

Automatisiertes Testen von Software

Rainer König
SUSE



Warum testen wir Software?



- Um die Fehlerfreiheit zu beweisen
- Um Fehler zu finden



Build & QA-Prozess



Welcome to openQA
Life is too short for manual testing!
[Learn more »](#)

openSUSE Tumbleweed

- Build20230329 (about 8 hours ago)
234 passed 31 softfailed 35 failed
- Build20230328 (about 20 hours ago)
251 passed 32 softfailed 37 failed
- Build20230327 (2 days ago)
237 passed 31 softfailed 33 failed

Bugzilla - Main Page

Home New Browse Search [Search] Reports rainer.koenig@suse.com My Saved Searches

Product Dashboard New Plan New Case New Run Search Help --Testopia-- Find

Important notice:
Bugzilla service will be unavailable for approximately 8h due to upgrade starting from April, 12 2023 8:00 CEST
Please, plan accordingly and we apologize for any inconvenience this may cause.

Welcome to Bugzilla

[Get Help](#) [File a Bug](#) [Search](#) [User Preferences](#)

Progress Diary Meine Seite Projekte Hilfe rainer.koenig Mein Konto Abmelden

QA / openQA Project / openQA Tests / qe-yam

Übersicht Aktivität Roadmap Tickets Gantt-Diagramm **Agiles Projektmanagement** Kalender Dokumente Wiki Konfiguration

Agiles Taskboard Suche nach Thema Diagramm

Filter

- Tracker list action
- Zielversion list qe-yam - Current

Filter hinzufügen

Optionen

Anwenden Zurücksetzen Speichern Vollbild

New (11)	Workable (14)	In Progress (15)	Blocked (0)	Feedback (0)	Closed (170)
action #120450 Call	action #125717 [ppredic] test	action #126125 As disk space	+ NEUES TICKET	+ NEUES TICKET	action #96155 Improve YAML schedule

Mitglieder

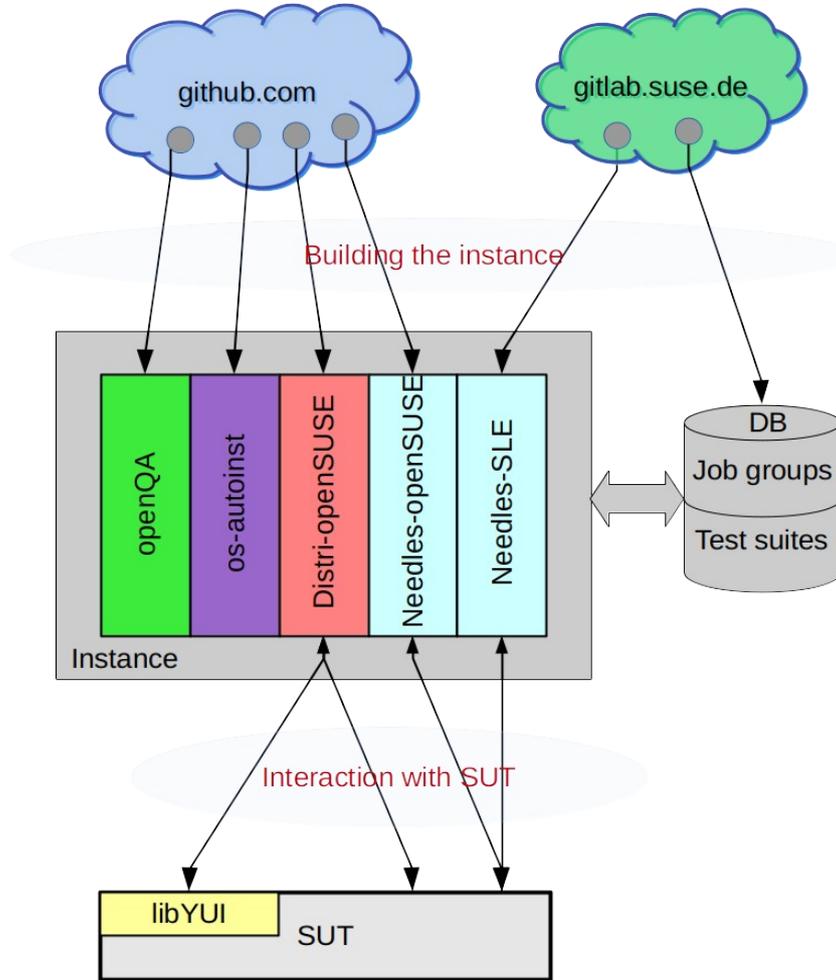
- a_fuerber
- acarvajal
- akervesato
- AdaLowelace
- AdamWill
- agnies

Begriffe

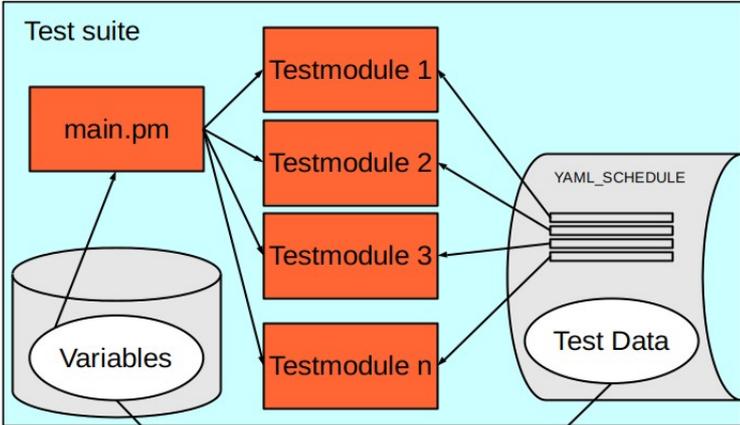
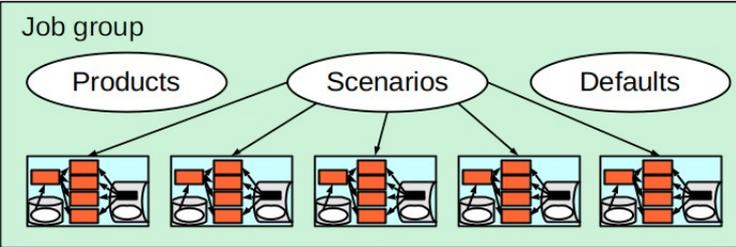
- Projekt: openSUSE Tumbleweed, SLE, ALP
- Testsuite: Ein spezifischer Test (Plan)
- Testmodul: Ein Test-Schritt in der Testsuite
- SUT: System under Test
- Worker: Interaktion mit dem SUT



