Changelog

This changelog documents user-facing updates (features, enhancements, fixes, and deprecations) to the `modal` client library. Patch releases are made on every change.

The client library is still in pre-1.0 development, and sometimes breaking changes are necessary. We try to minimize them and publish deprecation warnings / migration guides in advance, typically providing a transition window of several months.

We appreciate your patience while we speedily work towards a stable release of the client.

Latest

0.62.223 (2024-06-14)

- All modal CLI commands now accept `-e` as a short-form of `--env`

0.62.219 (2024-06-12)

- Fix an issue with `@web_server` decorator not working on image builder version 2023.12

0.62.208 (2024-06-08)

- `@web_server` endpoints can now return HTTP headers of up to 64 KiB in length. Previously, they were limited to 8 KiB due to an implementation detail.

0.62.201 (2024-06-04)

- `modal deploy` now accepts a `--tag` optional parameter that allows you to specify a custom tag for the deployed version, making it easier to identify and manage different deployments of your app.

0.62.199 (2024-06-04)

- `web_endpoint` s now have the option to include interactive SwaggerUI/redoc docs by setting `docs=True`
- `web_endpoint` s no longer include an OpenAPI JSON spec route by default

0.62.197 (2024-05-31)

Adds Source to PyPI metadata

0.62.190 (2024-05-29)

- `modal.Function` now supports requesting ephemeral disk (SSD) via the new `ephemeral_disk` parameter. Intended for use in doing large dataset ingestion and transform.

0.62.186 (2024-05-29)

- `modal.Volume` background commits are now enabled by default when using `spawn_sandbox` .

0.62.185 (2024-05-28)

- The `modal app stop` CLI command now accepts a `--name` (or `-n`) option to stop an App by name rather than by ID.

0.62.181 (2024-05-24)

- Background committing on `modal.Volume` mounts is now default behavior.

0.62.178 (2024-05-21)

- Added a `modal container stop` CLI command that will kill an active container and reassign its current inputs.

0.62.175 (2024-05-17)

- `modal.CloudBucketMount` now supports writing to Google Cloud Storage buckets.

0.62.174 (2024-05-17)

- Using `memory=` to specify the type of `modal.gpu.A100` is deprecated in favor of `size=`. Note that `size` accepts a string type (`"40GB"` or `"80GB"`) rather than an integer, as this is a request for a specific variant of the A100 GPU.

0.62.173 (2024-05-17)

- Added a `version` flag to the `modal.Volume` API and CLI, allow opting in to a new backend implementation.

0.62.172 (2024-05-17)

- Fixed a bug where other functions weren't callable from within an `asgi_app` or `wsgi_app` constructor function and side effects of `@enter` methods weren't available in that scope.

0.62.166 (2024-05-14)

- Disabling background commits on `modal.Volume` volumes is now deprecated. Background commits will soon become mandatory behavior.

0.62.165 (2024-05-13)

- Deprecated `wait_for_response=False` on web endpoints. See the docs for alternatives.

0.62.162 (2024-05-13)

- A deprecation warning is now raised when using `modal.Stub`, which has been renamed to `modal.App`. Additionally, it is recommended to use `app` as the variable name rather than `stub`, which matters when using the automatic app discovery feature in the `modal run` CLI command.

0.62.159 (2024-05-10)

- Added a `--stream-logs` flag to `modal deploy` that, if True, begins streaming the app logs once deployment is complete.

0.62.156 (2024-05-09)

- Added support for looking up a deployed App by its deployment name in `modal app logs`

0.62.150 (2024-05-08)

- Added validation that App `name`, if provided, is a string.

0.62.149 (2024-05-08)

- The `@app.function` decorator now raises an error when it is used to decorate a class (this was always invalid, but previously produced confusing behavior).

0.62.148 (2024-05-08)

- The `modal app list` output has been improved in several ways:
  - Persistent storage objects like Volumes or Dicts are no longer included (these objects receive an app ID internally, but this is an implementation detail and subject to future change). You can use the dedicated CLI for each object (e.g. `modal volume list`) instead.
  - For Apps in a *stopped* state, the output is now limited to those stopped within the past 2 hours.
  - The number of tasks running for each App is now shown.

0.62.146 (2024-05-07)

- Added the `region` parameter to the `modal.App.function` and `modal.App.cls` decorators. This feature allows the selection of specific regions for function execution. Note that it is available only on some plan types. See our blog post for more details.

0.62.144 (2024-05-06)

- Added deprecation warnings when using Python 3.8 locally or in a container. Python 3.8 is nearing EOL, and Modal will be dropping support for it soon.

