solidontrixie v5

tags: solid trixie root strasbourg

Solid Servers on Debian Trixie — Full Stack Setup v5

Stack: NSS · CSS · Solid Pivot · JSS · Caddy (xcaddy + DNS-01, mandatory) · PM2 · Netdata · Uptime Kuma

What Changed in v5

All four Solid servers now run in **subdomain pod mode**. This means user pods live at `alice.solidweb.org`, `alice.solidweb.me`, `alice.teamid.live`, `alice.solidweb.app` instead of path-suffixes like `solidweb.me/alice/`. This is a better UX, cleaner WebIDs, and what production Solid hosting services use.

The consequence: Subdomain mode requires wildcard DNS records (`*.solidweb.org`, etc.) and wildcard TLS certificates. Wildcard certificates require the DNS-01 ACME challenge. DNS-01 requires a Caddy binary compiled with a DNS provider plugin. Therefore **xcaddy is now mandatory, not optional** — it is required before any server can run correctly.

This changes the installation order: xcaddy must be installed and Caddy rebuilt **before** writing the Caddyfile or starting any service.

Master Port & Domain Reference

Service	Public URL	Pod URL pattern	Caddy listens	Internal port
Start page	`https://solidontrixie.net`	—	`:443`	static file
NSS prod	`https://solidweb.org`	`alice.solidweb.org`	`:443`	`127.0.0.1:8443`
CSS prod	`https://solidweb.me`	`alice.solidweb.me`	`:443`	`127.0.0.1:3000`
Pivot prod	`https://teamid.live`	`alice.teamid.live`	`:443`	`127.0.0.1:3001`
JSS prod	`https://solidweb.app`	`alice.solidweb.app`	`:443`	`127.0.0.1:3002`
NSS test	`https://solidontrixie.net:8443`	—	`:8443`	`127.0.0.1:8443`
CSS test	`https://solidontrixie.net:3000`	—	`:3000`	`127.0.0.1:3000`
Pivot test	`https://solidontrixie.net:3001`	—	`:3001`	`127.0.0.1:3001`
JSS test	`https://solidontrixie.net:3002`	—	`:3002`	`127.0.0.1:3002`
Uptime Kuma	`https://status.solidontrixie.net`	—	`:443`	`127.0.0.1:3003`
Netdata	`https://netdata.solidontrixie.net`	—	`:443`	`127.0.0.1:19999`

> **Port coexistence:** Each backend binds to `127.0.0.1` (loopback only). Caddy binds to `0.0.0.0` for the test port blocks. Different network interfaces — they coexist without conflict. Verify with `ss -tlnp` after every startup.

⚠ Architecture Notes — Read Before Touching Anything

xcaddy is now mandatory

All four production domains require wildcard TLS certificates for subdomain pod mode. Wildcard certificates require DNS-01 ACME challenge. DNS-01 requires a Caddy binary with a DNS provider plugin compiled in. The stock `apt install caddy` binary does not include DNS plugins. You must build a custom binary with xcaddy before the Caddyfile can work for production traffic.

If your DNS is at Cloudflare, use `github.com/caddy-dns/cloudflare`. If at Hetzner, `github.com/caddy-dns/hetzner`. The full provider list is at `https://github.com/caddy-dns`.

NSS runs HTTPS internally; all others run plain HTTP

NSS hardcodes HTTPS and cannot be reconfigured to plain HTTP. It runs on `localhost:8443` with a self-signed local certificate. Caddy proxies to it with `tls_insecure_skip_verify` — safe because this connection never leaves the loopback interface.

CSS, Pivot, JSS, and Uptime Kuma all speak plain HTTP internally. No local certs needed for them.

Wildcard DNS is required for every production domain

For each domain where subdomain pods will be created, you need **both** a root A record and a wildcard A record in DNS:

```
solidweb.org.   A  <YOUR_SERVER_IP>
*.solidweb.org. A  <YOUR_SERVER_IP>
solidweb.me.    A  <YOUR_SERVER_IP>
*.solidweb.me.  A  <YOUR_SERVER_IP>
teamid.live.    A  <YOUR_SERVER_IP>
*.teamid.live.  A  <YOUR_SERVER_IP>
solidweb.app.   A  <YOUR_SERVER_IP>
*.solidweb.app. A  <YOUR_SERVER_IP>
```

These must be in place before you start any Solid server. Caddy needs DNS to propagate before it can complete the DNS-01 challenge and issue wildcard certificates.

Pivot's actual start command

Pivot's README documents the real production launch command. It is **not** `npm start` — it is `npx community-solid-server` with specific `-c ./config/prod.json` and `-m .` arguments. `npm run build` must also be run after cloning. The PM2 ecosystem file must reflect this exactly.

Subdomain mode and test ports

The test URLs (`solidontrixie.net:3000` , etc.) still proxy to the same backend processes. Because the backends are configured with their production `baseUrl` (e.g. `https://solidweb.me/`), OIDC tokens and resource URLs generated through the test port will still reference the production domain. The test port is strictly for verifying that Caddy routing and TLS termination work — it is not a separate identity context.

Part A — ACME Challenges and xcaddy

A.1 — The Three ACME Challenge Types

When Let's Encrypt issues a certificate, it must first verify that you control the domain. This verification is an **ACME challenge**. There are three types.

HTTP-01 — automatic, no configuration, no wildcards

How it works: Let's Encrypt gives Caddy a token. Caddy temporarily serves a file at:

```
http://<your-domain>/.well-known/acme-challenge/<token>
```

Let's Encrypt makes an inbound HTTP GET request from multiple internet vantage points. If it retrieves the correct token, the certificate is issued.

What you do: Nothing. Caddy handles this fully automatically when you write any site block with a public domain name.

Requirements: Port 80 must be publicly reachable inbound.

Cannot do: Wildcard certificates (`*.solidweb.org`). This is a hard rule in the ACME protocol specification — not a Caddy limitation. HTTP-01 is cryptographically limited to single-name certificates.

In this guide: HTTP-01 is used automatically for `solidontrixie.net` and its subdomains (`status.solidontrixie.net` , `netdata.solidontrixie.net`). No configuration needed for these.

TLS-ALPN-01 — automatic fallback, also no wildcards

How it works: The challenge is answered during the TLS handshake on port 443. Caddy presents a specially crafted temporary certificate. Let's Encrypt connects to port 443, inspects the handshake, and grants the certificate if it finds the expected challenge value.

What you do: Nothing. Caddy falls back to this automatically if port 80 is unreachable.

Requirements: Port 443 must be publicly reachable inbound.

Cannot do: Wildcard certificates. Same hard protocol rule as HTTP-01.

In this guide: You will never need to think about this explicitly. It is a transparent fallback.

DNS-01 — required for wildcards, requires xcaddy

How it works: Instead of serving a file over HTTP or TLS, your ACME client creates a DNS TXT record:

```
_acme-challenge.solidweb.me.  TXT  "<token-value>"
```

Let's Encrypt queries the public DNS system for that TXT record. The challenge never touches port 80 or 443 — it is entirely DNS-based. This is why it can prove control of `*.solidweb.me` when HTTP-01 cannot: it operates at the domain level, not the per-server level.

Requirements:

- Your DNS provider must have an API that Caddy can call to create and delete TXT records.
- You need API credentials from your DNS provider.
- Your Caddy binary must include a compiled-in plugin for your specific DNS provider.
- DNS propagation must complete (usually 30–120 seconds) before Let's Encrypt checks.

The unique capability: DNS-01 is the **only ACME challenge type that can produce wildcard certificates**. If you need `alice.solidweb.me` to have valid TLS, you need DNS-01. There is no alternative.

Propagation caveat: After creating the TXT record, Caddy waits for DNS to propagate before proceeding. Caddy handles the waiting automatically. If your DNS TTLs are very high (> 5 minutes) or your provider is slow, first-time cert issuance can take a few minutes longer than usual.

In this guide: DNS-01 is used for all four production wildcard certificates: `*.solidweb.org` , `*.solidweb.me` , `*.teamid.live` , `*.solidweb.app` . This requires xcaddy.

A.2 — xcaddy: Installing and Building

Why the apt binary cannot be used for production

Caddy is a Go binary — all functionality is compiled in at build time, not loaded at runtime. DNS provider plugins are separate Go modules that must be compiled into the binary. The standard `apt install caddy` binary includes only core Caddy modules. No DNS plugins are included.

xcaddy is the official tool that wraps `go build` to produce a custom Caddy binary with any set of plugins. The result is a single static binary indistinguishable from a normal Caddy binary — same systemd unit, same config format, same everything — except it has the DNS plugin compiled in.

