Data & AI Community Day Durban (/tags/data-ai-community-day-durban) Generative AI (/tags/generative-ai) Claude Code (/tags/claude-code) Anthropic (/tags/anthropic) AI-assisted Coding (/tags/ai-assisted-coding) Event Management System (/tags/event-management-system)

Building an Event Management System with Claude Code: Claude Code Installation and Initialisation

Posted by nielsb on Tuesday, July 29, 2025

Claude Code Version: This post is based on Claude Code v2.x (December 2025). Installation methods may vary for earlier versions.

> ***Editor's Note (December 2025)****: This post has been updated to reflect the latest Claude Code installation methods. Claude Code can now be installed directly on Windows without WSL or Git for Windows, and no longer requires `Node.js` for installation. However, we still cover `Node.js` installation as it will be used throughout this series for our event management system development.*

If you follow my blog or LinkedIn, you know I'm a co-organiser of the **Data & AI Community Day Durban** (https://aimldatadurban.org/) events. Juggling the practical challenges of organising events can be rewarding. Still, I'll admit that managing contacts and logistics comes with its share of headaches. If you've faced similar struggles, you're not alone.

This post marks the launch of a series detailing my journey to develop an AI-driven contact and event management system.

- To see what other posts there are in the series, go to: **Building an Event Management System with Claude Code** (/contact-event-claude-code).

The goal is to efficiently manage contacts, streamline communications, and generate actionable reports for the **Data & AI Community Day Durban** (https://aimldatadurban.org/) events. At **Derivco**, as an Architect Lead, my team leverages AI to accelerate development and tackle complexity. I want to use **Claude Code** to apply these principles to this project and share the process here.

Problem Statement

Brevo (https://www.brevo.com/) (formerly Sendinblue) has been our event and contact management system for our **Data & AI Community Day Durban** events for a couple of years now, and it has been satisfactory. However, how we want to do things may not match fully how Brevo does it, so limitations have become apparent:

- Workflows are too generic for our events.
- Limited flexibility for managing contacts and participants.
- Reporting lacks the necessary insights for our needs.

The limitations above are some of the reasons I'm developing a new system. I'll explore these issues and their impact in more detail in a future post.

Why Claude Code?

Suppose you're not familiar with Claude Code. In that case, it's Anthropic's agentic command-line tool that allows developers to delegate coding tasks directly from their terminal. Think of it as having an AI pair programmer who can understand context, write code, run tests, and even help with debugging—all from your command line. The appeal for me is multifaceted:

- **Speed of development:** Instead of spending hours researching APIs or writing boilerplate code, I can focus on the business logic and user experience
- **Learning opportunity:** I can explore new technologies and patterns with AI assistance
- **Documentation:** Every interaction is naturally documented, making it easier to understand decisions later
- **Quality assurance:** AI can help catch potential issues early in the development process

So, what am I aiming for with this project?

Project Goals

Before diving into the installation, let me outline what I'm planning to build:

Core Features

- Event creation and management
- Contact management with segmentation
- Participant registration with custom fields
- Automated email communications
- Comprehensive reporting and analytics

The features may be developed in stages. Initially, communications and reporting/analytics might come later.

Technical Goals

Apart from leveraging Claude Code for rapid development, I have some technical goals for this project:

- Modern, responsive web application
- Clean, maintainable codebase
- Robust testing strategy
- Scalable architecture
- Security best practices

Documentation Goals

I want to document the entire process, from initial setup to deployment, so that others can learn from my experience and benefit from it. This includes:

- Share the entire development journey
- Demonstrate practical AI-assisted coding
- Provide insights for other developers
- Create a resource for event organizers

Installing Claude Code

To get started with Claude Code, you'll need to install it on your machine (duh). The Claude Code installation is now straightforward on all platforms. The tool is self-contained and doesn't require `Node.js` or other runtime dependencies for Claude Code itself to function.

> ***NOTE:*** *While Claude Code itself doesn't require `Node.js` any more, we'll be installing `Node.js` later in this post. We'll install it as we will be using it a bit later in this post, and also potentially as part of the event management system we are building.*

Prerquisites

Before installing Claude Code, ensure you have the following prerequisites:

- `curl` for file downloads (most likely pre-installed on both macOS and Windows).
- A **Claude.ai** (https://claude.com/product/claude-code) subscription, or a **Claude Console** (https://console.anthropic.com/) account.

For Claude.ai, you need a paid sucbscription to use Claude Code. If you have a Claude Console account, you can use an API key to authenticate Claude Code.

Installing Claude Code on Windows

Windows users now have a simple installation path - no WSL or Git for Windows required!

To install Claude Code on Windows you use the PowerShell `irm` (`Invoke-RestMethod`) command:

```
irm https://claude.ai/install.ps1 | iex
```

Code Snippet 1: *Install Claude Code on Windows*

The `irm` command downloads downloads the installation script from the Claude Code website and pipes it to `iex` (`Invoke-Expression`), which runs the script.

Claude Code is now installed on your Windows machine. Jump to the Verification section to complete the setup.

Installing Claude Code on macOS

The recommended (by Anthropic) way to install Claude Code on macOS is using `curl`:

```
curl -o- https://claude.ai/install.sh | bash
```

Code Snippet 2: *Using curl to Install Claude Code on macOS*

As with the Windows installation, the `curl` command downloads the installation script from the Claude Code website and pipes it to `bash`, which runs the script.

Even though using `curl` is the recommended way to install Claude Code on macOS, I prefer to use Homebrew:

```
brew install --cask claude-code
```

Code Snippet 3: *Using Homebrew to Install Claude Code on macOS*

With Claude Code now installed on either your Windows or macOS machine, proceed to the next section to verify the installation.

Verification

Let's verify that Claude Code is installed correctly.

Verifying Installation

Whether on Windows or on macOS, you can verify the installation by running the following command in your terminal (Command Prompt or PowerShell on Windows, Terminal on macOS):

```
claude --version
```

Code Snippet 4: *Verify Claude Code Installation*

If the installation was successful, you should see the version number of Claude Code displayed, and you are ready to use Claude Code for the first time.

Initialising Claude Code

When you start Claude Code for the first time, it performs an initial setup, which we will examine in this section.

1. In your terminal (on Windows, use the WSL/distro terminal), create a directory to test Claude Code:

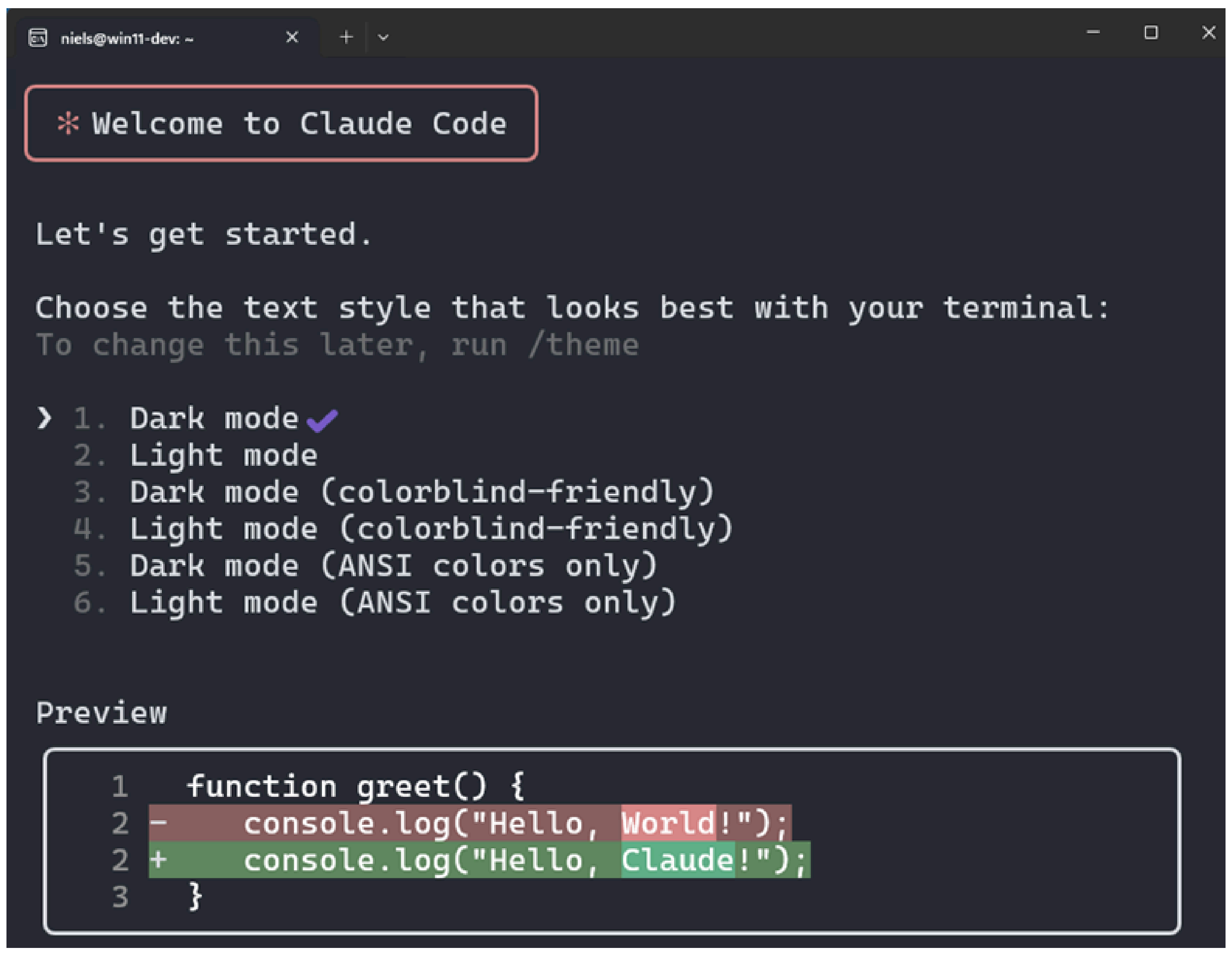
   ```
   mkdir claude-test
   cd claude-test
   ```

