DEPLOYMENT AND PERFORMANCE TESTING

NodeJS for Enterprise : Day 2/2

DAY 2 - OUTLINE

- Deployment
  - NodeJS as Service
  - Nginx as Reverse Proxy
  - Nginx Upstreams
- Basic Performance Testing with JMeter
- Performance Monitoring Tools with NewRelic and Keymetrics
- Real-time Application Analytics Sample Case (Hummingbird)
- Security Discussion
- Q&A

DEPLOYMENT

SIMPLE DEPLOYMENT

Start small with built-in static server, node.js app & database all on the same server

Sample of Tools:

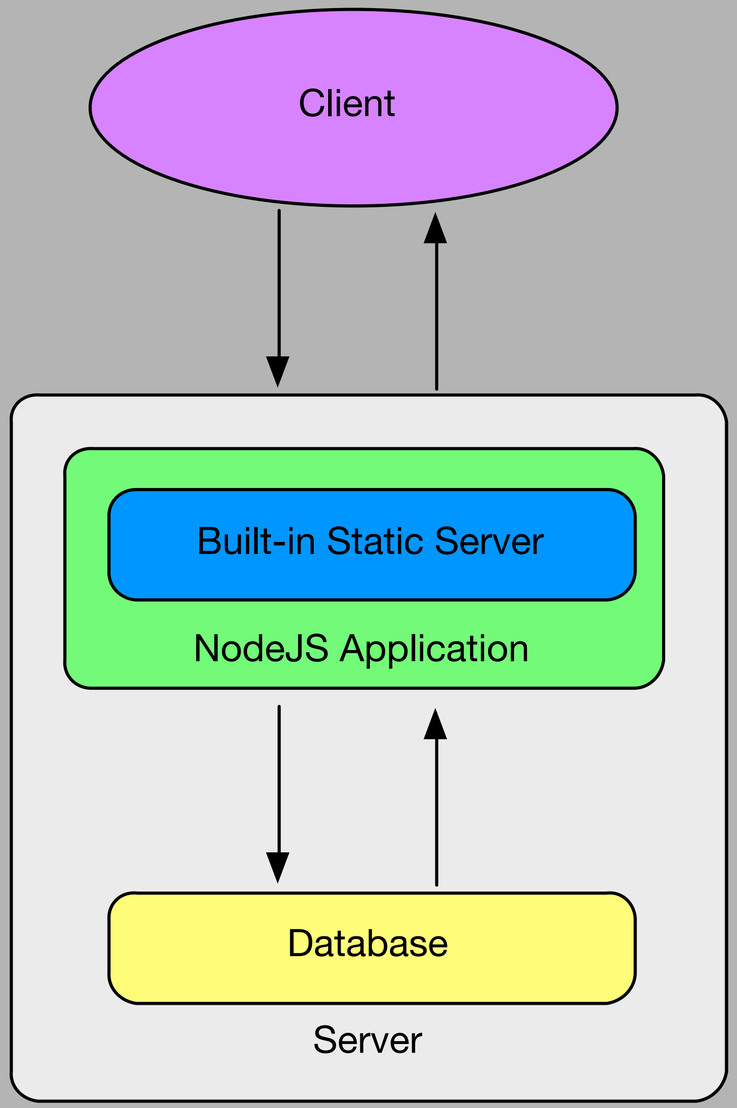
built-in static server

OR

node-windows

OR

pm2



LAB 2-1

node server.js

BUILT-IN STATIC SERVER

```
$ node server.js
```

LAB 2-2

node-windows, node-mac, node-linux

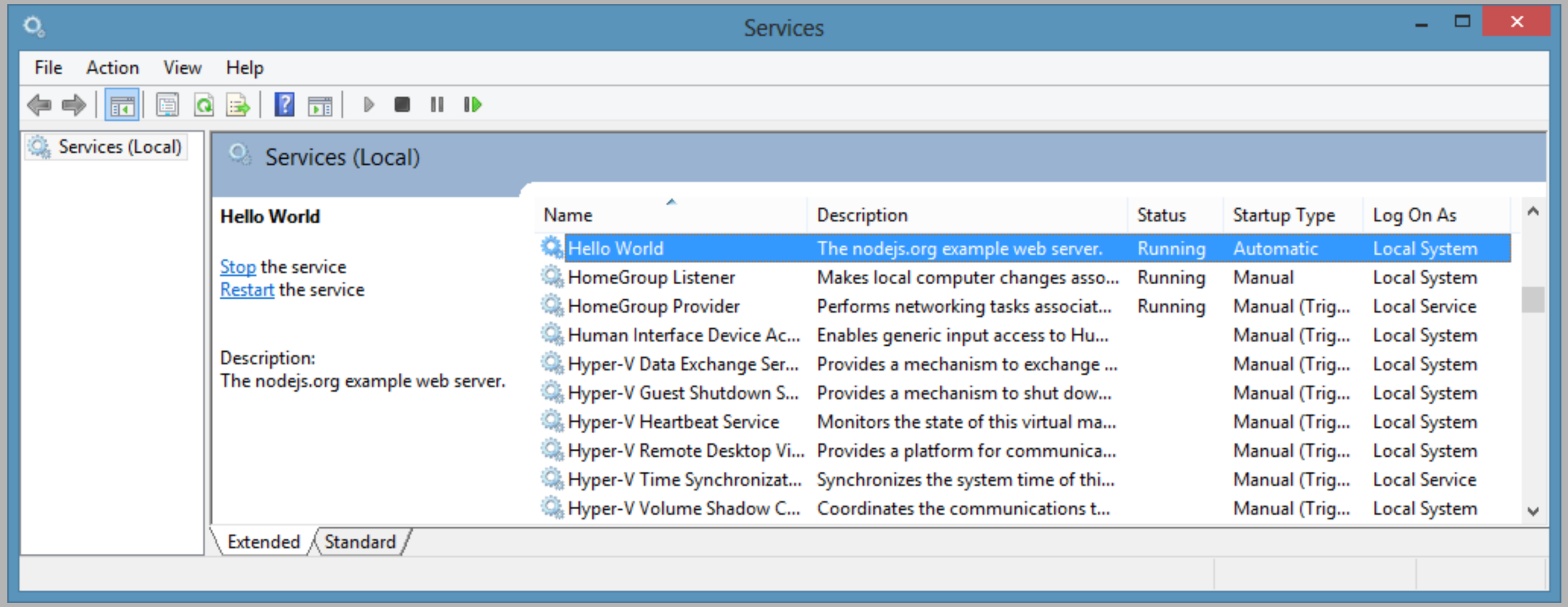
INSTALLATION

```
#node-windows, node-mac, node-linux

$ npm install -g node-windows

in your project root

$ npm link node-windows
```

INSTALL SERVICE

```
var Service = require('node-windows').Service;

// Create a new service object
var svc = new Service({
  name:'Hello World',
  description: 'The nodejs.org example web server.',
  script: 'C:\\path\\to\\helloworld.js'
});

// Listen for the "install" event, which indicates the
// process is available as a service.
svc.on('install',function(){
  svc.start();
});
```

INSTALL SERVICE



NODE-WINDOWS EVENTS

- install - Fired when the script is installed as a service.
- alreadyinstalled - Fired if the script is already known to be a service.
- invalidinstallation - Fired if an installation is detected but missing required files.
- uninstall - Fired when an uninstallation is complete.
- start - Fired when the new service is started.
- stop - Fired when the service is stopped.
- error - Fired in some instances when an error occurs.

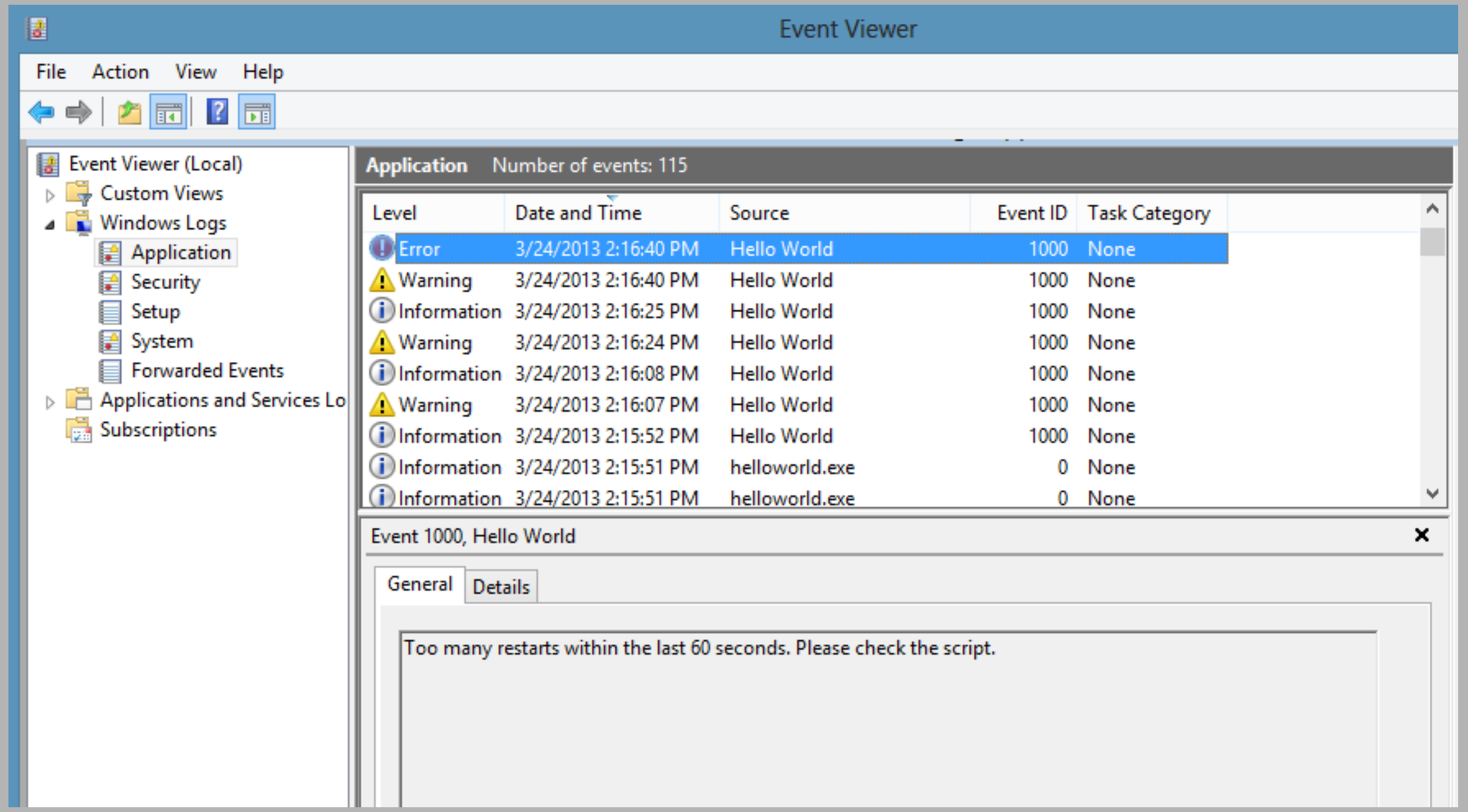
ENVIRONMENT VARIABLE

```
var svc = new Service({
  name:'Hello World',
  description: 'The nodejs.org example web server.',
  script: 'C:\\path\\to\\helloworld.js',
  env: [{
    name: "HOME",
    value: process.env["USERPROFILE"]
  },
  {
    name: "TEMP",
    value: path.join(process.env["USERPROFILE"],"/temp")
  }]
});

svc.install();
```

LOGGING

```
var EventLogger = require('node-windows').EventLogger;

var log = new EventLogger('Hello World');

log.info('Basic information.');
log.warn('Watch out!');
log.error('Something went wrong.');
```

LOGGING


Event Viewer
File Action View Help
Event Viewer (Local)
Custom Views
Windows Logs
Application
Security
Setup
System
Forwarded Events
Applications and Services Lo
Subscriptions
Application Number of events: 115
Level Date and Time Source Event ID Task Category
Error 3/24/2013 2:16:40 PM Hello World 1000 None
Warning 3/24/2013 2:16:40 PM Hello World 1000 None
Information 3/24/2013 2:16:25 PM Hello World 1000 None
Warning 3/24/2013 2:16:24 PM Hello World 1000 None
Information 3/24/2013 2:16:08 PM Hello World 1000 None
Warning 3/24/2013 2:16:07 PM Hello World 1000 None
Information 3/24/2013 2:15:52 PM Hello World 1000 None
Information 3/24/2013 2:15:51 PM helloworld.exe 0 None
Information 3/24/2013 2:15:51 PM helloworld.exe 0 None
Event 1000, Hello World
General Details
Too many restarts within the last 60 seconds. Please check the script.

UNINSTALL SERVICE

```
var Service = require('node-windows').Service;

// Create a new service object
var svc = new Service({
  name:'Hello World',
  script: require('path').join(__dirname,'helloworld.js')
});

// Listen for the "uninstall" event so we know when it's done.
svc.on('uninstall',function(){
  console.log('Uninstall complete.');
  console.log('The service exists: ',svc.exists);
});

// Uninstall the service.
svc.uninstall();
```

LAB 2-3

pm2

http://pm2.keymetrics.io/

INSTALLATION

```
$ npm install pm2 -g

OR

$ npm install pm2@latest -g
```

START NODE.JS APP PROCESS

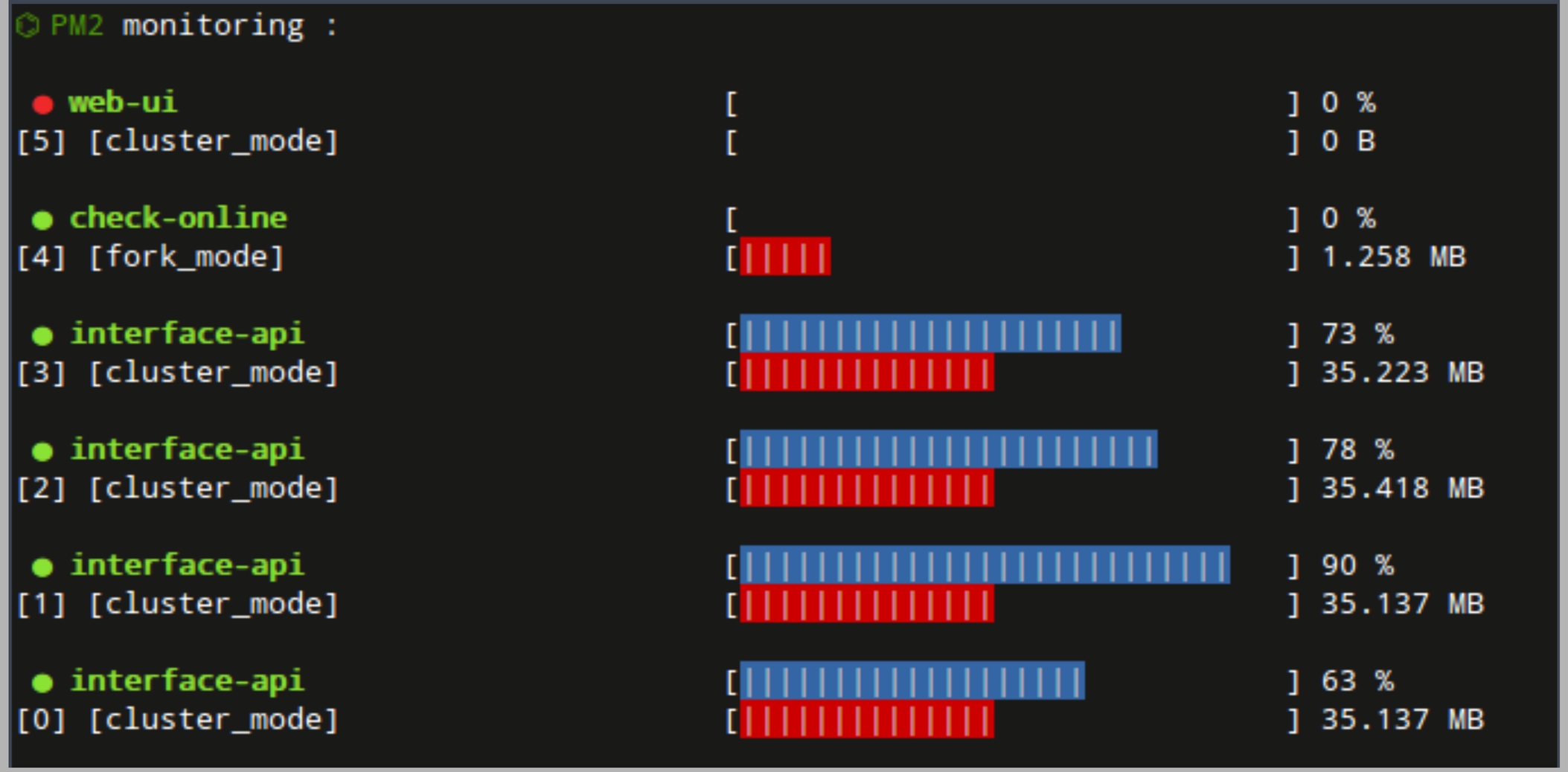
```
$ pm2 start app.js --name “app-name”

$ pm2 start app.js -i 0

$ pm2 start app.js -i max
```

LIST NODE.JS APP PROCESS

```
$ pm2 list

$ pm2 jlist

$ pm2 prettylist

$ pm2 describe 0

$ pm2 updatePM2          # Update in memory pm2

$ pm2 ping               # Ensure pm2 daemon has been launched
```

MONITORING NODE.JS

$ pm2 monit



NODE.JS APP LOGS

```
$ pm2 logs [—raw]

$ pm2 flush

$ pm2 reloadLogs
```

STOP/RESTART NODE.JS APP PROCESS

```
$ pm2 stop all          # Stop all processes
$ pm2 restart all       # Restart all processes

$ pm2 reload all        # Will 0s downtime reload (for NETWORKED apps)
$ pm2 gracefulReload all # Send exit message then reload (for networked apps)

$ pm2 stop 0            # Stop specific process id
$ pm2 restart 0         # Restart specific process id
```

DELETE NODE.JS APP PROCESS

```
$ pm2 delete 0          # Will remove process from pm2 list
$ pm2 delete all        # Will remove all processes from pm2 list
```

AUTOSTART NODE.JS APP

สำหรับ Windows Platform
$ npm install pm2-windows-startup -g

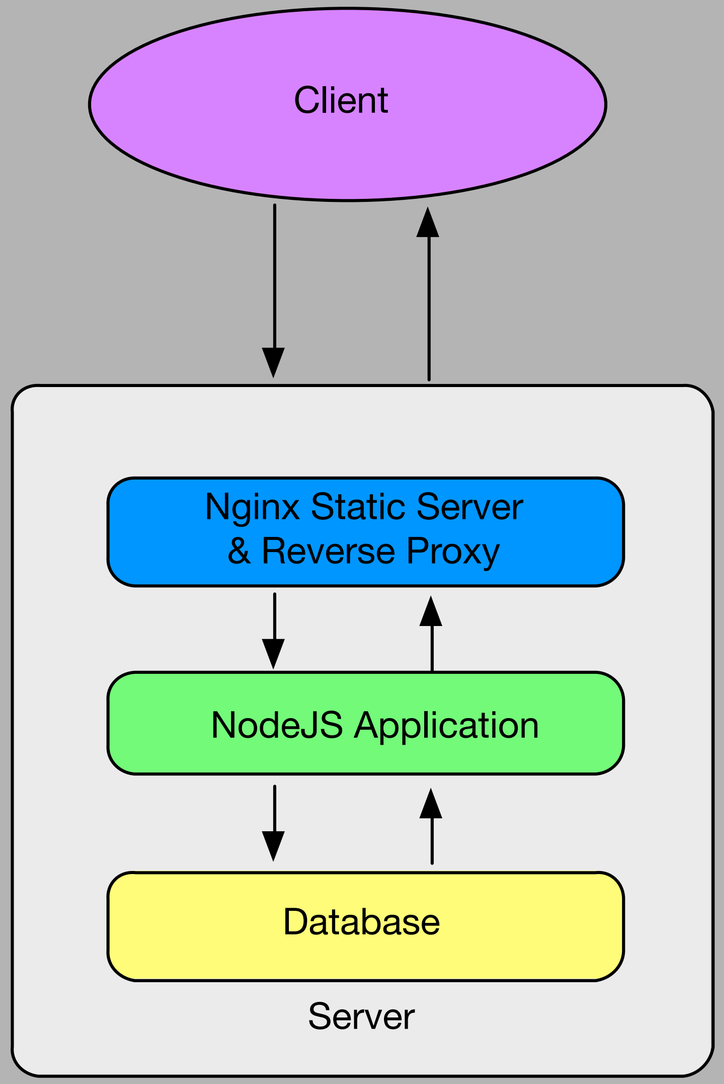
$ pm2-startup install

สำหรับ Unix/Linux Platform
$ pm2 startup

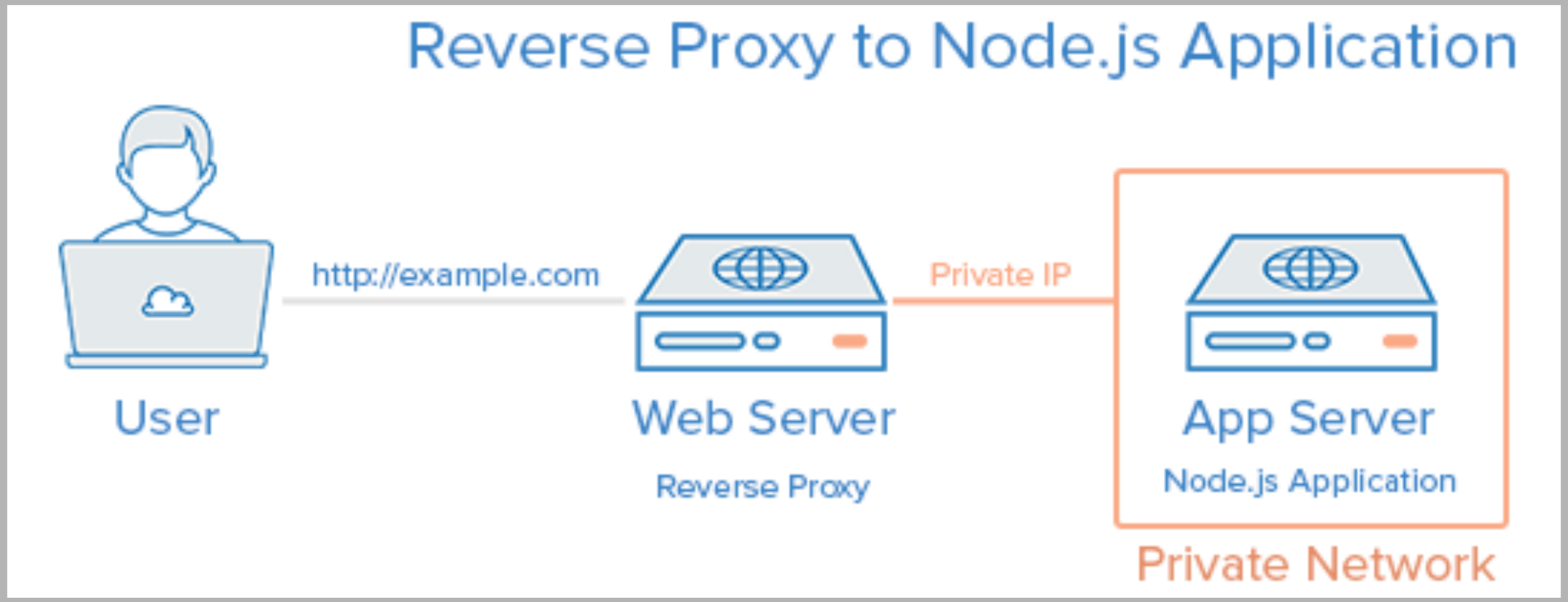
Process Autostart จะทำงานหลังจากสั่งบันทึก Process list
$ pm2 save

