

# IBM CCloud Functions and Cloudant

ПОДИВИТИСЯ ПОТІМ

<https://gist.github.com/ashvayka/4735c78541cebd26f3ed3d340743c697>

## Зв'язування IBM Cloud Functions з хмарними сервісами на прикладі NoSqlDB IBM Cloudant

Створення сервісу бази даних та Credential для інтеграції з іншими сервісами

- Створити сервіс можна, знайшовши в каталозі, або ж по прямому лінку: [Cloudant create service](#)

Екран показаний на pic-1

Select an environment

Multitenant Dedicated

Your instance will be running securely on environments with shared resources

Available regions

Frankfurt

Configure Cloudant instance

Instance name: Cloudant-3e

Resource group: Default

Tags: Examples: env:dev, version-1

Authentication method: IAM

Plan

Lite

Full functionality for development and evaluation with a set capacity. Only one Lite plan instance per account.

Standard

Granular control over provisioned throughput capacity allocated. Billing prorated hourly.

Standard on Transaction Engine

Uses IBM's next generation data layer architecture which provides lower costs.

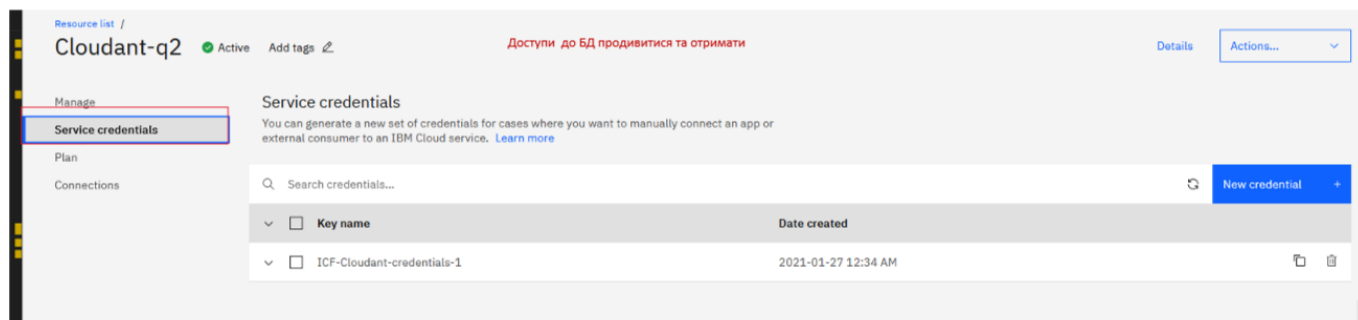
1 Cloudant Lite

20 Reads/sec	Free
10 Writes/sec	Free
5 Global Queries/sec	Free
1 GB Storage	Included

pic-1

Зверніть увагу на обмеження.

- Переглянути credentials, і створити, якщо вони не створилися автоматично. Екран та відповідне меню показано на [pic-2]



pic-2

## Створення прикладного пакету ICF та його зв'язування з сервісом БД

Після створення сервісу потрібно обов'язково вказати ресурсну групу сервісів, за замовчуванням "Default" в CLI. Пересвідчитися, що ви встановили групу ресурсів можна командою:

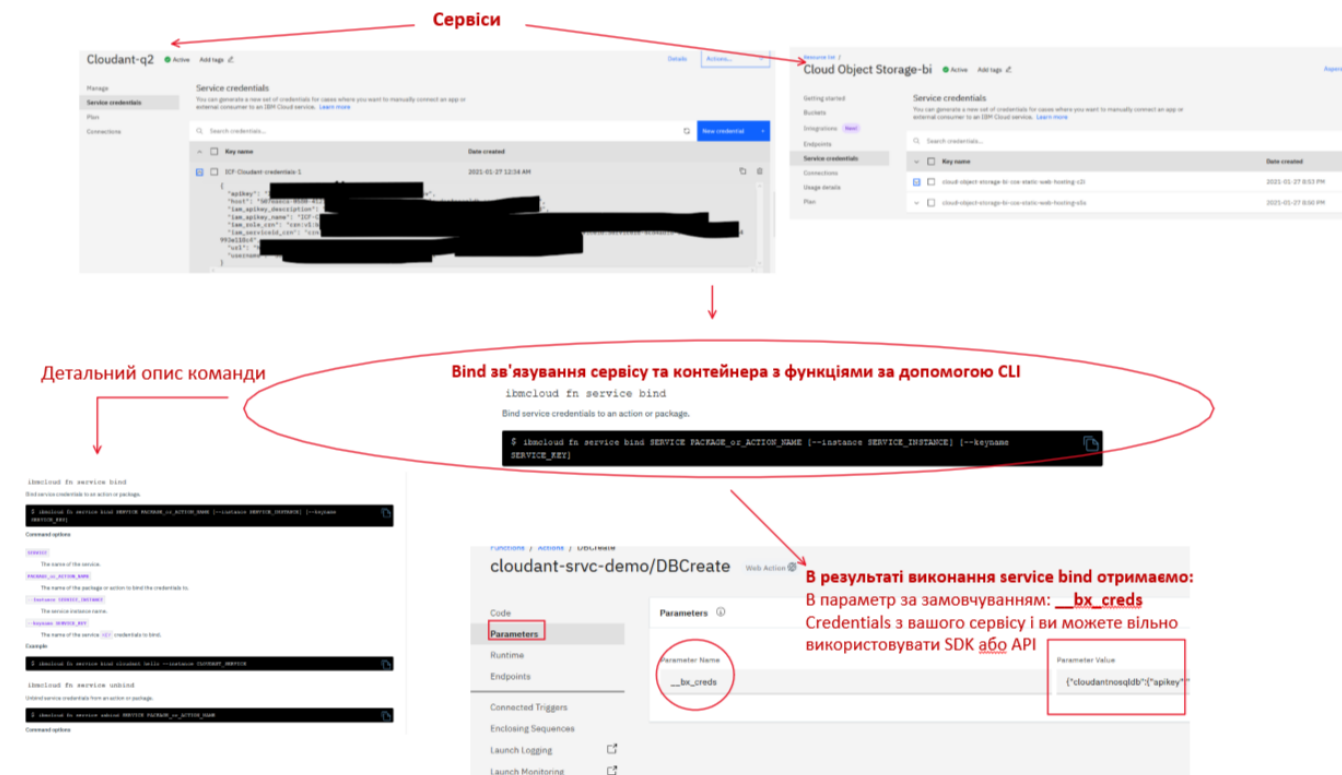
```
ibmcloud target -g Default
```

Отримати опис credentials для вибраного сервісу можна командою:

```
ibmcloud resource service-keys --instance-name Cloudant-q2
```

