CreateAccess

Build a 3D Geospatial Web App with CesiumJS

Overview

A 3D geospatial web app is a website that visualizes location-based data on a 3D globe. 3D geospatial technology helps represent these data dynamically and interactively. CesiumJS is a free, open-source JavaScript library that provides developers a complete 3D globe in a web browser. In the coding world, a library is a reusable collection of pre-written code that developers can plug into their own projects to add complex features quickly without building them from scratch. CesiumJS connects to Cesium ion, a cloud platform that provides satellite imagery, 3D terrain, and building data that make the globe look real. This helps developers tell stories or present information in geographic context, in ways that are not possible using two-dimensional maps.

This guide walks you through the complete process of building and deploying a 3D geospatial web app using CesiumJS to share information about data you care about in 3D. You will install the required software, generate a project blueprint using the CreateAccess CesiumJS Gem, build the application with Antigravity IDE, test it locally, and publish it to the web using Vercel. The CreateAccess CesiumJS Gem is a custom AI assistant with specific knowledge about CesiumJS and instructions for helping you develop your app.

No prior experience with geospatial tools is required to complete a project using this guide. By the end of this guide, you will have a working, publicly accessible application and an understanding of the process to create other applications in the future.

Phase 1 - Setup: Install Software and Create Accounts

Preview

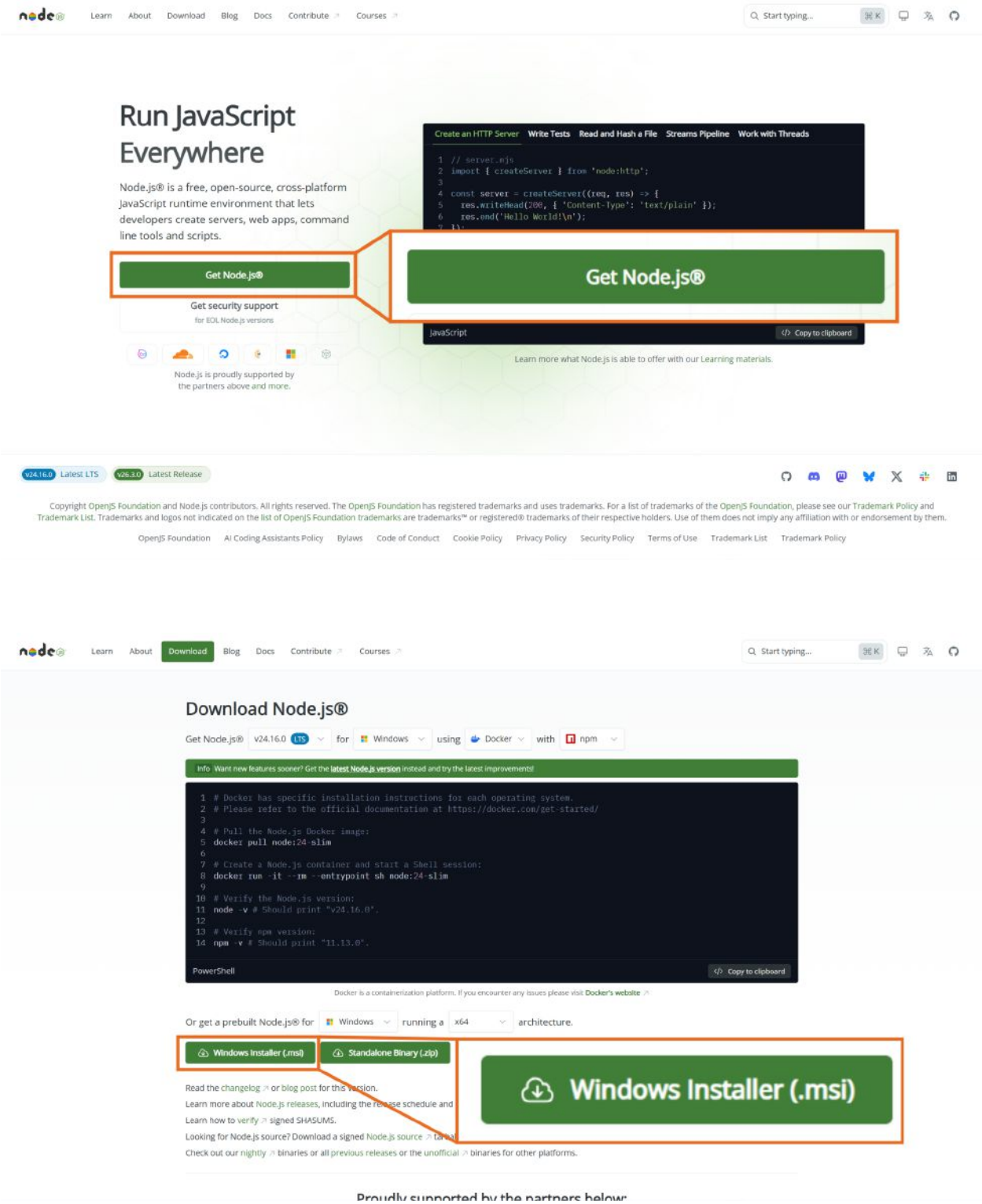
Before developing your app, you need the right tools installed and accounts created. This phase covers every install and sign-up in order. Plan for about 20–30 minutes the first time you're building a project. After the first time, you will not need to do it again.

Experiment

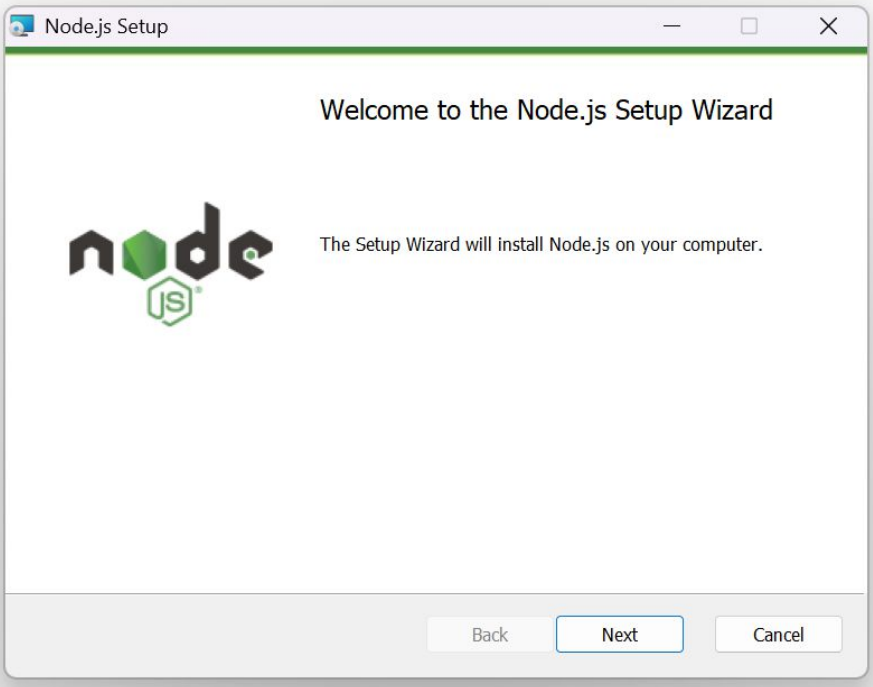
INSTALL NODE.JS (V20 LTS)

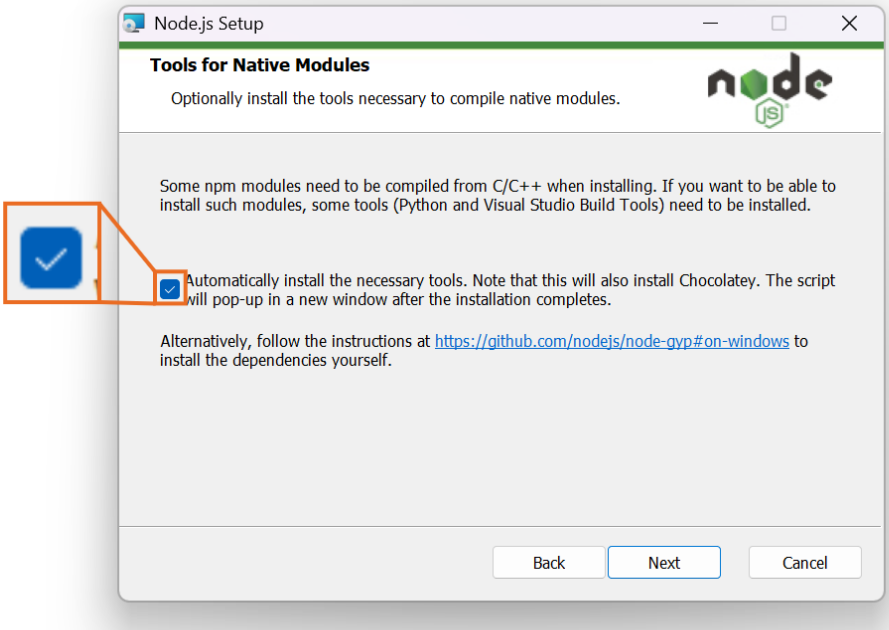
Node.js is the engine that runs your local development server, build tool (Vite), and package manager (npm). In a CesiumJS project, npm downloads the CesiumJS library and its code dependencies into your project, and Vite bundles those massive 3D geospatial files efficiently so they load instantly on your screen. CesiumJS and Antigravity both require Node v20 or newer.

1. Go to nodejs.org and download the **prebuilt version** for Windows or Mac depending on your hardware.

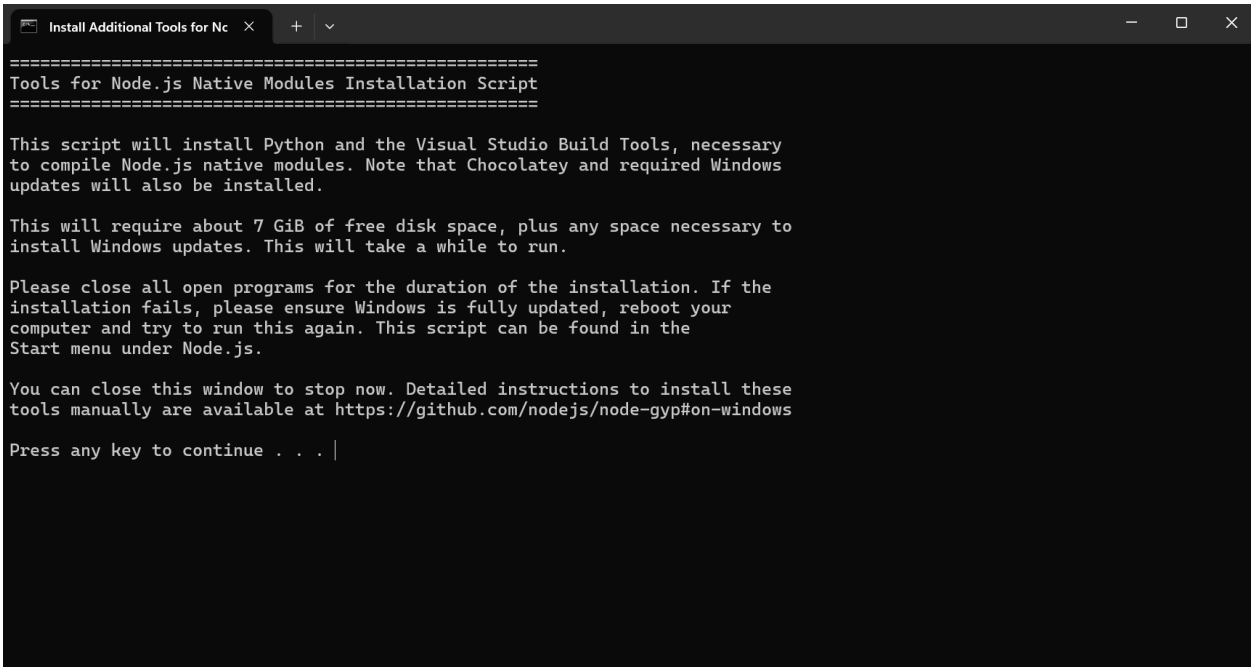


2. Run the installer, check the box to install necessary tools, and click “Next” to proceed.



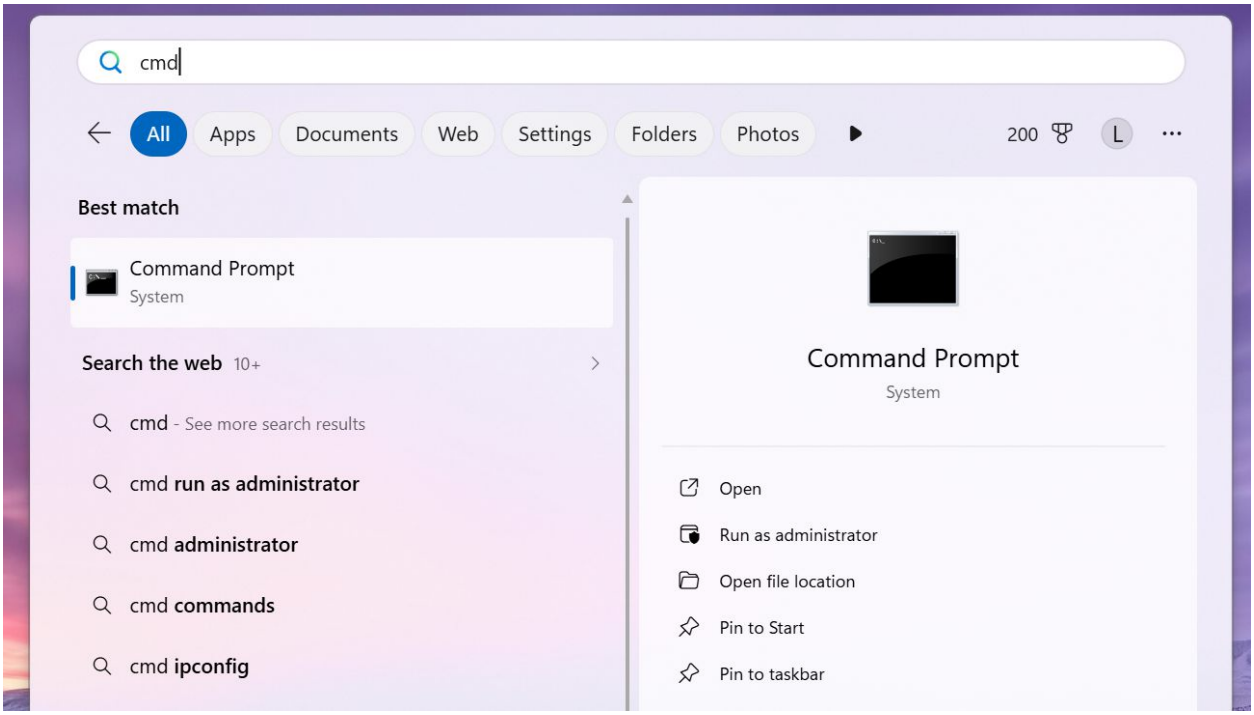


If a new window for installation appears, follow the prompts on screen to install all tools necessary.



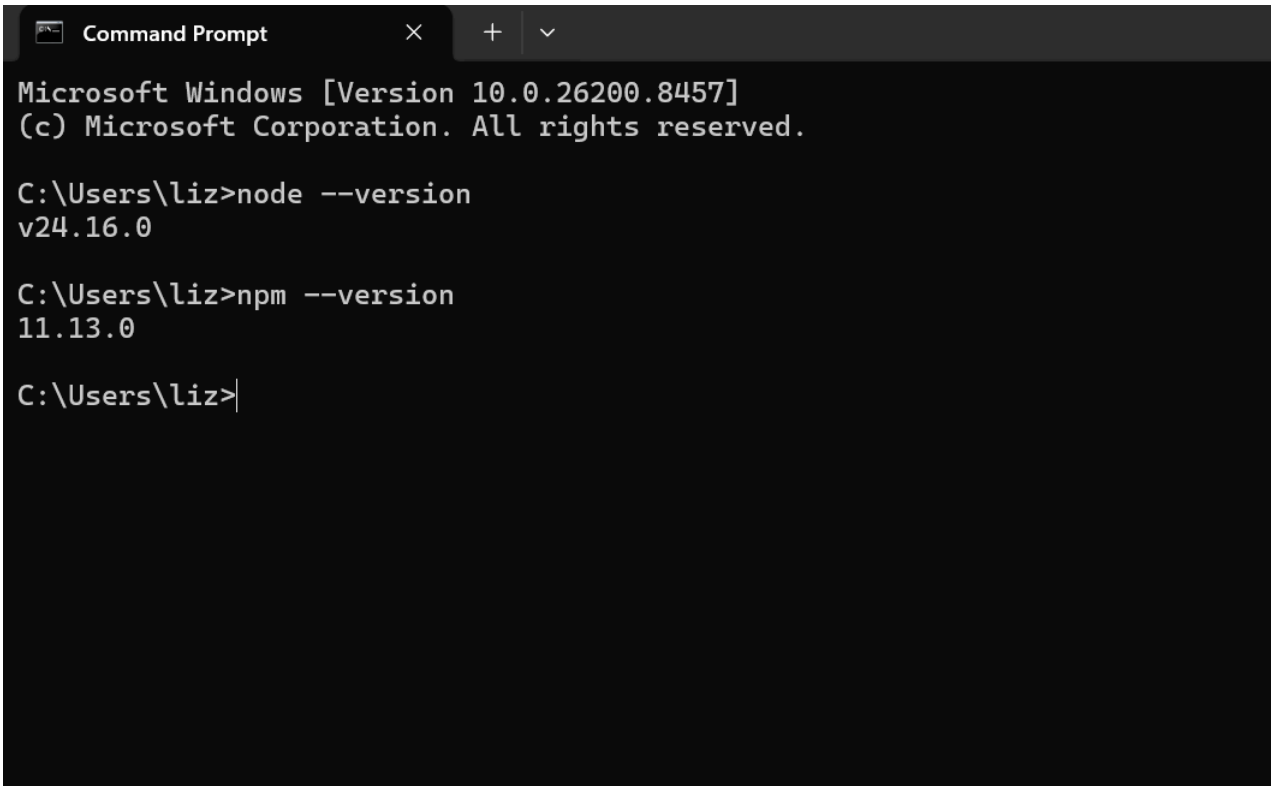
3. To check if the tools have successfully been downloaded to your computer, you can open a “terminal” and verify the install by typing a brief code. Follow the instructions below based on your operating system to open a terminal.

   Mac: press Command + Space, type Terminal, and press Enter.
   Windows: press the Windows key, type cmd, and press Enter.



