INSTALL（all-in-one）日本語

事前準備

ハードウェア

all-in-oneを実行するハードウェアの最小構成の想定は以下の通り。カッコ内のハードウェア名はテスト環境で使用したもの。記載と異なるハードウェアを利用する場合は、手順の読み替えや設定値の変更が必要となる。

- 8 CPU Cores
- 32GB RAM
- 100GB hard disk space
- 1x Ethernet NIC
- 2x SR-IOV PF(Dual Port Mellanox ConnectX-5)
- 1x NVIDIA GPU(A100)

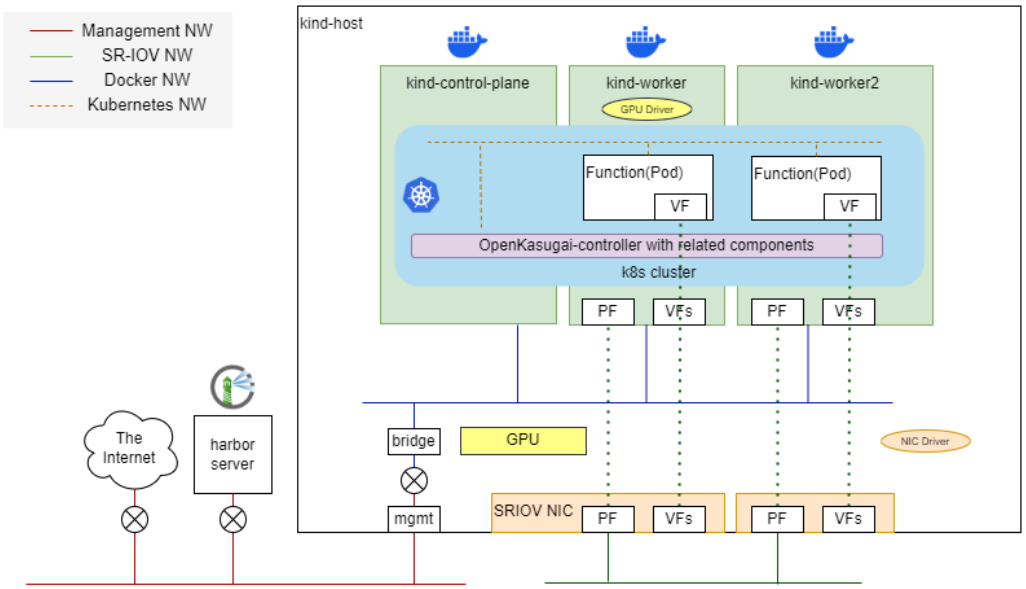
ソフトウェア

以下をインストールした環境を構築する。kind、golang、kubectlについてはPATHを設定しておく。

- Ubuntu
  - `22.04.1`
- Kernel version
  - `5.15.0-122-generic` or higher
- kind
  - `0.23.0`
- Docker
  - `26.1.3, build b72abbb`
- golang
  - `1.23.0`
- Mellanox OFED Driver
  - `MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64.tgz`
- kubectl
  - `1.31.1-1.1`

NW設計

事前に各セグメントのIPアドレスを設計しておく。`harbor server` はローカルなコンテナレジストリの例として記載する。



NW	用途	IP(CIDR)	関連設定ファイル	備考
Management	kind-hostのIP	192.168.10.2	-	-
	harbor serverのIP	192.168.10.100	./Makefile	• Functionイメージの取得先として利用 • kind-hostから到達できること
	Gateway	192.168.10.1	-	-
SR-IOV	CIDR	192.168.20.0/24	./Makefile	• nvidia-k8s-ipamの設定値として使用
	kind-workerのPF	192.168.20.2	./Makefile	-
	kind-worker2のPF	192.168.20.3	./Makefile	-
	kind-worker1と2のVFs	192.168.20.10-192.168.20.50	./Makefile	• nvidia-k8s-ipamがPodに付与する • 利用可能IPアドレスを分割するために、1Nodeあたりの IPアドレス数を30とする
	Gateway	192.168.20.1	./Makefile	• nvidia-k8s-ipamの設定値として使用 • 上記CIDRに含まれる必要がある
Docker	Docker NodeのIP	172.18.0.0/16	-	• Dockerによる自動設定 • 基本的に変更する必要はない
	Podの外部IP	172.18.9.0-172.18.9.10	./manifest/metallb/ipaddrespool.yaml	• MetalLBがServiceに付与するIPアドレス帯