- Розглянемо, що таке service bind (зв'язування сервісів).

На рис-4 показана діаграма, що можливо спростить розуміння цього процесу.



pic-4

Тепер створимо пакет IVF та виконаємо зв'язування пакета з сервісом БД Cloudant

### Швидкий доступ до переліку команд CLI

- Створимо пакет **cloudant-srvc-demo**

```
ibmcloud fn package create cloudant-srvc-demo
```

Отримаємо результат:

```
create cloudant-srvc-demo
ok: created package cloudant-srvc-demo
```

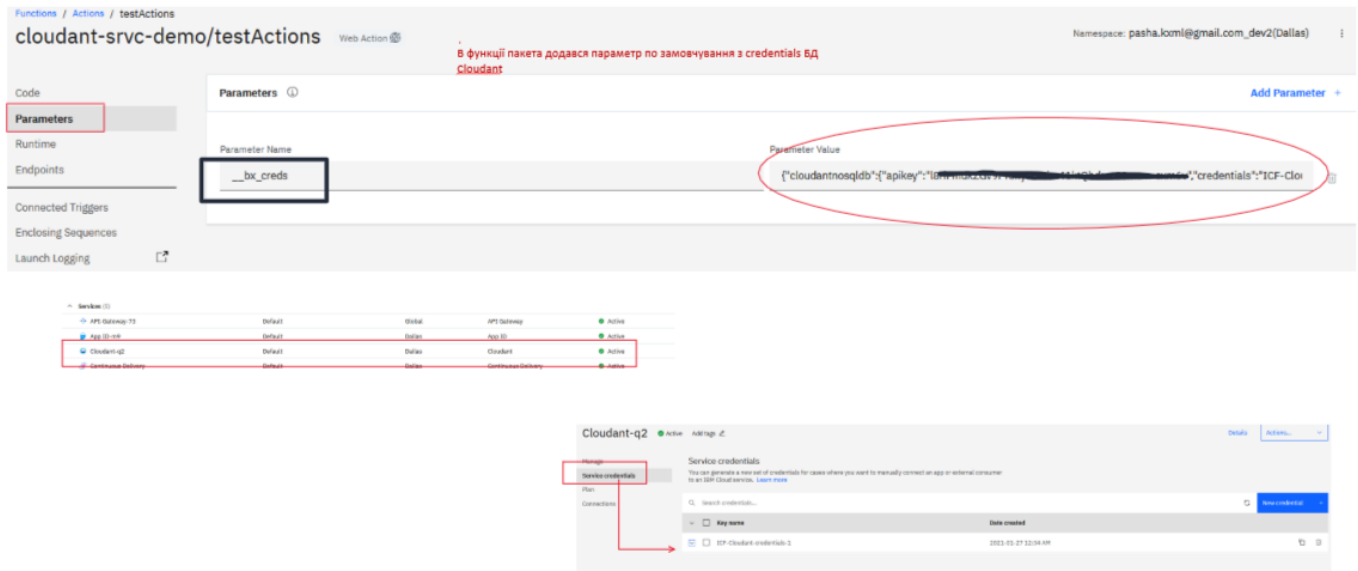
- Зв'яжемо пакет за БД Cloudant

```
ibmcloud fn service bind cloudantnosqlldb cloudant-srvc-demo --instance Cloudant-q2 --keyname ICF-Cloudant-credentials-1
```

результат:

Credentials 'ICF-Cloudant-credentials-1' from 'cloudantnosqldb' service instance 'Cloudant-q2' bound to 'cloudant-srvc-demo'.

На pic-5 показано результат.



pic-5

Виходячи з того, що є CLI через який можна виконати практично всі команди по управлінню ICF, можна побудувати процес колективної розробки з такими ключовими властивостями:

- Розробка та відладка функцій ведеться локально;
- Використання test driven development при розробці функцій
- Програмний код зберігається в Source control;
- Credentials повинні на локальній станції лежати окремо від прикладних параметрів і не попадати в source control;
- Автоматичний deployment по commit або по merge request;
- Після deployment провести невеликий тест функцій, щоб пересвідчитися що вони працюють

## Модель колективної розробки

Призначення каталогів.

- deployment Набір скриптів для deployment з приводу deployment цікаві такі лінки: [Deploying entities with a manifest file](#)

## Cloud Functions CLI reference

- doc Документація
- params

Json-файли з описом параметрів функцій для їх тестування. Використовується як локально, так і в хмарі при тестуванні функцій.

