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`

▸ Usage



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

Documentation: https://radiandevcore.gitlab.io/cloud/gitlab.local.dev

- **Copyright:** 2019-2026 Adrian DC
- **Project:** RadianDevCore - Cloud
- **Maintainer:** Adrian DC

Table of contents

- Sequences
- Installation
- Usage
- Updates
- Development
- References

Sequences

- **1. Settings :** Install `gitlab.local.dev` and configure the infrastructure settings
- **2. Virtual machines host :** Prepare the host machine running the virtual machines
- **3. GitLab server :** Create, install and configure the GitLab server
- **4. GitLab clients :** Configure the client computers accessing GitLab
- **5. GitLab Runner :** Create, install, register and configure the GitLab Runner

---

Installation

- **Dependencies:**
- `mise-en-place` : https://mise.jdx.dev/installing-mise.html
- **Install `gitlab.local.dev` globally with `mise-en-place` :**

```
mise use -g gitlab:RadianDevCore/cloud/gitlab.local.dev@latest
```

Usage

- **Configure the infrastructure settings on each machine :**

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

- **Run a task, interactively or accepting defaults with `--yes` :**

```
gitlab.local.dev [--yes] TASK [ARGUMENT...]
```

- **Show `gitlab.local.dev` options and tasks :**

```
gitlab.local.dev --help
```

- **GitLab server folder tasks (generated by `server:compose`) :**

```
cd "${GITLAB_PATH}" && mise run [backup|down|logs|restart|status|up]
```

Updates

- **Show `gitlab.local.dev` version :**

```
gitlab.local.dev --version
```

- **Update `gitlab.local.dev` to the latest release :**

```
gitlab.local.dev --update
```

Development

- **Use `gitlab.local.dev` from sources with `mise-en-place` :**

```
{
  git clone https://gitlab.com/RadianDevCore/cloud/gitlab.local.dev.git
  cd ./gitlab.local.dev/
  mise install
  mise use -g "gitlab:RadianDevCore/cloud/gitlab.local.dev@path:${PWD}"
}
```

- **Test tasks in a sandbox, without GitLab, VirtualBox or root access :**

```
mise run test:lint
mise run test:tasks
```

- **Build and test the release archive :**

```
mise run build:release
mise run test:release
```

- **Build or serve the documentation :**

```
mise run docs:build
mise run docs:serve
```

References

- gcil: Launch .gitlab-ci.yml jobs locally
- git-cliff: CHANGELOG generator
- mise-en-place: Development tools and tasks runner
- MkDocs Material: Documentation framework

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`

▸ Overview

Install `gitlab.local.dev` and configure the infrastructure settings, specific to each infrastructure.

- **Machine :** Each machine running `gitlab.local.dev` tasks

Steps

1. GitLab - Installation
2. GitLab - Settings

> **Navigation**
>
> Follow the steps in order with the **Next** button at the bottom of each page.

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`

▸ 1. GitLab - Installation

Install `gitlab.local.dev`

- **Install `mise-en-place` :** https://mise.jdx.dev/installing-mise.html

```
sudo apt update
sudo apt install -y curl git
curl https://mise.run | sh
echo 'eval "$(~/.local/bin/mise activate bash)"' >>~/.bashrc
source ~/.bashrc
```

- **Install `gitlab.local.dev` globally with `mise-en-place` :**

```
mise use -g gitlab:RadianDevCore/cloud/gitlab.local.dev@latest
```

- **Show the available tasks :**

```
gitlab.local.dev --help
```

- **Configure the settings on each machine :** Settings

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

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`

▸ 2. GitLab - Settings

Settings file

- **Configure the settings interactively, prefilled with the current values or defaults :**

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

- **Show the current settings :**

```
gitlab.local.dev settings:show
```

- **Settings file location :** `~/.config/gitlab.local.dev/settings.env` (private, `600` permissions)
- Override the location with the `GITLAB_LOCAL_DEV_SETTINGS` environment variable
- Override any setting temporarily with its environment variable, for example `VERSION_GITLAB=19.6.1`

Task sources: `settings:configure`

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

# Mise
#MISE alias="configure"
#MISE description="Configure the infrastructure settings interactively (prefilled with current values)"

# Configurations
set -e -o pipefail

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

# Load current settings
settings_load
root_previous="${DOMAIN_ROOT}"

# Header
step "Configure settings in '${SETTINGS_FILE}'"
info 'GitLab releases: https://about.gitlab.com/releases/categories/releases/'
info 'GitLab upgrade path: https://gitlab-com.gitlab.io/support/toolbox/upgrade-path/'
info 'GitLab images: https://hub.docker.com/r/gitlab/gitlab-ce/tags'
info 'GitLab Runner releases: https://gitlab.com/gitlab-org/gitlab-runner/-/releases'
echo '' >&2

# Configure settings
for name in "${SETTINGS_NAMES[@]}"; do
  value="${!name}"

  # Follow the root domain changes for derived settings
  if [ "${name}" != 'DOMAIN_ROOT' ] && [ "${DOMAIN_ROOT}" != "${root_previous}" ] \
      && [ "${value}" = "$(DOMAIN_ROOT="${root_previous}" settings_default "${name}")" ]; then
    value=$(settings_default "${name}")
  fi

  # Show running versions
  case "${name}" in
    VERSION_GITLAB)
      if running=$(docker inspect --format '{{ .Config.Image }}' gitlab 2>/dev/null); then
        info "Running GitLab container image: ${running}"
      fi
      ;;
    VERSION_RUNNER)
      if running=$(gitlab-runner --version 2>/dev/null | sed -n 's/^Version: *//p') && [ -n "${running}" ]; then
        info "Installed GitLab Runner version: ${running}"
      fi
      ;;
    GITLAB_PATH)
      if running=$(gitlab_path_detect) && [ -n "${running}" ]; then
        info "Running GitLab container folder: ${running}"
      fi
      ;;
  esac

  # Ask setting value
  while true; do
    value=$(input "$(settings_description "${name}") [${name}]" "${value}")
    if settings_validate "${name}" "${value}"; then
      break
    fi
    warn "Invalid '${name}' value: '${value}'"
    if ! interactive; then
      exit 1
    fi
```

```
  done
  printf -v "${name}" '%s' "${value}"
done

# Write settings
mkdir -p "$(dirname "${SETTINGS_FILE}")"
temporary=$(mktemp "${SETTINGS_FILE}.XXXXXX")
{
  echo '# gitlab.local.dev settings (generated by settings:configure)'
  for name in "${SETTINGS_NAMES[@]}"; do
    printf 'export %s=%q\n' "${name}" "${!name}"
  done
} >"${temporary}"
chmod 600 "${temporary}"
mv -f "${temporary}" "${SETTINGS_FILE}"

# Result
echo '' >&2
info "Settings saved in '${SETTINGS_FILE}'"
"${MISE_PROJECT_ROOT}/.mise/tasks/settings/show"
```

Table of contents

- Settings file
- Available settings
- GitLab versions
- Recommended configurations

Available settings

Variable	Value	Defaults or examples
DOMAIN_IP	The GitLab remote IP address	`10.16.0.20`
DOMAIN_ROOT	The remote root domain name	`local.dev`
DOMAIN_GITLAB	The GitLab remote domain name	`gitlab.${DOMAIN_ROOT}`
DOMAIN_PAGES	The GitLab Pages remote domain name	`pages.${DOMAIN_ROOT}`
DOMAIN_REGISTRY	The GitLab Container Registry remote domain name	`registry.${DOMAIN_ROOT}`
DOMAIN_CONTACT	The domain public contact email	`contact@${DOMAIN_ROOT}`
DOMAIN_LOCATION	The domain public "C/ST/O/L" location	`C=FR/ST=France/O=GitLab/L=Paris`
PORT_GITLAB	The GitLab HTTPS remote port number	`8343` or `443`
PORT_PAGES	The GitLab Pages remote port number	`8390` or `8090`
PORT_REGISTRY	The GitLab Container Registry remote port number	`8350` or `5000`
PORT_SSH	The GitLab SSH remote port number	`8322` or `22`
ACCESS_MODE	The network mode (IP address, /etc/hosts or DNS)	`ip`, `hosts` or `dns`
RUNNER_CACHE_DISABLED	The GitLab Runner cache deactivation	`false` or `true`
RUNNER_CONCURRENT	The GitLab Runner maximum concurrent jobs number	`4` or `1`
SSL_LETSENCRYPT	The optional Let's Encrypt certificates activation	`false` or `true`

VERSION_GITLAB	The GitLab release version to run	`19.6.1`
VERSION_RUNNER	The GitLab Runner release version to run	`19.6.0`
VERSION_VIRTUALBOX	The VirtualBox major release version (host machines)	`7.0`
BACKUP_GITLAB_UPDATE	The GitLab data backup upon server updates	`true` or `false`
GITLAB_PATH	The GitLab server folder (empty to detect from container)	`/home/administrator/gitlab`

- **One-time server certificates flags :**
- `gitlab.local.dev server:certificates --refresh-ssl` : Regenerate SSL certificates (or `REFRESH_SSL=true`)
- `gitlab.local.dev server:certificates --refresh-dns` : Reconfigure DNS authentication (or `REFRESH_DNS=true`)

GitLab versions

- **Check GitLab releases :** GitLab Releases
- **Check GitLab upgrade path :** GitLab Upgrade Path tool
- **Check GitLab images :** DockerHub gitlab/gitlab-ce
- **Check GitLab Runner releases :** GitLab Runner releases

> ⚠ **Required upgrade stops**
>
> GitLab upgrades must go through each required upgrade stop (for example `18.5`, `18.8`, `18.11`, `19.2`, `19.5`), with `gitlab.local.dev server:update` for each stop, and background migrations finished in `Admin Area / Monitoring / Background migrations` before the next stop.

Recommended configurations

- **Machines sizes :**
- **GitLab server :** `128 GB`
- **GitLab Runner :** `48 GB`
- **Host machine configurations :**
- **VirtualBox VMs location :** `/home/services/gitlab.local.dev`

- **Default network configurations :**
- **Host machine :** `10.16.0.1`
- **GitLab server :** `10.16.0.20`
- **GitLab Runner :** `10.16.0.21`
- **Ubuntu installation configurations :**
- **Encrypt the LVM group with LUKS :** Enable
- **Passphrase :** Specific to project
- **Confirm passphrase :** Specific to project

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`

▸ Overview

Prepare the host machine running the GitLab server and GitLab Runner virtual machines.

- **Machine :** Host machine running the VirtualBox machines

Steps

1. GitLab - Ubuntu
2. GitLab - Installation
3. GitLab - Hosts

Navigation

Follow the steps in order with the **Next** button at the bottom of each page.

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`

▸ 1. GitLab - Ubuntu

Installation

Ubuntu Server 24.04

- **Welcome :**
- **Language :** English (UK)
- **Action :** Enter
- **Installer update available (Optional) :**
- **Action :** Continue without updating
- **Keyboard configuration :**
- **Layout :** French
- **Variant :** French
- **Action :** Done
- **Choose type of install :**
- **Base for the installation :** Ubuntu Server (minimized)
- **Action :** Done
- **Network connections :**
- **Second interface :** Edit IPv4, IPv4 Method: Manual, Subnet, Address, Save
  - **Example for host machines :** `192.168.0.0/24`, `192.168.0.42`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
  - **Example for GitLab Server :** `10.16.0.0/24`, `10.16.0.20`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
  - **Example for GitLab Runner :** `10.16.0.0/24`, `10.16.0.21`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
- **Action :** Done or Continue without network
- **Configure proxy :**
- **Action :** Done
- **Configure Ubuntu archive mirror :**
- **Action :** Done, Continue if asked
- **Guided storage configuration :**
- **Storage layout :** Use an entire disk
- **Set up this disk as an LVM group :** Enable
- **Encrypt the LVM group with LUKS :** Disable (Enable if encryption required)
  - **Passphrase :** `...` (If encryption required)
  - **Confirm passphrase :** `...` (If encryption required)
  - **Also create a recovery key :** Disable

- **Action :** Done
- **Storage configuration :**
  - **Used devices/ubuntu-lv (Optional) :**
  - **Action :** Edit
  - **Name :** `ubuntu-lv-HOSTNAME` (Adapt with the machine's name)
  - **Size :** Maximum size (shown in `(max ...)`, , ,
  - **Action :** Save
  - **Action :** Done, Continue
- **Profile setup :**
- **Your name :** `Administrator`
- **Your server's name :** `HOSTNAME`
- **Pick a username :** `administrator`
- **Choose a password :** `...`
- **Confirm your password :** `...`
- **Action :** Done
- **Upgrade to Ubuntu Pro :**
- **Skip for now :** Enable
- **Action :** Continue
- **SSH Setup :**
- **Install OpenSSH server :** Enable
- **Action :** Done
- **Featured Server Snaps :**
- **Action :** Done
- **Install complete! :**
- **Action :** Reboot Now
- **Please remove the installation medium :**
- **Action :** Remove installation medium, Enter

Table of contents

- Installation
  - Ubuntu Server 24.04
- Network access
- System preparation

- APT
  - Configurations
  - Upgrades
  - Packages
  - Automated cleanups
  - Resize filesystem
  - Configure timezone
- Services
  - Disable suspend features
  - SSH
  - NetworkManager
- Optimizations
  - Swappiness
  - ZRAM
  - RAM temporary filesystem
  - Disable Swap RAM space
- Security
  - Firewall
- Applications
  - Docker
- System
  - Reboot

Network access

- **Connect Internet access :** Bridged Interface or Host-only Adapter
- **Access machine remotely through SSH :** `ssh administrator@SPECIFIC_IP`

System preparation

- **Prepare the system with all recommended tasks below (optional ones excluded) :**

