

December 11-12, 2019
Montreal, Canada

Node.js in a Kubernetes World

Michael Dawson, Community Lead for Node.js, IBM

About: Michael Dawson

IBM Community Lead for Node.js

- Active Node.js community member
 - Collaborator
 - Node.js Technical Steering Committee TSC Chair
 - Community Committee member
 - Node.js OpenJS Foundation Board Member
 - Working group(s) member/leadership



- Twitter: @mhdawson1
 - GitHub: @mhdawson
 - LinkedIn: <https://www.linkedin.com/in/michael-dawson-6051282>



Agenda

- IBM's JavaScript/Node.js Focus
- Cloud Native Development
- Key Cloud Native Development Tasks for the Node.js Developer
- Separating Concerns
- Available Tools
- Q/A

IBM's JavaScript/Node.js Focus

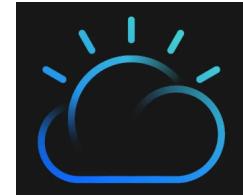
- Community Leadership



- IBM Platform Enablement



- Cloud Enablement



- Developer Enablement

- Commercial Support

IBM's JavaScript/Node.js Focus

- Community Leadership



- IBM Platform Enablement



- Cloud Enablement



- Developer Enablement

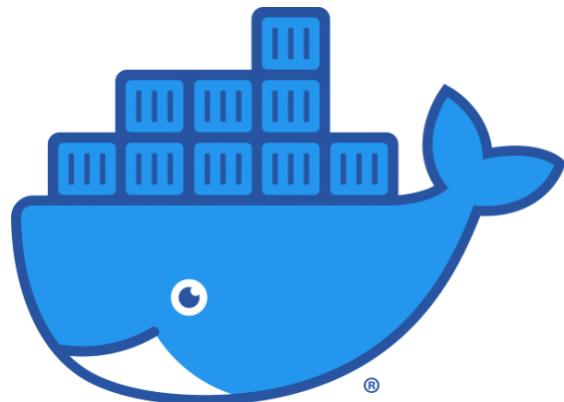
- Commercial Support

Developer Enablement

- Rapid Development
- Successful Production Deployments

Developer Enablement

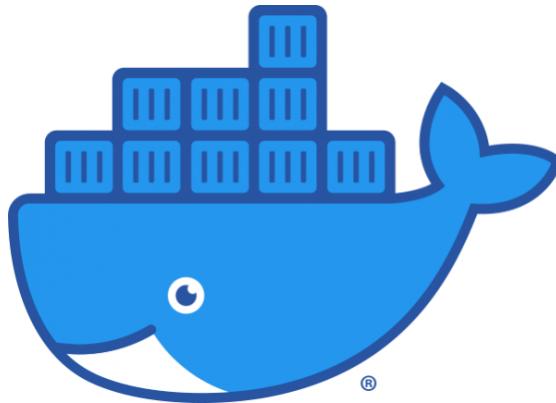
- Rapid Development
- Successful Production Deployments



