# UTAH Tutorial Documentation

***Release 0.1***

**UTAH development team**

July 18, 2016

# Contents

Contents:

CHAPTER 1

# PART I: Basic usage

## 1.1 Installation

To install the latest stable version of the utah client, let's add the UTAH stable PPA to our sources, and install the `utah-client` package:

```
$ sudo add-apt-repository -y ppa:utah/stable
$ sudo apt-get update
$ sudo atp-get install utah-client
```

The binary used to run the test cases is `utah`. We can take a look at all the available arguments using the `-h`/`--help` option:

```
$ utah -h
```

**Note:** `utah` is installed as part of the `utah-client` package.

In this example, we're interested just in the `-r`/`--runlist` argument which is used to tell the client which test suites should executed in a single run.

## 1.2 Writing tests

### 1.2.1 Test suite

To create a test suite and a test case from scratch, we'll use the `phoenix` command installed as part of the `utah-client` package:

```
$ cd /tmp
$ phoenix utah_howto test_one
```

This will create a new test suite under a directory called `utah_howto` with some files in it:

- `master.run`: main run list expected to be passed to `utah` in the `-r`/`--runlist` argument. As explained above, it contains a list of all the test suites to be executed in a single run.

  **Note:** In the general case, the run list will be in a different location, not in the same directory as the test suite.

- `tslist.run`: test suite list with a description of the test cases to be executed.

  **Note:** Test cases created by `phoenix` will be automatically added to the test suite list. In particular, note that `test_one` is already in the file.

- `ts_control`: test suite metadata file with additional information needed to set the environment to execute the test suite properly.
- `test_one/tc_control`: test case metdata file with specific information needed to run a particular test case.

**Note:** All the files above use yaml syntax, take advantage of the syntax highlighting feature of your preferred editor.

### 1.2.2 Test case

Let's edit `test_one/tc_control` to write a simple test case that verifies that `/bin/true` works as expected. The final result should be as follows:

```
description: System sanity check
dependencies: coreutils
action: |
 1. Run /bin/true
expected_results: |
 1. /bin/true exits with status 0
type: userland
timeout: 60
command: /bin/true
run_as: utah
```

where:

- `command`: is what will be executed to run the test case

  **Note:** the return code from the command is used by utah to determine whether the test case passed or not using the unix convention.

- `run_as`: is the user that will executed the command

**Note:** `dependencies`, `action` and `expected_results` are there for description purposes only. The utah client doesn't parse/use them for now, but that might change in the future.

### 1.2.3 Run list

Once we have a test suite and a test case, we need to edit the run list to be able to execute them:

```
---
testsuites:
  - name: utah_howto
    fetch_method: dev
    fetch_location: /tmp/utah_howto
```

where:

- `fetch_method`: tells the utah client how to get the test suite
- `fetch_location`: tells the utah client where to get the test suite from

**Note:** By default all test cases in the test suite are executed

## 1.3 Executing tests

Once the test suite and cases have been writen and the run list is ready, the utah client can be used to run the test cases as follows:

```
$ sudo utah -r master.run > report.yaml
$ vim report.yaml
```

**Note:** `utah` must be executed as `root` for now to make it possible to execute commands as a different user easily. In the future this might be improved to avoid the this.

The contents of the test execution report should be similar to the one below:

```
---
arch: amd64
build_number: '20121017.5'
commands:
- cmd_type: testsuite_fetch
  command: cp -r /tmp/utah_howto utah_howto
  returncode: 0
  start_time: '2012-11-08 14:08:21.972824'
  stderr: ''
  stdout: ''
  time_delta: '0:00:00.003381'
  user: root
- cmd_type: testsuite_fetch
  command: echo 'DEVELOPMENT'
  returncode: 0
  start_time: '2012-11-08 14:08:21.976431'
  stderr: ''
  stdout: |-
    DEVELOPMENT
  time_delta: '0:00:00.001907'
  user: root
- cmd_type: testcase_test
  command: /bin/true
  extra_info:
    action: |-
      1. Run /bin/true
    dependencies: coreutils
    description: System sanity check
    expected_results: |-
      1. /bin/true exits with status 0
  returncode: 0
  start_time: '2012-11-08 14:08:22.004614'
  stderr: ''
  stdout: ''
  testcase: test_one
```

```
36    testsuite: /var/lib/utah/testsuites/utah_howto
37    time_delta: '0:00:00.029548'
38    user: utah
39  errors: 0
40  failures: 0
41  fetch_errors: 0
42  install_type: desktop
43  media-info: Ubuntu 12.10 "Quantal Quetzal" - Release amd64 (20121017.5)
44  name: unnamed
45  passes: 1
46  ran_at: '2012-11-08 14:08:21.972824'
47  release: quantal
48  runlist: /tmp/utah_howto/master.run
49  uname:
50  - Linux
51  - xps8300
52  - 3.5.0-18-generic
53  - '#29-Ubuntu SMP Fri Oct 19 10:26:51 UTC 2012'
54  - x86_64
55  - x86_64
```

