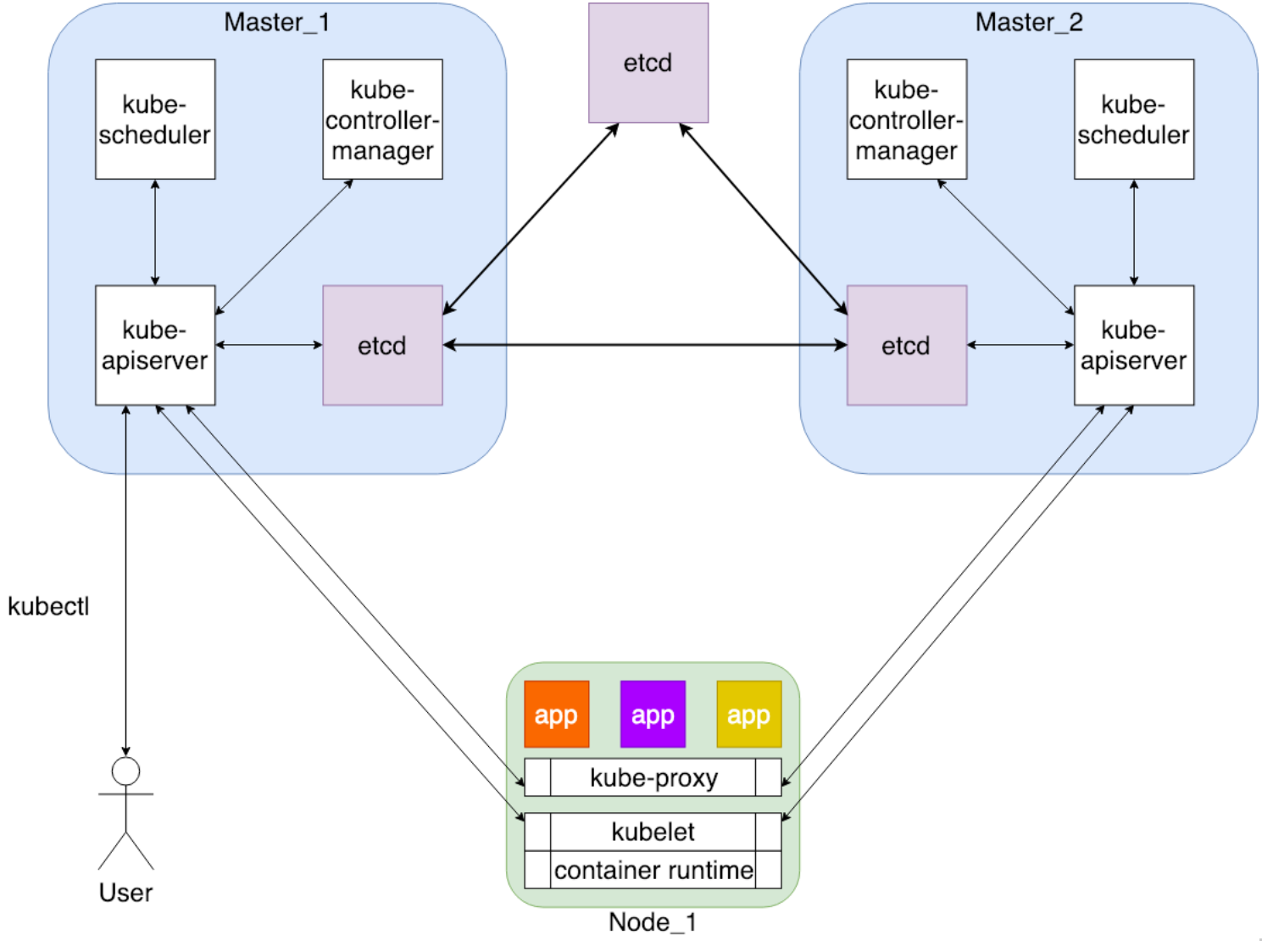
Container Runtime Interface. Обзор контейнерных runtime

Не забудь включить запись!

План

- Docker
- OCI
- CRI
- Обзор runtime
- Сборка образов внутри контейнеров

Kubernetes and runtimes



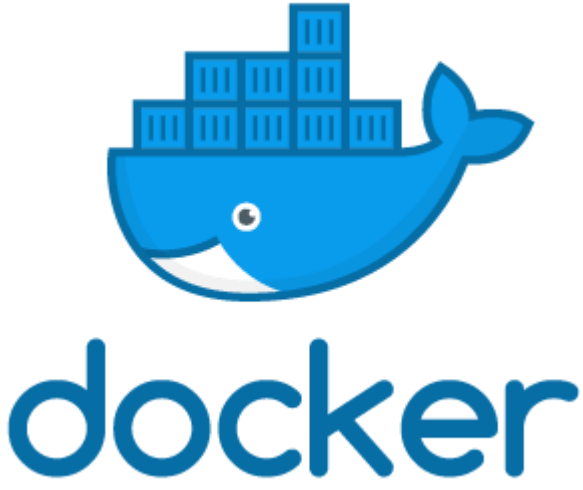
Kubernetes and runtimes

Kubernets	Runtimes
k8s API	Pod container lifecycle (start, stop, delete)
Storage	Image management (pull, status)
Networking (CNI)	Status
Healthchecks	Container interactions (exec, log)
Placement	

...

Docker

- Появился в 2013 году
- Представляет удобную обертку над технологиями linux (namespaces, cgroups, etc.)
- На ранних этапах развития был монолитным приложением



OCI (Open Container Initiative)

Две спецификации:

- Runtime Specification - что такое контейнер
- Image Specification - как запустить контейнер

