

Web-based Shell

- [Web based SSH Instant Access](#)
- [Customization](#)

Web based SSH Instant Access

This is a browser-based SSH terminal service (`webshell.drheck.dev`) authenticated via Authentik Single Sign-On (SSO) and short-lived OpenSSH certificates signed by Smallstep (`step-ca`), running under rootless Podman Quadlets (`docker-data`, `UID 1100`).

1. Architectural Overview

```
[ Browser Client ]
  |
  ▼ (HTTPS / WebSocket)
[ Nginx Reverse Proxy (/etc/nginx/sites-available/webshell) ]
  |
  └─ auth_request → [ Authentik Embedded Outpost ] (SSO Auth & Claims)
  |
  ▼ (HTTP / WebSocket + X-Authentik-Username Header)
[ Webshell Rootless Container (ttyd + login-wrapper.sh) on eth77 fd77:....:771d:7681 ]
  |
  └─ Step SSH Certificate Sign → [ Step-CA Container on webshell_net (step-ca:9000) ]
  |
  ▼ (SSH with Ephemeral Ed25519 Certificate over IPv6)
[ Host ul OpenSSH Daemon on [fd77:7777:5010:0:1337:f1ac:d00d:337]:22 ]
```

Key Components

1. `step-ca`: Smallstep Certificate Authority issuing 5-minute ephemeral Ed25519 user certificates (`step-ca.container` on `webshell.network`).
2. `webshell`: Debian Bookworm container running static `ttyd` 1.7.4, `step` CLI 0.25.0, and `login-wrapper.sh` (`webshell.container` on `webshell.network`).
3. **Authentik**: Provides OIDC integration for `step-ca` and ForwardAuth single sign-on (`webshell.drheck.dev`).

4. **Nginx**: Terminates TLS and proxies HTTP/WebSocket connections while passing authenticated identity headers (`X-Authentik-Username`).
5. **OpenSSH** (`sshd`): Configured to trust `step-ca` signed user certificates for passwordless OS login on host `u1`.

2. Network & ULA IPv6 Allocation

Every web-facing service on host `u1` binds to a unique ULA IPv6 address on interface `eth77` (`fd77:7777:5010:0:1337:f1ac:d00d:.../64`).

- **step-ca IPv6**: `fd77:7777:5010:0:1337:f1ac:d00d:ca` (Port `9000`, also used locally by `/data/minecraft/ipctl`)
- **webshell IPv6**: `fd77:7777:5010:0:1337:f1ac:d00d:771d` (Port `7681`)
- **Private Bridge Network**: `webshell.network` (`webshell_net`, serving `step-ca` container DNS via `aardvark-dns`)
- **Host `u1` ULA IPv6 SSH Target**: `fd77:7777:5010:0:1337:f1ac:d00d:337` (Port `22`)

3. Step 1: Rootless Podman Quadlet Configuration (`step-ca` & `webshell`)

A. Quadlet Unit Definitions (`/data/docker/webshell/`)

Both Quadlets are symlinked into `/home/docker-data/.config/containers/systemd/` and managed by `user@1100.service`:

`/data/docker/webshell/step-ca.container`:

```
[Unit]
Description=Smallstep SSH & X.509 CA (Rootless Podman)
After=network-online.target
```

Wants=network-online.target

[Container]

Pull=never

ContainerName=step-ca

Image=docker.io/smallstep/step-ca:latest

Network=webshell.network

AddHost=auth.drheck.dev:169.254.1.2

PublishPort=[fd77:7777:5010:0:1337:f1ac:d00d:ca]:9000:9000

EnvironmentFile=/data/docker/webshell/.env

Volume=/data/docker/webshell/step-ca:/home/step:rw

[Service]

Restart=always

RestartSec=5

[Install]

WantedBy=default.target

/data/docker/webshell/webshell.container :

[Unit]

Description=Webshell Browser SSH Terminal (Rootless Podman)

After=network-online.target step-ca.service

Requires=step-ca.service

[Container]

