

Node.js and JavaScript at the Edge

The why, what and how



About Michael Dawson



Node.js lead for Red Hat and IBM

Active Node.js community member

Node.js Collaborator, Node.js Technical Steering Committee,

Active in a number of Working group(s)

Active OpenJS Foundation member

Voting Cross Project Council Member

Community Director 2020-2022



Twitter: @mhdawson1

GitHub: @mhdawson

Linkedin: <https://www.linkedin.com/in/michael-dawson-6051282>



Overview

- Why Node.js at the Edge
- Running Node.js applications on the edge with RHEL/Fedora
 - The Hardware
 - The Software
- Containerizing your Node.js applications at the edge on RHEL/Fedora
- Advanced container management at the edge for Node.js applications

Why Node.js at the edge

- Existing C/C++ Developers reaching for “just one more language”
- Evolution of Edge devices

My journey through building an edge application

- What hardware?
- What Operating System
- How do I build my application
- How to I package and deploy my application
- How do update my application
- How do I monitor and manage my application

What Hardware?

- Often ARM base for small size and low power
- Raspberry PI ok for experimentation
 - Supported by Fedora IoT
- More robust/commercially supported H/W often used for production deployments



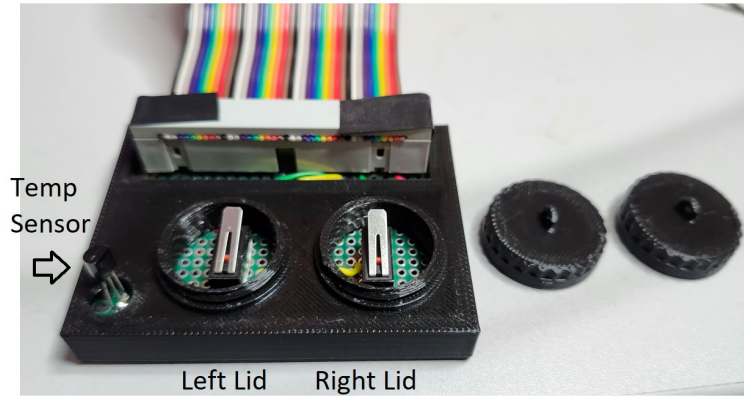
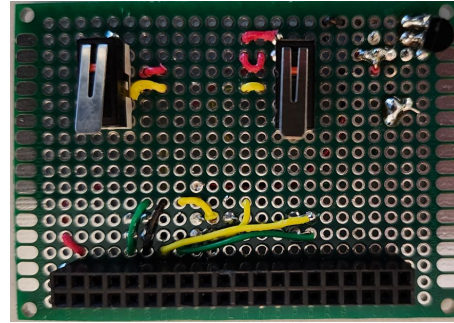
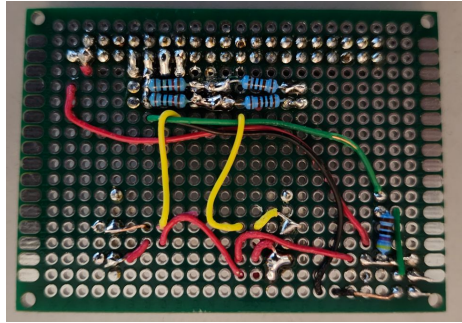
Hardware

Model name	fitlet2
CPU model name	Intel® Atom x5-E3930 [CE3930]
Memory capacity	4.00 GiB

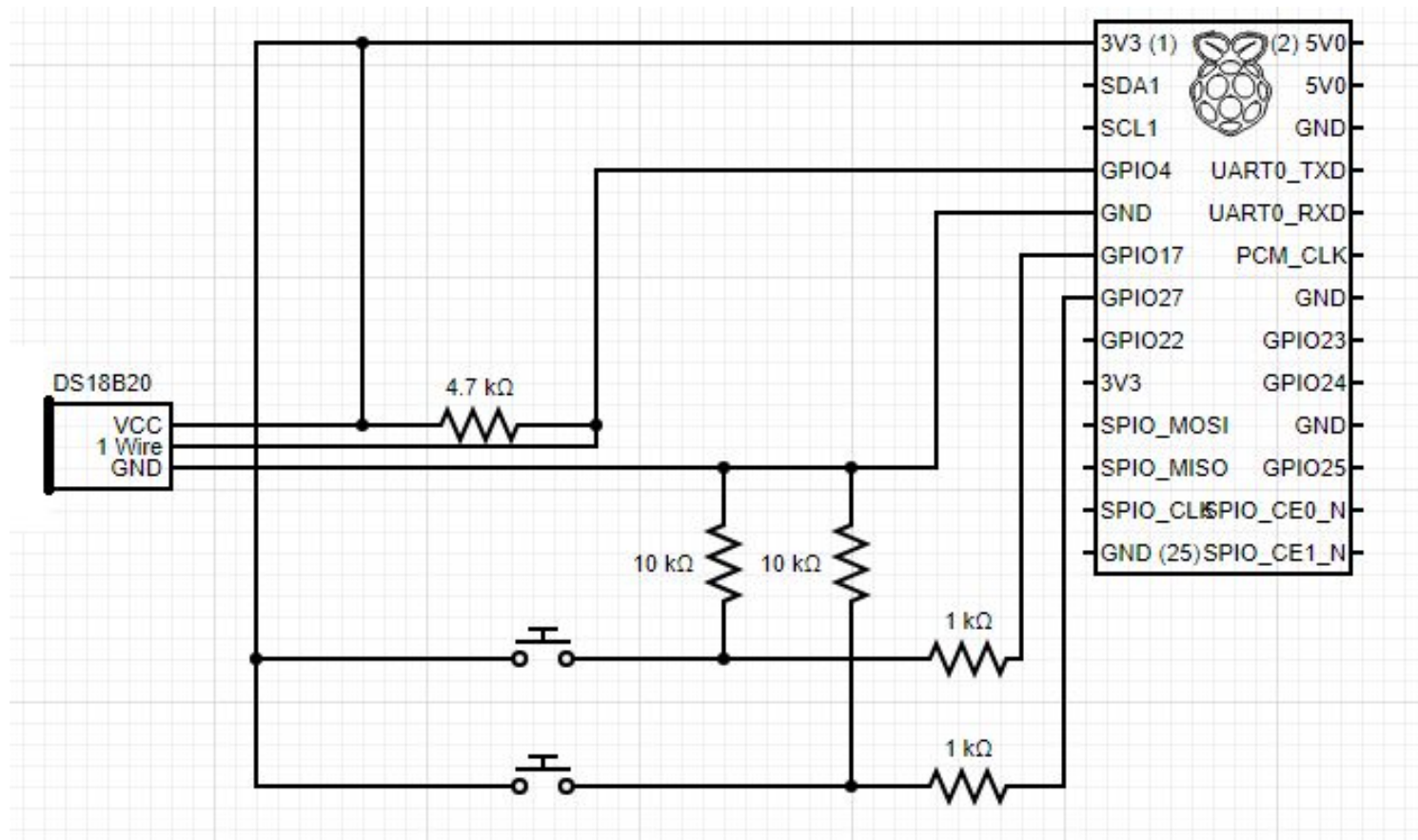


[Install RHEL for Edge on Compulab Fitlet 2](#)