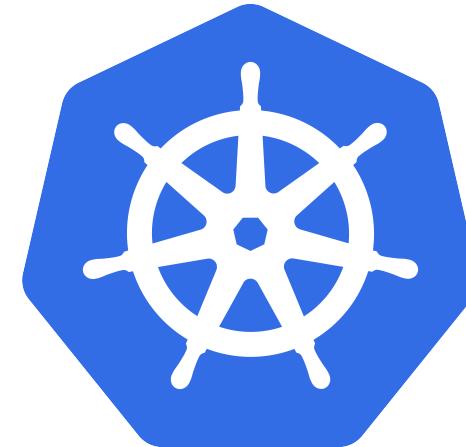
<https://www.docker.com/>

Developer Enablement

- Rapid Development
- Successful Production Deployments



<https://www.docker.com/>



<https://github.com/kubernetes/kubernetes>

Cloud Native Development

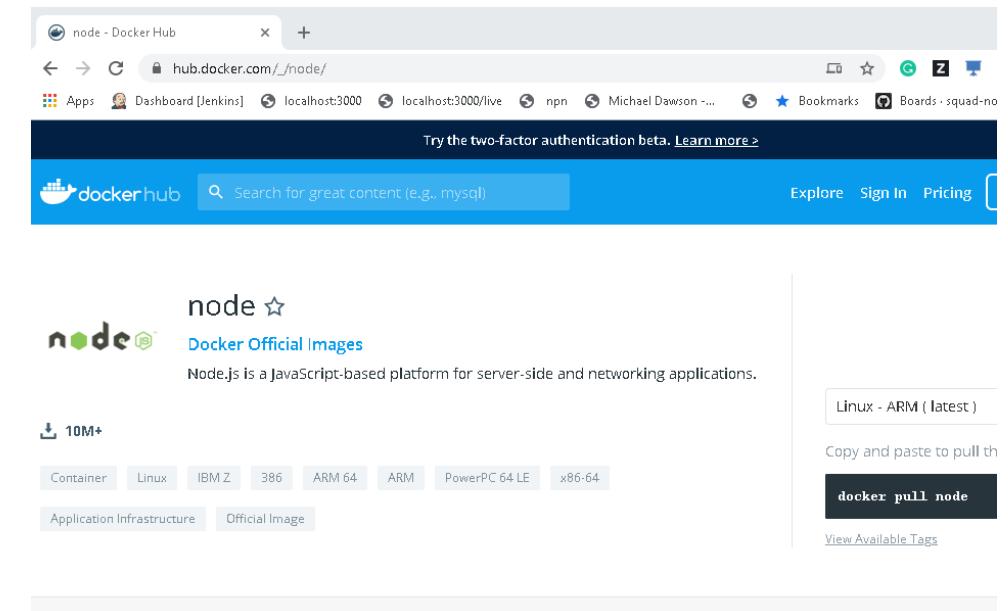
- 12 Factor Applications - <https://12factor.net/>
 - Keep development, staging, and production as similar as possible
 - Treat Logs as event streams
 - Export services via port binding
- Some key Tasks
 - Building Docker Images
 - Deployment
 - Testing
 - Health checking
 - Logs
 - Metrics
 - Upgrades



Building Docker Images – Base Images

- Community Node.js Images (10M+ downloads)
 - Ex node:12.13.1
 - Add Node.js binaries to existing distro, in default path
 - Variants
 - node:<version>
 - Node:<version>-slim – minimal packages to run node
 - <version>-alpine – smaller container

<https://github.com/nodejs/docker-node/blob/master/docs/BestPractices.md>

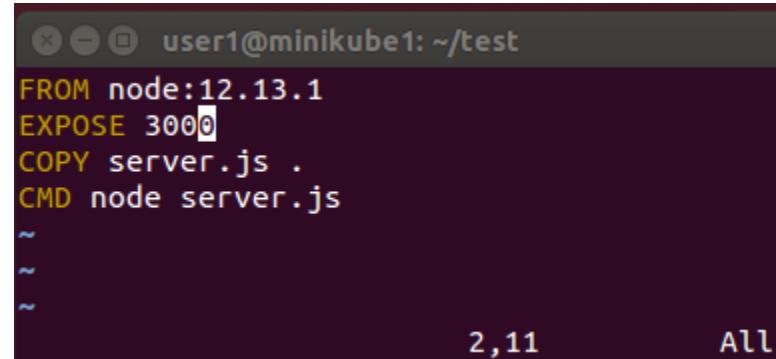


https://hub.docker.com/_/node/

- Others:
 - registry.access.redhat.com/ubi8/nodejs-10

Building Docker Images - Building

- Dockerfile



A screenshot of a terminal window titled "user1@minikube1: ~/test". The window displays a Dockerfile with the following content:

```
FROM node:12.13.1
EXPOSE 3000
COPY server.js .
CMD node server.js
```

The terminal shows three blank lines at the bottom, with the numbers "2,11" and "All" aligned to the right.

