Node.js in the Cloud

...

Red Hat Node.js Team

Welcome 👋

- Introduction
- Overview of cloud native concepts
- 3 Part workshop
  - Cloud Native concepts
  - Building your first REST API
  - Full stack deployment

Cloud Native Workshop

- Introduction to cloud-native development with Node.js.
- Walking you through how to extend an Express.js-based application to leverage cloud capabilities.
- The workshop will cover key concepts and technologies, including:
  - Health checks
  - Metrics
  - Docker
  - Kubernetes
  - Prometheus
  - Grafana

Tutorial Prerequisites

https://github.com/nodeshift/tutorial/tree/main/cloud-native#prerequisites

What are cloud native applications?

Cloud Native Glossary

Cloud Native Glossary

Building Enterprise Cloud Native Applications

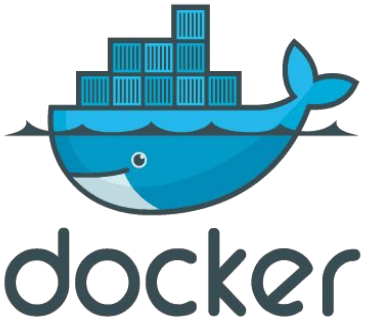
Building Enterprise Cloud Native Node.js Applications

- Module/dependency diligence
  - Consider licenses, security issues, maintenance, compatibility, support
- Functional Components
  - Web Framework, GraphQL, Caching, Authentication/Authorization
- Development
  - Code consistency, testing, proxying, etc.
- Operations
  - Health checks
  - Metrics

Reference Architecture for Node.js Applications

https://github.com/nodeshift/nodejs-reference-architecture

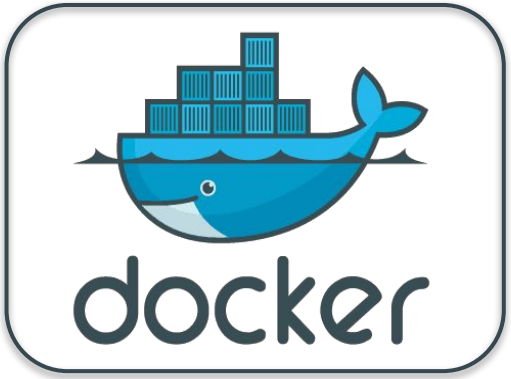
CLOUD NATIVE
COMPUTING FOUNDATION



kubernetes

Prometheus

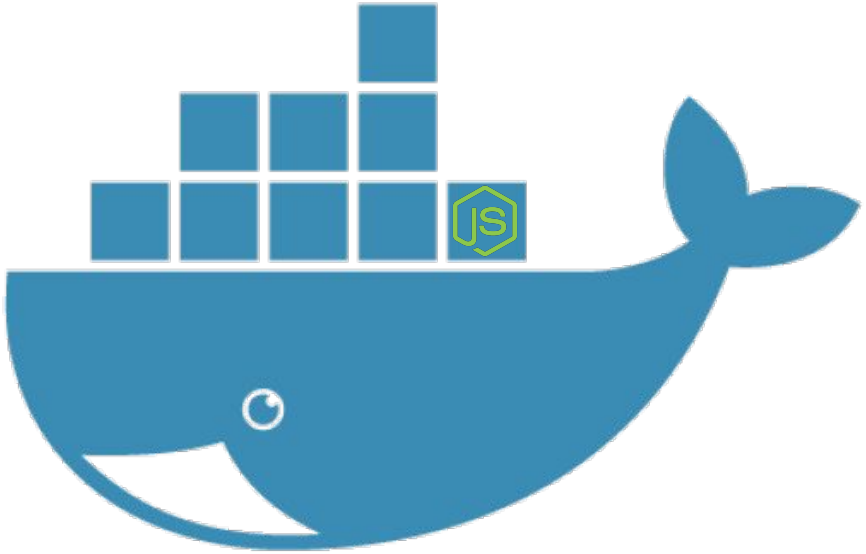
CLOUD NATIVE
COMPUTING FOUNDATION



kubernetes

Prometheus


docker



- Tool designed to make it easier to create, deploy, and run applications by using containers
- Containers allow you to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it out as one package
- Docker is *a bit like* a virtual machine, but rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

Node.js 14 Docker Image

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

package.json

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

node_modules

package.json

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

Application

node_modules

package.json

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

Application

node_modules

package.json

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

Application

node_modules

package.json

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

Application

node_modules

package.json

Operating System Updates

Node.js 14 Docker Image

986 MB

Dockerfile

```
FROM node:14

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install npm production packages
RUN npm install --production

COPY . /app

ENV NODE_ENV production
ENV PORT 3000

EXPOSE 3000

USER node

CMD ["npm", "start"]
```

node_modules

package.json

Operating System Updates

Node.js 14 Docker Image

Dockerfile

```
# Install the app dependencies in a full Node docker image
FROM node:14
WORKDIR "/app"

# Install OS updates
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install app dependencies
RUN npm install --production
```

```
# Copy the dependencies into a Slim Node docker image
FROM node:14-slim
 WORKDIR "/app"

# Install OS updates
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Install app dependencies
COPY --from=0 /app/node_modules /app/node_modules
COPY . /app

ENV NODE_ENV production
ENV PORT 3000

USER node

EXPOSE 3000
CMD ["npm", "start"]
```

node_modules

package.json

Operating System Updates

Node.js 14 Slim Docker Image

Dockerfile

```
# Install the app dependencies in a full Node docker image
FROM node:14
WORKDIR "/app"

# Install OS updates
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install app dependencies
RUN npm install --production

# Copy the dependencies into a Slim Node docker image
FROM node:14-slim
WORKDIR "/app"

# Install OS updates
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Install app dependencies
COPY --from=0 /app/node_modules /app/node_modules
COPY . /app

ENV NODE_ENV production
ENV PORT 3000

USER node

EXPOSE 3000
CMD ["npm", "start"]
```