приклад файлу з параметрами для функції DocCreate (./src/DocCreate.js) ./params/DocCreate.json

```
{  
  
  "dbname": "db-demo-02",  
  "doc": { "descr": "doc descr", "doctype": "DOC" }  
  
}
```

- params.localdev

Json-файли з описом параметрів функцій, що включають credentials. В github не передається, знаходиться в .gitignore

Приклад файлу з описом реквізитів credentials для підключення до БД cloudant  
./params.localdev/cloudant.json

```
{"cloudantnosqldb": {  
  "apikey": "",  
  "credentials": "",  
  "host": "",  
  "iam_apikey_description": "",  
  "iam_apikey_name": "",  
  "iam_role_crn": "",  
  "iam_serviceid_crn": "",  
  "instance": "",  
  "url": "",  
  "username": ""  
}
```

- src Програмний код функцій
- test

Програмний код тест-кейсів На прикладі показано як в тестовому кейсі готуються параметри функції. В файлі ./test-DocCreate.js тесткейси для DocCreate (./src/DocCreate.js)

```
it('function DocCreate:' + ' Expect create document', function(done){  
  
  var vfunc = require('../src/DocCreate');  
  var vprm = require('../params/DocCreate.json');  
  var vbxcreds = require('../params.localdev/cloudant.json');  
  vprm.__bx_creds = vbxcreds;
```

```
    vfunc.main( vprm )
    .then (res => {
        res.should.have.property('ok');
        res.ok.should.equal(true);
        res.should.have.property('dbname');
        res.dbname.should.equal( vprm.dbname );
        res.should.have.property('id');
        res.should.have.property('rev');

        if (test_env_lr) {
            console.log( JSON.stringify( res ) );
        }
        done();
    })
    .catch ( err => {
        console.log(err.message);
        done(err);
    });
}); //it
```

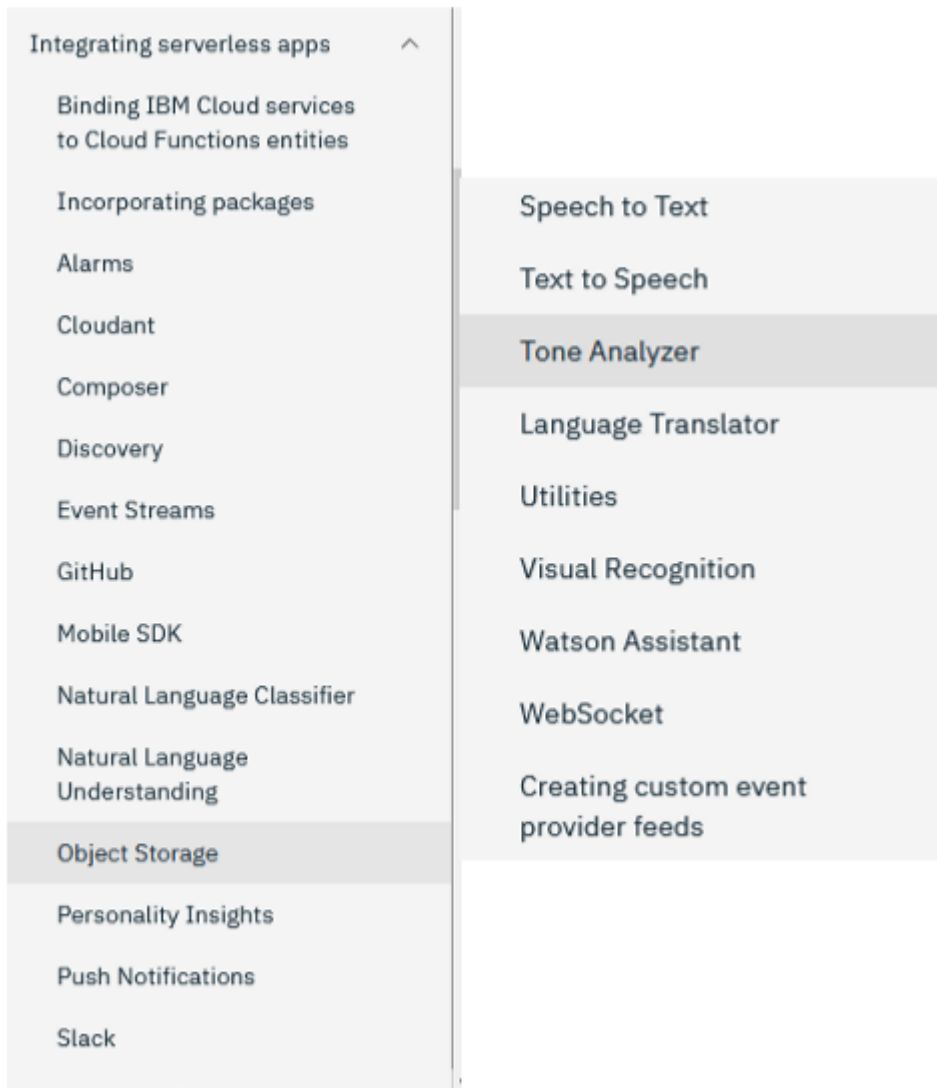
В кореневому каталозі package.json з залежностями та manifest.yml з описом deployment пакету.

Таким чином, модель локальної розробки співпадає з моделлю deployment по credential і взагалі вони (Credentials) можуть бути не залежними.

## Викоритсання SDK та API для сервісів

За ICF можна безшовно поєднати [такі сервіси в хмарі IBM](#)

На pic-6 показано перелік.



pic-6

Більш детально, можете почитати за лінком [Binding IBM Cloud services to Cloud Functions entities](#)

Базшовна інтеграція - зачить в контейнері де запускається ваша функція уже встановлено відповідне SDK. Перелік пакетів, які надаються за замовчуванням можна отримати за лінком: [Node.js packages](#)

Про всі runtimes можна почитати тут: [Runtimes](#)

Для роботи з NoSQL DB Cloudant (Couch DB) використовується Node.js SDK, яке доступне за лінком

Для Node.js створення документу в БД показано доить детально [pic-7]:

**Request**

**Custom Headers**

**Content-Type** string  
Allowable values: [application/json, multipart/mixed, multipart/related]

**Path Parameters**

**db** \* string  
Path parameter to specify the database name.  
Constraints: length ≤ 238. Value must match regular expression ^\_dbs\$|^\_global\_changes\$|^\_metadata\$|^\_nodes\$|^\_replicator\$|^\_users\$|^([a-z][a-z0-9\_\$()]+)\$

