OAuth2 流程



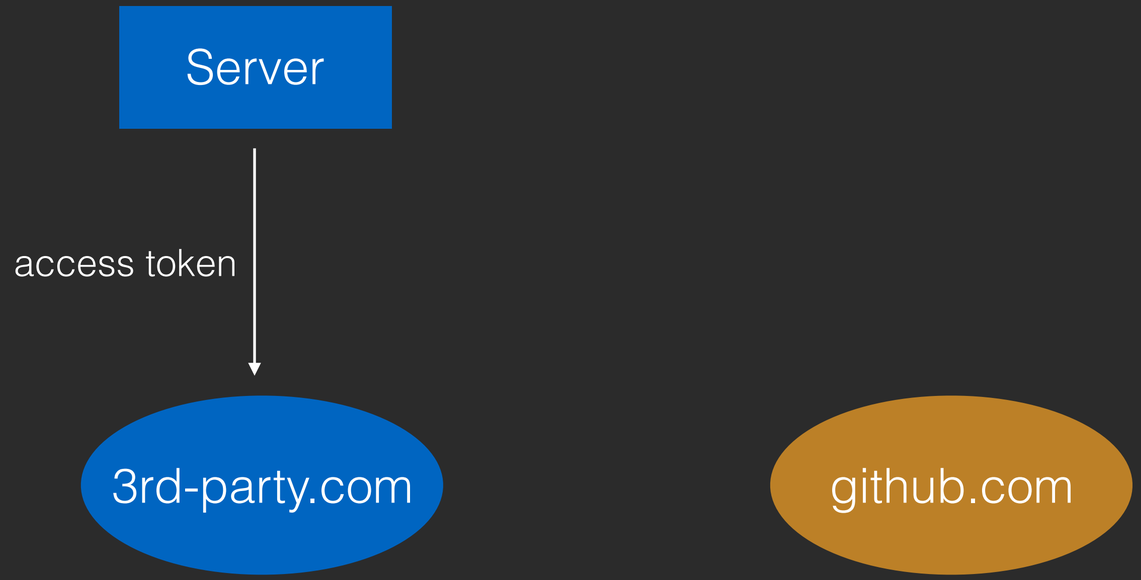
OAuth2 流程


Server
Authorization:Bearer <access token>
3rd-party.com
github.com

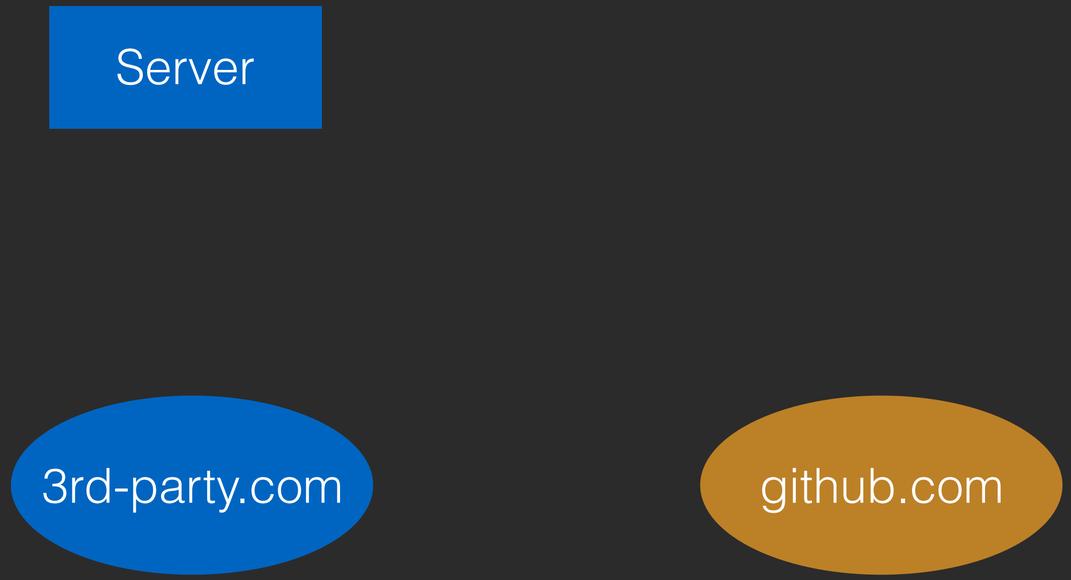
OAuth2 流程



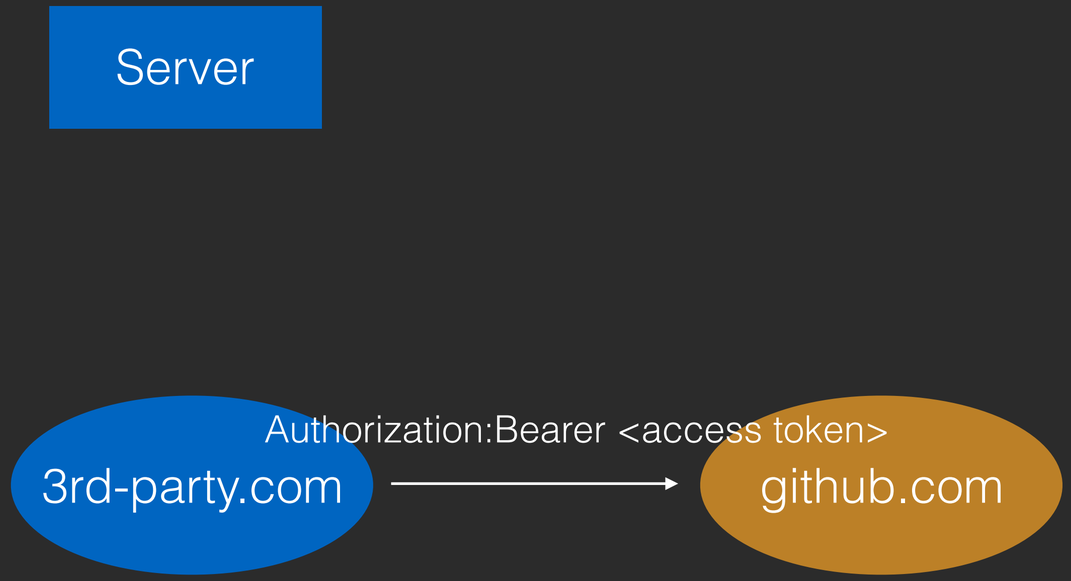
OAuth2 流程



OAuth2 流程



OAuth2 流程


Server
Authorization:Bearer <access token>
3rd-party.com
github.com

Authorization

- Basic
- Bearer
  - OAuth2
    - OAuth2 流程

Authorization

- Basic
- Bearer
  - OAuth2
    - OAuth2 流程
    - 微信登录

Authorization

- Basic
- Bearer
  - OAuth2
    - OAuth2 流程
    - 微信登录
  - 自家 App 中使用 Bearer token

Authorization

- Basic
- Bearer
  - OAuth2
    - OAuth2 流程
    - 微信登录
  - 自家 App 中使用 Bearer token
  - refresh token

refresh token

- {
  "token_type": "Bearer",
  "access_token": "xxxxx",
  "refresh_token": "xxxxx",
  "expires_time": "xxxxx"
  }

refresh token

- {
  "token_type": "Bearer",
  "access_token": "xxxxx",
  "refresh_token": "xxxxx",
  "expires_time": "xxxxx"
  }
- 目的：安全

Authorization

- Basic
- Bearer
  - OAuth2
    - OAuth2 流程
    - 微信登录
  - 自家 App 中使用 Bearer token
  - refresh token