```
gitlab.local.dev system:install
```

Task sources: `system:install`

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

# Mise
#MISE description="Prepare the Ubuntu system (all recommended system tasks, optional ones excluded)"

# Configurations
set -e -o pipefail

# Action
for task in \
  system:apt:configure \
  system:apt:upgrade \
  system:apt:packages \
  system:timezone \
  system:suspend:disable \
  system:ssh:enable \
  system:networkmanager:install \
  system:swappiness \
  system:tmpfs \
  system:firewall \
  system:docker:install; do
  mise run -q "${task}"
done
```

APT

Configurations

- **Disable the kernel restart hints :**

```
gitlab.local.dev system:apt:configure
```

Task sources: `system:apt:configure`

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

# Mise
#MISE description="Configure APT to skip the kernel restart hints"

# Configurations
set -e -o pipefail

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

# Action
step "Configure 'needrestart' kernel hints"
sudo mkdir -p "${SYSROOT}/etc/needrestart/conf.d"
echo '$nrconf{kernelhints} = 0;' | sudo tee "${SYSROOT}/etc/needrestart/conf.d/kernel_silent.conf"
```

Upgrades

- **Upgrade the system packages :**

```
gitlab.local.dev system:apt:upgrade
```

Task sources: `system:apt:upgrade`

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

# Mise
#MISE description="Upgrade the APT packages"

# Configurations
set -e -o pipefail

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

# Action
step 'Upgrade APT packages'
sudo apt-get update
sudo NEEDRESTART_MODE=a apt-get upgrade -y
```

Packages

- **Install the common packages :**

```
gitlab.local.dev system:apt:packages
```

Task sources: `system:apt:packages`

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

# Mise
#MISE description="Install the common APT packages"

# Configurations
set -e -o pipefail

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

# Action
step 'Install common APT packages'
apt_install \
  avahi-utils \
  byobu \
  dos2unix \
  entr \
  ethtool \
  htop \
  hwinfo \
  ifmetric \
  iftop \
  inotify-tools \
  iotop \
  keychain \
  make \
  nano \
  openssh-server \
  pipx \
  pv \
  python-is-python3 \
  python3-pip \
  rsync \
  screenfetch \
  tig \
  traceroute
```

Automated cleanups

- **Upgrade and cleanup the packages and journal logs :**

```
gitlab.local.dev system:apt:cleanup
```

Task sources: `system:apt:cleanup`

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

# Mise
#MISE description="Upgrade and cleanup the APT packages and journal logs"

# Configurations
set -e -o pipefail

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

# Action
step 'Cleanup APT packages and journal logs'
sudo apt-get update
sudo NEEDRESTART_MODE=a apt-get upgrade -y
sudo NEEDRESTART_MODE=a apt-get autoremove -y
sudo apt-get autoclean
sudo journalctl --vacuum-size=50M
```

Resize filesystem

- **Resize the LVM volume and filesystem to the full disk :**

```
gitlab.local.dev system:filesystem:resize
```

Task sources: `system:filesystem:resize`

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

# Mise
#MISE description="Resize the LVM logical volume and filesystem to the full disk"

# Configurations
set -e -o pipefail

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

# Action
step 'Resize LVM logical volume and filesystem'
for volume in /dev/mapper/ubuntu--vg-*; do
  sudo lvresize -v -l +100%FREE "${volume}" || true
  sudo resize2fs -p "${volume}"
done
```

Configure timezone

- **Configure the system timezone (asked, default: `Europe/Paris`) :**

```
gitlab.local.dev system:timezone
```

Task sources: `system:timezone`

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

# Mise
#MISE description="Configure the system timezone"
#MISE arg "[timezone]" help="System timezone (asked if missing, default: Europe/Paris)"

# Configurations
set -e -o pipefail

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

# Action
step 'Configure the system timezone'
timezone="${usage_timezone:-$(input 'Local timezone' 'Europe/Paris')}"
sudo timedatectl set-timezone "${timezone:?}"
timedatectl
```

Services

Disable suspend features

- **Disable the suspend features :**

```
gitlab.local.dev system:suspend:disable
```

Task sources: `system:suspend:disable`

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

# Mise
#MISE description="Disable the system suspend features"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable suspend features'
sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target
```

SSH

- **Enable the SSH server :**

```
gitlab.local.dev system:ssh:enable
```

Task sources: `system:ssh:enable`

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

# Mise
#MISE description="Enable the SSH server"

# Configurations
set -e -o pipefail

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

# Action
step 'Enable SSH server'
sudo systemctl enable --now ssh
```

- **Disable the SSH password authentication :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:ssh:passwords:disable
```

Task sources: `system:ssh:passwords:disable`

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

# Mise
#MISE description="Disable the SSH password authentication (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable SSH password authentication'
sudo mkdir -p "${SYSROOT}/etc/ssh/sshd_config.d"
printf '%s\n' 'PubkeyAuthentication yes' 'PasswordAuthentication no' \
  | sudo tee "${SYSROOT}/etc/ssh/sshd_config.d/50-password-authentication.conf"
sudo systemctl restart ssh
```

NetworkManager

- **Install NetworkManager and manage all network interfaces :**

```
gitlab.local.dev system:networkmanager:install
```

Task sources: `system:networkmanager:install`

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

# Mise
#MISE description="Install NetworkManager and manage all network interfaces"

# Configurations
set -e -o pipefail

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

# Install NetworkManager
step 'Install NetworkManager'
sudo apt-get update
apt_install network-manager
sudo systemctl enable --now NetworkManager

# Manage network interfaces
step 'Manage network interfaces with NetworkManager'
sudo mkdir -p "${SYSROOT}/etc/NetworkManager/conf.d"
sudo rm -f "${SYSROOT}/etc/NetworkManager/conf.d/10-globally-managed-devices.conf"
sudo touch "${SYSROOT}/etc/NetworkManager/conf.d/10-globally-managed-devices.conf"
sudo systemctl restart NetworkManager
```

Optimizations

Swappiness

- **Decrease the swap usage :**

```
gitlab.local.dev system:swappiness
```

Task sources: `system:swappiness`

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

# Mise
#MISE description="Decrease the swap usage (swappiness 10)"

# Configurations
set -e -o pipefail

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

# Action
step 'Configure swappiness'
printf '\n# Decrease swap usage to a more reasonable level\nvm.swappiness = 10\n' \
  | sudo tee "${SYSROOT}/etc/sysctl.d/50-swappiness.conf"
sudo sysctl -p "${SYSROOT}/etc/sysctl.d/50-swappiness.conf"
```

ZRAM

- **Install ZRAM :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:zram
```

Task sources: `system:zram`

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

# Mise
#MISE description="Install the ZRAM compressed swap (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Install ZRAM'
apt_install zram-config
```

RAM temporary filesystem

- **Mount /tmp as a RAM temporary filesystem :**

```
gitlab.local.dev system:tmpfs
```

Task sources: `system:tmpfs`

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

# Mise
#MISE description="Mount /tmp as a RAM temporary filesystem"

# Configurations
set -e -o pipefail

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

# Action
step 'Enable RAM temporary filesystem'
sudo cp -v "${SYSROOT}/usr/share/systemd/tmp.mount" "${SYSROOT}/etc/systemd/system/"
sudo systemctl enable tmp.mount
```

Disable Swap RAM space

- **Disable the swap file :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:swap:disable
```

Task sources: `system:swap:disable`

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

# Mise
#MISE description="Disable the swap file (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable swap file'
if [ -e "${SYSROOT}/swap.img" ]; then
  sudo swapoff "${SYSROOT}/swap.img"
  sudo rm -fv "${SYSROOT}/swap.img"
fi
sudo sed -i 's|^\(/swap\.img\)|#\1|g' "${SYSROOT}/etc/fstab"
free -h
```

Security

Firewall

- **Install the persistent firewall, allowing ICMP and SSH :**

