云原生WASM

刘振伟

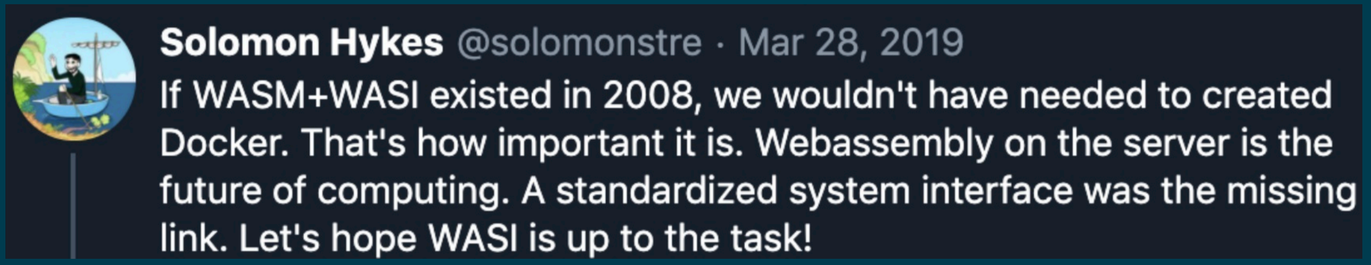
/thoughtworks

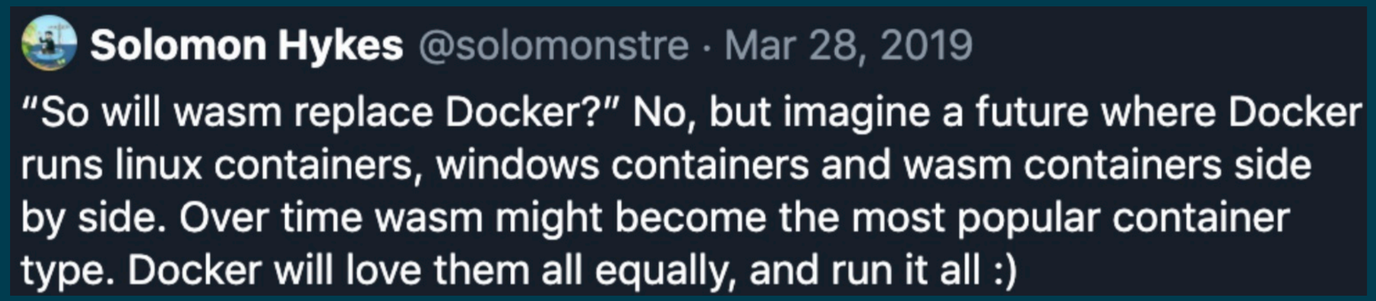
云原生WASM

In Scope: 云原生对WASM的支持

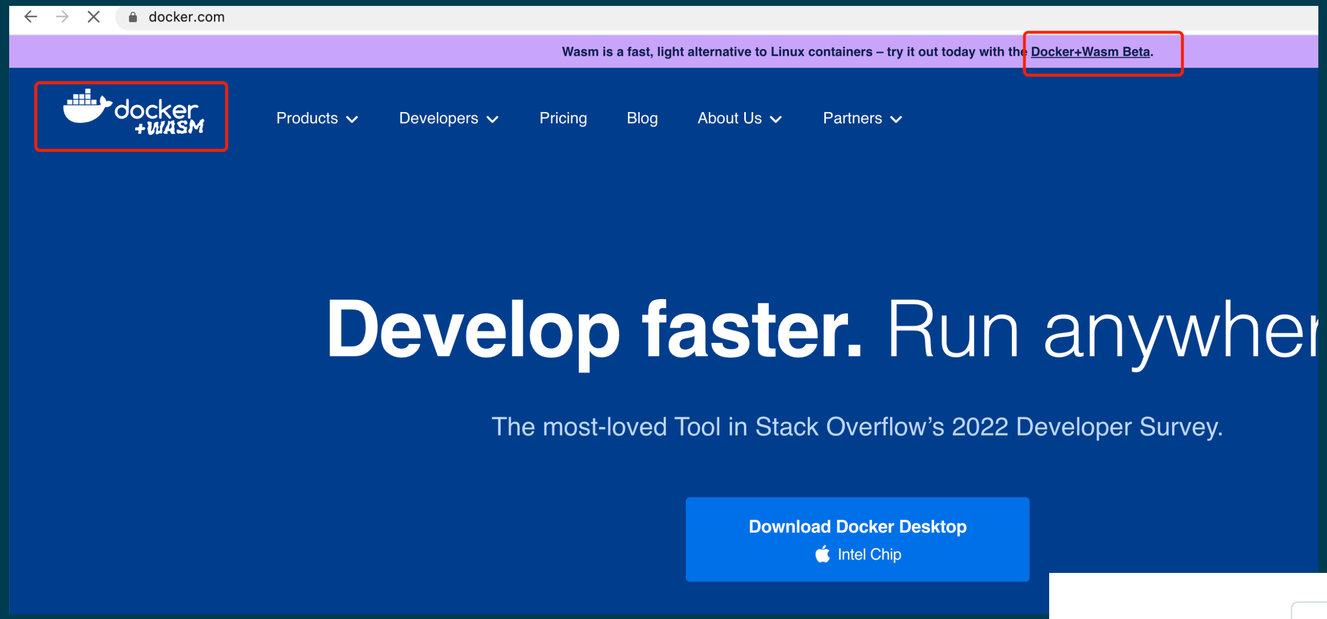
Out Of Scope: WASM自身的相关概念和原理等

WebAssembly(WASM)



