static-deploy

- Deploying a Static Site
  - Building the App
    - Testing the App Locally
  - GitHub Pages
  - GitLab Pages and GitLab CI
  - Netlify
    - Netlify CLI
    - Netlify with Git
  - Vercel
    - Vercel CLI
    - Vercel with Git
  - Cloudflare
    - Cloudflare Workers
    - Cloudflare Pages
      - Cloudflare Pages with Git
  - Google Firebase
  - Surge
  - Azure Static Web Apps
  - Render
  - Flightcontrol
  - xmit Static Site Hosting
  - Zephyr Cloud
  - EdgeOne Pages

Deploying a Static Site

The following guides are based on some shared assumptions:

- You are using the default build output location (`dist`). This location can be changed using `build.outDir`, and you can extrapolate instructions from these guides in that case.
- You are using npm. You can use equivalent commands to run the scripts if you are using Yarn or other package managers.
- Vite is installed as a local dev dependency in your project, and you have setup the following npm scripts:

```
{
  "scripts": {
    "build": "vite build",
    "preview": "vite preview"
  }
}
```

It is important to note that `vite preview` is intended for previewing the build locally and not meant as a production server.

::: tip NOTE These guides provide instructions for performing a static deployment of your Vite site. Vite also supports Server-Side Rendering. SSR refers to front-end frameworks that support running the same application in Node.js, pre-rendering it to HTML, and finally hydrating it on the client. Check out the SSR Guide to learn about this feature. On the other hand, if you are looking for integration with traditional server-side frameworks, check out the Backend Integration guide instead. :::

Building the App

You may run `npm run build` command to build the app.

```
$ npm run build
```

By default, the build output will be placed at `dist`. You may deploy this `dist` folder to any of your preferred platforms.

Testing the App Locally

Once you've built the app, you may test it locally by running `npm run preview` command.

```
$ npm run preview
```

The `vite preview` command will boot up a local static web server that serves the files from `dist` at `http://localhost:4173`. It's an easy way to check if the production build looks OK in your local environment.

You may configure the port of the server by passing the `--port` flag as an argument.

```
{
  "scripts": {
    "preview": "vite preview --port 8080"
  }
}
```

Now the `preview` command will launch the server at `http://localhost:8080`.

GitHub Pages

1. **Update Vite Config**
   Set the correct `base` in `vite.config.js`.
   If you are deploying to `https://<USERNAME>.github.io/`, or to a custom domain through GitHub Pages (eg. `www.example.com`), set `base` to `'/'`. Alternatively, you can remove `base` from the configuration, as it defaults to `'/'`.
   If you are deploying to `https://<USERNAME>.github.io/<REPO>/` (eg. your repository is at `https://github.com/<USERNAME>/<REPO>`), then set `base` to `'/<REPO>/'`.
2. **Enable GitHub Pages**
   In your repository, go to **Settings → Pages**. Under **Build and deployment**, open the **Source** dropdown, and select **GitHub Actions**.

   GitHub will now deploy your site using a GitHub Actions workflow, which is necessary since Vite requires a build step for deployment.
3. **Create a Workflow**
   Create a new file in your repository at `.github/workflows/deploy.yml`. You can also click on **“create your own”** from the previous step, which will generate a starter workflow file for you.
   Here’s a sample workflow that installs dependencies with npm, builds the site, and deploys it whenever you push changes to the `main` branch:
   <<< ./static-deploy-github-pages.yaml#content [.github/workflows/deploy.yml]

GitLab Pages and GitLab CI

1. Set the correct `base` in `vite.config.js`.

If you are deploying to `https://<USERNAME or GROUP>.gitlab.io/`, you can omit `base` as it defaults to `'/'`.
If you are deploying to `https://<USERNAME or GROUP>.gitlab.io/<REPO>/`, for example your repository is at `https://gitlab.com/<USERNAME>/<REPO>`, then set `base` to `'/<REPO>/'`.

2. Create a file called `.gitlab-ci.yml` in the root of your project with the content below. This will build and deploy your site whenever you make changes to your content:

```
image: node:lts
pages:
  stage: deploy
  cache:
    key:
      files:
        - package-lock.json
      prefix: npm
    paths:
      - node_modules/
  script:
    - npm install
    - npm run build
    - cp -a dist/. public/
  artifacts:
    paths:
      - public
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Netlify

Netlify CLI

1. Install the Netlify CLI via `npm install -g netlify-cli`.
2. Create a new site using `netlify init`.
3. Deploy using `netlify deploy`.

The Netlify CLI will share with you a preview URL to inspect. When you are ready to go into production, use the `prod` flag: `netlify deploy --prod`.

Netlify with Git

1. Push your code to a git repository (GitHub, GitLab, BitBucket, Azure DevOps).
2. Import the project to Netlify.
3. Choose the branch, output directory, and set up environment variables if applicable.
4. Click on **Deploy**.
5. Your Vite app is deployed!

After your project has been imported and deployed, all subsequent pushes to branches other than the production branch along with pull requests will generate Preview Deployments, and all changes made to the Production Branch (commonly “main”) will result in a Production Deployment.

Vercel

Vercel CLI

1. Install the Vercel CLI via `npm i -g vercel` and run `vercel` to deploy.
2. Vercel will detect that you are using Vite and will enable the correct settings for your deployment.
3. Your application is deployed! (e.g. vite-vue-template.vercel.app)

Vercel with Git

1. Push your code to your git repository (GitHub, GitLab, Bitbucket).
2. Import your Vite project into Vercel.
3. Vercel will detect that you are using Vite and will enable the correct settings for your deployment.
4. Your application is deployed! (e.g. vite-vue-template.vercel.app)

After your project has been imported and deployed, all subsequent pushes to branches will generate Preview Deployments, and all changes made to the Production Branch (commonly “main”) will result in a Production Deployment.

Learn more about Vercel’s Git Integration.

Cloudflare

Cloudflare Workers

The Cloudflare Vite plugin provides integration with Cloudflare Workers and uses Vite's Environment API to run your server-side code in the Cloudflare Workers runtime during development.

To add Cloudflare Workers to an existing Vite project, install the plugin and add it to your config:

```
$ npm install --save-dev @cloudflare/vite-plugin
```

```
import { cloudflare } from '@cloudflare/vite-plugin'
import { defineConfig } from 'vite'

export default defineConfig({
  plugins: [cloudflare()],
})
```

```
{
  "name": "my-vite-app",
}
```

After running `npm run build`, your application can now be deployed with `npx wrangler deploy`.

You can also easily add backend APIs to your Vite application to securely communicate with Cloudflare resources. This runs in the Workers runtime during development and deploys alongside your frontend. See the Cloudflare Vite plugin tutorial for a complete walkthrough.

Cloudflare Pages

Cloudflare Pages with Git

Cloudflare Pages gives you a way to deploy directly to Cloudflare without having to manage a Wrangler file.

1. Push your code to your git repository (GitHub, GitLab).
2. Log in to the Cloudflare dashboard and select your account in **Account Home** > **Workers & Pages**.

3. Select **Create a new Project** and the **Pages** option, then select Git.
4. Select the git project you want to deploy and click **Begin setup**
5. Select the corresponding framework preset in the build setting depending on the Vite framework you have selected. Otherwise enter your build commands for your project and your expected output directory.
6. Then save and deploy!
7. Your application is deployed! (e.g `https://<PROJECTNAME>.pages.dev/`)

After your project has been imported and deployed, all subsequent pushes to branches will generate Preview Deployments unless specified not to in your branch build controls. All changes to the Production Branch (commonly "main") will result in a Production Deployment.

You can also add custom domains and handle custom build settings on Pages. Learn more about Cloudflare Pages Git Integration.

Google Firebase

1. Install firebase-tools via `npm i -g firebase-tools`.
2. Create the following files at the root of your project:
   ::: code-group

```
{
  "hosting": {
    "public": "dist",
    "ignore": [],
    "rewrites": [
      {
        "source": "**",
        "destination": "/index.html"
      }
    ]
  }
}
```

```
{
  "projects": {
    "default": "<YOUR_FIREBASE_ID>"
  }
}
```

:::
3. After running `npm run build`, deploy using the command `firebase deploy`.

Surge

1. Install surge via `npm i -g surge`.
2. Run `npm run build`.
3. Deploy to surge by typing `surge dist`.

You can also deploy to a custom domain by adding `surge dist yourdomain.com`.

Azure Static Web Apps

You can quickly deploy your Vite app with Microsoft Azure Static Web Apps service. You need:

- An Azure account and a subscription key. You can create a free Azure account here.
- Your app code pushed to GitHub.
- The SWA Extension in Visual Studio Code.

Install the extension in VS Code and navigate to your app root. Open the Static Web Apps extension, sign in to Azure, and click the '+' sign to create a new Static Web App. You will be prompted to designate which subscription key to use.

Follow the wizard started by the extension to give your app a name, choose a framework preset, and designate the app root (usually `/`) and built file location `/dist`. The wizard will run and will create a GitHub action in your repo in a `.github` folder.

The action will work to deploy your app (watch its progress in your repo's Actions tab) and, when successfully completed, you can view your app in the address provided in the extension's progress window by clicking the 'Browse Website' button that appears when the GitHub action has run.

Render

You can deploy your Vite app as a Static Site on Render.

1. Create a Render account.
2. In the Dashboard, click the **New** button and select **Static Site**.

3. Connect your GitHub/GitLab account or use a public repository.
4. Specify a project name and branch.
   - **Build Command**: `npm install && npm run build`
   - **Publish Directory**: `dist`
5. Click **Create Static Site**. Your app should be deployed at `https://<PROJECTNAME>.onrender.com/`.

By default, any new commit pushed to the specified branch will automatically trigger a new deployment. Auto-Deploy can be configured in the project settings.

You can also add a custom domain to your project.

Flightcontrol

Deploy your static site using Flightcontrol by following these instructions.

xmit Static Site Hosting

Deploy your static site using xmit by following this guide.

Zephyr Cloud

Zephyr Cloud is a deployment platform that integrates directly into your build process and provides global edge distribution for module federation and other kind of applications.

Zephyr follows a different approach than other cloud providers. It integrates directly with Vite build process, so every time you build or run the dev server for your application, it will be automatically deployed with Zephyr Cloud.

Follow the steps in the Vite deployment guide to get started.

EdgeOne Pages

Deploy your static site using EdgeOne Pages by following these instructions.

□□□ #2
□ □□□ □□□ 2026-10-31 13:15:02 UTC
□ □□□ □□□ 2026-10-31 13:15:04 UTC