Restartable confidential guests on QEMU hypervisor - where is the challenge?

FOSDEM 2026

Ani Sinha
Principal Software Engineer,
Red Hat.

Red Hat

What we'll discuss today

- Background
- Implementation choices
- Challenges
- Demo
- Current state of patchset
- So where are the remaining challenges?
- Gratitude

Background

Current status in terms of restartability of QEMU VMs

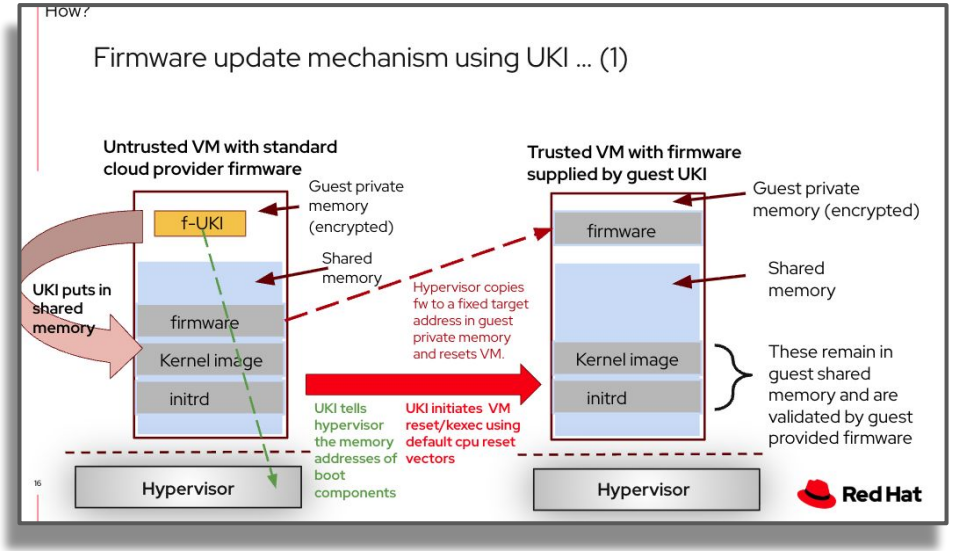
- Non-confidential VMs 
- Confidential SEV VMs 
- Confidential SEV-ES VMs 
- Confidential SEV-SNP VMs 
- Confidential TDX VMs 

Current status in terms of restartability of QEMU VMs

- SEV-ES, SEV-SNP and TDX encrypts and locks-in initial CPU register states at launch.
- Hypervisor is not able to modify the CPU state again which is required for reset.
- Reset is not possible unless we re-encrypt the initial CPU state again with a new key.
- A new confidential guest context from KVM's perspective needs to be established.
- SEV VMs only encrypt guest memory, not CPU register states.

Why resettability is important?

- Confidential guests can be at par with other types of guests.
  - Consistency across all types of guests.
- For F-UKI
  - "Introducing F-UKI, Guest firmware in a UKI for Confidential Cloud Deployments" - FOSDEM 2025.