openQA overview



openQA hierarchy

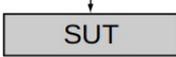


controls

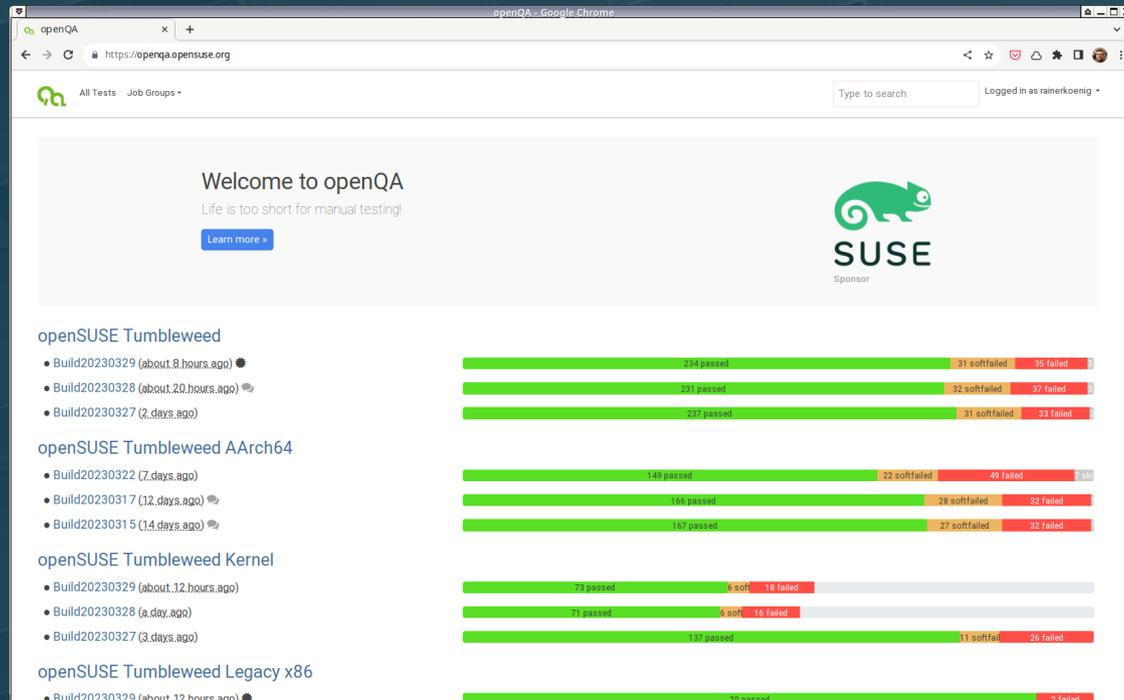
specifies



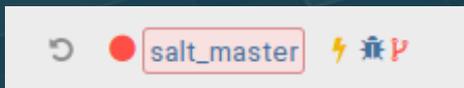
Interacts



Webbasiertes System



Build-Review Übersicht



openQA: Test summary

https://openqa.opensuse.org/tests/overview?distri=microos&distri=opensuse&version=Tumbleweed&build=20230330&groupid=1

All Tests Job Groups

Type to search Logged in as rainerkoenig

Test result overview

Overall Summary of openSUSE Tumbleweed build 20230330 showing latest jobs, overview fixed to the current time

Passed: 235 Incomplete: 2 Soft-Failed: 22 Failed: 25 Running: 3 Aborted: 4

Filter no filter present, click to toggle filter form

Distri: microos / Version: Tumbleweed / Flavor: DVD

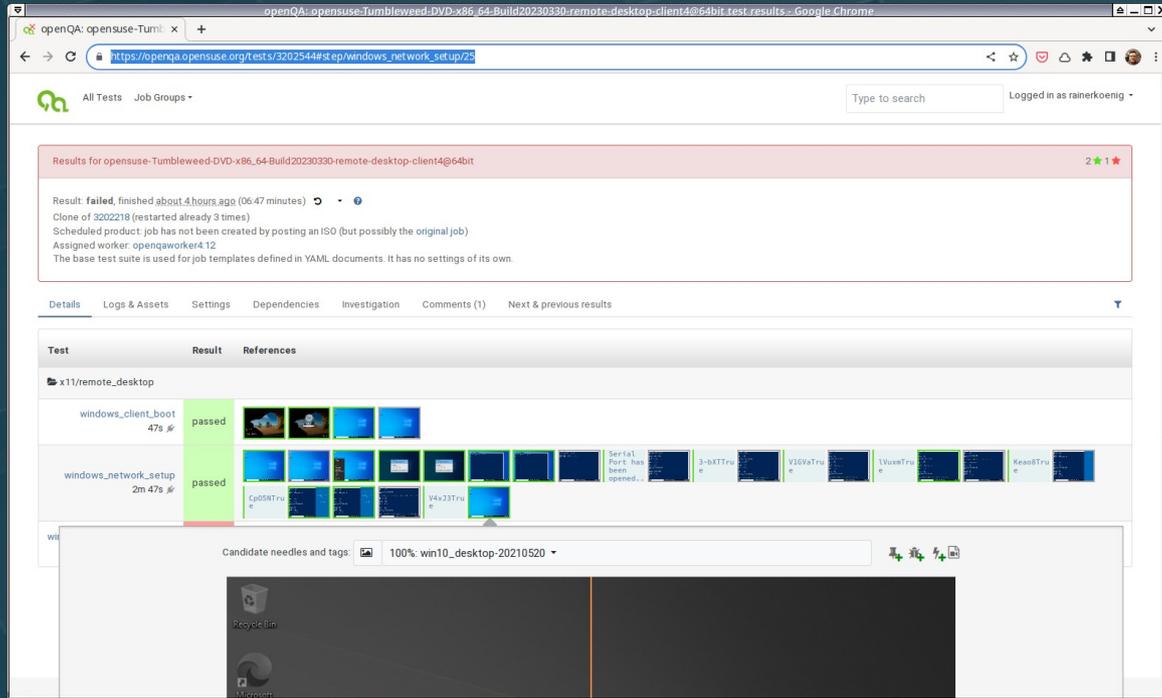
Test	x86_64
container-host	⌵ ●
microos	⌵ ●
microos@uefi	⌵ ●
microos_10G-disk	⌵ ●
microos_textmode	⌵ ●
rcshell	⌵ ●
remote_ssh_controller	⌵ ● P

Distri: microos / Version: Tumbleweed / Flavor: MicroOS-Image

Test	x86_64
microos	⌵ ●
microos-combustion	⌵ ●



Eine Testsuite im Detail



The screenshot displays the openQA web interface in a Google Chrome browser. The address bar shows the URL https://openqa.opensuse.org/tests/3202544#step/windows_network_setup/2. The page header includes the openQA logo, navigation links for "All Tests" and "Job Groups", a search bar, and the user "Logged in as rainerkoenig".

The main content area shows the results for the test "openSUSE-Tumbleweed-DVD-x86_64-Build20230330-remote-desktop-client4@64bit". The overall result is "failed", with a message indicating it finished about 4 hours ago (06:47 minutes) and was a clone of job 3202218 that has been restarted 3 times. Below this, a table lists the individual test steps:

Test	Result	References
x11/remote_desktop		
windows_client_boot 47s	passed	
windows_network_setup 2m 47s	passed	

At the bottom, a "Candidate needs and tags" section shows "100% win10_desktop-20210520" and a thumbnail of a Windows 10 desktop environment.