   Code Snippet 5: *Create Project Directory*
2. In the directory you `cd`:ed into, start Claude Code:

   ```
   claude
   ```

   Code Snippet 6: *Start Claude Code*

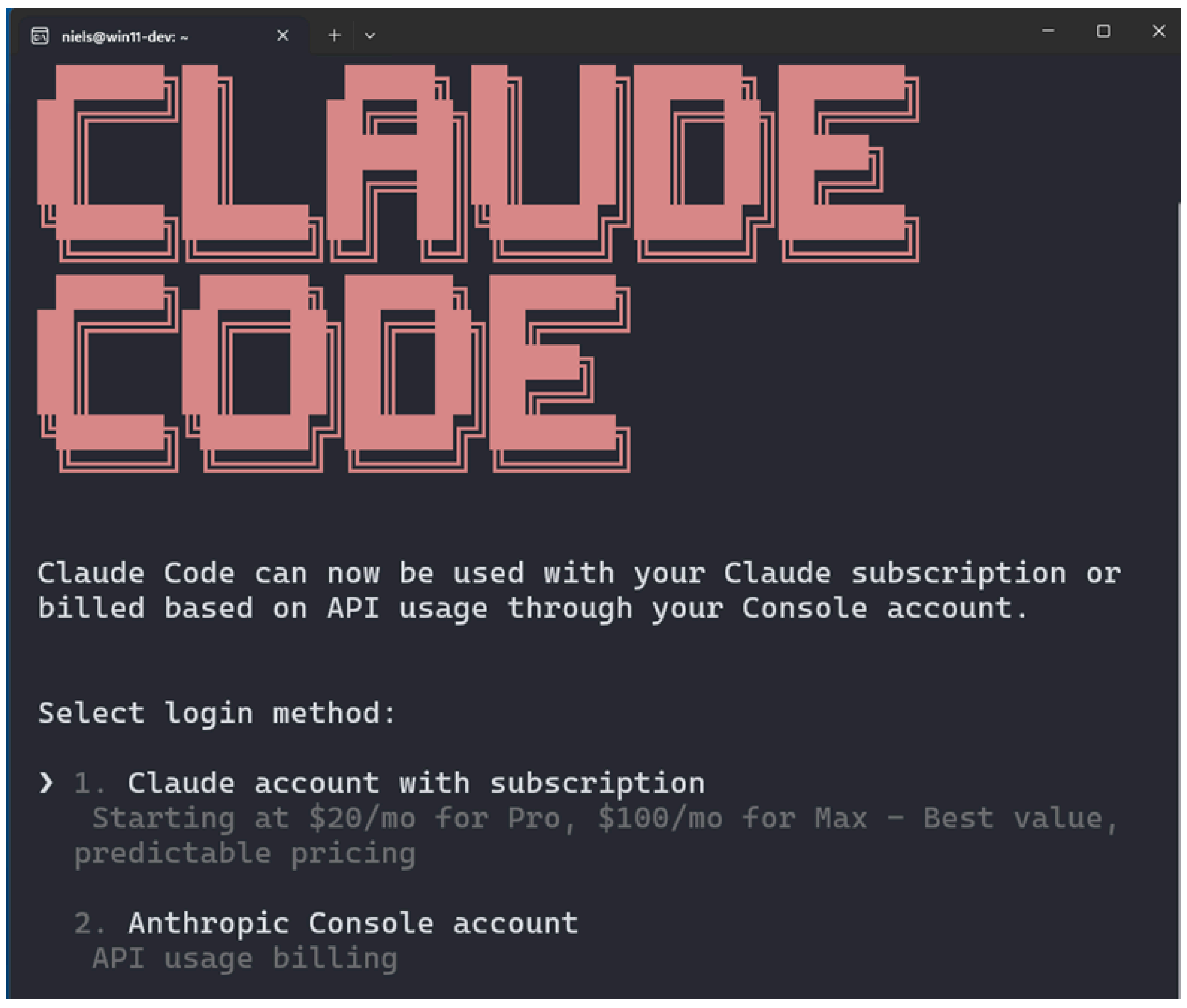
When you run the command in *Code Snippet 6*, it starts Claude Code's interactive shell. If this is the first time you run it, you will be prompted to perform some initial configuration and log in to Anthropic. You will see output similar to the following:



```
niels@win11-dev: ~

✻ Welcome to Claude Code

Let's get started.

Choose the text style that looks best with your terminal:
To change this later, run /theme

❯ 1. Dark mode ✓
  2. Light mode
  3. Dark mode (colorblind-friendly)
  4. Light mode (colorblind-friendly)
  5. Dark mode (ANSI colors only)
  6. Light mode (ANSI colors only)


Preview
   1   function greet() {
   2 -     console.log("Hello, World!");
   2 +     console.log("Hello, Claude!");
   3   }
```

Figure 1: *Claude Code Text Style*

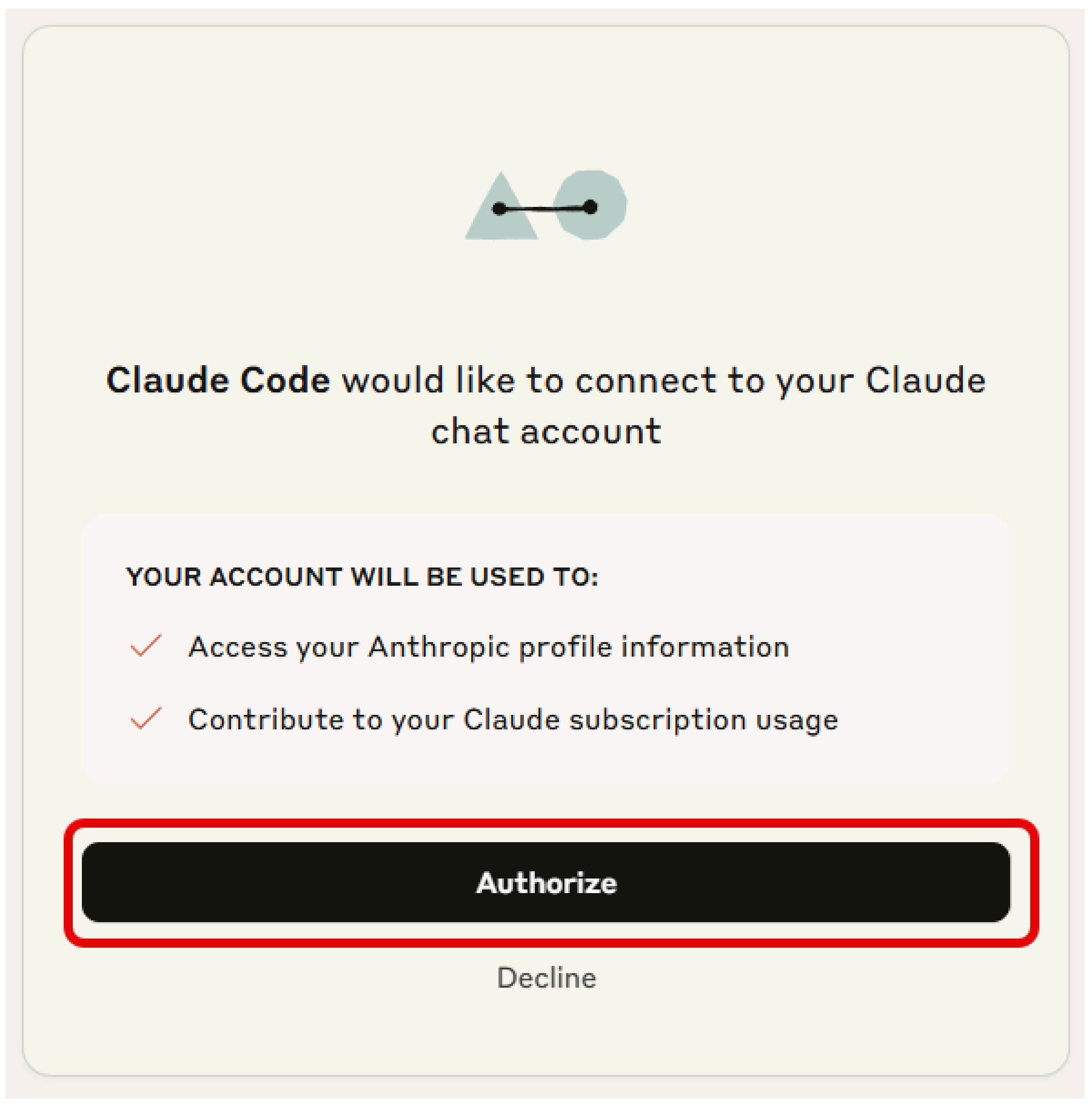
As you see in *Figure 1*, Claude Code asks you to choose a text style. After selecting a text style, you will be asked how to log in to Anthropic:



```
niels@win11-dev: ~

CLAUDE
CODE

Claude Code can now be used with your Claude subscription or
billed based on API usage through your Console account.


Select login method:

❯ 1. Claude account with subscription
     Starting at $20/mo for Pro, $100/mo for Max - Best value,
    predictable pricing

  2. Anthropic Console account
     API usage billing
```

