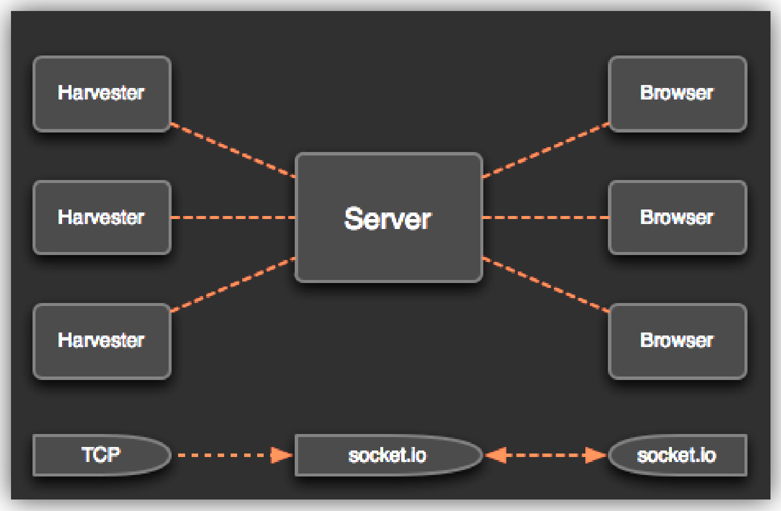
Configuring Centralize Log Monitoring Complete Solution



Harvesters watch log files for changes; send new log messages to the server, which broadcasts to web clients. Log messages are tagged with stream, node, and log level information based on user configuration.

Log.io has no persistence layer. Harvesters are informed of file changes via inotify, and log messages hop from harvester to server to web client via TCP and socket.io, respectively.

Setting Repos info

Date:

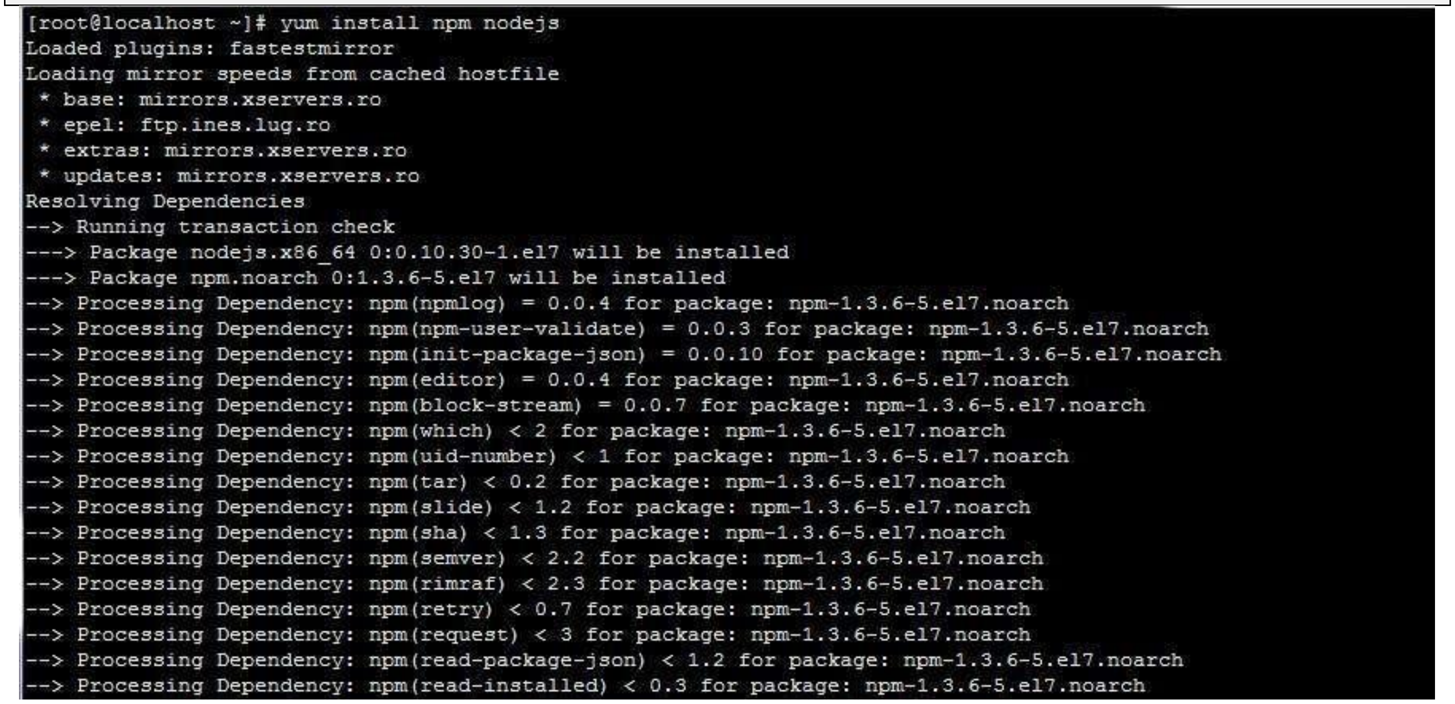
```
date -s "9 AUG 201% 11:32:08"

## RHEL/CentOS 6 64-Bit ##
yum install -y http://mirror.nus.edu.sg/fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Installing Node.js and NPM Binary Packages

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. **Node.js** uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

NPM is a package manager for JavaScript, and is the default for Node.js. As of Node.js version 0.6.3, **npm** is bundled and installed automatically with the environment. **npm** runs through the command line and manages dependencies for an application.

