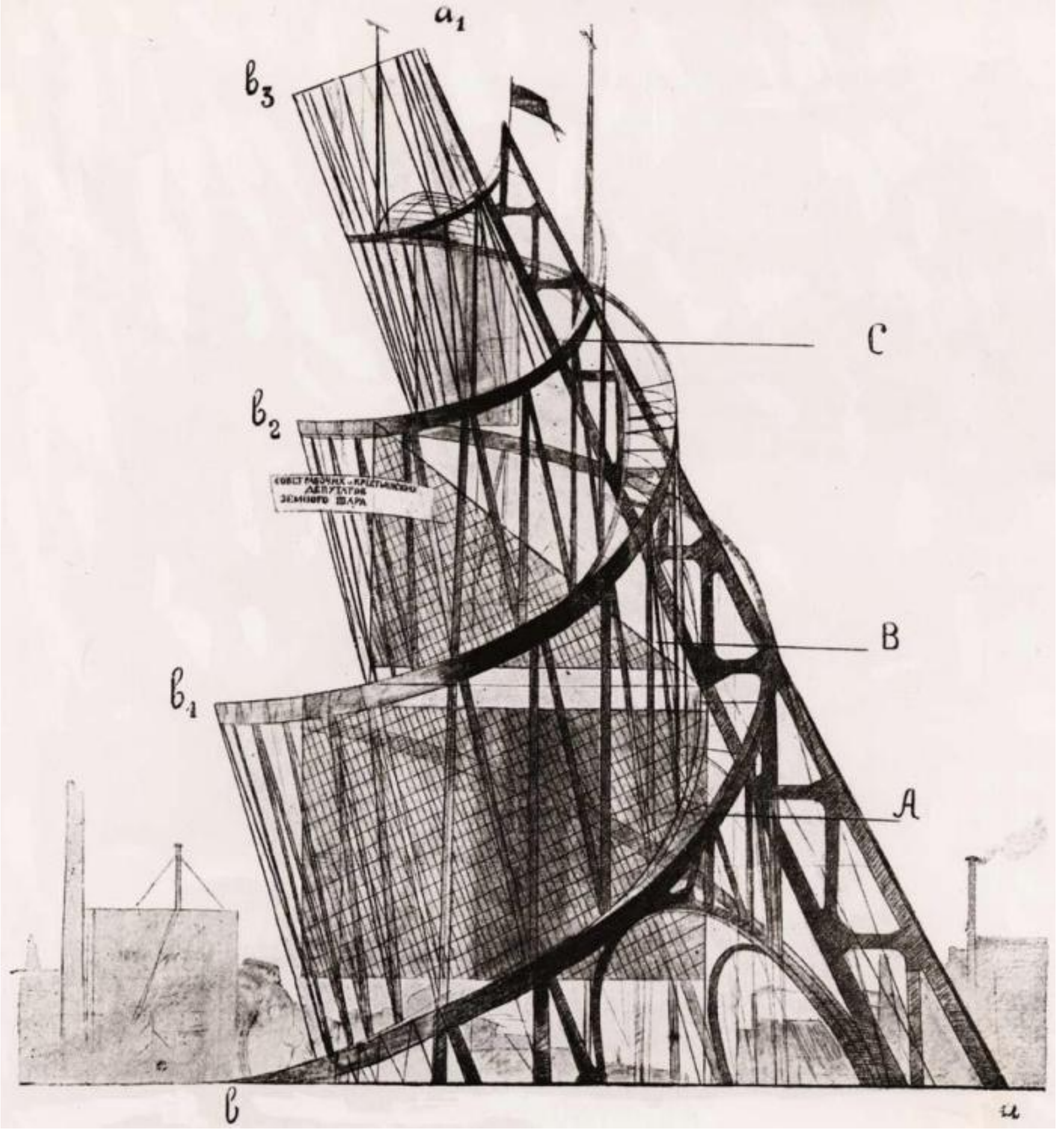


CI-friendly testing of kernel code

Roman Bolshakov
YADRO

What are we doing @ YADRO

- VESNIN: OpenPOWER-based server
- TATLIN: Unified Active-Active Storage Array
- Custom hardware engineered in-house
- Linux BIO/TCM-based Data Plane
- Proprietary Control Plane

Continuous Integration @ YADRO

- System / Component / Unit testing
- Multi-arch multi-platform parallel matrix builds/tests
- Pull Request validation
- On-demand Virtual TATLIN provisioning
- Automated QA is complemented with Manual QA
- Based on Ansible, Jenkins DSL, libvirt/KVM/QEMU, robot framework & openvswitch

«How can we test our kernel modules like other components?»