- Building
 - docker build -t test .
 - Use Multi-stage Build to minimize size
 - For example: native addons

<https://docs.docker.com/develop/develop-images/multistage-build/>

- Push to registry
- Run/test under docker/kubernetes

Deployment Artifacts

- **YAML**
kubectl apply –f deployment.yaml
kubectl apply –f service.yaml
kubectl apply –f ingress.yaml
- Helm Chart
- Operators



```
user1@minikube1: ~/test
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test1
  labels:
    app: test1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test1
  template:
    metadata:
      labels:
        app: test1
    spec:
      containers:
        - name: test1
          image: test1:new
          imagePullPolicy: Never
          args:
```

Deployment Artifacts

- YAML
- Helm Charts <https://github.com/helm/charts>
 - deployment.yaml
 - service.yaml
 - ingress.yaml
- Operators

Package-name/
charts/
templates/
Chart.yaml
LICENSE
README.md
requirements.yaml
values.yaml

Deployment Artifacts

- YAML
- Helm Charts
- Operators <https://github.com/operator-framework>

Testing

- Keep development, staging, and production as similar as possible
- Overhead ?
 - Docker build, start container and access Docker image to test each change!
 - Delete Pods, Re-deploy in Kubernetes



Liveness and Readiness

- Liveness – when to restart container
- Readiness – when ready to start accepting traffic

Liveness

- Probe Types

- Run a command in the container
- HTTP probe
- TCP probe

- http probe common for Node.js Deployments

- OK $\geq 200 < 400$

- Can/should check more than web server
 - Separate endpoint

```
containers:  
- name: test1  
  image: test1:latest  
  args:  
  - /server  
  livenessProbe:  
    httpGet:  
      path: /live  
      port: 3000  
    initialDelaySeconds: 5  
    periodSeconds: 10
```

Liveness

- Probe Types

- Run a command in the container
- HTTP probe
- TCP probe

- http probe common for Node.js Deployments

- OK $\geq 200 < 400$

- Can/should check more than web server

```
containers:  
- name: test1  
  image: test1:latest  
  args:  
  - /server  
  livenessProbe:  
    httpGet:  
      path: /live  
      port: 3000  
    initialDelaySeconds: 5  
    periodSeconds: 10
```

Readiness

- Same options/config as liveness probe
- No restart but traffic is not routed to instance

```
user1@minikube1:~/test$ kubectl get all
NAME      READY   STATUS    RESTARTS   AGE
pod/test1  0/1     Running   0          6s
```

```
user1@minikube1:~/test
apiVersion: v1
kind: Pod
metadata:
  name: test1
  labels:
    app: test1
spec:
  containers:
  - name: test1
    image: test1:latest
    imagePullPolicy: Never
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /live
        port: 3000
      initialDelaySeconds: 5
      periodSeconds: 10
    readinessProbe:
      httpGet:
        path: /ready
        port: 3000
      initialDelaySeconds: 1
      periodSeconds: 10
~
~
~
```