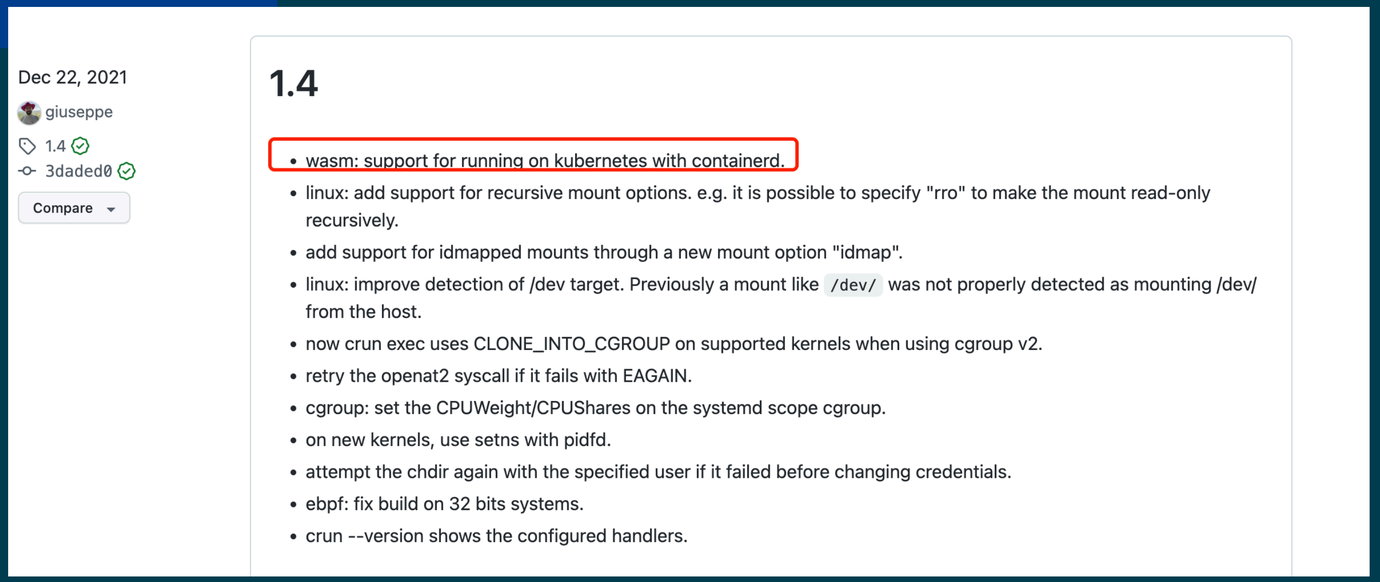


WASM 正在被云原生接纳

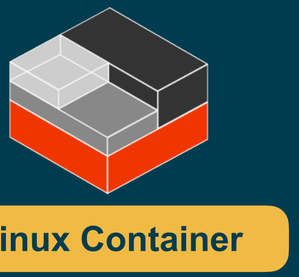


Docker

Crun



VS



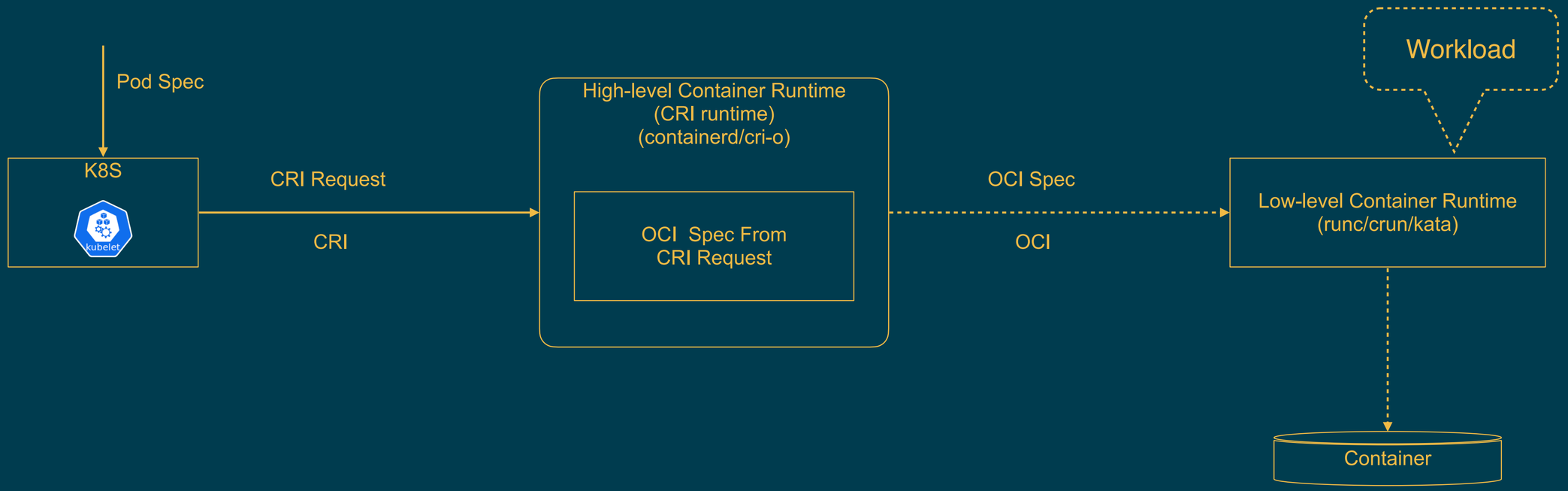
A container emulates a private operating system.

A WebAssembly instance emulates a process.

_ Taylor McMullen, CTO, Fastly

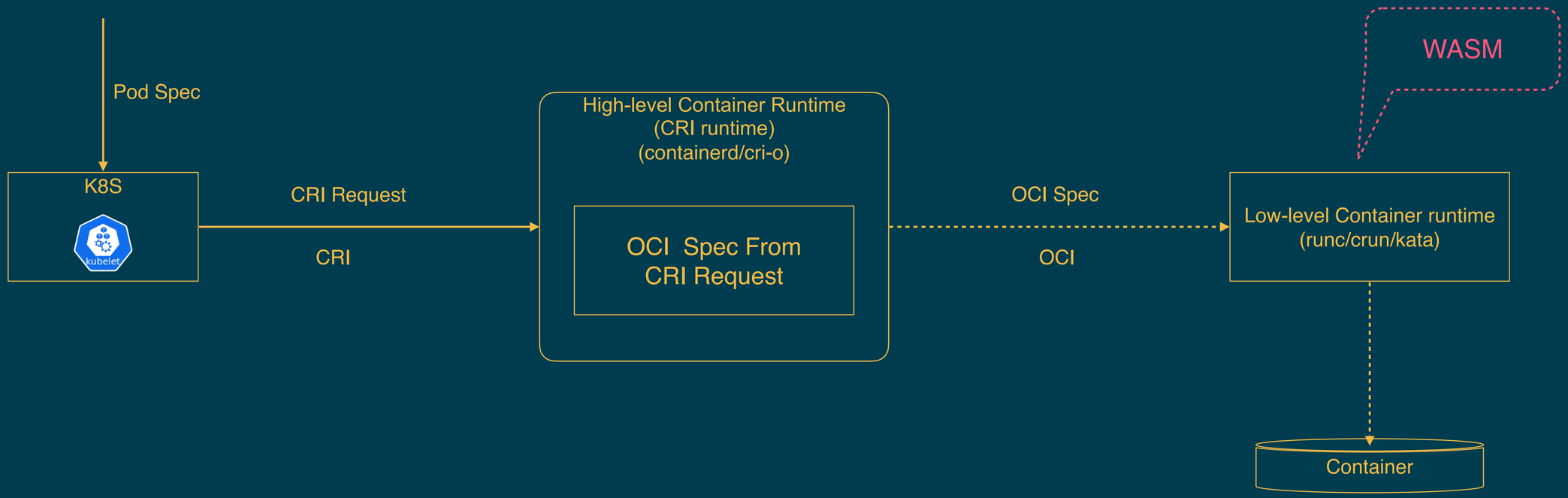
Wasm	Performance	Container
Several MBs	DISK FOOTPRINT	Several GBs
milliseconds	STARTUP PERFORMANCE	seconds
AOT is within 10% of native client	RUNTIME PERFORMANCE	10% to 20% loss to native client

