Web Real-Time Communications Deployment Guide

TURN Server Deployment

12/22/2025

Contents

TURN Server Deployment

The Genesys WebRTC Service has been tested with the coTURN TURN server, which is a free, high-performance open-source TURN and STUN server implementation. Genesys currently recommends v4.5.0.3 of the coTURN TURN server; however, more recent versions may exist.

If a TURN server is already deployed in your network or you are planning to deploy a TURN server other than coTURN, you can skip the following steps.

Host Requirements

- RHEL/CentOS 6.6+
- At least two vCPUs, 4GB Memory. SSD may be used, but is not mandatory.
- The most important factor is the networking performance. Make sure that your network has:
  - High packet per second (PPS) performance
  - Low network jitter (<= 30ms)
  - Low latencies (<= 150ms)

For example, if you are deploying the TURN server in an AWS instance, then Enhanced Networking is only available on instances launched with HVM AMIs. Preferably, use C3/C4/R3 instances. For details on AWS EC2 instance types, check here.

Installation and Configuration

Download and install the latest version of TURN server from this link.

You must install the TURN server on a separate server. If you want to install the TURN server in the same machine as the WebRTC Gateway, make sure there is no port conflict.

You must have super-user access to install the TURN server.

1. Download the TURN server installation package from the above link, and unzip the file into a temporary folder.
2. Run the **install.sh** file to install the TURN server:

   ```
   # chmod +x install.sh
   # ./install.sh
   ```

3. The installation creates a `turnserver` user, and the TURN server runs from this account. To enable the TURN server to bind lower ports (<1024), set the following:

   ```
   # setcap 'cap_net_bind_service=+ep' /usr/bin/turnserver
   ```

4. Update the parameters in the following configuration file:

   `/etc/turnserver/turnserver.conf`

   - **listening-port**: set to 443
   - **external-ip**: set to the `<public-ip>[/private-ip]` for this VM; for example `54.145.183.203/10.157.156.67`. The assumption is that the TURN server will be behind NAT in most of the deployments.
   - Uncomment **fingerprint**
   - Uncomment **lt-cred-mech**
   - **user**: set to `<user>:<password>`; for example `genesys:genesys`. The same user/password you need to configure for your web application.
   - **realm**: simply set it to `yourdomain.com` (for example `ec2-54-145-183-203.compute-1.amazonaws.com`)

5. Start the process:

   ```
   # service turnserver start
   ```

6. Check if the TURN server is running:

   ```
            # ps -elf | grep turnserver
   5 S 498      24882     1  0  80   0 - 138025 ep_pol 23:44 ?       00:00:00 /usr/bin/
   turnserver -o -c /etc/turnserver/turnserver.conf
   ```

7. For additional details on the TURN server parameters and configuration, you can check here.

Verifying your TURN Server

To verify whether your TURN server is running properly and is accessible, go to this demo page.

This page tests the trickle ICE functionality in a WebRTC implementation. It creates a `PeerConnection` with the specified STUN/TURN servers, and then starts candidate gathering for a session with a single audio stream. As candidates are gathered, they are displayed in a text box, along with an indication when candidate gathering is complete.

Select and remove the default STUN server that is populated in this page, and then add your TURN server(s) instead. For example, if you have installed the Genesys TURN server in `turn.example.net` and set password as `genesys` for `genesys` user, add both:

- `turn:turn.example.net:443` (and user/password as `genesys/genesys`)
- `turn:turn.example.net:443?transport=tcp` (and user/password as `genesys/genesys`)

Then press `Gather candidates` and specifically check for `relay` candidates in the result (see example below). If no relay candidates are generated, either your TURN server is not running properly or is not accessible from your network.

Time	Component	Type	Foundation	Protocol	Address	Port	Priority
0.001	1	host	2999745851	udp	192.168.56.1	50584	126 | 32542 | 255
0.002	1	host	2178013618	udp	192.168.33.1	50585	126 | 32286 | 255
0.002	1	host	4135718160	udp	192.168.250.241	50586	126 | 32030 | 255
0.002	2	host	2999745851	udp	192.168.56.1	50587	126 | 32542 | 254
0.002	2	host	2178013618	udp	192.168.33.1	50588	126 | 32286 | 254
0.002	2	host	4135718160	udp	192.168.250.241	50589	126 | 32030 | 254
0.101	1	host	4233069003	tcp	192.168.56.1	0	90 | 32542 | 255
0.101	1	host	3478267202	tcp	192.168.33.1	0	90 | 32286 | 255
0.104	1	host	3087135200	tcp	192.168.250.241	0	90 | 32030 | 255
0.104	2	host	4233069003	tcp	192.168.56.1	0	90 | 32542 | 254
0.104	2	host	3478267202	tcp	192.168.33.1	0	90 | 32286 | 254
0.104	2	host	3087135200	tcp	192.168.250.241	0	90 | 32030 | 254
0.228	1	srflx	2672313041	udp	64.47.225.126	50586	100 | 32030 | 255
0.232	2	srflx	2672313041	udp	64.47.225.126	50589	100 | 32030 | 254
0.252	1	relay	170195997	udp	54.167.137.125	51628	2 | 32030 | 255
0.254	2	relay	170195997	udp	54.167.137.125	62964	2 | 32030 | 254
0.302	2	relay	170195997	udp	54.167.137.125	57811	1 | 32030 | 254
0.304	1	relay	170195997	udp	54.167.137.125	55868	1 | 32030 | 255
9.687	Done						

Also, the type of candidates released to the application can be controlled via the `IceTransports` constraint.