4. In a new window, the terminal will open. Here, type in the code below 1 line at a time and press enter after entering each line to “run the commands”.

```
node --version
npm --version
```



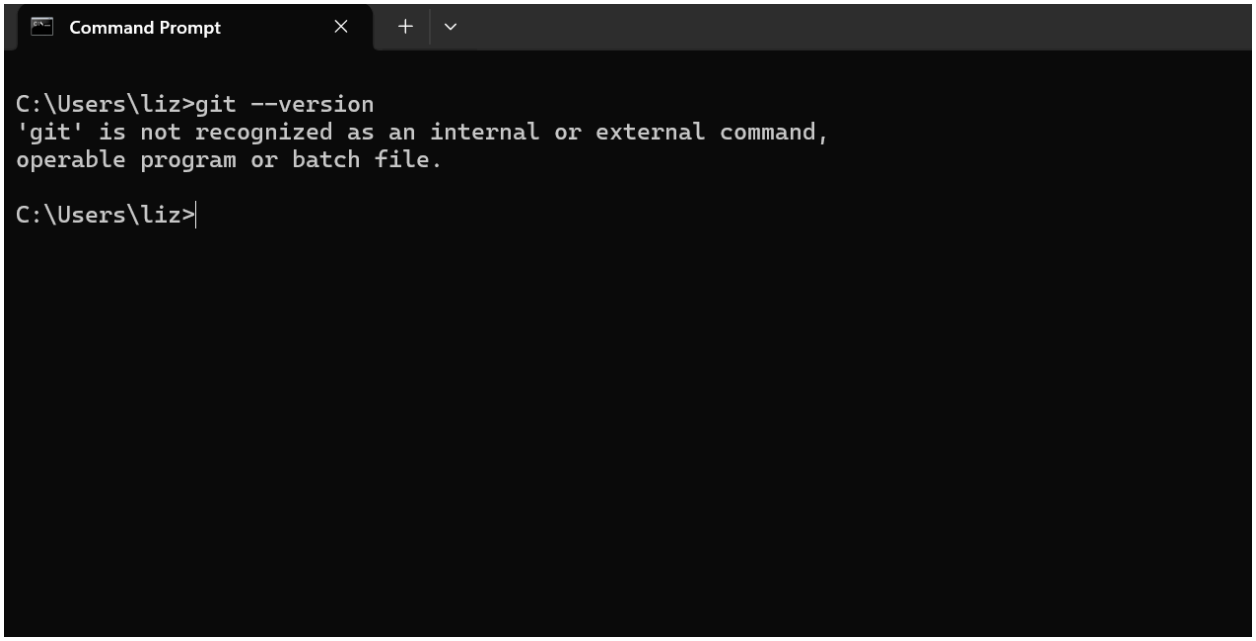
When you run the “node --version” command, you should get a response from the terminal that shows v20.x.x or higher. This is the version of “node.js” on your computer.

The “npm --version” command, you should get a response from the computer terminal that shows 10.x.x or higher. This is the version of “npm” on your computer.

INSTALL GIT

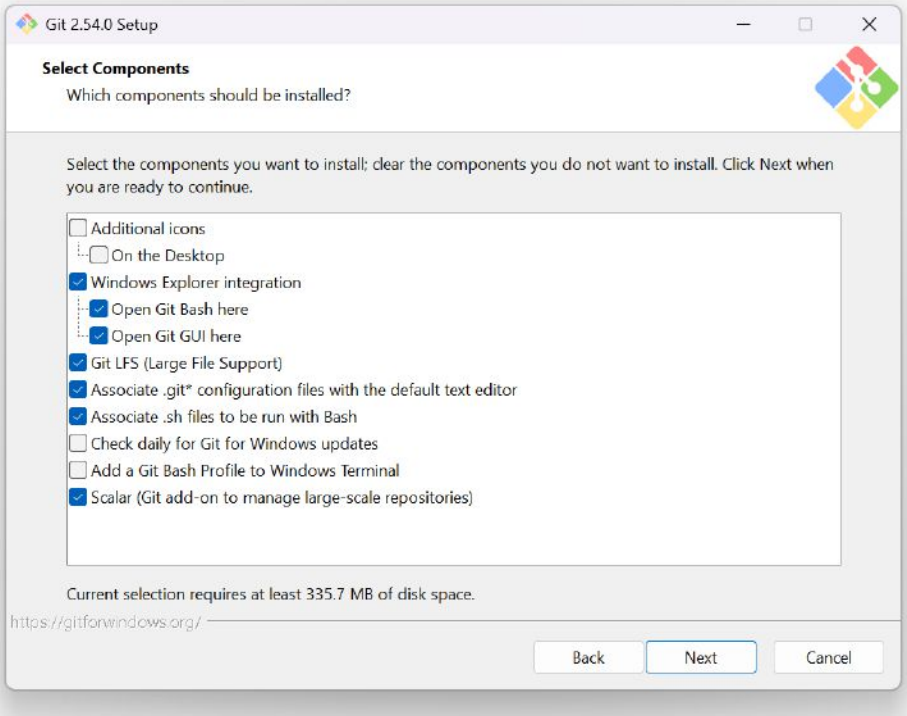
Git is a tool that runs on your computer to track changes in your code, acting like a time machine that lets you save snapshots of your progress and safely undo mistakes. GitHub is an online home where you upload those snapshots to the cloud, making it easy to store your project securely, collaborate with others, and automatically connect your code to web hosting services, like Vercel.

Open Command Prompt/Terminal again and type `git --version` and press enter.



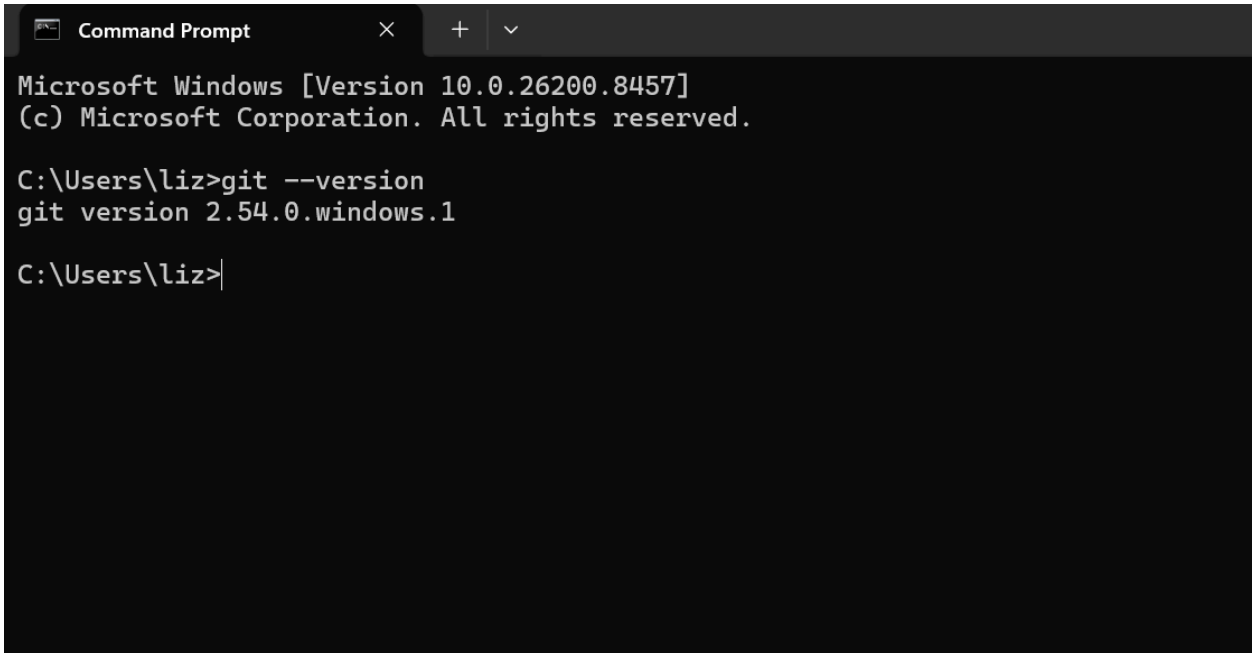
Mac: If Git is missing, macOS will prompt you to install it.

Windows: If Git is missing, download from git-scm.com/download/win and run the installer with all default options.



If the terminal is open from the previous step, close it. Then open a new terminal and verify by running this in your terminal:

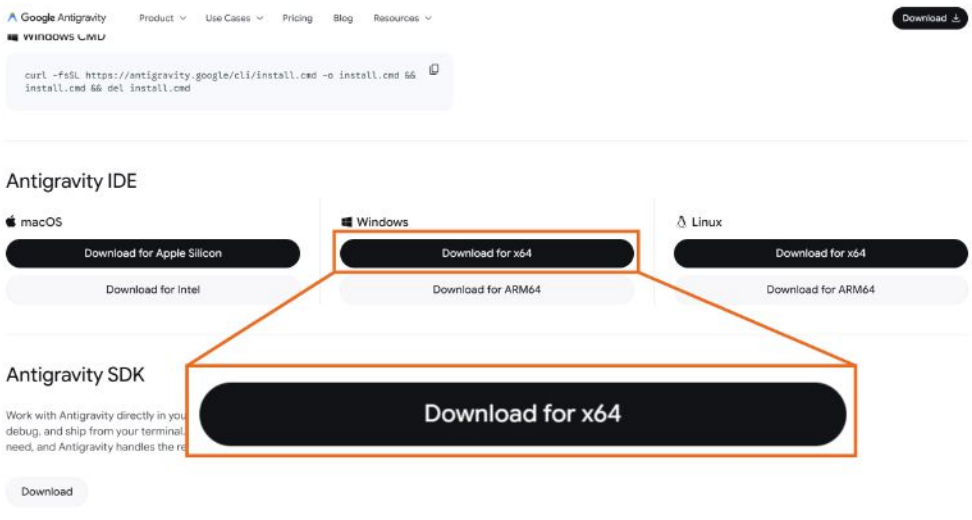
```
git --version    # Should show git version 2.x.x
```



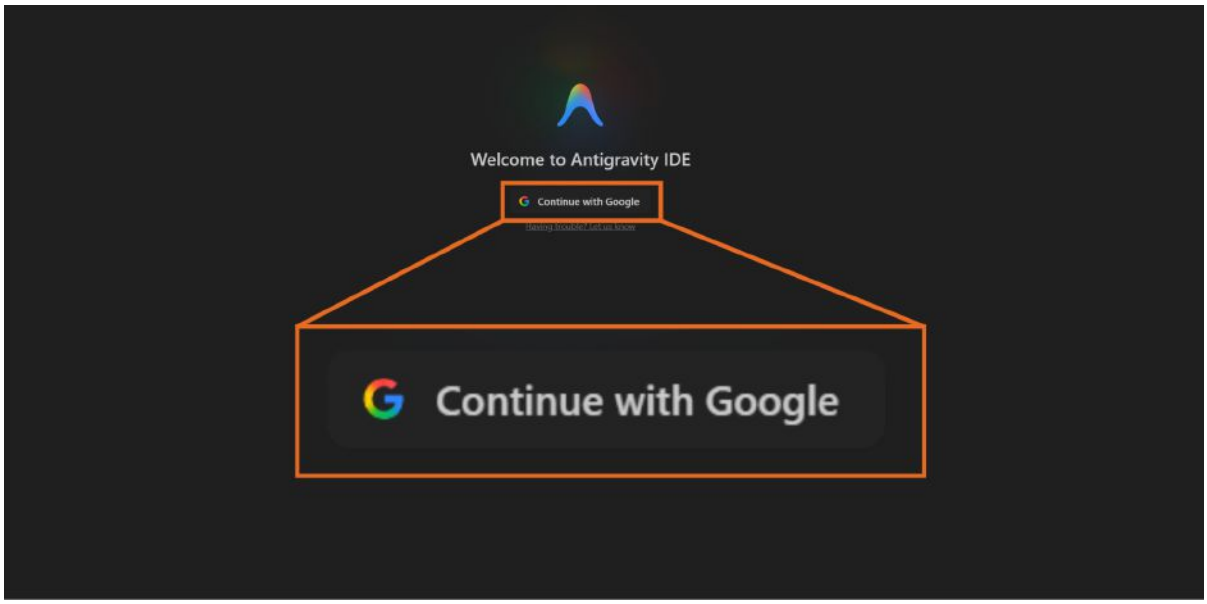
INSTALL ANTIGRAVITY IDE

Antigravity IDE is Google's AI-first development environment. An IDE is an Integrated Development Environment, a place where code can be written, compiled, and run. This is where an agent will write and help run your project code. ***Important: make sure you download "Antigravity IDE", NOT "Antigravity 2.0".***

5. Download from antigravity.google.com (macOS, Windows, Linux). Scroll down the page to find "Antigravity IDE".



6. Run the installer.
7. Sign in with a Gmail account and follow the default set up options.



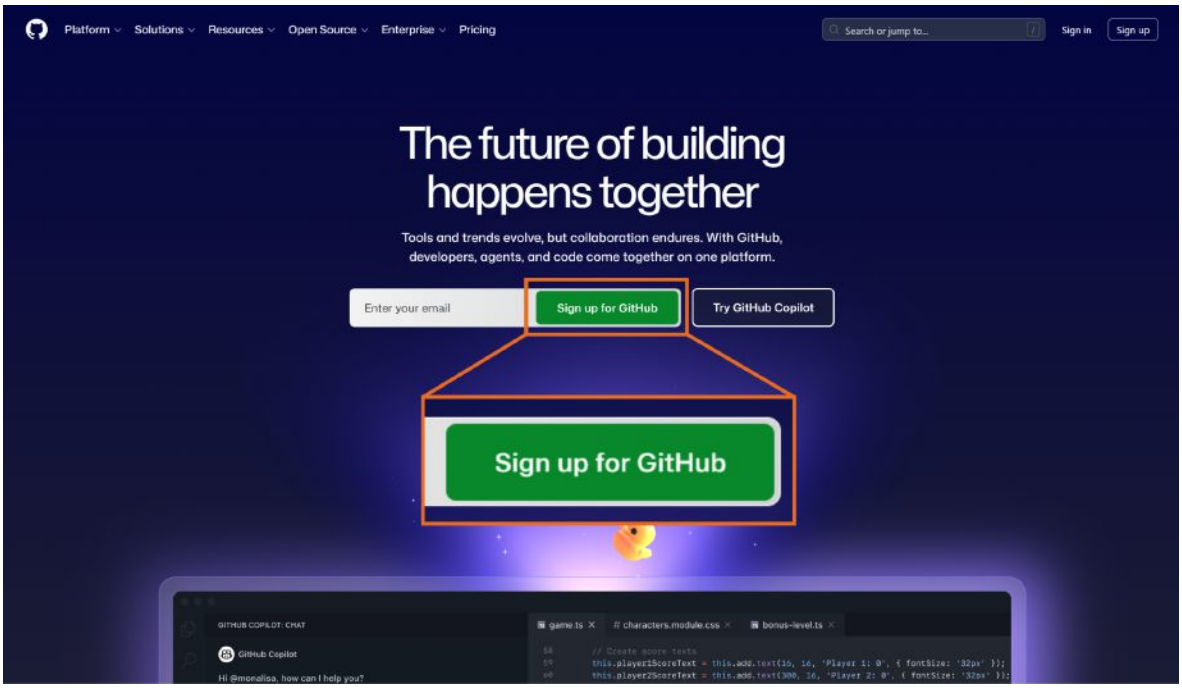
INSTALL CHROME

CesiumJS renders in the browser and Antigravity's built-in preview works best with Chrome. Download from google.com/chrome if needed.

CREATE A GITHUB ACCOUNT

GitHub stores your code and connects to Vercel so every push (or update) triggers an automatic re-deploy.

