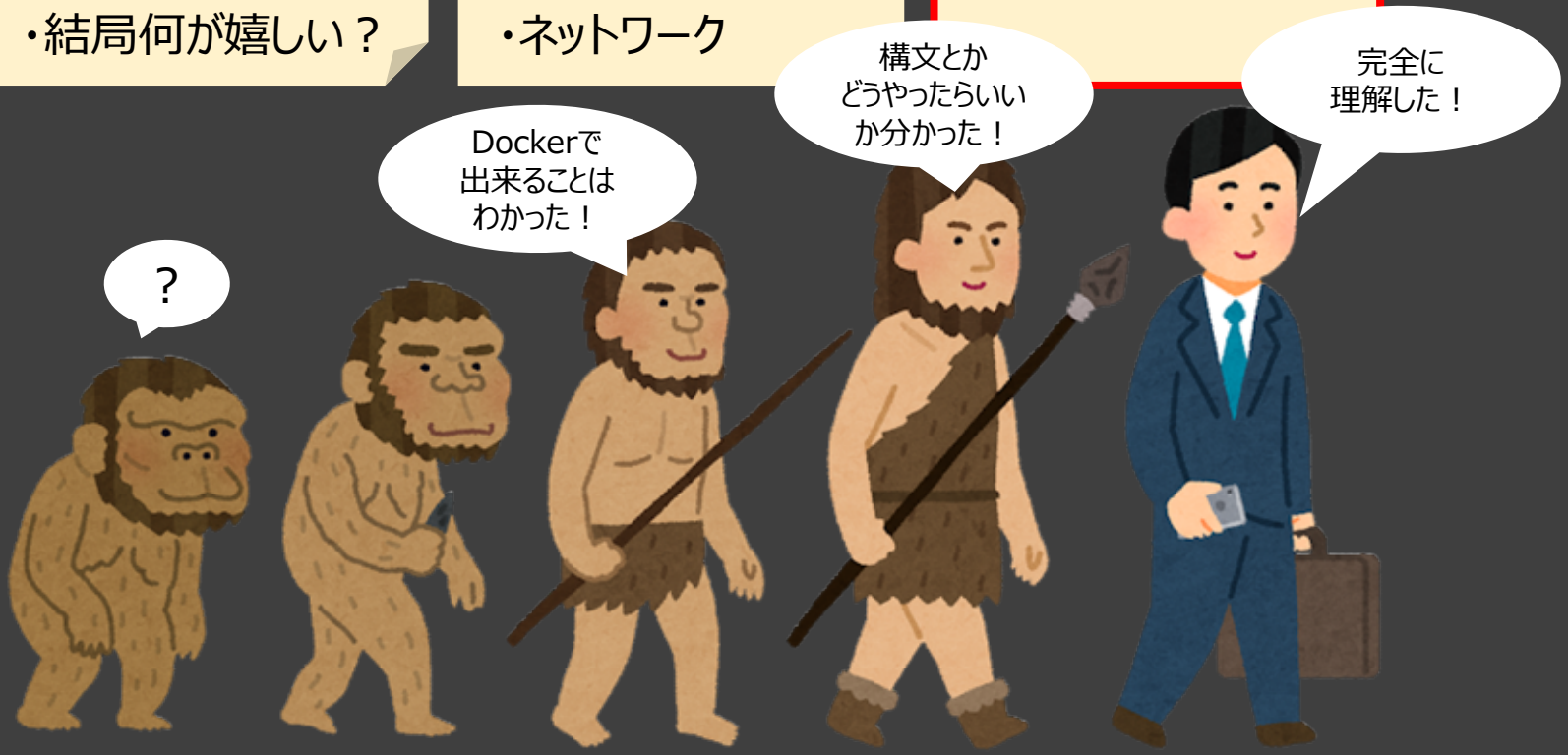
Docker勉強会 vol.1

～ハンズオン～

2020年8月21日

アジェンダと今日のゴール

Dockerを知る	使い方を知る	使ってみる
・Dockerって何？ ・アーキテクチャ ・基礎技術 ・結局何が嬉しい？	・イメージの作り方 ・コンテナ起動、終了 ・ボリューム ・ネットワーク	・ハンズオン



イラストに他意はありません

使ってみる～ハンズオン～

- ハンズオンでやること
  - 既存アプリをコンテナ化する
  - 既存アプリを組み合わせてサービスを構築する
- ハンズオンでやらないこと
  - アプリの開発および変更
  - ホストOSへのライブラリ等のインストール
  - マルチホストでのコンテナ連携
- ハンズオンは2チームに分かれて実施します
  - お題(ケーススタディ)を出します
    - チーム間でシステム構成のすり合わせ
    - チーム毎にモブワーク実施(各チームのチャンネルで実施)
      - ドライバーの画面を共有して作業してください
      - ナビゲータはggったり、質問してドライバーをサポートして下さい
    - 動作確認&情報共有
- ↑を出来たら2巡します