<" 25L, 439C 20,13 All

Readir

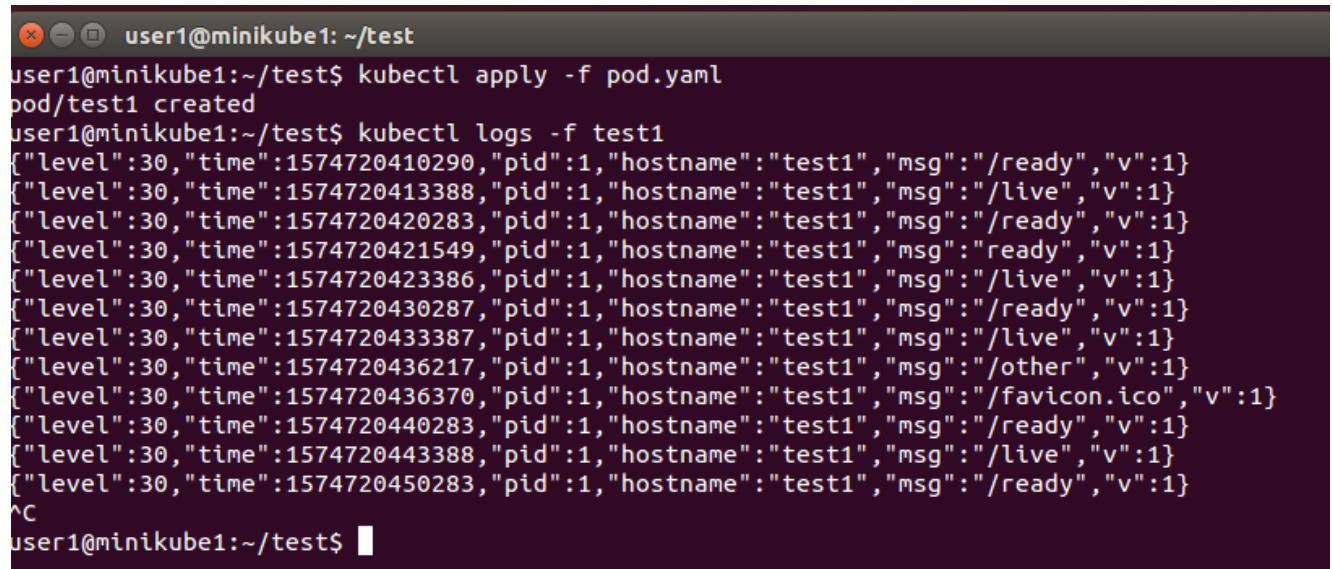
- Same op
- No restart

The screenshot shows a GitHub repository page for "CloudNativeJS/cloud-health". The repository name is at the top left. Below it is a search bar and a navigation bar with links to "Apps", "Dashboard [Jenkins]", "localhost:3000", and "localhost:3000/live". The main content area displays the repository's details: "CloudNativeJS / cloud-health", "Code" (selected), "Issues 3", "Pull requests 1", and "Actions". A brief description follows: "A core Health Check module for use in building a health endpoint". At the bottom, there are statistics: "27 commits", "2 branches", "0 packages", and a date range "20, 13 All". On the right side of the page, there is a terminal window titled "minikube1: ~ / test" showing a command-line session with several lines of configuration or log output.

```
v1
t1
st1
s:
est1
test1:latest
llPolicy: Never
er
sProbe:
et:
h: /live
t: 3000
alDelaySeconds: 5
dSeconds: 10
ssProbe:
et:
h: /ready
t: 3000
alDelaySeconds: 1
dSeconds: 10
```

