0. Agenda

FUJITSU

- What's Avocado?
- Composition and Architecture
- Key features of avocado
- Virtualization/Container Test
- To do in the future
- Hacking and Contributing

FUJITSU

- **Avocado** is a next generation testing framework, which is built on the experience accumulated with **Autotest**, while improving on its weaknesses and shortcomings.

AUTOTEST

AVOCADO

1.1 Achievement and Influence

FUJITSU

Received much attention and recognition :

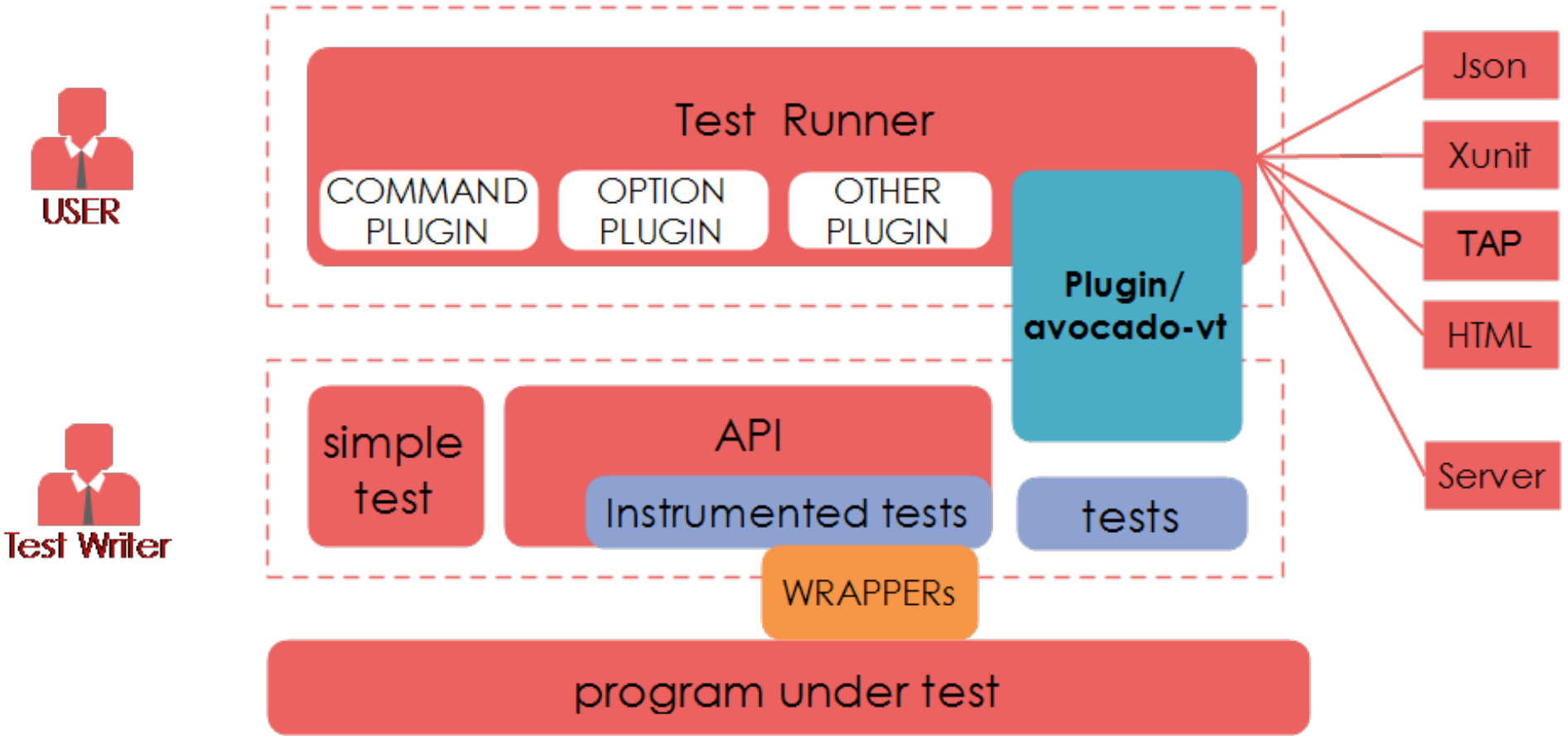
- "Avocado: Open Source Testing Made Easy" in LinuxCon North America, 2015 by **Lucas Meneghel Rodrigues,** [Doc]
- "Avocado: Next generation virt testing" in KVM Forum 2015 by **Cleber Rosa**, [video]

More and more companies/org/people have joined and contributed to avocado community :

2.0 Architecture

FUJITSU

Avocado includes three key components: Test runner、Libraries(APIs/utils) and plugins.