```
gitlab.local.dev system:firewall
```

Task sources: `system:firewall`

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

# Mise
#MISE description="Install the persistent firewall and allow ICMP and SSH"

# Configurations
set -e -o pipefail

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

# Install
step 'Install persistent firewall'
echo 'iptables-persistent iptables-persistent/autosave_v4 boolean false' | sudo debconf-set-selections
echo 'iptables-persistent iptables-persistent/autosave_v6 boolean false' | sudo debconf-set-selections
apt_install iptables-persistent

# Configure (idempotent rules)
step 'Configure firewall rules'
for rule in '-p icmp -j ACCEPT' '-p tcp --dport 22 -s 0/0 -j ACCEPT'; do
  # shellcheck disable=SC2086
  sudo iptables -C INPUT ${rule} 2>/dev/null || sudo iptables -A INPUT ${rule}
done
sudo iptables -L
sudo netfilter-persistent save
sudo netfilter-persistent reload
```

Applications

Docker

- **Install Docker :**

```
gitlab.local.dev system:docker:install
```

Task sources: `system:docker:install`

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

# Mise
#MISE description="Install Docker from the official Docker repository"

# Configurations
set -e -o pipefail

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

# Install dependencies
step 'Install Docker'
apt_install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common

# Install repository
sudo rm -f "${SYSROOT}/usr/share/keyrings/docker-archive-keyring.gpg"
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | sudo gpg --dearmor -o "${SYSROOT}/usr/share/keyrings/docker-archive-keyring.gpg"
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
  | sudo tee "${SYSROOT}/etc/apt/sources.list.d/docker.list"

# Install Docker
sudo apt-get update
apt_install \
  docker-ce \
  docker-compose-plugin

# Configure Docker rights (https://docs.docker.com/engine/install/linux-postinstall/)
sudo usermod -aG docker "${USER}"

# Test Docker
sudo docker run --rm --name test alpine:latest cat /etc/os-release
```

System

Reboot

- **Reboot the system :**

```
gitlab.local.dev system:reboot
```

Task sources: `system:reboot`

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

# Mise
#MISE description="Reboot the system"

# Configurations
set -e -o pipefail

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

# Action
confirm 'Reboot the system now ?'
sudo reboot
```

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`

▸ 2. GitLab - Installation

Install `gitlab.local.dev`

- **Install `mise-en-place` :** https://mise.jdx.dev/installing-mise.html

```
sudo apt update
sudo apt install -y curl git
curl https://mise.run | sh
echo 'eval "$(~/.local/bin/mise activate bash)"' >>~/.bashrc
source ~/.bashrc
```

- **Install `gitlab.local.dev` globally with `mise-en-place` :**

```
mise use -g gitlab:RadianDevCore/cloud/gitlab.local.dev@latest
```

- **Show the available tasks :**

```
gitlab.local.dev --help
```

- **Configure the settings on each machine :** Settings

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

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`

▸ 3. GitLab - Hosts

Installations

VirtualBox

Install VirtualBox

- **Install VirtualBox (`VERSION_VIRTUALBOX` setting) and its matching Extension Pack :**

```
gitlab.local.dev host:virtualbox:install
```

Task sources: `host:virtualbox:install`

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

# Mise
#MISE description="Install VirtualBox and its Extension Pack on the host machine"

# 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

# Install VirtualBox (Secure Boot asks a MOK password to enroll upon next reboot)
step "Install VirtualBox ${VERSION_VIRTUALBOX}"
wget -q -O- https://www.virtualbox.org/download/oracle_vbox_2016.asc \
  | sudo gpg --dearmor --yes --output "${SYSROOT}/usr/share/keyrings/oracle-virtualbox-2016.gpg"
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/oracle-virtualbox-2016.gpg]
http://download.virtualbox.org/virtualbox/debian $(lsb_release -sc) contrib" \
  | sudo tee "${SYSROOT}/etc/apt/sources.list.d/virtualbox.list"
sudo apt-get update
apt_install "virtualbox-${VERSION_VIRTUALBOX}"

# Configure VirtualBox
sudo usermod -G vboxusers -a "${USER}"

# Install VirtualBox Extension Pack (matching the installed version)
version=$(VBoxManage --version 2>/dev/null | tail -n1 | sed 's/[_r].*//')
step "Install VirtualBox Extension Pack ${version}"
extpack="Oracle_VM_VirtualBox_Extension_Pack-${version}.vbox-extpack"
wget -O "${temporary}/${extpack}" "https://download.virtualbox.org/virtualbox/${version}/${extpack}"
sudo VBoxManage extpack uninstall 'Oracle VM VirtualBox Extension Pack' || true
sudo VBoxManage extpack install --replace "${temporary}/${extpack}"
```

Secure Boot

- **Enter a password for Secure Boot :** `12345678`, `12345678`, Ok
- **Do you agree to these license terms and conditions (y/n)? :** `y`
- **Upon next reboot :** Enroll MOK, enter the same password

Configure VirtualBox

- **Preferences / General :**
- **Default Machine Folder :** `/home/services/gitlab.local.dev` (or `~/Data/VMs`)

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`

▸ Overview

Create, install and configure the GitLab server.

- **Machine :** GitLab server virtual machine (VirtualBox steps on the host machine)

Steps

1. GitLab - VirtualBox
2. GitLab - Ubuntu
3. GitLab - Installation
4. GitLab - Server
5. GitLab - Clients
6. GitLab - Configurations
7. GitLab - Users

Navigation

Follow the steps in order with the **Next** button at the bottom of each page.

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`

▸ 1. GitLab - VirtualBox

VirtualBox 7

Host machine

These steps run on the host machine running the VirtualBox machines.

Create virtual network

- **Create the virtual host-only network only once per infrastructure :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev host:virtualbox:network
```

Task sources: `host:virtualbox:network`

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

# Mise
#MISE description="Create the VirtualBox host-only network (once per infrastructure, optional)"

# Configurations
set -e -o pipefail

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

# User configurations
step 'Configure the VirtualBox host-only network'
interface=$(input 'Host-only Adapter network interface' 'vboxnet1')
network_subnet=$(input 'IP subnet and mask for the virtual network' '10.16.0.0/24')
network_ip=$(input 'Host IP address for the virtual network' '10.16.0.1')
network_mask=$(input 'IP mask for the virtual network' '255.255.255.0')

# VirtualBox interfaces
vbox_hostonlyif_index="${interface//[^0-9]/}"
for _ in $(seq 0 "${vbox_hostonlyif_index}"); do
  if ! sudo VBoxManage list hostonlyifs | grep '^Name:' | tr -d ' \t' | cut -d':' -f2 | grep -qx "${interface}"; then
    info 'Creating network interface...'
    interface=$(sudo VBoxManage hostonlyif create | grep 'Interface' | cut -d"'" -f2)
    info "Created network interface name: ${interface}"
  else
    break
  fi
done

# VirtualBox configurations
networks="${SYSROOT}/etc/vbox/networks.conf"
sudo mkdir -p "$(dirname "${networks}")"
if [ ! -e "${networks}" ]; then
  printf '* 192.168.%s.0/24\n' $(seq 56 63) | sudo tee "${networks}" >/dev/null
fi
if ! grep -qxF "* ${network_subnet}" "${networks}"; then
  echo "* ${network_subnet}" | sudo tee -a "${networks}" >/dev/null
fi
sudo sed -i '/^$/d' "${networks}"

# Interface configurations
VBoxManage hostonlyif ipconfig "${interface}" --ip="${network_ip}" --netmask="${network_mask}"
VBoxManage hostonlyif ipconfig "${interface}" --ipv6="fe80::800:27ff:fe00:${vbox_hostonlyif_index}" --
netmasklengthv6='64'
```

Table of contents

- VirtualBox 7
  - Create virtual network
  - Create virtual machine
- Network restrictions
- Autostart VirtualBox VM

Create virtual machine

- **Action :** New
- **Virtual machine Name and Operating System :**
- **Name :** `GitLab - Server` or `GitLab - Runner` for example
- **Type :** Linux
- **Version :** Ubuntu (64-bit)
- **Action :** Next
- **Hardware :**
- **Base Memory :** 50% minimum
- **Processors :** 50% minimum
- **Enable EFI (special OSes only) :** Enable
- **Action :** Next
- **Virtual Hard disk :**
- **Create a Virtual Hard Disk Now :** `50,00 GB` or more
- **Action :** Next
- **Summary :**
- **Action :** Finish
- **Settings :**
- **System / Motherboard :**
  - **Floppy :** Disable
- **System / Processor :**
  - **Enable PAE/NX :** Enable
  - **Enable Nested VT-x/AMD-V :** Enable (if available)
- **Storage / Controller: IDE :**
  - **Action :** Remove Controller
- **Storage / Controller: SATA :**
  - **Use Host I/O Cache :** Enable
  - **Action :** Adds optical Drive, Leave Empty
- **Storage / Controller: SATA / *.vdi :**
  - **Solid-state Drive :** Enable
- **Network / Adapter 1 :**
  - **Enable Network Adapter :** Enable

Copyright © 2026 - Adrian DC

  - **Attached to :** Bridged Adapter (or Not attached if Internal Network only)
  - **Name :** ... (specific network interface)
  - **Advanced / Promiscuous Mode :** Deny
  - **Advanced / Cable Connected :** Enable (or disable if external / Internet access is prohibited)
- **Network / Adapter 2 (optional) :**
  - **Enable Network Adapter :** Enable
  - **Attached to :** Host-only Adapter
  - **Name :** vboxnet1 (adapt per infrastructure)
  - **Advanced / Promiscuous Mode :** Deny
  - **Advanced / Cable Connected :** Enable
- **Network / Adapter 3 (optional) :**
  - **Enable Network Adapter :** Enable
  - **Attached to :** NAT
  - **Advanced / Cable Connected :** Enable (or disable if Internet access is prohibited)
- **Action :** OK

Network restrictions

- **Disconnect Internet access (if required) :** Bridged Interface and Host-only Adapter
- **Access machine locally through SSH :** `ssh administrator@HOSTNAME.local`

Autostart VirtualBox VM

- **Start the VirtualBox VM automatically from the host machine :**

> **Optional**
>
> This section is optional, use if needed.

