GitLab & GitLab Pages on Separate IPs

Self?Hosted GitLab & GitLab Pages on Separate IPs

> **Goal** Run the core GitLab instance and the GitLab Pages service on **different IP addresses** while using **Let’s Encrypt** certificates managed outside of Omnibus. This guide documents every key `gitlab.rb` setting required, why it exists, and the common pitfalls that bite first-time deployments.

---

1 Topology Overview

Component	FQDN	Listens on	Description
GitLab (core)	`git.❱ PRIMARY_DOMAIN ❰`	`PRIMARY_IP:443/80`	Standard web UI/API, served by Omnibus NGINX
GitLab Pages	`prod.❱ PRIMARY_DOMAIN ❰`	`PAGES_IP:443/80`	Serves static pages; runs its own Go HTTP server

```
Internet ─▶ 443 → PRIMARY_IP  ─┐
                               │  Omnibus NGINX  → GitLab Core
Internet ─▶ 443 → PAGES_IP    ─┴─ gitlab-pages (direct bind)
```

- **Distinct IPs** prevent port clashes and simplify TLS.
- **Let’s Encrypt via certbot** is used for both hostnames; GitLab’s internal ACME is disabled.

---

2 `gitlab.rb` – Directive?by?Directive Explanation

```
external_url 'https://git.PRIMARY_DOMAIN'
```

Sets the canonical URL for the **core** GitLab instance. All internal links, OAuth callbacks, and API clients rely on this value.

```
letsencrypt['enable'] = false
```

Disables Omnibus' automatic ACME integration. You manage certificates yourself with certbot (or any other tool).

```
nginx['listen_addresses'] = ['PRIMARY_IP']
```

Tells Omnibus NGINX **only** to bind to the primary IP. Prevents it from stealing `:443` on the Pages IP.

```
nginx['ssl_certificate']     = '/etc/letsencrypt/live/git.PRIMARY_DOMAIN/fullchain.pem'
nginx['ssl_certificate_key'] = '/etc/letsencrypt/live/git.PRIMARY_DOMAIN/privkey.pem'
```

Full-path PEM pair for the **core** GitLab site. Read directly from certbot's live directory.

GitLab Pages block

```
pages_external_url 'https://prod.PRIMARY_DOMAIN'
```

Public URL end-users visit for Pages content. Must match the CN/SAN in the cert below.

```
gitlab_pages['enable'] = true
```

Self-explanatory—starts the Pages service.

```
gitlab_pages['external_http']  = ['PAGES_IP:80']
gitlab_pages['external_https'] = ['PAGES_IP:443']
```

Direct binding mode. Pages listens on its own IP instead of being proxied through NGINX.

```
gitlab_pages['cert']     = '/etc/letsencrypt/live/prod.PRIMARY_DOMAIN/fullchain.pem'
gitlab_pages['cert_key'] = '/etc/letsencrypt/live/prod.PRIMARY_DOMAIN/privkey.pem'
```

PEM pair for the **Pages** hostname. Since `inplace_chroot` is disabled (see below), the service can reach the real FS path.

```
gitlab_pages['inplace_chroot'] = false
```

Disables the default chroot jail. Simplifies cert management in containerised environments where an extra security layer is less critical.

```
gitlab_pages['acme']['enabled'] = false
```

Stops Pages from requesting its own ACME certs—which would clash with certbot.

```
pages_nginx['enable'] = false
```

Omnibus can spawn an internal NGINX reverse-proxy in front of Pages. We turn it **off** because Pages is binding directly.

```
package['modify_kernel_parameters'] = false
```

On some cloud images/containers, Omnibus cannot change sysctl values. This flag avoids Chef failures.

---

3 Certbot Shortcuts

```
# Issue certs (example)
sudo certbot certonly --standalone -d git.PRIMARY_DOMAIN -d prod.PRIMARY_DOMAIN -m
you@example.com --agree-tos
```

Auto?reload Pages after renewal

Create `/etc/letsencrypt/renewal-hooks/post/gitlab-pages-reload.sh`:

```
#!/bin/sh
# Reload Pages after certbot renews prod.PRIMARY_DOMAIN
/usr/bin/gitlab-ctl hup gitlab-pages
```

`chmod +x` it. Certbot’s timer will run this automatically.

---

4 Firewall Rules

IP	80	443
`PRIMARY_IP`	☐	☐
`PAGES_IP`	☐	☐

Block all other inbound ports.

---

5 Troubleshooting Cheat?Sheet

Symptom	Common Cause	Fix
`address already in use :443` in Pages log	Omnibus NGINX bound to 0.0.0.0	Set `nginx['listen_addresses']` to primary IP only
`open /etc/…crt: no such file or directory`	Wrong cert path / chroot mismatch	Disable chroot or copy cert into `…/gitlab-pages/etc/`
`gitlab-pages: runsv not running`	`gitlab-runsvdir` service dead	`systemctl start gitlab-runsvdir && systemctl enable gitlab-runsvdir`
All services `runsv not running`	Container rebooted without runit	Same as above

---

5½ Keeping the supervisor (gitlab?runsvdir) alive

GitLab's **runit supervision tree** is launched by the systemd unit **`gitlab-runsvdir.service`**. If that unit is *inactive* every Omnibus component will show `runsv not running` and no ports will be open.

Why it dies

- The VM/container reboots and systemd starts services **before** networking is up; Pages fails to bind to its IP and runit exits.
- Manual `systemctl stop` or a runaway `OOM-killer` event.