0.62.141 (2024-05-03)

- Deprecated the `Image.conda` constructor and the `Image.conda_install` / `Image.conda_update_from_environment` methods. Conda-based images had a number of tricky issues and were generally slower and heavier than images based on `micromamba`, which offers a similar featureset and can install packages from the same repositories.
- Added the `spec_file` parameter to allow `Image.micromamba_install` to install dependencies from a local file. Note that `micromamba` supports conda yaml syntax along with simple text files.

0.62.131 (2024-05-01)

- Added a deprecation warning when object names are invalid. This applies to `Dict`, `NetworkFileSystem`, `Secret`, `Queue`, and `Volume` objects. Names must be shorter than 64 characters and may contain only alphanumeric characters, dashes, periods, and underscores. These rules were previously enforced, but the check had inadvertently been dropped in a recent refactor. Please update the names of your objects and transfer any data to retain access, as invalid names will become an error in a future release.

0.62.130 (2024-05-01)

- Added a command-line interface for interacting with `modal.Queue` objects. Run `modal queue --help` in your terminal to see what is available.

0.62.116 (2024-04-26)

- Added a command-line interface for interacting with `modal.Dict` objects. Run `modal dict --help` in your terminal to see what is available.

0.62.114 (2024-04-25)

- `Secret.from_dotenv` now accepts an optional filename keyword argument:

```
@app.function(secrets=[modal.Secret.from_dotenv(filename=".env-dev")])
def run():
    ...
```

0.62.110 (2024-04-25)

- Passing a glob `**` argument to the `modal volume get` CLI has been deprecated — instead, simply download the desired directory path, or `/` for the entire volume.
- `Volume.listdir()` no longer takes trailing glob arguments. Use `recursive=True` instead.
- `modal volume get` and `modal nfs get` performance is improved when downloading a single file. They also now work with multiple files when outputting to stdout.
- Fixed a visual bug where `modal volume get` on a single file will incorrectly display the destination path.

0.62.109 (2024-04-24)

- Improved feedback for deserialization failures when objects are being transferred between local / remote environments.

0.62.108 (2024-04-24)

- Added `Dict.delete` and `Queue.delete` as API methods for deleting named storage objects:

```
import modal
modal.Queue.delete("my-job-queue")
```

- Deprecated invoking `Volume.delete` as an instance method; it should now be invoked as a static method with the name of the Volume to delete, as with the other methods.

0.62.98 (2024-04-21)

- The `modal.Dict` object now implements a `keys` / `values` / `items` API. Note that there are a few differences when compared to standard Python dicts:
  - The return value is a simple iterator, whereas Python uses a dictionary view object with more features.
  - The results are unordered.
- Additionally, there was no key data stored for items added to a `modal.Dict` prior to this release, so empty strings will be returned for these entries.

0.62.81 (2024-04-18)

- We are introducing `modal.App` as a replacement for `modal.Stub` and encouraging the use of "app" terminology over "stub" to reduce confusion between concepts used in the SDK and the Dashboard. Support for `modal.Stub` will be gradually deprecated over the next few months.

0.62.72 (2024-04-16)

- Specifying a hard memory limit for a `modal.Function` is now supported. Pass a tuple of `memory=(request, limit)`. Above the `limit`, which is specified in MiB, a Function's container will be OOM killed.

0.62.70 (2024-04-16)

- `modal.CloudBucketMount` now supports read-only access to Google Cloud Storage

0.62.69 (2024-04-16)

- Iterators passed to `Function.map()` and similar parallel execution primitives are now executed on the main thread, preventing blocking iterators from possibly locking up background Modal API calls, and risking task shutdowns.

0.62.67 (2024-04-15)

- The return type of `Volume.listdir()`, `Volume.iterdir()`, `NetworkFileSystem.listdir()`, and `NetworkFileSystem.iterdir()` is now a `FileEntry` dataclass from the `modal.volume` module. The fields of this data class are the same as the old protobuf object returned by these methods, so it should be mostly backwards-compatible.

0.62.65 (2024-04-15)

- Cloudflare R2 bucket support added to `modal.CloudBucketMount`

0.62.55 (2024-04-11)

- When Volume reloads fail due to an open file, we now try to identify and report the relevant path. Note that there may be some circumstances in which we are unable to identify the specific file blocking a reload and will report a generic error message in that case.

0.62.53 (2024-04-10)

- Values in the `modal.toml` config file that are spelled as `0`, `false`, `"False"`, or `"false"` will now be coerced in Python to `False`, whereas previously only `"0"` (as a string) would have the intended effect.

