

전자정부표준프레임워크 기반의 Open PaaS 개발

Redis 서비스팩 설치 가이드

(Bosh-lite)

|  |  |  |
| --- | --- | --- |
| 작성자 | 송창학 (인) | 2015.8.18 |
| 검토자 | 안찬영 (인) | 2015.8. |
| 승인자 | 현재욱 (인) | 2015.8. |

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

개 정 이 력

| 버전 | 작성일 | 변경내용[[1]](#footnote-1) | 작성자 | 승인자 |
| --- | --- | --- | --- | --- |
| 1.0 | 2015.8.18 | 최초작성 | 송창학 | 현재욱 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목차

[1. 문서 개요 4](#_Toc433184763)

[*1.1.* *목적* 4](#_Toc433184764)

[*1.2.* *범위* 4](#_Toc433184765)

[*1.3.* *시스템 구성도* 4](#_Toc433184766)

[*1.4.* *참고자료* 5](#_Toc433184767)

[2. Redis 서비스팩 설치 6](#_Toc433184768)

[*2.1.* *설치전 준비사항* 6](#_Toc433184769)

[*2.2.* *Redis 서비스 릴리즈 업로드* 6](#_Toc433184770)

[*2.3.* *Redis 서비스 Deployment 파일 수정 및 배포* 10](#_Toc433184771)

[*2.4.* *Redis 서비스 브로커 등록* 21](#_Toc433184772)

[3. Redis 연동 Sample App 설명 24](#_Toc433184773)

[*3.1.* *Sample App 구조* 24](#_Toc433184774)

[*3.2.* *개방형 클라우드 플랫폼에서 서비스 신청* 24](#_Toc433184775)

[*3.3.* *Sample App에 서비스 바인드 신청 및 App 확인* 25](#_Toc433184776)

[4. Redis Client 툴 접속 29](#_Toc433184777)

[*4.1.* *Redis Desktop Manager 설치 & 연결* 29](#_Toc433184778)

# 문서 개요

## ***목적***

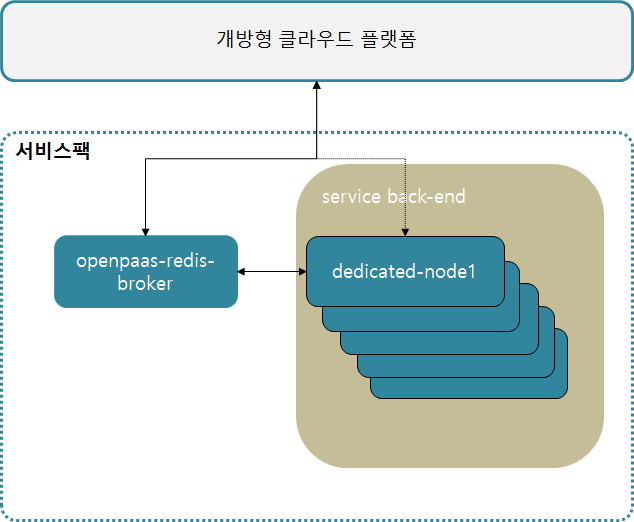
본 문서(Redis 서비스팩 설치 가이드)는 전자정부표준프레임워크 기반의 Open PaaS에서 제공되는 서비스팩인 Redis 서비스팩을 Bosh를 이용하여 설치 하는 방법과 Open PaaS의 SaaS 형태로 제공하는 Application 에서 Redis 서비스를 사용하는 방법을 기술하였다.

## ***범위***

설치 범위는 Redis 서비스팩을 검증하기 위한 기본 설치를 기준으로 작성하였다.

## ***시스템 구성도***

본 문서의 설치된 시스템 구성도입니다. Redis dedicated-node (5대), Redis 서비스 브로커로 최소사항을 구성하였다.



## ***참고자료***

<http://bosh.io/docs>

<http://docs.cloudfoundry.org/>

# Redis 서비스팩 설치

## ***설치전 준비사항***

본 설치 가이드는 Linux 환경에서 설치하는 것을 기준으로 하였다.

서비스팩 설치를 위해서는 먼저 BOSH-lite 가 설치 되어 있어야 하고 BOSH 에 로그인 및 타켓 설정이 되어 있어야 한다.

BOSH-lite 가 설치 되어 있지 않을 경우 먼저 BOSH-lite 설치 가이드 문서를 참고 하여 BOSH-lite를 설치 해야 한다.

OpenPaaS 에서 제공하는 압축된 릴리즈 파일들을 다운받는다. (OpenPaaS-Deployment.zip, OpenPaaS-Sample-Apps.zip, OpenPaaS-Services.zip)

## ***Redis 서비스 릴리즈 업로드***

* OpenPaaS-Services.zip 파일 압축을 풀고 폴더안에 있는 Redis 서비스 릴리즈 openpaas-redis-release-beta-1.0.tgz 파일을 복사한다.

