

第19回ICN研究会ワークショップ

Cefore Hands-on Practice A: Cefore/Docker を用いたマルチキャストストリーミング

速水祐作

情報通信研究機構(NICT)

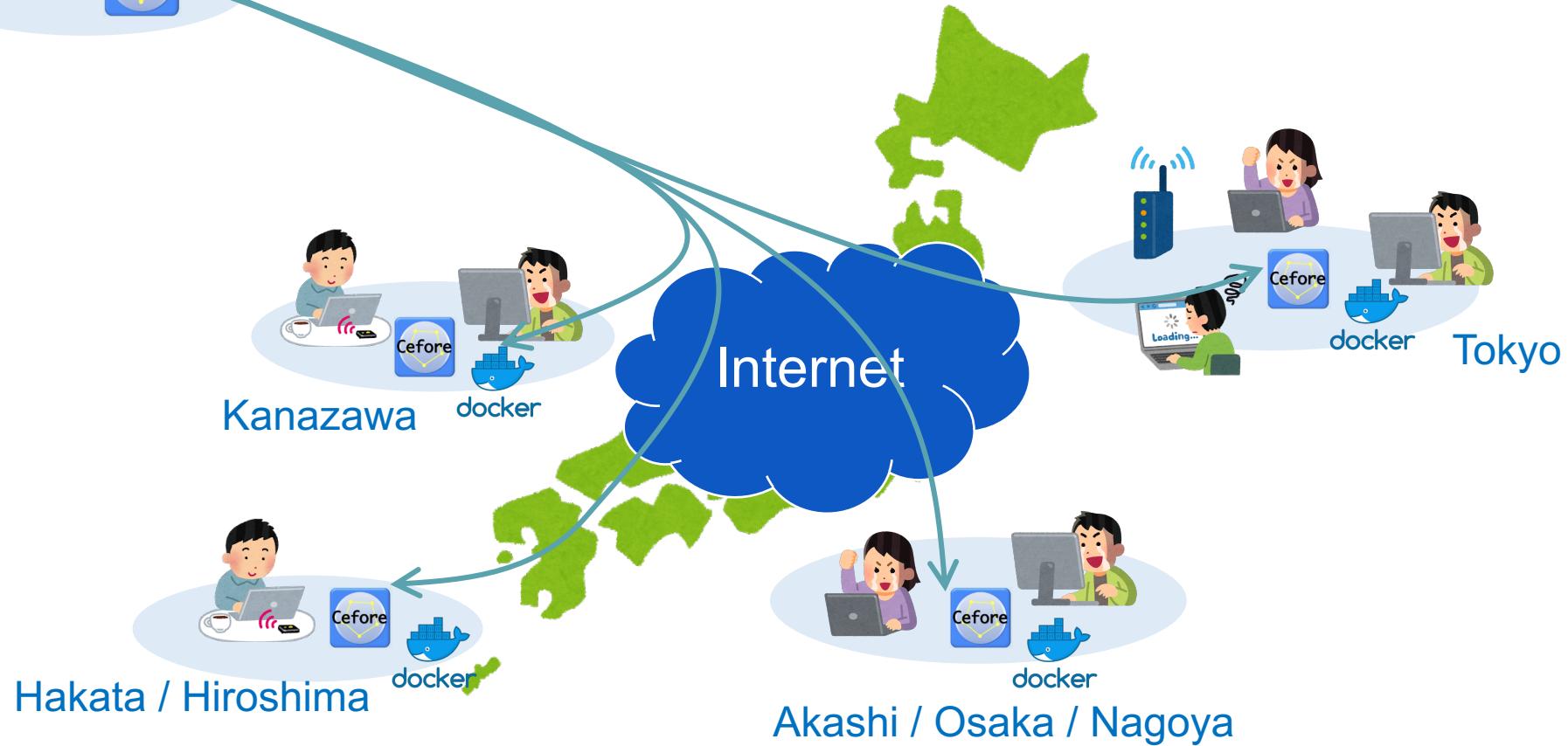
2021年8月26-27日

- 目的・全体像の説明
- Docker の概要
- Docker を用いた Cefore 導入方法
- Practice A
 - [A-1] グローバル通信シナリオ
 - [A-2] ローカル通信シナリオ
 - [A-3] 4K ビデオストリーミングシナリオ



Goal

Docker を通して Cefore に触れて ICN 通信の良さ
(マルチキャスト) を視覚的に体感すること



* <https://peach.blender.org/>

- Docker社開発の Container(コンテナ)型仮想環境を作成、配布、実行するためのプラットフォーム



- コンテナと仮想マシン(VM)の違い

- VirtualBoxなどの仮想マシンではホストOS上でハイパーバイザを利用しゲストOSを動かし、その上でミドルウェアなどを動かす
- Containerはホストマシンのカーネルを利用してプロセスやユーザなどを隔離することで、あたかも別マシンが動いているかのように実行できるため、軽量で高速に起動・停止等が可能



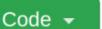
VM と コンテナ の 比較 ([1] より引用)

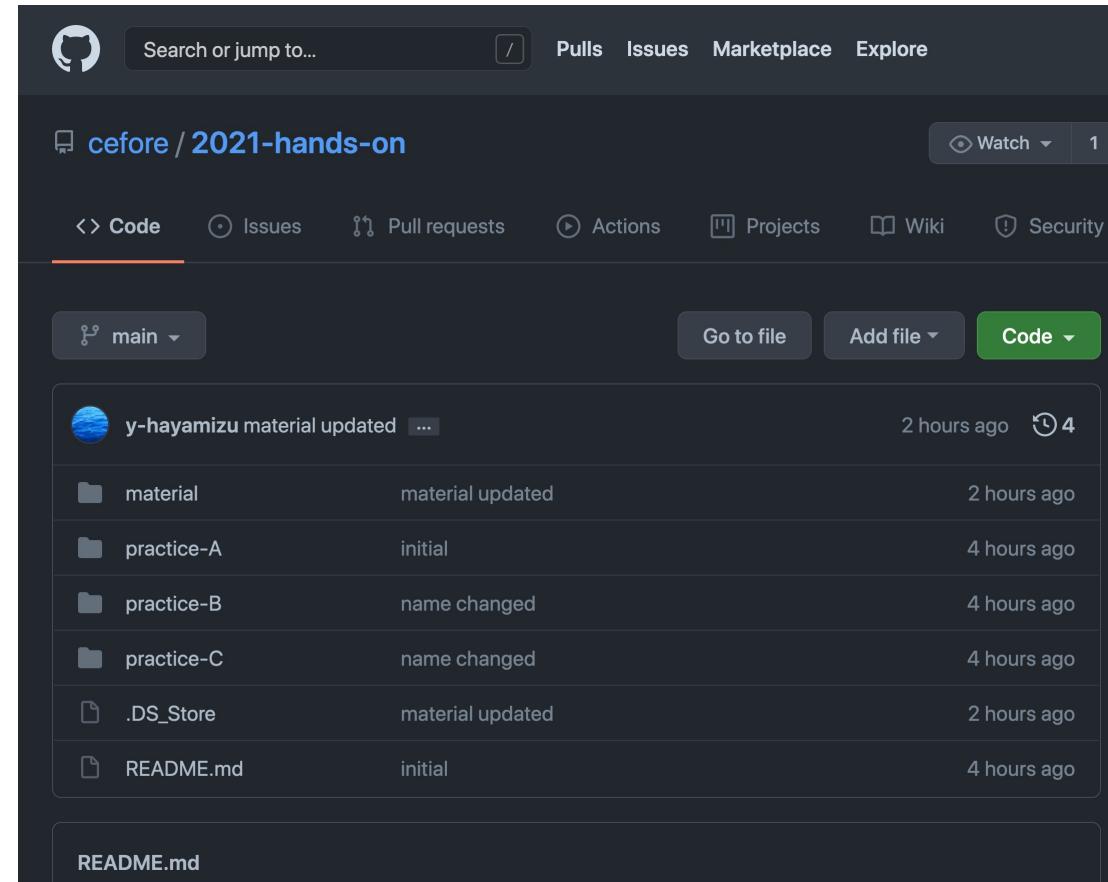
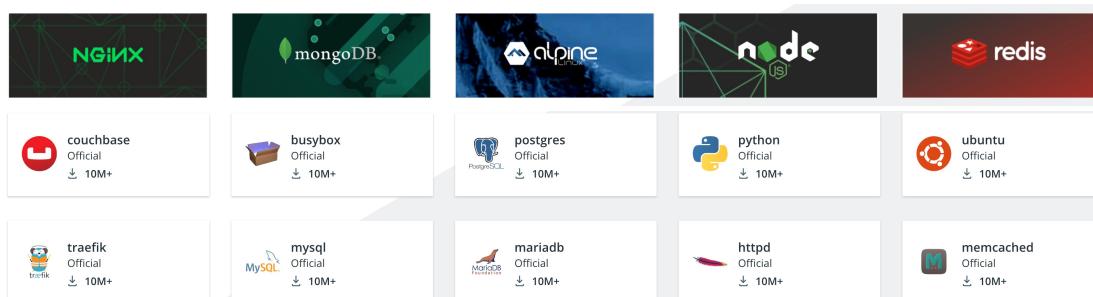
- Docker compose