				• Docker Nodeのセグメントに合わせて変更
Kubernetes	PodSubnet	10.100.0.0/16	./kind/kind-values/values.yaml	• cni通信に使用するCIDRを指定 • 環境で未使用のものを設定する

ホストの追加設定

Swapの無効化

必要に応じて、swap領域を削除する。

```
$ sudo swapoff -a
$ sudo swapon --show
```

Hugepageの設定

1. /etc/default/grub の編集

```
$ sudo vi /etc/default/grub
...（中略）...
GRUB_CMDLINE_LINUX_DEFAULT="default_hugepagesz=1G hugepagesz=1G hugepages=32"
```

2. Grub設定の反映と再起動

```
$ sudo update-grub
$ sudo reboot
```

3. Hugepageの設定確認

```
$ cat /proc/meminfo
:
HugePages_Total:      32
HugePages_Free:       32
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:    1048576 kB
```

SR-IOV機能の有効化

Mellanox ConnectX-5の例を示す

1. 以下よりドライバを入手

- https://developer.nvidia.com/networking/mlnx-ofed-eula?mtag=linux_sw_drivers&mrequest=downloads&mtype=ofed&mver=MLNX_OFED-24.04-0.7.0.0&mname=MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64.tgz
- 使用したファイル

```
$ md5sum MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64.tgz
8a15626083c14dcd64e33f200ebd142a  MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64.tgz
```

2. ドライバのインストール

- tarを展開

```
$ sudo su
# tar xzvf MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64.tgz
./MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64/
./MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64/DEBS/
...（中略）...
./MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64/.arch
./MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64/distro
```

- インストーラを実行

```
cd MLNX_OFED_LINUX-24.04-0.7.0.0-ubuntu22.04-x86_64

# ./mlnxofedinstall
Logs dir: /tmp/MLNX_OFED_LINUX.22943.logs
General log file: /tmp/MLNX_OFED_LINUX.22943.logs/general.log

Below is the list of MLNX_OFED_LINUX packages that you have chosen
(some may have been added by the installer due to package dependencies):

ofed-scripts
...(中略)...
ibarr

This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, RDMA or Distribution IB packages will be removed.
Those packages are removed due to conflicts with MLNX_OFED_LINUX, do not reinstall them.

Do you want to continue?[y/N]:y

Checking SW Requirements...
One or more required packages for installing MLNX_OFED_LINUX are missing.
...(中略)...
Installation passed successfully
To load the new driver, run:
/etc/init.d/openibd restart
```

3. ドライバのロード

```
# /etc/init.d/openibd restart
Unloading HCA driver:                                      [  OK  ]
Loading HCA driver and Access Layer:                       [  OK  ]
```

4. sr-iovの有効化

- 対象PFの調査

```
# ip link show ens9f0np0
8: ens9f0np0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    altname enp23s0f0np0
# ip link show ens9f1np1
9: ens9f1np1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    altname enp23s0f1np1

lspci -Dnn | grep Mellanox
0000:17:00.0 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex] [15b3:1019]
```

```
0000:17:00.1 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex] [15b3:1019]
```

- sr-iov有効化コマンドの実行

```
# mstconfig -d 0000:17:00.0 set SRIOV_EN=1 NUM_OF_VFS=8
# mstconfig -d 0000:17:00.1 set SRIOV_EN=1 NUM_OF_VFS=8
```

- 再起動

```
# reboot
```

- VFの作成

```
# echo 8 > /sys/class/net/ens9f0np0/device/sriov_numvfs
# echo 8 > /sys/class/net/ens9f1np1/device/sriov_numvfs
```

