HPS

Jule Anger, supervisor Jonathan Decker

Comparison of various runtimes in Kubernetes

Project Preliminary Results

Table of contents

Outline

Kubernetes

- Does Orchestration of containers and storage
- Scales applications automatically
- Monitors the containers actively (i.e. restarts crashed applications)
- Does load balancing

Source: John Arundel and Justin Domingus, *Cloud Native DevOps mit Kubernetes*

Structure of Kubernetes

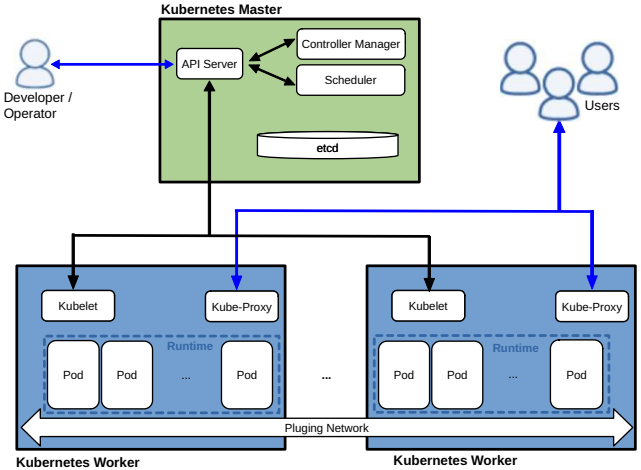


Image source: https://commons.wikimedia.org/wiki/File:Kubernetes.png (Accessed on 2023-11-25, changed)

Tasks of a container runtime

A container runtime...

- Manages the entire container life cycle
- Loads container images from a repository
- Monitors different local system resources
- Isolates system resources for use of a container

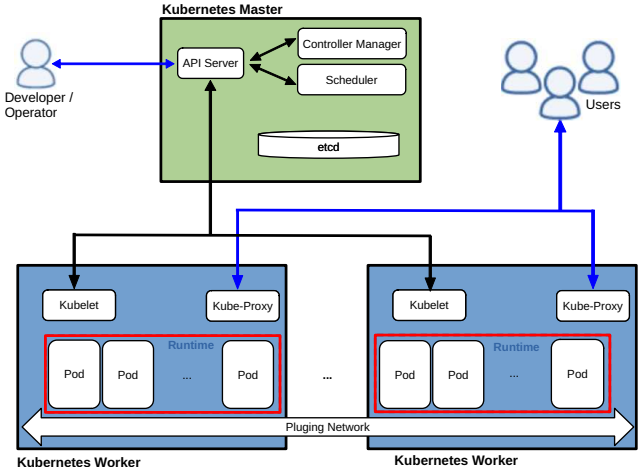


Image source: https://commons.wikimedia.org/wiki/File:Kubernetes.png (Accessed on 2023-11-25, changed)

Kubernetes Terminology

- Pod: contains one or more containers and volumes
- Deployment: is an object used to manage pods (and other objects)

John Arundel and Justin Domingus, *Cloud Native DevOps mit Kubernetes*

Outline

Types of runtimes

- Many different runtimes available
- Grouped into Low-Level and High-Level runtimes

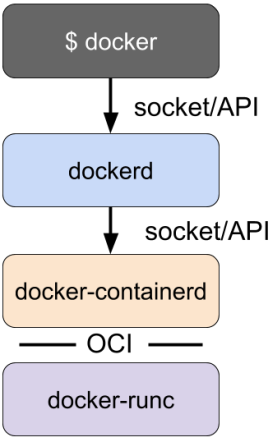
Source: Lewis, *Container Runtimes Part 1: An Introduction to Container Runtimes*

Low-level runtimes

- Limited Functionality: start and manage containers
- Examples: RunC, cRun, LXC

Source: Rentrop, *Low-Level Container Runtimes*; Lewis, *Container Runtimes Part 1: An Introduction to Container Runtimes*

High-Level runtimes



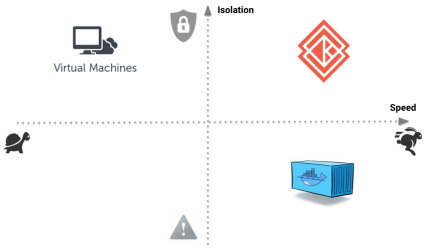
- More Functionality: image management, building, packaging, sharing, and running containers
- Runs on top of low-level or other high-level runtimes (each runtime delegates the more low-level one)
- Examples: Docker, containerd, CRI-O

Source & Image source: Lewis, *Container Runtimes Part 3: High-Level Runtimes*

Uncommon Container Runtimes

- This presentation focuses on some less common runtimes, namely
  - Kata Containers (low-level)
  - Firecracker (low-level)
  - gVisor (low-level)

