

# Webアプリケーション概論

# 目次

1. Webアプリケーションとは
2. Webアプリケーションの仕組み
  1. Webアプリケーションのシステム構成
  2. Web開発フレームワーク
  3. Webアプリケーションを公開するためには
3. まとめ
4. 参考文献

# 1. Webアプリケーションとは

**Q. Webアプリケーションとは**

**Q. Webアプリケーションとは  
ブラウザ上で動作するアプリケーションのこと**

# Q. Webアプリケーションとは

ブラウザ上で動作するアプリケーションのこと



各種OS上にインストールして利用するソフトウェア

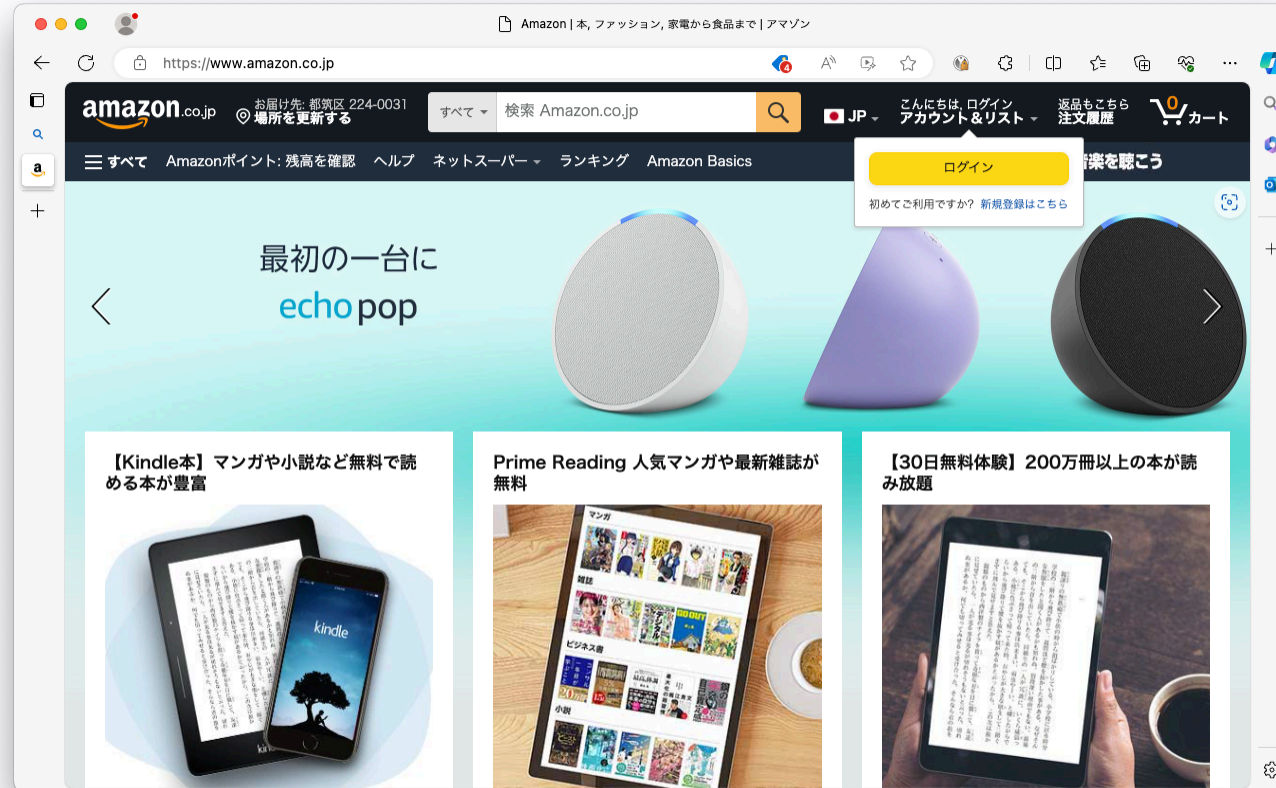
ネイティブアプリケーション

例えば？

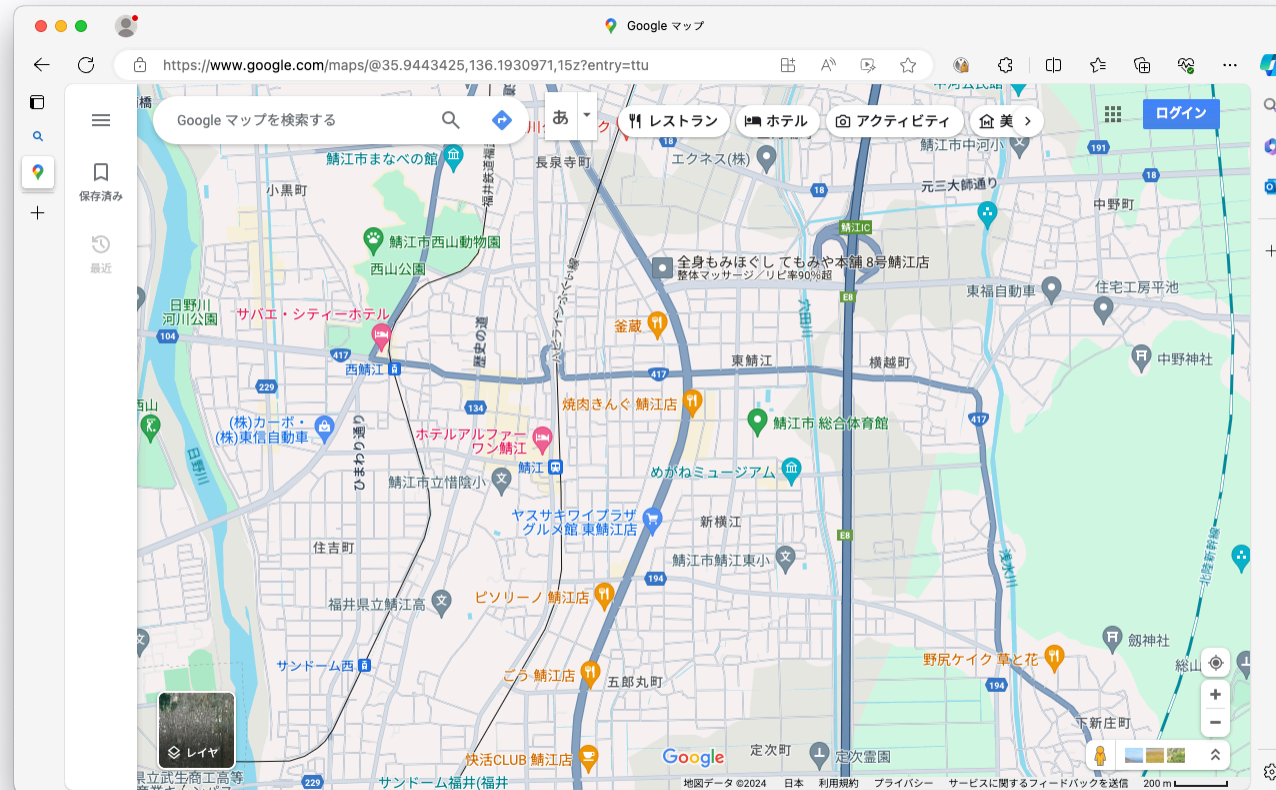
