

전자정부표준프레임워크 기반의 Open PaaS 개발

개발 가이드

Node.js 애플리케이션

|  |  |  |
| --- | --- | --- |
| 작성자 | 조민구(인) | 2015.12. |
| 검토자 | (인) |  |
| 승인자 | (인) |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

개 정 이 력

| 버전 | 작성일 | 변경내용[[1]](#footnote-2) | 작성자 | 승인자 |
| --- | --- | --- | --- | --- |
| 0.8 | 2015.10.20 | 최초작성 | 조민구 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목차

[1. 개요 4](#_Toc437868496)

[1.1. 문서 개요 4](#_Toc437868497)

[1.1.1. 목적 4](#_Toc437868498)

[1.1.2. 범위 4](#_Toc437868499)

[1.1.3. 참고 자료 4](#_Toc437868500)

[2. Node.js 애플리케이션개발가이드 5](#_Toc437868501)

[2.1. 개요 5](#_Toc437868502)

[2.2. 개발환경 구성 5](#_Toc437868503)

[2.2.1. Node.js 및 npm 설치 6](#_Toc437868504)

[2.3. 개발 12](#_Toc437868505)

[2.3.1. Node.js Express애플리케이션 생성 12](#_Toc437868506)

[2.3.2. Node.js 샘플 애플리케이션 14](#_Toc437868507)

[2.3.3. 애플리케이션 환경설정 16](#_Toc437868508)

[2.3.4. VCAP\_SERVICES 환경설정 정보 21](#_Toc437868509)

[2.3.5. Mysql 연동 22](#_Toc437868510)

[2.3.6. Cubrid 연동 24](#_Toc437868511)

[2.3.7. MongoDB 연동 26](#_Toc437868512)

[2.3.8. Redis 연동 27](#_Toc437868513)

[2.3.9. RabbitMQ연동 28](#_Toc437868514)

[2.3.10. GlusterFS 연동 29](#_Toc437868515)

[2.4. 배포 32](#_Toc437868516)

[2.4.1. 개방형 플랫폼 로그인 32](#_Toc437868517)

[2.4.2. 서비스 생성 33](#_Toc437868518)

[2.4.3. 애플리케이션 배포 33](#_Toc437868519)

[2.4.4. 애플리케이션, 서비스 연결 34](#_Toc437868520)

[2.4.5. 애플리케이션 실행 35](#_Toc437868521)

[2.5. 테스트 36](#_Toc437868522)

1. 개요
   1. 문서 개요
      1. 목적

본 문서(node.js 애플리케이션 개발 가이드)는 개방형 플랫폼 프로젝트의 서비스팩(Mysql, Cubrid, MongoDB, RabbitMQ, Radis, GlusterFS)을 Node.js 애플리케이션과 연동하여서비스를 사용하고 애플리케이션을 배포하는 방법에 대해 제시하는 문서이다.

* + 1. 범위

본 문서의 범위는 개방형 플랫폼 프로젝트의 Node.js 애플리케이션 개발과 서비스팩 연동에 대한 내용으로 한정되어 있다.

* + 1. 참고 자료
* <https://docs.cloudfoundry.org/devguide/>
* <https://docs.cloudfoundry.org/buildpacks/node/node-tips.html>
* <https://nodejs.org/>
* <http://expressjs.com/ko/>
* <https://github.com/felixge/node-mysql>
* <https://github.com/CUBRID/node-cubrid>
* <https://github.com/mongodb/node-mongodb-native>
* <https://github.com/NodeRedis/node_redis>
* <https://github.com/postwait/node-amqp>
* <https://github.com/pkgcloud/pkgcloud>
* <https://mochajs.org/>

1. Node.js 애플리케이션개발가이드
   1. 개요

Open PaaS에 등록된 다양한 서비스팩을 Node.js언어로 작성된 애플리케이션과 바인딩하고해당 애플리케이션에 바인딩된 환경정보(VCAP\_SERVICES)에서 각 서비스별 접속정보를 획득하여 애플리케이션에 적용하여 이용 할 수 있도록 Windows 환경에서 Node.js 애플리케이션을 작성 할 수 있도록한다.

* 1. 개발환경 구성

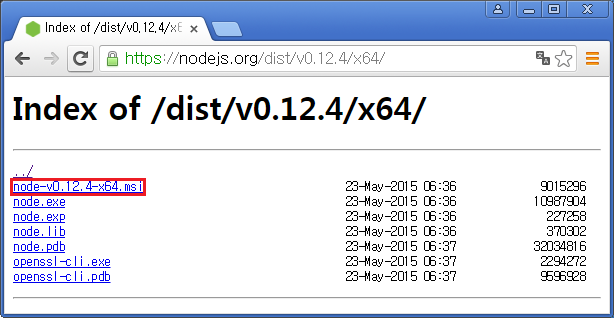
Node.js 애플리케이션 개발을 위해 다음과 같은 환경으로 개발환경을 구성 한다.

* BuildPack: v1.3.4
* OS : Windows 7 64bit
* Node.js : v0.12.4
* npm : v2.10.1
  + 1. Node.js 및 npm 설치

1. Node.js 다운로드

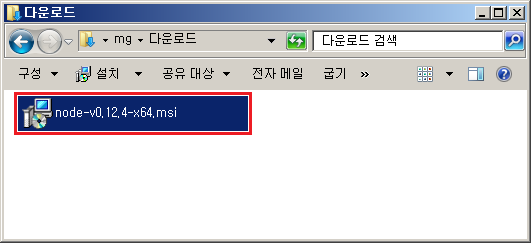
* 아래의 주소로 접속한 후 node-v0.12.4-x64.msi를 다운받는다.

|  |
| --- |
| https://nodejs.org/dist/v0.12.4/x64/node-v0.12.4-x64.msi |

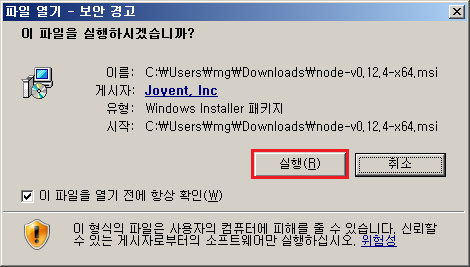


1. Node.js 설치

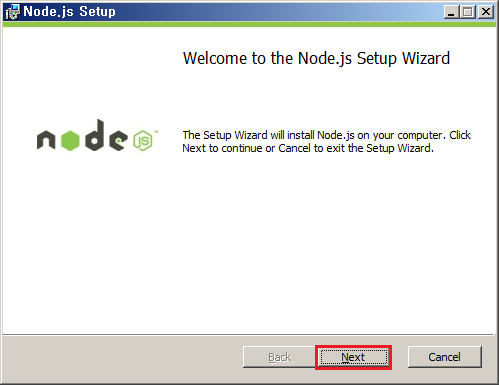
* 다운받은 폴더에서 node-v0.12.4-x64.msi를 더블클릭하여 설치를 시작한다.



