Робоча чернетка. Автор - Олександр Пупена, доц. ІАСУ НУХТ. Поширюється для тестування і зворотного зв'язку з побажаннями і правками. pupena_san@ukr.net

Лабораторна робота №3. Хмарні сервіси для Індустрії 4.0

Частина 3. Основи роботи з Cloud Foundry в IBM Cloud

Мета: навчитись керувати застосунками Cloud Foundry в IBM Cloud.

Цілі:

1) розібратися з механізмами роботи застосунків Cloud Foundry в IBM Cloud
2) встановити та навчитися працювати з Cloud Foundry Command Line Interface CLI
3) розбіратися з механізмами взаємодії застосунків CF з сервісами IBM Cloud
4) розбіратися з механізмами завантаження стартового пакету «Node-RED Starter»
5) виправити проблему повторного запуску Node-RED Starter

Зовнішній вигляд інтерфейсу налаштування IBM Cloud постійно змінюється! Увага, набір наданих сервісів та політика ліцензування IBM Cloud може змінитися. У будь якому випадку, без використання угоди та реєстрації кредитної картки ніякі кошти за використання стягуватися не можуть.

1. Механізми роботи застосунків Cloud Foundry в IBM Cloud

У частині 3.1 лабораторної роботи для створення застосунку використовувалася платформа Node-RED, яка базуються на SDK Node.js. Створення застосунку проводилося з використанням стартового набору «Node-RED Starter». Це швидкий шлях створення, який не передбачає знання механізмів функціонування застосунку в хмарі, однак у нештатних ситуаціях або при необхідності тонкого налаштування застосунку все ж таки треба розбиратися в основах його функціонування. Ціль даної частини лабораторної роботи отримати базові знання про основи функціонування застосунків Cloud Foundry в IBM Cloud та про їх зв'язки з хмарними сервісами.

SDK Node.js, що використовується в «Node-RED Starter», базується на **Cloud Foundry (CF),** який є провідним галузевим стандартом платформи як послуги (PaaS), що забезпечує швидке, просте та надійне розгортання хмарно-орієнтованих застосунків. Це платформа з відкритим вихідним кодом, яку можна розгортати для запуску своїх застосунків як на власній обчислювальній інфраструктурі, так і на існуючих IaaS (інфраструктура як послуга), таких як AWS, vSphere або OpenStack.

Хмари балансують свої навантаження на обробку на декількох машинах, оптимізуючи ефективність і стійкість до відмови. Установка Cloud Foundry виконує це на трьох рівнях:

1. **BOSH** - це ланцюг інструментів з відкритим кодом для розробки інженерних рішень, розгортання та управління життєвим циклом широкомасштабних розподілених сервісів. Для постачальників послуг Cloud Foundry, BOSH є платформою розгортання та керування за замовчуванням. BOSH створює і розгортає віртуальні машини (VM) поверх фізичної обчислювальної інфраструктури і розгортає і запускає Cloud Foundry поверх неї. Щоб налаштувати розгортання, BOSH слідує документу маніфесту.
2. CF Cloud Controller запускає програми та інші процеси на віртуальних машинах хмари, балансуючи попит і керуючи життєвим циклом застосунків.
3. Маршрутизатор (router) направляє вхідний трафік з усього світу до віртуальних машин, які працюють з застосунками, зазвичай працюючи з балансувачем навантаження, наданим клієнтом.

Основні елементи CF показані на рис.1. Розроблені застосунки виконуються на віртуальних машинах **Diego Cell**. Екземпляри застосунків (Application instances), прикладні задачі (application tasks) і задачі постановки (керування стадіями розгортання, staging tasks) виконуються як контейнери Garden на віртуальних машинах Diego Cell. Контейнер – це власне автономне середовище, в якому ізолюються процеси, пам'ять та файлова система, використовуючи функції операційної системи та характеристики віртуальної та фізичної інфраструктури, де розгортається CF. Ізоляція контейнерів досягається за допомогою використання окремих просторів імен ресурсів ядра (namespace). У кожному контейнері обмежується обсяг пам'яті, яку може використовувати контейнер, і кванти часу роботи CPU, доступ до яких потрібен також іншим контейнерам. Файлова система складається зі спільної для всіх контейнерів базової файлової системи (доступна тільки для читання), та контейнеро-залежної частини (доступний для читання/запису) унікальної для кожного контейнеру, розмір якої обмежений квотами проекту. Додатково про контейнери та їх порівняння з віртуальними машинами можна прочитати за цим посиланням.

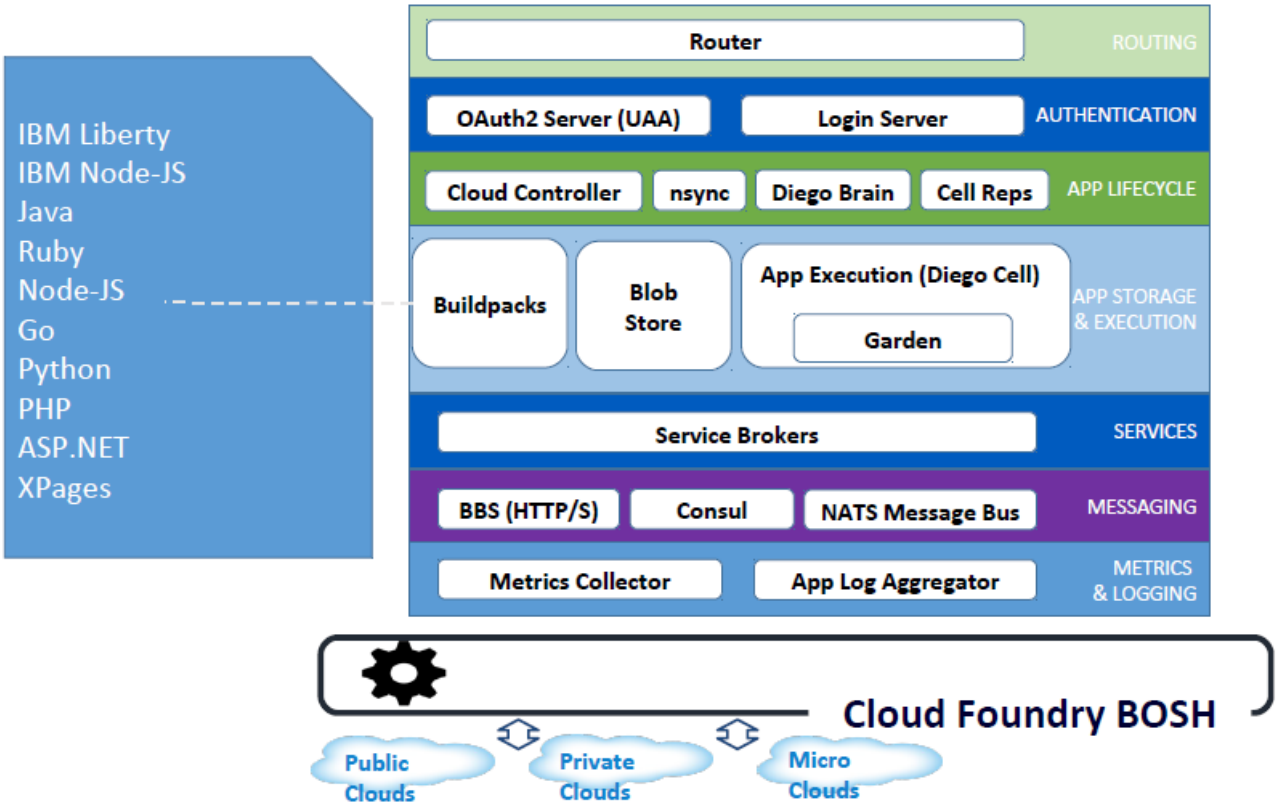

рис.1. Архітектура CF.

