Bunyan Logging in Production at Joyent

Trent Mick (@trentmick)
Joyent

Vancouver Node.js Meetup, 7 Aug 2014

Joyent

Bunyan Logging ...

- Blow through usage and some features.

... in production at Joyent

- Show a few (hopefully) interesting examples from production experiences.
- Discuss some motivations.

Bunyan is

① A node.js lib for structured service (and CLI) logging

```
$ npm install bunyan
$ cat foo.js

var bunyan = require('bunyan');
var log = bunyan.createLogger({
    name: 'foo',
    // ...
});

log.info('hi');

var err = new Error('ack');
log.error(err, 'boom!');
```

Bunyan is

② A (lightweight) JSON log format

- one line of JSON per record, small set of minimal fields:

```
{"name":"foo","hostname":"grape.local","pid":68313,"level":30,
    "msg":"hi","time":"2014-08-07T06:18:53.057Z","v":0}
```

- The primary audience of logs is machine processing.

Bunyan is

③ A `bunyan` CLI for viewing and filtering logs

Reading JSON sucks, but ANSI codes in files + custom parsing code suck more:

```
$ node foo.js | bunyan
[2014-08-07T06:35:48.340Z]  INFO: foo/68813 on grape.local: hi
[2014-08-07T06:35:48.343Z] ERROR: foo/68813 on grape.local: boom!
    Error: ack
        at Object.<anonymous> (/Users/trentm/tmp/play/foo.js:9:11)
        at Module._compile (module.js:456:26)
        at Object.Module._extensions..js (module.js:474:10)
```

```
bunyan foo.log [bar.log ...]
bunyan -l warn foo.log
bunyan -c 'this.user=="bob"' foo.log
```

Bunyan features

- standard *levels*: trace, debug, info, warn, error, fatal
- extensible *streams* for directing logs to stderr, stdout, files, rotating files, custom endpoints
- specialization of Loggers with `log.child`
- custom rendering of logged objects with *serializers*
- runtime log snooping via DTrace support

See README for more.

SmartDataCenter 7

- ~40 service/agent/tool log types (~90% bunyan)
- 1-5 service instances, one agent instance per server
- ~12 DCs (counting test and staging DCs)

Motivation: Log more details

- Make it convenient in code to log more and appropriate details.
- Painful, limited (traditional):

```
log.info('%s %s %s %s', req.method, req.url,
    res.statusCode, res.latency)
```

- Easier to code (bunyan):

```
log.info({req: req}, 'received request');
log.debug({res: res}, 'responded');
log.error({err: err, path: path}, 'save failed');
```

Log more details: render nicely

Make having the extra details not a burden when browsing logs by rendering them nicely.

HTTP request

```
[2014-08-07T17:08:24.071Z]  INFO: imgapi/74427 on e540e25b-6afc-41ee-a940-a1da39628676: handled: 200
    GET /images/2b683a82-a066-11e3-97ab-2faa44701c5a HTTP/1.1
    accept: application/json
    user-agent: restify/2.8.1 (ia32-sunos; v8/3.14.5.9; OpenSSL/1.0.1h) node/0.10.29
    accept-version: *
    date: Thu, 07 Aug 2014 17:08:22 GMT
    host: imgapi.nightly-1.joyent.us
    connection: close
```

HTTP response

```
[2014-08-07T17:12:01.601Z]  INFO: cnapi/12183 on 7b44b026-e23d-4117-9093-9c37da632501: handled: 500
    ...
    --
    HTTP/1.1 500 Internal Server Error
    content-type: application/json
    content-length: 77
    date: Thu, 07 Aug 2014 17:12:01 GMT
    server: Compute Node API
    x-request-id: f2692cf0-1e55-11e4-a698-b90f8ef63a7a
    x-response-time: 1
    x-server-name: 7b44b026-e23d-4117-9093-9c37da632501
```