The full list of available DNS provider plugins is at **https://github.com/caddy-dns (https://github.com/caddy-dns)**. Find your provider there before proceeding.

Step A.2.1 — Install Go (latest stable)

The Go version in Debian apt lags. Get the current release directly:

```
# Fetch latest stable Go version string
GO_VERSION=$(curl -s https://go.dev/VERSION?m=text | head -1)
echo "Installing ${GO_VERSION}"

wget "https://go.dev/dl/${GO_VERSION}.linux-amd64.tar.gz" -O /tmp/go.tar.gz
rm -rf /usr/local/go
tar -C /usr/local -xzf /tmp/go.tar.gz
rm /tmp/go.tar.gz

# System-wide PATH
cat > /etc/profile.d/golang.sh << 'EOF'
export PATH=$PATH:/usr/local/go/bin
export GOPATH=/root/go
export PATH=$PATH:/root/go/bin
EOF

source /etc/profile.d/golang.sh
go version   # must print go1.24.x or later
```

Step A.2.2 — Install apt Caddy first (for systemd + user + dirs)

Even though you will replace the binary, install the apt package first. It creates the `caddy` system user, the `/etc/caddy/` directory, the `/var/lib/caddy/` data directory, and the systemd unit. All of this is still used with the custom binary.

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' \
  | gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' \
  | tee /etc/apt/sources.list.d/caddy-stable.list

apt update && apt install -y caddy
```

Step A.2.3 — Install xcaddy

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/xcaddy/gpg.key' \
  | gpg --dearmor -o /usr/share/keyrings/caddy-xcaddy-archive-keyring.gpg

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/xcaddy/debian.deb.txt' \
  | tee /etc/apt/sources.list.d/caddy-xcaddy.list

apt update && apt install -y xcaddy
xcaddy version
```

Step A.2.4 — Build Caddy with your DNS provider plugin

Replace `cloudflare` with whichever provider is in your DNS: `hetzner`, `route53`, `digitalocean`, `namecheap`, `porkbun`, `gandi`, `ovh`, `desec`, etc. The module path follows the pattern `github.com/caddy-dns/<provider>`.

```
# Build with Cloudflare DNS plugin — change 'cloudflare' to your provider
xcaddy build \
  --with github.com/caddy-dns/cloudflare \
  --output /usr/local/bin/caddy-custom

# Verify the plugin is present in the binary
/usr/local/bin/caddy-custom list-modules | grep dns.providers
# Expected: dns.providers.cloudflare (or your provider name)
```

Step A.2.5 — Replace the system binary

```
systemctl stop caddy

# Preserve the original as a fallback
cp /usr/bin/caddy /usr/bin/caddy.orig

# Install the custom binary
cp /usr/local/bin/caddy-custom /usr/bin/caddy
chmod 755 /usr/bin/caddy

# Allow binding to ports below 1024 (needed for 80 and 443)
setcap cap_net_bind_service=+ep /usr/bin/caddy

# Verify
caddy version
caddy list-modules | grep dns.providers
```

Step A.2.6 — Store DNS API credentials

Never put API tokens directly in the Caddyfile. Store them in an environment file:

```
mkdir -p /etc/caddy
cat > /etc/caddy/caddy.env << 'EOF'
# Replace with your actual DNS provider token
CLOUDFLARE_API_TOKEN=your_cloudflare_api_token_here
EOF
chmod 600 /etc/caddy/caddy.env
chown root:caddy /etc/caddy/caddy.env
```

Add the EnvironmentFile to the Caddy systemd service via a drop-in override:

```
mkdir -p /etc/systemd/system/caddy.service.d
cat > /etc/systemd/system/caddy.service.d/env.conf << 'EOF'
[Service]
EnvironmentFile=/etc/caddy/caddy.env
EOF

systemctl daemon-reload
# Do not start Caddy yet — write the Caddyfile first (section 12)
```

Step A.2.7 — Rebuild script for future Caddy updates

Save this as `/usr/local/bin/rebuild-caddy.sh` :

```
#!/bin/bash
set -euo pipefail

# Change PROVIDER to match what you used in A.2.4
PROVIDER="cloudflare"

CADDY_VER=$(curl -s https://api.github.com/repos/caddyserver/caddy/releases/latest \
  | jq -r .tag_name)
echo "Building Caddy ${CADDY_VER} with caddy-dns/${PROVIDER}..."

xcaddy build "${CADDY_VER}" \
  --with "github.com/caddy-dns/${PROVIDER}" \
  --output /usr/local/bin/caddy-custom

systemctl stop caddy
cp /usr/local/bin/caddy-custom /usr/bin/caddy
chmod 755 /usr/bin/caddy
setcap cap_net_bind_service=+ep /usr/bin/caddy
systemctl start caddy

echo "Done. Caddy version: $(caddy version)"
echo "DNS modules: $(caddy list-modules | grep dns.providers)"
```

```
chmod +x /usr/local/bin/rebuild-caddy.sh
```

Run this whenever a new Caddy version is released.

Part B — System Installation

1. System Preparation

```
apt update && apt full-upgrade -y

apt install -y \
  curl git build-essential openssl libssl-dev ca-certificates \
  gnupg2 debian-keyring debian-archive-keyring apt-transport-https \
  net-tools ufw logrotate jq

# Service users — isolated, no interactive login, own home directory
useradd --system --shell /usr/sbin/nologin --create-home --home-dir /srv/nss    nss-solid
useradd --system --shell /usr/sbin/nologin --create-home --home-dir /srv/css    css-solid
useradd --system --shell /usr/sbin/nologin --create-home --home-dir /srv/pivot  pivot-solid
useradd --system --shell /usr/sbin/nologin --create-home --home-dir /srv/jss    jss-solid
useradd --system --shell /usr/sbin/nologin --create-home --home-dir /srv/kuma   kuma

# Log directories
mkdir -p /var/log/{nss,css,pivot,jss,kuma,caddy}
chown nss-solid:nss-solid     /var/log/nss
chown css-solid:css-solid     /var/log/css
chown pivot-solid:pivot-solid /var/log/pivot
chown jss-solid:jss-solid     /var/log/jss
chown kuma:kuma               /var/log/kuma
# /var/log/caddy — set after Caddy install
```

2. Install Caddy (xcaddy path — now mandatory)

Follow **Part A.2** in its entirety before proceeding. You must complete all steps through A.2.6 before writing the Caddyfile or starting any service. To summarise the order:

```
1. Install Go   (A.2.1)
2. apt install caddy   (A.2.2)
3. apt install xcaddy   (A.2.3)
4. xcaddy build --with github.com/caddy-dns/<provider>   (A.2.4)
5. Replace /usr/bin/caddy with custom binary   (A.2.5)
6. Create /etc/caddy/caddy.env + systemd drop-in   (A.2.6)
7. systemctl daemon-reload   (end of A.2.6)
```

Then set log ownership:

```
chown caddy:caddy /var/log/caddy
systemctl enable caddy
# Do NOT start caddy yet — write the Caddyfile first (section 12)
```

3. Self-Signed Certificate for NSS (loopback only)

NSS always runs HTTPS internally. This certificate is used only for the internal `Caddy → localhost:8443 → NSS` connection. It never leaves the machine and is never shown to external clients.

```
mkdir -p /etc/ssl/nss-local

openssl req \
  -newkey rsa:4096 \
  -nodes \
  -keyout /etc/ssl/nss-local/privkey.pem \
  -x509 \
  -days 3650 \
  -out /etc/ssl/nss-local/fullchain.pem \
  -subj "/CN=localhost" \
  -addext "subjectAltName=IP:127.0.0.1,DNS:localhost"

chown -R nss-solid:nss-solid /etc/ssl/nss-local
chmod 640 /etc/ssl/nss-local/privkey.pem
chmod 644 /etc/ssl/nss-local/fullchain.pem

# Verify
openssl x509 -in /etc/ssl/nss-local/fullchain.pem -noout -text \
  | grep -E 'Subject:|DNS:|IP Address:|Not After'
```

4. Install nvm + Node 24 Per Service User

```
install_node_for_user() {
  local SVCUSER="$1"
  local HOMEDIR="$2"
  echo "==> nvm + Node 24 for ${SVCUSER}"

  su -s /bin/bash "$SVCUSER" -c "
    export HOME='${HOMEDIR}'
    export NVM_DIR='${HOMEDIR}/.nvm'
    curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
    source \"\${NVM_DIR}/nvm.sh\"
    nvm install 24
    nvm alias default 24
    echo '${SVCUSER}: Node' \$(node --version) 'npm' \$(npm --version)
  "
}

install_node_for_user nss-solid   /srv/nss
install_node_for_user css-solid   /srv/css
install_node_for_user pivot-solid /srv/pivot
install_node_for_user jss-solid   /srv/jss
install_node_for_user kuma        /srv/kuma
```

5. Install PM2 Per Service User

```
install_pm2_for_user() {
  local SVCUSER="$1"
  local HOMEDIR="$2"
  echo "==> PM2 for ${SVCUSER}"

  su -s /bin/bash "$SVCUSER" -c "
    export HOME='${HOMEDIR}'
    export NVM_DIR='${HOMEDIR}/.nvm'
    source \"\${NVM_DIR}/nvm.sh\"
    npm install -g pm2
    pm2 install pm2-logrotate
    pm2 set pm2-logrotate:max_size 50M
    pm2 set pm2-logrotate:retain 14
    pm2 set pm2-logrotate:compress true
  "
}

install_pm2_for_user nss-solid   /srv/nss
install_pm2_for_user css-solid   /srv/css
install_pm2_for_user pivot-solid /srv/pivot
install_pm2_for_user jss-solid   /srv/jss
install_pm2_for_user kuma        /srv/kuma
```

6. Node Solid Server (NSS)

Pod mode: Subdomain — pods live at `alice.solidweb.org` (default for NSS with `multiuser: true`).