8. Go to github.com and click Sign up.
9. Create a free account. The free tier is all you need.

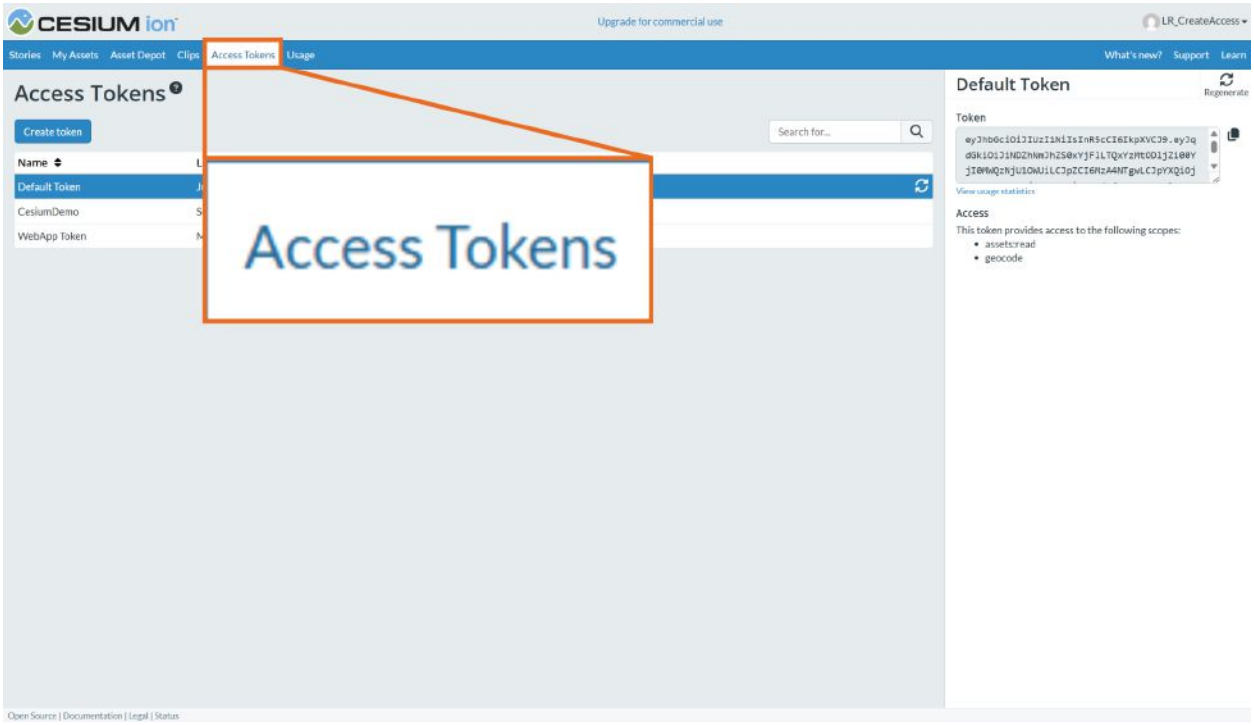


CREATE A CESIUM ION ACCOUNT

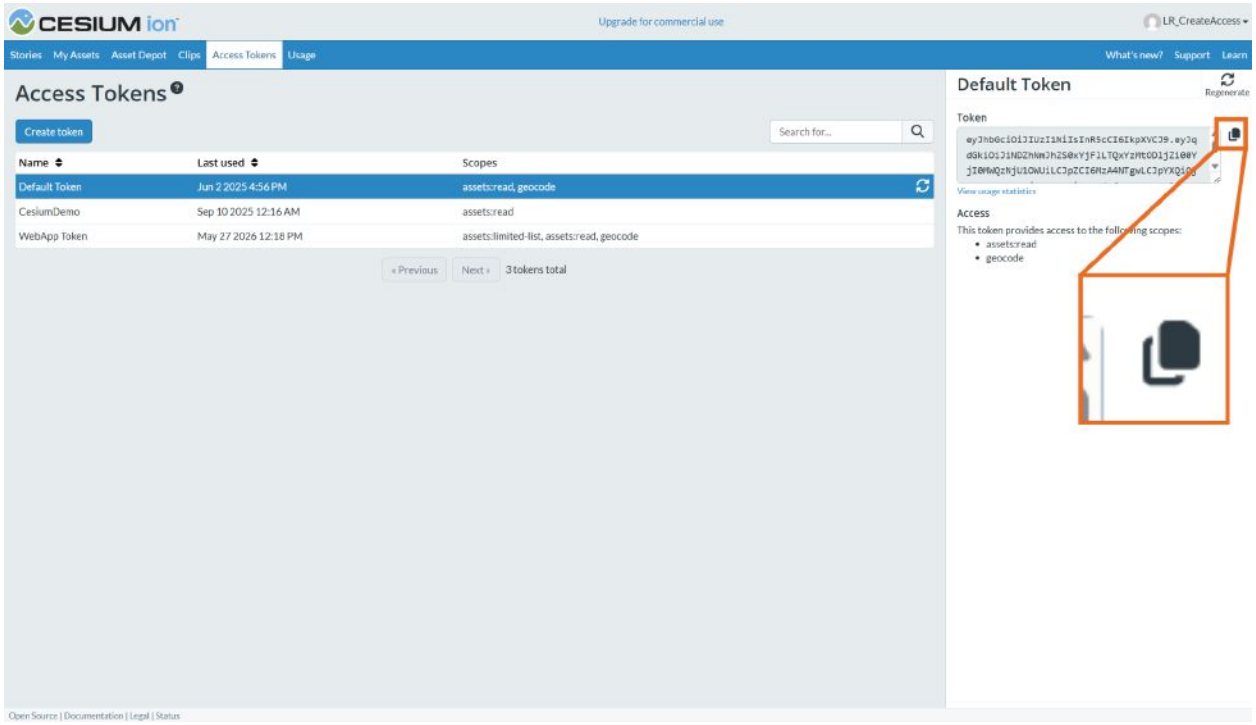
Cesium ion provides the satellite imagery, terrain, and 3D data that CesiumJS streams to render the Earth. You need a free API token to use this content in your web app. An “API” is an Application Programming Interface, which is like a “digital messenger” that allows two software applications to talk to each other and share data. An API token helps ensure that this data sharing happens securely.

10. Go to cesium.com/ion/signup and create a free account.

11. After signing in, open Access Tokens.



12. Copy your default token or have it easily accessible for later.

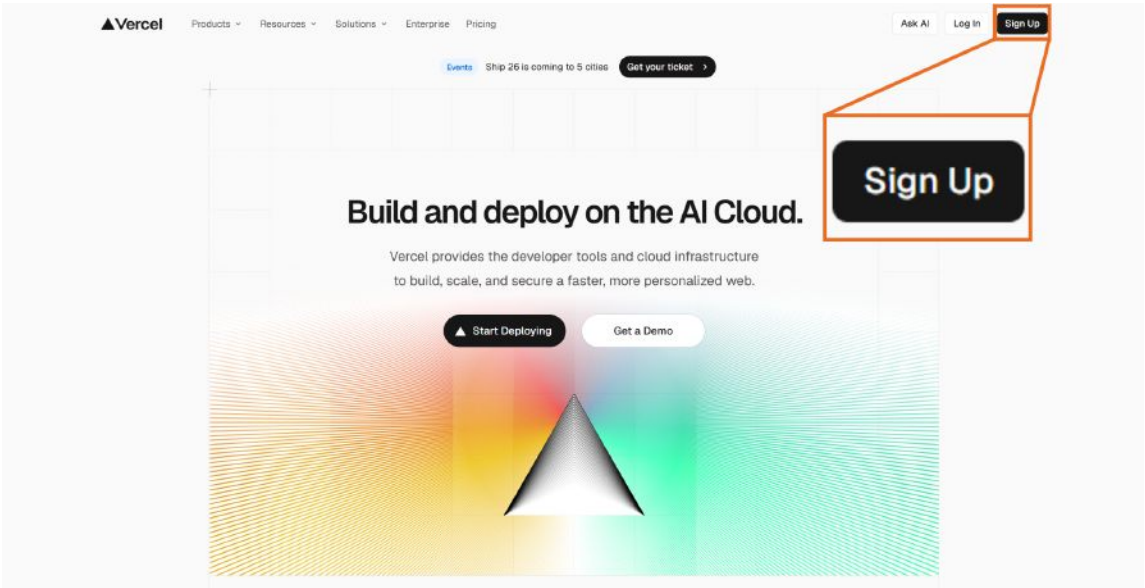


Keep your Cesium ion token private. Do not commit it to GitHub. You will store it in a local file that Git ignores (.env), so it is not publicly accessible on the web. This guide contains part of a team member's token but it was generated prior to release. If you ever are concerned that your token has been leaked or is no longer private, you can regenerate your token using the button in the upper-right corner.

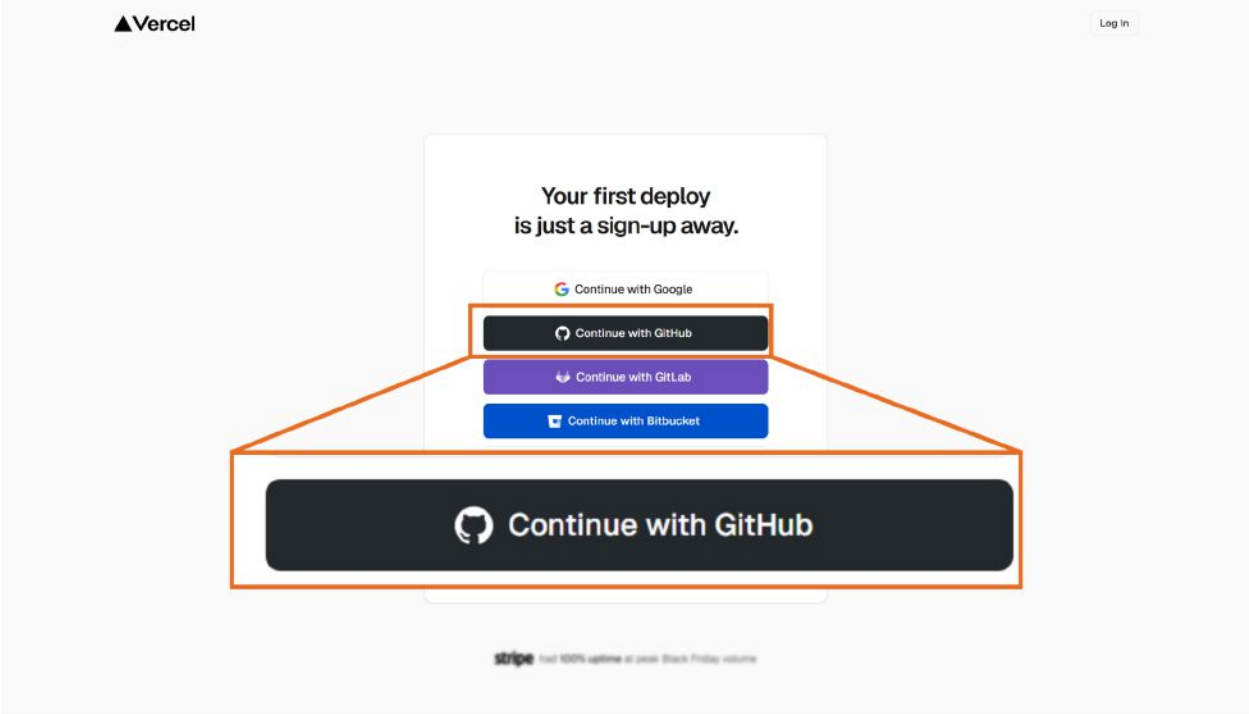
CREATE A VERCEL ACCOUNT

Vercel hosts your finished web app for free and re-deploys automatically whenever you push new code to GitHub.

13. Go to vercel.com and click Sign Up.



14. Choose “Continue with GitHub”. This links the two accounts.



15. Complete the onboarding prompts. You do not need to import a project yet.

Self Check

Could you complete all installs and account setups? Before continuing to Phase 2, confirm each item below:

- ☐ `node --version` in terminal shows v20.x.x or higher
- ☐ `npm --version` in terminal shows 10.x.x or higher
- ☐ `git --version` in terminal shows a version number
- ☐ Antigravity IDE installed and signed in with a Gmail account
- ☐ Chrome installed
- ☐ GitHub account created
- ☐ Cesium ion account created, access token copied somewhere handy
- ☐ Vercel account created and linked to GitHub

Phase 2 - Scaffold: Use the CreateAccess CesiumJS Gem

Preview

The CesiumJS Gem is a custom Gemini AI assistant that guides you through the process of scaffolding your web app. You can share your project idea and work through the design until it produces a spec.md file, a structured blueprint that describes your application precisely enough for an AI agent to build it.

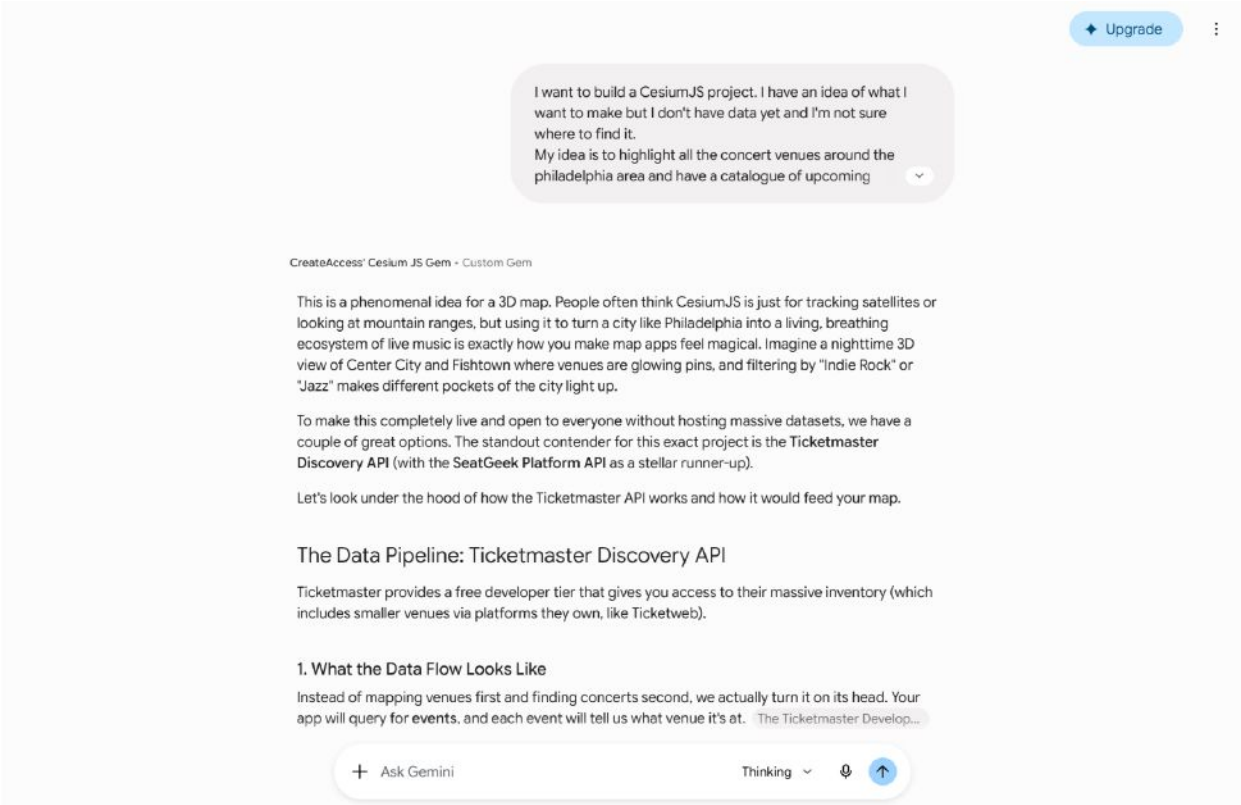
Experiment

OPEN THE GEM

The CesiumJS Gem is available at bit.ly/createaccess-gem.

16. Open the Gem in your browser.
17. Begin discussing your project with the Gem. Discuss your ideas in plain language. The Gem will ask for clarification when needed.

Below is an example of how a first interaction looks:



INTERACTING WITH THE GEM

Some important areas to consider when interacting with the Gem:

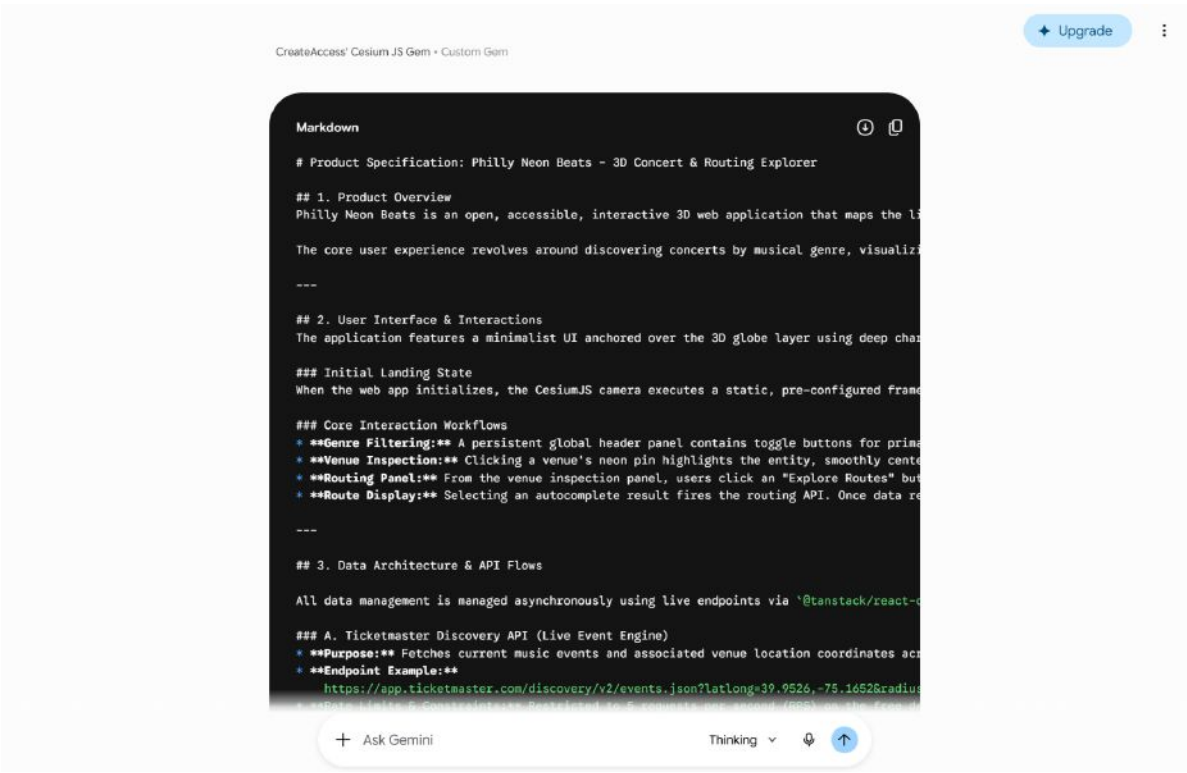
- **Project basics:** What is this map showing? Who is the audience?
- **Data:** What datasets will you load? Do you have them, or do you know where to get them? If you don't, the Gem can help you identify datasets that you may want to use in your project.
- **Geography:** Where does the map start? What is the initial camera view?
- **Visualization:** How should the data look? Colors, icons, pop-up info panels?
- **Interactivity:** Can users click on features, filter by date, or search a location?
- **User Interface:** Is there a sidebar, legend, or header beyond the globe itself?
- **Stack:** Vanilla JavaScript or React? Plain CSS or Tailwind? (The Gem will handle this but you can always ask questions about it, or make suggestions if you are familiar with web development).
- **Tileset:** Do you want realistic 3D terrain in the form of Google's Photorealistic Tiles or satellite 2D imagery with buildings blocked out using Open Street Map (OSM) buildings?

As you build your web app, you can use the Prompting Guidelines linked on our website to help with ideation, app building, and prompting the Gem.

GENERATE YOUR SPEC.MD

When the Gem has collected enough information, it produces spec.md content that includes:

- Project name and description
- Tech stack and dependencies (external code your app will need to function)
- File and folder structure
- Data sources with URLs or paths
- Visualization rules for each layer
- UI component descriptions
- Environment variables needed



This specification is written in Markdown, which is a simple formatting system that uses ordinary symbols (hashtags for headings and asterisks for bullet points) to create clean, organized documents that are easy for AI to parse. You will create the spec.md file in Phase 3 once your project folder is open in Antigravity, and copy-paste the Markdown content from the Gem.

Self Check

Were you able to complete the Gem conversation and receive a spec.md? The spec should describe your project clearly enough that someone else could build it from the document alone.