3.0 Features

FUJITSU

Avocado provides many practical features, only list part of them:

- External runner
- Plugin system
- Multiplex params system
- Wrapper
- Debugging with GDB
- Running tests remotely
- Others
  - Web interface/Dashboard
  - Logging system
  - Result format
  - Job ID/Job replay/Job diff
  - And so on

3.1 Feature : External runner

FUJITSU

Q: Sometimes, user want a very specific test runner that knows how to find and run their own tests, and do some custom built.

A: Avocado supports to run tests with an external runner.

- How this feature works?

Think of the "external runner" as some kind of interpreter , which recognizes and is able to execute the individual tests.

- Demo

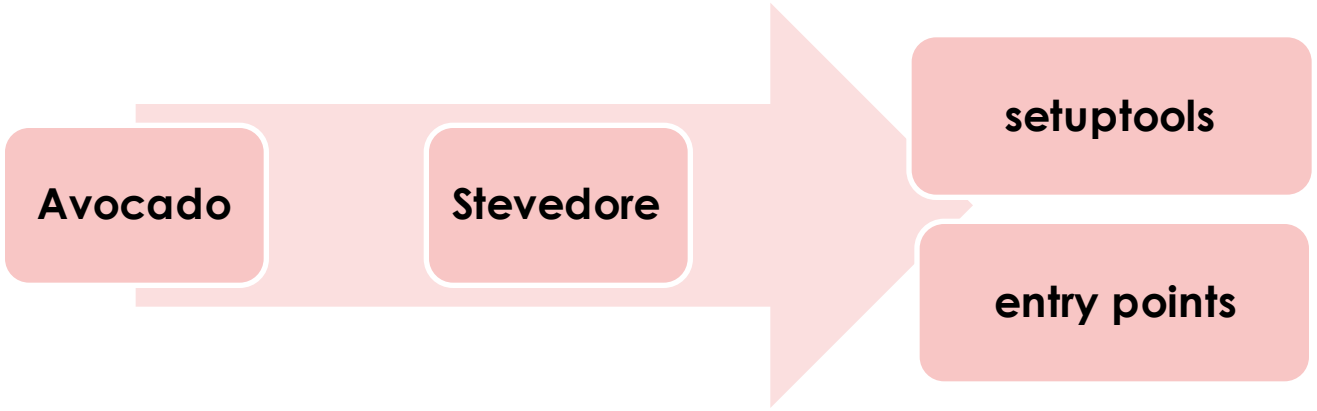
3.2 Feature：Plugin system

FUJITSU

Q: Is there any way to extend avocado or enable it to run third party test suites?

A: Avocado has a plugin system that can be used to extended it in a clean way.

- How this feature works?