Wildcard DNS required: `*.solidweb.org → <YOUR_SERVER_IP>`

6a. Install

```
su -s /bin/bash nss-solid -c "
  export HOME=/srv/nss
  export NVM_DIR=/srv/nss/.nvm
  source \$NVM_DIR/nvm.sh
  npm install -g solid
  mkdir -p /srv/nss/data /srv/nss/config /srv/nss/data/.db
"
```

6b. Config: `/srv/nss/config/config.json`

```
{
  "root": "/srv/nss/data",
  "port": "8443",
  "serverUri": "https://solidweb.org",
  "webid": true,
  "mount": "/",
  "configPath": "/srv/nss/config",
  "dbPath": "/srv/nss/data/.db",
  "auth": "oidc",
  "strictOrigin": true,
  "multiuser": true,
  "live": true,
  "secret": "CHANGE_THIS_TO_A_RANDOM_STRING_MIN_32_CHARS",
  "ssl": {
    "key": "/etc/ssl/nss-local/privkey.pem",
    "cert": "/etc/ssl/nss-local/fullchain.pem"
  },
  "corsProxy": false,
  "enforceToc": false,
  "supportEmail": "admin@solidweb.org",
  "useEmail": false
}
```

> `serverUri` must be `https://solidweb.org` exactly — no trailing slash. NSS derives OIDC `iss` values and redirect URIs from this. Any mismatch causes silent token rejection.
>
> NSS subdomain mode is automatic when `"multiuser": true`. User registration creates `username.solidweb.org`.

6c. PM2 ecosystem: `/srv/nss/ecosystem.config.js`

```
module.exports = {
  apps: [
    {
      name: 'nss',
      script: 'solid',
      args: 'start --config /srv/nss/config/config.json',
      cwd: '/srv/nss',
      interpreter: 'none',
      env: {
        HOME: '/srv/nss',
        NVM_DIR: '/srv/nss/.nvm',
        NODE_ENV: 'production',
      },
      out_file:           '/var/log/nss/stdout.log',
      error_file:         '/var/log/nss/stderr.log',
      log_date_format:    'YYYY-MM-DD HH:mm:ss Z',
      autorestart:        true,
      restart_delay:      5000,
      max_restarts:       10,
      min_uptime:         '30s',
      max_memory_restart: '1G',
      watch:              false,
    }
  ]
}
```

If `interpreter: 'none'` causes a spawn error, find the binary path first:

```
sudo -u nss-solid bash -c "source /srv/nss/.nvm/nvm.sh && which solid"
```

Then use that absolute path as `script`.

7. Community Solid Server (CSS)

Pod mode: Subdomain — pods live at `alice.solidweb.me`.

Wildcard DNS required: `*.solidweb.me → <YOUR_SERVER_IP>`

7a. Install

```
su -s /bin/bash css-solid -c "
  export HOME=/srv/css
  export NVM_DIR=/srv/css/.nvm
  source \$NVM_DIR/nvm.sh
  npm install -g @solid/community-server
  mkdir -p /srv/css/data /srv/css/config
"
```

7b. Custom config: `/srv/css/config/custom-config.json`

The key change from v4 is replacing `css:config/util/identifiers/suffix.json` with `css:config/util/identifiers/subdomain.json`. This switches from path-based (`solidweb.me/alice/`) to subdomain-based (`alice.solidweb.me`) pod URLs.

```
{
  "@context": "https://linkedsoftwaredependencies.org/bundles/npm/@solid/community-server/^7/components/context.jsonl
  "import": [
    "css:config/app/main/default.json",
    "css:config/app/init/initialize-root.json",
    "css:config/app/setup/optional-setup.json",
    "css:config/app/variables/default.json",
    "css:config/http/handler/default.json",
    "css:config/http/middleware/websockets.json",
    "css:config/http/server-factory/http.json",
    "css:config/http/static/default.json",
    "css:config/identity/access/public.json",
    "css:config/identity/email/default.json",
    "css:config/identity/handler/default.json",
    "css:config/identity/oidc/interaction/default.json",
    "css:config/ldp/authorization/webacl.json",
    "css:config/ldp/handler/default.json",
    "css:config/ldp/metadata-parser/default.json",
    "css:config/ldp/metadata-writer/default.json",
    "css:config/ldp/modes/default.json",
    "css:config/pod/manager/account-manager.json",
    "css:config/pod/static/root-pod.json",
    "css:config/storage/key-value/resource-store.json",
    "css:config/storage/middleware/default.json",
    "css:config/util/auxiliary/acl.json",
    "css:config/util/identifiers/subdomain.json",
    "css:config/util/index/default.json",
    "css:config/util/logging/winston.json",
    "css:config/util/representationconverter/default.json",
    "css:config/util/resource-locker/memory.json",
    "css:config/util/variables/default.json"
  ],
  "@graph": [
    {
      "@id": "urn:solid-server:default:variable:port",
      "@type": "Variable",
      "value": 3000
    },
    {
      "@id": "urn:solid-server:default:variable:baseUrl",
      "@type": "Variable",
      "value": "https://solidweb.me/"
    },
    {
      "@id": "urn:solid-server:default:variable:rootFilePath",
      "@type": "Variable",
      "value": "/srv/css/data"
    },
    {
      "@id": "urn:solid-server:default:variable:showStackTrace",
      "@type": "Variable",
      "value": false
    },
    {
      "@id": "urn:solid-server:default:variable:loggingLevel",
      "@type": "Variable",
      "value": "info"
    }
  ]
}
```

Critical diff from v4: Line `"css:config/util/identifiers/suffix.json"` is now `"css:config/util/identifiers/subdomain.json"`. That single change switches the pod addressing mode.

`baseUrl` must end with a trailing slash (`https://solidweb.me/`). CSS v7 is strict about this — omitting it causes broken resource URL generation.

When a user registers with pod name `alice`, their pod lives at `https://alice.solidweb.me/` and their WebID is `https://alice.solidweb.me/profile/card#me`.

7c. PM2 ecosystem: `/srv/css/ecosystem.config.js`

```
module.exports = {
  apps: [
    {
      name: 'css',
      script: 'community-solid-server',
      args: '--config /srv/css/config/custom-config.json',
      cwd: '/srv/css',
      interpreter: 'none',
      env: {
        HOME: '/srv/css',
        NVM_DIR: '/srv/css/.nvm',
        NODE_ENV: 'production',
      },
      out_file:           '/var/log/css/stdout.log',
      error_file:         '/var/log/css/stderr.log',
      log_date_format:    'YYYY-MM-DD HH:mm:ss Z',
      autorestart:        true,
      restart_delay:      5000,
      max_restarts:       10,
      min_uptime:         '30s',
      max_memory_restart: '1500M',
      watch:              false,
    }
  ]
}
```

8. Solid Pivot

Pod mode: Subdomain — pods live at `alice.teamid.live`.

Wildcard DNS required: `*.teamid.live → <YOUR_SERVER_IP>`

Pivot is a thin wrapper around CSS. Its production configuration (`config/prod.json`) uses subdomain-based pod identifiers. The actual launch command is `npx community-solid-server` with specific flags — **not** `npm start`. `npm run build` is also required after cloning.

