# 1　理論

本章では，本研究の前提となる知見や技術について説明する．

## 1.1　Docker

Docker はコンテナ技術の一種で，システムの環境ごとに隔離した空間を用意して独自のディレクトリツリーを提供する．Docker コンテナはそれ単体でシステムが完結しているため，Docker コンテナを別のサーバにコピーして起動させることも容易であり，ポータビリティ性を有している [2]．本論文では Docker を用いて機能ごとに Docker コンテナを作成する．

### 1.1.1　Docker イメージ

Docker イメージは Docker コンテナ作成を支援するために用意されたアーカイブパッケージである．Docker コンテナは独立したシステム実行環境であるため，一から構築するには OS やライブラリなど全てを作り込む必要があり非常に手間がかかる．Docker コンテナ作成に必要なファイルを纏めたアーカイブパッケージが Docker イメージである [2]．Docker イメージは Docker 社が運営している Docker Hub で登録・公開されており，Docker Hub に公開されている Docker イメージをダウンロードして Docker コンテナを作成することが可能である．

Docker イメージには以下の 2 種類ある．

- 基本的な Linux ディストリビューションだけの Docker イメージ
- アプリケーション入りの Docker イメージ

本論文では後者のアプリケーション入りの Docker イメージを用いてシステムの Docker コンテナを作成する．

### 1.1.2　Dockerfile

Dockerfile は自作のイメージを作るもので，ベースとなるイメージに対してどのような変更指示をするかをまとめたファイルである．Dockerfile は「docker build」コマンドを用いてビルドすることで Docker イメージを作成できる [2]．

### 1.1.3 Docker ネットワーク

Docker コンテナは独立しており，1 つの Docker コンテナがサーバマシンのような役割を持っている．Docker コンテナ間が通信するために Docker 内に仮想的なネットワークを構築する必要がある．Docker には以下の 3 種類のネットワークが用意されている．

- bridge ネットワーク
- host ネットワーク
- none ネットワーク

ここでは，本論文で利用する bridge のみ説明し，他 2 種類については説明を省く．

bridge ネットワークでは IP マスカレードを利用してホスト PC が所属する新たな内部ネットワークを構築している． 図 1.1 に Apache の Docker コンテナを 2 つ起動し，それぞれのポートをホスト PC の 8080 と 8081 に接続した時の例を示す．

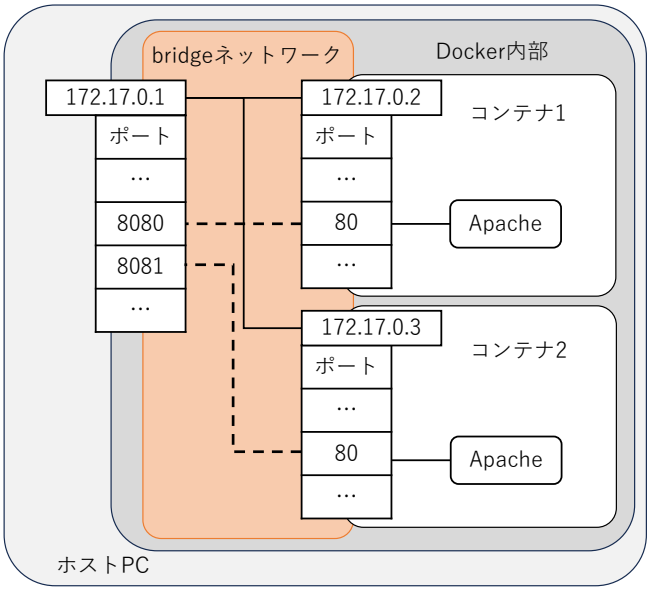


図 1.1: bridge ネットワーク例

### 1.1.4 Docker Compose

Docker Compose は複数の Docker コンテナの作成および Docker コンテナ間のネットワークの作成など，複数の Docker コンテナの作成・停止・破棄の一連の操作

をまとめて実行する仕組みのことである [2]．一連の Docker コンテナやシステムについての定義ファイル (Compose ファイル) は既定では「docker-compose.yml」というファイル名になっている．Compose ファイルを「docker-compose(docker compose)」コマンドで実行するとボリュームやネットワークが作成され，まとめて Docker コンテナを起動できる．停止や削除するときも同様のコマンドで行うことができる．本論文で実装する通知や行動履歴表示といった一連のシステムをまとめて起動・制御するために Docker Compose を用いる．

## 1.2 Node.js

Node.js は JavaScript のランタイム環境であり，Web ブラウザの中でなく単独で JavaScript のプログラムを実行できるようになっているものである [3]．ランタイム環境であることで，従来の JavaScript ではクライアント側の実装でしか使えなかったが，サーバ側でも JavaScript を利用でき，Web 開発の全てを JavaScript のみで完結することができるようになっている．Node.js は Web サーバのための機能を備えており，容易にサーバ側の実装を行うことができる．

### 1.2.1 npm

npm は Node.js 専用のパッケージマネージャの一種である [3]．Node.js では Node.js の機能を拡張する様々なプログラムがパッケージとして配布されている．その多種多様なパッケージをインストール・管理するシステムとして npm が用意されている．

### 1.2.2 Express

Express は Node.js のフレームワークの一種で，Node.js の機能をわかりやすくし基本的な Web アプリケーション機能を比較的軽量で提供できる [3]．

## 1.3 Leaflet.js

Leaflet.js は Volodymyr Agafonkin 氏によって作成された JavaScript のマップライブラリである．マーカの表示やポップアップ，地図の拡大縮小など基本的な地図アプリの機能を提供しており，Edge や Chrome，Firefox，Safari などの基本的なブラウザをサポートしている [5]．

本論文では畑の地図表示および害獣の行動経路を線として，センサノードの位置をマーカとして図示することに用いる．

## 1.4 WebPush

WebPush は Web アプリケーションがサーバからメッセージを受信できる仕組みのことである．Web アプリケーションがフォアグランド状態や読み込まれているかどうかに関わらず，リアルタイムな通知を提供することができる．Web アプリケーションが Push 通知のメッセージを送信するには受信側が「サービスワーカー」を登録している必要がある [6].

