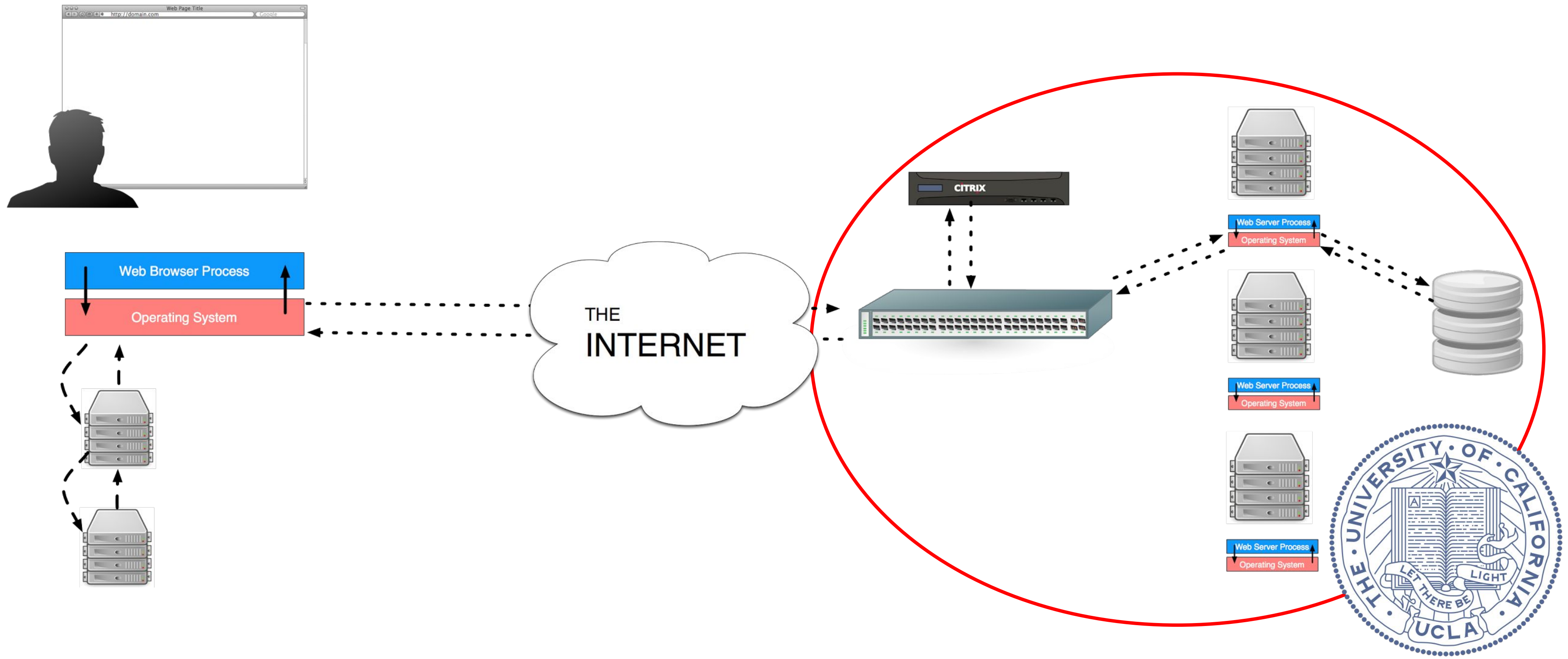
CS 188

Scalable Internet Services

Andrew Mutz
October 27, 2016

For Today


Web Page Title
http://domain.com
Web Browser Process
Operating System
THE
INTERNET
CITRIX
Web Server Process
Operating System
Web Server Process
Operating System
Web Server Process
Operating System
THE UNIVERSITY OF CALIFORNIA
LET THERE BE LIGHT
UCLA

Motivation

After today you will know how to evaluate the scalability of a deployed application using Tsung.

Today will be interactive, so if you’ve brought your laptops, please get them out.

Motivation

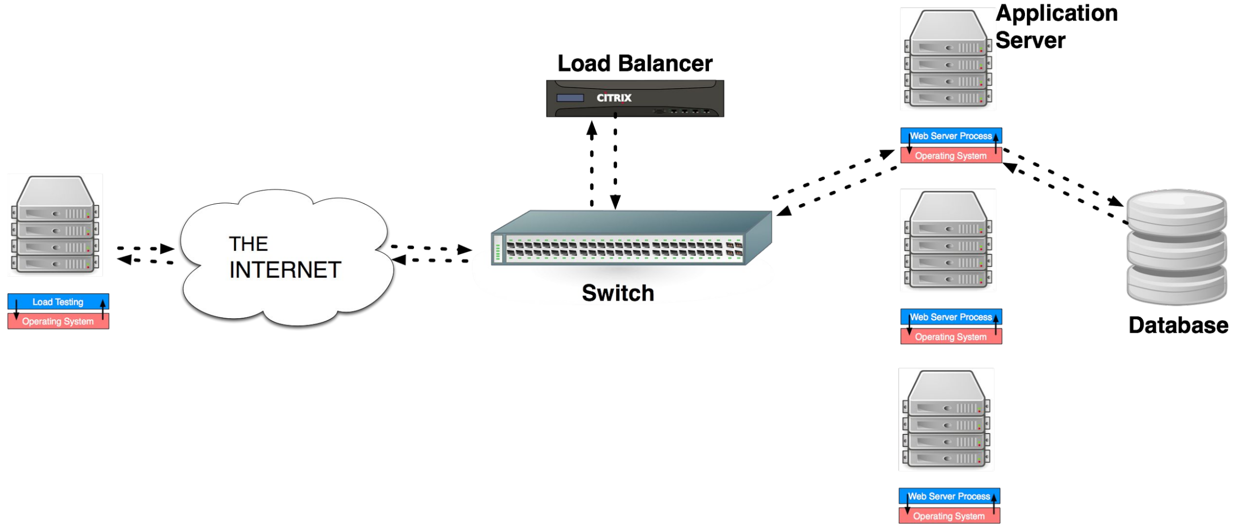
Let’s say we are considering a significant change to our web application.

We think it may improve scalability, but our intuition might be wrong.

How should we go about testing this?

Motivation

Let’s deploy the system, send actual requests to it, and watch how it responds.



Motivation

What should we observe?

- Response times
- Error rates
- Are our synthetic users able to finish their tasks?

Motivation

Some observations:

- We want a mixture of reads and writes
  - Why is this important?
- Not all users have the same habits
  - Why is this important?
- We want to be able to respond to application output.
  - When is this important?

Motivation

Some load testing tools have high performance

- apachebench, httperf

Some tools have rich feature sets

- Funkload

Tsung is a good combination of the two

Motivation

We will want one app and testing instance per team. So please:

- Sit with your team
- Deploy one micro instance of your app on EC2
- Deploy one Tsung instance per team
    - I will explain how to do this on the next slide.

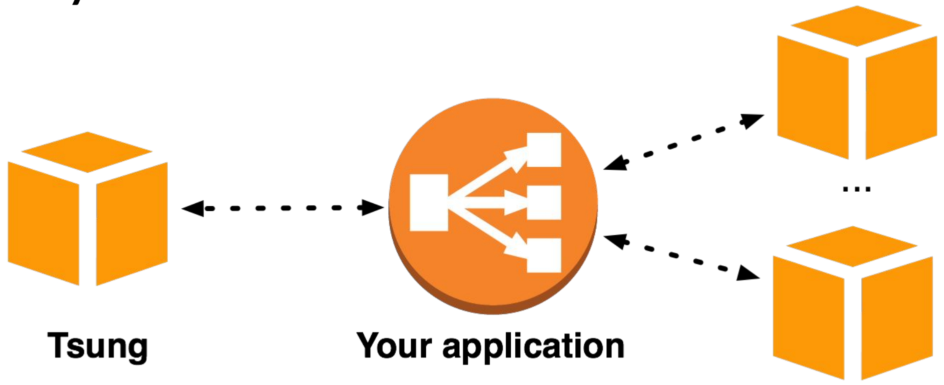
Tsung

Why Tsung?