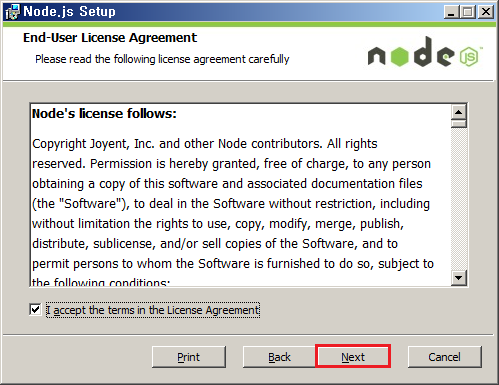
* "실행"버튼을 클릭하여 계속 진행한다.



* "Next"버튼을 클릭하여 계속 진행한다.

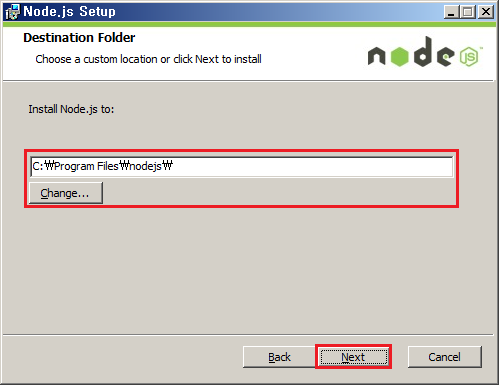


* "I accept the terms in the License Agreement"를 체크하여 라이센스에 동의한 후 "Next"버튼을 클릭하여 계속 진행한다.



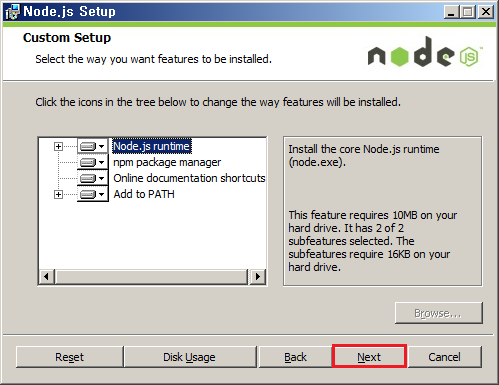
* 설치경로를 입력 혹은 선택한 후 "Next"버튼을 클릭하여 계속 진행한다.

여기서는 C:\Program Files\nodejs\ 를 설치경로로 설정하였다.

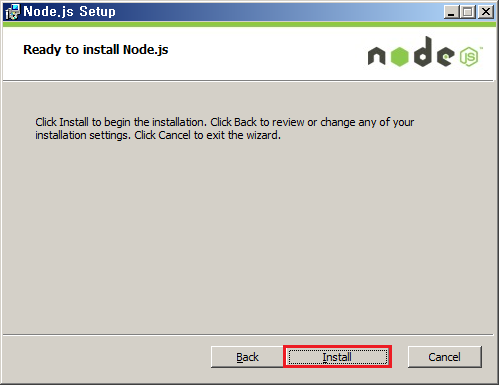


* 설치할 항목을 선택한 후 "Next"버튼을 클릭하여 계속 진행한다.

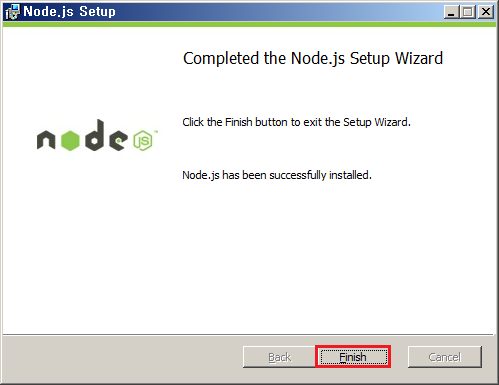
여기서는 선택하여 Node.js, npm, doc을 설치하고 환경변수 PATH까지 추가하였다.



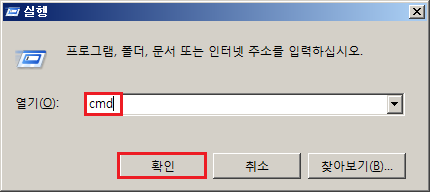
* "Install"버튼을 클릭하여 설치한다.



* "Finish"버튼을 클릭하여 설치를 완료한다.

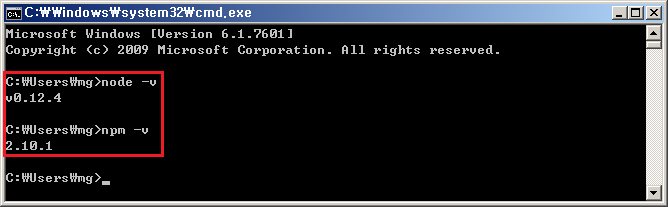


* '윈도우키+R' 또는 '시작->실행'아이콘을 클릭하여 실행창을 띄운 후 'cmd'를 입력하고 "확인"버튼을 눌러 커맨드창을 연다.



* 커맨드창에 아래의 명령어를 입력하여 node.js와 npm의 버젼과 제대로 설치되었는지 여부를 확인한다.

|  |
| --- |
| >node -v  >npm -v |



개발도구

Node.js는 javascript기반의 언어로 Notepad++, Sublim Text, EditPlus등 문서편집기를 개발도구로 사용할 수 있다. 또한 Eclipse의 플러그인 Nodeclipse를 설치하여 사용할 수도있다.

* 1. 개발

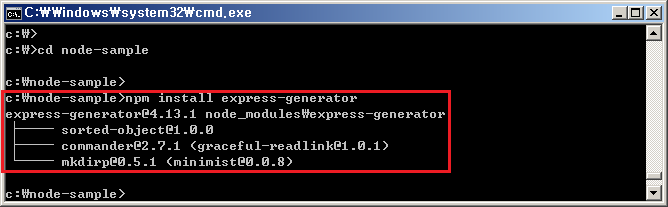
샘플 애플리케이션에의 데이터 관리는 MySQL, CubridDB, MongoDB 중에 하나를 이용하기 때문에 API 요청시 요청 본문에 DBType 값을 가지고 결정한다.

* + 1. Node.js Express애플리케이션 생성

1. 'express-generator'를 이용하여 Express 애플리케이션을 생성

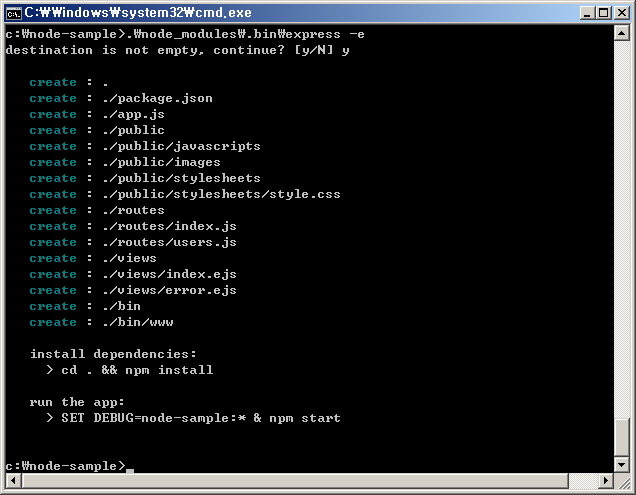
* 커맨드 창에서 개발을 진행할 경로로 이동후 아래의 명령어를 입력하여 'express-generator' npm을 설치한다.

|  |
| --- |
| >npm install express-generator |



* Express 애플리케이션을 생성한다. '-e'옵션은 view enjine을 ejs를 사용한다는 것이고 default view enjin은 jade이다.