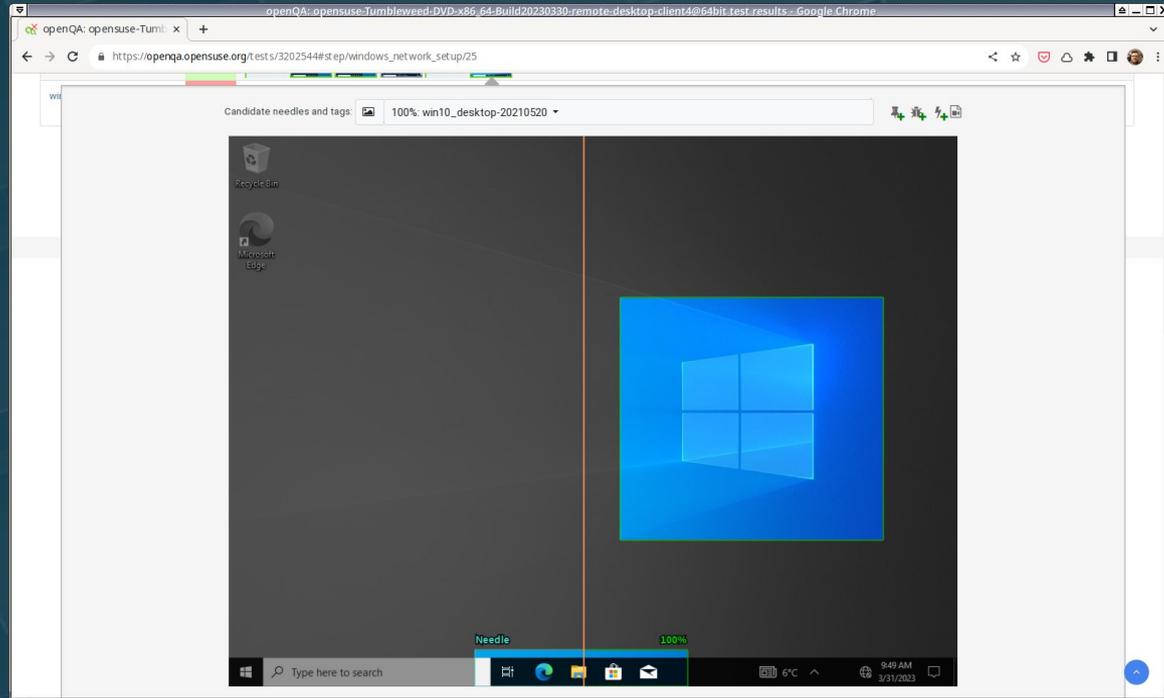
Testmodul Quelltext

```
openQA: windows_netwo: x | +
openQA: windows_network_setup - Google Chrome
https://openqa.opensuse.org/tests/3202544/modules/windows_network_setup/steps/1/src
All Tests Job Groups -
Type to search Logged in as rainerkoenig -

Test module: /var/lib/openqa/share/tests/opensuse/tests/x11/remote_desktop/windows_network_setup.pm
1 # SUSE's openQA tests
2 #
3 # Copyright 2018-2021 SUSE LLC
4 # SPDX-License-Identifier: FSFAP
5 #
6 # Summary: Remote Login: Windows access openSUSE/SLE over RDP
7 # Maintainer: GraceWang <gwangsuse.com>
8 # Tags: tcr1610388
9
10 use Mojo::Base qw(windowsbasetest);
11 use TestAPI;
12
13 sub approve_network_popup {
14     # if there is the networks popup that asks if it's ok to be discoverable, approve it
15     wait_still_screen stilltime => 5, timeout => 10;
16     assert_screen([qw(networks-popup-be-discoverable no-network-discover-popup)], 60);
17     if (match_has_tag 'networks-popup-be-discoverable') {
18         assert_and_click 'network-discover-yes';
19         wait_screen_change(sub { send_key 'ret' }, 10);
20     }
21     # We may miss some characters due to type command too fast after network popup
22     # so wait several seconds as a workaround
23     # See poo#109091
24     wait_still_screen 3;
25 }
26
27 sub run {
28     my $self = shift;
29
30     assert_screen 'windows-desktop';
31     $self->open_powershell_as_admin;
32     $self->run_in_powershell(cmd => 'Get-NetIPAddress', tags => 'win-remote-desktop');
33     $self->run_in_powershell(cmd => '$a = Get-NetAdapter -Name "Ethernet" ; echo $a.name', tags => 'win-remote-desktop');
34     $self->run_in_powershell(cmd => 'Set-NetIPInterface -InterfaceAlias $a.name -Dhcp Disabled', tags => 'win-remote-desktop');
35     approve_network_popup;
36     $self->run_in_powershell(cmd => 'Get-NetIPAddress -InterfaceAlias $a.name', tags => 'win-remote-desktop');
37     $self->run_in_powershell(cmd => 'New-NetIPAddress -InterfaceAlias $a.name -IPAddress 10.0.2.18 -PrefixLength 24 -
DefaultGateway 10.0.2.2 -Confirm:$false', tags => 'win-remote-desktop');
38     approve_network_popup;

```

Eine Needle im Detail



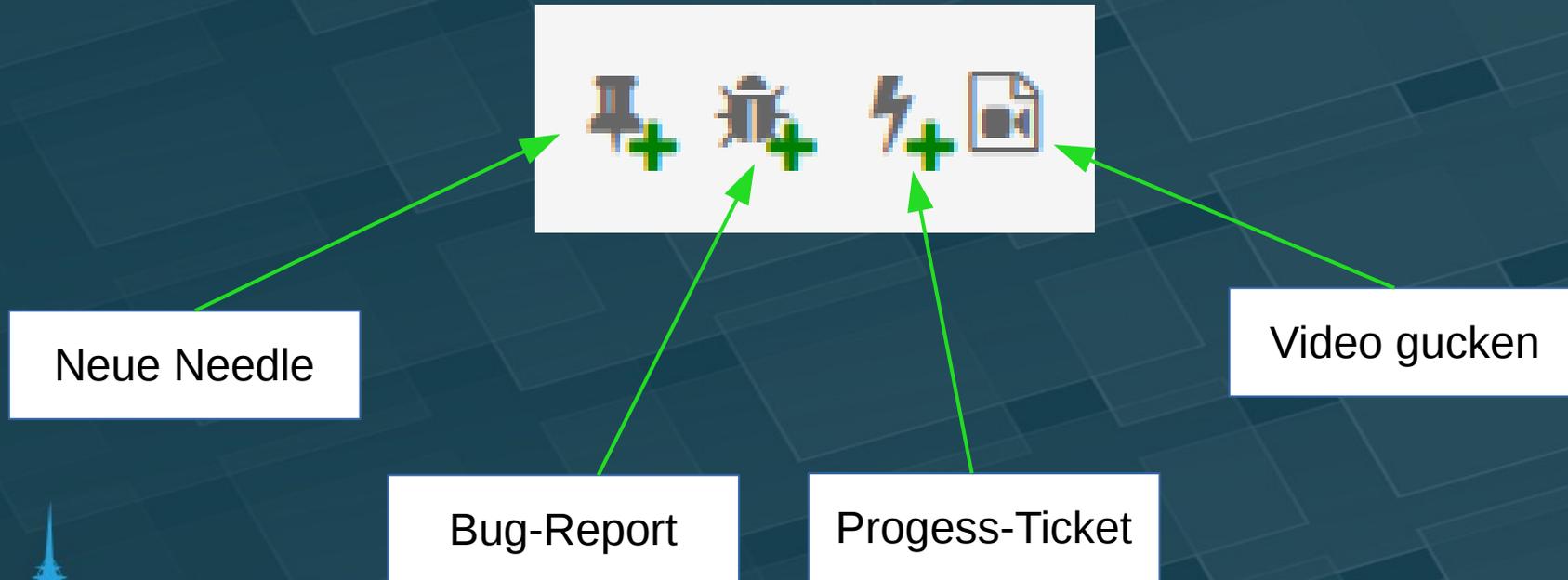
Anatomie eines Fehlers

The screenshot shows a browser window displaying a test result from openQA. The test is titled "0%: boot_encrypt-encrypted-disk-password-prompt-20191112". The test progress bar shows a red segment, indicating a failure. The main content area shows a video frame of a Linux desktop environment. The desktop has a dark theme and a "Welcome" dialog box open. The dialog box contains the text "Ahoy, this is openSUSE" and several buttons for "Read me", "Documentation", "Get Software", "GNOME Help", "Contribute", and "Build openSUSE". A "Please enter password" dialog box is overlaid on top of the welcome screen, with a red box highlighting the input field. The system clock shows "Mar 31 09:47".