- 複数のコンテナを一度に定義して実行するためのDockerアプリケーション用ツール
- YAML形式で記載

Cefore/Docker image のダウンロード

- GitHub web install
 - <https://github.com/cefore/2021-hands-on> にアクセス
 -  から zip をダウンロード
 - 任意のディレクトリに展開
- GitHub CUI install
 - sudo apt install git
 - git clone https://github.com/cefore/2021-hands-on.git
- Docker Hub
 - <https://hub.docker.com/>



A screenshot of a GitHub repository page for "cefore / 2021-hands-on". The repository has 1 star and 1 issue. The code tab is selected, showing the main branch. Recent commits are listed:

- y-hayamizu material updated ... 2 hours ago
- material material updated 2 hours ago
- practice-A initial 4 hours ago
- practice-B name changed 4 hours ago
- practice-C name changed 4 hours ago
- .DS_Store material updated 2 hours ago
- README.md initial 4 hours ago

The README.md file is expanded at the bottom of the screen.

- 2021-hand-on
 - material
 - 202108_ICN-ken_Cefore-hands-on_preparation.pdf
 - 202108_ICN-ken_ICN-description.pdf
 - Practice-A_Cefore-Docker-based_Multicast-Commn.pdf
 - Practice-B_cefore-and-cefpyco-tutorial.pdf
 - Practice-C_ICN-PUSH-Commn.pdf
 - practice-A
 - practice-B
 - practice-C

```
.  
|   └── base  
|       └── Dockerfile  
|   └── bin  
|       └── bbb.mp4  
|           └── consumer.bash  
|               └── publisher.bash  
|   └── build.bash  
|   └── cache  
|       └── Dockerfile  
|           └── entrypoint.bash  
|   └── csmgr  
|       └── Dockerfile  
|           └── entrypoint.bash  
|   └── docker-compose.yml  
|   └── docker-compose.yml_global  
|   └── docker-compose.yml_local  
|   └── min  
|       └── Dockerfile  
|           └── entrypoint.bash  
|       └── playback.bash  
|           └── start-A-1.bash  
|               └── start-A-2.bash  
|                   └── start-A-3.bash  
|           └── stop.bash  
|               └── teardown.bash
```

※ 詳細は後述

コンテナの作成(ビルド)

※ 時間を要するので先に ./build.bash を実行

- ビルド方法
 - docker build -f Dockerfile -t cefore/base

Dockerfile の内容

```
% cat base/Dockerfile
FROM ubuntu:18.04
LABEL maintainer="hayamizu <hayamizu@nict.go.jp>"
RUN mkdir -p /cefore
WORKDIR /cefore
RUN apt update
RUN apt install -y git build-essential libssl-dev
RUN apt install -y automake
RUN apt install -y iputils-ping net-tools tcpdump ifstat
RUN apt install -y emacs vim nano
RUN apt -y clean
RUN git clone https://github.com/cefore/cefore.git
WORKDIR /cefore/runner_test
RUN apt -y clean
```

docker build の実行画面

```
hayamizu@nu practice-A % ./build.bash
/Users/hayamizu/202108_ICN-ken_workshop/2021-hands-on/practice-A/base
[+] Building 31.4s (7/22)
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 662B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/ubuntu:18.04   1.7s
=> [ 1/19] FROM docker.io/library/ubuntu:18.04@sha256:7bd7a9ca99f868bf69c4b621          5.6s
=> => resolve docker.io/library/ubuntu:18.04@sha256:7bd7a9ca99f868bf69c4b6212f        0.0s
=> => sha256:39a8cfeef17302cb7ce93cefe12368560fe62ef9d51780885 1.46kB / 1.46kB    0.0s
=> => sha256:feac5306138255e28a9862d3f3d29025d0a4d0648855afe 26.71MB / 26.71MB   4.5s
=> => sha256:7bd7a9ca99f868bf69c4b6212f64f2af8e243f97ba13abb3e 1.41kB / 1.41kB    0.0s
=> => sha256:07782849f2cff04e9bc29449c27d0fb2076e61e8bdb4475ec5dbc 529B / 529B    0.0s
=> => extracting sha256:feac5306138255e28a9862d3f3d29025d0a4d0648855afe1acd613 1.0s
=> [ 2/19] RUN mkdir -p /cefore                           0.2s
=> [ 3/19] WORKDIR /cefore                             0.0s
=> [ 4/19] RUN apt update                                20.7s
=> [ 5/19] RUN apt install -y git build-essential libssl-dev   3.1s
=> => # Need to get 67.3 MB of archives.
=> => # After this operation, 286 MB of additional disk space will be used.
=> => # Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 liblocale-gettext-p
=> => # erl amd64 1.07-3build2 [16.6 kB]
=> => # Get:2 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 multiarch-s
=> => # upport amd64 2.27-3ubuntu1.4 [6944 B]
```

- ビルドしたイメージの確認

```
% docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
cefore/csmgr    latest   4c505f712521  12 minutes ago  885MB
cefore/cache    latest   59ef9d12b859  12 minutes ago  881MB
cefore/min      latest   935747e7cc84  12 minutes ago  881MB
cefore/base     latest   3694f2af7dd7  13 minutes ago  873MB
```

- コンテナの起動

```
% docker run --name consumer -it IMAGE_ID /bin/bash
root@7db7391ba03c:/cefore/runner_test#
```