The more important things to note for now are:

- lines 5-6: the test suite is fetched from its location.
- lines 22-23: the test case is executed
- line 45: the test case passed successfully

## 1.4 Including/excluding test cases

Let's continue the example by adding a new test case to the test suite we've already created:

```
$ phoenix . test_two
```

**Note:** `phoenix` will add `test_two` to `tslist.run` automatically

After that, let's edit `test_two/tc_control` and set the following contents:

```
description: Test FAIL protocol
dependencies: wget
action: |
  1. Use fail protocol to retrieve example.com
expected_results: |
  1. example.com retrieved
type: userland
timeout: 60
command: wget fail://example.com
run_as: utah
```

As it can be seen, the call to `wget` will fail because the protocol in the URL is invalid.

**Warning:** there's a bug and utah that will cause problems when trying this example depending on the locale configuration.

When we're done editing the test case metadata, the utah client can be executed again:

```
$ sudo utah -r master.run > report.yaml
$ vim report.yaml
```

Looking at the test execution report, the part about the new test case command is as follows:

```
---
arch: amd64
build_number: '20121017.5'
commands:
- cmd_type: testsuite_fetch
  command: cp -r /tmp/utah_howto utah_howto
  returncode: 0
  start_time: '2012-11-08 15:02:46.993684'
  stderr: ''
  stdout: ''
  time_delta: '0:00:00.003440'
  user: root
- cmd_type: testsuite_fetch
  command: echo 'DEVELOPMENT'
  returncode: 0
  start_time: '2012-11-08 15:02:46.997347'
  stderr: ''
  stdout: |-
    DEVELOPMENT
  time_delta: '0:00:00.001918'
  user: root
- cmd_type: testcase_test
  command: /bin/true
  extra_info:
    action: |-
      1. Run /bin/true
    dependencies: coreutils
    description: System sanity check
    expected_results: |-
      1. /bin/true exits with status 0
  returncode: 0
  start_time: '2012-11-08 15:02:47.024652'
  stderr: ''
  stdout: ''
  testcase: test_one
  testsuite: /var/lib/utah/testsuites/utah_howto
  time_delta: '0:00:00.010179'
  user: utah
- cmd_type: testcase_test
  command: wget fail://example.com
  extra_info:
    action: |-
      1. Use fail protocol to retrieve example.com
    dependencies: wget
    description: Test FAIL protocol
    expected_results: |-
      1. example.com retrieved
  returncode: 1
  start_time: '2012-11-08 15:02:47.064155'
  stderr: |-
    fail://example.com: Unsupported scheme `fail'.
  stdout: ''
```

```
53     testcase: test_two
54     testsuite: /var/lib/utah/testsuites/utah_howto
55     time_delta: '0:00:00.049322'
56     user: utah
57   errors: 0
58   failures: 1
59   fetch_errors: 0
60   install_type: desktop
61   media-info: Ubuntu 12.10 "Quantal Quetzal" - Release amd64 (20121017.5)
62   name: unnamed
63   passes: 1
64   ran_at: '2012-11-08 15:02:46.993684'
65   release: quantal
66   runlist: /tmp/utah_howto/master.run
67   uname:
68   - Linux
69   - xps8300
70   - 3.5.0-18-generic
71   - '#29-Ubuntu SMP Fri Oct 19 10:26:51 UTC 2012'
72   - x86_64
73   - x86_64
```

where it can be seen that:

- line 48: the test case command failed
- lines 50-51: the problem was indeed using an invalid protocol in the url
- line 58: the command failure was considered a test case failure

Let's say that we know the test case has a problem, but we don't have time to fix it now. Instead, what we want to do is skip it until it's fixed in the future.

