
Backend.AI Client SDK for Python Documentation

출시 버전 **22.03.2**

Lablup Inc.

2023년 03월 25일

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	13
3.3.1 Starting a session and connecting to its Jupyter Notebook	13
3.3.2 Accessing sessions via a web terminal	13
3.3.3 Accessing sessions via native SSH/SFTP	14
3.4 Storage Management	15
3.4.1 Creating vfolders and managing them	15
3.4.2 File transfers and management	16
3.4.3 Running sessions with storages	16
3.4.4 Creating default files for kernels	17
3.5 코드 실행하기 (고급)	17

3.5.1	동시 세션 실행하기	17
3.6	Session Templates	18
3.6.1	Creating and starting session template	18
3.6.2	Full syntax for task template	19
4	Developer Reference	21
4.1	Developer Guides	21
4.1.1	Client Session	21
4.1.2	Examples	23
Synchronous-mode execution		23
Asynchronous-mode Execution		26
4.1.3	Testing	27
Unit Tests		27
Integration Tests		28
4.2	고수준 함수 인터페이스	29
4.2.1	관리자용 함수	29
4.2.2	에이전트 함수	30
4.2.3	Auth Functions	31
4.2.4	Configuration	32
4.2.5	KeyPair 함수	35
4.2.6	매니저 함수	39
4.2.7	Scaling Group Functions	41
4.2.8	ComputeSession Functions	45
4.2.9	Session Template Functions	50
4.2.10	가상폴더 함수	51
4.3	저수준 레퍼런스	58
4.3.1	기반 함수	58
4.3.2	요청 API	58
4.3.3	예외 클래스들	61
4.3.4	Miscellaneous Utilities	61
5	색인과 표	63
	Python 모듈 목록	65
	색인	67

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](#).

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.

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 offer 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.
<code>--running</code>	PREPARING, PULLING, and RUNNING.
<code>--dead</code>	CANCELLED and TERMINATED.

Both commands offer options to specify which fields of sessions should be printed as follows.

Option	Included Session Fields
(no option)	Session ID, Owner, Image, Type, Status, Status Info, Last updated, and Result.
--id-only	Session ID.
--detail	Session ID, Owner, Image, Type, Status, Status Info, Last updated, Result, Tag, Created At, Occupied Resource, Used Memory (MiB), Max Used Memory (MiB), and CPU Using (%).
-f, --format	Specified fields by user.

참고: Fields for -f/--format option can be displayed by specifying comma-separated parameters.

Available parameters for this option are: id, status, status_info, created_at, last_updated, result, image, type, task_id, tag, occupied_slots, used_memory, max_used_memory, cpu_using.

For example:

```
backend.ai admin session --format id,status,cpu_using
```

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 specify 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 Backend.AI sessions expose an intrinsic application named "sshd" like "ttypd".

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
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 you 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 folder 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 completely:

```
$ 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. If you want to mount vfolders in other path, add '/' as prefix at the forefront of the vfolder path.

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.

3.4.4 Creating default files for kernels

Backend.AI has a feature called 'dotfile', created to all the kernels user spawns. As you can guess, dotfile's path should start with `..`. The following command creates dotfile named `.aws/config` with permission 755. This file will be created under `/home/work` every time user spawns Backend.AI kernel.

```
$ backend.ai dotfile create .aws/config < ~/.aws/config
```

3.5 코드 실행하기 (고급)

참고: 각 명령어의 help에서 자세한 사용법을 확인할 것을 권장합니다. help는 `-h` 혹은 `--help` 를 명령어의 인자로 입력하여 불러올 수 있습니다.

3.5.1 동시 세션 실행하기

`run` 명령어는 *Running simple sessions* 에 설명되어 있는 단일 세션 실행뿐만 아니라 여러 세션의 동시 실행 기능을 제공하고, 이 때 `--exec` 옵션으로 입력되는 인수와 `-e / --env` 옵션으로 입력되는 환경 변수가 사용됩니다.

`--exec` 옵션에 해당하는 변수를 설정할 때에는 `--exec-range` 이 사용되고, `--env` 옵션에 해당하는 변수를 설정할 때에는 `--env-range` 이 사용됩니다.

아래는 4개의 세션을 생성하도록 환경 변수의 범위를 설정하는 예시입니다.