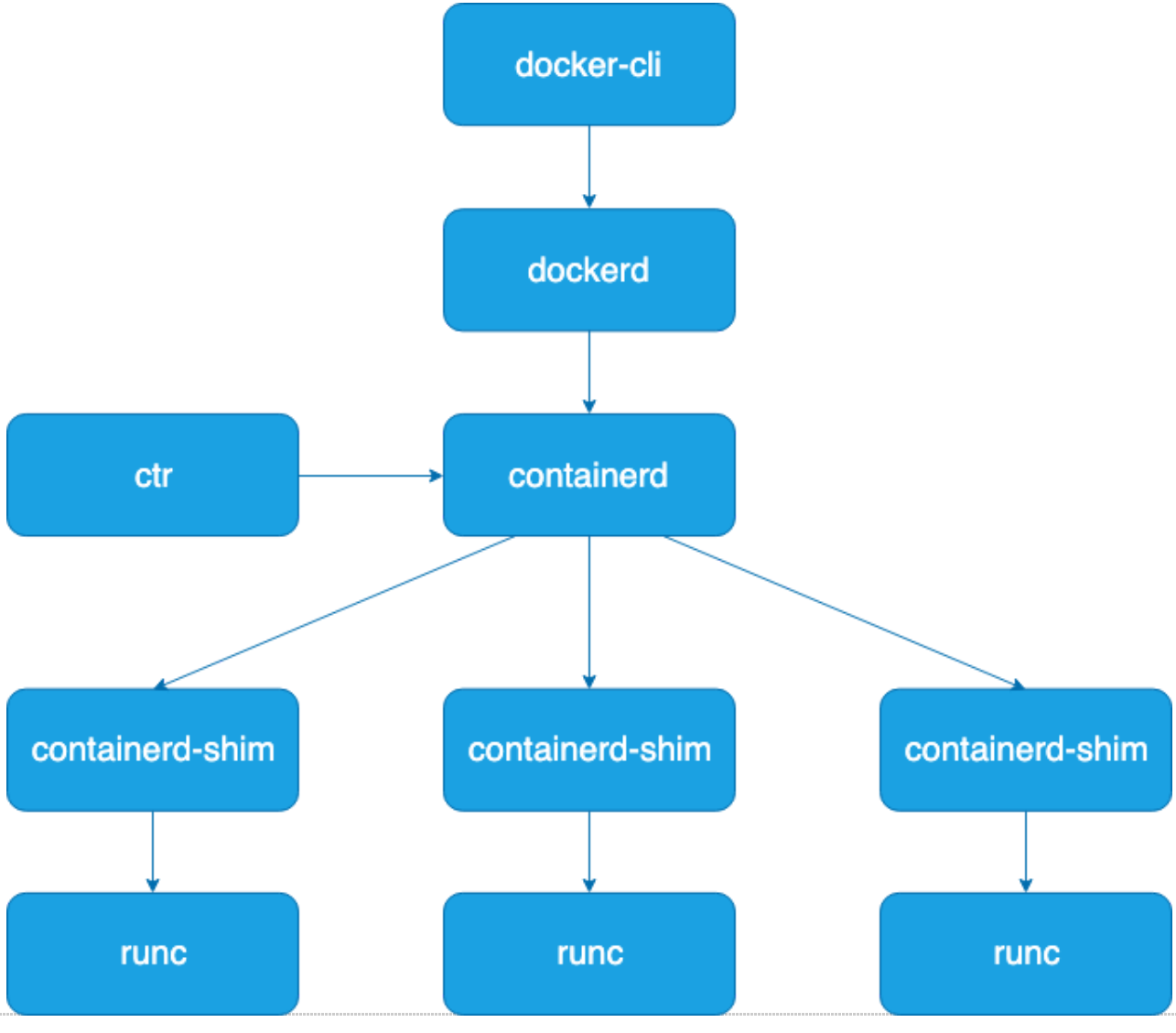
OCI (Open Container Initiative)

OCI (Open Container Initiative)

Процесс запуска контейнера

1. Загрузка OCI совместимого образа
2. Распаковка образа в OCI bundle
3. Запуск контейнера из bundle

Архитектура Docker



Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

Containerd

- Часть, вынесенная из монолитного Docker и отвечающая за управление жизненным циклом контейнеров (pull/push, run, stop)
- Создавалась для управления сторонними системами, такими как k8s (взаимодействие с containerd напрямую через cli не является целевой моделью)
- Поддерживает CRI*

Containerd

Упрощая, задача containerd состоит в следующем:

1. Получение образа контейнера из registry
2. Распаковка его в OCI bundle
3. Запуск данного OCI bundle
4. Контроль состояния запущенного контейнера

ctr

- CLI-утилита для дебага containerd
- Не поддерживается официально, но устанавливается в комплекте с Docker и containerd

```
root@k8s-worker-containerd:~# ctr --namespace=k8s.io containers list
CONTAINER                                                           IMAGE
1672bc2d186532ad3bb77d1b64a197d25e1256f4e8c9c133c734853af3d49e3e    sha256:eb516548c180f8a6e0235034ccee2428027896af16a509786da13022fe95fe8c
2dba41bacddc05d813889337419acfb46b955d3c72a81eeda95c4b0516298be3    sha256:8a9c4ced3ff92c63c446d55c2353c42410c78b497a0483d4048e8d30ebe37058
80efda30e9199eff3bf9340ad40d2b77c49630c59e707b987e3c1cd0d4e028a6    sha256:da86e6ba6ca197bf6bc5e9d900febd906b133eaa4750e6bed647b0fbe50ed43e
a305ec4147cf2c26a4544c898b55b4928b6f95a415ea9a5d6448bb75524b8982    sha256:8a9c4ced3ff92c63c446d55c2353c42410c78b497a0483d4048e8d30ebe37058
aa4f530d183c0910561c40f7dd7509b4352a4a0b2b9353698cb73e1fb51979e3    sha256:da86e6ba6ca197bf6bc5e9d900febd906b133eaa4750e6bed647b0fbe50ed43e
ad821408fd315d49597c0b1c6ee1d95e691f2e8d985ed7d98b5acb1f56b2fe55    sha256:da86e6ba6ca197bf6bc5e9d900febd906b133eaa4750e6bed647b0fbe50ed43e
c35c7628c3f9647b010f75160fe34cbcd9c867aae806ef5caf64bb956390a521    sha256:5c24210246bb67af5f89150e947211a1c2a127fb3825eb18507c1039bc6e86f8
```

Containerd-shim

- Позволяет runc завершиться непосредственно после старта контейнера (daemonless containers)
- Поддерживает файловые дескрипторы даже в случае остановки containerd/dockerd

runc

CLI утилита для запуска контейнеров в соответствии со спецификацией OCI

Что надо, чтобы запустить контейнер:

- rootfs контейнера
- config.json - спецификация

rootfs + **config.json** = **OCI bundle**

Демо. Запускаем контейнер при помощи runc

Container Runtime Interface (CRI)

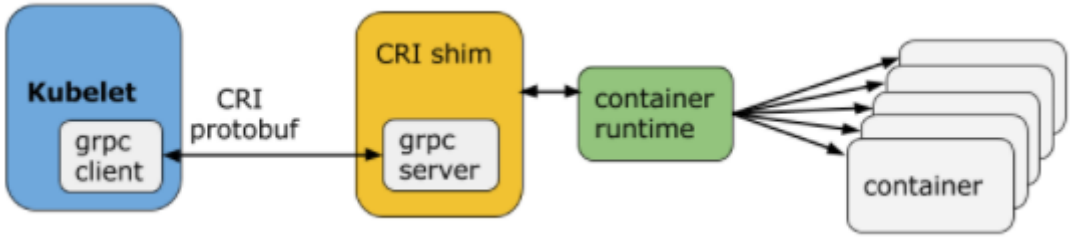
До CRI

> Docker and rkt were integrated directly and deeply into the kubelet source code through an internal and volatile interface.