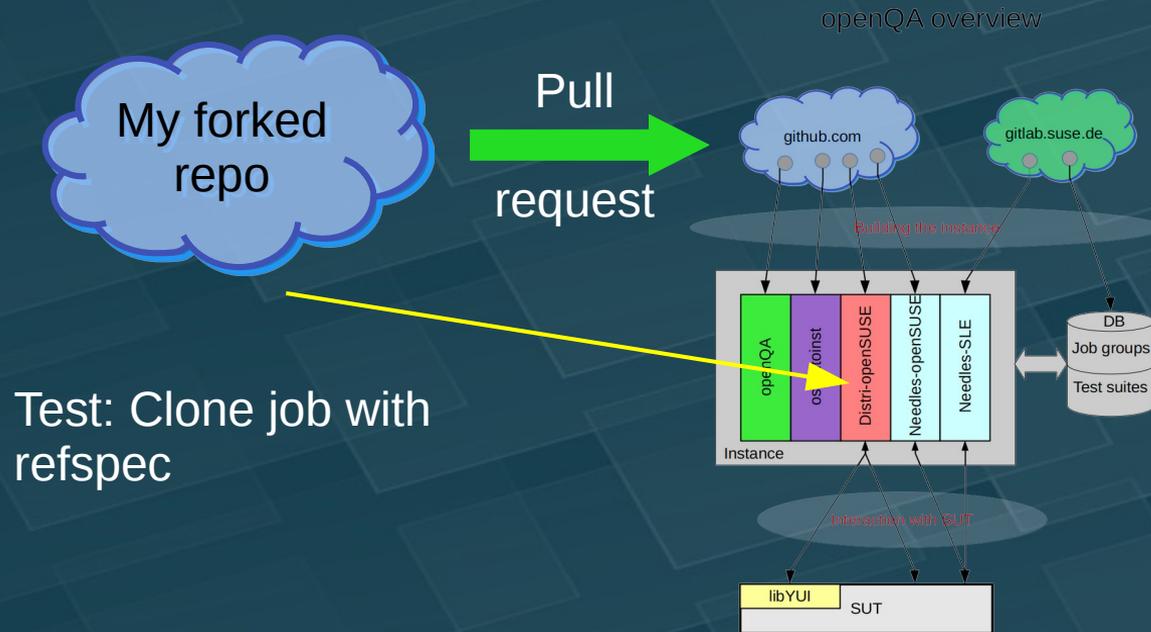
Candidate needles and tags: 0%: boot_encrypt-encrypted-disk-password-prompt-20191112

Test died: no candidate needle with tag(s) 'encrypted-disk-password-prompt' matched

Oh, ein Fehler. Was nun?



OpenQA Entwicklung



Danke für die Aufmerksamkeit.

Fragen?



Automatisiertes Testen von Software

Rainer König
SUSE



Rainer König:

- Studium zum Diplom-Informatiker an der Hochschule von 1980-1984
- 17 Jahre der „Mr. Linux-Desktop“ bei Fujitsu
- seit August 2021 bei SUSE als QE Autoation Engineer

Kontakt: rainer@koenig-haunstetten.de

Warum testen wir Software?



- Um die Fehlerfreiheit zu beweisen
- Um Fehler zu finden



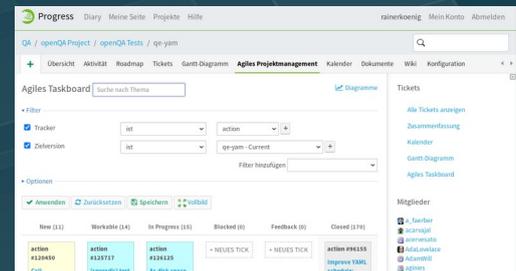
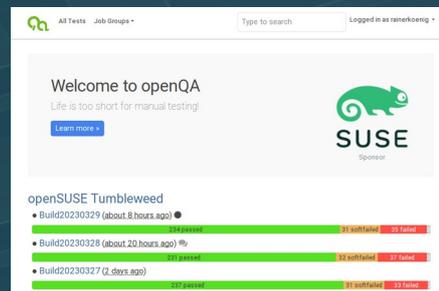
Testen kann NIE die Fehlerfreiheit beweisen.

Das Ziel von Tests ist es, Fehler zu finden.

Build & QA-Prozess



triggert



OBS ist der Build-Service um Programmpakete oder ganze Distributionen zu bauen.

Fertige Produkte von OBS werden mit openQA synchronisiert und „triggern“ dort den Start der Tests.

Tests werden dann von Test Engineers „reviewed“ und bei Fehlern werden Bugs in Bugzilla erstellt oder Tickets in Progress. OpenQA bietet hier auch direkte Verbindungen um „copy&paste“ zu minimieren.

Begriffe

- Projekt: openSUSE Tumbleweed, SLE, ALP
- Testsuite: Ein spezifischer Test (Plan)
- Testmodul: Ein Test-Schritt in der Testsuite
- SUT: System under Test
- Worker: Interaktion mit dem SUT