Workload 创建流程



关于docker的接入： kubelet—-> cri-dockerd—->dockerd—->containerd——>……

在Low-level Container Runtime中支持WASM Workload



通过扩展Low-level Container的能力来支持WASM Workload

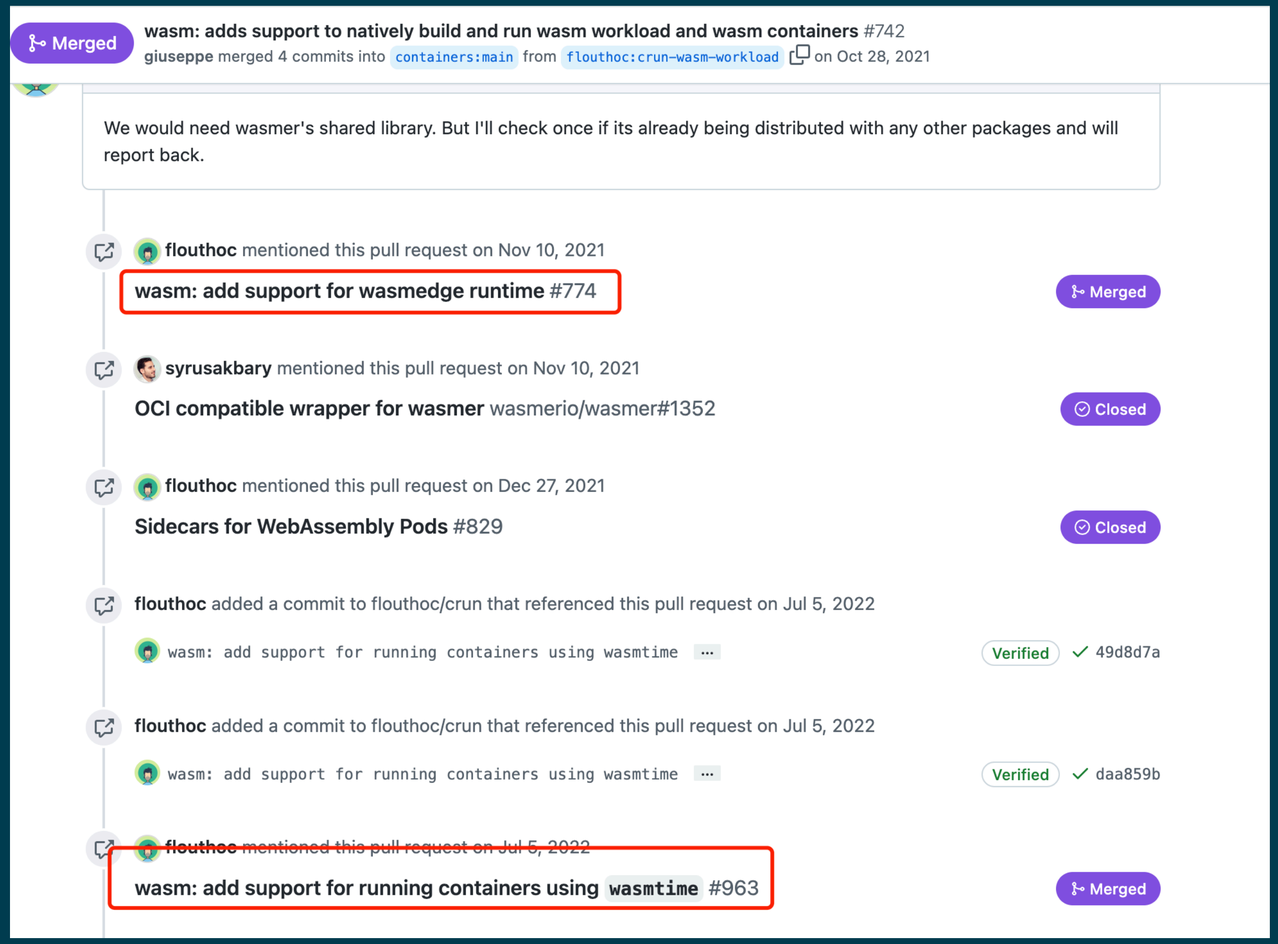
在CRun中运行WASM Workload

CRun为标准的OCI实现

Wasmer

Wasmtime

WasmEdge



在CRun中运行WASM Workload

Containerd配置 runtime修改为Crun

1

Build WASM 略

2

Build Image

安装Crun

```
git clone https://github.com/containers/crun
cd crun
./autogen.sh
./configure --with-wasmedge
make
sudo make install
```

Dockerfile

```
FROM scratch
ADD wasi_example_main.wasm /
CMD ["/wasi_example_main.wasm"]
```

```
buildah build --annotation "module.wasm.image/variant=compat-smart" -t wasm-wasi-example .
```