Застосунки створюються для конкретної платформи, і вихідний код для них може знаходитися як в локальному сховищі (ПК розробника) так і на GitHub чи зовнішніх сховищах. У Cloud Foundry для контролю версій вихідного коду, зборок (buildpacks), документації та інших ресурсів використовується git-система на GitHub. Однак для зберігання великих двійкових файлів, CF підтримує внутрішнє або зовнішнє блокове зберігання **Blob Store**. У Blobstore міститься наступне:

- пакети вихідного програмного коду
- Buildpacks (збірки)
- Droplets

Пакети вихідного програмного коду створюються для конкретних програмних інструментів з використанням ряду мов. Зокрема, Node-RED базується на Node.js, тому пакет вихідного коду для «Node-RED Starter» розроблений з урахуванням вимог SDK Node.js. Для перетворення цього

сирцевого коду в реальний застосунок використовується Buildpack. **Buildpack (збірка)** – це колекція коду, що відповідає за перетворення сирцевого коду застосунків в готові до запуску бінарні пакети (дроплети). Процеси, що відбуваються при такому перетворенні називаються **постановкою** (**staging**). Збірка містить всі мови, бібліотеки та служби, що використовує застосунок. Перш, ніж надсилати застосунок до віртуальної машини, контролер Хмари (Cloud Controller) перетворює її в образ, поєднуючи стек, buildpack та вихідний код у droplet, які VM може розпакувати, скомпілювати та запустити. **Droplet** (дроплет) – це пакет, що містить усе необхідне для успішного запуску вашого застосунку (наприклад, JRE, Liberty, сам застосунок) однак без урахування операційної системи.

Cloud Controller (Контролер хмари) керує розгортанням застосунків з коду. Процес завантаження коду застосунку в хмару називається **push** (пуш) надалі по тексту буде використовуватися слово «**запушити**». Щоб запушити код застосунку до Cloud Foundry, він спрямовується на Cloud Controller (через Router), який керує окремими комірками Diego для керування етапами та запусками застосунків. Щоб підтримувати доступність застсосунків, хмарні розгортання повинні постійно контролювати свої стани та узгоджувати їх з очікуваними станами, запускаючи та зупиняючи процеси за потребою. Компоненти nsync, BBS і Cell Rep співпрацюють разом по ланцюжку, щоб підтримувати роботу застосунків з необхідною кількістю екземплярів.

Зазвичай програми залежать від таких сервісів, як бази даних або сторонні постачальники SaaS. Коли розробник передбачає та прив'язує сервіс до застсоунку, брокер сервісу (**Service Broker**) для цієї служби несе відповідальність за надання екземпляру сервісу.

Маршрутизатор (**router**) направляє вхідний трафік на відповідний компонент - Cloud Controller, або розміщений застосунок, запущений в комірці Diego. Маршрутизатор періодично надсилає запит до системи дошки оголошень Diego (**BBS**), щоб визначити, в яких комірках і контейнерах зараз працює кожен застосунок. Використовуючи цю інформацію, маршрутизатор перераховує нові таблиці маршрутизації на основі IP-адрес кожної віртуальної машини (VM) і номера порту на стороні хоста для контейнерів комірки.

CF керує обліковими записами користувачів за допомогою двох серверів: автентифікації та авторизації користувачів. Для забезпечення керуванням ідентифікацією разом працюють сервери **OAuth2 server** (the UAA) та **Login Server**. Один сервер UAA надає доступ до BOSH, і має аккаунти для операторів CF, які розгортають середовище виконання, сервіси та інше програмне забезпечення безпосередньо на слой BOSH. Інший сервер UAA керує доступом до Cloud Controller і визначає, хто може йому надавати команди. UAA Cloud Controller означує різні ролі користувачів, такі як адміністратор, розробник або аудитор, і надає їм різні набори привілеїв для запуску CF-команд

Компоненти Cloud Foundry спілкуються двома способами (**Messaging**):

- Надсилаючи повідомлення внутрішньо, використовуючи протоколи HTTP і HTTPS
- Надсилаючи повідомлення NATS один одному безпосередньо

Cloud Foundry створює системні журнали з компонентів Cloud Foundry та журналів програм із розміщених застосунків. Коли працює Cloud Foundry, його компонентні та хост-віртуальні машини генерують журнали та метрики. Застосунки Cloud Foundry також зазвичай генерують журнали. Система Loggregator об'єднує метрики компонентів і журнали застосунків у структуровану, корисну форму Firehose, яку можна переглянути у будь який момент постановки.

IBM Bluemix є рішенням CF поверх IBM Cloud IaaS (рис.2). Доступ до конфігурування, керування та діагностування застосунками може відбуватися як через WEB IDE (консоль), що використовувався в частині 3.1 лабораторної роботи, так і через класичний клієнтський Cloud Foundry Command Line Interface (CLI). Доступне керування життєвим циклом застосунків через утиліти DevOps, також можлива інтеграція з Eclips IDE. В наступному пункті, для керування розгортуванням буде використовуватися утиліта CLI.