```
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
```

`range` 옵션은 “range expressions” 라는 형식의 인자를 입력 받습니다. `range` 옵션의 앞 부분은 해당 변수와 등호(=)로 이루어져 있습니다. 뒷 부분은 아래에 나열된 종류의 형식의 식입니다.

식 (Expression)	설명
<code>case:CASE1, CASE2, ..., CASEN</code>	문자열 또는 숫자로 구성된 배열
<code>linspace:START, STOP, POINTS</code>	<code>numpy.linspace()</code> 와 같은 문법으로 정의되는 구간의 값들. 예를 들어 <code>linspace:1,2,3</code> 은 길이가 3인 배열 <code>[1, 1.5, 2]</code> 을 생성합니다.
<code>range:START, STOP, STEP</code>	파이썬의 <code>range()</code> 와 같은 문법으로 정의되는 수의 범위 (range). 예를 들어 <code>range:1,6,2</code> 은 배열 <code>[1, 3, 5]</code> 을 생성합니다.

run 명령어에 여러 개의 range 옵션을 입력하면 클라이언트는 각 범위로 정의된 값들의 모든 조합에 해당하는 개수의 세션을 생성합니다.

참고: 생성된 세션을 실행하기 위한 사용자 혹은 클러스터의 자원이 부족하면 일부 세션은 큐에 추가되어 커맨드가 실행되는 데에 오랜 시간이 걸릴 수 있습니다.

경고: 위의 기능은 클라이언트에서 구현되었기 때문에 모든 케이스가 완료 될 때까지 클라이언트는 서버와의 연결을 유지해야 합니다. 서버의 batch job 스케줄링은 현재 개발 중입니다!

3.6 Session Templates

3.6.1 Creating and starting session template

Users may define commonly used set of session creation parameters as reusable templates.

A session template includes common session parameters such as resource slots, vfolder mounts, the kernel image to use, and etc. It also support an extra feature that automatically clones a Git repository upon startup as a bootstrap command.

The following sample shows how a session template looks like:

```
---
api_version: v1
kind: taskTemplate
metadata:
  name: template1234
  tag: example-tag
spec:
  kernel:
    environ:
      MYCONFIG: XXX
    git:
      branch: '19.09'
      commit: 10daee9e328876d75e6d0fa4998d4456711730db
      repository: https://github.com/lablup/backend.ai-agent
      destinationDir: /home/work/baiaagent
    image: python:3.6-ubuntu18.04
  resources:
    cpu: '2'
    mem: 4g
  mounts:
    hostpath-test: /home/work/hostDesktop
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
test-vfolder:
sessionType: interactive
```

The backend.ai sesstpl command set provides the basic CRUD operations of user-specific session templates.

The create command accepts the YAML content either piped from the standard input or read from a file using -f flag:

```
$ backend.ai sesstpl create < session-template.yaml
# -- or --
$ backend.ai sesstpl create -f session-template.yaml
```

Once the session template is uploaded, you may use it to start a new session:

```
$ backend.ai start-template <templateId>
```

with substituting <templateId> to your template ID.

Other CRUD command examples are as follows:

```
$ backend.ai sesstpl update <templateId> < session-template.yaml
$ backend.ai sesstpl list
$ backend.ai sesstpl get <templateId>
$ backend.ai sesstpl delete <templateId>
```

3.6.2 Full syntax for task template

```
---
api_version or apiVersion: str, required
kind: Enum['taskTemplate', 'task_template'], required
metadata: required
  name: str, required
  tag: str (optional)
spec:
  type or sessionType: Enum['interactive', 'batch'] (optional), default=interactive
  kernel:
    image: str, required
    environ: map[str, str] (optional)
    run: (optional)
      bootstrap: str (optional)
      stratup or startup_command or startupCommand: str (optional)
    git: (optional)
      repository: str, required
      commit: str (optional)
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
branch: str (optional)
credential: (optional)
    username: str
    password: str
destination_dir or destinationDir: str (optional)
mounts: map[str, str] (optional)
resources: map[str, str] (optional)
```


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,  
                                             proxy_mode=False)
```

The base abstract class for sessions.

proxy_mode

If set True, it skips API version negotiation when opening the session.

반환 형식 `bool`

abstractmethod open()

Initializes the session and perform version negotiation.