To do that, edit `master.run` and specify that `test_two` must be excluded:

```
---
testsuites:
  - name: utah_howto
    fetch_method: dev
    fetch_location: /tmp/utah_howto
    exclude_tests:
     - test_two
```

After this change, if the utah client is executed again:

```
$ sudo utah -r master.run > report.yaml
$ vim report.yaml
```

The report only shows a test case executed and no errors:

```
---
arch: amd64
build_number: '20121017.5'
commands:
- cmd_type: testsuite_fetch
  command: cp -r /tmp/utah_howto utah_howto
  returncode: 0
  start_time: '2012-11-08 15:34:58.501273'
  stderr: ''
  stdout: ''
  time_delta: '0:00:00.003482'
  user: root
- cmd_type: testsuite_fetch
  command: echo 'DEVELOPMENT'
  returncode: 0
  start_time: '2012-11-08 15:34:58.504980'
  stderr: ''
  stdout: |-
    DEVELOPMENT
  time_delta: '0:00:00.001902'
  user: root
- cmd_type: testcase_test
  command: /bin/true
  extra_info:
    action: |-
      1. Run /bin/true
    dependencies: coreutils
    description: System sanity check
    expected_results: |-
      1. /bin/true exits with status 0
  returncode: 0
  start_time: '2012-11-08 15:34:58.526534'
  stderr: ''
  stdout: ''
  testcase: test_one
  testsuite: /var/lib/utah/testsuites/utah_howto
  time_delta: '0:00:00.010364'
  user: utah
errors: 0
failures: 0
fetch_errors: 0
install_type: desktop
media-info: Ubuntu 12.10 "Quantal Quetzal" - Release amd64 (20121017.5)
name: unnamed
passes: 1
ran_at: '2012-11-08 15:34:58.501273'
release: quantal
runlist: /tmp/utah_howto/master.run
uname:
- Linux
- xps8300
- 3.5.0-18-generic
- '#29-Ubuntu SMP Fri Oct 19 10:26:51 UTC 2012'
- x86_64
- x86_64
```

## 1.5 Build/setup/cleanup

Sometimes, it might happen that a test case is written in a compiled language or that it requires a special configuration to be in place before it's executed. To handle those test cases, there's a special metadata that can be added to the test case.

Let's create another test case in our test suite:

```
$ phoenix . test_three
```

To simulate a test case that requires a build step, let's write a `Makefile` under the `test_three` directory that generates the a script we want to execute later in the test case:

```
test_three.sh:
    echo 'test -f /tmp/foo' > test_three.sh
    chmod +x test_three.sh
```

After that, let's edit `test_three/tc_control` to make sure that the `make` command is used in a build step before running the test case:

```
description: Test that /tmp/foo exists
dependencies: make
action: |
  1. Test that /tmp/foo exists
expected_results: |
  1. /tmp/foo indeed exists
type: userland
timeout: 60
command: ./test_three.sh
run_as: utah
build_cmd: make
```

At this point, we can run the utah client:

```
$ sudo utah -r master.run > report.yaml
$ vim report.yaml
```