Application

node_modules

package.json

Operating System Updates

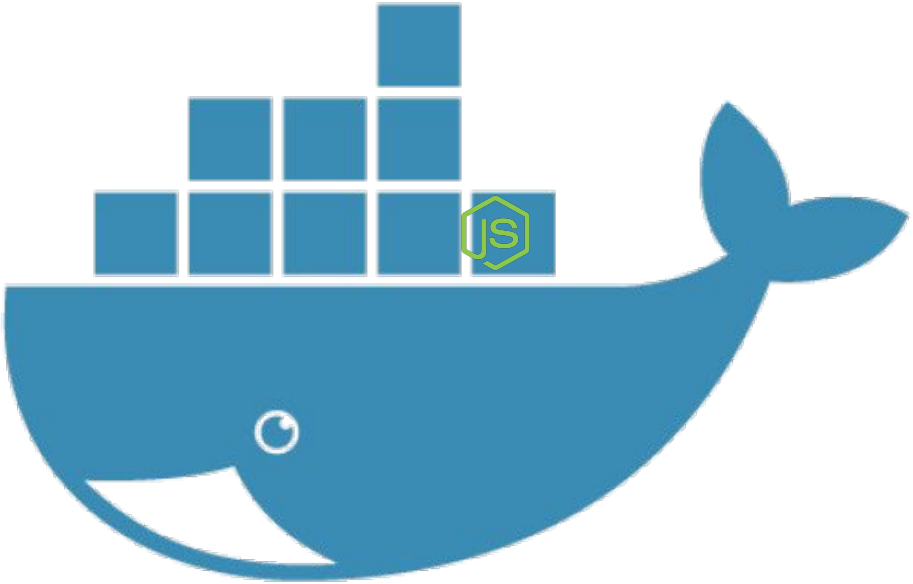
Node.js 14 Slim Docker Image

190 MB

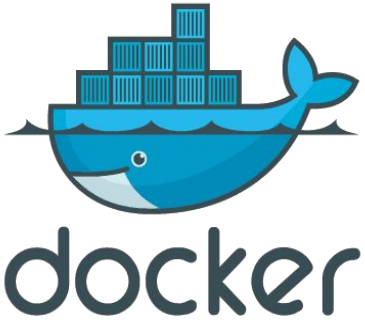
Dockerfile

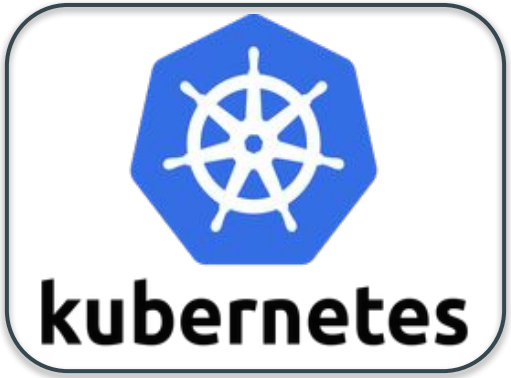
```
# Install the app dependencies in a full Node docker image
FROM node:14
WORKDIR "/app"

# Install OS updates
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Copy package.json and package-lock.json
COPY package*.json ./

# Install app dependencies
RUN npm install --production

# Copy the dependencies into a Slim Node docker image
FROM node:14-slim
WORKDIR "/app"

# Install OS updates
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Install app dependencies
COPY --from=0 /app/node_modules /app/node_modules
COPY . /app

ENV NODE_ENV production
ENV PORT 3000

USER node

EXPOSE 3000
CMD ["npm", "start"]
```


```
$ docker build --tag nodeserver --file Dockerfile-run .
$ docker run --detach --publish 3000:3000 --tty nodeserver
```

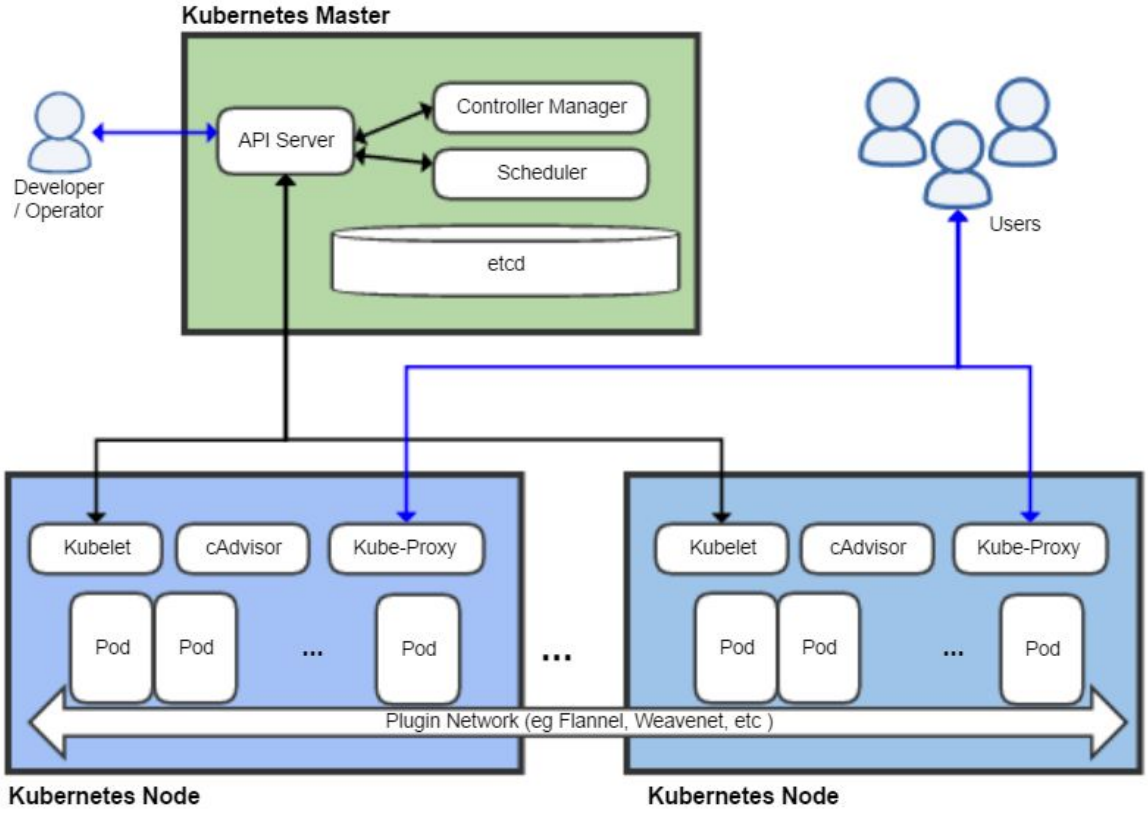


CLOUD NATIVE
COMPUTING FOUNDATION





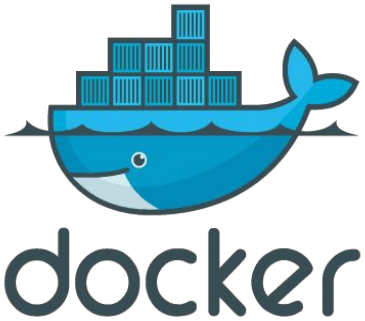
kubernetes


Kubernetes Master
Developer / Operator
API Server
Controller Manager
Scheduler
etcd
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