- コンテナへのログイン = 別ターミナルでbashを実行(exec)

```
% docker exec -it consumer /bin/bash
root@7db7391ba03c:/cefore/runner_test#
```



docker run は、指定したコマンド (ここでは /bin/bash) を実行するためにdockerを起動するためのコマンドなので、exit して bash shell を抜けると 当該 container にはアクセスできなくなる(STATUS = Exited (0)になる)。その場合、docker start 7db7391ba03c (CONTAINER ID) する必要がある。別ターミナルからログインするのが望ましい。

- コンテナの停止

```
root@5d731a71974f:/cefore/cefore# exit
```

or

```
% docker stop CONTAINER_ID
```

- コンテナの削除

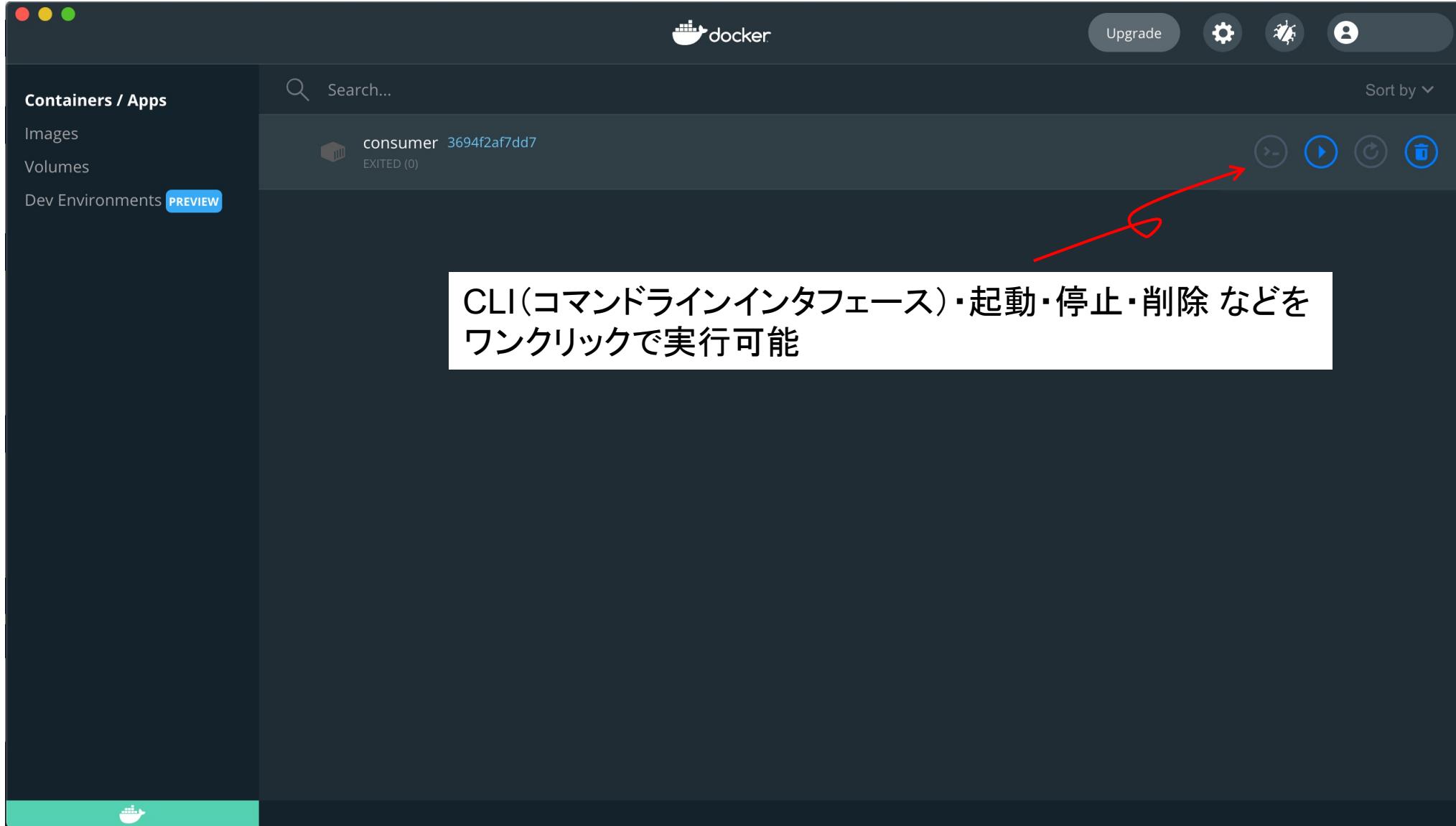
```
% docker rm -f CONTAINER_ID
```

- コンテナイメージの削除

```
% docker rmi IMAGE_ID
```

- キャッシュデータも含めた全削除

```
% docker builder prune  
WARNING! This will remove all dangling build cache. Are you sure you  
want to continue? [y/N] y
```



- docker-compose によりサービス(複数のコンテナで構築するシステム)の定義を簡略化可能

docker-compose.yml の中身

```
% cat docker-compose.yml
version: "3.3"
services:
  consumer:
    image: cefore/min
    container_name: "consumer"
    working_dir: "/cefore"
    volumes:
      - ${node_bindir:-$(pwd)}/bin:/cefore/bin
      - /tmp/video:/tmp/video
    networks:
      downward:
        ipv4_address: 10.0.1.101

networks:
  downward:
    name: downward
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 10.0.1.0/24
```

services:	構築するサービスの名前
image:	指定するDocker イメージ
volumes:	共有ディレクトリ hostOS:docker
networks:	構築するネットワークを指定
networks:	ネットワークの定義
name:	ネットワークの名前
driver:	ネットワークの種類(brdge, host, none)
ipam:	IPアドレス設定



YAML形式なので、Tabではなく、Spaceでインデントする必要がある

- docker-compose の実行

```
% docker-compose build  
% docker-compose up -d  
  
[+] Running 3/3  
  :: Network  
downward      Created          0.0s  
    :: Container  
publisher     Started          0.5s  
    :: Container  
consumer      Started          0.5s  
CONTAINER ID   IMAGE           CREATED          STATUS  
9c8f15594e11  9a5cd2ab7711  Less than a second ago  Up  Less than a second  
69b23c4ac1ab  31d67c47cf08  Less than a second ago  Up  Less than a second
```

docker-compose build: ビルド実行

docker-compose up -d: コンテナ作成 & 起動
-d はバックグラウンド実行

- 起動したコンテナの確認

```
% docker ps -a  
CONTAINER ID   IMAGE           COMMAND          CREATED          STATUS          PORTS          NAMES  
9c8f15594e11  9a5cd2ab7711  "/bin/sh -c /cefore/..."  11 seconds ago  Up  11 seconds  publisher  
69b23c4ac1ab  31d67c47cf08  "/bin/sh -c /cefore/..."  11 seconds ago  Up  11 seconds  consumer
```

