**Design Document**

Table of Contents

[Revision History 3](#_Toc41325886)

[Description: 4](#_Toc41325887)

[Environment Setup: 4](#_Toc41325888)

[Directory Structure 4](#_Toc41325889)

[Configuration File 5](#_Toc41325890)

[1) Configuration file name:”validated\_ledger.conf” 5](#_Toc41325891)

[2) Configuration file name:”plot.dat” 6](#_Toc41325892)

[How Does Script Work: 6](#_Toc41325893)

[Calculate\_polling.sh: 6](#_Toc41325894)

[Validated\_polling.sh: 8](#_Toc41325895)

[Graph Interpretation: 11](#_Toc41325896)

[Tests 12](#_Toc41325897)

[Question Answers 13](#_Toc41325898)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Comments** |
| Draft 1 | 25/05/2020 | Deepak Kukreja | Design Document Version 0.1 |

**Validated Ledger Bash script**

# Description:

The shell script periodically calls ripple server info command and record the sequence number of the latest validated ledger along with the current time. This data is then recorded in a file and then used to construct a plot time on X axis, and sequence number on y axis and age (time taken by server to validate ledger) of the each iteration on x2 axis that visualizes how frequently the ledger sequences incremented over time.

# Environment Setup:

Below is the list of software which I have installed on my PC

1) Ubuntu 16.04

2) BASH Shell

3) CURL version 7.47.0

4) JQ version 1.5.1

5) Gnuplot 5.0

6) Bats 0.4.0 (Bash Automated Test System)

7) git 2.7.4

# Directory Structure

root

etc

Contains configuraiotn fule

scripts

output

log

test

doc

# Configuration File

Project uses 2 configuration file “validated\_ledger.conf” and”plot.dat”

## 1) Configuration file name:”validated\_ledger.conf”

Configurations file parameter:

|  |  |  |
| --- | --- | --- |
| Parameter Name | Default Value | Description |
| CONFIG\_SERVER\_URL | http://s1.ripple.com:51234/ | URL of the Server |
| CONFIG\_SERVER\_COMMAND | server\_info | Server\_info command |
| CONFIG\_MAX\_REDIRECT | 1 | Used by Curl to redirect in case of proxy. Count is number of time curl will try to redirect. |
| CONFIG\_SERVER\_DOWN\_TIME\_PERIOD | 10 | No of times the script will check if server available. |
| CONFIG\_SERVER\_POLLING\_COUNT | 20 | No of times server info will be called to get validated  Timespan = CONFIG\_SERVER\_POLLING\_COUNT \* Polling interval |
| CONFIG\_SERVER\_SAMPLE\_COUNT | 10 | Number of unique validated ledger sequences required to calculate polling interval |
| CONFIG\_SERVER\_POLLING\_INTERVAL | 1 | Polling interval calculated by algorithm to fetch server data after given time |

## 2) Configuration file name:”plot.dat”

This file is input to gnuplot which contains graph settings.



# How Does Script Work:

There are 2 Shell Script one is **“validated\_ledger.sh”** and second one is **“calculate\_polling.sh”**. “validated\_ledger.sh” is the parent script which calls “calculate\_polling.sh” to get a polling interval during runtime.

## Calculate\_polling.sh:

This shell script calculates polling interval during runtime by calculating the average time taken by the server to generate unique validated ledger sequences. The count of **unique validated sequence** that needs to be captured for polling interval calculation is stored in config file parameter: **CONFIG\_SERVER\_SAMPLE\_COUNT.**

The script uses curl command to make an **“HTTP GET”** request to the ripple server and then read the HTTP response. When **HTTP OK (200)** is received, HTTP response is then passed on to JQ to parse and write the **“validated ledger seq”** and **“timestamp”** in comma separated format in **“polling.csv”** file.

{CODE :}

# HTTP get request

\_http\_response=$(curl -v -s -L --write-out "HTTPSTATUS:%{http\_code}" \

--max-redirs $CFG\_MAX\_REDIRECT --max-time 1\

-H "content-type: application/json" \

-X GET -d "{\"method\":\"$CFG\_COMMAND\"}" $CFG\_URL 2>/dev/null)

# Retreive http body and response code

\_http\_body=$(echo $\_http\_response | sed -e 's/HTTPSTATUS\:.\*//g')

\_http\_status=$(echo $\_http\_response | tr -d '\n' | sed -e 's/.\*HTTPSTATUS://')

if [[ `echo $\_http\_body | jq "(.result.info.validated\_ledger.seq)"` != "null" ]];then

\_output=$(echo $\_http\_body | \

jq -r "[.result.info.validated\_ledger.seq, .result.info.time] | @csv" | \

awk -F"[ \"]" '{print $1 $3}');

echo $\_output >> $POLLDATA\_FILE

{CODE}

Currently the script is collecting 10 (CONFIG\_SERVER\_SAMPLE\_COUNT, default value) unique sequences as sample and calculates the average time server took to validate 10 ledgers.

