gitlab.local.dev

Author: Adrian DC

Description: Install, configure and maintain the gitlab.local.dev infrastructure with mise tasks

Date: 23/11/2026

Version: `2.0.0-43-g11c8590`

▸ 4. GitLab - Runners

Settings required

- **Configure the GitLab Runner machine settings :** Settings

---

Table of contents

- Settings required
- GitLab Runner
  - Install GitLab Runner
  - Configure GitLab hosts
  - Certificate GitLab Runner
  - Register GitLab Runner
  - Configure GitLab Runner
- Cleanup runner images
  - Watch runner images
  - Configure runner client

GitLab Runner

Install GitLab Runner

- **Install GitLab Runner (`VERSION_RUNNER` setting) :**

```
gitlab.local.dev runner:install
```

Task sources: `runner:install`

```
#!/usr/bin/env bash

# Mise
#MISE description="Install GitLab Runner and its helper images packages ('VERSION_RUNNER' setting)"

# Configurations
set -e -o pipefail

# Sources
# shellcheck source=lib/common.sh
source "${MISE_PROJECT_ROOT:?}/lib/common.sh"

# Prepare settings
settings_require
temporary=$(mktemp -d)
trap 'rm -rf "${temporary:?}"' EXIT

# Download packages
step "Install GitLab Runner ${VERSION_RUNNER}"
for package in gitlab-runner_amd64.deb gitlab-runner-helper-images.deb; do
  wget -O "${temporary}/${package}" \
    "https://s3.dualstack.us-east-1.amazonaws.com/gitlab-runner-downloads/v${VERSION_RUNNER}/deb/${package}"
done

# Install packages
if dpkg -l gitlab-runner gitlab-runner-helper-images >/dev/null 2>&1; then
  sudo apt-get remove -y gitlab-runner gitlab-runner-helper-images
fi
apt_install "${temporary}/gitlab-runner_amd64.deb" "${temporary}/gitlab-runner-helper-images.deb"
```

Configure GitLab hosts

- **Configure the GitLab domains in `/etc/hosts` :**

```
gitlab.local.dev runner:hosts
```

Task sources: `runner:hosts`

```
#!/usr/bin/env bash

# Mise
#MISE description="Configure the GitLab domains in '/etc/hosts' of the runner machine"

# Configurations
set -e -o pipefail

# Sources
# shellcheck source=lib/common.sh
source "${MISE_PROJECT_ROOT:?}/lib/common.sh"

# Prepare settings
settings_require

# Configure system hosts
step "Configure GitLab hosts in '${SYSROOT}/etc/hosts'"
system_block "${SYSROOT}/etc/hosts" 'DNS' "\
${DOMAIN_IP:?} ${DOMAIN_GITLAB:?}
${DOMAIN_IP:?} ${DOMAIN_PAGES:?}
${DOMAIN_IP:?} ${DOMAIN_REGISTRY:?}"
```

Certificate GitLab Runner

- **Acquire and trust the GitLab and Container Registry certificates :**

```
gitlab.local.dev runner:certificates
```

Task sources: `runner:certificates`

```
#!/usr/bin/env bash

# Mise
#MISE description="Acquire and trust the GitLab and Container Registry certificates on the runner machine"

# Configurations
set -e -o pipefail

# Sources
# shellcheck source=lib/common.sh
source "${MISE_PROJECT_ROOT:?}/lib/common.sh"

# Prepare settings
settings_require
certs="${SYSROOT}/etc/gitlab-runner/certs"
sudo mkdir -p "${certs}/"

# Acquire certificates
for server in "gitlab|${PORT_GITLAB:?}|${DOMAIN_GITLAB:?}" "registry|${PORT_REGISTRY:?}|${DOMAIN_REGISTRY:?}"; do
  IFS='|' read -r name port domain <<<"${server}"
  step "Acquire certificate from ${DOMAIN_IP:?}:${port} (${domain})"
  sudo rm -fv "${certs:?}/${name:?}.server.crt"
  certificate_acquire "${DOMAIN_IP:?}:${port}" "${domain}" | sudo tee "${certs}/${name}.server.crt" >/dev/null
  sudo test -s "${certs}/${name}.server.crt" || fail "Empty certificate from ${DOMAIN_IP}:${port}"
  sudo openssl s_client -CAfile "${certs}/${name}.server.crt" -connect "${DOMAIN_IP:?}:${port}" -servername "${domain}"
</dev/null
done

# Install CA certificates
sudo cp -f "${certs}/registry.server.crt" "${SYSROOT}/usr/local/share/ca-certificates/${DOMAIN_REGISTRY:?}.crt"
sudo update-ca-certificates --fresh

# Restart Docker service
confirm 'Restart the Docker service ?'
sudo systemctl restart docker
```

