



# Spectra Guide v2.6.1

2023.06



# 목차

---

- Spectra Web Guide
- Spectra Open API Guide
- Jenkins Plugin Guide
- Github Action Guide



# Spectra Web Guide

2023.06



## 0. 프로젝트 구조

Spectra Web의 경우 다음과 같은 프로젝트 구조를 지원합니다.

(1) 하나의 sol 파일

a.sol

(2) 여러 sol파일의 압축파일 (.zip 확장자만 지원)

sample.zip  
└ a.sol  
└ b.sol  
└ c.sol

(3) hardhat, truffle, brownie의 표준 구조를 취하는 압축파일  
(O)는 optional을 의미합니다.)

sample.zip  
└ contracts/  
└ (O) node\_modules/  
└ (O) package.json  
└ (O) hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml

# 1. Introduction

<https://www.spectra-space.io> 로 접속하면 Spectra 메인 페이지로 이동할 수 있습니다.

Smart Contract 파일(.sol)을 직접 업로드하여 Static Audit을 수행할 수 있고 이에 대한 Report를 확인할 수 있습니다.



## 1. 로그인 버튼

Spectra 유저 로그인 페이지로 이동합니다.  
로그인 상태인 경우 로그아웃(logout)으로 표시됩니다.

## 2. 한/영 전환 버튼

언어를 한글 또는 영어로 변경합니다.

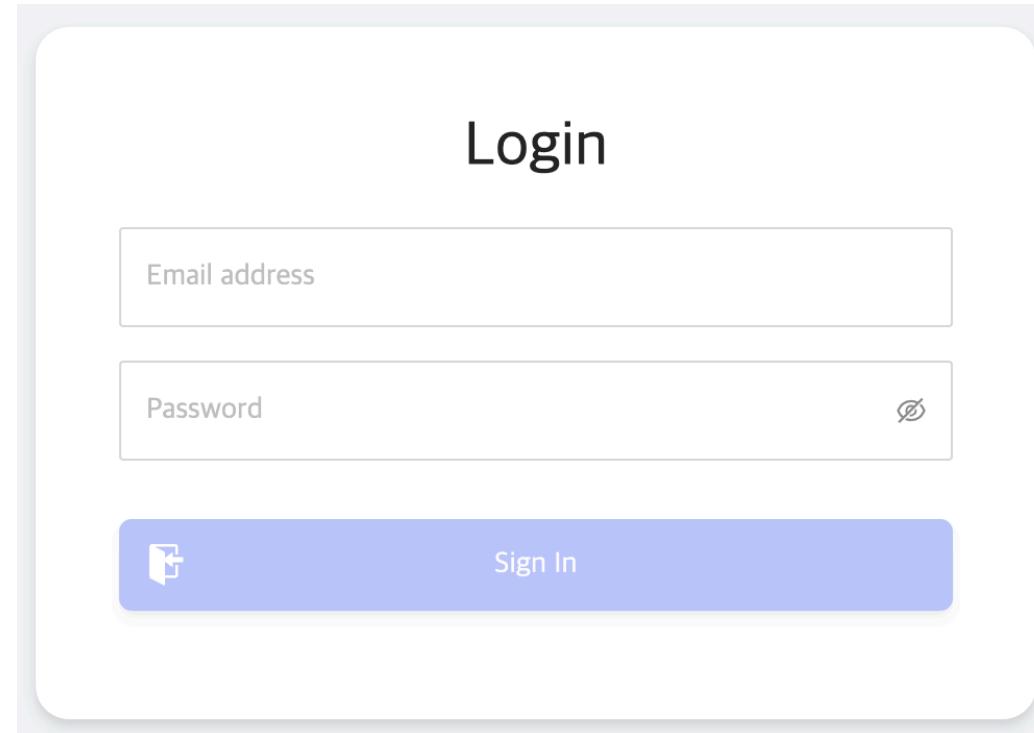
## 3. Smart Contract Audit 메뉴

Smart Contract Audit 수행을 위한 페이지로 이동합니다.

## 2. Login

관리자로부터 발급받은 ID(이메일주소) 및 Password로 로그인을 할 수 있습니다.

**신규 계정 발급은 반드시 Admin 관리자로부터 요청해주시기 바랍니다.**

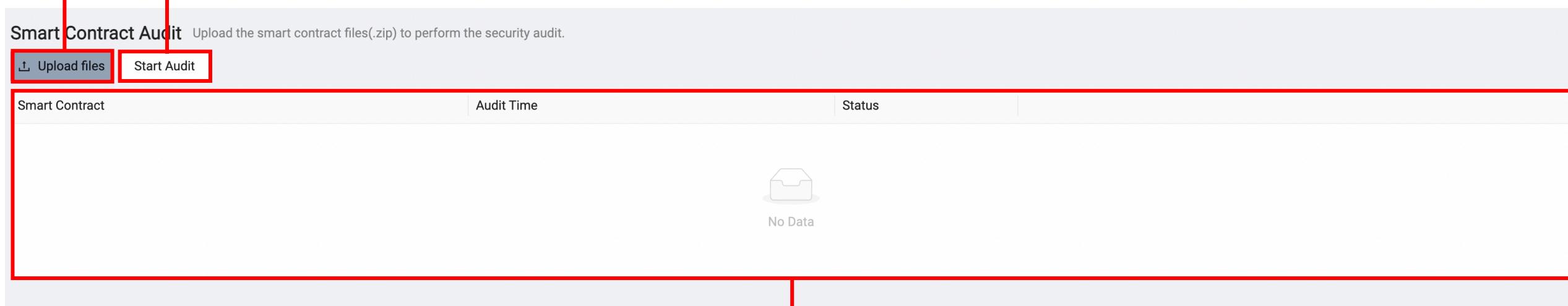


### 3. Smart Contract Audit

상단의 Smart Contract Audit 메뉴를 클릭하면 Audit을 수행하거나, Audit 히스토리를 조회할 수 있습니다.

**파일 업로드 버튼 :** Audit을 수행할 Smart Contract 파일(.sol 또는 컴파일 가능한 프로젝트 압축파일)을 업로드

**Audit 수행 버튼 :** 업로드한 파일을 기준으로 Audit 수행을 시작



The screenshot shows the 'Smart Contract Audit' section of a web application. At the top, there is a placeholder text: 'Upload the smart contract files(.zip) to perform the security audit.' Below it are two buttons: 'Upload files' (blue background) and 'Start Audit' (white background with a red border). A red double-headed vertical arrow points between the 'Upload files' button and the 'Start Audit' button. Below these buttons is a table with three columns: 'Smart Contract', 'Audit Time', and 'Status'. The 'Status' column contains a small envelope icon and the text 'No Data'. A red double-headed horizontal arrow points between the 'Audit Time' and 'Status' columns. A red arrow points downwards from the bottom of the table towards the 'Audit History' section below.

| Smart Contract | Audit Time | Status  |
|----------------|------------|---|
|                |            |  No Data |

**Audit History :** 현재 로그인한 계정의 Audit 히스토리가 보여지는 영역

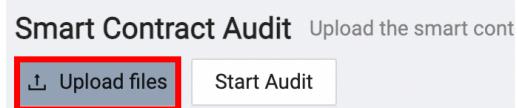
### 3. Smart Contract Audit (step 1. File upload)

Audit 을 수행할 파일을 업로드합니다.