**Query Parameters**

**batch** string  
Query parameter to specify whether to store in batch mode. The server will respond with a HTTP 202 Accepted response code immediately.  
Allowable values: [ok]

**Request Body** \* Document  
HTTP request body for Document operations.  
Example: [View](#)

**\_attachments**  
Schema for a map of attachment name to attachment metadata.

Did you find this useful?  
☐ Yes ☐ No

**Node**

```
import { CloudantV1 } from '@ibm-cloud/cloudant';

const service = CloudantV1.newInstance({});

interface ProductsDocument extends CloudantV1.Document {
  _id: string;
  type: string;
  productid: string;
  brand: string;
  name: string;
  description: string;
  price: number;
  image: string;
};

const productsDoc: ProductsDocument = {
  _id: 'small-appliances:1000042',
  type: 'product',
  productid: '1000042',
  brand: 'Salter',
  name: 'Digital Kitchen Scales',
  description: 'Slim Colourful Design Electronic Cooking Appliance for Home / Kitchen, Weigh up to 5kg + Aquatronic for Liquids ml + fl. oz. 15Yr Guarantee - Green',
  price: 14.99,
  image: 'assets/img/0gmsnghew.jpg'
};

service.postDocument({
  db: 'products',
  document: productsDoc
}).then(response => {
  console.log(response.result);
});
```

pic-7

Відладку функції виконуємо через запуск тестових кейсів.

## Deployment з GitHub - подібного репозиторію

Для Deployment використовується класична toolchain. Її складові показані на [pic-8]

**Developer tools (3)**

Toolchain Name	Default	Location	Type
NodejsExpressAppVAOHO2021-01-22	Default	Dallas	Toolchain
icf-cos-toolchain	Default	Dallas	Toolchain
icf-toolchain	Default	Dallas	Toolchain

**icf-cos-toolchain**

Overview

Think: ☒ Configured

Code: ☒ Configured

Deliver: ☒ Success

**cos-pipeline | Delivery Pipeline**

cosDeployment

cosTest

pic-8



Процес deployment складається з двох кроків:

- deployment
- test

Для встановлення toolchain потрібно виконати такі кроки

Підключити сервіс [Continuous Delivery](#)

The screenshot shows the IBM Cloud console interface for the Continuous Delivery service. The top section is titled 'Continuous Delivery' with a 'Create' button. Below it, there's a 'Select a region' dropdown menu set to 'Dallas'. The 'Select a pricing plan' section shows a table with one plan, 'Lite', which is free. The bottom section, 'Services (6)', lists installed services, with 'Continuous Delivery' highlighted.

Plan	Features	Pricing
Lite	Continuous Delivery for organizations (orgs) or resource groups of up to 5 users 5 users per organization or resource group 500 Delivery Pipeline jobs run per organization per month or per resource group per month 500 MB of private Git Repos storage per organization or resource group  The Lite plan offers the full capabilities of Continuous Delivery, with usage limits, to small teams at no cost. Lite plan services are deleted after 30 days of inactivity.	Free

Service	Default	Region	Service Name	Status
API-Gateway-73	Default	Global	API Gateway	Active
App ID-m9	Default	Dallas	App ID	Active
Cloudant-q2	Default	Dallas	Cloudant	Active
Continuous Delivery	Default	Dallas	Continuous Delivery	Active
Continuous Delivery	Default	Dallas	IBM Log Analysis with LogDNA	Active
Tone Analyzer-um	Default	Dallas	Tone Analyzer	Active

pic-9

Створити [Toolchain](#)

З списку потрібно вибрати: [Build your own toolchain](#)

## Other Templates



### Build your own toolchain

IBM

For advanced users, create your toolchain from scratch.

pic-10

[Toolchains](#) / [Create a toolchain](#) /

## Build your own toolchain

Create

About

Toolchain Name:

empty-toolchain-20210129115728189

Select Region:

Frankfurt

Select a resource group:

Default

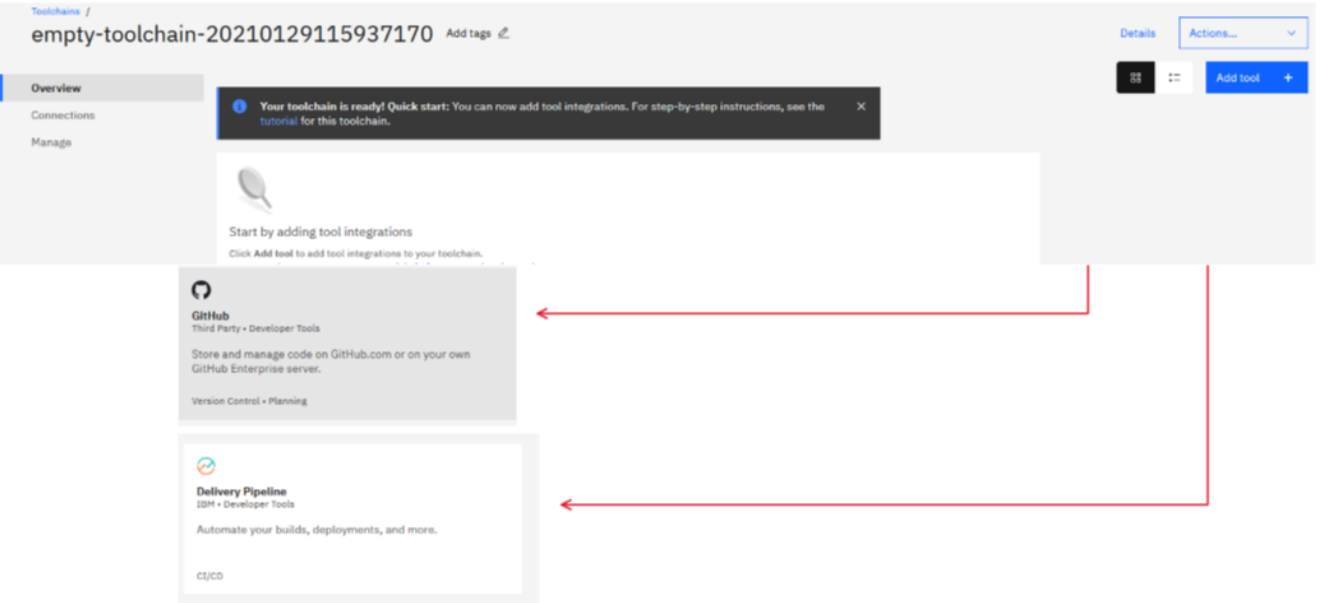
[Select a CF Organization \(deprecated\)](#)