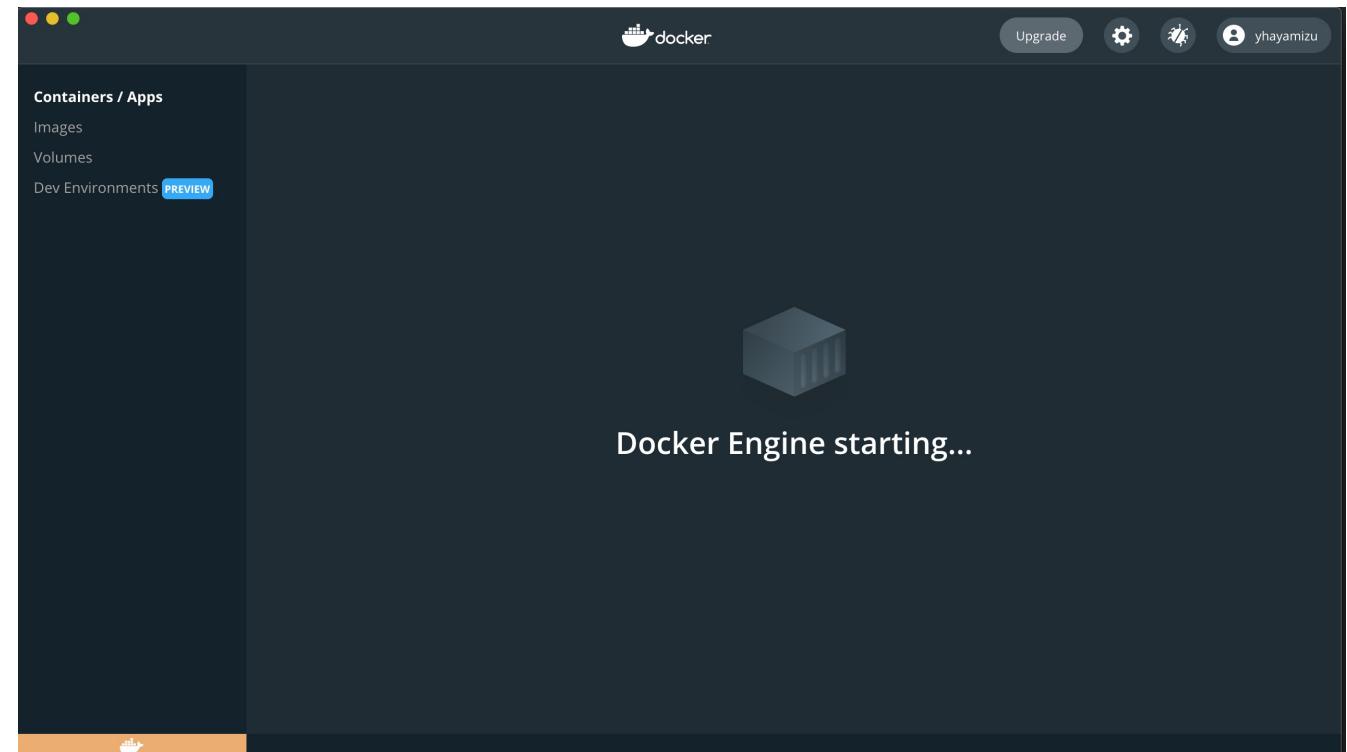
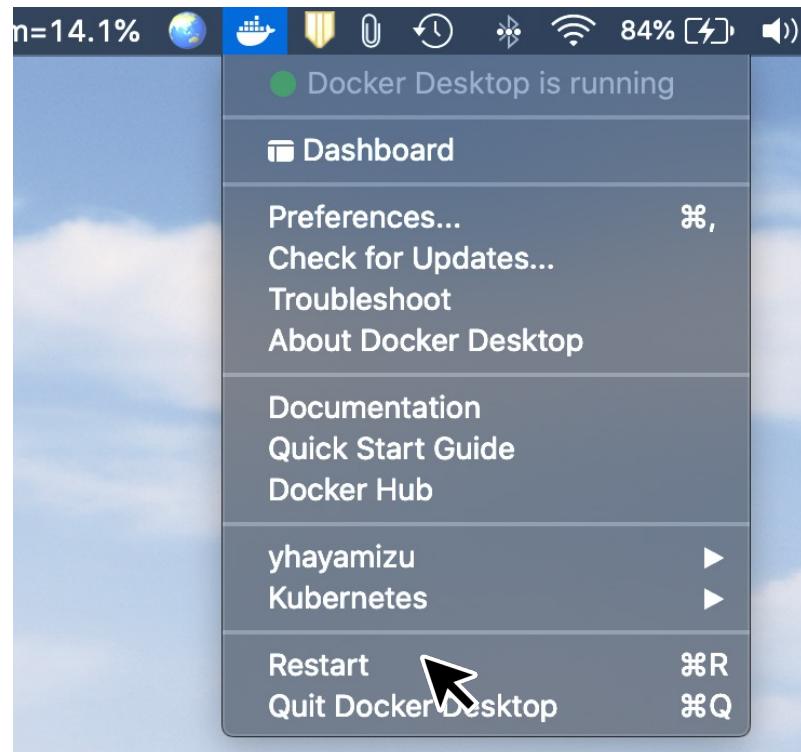
- コンテナの停止

```
% docker-compose down
```

