­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота №2

з курсу «Управління ІТ-проектами»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 11

Виконав студент гр. КНз-41

Чалий Михaйло

Перевірив

Денисюк П.Ю.

­­

Львів 2016

## Мета роботи

Мета роботи – ознайомитись з системами автоматизованоих збірок проектів. Зрозуміти принципи їх роботи. Набути практичних навичок автоматизації збірки проекту на прикладі інструментів.

## Теоретичні відомості

Автоматизація збірки – етап написання скриптів або автоматизація широкого спектру завдань стосовно до ПО, вживаному розробниками в їх повсякденній діяльності, включаючи такі дії, як:

компіляція вихідного коду в бінарний код; збірка бінарного коду; виконання тестів; розгортання програми на виробничій платформі; написання супровідної документації або опис змін нової версії.

Історично так склалося, що розробники застосовували автоматизацію збірки для виклику компіляторів і лінковщиків зі скрипта збірки, на відміну від виклику компілятора з командного рядка. Досить просто за допомогою командного рядка передати один вихідний модуль компілятору, а потім і лінковщик для створення кінцевого об'єкта. Однак, при спробі скомпілювати або злінкувати безліч модулів з вихідним кодом, причому в певному порядку, здійснення цього процесу вручну за допомогою командного рядка виглядає занадто незручним. Набагато привабливішою альтернативою є скриптова мова, підтримувана утилітою Make. Даний інструмент дозволяє писати скрипти збірки, визначаючи порядок їх виклику, етапи збирання та компонування для збірки програми. GNU Make також надає такі додаткові можливості, як наприклад, «залежності» («makedepend»), які дозволяють вказати умови підключення вихідного коду на кожному етапі складання. Це і стало початком автоматизації збирання. Основною метою була автоматизація викликів компіляторів і лінковщиків. У міру зростання і ускладнення процесу складання розробники почали додавати дії до і після викликів компіляторів, як наприклад, перевірку (checkout) версій копійованих об'єктів на тестову систему. Термін «автоматизація складання» вже включає в себе управління і дії до і після збирання та компонування, так само як і дії при компіляції і лінковці.

## Завдання

## Ознайомитись з принципами роботи систем автоматизації збірок проектів.

## Організувати автоматизацію збірки проекту зробленого у попередній лабораторній роботі.

## Реалізація

Для сборки використовується gulp.js:

1. Спеціальний дев кофігурація яка перезавантажує апплікацю коли щось змінюється.
2. Деплоймент на Amazon Beanstalk, з використанням Docker

Лістінг 1. gulp.js

var gulp = require('gulp');

var nodemon = require('gulp-nodemon');

var plumber = require('gulp-plumber');

var livereload = require('gulp-livereload');

var exec = require('gulp-exec');

var sequence = require('run-sequence');

gulp.task('dev', () => {

livereload.listen();

nodemon({

script: 'app.js',

ignore: ['test/\*', 'integration/\*', 'client/\*'],

ext: 'js handlebars',

stdout: false

}).on('readable', function () {

this.stdout.on('data', function (chunk) {

if(/^Express server listening on port/.test(chunk)){

livereload.changed(\_\_dirname);

}

});

this.stdout.pipe(process.stdout);

this.stderr.pipe(process.stderr);

});

});

gulp.task('publish:staging', () => {

return gulp.src('./package.json')

.pipe(exec('eb deploy dev'))

.pipe(exec.reporter());

});

Для оркестрація сборки використовується npm.

Літснг 2. Package.json

{

"name": "studies-octo-adventure-pm",

"version": "1.0.0",

"private": true,

"scripts": {

"start": "node app.js",

"dev": "gulp dev",

"eslint": "eslint app/\*\*/\*.js",

"fix": "eslint app/\*\*/\*.js --fix",

"publish:staging": "gulp publish:staging",

"test": "mocha test/\*\*/\*.js"

},

"dependencies": {

"body-parser": "^1.13.3",

"busboy-body-parser": "0.0.10",

"compression": "^1.5.2",

"cookie-parser": "^1.3.3",

"express": "^4.13.3",

"express-handlebars": "^3.0.0",

"express-session": "^1.13.0",

"glob": "^6.0.4",

"promise": "^7.1.1",

"serve-favicon": "^2.3.0",

"uuid": "^2.0.1"

},

"devDependencies": {

"chai": "^3.5.0",

"eslint": "^1.10.3",

"eslint-config-defaults": "^7.1.1",

"eslint-plugin-filenames": "^0.2.0",

"gulp": "^3.9.0",

"gulp-exec": "^2.1.2",

"gulp-livereload": "^3.8.0",

"gulp-nodemon": "^2.0.2",

"gulp-plumber": "^1.0.0",

"mocha": "^2.4.5"

