오픈 SmartX Platform을 통한 OF@KOREN Playground 사용자 지원 개선

Document No. 13
Version 1.0
Date 2016-12-16
Author(s) GIST NetCS UnderCloud Team

■ **문서의 연혁**

버전	날짜	작성자	비고
초안 - 0.1	2016. 10. 10	이준기 연구원	
0.2	2016. 10. 20	이준기 연구원	
0.3	2016. 11. 04	남택호 연구원	
0.7	2016. 11. 14	이준기 연구원	
1.0	2016. 12. 16	이준기 연구원	

본 연구는 한국정보화진흥원(NIA)의 미래네트워크선도시험망(KOREN) 사업 지원과제의 연구결과로 수행되었음 (16-951-00-001).

This research was one of KOREN projects supported by National Information Society Agency (16-951-00-001).

Contents

#13. 오픈 SmartX Platform을 통한 OF@KOREN Playground 사용자 지원 개선

그림 목차

#13. 오픈 SmartX Platform을 통한 OF@KOREN Playground 사용자 지원 개선

1. OF@KOREN Playground 및 오픈 SmartX Platform 개요

1.1. 목적 및 개요

o 본 문서에서는 KOREN 네트워크 상에서 운용중인 OF@KOREN Playground를 사용자가 쉽게 이용할 수 있도록 2015년도에 구축된 기존의 오픈 SmartX Platform의 개선에 대한 내용을 다룬다. 본 문서에서 다루는 오픈 SmartX Platform 개선의 내용은 기존 Redmine 기반의 커뮤니티 사이트에서 GitHub 기반의 사이트로의 이전 방법과 사용자 지원 개선을 위해 추가된 내용들을 다룬다. 더 나아가 OF@KOREN Playground 사용자들을 위해 Playground의 상태를 모니터링 할 수 있는 Open API에 대한 내용을 다룬다.

1.2. OF@KOREN Playground

o Playground란 사용자들이 하고 싶은 Play를 자유롭게 할 수 있게 제공되는 공간을 상징하고 사용자들이 Playground 위에서 하는 다양한 실증 실험들은 Play로 표현할 수 있다.

o SmartX Playground는 사용자들이 원하는 실증 실험을 할 수 있도록 SmartX Box들을 유연하게 엮어 컴퓨팅, 네트워킹, 스토리지로 대표되는 IT 자원들을 알맞게 제공할 수 있는 물리적인 인프라를 의미한다. SmartX Virtual Playgroud(SmartX 가상놀이터)는 하나의 공통 인프라인 SmartX Playground 상에 DevOps 기반 자동화된 설치/설정 소프트웨어 도구를 활용하여 각 사용자의 요구사항에 따라 동적이고 유연하게 제공되는 실증 실험 환경을 의미한다.

o 즉, Playground는 다양한 사용자간 공유되는 물리적인 인프라 이지만 Virtual Playground는 하나의 공유 인프라 상에 각 사용자 별로 독립적으로 제공되는 실증 실험 공간이다. 이러한 Virtual Playground는 사용자의 요구에 따라 자동화된 설치/설정 도구를 통해 동적, 유연, 신속하게 구성되어야 한다.

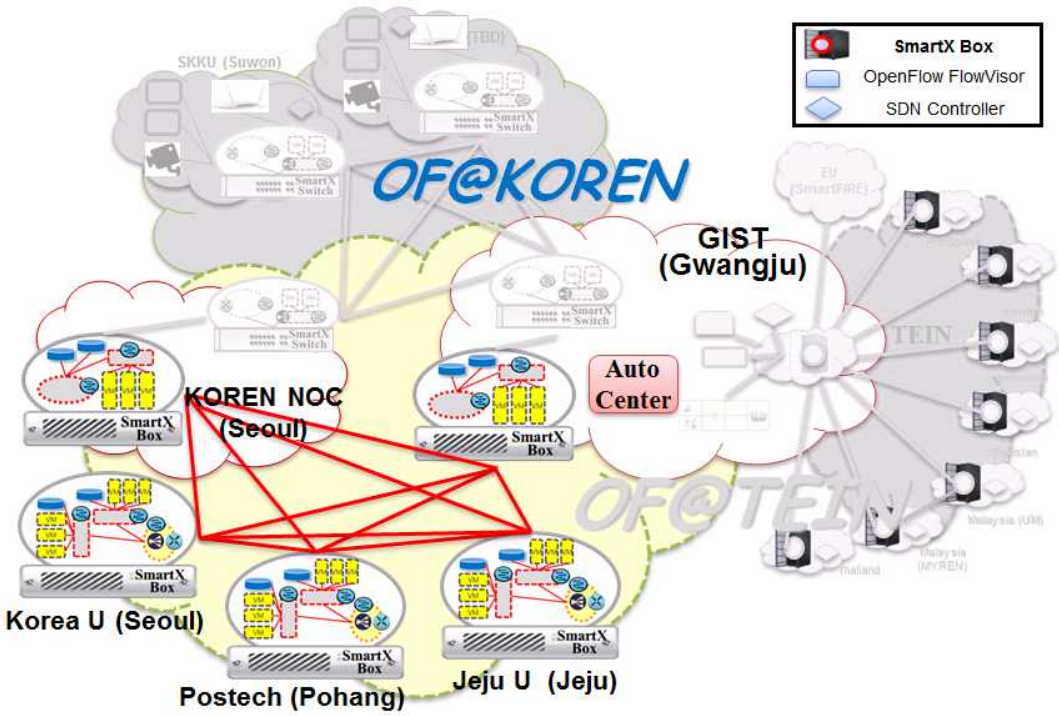

그림 1 OF@KOREN Playground 구성도

o 국내 미래인터넷 관련 연구자들을 대상으로 클라우드 기반의 미래인터넷 실증 실험 테스트베드를 제공하고자 2012년 광주과학기술원 NetCS 연구실을 중심으로 KOREN NOC(Network Operation Center), 고려대학교, 포항공과대학교, 제주대학교를 대상으로 초 융합형 자원 박스인 SmartX Box를 배포하고 이를 KOREN 연구망으로 연결하는 OF@KOREN Playground (i.e. 테스트베드)의 구축을 시작하였다.

o OF@KOREN Playground는 그림 1에서 보는 것과 같이 지리적으로 국내 각지에 분산되어 있는 SmartX Box를 OpenStack으로 엮어 하나의 분산 OpenStack 클라우드로 구성함으로써 컴퓨팅, 네트워킹, 스토리지 자원들을 사용자들에게 제공하고 있다. 이 때 KOREN 연구망을 중심으로 SmartX Box들을 연결하여 구성된 물리적인 인프라는 Playground라고 할 수 있고, 이 인프라 상에 구성된 OpenStack은 각 사용자들에게 독립적인 Cloud 환경인 Virtual Playground를 제공한다.

1.3. 오픈 SmartX Platform 개선의 필요성

o 기존 OF@KOREN Playground 사용자들을 위해 구축 및 제공되던 오픈 SmartX Platform의 경우 KOREN-NOC의 인프라에서 관리되는 서버에서 구축되어 운영되고 있었으며 이러한 특징으로 인해 KOREN 망에 접속 가용한 상황에서만 커뮤니티 사이트에 접속 가능하였다. 또한 OF@KOREN Playground의 활용을 위한 관련 문서 및 소스 코드의 제공에 있어서 사용자들의 불편함을 초래하는 구조로 구성되어 있었다. 따라서 OF@KOREN Playground를 효율적으로 사용자들에게 제공하기 위해서는 개선된 오픈 SmartX Platform을 구축, 운영하는 것이 반드시 필요하다.

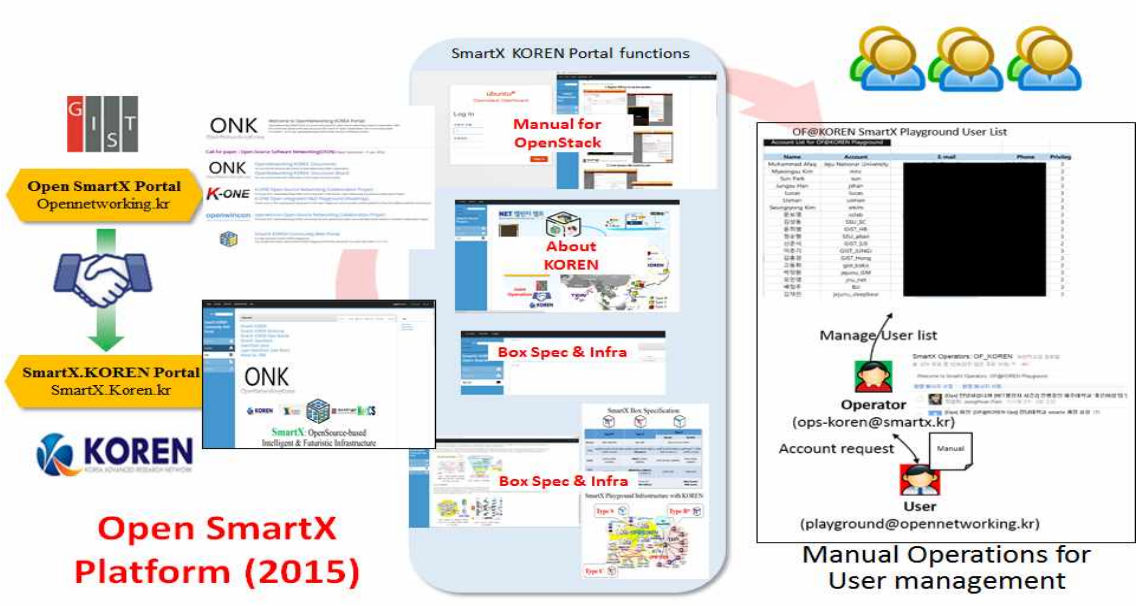

그림 2 2015 Open SmartX Platform for OF@KOREN Playground Users

2. 오픈 SmartX Platform 개선

2.1. 오픈 SmartX Platform 개선 개요

o 국내 여러 사이트의 분산형 SmartX Box를 활용하여 구성된 가상 놀이터 환경 OF@KOREN Playground의 사용자들이 쉽게 이용함과 더불어 동시에 관리자의 입장에서 사용자들을 관제(Monitor and Management)할 수 있도록 지원하는 오픈 SmartX Platform이 운용중이였으나, 사용자 친화를 목적으로 GitHub를 활용해 오픈 SmartX Platform을 개선한다.

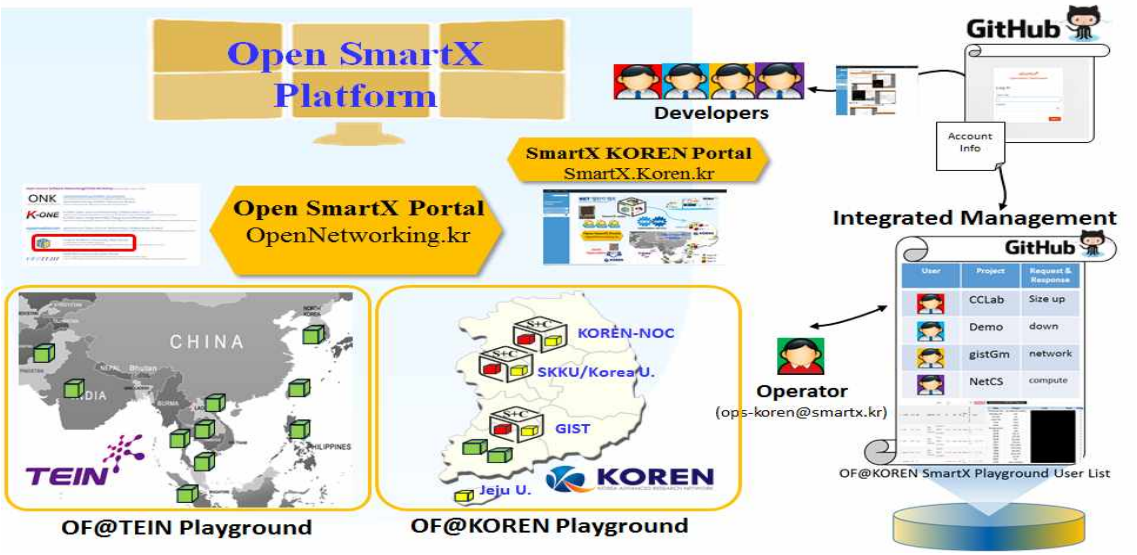

그림 3 오픈 SmartX Platform을 통한 OF@KOREN Playground 사용자 지원 개선 개념도

2.2. 오픈 SmartX Platform 개선 요구사항

o 기존 KOREN-NOC의 인프라에서 관리되는 서버에서 구축되어 운영되던 Redmine 기반 오픈 SmartX Platform의 경우 KOREN 망에 접속 가능한 경우에만 접속 가능하였다. 이번 기술문서에서는 이러한 단점을 개선 가능하도록 GitHub 기반으로 오픈 SmartX Platform을 구축하므로써 접근성을 확대한다.

o GitHub 기반 오픈 SmartX Platform 내부에는 OF@KOREN Playground를 처음 사용하는 사용자들을 위하여 GitHub 내의 wiki 페이지를 활용하여 튜토리얼을 제공한다. 또한 사용자-사용자, 사용자-개발자간의 소통을 위한 커뮤니티 사이트를 제공하며, OF@KOREN Playground 관련 정보 및 자료를 제공한다.