- Сложно добавлять новые runtime при необходимости
- Сложно поддерживать совместимость runtime и kubelet
- Разработчики runtime должны фокусироваться на архитектуре kubelet

Container Runtime Interface (CRI)

- gRPC
- Client (kubelet) - Server (runtime) model
- Plug & Play



Container Runtime Interface (CRI)

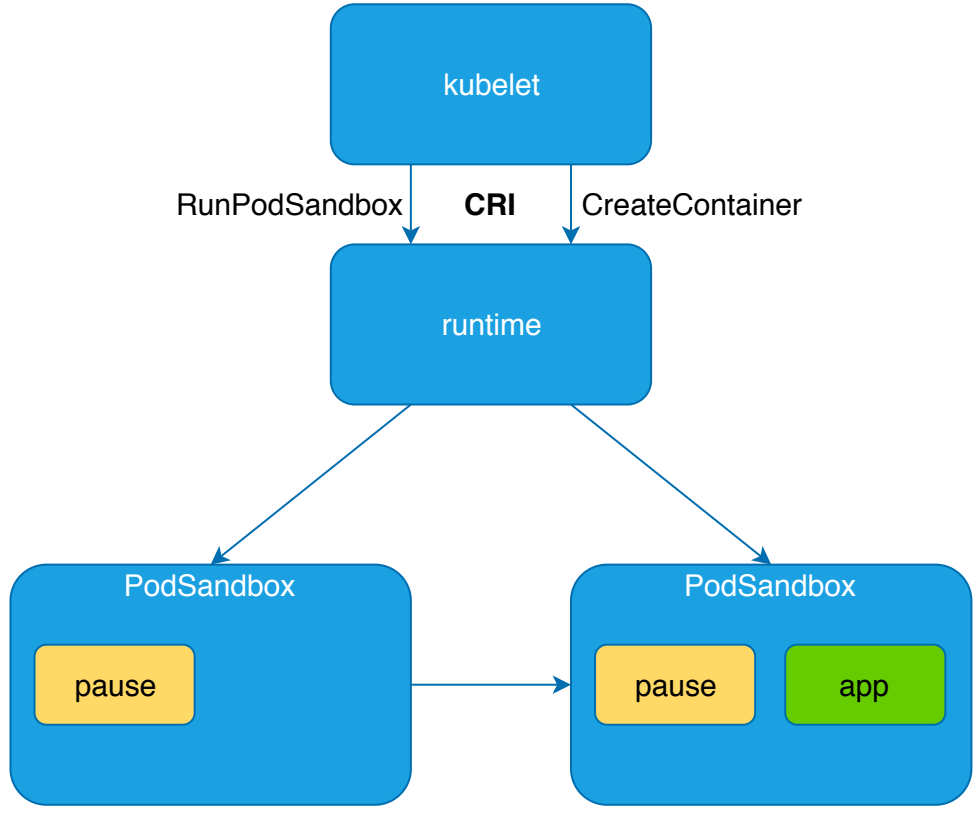
Процесс запуска pod на хосте

Создание **PodSandbox** - изолированного окружения для запуска pod. Оно обладает следующими характеристиками:

- Изоляция - обеспечивается при помощи linux namespaces или виртуализации
- Спецификация используемых ресурсов - **PodSandbox** должен реализовывать требования и ограничения ресурсов на уровне pod (например, с использованием cgroups)

Container Runtime Interface (CRI)

Процесс создания pod



Container Runtime Interface (CRI)

Операции с **PodSandbox**

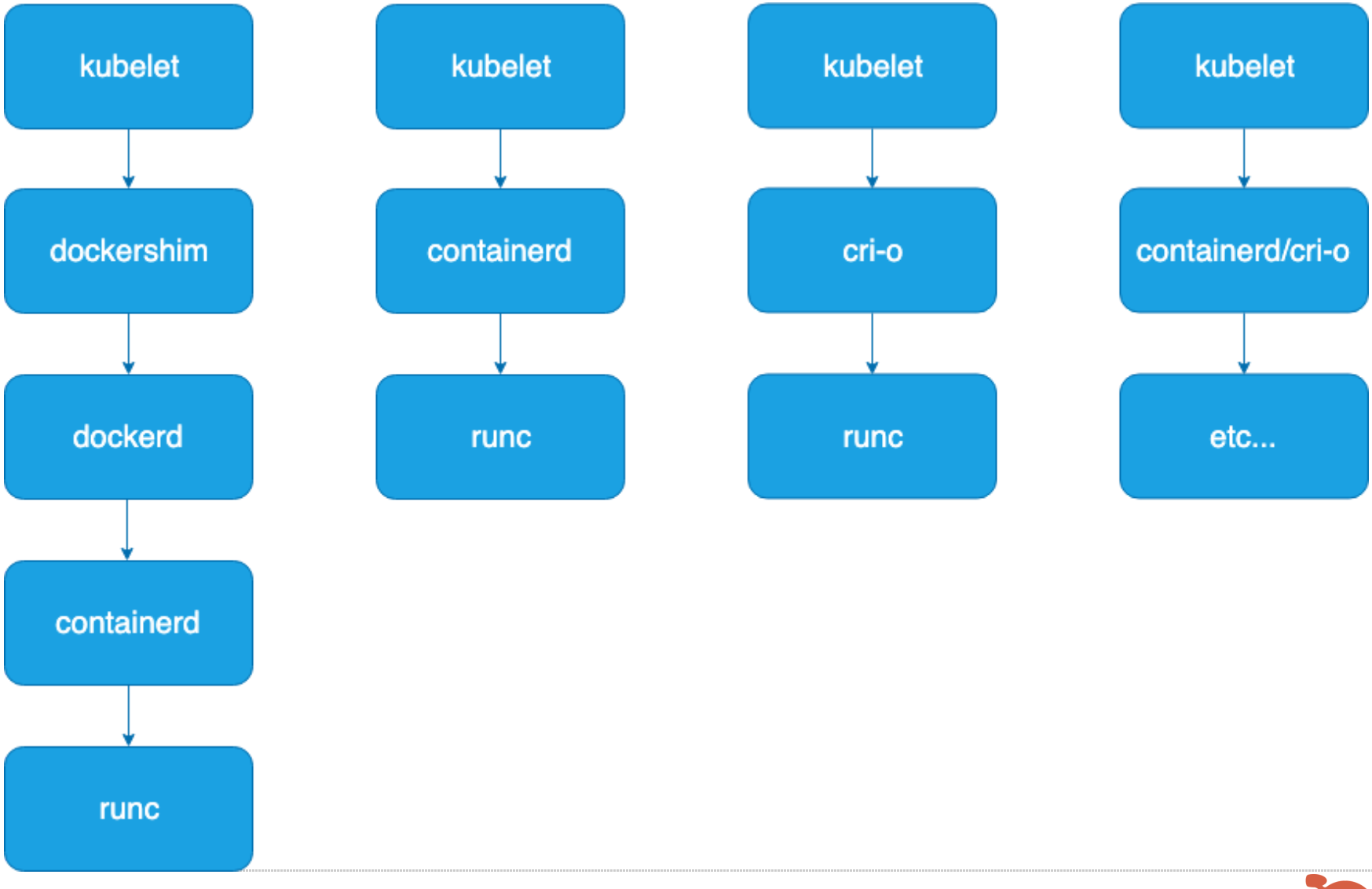
```
type PodSandboxManager interface {
    // RunPodSandbox creates and starts a pod-level sandbox. Runtimes should ensure
    // the sandbox is in ready state.
    RunPodSandbox(config *runtimeapi.PodSandboxConfig, runtimeHandler string) (string, error)
    // StopPodSandbox stops the sandbox. If there are any running containers in the
    // sandbox, they should be force terminated.
    StopPodSandbox(podSandboxID string) error
    // RemovePodSandbox removes the sandbox. If there are running containers in the
    // sandbox, they should be forcibly removed.
    RemovePodSandbox(podSandboxID string) error
    // PodSandboxStatus returns the Status of the PodSandbox.
    PodSandboxStatus(podSandboxID string) (*runtimeapi.PodSandboxStatus, error)
    // ListPodSandbox returns a list of Sandbox.
    ListPodSandbox(filter *runtimeapi.PodSandboxFilter) ([]*runtimeapi.PodSandbox, error)
    // PortForward prepares a streaming endpoint to forward ports from a PodSandbox, and
  returns the address.
    PortForward(*runtimeapi.PortForwardRequest) (*runtimeapi.PortForwardResponse, error)
}
```

