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 - Server

Settings required

- **Configure the GitLab server machine settings :** Settings
- `GITLAB_PATH` **:** GitLab server folder, containing `docker-compose.yml`, `mise.toml` and `gitlab/`

Table of contents

- Settings required
- Server installation
- Server declaration
- Server certificates
- Server tasks
- Server backups
- Server update
- Cleanup GitLab images
- Cleanup GitLab Containers images
- Configure remote client

Server installation

- **Install the GitLab server (compose, certificates, then start) :**

```
gitlab.local.dev server:install
```

Task sources: `server:install`

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

# Mise
#MISE description="Install the GitLab server (compose, certificates and start)"

# Configurations
set -e -o pipefail

# Action
mise run -q server:compose
mise run -q server:certificates
mise run -q server:up
```

Server declaration

- **Generate the GitLab server `docker-compose.yml` and `mise.toml` tasks :**

```
gitlab.local.dev server:compose
```

Task sources: `server:compose`

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

# Mise
#MISE description="Generate the GitLab server 'docker-compose.yml' and 'mise.toml' tasks"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
gitlab_path_access
gitlab_urls
compose=$(docker_compose)

# Cleanup fixes
rm -rf ./.tmp.fixes

# Generate configuration
step "Generate '${GITLAB_PATH}/docker-compose.yml'"
cat >./docker-compose.yml <<CONFIGURATION
#
# GitLab - Server
#
# - Copyright: 2019-2026 Adrian DC
# - Maintainer: Adrian DC
# - Sources: https://gitlab.com/RadianDevCore/cloud/gitlab.local.dev
#

version: '3.8'
services:
  gitlab:
    image: gitlab/gitlab-ce:${VERSION_GITLAB:?}-ce.0
    restart: always
    container_name: gitlab
    hostname: gitlab-dev
    environment:
      - |
        GITLAB_OMNIBUS_CONFIG=
          ### BEGIN: GITLAB_OMNIBUS_CONFIG ###

          # DNS settings
          external_url "${external_url}"
          gitlab_pages['auth_redirect_uri'] = "${gitlab_pages_auth_redirect_uri}"
          pages_external_url "${pages_external_url}"
          registry_external_url "${registry_external_url}"

          # HTTPS settings
          nginx['listen_port'] = ${PORT_GITLAB:?}
          nginx['listen_https'] = true
          nginx['listen_addresses'] = ['*', '[::]']
          nginx['redirect_http_to_https'] = true
          nginx['redirect_http_to_https_port'] = 80

          # SSH settings
          gitlab_rails['gitlab_shell_ssh_port'] = ${PORT_SSH:?}

          # SSL certificates
          nginx['ssl_certificate'] = "/etc/gitlab/ssl/${DOMAIN_ROOT:?}/certificate.pem"
          nginx['ssl_certificate_key'] = "/etc/gitlab/ssl/${DOMAIN_ROOT:?}/certificate.key"
          pages_nginx['ssl_certificate'] = "/etc/gitlab/ssl/${DOMAIN_PAGES:?}.crt"
```

```
pages_nginx['ssl_certificate_key'] = "/etc/gitlab/ssl/${DOMAIN_PAGES:?}.key"
registry_nginx['ssl_certificate'] = "/etc/gitlab/ssl/${DOMAIN_ROOT:?}/certificate.pem"
registry_nginx['ssl_certificate_key'] = "/etc/gitlab/ssl/${DOMAIN_ROOT:?}/certificate.key"

# SSL settings
letsencrypt['enable'] = ${SSL_LETSENCRYPT:?}
letsencrypt['auto_renew'] = true
letsencrypt['contact_emails'] = ["${DOMAIN_CONTACT:?}"]
letsencrypt['auto_renew_hour'] = "12"
letsencrypt['auto_renew_minute'] = "30"
letsencrypt['auto_renew_day_of_month'] = "*/7"

# Admin settings
gitlab_rails['gitlab_default_theme'] = 1
gitlab_rails['incoming_email_enabled'] = false
gitlab_rails['smtp_enable'] = false

# Locale settings
gitlab_rails['time_zone'] = "${TIMEZONE:?}"

# Pages settings
gitlab_pages['enable'] = true
gitlab_pages['external_http'] = []
gitlab_pages['external_https'] = []
gitlab_pages['listen_proxy'] = "localhost:8090"
gitlab_pages['log_verbose'] = false
gitlab_pages['namespace_in_path'] = true
gitlab_pages['cert'] = "/etc/gitlab/ssl/${DOMAIN_PAGES:?}.crt"