SCALING NODEJS

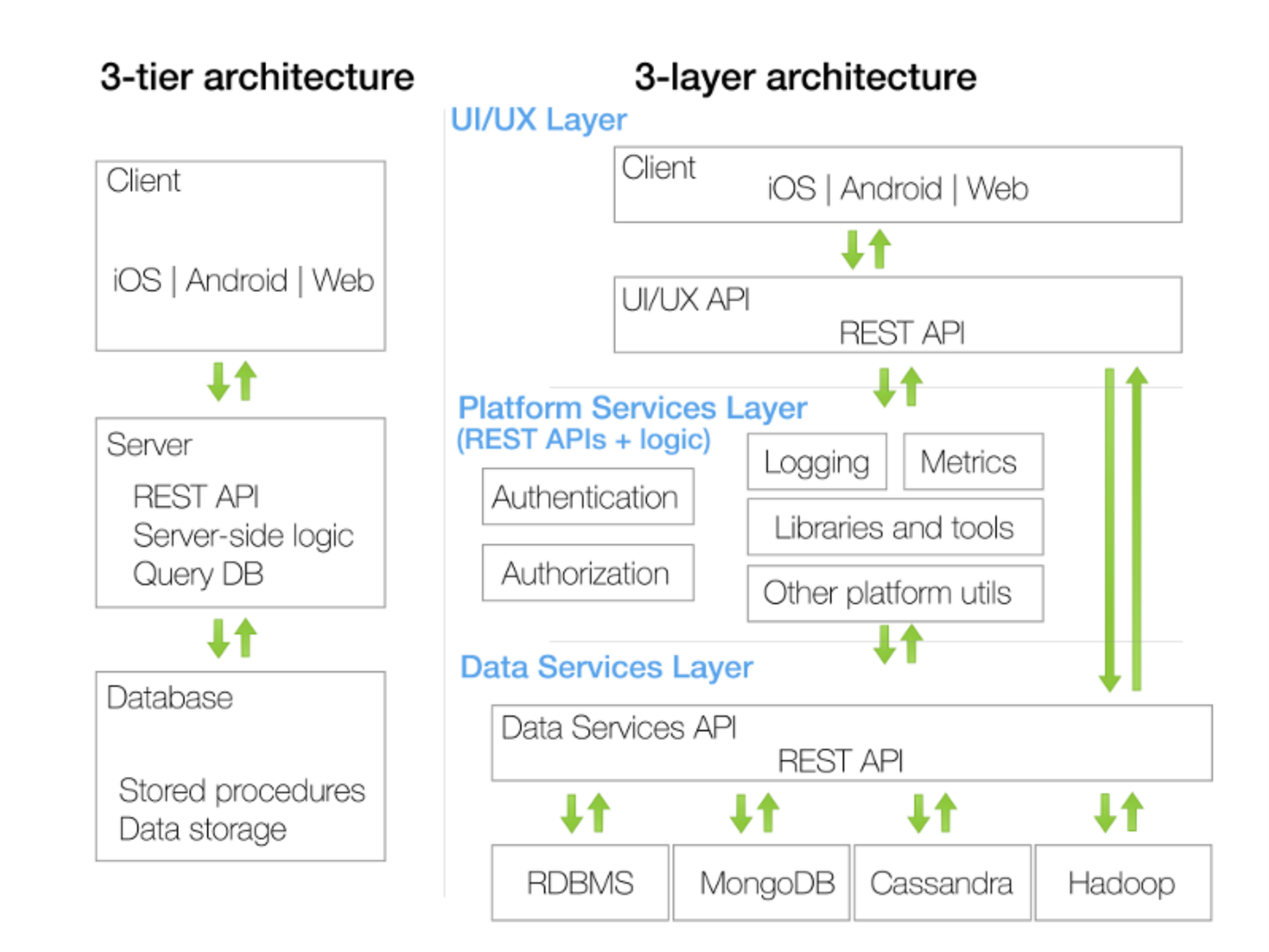
Use nginx as as proxy server as well to load balance requests.
The number of node.js app instance depends on how many CPU cores on the machine.



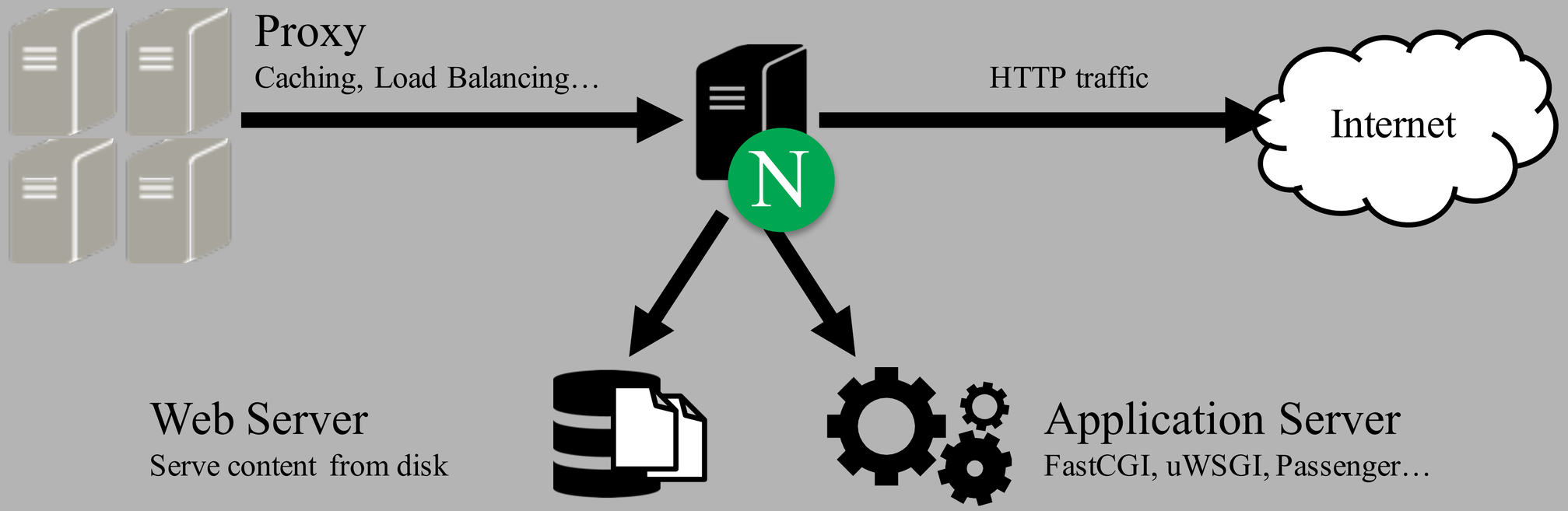
NODE.JS BEST PRACTICE
REVERSE PROXY



3-LAYER ARCHITECTURE



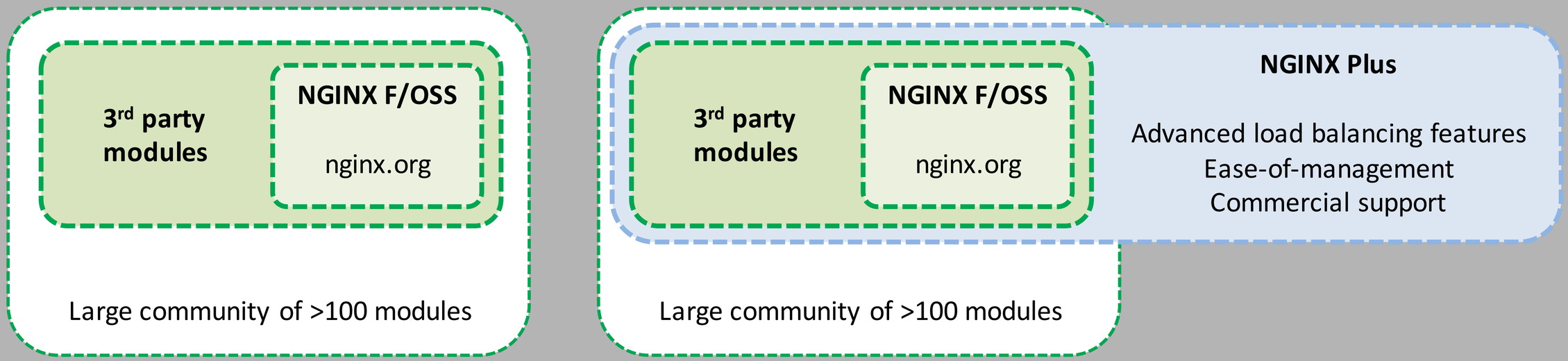
WHAT IS NGINX



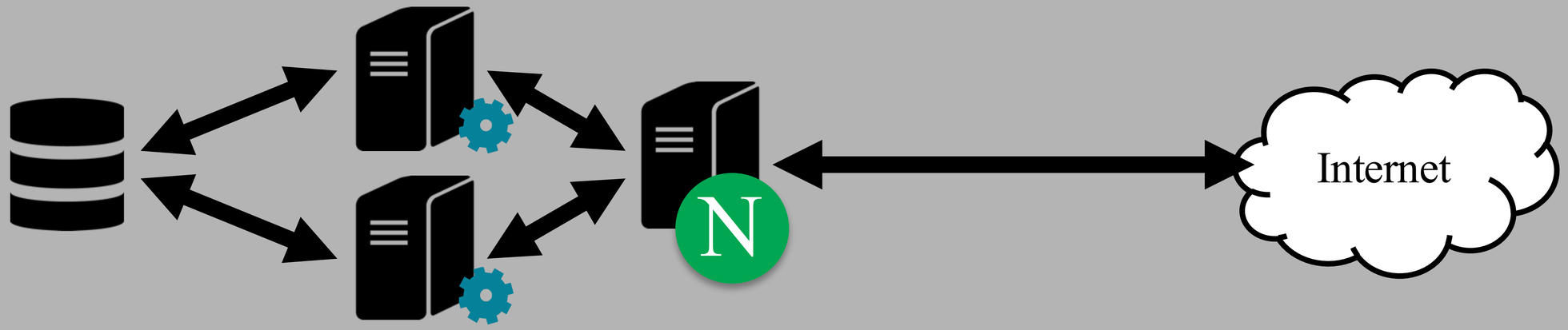
Advanced Features:

- ☑ Application Acceleration
- ☑ SSL and SPDY termination
- ☑ Performance Monitoring
- ☑ High Availability
- ☑ Bandwidth Management
- ☑ Content-based Routing
- ☑ Request Manipulation
- ☑ Response Rewriting
- ☑ Authentication
- ☑ Video Delivery
- ☑ Mail Proxy
- ☑ GeoLocation

NGINX AND NGINX PLUS



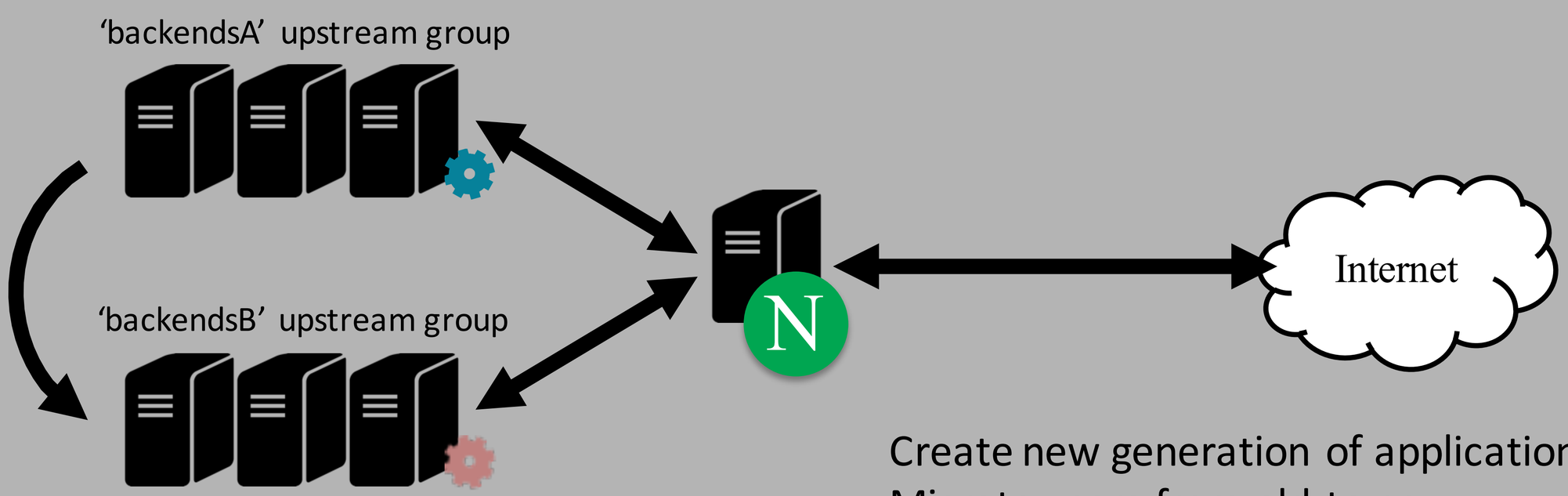
LOAD BALANCING



Why?
- ☑ Improved Application Availability
- ☑ Management
- ☑ Increased Capacity
- ☑ Advanced techniques e.g. A|B testing

How?
- ☑ DNS Round Robin
- ☑ Hardware L4 load balancer
- ☑ Software Reverse Proxy LB
- ☑ Cloud solution

LOAD BALANCING



Create new generation of application
Migrate users from old to new
Preserve sessions, no interruptions

LAB 2-4

nginx reverse proxy

ROOT CONTEXT

Unix/Linux
/etc/nginx/sites-available/default

```
server {
    listen 80;

    server_name example.com;

    location / {
        proxy_pass http://APP_PRIVATE_IP_ADDRESS:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

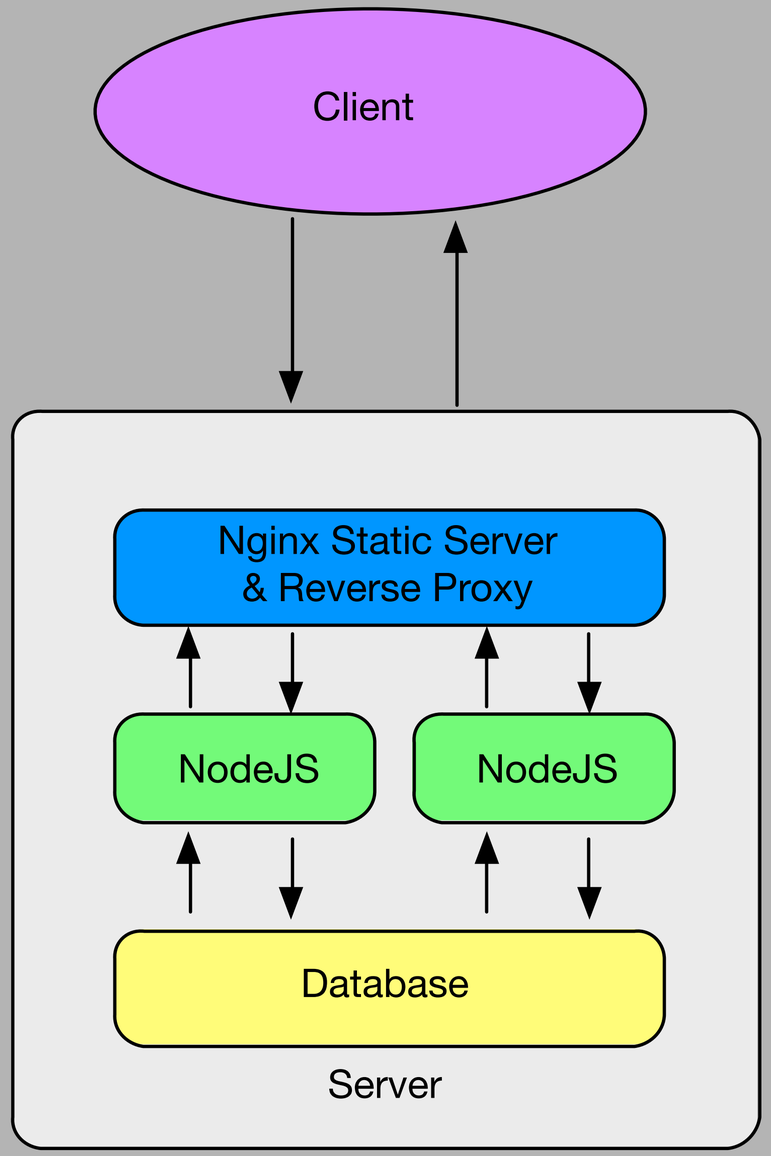
CUSTOM CONTEXT

Unix/Linux
/etc/nginx/sites-available/default

```
server {
    listen 80;

    server_name example.com;

    location /app2 {
        proxy_pass http://APP_PRIVATE_IP_ADDRESS:8081;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

SCALING NODEJS

Use nginx as as proxy server as well to load balance requests.
The number of node.js app instance depends on how many CPU cores on the machine.



LAB 2-5

nginx load balancer

UPSTREAM (ROUND ROBIN)

Unix/Linux
/etc/nginx/sites-available/default

```
upstream backend_hosts {
    server host1.example.com weight=3;
    server host2.example.com;
    server host3.example.com;
}

server {
    listen 80;
    server_name example.com;

    location /app {
        proxy_pass http://backend_hosts;
    }
}
```

UPSTREAM - LEAST_CONN

Unix/Linux
/etc/nginx/sites-available/default

```
upstream backend_hosts {
    least_conn;
    server host1.example.com;
    server host2.example.com;
    server host3.example.com;
}

server {
    listen 80;
    server_name example.com;

    location /app {
        proxy_pass http://backend_hosts;
    }
}
```

UPSTREAM - HASH

Unix/Linux
/etc/nginx/sites-available/default

```
upstream backend_hosts {
    hash $remote_addr$remote_port consistent;
    server host1.example.com;
    server host2.example.com;
    server host3.example.com;
}

server {
    listen 80;
    server_name example.com;

    location /app {
        proxy_pass http://backend_hosts;
    }
}
```

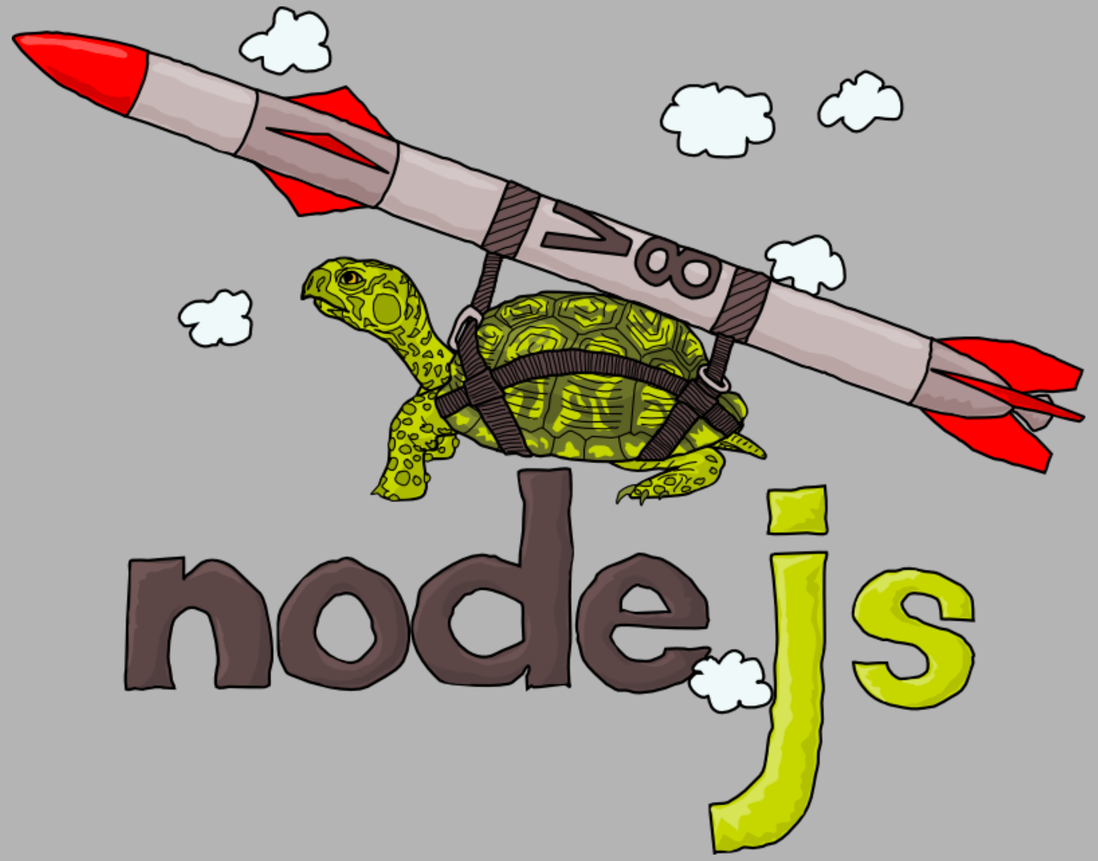
UPSTREAM - A/B TESTING

Unix/Linux
/etc/nginx/sites-available/default

```
upstream backend_hosts {
    hash $remote_addr$remote_port consistent;
    server host1.example.com;
    server host2.example.com;
    server host3.example.com;
}

map $cookie_group $group {
   ~(?P<value>.+)$ $value;
   default         backendB;  # The default upstream group
}

server {
   listen 80;

   location / {
      add_header Set-Cookie "group=$group; path=/"
      proxy_pass http://$group;
   }
}
```