# ● 動画配信



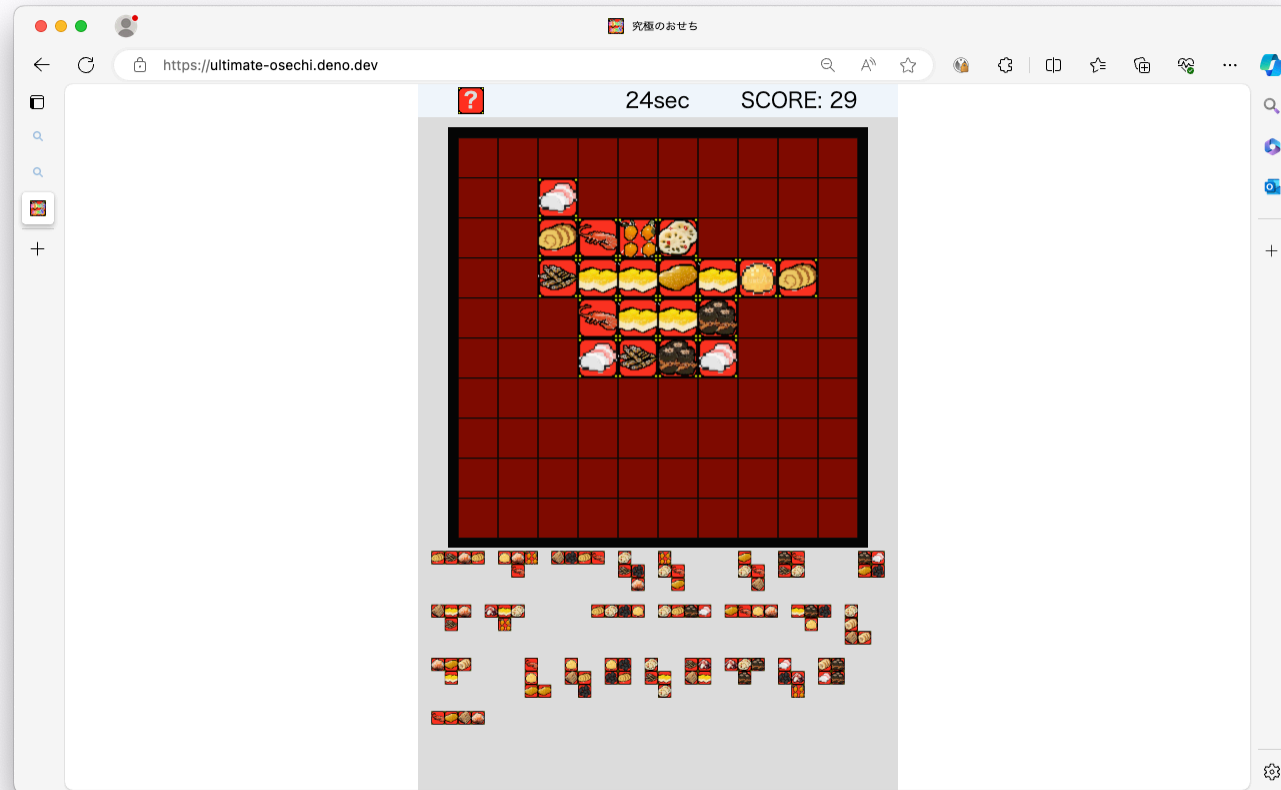
- EC



# ● 地図



- ゲーム



- 表示の動的な変化を伴うホームページ



すべて **Webアプリケーション** といえる

**インターンの目標**

**Webアプリケーションを作る！**

**Webアプリケーションはどんな仕組みで動いているの？**

# 2. Webアプリケーションの仕組み

- 2-1 Webアプリケーションのシステム構成
  - 2-1-1 Web開発の技術スタック
  - 2-1-2 フロントエンドとバックエンド
- 2-2 Web開発フレームワーク
  - 2-2-1 フロントエンドフレームワーク
  - 2-2-2 バックエンドフレームワーク
  - 2-2-3 フルスタックWebフレームワーク
- 2-3 Webアプリケーションを公開するためには
  - 2-3-1 コンテンツ配信とは
