Installing Repo

Repo is a tool that makes it easier to work with Git in the context of Android. For more information about Repo, see the [Developing](http://source.android.com/source/developing.html) section.

To install Repo:

1. Make sure you have a bin/ directory in your home directory and that it is included in your path:

$ mkdir ~/bin  
$ PATH=~/bin:$PATH

1. Download the Repo tool and ensure that it is executable:

$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo  
$ chmod a+x ~/bin/repo

For version 1.21, the SHA-1 checksum for repo is b8bd1804f432ecf1bab730949c82b93b0fc5fede

For version 1.22, the SHA-1 checksum for repo is da0514e484f74648a890c0467d61ca415379f791

For version 1.23, the SHA-1 checksum for repo is ac9d646f6d699f6822a6bc787d3e7338ae7ab6ed

Initializing a Repo client

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

1. Create an empty directory to hold your working files. If you're using MacOS, this has to be on a case-sensitive filesystem. Give it any name you like:

$ mkdir WORKING\_DIRECTORY  
$ cd WORKING\_DIRECTORY

1. Configure git with your real name and email address. To use the Gerrit code-review tool, you will need an email address that is connected with a [registered Google account](https://www.google.com/accounts). Make sure this is a live address at which you can receive messages. The name that you provide here will show up in attributions for your code submissions.

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

1. Run repo init to bring down the latest version of Repo with all its most recent bug fixes. You must specify a URL for the manifest, which specifies where the various repositories included in the Android source will be placed within your working directory.

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

To check out a branch other than "master", specify it with -b. For a list of branches, see [Source Code Tags and Builds](http://source.android.com/source/build-numbers.html#source-code-tags-and-builds).

$ repo init -u https://android.googlesource.com/platform/manifest -b android-4.0.1\_r1

A successful initialization will end with a message stating that Repo is initialized in your working directory. Your client directory should now contain a .repo directory where files such as the manifest will be kept.

Downloading the Android Source Tree

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

$ repo sync

The Android source files will be located in your working directory under their project names. The initial sync operation will take an hour or more to complete. For more about repo sync and other Repo commands, see the [Developing](http://source.android.com/source/developing.html) section.

Using Authentication

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

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

In that case, it is possible to use authenticated access, which then uses a separate quota for each user, regardless of the IP address.

The first step is to create a password with [the password generator](https://android.googlesource.com/new-password) and follow the instructions on the password generator page.

The second step is to force authenticated access, by using the following 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), it might be necessary to explicitly specify the proxy that is then used by repo:

$ 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"). It has been reported that tweaking the settings of the TCP/IP stack and using non-parallel commands can improve the situation. You need 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 is 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, while containing more information.

These instructions assume that the mirror is created in /usr/local/aosp/mirror. The first step is to create and sync the mirror itself. Notice the --mirror flag, which can be specified 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

Once the mirror is synced, new clients can be created from it. Note that it's important to 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, the mirror needs to be synced 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 around between users or between 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(s) below, then enter 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

If you haven't [set up ccache](http://source.android.com/source/initializing.html#ccache) yet, now would be a good time to do it.