

Forgejo Runner Configuration



- [Choosing labels](#)
 - [Docker or Podman](#)
 - [LXC](#)
 - [Host](#)
 - [Special labels](#)
 - [Enabling IPv6 in Docker & Podman networks](#)
 - [IPv6 connectivity issues with rootless podman](#)
 - [Cache configuration](#)
 - [Configuration file reference](#)
-

Choosing labels

When Forgejo identifies a job that needs to be run, it inspects the jobs' `runs-on` field to evaluate which `Forgejo Runner` instances are capable of running the job. `runs-on` is typically a single value like `ubuntu-latest`, but in more complex configurations it can also contain [multiple values](#).

When configuring a Forgejo Runner, labels must be defined in the configuration file which indicate what jobs this runner should execute. For example:

```
# ... the rest of the config file ...

runner:
  # ... other runner options ...
  labels:
    - debian:docker://docker.io/library/node:lts
    - ubuntu-latest:docker://ghcr.io/catthehacker/ubuntu:act-22.04
```

```
# ... the rest of the config file ...
```

This would cause a job with `runs-on: debian` to run on this Forgejo Runner within the docker image `docker://docker.io/library/node:lts`.

A label has the following structure:

```
<label-name>:<label-type>://<default-image>
```

The `label name` is a unique string that identifies the label. It is the part that is specified in the `runs-on` field of workflows to choose which runners the workflow can be executed on.

The `label type` determines what containerization system will be used to run the workflow. There are three options:

1. `docker` for [Docker or Podman](#) containerization,
2. `lxc` for [LXC](#) containerization,
3. `host` for [no containerization](#).

Docker or Podman

⚠ [!caution] Caution When starting a container, Forgejo Runner does the equivalent of `docker run <image>`. That means that container images, once downloaded, are **never updated**.

If a label specifies `docker` as its `label type`, the rest of it is interpreted as the default container image to use. The runner will execute all the steps, as root, within a container created from that image.

Label examples:

- `node20:docker://node:20-bookworm` defines `node20` to be the `node:20-bookworm` image from [hub.docker.com](#).
- `node20:docker://docker.io/node:20-bookworm` would have the same behaviour, fully defining the host to fetch the image from.
- `docker:docker://data.forgejo.org/oci/alpine:3.20` defines `docker` to be the `alpine:3.20` image from <https://data.forgejo.org/oci/-/packages/container/alpine/3.20>.

It is generally recommended to pin image versions instead of relying on tags, for example,

`debian:docker://node@sha256:91447bc57243b852a21e0ff3553f531f0d4b66257a564b106c79d9e00f3aa14e` instead of `debian:docker://node:lts`. That is a prerequisite for reproducible jobs.

To mimic GitHub runners, a label such as `ubuntu-latest:docker://node:lts` can be used. With this, the Forgejo Runner will respond to `runs-on: ubuntu-latest` and will use the `node:lts` image from hub.docker.com. This image is quite capable of running many of the workflows that are designed for the GitHub runners, but not all the same tools will be available as GitHub provides.

Many common actions (for example, `uses: actions/checkout@v6`) require Node.js in the image.

The container image specified in the label can be overridden by a workflow by using `jobs.<job_id>.container`.

LXC

If a label specifies `lxc` as its `label type`, the rest of it is interpreted as `template[:release[:lxc-helper config]]` where:

- `template[:release]` is the [template and release](#) to use.
- `lxc-helper config` is the value of the [--config option of lxc-helper](#) used when creating a container.

The runner will execute all the steps, as root, within a [LXC container](#) created from that template and release. The default template is `debian` and the default release is `bullseye`.

[nodejs](#) version 20 is installed.

Label examples:

- `bookworm:lxc://debian:bookworm:lxc docker` defines `bookworm` to be an LXC container running Debian GNU/Linux bookworm. It has the necessary capabilities to run a nested LXC container and a Docker engine.
- `bookworm:lxc://debian:bookworm` defines `bookworm` to be an LXC container running Debian GNU/Linux bookworm. It has the necessary capabilities to run a nested LXC container, KVM virtual machines and a Docker engine.

Host

If a label specifies `host` as its `label type`, the runner will execute all the steps in a shell forked from the runner, directly on the host.

⚠️ [!warning] Warning There is no isolation at all and a single job can permanently destroy the host.

Label example:

- `self-hosted:host` defines jobs that have `runs-on: self-hosted` to run without any container isolation.

Special labels

Runner labels can also be used to define other special features a runner has.

For example, if three runners were available for executing jobs and all of them defined the `docker::docker://node:20-bookworm` label, then jobs would be distributed between each of the three runners as they are available. If only a single one of these runners had a special hardware capability like a GPU, there may be jobs that require that capability and can't be run on the other systems.

The solution in this case is to add multiple labels to a job's `runs-on` field:

```
on: pull_request
jobs:
  my-job:
    runs-on: [docker, gpu]
    # ...
```

And then to specify both the `docker:...` and `gpu:...` labels in the Forgejo Runner configuration:

```
runner:
  labels:
    - docker:docker://ghcr.io/catthehacker/ubuntu:act-22.04
    - gpu:docker://ghcr.io/catthehacker/ubuntu:act-22.04
```

Only a Forgejo Runner with **both** the `docker` and `gpu` labels will be able to run the job. The job will be run on the containerization platform that is listed in the first label in the `runs-on` array.

