FUJITSU

shaping tomorrow with you

Avocado: The Next Generation Test Framework Used For Virt Test

October 22，2016

Fujitsu Nanda Software Technology, Co., Ltd

Wei Jiangang <weijg.fnst@cn.fujitsu.com>

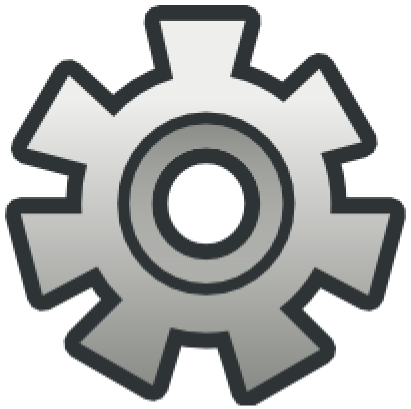
0. Agenda

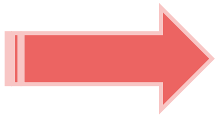
- What's Avocado?
- Composition and Architecture
- Key features of avocado
- Virtualization/Container Test
- To do in the future
- Hacking and Contributing

1.0 What's Avocado

- **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

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(people) have joined and contributed to avocado community:

?

1.2 Installation and Using

There're three ways to install the test framework avocado, choose one according to your requirements.

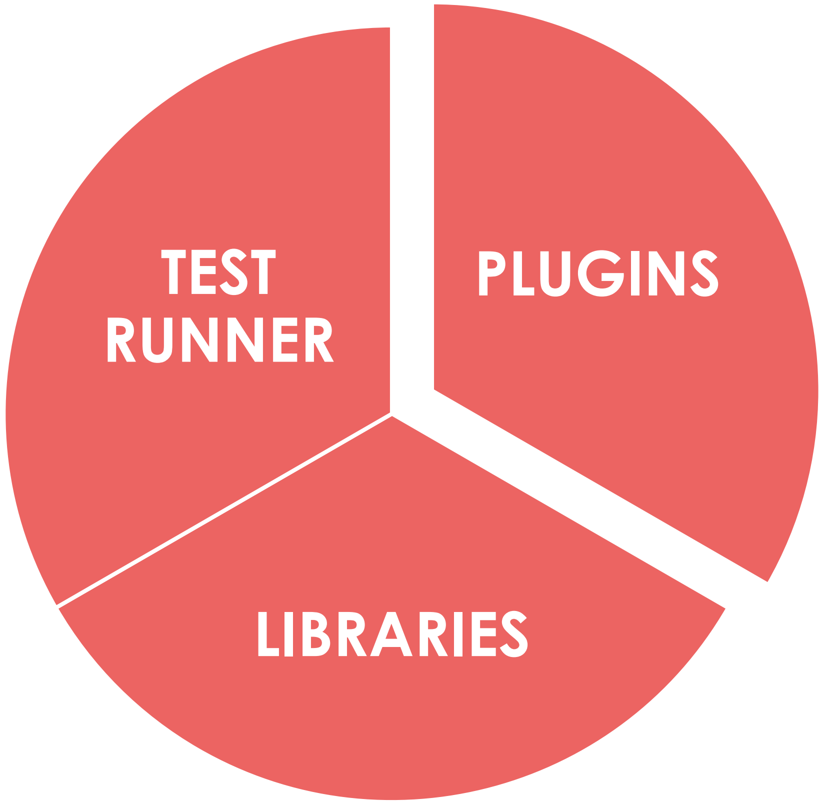
- Installing from Packages （RPM）
- Generic installation from a GIT repository
- Installing from standard python tools

Let's experience avocado by using the command line tool .