镜像必须指定annotation

3

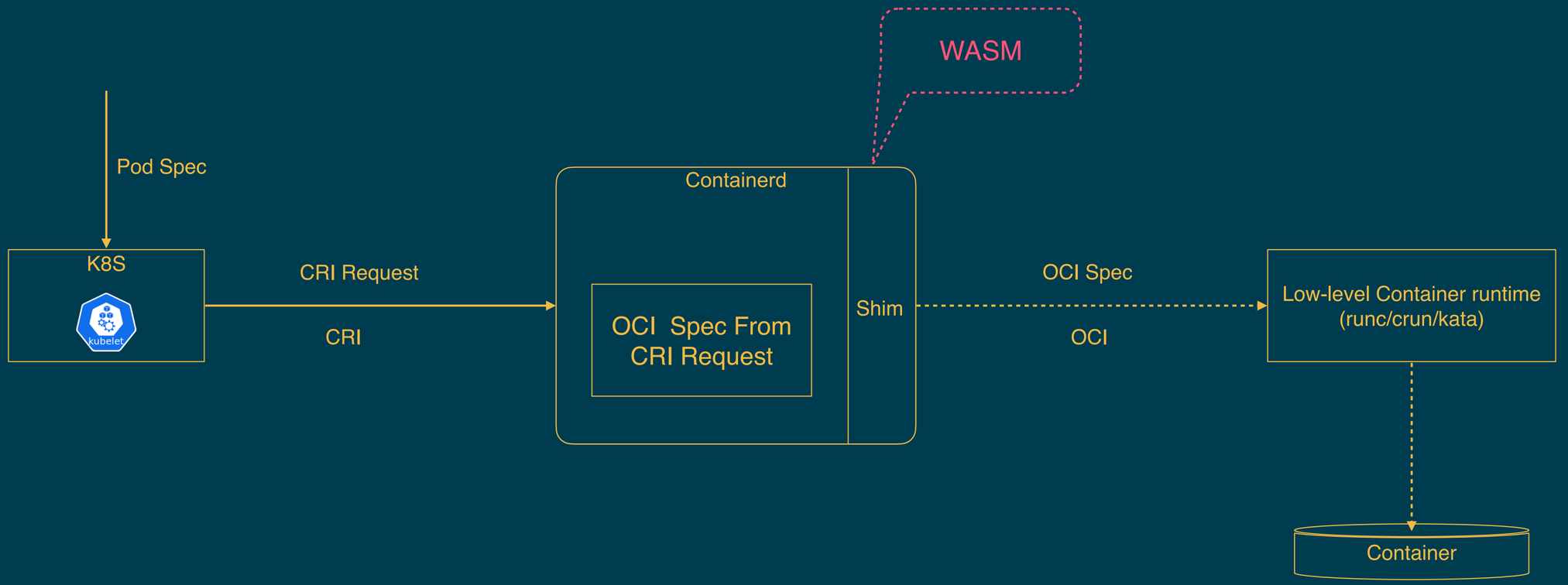
Start Container

```
ctr run --rm --runc-binary crun  --runtime io.containerd.runc.v2 \
--label module.wasm.image/variant=compat-smart \
docker.io/duizhang/example-wasi:latest wasm-example2 \
/wasi_example_main.wasm 50000000
```

运行结果

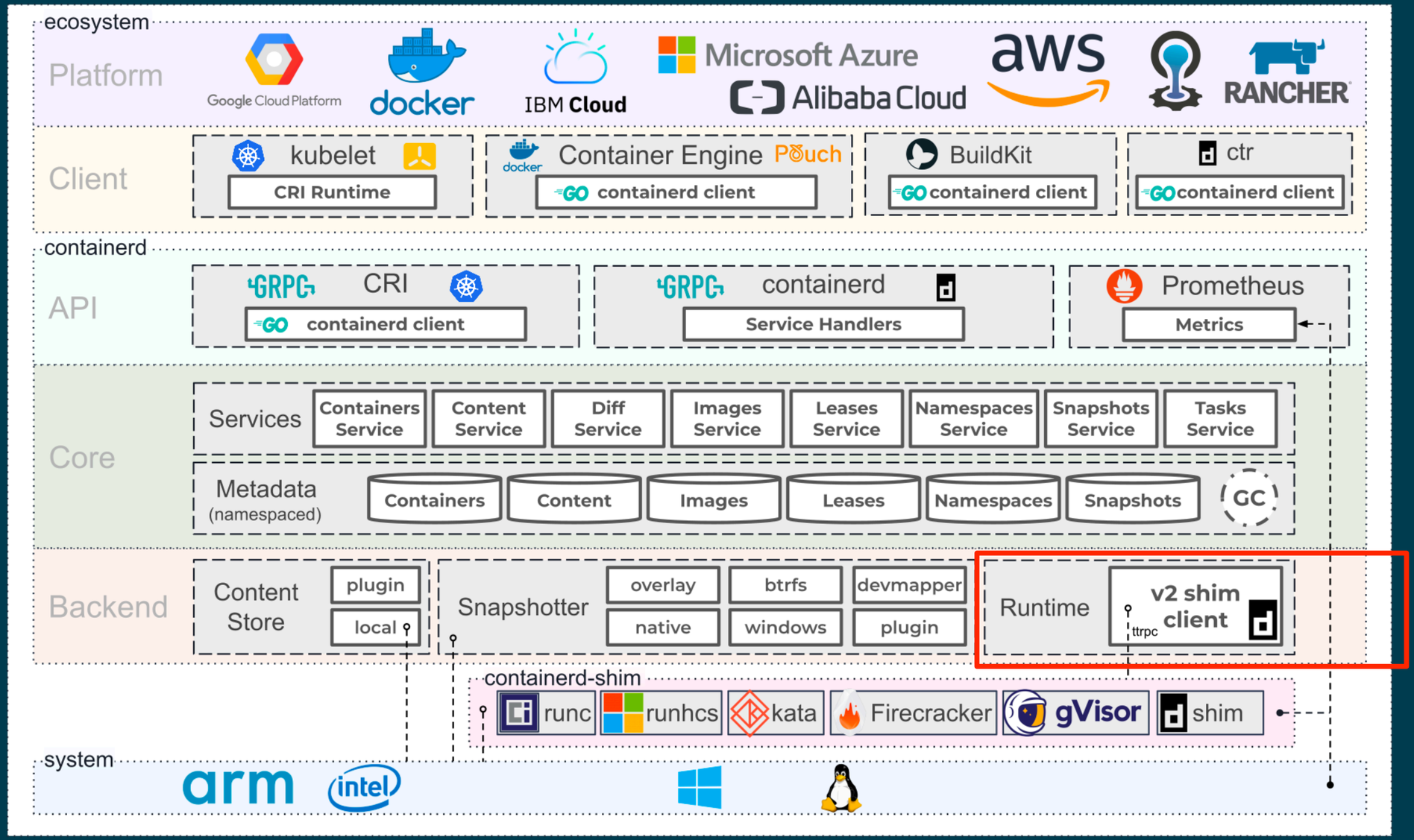
```
Random number: -279085236
Random bytes: [238, 254, 79, 160, 200, 217, 30, 62, 107, 211, 142,
78, 13, 245, 109, 216, 160, 217, 136, 226, 231, 31, 219, 115, 112,
73, 114, 141, 79, 2, 254, 44, 229, 110, 190, 9, 70, 106, 112, 6,
120, 32, 214, 50, 32, 162, 32, 20, 39, 68, 179, 15, 190, 113, 102,
60, 36, 212, 129, 147, 80, 178, 134, 80, 153, 126, 214, 108, 31, 61,
187, 4, 243, 89, 11, 226, 47, 165, 188, 84, 113, 86, 199, 166, 241,
70, 203, 160, 161, 125, 47, 173, 220, 219, 170, 217, 36, 52, 212,
177, 140, 212, 146, 251, 143, 67, 49, 36, 190, 34, 159, 21, 185,
102, 82, 182, 69, 150, 223, 2, 76, 94, 185, 74, 218, 16, 127, 226]
Printed from wasi: This is from a main function
This is from a main function
The env vars are as follows.
PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOME: /
The args are as follows.
/wasi_example_main.wasm
50000000
File content is This is in a file
```