- There are many load testing tools out there
- Most make you choose between flexibility and performance
- If you use a low-performance tool, you need to deploy a fleet of machines to do load testing
- Tsung is extremely configurable and delivers high performance

Tsung

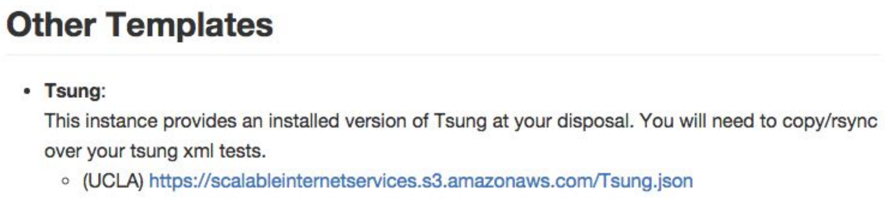
- We will do all load testing within AWS.
  - Saves money
  - More predictable
- Don’t use T-series (t1 micro, etc) for measurement.
  - Why?



Tsung

Starting up an instance of Tsung testing is easy.

- Create a new CF stack and name it something like TEAM_NAME-tsung
- Use the Tsung template from the utils repo:



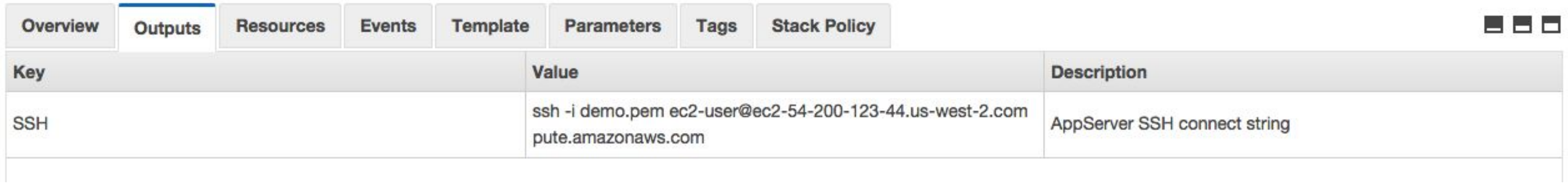
Other Templates

- **Tsung**:
  This instance provides an installed version of Tsung at your disposal. You will need to copy/rsync over your tsung xml tests.
  - (UCLA) https://scalableinternetservices.s3.amazonaws.com/Tsung.json

- Select your TeamName
- Turn off rollback.

Tsung

Once your Tsung stack is up, you will see important information in the “Outputs section”:

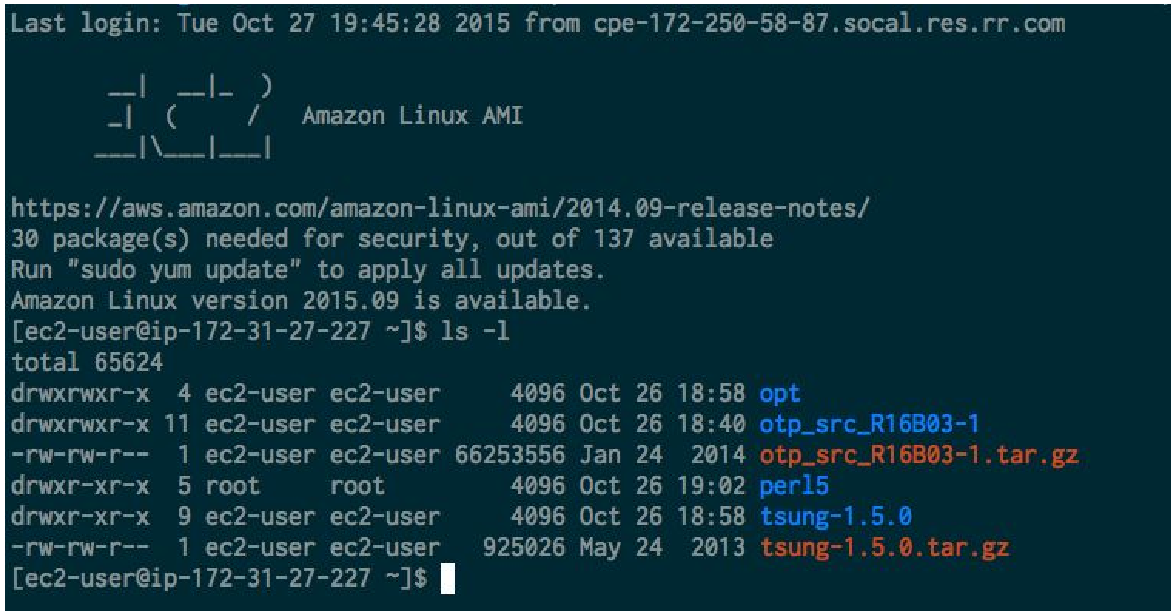


Key	Value	Description
SSH	ssh -i demo.pem ec2-user@ec2-54-200-123-44.us-west-2.compute.amazonaws.com	AppServer SSH connect string

- SSH tells you how to SSH into your Tsung instance
- The machine address tells you where to go to see results via HTTP

Tsung

Let's SSH into our tsung instance



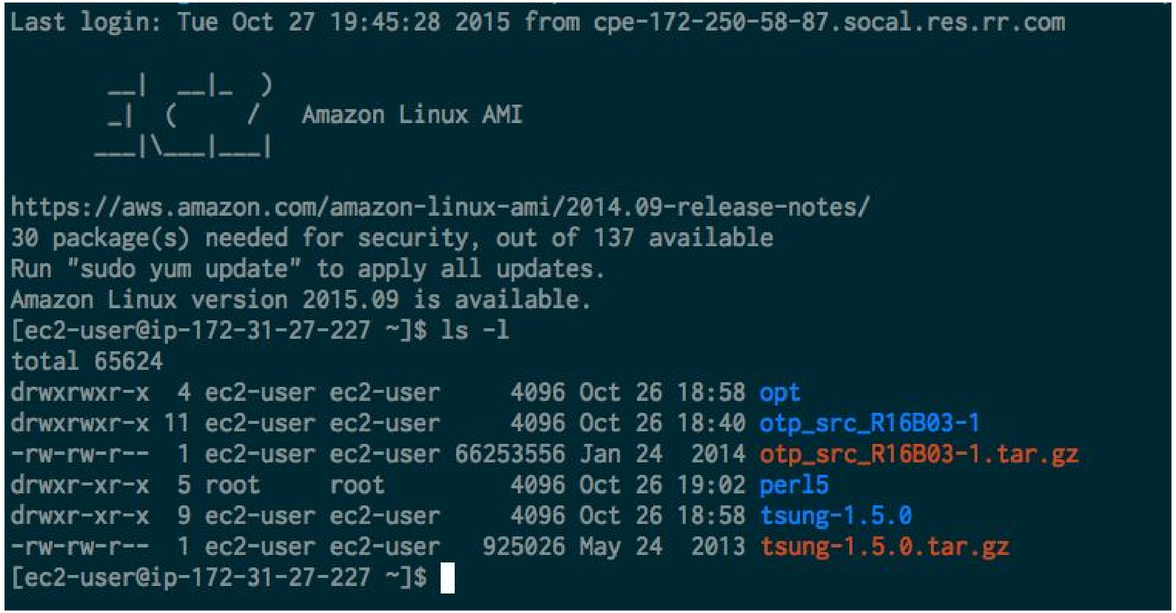
```
Last login: Tue Oct 27 19:45:28 2015 from cpe-172-250-58-87.socal.res.rr.com

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
30 package(s) needed for security, out of 137 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2015.09 is available.
[ec2-user@ip-172-31-27-227 ~]$ ls -l
total 65624
drwxrwxr-x  4 ec2-user ec2-user     4096 Oct 26 18:58 opt
drwxrwxr-x 11 ec2-user ec2-user     4096 Oct 26 18:40 otp_src_R16B03-1
-rw-rw-r--  1 ec2-user ec2-user 66253556 Jan 24  2014 otp_src_R16B03-1.tar.gz
drwxr-xr-x  5 root     root         4096 Oct 26 19:02 perl5
drwxr-xr-x  9 ec2-user ec2-user     4096 Oct 26 18:58 tsung-1.5.0
-rw-rw-r--  1 ec2-user ec2-user   925026 May 24  2013 tsung-1.5.0.tar.gz
[ec2-user@ip-172-31-27-227 ~]$
```