and check the test execution report:

```
---
arch: amd64
build_number: '20121017.5'
commands:
- cmd_type: testsuite_fetch
  command: cp -r /tmp/utah_howto utah_howto
  returncode: 0
  start_time: '2012-11-08 16:23:18.733182'
  stderr: ''
  stdout: ''
  time_delta: '0:00:00.003528'
  user: root
- cmd_type: testsuite_fetch
  command: echo 'DEVELOPMENT'
  returncode: 0
  start_time: '2012-11-08 16:23:18.736933'
  stderr: ''
  stdout: |-
    DEVELOPMENT
  time_delta: '0:00:00.001879'
  user: root
- cmd_type: testcase_test
  command: /bin/true
  extra_info:
    action: |-
      1. Run /bin/true
    dependencies: coreutils
    description: System sanity check
    expected_results: |-
      1. /bin/true exits with status 0
  returncode: 0
  start_time: '2012-11-08 16:23:18.765374'
```

```
33    stderr: ''
34    stdout: ''
35    testcase: test_one
36    testsuite: /var/lib/utah/testsuites/utah_howto
37    time_delta: '0:00:00.010362'
38    user: utah
39  - cmd_type: testcase_build
40    command: make
41    returncode: 0
42    start_time: '2012-11-08 16:23:18.796690'
43    stderr: ''
44    stdout: |-
45      echo 'test -f /tmp/foo' > test_three.sh
46      chmod +x test_three.sh
47    testcase: test_three
48    testsuite: /var/lib/utah/testsuites/utah_howto
49    time_delta: '0:00:00.005390'
50    user: root
51  - cmd_type: testcase_test
52    command: ./test_three.sh
53    extra_info:
54      action: |-
55        1. Test that /tmp/foo exists
56      dependencies: make
57      description: Test that /tmp/foo exists
58      expected_results: |-
59        1. /tmp/foo indeed exists
60    returncode: 1
61    start_time: '2012-11-08 16:23:18.817905'
62    stderr: ''
63    stdout: ''
64    testcase: test_three
65    testsuite: /var/lib/utah/testsuites/utah_howto
66    time_delta: '0:00:00.010506'
67    user: utah
68  errors: 0
69  failures: 1
70  fetch_errors: 0
71  install_type: desktop
72  media-info: Ubuntu 12.10 "Quantal Quetzal" - Release amd64 (20121017.5)
73  name: unnamed
74  passes: 1
75  ran_at: '2012-11-08 16:23:18.733182'
76  release: quantal
77  runlist: /tmp/utah_howto/master.run
78  uname:
79  - Linux
80  - xps8300
81  - 3.5.0-18-generic
82  - '#29-Ubuntu SMP Fri Oct 19 10:26:51 UTC 2012'
83  - x86_64
84  - x86_64
```

What we see here is that:

- lines 39-40: there's a new build command that generates the files needed to run the test case.
- line 60: the test case failed because the `/tmp/foo` doesn't exist.

Hence, we managed to generate the file needed to run the test case, but failed to configure the environment properly, that is, have the `/tmp/foo` file in place.

To address that issue, let's edit again `test_three/tc_control` as follows:

```
description: Test that /tmp/foo exists
dependencies: make
action: |
  1. Test that /tmp/foo exists
expected_results: |
  1. /tmp/foo indeed exists
type: userland
timeout: 60
command: ./test_three.sh
run_as: utah
build_cmd: make
tc_setup: touch /tmp/foo
tc_cleanup: rm /tmp/foo
```

where:

- `tc_setup` is a command that is executed to take care of all the configuration needed for the test case to work correctly.
- `tc_cleanup` is a command that is executed to undo whatever the setup command did and set the environment as it was before executing the test case.

Now if we run agan the utah client,

```
$ sudo utah -r master.run > report.yaml
$ vim report.yaml
```

