Is this command in archlinux correct - I amtrying to build gimp-git and at the very end after also C compilations, I get errors that dependencies babl and gegl not found but I have already included --nodeps above

The issue you're encountering suggests a mismatch between build-time dependencies and runtime/linking dependencies. Here's what's likely happening:

**The** --nodeps **flag only skips dependency checks during package installation**, but it doesn't help with actual compilation errors. When building GIMP from git, the build process itself needs to find the babl and gegl libraries during compilation and linking.

Here are a few things to check:

1. **Install the development packages**:

bash

sudo pacman -S babl gegl

1. **If you need the latest versions**, you might need the git versions too:

bash

yay -S babl-git gegl-git

1. **Check your build environment**:
   * Make sure pkg-config can find the libraries:

bash

pkg-config --list-all | grep -E "(babl|gegl)"

* Verify library paths are correct:

bash

pkg-config --libs babl gegl

1. **If using an AUR helper**, the correct approach would be:

bash

yay -S gimp-git

This will automatically handle the dependency chain (babl-git → gegl-git → gimp-git).

The --nodeps flag is typically used when you want to force install a package despite missing dependencies, but for building from source, you actually need those dependencies present for the compilation to succeed.

What build method are you using exactly? Are you using makepkg, an AUR helper, or building manually with meson/autotools?

Claude does not have the ability to run the code it generates yet.

D

Can PKGBUILD be modified to skip these dependencies?

**Technically yes, but it's not recommended** because GIMP literally cannot function without babl and gegl - they're core libraries, not optional features.

However, if you want to modify the PKGBUILD, here's what you'd need to change:

## In the PKGBUILD file:

1. **Remove from** depends=() **array**:

bash

*# Remove or comment out these lines*

*# 'babl'*

*# 'gegl'*

1. **Remove from** makedepends=() **array**:

bash

*# Remove these if present*

*# 'babl'*

*# 'gegl'*

1. **Modify the build configuration** to skip these checks:

bash

build() {

*# Add flags to skip dependency checks*

meson setup build \

--buildtype=release \

--prefix=/usr \

-Dforce-fallback-for=babl,gegl \

*# or try:*

*# -Dbabl:required=false \*

*# -Dgegl:required=false*

}

## But here's why this is problematic:

* **GIMP won't compile** without these libraries - they provide essential image processing functions
* **Even if it compiles**, GIMP will crash immediately on startup
* **You'll break the dependency chain** for future updates

## Better alternatives:

1. **If the issue is version conflicts**:

bash

*# Install specific versions*

sudo pacman -S babl=0.1.xx gegl=0.4.xx

1. **If you want to use system libraries instead of git versions**: Edit the PKGBUILD to depend on babl and gegl instead of babl-git and gegl-git
2. **Use a different approach**:

bash

*# Install regular GIMP first, then overlay with git version*

sudo pacman -S gimp

yay -S gimp-git --overwrite

**What specific error are you getting?** The actual error message would help determine if there's a better solution than removing dependencies.

Claude does not have the ability to run the code it generates yet.

D

See the commands. I commented out but since the git was already compiled before again issuing makepkg .... did not help:

[admin@archlinux-nvme gimp-git]$ makepkg -si --nocheck --nodeps ==> Making package: gimp-git 2:3.1.5.r99.3ec63979d8-1 (Tuesday 16 September 2025 03:44:43 PM) ==> WARNING: Skipping dependency checks. ==> Retrieving sources... -> Updating gimp git repo... -> Updating gimp-data git repo... -> Found linux.gpl ==> Validating source files with sha512sums... gimp ... Skipped gimp-data ... Skipped linux.gpl ... Passed ==> Extracting sources... -> Creating working copy of gimp git repo... Reset branch 'makepkg' -> Creating working copy of gimp-data git repo... Reset branch 'makepkg' ==> Starting prepare()... ==> Starting pkgver()... ==> WARNING: A package has already been built, installing existing package... ==> Installing package gimp-git with pacman -U... [sudo] password for admin: loading packages... warning: gimp-git-debug-2:3.1.5.r99.3ec63979d8-1 is up to date -- reinstalling resolving dependencies... warning: cannot resolve "babl-git>=0.1.114", a dependency of "gimp-git" warning: cannot resolve "gegl-git>=0.4.62", a dependency of "gimp-git" :: The following package cannot be upgraded due to unresolvable dependencies: gimp-git :: Do you want to skip the above package for this upgrade? [y/N] y looking for conflicting packages... Packages (1) gimp-git-debug-2:3.1.5.r99.3ec63979d8-1 Total Installed Size: 129.53 MiB Net Upgrade Size: 0.00 MiB :: Proceed with installation? [Y/n] Y

Only debug was built.

The issue is clear now! The package was already built previously, and makepkg is trying to install the **existing package file** which still has the original dependencies baked into it. Even though you commented out the dependencies in the PKGBUILD, the .pkg.tar.xz file in your directory still contains the old dependency information.

## Solution: Force a rebuild

You need to either clean and rebuild, or force makepkg to rebuild:

**Option 1: Clean and rebuild**

bash

makepkg -C *# Clean build files*

rm \*.pkg.tar.xz\* *# Remove old package files*

makepkg -si --nocheck --nodeps

**Option 2: Force rebuild**

bash

makepkg -f -si --nocheck --nodeps

The -f flag forces makepkg to rebuild even if the package file already exists.

**Option 3: Remove just the package files**

bash

rm gimp-git-\*.pkg.tar.xz\*

makepkg -si --nocheck --nodeps

## What happened:

1. You built gimp-git before with the original PKGBUILD (with babl-git/gegl-git dependencies)
2. You modified the PKGBUILD to comment out dependencies
3. makepkg saw the existing .pkg.tar.xz file and said "package already built, let's install it"
4. But that old package file still contained the dependency metadata from the original build
5. pacman -U tried to install it and complained about missing dependencies

After forcing the rebuild, you should see it actually go through the build process again instead of jumping straight to installation.

Claude does not have the ability to run the code it generates yet.

D

That worked! Your Option 2 - Force rebuild. It it would not have then I would be starting on a clea slate - re-downloading git, etc which would have taken significant time.

Excellent! I'm glad Option 2 worked for you. The -f flag is really handy in situations like this - it's much more efficient than starting from scratch since it:

* Keeps the already-downloaded git repositories
* Preserves any intermediate build files that are still valid
* Only rebuilds what's necessary
* Saves you from re-downloading potentially hundreds of MB of source code

This is a common gotcha when modifying PKGBUILDs after an initial build. The key insight is that makepkg checks for existing package files first, and if found, assumes they match the current PKGBUILD - which obviously wasn't the case here after your dependency modifications.

For future reference, other useful makepkg flags for similar situations:

* -C (clean) removes build artifacts but keeps sources
* -c (clean after build) automatically cleans up build files when done
* -r (install build dependencies) if you need to pull in makedepends

The force rebuild approach (-f) is definitely the way to go when you've modified a PKGBUILD but want to avoid the time penalty of re-downloading large git repositories. Smart troubleshooting on your part to try the dependency modification approach!

D

First let me know how to clone pkgconfig and replace gitlab with github.

I think you're asking about how to modify a PKGBUILD that currently uses GitLab as its source to instead use GitHub. Here's how to do that:

## If you want to clone and modify an existing PKGBUILD:

**1. Clone from AUR:**

bash

git clone https://aur.archlinux.org/pkgname.git

cd pkgname

**2. Edit the PKGBUILD to change source URLs:**

bash

nano PKGBUILD

**3. Find and replace the GitLab URL with GitHub:**

bash

*# Change from:*

source=("git+https://gitlab.com/user/repo.git")

*# To:*

source=("git+https://github.com/user/repo.git")

## Example modification:

bash

*# Original GitLab source*

source=("${pkgname}::git+https://gitlab.freedesktop.org/pipewire/pipewire.git")

*# Changed to GitHub mirror*

source=("${pkgname}::git+https://github.com/PipeWire/pipewire.git")

## Update checksums if needed:

bash

updpkgsums *# This updates the checksums automatically*

## Then build:

bash

makepkg -si

**Could you clarify:**

1. What specific package are you trying to modify?
2. Are you looking to switch from GitLab to GitHub for a particular reason (faster downloads, access issues, etc.)?

This will help me give you the exact commands for your specific case.

Claude does not have the ability to run the code it generates yet.

D

But there are sed commands to replace.

You're absolutely right! Using sed commands is much more efficient for bulk replacements. Here are the common sed commands to replace GitLab with GitHub in PKGBUILDs:

## Basic URL replacement:

bash

*# Replace GitLab with GitHub in source URLs*

sed -i 's|gitlab\.com|github.com|g' PKGBUILD

*# Or more specific - replace GitLab freedesktop URLs*

sed -i 's|gitlab\.freedesktop\.org|github.com|g' PKGBUILD

## More targeted replacements:

bash

*# Replace specific GitLab project paths with GitHub equivalents*

sed -i 's|gitlab\.freedesktop\.org/gstreamer|github.com/GStreamer|g' PKGBUILD

sed -i 's|gitlab\.gnome\.org/GNOME|github.com/GNOME|g' PKGBUILD

sed -i 's|gitlab\.com/user/repo|github.com/user/repo|g' PKGBUILD

## Multiple replacements in one command:

bash

sed -i \

-e 's|gitlab\.com|github.com|g' \

-e 's|gitlab\.freedesktop\.org|github.com|g' \

-e 's|gitlab\.gnome\.org/GNOME|github.com/GNOME|g' \

PKGBUILD

## For git clone URLs specifically:

bash

*# Replace git clone URLs*

sed -i 's|git+https://gitlab\.|git+https://github.|g' PKGBUILD

## Preview changes before applying:

bash

*# See what would change (without -i flag)*

sed 's|gitlab\.com|github.com|g' PKGBUILD

## After sed replacements:

bash

updpkgsums *# Update checksums*

makepkg -si

Which specific GitLab → GitHub replacement are you looking to make?

Claude does not have the ability to run the code it generates yet.

D

Got this error at the very end in archlinux. should I install through pacman -Which is this package:

:: unable to satisfy dependency 'libart-lgpl>=2.3.19' required by gimp-git ==> WARNING: Failed to install built package(s).

The libart-lgpl package is **legacy graphics library** from the old GNOME 2 era. It's quite old and has been largely replaced by Cairo and other modern graphics libraries.

## Install the missing dependency:

bash

sudo pacman -S libart-lgpl

## If that doesn't work, try AUR:

bash

yay -S libart-lgpl

## Alternative - check if it's really needed:

Since libart-lgpl is very old, you might want to check if this dependency is actually necessary for modern GIMP. You could try modifying the PKGBUILD to remove this dependency too:

bash

*# Edit PKGBUILD and comment out or remove:*

*# 'libart-lgpl>=2.3.19'*

*# Then rebuild*

makepkg -f -si --nocheck --nodeps

## Why this is happening:

* GIMP-git might have some legacy code paths or optional features that still reference libart-lgpl
* The dependency might be outdated in the PKGBUILD
* Some plugins or legacy file format support might require it

## Quick check:

bash

*# See if the package exists in repos*

pacman -Ss libart

*# Or check AUR*

yay -Ss libart

**Try installing it with pacman first**. If it's not available in the official repos, you might need to get it from AUR, or consider removing this dependency from the PKGBUILD as well if it's truly obsolete.

What does pacman -Ss libart show?

Claude does not have the ability to run the code it generates yet.

D

Unfortunately this error while building gimp-git on archlinux - See warnings also.

WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. [ 2% | 70<1<2586 | ?bps ] Generating gimp-data/cursors/gimp-color-picker-cursors\_c with a custom command WARNING: Glycin running without sandbox. [ 2% | 71<1<2585 | ?bps ] Generating gimp-data/cursors/gimp-color-picker-cursors\_h with a custom command WARNING: Glycin running without sandbox. [ 2% | 72<1<2584 | ?bps ] Generating gimp-data/icons/Default/gimp-icon-pixbufs\_c with a custom command WARNING: Glycin running without sandbox. [ 2% | 73<1<2583 | ?bps ] Generating gimp-data/icons/Default/gimp-icon-pixbufs\_h with a custom command WARNING: Glycin running without sandbox. [ 11% | 300<1<2356 | ?bps ] Generating gimp-data/icons/Default/gimp-core-pixbufs\_c with a custom command WARNING: Glycin running without sandbox. [ 11% | 301<1<2355 | ?bps ] Generating gimp-data/icons/Default/gimp-core-pixbufs\_h with a custom command WARNING: Glycin running without sandbox. [ 11% | 307<1<2349 | ?bps ] Generating libgimp/gimp-3.0.vapi with a custom command FAILED: libgimp/gimp-3.0.vapi /usr/bin/vapigen --quiet --library=gimp-3.0 --directory=/home/admin/Downloads/gimp-git/src/build/libgimp --pkg=babl-0.1 --pkg=cairo-1.0 --pkg=gdk-pixbuf-2.0 --pkg=gegl-0.4 --pkg=gio-2.0 --pkg=glib-2.0 --pkg=gobject-2.0 --pkg=gtk+-3.0 --metadatadir=/home/admin/Downloads/gimp-git/src/gimp/libgimp /home/admin/Downloads/gimp-git/src/build/libgimp/Gimp-3.0.gir GioUnix-2.0.gir:233.11-233.27: error: Stacked arrays are not supported

This is a **Vala binding generation error** during the GIMP build process. The error "Stacked arrays are not supported" in GioUnix-2.0.gir suggests a compatibility issue with the Vala compiler (vapigen).