Manages Your Containers

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing - specifying how much CPU and memory each container needs
- Self-healing
- Secret and configuration management

CLOUD NATIVE
COMPUTING FOUNDATION



kubernetes

Prometheus

HELM CHARTS

- Helm uses a packaging format called *charts*.
- A chart is a collection of files that describe a related set of Kubernetes resources.
- *A bit like* package.json for Kubernetes deployment

HELM CHARTS

```
$ helm create chart
$ ls
chart/
├── .helmignore   # Contains patterns to ignore when packaging Helm charts.
├── Chart.yaml    # Information about your chart
├── values.yaml   # The default values for your templates
├── charts/       # Charts that this chart depends on
└── templates/    # The template files
    └── tests/    # The test files
```

HELM CHARTS

```
$ helm create chart
$ ls
chart/
├── .helmignore
├── Chart.yaml
├── values.yaml
├── charts/
└── templates/
    └── tests/
```

```yaml
apiVersion: v2
name: node-app
description: A sample Helm chart
type: application
version: 0.1.0
appVersion: "1.16.0"
```

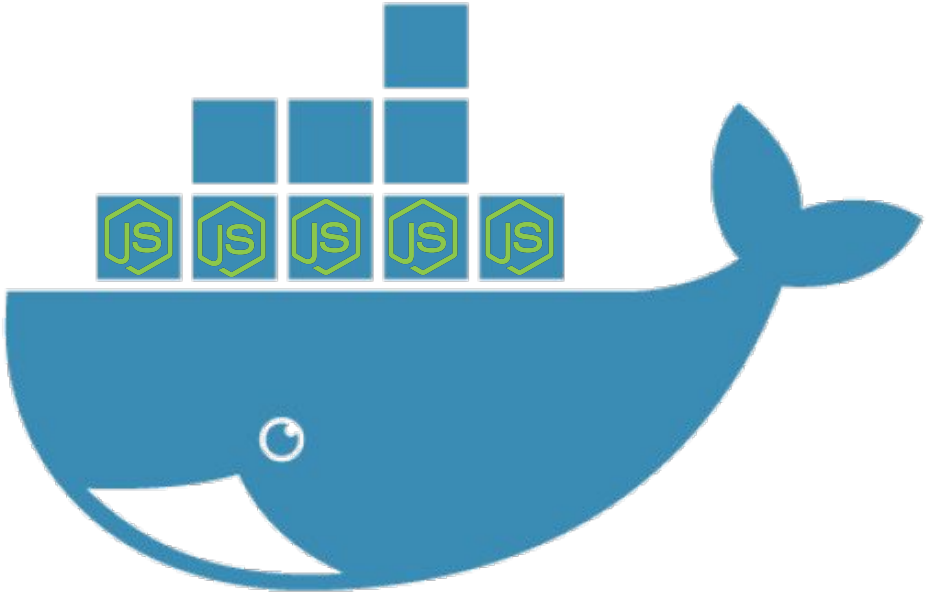
HELM CHARTS

```
$ helm create chart
$ ls
chart/
├── .helmignore
├── Chart.yaml
├── values.yaml
├── charts/
└── templates/
    └── tests/
```

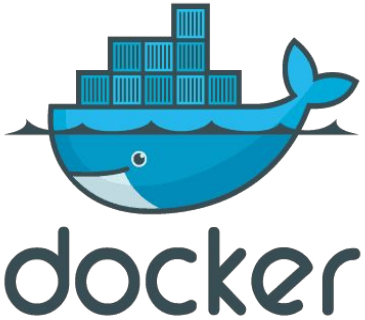
```yaml
replicaCount: 1
image:
 repository: nodeserver
 pullPolicy: IfNotPresent
 tag: ""
service:
 type: ClusterIP
 port: 80
resources:
 limits:
   cpu: 100m
   memory: 128Mi
autoscaling:
 enabled: false
 minReplicas: 1
 maxReplicas: 100
 targetCPUUtilizationPercentage: 80
```

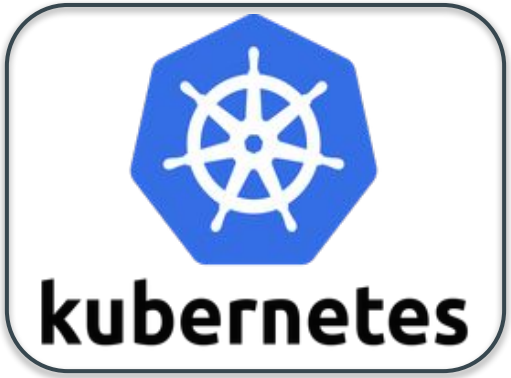
kubernetes

```
$ cd ./chart/nodeserver/
$ helm install --name nodeserver .
```