Register GitLab Runner

> ⚠ **First usage only**
>
> This section is optional, use only upon first usage.

- **From GitLab :** Admin Area / Settings / CI/CD / Runners
- **Action :** Create instance runner
- **Tags :** docker
- **Run untagged jobs :** Enable
- **Action :** Create runner
- **Platform / Operating systems :** Linux
- **Action :** Copy `...` after `The runner authentication token`

- **Register the GitLab Runner from the GitLab Runner machine :**

```
gitlab.local.dev runner:register
```

Task sources: `runner:register`

```
#!/usr/bin/env bash

# Mise
#MISE description="Register the GitLab Runner interactively (first usage only, token from GitLab)"

# Configurations
set -e -o pipefail

# Sources
# shellcheck source=lib/common.sh
source "${MISE_PROJECT_ROOT:?}/lib/common.sh"

# Prepare settings
settings_require
gitlab_urls

# Action
step 'Register GitLab Runner'
info "GitLab instance URL: ${external_url}"
info 'Executor: docker, default Docker image: alpine:3'
sudo gitlab-runner register --url "${external_url}" --tls-ca-file /etc/gitlab-runner/certs/gitlab.server.crt
```

- **Enter the GitLab instance URL :** Remote `https://IP[:PORT]` or `https://URL[:PORT]` to GitLab server (prefilled)
- **Enter the registration token :** `glrt-TOKEN_FROM_GITLAB...`
- **Enter a name for the runner :** `...` (or default)
- **Enter an executor :** `docker`
- **Enter the default Docker image :** `alpine:3`

Configure GitLab Runner

- **Configure the GitLab Runner and pull its common images :**

```
gitlab.local.dev runner:configure
```

Task sources: `runner:configure`

```
#!/usr/bin/env bash

# Mise
#MISE description="Configure the registered GitLab Runner and pull its common images"

# Configurations
set -e -o pipefail

# Sources
# shellcheck source=lib/common.sh
source "${MISE_PROJECT_ROOT:?}/lib/common.sh"

# Prepare settings
settings_require
config="${SYSROOT}/etc/gitlab-runner/config.toml"

# Configure GitLab Runner (from the original registered configuration)
step "Configure GitLab Runner in '${config}'"
if ! sudo test -e "${config}.original"; then
  sudo cp -v "${config}" "${config}.original"
fi
sudo cp -fv "${config}" "${config}.previous"
sudo cp -fv "${config}.original" "${config}"
sudo sed -i '/^  \[runners\.cache\]/,$d' "${config}"
sudo sed -i '/^  \[runners\.custom_build_dir\]/,$d' "${config}"
sudo sed -i "s/^\(concurrent\) = .*/\1 = ${RUNNER_CONCURRENT:?}/" "${config}"
sudo tee -a "${config}" <<CONFIGURATION
  environment = [
    "disabled_DOCKER_CERT_PATH=/etc/gitlab-runner/certs",
    "DOCKER_HOST=unix://",
    "DOCKER_TLS_CERTDIR=",
    "DOCKER_TLS_VERIFY=1",
    "FF_GITLAB_REGISTRY_HELPER_IMAGE=true",
    "FF_NETWORK_PER_BUILD=true",
    "GITLAB_CI_DOCKER_HOST=true"
  ]
  [runners.custom_build_dir]
  [runners.docker]
    tls_verify = true
    image = "alpine:3"
    privileged = true
    disable_entrypoint_overwrite = false
    helper_image = "registry.gitlab.com/gitlab-org/gitlab-runner/gitlab-runner-helper:x86_64-v${VERSION_RUNNER:?}"
    oom_kill_disable = false
    disable_cache = ${RUNNER_CACHE_DISABLED:?}
    volumes = [
      "/cache",
      "/certs/client",
      "/etc/gitlab-runner/certs/registry.server.crt:/etc/gitlab-runner/certs/ca.crt:ro",
      "/etc/gitlab-runner/certs/registry.server.crt:/etc/gitlab-runner/certs/ca.pem:ro",
      "/etc/gitlab-runner/certs/gitlab.server.crt:/etc/gitlab-runner/certs/gitlab.server.crt:ro",
      "/etc/gitlab-runner/certs/registry.server.crt:/etc/gitlab-runner/certs/registry.server.crt:ro",
      "/var/run/docker.sock:/var/run/docker.sock:rw"
    ]
    shm_size = 0
    pull_policy = ["always", "if-not-present"]
    allowed_pull_policies = ["always", "if-not-present"]
    extra_hosts = [
      "${DOMAIN_GITLAB:?}:${DOMAIN_IP:?}",
      "${DOMAIN_REGISTRY:?}:${DOMAIN_IP:?}"
    ]
    network_mtu = 0
  [runners.cache]
    MaxUploadedArchiveSize = 0
```

```
CONFIGURATION

# Pull common images
step 'Pull common Docker images'
for image in alpine:3 docker:latest docker:dind "registry.gitlab.com/gitlab-org/gitlab-runner/gitlab-runner-
helper:x86_64-v${VERSION_RUNNER:?}"; do
  docker pull "${image}"
  echo ''
done

# Restart GitLab Runner
step 'Restart GitLab Runner'
sudo gitlab-runner restart
```