8a. Install

```
su -s /bin/bash pivot-solid -c "
  export HOME=/srv/pivot
  export NVM_DIR=/srv/pivot/.nvm
  source \$NVM_DIR/nvm.sh

  git clone https://github.com/solid-contrib/pivot.git /srv/pivot/app
  cd /srv/pivot/app

  # npm ci is preferred (exact lockfile versions); --omit=dev skips devDependencies
  npm ci --omit=dev

  # Build is required — Pivot has a TypeScript compilation step
  npm run build

  # Create the data directory and copy in the required www assets
  mkdir -p /srv/pivot/data
  cp -r /srv/pivot/app/www /srv/pivot/data/
"
```

8b. Custom config: `/srv/pivot/app/custom-config.json`

Copy from the included template and edit it:

```
sudo -u pivot-solid cp /srv/pivot/app/config/customise-me.json \
                       /srv/pivot/app/custom-config.json
```

The key fields you must set in `custom-config.json`:

```
{
  // Email settings — fill in if you want account email verification.
  // At minimum set a valid SMTP host or leave the block empty.
  // Pod quota — default is 70 MB per pod
  "quota": 73400320,

  // Mashlib version — uses the version bundled in node_modules by default
  // You can pin to a specific CDN version if needed
}
```

After cloning, always read the actual `customise-me.json` to see the current schema:

```
cat /srv/pivot/app/config/customise-me.json
```

It may add new fields across Pivot releases.

8c. PM2 ecosystem: `/srv/pivot/ecosystem.config.js`

The start command reconstructed from Pivot's README and the teamid.live repository:

```
module.exports = {
  apps: [
    {
      name: 'pivot',
      // Pivot is launched via the CSS binary, not npm start
      // -c: config files (prod.json is Pivot's subdomain production config)
      // -f: data directory
      // -b: public base URL (must match exactly, with trailing slash)
      // -m: mainModulePath — tells Components.js to look in CWD for Pivot components
      // -p: port (plain HTTP, Caddy handles TLS)
      script: 'npx',
      args: [
        'community-solid-server',
        '-c', './config/prod.json', './custom-config.json',
        '-f', '/srv/pivot/data',
        '-b', 'https://teamid.live/',
        '-m', '.',
        '-p', '3001',
        '-l', 'info',
      ].join(' '),
      cwd: '/srv/pivot/app',
      interpreter: 'none',
      env: {
        HOME: '/srv/pivot',
        NVM_DIR: '/srv/pivot/.nvm',
        NODE_ENV: 'production',
      },
      out_file:           '/var/log/pivot/stdout.log',
      error_file:         '/var/log/pivot/stderr.log',
      log_date_format:    'YYYY-MM-DD HH:mm:ss Z',
      autorestart:        true,
      restart_delay:      5000,
      max_restarts:       10,
      min_uptime:         '30s',
      max_memory_restart: '1500M',
      watch:              false,
    }
  ]
}
```

> **`-b https://teamid.live/`** — the trailing slash is required by CSS/Pivot's URL resolution. Do not omit it.
>
> **`-m .`** — this tells Components.js to start its module lookup from the Pivot app directory (`/srv/pivot/app`). This is what makes Pivot's own CSS extensions (the `pivot:` namespace in `prod.json`) resolve correctly. Without `-m .` you will get `Error: Cannot resolve component` errors on startup.
>
> **`-p 3001`** — Pivot does not have a `--host` flag to bind to loopback. It binds to all interfaces by default. UFW blocks external access to port 3001 in production traffic. For the test URL `solidontrixie.net:3001` , Caddy mediates all inbound connections on that port.
>
> After any Pivot update, re-run `npm run build` before restarting.

8d. Port binding note for Pivot

Unlike JSS, Pivot (via CSS) does not expose a `--host` or `--bind` flag. It will bind to `0.0.0.0:3001` . UFW's `ufw deny 3001` from external sources (only allowing inbound on the test port via Caddy) is the protection mechanism here. Verify after startup:

```
ss -tlnp | grep ':3001'
# Will show 0.0.0.0:3001 for Pivot — this is expected.
# External traffic on :3001 goes to Caddy (0.0.0.0 is shared but Caddy gets it first
# via SO_REUSEPORT — actually Caddy and Pivot cannot both bind 0.0.0.0:3001).
```

⚠ **Important:** Because both Caddy (listening publicly on port 3001 for the test endpoint) and Pivot (listening on all interfaces on port 3001) would normally conflict, you have two options:

Option A (recommended): Run Pivot on an internal-only port `3101` and have Caddy proxy to `localhost:3101` . The test endpoint `solidontrixie.net:3001` then proxies to `localhost:3101` . Change `-p 3001` to `-p 3101` in the ecosystem args, update the Caddyfile to point to `localhost:3101` , and update UFW to not open 3101 externally (it's Caddy-only). This is the clean solution.

Option B: Do not open port 3001 publicly in UFW and do not provide a test endpoint for Pivot. Only the production domain `teamid.live` is accessible.

> **This guide uses Option A.** All Pivot references below use internal port `3101` .

Update the ecosystem args accordingly:

```
args: [
  'community-solid-server',
  '-c', './config/prod.json', './custom-config.json',
  '-f', '/srv/pivot/data',
  '-b', 'https://teamid.live/',
  '-m', '.',
  '-p', '3101',   // ← internal port, not exposed externally
  '-l', 'info',
].join(' '),
```

Update the port map: Pivot internal is now `127.0.0.1:3101` (though Pivot binds `0.0.0.0:3101`).

9. JavaScript Solid Server (JSS)

Pod mode: Subdomain — pods live at `alice.solidweb.app` .

Wildcard DNS required: `*.solidweb.app → <YOUR_SERVER_IP>`

JSS is installed by git clone. It is not published as a global npm package.

9a. Install

```
su -s /bin/bash jss-solid -c "
  export HOME=/srv/jss
  export NVM_DIR=/srv/jss/.nvm
  source \$NVM_DIR/nvm.sh

  git clone https://github.com/JavaScriptSolidServer/JavaScriptSolidServer.git /srv/jss/app
  cd /srv/jss/app
  npm install
  mkdir -p /srv/jss/data
"
```

9b. Inspect the repo before configuring

```
# Confirm the bin entry point
sudo -u jss-solid bash -c "ls /srv/jss/app/bin/"
sudo -u jss-solid bash -c "cat /srv/jss/app/package.json | jq '{name, version, bin, scripts}'"
```

9c. Config file: `/srv/jss/app/custom-config.json`

```
{
  "port": 3002,
  "host": "127.0.0.1",
  "root": "/srv/jss/data",
  "idp": true,
  "idpIssuer": "https://solidweb.app",
  "notifications": true,
  "conneg": true,
  "subdomains": true,
  "baseDomain": "solidweb.app",
  "mashlib": false,
  "mashlib-cdn": false,
  "quiet": false
}
```

`"host": "127.0.0.1"` — binds JSS to loopback only. Critical: without this, JSS defaults to `0.0.0.0` and will conflict with Caddy's public listener on port 3002.

`"subdomains": true` — enables subdomain-based pod mode. Pods live at `alice.solidweb.app/`.

`"baseDomain": "solidweb.app"` — the domain JSS uses to construct subdomain pod URLs. Must match the public production domain exactly, without protocol or trailing slash.

`"idpIssuer": "https://solidweb.app"` — the OIDC issuer URL. Must match the public URL exactly. This is what gets put into OIDC token `iss` fields.

9d. Root ACL: `/srv/jss/data/.acl`

```
mkdir -p /srv/jss/data
cat > /srv/jss/data/.acl << 'EOF'
{
  "@context": {
    "acl": "http://www.w3.org/ns/auth/acl#",
    "foaf": "http://xmlns.com/foaf/0.1/"
  },
  "@graph": [
    {
      "@id": "#owner",
      "@type": "acl:Authorization",
      "acl:agent": { "@id": "https://solidweb.app/profile/card#me" },
      "acl:accessTo": { "@id": "https://solidweb.app/" },
      "acl:default": { "@id": "https://solidweb.app/" },
      "acl:mode": [
        { "@id": "acl:Read" },
        { "@id": "acl:Write" },
        { "@id": "acl:Control" }
      ]
    },
    {
      "@id": "#public",
      "@type": "acl:Authorization",
      "acl:agentClass": { "@id": "foaf:Agent" },
      "acl:accessTo": { "@id": "https://solidweb.app/" },
      "acl:default": { "@id": "https://solidweb.app/" },
      "acl:mode": [
        { "@id": "acl:Read" }
      ]
    }
  ]
}
EOF

chown jss-solid:jss-solid /srv/jss/data/.acl
```

9e. PM2 ecosystem: `/srv/jss/ecosystem.config.js`

```
module.exports = {
  apps: [
    {
      name: 'jss',
      script: '/srv/jss/app/bin/jss',
      args: 'start --config /srv/jss/app/custom-config.json',
      cwd: '/srv/jss/app',
      interpreter: 'node',
      env: {
        HOME: '/srv/jss',
        NVM_DIR: '/srv/jss/.nvm',
        NODE_ENV: 'production',
        // Belt-and-suspenders env vars in case config file isn't picked up
        JSS_PORT: '3002',
        JSS_HOST: '127.0.0.1',
        JSS_ROOT: '/srv/jss/data',
      },
      out_file:           '/var/log/jss/stdout.log',
      error_file:         '/var/log/jss/stderr.log',
      log_date_format:    'YYYY-MM-DD HH:mm:ss Z',
      autorestart:        true,
      restart_delay:      5000,
      max_restarts:       10,
      min_uptime:         '30s',
      max_memory_restart: '1G',
      watch:              false,
    }
  ]
}
```

If the path `bin/jss` is wrong after cloning:

```
sudo -u jss-solid bash -c "cat /srv/jss/app/package.json | jq '.bin'"
```

Use whatever path that returns.

