Backend.AI Client SDK for Python Documentation

출시 버전 ***19.09.10***

Lablup Inc.

2020년 **08**월 **04**일

시작하기

1 Quickstart **3**

2 시작하기 **5**

2.1 설치하기 5

2.1.1 Linux/macOS 5

2.1.2 Windows 6

2.1.3 Verification 6

2.2 클라이언트 설정 6

3 Command-line Interface **9**

3.1 Configuration 9

3.1.1 Session Mode 9

3.1.2 API Mode 10

3.1.3 Checking out the current configuration 10

3.2 Compute Sessions 10

3.2.1 Listing sessions 10

3.2.2 Running simple sessions 11

3.2.3 Running sessions with accelerators 12

3.2.4 Terminating or cancelling sessions 12

3.3 Container Applications 12

3.3.1 Starting a session and connecting to its Jupyter Notebook 12

3.3.2 Accessing sessions via a web terminal 13

3.3.3 Accessing sessions via native SSH/SFTP 13

3.4 Storage Management 14

3.4.1 Creating vfolders and managing them 15

3.4.2 File transfers and management 15

3.4.3 Running sessions with storages 16

3.5 Advanced Code Execution 16

3.5.1 Running concurrent experiment sessions 16

4 Developer Reference **19**
4.1 Developer Guides . . . 19
4.1.1 Client Session . . . 19
4.1.2 Examples . . . 22
Synchronous-mode execution . . . 22
Asynchronous-mode Execution . . . 25
4.1.3 Testing . . . 26
Unit Tests . . . 26
Integration Tests . . . 27
4.2 고수준 함수 인터페이스 . . . 28
4.2.1 관리자용 함수 . . . 28
4.2.2 Auth Functions . . . 29
4.2.3 에이전트 함수 . . . 29
4.2.4 Configuration . . . 30
4.2.5 커널 함수 . . . 33
4.2.6 KeyPair 함수 . . . 37
4.2.7 매니저 함수 . . . 38
4.2.8 가상폴더 함수 . . . 39
4.2.9 Scaling Group Functions . . . 40
4.3 저수준 레퍼런스 . . . 42
4.3.1 기반 함수 . . . 42
4.3.2 요청 API . . . 42
4.3.3 예외 클래스들 . . . 45

5 색인과 표 **47**

Python 모듈 목록 **49**

색인 **51**

이 문서는 Backend.AI API를 구현하는 Python용 클라이언트 소프트웨어 개발킷(SDK)에 관한 매뉴얼입니다.

CHAPTER 1

Quickstart

Python 3.6 or higher is required.

python.org에서 공식 설치 패키지를 다운로드하거나, 사용 중인 운영환경에 적합한 별도의 패키지 관리자(예: homebrew, miniconda, pyenv 등)를 이용할 수 있습니다. 이 클라이언트 SDK는 Linux, macOS, Windows 환경에서 테스트되었습니다.

SDK 라이브러리와 도구를 다른 소프트웨어와의 충돌 없이 설치하기 위해서 별도의 Python 가상환경(virtual environment)을 만드실 것을 권장합니다.

```
$ python3 -m venv venv-backend-ai
$ source venv-backend-ai/bin/activate
(venv-backend-ai) $
```

PyPI로부터 클라이언트 라이브러리를 설치합니다.

```
(venv-backend-ai) $ pip install -U pip setuptools
(venv-backend-ai) $ pip install backend.ai-client
```

자신의 API keypair를 다음과 같이 환경변수에 설정합니다:

```
(venv-backend-ai) $ export BACKEND_ACCESS_KEY=AKIA...
(venv-backend-ai) $ export BACKEND_SECRET_KEY=...
```

그 다음엔 첫 명령어를 실행해봅니다:

```
(venv-backend-ai) $ backend.ai --help
...
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
(venv-backend-ai) $ backend.ai ps
...
```

Check out more details about *client configuration*, the command-line examples, and *SDK code examples*.

CHAPTER 2

시작하기

2.1 설치하기

2.1.1 Linux/macOS

We recommend using pyenv to manage your Python versions and virtual environments to avoid conflicts with other Python applications.

Create a new virtual environment (Python 3.6 or higher) and activate it on your shell. Then run the following commands:

```
pip install -U pip setuptools
pip install -U backend.ai-client-py
```

Create a shell script `my-backendai-env.sh` like:

```
export BACKEND_ACCESS_KEY=...
export BACKEND_SECRET_KEY=...
export BACKEND_ENDPOINT=https://my-precious-cluster
export BACKEND_ENDPOINT_TYPE=api
```

Run this shell script before using `backend.ai` command.

참고: The console-server users should set `BACKEND_ENDPOINT_TYPE` to `session`. For details, check out *the client configuration document*.

2.1.2 Windows

We recommend using the Anaconda Navigator to manage your Python environments with a slick GUI app.

Create a new environment (Python 3.6 or higher) and launch a terminal (command prompt). Then run the following commands:

```
python -m pip install -U pip setuptools
python -m pip install -U backend.ai-client-py
```

Create a batch file `my-backendai-env.bat` like:

```
chcp 65001
set PYTHONIOENCODING=UTF-8
set BACKEND_ACCESS_KEY=...
set BACKEND_SECRET_KEY=...
set BACKEND_ENDPOINT=https://my-precious-cluster
set BACKEND_ENDPOINT_TYPE=api
```

Run this batch file before using `backend.ai` command.

Note that this batch file switches your command prompt to use the UTF-8 codepage for correct display of special characters in the console logs.

2.1.3 Verification

Run `backend.ai ps` command and check if it says "there is no compute sessions running" or something similar.

If you encounter error messages about "ACCESS_KEY", then check if your batch/shell scripts have the correct environment variable names.

If you encounter network connection error messages, check if the endpoint server is configured correctly and accessible.

2.2 클라이언트 설정

Backend.AI API 설정에는 endpoint URL과 API keypair (access 및 secret key) 등이 포함됩니다.

설정 방법은 2가지가 있습니다:

1. Setting environment variables before running your program that uses this SDK. This applies to the command-line interface as well.
2. Manually creating *`APIConfig`* instance and creating sessions with it.

The list of configurable environment variables are:

- `BACKEND_ENDPOINT`
- `BACKEND_ENDPOINT_TYPE`
- `BACKEND_ACCESS_KEY`
- `BACKEND_SECRET_KEY`
- `BACKEND_VFOLDER_MOUNTS`

Please refer the parameter descriptions of *`APIConfig`*'s constructor for what each environment variable means and what value format should be used.

CHAPTER 3

Command-line Interface

3.1 Configuration

참고: Please consult the detailed usage in the help of each command (use `-h` or `--help` argument to display the manual).

Check out *the client configuration* for configurations via environment variables.

3.1.1 Session Mode

When the endpoint type is `"session"`, you must explicitly login and logout into/from the console server.

```
$ backend.ai login
Username: myaccount@example.com
Password:
✓ Login succeeded.

$ backend.ai ...  # any commands

$ backend.ai logout
✓ Logout done.
```

3.1.2 API Mode

After setting up the environment variables, just run any command:

```
$ backend.ai ...
```

3.1.3 Checking out the current configuration

Run the following command to list your current active configurations.

```
$ backend.ai config
```

3.2 Compute Sessions

참고: Please consult the detailed usage in the help of each command (use `-h` or `--help` argument to display the manual).

3.2.1 Listing sessions

List the session owned by you with various status filters. The most recently status-changed sessions are listed first. To prevent overloading the server, the result is limited to the first 10 sessions and it provides a separate `--all` option to paginate further sessions.

```
backend.ai ps
```

The `ps` command is an alias of the following `admin sessions` command. If you have the administrator privilege, you can list sessions owned by other users by adding `--access-key` option here.

```
backend.ai admin sessions
```

Both commands offers options to set the status filter as follows. For other options, please consult the output of `--help`.

Option	Included Session Status
(no option)	`PENDING`, `PREPARING`, `RUNNING`, `RESTARTING`, `TERMINATING`, `RESIZING`, `SUSPENDED`, and `ERROR`.
`--running`	`PREPARING`, `PULLING`, and `RUNNING`.
`--dead`	`CANCELLED` and `TERMINATED`.

3.2.2 Running simple sessions

The following command spawns a Python session and executes the code passed as `-c` argument immediately. `--rm` option states that the client automatically terminates the session after execution finishes.

```
backend.ai run --rm -c 'print("hello world")' python:3.6-ubuntu18.04
```

참고: By default, you need to specify language with full version tag like `python:3.6-ubuntu18.04`. Depending on the Backend.AI admin's language alias settings, this can be shortened just as `python`. If you want to know defined language aliases, contact the admin of Backend.AI server.