A look at the hardware for our example



Hardware - Circuit



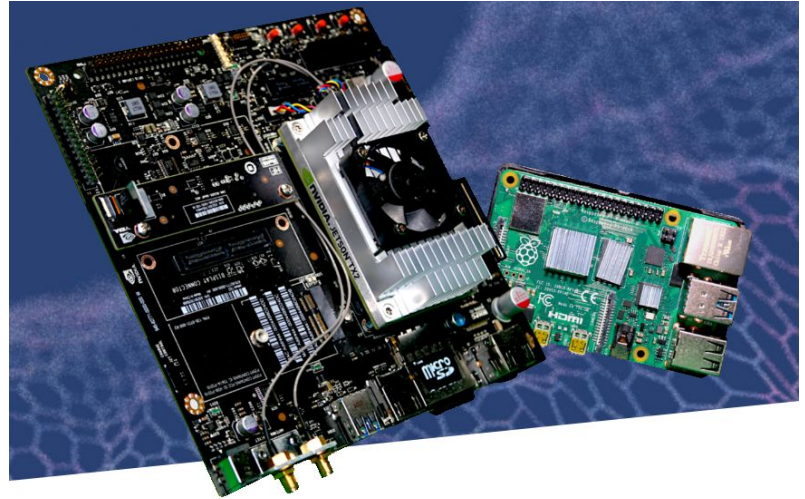
Hardware - Live View

The software

- Operating System
- Device Drivers
- Application

The Operating System

- [Fedora IoT](#)
- [RHEL for Edge](#)



The Operating System

- Key Attributes
 - Immutable
 - Image based
 - Atomic updates
 - Change on reboot/rollback
- [rpm-ostree](#)

The Operating System

Image Builder - <http://X1.X2.X3.X4:9090>

The screenshot shows the Image Builder web interface in a browser. The address bar indicates the URL is <https://10.1.1.104:9090/composer>. The interface has a dark sidebar on the left with navigation links: user1@localhost, Search, Apps, Image Builder (selected), System, Overview, Logs, Storage, Networking, Accounts, and Services. The main content area has a top bar with 'Administrative access', 'Help', and 'Session' menus. Below this is a search bar and three buttons: 'Import blueprint', 'Create blueprint', and 'Create image'. The 'Blueprints' tab is active, displaying a table of blueprints. The table has columns for Name, Version, Last image creation, Images, and Packages. There are four blueprints listed: fedora-base, fedora-base-iot, fedora-iot-microshift, and fedora-microshift. Each row has three buttons: 'Create image', 'Export blueprint', and 'Delete blueprint'.

Name	Version	Last image cre...	Images ...	Packages
fedora-base	0.0.0	-	0	2
fedora-base-iot	0.0.4	Dec 7, 2023	1	0
fedora-iot-microshift	0.0.1	-	0	2
fedora-microshift	0.0.2	Dec 8, 2023	1	2

Blueprints

```
name = "fedora-base"  
description = "base template for Node.js edge example"  
version = "0.0.4"  
modules = []  
groups = []  
distro = ""
```

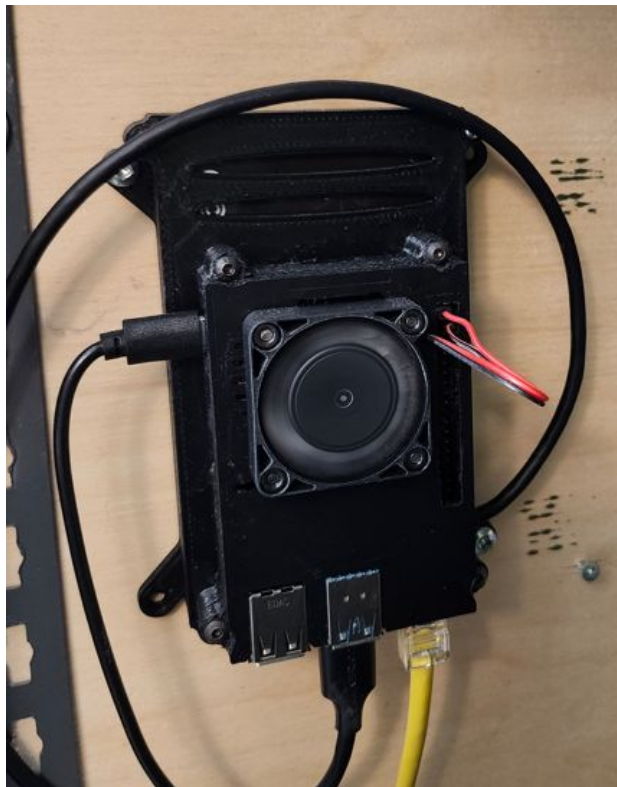
```
[[packages]]  
name = "nodejs20"
```

```
[[packages]]  
name = "podman"
```

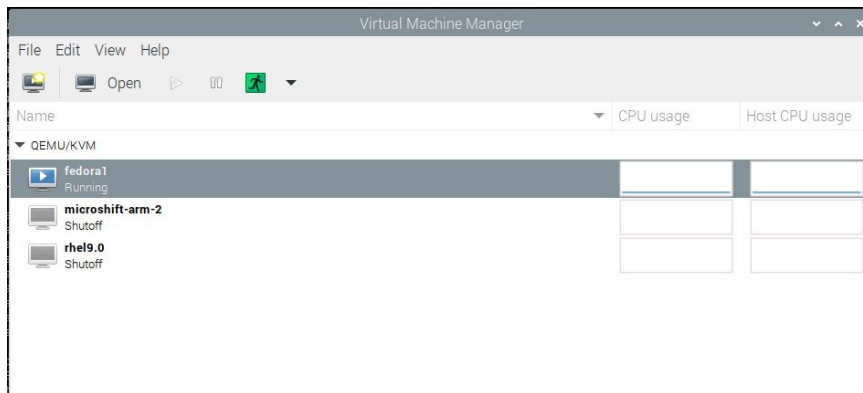
```
[customizations]  
[customizations.timezone]  
[customizations.locale]  
[customizations.firewall]  
ports = ["3000:tcp"]  
[customizations.firewall.services]  
enabled = ["http", "https", "ntp", "dhcp", "ssh"]  
disabled = ["telnet"]  
[customizations.services]  
enabled = ["sshd"]
```