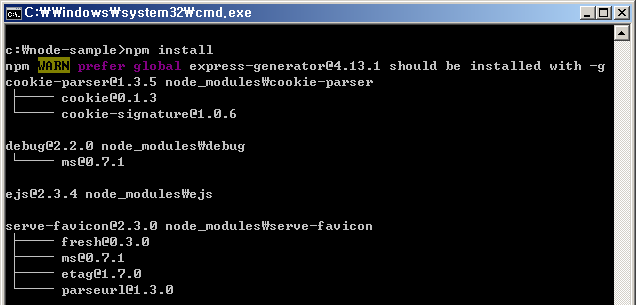
|  |
| --- |
| >.\node\_modules\.bin\express -e |



1. npm 설치

* Express 애플리케이션에 기본적으로 포함되어있는 npm을 설치한다. 설치할 npm에 대한 정의는 package.json에 정의되어있다.

|  |
| --- |
| >npm install |



1. Node.js Express 어플리캐이션 실행

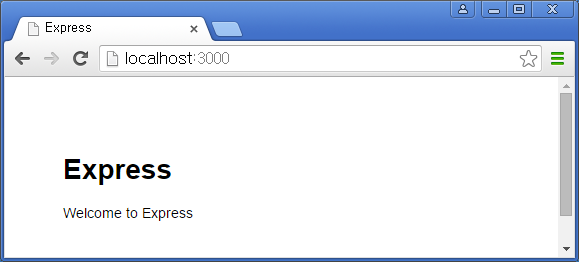
* 아래의 두 명령어중 하나를 이용해 애플리케이션 실행한다.

|  |
| --- |
| >npm start |
| >node bin/www |



* 브라우저로 아래의 주소로 접속하여 애플리케이션이 제대로 동작하는지 확인한다.

|  |
| --- |
| http://localhost:3000/ |



* + 1. Node.js 샘플 애플리케이션

1. Node.js 샘플 애플리케이션 다운로드 (현재는 비공개이며 지원예정)

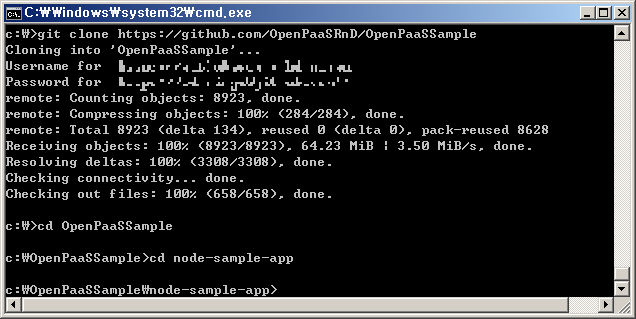
* git clone을 통해 Node.js 샘플 애플리케이션을 다운받는다.

|  |
| --- |
| git clone https://github.com/OpenPaaSRnD/OpenPaaSSample |

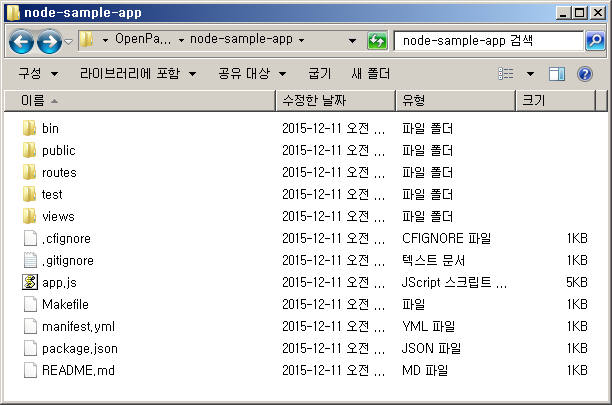
1. Node.js 샘플 애플리케이션 경로로 이동

* 다운받은 경로아래에 Node.js 샘플 애플리케이션 경로로 이동한다.

|  |
| --- |
| >cd node-sample-app |



1. Node.js 샘플 애플리케이션 디렉토리구조



|  |  |
| --- | --- |
| **파일/폴더** | **목적** |
| package.json | node.js 어플리케이션에 필요한 npm의 의존성 정보를 기술하는데 사용 한다.  npm install 명령을 실행시 install 뒤에 아무런 정보를 입력하지 않으면 이 파일의 정보를 이용하여 npm을 설치한다. |
| app.js | Express에 대한 설정을 한다. http요청에 대한 라우팅 정보 또한 이곳에서 정의한다. |
| bin/www | 실질적으로 node.js 샘플 애플리케이션의 시작점이며, http서버를 설정한다. 서버가 구동될 때 사용할 Port를 여기서 설정할 수 있다. |
| routes/ | app.js에서 라우팅 후 실제 수행할 내용이 작성되어있다. 서비스 연결에 대한 내용도 이곳에 있다. |
| public/ | 외부에서 접근가능한 디렉토리이다. css, js등 웹서비스에 필요한 정적파일들이 있다.  외부에서 접근가능한 디렉토리에 대한 설정은 app.js파일에서 설정한다. |
| views/ | ejs파일들이 위치하는 곳이다.  ejs파일은 html을 좀더 쉽게 작성할 수 있게 도와주는 template enjin 이며 express의 view enjin을 ejs로 설정했을시 사용하는 파일이다.  render() 메소드를 이용하여 ejs파일을 html로 변환하여 보여준다. |
| test/ | mocha test를 작성한 디렉토리이다. |
| (node\_modules) | 위의 그림에서는 보이지 않지만 npm install로 모듈설치시 이 디렉토리아래에 설치가 된다. 여기에 설치되어있는 npm모듈들을 애플리케이션에서 require로 불러와서 사용할 수 있다. |
| Makefile | linux에서 좀더 쉽게 mocha 테스트를 실행하기위한 파일이다. |
| manifest.yml | 개방형 플랫폼에 배포시 애플리케이션에 대한 설정이다. 애플리케이션의 이름, 배보될 경로, 인스턴스 수 등을 정의할 수 있다. |
| .cfignore | 개방형 플랫폼에 배포시 포함되지않을 디렉토리, 혹은 파일을 기술한다. |
| .gitignore | git에 배포시 포함되지않을 디렉토리, 혹은 파일을 기술한다. |
| README.md | Node.js 샘플 애플리케이션에 대한 간략한 설명이 기술되어 있다. |

* + 1. 애플리케이션 환경설정

이 샘플은 Node.js version 0.12.4, npm version 2.10.1.을 기준으로 각 모듈의 버전을 명시적으로 선택하여 설치하였다.

package.json 수정(설정)시 설치된 Node.js의 버전에 맞는 모듈을 설치하는 것을 권장한다.