```
yum install npm* nodejs*
```



```
[root@localhost ~]# yum install npm nodejs
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.xservers.ro
 * epel: ftp.ines.lug.ro
 * extras: mirrors.xservers.ro
 * updates: mirrors.xservers.ro
Resolving Dependencies
--> Running transaction check
---> Package nodejs.x86_64 0:0.10.30-1.el7 will be installed
---> Package npm.noarch 0:1.3.6-5.el7 will be installed
--> Processing Dependency: npm(npmlog) = 0.0.4 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(npm-user-validate) = 0.0.3 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(init-package-json) = 0.0.10 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(editor) = 0.0.4 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(block-stream) = 0.0.7 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(which) < 2 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(uid-number) < 1 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(tar) < 0.2 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(slide) < 1.2 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(sha) < 1.3 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(semver) < 2.2 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(rimraf) < 2.3 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(retry) < 0.7 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(request) < 3 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(read-package-json) < 1.2 for package: npm-1.3.6-5.el7.noarch
--> Processing Dependency: npm(read-installed) < 0.3 for package: npm-1.3.6-5.el7.noarch
```

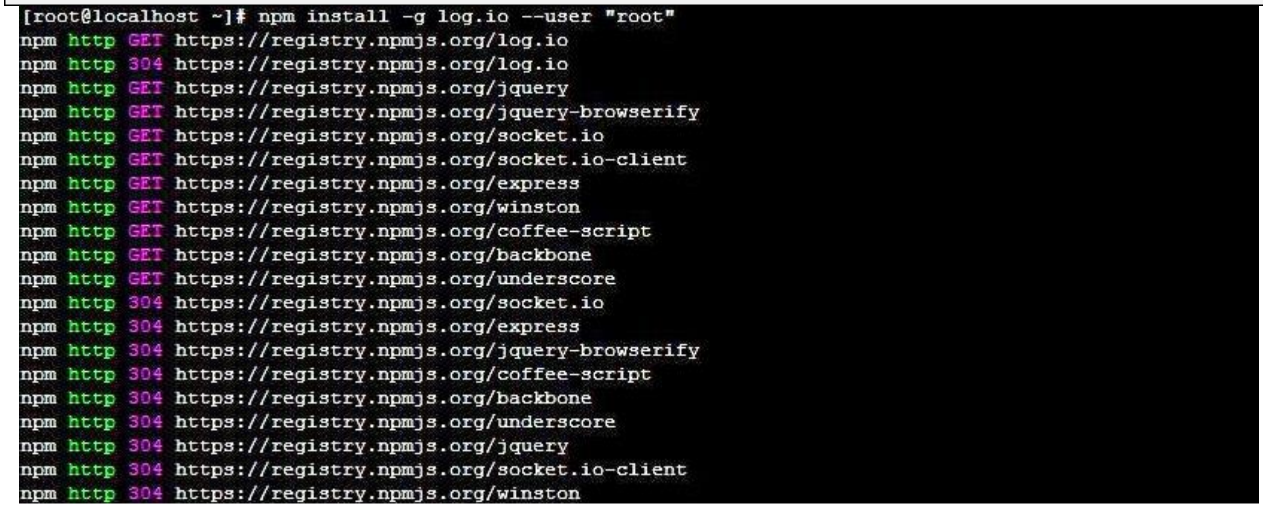
Configure Log.io Application

Log.io application must be installed on your system through NPM by specifying a valid local system user, through which the installation must take place. While you can use any valid system user to install Log.io, I personally recommend installing the application through **root** user or other system user with root privileges.

The reason for using this approach is that **Log.io** must have access to read locally log files and a user with non-privileges root privileges usually can't access and read some important log files.

So, login with root account and install Log.io application through root account by issuing the following command (if you use other user replace root account with your system user accordingly).