2.3. GitHub 기반 오픈 SmartX Platform 개선 구현 및 방법

o 상기 2.2절에서 명시한 요구사항에 따라 오픈 SmartX Platform의 Prototype을 아래 그림 3과 같은 구성으로 실제 구현하였다.

2.3.1. GitHub repository 생성 및 설정

o 오픈 SmartX Platform을 GitHub상에 구현하기 위해 아래와 같이 repository를 추가한다. 모든 사용자가 접근할 수 있도록 Public repository로 설정하고 메인 페이지를 구현하기 위해 Initialize this repository with a README 칸에 체크한다.

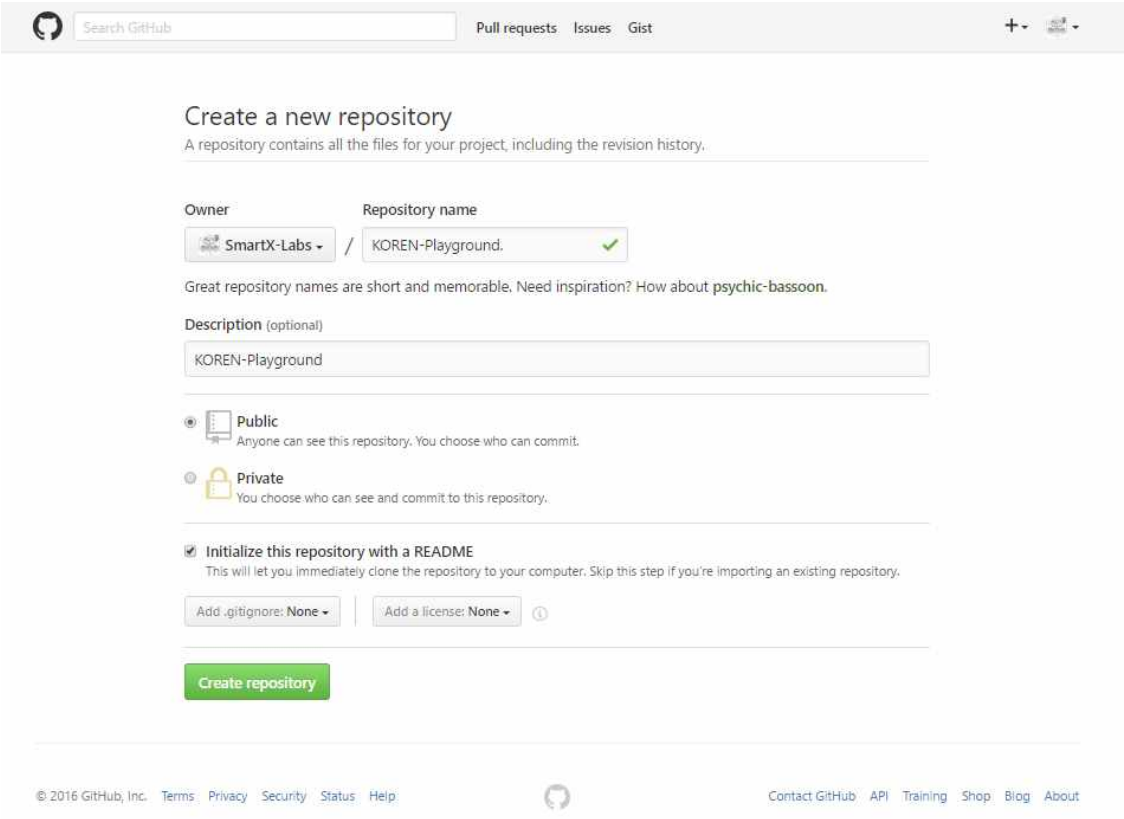

그림 4 SmartX Platform repository 생성

o GitHub 기반 오픈 SmartX Platform을 구축하기 앞서 사용될 사진들을 GitHub KOREN-Playground repository 내부 폴더에 다음 사진과 같이 업로드 한다.

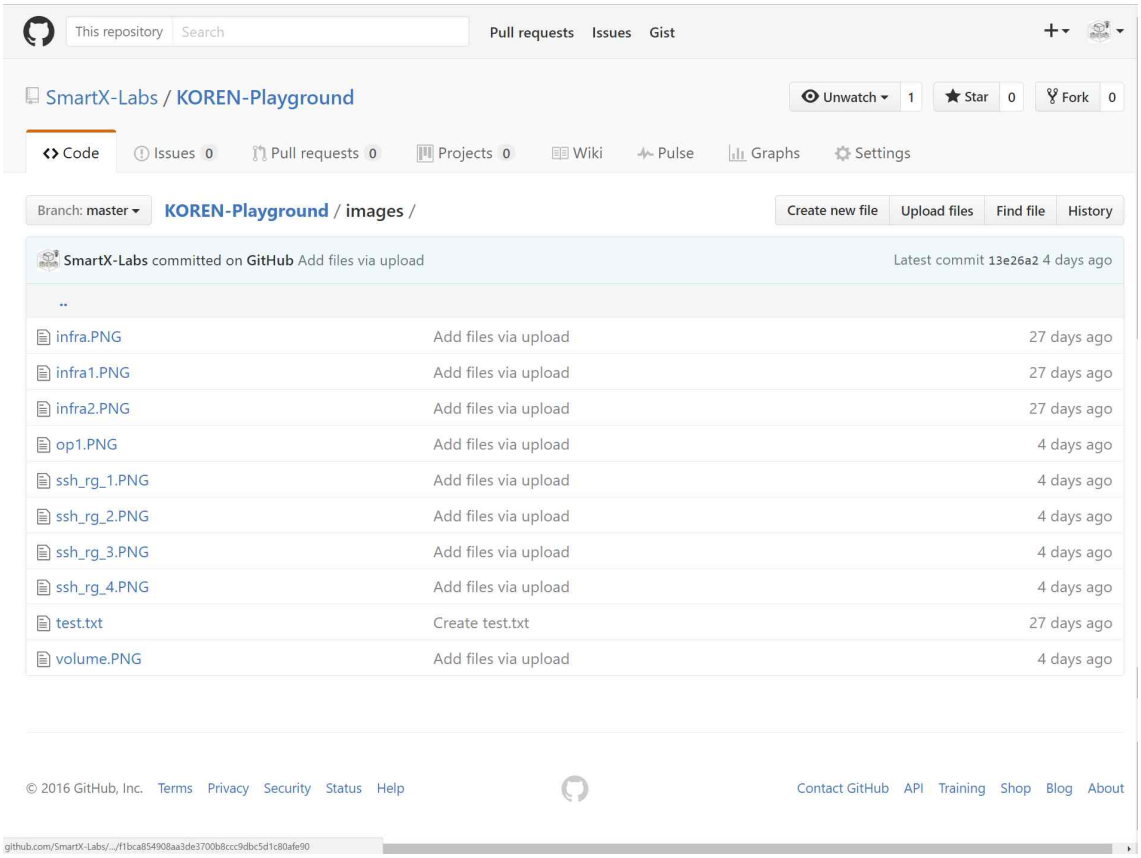

그림 5 SmartX Platform 내부 이미지 디렉토리

repository 내부 images 폴더에 업로드 된 사진들은 각 사진의 링크를 통해 GitHub 내부 페이지에 볼 수 있도록 첨부될 수 있다.

2.3.2. GitHub 메인 페이지 작성

o 오픈 SmartX Platform의 메인 페이지를 구축하기 위해 repository 최상위 폴더의 readme.md 파일을 수정한다. GitHub의 readme.md 파일은 마크다운을 기반으로 작성된다. 마크다운은 일종의 마크업 언어로 텍스트에 태그를 이용하여 글자의 크기 등을 설정하거나, 링크, 이미지 등을 삽입하는데 사용된다. [마크다운] 메인 페이지를 구현하는데 있어서 쓰이는 문법을 아래에서 설명한다.

o 오픈 SmartX Platform의 메인 페이지에서 OF@KOREN Playground의 OpenStack Dashboard 등의 다른 웹 페이지로 이동할 수 있는 링크를 삽입하기 위한 문법은 다음과 같다.

```
[텍스트][참조링크]
[참조명]: 링크주소 "링크제목"

예시
[About Us: ONK][onklink]
[onklink]: http://opennetworking.kr/projects/opennetworking/wiki "ONK"
```

위와 같은 명령어를 통해 메인 페이지에 성공적으로 링크가 작동하는 것을 확인할 수 있다.

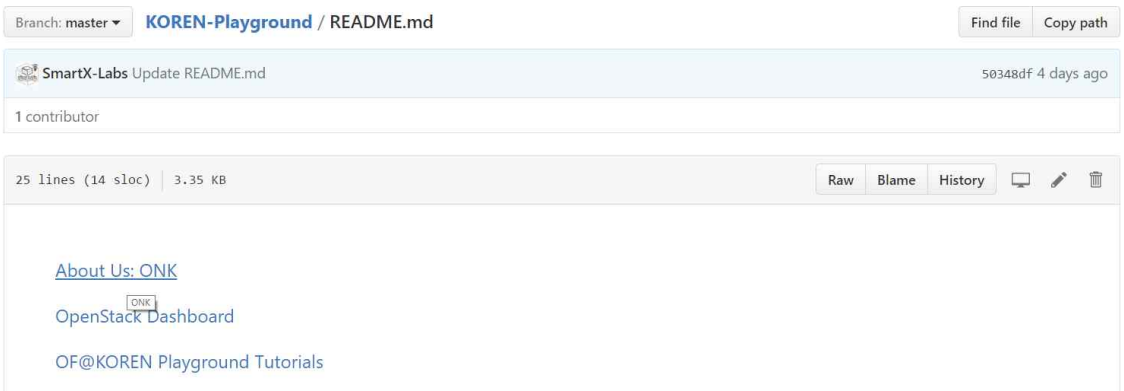

그림 6 오픈 SmartX Platform 메인페이지의 링크

o 오픈 SmartX Platform의 메인 페이지에서 OF@KOREN Playground에 대한 설명을 돕기위해 이미지를 첨부하기 위해서는 아래와 같은 명령어를 사용한다.

```
![](그림 주소)

예시
![](https://raw.githubusercontent.com/SmartX-Labs/KOREN-Playground/master/images/main.PNG)
```

위와 같은 명령어를 통해 메인 페이지에 성공적으로 사진이 업로드 되었음을 확인할 수 있다.