How Linux kernel is tested

- Linux Test Project (https://github.com/linux-test-project/ltp)
- blktests (https://github.com/osandov/blktests)
- xfstests (https://github.com/kdave/xfstests)
- libiscsi (https://github.com/sahlberg/libiscsi)
- lnst (https://github.com/jpirko/lnst)

The test suites can typically be run with

./configure

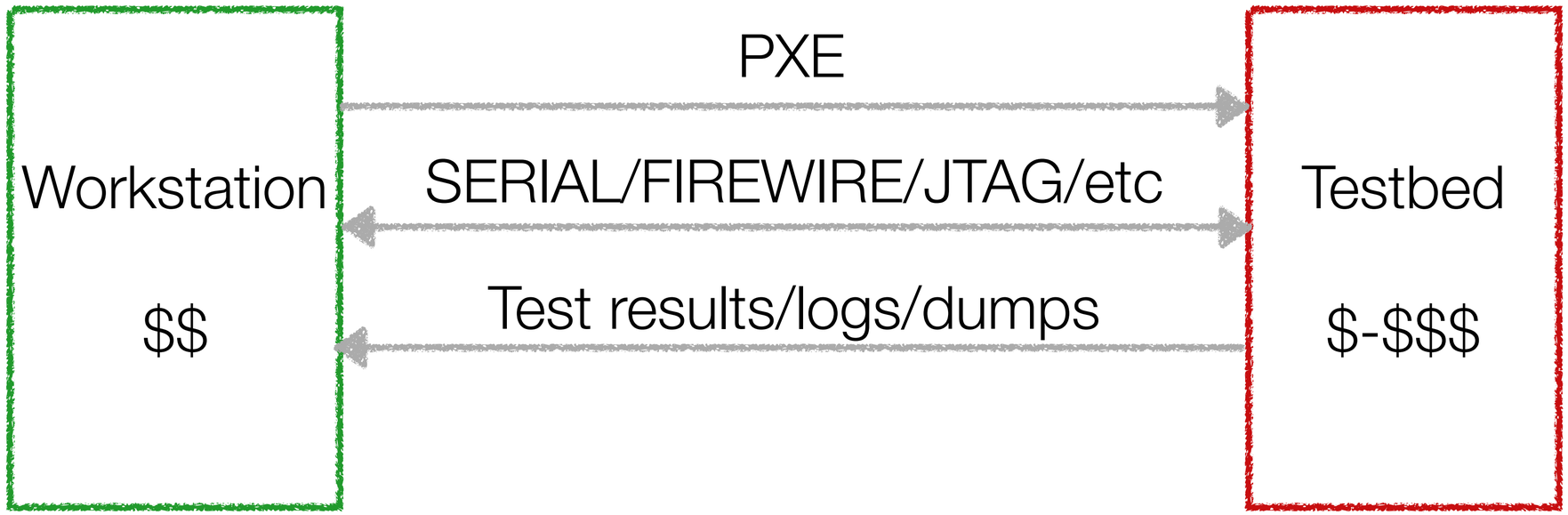
make

make install

./run-something

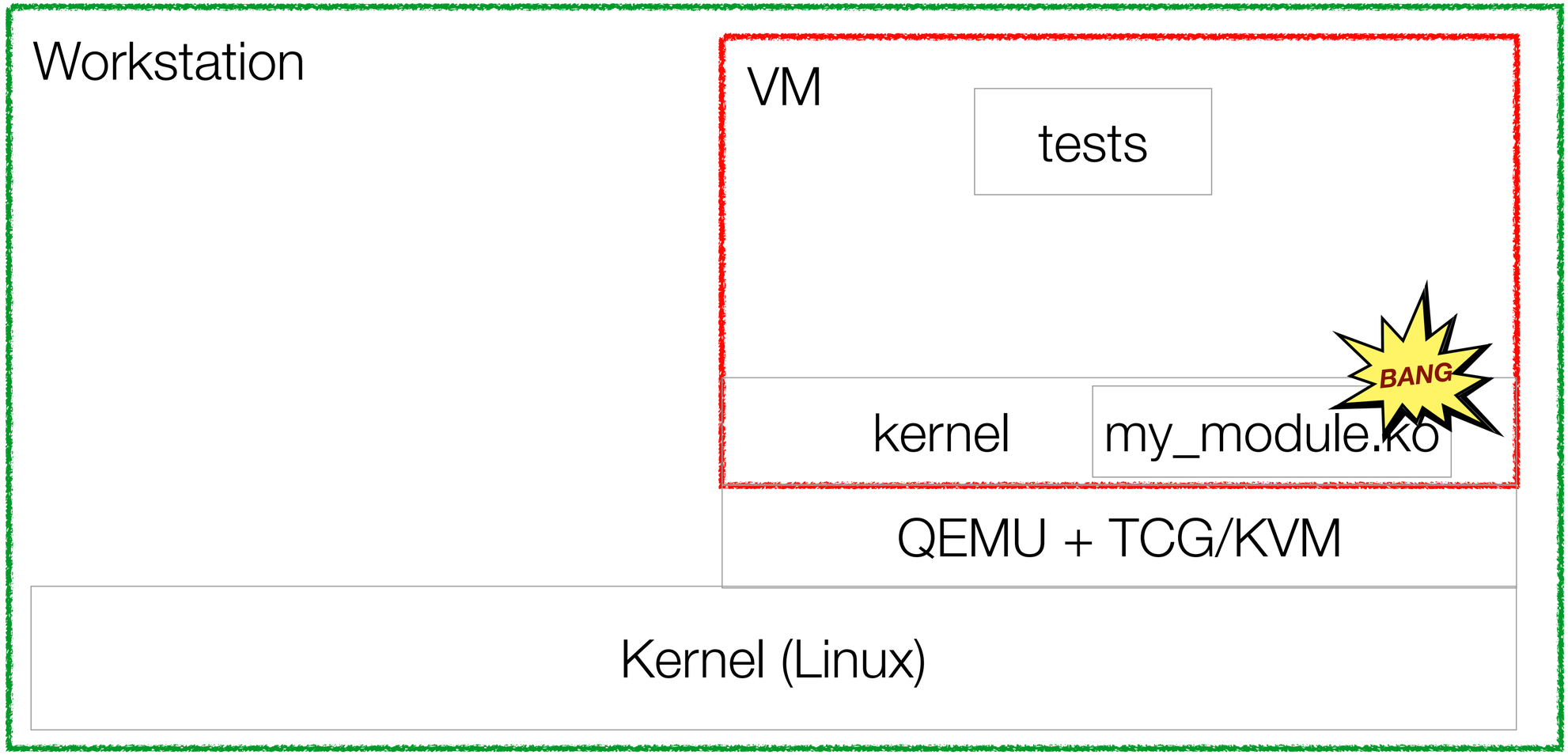
Running the tests directly on a workstation