Figure 2: *Claude Code Login Prompt*

Choose whether you want to log in using a subscription or an API key. If you choose the subscription option, it will open your default web browser and ask you to authorize Claude Code to connect to your Claude chat account:



Claude Code would like to connect to your Claude chat account

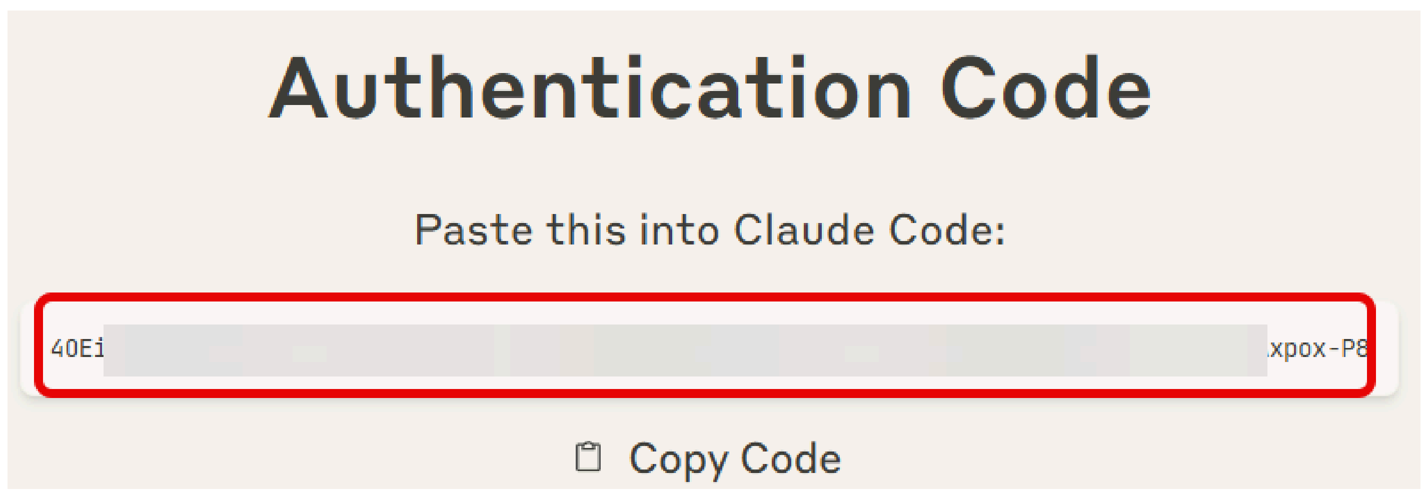
YOUR ACCOUNT WILL BE USED TO:

✓ Access your Anthropic profile information

✓ Contribute to your Claude subscription usage

Figure 3: *Claude Code Authentication Prompt*

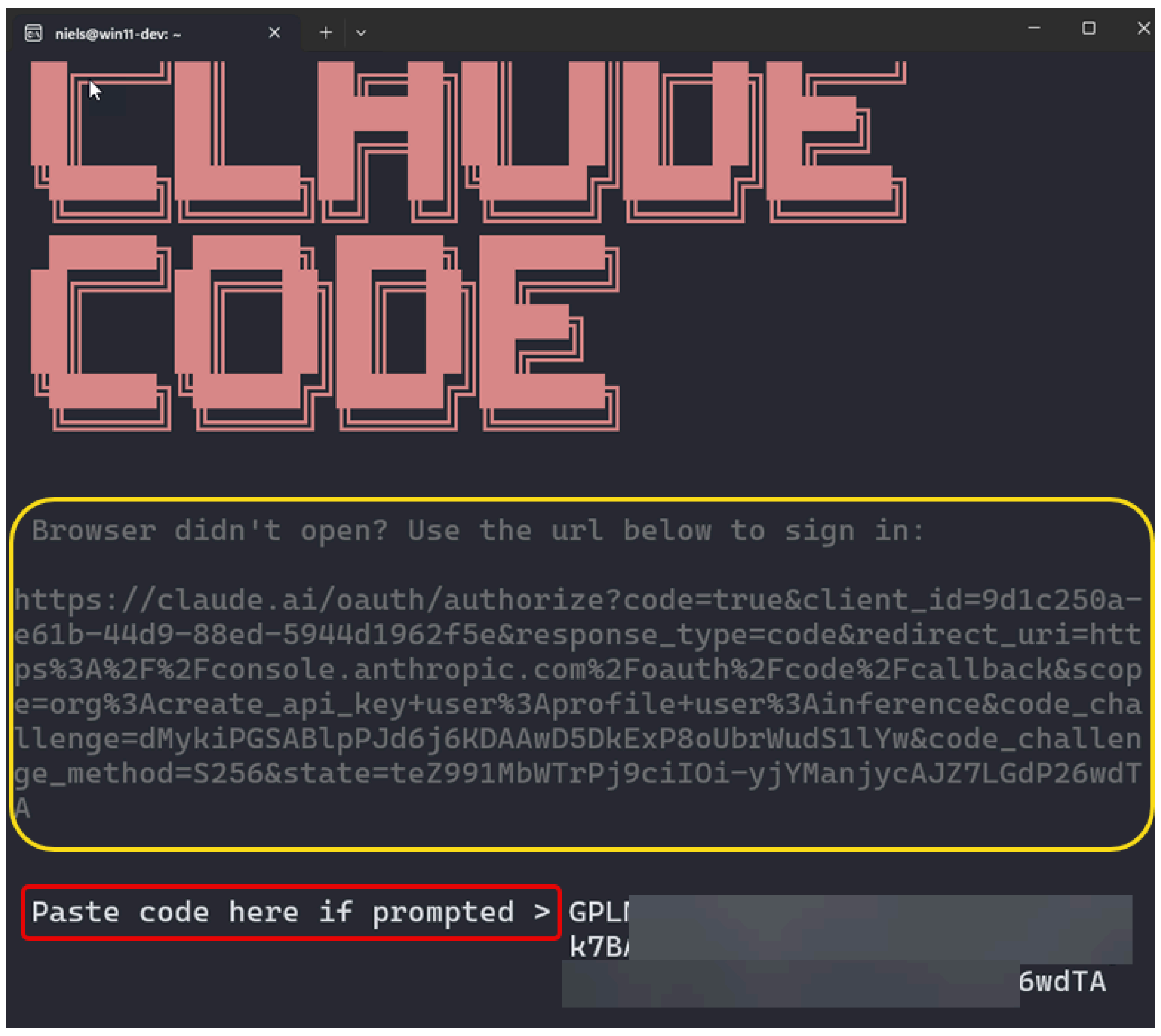
You just click on the **Authorize** button outlined in red in *Figure 3*. Clicking on the button authenticates you against your Claude subscription and opens a new page with an authentication code:



```
40Ei[redacted]xpox-P8
```

Figure 4: *Claude Code Authentication Code*

When you selected the subscription option as per *Figure 3*, the interactive shell changed to a form where you enter the authentication code:



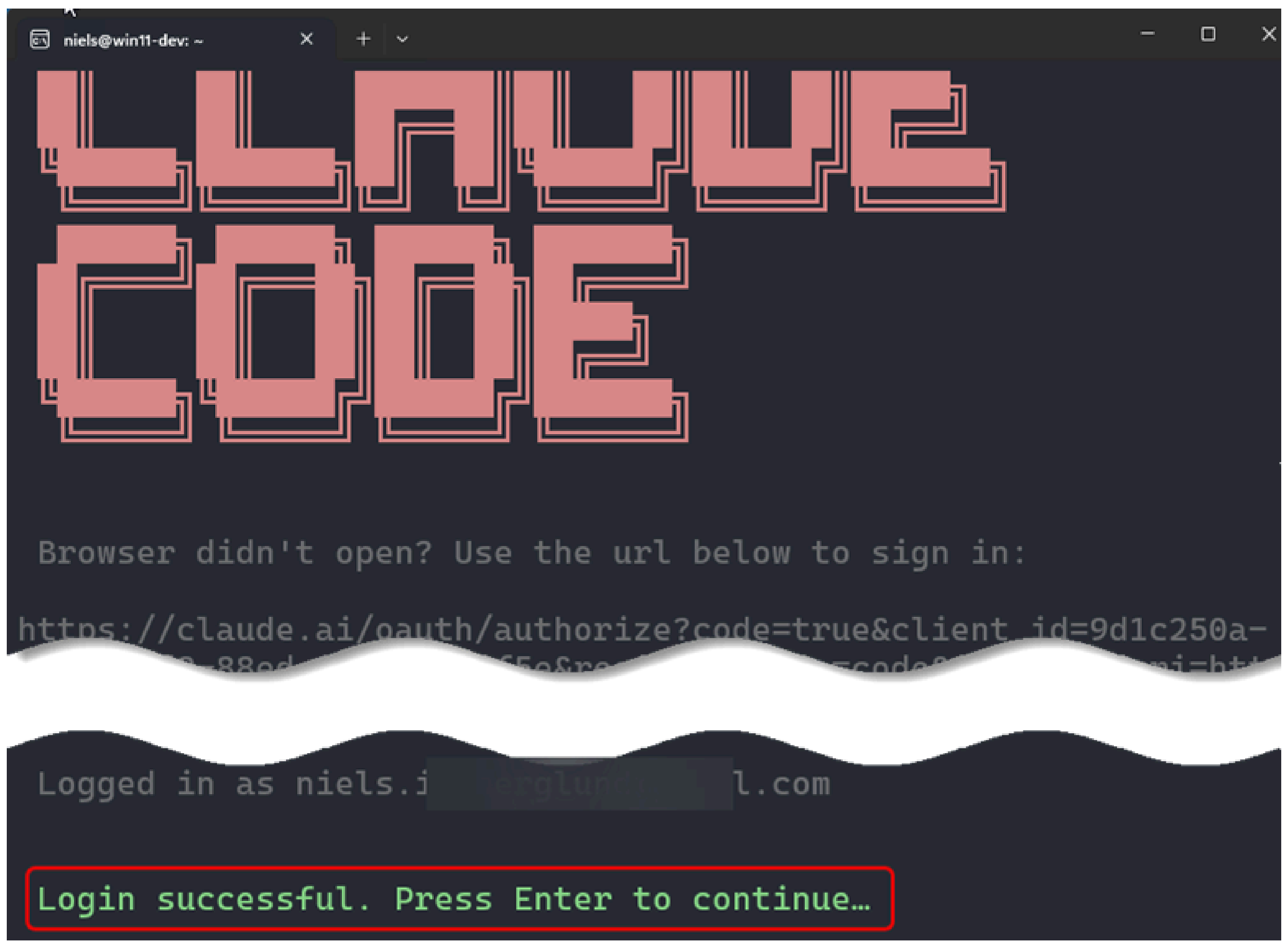
```
Browser didn't open? Use the url below to sign in:

https://claude.ai/oauth/authorize?code=true&client_id=9d1c250a-
e61b-44d9-88ed-5944d1962f5e&response_type=code&redirect_uri=htt
ps%3A%2F%2Fconsole.anthropic.com%2Foauth%2Fcode%2Fcallback&scop
e=org%3Acreate_api_key+user%3Aprofile+user%3Ainference&code_cha
llenge=dMykiPGSABlpPJd6j6KDAAwD5DkExP8oUbrWudS1lYw&code_challen
ge_method=S256&state=teZ991MbWTrPj9ciIOi-yjYManjycAJZ7LGdP26wdT
A

Paste code here if prompted > GPLI[redacted]
                               k7B/
                               [redacted]6wdTA
```

Figure 5: *Claude Code Interactive Shell Authentication*

Enter your authentication code that you received in the browser, and press `Enter` .

> ***NOTE:*** *If your browser doesn't open automatically (Figure 2), you can copy the URL from the terminal as in Figure 5 (outlined in yellow), open the URL in a browser. You will see the same as in Figure 3 and proceed from there.*

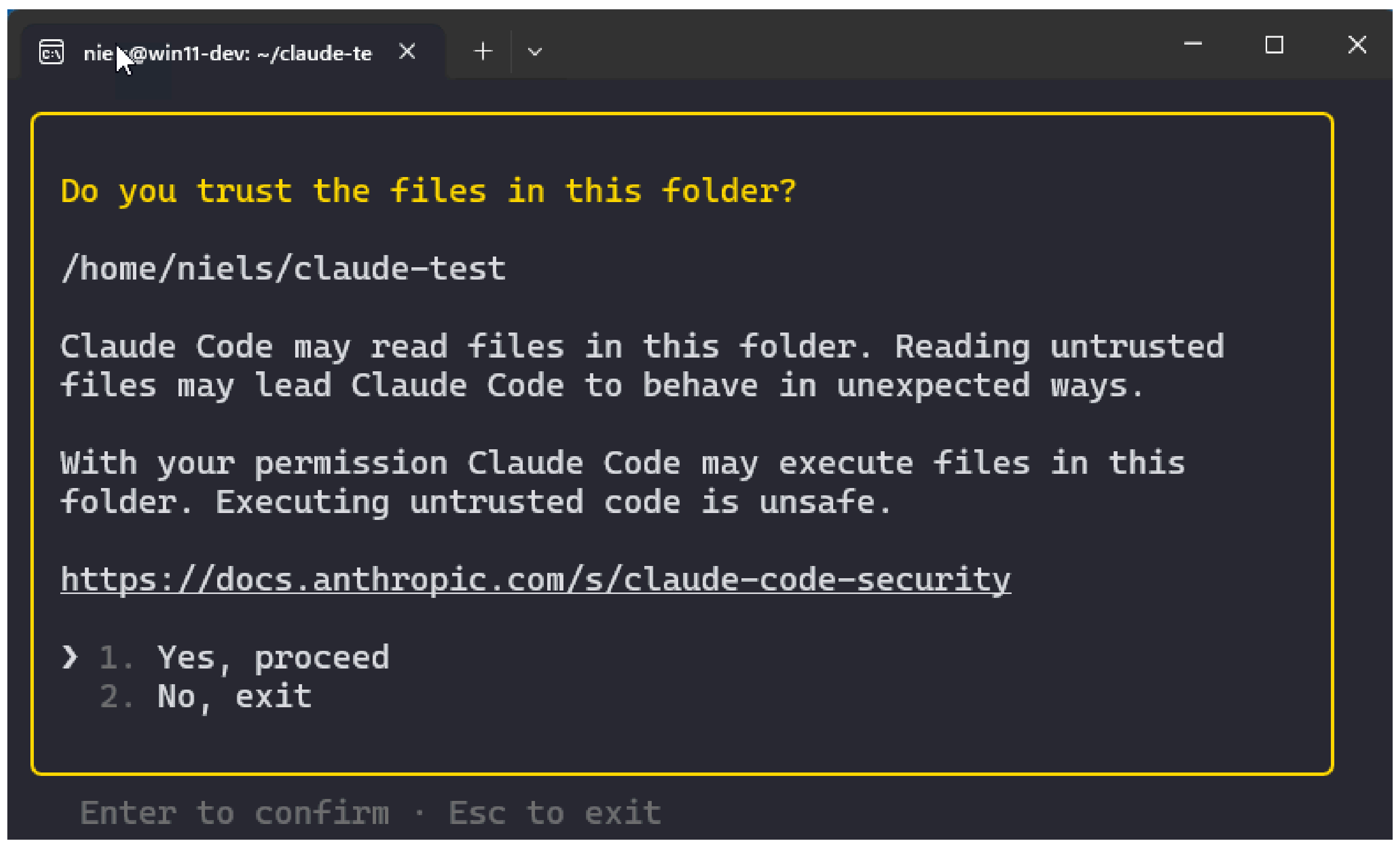
You see the following when you click `Enter` after entering the code:



```
Browser didn't open? Use the url below to sign in:

https://claude.ai/oauth/authorize?code=true&client_id=9d1c250a-

Logged in as niels.i[redacted]l.com

Login successful. Press Enter to continue…
```