10. Uptime Kuma — status.solidontrixie.net (http://status.solidontrixie.net) (port 3003)

```
su -s /bin/bash kuma -c "
  export HOME=/srv/kuma
  export NVM_DIR=/srv/kuma/.nvm
  source \$NVM_DIR/nvm.sh
  git clone https://github.com/louislam/uptime-kuma.git /srv/kuma/app
  cd /srv/kuma/app && npm run setup
  mkdir -p /srv/kuma/data
"
```

/srv/kuma/ecosystem.config.js

```
module.exports = {
  apps: [
    {
      name: 'uptime-kuma',
      script: '/srv/kuma/app/server/server.js',
      cwd: '/srv/kuma/app',
      env: {
        HOME: '/srv/kuma',
        NVM_DIR: '/srv/kuma/.nvm',
        NODE_ENV: 'production',
        UPTIME_KUMA_PORT: '3003',
        DATA_DIR: '/srv/kuma/data',
      },
      out_file:           '/var/log/kuma/stdout.log',
      error_file:         '/var/log/kuma/stderr.log',
      log_date_format:    'YYYY-MM-DD HH:mm:ss Z',
      autorestart:        true,
      restart_delay:      5000,
      max_restarts:       10,
      min_uptime:         '30s',
      max_memory_restart: '512M',
      watch:              false,
    }
  ]
}
```

11. Start Page for `solidontrixie.net`

```
mkdir -p /var/www/solidontrixie
```

`/var/www/solidontrixie/index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Solid on Trixie</title>
  <style>
    body { font-family: system-ui, sans-serif; max-width: 680px;
           margin: 4rem auto; padding: 1rem; line-height: 1.65; }
    h1 { font-size: 1.5rem; }
    h2 { font-size: 1.1rem; margin-top: 2rem; border-bottom: 1px solid #eee;
         padding-bottom: .3rem; }
    ul { padding-left: 1.2rem; }
    a  { color: #1a73e8; }
    code { background: #f4f4f4; padding: 2px 6px; border-radius: 3px; font-size: .9em; }
  </style>
</head>
<body>
  <h1>Solid on Trixie — Test Environment</h1>
  <p>Four <a href="https://solidproject.org">Solid</a> server implementations
     on Debian Trixie, managed by <a href="https://pm2.io">PM2</a>,
     fronted by <a href="https://caddyserver.com">Caddy</a>.</p>

  <h2>Production servers (subdomain pod mode)</h2>
  <ul>
    <li><a href="https://solidweb.org">solidweb.org</a>
        — <a href="https://github.com/nodeSolidServer/node-solid-server">Node Solid Server</a>
        — pods at <code>alice.solidweb.org</code></li>
    <li><a href="https://solidweb.me">solidweb.me</a>
        — <a href="https://github.com/CommunitySolidServer/CommunitySolidServer">Community Solid Server</a>
        — pods at <code>alice.solidweb.me</code></li>
    <li><a href="https://teamid.live">teamid.live</a>
        — <a href="https://github.com/solid-contrib/pivot">Solid Pivot</a>
        — pods at <code>alice.teamid.live</code></li>
    <li><a href="https://solidweb.app">solidweb.app</a>
        — <a href="https://javascriptsolidserver.github.io/docs/">JavaScript Solid Server</a>
        — pods at <code>alice.solidweb.app</code></li>
  </ul>

  <h2>Test endpoints (same backends)</h2>
  <ul>
    <li><a href="https://solidontrixie.net:8443">solidontrixie.net:8443</a> — NSS</li>
    <li><a href="https://solidontrixie.net:3000">solidontrixie.net:3000</a> — CSS</li>
    <li><a href="https://solidontrixie.net:3001">solidontrixie.net:3001</a> — Pivot</li>
    <li><a href="https://solidontrixie.net:3002">solidontrixie.net:3002</a> — JSS</li>
  </ul>

  <h2>Infrastructure</h2>
  <ul>
    <li><a href="https://status.solidontrixie.net">status.solidontrixie.net</a> — Uptime Kuma</li>
    <li><a href="https://netdata.solidontrixie.net">netdata.solidontrixie.net</a> — Netdata (auth required)</li>
  </ul>
</body>
</html>
```

12. Caddyfile (`/etc/caddy/Caddyfile`)

The Caddyfile is written after completing Part A.2 in full (xcaddy built, binary replaced, `caddy.env` created, systemd drop-in in place).

```
# ================================================================================
# Global options
# ================================================================================
{
  email admin@solidontrixie.net

  # ── STAGING — uncomment during initial testing to avoid LE rate limits ──
  # Staging certs are NOT browser-trusted; use `curl -k` for testing.
  # Comment out once all domains validate, then `systemctl restart caddy`.
  # acme_ca https://acme-staging-v02.api.letsencrypt.org/directory
}


# ================================================================================
# solidontrixie.net — start page + TLS anchor for test port endpoints
#
# Certificate obtained via HTTP-01 (no wildcard needed for this domain —
# the test endpoints reuse this cert because they share the same hostname).
# ================================================================================
solidontrixie.net {
  root * /var/www/solidontrixie
  file_server

  log {
    output file /var/log/caddy/solidontrixie.net.log {
      roll_size 20MiB
      roll_keep 14
      roll_keep_for 720h
    }
    format json
  }

  encode gzip zstd
}

status.solidontrixie.net {
  reverse_proxy localhost:3003 {
    header_up Host              {host}
    header_up X-Real-IP         {remote_host}
    header_up X-Forwarded-For   {remote_host}
    header_up X-Forwarded-Proto {scheme}
    header_up Upgrade           {http.upgrade}
    header_up Connection        {http.connection}
    flush_interval -1
  }
  log {
    output file /var/log/caddy/uptime-kuma.log { roll_size 20MiB; roll_keep 7 }
    format json
  }
}

netdata.solidontrixie.net {
  # Generate: caddy hash-password --plaintext 'your-password-here'
  basicauth {
    admin <PASTE_HASHED_PASSWORD_HERE>
  }
  reverse_proxy localhost:19999
  log {
    output file /var/log/caddy/netdata.log { roll_size 20MiB; roll_keep 7 }
    format json
  }
}


# ================================================================================
# Test port endpoints for solidontrixie.net
#
# All four reuse the certificate from the solidontrixie.net block above.
# Caddy handles this automatically because they share the same hostname.
# ================================================================================

# NSS test — NSS runs HTTPS internally with self-signed local cert
solidontrixie.net:8443 {
  reverse_proxy https://localhost:8443 {
    transport http {
      tls_insecure_skip_verify
    }
    header_up Host              {host}
```

```
      header_up X-Real-IP         {remote_host}
      header_up X-Forwarded-For   {remote_host}
      header_up X-Forwarded-Proto {scheme}
      header_up X-Forwarded-Host  {host}
      header_up Upgrade           {http.upgrade}
      header_up Connection        {http.connection}
      flush_interval -1
    }
    log {
      output file /var/log/caddy/solidontrixie-8443.log { roll_size 20MiB; roll_keep 14 }
      format json
    }
    encode gzip zstd
  }

  # CSS test — plain HTTP backend
  solidontrixie.net:3000 {
    reverse_proxy localhost:3000 {
      header_up Host              {host}
      header_up X-Real-IP         {remote_host}
      header_up X-Forwarded-For   {remote_host}
      header_up X-Forwarded-Proto {scheme}
      header_up X-Forwarded-Host  {host}
      header_up Upgrade           {http.upgrade}
      header_up Connection        {http.connection}
      flush_interval -1
    }
    log {
      output file /var/log/caddy/solidontrixie-3000.log { roll_size 20MiB; roll_keep 14 }
      format json
    }
    encode gzip zstd
  }