Cleanup runner images

- **Remove the GitLab images other than the configured versions :**

```
gitlab.local.dev runner:cleanup:images
```

Task sources: `runner:cleanup:images`

```
#!/usr/bin/env bash

# Mise
#MISE description="Remove the unused GitLab and GitLab Runner Docker images"

# Action
exec "${MISE_PROJECT_ROOT:?}/.mise/tasks/server/cleanup/images" "${@}"
```

Watch runner images

- **Install a daily cleanup timer of the built images :**

```
gitlab.local.dev runner:cleanup:timer
```

Task sources: `runner:cleanup:timer`

```
#!/usr/bin/env bash

# Mise
#MISE description="Install a daily systemd timer removing the GitLab built Docker images"

# Configurations
set -e -o pipefail

# Sources
# shellcheck source=lib/common.sh
source "${MISE_PROJECT_ROOT:?}/lib/common.sh"

# Prepare settings
settings_require
systemd="${SYSROOT}/etc/systemd/system"
sudo mkdir -p "${systemd}"

# Create service
step 'Install GitLab Runner cleanup service and timer'
sudo tee "${systemd}/gitlab-runner-cleanup.service" <<SERVICE
[Unit]
Description=Docker images cleanup

[Service]
ExecStartPre=/usr/bin/bash -cx '/usr/bin/docker images --format="{{.Repository}}:{{.Tag}}" | grep "${DOMAIN_GITLAB:?}"
| xargs -n1 /usr/bin/docker rmi || true'
ExecStart=/usr/bin/docker system prune -f
SERVICE

# Create timer
sudo tee "${systemd}/gitlab-runner-cleanup.timer" <<TIMER
[Unit]
Description=Docker images cleanup
After=docker.service

[Timer]
OnBootSec=1min
OnUnitActiveSec=24h
OnCalendar=*-*-* *:00:00
Unit=gitlab-runner-cleanup.service

[Install]
WantedBy=multi-user.target
TIMER

# Verify service and timer
sudo systemd-analyze verify "${systemd}/gitlab-runner-cleanup.service" "${systemd}/gitlab-runner-cleanup.timer"

# Enable service and timer
sudo systemctl daemon-reload
sudo systemctl enable gitlab-runner-cleanup.timer
sudo systemctl restart gitlab-runner-cleanup.timer
sudo systemctl status --no-pager gitlab-runner-cleanup.timer gitlab-runner-cleanup.service || true
```

Configure runner client

- **Configure the runner machine for GitLab access :** GitLab - Clients

gitlab.local.dev