0.62.25 (2024-04-01)

- Fixed a recent regression that caused functions using `modal.interact()` to crash.

0.62.15 (2024-03-29)

- Queue methods `put`, `put_many`, `get`, `get_many` and `len` now support an optional `partition` argument (must be specified as a `kwarg`). When specified, users read and write from new partitions of the queue independently. `partition=None` corresponds to the default partition of the queue.

0.62.3 (2024-03-27)

- User can now mount S3 buckets using Requester Pays. This can be done with `CloudBucketMount(..., requester_pays=True)`.

0.62.1 (2024-03-27)

- Raise an error on `@web_server(startup_timeout=0)`, which is an invalid configuration.

0.62.0 (2024-03-26)

- The `.new()` method has now been deprecated on all Modal objects. It should typically be replaced with `.from_name(...)` in Modal app code, or `.ephemeral()` in scripts that use Modal
- Assignment of Modal objects to a `Stub` via subscription (`stub["object"]`) or attribute (`stub.object`) syntax is now deprecated. This syntax was only necessary when using `.new()`.

0.61

0.61.104 (2024-03-25)

- Fixed a bug where images based on `micromamba` could fail to build if requesting Python 3.12 when a different version of Python was being used locally.

0.61.76 (2024-03-19)

- The `Sandbox`'s `LogsReader` is now an asynchronous iterable. It supports the `async for` statement to stream data from the sandbox's `stdout/stderr`.

```
@stub.function()
async def my_fn():
    sandbox = stub.spawn_sandbox(
      "bash",
      "-c",
      "while true; do echo foo; sleep 1; done"
    )
    async for message in sandbox.stdout:
        print(f"Message: {message}")
```

0.61.57 (2024-03-15)

- Add the `@web_server` decorator, which exposes a server listening on a container port as a web endpoint.

0.61.56 (2024-03-15)

- Allow users to write to the `Sandbox`'s `stdin` with `StreamWriter`.

```
@stub.function()
def my_fn():
    sandbox = stub.spawn_sandbox(
        "bash",
        "-c",
        "while read line; do echo $line; done",
    )
    sandbox.stdin.write(b"foo\\n")
    sandbox.stdin.write(b"bar\\n")
    sandbox.stdin.write_eof()
    sandbox.stdin.drain()
    sandbox.wait()
```

0.61.53 (2024-03-15)

- Fixed an bug where `Mount` was failing to include symbolic links.

0.61.45 (2024-03-13)

When called from within a container, `modal.experimental.stop_fetching_inputs()` causes it to gracefully exit after the current input has been processed.

0.61.35 (2024-03-12)

- The `@wsgi_app()` decorator now uses a different backend based on `a2wsgi` that streams requests in chunks, rather than buffering the entire request body.

0.61.32 (2024-03-11)

- Stubs/apps can now be "composed" from several smaller stubs using `stub.include(...)`. This allows more ergonomic setup of multi-file Modal apps.

0.61.31 (2024-03-08)

- The `Image.extend` method has been deprecated. This is a low-level interface and can be replaced by other `Image` methods that offer more flexibility, such as `Image.from_dockerfile`, `Image.dockerfile_commands`, or `Image.run_commands`.

0.61.24 (2024-03-06)

- Fixes `modal volume put` to support uploading larger files, beyond 40 GiB.

0.61.22 (2024-03-05)

- Modal containers now display a warning message if lingering threads are present at container exit, which prevents runner shutdown.

0.61.17 (2024-03-05)

- Bug fix: Stopping an app while a container's `@exit()` lifecycle methods are being run no longer interrupts the lifecycle methods.
- Bug fix: Worker preemptions no longer interrupt a container's `@exit()` lifecycle method (until 30 seconds later).
- Bug fix: Async `@exit()` lifecycle methods are no longer skipped for sync functions.
- Bug fix: Stopping a sync function with `allow_concurrent_inputs>1` now actually stops the container. Previously, it would not propagate the signal to worker threads, so they would continue running.
- Bug fix: Input-level cancellation no longer skips the `@exit()` lifecycle method.
- Improve stability of container entrypoint against race conditions in task cancellation.

0.61.9 (2024-03-05)

- Fix issue with pdm where all installed packages would be automounted when using package cache (MOD-2485)

0.61.6 (2024-03-04)

- For modal functions/classes with `concurrency_limit < keep_warm`, we'll raise an exception now. Previously we (silently) respected the `concurrency_limit` parameter.