- VFの確認

```
# ip addr
...（中略）...
5: ens9f0np0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    altname enp23s0f0np0
6: ens9f1np1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    altname enp23s0f1np1
7: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
8: ens9f0v0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:xx
    altname enp23s0f0v0
9: ens9f0v1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v1
10: ens9f0v2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v2
11: ens9f0v3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v3
12: ens9f0v4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v4
13: ens9f0v5: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v5
14: ens9f0v6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v6
15: ens9f0v7: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f0v7
16: ens9f1v0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v0
17: ens9f1v1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v1
```

```
18: ens9f1v2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v2
19: ens9f1v3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v3
20: ens9f1v4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v4
21: ens9f1v5: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v5
22: ens9f1v6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v6
23: ens9f1v7: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff permaddr xx:xx:xx:xx:xx:
    altname enp23s0f1v7

# lspci -Dnn | grep Mellanox  | grep Virtual
0000:17:00.2 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:00.3 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:00.4 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:00.5 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:00.6 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:00.7 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.0 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.1 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.2 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.3 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.4 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.5 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.6 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:01.7 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:02.0 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
0000:17:02.1 Ethernet controller [0200]: Mellanox Technologies MT28800 Family [ConnectX-5 Ex Virtual Function] [15b3:101a]
```

必要ファイルの配置

以下のファイルをリポジトリ内に配置する

配置先	ファイル名	備考
./src/	controller.tar.gz	• 以下のリポジトリをクローン https://github.com/openkasugai/controller • リポジトリをアーカイブする • ディレクトリ名を `controller` に変更すること • クローン時、 `recursive` オプションを設定すること
./driver/nvidia/	NVIDIA-Linux-x86_64-550.90.12.run	• 以下より入手する https://www.nvidia.com/Download/index.aspx

./work/	*.mp4	• 映像推論シナリオの実行に使用する映像ファイルなどを配置する ◦ 映像ファイル(`*.mp4`)は `./work/DATA/video/` 以下に配置する必要がある。 • 各Dockerコンテナの `/root/work` へbindされる

Makefile作成手順

構築環境に合わせ、Makefile中のパラメータの修正を実施する。

パラメータ説明

パラメータ名	説明	デフォルト値
DCI_REGISTRY_ADDR	OpenKasugaiコントローラのレジストリURL	`ghcr.io/openkasugai/controller`
DCI_REGISTRY_ADDR_ALT	harborサーバ向けのレジストリURL	`192.168.10.100/images`
DCI_REGISTRY_CERT_URL	harborサーバのレジストリ証明書ファイルのダウンロードURL	`http://192.168.10.100/harbor/ca.crt`
DCI_K8S_SOFTWARE_TARFILE	OpenKasugaiコントローラのコードをtar圧縮したファイル名	`controller.tar.gz`
NVIDIA_DRIVER_FILE	NVIDIAドライバのインストールスクリプトファイル名	`NVIDIA-Linux-x86_64-550.90.12.run`
K8S_VERSION	kindで構築するk8sクラスタのバージョン	`v1.31.0`
GO_VERSION	kindノードにインストールするgoバージョン	`1.23.0`
SRIOV_CNI_VERSION	使用するSRIOVのバージョン	`v2.8.1`
CRIO_OS	kindノードに使用するOSのバージョン	`xUbuntu_22.04`
CRIO_REPO_VERSION	k8sクラスタにインストールするcri-oのバージョン（レポジトリ）	`v1.31`
CRIO_PKG_VERSION	k8sクラスタにインストールするcri-oのバージョン（パッケージ）	`1.31.0-1.1`
METALLB_VERSION	使用するMetalLBのバージョン	`v0.14.8`
NVIDIA_K8S_IPAM_VERSION	使用するnvidia-k8s-ipamのバージョン	`v0.3.5`
IMG_TAG	DockerHubよりイメージを取得する際のイメージのタグ名	`22.04.5`
SRIOV_IF_REGEX_1	kind-workerに付与する、1本目のPFとVFの両方のIF名を取得するための正規表現	`ens9f0.*`