Definition der wichtigsten Begriffe im Tester-Jargon

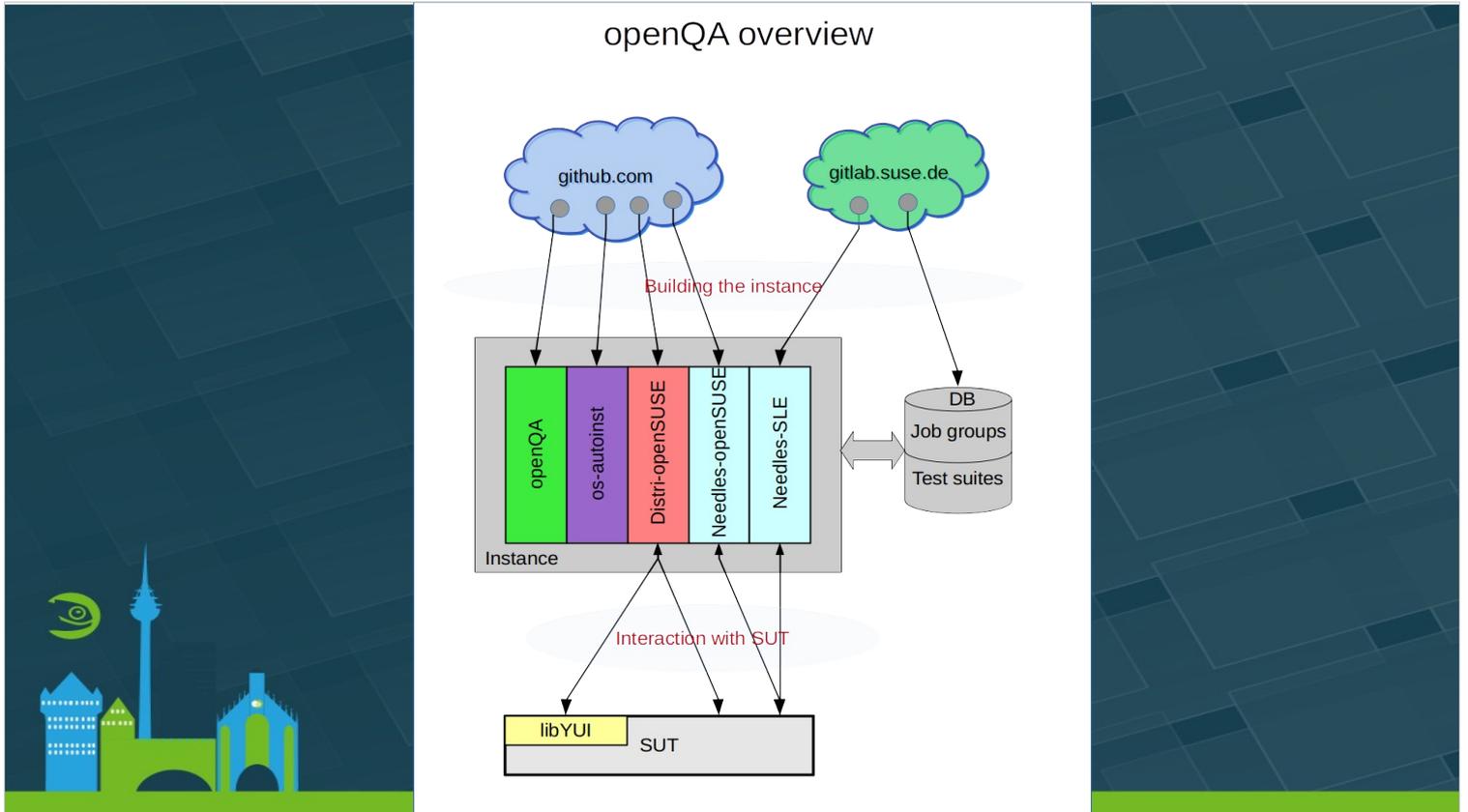


Schaubild wie sich eine Instanz von openQA aus Entwicklersicht darstellt.

Der Code ist in GitHub, Dinge, die für die Enterprise Version SLE relevant sind (Needles oder Registration Codes) liegen auf GitLab im nicht-öffentlichen Firmen-VPN.

Eine Instanz nutzt Code von openQA, os-autoinst, os-autoinst-distri-opensuse und die needles.

Die eigentlichen Testmodule liegen in os-autoinst-distri-opensuse.

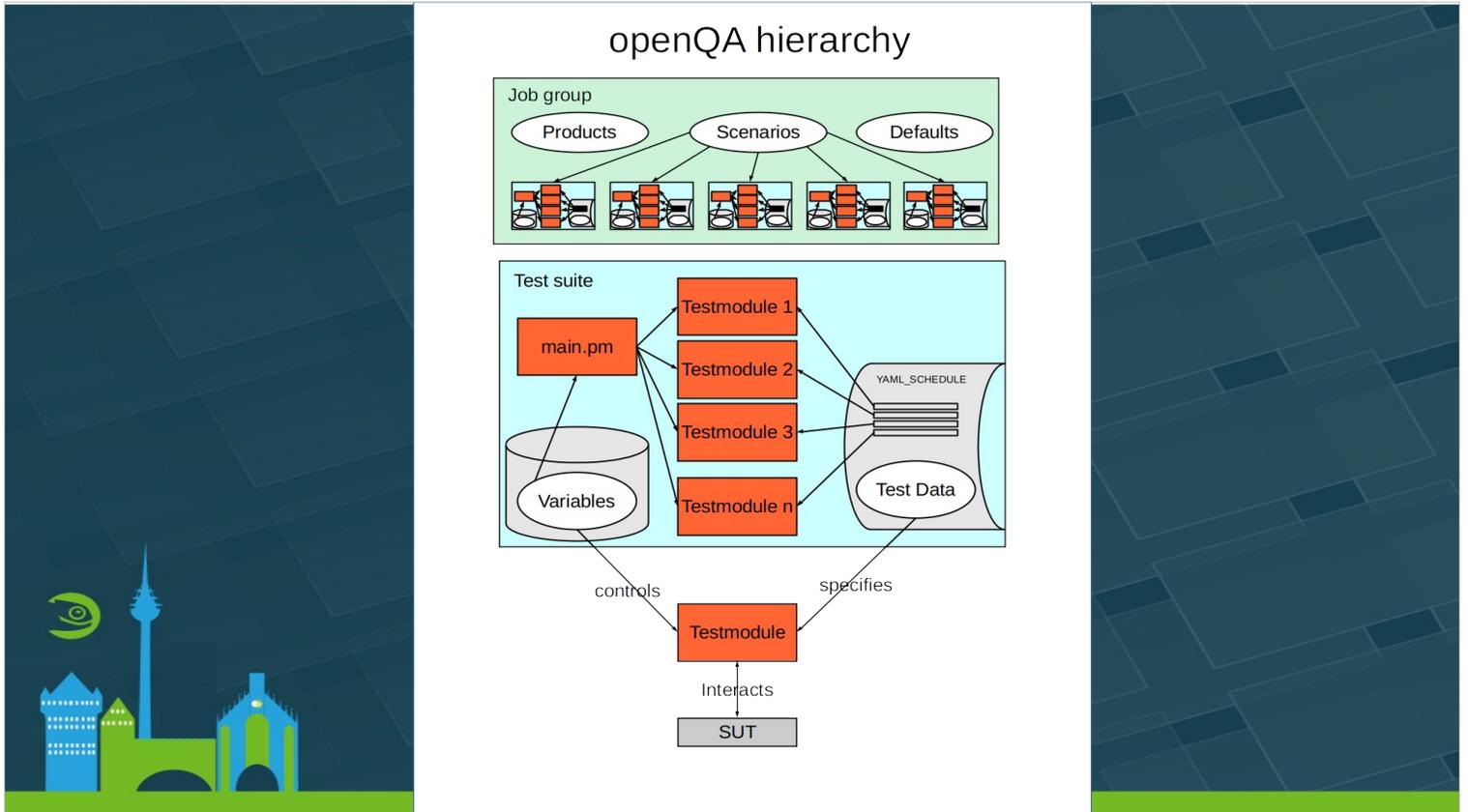
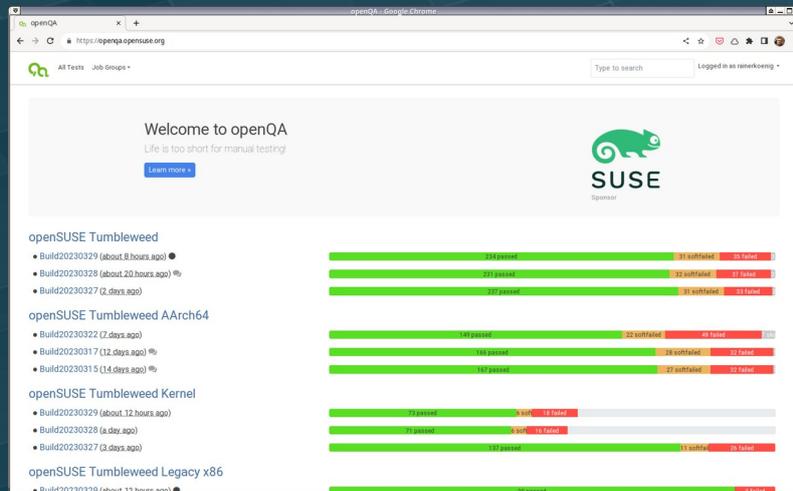


Schaubild wie ein Test in der Hierarchie aussieht.

