Transforming kernel developer workflows with CI/CD

Major Hayden // Red Hat
@majorhayden
Major Hayden

🤓 Principal Software Engineer at Red Hat
🤖 Working on Continuous Kernel Integration (CKI)
🤠 From Texas (but I don’t own a horse)
📻 Ham radio operator (W5WUT)
🐱 Creator of icanhazip.com
Kernel development is difficult.
It’s not easy being a kernel maintainer, either.
For one release:

~ 17,000 changes merged
~ 1,700 developers contributing

(150-300 are first time contributors)
For one release:

> 20,000 new bugs introduced

(estimated)

Source: Dmitry Vyukov's talk at LPC 2019 (and LWN article)
End users expect Linux distros to handle it, but the volume is massive
I fixed a bug! How do I test it?
How long does it take to compile the kernel on my machine?
How do I trigger the bug and test the change?
Does my fix introduce a performance problem?
Does my change cause problems on another architecture?
Does my change create a vulnerability? 😱
Do I have the right hardware to test it?
Sure, the hardware is within my budget.
Okay, this is not in my budget.
Do I have the patience to maintain the hardware?
Do I have git and an SMTP server set up properly?

(just kidding) 😏
Can we have a better experience for us all?

“After posting, remain patient and wait for a reply. Do not be discouraged by any negative response—at least you got a response! Discuss the issues and provide updated patches as needed. **If you fail to receive any response, try to discover what was wrong and resolve the issues.** Solicit additional comments from the mailing list and maintainer. With luck, you might see your changes in a future kernel release—congratulations!”

From *Linux Kernel Development* by Robert Love
What if the kernel had CI like many other software projects?
Linux CI projects*

0-Day
CKI
Fuego
KernelCI
ktests
LKFT
Syzbot

* small sample of current projects
Continuous Kernel Integration (CKI) facts

- Went into production in 2018
- Sponsored by Red Hat
- Frequent testing of downstream and upstream trees to find regressions
- Already in use with internal Red Hat Enterprise Linux kernels
CKI testing is triggered by events:

- git repository changes
- patches to mailing lists
- quilt patch list changes
- RPM build system
Step 1: Create a source artifact

- quilt patches
- email patches
- kernel repo
- kconfig
- tarball
Step 2: Compile

[Image: Diagram showing the compilation process for different architectures (aarch64, ppc64le, s390x, x86_64) using GitLab.]
Step 3: Test on hardware (VM or bare metal)
Step 4: Feedback via email

GitLab → ❤ reporter →
- mailing list
- patch submitter
- kernel maintainer
- test maintainer
What gets tested?
Over 50 tests are available in various categories

<table>
<thead>
<tr>
<th>ACPI tables</th>
<th>Module load</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMTU</td>
<td>Networking (lots)</td>
</tr>
<tr>
<td>Container system integration</td>
<td>NFS</td>
</tr>
<tr>
<td>CPU frequency drivers</td>
<td>NUMA</td>
</tr>
<tr>
<td>CPU idle</td>
<td>PCI</td>
</tr>
<tr>
<td>CPU MSR</td>
<td>Power management</td>
</tr>
<tr>
<td>FIPS</td>
<td>Realtime</td>
</tr>
<tr>
<td>IOMMU</td>
<td>Sound</td>
</tr>
<tr>
<td>JVM</td>
<td>Storage</td>
</tr>
<tr>
<td>KASLR</td>
<td>stress-ng</td>
</tr>
<tr>
<td>KVM</td>
<td>xfstests</td>
</tr>
<tr>
<td>Loopback devices</td>
<td>vm</td>
</tr>
<tr>
<td>LTP</td>
<td></td>
</tr>
</tbody>
</table>
Targeted tests are selected based on the:

- 📦 kernel repository
- 📩 patch content
- 🖥 system architecture
- 💬 tags and release data
What do kernel developers get?
Hello,

We ran automated tests on a recent commit from this kernel tree:

- Commit: fs579330a296 - Linux 5.2.10.rc1

The results of these automated tests are provided below:

Overall result: PASSED
- Merge: OK
- Compiles: OK
- Tests: OK

All kernel binaries, config files, and logs are available for download here:

https://artifacts.cki-project.org/pipelines/198120

Please reply to this email if you have any questions about the tests that we ran or if you have any suggestions on how to make future tests more effective.