  # Pivot test — proxies to internal port 3101 (Option A, see section 8d)
  solidontrixie.net:3001 {
    reverse_proxy localhost:3101 {
      header_up Host              {host}
      header_up X-Real-IP         {remote_host}
      header_up X-Forwarded-For   {remote_host}
      header_up X-Forwarded-Proto {scheme}
      header_up X-Forwarded-Host  {host}
      header_up Upgrade           {http.upgrade}
      header_up Connection        {http.connection}
      flush_interval -1
    }
    log {
      output file /var/log/caddy/solidontrixie-3001.log { roll_size 20MiB; roll_keep 14 }
      format json
    }
    encode gzip zstd
  }

  # JSS test — plain HTTP backend, binds to 127.0.0.1:3002
  solidontrixie.net:3002 {
    reverse_proxy localhost:3002 {
      header_up Host              {host}
      header_up X-Real-IP         {remote_host}
      header_up X-Forwarded-For   {remote_host}
      header_up X-Forwarded-Proto {scheme}
      header_up X-Forwarded-Host  {host}
      header_up Upgrade           {http.upgrade}
      header_up Connection        {http.connection}
      flush_interval -1
    }
    log {
      output file /var/log/caddy/solidontrixie-3002.log { roll_size 20MiB; roll_keep 14 }
      format json
    }
    encode gzip zstd
  }


  # ============================================================================
  # solidweb.org — NSS production + wildcard for alice.solidweb.org pods
  #
  # DNS-01 challenge — requires xcaddy binary with DNS provider plugin.
  # *.solidweb.org DNS record must exist before Caddy can issue the cert.
  # ============================================================================
```

```
    *.solidweb.org, solidweb.org {
      tls {
        # Replace 'cloudflare' and the env var name with your DNS provider
        dns cloudflare {env.CLOUDFLARE_API_TOKEN}
      }

      reverse_proxy https://localhost:8443 {
        transport http {
          tls_insecure_skip_verify
        }
        header_up Host              {host}
        header_up X-Real-IP         {remote_host}
        header_up X-Forwarded-For   {remote_host}
        header_up X-Forwarded-Proto {scheme}
        header_up X-Forwarded-Host  {host}
        header_up Upgrade           {http.upgrade}
        header_up Connection        {http.connection}
        flush_interval -1
      }

      log {
        output file /var/log/caddy/solidweb.org.log {
          roll_size 50MiB
          roll_keep 14
          roll_keep_for 720h
        }
        format json
      }

      encode gzip zstd
    }


    # ============================================================================
    # solidweb.me — CSS production + wildcard for alice.solidweb.me pods
    # ============================================================================
    *.solidweb.me, solidweb.me {
      tls {
        dns cloudflare {env.CLOUDFLARE_API_TOKEN}
      }

      reverse_proxy localhost:3000 {
        header_up Host              {host}
        header_up X-Real-IP         {remote_host}
        header_up X-Forwarded-For   {remote_host}
        header_up X-Forwarded-Proto {scheme}
        header_up X-Forwarded-Host  {host}
        header_up Upgrade           {http.upgrade}
        header_up Connection        {http.connection}
        flush_interval -1
      }

      log {
        output file /var/log/caddy/solidweb.me.log {
          roll_size 50MiB
          roll_keep 14
          roll_keep_for 720h
        }
        format json
      }

      encode gzip zstd
    }


    # ============================================================================
    # teamid.live — Pivot production + wildcard for alice.teamid.live pods
    # Proxies to internal port 3101 (Pivot's internal port, see section 8d)
    # ============================================================================
    *.teamid.live, teamid.live {
      tls {
        dns cloudflare {env.CLOUDFLARE_API_TOKEN}
      }

      reverse_proxy localhost:3101 {
        header_up Host              {host}
        header_up X-Real-IP         {remote_host}
        header_up X-Forwarded-For   {remote_host}
        header_up X-Forwarded-Proto {scheme}
```

```
      header_up X-Forwarded-Host  {host}
      header_up Upgrade           {http.upgrade}
      header_up Connection        {http.connection}
      flush_interval -1
    }

    log {
      output file /var/log/caddy/teamid.live.log {
        roll_size 50MiB
        roll_keep 14
        roll_keep_for 720h
      }
      format json
    }

    encode gzip zstd
  }


  # ================================================================================
  # solidweb.app — JSS production + wildcard for alice.solidweb.app pods
  # ================================================================================
  *.solidweb.app, solidweb.app {
    tls {
      dns cloudflare {env.CLOUDFLARE_API_TOKEN}
    }

    reverse_proxy localhost:3002 {
      header_up Host              {host}
      header_up X-Real-IP         {remote_host}
      header_up X-Forwarded-For   {remote_host}
      header_up X-Forwarded-Proto {scheme}
      header_up X-Forwarded-Host  {host}
      header_up Upgrade           {http.upgrade}
      header_up Connection        {http.connection}
      flush_interval -1
    }

    log {
      output file /var/log/caddy/solidweb.app.log {
        roll_size 50MiB
        roll_keep 14
        roll_keep_for 720h
      }
      format json
    }

    encode gzip zstd
  }



  # Validate config — must pass before starting Caddy
  caddy validate --config /etc/caddy/Caddyfile

  # Start Caddy (first time)
  systemctl start caddy

  # After initial cert issuance is verified, reload is preferred over restart
  systemctl reload caddy

  # Watch cert issuance in real time
  journalctl -u caddy -f
```

13. DNS Records

Every production domain needs both an apex A record and a wildcard A record. There are no separate records needed for the test ports — ports are not DNS names.

```
# solidontrixie.net — start page, monitoring, test port TLS anchor
solidontrixie.net.           A    <YOUR_SERVER_IP>
status.solidontrixie.net.    A    <YOUR_SERVER_IP>
netdata.solidontrixie.net.   A    <YOUR_SERVER_IP>

# solidweb.org (NSS) — apex + wildcard for subdomain pods
solidweb.org.                A    <YOUR_SERVER_IP>
*.solidweb.org.              A    <YOUR_SERVER_IP>

# solidweb.me (CSS) — apex + wildcard for subdomain pods
solidweb.me.                 A    <YOUR_SERVER_IP>
*.solidweb.me.               A    <YOUR_SERVER_IP>

# teamid.live (Pivot) — apex + wildcard for subdomain pods
teamid.live.                 A    <YOUR_SERVER_IP>
*.teamid.live.               A    <YOUR_SERVER_IP>

# solidweb.app (JSS) — apex + wildcard for subdomain pods
solidweb.app.                A    <YOUR_SERVER_IP>
*.solidweb.app.              A    <YOUR_SERVER_IP>

# IPv6 — add AAAA records if the Strasbourg server has a public IPv6 address
solidontrixie.net.           AAAA <YOUR_IPv6>
solidweb.org.                AAAA <YOUR_IPv6>
*.solidweb.org.              AAAA <YOUR_IPv6>
solidweb.me.                 AAAA <YOUR_IPv6>
*.solidweb.me.               AAAA <YOUR_IPv6>
teamid.live.                 AAAA <YOUR_IPv6>
*.teamid.live.               AAAA <YOUR_IPv6>
solidweb.app.                AAAA <YOUR_IPv6>
*.solidweb.app.              AAAA <YOUR_IPv6>
```

Before starting Caddy: Verify all records resolve correctly:

```
# Check apex records
for d in solidweb.org solidweb.me teamid.live solidweb.app solidontrixie.net; do
  echo -n "${d}: " && dig +short A ${d}
done

# Check wildcard records
for d in solidweb.org solidweb.me teamid.live solidweb.app; do
  echo -n "*.${d}: " && dig +short A alice.${d}
done
```

All should return your server IP. If any wildcard returns empty, DNS has not propagated yet — wait and retry.

14. Firewall (UFW)

```
# Standard web
ufw allow 22/tcp    comment 'SSH'
ufw allow 80/tcp    comment 'ACME HTTP-01 (for solidontrixie.net)'
ufw allow 443/tcp   comment 'HTTPS production + subdomains'

# Test ports — Caddy handles TLS, cert reused from solidontrixie.net block
ufw allow 8443/tcp  comment 'solidontrixie.net NSS test'
ufw allow 3000/tcp  comment 'solidontrixie.net CSS test'
ufw allow 3001/tcp  comment 'solidontrixie.net Pivot test (Caddy→3101 internally)'
ufw allow 3002/tcp  comment 'solidontrixie.net JSS test'