```
npm install -g log.io --user "root"
```



```
[root@localhost ~]# npm install -g log.io --user "root"
npm http GET https://registry.npmjs.org/log.io
npm http 304 https://registry.npmjs.org/log.io
npm http GET https://registry.npmjs.org/jquery
npm http GET https://registry.npmjs.org/jquery-browserify
npm http GET https://registry.npmjs.org/socket.io
npm http GET https://registry.npmjs.org/socket.io-client
npm http GET https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/winston
npm http GET https://registry.npmjs.org/coffee-script
npm http GET https://registry.npmjs.org/backbone
npm http GET https://registry.npmjs.org/underscore
npm http 304 https://registry.npmjs.org/socket.io
npm http 304 https://registry.npmjs.org/express
npm http 304 https://registry.npmjs.org/jquery-browserify
npm http 304 https://registry.npmjs.org/coffee-script
npm http 304 https://registry.npmjs.org/backbone
npm http 304 https://registry.npmjs.org/underscore
npm http 304 https://registry.npmjs.org/jquery
npm http 304 https://registry.npmjs.org/socket.io-client
npm http 304 https://registry.npmjs.org/winston
```

After the application has been installed change your working directory to **Log.io** folder, which is hidden, and do a directory listing to visualize folder content in order to configure the application further.

```
cd .log.io/
```

Now it’s time to configure **Log.io** to monitor local log files in real time. Let’s get an inside on how Log.io works.

1. The **harvester** file watches for changes in the specified local log files declared in its configuration and sends its output via socket.io TCP
   protocol which further send the messages to Log.io local server or any other remote server declared with its IP Address (0.0.0.0 address specified on harvesters broadcasts to all log.io listening servers) – file **harvester.conf**
2. Log.io server binds on all network interfaces (if not specified otherwise in log_server.conf file) and waits for messages from locally or remote harvesters nodes and sends their output to log.io Web server (0.0.0.0 means that it waits for messages from any local or remote harvesters) file **log_server.conf**

3. Log.io Web server binds on all network interfaces, listens for web clients connections on port 28778 and processes and outputs the messages that it receives internally from log.io server – file **web_server.conf**

First open **harvester.conf** file for editing, which by default only monitors Apache log files, and replace **nodeName** statement to match your hostname and define the **logStreams** statements with what internal log files you want to monitor (in this case I'm monitoring multiple log files such as audit, messages and secure logs). Use the below file excerpt as a guide.

ON THE CLIENT SIDE TO BE MONITORING

```
vim harvester.conf
exports.config = {

  nodeName: "Client Server 1",

  logStreams: {

audit: [

     "/var/log/audit/audit.log"

    ],

messages: [

     "/var/log/messages"

    ],

secure: [

     "/var/log/secure"

    ]

},

  server: {

    host: '172.17.100.201',
   port: 28777

  }

}
```

Also if you don’t need harvester output to be sent to a remote **Log.io** server change the line **host** on **server** statement to only send its output locally by modifying 0.0.0.0 address with loopback address (**127.0.0.1)**.

ON THE MAIN MONITORING SERVER

```
vim log_server.conf
exports.config = {
  host: '0.0.0.0',
  port: 28777
```

Other security features such as credentials login, HTTPS or restriction based on IPs to Log.io web server can be applied on web server-side. For this tutorial I will only use as a security measure just credential login.

So, open **web_server.conf** file, uncomment the entire **auth** statement by deleting all slashes and asterisks and replace **user** and **pass** directives accordingly as suggested on the bottom screenshot.

```
vim web_server.conf
exports.config = {

  host: '172.17.100.201',

  port: 28778,

   auth: {

    user: "admin",

    pass: "1234"

  },

  /*

  // Restrict access to http server (express)

  restrictHTTP: [
   "192.168.29.39",

   "10.0.*"

  ]

  */

}
```

Log.io Manage Script

In order to use a command that manages **Log.io** application with three switches (start, **stop** and **status**) create the following script named **log.io** on **/usr/local/bin** executable directory and append execution permissions to this script.