Pull=never

ContainerName=webshell

Image=localhost/webshell:latest

Network=webshell.network

PublishPort=[fd77:7777:5010:0:1337:f1ac:d00d:771d]:7681:7681

Environment=STEP_CA_URL=https://step-ca:9000

Volume=/data/docker/webshell/login-wrapper.sh:/usr/local/bin/login-wrapper.sh:ro

Volume=/data/docker/webshell/known_hosts:/etc/ssh/ssh_known_hosts:ro

Volume=/etc/passwd:/etc/passwd:ro

Volume=/data/docker/webshell/step-ca/certs/root_ca.crt:/etc/step-ca/root_ca.crt:ro

Volume=/data/docker/webshell/step-ca/secrets/webshell_svc_pwd:/etc/step-ca/webshell_svc_pwd:ro

Volume=/data/docker/webshell/index.html:/etc/ttyd/index.html:ro

Exec=-W -H X-Authentik-Username -t theme={"background":"#000000"} -I /etc/ttyd/index.html

```
/usr/local/bin/login-wrapper.sh
```

```
[Service]
```

```
Restart=always
```

```
RestartSec=5
```

```
[Install]
```

```
WantedBy=default.target
```

B. CA Directory Permissions (Rootless User Namespace)

Inside `docker-data`'s rootless user namespace (`docker-data:624288:65536`), the `step` user inside `smallstep/step-ca` maps cleanly to an isolated unprivileged subordinate UID on the host with zero world permissions (`o-rwx`).

4. Step 2: Host `ul` OpenSSH Daemon Trust & Web Session Banner Setup

To allow `sshd` on host `ul` (`[fd77:7777:5010:0:1337:f1ac:d00d:337]:22`) to accept certificates signed by `step-ca`, enforce strict `known_hosts` checking, and serve a dedicated login banner exclusively for web sessions:

1. Extract the CA user public key:

```
(cd /tmp && sudo -u docker-data HOME=/home/docker-data XDG_RUNTIME_DIR=/run/user/1100  
podman exec step-ca step certificate inspect --format json  
/home/step/certs/user_ca_key.pub)
```

2. Append the public key (`ecdsa-sha2-nistp256 ...` or `ssh-ed25519 ...`) to `/etc/ssh/drheck_ca_user_key.pub`.
3. Verify that `/etc/ssh/sshd_config.d/10-drheck-certs.conf` contains:

```
TrustedUserCAKeys /etc/ssh/drheck_ca_user_key.pub
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub

Match Address fd77:7777:5010:771d::/64
    Banner /etc/ssh/webshell_banner.txt
```

4. Web Session Banner & Rotation Script: `/etc/profile.d/webshell_banner.sh` executes when a web session connects. It randomly selects an ANSI art file from `/data/nfo/ansi_list.txt` (42,000+ files), renders it via `/usr/local/bin/ansview --wide-dupe` across terminal width, right-aligns the file path, and exports `$LAST_ANSI` for the `/usr/local/bin/bad_ansi` removal command.

The static fallback banner is stored at `/etc/ssh/webshell_banner.txt`.

5. Generate `/data/docker/webshell/known_hosts` from host `ul`'s public host key:

```
echo "fd77:7777:5010:0:1337:f1ac:d00d:337 $(cat /etc/ssh/ssh_host_ed25519_key.pub)" >
/data/docker/webshell/known_hosts
chmod 644 /data/docker/webshell/known_hosts
```

