IBM CLoud Functions and Cloudant

подивитися потім

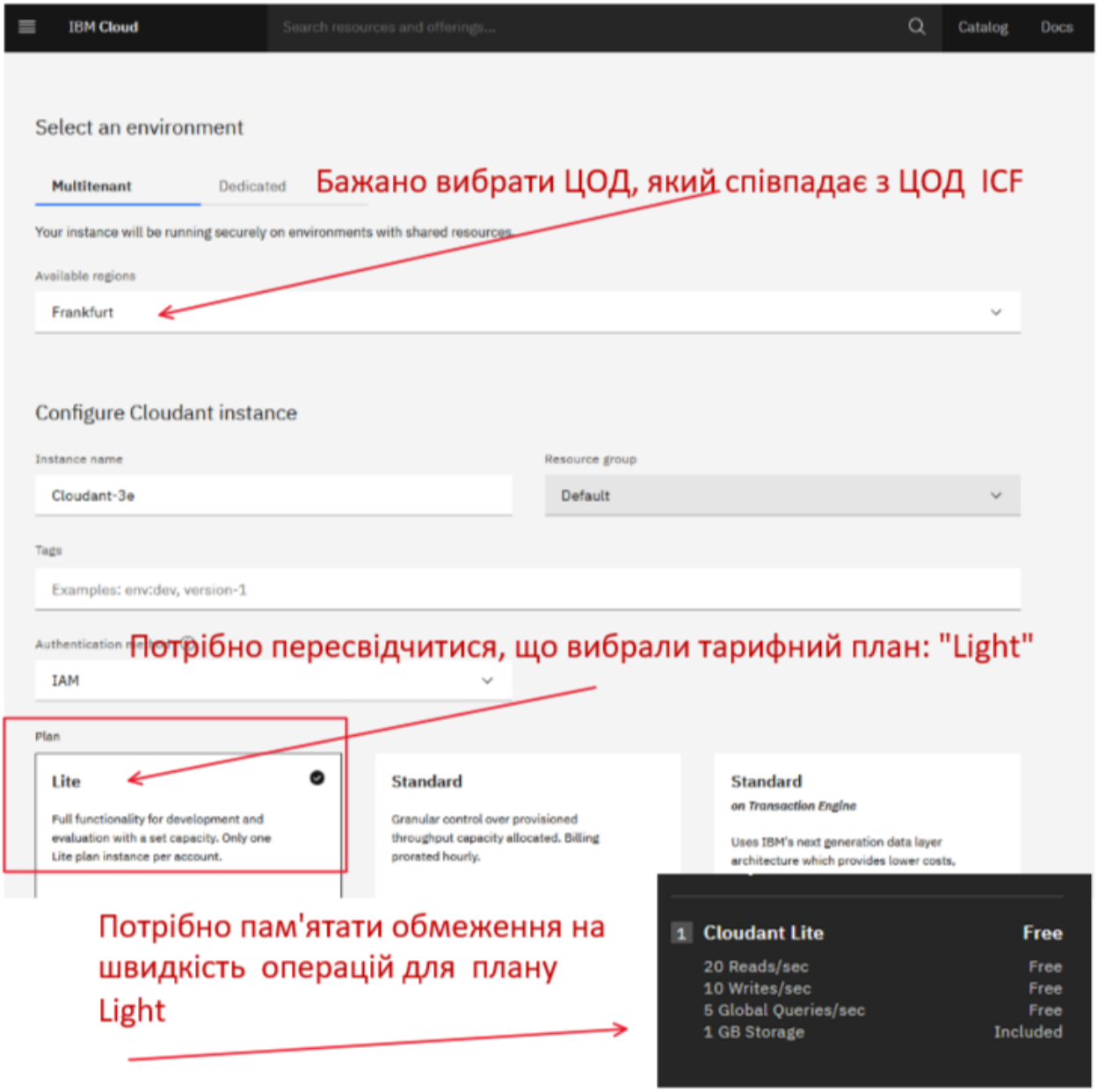
https://gist.github.com/ashvayka/4735c78541cebd26f3ed3d340743c697

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

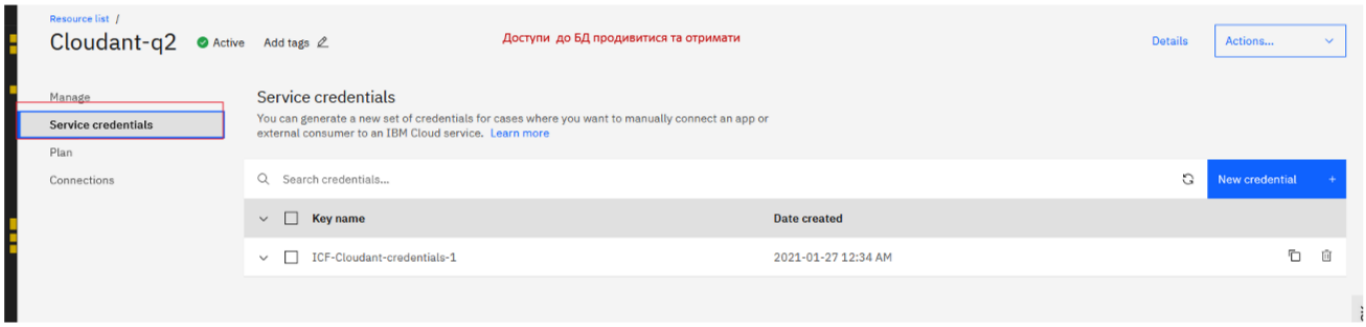
Екран показаний на pic-1



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'язуваня 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:

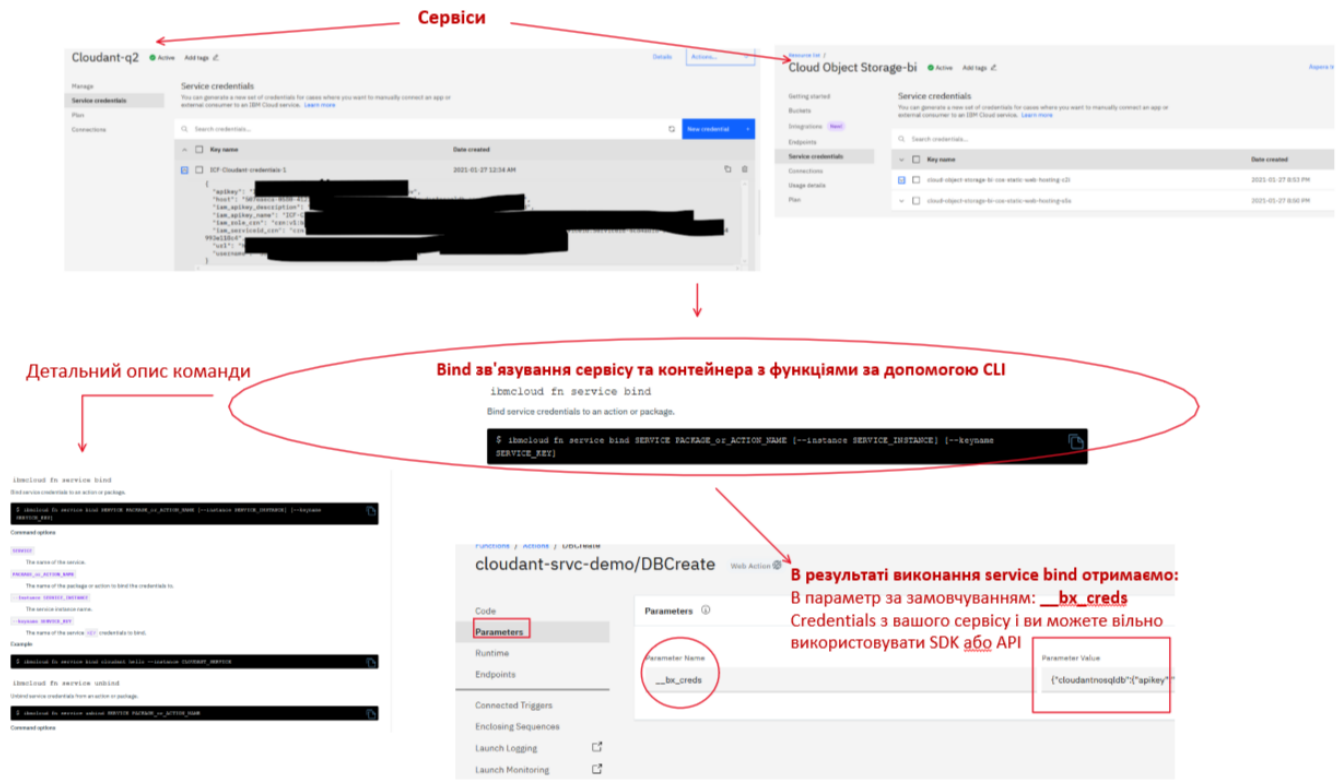
```
ibmcloud target -g Default
```

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 pic-4 показана діаграма, що можливо спросить розуміння цього процесу.