```
touch /usr/local/bin/log.io
chmod +x /usr/local/bin/log.io
vim /usr/local/bin/log.io
```

AT THE MONITORING SERVER

```
#!/bin/bash

          start() {

          echo "Starting log.io process..."

           nohup /usr/bin/log.io-server > /dev/null 2>&1 &

                                          }

          stop() {

          echo "Stopping io-log process..."

          pkill node

                              }

          status() {

          echo "Status io-log process..."

          netstat -tlp | grep node
```

```
                                }

case "$1" in

            start)

start

      ;;

            stop)

stop

      ;;

            status)

status

            ;;

            *)

echo "Usage: start|stop|status"

      ;;

esac
```

AT THE CLIENT SERVER TO BE MONITORING

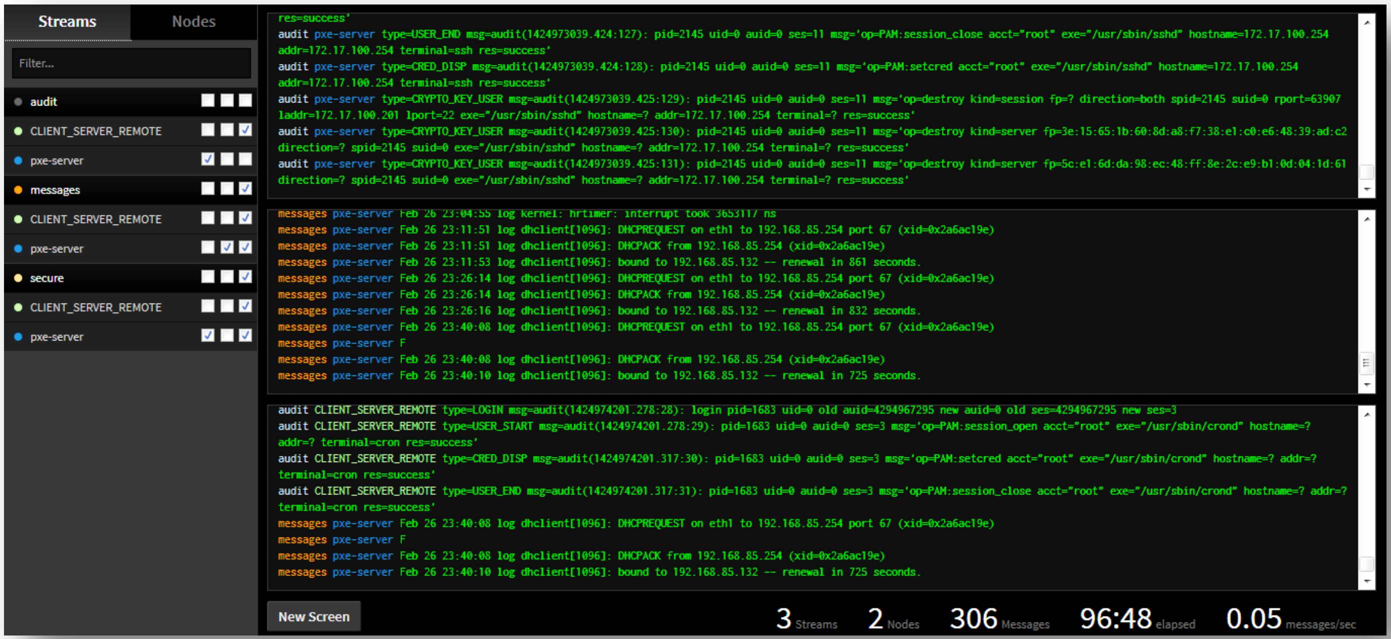
```
#!/bin/bash

            start() {

            echo "Starting log.io process..."

            nohup /usr/bin/log.io-harvester > /dev/null 2>&1 &

                                                    }

            stop() {

            echo "Stopping io-log process..."
            pkill node

                                }

            status() {

            echo "Status io-log process..."

            netstat -tlp | grep node

                                }

case "$1" in

            start)

start

      ;;

            stop)

stop

      ;;

            status)

status

            ;;

            *)

echo "Usage: start|stop|status"

      ;;

esac
```

To start, stop or view Log.io status login with root account (or the user that Log.io app has been installed) and just run the following commands to easily manage the application.

After the server has been started open a browser, enter your servers IP followed by **28778** port number using HTTP protocol on URL address and a prompt demanding your login credentials should appear.

Enter your user and password configured on **step 8** to proceed further and **Log.io** application should now be visible on your browser presenting monitored log files in real time.

Just visit http://172.17.100.201:28778/