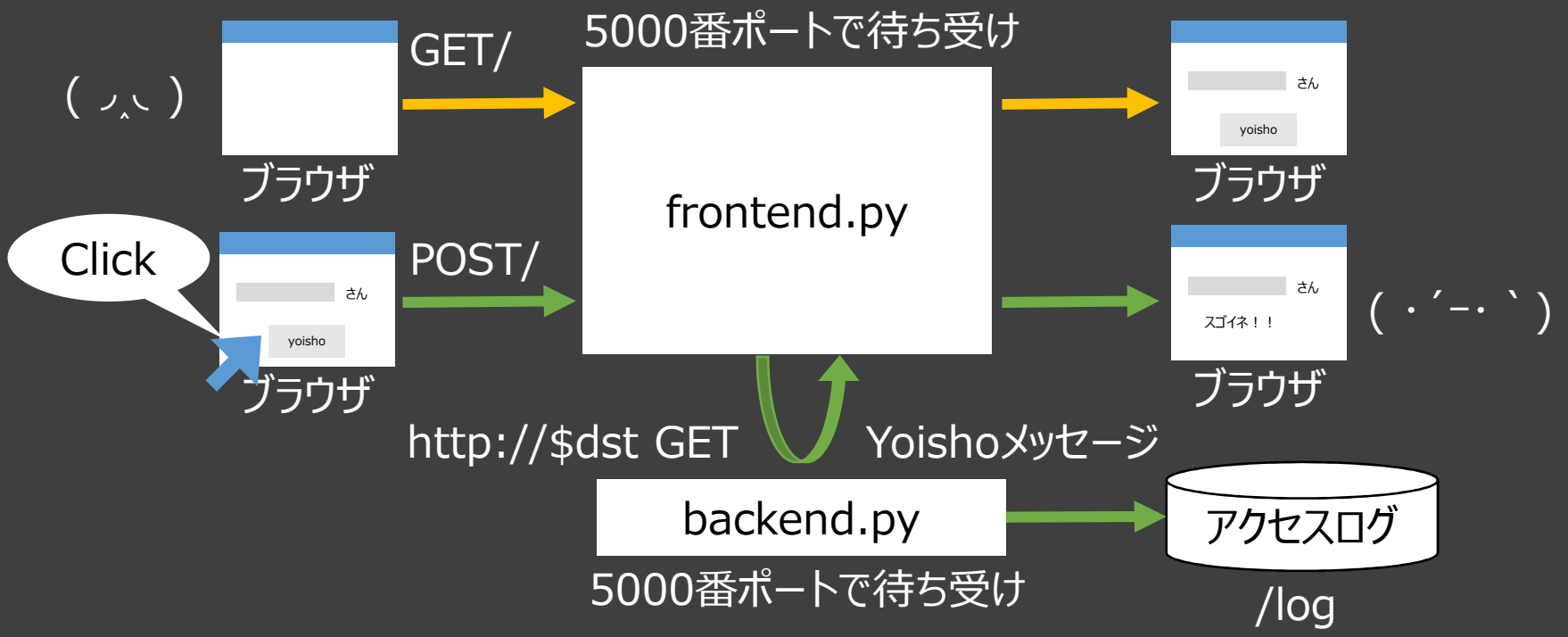
使ってみる～ハンズオン ケーススタディ～

- Masuno SoftWare(通称MSW)では、売り上げ5兆円を目指し、新サービスを立ち上げようとしています！
  - コンテナを組み合わせて以下のサービスを実現してください

パッケージデザイン

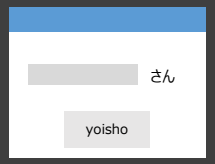
使ってみる～ハンズオン ケーススタディ～

- 必要なアプリは社長が実装済み
  - frontend.py
  - backend.py
  - ↑の2つはpythonライブラリのFlaskを利用して実装



使ってみる～ハンズオン ケーススタディ～

- システム構成すり合わせ
  - 1チーム：frontend.py
  - 2チーム：backend.py



コンテナ
frontend.py

コンテナ
backend.py

テキスト

使ってみる～ハンズオン ケーススタディ～

- コンテナ構築&動作確認を始めて下さい
  - 開発環境へのアクセス
    - http://XXXへアクセス
      - ユーザ名、パスワード共にmasuno
    - ログイン後、新しいターミナルを開き、以下のコマンドを実行
      ssh –l <チーム名> host
      - チーム名：
        - 1チーム：team1
        - 2チーム: team2
  - 制約事項
    - 社長のコード変更禁止！(当たり前だよね!)
    - コンテナのベースイメージはpython:3を使ってください
    - ブラウザからはhttp://XXXx:XXXでアクセスできるようにしてください

