# openQA Workshop – oSC14

## Learning how to make tests with openQA

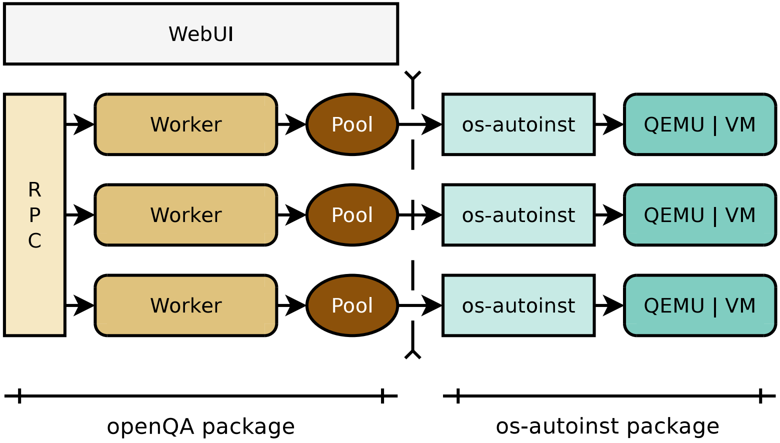
Alberto Planas,
Ludwig Nussel

SUSE Linux Products GmbH

April 27, 2014

## Introduction

## Architecture



## Overview

In this workshop we will ...

- start with testing a small live distro
- learn how to create or modify reference images
- learn how to create tests

## Installation

## Installation

- install packages

```
zypper ar -f obs://devel:openQA/openSUSE_13.1 openQA
zypper ar -f obs://devel:openQA:13.1/openSUSE_13.1 \
  openQA-perl-modules
zypper in openQA apache2
```

- start web interface

```
systemctl start openqa-webui
```

- More detailed instructions at
  https://github.com/os-autoinst/openQA

## Apache Setup

- use default vhost template

```
cp /etc/apache2/vhosts.d/openqa.conf.template \
   /etc/apache2/vhosts.d/openqa.conf
```

- enable required apache modules

```
a2enmod headers
a2enmod proxy
a2enmod proxy_http
```

- (re)start apache

```
rcapache2 restart
```

## Generate Secrets for Authentication

- switch off https in `/etc/openqa/openqa.ini`

  ```
  [openid]
  httpsonly = 0
  ```

- go to `http://localhost/` and log in
- go to Admin -> Secrets and generate a pair of key+secret
- edit `/etc/openqa/client.conf` and put key and secret there

## openQA Usage

## Pick some small distro to practice with

- download SliTaz

```
wget -P /var/lib/openqa/factory/iso \
  http://mirror.slitaz.org/iso/4.0/ \
  slitaz-4.0.iso
```

- clone template repo for distro

```
cd /var/lib/os-autoinst/tests
sudo -u geekotest \
  git clone git://github.com/os-autoinst/ \
  os-autoinst-distri-example.git slitaz
ln -s /var/lib/os-autoinst/tests/slitaz \
  /usr/lib/os-autoinst/distri/
```

## Exercise 1 – Testing an ISO

1. Create a job for the ISO

   ```
   /usr/share/openqa/script/client jobs post \
     DISTRI=slitaz VERSION=4.0 ARCH=i586 \
     TEST=foo MACHINE=bar NICMODEL=e1000 \
     DESKTOP=default ISO=slitaz-4.0.iso
   ```

2. Launch one worker

   ```
   systemctl start openqa-worker@1.service
   ```

3. Watch in the web UI live until it fails!

## Test Failed

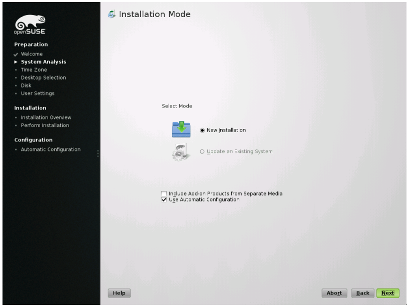
So... the test failed. Let's see what happened.

- Go to the result view
- Click on the thumbnail of the failed test
- Check the screenshot
- Check the "Source code" tab to see the test code

## 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"
  ]
}
```



● New Installation
○ Update an Existing System

☐ Include Add-on Products from Separate Media
✓ Use Automatic Configuration

## Exercise 2 – Create a Needle

1. Restart the test and open the new job
2. Set the job to interactive mode
3. Stop the test when it's at the bootloader
4. Create a needle in the failing test
5. Be careful with the Tags!

## openQA API

## Exercise 3 – Modify a Test

1. Tests are in `/var/lib/os-autoinst/tests`
2. Find the slitaz ones and have a look at the directories
   - Test driver: `main.pm`
   - Actual tests: `test.d`
   - Needles: `needles`
3. Open the boot test, figure out what is doing and modify it

## Modified Test

```
+       waitforneedle( "language", 30 );
+       sendkey "ret";
+
+       waitforneedle( "keyboard", 30 );
+       sendkey "ret";
+
+       waitforneedle( "keyboard_text", 30 );
+       sendkey "ret";
```

## 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() {
}
```

## Variables

Typcial variables available in openSUSE

- DESKTOP = kde | gnome | lxde | minimalx ...
- DISTRI | VERSION | ARCH | TEST | MACHINE
- 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 needle with this tag
- `checkneedle("tag", 1);` – Return needle if needle found
- `$self->check_screen;` – Assert using a synthetic tag name (`test-$testname-$count`)
- `$self->take_screenshot;` – Do not assert. Journal for the test

## Anatomy of a Test II

```perl
# 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 stops in failed state
- `{ 'important'=>1 }` – If fails, the overall state fails. ISO considered 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 nano

Problem: SliTaz has not default keyboard bindings

1. Create a test for nano
2. switch to text console
3. log in as root (password root)
4. launch nano
5. exit it again

## The test for nano

```
sub run {
    sendkey "ctrl-alt-f1";
    waitforneedle( "console", 30 );
    sendautotype "root\n";
    sendautotype "root\n";
    sendautotype "nano\n";
    waitforneedle( "nano", 30 );
}
```

## 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 – install a package

1. install the sudo package (`tazpkg get-install sudo`)
2. exit the shell
3. log in as tux (no password)

## Exercise 5 – The sudo Case

1. From time to time we want to run a program with sudo
2. sudo sometimes asks but not always
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

Solution: checkneedle

```
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

Questions?

## 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.