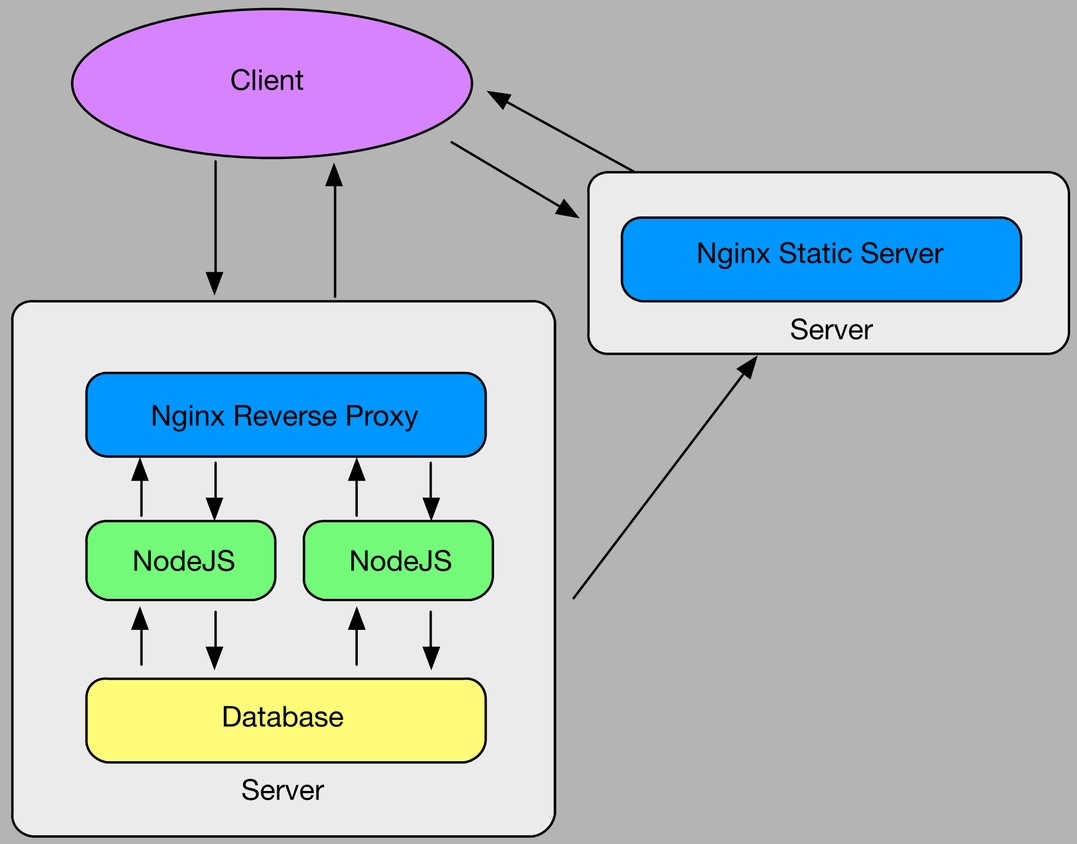
MORE ADVANCED SCENARIOS

for further experiment



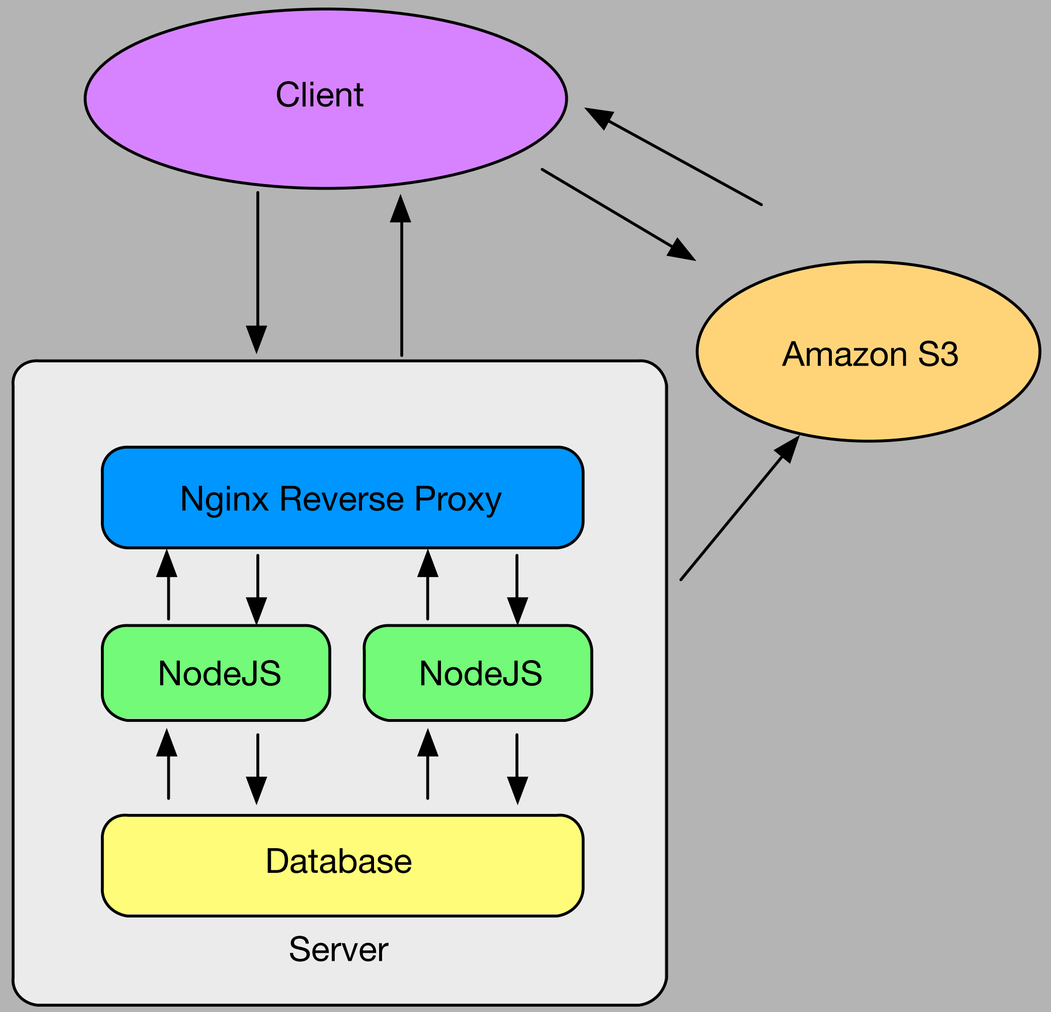
SCALING NODEJS

Use nginx as as proxy server as well to load balance requests.
The number of node.js app instance depends on how many CPU cores on the machine.



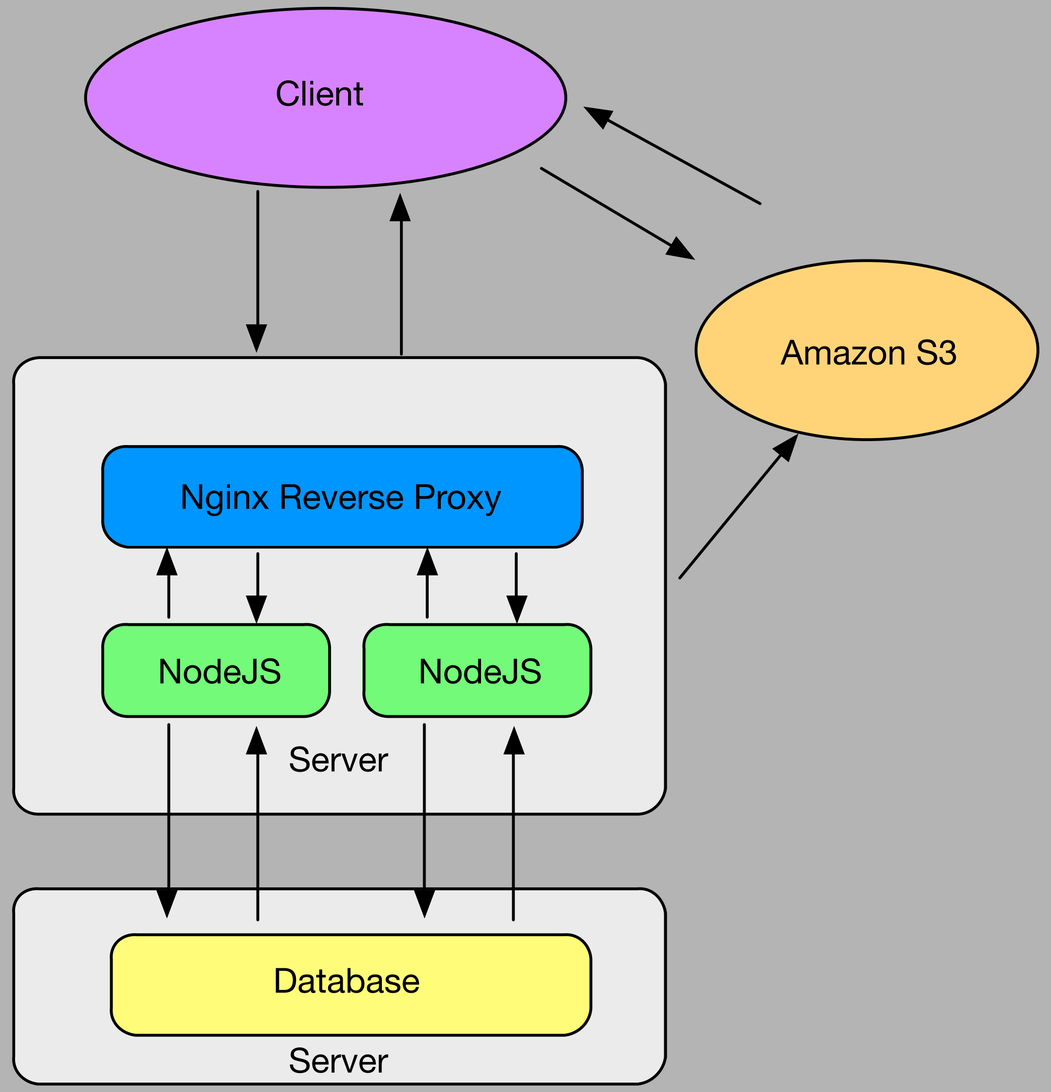
SCALING NODEJS

Use nginx as as proxy server as well to load balance requests.
The number of node.js app instance depends on how many CPU cores on the machine.



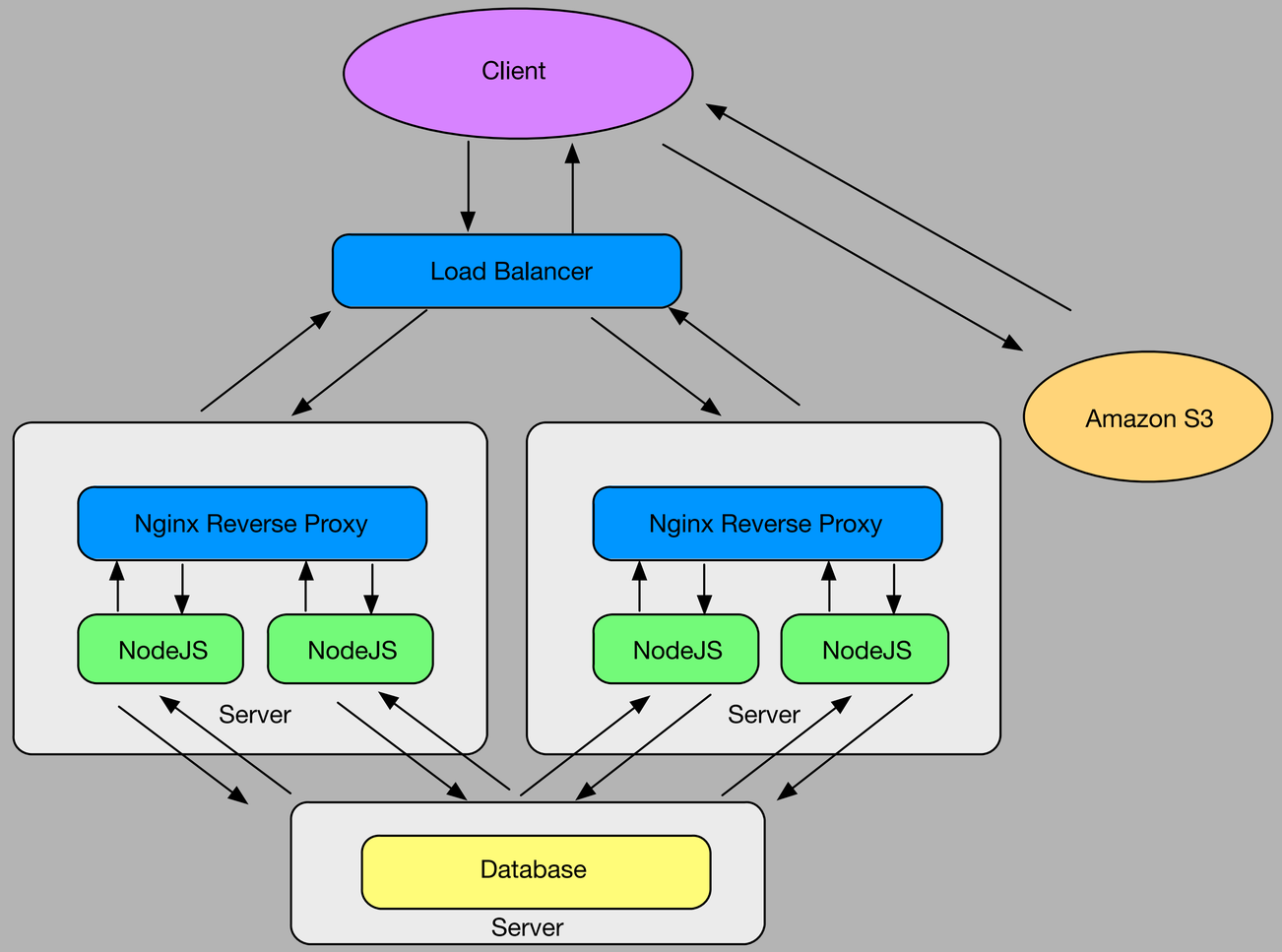
SCALING NODEJS

Use nginx as as proxy server as well to load balance requests.
The number of node.js app instance depends on how many CPU cores on the machine.

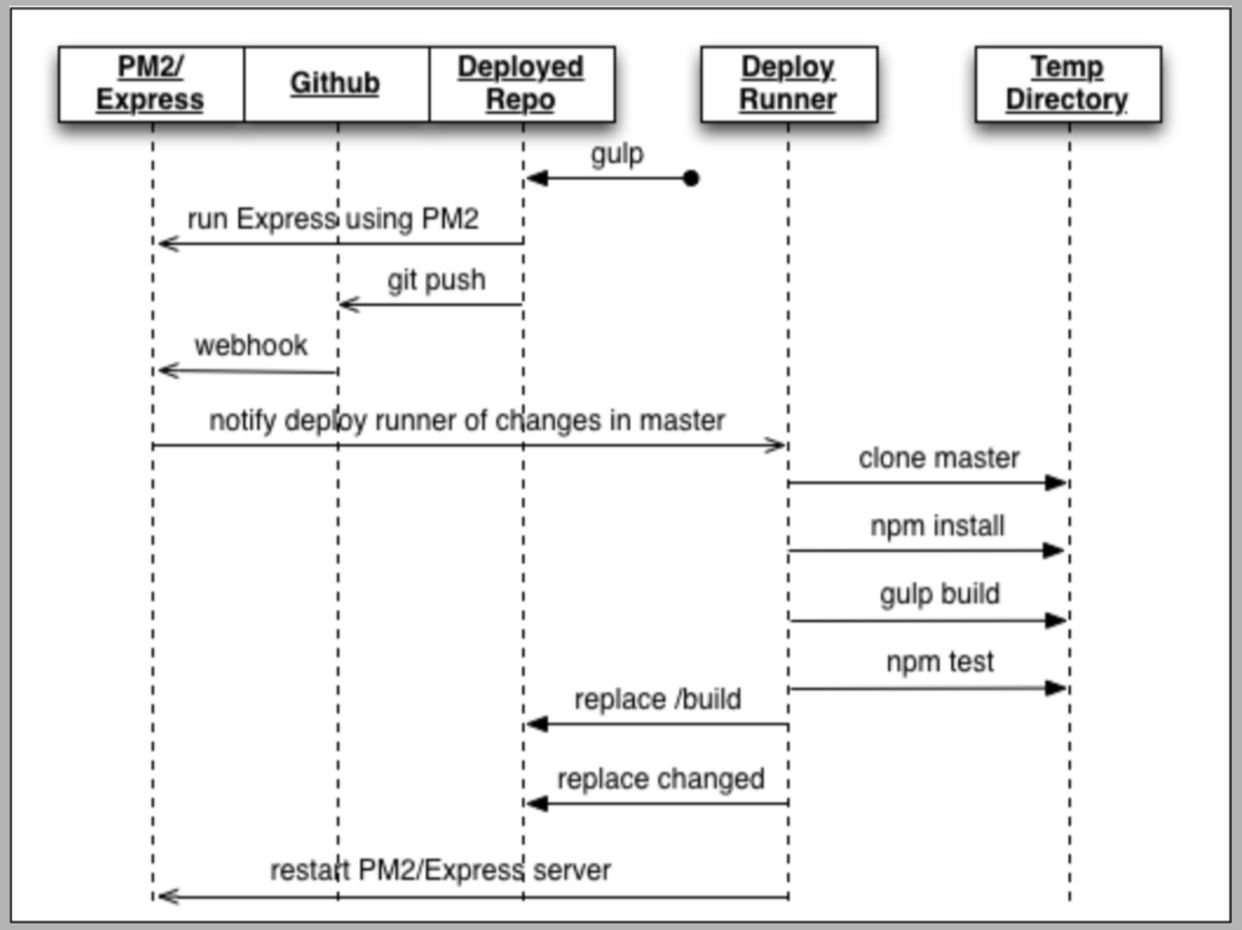


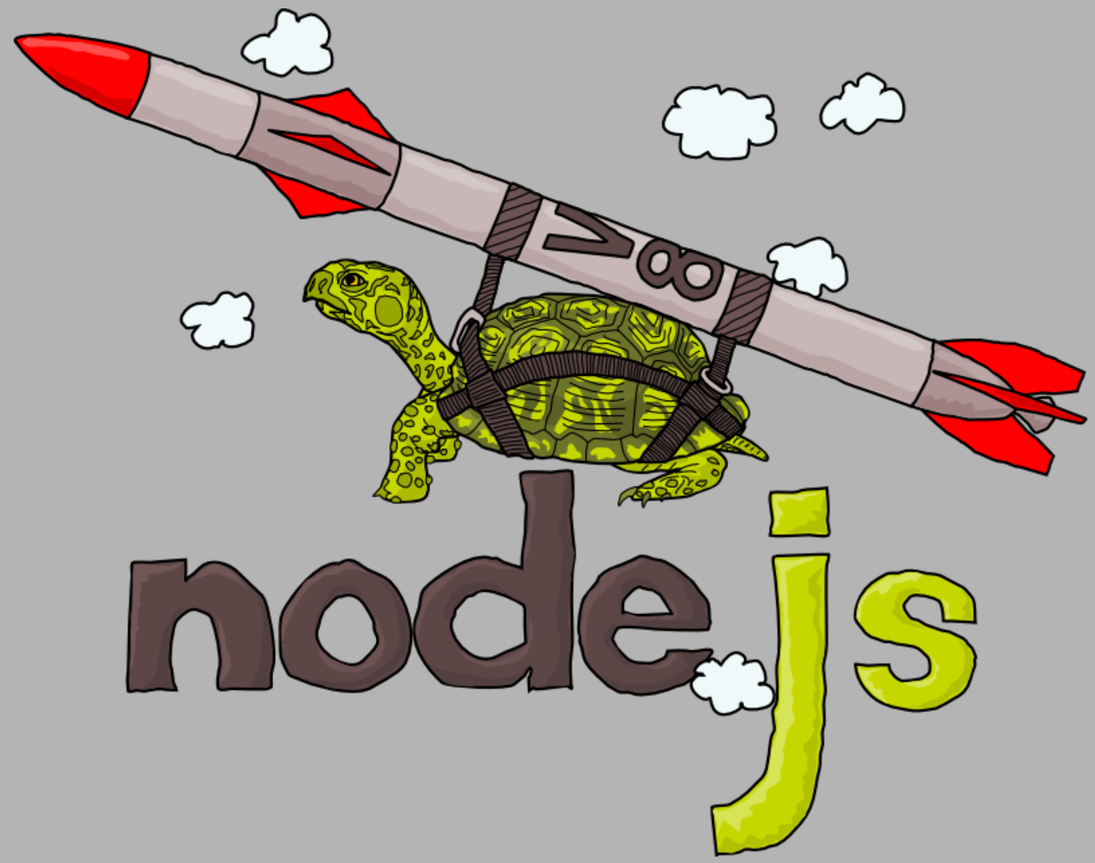
SCALING NODEJS

Use nginx as as proxy server as well to load balance requests.
The number of node.js app instance depends on how many CPU cores on the machine.