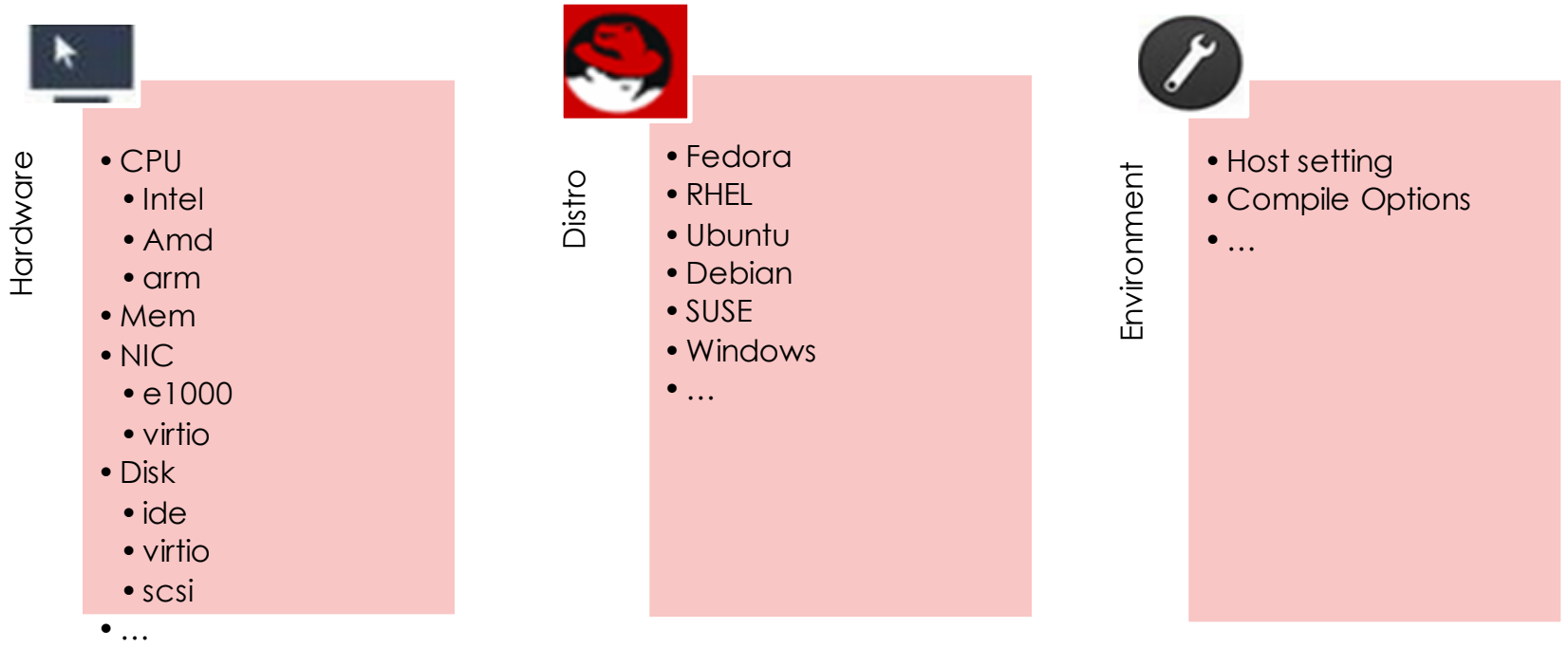


- Demo
  - Write, register and qualify a plugin

3.3 Feature：Multiplex params system

FUJITSU

Q: How to get a good coverage one always needs to execute the same test with different parameters or in various environments？ Take virtualization test as an example,



A: Avocado provides multiplexer to describe test matrix in a compact way, which use YAML files to define these variants and values, and allows the use of filters to reduce the scope of the matrix.

- Demo (mux-environment)

3.4 Feature：Wrapper

FUJITSU

Avocado allows the instrumentation of executables being run by a test in a transparent way. The user specifies a script ("the wrapper") to be used to run the actual program called by the test.

- Demo (rr - record and replay)

3.5 Feature：Debugging with GDB (external/internal)

FUJITSU

Avocado has two different types of GDB support that complement each other:

- The avocado.utils.process APIs that allows **the user** to interact with GDB by using a command line option.
- The avocado.utils.gdb APIs that allows **a test** to interact with GDB.