pic-4

Тепер створимо пакет IVF та виконаємо звязування 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
```

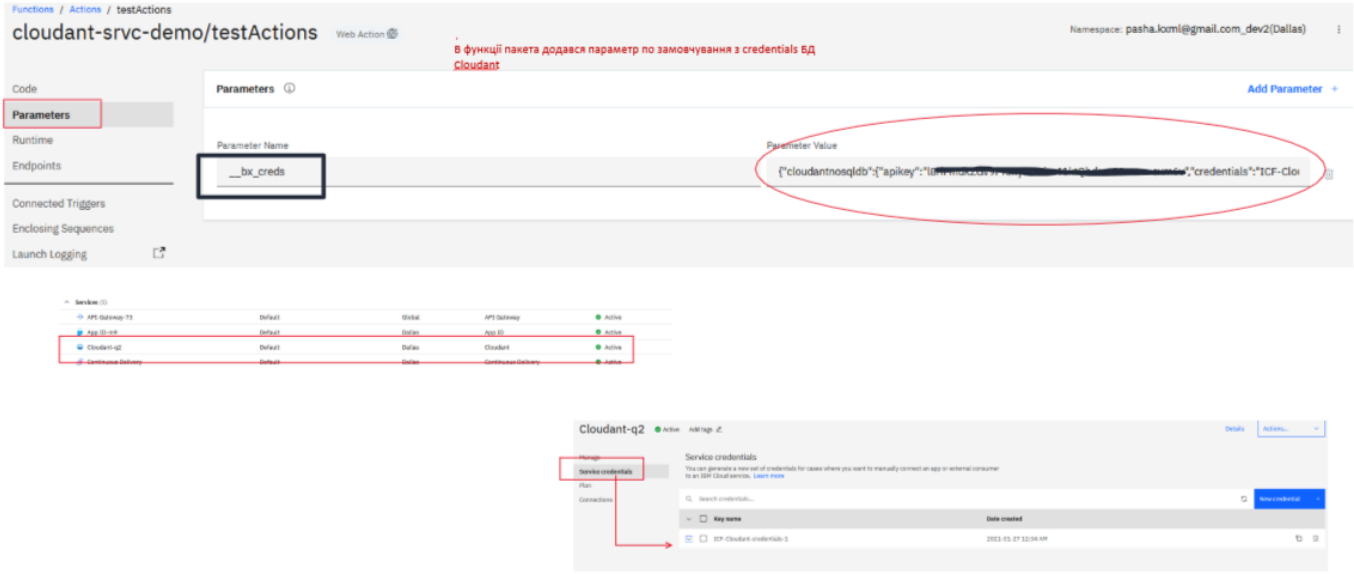
- Звяжемо пакет за БД Cloudant

```
ibmcloud fn service bind cloudantnosqldb  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 developent 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 пересідчитися що вони працюють

Модель колективної розробки

Призначення каталогів.

- deployment Набір скриптів для deployemnt З приводу deployment цікаві такі лінки: Deploying entities with a manifest file

Cloud Functions CLI reference

- doc Документациія
- 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 вайлу 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 vbx_creds = require('../params.localdev/cloudant.json');
    vprm.__bx_creds = vbx_creds;
```