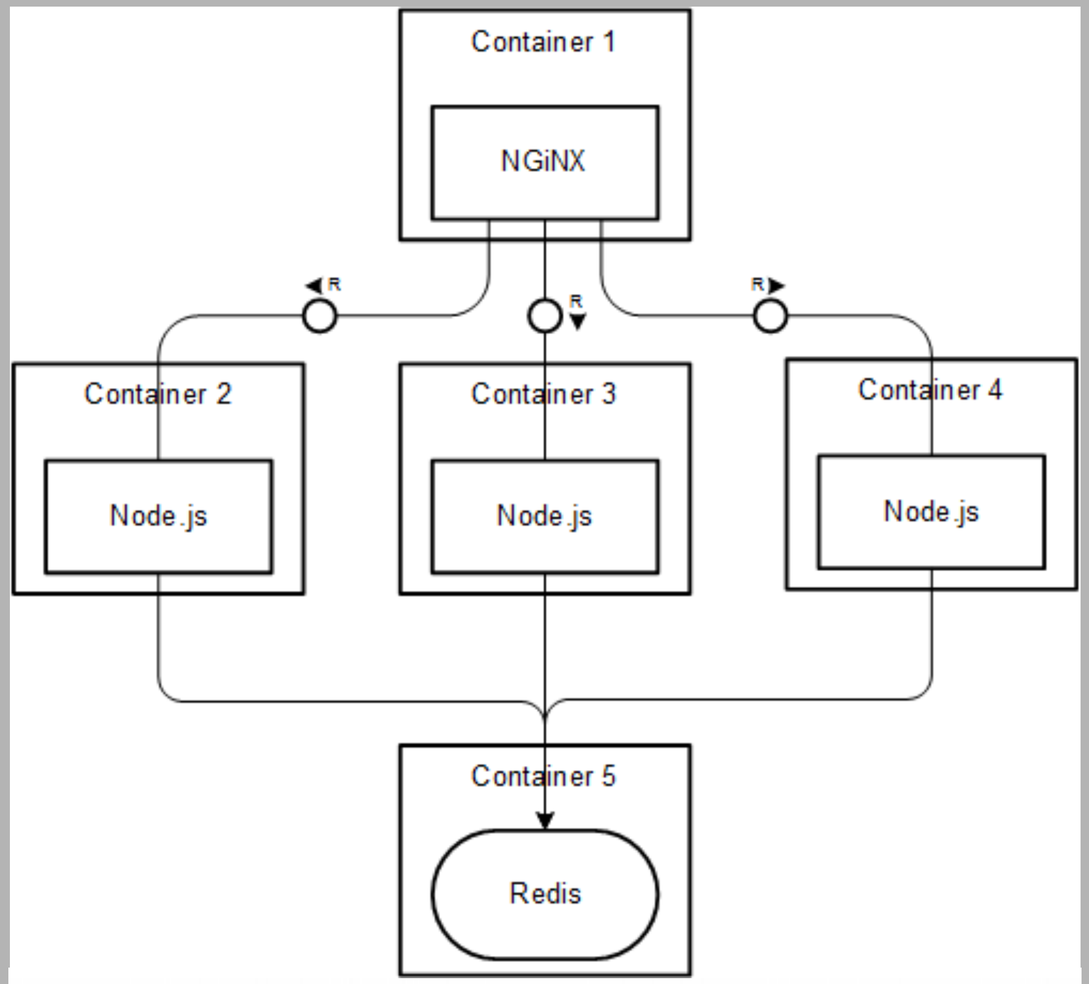


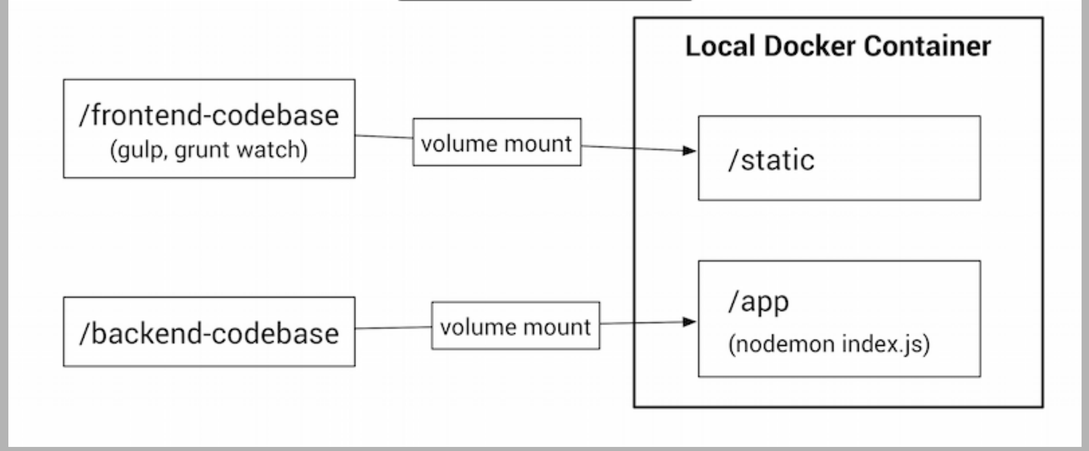
PUSH/DEPLOY WITH GIT

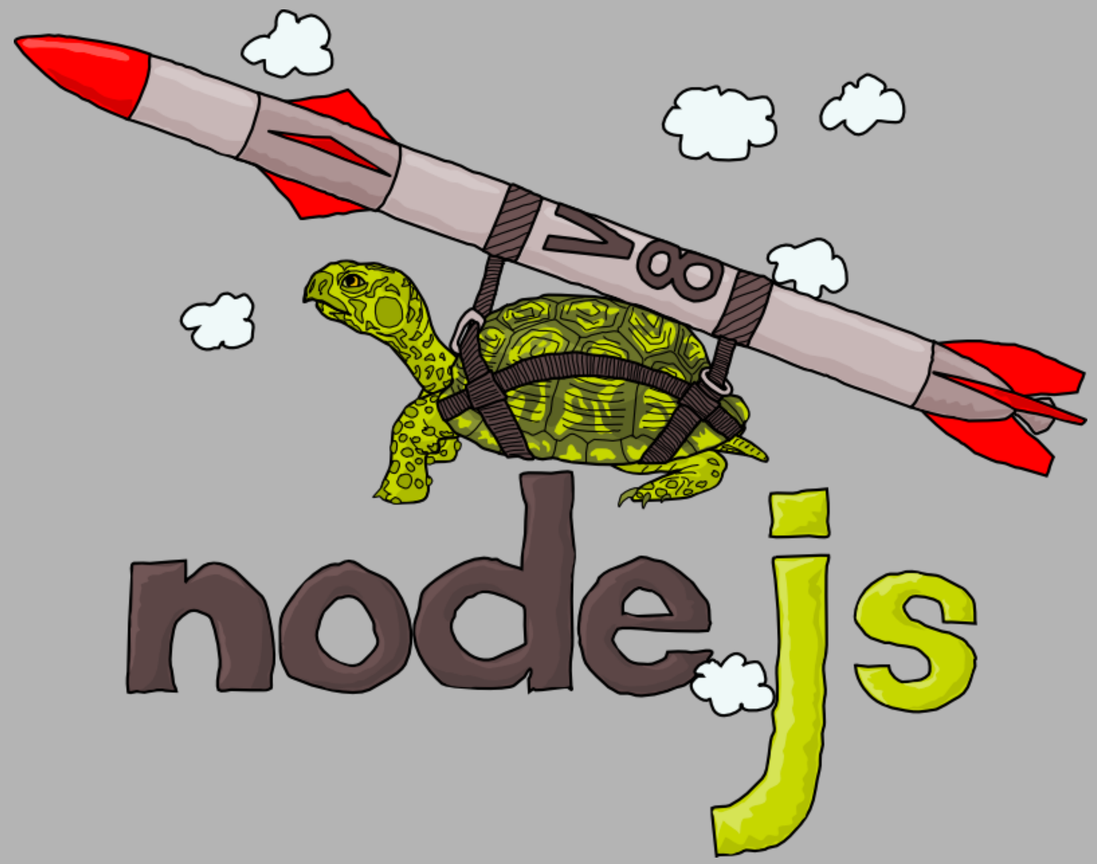




DOCKER NODEJS







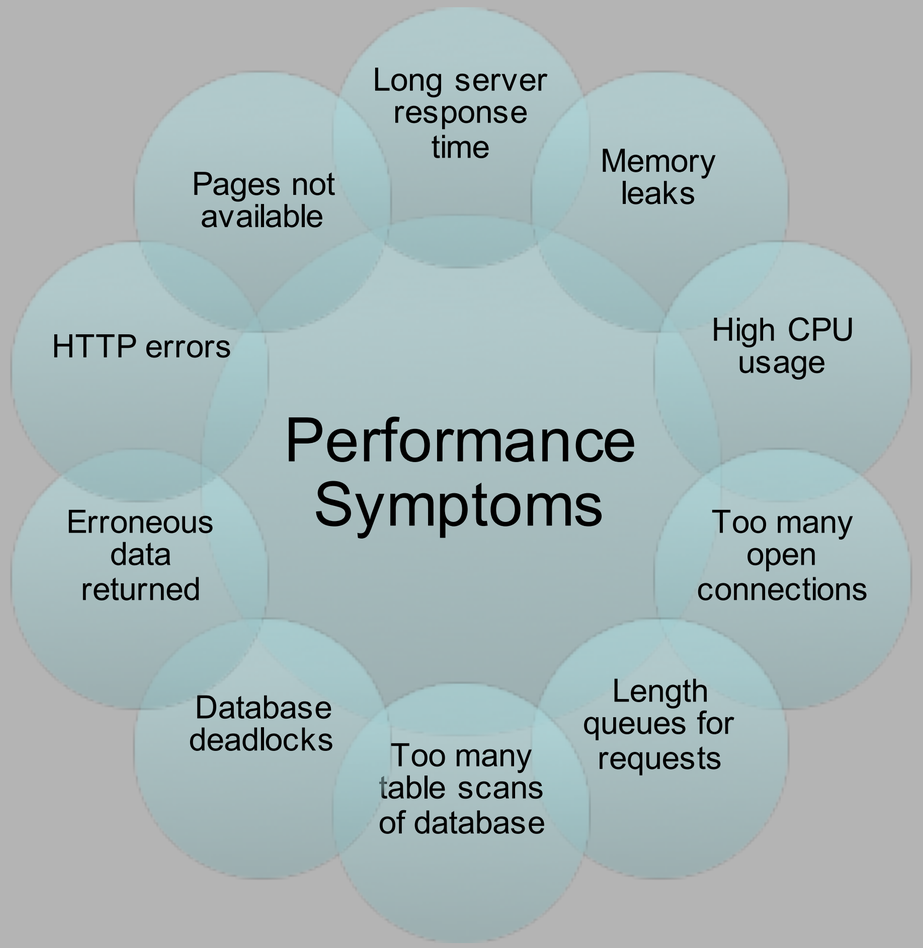
PERFORMANCE TESTING

WHAT IS PERFORMANCE TESTING

A non-functional testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage

GOAL/OBJECTIVE IS TO FIND THE BOTTLE NECK IN THE SYSTEM

SYMPTOMS OF PERFORMANCE PROBLEMS



Database

- Insufficient indexing
- Fragmented database
- Out-of-date statistics
- Faulty application design

Web Server

- Poor server design
- Memory problems
- High CPU usage

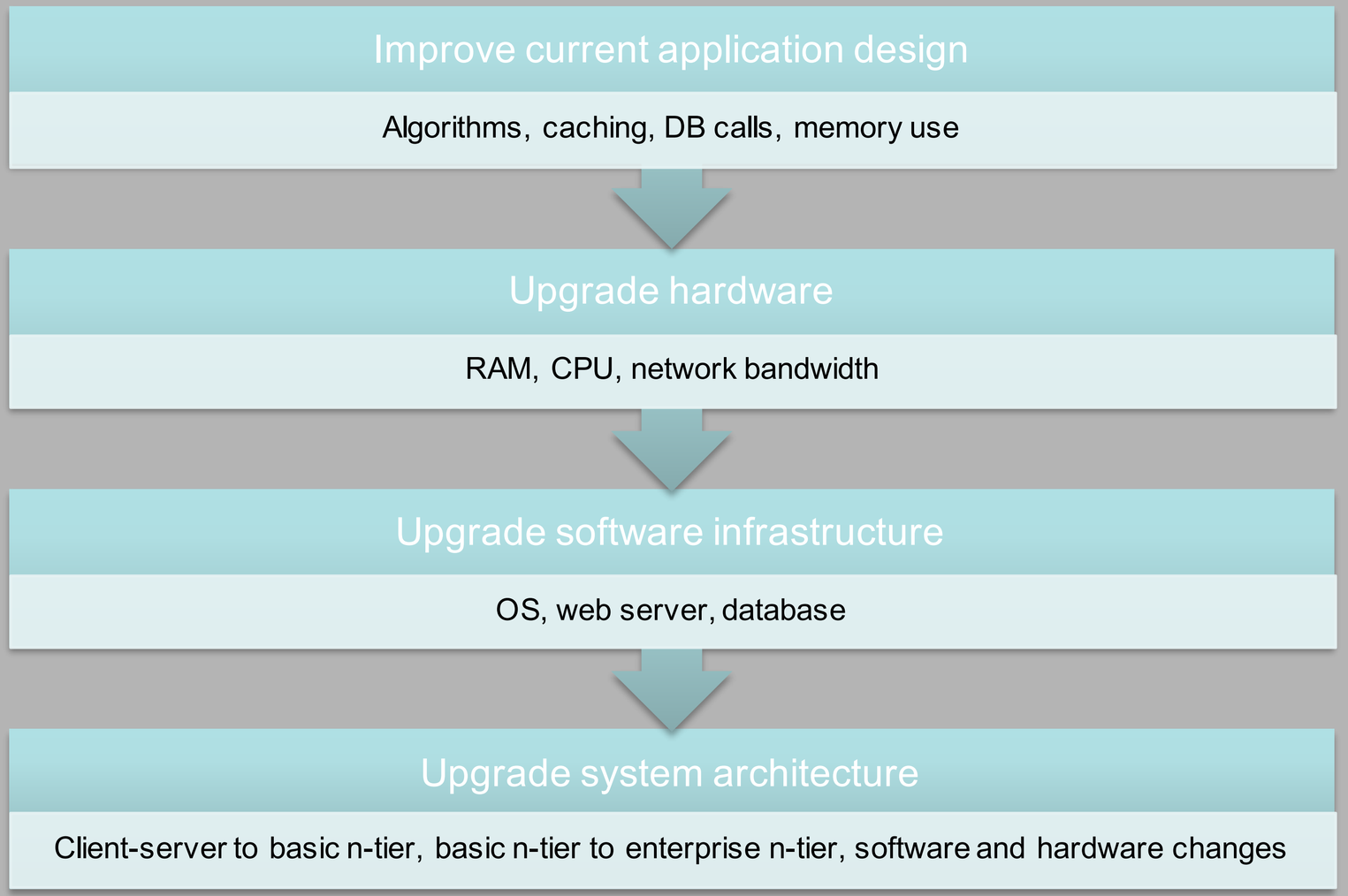
Application Server

- Poor database tuning
- Poor cache management
- Poor session management
- Poor security design

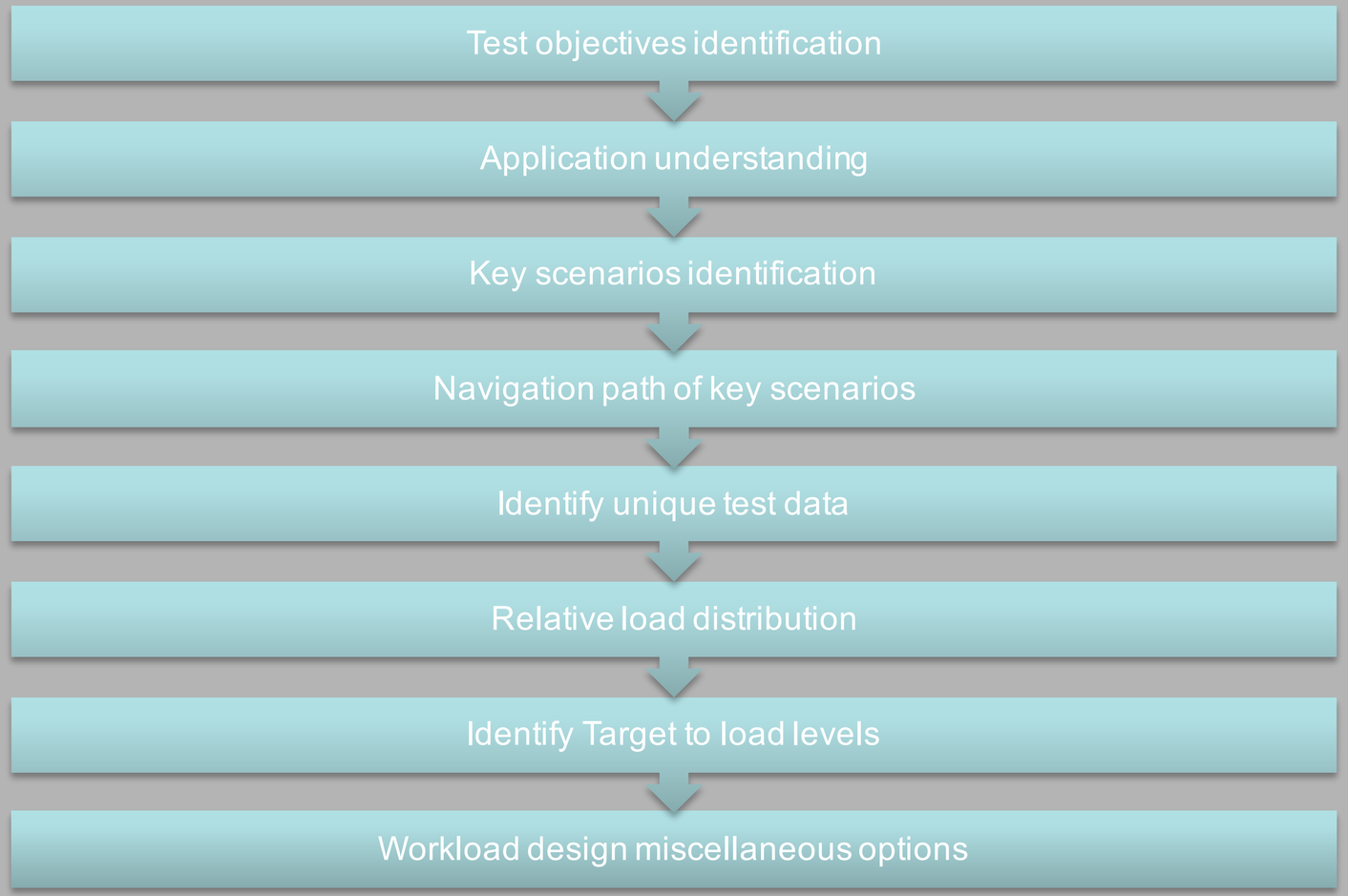
Network

- Firewall throughput
- Internet access throughput
- Load balancers, gateways, routers

SYMPTOMS OF PERFORMANCE PROBLEMS



WORKLOAD MODELING



TYPES OF PERFORMANCE TESTING

Load/Capacity Testing

Stress Testing

Volume Testing

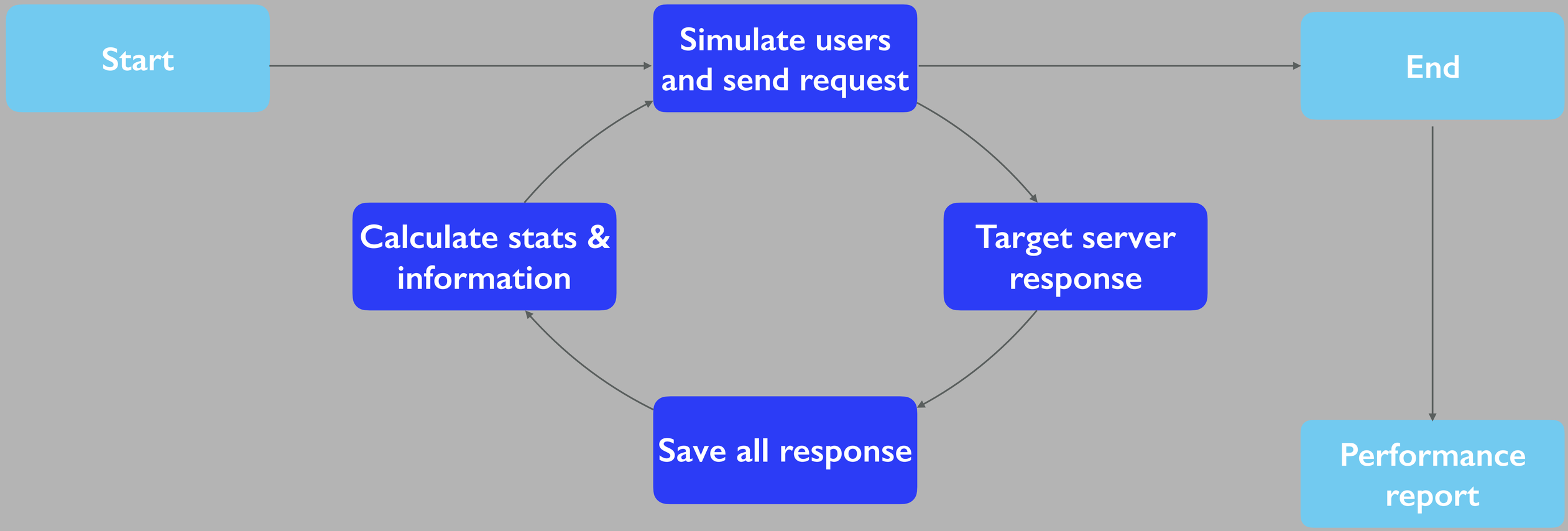
Endurance/Soak Testing

Spike Testing

PERFORMANCE TESTING TOOLS

Load Runner, commercial load testing tool from HP
JMeter, an open source tool from Apache
RPT, commercial load test tool from IBM
NeoLoad, commercial, for Windows, Linux, Solaris
Microsoft Visual Studio Team System 2010, commercial, for Windows, which includes Load Test Analyser and Load Test Monitor tools.
OpenLoad, commercial load testing tool and hosted service
OpenSTA, an open source tool
PureLoad, commercial, multi-platform load testing tool
The Grinder, an open source tool
WebServer Stress Tool, commercial and free, from Paessler

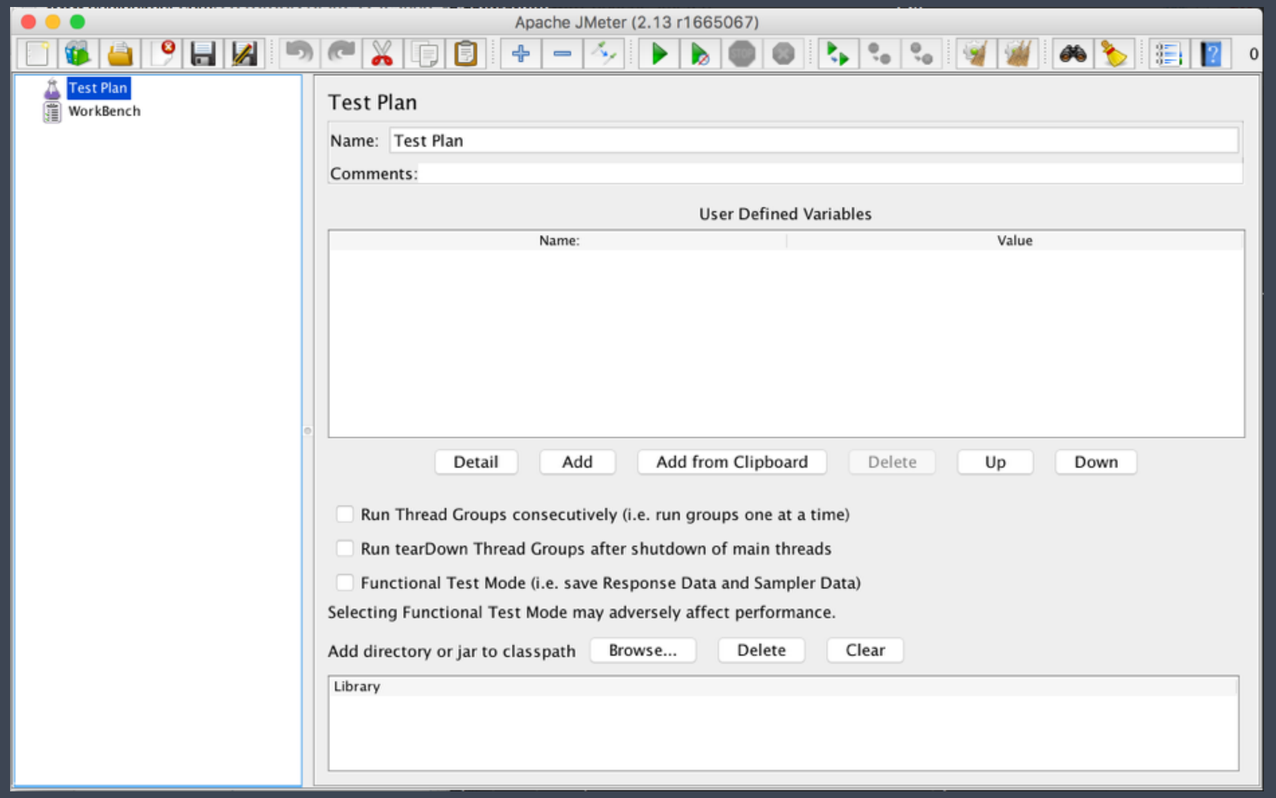
HOW JMETER WORK



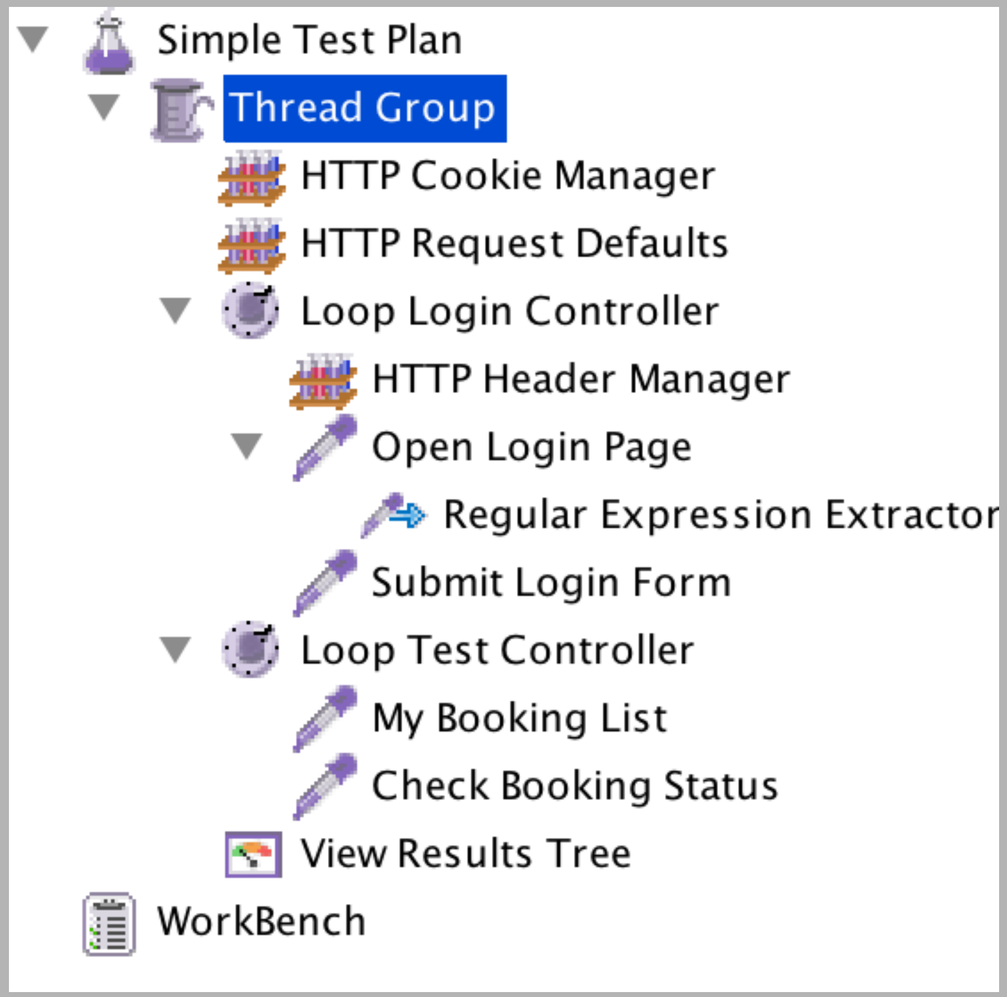
STARTING JMETER

$JMETER_HOME/bin/jmeter

$JMETER_HOME/bin/jmeter.bat



TEST PLAN BUILDING BLOCKS



Thread Group
- Setup number of thread
- Set up ramp up period
- No. of times test executes

Controllers
- Sampler (Send Request to Server)
- Logical Controller (Customize logic to send request)

Listener
- Graph Result
- View Results Tree and many more.