## Quick fixes to try:

**Option 1: Disable Vala bindings (easiest)**

bash

*# Edit PKGBUILD and modify the meson configure line to add:*

-Dvala=disabled

*# Example:*

meson setup build \

--buildtype=release \

--prefix=/usr \

-Dvala=disabled \

...other options...

**Option 2: Update your system first**

bash

sudo pacman -Syu *# Update everything, especially vala and gobject-introspection*

**Option 3: Check Vala version compatibility**

bash

vala --version

vapigen --version

**Option 4: Skip this specific target**

bash

*# You could try building without the failing target*

ninja -C build -k 0 *# Continue despite errors*

## About the Glycin warnings:

The "Glycin running without sandbox" warnings are **not fatal** - they're just security warnings about image processing running without sandboxing. You can ignore those.

## Recommended solution:

Since you're building GIMP-git (development version), **disable Vala bindings** unless you specifically need them:

1. Edit your PKGBUILD
2. Find the meson configure section
3. Add -Dvala=disabled
4. Run makepkg -f -si --nocheck --nodeps again

Vala bindings are mainly needed for developers writing GIMP plugins in Vala, so disabling them won't affect normal GIMP functionality.

Try the Vala disable option first - it's the most likely to resolve this quickly.

Claude does not have the ability to run the code it generates yet.

D

[admin@archlinux-nvme gimp-git]$ vala --version vapigen --version Vala 0.56.18-dirty Vala API Generator 0.56.18-dirty

The issue is that you have a **"dirty" version of Vala** (0.56.18-dirty), which suggests either:

* A development/modified version of Vala
* Incomplete installation
* Version mismatch with system libraries

## Solutions to try:

**Option 1: Reinstall Vala (recommended)**

bash

sudo pacman -S vala --overwrite='\*'

*# Or force reinstall*

sudo pacman -Rns vala

sudo pacman -S vala

**Option 2: Disable Vala bindings in PKGBUILD** Edit your PKGBUILD and find the meson configuration section, then add:

bash

meson setup build \

--buildtype=release \

--prefix=/usr \

-Dvala=disabled \

...other existing options...

**Option 3: Check for conflicting packages**

bash

*# Check what's providing vala*

pacman -Qo $(which vala)

*# Look for vala-related packages*

pacman -Qs vala

**Option 4: Try building with system's clean Vala**

bash

*# Make sure you have the latest versions*

sudo pacman -S vala gobject-introspection

## The "dirty" version issue:

The "-dirty" suffix usually means:

* Built from git with uncommitted changes
* Custom/modified version
* AUR version that's out of sync

**I'd recommend Option 2 (disable Vala)** first since it's fastest, then try Option 1 if you need Vala bindings later. Most GIMP users don't need the Vala bindings unless they're developing plugins.

Which approach would you like to try first?

Claude does not have the ability to run the code it generates yet.

D

Why these warnings. What can be installed to fulfill what they want:

WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox.

The "Glycin running without sandbox" warnings are related to **Glycin**, which is a new image loading library that GIMP 3.x uses for safer image format handling. Glycin is designed to run image decoders in a sandboxed environment for security.

## What Glycin needs for sandboxing:

**Option 1: Install bubblewrap (sandbox runtime)**

bash

sudo pacman -S bubblewrap

**Option 2: Install flatpak (includes sandboxing)**

bash

sudo pacman -S flatpak

**Option 3: Install firejail (alternative sandbox)**

bash

sudo pacman -S firejail

## Most likely solution:

bash

sudo pacman -S bubblewrap

Bubblewrap is the most common sandboxing solution that Glycin uses. It's what Flatpak uses internally for sandboxing applications.

## Why this happens:

* Glycin tries to isolate image processing to prevent malicious image files from compromising the system
* Without proper sandboxing, it falls back to running unsandboxed (hence the warnings)
* The image processing still works, but without the security benefits

## Alternative - Disable Glycin warnings:

If you don't want to install sandboxing and just want to suppress the warnings, you could look for a build option like:

bash

-Dglycin-sandbox=disabled

**Try installing** bubblewrap **first** - it's lightweight and will likely resolve all these warnings while providing the intended security benefits.

After installing, rebuild GIMP and the warnings should disappear.

Claude does not have the ability to run the code it generates yet.

D

Extending this discussion further, how do I emulate the above behaviour discussed - right from pkgconfig through cloning till last above step in fedora since fedora has nothing like yay -

If I clone from github and compile gimp-git in fedora rawhide, then it will not automatically create gimp-git desktop file in applications so I will need to do lots of manual sym-link type of work to make it work.

You're right that Fedora doesn't have AUR-like infrastructure, but here's how to achieve similar results:

## Fedora equivalent workflow:

### 1. Install development tools:

bash

*# Fedora's equivalent to base-devel*

sudo dnf groupinstall "Development Tools" "Development Libraries"

sudo dnf install rpm-build rpm-devel rpmdevtools

### 2. Set up RPM build environment:

bash

rpmdev-setuptree

cd ~/rpmbuild/SOURCES

### 3. Clone and prepare GIMP source:

bash

git clone https://github.com/GNOME/gimp.git gimp-git

cd gimp-git

git archive --format=tar.gz --prefix=gimp-git/ HEAD > ~/rpmbuild/SOURCES/gimp-git.tar.gz

### 4. Create a spec file:

bash

cd ~/rpmbuild/SPECS

Create gimp-git.spec based on the existing GIMP spec:

bash

*# Copy existing GIMP spec as template*

dnf download --source gimp

rpm -qp --qf '[%{FILENAMES}\n]' gimp-\*.src.rpm

rpm2cpio gimp-\*.src.rpm | cpio -idmv

cp gimp.spec gimp-git.spec

### 5. Modify the spec file:

bash

nano gimp-git.spec

Key changes needed:

* Change Name: to gimp-git
* Update Source: to your tar.gz
* Add Conflicts: gimp if needed
* Modify build dependencies as needed

### 6. Install build dependencies:

bash

sudo dnf builddep gimp-git.spec

*# Install sandbox tools for Glycin*

sudo dnf install bubblewrap

### 7. Build the RPM:

bash

rpmbuild -ba gimp-git.spec

### 8. Install the built RPM:

bash

sudo dnf install ~/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm

## Alternative: Direct build with proper installation

If you want to avoid RPM building:

### 1. Install dependencies:

bash

sudo dnf install meson ninja-build cmake gcc-c++ \

babl-devel gegl04-devel gtk3-devel glib2-devel \

cairo-devel pango-devel gdk-pixbuf2-devel \

bubblewrap vala gobject-introspection-devel

### 2. Clone and build:

bash

git clone https://github.com/GNOME/gimp.git

cd gimp

meson setup build --prefix=/usr/local --buildtype=release

ninja -C build

sudo ninja -C build install

### 3. Create desktop file manually:

bash

sudo tee /usr/local/share/applications/gimp-git.desktop << EOF

[Desktop Entry]

Version=1.0

Type=Application

Name=GIMP (Git)

GenericName=Image Editor

Comment=Create images and edit photographs

Exec=/usr/local/bin/gimp-2.99 %U

TryExec=/usr/local/bin/gimp-2.99

Icon=gimp

StartupNotify=true

MimeType=image/bmp;image/g3fax;image/gif;image/x-fits;image/x-pcx;image/x-portable-anymap;image/x-portable-bitmap;image/x-portable-graymap;image/x-portable-pixmap;image/x-psd;image/x-sgi;image/x-tga;image/x-xbitmap;image/x-xwindowdump;image/x-xcf;image/x-compressed-xcf;image/x-gimp-gbr;image/x-gimp-pat;image/x-gimp-gih;image/x-sun-raster;image/tiff;image/jpeg;image/x-psp;application/postscript;image/png;image/x-icon;image/x-xpixmap;image/svg+xml;application/pdf;image/x-wmf;image/jp2;image/jpeg2000;image/jpx;image/x-xcursor;

Categories=Graphics;2DGraphics;RasterGraphics;Photography;

Keywords=GIMP;graphic;design;illustration;painting;

EOF

### 4. Update desktop database:

bash

sudo update-desktop-database /usr/local/share/applications/

sudo gtk-update-icon-cache -t /usr/local/share/icons/hicolor/

The **RPM approach is cleaner** as it handles dependencies, desktop files, and uninstallation automatically, just like AUR packages do in Arch.

Would you prefer the RPM building approach or the direct build method?

Claude does not have the ability to run the code it generates yet.

D

Since rpm build process is cleaner - as per your suggestion - I would like to go with it.

Great choice! Let's go through the RPM build process step by step for GIMP-git on Fedora Rawhide:

Complete Fedora GIMP-git RPM Build Guide

Document

# Complete Fedora GIMP-git RPM Build Guide ## Step 1: Install Development Environment ```bash # Install RPM development tools sudo dnf groupinstall "Development Tools" "Development Libraries" sudo dnf install rpm-build rpm-devel rpmdevtools spectoo

This comprehensive guide will give you a clean RPM-based build of GIMP-git that integrates properly with your Fedora system. The key advantages are:

1. **Proper package management** - can be updated/removed cleanly with DNF
2. **Desktop integration** - automatically creates proper desktop files and menu entries
3. **Dependency handling** - RPM manages all the library dependencies
4. **Conflicts resolution** - can coexist with or replace stable GIMP
5. **System standards** - follows Fedora packaging guidelines

The process is more involved initially, but much cleaner in the long run. You'll get a properly packaged GIMP-git that behaves like any other Fedora package.

Want me to explain any specific part in more detail, or shall we start with Step 1?

D

Can you define this as bash script so that entire process is automated in Fedora like Arch - Just all take care to include switches --nodeps --nocheck also like discussed above.

Automated GIMP-git RPM Builder for Fedora

Code

#!/bin/bash # Automated GIMP-git RPM Builder for Fedora # Usage: ./build-gimp-git.sh [options] # Options: # --nodeps Skip dependency checks during RPM install # --nocheck Skip build tests # --force Force rebuild even if package exi

Here's your automated GIMP-git builder script for Fedora! This script replicates the AUR-like experience you had in Arch.

## How to use it:

### 1. Make it executable and run:

bash

chmod +x build-gimp-git.sh

*# Basic build*

./build-gimp-git.sh

*# With Arch-like options (skip deps and tests)*

./build-gimp-git.sh --nodeps --nocheck

*# Force rebuild*

./build-gimp-git.sh --force --nodeps --nocheck

*# Clean build from scratch*

./build-gimp-git.sh --clean --force

### 2. Available options (just like Arch):

* --nodeps - Skip dependency checks (like Arch's --nodeps)
* --nocheck - Skip build tests (like Arch's --nocheck)
* --force - Force rebuild even if package exists (like Arch's -f)
* --clean - Clean build directories first

### 3. What the script does automatically:

✅ Sets up RPM build environment  
✅ Downloads base GIMP spec file  
✅ Clones GIMP git source  
✅ Creates proper spec file with conflict handling  
✅ Handles dependency installation  
✅ Builds RPM package  
✅ Installs with proper desktop integration  
✅ Creates gimp-git command (separate from stable GIMP)  
✅ Handles Glycin sandbox warnings (installs bubblewrap)

### 4. Key features:

* **Conflict-free**: Can coexist with stable GIMP
* **Desktop integration**: Creates "GIMP (Git)" in applications menu
* **Clean uninstall**: sudo dnf remove gimp-git
* **Update friendly**: Just run script again with --force

This gives you the same convenience as AUR packages but with RPM's robustness!

Claude does not have the ability to run the code it generates yet.

D

Why this error in spite of no dependencies included:

admin@fedora-rh-nvme-external:~/Downloads$ chmod +x gimp-git.sh admin@fedora-rh-nvme-external:~/Downloads$ ./gimp-git.sh --nodeps --nocheck [INFO] Starting automated GIMP-git RPM build for Fedora [INFO] Setting up development environment... [INFO] Downloading base GIMP spec file... Downloading Packages: [INFO] Cloning GIMP git repository... Cloning into 'gimp-git'... remote: Enumerating objects: 580561, done. remote: Counting objects: 100% (1779/1779), done. remote: Compressing objects: 100% (565/565), done. remote: Total 580561 (delta 1326), reused 1224 (delta 1214), pack-reused 578782 (from 4) Receiving objects: 100% (580561/580561), 756.89 MiB | 3.81 MiB/s, done. Resolving deltas: 100% (487982/487982), done. [INFO] GIMP Version: 3.1.4 [INFO] Commit: 23a9758a7b [INFO] Commit Count: 55834 [INFO] Creating source tarball... [INFO] Creating GIMP-git spec file... [WARNING] Skipping dependency installation (--nodeps specified) [INFO] Building GIMP-git RPM package... setting SOURCE\_DATE\_EPOCH=1758758400 error: Failed build dependencies: babl-devel >= 0.1.108 is needed by gimp-git-3.1.4-55834.git23a9758a7b.fc44.x86\_64 gegl04-devel >= 0.4.46 is needed by gimp-git-3.1.4-55834.git23a9758a7b.fc44.x86\_64

The issue is that --nodeps only skips dependency installation in the script, but **RPM building itself still requires build dependencies** to compile the source code. This is different from Arch where --nodeps can skip both.

In RPM building, you need the actual libraries to compile against, even if you skip runtime dependency checks.