```
res=success'
audit pxe-server type=USER_END msg=audit(1424973039.424:127): pid=2145 uid=0 auid=0 ses=11 msg='op=PAM:session_close acct="root" exe="/usr/sbin/sshd" hostname=172.17.100.254
addr=172.17.100.254 terminal=ssh res=success'
audit pxe-server type=CRED_DISP msg=audit(1424973039.424:128): pid=2145 uid=0 auid=0 ses=11 msg='op=PAM:setcred acct="root" exe="/usr/sbin/sshd" hostname=172.17.100.254
addr=172.17.100.254 terminal=ssh res=success'
audit pxe-server type=CRYPTO_KEY_USER msg=audit(1424973039.425:129): pid=2145 uid=0 auid=0 ses=11 msg='op=destroy kind=session fp=? direction=both spid=2145 suid=0 rport=63907
laddr=172.17.100.201 lport=22 exe="/usr/sbin/sshd" hostname=? addr=172.17.100.254 terminal=? res=success'
audit pxe-server type=CRYPTO_KEY_USER msg=audit(1424973039.425:130): pid=2145 uid=0 auid=0 ses=11 msg='op=destroy kind=server fp=3e:15:65:1b:60:8d:a8:f7:38:e1:c0:e6:48:39:ad:c2
direction=? spid=2145 suid=0 exe="/usr/sbin/sshd" hostname=? addr=172.17.100.254 terminal=? res=success'
audit pxe-server type=CRYPTO_KEY_USER msg=audit(1424973039.425:131): pid=2145 uid=0 auid=0 ses=11 msg='op=destroy kind=server fp=5c:e1:6d:da:98:ec:48:ff:8e:2c:e9:b1:0d:04:1d:61
direction=? spid=2145 suid=0 exe="/usr/sbin/sshd" hostname=? addr=172.17.100.254 terminal=? res=success'
```

```
messages pxe-server Feb 26 23:04:55 log kernel: hrtimer: interrupt took 3653117 ns
messages pxe-server Feb 26 23:11:51 log dhclient[1096]: DHCPREQUEST on eth1 to 192.168.85.254 port 67 (xid=0x2a6ac19e)
messages pxe-server Feb 26 23:11:51 log dhclient[1096]: DHCPACK from 192.168.85.254 (xid=0x2a6ac19e)
messages pxe-server Feb 26 23:11:53 log dhclient[1096]: bound to 192.168.85.132 -- renewal in 861 seconds.
messages pxe-server Feb 26 23:26:14 log dhclient[1096]: DHCPREQUEST on eth1 to 192.168.85.254 port 67 (xid=0x2a6ac19e)
messages pxe-server Feb 26 23:26:14 log dhclient[1096]: DHCPACK from 192.168.85.254 (xid=0x2a6ac19e)
messages pxe-server Feb 26 23:26:16 log dhclient[1096]: bound to 192.168.85.132 -- renewal in 832 seconds.
messages pxe-server Feb 26 23:40:08 log dhclient[1096]: DHCPREQUEST on eth1 to 192.168.85.254 port 67 (xid=0x2a6ac19e)
messages pxe-server F
messages pxe-server Feb 26 23:40:08 log dhclient[1096]: DHCPACK from 192.168.85.254 (xid=0x2a6ac19e)
messages pxe-server Feb 26 23:40:10 log dhclient[1096]: bound to 192.168.85.132 -- renewal in 725 seconds.
```

```
audit CLIENT_SERVER_REMOTE type=LOGIN msg=audit(1424974201.278:28): login pid=1683 uid=0 old auid=4294967295 new auid=0 old ses=4294967295 new ses=3
audit CLIENT_SERVER_REMOTE type=USER_START msg=audit(1424974201.278:29): pid=1683 uid=0 auid=0 ses=3 msg='op=PAM:session_open acct="root" exe="/usr/sbin/crond" hostname=?
addr=? terminal=cron res=success'
audit CLIENT_SERVER_REMOTE type=CRED_DISP msg=audit(1424974201.317:30): pid=1683 uid=0 auid=0 ses=3 msg='op=PAM:setcred acct="root" exe="/usr/sbin/crond" hostname=? addr=?
terminal=cron res=success'
audit CLIENT_SERVER_REMOTE type=USER_END msg=audit(1424974201.317:31): pid=1683 uid=0 auid=0 ses=3 msg='op=PAM:session_close acct="root" exe="/usr/sbin/crond" hostname=? addr=?
terminal=cron res=success'
messages pxe-server Feb 26 23:40:08 log dhclient[1096]: DHCPREQUEST on eth1 to 192.168.85.254 port 67 (xid=0x2a6ac19e)
messages pxe-server F
messages pxe-server Feb 26 23:40:08 log dhclient[1096]: DHCPACK from 192.168.85.254 (xid=0x2a6ac19e)
messages pxe-server Feb 26 23:40:10 log dhclient[1096]: bound to 192.168.85.132 -- renewal in 725 seconds.
```

ENJOY..!