0.61.1 (2024-03-03)

`modal run --interactive` or `modal run -i` run the app in "interactive mode". This allows any remote code to connect to the user's local terminal by calling `modal.interact()`.

```
@stub.function()
def my_fn(x):
    modal.interact()
```

```
    x = input()
    print(f"Your number is {x}")
```

This means that you can dynamically start an IPython shell if desired for debugging:

```
@stub.function()
def my_fn(x):
    modal.interact()

    from IPython import embed
    embed()
```

For convenience, breakpoints automatically call `interact()`:

```
@stub.function()
def my_fn(x):
    breakpoint()
```

0.60

0.60.0 (2024-02-29)

- `Image.run_function` now allows you to pass args and kwargs to the function. Usage:

```
def my_build_function(name, size, *, variant=None):
    print(f"Building {name} {size} {variant}")


image = modal.Image.debian_slim().run_function(
    my_build_function, args=("foo", 10), kwargs={"variant": "bar"}
)
```

0.59

0.59.0 (2024-02-28)

- Mounted packages are now deduplicated across functions in the same stub

- Mounting of local Python packages are now marked as such in the mount creation output, e.g. `PythonPackage:my_package`
- Automatic mounting now includes packages outside of the function file's own directory. Mounted packages are mounted in /root/

0.58

0.58.92 (2024-02-27)

- Most errors raised through usage of the CLI will now print a simple error message rather than showing a traceback from inside the `modal` library.
- Tracebacks originating from user code will include fewer frames from within `modal` itself.
- The new `MODAL_TRACEBACK` environment variable (and `traceback` field in the Modal config file) can override these behaviors so that full tracebacks are always shown.

0.58.90 (2024-02-27)

- Fixed a bug that could cause `cls` -based functions to to ignore timeout signals.

0.58.88 (2024-02-26)

- `volume get` performance is improved for large (> 100MB) files

0.58.79 (2024-02-23)

- Support for function parameters in methods decorated with `@exit` has been deprecated. Previously, exit methods were required to accept three arguments containing exception information (akin to `__exit__` in the context manager protocol). However, due to a bug, these arguments were always null. Going forward, `@exit` methods are expected to have no parameters.

0.58.75 (2024-02-23)

- Function calls can now be cancelled without killing the container running the inputs. This allows new inputs by different function calls to the same function to be picked up immediately without having to cold-start new containers after cancelling calls.

0.57

0.57.62 (2024-02-21)

- An `InvalidError` is now raised when a lifecycle decorator (`@build`, `@enter`, or `@exit`) is used in conjunction with `@method`. Previously, this was undefined and could produce confusing failures.

0.57.61 (2024-02-21)

- Reduced the amount of context for frames in modal's CLI framework when showing a traceback.

0.57.60 (2024-02-21)

- The "dunder method" approach for class lifecycle management (`__build__`, `__enter__`, `__exit__`, etc.) is now deprecated in favor of the modal `@build`, `@enter`, and `@exit` decorators.

0.57.52 (2024-02-17)

- In `modal token new` and `modal token set`, the `--no-no-verify` flag has been removed in favor of a `--verify` flag. This remains the default behavior.

0.57.51 (2024-02-17)

- Fixes a regression from 0.57.40 where `@enter` methods used a separate event loop.

0.57.42 (2024-02-14)

- Adds a new environment variable/config setting, `MODAL_FORCE_BUILD` / `force_build`, that coerces all images to be built from scratch, rather than loaded from cache.

0.57.40 (2024-02-13)

- The `@enter()` lifecycle method can now be used to run additional setup code prior to function checkpointing (when the class is decorated with `stub.cls(enable_checkpointing=True)`. Note that there are currently some limitations on function checkpointing:
  - Checkpointing only works for CPU memory; any GPUs attached to the function will not available
  - Networking is disabled while the checkpoint is being created
- Please note that function checkpointing is still a beta feature.

0.57.31 (2024-02-12)

- Fixed an issue with displaying deprecation warnings on Windows systems.

0.57.22 (2024-02-09)

- Modal client deprecation warnings are now highlighted in the CLI

0.57.16 (2024-02-07)

- Fixes a regression in container scheduling. Users on affected versions (**0.57.5—0.57.15**) are encouraged to upgrade immediately.

0.57.15 (2024-02-07)

- The legacy `image_python_version` config option has been removed. Use the `python_version=` parameter on your image definition instead.

0.57.13 (2024-02-07)

- Adds support for mounting an S3 bucket as a volume.

0.57.9 (2024-02-07)

- Support for an implicit 'default' profile is now deprecated. If you have more than one profile in your Modal config file, one must be explicitly set to `active` (use `modal profile activate` or edit your `.modal.toml` file to resolve).
- An error is now raised when more than one profile is set to `active`.

