Testing a LiveCD

Fabian Deutsch, Aug 2013, FrOSCon

redhat.

http://www.flickr.com/photos/evil_cheese_scientist/4727444050/

Agenda

- Looking at the situation
- Challenges
- The trouble
- The solution
- What's next?

Situation

http://www.flickr.com/photos/autohistorian/8559227692/

Node

- We build oVirt Node
- oVirt Node is a distribution in LiveCD format
  - Firmware-like
  - Simple UI
  - Auto-installation using kargs
- This distribution needs to be tested

Assembly-line

```
for patch in patchqueue:
```

- Review / Gerrit
- Build / Jenkins
  - Do unit-tests
  - Build packages
  - Build ISO
- **Test**

...

http://www.flickr.com/photos/ninithedreamer/5534892178/

Challenges

- Support real-hardware and VMs
- Test installer and setup UI
- Perform auto-installations
- Check package integration
- ...

The Trouble with manual testing

Time consuming

Error prone

Boring

!

“Solutions”

~~Manual testing~~

Test automation

- os-autoinst – VM only
- beaker – kickstart based
- autotest – simple is different

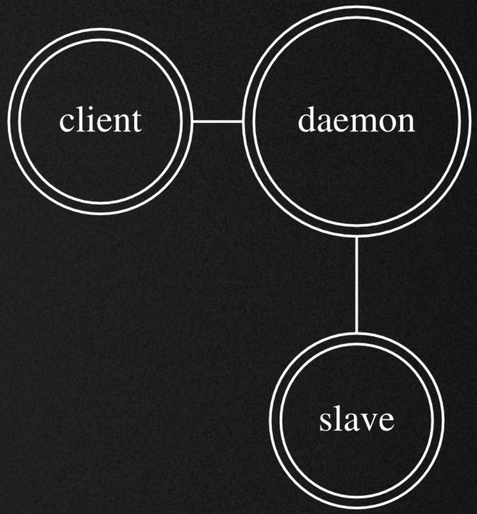
Igor

http://www.flickr.com/photos/jlcuasay/8530722359/

Igor

Runs tests on real hardware and/or VMs

- Daemon: Manages a hosts life-cycle
- Client: Control the daemon
- Slave: Test runner
- Test harness
  - Library functions
  - TUI testing



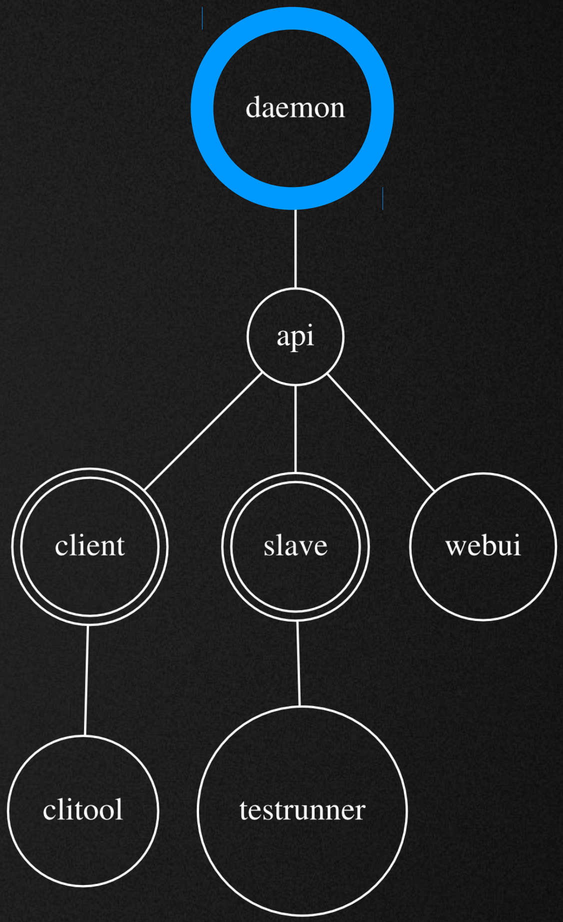
Igor - Demo

$$\begin{tikzcd}
 & \text{Jenkins} \arrow[d] & \\
 & \text{Igor} \arrow[dl] \arrow[dr] & \\
\text{Cobbler} \arrow[d] \arrow[drr] & & \text{libvirtd} \arrow[d] \\
\text{Host} \arrow[uur, dashed, bend left] & & \text{VM} \arrow[uul, dashed, bend right]
\end{tikzcd}$$