WebPush の一般的なモデルを 図 1.2 に示す．WebPush はまず利用者が Push サーバに登録を行い，同時にアプリケーションサーバに登録情報を共有する．この時にサービスワーカーを利用者側に登録する．アプリケーションサーバは共有された登録情報を用いて Push メッセージを Push サーバを経由して利用者に送り，Push 通知を実現する [7].

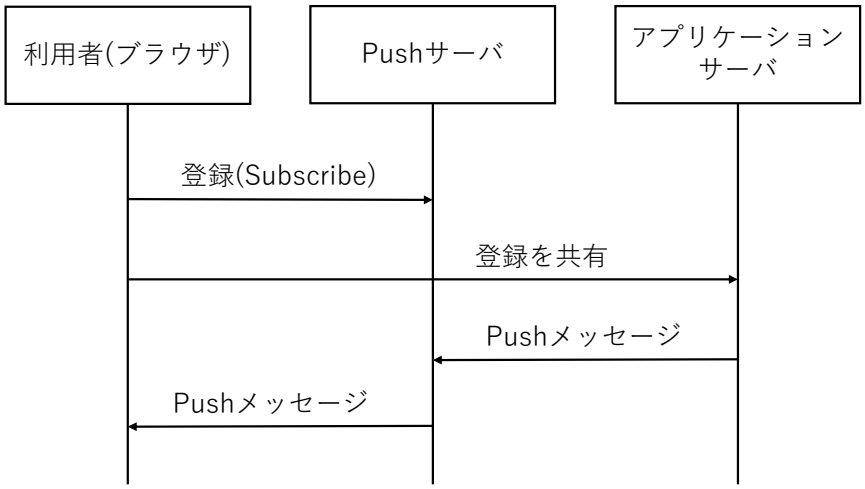


図 1.2: WebPush の模式図

WebPush は基本的なブラウザで保証されていて，表 1.1 に示すブラウザでは保証が確認されている [6].

### 1.4.1 サービスワーカー (Service Worker)

サービスワーカーはあるオリジン (プロトコル・ポート番号・ホストの 3 つの組) とパスに対して登録されたイベント駆動型の JavaScript ファイルである [8].

表 1.1: WebPush が保証されているブラウザ [6]

| パソコン | スマートフォン |
|---|---|
| Chrome | Chrome Android |
| Firefox | Firefox for Android |
| Opera | Opera Android |
| Safari | Safari on iOS |
| Edge | Samsung Internet |

### 1.4.2 VAPID(Voluntary Application Server Identification)

VAPID はアプリケーションサーバを Push サービスに認証・識別してもらうための仕組みである [9]．RFC8030[7] に記載されている WebPush にはアプリケーションサーバを認証する構造が含まれておらず，アプリケーションサーバになりすました通信が行われる問題点がある．ここで VAPID によってアプリケーションサーバの識別をすることで，アプリケーションサーバになりすました通信を防ぐことができるようになる．

## 1.5 Nginx

Nginx は元々 Igor Sysoev 氏が作成した HTTP サーバやリバースプロキシサーバ，メールプロキシサーバといった汎用的な TCP/UDP プロキシサーバである [10]．本論文においてはクラウドサーバ内のリバースプロキシサーバとして活用する．

# 参考文献


[1] 農林水産省, “野生鳥獣による農作物被害状況の推移”, https://www.maff.go.jp/j/seisan/tyozyu/higai/hogai_zyoukyou/attach/pdf/index-31.pdf, 2024年1月18日参照.

[2] 大澤文孝, 浅井尚, “触って学ぶクラウドインフラ docker 基礎からのコンテナ構築”, 日経BPマーケティング, 2020年.

[3] 掌田津耶乃, “Node.js 超入門”, 株式会社 秀和システム, 2017年.

[4] StrongLoop, IBM, “Express - Node.js web application framework”, https://expressjs.com/, 2024年1月18日参照.

[5] Volodymyr Agafonkin, “Leaflet - a JavaScript library for interactive maps”, https://leafletjs.com, 2024年1月18日参照.

[6] Mozilla Foundation, “プッシュ API - Web API | MDN”, https://developer.mozilla.org/ja/docs/Web/API/Push_API, 2024年1月18日参照.

[7] M. Thomson, E. Damaggio, B. Raymor, Ed., “Generic Event Delivery Using HTTP Push”, https://datatracker.ietf.org/doc/html/rfc8030, RFC8030, December 2016, 2024年1月21日参照.

[8] Mozilla Foundation, “サービスワーカー API - Web API | MDN”, https://developer.mozilla.org/ja/docs/Web/API/Service_Worker_API, 2024年1月18日参照.

[9] M. Thomson, P. Beverloo, “Voluntary Application Server Identification (VAPID)”, https://datatracker.ietf.org/doc/html/rfc8292, RFC8292, November 2017, 2024年1月21日参照.

[10] Nginx, “nginx”, https://nginx.org/en/, 2024年1月18日参照.

[11] Mozilla Foundation, “POST - HTTP | MDN”, https://developer.mozilla.org/ja/docs/Web/HTTP/Methods/POST, 2024年1月22日参照.