Logs

- Containers are ephemeral
- Log to stdout
- Structured logging is the trend



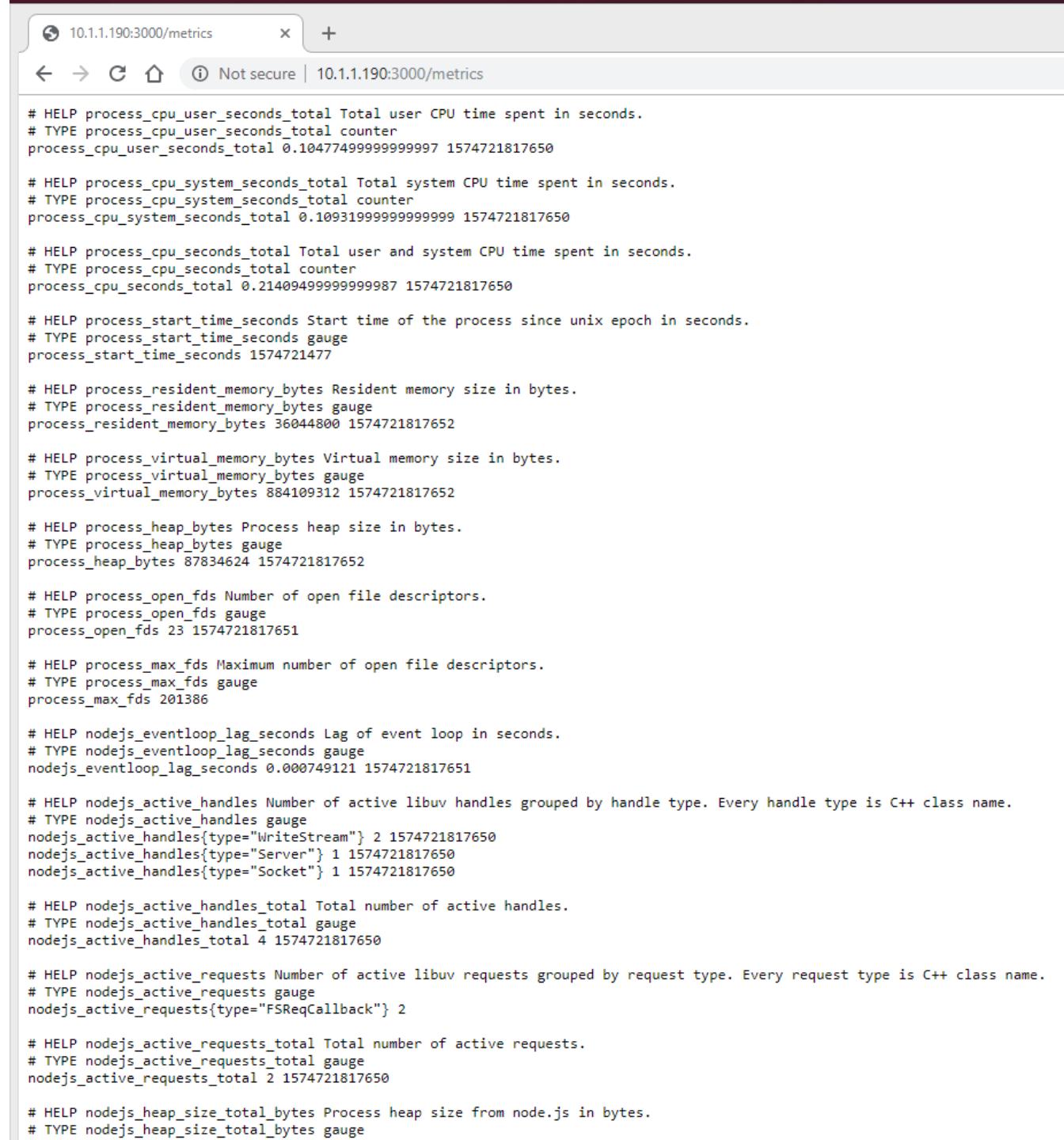
A terminal window titled "user1@minikube1: ~/test". The user runs "kubectl apply -f pod.yaml" which creates a pod named "test1". Then, "kubectl logs -f test1" is run, displaying a stream of structured log entries. The logs show repeated messages at level 30 (INFO) about the pod being ready and live, as well as other events like favicon.ico loading and other activity. The terminal ends with a Ctrl-C interrupt.

```
user1@minikube1:~/test$ kubectl apply -f pod.yaml
pod/test1 created
user1@minikube1:~/test$ kubectl logs -f test1
[{"level":30,"time":1574720410290,"pid":1,"hostname":"test1","msg":"/ready","v":1}
[{"level":30,"time":1574720413388,"pid":1,"hostname":"test1","msg":"/live","v":1}
[{"level":30,"time":1574720420283,"pid":1,"hostname":"test1","msg":"/ready","v":1}
[{"level":30,"time":1574720421549,"pid":1,"hostname":"test1","msg":"ready","v":1}
[{"level":30,"time":1574720423386,"pid":1,"hostname":"test1","msg":"/live","v":1}
[{"level":30,"time":1574720430287,"pid":1,"hostname":"test1","msg":"/ready","v":1}
[{"level":30,"time":1574720433387,"pid":1,"hostname":"test1","msg":"/live","v":1}
[{"level":30,"time":1574720436217,"pid":1,"hostname":"test1","msg":"/other","v":1}
[{"level":30,"time":1574720436370,"pid":1,"hostname":"test1","msg":"/favicon.ico","v":1}
[{"level":30,"time":1574720440283,"pid":1,"hostname":"test1","msg":"/ready","v":1}
[{"level":30,"time":1574720443388,"pid":1,"hostname":"test1","msg":"/live","v":1}
[{"level":30,"time":1574720450283,"pid":1,"hostname":"test1","msg":"/ready","v":1}
^C
user1@minikube1:~/test$
```