The following command spawns a Python session and executes the code passed as `./myscript.py` file, using the shell command specified in the `--exec` option.

```
backend.ai run --rm --exec 'python myscript.py arg1 arg2' \
           python:3.6-ubuntu18.04 ./myscript.py
```

Please note that your `run` command may hang up for a very long time due to queueing when the cluster resource is not sufficiently available.

To avoid indefinite waiting, you may add `--enqueue-only` to return immediately after posting the session creation request.

참고: When using `--enqueue-only`, the codes are *NOT* executed and relevant options are ignored. This makes the `run` command to the same of the `start` command.

Or, you may use `--max-wait` option to limit the maximum waiting time. If the session starts within the given `--max-wait` seconds, it works normally, but if not, it returns without code execution like when used `--enqueue-only`.

To watch what is happening behind the scene until the session starts, try `backend.ai events <sessionID>` to receive the lifecycle events such as its scheduling and preparation steps.

3.2.3 Running sessions with accelerators

Use one or more `-r` options to specify resource requirements when using `backend.ai run` and `backend.ai start` commands.

For instance, the following command spawns a Python TensorFlow session using a half of virtual GPU device, 4 CPU cores, and 8 GiB of the main memory to execute `./mygpucode.py` file inside it.

```
backend.ai run --rm \
           -r cpu=4 -r mem=8g -r cuda.shares=2 \
           python-tensorflow:1.12-py36 ./mygpucode.py
```

3.2.4 Terminating or cancelling sessions

Without `--rm` option, your session remains alive for a configured amount of idle timeout (default is 30 minutes). You can see such sessions using the `backend.ai ps` command. Use the following command to manually terminate them via their session IDs. You may specifcy multiple session IDs to terminate them at once.

```
backend.ai rm <sessionID> [<sessionID>...]
```

If you terminate `PENDING` sessions which are not scheduled yet, they are cancelled.

3.3 Container Applications

참고: Please consult the detailed usage in the help of each command (use `-h` or `--help` argument to display the manual).

3.3.1 Starting a session and connecting to its Jupyter Notebook

The following command first spawns a Python session named "mysession" without running any code immediately, and then executes a local proxy which connects to the "jupyter" service running inside the session via the local TCP port 9900. The `start` command shows application services provided by the created compute session so that you can choose one in the subsequent `app` command. In the `start` command, you can specify detailed resource options using `-r` and storage mounts using `-m` parameter.

```
backend.ai start -t mysession python
backend.ai app -b 9900 mysession jupyter
```

Once executed, the `app` command waits for the user to open the displayed address using appropriate application. For the jupyter service, use your favorite web browser just like the way you use Jupyter Notebooks. To stop the `app` command, press `Ctrl+C` or send the `SIGINT` signal.

3.3.2 Accessing sessions via a web terminal

All Backend.AI sessions expose an intrinsic application named `"ttyd"`. It is an web application that embeds xterm.js-based full-screen terminal that runs on web browsers.

```
backend.ai start -t mysession ...
backend.ai app -b 9900 mysession ttyd
```

Then open `http://localhost:9900` to access the shell in a fully functional web terminal using browsers. The default shell is `/bin/bash` for Ubuntu/CentOS-based images and `/bin/ash` for Alpine-based images with a fallback to `/bin/sh`.

참고: This shell access does *NOT* grant your root access. All compute session processes are executed as the user privilege.

3.3.3 Accessing sessions via native SSH/SFTP

Backend.AI offers direct access to compute sessions (containers) via SSH and SFTP, by auto-generating host identity and user keypairs for all sessions. All Baceknd.AI sessions expose an intrinsic application named `"sshd"` like `"ttyd"`.

To connect your sessions with SSH, first prepare your session and download an auto-generated SSH keypair named `id_container`. Then start the service port proxy ("app" command) to open a local TCP port that proxies the SSH/SFTP traffic to the compute sessions:

```
$ backend.ai start -t mysess ...
$ backend.ai download mysess id_container
$ mv id_container ~/.ssh
$ backend.ai app mysess sshd -b 9922
```

In another terminal on the same PC, run your ssh client like:

```
$ ssh -o StrictHostKeyChecking=no \
>     -o UserKnownHostsFile=/dev/null \
>     -i ~/.ssh/id_container \
>     work@localhost -p 9922
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
Warning: Permanently added '[127.0.0.1]:9922' (RSA) to the list of known hosts.
f310e8dbce83:~$
```

This SSH port is also compatible with SFTP to browse the container's filesystem and to upload/download large-sized files.

You could add the following to your `~/.ssh/config` to avoid type extra options every time.

```
Host localhost
  User work
  IdentityFile ~/.ssh/id_container
  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null
```

```
$ ssh localhost -p 9922
```

경고: Since the SSH keypair is auto-generated every time when your launch a new compute session, you need to download and keep it separately for each session.

To use your own SSH private key across all your sessions without downloading the auto-generated one every time, create a vfolder named `.ssh` and put the `authorized_keys` file that includes the public key. The keypair and `.ssh` directory permissions will be automatically updated by Backend.AI when the session launches.

```
$ ssh-keygen -t rsa -b 2048 -f id_container
$ cat id_container.pub > authorized_keys
$ backend.ai vfolder create .ssh
$ backend.ai vfolder upload .ssh authorized_keys
```

3.4 Storage Management

참고: Please consult the detailed usage in the help of each command (use `-h` or `--help` argument to display the manual).

Backend.AI abstracts shared network storages into per-user slices called **"virtual folders"** (aka **"vfolders"**), which can be shared between users and user group members.

3.4.1 Creating vfolders and managing them

The command-line interface provides a set of subcommands under `backend.ai vfolder` to manage vfolders and files inside them.

To list accessible vfolders including your own ones and those shared by other users:

```
$ backend.ai vfolder list
```

To create a virtual folder named "mydata1":

```
$ backend.ai vfolder create mydata1 mynas
```

The second argument `mynas` corresponds to the name of a storage host. To list up storage hosts that you are allowed to use:

```
$ backend.ai vfolder list-hosts
```

To delete the vfolder completey:

```
$ backend.ai vfolder delete mydata1
```

3.4.2 File transfers and management

To upload a file from the current working directory into the vfolder:

```
$ backend.ai vfolder upload mydata1 ./bigdata.csv
```

To download a file from the vfolder into the current working directory:

```
$ backend.ai vfolder download mydata1 ./bigresult.txt
```

To list files in the vfolder's specific path:

```
$ backend.ai vfolder ls mydata1 .
```

To delete files in the vfolder:

```
$ backend.ai vfolder rm mydata1 ./bigdata.csv
```

경고: All file uploads and downloads overwrite existing files and all file operations are irreversible.

3.4.3 Running sessions with storages

The following command spawns a Python session where the virtual folder "mydata1" is mounted. The execution options are omitted in this example. Then, it downloads `./bigresult.txt` file (generated by your code) from the "mydata1" virtual folder.

```
$ backend.ai vfolder upload mydata1 ./bigdata.csv
$ backend.ai run --rm -m mydata1 python:3.6-ubuntu18.04 ...
$ backend.ai vfolder download mydata1 ./bigresult.txt
```

In your code, you may access the virtual folder via `/home/work/mydata1` (where the default current working directory is `/home/work`) just like a normal directory.

By reusing the same vfolder in subsequent sessions, you do not have to donwload the result and upload it as the input for next sessions, just keeping them in the storage.

3.5 Advanced Code Execution

참고: Please consult the detailed usage in the help of each command (use `-h` or `--help` argument to display the manual).

3.5.1 Running concurrent experiment sessions

In addition to single-shot code execution as described in *Running simple sessions*, the `run` command offers concurrent execution of multiple sessions with different parameters interpolated in the execution command specified in `--exec` option and environment variables specified as `-e` / `--env` options.

To define variables interpolated in the `--exec` option, use `--exec-range`. To define variables interpolated in the `--env` options, use `--env-rage`.