(업로드 규칙1) 이미 업로드한 파일을 중복해서 업로드할 수 없음

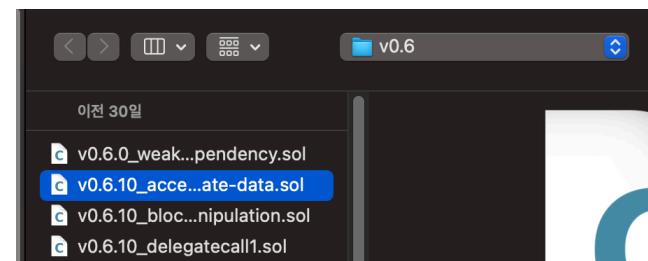
(업로드 규칙2) Audit을 수행한 파일인 경우, 다시 업로드할 수 있음

(1) 업로드버튼 클릭

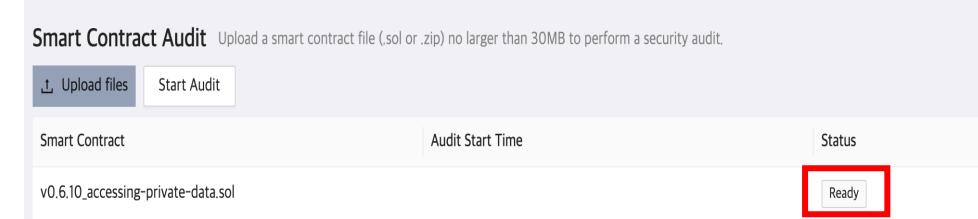


(2) 파일 선택

(파일 제한) .sol 또는 컴파일 가능한 프로젝트의 .zip파일  
(파일 개수) 1개 이상



(3) 업로드된 파일 확인 (Status : Ready)

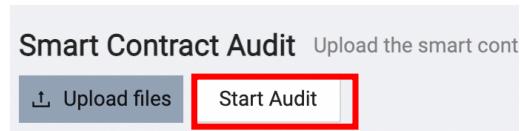


### 3. Smart Contract Audit (step 2. Audit Start)

Audit 을 수행합니다.

Audit 성공인 경우, Smart Contract 항목이 링크 표시됩니다. (컴파일 오류 발생 시, Audit 결과가 Fail로 나타날 수 있습니다.)

(1) Start Audit 버튼 클릭



(2) Audit 시작 및 완료 (10초 이내)

| Smart Contract                     | Audit Start Time    | Status  |
|------------------------------------|---------------------|---------|
| v0.6.10_accessing-private-data.sol | 2023-05-04 18:11:49 | Success |

### 3. Smart Contract Audit (step 3. View Audit Report)

Audit 결과가 성공인 경우, Audit 보고서를 확인할 수 있습니다.

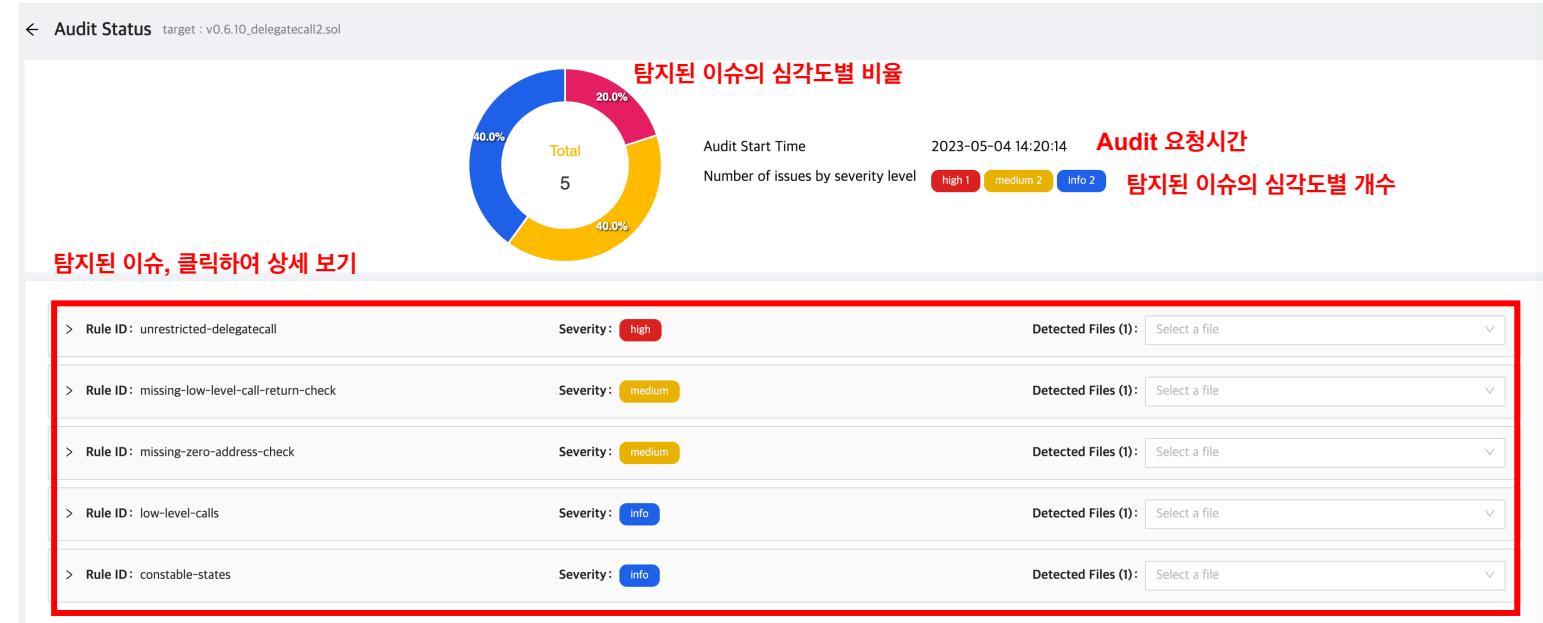
(1) Smart Contract 링크 클릭



(2) Audit Report 화면으로 이동

Smart Contract

v0.6.10\_accessing-private-data.sol



### 3. Smart Contract Audit (step 3. View Audit Report)

Rule ID: unrestricted-delegatecall  
Lines을 클릭하여 탐지된 영역으로 바로가기

Severity: high

Detected Files (1): v0.6.10\_delegatecall2.sol

Lines  
41 - 43

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) public {
    lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}