Phase 3 - Build: Hand Off to Antigravity IDE

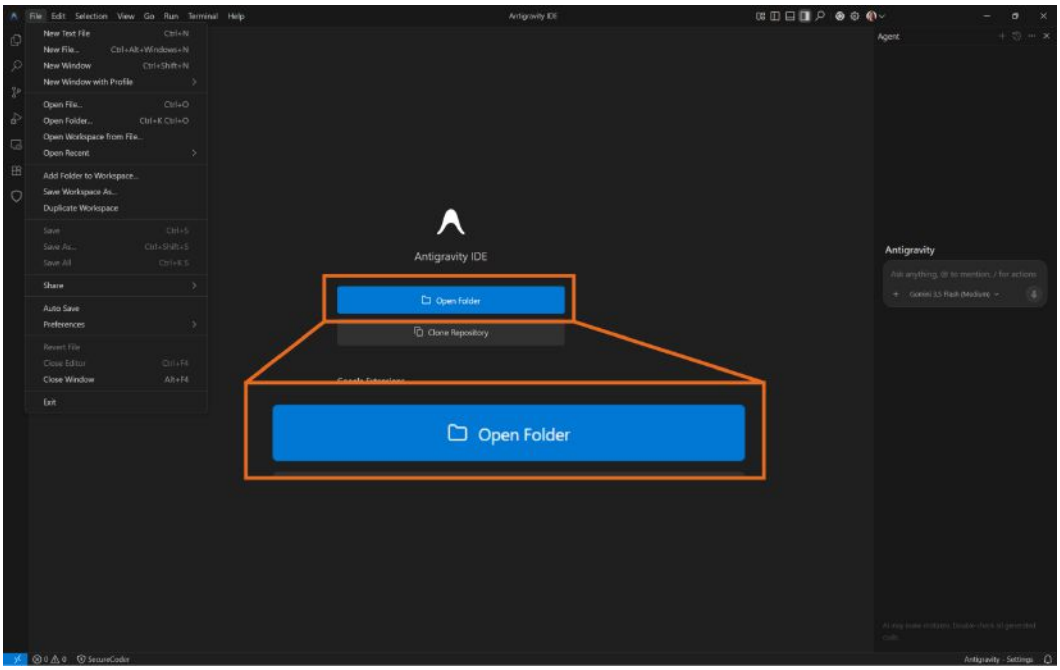
Preview

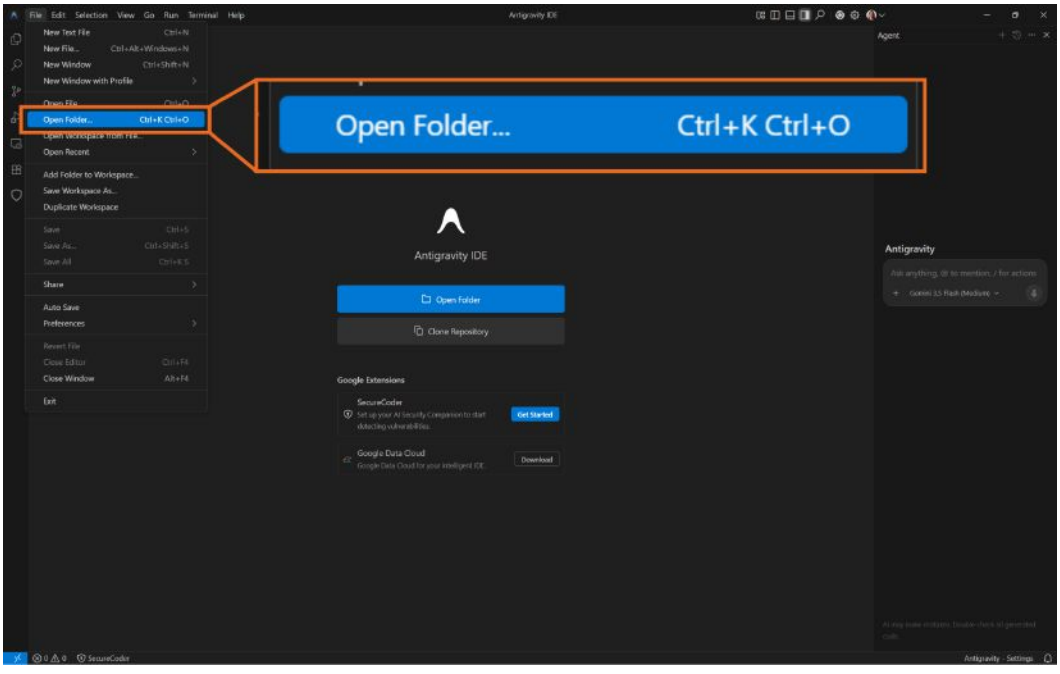
Antigravity is where the code gets written. You will open the Antigravity IDE, create a new project folder, paste in your spec.md, and let the AI agent scaffold the entire application. You will then work with the agent to refine anything that needs adjusting.

Experiment

OPEN A NEW PROJECT

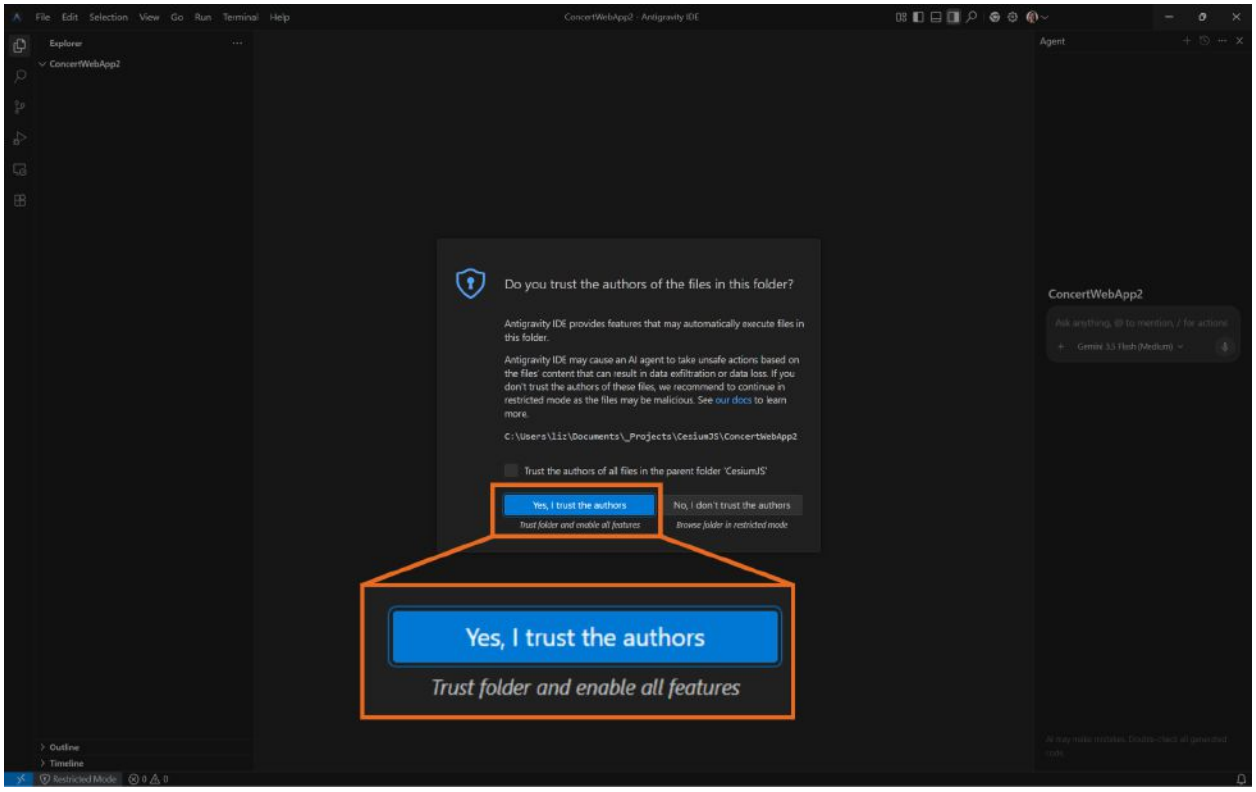
18. Open Antigravity IDE.
19. Click Open Folder (or File → Open Folder).





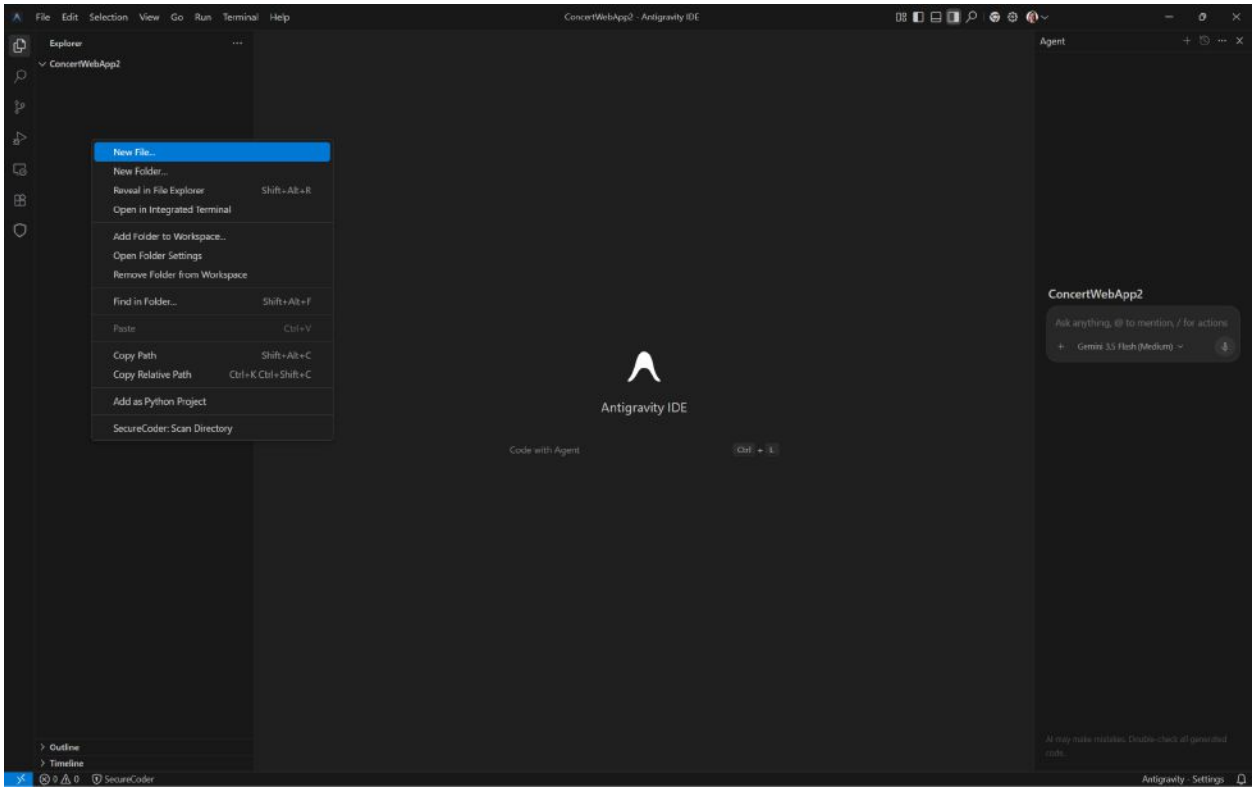
20. You can create a new folder or choose an empty one on your computer, and accept the “do you trust authors” question. An example for the project location could be:

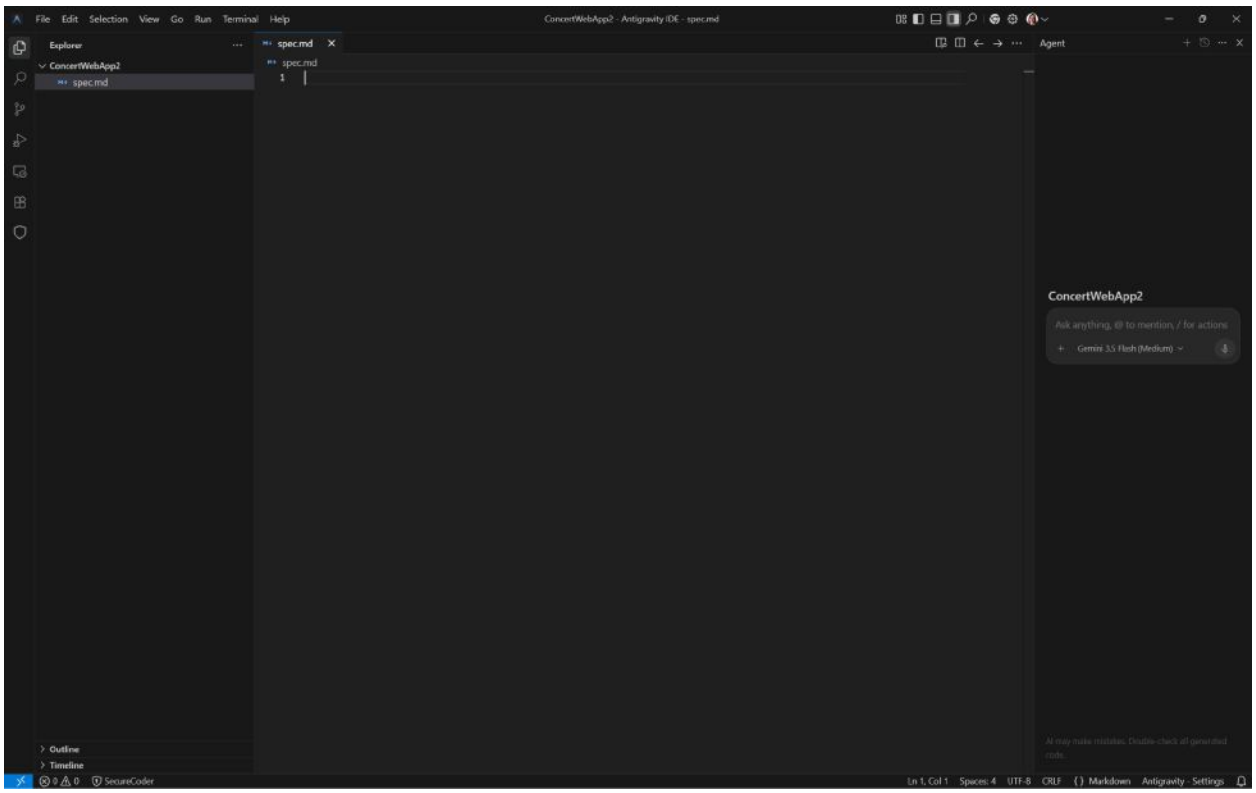
```
~/Documents/projects/my-cesium-map
```



CREATE YOUR SPEC.MD FILE

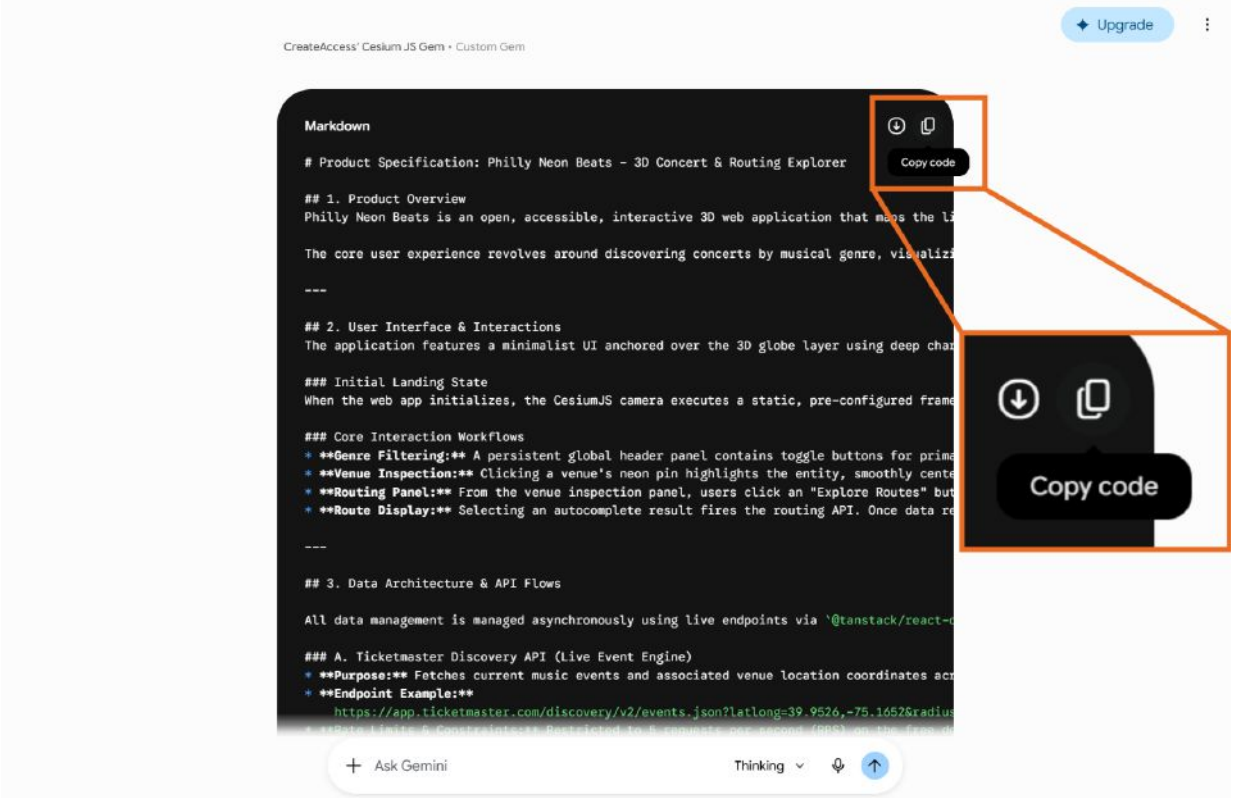
Now that your project folder is open, create the spec.md file the agent will read from. In the Antigravity file explorer (the panel on the left side), right-click your project folder and select New File or click the New File button next to the folder name. Name the file spec.md and press Enter.



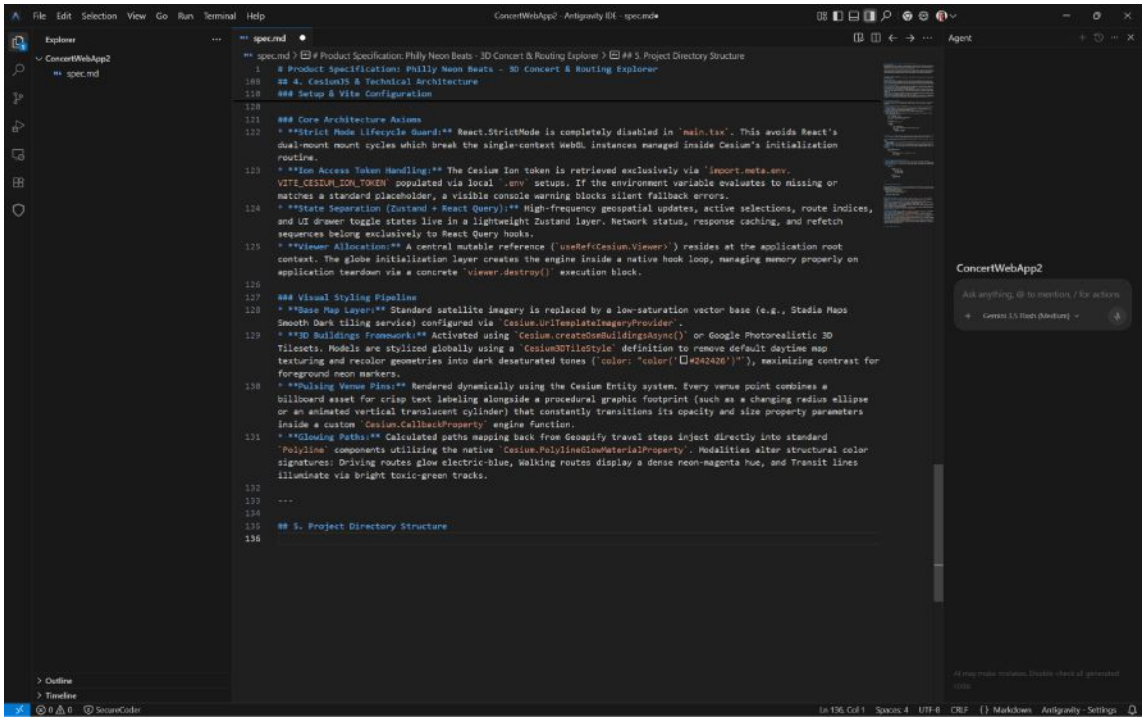


Go back to your Gemini tab where the spec is. Select all of the spec text using the copy button in the top right corner of the windowed content.