we see the following test execution report:

```
---
arch: amd64
build_number: '20121017.5'
commands:
- cmd_type: testsuite_fetch
  command: cp -r /tmp/utah_howto utah_howto
  returncode: 0
  start_time: '2012-11-08 16:37:12.817425'
  stderr: ''
  stdout: ''
  time_delta: '0:00:00.003487'
  user: root
- cmd_type: testsuite_fetch
  command: echo 'DEVELOPMENT'
  returncode: 0
  start_time: '2012-11-08 16:37:12.821134'
  stderr: ''
  stdout: |-
    DEVELOPMENT
  time_delta: '0:00:00.001893'
  user: root
- cmd_type: testcase_test
  command: /bin/true
  extra_info:
    action: |-
      1. Run /bin/true
    dependencies: coreutils
    description: System sanity check
    expected_results: |-
      1. /bin/true exits with status 0
  returncode: 0
  start_time: '2012-11-08 16:37:12.849988'
  stderr: ''
  stdout: ''
  testcase: test_one
  testsuite: /var/lib/utah/testsuites/utah_howto
  time_delta: '0:00:00.010241'
  user: utah
- cmd_type: testcase_build
  command: make
  returncode: 0
  start_time: '2012-11-08 16:37:12.874301'
  stderr: ''
  stdout: |-
    echo 'test -f /tmp/foo' > test_three.sh
    chmod +x test_three.sh
  testcase: test_three
  testsuite: /var/lib/utah/testsuites/utah_howto
  time_delta: '0:00:00.005345'
  user: root
- cmd_type: testcase_setup
  command: touch /tmp/foo
  returncode: 0
  start_time: '2012-11-08 16:37:12.889391'
  stderr: ''
  stdout: ''
  testcase: test_three
  testsuite: /var/lib/utah/testsuites/utah_howto
  time_delta: '0:00:00.002992'
  user: root
- cmd_type: testcase_test
  command: ./test_three.sh
  extra_info:
    action: |-
      1. Test that /tmp/foo exists
    dependencies: make
    description: Test that /tmp/foo exists
    expected_results: |-
      1. /tmp/foo indeed exists
  returncode: 0
  start_time: '2012-11-08 16:37:12.900769'
  stderr: ''
  stdout: ''
  testcase: test_three
  testsuite: /var/lib/utah/testsuites/utah_howto
  time_delta: '0:00:00.010171'
  user: utah
- cmd_type: testcase_cleanup
  command: rm /tmp/foo
  returncode: 0
  start_time: '2012-11-08 16:37:12.919748'
  stderr: ''
  stdout: ''
  testcase: test_three
  testsuite: /var/lib/utah/testsuites/utah_howto
```

```
86    time_delta: '0:00:00.002942'
87    user: root
88  errors: 0
89  failures: 0
90  fetch_errors: 0
91  install_type: desktop
92  media-info: Ubuntu 12.10 "Quantal Quetzal" - Release amd64 (20121017.5)
93  name: unnamed
94  passes: 2
95  ran_at: '2012-11-08 16:37:12.817425'
96  release: quantal
97  runlist: /tmp/utah_howto/master.run
98  uname:
99  - Linux
100 - xps8300
101 - 3.5.0-18-generic
102 - '#29-Ubuntu SMP Fri Oct 19 10:26:51 UTC 2012'
103 - x86_64
104 - x86_64
```

where:

- lines 51-52: there's a new setup step before executing the test case.
- lines 78-79: there's a new cleanup step after execution the test case.
- line 94: all test cases now pass.

**Todo**

Move the setup/cleanup code to the test suite to give an example about how to do the same thing at the suite level (useful when multiple test cases need the same configuration).

## 1.6 Timeout

**Todo**

Fix the formatting and provide more information about the example.

```
$ phoenix . test_four
```

Edit tc_control file:

```
   description: Sleep test
   despendencies: sleep
   action: |
    1. Sleep for 10 seconds
    expected_results: |
     system waits and returns 0
    command: sleep 10
    timeout: 5

- Run again
There's one failure because of the timeout
```

**Note:** Timeout returncode is -9 (process is killed). This is documented, but the yaml output file might explain this better in the future.

Different architectures? Override the timeout value in the master.run, so that the timeout value adjust to the target hardware.

- Edit master.run:

```
timeout: 15
```

(top level setting, not per test suite)

- Run again

Now we have three passes

CHAPTER 2

# PART II

1. Different architectures $ phoenix . test32 - Edit test32/tc_control description: Test for 32 bit systems dependencies: 32 bit system actions: |

1. Get hardware platform from uname

**expected_results: |** 1. Hardware platform is i386 or i686 type: userland timeout: 60 command: uname -i | grep -qE "i(3|6)86" run_as: utah

$ phoenix . test64 - Edit test64/tc_control description: Test for 64 bit systems dependencies: 64 bit system actions: |

1. Get hardware platform from uname

**expected_results: |** 1. Hardware platform is x86_64 type: userland timeout: 60 command: uname -i | grep -q "x86_64" run_as: utah