  - 2-3-2 コンテンツ配信の方法

## 2-1. Webアプリケーションのシステム構成

- 2-1-1 Web開発の技術スタック
- 2-1-2 フロントエンドとバックエンド

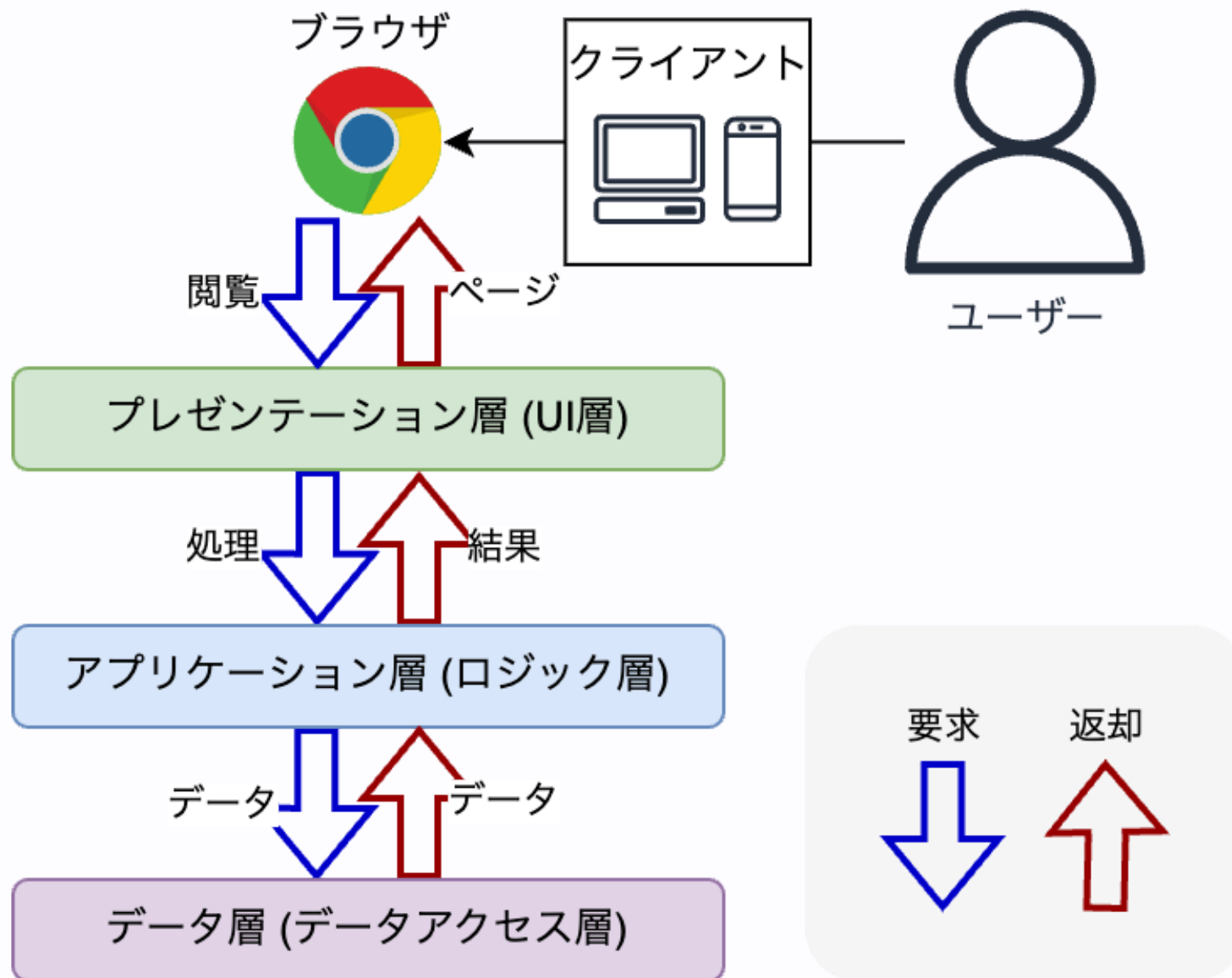
# モダンなWebアプリケーションの構造

## 3層クライアントサーバーシステム

プレゼンテーション層 (UI層)

アプリケーション層 (ロジック層)

データ層 (データアクセス層)



## 2-1-1. Web開発の技術スタック

- 技術スタック
  - 開発に用いるフレームワークやツールなどの組合せのこと
    - プログラミング言語、動作環境のOS、サーバー、サーバーサイドフレームワーク、サーバーサイドライブラリ、API、クライアントサイドフレームワーク、クライアントサイドライブラリ、クライアント、開発支援ツール、運用支援ツール、etc...