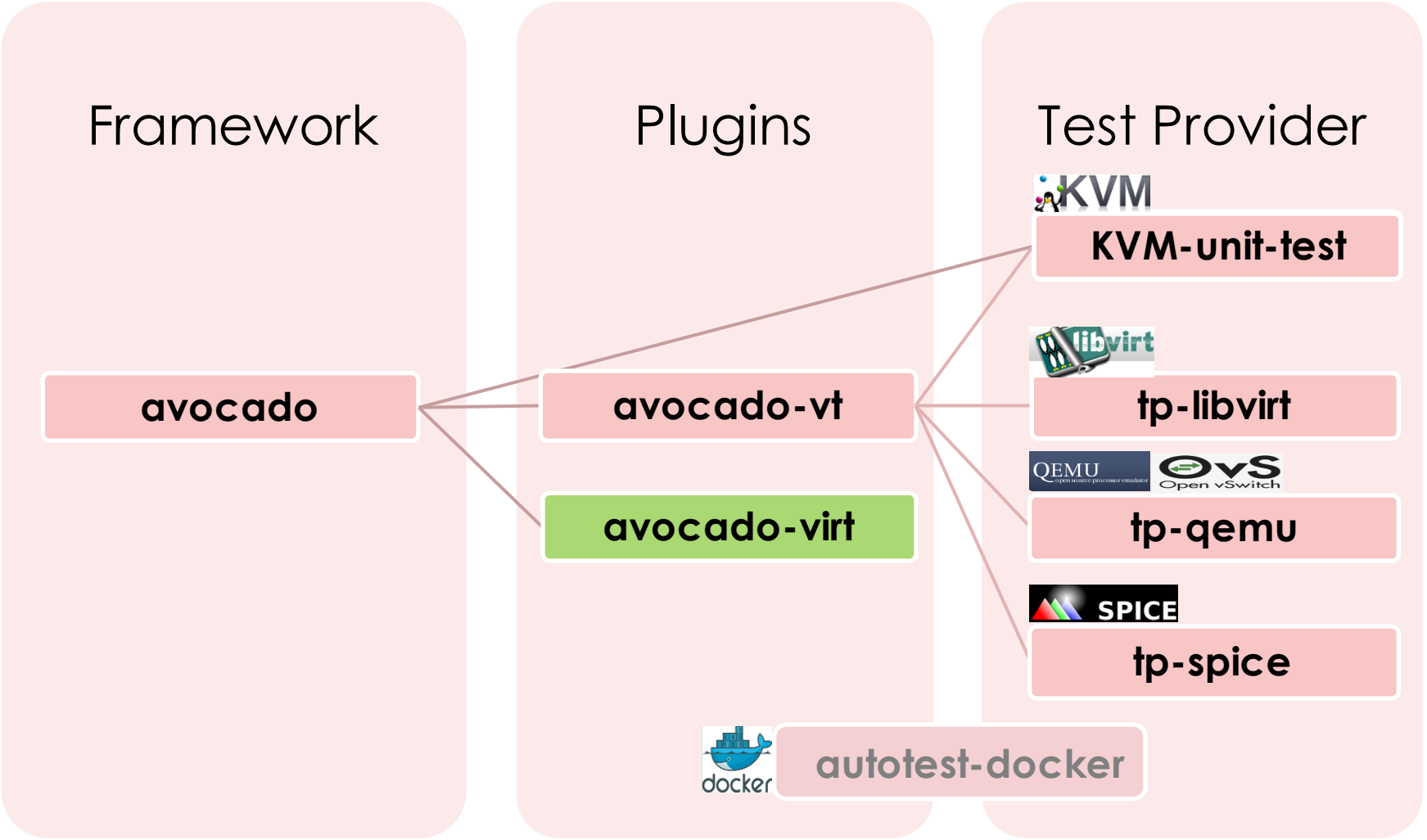
3.6 Feature：Running tests remotely

FUJITSU

- Running Tests on a Remote Host
- Running Tests on a Virtual Machine
- Running Tests on a Docker container

4.1 Virtualization test architecture

FUJITSU



4.2 Avocado-vt's capability

FUJITSU

- supported functions:
  - The CPU Arch:
    including i386, x86_64, x86_64, ppc64, ppc64le, arm64 (aarch64) , S390(new), …
  - Support hardware virtualization
  - Unattended install Guest OS

    Supported OS matrix:

Type	Distro
Linux	Fedora/RHEL/Centos/openSUSE/SLES/Debian/ubuntu/Jeos
windows	winxp/win-vista/win7/win8/win10/win2000/win2008/win2012

  - Guest Serial output for both linux and windows
  - Various installation methods (source tarball, git repo, rpm)
  - Migration testing
  - Performance testing (such as, iozone, fio, ffsb/aiostress/netperf/dbench/…)
  - Self-test(unitest)
  - …

4.3 Test providers (Concept)

FUJITSU

- The design goals behind test providers are:
  - Make it possible for other organizations to maintain test repositories
  - Stabilize API and enforce separation of core Avocado-VT functionality and tests

- The layout of test provider:

```
|-- backend  -> backend name
|  |-- cfg  -> test config directory. Holds base files for the test runner.
|  |-- deps  -> auxiliary files such as ELF files, Windows executables, images that tests
need.
|  |-- provider_lib -> shared libraries among tests.
   `-- tests -> python test files.
       `-- cfg -> config files for tests.
```