Here is an example with environment variable ranges that expands into 4 concurrent sessions.

```
backend.ai run -c 'import os; print("Hello world, {}".format(os.environ["CASENO"]))
↪' \
    -r cpu=1 -r mem=256m \
    -e 'CASENO=$X' \
    --env-range=X=case:1,2,3,4 \
    lablup/python:3.6-ubuntu18.04
```

Both range options accept a special form of argument: "range expressions". The front part

of range option value consists of the variable name used for interpolation and an equivalence sign (=). The rest of range expressions have the following three types:

Expression	Interpretation
`case:CASE1,CASE2,...,CASEN`	A list of discrete values. The values may be either string or numbers.
`linspace:START,STOP,POINTS`	An inclusive numerical range with discrete points, in the same way of `numpy.linspace()`. For example, `linspace:1,2,3` generates a list of three values: 1, 1.5, and 2.
`range:START,STOP,STEP`	A numerical range with the same semantics of Python's `range()`. For example, `range:1,6,2` generates a list of values: 1, 3, and 5.

If you specify multiple occurrences of range options in the `run` command, the client spawns sessions for *all possible combinations* of all values specified by each range.

참고: When your resource limit and cluster's resource capacity cannot run all spawned sessions at the same time, some of sessions may be queued and the command may take a long time to finish.

경고: Until all cases finish, the client must keep its network connections to the server alive because this feature is implemented in the client-side. Server-side batch job scheduling is under development!

CHAPTER 4

Developer Reference

4.1 Developer Guides

4.1.1 Client Session

This module is the first place to begin with your Python programs that use Backend.AI API functions.

The high-level API functions cannot be used alone -- you must initiate a client session first because each session provides *proxy attributes* that represent API functions and run on the session itself.

To achieve this, during initialization session objects internally construct new types by combining the *BaseFunction* class with the attributes in each API function classes, and makes the new types bound to itself. Creating new types every time when creating a new session instance may look weird, but it is the most convenient way to provide *class-methods* in the API function classes to work with specific *session instances*.

When designing your application, please note that session objects are intended to live long following the process' lifecycle, instead of to be created and disposed whenever making API requests.

class ai.backend.client.session.**BaseSession**(*, *config=None*)

The base abstract class for sessions.

abstractmethod close()

Terminates the session and releases underlying resources.

closed
Checks if the session is closed.

반환 형식 `bool`

config
The configuration used by this session object.

class ai.backend.client.session.**Session**(**, config=None*)
An API client session that makes API requests synchronously. You may call (almost) all function proxy methods like a plain Python function. It provides a context manager interface to ensure closing of the session upon errors and scope exits.

Admin
The *Admin* function proxy bound to this session.

Agent
The *Agent* function proxy bound to this session.

AgentWatcher
The `AgentWatcher` function proxy bound to this session.

Auth
The `Auth` function proxy bound to this session.

EtcdConfig
The `EtcdConfig` function proxy bound to this session.

Domain
The `Domain` function proxy bound to this session.

Group
The `Group` function proxy bound to this session.

Image
The `Image` function proxy bound to this session.

Kernel
The *Kernel* function proxy bound to this session.

KeyPair
The *KeyPair* function proxy bound to this session.

Manager
The *Manager* function proxy bound to this session.

Resource
The `Resource` function proxy bound to this session.

KeypairResourcePolicy
The `KeypairResourcePolicy` function proxy bound to this session.

User

The `User` function proxy bound to this session.

closed

Checks if the session is closed.

반환 형식 `bool`

config

The configuration used by this session object.

ScalingGroup

The *ScalingGroup* function proxy bound to this session.

VFolder

The *VFolder* function proxy bound to this session.

close()

Terminates the session. It schedules the `close()` coroutine of the underlying aiohttp session and then enqueues a sentinel object to indicate termination. Then it waits until the worker thread to self-terminate by joining.

worker_thread

The thread that internally executes the asynchronous implementations of the given API functions.

class ai.backend.client.session.**AsyncSession**(**, config=None*)

An API client session that makes API requests asynchronously using coroutines. You may call all function proxy methods like a coroutine. It provides an async context manager interface to ensure closing of the session upon errors and scope exits.

closed

Checks if the session is closed.

반환 형식 `bool`

config

The configuration used by this session object.

Admin

The *Admin* function proxy bound to this session.

Agent

The *Agent* function proxy bound to this session.

AgentWatcher

The `AgentWatcher` function proxy bound to this session.

Auth

The `Auth` function proxy bound to this session.

EtcdConfig
The `EtcdConfig` function proxy bound to this session.

Group
The `Group` function proxy bound to this session.

Image
The `Image` function proxy bound to this session.

Kernel
The *Kernel* function proxy bound to this session.

KeyPair
The *KeyPair* function proxy bound to this session.

Manager
The *Manager* function proxy bound to this session.

Resource
The `Resource` function proxy bound to this session.

KeypairResourcePolicy
The `KeypairResourcePolicy` function proxy bound to this session.

User
The `User` function proxy bound to this session.

ScalingGroup
The *ScalingGroup* function proxy bound to this session.

VFolder
The *VFolder* function proxy bound to this session.

await close()
Terminates the session and releases underlying resources.

4.1.2 Examples

Synchronous-mode execution

Query mode

This is the minimal code to execute a code snippet with this client SDK.

```
import sys
from ai.backend.client import Session

with Session() as session:
    kern = session.Kernel.get_or_create('python:3.6-ubuntu18.04')
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
    code = 'print("hello world")'
    mode = 'query'
    run_id = None
    while True:
        result = kern.execute(run_id, code, mode=mode)
        run_id = result['runId']  # keeps track of this particular run loop
        for rec in result.get('console', []):
            if rec[0] == 'stdout':
                print(rec[1], end='', file=sys.stdout)
            elif rec[0] == 'stderr':
                print(rec[1], end='', file=sys.stderr)
            else:
                handle_media(rec)
        sys.stdout.flush()
        if result['status'] == 'finished':
            break
        else:
            mode = 'continued'
            code = ''
    kern.destroy()
```

You need to take care of `client_token` because it determines whether to reuse kernel sessions or not. Backend.AI cloud has a timeout so that it terminates long-idle kernel sessions, but within the timeout, any kernel creation requests with the same `client_token` let Backend.AI cloud to reuse the kernel.

Batch mode

You first need to upload the files after creating the session and construct a `opts` struct.

```
import sys
from ai.backend.client import Session

with Session() as session:
    kern = session.Kernel.get_or_create('python:3.6-ubuntu18.04')
    kern.upload(['mycode.py', 'setup.py'])
    code = ''
    mode = 'batch'
    run_id = None
    opts = {
        'build': '*',  # calls "python setup.py install"
        'exec': 'python mycode.py arg1 arg2',
    }
    while True:
        result = kern.execute(run_id, code, mode=mode, opts=opts)
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
        opts.clear()
        run_id = result['runId']
        for rec in result.get('console', []):
            if rec[0] == 'stdout':
                print(rec[1], end='', file=sys.stdout)
            elif rec[0] == 'stderr':
                print(rec[1], end='', file=sys.stderr)
            else:
                handle_media(rec)
        sys.stdout.flush()
        if result['status'] == 'finished':
            break
        else:
            mode = 'continued'
            code = ''
    kern.destroy()
```

Handling user inputs

Inside the while-loop for `kern.execute()` above, change the if-block for `result['status']` as follows:

```
...
if result['status'] == 'finished':
    break
elif result['status'] == 'waiting-input':
    mode = 'input'
    if result['options'].get('is_password', False):
        code = getpass.getpass()
    else:
        code = input()
else:
    mode = 'continued'
    code = ''
...
```

A common gotcha is to miss setting `mode = 'input'`. Be careful!

Handling multi-media outputs

The `handle_media()` function used above examples would look like:

```
def handle_media(record):
    media_type = record[0]  # MIME-Type string
    media_data = record[1]  # content
    ...
```

The exact method to process `media_data` depends on the `media_type`. Currently the following behaviors are well-defined:

- For (binary-format) images, the content is a dataURI-encoded string.
- For SVG (scalable vector graphics) images, the content is an XML string.
- For `application/x-sorna-drawing`, the content is a JSON string that represents a set of vector drawing commands to be replayed the client-side (e.g., Javascript on browsers)

Asynchronous-mode Execution

The async version has all sync-version interfaces as coroutines but comes with additional features such as `stream_execute()` which streams the execution results via websockets and `stream_pty()` for interactive terminal streaming.

```
import asyncio
import json
import sys
import aiohttp
from ai.backend.client import AsyncSession

async def main():
    async with AsyncSession() as session:
        kern = await session.Kernel.get_or_create('python:3.6-ubuntu18.04',
                                                  client_token='mysession')
        code = 'print("hello world")'
        mode = 'query'
        async with kern.stream_execute(code, mode=mode) as stream:
            # no need for explicit run_id since WebSocket connection represents it!
            async for result in stream:
                if result.type != aiohttp.WSMsgType.TEXT:
                    continue
                result = json.loads(result.data)
                for rec in result.get('console', []):
                    if rec[0] == 'stdout':
                        print(rec[1], end='', file=sys.stdout)
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
                    elif rec[0] == 'stderr':
                        print(rec[1], end='', file=sys.stderr)
                    else:
                        handle_media(rec)
                sys.stdout.flush()
                if result['status'] == 'finished':
                    break
                elif result['status'] == 'waiting-input':
                    mode = 'input'
                    if result['options'].get('is_password', False):
                        code = getpass.getpass()
                    else:
                        code = input()
                    await stream.send_text(code)
                else:
                    mode = 'continued'
                    code = ''
        await kern.destroy()

loop = asyncio.get_event_loop()
try:
    loop.run_until_complete(main())
finally:
    loop.stop()
```

버전 1.5에 추가.