그림 7 오픈 SmartX Platform 메인페이지의 사진

o 또한 오픈 SmartX Platform의 메인 페이지에서 OF@KOREN Playground에 대한 설명을 작성하기 위한 텍스트의 옵션 문법은 아래와 같다.

```
제목 작성
# 제목

굵은 글씨
**텍스트**
```

o 오픈 SmartX Platform의 메인 페이지를 구축하기 위해 앞서 설명한 마크다운 언어의 문법을 활용하여, repository 최상위 폴더의 readme.md를 다음과 같이 수정한다.

```
# KOREN-Playground

![](https://raw.githubusercontent.com/SmartX-Labs/KOREN-Playground/master/images/main.png)

본 페이지는 OF@KOREN Playground 사용자들을 위한 커뮤니티 사이트입니다.

OF@KOREN Playground는 사용자들이 원하는 실증 실험을 할 수 있도록 SmartX Box들을 유연하게 엮어 컴퓨팅, 네트워킹, 스토리지로 대표되는 IT 자원들을 알맞게 제공할 수 있는 물리적인 인프라를 의미합니다.

SmartX Virtual Playground(SmartX 가상놀이터)는 하나의 공통 인프라인 OF@KOREN Playground 상에 DevOps 기반 자동화된 설치/설정 소프트웨어 도구를 활용하여 각 사용자의 요구사항에 따라 동적이고 유연하게 제공되는 실증 실험 환경을 의미합니다.

###아래의 그림을 클릭해 OF@KOREN Playground로 이동하세요.

[![](https://raw.githubusercontent.com/SmartX-Labs/KOREN-Playground/master/images/entrance.png)
][dashboard]
```

```
[dashboard]: http://210.114.90.12/horizon/auth/login/?next=/horizon/ "OpenStack Dashboard"
OF@KOREN Playground에 대한 보다 더 자세한 설명은 아래 링크를 통해 확인하세요.
####[OF@KOREN Playground wiki][tuto]
[tuto]: https://github.com/SmartX-Labs/KOREN-Playground/wiki "Tuturial Page"
[![](https://raw.githubusercontent.com/SmartX-Labs/KOREN-Playground/master/images/temp_ONK.png)
][onklink]
[onklink]: http://opennetworking.kr/projects/opennetworking/wiki "ONK"
```

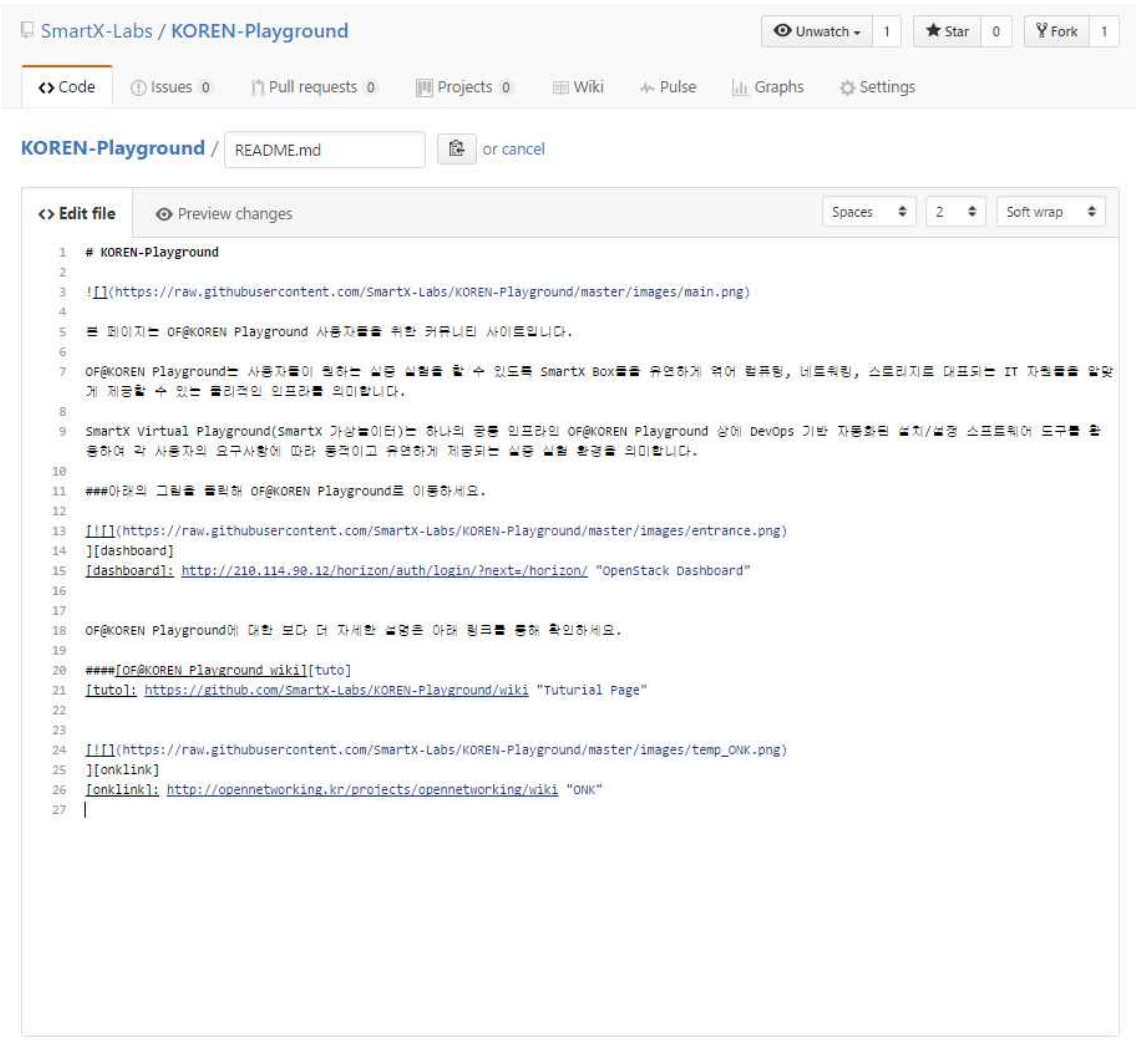

그림 8 오픈 SmartX Platform 메인 페이지를 위한 readme.md 설정

o 앞서 설명한 마크다운 문법을 활용해 위와 같은 내용으로 readme.md를 작성해 구현한 오픈 SmartX Platform의 메인 페이지는 다음 그림과 같다. 메인 페이지는 OF@KOREN Playground와 관련된 링크와 OF@KOREN Playground를 총괄하는 그림 및 설명으로 이루어져 있다.

2.3.3. GitHub wiki 페이지 작성

o 다음으로 처음 OF@KOREN Playground를 사용하는 OF@KOREN Playground 사용자들을 위한 튜토리얼 사이트를 작성하기 위해 KOREN-Playground repository 내부에 wiki 페이지를 추가한다. wiki page를 만드는 방법은 다음과 같다.

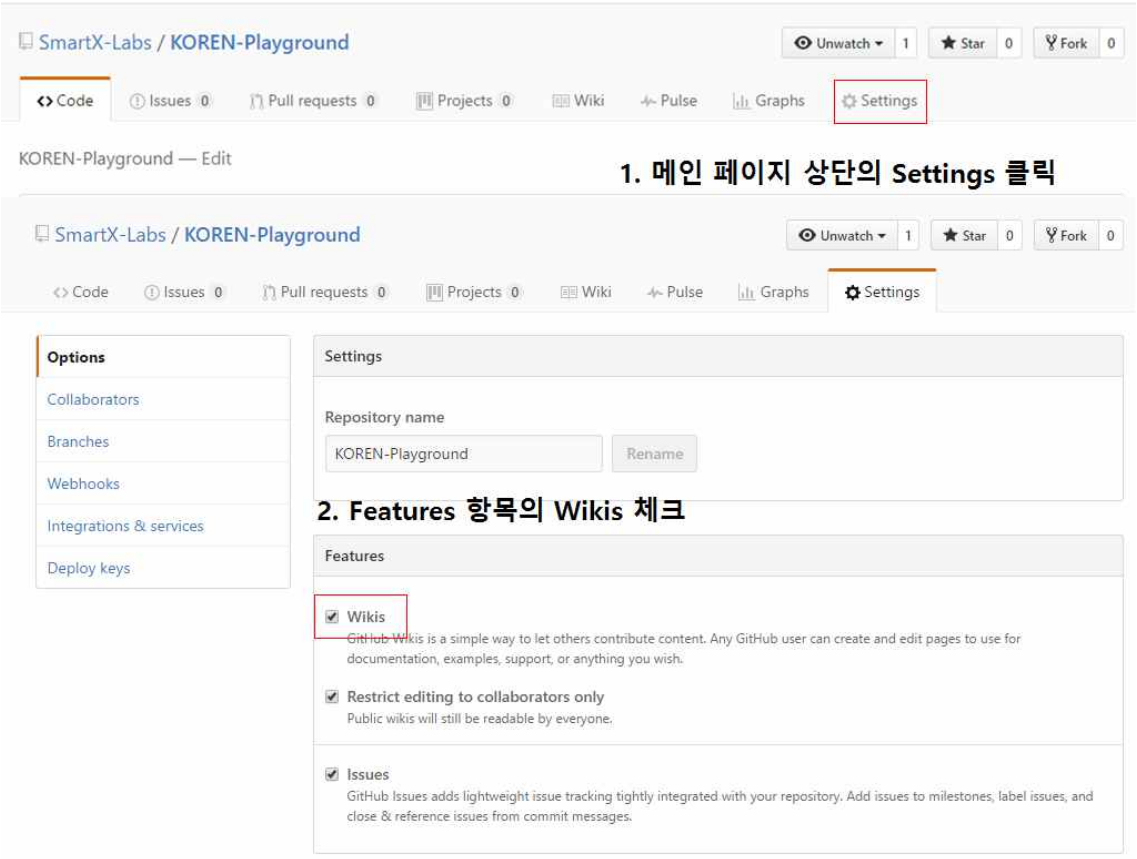

그림 9 GitHub 내부 wiki 탭 추가

o wiki page 또한 앞에서 설명한 마크다운 언어를 통해 작성할 수 있다. 처음 OF@KOREN Playground 사용자의 OpenStack 계정 신청 및 사용법을 설명하는 페이지를 만들기 위해 다음 내용을 마크다운 언어로 아래와 같이 작성한다.

```
# Welcome to OF@KOREN Playground wiki

![](https://raw.githubusercontent.com/SmartX-Labs/KOREN-Playground/master/images/main.png)

안녕하세요. 이 페이지는 OF@KOREN Playground 사용자를 위한 wiki 페이지입니다.

OF@KOREN Playground를 처음 사용하시는 분께서는

[Playground 사용법]
(https://github.com/SmartX-Labs/KOREN-Playground/wiki/Playground-%EC%82%AC%EC%9A%A9%EB%B2%95)으로 이동해주세요.

아래 링크들을 통해 원하는 페이지로 이동하실 수 있습니다.

질문이나 피드백은 언제나 환영합니다. 아래의 메일 주소나 issues 페이지를 통해 소중한 의견 부탁드립니다.

문의:
 ops@smartx.kr, [Playground issues]
(https://github.com/SmartX-Labs/KOREN-Playground/issues)

##OF@KOREN Platform Links

[Playground 소개]
(https://github.com/SmartX-Labs/KOREN-Playground/wiki/Playground-%EC%86%8C%EA%B0%9C)

[Playground 사용법]
(https://github.com/SmartX-Labs/KOREN-Playground/wiki/Playground-%EC%82%AC%EC%9A%A9%EB%B2%95)

[Playground Map]
(https://github.com/SmartX-Labs/KOREN-Playground/wiki/Playground-Map)
```

[Playground Software and Documents]
(https://github.com/SmartX-Labs/KOREN-Playground/wiki/Playground-Softwares-and-Documents)

[Playground issues]
(https://github.com/SmartX-Labs/KOREN-Playground/issues)

