リスト1　Dockerをインストールする手順

①Dockerのインストール

```
$ sudo curl -fsSL https://get.docker.com/ | sh⏎
```

②Dockerグループにユーザを追加

```
$ sudo usermod -aG docker ${USER}⏎
```

③再ログインしてユーザ権限の確認

```
$ docker --version⏎
$ docker run --rm hello-world⏎
```

④Docker networkの作成

```
$ docker network create mike-net -d bridge⏎
```

⑤Docker版mikeの展開

```
$ cp docker_mike_001.tar.gz  /tmp/docker_mike_001.tar.
gz⏎
$ cd /opt⏎
$ sudo mkdir mike⏎
$ sudo chown pi:docker mike⏎
$ cd mike⏎
$ tar xvzf /tmp/docker_mike_001.tar.gz⏎
```

リスト2　Julius用のDockerfile

```
# ①ベース・イメージとしてDebianのBookwormバージョンを
使用
FROM debian:bookworm

# ②パッケージ・リストを更新
RUN apt update
# サウンド・ライブラリの開発用パッケージをインストール
RUN apt install -y libasound2-dev=1.2.8-1+b1
# ビルド・ツールをインストール
RUN apt install -y build-essential=12.9
# gitおよびgit-lfsを指定バージョンでインストール
RUN apt install -y git=1:2.39.5-0+deb12u1 git-lfs=3.3.
0-1+b5
# unzipと圧縮ライブラリ、SDL2の開発用ライブラリをインス
トール
```

```
RUN apt install -y unzip=6.0-28 zlib1g-dev=1:1.2.13.df
sg-1 libsdl2-dev=2.26.5+dfsg-1

# sudoパッケージをインストール
# juliusをmike権限で実行させる準備
RUN apt install -y sudo=1.9.13p3-1+deb12u1

# git-lfsの設定を行う
RUN git lfs install

# ユーザとグループの名前，UID，GIDを引数として設定
ARG USERNAME=mike
ARG GROUPNAME=user
ARG UID=1000
ARG GID=1000

# ③指定したGIDでグループを作成し，指定したUIDとGIDで
# ユーザを作成，音声グループに追加
RUN groupadd -g $GID $GROUPNAME && \
    useradd -m -s /bin/bash -u $UID -g $GID -G audio $
USERNAME

# 作成したユーザをデフォルト・ユーザとして設定
USER $USERNAME

# ④作業ディレクトリを /opt/mike に設定
WORKDIR /opt/mike/

# コンテナ起動時にbashシェルを実行
CMD ["/bin/bash"]
```

リスト3　Julius用のdocker-compose.yml

```
services:
  julius:    #①サービス名をjuliusに設定
    build: .  # 同一ディレクトリ内のDockerfileからビル
ド
    image: julius-img  # 作成するイメージの名前をjuliu
s-imgに指定
    container_name: julius-cont  # コンテナ名をjulius-
contに設定
    restart: always  # コンテナが停止した場合に自動で再
起動
```

```
    hostname: julius-server  # ②コンテナのホスト名をjulius-serverに設定

    devices:
      - "/dev/snd:/dev/snd"  # ③ホストのサウンド・デバイスをコンテナ内で使用できるよう設定

    volumes:
      - ./opt_mike:/opt/mike  # ④ホストのopt_mikeディレクトリをコンテナ内の/opt/mikeにマウント

    user: root  # ⑤初回のセットアップでroot権限が必要なためrootユーザーで実行
    # コンテナ起動時に初回は `make` と `make install` を実行し、その後mikeユーザでjuliusを起動
    entrypoint: >
      sh -c "
        sh /opt/mike/bin/container-init.sh mike  # 初期化スクリプトの実行、引数 mike は内部juliusの権限付与のため
      "
    tty: true  # TTYを割り当て、シェルの対話モードをサポート

networks:
  default:
    name: mike-net  # ⑥使用するネットワーク名をmike-netに指定
    external: true  # mike-netという外部ネットワークを使用
```