Calculated average time (polling interval) is then updated in config file parameter: CONFIG\_SERVER\_POLLING\_INTERVAL

**Average time** = (timestamp of the 10th validated ledger sequence – timestamp of first validated ledger sequence)/ CONFIG\_SERVER\_SAMPLE\_COUNT

The script takes following argument

**Arguments:**

1) Server URL

2) Server command

3) Redirect count

4) Sample count

**Exception Handling**

The following exception are handled by the script

1) HTTP Response Code is Not 200

2) Server info command Status is Not Success

3) Checking if the validated ledger sequence is non null i.e. could be a closed ledger

**Error Exit:**

The script will try to connect to server for number of time mentioned in config parameter CONFIG\_SERVER\_DOWN\_TIME\_PERIOD. If http response is Not 200 then script will exit with error.

The script will try to connect to server for number of time mentioned in config parameter CONFIG\_SERVER\_DOWN\_TIME\_PERIOD. If command status is not success then script will exit with error.

## Validated\_polling.sh:

This shell script periodically calls server info command and records the sequence number of the latest validated ledger, current time, age and hostid. The sequence number, timestamp and age is then used to construct a plot graph where time on X axis and sequence number on y axis and age (time taken by server to validate ledger) of the each iteration on x2 axis.

**Initialization steps in the script:**

1) Create config directory if dir doesn't exist

2) Create output directory if dir doesn't exist

3) Create log directory if dir doesn't exist

4) Create config file with default values if file doesn't exist

5) Archiving previous plot file by adding date timestamp

6) Archiving previous log file by adding date timestamp

7) Archiving previous output file by adding date timestamp

The script then calls to “calculate\_polling.sh” to get the polling interval which is used as a sleep for every HTTP GET request call.

Script use curl command to make an **“HTTP GET”** request to the ripple server and read the HTTP response. When **HTTP OK (200)** is received, HTTP response is then passed on to JQ to parse and write the **“validated ledger seq”**, **“timestamp”, ”age” and “hostid”** in comma separated format in **“plot.csv”** file.

The script further checks if validated ledger sequence is null then it assumes it is a closed ledger and increment the closed ledger count and writes it to the output file. In case of non null validated ledger it uses JQ to write output in plot.csv file.

{code:}

# Checking if the ledger is not a closed ledger

if [[ `echo ${\_http\_body} | jq "(.result.info.validated\_ledger.seq)"` != "null" ]]; then

# write timestamp and validated ledger seq in plot file

# Not a closed ledger

\_output=$(echo ${\_http\_body} | jq -r "[.result.info.validated\_ledger.seq, .result.info.time] | @csv" | awk -F"[ \"]" '{print $1 $3}');

echo $\_output >> $PLOT\_FILE

{code}

The third column for plot.csv file is age which is derived by taking time difference between “end time” and “start time” between two unique validated ledger sequences and writes the age in plot.csv file.

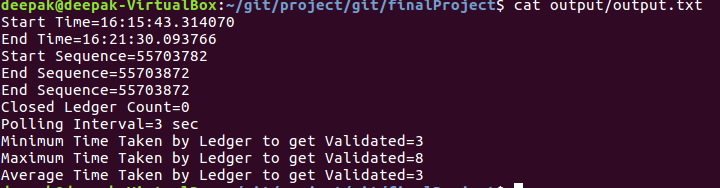
{code :}

age=`(echo "$(( $(date -d "$newtime" '+%s') - $(date -d "$prevtime" '+%s') ))")`

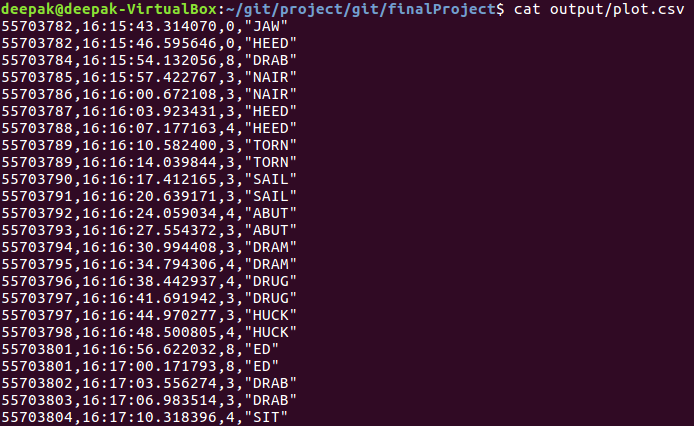
code}

The script also calculates minimum, maximum, average time taken by server to validate a ledger and writes to output.txt file.