1. ./package.json

* 애플리케이션에서 필요한 모듈을 정의한다.

|  |
| --- |
| {  "name": "node-sample-app",  "version": "0.8.0",  "private": true,  "scripts": {  "start": "node ./bin/www"  },  "dependencies": {  "body-parser": "1.13.2",  "cookie-parser": "1.3.5",  "debug": "2.2.0",  "morgan": "1.6.1",  "serve-favicon": "2.3.0",  "express": "4.13.1",  "ejs": "2.3.4",  "generic-pool": "2.2.1",  "mysql": "2.9.0",  "node-cubrid":"2.2.5",  "mongodb":"2.0.48",  "redis":"2.4.2",  "uuid":"2.0.1",  "amqp":"0.2.4",  "pkgcloud":"1.2.0-alpha.0",  "formidable":"1.0.17",  "mocha":"2.3.4",  "should":"7.1.1",  "supertest":"1.1.0"  },  "engines": {  "node": "0.12.4",  "npm": "2.10.1"  }  } |

|  |  |
| --- | --- |
| name | 애플리케이션 이름 |
| version | 애플리케이션 버젼 |
| private | npm에 게시할것인지 여부를 설정한다. (true: 게시하지않음) |
| scripts.start | npm start 명령어로 실행될 명령어(애플리케이션 구동 명령어) |

* dependencies

|  |  |
| --- | --- |
| body-parser | Express프레임워크에서 기본적으로 사용하는 모듈들. |
| cookie-parser |
| debug |
| morgan |
| serve-favicon |
| express |
| ejs |
| generic-pool | connection pool생성 및 관리 모듈 |
| mysql | mysql 모듈 |
| node-cubrid | cubrid 모듈 |
| mongodb | mongodb 모듈 |
| redis | redis 모듈 |
| uuid | 고유식별자를 생성해주는 모듈 |
| amqp | rabbitMQ 사용 모듈 |
| pkgcloud | swift, glusterfs 사용 모듈 |
| formidable | form data를 파싱해주는 모듈 |
| mocha | node.js test 모듈 |
| should | mocha test에 사용되는 모듈 |
| supertest | rest test에 사용되는 모듈 |

* engines

|  |  |
| --- | --- |
| node | 애플리케이션에서 사용할 node.js 모듈 버젼.  개방형 플랫폼에 배포하여 사용시 Node Buildpack에서 지원하는 Node.js 버젼에 따라 사용할 수 있는 버젼에 제약이 있다.   * [https://github.com/cloudfoundry/nodejs-buildpack/blob/master/CHANGELOG](%20https:/github.com/cloudfoundry/nodejs-buildpack/blob/master/CHANGELOG) |
| npm | 애플리케이션에서 사용할 npm 버젼  Node.js와 마찬가지로 Node Buildpack에서 지원하는 버젼에 따라 사용할 수 있는 버젼에 제약이 있다. |

1. 모듈 설치

* pakage.json에 정의된 모듈을 설치한다. 모듈이름을 지정하지 않으면 package.json의 depencencies의 모든 모듈을 설치한다.

|  |
| --- |
| >npm install |

1. ./bin/www

* HTTP서버가 사용할 PORT를 개방형 플랫폼이 제공하는 PORT를 사용하게 설정한다. 개방형 플랫폼은 이 값을 이용하여 애플리케이션이 제대로 동작하고 있는지 감지하는데 사용한다. 이 값 외의 다른 PORT를 사용하면 애플리케이션이 제대로 동작하지 않는다.

|  |
| --- |
| #!/usr/bin/env node  /\*\*  \* Module dependencies.  \*/  var app = require('../app');  var debug = require('debug')('node-sample-app:server');  var http = require('http');  /\*\*  \* Get port from environment and store in Express.  \* port 환경변수를 얻어와서 변수에 담는다.  \* 'process.env.PORT'는 Cloud에서 사용하는 환경변수.  \*/  var port = normalizePort(process.env.PORT || '3000');  app.set('port', port);  /\*\*  \* Create HTTP server.  \* HTTP 서버 생성.  \*/  var server = http.createServer(app).listen(app.get('port'), function(){  console.log('Express server listening on port ' + app.get('port'));  });  /\*\*  \* Listen on provided port, on all network interfaces.  \* 서버가 사용할 port를 설정한다.  \*/  server.listen(port);  server.on('error', onError);  server.on('listening', onListening);  ...(생략) |

1. ./app.js

* Request URL 매핑 설정

|  |
| --- |
| var express = require('express');  var path = require('path');  var favicon = require('serve-favicon');  var logger = require('morgan');  var cookieParser = require('cookie-parser');  var bodyParser = require('body-parser');  var routes = require('./routes/index');  var users = require('./routes/users');  var app = express();  **// URL 매핑 후 수행할 js 파일들**  var org\_chart\_mysql = require('./routes/rest/org\_chart/mysql')  , org\_chart\_mongo = require('./routes/rest/org\_chart/mongo')  , org\_chart\_cubrid = require('./routes/rest/org\_chart/cubrid')  , orgs\_mysql = require('./routes/rest/orgs/mysql')  , orgs\_mongo = require('./routes/rest/orgs/mongo')  , orgs\_cubrid = require('./routes/rest/orgs/cubrid')  , groups\_mysql = require('./routes/rest/groups/mysql')  , groups\_mongo = require('./routes/rest/groups/mongo')  , groups\_cubrid = require('./routes/rest/groups/cubrid')  , login = require('./routes/rest/login/login')  , image = require('./routes/rest/image/image')  , page = require('./routes/page/page\_processing');  // view engine setup  // 뷰 엔진 설정  app.set('views', path.join(\_\_dirname, 'views/'));  app.set('view engine', 'ejs');  // uncomment after placing your favicon in /public  //app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));  app.use(logger('dev'));  app.use(bodyParser.json());  app.use(bodyParser.urlencoded({ extended: false }));  app.use(cookieParser());  // 정적파일 위치 설정  app.use(express.static(path.join(\_\_dirname, 'public')));  **/\***  **\* URL 매핑 설정**  **\*/**  app.use( '/', routes);  app.use( '/users', users);  // page  app.get( '/login', page.login);  app.get( '/manage', page.manage);  app.get( '/main/:id', page.main);  // org-chart  app.get( '/org-chart/:org\_id/mysql', org\_chart\_mysql.index);  app.get( '/org-chart/:org\_id/cubrid', org\_chart\_cubrid.index);  app.get( '/org-chart/:org\_id/mongo', org\_chart\_mongo.index);  app.get( '/org-chart/:org\_id/status/mysql', org\_chart\_mysql.status);  app.get( '/org-chart/:org\_id/status/cubrid', org\_chart\_cubrid.status);  app.get( '/org-chart/:org\_id/status/mongo', org\_chart\_mongo.status);  //orgs  // mysql  app.get( '/orgs/mysql', orgs\_mysql.index);  app.post( '/orgs/mysql', orgs\_mysql.create);  app.get( '/orgs/:org\_id/mysql', orgs\_mysql.show);  app.put( '/orgs/:org\_id/mysql', orgs\_mysql.update);  app.delete('/orgs/:org\_id/mysql', orgs\_mysql.destroy);  …..(생략) |

* + 1. VCAP\_SERVICES 환경설정 정보

개방형 플랫폼에 배포되는 애플리케이션이 바인딩된서비스별 접속 정보를 얻기 위해서는 애플리케이션별로 등록되어있는 VCAP\_SERVICES 환경설정 정보를 읽어들여정보를 획득 할 수 있다.