リスト4　スクリプトcontainer-init.sh

```
#!/bin/bash
# ①julius_serverのインストールと起動
# 初期起動の場合はjuliusをGITからcloneしてコンパイルし
# 試験起動する
# ②２回目以降はコンパイルせずにサーバ・モードで起動する
# 初期起動の判定は、julius_serverのカレントにcheckとmadeを
# 記録することで判定している
# ③再セットアップしたいときは，checkとmadeを削除するこ
```

```
JULIUS_HOME=$(cd $(dirname $0);cd .. ;pwd)
echo $JULIUS_HOME
CHECK=${JULIUS_HOME}/check
MADE=${JULIUS_HOME}/made

IS_DOCKER="`echo $JULIUS_HOME | grep "julius_server$"`
"
if [ "$IS_DOCKER" ]; then
USR=${USER}
echo "NORMAL:"$USR
else
USR="mike"
echo "DOCKER:"$USR
fi

ls $CHECK
#rm $CHECK
#rm $MADE

if [ ! -e "$CHECK" ]; then
    echo "セットアップ"
    cd  $JULIUS_HOME
    # セットアップ済みの印としてcheckファイルを作成
    touch $CHECK
    if [ ! -e "$MADE" ]; then
        echo "コンパイル開始"
        # クローン済みのdictation-kitディレクトリが
        # あれば削除
        rm -fr dictation-kit
        git clone https://github.com/julius-speech/dic
tation-kit.git  # dictation-kitをGitHubからクローン
        cd $JULIUS_HOME/dictation-kit/src/
        # ダウンロードしたjuliusを解凍
        unzip julius-4.5.zip
        cd julius-4.5
        # ALSA対応で構成設定
        ./configure --with-mictype=alsa --build=aarch6
4
        make  # コンパイル
        # コンパイル済みの印としてmadeファイルを作成
        touch $MADE
    fi
```

```
    # インストール作業と初回起動
    cd $JULIUS_HOME/dictation-kit/src/julius-4.5
    make install
    cp julius/julius /opt/mike/bin/.
    echo "セットアップ終了"
    cd $JULIUS_HOME/dictation-kit
    sudo -u $USR julius -C main.jconf -C  am-gmm.jconf
else
    # 2回目以降
    echo "セットアップ済"
    cd  $JULIUS_HOME
    #USRモードで起動
    sudo -u $USR julius -C mike.jconf -module
fi
```

リスト5　Julius起動コマンド

```
$ cd /opt/mike/julius_server⏎
$ docker compose up --build⏎
```

リスト6　mike_data用のDockerfile

```
# ①ベース・イメージとして Python 3.11.2 を使用
FROM python:3.11.2
RUN apt update
# ②サウンド・ライブラリの開発用パッケージ
RUN apt -y install libasound2-dev=1.2.4-1.1
# 音声処理用ツール
RUN apt -y install ffmpeg=7:4.3.8-0+deb11u1
#RUN apt -y install netcat-openbsd=1.219-1
RUN apt -y install netcat-openbsd=1.217-3
RUN apt clean

# ③ユーザ情報を設定
ARG USERNAME=mike
ARG GROUPNAME=user
ARG UID=1000
ARG GID=1000

# 指定したUIDとGIDで新しいグループとユーザを作成し,
# 音声グループにも追加
RUN groupadd -g $GID $GROUPNAME && \
```

```
    useradd -m -s /bin/bash -u $UID -g $GID -G audio $
USERNAME

# ④作成したユーザを使用する
USER $USERNAME

# ⑤pipを24.3.1にアップグレード
RUN pip install -U --no-cache-dir  pip==24.3.1
# オーディオ再生用パッケージ
RUN pip install --no-cache-dir simpleaudio==1.0.4
# WAVファイルの操作パッケージ
RUN pip install --no-cache-dir wave==0.0.2
# HTTPリクエスト用のライブラリ
RUN pip install --no-cache-dir requests==2.32.3

# ⑥作業ディレクトリを設定
WORKDIR /opt/$USERNAME/

# コンテナ起動時にbashシェルを実行
CMD ["/bin/bash"]
```