gitlab_pages['cert_key'] = "/etc/gitlab/ssl/${DOMAIN_PAGES:?}.key"
gitlab_pages['access_control'] = true
gitlab_pages['auth_scope'] = 'read_api'
gitlab_pages['artifacts_server'] = false
gitlab_pages['artifacts_server_url'] = "${artifacts_server_url:?}/api/v4"
gitlab_pages['gitlab_server'] = "${gitlab_server:?}"
pages_nginx['enable'] = true

# Projects settings
gitlab_rails['gitlab_email_enabled'] = false
gitlab_rails['gitlab_default_projects_features_issues'] = true
gitlab_rails['gitlab_default_projects_features_merge_requests'] = true
gitlab_rails['gitlab_default_projects_features_wiki'] = true
gitlab_rails['gitlab_default_projects_features_snippets'] = false
gitlab_rails['gitlab_default_projects_features_builds'] = true
gitlab_rails['gitlab_default_projects_features_container_registry'] = false

# Performance settings
gitlab_rails['max_request_duration_seconds'] = 570
puma['min_threads'] = 2
puma['max_threads'] = 3
puma['per_worker_max_memory_mb'] = 250
puma['worker_timeout'] = 600
puma['worker_processes'] = 2
sidekiq['concurrency'] = 9

# Privacy settings
gitlab_rails['usage_ping_enabled'] = false
prometheus_monitoring['enable'] = false
sidekiq['metrics_enabled'] = false

# Storage settings
gitlab_rails['artifacts_enabled'] = true
gitlab_rails['lfs_enabled'] = true

# Registry settings
```

```
          gitlab_rails['registry_enabled'] = true
          registry['enable'] = true

          # User settings
          gitlab_rails['gitlab_username_changing_enabled'] = false

          ### END: GITLAB_OMNIBUS_CONFIG ###
      - DOMAIN_IP=${DOMAIN_IP:?}
      - DOMAIN_ROOT=${DOMAIN_ROOT:?}
      - TZ=${TIMEZONE:?}
    logging:
      driver: json-file
      options:
        max-size: '50m'
        max-file: '7'
    networks:
      bridge:
    ports:
      - ${PORT_SSH:?}:22/tcp
      - ${PORT_GITLAB:?}:${PORT_GITLAB:?}/tcp
      - ${PORT_REGISTRY:?}:${PORT_REGISTRY:?}/tcp
      - ${PORT_PAGES:?}:${PORT_PAGES:?}/tcp
    volumes:
      - ./gitlab/config:/etc/gitlab
      - ./gitlab/data:/var/opt/gitlab
      - ./gitlab/logs:/var/log/gitlab

networks:
  bridge:
CONFIGURATION
cat ./docker-compose.yml

# Generate server tasks
step "Generate '${GITLAB_PATH}/mise.toml'"
cat >./mise.toml <<TASKS
#
# GitLab - Server
#
# - Generated by: gitlab.local.dev server:compose
# - Sources: https://gitlab.com/RadianDevCore/cloud/gitlab.local.dev
# - Usage: mise run [TASK] (default: status)
#

[shell_alias]
make = 'mise run'

[tasks.default]
description = 'Show GitLab services'
depends = ['status']

[tasks.backup]
description = "Backup GitLab data into './gitlab.backup/'"
run = '''
if ! ${compose} run --entrypoint rsync --rm -v "\${PWD}:\${PWD}" -w "\${PWD}" gitlab -av --delete ./gitlab/
./gitlab.backup/; then
  sudo rsync -av --delete ./gitlab/ ./gitlab.backup/
fi
'''

[tasks.down]
description = 'Stop GitLab services'
run = [
  "echo ''",
  '${compose} down',
```

```
  "echo ''",
]

[tasks.logs]
description = 'Follow GitLab services logs (last 200 lines first, extra arguments are forwarded)'
run = '${compose} logs -f --tail=200'

[tasks.restart]
description = 'Restart GitLab services and follow logs for 120 seconds'
run = [
  "echo ''",
  '${compose} down',
  "echo ''",
  '${compose} up -d --remove-orphans',
  "echo ''",
  '${compose} ps -a',
  "echo ''",
  'timeout --foreground --kill-after=10 120 ${compose} logs -f --tail=100 || true',
  "echo ''",
]

[tasks.status]
description = 'Show GitLab services'
run = [
  "echo ''",
  '${compose} ps -a',
  "echo ''",
]

[tasks.up]
description = 'Start GitLab services'
run = [
  "echo ''",
  '${compose} up -d --remove-orphans',
  "echo ''",
  '${compose} ps -a',
  "echo ''",
]
TASKS
mise trust -q ./mise.toml
mise tasks ls -C "${GITLAB_PATH}"
echo ''
```

Server certificates

- **Generate the GitLab server SSL certificates (renewed 45 days before expiration) :**

```
gitlab.local.dev server:certificates
```

