Deploy a static website with a rollback directory

Uploading directly over a live website can leave a mixture of old and new files. A separate release directory makes verification and rollback easier. This guide covers a small static HTML, CSS and JavaScript site served by Nginx on Debian or Ubuntu. It does not cover database migrations, user uploads or applications that write into their release directory.

Record the existing setup

Use an administrative shell to inspect the relevant Nginx site file and its current root directory. Save a private copy of that configuration and record which release is live. Confirm working HTTPS and check `sudo nginx -t` before changing anything. An already invalid configuration should be fixed separately.

Choose a release directory outside the live root, for example `/srv/example/releases/release-2026-11-09`. The path is illustrative. Confirm the filesystem has space for both releases and that the deployment user can create the new directory without granting public write access. Preserve ownership expectations for the Nginx worker account.

Stage and inspect the files

Upload or copy the complete build into the new directory. Keep environment files, source-control metadata, private backups and build credentials out of the served tree. Review archive contents before extraction. Avoid a broad recursive copy from an unfamiliar parent directory.

Check the homepage, asset filenames, capitalization and relative links. Linux paths are case-sensitive. A local preview can miss an absolute URL still pointing at a development domain, so search generated files for staging hostnames. If a service worker or long-lived asset cache is used, plan its version change as part of the release.

Switch a single site deliberately

For a simple existing site, update only that server block's root to the new release directory. Use the established configuration layout rather than adding a second competing server block. Test again before reloading:

```
sudo nginx -t
sudo systemctl reload nginx
```

Run the reload only if the syntax test succeeds. A reload asks Nginx to use the new configuration without the same interruption as a full stop and start. Check its return status and fresh service logs. A directory-symlink release design is another option, but introduces its own ownership, path and cache considerations; do not mix designs during an incident.

Verify the public result

Open the real HTTPS URL in a fresh browser session. Check a page below the homepage, stylesheets, images and any client-side route fallback. Inspect HTTP errors and mixed-content warnings. Confirm the expected release marker or visible edit rather than relying only on a successful status code.

Test with cache bypass where appropriate, and distinguish the origin from a CDN's cached response. Do not purge unrelated customer content. Static files containing secrets remain a security issue even if the homepage renders correctly.

Roll back a bad release

Restore the previous root path in the saved site configuration, run the Nginx syntax check and reload after it passes. Repeat the public checks. Keep the failed release for investigation in a location that is not served. Because this procedure excludes mutable application data, rollback does not attempt to reverse database or upload changes. Retain the previous good release until your normal release-retention policy permits cleanup.

Related instructions

Configure a domain and HTTPS. Verify a downloaded backup.

Reference: Official documentation. Check the documentation installed on your own operating-system release when an option differs.

Revision #1
Created 2026-11-09 10:42:42 UTC by CloudExa Hosting Admin

Updated 2026-11-09 10:42:42 UTC by CloudExa Hosting Admin