4.1.3 Testing

Unit Tests

Unit tests perform function-by-function tests to ensure their individual functionality. This test suite runs without depending on the server-side and thus it is executed in Travis CI for every push.

How to run

```
$ python -m pytest -m 'not integration' tests
```

Integration Tests

Integration tests combine multiple invocations of high-level interfaces to make underlying API requests to a running gateway server to test the full functionality of the client as well as the manager.

They are marked as "integration" using the `@pytest.mark.integration` decorator to each test case.

> **경고:** The integration tests actually make changes to the target gateway server and agents. If some tests fail, those changes may remain in an inconsistent state and requires a manual recovery such as resetting the database and populating fixtures again, though the test suite tries to clean up them properly.
>
> So, DO NOT RUN it against your production server.

Prerequisite

Please refer the README of the manager and agent repositories to set up them. To avoid an indefinite waiting time for pulling Docker images:

- (manager) `python -m ai.backend.manager.cli etcd rescan-images`
- (agent) `docker pull`
  - `lablup/python:3.6-ubuntu18.04`
  - `lablup/lua:5.3-alpine3.8`

The manager must also have at least the following active suerp-admin account in the `default` domain and the `default` group.

- Example super-admin account:
  - User ID: `admin@lablup.com`
  - Password `wJalrXUt`
  - Access key: `AKIAIOSFODNN7EXAMPLE`
  - Secret key: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

One or more `testing-XXXX` domain, one or more `testing-XXXX` groups, and one ore more dummy users are created and used during the tests and destroyed after running tests. `XXXX` will be filled with random identifiers.

팁: The *halfstack* configuration and the `example-keypairs.json` fixture is compatible with

this integration test suite.

How to run

Execute the gateway and at least one agent in their respective virtualenvs and hosts:

```
$ python -m ai.backend.client.gateway.server
$ python -m ai.backend.client.agent.server
$ python -m ai.backend.client.agent.watcher
```

Then run the tests:

```
$ export BACKEND_ENDPOINT=...
$ python -m pytest -m 'integration' tests
```

4.2 고수준 함수 인터페이스

4.2.1 관리자용 함수

class ai.backend.client.admin.**Admin**

관리자 GraphQL 쿼리를 날리고 받아오는 함수형 인터페이스를 제공합니다.

참고: Depending on the privilege of your API access key, you may or may not have access to querying/mutating server-side resources of other users.

---

session = None

이 함수 클래스가 사용할 클라이언트 세션 인스턴스

classmethod await query(*query*, *variables=None*)

Sends the GraphQL query and returns the response.

매개변수

- **query** (str) -- The GraphQL query string.
- **variables** (Optional[Mapping[str, Any]]) -- An optional key-value dictionary to fill the interpolated template variables in the query.

반환 형식 Any

반환값 The object parsed from the response JSON string.

4.2.2 Auth Functions

class ai.backend.client.auth.**Auth**

Provides the function interface for login session management and authorization.

classmethod await login(*user_id, password*)

Log-in into the endpoint with the given user ID and password. It creates a server-side web session and return a dictionary with "authenticated" boolean field and JSON-encoded raw cookie data.

반환 형식 dict

classmethod await logout()

Log-out from the endpoint. It clears the server-side web session.

반환 형식 None

classmethod await update_password(*old_password, new_password, new_password2*)

Update user's password. This API works only for account owner.

반환 형식 dict

4.2.3 에이전트 함수

class ai.backend.client.agent.**Agent**

서버측 에이전트들에 관한 정보를 조회하기 위한 *Admin.query()* 함수의 단축 버전입니다.

참고: 이 함수 클래스의 모든 메소드들은 사용 중인 API access key가 관리자 권한을 가지고 있어야 작동합니다.

---

session = None

이 함수 클래스가 사용할 클라이언트 세션 인스턴스

classmethod await list_with_limit(*limit, offset, status='ALIVE', fields=None*)

Fetches the list of agents with the given status with limit and offset for pagination.

매개변수

- **limit** -- number of agents to get
- **offset** -- offset index of agents to get
- **status** (str) -- An upper-cased string constant representing agent status (one of 'ALIVE', 'TERMINATED', 'LOST', etc.)

- **fields** (Optional[Iterable[str]]) -- Additional per-agent query fields to fetch.

반환 형식 Sequence[dict]

classmethod await detail(*agent_id*, *fields=None*)

반환 형식 Sequence[dict]

4.2.4 Configuration

ai.backend.client.config.**get_env**(*key*, *default=<object object>*, ***, *clean=<function <lambda>>*)

Retrieves a configuration value from the environment variables. The given *key* is uppercased and prefixed by "BACKEND_" and then "SORNA_" if the former does not exist.

매개변수

- **key** (str) -- The key name.
- **default** (Any) -- The default value returned when there is no corresponding environment variable.
- **clean** (Callable[[str], Any]) -- A single-argument function that is applied to the result of lookup (in both successes and the default value for failures). The default is returning the value as-is.

반환값 The value processed by the *clean* function.

ai.backend.client.config.**get_config**()

Returns the configuration for the current process. If there is no explicitly set *APIConfig* instance, it will generate a new one from the current environment variables and defaults.

ai.backend.client.config.**set_config**(*conf*)

Sets the configuration used throughout the current process.

class ai.backend.client.config.**APIConfig**(***, *endpoint=None*, *endpoint_type=None*, *domain=None*, *group=None*, *version=None*, *user_agent=None*, *access_key=None*, *secret_key=None*, *hash_type=None*, *vfolder_mounts=None*, *skip_sslcert_validation=None*, *connection_timeout=None*, *read_timeout=None*, *announcement_handler=None*)

Represents a set of API client configurations. The access key and secret key are mandatory -- they must be set in either environment variables or as the explicit arguments.

매개변수

- **endpoint** (Union[str, URL, None]) -- The URL prefix to make API requests via HTTP/HTTPS. If this is given as `str` and contains multiple URLs separated by comma, the underlying HTTP request-response facility will perform client-side load balancing and automatic failover using them, assuming that all those URLs indicates a single, same cluster. The users of the API and CLI will get network connection errors only when all of the given endpoints fail -- intermittent failures of a subset of endpoints will be hidden with a little increased latency.
- **endpoint_type** (Optional[str]) -- Either `"api"` or `"session"`. If the endpoint type is `"api"` (the default if unspecified), it uses the access key and secret key in the configuration to access the manager API server directly. If the endpoint type is `"session"`, it assumes the endpoint is a Backend.AI console server which provides cookie-based authentication with username and password. In the latter, users need to use `backend.ai login` and `backend.ai logout` to manage their sign-in status, or the API equivalent in *login()* and *logout()* methods.
- **version** (Optional[str]) -- The API protocol version.
- **user_agent** (Optional[str]) -- A custom user-agent string which is sent to the API server as a `User-Agent` HTTP header.
- **access_key** (Optional[str]) -- The API access key. If deliberately set to an empty string, the API requests will be made without signatures (anonymously).
- **secret_key** (Optional[str]) -- The API secret key.
- **hash_type** (Optional[str]) -- The hash type to generate per-request authentication signatures.
- **vfolder_mounts** (Optional[Iterable[str]]) -- A list of vfolder names (that must belong to the given access key) to be automatically mounted upon any *Kernel.get_or_create()* calls.

DEFAULTS = {'connection_timeout': 10.0, 'domain': 'default', 'endpoint': 'https://ap

The default values for config parameterse settable via environment variables xcept the access and secret keys.

endpoint

The currently active endpoint URL. This may change if there are multiple configured endpoints and the current one is not accessible.

