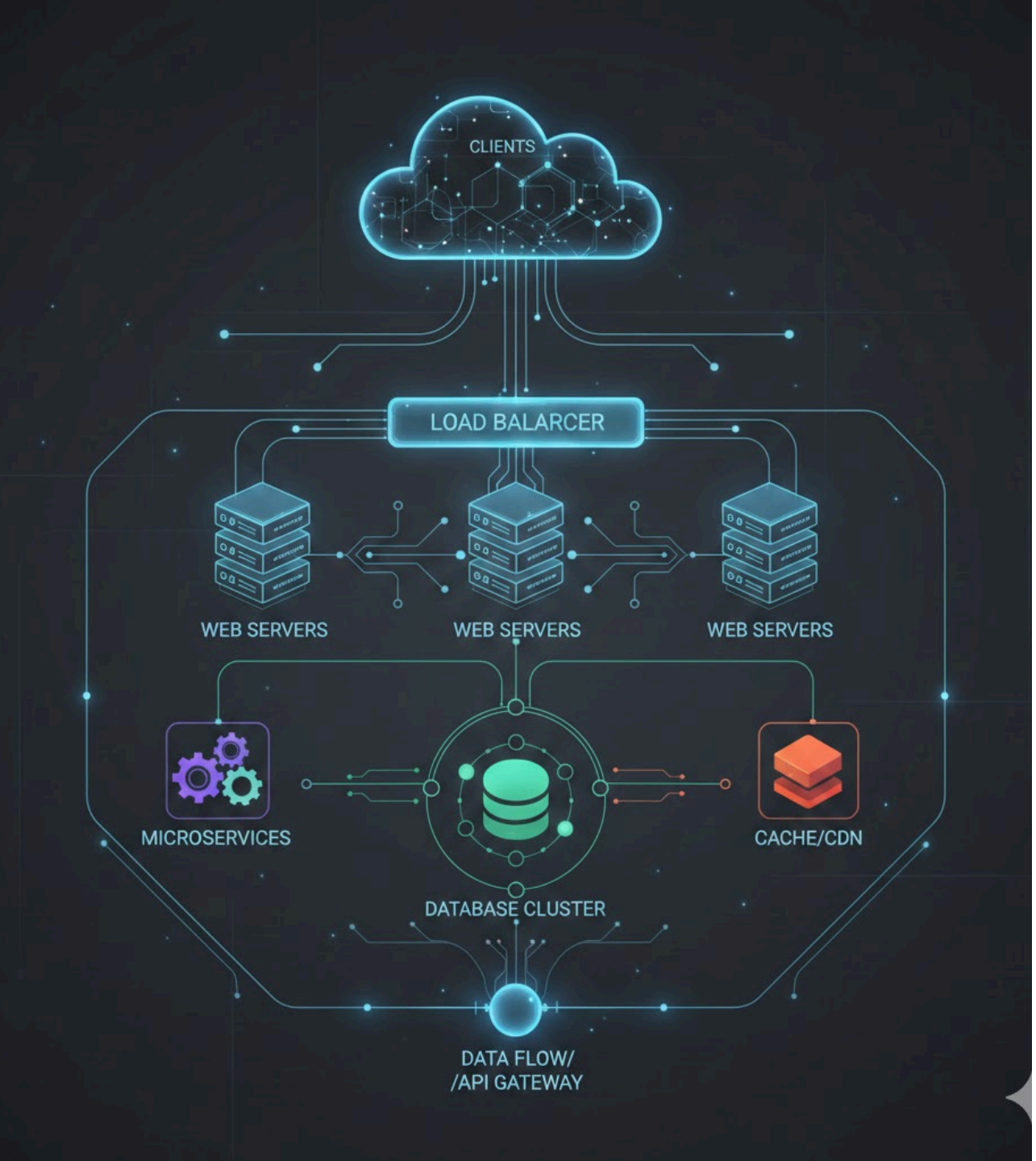
TÓPICO 02 - ARQUITETURA DA WEB

Backend - Professor Ramon Venson - SATC 2026.1



O que é a arquitetura da Web?

