Build and publish your first site

For a builder's first day on the Website Strategy platform, the hosting service that builds and serves customers' websites. By the end you have `ws` installed, a site open on your phone and a published **sample** with a link to share. A sample is a free site for a business that is not a customer yet. You also learn to answer an owner's request. It takes about 30 minutes.

The examples follow Jordan Bell, a builder, through a site for Saltmere Marina. Your names, links and codes will differ.

Before you start

You need:

- **An Outport staff account.** Outport is Sevenview's sign-in for its own tools, at https://outport.app. Jordan signs in as `jordan@example.com`.
- **Git.** `ws` uses Git to send and fetch a site's files.
- **A phone or a second browser,** to open previews.

1. Install ws and sign in

1. Sign in at https://outport.app with your Outport staff account.
2. Open **Tools > Software**, at https://outport.app/tools/software, and install `ws` for macOS or Linux. If you already have Outport's `out` tool, `out upgrade ws` installs it too, and `out upgrade` keeps it up to date.
3. Check that it runs:

   ```
   $ ws version
   ws 0.1.56
   ```

   Your number may be higher.
4. Check that Git is installed and knows your name and email, which every commit records:

```
$ git --version
$ git config user.name
$ git config user.email
```

If `git` is not found, install Git first. If either `git config` line prints nothing, set it:

```
$ git config --global user.name "Jordan Bell"
$ git config --global user.email jordan@example.com
```

5. Sign in:

```
$ ws login
Opening https://outport.app in your browser to sign in.
```

Sign in with your Outport staff account in the browser. `ws` then prints the name you are signed in as, and stays signed in.
When the browser is on another machine, such as `ws` running on a server or over SSH, sign in with a code instead. A coding agent signs in this way too, because it cannot open a browser:

```
$ ws login --device
On another device, open https://outport.app/oauth/device and enter 55NB-64GH.
Signed in as Jordan Bell.
```

Open that address in any browser, sign in with your Outport staff account and type the code. Each sign-in prints a new code.

6. Check the environment:

```
$ ws profile show
Environment: production (https://api.websitestrategy.com/api)
Profile: production
API: https://api.websitestrategy.com/api
```

An installed `ws` uses `production`, where real customers' sites live, and every step in this Quick Start runs there.

2. Make a site

Run `ws new` with the business's name:

```
$ ws new "Saltmere Marina"
Made the site "Saltmere Marina".
  Subdomain:     saltmere-marina
  Language:      English (en)
  Time zone:     America/Halifax
  Folder:        ./saltmere-marina (a Git clone, 6 files)

Next: cd saltmere-marina && ws preview
```

`ws new` gives the site two names:

- **The display name,** "Saltmere Marina", is the name people read. Pages print it with `{{ site.name }}`.
- **The subdomain,** `saltmere-marina`, starts the site's web addresses on our preview domain, `sitepeek.page`. It uses lowercase letters, numbers and single hyphens, and no two sites share one. To choose it yourself, add `--subdomain saltmere`.

The language and time zone come from your computer. Set them to the business's own: for a business in Newfoundland whose website is in English and French, run `ws new "Name" --languages en,fr --time-zone America/St_Johns`. `ws` does not look them up.

The folder is a Git clone of the site's repository on the platform, with one commit that holds the starter files:

```
$ cd saltmere-marina
$ ls -A
.git  .gitignore  .ws  AGENTS.md  _includes  _layouts  index.html  styles.css
```

`index.html` is the home page, at `/`. Each page is an HTML file named for its address: `about.html` is `/about`. `_layouts/default.html` is the HTML around every page, and `_includes/` holds the header and footer every page shares. Leave `.ws/` alone: `ws` keeps the site's ID there. Make a site and preview it walks through each file.

If this site is only a test, make its sample link end one day after its last publish. Run this in the site's folder:

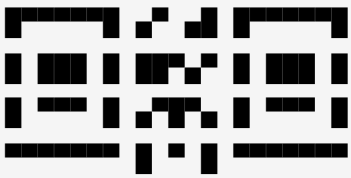
```
$ ws site --expires 1d
The sample's private link now lasts 1 day from each publish.
Saved settings for "Saltmere Marina".
```

When that link ends, the platform deletes the site, because nobody bought it.

3. Open it on your phone

```
$ ws preview
Built 1 page.

Private preview, open to anyone with this link until November 20:
https://saltmere-marina--k7f3.sitepeek.page/?key=Hn4pX7rKq2Lw9TzB3mVc8d
```



```
Scan the code with your phone's camera to open the preview.
```

Scan the code with your phone's camera, or open the link in any browser. The code prints only when `ws` writes to a terminal window; the link is always there. Each `ws preview` keeps the same link, open for 7 more days. The key at the end of the link gives that browser access to this one preview, then is removed from the address. A browser that never opened the link with its key sees only "This preview is private".

A preview is never published, and it is no backup: no command brings files back from it.

4. Edit the site

1. Leave the preview watching the folder, so every open page updates as you save:

```
$ ws preview --watch
Built 1 page.
Private preview: https://saltmere-marina--k7f3.sitepeek.page/?
key=Hn4pX7rKq2Lw9TzB3mVc8d
Watching this folder. Press Ctrl+C to stop.
```

2. In another terminal window, open `index.html`. This is the whole page:

   ```
   ---
   layout: default
   title: Home
   ---
   <h1>{{ site.name }}</h1>
   <p>Welcome. This page is index.html.</p>
   ```

   The lines between the `---` lines are the page's settings. The tags in curly braces are Liquid, the template language Shopify made.
3. Change the welcome sentence and save the file. The page on your phone reloads itself at the same scroll position:

   ```
   10:14:20  index.html changed. Rebuilt 1 page and reloaded it on 2 open pages.
   ```

   A changed style sheet is applied to the open pages without a reload.