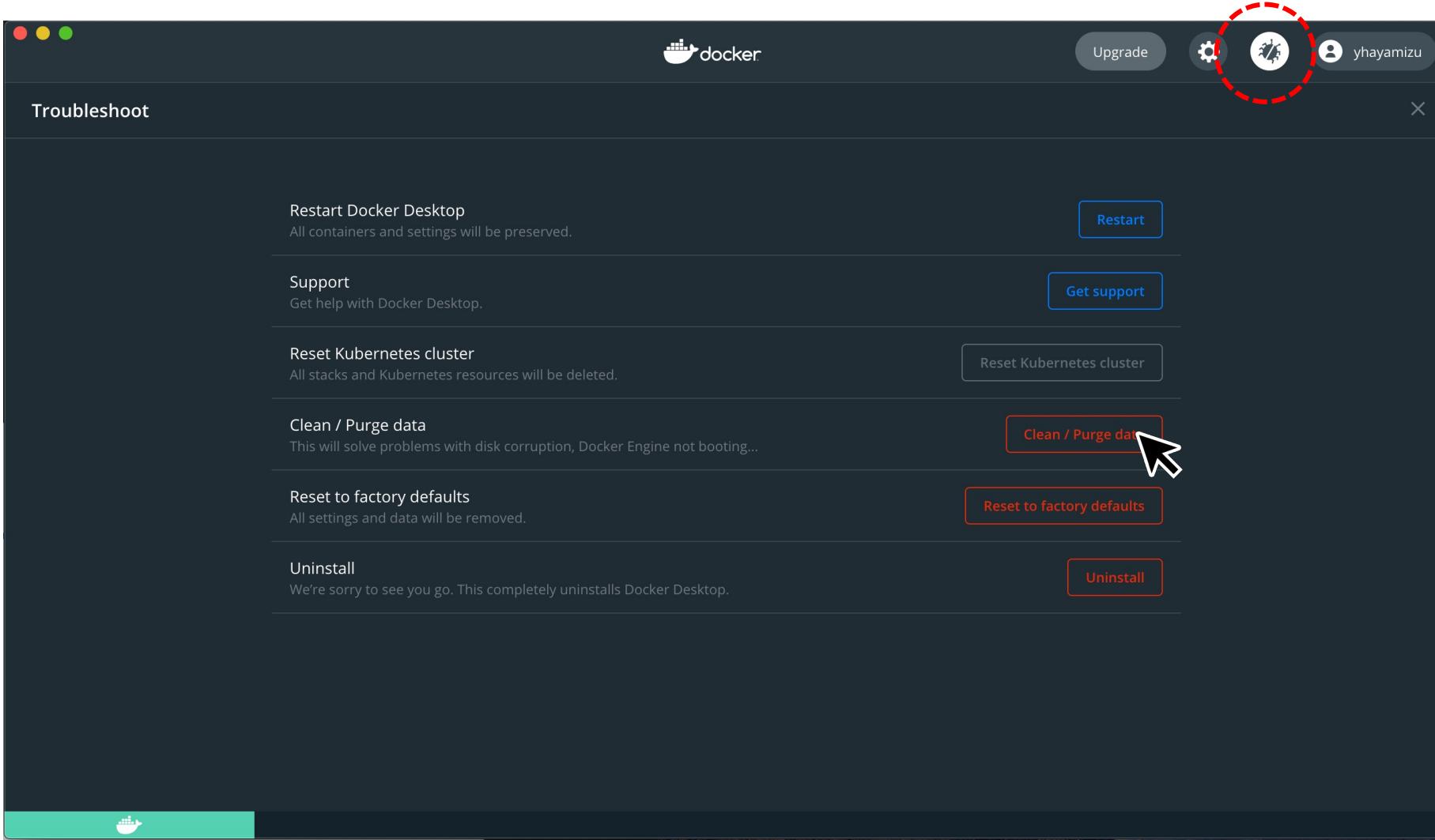
docker-compose down: コンテナの停止

- ./teardown.bash 時に Docker engine からの応答がない場合

```
% ./teardown.bash
[+] Running 0/2
  :: Container consumer  Error while Stopping
  :: Container publisher  Error while Stopping
Error response from daemon: Bad response from Docker engine
10.1s
10.1s
```



- Docker Desktop を再起動してもダメな場合、一度キャッシュを削除して、再度 build.bash を実行



ソケットバッファサイズの調整

- Ubuntu

```
$ sudo sysctl -w net.core.rmem_default=10000000  
$ sudo sysctl -w net.core.wmem_default=10000000  
$ sudo sysctl -w net.core.rmem_max=10000000  
$ sudo sysctl -w net.core.wmem_max=10000000
```

- mac OS

```
$ sudo sysctl -w net.local.stream.sendspace=10000000  
$ sudo sysctl -w net.local.stream.recvspace=10000000
```

⚠️ PC再起動時にパラメータが初期化されるので、再実行しやすいようスクリプト化するのを推奨

```
.  
|   └── base  
|       └── Dockerfile  
|  
|   └── bin  
|       ├── bbb.mp4  
|       └── consumer.bash  
|           └── publisher.bash  
|  
|   └── build.bash  
|  
|   └── cache  
|       └── Dockerfile  
|           └── entrypoint.bash  
|  
|   └── csmgr  
|       └── Dockerfile  
|           └── entrypoint.bash  
|  
|   └── docker-compose.yml  
|  
|   └── docker-compose.yml_global  
|  
|   └── docker-compose.yml_local  
|  
|   └── min  
|       └── Dockerfile  
|           └── entrypoint.bash  
|  
|   └── playback.bash  
|       └── start-A-1.bash  
|           └── start-A-2.bash  
|               └── start-A-3.bash  
|       └── stop.bash  
|  
└── teardown.bash
```

base:	全コンテナのベースとなる docker image
cache:	localcache mode の cefnetd 起動用 docker image
csmgr:	csmgrd と cefnetd 起動用 docker image(本Practiceでは不使用)
min:	最小構成で、cefnetd のみ(キャッシュ機能無し)を起動する docker image
bin:	host OS - docker container 間のデータ共有のための共有ディレクトリ
build.bash:	コンテナビルド用スクリプト
start-A-* .bash:	コンテナ起動 & シナリオ開始用スクリプト
teardown.bash:	コンテナ停止 & シナリオ終了用スクリプト
playback.bash:	ffplay でダウンロードした動画を再生するスクリプト
consumer.bash:	consumer コンテナ起動時に実行されるスクリプト*
publisher.bash:	publisherコンテナ起動時に実行されるスクリプト

緑色: docker image

黄色: 共有ディレクトリ

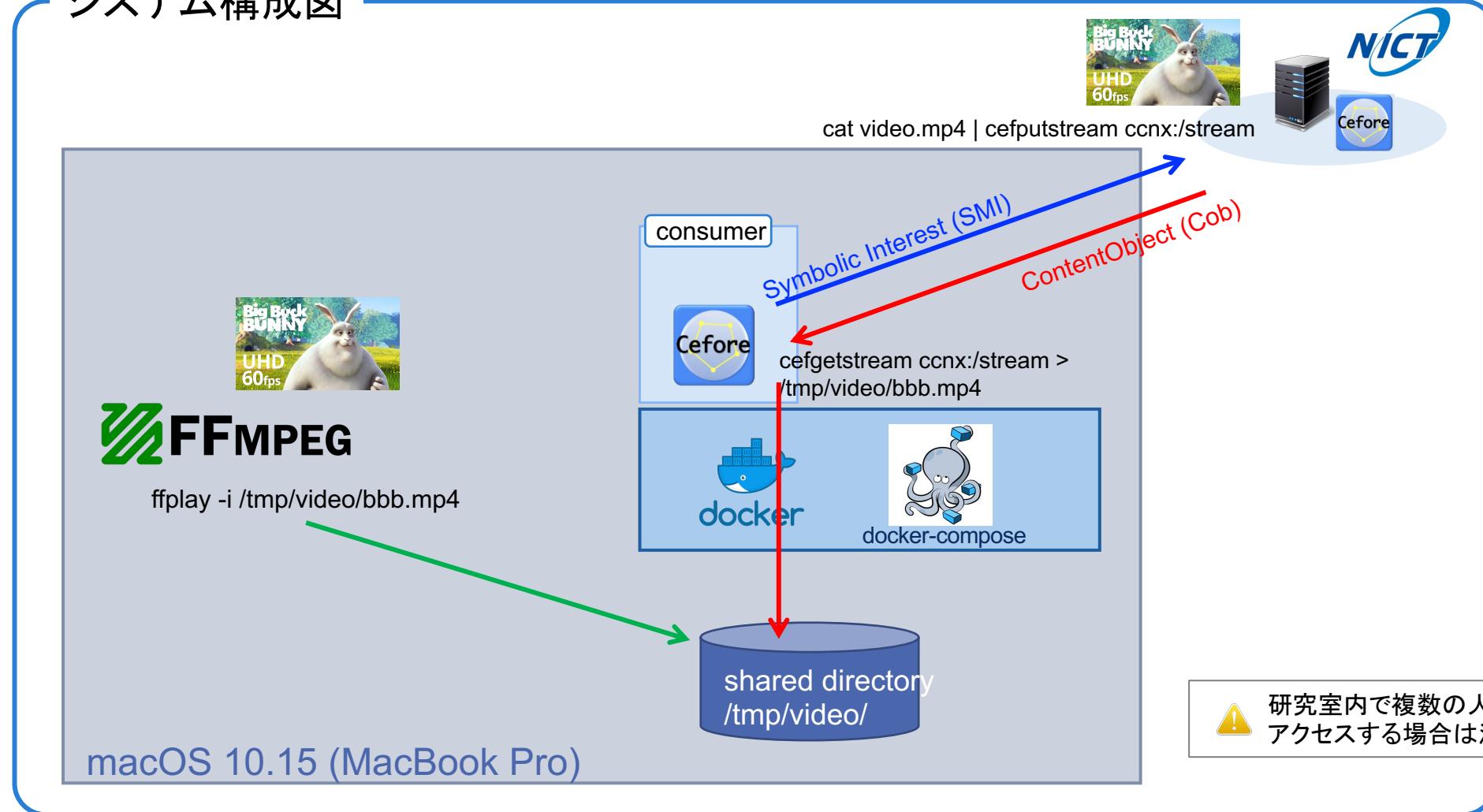
赤色: practice で触るファイル

⚠ * 複数のマシンがNAT以下で同一グローバルIPアドレスから通信する場合、
consumer.bash で設定されているFIB接続をTCP->UDPに変更してください。