```
gitlab.local.dev host:virtualbox:autostart
```

> **Task sources:** `host:virtualbox:autostart`

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

# Mise
#MISE description="Start a VirtualBox VM automatically with a systemd user service (optional)"
#MISE arg "[vm]" help="VirtualBox VM name (asked if missing, defaults to the latest VM)"

# Configurations
set -e -o pipefail

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

# List VMs
step 'Detected VirtualBox machines'
VBoxManage list vms | cut -d'"' -f2 | sed 's/^/ - /'
vm_latest=$(VBoxManage list vms | cut -d'"' -f2 | tail -n1)

# User choice
echo '' >&2
vm_name="${usage_vm:-$(input 'Autostart VirtualBox VM name' "${vm_latest}")}"
[ -n "${vm_name}" ] || fail 'Missing VirtualBox VM name'

# Configure VM
VBoxManage controlvm "${vm_name}" poweroff 2>/dev/null || true
VBoxManage modifyvm "${vm_name}" --defaultfrontend separate

# Autostart VM
vm_service=$(echo "virtualbox-${vm_name,,}" | sed -e 's/ /-/g; s/--*/-/g')
services="${XDG_CONFIG_HOME:-${HOME}/.config}/systemd/user"
mkdir -p "${services}"
cat >"${services}/${vm_service}.service" <<SERVICE
[Unit]
Description=Start VirtualBox VM for ${vm_name}
After=network.target vboxdrv.service
Before=shutdown.target

[Service]
ExecStart=/usr/bin/VBoxManage startvm "${vm_name}" --type headless
ExecStop=/usr/bin/VBoxManage controlvm "${vm_name}" acpipowerbutton
ExecStopPost=/bin/sleep 10
GuessMainPID=no
IgnoreSIGPIPE=no
KillMode=process
RemainAfterExit=yes
Restart=on-failure
Type=forking

[Install]
WantedBy=default.target
SERVICE
systemctl --user daemon-reload
systemctl --user enable "${vm_service}"
systemctl --user restart "${vm_service}"
systemctl --user status --no-pager "${vm_service}"

# Autostart user services
if ! loginctl show-user "${USER}" | grep -q 'Linger=yes'; then
  sudo loginctl enable-linger "${USER}"
fi
```

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`

▸ 2. GitLab - Ubuntu

Installation

Ubuntu Server 24.04

- **Welcome :**
- **Language :** English (UK)
- **Action :** Enter
- **Installer update available (Optional) :**
- **Action :** Continue without updating
- **Keyboard configuration :**
- **Layout :** French
- **Variant :** French
- **Action :** Done
- **Choose type of install :**
- **Base for the installation :** Ubuntu Server (minimized)
- **Action :** Done
- **Network connections :**
- **Second interface :** Edit IPv4, IPv4 Method: Manual, Subnet, Address, Save
  - **Example for host machines :** `192.168.0.0/24`, `192.168.0.42`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
  - **Example for GitLab Server :** `10.16.0.0/24`, `10.16.0.20`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
  - **Example for GitLab Runner :** `10.16.0.0/24`, `10.16.0.21`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
- **Action :** Done or Continue without network
- **Configure proxy :**
- **Action :** Done
- **Configure Ubuntu archive mirror :**
- **Action :** Done, Continue if asked
- **Guided storage configuration :**
- **Storage layout :** Use an entire disk
- **Set up this disk as an LVM group :** Enable
- **Encrypt the LVM group with LUKS :** Disable (Enable if encryption required)
  - **Passphrase :** `...` (If encryption required)
  - **Confirm passphrase :** `...` (If encryption required)
  - **Also create a recovery key :** Disable

- **Action :** Done
- **Storage configuration :**
  - **Used devices/ubuntu-lv (Optional) :**
  - **Action :** Edit
  - **Name :** `ubuntu-lv-HOSTNAME` (Adapt with the machine's name)
  - **Size :** Maximum size (shown in `(max ...)`, , ,
  - **Action :** Save
  - **Action :** Done, Continue
- **Profile setup :**
- **Your name :** `Administrator`
- **Your server's name :** `HOSTNAME`
- **Pick a username :** `administrator`
- **Choose a password :** `...`
- **Confirm your password :** `...`
- **Action :** Done
- **Upgrade to Ubuntu Pro :**
- **Skip for now :** Enable
- **Action :** Continue
- **SSH Setup :**
- **Install OpenSSH server :** Enable
- **Action :** Done
- **Featured Server Snaps :**
- **Action :** Done
- **Install complete! :**
- **Action :** Reboot Now
- **Please remove the installation medium :**
- **Action :** Remove installation medium, Enter

Table of contents

- Installation
  - Ubuntu Server 24.04
- Network access
- System preparation

- APT
  - Configurations
  - Upgrades
  - Packages
  - Automated cleanups
  - Resize filesystem
  - Configure timezone
- Services
  - Disable suspend features
  - SSH
  - NetworkManager
- Optimizations
  - Swappiness
  - ZRAM
  - RAM temporary filesystem
  - Disable Swap RAM space
- Security
  - Firewall
- Applications
  - Docker
- System
  - Reboot

Network access

- **Connect Internet access :** Bridged Interface or Host-only Adapter
- **Access machine remotely through SSH :** `ssh administrator@SPECIFIC_IP`

System preparation

- **Prepare the system with all recommended tasks below (optional ones excluded) :**

```
gitlab.local.dev system:install
```

Task sources: `system:install`

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

# Mise
#MISE description="Prepare the Ubuntu system (all recommended system tasks, optional ones excluded)"

# Configurations
set -e -o pipefail

# Action
for task in \
  system:apt:configure \
  system:apt:upgrade \
  system:apt:packages \
  system:timezone \
  system:suspend:disable \
  system:ssh:enable \
  system:networkmanager:install \
  system:swappiness \
  system:tmpfs \
  system:firewall \
  system:docker:install; do
  mise run -q "${task}"
done
```

APT

Configurations

- **Disable the kernel restart hints :**

```
gitlab.local.dev system:apt:configure
```

Task sources: `system:apt:configure`

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

# Mise
#MISE description="Configure APT to skip the kernel restart hints"

# Configurations
set -e -o pipefail

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

# Action
step "Configure 'needrestart' kernel hints"
sudo mkdir -p "${SYSROOT}/etc/needrestart/conf.d"
echo '$nrconf{kernelhints} = 0;' | sudo tee "${SYSROOT}/etc/needrestart/conf.d/kernel_silent.conf"
```

Upgrades

- **Upgrade the system packages :**

```
gitlab.local.dev system:apt:upgrade
```

Task sources: `system:apt:upgrade`

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

# Mise
#MISE description="Upgrade the APT packages"

# Configurations
set -e -o pipefail

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

# Action
step 'Upgrade APT packages'
sudo apt-get update
sudo NEEDRESTART_MODE=a apt-get upgrade -y
```

Packages

- **Install the common packages :**

```
gitlab.local.dev system:apt:packages
```

Task sources: `system:apt:packages`

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

# Mise
#MISE description="Install the common APT packages"

# Configurations
set -e -o pipefail

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

# Action
step 'Install common APT packages'
apt_install \
  avahi-utils \
  byobu \
  dos2unix \
  entr \
  ethtool \
  htop \
  hwinfo \
  ifmetric \
  iftop \
  inotify-tools \
  iotop \
  keychain \
  make \
  nano \
  openssh-server \
  pipx \
  pv \
  python-is-python3 \
  python3-pip \
  rsync \
  screenfetch \
  tig \
  traceroute
```

Automated cleanups

- **Upgrade and cleanup the packages and journal logs :**

```
gitlab.local.dev system:apt:cleanup
```

Task sources: `system:apt:cleanup`

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

# Mise
#MISE description="Upgrade and cleanup the APT packages and journal logs"

# Configurations
set -e -o pipefail

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

# Action
step 'Cleanup APT packages and journal logs'
sudo apt-get update
sudo NEEDRESTART_MODE=a apt-get upgrade -y
sudo NEEDRESTART_MODE=a apt-get autoremove -y
sudo apt-get autoclean
sudo journalctl --vacuum-size=50M
```

Resize filesystem

- **Resize the LVM volume and filesystem to the full disk :**

```
gitlab.local.dev system:filesystem:resize
```

Task sources: `system:filesystem:resize`

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

# Mise
#MISE description="Resize the LVM logical volume and filesystem to the full disk"

# Configurations
set -e -o pipefail

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

# Action
step 'Resize LVM logical volume and filesystem'
for volume in /dev/mapper/ubuntu--vg-*; do
  sudo lvresize -v -l +100%FREE "${volume}" || true
  sudo resize2fs -p "${volume}"
done
```

Configure timezone

- **Configure the system timezone (asked, default: `Europe/Paris`) :**

```
gitlab.local.dev system:timezone
```

Task sources: `system:timezone`

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

# Mise
#MISE description="Configure the system timezone"
#MISE arg "[timezone]" help="System timezone (asked if missing, default: Europe/Paris)"

# Configurations
set -e -o pipefail

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

# Action
step 'Configure the system timezone'
timezone="${usage_timezone:-$(input 'Local timezone' 'Europe/Paris')}"
sudo timedatectl set-timezone "${timezone:?}"
timedatectl
```

Services

Disable suspend features

- **Disable the suspend features :**

```
gitlab.local.dev system:suspend:disable
```

Task sources: `system:suspend:disable`

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

# Mise
#MISE description="Disable the system suspend features"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable suspend features'
sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target
```

SSH

- **Enable the SSH server :**

```
gitlab.local.dev system:ssh:enable
```

Task sources: `system:ssh:enable`

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

# Mise
#MISE description="Enable the SSH server"

# Configurations
set -e -o pipefail

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

# Action
step 'Enable SSH server'
sudo systemctl enable --now ssh
```

- **Disable the SSH password authentication :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:ssh:passwords:disable
```

Task sources: `system:ssh:passwords:disable`

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

# Mise
#MISE description="Disable the SSH password authentication (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable SSH password authentication'
sudo mkdir -p "${SYSROOT}/etc/ssh/sshd_config.d"
printf '%s\n' 'PubkeyAuthentication yes' 'PasswordAuthentication no' \
  | sudo tee "${SYSROOT}/etc/ssh/sshd_config.d/50-password-authentication.conf"
sudo systemctl restart ssh
```

NetworkManager

- **Install NetworkManager and manage all network interfaces :**

```
gitlab.local.dev system:networkmanager:install
```

Task sources: `system:networkmanager:install`

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

# Mise
#MISE description="Install NetworkManager and manage all network interfaces"

# Configurations
set -e -o pipefail

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

# Install NetworkManager
step 'Install NetworkManager'
sudo apt-get update
apt_install network-manager
sudo systemctl enable --now NetworkManager

# Manage network interfaces
step 'Manage network interfaces with NetworkManager'
sudo mkdir -p "${SYSROOT}/etc/NetworkManager/conf.d"
sudo rm -f "${SYSROOT}/etc/NetworkManager/conf.d/10-globally-managed-devices.conf"
sudo touch "${SYSROOT}/etc/NetworkManager/conf.d/10-globally-managed-devices.conf"
sudo systemctl restart NetworkManager
```

Optimizations

Swappiness

- **Decrease the swap usage :**

```
gitlab.local.dev system:swappiness
```

Task sources: `system:swappiness`

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

# Mise
#MISE description="Decrease the swap usage (swappiness 10)"

# Configurations
set -e -o pipefail

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

# Action
step 'Configure swappiness'
printf '\n# Decrease swap usage to a more reasonable level\nvm.swappiness = 10\n' \
  | sudo tee "${SYSROOT}/etc/sysctl.d/50-swappiness.conf"
sudo sysctl -p "${SYSROOT}/etc/sysctl.d/50-swappiness.conf"
```

ZRAM

- **Install ZRAM :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:zram
```

Task sources: `system:zram`

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

# Mise
#MISE description="Install the ZRAM compressed swap (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Install ZRAM'
apt_install zram-config
```

RAM temporary filesystem

- **Mount /tmp as a RAM temporary filesystem :**