업로드할 openpaas-redis-release-beta-1.0.tgz 파일을 확인한다.

|  |
| --- |
| $ ls –all |

* 업로드 되어 있는 릴리즈 목록을 확인한다.

|  |
| --- |
| $ bosh releases    Redis 서비스 릴리즈가 업로드 되어 있지 않은 것을 확인 |

* Redis 서비스 릴리즈 파일을 업로드한다.

|  |
| --- |
| $ bosh upload release {서비스 릴리즈 파일 PATH}  $ bosh upload release openpaas-redis-release-beta-1.0.tgz |

* 업로드 된 Redis 릴리즈를 확인한다.

|  |
| --- |
| $ bosh releases    Redis 서비스 릴리즈가 업로드 되어 있는 것을 확인 |

## ***Redis 서비스 Deployment 파일 수정 및 배포***

BOSH Deployment manifest 는 components 요소 및 배포의 속성을 정의한 YAML[[2]](#footnote-2) 파일이다.

Deployment manifest 에는 sotfware를 설치 하기 위해서 어떤 Stemcell[[3]](#footnote-3)(OS, BOSH agent) 을 사용할것이며 Release[[4]](#footnote-4)(Software packages, Config templates, Scripts) 이름과 버전, VMs 용량, Jobs params 등을 정의가 되어 있다.

* OpenPaaS-Deployment.zip 파일 압축을 풀고 폴더안에 있는 lite용 Redis Deployment 화일인 openpaas-redis-lite.yml 를 복사한다.

다운로드 받은 Deployment Yml 파일을 확인한다. (openpaas-redis-lite.yml)

|  |
| --- |
| $ ls –all |

* Director UUID를 확인한다.

BOSH CLI가 배포에 대한 모든 작업을 허용하기위한 현재 대상 BOSH Director의 UUID와 일치해야한다. ‘bosh status’ CLI 을 통해서 현재 BOSH Director 에 target 되어 있는 UUID를 확인 할 수 있다.

|  |
| --- |
| $ bosh status |

* Deploy시 사용할 Stemcell을 확인한다. (Stemcell 2776 버전 사용)

|  |
| --- |
| $ bosh stemcells    Stemcell 목록이 존재 하지 않을 경우 BOSH-lite 설치 가이드 문서를 참고 하셔서 Stemcell 2776 버전을 업로드를 해야 한다. |

* openpaas-redis-lite.yml Deployment 파일을 서버 환경에 맞게 수정한다. (빨간색으로 표시된 부분만 수정)