※./build.bash の状態を確認

- NICTサーバと通信して動画をストリーミング再生してみよう！

システム構成図





1. build.bash

- キャッシュからビルドされることを確認

```
% ./build.bash
```

2. playback.bash

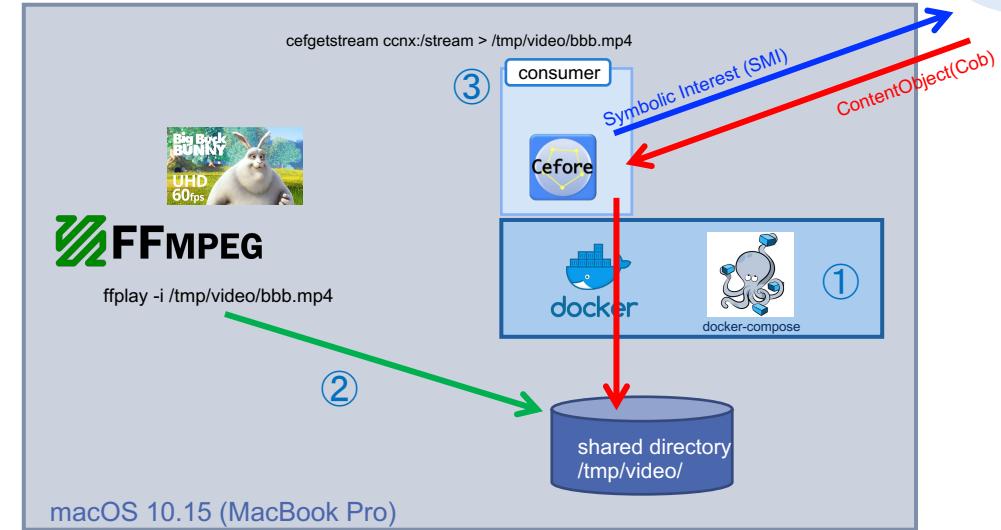
- 後に cefgetstream でダウンロードされるファイルの再生を ffplay で待ち受け

```
% ./playback.bash
Trying to play the video .....
```

3. start-A-1.bash

- consumer コンテナ起動

```
% ./start-A-1.bash
[+] Running 1/1
  :: Container consumer Started          0.4s
CONTAINER ID        IMAGE               CREATED             STATUS
a9bb2519dfd3      106532587b610   1 second ago       Up Less than a second
```





4. consumer docker にログインして 通信開始

```
% docker exec -it consumer bash
% cefgetstream ccnx:/stream -z 2 > /tmp/log/bbb.mp4
```

5. cefstatus で Cob 受信を確認

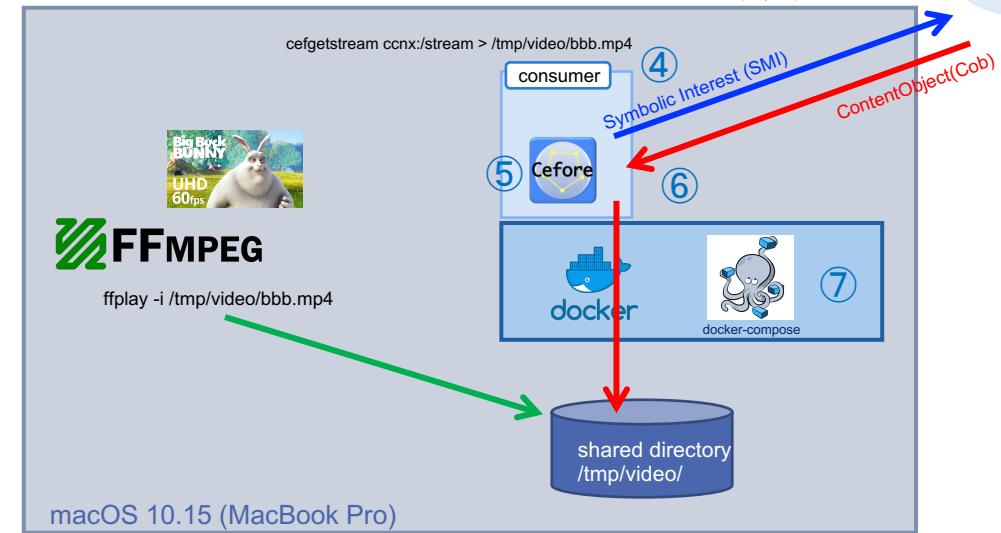
```
/cefore# watch cefstatus
```

6. 受信スループットを計測

```
/cefore# tail -f /tmp/log/throughput.log
Time           eth0
HH:MM:SS   KB/s in  KB/s out
08:14:44    3916.39   54.87
08:14:45    3977.96   53.96
08:14:46    3941.91   60.92
```

