21th October 2020 - Meeting Summary

- We have 2 key CIs:
    - The openshift upstream CI which also includes the CRI-O part (no Jenkins, a different system)
    - The kata upstream CI Jenkins job
- **For the openshift CI Wainer already got the kata onboarded**
    - This CI runs daily and later on we want a PR to trigger this job as well
    - It uses the latest kata from the master branch (not 2.0)
    - It runs a simple smoke test on openshift 4.5
        - Install kata
        - Run a few tests
        - Uninstall kata
        - Note that a new cluster is provisioned every time (nested)
    - Wainer is working to run the openshift E2E conformance test on it
    - Wainer already created a list of improvements for this work
    - Critical point here is the E2E test - MVP
    - Snir is doing something similar for 2.0 (manually)
- **Wainer sent the requirements for the kata Jenkins CI**
    - There are many areas that would benefit from some improvements
    - Main tasks that need to be done:

EPIC: # R1. Close the gaps in kata upstream#

There are some configurations which Red Hat cares about not being tested.

From the business point of view strictly this is high priority among the other requirements listed here. However contributing changes to the CI is currently very difficult, so I think the items R2 and R3 below should be considered equally important.

EPIC: # R2. Ease the management of jobs #

Just a list of the problems that quickly came up to my mind:

- If I want to change a job because I, for example, need to export a new variable in the build script then I need to change the job XML found in [2]. Those XMLs aren't meant for humans to edit, they are just Jenkins internal representation of jobs. Once I changed the XML (actually several xmls, one per job), I opened a PR to [2]. Once merged, someone needs to manually update the job in the Jenkins instance. I've not idea who is going to do it. This is completely inefficient and error prone.

- Now suppose I want to create a *new* job then I need to somehow write several XMLs. I can copy-and-edit from existing XMLs. And I won't waste time explaining how insane it is.

- Because this manual handling of the jobs, what is running in the instance may not be exactly the code found in the repo [2].

- It has a complex combination of tests vs configuration vs OSes being mapped to a lot of jobs, and this is all managed manually (and editing the XMLs!). Again completely error prone and inefficient.

Currently there are tools to solve those problems. Days ago I proposed to the community a solution and showed a PoC, see the issue #343 below.

Issues: https://github.com/kata-containers/ci/issues/343

EPIC: # R3. Test environment should be reproducible locally #

We can look at this from the CI and developer angles. There are various scenarios, let me illustrate just a few:

- I, as a developer working on my Fedora 32 laptop, who opened a PR that eventually broke CI on Ubuntu, want to have it created the same Ubuntu VM environment which is used on CI so that I can locally reproduce/debug/fix the issue

- I, as a developer, need to test a change won't break it on k8s so I want to have a VM for the various OSses with k8s installed (same way as in CI)

- I, as a CI admin, want to test a change won't break the jobs on CentOS 8 before I open the pull request.

Because Kata is supported on many OSes the CI admin should be able to test new/changed jobs on all those OSes. Also the developer should be able to easily reproduce the CI locally.

This is related to R2.

Issues: https://github.com/kata-containers/documentation/issues/744

EPIC: # R4. Easily spot the problem with a job #

- I, as a developer, who opened a PR that eventually broke CI on Ubuntu, want to know what was wrong so that I can fix the issue.

This is not very easy because the artifacts are just a bunch of logs.

By the rule the jobs don't export the test results in Jenkins UI. So if you want to know what is wrong, you need to inspect the console log (single and long txt file where all the test suites append their results). Know what is wrong for a single job is hard, sum to this the fact that many jobs will be executed (and some likely to fail) when you open a PR.

Action items for the next meeting:

- People o review the document from Cameron
- Cameron/Wainer to try and identify a simple smoke test job we can still push to the upstream kata (hello world)
- Ariel to set another meeting

CI pipeline

Ci_entry_point.sh

- Set tests_repo (kata-containers/tests)
- Set repo_to_test (repo that triggered the job)
- PR related variables from GHPRB plugin
- Checks out tests repo
- Checks out origin/${ghprbTargetBranch} (which has to be wrong)
- Call jenkins_job_build.sh with repo_to_test

