Scaling KVM (and its community)
I was told there would be cake.

Milton Waddams
KVM in 2022

6 architectures
300+ files
150,000+ lines of code

150+ contributors
1100+ commits
90,000+ lines of test code
What?

01 Development

02 Maintenance
More contributions means more patches to review and more code to maintain.

03 Validation
More of *everything* means more things that can break.
The first step in solving a problem is recognizing there is one.

Unknown
How?

If KVM were a network...

- More cat videos!
- Delivered faster!
- Less downtime!

1. Latency
2. Efficiency
3. Monitoring
4. Durability
Latency
Ping

Everyone Except Paolo
Efficiency
Queued, thanks.

Paolo Bonzini
Monitoring
404. That’s an error.

The requested URL /kvm_health was not found on this server. That’s all we know.
Durability
Close only counts in horseshoes and hand grenades. **And KVM.**

Frank Robinson
<table>
<thead>
<tr>
<th>Years</th>
<th>Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>3+</td>
<td>Maxim Levitsky, Takuya Yoshikawa (<em>), Marcelo Tosatti (</em>)</td>
</tr>
<tr>
<td>4+</td>
<td>Xiao Guangrong (<em>), Radim Krčmář (</em>), Joerg Roedel (*)</td>
</tr>
<tr>
<td>5+</td>
<td>Wanpeng Li, Sean Christopherson, Vitaly Kuznetsov, Gleb Natapov (*)</td>
</tr>
<tr>
<td>6+</td>
<td>Paolo Bonzini (10), Avi Kivity (<em>), Jan Kiszka (</em>)</td>
</tr>
</tbody>
</table>
No, really. How?
Document

- Processes
  - Key dates, e.g. KVM's effective merge window
  - KVM patch lifecycle
- Expectations
  - KVM "health" requirements
  - Preferred coding style
  - Preferred shortlog scope
- Errata
  - Deviations from architectural specification
Test

- Run existing tests
- Improve coverage of tests
  - Verify tests find bugs
  - Use brute force when possible
- Improve quality of tests
  - Make it easier to develop tests
  - Make it easier to maintain tests

Document
Test
Share
Automate
Adapt
Share

- **Tips and Tricks**
  - Methodologies, scripts, aliases, etc…
  - [https://github.com/sean-jc/settings](https://github.com/sean-jc/settings)

- **Code**
  - Solve common problems once
  - Consolidate code instead of copy+pasting

- **Maintainers**
  - Reduce Paolo’s x86 responsibilities
  - Train next generation on multiple architectures
Automate

- Integration Testing
  - Automated testing of “queued” patches
  - Automated reporting and bisection of failures

- Developer Testing
  - Automated testing of individual series!
  - Manual “automation” as a rough facsimile
Adapt

- Mindset
  - Pursue perfection, not “good enough”
  - Implement to hardware specifications
  - Don’t make assumptions about the guest

- Speak up!
  - Propose process changes
  - Request documentation

1. Document
2. Test
3. Share
4. Automate
5. Adapt
Job is done. Figured it was time for a little chat.

Malcolm Reynolds
Thank You!