# Downloading the Source

The Android source tree is located in a Git repository hosted by Google. The Git repository includes metadata for the Android source, including changes to the source and when the changes were made. This page describes how to download the source tree for a specific Android code line.

To start with a factory image for a specific device instead of downloading the source, see Selecting a device build.

## [ Initializing a Repo client ]

After installing the Repo Launcher, set up your client to access the Android source repository:

1. Create an empty directory to hold your working files. Give it any name you like:

   ```
   mkdir WORKING_DIRECTORY
   cd WORKING_DIRECTORY
   ```

2. Configure Git with your real name and email address. To use the Gerrit code-review tool, you need an email address that's connected with a registered Google account. Ensure that this is a live address where you can receive messages. The name that you provide here shows up in attributions for your code submissions.

   ```
   git config --global user.name Your Name
   git config --global user.email you@example.com
   ```

3. Run `repo init` to get the latest version of Repo with its most recent bug fixes. You must specify a URL for the manifest, which specifies where the various repositories included in the Android source are placed within your working directory.

   ```
   repo init -u https://android.googlesource.com/platform/manifest
   ```

   To check out the master branch:

   ```
   repo init -u https://android.googlesource.com/platform/manifest -b master
   ```

   To check out a branch *other than master*, specify it with `-b`. For a list of branches, see Source code tags and builds.

   ***For Python 2***

   **Warning:** Python 2 support was sunset on Jan 1, 2020 as detailed in Sunsetting Python 2. All major Linux distributions are deprecating support for the Python 2 package. Google strongly recommends that you migrate all your scripts over to Python 3.

   **Note:** AOSP ships with its own copies of the Python 2 and Python 3 packages, and you can use the version (such as SEPolicy) that's included in the source tree. Google is migrating all scripts in the Android source tree to Python 3, and the embedded copy of Python 2 might be deprecated.

   For additional details, see Porting Python 2 Code to Python 3, and advice on sunsetting your Python 2 code.

   ***For Python 3***

   If you get a "`/usr/bin/env 'python' no such file or directory`" error message, use one of the following solutions:

   If your Ubuntu 20.04.2 LTS is a newly installed (vs. upgraded) Linux version:

```
sudo ln -s /usr/bin/python3 /usr/bin/python
```

If using Git version 2.19 or greater, you can specify `--partial-clone` when performing `repo init`. This makes use of Git's partial clone capability to only download Git objects when needed, instead of downloading everything. Because using partial clones means that many operations must communicate with the server, use the following if you're a developer and you're using a network with low latency:

```
repo init -u https://android.googlesource.com/platform/manifest -b master --
partial-clone --clone-filter=blob:limit=10M
```

For Windows OS only: if you get an error message stating that symbolic links couldn't be created, causing `repo init` to fail, reference the GitHub Symbolic Links documentation to create these, or to enable their support. For non-administrators, see the Allowing non-administrators to create symbolic links section.

A successful initialization ends with a message stating that Repo is initialized in your working directory. Your client directory now contains a `.repo` directory where files such as the manifest are kept.

## [ Downloading the Android source tree ]

To download the Android source tree to your working directory from the repositories as specified in the default manifest, run:

```
repo sync
```

To speed syncs, pass the `-c` (current branch) and `-jthreadcount` flags:

```
repo sync -c -j8
```

The Android source files are downloaded in your working directory under their project names.

To suppress output, pass the `-q` (quiet) flag. See the Repo Command Reference for all options.

## [ Using authentication ]

By default, access to the Android source code is anonymous. To protect the servers against excessive use, each IP address is associated with a quota.

When sharing an IP address with other users (for example, when accessing the source repositories from beyond a NAT firewall), the quotas can trigger even for regular use patterns (for example, if many users sync new clients from the same IP address within a short period).

In that case, you can use authenticated access, which then uses a separate quota for each user, regardless of the IP address.

First, create a password with the password generator and follow the instructions on the password generator page.

Next, force authenticated access by using the manifest URI `https://android.googlesource.com/a/platform/manifest`. Notice how the `/a/` directory prefix triggers mandatory authentication. You can convert an existing client to use mandatory authentication with the following command:

```
repo init -u https://android.googlesource.com/a/platform/manifest
```

## [ Troubleshooting network issues ]

When downloading from behind a proxy (which is common in some corporate environments), you might need to to explicitly specify the proxy for Repo to use:

```
export HTTP_PROXY=http://<proxy_user_id>:<proxy_password>@<proxy_server>:<proxy_port>
export HTTPS_PROXY=http://<proxy_user_id>:<proxy_password>@<proxy_server>:<proxy_port>
```

More rarely, Linux clients experience connectivity issues, getting stuck in the middle of downloads (typically during *receiving objects*). Adjusting the settings of the TCP/IP stack and using non-parallel commands can improve the situation. You must have root access to modify the TCP setting:

```
sudo sysctl -w net.ipv4.tcp_window_scaling=0
repo sync -j1
```

## [ Using a local mirror ]

When using several clients, especially in situations where bandwidth is scarce, it's better to create a local mirror of the entire server content, and to sync clients from that mirror (which requires no network access). The download for a full mirror is smaller than the download of two clients, and it contains more information.

These instructions assume that the mirror is created in `/usr/local/aosp/mirror`. First, create and sync the mirror itself. Notice the `--mirror` flag, which you can specify only when creating a new client:

```
mkdir -p /usr/local/aosp/mirror
cd /usr/local/aosp/mirror
repo init -u https://android.googlesource.com/mirror/manifest --mirror
repo sync
```

When the mirror is synced, you can create new clients from it. Note that you must specify an absolute path:

```
mkdir -p /usr/local/aosp/master
cd /usr/local/aosp/master
repo init -u /usr/local/aosp/mirror/platform/manifest.git
repo sync
```

Finally, to sync a client against the server, sync the mirror against the server, then the client against the mirror:

```
cd /usr/local/aosp/mirror
repo sync
cd /usr/local/aosp/master
repo sync
```

It's possible to store the mirror on a LAN server and to access it over NFS, SSH, or Git. It's also possible to store it on a removable drive and to pass that drive among users or machines.