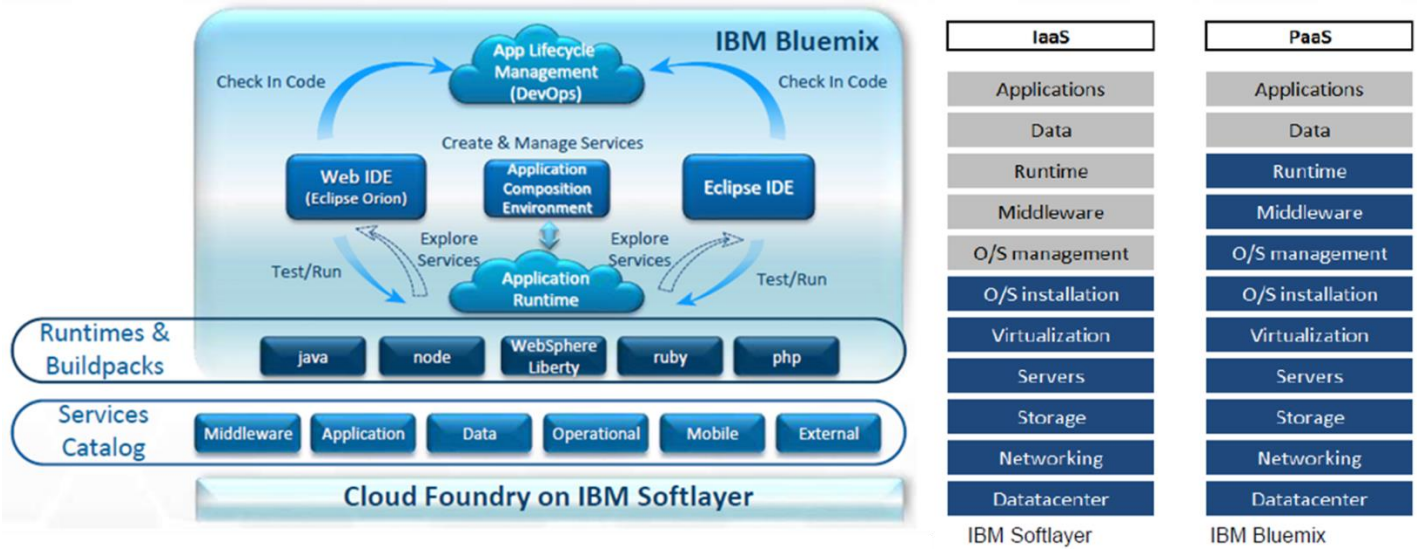

рис.2.Функціонування Cloud Foundry в IBM Cloud

2. Інсталяція та знайомство з CLI (Cloud Foundry Command Line Interface)

2.1. Інсталяція CLI.

Завантажте **Cloud Foundry Command Line Interface** з цієї сторінки . Для ОС Windows 64 bit пряме посилання для завантаження архіву знаходиться за цим посиланням. Для 32-х бітних версій Windows та інших ОС можна скористатися попередніми версіями CLI, зокрема звідси ставиться для Win32.

Розпакуйте архів і запустіть інсталяцію з файлу cf_installer.exe.

2.2. Виведення базових опцій команди CLI.

CLI є консольною утилітою. Запустіть командний рядок (cmd) в якому наберіть команду

cf

Набір cf без аргументів виводить перелік найбільш вживаних глобальних опцій. Перелік всіх опцій наведений за цим посиланням.

2.3. Конфігурування точки доступу консолі до хмари.

Для доступу для керування Cloud Foundry необхідно ввести точку доступу API. Для цього використовується опція api. Для отримання допомоги по даній опції в командному рядку наберіть:

cf api -h

Введення опції передбачає вказівку URL в якості точки введення. Якщо URL не вказаний, команда виводить точку доступу API, яка зараз використовується. Для керування хмарним застосунком Node-RED необхідно підключитися до https://api.eu-gb.bluemix.net, тобто ввести:

cf api https://api.eu-gb.bluemix.net

2.4. Реєстрація користувача.

Після успішного задавання точки доступу необхідно залигуватися за допомогою опції login

cf login

після чого ввести

- ім'я користувача (пошта)

- і пароль (при введенні паролю символи не відображаються)

2.5. Перегляд списку та стану застосунків

Для перегляду стану застосунків введіть команду

cf apps

При нормальному стані список матиме вигляд як на рис.3.

```
C:\Users\2017>cf apps
Getting apps in org other_man@ukr.net / space dev as other_man@ukr.net...
OK

name        requested state   instances   memory   disk   urls
node-red    started           1/1         256M     1G     OtherManNodeRED.eu-gb.mybluemix.net
```

рис.3.

Однак, якщо після минулого запуску застосунок переходив у режим sleep (про це мало повідомлятися поштою), він можливо буде знаходитися в режимі помилки. Як виправити цю ситуацію буде розглянуто нижче.

Більш детальну інформацію про застосунок можна отримати по команді

cf app app_name

де app_name – назва застосунку Node-RED.
Зробіть копію екрану для звіту.

2.6. Перегляд значення змінних середовища

Змінні середовища (Environment Variables) - це засіб, за допомогою якого середовище виконання CF взаємодіє з розгорнутим застосунком. Змінні середовища можуть використовуватися для задавання параметрів роботи застосунків. Значення змінних середовища можна подивитися через IBM Cloud консоль.

Увійдіть у консоль вашого аккаунту IBM Cloud https://cloud.ibm.com/login. Перейдіть до налаштування вашого застосунку Node-RED. Перейдіть на вкладку «Runtime» -> «Environment variables» (рис.4)

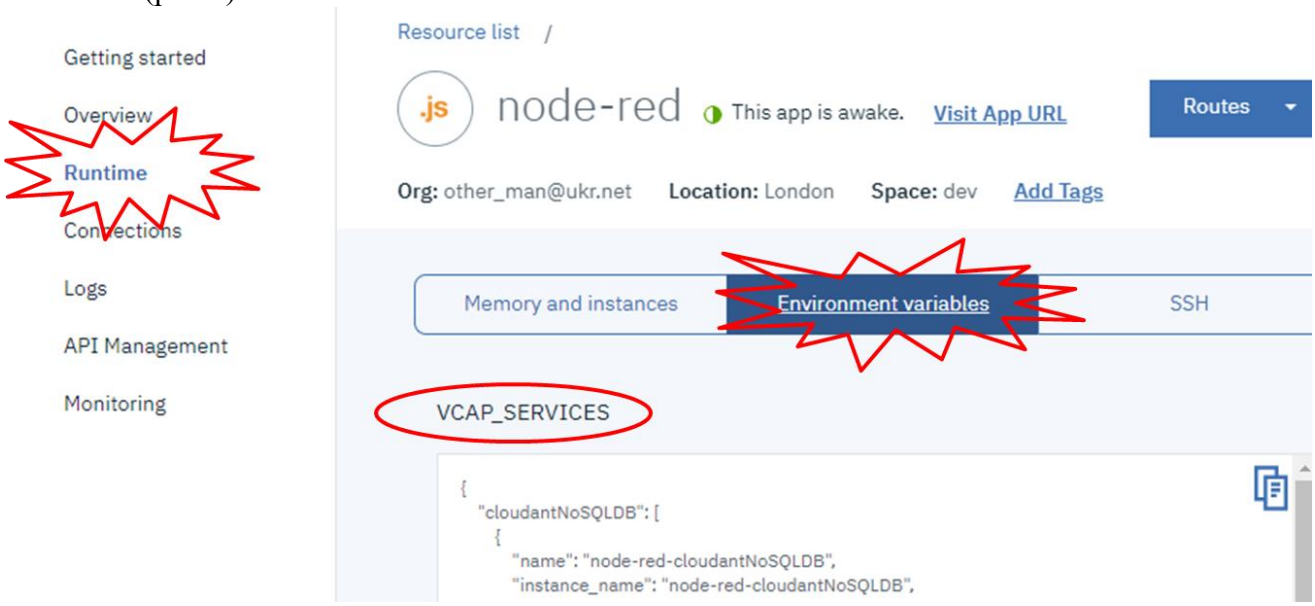

рис.4

Для перегляду змінних середовища можна скористатися командою CLI. Наберіть команду:

cf env ap_name

де ap_name – назва застосунку Node-RED.