```
gitlab.local.dev system:tmpfs
```

Task sources: `system:tmpfs`

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

# Mise
#MISE description="Mount /tmp as a RAM temporary filesystem"

# Configurations
set -e -o pipefail

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

# Action
step 'Enable RAM temporary filesystem'
sudo cp -v "${SYSROOT}/usr/share/systemd/tmp.mount" "${SYSROOT}/etc/systemd/system/"
sudo systemctl enable tmp.mount
```

Disable Swap RAM space

- **Disable the swap file :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:swap:disable
```

Task sources: `system:swap:disable`

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

# Mise
#MISE description="Disable the swap file (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable swap file'
if [ -e "${SYSROOT}/swap.img" ]; then
  sudo swapoff "${SYSROOT}/swap.img"
  sudo rm -fv "${SYSROOT}/swap.img"
fi
sudo sed -i 's|^\(/swap\.img\)|#\1|g' "${SYSROOT}/etc/fstab"
free -h
```

Security

Firewall

- **Install the persistent firewall, allowing ICMP and SSH :**

```
gitlab.local.dev system:firewall
```

Task sources: `system:firewall`

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

# Mise
#MISE description="Install the persistent firewall and allow ICMP and SSH"

# Configurations
set -e -o pipefail

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

# Install
step 'Install persistent firewall'
echo 'iptables-persistent iptables-persistent/autosave_v4 boolean false' | sudo debconf-set-selections
echo 'iptables-persistent iptables-persistent/autosave_v6 boolean false' | sudo debconf-set-selections
apt_install iptables-persistent

# Configure (idempotent rules)
step 'Configure firewall rules'
for rule in '-p icmp -j ACCEPT' '-p tcp --dport 22 -s 0/0 -j ACCEPT'; do
  # shellcheck disable=SC2086
  sudo iptables -C INPUT ${rule} 2>/dev/null || sudo iptables -A INPUT ${rule}
done
sudo iptables -L
sudo netfilter-persistent save
sudo netfilter-persistent reload
```

Applications

Docker

- **Install Docker :**

```
gitlab.local.dev system:docker:install
```

Task sources: `system:docker:install`

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

# Mise
#MISE description="Install Docker from the official Docker repository"

# Configurations
set -e -o pipefail

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

# Install dependencies
step 'Install Docker'
apt_install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common

# Install repository
sudo rm -f "${SYSROOT}/usr/share/keyrings/docker-archive-keyring.gpg"
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | sudo gpg --dearmor -o "${SYSROOT}/usr/share/keyrings/docker-archive-keyring.gpg"
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
  | sudo tee "${SYSROOT}/etc/apt/sources.list.d/docker.list"

# Install Docker
sudo apt-get update
apt_install \
  docker-ce \
  docker-compose-plugin

# Configure Docker rights (https://docs.docker.com/engine/install/linux-postinstall/)
sudo usermod -aG docker "${USER}"

# Test Docker
sudo docker run --rm --name test alpine:latest cat /etc/os-release
```

System

Reboot

- **Reboot the system :**

```
gitlab.local.dev system:reboot
```

Task sources: `system:reboot`

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

# Mise
#MISE description="Reboot the system"

# Configurations
set -e -o pipefail

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

# Action
confirm 'Reboot the system now ?'
sudo reboot
```

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`

▸ 3. GitLab - Installation

Install `gitlab.local.dev`

- **Install `mise-en-place` :** https://mise.jdx.dev/installing-mise.html

```
sudo apt update
sudo apt install -y curl git
curl https://mise.run | sh
echo 'eval "$(~/.local/bin/mise activate bash)"' >>~/.bashrc
source ~/.bashrc
```

- **Install `gitlab.local.dev` globally with `mise-en-place` :**

```
mise use -g gitlab:RadianDevCore/cloud/gitlab.local.dev@latest
```

- **Show the available tasks :**

```
gitlab.local.dev --help
```

- **Configure the settings on each machine :** Settings

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

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

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`

▸ 5. GitLab - Clients

Settings required

- **Configure the client machine settings :** Settings

Table of contents

- Settings required
- Configure GitLab system clients
- Configure GitLab user clients

Configure GitLab system clients

- **Configure the system hosts and certificates (as administrator) :**

```
gitlab.local.dev clients:system
```

Task sources: `clients:system`

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

# Mise
#MISE description="Configure the client system hosts and trusted GitLab certificates (as administrator)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
sudo -v

# Install dependencies
if ! type certutil >/dev/null 2>&1; then
  sudo apt-get update
  sudo apt-get install -y libnss3-tools
fi

# 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:?}
${DOMAIN_IP:?} projects.${DOMAIN_PAGES:?}
${DOMAIN_IP:?} ${DOMAIN_REGISTRY:?}"

# Acquire certificates
certificates="${SYSROOT}/usr/local/share/ca-certificates"
if [ -e "${certificates}" ]; then
  for server in "${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}" "${DOMAIN_PAGES:?}:${PORT_PAGES:?}"
"${DOMAIN_REGISTRY:?}:${PORT_REGISTRY:?}"; do
    domain="${server%:*}"
    step "Acquire certificate from ${server}"
    sudo rm -fv "${SYSROOT}/etc/ssl/certs/${domain:?}.crt"
    certificate_acquire "${server}" "${domain}" | sudo tee "${certificates}/${domain}.crt" >/dev/null
    test -s "${certificates}/${domain}.crt" || fail "Empty certificate from ${server}"
  done
fi

# Refresh CA certificates
if sudo which update-ca-certificates >/dev/null 2>&1; then
  sudo update-ca-certificates --fresh | grep -v '^Replacing\|Mono' || true
fi

# Test GitLab certificate
step "Test https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}"
curl -sS -o /dev/null -w ' HTTP status: %{http_code}\n' "https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}"

# Restart Docker service
if systemctl cat docker.service >/dev/null 2>&1; then
  confirm 'Restart the Docker service ?'
  sudo systemctl restart docker
fi
```

Configure GitLab user clients

- **Configure the user certificates of Chrome and Firefox (as user) :**

```
gitlab.local.dev clients:user
```

Task sources: `clients:user`

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

# Mise
#MISE description="Trust the GitLab certificates in the user browsers (Chrome NSS and Firefox, as user)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
certificates="${SYSROOT}/usr/local/share/ca-certificates"

# Import certificates in a NSS database
nss_import() {
  local database="${1:?}"
  local trust="${2:?}"
  local certificate_file certificate_name
  for certificate_file in "${certificates}"/*.crt; do
    [ -e "${certificate_file}" ] || continue
    certificate_name="${certificate_file##*/}"
    certificate_name="${certificate_name%.crt}"
    certutil -d "${database}" -D -n "${certificate_name}" 2>/dev/null | grep -v '^Notice:' || true
    certutil -d "${database}" -A -t "${trust}" -n "${certificate_name}" -i "${certificate_file}" | grep -v '^Notice:'
|| true
  done
}

# Refresh user certificates
if [ -e "${HOME}/.pki/nssdb" ]; then
  step 'Refresh user NSS certificates'
  nss_import "sql:${HOME}/.pki/nssdb" 'CP,CP,'
fi

# Refresh Firefox certificates
while IFS= read -r profile_db; do
  step "Refresh Firefox certificates of '$(dirname "${profile_db}")'"
  nss_import "$(dirname "${profile_db}")" 'TCu,Cuw,Tuw'
done < <(find ~/.mozilla/ -name 'cert*.db' 2>/dev/null || true)

# Test browsers
for browser in firefox google-chrome; do
  if type "${browser}" >/dev/null 2>&1 && interactive; then
    "${browser}" "https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}" >/dev/null 2>&1 &
    sleep 1
  fi
done
```

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`

▸ 6. GitLab - Configurations

Prepare GitLab password

- **Show the initial `root` password, from the GitLab server machine :**

```
gitlab.local.dev server:password
```

Task sources: `server:password`

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

# Mise
#MISE description="Show the GitLab initial 'root' password (valid 24 hours after installation)"

# Configurations
set -e -o pipefail

# Action
docker exec gitlab cat /etc/gitlab/initial_root_password
```

- **Initial GitLab access :**
- **Open :** `https://DOMAIN_GITLAB:PORT_GITLAB`
- **Username :** `root`
- **Password :** Paste the initial password seen above

Table of contents

- Prepare GitLab password
- Prepare GitLab settings
  - Admin Area / Settings / General
  - Admin Area / Overview / Users
  - Admin Area / Settings / CI/CD
  - Admin Area / Settings / Metrics and profiling
  - Admin Area / Settings / Preferences
  - Account / Edit profile
  - Admin Area / Overview / Users (repeat)
  - Account / Sign out
  - Authentication (repeat)
  - Welcome to GitLab

Prepare GitLab settings

Admin Area / Settings / General

- **Visibility and access controls :**
- **Restricted visibility levels :**
  - **Internal :** Enable
  - **Public :** Enable
  - **Action :** Save changes
- **Import and export settings :**
- **Repository by URL :** Enable
- **GitLab export :** Enable
- **Action :** Save changes
- **Sign-up restrictions :**
- **Sign-up enabled :** Disable
- **Require admin approval for new sign-ups :** Enable
- **Email confirmation settings :** Off
- **Action :** Save changes
- **Sign-in restrictions :**
- **Admin Mode / Enable Admin Mode :** Enable
- **Action :** Save changes

Admin Area / Overview / Users

- **New user :**
- **Name :** `GitLab CI`
- **Username :** `gitlab-ci`
- **Email :** `CONTACT_EMAIL@DOMAIN`
- **Access level :** Regular
- **Avatar :** `GITLAB_AVATAR.png`
- **Action :** Create user
- **Groups :** Add 'gitlab-ci' as 'Reporter' to all projects using GitLab CI/CD or GitLab Container Registry

Admin Area / Settings / CI/CD

- **Admin Area / Overview / Users / gitlab-ci / Impersonation Tokens :**
- **Action :** Add new token
  - **Token name :** `runner-HOSTNAME-registry`
  - **Expiration date :** `...-12-31` (current year's last day or less)
  - **Select scopes :** read_registry
  - **Action :** Create impersonation token
  - **Action :** Copy impersonation token
- **Generate the Docker authentication configuration from the Runner machine (impersonation token as password) :**

```
gitlab.local.dev configurations:docker-auth
```

Task sources: `configurations:docker-auth`

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

# Mise
#MISE description="Generate the 'DOCKER_AUTH_CONFIG' CI/CD variable value for the 'gitlab-ci' user"
#MISE arg "[user]" default="gitlab-ci" help="GitLab user owning the 'read_registry' impersonation token"

# 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

# Generate credentials (impersonation token as password)
step "Login to ${DOMAIN_REGISTRY:?}:${PORT_REGISTRY:?}"
docker --config "${temporary}" login -u "${usage_user:-gitlab-ci}" "${DOMAIN_REGISTRY:?}:${PORT_REGISTRY:?}"

# Show configuration
step "'DOCKER_AUTH_CONFIG' value"
cat "${temporary}/config.json"
echo ''
```

- **Admin Area / Settings / CI/CD / Variables :**
- **Action :** Add variable
  - **Type :** Variable (default)
  - **Protect variable :** Disable
  - **Mask variable :** Disable

- **Expand variable reference :** Disable
- **Description :** Empty
- **Key :** `DOCKER_AUTH_CONFIG`
- **Value :** Paste previously generated Docker authentication configuration, similar to:

```
{
        "auths": {
                "DOMAIN_REGISTRY:PORT_REGISTRY": {
                        "auth": "REGISTRY_AUTH"
                }
        }
}
```

- **Action :** Add variable

- **Admin Area / Settings / CI/CD / Continuous Integration and Deployment :**
- **Maximum artifacts size (MB) :** `5000`
- **Default artifacts expiration :** `60 minutes`
- **Action :** Save changes
- **Admin Area / Settings / CI/CD / Container Registry :**
- **Enable cleanup policies for projects created earlier than GitLab 12.7 :** Enable
- **Action :** Save changes

Admin Area / Settings / Metrics and profiling

- **Metrics - Prometheus :**
- **Enable GitLab Prometheus metrics endpoint :** Disable
- **Action :** Save changes
- **Usage statistics :**
- **Enable version check :** Disable
- **Enable Service Ping :** Disable
- **Action :** Save changes

Admin Area / Settings / Preferences

- **Pages :**
- **Maximum size of pages (MiB) :** `500`
- **Action :** Save changes
- **Localization :**
- **Default first day of the week :** Monday

- **Action :** Save changes

Account / Edit profile

- **Upload new avatar :**
- **Sources path :** `GITLAB_AVATAR.png`
- **Action :** Fully unzoom, Set new profile picture
- **Action :** Update profile settings

Admin Area / Overview / Users (repeat)

- **New user :**
- **Name :** `Firstname LASTNAME`
- **Username :** `flastname`
- **Email :** `USER_EMAIL@DOMAIN`
- **Access level :** Regular or Administrator
- **Avatar :** Optional
- **Action :** Create user
- **Edit :**
- **Password :** `A1z2e3r4t5y6`
- **Password confirmation :** `A1z2e3r4t5y6`
- **Action :** Save changes

Account / Sign out

Authentication (repeat)

- **Username :** `flastname`
- **Password :** `A1z2e3r4t5y6`
- **Set up new password :**
- **Current password :** `A1z2e3r4t5y6`
- **New password :** `NEW_PASSWORD`
- **Password confirmation :** `NEW_PASSWORD`
- **Action :** Set new password
- **Information :** Password successfully changed

Welcome to GitLab

- **Create a group :** `...`
- **Create a project :** `...`

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`

▸ 7. GitLab - Users

Create new GitLab users

- **Admin Area / Overview / Users :**
- **New user :**
  - **Name :** `Firstname LASTNAME`
  - **Username :** `flastname`
  - **Email :** `f.lastname@email.com`
  - **Access level :** Regular or Administrator
  - **Avatar :** Optional
  - **Action :** Create user
- **Edit (repeat) :**
  - **Password :** `A1z2e3r4t5y6`
  - **Password confirmation :** `A1z2e3r4t5y6`
  - **Action :** Save changes
- **Groups / View all groups :**
- **Group information / Members / Invite members :**
  - **Username :** New user
  - **Select a role :** Role
  - **Action :** Invite

Copyright © 2026 - Adrian DC

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`

▸ Overview

Configure the client computers to access GitLab with trusted certificates.

- **Machine :** Each client computer accessing GitLab

Steps

1. GitLab - Installation
2. GitLab - Clients

> **Navigation**
>
> Follow the steps in order with the **Next** button at the bottom of each page.

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`

▸ 1. GitLab - Installation

Install `gitlab.local.dev`

- **Install `mise-en-place` :** https://mise.jdx.dev/installing-mise.html

```
sudo apt update
sudo apt install -y curl git
curl https://mise.run | sh
echo 'eval "$(~/.local/bin/mise activate bash)"' >>~/.bashrc
source ~/.bashrc
```

- **Install `gitlab.local.dev` globally with `mise-en-place` :**

```
mise use -g gitlab:RadianDevCore/cloud/gitlab.local.dev@latest
```

- **Show the available tasks :**

```
gitlab.local.dev --help
```

- **Configure the settings on each machine :** Settings

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

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`

▸ 2. GitLab - Clients

Settings required

- **Configure the client machine settings :** Settings

Table of contents

- Settings required
- Configure GitLab system clients
- Configure GitLab user clients

Configure GitLab system clients

- **Configure the system hosts and certificates (as administrator) :**

```
gitlab.local.dev clients:system
```

Task sources: `clients:system`

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

# Mise
#MISE description="Configure the client system hosts and trusted GitLab certificates (as administrator)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
sudo -v

# Install dependencies
if ! type certutil >/dev/null 2>&1; then
  sudo apt-get update
  sudo apt-get install -y libnss3-tools
fi

# 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:?}
${DOMAIN_IP:?} projects.${DOMAIN_PAGES:?}
${DOMAIN_IP:?} ${DOMAIN_REGISTRY:?}"

# Acquire certificates
certificates="${SYSROOT}/usr/local/share/ca-certificates"
if [ -e "${certificates}" ]; then
  for server in "${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}" "${DOMAIN_PAGES:?}:${PORT_PAGES:?}"
"${DOMAIN_REGISTRY:?}:${PORT_REGISTRY:?}"; do
    domain="${server%:*}"
    step "Acquire certificate from ${server}"
    sudo rm -fv "${SYSROOT}/etc/ssl/certs/${domain:?}.crt"
    certificate_acquire "${server}" "${domain}" | sudo tee "${certificates}/${domain}.crt" >/dev/null
    test -s "${certificates}/${domain}.crt" || fail "Empty certificate from ${server}"
  done
fi

# Refresh CA certificates
if sudo which update-ca-certificates >/dev/null 2>&1; then
  sudo update-ca-certificates --fresh | grep -v '^Replacing\|Mono' || true
fi

# Test GitLab certificate
step "Test https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}"
curl -sS -o /dev/null -w ' HTTP status: %{http_code}\n' "https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}"

# Restart Docker service
if systemctl cat docker.service >/dev/null 2>&1; then
  confirm 'Restart the Docker service ?'
  sudo systemctl restart docker
fi
```

Configure GitLab user clients

- **Configure the user certificates of Chrome and Firefox (as user) :**

```
gitlab.local.dev clients:user
```

Task sources: `clients:user`

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

# Mise
#MISE description="Trust the GitLab certificates in the user browsers (Chrome NSS and Firefox, as user)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
certificates="${SYSROOT}/usr/local/share/ca-certificates"

# Import certificates in a NSS database
nss_import() {
  local database="${1:?}"
  local trust="${2:?}"
  local certificate_file certificate_name
  for certificate_file in "${certificates}"/*.crt; do
    [ -e "${certificate_file}" ] || continue
    certificate_name="${certificate_file##*/}"
    certificate_name="${certificate_name%.crt}"
    certutil -d "${database}" -D -n "${certificate_name}" 2>/dev/null | grep -v '^Notice:' || true
    certutil -d "${database}" -A -t "${trust}" -n "${certificate_name}" -i "${certificate_file}" | grep -v '^Notice:'
|| true
  done
}

# Refresh user certificates
if [ -e "${HOME}/.pki/nssdb" ]; then
  step 'Refresh user NSS certificates'
  nss_import "sql:${HOME}/.pki/nssdb" 'CP,CP,'
fi

# Refresh Firefox certificates
while IFS= read -r profile_db; do
  step "Refresh Firefox certificates of '$(dirname "${profile_db}")'"
  nss_import "$(dirname "${profile_db}")" 'TCu,Cuw,Tuw'
done < <(find ~/.mozilla/ -name 'cert*.db' 2>/dev/null || true)

# Test browsers
for browser in firefox google-chrome; do
  if type "${browser}" >/dev/null 2>&1 && interactive; then
    "${browser}" "https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}" >/dev/null 2>&1 &
    sleep 1
  fi
done
```

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`

▸ Overview

Create, install, register and configure the GitLab Runner.

- **Machine :** GitLab Runner virtual machine (VirtualBox steps on the host machine)

Steps

1. GitLab - VirtualBox
2. GitLab - Ubuntu
3. GitLab - Installation
4. GitLab - Runners
5. GitLab - Clients

Navigation

Follow the steps in order with the **Next** button at the bottom of each page.

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`

▸ 1. GitLab - VirtualBox

VirtualBox 7

Host machine

These steps run on the host machine running the VirtualBox machines.

Create virtual network

- **Create the virtual host-only network only once per infrastructure :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev host:virtualbox:network
```

Task sources: `host:virtualbox:network`

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

# Mise
#MISE description="Create the VirtualBox host-only network (once per infrastructure, optional)"

# Configurations
set -e -o pipefail

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

# User configurations
step 'Configure the VirtualBox host-only network'
interface=$(input 'Host-only Adapter network interface' 'vboxnet1')
network_subnet=$(input 'IP subnet and mask for the virtual network' '10.16.0.0/24')
network_ip=$(input 'Host IP address for the virtual network' '10.16.0.1')
network_mask=$(input 'IP mask for the virtual network' '255.255.255.0')

# VirtualBox interfaces
vbox_hostonlyif_index="${interface//[^0-9]/}"
for _ in $(seq 0 "${vbox_hostonlyif_index}"); do
  if ! sudo VBoxManage list hostonlyifs | grep '^Name:' | tr -d ' \t' | cut -d':' -f2 | grep -qx "${interface}"; then
    info 'Creating network interface...'
    interface=$(sudo VBoxManage hostonlyif create | grep 'Interface' | cut -d"'" -f2)
    info "Created network interface name: ${interface}"
  else
    break
  fi
done

# VirtualBox configurations
networks="${SYSROOT}/etc/vbox/networks.conf"
sudo mkdir -p "$(dirname "${networks}")"
if [ ! -e "${networks}" ]; then
  printf '* 192.168.%s.0/24\n' $(seq 56 63) | sudo tee "${networks}" >/dev/null
fi
if ! grep -qxF "* ${network_subnet}" "${networks}"; then
  echo "* ${network_subnet}" | sudo tee -a "${networks}" >/dev/null
fi
sudo sed -i '/^$/d' "${networks}"

