Jitsi Meet on Arch Linux

Jitsi Meet is a secure videoconference service that allows you to host your conference on your server. You can invite users. They don’t need to subscribe to a service to join the conversation.

Presentation

Jitsi Meet is a secure videoconference service that allows you to host your conference on your server. You can invite users. They don't need to subscribe to a service to join the conversation.

Objective

The objective is to set up the packages to make jitsi-meet works on your standalone server:

- jitsi-meet-bin
  the web interface
- jitsi-meet-prosody-bin
  the config of jitsi-meet for prosody
- jitsi-meet-turnserver-bin
  example for setting coturn
- jitsi-videobridge-bin
  the bridge to host 3 or more users
- jicofo-bin
  the bridge between prosody and videobridge
- coturn
  the proxy for restricted networks
- prosody
  the authenticate and chat system
- nginx
  the proxy to serve the web interface

Installation

Sudo

To use ‘yay’ you need to have a user in a group that can run **sudo**. With the root user:

```
adduser -m myuser
usermod -G wheel myuser
pacman -S sudo vim
EDITOR=vim visudo
# lookup %wheel and uncomment by removing the #
# %wheel ALL=(ALL) NOPASSWD: ALL
```

The rest of the documentation will assume that you are using your user (here *myuser*) to do the installation.

Documentations

The installation packages above contains examples that are extract in “/usr/share/doc”. Comment the line in the configuration that disables the extract of the documentation in your “pacman.conf”:

/etc/pacman.conf:

```
#NoExtract = usr/share/gtk-doc/html/* usr/share/doc/*
```

Development tools

Let’s install the development tools and **yay**:

```
sudo pacman -S base-devel pacman-contrib git
git clone https://aur.archlinux.org/yay.git
cd yay
makepkg -si
cd
yay
```

Packages

To install the stable release:

```
yay -S --needed --noconfirm \
  nginx coturn prosody lua52 \
  lua52-sec lua52-zlib lua52-event
yay -S \
  jitsi-meet-bin jitsi-meet-prosody-bin \
  jitsi-meet-turnserver-bin jitsi-meet-prosody-bin \
  jicofo-bin jitsi-videobridge-bin
```

Tools

Those tools will help to generate passwords or random uuid for the configurations under.

Generate a random password

There are several methods to generated a random password. This is one of them:

```
openssl rand -hex 24
```

You need a password for:

- The user “focus” on the subdomain auth.YOUR_DOMAIN __PASSWORD_FOR_USER_focus__
- The user “jvb” on the subdomain auth.YOUR_DOMAIN __PASSWORD_FOR_USER_jvb__

Generate a random UUID

To generate MUC random UUID (see under), you can use the uuidgen tool provided by archlinux.

```
uuidgen
```

Configuration

Let’s configure jitsi-meet. The first step is to set up jitsi-meet and allow starting conference on your server like the website meet.ji.si

If you use my package, all configurations files are secure to the user that runs the service. So to edit them easily I suggest using the user "root".

Organization of the files

The package’s files are organized like this:

- jicofo-bin
  - /etc/jicofo (configurations)
  - /usr/lib/jicofo (binaries)
- jitsi-videobridge-bin
  - /etc/jitsi-videobridge (configurations)
  - /usr/lib/jitsi-videobridge (binaries)
- jitsi-meet-bin
  - /etc/webapps/jitsi-meet (configurations)
  - /usr/share/webapps/jitsi-meet (html files)
  - /usr/share/doc/jitsi-meet (examples of configuration for nginx or apache)
- prosody
  - /etc/prosody (general configurations)
  - /etc/prosody/conf.d (configurations)
  - /usr/lib/jitsi-meet-prosody (lua plugins)
  - /usr/share/doc/jitsi-meet-prosody (example of configuration)
- coturn
  - /etc/turnserver (configurations)
  - /usr/share/doc/jitsi-meet-turnserver (example of configuration)
- nginx
  - /etc/nginx

Loopback

Let’s set up a local loopback for your jitsi-meet domain to allow each project to communicate locally together.