Перелік усіх змінних середовища CF доступний за цим посиланням. Змінна VCAP_SERVICES вміщує дані про ті сервіси, до яких відбувається під'єднання, і вона може бути використана на всіх стадіях розгортування та роботи застосунку. Згідно опису, ця змінна має JSON формат, який включає в себе об'єкти, кожен з яких відповідає за сервіс, який підключений до застосунку.

2.7. Перегляд списку підключених сервісів

Подивіться на змінну VCAP_SERVICES, знайдіть, які сервіси підключені. Перелік цих сервісів також доступний у консолі на вкладці Connections (рис.5)

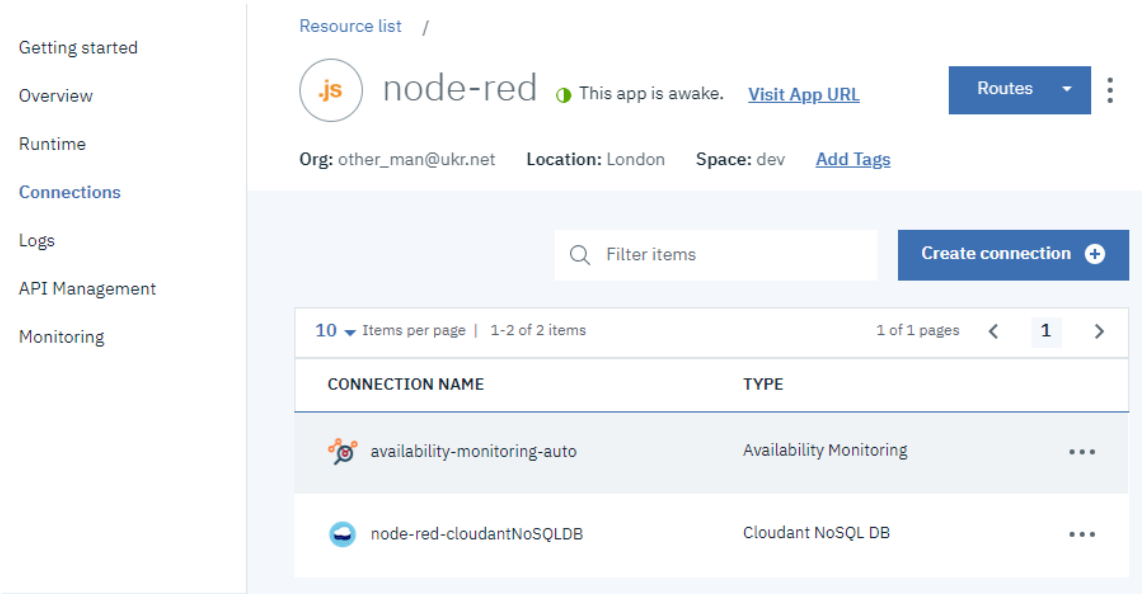

рис.5

Також вони будуть доступними в переліку сервісів Cloud Foundry (Resource list) (рис.6).

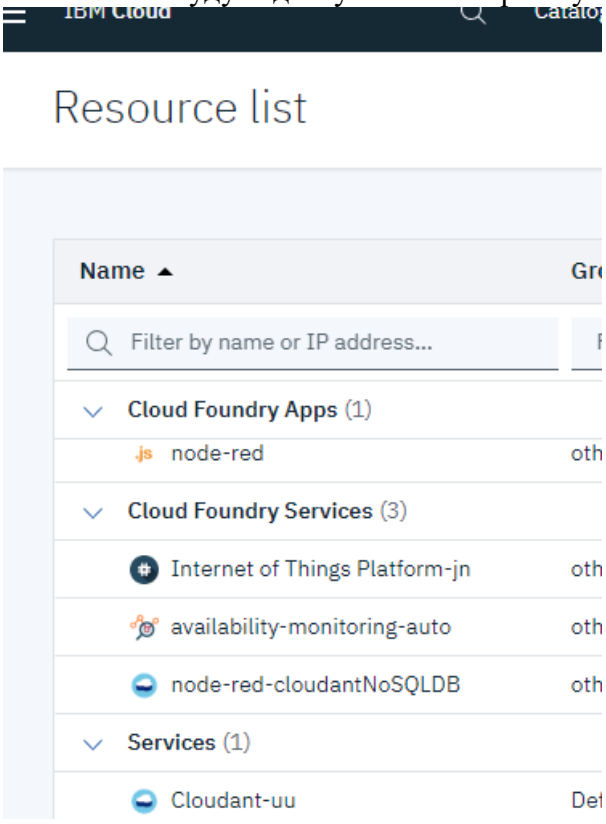

рис.6

Також їх можна подивитися через CLI, у командному рядку введіть

cf services

Зробіть копію екрану.

У таблиці 1 наведені поля сервісів. Проаналізуйте значення цих полів та знайдіть їх в налаштуваннях сервісу Cloudant з Cloud Foundry Services консолі.

Таб.1 Призначення полів VCAP_SERVICES

Attribute	Description
binding_name	Ім'я, присвоєне сервісу, яке прив'язується користувачем
instance_name	Ім'я, присвоєне користувачем
name	Ім'я binding_name, якщо воно існує; інакше ім'я екземпляру
label	Назва мітки сервісу
tags	Масив рядків, які застосунок може використовувати для ідентифікації екземпляра сервісу
plan	План обслуговування, вибраний під час створення екземпляра сервісу
credentials	Об'єкт JSON, що містить облікові дані, необхідні для доступу до екземпляра сервісу

3. Зв'язок застосунку Node-RED з сервісом Cloudant

3.1. Робота застосунку Node-RED з сервісом Cloudant

Хмарний застосунок Node-RED зі стартового набору Node-RED Starter побудований таким чином, що зберігає усі необхідні налаштування в Cloudant. Зокрема, це перелік встановлених модулів та потоки (програма користувача). Необхідність такого збереження пов’язана з тим, що при кожному розгортанні застосунку в droplet, весь його зміст замінюється, тому увесь довготривалий контент необхідно зберігати в зовнішньому сховищі. Cloudant для цього добре підходить, так як Node-RED увесь зміст зберігає в форматі JSON. Щоб переконатися в цьому, достатньо відкрити базу даних Cloudant, що відповідає за Node-RED.

Використовуючи список ресурсів перейдіть на вікно керування сервісом Cloudant (Cloudant Dashboard). У вікні Manage, натисніть “Launch Cloudant Dashboard” для запуску сторінки адміністрування сервісу. У переліку баз даних знайдіть ту, що відповідає за Node-RED, як правило її назва відповідає назві застосунку Node-RED, або має назву «nodered». Структура БД повинна мати три документи (див.рис.7).