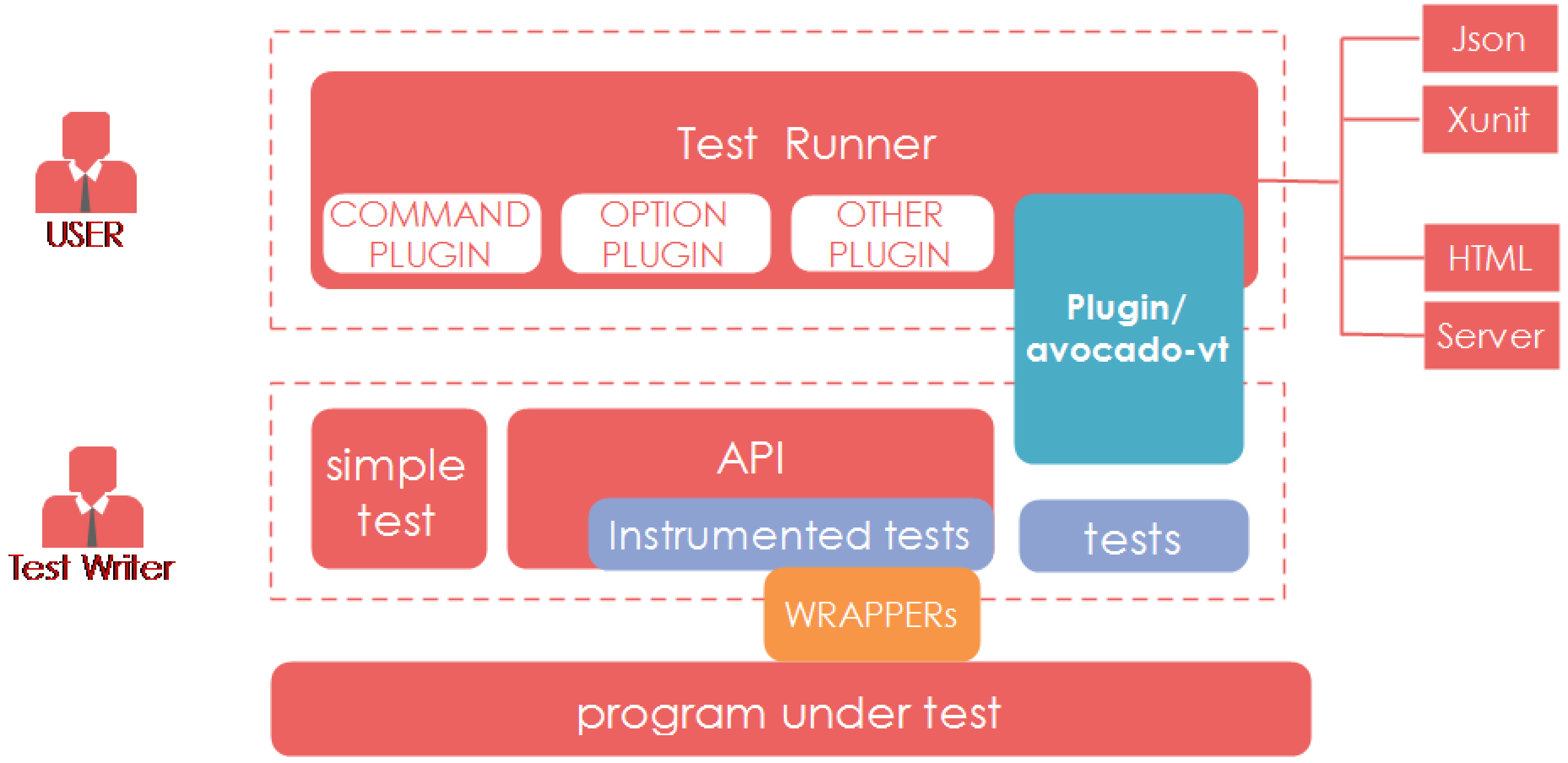
- Listing tests
- Running tests
- Debugging tests

2.0 Composition （General）

Avocado is a set of tools and libraries to help with automated testing, Avocado includes three key components: Test runner、Libraries(API) and plugins.



2.1 Architecture



3.0 Features

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

- External runner
- Plugin system
- Multiplex configuration
- Wrap executables run by tests
- Debugging with GDB
- Running tests remotely
- Others
  - Web interface/Dashboard
  - Job ID
  - Job replay
  - Job diff
  - Result formats
  - And so on

3.1 Feature：External runner

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 and the individual tests as anything that this interpreter recognizes and is able to execute.

- Demo

3.2 Feature：Plugin system

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?

  Avocado makes use of the Stevedore library to load and activate plugins.

  Stevedore itself uses setuptools and its entry points to register and find Python objects.