Here are your options:

## Option 1: Install minimal build deps (recommended)

bash

*# Install just the essential build dependencies*

sudo dnf install babl-devel gegl04-devel

*# Then run your script*

./gimp-git.sh --nodeps --nocheck

## Option 2: Modify the script to remove build dependencies from spec file

I'll update the script to add a --skip-builddeps option that removes problematic build requirements from the spec file:

Automated GIMP-git RPM Builder for Fedora

Code ∙ Version 2

# Default options SKIP\_DEPS=false SKIP\_TESTS=false FORCE\_BUILD=false CLEAN\_BUILD=false SKIP\_BUILDDEPS=false # Parse command line arguments while [[ $# -gt 0 ]]; do case $1 in --nodeps) SKIP\_DEPS=true shift

Now try this updated approach:

## Option 3: Use the new --skip-builddeps flag

bash

./gimp-git.sh --nodeps --nocheck --skip-builddeps

This will remove the babl-devel and gegl04-devel requirements from the spec file entirely.

## Option 4: Quick fix - Install just those two packages

bash

*# Install just the minimal deps that are failing*

sudo dnf install babl-devel gegl04-devel

*# Then run your original command*

./gimp-git.sh --nodeps --nocheck

**However, I should warn you**: Removing babl-devel and gegl04-devel will likely cause the build to fail during compilation since GIMP literally cannot compile without these core libraries. Unlike Arch where you might have had these libraries installed as part of a previous build, in RPM building the -devel packages are essential.

**I'd recommend Option 4** (installing just babl-devel and gegl04-devel) as the most practical solution. These are core dependencies that GIMP absolutely needs to compile.

Which approach would you like to try?

Claude does not have the ability to run the code it generates yet.

D

I did not change your previous script and installed both gegl and babl, but still ...

* /usr/bin/meson setup --buildtype=plain --prefix=/usr --libdir=/usr/lib64 --libexecdir=/usr/libexec --bindir=/usr/bin --sbindir=/usr/bin --includedir=/usr/include --datadir=/usr/share --mandir=/usr/share/man --infodir=/usr/share/info --localedir=/usr/share/locale --sysconfdir=/etc --localstatedir=/var --sharedstatedir=/var/lib --wrap-mode=nodownload --auto-features=enabled . redhat-linux-build -Dvala=enabled -Dg-ir-doc=false -Dvector-icons=true -Dwebp=enabled -Dopenexr=enabled -Dpython=enabled -Djavascript=enabled -Dvala=disabled The Meson build system Version: 1.9.0 Source dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4 Build dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build Build type: native build meson.build:1:0: ERROR: Unknown option: "g-ir-doc". A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.dmsuFr (%build) RPM build errors: Bad exit status from /var/tmp/rpm-tmp.dmsuFr (%build)

The error shows that g-ir-doc is not a valid meson option for GIMP-git. The meson build options in the spec file are outdated. Let me fix the script with the correct build options:

Automated GIMP-git RPM Builder for Fedora

Code ∙ Version 5

%build %meson \ -Dvala=disabled \ -Dvector-icons=true \ -Dwebp=enabled \ -Dopenexr=enabled \ -Dpython=enabled \ -Djavascript=enabled \ -Dgi-docgen=disabled \ -Dgtk-doc=false

The issue is with invalid meson options. I've updated the script to use correct options:

* Removed: -Dg-ir-doc=false (invalid option)
* Fixed: Removed duplicate -Dvala options
* Added: -Dgi-docgen=disabled and -Dgtk-doc=false (correct documentation options)

## To fix this immediately:

**Option 1: Use the updated script**

bash

*# Download the updated script and run*

./gimp-git.sh --nodeps --nocheck --force

**Option 2: Quick manual fix**

bash

*# Edit the spec file that was already created*

cd ~/rpmbuild/SPECS

nano gimp-git.spec

*# Find the %build section and replace the meson options with:*

%meson \

-Dvala=disabled \

-Dvector-icons=true \

-Dwebp=enabled \

-Dopenexr=enabled \

-Dpython=enabled \

-Djavascript=enabled \

-Dgi-docgen=disabled \

-Dgtk-doc=false

*# Then rebuild*

rpmbuild -ba gimp-git.spec

**Option 3: Check valid meson options**

bash

*# To see what options are actually available:*

cd ~/rpmbuild/SOURCES/gimp-git

meson configure

The main issue was that GIMP's meson build system has changed and the old -Dg-ir-doc=false option no longer exists. The updated script should work now.

Try running: ./gimp-git.sh --nodeps --nocheck --force

Claude does not have the ability to run the code it generates yet.

D

No I used version 5 and still:

* export FFLAGS
* FCFLAGS='-O2 -flto=auto -ffat-lto-objects -fexceptions -g -grecord-gcc-switches -pipe -Wall -Wp,-U\_FORTIFY\_SOURCE,-D\_FORTIFY\_SOURCE=3 -Wp,-D\_GLIBCXX\_ASSERTIONS -specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -m64 -march=x86-64 -mtune=generic -fasynchronous-unwind-tables -fstack-clash-protection -fcf-protection -mtls-dialect=gnu2 -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -I/usr/lib64/gfortran/modules '
* export FCFLAGS
* VALAFLAGS=-g
* export VALAFLAGS
* RUSTFLAGS='-Copt-level=3 -Cdebuginfo=2 -Ccodegen-units=1 -Cstrip=none -Cforce-frame-pointers=yes -Clink-arg=-specs=/usr/lib/rpm/redhat/redhat-package-notes --cap-lints=warn'
* export RUSTFLAGS
* LDFLAGS='-Wl,-z,relro -Wl,--as-needed -Wl,-z,pack-relative-relocs -Wl,-z,now -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -specs=/usr/lib/rpm/redhat/redhat-hardened-ld-errors -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -Wl,--build-id=sha1 -specs=/usr/lib/rpm/redhat/redhat-package-notes '
* export LDFLAGS
* LT\_SYS\_LIBRARY\_PATH=/usr/lib64:
* export LT\_SYS\_LIBRARY\_PATH
* CC=gcc
* export CC
* CXX=g++
* export CXX
* /usr/bin/meson setup --buildtype=plain --prefix=/usr --libdir=/usr/lib64 --libexecdir=/usr/libexec --bindir=/usr/bin --sbindir=/usr/bin --includedir=/usr/include --datadir=/usr/share --mandir=/usr/share/man --infodir=/usr/share/info --localedir=/usr/share/locale --sysconfdir=/etc --localstatedir=/var --sharedstatedir=/var/lib --wrap-mode=nodownload --auto-features=enabled . redhat-linux-build -Dvala=enabled -Dg-ir-doc=false -Dvector-icons=true -Dwebp=enabled -Dopenexr=enabled -Dpython=enabled -Djavascript=enabled -Dvala=disabled The Meson build system Version: 1.9.0 Source dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4 Build dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build Build type: native build meson.build:1:0: ERROR: Unknown option: "g-ir-doc". A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.XijSKa (%build) RPM build errors: Bad exit status from /var/tmp/rpm-tmp.XijSKa (%build)

I see the issue! You're still getting the old spec file with the invalid -Dg-ir-doc=false option. The script isn't being updated properly. Let me check what's happening in the actual GIMP git repository to get the correct meson options:

GIMP git meson options 2025

10 results

gimp/meson.build at master · GNOME/gimp

github.com

GIMP Developer - Building GIMP

gimp.org

Gimp

gimp.org

AUR (en) - gimp-git

archlinux.org

meson\_options.txt · master · GNOME / GIMP · GitLab

gnome.org

First Time Gimp Build - Community - GNOME Discourse

gnome.org

GIMP - Development Downloads

gimp.org

GIMP Developer - Building GIMP for Windows

gimp.org

GIMP Developer - Building GIMP for Linux

gimp.org

AUR (en) - gimp-develop-git

archlinux.org

Let me get the actual meson options from the GIMP repository:

Fetched Oh noes!

gnome.org

Let me try the GitHub mirror instead:

site:github.com GIMP meson\_options.txt

10 results

gimp/meson.build at master · GNOME/gimp

github.com

Build Resynthesizer from source

github.com

Replace autotools with meson by palindrom615 · Pull Request #16 · hamonikr/nimf

github.com

gthumb/meson\_options.txt at master · GNOME/gthumb

github.com

gimp-data/meson.build at main · GNOME/gimp-data

github.com

Query options from Meson. · mesonbuild/meson · Discussion #13309

github.com

xorg-server/meson\_options.txt at master · Hobart-l/xorg-server

github.com

meson/docs/meson\_options.txt at master · mesonbuild/meson

github.com

gimp/meson\_dist\_script.sh at master · GNOME/gimp

github.com

GitHub - LinuxBeaver/GEGL-GIMP-PLUGIN\_Edge\_Bevel: A new bevel algorithm GEGL plugin for GIMP

github.com

The issue is clear now. Let me provide you with the immediate fix. The problem is that the spec file was already generated with the old invalid options. Here's the fastest solution:

## Immediate Fix - Clean and Rebuild with Corrected Options

bash

*# 1. Clean the build completely*

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rm -rf ~/rpmbuild/SPECS/gimp-git.spec

*# 2. Create a minimal working spec file manually*

cd ~/rpmbuild/SPECS

cat > gimp-git.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 55834.git23a9758a7b%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel >= 0.1.108

BuildRequires: gegl04-devel >= 0.4.46

BuildRequires: gtk3-devel >= 3.24.0

BuildRequires: glib2-devel >= 2.70.0

Requires: bubblewrap

%description

GIMP-git is the development version of the GNU Image Manipulation Program.

%prep

%setup -q -n gimp-git-%{version}

%build

%meson \

-Dvala=disabled \

-Dvector-icons=true \

-Dwebp=enabled \

-Dopenexr=enabled

%meson\_build

%install

%meson\_install

# Rename to avoid conflicts

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

mv %{buildroot}%{\_datadir}/applications/org.gimp.GIMP.desktop \

%{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Exec=gimp-2.99|Exec=gimp-git|g' \

%{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Name=GIMP|Name=GIMP (Git)|g' \

%{buildroot}%{\_datadir}/applications/gimp-git.desktop

%files

%{\_bindir}/gimp-git

%{\_libdir}/gimp/

%{\_datadir}/gimp/

%{\_datadir}/applications/gimp-git.desktop

%{\_datadir}/icons/hicolor/\*/apps/gimp.\*

%{\_datadir}/metainfo/org.gimp.GIMP.appdata.xml

%changelog

\* Thu Sep 26 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b

- Automated build of GIMP-git

EOF

*# 3. Build with minimal valid options*

rpmbuild -bb --nodeps gimp-git.spec

*# 4. Install*

sudo rpm -Uvh --nodeps ~/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm

This uses only **valid, minimal meson options** that definitely exist:

* -Dvala=disabled (disable problematic Vala bindings)
* -Dvector-icons=true (use vector icons)
* -Dwebp=enabled (enable WebP support)
* -Dopenexr=enabled (enable OpenEXR support)

The key changes:

* Removed all invalid options (-Dg-ir-doc, -Dpython, -Djavascript)
* Simplified to only essential, known-working options
* Added --nodeps to both rpmbuild and rpm install

This should work and get you a working GIMP-git installation.

Claude does not have the ability to run the code it generates yet.

D

Library execinfo found: NO Has header "execinfo.h" : YES (cached) Run-time dependency x11 found: YES 1.8.12 Run-time dependency xmu found: YES 1.2.1 Run-time dependency xext found: YES 1.3.6 Run-time dependency xfixes found: YES 6.0.1 Has header "X11/Xmu/WinUtil.h" with dependencies xext, xmu: YES Has header "X11/extensions/shape.h" with dependencies xext, xmu: YES Checking for function "XmuClientWindow" with dependencies xext, xmu: YES Checking for function "XShapeGetRectangles" with dependencies xext, xmu: YES Has header "X11/extensions/shape.h" : YES Has header "X11/Xmu/WinUtil.h" : YES Run-time dependency libtiff-4 found: YES 4.7.0 Run-time dependency libjpeg found: YES 3.1.2 Run-time dependency zlib found: YES 1.3.1.zlib-ng Library bz2 found: YES Run-time dependency liblzma found: YES 5.8.1 meson.build:777:17: ERROR: C shared or static library 'gs' not found A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.Lu5iDm (%build) RPM build warnings: bogus date in %changelog: Thu Sep 26 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b RPM build errors: Bad exit status from /var/tmp/rpm-tmp.Lu5iDm (%build) error: open of /home/admin/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm failed: No such file or directory

Good progress! Now it's failing on a missing gs (Ghostscript) library. Let's fix this step by step:

## Quick Fix:

bash

*# 1. Install the missing ghostscript library*

sudo dnf install ghostscript-devel

*# 2. If that doesn't work, disable ghostscript support*

cd ~/rpmbuild/SPECS

*# Edit the spec file to disable ghostscript*

sed -i 's|-Dopenexr=enabled|-Dopenexr=enabled -Dgs=disabled|g' gimp-git.spec

*# 3. Rebuild*

rpmbuild -bb --nodeps gimp-git.spec

*# 4. Install if successful*

sudo rpm -Uvh --nodeps ~/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm

## Or try the comprehensive fix:

bash

*# Install common missing dependencies that might come up*

sudo dnf install ghostscript-devel libmypaint-devel \

poppler-glib-devel librsvg2-devel \

libheif-devel libavif-devel \

libwebpmux-devel libwebpdemux-devel

*# Then rebuild*

cd ~/rpmbuild/SPECS

rpmbuild -bb --nodeps gimp-git.spec

## Alternative - Disable problematic features:

If you keep hitting dependency issues, create a minimal spec file that disables optional features:

bash

cd ~/rpmbuild/SPECS

cat > gimp-git.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 55834.git23a9758a7b%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel >= 0.1.108

BuildRequires: gegl04-devel >= 0.4.46

BuildRequires: gtk3-devel >= 3.24.0

BuildRequires: glib2-devel >= 2.70.0

%description

GIMP-git minimal build

%prep

%setup -q -n gimp-git-%{version}

%build

%meson \

-Dvala=disabled \

-Dgs=disabled \

-Dwebp=disabled \

-Dopenexr=disabled \

-Dpython=disabled \

-Djavascript=disabled \

-Dvector-icons=false \

-Ddocumentation=false

%meson\_build

%install

%meson\_install

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git || true

mv %{buildroot}%{\_datadir}/applications/org.gimp.GIMP.desktop \

%{buildroot}%{\_datadir}/applications/gimp-git.desktop || true

%files

%{\_bindir}/gimp-git

%{\_bindir}/gimp-\*

%{\_libdir}/gimp/

%{\_datadir}/gimp/

%{\_datadir}/applications/\*.desktop

%{\_datadir}/icons/hicolor/\*/apps/gimp.\*

%{\_datadir}/metainfo/\*.xml

%changelog

\* Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b

- Minimal build of GIMP-git

EOF

*# Build with minimal features*

rpmbuild -bb --nodeps gimp-git.spec

sudo rpm -Uvh --nodeps ~/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm

The first approach (installing ghostscript-devel) is probably the quickest fix. Ghostscript is used for PostScript/PDF import support in GIMP.