반환 형식 URL

endpoints

All configured endpoint URLs.

반환 형식 Sequence[URL]

endpoint_type

The configured endpoint type.

반환 형식 str

domain

The configured domain.

반환 형식 str

group

The configured group.

반환 형식 str

user_agent

The configured user agent string.

반환 형식 str

access_key

The configured API access key.

반환 형식 str

secret_key

The configured API secret key.

반환 형식 str

version

The configured API protocol version.

반환 형식 str

hash_type

The configured hash algorithm for API authentication signatures.

반환 형식 str

vfolder_mounts

The configured auto-mounted vfolder list.

반환 형식 Tuple[str, ...]

skip_sslcert_validation
Whether to skip SSL certificate validation for the API gateway.

반환 형식 bool

connection_timeout
The maximum allowed duration for making TCP connections to the server.

반환 형식 float

read_timeout
The maximum allowed waiting time for the first byte of the response from the server.

반환 형식 float

announcement_handler
The announcement handler to display server-set announcements.

반환 형식 Optional[Callable[[str], None]]

4.2.5 커널 함수

class ai.backend.client.kernel.**Kernel**(*kernel_id*, *owner_access_key=None*)
연산 세션들을 관리하면서 다양한 상호작용을 제공합니다.

Backend.AI의 개발이 진행됨에 따라 '커널'이라는 용어는 '연산 세션'이라는 용어로 대체되었지만, 클라이언트 세션과의 혼동을 피하고 과거 코드와의 호환성을 위해 여전히 이 함수 클래스는 커널이라는 이름을 사용하고 있습니다.

여러 개의 다중 컨테이너로 이뤄진 연산 세션에서는, *destroy()* 및 *restart()* 메소드를 제외한 모든 메소드는 세션의 마스터 컨테이너에게만 유효합니다. 따라서 여러 컨테이너가 동일한 데이터를 바라보거나 분산 처리하도록 하기 위해서는 가상폴더 탑재 옵션을 활용하여야 합니다. 현재 이러한 작업은 사용자의 몫입니다. (추후 업데이트로 편의성 개선 예정)

session = None
이 함수 클래스가 사용할 클라이언트 세션 인스턴스

classmethod await get_or_create(*image*, ***, *client_token=None*, *type_='interactive'*, *enqueue_only=False*, *max_wait=0*, *no_reuse=False*, *mounts=None*, *envs=None*, *startup_command=None*, *resources=None*, *resource_opts=None*, *cluster_size=1*, *domain_name=None*, *group_name=None*, *tag=None*, *scaling_group=None*, *owner_access_key=None*)

Get-or-creates a compute session. If *client_token* is `None`, it creates a new compute session as long as the server has enough resources and your API key has remaining quota. If *client_token* is a valid string and there is an existing compute session with the same token and the same *image*, then it returns the *Kernel* instance representing the existing session.

매개변수

- **image** (str) -- The image name and tag for the compute session. Example: `python:3.6-ubuntu`. Check out the full list of available images in your server using (TODO: new API).
- **client_token** (Optional[str]) -- A client-side identifier to seamlessly reuse the compute session already created.
- **type** -- Either `"interactive"` (default) or `"batch"`.

  버전 19.09.0에 추가.
- **enqueue_only** (bool) -- Just enqueue the session creation request and return immediately, without waiting for its startup. (default: `false` to preserve the legacy behavior)

  버전 19.09.0에 추가.
- **max_wait** (int) -- The time to wait for session startup. If the cluster resource is being fully utilized, this waiting time can be arbitrarily long due to job queueing. If the timeout reaches, the returned *status* field becomes `"TIMEOUT"`. Still in this case, the session may start in the future.

  버전 19.09.0에 추가.
- **no_reuse** (bool) -- Raises an explicit error if a session with the same *image* and the same *client_token* already exists instead of returning the information of it.

  버전 19.09.0에 추가.
- **mounts** (Optional[Iterable[str]]) -- The list of vfolder names that belongs to the currrent API access key.
- **envs** (Optional[Mapping[str, str]]) -- The environment variables which always bypasses the jail policy.
- **resources** (Optional[Mapping[str, int]]) -- The resource specification. (TODO: details)
- **cluster_size** (int) -- The number of containers in this compute session. Must be at least 1.

- **tag** (Optional[str]) -- An optional string to annotate extra information.
- **owner** -- An optional access key that owns the created session. (Only available to administrators)

반환 형식 *Kernel*

반환값 The *Kernel* instance.

await destroy(**, forced=False*)

Destroys the compute session. Since the server literally kills the container(s), all ongoing executions are forcibly interrupted.

await restart()

Restarts the compute session. The server force-destroys the current running container(s), but keeps their temporary scratch directories intact.

await interrupt()

Tries to interrupt the current ongoing code execution. This may fail without any explicit errors depending on the code being executed.

await complete(*code*, *opts=None*)

Gets the auto-completion candidates from the given code string, as if a user has pressed the tab key just after the code in IDEs.

Depending on the language of the compute session, this feature may not be supported. Unsupported sessions returns an empty list.

매개변수

- **code** (str) -- An (incomplete) code text.
- **opts** (Optional[dict]) -- Additional information about the current cursor position, such as row, col, line and the remainder text.

반환 형식 Iterable[str]

반환값 An ordered list of strings.

await get_info()

Retrieves a brief information about the compute session.

await get_logs()

Retrieves the console log of the compute session container.

await execute(*run_id=None*, *code=None*, *mode='query'*, *opts=None*)

Executes a code snippet directly in the compute session or sends a set of build/clean/execute commands to the compute session.

For more details about using this API, please refer the official API documentation.

매개변수

- **run_id** (Optional[str]) -- A unique identifier for a particular run loop. In the first call, it may be None so that the server auto-assigns one. Subsequent calls must use the returned runId value to request continuation or to send user inputs.
- **code** (Optional[str]) -- A code snippet as string. In the continuation requests, it must be an empty string. When sending user inputs, this is where the user input string is stored.
- **mode** (str) -- A constant string which is one of "query", "batch", "continue", and "user-input".
- **opts** (Optional[dict]) -- A dict for specifying additional options. Mainly used in the batch mode to specify build/clean/execution commands. See the API object reference for details.

반환값 An execution result object

await upload(*files*, *basedir=None*, *show_progress=False*)

Uploads the given list of files to the compute session. You may refer them in the batch-mode execution or from the code executed in the server afterwards.

매개변수

- **files** (Sequence[Union[str, Path]]) -- The list of file paths in the client-side. If the paths include directories, the location of them in the compute session is calculated from the relative path to *basedir* and all intermediate parent directories are automatically created if not exists.

  For example, if a file path is /home/user/test/data.txt (or test/data.txt) where *basedir* is /home/user (or the current working directory is /home/user), the uploaded file is located at /home/work/test/data.txt in the compute session container.
- **basedir** (Union[str, Path, None]) -- The directory prefix where the files reside. The default value is the current working directory.
- **show_progress** (bool) -- Displays a progress bar during uploads.

await download(*files*, *dest='.'*, *show_progress=False*)

Downloads the given list of files from the compute session.

매개변수

- **files** (Sequence[Union[str, Path]]) -- The list of file paths in the compute session. If they are relative paths, the path is calculated from /home/work in the compute session container.

- **dest** (Union[str, Path]) -- The destination directory in the client-side.
- **show_progress** (bool) -- Displays a progress bar during downloads.

await list_files(*path='.'*)

Gets the list of files in the given path inside the compute session container.

매개변수 **path** (Union[str, Path]) -- The directory path in the compute session.

stream_events()

Opens the stream of the kernel lifecycle events. Only the master kernel of each session is monitored.

반환 형식 SSEResponse

반환값 a StreamEvents object.

stream_pty()

Opens a pseudo-terminal of the kernel (if supported) streamed via websockets.

반환 형식 *StreamPty*

반환값 a *StreamPty* object.

stream_execute(*code=''*, ***, *mode='query'*, *opts=None*)

Executes a code snippet in the streaming mode. Since the returned websocket represents a run loop, there is no need to specify *run_id* explicitly.

반환 형식 *WebSocketResponse*

class ai.backend.client.kernel.**StreamPty**(*session*, *underlying_ws*)

A derivative class of *WebSocketResponse* which provides additional functions to control the terminal.