Development system



- Raspberry Pi 4 - 8 G
- 512G SSD
- KVM
 - Easy spin up/down of virtual machines

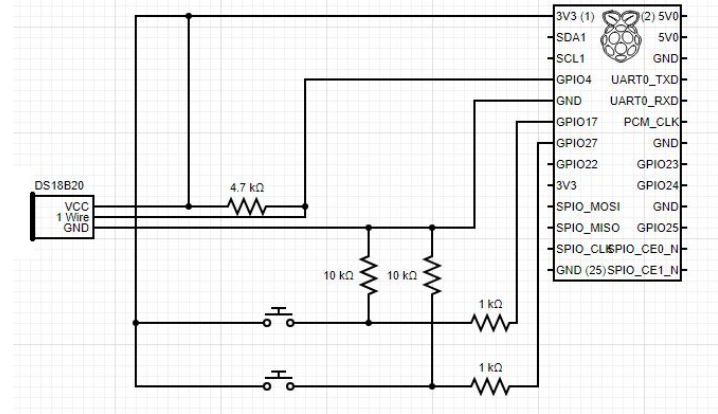


Creating the SD card

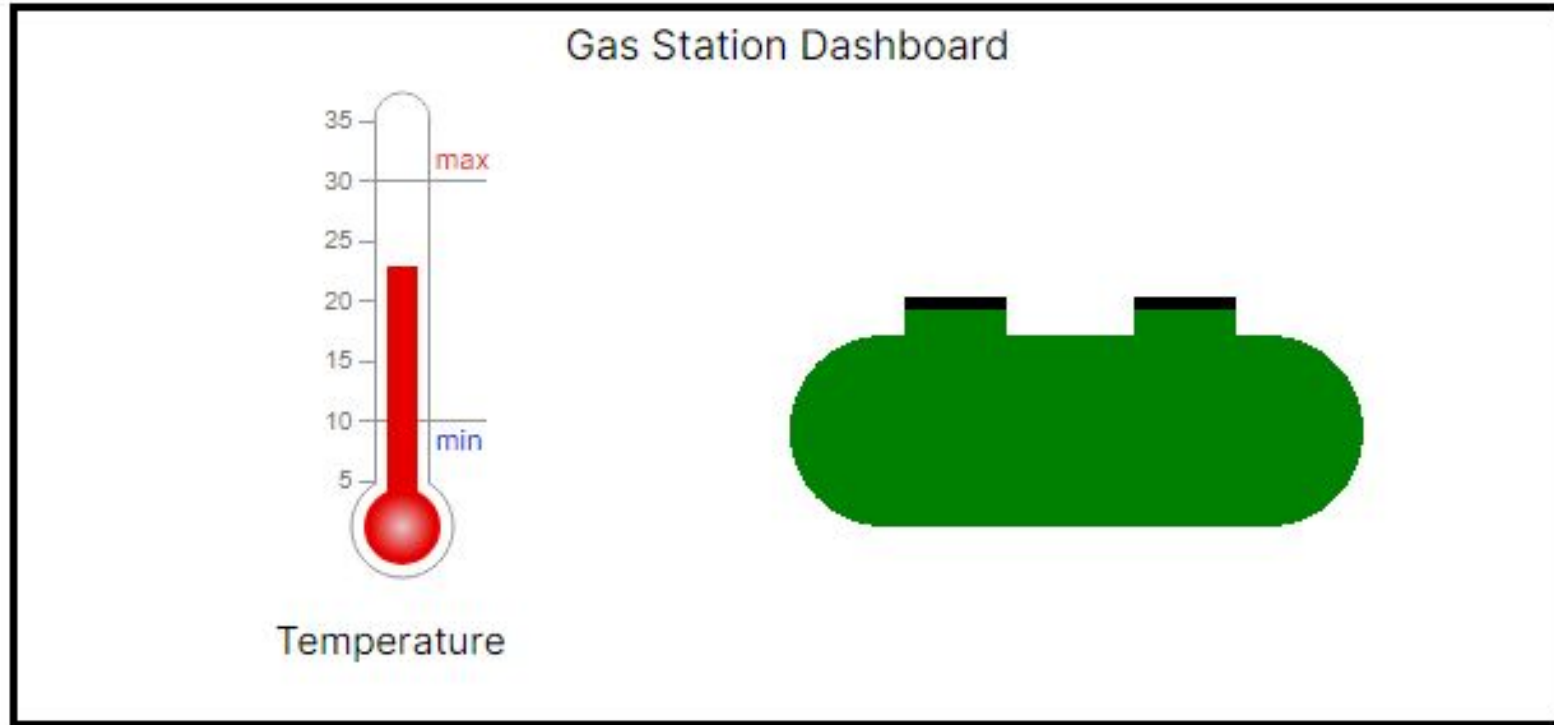
- Building the edge commit and starting a container which serves that commit
- Building a raw image
- Burning the raw image to an SD card

Device Drivers

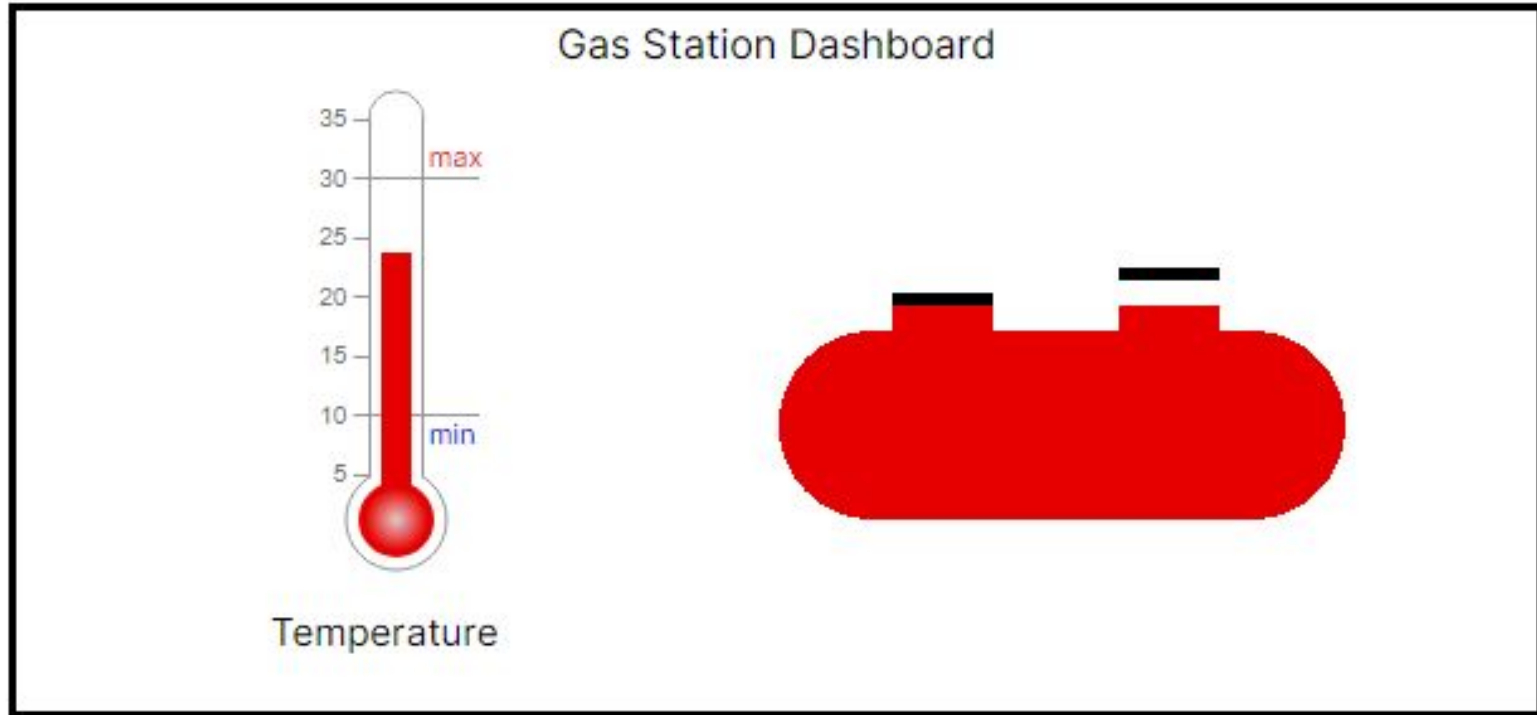
- GPIO
 - gpioget
 - /dev/gpiochip0
 - chmod to make accessible
 - /boot/efi/config.txt
 - dtoverlay=w1-gpio
- Temperature Sensor
 - DS18B20
 - kernel-modules-extra