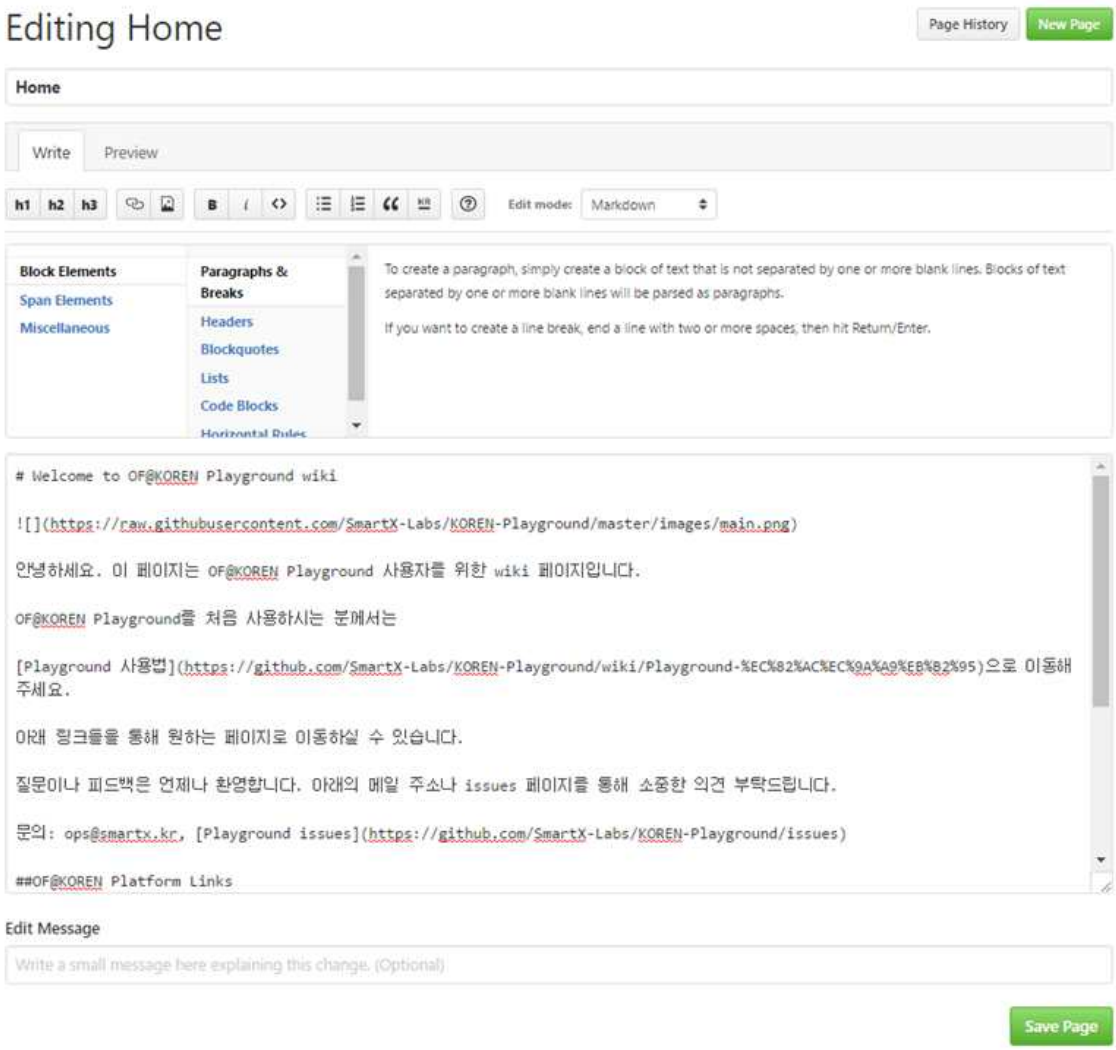

그림 10 OF@KOREN Playground 사용자를 위한 안내 wiki 페이지 생성

o 위와 같은 설정으로 생성된 wiki 페이지는 다음 그림과 같다.

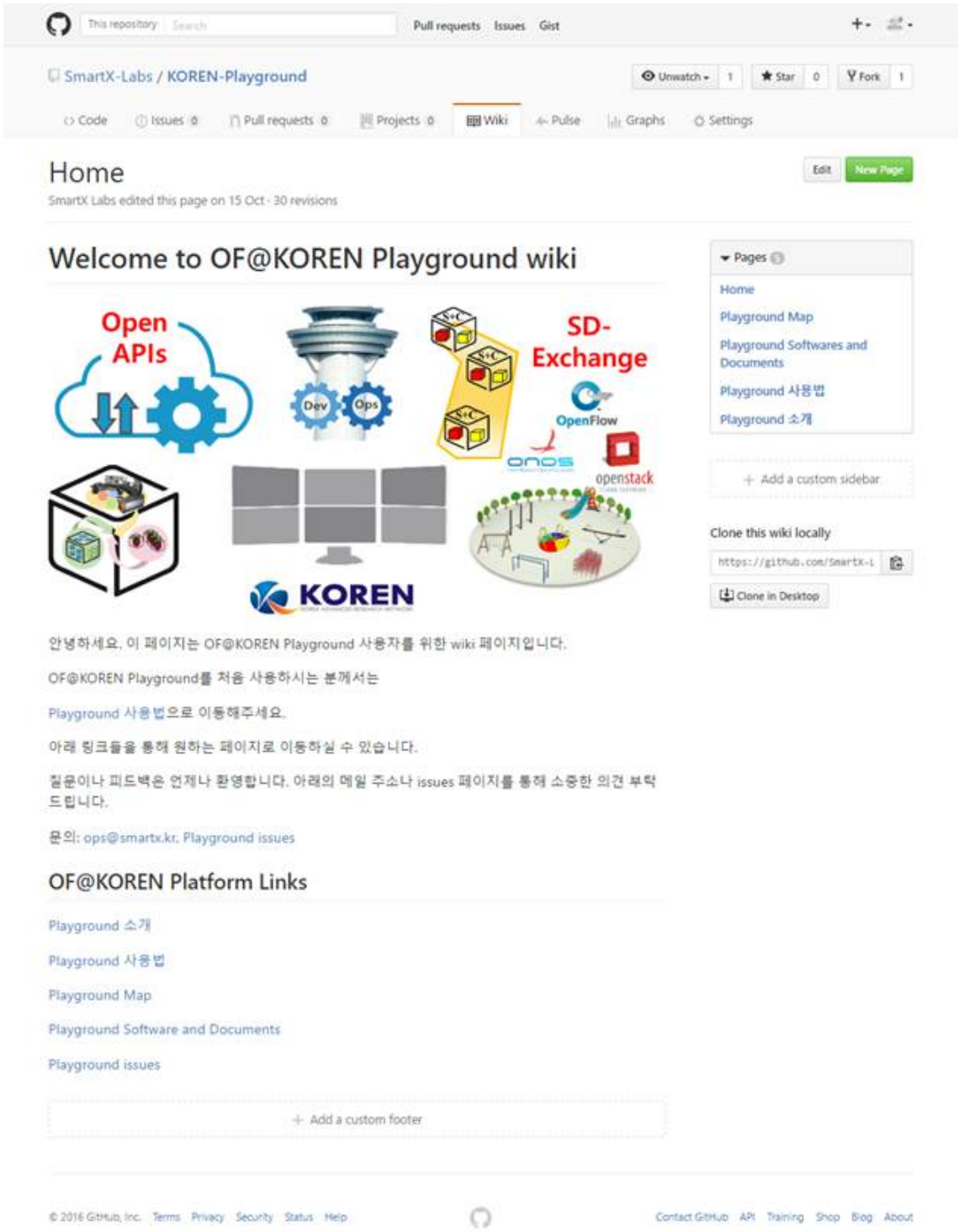

그림 11 OF@KOREN Playground 사용자를 위한 안내 wiki 페이지

2.3.4. GitHub 내부 사용자 커뮤니티 생성

o OF@KOREN Playground의 사용자들의 커뮤니티 사이트를 구현하기 위해 issues 탭을 활성화 한다. 방법은 다음과 같다.

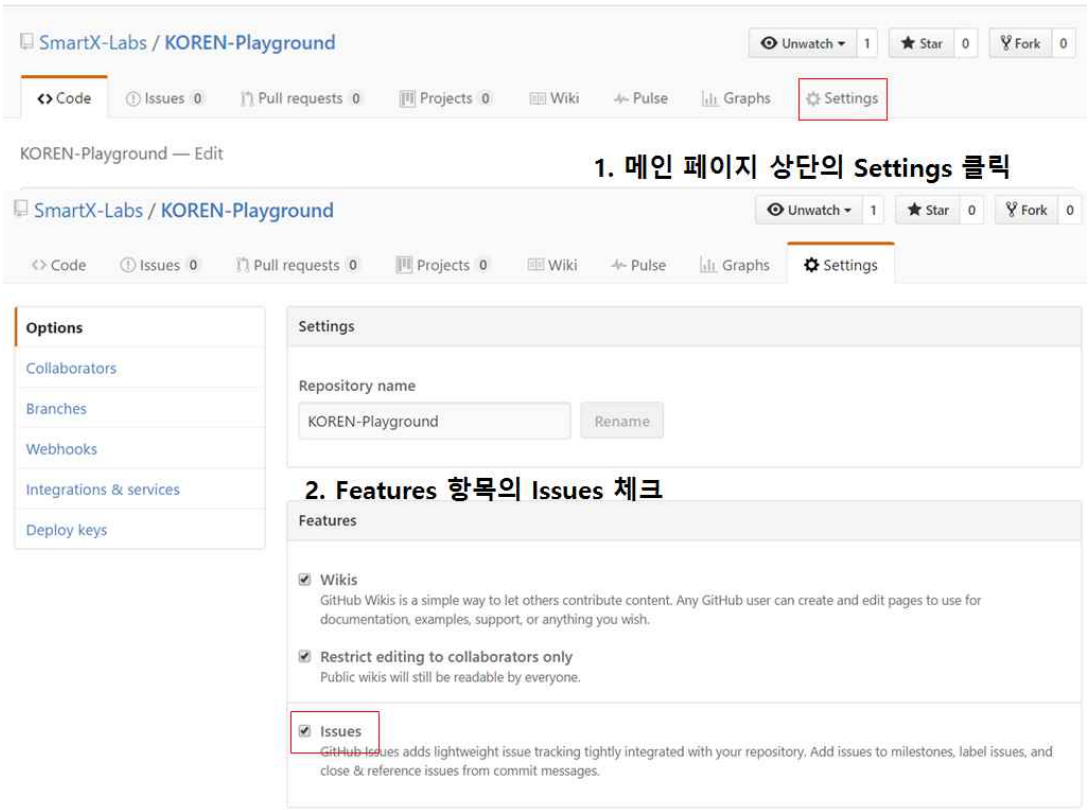

그림 12 GitHub 내부 issues 탭 추가

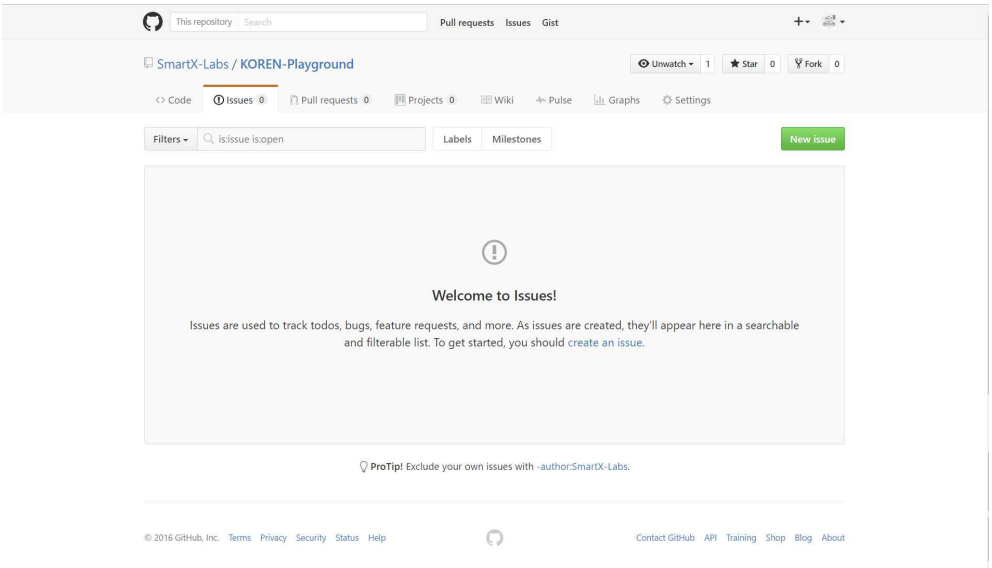

그림 13 GitHub 내 생성된 Issues 탭

2.4. 오픈 SmartX Platform의 개선 및 실증

o 기존 KOREN-NOC의 인프라에서 운용되던 redmine 기반의 오픈 SmartX Platform의 경우 접속하기 위해서는 KOREN 망에 접속 가능한 상황에서만 커뮤니티 사이트에 접속이 가능했다. 또한 정해진 주소를 통해서만 접속이 가능했기 때문에 주소를 기억하지 못하면 쉽게 접근이 어려운 문제점이 있었다.

o 이와 같은 문제점에 대한 개선사항으로 사용자 입장에서 사이트의 URL 주소가 아닌 Github에서 repository에 대한 검색을 통해서 보다 쉽게 접근이 가능하게 개선했다. 개선된 형태의 오픈 SmartX Platform의 경우 Github 내에서 KOREN-Playground라는 단어 검색을 통해 쉽게 접근할 수 있으며, KOREN 망에 접속이 불가능한 상황에서도 Github 서버 접속을 통해 시간 및 장소에 구애받지 않고 접속이 가능하다.

