

Configurações Jitsi Docker

Procedimentos de configurações e customizações.

- [Jitsi Meet Load Balancing](#)
- [Load Testing Jitsi Meet](#)

Jitsi Meet Load Balancing

Link: <https://meetrix.io/blog/webrtc/jitsi/jitsi-meet-load-balancing.html>

Setup multi server, load balanced videobriges

One of the amazing features in Jitsi Meet is the inbuilt horizontal scalability. When you want to cater large number of concurrent users, you can spin up multiple video bridges to handle the load. If we setup mulple video bridges and connect them to the same shard, Jicofo, the conference manager selects the least loaded Videobridge for the next new conference.

Run Prosody on all interfaces

By default prosody runs only on local interface (127.0.0.0), to allow the xmpp connections from external servers, we have to run prosody on all interfaces. To do that, add the following line at the beginning of `/etc/prosody/prosody.cfg.lua`

```
component_interface = "0.0.0.0"
```

Allow inbound traffic for port 5222

Open inbound traffic from JVB server on port `5222` (TCP) of Prosody server. But DO NOT open this publicly.

Install JVB on a sepearate server

Install a Jitsi Video Bridge on a different server.

Copy JVB configurations from Jitsi Meet server

Replace `/etc/jitsi/videobridge/config` and `/etc/jitsi/video-bridge/sip-communicator.properties` of JVB server with the same files from the original Jitsi Meet server

Update JVB config file

In `/etc/jitsi/videobridge/config` set the `XMPP_HOST` to the ip address/domain of the prosody server

```
# Jitsi Videobridge settings

# sets the XMPP domain (default: none)
JVB_HOSTNAME=example.com

# sets the hostname of the XMPP server (default: domain if set, localhost otherwise)
JVB_HOST=<XMPP_HOST>

# sets the port of the XMPP server (default: 5275)
JVB_PORT=5347

# sets the shared secret used to authenticate to the XMPP server
JVB_SECRET=<JVB_SECRET>

# extra options to pass to the JVB daemon
JVB_OPTS="--apis=rest,xmpp"

# adds java system props that are passed to jvb (default are for home and logging config file)
JAVA_SYS_PROPS="-Dnet.java.sip.communicator.SC_HOME_DIR_LOCATION=/etc/jitsi -
Dnet.java.sip.communicator.SC_HOME_DIR_NAME=videobridge -
Dnet.java.sip.communicator.SC_LOG_DIR_LOCATION=/var/log/jitsi -
Djava.util.logging.config.file=/etc/jitsi/videobridge/logging.properties"
```

In `/etc/jitsi/video-bridge/sip-communicator.properties` file update the following properties

1. `<XMPP_HOST>`: The ip address of the prosody server. Better if you can use the private IP address if that can be accessed from JVB server.
2. `<JVB_NICKNAME>`: This should be a unique string used by Jicofo to identify each JVB

```
org.ice4j.ice.harvest.DISABLE_AWS_HARVESTER=true
org.ice4j.ice.harvest.STUN_MAPPING_HARVESTER_ADDRESSES=meet-jit-si-turnrelay.jitsi.net:443
org.jitsi.videobridge.ENABLE_STATISTICS=true
```

```
org.jitsi.videobridge.STATISTICS_TRANSPORT=muc
org.jitsi.videobridge.xmpp.user.shard.HOSTNAME=<XMPP_HOST>
org.jitsi.videobridge.xmpp.user.shard.DOMAIN=auth.example.com
org.jitsi.videobridge.xmpp.user.shard.USERNAME=jvb
org.jitsi.videobridge.xmpp.user.shard.PASSWORD=<JVB_SECRET>
org.jitsi.videobridge.xmpp.user.shard.MUC_JIDS=JvbBrewery@internal.auth.example.com
org.jitsi.videobridge.xmpp.user.shard.MUC_NICKNAME=<JVB_NICKNAME>
org.jitsi.videobridge.xmpp.user.shard.DISABLE_CERTIFICATE_VERIFICATION=true
```