Tsung

opt, otp_src_R16B03-1, otp_src_R16B03-1.tar.gz, tsung-1.5.0, tsung-1.5.0.tar.gz can be ignored. They are the installed version of erlang and tsung.



```
Last login: Tue Oct 27 19:45:28 2015 from cpe-172-250-58-87.socal.res.rr.com

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
30 package(s) needed for security, out of 137 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2015.09 is available.
[ec2-user@ip-172-31-27-227 ~]$ ls -l
total 65624
drwxrwxr-x  4 ec2-user ec2-user     4096 Oct 26 18:58 opt
drwxrwxr-x 11 ec2-user ec2-user     4096 Oct 26 18:40 otp_src_R16B03-1
-rw-rw-r--  1 ec2-user ec2-user 66253556 Jan 24  2014 otp_src_R16B03-1.tar.gz
drwxr-xr-x  5 root     root         4096 Oct 26 19:02 perl5
drwxr-xr-x  9 ec2-user ec2-user     4096 Oct 26 18:58 tsung-1.5.0
-rw-rw-r--  1 ec2-user ec2-user   925026 May 24  2013 tsung-1.5.0.tar.gz
[ec2-user@ip-172-31-27-227 ~]$ 
```

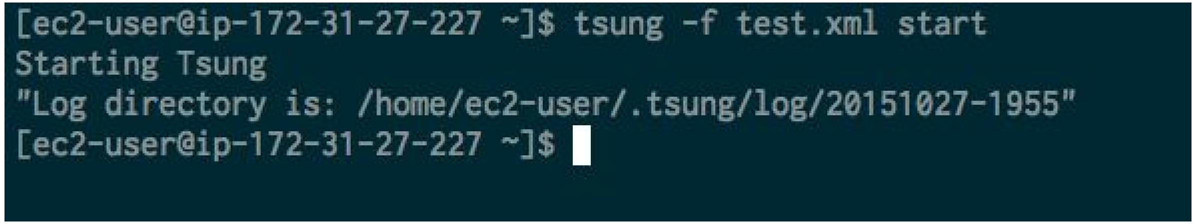
Tsung

Clone your repo to the load testing instance so you can get at your load testing scripts:

```
git clone git@github.com:scalableinternetservices/demo.git
```

Let's kick the tires by testing how www.google.com responds to light load:

```
tsung -f test.xml start
```

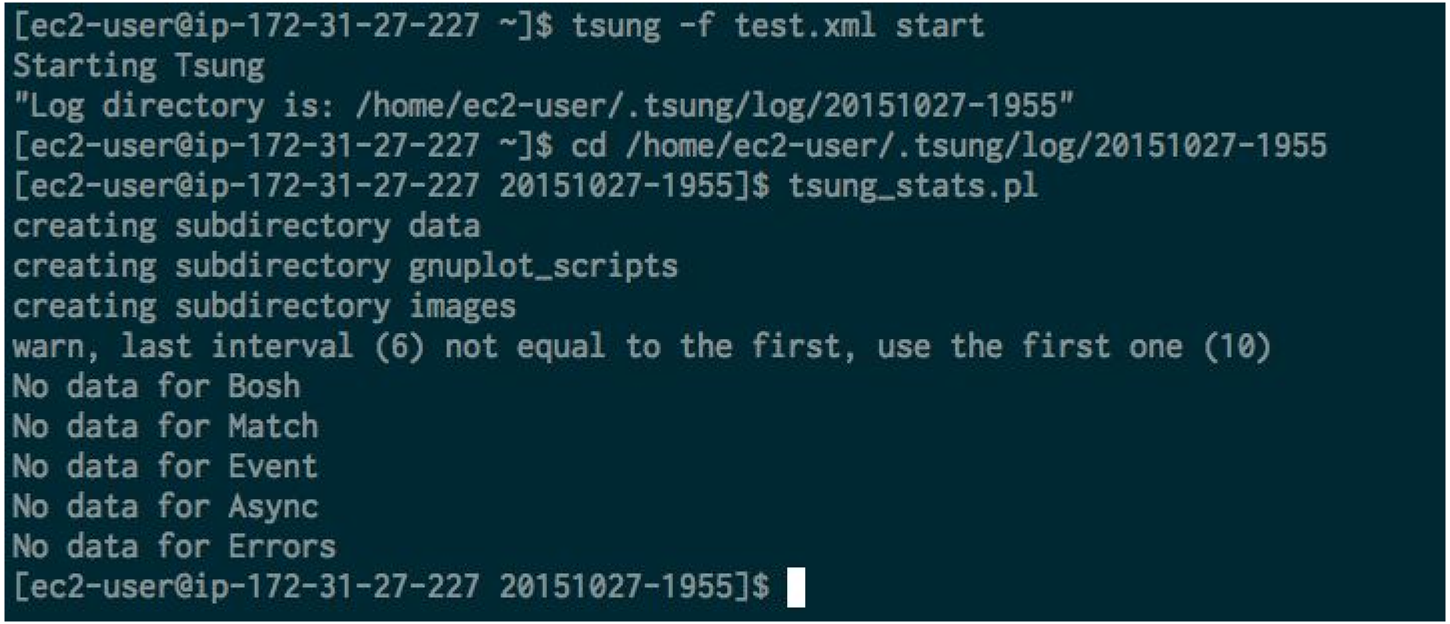


```
[ec2-user@ip-172-31-27-227 ~]$ tsung -f test.xml start
Starting Tsung
"Log directory is: /home/ec2-user/.tsung/log/20151027-1955"
[ec2-user@ip-172-31-27-227 ~]$
```

Tsung

This will take a little time to run. Afterwards we compile the report:

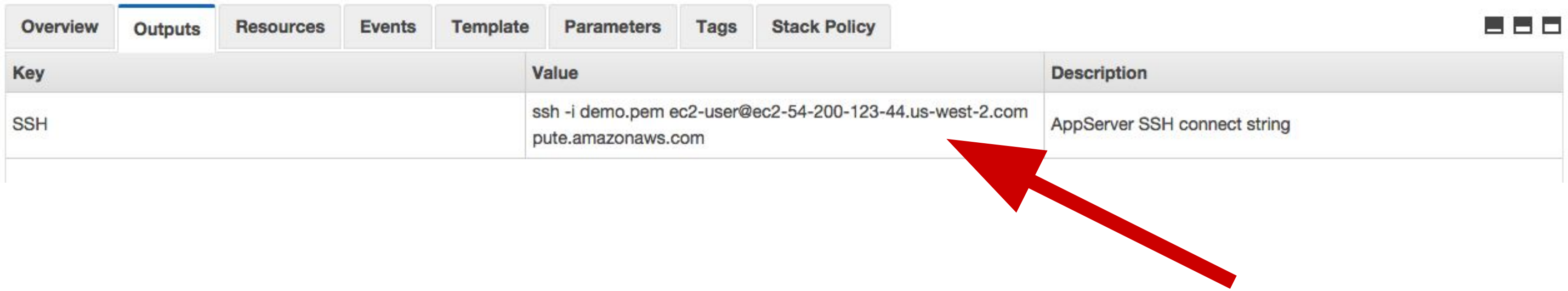
```
cd /home/ec2-user/.tsung/log/20150504-0444; tsung_stats.pl
```



```
[ec2-user@ip-172-31-27-227 ~]$ tsung -f test.xml start
Starting Tsung
"Log directory is: /home/ec2-user/.tsung/log/20151027-1955"
[ec2-user@ip-172-31-27-227 ~]$ cd /home/ec2-user/.tsung/log/20151027-1955
[ec2-user@ip-172-31-27-227 20151027-1955]$ tsung_stats.pl
creating subdirectory data
creating subdirectory gnuplot_scripts
creating subdirectory images
warn, last interval (6) not equal to the first, use the first one (10)
No data for Bosh
No data for Match
No data for Event
No data for Async
No data for Errors
[ec2-user@ip-172-31-27-227 20151027-1955]$
```

Tsung

Once we have compiled the report, switch over to the web interface:

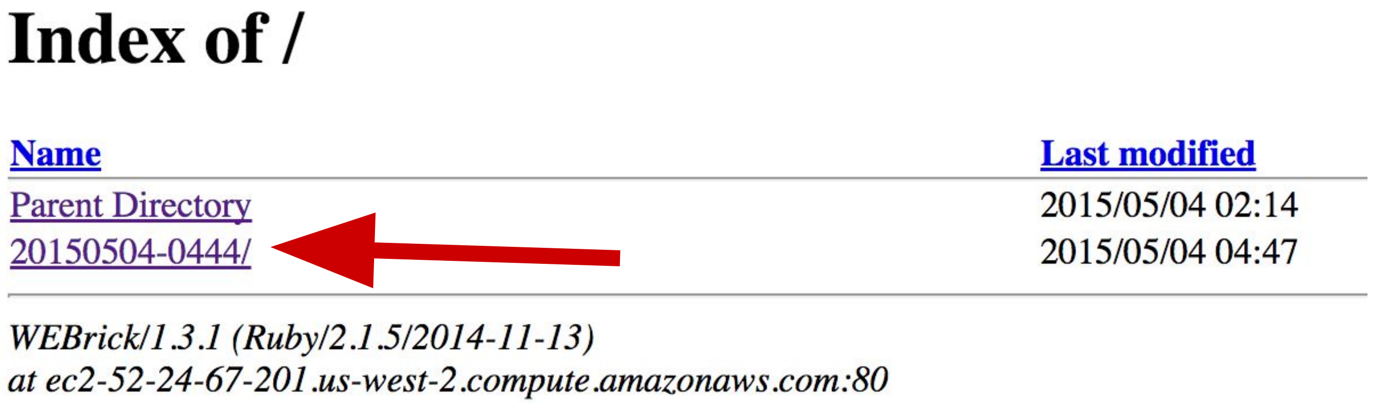


Key	Value	Description
SSH	ssh -i demo.pem ec2-user@ec2-54-200-123-44.us-west-2.com pute.amazonaws.com	AppServer SSH connect string

This just serves up the files from ~/.tsung/log/ through a web interface

Tsung

Once we have compiled the report, switch over to the web interface:



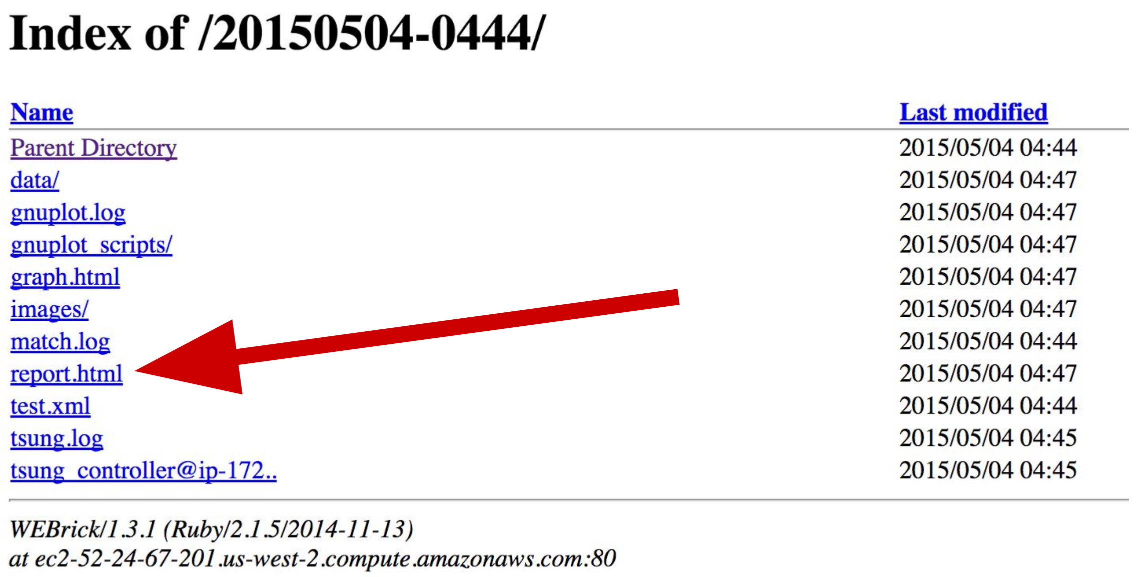
Index of /

Name	Last modified
Parent Directory	2015/05/04 02:14
20150504-0444/	2015/05/04 04:47

WEBrick/1.3.1 (Ruby/2.1.5/2014-11-13)
at ec2-52-24-67-201.us-west-2.compute.amazonaws.com:80

Tsung

Once we have compiled the report, switch over to the web interface:



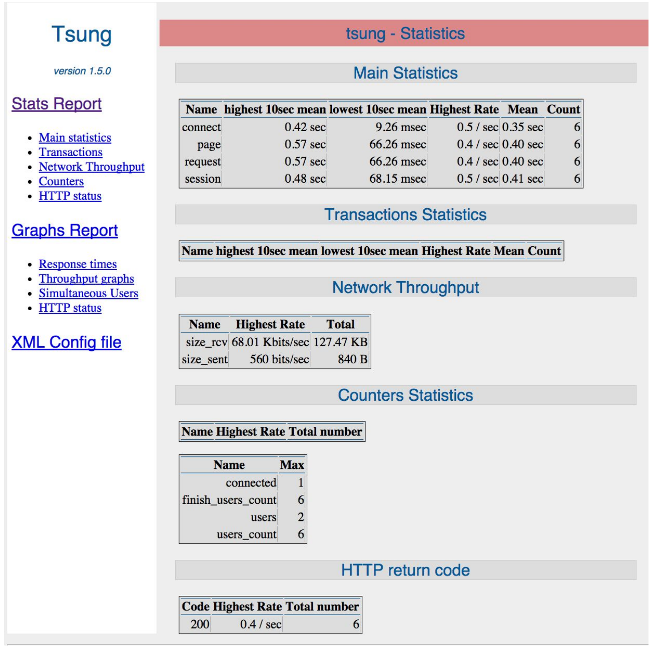
```
Index of /20150504-0444/

Name                                          Last modified
Parent Directory                              2015/05/04 04:44
data/                                         2015/05/04 04:47
gnuplot.log                                   2015/05/04 04:47
gnuplot_scripts/                              2015/05/04 04:47
graph.html                                    2015/05/04 04:47
images/                                       2015/05/04 04:47
match.log                                     2015/05/04 04:44
report.html                                   2015/05/04 04:47
test.xml                                      2015/05/04 04:44
tsung.log                                     2015/05/04 04:45
tsung_controller@ip-172..                     2015/05/04 04:45

WEBrick/1.3.1 (Ruby/2.1.5/2014-11-13)
at ec2-52-24-67-201.us-west-2.compute.amazonaws.com:80
```