```
    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
```

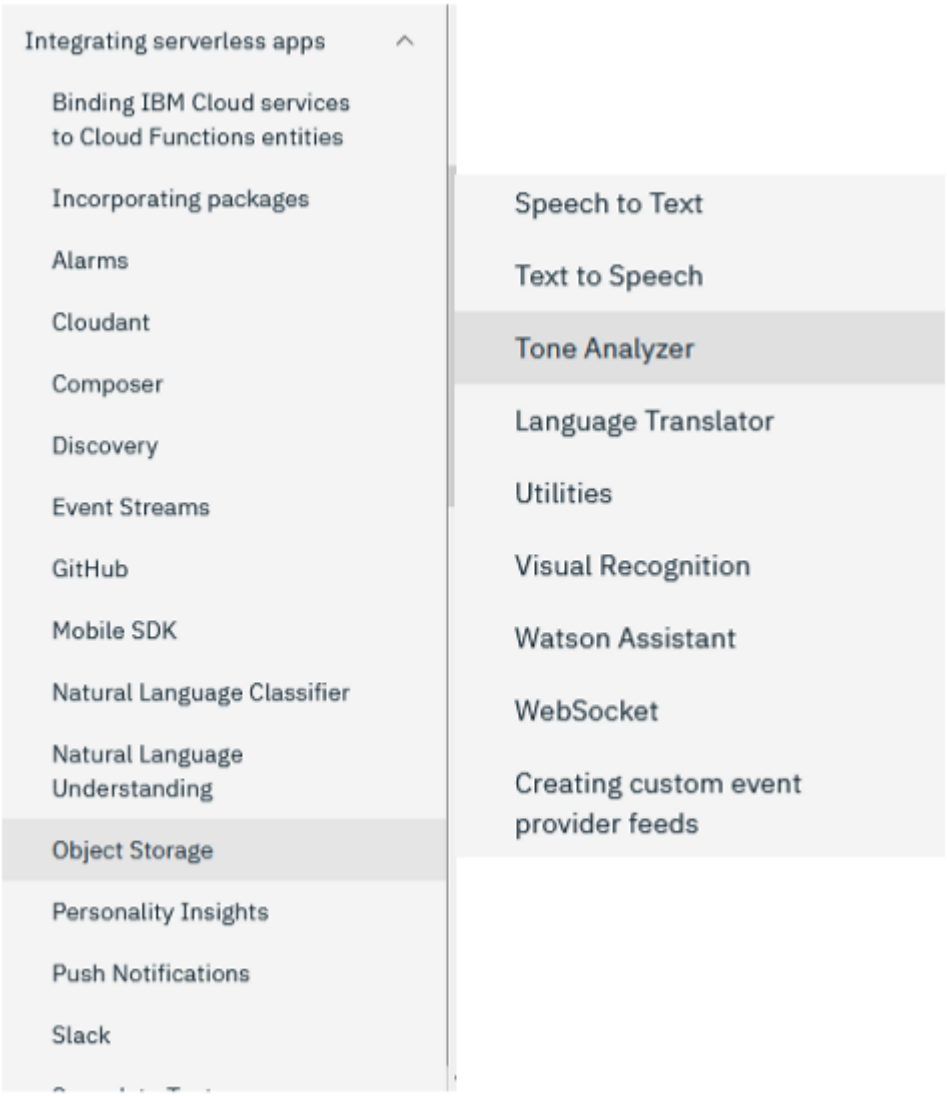
В кореневому каталозі package.json з залежностями та manifest.yml з описом deployment пакету.

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

Використання SDK та API для сервісів

За ICF можна безшовно поєднати такі сервіси в хмарі IBM

На pic-6 показано перелік.



pic-6

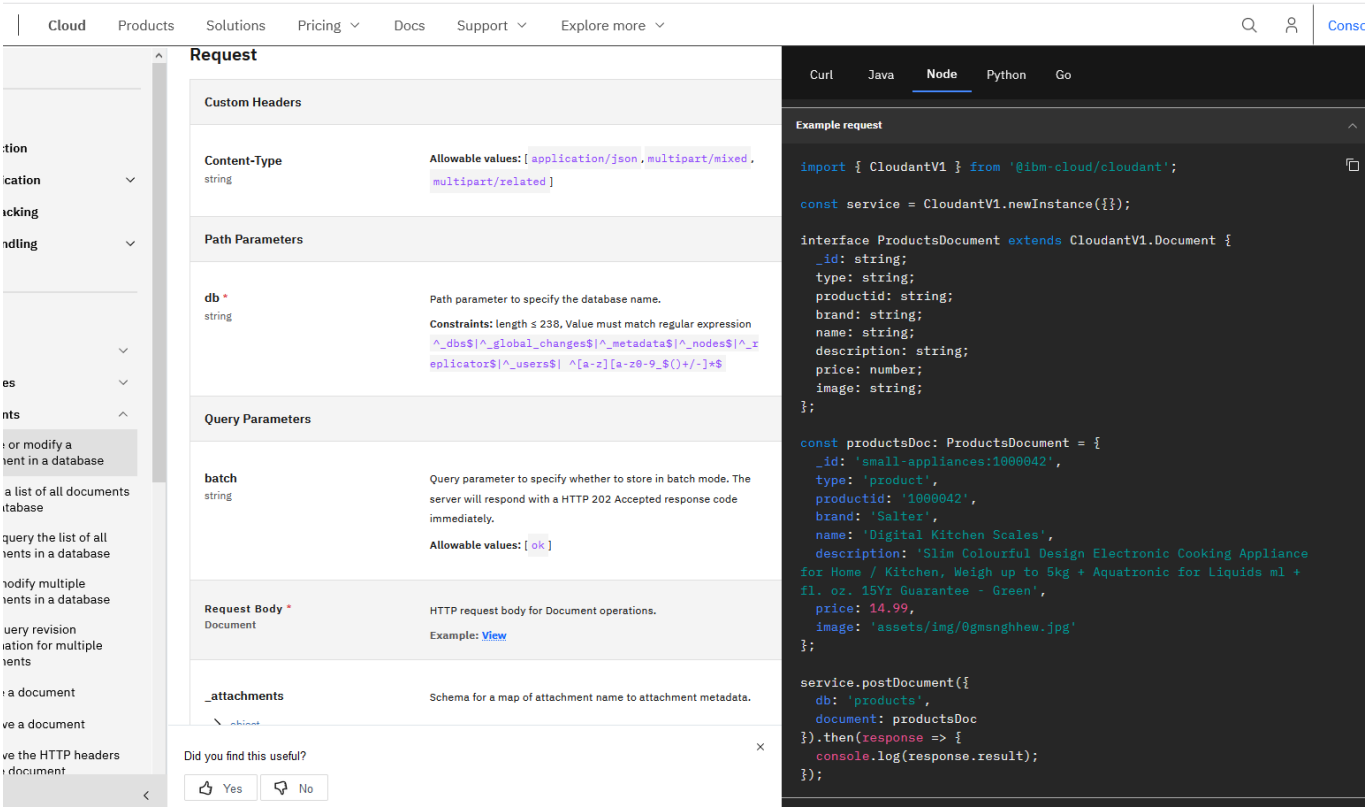
Більш детально, можете почитати за лінком Binding IBM Cloud services to Cloud Functions entities

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 линком