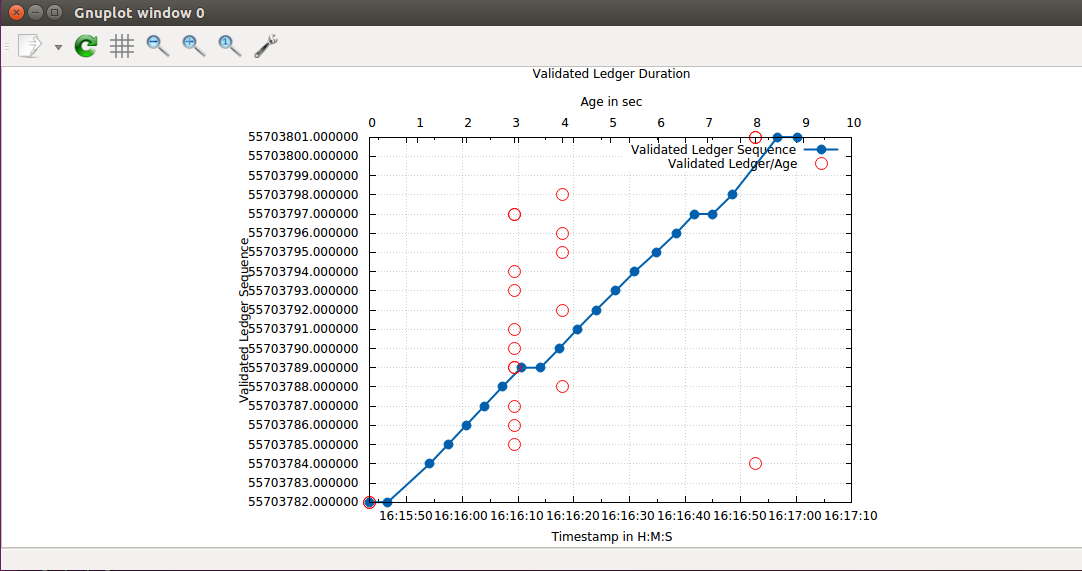
**OUTPUT.TXT FILE**



**PLOT.CSV FILE**

****

Once the script reaches maximum time span (CONFIG\_SERVER\_POLLING\_COUNT \* Polling interval), it calls gnuplot to plot the graph.



Limitation: The graph only plots the first 20 rows of the plot.csv file as mentioned in plot.dat file. This is because x tics and y tics for the graph are hard coded currently in the file and if the graph is run for more record the labels at the bottom overlap. This problem can be resolved in future.

{code:}

plot "<(sed -n '1,21p' output/plot.csv)" using 2:1 axes x1y1 title "Validated Ledger Sequence" with linespoints linestyle 1, \  
     "" using 3:1 axes x2y1 title "Validated Ledger/Age" with p ps 2 pointtype 6 lc rgb "red"