반환 형식 `Optional[Awaitable[None]]`

abstractmethod close()

Terminates the session and releases underlying resources.

반환 형식 `Optional[Awaitable[None]]`

closed

Checks if the session is closed.

반환 형식 `bool`

config

The configuration used by this session object.

반환 형식 `APIConfig`

class ai.backend.client.session.Session(*, config=None, proxy_mode=False)

A context manager for API client sessions that makes API requests synchronously. You may call simple request-response APIs like a plain Python function, but cannot use streaming APIs based on WebSocket and Server-Sent Events.

closed

Checks if the session is closed.

반환 형식 `bool`

config

The configuration used by this session object.

반환 형식 `APIConfig`

proxy_mode

If set True, it skips API version negotiation when opening the session.

반환 형식 `bool`

open()

Initializes the session and perform version negotiation.

반환 형식 `None`

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.

반환 형식 `None`

worker_thread

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

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

A context manager for API client sessions that makes API requests asynchronously. You may call all APIs as coroutines. WebSocket-based APIs and SSE-based APIs returns special response types.

closed

Checks if the session is closed.

반환 형식 `bool`

config

The configuration used by this session object.

반환 형식 `APIConfig`

proxy_mode

If set True, it skips API version negotiation when opening the session.

반환 형식 `bool`

open()

Initializes the session and perform version negotiation.

반환 형식 `Awaitable[None]`

close()

Terminates the session and releases underlying resources.

반환 형식 `Awaitable[None]`

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.ComputeSession.get_or_create('python:3.6-ubuntu18.04')
    code = 'print("hello world")'
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```

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.ComputeSession.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)
        opts.clear()

```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```

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.ComputeSession.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.func.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.

classmethod await query()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

class C: @classmethod def f(cls, arg1, arg2, ...):

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

4.2.2 에이전트 함수

class ai.backend.client.func.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.

classmethod await paginated_list()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await detail()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

4.2.3 Auth Functions

class ai.backend.client.func.auth.Auth

Provides the function interface for login session management and authorization.

classmethod await login()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await logout()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await update_password()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```


It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

4.2.4 Configuration

`ai.backend.client.config.get_env(key, default=<Undefined.token: <object object>>, *, clean=<function default_clean>)`

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** (`Union[str, Mapping, Undefined]`) – The default value returned when there is no corresponding environment variable.
- **clean** (`Callable[[Any], ~T]`) – 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.

반환 형식 ~T

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, storage_proxy_address_map=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[URL, str, 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 fail-over 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: `Mapping[str, Union[str, Mapping]] = {'connection_timeout': '10.0', 'domain'`

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`

storage_proxy_address_map

The storage proxy address map for overriding.

반환 형식 `Mapping[str, 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 `Sequence[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 KeyPair 함수

class `ai.backend.client.func.keypair.KeyPair(access_key)`

Keypair들을 다룹니다.

classmethod `await create()`

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:


```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await update()

`classmethod(function)` -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await delete()

`classmethod(function)` -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await list()

`classmethod(function)` -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await paginated_list()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

```
await info(fields=(FieldSpec(field_ref='user_id',    humanized_name='Email',
                             field_name='user_id',    alt_name='user_id',    format-
                             ter=<ai.backend.client.output.formatters.OutputFormatter
                             object>,    subfields={})),    FieldSpec(field_ref='access_key',
                             humanized_name='Access    Key',    field_name='access_key',
                             alt_name='access_key', formatter=<ai.backend.client.output.formatters.OutputForma
                             object>,    subfields={})),    FieldSpec(field_ref='secret_key',
                             humanized_name='Secret    Key',    field_name='secret_key',
                             alt_name='secret_key', formatter=<ai.backend.client.output.formatters.OutputForma
                             object>,    subfields={})),    FieldSpec(field_ref='is_active',    human-
                             ized_name='Active?',    field_name='is_active',    alt_name='is_active',
                             formatter=<ai.backend.client.output.formatters.OutputFormatter
                             object>,    subfields={})),    FieldSpec(field_ref='is_admin',    human-
                             ized_name='Admin?',    field_name='is_admin',    alt_name='is_admin',
                             formatter=<ai.backend.client.output.formatters.OutputFormatter
                             object>,    subfields={})))
```

Returns the keypair's information such as resource limits.