kubectl logs -f pod test1

Metrics

- Prometheus <https://prometheus.io/>
- Expose /metrics endpoint to return metrics
 - prom-client
 - appmetrics-Prometheus
- Default and Application specific metrics



The screenshot shows a browser window with the URL 10.1.1.190:3000/metrics. The page displays a list of Prometheus metrics for a process. The metrics are listed in a structured format with HELP comments, metric names, types, and values.

```
# HELP process_cpu_user_seconds_total Total user CPU time spent in seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total 0.1047749999999997 1574721817650

# HELP process_cpu_system_seconds_total Total system CPU time spent in seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total 0.1093199999999999 1574721817650

# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.2140949999999987 1574721817650

# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1574721477

# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 36044800 1574721817652

# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 884109312 1574721817652

# HELP process_heap_bytes Process heap size in bytes.
# TYPE process_heap_bytes gauge
process_heap_bytes 87834624 1574721817652

# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 23 1574721817651

# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 201386

# HELP nodejs_eventloop_lag_seconds Lag of event loop in seconds.
# TYPE nodejs_eventloop_lag_seconds gauge
nodejs_eventloop_lag_seconds 0.000749121 1574721817651

# HELP nodejs_active_handles Number of active libuv handles grouped by handle type. Every handle type is C++ class name.
# TYPE nodejs_active_handles gauge
nodejs_active_handles{type="WriteStream"} 2 1574721817650
nodejs_active_handles{type="Server"} 1 1574721817650
nodejs_active_handles{type="Socket"} 1 1574721817650

# HELP nodejs_active_handles_total Total number of active handles.
# TYPE nodejs_active_handles_total gauge
nodejs_active_handles_total 4 1574721817650

# HELP nodejs_active_requests Number of active libuv requests grouped by request type. Every request type is C++ class name.
# TYPE nodejs_active_requests gauge
nodejs_active_requests{type="FSReqCallback"} 2

# HELP nodejs_active_requests_total Total number of active requests.
# TYPE nodejs_active_requests_total gauge
nodejs_active_requests_total 2 1574721817650

# HELP nodejs_heap_size_total_bytes Process heap size from node.js in bytes.
# TYPE nodejs_heap_size_total_bytes gauge
```

Upgrades

- Kubernetes – handles rolling out upgraded containers
- But not how to update Docker containers contents....
 - X projects * Y containers can be a big number
 - Which ones need to be upgraded?
 - Which versions of common components do they use?
- Imagine moving to a new team
 - What do their docker files look like
 - How are deployment artifacts organized
 - Endpoint names (live, ready, metrics or something else?)
 - Which logger
 - Etc....
- Consistency can help