Container Runtime Interface (CRI)

Операции над контейнерами в **PodSandbox**

```
type ContainerManager interface {
    CreateContainer(podSandboxID string, config *runtimeapi.ContainerConfig, sandboxConfig *runtimeapi.PodSandboxConfig) (string, error)
    StartContainer(containerID string) error
    StopContainer(containerID string, timeout int64) error
    RemoveContainer(containerID string) error
    ListContainers(filter *runtimeapi.ContainerFilter) ([]*runtimeapi.Container, error)
    ContainerStatus(containerID string) (*runtimeapi.ContainerStatus, error)
    UpdateContainerResources(containerID string, resources *runtimeapi.LinuxContainerResources) error
    ExecSync(containerID string, cmd []string, timeout time.Duration) (stdout []byte, stderr []byte, err error)
    Exec(*runtimeapi.ExecRequest) (*runtimeapi.ExecResponse, error)
    Attach(req *runtimeapi.AttachRequest) (*runtimeapi.AttachResponse, error)
    ReopenContainerLog(ContainerID string) error
}
```

Container Runtime Interface (CRI)



Runtimes

Docker

- Первый runtime в k8s
- Наиболее популярен на текущий момент
- Включает в себя containerd, но взаимодействие с CRI реализовано через dockershim (часть kubelet)

Containerd

- Продукт от компании Docker
- Возможно подключение в GKE в beta режиме
- Входит в состав Docker
- Содержит имплементацию CRI - `cri-containerd`

Containerd

Graduated

Kubernetes

Orchestration

Prometheus

Monitoring

Envoy

Service Proxy

CoreDNS

Service Discovery

containerd

Container Runtime

Fluentd

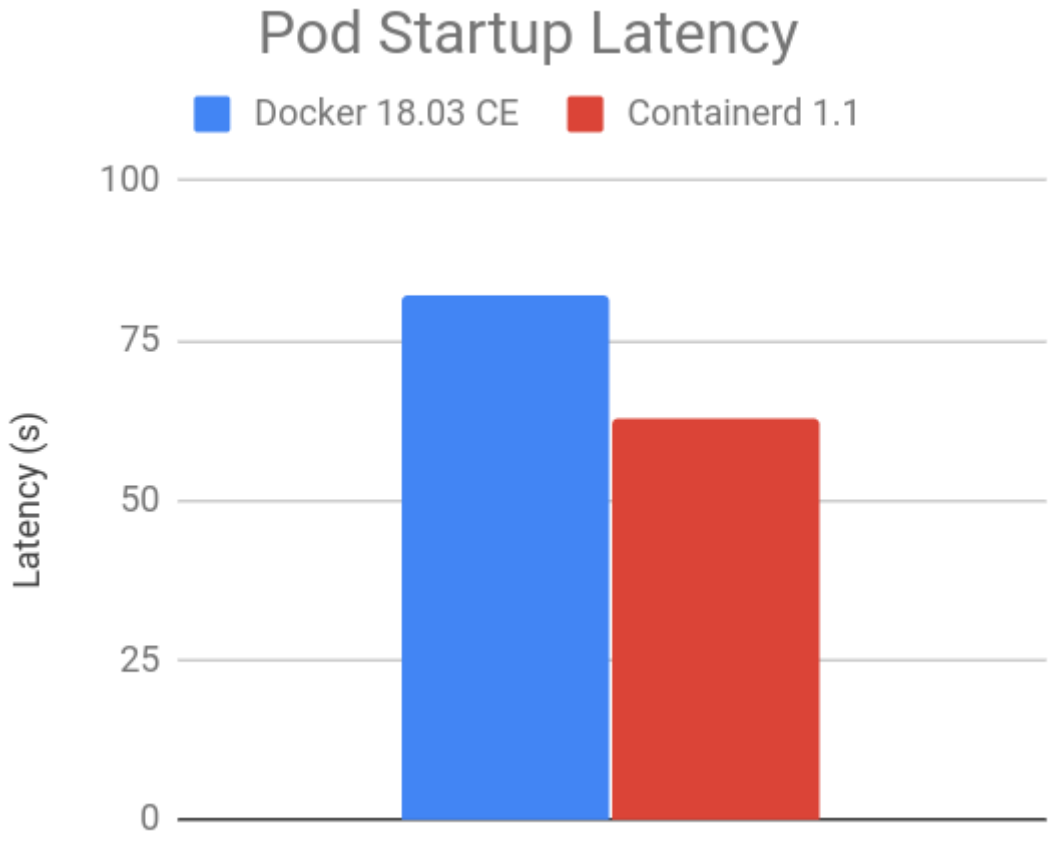
Logging

Graduated («выпускники») — проекты с коммитерами хотя бы из двух организаций, бейджем соответствия лучшим практикам CII (Core Infrastructure Initiative), принятым CNCF Code of Conduct, ясно определённой схемой управления и принятия коммитов, публичным списком пользователей и, конечно, положительным исходом соответствующего голосования технического комитета CNCF.