  Refer to http://docs.openstack.org/developer/stevedore/index.html

- Demo
  - Writing a plugin
  - Registering plugin
  - Fully qualified named for a plugin

3.3 Feature：Multiplex configuration

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，

Hardware

- CPU
- Mem
- NIC
    - e1000
    - virtio
- Disk
    - ide
    - virtio
    - scsi
- …

Distro

- Fedora
- RHEL
- Ubuntu
- Debian
- SUSE
- Windows
- …

Environment

- Host setting
- Compile Options
- …

3.3 Feature：Multiplex configuration

A: Avocado uses the term **Multiplexation** to generate multiple variants of the same test with different values.

- Mechanism

  The multiplexer is a mechanism of describing a 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

3.4 Feature： Wrapper

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

3.5 Feature：Debugging with GDB

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

Sometimes you don’t want to run a given test directly in your own machine.

Maybe the test is dangerous,

Maybe you need to run it in another Linux distribution,

so on and so forth…

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

4.0 Avocado Resources

- 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 **Anonymous**,[video]

5.0 Virtualization and Container

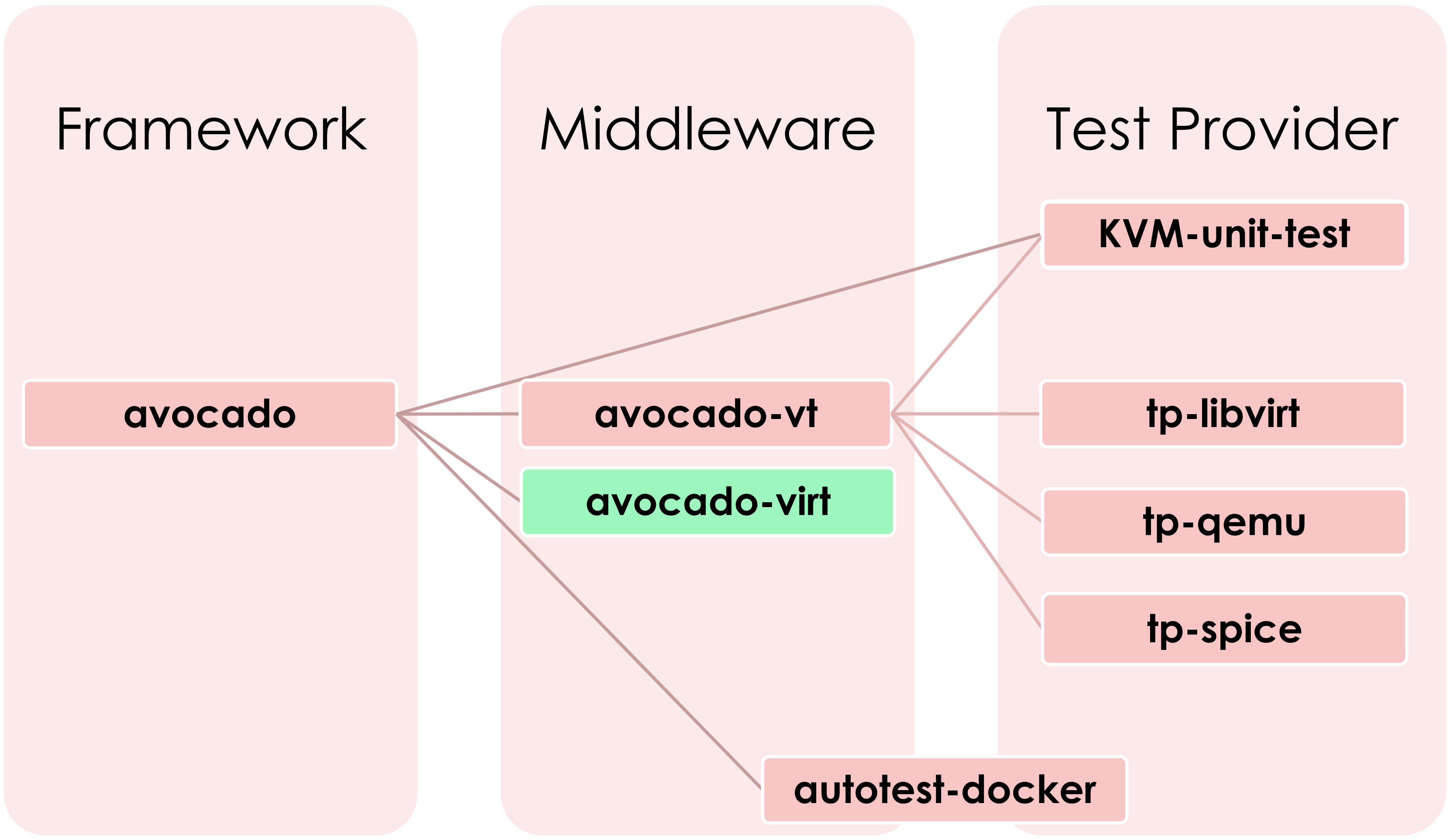
FUJITSU

- Avocado supports the following virtualization/container products

KVM	QEMU open source processor emulator	libvirt	OvS Open vSwitch	docker

- Qemu/Libvirt/Docker have their own independent test sets in avocado community , and support run the unit-test set of KVM by use of avocado and its libraries.

5.1 Virtualization test architecture



5.2 Middleware

- avocado-vt is the current generation virt testing plugin. It's an evolution of the virt-test project. It aims to be a centralizing project for most of the virt functional and performance testing needs.
- avocado-virt is the next generation virt testing plugin, and intends to be more flexible and clean than avocado-vt.