TCP / IP 协议族

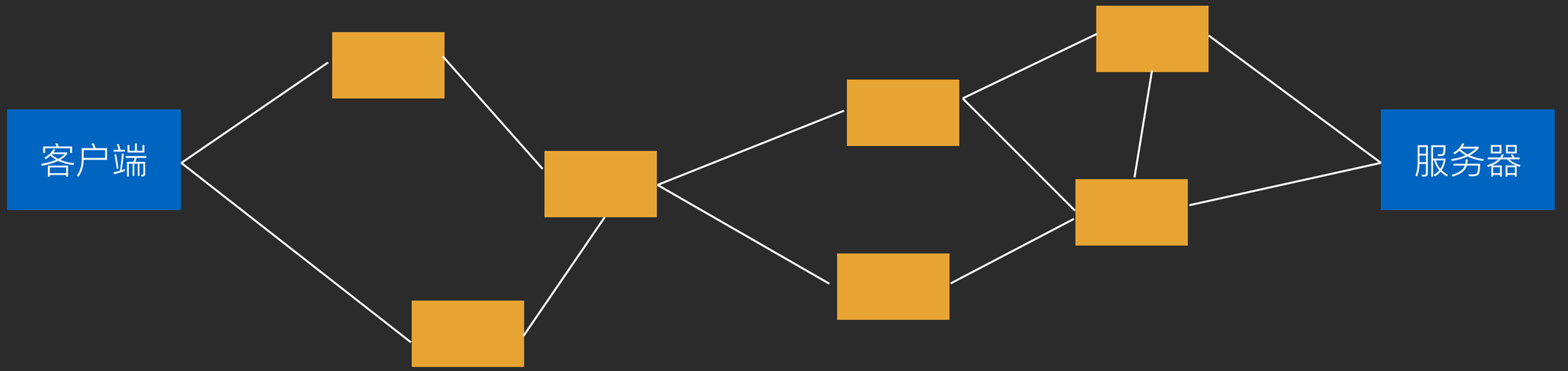
TCP / IP 协议族

- 一系列协议所组成的一个网络分层模型

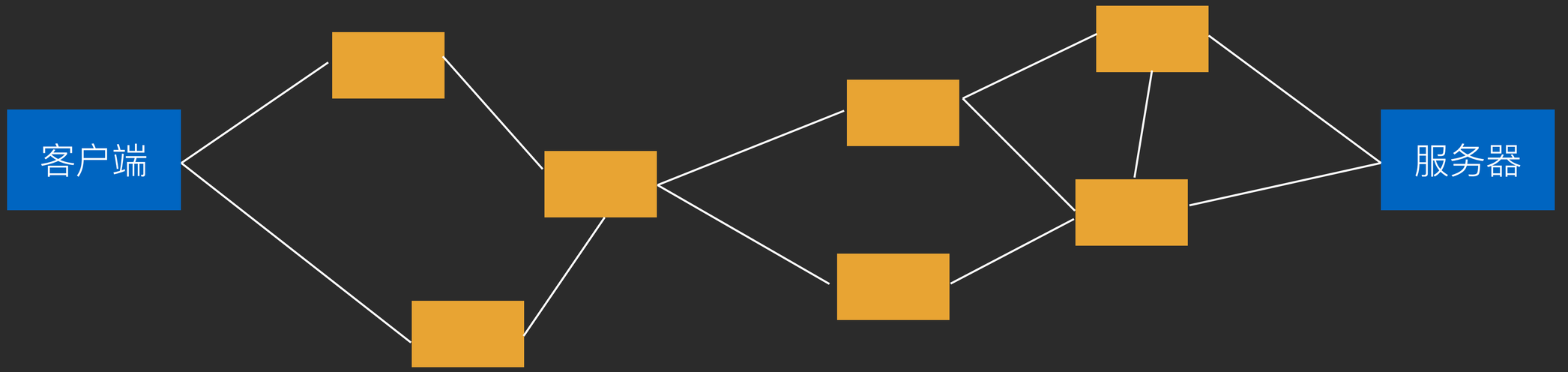
TCP / IP 协议族

- 一系列协议所组成的一个网络分层模型
- 为什么要分层?

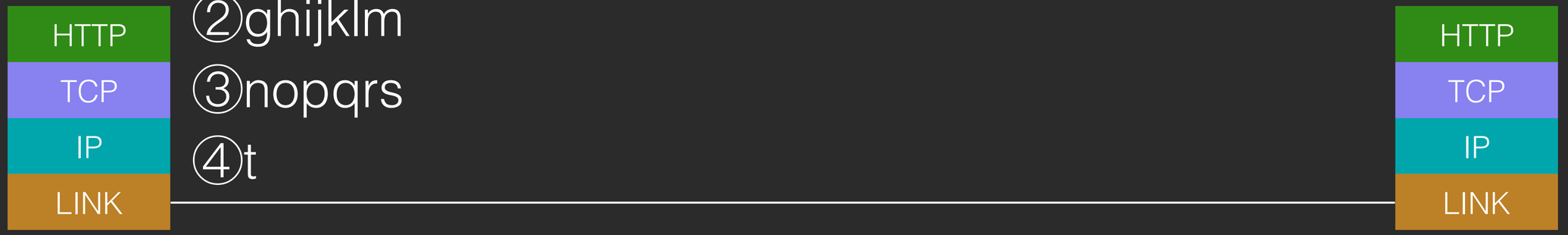
TCP / IP 协议族



TCP / IP 协议族



①abcdef
②ghijklm
③nopqrs
④t



TCP / IP 协议族

- 一系列协议所组成的一个网络分层模型
- 为什么要分层?