4.2.6 KeyPair 함수

class ai.backend.client.keypair.**KeyPair**(*access_key*)

Keypair들을 다룹니다.

session = None

이 함수 클래스가 사용할 클라이언트 세션 인스턴스

classmethod await create(*user_id*, *is_active=True*, *is_admin=False*, *resource_policy=None*, *rate_limit=None*, *fields=None*)

Creates a new keypair with the given options. You need an admin privilege for this operation.

반환 형식 dict

classmethod await update(*access_key*, *is_active=None*, *is_admin=None*, *resource_policy=None*, *rate_limit=None*)

Creates a new keypair with the given options. You need an admin privilege for this operation.

반환 형식 dict

classmethod await delete(*access_key*)

Deletes an existing keypair with given ACCESSKEY.

classmethod await list(*user_id=None*, *is_active=None*, *fields=None*)

Lists the keypairs. You need an admin privilege for this operation.

반환 형식 Sequence[dict]

await info(*fields=None*)

Returns the keypair's information such as resource limits.

매개변수 fields (Optional[Iterable[str]]) -- Additional per-agent query fields to fetch.

버전 18.12에 추가.

반환 형식 dict

classmethod await activate(*access_key*)

Activates this keypair. You need an admin privilege for this operation.

반환 형식 dict

classmethod await deactivate(*access_key*)

Deactivates this keypair. Deactivated keypairs cannot make any API requests unless activated again by an administrator. You need an admin privilege for this operation.

반환 형식 dict

4.2.7 매니저 함수

class ai.backend.client.manager.**Manager**

게이트웨이 및 매니저 서버의 상태를 관리합니다.

버전 18.12에 추가.

session = None

이 함수 클래스가 사용할 클라이언트 세션 인스턴스

classmethod await status()

Returns the current status of the configured API server.

classmethod await freeze(*force_kill=False*)

Freezes the configured API server. Any API clients will no longer be able to create new compute sessions nor create and modify vfolders/keypairs/etc. This is used to enter the maintenance mode of the server for unobtrusive manager and/or agent upgrades.

매개변수 **force_kill** (bool) -- If set True, immediately shuts down all running compute sessions forcibly. If not set, clients who have running compute session are still able to interact with them though they cannot create new compute sessions.

classmethod await unfreeze()

Unfreezes the configured API server so that it resumes to normal operation.

classmethod await get_announcement()

Get current announcement.

classmethod await update_announcement(*enabled=True*, *message=None*)

Update (create / delete) announcement.

매개변수

- **enabled** (bool) -- If set False, delete announcement.
- **message** (Optional[str]) -- Announcement message. Required if enabled is True.

4.2.8 가상폴더 함수

class ai.backend.client.vfolder.**VFolder**(*name*)

session = None

이 함수 클래스가 사용할 클라이언트 세션 인스턴스

classmethod await create(*name*, *host=None*, *group=None*)

classmethod await delete_by_id(*oid*)

classmethod await list(*list_all=False*)

classmethod await list_hosts()

classmethod await list_all_hosts()

classmethod await list_allowed_types()

await info()

await delete()

await rename(*new_name*)

await upload(*files*, *basedir=None*, *show_progress=False*)

await mkdir(*path*)

await rename_file(*target_path*, *new_name*)

await delete_files(*files*, *recursive=False*)

await download(*files*, *show_progress=False*)

await list_files(*path='.'*)

await invite(*perm*, *emails*)

classmethod await invitations()

classmethod await accept_invitation(*inv_id*)

classmethod await delete_invitation(*inv_id*)

classmethod await get_fstab_contents(*agent_id=None*)

classmethod await list_mounts()

classmethod await mount_host(*name*, *fs_location*, *options=None*, *edit_fstab=False*)

classmethod await umount_host(*name*, *edit_fstab=False*)

4.2.9 Scaling Group Functions

class ai.backend.client.scaling_group.**ScalingGroup**(*name*)

Provides getting scaling-group information required for the current user.

The scaling-group is an opaque server-side configuration which splits the whole cluster into several partitions, so that server administrators can apply different auto-scaling policies and operation standards to each partition of agent sets.

session = None

The client session instance that this function class is bound to.

classmethod await list_available(*group*)

List available scaling groups for the current user, considering the user, the user's domain, and the designated user group.

classmethod await list(*fields=None*)

List available scaling groups for the current user, considering the user, the user's domain, and the designated user group.

반환 형식 Sequence[dict]

classmethod await detail(*name*, *fields=None*)

Fetch information of a scaling group by name.

매개변수

- **name** (str) -- Name of the scaling group.
- **fields** (Optional[Iterable[str]]) -- Additional per-scaling-group query fields.

반환 형식 Sequence[dict]

classmethod await create(*name*, *description=''*, *is_active=True*, *driver=None*, *driver_opts=None*, *scheduler=None*, *scheduler_opts=None*, *fields=None*)

Creates a new scaling group with the given options.

반환 형식 dict

classmethod await update(*name*, *description=''*, *is_active=True*, *driver=None*, *driver_opts=None*, *scheduler=None*, *scheduler_opts=None*, *fields=None*)

Update existing scaling group.

반환 형식 dict

classmethod await delete(*name*)

Deletes an existing scaling group.

classmethod await associate_domain(*scaling_group*, *domain*)

Associate scaling_group with domain.

매개변수

- **scaling_group** (str) -- The name of a scaling group.
- **domain** (str) -- The name of a domain.

classmethod await dissociate_domain(*scaling_group*, *domain*)

Dissociate scaling_group from domain.

매개변수

- **scaling_group** (str) -- The name of a scaling group.
- **domain** (str) -- The name of a domain.

classmethod await dissociate_all_domain(*domain*)

Dissociate all scaling_groups from domain.

매개변수 **domain** (str) -- The name of a domain.

classmethod await associate_group(*scaling_group*, *group_id*)

Associate scaling_group with group.

매개변수

- **scaling_group** (str) -- The name of a scaling group.

- **group_id** (str) -- The ID of a group.

classmethod await dissociate_group(*scaling_group*, *group_id*)

Dissociate scaling_group from group.

매개변수

- **scaling_group** (str) -- The name of a scaling group.
- **group_id** (str) -- The ID of a group.

classmethod await dissociate_all_group(*group_id*)

Dissociate all scaling_groups from group.

매개변수 **group_id** (str) -- The ID of a group.

4.3 저수준 레퍼런스

4.3.1 기반 함수

This module defines a few utilities that ease complexities to support both synchronous and asynchronous API functions, using some tricks with Python metaclasses.

Unless your are contributing to the client SDK, probably you won't have to use this module directly.

class ai.backend.client.base.**APIFunctionMeta**(*name*, *bases*, *attrs*, ***kwargs*)

Converts all methods marked with *api_function()* into session-aware methods that are either plain Python functions or coroutines.

mro() → list

return a type's method resolution order

class ai.backend.client.base.**BaseFunction**

The class used to build API functions proxies bound to specific session instances.

@ai.backend.client.base.**api_function**(*meth*)

Mark the wrapped method as the API function method.

4.3.2 요청 API

This module provides low-level API request/response interfaces based on aiohttp.

Depending on the session object where the request is made from, *Request* and *Response* differentiate their behavior: works as plain Python functions or returns awaitables.

class ai.backend.client.request.**Request**(*session*, *method='GET'*, *path=None*, *content=None*, ***, *content_type=None*, *params=None*, *reporthook=None*)

The API request object.

with async with fetch(***kwargs*) **as Response**

Sends the request to the server and reads the response.

You may use this method either with plain synchronous Session or AsyncSession. Both the followings patterns are valid:

```
from ai.backend.client.request import Request
from ai.backend.client.session import Session

with Session() as sess:
  rqst = Request(sess, 'GET', ...)
  with rqst.fetch() as resp:
    print(resp.text())
```

```
from ai.backend.client.request import Request
from ai.backend.client.session import AsyncSession

async with AsyncSession() as sess:
  rqst = Request(sess, 'GET', ...)
  async with rqst.fetch() as resp:
    print(await resp.text())
```

반환 형식 *FetchContextManager*

async with connect_websocket(***kwargs*) **as WebSocketResponse or its derivatives**

Creates a WebSocket connection.

경고: This method only works with *AsyncSession*.

반환 형식 *WebSocketContextManager*

content

Retrieves the content in the original form. Private codes should NOT use this as it incurs duplicate encoding/decoding.

