Code execution tool

Copy page

Claude can analyze data, create visualizations, perform complex calculations, run system commands, create and edit files, and process uploaded files directly within the API conversation. The code execution tool allows Claude to run Bash commands and manipulate files, including writing code, in a secure, sandboxed environment.

The code execution tool is currently in public beta.

To use this feature, add the `"code-execution-2025-08-25"` beta header to your API requests.

Please reach out through our feedback form to share your feedback on this feature.

Model compatibility

The code execution tool is available on the following models:

Model	Tool Version
Claude Opus 4.5 (`claude-opus-4-5-20251101`)	`code_execution_20250825`
Claude Opus 4.1 (`claude-opus-4-1-20250805`)	`code_execution_20250825`
Claude Opus 4 (`claude-opus-4-20250514`)	`code_execution_20250825`
Claude Sonnet 4.5 (`claude-sonnet-4-5-20250929`)	`code_execution_20250825`
Claude Sonnet 4 (`claude-sonnet-4-20250514`)	`code_execution_20250825`
Claude Sonnet 3.7 (`claude-3-7-sonnet-20250219`) (deprecated)	`code_execution_20250825`
Claude Haiku 4.5 (`claude-haiku-4-5-20251001`)	`code_execution_20250825`
Claude Haiku 3.5 (`claude-3-5-haiku-latest`) (deprecated)	`code_exe`

See Upgrade to latest tool version for migration details.

Older tool versions are not guaranteed to be backwards-compatible with newer models. Always use the tool version that corresponds to your model version.

Quick start

Here's a simple example that asks Claude to perform a calculation:

Shell

```
curl https://api.anthropic.com/v1/messages \
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: code-execution-2025-08-25" \
    --header "content-type: application/json" \
    --data '{
        "model": "claude-sonnet-4-5",
        "max_tokens": 4096,
        "messages": [
            {
                "role": "user",
                "content": "Calculate the mean and standard deviation of [1, 2
            }
        ],
        "tools": [{
            "type": "code_execution_20250825",
            "name": "code_execution"
        }]
    }'
```

How code execution works

When you add the code execution tool to your API request:

1. Claude evaluates whether code execution would help answer your question
2. The tool automatically provides Claude with the following capabilities:
   - **Bash commands:** Execute shell commands for system operations and package management

Claude Docs

4. All operations run in a secure sandbox environment
5. Claude provides results with any generated charts, calculations, or analysis

How to use the tool

Execute Bash commands

Ask Claude to check system information and install packages:

Shell

```
curl https://api.anthropic.com/v1/messages \
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: code-execution-2025-08-25" \
    --header "content-type: application/json" \
    --data '{
        "model": "claude-sonnet-4-5",
        "max_tokens": 4096,
        "messages": [{
            "role": "user",
            "content": "Check the Python version and list installed packages"
        }],
        "tools": [{
            "type": "code_execution_20250825",
            "name": "code_execution"
        }]
    }'
```

Create and edit files directly

Claude can create, view, and edit files directly in the sandbox using the file manipulation capabilities:

Shell

Claude Docs

```
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: code-execution-2025-08-25" \
    --header "content-type: application/json" \
    --data '{
        "model": "claude-sonnet-4-5",
        "max_tokens": 4096,
        "messages": [{
            "role": "user",
            "content": "Create a config.yaml file with database settings, then
        }],
        "tools": [{
            "type": "code_execution_20250825",
            "name": "code_execution"
        }]
    }'
```

Upload and analyze your own files

To analyze your own data files (CSV, Excel, images, etc.), upload them via the Files API and reference them in your request:

Using the Files API with Code Execution requires two beta headers: `"anthropic-beta": "code-execution-2025-08-25,files-api-2025-04-14"`

The Python environment can process various file types uploaded via the Files API, including:

- CSV
- Excel (.xlsx, .xls)
- JSON
- XML
- Images (JPEG, PNG, GIF, WebP)
- Text files (.txt, .md, .py, etc)

Upload and analyze files

1. **Upload your file** using the Files API
2. **Reference the file** in your message using a `container_upload` content block

Claude Docs

Shell

```
# First, upload a file
curl https://api.anthropic.com/v1/files \
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: files-api-2025-04-14" \
    --form 'file=@"data.csv"' \

# Then use the file_id with code execution
curl https://api.anthropic.com/v1/messages \
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: code-execution-2025-08-25,files-api-2025-04-14"
    --header "content-type: application/json" \
    --data '{
        "model": "claude-sonnet-4-5",
        "max_tokens": 4096,
        "messages": [{
            "role": "user",
            "content": [
                {"type": "text", "text": "Analyze this CSV data"},
                {"type": "container_upload", "file_id": "file_abc123"}
            ]
        }],
        "tools": [{
            "type": "code_execution_20250825",
            "name": "code_execution"
        }]
    }'
```