|  |
| --- |
| $ vi openpaas-redis-lite.yml  # openpaas-redis-lite 설정 파일 내용  ---  name: openpaas-redis-service # 서비스 배포이름(필수)  director\_uuid: xxxxx #bosh status 에서 확인한 Director UUID을 입력(필수)  releases:  - name: openpaas-redis #서비스 릴리즈 이름(필수)  version: beta-1.0 #서비스 릴리즈 버전(필수): latest 시 업로드된 서비스 릴리즈 최신버전  compilation: # 컴파일시 필요한 가상머신의 속성(필수)  cloud\_properties: # 컴파일 VM을 만드는 데 필요한 IaaS의 특정 속성 (instance\_type, availability\_zone), 직접 cpu,disk,ram 사이즈를 넣어도 됨  name: random  network: services # Networks block에서 선언한 network 이름(필수)  reuse\_compilation\_vms: true # 컴파일지 VM 재사용 여부(옵션)  workers: 2 # 컴파일 하는 가상머신의 최대수(필수)  update:  canaries: 1 # canary 인스턴스 수(필수)  canary\_watch\_time: 30000-180000 # canary 인스턴스가 수행하기 위한 대기 시간(필수)  update\_watch\_time: 30000-180000 # non-canary 인스턴스가 수행하기 위한 대기 시간(필수)  max\_in\_flight: 4 # non-canary 인스턴스가 병렬로 update 하는 최대 개수(필수)  networks: # 네트워크 블록에 나열된 각 서브 블록이 참조 할 수있는 작업이 네트워크 구성을 지정, bosh lite 에서는 제공하는 네트워크를 수정 없이 사용  - name: services  subnets:  - cloud\_properties:  name: random  range: 10.244.3.32/30  reserved:  - 10.244.3.33  - cloud\_properties:  name: random  range: 10.244.3.36/30  reserved:  - 10.244.3.37  - cloud\_properties:  name: random  range: 10.244.3.40/30  reserved:  - 10.244.3.41  - cloud\_properties:  name: random  range: 10.244.3.44/30  reserved:  - 10.244.3.45  static:  - 10.244.3.46  - cloud\_properties:  name: random  range: 10.244.3.48/30  reserved:  - 10.244.3.49  - cloud\_properties:  name: random  range: 10.244.3.52/30  reserved:  - 10.244.3.53  static:  - 10.244.3.54  - cloud\_properties:  name: random  range: 10.244.3.56/30  reserved:  - 10.244.3.57  static:  - 10.244.3.58  - cloud\_properties:  name: random  range: 10.244.3.60/30  reserved:  - 10.244.3.61  static:  - 10.244.3.62  - cloud\_properties:  name: random  range: 10.244.3.64/30  reserved:  - 10.244.3.65  static:  - 10.244.3.66  - cloud\_properties:  name: random  range: 10.244.3.68/30  reserved:  - 10.244.3.69  static:  - 10.244.3.70  resource\_pools: # 배포시 사용하는 resource pools를 명시하며 여러 개의 resource pools 을 사용할 경우 name 은 unique 해야함(필수)  - cloud\_properties: # 컴파일 VM을 만드는 데 필요한 IaaS의 특정 속성을 설명 (instance\_type, availability\_zone), 직접 cpu, disk, 메모리 설정가능  ram: 1024  name: random  name: services-small  network: services  stemcell:  name: bosh-warden-boshlite-ubuntu-trusty-go\_agent # stemcell 이름(필수)  version: latest # stemcell 버전(필수)  jobs:  - name: dedicated-node # 작업 이름(필수)  templates:  - name: dedicated-node # job template 이름(필수)  release: openpaas-redis #서비스 릴리즈 이름(필수)  - name: syslog-configurator  release: openpaas-redis  instances: 5 # 생성하는 인스턴스 수(필수)  resource\_pool: services-small  persistent\_disk: 4096 # 영구적 디스크 사이즈 정의(옵션): 4G  networks:  - name: services  static\_ips: # 전용 노드 IP 목록(필수)  - 10.244.3.54  - 10.244.3.58  - 10.244.3.62  - 10.244.3.66  - 10.244.3.70  - name: openpaas-redis-broker  templates:  - name: cf-redis-broker # job template 이름(필수)  release: openpaas-redis #서비스 릴리즈 이름(필수)  - name: syslog-configurator  release: openpaas-redis  instances: 1  resource\_pool: services-small  persistent\_disk: 4096 # 영구적 디스크 사이즈 정의(옵션): 4G  networks:  - name: services  static\_ips:  - 10.244.3.46  - name: broker-registrar # 작업 이름: 서비스 브로커 등록  instances: 1  lifecycle: errand # bosh deploy시 vm에 생성되어 설치 되지 않고 bosh errand 로 실행할때 설정, 주로 테스트 용도에 쓰임  networks:  - name: services  resource\_pool: services-small  templates:  - name: broker-registrar  release: openpaas-redis  properties:  broker: # 서비스 브로커 설정 정보  name: openpaas-redis-service  host: redis-broker.10.244.0.34.xip.io  username: admin  password: admin  - name: broker-deregistrar  instances: 1  lifecycle: errand  networks:  - name: services  resource\_pool: services-small  templates:  - name: broker-deregistrar  release: openpaas-redis  properties:  broker:  name: openpaas-redis-service  host: redis-broker.10.244.0.34.xip.io  username: admin  password: admin  - name: smoke-tests  instances: 1  lifecycle: errand  networks:  - name: services  resource\_pool: services-small  templates:  - name: smoke-tests  release: openpaas-redis  properties:  syslog\_aggregator:  address: 127.0.0.1  port: 1234  cf:  api\_url: http://api.10.244.0.34.xip.io # CF API url(필수)  apps\_domain: 10.244.0.34.xip.io # CF 도메인(필수)  admin\_username: admin # CF 어드민 아이디 (필수)  admin\_password: admin # CF 어드민 아이디 비밀번호(필수)  nats: # CF 설치시 설치한 nats 정보 (필수)  host: 10.244.0.6  port: 4222  username: nats  password: nats  redis:  host: 10.244.3.46  maxmemory: 262144000 # 최대 메모리  config\_command: config # config 설정 명령어가 있을 경우(옵션)  save\_command: save # 저장 명령어가 있을 경우(옵션)  bg\_save\_command: bgsave  agent:  backend\_port: 54321 # Redis backend 포트번호(필수)  broker:  network: services  backend\_host: 10.244.3.46  backend\_port: 12345  dedicated\_nodes:  - 10.244.3.54  - 10.244.3.58  - 10.244.3.62  - 10.244.3.66  - 10.244.3.70  service\_instance\_limit: 3 # 서비스 인스턴스 생성 가능 수(필수)  service\_name: redis-sb  auth:  username: admin  password: admin  backups:  path: "/backups"  access\_key\_id: ZZZZZZZZZZZZZZZ  secret\_access\_key: ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ  endpoint\_url: https://s3.amazonaws.com  bucket\_name: my-backup-bucket |

* Deploy 할 deployment manifest 파일을 BOSH 에 지정한다.

|  |
| --- |
| $ bosh deployment {Deployment manifest 파일 PATH}  $ bosh deployment openpaas-redis-lite.yml |

* Redis 서비스팩을 배포한다.

|  |
| --- |
| $ bosh deploy |

* 배포된 Redis 서비스팩을 확인한다.

|  |
| --- |
| $ bosh vms |

## ***Redis 서비스 브로커 등록***

Redis 서비스팩 배포가 완료 되었으면 Application에서 서비스 팩을 사용하기 위해서 먼저 Redis 서비스 브로커를 등록해 주어야 한다.

서비스 브로커 등록시 개방형 클라우드 플랫폼에서 서비스 브로커를 등록 할 수 있는 사용자로 로그인이 되어 있어야 한다.