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

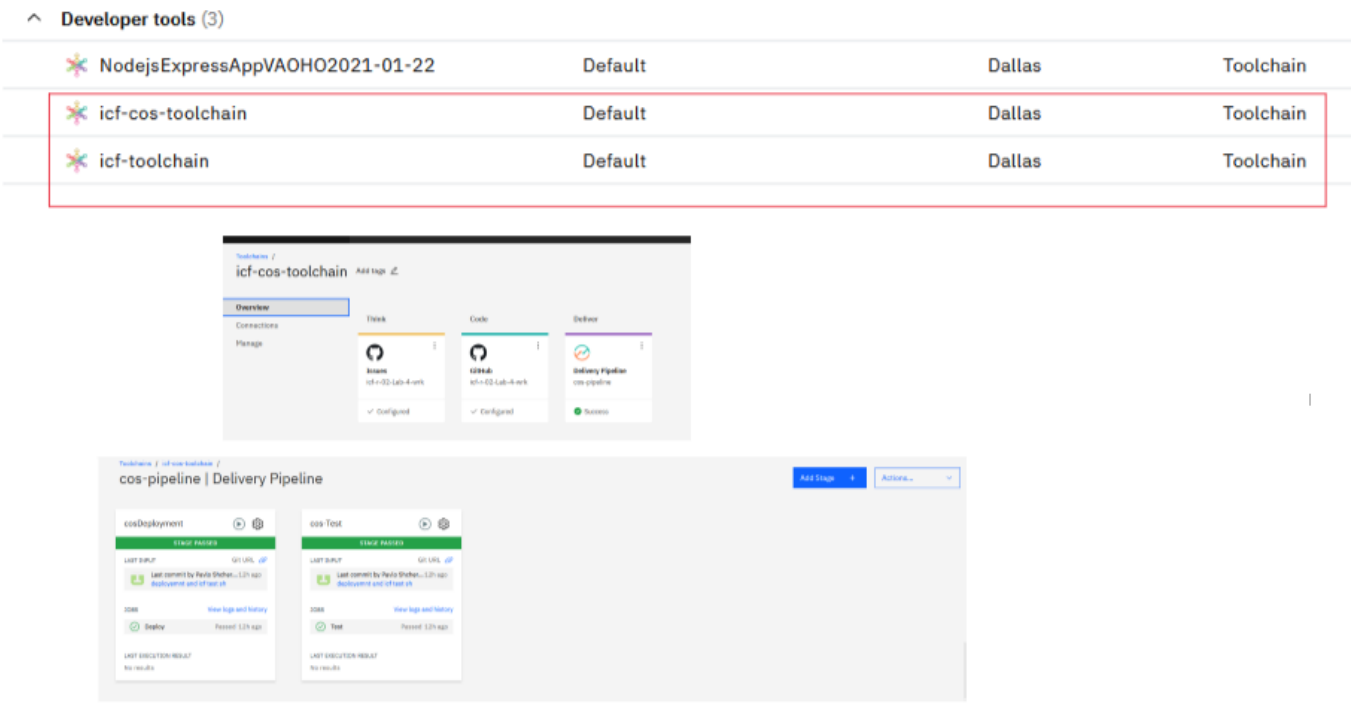


pic-7

Відладку функції виконуємо через запуск тестових кейсів.

Deploeyment з GitHub - подібного репозиторію

Для Deployemnt вкористовується класична toolchain . Її складові показані на [pic-8]



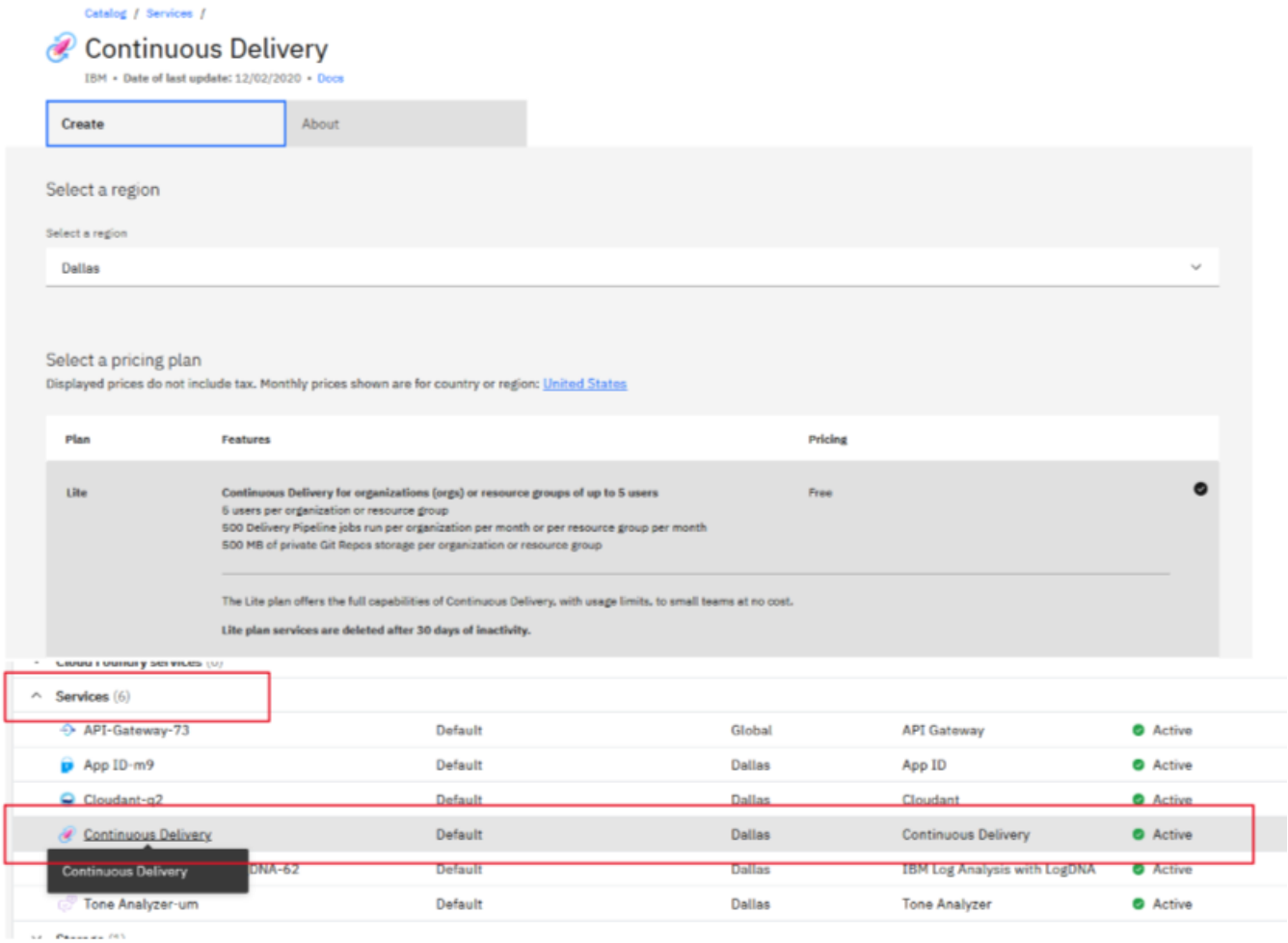
pic-8

Процес deployemnt складається з двох кроків:

- deployemnt
- test

Для встановлення toolchain потрібно виконати такі кроки

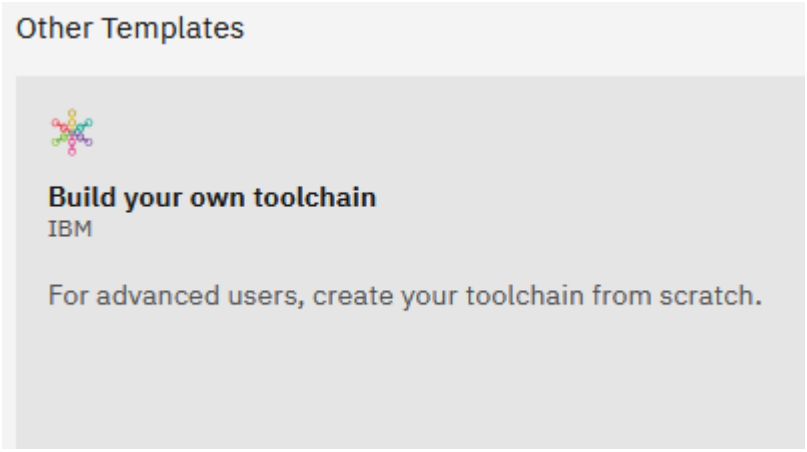
Підключити сервіс Continuous Delivery



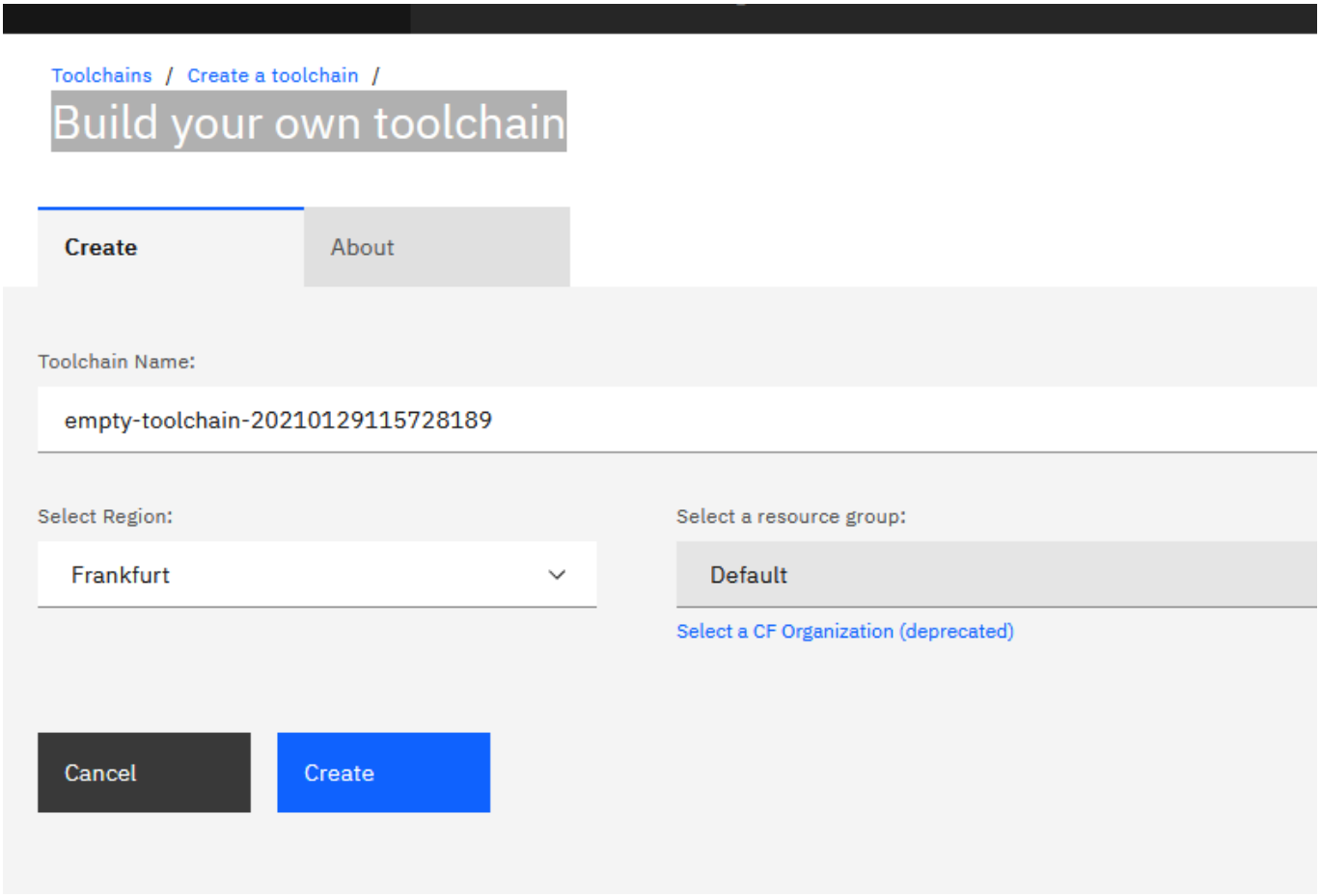
pic-9

Створити Toolchain