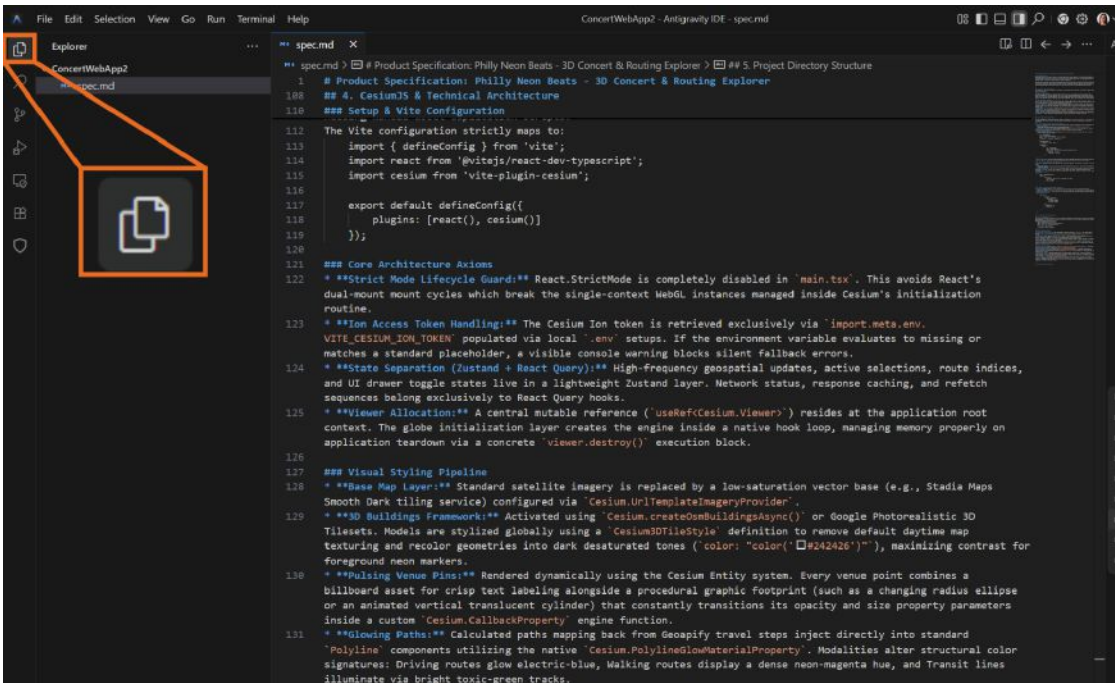
Sometimes the content does not all format into that block; if that happens, select the entirety of the spec manually by dragging and copying the content.



Then, click back into the spec.md file in Antigravity and paste (Ctrl+V or Command+V). You should see the full spec text appear in the editor.

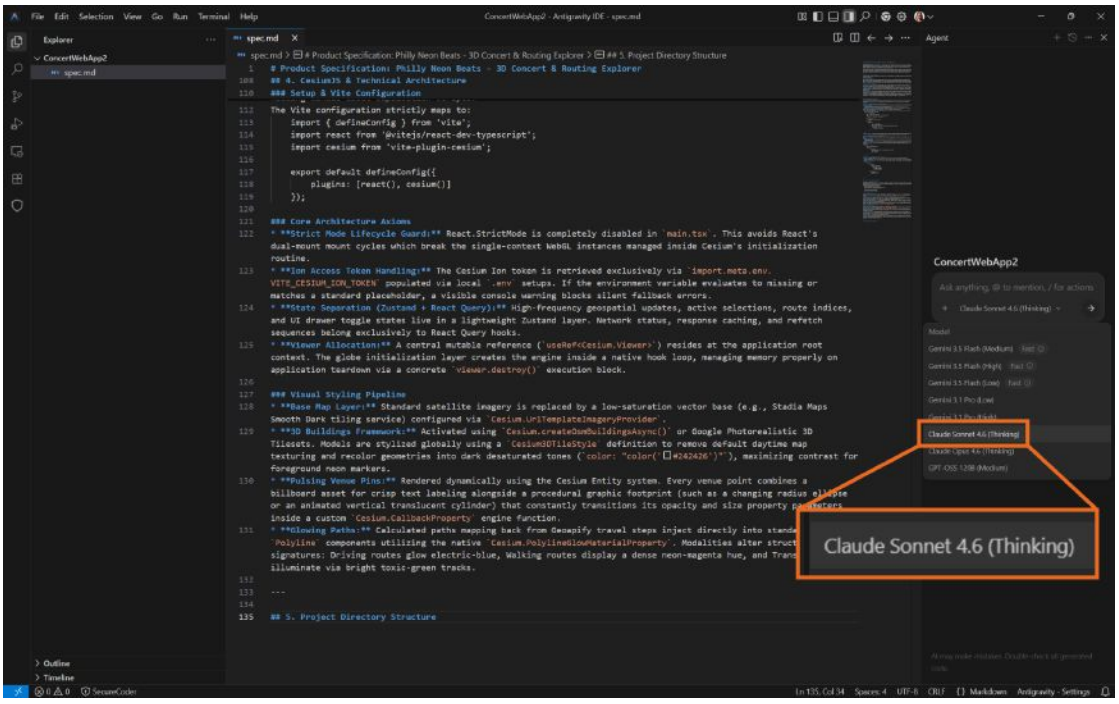


Save the file with Ctrl+S (Windows) or Command+S (Mac). The file is now in your project and the agent can reference it directly using @spec.md. Once you do this, you will see that the small ‘1’ that was once on the icon below has disappeared because there are no files with unsaved changes.



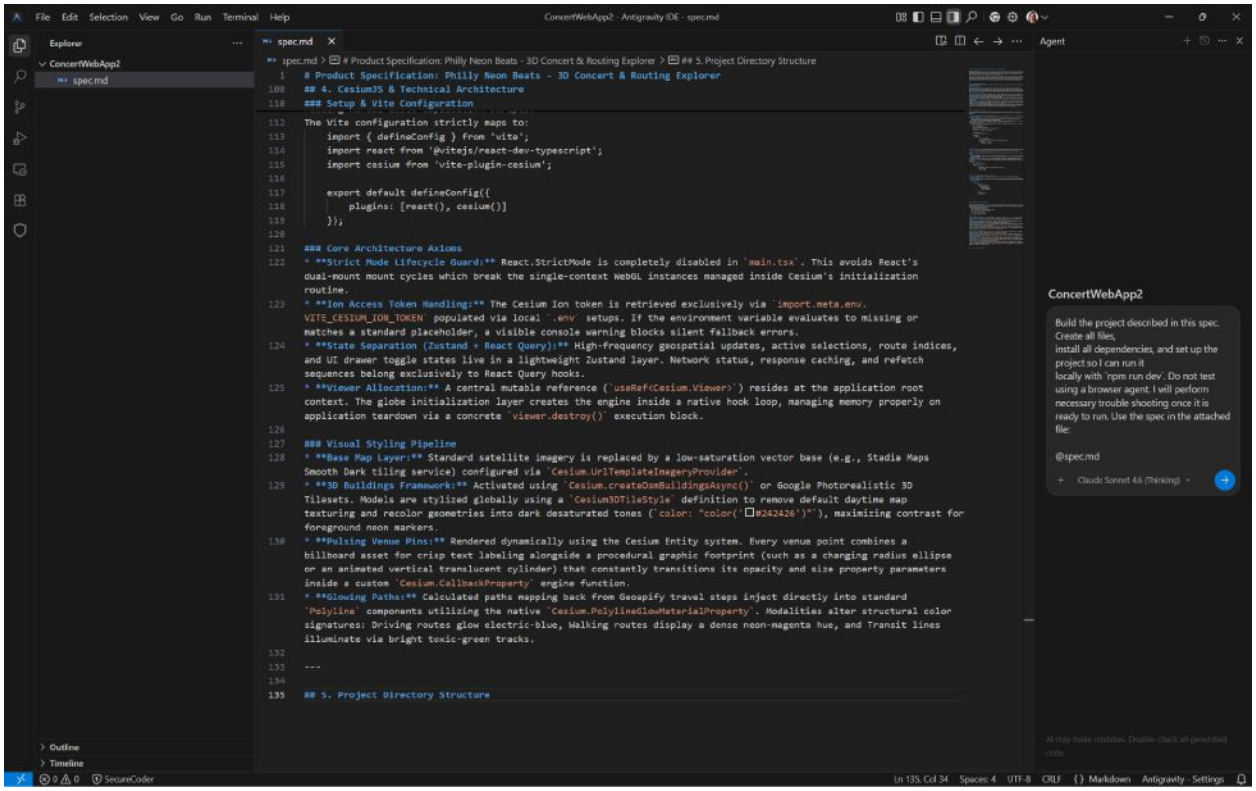
HAND OFF THE SPEC TO THE AGENT

21. The AI chat panel in Antigravity is open by default on the right side.
22. In the AI chat panel, you can switch between different AI models. We recommend using Claude Sonnet 4.6 until your usage runs out, and then switching over to Google’s Gemini models if need be. These are available in the same drop-down menu. Something to note: sometimes, the model’s usage may expire mid-work. Feel free to switch over to any of the other available models, provide brief context, and ask it to “continue working where the last agent left off.”



23. Type the following message. Use @spec.md to reference the file you just created so the agent reads it directly:

```
Build the project described in this spec. Create all files,
install all dependencies, and set up the project so I can run it
locally with `npm run dev`. Do not test using a browser agent. I will perform
necessary trouble shooting once it is ready to run. Use the spec in the attached
file:

@spec.md
```



Send the message and let the agent work. It may ask for permission to run commands. If you do not want to manually select “Allow” every time, you can edit the agent’s permissions in “Antigravity - Settings” in the bottom right corner of the IDE, and change “Auto Execution” to “Always Proceed.” Be aware that the agent will no longer ask you to run commands and will just run them for you.

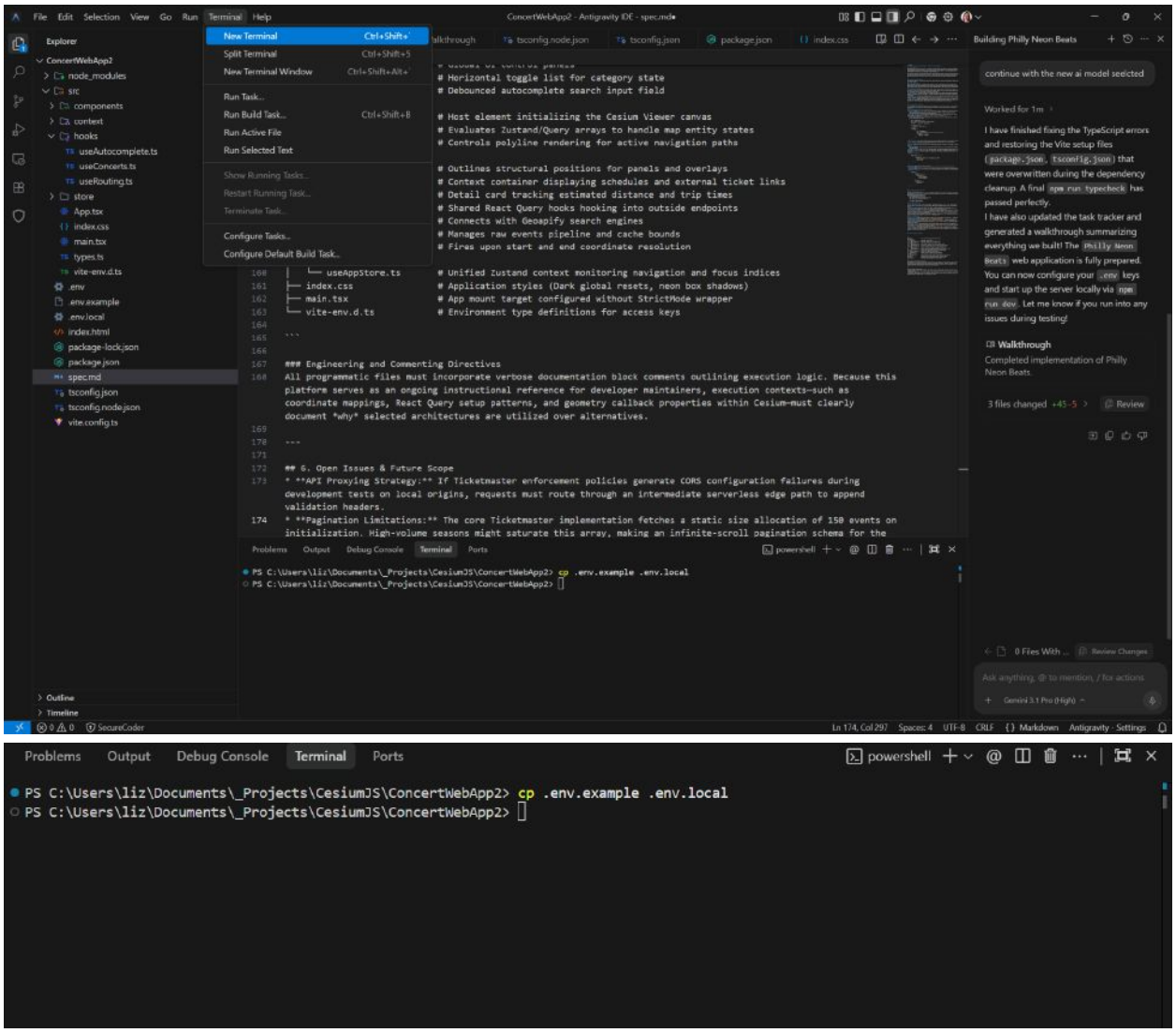
It will create the folder structure, generate all source files, configure Vite with the CesiumJS plugin, and create a .env.example listing the environment variables needed. Let the agent finish before typing again.

SET YOUR ENVIRONMENT VARIABLE

Your Cesium ion token is not in the code. It lives in a local file that Git ignores so it is not published to the web. The agent will have created a .env.example. You will copy it and fill it in with your Access Token from earlier.

24. In the Antigravity terminal, which you can open using: Ctrl+` or Cmd+’ (the backtick next to ‘1’ on the keyboard), run:

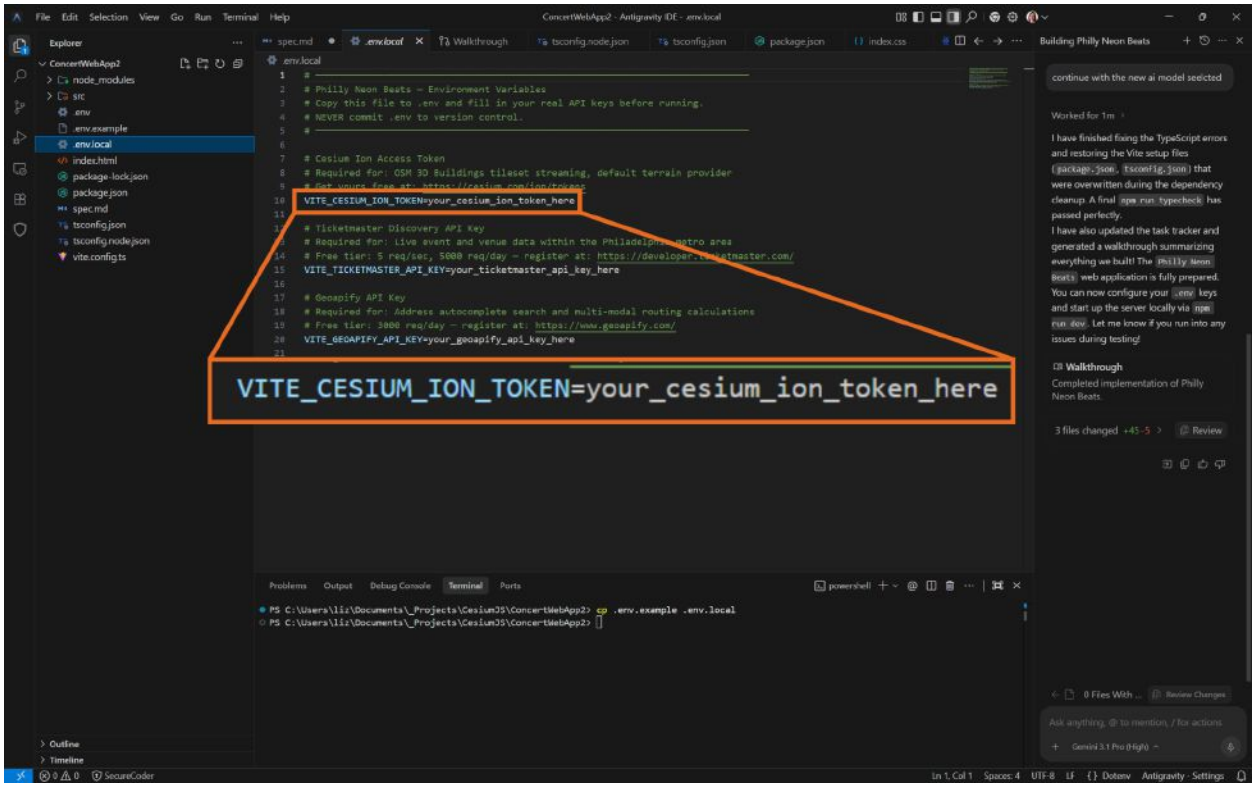
```
cp .env.example .env.local
```



From the “Explorer” panel on the left, open .env.local in the editor. It will look like this:

```
VITE_CESIUM_ION_TOKEN=your_token_here
```

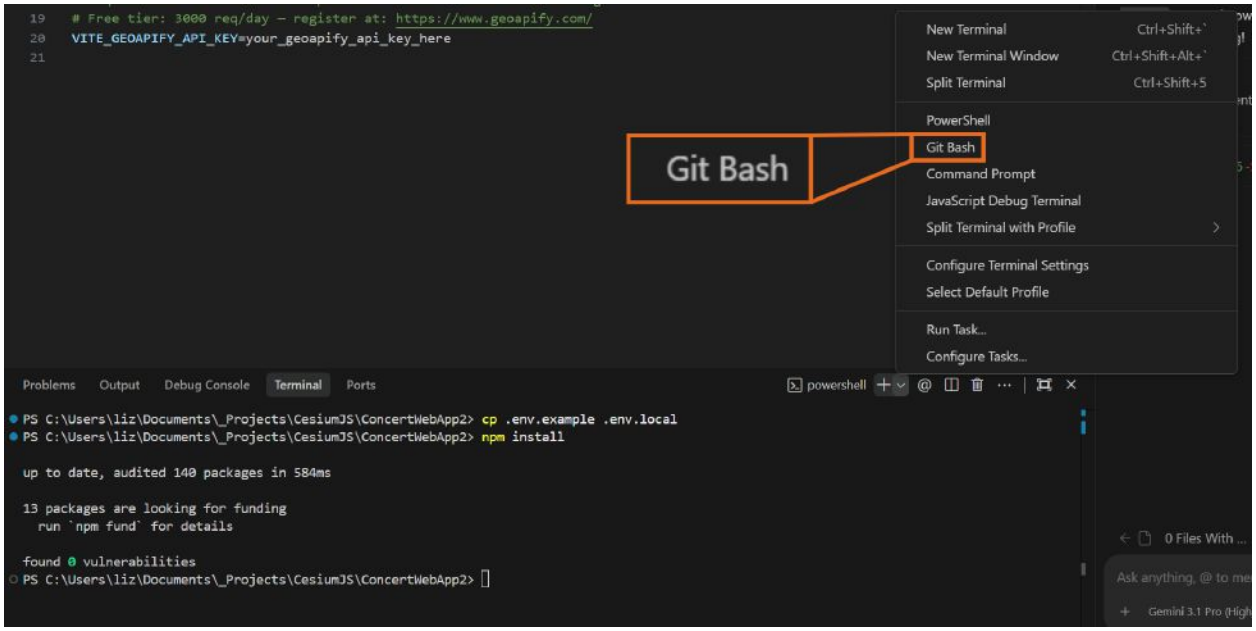
Replace your_token_here with the token you copied from Cesium ion in Phase 1 and save the file.