# Interface configurations
VBoxManage hostonlyif ipconfig "${interface}" --ip="${network_ip}" --netmask="${network_mask}"
VBoxManage hostonlyif ipconfig "${interface}" --ipv6="fe80::800:27ff:fe00:${vbox_hostonlyif_index}" --
netmasklengthv6='64'
```

Table of contents

- VirtualBox 7
  - Create virtual network
  - Create virtual machine
- Network restrictions
- Autostart VirtualBox VM

Create virtual machine

- **Action :** New
- **Virtual machine Name and Operating System :**
- **Name :** `GitLab - Server` or `GitLab - Runner` for example
- **Type :** Linux
- **Version :** Ubuntu (64-bit)
- **Action :** Next
- **Hardware :**
- **Base Memory :** 50% minimum
- **Processors :** 50% minimum
- **Enable EFI (special OSes only) :** Enable
- **Action :** Next
- **Virtual Hard disk :**
- **Create a Virtual Hard Disk Now :** `50,00 GB` or more
- **Action :** Next
- **Summary :**
- **Action :** Finish
- **Settings :**
- **System / Motherboard :**
  - **Floppy :** Disable
- **System / Processor :**
  - **Enable PAE/NX :** Enable
  - **Enable Nested VT-x/AMD-V :** Enable (if available)
- **Storage / Controller: IDE :**
  - **Action :** Remove Controller
- **Storage / Controller: SATA :**
  - **Use Host I/O Cache :** Enable
  - **Action :** Adds optical Drive, Leave Empty
- **Storage / Controller: SATA / *.vdi :**
  - **Solid-state Drive :** Enable
- **Network / Adapter 1 :**
  - **Enable Network Adapter :** Enable

Copyright © 2026 - Adrian DC

  - **Attached to :** Bridged Adapter (or Not attached if Internal Network only)
  - **Name :** ... (specific network interface)
  - **Advanced / Promiscuous Mode :** Deny
  - **Advanced / Cable Connected :** Enable (or disable if external / Internet access is prohibited)
- **Network / Adapter 2 (optional) :**
  - **Enable Network Adapter :** Enable
  - **Attached to :** Host-only Adapter
  - **Name :** vboxnet1 (adapt per infrastructure)
  - **Advanced / Promiscuous Mode :** Deny
  - **Advanced / Cable Connected :** Enable
- **Network / Adapter 3 (optional) :**
  - **Enable Network Adapter :** Enable
  - **Attached to :** NAT
  - **Advanced / Cable Connected :** Enable (or disable if Internet access is prohibited)
- **Action :** OK

Network restrictions

- **Disconnect Internet access (if required) :** Bridged Interface and Host-only Adapter
- **Access machine locally through SSH :** `ssh administrator@HOSTNAME.local`

Autostart VirtualBox VM

- **Start the VirtualBox VM automatically from the host machine :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev host:virtualbox:autostart
```

Task sources: `host:virtualbox:autostart`

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

# Mise
#MISE description="Start a VirtualBox VM automatically with a systemd user service (optional)"
#MISE arg "[vm]" help="VirtualBox VM name (asked if missing, defaults to the latest VM)"

# Configurations
set -e -o pipefail

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

# List VMs
step 'Detected VirtualBox machines'
VBoxManage list vms | cut -d'"' -f2 | sed 's/^/ - /'
vm_latest=$(VBoxManage list vms | cut -d'"' -f2 | tail -n1)

# User choice
echo '' >&2
vm_name="${usage_vm:-$(input 'Autostart VirtualBox VM name' "${vm_latest}")}"
[ -n "${vm_name}" ] || fail 'Missing VirtualBox VM name'

# Configure VM
VBoxManage controlvm "${vm_name}" poweroff 2>/dev/null || true
VBoxManage modifyvm "${vm_name}" --defaultfrontend separate

# Autostart VM
vm_service=$(echo "virtualbox-${vm_name,,}" | sed -e 's/ /-/g; s/--*/-/g')
services="${XDG_CONFIG_HOME:-${HOME}/.config}/systemd/user"
mkdir -p "${services}"
cat >"${services}/${vm_service}.service" <<SERVICE
[Unit]
Description=Start VirtualBox VM for ${vm_name}
After=network.target vboxdrv.service
Before=shutdown.target

[Service]
ExecStart=/usr/bin/VBoxManage startvm "${vm_name}" --type headless
ExecStop=/usr/bin/VBoxManage controlvm "${vm_name}" acpipowerbutton
ExecStopPost=/bin/sleep 10
GuessMainPID=no
IgnoreSIGPIPE=no
KillMode=process
RemainAfterExit=yes
Restart=on-failure
Type=forking

[Install]
WantedBy=default.target
SERVICE
systemctl --user daemon-reload
systemctl --user enable "${vm_service}"
systemctl --user restart "${vm_service}"
systemctl --user status --no-pager "${vm_service}"

# Autostart user services
if ! loginctl show-user "${USER}" | grep -q 'Linger=yes'; then
  sudo loginctl enable-linger "${USER}"
fi
```

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`

▸ 2. GitLab - Ubuntu

Installation

Ubuntu Server 24.04

- **Welcome :**
- **Language :** English (UK)
- **Action :** Enter
- **Installer update available (Optional) :**
- **Action :** Continue without updating
- **Keyboard configuration :**
- **Layout :** French
- **Variant :** French
- **Action :** Done
- **Choose type of install :**
- **Base for the installation :** Ubuntu Server (minimized)
- **Action :** Done
- **Network connections :**
- **Second interface :** Edit IPv4, IPv4 Method: Manual, Subnet, Address, Save
  - **Example for host machines :** `192.168.0.0/24`, `192.168.0.42`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
  - **Example for GitLab Server :** `10.16.0.0/24`, `10.16.0.20`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
  - **Example for GitLab Runner :** `10.16.0.0/24`, `10.16.0.21`, `[EMPTY]`, `[EMPTY]`, `[EMPTY]`
- **Action :** Done or Continue without network
- **Configure proxy :**
- **Action :** Done
- **Configure Ubuntu archive mirror :**
- **Action :** Done, Continue if asked
- **Guided storage configuration :**
- **Storage layout :** Use an entire disk
- **Set up this disk as an LVM group :** Enable
- **Encrypt the LVM group with LUKS :** Disable (Enable if encryption required)
  - **Passphrase :** `...` (If encryption required)
  - **Confirm passphrase :** `...` (If encryption required)
  - **Also create a recovery key :** Disable

- **Action :** Done
- **Storage configuration :**
  - **Used devices/ubuntu-lv (Optional) :**
  - **Action :** Edit
  - **Name :** `ubuntu-lv-HOSTNAME` (Adapt with the machine's name)
  - **Size :** Maximum size (shown in `(max ...)`, , ,
  - **Action :** Save
  - **Action :** Done, Continue
- **Profile setup :**
- **Your name :** `Administrator`
- **Your server's name :** `HOSTNAME`
- **Pick a username :** `administrator`
- **Choose a password :** `...`
- **Confirm your password :** `...`
- **Action :** Done
- **Upgrade to Ubuntu Pro :**
- **Skip for now :** Enable
- **Action :** Continue
- **SSH Setup :**
- **Install OpenSSH server :** Enable
- **Action :** Done
- **Featured Server Snaps :**
- **Action :** Done
- **Install complete! :**
- **Action :** Reboot Now
- **Please remove the installation medium :**
- **Action :** Remove installation medium, Enter

Table of contents

- Installation
  - Ubuntu Server 24.04
- Network access
- System preparation

- APT
  - Configurations
  - Upgrades
  - Packages
  - Automated cleanups
  - Resize filesystem
  - Configure timezone
- Services
  - Disable suspend features
  - SSH
  - NetworkManager
- Optimizations
  - Swappiness
  - ZRAM
  - RAM temporary filesystem
  - Disable Swap RAM space
- Security
  - Firewall
- Applications
  - Docker
- System
  - Reboot

Copyright © 2026 - Adrian DC

Network access

- **Connect Internet access :** Bridged Interface or Host-only Adapter
- **Access machine remotely through SSH :** `ssh administrator@SPECIFIC_IP`

System preparation

- **Prepare the system with all recommended tasks below (optional ones excluded) :**

```
gitlab.local.dev system:install
```

Task sources: `system:install`

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

# Mise
#MISE description="Prepare the Ubuntu system (all recommended system tasks, optional ones excluded)"

# Configurations
set -e -o pipefail

# Action
for task in \
  system:apt:configure \
  system:apt:upgrade \
  system:apt:packages \
  system:timezone \
  system:suspend:disable \
  system:ssh:enable \
  system:networkmanager:install \
  system:swappiness \
  system:tmpfs \
  system:firewall \
  system:docker:install; do
  mise run -q "${task}"
done
```

APT

Configurations

- **Disable the kernel restart hints :**

```
gitlab.local.dev system:apt:configure
```

Task sources: `system:apt:configure`

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

# Mise
#MISE description="Configure APT to skip the kernel restart hints"

# Configurations
set -e -o pipefail

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

# Action
step "Configure 'needrestart' kernel hints"
sudo mkdir -p "${SYSROOT}/etc/needrestart/conf.d"
echo '$nrconf{kernelhints} = 0;' | sudo tee "${SYSROOT}/etc/needrestart/conf.d/kernel_silent.conf"
```

Upgrades

- **Upgrade the system packages :**

```
gitlab.local.dev system:apt:upgrade
```

Task sources: `system:apt:upgrade`

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

# Mise
#MISE description="Upgrade the APT packages"

# Configurations
set -e -o pipefail

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

# Action
step 'Upgrade APT packages'
sudo apt-get update
sudo NEEDRESTART_MODE=a apt-get upgrade -y
```

Packages

- **Install the common packages :**

```
gitlab.local.dev system:apt:packages
```

Task sources: `system:apt:packages`

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

# Mise
#MISE description="Install the common APT packages"

# Configurations
set -e -o pipefail

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

# Action
step 'Install common APT packages'
apt_install \
  avahi-utils \
  byobu \
  dos2unix \
  entr \
  ethtool \
  htop \
  hwinfo \
  ifmetric \
  iftop \
  inotify-tools \
  iotop \
  keychain \
  make \
  nano \
  openssh-server \
  pipx \
  pv \
  python-is-python3 \
  python3-pip \
  rsync \
  screenfetch \
  tig \
  traceroute
```

Automated cleanups

- **Upgrade and cleanup the packages and journal logs :**

```
gitlab.local.dev system:apt:cleanup
```

Task sources: `system:apt:cleanup`

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

# Mise
#MISE description="Upgrade and cleanup the APT packages and journal logs"

# Configurations
set -e -o pipefail

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

# Action
step 'Cleanup APT packages and journal logs'
sudo apt-get update
sudo NEEDRESTART_MODE=a apt-get upgrade -y
sudo NEEDRESTART_MODE=a apt-get autoremove -y
sudo apt-get autoclean
sudo journalctl --vacuum-size=50M
```

Resize filesystem

- **Resize the LVM volume and filesystem to the full disk :**

```
gitlab.local.dev system:filesystem:resize
```

Task sources: `system:filesystem:resize`

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

# Mise
#MISE description="Resize the LVM logical volume and filesystem to the full disk"

# Configurations
set -e -o pipefail

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

# Action
step 'Resize LVM logical volume and filesystem'
for volume in /dev/mapper/ubuntu--vg-*; do
  sudo lvresize -v -l +100%FREE "${volume}" || true
  sudo resize2fs -p "${volume}"
done
```

Configure timezone

- **Configure the system timezone (asked, default: `Europe/Paris`) :**

```
gitlab.local.dev system:timezone
```

Task sources: `system:timezone`

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

# Mise
#MISE description="Configure the system timezone"
#MISE arg "[timezone]" help="System timezone (asked if missing, default: Europe/Paris)"

# Configurations
set -e -o pipefail

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

# Action
step 'Configure the system timezone'
timezone="${usage_timezone:-$(input 'Local timezone' 'Europe/Paris')}"
sudo timedatectl set-timezone "${timezone:?}"
timedatectl
```