TCP / IP 协议族

- 一系列协议所组成的一个网络分层模型
- 为什么要分层？因为现实网络的不可靠性

TCP / IP 协议族

- 一系列协议所组成的一个网络分层模型
- 为什么要分层？因为现实网络的不可靠性
- 具体分层：
  - Application Layer 应用层：HTTP、FTP、DNS
  - Transport Layer 传输层：TCP、UDP
  - Internet Layer 网络层：IP
  - Link Layer 数据链路层：以太网、Wi-Fi

TCP 连接

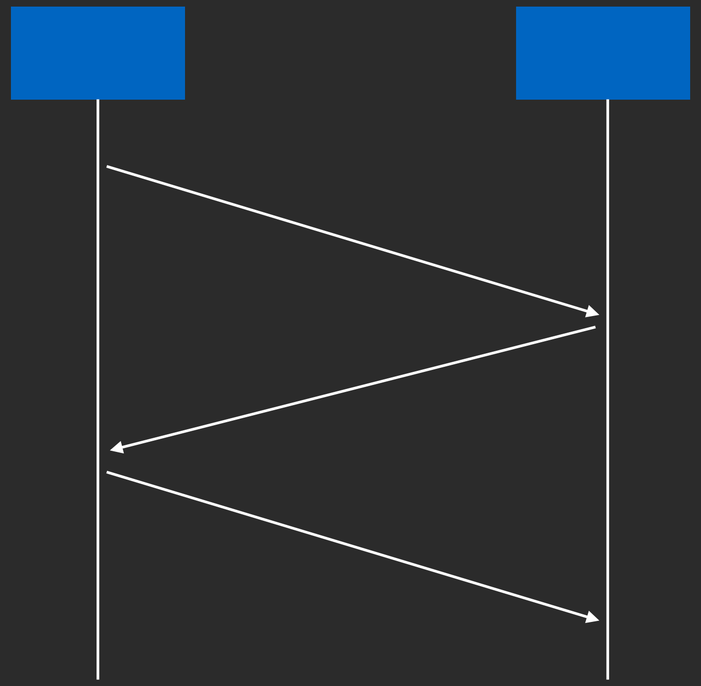
TCP 连接

- 什么叫做连接?

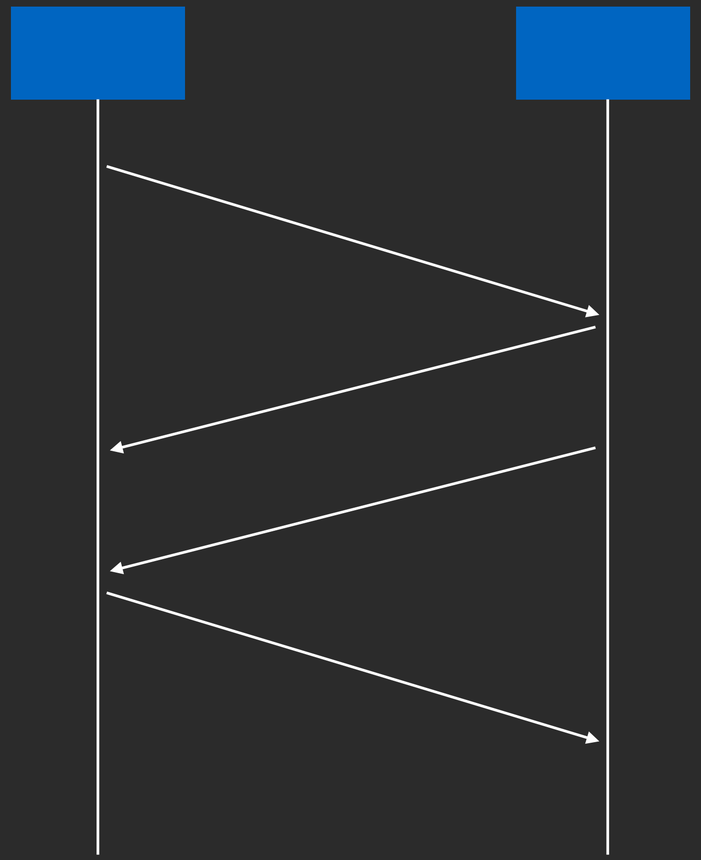
TCP 连接

- 什么叫做连接?
- TCP 连接的建立与关闭

TCP 连接的建立



TCP 连接的关闭



TCP 连接

- 什么叫做连接?
- TCP 连接的建立与关闭

TCP 连接

- 什么叫做连接?
- TCP 连接的建立与关闭
- 长连接

TCP 连接

- 什么叫做连接?
- TCP 连接的建立与关闭
- 长连接
  - 为什么要长连接?