Tsung

Once we have compiled the report, switch over to the web interface:



Tsung

version 1.5.0

Stats Report

- Main statistics
- Transactions
- Network Throughput
- Counters
- HTTP status

Graphs Report

- Response times
- Throughput graphs
- Simultaneous Users
- HTTP status

XML Config file

tsung - Statistics

Main Statistics

Name	highest 10sec mean	lowest 10sec mean	Highest Rate	Mean	Count
connect	0.42 sec	9.26 msec	0.5 / sec	0.35 sec	6
page	0.57 sec	66.26 msec	0.4 / sec	0.40 sec	6
request	0.57 sec	66.26 msec	0.4 / sec	0.40 sec	6
session	0.48 sec	68.15 msec	0.5 / sec	0.41 sec	6

Transactions Statistics

Name	highest 10sec mean	lowest 10sec mean	Highest Rate	Mean	Count

Network Throughput

Name	Highest Rate	Total
size_rcv	68.01 Kbits/sec	127.47 KB
size_sent	560 bits/sec	840 B

Counters Statistics

Name	Highest Rate	Total number

Name	Max
connected	1
finish_users_count	6
users	2
users_count	6

HTTP return code

Code	Highest Rate	Total number
200	0.4 / sec	6

We will go over this report in more detail later today

Tsung - test.xml

Let’s go over the basic xml file:

```
<?xml version="1.0"?><tsung loglevel="notice" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true" maxusers="15000"/>
  </clients>
  <servers>
    <server host="www.google.com" port="80" type="tcp"/>
  </servers>
  <load>
    <arrivalphase phase="1" duration="10" unit="second">
      <users arrivalrate="1" unit="second"/>
    </arrivalphase>
  </load>
```

Tsung - test.xml

XML Boilerplate

```
<?xml version="1.0"?><tsung loglevel="notice" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true" maxusers="15000"/>
  </clients>
  <servers>
    <server host="www.google.com" port="80" type="tcp"/>
  </servers>
  <load>
    <arrivalphase phase="1" duration="10" unit="second">
      <users arrivalrate="1" unit="second"/>
    </arrivalphase>
  </load>
```

Tsung - test.xml

Client-side configuration. Maxusers is the maximum number of simulated users.

```
<?xml version="1.0"?><tsung loglevel="notice" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true" maxusers="15000"/>
  </clients>
  <servers>
    <server host="www.google.com" port="80" type="tcp"/>
  </servers>
  <load>
    <arrivalphase phase="1" duration="10" unit="second">
      <users arrivalrate="1" unit="second"/>
    </arrivalphase>
  </load>
```

Tsung - test.xml

Server configuration: where we are directing load.

```
<?xml version="1.0"?><tsung loglevel="notice" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true" maxusers="15000"/>
  </clients>
  <servers>
    <server host="www.google.com" port="80" type="tcp"/>
  </servers>
  <load>
    <arrivalphase phase="1" duration="10" unit="second">
      <users arrivalrate="1" unit="second"/>
    </arrivalphase>
  </load>
```

Tsung - test.xml

Defining phases of user arrival rates

```
<?xml version="1.0"?><tsung loglevel="notice" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true" maxusers="15000"/>
  </clients>
  <servers>
    <server host="www.google.com" port="80" type="tcp"/>
  </servers>
  <load>
    <arrivalphase phase="1" duration="10" unit="second">
      <users arrivalrate="1" unit="second"/>
    </arrivalphase>
  </load>
```

Tsung - test.xml

A section to set options (timeouts, useragents)

```
<options>
    <option name="glocal_ack_timeout" value="2000"/>
    <option type="ts_http" name="user_agent">
      <user_agent probability="100">Mozilla/5.0 (Windows; U; Windows NT 5.2; fr-FR; rv:1.7.8) Gecko/20050511
Firefox/1.0.4</user_agent>
    </option>
  </options>
  <sessions>
    <session name="http-example" probability="100" type="ts_http">
      <request>
        <http url="/" version="1.1" method="GET"/>
      </request>
    </session>
  </sessions>
</tsung>
```

Tsung - test.xml

We define the actual series of requests that a user will perform.

```
<options>
    <option name="glocal_ack_timeout" value="2000"/>
    <option type="ts_http" name="user_agent">
      <user_agent probability="100">Mozilla/5.0 (Windows; U; Windows NT 5.2; fr-FR; rv:1.7.8) Gecko/20050511 
Firefox/1.0.4</user_agent>
    </option>
  </options>
  <sessions>
    <session name="http-example" probability="100" type="ts_http">
      <request>
        <http url="/" version="1.1" method="GET"/>
      </request>
    </session>
  </sessions>
</tsung>
```

Tsung - test.xml

Lets change our tests to point to our server

```
<?xml version="1.0"?><tsung loglevel="notice" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true" maxusers="15000"/>
  </clients>
  <servers>
    <server host="ec2-52-24-120-129.us-west-2.compute.amazonaws.com" port="80"
type="tcp"/>
  </servers>
  <load>
    <arrivalphase phase="1" duration="10" unit="second">
      <users arrivalrate="1" unit="second"/>
    </arrivalphase>
  </load>
```

Tsung - test.xml

```
 <load>
  <arrivalphase phase="1" duration="30" unit="second">
    <users arrivalrate="1" unit="second"></users>
  </arrivalphase>
  <arrivalphase phase="2" duration="30" unit="second">
    <users arrivalrate="2" unit="second"></users>
  </arrivalphase>
  <arrivalphase phase="3" duration="30" unit="second">
    <users arrivalrate="4" unit="second"></users>
  </arrivalphase>
 <arrivalphase phase="4" duration="30" unit="second">
   <users arrivalrate="8" unit="second"></users>
 </arrivalphase>
</load>
```

For examples of more complex load testing, see the demo app’s `load_tests/critical.xml`

Increasing the rate of user creation over time.

Tsung - test.xml

```
 <load>
  <arrivalphase phase="1" duration="30" unit="second">
    <users arrivalrate="1" unit="second"></users>
  </arrivalphase>
  <arrivalphase phase="2" duration="30" unit="second">
    <users arrivalrate="2" unit="second"></users>
  </arrivalphase>
  <arrivalphase phase="3" duration="30" unit="second">
    <users arrivalrate="4" unit="second"></users>
  </arrivalphase>
 <arrivalphase phase="4" duration="30" unit="second">
   <users arrivalrate="8" unit="second"></users>
 </arrivalphase>
</load>
```

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

Increasing the rate of user creation over time.

Tsung - test.xml

```
<sessions>
  <session name="http-example" probability="100" type="ts_http">

    <!-- start out at the dashboard. -->
    <request>
      <http url='/' version='1.1' method='GET'></http>
    </request>
```

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

The session defines the virtual user that you will be simulating.