// contract Attack {
//     // Make sure the storage layout is the same as HackMe
//     // This will allow us to correctly update the state variables
//     address public lib;
//     address public owner;
//     uint public someNumber;
//
//     HackMe public hackMe;
//
//     constructor(HackMe hackMe) public {

```

탐지된 영역은 하이라이트 처리

#### Description 탐지된 취약점에 대한 설명

Calling into untrusted contracts is very dangerous, as the code at the target address can change any storage values of the caller and has full control over the caller's balance.

#### Countermeasure 탐지된 취약점에 대한 recommendation

1. Control what is executed with delegatecall  
: Use permissions or authentication or some other form of control for specifying or changing what functions and contracts are executed with delegatecall.
2. Only call delegatecall on addresses that have code  
: check address  
==>  
function isContract (address account) internal view returns (bool) {  
 uint size;  
 assembly { size := extcodesize(account) }  
 return size > 0;  
}

### 3. Smart Contract Audit (step 4. Delete audit history)

Audit 내역을 삭제할 수 있습니다

삭제 버튼 클릭

| Smart Contract                     | Audit Start Time    | Status  |                                   |
|------------------------------------|---------------------|---------|-----------------------------------|
| v0.6.10_accessing-private-data.sol | 2023-05-04 18:11:49 | Success | <input type="button" value="삭제"/> |
| v0.6.10_delegatecall2.sol          | 2023-05-04 14:20:14 | Success | <input type="button" value="삭제"/> |
| v0.6.10_front-running.sol          | 2023-05-04 11:32:07 | Success | <input type="button" value="삭제"/> |



# Spectra Open API Guide

2023.06



## 0. 프로젝트 구조

Spectra Open API의 경우 다음과 같은 프로젝트 구조를 지원합니다.

(1) 하나의 sol 파일

a.sol

(2) 여러 sol파일의 압축파일 (.zip 확장자만 지원)

sample.zip  
└ a.sol  
└ b.sol  
└ c.sol

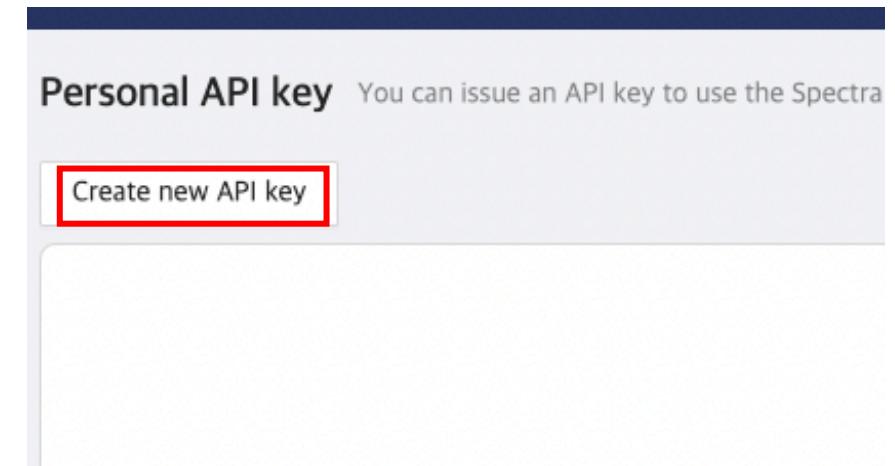
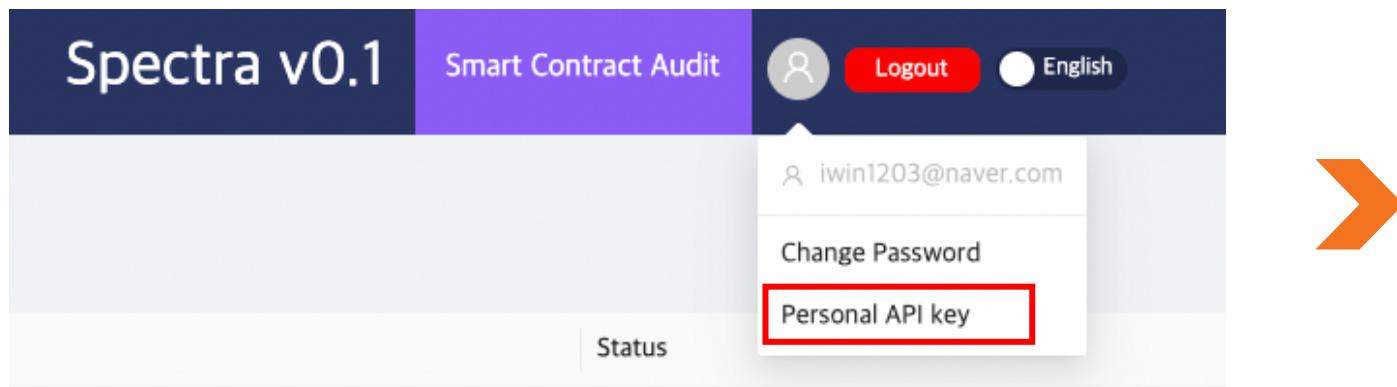
(3) hardhat, truffle, brownie의 표준 구조를 취하는 압축파일  
(O)는 optional을 의미합니다.)

sample.zip  
└ contracts/  
└ (O) node\_modules/  
└ (O) package.json  
└ (O) hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml

# 1. Setup

Spectra는 CI/CD 툴 Jenkins의 Open API를 제공합니다.

Spectra Web에서 프로필 아이콘 클릭 > Personal API key > Create new API key를 클릭하여 api key 발급 창으로 이동합니다.



# 1. Setup

Spectra는 CI/CD 툴 Jenkins의 Open API를 제공합니다.

Expiration과 role을 선택하여 API key를 발급받습니다.

Create New API key

Key 만료 기한

\* Expiration: 90 days Select date

\* roles:  request audit  view audit

권한: audit 요청 권한 / 결과 조회

Cancel OK

Personal API key You can issue an API key to use the Spectra service.

Create new API key

**발급된 Key**

5f127d95-244e-42d7-933c-5b3b1cc7a747

roles : view-audit,request-audit created : 2023-05-16 expiration date : 2023-07-15

## 2. API Specification

방식: http/REST

엔드포인트: [POST] <https://api.spectra-space.io/v1/request>

### Request Field:

header

X-Apikey: <api key>

Content-Type: multipart/form-data

| KEY          | VALUE                |
|--------------|----------------------|
| X-Apikey     | <api key>            |
| Content-Type | multipart/form-data; |

body (multipart/form-data)

file : <file to audit>

| KEY  | VALUE        |
|------|--------------|
| file | Select Files |

## 2. API Specification

---

### Response

401 Unauthorized: API Key가 올바르지 않거나 만료된 경우

404 Bad Request: Request Body가 잘못된 경우

200 Ok: 분석이 성공적으로 진행된 경우.

### Result

‘200 Ok’가 리턴되었다면, Spectra Web에서 설정한 이메일로 링크가 전송됩니다. 해당 링크를 클릭하면 audit 결과를 확인할 수 있습니다.

### 3. 결과 확인하기

빌드에 성공한 경우, Spectra Web에서 설정한 이메일로 결과 링크가 주어집니다. 해당 링크를 클릭하여 결과를 볼 수 있습니다.

The screenshot shows the audit results for a target with a total of 308 issues. The distribution by severity level is as follows:

| Severity Level | Count |
|----------------|-------|
| high           | 6     |
| medium         | 6     |
| low            | 14    |
| info           | 282   |

The report lists six specific rule violations:

- Rule ID: reentrancy (Severity: high)
- Rule ID: missing-return-value-check (Severity: medium)
- Rule ID: using-shadowed-local-variable (Severity: low)
- Rule ID: low-level-calls (Severity: info)
- Rule ID: dead-code (Severity: info)
- Rule ID: unused-state-variable (Severity: info)

For each rule, there is a dropdown menu labeled "Select a file".



# Spectra Jenkins Plugin Guide

2023.06

## 0. 프로젝트 구조

Spectra Jenkins Plugin의 경우, 다음과 같은 프로젝트 구조를 지원합니다.

(1) 한 개 이상의 sol 파일

```
a.sol  
b.sol  
c.sol
```

(2) hardhat, truffle, brownie의 표준 구조

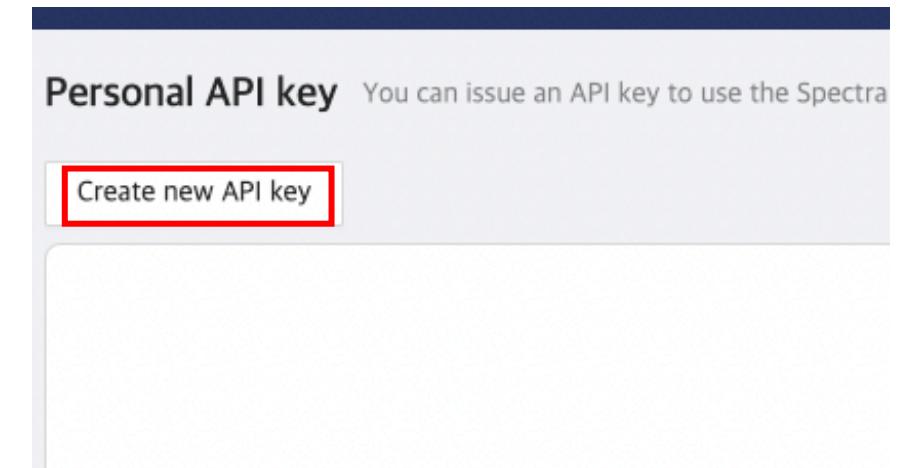
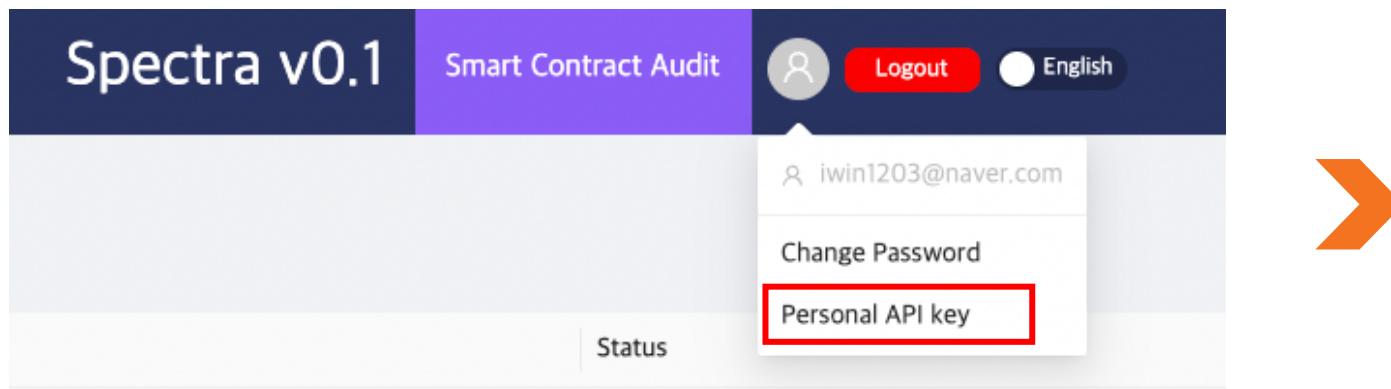
주의) 다음 중 필수는 “contracts” 경로이며, 그 외는 선택적입니다.