Claude does not have the ability to run the code it generates yet.

D

Still same issue:

* export CXX
* /usr/bin/meson setup --buildtype=plain --prefix=/usr --libdir=/usr/lib64 --libexecdir=/usr/libexec --bindir=/usr/bin --sbindir=/usr/bin --includedir=/usr/include --datadir=/usr/share --mandir=/usr/share/man --infodir=/usr/share/info --localedir=/usr/share/locale --sysconfdir=/etc --localstatedir=/var --sharedstatedir=/var/lib --wrap-mode=nodownload --auto-features=enabled . redhat-linux-build -Dvala=disabled -Dvector-icons=true -Dwebp=enabled -Dopenexr=enabled -Dgs=disabled The Meson build system Version: 1.9.0 Source dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4 Build dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build Build type: native build meson.build:1:0: ERROR: Unknown option: "gs". A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.u7aDHQ (%build) RPM build warnings: bogus date in %changelog: Thu Sep 26 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b RPM build errors: Bad exit status from /var/tmp/rpm-tmp.u7aDHQ (%build) error: open of /home/admin/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm failed: No such file or directory

The issue is that the meson options we're using don't exist in GIMP's current build system. We need to find out what the actual valid options are. Let me check what's available:

bash

*# Let's see what meson options are actually available in the GIMP git source*

cd ~/rpmbuild/SOURCES/gimp-git

meson configure --help | grep -A 50 "Project options:"

*# or*

cat meson\_options.txt 2>/dev/null || echo "No meson\_options.txt file found"

But for now, let's use the **absolute minimal approach** - no custom options at all:

bash

cd ~/rpmbuild/SPECS

cat > gimp-git.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 55834.git23a9758a7b%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel >= 0.1.108

BuildRequires: gegl04-devel >= 0.4.46

BuildRequires: gtk3-devel >= 3.24.0

BuildRequires: glib2-devel >= 2.70.0

%description

GIMP-git minimal build

%prep

%setup -q -n gimp-git-%{version}

%build

# Use default meson setup with NO custom options

%meson

%meson\_build

%install

%meson\_install

# Try to rename binary files to avoid conflicts

if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

fi

if [ -f %{buildroot}%{\_bindir}/gimp ]; then

mv %{buildroot}%{\_bindir}/gimp %{buildroot}%{\_bindir}/gimp-git

fi

# Handle desktop file

for desktop\_file in %{buildroot}%{\_datadir}/applications/\*.desktop; do

if [ -f "$desktop\_file" ]; then

mv "$desktop\_file" %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Exec=gimp|Exec=gimp-git|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Name=GIMP|Name=GIMP (Git)|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

break

fi

done

%files

%{\_bindir}/gimp-git

%{\_bindir}/gimp-\*

%{\_libdir}/gimp/

%{\_libdir}/libgimp\*.so\*

%{\_datadir}/gimp/

%{\_datadir}/applications/gimp-git.desktop

%{\_datadir}/icons/hicolor/\*/apps/gimp.\*

%{\_datadir}/metainfo/\*.xml

%{\_mandir}/man\*/gimp\*

%changelog

\* Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b

- Minimal build of GIMP-git with default options

EOF

*# Clean and rebuild*

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rpmbuild -bb --nodeps gimp-git.spec

This approach:

1. **Uses NO custom meson options** - just %meson with defaults
2. **Lets meson auto-detect** what libraries are available
3. **Uses flexible file handling** for binaries and desktop files
4. **Should work** with whatever the current GIMP git build system supports

If this still fails, then we need to install the missing ghostscript library:

bash

*# Install ghostscript development files*

sudo dnf install ghostscript-devel

*# Then try building again*

rpmbuild -bb --nodeps gimp-git.spec

The key insight is that we shouldn't assume what build options are available - let meson use its defaults and auto-detection instead.

Claude does not have the ability to run the code it generates yet.

D

Library gs found: YES Run-time dependency libpng found: YES 1.6.50 Run-time dependency libmng found: YES 2.0.2 meson.build:800:11: ERROR: C shared or static library 'aa' not found A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.2x0QDT (%build) RPM build warnings: bogus date in %changelog: Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b RPM build errors:

Great progress! Now it's failing on the aa library (ASCII Art library). Let's install it:

bash

*# Install the missing ASCII Art library*

sudo dnf install aalib-devel

*# Then rebuild*

cd ~/rpmbuild/SPECS

rpmbuild -bb --nodeps gimp-git.spec

If you want to avoid these dependency hunts, we can also disable the auto-features in meson:

bash

cd ~/rpmbuild/SPECS

*# Edit the spec to disable auto-detection of optional features*

sed -i 's|%meson|%meson -Dauto\_features=disabled|g' gimp-git.spec

*# Then rebuild*

rpmbuild -bb --nodeps gimp-git.spec

The auto\_features=disabled option tells meson to only build with explicitly enabled features rather than auto-detecting what's available on your system.

**Quick choice:**

* **Option 1**: sudo dnf install aalib-devel (keep auto-detection, install what's needed)
* **Option 2**: Use -Dauto\_features=disabled (minimal build, skip optional features)

I'd recommend **Option 1** first since you're already making good progress, and if it keeps asking for more libraries, then switch to **Option 2** for a minimal build.

What would you prefer?

Claude does not have the ability to run the code it generates yet.

D

Still some dependencies missing: Run-time dependency libiff found: NO (tried pkgconfig and cmake) meson.build:806:9: ERROR: Dependency "libiff" not found, tried pkgconfig and cmake A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.uvlA0J (%build) RPM build warnings: bogus date in %changelog: Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b RPM build errors: Bad exit status from /var/tmp/rpm-tmp.uvlA0J (%build) admin@fedora-rh-nvme-external:~/rpmbuild/SPECS$

The libiff library is quite obscure (it's for Interchange File Format, an old Amiga format). Let's switch to the minimal build approach to avoid chasing down every optional dependency:

bash

cd ~/rpmbuild/SPECS

*# Replace the meson line to disable auto-features*

sed -i 's|%meson|%meson -Dauto\_features=disabled|g' gimp-git.spec

*# Clean and rebuild*

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rpmbuild -bb --nodeps gimp-git.spec

If that still fails, let's try an even more minimal approach:

bash

cd ~/rpmbuild/SPECS

cat > gimp-git.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 55834.git23a9758a7b%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel >= 0.1.108

BuildRequires: gegl04-devel >= 0.4.46

BuildRequires: gtk3-devel >= 3.24.0

BuildRequires: glib2-devel >= 2.70.0

BuildRequires: cairo-devel

BuildRequires: gdk-pixbuf2-devel

%description

GIMP-git minimal build with disabled optional features

%prep

%setup -q -n gimp-git-%{version}

%build

# Minimal build - disable auto-detection and most optional features

%meson \

-Dauto\_features=disabled \

-Dgtk3=enabled \

-Dgimp\_console=enabled

%meson\_build

%install

%meson\_install

# Handle binary naming

if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

elif [ -f %{buildroot}%{\_bindir}/gimp ]; then

mv %{buildroot}%{\_bindir}/gimp %{buildroot}%{\_bindir}/gimp-git

fi

# Handle desktop file

if [ -f %{buildroot}%{\_datadir}/applications/org.gimp.GIMP.desktop ]; then

mv %{buildroot}%{\_datadir}/applications/org.gimp.GIMP.desktop \

%{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Exec=gimp|Exec=gimp-git|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Name=GIMP|Name=GIMP (Git)|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

fi

%files

%{\_bindir}/gimp-git

%{\_bindir}/gimp-\*

%{\_libdir}/gimp/

%{\_libdir}/libgimp\*.so\*

%{\_datadir}/gimp/

%{\_datadir}/applications/gimp-git.desktop

%{\_datadir}/icons/hicolor/\*/apps/gimp.\*

%{\_datadir}/metainfo/\*.xml

%{\_mandir}/man\*/gimp\* || true

%changelog

\* Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b

- Minimal GIMP-git build with disabled optional features

EOF

*# Build the minimal version*

rpmbuild -bb --nodeps gimp-git.spec

This should give you a working GIMP-git with just the core functionality and none of the optional file format support that's causing dependency issues.

The -Dauto\_features=disabled is the key - it tells meson to not automatically enable features just because the libraries are present on your system.

Claude does not have the ability to run the code it generates yet.

D

* export FCFLAGS
* VALAFLAGS=-g
* export VALAFLAGS
* RUSTFLAGS='-Copt-level=3 -Cdebuginfo=2 -Ccodegen-units=1 -Cstrip=none -Cforce-frame-pointers=yes -Clink-arg=-specs=/usr/lib/rpm/redhat/redhat-package-notes --cap-lints=warn'
* export RUSTFLAGS
* LDFLAGS='-Wl,-z,relro -Wl,--as-needed -Wl,-z,pack-relative-relocs -Wl,-z,now -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -specs=/usr/lib/rpm/redhat/redhat-hardened-ld-errors -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -Wl,--build-id=sha1 -specs=/usr/lib/rpm/redhat/redhat-package-notes '
* export LDFLAGS
* LT\_SYS\_LIBRARY\_PATH=/usr/lib64:
* export LT\_SYS\_LIBRARY\_PATH
* CC=gcc
* export CC
* CXX=g++
* export CXX
* /usr/bin/meson setup --buildtype=plain --prefix=/usr --libdir=/usr/lib64 --libexecdir=/usr/libexec --bindir=/usr/bin --sbindir=/usr/bin --includedir=/usr/include --datadir=/usr/share --mandir=/usr/share/man --infodir=/usr/share/info --localedir=/usr/share/locale --sysconfdir=/etc --localstatedir=/var --sharedstatedir=/var/lib --wrap-mode=nodownload --auto-features=enabled . redhat-linux-build -Dauto\_features=disabled -Dgtk3=enabled -Dgimp\_console=enabled Got argument auto\_features as both -Dauto\_features= and --auto-features. Pick one. error: Bad exit status from /var/tmp/rpm-tmp.9RI0A6 (%build) RPM build warnings: bogus date in %changelog: Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b RPM build errors: Bad exit status from /var/tmp/rpm-tmp.9RI0A6 (%build)

The issue is that %meson macro automatically adds --auto-features=enabled and we're also adding -Dauto\_features=disabled, causing a conflict. Let's use the raw meson command instead:

bash

cd ~/rpmbuild/SPECS

cat > gimp-git.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 55834.git23a9758a7b%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel >= 0.1.108

BuildRequires: gegl04-devel >= 0.4.46

BuildRequires: gtk3-devel >= 3.24.0

BuildRequires: glib2-devel >= 2.70.0

BuildRequires: cairo-devel

BuildRequires: gdk-pixbuf2-devel

%description

GIMP-git minimal build with disabled optional features

%prep

%setup -q -n gimp-git-%{version}

%build

# Use raw meson setup instead of %meson macro to avoid conflicts

meson setup builddir \

--prefix=%{\_prefix} \

--libdir=%{\_libdir} \

--buildtype=plain \

-Dauto\_features=disabled \

-Dgtk3=enabled

ninja -C builddir

%install

DESTDIR=%{buildroot} ninja -C builddir install

# Handle binary naming

if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

elif [ -f %{buildroot}%{\_bindir}/gimp ]; then

mv %{buildroot}%{\_bindir}/gimp %{buildroot}%{\_bindir}/gimp-git

fi

# Handle desktop file

for desktop in %{buildroot}%{\_datadir}/applications/\*.desktop; do

if [ -f "$desktop" ]; then

mv "$desktop" %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Exec=gimp|Exec=gimp-git|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Name=GIMP|Name=GIMP (Git)|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

break

fi

done

%files

%{\_bindir}/gimp-git

%{\_bindir}/gimp-\* || true

%{\_libdir}/gimp/

%{\_libdir}/libgimp\*.so\*

%{\_datadir}/gimp/

%{\_datadir}/applications/gimp-git.desktop

%{\_datadir}/icons/hicolor/\*/apps/gimp.\* || true

%{\_datadir}/metainfo/\*.xml || true

%changelog

\* Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b

- Minimal GIMP-git build with raw meson commands

EOF

*# Clean and rebuild*

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rpmbuild -bb --nodeps gimp-git.spec

Key changes:

1. **Used raw** meson setup instead of %meson macro
2. **Used** ninja -C builddir instead of %meson\_build
3. **Used** DESTDIR=%{buildroot} ninja -C builddir install instead of %meson\_install
4. **Added** || true to optional file patterns to prevent failures if files don't exist

This bypasses the RPM macro system that was causing the auto\_features conflict.