매개변수 **fields** (`Sequence[FieldSpec]`) – Additional per-agent query fields to fetch.

버전 18.12에 추가.

반환 형식 `dict`

classmethod await activate()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await deactivate()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

4.2.6 매니저 함수

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

게이트웨이 및 매니저 서버의 상태를 관리합니다.

버전 18.12에 추가.

classmethod **await status()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod **await freeze()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod **await unfreeze()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```


...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await get_announcement()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await update_announcement()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await scheduler_op()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

4.2.7 Scaling Group Functions

class ai.backend.client.func.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.

classmethod **await list_available()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod **await list()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```


...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await detail()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await create()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await update()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await delete()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await associate_domain()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await dissociate_domain()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await dissociate_all_domain()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await associate_group()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

`classmethod await dissociate_group()`

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

`classmethod await dissociate_all_group()`

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

4.2.8 ComputeSession Functions

```
class ai.backend.client.func.session.ComputeSession(name,  
                                                    owner_access_key=None)
```

Provides various interactions with compute sessions in Backend.AI.

The term 'kernel' is now deprecated and we prefer 'compute sessions'. However, for historical reasons and to avoid confusion with client sessions, we keep the backward compatibility with the naming of this API function class.

For multi-container sessions, all methods take effects to the master container only, except `destroy()` and `restart()` methods. So it is the user's responsibility to distribute uploaded files to multiple containers using explicit copies or virtual folders which are commonly mounted to all containers belonging to the same compute session.

classmethod await paginated_list()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await hello()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod async for ... in get_task_logs()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```


...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await get_or_create()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await create_from_template()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

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 rename(*new_id*)

Renames Session ID of running compute session.

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.

listen_events(*scope='*'*)

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

반환 형식 `SSEContextManager`

반환값 a `StreamEvents` object.

stream_events(*scope='*'*)

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

반환 형식 `SSEContextManager`

반환값 a `StreamEvents` object.

stream_ptty()

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

반환 형식 `WebSocketContextManager`

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 `WebSocketContextManager`

class ai.backend.client.func.session.**StreamPty**(*session,* *underlying_response, **kwargs*)

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

4.2.9 Session Template Functions

class ai.backend.client.func.session_template.**SessionTemplate**(*template_id,* *owner_access_key=None*)

classmethod **await create**()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

class C: @classmethod def f(cls, arg1, arg2, ...):

...

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod **await** **list_templates()**

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
    ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

await **get**(*body_format*='yaml')

반환 형식 `str`

await **put**(*template*)

반환 형식 `Any`

await **delete**()

반환 형식 `Any`

4.2.10 가상폴더 함수

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

classmethod **await** **create**()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```


...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await delete_by_id()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await list()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

classmethod await paginated_list()

`classmethod(function) -> method`

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await list_hosts()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await list_all_hosts()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.


```
classmethod await list_allowed_types()
```

```
    classmethod(function) -> method
```

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
    class C: @classmethod def f(cls, arg1, arg2, ...):
```

```
        ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

```
await info()
```

```
await delete()
```

```
await rename(new_name)
```

```
await download(relative_paths, *, basedir=None, chunk_size=16777216,  
               show_progress=False, address_map=None)
```

```
    반환 형식 None
```

```
await upload(files, *, basedir=None, chunk_size=16777216, address_map=None,  
             show_progress=False)
```

```
    반환 형식 None
```

```
await mkdir(path)
```

```
await rename_file(target_path, new_name)
```

```
await move_file(src_path, dst_path)
```

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

```
await list_files(path='.')
```

```
await invite(perm, emails)
```

```
classmethod await invitations()
```

```
    classmethod(function) -> method
```

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
    class C: @classmethod def f(cls, arg1, arg2, ...):
```


...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await accept_invitation()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await delete_invitation()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
```

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await get_fstab_contents()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await get_performance_metric()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod await list_mounts()

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod **await** **mount_host()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

classmethod **await** **umount_host()**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):  
    ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

await **share**(*perm*, *emails*)

await **unshare**(*emails*)

await **leave**()

await **clone**(*target_name*, *target_host*=None, *usage_mode*='general', *permission*='rw')

await **update_options**(*name*, *permission*=None, *cloneable*=None)

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.