This one starts at "/"

Tsung - test.xml

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

```
<!-- wait for up to 2 seconds, user is looking at posts -->
<thinktime value="2" random="true"></thinktime>
```

Insert realistic random wait times in your simulations.

Tsung - test.xml

```
      <!-- create a random number to make a unique community name -->
      <setdynvars sourcetype="random_number" start="1000" 
end="9999999">
        <var name="random_community_name" />
      </setdynvars>


      <!-- post to /communities to create a community.
           remember the url of the created community so we can delete 
it later -->
      <request subst="true">
        <http
          url='/communities'
          version='1.1'
          method='POST'
contents='community%5Bname%5D=community_name_%%_random_community_name%%
&amp;commit=Create+Community'>
        </http>
      </request>
```

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

Deal with uniqueness constraints by defining dynamic variables.

Insert dynamic variables by using %% syntax.

Tsung - test.xml

```
      <request subst="true">
        <dyn_variable name="created_submission_url" re="[Ll]ocation:
(http://.*)\r"/>
        <dyn_variable name="created_submission_id" re="[Ll]ocation:
http://.*/submissions/(.*)\r"/>
        <http
          url='/submissions'
          version='1.1'
          method='POST'

contents='submission%5Btitle%5D=link_%%_random_submission_name%%&amp;su
bmission%5Burl%5D=http%3A%2F%2Fwww.article.com%2F%%_random_submission_n
ame%%&amp;submission%5Bcommunity_id%5D=%%_created_community_id%%&amp;co
mmit=Create+Submission'>
        </http>
      </request>
```

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

At times, you will need the output of one request to feed into the next.

Use dynamic variables for this

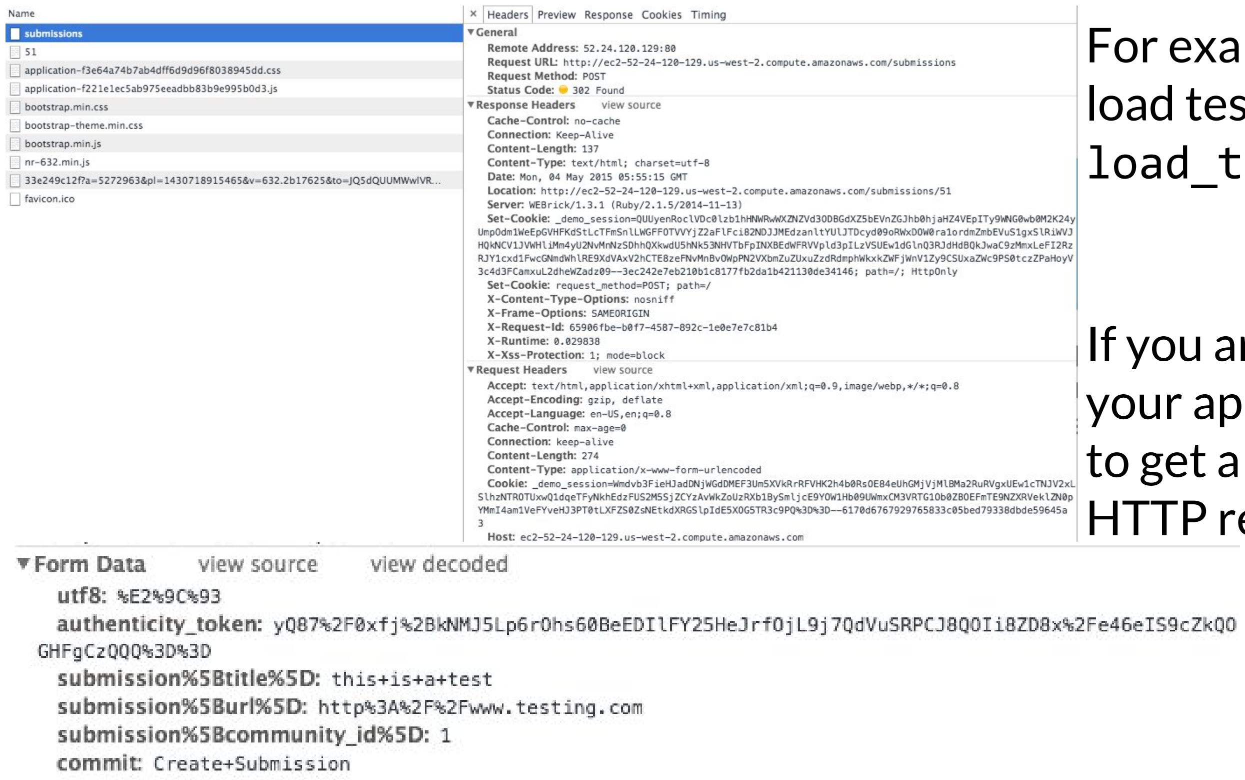
Tsung - test.xml

```
  <!-- Uncomment the following to debug print your dynamic variables
-->
      <!--
      <setdynvars sourcetype="eval" code="fun( {Pid, DynVars} ) ->
        io:format([126, $p, 126, $n, 126, $n], [DynVars]),
        ok end.">
        <var name="dump" />
      </setdynvars>
      -->
```

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

If you have difficulty with your dynamic variables, use this code to debug

Tsung - test.xml



```
Name
submissions
51
application-f3e64a74b7ab4dff6d9d96f8038945dd.css
application-f221e1ec5ab975eeadbb83b9e995b0d3.js
bootstrap.min.css
bootstrap-theme.min.css
bootstrap.min.js
nr-632.min.js
33e249c12f?a=5272963&pl=1430718915465&v=632.2b17625&to=JQ5dQUUMWwlVR...
favicon.ico
```

```
× Headers  Preview  Response  Cookies  Timing
▼General
   Remote Address: 52.24.120.129:80
   Request URL: http://ec2-52-24-120-129.us-west-2.compute.amazonaws.com/submissions
   Request Method: POST
   Status Code: ● 302 Found
▼Response Headers     view source
   Cache-Control: no-cache
   Connection: Keep-Alive
   Content-Length: 137
   Content-Type: text/html; charset=utf-8
   Date: Mon, 04 May 2015 05:55:15 GMT
   Location: http://ec2-52-24-120-129.us-west-2.compute.amazonaws.com/submissions/51
   Server: WEBrick/1.3.1 (Ruby/2.1.5/2014-11-13)
   Set-Cookie: _demo_session=QUUyenRoclVDc0lzb1hHNWRwWXZNZVd3ODBGdXZ5bEVnZGJhb0hjaHZ4VEpITy9WNG0wb0M2K24y
 UmpOdm1WeEpGVHFKdStLcTFmSnlLWGFFOTVVYjZ2aFlFci82NDJJMEdzanltYUlJTDcyd09oRWxDOW0ra1ordmZmbEVuS1gxSlRiWVJ
 HQkNCV1JVWHliMm4yU2NvMnNzSDhhQXkwdU5hNk53NHVTbFpINXBEdWFRVVpld3pILzVSUEw1dGlnQ3RJdHdBQkJwaC9zMmxLeFI2Rz
 RJY1cxd1FwcGNmdWhlRE9XdVAxV2hCTE8zeFNvMnBvOWpPN2VXbmZuZUxuZzdRdmphWkxkZWFjWnV1Zy9CSUxaZWc9PS0tczZPaHoyV
 3c4d3FCamxuL2dheWZadz09--3ec242e7eb210b1c8177fb2da1b421130de34146; path=/; HttpOnly
   Set-Cookie: request_method=POST; path=/
   X-Content-Type-Options: nosniff
   X-Frame-Options: SAMEORIGIN
   X-Request-Id: 65906fbe-b0f7-4587-892c-1e0e7e7c81b4
   X-Runtime: 0.029838
   X-Xss-Protection: 1; mode=block
▼Request Headers     view source
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
   Accept-Encoding: gzip, deflate
   Accept-Language: en-US,en;q=0.8
   Cache-Control: max-age=0
   Connection: keep-alive
   Content-Length: 274
   Content-Type: application/x-www-form-urlencoded
   Cookie: _demo_session=Wmdvb3FieHJadDNjWGdDMEF3Um5XVkRrRFVHK2h4b0RsOE84eUhGMjVjMlBMa2RuRVgxUEw1cTNJV2xL
 SlhzNTROTUxwQ1dqeTFyNkhEdzFUS2M5SjZCYzAvWkZoUzRXb1BySmljcE9YOW1Hb09UWmxCM3VRTG1Ob0ZBOEFmTE9NZXRVeklZN0p
 YMmI4am1VeFYveHJ3PT0tLXFZS0ZsNEtkdXRGSlpIdE5XOG5TR3c9PQ%3D%3D--6170d6767929765833c05bed79338dbde59645a
 3
   Host: ec2-52-24-120-129.us-west-2.compute.amazonaws.com
```

