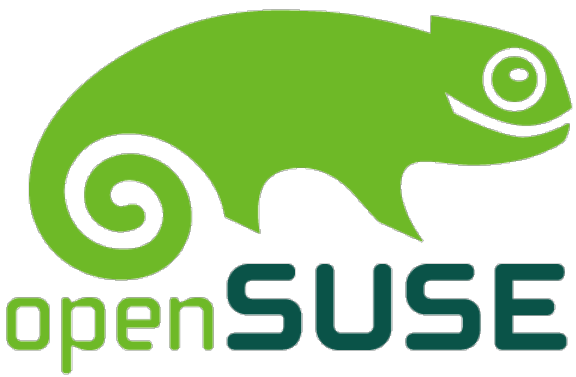
openQA Workshop – oSC13

Learning how to make tests with openQA



Alberto Planas
<aplanas@suse.de>
openSUSE Team

September 3, 2013

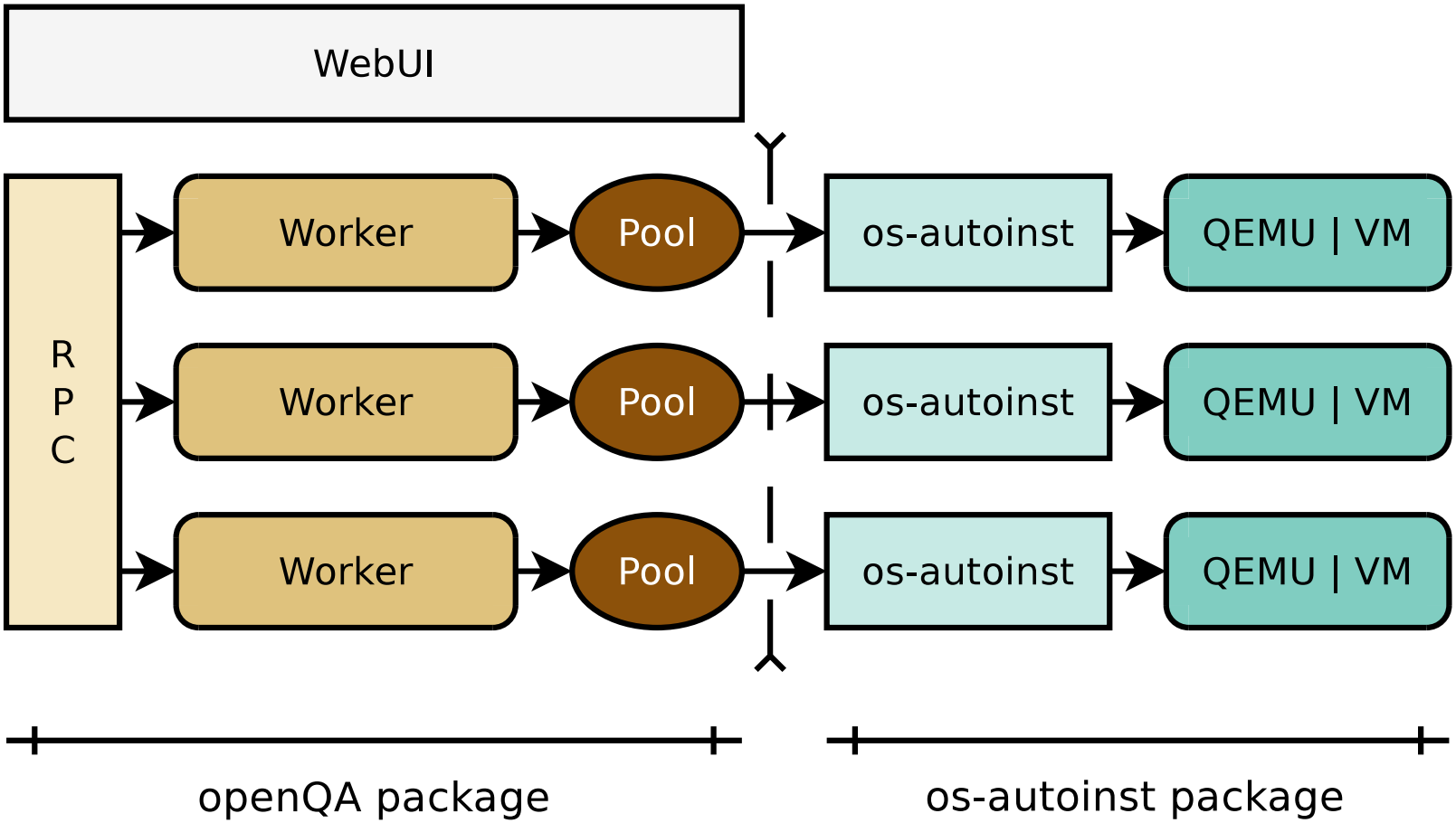
Introduction

Overview

With openQA we can test the installation process of a distribution.