- Run

$ sudo utah -r master.run > output.yaml; view output.yaml - Look at the output test32 failed test64 passed (assuming you've got a 64 bits system) - Create a new master.run list for 32/64bits $ cp master.run utah32.run Edit file: - exclude_tests:

test64

$ cp master.run utah64.run - exclude_tests:

test32

Note: Right now utah isn't able to exclude automatically the tests. That could probaly be supported in the future using the dependencies field. 2. Version control $ bzr init $ bzr push lp:~<lp_username>/+junk/utah_howto - Create new runlist cp master.run launchpad.run - Edit: fetch_method: bzr fetch_location: lp:~<lp_username>/+junk/utah_howto - Run $ sudo utah -r launchpad.run > output.yaml; view output.yaml Note: This failed because "bzr branch" is executed as root. I should be possible to execute "bzr branch" as my user. - Use this url: https://code.launchpad.net/~<lp_username>/+junk/utah_howto Note: In a test machine in the lab, nobody wants to put his own ssh keys. - Look at the output It's the same as when running locally except for the fetch command. 3. Provisioning Note: Hardware virtualization recommended. With qemu will work as well, but it will be very slow (and timeouts will need to be overriden on some test cases). - Install the server $ sudo apt-get install utah - Explain what the server does $ run_utah_tests.py -h This cover virtual, physical, arm boards, etc. What we need for this example: - runlist: A positional argument $ run_utah_tests.py $HOME/launchpad.run Note: This will download the ISO (precise i386 desktop) $ run_utah_tests.py $HOME/launchpad.run -s quantal -a amd64 -t server Note: This will download the ISO (quantal server amd64) $ run_utah_tests.py $HOME/launchpad.run -i <path_to_iso> Note: <path_to_iso> can be a local path or an http url as well. $ run_utah_tests.py $HOME/launchpad.run -i http://archvie.ubuntu.com/ubuntu/dists/quantal/main/installer/images/netboot/mini.iso Download image - Unpack kernel, initrd - Create preseed - Create vm (around 30 minutes) Note: For now the VM is always created from an ISO. In the future, support for existing VMs might be provided. Note: Additional configuration for the VM XML can be passed to prevent VM disk caching (which might invalidate some disk tests). 4. Reboot tests Note: No to be used in your own laptop, but in a vm or in another device to be tested. $ phoenix . reboot_test - Edit tc_control file description: Create a file in /tmp dependencies: none action: |

1. Create /tmp/utah
2. Reboot if successful

**expected_results: |**

> 1. File is created
>
> 2. system reboots type userland timeout: 60 command: touch /tmp/utah run_as: utah reboot: pass # (always, never)

- Create another test case to be executed after the reboot

$ phoenix . post_reboot_test - Edit tc_control file description: Check that a reboot cleans up files in /tmp dependencies: reboot_test action: |

> 1. Check for /tmp/utah after reboot expected_results
>
> > 1. /tmp/utah does not exist

type: userland timeout: 60 command: ls /tmp/utah run_as: utah Note: tslist.run defines the ordering for test cases. User is expected to put the test cases in the right order so that the ones after the reboot are executed when they should. Note: master.run is supposed in a different location Note: provisioners - libvirt - cobbler - panda board (in the lab)

Note: Passing parameters to the test cases: - Option 1: Use an environment variable - Option 2: Generate a data file on the fly and read from it in the setup

# PART III

How to write good test cases: <link> - Let's work on an example: $ bzr branch lp:~utah/utah/utah_ls_example - Look at the ts_control file There are both a setup and a cleanup command - Look at ts_util.py *sys.path* used to import from *common* module - Look at the common module

- STATE_FILE, DATA_FILE defined in terms of the module's directory
- run_cmd used as a method to run a command and get stdout, stderr and returncode.
- setup_logging
- get_testfiles_data: Used to get the data used by the test cases
- Look at *data.json*: contains filenames and permissions (both in octal number and as string)
- Look again at ts_util.py - setup: create directory and files according to the information in the data file

Note: The creation of a tmp directory is something that will be commonly needed and worth having in a library. - Look at permissions/tc_control action and expected results describe what the test does. - Look at permissions.py

- sys.path used to import from common
- Using unittest module
- Look at runTest method
- Directory and files exist
- Permission string for every file is also correct
- Look at dotfiles/tc_control

action and expected results describe what the test does. - Look at dotfiles.py

- Directory exists
- Dot files are there

Note: We're not using the unittest runner on purpose. Note: Quite a lot of discussion of whether unittest only for the assertions should be a good practice because it's confusing. Question: Do you have a skeleton file to encourage users to follow best practice? - Edit master.run - Run Two test case passes Feature request: sudo utah -r . (Run test suite and cases without any master.run file)

CHAPTER 4

# PART IV

Discussion about writing test cases for UTAH

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search