TCP 连接

- 什么叫做连接?
- TCP 连接的建立与关闭
- 长连接
  - 为什么要长连接?
  - 长连接的实现方式：心跳

HTTPS

HTTPS

- HTTP over SSL

HTTPS

- HTTP over SSL
- SSL：Secure Socket Layer -> TLS Transport Layer Secure

HTTPS

- HTTP over SSL
- SSL：Secure Socket Layer -> TLS Transport Layer Secure
- 定义：在 HTTP 之下增加的一个安全层，用于保障 HTTP 的加密传输

HTTPS

- HTTP over SSL
- SSL：Secure Socket Layer -> TLS Transport Layer Secure
- 定义：在 HTTP 之下增加的一个安全层，用于保障 HTTP 的加密传输
- 本质：在客户端和服务器之间协商出一个对称密钥，每次发送信息之前将内容加密，收到之后解密，达到内容的加密传输

HTTPS

- HTTP over SSL
- SSL：Secure Socket Layer -> TLS Transport Layer Secure
- 定义：在 HTTP 之下增加的一个安全层，用于保障 HTTP 的加密传输
- 本质：在客户端和服务器之间协商出一套对称密钥，每次发送信息之前将内容加密，收到之后解密，达到内容的加密传输
- 为什么不直接用非对称加密?

HTTPS 连接

HTTPS 连接

- 客户端请求建立 TLS 连接

HTTPS 连接

- 客户端请求建立 TLS 连接
- 服务器发回证书

HTTPS 连接

- 客户端请求建立 TLS 连接
- 服务器发回证书
- 客户端验证服务器证书

HTTPS 连接

- 客户端请求建立 TLS 连接
- 服务器发回证书
- 客户端验证服务器证书
- 客户端信任服务器后，和服务器协商对称密钥

HTTPS 连接

- 客户端请求建立 TLS 连接
- 服务器发回证书
- 客户端验证服务器证书
- 客户端信任服务器后，和服务器协商对称密钥
- 使用对称密钥开始通信

HTTPS 连接

1. Client Hello
2. Server Hello
3. 服务器证书
4. Pre-master secret
5. 客户端说：我要使用加密通信了
6. 客户端发送：Finished
7. 服务器说：我要使用加密通信了
8. 服务器说：Finished

应用层消息：#@R#RTERT#$TWERT

服务器证书公钥

客户端随机数
服务端随机数

TLS 版本
对称加密算法
非对称加密算法
hash 算法

客户端

Pre-master Secret
Master Secret

客户端加密密钥
服务端加密密钥
客户端 MAC Secret
服务端 MAC Secret

客户端随机数
服务端随机数

TLS 版本
对称加密算法
非对称加密算法
hash 算法

服务器

Pre-master Secret
Master Secret

客户端加密密钥
服务端加密密钥
客户端 MAC Secret
服务端 MAC Secret

在 Android 中使用

在 Android 中使用

- 正常情况：直接使用

在 Android 中使用

- 正常情况：直接使用
- 什么时候会不行?

在 Android 中使用

- 正常情况：直接使用
- 什么时候会不行?
  - 用的是自签名证书（例如只用于内网的 https）

在 Android 中使用

- 正常情况：直接使用
- 什么时候会不行?
  - 用的是自签名证书（例如只用于内网的 https）
  - 证书信息不全，缺乏证书机构信息

在 Android 中使用

- 正常情况：直接使用
- 什么时候会不行?
  - 用的是自签名证书（例如只用于内网的 https）
  - 证书信息不全，缺乏证书机构信息
  - 手机操作系统较旧，没有安装最新加入的根证书

在 Android 中使用

- 正常情况：直接使用
- 什么时候会不行?
  - 用的是自签名证书（例如只用于内网的 https）
  - 证书信息不全，缺乏证书机构信息
  - 手机操作系统较旧，没有安装最新加入的根证书
- 怎么办?

在 Android 中使用

- 正常情况：直接使用
- 什么时候会不行?
  - 用的是自签名证书（例如只用于内网的 https）
  - 证书信息不全，缺乏证书机构信息
  - 手机操作系统较旧，没有安装最新加入的根证书
- 怎么办?
  - 自己写证书验证过程

下期内容

- 从 Retrofit 的原理来看 HTTP
- 问题和建议：丢物线
- 网站：hencoder.com
- 微信公众号：HenCoder

HTTPS 连接

1. Client Hello
2. Server Hello
3. 服务器证书 信任建立
4. Pre-master Secret
5. 客户端通知：将使用加密通信
6. 客户端发送：Finished
7. 服务器通知：将使用加密通信
8. 服务器发送：Finished

SSL / TLS 版本
非对称加密算法
对称加密算法
Hash 算法

客户端

Master Secret
客户端密钥
服务器密钥
客户端 MAC Secret
服务器 MAC Secret

应用层消息：@R@#$RER@#$@#ER

SSL / TLS 版本
非对称加密算法
对称加密算法
Hash 算法

服务器

Master Secret
客户端密钥
服务器密钥
客户端 MAC Secret
服务器 MAC Secret