ADVANCED CLOUD COMPUTING

CONTAINERS

DEEP DIVE, PART 2

CONTAINER ENGINE: PODMAN

- singlenode container projects (no orchestration)
- developing, building, and running containerized applications.
- as simple to use as Docker, same command-line interface
- clean interaction with systemd
- out-of-the-box rootless mode.

config files:

- `/etc/containers`
- `~/.config/containers`

CONTAINER ENGINE: CRI-O

- designed for kuberneets, little use outside
- limeted docker compatibility
- `crictl` can be used to interact with it

config files:

- `/etc/crio`
- `/etc/cni`, standard support for differnt network plugins

PODMAN COMMANDS

command	action
`podman ps (--all)`	List running containers (and stopped)
`podman run (-it --rm)`	runs a container (interactive mode, remove at the end)
`podman stop`	stop a running container
`podman start`	starts a stopped container
`podman rm`	remove a stopeed container (possible data loss)
`podman images`	list available container images
`podman image rm`	remove an image
`podman exec`	execute a command in a pod

PODMAN DEMO INTERACTIVE

Stream online

VOLUMES

`--volume HOST-DIR:CONTAINER-DIR`

several options are available, most important is z:Z for selinux Choose the z option to label volume content as shared among multiple containers. Choose the Z option to label content as unshared and private.

Podman supports other kinds of volumes as well

NAMED VOLUMES

```
$ podman  volume create webdata
# where data are stored
$ podman volume inspect --all | jq ".[].Mountpoint"
```

EMPTY DIR

managing temporary storage within a Pod's lifecycle, initially empty

(they survive pod crash, not rescheduling/termination)

containers in the Pod can read and write the same files in the emptyDir volume

EMPTY DIR

Other key Considerations:

- Node-Local Storage:
  EmptyDir volumes are tied to the node where the Pod is scheduled.
- Performance:
  While EmptyDir volumes can provide high performance, the actual performance depends on the underlying storage medium and node configuration.

EMPTY DIR

usage:

- Temporary Data Storage(Scratch space, Caching)
- Inter-Container Communication within a pod
- High-Performance Storage (if `medium: Memory`)

  Common applications

  - Data Processing Pipelines
  - Machine Learning
  - Web Applications
  - Database Systems

