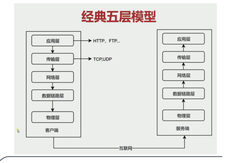

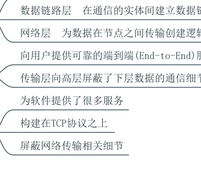

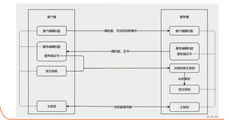

- HTTP协议原理+Nginx+实践
  - 网络协议分层
    - 低三层
      - 物理层　主要作用是定义物理设备如何传输数据
      - 数据链路层　在通信的实体间建立数据链路连接
      - 网络层　为数据在节点之间传输创建逻辑链路
    - 传输层
      - 向用户提供可靠的端到端(End-to-End)服务
      - 传输层向高层屏蔽了下层数据的通信细节
    - 应用层
      - 为软件提供了很多服务
      - 构建在TCP协议之上
      - 屏蔽网络传输相关细节
  - HTTP协议发展历史
    - HTTP/0.9
      - 只有一个GET命令
      - 没有HEADER等描述数据的信息
      - 服务器发送完毕就关闭TCP连接
    - HTTP/1.0
      - 增加了很多命令
      - 增加了status code 和 header
      - 多字符集支持，多部分发送，权限，缓存等
    - HTTP/1.1
      - 持久连接
      - pipeline
      - 增加了host和其他的一些命令
    - HTTPS
      - 加密流程
      - 加密
        - 公钥
        - 私钥
    - HTTP/2
      - 所有数据以二进制传输
      - 同一个连接发送多个请求不需要按照顺序
      - 头信息压缩以及推送等提高效率的功能
      - 特点
        - 信道复用
        - 分帧传输
        - Server Push
  - URI
    - Uniform Resource Identifier / 统一资源标识符
    - 用来唯一标识互联网上的信息资源
    - Location / 统一资源定位符
    - 包括URL URN
      - URL
        - http://user:password@host.com:port/path?query=string#hash
        - 此类格式都叫做URL，比如FTP协议
      - URN
        - 永久统一资源定位符
        - 在资源移动之后还能被找到
        - 目前还没有非常成熟的使用方案
  - HTTP
    - HTTP的三次握手
    - HTTP报文
      - HTTP CODE
        - 定义服务器对请求的处理结果
        - 各个区间的CODE有各自的语义
      - 允许方法
        - GET
        - HEAD
        - POST
        - 不需要使用预请求
      - 允许Content-Type
        - text-plain
        - multipart/form-data
        - application/x-www-form-urlencoded
        - 不需要使用预请求
    - Headers
      - Cookie Session
        - 通过Set-Cookie设置
        - 下次请求会自动带上
        - 键值对，可设置多个
        - 属性
          - max-age　expires　设置过期时间
          - secure　只在https的时候发送
          - httponly 无法通过document.cooke访问
      - Cache-Control
        - max-age
        - public
          - s-maxage
          - max-stale
        - private　只有发起请求的浏览器可以缓存
        - no-cache
        - no-store
        - must-revalidate
        - proxy-revalidate
        - no-transform
        - 缓存验证
          - last-modified + if-modified-since　对比上次修改时间验证资源是否需要更新
          - etag + if-none-match　对比资源的签名判断是否使用缓存
      - 数据协商
        - Accept
          - Accept-Encoding
          - Accept-Langusge
        - User-Agent
        - Content
          - Type
          - Encoding
          - Language
      - Content-Security-Policy
        - 作用
          - 限制资源获取
          - 报告资源获取越权
        - 限制方式　default-src
    - Redirect
      - 301　永久重定向
      - 302　临时重定向
  - Nginx配置
    - 设置代理

```
server {
  listen       80;
  server_name  test.com;

  location / {
    proxy_pass http://127.0.0.1:8888;
    proxy_set_header Host $host;
  }
}

server {
  listen       80;
  server_name  a.test.com;

  location / {
    proxy_pass http://127.0.0.1:8888;
```

    - 设置缓存

```
proxy_cache_path cache levels=1:2 keys_zone=my_cache:10m

server {
  listen       80;
  server_name  test.com;

  location / {
    proxy_pass http://127.0.0.1:8888;
    proxy_set_header Host $host;
  }
}

server {
  listen       80;
  server_name  a.test.com;
```

    - 生成公钥和私钥
    - 部署HTTPS

```
proxy_cache_path cache levels=1:2 keys_zone=my_cache:10m;

server {
  listen       80 default_server;
  listen       [::]:80 default_server;
  server_name  test.com;
  return 302 https://$server_name$request_uri;
}

server {
  listen       443;
  server_name  test.com;

  ssl on;
  ssl_certificate_key ../certs/localhost-privkey.pem;
  ssl_certificate ../certs/localhost-cert.pem;
```

```
server {
  listen       443;
  server_name  test.com;

  ssl on;
  ssl_certificate_key ../certs/localhost-privkey.pem;
  ssl_certificate ../certs/localhost-cert.pem;

  location / {
    proxy_cache my_cache;
    proxy_pass http://127.0.0.1:8888;
    proxy_set_header Host $host;
  }
}
```

    - 建立在HTTPS基础之上
    - 部署HTTP2

```
server {
  listen       443 http2;
  server_name  test.com;
  http2_push_preload on;

  ssl on;
  ssl_certificate_key ../certs/localhost-privkey.pem;
  ssl_certificate ../certs/localhost-cert.pem;
```