반환 형식 Union[bytes, bytearray, str, StreamReader, IOBase, None]

set_content(*value*, ***, *content_type=None*)

Sets the content of the request.

set_json(*value*)

A shortcut for set_content() with JSON objects.

attach_files(*files*)

Attach a list of files represented as AttachedFile.

connect_events(***kwargs*)

Creates a Server-Sent Events connection.

> **경고:** This method only works with *AsyncSession*.

반환 형식 SSEContextManager

class ai.backend.client.request.**Response**(*session*, *underlying_response*, ***, *async_mode=False*)

Represents the Backend.AI API response. Also serves as a high-level wrapper of aiohttp.ClientResponse.

The response objects are meant to be created by the SDK, not the callers.

text(), json() methods return the resolved content directly with plain synchronous Session while they return the coroutines with AsyncSession.

class ai.backend.client.request.**WebSocketResponse**(*session*, *underlying_ws*)

A high-level wrapper of aiohttp.ClientWebSocketResponse.

class ai.backend.client.request.**FetchContextManager**(*session*, *rqst_ctx_builder*, ***, *response_cls=<class 'ai.backend.client.request.Response'>*, *check_status=True*)

The context manager returned by *Request.fetch()*.

It provides both synchronouse and asynchronous contex manager interfaces.

class ai.backend.client.request.**WebSocketContextManager**(*session*, *ws_ctx_builder*, ***, *on_enter=None*, *response_cls=<class 'ai.backend.client.request.WebSocketR*

The context manager returned by *Request.connect_websocket()*.

class ai.backend.client.request.**AttachedFile**(*filename*, *stream*, *content_type*)

A struct that represents an attached file to the API request.

매개변수

- **filename** (*str*) -- The name of file to store. It may include paths and the server will create parent directories if required.
- **stream** (*Any*) -- A file-like object that allows stream-reading bytes.
- **content_type** (*str*) -- The content type for the stream. For arbitrary binary data, use "application/octet-stream".

content_type
Alias for field number 2

count(*value*) → integer -- return number of occurrences of value

filename
Alias for field number 0

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.
Raises ValueError if the value is not present.

stream
Alias for field number 1

4.3.3 예외 클래스들

class ai.backend.client.exceptions.**BackendError**
Exception type to catch all ai.backend-related errors.

with_traceback()
Exception.with_traceback(tb) -- set self.__traceback__ to tb and return self.

class ai.backend.client.exceptions.**BackendAPIError**(*status, reason, data*)
Exceptions returned by the API gateway.

with_traceback()
Exception.with_traceback(tb) -- set self.__traceback__ to tb and return self.

class ai.backend.client.exceptions.**BackendClientError**
Exceptions from the client library, such as argument validation errors.

with_traceback()
Exception.with_traceback(tb) -- set self.__traceback__ to tb and return self.

CHAPTER 5

색인과 표

- genindex
- modindex
- search

Python 모듈 목록

a

ai.backend.client.admin, 28
ai.backend.client.agent, 29
ai.backend.client.auth, 29
ai.backend.client.base, 42
ai.backend.client.config, 30
ai.backend.client.exceptions, 45
ai.backend.client.kernel, 33
ai.backend.client.keypair, 37
ai.backend.client.manager, 38
ai.backend.client.request, 42
ai.backend.client.scaling_group, 40
ai.backend.client.session, 19
ai.backend.client.vfolder, 39

색인

A

accept_invitation() (*ai.backend.client.vfolder.VFolder의 클래스 메서드*), 40
access_key() (*ai.backend.client.config.APIConfig property*), 32
activate() (*ai.backend.client.keypair.KeyPair의 클래스 메서드*), 38
Admin (*ai.backend.client.admin 클래스*), 28
Admin (*ai.backend.client.session.AsyncSession의 속성*), 21
Admin (*ai.backend.client.session.Session의 속성*), 20
Agent (*ai.backend.client.agent 클래스*), 29
Agent (*ai.backend.client.session.AsyncSession의 속성*), 21
Agent (*ai.backend.client.session.Session의 속성*), 20
AgentWatcher (*ai.backend.client.session.AsyncSession의 속성*), 21
AgentWatcher (*ai.backend.client.session.Session의 속성*), 20
ai.backend.client.admin (*모듈*), 28
ai.backend.client.agent (*모듈*), 29
ai.backend.client.auth (*모듈*), 29
ai.backend.client.base (*모듈*), 42
ai.backend.client.config (*모듈*), 30
ai.backend.client.exceptions (*모듈*), 45
ai.backend.client.kernel (*모듈*), 33
ai.backend.client.keypair (*모듈*), 37
ai.backend.client.manager (*모듈*), 38
ai.backend.client.request (*모듈*), 42
ai.backend.client.scaling_group (*모듈*), 40
ai.backend.client.session (*모듈*), 19
ai.backend.client.vfolder (*모듈*), 39
announcement_handler() (*ai.backend.client.config.APIConfig property*), 33
api_function() (*ai.backend.client.base 모듈*), 42
APIConfig (*ai.backend.client.config 클래스*), 30
APIFunctionMeta (*ai.backend.client.base 클래스*), 42
associate_domain() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 41
associate_group() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 41
AsyncSession (*ai.backend.client.session 클래스*), 21
attach_files() (*ai.backend.client.request.Request 메서드*), 44
AttachedFile (*ai.backend.client.request 클래스*), 44
Auth (*ai.backend.client.auth 클래스*), 29
Auth (*ai.backend.client.session.AsyncSession의 속성*), 21
Auth (*ai.backend.client.session.Session의 속*

성), 20

B

BackendAPIError (*ai.backend.client.exceptions* 클래스), 45
BackendClientError (*ai.backend.client.exceptions* 클래스), 45
BackendError (*ai.backend.client.exceptions* 클래스), 45
BaseFunction (*ai.backend.client.base* 클래스), 42
BaseSession (*ai.backend.client.session* 클래스), 19

C

close() (*ai.backend.client.session.AsyncSession* 메서드), 22
close() (*ai.backend.client.session.BaseSession* 메서드), 19
close() (*ai.backend.client.session.Session* 메서드), 21
closed() (*ai.backend.client.session.AsyncSession* *property*), 21
closed() (*ai.backend.client.session.BaseSession* *property*), 19
closed() (*ai.backend.client.session.Session* *property*), 21
complete() (*ai.backend.client.kernel.Kernel* 메서드), 35
config() (*ai.backend.client.session.AsyncSession* *property*), 21
config() (*ai.backend.client.session.BaseSession* *property*), 20
config() (*ai.backend.client.session.Session* *property*), 21
connect_events() (*ai.backend.client.request.Request* 메서드), 44
connect_websocket() (*ai.backend.client.request.Request* 메서드), 43
connection_timeout() (*ai.backend.client.config.APIConfig* *property*), 33
content() (*ai.backend.client.request.Request* *property*), 43
content_type() (*ai.backend.client.request.AttachedFile* *property*), 45
count() (*ai.backend.client.request.AttachedFile* 메서드), 45
create() (*ai.backend.client.keypair.KeyPair* 의 클래스 메서드), 37
create() (*ai.backend.client.scaling_group.ScalingGroup* 의 클래스 메서드), 41
create() (*ai.backend.client.vfolder.VFolder* 의 클래스 메서드), 39

D

deactivate() (*ai.backend.client.keypair.KeyPair* 의 클래스 메서드), 38
DEFAULTS (*ai.backend.client.config.APIConfig* 의 속성), 31
delete() (*ai.backend.client.keypair.KeyPair* 의 클래스 메서드), 38
delete() (*ai.backend.client.scaling_group.ScalingGroup* 의 클래스 메서드), 41
delete() (*ai.backend.client.vfolder.VFolder* 메서드), 39
delete_by_id() (*ai.backend.client.vfolder.VFolder*의 클래스 메서드), 39
delete_files() (*ai.backend.client.vfolder.VFolder* 메서드), 40
delete_invitation() (*ai.backend.client.vfolder.VFolder*의 클래스 메서드), 40
destroy() (*ai.backend.client.kernel.Kernel* 메서드), 35
detail() (*ai.backend.client.agent.Agent*의 클래스 메서드), 30
detail() (*ai.backend.client.scaling_group.ScalingGroup* 의 클래스 메서드), 40

dissociate_all_domain() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 41
dissociate_all_group() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 42
dissociate_domain() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 41
dissociate_group() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 42
Domain (*ai.backend.client.session.Session의 속성*), 20
domain() (*ai.backend.client.config.APIConfig property*), 32
download() (*ai.backend.client.kernel.Kernel 메서드*), 36
download() (*ai.backend.client.vfolder.VFolder 메서드*), 40

