jibri

docker

download stable release from https://github.com/jitsi/docker-jitsi-meet/releases

```
wget
https://github.com/jitsi/docker-jitsi-meet/archive/refs/tags/stable-7439-2.t
ar.gz
tar ztf stable-7439-2.tar.gz
```

```
mkdir -p ~/jitsi/conf/{web,transcripts,prosody/config,prosody/prosody-
plugins-custom,jicofo,jvb,jigasi,jibri}
```

jibri.yml inside ufficial docker-jitsi-meet does not work in swarm service (/dev/snd and xorg problems)

clone a version of jibri with pulse and xdummy support

download

```
git clone https://github.com/prayagsingh/docker-jibri-pulseaudio.git
```

change Dockerfile with specific chrome version

Dockerfile

```
ARG CHROME_RELEASE=96.0.4664.45
ARG CHROMEDRIVER_MAJOR_RELEASE=96
```

create empty file ${CONFIG}/conf/jibri/finalize.sh

build image

```
docker build -t "galileo/jibri:stable-6726-1" .
```

(deprecated) on jitsi meet server

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

```
-- internal muc component, meant to enable pools of jibri and jigasi
clients
Component "internal.auth.csgalileo.org" "muc"
    modules_enabled = {
```

```
      "ping";
    }
    storage = "null"
    muc_room_cache_size = 1000

VirtualHost "jibri.csgalileo.org"
  modules_enabled = {
    "ping";
  }
  authentication = "internal_plain"
```

```
systemctl reload prosody
systemctl status prosody
```

jitsi

```
apt update
apt install -y curl && curl -s
"http://wiki.csgalileo.org/tips:ubuntu:locale?do=export_code&codeblock=1" |
/bin/bash
apt install nginx
wget -qO - https://download.jitsi.org/jitsi-key.gpg.key | sudo apt-key add -
echo 'deb https://download.jitsi.org stable/' >>
/etc/apt/sources.list.d/jitsi-stable.list
apt-get -y update
apt-get -y install jitsi-meet
```

Se la macchina è nattata aggiungere le seguenti proprietà alla configurazione del videobridge:

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

        org.ice4j.ice.harvest.NAT_HARVESTER_LOCAL_ADDRESS=10.45.X.X
        org.ice4j.ice.harvest.NAT_HARVESTER_PUBLIC_ADDRESS=94.230.Y.Y

systemctl restart jitsi-videobridge.service
systemctl restart jicofo
```

HAProxy

global

```
  stats socket /tmp/haproxy
  lua-load /etc/haproxy/routing.lua
```

defaults

```
  timeout connect 5000
  timeout client  50000
  timeout server  50000
  timeout check   10000
  log global
  option httplog
  mode http
  option dontlognull
```

frontend ft_http

```
  bind :80
  use_backend %[lua.custom_router]
  option forwardfor header X-Real-IP
  default_backend bk_http_default
```

frontend ft_https

```
  bind *:443 ssl crt /etc/haproxy/cert.pem alpn h2,http/1.1
  option forwardfor header X-Real-IP
  option httpchk
  use_backend %[lua.custom_router]
  default_backend bk_https_default
  http-response set-header Strict-Transport-Security max-age=31536000;\
includeSubdomains;\ preload
  http-response set-header X-Frame-Options sameorigin
  http-response set-header X-Content-Type-Options nosniff
  http-response set-header X-XSS-Protection 1;mode=block
  http-response set-header Referrer-Policy no-referrer-when-downgrade
```

backend bk_http_default

```
      mode http
      server s1 94.230.76.84:8080 check id 1
```

backend bk_http

```
      mode http
      server s2 localhost:8080 check id 2
```

backend bk_https

```
      mode http
      server s2 localhost:4444 check ssl verify none
      server s1 94.230.76.84:4444 check ssl verify none backup
```

backend bk_https_default

```
      mode http
      server s1 94.230.76.84:4444 check ssl verify none
      server s2 localhost:4444 check ssl verify none backup
```

routing.lua

```
local function router(txn, value)
    local fe_name = txn.f:fe_name()
    local fe_room = txn.f:url_param("room")
    local fe_char = fe_room:byte(1)
    core.Debug("Returning bk_https \n")
    core.Debug(fe_name)
    core.Debug(fe_room)
    core.Debug(fe_char)
    if fe_char % 2 == 1 then
        if fe_name == "ft_https" then
            core.Debug("Returning bk_https \n")
            return "bk_https"
        else
            core.Debug("Returning bk_http \n")
            return "bk_http"
        end
    else
        if fe_name == "ft_https" then
            core.Debug("Returning bk_https \n")
            return "bk_https_default"
        else
            core.Debug("Returning bk_http \n")
            return "bk_http_default"
        end
    end
end

core.register_fetches("custom_router", router)
```

Prosody - upgrade last version

Versione da 0.10.x (Bionic) a 0.11.5

Eseguire preventivamente il backup delle configurazioni in `/etc/prosody`, `/var/lib/prosody`.

```
echo deb http://packages.prosody.im/debian $(lsb_release -sc) main | sudo
tee -a /etc/apt/sources.list
wget https://prosody.im/files/prosody-debian-packages.key -O- | sudo apt-key
add -
apt-get update
apt-get install prosody
```

Modificare lo storage in `/etc/prosody/conf.d/meet.x.y.lua` sostituendo `storage=None` con `storage='memory` in tutti i servizi dove è dichiarato.

Correggere il permesso di lettura al certificato:

```
chmod +r /etc/prosody/certs/localhost.key
```

Eseguire `update-ca-certificates -f` se al riavvio di prosody nei log viene riportato:

```
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path validation failed:
java.security.cert.CertPathValidatorException
```

Eseguire un controllo della porta 5347, se non è in ascolto controllare che in fondo a `/etc/prosody/prosody.cfg.lua` vi sia la riga `Include “conf.d/*.cfg.lua”`

Decommentare in `/etc/prosody/prosody.cfg.lua` eventualmente la riga

```
"posix"; -- POSIX functionality, sends server to background, enables syslog,
etc.
```

Abilitare in `/etc/prosody/prosody.cfg.lua` la tipologia di backend `epoll`:

```
admins = { }
network_backend = "epoll"
```

Riavviare i servizi:

```
systemctl restart prosody
systemctl restart jicofo
systemctl restart jitsi-videobridge2
```

Controllare i logs di questi servizi per eventuali altre sorprese.

LDAP in seguito upgrade

Se si esegue l'upgrade di prosody la versione di lua viene cambiata da `5.1` a `5.2` e l'autenticazione LDAP cessa di funzionare.

Installare `luarocks`

```
apt-get install liblua5.2-dev
cd /tmp
wget https://github.com/luarocks/luarocks/archive/master.zip .
unzip master.zip
cd luarocks-master/
./configure --lua-version=5.2
make build
make install
```

Installare le dipendenze per LDAP

```
apt-get install libldap2-dev
apt-get install libssl1.0-dev  # Questa non c'e' più in ubuntu 20
luarocks install lualdap
luarocks install luacrypto
luarocks install jwt-jitsi
```

Inserire in `/etc/prosody/prosody.cfg.lua`:

```
consider_bosh_secure = true
```

Riavviare i servizi.