Cancel

Create

pic-11

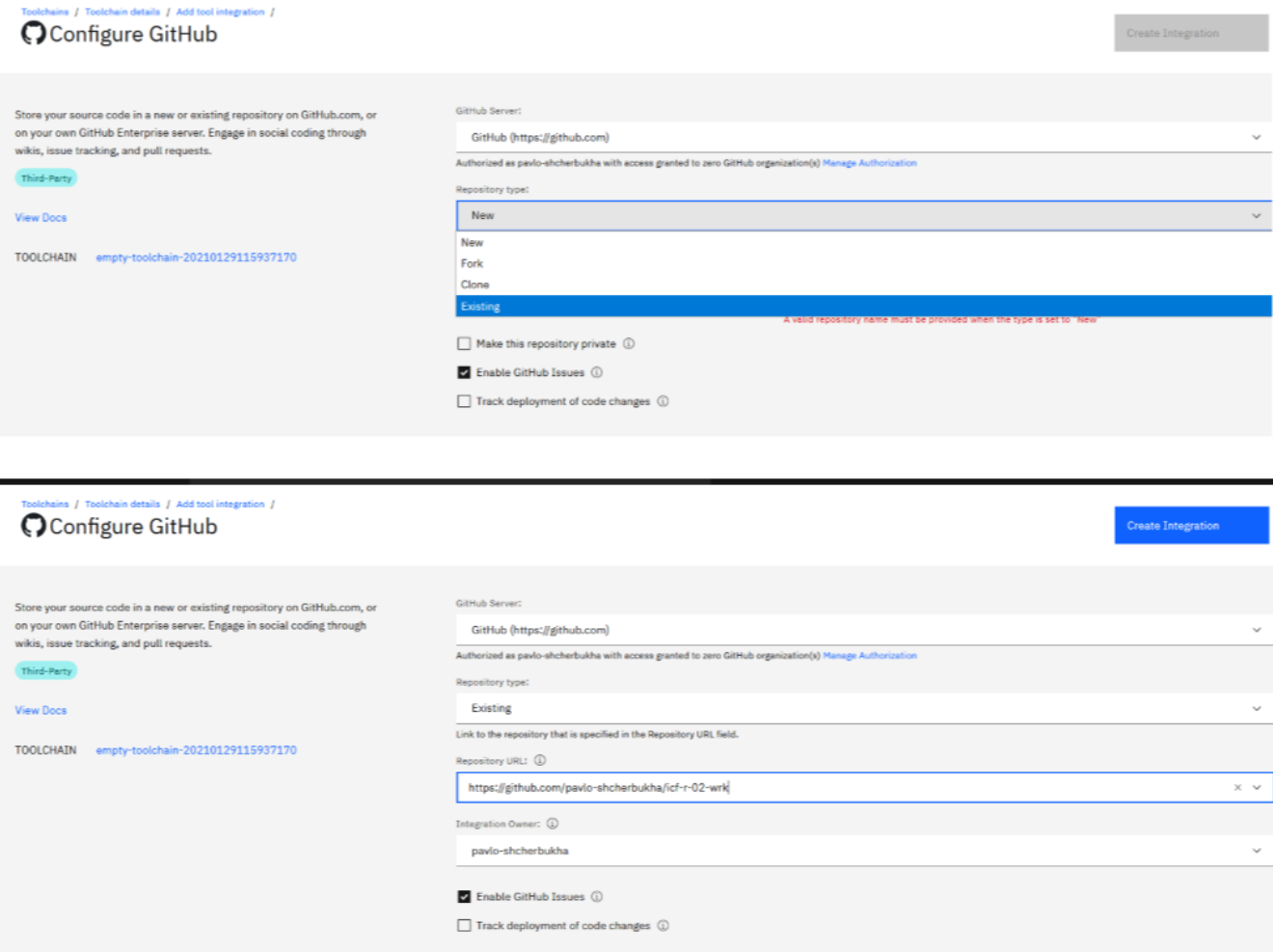
Потім по черзі додаємо Github Integration та delivery pipeline [pic-12]