}

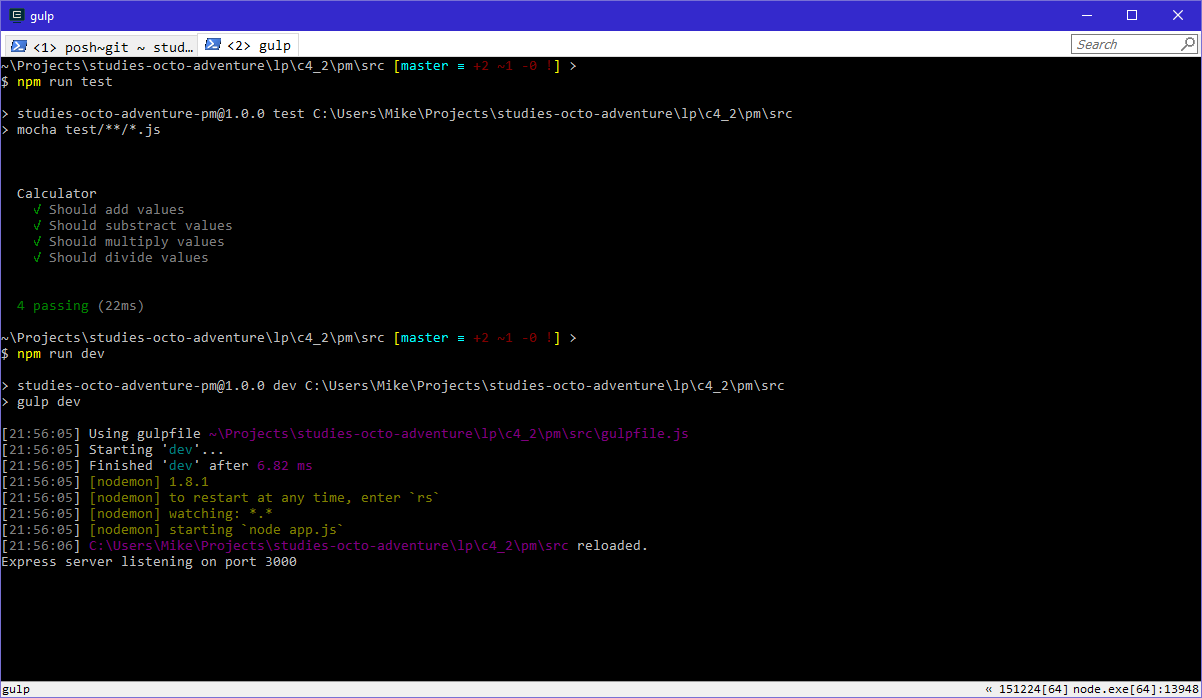
}

## Результат

npm install

npm run test # Unit tests...

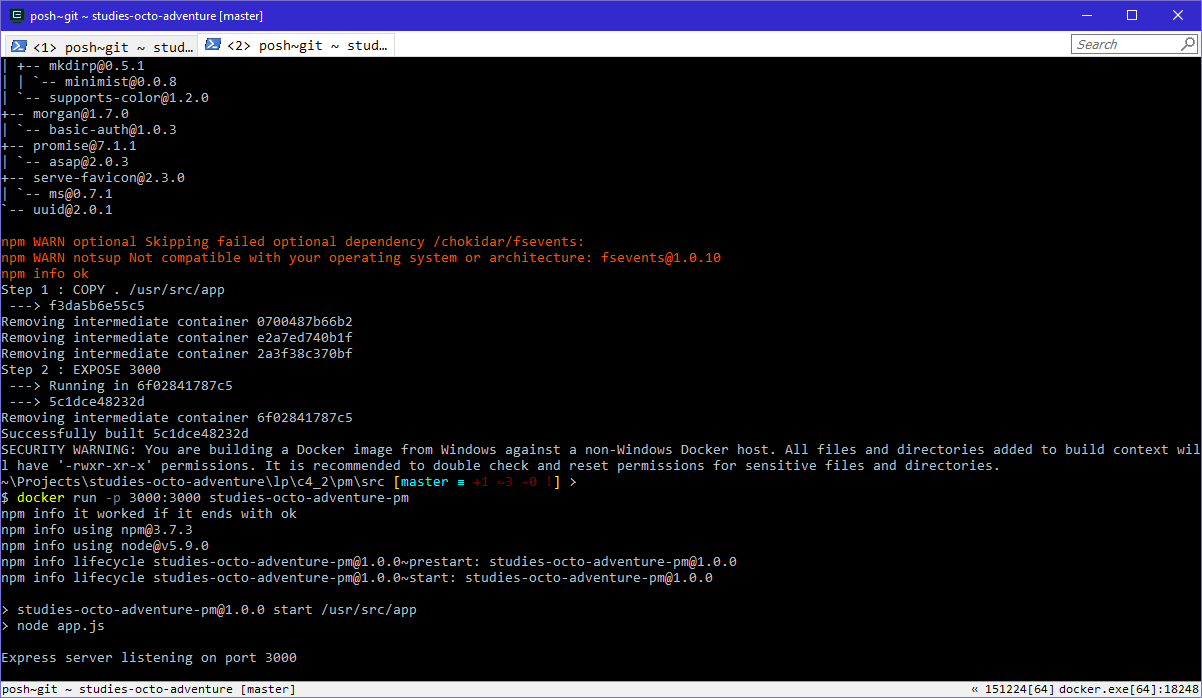
npm run dev # Run application server in dev mode(e.g. reload with something changed in code)’



Для деплоймента в докер

docker build -t studies-octo-adventure-pm .

docker run -p 3000:3000 studies-octo-adventure-pm



Для деплоймента на Amazon Beanstalk

npm run publish:staging

## Висновки

Ознайомився з системами автоматизованоих збірок проектів. Зрозумів принципи їх роботи. Набув практичних навичок автоматизації збірки проекту на прикладі інструментів.