- We provide the tests (Perl code)
- We provide the ISO image
- A worker will launch an instance of os-autoinst
- os-autoinst create the environment
  - Environment variables
  - HDD images
- ... and run the tests

Architecture



Installation

Installation I

Add the devel:openQA repository and install openQA ...

```
OOR=http://download.opensuse.org/repositories
URI=$OOR/devel:openQA/openSUSE_12.3
zypper ar $URI devel:openQA
zypper ref devel:openQA
zypper in openQA kvm OVMF
# Create the pool directory
/usr/lib/os-autoinst/tools/preparepool 2
reboot
```

... install needles ...

```
/usr/lib/os-autoinst/tools/fetchneedles
cd /var/lib/os-autoinst/needles
sudo chown -R wwwrun distri
```

Installation II

... and configure apache.

```
zypper in apache
cd /etc/apache2/vhosts.d
cp openqa.conf.template openqa.conf
# Edit openqa.conf
/usr/sbin/a2enmod rewrite
/usr/sbin/a2enmod headers
systemctl restart apache2.service
```

Installation III Extra

If we want to run openQA in console and interactive mode.

```
zypper in python-tk
zypper in python-imaging
```

But this is optional and a bit deprecated.

Test the Installation

Download the ISO from
`http://download.opensuse.org/factory/iso/`

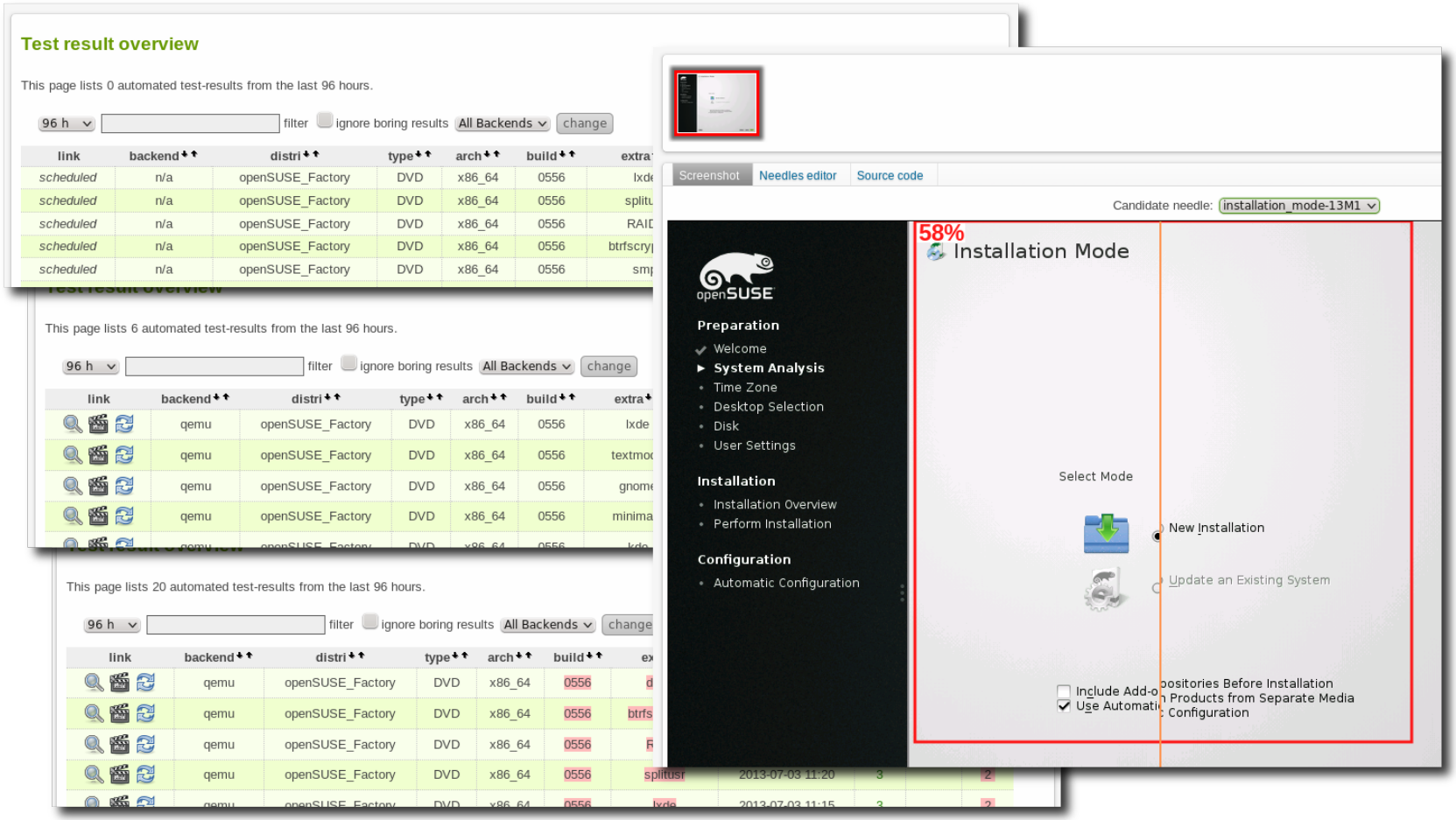
Register the ISO ...

```
cp $ISO /var/lib/openqa/factory/iso
export PATH=$PATH:/usr/share/openqa/tools
rpc.pl --host localhost iso_new $ISO
```