Note：The second is currently experimental and has a reduced feature set compared with avocado-vt, and is more suitable for virtualization developer.

5.3 avocado-vt's capability

FUJITSU

- supported functions:
  - The CPU Arch:
    including i386, x86_64, x86_64, ppc64, ppc64le, arm , S390, …
  - Hardware virtualization support (AMD and Intel)
  - Unattended install Guest OS

    Supportted 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 Linux guests
  - Various installation methods (source tarball, git repo, rpm)
  - Migration testing ()
  - Performance testing (such as, iozone, fio, ffsb/aiostress/netperf/dbench/…)
  - Self-test(unitest)
  - …

5.4 Test providers (Concept)

- Test providers are the conjunction of a loadable module mechanism that can pull a directory that will provide tests, config files and any dependencies, and those directories.
- 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. The actual name doesn't matter.
|  |-- 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.
```

5.4 Test providers : tp-libvirt and tp-qemu

- tp-libvirt has more than 8000 cases, 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 mainly aims at qemu and has more than 3000 cases, supports:
  - Gerneric (such as install, kdump,…)
  - Openvswitch

5.5 Cartesian Configuration

- It is a highly specialized way of providing lists of key/value pairs within combination's of various categories(setting variables). Each pairs pertaining to a single test
- The basic factors in configuration file:
  - Keys and values
  - Variants / Named variants
  - Key sub-arrays
  - Dependencies
  - Filters
  - Default Configuration Files
  - Include statements
- Demo

5.6 How does avocado-vt know about these test providers

- 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
```

5.7 How tests are run

Steps:

- Get a dict with test parameters(created from cartesian configuration)
- Based on these params, prepare the environment - create or destroy vm instances, create/check disk images, among others
- Execute the test itself:

  - If a test did not raise an exception, it PASSed

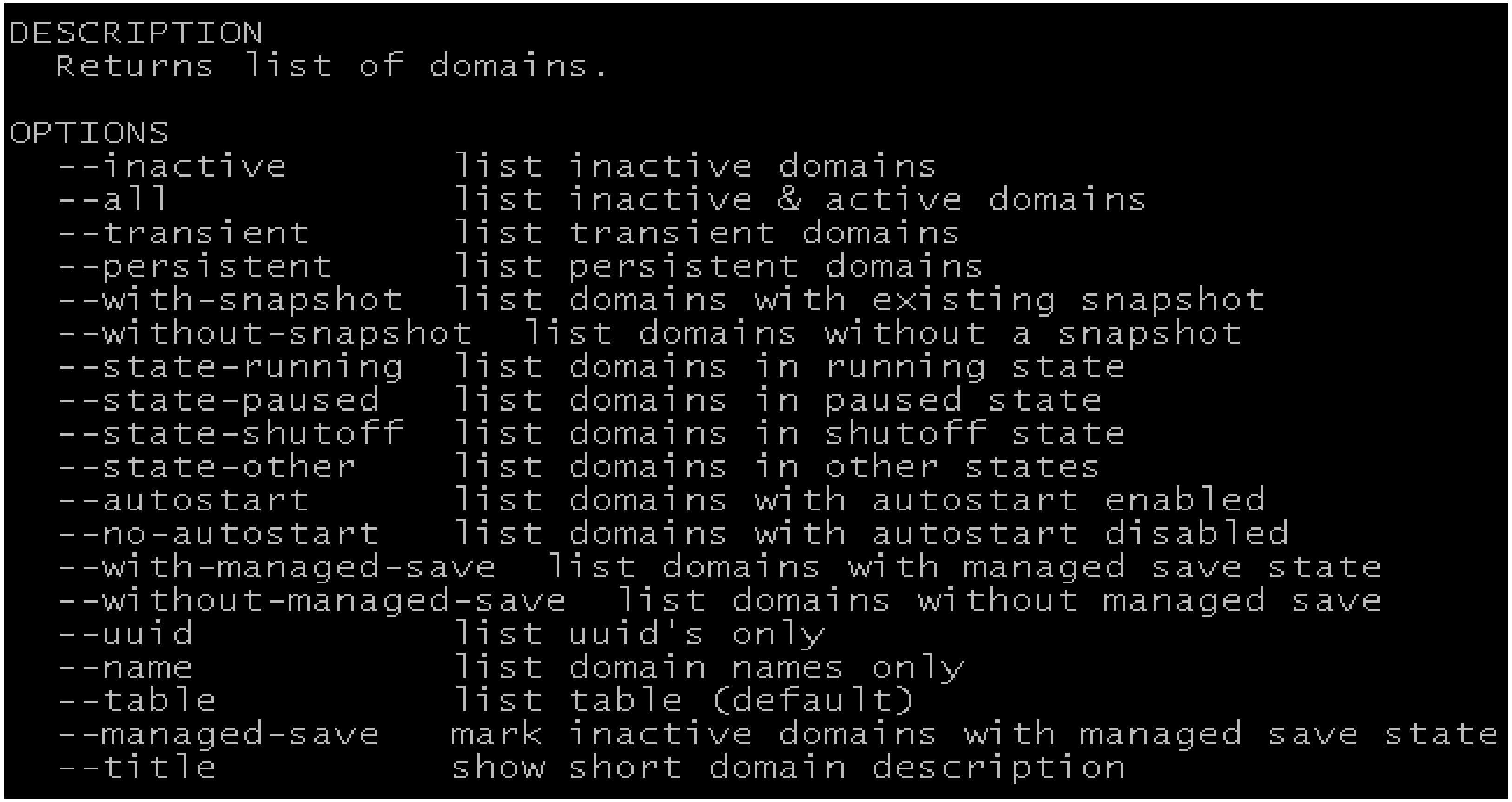
  - If a test raised a TestFail exception, it FAILed.

  - If a test raised a TestNAError, it SKIPPed.

  - Otherwise, it ERRORed
- Based on what happened during the test, perform cleanup actions, such as killing vms, and remove unused disk images.

5.8 Practice of avocado-vt (confirm test object – domain's status)

- Simply select --(in)active and --state-xxx as variants, which are related to domain's status



```
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
```

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

virsh list	scope	status
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	inactive>=shutoff
9	all	running	Normal	all>=running
10	all	paused	Normal	all>=running
11	all	shutoff	Normal	all>=running
12	all	other	Normal	all>=running

5.8 Practice of avocado-vt (Cartesian Configuration)

```
variants:
   - active:
       scope = None
   - inactive:
       scope = --inactive
   - all:
       scope = --all
variants:
   - running:
       state = --state-running
   - paused:
       state = --state-paused
   - shutoff:
       state = --state-shutoff
   - other:
       state = --other
```

```
dict    1:  running.active
dict    2:  running.inactive
dict    3:  running.all
dict    4:  paused.active
dict    5:  paused.inactive
dict    6:  paused.all
dict    7:  shutoff.active
dict    8:  shutoff.inactive
dict    9:  shutoff.all
dict   10:  other.active
dict   11:  other.inactive
dict   12:  other.all
```

5.8 Practice of avocado-vt (Write case and run)

- Demo

  Talk is cheap, let me show the code.

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

```
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;
```

6.0 To do in the future

- Fix bugs

  This is a long-term task for any open source projects
- 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
- …

7.0 Hacking and Contributing

FUJITSU

If you want to start hacking and contributing right away,

- Contribution and Community Guide
  - avocado
  - avocado-vt

- Trello（Ideas & Schedules）
  - https://trello.com/b/WbqPNl2S/avocado
- Email list [Register]
  - avocado-devel@redhat.com
- Github Help
  - https://help.github.com/

FUJITSU

shaping tomorrow with you