http://en.wikipedia.org/wiki/File:NDOR_W15-2.svg

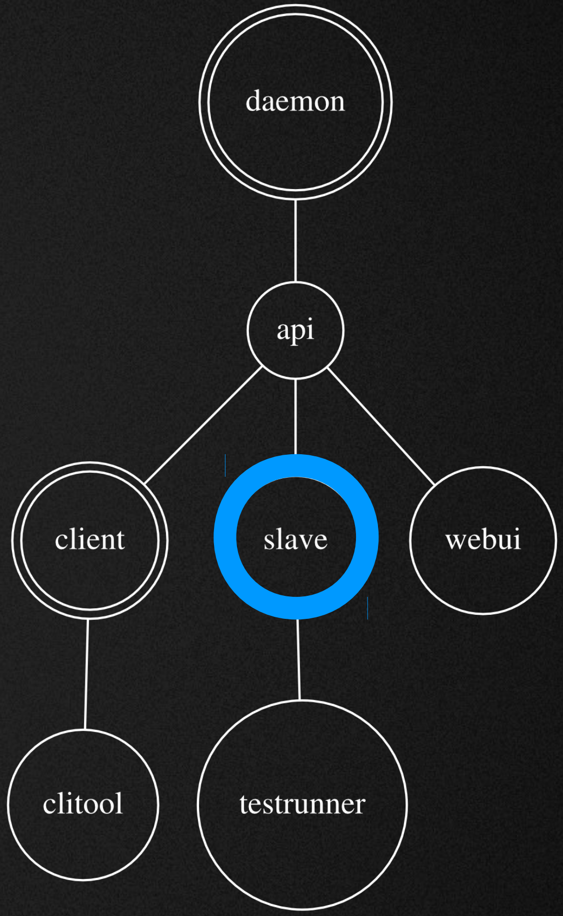
Igor – The daemon

- Manages a host's life-cycle
  - Prepare a host, run tests (tasks), and tear the host down
- Doesn't distinguish between real HW and VM
- HTTP API, web UI, and CLI