Task sources: `server:certificates`

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

# Mise
#MISE description="Generate or renew the GitLab server SSL certificates"
#MISE flag "--refresh-ssl" env="REFRESH_SSL" help="Regenerate all SSL certificates"
#MISE flag "--refresh-dns" env="REFRESH_DNS" help="Reconfigure the GitLab Pages DNS authentication"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
gitlab_path_access
ssl="./gitlab/config/ssl/${DOMAIN_ROOT:?}"

# Remove old certificates
if [ "${usage_refresh_ssl:-${REFRESH_SSL:-false}}" = 'true' ]; then
  step 'Remove old SSL certificates'
  sudo rm -rfv ./gitlab/config/ssl
fi

# Ownership handlings
for folder in ./gitlab/config/ssl ./gitlab/config/trusted-certs; do
  if [ ! -w "${folder}" ]; then
    sudo mkdir -p "${folder}/"
    sudo chown -Rv --reference "${HOME}" "${folder}"
  fi
done

# SSL certificates
if [ "${SSL_LETSENCRYPT:?}" != 'true' ]; then
  step "Generate SSL certificates in '${GITLAB_PATH}/${ssl#./}'"

  # Access settings (IP)
  if [ "${ACCESS_MODE:?}" = 'ip' ]; then
    subject_alt_name_append=",IP:${DOMAIN_IP:?}"
  else
    subject_alt_name_append=''
  fi

  # Drop expiring certificates
  if [ -e "${ssl}/ca.pem" ] && ! openssl x509 -checkend $((45 * 24 * 3600)) -noout -in "${ssl}/ca.pem"; then
    rm -fv "${ssl:?}/ca.key" "${ssl}/ca.pem" "${ssl}/certificate.csr" "${ssl}/certificate.pem"
  fi

  # Generate CA key
  mkdir -p "${ssl}/"
  if [ ! -e "${ssl}/ca.key" ]; then
    openssl genrsa -out "${ssl}/ca.key" 4096
  fi

  # Generate CA certificate
  if [ ! -e "${ssl}/ca.pem" ]; then
    openssl req \
      -new \
      -x509 \
      -days 365 \
      -key "${ssl}/ca.key" \
      -subj "/${DOMAIN_LOCATION:?}/CN=*.${DOMAIN_ROOT:?}/OU=${DOMAIN_ROOT:?}" \
      -out "${ssl}/ca.pem"
```

```
  fi

  # Generate user certificates
  if [ ! -e "${ssl}/certificate.pem" ]; then
    openssl req \
      -newkey rsa:4096 \
      -nodes -keyout "${ssl}/certificate.key" \
      -subj "/${DOMAIN_LOCATION:?}/CN=*.${DOMAIN_ROOT:?}/OU=${DOMAIN_ROOT:?}" \
      -out "${ssl}/certificate.csr"
    printf '%s'
"subjectAltName=DNS:${DOMAIN_ROOT:?},DNS:*.${DOMAIN_ROOT:?},DNS:${DOMAIN_GITLAB:?},DNS:${DOMAIN_PAGES:?},DNS:*.${DOMAIN
_PAGES:?},DNS:${DOMAIN_REGISTRY:?}${subject_alt_name_append}" \
      >"${ssl}/certificate.conf"
    openssl x509 \
      -req \
      -days 365 \
      -extfile "${ssl}/certificate.conf" \
      -in "${ssl}/certificate.csr" \
      -CA "${ssl}/ca.pem" \
      -CAkey "${ssl}/ca.key" \
      -CAcreateserial \
      -out "${ssl}/certificate.pem"
  fi

  # Trust certificates
  mkdir -p ./gitlab/config/trusted-certs/
  rm -f ./gitlab/config/trusted-certs/*
  cp -f "${ssl}/certificate.pem" "./gitlab/config/trusted-certs/${DOMAIN_ROOT:?}.pem"

  # Show certificates
  echo ''
  ls -lR "${ssl}/"
  echo ''
  ls -lR ./gitlab/config/trusted-certs/
  echo ''
  openssl x509 -text -noout -in "${ssl}/certificate.pem" | head -n 10
  echo ''
fi

# Pages settings
if [ -e "${ssl}/certificate.pem" ]; then
  cp -f "${ssl}/certificate.pem" "./gitlab/config/ssl/${DOMAIN_PAGES:?}.crt"
  cp -f "${ssl}/certificate.key" "./gitlab/config/ssl/${DOMAIN_PAGES:?}.key"
fi
if [ "${usage_refresh_dns:-${REFRESH_DNS:-false}}" = 'true' ] && [ -e ./gitlab/config/gitlab-secrets.json ]; then
  step 'Reset GitLab Pages DNS authentication secrets'
  sudo sed -i '/^  "gitlab_pages": {/,/^  },/d' ./gitlab/config/gitlab-secrets.json
fi
```

Server tasks

- **GitLab server folder tasks, generated by `server:compose` in `GITLAB_PATH` :**

Task	From the GitLab server folder	Description

`gitlab.local.dev server:status`	`mise run` or `mise run status`	Show the GitLab services
`gitlab.local.dev server:up`	`mise run up`	Start the GitLab services
`gitlab.local.dev server:down`	`mise run down`	Stop the GitLab services
`gitlab.local.dev server:restart`	`mise run restart`	Restart the GitLab services, follow logs for 120 seconds
`gitlab.local.dev server:logs`	`mise run logs`	Follow the GitLab services logs (last 200 lines first)
`gitlab.local.dev server:backup`	`mise run backup`	Backup the GitLab data into `./gitlab.backup/`

Generated GitLab server folder `mise.toml`

