INSTALL（all-in-one）English

Preparation

Hardware

This is the minimum hardware configuration for running all-in-one as follows. The hardware names in parentheses are those used in the test environment.

When you use different hardware below, you will need to read the steps again and change the settings.

- 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)

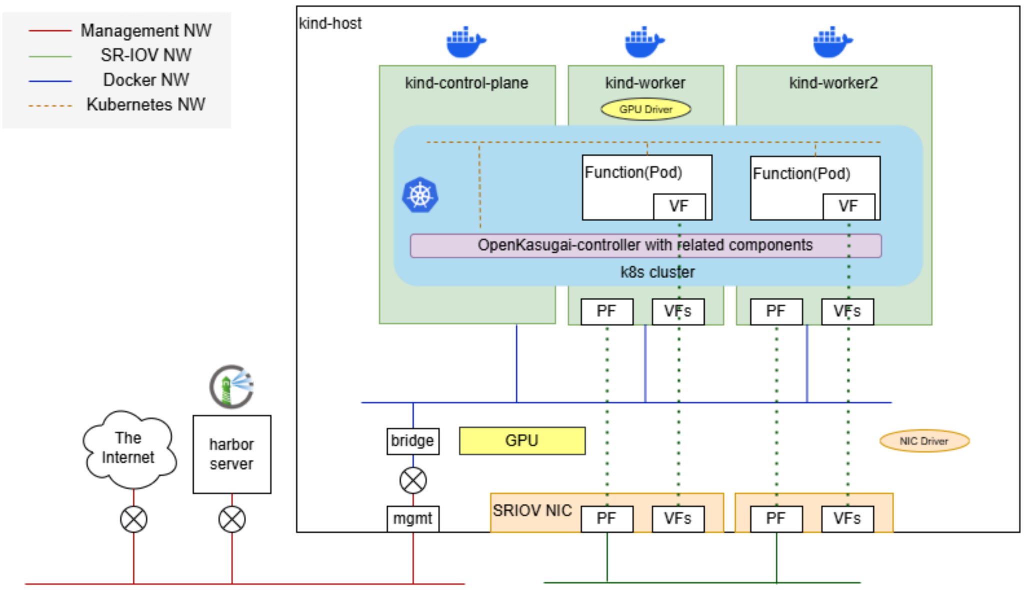
Software

Create an environment with the following installed. Set PATH for kind, golang, and kubectl.

- 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`

Network design

Design the IP addresses for each segment in advance. The `harbor server` is described as an example of a local container registry.



NW	uses	IP(CIDR)	Related configration files	Note
Management	kind-host IP	192.168.10.2	-	-
	harbor server IP	192.168.10.100	./Makefile	• Used to obtain function images • Must be reachable from kind-host
	Gateway	192.168.10.1	-	-
SR-IOV	CIDR	192.168.20.0/24	./Makefile	• Used as a config value for nvidia-k8s-ipam
	PF of kind-worker	192.168.20.2	./Makefile	-
	PF of kind-worker2	192.168.20.3	./Makefile	-
	VFs of kind-worker1 and 2	192.168.20.10-192.168.20.50	./Makefile	• nvidia-k8s-ipam assigns it to the Pod • In order to divide the available IP addresses, the number of IP addresses is set to 30 per node.
	Gateway	192.168.20.1	./Makefile	• Used as a config value for nvidia-k8s-ipam • Must be included in the above CIDR
Docker	Docker Node IP	172.18.0.0/16	-	• Automatic configuration with Docker • No changes are required basically

	External IP of the Pod	172.18.9.0-172.18.9.10	./manifest/metallb/ipaddrespool.yaml	• IP address range that MetalLB assigns to the Service • Change according to the Docker Node IP segment
Kubernetes	PodSubnet	10.100.0.0/16	./kind/kind-values/values.yaml	• Specify CIDR to be used for CNI communication • Set an unused one in the environment

Additional host settings

Disable swap

Delete swap space as needed.

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

Configure Hugepages

1. Edit /etc/default/grub

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

2. Apply Grub settings and reboot

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

3. Check Hugepage settings

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

Enable SR-IOV feature

Here is an example of Mellanox ConnectX-5

1. Get the driver below

- 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

- Files used

```
$ 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. Install driver

- Extract tar

```
$ 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/
...
./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
```

- Run installer

```
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
...
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.
...
Installation passed successfully
To load the new driver, run:
/etc/init.d/openibd restart
```

3. Load driver

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

4. Enable SR-IOV

- Survey of target PF

```
# 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]
```

- Run the SR-IOV enable command

```
# 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
```

- Reboot host

```
# reboot
```

- Create VFs

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

- Check VFs