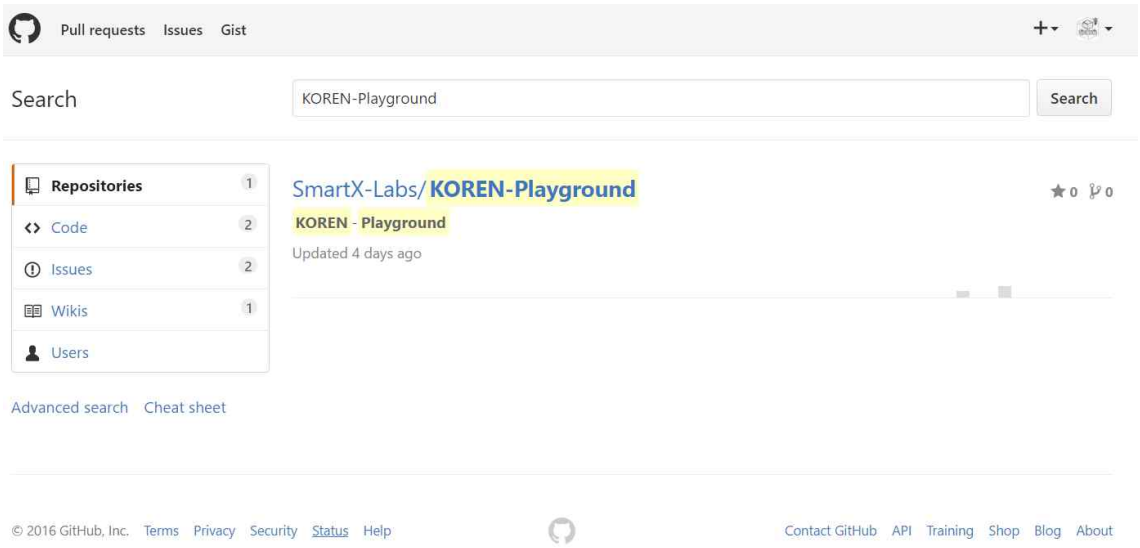

그림 14 GitHub 내부 KOREN-Playground 검색 결과

o 또한 기존 redmie 기반 오픈 SmartX Platform의 경우 OF@KOREN Playground 관련 문서 및 자료들을 찾기 위해서는 여러 메뉴를 직접 방문해야 파일들을 찾을 수 있었으나, 개선된 오픈 SmartX Platform의 경우 통합된 Github 내부 폴더를 통해 사용자가 손쉽게 원하는 자료를 얻을 수 있다. 더불어 자료의 업데이트가 발생한 경우 Github 자체 알림 및 e-mail을 통해 업데이트 정보를 얻을 수 있다.

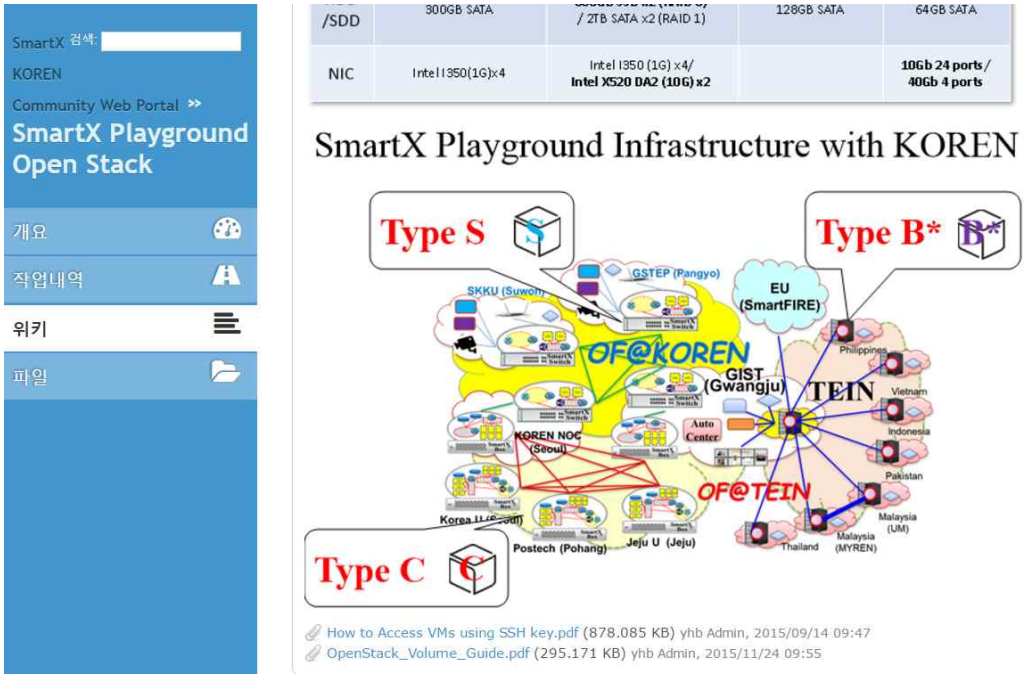

그림 15 기존 여러 페이지에 분산되어 있는 관련 자료들

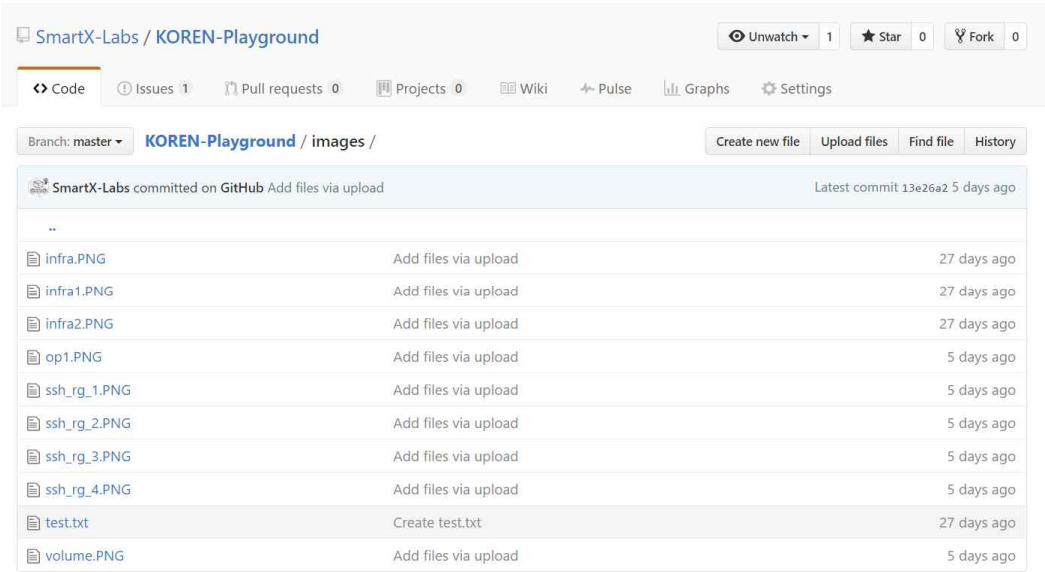

그림 16 GitHub 저장소에 정돈되어 있는 자료들

o 기존 redmine 기반 오픈 SmartX 플랫폼 내 사용자 커뮤니티의 경우 새로운 issue가 등록되거나, 그에 대한 답변이 등록되었을 때 수동으로 일일이 확인 해야 하는 불편함이 존재했다. 이런 문제점으로 인해 사용자들 사이에서, 또는 사용자와 개발자들 사이에서 소통에 어려움이 발생했다. 개선된 사용자 커뮤니티의 경우 Github 서버가 제공하는 메일 서비스를 통해 사용자 및 개발자들 사이의 빠르고 정확한 소통이 가능하다.