```
contracts/  
└ a.sol  
node_modules/  
└ @openzeppelin/  
package.json  
hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml
```

# 1. API Key Setup

Spectra는 CI/CD 툴 Jenkins의 플러그인을 제공합니다. \* 본 가이드라인은 Jenkins를 활용하고 있는 프로젝트를 대상으로 합니다.

Spectra Web에서 프로필 아이콘 클릭 > Personal API key > Create new API key를 클릭하여 api key 발급 창으로 이동합니다.



# 1. API Key Setup

Expiration과 role을 선택하여 API key를 발급받습니다.

Create New API key

Key 만료 기한

\* Expiration: 90 days Select date

\* roles:  request audit  view audit

권한: audit 요청 권한 / 결과 조회

Cancel OK

Personal API key You can issue an API key to use the Spectra service.

발급된 Key

5f127d95-244e-42d7-933c-5b3b1cc7a747

roles : view-audit,request-audit

created : 2023-05-16 expiration date : 2023-07-15

## 2. Installation

<https://github.com/spark63/Jenkins-Plugin> 레포지토리에서 spectra-plugin.hpi 파일을 다운로드합니다.

The screenshot shows the GitHub repository page for 'spark63/Jenkins-Plugin'. The 'Code' tab is selected. In the commit list, the 'spectra-plugin.hpi' file is highlighted with a red box. The commit details are as follows:

| File                   | Message                  | Time Ago       |
|------------------------|--------------------------|----------------|
| README.md              | fix: remove image crashC | 6 minutes ago  |
| spectra-plugin.hpi     | release: v1.0            | 10 minutes ago |
| spectra_guide_v2.2.pdf | feat: guide updated      | 3 minutes ago  |

The repository page also includes sections for 'About', 'Releases', and 'Packages'.

**About**  
No description, website, or topics provided.

**Releases** 1  
spectra-jenkins-plugin Latest  
1 minute ago

**Packages**  
No packages published  
Publish your first package

**Spectra Plugin For Jenkins - Build Steps / Post-build Actions / Pipeline**

- Spectra plugin for Jenkins helps you integrate Spectra's automated audit process into your project CI/CD flow.
- Check "spectra\_guide" in this repository for details.

**Set up**

## 2. Installation

Jenkins에서, Dashboard > Manage Jenkins > Plugin Manager > Advanced settings로 이동합니다.

The screenshot shows the Jenkins 'Advanced settings' configuration page. On the left, there is a sidebar with links: 'Updates', 'Available plugins', 'Installed plugins', and 'Advanced settings'. The 'Advanced settings' link is highlighted with a blue background. The main area is titled 'Advanced settings' and contains the 'HTTP Proxy Configuration' section. It includes fields for 'Server' (with a placeholder), 'Port' (with a placeholder), 'User name' (with a placeholder), 'Password' (with a placeholder), and 'No Proxy Host' (with a placeholder). At the bottom left is a 'Submit' button.

## 2. Installation

Deploy Plugin에서, 다운로드한 .hpi 파일을 선택 후 Deploy 버튼을 클릭합니다.

### Deploy Plugin

You can select a plugin file from your local system or provide a URL to install a plugin from outside the central plugin repository.

File

파일 선택

spectra-plugin.hpi

Or

URL

Deploy

### 3. Network Setup

Jenkins plugin은 다음 IP에 대한 인바운드 / 아웃바운드 허용이 필요합니다.

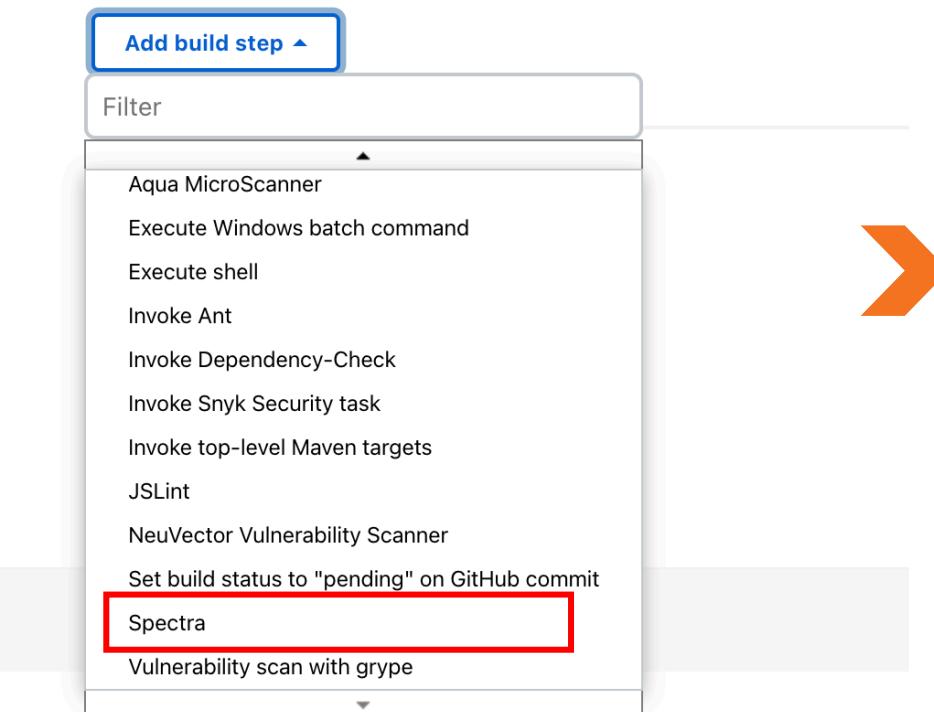
인바운드: 223.130.162.168 (포트: 유저의 젠킨스 서버 포트와 동일)

아웃바운드: 223.130.162.168 (포트: 443)

## 4. Freestyle Project에 적용하기

작성해둔 freestyle project의 configure에서, Build Steps - Spectra 플러그인을 선택합니다.

### Build Steps



### Build Steps

#### Spectra

##### API Key

Put API key obtained from Spectra website.

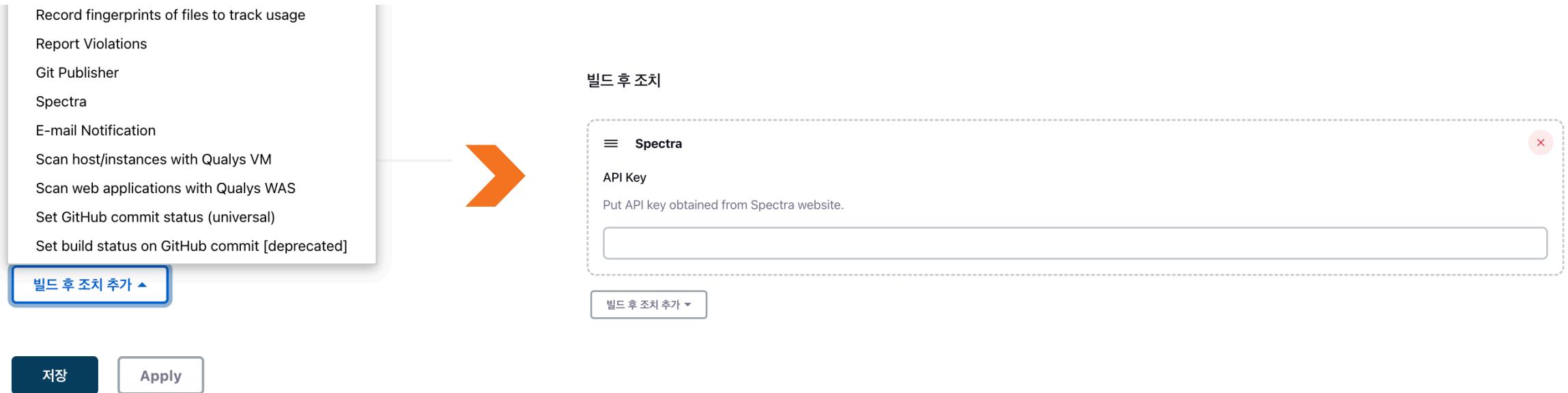
abcd123

##### Add build step ▾

발급받은 API Key를 입력합니다.

## 4. Freestyle Project에 적용하기

빌드 후 조치 (Post Build Action)에도 동일한 방식으로 Spectra 플러그인을 적용할 수 있습니다.



## 4. Freestyle Project에 적용하기

프로젝트 빌드 후 Spectra 분석 성공 여부는 Console Output에서 확인할 수 있습니다.

↑ 프로젝트로 돌아가기

☰ 상태

</> 바뀐점

- Console Output

View as plain text

빌드 정보 수정



### 콘솔 출력