{code}

**Arguments:**

NONE

**Exception Handling**

The following exception are handled by the script

1) HTTP Response Code is Not 200

2) Server info command Status is Not Success

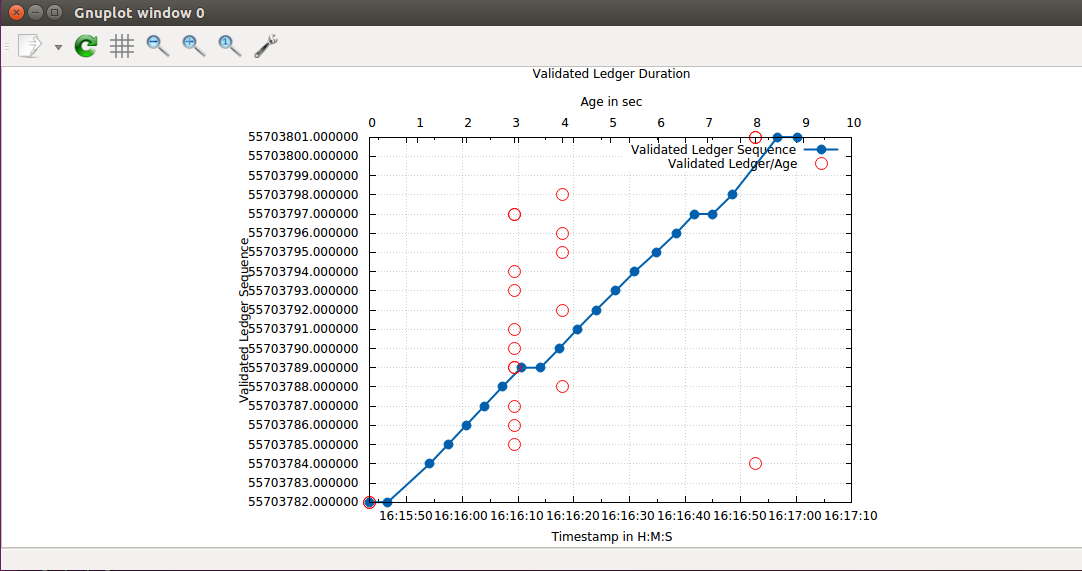
**Error Exit:**

The script will try to connect to server for number of time mentioned in config parameter CONFIG\_SERVER\_DOWN\_TIME\_PERIOD. If http response is Not 200 then script will exit with error.

The script will try to connect to server for number of time mentioned in config parameter CONFIG\_SERVER\_DOWN\_TIME\_PERIOD. If command status is not success then script will exit with error.

# Graph Result Interpretation:

* The graph shows Timestamp on X axis, Validated Ledger Sequence on Y Axis and Age on X2 axis.
* Blue line in the graph shows how frequently the ledger sequence is incremented over time. As can be seen in below graph a new validated ledger is coming after every 3 sec which is the polling interval derived by the initial algorithm.
* The red circle depicts the time taken by the server to validate sequence on y axis.
* As can be seen from red circle that the minimum time taken by server to validate ledger is 3 sec and maximum time taken is 8 sec.
* Below graph also shows that there are some missing sequences like 55703783, 55703799 and 557037800. This could be due to closed ledger received during that time span or these missing ledger sequences would have been validated in time which is much less than polling interval calculated by our script.

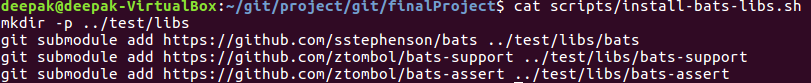


# Tests

You can execute automated test suite using BATS. Bats is a [TAP](http://testanything.org/)-compliant testing framework for Bash. It provides a simple way to verify that the LINUX programs you write behave as expected.

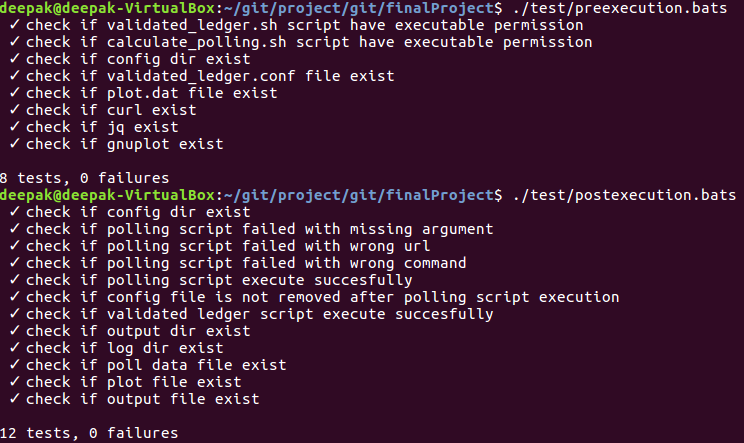
A Bats test file is a Bash script with special syntax for defining test cases. Under the hood, each test case is just a function with a description.

Installation steps are:



There are 2 bats file written one is pre execution test suite and second is post execution test suite.

Should output something like:



# Question Answers

How does your script work?

How did you decide on your polling interval?

What do the results tell you?

What might explain the variation in time between new ledgers?

Bonus question #1: Enhance your script to calculate the min, max, and average time that it took for a new ledger to be validated during the span of time captured.

Bonus question #2: There are some other (better) ways that you could use the rippled API to find how long each ledger took to close/validate. Using the API documentation, find and describe one of these methods (you don’t need to actually implement it).