Run your first cloud sandbox

- Prerequisites
- Create a project
- Create and use a sandbox
- Run the program
- Next steps

Create a cloud sandbox, run a command inside it, and delete it using the Docker Sandboxes TypeScript SDK. This tutorial uses the bundled `shell` kit to print `Hello from Docker Sandboxes`. You don't need an AI agent or a model provider API key to run it.

Prerequisites

To follow this tutorial, you need:

- A Docker account with an active Docker Agentic Platform subscription
- Node.js 20 or later and npm

Create a project

Create a directory and initialize a Node.js project:

```
$ mkdir sandboxes-api-tutorial
$ cd sandboxes-api-tutorial
$ npm init --yes
$ npm pkg set type=module
```

Install the SDK and TypeScript tooling:

```
$ npm install @docker/sandboxes
$ npm install --save-dev tsx typescript @types/node
```

Create and use a sandbox

A kit supplies the sandbox's image and configuration for an agent or tool. The `shell` kit provides the environment for this example.

The SDK launches it with the default `small` compute size: two CPUs and 4 GiB of memory. Cloud compute is billed to your subscription. See Compute sizes and limits for other sizes.

Create a file named `index.ts` with the following code. The program prompts you to sign in, creates a sandbox, runs a command, and deletes the sandbox.

```
import { oauth, Sandboxes } from '@docker/sandboxes';

const client = new Sandboxes({
  auth: oauth({
    onVerification({ verificationUriComplete, verificationUri, userCode }) {
      console.log(`Open ${verificationUriComplete ?? verificationUri}`);
      console.log(`Verification code: ${userCode}`);
    },
  }),
});

try {
  const sandbox = await client.kits.launchAndWait('shell');
  console.log('Sandbox:', sandbox.name);

  const result = await sandbox.processes.run(
    { args: ['echo', 'Hello from Docker Sandboxes'] },
    { timeoutMs: 300_000 },
  );
  console.log(result.stdout.trim());

  const latest = await sandbox.refresh();
  const deleting = await latest.delete({ force: true });
  await deleting?.waitUntilDeleted();
  console.log('Deleted', sandbox.name);
```

```
} finally {
  await client.close();
}
```

`kits.launchAndWait` creates the sandbox and waits until it's running. `processes.run` runs the command and collects its output. The program then reads the sandbox's latest state, deletes it, and waits for deletion to finish. The `force` option permits deletion while the sandbox is running.

The SDK handles sign-in, access tokens, and the connection to the sandbox. Closing the client releases its local resources.

Run the program

Run the program with `tsx`:

```
$ npx tsx index.ts
```

Open the printed verification URL and sign in with the Docker account that has your cloud subscription. The program continues after sign-in. On success, it prints `Hello from Docker Sandboxes`, then confirms sandbox deletion.

For CI jobs and other unattended applications, use PAT authentication.

If the program stops after printing the sandbox name, retrieve the sandbox with `client.get(name)` and delete it when you're finished. Closing the client doesn't delete the sandbox.

Next steps

- Explore the SDK cookbook for examples you can use in your application.
- If you use a kit that clones a repository or installs tools, wait for kit setup before using its results.
- Read API concepts to learn about resource names, lifecycle states, and supported Cloud options.
- Review Errors and retries before adding recovery logic.

□□□ #2
□ □□□ □□□ 2026-10-31 18:57:27 UTC
□ □□□ □□□ 2026-10-31 18:57:29 UTC