Runtime Kata Containers

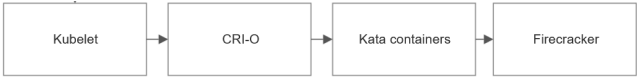


- Runs with containerd or CRI-O
- Uses a combination of Virtual Machines and Containers
- Has been developed by a community since 2017
- Is published as open source (https://github.com/kata-containers/, around 4 000 pull requests)
- Is written in Rust & Go

Source: https://katacontainers.io/community/
Source & Image Source: https://www.katacontainers.io/collateral/kata-containers-onboarding-deck.pptx

Runtime Firecracker

- Runs with CRI-O or containerd and Kata
- Also uses a combination of Virtual Machines and Containers: "MicroVMs"
- Has been developed by Amazon and a communitysince 2014
- Is published as open source (`https://github.com/firecracker-microvm/firecracker`, around 3 000 pull requests)
- Is written in Rust & Python



Source: F. Community, *Firecracker FAQs*
Image source: Foy, *How to deploy Kubernetes with Firecracker?*

Runtime gVisor

- Runs with containerd
- Uses nested virtualization (running containers in virtual machines)
- Has been developed by Google and a community since 2019
- Is published as open source (`https://github.com/google/gvisor`, around 8.000 pull requests)
- Is written in Go & C++

Source: `https://gvisor.dev/`

Outline

Setup: Kubernetes cluster

- Build on OpenStack by the GWDG (`cloud.gwdg.de`)
- Consists of one master node and four worker nodes

```
cloud@master:~$ kubectl get nodes
NAME            STATUS   ROLES           AGE   VERSION
master          Ready    control-plane   31d   v1.28.2
worker-contd    Ready    <none>          16d   v1.28.2
worker-fire     Ready    <none>          15d   v1.28.2
worker-gvisor   Ready    <none>          10d   v1.28.2
worker-kata     Ready    <none>          25d   v1.28.2
```

Setup: Kubernetes cluster

```
cloud@worker-kata:~$ sudo ctr --namespace k8s.io containers ls |  cut -d\
IMAGE                                    RUNTIME
registry.k8s.io/pause:3.6                io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-npc:latest    io.containerd.runtime.v1.linux
registry.k8s.io/pause:3.6                io.containerd.kata.v2
registry.k8s.io/pause:3.6                io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-kube:latest   io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-npc:latest    io.containerd.runtime.v1.linux
registry.k8s.io/pause:3.6                io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-kube:latest   io.containerd.runtime.v1.linux
docker.io/library/nginx:latest           io.containerd.kata.v2
registry.k8s.io/pause:3.6                io.containerd.runtime.v1.linux
registry.k8s.io/kube-proxy:v1.28.4       io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-kube:latest   io.containerd.runtime.v1.linux
registry.k8s.io/kube-proxy:v1.28.4       io.containerd.runtime.v1.linux
```

Setup: Kubernetes cluster

```
cloud@worker-gvisor:~$ sudo ctr --namespace k8s.io containers ls |  cut -d\
IMAGE                                RUNTIME
registry.k8s.io/pause:3.6            io.containerd.runc.v2
docker.io/weaveworks/weave-npc:latest    io.containerd.runc.v2
docker.io/weaveworks/weave-npc:latest    io.containerd.runtime.v1.linux
registry.k8s.io/pause:3.6            io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-kube:latest   io.containerd.runtime.v1.linux
registry.k8s.io/pause:3.6            io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-kube:latest   io.containerd.runc.v2
docker.io/library/nginx:latest       io.containerd.runsc.v1
registry.k8s.io/kube-proxy:v1.28.4   io.containerd.runtime.v1.linux
registry.k8s.io/pause:3.6            io.containerd.runc.v2
registry.k8s.io/kube-proxy:v1.28.4   io.containerd.runc.v2
registry.k8s.io/pause:3.6            io.containerd.runtime.v1.linux
docker.io/weaveworks/weave-kube:latest   io.containerd.runtime.v1.linux
registry.k8s.io/pause:3.6            io.containerd.runsc.v1
```

Example deployment

yaml

```yaml
apiVersion: apps/v1
kind: Deployment
[...]
spec:
  replicas: 3
  template:
    [...]
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
      runtimeClassName: kata
      nodeSelector:
              nodetype: kata
```

Testing tool

- The tool used is called clusterloader2
- Has been published on the Kubernetes GitHub page
- Can test the following aspects:
  - Availability of cluster's control plane
  - Reaction of the cluster to failed nodes
  - Time required to create objects (e.g. deployments and pods)
  - ...