在High-level Container Runtime中支持WASM（Containerd-Shim）



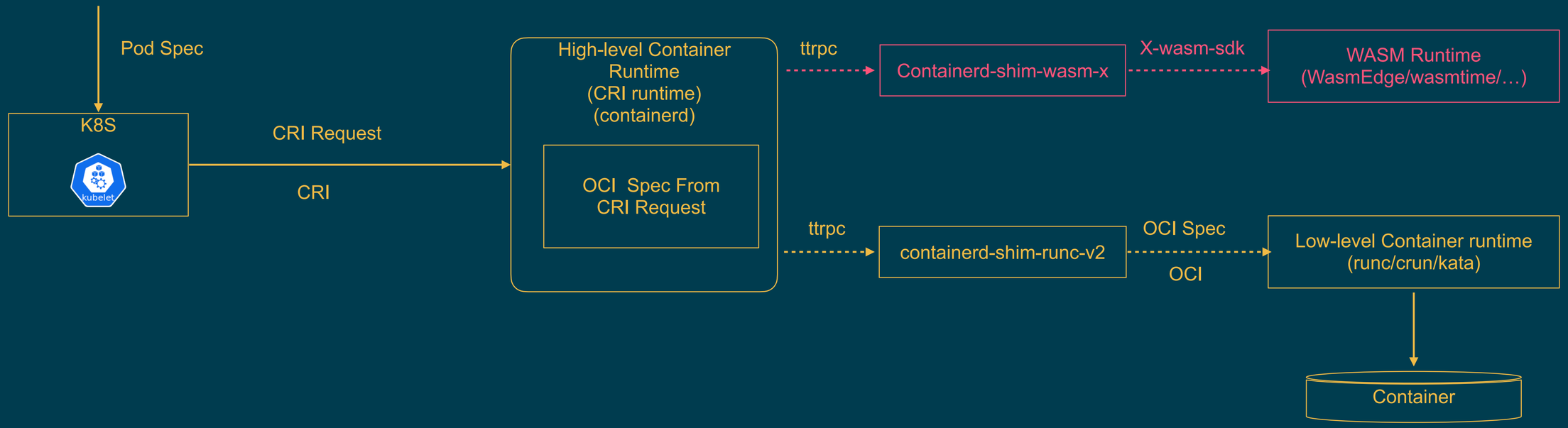
通过新增contianerd-shim支持WASM workload

Contained



Containerd整体架构

Containerd创建WASM Workload流程



通过新增containerd-shim支持WASM Workload

Containerd Shim 接口定义

Shim Server Interface(Golang)

```
type Shim interface {
    shimapi.TaskService
    Cleanup(ctx context.Context) (*shimapi.DeleteResponse, error)
    StartShim(ctx context.Context, opts StartOpts) (string, error)
}
```

```
type TaskService interface {
    State(context.Context, *StateRequest) (*StateResponse, error)
    Create(context.Context, *CreateTaskRequest) (*CreateTaskResponse, error)
    Start(context.Context, *StartRequest) (*StartResponse, error)
    Delete(context.Context, *DeleteRequest) (*DeleteResponse, error)
    Pids(context.Context, *PidsRequest) (*PidsResponse, error)
    Pause(context.Context, *PauseRequest) (*emptypb.Empty, error)
    Resume(context.Context, *ResumeRequest) (*emptypb.Empty, error)
    Checkpoint(context.Context, *CheckpointTaskRequest) (*emptypb.Empty, error)
    Kill(context.Context, *KillRequest) (*emptypb.Empty, error)
    Exec(context.Context, *ExecProcessRequest) (*emptypb.Empty, error)
    ResizePty(context.Context, *ResizePtyRequest) (*emptypb.Empty, error)
    CloseIO(context.Context, *CloseIORequest) (*emptypb.Empty, error)
    Update(context.Context, *UpdateTaskRequest) (*emptypb.Empty, error)
    Wait(context.Context, *WaitRequest) (*WaitResponse, error)
    Stats(context.Context, *StatsRequest) (*StatsResponse, error)
    Connect(context.Context, *ConnectRequest) (*ConnectResponse, error)
    Shutdown(context.Context, *ShutdownRequest) (*emptypb.Empty, error)
}
```