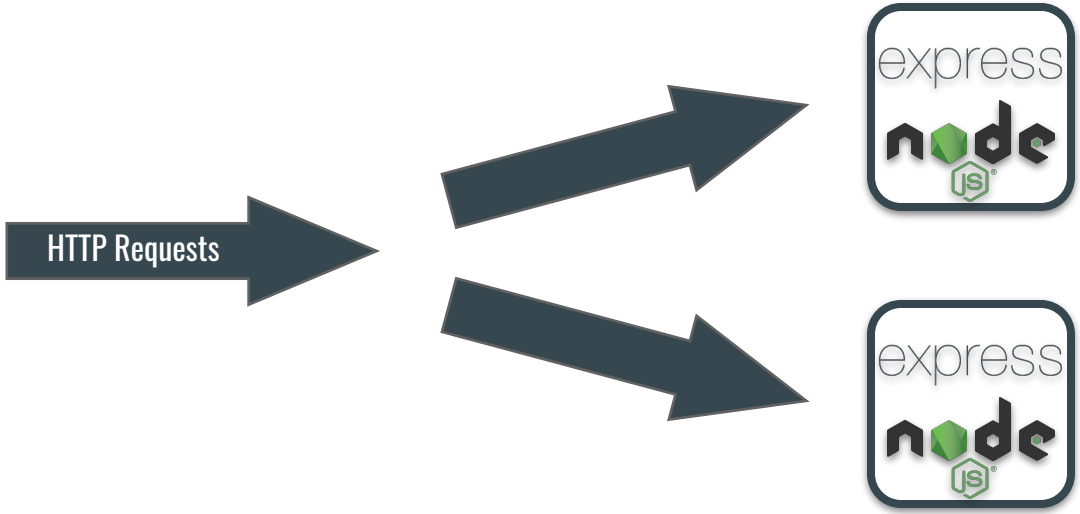


CLOUD NATIVE
COMPUTING FOUNDATION

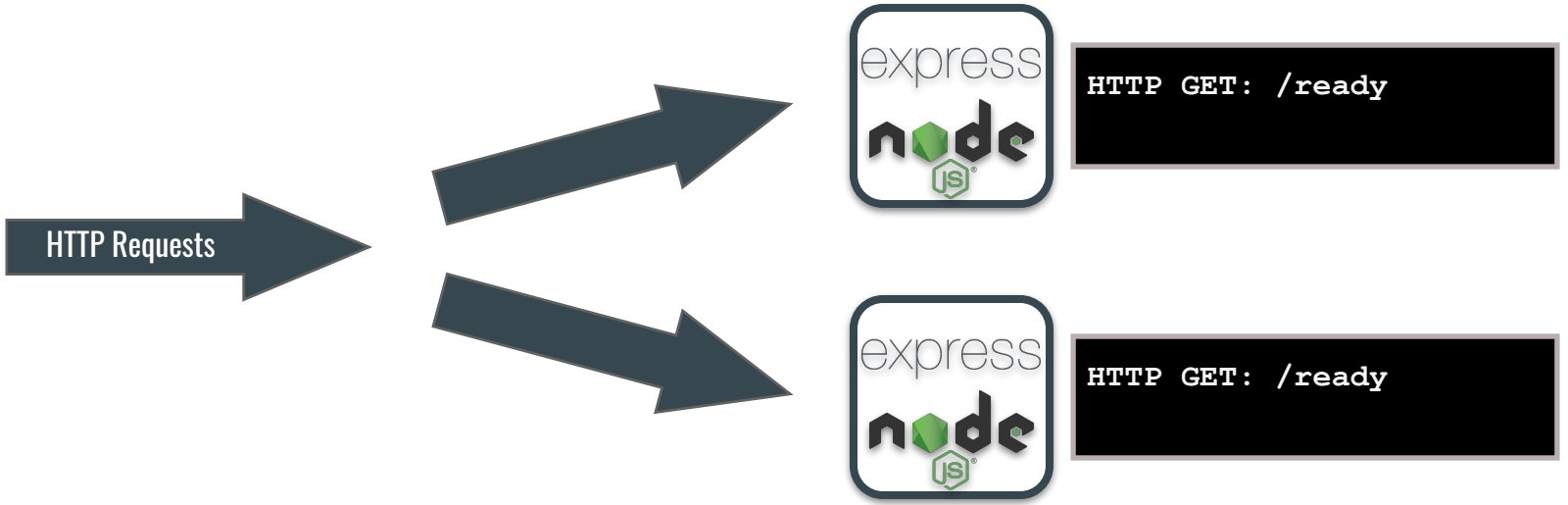




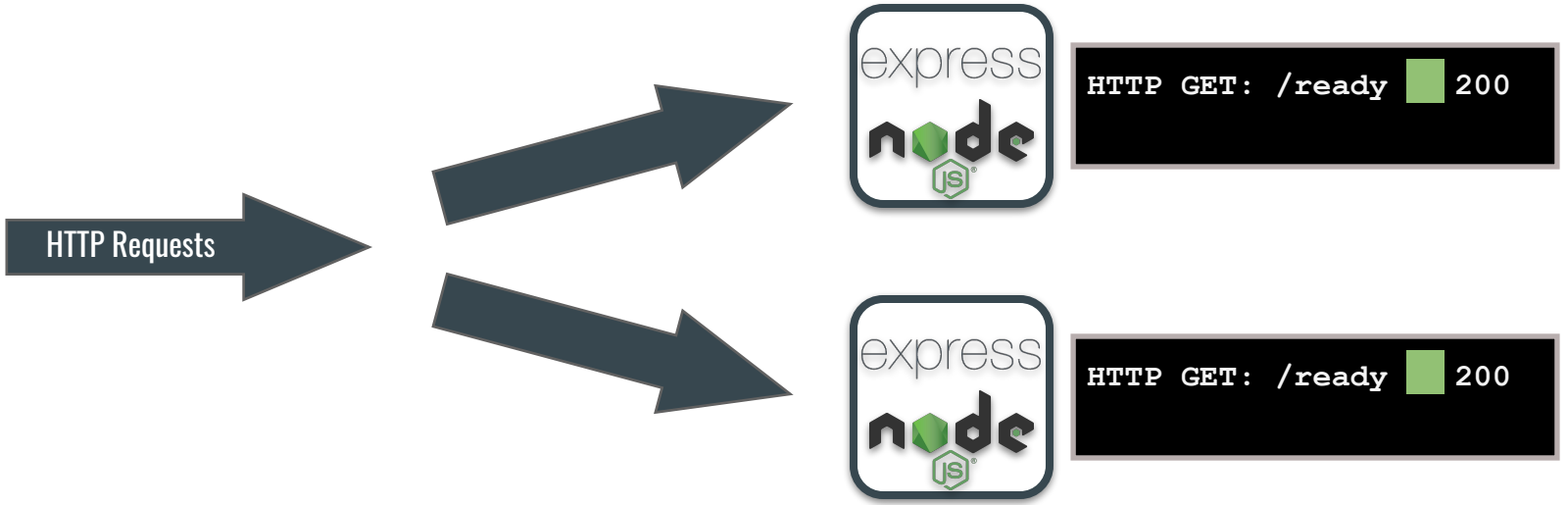
Health Checks



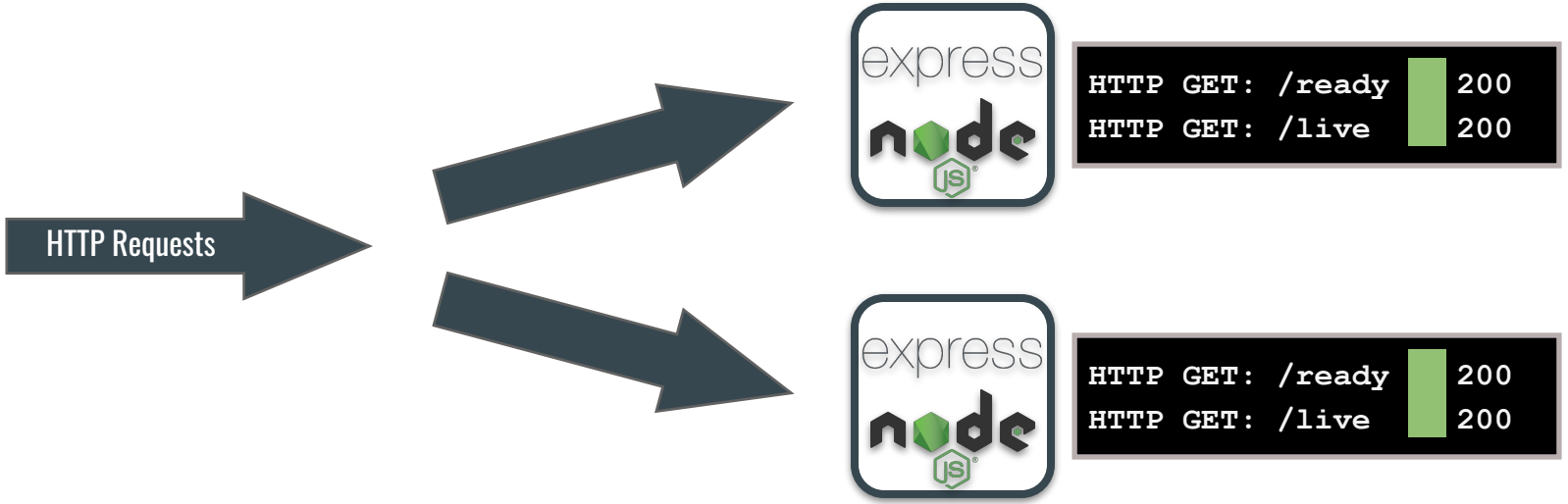
Health Checks



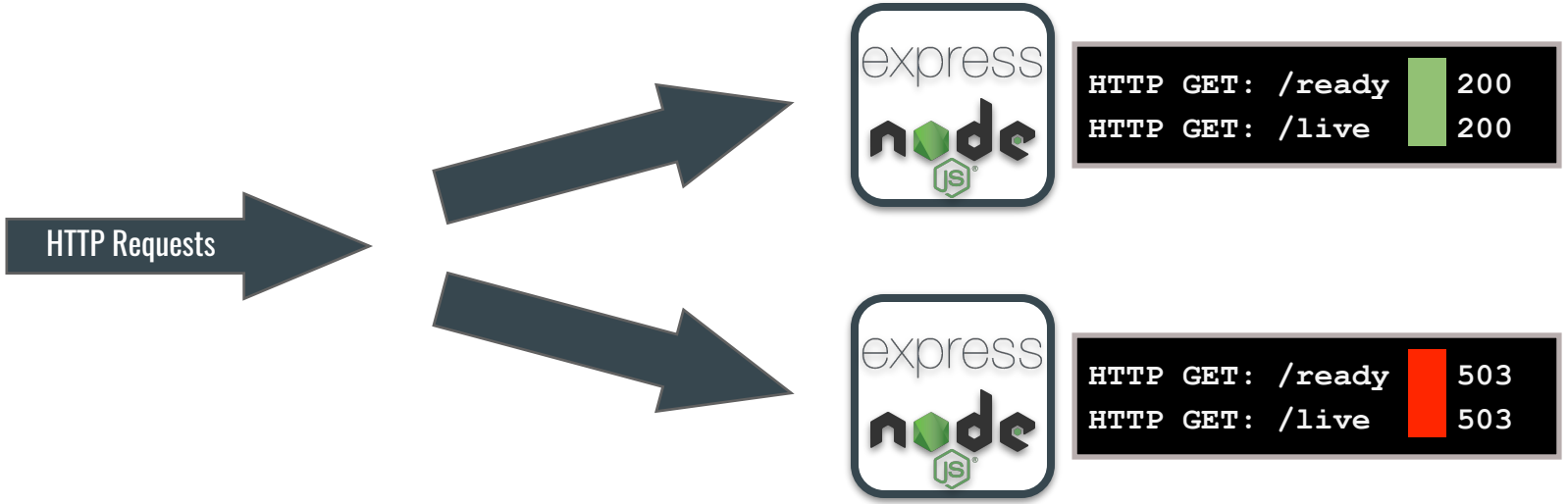