Jenkins_job_build.sh

- Set CI=true if KATA_DEV_MODE not set
- Set ci dir for exception of kata-containers repo (ci rather than .ci)
- Setup jenkins workspace
    - Special setup for baremetal
- Install go with install_go.sh -p -f
- Resolve-kata-dependencies.sh
- Static analysis if not metrics run on arches that travis doesn’t support
- Fast fail after static analysis if possible (ci-fast-return.sh)
- Setup variables for kata env
- Run setup.sh (in trigger repo which in turn calls setup.sh in tests)
- Log kata-runtime env
- Metrics stuff for metrics run (METRICS_CI): run_metrics_PR_ci.sh
- VFIO_CI=yes
    - Install initrd (TEST_INITRD=yes): install_kata_image.sh

        - Install_qemu_experimental.sh
        - Install_kata_kernel.sh
        - Install_cloud_hypervisor.sh
        - Run.sh (in trigger repo then in run.sh in tests)
    - Else do default
        - Run unit tests for everything but tests repo
        - Exception for rhel for runtime repo: skip
        - run.sh
        - Report code coverage

Install_go.sh

- Use versions file
- Force
- Install specific version of go into /usr/local/go

Resolve-kata-dependencies.sh

- Clone all the kata-containers repos using go get
- Checkout branches for dependent repos

Setup.sh

setup_type=minimum for travis
Default for everything else
Distro env (setup_env_$distro.sh)

- Install dependent packages

Install Docker

- No docker for cgroups v2
- cmd/container-manager/manage_ctr_mgr.sh" docker install
- If not the version in versions.yaml then same command with “-f”

Enable nested virt

- Only for x86_64 and s390x
- Modprobe option

Install kata: install_kata.sh
Install extra tools:

- Install CNI plugins: install_cni_plugins.sh
- Load arch-specific lib file: ${arch}/lib_setup_${arch}.sh
- Install CRI
    - For fedora: KUBERNETES=no
    - CRIO: install_crio.sh, configure_crio_for_kata.sh
    - CRI_CONTAINERD: install_cri_containerd.sh, configure_containerd_for kata.sh

        - KUBERNETES: install_kubernetes.sh
        - OPENSHIFT: install_openshift.sh

Disable systemd-journald rate limit
- RateLimitInterval 0s
- RateLimitBurst 0

Drop caches
- echo 3 > /proc/sys/vm/drop_caches

If rhel 7
- echo 1 > /proc/sys/fs/may_detach_mounts

Install_kata.sh

Install kata image
- rust agent image install_kata_image_rust.sh
- or non-rust agent image install_kata_image.sh

Install kata kernel
- instal_kata_kernel.sh

Install shim
- install_shim.sh

Install runtime
- install_runtime.sh

Install qemu
- For cloud-hypervisor: install_cloud_hypervisor and install_qemu with experimental_qemu=true
- For firecracker: install_firecracker.sh
- For qemu: install_qemu.sh

Configure podman if cgroupsv2 is being used
- configure_podman_for_kata.sh

Check kata: kata-runtime kata-check

Run.sh

RUNTIME=”kata-runtime”
Scenarios with case statement using CI_JOB

CI_JOB env and tests

Defaults in setup

CRIO="${CRIO:-yes}"

CRI_CONTAINERD="${CRI_CONTAINERD:-no}"
KUBERNETES="${KUBERNETES:-yes}"
OPENSHIFT="${OPENSHIFT:-yes}"
TEST_CGROUPSV2="${TEST_CGROUPSV2:-false}"

Defaults in jenkins_job_build

Invoked with init_ci_flags
CI="true"
KATA_DEV_MODE="false"
CRIO="no"
CRI_CONTAINERD="no"
CRI_RUNTIME=""
DEFSANDBOXCGROUPONLY="false"
KATA_HYPERVISOR=""
KUBERNETES="no"
MINIMAL_K8S_E2E="false"
TEST_CGROUPSV2="false"
TEST_CRIO="false"
TEST_DOCKER="no"
experimental_kernel="false"
RUN_KATA_CHECK="true"
METRICS_CI=""
METRICS_CI_PROFILE=""
METRICS_CI_CLOUD=""
METRICS_JOB_BASELINE=""

CRI_CONAINTERD_K8S

# This job only tests containerd + k8s

Environment

CRI_CONTAINERD="yes"
KUBERNETES="yes"
CRIO="no"
OPENSHIFT="no"

Tests

Containerd checks

- make cri-containerd

Running kubernetes testswith containerd as CRI

- CRI_RUNTIME=containerd make kubernetes

[configure for sandbox cgroup only]
Run test for cri-containerd with Runtime.SandboxCgroupOnly as True