1. 개방형 플랫폼의 애플리케이션 환경정보

* 서비스를 바인딩하면 JSON 형태로 환경설정 정보가 애플리케이션 별로 등록된다.

|  |
| --- |
| {  "VCAP\_SERVICES": {  "CubridDB": [  {  "credentials": {  "host": "10.30.60.23::",  "hostname": "10.30.60.23",  "jdbcurl": "jdbc:cubrid:10.30.60.23::fccf1d7869ff72ce:b2f6b4af1e7bd7d8:45f179c648ee60a5:",  "name": "fccf1d7869ff72ce",  "password": "45f179c648ee60a5",  "port": "",  "uri": "cubrid:10.30.60.23::fccf1d7869ff72ce:b2f6b4af1e7bd7d8:45f179c648ee60a5:",  "username": "b2f6b4af1e7bd7d8"  },  "label": "CubridDB",  "name": "sample-cubrid-instance",  "plan": "utf8",  "tags": [  "cubrid",  "document"  ]  }  ],  …..(이하 생략)….. |

1. Node.js에서 개방형 플랫폼의 애플리케이션 환경정보에 접근하는 방법

* 시스템환경변수의 VCAP\_SERVICES값을 읽어서 접근 할 수 있다.

|  |
| --- |
| process.env.VCAP\_SERVICES |

* + 1. Mysql 연동

1. ./route/db/mysql/db\_pooling.js

* 개방형 플랫폼의 애플리케이션 환경정보에 접근하여 mysql Connection Pool을 생성

|  |
| --- |
| /\*\*  \* generic-pool 연동  \* mysql 풀 모듈 구현  \*/  var generic\_pool = require("generic-pool");  var mysql = require("mysql");  config = {};  if (process.env.VCAP\_SERVICES) {  **// cloud env 설정. 데이터 구조는 2.3.4 VCAP\_SERVICES 환경정보 참고**  var cloud\_env = JSON.parse(process.env.VCAP\_SERVICES);  var mysql\_env = cloud\_env["Mysql-DB"][0]["credentials"];  config = {  host:mysql\_env.hostname,  user:mysql\_env.username,  password:mysql\_env.password,  database:mysql\_env.name  };  } else {  // local env  config = {  host:'10.30.40.63',  user:'cESTBl9QpxGVF5Xa',  password:'aVu1ynInBnaEeFY0',  database:'cf\_ea68784e\_3de6\_439d\_afc1\_d51b4e95627b'  };  }  var pooling = generic\_pool.Pool({  name:"mysql",  create:function(cb){  **// create Connection**  var conn = mysql.createConnection(config);  conn.connect(function(err){  if( err) console.log("mysql 연결오류");  else {  // console.log("mysql 연결성공");  } cb(err, conn);  // 콜백함수를 통해 풀링에 커넥션 객체를 던짐  });  },  destroy:function(myConn){  myConn.end(function(err){  if( err) console.log("mysql 연결해제오류");  // else console.log("mysql 연결해제성공");  });  },  min:3,  max:5,  idleTimeoutMillis:1000\*500,  log:false  });  process.on("exit", function(){  pooling.drain(function(){  pooling.destroyAllNow();  });  });  module.exports = pooling; |

* + 1. Cubrid 연동

1. ./route/db/cubrid/db\_pooling.js

* 개방형 플랫폼의 애플리케이션 환경정보에 접근하여 cubrid Connection Pool을 생성

|  |
| --- |
| /\*\*  \* generic-pool 연동  \* cubrid 풀 모듈 구현  \*/  var generic\_pool = require("generic-pool");  var cubrid = require("node-cubrid");  var database  , port  , hostname  , username  , password;  if (process.env.VCAP\_SERVICES) {  **// cloud env 설정. 데이터 구조는 2.3.4 VCAP\_SERVICES 환경정보 참고**  var cloud\_env = JSON.parse(process.env.VCAP\_SERVICES);  var cubrid\_env = cloud\_env["CubridDB"][0]["credentials"];  database = cubrid\_env.name  port = cubrid\_env.port  hostname = cubrid\_env.hostname  username = cubrid\_env.username  password = cubrid\_env.password;  } else {  // local env  database = 'fccf1d7869ff72ce'  port = ''  hostname = '10.30.60.23'  username = 'b2f6b4af1e7bd7d8'  password = '45f179c648ee60a5';  }  var pooling = generic\_pool.Pool({  name:"cubrid",  create:function(cb){  // console.log("cubrid\_env.uri:" + cubrid\_env.uri);  var conn = cubrid.createCUBRIDConnection(hostname, port, username, password, database);  **// create Connection**  conn.connect(function(err){  if( err) console.log("cubrid 연결오류");  else{  // console.log("cubrid 연결성공");  cb(err, conn);  }  // 콜백함수를 통해 풀링에 커넥션 객체를 던짐  });  },  destroy:function(myConn){  myConn.end(function(err){  if( err) console.log("cubrid 연결해제오류");  // else console.log("cubrid 연결해제성공");  });  },  min:3,  max:5,  idleTimeoutMillis:1000\*50,  log:false,    });  process.on("exit", function(){  pooling.drain(function(){  pooling.destroyAllNow();  });  });  module.exports = pooling; |

* + 1. MongoDB 연동

1. ./route/db/mongo/db\_pooling.js

* 개방형 플랫폼의 애플리케이션 환경정보에 접근하여 mongodb Connection Pool을 생성

|  |
| --- |
| /\*\*  \* generic-pool 연동  \* mongo 풀 모듈 구현  \*/  var generic\_pool = require("generic-pool");  var mongoClient = require("mongodb").MongoClient;  var url = '';  if (process.env.VCAP\_SERVICES) {  **// cloud env. 설정. 데이터 구조는 2.3.4 VCAP\_SERVICES 환경정보 참고**  var cloud\_env = JSON.parse(process.env.VCAP\_SERVICES);  var mongo\_env = cloud\_env["Mongo-DB"][0]["credentials"];  url = mongo\_env.uri;  } else {  // local env.  url = 'mongodb://d3e35ad5-9f49-43ae-bc85-08e39ec1d8eb:fc23791e-b2d8-402d-b070-90bfbdb5dcfa@10.30.60.53:27017/e37e541c-75de-4f01-8196-63e2d902e768';  }  var pooling = generic\_pool.Pool({  name:"mongo",  create:function(cb){  **// create Connection**  mongoClient.connect(url, function(err, db){  if (err) console.log("mongo 연결오류");  else {  cb(err, db);  }  });  },  destroy:function(myDb){  myDb.close(function(err){  if( err) console.log("mysql 연결해제오류");  // else console.log("mysql 연결해제성공");  });  },  min:3,  max:5,  idleTimeoutMillis:1000\*500,  log:false  });  module.exports = pooling; |

