# Publishing Loopback Services

Hybrid Integration Enablement

October 2017

IBM

# How do I Debug Loopback

- In Package.json, built in commands exist to start project locally
- Once started, you can hit endpoints locally in browser
  - [Http://localhost:3000/api/items](Http://localhost:3000/api/items)

```
"scripts": {
  "lint": "eslint .",
  "start": "node .",
  "posttest": "npm run lint && nsp check"
},
```

Running this on command line will give you Full Debugging of how the loopback stacks works at every level

* How all the components work at each level

```
DEBUG=* node .
```

```
DEBUG=* npm start
```

```
  loopback:boot:executor Defining middleware phases ["ini
","parse","routes","files","final"] +1ms
  express:router use '/' query +0ms
  express:router:layer new '/' +0ms
  express:router use '/' expressInit +1ms
  express:router:layer new '/' +0ms
  loopback:boot:executor Configuring middleware "/Users/j
reinventory/node_modules/loopback"#favicon +0ms
  loopback:app use initial:before  favicon +1ms
  express:router use [] favicon +0ms
  express:router:layer new [] +1ms
  loopback:boot:executor Configuring middleware "/Users/j
reinventory/node_modules/compression" +0ms
  loopback:app use initial  compression +1ms
  express:router use [] compression +0ms
  express:router:layer new [] +0ms
```

# Filtering Debug Logging

- Once you get to know the global logging for debugging you can filter the output
  - Isolate loopback activity based on the component name with wildcard
    - DEBUG=express:*
    - DEBUG=*rest
    - DEBUG=*mysql

```
Web server listening at: http://0.0.0.0:3000
  loopback:connector:mysql SQL: SELECT `id`,`name`,`description`,`img`,`price` FROM `item` ORDER BY `id`, params: [] +0ms
  loopback:connector:mysql Data:  [ RowDataPacket {
    id: 1,
    name: 'Dayton Meat Chopper',
```

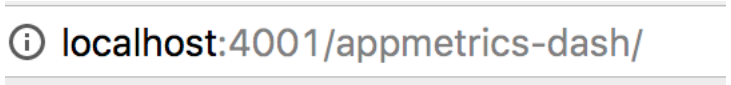
- Add your own quick script to the package.json
  - Modify the “npm start” node .  ( start the project using npm start )
  - Make your own line, but must run it different. ”npm run-script debugme”