pic-12

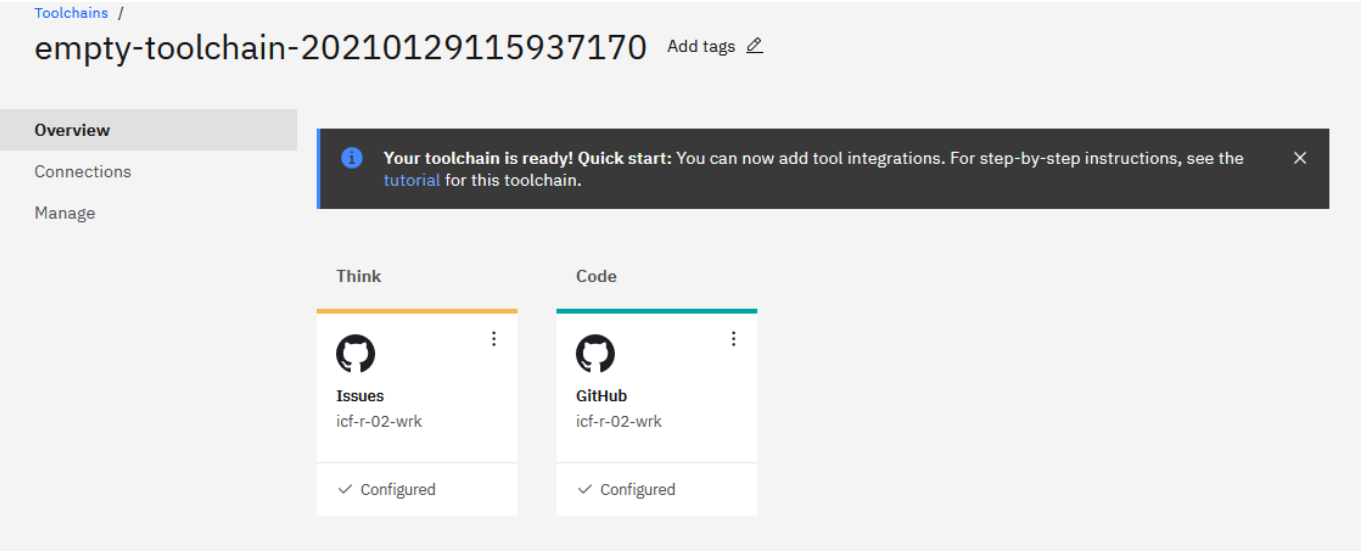
Настройка інтеграції з github

На малюнку [pic-13] показано як настроїти github.