E

endpoint() (*ai.backend.client.config.APIConfig property*), 31
endpoint_type() (*ai.backend.client.config.APIConfig property*), 32
endpoints() (*ai.backend.client.config.APIConfig property*), 32
EtcdConfig (*ai.backend.client.session.AsyncSession의 속성*), 21
EtcdConfig (*ai.backend.client.session.Session의 속성*), 20
execute() (*ai.backend.client.kernel.Kernel 메서드*), 35

F

fetch() (*ai.backend.client.request.Request 메서드*), 43
FetchContextManager (*ai.backend.client.request 클래스*), 44
filename() (*ai.backend.client.request.AttachedFile property*), 45
freeze() (*ai.backend.client.manager.Manager의 클래스 메서드*), 38

G

get_announcement() (*ai.backend.client.manager.Manager의 클래스 메서드*), 39
get_config() (*ai.backend.client.config 모듈*), 30
get_env() (*ai.backend.client.config 모듈*), 30
get_fstab_contents() (*ai.backend.client.vfolder.VFolder의 클래스 메서드*), 40
get_info() (*ai.backend.client.kernel.Kernel 메서드*), 35
get_logs() (*ai.backend.client.kernel.Kernel 메서드*), 35
get_or_create() (*ai.backend.client.kernel.Kernel의 클래스 메서드*), 33
Group (*ai.backend.client.session.AsyncSession의 속성*), 22
Group (*ai.backend.client.session.Session의 속성*), 20
group() (*ai.backend.client.config.APIConfig property*), 32

H

hash_type() (*ai.backend.client.config.APIConfig property*), 32

I

Image (*ai.backend.client.session.AsyncSession의 속성*), 22
Image (*ai.backend.client.session.Session의 속성*), 20
index() (*ai.backend.client.request.AttachedFile 메서드*), 45
info() (*ai.backend.client.keypair.KeyPair 메서드*), 38

info() (*ai.backend.client.vfolder.VFolder 메서드*), 39
interrupt() (*ai.backend.client.kernel.Kernel 메서드*), 35
invitations() (*ai.backend.client.vfolder.VFolder 의 클래스 메서드*), 40
invite() (*ai.backend.client.vfolder.VFolder 메서드*), 40

K

Kernel (*ai.backend.client.kernel 클래스*), 33
Kernel (*ai.backend.client.session.AsyncSession 의 속성*), 22
Kernel (*ai.backend.client.session.Session의 속성*), 20
KeyPair (*ai.backend.client.keypair 클래스*), 37
KeyPair (*ai.backend.client.session.AsyncSession 의 속성*), 22
KeyPair (*ai.backend.client.session.Session 의 속성*), 20
KeypairResourcePolicy (*ai.backend.client.session.AsyncSession 의 속성*), 22
KeypairResourcePolicy (*ai.backend.client.session.Session의 속성*), 20

L

list() (*ai.backend.client.keypair.KeyPair의 클래스 메서드*), 38
list() (*ai.backend.client.scaling_group.ScalingGroup 의 클래스 메서드*), 40
list() (*ai.backend.client.vfolder.VFolder의 클래스 메서드*), 39
list_all_hosts() (*ai.backend.client.vfolder.VFolder의 클래스 메서드*), 39
list_allowed_types() (*ai.backend.client.vfolder.VFolder의 클래스 메서드*), 39
list_available() (*ai.backend.client.scaling_group.ScalingGroup 의 클래스 메서드*), 40
list_files() (*ai.backend.client.kernel.Kernel 메서드*), 37
list_files() (*ai.backend.client.vfolder.VFolder 메서드*), 40
list_hosts() (*ai.backend.client.vfolder.VFolder 의 클래스 메서드*), 39
list_mounts() (*ai.backend.client.vfolder.VFolder 의 클래스 메서드*), 40
list_with_limit() (*ai.backend.client.agent.Agent의 클래스 메서드*), 29
login() (*ai.backend.client.auth.Auth의 클래스 메서드*), 29
logout() (*ai.backend.client.auth.Auth의 클래스 메서드*), 29

M

Manager (*ai.backend.client.manager 클래스*), 38
Manager (*ai.backend.client.session.AsyncSession 의 속성*), 22
Manager (*ai.backend.client.session.Session 의 속성*), 20
mkdir() (*ai.backend.client.vfolder.VFolder 메서드*), 40
mount_host() (*ai.backend.client.vfolder.VFolder 의 클래스 메서드*), 40
mro() (*ai.backend.client.base.APIFunctionMeta 메서드*), 42

Q

query() (*ai.backend.client.admin.Admin의 클래스 메서드*), 28

R

read_timeout() (*ai.backend.client.config.APIConfig property*), 33
rename() (*ai.backend.client.vfolder.VFolder 메서드*), 39
rename_file() (*ai.backend.client.vfolder.VFolder 메서드*), 40

Request (*ai.backend.client.request 클래스*), 42
Resource (*ai.backend.client.session.AsyncSession의 속성*), 22
Resource (*ai.backend.client.session.Session의 속성*), 20
Response (*ai.backend.client.request 클래스*), 44
restart() (*ai.backend.client.kernel.Kernel 메서드*), 35

S

ScalingGroup (*ai.backend.client.scaling_group 클래스*), 40
ScalingGroup (*ai.backend.client.session.AsyncSession의 속성*), 22
ScalingGroup (*ai.backend.client.session.Session의 속성*), 21
secret_key() (*ai.backend.client.config.APIConfig property*), 32
session (*ai.backend.client.admin.Admin의 속성*), 28
session (*ai.backend.client.agent.Agent의 속성*), 29
session (*ai.backend.client.kernel.Kernel의 속성*), 33
session (*ai.backend.client.keypair.KeyPair의 속성*), 37
session (*ai.backend.client.manager.Manager의 속성*), 38
session (*ai.backend.client.scaling_group.ScalingGroup의 속성*), 40
Session (*ai.backend.client.session 클래스*), 20
session (*ai.backend.client.vfolder.VFolder의 속성*), 39
set_config() (*ai.backend.client.config 모듈*), 30
set_content() (*ai.backend.client.request.Request 메서드*), 43
set_json() (*ai.backend.client.request.Request 메서드*), 43
skip_sslcert_validation() (*ai.backend.client.config.APIConfig property*), 32
status() (*ai.backend.client.manager.Manager의 클래스 메서드*), 38
stream() (*ai.backend.client.request.AttachedFile property*), 45
stream_events() (*ai.backend.client.kernel.Kernel 메서드*), 37
stream_execute() (*ai.backend.client.kernel.Kernel 메서드*), 37
stream_pty() (*ai.backend.client.kernel.Kernel 메서드*), 37
StreamPty (*ai.backend.client.kernel 클래스*), 37

U

umount_host() (*ai.backend.client.vfolder.VFolder의 클래스 메서드*), 40
unfreeze() (*ai.backend.client.manager.Manager의 클래스 메서드*), 39
update() (*ai.backend.client.keypair.KeyPair의 클래스 메서드*), 38
update() (*ai.backend.client.scaling_group.ScalingGroup의 클래스 메서드*), 41
update_announcement() (*ai.backend.client.manager.Manager의 클래스 메서드*), 39
update_password() (*ai.backend.client.auth.Auth의 클래스 메서드*), 29
upload() (*ai.backend.client.kernel.Kernel 메서드*), 36
upload() (*ai.backend.client.vfolder.VFolder 메서드*), 39
User (*ai.backend.client.session.AsyncSession의 속성*), 22
user (*ai.backend.client.session.Session의 속성*), 20
user_agent() (*ai.backend.client.config.APIConfig property*), 32

V

version() (*ai.backend.client.config.APIConfig property*), 32
VFolder (*ai.backend.client.session.AsyncSession 의 속성*), 22
VFolder (*ai.backend.client.session.Session 의 속성*), 21
VFolder (*ai.backend.client.vfolder 클래스*), 39
vfolder_mounts() (*ai.backend.client.config.APIConfig property*), 32

W

WebSocketContextManager (*ai.backend.client.request 클래스*), 44
WebSocketResponse (*ai.backend.client.request 클래스*), 44
with_traceback() (*ai.backend.client.exceptions.BackendAPIError 메서드*), 45
with_traceback() (*ai.backend.client.exceptions.BackendClientError 메서드*), 45
with_traceback() (*ai.backend.client.exceptions.BackendError 메서드*), 45
worker_thread() (*ai.backend.client.session.Session property*), 21