使用kuasar运行wasm

quick start

- 具体内容参考：https://github.com/kuasar-io/kuasar#quick-start

1. OS

The minimum versions of Linux distributions supported by Kuasar are *Ubuntu 22.04* or *CentOS 8* or openEuler 23.03.

Please also note that Quark requires a Linux kernel version >= 5.15.

2. Sandbox

- MicroVM: To launch a microVM-based sandbox, a hypervisor must be installed on the host.
  - It is recommended to install Cloud Hypervisor by default. You can find Cloud Hypervisor installation instructions here.
  - If you want to run kuasar with iSulad container engine and StratoVirt hypervisor, you can refer to this guide how-to-run-kuasar-with-isulad-and-stratovirt.
- Quark: To use Quark, please refer to the installation instructions here.
- WasmEdge: To start WebAssembly sandboxes, you need to install WasmEdge v0.11.2. Instructions for installing WasmEdge can be found in install.html.

- 安装wasmedge：https://wasmedge.org/book/en/quick_start/install.html
- （目前kuasar只支持wasmedge）

使用kuasar运行wasm

quick start

3. containerd

Kuasar sandboxers are external plugins of containerd, so both containerd and its CRI plugin are required in order to manage the sandboxes and containers.

We offer two ways to interact Kuasar with containerd:

- **EXPERIMENTAL in containerd 2.0 milestone**: If you desire the full experience of Kuasar, please install containerd under kuasar-io organization. Rest assured that our containerd is built based on the official v1.7.0, so there is no need to worry about missing any functionalities.
- If the compatibility is a real concern, you need to install official containerd v1.7.0 with an extra kuasar-shim for request forwarding, see here. However, it's possible that this way may be deprecated in the future as containerd 2.0 evolves.

- 安装&配置containerd参考：https://github.com/kuasar-io/kuasar/blob/main/docs/containerd.md
- **注意**：配置/etc/containerd/config.toml之前，一定要用**kuasar的containerd生成的配置文件**（它与原containerd生成的配置文件不一样，会有一些额外的字段）
  - 生成配置文件命令：containerd config default > /etc/containerd/config.toml
  - 否则会报错：

```
root@VM-0-11-ubuntu:~/kuasar# bash examples/run_example_wasm_container.sh
info: component 'rust-std' for target 'wasm32-wasi' is up to date
RepoTags: ghcr.io/containerd/runwasi/wasi-demo-app:latest
E0509 12:20:07.546792    6563 remote_runtime.go:176] "RunPodSandbox from runtime service failed" err="rpc error: code = Unknown desc = failed to creat
e containerd task: failed to start shim: failed to resolve runtime path: runtime \"io.containerd.wasm.v1\" binary not installed \"containerd-shim-wasm
-v1\": file does not exist: unknown"
FATA[0000] running container: run pod sandbox: rpc error: code = Unknown desc = failed to create containerd task: failed to start shim: failed to reso
lve runtime path: runtime "io.containerd.wasm.v1" binary not installed "containerd-shim-wasm-v1": file does not exist: unknown
```

使用kuasar运行wasm

quick start

4. crictl

Since Kuasar is built on top of the Sandbox API, which has already been integrated into the CRI of containerd, it makes sense to experience Kuasar from the CRI level.

- `crictl` is a debug CLI for CRI. To install it, please see here

- 安装crictl参考： https://github.com/kubernetes-sigs/cri-tools/blob/master/docs/crictl.md#install-crictl

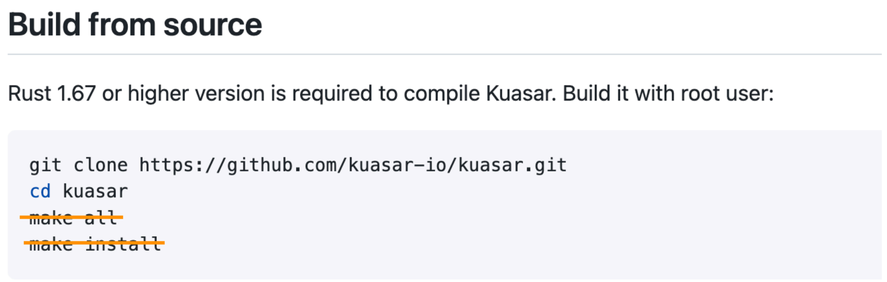
5. virtiofsd

MicroVMs like Cloud Hypervisor needs a virtiofs daemon to share the directories on the host. Please refer to virtiofsd guide.