Services

Disable suspend features

- **Disable the suspend features :**

```
gitlab.local.dev system:suspend:disable
```

Task sources: `system:suspend:disable`

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

# Mise
#MISE description="Disable the system suspend features"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable suspend features'
sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target
```

SSH

- **Enable the SSH server :**

```
gitlab.local.dev system:ssh:enable
```

Task sources: `system:ssh:enable`

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

# Mise
#MISE description="Enable the SSH server"

# Configurations
set -e -o pipefail

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

# Action
step 'Enable SSH server'
sudo systemctl enable --now ssh
```

- **Disable the SSH password authentication :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:ssh:passwords:disable
```

Task sources: `system:ssh:passwords:disable`

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

# Mise
#MISE description="Disable the SSH password authentication (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable SSH password authentication'
sudo mkdir -p "${SYSROOT}/etc/ssh/sshd_config.d"
printf '%s\n' 'PubkeyAuthentication yes' 'PasswordAuthentication no' \
  | sudo tee "${SYSROOT}/etc/ssh/sshd_config.d/50-password-authentication.conf"
sudo systemctl restart ssh
```

NetworkManager

- **Install NetworkManager and manage all network interfaces :**

```
gitlab.local.dev system:networkmanager:install
```

Task sources: `system:networkmanager:install`

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

# Mise
#MISE description="Install NetworkManager and manage all network interfaces"

# Configurations
set -e -o pipefail

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

# Install NetworkManager
step 'Install NetworkManager'
sudo apt-get update
apt_install network-manager
sudo systemctl enable --now NetworkManager

# Manage network interfaces
step 'Manage network interfaces with NetworkManager'
sudo mkdir -p "${SYSROOT}/etc/NetworkManager/conf.d"
sudo rm -f "${SYSROOT}/etc/NetworkManager/conf.d/10-globally-managed-devices.conf"
sudo touch "${SYSROOT}/etc/NetworkManager/conf.d/10-globally-managed-devices.conf"
sudo systemctl restart NetworkManager
```

Optimizations

Swappiness

- **Decrease the swap usage :**

```
gitlab.local.dev system:swappiness
```

Task sources: `system:swappiness`

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

# Mise
#MISE description="Decrease the swap usage (swappiness 10)"

# Configurations
set -e -o pipefail

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

# Action
step 'Configure swappiness'
printf '\n# Decrease swap usage to a more reasonable level\nvm.swappiness = 10\n' \
  | sudo tee "${SYSROOT}/etc/sysctl.d/50-swappiness.conf"
sudo sysctl -p "${SYSROOT}/etc/sysctl.d/50-swappiness.conf"
```

ZRAM

- **Install ZRAM :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:zram
```

Task sources: `system:zram`

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

# Mise
#MISE description="Install the ZRAM compressed swap (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Install ZRAM'
apt_install zram-config
```

RAM temporary filesystem

- **Mount /tmp as a RAM temporary filesystem :**

```
gitlab.local.dev system:tmpfs
```

Task sources: `system:tmpfs`

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

# Mise
#MISE description="Mount /tmp as a RAM temporary filesystem"

# Configurations
set -e -o pipefail

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

# Action
step 'Enable RAM temporary filesystem'
sudo cp -v "${SYSROOT}/usr/share/systemd/tmp.mount" "${SYSROOT}/etc/systemd/system/"
sudo systemctl enable tmp.mount
```

Disable Swap RAM space

- **Disable the swap file :**

Optional

This section is optional, use if needed.

```
gitlab.local.dev system:swap:disable
```

Task sources: `system:swap:disable`

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

# Mise
#MISE description="Disable the swap file (optional)"

# Configurations
set -e -o pipefail

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

# Action
step 'Disable swap file'
if [ -e "${SYSROOT}/swap.img" ]; then
  sudo swapoff "${SYSROOT}/swap.img"
  sudo rm -fv "${SYSROOT}/swap.img"
fi
sudo sed -i 's|^\(/swap\.img\)|#\1|g' "${SYSROOT}/etc/fstab"
free -h
```

Security

Firewall

- **Install the persistent firewall, allowing ICMP and SSH :**

```
gitlab.local.dev system:firewall
```

Task sources: `system:firewall`

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

# Mise
#MISE description="Install the persistent firewall and allow ICMP and SSH"

# Configurations
set -e -o pipefail

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

# Install
step 'Install persistent firewall'
echo 'iptables-persistent iptables-persistent/autosave_v4 boolean false' | sudo debconf-set-selections
echo 'iptables-persistent iptables-persistent/autosave_v6 boolean false' | sudo debconf-set-selections
apt_install iptables-persistent

# Configure (idempotent rules)
step 'Configure firewall rules'
for rule in '-p icmp -j ACCEPT' '-p tcp --dport 22 -s 0/0 -j ACCEPT'; do
  # shellcheck disable=SC2086
  sudo iptables -C INPUT ${rule} 2>/dev/null || sudo iptables -A INPUT ${rule}
done
sudo iptables -L
sudo netfilter-persistent save
sudo netfilter-persistent reload
```

Applications

Docker

- **Install Docker :**

```
gitlab.local.dev system:docker:install
```

Task sources: `system:docker:install`

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

# Mise
#MISE description="Install Docker from the official Docker repository"

# Configurations
set -e -o pipefail

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

# Install dependencies
step 'Install Docker'
apt_install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common

# Install repository
sudo rm -f "${SYSROOT}/usr/share/keyrings/docker-archive-keyring.gpg"
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | sudo gpg --dearmor -o "${SYSROOT}/usr/share/keyrings/docker-archive-keyring.gpg"
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
  | sudo tee "${SYSROOT}/etc/apt/sources.list.d/docker.list"

# Install Docker
sudo apt-get update
apt_install \
  docker-ce \
  docker-compose-plugin

# Configure Docker rights (https://docs.docker.com/engine/install/linux-postinstall/)
sudo usermod -aG docker "${USER}"

# Test Docker
sudo docker run --rm --name test alpine:latest cat /etc/os-release
```

System

Reboot

- **Reboot the system :**

```
gitlab.local.dev system:reboot
```

Task sources: `system:reboot`

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

# Mise
#MISE description="Reboot the system"

# Configurations
set -e -o pipefail

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

# Action
confirm 'Reboot the system now ?'
sudo reboot
```

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`

▸ 3. GitLab - Installation

Install `gitlab.local.dev`

- **Install `mise-en-place` :** https://mise.jdx.dev/installing-mise.html

```
sudo apt update
sudo apt install -y curl git
curl https://mise.run | sh
echo 'eval "$(~/.local/bin/mise activate bash)"' >>~/.bashrc
source ~/.bashrc
```

- **Install `gitlab.local.dev` globally with `mise-en-place` :**

```
mise use -g gitlab:RadianDevCore/cloud/gitlab.local.dev@latest
```

- **Show the available tasks :**

```
gitlab.local.dev --help
```

- **Configure the settings on each machine :** Settings

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

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

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`

▸ 5. GitLab - Clients

Settings required

- **Configure the client machine settings :** Settings

Table of contents

- Settings required
- Configure GitLab system clients
- Configure GitLab user clients

Configure GitLab system clients

- **Configure the system hosts and certificates (as administrator) :**

```
gitlab.local.dev clients:system
```

Task sources: `clients:system`

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

# Mise
#MISE description="Configure the client system hosts and trusted GitLab certificates (as administrator)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
sudo -v

# Install dependencies
if ! type certutil >/dev/null 2>&1; then
  sudo apt-get update
  sudo apt-get install -y libnss3-tools
fi

# 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:?}
${DOMAIN_IP:?} projects.${DOMAIN_PAGES:?}
${DOMAIN_IP:?} ${DOMAIN_REGISTRY:?}"

# Acquire certificates
certificates="${SYSROOT}/usr/local/share/ca-certificates"
if [ -e "${certificates}" ]; then
  for server in "${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}" "${DOMAIN_PAGES:?}:${PORT_PAGES:?}"
"${DOMAIN_REGISTRY:?}:${PORT_REGISTRY:?}"; do
    domain="${server%:*}"
    step "Acquire certificate from ${server}"
    sudo rm -fv "${SYSROOT}/etc/ssl/certs/${domain:?}.crt"
    certificate_acquire "${server}" "${domain}" | sudo tee "${certificates}/${domain}.crt" >/dev/null
    test -s "${certificates}/${domain}.crt" || fail "Empty certificate from ${server}"
  done
fi

# Refresh CA certificates
if sudo which update-ca-certificates >/dev/null 2>&1; then
  sudo update-ca-certificates --fresh | grep -v '^Replacing\|Mono' || true
fi

# Test GitLab certificate
step "Test https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}"
curl -sS -o /dev/null -w ' HTTP status: %{http_code}\n' "https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}"

# Restart Docker service
if systemctl cat docker.service >/dev/null 2>&1; then
  confirm 'Restart the Docker service ?'
  sudo systemctl restart docker
fi
```

Configure GitLab user clients

- **Configure the user certificates of Chrome and Firefox (as user) :**

```
gitlab.local.dev clients:user
```

Task sources: `clients:user`

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

# Mise
#MISE description="Trust the GitLab certificates in the user browsers (Chrome NSS and Firefox, as user)"

# Configurations
set -e -o pipefail

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

# Prepare settings
settings_require
certificates="${SYSROOT}/usr/local/share/ca-certificates"

# Import certificates in a NSS database
nss_import() {
  local database="${1:?}"
  local trust="${2:?}"
  local certificate_file certificate_name
  for certificate_file in "${certificates}"/*.crt; do
    [ -e "${certificate_file}" ] || continue
    certificate_name="${certificate_file##*/}"
    certificate_name="${certificate_name%.crt}"
    certutil -d "${database}" -D -n "${certificate_name}" 2>/dev/null | grep -v '^Notice:' || true
    certutil -d "${database}" -A -t "${trust}" -n "${certificate_name}" -i "${certificate_file}" | grep -v '^Notice:'
|| true
  done
}

# Refresh user certificates
if [ -e "${HOME}/.pki/nssdb" ]; then
  step 'Refresh user NSS certificates'
  nss_import "sql:${HOME}/.pki/nssdb" 'CP,CP,'
fi

# Refresh Firefox certificates
while IFS= read -r profile_db; do
  step "Refresh Firefox certificates of '$(dirname "${profile_db}")'"
  nss_import "$(dirname "${profile_db}")" 'TCu,Cuw,Tuw'
done < <(find ~/.mozilla/ -name 'cert*.db' 2>/dev/null || true)

# Test browsers
for browser in firefox google-chrome; do
  if type "${browser}" >/dev/null 2>&1 && interactive; then
    "${browser}" "https://${DOMAIN_GITLAB:?}:${PORT_GITLAB:?}" >/dev/null 2>&1 &
    sleep 1
  fi
done
```

gitlab.local.dev