# Internal only — never expose directly
ufw deny  3003      comment 'Uptime Kuma — Caddy-proxied on 443 only'
ufw deny  3101      comment 'Pivot internal — Caddy-proxied only'
ufw deny  19999     comment 'Netdata — Caddy-proxied on 443 only'

ufw enable
ufw status verbose
```

15. Start All Services with PM2

```
start_pm2_user() {
  local SVCUSER="$1"
  local HOMEDIR="$2"
  local ECOFILE="$3"

  sudo -u "$SVCUSER" bash -c "
    export HOME='${HOMEDIR}'
    export NVM_DIR='${HOMEDIR}/.nvm'
    source \"\${NVM_DIR}/nvm.sh\"
    pm2 start '${ECOFILE}' --env production
    pm2 save
  "
}

start_pm2_user nss-solid   /srv/nss   /srv/nss/ecosystem.config.js
start_pm2_user css-solid   /srv/css   /srv/css/ecosystem.config.js
start_pm2_user pivot-solid /srv/pivot /srv/pivot/ecosystem.config.js
start_pm2_user jss-solid   /srv/jss   /srv/jss/ecosystem.config.js
start_pm2_user kuma        /srv/kuma  /srv/kuma/ecosystem.config.js
```

16. PM2 Startup on Boot

```
setup_pm2_startup() {
  local SVCUSER="$1"
  local HOMEDIR="$2"

  NODE_BIN=$(sudo -u "$SVCUSER" bash -c "
    export HOME='${HOMEDIR}'
    source '${HOMEDIR}/.nvm/nvm.sh'
    which node
  ")
  NODE_DIR=$(dirname "$NODE_BIN")
  PM2_BIN="${NODE_DIR}/pm2"

  echo "==> PM2 startup for ${SVCUSER} (node: ${NODE_DIR})"

  sudo env PATH="${NODE_DIR}:${PATH}" \
    "${PM2_BIN}" startup systemd \
    -u "${SVCUSER}" \
    --hp "${HOMEDIR}"
}

setup_pm2_startup nss-solid   /srv/nss
setup_pm2_startup css-solid   /srv/css
setup_pm2_startup pivot-solid /srv/pivot
setup_pm2_startup jss-solid   /srv/jss
setup_pm2_startup kuma        /srv/kuma

systemctl daemon-reload
```

17. Install Netdata

```
curl https://get.netdata.cloud/kickstart.sh | sh
systemctl enable --now netdata
```

Bind to localhost: `/etc/netdata/netdata.conf`

```
[global]
  run as user = netdata

[web]
  bind to = 127.0.0.1
  port = 19999
  allow connections from = localhost
```

```
systemctl restart netdata
ss -tlnp | grep 19999   # must show 127.0.0.1:19999
```

18. PM2 Quick Reference Aliases

Add to `/root/.bashrc`:

```
alias pm2-nss="sudo -u nss-solid bash -c \
  'export HOME=/srv/nss; source /srv/nss/.nvm/nvm.sh; pm2'"
alias pm2-css="sudo -u css-solid bash -c \
  'export HOME=/srv/css; source /srv/css/.nvm/nvm.sh; pm2'"
alias pm2-pivot="sudo -u pivot-solid bash -c \
  'export HOME=/srv/pivot; source /srv/pivot/.nvm/nvm.sh; pm2'"
alias pm2-jss="sudo -u jss-solid bash -c \
  'export HOME=/srv/jss; source /srv/jss/.nvm/nvm.sh; pm2'"
alias pm2-kuma="sudo -u kuma bash -c \
  'export HOME=/srv/kuma; source /srv/kuma/.nvm/nvm.sh; pm2'"


source /root/.bashrc
```

19. Uptime Kuma Monitor List

Monitor name	Type	Target	Interval
solidontrixie.net (http://solidontrixie.net)	HTTP(s)	`https://solidontrixie.net`	60s
NSS test port	HTTP(s)	`https://solidontrixie.net:8443`	60s
CSS test port	HTTP(s)	`https://solidontrixie.net:3000`	60s
Pivot test port	HTTP(s)	`https://solidontrixie.net:3001`	60s
JSS test port	HTTP(s)	`https://solidontrixie.net:3002`	60s
solidweb.org (http://solidweb.org)	HTTP(s)	`https://solidweb.org`	60s
solidweb.me (http://solidweb.me)	HTTP(s)	`https://solidweb.me`	60s
teamid.live	HTTP(s)	`https://teamid.live`	60s
solidweb.app	HTTP(s)	`https://solidweb.app`	60s
NSS internal	TCP Port	`localhost` / `8443`	30s
CSS internal	TCP Port	`localhost` / `3000`	30s
Pivot internal	TCP Port	`localhost` / `3101`	30s
JSS internal	TCP Port	`localhost` / `3002`	30s
Kuma self	TCP Port	`localhost` / `3003`	30s
Caddy HTTPS	TCP Port	`localhost` / `443`	30s
Netdata	HTTP(s)	`http://localhost:19999`	60s

20. Logrotate (`/etc/logrotate.d/solid-servers`)

```
/var/log/nss/*.log
/var/log/css/*.log
/var/log/pivot/*.log
/var/log/jss/*.log
/var/log/kuma/*.log
/var/log/caddy/*.log {
    daily
    rotate 14
    compress
    delaycompress
    missingok
    notifempty
    copytruncate
}
```

21. Verification Checklist

```
echo "=== 0. Pre-flight: DNS wildcard resolution ==="
for d in solidweb.org solidweb.me teamid.live solidweb.app; do
  echo -n "  apex  ${d}: " && dig +short A ${d}
  echo -n "  wild  *.${d}: " && dig +short A alice.${d}
done

echo "=== 1. Pre-flight: no port conflicts ==="
ss -tlnp | grep -E ':8443|:3000|:3001|:3002|:3003|:3101|:19999'
# All should be empty before any service starts

echo "=== 2. Caddy custom binary verified ==="
caddy version
caddy list-modules | grep dns.providers   # must show your provider
caddy validate --config /etc/caddy/Caddyfile

echo "=== 3. PM2 processes ==="
for u_h in "nss-solid /srv/nss" "css-solid /srv/css" \
           "pivot-solid /srv/pivot" "jss-solid /srv/jss" "kuma /srv/kuma"; do
  read U H <<< "$u_h"
  echo "--- ${U} ---"
  sudo -u "$U" bash -c "export HOME='${H}'; source '${H}/.nvm/nvm.sh'; pm2 list"
done

echo "=== 4. Listening ports after startup ==="
ss -tlnp | grep -E ':80 |:443 |:8443 |:3000 |:3001 |:3002 |:3003 |:3101 |:19999'

echo "=== 5. Backend loopback health ==="
curl -sk https://localhost:8443 | head -3   # NSS (-k: local self-signed)
curl -s  http://localhost:3000  | head -3   # CSS
curl -s  http://localhost:3101  | head -3   # Pivot (internal port)
curl -s  http://localhost:3002  | head -3   # JSS
curl -s  http://localhost:3003  | head -3   # Uptime Kuma
curl -s  http://localhost:19999/api/v1/info | jq '{version: .version, os: .os}'

echo "=== 6. Public HTTPS — production domains ==="
for d in solidweb.org solidweb.me teamid.live solidweb.app; do
  printf "%-22s " "${d}:"
  curl -sI "https://${d}" | grep -E 'HTTP|Location'
done

echo "=== 7. Public HTTPS — test ports ==="
curl -sI https://solidontrixie.net                  | grep HTTP
curl -sI https://solidontrixie.net:8443             | grep HTTP
curl -sI https://solidontrixie.net:3000             | grep HTTP
curl -sI https://solidontrixie.net:3001             | grep HTTP
curl -sI https://solidontrixie.net:3002             | grep HTTP

echo "=== 8. TLS certificate issuer per domain ==="
for d in solidontrixie.net solidweb.org solidweb.me teamid.live solidweb.app; do
  printf "%-25s " "${d}:"
  echo | openssl s_client -connect "${d}:443" -servername "${d}" 2>/dev/null \
    | openssl x509 -noout -issuer 2>/dev/null
done

echo "=== 9. Wildcard TLS — test a pod subdomain on each domain ==="
for d in solidweb.org solidweb.me teamid.live solidweb.app; do
  printf "  test.%-22s " "${d}:"
  echo | openssl s_client -connect "test.${d}:443" -servername "test.${d}" 2>/dev/null \
    | openssl x509 -noout -issuer 2>/dev/null
done

echo "=== 10. Node 24 per user ==="
for u_h in "nss-solid /srv/nss" "css-solid /srv/css" \
           "pivot-solid /srv/pivot" "jss-solid /srv/jss" "kuma /srv/kuma"; do
  read U H <<< "$u_h"
  VER=$(sudo -u "$U" bash -c "source '${H}/.nvm/nvm.sh' && node --version")
  echo "  ${U}: ${VER}"
done

echo "=== 11. Caddy + Netdata service status ==="
systemctl status caddy   --no-pager -l | head -10
systemctl status netdata --no-pager -l | head -10
```