Running the tests on dedicated hardware



Running the tests on a local VM

qemu-system-x86_64 -hda rootfs.qcow2



How to run the tests on local VM (9P-based wrappers for QEMU)

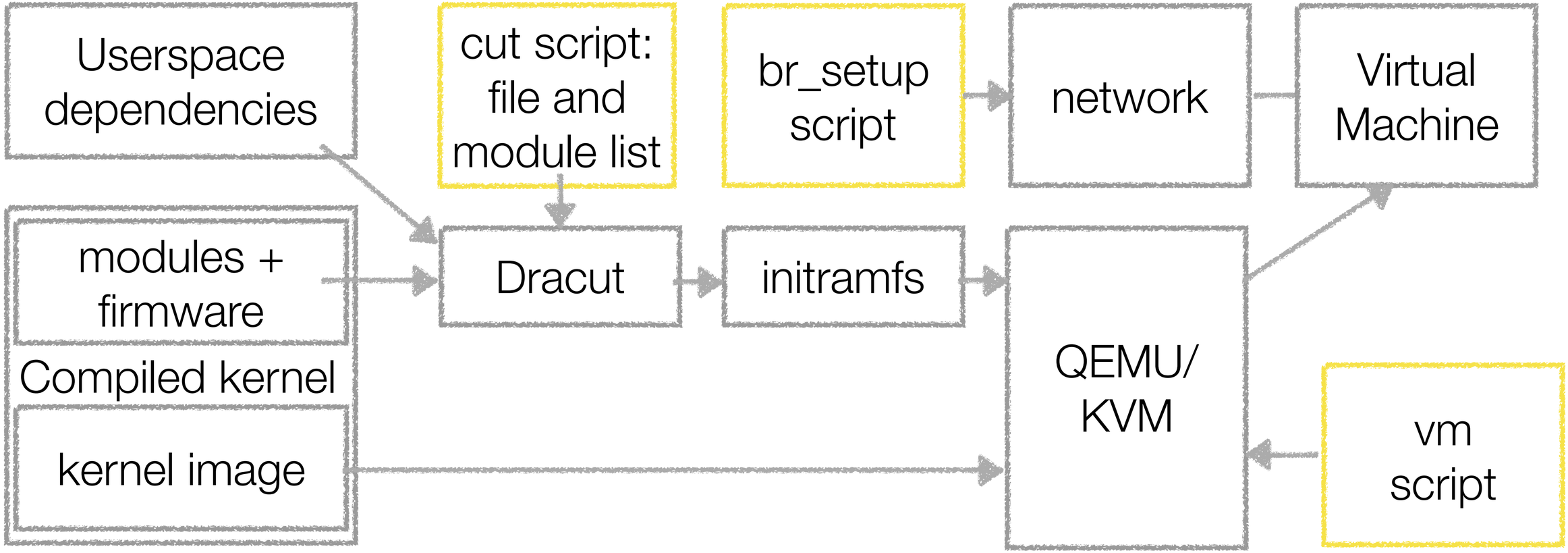
- virtme (https://github.com/amluto/virtme)
- eudyptula-boot (https://github.com/vincentbernat/eudyptula-boot)
- vido (https://github.com/g2p/vido)

Pros:

- fast VM start (especially on KVM hosts) ~7 seconds
- interactive mode
- no need to build rootfs or initramfs
- can be run as one command

How to run the tests on local VM with rapido

- rapido (https://github.com/rapido-linux/rapido)



autotest & avocado

- autotest framework (http://autotest.github.io)
- avocado framework (http://avocado-framework.github.io)
  - Fully-fledged test framework and test runner with plenty of plugins
  - Can be used as standalone framework for kernel testing per se with or without Jenkins
  - Plenty of reporting features and other extras
  - Can run tests locally, on a remote host, on a virtual machine or on a docker container
  - Can't provision the VMs/remote hosts