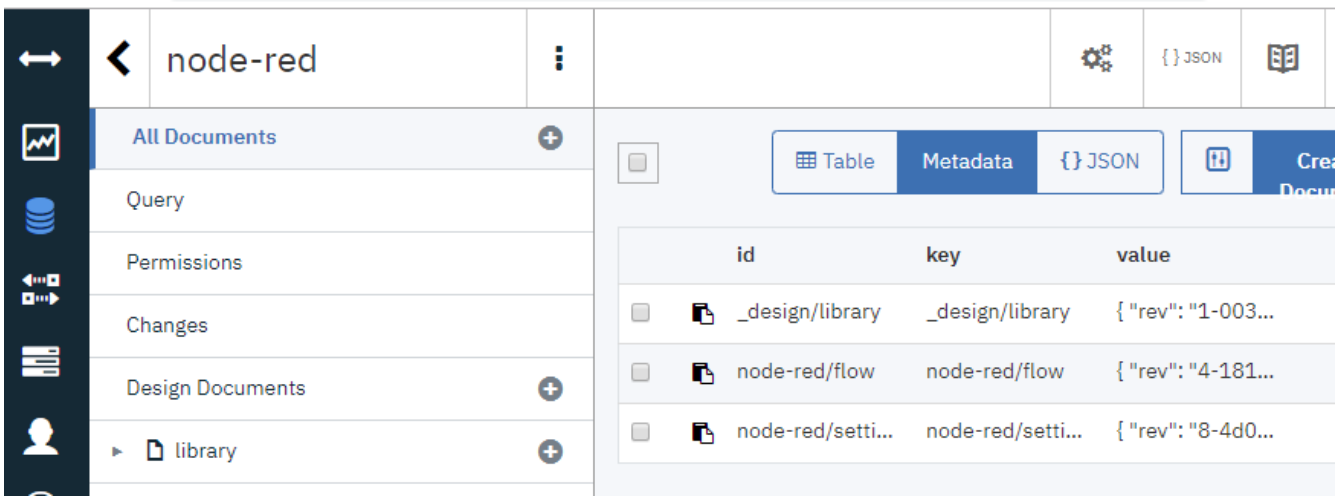

рис.7

Відкрийте документ з назвою, що закнічується на «/settings». У ньому вміщуються налаштування модулів. Там також будуть модулі, які Ви встановлювали в минулій частині лабораторної роботи: "node-red-dashboard" та "node-red-contrib-cos".

Відкрийте документ з назвою, що закнічується на «/flow». У ньому вміщується зміст потоків, які Ви розробляли на минулій частині лабораторної роботи.

Нижче розглянемо як відбувається зв'язок застосунку Node-RED із сервісом Cloudant.

3.2. Доступ до сервісів Cloud Foundry

У частині 3.2 лабораторної роботи було показано, що доступ до сервісів IBM відбувається за схемою IAM (IBM Cloud Identity and Access Management). Однак з точки зору Cloud Foundry доступ до сервісів відбувається за іншою схемою, яка показана на рис.8.

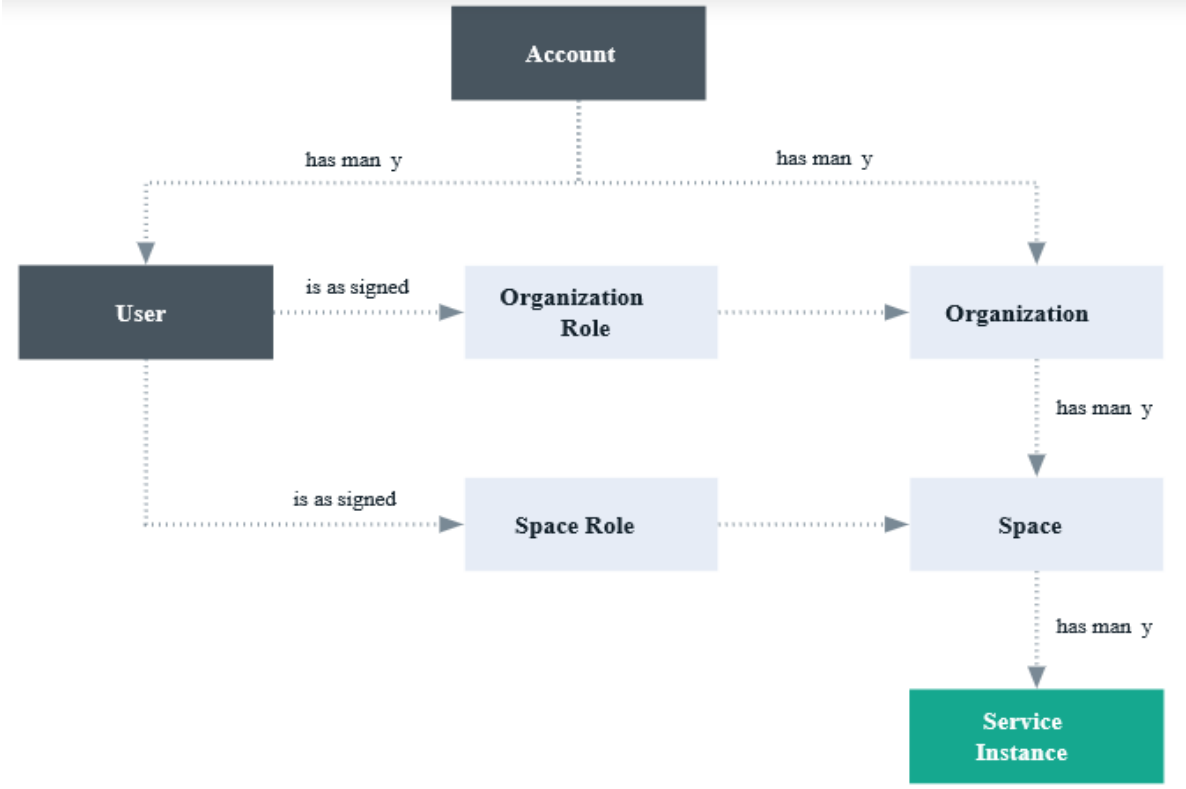

рис.8

Ролі Cloud Foundry надають доступ до організацій (organizations) і просторів (spaces) у межах облікового запису. Ролі Cloud Foundry не дозволяють користувачам виконувати дії в контексті сервісу в обліковому записі, як це робиться в IAM. На рівні організації можна призначити такі ролі:

Таблиця 2. Ролі на рівні org.

Organization role	Permissions (Дозволи)
Manager	Керівники організацій можуть створювати, переглядати, редагувати або видаляти spaces в організації, переглядати використання організації та квоти, запрошувати користувачів до організації, керувати тим, хто має доступ до організації та їхні ролі в організації, а також керувати власними доменами для організації .
Billing manager	Billing managers можуть переглядати інформацію про час виконання та використання служби для організації на сторінці "Інформаційна панель використання".
Auditor	Аудитори організації можуть переглядати вміст програми та послуги в організації. Аудитори можуть також переглядати користувачів в організації та їх призначені ролі, а також квоти для організації.

На рівні space можна призначити такі ролі:

Таблиця 3. Ролі на рівні space.