Source: https://github.com/kubernetes/perf-tests/tree/master/clusterloader2

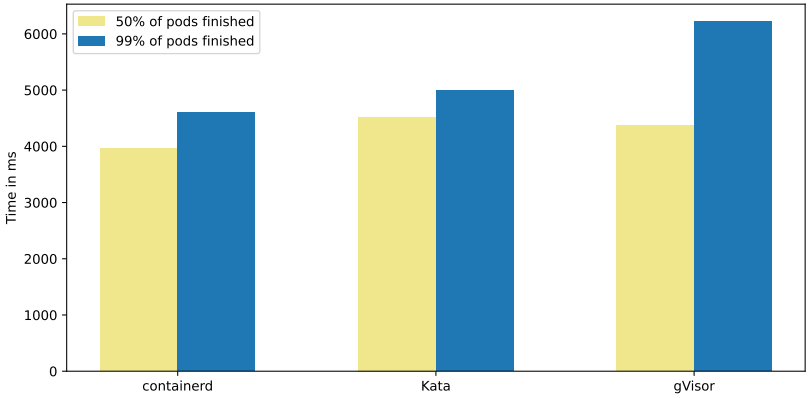
Testing tool

Demo

Some notes about the performance tests

- The tool provides various metrics
- Here the time between *Scheduled* and *Running* is used
- The application start time is not considered
- The tool indicates the time until 50%, 90% and 99% of the pods have fullfilled the desired state
- The tool gives the average times of all pods under consideration
- I ran each test several times and used the average value

Performance tests with a deployment with 10 Pods



Outline

Installation

- containerd: simple (default)
- Docker, CRI-O: relative simple (common)
- Kata: bit complicated (needs common high level runtime and kata)
- Firecracker: more complicated (needs common high level runtime and kata and own storage partition)
- gVisor: very simple in Google Kubernetes, otherwise a bit complicated

Source: K. C. Community, *How to use Kata Containers and Containerd*, F. M. Community, *Getting Started with Firecracker*, Community, *Containerd Quick Start*

Security

- containerd: normal
- Kata: high
- Firecracker: high
- gVisor: high

Source: Kienzler, *Welcome To The Container Jungle: Docker vs. containerd vs. Nabla vs. Kata vs. Firecracker and more!*

Performance

- containerd: normal
- Kata: around 8% higher than containerd
- Firecracker: ?
- gVisor: around 35% higher than containerd

Outline

Further work

- Firecracker
  - Finish installation
  - Run performance tests

Further work

- Firecracker
  - Finish installation
  - Run performance tests
- Performance tests
  - Test more object types
  - Run with higher number of pods
  - Maybe use a second tool

Further work

- Firecracker
  - Finish installation
  - Run performance tests
- Performance tests
  - Test more object types
  - Run with higher number of pods
  - Maybe use a second tool
- More research
  - Security improvements
  - Internal functioning
  - Resource consumption

(Preliminary) Results

- Containerd is the easiest to install
- Kata, Firecracker and gVisor have increased security precautions
- Kata is 8% slower than containerd, gVisor 35%

References I

Community, Firecracker. *Firecracker FAQs*. https://firecracker-microvm.github.io/. Accessed: 2024-01-09.

Community, Firecracker MikroVM. *Getting Started with Firecracker*. https://github.com/firecracker-microvm/firecracker/blob/main/docs/getting-started.md. Accessed: 2023-12-29.

Community, gVisor. *Containerd Quick Start*. https://gvisor.dev/docs/user_guide/containerd/quick_start/. Accessed: 2024-01-05.

Community, Kata Containers. *How to use Kata Containers and Containerd*. https://github.com/kata-containers/kata-containers/blob/main/docs/how-to/containerd-kata.md. Accessed: 2023-12-18.

Foy, Baptiste. *How to deploy Kubernetes with Firecracker?* https://www.padok.fr/en/blog/deploy-kubernetes-firecracker. Accessed: 2024-01-12.

John Arundel and Justin Domingus. *Cloud Native DevOps mit Kubernetes*. dpunkt.verlag, 2019.

Kienzler, Simon. *Welcome To The Container Jungle: Docker vs. containerd vs. Nabla vs. Kata vs. Firecracker and more!* https://www.inovex.de/de/blog/containers-docker-containerd-nabla-kata-firecracker/. Accessed: 2023-11-28.

References II

Lewis, Ian. *Container Runtimes Part 1: An Introduction to Container Runtimes*. https://www.ianlewis.org/en/container-runtimes-part-1-introduction-container-r. Accessed: 2024-01-12.

— .*Container Runtimes Part 3: High-Level Runtimes*. https://www.ianlewis.org/en/container-runtimes-part-3-high-level-runtimes. Accessed: 2024-01-12.

Rentrop, Christian. *Low-Level Container Runtimes*. https://www.dev-insider.de/low-level-container-runtimes-a-7c2c3ce41340e09d0ffc94e705a6f965/. Accessed: 2024-01-10.

Performance tests with a deployment with 10 Pods

Node	Runtime	Time (50%)	Time (99%)
worker-contd	containerd	4161 ms	4628 ms
worker-kata	containerd	3780 ms	4875 ms
	kata	4525 ms	4977 ms
worker-gvisor	containerd	3958 ms	4289 ms
	gvisor (runsc)	4373 ms	6218 ms