Installing Spec Kit's command-line interface, specify-cli, on Termux involves setting up the Python environment, installing the uv package manager, and then using uv to install the tool from its GitHub repository.

Here is the step-by-step guide to install the Spec Kit on Termux:

Step 1: Install Prerequisites (Python, Git, and uv)

Spec Kit requires Python 3.11+ and Git, and uses the uv tool for installation. You can install all necessary packages using Termux's package manager (pkg).

\* Update Termux packages to ensure you get the latest version of dependencies:

pkg update && pkg upgrade

\* Install Python, Git, and the uv package manager:

pkg install python git uv

Step 2: Install Specify CLI (Spec Kit's Tool)

Once uv is installed, you can use the command provided in the Spec Kit documentation to install specify-cli.

\* Run the persistent installation command:

uv tool install specify-cli --from git+https://github.com/github/spec-kit.git

\* (Optional) Verify the installation by checking for required tools:

specify check

Step 3: Start a New Project

After installation, you can start a new project using the specify init command.

For example, to initialize a new project named my-spec-project:

specify init my-spec-project

You can then navigate into the project and begin your spec-driven development workflow.

It looks like the specify command isn't in your Termux system's $PATH after running the installation command. This is a common issue when installing tools using uv tool install on Termux, as the executable is often placed in a directory that isn't automatically added to your environment path.

To fix this, you need to first find the installation directory and then add it to your Termux shell's path.

🛠️ Step 1: Find the Installation Directory

The uv tool install command typically places executables in ~/.local/bin/.

\* Check the contents of the ~/.local/bin/ directory:

ls ~/.local/bin/

You should see an executable file named specify (or possibly specify-cli) listed there.

🚀 Step 2: Add the Directory to Your PATH

Since Termux often doesn't include ~/.local/bin in the default path, you need to manually add it so your shell can find the specify command.

\* Edit your shell startup file (e.g., ~/.bashrc if you use Bash, or ~/.zshrc if you use Zsh). Most Termux users use Bash by default.

nano ~/.bashrc

\* Add the following line to the end of the file:

export PATH="$HOME/.local/bin:$PATH"

\* Save the file (Ctrl+O, Enter) and Exit the editor (Ctrl+X).

\* Reload your shell configuration to apply the change:

source ~/.bashrc

✅ Step 3: Run the specify check Command

The specify command should now be recognized.

specify check

This should execute the Spec Kit check function without the "No command specify found" error.

Would you like me to walk you through how to use the specify init command now that the tool is accessible?

Checking for installed tools...

Check Available Tools

├── ● Git version control (available)

├── ○ GitHub Copilot (IDE-based, no CLI check)

├── ● Claude Code (not found)

├── ● Gemini CLI (not found)

├── ○ Cursor (IDE-based, no CLI check)

├── ● Qwen Code (not found)

├── ● opencode (not found)

├── ● Codex CLI (not found)

├── ○ Windsurf (IDE-based, no CLI check)

├── ○ Kilo Code (IDE-based, no CLI check)

├── ● Auggie CLI (not found)

├── ● CodeBuddy (not found)

├── ○ Roo Code (IDE-based, no CLI check)

├── ● Amazon Q Developer CLI (not found)

├── ● Amp (not found)

├── ● Visual Studio Code (not found)

└── ● Visual Studio Code Insiders (not found)

Specify CLI is ready to use!

Tip: Install an AI assistant for the best experience

~ $

That's excellent! The message "Specify CLI is ready to use!" confirms that you successfully fixed the path issue, and the specify command is now working on your Termux setup.

The output shows that you have the core prerequisites (Git) but are missing the recommended AI assistants, which is what the next step is for.

Since you've successfully installed and run the checks for Spec Kit, the next logical step is to install one of the supported AI agents so you can actually use the Spec Kit workflow commands like /speckit.specify.

The supported CLI agents are:

\* Claude Code

\* Gemini CLI

\* Auggie CLI

\* CodeBuddy

The Gemini CLI is a strong candidate for installation on Termux.