... and start the workers.

```
systemctl start openqa-worker.target
# You can control every worker manually:
# systemctl start openqa-worker@1.service
```

Screenshot


Test result overview
This page lists 0 automated test-results from the last 96 hours.
96 h
filter
ignore boring results
All Backends
change
link
backend
distri
type
arch
build
scheduled
n/a
openSUSE_Factory
DVD
x86_64
0556
This page lists 6 automated test-results from the last 96 hours.
qemu
lxde
This page lists 20 automated test-results from the last 96 hours.
Screenshot
Needles editor
Source code
Candidate needle:
installation_mode-13M1
58%
Installation Mode
openSUSE
Preparation
Welcome
System Analysis
Time Zone
Desktop Selection
Disk
User Settings
Installation
Installation Overview
Perform Installation
Configuration
Automatic Configuration
Select Mode
New Installation
Update an Existing System

openQA Usage

Running a Test

- Go to the http://localhost/
- Cancel all the test except KDE
- Launch in a terminal a new worker

```
worker --host localhost --instance 1
```

- Refresh and see the test running
- You can connect via VNC

```
vncviewer localhost:91
```

Exercise 1 – Testing an ISO

1. Use Build0575 to test a good ISO
2. Register the ISO in the system
3. Remove all jobs except the KDE one
4. Launch only one worker
5. Wait until fails!

Test Failed

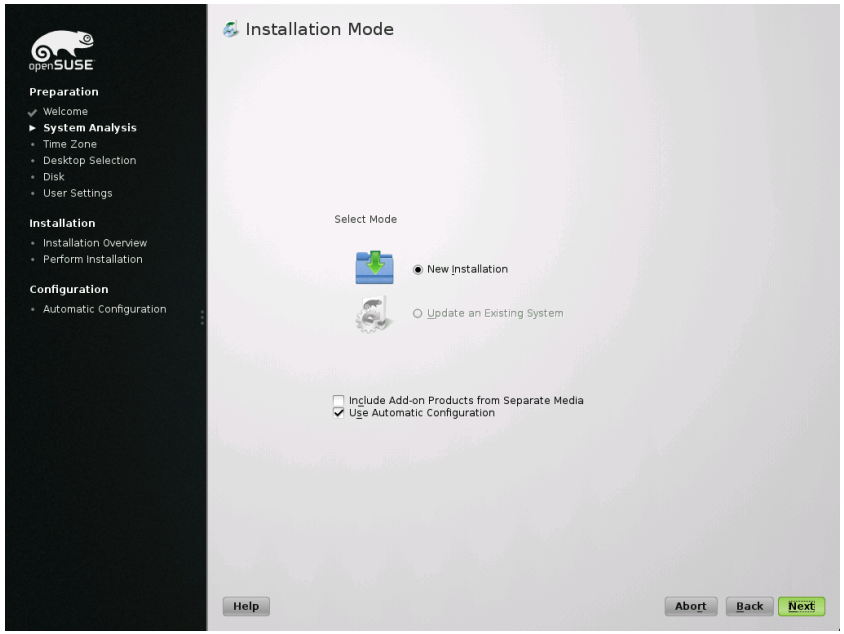
So... the test failed. Let see what happen.

- Go to the result view
- Select the failed test
- Use the screenshot to check the image
- A real bug in Factory?
- If not, create a needle!