Space role	Permissions (Дозволи)
Manager	Space managers можуть додавати існуючих користувачів і керувати ролями в

Space role	Permissions (Дозволи)
	просторі. Менеджер простору може також переглядати кількість примірників, прив'язки служб і використання ресурсів для кожного додатка в просторі.
Developer	Розробники простору можуть створювати, видаляти та керувати програмами та послугами в межах простору. Деякі з керуючих завдань включають розгортання додатків, запуск або припинення додатків, перейменування програми, видалення програми, перейменування простору, прив'язки або роз'єднання служби до програми, перегляд кількості чи екземплярів, прив'язки служб і використання ресурсів для кожного додатка в просторі. Крім того, розробник простору може асоціювати внутрішню або зовнішню URL-адресу з додатком у просторі.
Auditor	Space аудитори мають доступ тільки для читання до всієї інформації про простір, наприклад, інформацію про кількість примірників, прив'язки служб і використання ресурсів для кожного додатка в просторі.

Користувачам, яким призначено роль менеджера або developer space, можна отримати доступ до змінної середовища VCAP_SERVICES. Однак користувачеві, якому призначено роль аудитора, не вдасться отримати доступ до VCAP_SERVICES.

Перейдіть на налаштування аккаунту https://cloud.ibm.com/account. У переліку CF orgs буде один запис, який відповідає за ID вашої організації (по суті ваш поштовий ящик). Подивіться яка роль надана вашому обліковому запису і за таблицею 2 подивіться які в неї дозволи. Зробіть клік по імені і зайдіть в налаштування org. Зайдіть на вкладку spaces і за таблицею 3 визначте ваші права на рівні space. Перегляньте зміст вкладок Users, Domains та Quotas.

3.3. Перегляд доступів до сервісів IAM із застосунків Cloud Foundry

Таким чином доступ застосунків CF до сервісів CF, проводиться за ідентифікацією користувача. Існуючий сервіс підключається до існуючого або нового застосунку IBM® Cloud на вкладці Connections інформаційної панелі сервісу. Підключення сервісу CF до застосунку Cloud Foundry створює прив'язку (**bind**) між ними. Однак при підключенні екземпляра сервісу, керованого IBM IAM до застосунку CF, у відповідному просторі Cloud Foundry автоматично створюється псевдонім (alias) сервісу, керованого IAM, з обов'язковою інформацією про доступ. Цей псевдонім представлений у вигляді екземпляра сервісу Cloud Foundry для відповідного сервісу, керованого IAM. Псевдонім (**alias**) - це з'єднання між сервісом, керованим IAM, в межах групи ресурсів і застосунком Cloud Foundry в межах org або простору(space). У консолі IBM Cloud з'єднання (псевдонім) представлено у вигляді екземпляра сервісу Cloud Foundry. Можна керувати своїм псевдонімом, змінюючи екземпляр сервісу, що представляє з'єднання.

У консолі IBM Cloud перейдіть в список ресурсів Cloud Foundry Services знайдіть псевдонім ресурсу, що відповідає за з’єднання вашого CF застосунку Node-RED з сервісом Cloudant. Подивіться на значення “API key Name”, воно повинно співпадати з тим, що означено в змінній середовища VCAP_SERVICES.

Перейдіть в налаштування доступу IAM: Manage->Access (IAM) https://cloud.ibm.com/iam#/overview. Перейдіть до вкладки Service ID, потім до пункту контекстного меню Manage Service ID (рис.9).

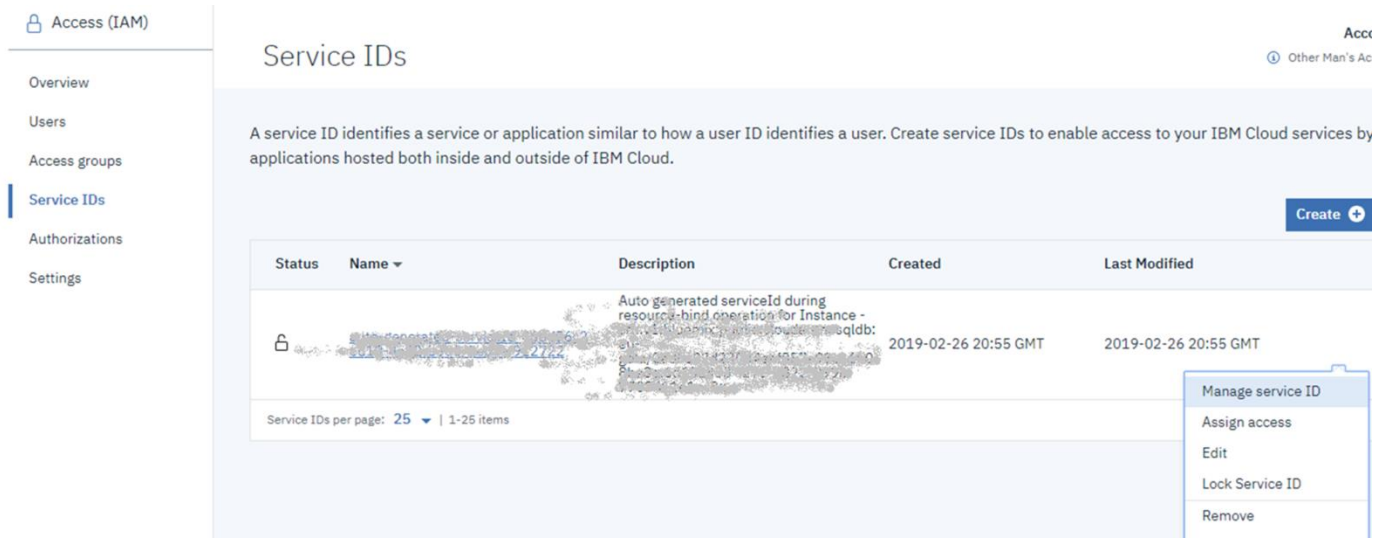

рис.9

На вкладці Access policies знайдіть роль Manager, що базується на типі Resource (рис.10). Подвійним кліком зайдіть в редагування ролі. У даній ролі повинно бути налаштування доступу до сервісу Cloudant з правами Manager.

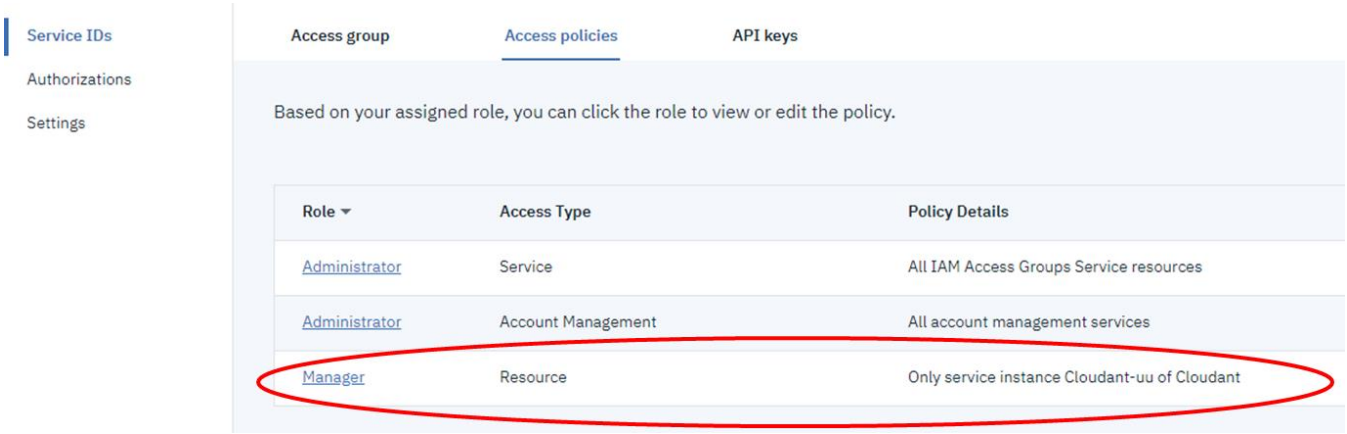

рис.10

Таким чином, застосунок Node-RED, підключаючись до псевдоніму сервісу через означений Service ID, що записаний в змінній VCAP_SERVICES зможе підключитися до його екземпляру через IAM доступ.