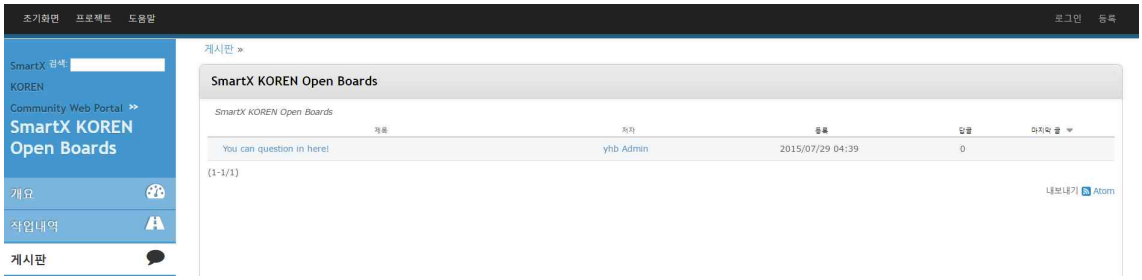

그림 17 기존 redmine 기반 오픈 SmartX Platform의 사용자 커뮤니티

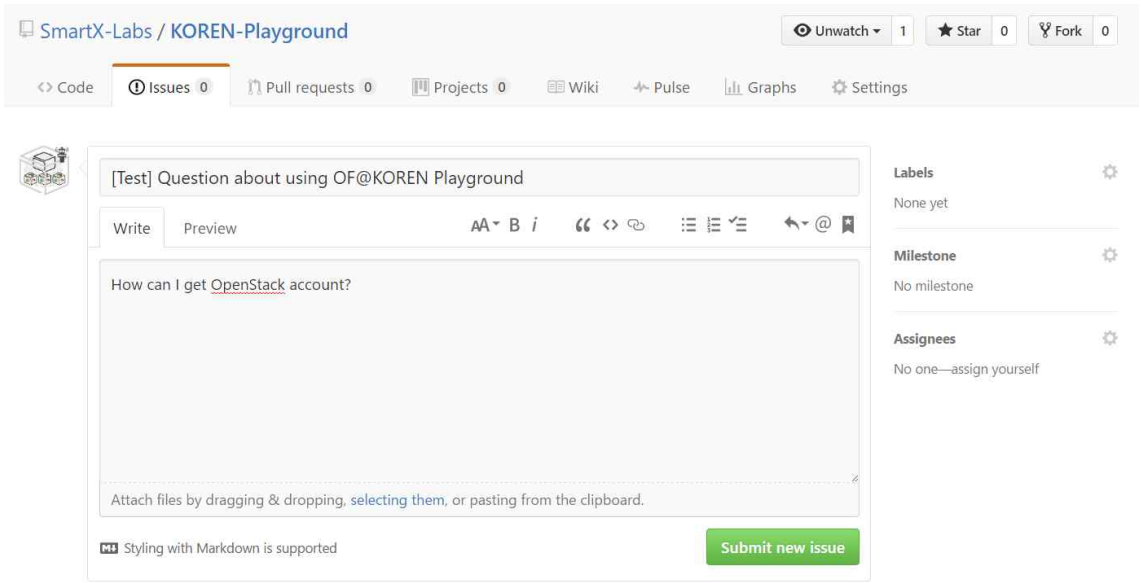

그림 18 OF@KOREN Playground 사용자 커뮤니티를 통한 질문

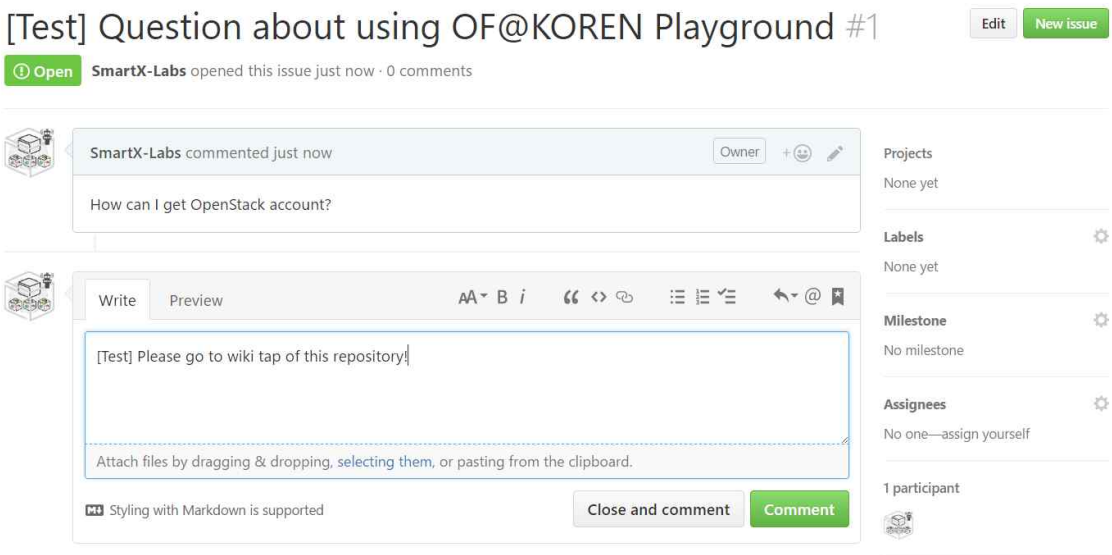

그림 19 OF@KOREN Playground 사용자 커뮤니티를 통한 답변

3. 오픈 SmartX 플랫폼을 통한 OF@KOREN Playground 가시성 지원

3.1. OF@KOREN 가시성 지원을 위한 Open API 정의

o API란 Application Programming Interface의 약자로, 프로그래밍에서 사용할 수 있는 라이브러리 혹은 기능들의 집합이다. 개발자들은 이미 제작되어 제공되는 API를 통해 빠르고, 정확하며, 간편하게 소프트웨어를 제작하거나 서비스를 구축할 수 있다.

o 일반적인 API는 특정 개발환경 혹은 권한이 주어지는 상황에서만 호출(call)하여 기능을 제공받을 수 있는데, 제한적인 부분도 적지 않다.

o 이에 Open API는 포털 사이트 또는 서비스의 개방성을 높이기 위한 기술적 기반의 개방성 응용 프로그램 인터페이스라고 할 수 있다. 대표적인 예시로는 구글, 네이버, 카카오에서 제공하는 지도 API가 있으며, 이 밖에도 다양한 서비스가 포함하는 데이터에 접근 가능하도록 API를 개방하여 일반 사용자들도 접근할 수 있게 제공하고 있다.

o Open API의 보편적이며, 가장 기본적인 호출(call) 및 반환(return)의 형태는 웹 페이지의 URL을 활용하는 호출 방식과 그에 대한 JSON 방식의 데이터 반환이다.

o 이와 동일한 방식으로 OF@KOREN 망에서의 가시성 지원을 위한 API를 제작하여, Github KOREN Playground를 통해 사용자들에게 개방(Open) 한다.

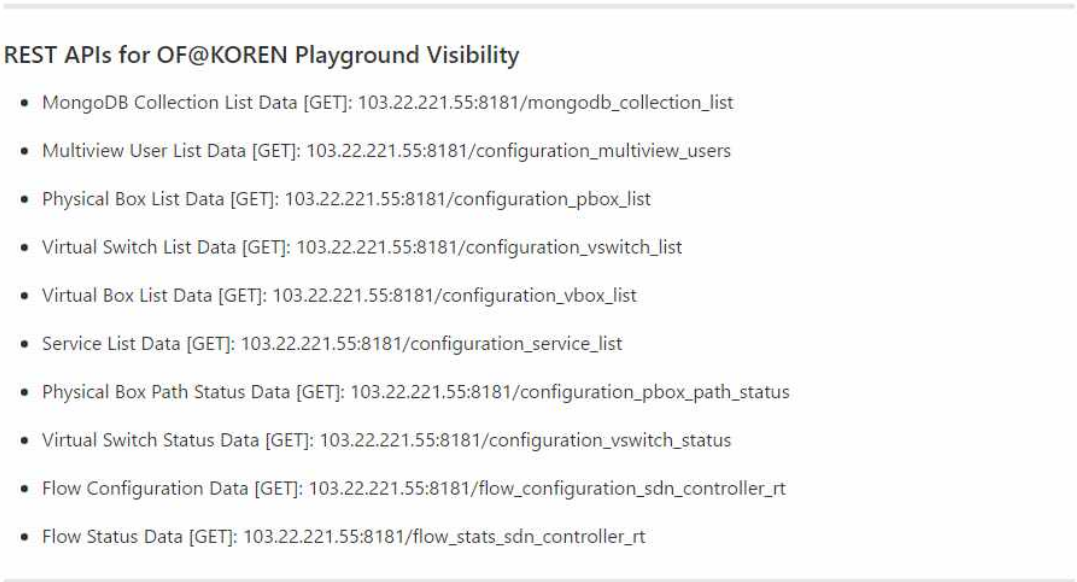

그림 20 Github/KOREN-Playground repository에서 제공하는 Open API

- MongoDB Collection List Data [GET]: MongoDB는 Relational Database와 달리 NoSQL로써 Table이 아닌 Collection 단위로 데이터를 저장한다. 이에 하나의 데이터베이스에는 다수의 Collection이 존재할 수 있다. 해당 API는 데이터베이스를 구성하는 모든 Collection에 대한 정보를 반환한다.

- Multiview User List Data [GET]: Visibility Monitoring 서비스에 접근가능한 모든 사용자 리스트를 반환한다. 현재는 아래와 같이 관리자 계정인 admin 계정과 임의로 제작한 demo 사용자 계정, 총 두 개의 사용자 목록을 반환한다.


```
[
    " {role: operator",
    " _id: {$oid: 5780d14682a7670855f447b0}",
    " password: admin",
    " username: admin}",
    " {role: developer",
    " _id: {$oid: 5780d17982a7670855f447b1}",
    " password: demo",
    " username: demo} "
]
```


그림 21 Multiview User List Data API 반환값

- Physical Box List Data [GET]: Visibility Monitoring 서비스에서 감시하고 있는 모든 물리적인 장비(box)들의 목록을 반환한다.

- Virtual Switch List Data [GET]: Visibility Monitoring 서비스에서 감시하고 있는 모든 가상적인 스위치들의 목록을 반환한다.

- Virtual Box List Data [GET]: Visibility Monitoring 서비스에서 감시하고 있는 모든 가상적인 장비(box)들의 목록을 반환한다.


```
[
    " {bridge: brcap",
    " topologyorder: 1",
    " _id: {$oid: 5780b30482a7670855f44787}",
    " type: B++}",
    " {bridge: brdev",
    " topologyorder: 2",
    " _id: {$oid: 5780b35f82a7670855f44788}",
    " type: B++}",
    " {bridge: brvlan",
    " topologyorder: 3",
    " _id: {$oid: 5780b35f82a7670855f44789}",
    " type: B++}",
    " {bridge: br-ex",
    " topologyorder: 3",
    " _id: {$oid: 5780b35f82a7670855f4478a}",
    " type: B++}",
    " {bridge: br-int",
    " topologyorder: 4",
    " _id: {$oid: 5780b35f82a7670855f4478b}",
    " type: B++} "
]
```


그림 22 Virtual Box List Data API 반환값

- Service List Data [GET]: Visibility Monitoring 서비스 위에서 구동중인 또 다른 서비스 목록들을 반환한다. 추가적인 서비스는 아직 구현중에 있어서 해당 API 호출에 대한 반환 데이터는 공백이다.

- Physical Box Path Status Data [GET]: Visibility Monitoring 서비스에서 감시하는 모든 물리적인 장비들 사이의 연결된 상태를 확인한다.

- Virtual Switch Status Data [GET]: Visibility Monitoring 서비스에서 감시하는 모든 가상적인 스위치의 상태를 반환한다.

- Flow Configuration Data [GET]: Visibility Monitoring 서비스에서 감시하는 모든 Flow들의 설정 상태를 반환한다.

- Flow Status Data [GET]: Visibility Monitoring 서비스에서 감시하는 모든 Flow들의 현재 상태를 반환한다.

o KOREN Playground에서 제공 중인 Open API의 종류는 총 10 가지로, 모두 사용자의 API 호출에 대한 응답으로 특정 데이터를 반환한다. 각 기능은 위에서 명세한 바와 동일하며, 아래의 표와 같이 정리할 수 있다.

표 1 KOREN Playground Open APIs

Name	Open API
MongoDB Collection List Data [GET]	103.22.221.55:8181/mongodb_collection_list
Multi-view User List Data [GET]	103.22.221.55:8181/configuration_multiview_users
Physical Box List Data [GET]	103.22.221.55:8181/configuration_pbox_list
Virtual Switch List Data [GET]	103.22.221.55:8181/configuration_vswitch_list
Virtual Box List Data [GET]	103.22.221.55:8181/configuration_vbox_list
Service List Data [GET]	103.22.221.55:8181/configuration_service_list
Physical Box Path Status Data [GET]	103.22.221.55:8181/configuration_pbox_path_status
Virtual Switch Status Data [GET]	103.22.221.55:8181/configuration_vswitch_status
Flow Configuration Data [GET]	103.22.221.55:8181/flow_configuration_sdn_controller_rt
Flow Status Data [GET]	103.22.221.55:8181/flow_stats_sdn_controller_rt

3.2. OF@KOREN Playground 가시성 지원

o 개선된 오픈 SmartX 플랫폼에서 KOREN 망에 대한 가시성을 지원한다는 점이 가장 큰 특징이다. 가시성 지원을 위해서 Java 기반의 visibility 제공 응용 프로그램, visibility 및 resource 정보 저장을 위한 NoSQL 기반의 데이터베이스 MongoDB, 그리고 저장된 정보를 사용자들에게 공개하기 위한 Open API를 제작

했다.

o 사용자 및 관리자에게 KOREN 망에 대한 직관적인 정보 전달을 위해서, Java 기반의 visibility 제공을 위한 응용 프로그램을 제작하여, 실시간으로 각 사이트에 위치한 Type C/S 박스의 네트워킹 상태 및 자원 정보를 모니터링 할 수 있다.