```
echo "=== 12. Firewall ==="
ufw status verbose
```

22. Directory Layout

```
/etc/
├── caddy/
│   ├── Caddyfile
│   └── caddy.env             ← DNS API token (chmod 600, chown root:caddy)
├── ssl/
│   └── nss-local/
│       ├── privkey.pem       ← self-signed key (loopback TLS, NSS only)
│       └── fullchain.pem     ← self-signed cert (loopback TLS, NSS only)
├── systemd/system/caddy.service.d/
│   └── env.conf              ← EnvironmentFile drop-in
└── logrotate.d/
    └── solid-servers

/srv/
├── nss/                      ← nss-solid home
│   ├── .nvm/
│   ├── data/ + data/.db/
│   ├── config/config.json
│   └── ecosystem.config.js
├── css/                      ← css-solid home
│   ├── .nvm/
│   ├── data/
│   ├── config/custom-config.json
│   └── ecosystem.config.js
├── pivot/                    ← pivot-solid home
│   ├── .nvm/
│   ├── app/                  ← git clone of solid-contrib/pivot
│   │   ├── config/           ← prod.json, dev configs (from repo)
│   │   └── custom-config.json ← copied from config/customise-me.json
│   ├── data/
│   │   └── www/              ← copied from app/www after npm run build
│   └── ecosystem.config.js
├── jss/                      ← jss-solid home
│   ├── .nvm/
│   ├── app/                  ← git clone of JavaScriptSolidServer
│   │   ├── bin/jss
│   │   └── custom-config.json
│   ├── data/
│   │   └── .acl              ← required root ACL
│   └── ecosystem.config.js
└── kuma/                     ← kuma home
    ├── .nvm/
    ├── app/                  ← git clone of louislam/uptime-kuma
    ├── data/                 ← SQLite database
    └── ecosystem.config.js

/var/
├── www/solidontrixie/
│   └── index.html
└── log/
    ├── caddy/                ← 10 log files, one per Caddyfile site block
    ├── nss/   {stdout,stderr}.log
    ├── css/   {stdout,stderr}.log
    ├── pivot/ {stdout,stderr}.log
    ├── jss/   {stdout,stderr}.log
    └── kuma/  {stdout,stderr}.log

/usr/
├── bin/caddy                 ← xcaddy custom binary (replaces apt binary)
├── bin/caddy.orig            ← apt binary backup
└── local/bin/
    ├── caddy-custom          ← xcaddy build output (keep as source)
    └── rebuild-caddy.sh      ← rebuild script for Caddy upgrades
```

23. Upgrade Paths

```
# NSS
sudo -u nss-solid bash -c "source /srv/nss/.nvm/nvm.sh && npm update -g solid"
pm2-nss restart nss

# CSS
sudo -u css-solid bash -c "source /srv/css/.nvm/nvm.sh && npm update -g @solid/community-server"
pm2-css restart css

# Pivot
sudo -u pivot-solid bash -c "
  source /srv/pivot/.nvm/nvm.sh
  cd /srv/pivot/app
  git pull
  npm ci --omit=dev
  npm run build
  cp -r /srv/pivot/app/www /srv/pivot/data/
"
pm2-pivot restart pivot

# JSS
sudo -u jss-solid bash -c "
  source /srv/jss/.nvm/nvm.sh
  cd /srv/jss/app && git pull && npm install
"
pm2-jss restart jss

# Uptime Kuma
sudo -u kuma bash -c "
  source /srv/kuma/.nvm/nvm.sh
  cd /srv/kuma/app
  git fetch --all
  git checkout \$(git tag -l | grep -v beta | sort -V | tail -n1)
  npm run setup
"
pm2-kuma restart uptime-kuma

# Caddy (with xcaddy — use the rebuild script)
/usr/local/bin/rebuild-caddy.sh

# Netdata
/usr/libexec/netdata/netdata-updater.sh
```

24. Critical Gotchas — Everything That Will Bite You

xcaddy is now mandatory, not optional. All four production domains need wildcard certs. Wildcard certs need DNS-01. DNS-01 needs a DNS provider plugin. The plugin needs xcaddy. If you start Caddy with the apt binary and the wildcard Caddyfile blocks, Caddy will start fine but then fail cert issuance with an error like `acme: error: 403 :: urn:ietf:params:acme:error:unauthorized`. Follow Part A.2 completely before writing the Caddyfile.

Wildcard DNS records before starting Caddy. Caddy needs to create a DNS TXT record for the DNS-01 challenge. It does this via the DNS provider API. But the challenge verification itself happens over DNS. If the A records for `*.solidweb.me` don't exist yet, no HTTPS traffic can reach the server anyway — but more importantly, Caddy needs the DNS API credentials and provider module to be correct. Verify DNS records with `dig` before starting Caddy.

Staging certs first, always. Uncomment the `acme_ca` staging line in the global block, test everything with `curl -k`, then comment it back out and restart for real certs. One failed real-cert run per domain can impose a 7-day rate limit on that domain from Let's Encrypt.

CSS subdomain mode: the one-line config change. The only difference from suffix mode is `subdomain.json` instead of `suffix.json` in the identifiers import. That single line determines whether pods are at `solidweb.me/alice/` or `alice.solidweb.me/`. If you copy-paste the v4 config without checking this, you will have path-based pods despite having wildcard DNS and certs ready.

Pivot internal port 3101, not 3001. Pivot cannot bind to loopback-only. If both Caddy and Pivot tried to bind `0.0.0.0:3001`, the second one would fail. The solution is Pivot on port 3101 (internal, never exposed externally) and Caddy's test endpoint on public port 3001 proxying to `localhost:3101`. Every reference to Pivot's internal port in this guide uses 3101. The `ufw deny`

`3101` rule ensures it is never directly reachable. The Caddyfile test block proxies `solidontrixie.net:3001 → localhost:3101`, and the production block proxies `teamid.live → localhost:3101`.

Pivot requires `npm run build`. After cloning and after every `git pull`, you must run `npm run build`. Pivot has a TypeScript compilation step. The compiled output in `dist/` is what actually runs. Skipping this gives `Cannot find module` errors.

Pivot requires `cp -r www data/`. The static Mashlib assets must be in the data directory at startup. The Pivot README documents this. If the `www/` directory is missing from `/srv/pivot/data/`, the Mashlib data browser will not load.

Pivot `-m .` flag is non-negotiable. The `-m .` argument tells Components.js (the dependency injection framework) to look in the current working directory for module definitions. Without it, the `pivot:` namespace entries in `prod.json` cannot be resolved, and the server crashes on startup with `Error: Cannot resolve component`.

JSS `"host": "127.0.0.1"` is critical. Without it, JSS binds to `0.0.0.0:3002`. Caddy also binds to `0.0.0.0:3002` for the test endpoint. Both binding to the same interface and port causes a conflict — whichever starts second will fail. Set `"host": "127.0.0.1"` in the JSS config so it binds to loopback only. This is the most likely JSS-specific failure.

JSS root `.acl` must exist before first start. JSS refuses to serve resources without the root `.acl` file. The file must be at `data/.acl` and use JSON-LD format. The content provided in section 9d is a correct baseline.

NSS `serverUri` no trailing slash. `https://solidweb.org` correct. `https://solidweb.org/` wrong. NSS generates OIDC `iss` values by appending paths to `serverUri`. A trailing slash produces double-slash URLs that break OIDC validation.

CSS `baseUrl` trailing slash required. `https://solidweb.me/` correct. `https://solidweb.me` wrong. CSS v7 constructs resource URLs by appending paths to `baseUrl`. Without the trailing slash it produces paths like `https://solidweb.mealice/profile/card` which is obviously broken.

The wildcard cert covers `*.solidweb.me` but NOT `solidweb.me`. A wildcard certificate only covers names one label below the wildcard. `*.solidweb.me` covers `alice.solidweb.me` but not `solidweb.me` itself. In the Caddyfile, the site address `*.solidweb.me, solidweb.me { ... }` includes both, and Caddy requests a certificate with both SANs. Do not separate these into two blocks — they must be combined in one block sharing a single `tls { dns ... }` directive.

Uptime Kuma WebSocket. The Uptime Kuma UI is entirely WebSocket-driven. The `header_up Upgrade` and `header_up Connection` lines in the Caddyfile block for `status.solidontrixie.net` are required. Without them the dashboard loads then immediately goes blank.

PM2 startup command and nvm PATH. PM2's systemd startup unit hard-codes the PATH including the nvm Node binary directory. Never try to `source nvm.sh` inside a systemd unit — systemd does not process shell profile files. The `setup_pm2_startup` function in section 16 generates the correct unit with the resolved path.