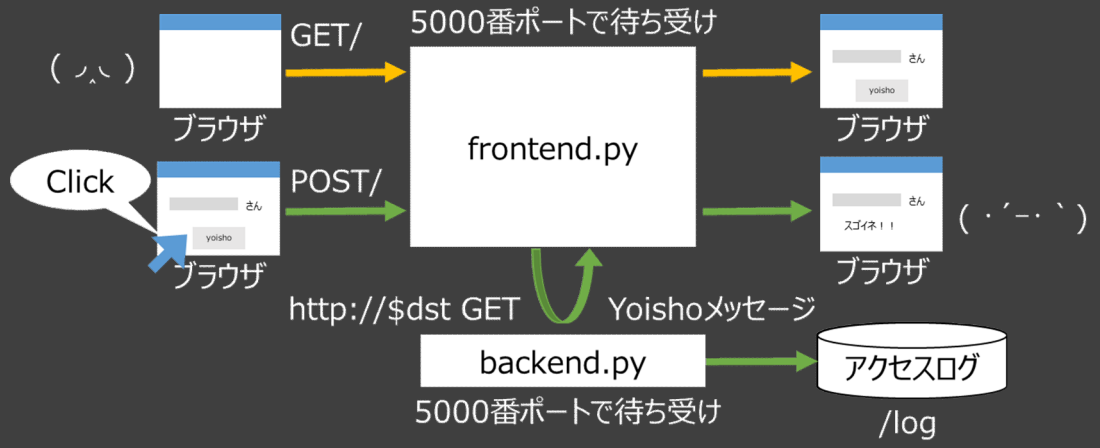
使ってみる～ハンズオン～

- チートシート
  - 各コマンドのリファレンスは公式を確認
    - https://docs.docker.jp/engine/reference/index.html

大項目	項目	コマンド
イメージ関連	イメージのビルド	sudo docker build –t <イメージ名>:<TAG> .
	イメージ一覧	sudo docker images
	イメージ削除	sudo docker rmi <イメージ名>
	<none>:<none>イメージ削除	sudo docker image prune
コンテナ関連	動いているコンテナ一覧	sudo docker ps
	すべてのコンテナ一覧	sudo docker ps –a
	コンテナ起動(バックグラウンド)	sudo docker run –d <イメージ名>:<TAG> [CMD(上書きする場合)]
	コンテナ起動(ターミナルに入る)	sudo docker run –it <イメージ名>:<TAG> bash
	起動中のコンテナのターミナルに入る	sudo docker exec –it <コンテナ名> bash
	コンテナ停止	sudo docker stop <コンテナ名>
	コンテナ削除	sudo docker rm <コンテナ名>
	止まっているコンテナ全削除	sudo docker rm $(sudo docker ps –q –a)
	コンテナのstdout/err表示	sudo docker logs <コンテナ名>
	コンテナの設定確認	sudo docker inspect <コンテナ名>

使ってみる～ハンズオン ケーススタディ～

- 不具合修正と機能追加
  - クレーム発生！
    - ユーザ名が空白の時、「¯¥_(ツ)_/¯」というナメた応答が来る！
  - 機能追加要望！
    - 社長「管理用にアクセスログをWebでみたいなー」
  - 社長「あ、でも僕が書いたコードは修正しないでね」