Health Checks



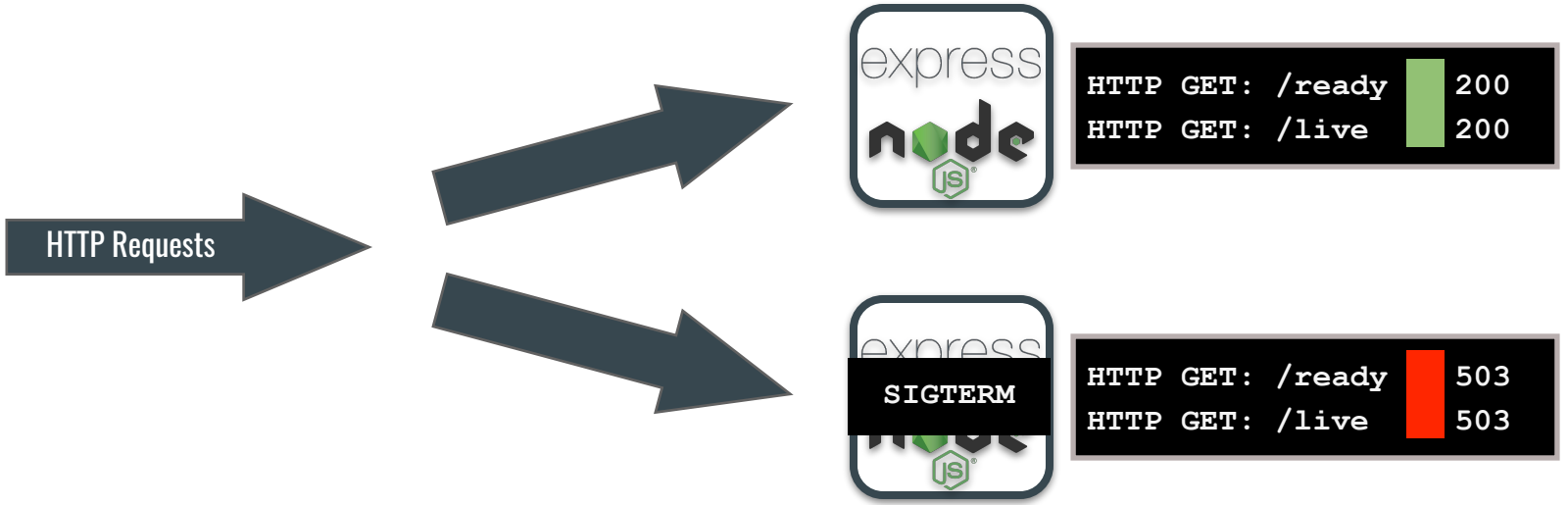
Health Checks



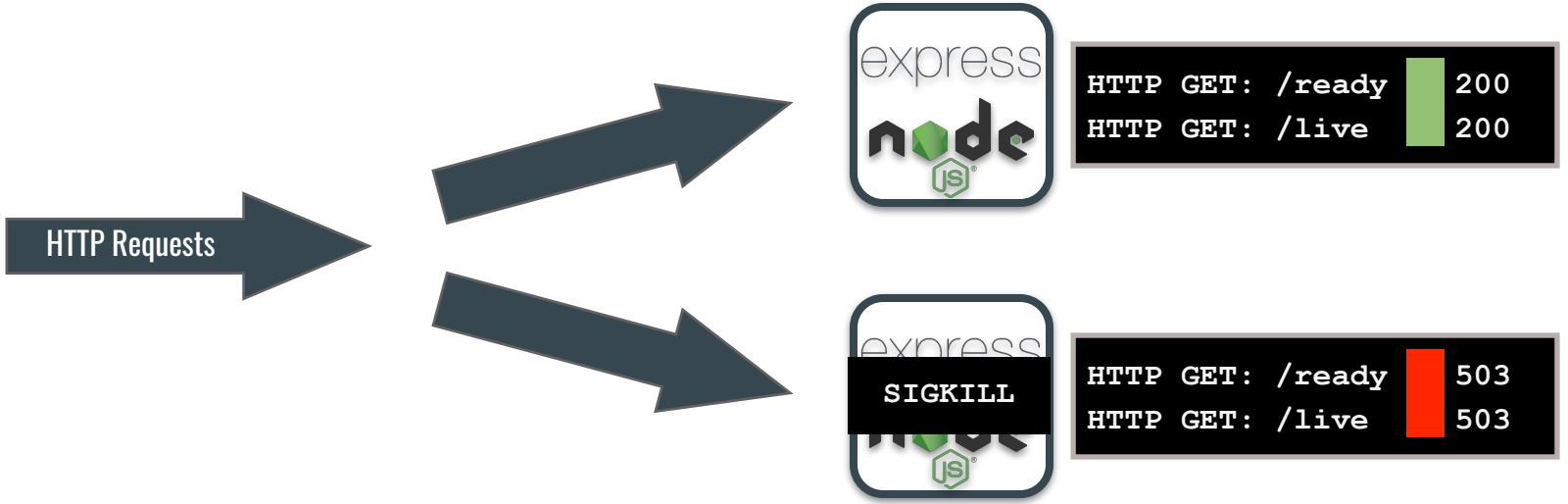
Health Checks



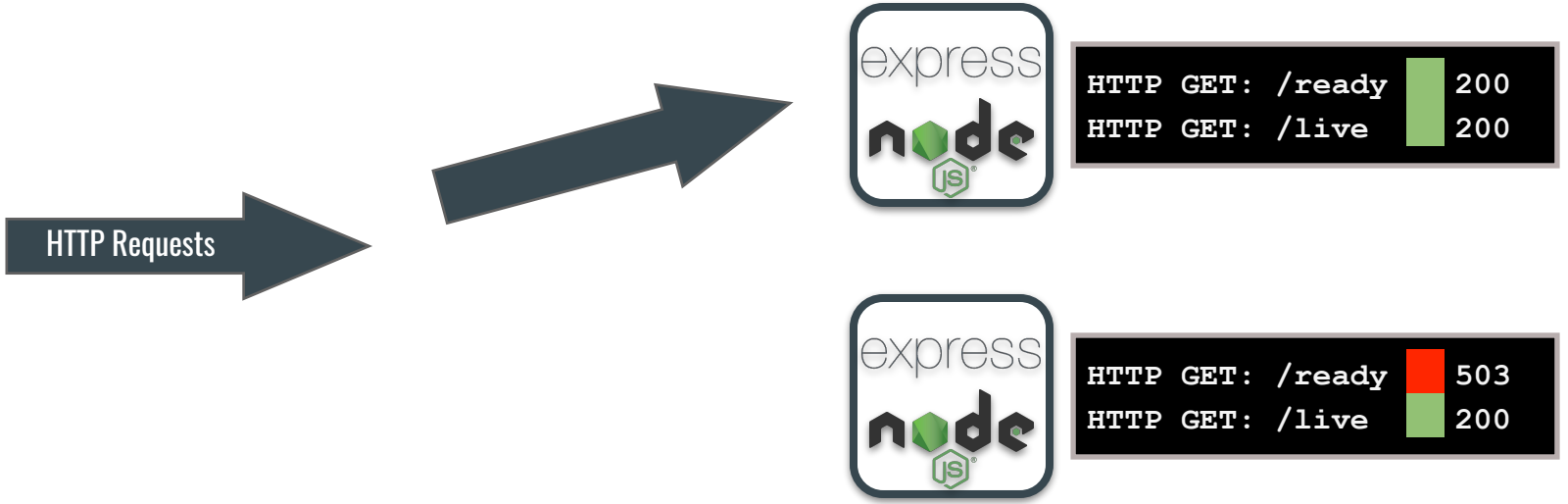
Health Checks



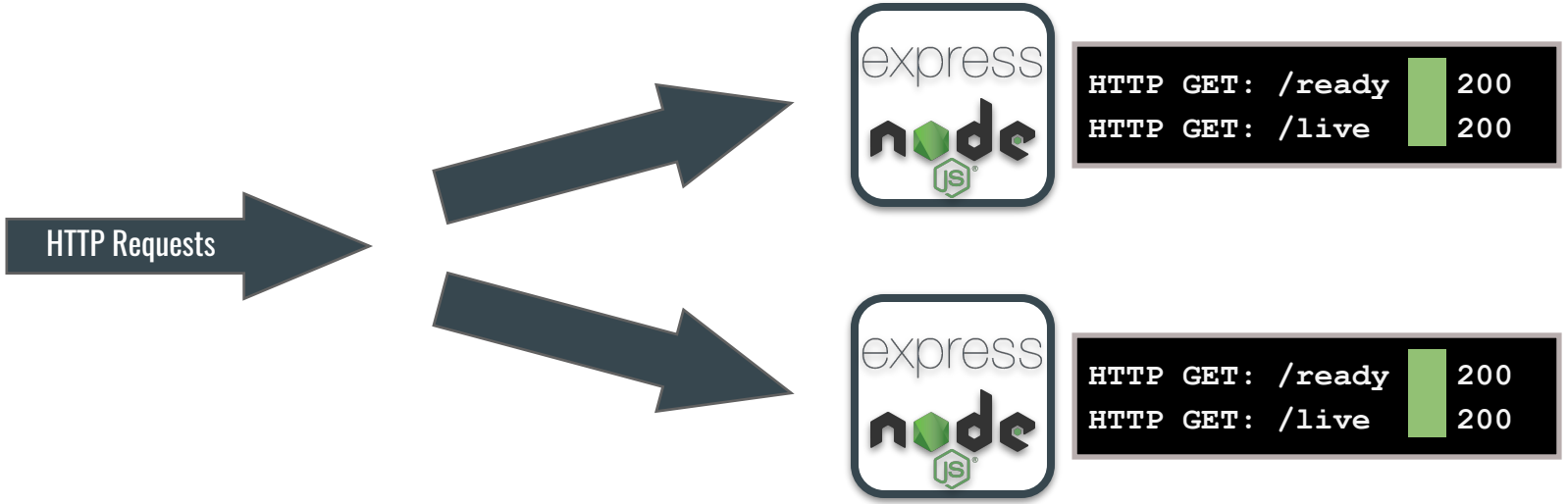
Health Checks



Health Checks