Enabling IPv6 in Docker & Podman networks

When a Forgejo Runner creates its own Docker or Podman networks, IPv6 is not enabled by default, and must be enabled explicitly in the Forgejo Runner configuration.

Docker only: The Docker daemon requires additional configuration to enable IPv6. To make use of IPv6 with Docker, you need to provide a `/etc/docker/daemon.json` configuration file with at least the following keys:

```
{
  "ipv6": true,
  "experimental": true,
  "ip6tables": true,
  "fixed-cidr-v6": "fd00:d0ca:1::/64",
  "default-address-pools": [
    { "base": "172.17.0.0/16", "size": 24 },
    { "base": "fd00:d0ca:2::/104", "size": 112 }
  ]
}
```

Afterwards restart the Docker daemon with `systemctl restart docker.service`.

⚠️ [!note] Note These are example values. While this setup should work out of the box, it may not meet your requirements. Please refer to the Docker documentation regarding [enabling IPv6](#) and [allocating IPv6 addresses to subnets dynamically](#).

Docker & Podman: To test IPv6 connectivity in Forgejo-Runner-created networks, create a small workflow such as the following:

```
---
on: push
jobs:
  ipv6:
    runs-on: docker
    steps:
      - run: |
          apt update; apt install --yes curl
          curl -s -o /dev/null http://ipv6.google.com
```

If you run this action with `forgejo-runner exec`, you should expect this job fail:

```
$ forgejo-runner exec
...
```

```
| curl: (7) Couldn't connect to server
[ipv6.yml/ipv6]   ❌ Failure - apt update; apt install --yes curl
curl -s -o /dev/null http://ipv6.google.com
[ipv6.yml/ipv6] exitcode '7': failure
[ipv6.yml/ipv6] Cleaning up services for job ipv6
[ipv6.yml/ipv6] Cleaning up container for job ipv6
[ipv6.yml/ipv6] Cleaning up network for job ipv6, and network name is: FORGEJO-ACTIONS-TASK-
push_WORKFLOW-ipv6-yml_JOB-ipv6-network
[ipv6.yml/ipv6]   ❌ Job failed
```

To actually enable IPv6 with `forgejo-runner exec`, the flag `--enable-ipv6` must be provided. If you run this again with `forgejo-runner exec --enable-ipv6`, the job should succeed:

```
$ forgejo-runner exec --enable-ipv6
...
[ipv6.yml/ipv6]   ✅ Success - Main apt update; apt install --yes curl
curl -s -o /dev/null http://ipv6.google.com
[ipv6.yml/ipv6] Cleaning up services for job ipv6
[ipv6.yml/ipv6] Cleaning up container for job ipv6
[ipv6.yml/ipv6] Cleaning up network for job ipv6, and network name is: FORGEJO-ACTIONS-TASK-
push_WORKFLOW-ipv6-yml_JOB-ipv6-network
[ipv6.yml/ipv6]   ✅ Job succeeded
```

Finally, if this test was successful, enable IPv6 in the `runner-config.yml` file of the Forgejo Runner daemon and restart the daemon:

```
container:
  enable_ipv6: true
```

Now, Forgejo Runner will create networks with IPv6 enabled, and workflow containers will be assigned addresses from the pools defined in the Docker daemon configuration.

IPv6 connectivity issues with rootless podman

Because creation of real networks is limited to the root user, rootless podman cannot create actual networks. To work around this issue, podman creates so-called tap-networks which come with [their own limitations](#). At the time of writing, only Podman version 5.3 and later have been observed to have proper IPv6 support in rootless bridge networks. Podman 5 switched to

`passt` for rootless networking and 5.3 includes fixes like host-service reachability. (Earlier versions appear to cause "host unreachable" or "network unreachable" issues with bridge networks - which is the type Forgejo Runners create.)

Cache configuration

Some actions such as `https://data.forgejo.org/actions/cache` or `https://data.forgejo.org/actions/setup-go` can communicate with the Forgejo Runner to save and restore commonly used files such as compilation dependencies. They are stored as compressed tar archives, fetched when a job starts and saved when it completes.

If the machine has a fast disk, uploading the cache when the job starts may significantly reduce the bandwidth required to download and rebuild dependencies.

For more information see the `cache` section of the [runner configuration file](#).

Configuration file reference

Installation procedures for Forgejo Runner describe how to generate a default configuration file, eg. [for a binary installation](#) or [for a docker compose installation](#). Those procedures are recommended to ensure the correct configuration file format is provided, with all the available options that match the version of Forgejo Runner that you are installing.

All Forgejo Runner configuration options are documented in the default configuration file in YAML comments.

The most recent version of the configuration file can be viewed [in the forgejo/runner repo](#).

`##` #3
`##` cocobook-sync `##` 2026-10-28 16:41:34 UTC
`##` cocobook-sync `##` 2026-10-29 01:09:34 UTC