* 서비스 브로커 목록을 확인한다.

|  |
| --- |
| $ cf service-brokers |

* Redis 서비스 브로커를 등록한다.

|  |
| --- |
| $ cf create-service-broker {서비스팩 이름} {서비스팩 사용자ID} {서비스팩 사용자비밀번호} http://{서비스팩 URL}   * 서비스팩 이름 : 서비스 팩 관리를 위해 개방형 클라우드 플랫폼에서 보여지는 명칭이다. 서비스 Marketplace에서는 각각의 API 서비스 명이 보여지니 여기서 명칭은 서비스팩 리스트의 명칭이다. * 서비스팩 사용자ID / 비밀번호 : 서비스팩에 접근할 수 있는 사용자 ID이다. 서비스팩도 하나의 API 서버이기 때문에 아무나 접근을 허용할 수 없어 접근이 가능한 ID/비밀번호를 입력한다. * 서비스팩 URL : 서비스팩이 제공하는 API를 사용할 수 있는 URL을 입력한다.   $ cf create-service-broker redis-service-broker admin admin http://10.244.3.46:80 |

* 등록된 Redis 서비스 브로커를 확인한다.

|  |
| --- |
| $ cf service-brokers |

* 접근 가능한 서비스 목록을 확인한다.

|  |
| --- |
| $ cf service-access    서비스 브로커 생성시 디폴트로 접근을 허용하지 않는다. |

* 특정 조직에 해당 서비스 접근 허용을 할당하고 접근 서비스 목록을 다시 확인한다. (전체 조직)

|  |
| --- |
| $ cf enable-service-access redis-sb  $ cf service-access |

# Redis 연동 Sample App 설명

본 Sample App은 개발형 클라우드 플랫폼에 배포되며 Redis의 서비스를 Provision과 Bind를 한 상태에서 사용이 가능하다.

## ***Sample App 구조***

Sample App은 개방형 클라우드 플랫폼에 App으로 배포가 된다. 배포 완료 후 정상적으로 App 이 구동되면 curl 명령어로 배포된 도메인 정보를 이용하여 Redis에 특정 key 에 값을 저장/조회/삭제를 한다.

Sample App 구조는 다음과 같다.

|  |  |
| --- | --- |
| **이름** | **설명** |
| lib | Sample 소스 디렉토리 |
| manifest.yml | 개방형 클라우드 플랫폼에 app 배포시 필요한 설정을 저장하는 파일 |
| Gemfile | Sample App 구동시 필요한 ruby gem 설정 파일 |
| config.ru | Sample App 구동 파일 |

* OpenPaaS-Sample-Apps.zip 파일 압축을 풀고 Service 폴더안에 있는 Redis Sample Web App인 redis-example-app을 복사한다.

|  |
| --- |
| $ ls -all |

## ***개방형 클라우드 플랫폼에서 서비스 신청***

Sample App에서 Redis 서비스를 사용하기 위해서는 서비스 신청(Provision)을 해야 한다.

\*참고: 서비스 신청시 개방형 클라우드 플랫폼에서 서비스를 신청 할 수 있는 사용자로 로그인이 되어 있어야 한다.

* 먼저 개방형 클라우드 플랫폼 Marketplace에서 서비스가 있는지 확인을 한다.

|  |
| --- |
| $ cf marketplace |

* Marketplace에서 원하는 서비스가 있으면 서비스 신청(Provision)을 한다.

|  |
| --- |
| $ cf create-service {서비스명} {서비스플랜} {내서비스명}   * 서비스명 : redis-sb 로 Marketplace에서 보여지는 서비스 명칭이다. * 서비스플랜 : 서비스에 대한 정책으로 plans에 있는 정보 중 하나를 선택한다. Redis 서비스는 shared-vm 과 dedicated-vm plan을 지원한다. * 내서비스명 : 내 서비스에서 보여지는 명칭이다. 이 명칭을 기준으로 환경설정정보를 가져온다.   $ cf create-service redis-sb shared-vm redis-service-instance |

* 생성된 Redis 서비스 인스턴스를 확인한다.

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

## ***Sample App에 서비스 바인드 신청 및 App 확인***

서비스 신청이 완료되었으면 Sample App 에서는 생성된 서비스 인스턴스를 Bind 하여 App에서 Redis 서비스를 이용한다.

\*참고: 서비스 Bind 신청시 개방형 클라우드 플랫폼에서 서비스 Bind 신청 할 수 있는 사용자로 로그인이 되어 있어야 한다.

* Sample App 디렉토리로 이동하여 manifest 파일을 확인한다.

|  |
| --- |
| $ cd redis-example-app  $ vi manifest.yml  ---  applications:  - name: redis-example-app # 배포할 App 이름  memory: 256M # 배포시 메모리 크기  instances: 1 # 배포시 인스턴스 수  path: . # 배포시 구동되는 Path |

* --no-start 옵션으로 App을 배포한다.

--no-start: App 배포시 구동은 하지 않는다.

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

* 배포된 Sample App을 확인하고 로그를 수행한다.

|  |
| --- |
| $ cf apps    $ cf logs {배포된 App명}  $ cf logs redis-example-app |

* Sample App에서 생성한 서비스 인스턴스 바인드 신청을 한다.

|  |
| --- |
| $ cf bind-service redis-example-app redis-service-instance |

* 바인드가 적용되기 위해서 App을 재기동한다.

|  |
| --- |
| $ cf restart redis-example-app |

* App이 정상적으로 Redis 서비스를 사용하는지 확인한다.