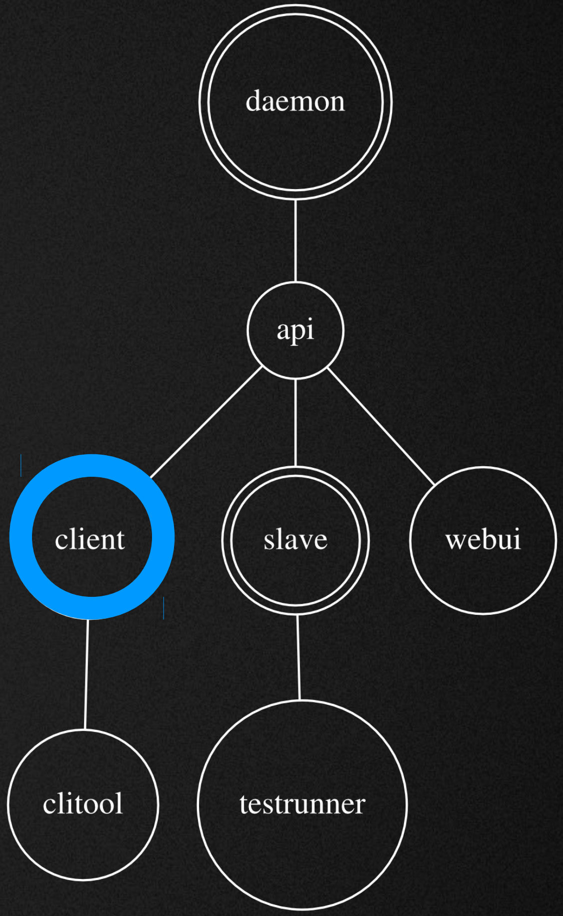
Igor – The slave

- Is running on the system under test
  - HTTP GET test suite from daemon
  - Executes test cases on host
    - Protocol: Success on exit(0)
- Bundles Igor's test harness
  - Upload files, Annotations, ...
  - TUI testing helpers



Igor – The client

- Manage daemon
  - Initiate test runs
- Prepare ISOs
- Provide feedback to daemon
  - Used by Jenkins



Assembly-line w/ Igor

```
for patch in patchqueue:
```

- Review / Gerrit
- Build / Jenkins
  - Do unit-tests
  - Build packages
  - Build ISO
  - Inject igor-slave
- Test / Igor
  - Submit and follow igor testplan
  - Publish results

Developer work-flow

- Edit
- Build
  - Build packages
  - Build image + igor-slave
- Test
  - `$ igorc testplan_on_iso $TESTPLAN $ISO $KARGS`

Relax

http://www.flickr.com/photos/semiautomatic/255348871/

What's next?

- BDD / gherkin as a front-end
  - Specifying test cases is still hard
  - Gherkin is natural language-like, implementation is hidden
- Run Igor as an unprivileged user
- Other backends
  - Beaker for HW and VM?
  - Foreman for HW and VM?
- Improve slave to run tests w/o a connection to the daemon
- Authorization
- ARM?

Resources

- Igor https://github.com/fabiand/igor/
- oVirt Node http://ovirt.org/wiki/Node
  - Gerrit http://gerrit.ovirt.org/p/ovirt-node
  - Jenkins http://jenkins.ovirt.org/view/ovirt_node/