4.3 Test providers: tp-libvirt and tp-qemu

FUJITSU

- tp-libvirt has more than 8000 cases and supports:
  - Libvirt, The virtualization API
  - LVSB, libvirt sandbox container test
  - V2V
  - Libguestfs, the library and tools for acessing and modify disk images
  - Svirt, A technology that integrates Selinux and virtualization applies MAC
  - Others
- tp-qemu has more than 3000 cases and supports:
  - Qemu (focused on the low-level qemu stuff such as drivers, cpu types/features, hotplug/unplug,...)
  - Gerneric (such as install, kdump,...)
  - Openvswitch

4.4 How does avocado-vt know about these test providers

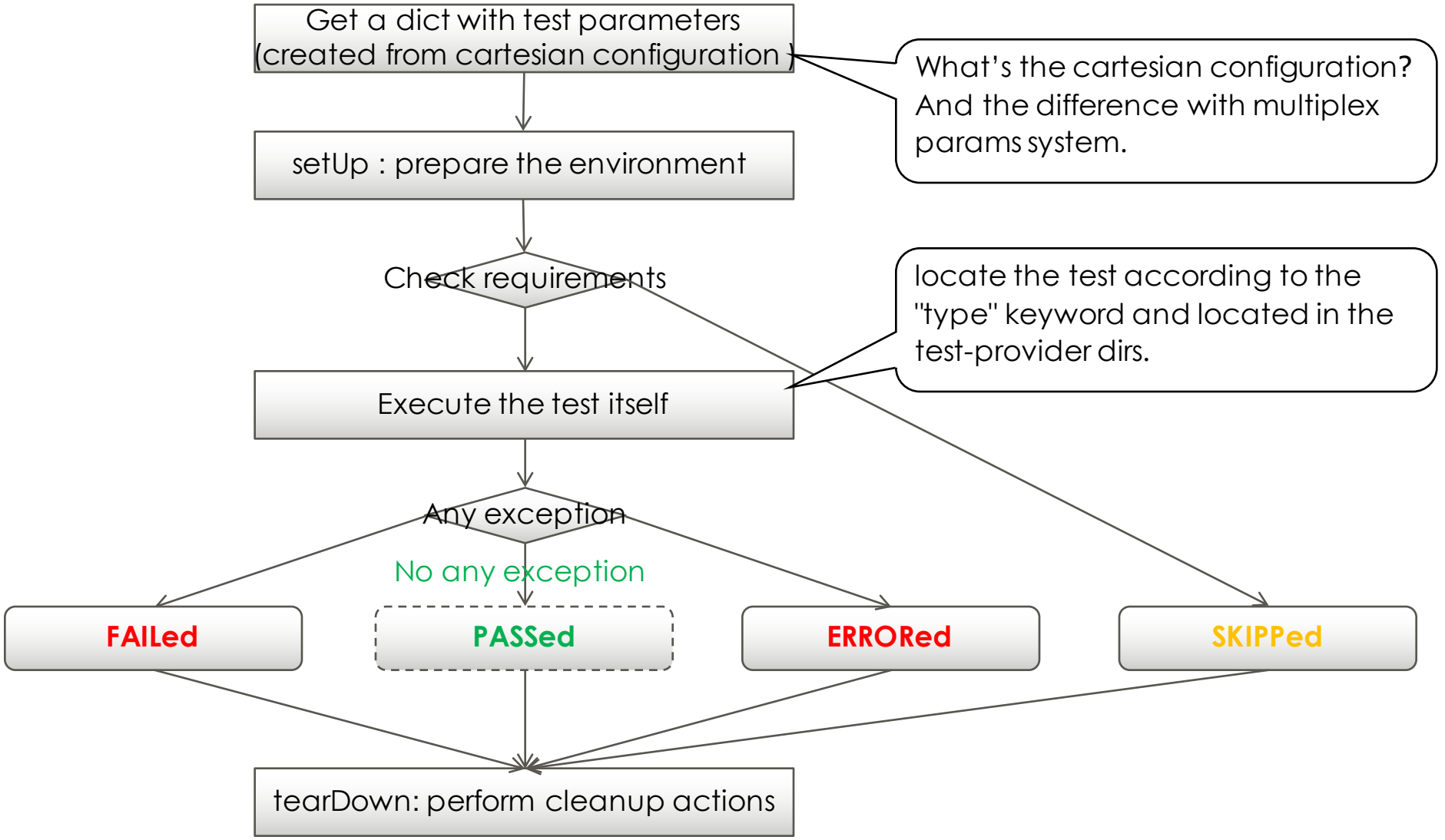
FUJITSU

- Avocado-vt finds and recognises these test providers by scanning definition files inside the 'test-providers.d' sub directory
- The definition/config files are **.ini** files that have the following structure:

```
[provider]
# Test provider URI (default is a git repository, fallback to standard dir)
uri: git://git-provider.com/repo.git
#uri: file:///path/to/tests/
#uri: /path-to-my-git-dir/repo.git
#uri: https://github.com/autotest/tp-qemu.git

# Virt backend
backend: qemu
```