Übergordnet sind Jobgroups, die Steuern welche Testssuites ausgeführt werden sollen.

Eine Testsuite kann entweder über einen YAML-Schedule gesteuert werden oder über Umgebungsvariablen. In diesem Fall übernimmt „main.pm“ die Rolle des Schedulers. Ziel ist aber langfristig alles auf YAML-Schedules umzustellen.

Webbasiertes System



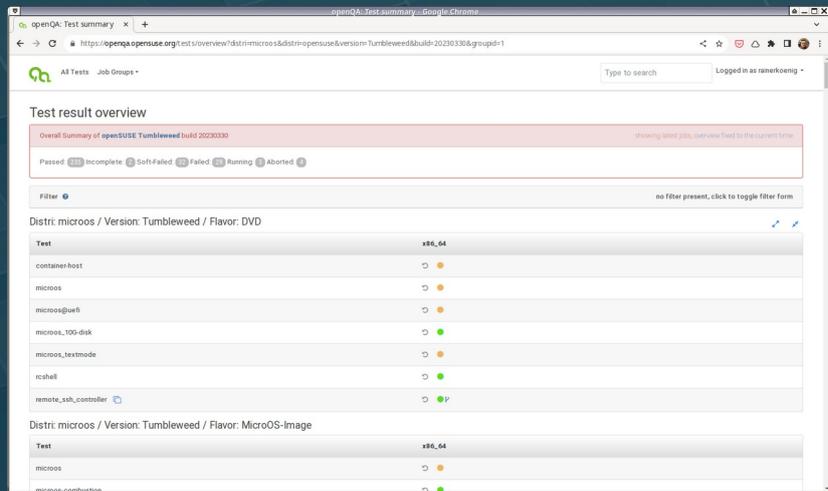
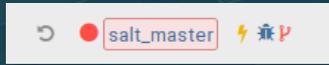
Übersicht, wie sich

<http://openqa.opensuse.org> beim Aufruf darstellt.

Jede Zeile enthält einen Build, hier sind die Oberbegriffe die Produkte wie Tumbleweed oder Leap.

Die Balken zeigen an wie viele Tests problemlos liefen, wie viel „softfailed“ oder „failed“ waren.

Build-Review Übersicht

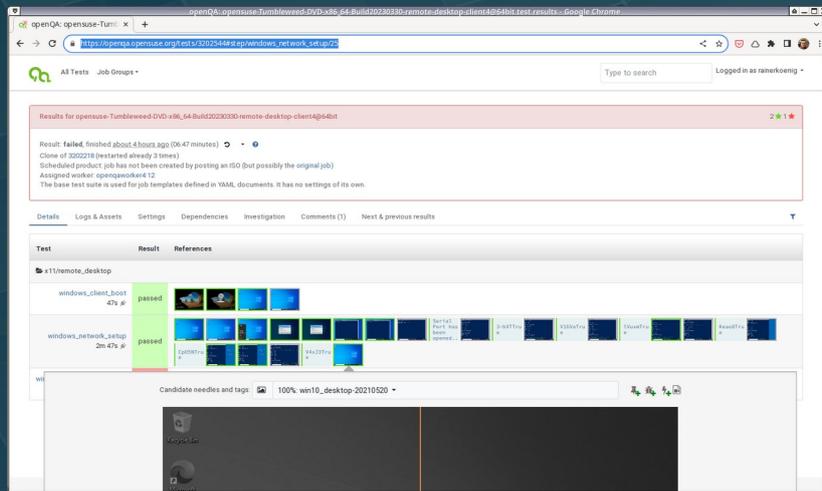


Klickt man auf einen Build, dann sieht man die Liste der Testsuites die hier getestet wurden.

Jede Zeile ist eine Testsuite, in der Spalte der Architekturt sieht man das Ergebnis. Das kann „grün“ (ok), „gelb“ (softfail), „rot“ (fail) oder auch blau („scheduled“ oder „running“) oder grau „incomplete“ sein.

Bei „roten“ Tests sieht man daneben eingeblendet, welches Testmodule den Fehler gefunden hat. Die Testsuite kann zudem mit Bugzilla oder Progress-Tickets „gelabelled“ sein, erkennbar an den Icons (klickbar, führen zu den Tickets). Bugs führen zu Bugzilla, der Blitz zu Progress.

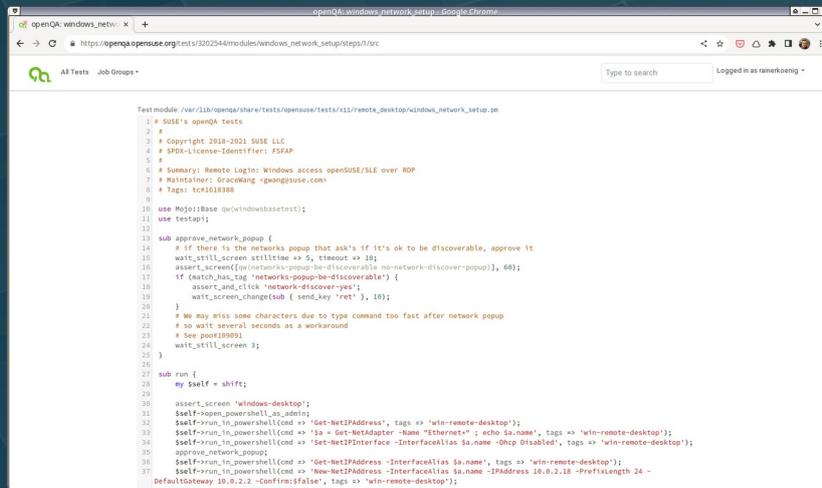
Eine Testsuite im Detail



Ein Klick auf die Testsuite zeigt dann das Ablaufprotokoll dieser Testsuite an.

Im Beispiel sehen wir, dass wir sogar Windows 10 testen können. Die Zeilen im Protokoll stehen für das jeweilige Testmodul, rechts davon sieht man das Ergebnis des Moduls und eine Sammlung von Screenshots die bei API-Aufrufen erstellt wurden.