```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace /Users/jeongdaeyong/Desktop/ongoing-task/spectra-ci/spectra-plugin/work/workspace/guide
[Spectra] Ready
ERROR: Step 'Spectra' aborted due to exception:
java.io.IOException: [Spectra] Invalid API key.
        at io.jenkins.plugins.SpectraPublisher.perform(SpectraPublisher.java:48)
        at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
```

컨트랙트 분석에 성공한 경우 Spectra Web에서 설정한 이메일로 결과가 전송됩니다.  
API Key가 유효하지 않은 경우 빌드 실패하며 관련 오류는 Console Output에 기록됩니다.

## 5. Pipeline Project에 적용하기

작성해둔 pipeline project의 configure에서, Pipeline script 작성 창으로 이동합니다.

**Configuration**

Advanced Project Options

General

고급...

Advanced Project Options

Pipeline

Pipeline

Definition

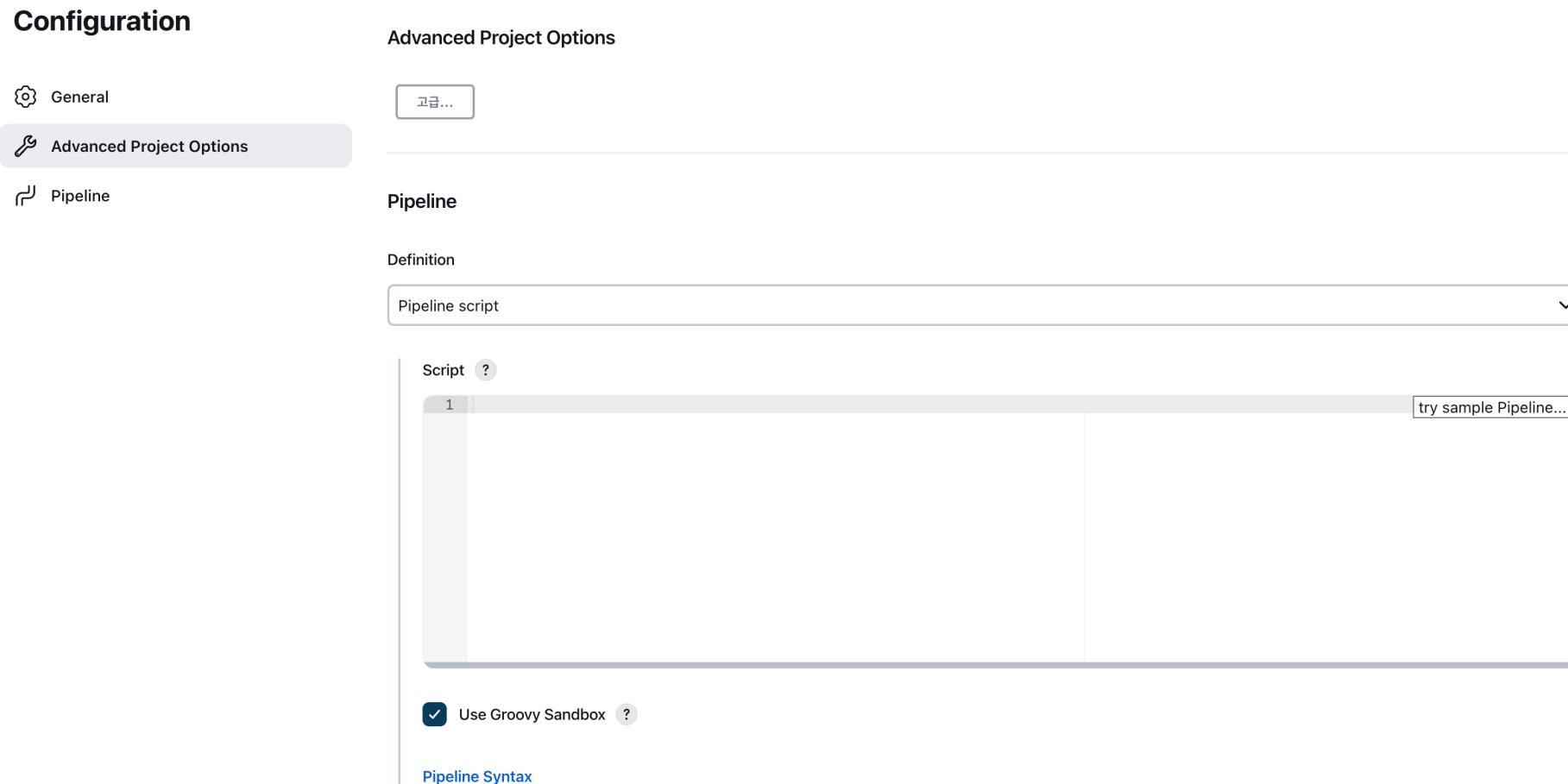
Pipeline script

Script ?

1 try sample Pipeline... ▾

Use Groovy Sandbox ?

Pipeline Syntax



## 5. Pipeline Project에 적용하기

다음과 같이 ‘Spectra’ stage를 원하는 위치에 삽입해줍니다.

```
pipeline {  
    agent any  
    stages {  
        stage('Stage A') {  
            ...  
        }  
        stage('Spectra') {  
            steps {  
                step([$class: 'SpectraBuilder', apikey: 'abcd1234'])  
            }  
        }  
    }  
}
```

**apikey: {발급받은 api key} 형식으로 입력합니다. ‘apikey’의 철자 및 대소문자에 유의해주세요.**

## 5. Pipeline Project에 적용하기

프로젝트 빌드 후 Spectra 분석 성공 여부는 Console Output에서 확인할 수 있습니다.

 Status  
  </> Changes  
 **Console Output**    View as plain text  
 Edit Build Information  
 Delete build '#1'  
 Restart from Stage  
 Replay  
 Pipeline Steps  
 Workspaces

**콘솔 출력**