6. Reload OpenSSH daemon:

```
sudo systemctl reload sshd
```

5. Step 3: Web Terminal Backend (`ttyd` + `login-wrapper.sh`) Setup

A. Dockerfile (`/data/docker/webshell/Dockerfile`)

Rootless Podman `ttyd` Listener Note: Do **NOT** pass `-6 -i ::` under rootless Podman. When rootless Podman forwards traffic from host `[fd77:...:771d]:7681` into `webshell.network`, the forwarded socket arrives over container IPv4 (`10.89.X.X:7681`). Omitting `-6 -i ::` allows `ttyd` to bind all interfaces inside the container cleanly instead of rejecting IPv4 forwards with `502 Bad Gateway`.

```

FROM debian:bookworm-slim

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y --no-install-recommends \
    ca-certificates curl openssh-client bash procps libnss-sss \
    && sed -i 's/passwd: files/passwd: files sss/' /etc/nsswitch.conf \
    && rm -rf /var/lib/apt/lists/*

# Install static tttyd 1.7.4 & step CLI 0.25.0
RUN curl -sSL https://github.com/tsl0922/ttyd/releases/download/1.7.4/ttyd.x86_64 -o
/usr/local/bin/ttyd \
    && chmod +x /usr/local/bin/ttyd \
    && curl -sSL https://dl.step.sm/gh-release/cli/docs-step-
cli/v0.25.0/step_linux_0.25.0_amd64.tar.gz | tar -xz -C /tmp \
    && mv /tmp/step_0.25.0/bin/step /usr/local/bin/step \
    && chmod +x /usr/local/bin/step \
    && rm -rf /tmp/step_0.25.0

COPY login-wrapper.sh /usr/local/bin/login-wrapper.sh
RUN chmod +x /usr/local/bin/login-wrapper.sh

EXPOSE 7681
ENTRYPOINT ["/usr/local/bin/ttyd"]
CMD ["-W", "-H", "X-Authentik-Username", "-t", "theme={\"background\": \"\#000000\"}", "-I",
"/etc/ttyd/index.html", "/usr/local/bin/login-wrapper.sh"]

```

B. Backend Login Wrapper (

`/data/docker/webshell/login-wrapper.sh)`

```

#!/usr/bin/env bash
set -euo pipefail

# Resolve username across CGI-style environment variables exported by ttyd/proxy
TARGET_USER="${X_AUTHENTIK_USERNAME:-${TTYD_USER:-${HTTP_X_AUTHENTIK_USERNAME:-
${AUTHENTIK_USERNAME:-}}}}}"

if [[ -z "${TARGET_USER}" ]]; then
    echo "Authentication Error: Missing Authentik identity." >&2

```

```

    exit 1
fi

# Strict alphanumeric regex check
if [[ ! "${TARGET_USER}" =~ ^[a-z_][a-z0-9_-]{0,31}$ ]]; then
    echo "Authentication Error: Invalid username format." >&2
    exit 1
fi

# Retrieve numeric UID from host passwd database
TARGET_UID=$(awk -F: -v u="${TARGET_USER}" ' $1 == u {print $3}' /etc/passwd 2>/dev/null ||
true)
if [[ -z "${TARGET_UID}" ]]; then
    echo "Authentication Error: User does not exist on target host." >&2
    exit 1
fi

# Reject system accounts (UID < 1000), admin, and docker* accounts
if (( TARGET_UID < 1000 )) || [[ "${TARGET_USER}" == "admin" ]] || [[ "${TARGET_USER}" ==
docker* ]]; then
    echo "Authentication Error: Access to privileged or system account is forbidden." >&2
    exit 1
fi

KEY_DIR="$(mktemp -d -t ssh_XXXXXX)"
KEY_PATH="${KEY_DIR}/id_ed25519"

cleanup() {
    rm -rf "${KEY_DIR}"
}

trap cleanup EXIT INT TERM

# Generate ephemeral Ed25519 keypair silently
ssh-keygen -q -t ed25519 -f "${KEY_PATH}" -N "" >/dev/null 2>&1

# Sign user certificate via JWK provisioner (--sign flag is mandatory for non-interactive
execution)
if command -v step >/dev/null 2>&1 && [[ -n "${STEP_CA_URL:-}" ]] && [[ -f "/etc/step-
ca/webshell_svc_pwd" ]]; then
    step ssh certificate "${TARGET_USER}" "${KEY_PATH}.pub" --sign --