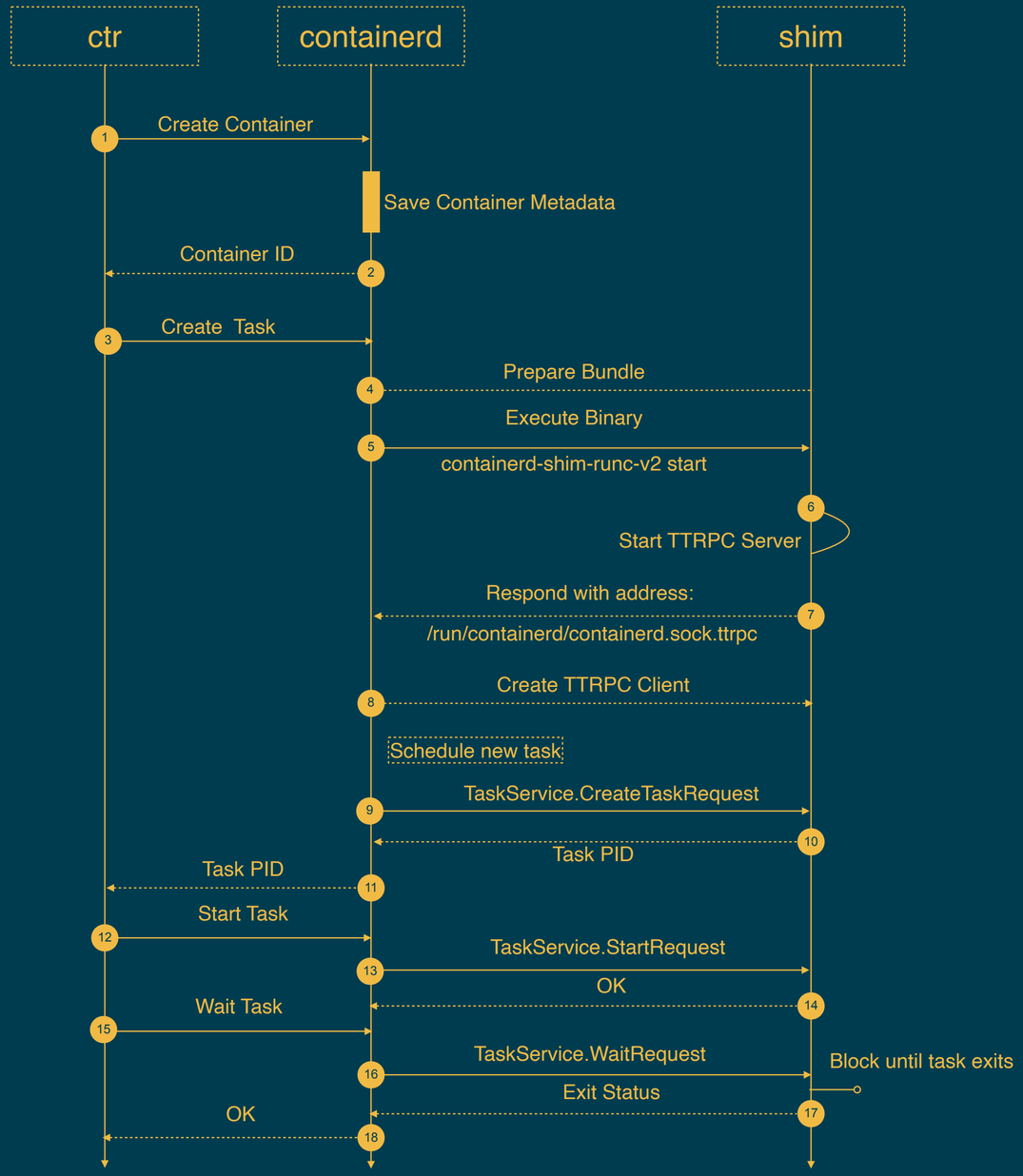
Containerd内置Shim

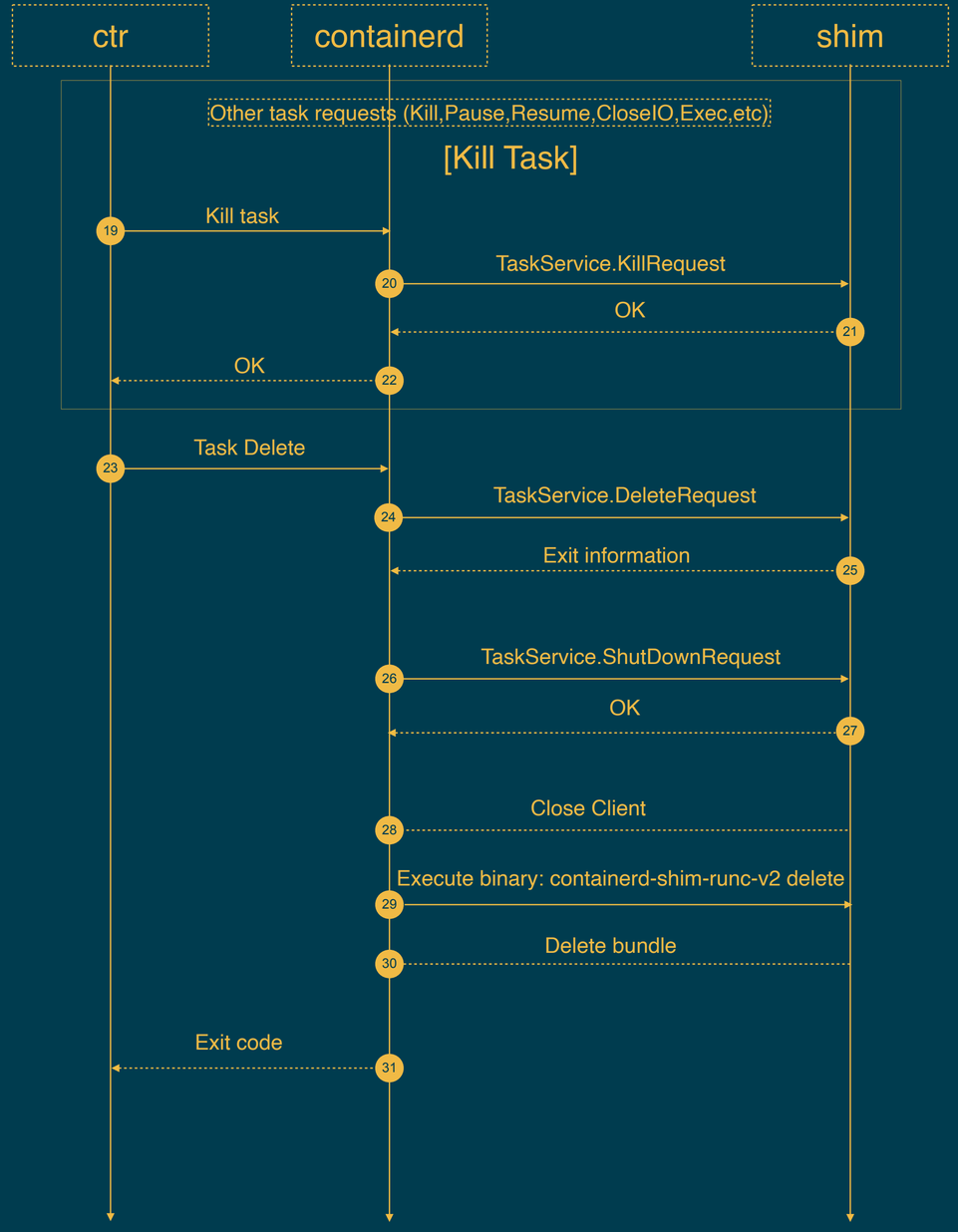
```
io.containerd.runc.v1
io.containerd.runc.v2
io.containerd.runhcs.v1(for windows)
```

三方Shim

```
containerd-shim-kata-v2(kata-containers)
containerd-shim-runsc-v1(gVisor)
......
```

Containerd Shim Flow





Containerd runwasi

runwasi(rust)

```rust
pub trait Instance {
    // Create a new instance
    fn new(id: String, cfg: Option<&InstanceConfig<Self::E>>) -> Self;
    // Start the instance and return the pid
    fn start(&self) -> Result<u32, Error>;
    // Send the specified signal to the instance
    fn kill(&self, signal: u32) -> Result<(), Error>;
    // Delete the instance
    fn delete(&self) -> Result<(), Error>;
    // wait for the instance to exit and send the exit code and exit timestamp to the provided sender.
    fn wait(&self, send: Sender<(u32, DateTime<Utc>)>) -> Result<(), Error>;
}
```