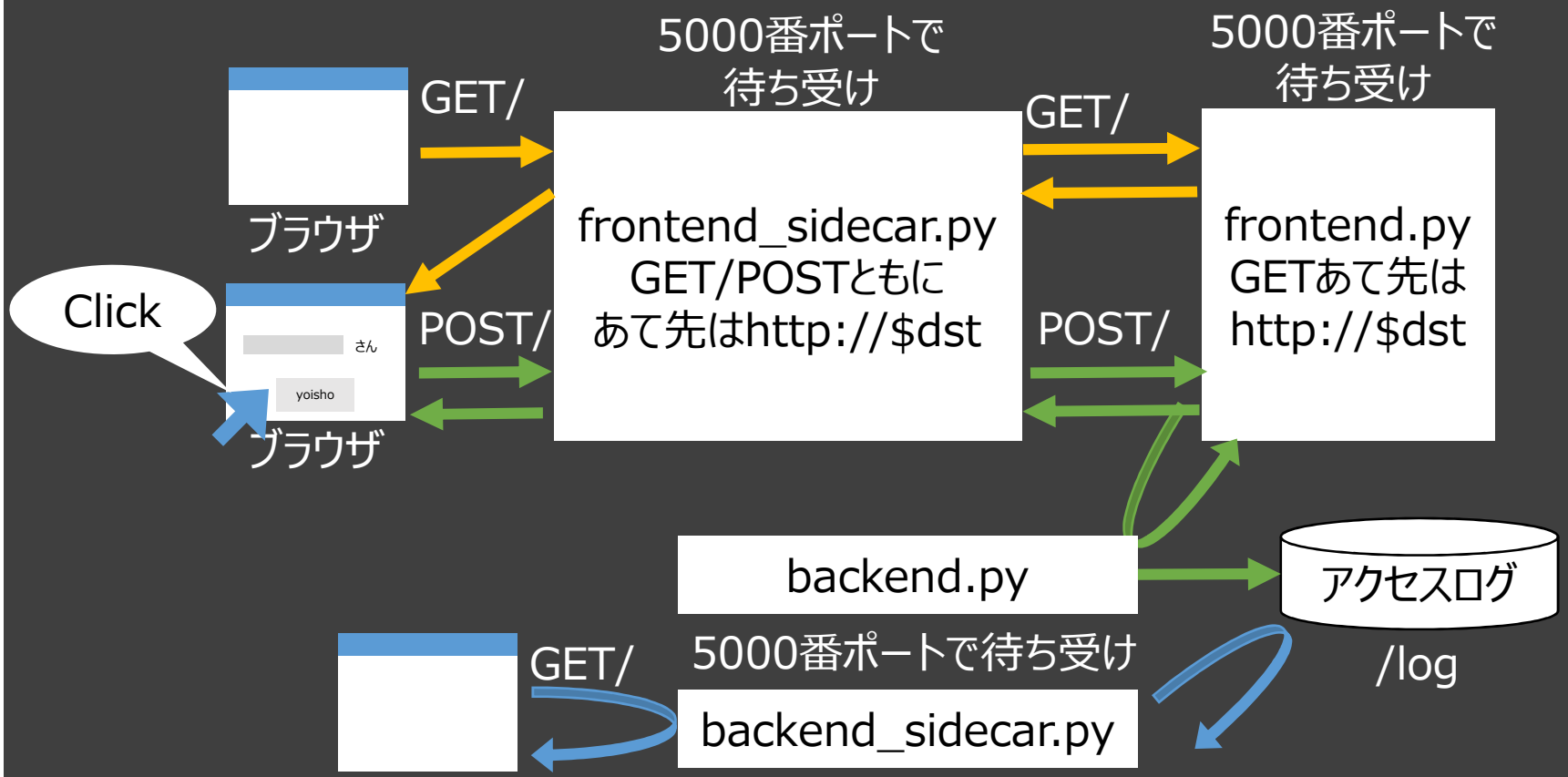
↑をコード修正なく実現できる方法を議論しましょう

使ってみる～ハンズオン ケーススタディ～

- 実現例
  - クレーム発生！
    - ユーザ名が空白の時、「¯¥_(ツ)_/¯」というナメた応答が来る！
    - ⇒frontendの前に中継を入れて入力を補正
  - 機能追加要望！
    - 社長「管理用にアクセスログをWebでみたいなー」
    - ⇒backendのlogを読み込んで応答するwebサーバを追加

使ってみる～ハンズオン ケーススタディ～

- Sidecarパターンで実現してみる
  - frontend_sidecar.py
  - backend_sidecar.py
  - ↑の2つはpythonライブラリのFlaskを利用して実装



使ってみる～ハンズオン ケーススタディ～

- コンテナ構築&動作確認を始めて下さい
  - 制約事項
    - 社長のコード変更禁止！(当たり前だよね!)
    - コンテナのベースイメージはpython:3を使ってください
    - ブラウザからは
      - Yoisho http://XXXx:XXXでアクセス
      - ログ http://XXXX:XXXでアクセス

使ってみる～ハンズオン　ケーススタディ～

- コンテナが増えて、起動順序、ネットワークとかボリューム設定がうざくなかったですか?
  ⇒それ、docker-composeでできます！

```
version: '3'

services:
  frontend_sidecar:
    environment:
      - dst=http://frontend:5000
    build: ./frontend_sidecar
    image: frontend_sidecar:1.0
    container_name: frontend_sidecar
    networks:
      - yoisho-net
    ports:
      - 8080:5000


  frontend:
    depends_on:
      - backend
      - frontend_sidecar
    environment:
      - dst=http://backend:5000
    build: ./frontend
    image: frontend:1.0
    container_name: frontend
    networks:
      - yoisho-net
```

```
backend:
    build: ./backend
    image: backend:1.0
    container_name: backend
    networks:
      - yoisho-net
    volumes:
      - yoisho-log:/log


 backend_sidecar:
    depends_on:
      - backend
    build: ./backend_sidecar
    image: backend_sidecar:1.0
    container_name: backend_sidecar
    networks:
      - yoisho-net
    ports:
      - 8081:5000
    volumes:
      - yoisho-log:/log:ro
```

```
networks:
  yoisho-net:
    driver: bridge

volumes:
  yoisho-log:
    driver: local
```