/etc/hosts:

```
127.0.0.1 YOUR_DOMAIN auth.YOUR_DOMAIN
::1 YOUR_DOMAIN auth.YOUR_DOMAIN
```

Jicofo

/etc/jicofo/jicofo.conf

```
jicofo {
  xmpp: {
    client: {
      client-proxy: "focus.YOUR_DOMAIN"
      xmpp-domain: "YOUR_DOMAIN"
      domain: "auth.YOUR_DOMAIN"
      username: "focus"
      password: "FOCUS_PASSWORD"
      conference-muc-jid = conference.YOUR_DOMAIN
    }
    trusted-domains: [ "recorder.YOUR_DOMAIN" ]
  }
  bridge: {
    brewery-jid: "JvbBrewery@internal.auth.YOUR_DOMAIN"
  }
}
```

Jitsi videobridge

/etc/jitsi-videobridge/sip-communicator.properties

```
org.jitsi.videobridge.xmpp.user.shard.DOMAIN=auth.YOUR_DOMAIN
org.jitsi.videobridge.xmpp.user.shard.PASSWORD=__PASSWORD_FOR_USER_jvb__
org.jitsi.videobridge.xmpp.user.shard.MUC_JIDS=JvbBrewery@internal.auth.YOUR_DOMAIN
#use uuidgen
org.jitsi.videobridge.xmpp.user.shard.MUC_NICKNAME=1a988801-1476-4417-9bcc-03f648be86c7
```

Jitsi meet

/etc/webapps/jitsi-meet/config.js

```
var config = {
  hosts: {
    domain: 'YOUR_DOMAIN',
    // ...
    muc: 'conference.YOUR_DOMAIN'
  },
  bosh: '//YOUR_DOMAIN/http-bind',
  // ...
}
```

Prosody

"jitsi-meet-prosody" provide an example of configuration at "/usr/share/doc/jitsi-meet-prosody/prosody.cfg.lua-jvb.example".

Let’s copy the example to the prosody directory:

```
cd /etc/prosody
mkdir conf.d
cp /usr/share/doc/jitsi-meet-prosody/prosody.cfg.lua-jvb.example conf.d/jitsi.cfg.lua
```

Include this file to the main prosody configuration:

/etc/prosody/prosody.cfg.lua

```
-- at the end of the file
Include "conf.d/*.cfg.lua"
```

/etc/prosody/conf.d/jitsi.cfg.lua

Replace all:

- "jitmeet.example.com" with "YOUR_DOMAIN"
- "focusUser@auth.YOUR_DOMAIN" with "focus@auth.YOUR_DOMAIN"

Then adjust the configuration at your needs.

```
plugin_paths = { "/usr/lib/jitsi-meet-prosody/" }
```

SSL

We need to generate SSL for prosody and nginx for YOUR_DOMAIN and auth.YOUR_DOMAIN.

Self-signed

Generate your certificate

```
sudo -u prosody prosodyctl cert generate YOUR_DOMAIN
sudo -u prosody prosodyctl cert generate auth.YOUR_DOMAIN
mv /var/lib/prosody/*.{crt,cnf,key} /etc/prosody/certs/
trust anchor /etc/prosody/certs/YOUR_DOMAIN.crt
trust anchor /etc/prosody/certs/auth.YOUR_DOMAIN.crt
update-ca-trust # for java
```

Letsencrypt

If you use letsencrypt, you can import your certificate automatically to prosody.
Prosody warns you about "No certificate for ...". Don’t worry, these virtual hosts are internal. The only ones you needs are YOUR_DOMAIN and auth.YOUR_DOMAIN.

/etc/letsencrypt/renewal-hooks/deploy/prosody.sh

```
#!/bin/sh
/usr/bin/prosodyctl --root cert import /etc/letsencrypt/live
```

The set execution flag to the file and run it once

```
chmod +x /etc/letsencrypt/renewal-hooks/deploy/prosody.sh
/etc/letsencrypt/renewal-hooks/deploy/prosody.sh
```

Each time letsencrypt renew his certificates, it will automatically import the certificate to prosody.

Prosody

The prosody example should point to the directory "/etc/prosody/certs". Readjust the path if needed. Also, add the certificate for the "auth" virtual host.

/etc/prosody/conf.d/jitsi.cfg.lua

```
VirtualHost "YOUR_DOMAIN"
ssl = {
  key = "/etc/prosody/certs/YOUR_DOMAIN.key";
  certificate = "/etc/prosody/certs/YOUR_DOMAIN.crt";
```

```
}


VirtualHost "auth.YOUR_DOMAIN"
ssl = {
  key = "/etc/prosody/certs/auth.YOUR_DOMAIN.key";
  certificate = "/etc/prosody/certs/auth.YOUR_DOMAIN.crt";
}
authentication = "internal_hashed"
```

Nginx

Nginx is a proxy that allows us to deliver jitsi-meet webpages to the users' web browser.