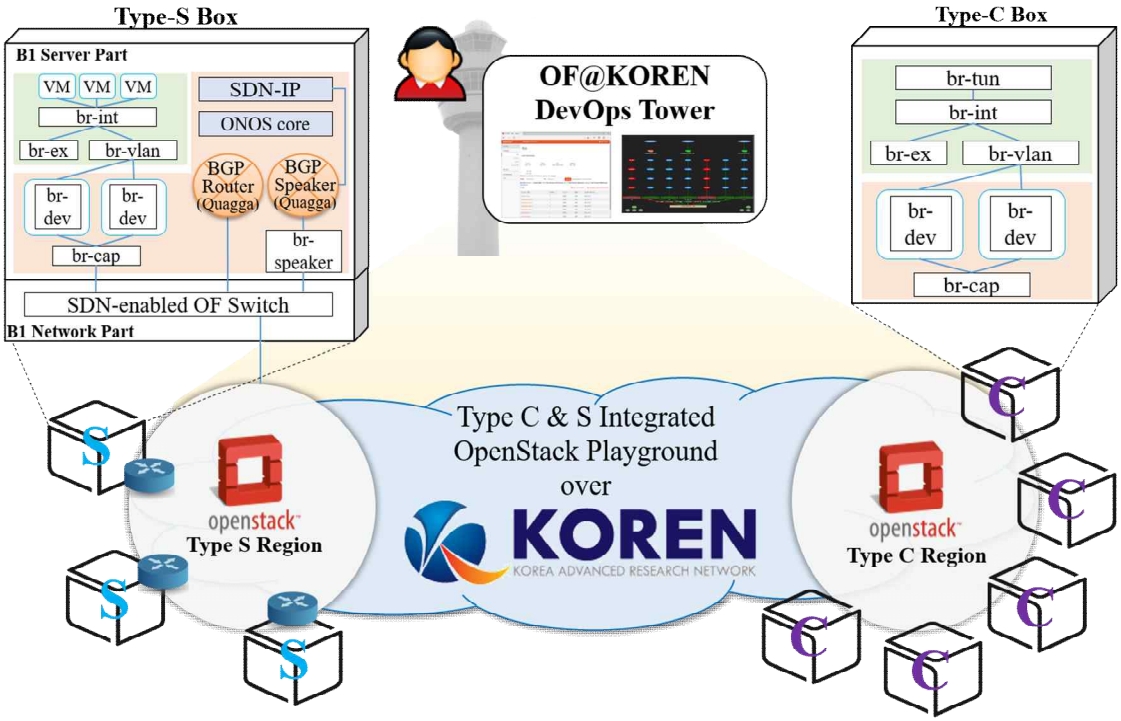

그림 23 OF@KOREN Playground 가시성 지원을 위한 응용프로그램 구조

o Visibility 제공을 위한 응용 프로그램을 통해 각 사이트에 위치한 박스들의 정보를 NoSQL 방식의 데이터베이스인 MongoDB에 저장한다. 저장된 정보를 웹 페이지 화면을 통해서 제공함으로써, 장소나 시간에 구애받지 않고 웹 브라우저를 통해 모니터링 내용을 확인할 수 있다.

o Visibility 정보에 접근하는 계정 정보에 따라서 다른 정보를 노출할 수 있도록, 웹 브라우저를 통해 해당 페이지에 접근할 경우 접속 정보를 요청하는 로그인 화면을 출력한다.

o 현재까지는 관리자 계정인 admin 계정과 실습 및 실험을 위해 생성한 임시 계정인 demo 계정으로만 접근이 가능하다. 접근 가능한 사용자 계정들에 대한 정보는 위에서 언급했던, Multiview User List Data API 호출을 통해 알 수 있다.

그림 24 Java 기반의 visibility 응용 프로그램의 웹 페이지 화면 1 – 로그인 화면

o 접근 가능한 사용자 계정을 입력하여 서비스에 접속하면, 아래와 같은 화면을 확인할 수 있다. 물리적인 장치, 가상적인 장치/스위치 등의 상태에 대한 정보를 네 가지 색상으로 표현하여 확인할 수 있다.

- 녹색(green): 물리적인 장치 상태 및 내부의 가상 스위치 OvS 기반의 가상 브릿지들이 모두 정상을 의미할 경우를 표시한다.

- 황색(yellow): 물리적인 장치 상태 정상 및 내부의 가상 스위치 OvS 기반의 가상 브릿지는 존재하지만 OvS 프로세스가 동작하지 않는 경우를 표시한다.

- 적색(red): 물리적인 장치 상태 정상, 그러나 내부의 가상 스위치 OvS 기반의 가상 브릿지 추가에 문제가 발생하여 존재하지 않는 경우를 표시한다.

- 회색(grey): 물리적인 장치 내부의 OpenStack Neutron 내부에 위치하는 가상 스위치 OvS에 문제가 발생한 경우를 표시한다.

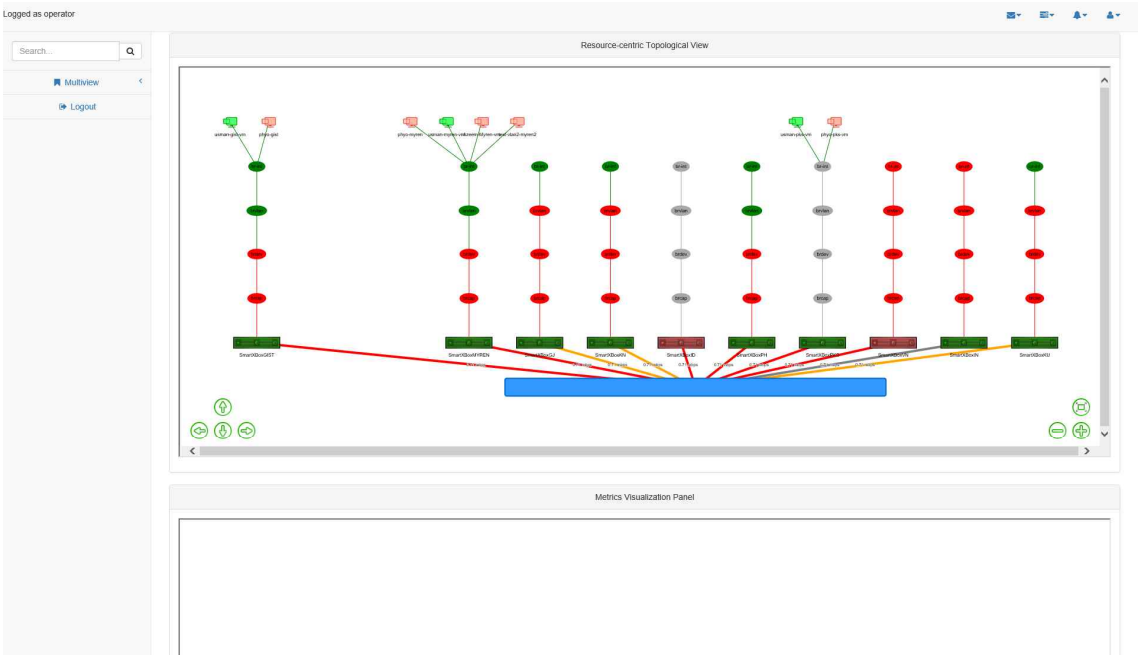

그림 25 Java 기반의 visibility 응용 프로그램의 웹 페이지 화면 2 – 모니터링 화면

3.3. KOREN Playground 내 Open API를 통한 visibility 정보 접근

o Java 기반의 visibility 응용프로그램을 통해 수집된 정보는 MongoDB에 꾸준히 저장되고 갱신된다.

o 저장된 정보 중 일부를 일반인 혹은 외부 사용자에게 공개하기 위해 Restful API 형태로 공개한다. 오픈 SmartX 플랫폼 및 KOREN Playground에 Restful API를 Open API로 제공함으로써 외부 사용자들도 모니터링 데이터를 활용할 수 있도록 한다.

o Open API 제작을 위해서 Python 기반의 API 서버를 제작한다. API 서버 제작을 위해서 Python framework인 django를 활용한다. Python 기반의 API 서버가 직접 MongoDB에 저장 되어있는 visibility 데이터에 접근한다.

o 외부 사용자가 Open API를 통해 visibility 정보에 접근하게 되는 과정은 아래와 같다.

(1) 오픈 SmartX 플랫폼, KOREN Playground에 접근한다.
(2) Software directory에 저장되어있는 Open API를 호출한다.
(3) 사용자가 Open API를 호출하면, API 서버는 MongoDB에 접근하여, 해당 정보를 JSON 타입의 데이터로 포맷한다.
(4) 사용자의 웹 페이지를 통해 JSON 타입의 visibility 데이터를 반환한다.

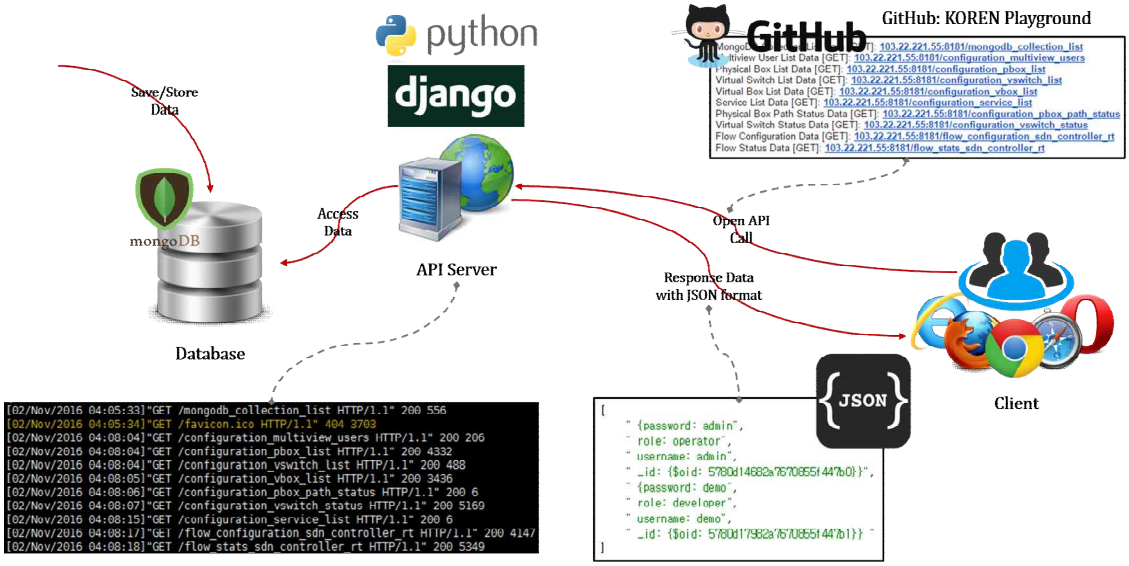

그림 26 Open API 제작을 위한 Python 기반의 API 서버 구조도

o Open API 제작을 위해서 사용한 django는 Python 기반의 Web Framework로 manage.py, url.py 및 view.py 파일의 수정을 통해 API 서버를 구현했다.

o mongoDB에 저장된 visibility 데이터에 접근하여, 사용자에게 반환하기 위한 프로세스는 view.py에 정의된 함수를 통해서 수행된다. view.py 함수에는 위에서 언급한 열 개의 API에 대한 함수들이 각각 구현되어 있다. 모든 함수는 JSON 형식의 데이터를 반환하며, 사용자의 request를 인수로 갖는다.