```
# ip addr
...
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]
```

Placement of necessary files

Place the following files in the repository dir

Location	File name	Note
./src/	controller.tar.gz	- Clone the following repository https://github.com/openkasugai/controller - Archive the repository into a tar.gz archive

		• Change directory name to `controller` • Set recursive option when cloning repository
./driver/nvidia/	NVIDIA-Linux-x86_64-550.90.12.run	• Get it from below https://www.nvidia.com/Download/index.aspx
./work/	*.mp4	• Place video files used to run video inference scenarios ◦ Video files (`*.mp4`) should be located in `./work/DATA/video`. • This directory will bind to `/root/work` in each Docker container

Makefile creation procedure

Modify the parameters in the Makefile to suit your build environment.

Parameter Description

Parameter Name	Description	Default
DCI_REGISTRY_ADDR	OpenKasugai Controller Registry URL	`ghcr.io/openkasugai/controller`
DCI_REGISTRY_ADDR_ALT	Registry URL for the harbor server	`192.168.10.100/images`
DCI_REGISTRY_CERT_URL	Download URL of the registry certificate file of the harbor server	`http://192.168.10.100/harbor/ca.crt`
DCI_K8S_SOFTWARE_TARFILE	The file name of the tar-compressed OpenKasugai controller code	`controller.tar.gz`
NVIDIA_DRIVER_FILE	NVIDIA driver install script file name	`NVIDIA-Linux-x86_64-550.90.12.run`
K8S_VERSION	The version of the k8s cluster to be built with kind	`v1.31.0`
GO_VERSION	Go version to install on kind node	`1.23.0`
SRIOV_CNI_VERSION	The version of SRIOV to use	`v2.8.1`
CRIO_OS	The OS version to use for the kind node	`xUbuntu_22.04`
CRIO_REPO_VERSION	The version of cri-o to install on the k8s cluster (repository)	`v1.31`
CRIO_PKG_VERSION	The version of cri-o to install on the k8s cluster (package)	`1.31.0-1.1`
METALLB_VERSION	The version of MetalLB to use	`v0.14.8`

NVIDIA_K8S_IPAM_VERSION	The version of nvidia-k8s-ipam to use	`v0.3.5`
IMG_TAG	Image tag name when retrieving an image from DockerHub	`22.04.5`
SRIOV_IF_REGEX_1	Regular expression to get the IF names of both the first PF and VF to be assigned to kind-worker	`ens9f0.*`
SRIOV_IF_REGEX_2	Regular expression to get the IF names of both the second PF and VF to be assigned to kind-worker	`ens9f1.*`
SRIOV_PF_IF_1	IF name of the first PF to be assigned to kind-worker	`ens9f0np0`
SRIOV_PF_IF_2	IF name of the second PF to be assigned to kind-worker	`ens9f1np1`
SRIOV_PF_IPADDR_1	IP address (CIDR) to be assigned to the first PF	`192.168.20.2/24`
SRIOV_PF_IPADDR_2	IP address (CIDR) to be assigned to the second PF	`192.168.20.3/24`
SRIOV_NUM_VFS	Number of VFs to be created	`8`
NV_IPAM_SUBNET	Subnet of IP address automatically assigned to VF	`192.168.20.0/24`
NV_IPAM_PER_NODE_BLOCK_SIZE	Number of IP addresses per node for dividing IP addresses automatically assigned to VFs	`30`
NV_IPAM_GATEWAY	Gateway IP address automatically assigned to the VF	`192.168.20.1`
NV_IPAM_EXCLUDE_START_1	Start of IP address to be excluded (first half) from IP addresses automatically assigned to VFs	`192.168.20.0`
NV_IPAM_EXCLUDE_END_1	End of IP address to be excluded (first half) from IP addresses automatically assigned to VFs	`192.168.20.9`
NV_IPAM_EXCLUDE_START_2	Start of IP address to be excluded (second half) from IP addresses automatically assigned to VFs	`192.168.20.51`
NV_IPAM_EXCLUDE_END_2	End of IP address to be excluded (second half) from IP addresses automatically assigned to VF	`192.168.20.255`

Explanation of main commands

This section describes the main commands accepted by Makefile.

Refer to README.md for instructions on how to build a cluster.

Commands	Description
make images	Run the image build.
make create-cluster-with-all	Starting the OpenKasugai controler cluster.
make dataflow-<PatternName>	Deploying a Scenario.

Image Creation Procedure

Run below command and create the container images used by kind-control-plane, kind-worker, and kind-worker2.

```
make images
```

The images created above can be used for the following purposes.

Image	Parent image	Description	kind-control-plane	kind-worker ※with accelerator	kind-worker2 ※without accelerator
kind-ubuntu	ubuntu official image	Base image for kind node, determines the OS version.	—	—	—
kind-ubuntu-node	kind-ubuntu	The image with k8s etc installed on base.	—	—	—
kind-ubuntu-node-crio	kind-ubuntu-node	The above image, installed crio as container runtime.	—	—	—
dci-kind-node-non-acc	kind-ubuntu-node-crio	Image for nodes without GPUs	○	—	○
dci-kind-node-with-acc	dci-kind-node-non-acc	Image for nodes with GPUs **NVIDIA driver installation is done internally.**	—	○	—

Cluster construction procedure

Execute one of the following. Select the latter if you want to learn the role of OpenKasugai-controller resources or run a scenario you created yourself.

- Bulk build procedure
- Procedure for manually registering resources

Bulk build procedure

By executing the following, all the settings required to run the sample scenario will be configured.

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

Procedure for manually registering resources

If you want to manually register the resources required for scenario execution later, do the following:

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

Cluster usage procedure

Running kubectl from the host

The `.kube/config` of the cluster created by kind is placed in the home directory of the root user. To use kubectl, run it with root privileges as follows, or copy `.kube/config` to the home directory of any user.

```
sudo kubectl get nodes
```

Scenario execution

- See RUN_SCENARIOS

Cluster deletion procedure

By executing the following, the OpenKasugai-controller execution environment will be deleted.

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