PROJEKTOWANIE SYSTEMÓW OBIEKTOWYCH I ROZPROSZONYCH

LABORATORIUM 7

Continuous Integration & Deployment

wersja 1.0

przygotował:
Radosław Adamus

.

HISTORIA WERSJI

DATA	WERSJA	AUTOR	OPIS
27.04.2014	0.1	Radosław Adamus	Pierwsza robocza wersja dokumentu
28.04.2014	1.0	Radosław Adamus	Pierwsza oficjalna wersja dokumentu

Cel:

Celem laboratorium jest:
Konfiguracja procesu ciągłej integracji (CI) wykorzystującego usługi w chmurze obliczeniowej.

Wymagania wstępne:

1. Posiadanie konta na platformach Github, Travis-ci oraz Cloud9 (opcjonalnie).
2. Skonfigurowane konto AWS

Narzędzia:

Git, nodeJS, edytor programistyczny (np. Notepad++), konsola AMC, travis command line.

Reguły wykonywania ćwiczeń laboratoryjnych:

1. Po ukończeniu laboratorium należy wyłączyć wszystkie działające instancje EC2, i/lub wyzerować ustawienia dotyczące docelowej, minimalnej i maksymalnej liczby instancji w usłudze ASG.
2. Otrzymane dane autoryzacyjne (hasła oraz klucze dostępu) są danymi wrażliwymi i muszą być chronione. W szczególności nie można dodawać do repozytorium kontroli wersji oraz pozwolić na wysłanie na usługi hostujące repozytoria kontroli wersji kodu źródłowego (GitHub) plików zawierających konfigurację autoryzacji dostępu do API AWS.
3. Zmiany zatwierdzane w repozytorium kontroli wersji powinny mieć znaczące komentarze.

Opis laboratorium:

1. Zadania

1. (opcjonalnie) Utwórz projekt (workspace) na platformie cloud9.
2. Utworzenie i konfiguracja repozytorium kodu projektu.
    2.1. Zainicjalizuj repozytorium kodu na platformie GitHub

    opcje:

nazwa: ciTest[Nazwisko_bez_polskich_znaków], .gitignore: node, licence: MIT, plik readme.

2.2. Pobierz (git clone) projekt do przestrzeni roboczej (lokalny komputer lub cloud9).

3. Utworzenie i konfiguracja projektu nodejs

3.1. Utwórz plik app.js oraz podkatalog test w folderze projektu

3.2. Zainicjalizuj projekt dla nodejs wykorzystując manager pakietów npm.

npm init

opcje: test script: mocha

Zatwierdź zmiany w repozytorium i wypchnij je do zdalnego repozytorium (git push)

4. Utworzenie testu i inicjalizacja środowiska testowego

4.1. W folderze test utwórz plik test.js i wprowadź do niego poniższy kod:

```
var assert = require("assert");
var discountCalculator = require("../discount");


describe("discount Calculator", function(){
    it("should have discount function", function(){
        assert.equal(discountCalculator.discount  instanceof  Function,  true,
"typeof discount: "  + typeof discountCalculator.discount);
    });
});
```

4.2. Z poziomu linii poleceń uruchom komendę `mocha`. (W przypadku jak braku zainstaluj ją poleceniem `npm install -g mocha`)

4.3. Spraw aby test wykonał się poprawnie. Utwórz, w głównym folderze projektu plik discount.js i wprowadź do niego poniższy kod.

```
module.exports.discount = function(){    }
```

Po poprawnym wykonaniu testu, zatwierdź zmiany w repozytorium. Prześlij je do zdalnego repozytorium.

5. Konfiguracja systemu ciągłej integracji (travis ci)

5.1. Utwórz konto w usłudze travis (https://travis-ci.org).

5.2. Z poziomu linii poleceń uruchom komendę `travis login`.

5.2.1. Jeżeli komenda nie jest zainstalowana, należy zainstalować środowisko Ruby (http://rubyinstaller.org/downloads/ - wersja 2.0.0). W celu kontroli poprawności instalacji wykonaj komendy: `ruby -v` oraz `gem -v`. Następnie, w celu zainstalowania linii komend dla systemu travis wykonaj polecenie:

```
gem install travis --no-rdoc --no-ri
```

5.3. Z poziomu głównego folderu projektu wykonaj polecenie `travis init`.

5.4. Dokonaj edycji utworzonego pliku .travis.yml, usuń wersje node_js 0.11 i dodaj poniższą konfigurację:

before_install: npm install -g mocha

5.5. Zatwierdź zmiany w repozytorium i prześlij je do usługi GitHub.

5.6. Sprawdź w usłudze travis rezultaty (może to potrwać kilka minut).

6. Konfiguracja umieszczania (deploy) zbudowanych rezultatów integracji na s3.

6.1. Z poziomu głównego folderu projektu wykonaj polecenie `travis setup s3`.

Opcje:

Bucket: deploy-weeia, S3 upload directory: [nazwisko_bez_polskich_znaków]

6.2. Zatwierdź zmiany w repozytorium i prześlij je do usługi GitHub. Sprawdź działanie usługi travis i rezultat w usłudze S3.

7. Dalszy rozwój projektu

7.1. Dodaj do pliku test.js poniższy test, uruchom go i spraw, aby wykonał się poprawnie:

```
describe("discount function", function(){

    it("discount should return 0 with no args  ", function(){
       var discount = discountCalculator.discount();
       assert.equal(discount, 0, discount + "");
    });
});
```

7.2. Zatwierdź zmiany i prześlij do usługi GitHub.

8. (Opcjonalne) Konfiguracja publikowania pakietów npm

8.1. zapoznaj się z zasadami działania systemu npm oraz publikowaniem, za jego pośrednictwem, własnych pakietów (patrz linki w materiałach)

8.2. Skonfiguruj usługę Travis-ci, tak aby publikowała pakiet w npm (http://docs.travis-ci.com/user/deployment/npm/) ale tylko wówczas, gdy wersja

kodu w repozytorium będzie oznaczona tag'iem (git tag …, git push origin master --tags).

Materiały:

1. Dokumentacja Travis-ci: http://docs.travis-ci.com/user/getting-started/
2. Travis-ci S3 deplyment: http://docs.travis-ci.com/user/deployment/s3/
3. npm guide: https://www.npmjs.org/doc/developers.html
4. Jak publikować własny pakiet za pomocą npm: http://quickleft.com/blog/creating-and-publishing-a-node-js-module