```
Started by user unknown or anonymous
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/jeongdaeyong/Desktop/ongoing-task/spectra-ci/spectra-plugin/work/workspace/plguide
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Spectra)
[Pipeline] step
[Spectra] Ready
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
java.io.IOException: [Spectra] Invalid API key.
        at io.jenkins.plugins.SpectraBuilder.perform(SpectraBuilder.java:55)
        at org.jenkinsci.plugins.workflow.steps.CoreStep$Execution.run(CoreStep.java:101)
        at org.jenkinsci.plugins.workflow.steps.CoreStep$Execution.run(CoreStep.java:71)
        at org.jenkinsci.plugins.workflow.steps.SynchronousNonBlockingStepExecution.lambda$start$0(SynchronousNonBlockingStepExecution.java:47)
        at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
```

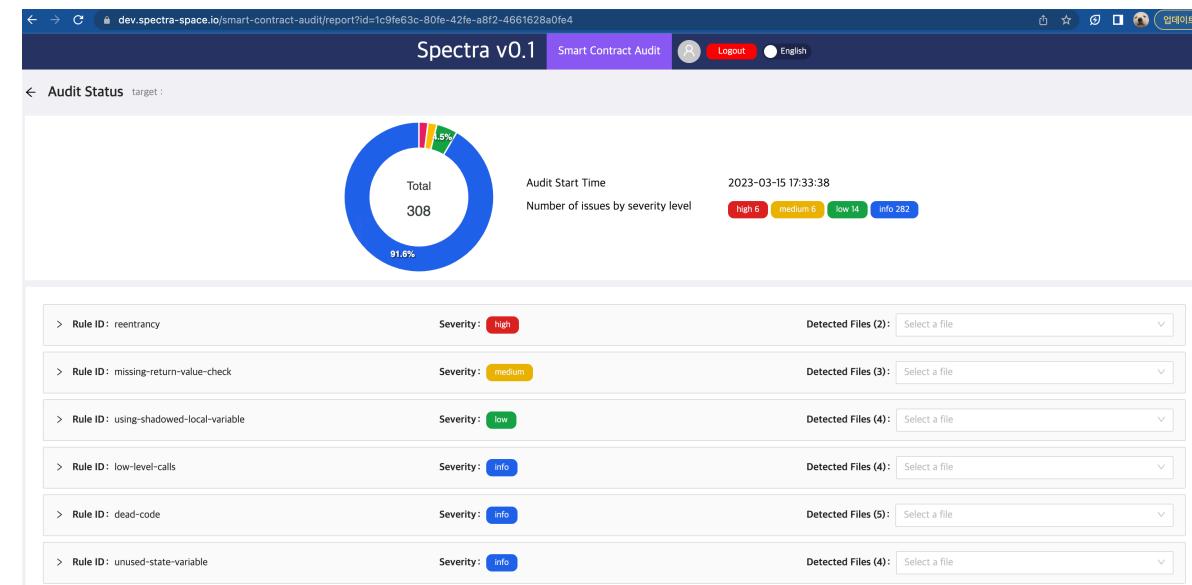
컨트랙트 분석에 성공한 경우 Spectra Web에서 설정한 이메일로 결과가 전송됩니다.  
API Key가 유효하지 않은 경우 빌드 실패하며 관련 오류는 Console Output에 기록됩니다.

Finished: FAILURE

## 6. 결과 확인하기

빌드에 성공한 경우, Spectra Web에서 설정한 이메일로 결과 링크가 주어집니다. 해당 링크를 클릭하여 결과를 볼 수 있습니다.

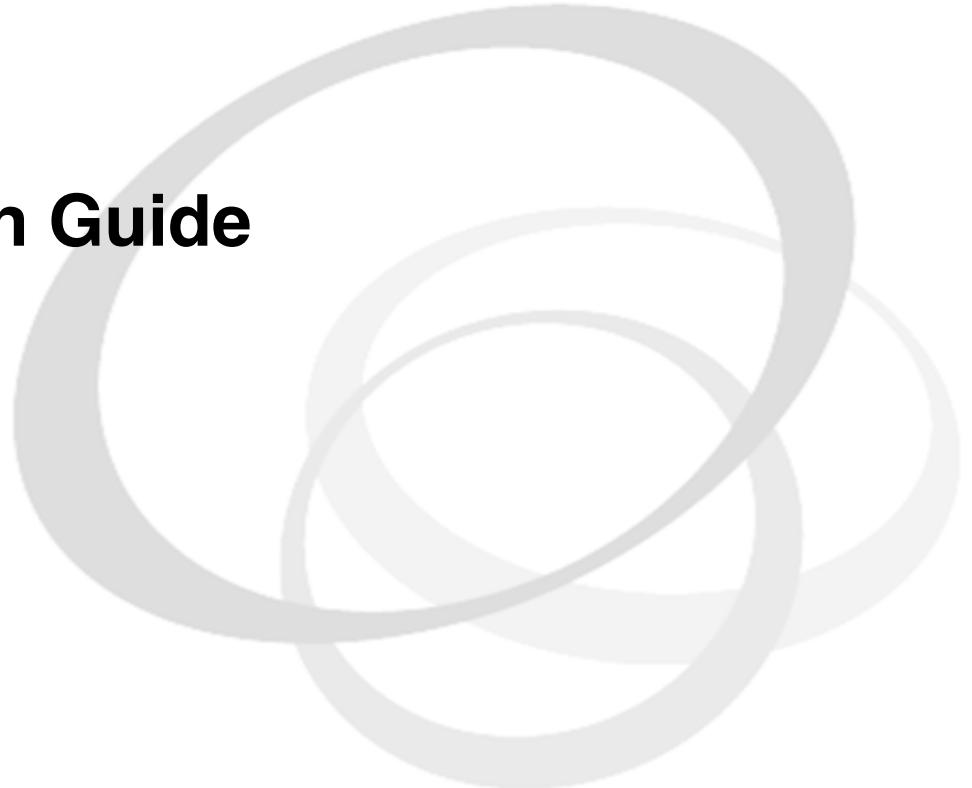
★ [Spectra] 오딧 완료 🎉  
~ 보낸사람 spectra@mdevi.io VIP  
받는사람 iwin1203@naver.com  
2023년 5월 24일 (수) 오전 10:31  
오딧을 완료했습니다. [link](#)





# Spectra Github Action Guide

2023.06



## 0. 프로젝트 구조

Spectra Github Action의 경우, 다음과 같은 프로젝트 구조를 지원합니다.

(1) 한 개 이상의 sol 파일

```
a.sol  
b.sol  
c.sol
```

(2) hardhat, truffle, brownie의 표준 구조

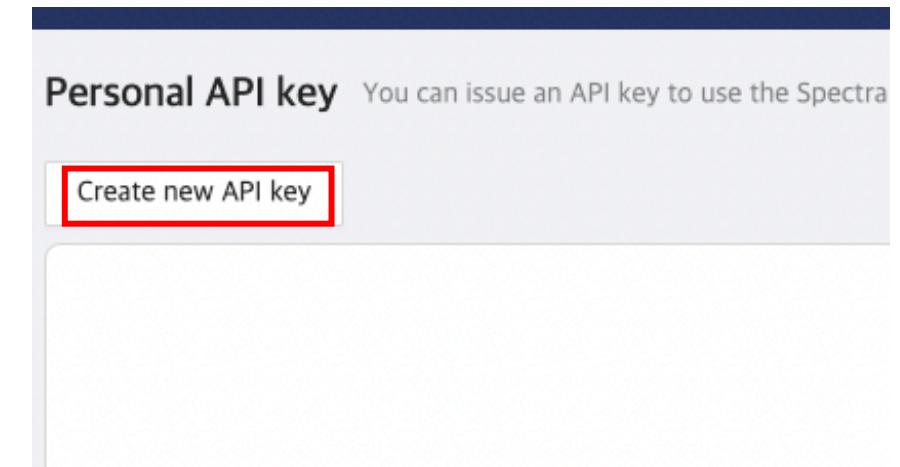
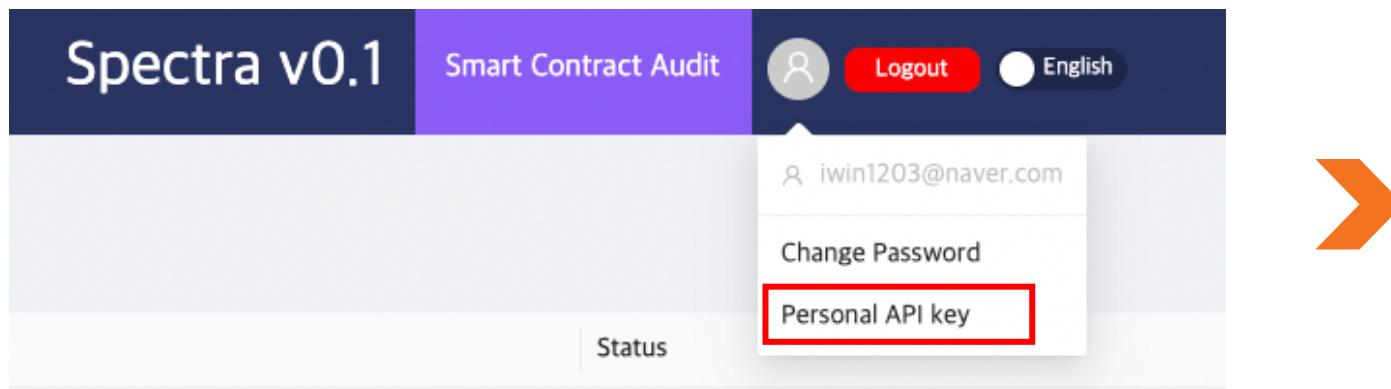
주의) 다음 중 필수는 “contracts” 경로이며, 그 외는 선택적입니다.