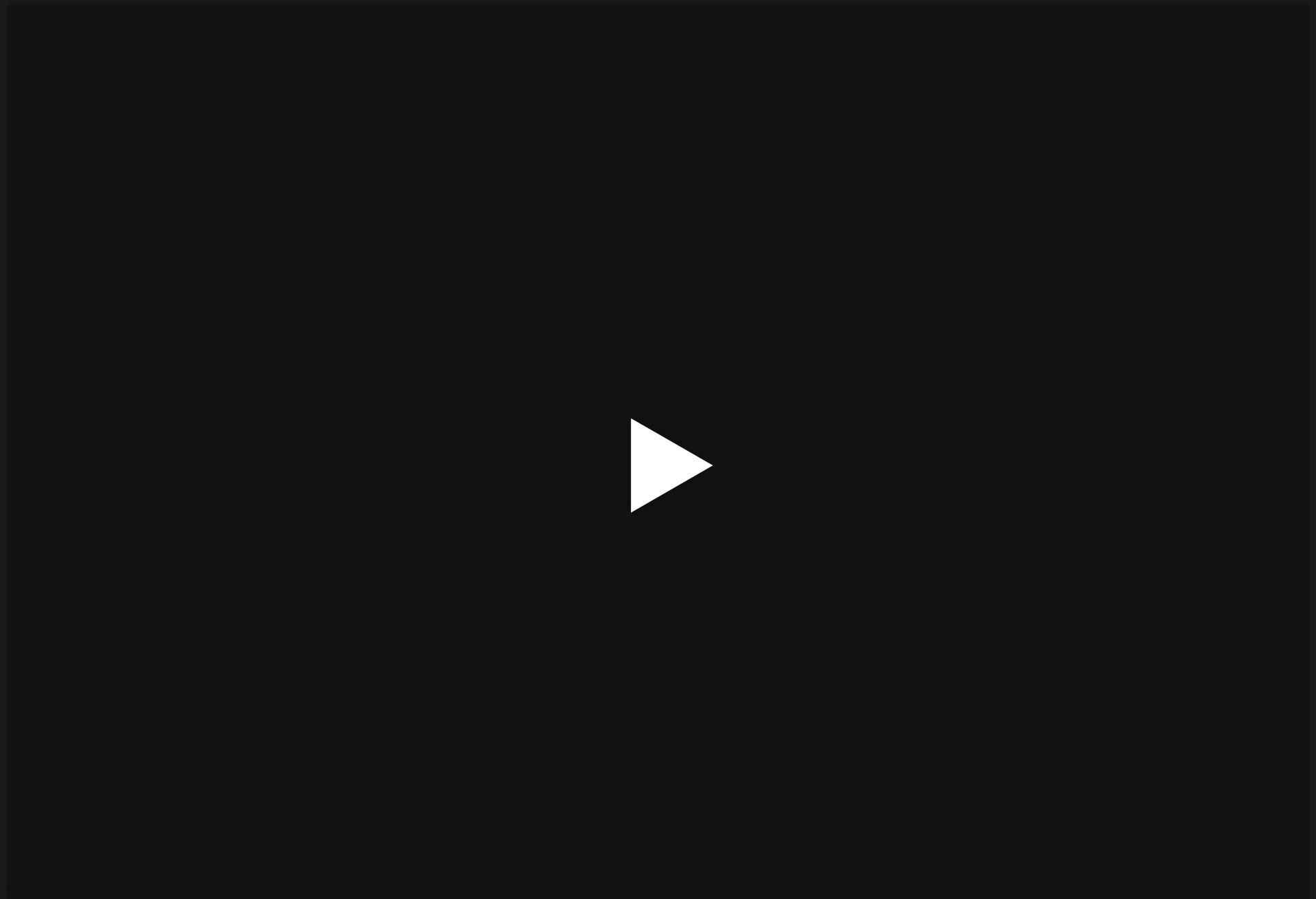
PORTS

`-p HOST-PORT:POD-PORT`

to expose a service on localhost

EXERCISE: A WEBSERVER

Stream online



YAML IN ONE SLIDE

```yaml
---
a_string: "string"
a_float: 3.14159
a_int: 3
a_bool: true
a_bool2: no
a_list:
  - s1
  - s2
  - s3
  - s4
a_dict:
  key: value
  list: [a, b, c]
  sub_dict:
```

TO PARSE IT

```
#!/usr/bin/env python

import json
import yaml
import sys

with open(sys.argv[1]) as f:
    print(json.dumps(yaml.load(f, Loader=yaml.SafeLoader),
          indent=2))
```

or yq, useful for paring kubernetes output

WHY YAML?

