Deploy the docs site

This runbook builds a small static site, checks it, and ships it as a new release behind a `current` symlink, the way many hosts serve static sites. "Production" is a scratch directory under `$TMPDIR`, so running it changes nothing on your machine. Swap the `build` and `ship` blocks for your own commands to use it for real.

The run stops at the first failing step (`on_error: stop`), and shipping asks first (`confirm`). `mdsh --yes` answers yes for you, as CI would.

Build

```
sh {#build}
```

```
export WORK="$(mktemp -d)"
export SITE="$WORK/public" PROD="$WORK/prod"
export RELEASE="2026-01-02.1"
mkdir -p "$SITE/guide" "$PROD/releases"
page() {
  printf '<!doctype html>\n<title>%s</title>\n<h1>%s</h1>\n%s\n' "$2" "$2" "$3" > "$SI
TE/$1"
}
page index.html "Home" '<a href="guide/">Guide</a> <a href="about.html">About</a>'
page about.html "About" '<a href="index.html">Home</a>'
page guide/index.html "Guide" '<a href="install.html">Install</a> <a href="../index.ht
ml">Home</a>'
page guide/install.html "Install" '<a href="index.html">Guide</a>'
PAGES="$(find "$SITE" -name '*.html' | wc -l | tr -d ' ')"
echo "built $PAGES pages for release $RELEASE"
```

```
built 4 pages for release 2026-01-02.1
```

Check

Two checks run side by side. The link checker is a `background` block: it starts, and the run goes straight on to the size check below while it works.

```
sh {#links background timeout=30s}
```

```
cd "$SITE"
broken=0
for f in $(find . -name '*.html' | sort); do
  dir="$(dirname "$f")"
  for href in $(grep -o 'href="[^"]*"' "$f" | sed 's/^href="//; s/"$//'); do
    target="$dir/$href"
    case "$href" in */) target="${target}index.html" ;; esac
    if [ ! -e "$target" ]; then
      echo "broken: $f -> $href"
      broken=$((broken + 1))
    fi
  done
done
echo "$broken broken links"
test "$broken" -eq 0
```

```
0 broken links
```

```sh {#size}
SIZE_KB="$(du -sk "$SITE" | cut -f1)"
echo "site is ${SIZE_KB} KiB"
test "$SIZE_KB" -lt 10240
```

```
site is 24 KiB
```

Before shipping, wait for the link checker. `wait` gives its exit status, so broken links stop the run here.

```sh {#links-done}
wait "$MDSH_BG_links" && echo "links ok"
```

```
links ok
```

Ship

Copy the build into a new release directory, then point the `current` symlink at it, so visitors never see a half-copied site.

```sh {#ship confirm="Ship release to production?"}
cp -R "$SITE" "$PROD/releases/$RELEASE"
ln -sfn "releases/$RELEASE" "$PROD/current"
echo "current -> $(readlink "$PROD/current")"
```

```
current -> releases/2026-01-02.1
```

Verify

What production serves must match what was built.

```sh {#verify}
diff -r "$SITE" "$PROD/current/" && echo "production matches the build"
```

```
production matches the build
```

Summary

```sh {#summary out=markdown}
{
  printf 'step\tresult\n'
  printf 'release\t%s\n' "$RELEASE"
  printf 'pages\t%s\n' "$PAGES"
  printf 'size\t%s KiB\n' "$SIZE_KB"
  printf 'links\tok\n'
  printf 'verified\tyes\n'
} | md_table
```

step	result

release	`2026-01-02.1`
pages	4
size	24 KiB
links	ok
verified	yes

Shipped release **`2026-01-02.1`**.

Roll back

Not part of a normal run, so this block is marked `skip`: it is here to copy into a shell when a release goes wrong.

```sh {#rollback skip}
ln -sfn "releases/$PREVIOUS" "$PROD/current"
```

Clean up