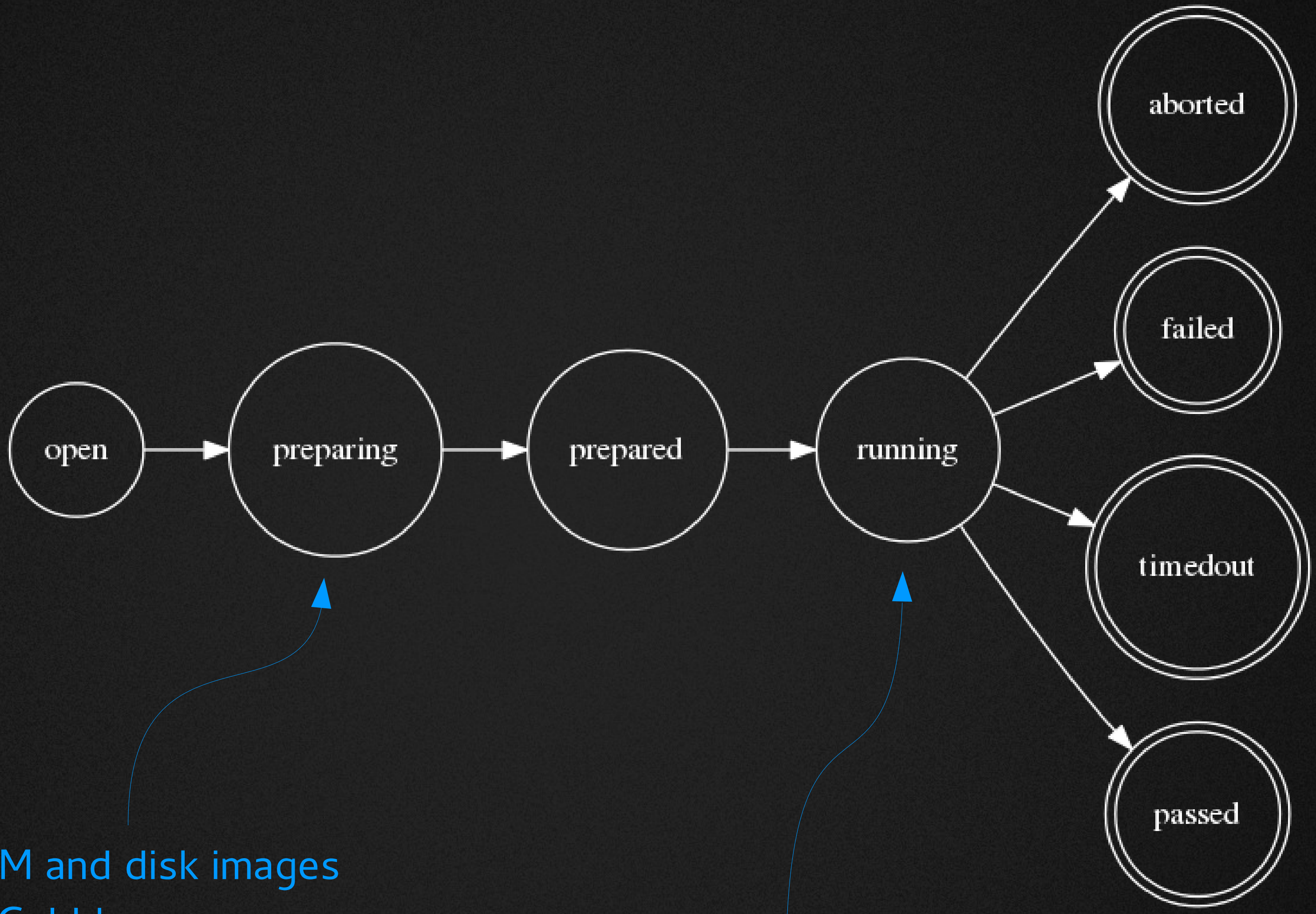
Syntax: Testplan (yaml)

```
---
description: 'AI with {tbd_profile} on VMs'
---
description: 'A basic auto installation without
any TUI testing'
testsuite: 'ai_basic'
profile: '{tbd_profile}'
host: 'default-libvirt'
additional_kargs: 'storage_init BOOTIF=link'
---
```

Gherkin

```
Feature: Auto-Install completes
  In order to ensure working auto-installs,
  As a QA focused developer
  I want Node to do several auto-installs with different kernel arguments


  Scenario: Minimal AI should complete

      Given  a VM with 2GB RAM
        And  4 CPUs
        And  1 20GB disk

       When  The ISO is booted with the kernel arguments
             'BOOTIF=link storage_init'

       Then  the basic test suite must pass


  Scenario: Extended AI should complete
      Given  ...
```