4.5 How tests are run



4.6 Cartesian Configuration

FUJITSU

- It's a tool to create all possible variants of the specified categories with assigned params (key/value pairs).
- The basic factors in configuration file:
  - Keys and values
  - Variants/Named variants
  - Key sub-arrays
  - Dependencies
  - Filters
  - Default Configuration Files
  - Include statements
- The cartesian configuration is the precursor of the multiplex params system .

4.7 Practice of avocado-vt (confirm test object – the status of domains)

FUJITSU

- Simply select –-(in)active and –-state-xxx as variants, which are related to the status of domain

```
DESCRIPTION
  Returns list of domains.

OPTIONS
  --inactive          list inactive domains
  --all               list inactive & active domains
  --transient         list transient domains
  --persistent        list persistent domains
  --with-snapshot     list domains with existing snapshot
  --without-snapshot  list domains without a snapshot
  --state-running     list domains in running state
  --state-paused      list domains in paused state
  --state-shutoff     list domains in shutoff state
  --state-other       list domains in other states
  --autostart         list domains with autostart enabled
  --no-autostart      list domains with autostart disabled
  --with-managed-save  list domains with managed save state
  --without-managed-save  list domains without managed save
  --uuid              list uuid's only
  --name              list domain names only
  --table             list table (default)
  --managed-save      mark inactive domains with managed save state
  --title             show short domain description
```

4.8 Practice of avocado-vt (The combination of variants)

FUJITSU

virsh list	scope	state
1	active	running
2	inactive	paused
3	all	shutoff
4		other

No.	active	status	Type	Expectation
1	active	running	Normal	active>=running
2	active	paused	Normal	active>=paused
3	active	shutoff	Negative	-
4	active	other	Negative	-
5	inactive	running	Negative	-
6	inactive	paused	Negative	-
7	inactive	shutoff	Normal	inactive>=shutoff
8	inactive	other	Normal	all>=other
9	all	running	Normal	all>=running
10	all	paused	Normal	all>=running
11	all	shutoff	Normal	all>=running
12	all	other	Normal	all>=running

4.9 Practice of avocado-vt (Cartesian Configuration)

FUJITSU

```
- virsh.domain_list:
    type = domain_list
    # start main vm or not
    start_vm = "no"
    variants:
        - active:
            start_vm = "yes"
                    scope = ""
        - inactive:
                    scope = "--inactive"
        - all:
            scope = "--all"
    variants:
        - running:
            no inactive
            state = "--state-running"
        - paused:
            no inactive
            state = "--state-paused"
        - shutoff:
            no active
            state = "--state-shutoff"
        - other:
            no active,inactive
            state = "--state-other"
```



```
[root@localhost avocado]# scripts/avocado list --vt-type libvirt --vt-only-filter domain_list
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.running.active
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.running.all
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.paused.active
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.paused.all
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.shutoff.inactive
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.shutoff.all
VT type_specific.io-github-autotest-libvirt.virsh.domain_list.other.all
```

4.10 Practice of avocado-vt (Write a case and run)

FUJITSU

- Write the testcase

  Talk is cheap, let me show the code.