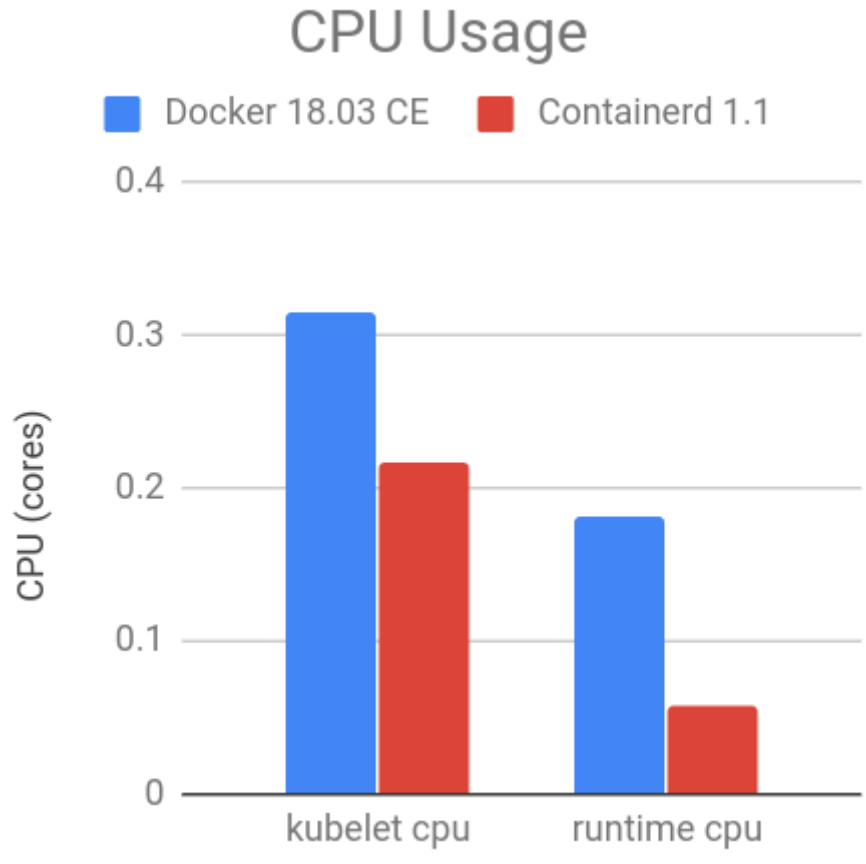
Containerd

Взаимодействие kubelet и containerd

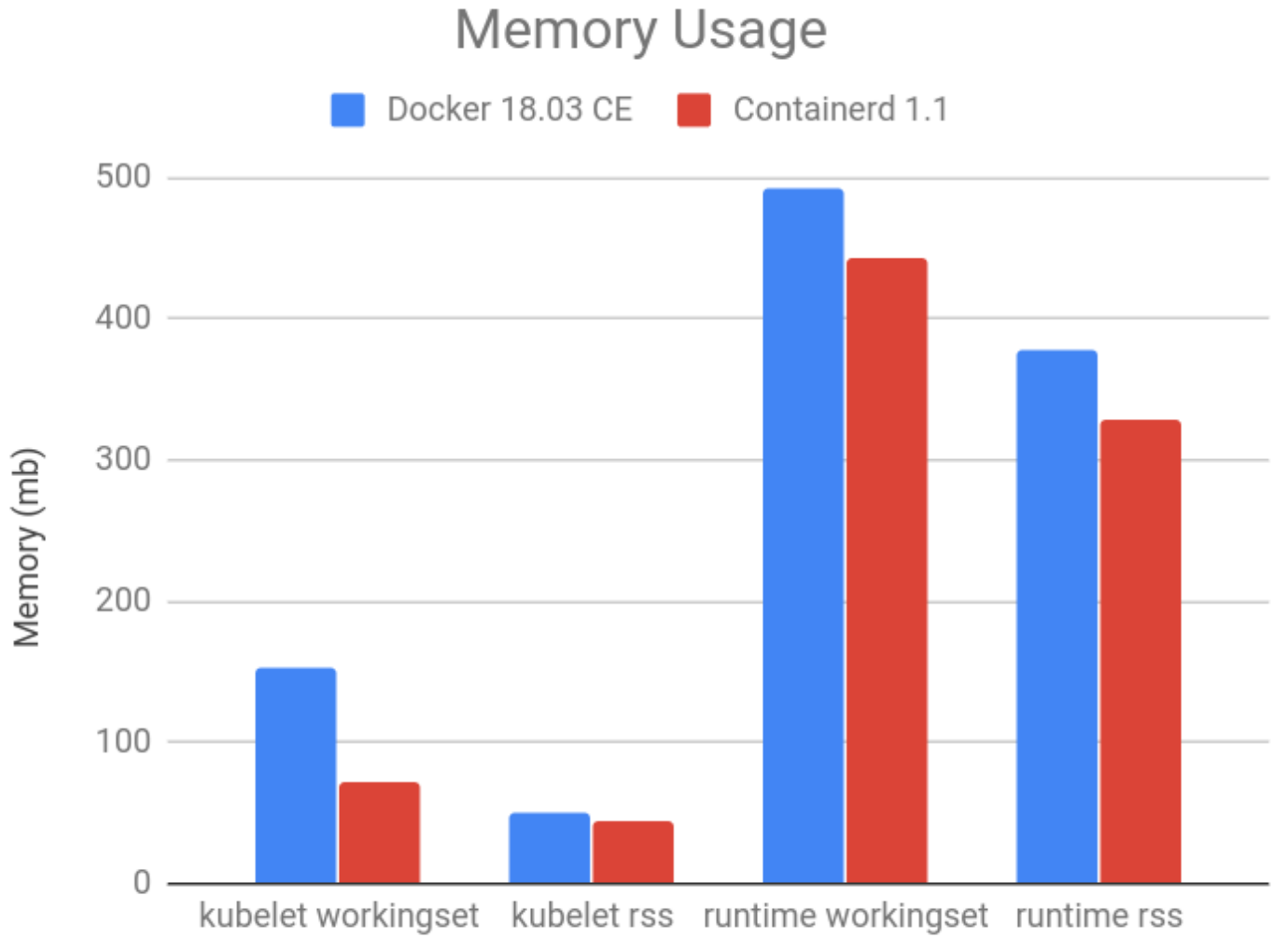
Docker vs Containerd



Docker vs Containerd



Docker vs Containerd

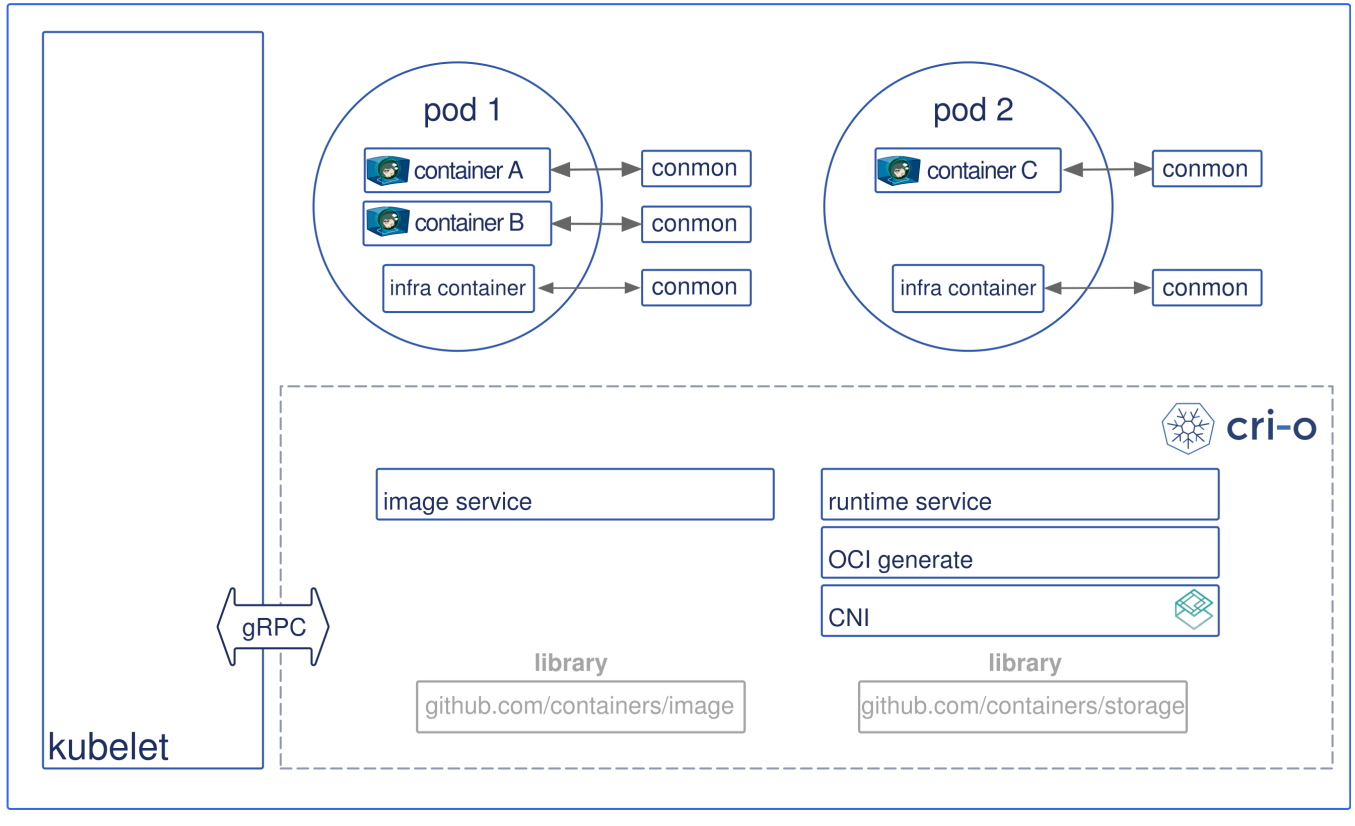


CRI-O

- Имплементация CRI-совместимого runtime от компании Redhat
- Официально поддерживает runc и kata контейнеры, но можно подключить любой OCI совместимый runtime
- Используется в OpenShift 4.x по умолчанию
- Релизный цикл аналогичен релизному циклу k8s:

Version - Branch	Kubernetes branch/version	Maintenance status
CRI-O 1.12.x - release-1.12	Kubernetes 1.12 branch, v1.12.x	=