The application



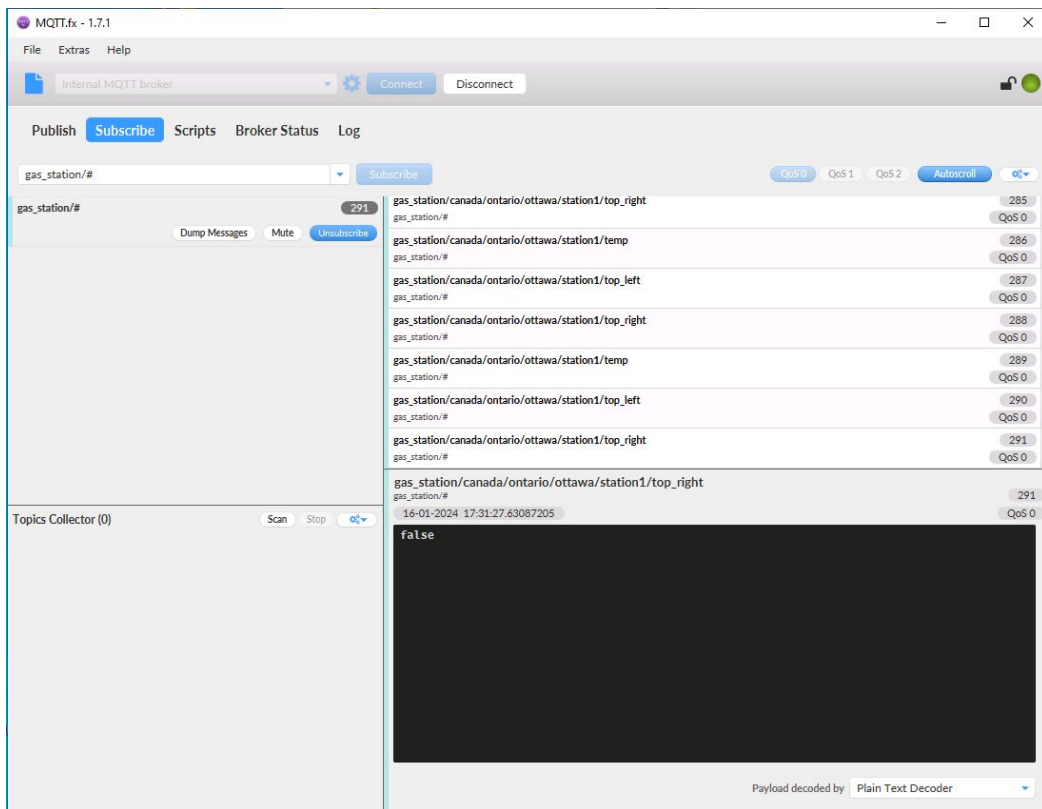
The application



The application

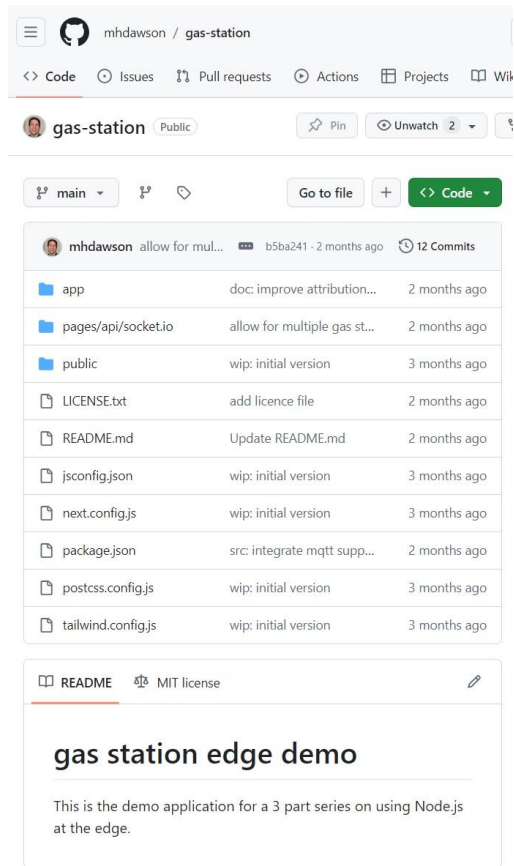
Publishing to MQTT

- gas_station/XXX/temp
- gas_station/XXX/top_left
- gas_station/XXX/top_right



The application - Live View

The application - Next.js based application



The screenshot shows the GitHub repository page for 'gas-station' by user 'mhdawson'. The repository is public and has 12 commits. The file list includes:

File	Commit Message	Time
app	doc: improve attribution...	2 months ago
pages/api/socket.io	allow for multiple gas st...	2 months ago
public	wip: initial version	3 months ago
LICENSE.txt	add licence file	2 months ago
README.md	Update README.md	2 months ago
jsconfig.json	wip: initial version	3 months ago
next.config.js	wip: initial version	3 months ago
package.json	src: integrate mqtt supp...	2 months ago
postcss.config.js	wip: initial version	3 months ago
tailwind.config.js	wip: initial version	3 months ago

The README file is selected, showing the title 'gas station edge demo' and the description: 'This is the demo application for a 3 part series on using Node.js at the edge.'