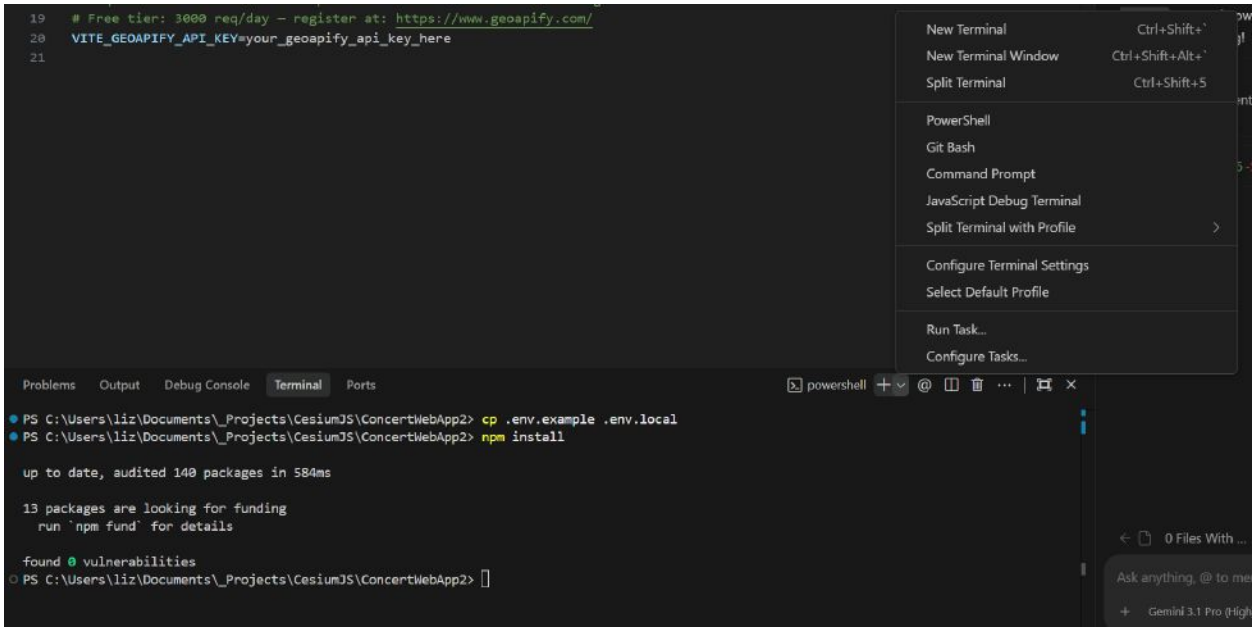
INSTALL DEPENDENCIES

Run this command in the Antigravity terminal to download all libraries the project needs, including CesiumJS, Vite, React, and others. If you run into errors the first time you run it, try switching your terminal to ‘Git Bash’ in the upper right corner of the terminal window.

```
npm install
```



This reads package.json and downloads everything into a node_modules folder. It will likely only take a few seconds, but may take longer—don't be alarmed by this!



Self Check

Could you open Antigravity, hand off the spec, and run npm install without errors? Check that your .env.local file exists and contains your Cesium ion token before continuing.

Phase 4 - Run Locally: Test Before Publishing

Preview

Before deploying, run the app on your own machine. This lets you catch issues in a fast feedback loop before they go live. You will also run a production build test, which catches a separate class of problems that only appear when Vite optimizes the code for deployment.

Experiment

START THE DEV SERVER

In the Antigravity terminal, run:

```
npm run dev
```

Vite is the build tool powering your project. It handles compiling your code and running a fast local development server so you can see your app in the browser without deploying it. Once you run the command above, it will start that server and you will see output like:

Open Chrome and go to http://localhost:5173. You should see your CesiumJS globe. If you want to close the server or need to run other commands, use **Ctrl+C** to terminate the command.

WHAT TO CHECK

Before moving to deployment, confirm the following work correctly:

- The globe loads and you can orbit, zoom, and pan with the mouse
- Satellite imagery appears. If the globe is blank, your Cesium ion token may be wrong or missing
- Your data layers load and display correctly
- Clicking on features shows the info panel or tooltip you designed
- No errors in the browser console. Open it with **F12** → Console tab

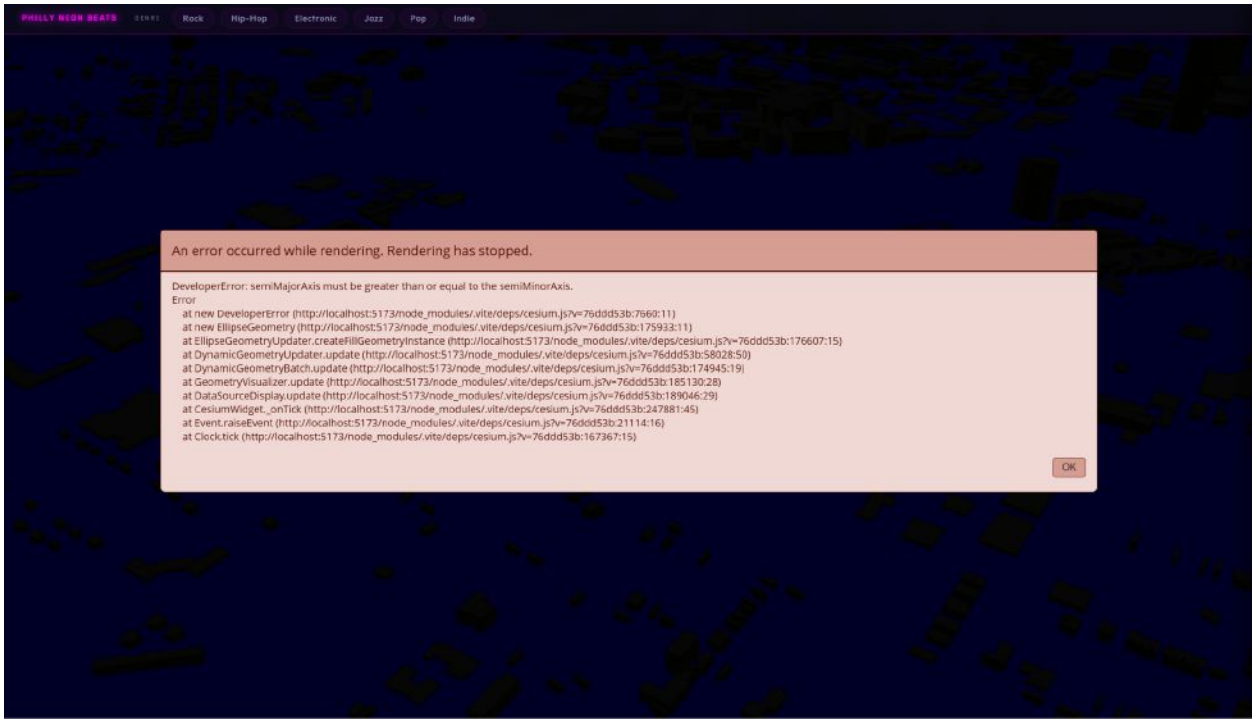
DESCRIBE PROBLEMS TO THE AGENT

If something does not look right, tell the Antigravity agent in plain language. For example:

```
The data points are not showing up. The console shows:
"Failed to load resource" for my GeoJSON file.
What is wrong and how do I fix it?
```

The agent can read the error, look at the relevant files, and make targeted fixes. Iterate until the app behaves as intended. You can copy error messages directly from the browser console (F12 → Console tab) by right-clicking the red error text and selecting Copy. Paste it into the agent chat for the most accurate diagnosis.

Below is an example error that occurred while building a web app:



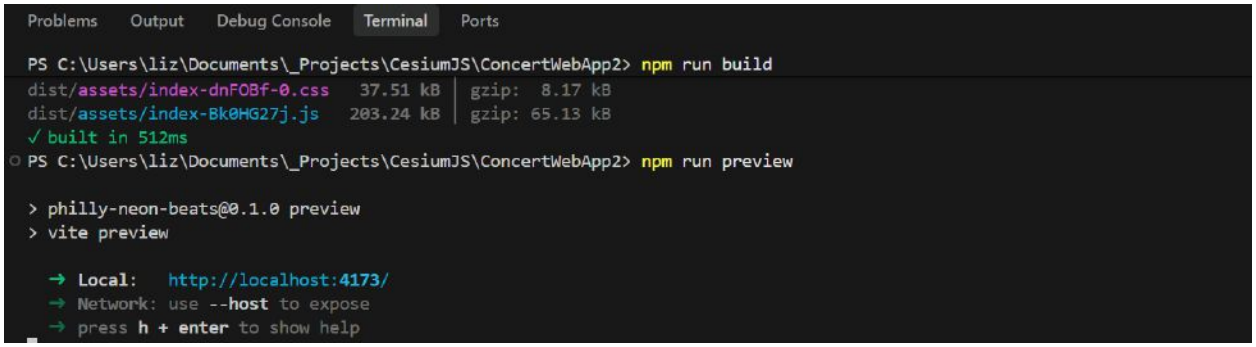
You can copy an error like this directly into the chat with the agent and ask it to assess, solve, and explain it.

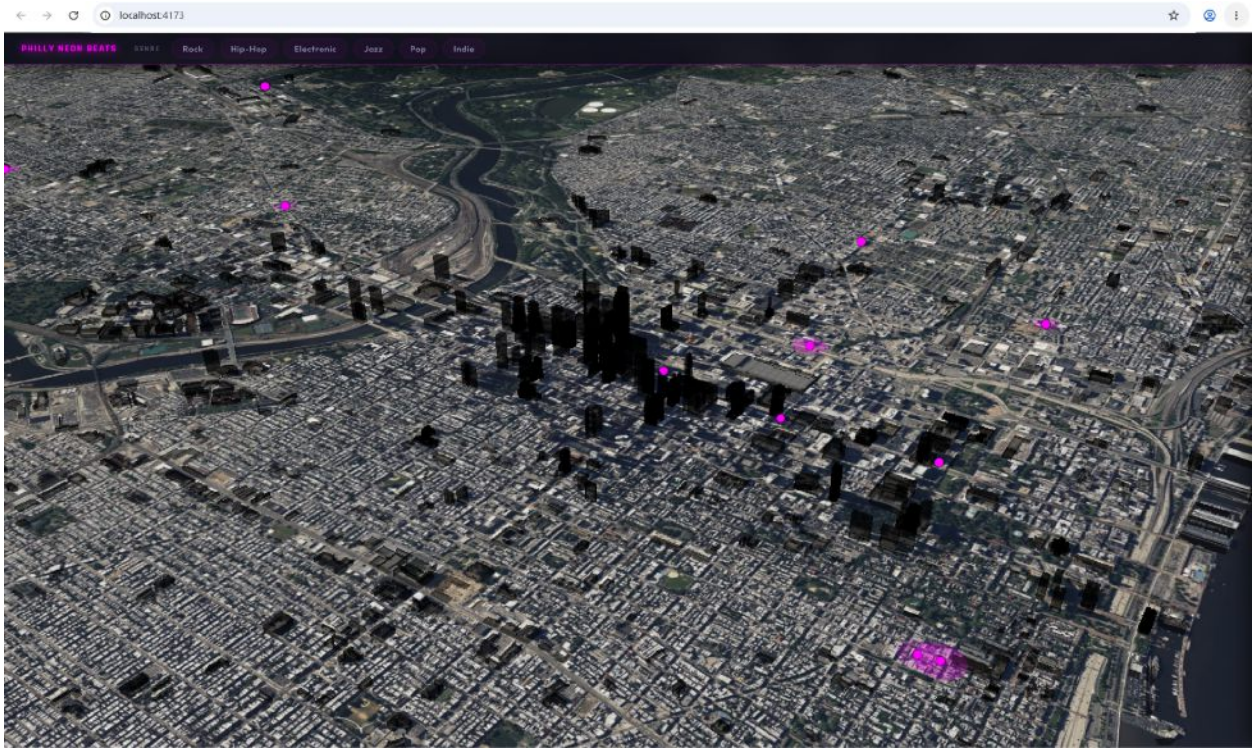
RUN A PRODUCTION BUILD TEST

Run a production build locally before deploying. Catching any issues now saves debugging time on a live server.

```
npm run build    # Creates optimized output in dist/
npm run preview  # Serves dist/ at http://localhost:4173
```

Open http://localhost:4173 in Chrome and run the development server again if you have closed it. Most changes will appear with a refresh in the browser window; however, if that does not work try restarting by pressing **Ctrl+C** while in the terminal window and then run 'npm run dev' again. If that is the last command you ran, you can also click the up arrow on the keyboard to put the same command in the active terminal and press Enter.





Self Check

Could you start the dev server and see your CesiumJS globe with your data layers loaded? Did the production build preview also work correctly? If yes, you are ready to deploy.

Phase 5: Deploy: Go Live on Vercel

Preview

With a working local build, you are ready to publish your web app. The process has two parts: push your code to GitHub, then connect that GitHub repository to Vercel. After the initial setup, all future deployments happen automatically whenever you push new code.

Experiment

INITIALIZE A GIT REPOSITORY

In the terminal, still within the project folder you've been working in, run these commands:

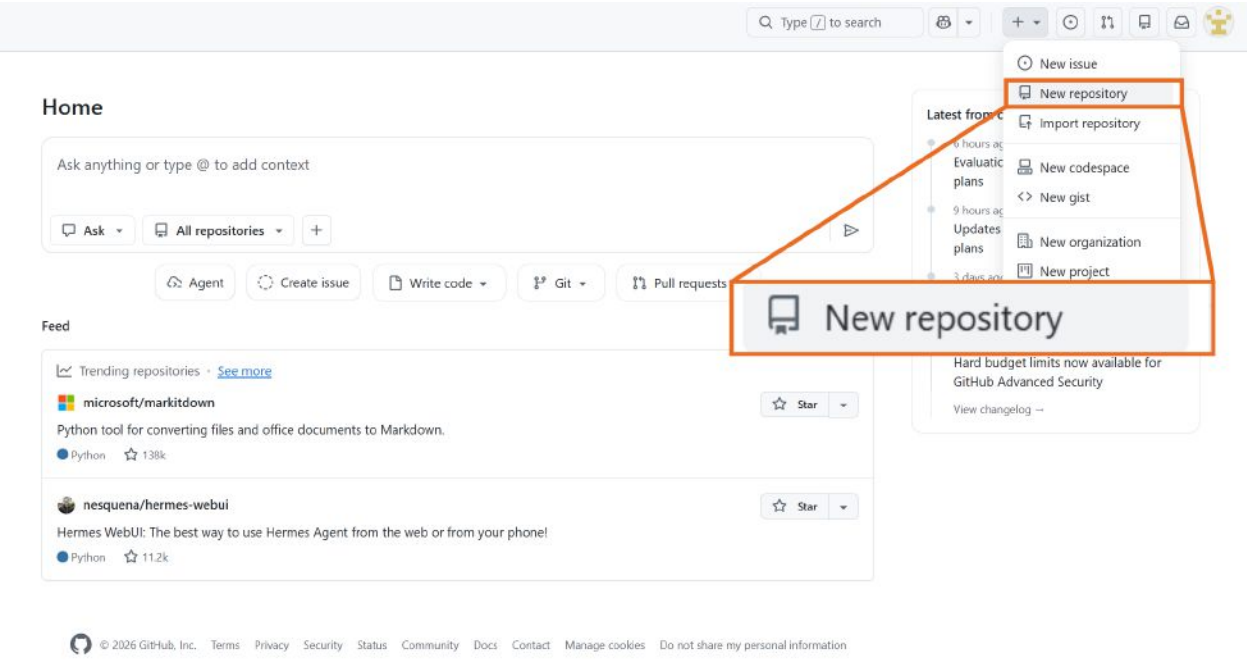
```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"

[note the double dashes '--' above]

git init
git add .
git commit -m "Initial commit"
```

This creates a Git repository and commits all your project files. The .gitignore file (created by the agent) excludes node_modules and .env.local automatically.