Gaps of existing tools

- can't gather vmcore if the kernel crashes during tests
- no ability to run on a remote machine (unless installed there)
- all tools used in testing have to be installed on the host
  - as a result can't run on macOS
  - closely tied to the environment where they're running
- lack of clear integration path with Jenkins

«What if we combine the best qualities of the tools and close some of the gaps?»

ktest Features/Goals

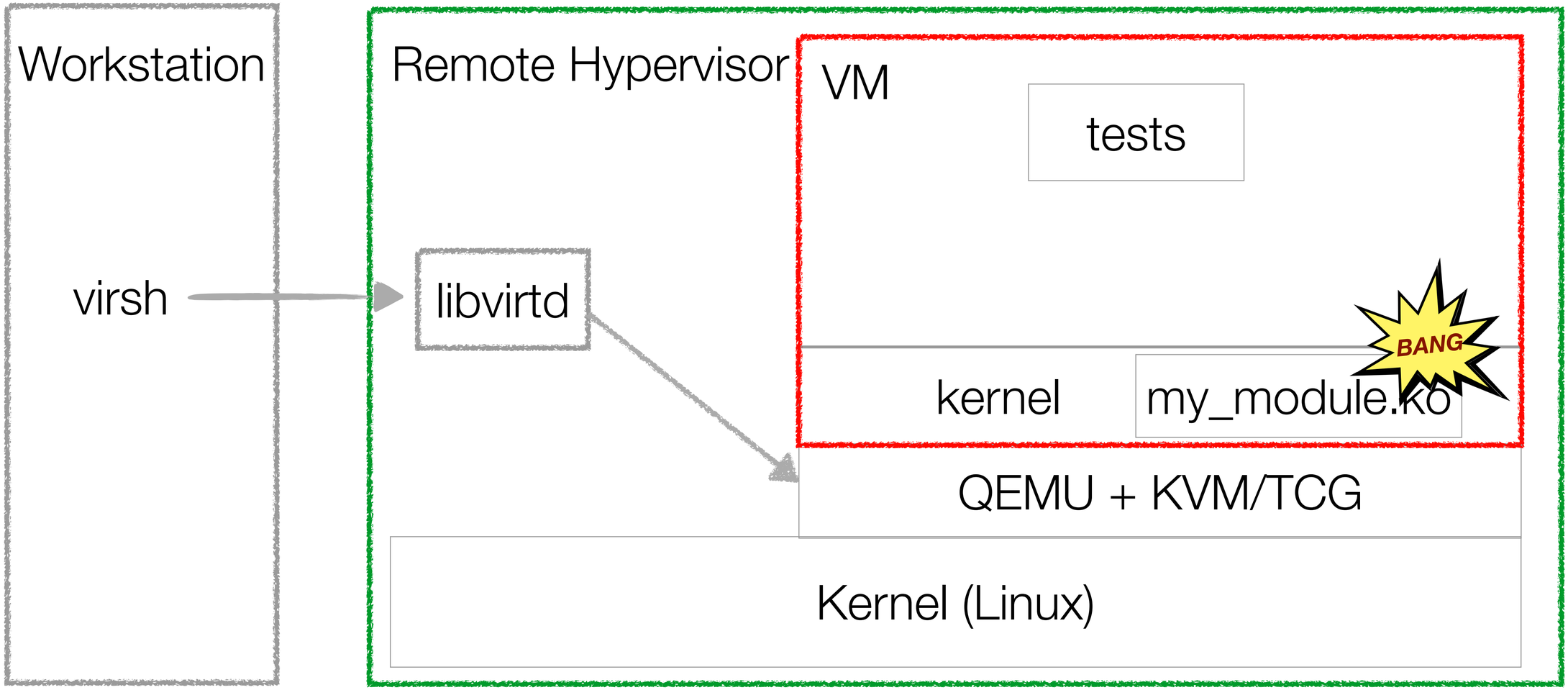
- Local/remote test execution
- Independence from environment
- Ease of use/interactive mode
- On-demand/throw-away VM provisioning
- Quick feedback
- Support of out-of-tree kernel modules
- Simple integration with Jenkins
- Simultaneous usage of the same hypervisor by many ktest instances

libvirt

- Can manage local or remote virtualization hosts
- Storage pool management, networking and many more
- Client - virsh/libvirt API
- Server - libvirtd
  - supports Linux, macOS and many other operating systems
  - QEMU with KVM acceleration can be used on Linux
  - Hypervisor.framework acceleration patchset posted https://www.redhat.com/archives/libvir-list/2018-October/msg01090.html