Timers
- Delay next request for certain amount of time

Assertions
- Allow you to assert fact about responses received from HTTP request

Configuration Elements
- Allow you configure settings

Preprocessor
- Execute prior to sampler request

Post Processor
- Execute some action after sampler request

LAB 2-6

Basic Load Testing

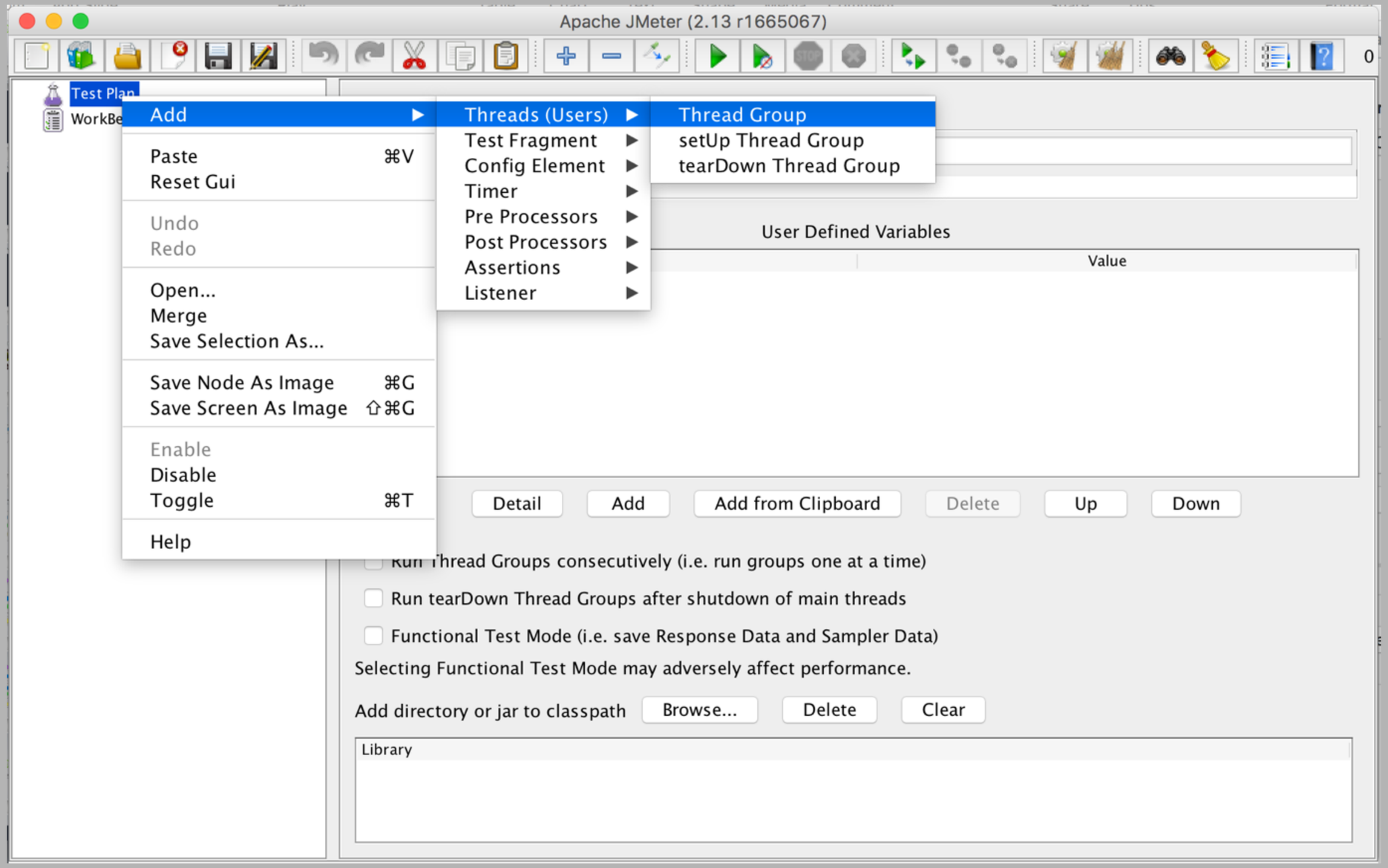
ASSIGNMENT

Given Scenarios:

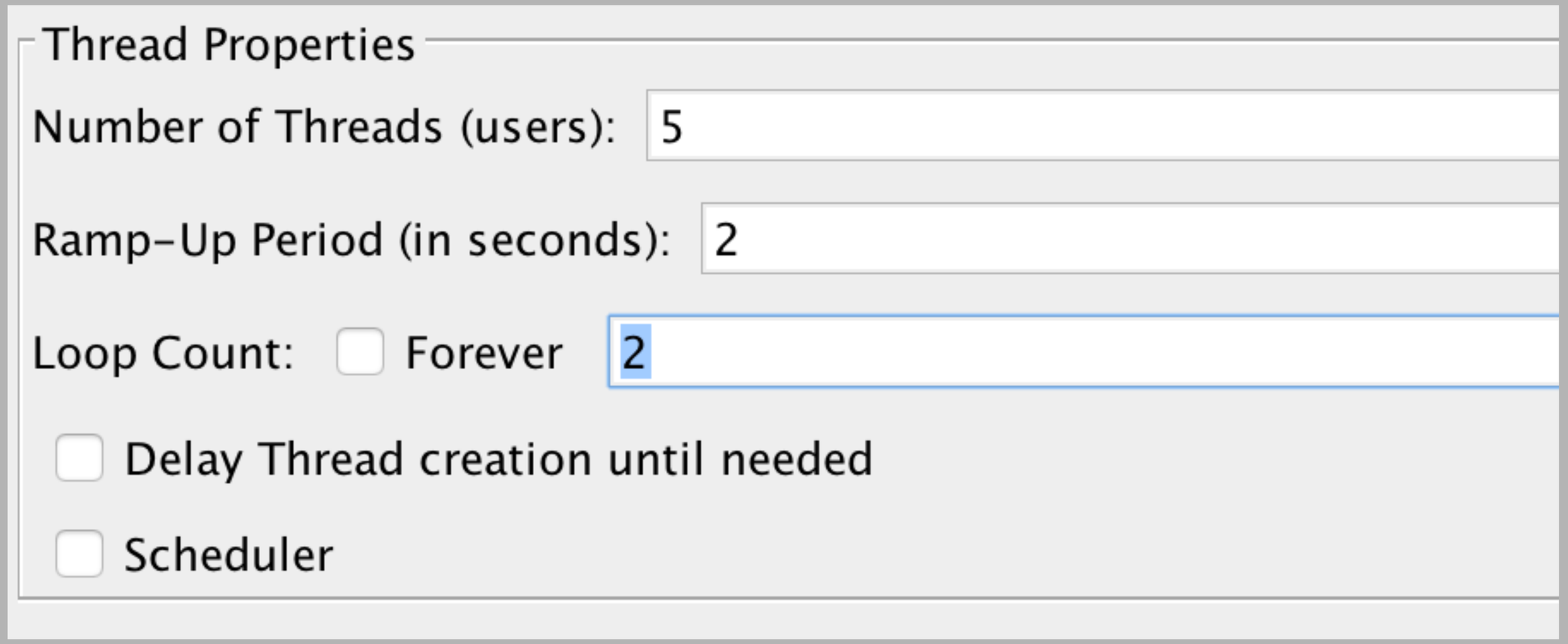
Create a test plan having 5 Users which send 2 request to your API and repeat it twice.

$$= 5 \times 2 \times 2$$

CREATE THREAD GROUP

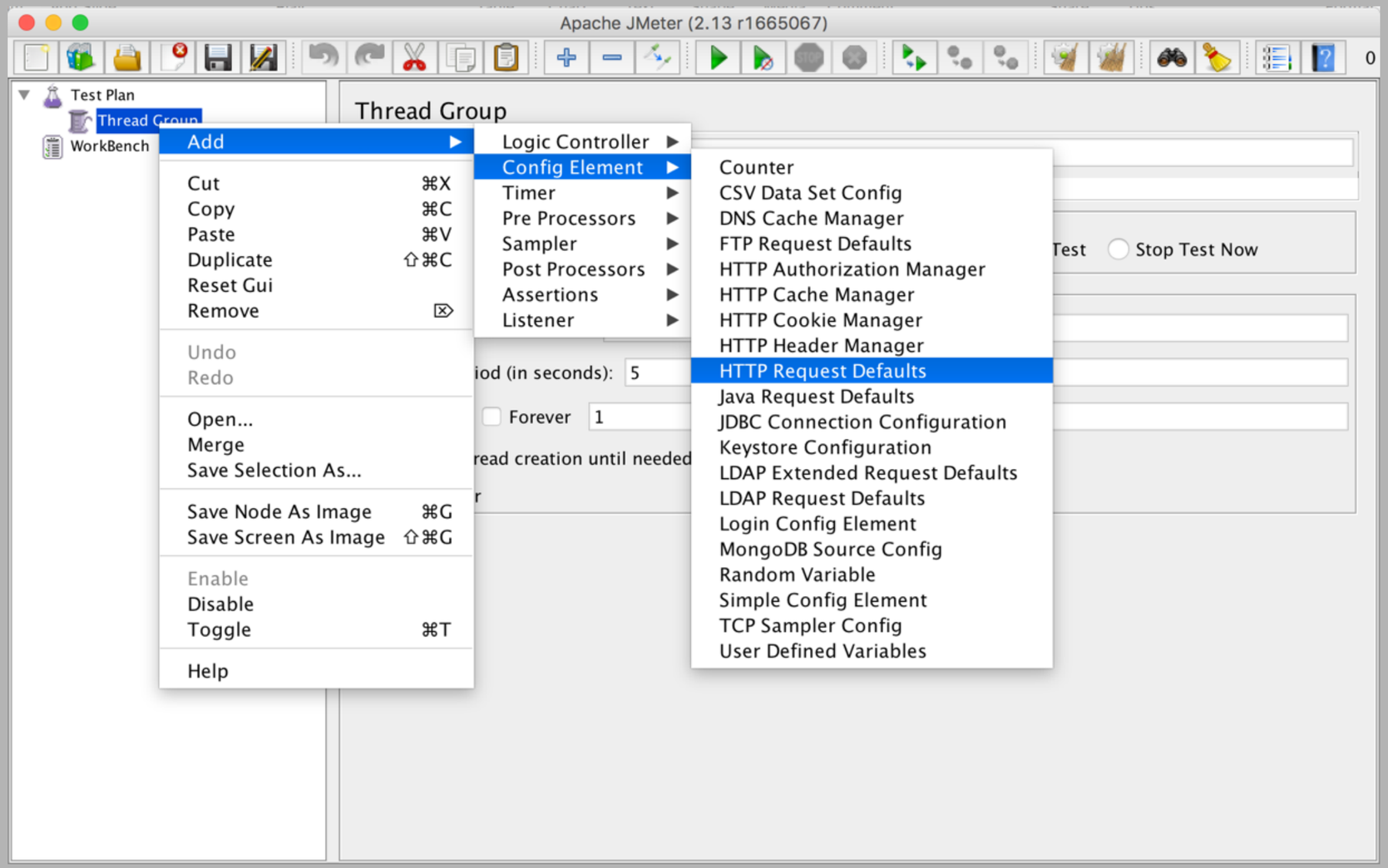


CREATE THREAD GROUP



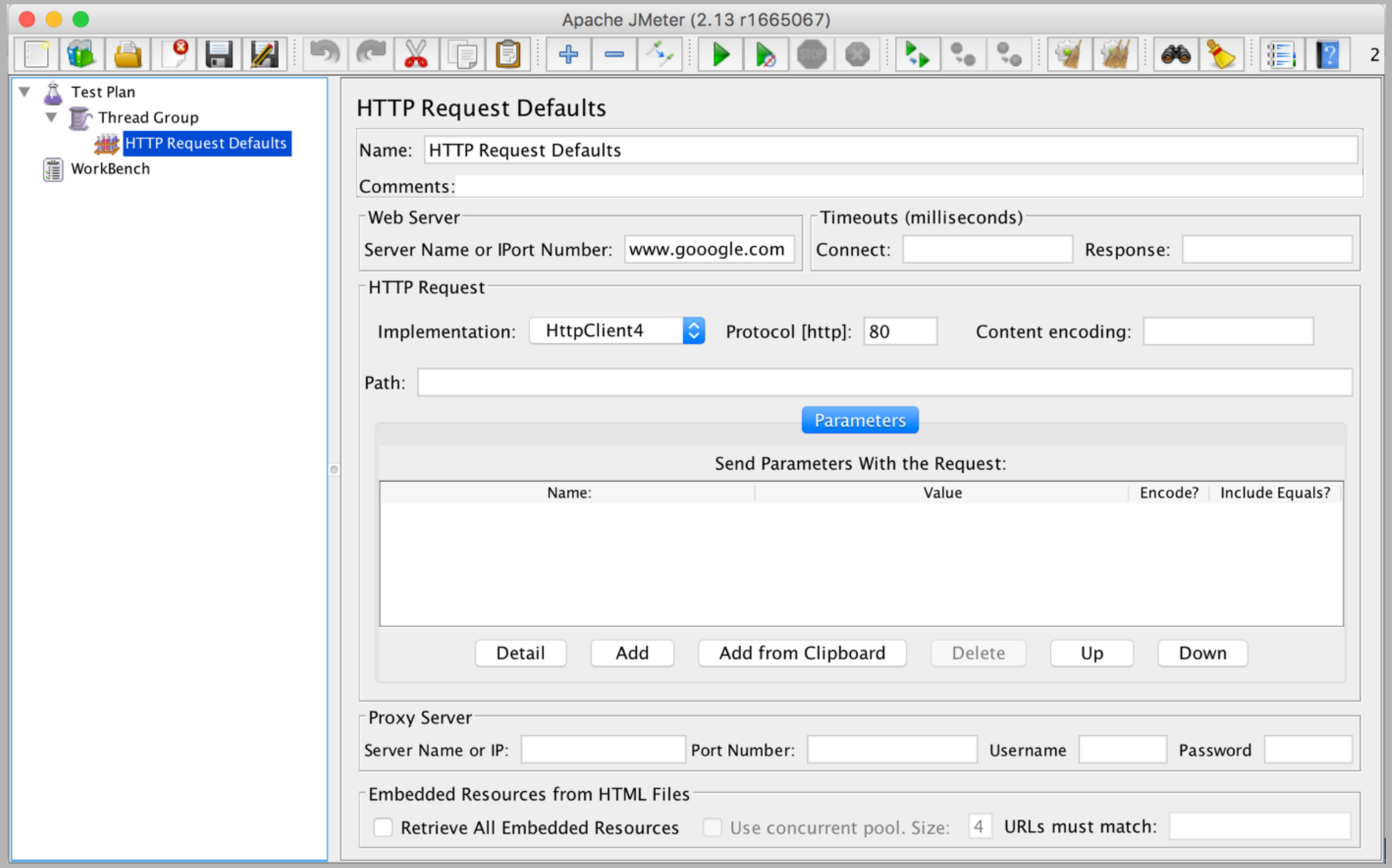
- Number of Threads (users): The number of users that JMeter will attempt to simulate.
- Ramp-Up Period (in seconds): The duration of time that JMeter will distribute the start of the threads over.
- Loop Count: The number of times to execute the test.

CREATE CONFIG ELEMENT

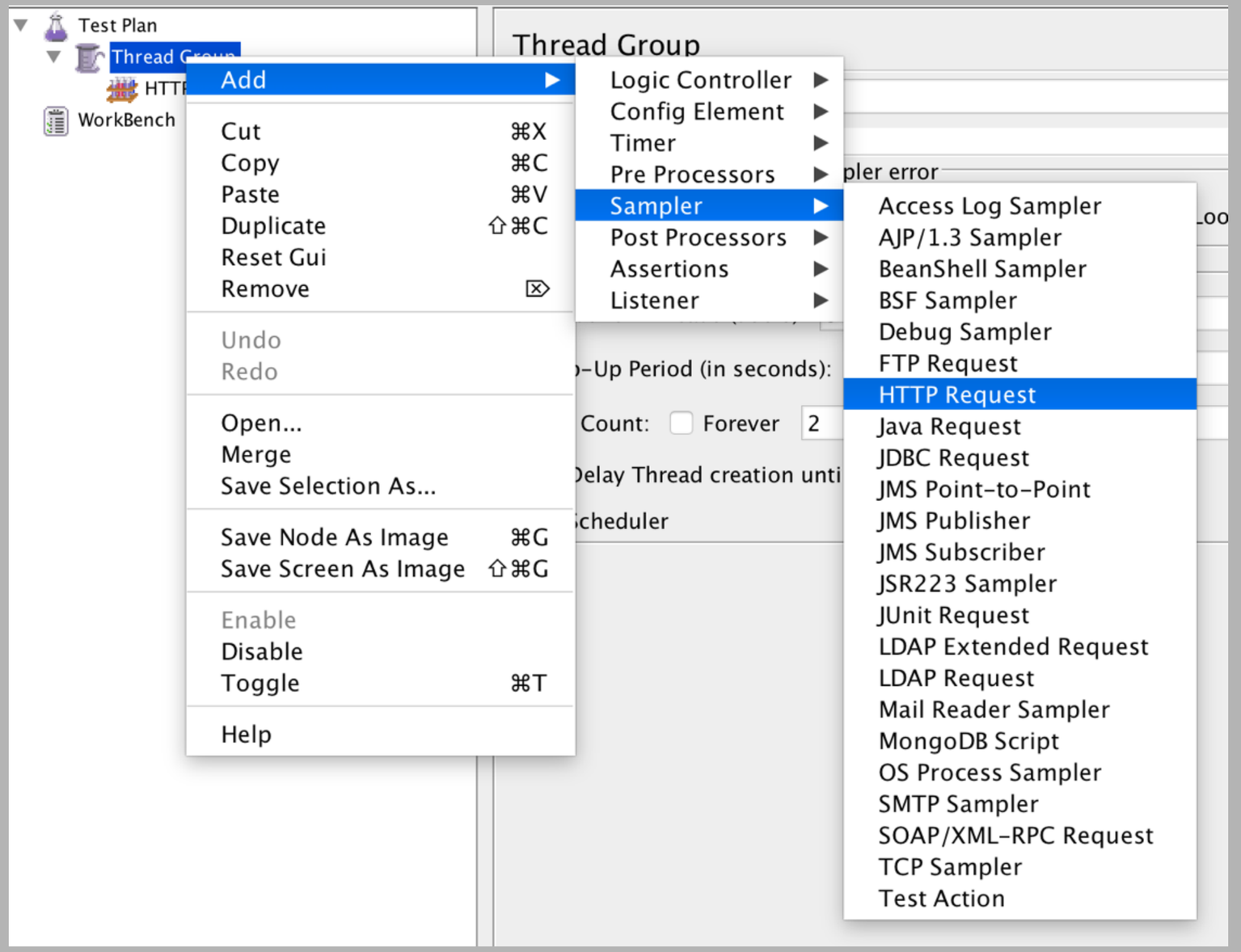


- HTTP Request Defaults
- HTTP Header Manager
- HTTP Cookie Manager
- HTTP Cache Manager
- HTTP Authorisation Manager

