Run Claude Code in a Docker Sandbox with Docker Model Runner

- How the pieces fit together
- Prerequisites
- Step 1: Pull a coding model
- Step 2: Allow the sandbox to reach Docker Model Runner
- Step 3: Create a Claude Code sandbox
- Step 4: Verify connectivity to Docker Model Runner
- Step 5: Launch Claude Code with the local model
- Step 6: Inspect Claude Code requests
- Step 7: Package gpt-oss with a larger context window
- Clean up
- Learn more

This guide shows how to run Claude Code inside a Docker Sandbox with Docker Model Runner as the backend model provider. You'll keep the agent isolated from your host in a microVM, point it at a local model on your machine, and keep all model traffic on-device.

> **Acknowledgment**
>
> Docker would like to thank Pradumna Saraf for his contribution to this guide.

In this guide, you'll learn how to:

- Pull a coding model and start Docker Model Runner with TCP enabled
- Allow the sandbox to reach Docker Model Runner on your host

- Create a Claude Code sandbox and set the local endpoint persistently
- Launch Claude Code with a local model and verify the connection
- Package `gpt-oss` with a larger context window for longer prompts

How the pieces fit together

Three components cooperate at runtime:

- **Docker Model Runner** runs on your host and serves an Anthropic-compatible API at `http://localhost:12434`.
- **The Docker Sandbox** runs Claude Code inside an isolated microVM. The microVM has its own network and can't reach your host's `localhost` directly.
- **The sandbox proxy** sits on your host and brokers every outbound request from the sandbox. It enforces network policy and translates the special hostname `host.docker.internal` to `localhost`.

Claude Code inside the sandbox sends requests to `http://host.docker.internal:12434`. The proxy rewrites the destination to `localhost:12434`, which Docker Model Runner answers. No model traffic leaves your machine.

Prerequisites

Before you start, make sure you have:

- Docker Desktop or Docker Engine installed
- Docker Model Runner enabled
- Docker Sandboxes (`sbx`) version 0.39.0 or later installed and signed in

If you use Docker Desktop, turn on TCP access in **Settings** > **AI**, or run:

```
$ docker desktop enable model-runner --tcp 12434
```

Step 1: Pull a coding model

Pull a model on your host before you create the sandbox:

```
$ docker model pull ai/devstral-small-2
```

You can also use `ai/qwen3-coder` if you want another coding-focused model with a large context window.

Step 2: Allow the sandbox to reach Docker Model Runner

Sandboxes are network-isolated by default, so you need a policy rule before the sandbox can reach Docker Model Runner.

The rule is matched against the destination the proxy forwards to, not the hostname the sandbox uses. Because the proxy rewrites `host.docker.internal` to `localhost` before forwarding, the rule allows `localhost:12434` even though Claude Code will use `host.docker.internal` in its requests:

```
$ sbx policy allow network localhost:12434
```

For background on host access from sandboxes, see Accessing host services from a sandbox.

Step 3: Create a Claude Code sandbox

From your project directory, create a sandbox without launching the agent. Set `ANTHROPIC_BASE_URL` so Claude Code uses Docker Model Runner whenever the sandbox starts:

```
$ cd ~/my-project
$ sbx create --name claude-dmr \
  -e ANTHROPIC_BASE_URL=http://host.docker.internal:12434 \
  claude .
```

`sbx run` would also work, but it launches Claude Code immediately. Creating the sandbox first lets you confirm the variable and test connectivity before the agent starts.

You don't need to set an Anthropic API key or run `sbx secret set anthropic`. Docker Model Runner doesn't authenticate the local endpoint, and the sandbox proxy only injects credentials for requests bound for `api.anthropic.com`. See Credentials for the full list of services the proxy authenticates.

For more ways to set variables, see Set environment variables.

To confirm the variable is set, open a shell in the sandbox:

```
$ sbx exec -it claude-dmr bash
$ echo $ANTHROPIC_BASE_URL
http://host.docker.internal:12434
```

Step 4: Verify connectivity to Docker Model Runner

Still inside the sandbox shell, send a test request to the host endpoint:

```
$ curl http://host.docker.internal:12434/v1/messages \
  -H "Content-Type: application/json" \
  -d '{
    "model": "ai/devstral-small-2",
    "max_tokens": 32,
    "messages": [{"role": "user", "content": "Say hello"}]
  }'
```

A successful response confirms the policy rule and base URL are correct. Type `exit` to leave the shell. For more details about the request format, see the Anthropic-compatible API reference.

Step 5: Launch Claude Code with the local model

Run Claude Code in the sandbox and pass the model flag through to the agent:

```
$ sbx run --name claude-dmr -- --model ai/devstral-small-2
```

Everything after `--` is forwarded to the Claude Code CLI. `ANTHROPIC_BASE_URL` is stored with the sandbox and available to every session, so Claude Code routes requests to Docker Model Runner on your host instead of `api.anthropic.com`.

Step 6: Inspect Claude Code requests

To inspect the requests Claude Code sends, run on your host:

```
$ docker model requests --model ai/devstral-small-2 | jq .
```

This helps you debug prompts, context usage, and compatibility issues without attaching to the sandbox.

Step 7: Package `gpt-oss` with a larger context window

`ai/gpt-oss` defaults to a smaller context window than coding-focused models. To use it for repository-scale prompts, package a larger variant on the host:

```
$ docker model pull ai/gpt-oss
$ docker model package --from ai/gpt-oss --context-size 32000 gpt-oss:32k
```

Then point Claude Code at the packaged model the next time you run the sandbox:

```
$ sbx run --name claude-dmr -- --model gpt-oss:32k
```

Clean up

Sandboxes persist after Claude Code exits. To stop the sandbox without deleting it:

```
$ sbx stop claude-dmr
```

To remove the sandbox and everything inside:

```
$ sbx rm claude-dmr
```

Files in your workspace are unaffected.

Learn more

- Use Claude Code with Docker Model Runner
- Get started with Docker Sandboxes
- Claude Code in Docker Sandboxes
- Docker Model Runner overview
- Docker Model Runner API reference

□□□ #2
□ □□□ □□□ 2026-10-31 18:44:44 UTC
□ □□□ □□□ 2026-10-31 18:44:47 UTC