- make cri-containerd

Run tests for kubernetes with containerd as CRI with Runtime.SandboxCgroupOnly as True

- CRI_RUNTIME=containerd make kubernetes

[remove configuration for sandbox cgroups only]

Running docker integration tests with sandbox cgroup enabled

- make sandbox-cgroup

FIRECRACKER

Environment

Tests

Running docker integration tests

- make docker

Running soak test

- make docker-stability

Running oci call test

- make oci

Running networking tests

- make network

Running crio tests

- make crio

CLOUD-HYPERVISOR

Environment

CRIO="no"
CRI_CONTAINERD="yes"
CRI_RUNTIME="containerd"
KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="yes"
OPENSHIFT="no"
TEST_CRIO="false"
TEST_DOCKER="true"
experimental_kernel="true"

Tests

Running soak test

- make docker-stability

Running oci call test

- make oci

Running networking tests

- make network

Running filesystem tests

- make conformance

CLOUD-HYPERVISOR-DOCKER

Environment

CRIO="no"
CRI_CONTAINERD="no"
KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="no"
OPENSHIFT="no"
TEST_CRIO="false"
TEST_DOCKER="true"
experimental_kernel="true"

Tests

Running docker integration tests

- make docker

CLOUD-HYPERVISOR-PODMAN

Environment

KATA_HYPERVISOR="cloud-hypervisor"
TEST_CGROUPSV2="true"
experimental_kernel="true"

Tests

[create trusted group]
Running podman integration tests

- make podman

CLOUD-HYPERVISOR-K8S-CONTAINERD

Environment

Init_ci_flags
CRI_CONTAINERD="yes"
CRI_RUNTIME="containerd"
KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="yes"
experimental_kernel="true"

Tests

Containerd checks

- make cri-containerd

Running kubernetes tests

- make kubernetes

CLOUD-HYPERVISOR-K8S-E2E-CRIO-MINIMAL

Environment

init_ci_flags
CRIO="yes"
CRI_RUNTIME="crio"
KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="yes"
MINIMAL_K8S_E2E="true"
experimental_kernel="true"

Tests

Run kubernetes e2e tests

- make kubernetes-e2e

CLOUD-HYPERVISOR-K8S-E2E-CONTAINERD-MINIMAL

Environment

init_ci_flags
CRI_CONTAINERD="yes"
CRI_RUNTIME="containerd"
KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="yes"
MINIMAL_K8S_E2E="true"
experimental_kernel="true"

Tests

Run kubernetes e2e tests

- make kubernetes-e2e

CLOUD-HYPERVISOR-K8S-E2E-CRIO-FULL

Environment

init_ci_flags
CRIO="yes"
CRI_RUNTIME="crio"

KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="yes"
MINIMAL_K8S_E2E="false"
experimental_kernel="true"

Tests

Run kubernetes e2e tests
- make kubernetes-e2e

CLOUD-HYPERVISOR-K8S-E2E-CONTAINERD-FULL

Environment

init_ci_flags
CRI_CONTAINERD="yes"
CRI_RUNTIME="containerd"
KATA_HYPERVISOR="cloud-hypervisor"
KUBERNETES="yes"
MINIMAL_K8S_E2E="false"
experimental_kernel="true"

Tests

Run kubernetes e2e tests
- make kubernetes-e2e

PODMAN

Environment

TEST_CGROUPSV2="true"

Tests

[create trusted group]
Running podman integration tests
- make podman

RUST_AGENT

Environment

Tests

Running docker integration tests
- make docker

Running soak test

- make docker-stability

Running kubernetes tests

- make kubernetes

VFIO

Environment

Tests

Running VFIO functional tests

- make vfio

SNAP

Environment

Tests

Running docker tests ($PWD)

- make docker

Running crio tests ($PWD)

- make crio

Running kubernetes tests ($PWD)

- make kubernetes

Running shimv2 tests ($PWD)

- make shimv2

VIRTIOFS-METRICS-BAREMETAL

Environment

experimental_qemu="true"
experimental_kernel="true"
METRICS_CI="true"
METRICS_CI_PROFILE="virtiofs-baremetal"

Tests

Running checks

- make check

Running functional and integration tests ($PWD)