## 有名な技術スタック

- LAMP/WAMP
  - Linux/Windows Server, Apache HTTP Server, MySQL, PHP
  - 比較的レガシーな構成でWeb上の情報が多い
- MEAN/MERN/MEVN
  - MongoDB, Express.js, Angular/React/Vue, Node.js
  - サーバーとクライアントを同じ言語で開発できる

# LAMPスタック→3層クライアントサーバーシステム

## LAMPスタック

## 3層C/Sシステム

---

Linux

-

---

Apache HTTP Server

プレゼンテーション層

---

MySQL

データ層

---

PHP

アプリケーション層

## Deno/Deno Deployを活用した技術スタックの例

### 管理するものがDeno Deployとソースのみでシンプル

- クライアントサイド
  - HTML/CSS/JS (バニラ)
- サーバーサイド
  - Deno HTTP Server APIs
- サーバー
  - Deno Deploy

# AWSを活用した技術スタックの例

## Web上に情報が多くあり利用障壁が低い

- クライアントサイドフレームワーク
  - Angular
- クライアントサイド構成
  - Amazon CloudFront
  - Amazon S3
- サーバーサイド構成
  - Amazon API Gateway
    - AWS Lambda (Node.jsランタイム)
  - Amazon DynamoDB

## Microsoft Azure を活用した技術スタックの例

### 既存のMicrosoft製サービスと親和性が高い

- フルスタックフレームワーク
  - Flask + Jinja2
- サーバー
  - Azure App Service (Pythonランタイム)
  - Azure SQL Database

## 2-1-2. フロントエンドとバックエンド

これまでに出た単語とほぼ対応する  
**フロントエンド** = クライアントサイド  
**バックエンド** = サーバーサイド

## フロントエンド

- 役割
  - ユーザーに情報を提示する
  - ユーザーの操作を処理する
- 例) バックエンドから受け取ったデータの表示

## バックエンド

- 役割
  - フロントエンドが要求した処理を行う
  - データの保存や取得を行う
- 例) Webページから送信されたデータの処理

## 2-2. Web開発フレームワーク

- 2-2-1 フロントエンドフレームワーク
- 2-2-2 バックエンドフレームワーク
- 2-2-3 フルスタックWebフレームワーク

## 2-2-1. フロントエンドフレームワーク

### 三大Webフレームワーク (JS系)

- React
- Vue.js
- Angular

他にも

- Svelte
- Solid
- Lit

## 2-2-2. バックエンドフレームワーク

- Express (JS)
- NestJS (JS)
- Spring Boot (Java)

## 2-2-3. フルスタックWebフレームワーク

- Deno Fresh (JS)
- Django (Python)
- Flask + Jinja2 (Python)
- Ruby on Rails (Ruby)
- Laravel (PHP)