```
contracts/  
└ a.sol  
node_modules/  
└ @openzeppelin/  
package.json  
hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml
```

# 1. API Key Setup

\* 본 가이드라인은 Github Action을 CI에 활용하고 있는 프로젝트를 대상으로 합니다.

Spectra Web에서 프로필 아이콘 클릭 > Personal API key > Create new API key를 클릭하여 api key 발급 창으로 이동합니다.



# 1. API Key Setup

Expiration과 role을 선택하여 API key를 발급받습니다.

Create New API key

Key 만료 기한

\* Expiration: 90 days Select date

\* roles:  request audit  view audit

권한: audit 요청 권한 / 결과 조회

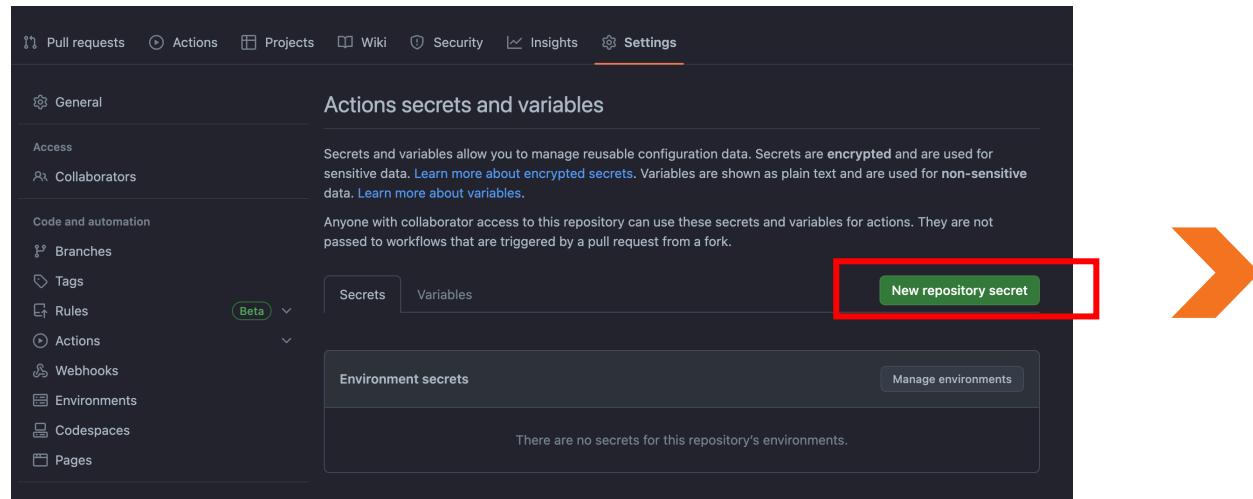
Cancel OK



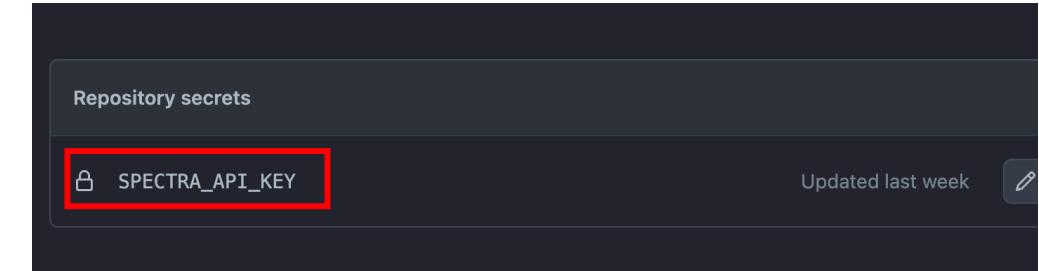
| Personal API key You can issue an API key to use the Spectra service. |  |  |
|---|--|--|
| <button>Create new API key</button>                                   | <b>발급된 Key</b>   |  |
| 5f127d95-244e-42d7-933c-5b3b1cc7a747                                  | roles : view-audit,request-audit<br>created : 2023-05-16<br>expiration date : 2023-07-15 |  |

# 1. API Key Setup

Audit을 수행할 레포지토리의 Settings > Secrets and variables > Actions에서 New repository secret를 생성합니다.



Name: SPECTRA\_API\_KEY  
Secret: 발급받은 API key



## 2. Github Action Setup

Audit을 받고자 하는 레포지토리에, “.github/workflows/spectra.yml”을 생성합니다.

### .github/workflows/spectra.yml

```
name: Spectra Github Action

on:
  push:
    branches: ["main"]

jobs:
  main:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Spectra Github Action Step
        uses: spark63/spectra-github-action@v0.1.0
        with:
          api_key: ${{ secrets.SPECTRA_API_KEY }}
          path: "target"
```

## 2. Github Action Setup

Audit을 받고자 하는 레포지토리에, “.github/workflows/spectra.yml”을 생성합니다.

```
name: Spectra Github Action

on:
  push:
    branches: ["main"]

jobs:
  main:
    runs-on: ubuntu-latest

  steps:
    - name: Checkout
      uses: actions/checkout@v3

    - name: Spectra Github Action Step
      uses: spark63/spectra-github-action@v0.1.0
      with:
        api_key: ${{ secrets.SPECTRA_API_KEY }}
        path: "target"
```

### api\_key

**Required: true**

발급받은 API key. Repository의 secret으로 넣는 경우를 상정

### path

**Required: false**

audit 대상 경로. path가 없으면 자동으로 repository의 root경로로 설정.

\* path로 진입하였을 때 프로젝트 구조는 p36과 동일하여야 합니다.

### 3. Run Github Action

Github Action이 수행되면, “Spectra Github Action Step” 단계에서 분석 여부를 확인할 수 있습니다.

```
✓ Spectra Github Action Step 6s

1 ► Run spark63/spectra-github-action@v0.1.0
5 /usr/bin/docker run --name ed866e02e7d4ad4e484644baf199c6ab23e64a_a44f98 --label ed866e --workdir /github/workspace --rm -e "INPUT_API_KEY" -e "INPUT_PATH" -e
"HOME" -e "GITHUB_JOB" -e "GITHUB_REF" -e "GITHUB_SHA" -e "GITHUB_REPOSITORY" -e "GITHUB_REPOSITORY_OWNER" -e "GITHUB_REPOSITORY_OWNER_ID" -e "GITHUB_RUN_ID" -e
"GITHUB_RUN_NUMBER" -e "GITHUB_RETENTION_DAYS" -e "GITHUB_RUN_ATTEMPT" -e "GITHUB_REPOSITORY_ID" -e "GITHUB_ACTOR_ID" -e "GITHUB_ACTOR" -e
"GITHUB_TRIGGERING_ACTOR" -e "GITHUB_WORKFLOW" -e "GITHUB_HEAD_REF" -e "GITHUB_BASE_REF" -e "GITHUB_EVENT_NAME" -e "GITHUB_SERVER_URL" -e "GITHUB_API_URL" -e
"GITHUB_GRAPHQL_URL" -e "GITHUB_REF_NAME" -e "GITHUB_REF_PROTECTED" -e "GITHUB_REF_TYPE" -e "GITHUB_WORKFLOW_REF" -e "GITHUB_WORKFLOW_SHA" -e "GITHUB_WORKSPACE"
-e "GITHUB_ACTION" -e "GITHUB_EVENT_PATH" -e "GITHUB_ACTION_REPOSITORY" -e "GITHUB_ACTION_REF" -e "GITHUB_PATH" -e "GITHUB_ENV" -e "GITHUB_STEP_SUMMARY" -e
"GITHUB_STATE" -e "GITHUB_OUTPUT" -e "RUNNER_OS" -e "RUNNER_ARCH" -e "RUNNER_NAME" -e "RUNNER_TOOL_CACHE" -e "RUNNER_TEMP" -e "RUNNER_WORKSPACE" -e
"ACTIONS_RUNTIME_URL" -e "ACTIONS_RUNTIME_TOKEN" -e "ACTIONS_CACHE_URL" -e GITHUB_ACTIONS=true -e CI=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v
"/home/runner/work/_temp/_github_home":"/github/home" -v "/home/runner/work/_temp/_github_workflow":"/github/workflow" -v
"/home/runner/work/_temp/_runner_file_commands":"/github/file_commands" -v "/home/runner/work/spectra-github-action-test/spectra-github-action-
test":"github/workspace" ed866e•02e7d4ad4e484644baf199c6ab23e64a
6 [SPECTRA] API Key authorized. Check your email for audit result.
```

성공한 경우, 이메일로 결과가 전송됩니다.  
이메일 도착까지는 최대 4분정도 소요될 수 있습니다.

### 3. Run Github Action

Github Action이 수행되면, “Spectra Github Action Step” 단계에서 분석 여부를 확인할 수 있습니다.