- make test

SANDBOX_CGROUP_ONLY

Used by runtime makefile to enable option on install

Environment

DEFSANDBOXCGROUPONLY=true

Tests

Running checks
- make check

Running functional and integration tests ($PWD)
- make test

CLOUD-HYPERVISOR-METRICS-BAREMETAL

Environment

init_ci_flags
KATA_HYPERVISOR="cloud-hypervisor"
METRICS_CI="true"
experimental_kernel="true"
METRICS_CI_PROFILE="clh-baremetal"
METRICS_JOB_BASELINE="metrics/job/clh-master"

Tests

Running checks
- make check

Running functional and integration tests ($PWD)
- make test

Make Targets

check

checkcommits: make -C cmd/checkcommits
go test .
go install -ldflags "-X main.appCommit=${COMMIT} -X main.appVersion=${VERSION}" .

Log-parser: make -C cmd/log-parser
install -d $(shell dirname $(DESTTARGET))

```
install $(TARGET) $(DESTTARGET)
go build -o "$(TARGET)" -ldflags "-X main.name=${TARGET} -X main.commit=${COMMIT} -X
main.version=${VERSION}" .
go test .
```

test

crio
compatibility
configuration
conformance (if CI and TEST_CONFORMANCE are true)
debug-console
docker (if CI and TEST_DOCKER are true)
docker-compose
docker-stability
entropy
functional
kubernetes
netmon
network
oci
openshift
pmem
podman (if CI and TEST_CGROUPSV2 are true)
ramdisk
shimv2
swarm
time-drift
tracing
vcpus
vm-factory

cri-containerd

```
bash integration/containerd/cri/integration-tests.sh
```

kubernetes

```
bash -f .ci/install_bats.sh
bash -f integration/kubernetes/run_kubernetes_tests.sh
```

sandbox-cgroup

```
bash -f integration/sandbox_cgroup/sandbox_cgroup_test.sh
bash -f integration/sandbox_cgroup/check_cgroups_sandbox.sh
```

docker

```
ginkgo
bash sanity/check_sanity.sh
```

docker-stability

```
systemctl is-active --quiet docker || sudo systemctl start docker
cd integration/stability && \
export ITERATIONS=2 && export MAX_CONTAINERS=20 && ./soak_parallel_rm.sh
cd integration/stability && ./bind_mount_linux.sh
cd integration/stability && ./hypervisor_stability_kill_test.sh
```

oci

```
systemctl is-active --quiet docker || sudo systemctl start docker
cd integration/oci_calls && \
bash -f oci_call_test.sh
```

network

```
systemctl is-active --quiet docker || sudo systemctl start docker
bash -f .ci/install_bats.sh
bats integration/network/macvlan/macvlan_driver.bats
bats integration/network/ipvlan/ipvlan_driver.bats
bats integration/network/disable_net/net_none.bats
```

crio

```
bash .ci/install_bats.sh
RUNTIME=${RUNTIME} ./integration/cri-o/cri-o.sh
```

conformance

```
bash -f conformance/posixfs/fstests.sh
```

podman

```
bash -f integration/podman/run_podman_tests.sh
```

kubernetes-e2e

```
cd "integration/kubernetes/e2e_conformance" &&\
cat skipped_tests_e2e.yaml &&\
bash ./setup.sh &&\
bash ./run.sh
```

vfio

```
bash -f functional/vfio/run.sh -s false -p clh -i image
bash -f functional/vfio/run.sh -s true -p clh -i image
#   bash -f functional/vfio/run.sh -s false -p clh -i initrd
#   bash -f functional/vfio/run.sh -s true -p clh -i initrd
bash -f functional/vfio/run.sh -s false -p qemu -m pc -i image
bash -f functional/vfio/run.sh -s true -p qemu -m pc -i image
bash -f functional/vfio/run.sh -s false -p qemu -m q35 -i image
bash -f functional/vfio/run.sh -s true -p qemu -m q35 -i image
bash -f functional/vfio/run.sh -s false -p qemu -m pc -i initrd
bash -f functional/vfio/run.sh -s true -p qemu -m pc -i initrd
bash -f functional/vfio/run.sh -s false -p qemu -m q35 -i initrd
bash -f functional/vfio/run.sh -s true -p qemu -m q35 -i initrd
```

shimv2

```
bash integration/containerd/shimv2/shimv2-tests.sh
bash integration/containerd/shimv2/shimv2-factory-tests.sh
```