You can add the following line at the beginning of `/usr/share/jitsi/jvb/jvb.sh` to generate a unique nickname for the JVB at each startup. This might be useful if you are using an auto-scaling mechanism.

```
sed -i
"s/org.jitsi.videobridge.xmpp.user.shard.MUC_NICKNAME=.*\/org.jitsi.videobridge.xmpp.user.shard.MUC_NICKNAME=$(cat /proc/sys/kernel/random/uuid)/g" /etc/jitsi/videobridge/sip-communicator.properties
```

Looking for commercial support for Jitsi Meet ? Please contact us via hello@meetrix.io

Updated: October 19, 2020

Load Testing Jitsi Meet

Link: <https://meetrix.io/blog/webrtc/jitsi/jitsi-meet-load-testing.html>

Make sure your Jitsi Meet infrastructure is ready for production

By deploying a horizontally scalable Jitsi Meet setup, you can scale your Jitsi Meet conferencing infrastructure to cater thousands of concurrent users. But, before you go live you might want to make sure that your infrastructure is capable of handling the desired number of concurrent users.

Jitsi Meet Torture is a tool that can be used to run tests against a Jitsi Instance and it is capable of load testing Jitsi Meet.

Jitsi Meet Load Testing

Jitsi Meet Load Testing with Torture

Selenium Grid

To simulate hundreds of concurrent users, we need to deploy Jitsi Meet Torture against a selenium grid setup. There would be a `selenium hub` which accepts commands from Jitsi Meet Torture and pass them to `selenium nodes`. You can have hundreds of `selenium nodes` to simulate users.

Jitsi Meet Load Testing

Jitsi Meet Load Testing with Torture

Overwhelmed with managing
Jitsi Infrastructure?

Outsource full-time, high-cost Jitsi infrastructure management and maintenance

[Talk to an expert](#)

Minimal setup with docker-compose

1. Install Docker. You can use following scripts on Ubuntu 18.04

```
sudo apt-get update
```

```
sudo apt-get -y install apt-transport-https ca-certificates curl software-properties-common
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository -y \
```

```
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
```

```
$(lsb_release -cs) \
```

```
stable"
```

```
sudo apt-get -y update
```

```
sudo apt-get -y install docker-ce
```

```
sudo usermod -a -G docker $(whoami)
```

2. Install the latest version of Docker Compose

3. Create a `docker-compose.yml` file with following content.

```
version: "3.3"
services:
  torture:
    image: meetrix/jitsi-meet-torture
  hub:
    image: selenium/hub:3.141.59
  node:
    build: ./node
    image: meetrix/jitsi-meet-torture-selenium-node
  volumes:
    - /dev/shm:/dev/shm
  depends_on:
    - hub
  environment:
    HUB_HOST: hub
```

4. Please note that `node` are configured to run only one chrome instance at a time. Run the setup with `docker-compose up -d --scale node=2` to run two nodes.

5. `docker-compose exec torture /bin/bash`

6. Change `<YOUR_JITSI_MEET_INSTANCE_URL>` to your Jitsi Meet installation (for example `https://meet.example.com`) in following script and run to perform the test.

```
./scripts/malleus.sh --conferences=1 --participants=2 --senders=2 --audio-senders=2 --duration=300 --room-name-prefix=test --hub-url=http://hub:4444/wd/hub --instance-url=<YOUR_JITSI_MEET_INSTANCE_URL>
```

Running the test with large number of nodes

You can scale the number of nodes with `docker-compose up --scale node=<NUMBER>` or you can create a VM using same docker images for `selenium hub` and `selenium-node`. Then you can use an autoscaling mechanism (Such as Autoscaling groups on AWS) to scale the number of nodes.

Looking for commercial support for Jitsi Meet ? Please contact us via hello@meetrix.io

Updated: June 19, 2020