https://github.com/containerd/runwasi

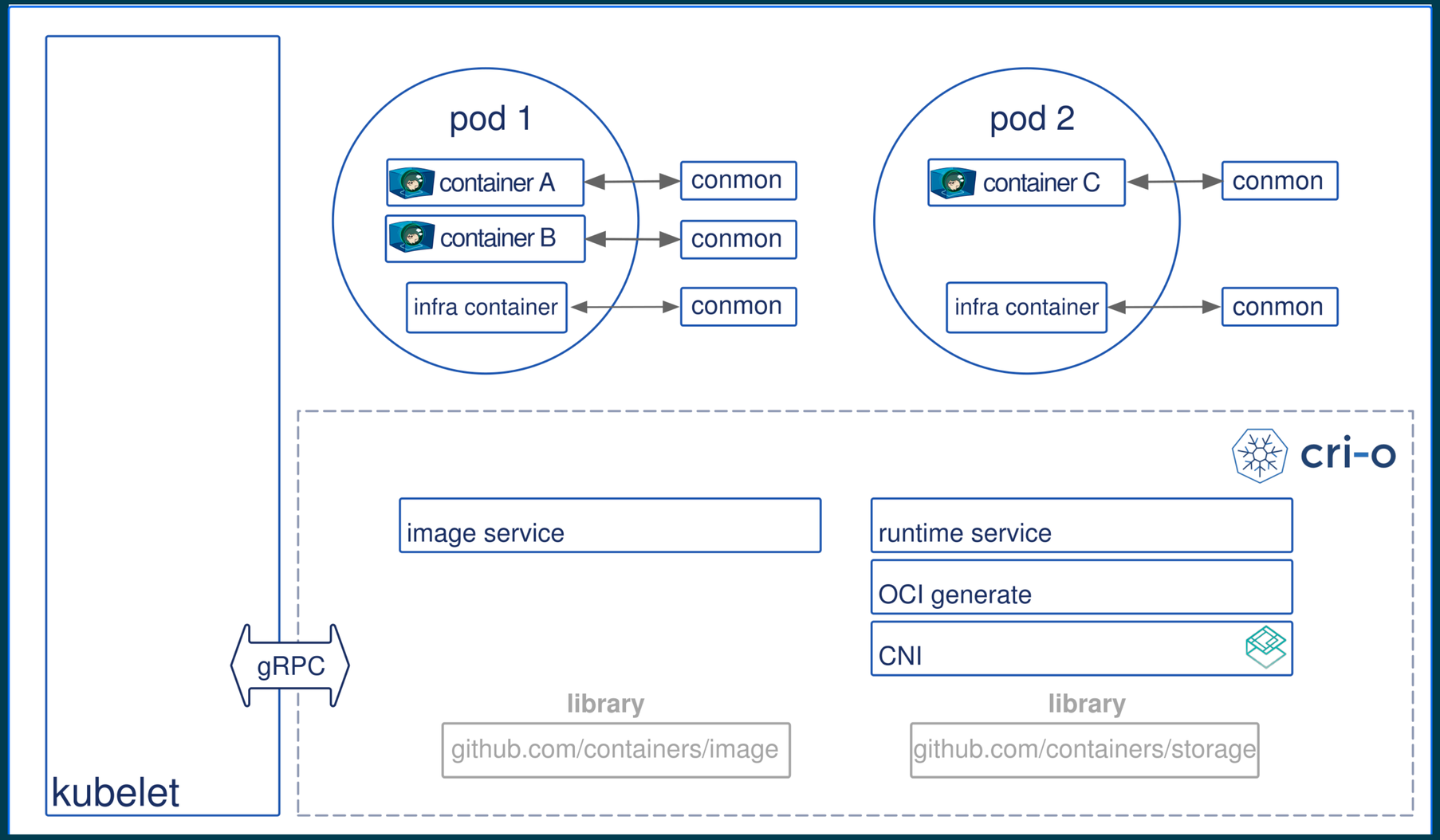
Containerd Shim实例—wasmtime

以containerd-shim-wasmtime-v1为例

```
ctr run --rm --runtime=io.containerd.wasmtime.v1 \
ghcr.io/containerd/runwasi/wasi-demo-app:latest \
testwasm2 /wasi-demo-app.wasm echo 'hello'
```

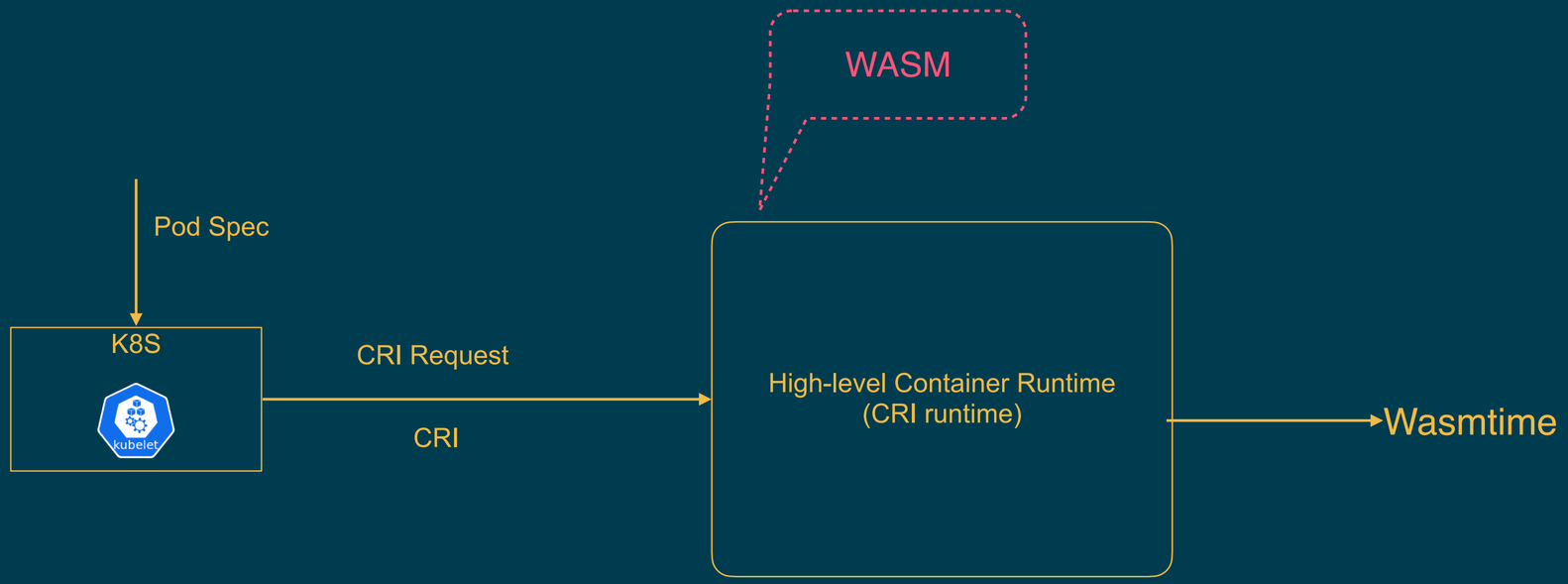
https://github.com/containerd/runwasi

CRI-O



每一个container都由一个单独的conmon进程管理

实现新的CRI组件支持WASM Workload



通过实现新的CRI支持WASM Workload

OCI Spec

WASMRun为例

1 kubelet配置中指定runtime为wasmrun

```
KUBELET_KUBEADM_ARGS="--container-runtime-endpoint=unix:///var/run/cri-wasm.sock --pod-infra-container-image=registry.k8s.io/pause:3.9"
```