Retrieve generated files

When Claude creates files during code execution, you can retrieve these files using the Files API:

Python

Claude Docs

```
# Initialize the client
client = Anthropic()

# Request code execution that creates files
response = client.beta.messages.create(
    model="claude-sonnet-4-5",
    betas=["code-execution-2025-08-25", "files-api-2025-04-14"],
    max_tokens=4096,
    messages=[{
        "role": "user",
        "content": "Create a matplotlib visualization and save it as output.pn
    }],
    tools=[{
        "type": "code_execution_20250825",
        "name": "code_execution"
    }]
)

# Extract file IDs from the response
def extract_file_ids(response):
    file_ids = []
    for item in response.content:
        if item.type == 'bash_code_execution_tool_result':
            content_item = item.content
            if content_item.type == 'bash_code_execution_result':
                for file in content_item.content:
                    if hasattr(file, 'file_id'):
                        file_ids.append(file.file_id)
    return file_ids

# Download the created files
for file_id in extract_file_ids(response):
    file_metadata = client.beta.files.retrieve_metadata(file_id)
    file_content = client.beta.files.download(file_id)
    file_content.write_to_file(file_metadata.filename)
    print(f"Downloaded: {file_metadata.filename}")
```

Combine operations

A complex workflow using all capabilities:

Shell

Claude Docs

```
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: files-api-2025-04-14" \
    --form 'file=@"data.csv"' \
    > file_response.json

# Extract file_id (using jq)
FILE_ID=$(jq -r '.id' file_response.json)

# Then use it with code execution
curl https://api.anthropic.com/v1/messages \
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: code-execution-2025-08-25,files-api-2025-04-14"
    --header "content-type: application/json" \
    --data '{
        "model": "claude-sonnet-4-5",
        "max_tokens": 4096,
        "messages": [{
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "Analyze this CSV data: create a summary report, s
                },
                {
                    "type": "container_upload",
                    "file_id": "'$FILE_ID'"
                }
            ]
        }],
        "tools": [{
            "type": "code_execution_20250825",
            "name": "code_execution"
        }]
    }'
```

Tool definition

The code execution tool requires no additional parameters:

JSON

```
  "name": "code_execution"
}
```

When this tool is provided, Claude automatically gains access to two sub-tools:

- `bash_code_execution` : Run shell commands
- `text_editor_code_execution` : View, create, and edit files, including writing code

Response format

The code execution tool can return two types of results depending on the operation:

Bash command response

```
{
  "type": "server_tool_use",
  "id": "srvtoolu_01B3C4D5E6F7G8H9I0J1K2L3",
  "name": "bash_code_execution",
  "input": {
    "command": "ls -la | head -5"
  }
},
{
  "type": "bash_code_execution_tool_result",
  "tool_use_id": "srvtoolu_01B3C4D5E6F7G8H9I0J1K2L3",
  "content": {
    "type": "bash_code_execution_result",
    "stdout": "total 24\ndrwxr-xr-x 2 user user 4096 Jan 1 12:00 .\ndrwxr-xr-x
    "stderr": "",
    "return_code": 0
  }
}
```

File operation responses

View file:

Claude Docs

```
    "id": "srvtoolu_01C4D5E6F7G8H9I0J1K2L3M4",
    "name": "text_editor_code_execution",
    "input": {
      "command": "view",
      "path": "config.json"
    }
  },
  {
    "type": "text_editor_code_execution_tool_result",
    "tool_use_id": "srvtoolu_01C4D5E6F7G8H9I0J1K2L3M4",
    "content": {
      "type": "text_editor_code_execution_result",
      "file_type": "text",
      "content": "{\n  \"setting\": \"value\",\n  \"debug\": true\n}",
      "numLines": 4,
      "startLine": 1,
      "totalLines": 4
    }
  }
```