If a page has a mistake, the platform names the file, the line and a fix, and the preview keeps the last build that worked:

```
index.html line 5: there is no "site.titel". Did you mean "site.name"?
Nothing changed: the preview still shows the last build that worked.
```

Press Ctrl+C to stop watching. Add pages and a shared header adds a page and a menu link, and Add news, live weather and a form adds news posts, live weather and a form.

5. Publish with ws push

Publishing is a Git commit and a `ws push`:

```
$ git add -A && git commit -m "First version"
$ ws push
Built 1 page.
Published version 1 of "Saltmere Marina".

Private link to the sample, open to anyone with it until December 14:
https://saltmere-marina-k7f3q9.sitepeek.page/?key=Vq3x9Lm2Tz8RkP4wNc7bYd

Automatic checks are running. Run ws status for the result.
```

While it works, `ws push` also prints progress lines, such as `Building the site.`, which this example leaves out. Each publish makes a numbered **version**: a saved copy of every file in the commit and of the site's settings. `ws versions` lists them, with each commit message as the version's note.

`ws push` publishes only committed work. It refuses while a file has changes you have not committed, and says `Commit your changes first: git add -A && git commit`. Run `git status --short` to see what is not committed yet. Until you push, your commits are only in your folder, so push when you stop for the day if the work is ready for the people who have the link.

The automatic checks test page speed, accessibility and links, plus a test entry through each form and missing translations on a site with more than one language. A sample goes live first and is checked in the background, which takes a few minutes. Run `ws status` until it shows the result:

```
$ ws status
"Saltmere Marina" (saltmere-marina)
This clone includes published version 1.
No unpublished commits.
No uncommitted changes.
Version 1 passed checks: speed, accessibility, links.
```

While the checks run, its last line says `Automatic checks for version 1 are running.`

If a check fails, the sample stays live and the result names the problem. Fix it and push again.

If a newer version was published that your folder does not have yet, from another computer or the owner panel, the push is refused with `Not published: someone published first. Run git pull,`

`then ws push.` Publish with Git and work with other builders covers a second computer and two people changing one file.

6. Share the sample link

The sales team sends a sample's private link to the business.

1. Copy the link that `ws push` printed. Later publishes keep the same link, and `ws site` shows it again.
2. Give it to the sales team. Anyone with the link can open the sample, so send it only to the people it is for.

The link lasts 30 days from the last publish. When it ends, the link stops working and the unsold sample is deleted with all its versions. To keep a sample longer:

```
$ ws site --expires 60d
```

When the business buys its site, staff mark it bought on the ops panel, the staff website at `ops.websitestrategy.com`, and attach the business's own domain. From then on, `ws push` prints that address, and the checks must pass before each new version goes live.

7. Answer an owner's request

A sample has no owner yet, so on your first site `ws requests` prints `No open requests for "Saltmere Marina".` Once a business buys its site, its **owner** manages it in the owner panel at `app.websitestrategy.com`. To ask for a change, the owner opens **Change**, points at part of a page and types what should change. That makes a numbered **request**, and the site's builders get an email.

1. List the open requests, inside the site's folder:

   ```
   $ ws requests
   Open requests for "Saltmere Marina":
     1  Dana MacLeod, 10:02 AM   "Diesel is $1.89 a litre now."
   ```

2. Show the request. It names the file and line the owner pointed at:

```
$ ws requests 1
Request 1 from Dana MacLeod (owner), 10:02 AM, on version 3:
  "Diesel is $1.89 a litre now."

Dana selected this sentence on saltmeremarina.ca/news/fuel-dock:
  "Diesel is $1.79 a litre."
It comes from _data/news/2026-11-02-fuel-dock.md, line 6.
Its French copy: _data/fr/news/2026-11-02-fuel-dock.md, line 5.
Rebuilt from what Dana's browser showed: https://ops.websitestrategy.com/sites/s-kqm7qrxztcwxrrpdnfvbcebszd/requests/1/picture.png
```

   The picture matches what the owner saw.
3. Bring the folder up to date. Edits the owner makes in the owner panel arrive as commits the platform makes for the owner, and `ws push` refuses until you have them:

   ```
   $ git pull
   ```

4. Make the change and check it with `ws preview`.
5. Commit it, and push it as the answer to that request:

   ```
   $ git add -A && git commit -m "diesel price"
   $ ws push --request 1
   Built 10 pages.
   Checks passed: speed, accessibility, links, forms, translations.
   Published version 4 of "Saltmere Marina" at https://saltmeremarina.ca.
   Dana MacLeod gets an email: request 1 is done.
   ```

The change goes live as soon as the checks pass, with no approval step. To undo a version, `ws undo N` reverses only what version N changed. To show the owner a change before visitors see it, push it with `ws push --needs-approval`. Answer owner requests and ask for approval covers both.

Where to look next

The Builder Guide has a part for each next step: more pages and a shared header, news, live weather and a form, a second language, working with other builders, and undo, rollback and approval. Look up any command or rule in the Reference.