```
▼ ✘ Spectra Github Action Step 0s

1 ► Run spark63/spectra-github-action@v0.1.0
5 /usr/bin/docker run --name ed866e822a6f8f64774127bc7dd8bdb72e13e1_f62854 --label ed866e --workdir /github/workspace --rm -e "INPUT_API_KEY" -e "INPUT_PATH" -e
"HOME" -e "GITHUB_JOB" -e "GITHUB_REF" -e "GITHUB_SHA" -e "GITHUB_REPOSITORY" -e "GITHUB_REPOSITORY_OWNER" -e "GITHUB_REPOSITORY_OWNER_ID" -e "GITHUB_RUN_ID" -e
"GITHUB_RUN_NUMBER" -e "GITHUB_RETENTION_DAYS" -e "GITHUB_RUN_ATTEMPT" -e "GITHUB_REPOSITORY_ID" -e "GITHUB_ACTOR_ID" -e "GITHUB_ACTOR" -e
"GITHUB_TRIGGERING_ACTOR" -e "GITHUB_WORKFLOW" -e "GITHUB_HEAD_REF" -e "GITHUB_BASE_REF" -e "GITHUB_EVENT_NAME" -e "GITHUB_SERVER_URL" -e "GITHUB_API_URL" -e
"GITHUB_GRAPHQL_URL" -e "GITHUB_REF_NAME" -e "GITHUB_REF_PROTECTED" -e "GITHUB_REF_TYPE" -e "GITHUB_WORKFLOW_REF" -e "GITHUB_WORKFLOW_SHA" -e "GITHUB_WORKSPACE"
-e "GITHUB_ACTION" -e "GITHUB_EVENT_PATH" -e "GITHUB_ACTION_REPOSITORY" -e "GITHUB_ACTION_REF" -e "GITHUB_PATH" -e "GITHUB_ENV" -e "GITHUB_STEP_SUMMARY" -e
"GITHUB_STATE" -e "GITHUB_OUTPUT" -e "RUNNER_OS" -e "RUNNER_ARCH" -e "RUNNER_NAME" -e "RUNNER_TOOL_CACHE" -e "RUNNER_TEMP" -e "RUNNER_WORKSPACE" -e
"ACTIONS_RUNTIME_URL" -e "ACTIONS_RUNTIME_TOKEN" -e "ACTIONS_CACHE_URL" -e GITHUB_ACTIONS=true -e CI=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v
"/home/runner/work/_temp/_github_home":"/github/home" -v "/home/runner/work/_temp/_github_workflow":"/github/workflow" -v
"/home/runner/work/_temp/_runner_file_commands":"/github/file_commands" -v "/home/runner/work/spectra-github-action-test/spectra-github-action-
test":"/github/workspace" ed866e:822a6f8f64774127bc7dd8bdb72e13e1
6 [SPECTRA] Target path (target) not found.
```

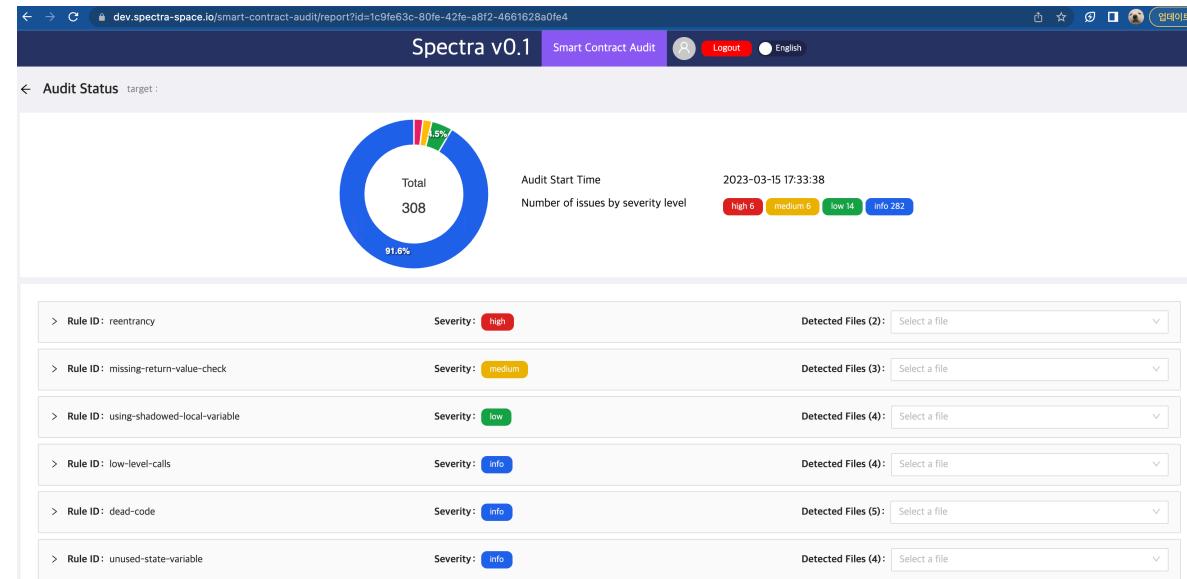
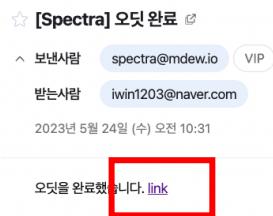
실패한 경우, 실패 원인을 살펴볼 수 있습니다.

- (1) API key 인증 실패
- (2) Argument로 설정한 ‘path’ 경로 확인 실패
- (3) 압축 파일 사이즈 30MB 초과 / 단일 파일 사이즈 8MB 초과

등이 주요 원인입니다.

## 4. Check Result

빌드에 성공한 경우, Spectra Web에서 설정한 이메일로 결과 링크가 주어집니다. 해당 링크를 클릭하여 결과를 볼 수 있습니다.



## Tip - Audit 결과가 한참 후에 도착합니다

Spectra는 package.json이 존재하면, 확실한 dependency 해결을 위해 “npm install”을 실행합니다.  
이 과정에서 시간이 다소 소요될 수 있습니다. (~5분)

만약 시간이 5분 이상 소요되는 경우, package.json을 제외하고 직접 node\_modules나 라이브러리를 직접 동봉하시는 것을 추천드립니다.  
package.json가 필요한 경우라면, 되도록 smart contract가 실제 import하는 라이브러리만 포함시키는 것을 추천드립니다.

(2) hardhat, truffle, brownie의 표준 구조  
주의) 다음 중 필수는 “contracts” 경로이며, 그 외는 선택적입니다.

```
contracts/
└ a.sol
node_modules/
└ @openzeppelin/
  package.json
hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml
```

# Tip - 원하는 파일만 audit을 하고 싶습니다

## 1. 'path' 옵션을 사용하는 방법

.github/workflows/spectra.yml 파일에서, path를 원하는 파일들만 존재하는 경로로 설정합니다.  
이 경우, 해당 경로 역시 Spectra가 허용하는 프로젝트 구조를 따라야 합니다. (36p 참고)

```
contracts/  
└ a.sol  
└ b.sol  
node_modules/  
└ @openzeppelin/  
package.json  
hardhat.config.js
```



path = "target"

```
target/  
└ b.sol  
└ node_modules/  
  └ @openzeppelin/  
contracts/  
└ a.sol  
node_modules/  
└ @openzeppelin/  
package.json  
hardhat.config.js
```

## 2. 'contract' 디렉토리를 활용하는 방법

위 왼쪽 구조에서는 contracts/ 아래의 파일만 audit의 대상이 됩니다.

원하는 파일을 contracts/에 넣고, 원치 않는 파일은 상위로 이동시키거나 혹은 다른 폴더에 넣으시면 audit에서 제외됩니다.

E O D