```

```

provisioner="webshell-svc"      --provisioner-password-file="/etc/step-
ca/webshell_svc_pwd"          --ca-url="${STEP_CA_URL}"      --root="/etc/step-ca/root_ca.crt"
--not-after=5m                --force >/dev/null 2>&1 || true
fi

# Login passwordless to host ul SSH daemon over local ULA IPv6 with strict known_hosts
checking
exec ssh -o StrictHostKeyChecking=yes          -i "${KEY_PATH}"
"${TARGET_USER}@fd77:7777:5010:0:1337:f1ac:d00d:337" "$@"

```

6. Step 4: Authentik Single Sign-On Provisioning (ak shell)

To configure Authentik ForwardAuth without encountering `500 Internal Server Error`, `Redirect URI Error`, or `ERR_TOO_MANY_REDIRECTS`, execute the following Python provisioning script in `authentik-server`:

```

(cd /tmp && sudo -u docker-data HOME=/home/docker-data XDG_RUNTIME_DIR=/run/user/1100 podman
exec -i authentik-server ak shell) << 'EOF'
from authentik.core.models import Application
from authentik.providers.proxy.models import ProxyProvider, ProxyMode
from authentik.providers.oauth2.models import RedirectURI, RedirectURIMatchingMode,
RedirectURIType
from authentik.outposts.models import Outpost

SERVICE_NAME = "webshell"
EXTERNAL_HOST = f"https://{SERVICE_NAME}.drheck.dev"

# 1. Inherit default scope property mappings from a working provider template
template_provider = ProxyProvider.objects.get(name="cockpit")

# 2. Create ProxyProvider with complete OAuth, redirect URI, and cookie settings
provider, _ = ProxyProvider.objects.get_or_create(
    name=SERVICE_NAME,
    defaults={
        "external_host": EXTERNAL_HOST,

```

```

        "mode": ProxyMode.FORWARD_SINGLE,
        "authorization_flow_id": "f9ad5326-f668-4d1f-861d-43b47f0a002e",
        "cookie_domain": "drheck.dev",
        "grant_types": ["authorization_code", "client_credentials", "password"],
        "redirect_uris": [
            RedirectURI(
                matching_mode=RedirectURIMatchingMode.STRICT,
                url=f"{EXTERNAL_HOST}/outpost.goauthentik.io/callback?X-authentik-auth-
callback=true",
                redirect_uri_type=RedirectURIType.AUTHORIZATION
            ),
            RedirectURI(
                matching_mode=RedirectURIMatchingMode.STRICT,
                url=f"{EXTERNAL_HOST}?X-authentik-auth-callback=true",
                redirect_uri_type=RedirectURIType.AUTHORIZATION
            ),
        ]
    }
)

```

```

provider.property_mappings.set(template_provider.property_mappings.all())
provider.save()

```

3. Create Application

```

app, _ = Application.objects.get_or_create(
    slug=SERVICE_NAME,
    defaults={
        "name": SERVICE_NAME.capitalize(),
        "meta_launch_url": f"{EXTERNAL_HOST}/",
        "open_in_new_tab": True,
        "provider": provider,
    }
)

```

4. Mandatory: Assign provider to authentik Embedded Outpost

```

outpost = Outpost.objects.get(name="authentik Embedded Outpost")
if provider not in outpost.providers.all():
    outpost.providers.add(provider)
outpost.save()

```

```
print(f"Successfully provisioned {SERVICE_NAME} and attached to Embedded Outpost.")
EOF
```

