

How to build Nitro locally (Debian, Ubuntu, MacOS)

PUBLIC PREVIEW DOCUMENT This document is currently in public preview and may change significantly as feedback is captured from readers like you. Click the [Request an update](#) button at the top of this document or [join the Arbitrum Discord](#) to share your feedback. Arbitrum Nitro is the software that powers all Arbitrum chains. This how-to shows how you can build a Docker image, or binaries, directly from Nitro's source code. If you want to run a node for one of the Arbitrum chains, however, it is recommended that you use the docker image available on DockerHub, as explained in [How to run a full node](#).

This how-to assumes that you're running one of the following operating systems:

- [Debian 11.7 \(arm64\)](#)
- [Ubuntu 22.04 \(amd64\)](#)
- [MacOS Ventura 13.4](#)
- .

Build a Docker image

Step 1. Configure [Docker](#)

For [Debian](#)

[/Ubuntu](#)

for

pkg

in docker.io docker-doc docker-compose podman-docker containerd runc ;

do

sudo

apt-get remove pkg ;

done

Add Docker's official GPG key:

sudo

apt-get update sudo

apt-get

install ca-certificates curl gnupg sudo

install -m 0755 -d /etc/apt/keyrings curl -fsSL https://download.docker.com/linux/debian/gpg |

sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg sudo

chmod a+r /etc/apt/keyrings/docker.gpg

Add the repository to Apt sources:

echo

```
\ "deb [arch=" ( dpkg --print-architecture ) " signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \ " ( . /etc/os-release &&
```

echo

```
" VERSION_CODENAME " ) " stable"
```

|

```
\ sudo
```

```
tee /etc/apt/sources.list.d/docker.list
```

```
/dev/null sudo
```

```
apt-get update sudo
```

```
apt-get
```

```
install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin sudo
```

```
service
```

```
docker start
```

For [MacOS](#)

Depending on whether your Mac has an Intel processor or Apple silicon, download the corresponding disk image from [Docker](#) , and move it into your Applications folder.

[Optional]Run docker from a different user

After installing docker, you might want to be able to run it with your current user instead of root. You can run the following commands to do so.

```
sudo
```

```
groupadd
```

```
docker sudo
```

```
usermod -aG docker
```

USER newgrp docker For troubleshooting, check Docker's section in [their documentation](#)

Step 2. Download the Nitro source code

```
git clone --branch v2.2.5 https://github.com/OffchainLabs/nitro.git cd nitro git submodule update --init --recursive --force
```

Step 3. Build the Nitro node Docker image

`docker build . --tag nitro-node` That command will build a Docker image called `nitro-node` from the local source.

Build Nitro's binaries natively

If you want to build the node binaries natively, execute steps 1-3 of the [Build a Docker image](#) section and continue with the steps described here. Notice that even though we are building the binaries outside of Docker, it is still used to help build some WebAssembly components.

Step 4. Configure prerequisites

For Debian/Ubuntu

```
apt
```

```
install
```

```
git
```

```
curl build-essential cmake npm golang clang make gotestsum wabt lld-13 python3 npm
```

```
install --global yarn ln -s /usr/bin/wasm-ld-13 /usr/local/bin/wasm-ld
```

For MacOS

Install [Homebrew](#) package manager and add it to your `PATH` environment variable:

```
/bin/bash -c " ( curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh ) " echo
```

```
"export PATH=/opt/homebrew/bin: PATH "
```

```
~/.zprofile &&
```

source ~/.zprofile (replace ~/.zprofile with ~/.bash_profile if you use bash instead of zsh).

Install essentials:

```
brew install
```

```
git
```

```
curl
```

```
make cmake npm go gvm golangci-lint wabt llvm gotestsum npm
```

```
install --global yarn sudo
```

```
mkdir -p /usr/local/bin sudo
```

```
ln -s /opt/homebrew/opt/llvm/bin/wasm-ld /usr/local/bin/wasm-ld
```

Step 5. Configure node[16.19](#)

For Debian/Ubuntu

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh |
```

```
bash source
```

```
" HOME ~/.bashrc" nvm install
```

```
16.19 nvm use 16.19
```

For MacOS

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh |
```

```
bash export
```

NVM_DIR

```
" HOME ~/.nvm" [ -s " NVM_DIR /nvm.sh"
```

```
]
```

```
&&
```

```
\ . " NVM_DIR /nvm.sh" nvm install
```

```
16.19 nvm use 16.19
```

Step 6. Configure Rust[1.72.1](#)

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs |
```

```
sh source
```

```
" HOME ~/.cargo/env" rustup install
```

```
1.72 .1 rustup default 1.72 .1 rustup target add wasm32-unknown-unknown --toolchain 1.72 .1 rustup target add wasm32-wasi --toolchain 1.72 .1 cargo install cbindgen
```

Step 7. Configure Go[1.20](#)

Install Bison

For Debian/Ubuntu

```
sudo  
apt-get  
install bison
```

For MacOS

```
brew install bison
```

Install and configure Go

```
bash  
  
<  
< ( curl -s -S -L https://raw.githubusercontent.com/moovweb/gvm/master/binscripts/gvm-installer ) source  
  
" HOME /.gvm/scripts/gvm" gvm install go1.20 gvm use go1.20 --default curl -sSfL  
https://raw.githubusercontent.com/golangci/golangci-lint/master/install.sh |  
  
sh -s -- -b ( go env GOPATH ) /bin v1.54.2 If you use zsh, replace bash with zsh .
```

Step 8. Start build

```
make
```

Step 9. Produce binaries

```
make build
```

Step 10. Run your node

To run your node using the generated binaries, use the following command from the nitro folder, with your desired parameters

```
./target/bin/nitro < node parameters
```

[Edit this page](#) Last updated on Mar 7, 2024 [Previous](#) [How to run a Sequencer Coordinator Manager \(SQM\)](#) [Next](#)
[How to migrate state and history from a classic \(pre-Nitro\) node to a Nitro node](#)