Install and set up Astro

The `create astro` CLI command is the fastest way to start a new Astro project from scratch. It will walk you through every step of setting up your new Astro project and allow you to choose from a few different official starter templates.

Or, you can begin your project using any existing theme or starter template.

To install Astro manually instead, see our step-by-step manual installation guide.

> **Online previews**
>
> Prefer to try Astro in your browser? Visit astro.new to browse our starter templates and spin up a new Astro project without ever leaving your browser.

Prerequisites

- **Node.js** - `v18.17.1` or `v20.3.0` or higher. (`v19` is not supported.)
- **Text editor** - We recommend VS Code with our Official Astro extension.
- **Terminal** - Astro is accessed through its command-line interface (CLI).

Start a new project

Install from the CLI wizard

1. Run the following command in your terminal to start our handy install wizard:

npm pnpm yarn

```
# create a new project with npm
npm create astro@latest
```

You can run `create astro` anywhere on your machine, so there's no need to create a new empty directory for your project before you begin. If you don't have an empty directory yet for your new project, the wizard will help create one for you automatically.

If all goes well, you will see a success message followed by some recommended next steps. Now that your project has been created, you can `cd` into your new project directory to begin using Astro.

2. If you skipped the `npm install` step during the CLI wizard, then be sure to install your dependencies before continuing.

3. You can now start the Astro dev server and see a live preview of your project while you build!

Use a theme or starter template

You can also start a new astro project based on an official example or the `main` branch of any GitHub repository by passing a `--template` argument to the `create astro` command.

1. Explore our themes and starters showcase where you can browse themes for blogs, portfolios, documentation sites, landing pages, and more! Or, search on GitHub for even more starter projects.

2. Run the following command in your terminal, substituting the official Astro starter template name, or the GitHub username and repository of the theme you want to use:

npm pnpm yarn

```
# create a new project with an official example
npm create astro@latest -- --template <example-name>

# create a new project based on a GitHub repository's main branch
npm create astro@latest -- --template <github-username>/<github-repo>
```

By default, this command will use the template repository's `main` branch. To use a different branch name, pass it as part of the `--template` argument: `<github-username>/<github-repo>#<branch>`.

3. Answer the questions and follow the instructions of the CLI wizard.

4. You can now start the Astro dev server and see a live preview of your project while you make it your own!

Edit your project

To make changes to your project, open your project folder in your code editor. Working in development mode with the dev server running allows you to see updates to your site as you edit the code.

You can also customize aspects of your development environment such as configuring TypeScript or installing the official Astro editor extensions.

Start the Astro dev server

Astro comes with a built-in development server that has everything you need for project development. The `astro dev` CLI command will start the local development server so that you can see your new website in action for the very first time.

Every starter template comes with a pre-configured script that will run `astro dev` for you. After navigating into your project directory, use your favorite package manager to run this command and start the Astro development server.

npm pnpm yarn

```
npm run dev
```

If all goes well, Astro will now be serving your project on http://localhost:4321/. Visit that link in your browser and see your new site!

Work in development mode

Astro will listen for live file changes in your `src/` directory and update your site preview as you build, so you will not need to restart the server as you make changes during development. You will always be able to see an up-to-date version of your site in your browser when the dev server is running.

When viewing your site in the browser, you'll have access to the Astro dev toolbar. As you build, it will help you inspect your islands, spot accessibility issues, and more.

If you aren't able to open your project in the browser after starting the dev server, go back to the terminal where you ran the `dev` command and check the message displayed. It should tell you if an error occurred, or if your project is being served at a different URL than http://localhost:4321/.

Configure your dev environment

Explore the guides below to customize your development experience.

Editor Setup →

Customize your code editor to improve the Astro developer experience and unlock new features.

Dev Toolbar →

Explore the helpful features of the dev toolbar.

TypeScript in Astro

Astro ships with built-in support for TypeScript, which can help prevent errors at runtime by defining the shapes of objects and components in your code.

You don't need to write TypeScript code in your Astro projects to benefit from it. Astro always treats your component code as TypeScript, and the Astro VSCode Extension will infer as much as it can to provide autocompletion, hints, and errors in your editor.

Read more about using and configuring TypeScript in Astro

Build and preview your site

To check the version of your site that will be created at build time, quit the dev server (`Ctrl + C`) and run the appropriate build command for your package manager in your terminal:

npm pnpm yarn

```
npm run build
```

Astro will build a deploy-ready version of your site in a separate folder (`dist/` by default) and you can watch its progress in the terminal. This will alert you to any build errors in your project before you deploy to production. If TypeScript is configured to `strict` or `strictest`, the `build` script will also check your project for type errors.

When the build is finished, run the appropriate `preview` command (e.g. `npm run preview`) in your terminal and you can view the built version of your site locally in the same browser preview window.

Note that this previews your code as it existed when the build command was last run. This is meant to give you a preview of how your site will look when it is deployed to the web. Any later changes you make to your code after building will **not** be reflected while you preview your site until you run the build command again.

Use (`Ctrl + C`) to quit the preview and run another terminal command, such as restarting the dev server to go back to working in development mode which does update as you edit to show a live preview of your code changes.

Read more about the Astro CLI and the terminal commands you will use as you build with Astro.