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(method='GET', path=None, content=None, *, content_type=None, params=None, reporthook=None, override_api_version=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('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('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.

반환 형식 *None*

set_json(value)

A shortcut for set_content() with JSON objects.

반환 형식 *None*

attach_files(files)

Attach a list of files represented as AttachedFile.

반환 형식 *None*

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, **kwargs)
```

```
class ai.backend.client.request.WebSocketResponse(session, underlying-
                                                    ing_response, **kwargs)
    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 synchronous and asynchronous context manager interfaces.

```
class ai.backend.client.request.WebSocketContextManager(session,  
                                                    ws_ctx_builder,  
                                                    *, on_enter=None,  
                                                    re-  
                                                    sponse_cls=<class  
                                                    'ai.backend.client.request.WebSocketF
```

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

```
class ai.backend.client.request.AttachedFile(filename,      stream,      con-  
                                                    tent_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*, /)

Return number of occurrences of value.

filename

Alias for field number 0

index(*value*, *start*=0, *stop*=9223372036854775807, /)

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 and connection failures.

with_traceback()

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

4.3.4 Miscellaneous Utilities

ai.backend.client.utils.**undefined**

A placeholder to signify an undefined value as a singleton object of Undefined and distinguish it from a null (None) value.

- genindex
- modindex
- search

a

`ai.backend.client.base`, 58
`ai.backend.client.config`, 32
`ai.backend.client.exceptions`, 61
`ai.backend.client.func.admin`, 29
`ai.backend.client.func.agent`, 30
`ai.backend.client.func.auth`, 31
`ai.backend.client.func.keypair`, 35
`ai.backend.client.func.manager`, 39
`ai.backend.client.func.scaling_group`,
41
`ai.backend.client.func.session`, 45
`ai.backend.client.func.session_template`,
50
`ai.backend.client.func.vfolder`, 51
`ai.backend.client.request`, 58
`ai.backend.client.session`, 21
`ai.backend.client.utils`, 61

A

- `accept_invitation()`
 - (`ai.backend.client.func.vfolder.VFolder` 모듈, 50
 - 의 클래스 메서드), 55
- `access_key()` (`ai.backend.client.config.APIConfig` 모듈, 51
- `property`), 34
- `activate()` (`ai.backend.client.func.keypair.KeyPair` 모듈, 58
- 의 클래스 메서드), 38
- `Admin` (`ai.backend.client.func.admin` 클래스
- 스), 29
- `Agent` (`ai.backend.client.func.agent` 클래스),
- 30
- `ai.backend.client.base`
- 모듈, 58
- `ai.backend.client.config`
- 모듈, 32
- `ai.backend.client.exceptions`
- 모듈, 61
- `ai.backend.client.func.admin`
- 모듈, 29
- `ai.backend.client.func.agent`
- 모듈, 30
- `ai.backend.client.func.auth`
- 모듈, 31
- `ai.backend.client.func.keypair`
- 모듈, 35
- `ai.backend.client.func.manager`
- 모듈, 39
- `ai.backend.client.func.scaling_group`
- 모듈, 41
- `ai.backend.client.func.session`
- 모듈, 45
- `ai.backend.client.func.session_template`
- 모듈, 50
- `ai.backend.client.func.vfolder`
- 모듈, 51
- `ai.backend.client.request`
- 모듈, 58
- `ai.backend.client.session`
- 모듈, 21
- `ai.backend.client.utils`
- 모듈, 61
- `announcement_handler()`
- (`ai.backend.client.config.APIConfig`
- `property`), 35
