Deployment on kubernetes with 1 simple container service

Kompose convert
Kompose up
Kubectl create -f weather.yml

//Verify
Kubectl get pods
Kubectl get pods --show-all

index.js

```
process.env.DEBUG = 'WeatherHost'

const debug = require('debug')('WeatherHost'),
      Config = require('./config'),
      request = require('superagent'),
      HostBase = require('microservice-core/HostBase')

const POLL_TIME = 60 * 5        // in seconds

function ctof(c) {
  return Math.round(c * (9 / 5) + 32)
}

class WeatherHost extends HostBase {
  constructor(zip) {
    const host = Config.mqtt.host,
          topic = Config.mqtt.topic

    debug('constructor', topic, zip)
    super(host, topic + '/' + zip)
    this.zip = zip
    this.poll()
  }

  command() {
    return Promise.resolve()
  }

  pollOnce() {
    return new Promise((resolve, reject) => {
      request
        .get('https://home.nest.com/api/0.1/weather/forecast/' + this.zip)
        .end((err, res) => {
          if (err) {
            reject(err)
```

```
        }
        else {
          try {
            resolve(JSON.parse(res.text))
          }
          catch (e) {
            console.dir(e.stack)
            reject(e)
          }
        }
      })
  })
 }

 async poll() {
  while (1) {
    debug(this.zip, 'polling')
    try {
      const new_state = await this.pollOnce()
      new_state.now.current_temperature = ctof(new_state.now.current_temperature)
      // flatten
      this.state = new_state // { status: JSON.stringify(new_state) }
    }
    catch (e) {
      this.exception(e)
      debug('Exception', e.message, e.stack)
    }
    await this.wait(POLL_TIME * 1000)
  }
 }
}

function main() {
 const hosts = {}

 Config.weather.locations.forEach((zip) => {
  debug('starting', zip)
  hosts[zip] = new WeatherHost(zip)
 })
}

main()
```

docker file

```
FROM node:8
ENV TZ=America/NewJersey
```

```
RUN npm install forever -g && \
        useradd --user-group --create-home --shell /bin/false app
ENV HOME=/home/app
WORKDIR /home/app
COPY . /home/app
RUN cd $HOME && npm install
PORT 80:80
CMD ["npm", "start"]
```

config.js

```
module.exports = {
  mqtt: {
    // host where mqtt server resides
    // You can add an entry to this server's /etc/hosts to point
    // the name 'ha' at the mqtt server, or set the MQTT_HOST env
    // variable with your mqtt connect string.
    host:  process.env.MQTT_HOST || 'mqtt://ha',
    // This is the base of the topic used to publish/subscribe.
    // For example, autelis/# (on the client) to listen to all updates
    // or autelis/jets to listen for events about the Spa jets.
    // Client can turn on the jets with
    //      topic: autelist/jets
    //      message: on
    //
    topic: process.env.topic || 'weather',
  },
  weather: {
    locations: process.env.WEATHER_LOCATIONS ?
process.env.WEATHER_LOCATIONS.split(',') : [
      '08851',    // NJ
    ]
  },
}
```

100 servers

```
cat servers.txt
Server 1 <IP address 1>
Server2 <IP address 2>
.
.
.
<EOF>
```

vi sshtrigger1.sh

```
while read -r ip;do
  ssh-copy-id -i .ssh/id_rsa.pub $ip
done < servers.txt
```