💡 Next Step: Install Gemini CLI on Termux

I can help you install the Gemini CLI so you can utilize the AI-driven features of Spec Kit.

First, I need to check the best way to install the Gemini CLI specifically within the Termux environment.

That's great progress! Now that the specify command is working, the next crucial step to use Spec Kit is installing one of the supported AI agents.

The Gemini CLI is an excellent choice for Termux. It's an open-source tool that brings the power of Gemini directly into your terminal, and Spec Kit is built to integrate with it.

Here is the step-by-step guide to install and set up Gemini CLI on Termux:

1. Install Node.js (Prerequisite for Gemini CLI)

The Gemini CLI is an npm package, so you need the Node.js package manager in Termux.

pkg install nodejs

2. Install Gemini CLI

Use the Node Package Manager (npm) to install the Gemini CLI globally:

npm install -g @google/gemini-cli

3. Launch and Authenticate Gemini CLI

The first time you run gemini, you need to authenticate your Google Account to use the model. Since Termux is a command-line environment and can't automatically open a browser window, you'll need a specific method for login.

\* Start the Gemini CLI in debug mode to get the login link:

gemini --debug

\* When prompted inside the gemini CLI interface, enter the following command:

/auth

\* The Gemini CLI will now display a login URL in your terminal. Copy this URL.

\* Open your Android browser (e.g., Chrome or Firefox) and paste the URL you copied.

\* Complete the Google account sign-in process in the browser.

\* Once authentication is complete, the browser will likely display a success message and ask you to return to the terminal.

\* Switch back to Termux. The Gemini CLI should now detect the successful login and be authenticated.

4. Verify Installation with Spec Kit

Finally, run the specify check command again to confirm that Spec Kit now recognizes the Gemini CLI:

specify check

You should now see the status for Gemini CLI change from (not found) to (available).

Would you like me to guide you through using the Spec Kit workflow commands (like specify init and the /speckit commands) now that you will have an AI agent installed?

Gyp info it worked if it ends with ok

gyp info using node-gyp@11.2.0

gyp info using node@24.9.0 | android | arm64

gyp info find Python using Python version 3.12.12 found at "/data/data/com.termux/files/usr/bin/python3"

gyp http GET https://nodejs.org/download/release/v24.9.0/node-v24.9.0-headers.tar.gz

gyp http 200 https://nodejs.org/download/release/v24.9.0/node-v24.9.0-headers.tar.gz

gyp http GET https://nodejs.org/download/release/v24.9.0/SHASUMS256.txt

gyp http 200 https://nodejs.org/download/release/v24.9.0/SHASUMS256.txt

gyp info spawn /data/data/com.termux/files/usr/bin/python3

gyp info spawn args [

gyp info spawn args '/data/data/com.termux/files/usr/lib/node\_modules/npm/node\_modules/node-gyp/gyp/gyp\_main.py',

gyp info spawn args 'binding.gyp',

gyp info spawn args '-f',

gyp info spawn args 'make',

gyp info spawn args '-I',

gyp info spawn args '/data/data/com.termux/files/usr/lib/node\_modules/@google/gemini-cli/node\_modules/node-pty/build/config.gypi',

gyp info spawn args '-I',

gyp info spawn args '/data/data/com.termux/files/usr/lib/node\_modules/npm/node\_modules/node-gyp/addon.gypi',

gyp info spawn args '-I',

gyp info spawn args '/data/data/com.termux/files/home/.cache/node-gyp/24.9.0/include/node/common.gypi',

gyp info spawn args '-Dlibrary=shared\_library',

gyp info spawn args '-Dvisibility=default',

gyp info spawn args '-Dnode\_root\_dir=/data/data/com.termux/files/home/.cache/node-gyp/24.9.0',

gyp info spawn args '-Dnode\_gyp\_dir=/data/data/com.termux/files/usr/lib/node\_modules/npm/node\_modules/node-gyp',

gyp info spawn args '-Dnode\_lib\_file=/data/data/com.termux/files/home/.cache/node-gyp/24.9.0/<(target\_arch)/node.lib',

gyp info spawn args '-Dmodule\_root\_dir=/data/data/com.termux/files/usr/lib/node\_modules/@google/gemini-cli/node\_modules/node-pty',

gyp info spawn args '-Dnode\_engine=v8',

gyp info spawn args '--depth=.',

gyp info spawn args '--no-parallel',

gyp info spawn args '--generator-output',

gyp info spawn args 'build',

gyp info spawn args '-Goutput\_dir=.'