* + 1. Redis 연동

1. ./route/redis/redis.js

* 개방형 플랫폼의 애플리케이션 환경정보에 접근하여 redis Connection을 생성

|  |
| --- |
| var redis = require("redis");  var options = {};  if (process.env.VCAP\_SERVICES) {  **// cloud env. 설정. 데이터 구조는 2.3.4 VCAP\_SERVICES 환경정보 참고**  var services = JSON.parse(process.env.VCAP\_SERVICES);  var redisConfig = services["redis-sb"];  if (redisConfig) {  var node = redisConfig[0];  options = {  host: node.credentials.host,  port: node.credentials.port,  pass: node.credentials.password,  };  }  } else {  // local env.  options = {  host: '10.30.40.71',  port: '34838',  pass: 'c239b721-d986-4ee3-8816-b5f5fa9f3ffb',  };  }  var client = null;  exports.open = function(cb) {  // console.log(JSON.stringify(options));  **//create Client**  client = redis.createClient(options);  // get auth.  client.auth(options.pass);  cb(client);  }  exports.close = function(){  client.end();  } |

* + 1. RabbitMQ연동

1. ./route/rabbitMQ/rabbitMQ.js

* 개방형 플랫폼의 애플리케이션 환경정보에 접근하여 rabbirMQ Connection을 생성

|  |
| --- |
| var amqp = require('amqp');  var url = '';  if (process.env.VCAP\_SERVICES) {  **// cloud env. 설정. 데이터 구조는 2.3.4 VCAP\_SERVICES 환경정보 참고**  var services = JSON.parse(process.env.VCAP\_SERVICES);  var rabbitMQConfig = services["p-rabbitmq"];  if (rabbitMQConfig) {  var node = rabbitMQConfig[0];  url = node.credentials.uri;  }  } else {  // local env.  url = 'amqps://14b1ab93-4cdb-46af-8cdd-8d8073bbe282:cl71e9ihgu6gvhj1eiqj9uh4um@10.30.40.82:5671/6ffb4d8a-8748-4f00-a338-80e6eadee822';  }  exports.open = function(cb){  **// create connection.**  var conn = amqp.createConnection({url: url});    // it must be cb(callback) after the 'ready' event.  conn.on('ready', function(){  cb(conn);  });  }  // not used.  /\*  exports.close = function(){  conn.disconnect();  }  \*/ |

* + 1. GlusterFS 연동

1. ./route/glusterfs/glusterfs.js

* 개방형 플랫폼의 애플리케이션 환경정보에 접근하여 glusterfs Connection을 생성

|  |
| --- |
| var pkgcloud = require('pkgcloud');  var http = require('http');  var url = require('url');  var credentials = {};  var container\_name = 'node\_container';  if (process.env.VCAP\_SERVICES) {  **// cloud env.** **설정. 데이터 구조는 2.3.4 VCAP\_SERVICES 환경정보 참고**  var services = JSON.parse(process.env.VCAP\_SERVICES);  var glusterfsConfig = services["glusterfs"];    if (glusterfsConfig) {  var config = glusterfsConfig[0];  credentials = {  provider: 'openstack', //  username: config.credentials.username,  password: config.credentials.password,  authUrl: config.credentials.auth\_url.substring(0, config.credentials.auth\_url.lastIndexOf('/')),  region: 'RegionOne' //  };  }  } else {  **// local env.**  credentials = {  provider: 'openstack',  username: 'cf13d551d997458e',  password: 'b45cc01d53a4f0e0',  authUrl: 'http://54.199.136.22:5000/',  region: 'RegionOne'  };  }  **// create Client**  var client = pkgcloud.storage.createClient(credentials);  // delete container for test  /\*  client.destroyContainer(container\_name, function(err, result){  if (err) console.log(err);  else console.log(result);  });  \*/  // check container  client.getContainer(container\_name, function(err, container){  if (err)  {  // if container not exist  if (err.statusCode === 404)  {  // create container  client.createContainer({name:container\_name}, function(create\_err, create\_container){  if (create\_err) console.log(err);  else  {  // if container created successfully, setting a readable member(X-Contaner-Read: .r:\*)  // 컨테이너가 성공적으로 생성되었다면 컨테이너를 누구나 읽을 수 있게 설정한다.(X-Contaner-Read: .r:\*)  // There is a bug in the code(pkgcloud). so i used api call.  // pkgcloud 모듈에서 metadata를 넣을 경우 prefix가 붙는 로직때문에 제대로 위의 값이 입력이 안되므로 api를 통해서 설정.  var serviceUrl = url.parse(create\_container.client.\_serviceUrl);  var option = {  host: serviceUrl.hostname,  port: serviceUrl.port,  path: serviceUrl.path+'/'+container\_name,  method: 'POST',  headers: {  'X-Auth-Token': create\_container.client.\_identity.token.id,  'X-Container-Read': '.r:\*' // ACL form  }  };  var req = http.request(option, function(res){  });  req.end();  }  });  }  else console.log(err);  }  });  module.exports = client; |

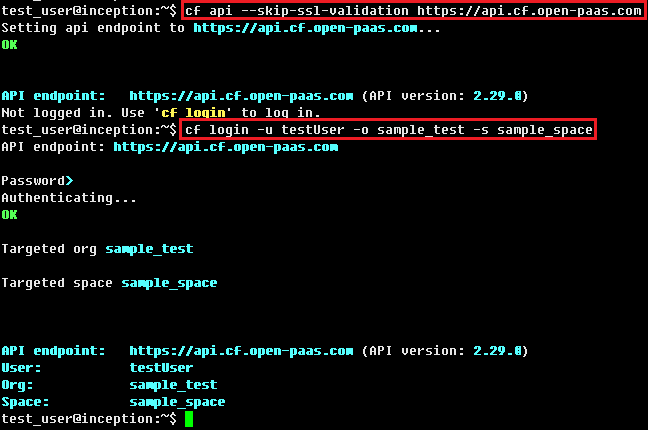
* 1. 배포

개방형 플랫폼에 애플리케이션을 배포하면 배포한 애플리케이션과 개방형 플랫폼이 제공하는 서비스를 연결하여 사용할 수 있다. 개방형 플랫폼상에서 실행을 해야만 개방형 플랫폼의 애플리케이션 환경변수에 접근하여 서비스에 접속할 수 있다.