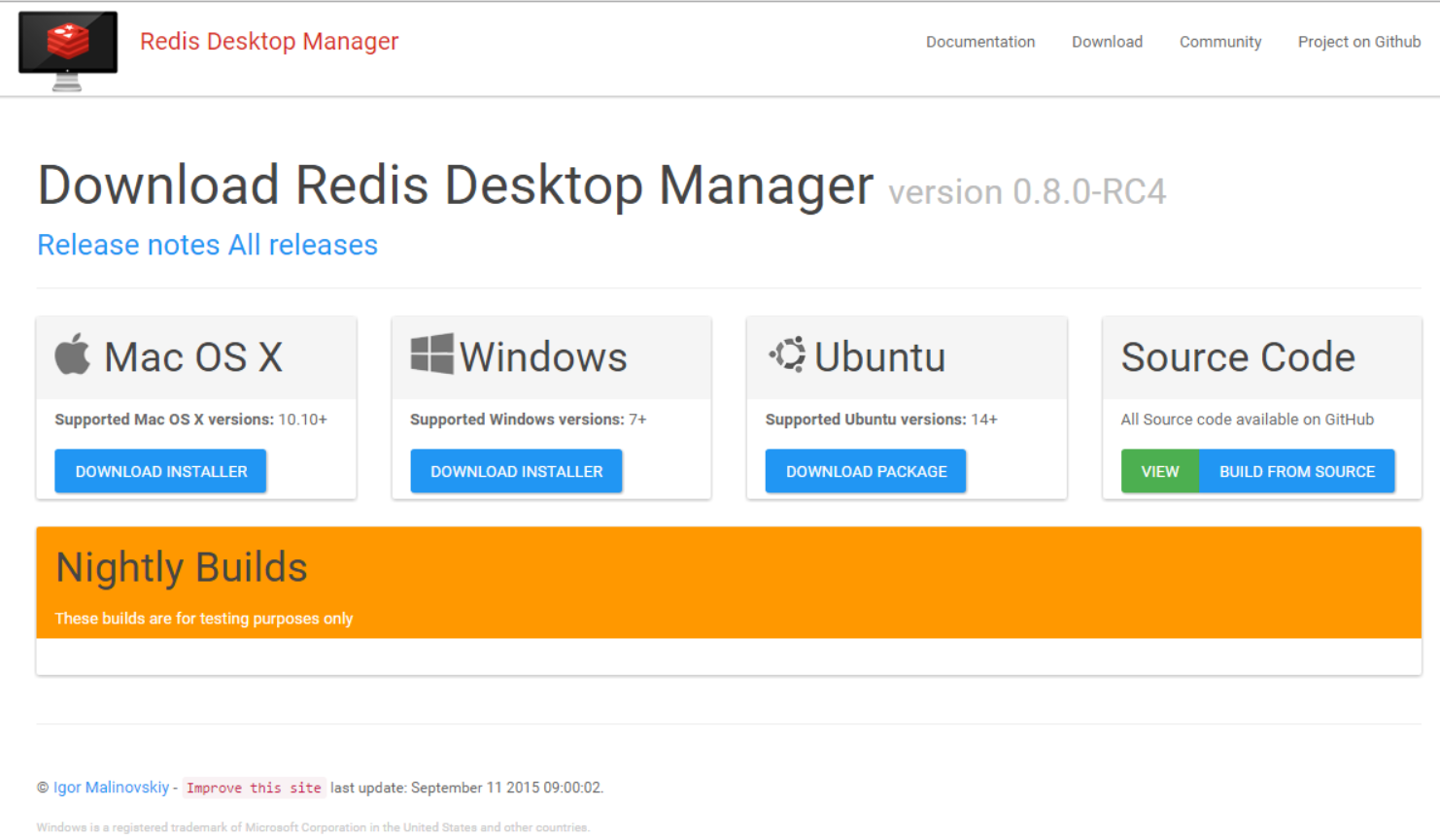
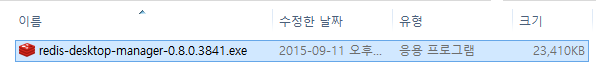
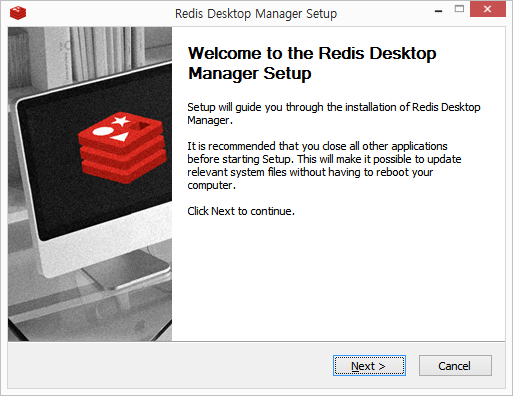
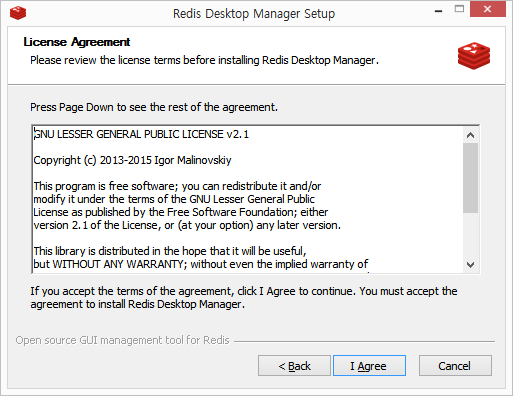
|  |
| --- |
| * curl 로 확인 * Redis Application 을 이용하여 Key 값으로 데이터 저장   $ export APP=redis-example-app.10.244.0.34.xip.io  $ curl -X PUT $APP/foo -d 'data=bar'     * Key 값을 이용하여 Redis 에 저장되어 있는 Value 값 Get   $ curl -X GET $APP/foo     * Key 값을 이용하여 Redis 에 저장되어 있는 Value 삭제   $ curl -X DELETE $APP/foo |