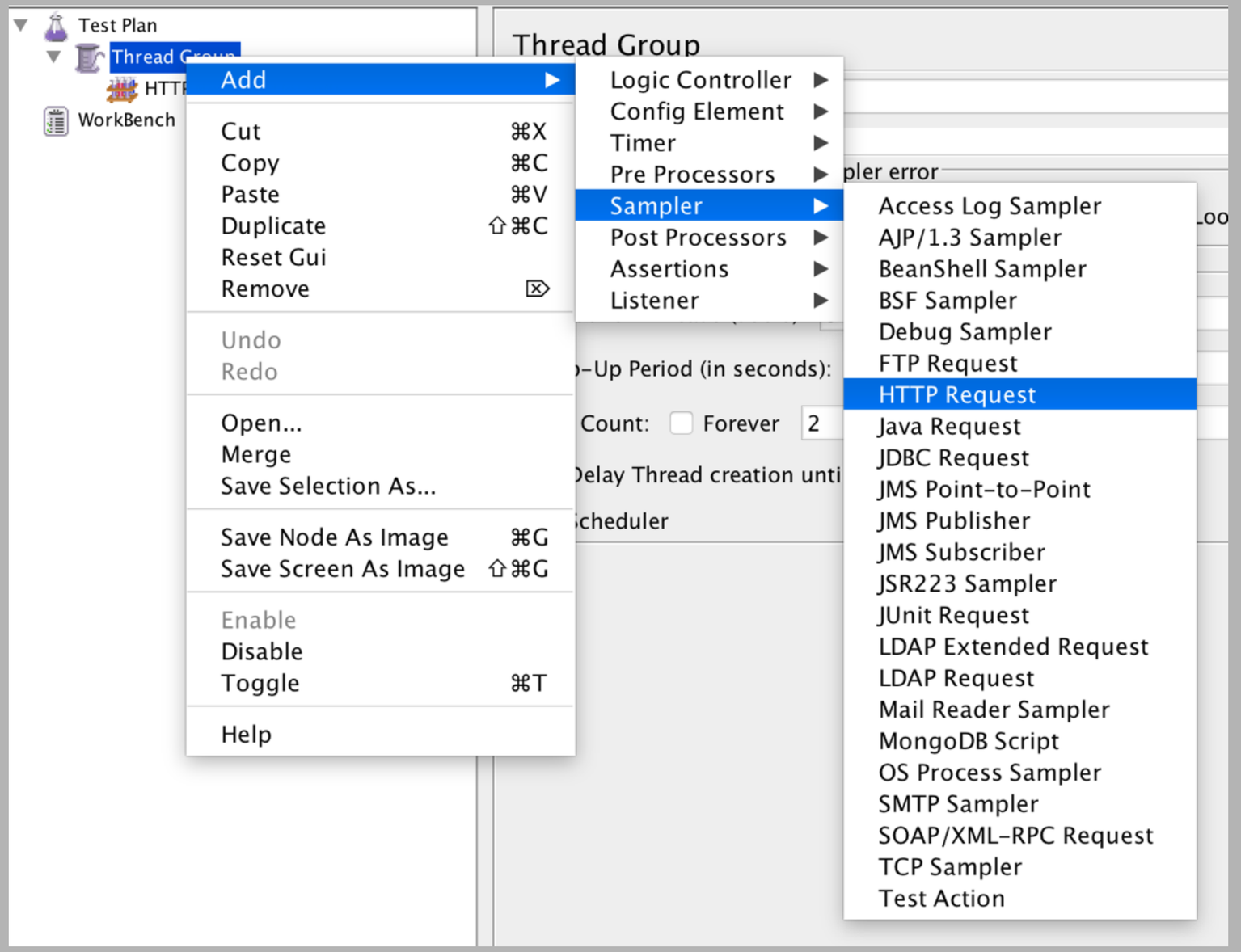
CREATE HTTP REQUEST DEFAULT



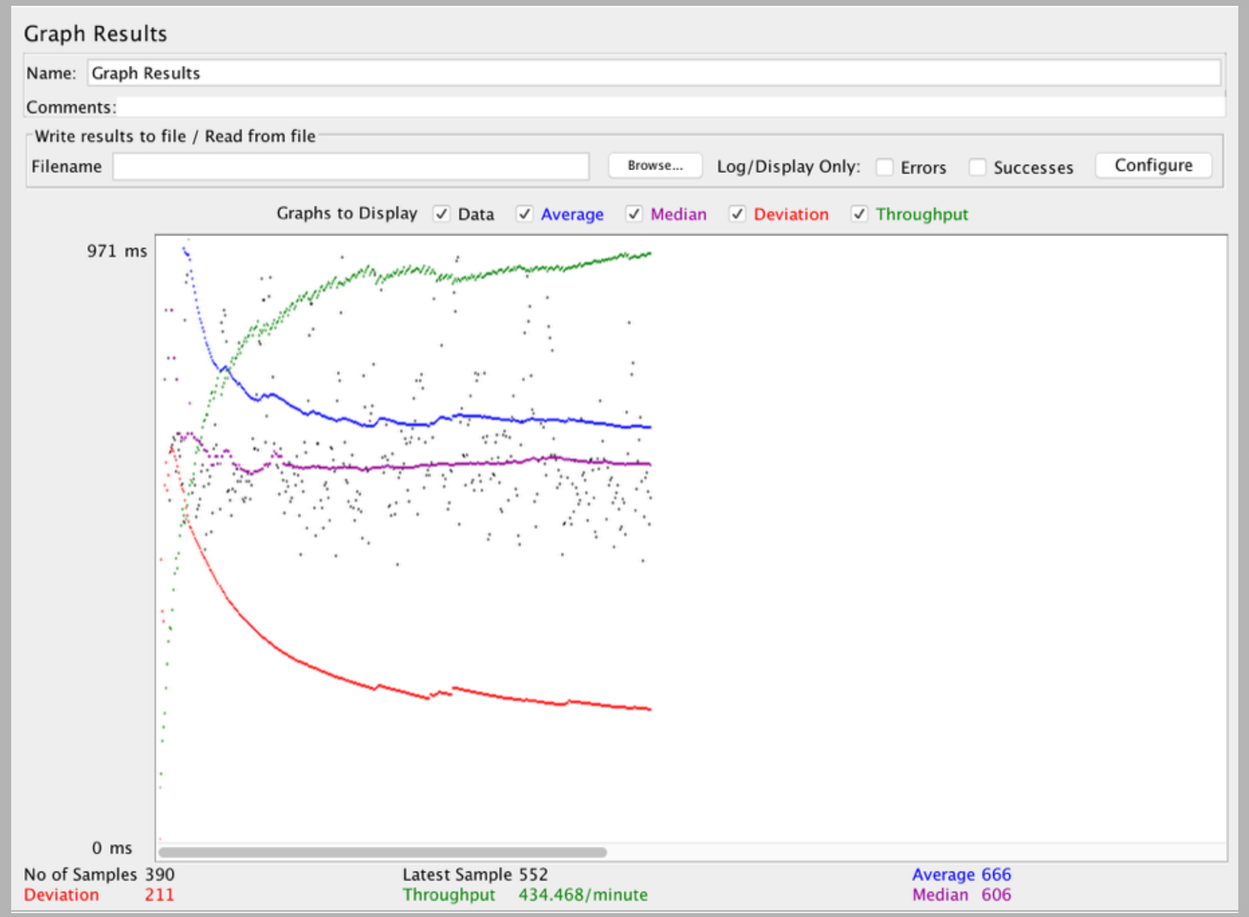
CREATE HTTP REQUEST



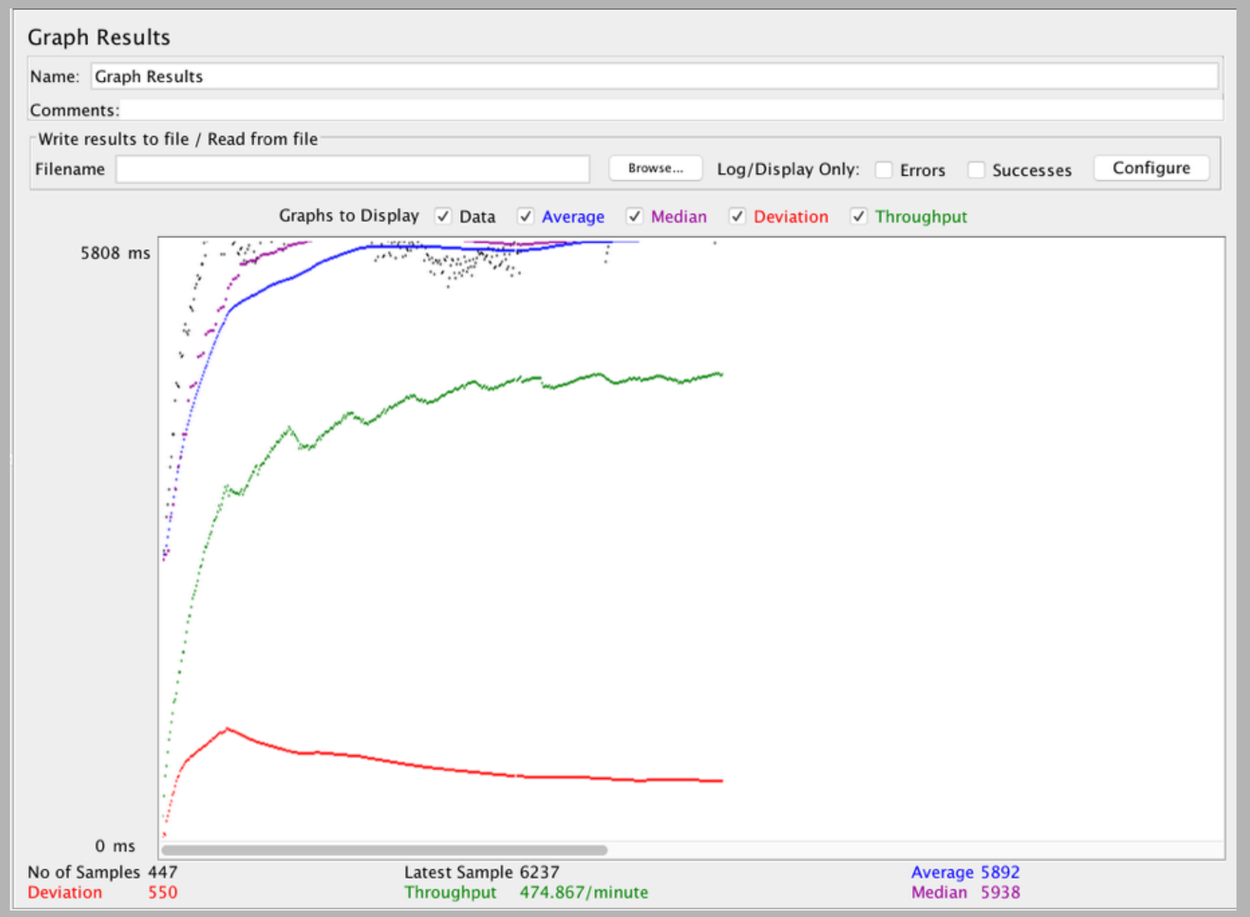
CREATE HTTP REQUEST



GRAPH RESULT



5 Users



50 Users

SUMMARY REPORT

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
API1	490	620	67	1742	168.28	1.02%	7.7/sec	140.78	18660.7
TOTAL	490	620	67	1742	168.28	1.02%	7.7/sec	140.78	18660.7

5 Users

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

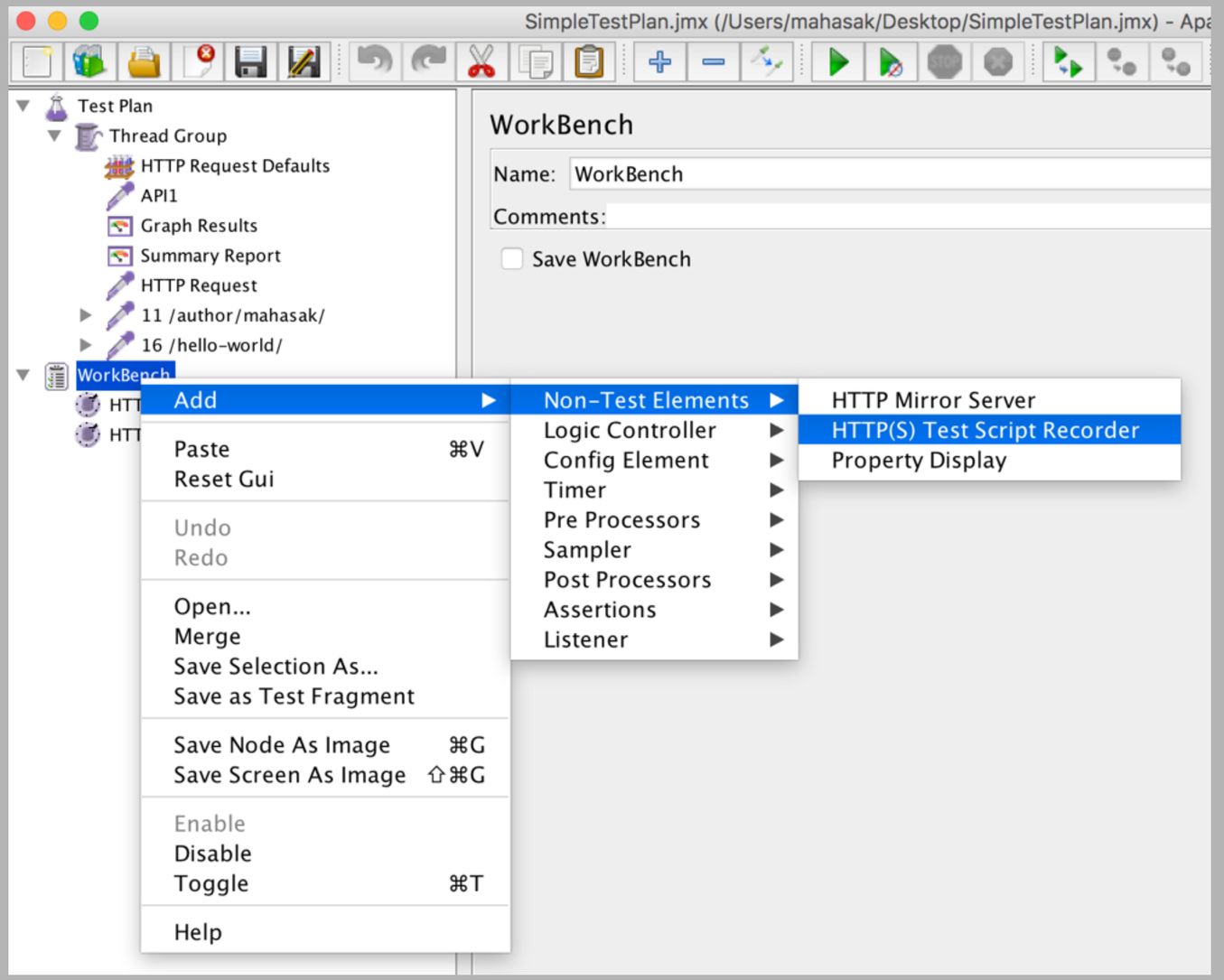
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
API1	518	5273	3	7168	1070.90	9.46%	9.3/sec	157.35	17400.4
TOTAL	518	5273	3	7168	1070.90	9.46%	9.3/sec	157.35	17400.4

50 Users

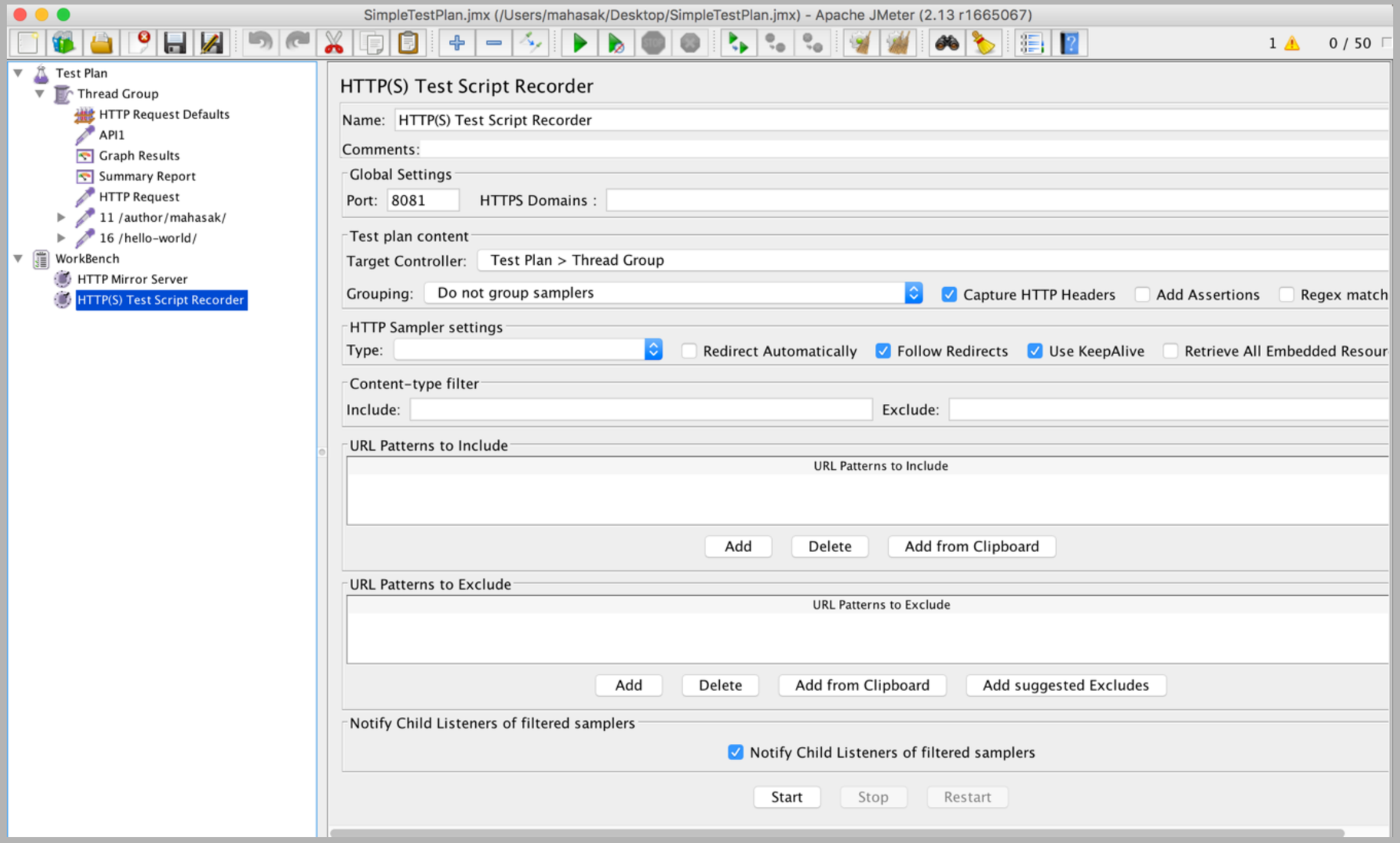
LAB 2-7

Recording Test Plan

CREATE TEST SCRIPT RECORDER



CONFIGURE TEST RECORDER



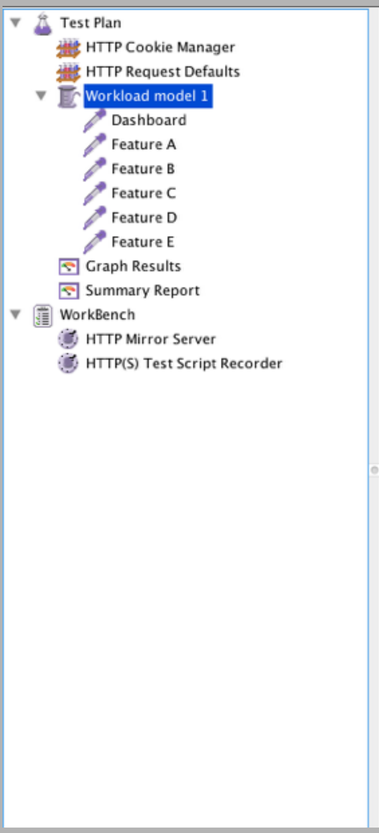
LAB 2-8

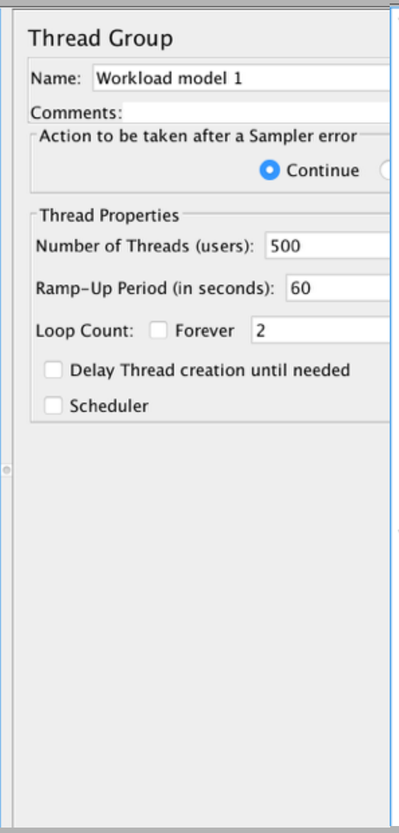
Design Workload Model

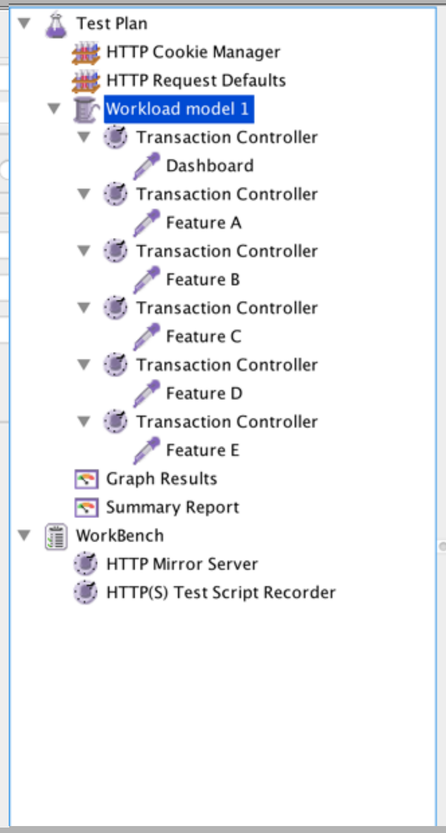
DESIGN WORKLOAD MODEL

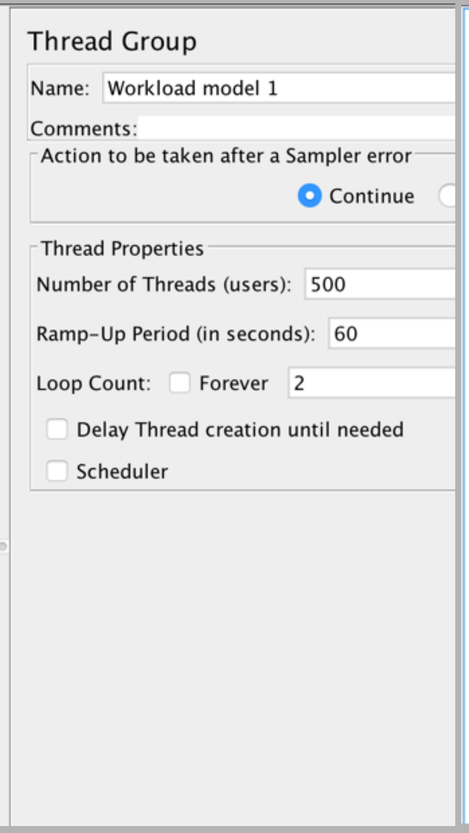
Workload Model		
User Scenarios	Percentage of workload distribution	Active users (500)
Dashboard	40	200
Feature A	10	50
Feature B	10	50
Feature C	20	100
Feature D	20	100