4. Керування розгортанням проекту

4.1. Перевірка доступних збірок.

Вище було описано, що застосунки CF розгортаються з сирцевого коду з використанням готових збірок (Buildpack). На основі цих збірок та коду формується droplet, який запускається в контейнерах Garden віртуальних машин Diego Cell. Доступні збірки можна перевірити з CLI.

У командному рядку введіть (рис.11):

cf buildpacks

```
C:\Users\2017>cf buildpacks
Getting buildpacks...

buildpack                                position   enabled   locked   filename
liberty-for-java                         1          true      false    buildpack_liberty-f
sdk-for-nodejs                           2          true      false    buildpack_sdk-for-n
dotnet-core                              3          true      false    buildpack_dotnet-co
swift_buildpack                          4          true      false    buildpack_swift_v2.
java_buildpack                           5          true      false    java-buildpack-v4.9
ruby_buildpack                           6          true      false    ruby-buildpack-v1.7
nodejs_buildpack                         7          true      false    nodejs-buildpack-v1
go_buildpack                             8          true      false    go-buildpack-v1.8.2
python_buildpack                         9          true      false    python-buildpack-v1
php_buildpack                            10         true      false    php-buildpack-v4.3.
xpages_buildpack                         11         true      false    xpages_buildpack_v1
staticfile_buildpack                     12         true      false    staticfile-buildpac
binary_buildpack                         13         true      false    binary-buildpack-v1
xpages_buildpack_v1_2_1-20160913-103     14         true      false    xpages_buildpack_v1
dotnet-core_v2_0-20180918-1356           15         true      false    buildpack_dotnet-co
dotnet-core_v2_1-20181205-1536           16         true      false    buildpack_dotnet-co
sdk-for-nodejs_v3_25-20181219-1036       17         true      false    buildpack_sdk-for-n
sdk-for-nodejs_v3_25_1-20190115-1637     18         true      false    buildpack_sdk-for-n
swift_buildpack_v2_0_16-20181214-0434    19         true      false    buildpack_swift_v2.
swift_buildpack_v2_0_17-20190212-2123    20         true      false    buildpack_swift_v2.
liberty-for-java_v3_28-20190127-1723     21         true      false    buildpack_liberty-f
liberty-for-java_v3_29-20190223-2128     22         true      false    buildpack_liberty-f
```

рис.11

Використовуючи команду cf app, дізнайтеся на основі якої збірки створений ваш застосунок Node-RED.

4.2. Процедура розгортання застосунків CF та перегляд журналів роботи.

Процедура розгортання застосунку складається з наступних етапів (див. рис.12) :

1. У командному рядку розробник входить до каталогу, що містить вихідний код програми, і використовує інтерфейс командного рядка Cloud Foundry для видачі команди **push**.
2. CL CLI повідомляє Cloud Controller, що необхідно створити запис для застосунку.
3. Cloud Controller зберігає метадані застосунку (application metadata). Метадані застосунків можуть включати в себе назву програми, кількість екземплярів, вказаних користувачем, і buildpack та іншу інформацію про застосунок.
4. Перш ніж завантажувати всі вихідні файли програми, cf CLI видає до Cloud Controller запит на відповідність ресурсу, щоб визначити, чи існує який-небудь з файлів застосунку в кеші ресурсів. Якщо файли застосунків завантажуються, cf CLI пропускає файли, які існують в кеші ресурсів, надаючи результат запиту на відповідність ресурсу. Файли завантажених програм об'єднуються з файлами з кешу ресурсів для створення пакета застосунку(application package).
5. Cloud Controller зберігає application package в blobstore.
6. cf CLI видає команду запуску програми.
7. Cloud Controller видає запит на постановку (staging) в Diego, який потім планує (шедулить) Diego cell ("Cell"), щоб виконати здачу постановки ("Task"). Задача завантажує збірки (buildpacks) та кеш-код buildpack застосунку, якщо вони є. Потім він використовує buildpack, який виявляється автоматично (за структурою сирцевого коду) або задається за допомогою прапора -b, для компіляції та стадії програми.
8. Cell направляє вихідні дані процесу постановки, щоб розробник міг усунути неполадки, що виникають у застосунку.
9. Задача пакує отриманий скомпільований застосунок в архів, який називається " droplet ", а Cell зберігає droplet в blobstore. Задача також завантажує кеш buildpack до blobstore для використання під час наступної постановки (staged) зстосунку.
10. Система Diego Bulletin Board System повідомляє Cloud Controller, що проходження завершено (staging is complete). Постановка (Staging) повинна завершитися протягом 15

хвилин або постановка вважається невдалою. Застосунку надається мінімум 1 Гб пам'яті, навіть якщо потрібна оперативна пам'ять менша.

11. Дієго планує (шедулить) додаток як довготривалий процес (Long Running Process) на одній або декількох Diego cells.
12. Diego cells повідомляють про стан застосунку до Cloud Controller