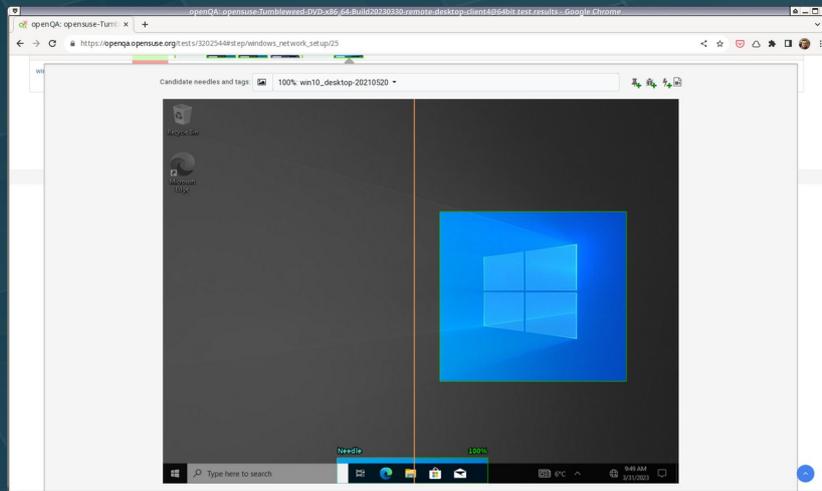
Testmodul Quelltext



```
Test module /var/lib/openqa/share/tests/openuse/test/13/remote_desktop/windows_network_setup.pm
1 # SUSE's openQA tests
2 #
3 # Copyright 2018-2021 SUSE LLC
4 # SPDX-License-Identifier: FSFAP
5 #
6 # Summary: Remote login: Windows Access openUSE/SLE over RDP
7 # Maintainers: @racoebng @openqaopenuse.com
8 # Tags: tc#10388
9
10 use Mojo::Base qw(WindowsBasetest);
11 use TestAPI;
12
13 sub approve_network_popup {
14     # If there is the network popup that ask's if it's ok to be discoverable, approve it
15     wait_still_screen stilltime => 5, timeout => 10;
16     assert_screen([@networks_popup_be_discoverable, @networks_discover_popup], 60);
17     if {match_has_tag 'networks-popup-be-discoverable'} {
18         assert_and_click 'network-discover-yes';
19         wait_screen_change(sub { send_key 'ret' }, 10);
20     }
21     # We may miss some characters due to type command too fast after network popup
22     # so wait several seconds as a workaround
23     # See pool199091
24     wait_still_screen 3;
25 }
26
27 sub run {
28     my $self = shift;
29
30     assert_screen 'windows-desktop';
31     $self->open_powershell_as_admin;
32     $self->run_in_powershell(cmd => 'Get-NetIPAddress', tags => 'win-remote-desktop');
33     $self->run_in_powershell(cmd => '$a = Get-NetAdapter -Name "Ethernet"; echo $a.name', tags => 'win-remote-desktop');
34     $self->run_in_powershell(cmd => 'Set-NetIPInterface -InterfaceAlias $a.name -Dhcp Disabled', tags => 'win-remote-desktop');
35     approve_network_popup;
36     $self->run_in_powershell(cmd => 'Get-NetIPAddress -InterfaceAlias $a.name', tags => 'win-remote-desktop');
37     $self->run_in_powershell(cmd => 'New-NetIPAddress -InterfaceAlias $a.name -IPAddress 10.0.2.10 -PrefixLength 24 -DefaultGateway 10.0.2.2 -Confirm:$false', tags => 'win-remote-desktop');
```

Ein Klick auf den Namen des Testmoduls im Testsuite-Ablaufprotokoll führt direkt zur Anzeige des Source-Codes dieses Moduls.

Eine Needle im Detail



Eine „Needle“ besteht aus zwei Dateien:

- Screenshot
- JSON-File

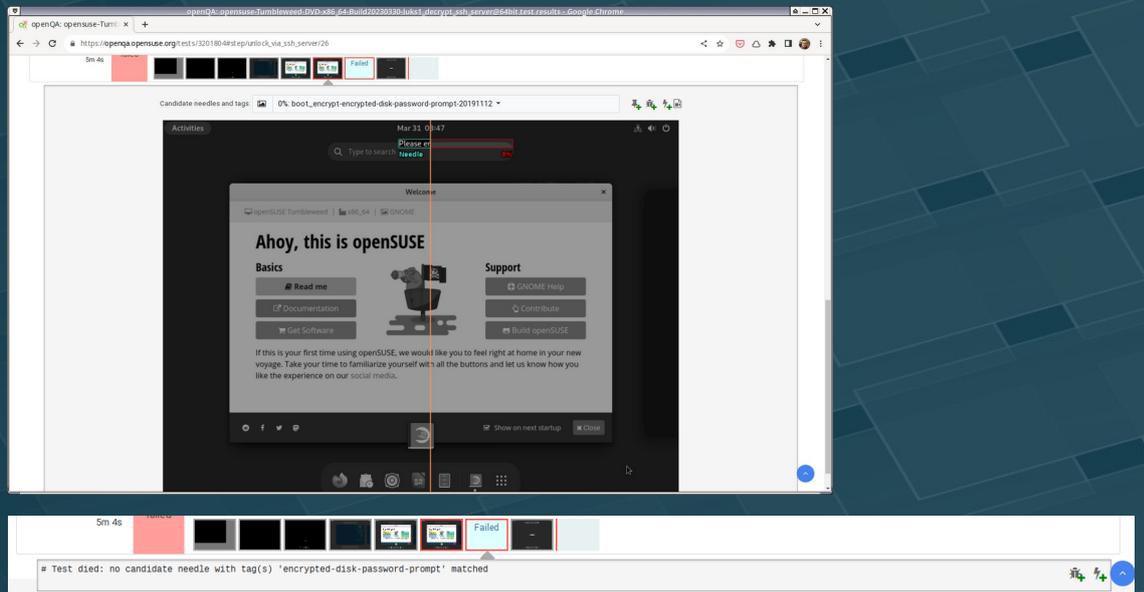
Der Screenshot zeigt an, was man erwartet. Im JSON sind dann „Needle-Tags“ definiert die vom Testmodul referenziert werden.

Außerdem kann ich „Matching Areas“ (grün) definieren, d.h. diese Bereiche müssen im aktuell angezeigten Bildschirm mit dem Screenshot übereinstimmen.

Man kann auch „Exclude-Areas“ festlegen, z.B. dort wo Uhrzeiten oder Fortschrittsbalken angezeigt werden.

Zudem gibt es „click.regions“ für `assert_and_click`. Der Senkrechts Strich kann bewegt werden und zeigt Needle und Screenshot im direkten Vergleich.

