User Guide for Web Testing Service Setting Up

Web Testing Service Overview

The Web Testing Service (WTS) is a web server that hosts the comprehensive web test suites and supports users to run them online through Browsers or Web Runtimes. Its backend also provides the web servers such as wpt server, web socket server and some test media content; its frontend is a friendly web page to customize tests, control testing, and show test results.
Figure 1 shows the Service architecture

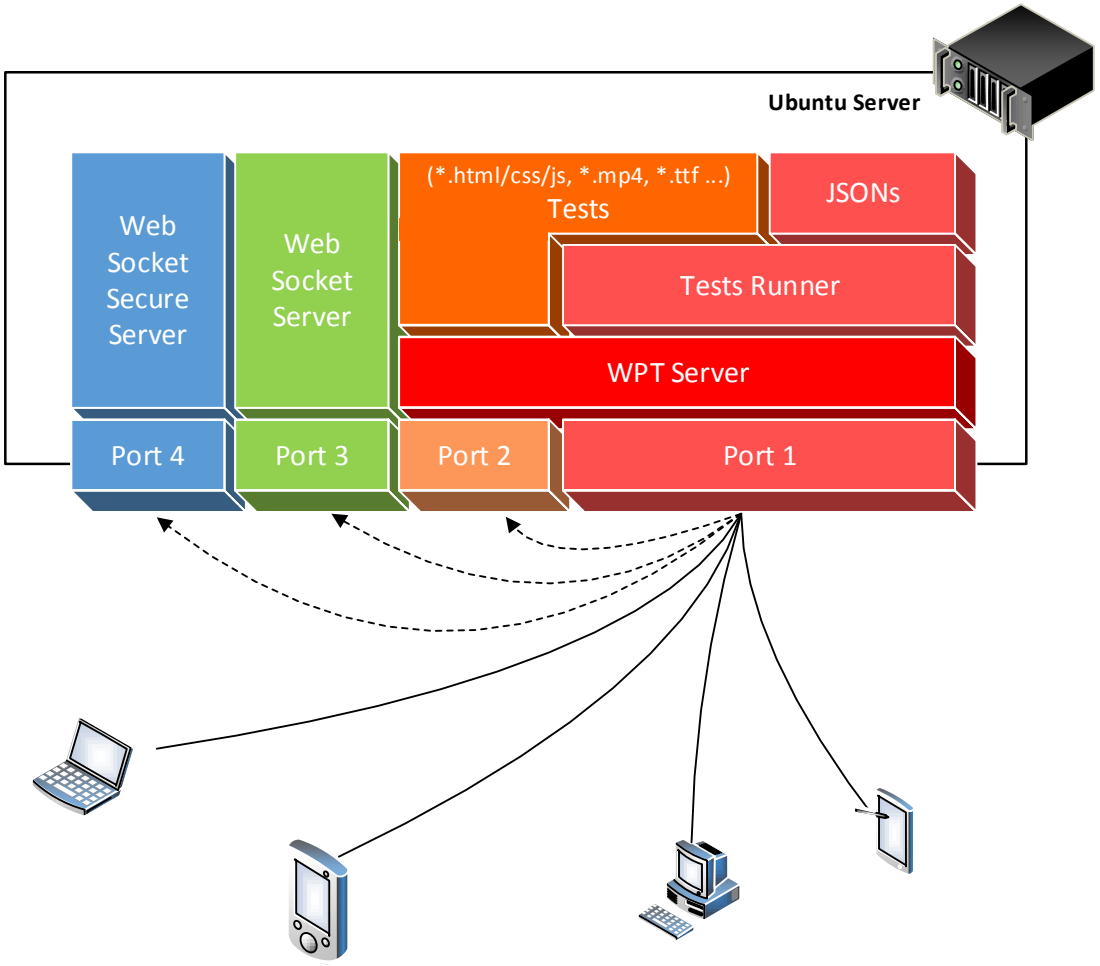

Figure 1 WTS Architecture

Prerequisites

Before setting up a WTS, ensure that the following items are met:

1. You have obtained sudo rights
2. Suggested Hardware
   - CPU: CPU version (cpu64-rhel6) @ 3.0 GHz * 4
   - Memory: at least 4 GB
   - Hard Disk: at least 200 GB
3. Software environment (Ubuntu 12.04 LTS)
   - Python 2.7.3: Ubuntu build-in
4. Network environment
   - At least 4 ports for service public accessing

Steps

To set up a WTS, perform the following steps:

1. Install GIT tool

```
$ sudo apt-get install git
```

2. Download the service source by git

```
$ sudo cd path-to-service/
$ sudo git clone git@github.com:crosswalk-project/web-testing-service.git
```

3. Config the service
   - Change the host name of the path-to-service/web-testing-service/config.json

```
{"host": "hostname-of-server",
  "ports":{"http":[80, 8000],
                "ws":[8001],
                "wss":[8002]},
  "log_level":"debug",
  "bind_hostname": true}
```

Hostname-of-server is the server name, e.g. wts.sh.intel.com

   - Change the 3 ports of the path-to-service/web-testing-service/config.json

```
{"host": "hostname-of-server",
  "ports":{"http":[port-1, port-2],
                "ws":[port-3],
                "wss":[port-4]},
  "log_level":"debug",
  "bind_hostname": true}
```

Port-1 is for main accessing, normally, it is “80” port. Port-2 is for cross domains testing, some tests of the service need access this port to check cross domains from “80” port. Port-3 is for Web Socket testing. Port-4 is reserved for security Web Socket testing now.

4. Start the service

```
$ sudo path-to-service/web-testing-service/wts.sh start
```

5. Verify WTS deployment in browser

Log in to http://<hostname-of-server>:<port-1>/. The test runner displays, as shown in Figure 2.

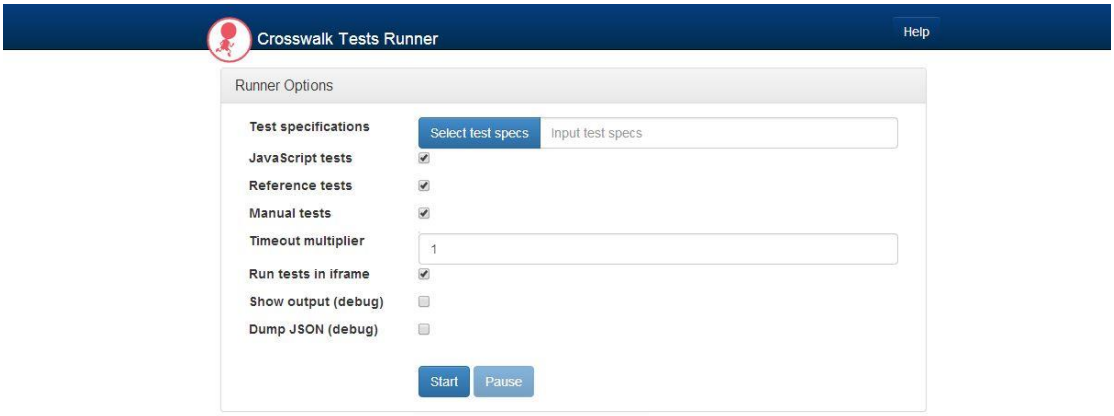

Figure 1 WTS Test Runner

6. Stop/restart/upgrade the service

```
$ sudo path-to-service/web-testing-service/wts.sh stop
$ sudo path-to-service/web-testing-service/wts.sh restart
$ sudo path-to-service/web-testing-service/wts.sh upgrade
```

7. Upgrade the service automatically

To support automatic upgrading, just need add an item as following to “/etc/crontab”:

```
schedule-settings root path-to-service/web-testing-service/wts.sh upgrade 1> /tmp/wts.log 2>&1
```

Update “schedule-settings” filed to follow actual requirements, e.g. weekly to update service:

```
1 1 * * 7 root path-to-service/web-testing-service/wts.sh upgrade 1> /tmp/wts.log 2>&1
```

In wts.sh, the root user will run “git pull” to download the service code and restart the service automatically.