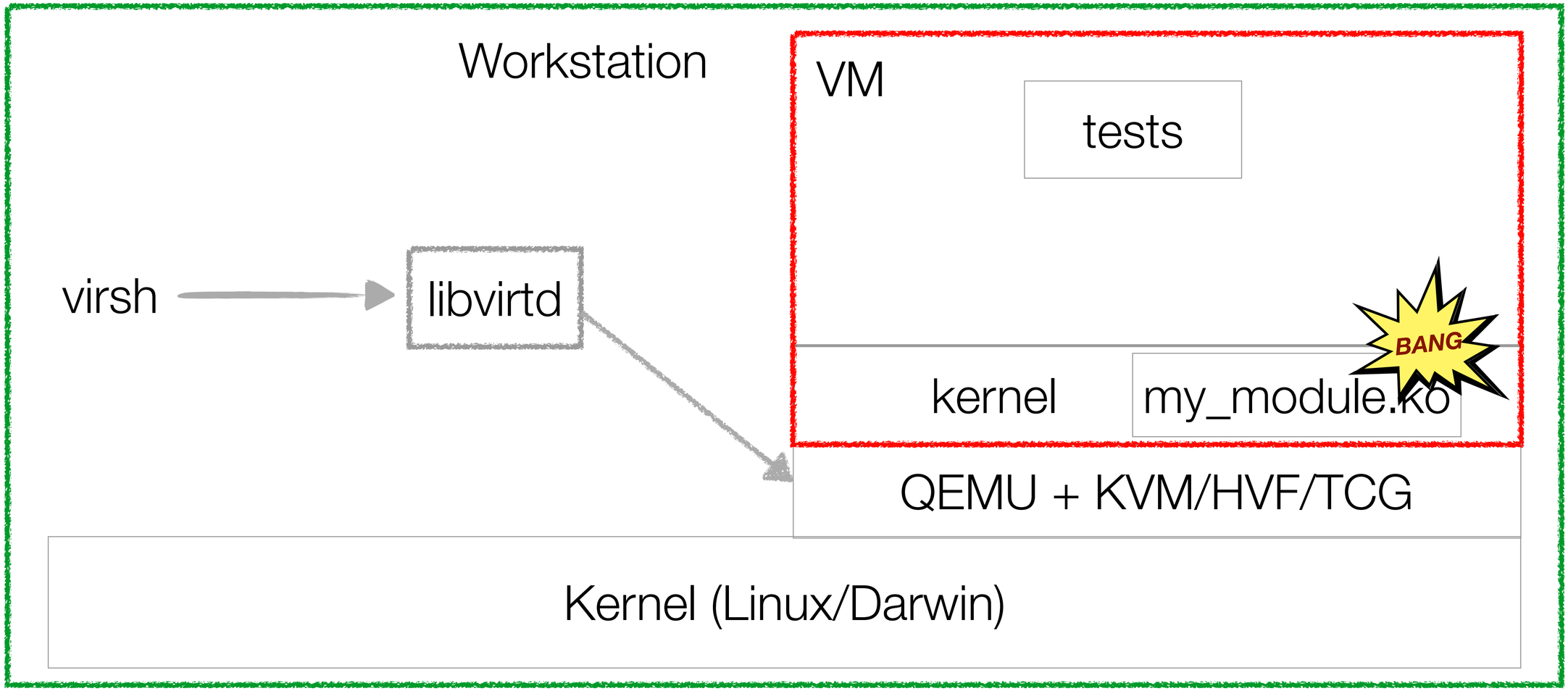
Running the tests on a remote VM

virsh define domain.xml && virsh start domain



Running the tests on a local VM like on remote

virsh define domain.xml && virsh start domain



ktest execution flow

VM setup

(can be done in docker)

build initramfs

prepare disk images

generate domain xml
import disks, kernel
and initramfs
reserve TCP port
for virtio ktest channel

define domain

Test run

Start domain

(within the VM)

Setup devices

Set up panic kernel

Run tests

Save logs/vmcore

Teardown

(within the VM)