Let's copy the provided example.

```
cd /etc/nginx
mkdir sites
cp /usr/share/doc/jitsi-meet/jitsi-meet.example sites/jitsi.conf
```

/etc/nginx/nginx.conf

```
http {
  // ...
  // this should be placed near to the close bracket of the http block
  include sites/*.conf;
}
```

/etc/nginx/sites/jitsi.conf

```
server {
  # ...
  server_name YOUR_DOMAIN;



  # ...
  # use prosody path directly
  ssl_certificate /etc/prosody/certs/YOUR_DOMAIN.crt;
  ssl_certificate_key /etc/prosody/certs/YOUR_DOMAIN.key;
  # or use letencrypt path
```

```
  ssl_certificate /etc/letsencrypt/live/YOUR_DOMAIN/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/YOUR_DOMAIN/privkey.pem;



  # set the config path
  # replace alias /etc/jitsi/meet/jitmeet.example.com-config.js by
  location = /config.js {
    alias /etc/webapps/jitsi-meet/config.js;
  }
  # ...
  location ~ ^/([^/?&:'"]+)/config.js$
  {
    set $subdomain "$1.";
    set $subdir "$1/";
    alias /etc/webapps/jitsi-meet/config.js
  }
}
```

Check your configuration.

```
nginx -t
```

Register users

We register user “jvb” and “focus” to prosody.

```
prosodyctl register jvb auth.YOUR_DOMAIN __PASSWORD_FOR_USER_jvb__
prosodyctl register focus auth.YOUR_DOMAIN __PASSWORD_FOR_USER_focus__
prosodyctl mod_roster_command subscribe focus.YOUR_DOMAIN focus@auth.YOUR_DOMAIN
```

Start the services

We are now ready to start jitsi-meet

```
systemctl start prosody.service
systemctl start jitsi-videobridge.service
systemctl start jicofo.service
systemctl start nginx.service
```

Checks that everything looks fine

```
systemctl status \
  prosody.service \
  jitsi-videobridge.service \
  jicofo.service nginx.service
```

Activate the service to start at boot

```
systemctl enable prosody.service
systemctl enable jitsi-videobridge.service
systemctl enable jicofo.service
systemctl enable nginx.service
```

Activate WebSocket

XMPP

The default configuration uses long pooling requests to fetch changes of statuses and messages between the participants.

You can switch to WebSocket. It uses fewer resources. It is more reactive. It needs a bit more configuration.

Jitsi meet

/etc/webapps/jitsi-meet/config.js

```
var config = {
  websocket: 'wss://YOUR_DOMAIN/xmpp-websocket',
}
```

Prosody

/etc/prosody/conf.d/jitsi.cfg.lua

```
-- above virtualhost
consider_websocket_secure = true;
cross_domain_websocket = true;
-- in your virtual host
VirtualHost "YOUR_DOMAIN"
    modules_enabled = {
        -- ...
        "websocket";
    }
```

Reload prosody

```
systemctl restart prosody
```

Bridge

WebSockets can be used instead of WebRTC Data Channels for the transport of Colibri client-to-bridge messages. This needs support from the bridge as well as the client.

When this is enabled, a bridge will advertise a Colibri WebSocket URL together with its ICE candidates. The URL will be specific to an endpoint (in fact the ICE username fragment is reused, encoded as a URL parameter, for authentication), and connections to it will be routed to the Endpoint representation in the bridge.

Jitsi videobridge

/etc/jitsi-videobridge/jvb.conf

```
videobridge {
  http-servers {
    public {
      port = 9090
    }
  }
  websockets {
    enabled=true
    server-id="default-id"
    tls=true
    domain="YOUR_DOMAIN:443"
  }
}
```

Reload jitsi-videobridge

```
systemctl restart jitsi-videobridge
```

Restrict your configuration

The objective is to limit the creation of a conference room to a list of identified users. Guests will have to wait for one of these users to come and unlock the room.