Figure 6: *Claude Code Login Confirmation*

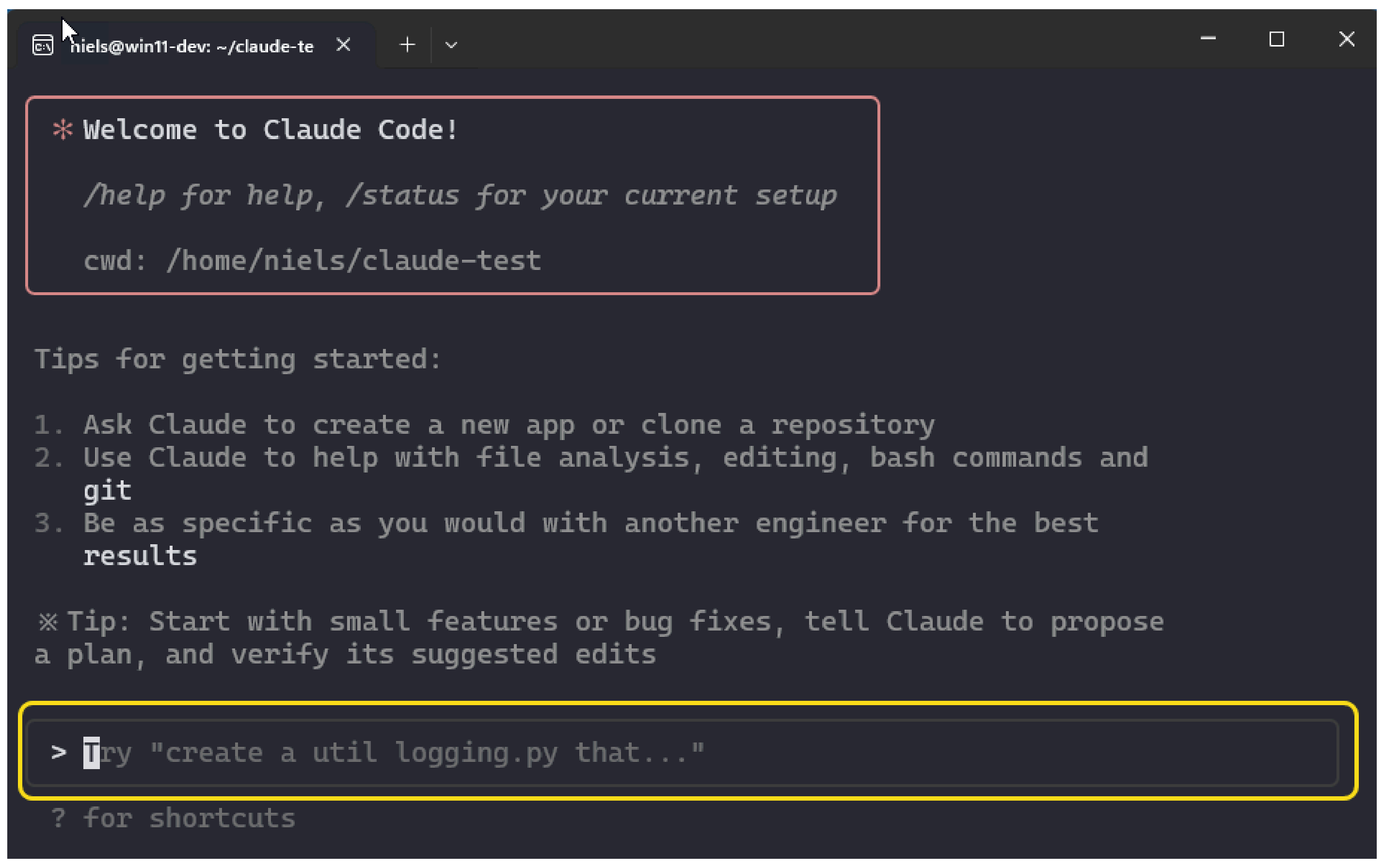
Having logged in successfully, you see a message confirming that you are logged in, as in *Figure 6* above, and when you press `Enter` again, you are almost there:


```
Do you trust the files in this folder?

/home/niels/claude-test

Claude Code may read files in this folder. Reading untrusted
files may lead Claude Code to behave in unexpected ways.

With your permission Claude Code may execute files in this
folder. Executing untrusted code is unsafe.

https://docs.anthropic.com/s/claude-code-security

❯ 1. Yes, proceed
  2. No, exit

Enter to confirm · Esc to exit
```
Figure 7: *Claude Code Trust Files*

In *Figure 7*, you are asked to trust the files in the current directory. This is a security measure to ensure that Claude Code can access the files it needs to work with.

> ***NOTE:*** *Whenever you start Claude Code in a new directory, you will be asked to trust the files in that directory.*

Press `Enter` to trust the files:


```
✻ Welcome to Claude Code!

  /help for help, /status for your current setup

  cwd: /home/niels/claude-test

Tips for getting started:

1. Ask Claude to create a new app or clone a repository
2. Use Claude to help with file analysis, editing, bash commands and
   git
3. Be as specific as you would with another engineer for the best
   results

※ Tip: Start with small features or bug fixes, tell Claude to propose
a plan, and verify its suggested edits

> Try "create a util logging.py that..."
  ? for shortcuts
```
Figure 8: *Claude Code CLI*

You are now in the Claude Code shell.

Before we look at using Claude Code, we will take a small detour to install `Node.js`, as we will be using it later in this post.

Installing Node.js

While Claude Code is now installed, let us set up `Node.js`, so we can use it later.

We have already said that Claude Code requires `Node.js` version 18 or newer. Since we may want to use different versions of `Node.js` in our development projects, I recommend using `nvm` (Node Version Manager) to manage `Node.js` versions. Here are the steps (in your Terminal/Powershell/Command shell):

1. Install `nvm` (Node Version Manager):

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
   ```

   Code Snippet 7: *Install nvm*

   Close and reopen your terminal to load `nvm`.

2. Verify that `nvm` is installed correctly:

   ```
   nvm -v
   ```

   Code Snippet 8: *Check nvm Version*

Having installed `nvm`, we can now install `Node.js`:

3. Install the latest long-term support (LTS) version of `Node.js`:

   ```
   nvm install --lts
   ```

   Code Snippet 9: *Install Node.js*

4. Verify that `Node.js` is installed correctly:

```
node -v
```

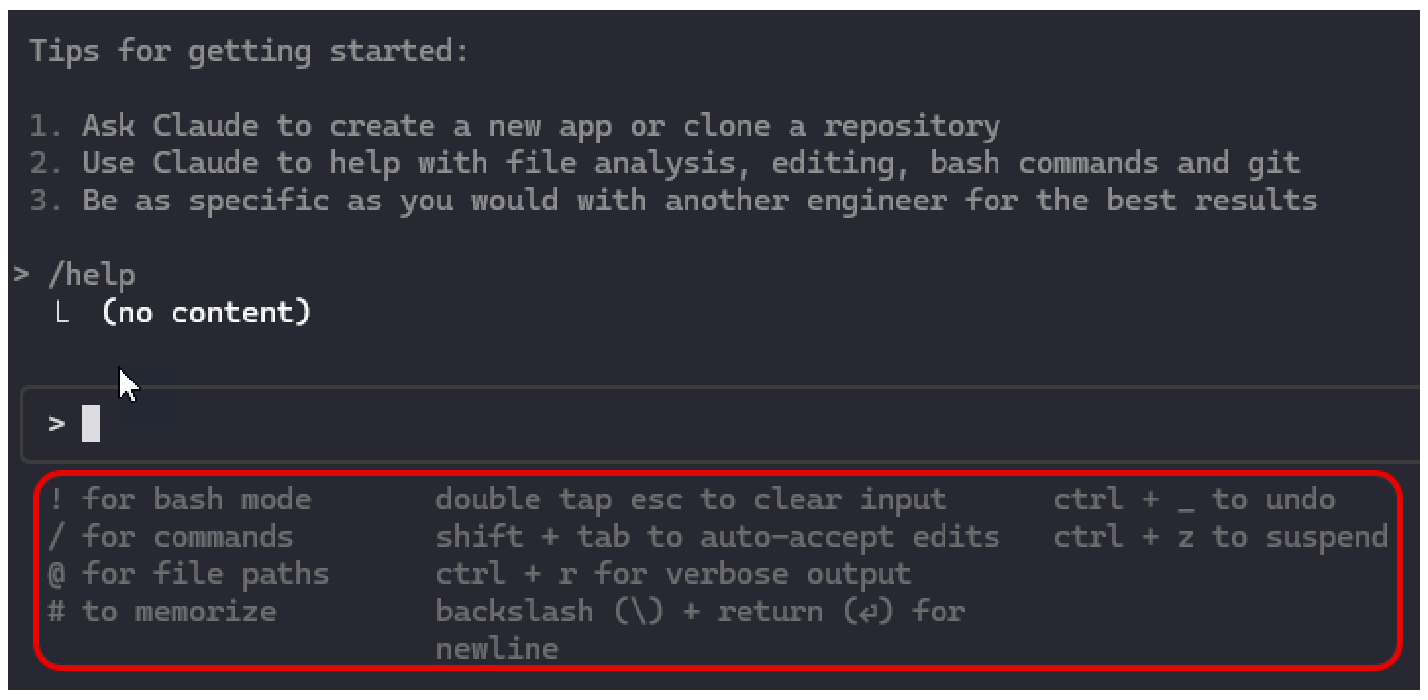
Code Snippet 10: *Verify Node.js Installation*

Using Claude Code

Claude Code is a powerful tool that can help you with various coding tasks. You can use it to generate code snippets, refactor code, write tests, and more. You can even ask it to create entire applications/systems based on your requirements. This is what we will do in subsequent posts in this series. For now, however, just do a couple of simple things to get a feel for how it works.

The assumption is that you are in the Claude Code interactive shell, as in *Figure 8*. If you are not, start it by running the command `claude` in your terminal from within the directory you want to run Claude Code.

Let's start by examining the available shortcuts. To list shortcuts, you enter `?` in the Claude Code's textbox (highlighted in yellow in *Figure 9*). That displays a list of shortcuts that you can use:

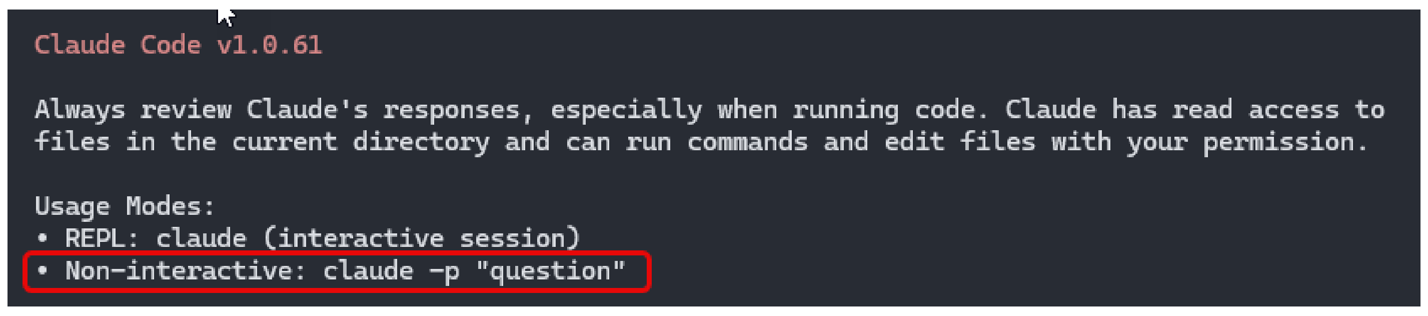