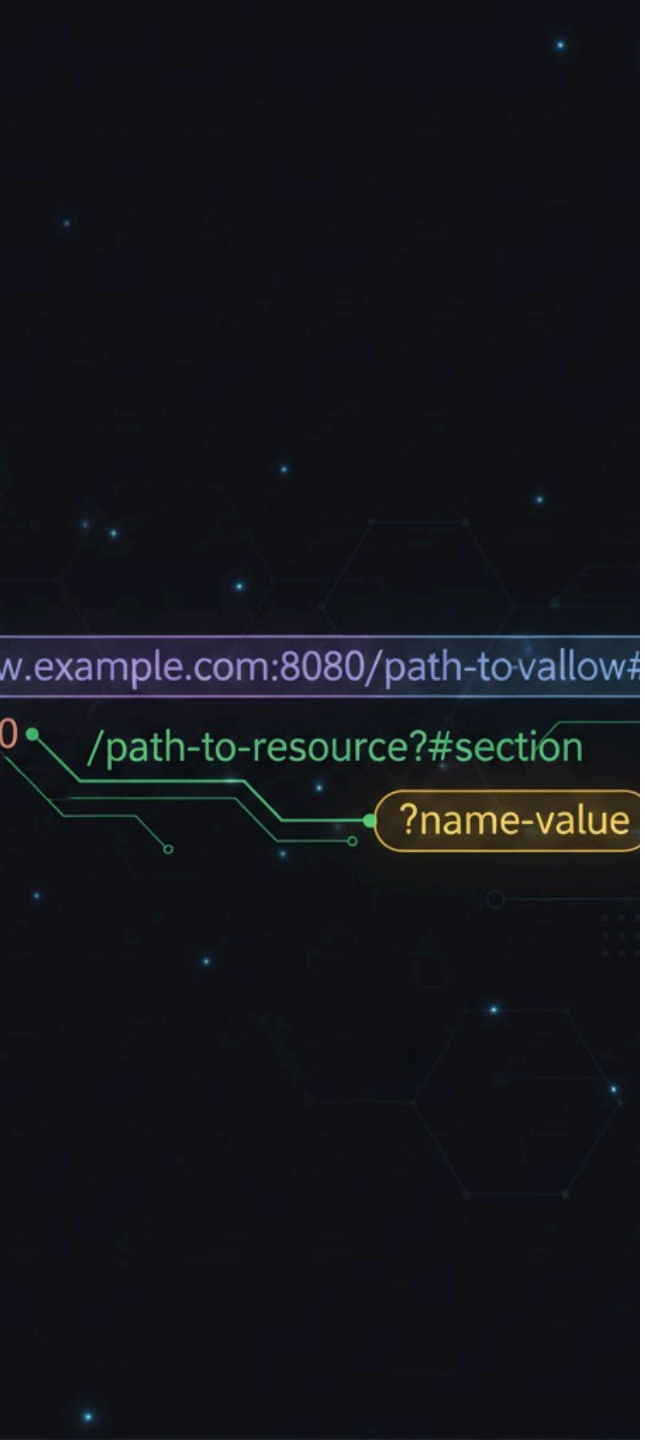
- Conjunto de tecnologias e padrões que fazem a Web funcionar
- Permite comunicação entre pessoas, sistemas e serviços no mundo todo
- Baseada em protocolos de rede e em documentos interligados

Objetivos

- Entender os pilares da Web
- Compreender o modelo cliente-servidor
- Ver como funciona uma requisição HTTP
- Entender por que usamos TLS (HTTPS)
- Diferenciar servidor web e servidor de aplicação