HTTP response + internal error details

```
[2014-08-07T17:12:01.601Z]  INFO: cnapi/12183 on 7b44b026-e23d-4117-9093-9c37da632501: handled: 500
    ...
    --
    HTTP/1.1 500 Internal Server Error
    content-type: application/json
    content-length: 77
    date: Thu, 07 Aug 2014 17:12:01 GMT
    server: Compute Node API
    x-request-id: f2692cf0-1e55-11e4-a698-b90f8ef63a7a
    x-response-time: 1
    x-server-name: 7b44b026-e23d-4117-9093-9c37da632501
    --
    InternalError: Precondition failed: no connection moray
        at Object.ensureConnectedToMoray [as moray] (/opt/smartdc/cnapi/lib/endpoints/index.js:38:14)
        at Server.<anonymous> (/opt/smartdc/cnapi/lib/endpoints/index.js:163:39)
        at next (/opt/smartdc/cnapi/node_modules/restify/lib/server.js:736:42)
        at f (/opt/smartdc/cnapi/node_modules/once/once.js:16:25)
        at ensureConnectionTimeout (/opt/smartdc/cnapi/lib/endpoints/index.js:65:5)
        at Server.<anonymous> (/opt/smartdc/cnapi/lib/endpoints/index.js:149:9)
        at next (/opt/smartdc/cnapi/node_modules/restify/lib/server.js:736:42)
        at f (/opt/smartdc/cnapi/node_modules/once/once.js:16:25)
        at Server.<anonymous> (/opt/smartdc/cnapi/lib/endpoints/index.js:145:9)
```

Error + underlying error cause

```
[2014-08-07T17:15:28.452Z]  INFO: imgapi/74427 on e540e25b-6afc-41ee-a940-a1da39628676: handled: 404
    ...
    --
    ResourceNotFoundError: image not found; caused by ObjectNotFoundError: imgapi_images::f3785e6b-cb9b-
        at /opt/smartdc/imgapi/lib/database.js:546:26
        at f (/opt/smartdc/imgapi/node_modules/once/once.js:16:25)
        at EventEmitter.<anonymous> (/opt/smartdc/imgapi/node_modules/moray/lib/objects.js:137:9)
        at EventEmitter.g (events.js:180:16)
        at EventEmitter.emit (events.js:95:17)
        at Client._handleMessage (/opt/smartdc/imgapi/node_modules/moray/node_modules/fast/lib/client.js
        at MessageDecoder.onMessage (/opt/smartdc/imgapi/node_modules/moray/node_modules/fast/lib/client
        at MessageDecoder.EventEmitter.emit (events.js:95:17)
        at MessageDecoder._write (/opt/smartdc/imgapi/node_modules/moray/node_modules/fast/lib/protocol/
        at doWrite (/opt/smartdc/imgapi/node_modules/moray/node_modules/fast/node_modules/readable-strea
    Caused by: ObjectNotFoundError: imgapi_images::f3785e6b-cb9b-e31b-8ac5-852e57d2662a does not exist
        at EventEmitter.<anonymous> (/opt/smartdc/moray/lib/objects/get.js:102:16)
        at EventEmitter.g (events.js:180:16)
        at EventEmitter.emit (events.js:95:17)
        at endOrError (/opt/smartdc/moray/lib/pg.js:89:13)
        at f (/opt/smartdc/moray/node_modules/once/once.js:16:25)
        at onQueryEnd (/opt/smartdc/moray/lib/pg.js:116:9)
        at EventEmitter.emit (events.js:95:17)
        at Query.handleReadyForQuery (/opt/smartdc/moray/node_modules/pg/lib/query.js:86:8)
        at null.<anonymous> (/opt/smartdc/moray/node_modules/pg/lib/client.js:159:19)
        at EventEmitter.emit (events.js:117:20)
        at null.<anonymous> (/opt/smartdc/moray/node_modules/pg/lib/connection.js:97:12)
```

Maintain error cause chains with node-verror. Bunyan will render these.

Log more details: filtering

Motivate logging of more structured details by making filtering convenient and compelling.

```
# ERROR level and above
$ bunyan foo.log -l error

# Watch incoming requests.
$ tail -f foo.log | bunyan -c 'this.req'

# Server errors in HTTP responses.
$ tail -f foo.log | bunyan -c 'this.res && this.res.statusCode >= 500'
```