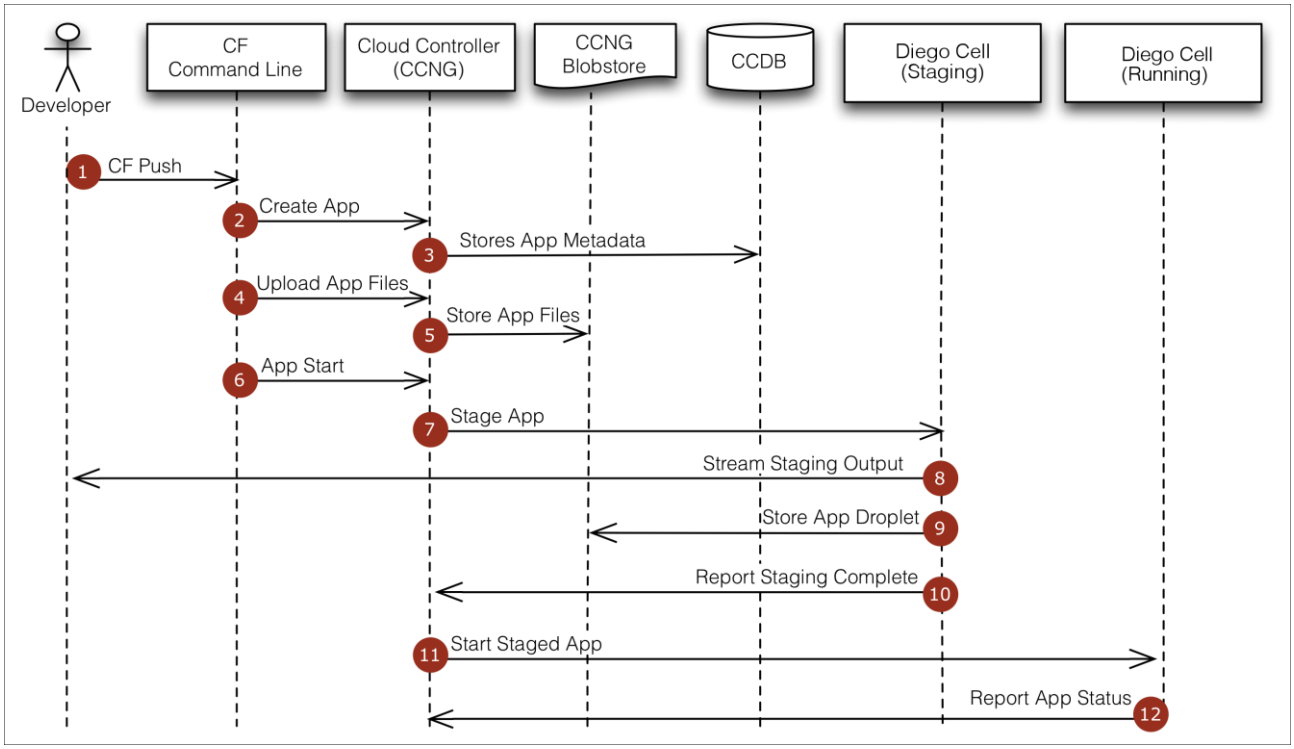

рис.12

Детально процедура описано за цим посиланням.

На кожній стадії застосунок і служби можуть формувати повідомлення в журналі. Для перегляду журналу скористуйтеся командою:

cf logs app_name –recent

де *app_name* – ім'я застосунку Node-RED

Якщо після попереднього використання застосунок переходив у режим sleep (про це мало повідомлятися поштою), і він знаходиться в режимі помилки, в журналі буде висвітлено причину помилки. Наприклад,

OUT Exit status 1 (out of memory)

При правильному повторному розгортанні, ця помилка буде виправлена.

4.3. Завантаження коду Node-RED Starter в локальну папку

Кнопка створення «Node-RED IBM Cloud Starter Application», якою Ви користувалися для створення застосунку в частині 3.1 лабораторної роботи містить таке посилання

https://bluemix.net/deploy?repository=https://github.com/knolleary/node-red-bluemix-starter.git

Це посилання розгортає застосунок з сирцевого коду, що розміщується на github. Коли Ви натискаєте кнопку, то IBM Cloud вибирає код з цього сховища і робить його розгортання. При цьому автоматично створюється екземпляр служби Cloudant, який прив'язується застосунку.

Робоча чернетка. Автор - Олександр Пупена, доц. ІАСУ НУХТ. Поширюється для тестування і зворотного зв'язку з побажаннями і правками. pupena_san@ukr.net

Код застосунку можна завантажити з github і залити з використанням команд CLI.

Завантажте та розархівуйте з github код застосунку **bluemix-starter** https://github.com/knolleary/node-red-bluemix-starter ("clone or download" -> "download zip"). Подивіться на структуру каталогу проекту.

Зокрема в файлі «\.bluemix\pipeline.yml» вказана стадійність розсортування.

Деталі доступні за наступними посиланнями:

- Creating a Deploy to IBM Cloud button
- A Deploy-To-Bluemix enabled instance of Node-RED that can be forked, customised and reused. .

4.4. Робота з файлом маніфесту

Для розгортання застосунків використовується інструмент інтерфейсу командного рядка Cloud Foundry (див. CLI) командою cf push. **cf push** розгортає програми з типовою кількістю екземплярів, обмеженням дискового простору та обмеженням пам'яті. Ви можете перевизначити значення за замовчуванням, викликавши cf push з прапорами та значеннями, або вказавши пари ключ-значення в файлі маніфесту **manifest.yml**. Маніфести забезпечують узгодженість і відтворюваність, а також допомагають автоматизувати розгортання програм. Детальний опис доступний за посиланням.

При першому розгортанні Node-RED Starter, створюється сервіс Cloudant та БД, ім'я якої визначається за назвою застосунку. Це визначено в файлі pipeline.yml, який вказує IBM в якому порядку що створюється. Враховуючи що в назві застосунку можуть бути великі літери, які не дозволені в назві БД, база даних за замовчуванням може отримати назву «nodered». У цьому випадку, при повторному розгортанні застосунок не зможе знайти БД, тому її назву треба прописати в файлі маніфесту через поле NODE_RED_STORAGE_DB_NAME, так як застосунок шукає її саме в цій змінній середовища.

Відкрийте у завантаженій папці проекту файл маніфесту **manifest.yml.** Добавте в файлі маніфесту (**manifest.yml**) запис в розділ **env** , в якому змінній середовища NODE_RED_STORAGE_DB_NAME вказується ім'я бази даних в Cloudant, яка відповідає з Node-RED (наприклад **nodered**), а також три дефіса на початку («---»). Перед «NODE_RED_STORAGE_DB_NAME» треба зробити чотири пробіли. Зміни краще робити за допомогою редактору Notepad++.

```
---
applications:
- memory: 256M
  env:
    OPTIMIZE_MEMORY: true
    NODE_RED_STORAGE_DB_NAME: nodered
  command: node index.js --settings ./bluemix-settings.js –v
```

Після внесення змін необхідно зберегти файл.

4.5. Повторна відправка та перебудова застосунку.

Увага! Даний пункт виконується тільки при невдалому повторному запуску застосунку Node-RED!

Робоча чернетка. Автор - Олександр Пупена, доц. ІАСУ НУХТ. Поширюється для тестування і зворотного зв'язку з побажаннями і правками. pupena_san@ukr.net

1. Запустити командний рядок (**cmd**) в ньому:
    a. перейти на папку з завантаженим проектом, наприклад у Windows

    cd c:/node-red-bluemix-starter

    b. вказати api endpoint, де знаходиться застосунок, якщо цього не зроблено, у нашому випадку:

    cf api https://api.eu-gb.bluemix.net

    c. визвати команду реєстрації в хмарі, якщо цього ще не зроблено

    cf login

    d. вказати пошту та пароль (пароль вводиться без явного відображення символів)
    e. зробити push проекту, вказавши ім'я Вашого екземпляру Node-RED, наприклад ***app_name***

    cf push app_name

2. Якщо в кінці розгортання з'явиться запис «FAILED», це значить що якась процедура зроблена невірно і треба повторити спробу. У випадку вдалого розгортання перевірити роботу Node-RED.

Короткий опис послідовності повторного розгортання доступний за цим посиланням.

04.03.2019