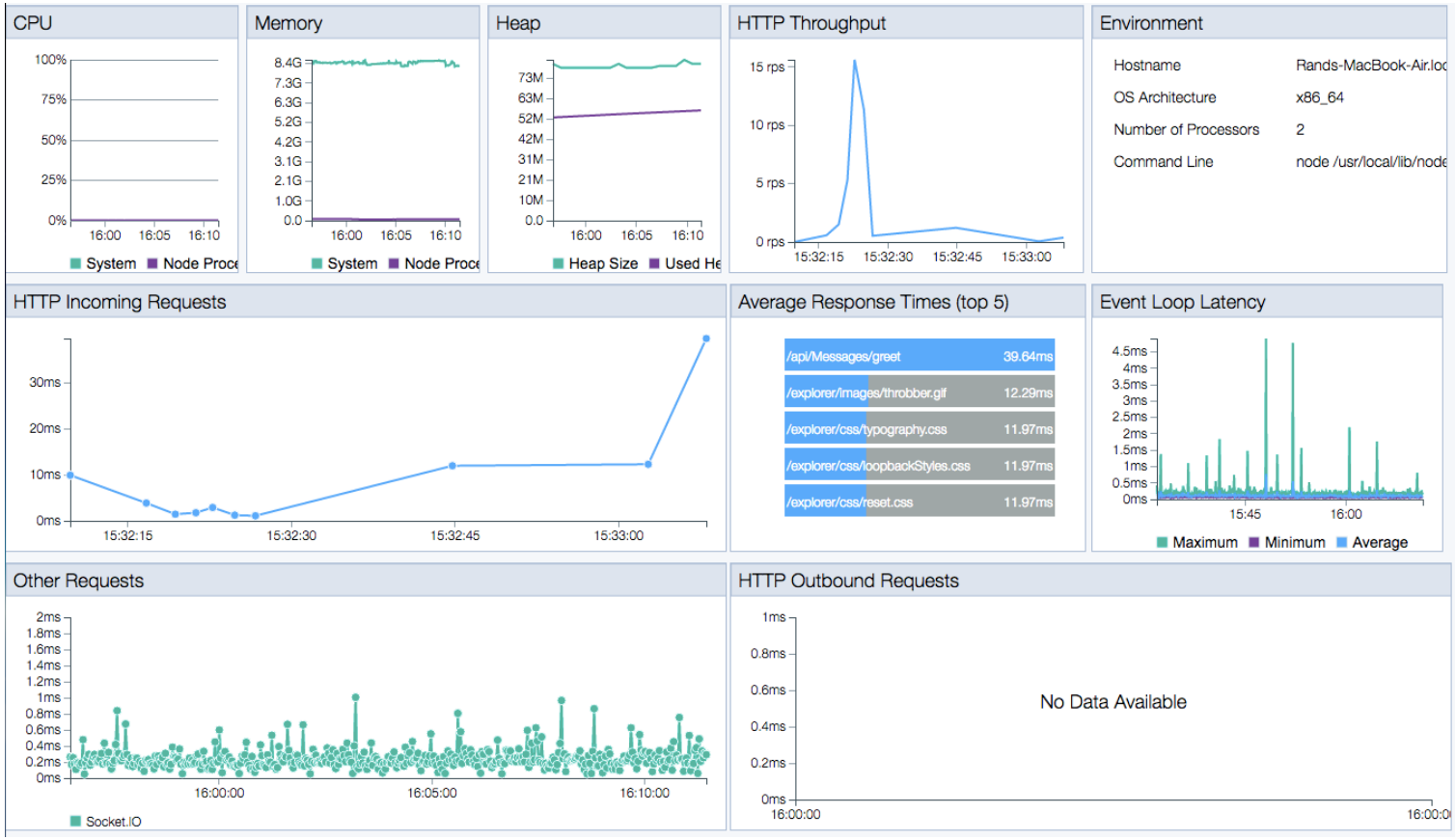
```
"scripts": {
  "lint": "eslint .",
  "start": "node .",
  "debugme": "DEBUG=* node .",
  "posttest": "npm run lint && nsp check"
},
```

# API Designer Application Metrics

- Once your loopback project is running through the API Designer, additional profiling is available.


- This enables application metrics for your node application. Browse there using the Application link with an /appmetrics-dash



# What's included with Metrics Dashboard

- Includes these datapoints via graphs
- All applications created with apic CLI are enabled for this
- View more at documentation link below
- Keep this Dashboard up locally as your API is accessed for testing

- **CPU** - Percentage of CPU time spent on the Node process.
- **Memory** - Amount of memory used by the Node process and available in the system.
- **Heap** - Amount of heap memory used and available.
- **HTTP Throughput** (requests per second) versus time.
- **HTTP Incoming Requests** - Response time for HTTP requests versus time.
- **Average Response Time** for the top five routes.
- **Event Loop Latency** (minimum / maximum / average) - Amount of time spent for a event loop "tick."
- **Other Requests**
- **HTTP Outbound Requests**

- https://www.ibm.com/support/knowledgecenter/SSMNED_5.0.0/com.ibm.apic.toolkit.doc/tapic_view_appmetrics.html

# Ready for Publishing?

- Collectives have been deprecated and end of life announced.
  - Originally the purpose was to give customers an option to publish node runtimes to account for inexperience on the topic

- Following that same pattern, you can still let the API Designer Publish your application
- It will manage …
  - your application runtime via Cloud Foundry
  - Updated your YAML to include proper server properties

| Catalog | Value |
| --- | --- |
| Default | |
| apic-dev | $(TARGET_URL) |
| sb | https://apiconnect-00050c71-9964-4a06-bc24-1a0799ecc979.jhsp |

# Publish the loopback project directly to CF

- You can publish the application to Cloud Foundry yourself as well and update the server properties.

- Add a manifest.yml file to your project root folder and run a ‘bx cf push’ and push your application

```
applications:
- name: bluemix-todo-python-mongo
memory: 256M
```

View more information here about all the file options available

https://docs.cloudfoundry.org/devguide/deploy-apps/manifest.html

# Extend your Cloud Foundry Deployment

- Explore better toolchaining with our Bluemix DevOps toolchain service
- It will allow the following workflow


- Get more information about the service here:
- https://console.bluemix.net/catalog/services/continuous-delivery

# Publishing your Loopback Questions

- Open Conversation / Questions?

Up Next: Secure Gateway

# Lab 5