BECAUSE KUBECTL AND HELM SPEAK YAML

A POD OBJECT

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: server
    image: python:3.11.6-alpine
    command: ["sh", "-c", "python /opt/server.py"]
    volumeMounts:
    - mountPath: /opt/
      name: py-sources
    - mountPath: /workdir/
      name: data-sharing
  - name: client
```

SEVER CODE

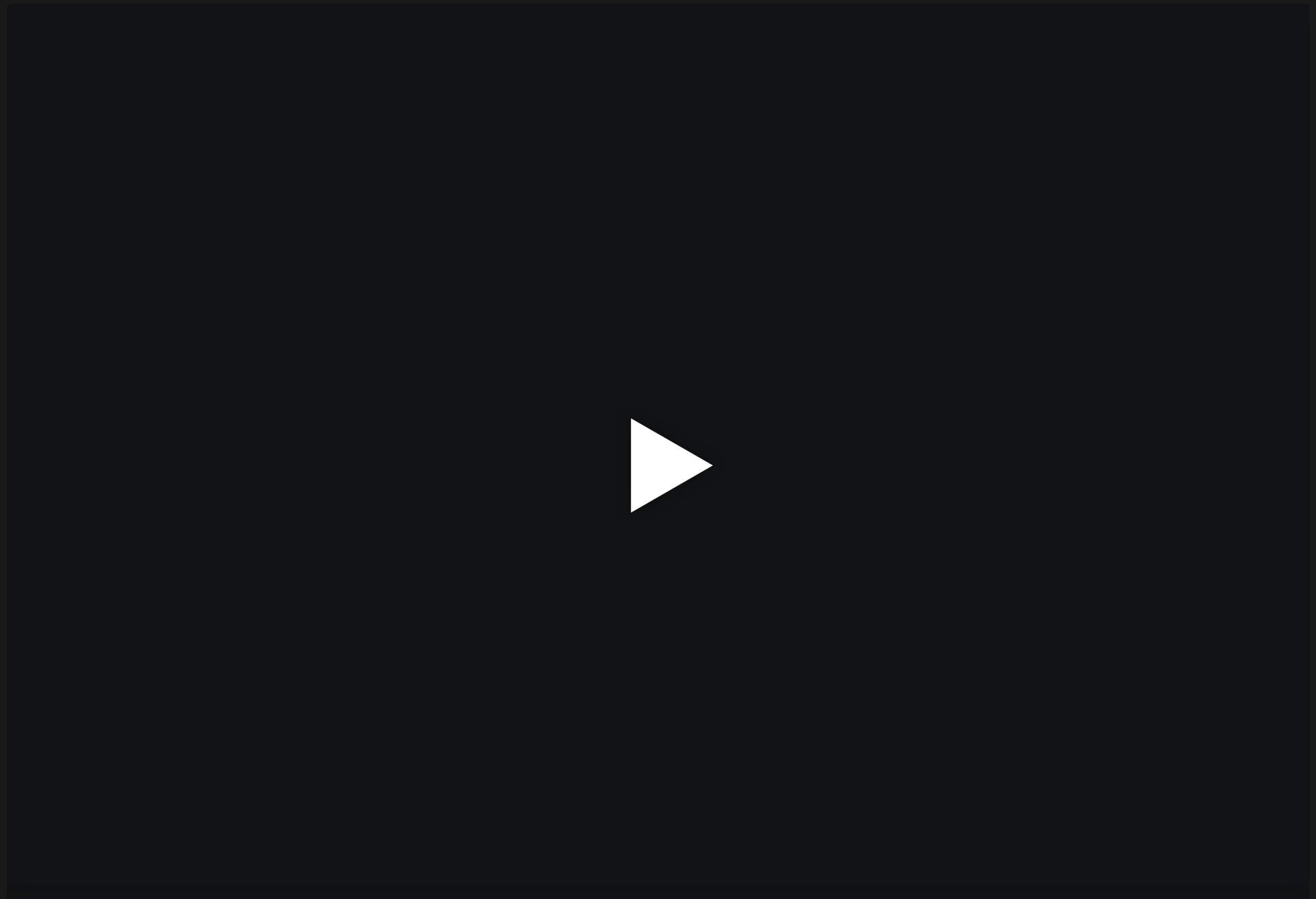
```python
import socket
import os

socket_name = "/workdir/application.socket"
s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)