З списку потрібно вибрати: Build your own toolchain

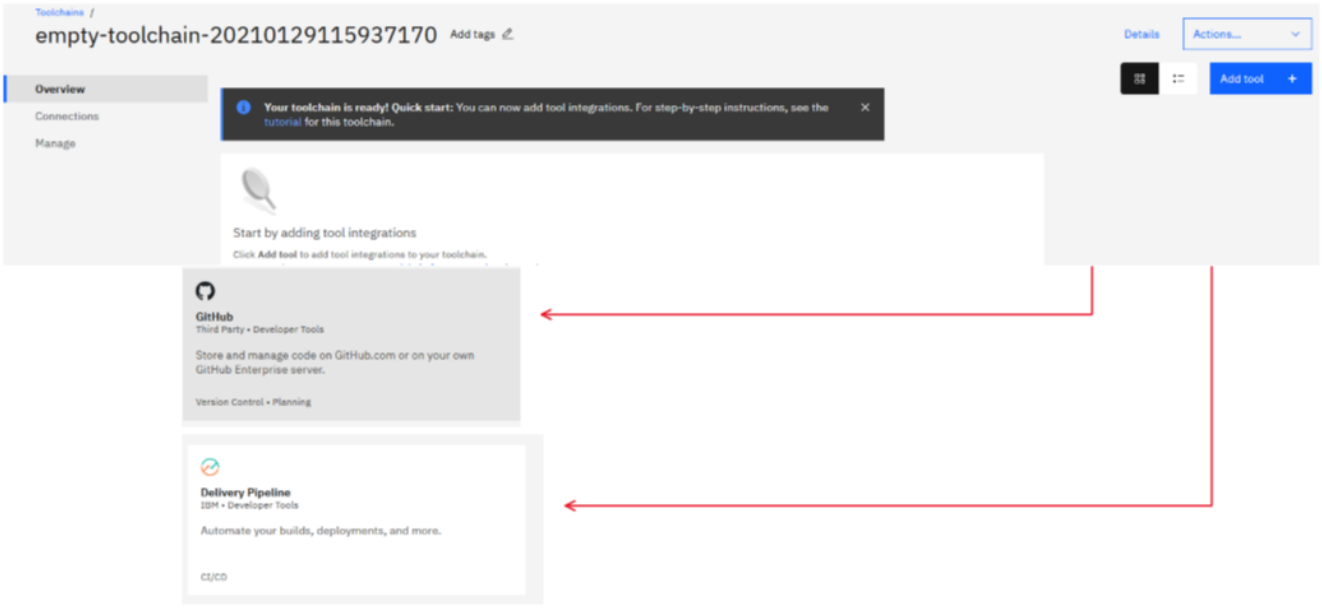


pic-10



pic-11

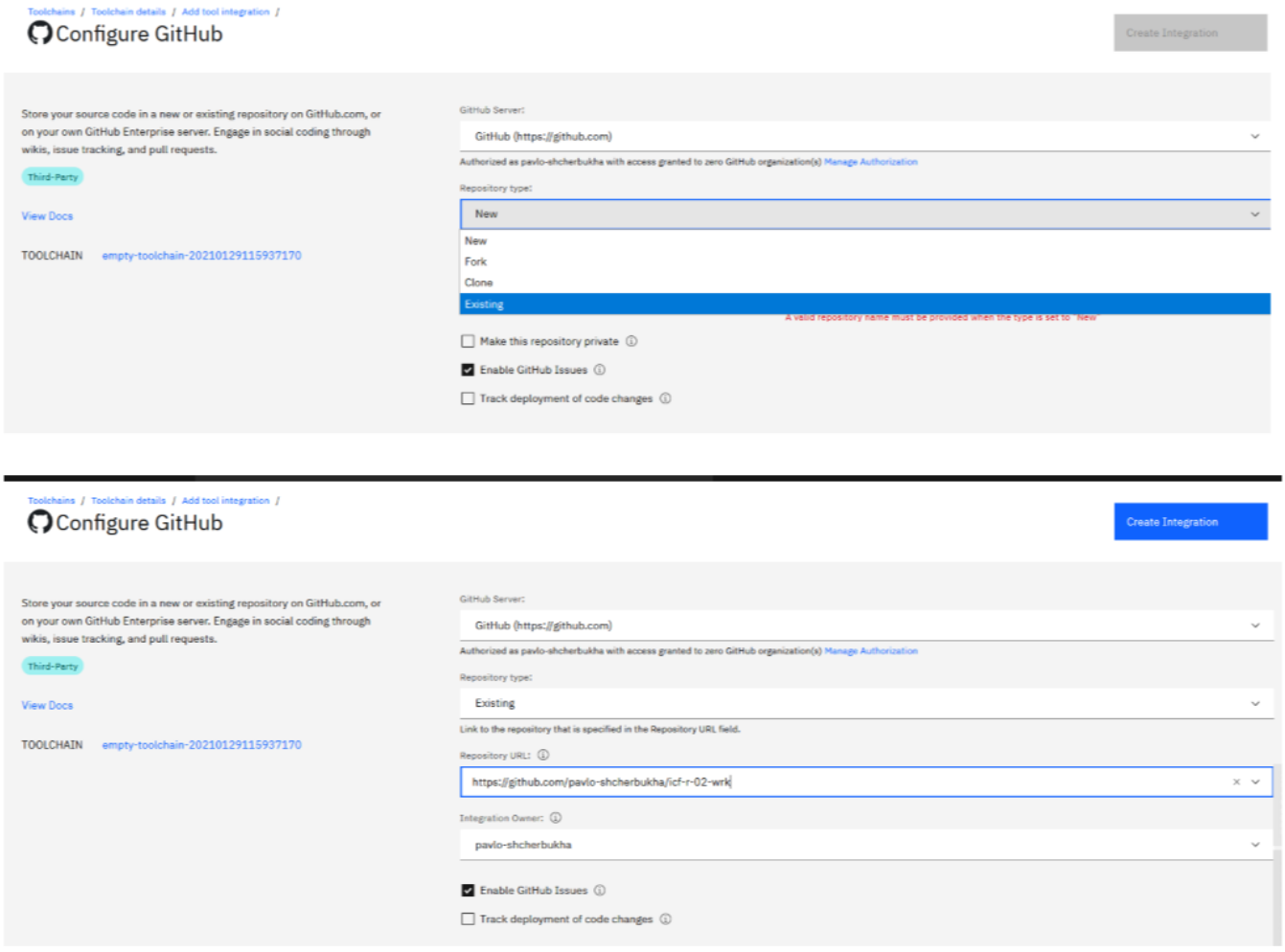
Потім по черзі додаємо Github Integration та delivery pipeline [pic-12]



pic-12

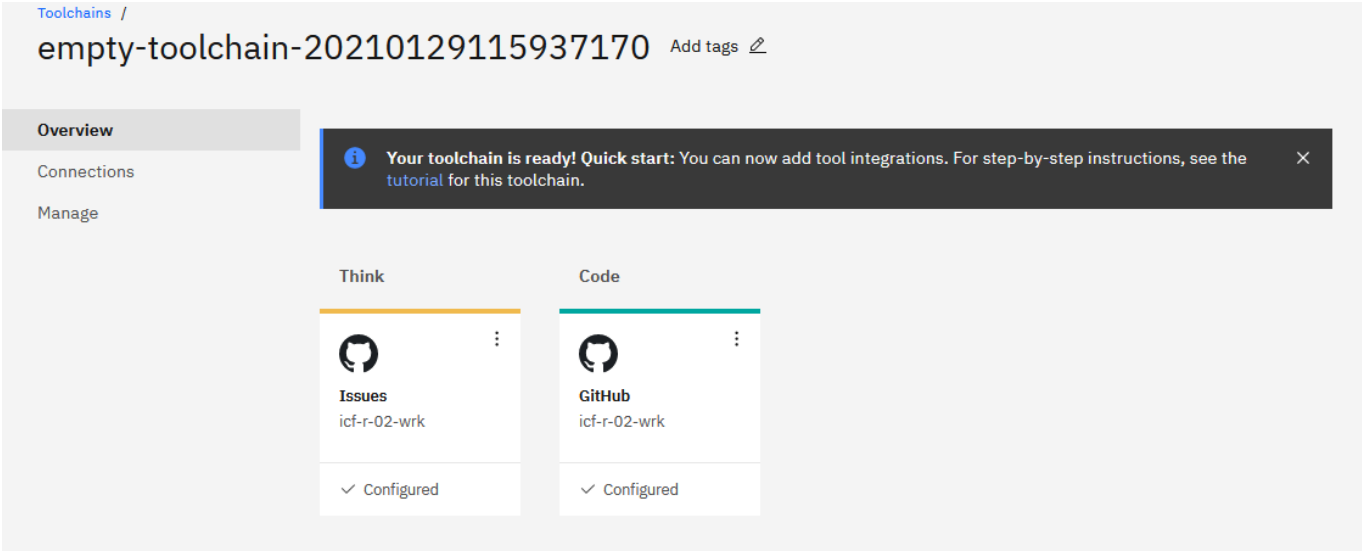
Настройка інтеграції з github

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

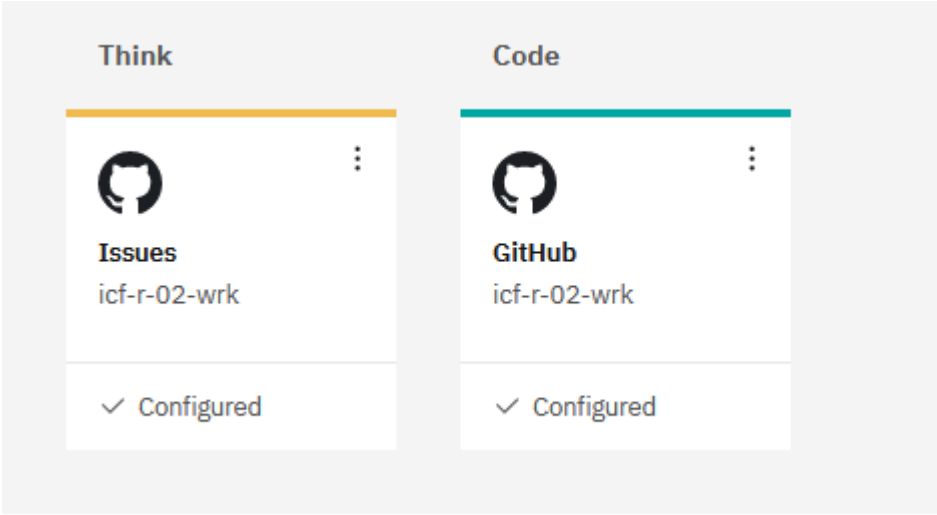


pic-13

та що отримаємо в результаті



pic-14

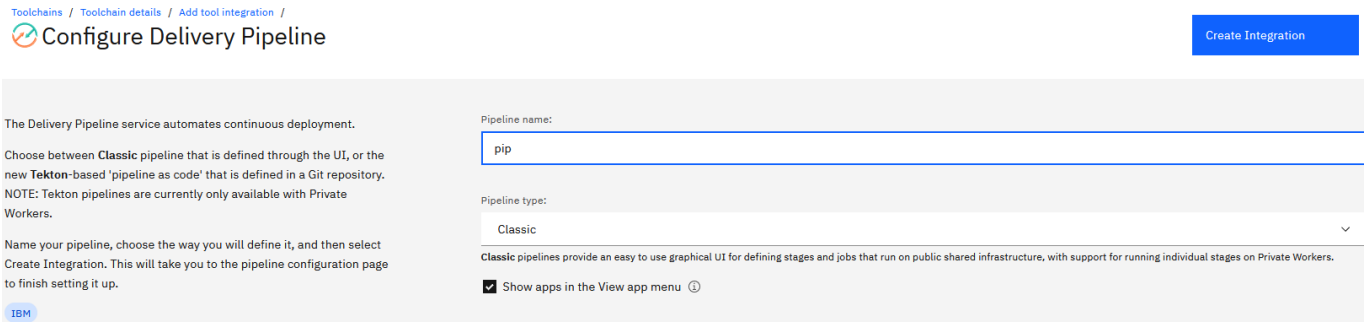


pic-15

Настройка Delivery Pipeline

Додати Delivery Pipeline та вказати найменування

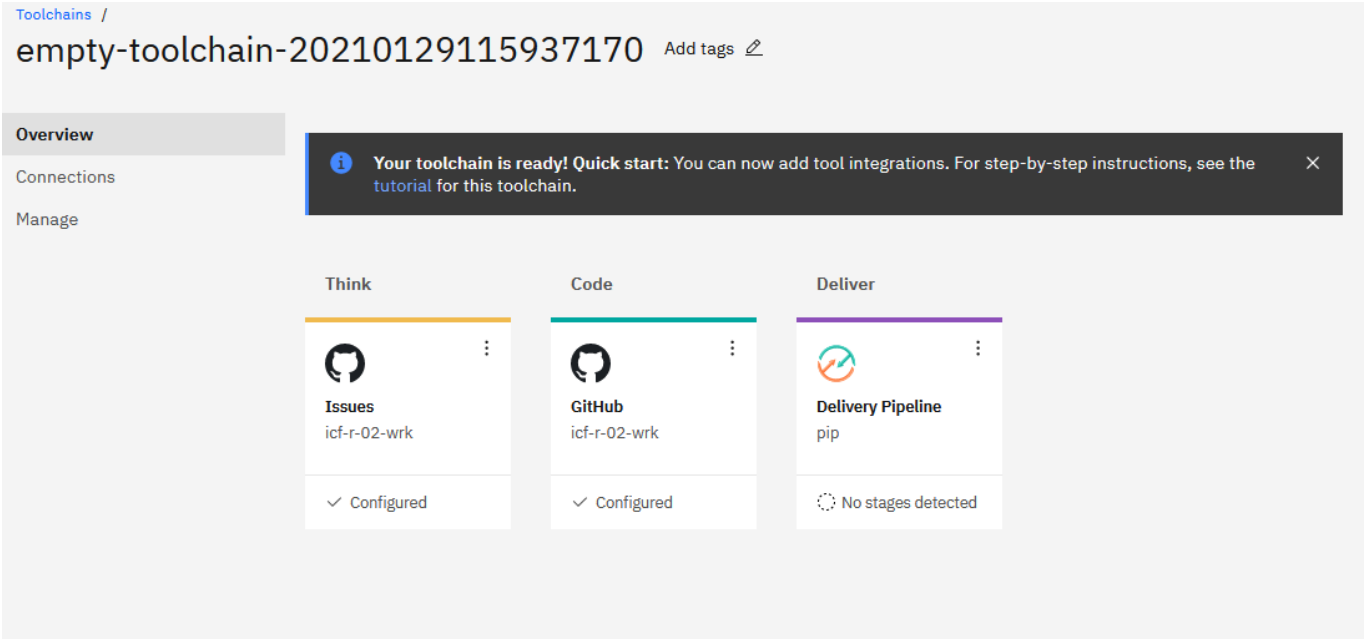