Igor – Daemon states



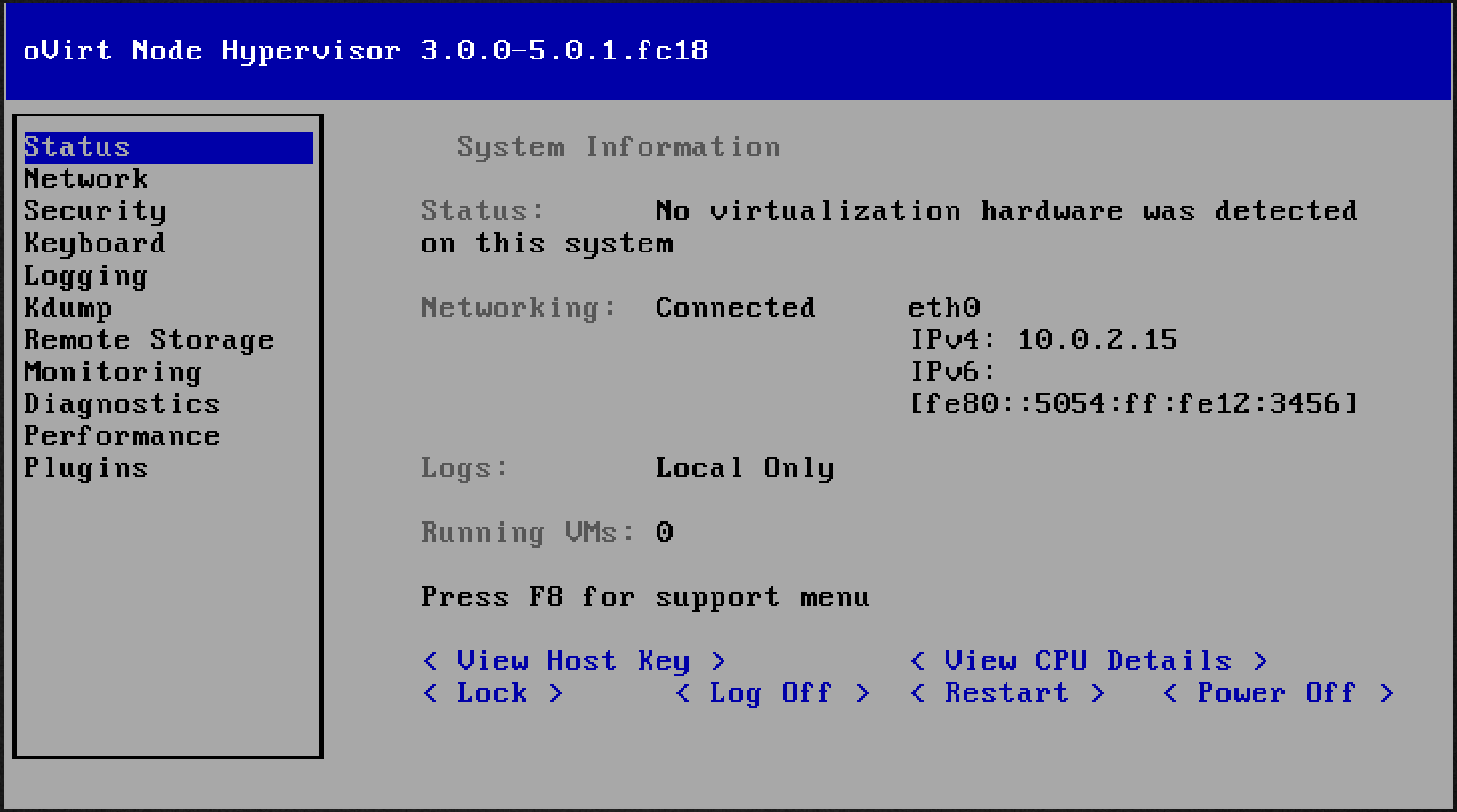
Create the VM and disk images
Push ISO to Cobbler

Daemon awaits results from slave
Communicate with daemon via HTTP

Igor – Stats

- It's a tool.
- ~2 years old
- In production use
- In development
- Packaged in Fedora