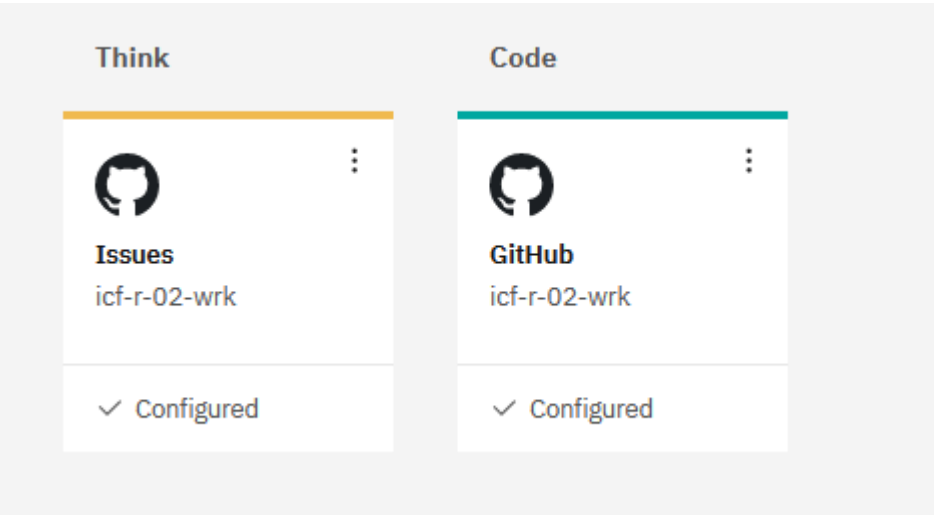


pic-13

та що отримуємо в результаті



pic-14

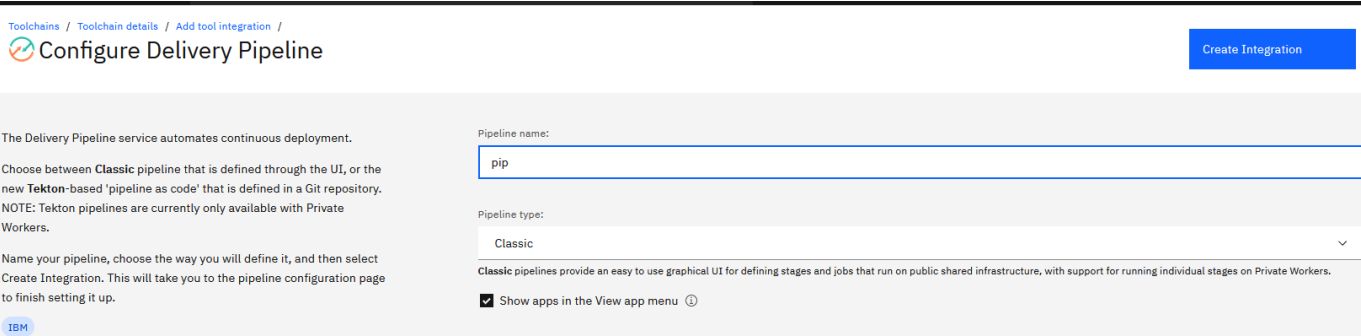


pic-15

Настройка Delivery Pipeline

Додати Delivery Pipeline та вказати найменування

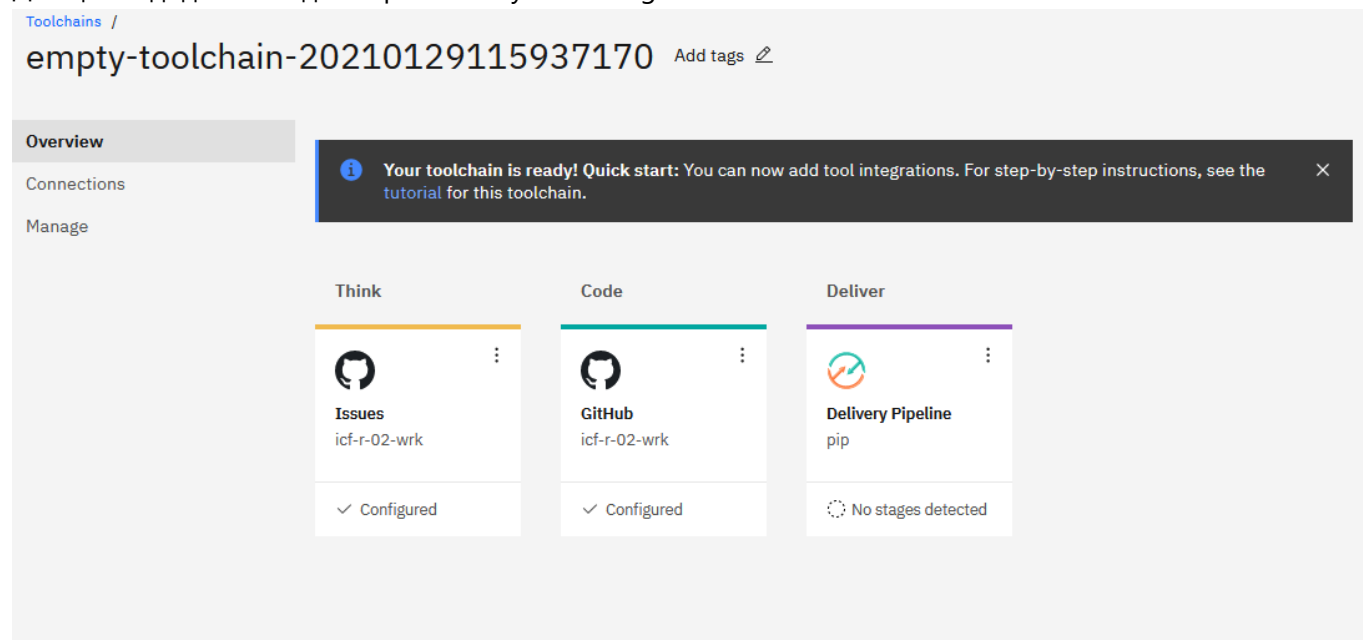
Додати Delivery Pipeline та вказати найменування



pic-16

Створити стейджі Delivery Pipeline

Для цього додати стейджі через кнопку "Add Stage"



pic-17



pic-17

## Створення Deployment Stage

- Спершу потрібно настроїти вкладку Input для Deployment Stage, що забирає програмний код з git

Toolchains / empty-toolchain-20210129115937170 / pip /  
pip | Stage Configuration

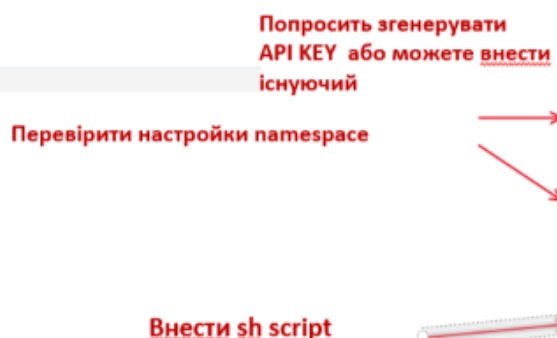
pic-19

Та вибати опції, як показано.

- Перейти у вкладку environment properties та внести туди ваш API KEY з яким система буде логінитися в IBM-Cloud

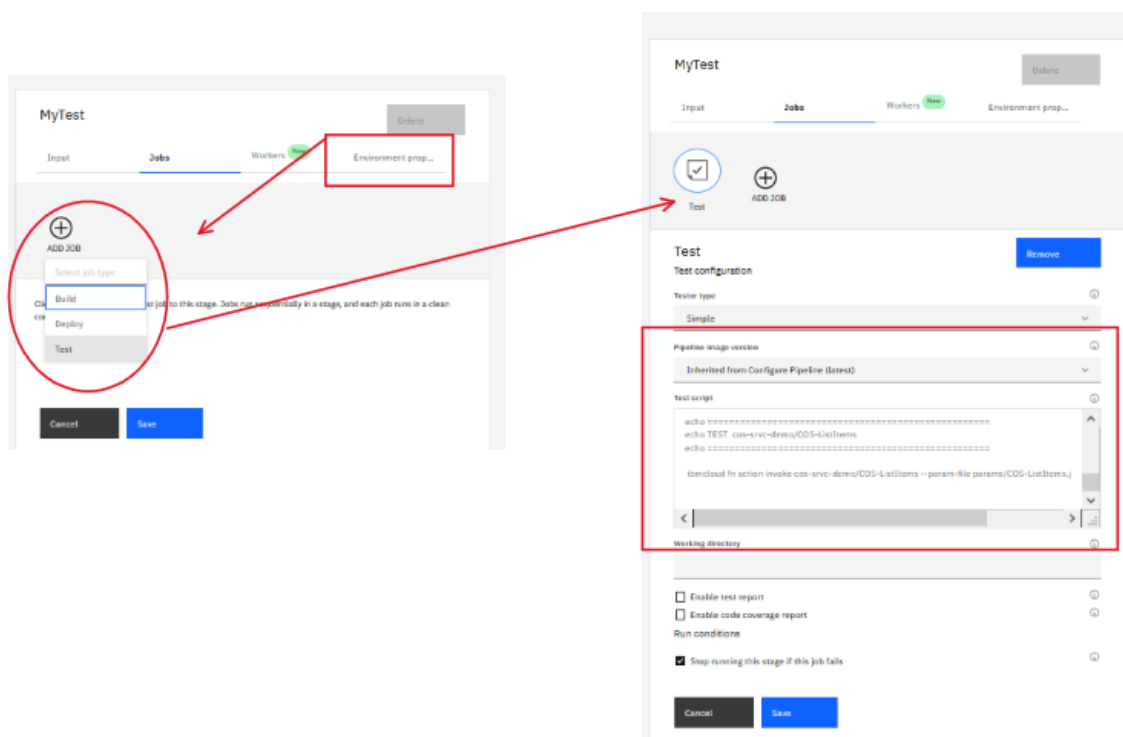
pic-20

- Зайти у вкладку JOBS, створити deployment job та вставити в нього .sh скрипт з папки ./deployemnt/deploy-job.sh