7. teardown.bash

```
% ./teardown.bash
```

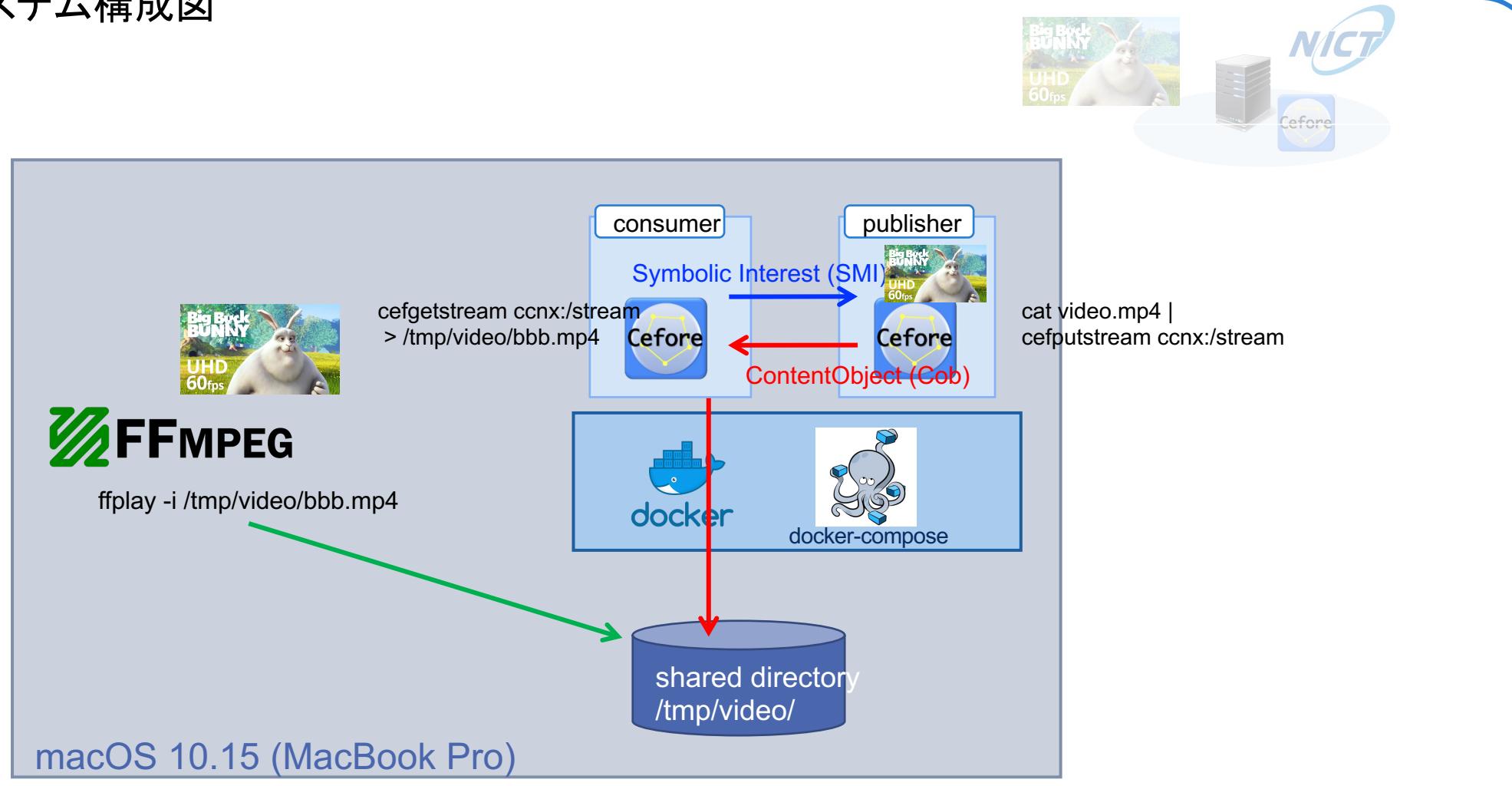


⚠️ `throughput.log` は `teardown.bash` 実行時に Host OS のホームディレクトリに自動バックアップ

Practice A-2: ローカル通信

- Docker 内に publisher を用意して、ローカルで通信してみよう！

システム構成図



Practice A-2(準備)

1. docker-compose の設定変更 (A-1 -> A-2)

```
% cp docker-compose.yml_local docker-compose.yml
```

2. シナリオ開始

```
% ./start-A-2.bash
```

3. consumer docker にログイン

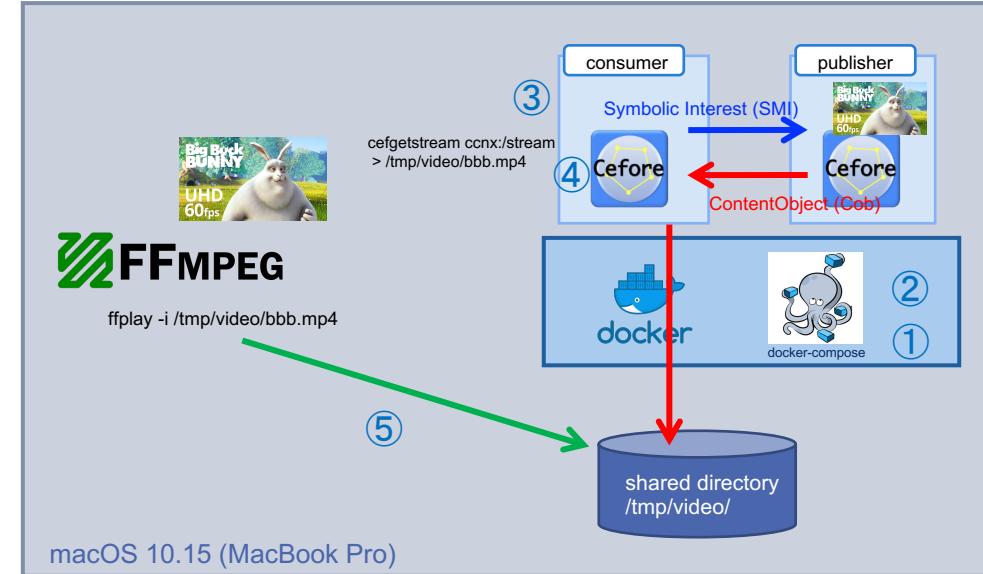
```
% docker exec -it consumer bash
```

4. FIB 切替(NICT server -> local publisher)

```
/cefore# cefroute del ccnx:/stream tcp NICT_SERVER_IP
/cefore# cefroute add ccnx:/stream tcp ??
/cefore# cefstatus
```

5. playback.bash

```
% ./playback.bash
Trying to play the video .....
```



Practice A-2(準備)

1. docker-compose の設定変更 (A-1 -> A-2)

```
% cp docker-compose.yml_local docker-compose.yml
```

2. シナリオ開始

```
% ./start-A-2.bash
```

3. consumer docker にログイン

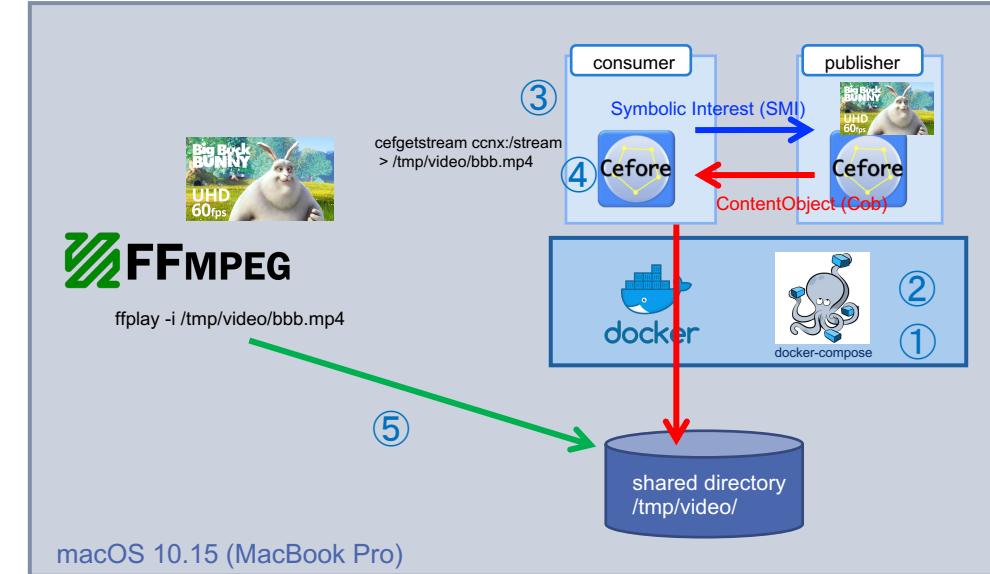
```
% docker exec -it consumer bash
```

4. FIB 切替(NICT server -> local publisher)

```
/cefore# cefroute del ccnx:/stream tcp NICT_SERVER_IP
/cefore# cefroute add ccnx:/stream tcp 10.0.1.10
/cefore# cefstatus
```

5. playback.bash

```
% ./playback.bash
Trying to play the video .....
```



docker-compose.yml で設定

Practice A-2(実行)