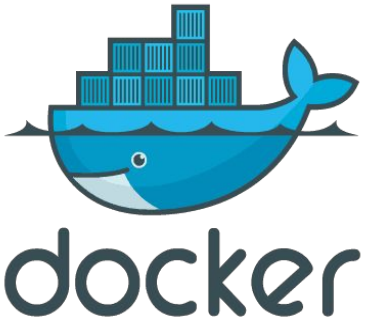
Health Checks



Health Checks

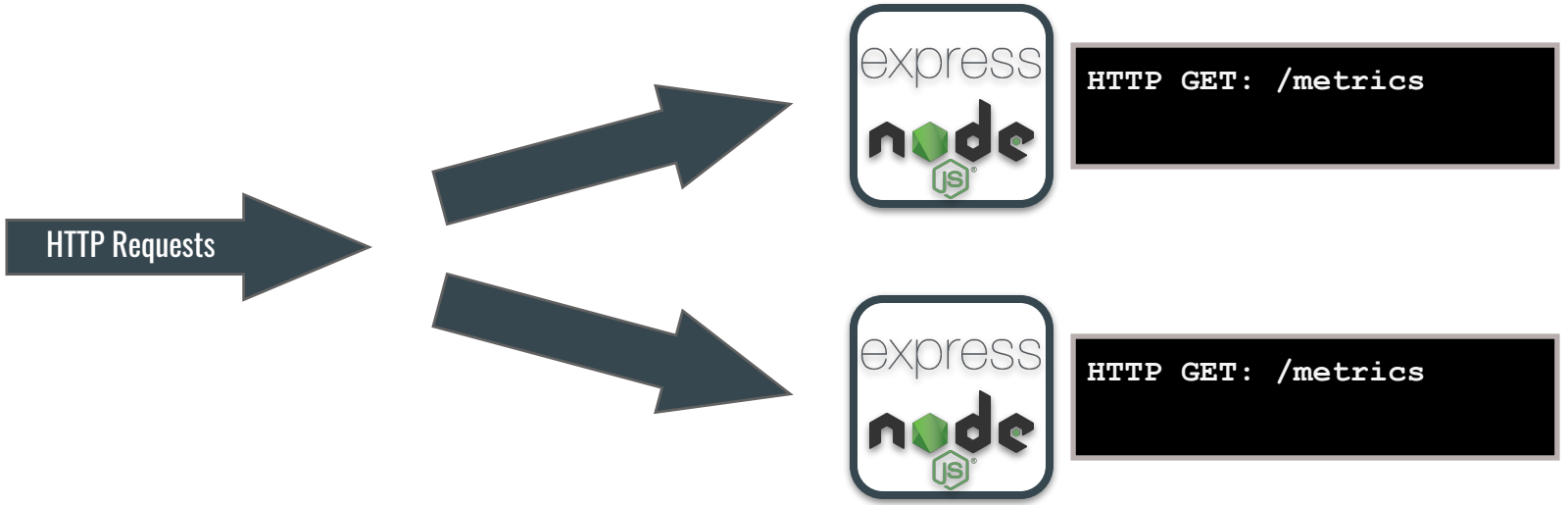
```
const app = require("express")();

// Note that when collecting metrics, the management endpoints should be
// implemented before the instrumentation that collects metrics, so that
// these endpoints are not counted in the metrics.
app.get("/ready", (req, res) => res.status(200).json({ status: "ok" }));
app.get("/live", (req, res) => res.status(200).json({ status: "ok" }));

// ... rest of app...

app.listen();
```