Jicofo

/etc/jicofo/jicofo.conf

```
jicofo {
  authentication {
    enabled = true
    type = XMPP
    login-url = YOUR_DOMAIN
    enable-auto-login = true
  }
}
```

The restart.

```
systemctl restart jicofo
```

Jitsi meet

/etc/webapps/jitsi-meet/config.js

```
var config = {
  host: {
    // anonymous users need to use a dedicated muc without authentication
    anonymousdomain: 'guest.YOUR_DOMAIN',
  },
}
```

Prosody

/etc/prosody/conf.d/jitsi.cfg.lua

```
-- change authentification of your domain
VirtualHost "YOUR_DOMAIN"
    authentification = "internal_plain"
-- add guest virtual host to allow anonymous user to join your room
VirtualHost "guest.YOUR_DOMAIN"
        authentication = "jitsi-anonymous"
        c2s_require_encryption = false
        modules_enabled = {
          -- copy the content of the modules_enabled
          -- of the VirtualHost "YOUR_DOMAIN"
          -- remove only the module "muc_lobby_rooms" of the list
          -- example:
          "bosh";
          "pubsub";
          "ping"; -- Enable mod_ping
          "speakerstats";
          "external_services";
          "conference_duration";
          "websocket";
        }
```

Then restart

```
systemctl restart prosody
```

Register an user

Let’s add a registered user to your domain. Only registered user can now create a room.

```
prosodyctl register USER YOUR_DOMAIN PASSWORD
```

Restrict your configuration with a JWT Token

The objective is to limit the creation of a conference room to any user authenticate with a JWT token. Guests will have to wait for one of these users to come and unlock the room.

You can find the spec here: https://github.com/jitsi/lib-jitsi-meet/blob/master/doc/tokens.md

Packages

You need those package to user the tokens authentication:

```
yay -S --needed lua52-base64 lua52-basexx lua52-cjson lua52-jwtjitsi lua52-luaossl
```

Jicofo

/etc/jicofo/jicofo.conf

```
jicofo {
  authentication {
    enabled = true
    type = JWT
    login-url = YOUR_DOMAIN
    enable-auto-login = true
  }
}
```

The restart.

```
systemctl restart jicofo
```

Jitsi meet

/etc/webapps/jitsi-meet/config.js

```
var config = {
  host: {
    // anonymous users need to use a dedicated muc without authentication
    anonymousdomain: 'guest.YOUR_DOMAIN',
  },
}
```

Prosody

/etc/prosody/conf.d/jitsi.cfg.lua

```
-- change authentification of your domain
VirtualHost "YOUR_DOMAIN"
    authentification = "token"
    app_id = "APP_ID"
    app_secret = "APP_SECRET"
    allow_empty_token = false
    modules_enabled = {
        -- keep existing modules and add
        "presence_identity";
    }
    c2s_require_encryption = false
-- add guest virtual host to allow anonymous user to join your room
VirtualHost "guest.YOUR_DOMAIN"
        authentication = "jitsi-anonymous"
        c2s_require_encryption = false
        modules_enabled = {
          -- copy the content of the modules_enabled
          -- of the VirtualHost "YOUR_DOMAIN"
          -- remove only the module "muc_lobby_rooms" of the list
          -- add presence_identity
          -- example:
          "bosh";
          "pubsub";
          "ping"; -- Enable mod_ping
          "speakerstats";
          "external_services";
```

```
        "conference_duration";
        "websocket";
        "presence_identity";
      }
Component "conference.YOUR_DOMAIN" "muc"
    modules_enabled = {
      -- add this to the modules_enabled
      "token_verification";
    }
```

Then restart

```
systemctl restart prosody
```

Generate a JWT token

Here a quick example in nodejs that will generate a valid link with a JWT token:

```
const jwt = require('jsonwebtoken')
const crypto = require('crypto');
const words = require('random-words')
const yourDomain = "YOUR_DOMAIN"
const appId = "APP_ID"
const appSecret = "APP_SECRET"
const userName = "YOUR_USERNAME"
const userEmail = "YOUR_EMAIL"
function getBody(domain, appId, name, email, room) {
    const md5Email = crypto.createHash('md5').update(email).digest("hex");
    const id = crypto.createHash('sha1').update(`${name}:${email}`).digest("hex")
    return {
        context: {
            user: {
                avatar: `https:/gravatar.com/avatar/${md5Email}`,
                name,
                email,
                id,
            },
            group: 'users'
        },
```

```
        "aud": "jitsi",
        "iss": appId,
        "sub": domain,
        room,
    }
}
const room = process.argv[2] || words({exactly: 3, join: '-'})
const data = getBody(
    yourDomain,
    appId,
    userName,
    userEmail,
    room,
)
const options = {
    algorithm: 'HS256',
    expiresIn: '2h',
}
const jwtToken = jwt.sign(data, appSecret, options)
console.log(`https://${yourDomain}/${room}?jwt=${jwtToken}`)
```

Turnserver (stun/turn)

When you do a one to one communication with a peer, you have 3 possibilities:

- Direct P2P
- P2P over a turn server
- Jitsi VideoBridge

If you and your peer can’t connect together then a proxy is necessary. In that case, the best way to communicate is to go through a turn server. It uses few resources and is far more efficient than going through the jitsi VideoBridge service.