6. publisher にログインし、ストリーミングデータを送信

```
% docker exec -it publisher bash
% cat /cefore/bin/bbb.mp4 | cefputstream ccnx:/stream -r 3 -b 1400
```

7. consumer で通信開始

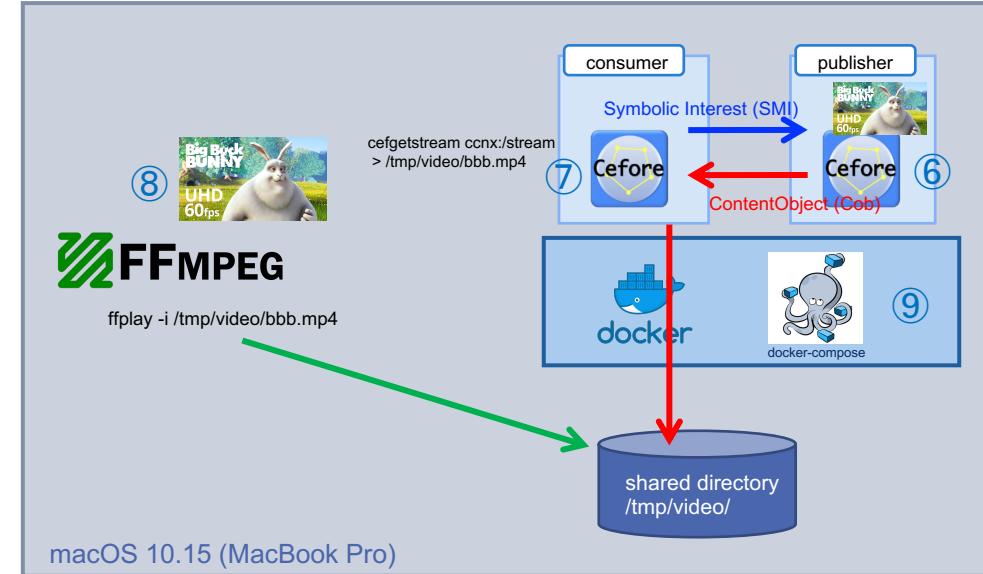
```
% cefgetstream ccnx:/stream -z 2 > /tmp/log/bbb.mp4
```

⚠️ 動画の再生時間は短い(約30秒)ので、cefputstream -> cefgetstream は素早く実行できるよう terminal を分けて実行

8. 動画の再生を視認

9. teardown.bash

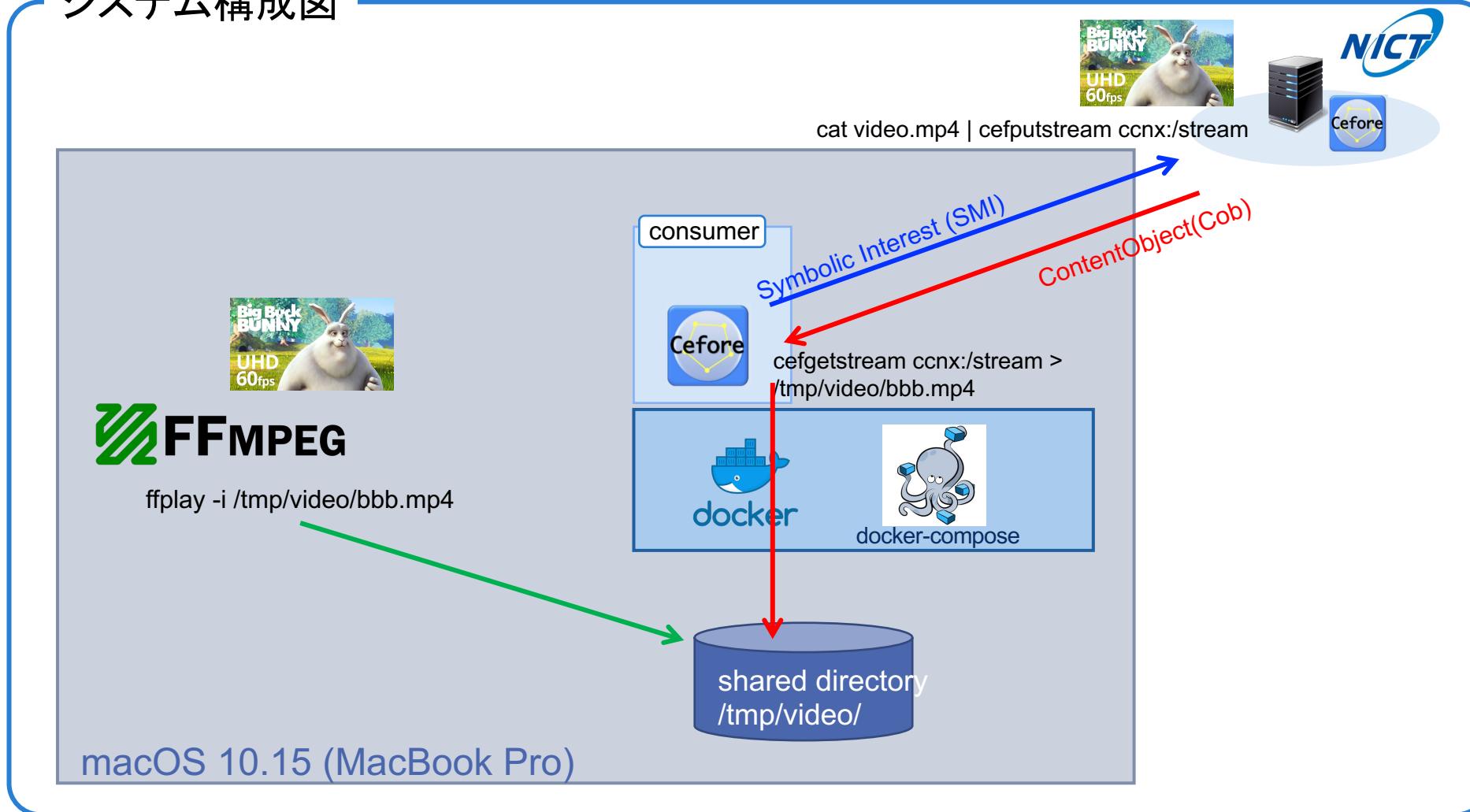
```
% ./teardown.bash
[+] Running 3/2
  ⚡ Container consumer    Removed          10.2s
  ⚡ Container publisher   Removed          10.2s
  ⚡ Network downward     Removed          0.1s
```



Practice A-3: 4Kマルチキャストストリーミング

- 4Kの高画質動画を見てみよう！

システム構成図





1. playback.bash

```
% ./playback.bash
Trying to play the video .....
```

2. start-A-3.bash

```
% ./start-A-1.bash
[+] Running 1/1
:: Container consumer Started          0.4s
CONTAINER ID   IMAGE      CREATED     STATUS
a9bb2519dfd3  106532587b610  1 second ago  Up Less than a second
```

3. consumer にログインして 通信開始

```
% docker exec -it consumer bash
% cefgetstream ccnx:/stream -z 2 > /tmp/log/bbb.mp4
```

4. 動画の再生品質を視認

5. teardown.bash

⚠ ホームディレクトリに保存された throughput.log をメールでご送付していただきたいです。ご協力のほどよろしくお願ひ申し上げます。

hayamizu@nict.go.jp