2 查看节点的CONTAINER-RUNTIME

```
zwliu@liuzhenweideMacBook-Pro ~ % kubectl get node -o wide
NAME             STATUS     ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
vm-0-10-ubuntu   NotReady   <none>          70d   v1.26.3   172.19.0.10    <none>        Ubuntu 20.04.6 LTS   5.4.0-144-generic   wasmrun://0.1.0
vm-0-11-ubuntu   Ready      control-plane   74d   v1.26.0   172.19.0.11    <none>        Ubuntu 20.04 LTS     5.4.0-126-generic   containerd://1.6.8
```

3 创建测试用Job,并查看Pod日志

```
zwliu@liuzhenweideMacBook-Pro KubeFunction % kubectl get pod
NAME                          READY   STATUS      RESTARTS   AGE
wasmrun-job-mg4h6             0/1     Completed   0          6s
wasmtime-5f5645c98c-2gqwz     1/1     Running     0          15h
zwliu@liuzhenweideMacBook-Pro KubeFunction % kubectl logs -f wasmrun-job-mg4h6
Hello, duizhangwww!
```

```
go run main.go -wasm-runtime-endpoint unix:///var/run/cri-wasm.sock \
    -container-runtime-socket=/run/containerd/containerd.sock \
    -v 3
```

wasmrun运行参数

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: "wasmrun"
handler: "wasmrun"
scheduling:
  nodeSelector:
    "app": "wasmrun"
```

runtimeClass定义

```
apiVersion: batch/v1
kind: Job
metadata:
  name: wasmrun-job
spec:
  template:
    metadata:
      labels:
        app: wasmrun
    spec:
      runtimeClassName: wasmrun
      restartPolicy: Never
      containers:
        - name: testwasm
          image: docker.io/duizhang/hello-wasm:9.0
          command: ["greet"]
          args: ["duizhangwww"]
```

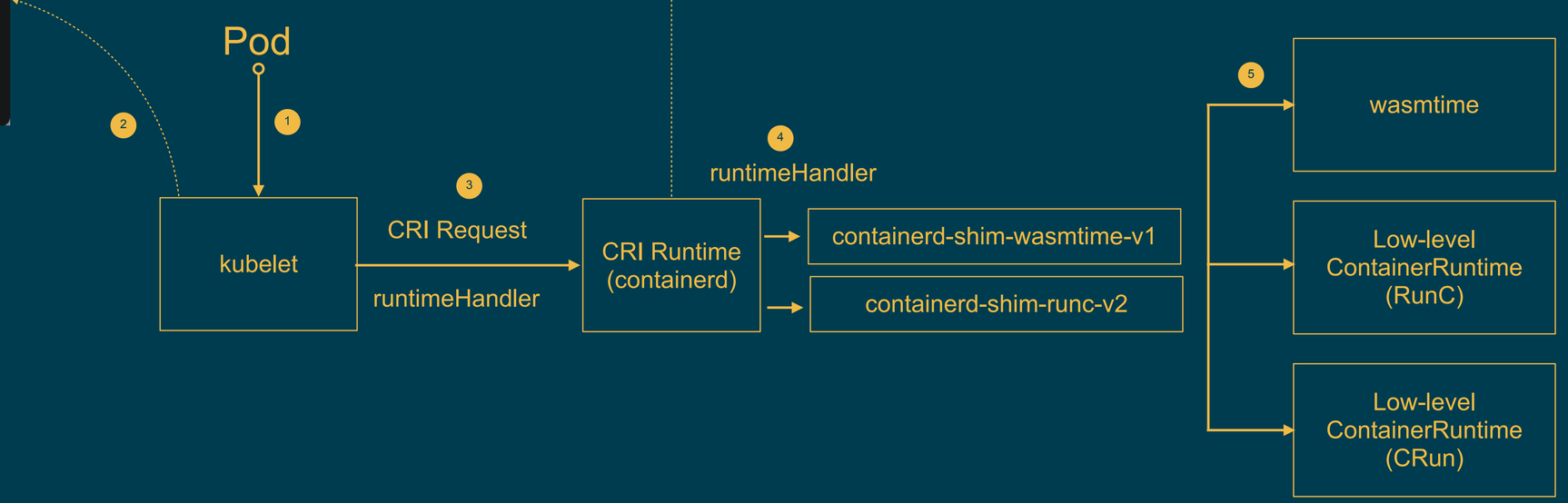
测试用Job

Kubernetes多运行时整体流程

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.wasmtime]
    runtime_type = "io.containerd.wasmtime.v1"
```

containerd配置支持wasmtime

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: "wasmtime"
handler: "wasmtime"
scheduling:
  nodeSelector:
    "wasm": "wasmtime"
```




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wasmtime
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wasmtime
  template:
    metadata:
      labels:
        app: wasmtime
    spec:
      runtimeClassName: wasmtime
      containers:
      - name: wasm-hello
        image: docker.io/duizhang/wasi-demo-app:latest
        command: ["/wasi-demo-app.wasm","echo","duizhang"
```

测试用工作负载

```
zwliu@liuzhenweideMacBook-Pro KubeFunction % kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
wasmtime-5f5645c98c-2gqwz   1/1     Running   0          9m45s
zwliu@liuzhenweideMacBook-Pro KubeFunction % kubectl logs -f wasmtime-5f5645c98c-2gqwz
duizhang
exiting
```

结果输出

定义新的kubelet以支持WASM Workload

krustlet

```
kubectl get nodes -o wide
NAME       STATUS   ROLES    AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE               KERNEL-VERSION   CONTAINER-RUNTIME
minikube   Ready    master   18m   v1.18.0   192.168.99.165   <none>        Buildroot 2019.02.10   4.19.107         docker://19.3.8
krustlet   Ready    agent    9s    v1.17.0   10.0.2.2         <none>        <unknown>              <unknown>        mvp
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-wasm
spec:
  containers:
  - name: hello-wasm
    image: webassembly.azurecr.io/hello-wasm:v1
  tolerations:
  - effect: NoExecute
    key: kubernetes.io/arch
    operator: Equal
    value: wasm32-wasi   # or wasm32-wasmcloud according to module target arch
  - effect: NoSchedule
    key: kubernetes.io/arch
    operator: Equa
    value: wasm32-wasi   # or wasm32-wasmcloud according to module target arch
```

谢谢

姓名 刘振伟
职位
email@thoughtworks.com

/thoughtworks