7. Step 5: Nginx Reverse Proxy Configuration (/etc/nginx/sites-available/webshell)

```
map $http_upgrade $proxy_target_connection_webshell {
    default upgrade;
    ""        "";
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    http2 on;
    server_name webshell.drheck.dev;

    ssl_certificate      /etc/nginx/certs/fullchain.pem;
    ssl_certificate_key  /etc/nginx/certs/privkey.pem;

    client_max_body_size 64M;

    # Authentik ForwardAuth Subrequest Configuration
    auth_request          /outpost.goauthentik.io/auth/nginx;
    error_page            401 = @goauthentik_proxy_signin;
    auth_request_set $auth_cookie $upstream_http_set_cookie;
    add_header            Set-Cookie $auth_cookie;
    auth_request_set $authentik_username $upstream_http_x_authentik_username;
    auth_request_set $authentik_email $upstream_http_x_authentik_email;

    # Per-User Custom Fonts Route
    location /user-fonts/ {
        auth_request       /outpost.goauthentik.io/auth/nginx;
```

```

auth_request_set $authentik_username $upstream_http_x_authentik_username;
alias /home/$authentik_username/.config/webshell/fonts/;
types {
    font/woff2 woff2;
    font/ttf ttf;
    font/otf otf;
}
add_header Cache-Control "public, max-age=86400";
}

location / {
    proxy_pass http://[fd77:7777:5010:0:1337:f1ac:d00d:771d]:7681;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $proxy_target_connection_webshell;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Authentik-Username $authentik_username;
    proxy_set_header X-Authentik-Email $authentik_email;
}

location /outpost.goauthentik.io {
    proxy_pass https://auth.drheck.dev/outpost.goauthentik.io;
    proxy_set_header Host auth.drheck.dev;
    proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
    proxy_set_header X-Forwarded-Host $http_host;
    proxy_set_header X-Forwarded-Proto $scheme;
    add_header Set-Cookie $auth_cookie;
    auth_request_set $auth_cookie $upstream_http_set_cookie;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
}

location @goauthentik_proxy_signin {
    internal;
    add_header Set-Cookie $auth_cookie;
    return 302 /outpost.goauthentik.io/start?rd=$scheme://$http_host$request_uri;
}

```

8. Step 6: Backup & Restore Scripts

Every service on host `ul` uses `shellcheck`-verified backup and restore scripts located in `/home/luke/git/backups/ul/scripts/` with 10% `par2` redundancy:

- `/home/luke/git/backups/ul/scripts/backup/webshell`: Archives `/data/docker/webshell/` (`.env`, `step-ca.container`, `webshell.container`, `webshell.network`, `step-ca/`) to `webshell_data.tar.xz`, and system configuration/scripts (`/etc/ssh/webshell_banner.txt`, `/etc/ssh/drheck_ca_user_key.pub`, `/etc/ssh/sshd_config.d/10-drheck-certs.conf`, `/etc/nginx/sites-available/webshell`, `/etc/profile.d/webshell_banner.sh`, `/usr/local/bin/bad_ansi`, `/data/nfo/ansi_list.txt`) to `webshell_system_config.tar.xz`, generating `.par2` recovery files via `par2 create -r10`.
- `/home/luke/git/backups/ul/scripts/restore/webshell`: Verifies archives with `par2 verify`, restores configurations into `/data/docker/webshell/` and `/`, and restarts `step-ca.service` and `webshell.service`.

9. Initial Assumptions & Attempts vs. Actual Working Configuration

Technical Area	Initial Assumption / Attempt	Actual Working Configuration & Root Cause	Prevention Standard
Outpost Assignment (<code>500 Internal Server Error</code>)	Creating <code>ProxyProvider</code> in <code>Authentik</code> automatically activates <code>ForwardAuth</code> routing.	Creating a provider does not assign it to an Outpost. Unassigned hostnames return <code>404 Not Found</code> from <code>/outpost.goauthentik.io/auth/nginx</code> , which Nginx converts to <code>500 Internal Server Error</code> .	Explicitly attach providers to <code>authentik Embedded Outpost</code> (<code>outpost.providers.add(provider)</code>) before enabling Nginx traffic.
Redirect URIs (<code>Redirect URI Error</code>)	Instantiating <code>ProxyProvider</code> sets default authorization callback URLs.	The Django ORM defaults <code>redirect_uris</code> to <code>[]</code> . <code>Authentik</code> rejects callback URIs against an empty database list.	Explicitly define <code>RedirectURI</code> o for both <code>/outpost.goauthentik.io/callback</code> root <code>/</code> in <code>ProxyProvider.redirect_uris</code> .