try:
    os.remove(socket_name)
except OSError:
    pass

s.bind(socket_name)
s.listen(1)
conn, addr = s.accept()
```

CLIENT CODE

```
1  import socket
2
3  socket_name = "/workdir/application.socket"
4  s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
5  s.connect(socket_name)
6
7  s.send(b'Hello, server!')
8  data = s.recv(1024)
9  s.close()
10
11 print('Received form server: ' + repr(data))
12
```

RUNNING IT

Stream online



CONTAINER DEVICE INTERFACE (CDI)

SPECIFIC DEVICES (EG. GPU)

specification, for container-runtimes, to support third-party devices.

- abstract notion of a device as a resource,
- devices are specified by a Fully qualified name (FQN) `vendor.com/class=unique_name`

SCOPE: enabling containers to be device aware. NO resource management.

WHY?

Theory: passing a device to a container = exposing a device node

Practice: is more than that...

- exposing more than one device node,
- mounting files from the runtime namespace,
- hiding procfs entries,
- Performing compatibility checks
  (Can this container run on this device?).
- Performing runtime-specific operations
  (e.g: VM vs Linux container-based runtimes).
- Performing device-specific operations
  (e.g: scrubbing the memory of a GPU).

WHY?

To converge in from the currenty situation of vendors multiple plugins, for different runtimes or even directly contribute vendor-specificcode in the runtime.

HOW DOES IT WORKS?

- JSON or YAML in `/etc/cdi` and `/var/run/cdi` (depends on runtime configuration)
- FQN's should be passed to the runtime using the container engine options

Usually there are tools to fill these folders

```
nvidia-ctk cdi generate --output=/etc/cdi/nvidia.yaml
```

RUNNING AN LLM

```
podman run --device nvidia.com/gpu=all \
   -v ~/AI/ollama:/root/.ollama \
   -p 11434:11434 \
   --security-opt=label=disable \
   --name ollama ollama/ollama
```

Usage