Claude does not have the ability to run the code it generates yet.

D

* export RUSTFLAGS
* LDFLAGS='-Wl,-z,relro -Wl,--as-needed -Wl,-z,pack-relative-relocs -Wl,-z,now -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -specs=/usr/lib/rpm/redhat/redhat-hardened-ld-errors -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -Wl,--build-id=sha1 -specs=/usr/lib/rpm/redhat/redhat-package-notes '
* export LDFLAGS
* LT\_SYS\_LIBRARY\_PATH=/usr/lib64:
* export LT\_SYS\_LIBRARY\_PATH
* CC=gcc
* export CC
* CXX=g++
* export CXX
* /usr/bin/meson setup --buildtype=plain --prefix=/usr --libdir=/usr/lib64 --libexecdir=/usr/libexec --bindir=/usr/bin --sbindir=/usr/bin --includedir=/usr/include --datadir=/usr/share --mandir=/usr/share/man --infodir=/usr/share/info --localedir=/usr/share/locale --sysconfdir=/etc --localstatedir=/var --sharedstatedir=/var/lib --wrap-mode=nodownload --auto-features=enabled . redhat-linux-build macro to avoid conflicts usage: meson [-h] {setup,configure,dist,install,introspect,init,test,wrap,subprojects,rewrite,compile,devenv,env2mfile,reprotest,format,fmt,help} ... meson: error: unrecognized arguments: macro to avoid conflicts error: Bad exit status from /var/tmp/rpm-tmp.MRqysb (%build) RPM build warnings: bogus date in %changelog: Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b RPM build errors: Bad exit status from /var/tmp/rpm-tmp.MRqysb (%build)

There's a syntax error in the spec file - it looks like part of a comment got included in the meson command. Let me fix this:

bash

cd ~/rpmbuild/SPECS

cat > gimp-git.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 55834.git23a9758a7b%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel >= 0.1.108

BuildRequires: gegl04-devel >= 0.4.46

BuildRequires: gtk3-devel >= 3.24.0

BuildRequires: glib2-devel >= 2.70.0

BuildRequires: cairo-devel

BuildRequires: gdk-pixbuf2-devel

%description

GIMP-git minimal build with disabled optional features

%prep

%setup -q -n gimp-git-%{version}

%build

meson setup builddir \

--prefix=%{\_prefix} \

--libdir=%{\_libdir} \

--buildtype=plain \

-Dauto\_features=disabled \

-Dgtk3=enabled

ninja -C builddir

%install

DESTDIR=%{buildroot} ninja -C builddir install

if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

elif [ -f %{buildroot}%{\_bindir}/gimp ]; then

mv %{buildroot}%{\_bindir}/gimp %{buildroot}%{\_bindir}/gimp-git

fi

for desktop in %{buildroot}%{\_datadir}/applications/\*.desktop; do

if [ -f "$desktop" ]; then

mv "$desktop" %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Exec=gimp|Exec=gimp-git|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

sed -i 's|Name=GIMP|Name=GIMP (Git)|g' %{buildroot}%{\_datadir}/applications/gimp-git.desktop

break

fi

done

%files

%{\_bindir}/gimp-git

%{\_libdir}/gimp/

%{\_libdir}/libgimp\*.so\*

%{\_datadir}/gimp/

%{\_datadir}/applications/gimp-git.desktop

%changelog

\* Wed Sep 25 2025 Build User <user@localhost> - 3.1.4-55834.git23a9758a7b

- Minimal GIMP-git build

EOF

*# Clean and rebuild*

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rpmbuild -bb --nodeps gimp-git.spec

I've cleaned up the spec file to remove any stray text and simplified the %files section to only include essential files that should definitely exist.

The key fix is removing the errant text that was getting added to the meson command line. This should now run properly with the minimal build configuration.

Claude does not have the ability to run the code it generates yet.

D

* LDFLAGS='-Wl,-z,relro -Wl,--as-needed -Wl,-z,pack-relative-relocs -Wl,-z,now -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -specs=/usr/lib/rpm/redhat/redhat-hardened-ld-errors -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -Wl,--build-id=sha1 -specs=/usr/lib/rpm/redhat/redhat-package-notes '
* export LDFLAGS
* LT\_SYS\_LIBRARY\_PATH=/usr/lib64:
* export LT\_SYS\_LIBRARY\_PATH
* CC=gcc
* export CC
* CXX=g++
* export CXX
* /usr/bin/meson setup --buildtype=plain --prefix=/usr --libdir=/usr/lib64 --libexecdir=/usr/libexec --bindir=/usr/bin --sbindir=/usr/bin --includedir=/usr/include --datadir=/usr/share --mandir=/usr/share/man --infodir=/usr/share/info --localedir=/usr/share/locale --sysconfdir=/etc --localstatedir=/var --sharedstatedir=/var/lib --wrap-mode=nodownload --auto-features=enabled . redhat-linux-build -Dvala=enabled -Dg-ir-doc=false -Dvector-icons=true -Dwebp=enabled -Dopenexr=enabled -Dpython=enabled -Djavascript=enabled -Dvala=disabled The Meson build system Version: 1.9.0 Source dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4 Build dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build Build type: native build meson.build:1:0: ERROR: Unknown option: "g-ir-doc". A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.vPKDV3 (%build) RPM build errors: Bad exit status from /var/tmp/rpm-tmp.vPKDV3 (%build) admin@fedora-rh-nvme-external:~/Downloads$

The problem is that the RPM build system is still using the %meson macro from somewhere else (likely the original script-generated spec file is still being used). You're seeing the old invalid options like g-ir-doc again.

Let's completely start fresh:

bash

*# 1. Clean everything completely*

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rm -rf ~/rpmbuild/SPECS/gimp-git.spec

rm -rf ~/rpmbuild/SRPMS/gimp-git-\*

rm -rf ~/rpmbuild/RPMS/\*/gimp-git-\*

*# 2. Create a completely new, clean spec file*

cd ~/rpmbuild/SPECS

cat > gimp-git-clean.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 1%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel

BuildRequires: gegl04-devel

BuildRequires: gtk3-devel

BuildRequires: glib2-devel

BuildRequires: cairo-devel

BuildRequires: gdk-pixbuf2-devel

%description

GIMP-git development version with minimal configuration

%prep

%setup -q -n gimp-git-%{version}

%build

meson setup builddir --prefix=%{\_prefix} --libdir=%{\_libdir} --buildtype=plain -Dauto\_features=disabled

ninja -C builddir

%install

DESTDIR=%{buildroot} ninja -C builddir install

# Rename to avoid conflicts

if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

fi

%files

%{\_bindir}/gimp-git

%{\_libdir}/gimp/

%{\_datadir}/gimp/

%changelog

\* Wed Sep 25 2025 User - 3.1.4-1

- Clean build

EOF

*# 3. Build with the new clean spec*

rpmbuild -bb --nodeps gimp-git-clean.spec

*# 4. If successful, install*

sudo rpm -Uvh --nodeps ~/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm

This approach:

1. **Completely cleans** all previous build attempts
2. **Uses a new spec filename** (gimp-git-clean.spec) to avoid any cached issues
3. **Uses absolutely minimal options** - only auto\_features=disabled
4. **Avoids all RPM macros** that might be adding unwanted options
5. **Has minimal %files** section to avoid missing file errors

If this still shows the old invalid options, it means there's another spec file or macro being used. In that case, check:

bash

*# See what spec files exist*

ls -la ~/rpmbuild/SPECS/gimp\*

*# Check if there are any cached builds*

find ~/rpmbuild -name "\*gimp\*" -type f

The key is using the new filename gimp-git-clean.spec to ensure we're not hitting any cached configurations.

Claude does not have the ability to run the code it generates yet.

D

%description GIMP-git development version with minimal configuration %prep %setup -q -n gimp-git-%{version} %build meson setup builddir --prefix=%{\_prefix} --libdir=%{\_libdir} --buildtype=plain -Dauto\_features=disabled ninja -C builddir %install DESTDIR=%{buildroot} ninja -C builddir install # Rename to avoid conflicts if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git fi %files %{\_bindir}/gimp-git %{\_libdir}/gimp/ %{\_datadir}/gimp/ %changelog \* Wed Sep 25 2025 User - 3.1.4-1 - Clean build EOF # 3. Build with the new clean spec rpmbuild -bb --nodeps gimp-git-clean.sp error: failed to stat /home/admin/rpmbuild/SPECS/gimp-git-clean.sp: No such file or directory admin@fedora-rh-nvme-external:~/rpmbuild/SPECS$ rpmbuild -bb --nodeps gimp-git-clean.sp error: failed to stat /home/admin/rpmbuild/SPECS/gimp-git-clean.sp: No such file or directory

You have a typo in the filename. It should be .spec not .sp. Let me fix that:

bash

cd ~/rpmbuild/SPECS

*# Build with the correct filename*

rpmbuild -bb --nodeps gimp-git-clean.spec

*# If successful, install*

sudo rpm -Uvh --nodeps ~/rpmbuild/RPMS/x86\_64/gimp-git-\*.rpm

The file should be gimp-git-clean.spec (with .spec extension), not gimp-git-clean.sp.

If you're still having issues, let's verify the file was created correctly:

bash

*# Check if the file exists*

ls -la ~/rpmbuild/SPECS/gimp-git-clean.spec

*# If it doesn't exist, recreate it*

cd ~/rpmbuild/SPECS

cat > gimp-git-clean.spec << 'EOF'

Name: gimp-git

Version: 3.1.4

Release: 1%{?dist}

Summary: GNU Image Manipulation Program (Development Version)

License: GPLv3+

URL: https://www.gimp.org/

Source0: gimp-git-%{version}.tar.gz

Conflicts: gimp

BuildRequires: gcc-c++

BuildRequires: meson >= 0.60.0

BuildRequires: ninja-build

BuildRequires: pkgconfig

BuildRequires: babl-devel

BuildRequires: gegl04-devel

BuildRequires: gtk3-devel

BuildRequires: glib2-devel

BuildRequires: cairo-devel

BuildRequires: gdk-pixbuf2-devel

%description

GIMP-git development version with minimal configuration

%prep

%setup -q -n gimp-git-%{version}

%build

meson setup builddir --prefix=%{\_prefix} --libdir=%{\_libdir} --buildtype=plain -Dauto\_features=disabled

ninja -C builddir

%install

DESTDIR=%{buildroot} ninja -C builddir install

if [ -f %{buildroot}%{\_bindir}/gimp-2.99 ]; then

mv %{buildroot}%{\_bindir}/gimp-2.99 %{buildroot}%{\_bindir}/gimp-git

fi

%files

%{\_bindir}/gimp-git

%{\_libdir}/gimp/

%{\_datadir}/gimp/

%changelog

\* Wed Sep 25 2025 User - 3.1.4-1

- Clean build

EOF

*# Then build*

rpmbuild -bb --nodeps gimp-git-clean.spec

Claude does not have the ability to run the code it generates yet.

D

This issue?

Configuring config.h using configuration Configuring gimpversion.h using configuration Program meson-enumgen.py found: YES (/usr/bin/python3 /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/pdb/meson-enumgen.py) Program meson-pdbgen.py found: YES (/usr/bin/python3 /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/pdb/meson-pdbgen.py) Program meson-enumcode.py found: YES (/usr/bin/python3 /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/pdb/meson-enumcode.py) Program msgfmt found: YES (/usr/bin/msgfmt) Program msginit found: YES (/usr/bin/msginit) Program msgmerge found: YES (/usr/bin/msgmerge) Program xgettext found: YES (/usr/bin/xgettext) Configuring gimp-release using configuration Configuring gimprc using configuration Configuring dockable-dialogs-menuitems.ui using configuration Configuring dockable-menu.ui.in using configuration Configuring app-dialogs-menuitems.ui using configuration Configuring image-menu.ui.in using configuration meson.build:1846:0: ERROR: Nonexistent build file 'gimp-data/meson.build' A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/builddir/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.QZibvy (%build) RPM build warnings: bogus date in %changelog: Wed Sep 25 2025 User - 3.1.4-1 RPM build errors: Bad exit status from /var/tmp/rpm-tmp.QZibvy (%build)

The issue is that GIMP expects a gimp-data subdirectory which is missing from our source tarball. The original automated script was supposed to handle this by cloning both the main GIMP repo and the gimp-data repo separately.

Let's fix this by recreating the source tarball properly:

bash

*# 1. Go back to sources and fix the tarball*

cd ~/rpmbuild/SOURCES

*# 2. Remove the incomplete tarball*

rm -f gimp-git-3.1.4.tar.gz

*# 3. Clone both repositories properly*

rm -rf gimp-git

git clone --recursive https://github.com/GNOME/gimp.git gimp-git

*# The --recursive flag should get gimp-data as a submodule*

*# If that doesn't work, we need to clone gimp-data separately:*

cd gimp-git

if [ ! -d gimp-data ]; then

git clone https://github.com/GNOME/gimp-data.git gimp-data

fi

cd ..

*# 4. Create the proper tarball with gimp-data included*

tar -czf gimp-git-3.1.4.tar.gz --exclude='.git' gimp-git/

*# 5. Clean and rebuild*

cd ~/rpmbuild/SPECS

rm -rf ~/rpmbuild/BUILD/gimp-git-\*

rpmbuild -bb --nodeps gimp-git-clean.spec

The key issue is that GIMP's build system expects the gimp-data directory to be present alongside the main source code. This contains brushes, patterns, palettes, and other data files that GIMP needs.