Implementation Choices

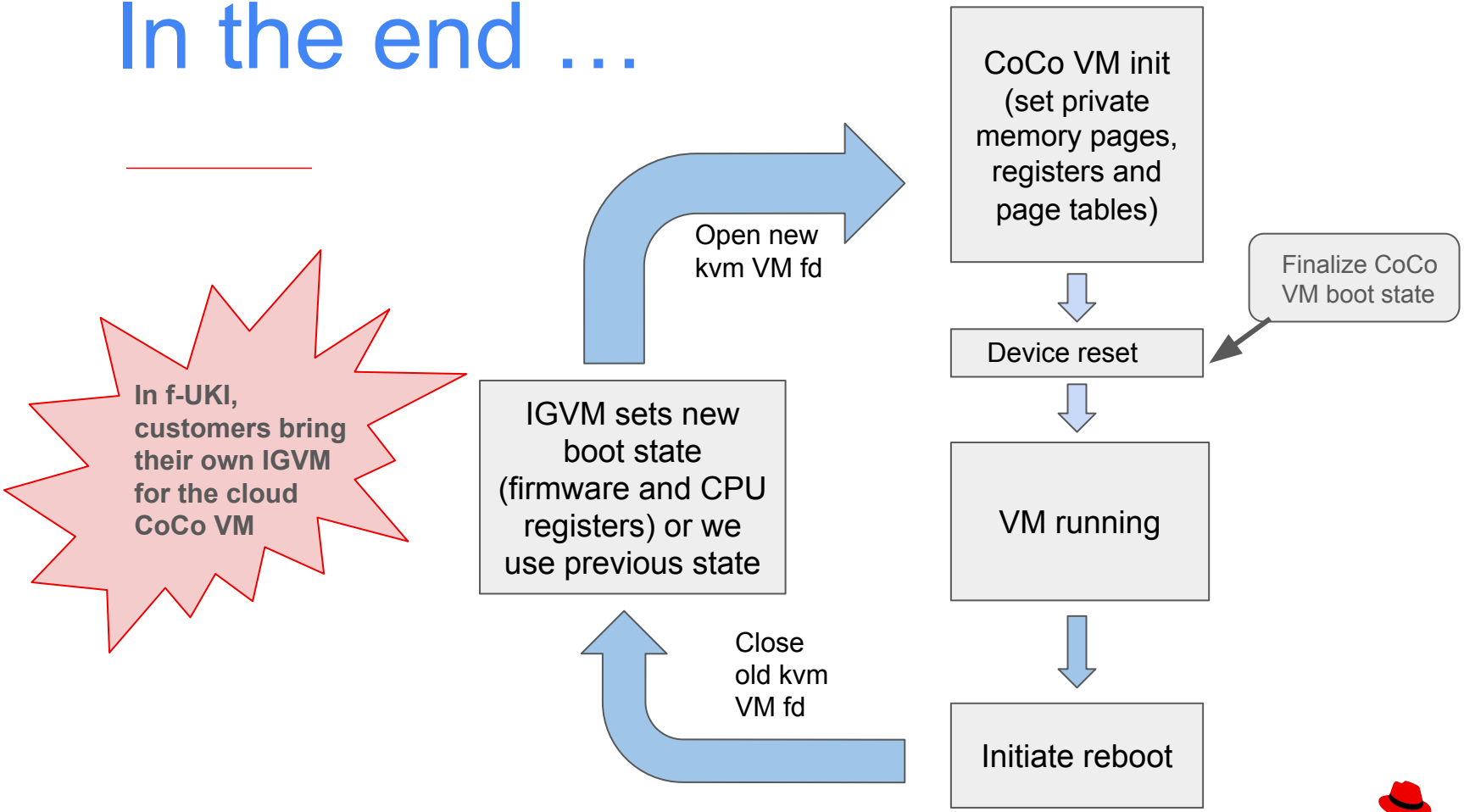
Implementation choices before us

- *"Unlike normal reset, resetting a confidential VMs entails performing all the encryption and measurement from scratch for memory and registers, and the data is not available to KVM anymore"* - Paolo Bonzini
- **Option 1**: KVM can save pre-encrypted initial state of memory, page tables and CPU registers.
  - Bloats VM state with duplicate guest data (one encrypted and another unencrypted).
  - What if we wanted to reset into a ***different*** initial launch state?
    - F-UKI needs this.
  - Potentially requires a new KVM API.

Implementation choices before us

- **Option 2**: Throw away current encrypted initial state, close the old KVM fd, open a new one and reset after encrypting the initial launch state with ***same or different*** data in guest memory, CPU registers etc.
  - Effectively, re-execute the same steps we did while launching a new VM for the first time.
  - Transition from unencrypted state to encrypted state as before.
  - From KVM's perspective, it's a **new/different** VM.
  - Same old QEMU process keeps running.
  - No new KVM API.