- Run and check the result.

```
[root@localhost avocado]# scripts/avocado run --vt-type libvirt type_specific.io-github-autotest-libvirt.virsh.domain_list.shut
off.inactive
JOB ID     : 1bf9c368cd7b9060be63720b5f74d588ac79bd0d
JOB LOG    : /root/avocado/job-results/job-2016-10-21T22.03-1bf9c36/job.log
TESTS      : 1
 (1/1) type_specific.io-github-autotest-libvirt.virsh.domain_list.shutoff.inactive: FAIL (0.69 s)
RESULTS    : PASS 0 | ERROR 0 | FAIL 1 | SKIP 0 | WARN 0 | INTERRUPT 0
TESTS TIME : 0.69 s
JOB HTML   : /root/avocado/job-results/job-2016-10-21T22.03-1bf9c36/html/results.html
```

```
2016-10-21 22:03:55,001 stacktrace       L0044 ERROR| Traceback (most recent call last):
2016-10-21 22:03:55,001 stacktrace       L0044 ERROR|   File "/mnt/extra_disk/osc/avocado-devel/avocado-vt/avocado_vt/test.
py", line 206, in runTest
2016-10-21 22:03:55,001 stacktrace       L0044 ERROR|     raise exceptions.TestFail(details)
2016-10-21 22:03:55,001 stacktrace       L0044 ERROR| TestFail: No shutoff vm with --state-shutoff!
2016-10-21 22:03:55,002 stacktrace       L0045 ERROR|
2016-10-21 22:03:55,002 test             L0589 ERROR| FAIL 1-type_specific.io-github-autotest-libvirt.virsh.domain_list.
shutoff.inactive -> TestFail: No shutoff vm with --state-shutoff!
```

4.11 Practice of avocado-vt (Fixed in libvirt 1.3.0: Dec 09 2015)

FUJITSU

```
commit 8dd47ead18ba64ee231dcef0a54e1b6ad797051e
Author: Wei Jiangang <weijg.fnst@cn.fujitsu.com>
Date:   Mon Nov 30 18:08:40 2015 +0800

    tools: fix output of list with state-shutoff

    Due to the default of flags is VIR_CONNECT_LIST_DOMAINS_ACTIVE,
    It doesn't show the domains that have been shutdown when we use
    'virsh list' with only --state-shutoff.

    Signed-off-by: Wei Jiangang <weijg.fnst@cn.fujitsu.com>

diff --git a/tools/virsh-domain-monitor.c b/tools/virsh-domain-monitor.c
index abc18e5..64ec03d 100644
--- a/tools/virsh-domain-monitor.c
+++ b/tools/virsh-domain-monitor.c
@@ -1873,7 +1873,8 @@ cmdList(vshControl *ctl, const vshCmd *cmd)
     unsigned int flags = VIR_CONNECT_LIST_DOMAINS_ACTIVE;

     /* construct filter flags */
-    if (vshCommandOptBool(cmd, "inactive"))
+    if (vshCommandOptBool(cmd, "inactive") ||
+        vshCommandOptBool(cmd, "state-shutoff"))
         flags = VIR_CONNECT_LIST_DOMAINS_INACTIVE;
```

5.0 To do in the future

FUJITSU

- Fix bugs
- Develop more new cases for virtualization products/technology
- Move test providers (tp-*) under the avocado-umbrella
- Remove the dependency on autotest.
- Turn to avocado-virt and discard avocado-vt ?
- Any new feature user needs
- …

6.0 Hacking and Contributing

FUJITSU

If you want to start hacking and contributing right away,

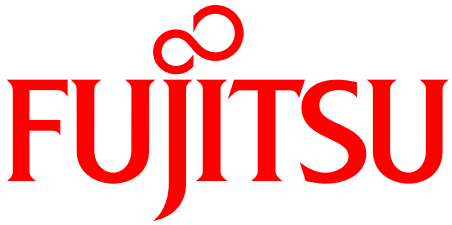
- Contribution and Community Guide
  - http://avocado-framework.readthedocs.org/en/latest/ContributionGuide.html
  - http://avocado-vt.readthedocs.io/en/latest/contributing/index.html
- Trello（Ideas & Schedules）
  - https://trello.com/b/WbqPNl2S/avocado
- Email list [Register]
  - avocado-devel@redhat.com
- Github Help
  - https://help.github.com/

Appendix: Avocado Resources

FUJITSU

- Main website
  - http://avocado-framework.github.io/
- Documents
  - http://avocado-framework.readthedocs.io/en/latest/
- Email archives
  - https://www.redhat.com/archives/avocado-devel/

- Other great learning materials
  - "Avocado - Next Generation Test Framework " by **Lucas Meneghel Rodrigues**, [video]
  - "Avocado and Jenkins: Test Automation and CI " by **Lukáš Doktor**, [video]
  - "Avocado Testing Framework - Advanced logging capabilities" by **Amador Pahim**, [video]
  - "The Planning and mind mapping of avocado" by **Lukáš Doktor**, [MisterMind]



shaping tomorrow with you