```
from unittest.mock import patch

import pytest

from swarms_cloud.sky_api import SkyInterface


# Create a fixture for an instance of SkyInterface
@pytest.fixture
def sky_interface():
    return SkyInterface()


# Create a fixture for a mock of the sky module
@pytest.fixture
def mock_sky():
    with patch("swarms_cloud.sky_api.sky") as mock:
        yield mock


def test_launch(sky_interface, mock_sky):
    mock_sky.launch.return_value = ("job_id", "handle")
    job_id = sky_interface.launch("task", "cluster_name")
    mock_sky.launch.assert_called_once_with("task", cluster_name="cluster_name")
    assert job_id == "job_id"
    assert sky_interface.clusters["cluster_name"] == "handle"
```

```python
def test_execute(sky_interface, mock_sky):
    sky_interface.clusters["cluster_name"] = "handle"
    sky_interface.execute("task", "cluster_name")
    mock_sky.exec.assert_called_once_with("task", "cluster_name")


def test_stop(sky_interface, mock_sky):
    sky_interface.clusters["cluster_name"] = "handle"
    sky_interface.stop("cluster_name")
    mock_sky.stop.assert_called_once_with("cluster_name")


def test_start(sky_interface, mock_sky):
    sky_interface.clusters["cluster_name"] = "handle"
    sky_interface.start("cluster_name")
    mock_sky.start.assert_called_once_with("cluster_name")


def test_down(sky_interface, mock_sky):
    sky_interface.clusters["cluster_name"] = "handle"
    sky_interface.down("cluster_name")
    mock_sky.down.assert_called_once_with("cluster_name")
    assert "cluster_name" not in sky_interface.clusters
```

```
def test_status(sky_interface, mock_sky):
    sky_interface.status()
    mock_sky.status.assert_called_once()


def test_autostop(sky_interface, mock_sky):
    sky_interface.clusters["cluster_name"] = "handle"
    sky_interface.autostop("cluster_name")
    mock_sky.autostop.assert_called_once_with("cluster_name")
```