```
Tips for getting started:

1. Ask Claude to create a new app or clone a repository
2. Use Claude to help with file analysis, editing, bash commands and git
3. Be as specific as you would with another engineer for the best results

> /help
  └ (no content)

> 

! for bash mode        double tap esc to clear input      ctrl + _ to undo
/ for commands         shift + tab to auto-accept edits   ctrl + z to suspend
@ for file paths       ctrl + r for verbose output
# to memorize          backslash (\) + return (↵) for
                       newline
```

Figure 9: *Claude Code Shortcuts*

These shortcuts can help you navigate and use Claude Code more efficiently.

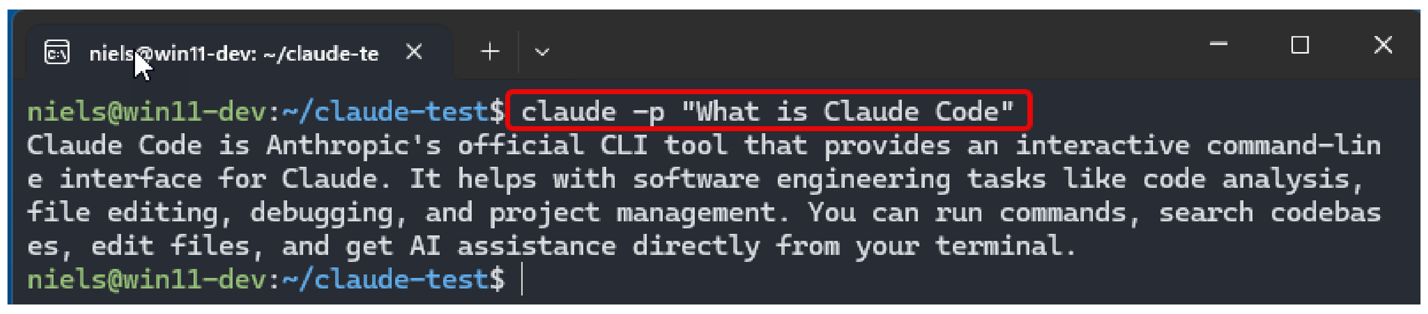
Let's try a simple command to get a feel for how Claude Code works. You can type `/help` in the input box and press `Enter`. This will display a list of commands that you can use. I will not discuss all the commands here. I leave that up to you to try out. However, I do want to highlight `Usage Modes`, which you see when you execute `/help`:



```
Claude Code v1.0.61

Always review Claude's responses, especially when running code. Claude has read access to
files in the current directory and can run commands and edit files with your permission.

Usage Modes:
• REPL: claude (interactive session)
• Non-interactive: claude -p "question"
```

Figure 10: *Claude Code Usage Modes*

This illustrates the various usage modes of Claude Code. Why I bring this up is that in addition to the interactive shell, `REPL` mode in *Figure 11* above, you can use Claude Code almost as a chatbot:



```
niels@win11-dev:~/claude-test$ claude -p "What is Claude Code"
Claude Code is Anthropic's official CLI tool that provides an interactive command-lin
e interface for Claude. It helps with software engineering tasks like code analysis,
file editing, debugging, and project management. You can run commands, search codebas
es, edit files, and get AI assistance directly from your terminal.
niels@win11-dev:~/claude-test$
```

Figure 11: *Claude Code Chatbot Mode*

In *Figure 11*, you see that Claude Code is running in a non-interactive mode, where you can ask it questions and get answers without having to type commands. This is similar to how you would interact with a chatbot.

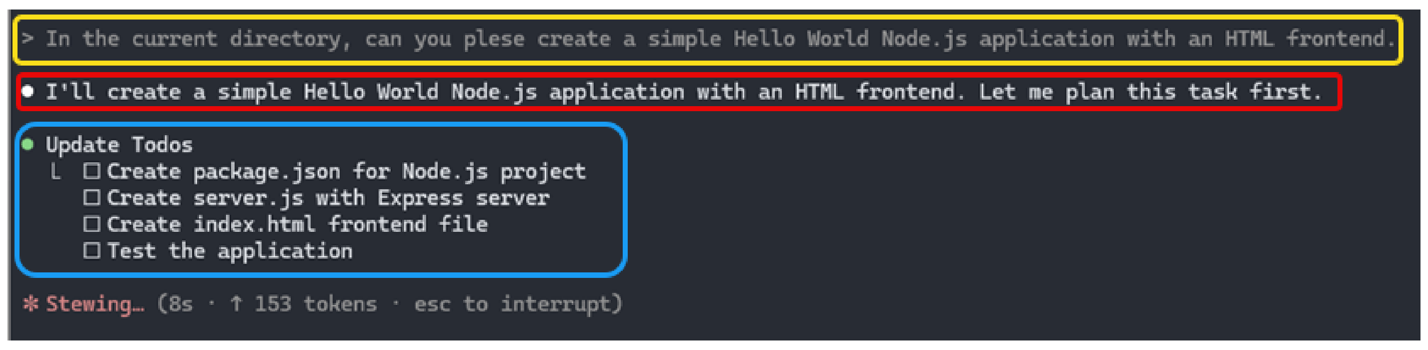
Write Code

Now, let's finish this post by asking Claude Code to write some code. We want Claude Code to create a simple "Hello World" application in `Node.js`. In the interactive shell, type the following command:

```
In the current directory, can you please create a simple Hello World Node.js application wi
```

Code Snippet 11: *Claude Code Command*

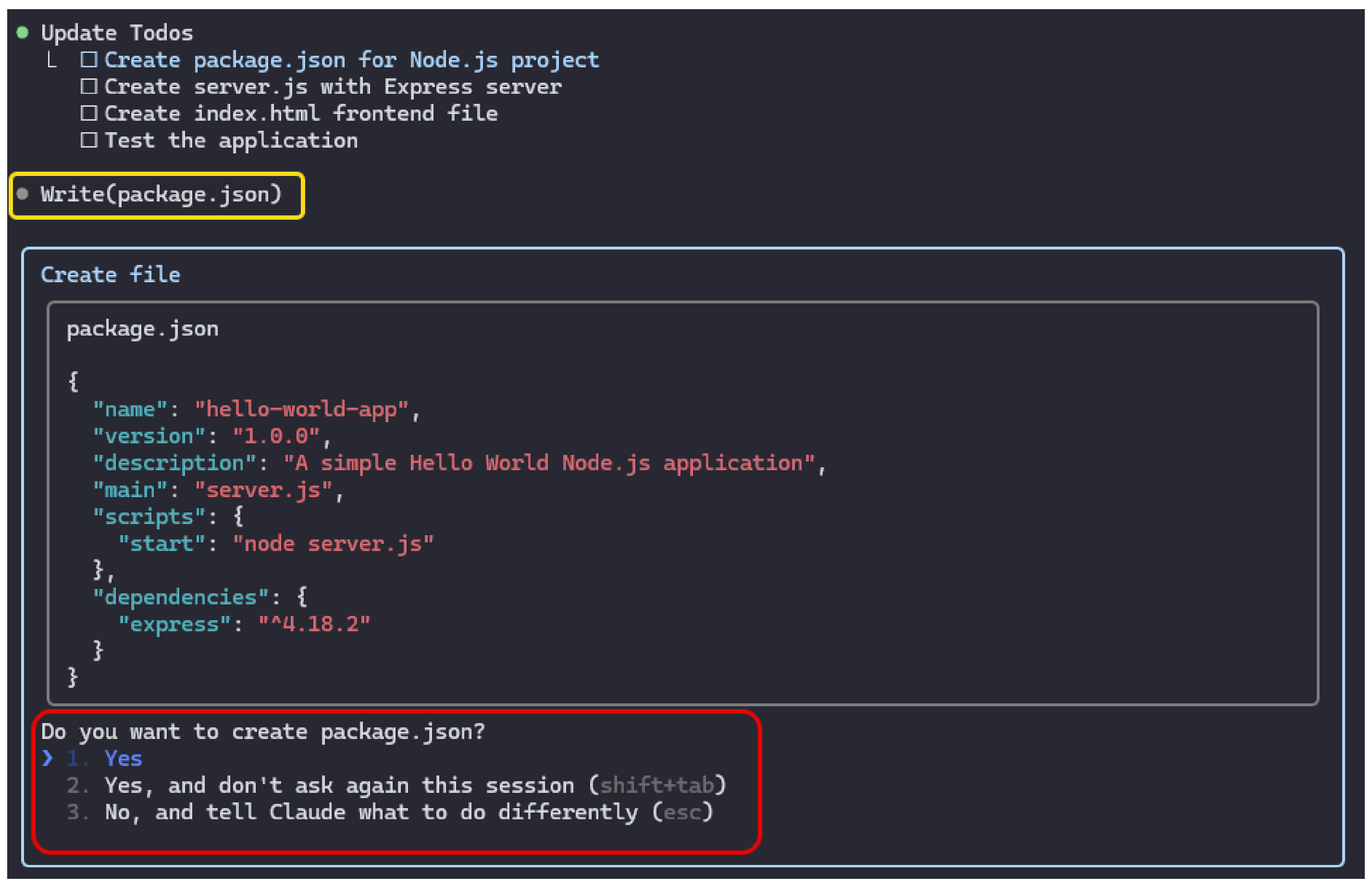
When you press `Enter`, Claude Code will start generating the code for you:



```
> In the current directory, can you plese create a simple Hello World Node.js application with an HTML frontend.

● I'll create a simple Hello World Node.js application with an HTML frontend. Let me plan this task first.

● Update Todos
  └ ☐ Create package.json for Node.js project
    ☐ Create server.js with Express server
    ☐ Create index.html frontend file
    ☐ Test the application

✻ Stewing… (8s · ↑ 153 tokens · esc to interrupt)
```

Figure 12: *Claude Code Generating Code*

As you see in *Figure 12*, based on the instructions I gave Claude Code (outlined in yellow), it starts planning the task (outlined in red). Part of the planning is creating and updating **Todos** (outlined in blue). After planning of the **Todos**, Claude Code starts implementing the **Todos**:

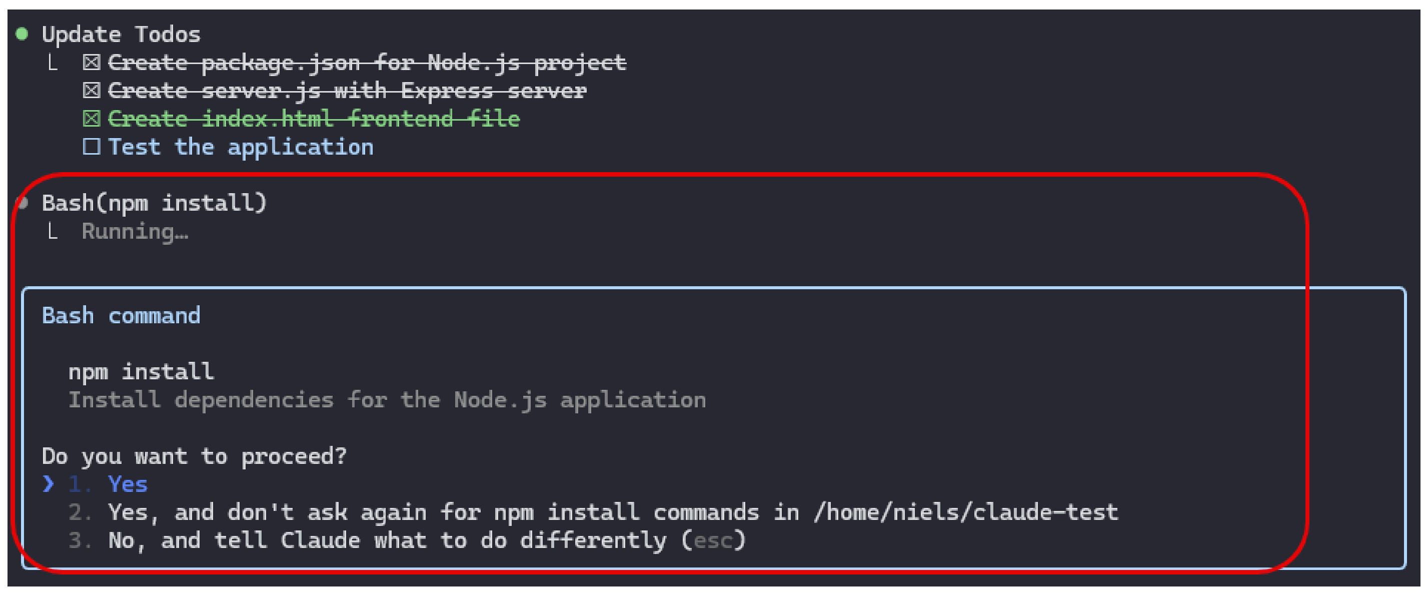


```
● Update Todos
  └ ☐ Create package.json for Node.js project
    ☐ Create server.js with Express server
    ☐ Create index.html frontend file
    ☐ Test the application

● Write(package.json)

 Create file

  package.json

  {
    "name": "hello-world-app",
    "version": "1.0.0",
    "description": "A simple Hello World Node.js application",
    "main": "server.js",
    "scripts": {
      "start": "node server.js"
    },
    "dependencies": {
      "express": "^4.18.2"
    }
  }

 Do you want to create package.json?
 ❯ 1. Yes
   2. Yes, and don't ask again this session (shift+tab)
   3. No, and tell Claude what to do differently (esc)
```

Figure 13: *Claude Code Implementing Todos*

As shown in *Figure 13*, Claude Code is implementing the **Todos** it created during the planning phase. It creates a new file called `package.json` as per the plan. Before it creates the file, it asks you to confirm that you want to create the file (outlined in red in *Figure 13*). Claude Code will, by default, do this for all files it creates. Pausing and asking for each file can be time-consuming, so you have the option to suppress this behaviour. You see the options outlined in red in *Figure 13*. I suggest you do not suppress this behaviour until you are comfortable with Claude Code and how it works.

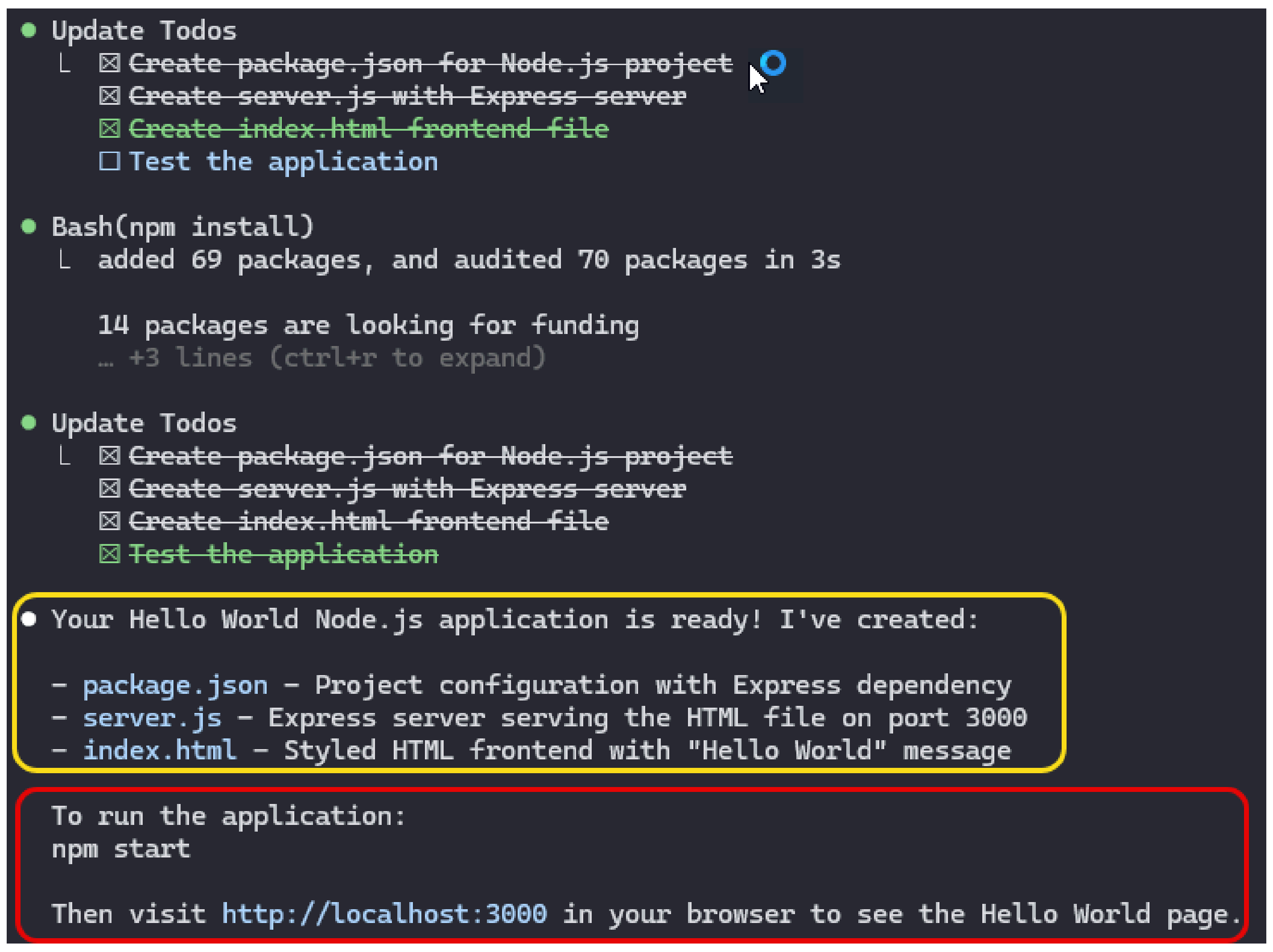
Eventually, after having created the required files, Claude Code reaches the **Test the application** stage. Here, it understands that `npm` is needed to run the application, and asks you if you want to install `npm`:



```
● Update Todos
  └ ☒ ~~Create package.json for Node.js project~~
    ☒ ~~Create server.js with Express server~~
    ☒ ~~Create index.html frontend file~~
    ☐ Test the application

● Bash(npm install)
  └ Running…

 Bash command

   npm install
   Install dependencies for the Node.js application

 Do you want to proceed?
 ❯ 1. Yes
   2. Yes, and don't ask again for npm install commands in /home/niels/claude-test
   3. No, and tell Claude what to do differently (esc)
```

Figure 14: *Claude Code Asking to Install npm*

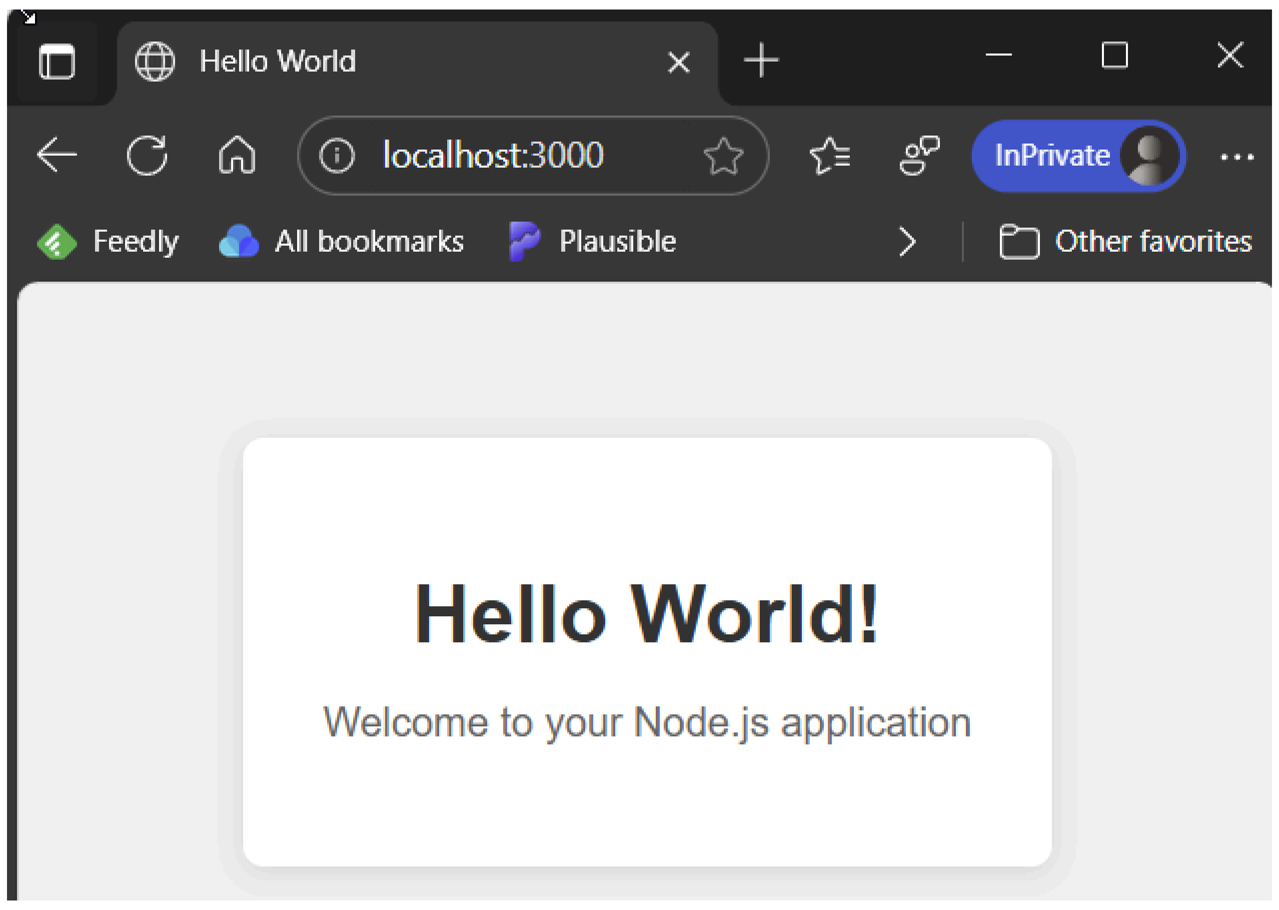
After you accept the installation of `npm`, Claude Code will install it for you and finish the code creation process:



```
● Update Todos
  └ ☒ ~~Create package.json for Node.js project~~
    ☒ ~~Create server.js with Express server~~
    ☒ ~~Create index.html frontend file~~
    ☐ Test the application

● Bash(npm install)
  └ added 69 packages, and audited 70 packages in 3s

    14 packages are looking for funding
    … +3 lines (ctrl+r to expand)

● Update Todos
  └ ☒ ~~Create package.json for Node.js project~~
    ☒ ~~Create server.js with Express server~~
    ☒ ~~Create index.html frontend file~~
    ☒ ~~Test the application~~

● Your Hello World Node.js application is ready! I've created:

  - package.json - Project configuration with Express dependency
  - server.js - Express server serving the HTML file on port 3000
  - index.html - Styled HTML frontend with "Hello World" message

  To run the application:
  npm start

  Then visit http://localhost:3000 in your browser to see the Hello World page.
```

Figure 15: *Claude Code Finished Creating Application*

As per *Figure 15*, you can now run the application by typing: `npm start` in the terminal, and then browsing to `http://localhost:3000`:



Hello World!

Welcome to your Node.js application

Figure 16: *Running Hello World Application*

Congratulations! You have successfully created a simple “Hello World” application using Claude Code.

What’s Next?

Now that you have installed and initialised Claude Code, you can theoretically start using it to build your event management system. However, as a developer, you are likely accustomed to developing in an IDE or code editor.

You do not have to develop in the Claude Code shell, but you can hook it up to your favourite IDE. In the next post, we will explore how to set up your development environment to work seamlessly with Claude Code and your preferred IDE.

Summary

This inaugural post marks the beginning of a comprehensive series that documents the journey of building a custom contact and event management system using Claude Code, Anthropic’s AI-assisted command-line development tool. We outlined the motivation for moving away from Brevo’s generic event management platform to create a tailored solution that better serves the community’s specific needs. The post establishes clear project goals, including core features like event creation, contact management, and automated communications, alongside technical objectives focused on modern architecture, maintainable code, and robust testing practices.

The post provides detailed installation instructions for Claude Code on both Mac and Windows systems. Through step-by-step guidance covering everything from initial installation to authentication and first usage, you learn how to configure Claude Code’s interactive shell and experience your first AI-assisted development task by creating a simple Node.js “Hello World” application. This hands-on demonstration showcases Claude Code’s planning capabilities, file creation process, and the collaborative nature of AI-assisted development, setting the foundation for the more complex event management system that will be built throughout the series.

Stay tuned for more updates as we continue to build this system and share our experiences along the way.

Finally

That’s all for now. I hope you find this information valuable. Please share your thoughts and ideas on this post, or ping (mailto:niels.it.berglund@gmail.com) me if you have any suggestions. Your input is highly valued and can help shape the direction of our discussions. If you found this post helpful, please consider sharing it with your network. Follow me on LinkedIn (https://www.linkedin.com/in/nielsberglund/) for more updates on this project and other AI-related topics.

← PREVIOUS POST (/POST/2025-07-27-INTERESTING-STUFF---WEEK-30-2025/)

NEXT POST → (/POST/2025-08-03-INTERESTING-STUFF---WEEK-31-2025/)

Disqus comments not available by default when the website is previewed locally.
comments powered by Disqus (https://disqus.com)