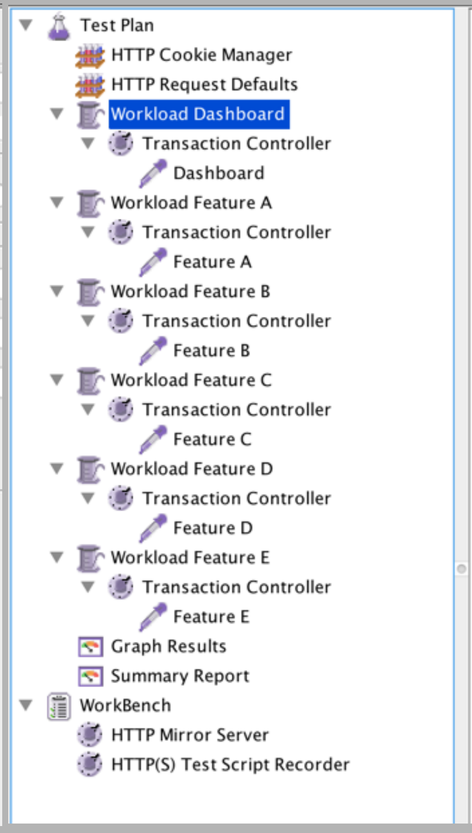
DESIGN WORKLOAD MODEL

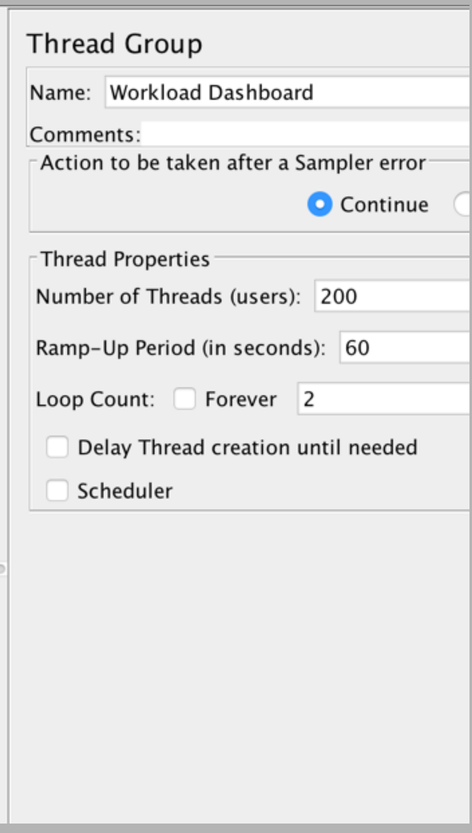












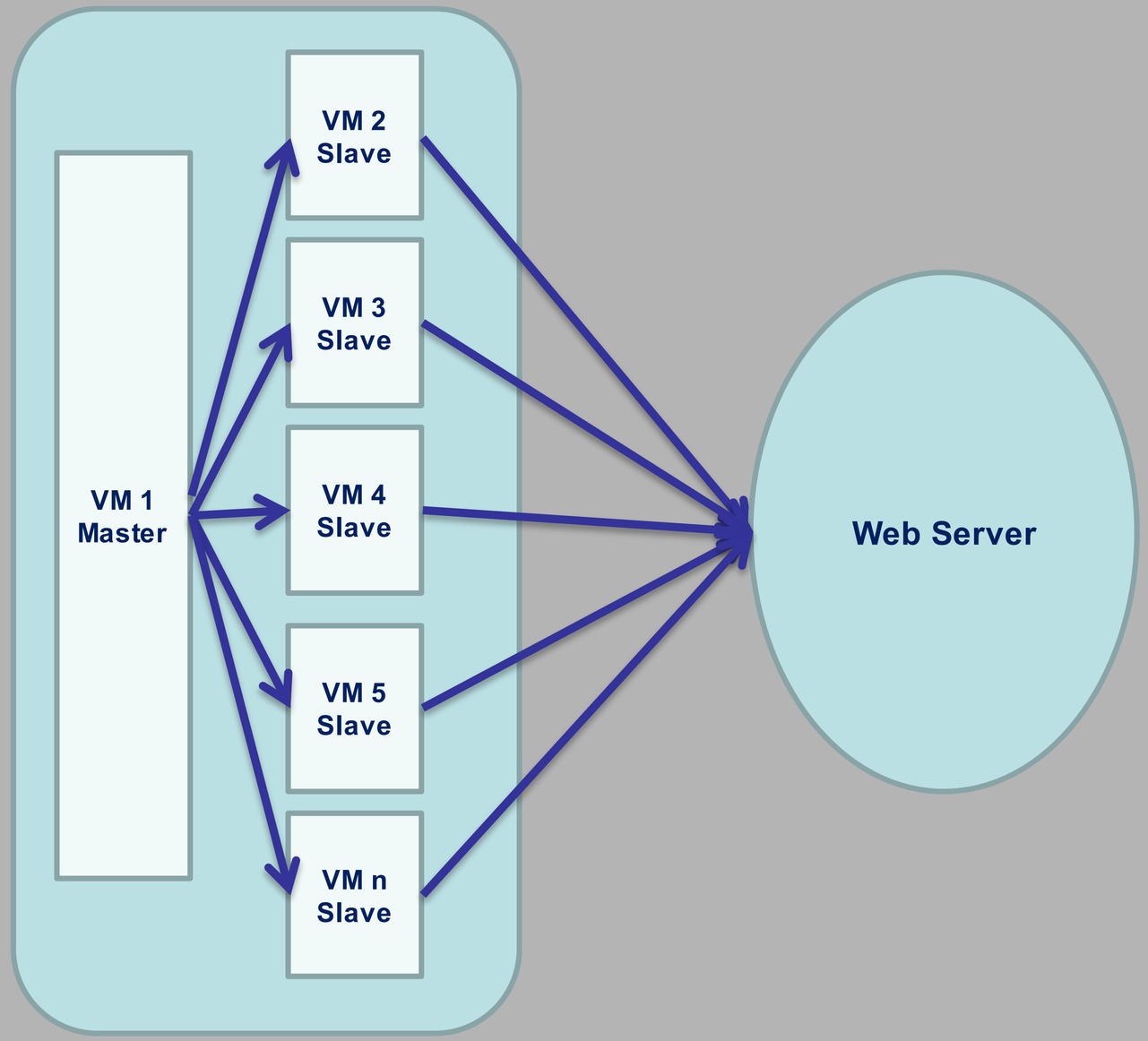
http://jmeter-plugins.org/

RECOMMENDED PLUGIN TO EXPLORE

jp@gc — Stepping Thread Group
jp@gc — Response Time vs Threads
jp@gc — Active Threads Over Time
jp@gc — Response Codes per Second
jp@gc — Response Times Distribution
jp@gc - Transactions per Second
jp@gc - Response Codes per Second

http://jmeter-plugins.org/

CREATE HTTP REQUEST DEFAULT



Master: The system running Jmeter GUI which control the test.

Slave: The System running Jmeter-server which takes commends from the GUI and send the requests to the target system.

Target: The Web Server planned for the load test.

PERFORMANCE MONITORING

http://newrelic.com/application-monitoring

https://keymetrics.io/

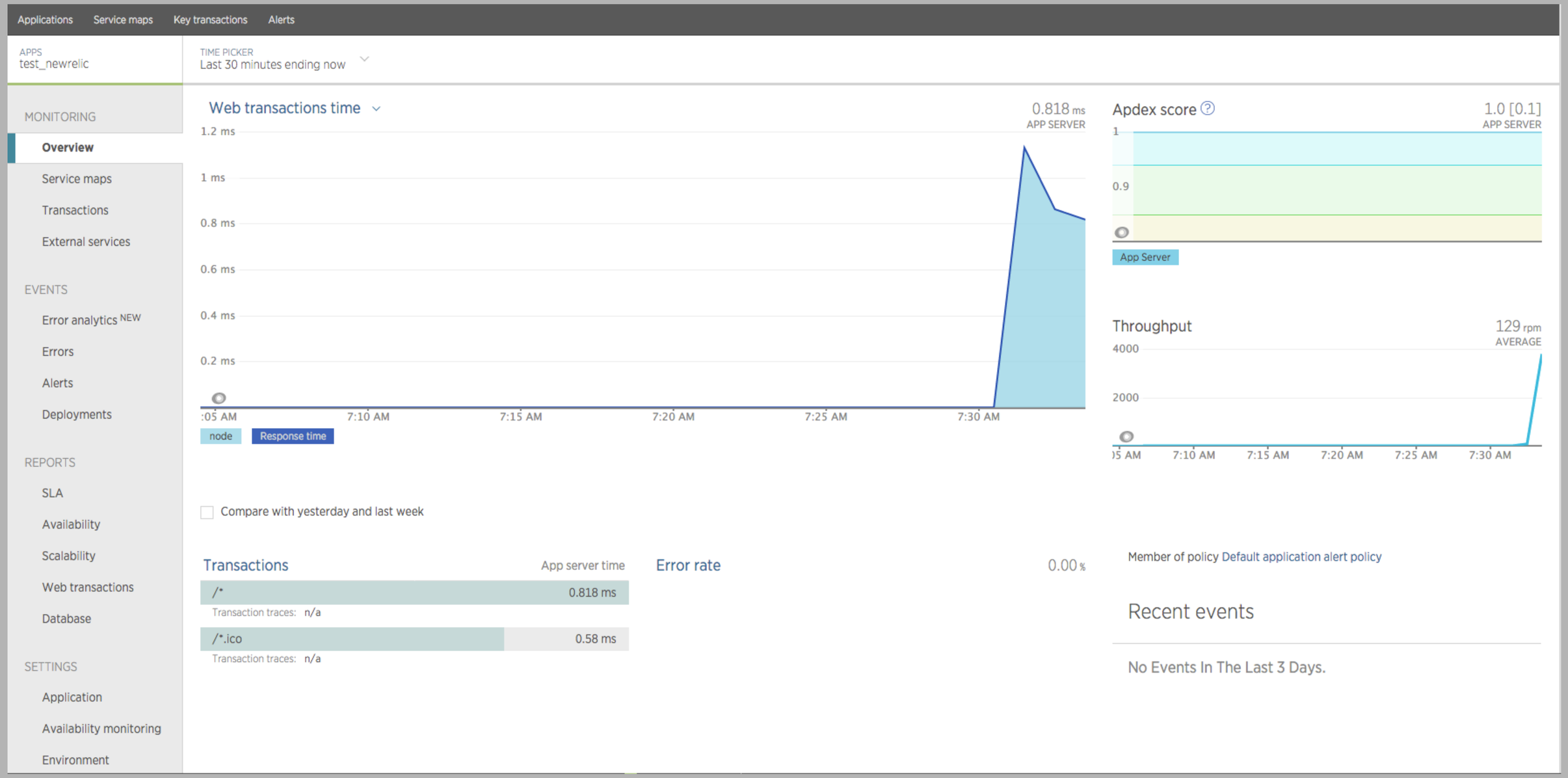
NEWRELIC

$ npm install newrelic

Copy newrelic.js from node_modules/newrelic into the root directory
Set a value for app_name, license_key

```
require('newrelic');
```

NEWRELIC


Applications
Service maps
Key transactions
Alerts
APPS
test_newrelic
TIME PICKER
Last 30 minutes ending now
MONITORING
Overview
Service maps
Transactions
External services
EVENTS
Error analytics NEW
Errors
Alerts
Deployments
REPORTS
SLA
Availability
Scalability
Web transactions
Database
SETTINGS
Application
Availability monitoring
Environment
Web transactions time
0.818 ms
APP SERVER
1.2 ms
1 ms
0.8 ms
0.6 ms
0.4 ms
0.2 ms
:05 AM
7:10 AM
7:15 AM
7:20 AM
7:25 AM
7:30 AM
node
Response time
Apdex score
1.0 [0.1]
APP SERVER
1
0.9
App Server
Throughput
129 rpm
AVERAGE
4000
2000
7:10 AM
7:15 AM
7:20 AM
7:25 AM
7:30 AM
Compare with yesterday and last week
Transactions
App server time
/*
0.818 ms
Transaction traces: n/a
/*.ico
0.58 ms
Transaction traces: n/a
Error rate
0.00 %
Member of policy Default application alert policy
Recent events
No Events In The Last 3 Days.

LAB 2-9

Newrelic

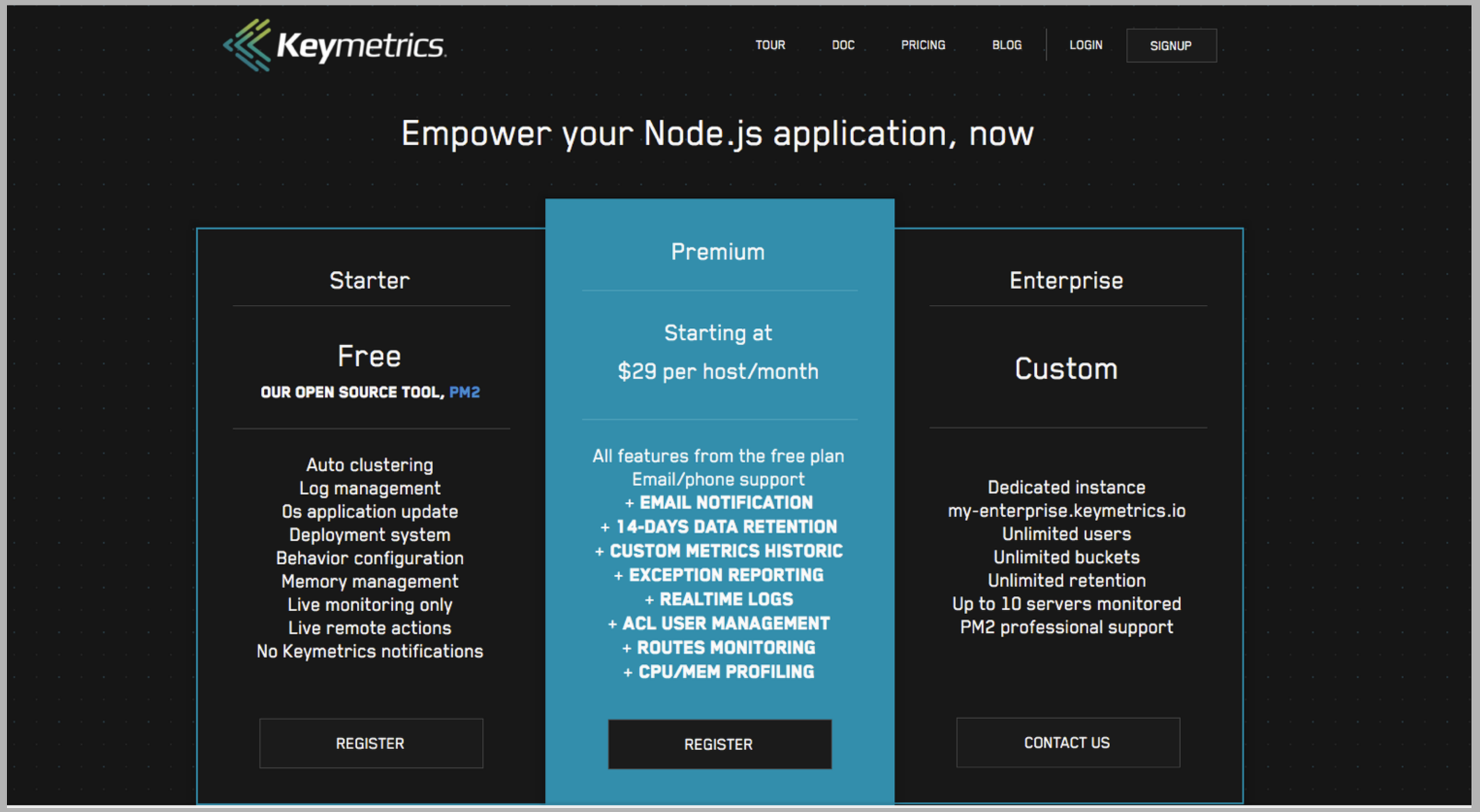
KEYMETRICS

$ npm install newrelic

Copy newrelic.js from node_modules/newrelic into the root directory
Set a value for app_name, license_key

```
require('newrelic');
```

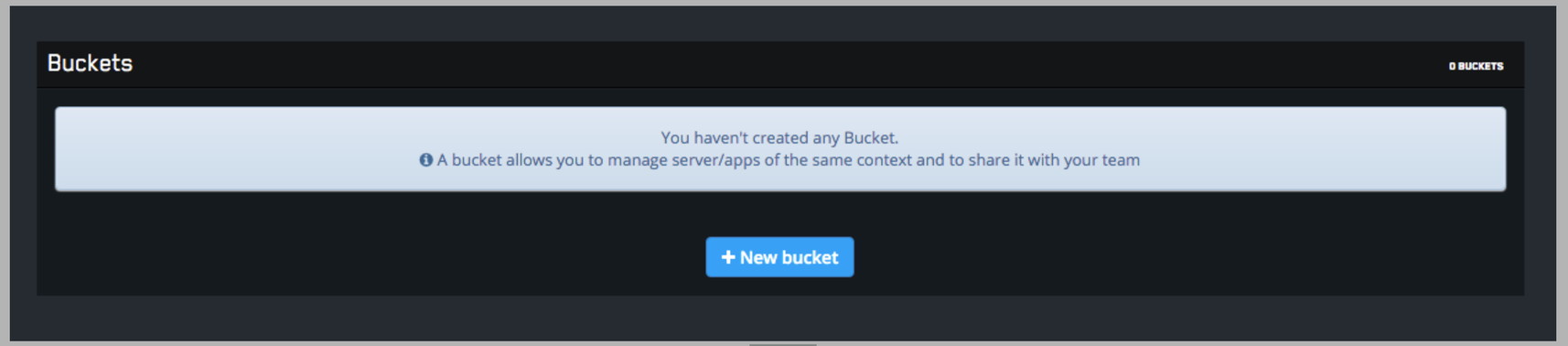
KEYMETRICS

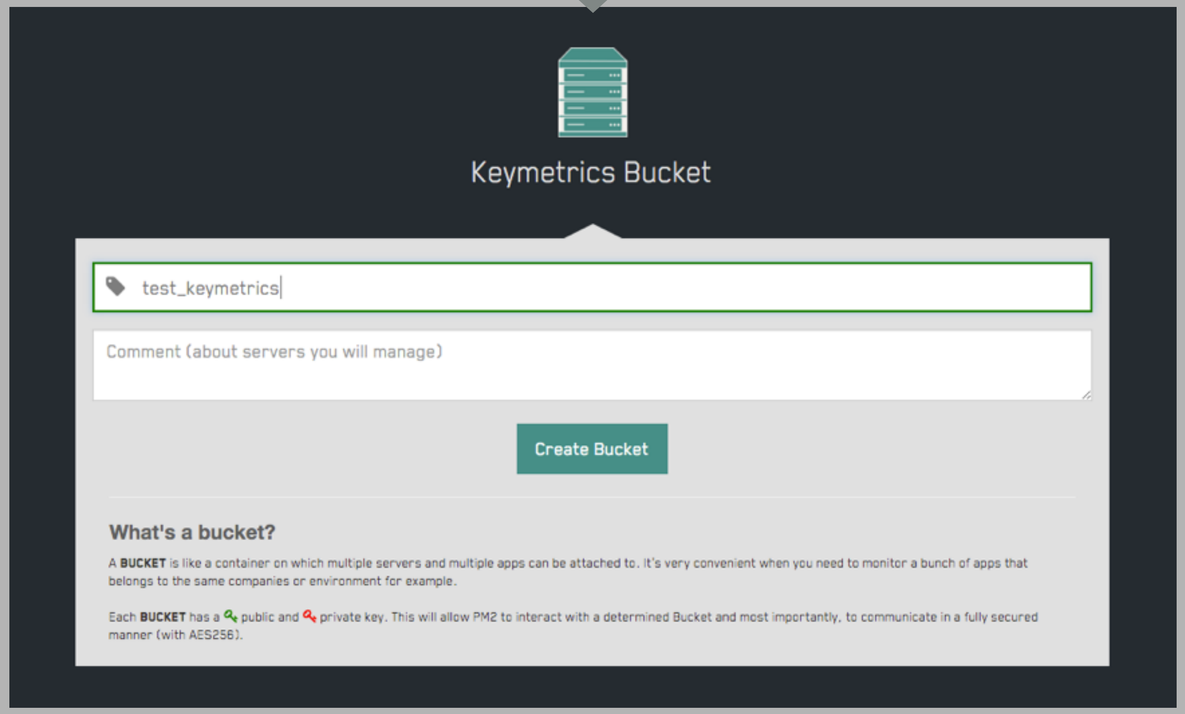

Keymetrics
TOUR
DOC
PRICING
BLOG
LOGIN
SIGNUP
Empower your Node.js application, now
Starter
Free
OUR OPEN SOURCE TOOL, PM2
Auto clustering
Log management
Os application update
Deployment system
Behavior configuration
Memory management
Live monitoring only
Live remote actions
No Keymetrics notifications
REGISTER
Premium
Starting at
$29 per host/month
All features from the free plan
Email/phone support
+ EMAIL NOTIFICATION
+ 14-DAYS DATA RETENTION
+ CUSTOM METRICS HISTORIC
+ EXCEPTION REPORTING
+ REALTIME LOGS
+ ACL USER MANAGEMENT
+ ROUTES MONITORING
+ CPU/MEM PROFILING
REGISTER
Enterprise
Custom
Dedicated instance
my-enterprise.keymetrics.io
Unlimited users
Unlimited buckets
Unlimited retention
Up to 10 servers monitored
PM2 professional support
CONTACT US

LAB 2-9

Keymetrics

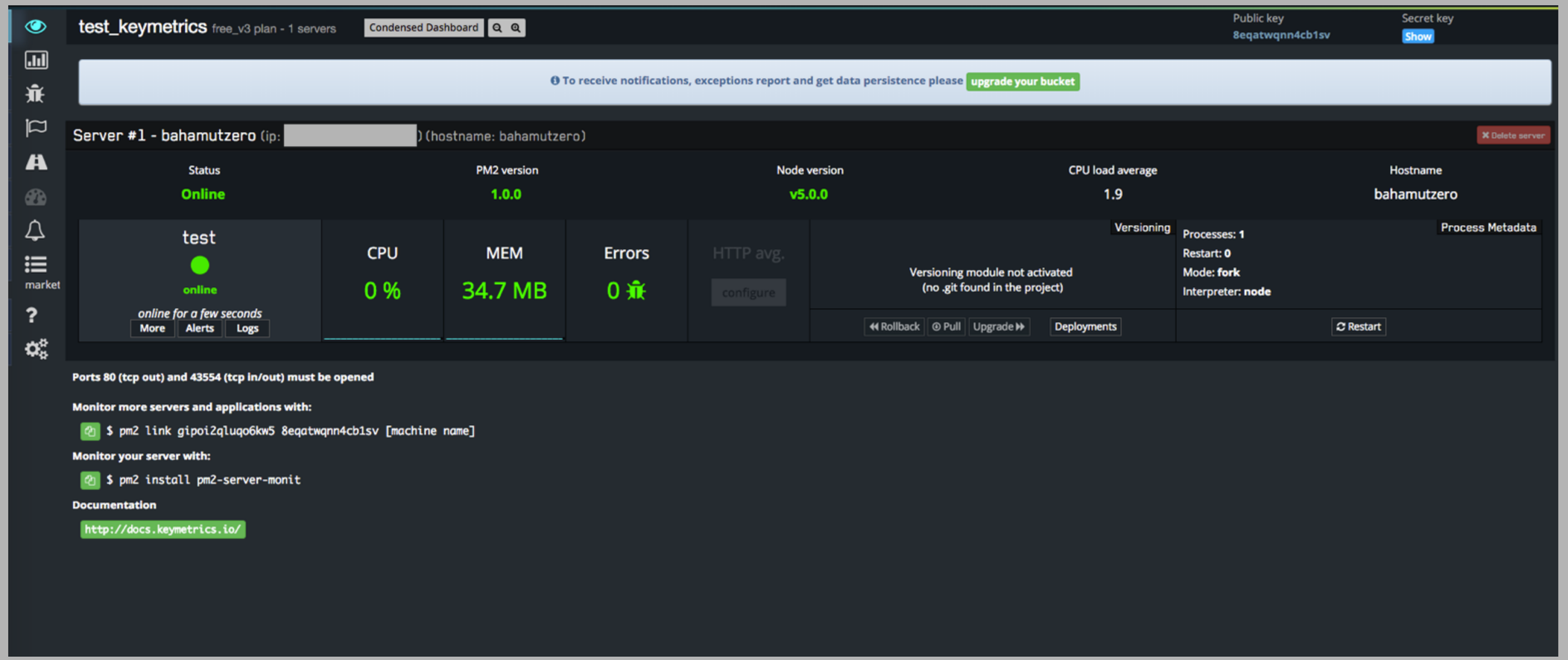
KEYMETRICS


Buckets
0 BUCKETS
You haven't created any Bucket.
A bucket allows you to manage server/apps of the same context and to share it with your team
+ New bucket

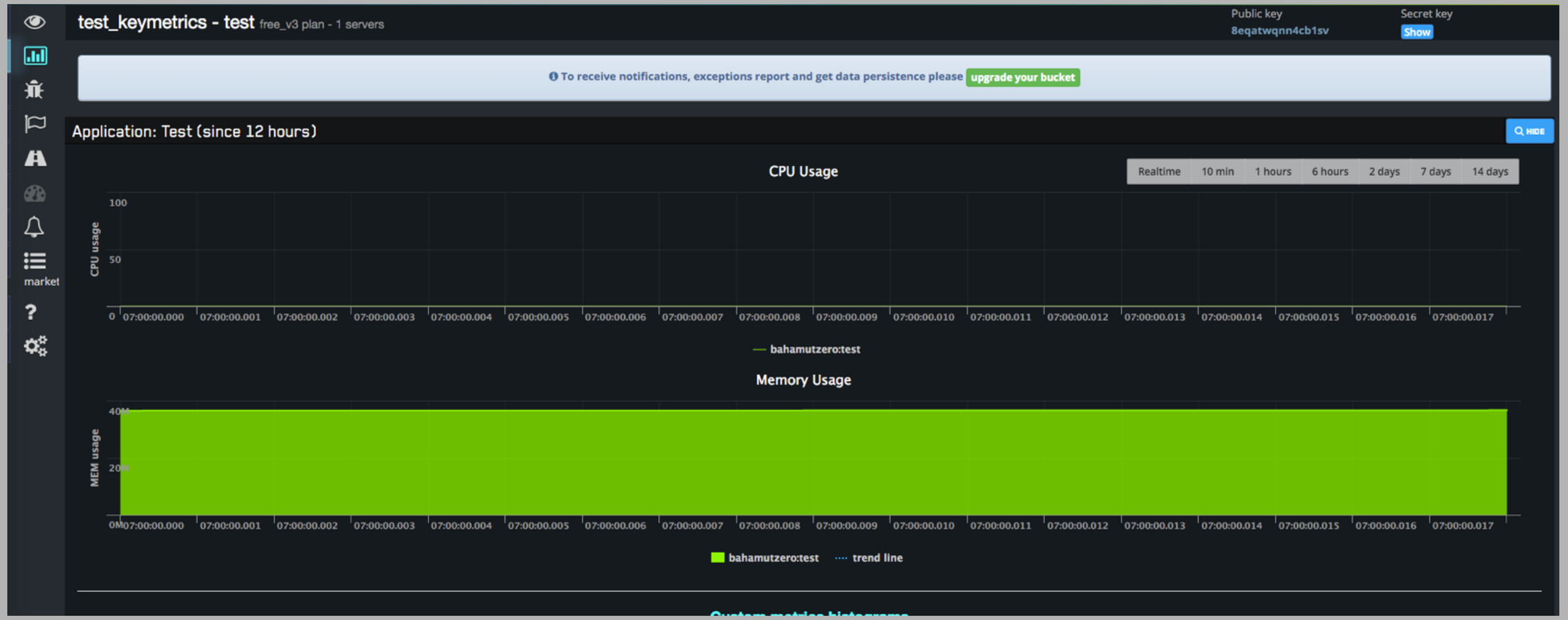

Keymetrics Bucket
test_keymetrics
Comment (about servers you will manage)
Create Bucket
What's a bucket?
A BUCKET is like a container on which multiple servers and multiple apps can be attached to. It's very convenient when you need to monitor a bunch of apps that belongs to the same companies or environment for example.
Each BUCKET has a public and private key. This will allow PM2 to interact with a determined Bucket and most importantly, to communicate in a fully secured manner (with AES256).