* + 1. 개방형 플랫폼 로그인

아래의 과정을 수행하기 위해서 개방형 플랫폼에 로그인

|  |
| --- |
| $ cf api --skip-ssl-validation https://api.cf.open-paas.com **# 개방형 플랫폼 TARGET 지정**  **#cf api [target url]**  $ cf login -u testUser -o sample\_test -s sample\_space **# 로그인 요청**  **#cf login –u [user name] –o [org name] –s [space name]** |

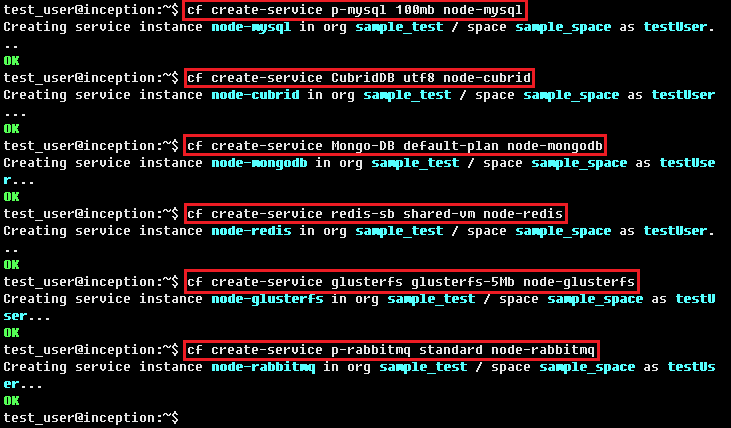


* + 1. 서비스 생성

애플리케이션에서 사용할 서비스를 개방형 플랫폼을 통하여 생성한다. 별도의 서비스 설치과정 없이 생성할 수 있으며, 애플리케이션과 바인딩과정을 통해 접속정보를 얻을 수있다.

* 서비스 생성 (cf marketplace 명령을 통해 서비스 목록과 각 서비스의 플랜을 조회할 수 있다.)

|  |
| --- |
| **# cf create-service SERVICE PLAN SERVICE\_INSTANCE [-c PARAMETERS\_AS\_JSON] [-t TAGS]**  $ cf create-service p-mysql 100mb node-mysql  $ cf create-service CubridDB utf8 node-cubrid  $ cf create-service Mongo-DB default-plan node-mongodb  $ cf create-service redis-sb shared-vm node-redis  $ cf create-service glusterfs glusterfs-5Mb node-glusterfs  $ cf create-service p-rabbitmq standard node-rabbitmq |



* + 1. 애플리케이션 배포

애플리케이션을 개방형 플랫폼에 배포한다. 배포된 애플리케이션은 생성된 서비스와 바인드하여 서비스를 사용할 수 있다.

1. manifest.yml 생성

* cf push 명령시 현재 디렉토리의 manifest.yml을 참조하여 배포가 진행된다.

|  |
| --- |
| ---  applications:  - name: node-sample-app **# 애플리케이션 이름**  memory: 512M **# 애플리케이션 메모리 사이즈**  instances: 1 **# 애플리케이션 인스턴스 개수**  command: npm start **# 애플리케이션 실행 명령어**  path: ./ **# 배포될 애플리케이션의 위치** |

1. Mysql, Cubrid 테이블 생성

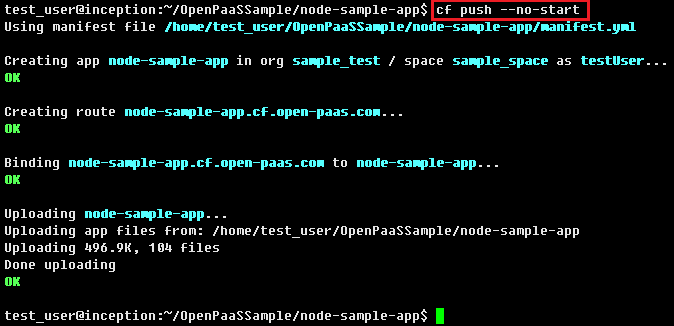
* Sample App의 조직관리 기능을 위해 DB에 테이블을 생성해 주어야 한다.
* Mysql과 Cubrid에 테이블을 추가하는 방법은 OpenPaaS Mysql, Cubrid 서비스팩 설치 가이드의 'Client 툴 접속'을 참고한다.
* Client 툴을 이용하여 아래의 테이블 생성 sql를 각각 실행한다. (Mysql과 Cubrid 양쪽다 동일한 sql로 생성가능하다.)

|  |
| --- |
| DROP TABLE IF EXISTS ORG\_TBL;  DROP TABLE IF EXISTS GROUP\_TBL;  CREATE TABLE ORG\_TBL (  id INT AUTO\_INCREMENT PRIMARY KEY  , label VARCHAR(40) NOT NULL  , `desc` VARCHAR(150)  , url VARCHAR(500) DEFAULT '#'  , created TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL  , modified TIMESTAMP  );  CREATE TABLE GROUP\_TBL (  id INT AUTO\_INCREMENT PRIMARY KEY  , org\_id INTEGER NOT NULL  , parent\_id INTEGER  , label VARCHAR(40) NOT NULL  , `desc` VARCHAR(150)  , thumb\_img\_name VARCHAR(256)  , thumb\_img\_path VARCHAR(512)  , url VARCHAR(500) DEFAULT '#'  , created TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL  , modified TIMESTAMP  );  ALTER TABLE GROUP\_TBL  ADD FOREIGN KEY(org\_id)  REFERENCES ORG\_TBL(id)  ON DELETE CASCADE;  ALTER TABLE GROUP\_TBL  ADD FOREIGN KEY(parent\_id)  REFERENCES GROUP\_TBL(id)  ON DELETE CASCADE; |

1. 애플리케이션 배포

* cf push 명령으로 배포한다. 별도의 값을 넣지않으면 manifest.yml의 설정을 사용한다. 아직 서비스를 연결하지 않았기 때문에 --no-start 옵션으로 배포후 실행은 하지않는다.

|  |
| --- |
| $ cf push --no-start |

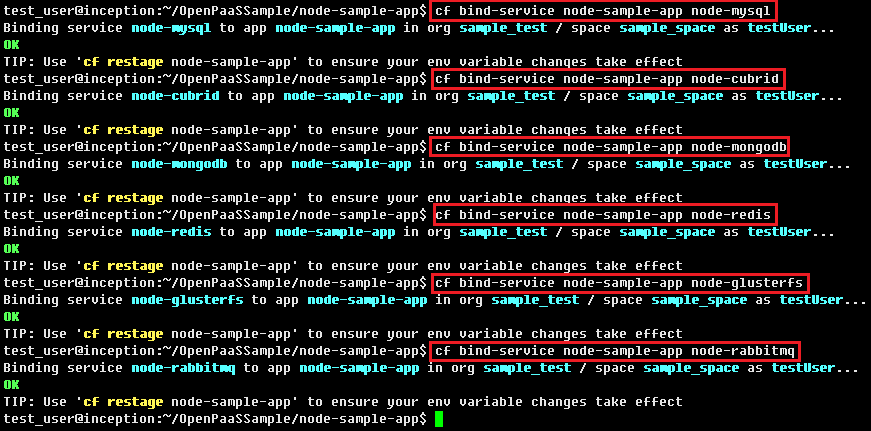


* + 1. 애플리케이션, 서비스 연결

애플리케이션과 서비스를 연결하는 과정을 '바인드(bind)라고 하며, 이 과정을 통해 서비스에 접근할 수 있는 접속정보를 생성한다.