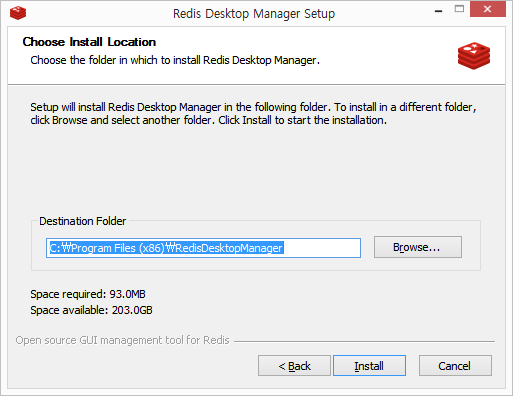
# Redis Client 툴 접속

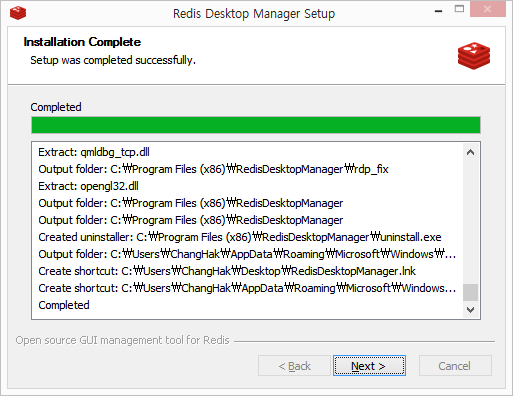
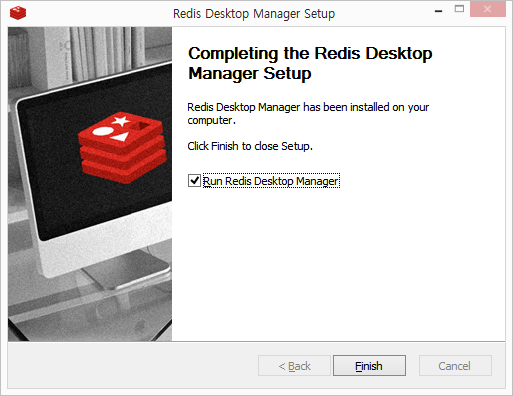
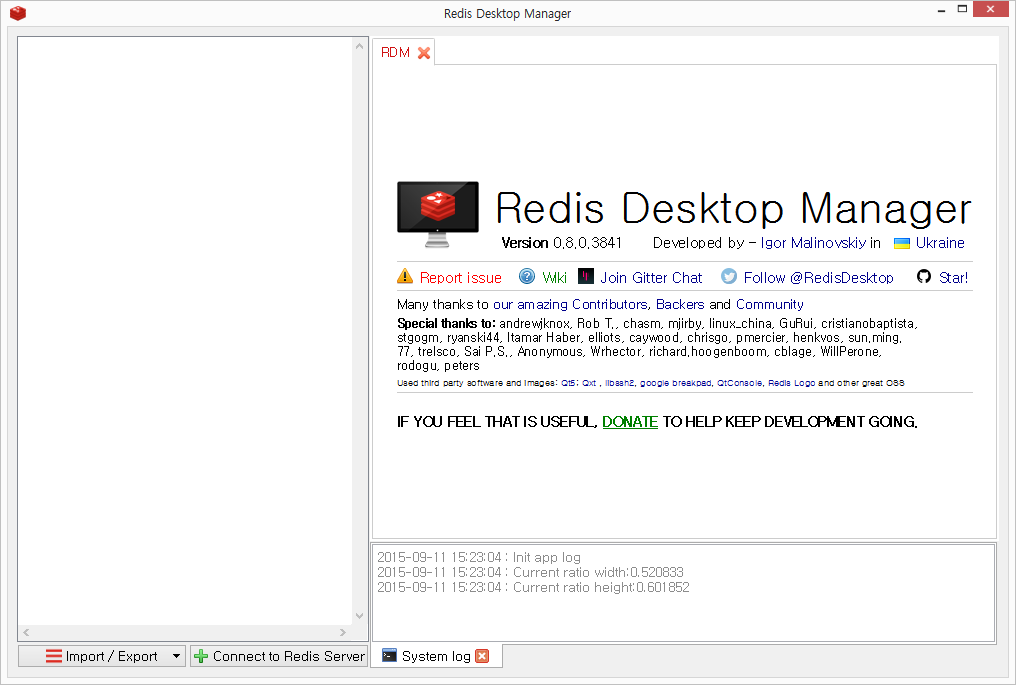
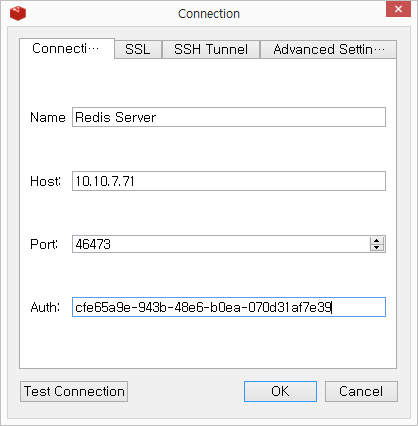
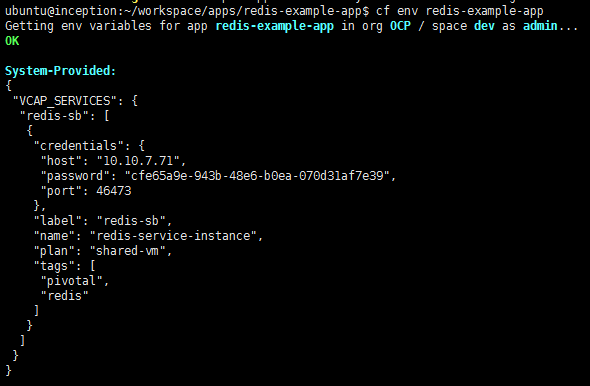
Application에 바인딩된 Redis 서비스 연결정보는 Private IP로 구성되어 있기 때문에 Redis Client 툴에서 직접 연결할수 없다. 따라서 Redis Client 툴에서 SSH 터널, Proxy 터널 등을 제공하는 툴을 사용해서 연결하여야 한다. 본 가이드는 SSH 터널을 이용하여 연결 하는 방법을 제공하며 Redis Client 툴로써는 오픈 소스인 **Redis Desktop Manager**로 가이드한다. 또한 Bosh lite를 AWS 환경에서 구성 한 경우를 전제로 하였다. AWS에서 Bosh lite를 구성하면 Vagrant VM이 생성되는데 Vagrant VM 에서는 서비스팩의 Private IP 와 해당 포트로 접근이 가능하도록 구성되어 있다.