Додати Delivery Pipeline та вказати найменування



pic-16

Створити стейджі Delivery Pipeline

Для цього додати стейджі через кнопку "Add Stage"

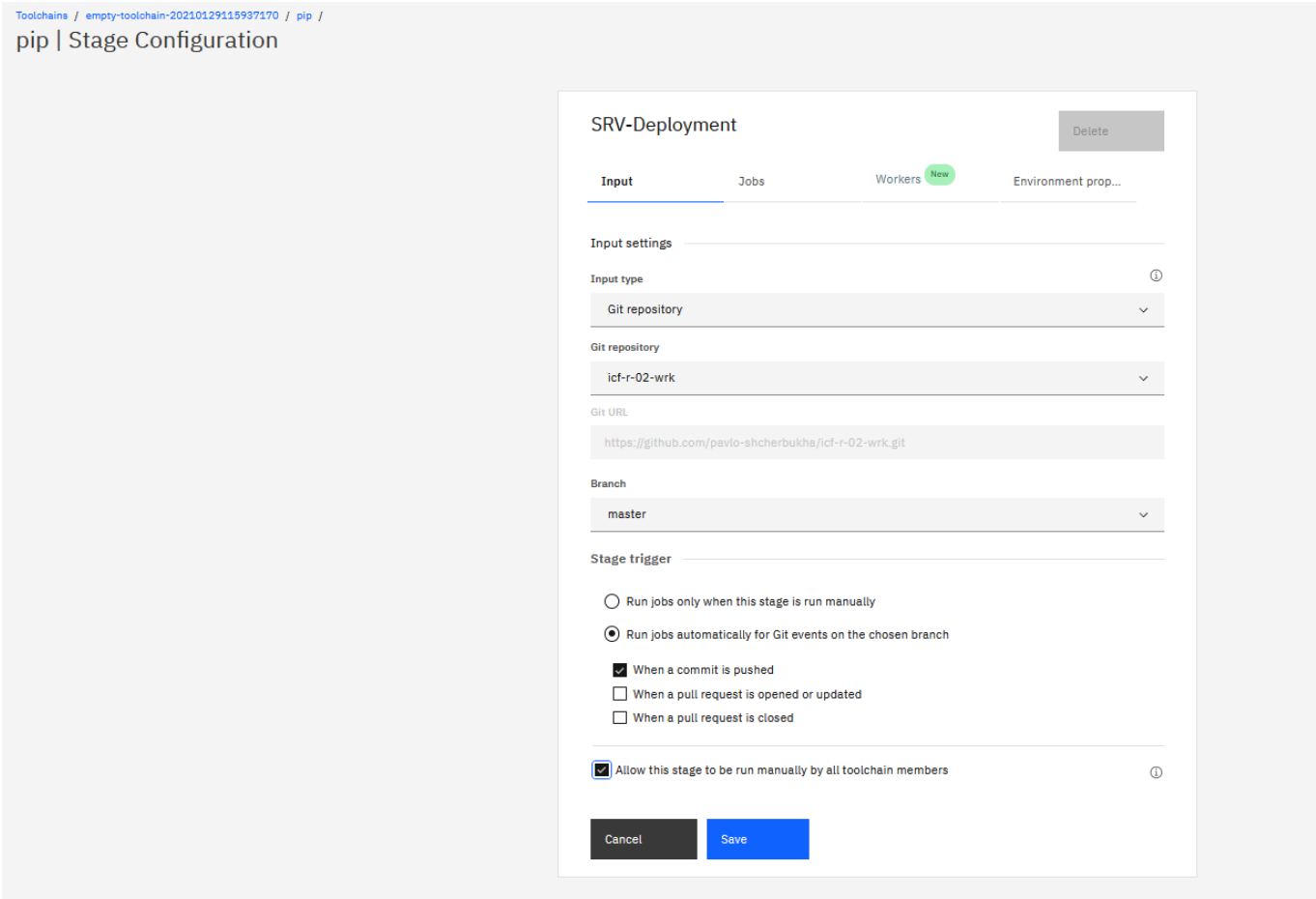


pic-17

pic-17

Створення Deployemnt Stage

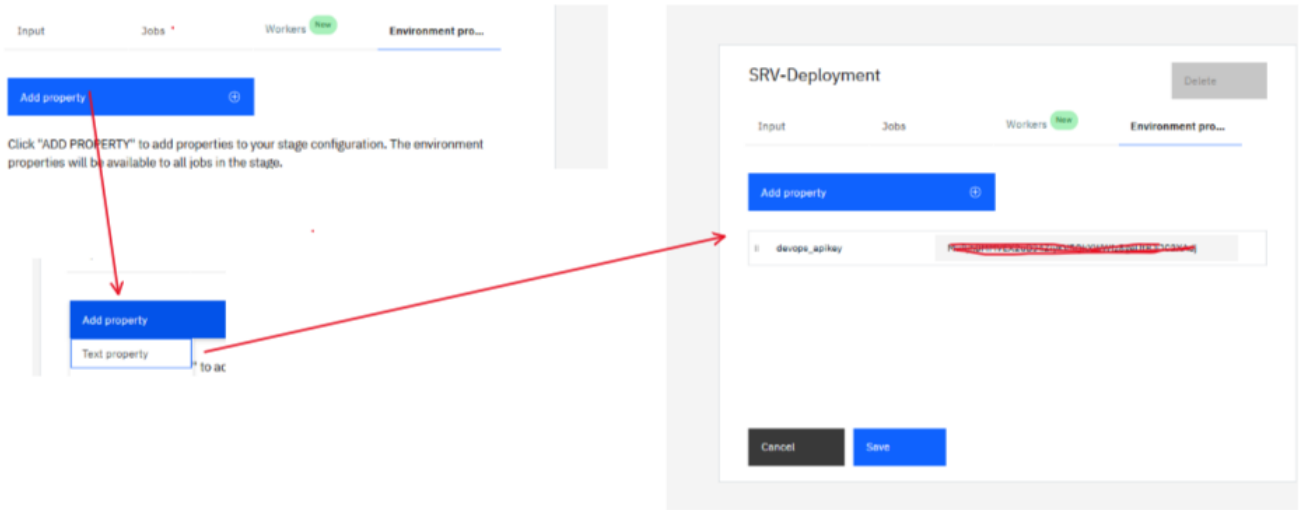
- Спершу потрібно настроїти вкладку Input для Deployemnt Stage, що забирає пограмний код з git



pic-19

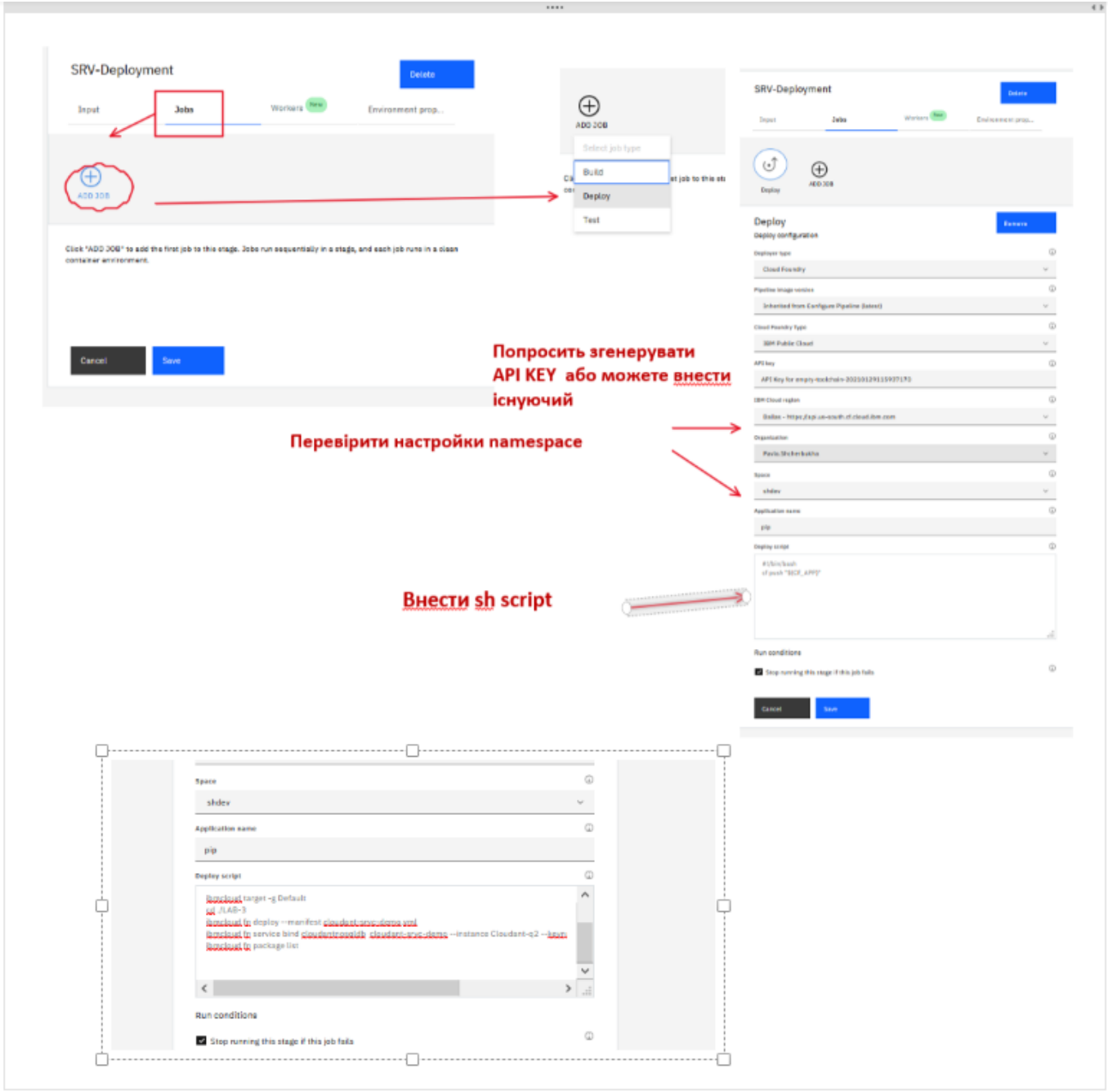
Та вибати опції, як показано.

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



pic-20

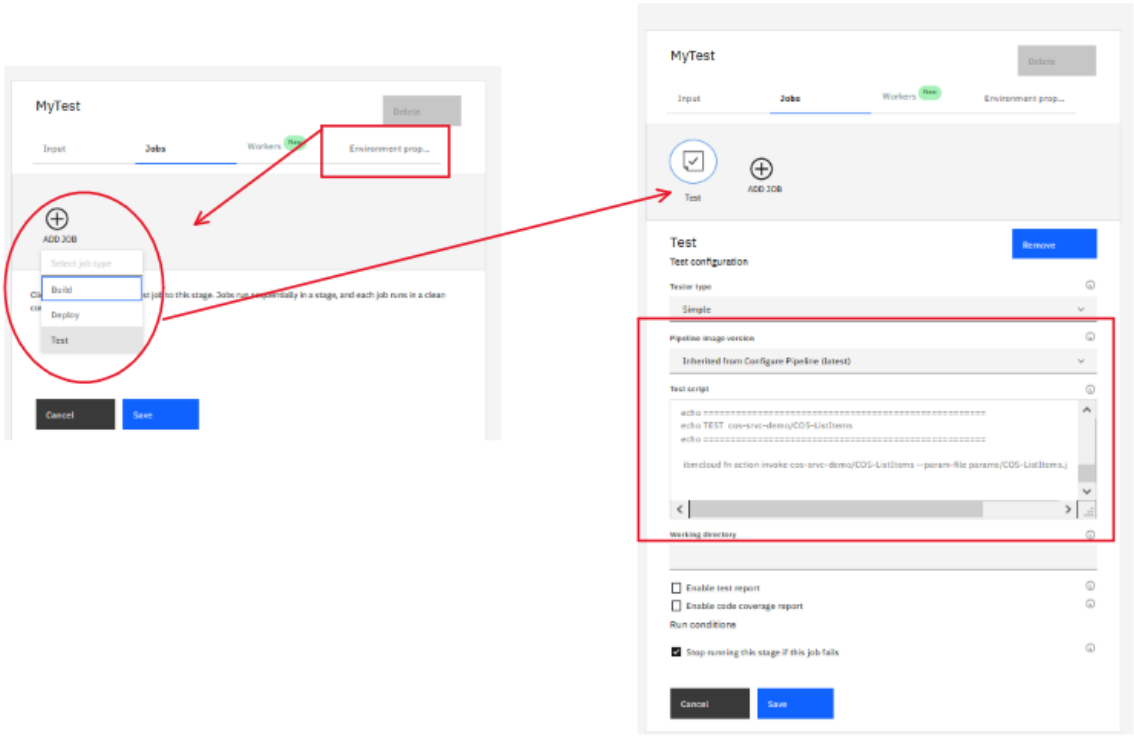
- Зайти у вкладку JOBS, створити deployment job та вставити в нього .sh скрипт з папки ./deployemnt/deploy-job.sh



