Retry without creating duplicates

- Retry the same create request {#1-retry-the-same-create-request}
- Start a process once {#2-start-a-process-once}

Retry a request without creating duplicate work. An idempotency key identifies one logical operation, such as creating the sandbox for a particular job.

Use an authenticated client. Generate the key once, then persist it with the job if retries can happen in another program run.

Retry the same create request {#1-retry-the-same-create-request}

Reuse the same key and payload on every attempt. The example retries a transient availability refusal and disables automatic SDK retries. Do not run your own retry loop and the SDK's together without accounting for their total attempts.

Use a deadline and a bounded attempt count. For production retry scheduling, add delay and jitter or use the SDK's configured retry behavior. Changing the payload while keeping the key is not an update; it conflicts with the original request.

An accepted creation returns the sandbox's resource data. Save its name and use the client's get method to obtain a handle. Read or wait on that resource to follow progress instead of repeating creation to make it advance.

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

```
for (let attempt = 0; attempt < attempts; attempt++) {
  try {
    return await client.kits.launch(
      'shell',
      { displayName, resources: { cpus: 2, memoryMib: 4096 } },
      {
        idempotencyKey: requestId,
        timeoutMs: 300_000,
        signal,
        maxRetries: 0,
      },
    );
  } catch (error) {
    if (
      !(error instanceof RequestError) ||
      error.raw.code !== 'unavailable' ||
      attempt + 1 === attempts
    )
      throw error;
  }
}
```

Complete TypeScript example: idempotency/retry.ts

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

export async function createWithRetry(
  client: Sandboxes,
  displayName: string,
  requestId: string,
  attempts: number,
) {
  if (!Number.isInteger(attempts) || attempts < 1)
    throw new RangeError('attempts must be at least 1');
  const signal = AbortSignal.timeout(300_000);
  for (let attempt = 0; attempt < attempts; attempt++) {
    try {
      return await client.kits.launch(
        'shell',
```

```
        { displayName, resources: { cpus: 2, memoryMib: 4096 } },
        {
          idempotencyKey: requestId,
          timeoutMs: 300_000,
          signal,
          maxRetries: 0,
        },
      );
    } catch (error) {
      if (
        !(error instanceof RequestError) ||
        error.raw.code !== 'unavailable' ||
        attempt + 1 === attempts
      )
        throw error;
    }
  }
  throw new Error('attempt count was validated');
}
```

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

Start a process once {#2-start-a-process-once}

Start the process with its own idempotency key and keep the returned process name. After losing a response or connection, find that process before starting another one.

Keys do not make every operation safe to replay. Repeating process input, a signal, or a file write can have a second effect. Use process reconnection for a lost stream and inspect files before repeating an uncertain write.

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

```
return sandbox.processes.start({ args }, { idempotencyKey: requestId });
```

Complete TypeScript example: idempotency/unsafe.ts

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

export async function createProcessOnce(
  sandbox: Sandbox,
  args: string[],
  requestId: string,
) {
  return sandbox.processes.start({ args }, { idempotencyKey: requestId });
}
```

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

□□□ #2
□ □□□ □□□ 2026-10-31 18:56:48 UTC
□ □□□ □□□ 2026-10-31 18:56:49 UTC