If the recursive clone doesn't work, try this alternative:

bash

cd ~/rpmbuild/SOURCES

rm -rf gimp-git

git clone https://github.com/GNOME/gimp.git gimp-git

cd gimp-git

*# Manually clone gimp-data into the right place*

git clone https://github.com/GNOME/gimp-data.git gimp-data

*# Go back and create tarball*

cd ..

tar -czf gimp-git-3.1.4.tar.gz --exclude='.git\*' gimp-git/

*# Rebuild*

cd ~/rpmbuild/SPECS

rpmbuild -bb --nodeps gimp-git-clean.spec

This should resolve the "Nonexistent build file 'gimp-data/meson.build'" error.

Claude does not have the ability to run the code it generates yet.

D

I think this is too much. Let's switch back to manual method which can be made as close to rpm as possible. Create bash script to automatically clone gimp, gimp-data, building, creating desktop file, etc. for fedora based on above discussions. Do not include comments and shorten code as much as possible.

Fedora GIMP-git Manual Build Script

Code

#!/bin/bash set -e SKIP\_DEPS=false SKIP\_TESTS=false CLEAN=false while [[ $# -gt 0 ]]; do case $1 in --nodeps) SKIP\_DEPS=true; shift ;; --nocheck) SKIP\_TESTS=true; shift ;; --clean) CLEAN=true; shift ;; \*) shift ;

This script does everything you need:

**Usage:**

bash

chmod +x gimp-build.sh

./gimp-build.sh *# Normal build*

./gimp-build.sh --nodeps --nocheck *# Skip deps and tests*

./gimp-build.sh --clean *# Clean build from scratch*

**What it does:**

* Installs dependencies (unless --nodeps)
* Clones GIMP and gimp-data repos
* Links gimp-data properly
* Builds with minimal options
* Installs to /usr/local
* Creates desktop file as "GIMP (Git)"
* Updates desktop and icon databases

**Result:**

* Binary: /usr/local/bin/gimp-2.99
* Desktop entry: "GIMP (Git)" in applications menu
* No conflicts with system GIMP

Much simpler than RPM building!

Claude does not have the ability to run the code it generates yet.

D

To clean up earlier setup of rpm build, I deleted rpmbuild directory. Is that enough?

Yes, deleting the ~/rpmbuild directory is enough to clean up the RPM build setup. That directory contained:

* ~/rpmbuild/SOURCES/ - source files and tarballs
* ~/rpmbuild/SPECS/ - spec files
* ~/rpmbuild/BUILD/ - temporary build files
* ~/rpmbuild/RPMS/ - built RPM packages
* ~/rpmbuild/SRPMS/ - source RPM packages

However, you might also want to check for any installed gimp-git RPM packages:

bash

*# Check if any gimp-git RPM was installed*

rpm -qa | grep gimp-git

*# If found, remove it*

sudo dnf remove gimp-git

Also, the RPM development tools are still installed. If you want to completely clean up:

bash

*# Optional: Remove RPM build tools if you don't need them*

sudo dnf remove rpm-build rpm-devel rpmdevtools spectool

But keeping the development tools won't hurt - they're useful for other projects too.

Now you can use the manual build script I provided, which will install GIMP-git to /usr/local instead of system directories, making it completely separate from any RPM-managed software.

Claude does not have the ability to run the code it generates yet.

D

Why this bug again makes miserable to build:

Version: 1.9.0 Source dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4 Build dir: /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build Build type: native build meson.build:1:0: ERROR: Unknown option: "g-ir-doc". A full log can be found at /home/admin/rpmbuild/BUILD/gimp-git-3.1.4-build/gimp-git-3.1.4/redhat-linux-build/meson-logs/meson-log.txt error: Bad exit status from /var/tmp/rpm-tmp.QAs112 (%build) RPM build errors: Bad exit status from /var/tmp/rpm-tmp.QAs112 (%build) admin@fedora-rh-nvme-external:~/Downloads$

You're still trying to use RPM building even though you said you deleted the rpmbuild directory. The error shows it's recreating /home/admin/rpmbuild/BUILD/ and using the old invalid meson options.

**Stop using RPM building!** Use the manual build script I just provided instead:

bash

*# Make sure you're using the NEW manual script, not any old RPM commands*

cd ~/Downloads *# or wherever you saved the script*

chmod +x gimp-build.sh

./gimp-build.sh --nodeps --nocheck

**Do NOT run:**

* rpmbuild commands
* makepkg commands
* The old automated RPM script

**The manual script will:**

* Build directly with meson/ninja (no RPM)
* Install to /usr/local (no conflicts)
* Create proper desktop file
* Work without dependency issues

If you're still seeing RPM errors, you're running the wrong script. The manual build script I provided completely avoids the RPM system and the invalid g-ir-doc option that keeps causing problems.

**Use the manual build method - forget about RPM for GIMP-git.**

Claude does not have the ability to run the code it generates yet.

D

Which directory is being overwritten:

Nothing to do. Cloning into 'gimp'... remote: Enumerating objects: 5397, done. remote: Counting objects: 100% (5397/5397), done. remote: Compressing objects: 100% (3834/3834), done. remote: Total 5397 (delta 2423), reused 2338 (delta 1534), pack-reused 0 (from 0) Receiving objects: 100% (5397/5397), 29.19 MiB | 5.52 MiB/s, done. Resolving deltas: 100% (2423/2423), done. Cloning into 'gimp-data'... remote: Enumerating objects: 1731, done. remote: Counting objects: 100% (1731/1731), done. remote: Compressing objects: 100% (1037/1037), done. remote: Total 1731 (delta 707), reused 1683 (delta 679), pack-reused 0 (from 0) Receiving objects: 100% (1731/1731), 15.57 MiB | 4.42 MiB/s, done. Resolving deltas: 100% (707/707), done. ln: ./gimp-data: cannot overwrite directory

The ln -sf command is failing because gimp-data already exists as a directory. Let me fix the script:

Fedora GIMP-git Manual Build Script

Code ∙ Version 2

cd gimp rm -rf gimp-data ln -sf ../gimp-data .

Or run this quick fix manually:

bash

cd ~/gimp-build/gimp

rm -rf gimp-data

ln -sf ../gimp-data .

*# Then continue with the build*

MESON\_OPTS="-Dauto\_features=disabled --prefix=/usr/local"

meson setup build $MESON\_OPTS

ninja -C build

sudo ninja -C build install

The issue was that GIMP has a gimp-data subdirectory in its git repo, but we need to replace it with a symlink to our separately cloned gimp-data repository. The rm -rf gimp-data removes the existing directory first, then creates the symlink.

Claude does not have the ability to run the code it generates yet.

D

admin@fedora-rh-nvme-external:~/Downloads$ ./gimp\_git.sh --nodeps --nocheck Cloning into 'gimp'... remote: Enumerating objects: 5397, done. remote: Counting objects: 100% (5397/5397), done. remote: Compressing objects: 100% (3834/3834), done. remote: Total 5397 (delta 2423), reused 2338 (delta 1534), pack-reused 0 (from 0) Receiving objects: 100% (5397/5397), 29.19 MiB | 5.55 MiB/s, done. Resolving deltas: 100% (2423/2423), done. Cloning into 'gimp-data'... remote: Enumerating objects: 1731, done. remote: Counting objects: 100% (1731/1731), done. remote: Compressing objects: 100% (1037/1037), done. remote: Total 1731 (delta 707), reused 1683 (delta 679), pack-reused 0 (from 0) Receiving objects: 100% (1731/1731), 15.57 MiB | 6.54 MiB/s, done. Resolving deltas: 100% (707/707), done. The Meson build system Version: 1.9.0 Source dir: /home/admin/gimp-build/gimp Build dir: /home/admin/gimp-build/gimp/build Build type: native build meson.build:1:0: ERROR: Unknown option: "tests". A full log can be found at /home/admin/gimp-build/gimp/build/meson-logs/meson-log.txt

The meson option tests doesn't exist in GIMP. Let me fix the script:

Fedora GIMP-git Manual Build Script

Code ∙ Version 3

MESON\_OPTS="-Dauto\_features=disabled --prefix=/usr/local"

Or run this quick fix:

bash

cd ~/gimp-build/gimp

rm -rf build

meson setup build -Dauto\_features=disabled --prefix=/usr/local

ninja -C build

sudo ninja -C build install

The --nocheck option doesn't need any special meson flags - with auto\_features=disabled, tests won't be built or run anyway. The script now uses only the minimal, valid meson options.

Claude does not have the ability to run the code it generates yet.

D

???

Compiler for C supports arguments -Wsometimes-uninitialized: NO Compiler for C supports arguments -Wtautological-unsigned-enum-zero-compare: NO Compiler for C supports arguments -Wunneeded-internal-declaration: NO Compiler for C supports arguments -Wunused-function: YES Compiler for C supports arguments -Wunused-value: YES Compiler for C supports arguments -Werror=implicit-function-declaration: YES Library m found: YES Library dl found: YES Found pkg-config: YES (/usr/bin/pkg-config) 2.3.0 Run-time dependency atk found: YES 2.58.0 Dependency babl-0.1 found: NO. Found 0.1.109 but need: '>=0.1.114' Found CMake: /usr/bin/cmake (4.0.0) WARNING: CMake Toolchain: Failed to determine CMake compilers state Run-time dependency babl-0.1 found: NO (tried pkgconfig and cmake) Run-time dependency babl found: NO (tried pkgconfig and cmake) meson.build:415:20: ERROR: Dependency "babl" not found, tried pkgconfig and cmake A full log can be found at /home/admin/gimp-build/gimp/build/meson-logs/meson-log.txt admin@fedora-rh-nvme-external:~/Downloads$ dnf -y install babl\* The requested operation requires superuser privileges. Please log in as a user with elevated rights, or use the "--assumeno" or "--downloadonly" options to run the command without modifying the system state. admin@fedora-rh-nvme-external:~/Downloads$ sudo dnf -y install babl\* Updating and loading repositories: Repositories loaded. Package "babl-0.1.114-2.fc43.x86\_64" is already installed. Nothing to do.

You have babl version 0.1.109 but GIMP-git needs >= 0.1.114. You need to update babl to a newer version. Try this:

bash

*# Try to get newer babl from updates-testing repo*

sudo dnf install --enablerepo=updates-testing babl babl-devel

*# Or build babl from source*

cd ~/gimp-build

git clone https://gitlab.gnome.org/GNOME/babl.git

cd babl

meson setup build --prefix=/usr/local

ninja -C build

sudo ninja -C build install

*# Update library cache*

sudo ldconfig

*# Then try building GIMP again*

cd ~/gimp-build/gimp

rm -rf build

PKG\_CONFIG\_PATH=/usr/local/lib64/pkgconfig:/usr/local/lib/pkgconfig:$PKG\_CONFIG\_PATH \

meson setup build -Dauto\_features=disabled --prefix=/usr/local

ninja -C build

sudo ninja -C build install

The issue is that Fedora's babl package is too old for the current GIMP git version. Building babl from source to /usr/local should fix this.

Alternatively, if you want a quick solution:

bash

*# Try with system babl anyway (might work)*

cd ~/gimp-build/gimp

rm -rf build

meson setup build -Dauto\_features=disabled --prefix=/usr/local -Dbabl:required=false

But the first approach (building newer babl) is better.

Claude does not have the ability to run the code it generates yet.

D

modify your script to build babl and gegl from github and then finally gimp-git from github

Fedora GIMP-git Manual Build Script

Code ∙ Version 4

#!/bin/bash set -e SKIP\_DEPS=false SKIP\_TESTS=false CLEAN=false while [[ $# -gt 0 ]]; do case $1 in --nodeps) SKIP\_DEPS=true; shift ;; --nocheck) SKIP\_TESTS=true; shift ;; --clean) CLEAN=true; shift ;; \*) shift ;

The updated script now:

1. **Builds BABL first** - clones from GitHub and installs to /usr/local
2. **Builds GEGL second** - depends on the freshly built BABL
3. **Builds GIMP last** - depends on both BABL and GEGL
4. **Sets proper environment variables** - so each build finds the previous ones
5. **Runs ldconfig** after each library install to update library cache

**Usage:**

bash

chmod +x gimp-build.sh

./gimp-build.sh --nodeps --nocheck

This ensures you get the latest compatible versions of all three components, built in the correct dependency order. The whole process will take longer but should work reliably.

Claude does not have the ability to run the code it generates yet.

D

???

NOTICE: Future-deprecated features used: \* 0.62.0: {'pkgconfig.generate variable for builtin directories'} babl 0.1.115 Directories prefix : /usr/local libdir : lib64 Optional features BABL docs (babl website): true gi-docgen : false Introspection : true VALA support : true Debug symbols format : Native Processor extensions mmx : true sse : true sse2 : true sse4\_1 : true avx2 : true f16c (half fp) : true Optional dependencies lcms : true User defined options prefix : /usr/local Found ninja-1.13.1 at /usr/bin/ninja-build Traceback (most recent call last): File "/home/admin/.local/bin/ninja", line 5, in <module> from ninja import ninja ModuleNotFoundError: No module named 'ninja' admin@fedora-rh-nvme-external:~/Downloads$ sudo dnf -y install ninja\* meson\* Updating and loading repositories: Repositories loaded. Package "ninja-build-1.13.1-4.fc44.x86\_64" is already installed. Package "meson-1.9.0-3.fc44.noarch" is already installed. Nothing to do.