CRI-O 1.13.x - release-1.13	Kubernetes 1.13 branch, v1.13.x	=
CRI-O 1.14.x - release-1.14	Kubernetes 1.14 branch, v1.14.x	=
CRI-O HEAD - master	Kubernetes master branch	✓

Архитектура CRI-O



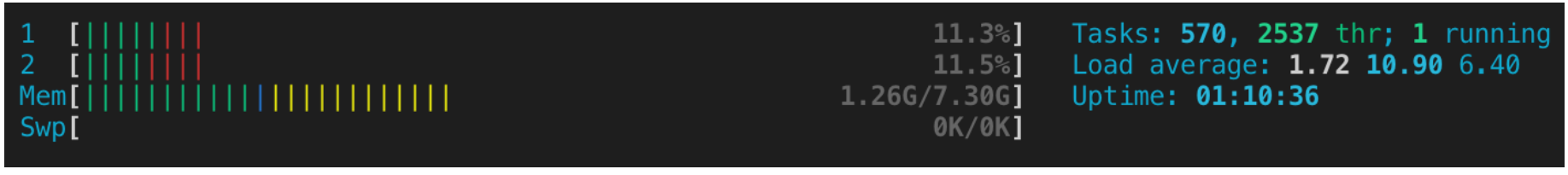
Docker vs Containerd vs CRI-O

Время перехода 105 pod nginx в состояние "Running" (образ предварительно выкачан на ноды):

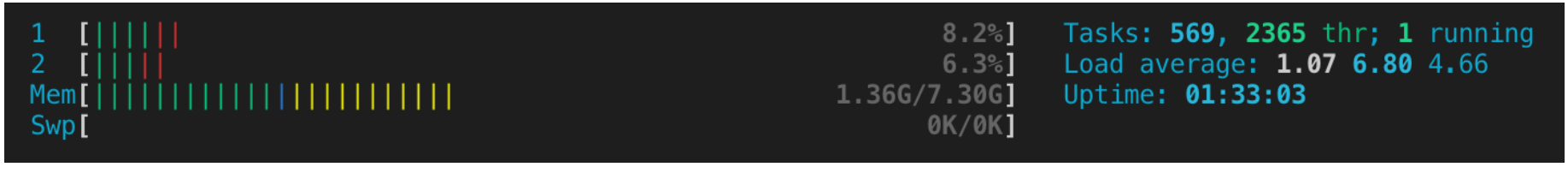
Runtime	Старт первого pod	Старт последнего pod
Docker	67 секунд	109 секунд
Containerd	36 секунд	98 секунд
CRI-O	28 секунд	80 секунд

Docker vs Containerd vs CRI-O (CPU/RAM)

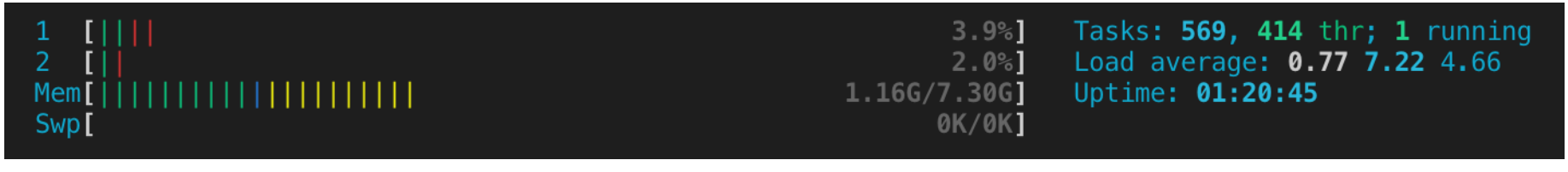
Docker:



Containerd:



CRI-O:



How Container Runtimes matter in Kubernetes

Демо. Сравниваем производительность runtime

Podman

```
alias docker=podman
```

Podman

- Часть контейнерной экосистемы RedHat
- Помогает перейти на k8s
- Оперирует как контейнерами, так и pod'ами
- Daemonless and rootless

podman

Rootless Docker

- Функционал добавлен в релизе 19.03.0
- PR
- Katakoda
- Описание
- Дополнение

Демо. Запускаем контейнеры в Podman

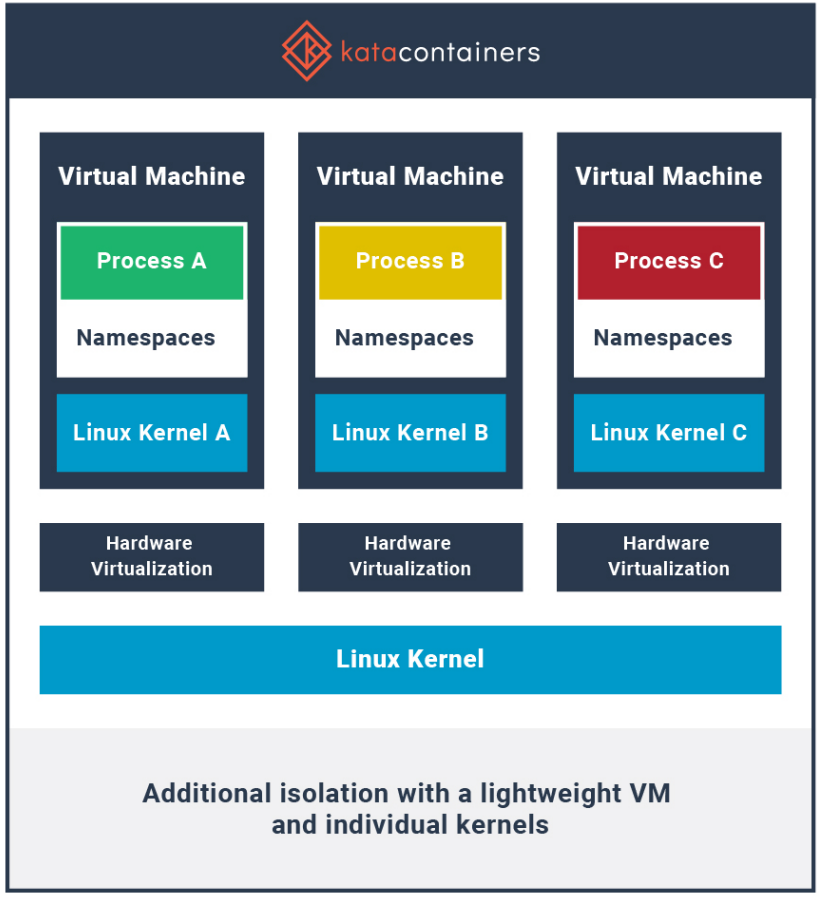
Kata Containers

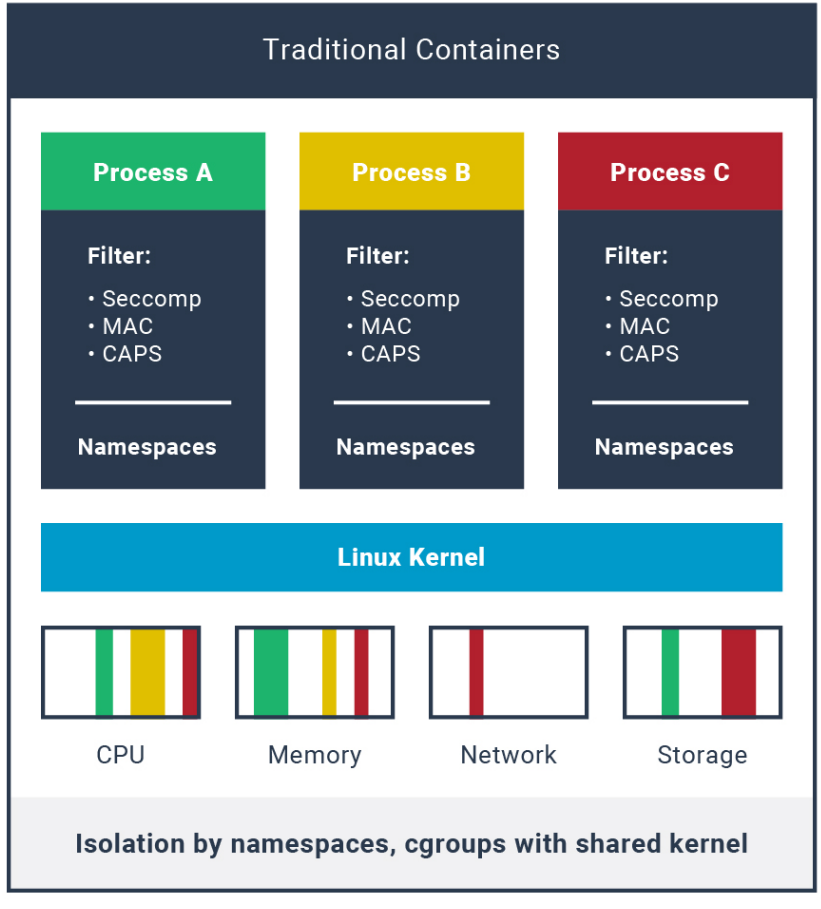
The agility of containers, the security of VMs

- Легковесные виртуальные машины на основе QEMU/KVM или AWS Firecracker*
- OCI compatible
- CRI-O/Containerd compatible
- Подходит для запуска недоверенных сервисов