The Needle

A needle is a PNG image and a metadata in JSON

```
{
  "area": [
    {
      "width": 514,
      "xpos": 255,
      "type": "match",
      "ypos": 0,
      "height": 538
    }
  ],
  "tags": [
    "inst-instmode"
  ]
}
```



Exercise 2 – Create a Needle

1. Go to the result page
2. Create a needle in the failing test
3. Be careful with the TAGs!
4. Relaunch the job and repeat the operation if needed

openQA API

Exercise 3 – Analyze a Test

1. Use rpm -ql os-autoinst to find where are the tests
2. Find the opensuse one and see the directories
3. Read the ooffice test and figure out what is doing

Variables

Check the variables to see in what configuration are you

- DESKTOP = kde | gnome | lxde | minimalx ...
- USBBOOT | LIVETEST | NETBOOT
- BTRFS | ENCRYPT | LVM | RAIDLEVEL
- UEFI

API for Input

Sending events to the VM

- `sendkey "alt-n";` – Send a keystroke
- `sendkey $cmd{"next"};` – Use the `$cmd{}` map for shortcuts
- `sendautotype "string";` – Send a set of keys
- `sendautotype("string", 3);`

API for Needles

Sending events to the VM

- `waitforneedle("tag", 1);` – Assert for a needle that have this tag
- `checkneedle("tag", 1);` – Return true is the needle is present
- `$self->check_screen;` – Assert using a synthetic tag name (`test-$testname-$count`)
- `$self->take_screenshot;` – Do not assert. Journal for the test

Directory layout

Two important directories:

- Tests: /usr/lib/os-autoinst/distri/opensuse
  - Installation tests: inst.d
  - Console tests: consoletest.d
  - Application tests: x11test.d
- Needles: /usr/lib/os-autoinst/distri/opensuse/needles

Anatomy of a Test I

```
use base "basetest";
use strict;
use bmwqemu;

# Determine, using the $ENV variables, if the
# test can be selected for this configuration.
sub is_applicable() {
}

# Main code of the test
sub run() {
}
```

Anatomy of a Test II

```
# Return a map of flags to decide if the fail
# of this test is important, or to decide a
# rollback of the VM status.
sub test_flags() {
}

1;
```

Test Flags

With the tests flags we control the behavior of the test if it fails.

- `{ 'fatal'=>1 }` – If fails, the test suite fails
- `{ 'important'=>1 }` – If fails, the overall state fails. Factory is broken
- `{ 'milestone'=>1 }` – If ok, generate a new 'lastgood' snapshot
- `{ }` – If fails, recover the 'lastgood' snapshot and continue to the next test

Exercise 4 – A test for Okular

1. Create a test for okular
2. ... or evince

A test for Okular

This is an application test

```
use base "basetest";
use strict;
use bmwqemu;

sub is_applicable() {
  return $ENV{DESKTOP} eq "kde";
}

sub run() {
  my $self=shift;
  sendkey "alt-f2"; sendautotype "okular";
  sendkey "ret"; waitforneedle("okular", 5);
  sendkey "alt-f4"; sendkey "ret";
}

1;
```

A test for Okular – Sleeping

But is not working!! – We need to sleep.

```
[...]
sub run() {
  my $self=shift;
  sendkey "alt-f2"; sleep 2;
  sendautotype "okular"; sleep 2;
  sendkey "ret"; sleep 2;
  waitforneedle("okular", 5);
  sendkey "alt-f4"; sleep 2;
  sendkey "ret";
}
[...]
```

API to Run Programs

We can hide the sleep command... a bit.

- `script_run("program");` – Run the program from a terminal
- `script_sudo("program");` – Run the program as root
- `x11_start_program("program");` – You have implemented that

Exercise 5 – The sudo Case

1. From time to time we want to run a program with sudo
2. sudo can ask for a password, sometimes
3. Figure out how to resolve this problem with the current API
4. If you are out of ideas, find the implementation and try to understand it

The sudo Case

With checkneedle we can add state into a test. This state is needed to check when the terminal is asking for the root password.

```
sendautotype("sudo␣ls\n");
if (checkneedle("sudo-prompt", 2)) {
  sendautotype("p4ssw0rd");
  sendkey "ret";
}
```

A Complex Case

How to resolve when a dialog box appears during the installation process?

```
my @tags = (@{needle::tags("good-needle")},
            @{needle::tags("bad-needle")});
my $err = 0;
while (1) {
  my $ret = waitforneedle(\@tags, 200);
  last if $ret->{needle}->has_tag("good-needle");
  $self->take_screenshot;
  sendkey "ret";
  $err = 1;
}
mydie if $err;
```

Advanced API

- `qemusend "command";` – Send QEMU commands directly
- `makesnapshot "name";` – Create a VM snapshot
- `loadsnapshot "name";` – Recover a VM snapshot
- `mouse_[move|set|click](x, y);` – Control the mouse
- `mouse_hide;` – Hide the mouse
- `waitserial "regexp";` – Result 1 if found in the serial port

Endnote

Suse is hiring

Join the Geeko!

Learn more about SUSE's openings, globally:

1. Talk to our colleagues at the booth
2. Check out our careers page www.suse.com/careers
3. Contact our recruiting team at jobs@suse.com

Thanks

Thank you for your attention.