표 2 KOREN Playground Open APIs와 view.py 내부에 정의된 함수

Open API	Definition in view.py
MongoDB Collection List Data [GET] →	← def mongodb_collection_list (request)
Multi-view User List Data [GET] →	← def configuration_multiview_users (request)
Physical Box List Data [GET] →	← def configuration_pbox_list (request)
Virtual Switch List Data [GET] →	← def configuration_vswitch_list (request)
Virtual Box List Data [GET] →	← def configuration_vbox_list (request)
Service List Data [GET] →	← def configuration_service_list (request)
Physical Box Path Status Data [GET] →	← def configuration_pbox_path_status (request)
Virtual Switch Status Data [GET] →	← def configuration_vswitch_status (request)
Flow Configuration Data [GET] →	← def flow_configuration_sdn_controller_rt (request)
Flow Status Data [GET] →	← def flow_stats_sdn_controller_rt (request)

```
from django.shortcuts import render
from django.http import JsonResponse
from pymongo import MongoClient
from bson import Binary, Code
from bson.json_util import dumps, loads


client = MongoClient('mongodb://127.0.0.1:27017/')
db = client.smartxdb

# Create your views here.

def mongodb_collection_list(request):
    data = db.collection_names()
    return JsonResponse(data, safe=False)


def configuration_multiview_users(request):
    data = []
    collection = db['configuration-multiview-users'].find()
    raw_data = dumps(collection).replace("", '').split(',')
    for i in raw_data:
        i = i.replace( '[', '  ).replace(']','  )
        data.append(i)
    return JsonResponse(data, safe=False)


def configuration_pbox_list(request):
    data = []
    collection = db['configuration-pbox-list'].find()
    raw_data = dumps(collection).replace("", '').split(',')
    for i in raw_data:
        i = i.replace( '[', '  ).replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def configuration_vswitch_list(request):
    data = []
    collection = db['configuration-vswitch-list'].find()
    raw_data = dumps(collection).replace("", '').split(',')
    for i in raw_data:
        i = i.replace( '[', '  ).replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def configuration_vbox_list(request):
    data = []
    collection = db['configuration-vbox-list'].find()
    raw_data = dumps(collection).replace("", '').split(',')
    for i in raw_data:
        i = i.replace( '[', '  ).replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def configuration_service_list(request):
    data = []
    collection = db['configuration-service-list'].find()
    raw_data = dumps(collection).replace("", '').split(',')
search hit TOP, continuing at BOTTOM
```

그림 27 django Web Framework 내부의 view.py 소스코드

```
    collection = db['configuration-service-list'].find()
    raw_data = dumps(collection).replace('"', '').split(',')
    for i in raw_data:
        i = i.replace('[', '  ').replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def configuration_pbox_path_status(request):
    data = []
    collection = db['configuration-pbox-path-status'].find()
    raw_data = dumps(collection).replace('"', '').split(',')
    for i in raw_data:
        i = i.replace('[', '  ').replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def configuration_vswitch_status(request):
    data = []
    collection = db['configuration-vswitch-status'].find()
    raw_data = dumps(collection).replace('"', '').split(',')
    for i in raw_data:
        i = i.replace('[', '  ').replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def flow_configuration_sdn_controller_rt(request):
    data = []
    collection = db['flow-configuration-sdn-controller-rt'].find()
    raw_data = dumps(collection).replace('"', '').split(',')
    for i in raw_data:
        i = i.replace('[', '  ').replace(']', ' ')
        data.append(i)
    return JsonResponse(data, safe=False)


def flow_stats_sdn_controller_rt(request):
    data = []
    collection = db['flow-stats-sdn-controller-rt'].find()
    raw_data = dumps(collection).replace('"', '').split(',')
    for i in raw_data:
        i = i.replace('[', '  ').replace(']', ' ').replace('\\','')
        data.append(i)
    return JsonResponse(data, safe=False)
```

그림 28 django Web Framework 내부의 view.py 소스코드

o view.py에 정의된 함수가 mongoDB로부터 가져온 데이터들은 url.py를 통해서 각각의 웹 url과 매핑되고, 사용자의 Open API 호출에 대해서 웹 브라우저 상에서 응답하게 된다.

표 3 KOREN Playground Open APIs와 view.py 내부에 정의된 함수

Web URL in url.py	Definition in view.py
103.22.221.55:8181/mongodb_collection_list →	← def mongodb_collection_list (request)
103.22.221.55:8181/configuration_multiview_users →	← def configuration_multiview_users (request)
103.22.221.55:8181/configuration_pbox_list →	← def configuration_pbox_list (request)
103.22.221.55:8181/configuration_vswitch_list →	← def configuration_vswitch_list (request)
103.22.221.55:8181/configuration_vbox_list →	← def configuration_vbox_list (request)
103.22.221.55:8181/configuration_service_list →	← def configuration_service_list (request)
103.22.221.55:8181/configuration_pbox_path_status →	← def configuration_pbox_path_status (request)
103.22.221.55:8181/configuration_vswitch_status →	← def configuration_vswitch_status (request)
103.22.221.55:8181/flow_configuration_sdn_controller_rt →	← def flow_configuration_sdn_controller_rt (request)
103.22.221.55:8181/flow_stats_sdn_controller_rt →	← def flow_stats_sdn_controller_rt (request)

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^mongodb_collection_list$', views.mongodb_collection_list),
    url(r'^configuration_multiview_users$', views.configuration_multiview_users),
    url(r'^configuration_pbox_list$', views.configuration_pbox_list),
    url(r'^configuration_vswitch_list$', views.configuration_vswitch_list),
    url(r'^configuration_vbox_list$', views.configuration_vbox_list),
    url(r'^configuration_service_list$', views.configuration_service_list),
    url(r'^configuration_pbox_path_status$', views.configuration_pbox_path_status),
    url(r'^configuration_vswitch_status$', views.configuration_vswitch_status),
    url(r'^flow_configuration_sdn_controller_rt$', views.flow_configuration_sdn_controller_rt),
    url(r'^flow_stats_sdn_controller_rt$', views.flow_stats_sdn_controller_rt),
]
```

그림 29 django Web Framework 내부의 url.py 소스코드

o manage.py를 통해 작성한 view.py와 url.py 소스코드의 내용이 동작하도록 API 서버를 구동시킨다. API 서버로는 8181 포트를 사용한다.

```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "jsonmid.settings")

    from django.core.management import execute_from_command_line

    execute_from_command_line(sys.argv)
```

그림 30 django Web Framework 내부의 manage.py 소스코드

o Open API 서버가 구동 중에 사용자가 Open API를 호출하면, 웹 서버의 CLI 환경에는 사용자의 request 및 이에 상응하는 response에 대한 로그가 작성된다.

```
November 02, 2016 - 04:05:20
Django version 1.8, using settings 'jsonmid.settings'
Starting development server at http://0.0.0.0:8181/
Quit the server with CONTROL-C.
[02/Nov/2016 04:05:33]"GET /mongodb_collection_list HTTP/1.1" 200 556
[02/Nov/2016 04:05:34]"GET /favicon.ico HTTP/1.1" 404 3703
[02/Nov/2016 04:08:04]"GET /configuration_multiview_users HTTP/1.1" 200 206
[02/Nov/2016 04:08:04]"GET /configuration_pbox_list HTTP/1.1" 200 4332
[02/Nov/2016 04:08:04]"GET /configuration_vswitch_list HTTP/1.1" 200 488
[02/Nov/2016 04:08:05]"GET /configuration_vbox_list HTTP/1.1" 200 3436
[02/Nov/2016 04:08:06]"GET /configuration_pbox_path_status HTTP/1.1" 200 6
[02/Nov/2016 04:08:07]"GET /configuration_vswitch_status HTTP/1.1" 200 5169
[02/Nov/2016 04:08:15]"GET /configuration_service_list HTTP/1.1" 200 6
[02/Nov/2016 04:08:17]"GET /flow_configuration_sdn_controller_rt HTTP/1.1" 200 4147
[02/Nov/2016 04:08:18]"GET /flow_stats_sdn_controller_rt HTTP/1.1" 200 5349
^T^Gg[03/Nov/2016 04:09:31]"GET /flow_stats_sdn_controller_rt HTTP/1.1" 200 5349
[03/Nov/2016 04:09:31]"GET /favicon.ico HTTP/1.1" 404 3703
```

그림 31 django 기반의 Open API 서버 구동 화면

o 최종적으로 Open API를 호출한 사용자의 웹 페이지에 mongoDB에 저장된 visibility 데이터를 JSON 타입으로 포맷된 형태의 데이터를 반환한다. 아래의 화면은 열 개의 Open API 중에서 Flow Status Data API 호출에 대한 반환값이다.

[
" {packetCount: 356397",
" matchField: matchField: {type:IN_PORT",
"value:OF|1@OF|33:33:33:33:33:33:33:11} ",
" controllerIP: 103.22.221.152",
" timestamp: {$date: 1478146141291}",
" durationSeconds: 663555",
" boxID: SmartXBoxGIST",
" actions: actions: {port:{node:{id:33:33:33:33:33:33:33:11",
"type:OF}",
"id:4",
"type:OF}",
"type:OUTPUT} ",
" node: 33:33:33:33:33:33:33:11",
" byteCount: 57969529",
" _id: {$oid: 581ab85dd95a8e5eee451250}}",
" {packetCount: 0",
" matchField: matchField: {type:IN_PORT",
"value:OF|5@OF|33:33:33:33:33:33:33:11} ",
" controllerIP: 103.22.221.152",
" timestamp: {$date: 1478146141291}",
" durationSeconds: 663555",
" boxID: SmartXBoxGIST",
" actions: actions: {port:{node:{id:33:33:33:33:33:33:33:11",
"type:OF}",
"id:2",
"type:OF}",
"type:OUTPUT} ",
" node: 33:33:33:33:33:33:33:11",
" byteCount: 0",
" _id: {$oid: 581ab85dd95a8e5eee451251}}",
" {packetCount: 145086",
" matchField: matchField: {type:IN_PORT",
"value:OF|2@OF|33:33:33:33:33:33:33:11} ",
" controllerIP: 103.22.221.152",
" timestamp: {$date: 1478146141291}",
" durationSeconds: 663555",
" boxID: SmartXBoxGIST",
" actions: actions: {port:{node:{id:33:33:33:33:33:33:33:11",
"type:OF}",
"id:5",
"type:OF}",
"type:OUTPUT} ",
" node: 33:33:33:33:33:33:33:11",
" byteCount: 22796144",
" _id: {$oid: 581ab85dd95a8e5eee451252}}",
" {packetCount: 61839913",
" matchField: matchField: {type:IN_PORT",
"value:OF|6@OF|33:33:33:33:33:33:33:11} ",
" controllerIP: 103.22.221.152",
" timestamp: {$date: 1478146141291}",
" durationSeconds: 663555",
" boxID: SmartXBoxGIST",
" actions: actions: {port:{node:{id:33:33:33:33:33:33:33:11",
[4]

그림 32 Open API 호출에 대한 JSON 방식의 데이터 반환 화면

4. 오픈 SmartX Platform의 개선 결론 및 기대 효과

4.1. 결론 및 기대 효과

o 본 문서에서는 KOREN 네트워크 상에서 운용되고 있는 OF@KOREN Playground 사용자들을 위한 오픈 SmartX Platform의 개선에 대한 내용을 다루었다. GitHub 기반으로 개선된 형태의 오픈 SmartX Platform 커뮤니티 사이트의 접근성 향상 및 OF@KOREN Playground 관련 자료 및 소스 코드의 제공을 통해 향상된 사용자 지원 기능을 기대할 수 있다.

o 또한 OF@KOREN Playground의 상태 정보를 확인할 수 있는 Open API를 개발하여 제공함으로써 OF@KOREN Playground의 사용자들이 모니터링 정보를 활용한 서비스를 구현하여 적극적인 OF@KOREN Playground의 활용이 가능할 것으로 기대된다.

References

[1] DevOps, http://en.wikipedia.org/wiki/DevOps

[2] IPMI, http://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface

[3] J. Kim et. al., "OF@TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites," in *Proc. APAN –Networking Research Workshop*, Aug, 2013.

[4] Devstack, http://docs.openstack.org/developer/devstack/

[5] Ubuntu MAAS, http://maas.ubuntu.com

[6] OpenStack, http://openstack.org

[7] J. Shin and J.Kim, "Functionality verification of automated remote installation of converged resource boxes for distributed cloud," *in Proc. 27th Korean Signal Processing Conference (KSPC)*, Seoul, Korea, Sep. 2014

[8] PXE, http://en.wikipedia.org/wiki/Preboot_Execution_Environment

SmartX 기술 문서

작성기관: 광주과학기술원
작성년월: 2016/12