SRIOV_IF_REGEX_2	kind-workerに付与する、2本目のPFとVFの両方のIF名を取得するための正規表現	`ens9f1.*`
SRIOV_PF_IF_1	kind-workerに付与する、1本目のPFのIF名	`ens9f0np0`
SRIOV_PF_IF_2	kind-workerに付与する、2本目のPFのIF名	`ens9f1np1`
SRIOV_PF_IPADDR_1	1本目のPFに付与するIPアドレス(CIDER)	`192.168.20.2/24`
SRIOV_PF_IPADDR_2	2本目のPFに付与するIPアドレス(CIDER)	`192.168.20.3/24`
SRIOV_NUM_VFS	作成するVFの数	`8`
NV_IPAM_SUBNET	VFに自動付与するIPアドレスのサブネット	`192.168.20.0/24`
NV_IPAM_PER_NODE_BLOCK_SIZE	VFに自動付与するIPアドレスを分割するための、1Nodeあたりの IPアドレス数	`30`
NV_IPAM_GATEWAY	VFに自動付与するIPアドレスのゲートウェイ	`192.168.20.1`
NV_IPAM_EXCLUDE_START_1	VFに自動付与するIPのうち、除外するIP(前半)の開始	`192.168.20.0`
NV_IPAM_EXCLUDE_END_1	VFに自動付与するIPのうち、除外するIP(前半)の終了	`192.168.20.9`
NV_IPAM_EXCLUDE_START_2	VFに自動付与するIPのうち、除外するIP(後半)の開始	`192.168.20.51`
NV_IPAM_EXCLUDE_END_2	VFに自動付与するIPのうち、除外するIP(後半)の終了	`192.168.20.255`

主なコマンドの説明

Makefileが受け付ける主なコマンドを記載する。

クラスタの構築方法は、README.md を参照。

コマンド名	説明
make images	イメージビルドの実行
make create-cluster-with-all	OpenKasugaiコントローラクラスタの起動
make dataflow-<PatternName>	シナリオのデプロイ

イメージ作成手順

以下を実行することにより、kind-control-plane, kind-worker, kind-worker2が使用するコンテナイメージを作成する。

```
1 make images
```

上記により作成されるイメージと用途は以下の通り。

イメージ名	親イメージ	説明	kind-control-plane	kind-worker ※アクセラレータ有	kind-worker2 ※アクセラレータ無

kind-ubuntu	ubuntu official image	kind nodeのベースイメージ。OSバージョンを決定する	ー	ー	ー
kind-ubuntu-node	kind-ubuntu	上記にk8s等をインストールしたもの	ー	ー	ー
kind-ubuntu-node-crio	kind-ubuntu-node	上記のコンテナランタイムをcrioに変更したもの	ー	ー	ー
dci-kind-node-non-acc	kind-ubuntu-node-crio	GPUを搭載していないノード向けのイメージ	○	ー	○
dci-kind-node-with-acc	dci-kind-node-non-acc	GPU搭載ノード向けのイメージ **NVIDIAドライバのインストールを内部で実施している**	ー	○	ー

クラスタ構築手順

以下のどちらかを実行する。OpenKasugai-controllerのリソースの役割を学習する場合や、自身で作成したシナリオを実行する場合は後者を選択する。

- 一括構築手順
- リソースを手動登録する場合の手順

一括構築手順

以下を実行することにより、サンプルシナリオ実行に必要な全ての設定が行われる

```
make create-cluster-with-all
```

リソースを手動登録する場合の手順

シナリオの実行時に必要なリソースを後で手動登録したい場合、以下を実行する。

```
make create-cluster-without-senario
```

クラスタ利用手順

ホスト上からのkubectlの実行

kindによって作成されたクラスタの `.kube/config` は root ユーザのホームディレクトリに配置される。kubectlは、以下のようにroot権限で実行するか、 `.kube/config` を任意のユーザのホームディレクトリにコピーして使用する。

```
sudo kubectl get nodes
```

シナリオの実行

- RUN_SCENARIOS 参照

クラスタ削除手順

以下を実行することにより、OpenKasugai-controllerの実行環境が削除される

```
make delete-cluster
make clean
```