<https://github.com/mhdawson/gas-station>

The application - Next.js based application

```
React.useEffect(() => {  
  // Create a socket connection  
  const socket = io({ path: '/api/socket.io' });  
  
  // Listen for incoming messages  
  socket.on('message', (message) => {  
    if (message.temp)  
      setTemp(message.temp);  
  
    if (message.topLeft !== undefined)  
      setLeft(message.topLeft);  
  
    if (message.topRight !== undefined)  
      setRight(message.topRight);  
  });  
  
  // Clean up the socket connection on unmount  
  return () => {  
    socket.disconnect();  
  };  
}, []);
```


The application - Next.js based application

```
setInterval(() => {
  const temp = readTemp(tempSensorFile);
  readGpio(TOP_LEFT_PIN, (left) => {
    readGpio(TOP_RIGHT_PIN, (right) => {
      io.emit('message', {temp: temp, topLeft: left, topRight: right});
      mqttClient.publish(TEMP_TOPIC, temp.toString());
      mqttClient.publish(TOP_LEFT_TOPIC, left.toString());
      mqttClient.publish(TOP_RIGHT_TOPIC, right.toString());
    });
  });
}, POLL_INTERVAL);
```

The application - Next.js based application

```
function readGpio(pin, callback) {  
  exec('gpioget gpiochip0 ' + pin, (err, stdout, stderr) => {  
    const pinValueString = stdout.replace(/\s+/g, '');  
  
    let pinValue = false;  
    if (pinValueString === '0')  
      pinValue = true;  
  
    callback(pinValue);  
  });  
};
```

The application - Next.js based application

```
function readTemp(tempSensorFile) {  
  if (tempSensorFile) {  
    const valueRead = readFileSync(tempSensorFile);  
    return valueRead/1000;  
  }  
  return 0;  
};
```

Deploying the Application

- Challenges with using rpm-ostree or image builder
 - It requires a different development workflow from how typical hybrid cloud applications are delivered.
 - It results in a tight binding to operating system components, versions and dependencies, beyond those needed by the code integrating the devices.
 - Potential conflicts between different pieces of an application in terms of operating system component versions.
 - A reboot would be required each time the application was updated

Deploying the Application

- We already have existing flows to build/package cloud native applications
 - Dockerfiles -> Containers

Containerizing the Application

```
# Install dependencies only when needed
FROM registry.access.redhat.com/ubi8/nodejs-20 AS deps
USER 0
WORKDIR /app

# Install dependencies based on the preferred package manager
COPY package.json yarn.lock* package-lock.json* pnpm-lock.yaml* ./
RUN \
  if [ -f yarn.lock ]; then yarn --frozen-lockfile; \
  elif [ -f package-lock.json ]; then npm ci; \
  elif [ -f pnpm-lock.yaml ]; then yarn global add pnpm && pnpm i; \
  else echo "Lockfile not found." && exit 1; \
  fi

# Rebuild the source code only when needed
FROM registry.access.redhat.com/ubi8/nodejs-20 AS builder
USER 0
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

# Next.js collects completely anonymous telemetry data about general usage.
# Learn more here: https://nextjs.org/telemetry
# Uncomment the following line in case you want to disable telemetry during the build.
ENV NEXT_TELEMETRY_DISABLED 1

# If using yarn uncomment out and comment out npm below
# RUN yarn build

# If using npm comment out above and use below instead
RUN npm run build
```

<https://github.com/mhdawson/gas-station/blob/main/Dockerfile>

Containerizing the Application

```
ENV NODE_ENV production
# Uncomment the following line in case you want to enable telemetry during runtime.
ENV NEXT_TELEMETRY_DISABLED 1

COPY --from=builder /app/public ./public

# Set the correct permission for prerender cache
RUN mkdir .next
RUN chown 1001:1001 .next

# Automatically leverage output traces to reduce image size
# https://nextjs.org/docs/advanced-features/output-file-tracing
COPY --from=builder --chown=1001:1001 /app/.next/standalone ./
COPY --from=builder --chown=1001:1001 /app/.next/static ./next/static

# add in libpod for access to Raspberry PI IO pins
RUN rpm -ivh https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
RUN microdnf install libgpiod-utils

USER 1001

EXPOSE 3000

ENV PORT 3000

CMD ["node", "server"]
```

Building the containers

```
podman build . -t gas-station:latest
```

```
podman build . --arch=arm64 -t gas-station:latest
```

```
[user1@localhost gas-station]$ podman images |grep gas-station  
localhost/gas-station          latest  
30ded1b2fcb5    2 hours ago    295 MB
```


Podman/Podman Desktop

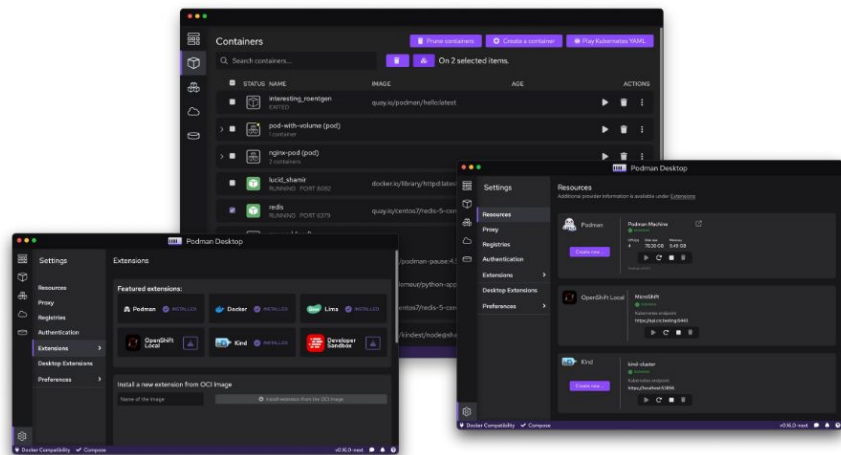
Containers and Kubernetes for application developers

Podman Desktop is an open source graphical tool enabling you to seamlessly work with containers and Kubernetes from your local environment.

Download Now

For Windows (browser-detected)

[Other downloads](#)



Getting the container onto the edge device

```
name = "fedora-base"  
description = "base template for Node.js edge example"  
version = "0.0.4"  
modules = []  
groups = []  
distro = ""
```

```
[[packages]]  
name = "nodejs20"
```

```
[[packages]]  
name = "podman"
```

```
[customizations]  
[customizations.timezone]  
[customizations.locale]  
[customizations.firewall]  
ports = ["3000:tcp"]  
[customizations.firewall.services]  
enabled = ["http", "https", "ntp", "dhcp", "ssh"]  
disabled = ["telnet"]  
[customizations.services]  
enabled = ["sshd"]
```

Getting the container onto the edge device

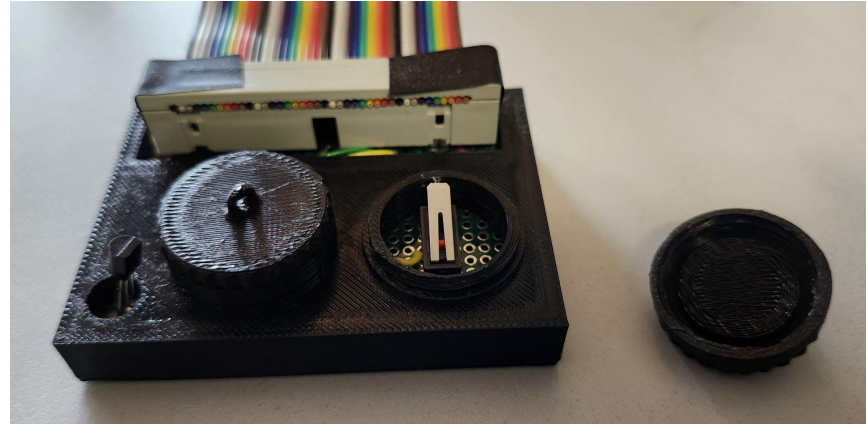
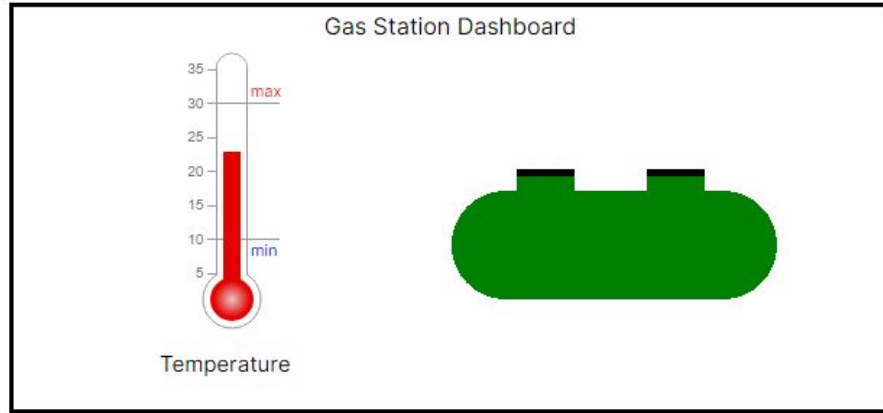
```
podman tag 30ded1b2fcb5 quay.io/midawson/gas-station:latest
```

```
podman push quay.io/midawson/gas-station:latest
```

```
podman pull quay.io/midawson/gas-station:latest
```

```
podman run -d -p 3000:3000 quay.io/midawson/gas-station:latest
```