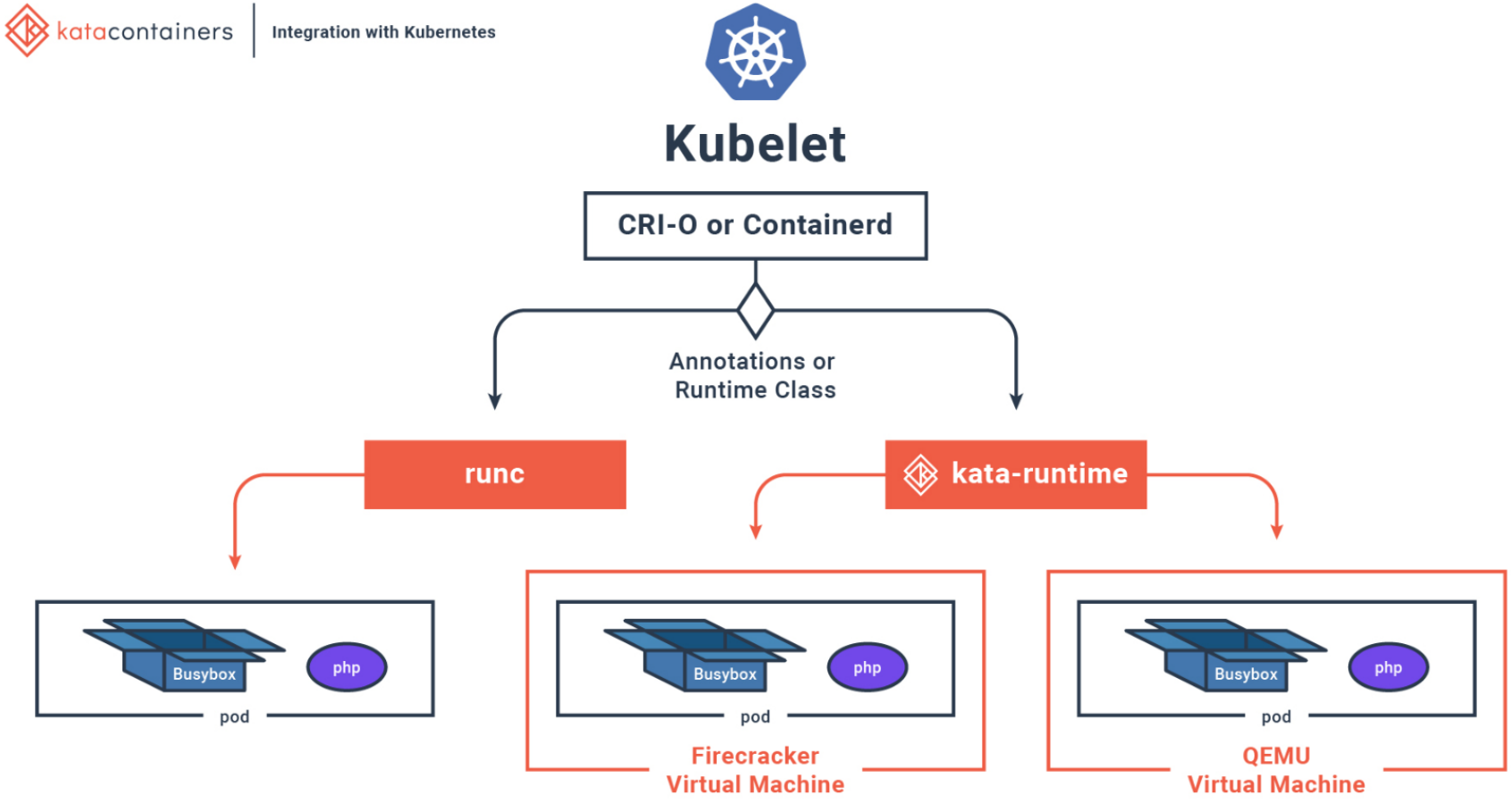
Kata Containers

Контейнеры и виртуальные машины





Kata Containers



Демо. Запускаем контейнеры (ВМ) в Kata

AWS Firecracker

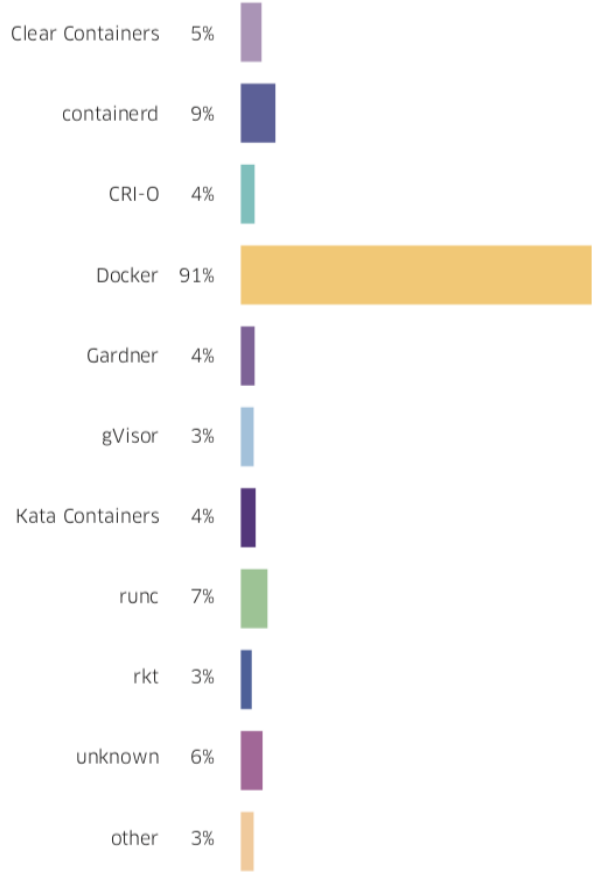
Can Firecracker be used with Kubernetes, Docker, or Kata containers today?

Not yet. We are making Firecracker open source because it provides a meaningfully different approach to security for running containers. We hope that others in the communities that build open source container technology find it useful. We are working to make Firecracker integrate naturally with the container ecosystem, with the goal to provide seamless integration in the future to provide more choices in how container workloads are isolated.

Runtimes

- gVisor
- Singularity
- Nabla
- ...

Container Runtimes usage



The State of Container and Kubernetes Security. Spring 2019

Сборка образов внутри контейнеров

Docker in Docker (dind)

- Docker Engine
- Privileged*
- Bind Mount /var/lib/docker*

> Using Docker-in-Docker for your CI or testing environment? Think twice. 2015
>
> Towards unprivileged container builds

Socket Mount

- Docker-cli
- Использование Dockerd хоста
- Стандартный* и наиболее широко используемый способ на текущий момент (но не в GitLab CI)

```
docker run -ti -v /var/run/docker.sock:/var/run/docker.sock docker:stable
```

Buildah

- Часть контейнерной экосистемы RedHat
- Свой способ описания сборки, но поддерживается Dockerfiles
- Не требует Docker
- Используется в OpenShift 4.x

buildah

Buildah

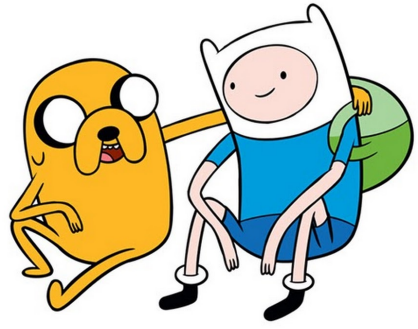
- Не позволяет собирать образы внутри контейнеров без повышения привилегий (необходимо создание user namespace)
- 'overlay' is not supported over overlayfs (решается запуском с ключом `--storage-driver=vfs`)
- Встроен в Podman, но может использоваться как отдельный инструмент

buildah

Kaniko

- Сборка образа без Docker Daemon
- Не требует повышения привилегий
- Рекомендуется запуск из заранее собранного образа `gcr.io/kaniko-project/executor`
- Отлично подходит для CI-агента сборщика
- Поддержка кеширования*

Демо. Собираем Docker образ в Kaniko



Спасибо за внимание!

Время для ваших вопросов!