CLOUD NATIVE
COMPUTING FOUNDATION



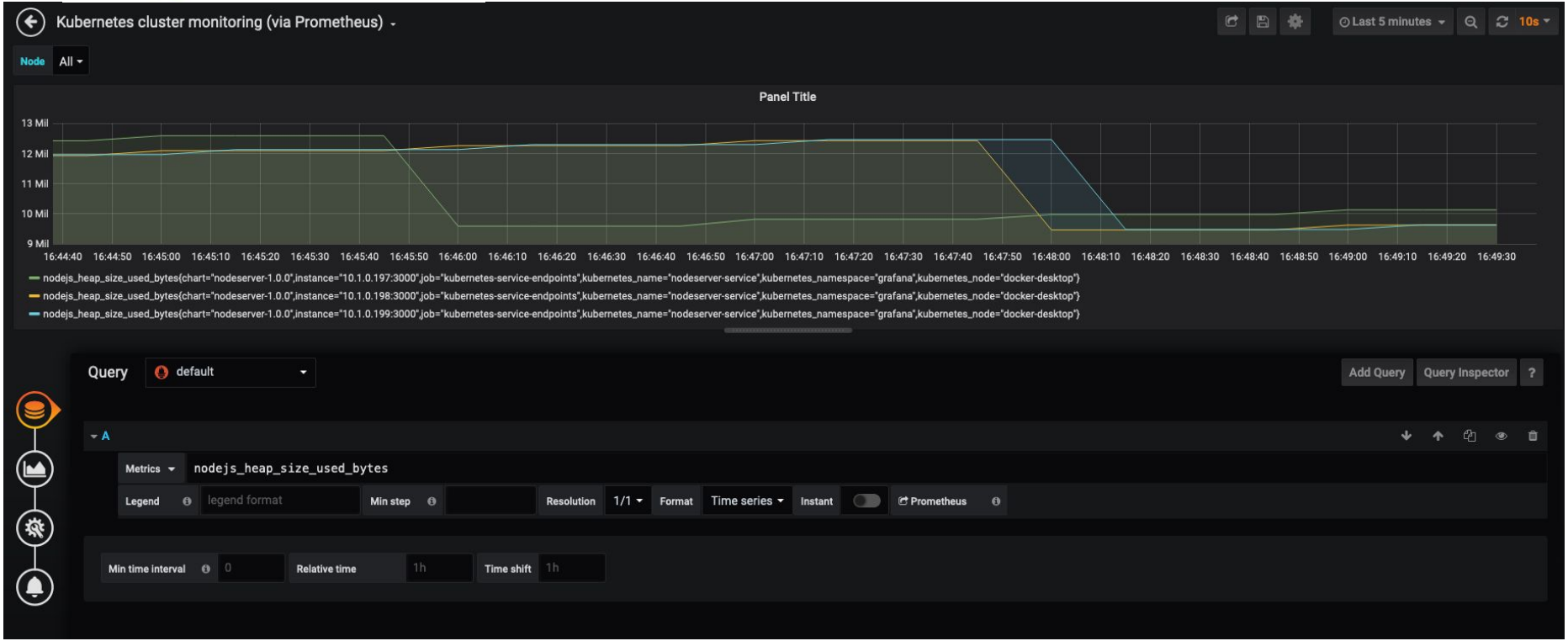
kubernetes

Prometheus


HTTP Requests
express
node
express
node
HTTP GET: /metrics
HTTP GET: /metrics


```
// Prometheus client setup
const Prometheus = require('prom-client');
Prometheus.collectDefaultMetrics();

app.get('/metrics', async (req, res, next) => {
  try {
     res.set('Content-Type', Prometheus.register.contentType);
     const metrics = await Prometheus.register.metrics();
     res.end(metrics);
  } catch {
     res.end('');
  }
});
```


```
# 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 0.020806

# 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 0.005973

# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.026779

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


Kubernetes cluster monitoring (via Prometheus)
Last 5 minutes
10s
Node
All
Panel Title
13 Mil
12 Mil
11 Mil
10 Mil
9 Mil
16:44:40 16:44:50 16:45:00 16:45:10 16:45:20 16:45:30 16:45:40 16:45:50 16:46:00 16:46:10 16:46:20 16:46:30 16:46:40 16:46:50 16:47:00 16:47:10 16:47:20 16:47:30 16:47:40 16:47:50 16:48:00 16:48:10 16:48:20 16:48:30 16:48:40 16:48:50 16:49:00 16:49:10 16:49:20 16:49:30
nodejs_heap_size_used_bytes{chart="nodeserver-1.0.0",instance="10.1.0.197:3000",job="kubernetes-service-endpoints",kubernetes_name="nodeserver-service",kubernetes_namespace="grafana",kubernetes_node="docker-desktop"}
nodejs_heap_size_used_bytes{chart="nodeserver-1.0.0",instance="10.1.0.198:3000",job="kubernetes-service-endpoints",kubernetes_name="nodeserver-service",kubernetes_namespace="grafana",kubernetes_node="docker-desktop"}
nodejs_heap_size_used_bytes{chart="nodeserver-1.0.0",instance="10.1.0.199:3000",job="kubernetes-service-endpoints",kubernetes_name="nodeserver-service",kubernetes_namespace="grafana",kubernetes_node="docker-desktop"}
Query
default
Add Query
Query Inspector
A
Metrics
nodejs_heap_size_used_bytes
Legend
legend format
Min step
Resolution
1/1
Format
Time series
Instant
Prometheus
Min time interval
0
Relative time
1h
Time shift
1h

Summary

- Introduction to cloud-native development with Node.js.
- Walking you through how to extend an Express.js-based application to leverage cloud capabilities.
- The workshop will cover key concepts and technologies, including:
  - Health checks
  - Metrics
  - Docker
  - Kubernetes
  - Prometheus
  - Grafana

Part 2 - Building your first REST API

In this tutorial, we will take you through the basic steps of creating an API that implements the four base operations of persistent storage: create, read, update, and delete (CRUD).

We will do this manually so that you learn the basics of what is going on behind the scenes.

Part 3 - Full stack deployment

This hands-on lab guides you through each step of deploying a full-stack JavaScript application in a Red Hat OpenShift cluster using the Developer Sandbox for Red Hat OpenShift:

- Part 1: Set up your environment and run the application locally, then build a container for the front end and deploy it to OpenShift.
- Part 2: Deploy the back end and connect it to the front end using environment variables, then add a health check.
- **Part 3: Deploy a database and connect everything together, then deploy a new microservice from an existing container.**

Workshop - 3 Parts

- Cloud Native concepts -
  - Self-paced workshop - Instructions provided as GitHub README
  - https://github.com/nodeshift/tutorial/blob/main/cloud-native/README.md
- Building your first REST API -
  - Self-paced workshop - Instructions provided as GitHub README
  - https://github.com/nodeshift/tutorial/blob/main/api/README.md
- Full stack deployment -
  - Sandbox based interactive tutorial
  - https://developers.redhat.com/developer-sandbox/activities/deploying-full-stack-javascript-applications-to-the-sandbox/part1

Node.js in the Cloud Workshop

🔗 Tutorial - https://dn.dev/rh-nodeconf

✋ Zoom - https://nearform.zoom.us/j/92931518997