## [ Verifying Git tags ]

Load the following public key into your GnuPG key database. The key is used to sign annotated tags that represent releases.

```
gpg --import
```

Copy and paste the key below, then type `EOF` (**Ctrl-D**) to end the input and process the keys.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.2.2 (GNU/Linux)

mQGiBEnnWD4RBACt9/h4v9xnnGDou13y3dvOx6/t43LPPIxeJ8eX9WB+8LLuROSV
lFhpHawsVAcFlmi7f7jdSRF+OvtZL9ShPKdLfwBJMNkU66/TZmPewS4m782ndtw7
8tR1cXb197Ob8kOfQB3A9yk2XZ4ei4ZC3i6wVdqHLRxABdncwu5hOF9KXwCgkxMD
u4PVgChaAJzTYJ1EG+UYBIUEAJmfearb0qRAN7dEoff0FeXsEaUA6U90sEoVks0Z
wNj96SA8BL+a1OoEUUfpMhiHyLuQSftxisJxTh+2QclzDviDyaTrkANjdYY7p2cq
/HMdOY7LJlHaqtXmZxXjjtw5Uc2QG8UY8aziU3IE9nTjSwCXeJnuyvoizl9/I1S5
jU5SA/9WwIps4SC84ielIXiGWEqq6i6/sk4I9q1YemZF2XVVKnmI1F4iCMtNKsR4
MGSa1gA8s4iQbsKNWPgp7M3a51JCVCu6l/8zTpA+uUGapw4tWCp4o0dpIvDPBEa9
b/aF/ygcR8mh5hgUfpF9IpXdknOsbKCvM9lSSfRciETykZc4wrRCVGhlIEFuZHJv
aWQgT3BlbiBTb3VyY2UgUHJvamVjdCA8aW5pdGlhbC1jb250cmlidXRpb25AYW5k
cm9pZC5jb20+iGAEExECACAFAknnWD4CGwMGCwkIBwMCBBUCCAMEFgIDAQIeAQIX
gAAKCRDorT+BmrEOeNr+AJ42Xy6tEW7r3KzrJxnRX8mij9z8tgCdFfQYiHpYngkI
2t09Ed+9Bm4gmEO5Ag0ESedYRBAIAKVW1JcMBWvV/0Bo9WiByJ9WJ5swMN36/vAl
QN4mWRhfzDOk/Rosdb0csAO/l8Kz0gKQPOfObtyYjvI8JMC3rmi+LIvSUT9806Up
hisyEmmHv6U8gUb/xHLIanXGxwhYzjgeuAXVCsv+EvoPIHbY4L/KvP5x+oCJIDbk
C2b1TvVk9PryzmE4BPIQL/NtgR1oLWm/uWR9zRUFtBnE411aMAN3qnAHBBMZzKMX
LWBGWE0znfRrnczI5p49i2YZJAjyX1P2WzmScK49CV82dzLo71MnrF6fj+Udtb5+
OgTg7Cow+8PRaTkJEW5Y2JIZpnRUq0CYxAmHYX79EMKHDSThf/8AAwUIAJPWsB/M
pK+KMs/s3r6nJrnYLTfdZhtmQXimpoDMJg1zxmL8UfNUKiQZ6esoAWtDgpqt7Y7s
KZ8laHRARonte394hidZzM5nb6hQvpPjt2OlPRsyqVxw4c/KsjADtAuKW9/d8phb
N8bTyOJo856qg4oOEzKG9eeF7oaZTYBy33BTL0408sEBxiMior6b8LrZrAhkqDjA
vUXRwm/fFKgpsOysxC6xi553CxBUCH2omNV6Ka1LNMwzSp9ILz8jEGqmUtkBszwo
G1S8fXgE0Lq3cdDM/GJ4QXP/p6LiwNF99faDMTV3+2SAOGvytOX6KjKVzKOSsfJQ
hN0DlsIw8hqJc0WISQQYEQIACQUCSedYRAIbDAAKCRDorT+BmrEOeCUOAJ9qmR0l
EXzeoxcdoafxqf6gZlJZlACgkWF7wi2YLW3Oa+jv2QSTlrx4KLM=
=Wi5D
-----END PGP PUBLIC KEY BLOCK-----
```

After importing the keys, you can verify any tag with:

```
git tag -v TAG_NAME
```

## [Obtain proprietary binaries]

AOSP can't be used from pure source code only and requires additional hardware-related proprietary libraries to run, such as for hardware graphics acceleration. See the sections below for download links and Device binaries for additional resources.

Some devices package these proprietary binaries on their `/vendor` partition.

### [Download proprietary binaries]

You can download official binaries for the supported devices running tagged AOSP release branches from Google's drivers. These binaries add access to additional hardware capabilities with non-open source code. To build the AOSP master branch, use the Binaries Preview instead. When building the master branch for a device, use the binaries for the most recent numbered release or with the most recent date.

### [Extract proprietary binaries]

Each set of binaries comes as a self-extracting script in a compressed archive. Uncompress each archive, run the included self-extracting script from the root of the source tree, then confirm you agree to the terms of the enclosed license agreement. The binaries and their matching makefiles are installed in the `vendor/` hierarchy of the source tree.

### [Clean up]

To ensure the newly installed binaries are properly taken into account after being extracted, delete the existing output of any previous build using:

```
make clobber
```