KEYMETRICS

$ pm2 link [private] [public]



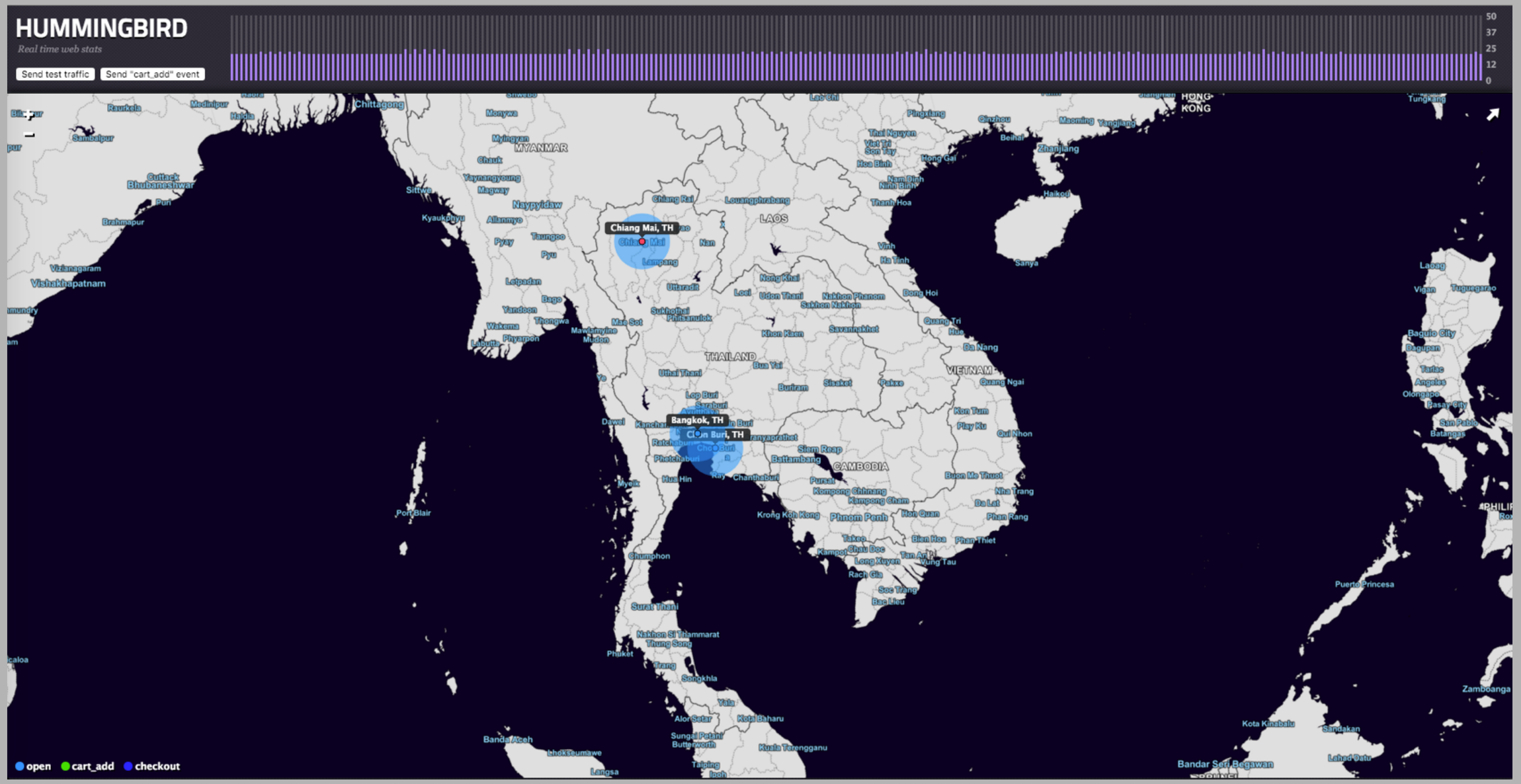
KEYMETRICS


test_keymetrics - test free_v3 plan - 1 servers
Public key
8eqatwqnn4cb1sv
Secret key
Show
To receive notifications, exceptions report and get data persistence please upgrade your bucket
Application: Test (since 12 hours)
HIDE
CPU Usage
Realtime
10 min
1 hours
6 hours
2 days
7 days
14 days
CPU usage
100
50
0
07:00:00.000
07:00:00.001
07:00:00.002
07:00:00.003
07:00:00.004
07:00:00.005
07:00:00.006
07:00:00.007
07:00:00.008
07:00:00.009
07:00:00.010
07:00:00.011
07:00:00.012
07:00:00.013
07:00:00.014
07:00:00.015
07:00:00.016
07:00:00.017
bahamutzero:test
Memory Usage
MEM usage
40M
20M
0M
bahamutzero:test
trend line
market

REAL-TIME APPLICATION ANALYTICS SAMPLE CASE

https://github.com/mnutt/hummingbird

HUMMINGBIRD



HUMMINGBIRD CLIENT

```
var Buffer = require('buffer').Buffer;
var dgram = require('dgram');

var sock = dgram.createSocket("udp4");

var data = {
  "ip" : "127.0.0.1",
  "timestamp" : "Sat Oct 23 2010 21:39:35 GMT+0700 (EDT)",
  "url_key" : 123,
  "product_id" : 456
};

var buf = new Buffer(JSON.stringify(data));

console.log('buf:' + buf);

setInterval(function() {
        sock.send(buf, 0, buf.length, 8000, "0.0.0.0");
},100);
```

SECURITY DISCUSSION

CONFIGURATION : SECURITY HTTP HEADER

- Strict-Transport-Security enforces secure (HTTP over SSL/TLS) connections to the server
- X-Frame-Options provides clickjacking protection
- X-XSS-Protection enables the Cross-site scripting (XSS) filter built into most recent web browsers
- X-Content-Type-Options prevents browsers from MIME-sniffing a response away from the declared content-type
- Content-Security-Policy prevents a wide range of attacks, including Cross-site scripting and other cross-site injections

```
# nginx.conf

add_header X-Frame-Options SAMEORIGIN;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
add_header Content-Security-Policy "default-src 'self'";
```

AUTHENTICATION : BRUTE FORCE

```
$ npm install ratelimiter

#app.js
var email = req.body.email;
var limit = new Limiter({ id: email, db: db });

limit.get(function(err, limit) {

});
```

RATE LIMIT

SESSION MANAGEMENT

Cookie Flags

secure - this attribute tells the browser to only send the cookie if the request is being sent over HTTPS.

HttpOnly - this attribute is used to help prevent attacks such as cross-site scripting, since it does not allow the cookie to be accessed via JavaScript.

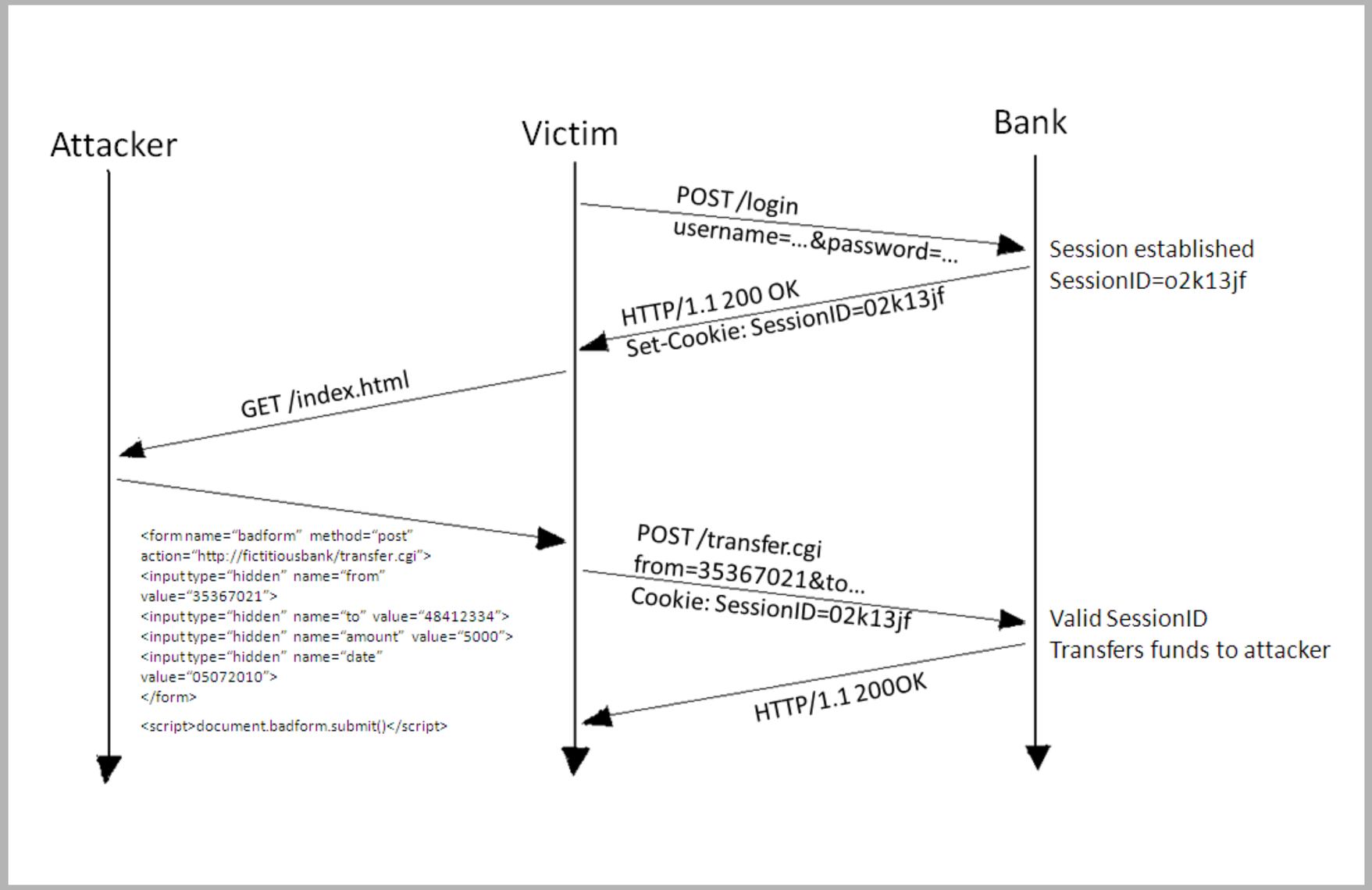
Cookie Scope

domain - this attribute is used to compare against the domain of the server in which the URL is being requested. If the domain matches or if it is a sub-domain, then the path attribute will be checked next.

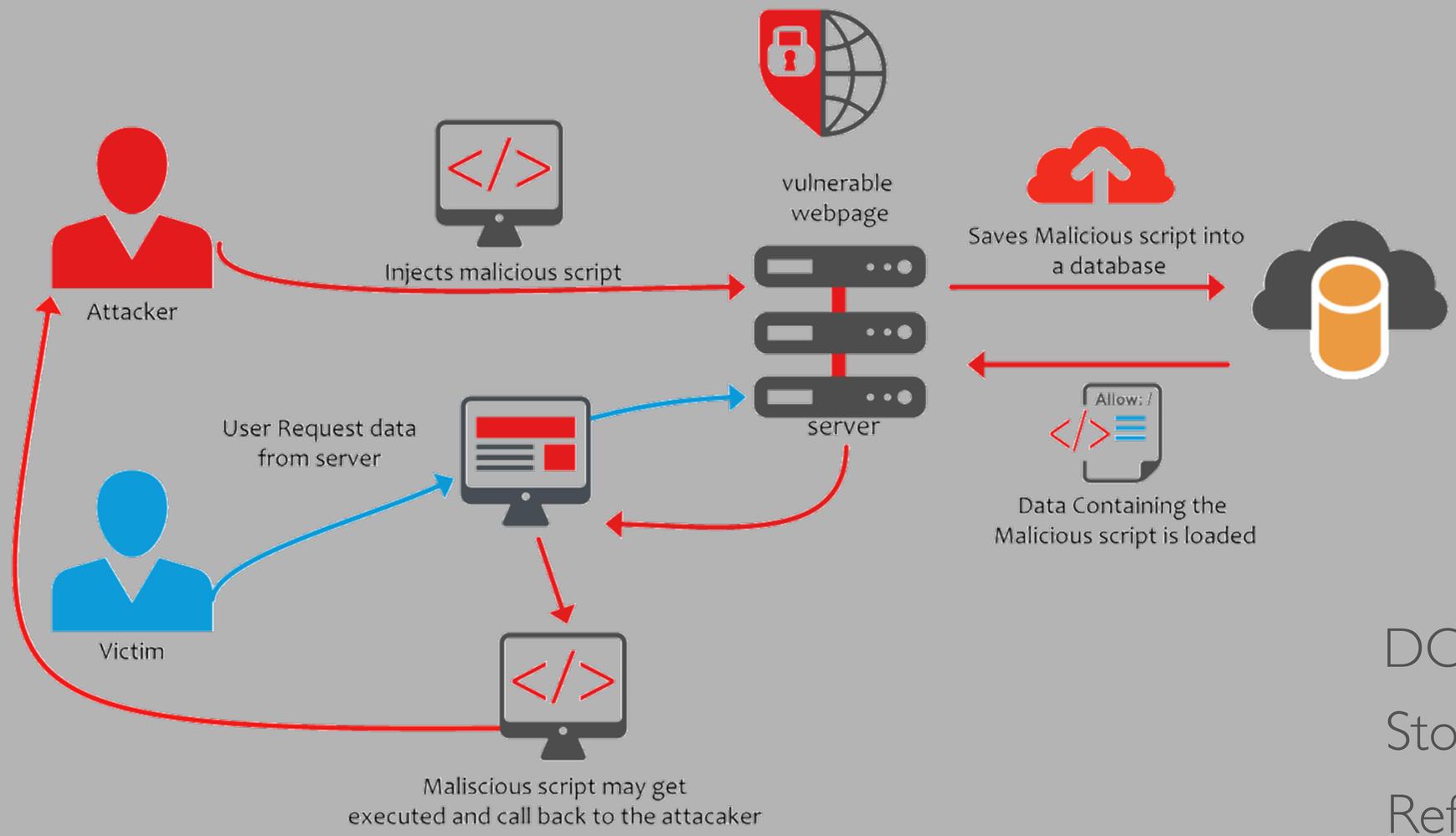
path - in addition to the domain, the URL path that the cookie is valid for can be specified. If the domain and path match, then the cookie will be sent in the request.

expires - this attribute is used to set persistent cookies, since the cookie does not expire until the set date is exceeded

CSRF: CROSS SITE REQUEST FORGERY



XSS : CROSS SITE SCRIPTING

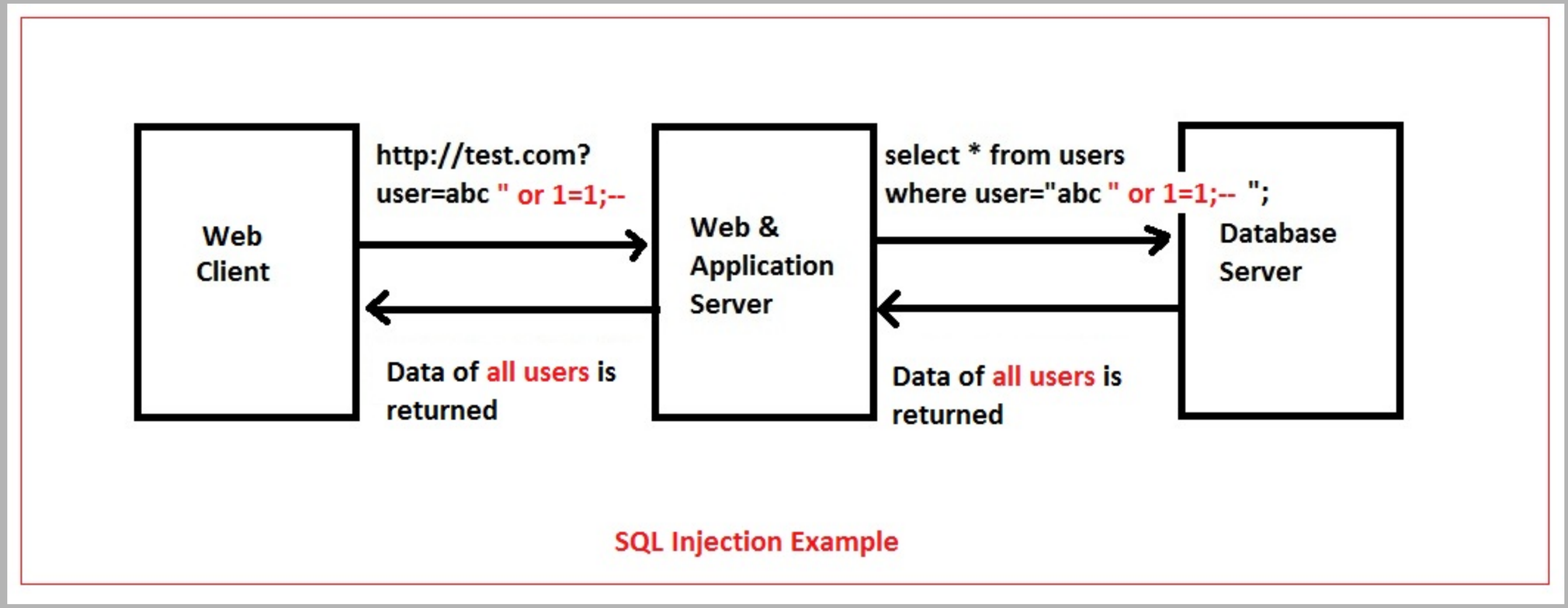


DOM Based XSS

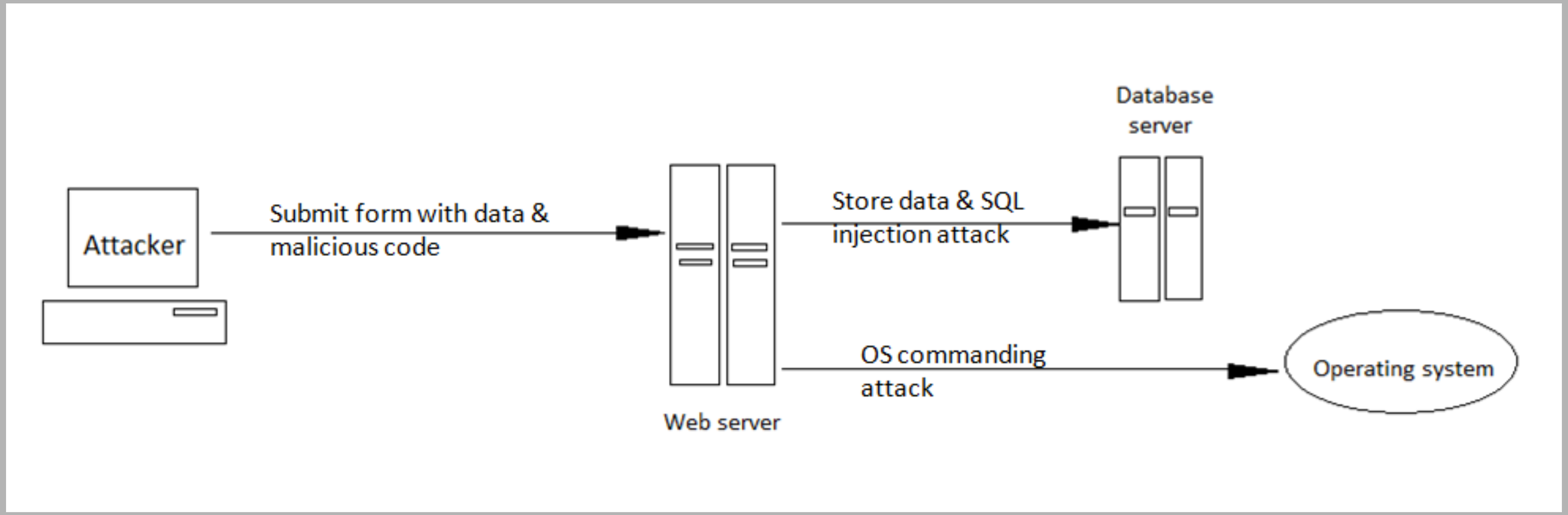
Stored XSS

Reflected XSS

SQL INJECTION



COMMAND INJECTION



Original URL: https://example.com/downloads?file=user1.txt

Attacking URL: https://example.com/downloads?file=%3Bcat%20/etc/passwd

Q & A