## 2-3. Webアプリケーションを公開するため には

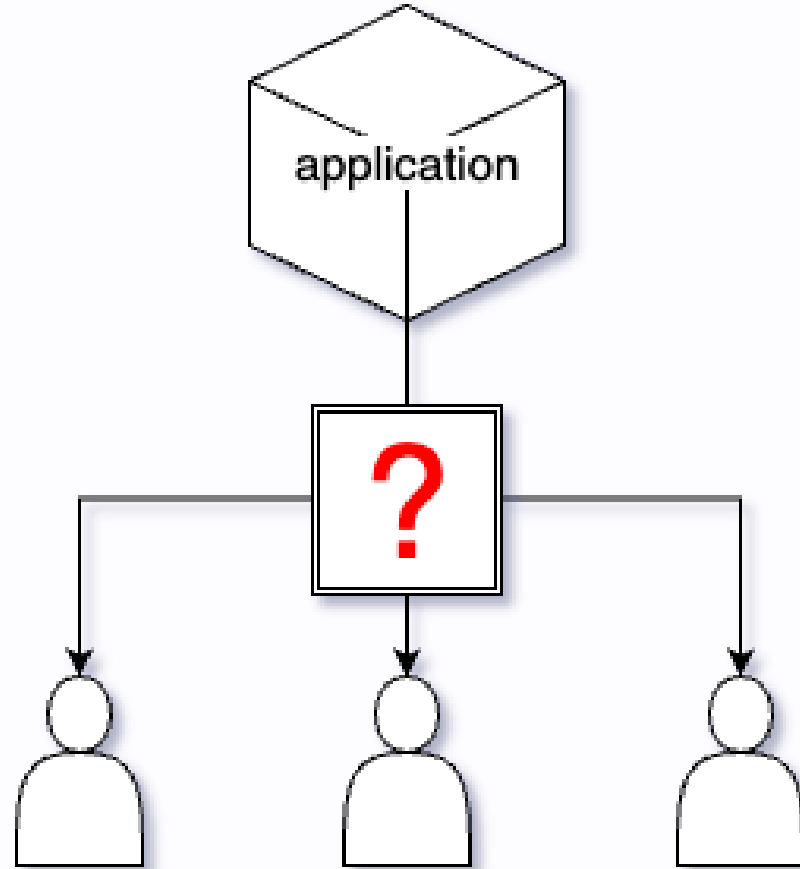
- 2-3-1 コンテンツ配信とは
- 2-3-2 コンテンツ配信の方法

## 2-3-1. コンテンツ配信とは

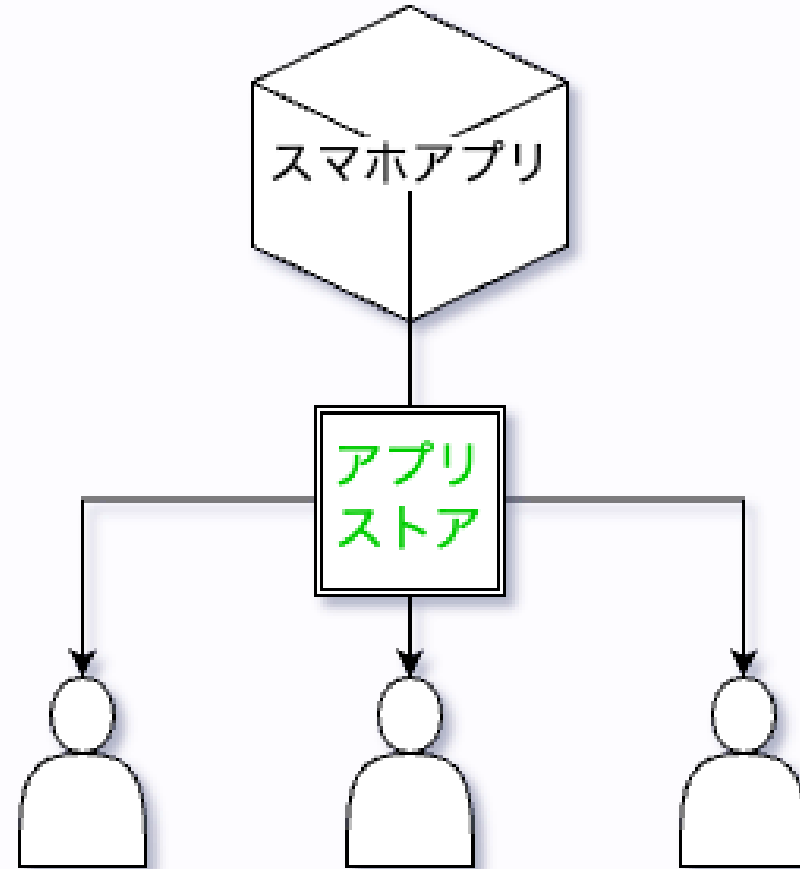
コンテンツは何らかの方法で配信され  
ユーザーがアクセス可能な状態にある必要がある

