Snapshot and fork a sandbox

- Capture a snapshot {#1-capture-a-snapshot}
- Restore into another sandbox {#2-restore-into-another-sandbox}
- Find saved snapshots {#3-find-saved-snapshots}
- Delete a snapshot {#4-delete-a-snapshot}

Save a sandbox's state so you can start another sandbox from the same point later. A snapshot is separate from its source sandbox; restoring it creates a new sandbox with its own identity.

Use an authenticated client and the resource name of a running sandbox.

Capture a snapshot {#1-capture-a-snapshot}

Choose a display name, capture mode, and idempotency key. Use disk-only capture when you need the filesystem. Request memory only when your environment supports it and you need the running state.

The example waits until capture finishes and returns a snapshot handle. Keep the returned name for restoration.

A failed capture carries failure details; a timed-out wait does not prove the capture was cancelled.

Snapshots can contain credentials and private project data. Restrict access to them, especially when capturing memory.

{{< tabs >}} {{< tab name="TypeScript" >}}

```
const snapshot = await sandbox.snapshot(
  {
    displayName: snapshotName,
    captureMode: withMemory ? 'all' : 'disk',
```

```
  },
  { idempotencyKey: requestId },
);
return snapshot.waitUntilReady();
```

Complete TypeScript example: snapshots/create.ts

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

export async function createSnapshot(
  client: Sandboxes,
  sandboxName: string,
  snapshotName: string,
  withMemory: boolean,
  requestId: string,
) {
  const sandbox = await client.get(sandboxName);
  const snapshot = await sandbox.snapshot(
    {
      displayName: snapshotName,
      captureMode: withMemory ? 'all' : 'disk',
    },
    { idempotencyKey: requestId },
  );
  return snapshot.waitUntilReady();
}
```

{{< /tab >}} {{< /tabs >}}

Restore into another sandbox {#2-restore-into-another-sandbox}

Use a ready snapshot's name and a display name for the new sandbox. The example waits for the restored sandbox to run.

The call returns a sandbox handle, ready for running commands or transferring files.

The source sandbox does not need to remain running. Restoration does not replace it. Save the new sandbox's name and delete it separately when you finish.

{{< tabs >}} {{< tab name="TypeScript" >}}

```
const sandbox = await snapshot.restore(
  { displayName: newSandboxName },
  { timeoutMs: 300_000, idempotencyKey: requestId },
);
return sandbox.waitUntilRunning();
```

Complete TypeScript example: snapshots/restore.ts

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

export async function restoreSnapshot(
  client: Sandboxes,
  snapshotName: string,
  newSandboxName: string,
  requestId: string,
) {
  const snapshot = await client.snapshots.get(snapshotName);
  const sandbox = await snapshot.restore(
    { displayName: newSandboxName },
    { timeoutMs: 300_000, idempotencyKey: requestId },
  );
  return sandbox.waitUntilRunning();
}
```

{{< /tab >}} {{< /tabs >}}

Find saved snapshots {#3-find-saved-snapshots}

List snapshots for the source sandbox. The example follows all pages and returns summaries. Read a selected snapshot before restoring it if you need its current status or complete metadata.

{{< tabs >}} {{< tab name="TypeScript" >}}

```
return client.snapshots.all({ sandbox: sandboxName }).collect();
```

Complete TypeScript example: snapshots/list.ts

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

export async function listSnapshots(
  client: Sandboxes,
  sandboxName: string,
) {
  return client.snapshots.all({ sandbox: sandboxName }).collect();
}
```

{{< /tab >}} {{< /tabs >}}

Delete a snapshot {#4-delete-a-snapshot}

Delete through a handle when you no longer need that restore point. Deletion removes the snapshot, not sandboxes that have already been restored from it. Snapshot deletion is refused while a sandbox restored from it is still running.

Deleting the source sandbox and deleting its snapshots are separate cleanup steps.

{{< tabs >}} {{< tab name="TypeScript" >}}

```
await snapshot.delete();
```

Complete TypeScript example: snapshots/delete.ts

```
import type { Snapshot } from '@docker/sandboxes';

export async function deleteSnapshot(snapshot: Snapshot) {
  await snapshot.delete();
```

```
}
```

{{< /tab >}} {{< /tabs >}}

□□□ #2
□ □□□ □□□ 2026-10-31 18:57:12 UTC
□ □□□ □□□ 2026-10-31 18:57:14 UTC