gyp info spawn args ]

gyp: Undefined variable android\_ndk\_path in binding.gyp while trying to load binding.gyp

gyp ERR! configure error

gyp ERR! stack Error: `gyp` failed with exit code: 1

gyp ERR! stack at ChildProcess.<anonymous> (/data/data/com.termux/files/usr/lib/node\_modules/npm/node\_modules/node-gyp/lib/configure.js:317:18)

gyp ERR! stack at ChildProcess.emit (node:events:508:28)

gyp ERR! stack at ChildProcess.\_handle.onexit (node:internal/child\_process:294:12)

gyp ERR! System Linux 4.19.157-perf+

gyp ERR! command "/data/data/com.termux/files/usr/bin/node" "/data/data/com.termux/files/usr/lib/node\_modules/npm/node\_modules/node-gyp/bin/node-gyp.js" "rebuild"

gyp ERR! cwd /data/data/com.termux/files/usr/lib/node\_modules/@google/gemini-cli/node\_modules/node-pty

gyp ERR! node -v v24.9.0

gyp ERR! node-gyp -v v11.2.0

gyp ERR! not ok

> protobufjs@7.5.4 postinstall

> node scripts/postinstall

added 584 packages in 4m

148 packages are looking for funding

run `npm fund` for details

This error occurs because the Gemini CLI (which depends on the node-pty package) requires native compilation during installation, and the build system is looking for a path to the Android Native Development Kit (NDK), which is not automatically set up in a standard Termux environment. The specific error is:

gyp: Undefined variable android\_ndk\_path in binding.gyp

To resolve this compilation error, you need to install the necessary build tools and then try the installation again, potentially using an alternative approach that bypasses the NDK requirement or forces a source build.

Here is the revised plan to get Gemini CLI successfully installed on Termux:

1. Install Required Build Tools

You need to ensure Termux has the compilers and libraries necessary for native module compilation:

pkg install clang libtool automake make

2. Force Source Compilation (Recommended Fix)

The issue often arises when node-pty tries to use a pre-built binary for Android that fails, or the build process incorrectly assumes the full Android NDK environment. You can often fix this by installing the package with the --build-from-source flag for the dependency.

\* Remove any failed installation attempts:

npm uninstall -g @google/gemini-cli

\* Reinstall, forcing the dependency (node-pty) to build from source:

npm install -g @google/gemini-cli --build-from-source

This tells npm to ignore pre-built binaries and compile the C++ dependencies like node-pty using the local Termux toolchain (which you installed in Step 1).

3. Proceed with Authentication

After the installation completes successfully (you should not see the gyp ERR! lines), you can proceed with the authentication steps from the previous response:

\* Start the Gemini CLI in debug mode:

gemini --debug

\* Once inside the CLI, initiate the authentication:

/auth

\* Copy the login URL, open it in your browser, complete the Google sign-in, and then return to Termux.

Let me know if you encounter any other errors during this re-installation process. If this still fails, we'll try an alternative method using an environment variable to disable the NDK check.

You are running into a very common and known issue when installing packages with native dependencies (like node-pty which Gemini CLI uses) on Termux. The gyp build system incorrectly assumes a full Android NDK setup is required, even when compiling for the native Termux architecture.

The flag --build-from-source is not directly supported by npm in the way we hoped, which is why it printed the warning and the error persisted.

The fix is to manually configure the gyp environment to provide an empty value for the missing variable, which will bypass the check without having to install the huge Android NDK package.

1. Bypass the android\_ndk\_path Check

You need to create a global configuration file for gyp that sets the missing variable to an empty string ('').