But there are problems



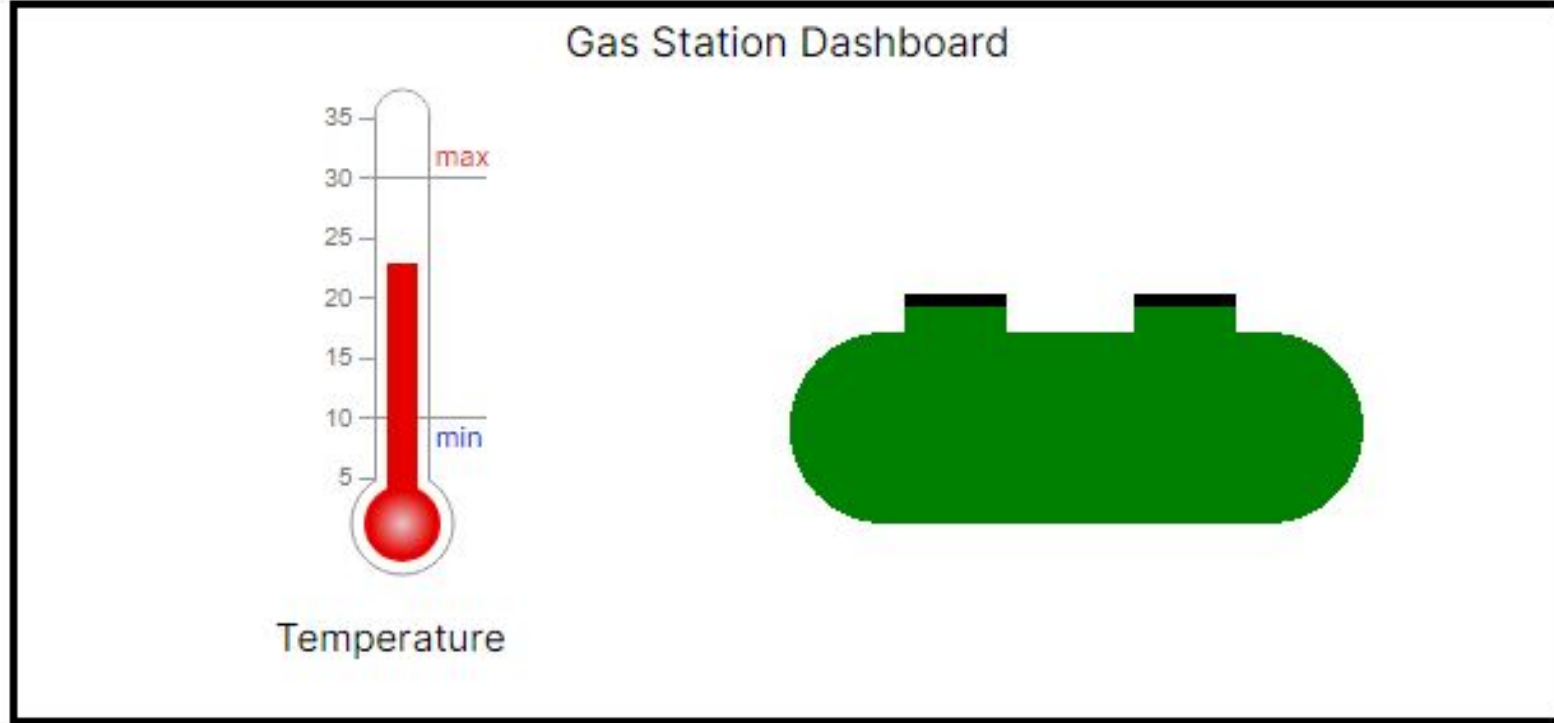
Passing devices into the container

```
podman run --device=/dev/gpiochip0 -d -p 3000:3000  
quay.io/midawson/gas-station:latest
```

```
podman run --group-add keep-groups --device=/dev/gpiochip0 -d -p  
3000:3000 quay.io/midawson/gas-station:latest.
```

[Using files and devices in Podman rootless containers](#)

Why did the thermometer work in the container?



Starting at Boot

Podman Quadlet

```
/etc/containers/systemd/gas-station.container
```

```
[Service]
```

```
Restart=always
```

```
ExecStartPre=chmod 666 /dev/gpiochip0
```

```
ExecStartPre=modprobe wl-therm
```

```
ExecStartPre=/bin/sleep 30
```

```
[Container]
```

```
ContainerName=gas-station
```

```
Environment=STATION_TOPIC_PATH=gas_station/ottawa/bank-street
```

```
Image=quay.io/midawson/gas-station:latest
```

```
Label="io.containers.autoupdate=image"
```

```
PublishPort=3000:3000
```

```
AddDevice=/dev/gpiochip0
```

```
PodmanArgs=--group-add keep-groups
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Starting at Boot

```
systemctl stop gas-station
systemctl start gas-station
systemctl restart gas-station
```

```
bash-5.2# systemctl status gas-station
```

```
• gas-station.service
   Loaded: loaded (/etc/containers/systemd/gas-station.container; generated)
   Drop-In: /usr/lib/systemd/system/service.d
            └─10-timeout-abort.conf
   Active: active (running) since Fri 2024-03-22 19:47:21 UTC; 15min ago
   Process: 29720 ExecStartPre=chmod 666 /dev/gpiochip0 (code=exited, status=0/SUCCESS)
   Process: 29722 ExecStartPre=modprobe wl-therm (code=exited, status=0/SUCCESS)
   Process: 29723 ExecStartPre=/bin/sleep 30 (code=exited, status=0/SUCCESS)
   Main PID: 29857 (conmon)
     Tasks: 12 (limit: 8976)
    Memory: 55.2M
       CPU: 22.441s
    CGroup: /system.slice/gas-station.service
            └─libpod-payload-484377384e684b45377a90e3f5d220684145d81f1c8bfafb66efabd4d340f108
              └─29859 next-server
                └─runtime
                  └─29857 /usr/bin/conmon --api-version 1 -c 484377384e684b45377a90e3f5d220684145d81f1c8bfafb66efabd4d340f108
-u 484377384e684b45377a90e3f5d220684145d81f1c8bfafb66efabd4d340f108 -r /usr/bin/>
```