```
podman exec -it ollama ollama run gemma:7b
```

RUNNING AN LLM

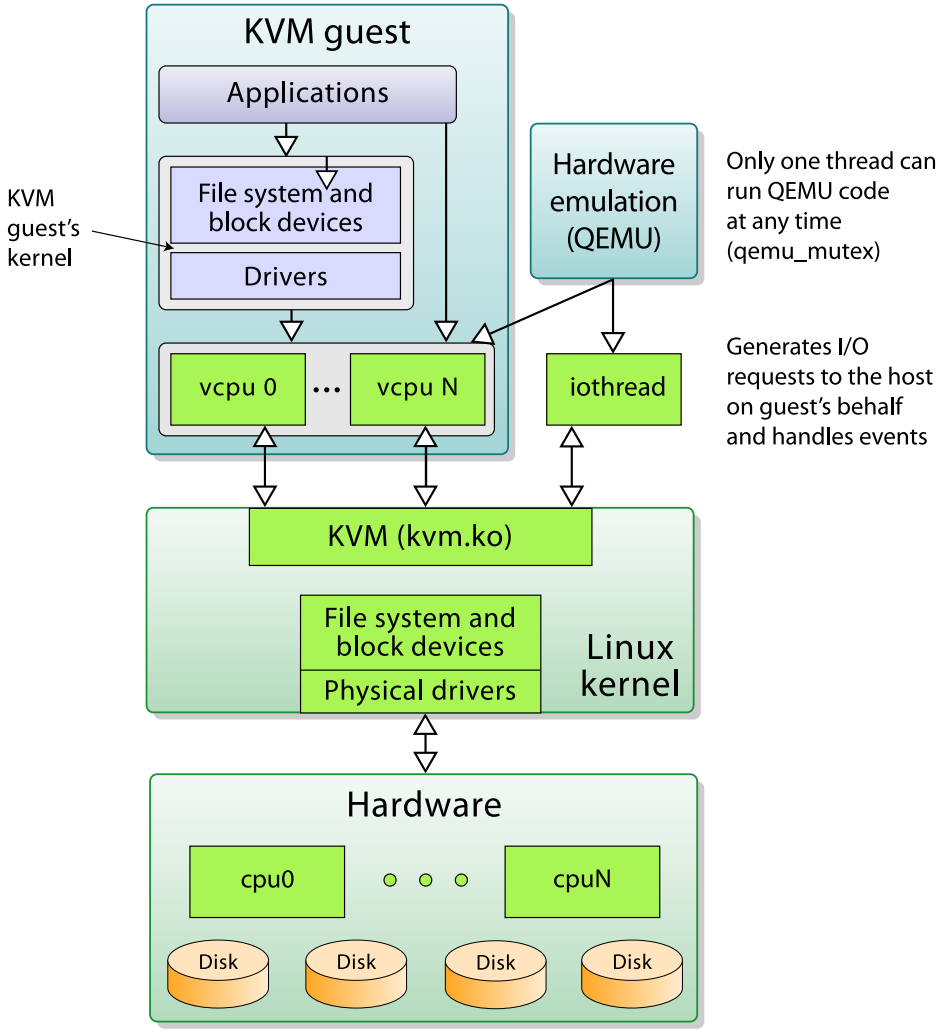
```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 560.35.03              Driver Version: 560.35.03      CUDA Version: 12.6     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |                  MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce MX250           Off |   00000000:3C:00.0 Off |                  N/A |
| N/A   71C    P0             N/A / ERR!  |    1369MiB /   2048MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A   1775090      C   ...unners/cuda_v12/ollama_llama_server       1366MiB |
+-----------------------------------------------------------------------------------------+
```

KUBERNETS

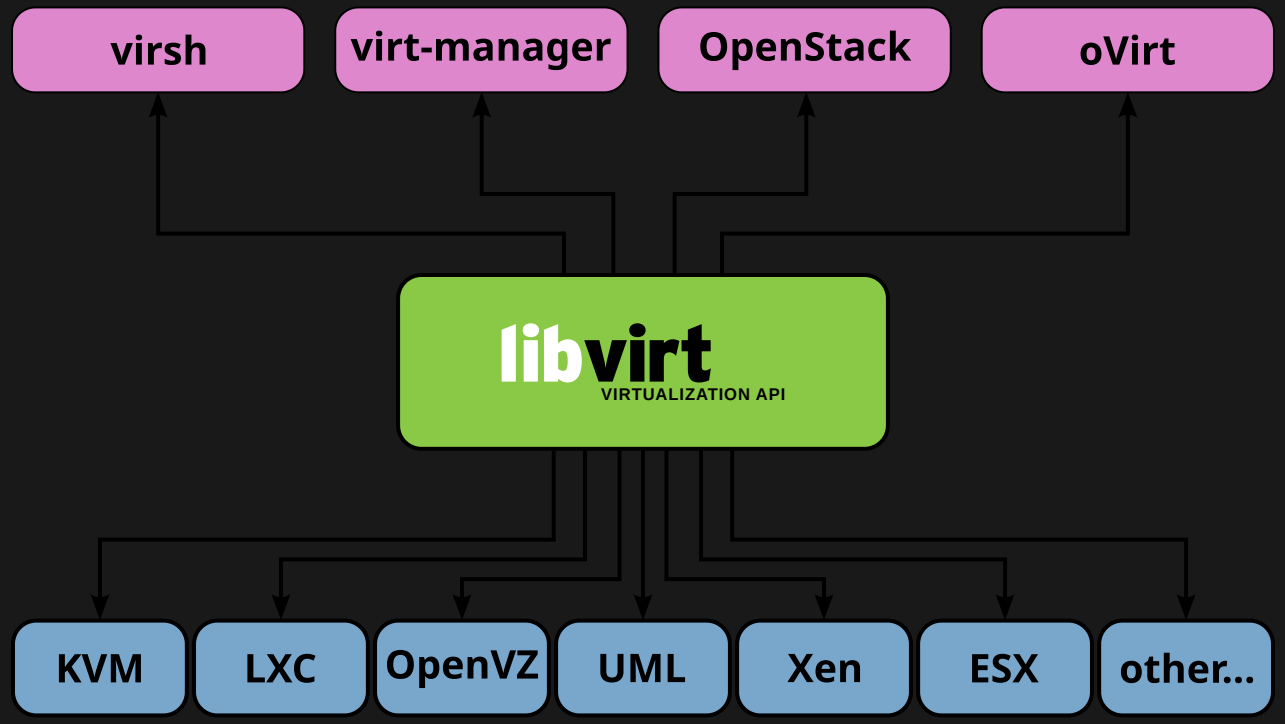
DEPLOYMENT

IN A REALISTIC ENVIRONMENT

QUEMU/KVM


KVM guest
Applications
File system and block devices
Drivers
vcpu 0
vcpu N
KVM guest's kernel
Hardware emulation (QEMU)
Only one thread can run QEMU code at any time (qemu_mutex)
iothread
Generates I/O requests to the host on guest's behalf and handles events
KVM (kvm.ko)
File system and block devices
Physical drivers
Linux kernel
Hardware
cpu0
cpuN
Disk
Disk
Disk
Disk

VIRSH


virsh
virt-manager
OpenStack
oVirt
libvirt
VIRTUALIZATION API
KVM
LXC
OpenVZ
UML
Xen
ESX
other...

VIRSH