To advertise your system that a turn server exists, you need to call a stun server. It returns the list of all p2p possibilities, including the turn server.

When you do a peer to peer videoconference, and each peer can’t connect directly to each other, you may use a turnserver to act as a proxy between the peers.

Configuration

```
cp /usr/share/doc/jitsi-meet-turnserver/turnserver.conf /etc/turnserver/turnserver.conf
```

/etc/turnserver/turnserver.conf

```
# Do not remove any line
# Replace this line
static-auth-secret=turnSecretPassword
server-name=YOUR_DOMAIN
realm=YOUR_DOMAIN
# Change the path to fix one (we will use it below)
cert=/etc/turnserver/certs/cert.pem
pkey=/etc/turnserver/certs/pkey.pem
# Add this line at the end (it force turnserver to only listen on this ips)
listening-ip=YOUR_PUBLIC_IPV4
listening-ip=YOUR__PUBLIC_IPV6
relay-ip=YOUR_IPV4
relay-ip=YOUR_IPV6
# If you are behind nat, don't use the relay-ip and use listening-ip that way
```

```
listening-ip=YOUR_PUBLIC_IPV4/YOUR_PRIVATE_IPV4
listening-ip=YOUR__PUBLIC_IPV6/YOUR_PRIVATE_IPV6
```

Certificates

You need to copy the certificates for turnserver. The process uses the user “turnserver” to run and need to be able to read the certificates.

Self-signed

```
mkdir /etc/turnserver/certs
cp /etc/prosody/certs/YOUR_DOMAIN.crt /etc/turnserver/certs/cert.pem
cp /etc/prosody/certs/YOUR_DOMAIN.key /etc/turnserver/certs/pkey.pem
chown -R turnserver: /etc/turnserver/certs
```

Letsencrypt

/etc/letsencrypt/renewal-hooks/deploy/turn.sh

```
#!/bin/sh
### turnserver
/usr/bin/install -d -m770 -o turnserver -g turnserver /etc/turnserver/certs
/usr/bin/install    -m640 -o turnserver -g turnserver /etc/letsencrypt/live/YOUR_DOMAIN/fullchain.pem
/etc/turnserver/certs/cert.pem
/usr/bin/install    -m600 -o turnserver -g turnserver /etc/letsencrypt/live/YOUR_DOMAIN/privkey.pem
/etc/turnserver/certs/pkey.pem
/bin/systemctl restart turnserver.service
```

The set execution flag to the file and run it once

```
chmod +x /etc/letsencrypt/renewal-hooks/deploy/turn.sh
/etc/letsencrypt/renewal-hooks/deploy/turn.sh
```

Jitsi meet

/etc/webapps/jitsi-meet/config.js

```
{
   p2p: {
      enabled: true,
```

```
        stunServers: [
            { urls: 'stun:YOUR_DOMAIN:3478' }
        ],
        // activate this option to force the usage of the relay
        // it will force the usage of turnserver or jvb and avoid direct connection
        // it is nice for testing, it seems to improve the stability of the connection,
        // remove it if you want to allow direct connection without your server as a relay
        iceTransportPolicy: 'relay',
        // take care to avoid extra coma, or the json will became invalid
        backToP2PDelay: 5
    }
}
```

Prosody

/etc/prosody/conf.d/jitsi.cfg.lua

```
external_service_secret = "turnSecretPassword";
external_services = {
    { type = "stun", host = "YOUR_DOMAIN", port = 3478 },
    { type = "turn", host = "YOUR_DOMAIN", port = 3478, transport = "udp", secret = true, ttl = 86400,
algorithm = "turn" },
    { type = "turns", host = "YOUR_DOMAIN", port = 5349, transport = "tcp", secret = true, ttl = 86400,
algorithm = "turn" }
};
```

Start services

```
# restart prosody
systemctl restart prosody
# start turnserver
systemctl start turnserver
# check turnserver
systemctl status turnserver
# activate at boot
systemctl enable turnserver
```

Conclusion

I hope this tutorial helps you to install your jitsi-meet instance. Let us know in the comments below.

Thanks!