- ※ コンテンツ:  
アプリケーションやWebページ、画像、動画など

アプリケーションをユーザーが利用するには  
ユーザーにアプリケーションを届ける仕組みが必要

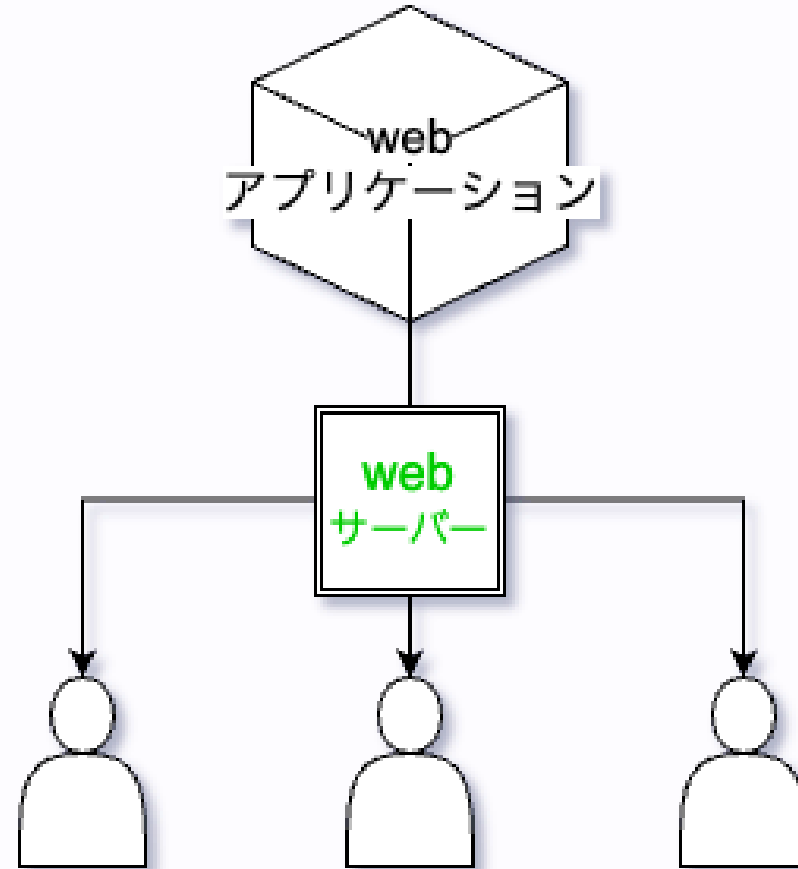


例えば：スマホアプリ  
各種アプリストアを通してユーザーに届けられる



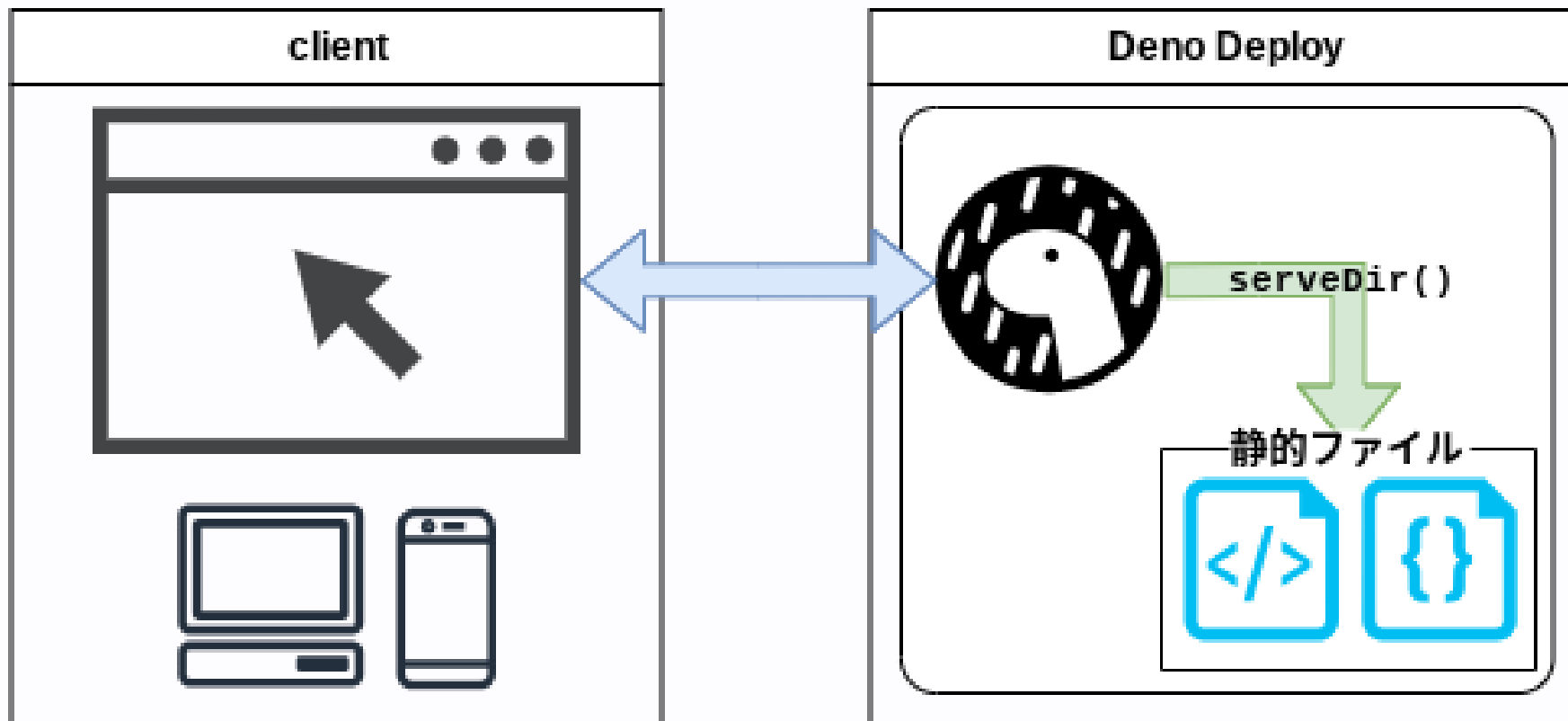
では Web アプリケーション は？

アプリケーションの実行に必要な情報を  
**Webサーバーから配信することでユーザーから利用可能に**

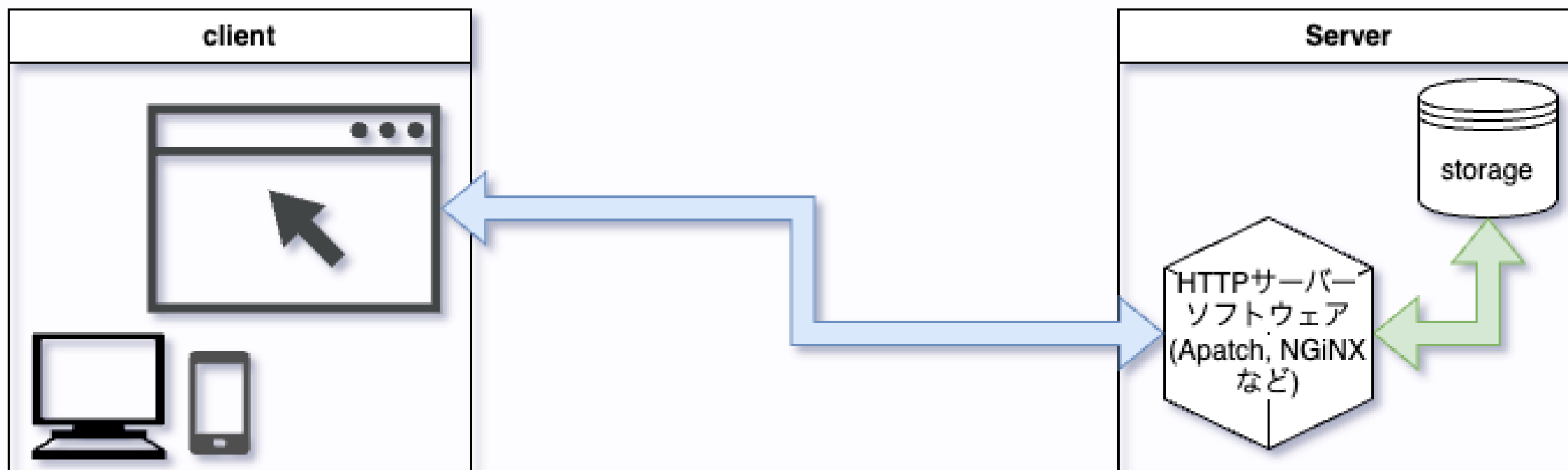


## 2-3-2. コンテンツ配信の方法

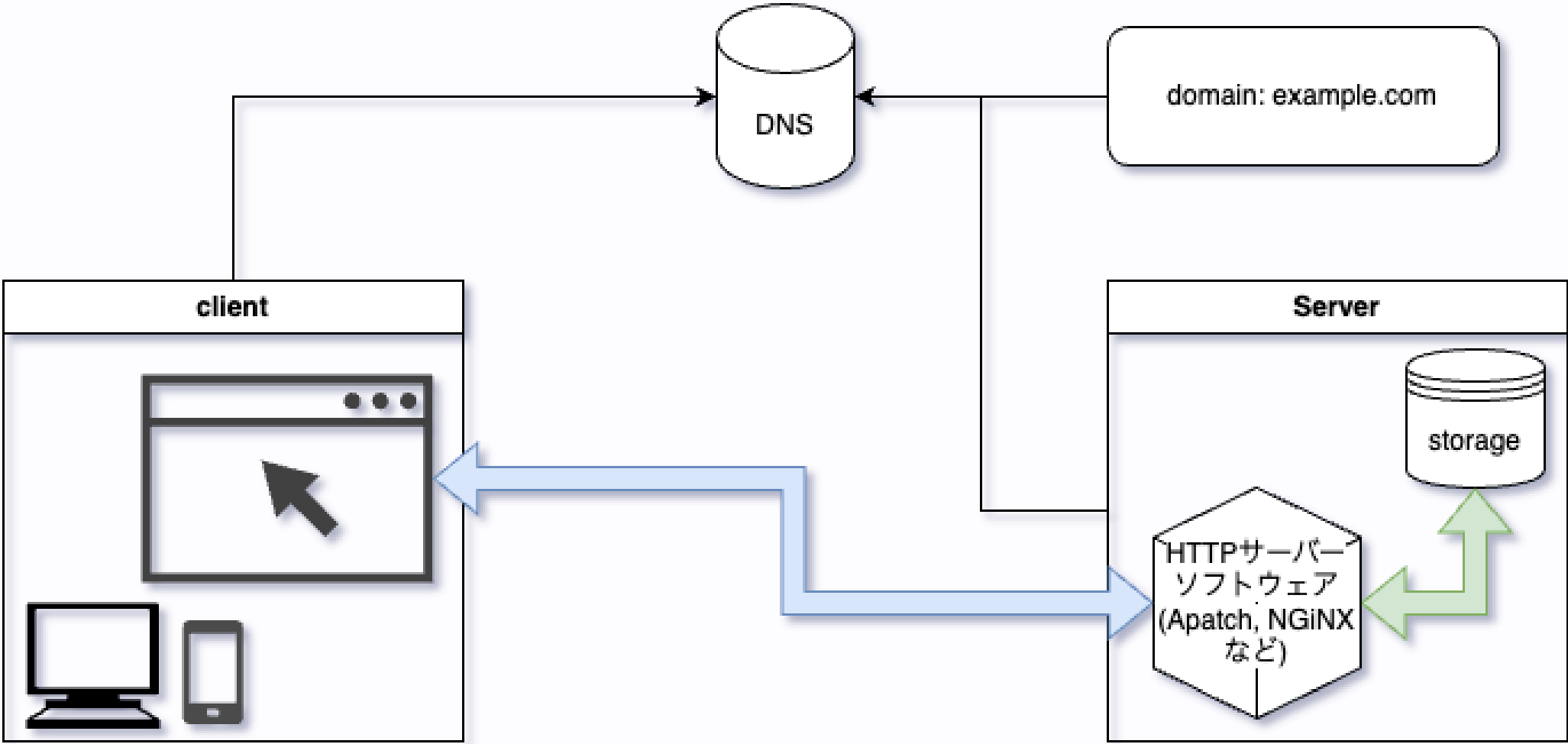