Create file:

```
  {
    "type": "server_tool_use",
    "id": "srvtoolu_01D5E6F7G8H9I0J1K2L3M4N5",
    "name": "text_editor_code_execution",
    "input": {
      "command": "create",
      "path": "new_file.txt",
      "file_text": "Hello, World!"
    }
  },
  {
    "type": "text_editor_code_execution_tool_result",
    "tool_use_id": "srvtoolu_01D5E6F7G8H9I0J1K2L3M4N5",
    "content": {
      "type": "text_editor_code_execution_result",
      "is_file_update": false
    }
  }
```

Edit file (str_replace):

```
    "id": "srvtoolu_01E6F7G8H9I0J1K2L3M4N5O6",
    "name": "text_editor_code_execution",
    "input": {
      "command": "str_replace",
      "path": "config.json",
      "old_str": "\"debug\": true",
      "new_str": "\"debug\": false"
    }
  },
  {
    "type": "text_editor_code_execution_tool_result",
    "tool_use_id": "srvtoolu_01E6F7G8H9I0J1K2L3M4N5O6",
    "content": {
      "type": "text_editor_code_execution_result",
      "oldStart": 3,
      "oldLines": 1,
      "newStart": 3,
      "newLines": 1,
      "lines": ["-  \"debug\": true", "+  \"debug\": false"]
    }
  }
```

Results

All execution results include:

- `stdout` : Output from successful execution
- `stderr` : Error messages if execution fails
- `return_code` : 0 for success, non-zero for failure

Additional fields for file operations:

- **View**: `file_type` , `content` , `numLines` , `startLine` , `totalLines`
- **Create**: `is_file_update` (whether file already existed)
- **Edit**: `oldStart` , `oldLines` , `newStart` , `newLines` , `lines` (diff format)

Errors

Each tool type can return specific errors:

Common errors (all tools):

```
    "tool_use_id": "srvtoolu_01VfmxgZ46TiHbmXgy928hQR",
    "content": {
      "type": "bash_code_execution_tool_result_error",
      "error_code": "unavailable"
    }
  }
```

Error codes by tool type:

Tool	Error Code	Description
All tools	`unavailable`	The tool is temporarily unavailable
All tools	`execution_time_exceeded`	Execution exceeded maximum time limit
All tools	`container_expired`	Container expired and is no longer available
All tools	`invalid_tool_input`	Invalid parameters provided to the tool
All tools	`too_many_requests`	Rate limit exceeded for tool usage
text_editor	`file_not_found`	File doesn't exist (for view/edit operations)
text_editor	`string_not_found`	The `old_str` not found in file (for str_replace)

`pause_turn` stop reason

The response may include a `pause_turn` stop reason, which indicates that the API paused a long-running turn. You may provide the response back as-is in a subsequent request to let Claude continue its turn, or modify the content if you wish to interrupt the conversation.

Containers

The code execution tool runs in a secure, containerized environment designed specifically for code execution, with a higher focus on Python.

Runtime environment

- **Python version**: 3.11.12
- **Operating system**: Linux-based container
- **Architecture**: x86_64 (AMD64)

- **Memory**: 5GiB RAM
- **Disk space**: 5GiB workspace storage
- **CPU**: 1 CPU

Networking and security

- **Internet access**: Completely disabled for security
- **External connections**: No outbound network requests permitted
- **Sandbox isolation**: Full isolation from host system and other containers
- **File access**: Limited to workspace directory only
- **Workspace scoping**: Like Files, containers are scoped to the workspace of the API key
- **Expiration**: Containers expire 30 days after creation

Pre-installed libraries

The sandboxed Python environment includes these commonly used libraries:

- **Data Science**: pandas, numpy, scipy, scikit-learn, statsmodels
- **Visualization**: matplotlib, seaborn
- **File Processing**: pyarrow, openpyxl, xlsxwriter, xlrd, pillow, python-pptx, python-docx, pypdf, pdfplumber, pypdfium2, pdf2image, pdfkit, tabula-py, reportlab[pycairo], Img2pdf
- **Math & Computing**: sympy, mpmath
- **Utilities**: tqdm, python-dateutil, pytz, joblib, unzip, unrar, 7zip, bc, rg (ripgrep), fd, sqlite

Container reuse

You can reuse an existing container across multiple API requests by providing the container ID from a previous response. This allows you to maintain created files between requests.