Deploy your new site

You may wish to deploy your new site right away, before you begin to add or change too much code. This is helpful to get a minimal, working version of your site published and can save you extra time and effort troubleshooting your deployment later.

Next Steps

Success! You are now ready to start building with Astro! 🥳

Here are a few things that we recommend exploring next. You can read them in any order. You can even leave our documentation for a bit and go play in your new Astro project codebase, coming back here whenever you run into trouble or have a question.

Explore Astro's Features

Understand your codebase →

Learn about Astro's file structure in our Project Structure guide.

Create content collections →

Add content to your new site with frontmatter validation and automatic type-safety.

Add view transitions →

Create seamless page transitions and animations.

Learn about Islands →

Read about Astro's island architecture.

Take the introductory tutorial

Build a fully functional Astro blog starting from a single blank page in our introductory tutorial.

This is a great way to see how Astro works and walks you through the basics of pages, layouts, components, routing, islands, and more. It also includes an optional, beginner-friendly unit for those newer to web development concepts in general, which will guide you through installing the necessary applications on your computer, creating a GitHub account, and deploying your site.

Manual Setup

This guide will walk you through the steps to manually install and configure a new Astro project.

If you prefer not to use our automatic `create astro` CLI tool, you can set up your project yourself by following the guide below.

1 Create your directory

Create an empty directory with the name of your project, and then navigate into it.

```
mkdir my-astro-project
cd my-astro-project
```

Once you are in your new directory, create your project `package.json` file. This is how you will manage your project dependencies, including Astro. If you aren't familiar with this file format, run the following command to create one.

npm pnpm yarn

```
npm init --yes
```

2 Install Astro

First, install the Astro project dependencies inside your project.

> **Important**
>
> Astro must be installed locally, not globally. Make sure you are *not* running `npm install -g astro` `pnpm add -g astro` or `yarn add global astro`.

npm pnpm yarn

```
npm install astro
```

Then, replace any placeholder "scripts" section of your `package.json` with the following:

package.json

```
  "scripts": {
-   "test": "echo \"Error: no test specified\" && exit 1",
+   "dev": "astro dev",
+   "start": "astro dev",
+   "build": "astro build",
+   "preview": "astro preview"
  },
```

You'll use these scripts later in the guide to start Astro and run its different commands.

3 Create your first page

In your text editor, create a new file in your directory at `src/pages/index.astro`. This will be your first Astro page in the project.

For this guide, copy and paste the following code snippet (including `---` dashes) into your new file:

src/pages/index.astro

```
---
// Welcome to Astro! Everything between these triple-dash code fences
// is your "component frontmatter". It never runs in the browser.
console.log('This runs in your terminal, not the browser!');
---
<!-- Below is your "component template." It's just HTML, but with
    some magic sprinkled in to help you build great templates. -->
<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
<style>
  h1 {
    color: orange;
  }
</style>
```

4 Create your first static asset

You will also want to create a `public/` directory to store your static assets. Astro will

always include these assets in your final build, so you can safely reference them from inside your component templates.

In your text editor, create a new file in your directory at `public/robots.txt`. `robots.txt` is a simple file that most sites will include to tell search bots like Google how to treat your site.

For this guide, copy and paste the following code snippet into your new file:

public/robots.txt

```
# Example: Allow all bots to scan and index your site.
# Full syntax: https://developers.google.com/search/docs/advanced/robots/create-ro
User-agent: *
Allow: /
```

5 Create `astro.config.mjs`

Astro is configured using `astro.config.mjs`. This file is optional if you do not need to configure Astro, but you may wish to create it now.

Create `astro.config.mjs` at the root of your project, and copy the code below into it:

astro.config.mjs

```
import { defineConfig } from 'astro/config';

// https://astro.build/config
export default defineConfig({});
```

If you want to include UI framework components such as React, Svelte, etc. or use other tools such as Tailwind or Partytown in your project, here is where you will manually import and configure integrations.

Read Astro's API configuration reference for more information.

6 Add TypeScript support

TypeScript is configured using `tsconfig.json`. Even if you don't write TypeScript code, this file is important so that tools like Astro and VS Code know how to understand your project. Some features (like npm package imports) aren't fully supported in the editor without a

`tsconfig.json` file.

If you do intend to write TypeScript code, using Astro's `strict` or `strictest` template is recommended. You can view and compare the three template configurations at astro/tsconfigs/.

Create `tsconfig.json` at the root of your project, and copy the code below into it. (You can use `base`, `strict`, or `strictest` for your TypeScript template):

tsconfig.json

```
{
  "extends": "astro/tsconfigs/base"
}
```

Finally, create `src/env.d.ts` to let TypeScript know about ambient types available in an Astro project:

src/env.d.ts

```
/// <reference types="astro/client" />
```

Read Astro's TypeScript setup guide for more information.

7 Next Steps

If you have followed the steps above, your project directory should now look like this:

```
▶ node_modules/
▼ public/
    robots.txt
▼ src/
  ▼ pages/
      index.astro
    env.d.ts
  astro.config.mjs
  package-lock.json    or yarn.lock, pnpm-lock.yaml, etc.
  package.json
  tsconfig.json
```

8 You can now start the Astro dev server and see a live preview of your project while you build!

Edit page

Previous
Getting Started

Next
Deploy Your Site