## 実際の構成の例：Deno Deployを利用する場合



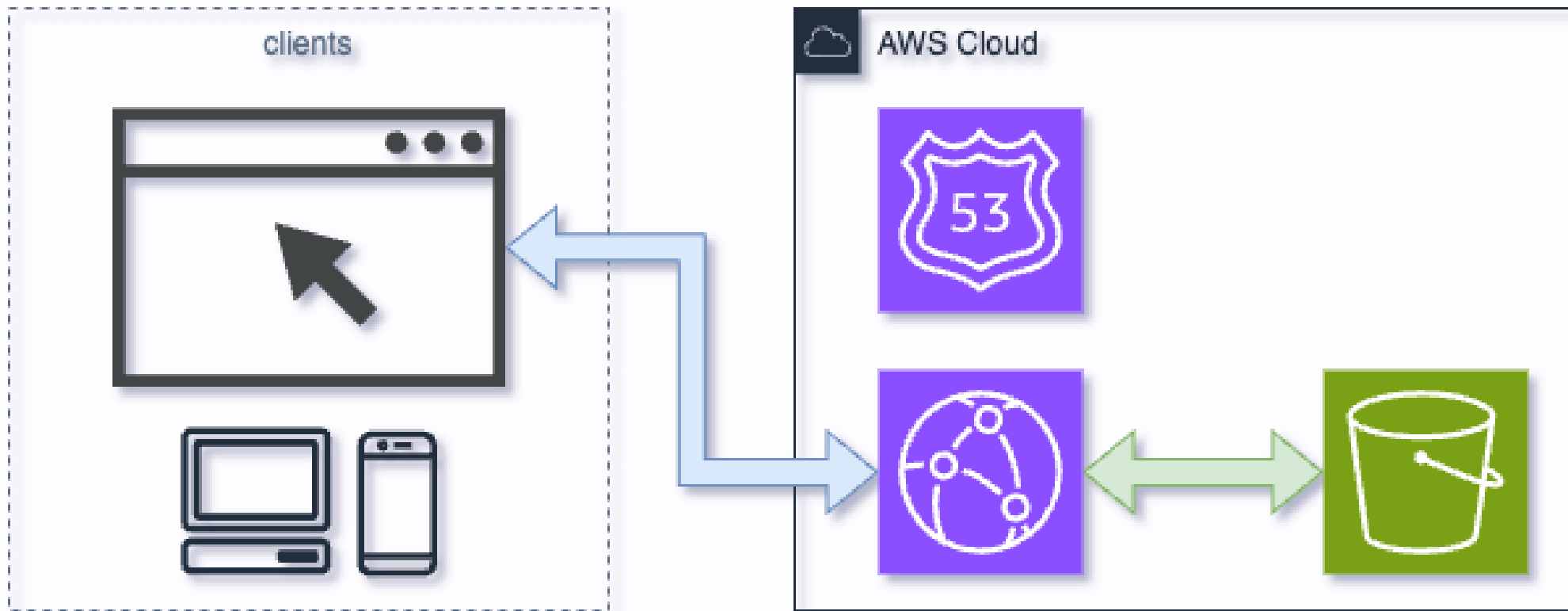
## 実際の構成の例：オンプレミスの場合



より詳細には...



## 実際の構成の例：クラウドを利用する場合(AWS)



# 3. まとめ

- Webアプリケーションとは
  - **ブラウザ上で動作するアプリケーションのこと**
- 世の中のWebアプリケーションはどんな仕組みの上で成り立っているのか
  - 3層クライアントサーバーシステム
  - フロントエンドとバックエンド
  - フレームワーク
  - コンテンツ配信

- これからWebアプリケーションを作っていくために
  - **まずはDeno Deployで作ってみる** ←イマココ
    - 構成がシンプル
    - コンテンツ配信やフレームワークを意識しなくていい

## インターンがおわったあと...

- フレームワークに挑戦する
  - Deno Fresh など
- 静的サイトを作る
  - フレームワーク・コンテンツ配信
- フルスタック開発へ...

**ようこそ、Webアプリケーションの世界へ**

# 4. 参考文献

- ウェブアプリケーションとは? - ウェブアプリケーションの説明 - AWS
- 3層クライアントサーバシステムとは? 概要と仕組みを理解する
- クライアントサイドフレームワークの概要 - ウェブ開発を学ぶ | MDN
- CDNとは? コンテンツ配信の方法とCDNの仕組み | コラム | クラウドソリューション | サービス | 法人のお客さま | NTT東日本

## 4. 参考文献

- [サーバーサイドウェブフレームワーク - ウェブ開発を学ぶ | MDN](#)