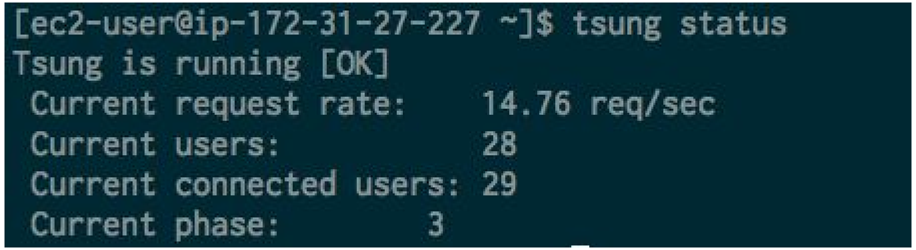
```
▼Form Data     view source     view decoded
   utf8: %E2%9C%93
   authenticity_token: yQ87%2F0xfj%2BkNMJ5Lp6rOhs60BeEDIlFY25HeJrfOjL9j7QdVuSRPCJ8QOIi8ZD8x%2Fe46eIS9cZkQO
 GHFgCzQQQ%3D%3D
   submission%5Btitle%5D: this+is+a+test
   submission%5Burl%5D: http%3A%2F%2Fwww.testing.com
   submission%5Bcommunity_id%5D: 1
   commit: Create+Submission
```

For examples of more complex load testing, see the demo app's `load_tests/critical.xml`

If you are unsure how to simulate your application, use the browser to get a firm idea of the exact HTTP requests.

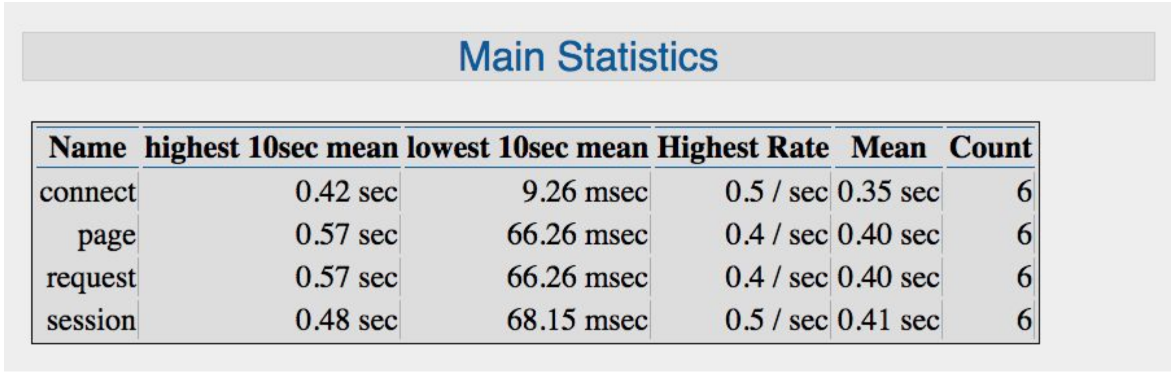
Tsung - test.xml

At any point in time you can see how tsung is doing, by typing “tsung status” from another terminal



```
[ec2-user@ip-172-31-27-227 ~]$ tsung status
Tsung is running [OK]
 Current request rate:    14.76 req/sec
 Current users:           28
 Current connected users: 29
 Current phase:         3
```

Tsung - Output

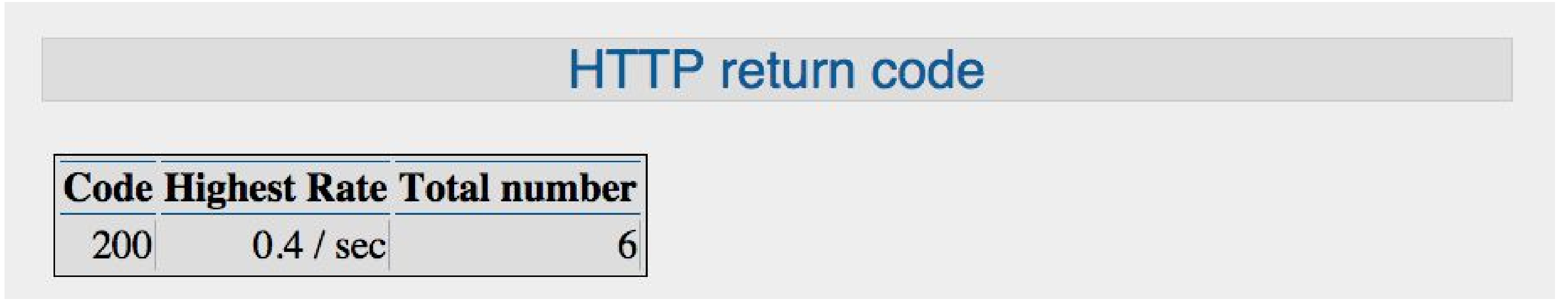


Main Statistics

Name	highest 10sec mean	lowest 10sec mean	Highest Rate	Mean	Count
connect	0.42 sec	9.26 msec	0.5 / sec	0.35 sec	6
page	0.57 sec	66.26 msec	0.4 / sec	0.40 sec	6
request	0.57 sec	66.26 msec	0.4 / sec	0.40 sec	6
session	0.48 sec	68.15 msec	0.5 / sec	0.41 sec	6

- **request** Response time for each request.
- **page** Response time for each set of requests (a page is a group of request not separated by a thinktime).
- **connect** Duration of the connection establishment.
- **session** Duration of a user's session.