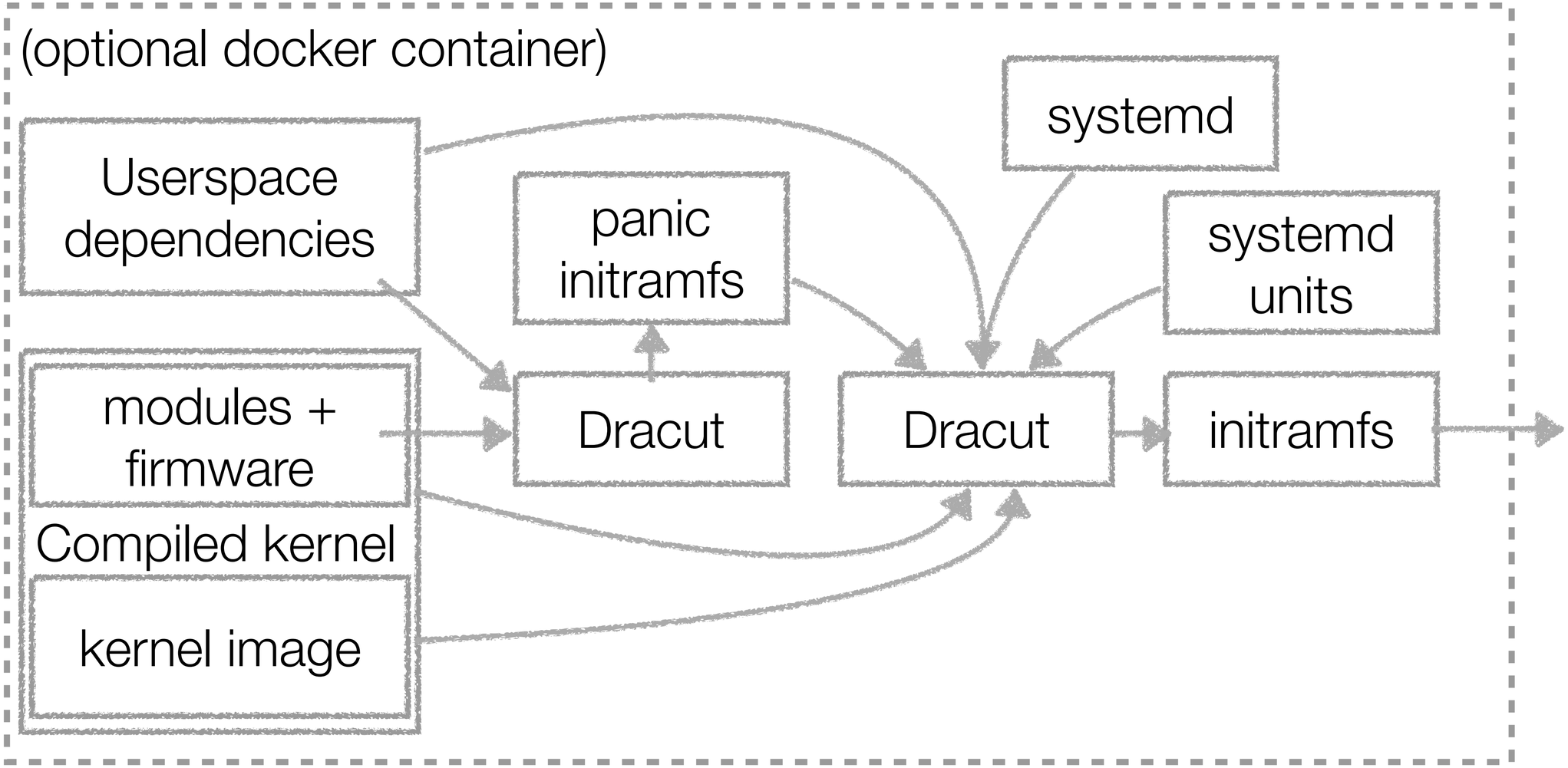
Funnel test results,
cores, logs
through virtio channel
Poweroff