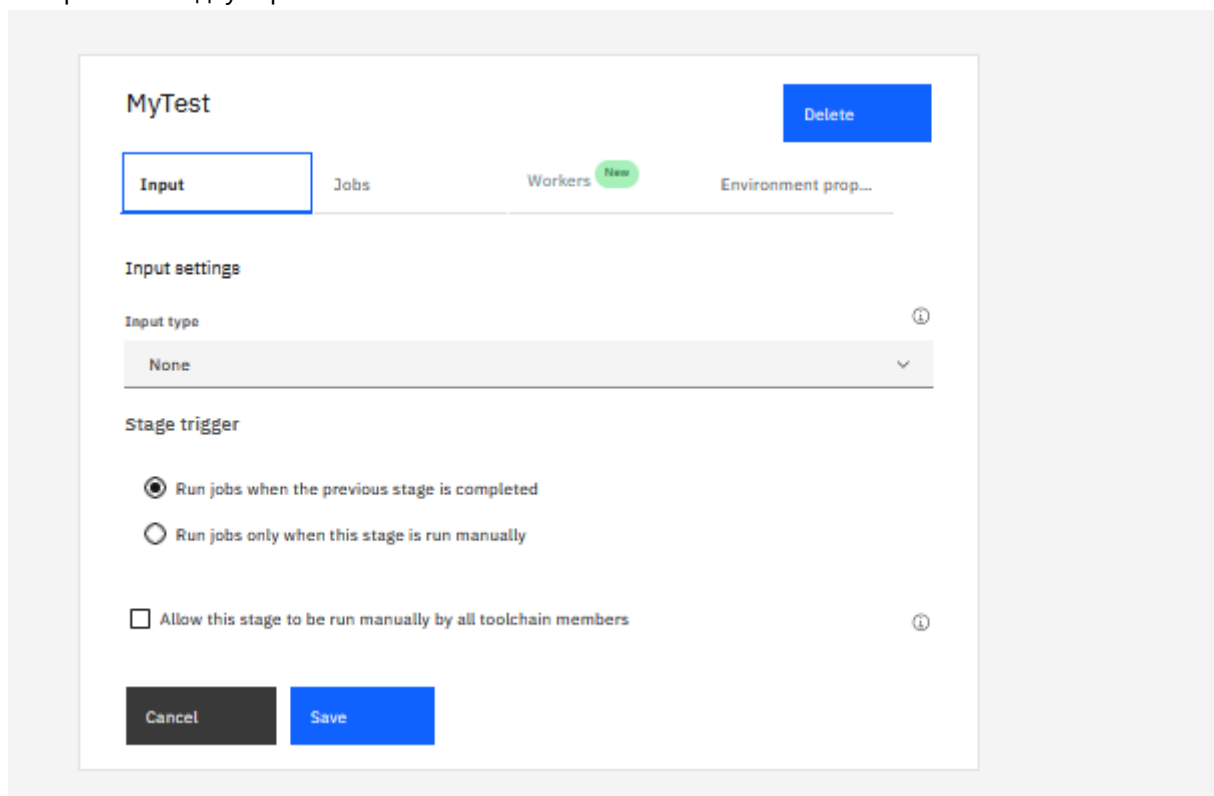
## Створення Test Stage

- 15 / 17



pic-22

- настроїти вкладку Input

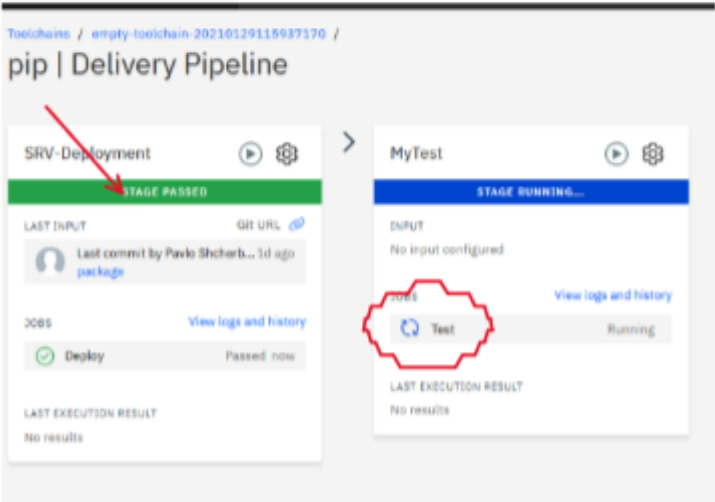
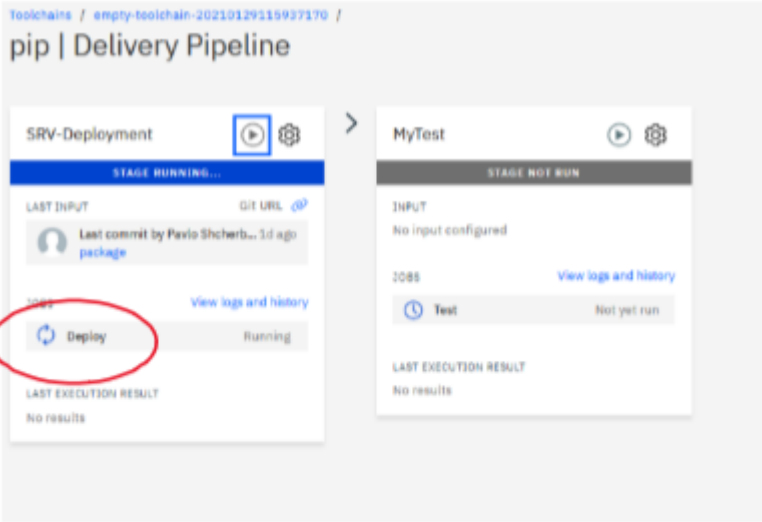


pic-23

## Запустити Stages вручну для перевірки

Результати показані на pic-24





pic-24