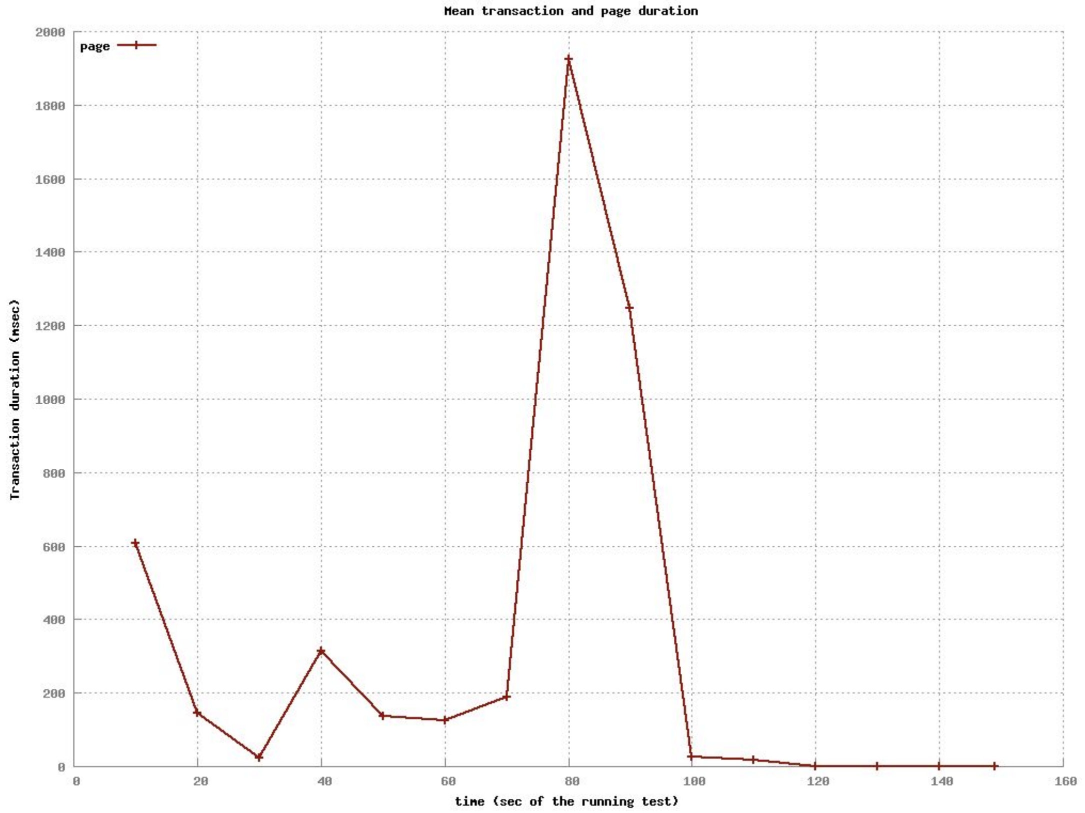
Tsung - Output


HTTP return code

Code	Highest Rate	Total number
200	0.4 / sec	6

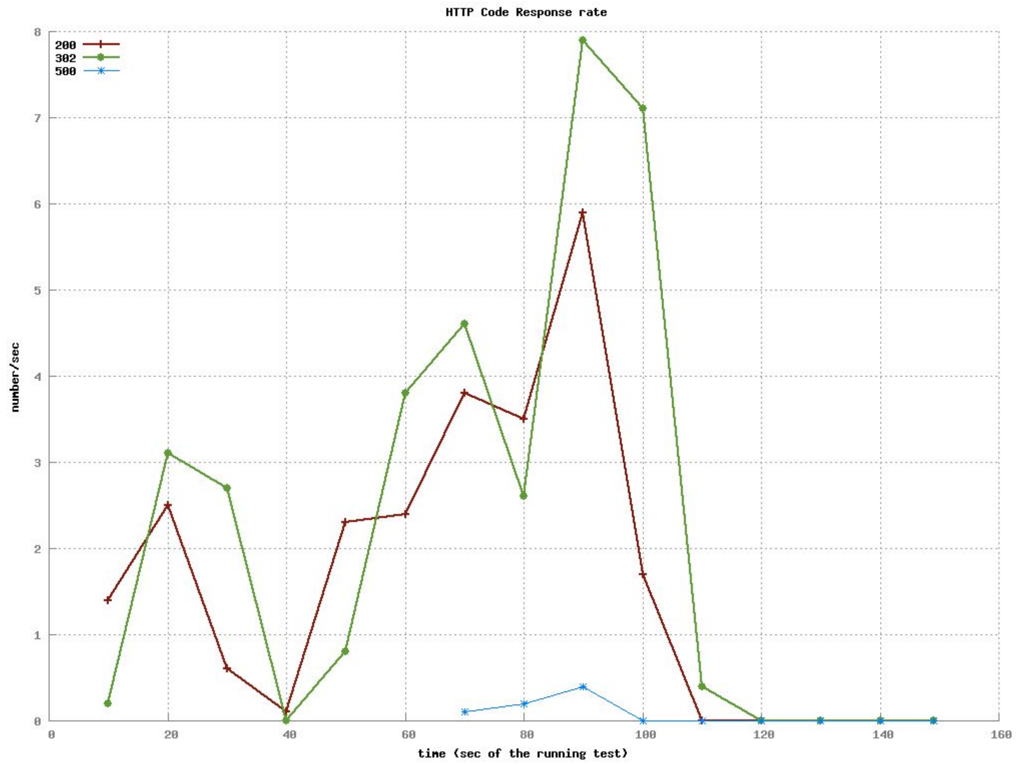
- Make sure you are getting back good status codes.
  - 200s and 300s are good
  - 400s and 500s are usually bad

Tsung - Output



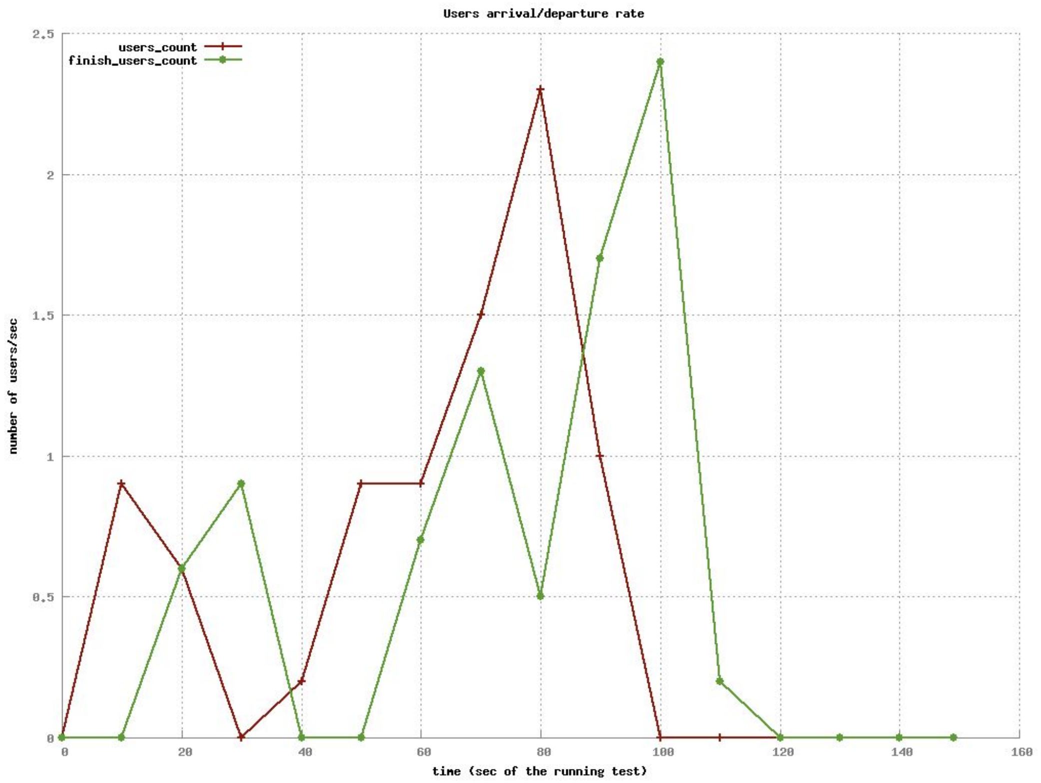
- Response times

Tsung - Output



- Error rates

Tsung - Output



- Are our synthetic users able to finish their tasks?

Tsung

Always remember that your Tsung instances will go down automatically (8 hours), so please scp off any important data or results immediately.

For Next Time...

Attempt to create a simple load testing script for your current app

Keep completing stories!