- `APIConfig` (`ai.backend.client.config` 클래스),
- 32
- `associate_domain()`
- (`ai.backend.client.func.scaling_group.ScalingGroup`
- 의 클래스 메서드), 43
- `associate_group()`
- (`ai.backend.client.func.scaling_group.ScalingGroup`
- 의 클래스 메서드), 44
- `AsyncSession` (`ai.backend.client.session` 클
- 래스), 23
- `attach_files()`
- (`ai.backend.client.request.Request`
- 메서드), 59
- `AttachedFile` (`ai.backend.client.request` 클
- 래스), 60
- `Auth` (`ai.backend.client.func.auth` 클래스),
- 31

B

BackendAPIError

(ai.backend.client.exceptions 클래스), 61

BackendClientError

(ai.backend.client.exceptions 클래스), 61

BackendError (ai.backend.client.exceptions 클래스), 61

BaseSession (ai.backend.client.session 클래스), 21

C

clone() (ai.backend.client.func.vfolder.VFolder 메서드), 57

close() (ai.backend.client.session.AsyncSession 메서드), 23

close() (ai.backend.client.session.BaseSession 메서드), 22

close() (ai.backend.client.session.Session 메서드), 22

closed() (ai.backend.client.session.AsyncSession property), 23

closed() (ai.backend.client.session.BaseSession property), 22

closed() (ai.backend.client.session.Session property), 22

complete() (ai.backend.client.func.session.ComputeSession 메서드), 48

ComputeSession

(ai.backend.client.func.session 클래스), 45

config() (ai.backend.client.session.AsyncSession property), 23

config() (ai.backend.client.session.BaseSession property), 22

config() (ai.backend.client.session.Session property), 22

connect_events()

(ai.backend.client.request.Request 메서드), 59

connect_websocket()

(ai.backend.client.request.Request

메서드), 58

connection_timeout()

(ai.backend.client.config.APIConfig property), 35

content() (ai.backend.client.request.Request property), 59

content_type (ai.backend.client.request.AttachedFile의 속성), 60

count() (ai.backend.client.request.AttachedFile 메서드), 60

create() (ai.backend.client.func.keypair.KeyPair의 클래스 메서드), 35

create() (ai.backend.client.func.scaling_group.ScalingGroup의 클래스 메서드), 42

create() (ai.backend.client.func.session_template.SessionTemplate의 클래스 메서드), 50

create() (ai.backend.client.func.vfolder.VFolder의 클래스 메서드), 51

create_from_template()

(ai.backend.client.func.session.ComputeSession의 클래스 메서드), 47

D

deactivate() (ai.backend.client.func.keypair.KeyPair의 클래스 메서드), 38

DEFAULTS (ai.backend.client.config.APIConfig의 속성), 34

delete() (ai.backend.client.func.keypair.KeyPair의 클래스 메서드), 36

delete() (ai.backend.client.func.scaling_group.ScalingGroup의 클래스 메서드), 43

delete() (ai.backend.client.func.session_template.SessionTemplate의 클래스 메서드), 51

delete() (ai.backend.client.func.vfolder.VFolder의 클래스 메서드), 54

delete_by_id()

(ai.backend.client.func.vfolder.VFolder의 클래스 메서드), 52

delete_files()

(ai.backend.client.func.vfolder.VFolder 메서드), 54

delete_invitation()

(ai.backend.client.func.vfolder.VFolder

의 클래스 메서드), 55
destroy() (ai.backend.client.func.session.ComputeSession의 메서드), 60
메서드), 47
freeze() (ai.backend.client.func.manager.Manager의 클래스 메서드), 39
detail() (ai.backend.client.func.agent.Agent의 클래스 메서드), 30
detail() (ai.backend.client.func.scaling_group.GScalingGroup의 클래스 메서드), 42
dissociate_all_domain()
(ai.backend.client.func.scaling_group.GScalingGroup의 클래스 메서드), 44
dissociate_all_group()
(ai.backend.client.func.scaling_group.GScalingGroup의 클래스 메서드), 45
dissociate_domain()
(ai.backend.client.func.scaling_group.GScalingGroup의 클래스 메서드), 43
dissociate_group()
(ai.backend.client.func.scaling_group.GScalingGroup의 클래스 메서드), 45
domain() (ai.backend.client.config.APIConfig property), 34
download() (ai.backend.client.func.session.ComputeSession의 클래스 메서드), 49
download() (ai.backend.client.func.vfolder.VFolder의 클래스 메서드), 54
E
endpoint() (ai.backend.client.config.APIConfig property), 34