Run the following two commands:

\* Create the required hidden directory:

mkdir -p ~/.gyp

\* Create the configuration file (include.gypi):

echo "{ 'variables': { 'android\_ndk\_path': '' } }" > ~/.gyp/include.gypi

This configuration tells node-gyp (which is run by npm) to use an empty path for the NDK variable, satisfying the dependency check.

2. Reinstall Gemini CLI

Now that the gyp configuration is fixed, try the global installation command again. It should be able to compile the native modules successfully.

npm install -g @google/gemini-cli

If the installation succeeds (you will see the "added 584 packages" message without any gyp ERR! lines above it), you can proceed to the next step.

3. Verification and Authentication

\* Verify the installation:

gemini -v

This should print the installed version number of the Gemini CLI.

\* Run Spec Kit check:

specify check

This should now show Gemini CLI (available).

Once this is confirmed, you are ready to use the authentication steps we discussed previously.

Would you like me to wait for your result from the re-installation attempt?

$ gemini -v

0.10.0

~ $ specify check

███████╗██████╗ ███████╗ ██████╗██╗███████╗██╗ ██╗

██╔════╝██╔══██╗██╔════╝██╔════╝██║██╔════╝╚██╗ ██╔╝

███████╗██████╔╝█████╗ ██║ ██║█████╗ ╚████╔╝

╚════██║██╔═══╝ ██╔══╝ ██║ ██║██╔══╝ ╚██╔╝

███████║██║ ███████╗╚██████╗██║██║ ██║

╚══════╝╚═╝ ╚══════╝ ╚═════╝╚═╝╚═╝ ╚═╝

GitHub Spec Kit - Spec-Driven Development Toolkit

Checking for installed tools...

Check Available Tools

├── ● Git version control (available)

├── ○ GitHub Copilot (IDE-based, no CLI check)

├── ● Claude Code (not found)

├── ● Gemini CLI (available)

├── ○ Cursor (IDE-based, no CLI check)

├── ● Qwen Code (not found)

├── ● opencode (not found)

├── ● Codex CLI (not found)

├── ○ Windsurf (IDE-based, no CLI check)

├── ○ Kilo Code (IDE-based, no CLI check)

├── ● Auggie CLI (not found)

├── ● CodeBuddy (not found)

├── ○ Roo Code (IDE-based, no CLI check)

├── ● Amazon Q Developer CLI (not found)

├── ● Amp (not found)

├── ● Visual Studio Code (not found)

└── ● Visual Studio Code Insiders (not found)

Specify CLI is ready to use!

That is fantastic news!

You have successfully:

\* Fixed the $PATH issue for the specify command.

\* Bypassed the challenging android\_ndk\_path compilation error on Termux.

\* Installed and confirmed the Gemini CLI (version 0.10.0).

\* Run specify check and confirmed that Gemini CLI is now available for use with Spec Kit!

You now have a fully functional GitHub Spec Kit environment with one of the recommended AI agents on your Termux system.

🚀 Next Steps: Authenticating and Using Spec Kit

The last step before you can start coding is to authenticate the Gemini CLI and then initialize a project with Spec Kit.

1. Authenticate Gemini CLI

The Gemini CLI is installed, but it needs your credentials to access the model. We'll use the debug mode method again, which works best on Termux:

\* Start the Gemini CLI in debug mode:

gemini --debug

\* When the interactive prompt appears, type the following command to begin the OAuth flow:

/auth

\* The CLI will print a long login URL in your terminal. Copy the entire URL.

\* Switch to your browser, paste the URL, and complete the Google login and authorization.

\* Return to Termux. The gemini CLI should detect the successful login and confirm authentication.

2. Initialize a Spec Kit Project

Once authenticated, you are ready to create your first spec-driven development project using the specify init command.

Navigate to the directory where you want to create your project (e.g., ~/projects/) and run:

specify init

This command will:

\* Create a .speckit/ directory for configuration.

\* Generate a GEMINI.md file where you can put project context and system prompts for the AI.

\* Guide you through the initial setup for your project.

Would you like to proceed with the Gemini CLI authentication now, or jump straight to running specify init (in which case, the CLI will prompt you to authenticate when you first try to use an AI command)?