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