Fundamentos da Arquitetura da Web

- A Web é construída para ser:
  - **Simples**
  - **Universal**
  - **Escalável**
- Baseada em recursos técnicos (URI, HTTP, DNS, TLS)



URI (Uniform Resource Identifier)

- Forma padrão de **identificar recursos** na Web
- Diz **onde** está o recurso e **como** acessá-lo
- Exemplo de URL (um tipo de URI):

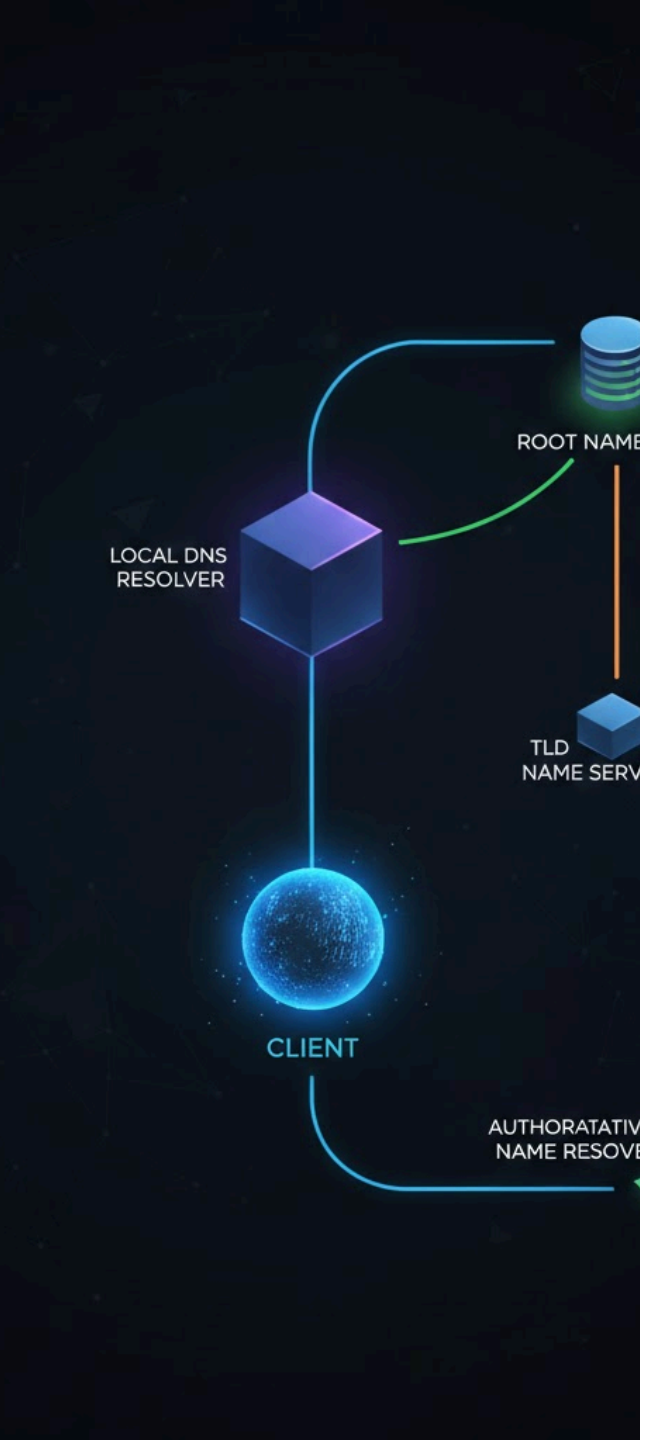
```
https://exemplo.com/artigos/123?ref=homepage
```

- Esquema: `https`
- Domínio: `exemplo.com`
- Caminho: `/artigos/123`
- Query / parâmetros: `ref=homepage`

HTTP (HyperText Transfer Protocol)