- 这一步对于wasm来说不需要

使用kuasar运行wasm

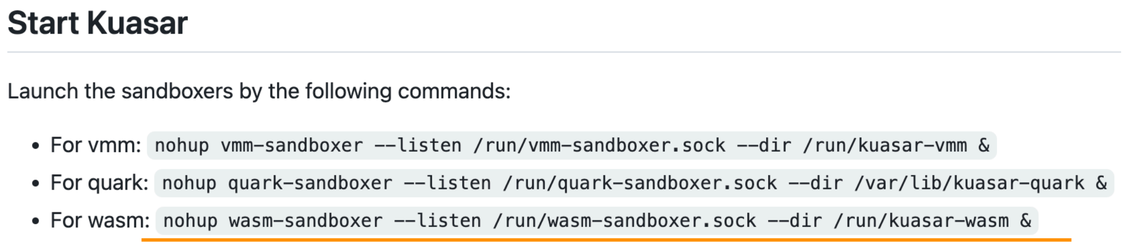
quick start

- 

  - 我们不用全部构建，只需要构建wasm-sandboxer，运行下列命令：
    - make wasm
    - make install-wasm

使用kuasar运行wasm

quick start

- 

  - 启动wasm-sandboxer
  - 然后启动containerd：
    - ENABLE_CRI_SANDBOXES=1 containerd
    - ENABLE_CRI_SANDBOXES=1是为了使用containerd的sandbox api

使用kuasar运行wasm

quick start

Start Container

Since Kuasar is a low-level container runtime, all interactions should be done via CRI in containerd, such as crictl or Kubernetes. We use crictl as examples:

- For vmm and quark, run the following scripts:

  `examples/run_example_container.sh vmm` or `examples/run_example_container.sh quark`

- For wasm: Wasm container needs its own container image so our script has to build and import the container image at first.

- `examples/run_example_wasm_container.sh`

```
root@VM-0-11-ubuntu:~/kuasar# tail /tmp/wasm.log
2023-05-09T13:59:59.981265647+08:00 stdout F This is a song that never ends.
2023-05-09T13:59:59.981291019+08:00 stdout F Yes, it goes on and on my friends.
2023-05-09T13:59:59.981295134+08:00 stdout F Some people started singing it not kn
2023-05-09T13:59:59.981298008+08:00 stdout F So they'll continue singing it forever
2023-05-09T13:59:59.98130074+08:00 stdout F
2023-05-09T14:00:00.981396004+08:00 stdout F This is a song that never ends.
2023-05-09T14:00:00.981425618+08:00 stdout F Yes, it goes on and on my friends.
2023-05-09T14:00:00.981430072+08:00 stdout F Some people started singing it not kn
2023-05-09T14:00:00.981433295+08:00 stdout F So they'll continue singing it forever
2023-05-09T14:00:00.98143607+08:00 stdout F
```

- 运行wasm示例：example/run_example_wasm_container.sh

```
cat > container.json <<EOF
{
    "metadata": {
        "name": "wasm",
        "namespace": "default"
    },
    "image": {
      "image": "ghcr.io/containerd/runwasi/wasi-demo-app:latest"
    },
    "log_path":"wasm.log",
    "linux": {
        "security_context": {
            "namespace_options": {
                "network": 2,
                "pid": 1
```

```
root@VM-0-11-ubuntu:~/kuasar# bash examples/run_example_wasm_container.sh
info: component 'rust-std' for target 'wasm32-wasi' is up to date
RepoTags: ghcr.io/containerd/runwasi/wasi-demo-app:latest
INFO[2023-05-09T13:59:37.894368918+08:00] RunPodSandbox for &PodSandboxMetadata{Name:test-sandbox1683611977,Uid:,Namespace:default,Attempt:0,}
INFO[2023-05-09T13:59:37.910113483+08:00] RunPodSandbox for &PodSandboxMetadata{Name:test-sandbox1683611977,Uid:,Namespace:default,Attempt:0,} returns sandbox id "2c8317db8
f121643eb8f4d23fea0450f687fcff3b5575ae96bdd3a3e701"
INFO[2023-05-09T13:59:37.911378758+08:00] CreateContainer within sandbox "2c8317db8f13c4f121643eb8f4d23fea0450f687fcff3b5575ae96bdd3a3e701" for container &ContainerMetadata
:wasm,Attempt:0,}
INFO[2023-05-09T13:59:37.930137314+08:00] CreateContainer within sandbox "2c8317db8f13c4f121643eb8f4d23fea0450f687fcff3b5575ae96bdd3a3e701" for &ContainerMetadata{Name:wasm
mpt:0,} returns container id "51c970a9f2b4a8d238776d9d36c79ca9b4f0d72aa2e0cdfcd1a434a8412fd003"
INFO[2023-05-09T13:59:37.930660790+08:00] StartContainer for "51c970a9f2b4a8d238776d9d36c79ca9b4f0d72aa2e0cdfcd1a434a8412fd003"
INFO[2023-05-09T13:59:37.951769887+08:00] StartContainer for "51c970a9f2b4a8d238776d9d36c79ca9b4f0d72aa2e0cdfcd1a434a8412fd003" returns successfully
51c970a9f2b4a8d238776d9d36c79ca9b4f0d72aa2e0cdfcd1a434a8412fd003
```

使用kuasar继续性能优化的问题

- 镜像问题：kuasar使用的镜像是oci镜像（含wasm文件），仍然要通过registry来拉取；
- 删除pod功能有问题，不能正常删除；
- 要达成100ms的目标，用kuasar似乎还是差点；

后续工作方向

- 先用kuasar跑起来，测试性能；
- 如果性能不达标，可能要研究kuasar，并对其改造；