Anatomie eines Fehlers

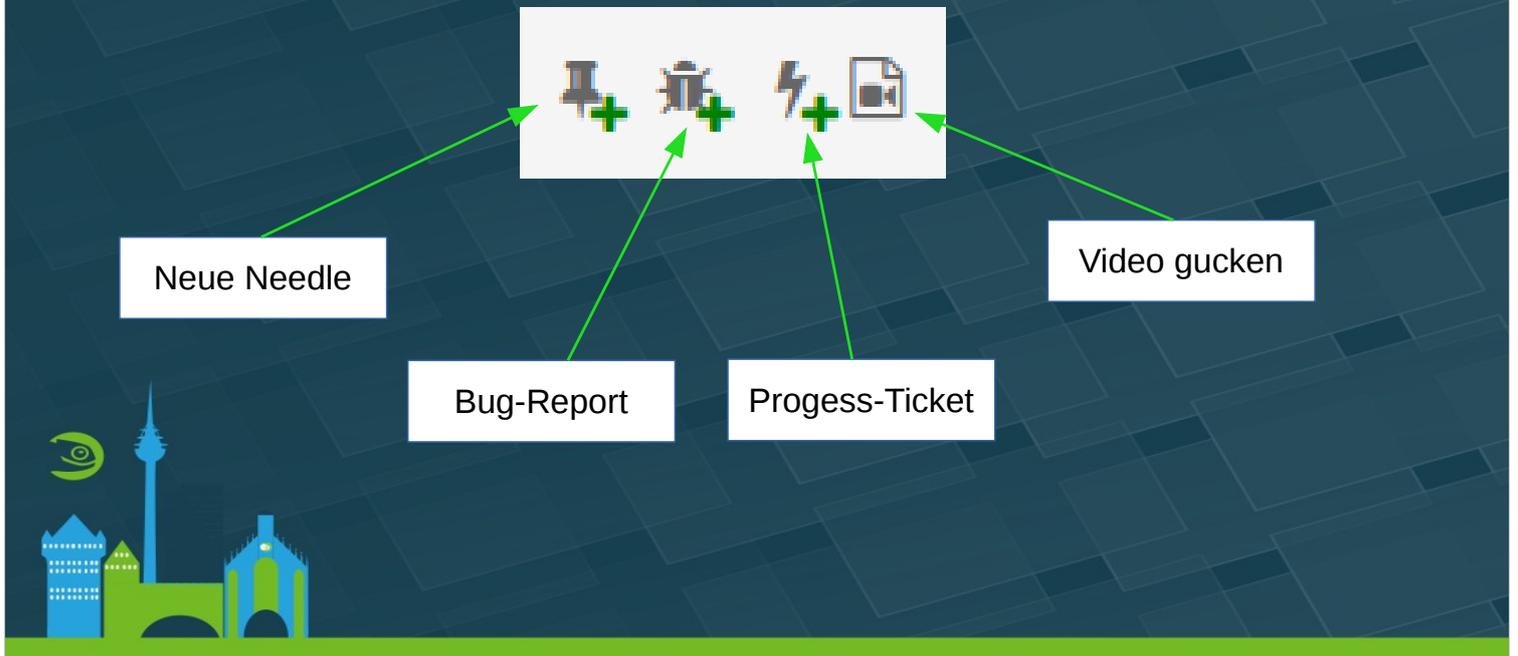


Hier haben wir einen Fehler.

Der aktuelle Screenshot vom Testlauf zeigt einen Willkommensbildschirm, die Needle würde aber einen Prompt zur Eingabe eines Festplatten-Encryption-Passwortes erwarten.

Unten sieht man dann die Meldung mit der openQA den Fehler meldet, eben, dass keine passende „Needle“ für den Tag xyz gefunden wurde.

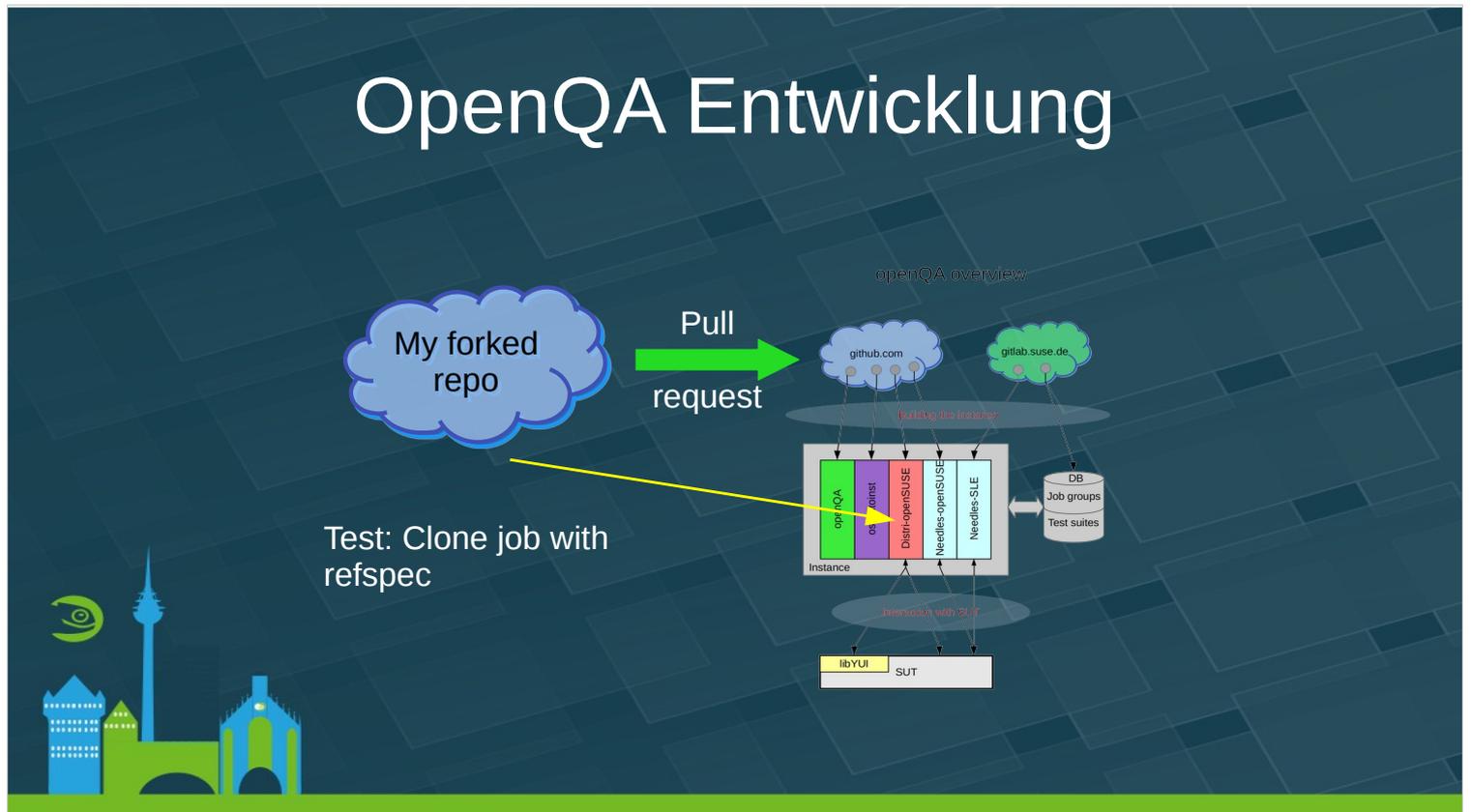
Oh, ein Fehler. Was nun?



Im Fehlerfall habe ich folgende Optionen:

- Neue Needle erstellen (Vorsicht, jede Needle mehr kostet Rechenzeit).
- Bug-Report wenn das Problem ein Problem des Produkts ist
- Progress-Ticket erstellen wenn die Testsuite hier verbessert werden muss.
- Ein Video des Testlaufes anschauen um mehr Informationen zu erhalten.

OpenQA Entwicklung



Die Entwicklung findet öffentlich auf GitHub statt. Man forkt das upstream Repository, ändert ein einm eigenen Branch und schickt einen Pull-Request.

Um die Wirkung von Änderungen zu testen kann man mit `openqa-clone-cusom-git-refspec` den Code für den Testlauf austauschen und so seinen eigenen Branch verwenden.

Danke für die Aufmerksamkeit.

Fragen?

