

IBM Node-RED IBM Cloud template 2020. Запуск локально

LAB-0-03 - Запуск хмарного шаблону локально

- 1 [Вступ](#)
- 2 [Призначення каталогів](#)
- 3 [Призначення файлів в кореновому каталозі](#)
- 4 [Запуск нашого app локально з Node.js під локальною ОС](#)
- 5 [Запуск нашого app локально в контейнері](#)
- 6 [Анатомія локального запуску](#)

Вступ

Документація написана з використанням загально-прийнятого формату markdown (файли типу *.md). Короткий довідник по markdown знаходиться по лінку [markdown help](#).

Лабораторні роботи розраховані на роботу з OS windows-10

Призначення каталогів

- **defaults** Тут розміщується файл з наперед розробленими потоками обробки з найменуванням **flow.json**. Якщо маємо кореспондуючий файл з credentials, то їх розміщують ту же в файлі **flow_cred.json**. defaults/readme.md
- **nodes** Тут розміщуються власні додаткові вузли обробки
Щоб додати додаткові вузли потрібно:
 - скопіювати їх в цей каталог та додати їх залежності в **../package.json**
 - додати їх **npm package name** в **../package.json** nodes/readme.md
- **public** Тут знаходиться головна запускаюча сторінка додатку. Основні файли:
 - first-run.html - діалог першого запуску
 - index.html - відома червона сторінка Node-RED
- **routers** Тут знаходиться звичайний node.js-express роутер, що використовується для контролю живучості app. Використовується при запуску в kubernetes та OpenShift.

```
const express = require('express');
module.exports = function (app) {
  const router = express.Router();
  router.get('/', function (req, res, next) {
```

```

    res.json({
      status: 'UP'
    });
  });
  app.use('/health', router);
}

```

- **server** Тут можна розмістити додаткові сервіси та класи. Також, тут є вкладений каталог **config**
 - **config**
в ньому розміщується конфігураційний файл **/config/mappings.json**, в якому прописується правила доступу до env variables, через які виконується зв'язування з іншими сервісами. Також, при необхідності підкладається файл **server/localdev-config.json** в якому зберігаються credentials при запуску на локальній станції. Цей файл стоїть в .gitingore і не повинен попадати в SourceControl.

server/localdev-config.json

```

{
  "cloudant_apikey": "xu9KREDF",
  "cloudant_host": "xxxx-bluemix.cloudantnosqldb.appdomain.cloud",
  "cloudant_iam_apikey_description": "Auto-generated for key a659120d",
  "cloudant_iam_apikey_name": "153348d7",
  "cloudant_iam_role_crn": "crn:v1:bluemix:public:iam",
  "cloudant_iam_serviceid_crn": "crn:v1:bluemix:serviceid:ServiceId",
  "cloudant_password": "a4e74b446d0e9d17a998902035cbf24",
  "cloudant_port": 443,
  "cloudant_url": "https://user:token@xxx-
bluemix.cloudantnosqldb.appdomain.cloud",
  "cloudant_username": "user"
}

```

Приклад опису доступу з файлу **/config/mappings.json** для реквізиту **cloudant_url**

```

"cloudant_url": {
  "searchPatterns": [
    "user-provided:node-red-app-cloudant-1580722484242-67617:url",
    "cloudfoundry:['cloudantNoSQLDB'][0].credentials.url",
    "env:service_cloudant:$.url",
    "env:cloudant_url",
    "file:/server/localdev-config.json:$.cloudant_url"
  ]
}

```

Доступ з файла: "file:/server/localdev-config.json:\$.cloudant_url" В хмарі CloudFoundry параметр вичитується з "cloudfoundry:['cloudantNoSQLDB'][0].credentials.url"

Призначення файлів в кореневому каталозі

- **package.json**

Основний файл з реєстром node.js пакетів та бібліотек. Наповнюється через `npm install`