Technical Area	Initial Assumption / Attempt	Actual Working Configuration & Root Cause	Prevention Standard
Cookie Domain & Scopes (<code>ERR_TOO_MANY_REDIRECTS</code>)	Providers operate without defining <code>cookie_domain</code> , <code>grant_types</code> , or <code>property_mappings</code> .	Without <code>property_mappings</code> , the OAuth request omits critical scopes (<code>ak_proxy</code>). Without explicit <code>cookie_domain='drheck.dev'</code> , Authentik refuses to grant ForwardAuth proxy tokens, causing an endless redirect loop.	Clone <code>property_mappings</code> from <code>cockpit</code> and explicitly set <code>cookie_domain='drheck.dev'</code> and <code>grant_types</code> .
CGI Env Mapping (Missing Authentik identity)	<code>ttyd -H X-Authentik-Username</code> exports <code>X_AUTHENTIK_USERNAME</code> directly to child scripts.	<code>ttyd</code> transforms HTTP headers into standard CGI environment variables (<code>HTTP_X_AUTHENTIK_USERNAME</code>).	Always check CGI-style fallback variations: <code>TARGET_USER="\${X_AUTHENTIK_USERNAME:-\${TTYD_USER:-\${HTTP_X_AUTHENTIK_USERNAME:-\${AUTHENTIK_USERNAME:-}}}}"</code> .
Rootless Port Forwarding vs. <code>-6 -i ::</code> (<code>502 Bad Gateway</code>)	Keeping legacy Docker <code>"-6", "-i", "::-"</code> flags works identically in rootless Podman.	Under rootless Podman, <code>PublishPort=[fd77:::771d]:7681:7681</code> forwards traffic into <code>webshell_net</code> over IPv4 (<code>10.89.X.X:7681</code>). Passing <code>-6 -i ::</code> restricts <code>ttyd</code> to IPv6-only and rejects forwarded connections.	Omit <code>-6 -i ::</code> in <code>webshell.container</code> so <code>ttyd</code> listens on all interfaces inside the container namespace.
Container Script Updates (old script execution)	Modifying <code>login-wrapper.sh</code> on host <code>ul</code> and restarting runs the updated script only if mounted.	Restarting only re-reads bind-mounted files, not <code>COPY</code> ed image layers.	Bind-mount operational scripts <code>Volume=/data/docker/webshell/login-wrapper.sh:/usr/local/bin/login-wrapper.sh:ro</code> in <code>webshell.container</code>
CA Secret Privilege Restrictions	Administrative or personal user accounts own the CA keys and JWK provisioner secret.	Standard unprivileged or administrative (<code>wheel</code>) user accounts should not have read access to CA private keys or provisioner passwords.	Isolate <code>/data/docker/webshell/login-wrapper.sh</code> inside <code>docker-data</code> 's rootless subordinate UID namespace with <code>rwX</code> permissions.
SSH Client Host Warnings (Warning: Permanently added...)	Invoking <code>ssh</code> with <code>/dev/null</code> as <code>known_hosts</code> prints key addition warnings on every login.	Connecting without a populated <code>known_hosts</code> file causes OpenSSH to log a warning to stderr whenever an ephemeral host entry is added.	Mount a static <code>/etc/ssh/ssh_known_hosts</code> file generated from <code>/etc/ssh/ssh_host_ed25519_key.pub</code> and enforce <code>StrictHostKeyChecking=yes</code> .
Cockpit Login MOTD Banner (Web console: https://ul...:9090)	Cockpit's MOTD issue updater (<code>update-issue</code>) injects <code>:9090</code> URLs into <code>/etc/motd.d/</code> .	Cockpit's <code>cockpit-issue.service</code> inspects local interfaces and writes direct port URLs to <code>/run/cockpit/issue</code> , which <code>/etc/motd.d/cockpit</code> symlinks into login greetings.	Remove <code>/etc/motd.d/cockpit</code> and <code>/etc/issue.d/cockpit.issue</code> symlinks and mask <code>cockpit-issue.service</code> with <code>systemctl mask cockpit-issue.service</code> .
Dynamic Host UID Verification	Hardcoding system username lists (<code>root</code>	<code>admin</code>	<code>...`)</code> in wrapper scripts to block accounts.