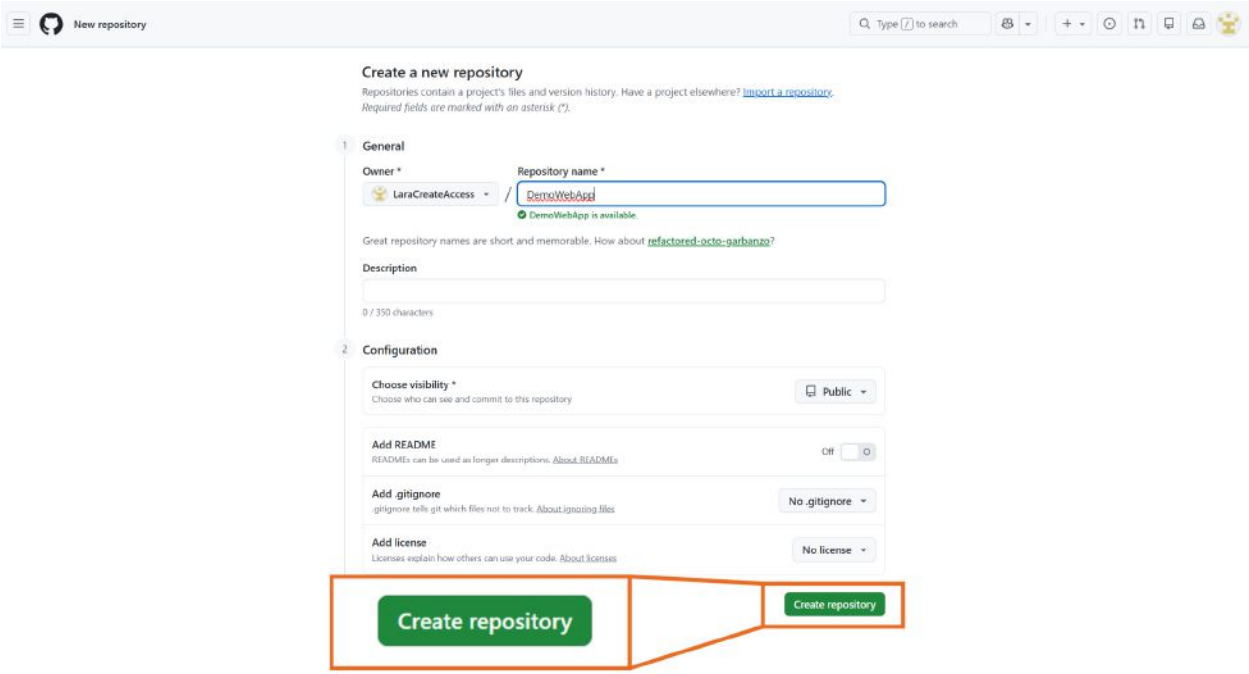
To setup GitHub on your computer,

PUSH TO GITHUB

25. Go to github.com and click + → New repository.

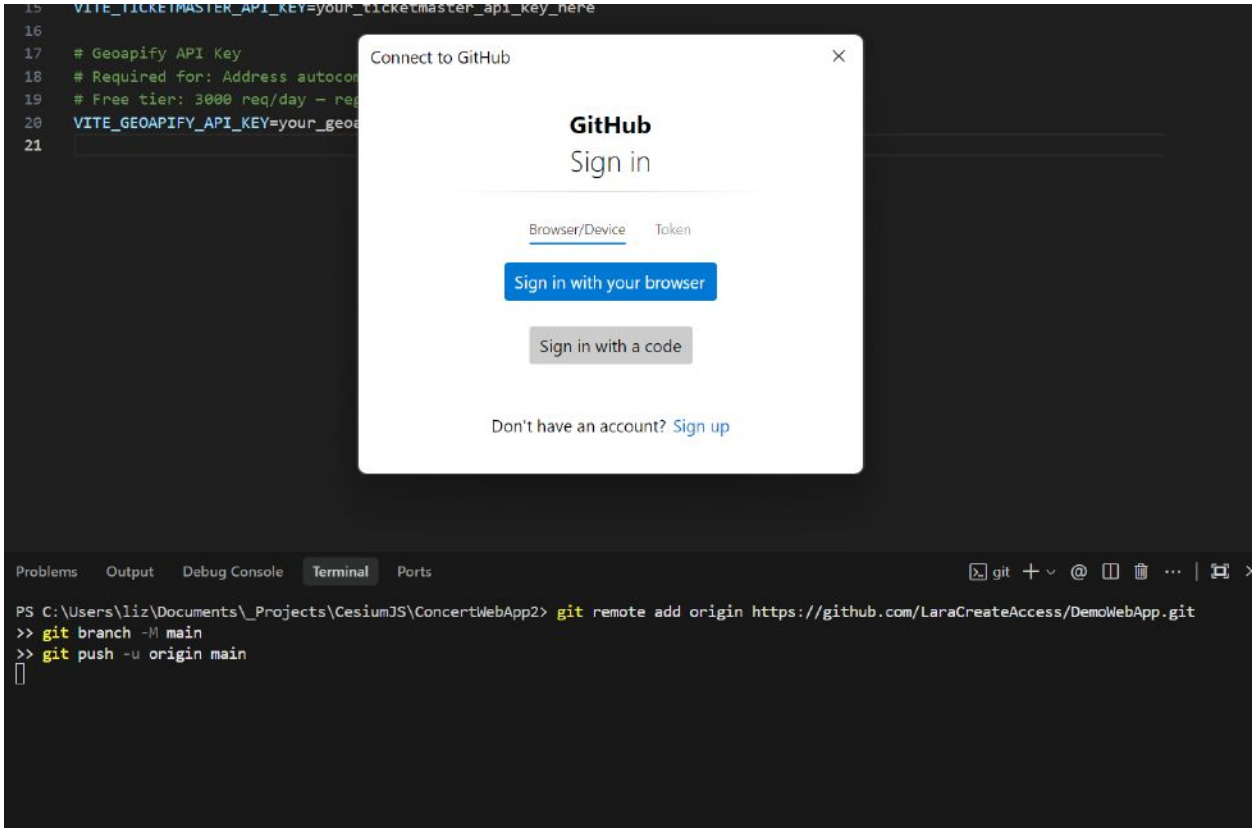


26. Give it a name, leave the other settings as defaults, and click Create repository.



27. GitHub will show commands to push an existing repository. Copy the three lines under "...or push an existing repository from the command line" and run them individually in the terminal. You may be prompted to connect your GitHub account. Do this if you have not already.

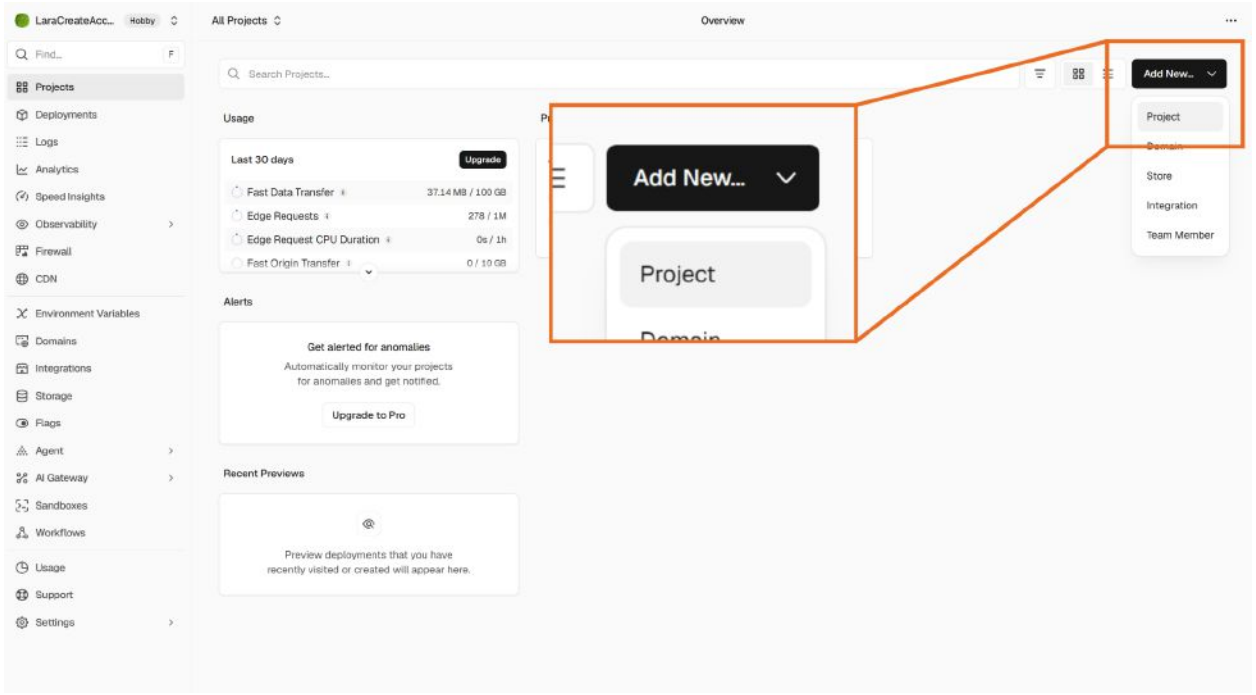
```
git remote add origin https://github.com/YOUR_USERNAME/my-cesium-map.git
git branch -M main
git push -u origin main
```



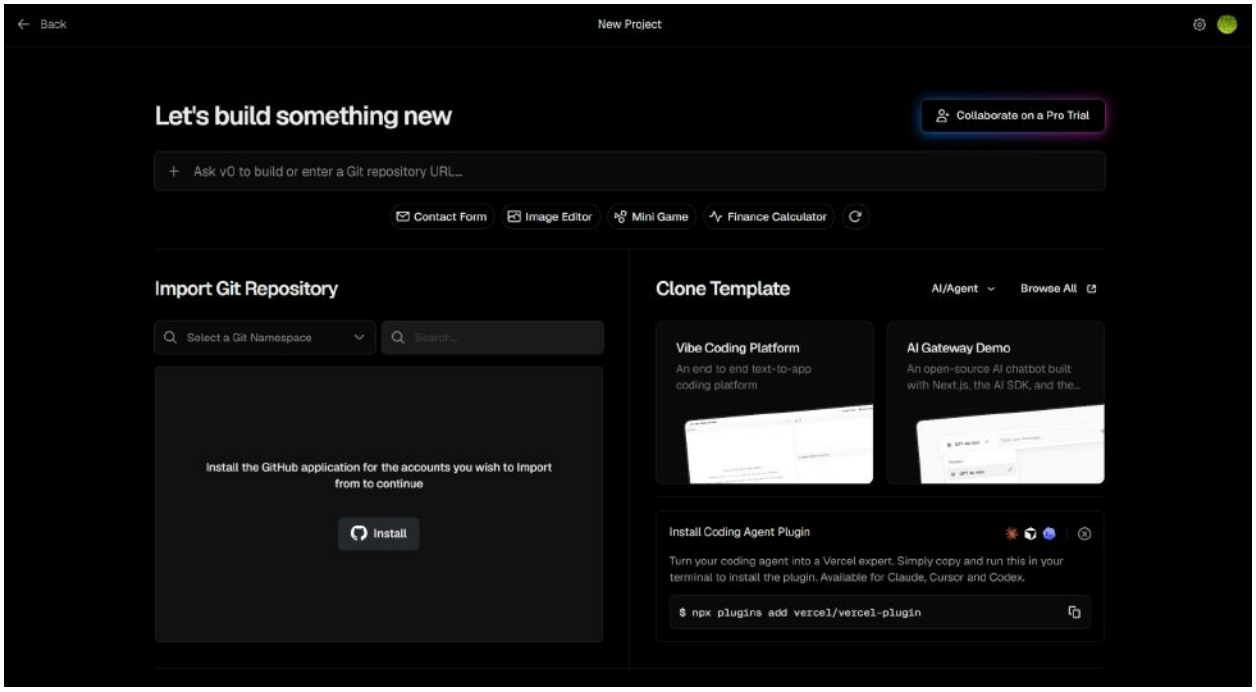
28. Refresh your GitHub page. All your project files should appear there.

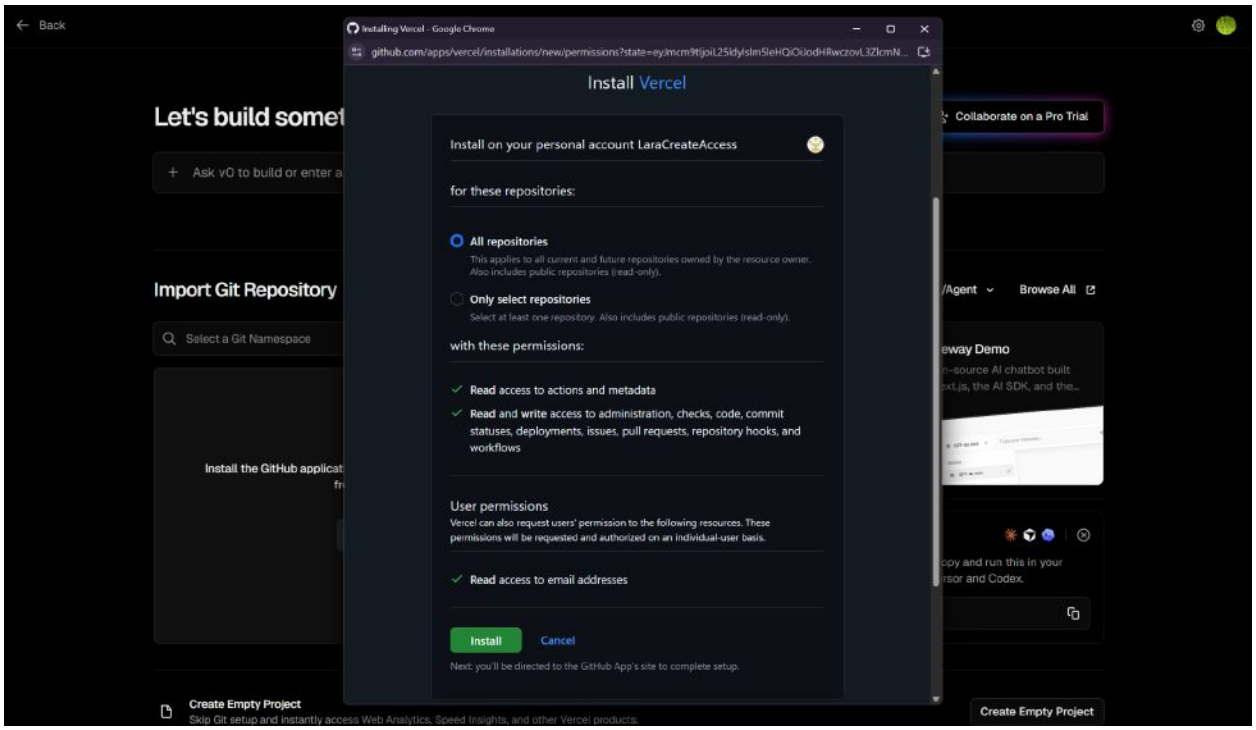
CONNECT VERCEL TO YOUR REPOSITORY

29. Go to vercel.com and sign in.
30. Click Add New → Project.

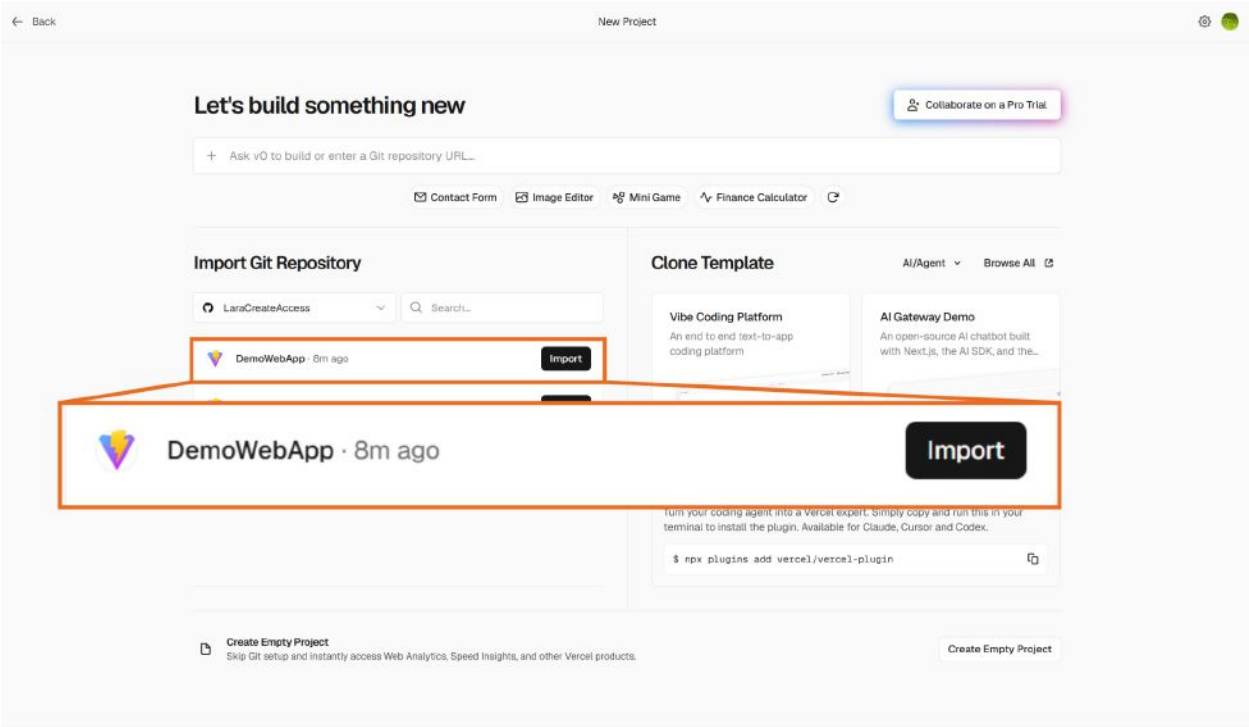


31. Find the “Import Git Repository” section. If using Vercel for the first time, select the “Install” button to sync your Github account. When asked to install Vercel, continue with the default options.