oVirt Node

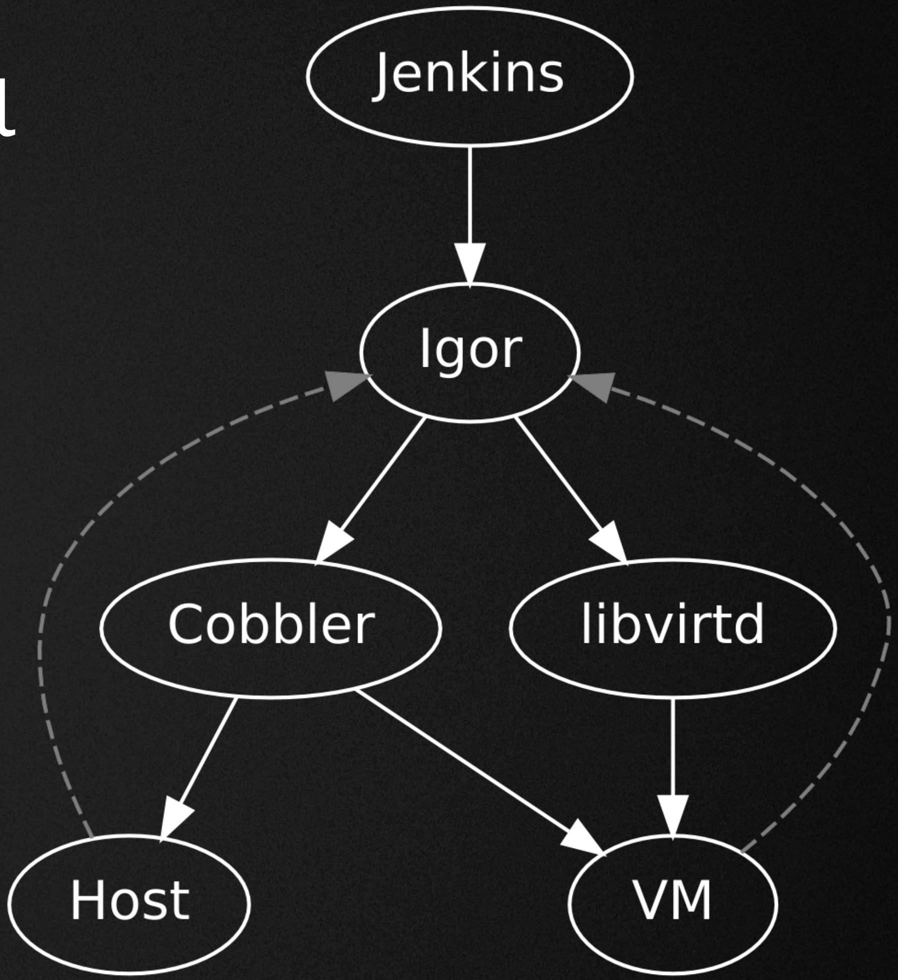


```
oVirt Node Hypervisor 3.0.0-5.0.1.fc18

Status                 System Information
Network
Security             Status:       No virtualization hardware was detected
Keyboard             on this system
Logging
Kdump                Networking:   Connected      eth0
Remote Storage                                    IPv4: 10.0.2.15
Monitoring                                        IPv6:
Diagnostics                                       [fe80::5054:ff:fe12:3456]
Performance
Plugins              Logs:         Local Only

                     Running VMs:  0

                     Press F8 for support menu

                     < View Host Key >            < View CPU Details >
                     < Lock >      < Log Off >    < Restart >   < Power Off >
```

Challenge: HW and VM!

- Keep core logic high-level
- Backends
  - HW: Cobbler and PXE
  - VM: libvirt and PXE

  - VM: libvirt only



Challenge: Kernel Arguments

Or: How not to modify the image

- Image has to be re-packaged to modify the kernel arguments
  - Possibly hides boot-loader problems
  - Limits on PXE

Testing a LiveCD? Igor is doing it.

- or: How to make testing a distribution fun.
- You will be facing different challenges when you try to test a distribution, compared to testing a single software component. This talk is about what challenges appear when testing a distribution, and how to address them.

- The trouble with testing a LiveCD (like oVirt Node) or any other distribution is, that a host is needed before it can be tested. Normally it should not only be tested on one, but on several different hosts (bare-metal or virtual, with SAN or local storage, ...).
- Testing is a boring, time consuming, and an error prone part of the development process, which would profit from any kind of test automation.
- Igor is a tool which was made for this prupose, for testing an OS.
- We'll take a look at how Igor integrates into an existing continuous integration pipeline, consisting of Gerrit and Jenkins.
- More light will also be shed on how Igor takes care of your host's lifecycle, how it runs or installs the LiveCD, and initiates tests.