Technical Area	Initial Assumption / Attempt	Actual Working Configuration & Root Cause	Prevention Standard
Web-Session-Only Login Banner	Printing greetings via script <code>echo</code> or editing server-wide <code>/etc/motd</code> .	Script <code>echo</code> output is lost before terminal attachment, while <code>/etc/motd</code> changes affect standard SSH logins.	Configure <code>Match Address</code> <code>fd77:7777:5010:771d::/64</code> in <code>/etc/ssh/sshd_config.d/10-drehel</code> <code>certs.conf</code> with <code>Banner</code> <code>/etc/ssh/webshell_banner.txt</code> .

Customization

Webshell Configuration Guide

Per-user web terminal preferences are stored in `~/.config/webshell/`.

Directory Structure

```
~/.config/webshell/  
├─ config.json      # Theme, font size, and cursor settings  
├─ fonts/           # Dir for local fonts  
  └─ font.woff2     # (Optional) Local custom font file (.woff2, .ttf, or .otf)
```

`config.json` Options

Key	Type	Default	Description
<code>fontFamily</code>	String	<code>'JetBrains Mono', monospace</code>	Font family stack
<code>fontSize</code>	Number	<code>15</code>	Font size in pixels
<code>cursorStyle</code>	String	<code>"block"</code>	Cursor shape (<code>"block"</code> , <code>"underline"</code> , or <code>"bar"</code>)
<code>cursorBlink</code>	Boolean	<code>true</code>	Enable or disable cursor blinking
<code>theme</code>	Object	<i>See below</i>	Color palette overrides

Theme Options (`theme` object)

Key	Description
<code>background</code>	Terminal background color (e.g. <code>"#000000"</code>)
<code>foreground</code>	Default text color (e.g. <code>"#e0e0e0"</code>)
<code>cursor</code>	Cursor color (e.g. <code>"#ffffff"</code>)
<code>black</code> - <code>white</code>	Standard 8 ANSI colors
<code>brightBlack</code> - <code>brightWhite</code>	High-intensity 8 ANSI colors

Example

`~/.config/webshell/config.json`

```
{
  "fontFamily": "'JetBrains Mono', 'Fira Code', monospace",
  "fontSize": 15,
  "cursorStyle": "block",
  "cursorBlink": true,
  "theme": {
    "background": "#000000",
    "foreground": "#e0e0e0",
    "cursor": "#00ff00",
    "green": "#00ff00",
    "cyan": "#00ffff"
  }
}
```

Using Local Font Files

Location

Place your font file in `~/.config/webshell/` as one of the following:

- `font.woff2` (Create your own if needed! See below)

- `font.ttf`
- `font.otf`

The webshell will automatically encode and load your font file into the web browser upon login.

Determining Font Family Name from a Font File

fc-scan will return the font family name for you. It is possible to return multiple aliases so you can use the first one to simplify the process.

```
$ fc-scan GoogleSansCodeNerdFontMono-SemiBoldItalic.ttf --format "%{family}\n" | cut -d, -f1
GoogleSansCode Nerd Font Mono
```

Use woff2 files instead of ttf or otf

These are normal font files, but compressed so it will be nicer for web use:

```
# one at a time
for f in *.ttf *.otf; do woff2_compress "$f"; done
# use all those cores!
parallel woff2_compress ::: *.[ot]tf
```

On 181 of my fonts the size dropped from 381M to 220M.