```
<network>
    <name>kub-devel</name>
    <forward mode="nat">
        <nat>
            <port start="1024" end="65535">
        </port></nat>
    </forward>
    <ip address="192.168.133.1" netmask="255.255.255.0">
        <dhcp>
            <range start="192.168.133.2" end="192.168.133.29">
        </range></dhcp>
    </ip>
</network>
```

VAGRANT

tool that simplifies the process of setting up and managing virtual development environments

VAGRANT

KEY CONCEPTS:

- Install a Provider (libvirt)
- Create a Vagrantfile (ruby)
- Start/destroy the VM
- SSH into the VM
- Provision the VM

VAGRANTFILE

```
servers = [
  { :hostname => "k01", :ip => "192.168.133.80" },
  # { :hostname => "k02", :ip => "192.168.133.81" },
]

Vagrant.configure("2") do |config|
  config.vm.box = "fedora/41-cloud-base"

  config.vm.provider :libvirt do |lv|
    lv.qemu_use_session = false
    lv.memory = 2048
    lv.cpus = 2
  end

  servers.each do |conf|
```

VIRTUAL ORFEO

Name	Last commit	Last update
.reuse	started using reuse	1 year ago
LICENSES	fix licences	1 year ago
manifests	rm submodule	1 month ago
playbooks	remove submodule dependency	1 month ago
scripts	Add license	10 months ago
vagrantfiles	more powerfull ipa vm	1 month ago
.gitignore	remove personal git ignore	7 months ago
.gitmodules	rm submodule	1 month ago
README.md	updated command to checkout correct ...	4 months ago
requirements.txt	missing requirement	7 months ago

Gitlab

Repo

SYSTEM SETUP

```
# Load modules
modprobe overlay
modprobe br_netfilter

# make load permanent
cat << EOF | tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

# change kernel parameters
cat << EOF |  tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
```

installation

```
# attention to exclude!!
cat << EOF | tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/re
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

KUBERNETES INSTALLATION

```
# utils
dnf install iproute-tc wget vim bash-completion bat -y
# CRI-o
dnf install crio -y
# real kube
dnf install -y kubelet kubeadm kubectl \
  --disableexcludes=kubernetes
```

KUBERNETES INSTALLATION

```
sed -i 's/10.85.0.0\/16/10.17.0.0\/16/' /etc/cni/net.d/100-cri
systemctl enable --now crio
systemctl enable --now kubelet

kubeadm init --pod-network-cidr=10.17.0.0/16
# --services-cidr=10.96.0.0/12 /default
# --control-plane-endpoint 192.168.132.80 /needed for HA
```

BASIC MANAGEMENT OPERATIONS

A USEFUL DASHBOARD K9S

DISECTING THE COMPONENTS THAT ARE PRESENT

BEYOND SINGLE NODE DEPLOYMENT

CNI