```
cat >./mise.toml <<TASKS
#
# GitLab - Server
#
# - Generated by: gitlab.local.dev server:compose
# - Sources: https://gitlab.com/RadianDevCore/cloud/gitlab.local.dev
# - Usage: mise run [TASK] (default: status)
#

[shell_alias]
make = 'mise run'

[tasks.default]
description = 'Show GitLab services'
depends = ['status']

[tasks.backup]
description = "Backup GitLab data into './gitlab.backup/'"
run = '''
if ! ${compose} run --entrypoint rsync --rm -v "\${PWD}:\${PWD}" -w "\${PWD}" gitlab -av --delete ./gitlab/
./gitlab.backup/; then
  sudo rsync -av --delete ./gitlab/ ./gitlab.backup/
fi
'''

[tasks.down]
description = 'Stop GitLab services'
run = [
  "echo ''",
  '${compose} down',
  "echo ''",
]

[tasks.logs]
description = 'Follow GitLab services logs (last 200 lines first, extra arguments are forwarded)'
run = '${compose} logs -f --tail=200'

[tasks.restart]
description = 'Restart GitLab services and follow logs for 120 seconds'
run = [
  "echo ''",
  '${compose} down',
  "echo ''",
  '${compose} up -d --remove-orphans',
  "echo ''",
  '${compose} ps -a',
  "echo ''",
  'timeout --foreground --kill-after=10 120 ${compose} logs -f --tail=100 || true',
  "echo ''",
]

[tasks.status]
description = 'Show GitLab services'
run = [
  "echo ''",
  '${compose} ps -a',
  "echo ''",
]

[tasks.up]
description = 'Start GitLab services'
run = [
  "echo ''",
  '${compose} up -d --remove-orphans',
```

```
  "echo ''",
  '${compose} ps -a',
  "echo ''",
]
TASKS
```

Server backups

- **Backup the GitLab data into `./gitlab.backup/` (2 passes) :**

```
gitlab.local.dev server:backup
```

Task sources: `server:backup`

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

# Mise
#MISE description="Backup the GitLab server data into './gitlab.backup/' (2 passes)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
gitlab_path_access

# Backup data (second pass to catch the changes of the first long pass)
step "Backup GitLab data of '${GITLAB_PATH}'"
mise run -q backup
mise run -q backup
```

Server update

- **Configure the new versions :** `VERSION_GITLAB` and `VERSION_RUNNER` settings

```
gitlab.local.dev settings:configure
```

- **Update the GitLab server (compose, certificates, backup if `BACKUP_GITLAB_UPDATE`, then restart) :**

```
gitlab.local.dev server:update
```

Task sources: `server:update`

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

# Mise
#MISE description="Update the GitLab server to the configured settings (compose, certificates, backup and restart)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require

# Action
mise run -q server:compose
mise run -q server:certificates
if [ "${BACKUP_GITLAB_UPDATE}" != 'false' ]; then
  mise run -q server:backup
fi
mise run -q server:restart
```

Cleanup GitLab images

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

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

Task sources: `server:cleanup:images`

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

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

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require

# Remove unused images
step "Remove GitLab images other than '${VERSION_GITLAB}' and '${VERSION_RUNNER}'"
docker images --format='{{.Repository}}:{{.Tag}}' \
  | grep '^gitlab/\|^registry\.gitlab\.com/gitlab-org/gitlab-runner' \
  | grep -v "${VERSION_GITLAB}\|${VERSION_RUNNER}" \
  | xargs -n1 -r docker rmi \
  || true
docker system prune -f
```

Cleanup GitLab Containers images

- **Remove the untagged GitLab Container Registry images :**

```
gitlab.local.dev server:cleanup:registry
```

Task sources: `server:cleanup:registry`

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

# Mise
#MISE description="Remove the untagged GitLab Container Registry images (garbage collection)"

# Configurations
set -e -o pipefail

# Action
docker exec gitlab gitlab-ctl registry-garbage-collect --delete-manifests --delete-untagged
```

Configure remote client

- **Configure the remote client computer for GitLab access :** GitLab - Clients

gitlab.local.dev