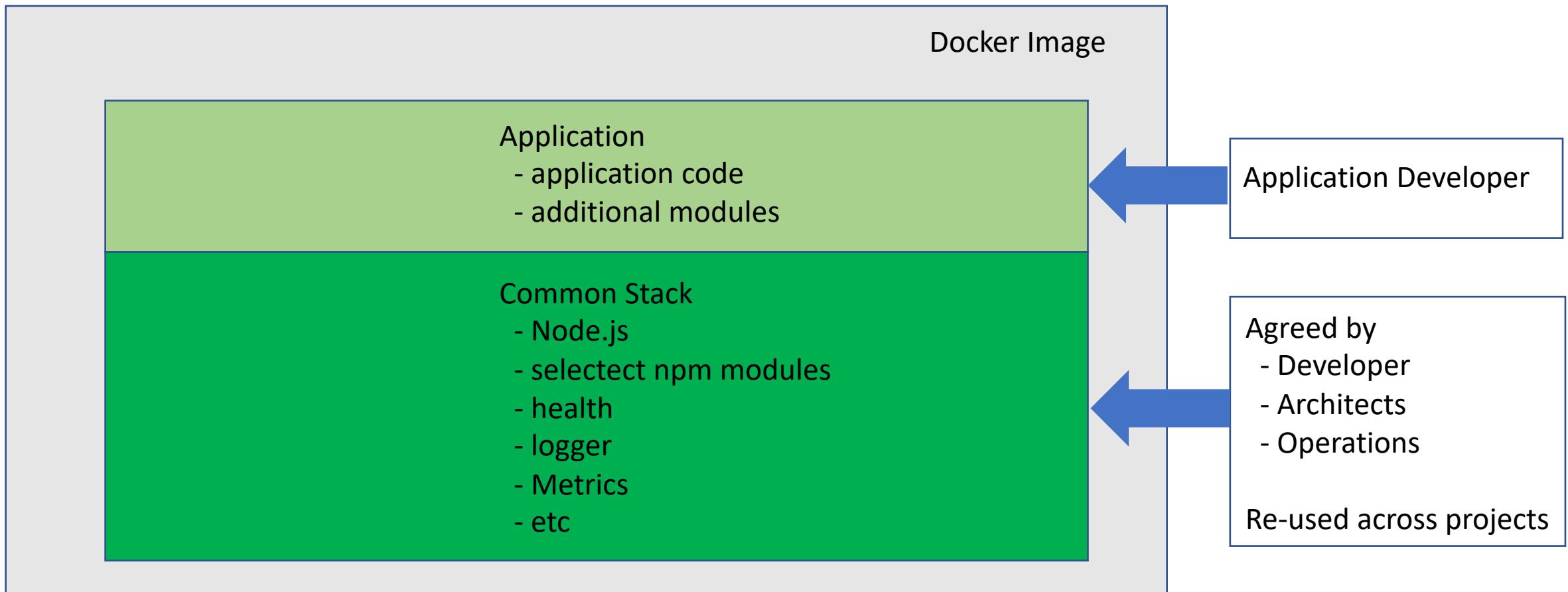
WHAT?

WHAT?

I just want to focus on the
Application Code

Let's Separate the Concerns

Tooling to build/test **efficiently** using best practices



Consistency...

Consistency...

...Not Uniformity

Tools?