The issue is that you have a local Python ninja module that's conflicting with the system ninja-build. Let's fix this:

bash

*# Remove the problematic local ninja*

rm -f ~/.local/bin/ninja

*# Or if that doesn't work, temporarily rename it*

mv ~/.local/bin/ninja ~/.local/bin/ninja.bak 2>/dev/null || true

*# Make sure the system ninja-build is used*

which ninja-build

*# Should show: /usr/bin/ninja-build*

*# Continue the build*

cd ~/gimp-build/babl

ninja-build -C build

sudo ninja-build -C build install

sudo ldconfig

*# Then continue with GEGL*

cd ../gegl

rm -rf build

meson setup build --prefix=/usr/local -Dauto\_features=disabled

ninja-build -C build

sudo ninja-build -C build install

sudo ldconfig

*# Finally GIMP*

cd ../gimp

rm -rf build

meson setup build -Dauto\_features=disabled --prefix=/usr/local

ninja-build -C build

sudo ninja-build -C build install

Or update the script to explicitly use ninja-build instead of ninja:

Fedora GIMP-git Manual Build Script

Code ∙ Version 5

ninja-build -C build

The script now uses ninja-build explicitly, which is the correct binary name on Fedora systems.

Claude does not have the ability to run the code it generates yet.

D

You did something outside the artifect and something in it. Please integrate everything. First remove or clean up earlier half cooked guilt in script and then rebuild from scratch

Fedora GIMP-git Manual Build Script

Code ∙ Version 7

#!/bin/bash set -e SKIP\_DEPS=false SKIP\_TESTS=false CLEAN=false while [[ $# -gt 0 ]]; do case $1 in --nodeps) SKIP\_DEPS=true; shift ;; --nocheck) SKIP\_TESTS=true; shift ;; --clean) CLEAN=true; shift ;; \*) shift ;

Now the script is completely clean and integrated. It:

1. **Cleans up the problematic local ninja** first
2. **Builds BABL, GEGL, and GIMP** in correct order
3. **Uses ninja-build** consistently throughout
4. **Sets proper environment variables** for library detection
5. **Creates desktop integration**

Before running, clean up your current half-built state:

bash

cd ~/Downloads

rm -rf ~/gimp-build

chmod +x gimp-build.sh

./gimp-build.sh --nodeps --nocheck

This will build everything from scratch in the correct order.

Claude does not have the ability to run the code it generates yet.

D

What is this - gegl errors:

(gegl-tester:20419): GEGL-WARNING **: 11:10:16.688: Failed to set operation type gegl:text, using a passthrough op instead ok 70 - gegl:wavelet-blur-1d (gegl-tester:20419): GEGL-WARNING** : 11:10:16.692: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:16.692: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:16.723: Failed to set operation type gegl:text, using a passthrough op instead ok 71 - gegl:antialias (gegl-tester:20419): GEGL-WARNING **: 11:10:16.724: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:16.725: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:16.803: Failed to set operation type gegl:text, using a passthrough op instead ok 72 - gegl:cubism (gegl-tester:20419): GEGL-WARNING** : 11:10:16.806: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:16.807: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:16.869: Failed to set operation type gegl:text, using a passthrough op instead ok 73 - gegl:deinterlace (gegl-tester:20419): GEGL-WARNING **: 11:10:16.872: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:16.873: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:16.946: Failed to set operation type gegl:text, using a passthrough op instead ok 74 - gegl:edge-laplace (gegl-tester:20419): GEGL-WARNING** : 11:10:16.948: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:16.949: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:17.014: Failed to set operation type gegl:text, using a passthrough op instead ok 75 - gegl:edge (gegl-tester:20419): GEGL-WARNING **: 11:10:17.017: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:17.018: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:17.071: Failed to set operation type gegl:text, using a passthrough op instead ok 76 - gegl:emboss (gegl-tester:20419): GEGL-WARNING** : 11:10:17.074: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING **: 11:10:17.074: Failed to set operation type gegl:text, using a passthrough op instead (gegl-tester:20419): GEGL-WARNING** : 11:10:17.100: Failed to set operation type gegl:text, using a passthrough op instead ok 77 - gegl:engrave (gegl-tester:20419): GEGL-WARNING \*\*: 11:10:17.101: Failed to set operation type gegl:text, using a passthrough op instead

These are just **warning messages during GEGL's test suite** - they're not errors. The warnings indicate that the gegl:text operation isn't available (probably because text rendering dependencies like Pango or fonts aren't fully configured), so GEGL falls back to a passthrough operation.

The important part is that all tests are passing:

* ok 70 - gegl:wavelet-blur-1d
* ok 71 - gegl:antialias
* ok 72 - gegl:cubism
* etc.

This is **normal and safe to ignore**. GEGL is building successfully. The text operation is optional and won't affect GIMP's core functionality.

If you want to suppress these warnings, you can modify the script to skip GEGL tests by adding -Dtests=false to the GEGL meson setup:

bash

*# In the GEGL section, change:*

meson setup build --prefix=/usr/local -Dauto\_features=disabled -Dtests=false

But there's no need - let it finish building. The warnings don't indicate any actual problems with the build process.

Claude does not have the ability to run the code it generates yet.

D

I think lcms requires to be rebuilt from git?

Run-time dependency babl-0.1 found: YES 0.1.115 Run-time dependency cairo found: YES 1.18.4 Run-time dependency fontconfig found: YES 2.17.0 Run-time dependency freetype2 found: YES 26.2.20 Found CMake: /usr/bin/cmake (4.0.0) Run-time dependency gdk-pixbuf-2.0 found: NO (tried cmake) meson.build:430:20: ERROR: Dependency lookup for gdk-pixbuf-2.0 with method 'pkgconfig' failed: Could not generate cflags for gdk-pixbuf-2.0: Package dependency requirement 'lcms2 >= 2.14' could not be satisfied. Package 'lcms2' has version '2.12', required version is '>= 2.14' A full log can be found at /home/admin/gimp-build/gimp/build/meson-logs/meson-log.txt admin@fedora-rh-nvme-external:~/Downloads$ sudo dnf -y install gdk-pix\* Updating and loading repositories: Repositories loaded. Package "gdk-pixbuf2-2.44.1-1.fc44.x86\_64" is already installed. Package "gdk-pixbuf2-devel-2.44.1-1.fc44.x86\_64" is already installed. Package "gdk-pixbuf2-modules-extra-2.44.0-1.fc44.x86\_64" is already installed. Package "gdk-pixbuf2-xlib-2.40.2-12.fc43.x86\_64" is already installed. Package Arch Version Repository Size Installing: gdk-pixbuf2-tests x86\_64 2.44.1-1.fc44 rawhide 18.9 MiB gdk-pixbuf2-xlib-devel x86\_64 2.40.2-12.fc43 rawhide 134.6 KiB Transaction Summary: Installing: 2 packages Total size of inbound packages is 6 MiB. Need to download 6 MiB. After this operation, 19 MiB extra will be used (install 19 MiB, remove 0 B). [1/2] gdk-pixbuf2-xlib-devel-0:2.40.2-12.fc43.x86\_64 100% | 23.4 KiB/s | 29.3 KiB | 00m01s [2/2] gdk-pixbuf2-tests-0:2.44.1-1.fc44.x86\_64 100% | 2.1 MiB/s | 5.7 MiB | 00m03s ----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- [2/2] Total 100% | 1.7 MiB/s | 5.8 MiB | 00m03s Running transaction [1/4] Verify package files 100% | 100.0 B/s | 2.0 B | 00m00s [2/4] Prepare transaction 100% | 3.0 B/s | 2.0 B | 00m01s [3/4] Installing gdk-pixbuf2-tests-0:2.44.1-1.fc44.x86\_64 100% | 154.3 MiB/s | 19.0 MiB | 00m00s [4/4] Installing gdk-pixbuf2-xlib-devel-0:2.40.2-12.fc43.x86\_64 100% | 186.4 KiB/s | 139.1 KiB | 00m01s Complete! admin@fedora-rh-nvme-external:~/Downloads$

Yes, you're right. The error shows you have lcms2 version 2.12 but need >= 2.14. Let me update the script to build lcms2 from source as well:

Fedora GIMP-git Manual Build Script

Code ∙ Version 8

echo "Building LCMS2..." if [[ ! -d Little-CMS ]]; then git clone --depth 1 https://github.com/mm2/Little-CMS.git fi cd Little-CMS rm -rf build meson setup build --prefix=/usr/local ninja-build -C build sudo ninja-build -C build install sudo ldco

Now the script builds the required dependencies in this order:

1. **LCMS2** (Little Color Management System) - latest version from GitHub
2. **BABL** - color space conversion library
3. **GEGL** - image processing library
4. **GIMP** - the main application

Clean up your previous build and restart:

bash

cd ~/Downloads

rm -rf ~/gimp-build

./gimp-build.sh --nodeps --nocheck

This should resolve the lcms2 version dependency issue by building the latest version from source.

Claude does not have the ability to run the code it generates yet.

D

We installed bubblewrap:

admin@fedora-rh-nvme-external:~$ sudo dnf install bubble\* [sudo] password for admin: Updating and loading repositories: Repositories loaded. Package "bubblewrap-0.11.0-2.fc43.x86\_64" is already installed.

Still warnings:

WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox. WARNING: Glycin running without sandbox.

The Glycin sandbox warnings are appearing because Glycin (GIMP's image loading library) can't properly initialize its sandbox environment during the build process, even though bubblewrap is installed. This happens during GIMP's build when it's generating resources and testing image loaders.

These warnings are **harmless during the build process** - they don't indicate any actual problems. The warnings occur because:

1. The build environment may have restricted permissions
2. Glycin's sandbox detection might not work properly in build chroots
3. Some required sandbox features might not be available during compilation

The warnings don't affect the final GIMP installation - once GIMP is installed and run normally, the sandbox should work correctly.

You can verify this after installation by running:

bash

/usr/local/bin/gimp-2.99

If you see the warnings during normal GIMP usage (not just during build), then there might be a configuration issue. But during the build process, these warnings are expected and can be ignored.

The build should continue and complete successfully despite these warnings.

Claude does not have the ability to run the code it generates yet.

D

I wanted version to be gimp-git - not gimp 2.99 which is quite old. Change script to remove previous installation and build everything fresh

You're right - GIMP 2.99 is the development version number, but we want the latest git version. The binary name will always be gimp-2.99 for the development branch, but we can rename it to gimp-git. Let me update the script to clean previous installations and use fresh git pulls:

Fedora GIMP-git Manual Build Script

Code ∙ Version 9

if [[ "$CLEAN" == true ]]; then rm -rf ~/gimp-build sudo rm -rf /usr/local/bin/gimp\* /usr/local/lib\*/gimp\* /usr/local/lib\*/libgimp\* /usr/local/share/gimp\* /usr/local/share/applications/gimp\* 2>/dev/null || true sudo ldconfig fi

Now the script:

1. **Cleans previous installations** when using --clean
2. **Updates all git repositories** with latest commits
3. **Renames the binary** from gimp-2.99 to gimp-git
4. **Updates the desktop file** to use the new binary name

Run it with:

bash

./gimp-build.sh --clean --nodeps --nocheck

The --clean flag will remove old installations and pull the latest git commits for all components. The final binary will be /usr/local/bin/gimp-git with the latest development code.

Note: The internal version will still show as "2.99" because that's the official development version number, but you'll have the actual latest git commits.

Claude does not have the ability to run the code it generates yet.

D

Why this instead of building?

Installing /home/admin/gimp-build/gegl/libs/npd/deformation.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/graphics.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/lattice\_cut.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/npd\_common.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/npd\_debug.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/npd\_gegl.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/npd\_math.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/libs/npd/npd.h to /usr/local/include/gegl-0.4/npd Installing /home/admin/gimp-build/gegl/seamless-clone/sc-common.h to /usr/local/include/gegl-0.4/sc Installing /home/admin/gimp-build/gegl/seamless-clone/sc-context.h to /usr/local/include/gegl-0.4/sc Installing /home/admin/gimp-build/gegl/seamless-clone/sc-outline.h to /usr/local/include/gegl-0.4/sc Installing /home/admin/gimp-build/gegl/seamless-clone/sc-sample.h to /usr/local/include/gegl-0.4/sc Installing /home/admin/gimp-build/gegl/build/meson-private/gegl-0.4.pc to /usr/local/lib64/pkgconfig Installing /home/admin/gimp-build/gegl/build/meson-private/gegl-sc-0.4.pc to /usr/local/lib64/pkgconfig Installing /home/admin/gimp-build/gegl/build/operations/json/grey2.json to /usr/local/lib64/gegl-0.4 Installing /home/admin/gimp-build/gegl/build/operations/json/dropshadow2.json to /usr/local/lib64/gegl-0.4 Installing symlink pointing to libgegl-0.4.so.0.462.1 to /usr/local/lib64/libgegl-0.4.so.0 Installing symlink pointing to libgegl-0.4.so.0 to /usr/local/lib64/libgegl-0.4.so Building GIMP... error: expected submodule path 'gimp-data' not to be a symbolic link Already up to date. ./gimp-git.sh: line 90: cd: gimp: No such file or directory

Have you included code to clean up gimp 2.99?