Make it start reliably

```
# one-off recovery
sudo systemctl start gitlab-runsvdir

# persistent across reboots
sudo systemctl enable gitlab-runsvdir
```

Add a network dependency so the secondary IPs exist before runit starts:

```
# /etc/systemd/system/gitlab-runsvdir.service (snippet)
[Unit]
After=network-online.target
Wants=network-online.target
```

Optional watchdog timer

A tiny timer restarts the supervisor if it ever stops unexpectedly:

```
# /etc/systemd/system/gitlab-runsvdir-watchdog.timer
[Unit]
Description=Restart gitlab-runsvdir if it exits

[Timer]
OnBootSec=5min
OnUnitInactiveSec=1min
Unit=gitlab-runsvdir.service

[Install]
WantedBy=timers.target
```

Enable with `systemctl enable --now gitlab-runsvdir-watchdog.timer`.

When `gitlab-runsvdir` is healthy you will always see both listeners after boot:

```
ss -ltnp | grep :443
# 172.31.14.12:443 nginx
# 172.31.14.11:443 gitlab-pages
```

---

6 Security Notes

- **Direct binding** (no Pages NGINX) means the Go Pages server terminates TLS itself.
- **Disabling chroot** removes one sandbox layer. On single-tenant VMs or Docker containers this is usually acceptable; on multi-tenant hosts you might prefer to keep the chroot and copy the PEMs into the jail instead.
- `gitlab-secrets.json` **relocated** If you move or mount-inject the secrets file, Omnibus can no longer create its fallback self-signed certs. In this guide we disable *all* Omnibus ACME features (`letsencrypt['enable'] = false`, `gitlab_pages['acme']['enabled'] = false`) and provide Let's Encrypt PEMs manually, so the missing secrets file is harmless—**just ensure certbot renewal is working**.
- **Automatic renewal** Remember to reload services after certbot renews. A one-line renewal hook can do this:

```
#!/bin/sh
/usr/bin/gitlab-ctl hup gitlab-pages
/usr/bin/gitlab-ctl hup nginx
```

7 Full Example `gitlab.rb` Full Example `gitlab.rb`

```
external_url 'https://git.PRIMARY_DOMAIN'

letsencrypt['enable'] = false

nginx['listen_addresses'] = ['PRIMARY_IP']
nginx['ssl_certificate']     = '/etc/letsencrypt/live/git.PRIMARY_DOMAIN/fullchain.pem'
nginx['ssl_certificate_key'] = '/etc/letsencrypt/live/git.PRIMARY_DOMAIN/privkey.pem'

pages_external_url 'https://prod.PRIMARY_DOMAIN'
gitlab_pages['enable'] = true

gitlab_pages['external_http']  = ['PAGES_IP:80']
gitlab_pages['external_https'] = ['PAGES_IP:443']

# direct certbot PEMs
gitlab_pages['cert']     = '/etc/letsencrypt/live/prod.PRIMARY_DOMAIN/fullchain.pem'
gitlab_pages['cert_key'] = '/etc/letsencrypt/live/prod.PRIMARY_DOMAIN/privkey.pem'

gitlab_pages['inplace_chroot'] = false
gitlab_pages['acme']['enabled'] = false
```

```
pages_nginx['enable'] = false
package['modify_kernel_parameters'] = false
```

Replace:

- `PRIMARY_DOMAIN` → your apex domain (e.g. *jack.water.house*)
- `PRIMARY_IP` → IP mapped to *git.PRIMARY_DOMAIN*
- `PAGES_IP` → IP mapped to *prod.PRIMARY_DOMAIN*

---

8 Command Quick?Reference

```
# Apply config
gitlab-ctl reconfigure

# Start / stop Pages
gitlab-ctl restart gitlab-pages
gitlab-ctl tail gitlab-pages

# Restart entire stack after system boot
systemctl start gitlab-runsvdir
```

Document prepared · May 2025

9 When `gitlab-secrets.json` (aka *secrets.rb*) is relocated

Omnibus keeps its encryption keys (CI JWTs, LDAP secrets, backup encryption keys, etc.) in `/etc/gitlab/gitlab-secrets.json`—older docs sometimes call this *secrets.rb*. If the file is moved outside **/etc/gitlab**, GitLab can no longer read the self-signed certificate or private keys it once generated. The result is TLS mis-configuration and, if `letsencrypt['enable']` is turned on, ACME registration failures.

Fix

- Keep `letsencrypt['enable'] = false` (use certbot externally).
- Do **not** delete the secrets file—back it up and keep it under `/etc/gitlab`.

10 Chroot ON vs OFF—trade?offs at a glance

Mode	Advantages	Drawbacks
Chroot ON `gitlab_pages['inplace_chroot'] = true`	• Additional isolation (Pages can only see its own tree). • Blocks path-traversal exploits inside user pages.	• Certs must be **copied** into `/var/opt/gitlab/gitlab-pages/etc/`. • Debugging more complex. • Breaks on minimal containers lacking `pivot_root`.
Chroot OFF `gitlab_pages['inplace_chroot'] = false`	• Pages reads PEMs directly from `/etc/letsencrypt/live/...`—no duplication. • Simple certbot renewal hook (`gitlab-ctl hup gitlab-pages`). • Works on any container runtime.	• One less defence layer; rely on VM/container isolation and Unix perms.

Rule of thumb: In single-tenant VMs or containers, disabling the chroot is pragmatic. On a shared host or if you let untrusted users push Pages content, keep the chroot and script the PEM copy in a certbot post-renew hook.

Revision #2
Created 5 May 2025 22:15:37 by Jack Waterhouse
Updated 6 May 2025 22:07:36 by Jack Waterhouse