appsody



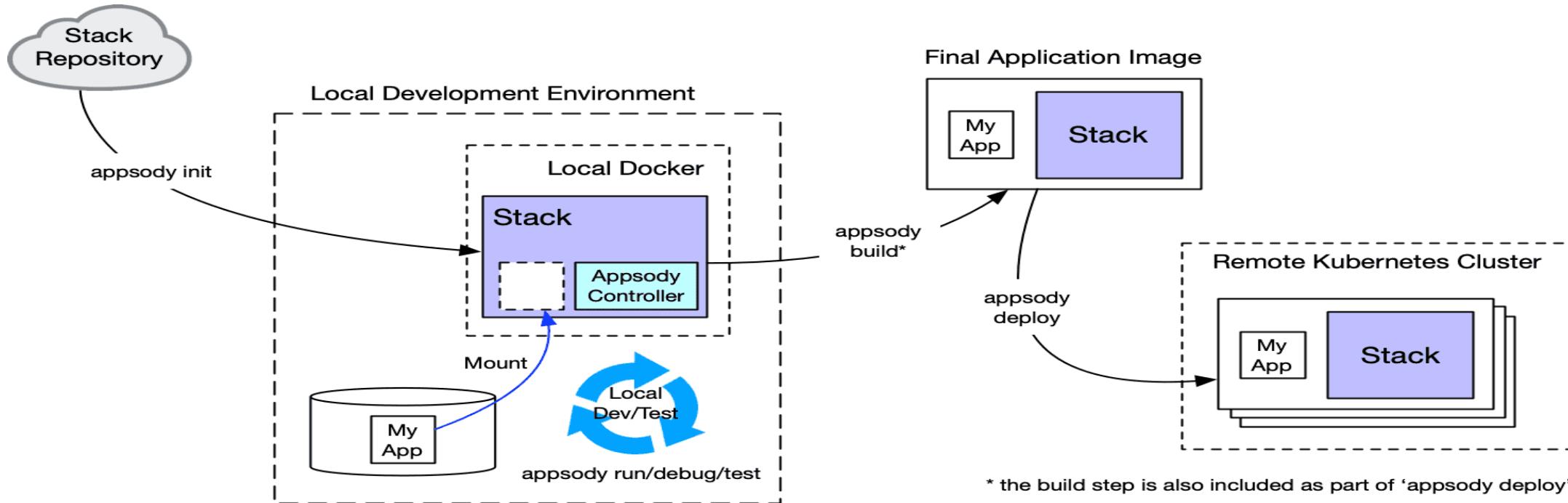
Appsody.dev



[GitHub.com/appsody](https://github.com/appsody)



@appsodydev



Stacks

- Node.js Stacks
 - nodejs
 - nodejs-express
 - nodejs-loopback
 - nodejs-functions

FaaS meets Node.js Frameworks:
Developing Cloud Native Node.js
Applications at Speed !

<https://sched.co/T5G6>

11:40 immediately after this session.

The screenshot shows a GitHub repository page for the 'stacks' repository under the 'incubator' branch. The repository has 56 issues and 12 pull requests. The pull requests listed are:

- Kamran64 java-microprofile: run mvn command in batch (#523)
- java-microprofile java-microprofile: run mvn command in batch (#523)
- java-spring-boot2 Add monitoring to deploy yaml (#491)
- kitura kitura: update to respin Docker image (#481)
- nodejs-express nodejs-express: update express to 4.17 (#507)
- nodejs-loopback nodejs-loopback: upgrade/add application dependencies (#488)
- nodejs nodejs-functions: fix typos in readme (#370)
- python-flask [python-flask] Ensure liveness port is exposed (#475)
- starter [starter-stack] Improve README and name (#437)
- swift swift: bump stack version for Swift 5.1.2 (#478)

Appsody – Demo



- Adds nice GUI
 - Appsody integration
 - Bring your own visual editor (ex VS Code)
 - Performance Monitoring
 - Tekton Pipelines
- <https://developer.ibm.com/blogs/codewind-introduction/>



A modern microservices-based framework for continuous deploy of apps on
Kubernetes and Knative



Built on popular open source projects



To learn more

- IBM Booth
 - Node.js Quick lab (15 mins)
 - Come talk to us
- Node.js On-line Lab (75 mins) - <https://kabanero-workshop-nodejs.mybluemix.net/guides/dev-getting-started>
- <https://www.cloudnativejs.io/>



Q/A

Copyright and Trademarks

© IBM Corporation 2019. All Rights Reserved

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at

www.ibm.com/legal/copytrade.shtml

Node.js is an official trademark of Joyent. IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

npm is a trademark of npm, Inc.

Other trademarks or logos are owned by their respective owners.