Retrieve all test results

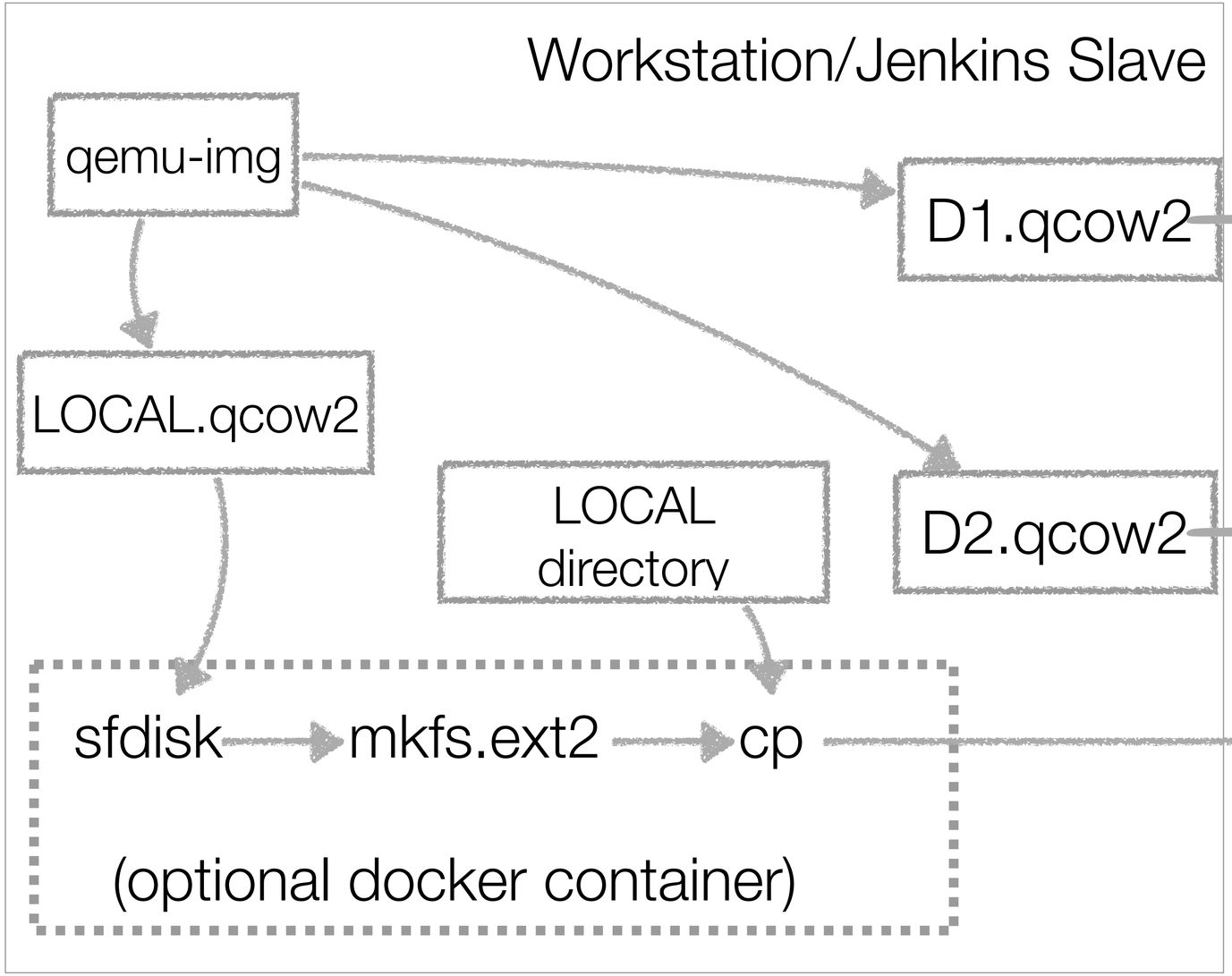
NMI or Destroy VM
if it times out

Clean up

ktest initramfs build

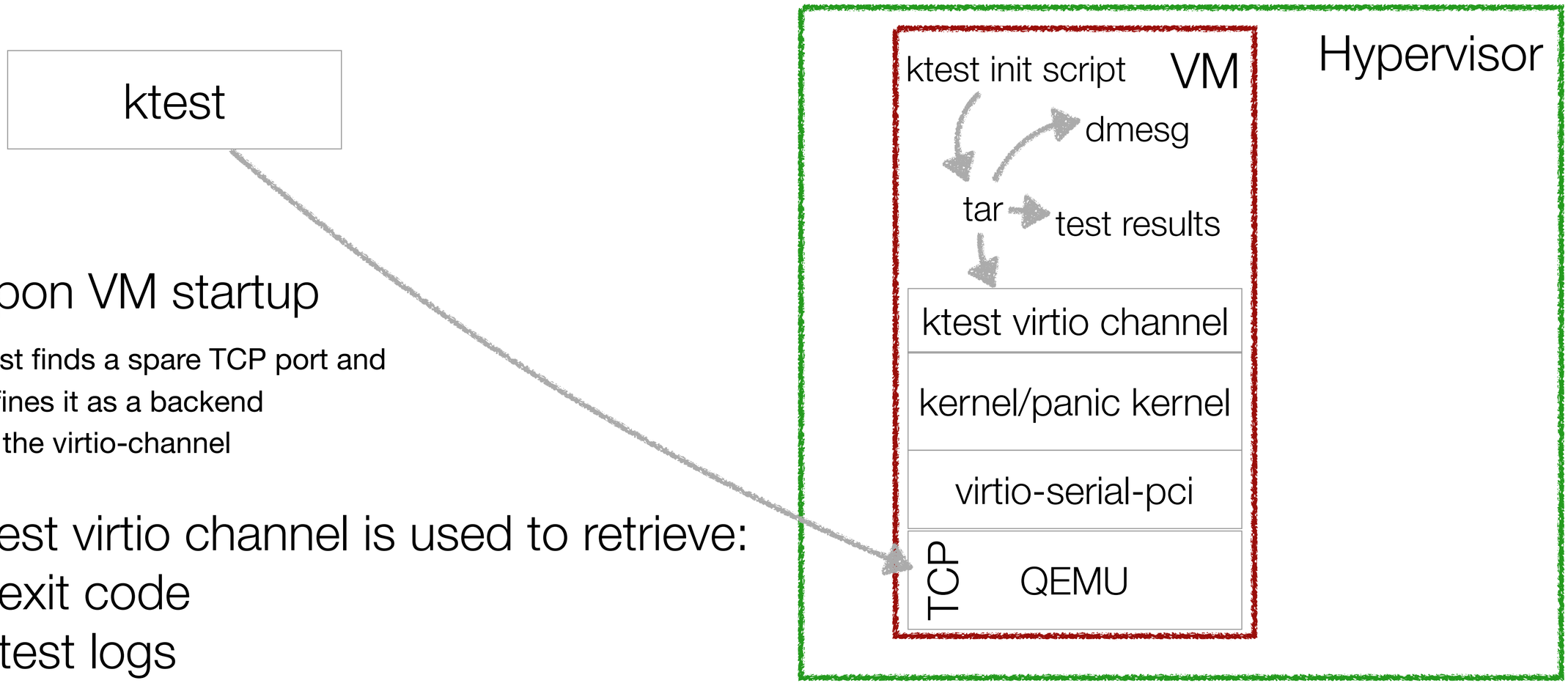


ktest user volumes and test disks




```
ktest --directory LOCAL:/REMOTE
      --disk D1:blk:100M
      --disk D2:scsi:1G
      --entry-point /REMOTE/run-test
      vmlinux
```

Retrieval of results/logs/vmcores



Upon VM startup

ktest finds a spare TCP port and
defines it as a backend
for the virtio-channel

ktest virtio channel is used to retrieve:

- exit code
- test logs
- dmesg
- vmcore

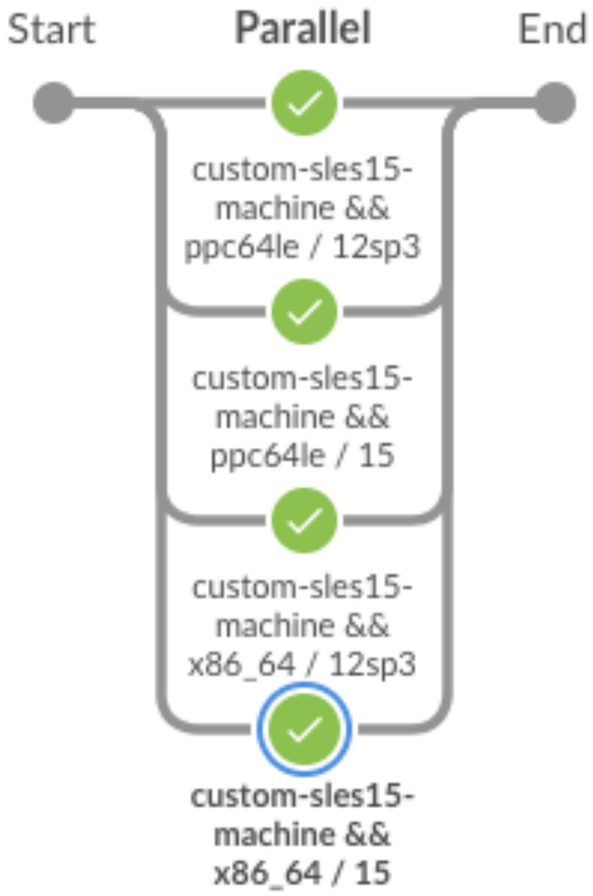
Integration of ktest with Jenkins for out-of-tree kernel modules (1)

- Prepare a set of Docker images that contain kernel, kernel-headers, compilers, etc
- Add ktest submodule to a kernel module git repo
- Create a Jenkinsfile that invokes build & test on each platform/arch
- Setup a set of libvirt hypervisors with KVM support
- Define ktest parameters in .ktest
- Add the command to test step in Jenkinsfile/test target in Makefile
- ktest/run --docker-image IMG --uri QEMU_URI vmlinux

Integration of ktest with Jenkins for out-of-tree kernel modules (2)

```
$ cat .ktest
ENTRY_POINT=/test/start
DIRS=(test:/test tatlin:/test/modules)
DISKS=(TEST:blk:10M)
MODULES=(target_core_mod tcm_loop)
INSTALL=(wc find rmdir dirname)
OUTPUT=test-results
```

Integration of ktest with Jenkins for out-of-tree kernel modules (3)



Все тесты проходят

Чудесно! Проходят все тесты 12 для этого конвейера.

Passed - 12

✓	custom-sles15-machine && x86_64 / 12sp3 / Unit Tests / 001 - RUNS /test/insmod-rmmod – start	<1s
✓	custom-sles15-machine && x86_64 / 12sp3 / Unit Tests / 002 - RUNS /test/backstore-create-delete – start	<1s
✓	custom-sles15-machine && x86_64 / 12sp3 / Unit Tests / 003 - RUNS /test/model-rev – start	<1s
✓	custom-sles15-machine && ppc64le / 12sp3 / Unit Tests / 001 - RUNS /test/insmod-rmmod – start	<1s
✓	custom-sles15-machine && ppc64le / 12sp3 / Unit Tests / 002 - RUNS /test/backstore-create-delete – start	<1s
✓	custom-sles15-machine && ppc64le / 12sp3 / Unit Tests / 003 - RUNS /test/model-rev – start	<1s
✓	custom-sles15-machine && ppc64le / 15 / Unit Tests / 001 - RUNS /test/insmod-rmmod – start	<1s
✓	custom-sles15-machine && ppc64le / 15 / Unit Tests / 002 - RUNS /test/backstore-create-delete – start	<1s
✓	custom-sles15-machine && ppc64le / 15 / Unit Tests / 003 - RUNS /test/model-rev – start	<1s
✓	custom-sles15-machine && x86_64 / 15 / Unit Tests / 001 - RUNS /test/insmod-rmmod – start	<1s
✓	custom-sles15-machine && x86_64 / 15 / Unit Tests / 002 - RUNS /test/backstore-create-delete – start	<1s
✓	custom-sles15-machine && x86_64 / 15 / Unit Tests / 003 - RUNS /test/model-rev – start	<1s

DEMO

Where to get

- https://github.com/YADRO-KNS/ktest

Further development

- ktest-compose to run a cluster of VMs
- watchdog device to panic on hard and soft-lockups
- VFIO-PCI passthrough
- Simplify installation of binaries into initramfs

Credits for inspiration

- virtme, a easy way to test your kernel changes in kvm/qemu - Andy Lutomirski (https://lwn.net/Articles/584620/)

- Kernel Recipes 2015 - Speed up your kernel development cycle with QEMU - Stefan Hajnoczi (https://lwn.net/Articles/660404/)

Q & A

THANK YOU

BACKUP

QEMU/Hypervisor.framework status on macOS

- 8-10 times faster than TCG
- Still experimental
  - Sometimes doesn't proceed beyond SeaBIOS
  - MMX/SSE emulation on EPT FAULT is missing
  - kvm-unit-tests do not pass
- Certain bugs have been fixed only in master branch