```
Mar 22 19:47:21 localhost.localdomain podman[29774]: 2024-03-22 19:47:21.289284965 +0000 UTC m=+0.844523639 container init
484377384e684b45377a90e3f5d220684145d81f1c8bfafb66efabd4d340f108 (image=quay.io/>
```


Starting at Boot

```
/etc/containers/systemd/gas-station.container
```

```
[Service]
```

```
Restart=always
```

```
ExecStartPre=chmod 666 /dev/gpiochip0
```

```
ExecStartPre=modprobe wl-therm
```

```
ExecStartPre=/bin/sleep 30
```

```
[Container]
```

```
ContainerName=gas-station
```

```
Environment=STATION_TOPIC_PATH=gas_station/ottawa/bank-street
```

```
Image=quay.io/midawson/gas-station:latest
```

```
Label="io.containers.autoupdate=image"
```

```
PublishPort=3000:3000
```

```
AddDevice=/dev/gpiochip0
```

```
PodmanArgs=--group-add keep-groups
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Starting at Boot

The screenshot shows the MQTT.fx 1.7.1 application window. The interface includes a menu bar (File, Extras, Help), a toolbar with 'Internal MQTT broker', 'Connect', and 'Disconnect' buttons, and a status bar with a lock icon. The main area is divided into several sections:

- Top Bar:** Contains 'Publish', 'Subscribe' (highlighted), 'Scripts', 'Broker Status', and 'Log' buttons.
- Subscription Filter:** A text input field containing 'gas_station/#' and a 'Subscribe' button.
- Message List:** A table of received messages with columns for the topic, message ID, and QoS. The message 'gas_station/gas_station/ottawa/bank-street/top_left' is circled in red.
- Message Details:** A section on the left for the selected message, showing 'gas_station/#' and buttons for 'Dump Messages', 'Mute', and 'Unsubscribe'.
- Topics Collector:** A section at the bottom left showing 'Topics Collector (0)' with 'Scan', 'Stop', and a dropdown menu.

Topic	Message ID	QoS
gas_station/gas_station/ottawa/bank-street/top_right	138	QoS 0
gas_station/gas_station/ottawa/bank-street/temp	139	QoS 0
gas_station/gas_station/ottawa/bank-street/top_left	140	QoS 0
gas_station/gas_station/ottawa/bank-street/top_right	141	QoS 0
gas_station/gas_station/ottawa/bank-street/temp	142	QoS 0
gas_station/gas_station/ottawa/bank-street/top_left	143	QoS 0
gas_station/gas_station/ottawa/bank-street/top_right	144	QoS 0
gas_station/gas_station/ottawa/bank-street/top_right	144	QoS 0

The selected message (ID 143) has the following details:

- Topic: gas_station/gas_station/ottawa/bank-street/top_left
- Message ID: 22-03-2024 15:56:38.57398249
- QoS: 0
- Message Content: true

Updating the application

- 1) building a new version of the application and pushing it to the registry with
podman push [quay.io/midawson/gas-station:latest](https://quay.io/repository/midawson/gas-station/latest)
- 2) logging into the device and running:

```
podman pull quay.io/midawson/gas-station:latest  
systemctl restart gas-station
```

Updating the application

```
/etc/containers/systemd/gas-station.container
```

```
[Service]
```

```
Restart=always
```

```
ExecStartPre=chmod 666 /dev/gpiochip0
```

```
ExecStartPre=modprobe wl-therm
```

```
ExecStartPre=/bin/sleep 30
```

```
[Container]
```

```
ContainerName=gas-station
```

```
Environment=STATION_TOPIC_PATH=gas_station/ottawa/bank-street
```

```
Image=quay.io/mldawson/gas_station:latest
```

```
Label="io.containers.autoupdate=image"
```

```
PodmanArgs=--group-add keep-groups
```

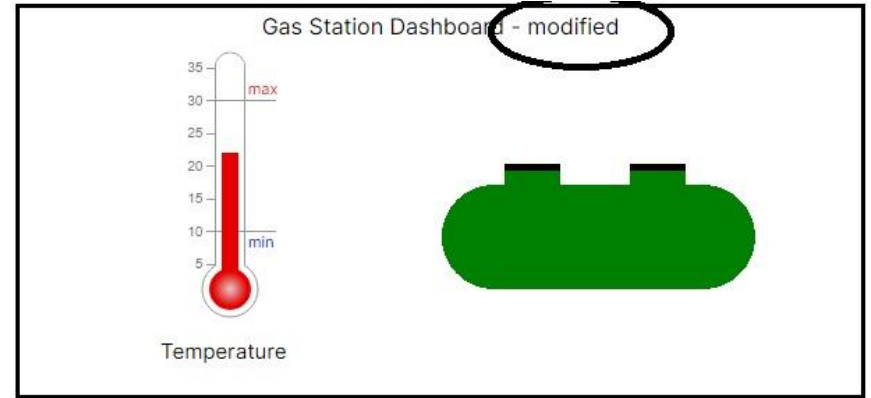
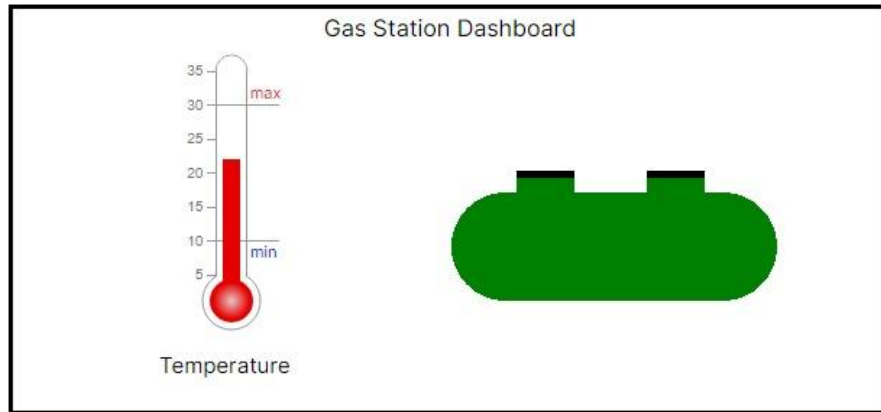
```
AddDevice=/dev/gpiochip0
```

```
PodmanArgs=--group-add keep-groups
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Updating the application



Fully configured SD Card - Updated blueprint

```
name = "fedora-base-container"
description = "base container launch template for Node.js edge example"
version = "0.0.1"
modules = []
groups = []
distro = ""
```

```
[[packages]]
name = "podman"
```

```
[[packages]]
name = "kernel-modules-extra"
```

```
[customizations]
[customizations.timezone]
[customizations.locale]
[customizations.firewall]
ports = ["3000:tcp"]
[customizations.firewall.services]
enabled = ["http", "https", "ntp", "dhcp", "ssh"]
disabled = ["telnet"]
[customizations.services]
enabled = ["sshd"]
```



Fully configured SD Card