- **bluemix-settings.js** Модуль в якому вичитуються конфігурації додатку. порти, папараметри cloudant. До нього звертається **index.js** при старті
- **index.js** Головний файл, являє собою звичайний Node.js express сервер.
- **cloudantStorage.js** Модуль роботи з базою даних Cloudant
- **.cfignore** Файл відноситься до CloudFoundry deployment. Це список файлів та каталогів які ігноруються при deployment в cloudfoundry.
- **manifest.yml** Конфігурація cloudFoundry application при deployment.
- **cli-config.yml** Конфігурація ibmCloud CLI. До речі дуже цікавий файл. Потрібно з ним розібратися детальніше.
- **run-debug** Для запуску в режимі remote debug (не працює, здається переїхав з минулої версії node.js app шаблону). Дуже схоже, що зараз його функції виконує **cli-config.yml**
- **run-dev** Для запуску в режимі розробника (не працює, здається переїхав з минулої версії node.js app шаблону). Дуже схоже, що зараз його функції виконує **cli-config.yml**
- **README.md** описовий
- **CONTRIBUTING.md** описовий
- **DCO1.1.txt** описовий
- **LICENSE** описовий

Запуск нашого app локально:

Існує 2 варіанту запуску нашого app локально

- традиційний і класичний спосіб, використовуючи команду

```
npm install  
npm start
```

Цей скрипт запускає команду:

```
node --max-old-space-size=160 index.js --settings ./bluemix-settings.js -v
```

При цьому application буде доступне по `http://localhost:1880/`

Запуск нашего app локально в Docker контейнері

Запуск в docker контейнері ділиться на 2 етапи Спершу створюється образ операційного середовища, описаний в файлі: `Dockerfile-tools` Потім уже створюється новий образ шляхом копіювання app кода. Взято від шаблону звичайного nodejs app

Побудова базовго образу (чомусь перестало працювати. Є підозра в сумісності версій Node)

```
ibmcloud dev build
```

Створюється контейнер з прикладним кодом

```
ibmcloud dev run
```

В каталозі `from-rhat-img` дежать Docker файли для запуску контейнерів та образи від RHat В каталозі `from-node10stretch-img` Docker файли з `node:10-stretch`

Якщо команди не спрацьовують, то дають помилку як показано нижче,

Logs for the `nodredwshp-express-run` container:

```
> node-red-app@1.1.1 start /app
> node --max-old-space-size=160 index.js --settings ./bluemix-settings.js -v
internal/modules/cjs/loader.js:1021
    return process.dlopen(module, path.toNamespacedPath(filename));
                                ^Error: /app/node_modules/bcrypt/lib/binding/bcrypt_lib.node:
invalid ELF header
```

то можна використати прямі докер команди

```
## similar Dockerfile-tools
docker image build --file Dockerfile-tools --tag nodredwshp-express-tools --rm
--pull --build-arg bx_dev_userid=0 --build-arg bx_dev_user=root .

## similar Dockerfile
docker image build --file Dockerfile --tag nodredwshp-express-run --rm --pull -
-build-arg bx_dev_userid=0 --build-arg bx_dev_user=root .

## run container
docker run -p 1880:18806 -p3000:3000 nodredwshp-express-run
```

Анатомія локального запуску

Якщо Node-RED стартує локально, то створює каталог

```
.node-red
```

Щоб не передавати його в контейнер та git цей каталог потрібно додати в так звані ignore-файли.

Приклади файлів показані в **.gitignore**, **.dockerignore**, **.cfigure**.

При цьому, якщо не конфігурувати env variables то система вважає, що потоки обробки зберігаються в локальному файлі

```
PS C:\PSHDEV\PSH-WorkShops\IBM-Node-Red-APP2020\Lab0-02-app\nod-red-wshp> npm
start
> node-red-app@1.1.1 start C:\PSHDEV\PSH-WorkShops\IBM-Node-Red-APP2020\Lab0-02-
app\nod-red-wshp
> node --max-old-space-size=160 index.js --settings ./bluemix-settings.js -v

9 Feb 13:35:03 - Starting Node-RED on IBM Cloud bootstrap
9 Feb 13:35:03 - Loading bluemix-settings.js

## !!!!!!!При старті вказано, що Cloudant не знайдений!!!!!!!
9 Feb 13:35:03 - No Cloudant service found
9 Feb 13:35:03 - Falling back to localfilesystem storage. Changes will *not* be
saved across application restarts.
## !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

9 Feb 13:35:03 - Loading application settings
9 Feb 13:35:03 - Starting first-use setup
9 Feb 13:35:03 - Waiting for first-use setup to complete
```