リスト7　mike_data用のdocker-compose.yml

```
services:
  data: #①サービス名をdataに設定
    # 同じディレクトリ内のDockerfileからビルド
    build: .
    # イメージ名をmike_data-imgに指定
    image: mike_data-img
    # コンテナ名をmike_data-contに設定
    container_name: mike_data-cont
    # コンテナのホスト名をmike-dataに設定
    hostname: mike-data

    devices:
      # ②ホストのサウンド・デバイスをコンテナ内で
      # 使用できるよう設定
      - "/dev/snd:/dev/snd"
    volumes:
      # ③ホストの/opt/mikeディレクトリをコンテナの
      # /opt/mikeにマウントし共有
      -  /opt/mike:/opt/mike
    # コンテナ内の作業ディレクトリを/opt/mikeに設定
```

```
    working_dir: /opt/mike

    # ④プログラムを実行するためのコマンドを指定
    command: ["python", "/opt/mike/mike_data/opt_mike/
bin/mike_data.py" ]


networks:
  default:
    # ⑤mike-netという外部ネットワークを使用
    name: mike-net
    external: true  # 外部ネットワーク設定
```

リスト8　スクリプトmike_data.sh

```
#!/bin/bash
# MIKE 用のデータを生成し各プログラムに展開する

if [ $# ① -ne 1 ];then
    echo "Usage: $0 [VOICEVOX_ADDR] ";
    exit 1
fi
VOICEVOX_ADDR=$1
VOICEVOX_PORT=50021

# ②Mike_data.sh のパスを基にMIKE_DATA_HOMEを求める
MIKE_DATA_HOME=$(cd $(dirname $0);cd .. ;pwd)
OPT_MIKE=""
if [ "`echo $MIKE_DATA_HOME | grep "opt_mike"`" ]; the
n
 #echo DOCKER版
 OPT_MIKE="/opt_mike"
fi

# ③VOICEVOXサーバ接続確認
if nc -z -w 2 "$VOICEVOX_ADDR" "$VOICEVOX_PORT" ; then
  echo "データ作成を開始します"
else
  echo "エラー: VOICEVOX サーバーが起動していません。サー
バーを立ち上げて再試行してください"
  exit 1
fi
```

```
# ④MIKE_DATA__HOMEに移動して、データを展開する。
cd ${MIKE_DATA_HOME}
python bin/mike_data.py $VOICEVOX_ADDR
bin/copy_voice.sh
echo "作成したデータを各サーバに展開しました"
```

リスト9　Containerの構築と起動

```
$ cd /opt/mike/mike_data/⏎
$ docker compose run --rm data bash mike_data/opt_mike
/bin/mike_data.sh [VOICEVOXのアドレス]⏎
```

リスト10　mike_server用のDockerfile

```
# ①ベース・イメージとして Python 3.11.2 を使用
FROM python:3.11.2

# ②必要なパッケージをインストール
# - libasound2-dev: ALSAの開発ライブラリ,
#      音声デバイス対応
# - usbutils: USBデバイス管理ツール
RUN apt update && apt install -y \
    libasound2-dev=1.2.4-1.1
RUN apt clean # キャッシュを削除してイメージ・サイズを
削減

# ユーザ情報を設定するための変数
ARG USERNAME=mike
ARG GROUPNAME=user
ARG UID=1000
ARG GID=1000

# ③指定したUIDとGIDで新しいグループとユーザを作成し,
# 音声グループにも追加
RUN groupadd -g $GID $GROUPNAME && \
    useradd -m -s /bin/bash -u $UID -g $GID -G audio $
USERNAME

# ④作成したユーザを使用する
USER $USERNAME

# ⑤作業ディレクトリを設定
```

```
WORKDIR /opt/$USERNAME/

# ⑥必要なPythonパッケージをインストール
RUN pip install -U --no-cache-dir pip==24.3.1
# オーディオ再生用パッケージ
RUN pip install --no-cache-dir simpleaudio==1.0.4
# WAVファイルの操作パッケージ
RUN pip install --no-cache-dir wave==0.0.2
# HTTPリクエスト用のライブラリ
RUN pip install --no-cache-dir requests==2.32.3
```