- Protocolo que define **como cliente e servidor conversam**
- Usa o modelo **requisição → resposta**
- Suporta vários métodos (GET, POST, PUT, DELETE, …)
- Respostas trazem **códigos de status** (200, 404, 500, …)
- Exemplo simples:

```
GET /index.html HTTP/1.1
Host: exemplo.com
```



DNS (Domain Name System)

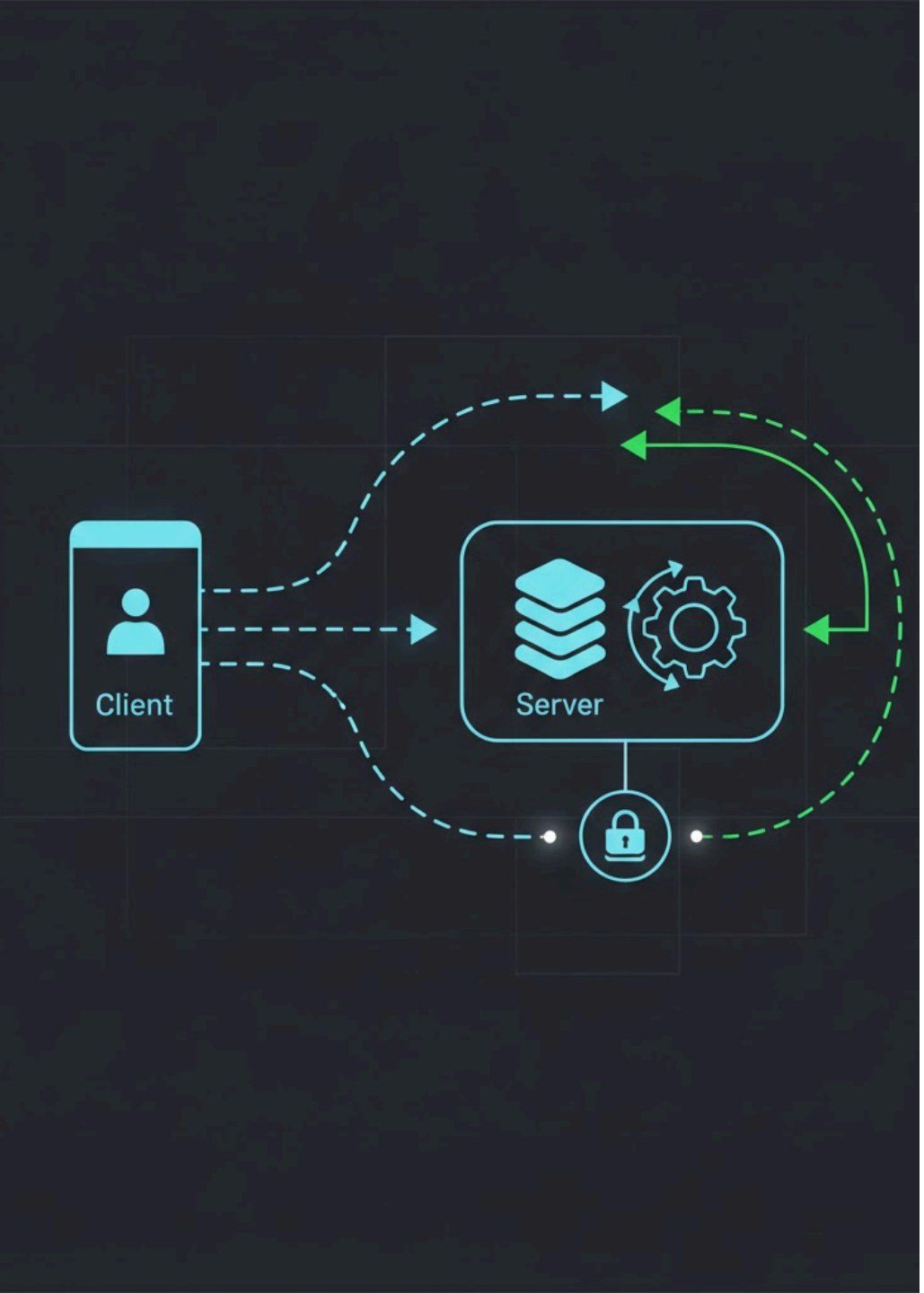
- “Lista telefônica” da Internet
- Traduz **nomes fáceis de lembrar** em **endereços IP**
- Exemplo:

```
exemplo.com → 2.2.2.2
```