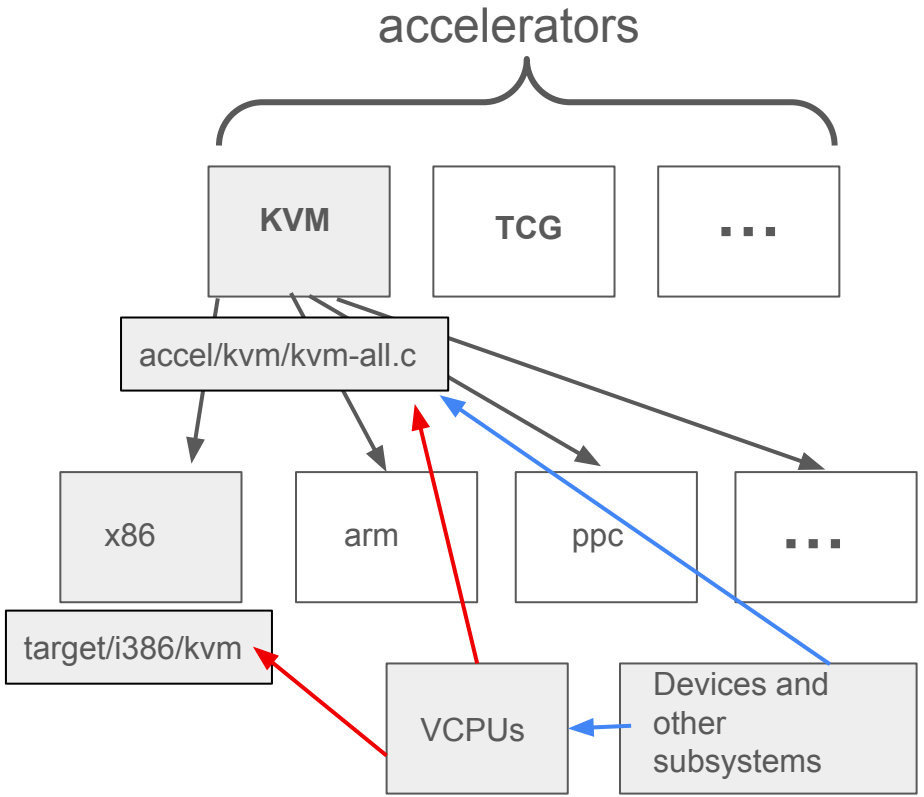
In the end …


CoCo VM init (set private memory pages, registers and page tables)
Finalize CoCo VM boot state
Device reset
VM running
Initiate reboot
Close old kvm VM fd
IGVM sets new boot state (firmware and CPU registers) or we use previous state
Open new kvm VM fd
In f-UKI, customers bring their own IGVM for the cloud CoCo VM