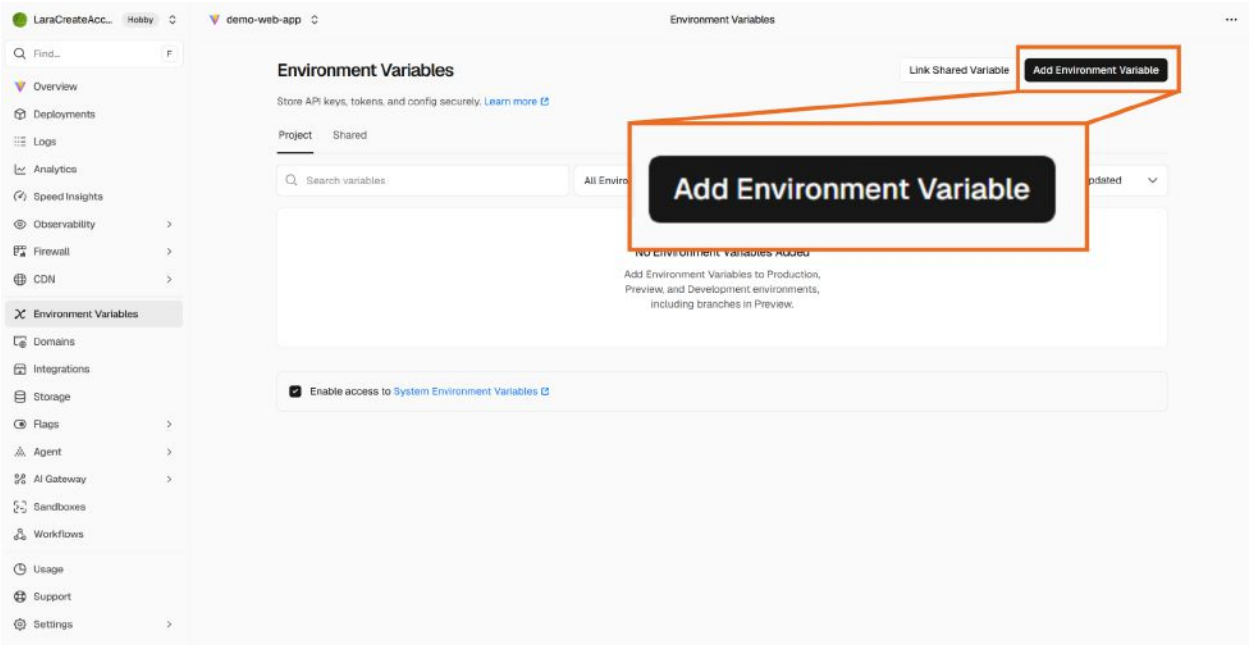
32. Find your project repository under the Import section and click “Import”.



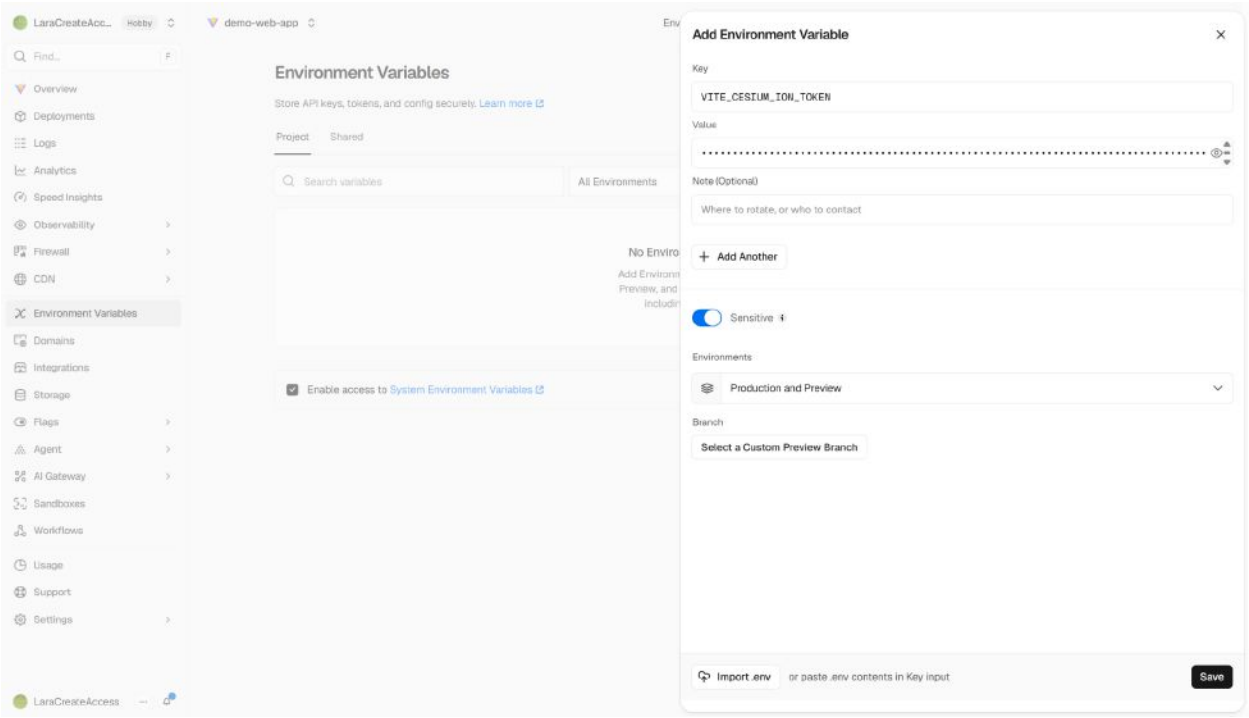
ADD YOUR ENVIRONMENT VARIABLE: DO NOT SKIP THIS STEP

For safety reasons, your .env.local file is not on GitHub (it is gitignored), so Vercel does not know your Cesium ion token. You must add it manually before deploying.

33. In Vercel, in your project, find and expand the **Environment Variables** section on the left panel and then click the “Add Environment Variable” button that appears on the right.

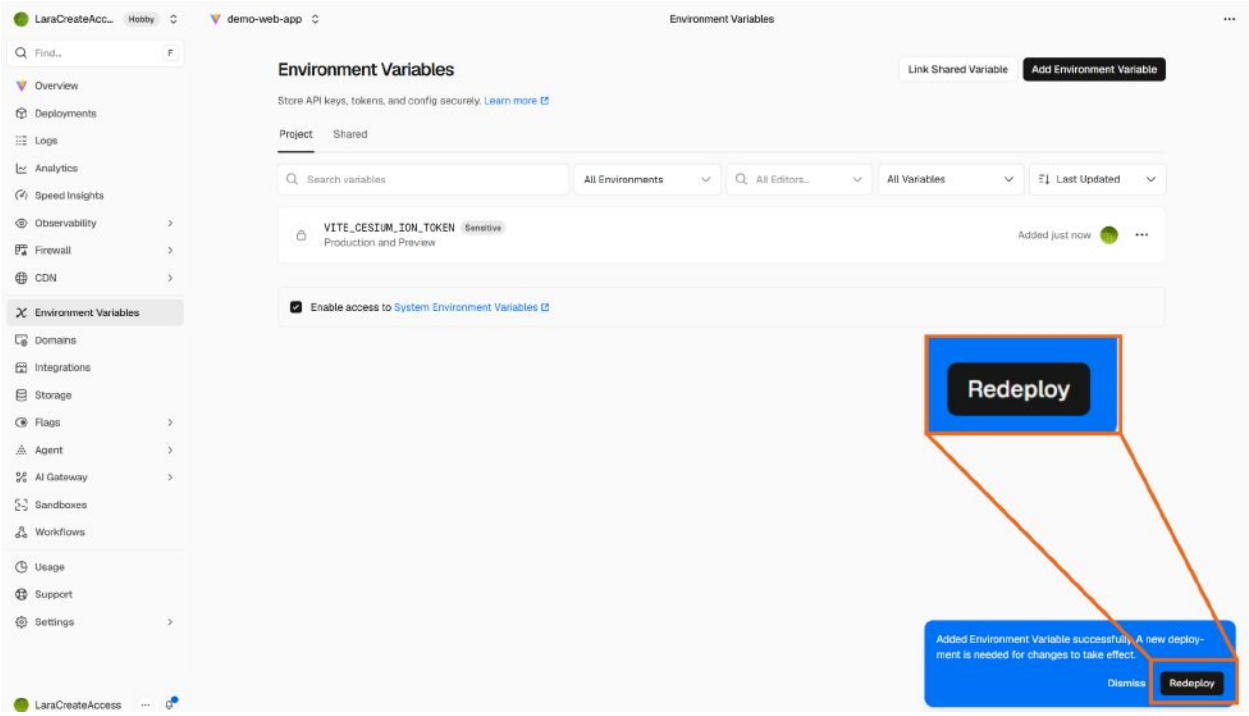


34. On the new panel, enter your Cesium ion token as a new variable using the key and value below.



Key	Value
VITE_CESIUM_ION_TOKEN	*Paste your Cesium ion token here*

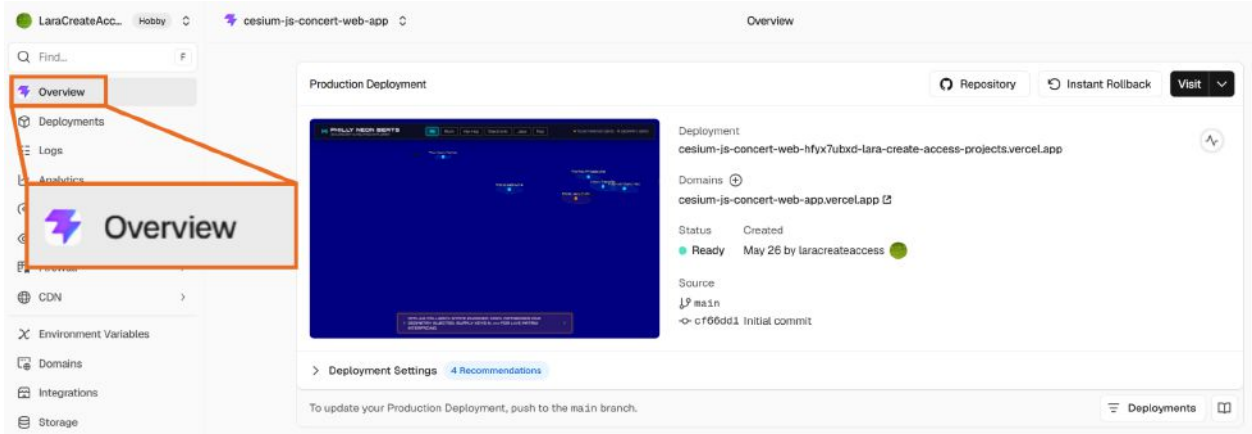
35. Click “Save” and Vercel will prompt you to Redeploy.



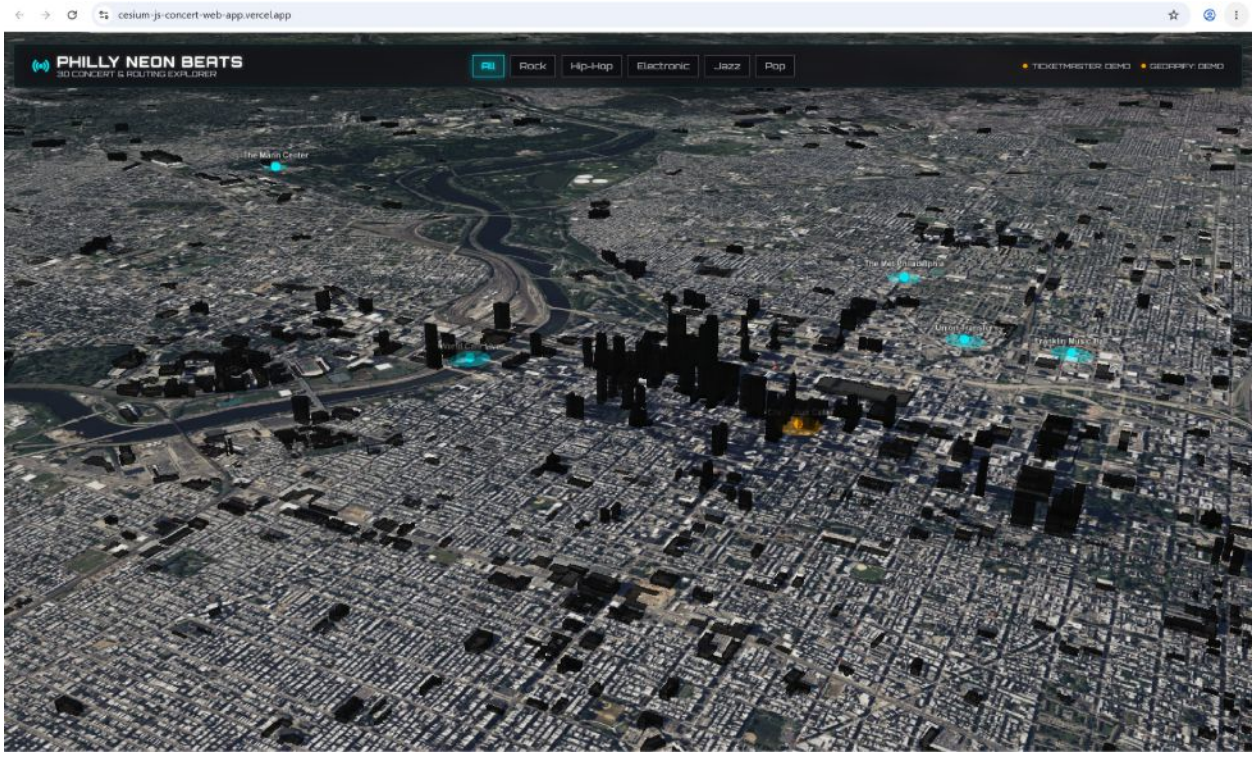
WAIT FOR THE BUILD

Vercel runs “npm run build” on its servers and delivers the output. When it finishes running, Vercel gives you a public URL that you can access in the “Overview” window of your project in the menu on the left. It looks like this:

```
https://my-cesium-map.vercel.app
```



Open it in Chrome. Your CesiumJS web app is now live on the internet.



FUTURE DEPLOYS ARE AUTOMATIC

From now on, every time you make edits to your project and push the code to GitHub, Vercel detects the change in your GitHub repository and re-deploys within about a minute. The next steps will show you how to iterate on your project. Remember, whenever you make changes, run these three commands in succession to update the code in GitHub:

```
git add .
git commit -m "Describe what you changed"
git push
```

Self Check

Could you push your project to GitHub and deploy it on Vercel with your Cesium ion token configured? Open the live URL in Chrome and confirm the globe loads with your data layers.

Phase 6: Iterating on an Existing Project

Preview

Once your project is built and running, you may want to change things. Maybe the data layer needs adjusting, the colors are not right, or you want to add a new feature. This phase explains how to continue working with the agent on a project that already exists.

Experiment

OPEN YOUR EXISTING PROJECT

In Antigravity IDE, open the folder for your existing project using File → Open Folder. Make sure your spec.md file is still in the project root. All your previous code will be exactly as you left it.

DESCRIBE WHAT YOU WANT TO CHANGE

Open the AI chat panel and describe the change you want. Be specific. Instead of saying "make it look better," say what you actually want to change. The agent will read your existing code and make targeted edits without rewriting everything.

Good examples of iteration prompts:

```
The pop-up info panel is not showing the building name. Can you fix it to display
the name field from the GeoJSON?
```

```
Add a button in the top-right corner that resets the camera to the default starting
position.
```

```
The data points are too small on mobile. Make them larger and add a minimum size so
they are always visible.
```

REVIEW AND TEST AFTER EVERY CHANGE

After the agent makes a change, run npm run dev and check the result in Chrome. If something broke, copy the error from the browser console (F12 → Console) and paste it to the agent. Never push changes to Github until the local version looks correct.

Self Check

Could you open your existing project and make a change using the agent? After any successful change, remember to push to GitHub so your live Vercel site stays up to date.

Phase 7: GitHub: Push, Pull, and Update

Preview

Git is the system that tracks every change you make to your code. GitHub is the online home for your project where those changes are stored. This phase explains the three commands you will use most often and how to keep your local project and your GitHub repository in sync.

Experiment

THE THREE-STEP UPDATE PROCESS

Every time you finish a change and want to save it to GitHub, you run the same three commands in the Antigravity terminal:

```
git add .
```

```
git commit -m "Describe what you changed"
```

```
git push
```

What each command does:

- git add . tells Git to include all the files you changed in the next save. The dot means add all of the new or modified files in the project.
- git commit -m "..." creates a snapshot of your code with a short note describing what changed. Write something meaningful, for example: "Fixed pop-up showing wrong building name" or "Added reset camera button".
- git push sends that snapshot to GitHub. This also triggers Vercel to re-deploy your live site automatically.

PULLING UPDATES (WHEN WORKING ON MULTIPLE COMPUTERS)

If you ever work on the same project from a different computer, or if a collaborator has pushed changes, you will need to pull the latest version down before you start working. Run this in the Antigravity terminal:

```
git pull
```

This downloads all changes from GitHub and merges them into your local copy. Always pull before you start working if you have been away from the project for a while.

CHECKING WHAT HAS CHANGED

If you are not sure what files you have changed since your last commit, run:

```
git status
```

Git will list all modified files in red (not yet staged) and green (staged and ready to commit). This is useful before running git add . so you know exactly what is about to be saved.

WRITING GOOD COMMIT MESSAGES

A good commit message says what changed and why, in plain language. You will thank yourself later when you are scrolling back through your history trying to find when something broke.

Good examples:

```
git commit -m "Added neighborhood filter to sidebar"
```

```
git commit -m "Fixed crash when clicking outside data layer"
```

```
git commit -m "Updated Cesium ion token to new value"
```

Avoid vague messages like "stuff" or "fix" or "update". They are not helpful when you need to look back later.

Self Check

Can you run the three-step add, commit, push process and see your changes appear on GitHub? Check your Vercel deployment to confirm the live site is updated as well.

Lesson Closure

Demonstration of Learning

You have installed all the required tools, created the necessary accounts, generated a project blueprint using the CesiumJS Gem, built the application with Antigravity IDE, tested it locally with both the dev server and a production build, and published it to the internet using Vercel. Your CesiumJS geospatial web app is live at a public URL.

Exploration Opportunities

Now that your project is live, you can iterate on it freely. Update data layers, add new interactivity, or refine the visual design. Then push to GitHub and Vercel re-deploys automatically.

When you are ready to add a new feature, return to the CesiumJS Gem and describe what you want to add or build directly in Antigravity. The agent can extend the codebase without breaking what already works.

CreateAccess

Quick Reference

Terminal Commands

Command	What it does
`npm install`	Download all project dependencies
`npm run dev`	Start local dev server at localhost:5173
`npm run build`	Create optimized production build in dist/
`npm run preview`	Serve the production build at localhost:4173
`git add .`	Stage all changed files
`git commit -m "..."`	Commit staged changes with a message
`git push`	Push commits to GitHub (triggers Vercel redeploy)

Tools and Accounts

Tool	Where to get it	Notes
Node.js v20 LTS	nodejs.org	Choose LTS. Comes with npm
Git	git-scm.com (Windows) / built-in Mac	
Antigravity IDE	antigravity.google	Sign in with personal Gmail
Chrome	google.com/chrome	Preferred browser
GitHub account	github.com	Free; link to Vercel
Cesium ion account	cesium.com/ion/signup	Free; copy your access token
Vercel account	vercel.com	Sign up with GitHub