---

Compile testing

We compiled the kernel for 3 architectures:

```make
arch64:
make options: -j30 INSTALL_MOD_STRIP=1 targa2-pkg
```

```make
pc64le:
make options: -j30 INSTALL_MOD_STRIP=1 targa-pkg
```

```make
x86_64:
make options: -j30 INSTALL_MOD_STRIP=1 targa-pkg
```

Hardware testing

We booted each kernel and ran the following tests:

```
arch64:
most 1:
  - Boot test [6]
  - Podman system integration test (as root) [1]
  - Podman system integration test (as user) [1]
  - LTP lite [2]
  - Ltpdev sanity [3]
  - Jem test suite [4]
  - Memory function: memfd_create [5]
  - AMT (Abstract Machine Test Utility) [6]
  - LTP: omevogix test suite [7]
  - Ethernet drivers sanity [8]
  - Networking socket: fuzz [9]
  - Networking scsi-path: scsioptests test [10]
  - Networking TCP: keepalive test [12]
  - Networking UDP: socket [13]
  - Networking tunnel: gre basic [14]
  - Networking tunnel: valan basic [15]
  - Audit: audit test sulog test [16]
  - httpd: mod_ssl smoke sanity [17]
  - Slog: sanity [18]
  - tuned: tune-processes-through-perf [19]
  - Users - version 1.9.29 [20]
  - Storage: SCSI VMD [21]
  - Stress: stress-ng [22]
  - Networking route:קטו [23]
  - Networking route func: local [24]
  - Networking route func: floppy [25]
```

Links to download built kernel artifacts

Links to test source code
Reports are delivered quickly after changes appear upstream:

- Failed merge: 2-5 minutes
- Failed compile: 10-15 minutes
- Full test results: < 5 hours
Fewer ignored patches

Submitted patch volume is increasing, but the ratio of ignored patches is dropping.

From comments during the talk at LPC 2019, CI is helping with some of this reduction.

Failures are identified early, failures are reviewed, and developers know when to their patches need work.
Maintainers also get additional context during patch reviews.
More context means maintainers can make decisions efficiently with more confidence.
Maintainers can find issues faster

> The results of these automated tests are provided below.
> 
> Overall result: FAILED  (see details below)
>     Patch merge: OK
>     Compile: FAILED

Should be fixed now, sorry about that. Your failure message came back faster than my internal build systems caught this. I think I need to upgrade my build system :)

thanks,

greg k-h
(The maintainers also like to test us.)

```
[ 2.947836] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.20.4-rc1.cki+
[ 2.963850] Call trace:
[ 2.966286]   dump_backtrace+0x0/0x158
[ 2.969930]   show_stack+0x24/0x30
[ 2.973228]   dump_stack+0x90/0xb4
[ 2.976526]   panic+0x12c/0x294
[ 2.979565]   _cpu_down+0x0/0x228
[ 2.982776]   ret_from_fork+0x10/0x18
[ 2.986337] SMP: stopping secondary CPUs
[ 2.993711] CPU features: 0x0,20802000
[ 3.000481] ---[ end Kernel panic - not syncing: boot canary hit ]---
```
(The maintainers also like to test us.)

```c
+ /*
+ * If you notice this, your test infrastructure succeeded in finding a
+ * failure, congratulations
+ */
+ panic("boot canary hit");
+```
Developers now know they are not working alone.
What’s next?

- Scaling up to work with more upstream trees
- Testing patches sent to mailing lists in upstream trees
- More architecture-specific testing
- Integration with KernelCI (and collaboration with LKFT)
- Fewer false positives and more clear reports
- Data warehouse
Talk to us!

- **Blog**: [https://cki-project.org/](https://cki-project.org/)
- **E-mail**: cki-project@redhat.com
- **IRC**: #kernelci on Freenode
- **Code**: cki-project organization on GitLab and GitHub
Thank you!

Major Hayden // Red Hat
@majorhayden