Правила визначення сховища потоків описані в bluemix-settings.js

```
// Identify the Cloudant storage instance the application should be using.
var storageServiceName;
var storageServiceDetails;

if (process.env.NODE_RED_STORAGE_NAME) {
  // A service has been identified by the NODE_RED_STORAGE_NAME env var.
  // - check to see if the named service exists
  storageServiceDetails = appEnv.getService(process.env.NODE_RED_STORAGE_NAME);
  if (!storageServiceDetails) {
    util.log("Failed to find Cloudant service:
"+process.env.NODE_RED_STORAGE_NAME+ " (NODE_RED_STORAGE_NAME)");
  } else {
    storageServiceName = process.env.NODE_RED_STORAGE_NAME
```

```

    }
  } else {
    // No couch service specified by env var - look at the attached services
    var candidateServices = Object.values(appEnv.getServices()).filter(app =>
app.label === "cloudantNoSQLDB");
    if (candidateServices.length === 0) {
      util.log("No Cloudant service found");
    } else {
      // Use the first in the list - but warn if there are multiple incase we
      // are using the 'wrong' one.
      storageServiceName = candidateServices[0].name;
      storageServiceDetails = candidateServices[0];
      if (candidateServices.length > 1) {
        util.log("Multiple Cloudant services found - using
"+storageServiceName+". Use NODE_RED_STORAGE_NAME env var to specify the required
instance.");
      }
    }
  }

  if (!storageServiceName) {
    // No suitable service has been found. Fall back to localfilesystem storage
    util.log("Falling back to localfilesystem storage. Changes will *not* be saved
across application restarts.");
  } else {
    // Set the Cloudant storage module settings
    settings.cloudantService = {
      // The name of the service instance to use.
      name: storageServiceName,
      // The URL to use
      url: storageServiceDetails.credentials.url,
      // The name of the database to use
      db: process.env.NODE_RED_STORAGE_DB_NAME || _sanitizeAppName(appEnv.name),
      // The prefix for all document names stored by this instance.
      prefix: process.env.NODE_RED_STORAGE_APP_NAME ||
_sanitizeAppName(appEnv.name)
    }

    util.log("Using Cloudant service: "+storageServiceName+
db:"+settings.cloudantService.db+" prefix:"+settings.cloudantService.prefix);
    settings.storageModule = require("./cloudantStorage");
  }
}

```

При старті локально з файловим сховищем потоки, що задеплойені зберігаються в файлі:

```

.node-red/flows.json
.node-red/flows_cred.json

```

При цьому, з каталога **default** потоки не переносяться. Крім того, потрібно мати на увазі, що при локальному старті перевіряються тільки логін та пароль і стартує звичайна стандартна NodeRed

бібліотека: **require('./node_modules/node-red/red.js');**

Ось цитата з **index.js**

```
function startNodeRED(config) {
  if (config.adminAuth && !settings.adminAuth) {
    util.log("Enabling adminAuth security - set NODE_RED_USERNAME and
NODE_RED_PASSWORD to change credentials");
    settings.adminAuth = {
      type: "credentials",
      users: function (username) {
        if (config.adminAuth.username == username) {
          return Promise.resolve({ username: username, permissions:
"*" });
        } else {
          return Promise.resolve(null);
        }
      },
      authenticate: function (username, password) {
        if (config.adminAuth.username === username &&
bcrypt.compareSync(password, config.adminAuth.password)) {
          return Promise.resolve({ username: username, permissions:
"*" });
        } else {
          return Promise.resolve(null);
        }
      }
    };
    if ((process.env.NODE_RED_GUEST_ACCESS === 'true') ||
(process.env.NODE_RED_GUEST_ACCESS === undefined &&
config.adminAuth.allowAnonymous)) {
      util.log("Enabling anonymous read-only access - set
NODE_RED_GUEST_ACCESS to 'false' to disable");
      settings.adminAuth.default = function () {
        return Promise.resolve({ anonymous: true, permissions: "read"
});
      };
    } else {
      util.log("Disabled anonymous read-only access - set
NODE_RED_GUEST_ACCESS to 'true' to enable");
    }
  }
  require('./node_modules/node-red/red.js');
}
```

Тобто практично, стартує не залежний екземпляр Node-RED у вигляді бібліотеки. Але є 2 практичні користі:

- Спільні бібліотеки
- можливість переноса потоку під управління git і потім його доставка в хмару.