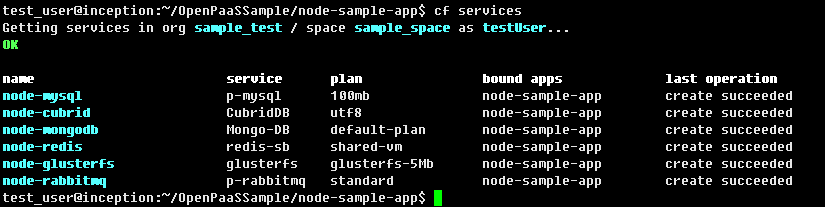
* 애플리케이션과 서비스 연결

|  |
| --- |
| **cf bind-service APP\_NAME SERVICE\_INSTANCE [-c PARAMETERS\_AS\_JSON]**  $ cf bind-service node-sample-app node-mysql  $ cf bind-service node-sample-app node-cubrid  $ cf bind-service node-sample-app node-mongodb  $ cf bind-service node-sample-app node-redis  $ cf bind-service node-sample-app node-glusterfs  $ cf bind-service node-sample-app node-rabbitmq |



연결확인

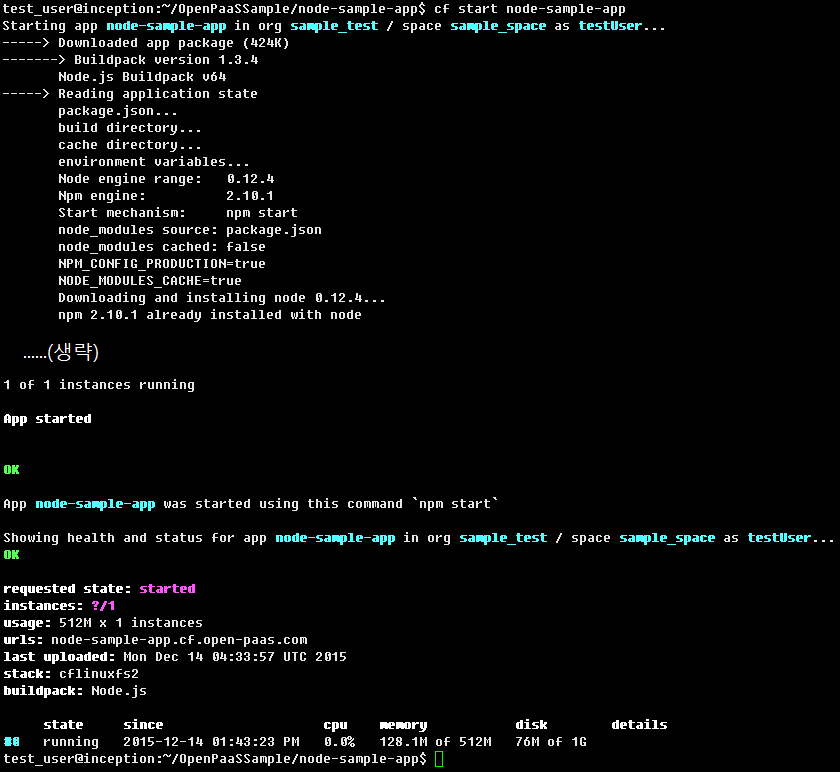
|  |
| --- |
| $ cf services |



* + 1. 애플리케이션 실행

서비스 바인드 과정을 통해 생성된 접속정보 환경변수를 가지고 어플리케이션이 실행된다.

|  |
| --- |
| $ cf start node-sample-app |



* 1. 테스트

샘플 어플리케이션은 REST 서비스로 구현되어있으며 REST 테스트를 위해서 mocha 모듈을 사용하였다. 테스트를 진행하기 위해서는 mocha 모듈을 포함한 package.json 안의 모듈들이 설치 되어 있어야한다. (npm install)

1. Makefile

* 매번 bin파일에 접근하여 실행하는 불편함을 해결하기 위해 작성. 리눅스 운영체제에서 사용할 수 있다.

|  |
| --- |
| test:  @./node\_modules/.bin/mocha -u tdd  .PHONY: test |

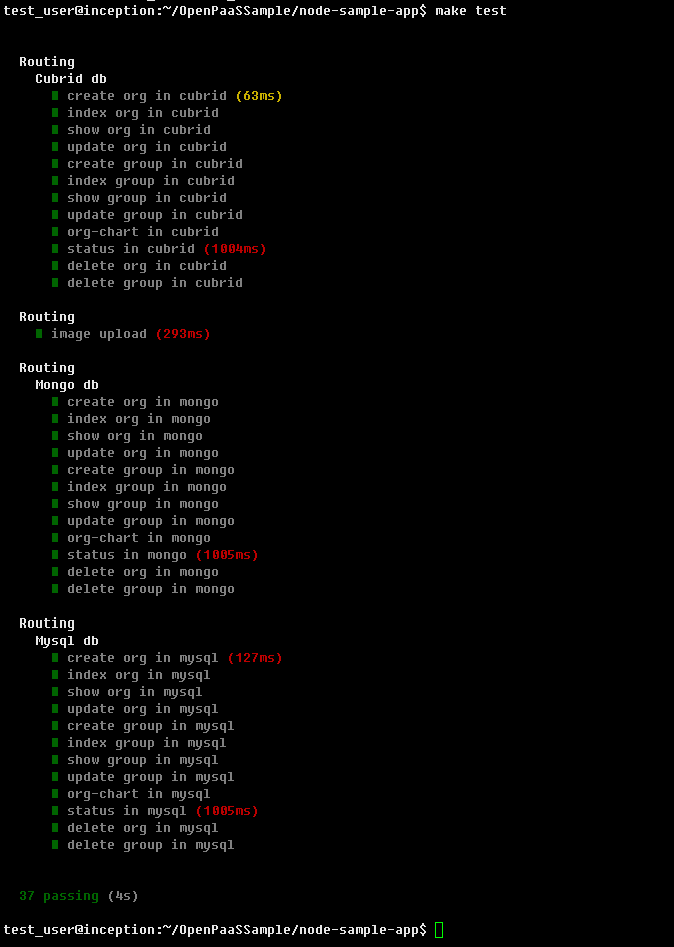
1. 테스트 실행

* test디렉토리 아래에 있는 테스트를 실행한다.
  1. 윈도우

|  |
| --- |
| > .\node\_modules\.bin\mocha -u tdd test |

* 1. 리눅스

|  |
| --- |
| $ make test |



1. 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.) [↑](#footnote-ref-2)