Example

Claude Docs

```
import os
from anthropic import Anthropic

# Initialize the client
client = Anthropic(
    api_key=os.getenv("ANTHROPIC_API_KEY")
)

# First request: Create a file with a random number
response1 = client.beta.messages.create(
    model="claude-sonnet-4-5",
    betas=["code-execution-2025-08-25"],
    max_tokens=4096,
    messages=[{
        "role": "user",
        "content": "Write a file with a random number and save it to '/tmp/num
    }],
    tools=[{
        "type": "code_execution_20250825",
        "name": "code_execution"
    }]
)

# Extract the container ID from the first response
container_id = response1.container.id

# Second request: Reuse the container to read the file
response2 = client.beta.messages.create(
    container=container_id,  # Reuse the same container
    model="claude-sonnet-4-5",
    betas=["code-execution-2025-08-25"],
    max_tokens=4096,
    messages=[{
        "role": "user",
        "content": "Read the number from '/tmp/number.txt' and calculate its s
    }],
    tools=[{
        "type": "code_execution_20250825",
        "name": "code_execution"
    }]
)
```

Streaming

```
event: content_block_start
data: {"type": "content_block_start", "index": 1, "content_block": {"type": "s

// Code execution streamed
event: content_block_delta
data: {"type": "content_block_delta", "index": 1, "delta": {"type": "input_jsc

// Pause while code executes

// Execution results streamed
event: content_block_start
data: {"type": "content_block_start", "index": 2, "content_block": {"type": "c
```

Batch requests

You can include the code execution tool in the Messages Batches API. Code execution tool calls through the Messages Batches API are priced the same as those in regular Messages API requests.

Usage and pricing

Code execution tool usage is tracked separately from token usage. Execution time has a minimum of 5 minutes. If files are included in the request, execution time is billed even if the tool is not used due to files being preloaded onto the container.

Each organization receives 1,550 free hours of usage with the code execution tool per month. Additional usage beyond the first 1,550 hours is billed at $0.05 per hour, per container.

Upgrade to latest tool version

By upgrading to `code-execution-2025-08-25`, you get access to file manipulation and Bash capabilities, including code in multiple languages. There is no price difference.

What's changed

Component	Legacy	Current
Beta header	`code-execution-2025-05-`	`code-execution-2025-08-25`

Tool type	`code_execution_20250522`	`code_execution_20250825`
Capabilities	Python only	Bash commands, file operations
Response types	`code_execution_result`	`bash_code_execution_result` , `text_editor_code_execution_result`

Backward compatibility

- All existing Python code execution continues to work exactly as before
- No changes required to existing Python-only workflows

Upgrade steps

To upgrade, you need to make the following changes in your API requests:

1. **Update the beta header**:

```
- "anthropic-beta": "code-execution-2025-05-22"
+ "anthropic-beta": "code-execution-2025-08-25"
```

2. **Update the tool type**:

```
- "type": "code_execution_20250522"
+ "type": "code_execution_20250825"
```

3. **Review response handling** (if parsing responses programmatically):

   - The previous blocks for Python execution responses will no longer be sent
   - Instead, new response types for Bash and file operations will be sent (see Response Format section)

Programmatic tool calling

The code execution tool powers programmatic tool calling, which allows Claude to write code that calls your custom tools programmatically within the execution container. This enables efficient multi-tool workflows, data filtering before reaching Claude's context, and complex conditional logic.

```
# Enable programmatic calling for your tools
response = client.beta.messages.create(
    model="claude-sonnet-4-5",
    betas=["advanced-tool-use-2025-11-20"],
    max_tokens=4096,
    messages=[{
        "role": "user",
        "content": "Get weather for 5 cities and find the warmest"
    }],
    tools=[
        {
            "type": "code_execution_20250825",
            "name": "code_execution"
        },
        {
            "name": "get_weather",
            "description": "Get weather for a city",
            "input_schema": {...},
            "allowed_callers": ["code_execution_20250825"]  # Enable programma
        }
    ]
)
```

Learn more in the Programmatic tool calling documentation.

Using code execution with Agent Skills

The code execution tool enables Claude to use Agent Skills. Skills are modular capabilities consisting of instructions, scripts, and resources that extend Claude's functionality.

Learn more in the Agent Skills documentation and Agent Skills API guide.

Claude Docs

Customer support

Education

Financial services

Government

Life sciences

Partners

Amazon Bedrock

Google Cloud's Vertex AI

Use cases

Connectors

Customer stories

Engineering at Anthropic

Events

Powered by Claude

Service partners

Startups program

Company

Anthropic

Careers

Economic Futures

Research

News

Responsible Scaling Policy

Security and compliance

Transparency

Help and security

Availability

Status

Support

Discord

Terms and policies

Privacy policy

Responsible disclosure policy

Terms of service: Commercial

Terms of service: Consumer

Usage policy