- Sem DNS, teríamos que memorizar apenas números
- É usado sempre que acessamos um site por nome

TLS (Transport Layer Security)

- Protocolo que adiciona **segurança** à conexão
- Garante:
  - **Confidencialidade**: dados criptografados
  - **Integridade**: detecção de alterações
  - **Autenticidade**: confirma com quem estamos falando
- Quando combinado com HTTP, temos **HTTPS** (`https://`)
- Usado em:
  - Login, transações financeiras, dados sensíveis em geral



Modelo cliente-servidor

- Base da arquitetura da Web
- **Cliente**: faz a requisição (normalmente o navegador)
- **Servidor**: recebe a requisição, processa e responde
- Comunicação usando HTTP sobre TCP/IP
- Cada recurso é identificado por um URI

Fluxo simplificado de uma requisição (1/2)

1. O cliente recebe ou digita um URI
2. O domínio é resolvido via **DNS** → IP do servidor
3. Cliente abre conexão TCP com o servidor (porta 80 ou 443)
4. Opcionalmente, é feito o **handshake TLS** (para HTTPS)
5. Cliente envia a **requisição HTTP**

Fluxo simplificado de uma requisição (2/2)

6. Servidor processa a requisição
7. Servidor envia a **resposta HTTP** com status e conteúdo
8. Cliente recebe a resposta e renderiza a página
9. Conexão pode ser **fechada** ou **mantida aberta**
10. Novas requisições podem seguir o mesmo fluxo

Exemplo de resposta HTTP

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <head><title>Exemplo</title></head>
  <body><h1>Olá, Mundo!</h1></body>
</html>
```

- `200 OK`: requisição bem sucedida
- `Content-Type`: tipo do conteúdo enviado
- Corpo: HTML que o navegador vai renderizar

Conexões persistentes (keep-alive)

- No HTTP/1.1, a mesma conexão TCP pode atender **várias requisições**
- Isso reduz **latência** e **custo** de abrir novas conexões
- Cabeçalho usado:

```
GET /index.html HTTP/1.1
Host: exemplo.com
Connection: keep-alive
```

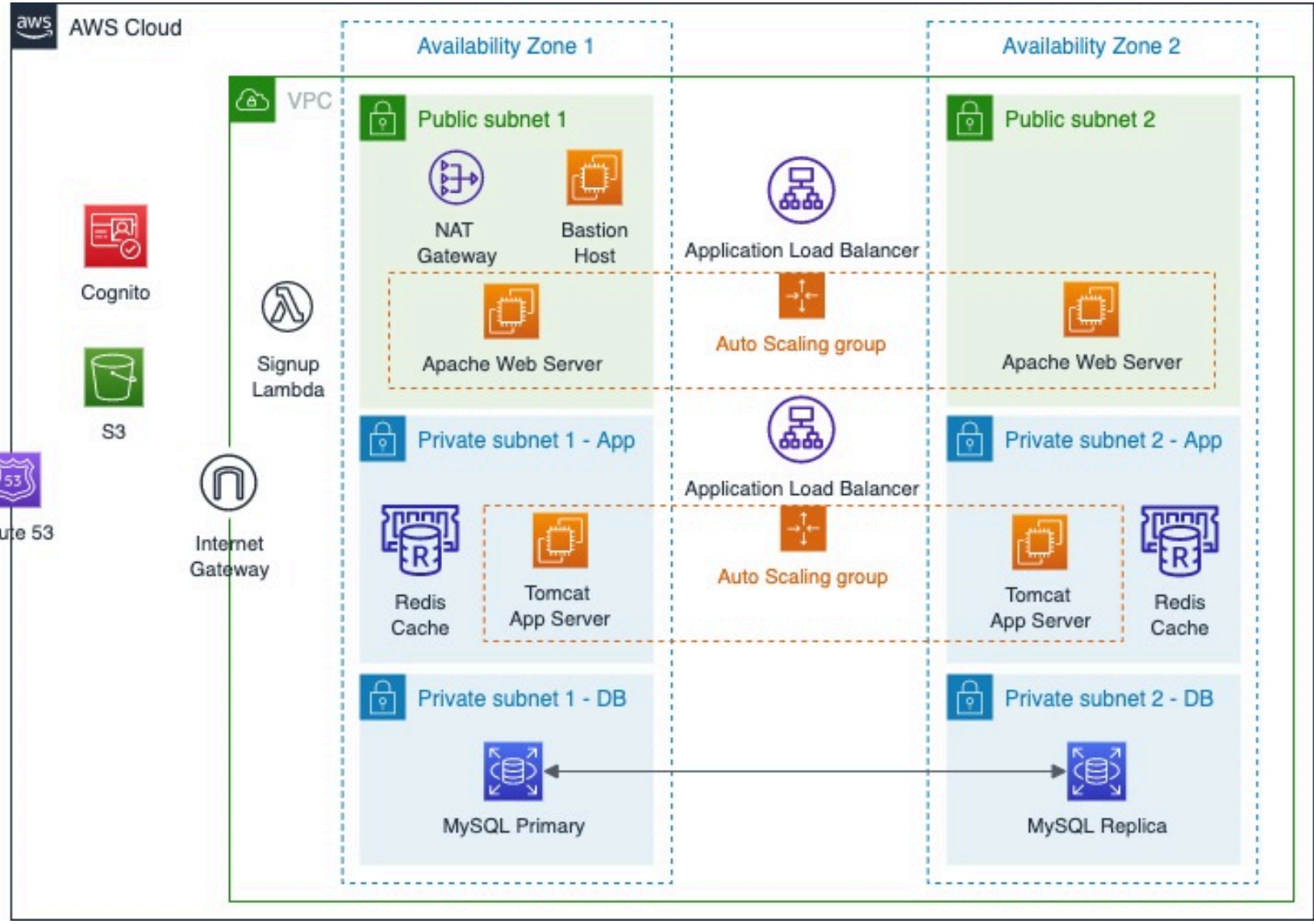
- Servidor pode manter a conexão aberta por algum tempo

Handshake TLS (visão geral)

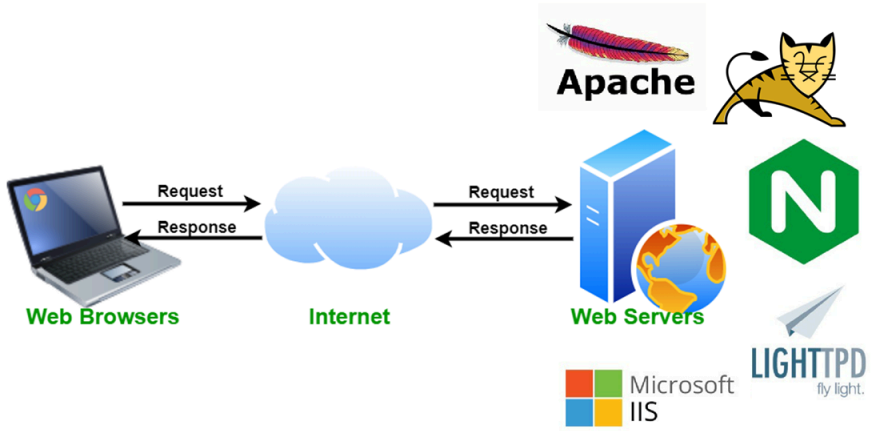
1. Cliente envia informações iniciais (algoritmos suportados, número aleatório)
2. Servidor responde com:
    - Seu número aleatório
    - Certificado digital
    - Algoritmo escolhido
3. Cliente verifica o certificado
4. Cliente e servidor geram uma **chave de sessão** e passam a criptografar a comunicação

Infraestrutura

A web é composta por uma infraestrutura global de servidores, data centers, redes e serviços



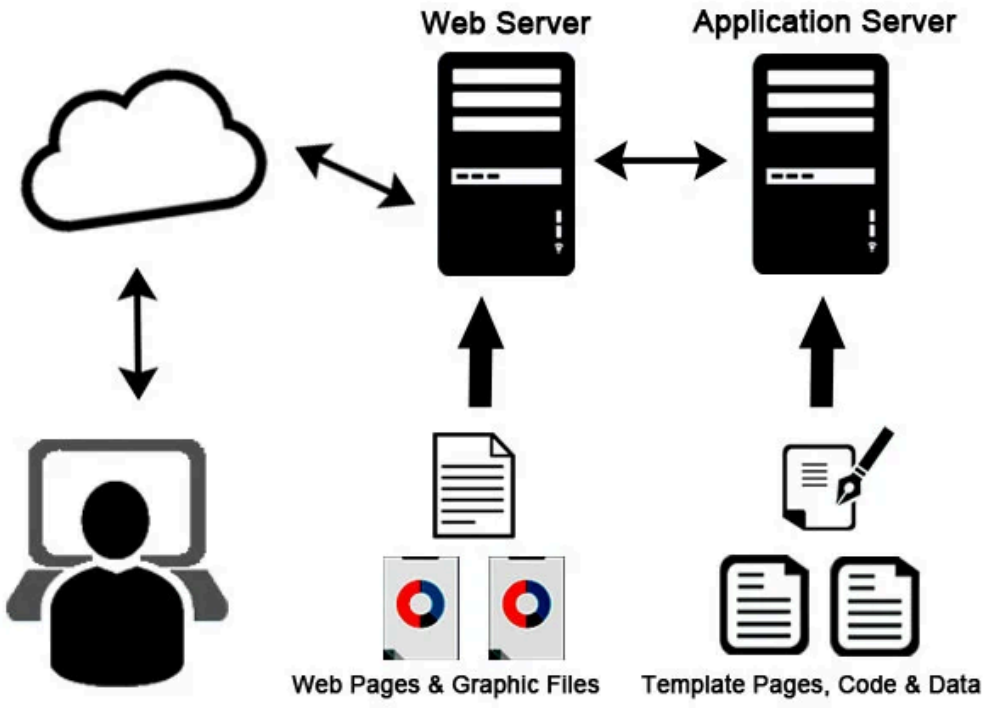
Servidor web



- Software que processa requisições HTTP
- Entrega:
  - Arquivos estáticos (HTML, CSS, JS, imagens)
  - Respostas geradas por aplicações
- Componentes típicos:
  - Processamento de requisições
  - Gerenciamento de recursos
  - Segurança (autenticação, autorização)
  - Logs

Servidor de aplicação

- Ambiente para executar **aplicações web dinâmicas**
- Focado na **lógica de negócio**
- Faz ponte entre:
  - Requisições HTTP
  - Código da aplicação
  - Banco de dados e outros serviços
- Exemplos de uso: APIs REST, aplicações corporativas, servidores de jogos, etc.



Exemplos de frameworks

- **Java / Spring Boot**
  - Usa um servidor como **Tomcat** embutido
- **Python / Django, Flask**
  - Executados em servidores WSGI/ASGI (Werkzeug, Gunicorn, Uvicorn)
- **Node.js / Express.js**
  - O próprio Node.js atua como servidor de aplicação