## ***Redis Desktop Manager 설치 & 연결***

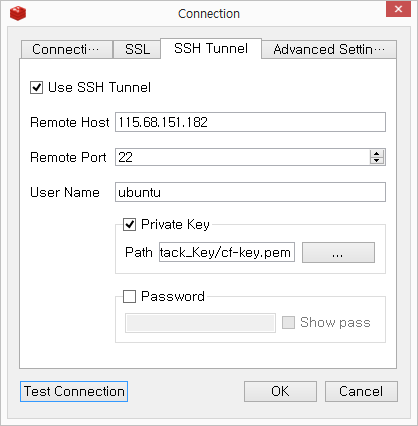
Redis Desktop Manager 프로그램은 무료로 사용할 수 있는 오픈소스 소프트웨어이다.

* Redis Desktop Manager를 다운로드 하기 위해 아래 URL로 이동하여 설치파일을 다운로드 한다.  
  <http://redisdesktop.com/download>  
  
* 다운로드한 설치파일을 실행한다.  
  
* Redis Desktop Manager 설치를 위한 안내사항이다. Next 버튼을 클릭한다.  
  
* 프로그램 라이선스에 관련된 내용이다. I Agree 버튼을 클릭한다.  
  
* Redis Desktop Manager를 설치할 경로를 설정 후 Install 버튼을 클릭한다.

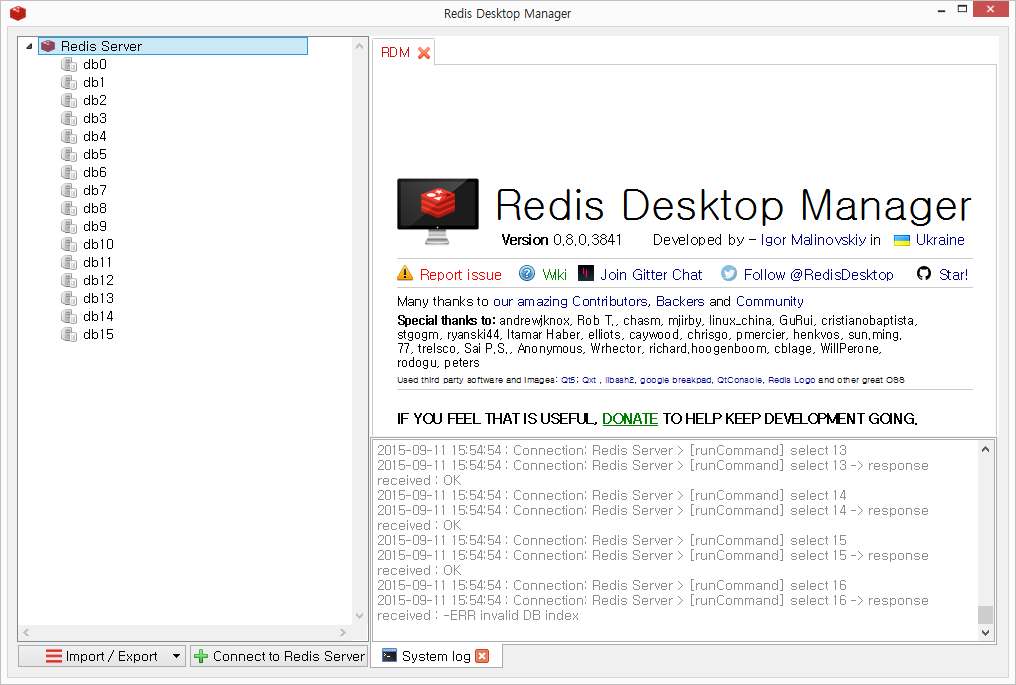
별도의 경로 설정이 필요 없을 경우 default로 C드라이브 Program Files 폴더에 설치가 된다.  