0.57.2 (2024-02-06)

- Improve error message when generator functions are called with `.map(...)`.

0.57.0 (2024-02-06)

- Greatly improved streaming performance of generators and WebSocket web endpoints.
- **Breaking change:** You cannot use `.map()` to call a generator function. (In previous versions, this merged the results onto a single stream, but the behavior was undocumented and not widely used.)
- **Incompatibility:** Generator outputs are now on a different internal system. Modal code on client versions before 0.57 cannot trigger deployed functions with `.remote_gen()` that are on client version 0.57, and vice versa.

0.56

Note that in version 0.56 and prior, Modal used a different numbering system for patch releases.

0.56.4964 (2024-02-05)

- When using `modal token new` or `model token set`, the profile containing the new token will now be activated by default. Use the `--no-activate` switch to update the `modal.toml` file without activating the corresponding profile.

0.56.4953 (2024-02-05)

- The `modal profile list` output now indicates when the workspace is determined by a token stored in environment variables.

0.56.4952 (2024-02-05)

- Variadic parameters (e.g. *args and **kwargs) can now be used in scheduled functions as long as the function doesn't have any other parameters without a default value

0.56.4903 (2024-02-01)

- `modal container exec`'s `--no-tty` flag has been renamed to `--no-pty`.

0.56.4902 (2024-02-01)

- The singular form of the `secret` parameter in `Stub.function`, `Stub.cls`, and `Image.run_function` has been deprecated. Please update your code to use the plural form instead: `secrets=[Secret(...)]`.

0.56.4885 (2024-02-01)

- In `modal profile list`, the user's GitHub username is now shown as the name for the "Personal" workspace.

0.56.4874 (2024-01-31)

- The `modal token new` and `modal token set` commands now create profiles that are more closely associated with workspaces, and they have more explicit profile activation behavior:

    - By default, these commands will create/update a profile named after the workspace that the token points to, rather than a profile named “default”
    - Both commands now have an `--activate` flag that will activate the profile associated with the new token
    - If no other profiles exist at the time of creation, the new profile will have its `active` metadata set to True
- With these changes, we are moving away from the concept of a “default” profile. Implicit usage of the “default” profile will be deprecated in a future update.

0.56.4849 (2024-01-29)

- Adds tty support to `modal container exec` for fully-interactive commands. Example: `modal container exec [container-id] /bin/bash`

0.56.4792 (2024-01-26)

- The `modal profile list` command now shows the workspace associated with each profile.

0.56.4715 (2024-01-24)

- `Mount.from_local_python_packages` now places mounted packages at `/root` in the Modal runtime by default (used to be `/pkg`). To override this behavior, the function now takes a `remote_dir: Union[str, PurePosixPath]` argument.

0.56.4707 (2024-01-23)

- The Modal client library is now compatible with Python 3.12, although there are a few limitations:
    - Images that use Python 3.12 without explicitly specifing it through `python_version` or `add_python` will not build properly unless the modal client is also running on Python 3.12.
    - The `conda` and `microconda` base images currently do not support Python 3.12 because an upstream dependency is not yet compatible.

0.56.4700 (2024-01-22)

- `gpu.A100` class now supports specifying GiB memory configuration using a `size: str` parameter. The `memory: int` parameter is deprecated.

0.56.4693 (2024-01-22)

- You can now execute commands in running containers with `modal container exec [container-id] [command]`.

0.56.4691 (2024-01-22)

- The `modal` cli now works more like the `python` cli in regard to script/module loading:
  - Running `modal my_dir/my_script.py` now puts `my_dir` on the PYTHONPATH.
  - `modal my_package.my_module` will now mount to /root/my_package/my_module.py in your Modal container, regardless if using automounting or not (and any intermediary `__init__.py` files will also be mounted)

0.56.4687 (2024-01-20)

- Modal now uses the current profile if `MODAL_PROFILE` is set to the empty string.

0.56.4649 (2024-01-17)

- Dropped support for building Python 3.7 based `modal.Image`s. Python 3.7 is end-of-life since late June 2023.

0.56.4620 (2024-01-16)

- modal.Stub.function now takes a `block_network` argument.

0.56.4616 (2024-01-16)

- modal.Stub now takes a `volumes` argument for setting the default volumes of all the stub's functions, similarly to the `mounts` and `secrets` argument.

0.56.4590 (2024-01-13)

- `modal serve`: Setting MODAL_LOGLEVEL=DEBUG now displays which files cause an app reload during serve

0.56.4570 (2024-01-12)

- `modal run` cli command now properly propagates `--env` values to object lookups in global scope of user code