リスト11　mike_server用のdocker-compose.yml

```
services:
  server: # ①サービスの基本設定
    build: .  # 同じディレクトリ内のDockerfileからビル
ド
    # イメージ名をmike_server-imgに指定
    image: mike_server-img
    # コンテナ名をmike_srv-contに設定
    container_name: mike_srv-cont
    # コンテナが停止した場合に自動で再起動する設定
    restart: always
    hostname: mike-server
    volumes:
      # ②ホストの./opt_mikeディレクトリをコンテナの
      # /opt/mikeにマウントし，共有
      - ./opt_mike:/opt/mike

    # ③コンテナ内の作業ディレクトリを/opt/mikeに設定
    working_dir: /opt/mike
    # ④プログラムを実行するためのコマンドを指定
    command: ["python", "bin/mike_server.py"]
    #command: bash -c "ls / && bash"
    # ディレクトリー覧表示後、シェルを起動する
    # 代替コマンド（コメントアウト）
    #tty: true

networks:   # ⑤ネットワーク設定
  default:
    name: mike-net
    external: true
```

リスト12　Containerの構築と起動

①Containerの構築と起動

```
$ cd /opt/mike/mike_server⏎
$ docker compose up –build⏎
```

②メッセージ

```
INFO Server started. Waiting for a connection...
```

リスト13　mike_client用のDockerfile

```
# ①ベース・イメージとしてPython 3.11.2を使用
FROM python:3.11.2

# ②パッケージ・リストを更新し，音声処理に必要な
# ライブラリをインストール
RUN apt update && apt install -y \
    libasound2-dev=1.2.4-1.1
RUN apt clean

# ユーザ情報を設定するための変数
ARG USERNAME=mike
ARG GROUPNAME=user
ARG UID=1000
ARG GID=1000

# ③指定したUIDとGIDで新しいグループとユーザを作成し，
# 音声グループにも追加
RUN groupadd -g $GID $GROUPNAME && \
    useradd -m -s /bin/bash -u $UID -g $GID -G audio $
USERNAME

# 作成したユーザを使用する
USER $USERNAME

# 作業ディレクトリを設定
WORKDIR /opt/$USERNAME/

# ④必要なPythonパッケージをインストール
RUN pip install -U --no-cache-dir pip==24.3.1
RUN pip install --no-cache-dir simpleaudio==1.0.4
RUN pip install --no-cache-dir wave==0.0.2
```

```
RUN pip install --no-cache-dir requests==2.32.3

# 実行コマンドを設定
CMD ["/bin/bash"]
```

リスト14　mike_client用のdocker-compose.yml

```
services:    # ①サービスの基本設定
  client:
    build: .
    image: mike_client-img  # イメージ名を指定
    container_name: mike_cli-cont  # コンテナ名を指定
    hostname: mike-client # Host名を指定
    restart: always  # コンテナが停止した場合に自動で再
起動
    # ②ボリュームのマウント
    volumes:
      # "/opt/mike" を共有
      - ./opt_mike:/opt/mike
    # ③デバイス設定
    devices:
      - "/dev/snd:/dev/snd"  # ホストのオーディオデバイ
スをコンテナ内で利用可能に設定
    working_dir: /opt/mike  # ④コンテナ内の作業ディレ
クトリを "/opt/mike" に設定

    command: ["python","bin/mike_client.py"]  # ⑤コン
テナ起動時に実行するコマンド

networks:    # ⑥ネットワーク設定
  default:
    name: mike-net #Docker内のネットワーク名
    external: true
```

リスト15　Containerの構築と起動

```
$ cd /opt/mike/mike_client⏎
$ docker compose up --build⏎
```