* 설치 완료 후 Next 버튼을 클릭하여 다음 과정을 진행한다.  
  
* Finish 버튼 클릭으로 설치를 완료한다.  
  
* Redis Desktop Manager를 실행했을 때 처음 뜨는 화면이다. 이 화면에서 Server에 접속하기 위한 profile을 설정/저장하여 접속할 수 있다. Connect to Redis Server 버튼을 클릭한다.  
  
* Connection 탭에서 아래 붉은색 영역에 접속하려는 서버 정보를 모두 입력한다.  
    
  서버 정보는 Application에 바인드되어 있는 서버 정보를 입력한다. cf env <app\_name> 명령어로 이용하여 확인한다.  
  예) $ cf env redis-example-app  
  
* SSH Tunnel탭을 클릭하고 OpenPaaS 운영 관리자에게 제공받은 SSH 터널링 가능한 서버 정보를 입력하고 공개키(.pem) 파일을 불러온다. Test Connection 버튼을 클릭하면 Redis 서버에 접속이 되는지 테스트 하고 OK 버튼을 눌러 Redis 서버에 접속한다.

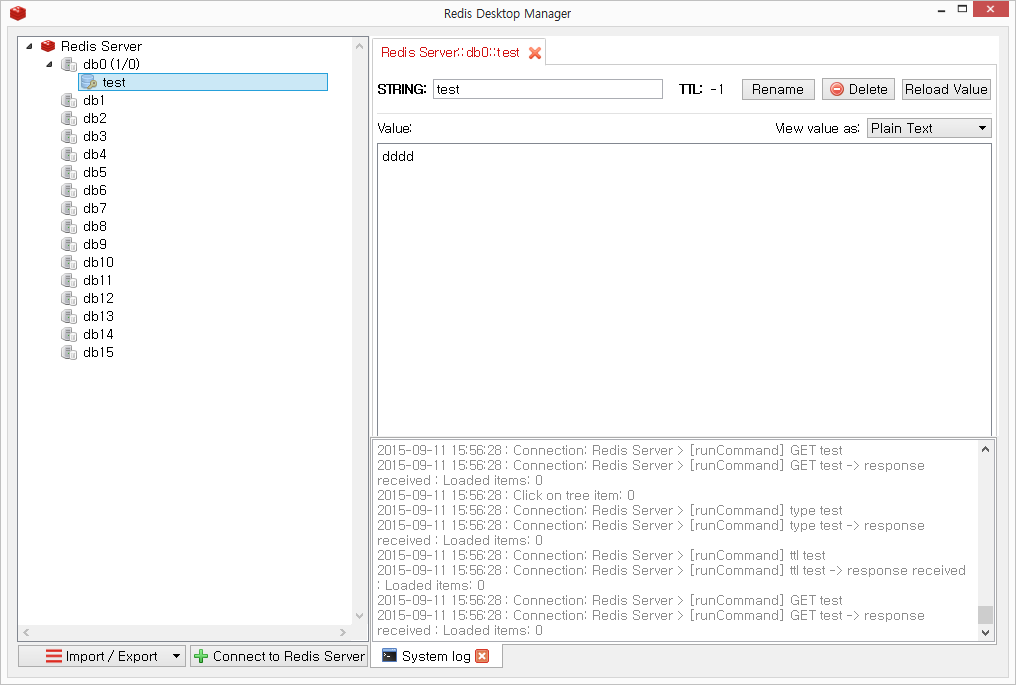
(참고) 만일 공개키 없이 ID/Password로 접속이 가능한 경우에는 공개키 대신 사용자 이름과 암호를 입력한다.



* 접속이 완료되고 좌측 서버 정보를 더블 클릭하면 좌측에 스키마 정보가 나타난다.



신규 키 등록후 확인



1. 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.) [↑](#footnote-ref-1)
2. YAML Ain’t Markup Language, http://www.yaml.org, http://ko.wikipedia.org/wiki/YAML [↑](#footnote-ref-2)
3. BOSH가 Stemcell로부터 복사된 VM을 제어할 수 있도록 BOSH Agent가 내장되어 있는데 이를 “Stemcell”이라 부른다. [↑](#footnote-ref-3)
4. Release는 시스템에서 설치될 구성 및 소프트웨어들을 포함한다. [↑](#footnote-ref-4)