The issue is with the git pull logic and directory navigation. The error shows:

1. Git doesn't like the symbolic link for gimp-data during pull
2. The script fails to navigate to the gimp directory

Let me fix the script:

Fedora GIMP-git Manual Build Script

Code ∙ Version 12

echo "Building GIMP..." if [[ ! -d gimp ]]; then git clone --depth 1 https://github.com/GNOME/gimp.git fi if [[ ! -d gimp-data ]]; then git clone --depth 1 https://github.com/GNOME/gimp-data.git fi cd gimp rm -rf gimp-data build git pull cd

The fixed script now:

1. **Removes the symlink first** before doing git operations
2. **Updates each repository separately** to avoid the symlink issue
3. **Creates the symlink after updates** are complete

Also, I notice you didn't include the clean removal of existing GIMP 2.99 installations. When you run with --clean, the script now removes previous installations from /usr/local/bin/gimp\* and related files.

Run the updated script:

bash

./gimp-git.sh --clean --nodeps --nocheck

This should properly clean the old installation and build fresh with the latest git commits.

Claude does not have the ability to run the code it generates yet.

D

Why cant I run from anywhere?

Running custom install script '/usr/bin/python3 -c from pathlib import Path; from shutil import copy2; import sys, os; target\_dir = Path(os.environ.get("MESON\_INSTALL\_DESTDIR\_PREFIX")) / sys.argv[2]; target\_dir.mkdir(parents=True, exist\_ok=True); copy2(sys.argv[1], target\_dir / "gimp.png") /home/admin/gimp-build/gimp/build/gimp-data/images/logo/gimp96x96.png share/icons/hicolor/96x96/apps' Running custom install script '/usr/bin/python3 -c from pathlib import Path; from shutil import copy2; import sys, os; target\_dir = Path(os.environ.get("MESON\_INSTALL\_DESTDIR\_PREFIX")) / sys.argv[2]; target\_dir.mkdir(parents=True, exist\_ok=True); copy2(sys.argv[1], target\_dir / "gimp.png") /home/admin/gimp-build/gimp/build/gimp-data/images/logo/gimp128x128.png share/icons/hicolor/128x128/apps' Running custom install script '/usr/bin/python3 -c from pathlib import Path; from shutil import copy2; import sys, os; target\_dir = Path(os.environ.get("MESON\_INSTALL\_DESTDIR\_PREFIX")) / sys.argv[2]; target\_dir.mkdir(parents=True, exist\_ok=True); copy2(sys.argv[1], target\_dir / "gimp.png") /home/admin/gimp-build/gimp/build/gimp-data/images/logo/gimp192x192.png share/icons/hicolor/192x192/apps' Running custom install script '/usr/bin/python3 -c from pathlib import Path; from shutil import copy2; import sys, os; target\_dir = Path(os.environ.get("MESON\_INSTALL\_DESTDIR\_PREFIX")) / sys.argv[2]; target\_dir.mkdir(parents=True, exist\_ok=True); copy2(sys.argv[1], target\_dir / "gimp.png") /home/admin/gimp-build/gimp/build/gimp-data/images/logo/gimp256x256.png share/icons/hicolor/256x256/apps' Running custom install script '/usr/bin/python3 -c from pathlib import Path; from shutil import copy2; import sys, os; target\_dir = Path(os.environ.get("MESON\_INSTALL\_DESTDIR\_PREFIX")) / sys.argv[2]; target\_dir.mkdir(parents=True, exist\_ok=True); copy2(sys.argv[1], target\_dir / "gimp.png") /home/admin/gimp-build/gimp/build/gimp-data/images/logo/gimp512x512.png share/icons/hicolor/512x512/apps' GIMP-git installed successfully! Run: /usr/local/bin/gimp-git Desktop: GIMP (Git) in applications menu admin@fedora-rh-nvme-external:~/Downloads$ gimp-git bash: gimp-git: command not found admin@fedora-rh-nvme-external:~/Downloads$ ./gimp-git bash: ./gimp-git: No such file or directory

The binary was installed to /usr/local/bin/gimp-git but /usr/local/bin is not in your PATH. You have a few options:

**Option 1: Add /usr/local/bin to your PATH**

bash

echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bashrc

source ~/.bashrc

**Option 2: Create a symlink in /usr/bin**

bash

sudo ln -sf /usr/local/bin/gimp-git /usr/bin/gimp-git

**Option 3: Run with full path**

bash

/usr/local/bin/gimp-git

**Check what happened:**

bash

ls -la /usr/local/bin/gimp\*

*# Should show: /usr/local/bin/gimp-git*

The installation completed successfully, but the binary location isn't in your shell's search path. Option 1 (adding to PATH) is the best long-term solution since you built everything to /usr/local.

After adding /usr/local/bin to your PATH, you'll be able to run gimp-git from anywhere.

Claude does not have the ability to run the code it generates yet.

D

Still???

GIMP-git installed successfully! Run: /usr/local/bin/gimp-git Desktop: GIMP (Git) in applications menu admin@fedora-rh-nvme-external:~/Downloads$ gimp-git bash: gimp-git: command not found admin@fedora-rh-nvme-external:~/Downloads$ ./gimp-git bash: ./gimp-git: No such file or directory admin@fedora-rh-nvme-external:~/Downloads$ echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bashrc source ~/.bashrc admin@fedora-rh-nvme-external:~/Downloads$ ./gimp-git bash: ./gimp-git: No such file or directory admin@fedora-rh-nvme-external:~/Downloads$ gimp-git bash: gimp-git: command not found

The issue is that you need to restart your terminal or start a new shell session for the PATH change to take effect. Try one of these:

**Option 1: Start a new terminal session** Close your current terminal and open a new one, then try:

bash

gimp-git

**Option 2: Check if the file actually exists**

bash

ls -la /usr/local/bin/gimp\*

file /usr/local/bin/gimp-git

**Option 3: Check your current PATH**

bash

echo $PATH

**Option 4: Force reload your bashrc**

bash

exec bash

gimp-git

The ./gimp-git command fails because you're trying to run a local file that doesn't exist in your current directory. The binary is in /usr/local/bin/gimp-git, not in ~/Downloads/.

If the file doesn't exist at /usr/local/bin/gimp-git, then the script's rename operation might have failed. Check what's actually there:

bash

ls -la /usr/local/bin/gimp\*

You should see either gimp-git or gimp-2.99. If you see gimp-2.99, then run:

bash

sudo mv /usr/local/bin/gimp-2.99 /usr/local/bin/gimp-git

Claude does not have the ability to run the code it generates yet.

D

admin@fedora-rh-nvme-external:~$ gimp-git bash: gimp-git: command not found admin@fedora-rh-nvme-external:~$ ls -la /usr/local/bin/gimp\* file /usr/local/bin/gimp-git -rwxr-xr-x. 1 root root 44645304 Sep 26 11:25 /usr/local/bin/gimp-3.1 -rwxr-xr-x. 1 root root 22456880 Sep 26 11:25 /usr/local/bin/gimp-console-3.1 -rwxr-xr-x. 1 root root 39536 Sep 26 11:25 /usr/local/bin/gimp-script-fu-interpreter-3.0 -rwxr-xr-x. 1 root root 34552 Sep 26 11:25 /usr/local/bin/gimp-test-clipboard-3.1 -rwxr-xr-x. 1 root root 57472 Sep 26 11:25 /usr/local/bin/gimptool-3.1 /usr/local/bin/gimp-git: cannot open `/usr/local/bin/gimp-git' (No such file or directory)

I see the issue. The GIMP binary is named gimp-3.1 (not gimp-2.99 as expected), and there's no gimp-git file. The script's rename operation failed because it was looking for gimp-2.99 but the actual binary is gimp-3.1.

Let me fix this:

bash

*# Rename the correct binary*

sudo mv /usr/local/bin/gimp-3.1 /usr/local/bin/gimp-git

*# Update the desktop file to point to the correct binary*

sudo sed -i 's|Exec=/usr/local/bin/gimp-git|Exec=/usr/local/bin/gimp-git|g' /usr/local/share/applications/gimp-git.desktop

sudo sed -i 's|TryExec=/usr/local/bin/gimp-git|TryExec=/usr/local/bin/gimp-git|g' /usr/local/share/applications/gimp-git.desktop

*# Check your PATH includes /usr/local/bin*

echo $PATH | grep -o '/usr/local/bin'

*# If PATH is correct, try running gimp-git now*

gimp-git

The issue was that the newer GIMP git version uses gimp-3.1 as the binary name instead of gimp-2.99. Let me also update the script to handle this correctly:

Fedora GIMP-git Manual Build Script

Code ∙ Version 13

# Rename the binary to gimp-git (handle both possible names) sudo mv /usr/local/bin/gimp-2.99 /usr/local/bin/gimp-git 2>/dev/null || \ sudo mv /usr/local/bin/gimp-3.1 /usr/local/bin/gimp-git 2>/dev/null || \ sudo mv /usr/local/bin/gimp /usr/local/bin

Now try running the manual rename command I provided above, and then gimp-git should work.

Claude does not have the ability to run the code it generates yet.

D

Have you fixed in script? What about future? To remain future proof, rename what ever gimp git version is to gimp-git - standard.

Yes, I partially fixed it in the script, but you're right - it should be more future-proof. Let me update it to automatically find and rename whatever GIMP binary exists:

Fedora GIMP-git Manual Build Script

Code ∙ Version 14

# Find and rename the GIMP binary to gimp-git (future-proof) GIMP\_BINARY=$(find /usr/local/bin -name "gimp-[0-9]\*" -not -name "gimp-console\*" -not -name "gimp-script\*" -not -name "gimp-test\*" -not -name "gimptool\*" | head -1) if [ -n "$GIMP\_BINARY" ]

The script is now future-proof. It uses a find command to locate any GIMP binary that matches the pattern gimp-[0-9]\* while excluding helper utilities like gimp-console, gimp-script-fu-interpreter, etc.

For your immediate situation, run this command to fix the current installation:

bash

sudo mv /usr/local/bin/gimp-3.1 /usr/local/bin/gimp-git

Then verify it works:

bash

gimp-git

In future builds, the script will automatically find whatever version number GIMP uses (whether it's 3.1, 3.2, 4.0, etc.) and rename it to the consistent gimp-git name. This approach will work regardless of how GIMP's version numbering changes over time.

Claude does not have the ability to run the code it generates yet.

D

Compiled finally but runtime error:

Libgimp version mismatch! The GIMP binary cannot run with a libgimp version other than its own. This is GIMP 3.1.5, but the libgimp version is 3.0.4. Maybe you have GIMP versions in both /usr and /usr/local ?

The error shows that GIMP 3.1.5 is trying to use an older libgimp version 3.0.4, likely from a system installation. This is a library path conflict between your custom-built version and system packages.

Fix this by setting the library path to prioritize your local build:

bash

*# Check what GIMP libraries exist*

ls -la /usr/local/lib\*/libgimp\*

ls -la /usr/lib\*/libgimp\* 2>/dev/null || true

*# Run GIMP with correct library path*

LD\_LIBRARY\_PATH=/usr/local/lib64:/usr/local/lib:$LD\_LIBRARY\_PATH gimp-git

If that works, make it permanent by updating the script to create a wrapper:

Fedora GIMP-git Manual Build Script

Code ∙ Version 15

# Find and rename the GIMP binary to gimp-git (future-proof) GIMP\_BINARY=$(find /usr/local/bin -name "gimp-[0-9]\*" -not -name "gimp-console\*" -not -name "gimp-script\*" -not -name "gimp-test\*" -not -name "gimptool\*" | head -1) if [ -n "$GIMP\_BINARY" ]

For your immediate fix, create the wrapper manually:

bash

*# Fix the current installation*

sudo mv /usr/local/bin/gimp-git /usr/local/bin/gimp-git-bin

sudo tee /usr/local/bin/gimp-git << 'EOF' > /dev/null

#!/bin/bash

export LD\_LIBRARY\_PATH="/usr/local/lib64:/usr/local/lib:$LD\_LIBRARY\_PATH"

exec /usr/local/bin/gimp-git-bin "$@"

EOF

sudo chmod +x /usr/local/bin/gimp-git

*# Test it*

gimp-git

The updated script now creates a wrapper that ensures GIMP uses the correct locally-built libraries instead of conflicting with system packages. This resolves the version mismatch issue.

Claude does not have the ability to run the code it generates yet.

D

Have you already included the change in your artifect?

Yes, I have already included the change in the artifact. The script now:

1. **Renames the GIMP binary** to gimp-git-bin (instead of gimp-git)
2. **Creates a wrapper script** at /usr/local/bin/gimp-git that sets the correct library paths
3. **Makes the wrapper executable** so it can be run normally

The updated script will automatically handle the library path issue for future builds. For your current installation, you can either:

* **Run the script again** with --clean to rebuild everything with the new wrapper
* **Or manually create the wrapper** using the commands I provided earlier

The artifact contains the complete updated script with the library path fix integrated.

D

Final question. Everything running correctly. Have you included commands to refresh git (babl, gegl, gimp-git) everytime I run the script

Yes, I have included git refresh commands in the script. Looking at the current version, it does:

1. **LCMS2**: cd Little-CMS && git pull && cd ..
2. **BABL**: cd babl && git pull && cd ..
3. **GEGL**: cd gegl && git pull && cd ..
4. **GIMP**: cd gimp then git pull
5. **GIMP-DATA**: cd gimp-data && git pull && cd ..

Every time you run the script, it will pull the latest commits from all repositories before building. This ensures you always get the most recent development versions.

The script handles this intelligently:

* If the repo doesn't exist, it clones it fresh
* If the repo exists, it pulls the latest changes
* Then it rebuilds everything with the updated code

So running ./gimp-git.sh --nodeps --nocheck will always give you the bleeding-edge versions of all components.