```
[[customizations.files]]
path = "/etc/containers/systemd/gas-station.container"
user = "root"
group = "root"
mode = "644"
data = "[Service]\nRestart=always\nExecStartPre=chmod 666 /dev/gpiochip0\nExecStartPre=modprobe
w1-therm\nExecStartPre=/bin/sleep
30\n\n[Container]\nContainerName=gas-station\nEnvironment=STATION_TOPIC_PATH=gas_station/ottawa/bank-street\nImage=quay.io/midawson/gas-station:latest\nLabel=\"io.containers.autoupdate=image\"\nPublishPort=3000:3000\nAddDevice=/dev/gpiochip0\nPodmanArgs=--group-add keep-groups\n\n[Install]\nWantedBy=multi-user.target\n"
```

Advanced container management

- [Red Hat Advance Cluster Management for Kubernetes](#)
- [Microshift](#)
- [Red Hat Device Edge with Microshift](#)

Advanced container management

OpenShift Local cluster is for development and testing purposes. DON'T use it for production.

☰

Red Hat OpenShift

All Clusters ▾

Home >

Infrastructure >

Applications

Governance

Credentials

Clusters ⓘ

Cluster listCluster setsCluster poolsDiscovered clusters

Get started with Multicluster Hub

☐ ▾

Search

Filter ▾

Create cluster

Import cluster

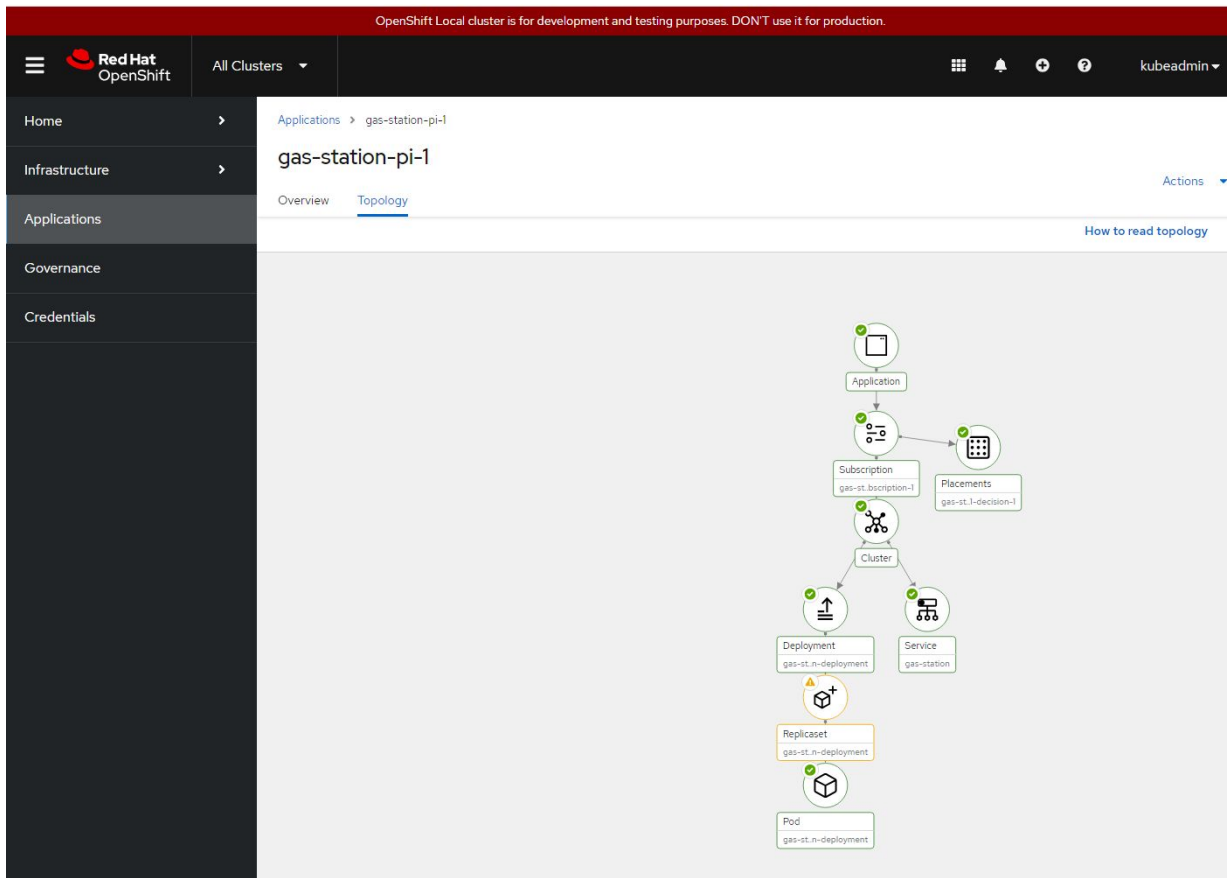
Actions ▾

1 - 3 of 3 ▾ < >

Name ↑ ⓘ	Namespace ↑ ⓘ	Status ↑	Infrastructure ↑	Control plane type	Distribution version ↑	Labels	Nodes ↑	Add-ons ↑	Creation date ↑
<input type="checkbox"/> gas-station-pi-4	gas-station-pi-4	✓ Ready	Red Hat Device Edge	Standalone	MicroShift 4.14.5	microshiftVersi... 13 more	✓ 1	✓ 9	4/10/2024, 8:33:54 PM ⓘ
<input type="checkbox"/> local-cluster	local-cluster	✓ Ready	Other	Hub	OpenShift 4.15.3 ⓘ Upgrade available	openshiftVersi... openshiftVersi... velero.io/exclu... 16 more	✓ 1	✓ 9	4/9/2024, 8:31:43 AM ⓘ
<input type="checkbox"/> microshift-new-1	microshift-new-1	✓ Ready	Red Hat Device Edge	Standalone	MicroShift 4.14.20	microshiftVersi... 13 more	✓ 1	✓ 9	4/9/2024, 9:20:07 PM ⓘ

1 - 3 of 3 items ▾ << < 1 of 1 page > >>

Advanced container management



Advanced container management

```
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2#  
bash-5.2# oc get pods  


| NAME                                    | READY | STATUS  | RESTARTS | AGE |
|-----------------------------------------|-------|---------|----------|-----|
| gas-station-deployment-5df5dbf989-dgsk4 | 1/1   | Running | 2        | 15h |

  
bash-5.2#
```

To dive into more details

- Three part blog post series:
 - [Run Node.js applications on the edge with RHEL and Fedora](#)
 - [Containerizing your Node.js applications at the Edge on RHEL/Fedora](#)
 - Advanced container management at the Edge for Node.js applications (coming soon)

Q/A and Discussion

Copyright and Trademarks

© Red Hat, IBM. All Rights Reserved

Red Hat, the Red Hat logos are trademarks or registered trademarks of Red Hat

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corp.,

registered in many jurisdictions worldwide.

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.