pic-21

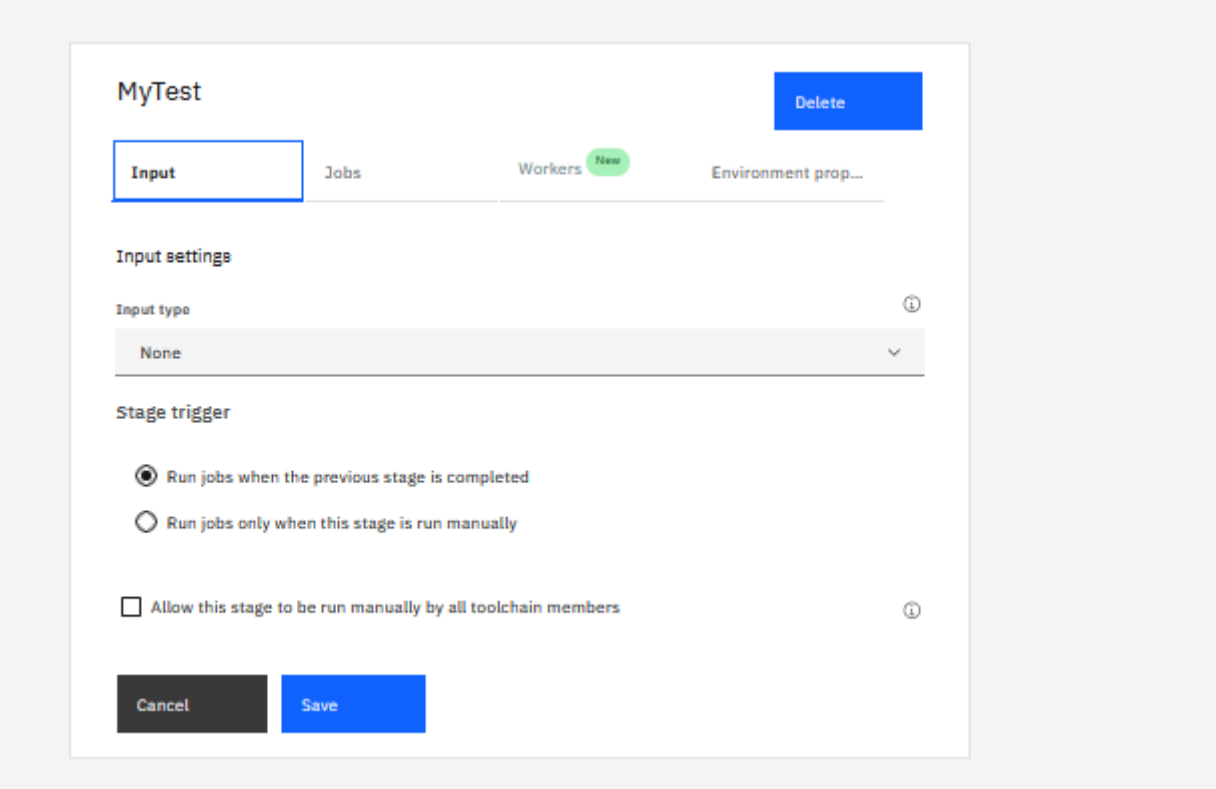
Створення Test Stage

- Додати ще stage та назвати її Test
- У вкладці environemt внести api key
- У вклаці JOB додати test job
- Вклеяти sh скрипт з файлу ./deployemnt/test-job.sh



pic-22

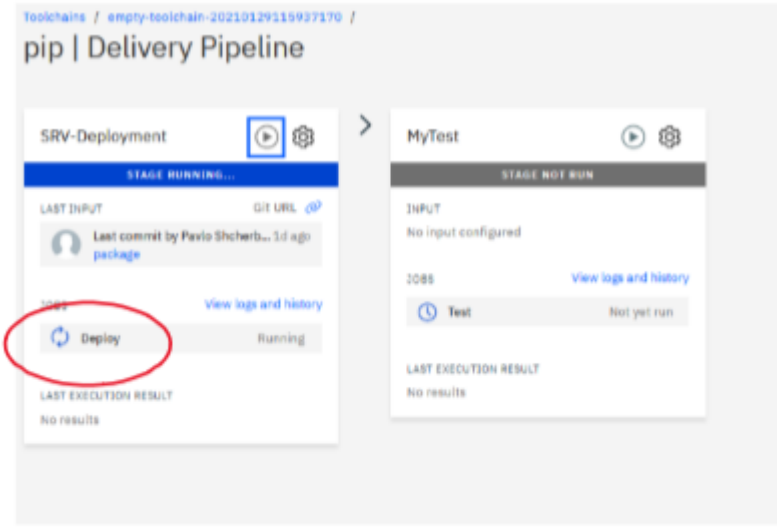
- настроїти вкладку Input

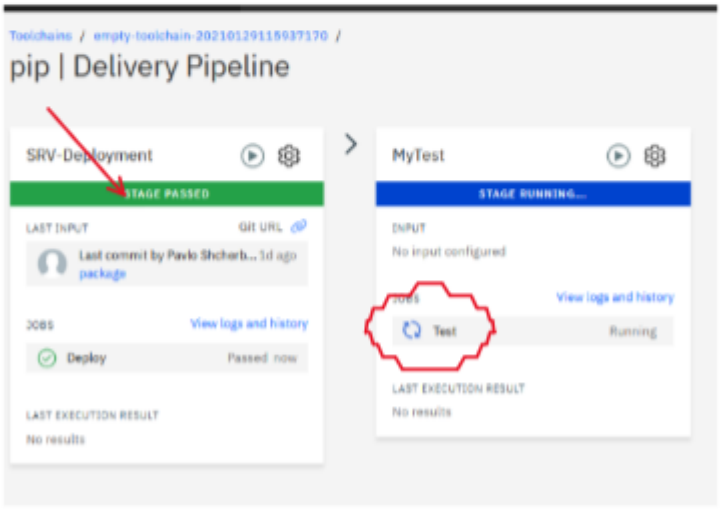


pic-23

Запустити Stages вручну для перевірки

Результати показані на pic-24


Toolchains / empty-toolchain-20210129115937170 /
pip | Delivery Pipeline
SRV-Deployment
STAGE RUNNING...
LAST INPUT
GIT URL
Last commit by Pavlo Shcherb... 1d ago
package
JOBS
View logs and history
Deploy
Running
LAST EXECUTION RESULT
No results
MyTest
STAGE NOT RUN
INPUT
No input configured
JOBS
View logs and history
Test
Not yet run
LAST EXECUTION RESULT
No results


Toolchains / empty-toolchain-20210129115937170 /
pip | Delivery Pipeline
SRV-Deployment
STAGE PASSED
LAST INPUT
GIT URL
Last commit by Pavlo Shcherb... 1d ago
package
JOBS
View logs and history
Deploy
Passed now
LAST EXECUTION RESULT
No results
MyTest
STAGE RUNNING...
INPUT
No input configured
JOBS
View logs and history
Test
Running
LAST EXECUTION RESULT
No results

pic-24