endpoint_type()
(ai.backend.client.config.APIConfig property), 34
endpoints() (ai.backend.client.config.APIConfig property), 34
execute() (ai.backend.client.func.session.ComputeSession의 클래스 메서드), 48
F
fetch() (ai.backend.client.request.Request의 클래스 메서드), 58
FetchContextManager
(ai.backend.client.request 클래스), 59
filename (ai.backend.client.request.AttachedFile의 속성), 60
freeze() (ai.backend.client.func.manager.Manager의 클래스 메서드), 39
G
get() (ai.backend.client.func.session_template.SessionTemplate의 클래스 메서드), 51
get_announcement()
(ai.backend.client.func.manager.Manager의 클래스 메서드), 40
get_config() (ai.backend.client.config 모듈), 32
get_env() (ai.backend.client.config 모듈), 32
get_fstab_contents()
(ai.backend.client.func.vfolder.VFolder의 클래스 메서드), 55
get_info() (ai.backend.client.func.session.ComputeSession의 클래스 메서드), 48
get_logs() (ai.backend.client.func.session.ComputeSession의 클래스 메서드), 48
get_or_create()
(ai.backend.client.func.session.ComputeSession의 클래스 메서드), 47
get_performance_metric()
(ai.backend.client.func.vfolder.VFolder의 클래스 메서드), 56
get_task_logs()
(ai.backend.client.func.session.ComputeSession의 클래스 메서드), 46
group() (ai.backend.client.config.APIConfig property), 34
H
hash_type() (ai.backend.client.config.APIConfig property), 35
hello() (ai.backend.client.func.session.ComputeSession의 클래스 메서드), 46
I
index() (ai.backend.client.request.AttachedFile의 클래스 메서드), 60

`info()` (`ai.backend.client.func.keypair.KeyPair` 의 클래스 메서드), 51
`listen_events()` (ai.backend.client.func.vfolder.VFolder (ai.backend.client.func.session.ComputeSession의 메서드), 54 메서드), 49
`interrupt()` (`ai.backend.client.func.session.ComputeSession`의 메서드), 48
`invitations()` (`ai.backend.client.func.vfolder.VFolder` (ai.backend.client.func.auth.Auth의 클래스 메서드), 54 의 클래스 메서드), 31
`invite()` (`ai.backend.client.func.vfolder.VFolder`의 메서드), 54
K
`KeyPair` (`ai.backend.client.func.keypair` 클래스), 35
L
`leave()` (`ai.backend.client.func.vfolder.VFolder`의 메서드), 57
`list()` (`ai.backend.client.func.keypair.KeyPair`의 클래스 메서드), 36
`list()` (`ai.backend.client.func.scaling_group.ScalingGroup`의 클래스 메서드), 41
`list()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 52
`list_all_hosts()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 53
`list_allowed_types()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 53
`list_available()` (`ai.backend.client.func.scaling_group.ScalingGroup`의 클래스 메서드), 41
`list_files()` (`ai.backend.client.func.session.ComputeSession`의 메서드), 49
`list_files()` (`ai.backend.client.func.vfolder.VFolder`의 메서드), 54
`list_hosts()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 53
`list_mounts()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 56
`list_templates()` (`ai.backend.client.session_template.SessionTemplate`의 메서드), 21
`Manager` (`ai.backend.client.func.manager` 클래스), 39
`mkdir()` (`ai.backend.client.func.vfolder.VFolder`의 메서드), 54
`mount_host()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 56
`move_file()` (`ai.backend.client.func.vfolder.VFolder`의 메서드), 54
O
`open()` (`ai.backend.client.session.AsyncSession`의 메서드), 23
`open()` (`ai.backend.client.session.BaseSession`의 메서드), 22
`open()` (`ai.backend.client.session.Session`의 메서드), 22
P
`paginated_list()` (`ai.backend.client.func.agent.Agent`의 클래스 메서드), 30
`paginated_list()` (`ai.backend.client.func.keypair.KeyPair`의 클래스 메서드), 37
`paginated_list()` (`ai.backend.client.func.session.ComputeSession`의 클래스 메서드), 46
`paginated_list()` (`ai.backend.client.func.vfolder.VFolder`의 클래스 메서드), 52
`proxy_mode()` (`ai.backend.client.session.AsyncSession`의 property), 23
`proxy_mode()` (`ai.backend.client.session.BaseSession`의 property), 21

[proxy_mode\(\) \(ai.backend.client.session.Session property\), 22](#)