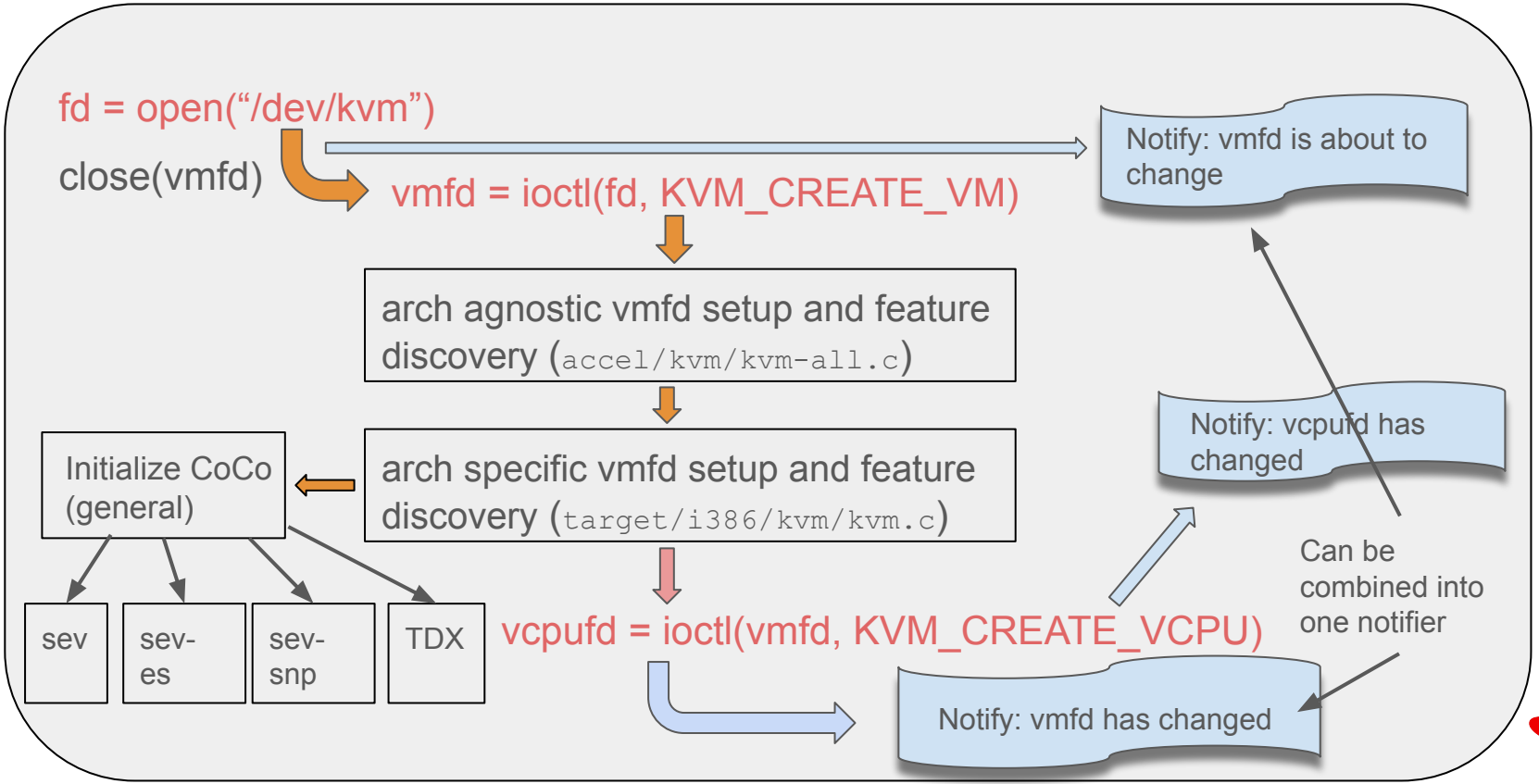
Challenges

Code Organization in QEMU

- Changing KVM VM file descriptor
  - Like pulling the carpet from under the feet and replacing it with another.
- VCPUs are created against this VM fd using ioctl() calls.
- Various devices and subsystems do ioctl() calls against this VM fd and against VCPU fds.



Flow of control



Challenges

- Find all places where `ioctl(vmfd …)` is used
  - Reissue those ioctl calls …
  - Except for feature discovery (results cached).
  - Same for other subsystems and devices.
  - Free old data where required and let init allocate it again.
- Setup VCPUs again using `ioctl(vcpufd …)`
  - Subsystems needs to setup vcpu features again.
- Understand the code first
  - Migration related save and restores are exempted.

Challenges

- strace() is the friend
  - It's easy to miss all call paths.
  - Need to cover all ioctl() paths and all ioctl() values.
- Unfamiliarity with different subsystems
  - Opportunity to learn random stuff.
  - Like not all init functions are meant to be called again!
    - Fix them!
- Lots of test and debug cycles.
- Most of the debugging can happen in non-coco environment.
  - Except CoCo specific initialization.

Demo (non-CoCo)!

For SEV-ES, SEV-SNP and TDX demos, visit …

… or the FOSDEM talk page.

Current state of patchset

Current state of patchset

- **x86-64** ONLY for now.
- Version 3 of the patchset posted on the qemu-devel mailing list.
- Testing status:
  - **Confidential**: SEV-ES, SEV-SNP, TDX.
  - **Non-confidential hosts**: with special debug machine flag.
  - With KVM **Xen** emulation enabled.
  - QEMU upstream **CI pipeline** is good. No regressions.
  - Some of my test scripts are here. A f**unctional test** is added as a part of v2/v3 patchset for non-coco case.
- Targeting upstream QEMU version **11.0 or 11.1**.

So where are the remaining challenges?

Remaining challenges

- Get everything right.
- Get it tested with various combinations of devices/settings (CoCo and non-CoCo).
- Need community help.
- Need code reviewers looking at the patch-set for possible mistakes.
  - Have anything been missed?

Remaining challenges - FUKI integration

- IGVM integration with the reset mechanism.
  - Gerd Hoffman has patches on the mailing list.
- Fw-cfg device (`vmfwupdate`) to pass the customer IGVM bundle to hypervisor.
  - Ties all the pieces together.
  - An old version of the patch is available here.

Beyond the scope of this talk.

Gratitude!

AI generated image

- Paolo Bonzini (Red Hat).
- Gerd Hoffman (Red Hat).
- QEMU Community.
- Red Hat.

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make
Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/@redhat


facebook.com/RedHat


x.com/RedHat