SCIĄGAWKA DEVOPS

Checklista zabezpieczenia GitLab Runnera

Własny VPS, executor Docker, token glrt-, chroniony runner do deployu

TL;DR Ciepły cache, bez minut, prywatny deploy. Runner wykonuje kod pipeline'u z Twoimi sekretami: osobna maszyna, executor `docker`, `privileged = false`, deploy na runnerze **chronionym**.

WYMAGANIA

- Token `glrt-` z UI. Tokeny rejestracyjne: wyłączone od GitLaba 17.0, usunięcie zaplanowane na 20.0.
- `major.minor` runnera = GitLaba. Tu: `alpine-v19.5.3` (2026-11-10).
- Linux + Docker Engine, rozmiar pod `concurrent` x najcięższy job.
- Tylko wychodzący HTTPS, **bez portu przychodzącego**. Nie na hoście GitLaba ani aplikacji.

BUDOWANIE OBRAZÓW

- **DinD:** `privileged = true`, `/certs/client`, `DOCKER_TLS_CERTDIR: "/certs"`. Ryzyko ucieczki.
- **Podpięcie socketu:** joby sterują demonem hosta = root na hoście.
- **Buildah:** bez demona. **Kaniko:** zarchiwizowane 3 czerwca 2025, najpierw sprawdź upstream.
- Buildy privileged: osobny chroniony runner, osobna VM.

KIEDY NIE Mało buildów. Nikt nie łata maszyny. Projekty publiczne i forki. Skoki równoległości albo wiele architektur.

REJESTRACJA

```
docker compose run --rm gitlab-runner register \
  --non-interactive \
  --url "https://gitlab.com" \
  --token "$RUNNER_TOKEN" \
  --executor docker \
  --docker-image alpine:3.22 \
  --description "vps-runner-1"
```

Potem `chmod 600 config/config.toml`, `docker compose up -d`. Tagi i protected: w UI.

CONFIG.TOML

```
concurrent = 2
listen_address = ":9252"
[[runners]]
  executor = "docker"
  limit = 2
  request_concurrency = 2
  [runners.docker]
    privileged = false
    volumes = ["/cache"]
    pull_policy = ["always", "if-not-present"]
    memory = "4g"
    cpus = "2"
```

Obszar	Sprawdź
Executor	`docker`, nie `shell`; `privileged = false` (inaczej root na hoście); `memory`, `cpus`
Obrazy	`allowed_images`, `allowed_services`, `allowed_pull_policies`; wspólny: nie samo `if-not-present`
Deploy	Runner z tagiem `deploy`, **Protected**; `tags: [deploy]`; `main`, `develop`, zmienne chronione
Tagi	Runner musi mieć **wszystkie** tagi joba; bez run untagged joby bez tagów utkną
Host	Socket tylko tam, gdzie nic innego nie działa; `config.toml` `chmod 600`, poza gitem
Cache	Lokalny `/cache`; S3 od drugiego runnera; mierz `queued_duration` kontra `duration`
Metryki	`/metrics` na `127.0.0.1:9252` (bez auth); alert na niedostępny target
Dysk	Co tydzień `clear-docker-cache`; `docker system prune` na obrazy; potem zimny przebieg
Wersje	Pauza w UI, nowy tag, `docker compose pull && docker compose up -d`