Log more details: serializers

- Problem: Dumping full objects -> extraneous goop in the logs.
- Define how an obj is logged with serializer functions for well-defined log *field names*.
- Bunyan defines for `err`, `req`, and `res`. Use is explicit, e.g.:

```
var log = bunyan.createLogger({
    //...
    serializers: {
        err: bunyan.stdSerializers.err,
        // custom
        user: function (u) { if (u) return {login: u.login, name: u.name}; }
    }
});

log.info({user: theUser}, 'logged in');
log.error({err: e, user: theUser}, 'logged failed');
```

```
[2014-08-07T17:25:55.126Z]  INFO: imgapi/74427 on e540e25b-6afc-41ee-a940-a1da396
    POST /images?action=create-from-vm&vm_uuid=c691c60d-8f65-6eb2-81d6-d163d65ea4
    x-request-id: dd8d4ad0-1e57-11e4-8499-df7725ba1cdc
    accept: application/json
    content-type: application/json
    user-agent: restify/2.6.1 (ia32-sunos; v8/3.14.5.9; OpenSSL/1.0.1h) node/0.10
    date: Thu, 07 Aug 2014 17:25:48 GMT
    content-length: 65
    content-md5: d02bUF89wkeSJT+jGTEENg==
    host: imgapi.nightly-1.joyent.us
    connection: keep-alive

    {
      "name": "1cfeb76f-d951-c1e3-b9e7-97825c7c4efc",
      "version": "1.0.0",
      "owner": "551a64ca-3e08-c0de-cfd2-da81641890ec",
      "uuid": "1c7a2b2d-f5da-eed2-d37c-e3a01de9a8ee",
      "requirements": {
        "min_platform": {
          "7.0": "20140807T022845Z"
        }
      }
    }
    --
    HTTP/1.1 200 OK
```

Necessity: balancing volume

- Serializers help limit the size of a logged object.
- Obviously, set runtime log level to control log volume.
- Not enough.
- Two interesting examples: RequestCaptureStream and DTrace.

Balancing volume: RequestCaptureStream

- Provided by restify web service library.
- Store low level log records in in-memory ring buffer, *per request id*.
- If there is a `log.warn` or above, then all low level records for *for that request id* are logged.
- Used in Joyent's public web APIs.

Code for a RequestCaptureStream

```
var log = bunyan.createLogger({
    name: 'cloudapi',
    serializers: restify.bunyan.serializers,
    streams: [
        {
            level: 'info',
            stream: process.stderr
        },
        {
            type: 'raw',
            stream: new RequestCaptureStream({
                level: bunyan.WARN,
                maxRecords: 100,
                maxRequestIds: 100,
                streams: [ {
                    stream: process.stderr
                } ]
            })
        }
    ]
});
```

Balancing volume: DTrace

- Typically log level is `INFO` or `DEBUG` at best.
- How to get `TRACE` logging details?
- Option 1: Update app config and restart.
- Option 2: DTrace `bunyan` provider. No restart.

```
$ bunyan -p $PID
$ bunyan -p $NAME    # search processes by name via `pgrep` or `ps`
$ bunyan -p '*'      # instrument *all* processes!
$ bunyan -p '*' -l DEBUG
$ bunyan -p '*' -c 'this.name === "cloudapi"'
```

`bunyan -p`demo

Motivation: Tie things together

- Service logs can be a jumble of simultaneous operations.
- How to meaningfully separate?

Tie things together: module-scope loggers don't help

```
lib/main.js:    var log = log4j.getLogger('foo')
lib/db.js:      var log = log4j.getLogger('foo.db')
lib/model.js:   var log = log4j.getLogger('foo.model')
```

Common with log4j-inspired libraries.
Groups on module, not on related operations.

Tie things together: always log context -> PITA

```
log.info('user=%s start blerging', user)
log.info('user=%s frob the frib', user);
log.info('bliging bloop');    // oops, forgot to log `user`
```

- Easy to forget to include relevant context.
- Labourious to add, so gets skipped.

Tie things together: `log.child`

```
req.log = app.log.child({req_id: req.getId()});
// ...
req.log.info({route: route}, 'routed request');
// ...
req.log.debug(err, 'validation failed');
```

- Group related logging via logger specialization with `log.child`.
- Annoyance: Now need to pass a logger around code.
- Attach to relevant context object, if available.
  E.g. restify creates a `<request>.log` for each HTTP request.

Tie things together: `req_id`

```
$ bunyan foo.log \
    -c 'this.req_id === 'd6740300-1e58-11e4-8f85-2fef7c7c36dd'"
```

Now we can see all logging for a particular request.

Tie things together: `req_id++`

How about tracking a request through multiple services?
Example: provision a VM in SmartDataCenter.

- Cloud API: public facing API
- VM API: coordinates creation of VM
- Server API (CNAPI): handles talking to server agents
- Network API (NAPI): network provisioning
- Firewall API (FWAPI): firewall management for VMs

Tie things together: `req_id++`

Demo tracking VM creation in SmartDataCenter via `req_id`.

Questions?

Trent Mick (@trentmick)

github.com/trentm/node-bunyan
bunyan-logging@googlegroups.com

Slides: trentm.com/talk-bunyan-in-prod/
and on GitHub: github.com/trentm/talk-bunyan-in-prod