[put\(\) \(ai.backend.client.func.session_template.SessionTemplate 클래스 메서드\), 51](#)
[put\(\) \(ai.backend.client.func.vfolder.VFolder 클래스 메서드\), 57](#)
[skip_sslcert_validation\(\) \(ai.backend.client.config.APIConfig property\), 35](#)
[status\(\) \(ai.backend.client.func.manager.Manager 클래스 메서드\), 39](#)
[storage_proxy_address_map\(\) \(ai.backend.client.config.APIConfig property\), 34](#)
[stream\(\) \(ai.backend.client.request.AttachedFile의 속성\), 60](#)
[stream_events\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 50](#)
[stream_execute\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 50](#)
[streampty\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 50](#)
[StreamPty \(ai.backend.client.func.session 클래스\), 50](#)
[restart\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 47](#)
[S](#)
[ScalingGroup \(ai.backend.client.func.scaling_group 클래스\), 41](#)
[scheduler_op\(\) \(ai.backend.client.func.manager.Manager 클래스 메서드\), 40](#)
[secret_key\(\) \(ai.backend.client.config.APIConfig property\), 34](#)
[Session \(ai.backend.client.session 클래스\), 22](#)
[SessionTemplate \(ai.backend.client.func.session_template 클래스\), 50](#)
[set_config\(\) \(ai.backend.client.config 모듈\), 32](#)
[set_content\(\) \(ai.backend.client.request.Request 클래스 메서드\), 59](#)
[set_json\(\) \(ai.backend.client.request.Request 클래스 메서드\), 59](#)
[SessionTemplate \(ai.backend.client.func.vfolder.VFolder 클래스 메서드\), 57](#)
[skip_sslcert_validation\(\) \(ai.backend.client.config.APIConfig property\), 35](#)
[status\(\) \(ai.backend.client.func.manager.Manager 클래스 메서드\), 39](#)
[storage_proxy_address_map\(\) \(ai.backend.client.config.APIConfig property\), 34](#)
[stream\(\) \(ai.backend.client.request.AttachedFile의 속성\), 60](#)
[stream_events\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 50](#)
[stream_execute\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 50](#)
[streampty\(\) \(ai.backend.client.func.session.ComputeSession 클래스 메서드\), 50](#)
[StreamPty \(ai.backend.client.func.session 클래스\), 50](#)
[U](#)
[umount_host\(\) \(ai.backend.client.func.vfolder.VFolder 클래스 메서드\), 57](#)
[undefined \(ai.backend.client.utils 모듈\), 61](#)
[unfreeze\(\) \(ai.backend.client.func.manager.Manager 클래스 메서드\), 39](#)
[unshare\(\) \(ai.backend.client.func.vfolder.VFolder 클래스 메서드\), 57](#)
[update\(\) \(ai.backend.client.func.keypair.KeyPair 클래스 메서드\), 36](#)
[update\(\) \(ai.backend.client.func.scaling_group.ScalingGroup 클래스 메서드\), 42](#)
[update_announcement\(\) \(ai.backend.client.func.manager.Manager 클래스 메서드\), 40](#)
[update_options\(\) \(ai.backend.client.func.vfolder.VFolder 클래스 메서드\), 57](#)

`update_password()`
 (`ai.backend.client.func.auth.Auth`의
 클래스 메서드), 31

`upload()` (`ai.backend.client.func.session.ComputedSession`
 메서드), 49

`upload()` (`ai.backend.client.func.vfolder.VFolder`
 메서드), 54

`user_agent()` (`ai.backend.client.config.APIConfig`
 property), 34

V

`version()` (`ai.backend.client.config.APIConfig`
 property), 35

`VFolder` (`ai.backend.client.func.vfolder` 클래스
 스), 51

`vfolder_mounts()`
 (`ai.backend.client.config.APIConfig`
 property), 35

W

`WebSocketContextManager`
 (`ai.backend.client.request` 클래스),
 60

`WebSocketResponse`
 (`ai.backend.client.request` 클래스),
 59

`with_traceback()`
 (`ai.backend.client.exceptions.BackendAPIError`
 메서드), 61

`with_traceback()`
 (`ai.backend.client.exceptions.BackendClientError`
 메서드), 61

`with_traceback()`
 (`ai.backend.client.exceptions.BackendError`
 메서드), 61

`worker_thread()`
 (`ai.backend.client.session.Session`
 property), 22

모듈

`ai.backend.client.base`, 58
`ai.backend.client.config`, 32
`ai.backend.client.exceptions`, 61