Teleport - using local ecflow_ui

Run ecflow_ui at home and interact with ecFlow suites as if you were at the Centre!

- 1. Setting up SSH access
- 2. Connecting to ecFlow servers at the Centre via SSH
  - 2.1. Method #1: Local Port Forwarding
    - 2.1.1. Start SSH port forwarding session
    - 2.1.2. Configure your ecflow_ui
    - 2.1.3. Redirect logserver requests to localhost
  - 2.2. Method #2: Dynamic Port Forwarding
    - 2.2.1. Start SOCKS proxy session
    - 2.2.2. SOCKS-ify your ecflow_ui
- 3. Accessing restricted ecFlow servers
  - 3.1. Servers with username-based access control
  - 3.2. Servers with password-based access control

1. Setting up SSH access

As a prerequisite, you will need SSH access to the Centre. ECMWF Teleport gateway offers relatively hassle-free SSH access service. Follow instructions on the Centre's User Documentation pages to install and set up Teleport client on your laptop.

Each time you connect to the Centre via ssh, you will need a valid SSH certificate. Fresh certificate can be obtained from the Centre's Teleport service by executing `"tsh login"` command on the laptop. These certificates are valid for 12 hours. "tsh login" may ask you for your ECMWF login credentials (security token). If you already have a valid certificate, this authentication prompt will be skipped.

2. Connecting to ecFlow servers at the Centre via SSH

At home, you don't have direct access to the Centre's ecFlow servers. However, you can set up such access using SSH. The basic idea is that the SSH daemon running on your ECMWF workstation will "pretend" to be an ecflow_ui, forwarding network connections from your ecflow_ui at home to the Centre's ecFlow servers. There are two ways of doing it. Both methods rely on the Port Forwarding functionality built into OpenSSH software suite.

2.1. Method #1: Local Port Forwarding

You will start SSH local port forwarding session on your laptop and connect your ecflow_ui to it. The local SSH session will forward network connections to the SSH daemon running on your ECMWF workstation and from there to the Centre's ecFlow servers.

2.1.1. Start SSH port forwarding session

Here I wish to connect to ECMWF workstation hostname machine1 where I have a ecFlow server running on port 4141, also wish to connect to host machine2 where I have a ecFlow server running on port 3142 (replace "user1" with your ECMWF username and "machine1", "machine2" with real ECMWF hostnames). To start the SSH tunnel session on your laptop, execute:

```
ssh -J user1@jump.ecmwf.int user1@machine1 -C -N -L 4141:machine1:4141 -L 3142:machine2:3142
```

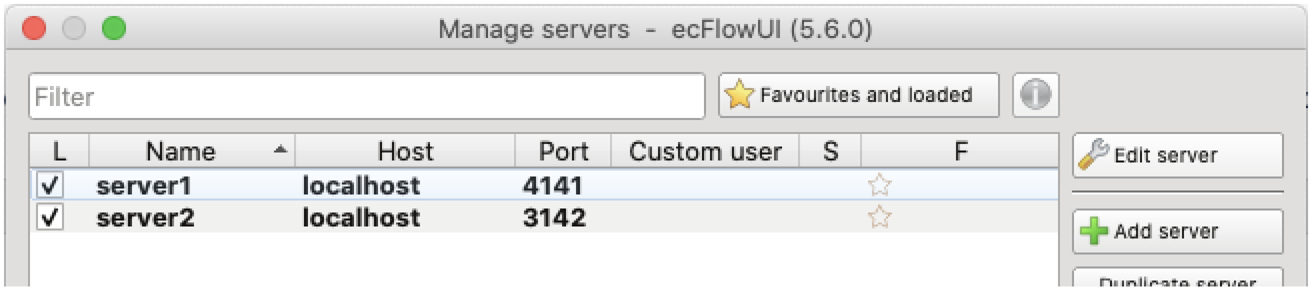
The SSH session will keep running in the terminal, listening on local ports 4141 and 3142 and printing various log messages as it forwards your ecFlow network traffic to the Centre.

To access HPC job output via the logserver, you need to establish a tunnel for this as well. Here we use 'logserver' as an example (replace this with an actual logserver).

```
ssh -J user1@jump.ecmwf.int user1@machine1 -C -N -L 4141:machine1:4141 -L 3142:machine2:3142 -L 9316:logserver:
9316
```

2.1.2. Configure your ecflow_ui

Now you will connect your ecflow_ui to your local SSH port forwarding session. In ecflow_ui, edit ecFlow connection settings (Servers Manage Servers ...). Make sure to use "localhost" in the "Host" fields, not actual ECMWF hostnames.



L	Name	Host	Port	Custom user	S	F
✓	server1	localhost	4141			☆
✓	server2	localhost	3142			☆

2.1.3. Redirect logserver requests to localhost

In Section 2.1.1 you have also established a tunnel for forwarding logserver traffic. In order to send the log requests from ecflow_ui to the local tunnel entrance, you need to add "**127.0.0.1 logserver**" alias in **/etc/hosts** on your laptop (replace "logserver" with an actual ECMWF logserver name).

In practice, using the Local Port Forwarding method we can only connect to a single ECMWF logserver. This is because:

- different logservers at the Centre use the same port number (9316).
- we cannot map all of them to a single 127.0.0.1:9316 local endpoint
- we could map each logserver:9316 to a different local port number, but we cannot reconfigure ecflow_ui to use these local port numbers (i.e. we cannot change ECF_LOGPORT=9316 defined on the ecFlow server without affecting other users).

2.2. Method #2: Dynamic Port Forwarding

SSH also offers Dynamic Port Forwarding, a.k.a. **SOCKS proxy** functionality. This method has some advantages over Local Port Forwarding:

- There is no need to manually specify port number mapping for each ecflow server
- You can use original host names in the ecFlow servers connection settings, instead of 'localhost'
- You are not limited to a single logserver; there is also no need to modify /etc/hosts file.

However, to use the proxy, the client application (ecflow_ui) must be able to speak SOCKS protocol. We will show how to enable it for ecflow_ui.

NOTE: if you were using Local Port Forwarding method, you will now need to:

- Remove "127.0.0.1 logserver" from your /etc/hosts
- Use actual ECMWF hostnames in the "Host" fields of ecflow_ui connection settings, instead of "localhost".

2.2.1. Start SOCKS proxy session

In a terminal on your laptop, start the SOCK proxy (Dynamic Port Forwarding) session with:

```
% ssh -v -C -N -D 9050 -J myecuser@jump.ecmwf.int myecuser@myecworkstation
```

(replace "myecuser" and "myecworkstation" with your real ECMWF username and workstation name).

The session will keep running in the terminal, listening on local port 9050 and printing various log messages as it forwards your network traffic to the Centre.

2.2.2. SOCKS-ify your ecflow_ui

Applications which want to use SOCKS proxy must speak SOCKS protocol. Some applications can be SOCKS-ified using a tool called proxychains. The tool intercepts the application's network traffic, adds a protocol layer and redirects traffic to the proxy. Luckily, the ecflow_ui.x executable can be SOCKS-ified this way.

First, install proxychains in your laptop. On MacOS, you can do it with "`brew install proxychains-ng`". Some Linux distributions come with `proxychains` tool preinstalled.

If you are using ecflow_ui **version >= 5.7.0**, you can start the ui with this command:

```
ecflow_ui -cmd proxychains4
```

However, if you are using an older version you need to edit the ecflow_ui launch script and replace the ***"$exe"*** with ***proxychains4 "$exe"*** at the end of the script. On my machine, ecflow_ui launch script is installed as /opt/miniconda3/bin/ecflow_ui.

You should now be able to start ecflow_ui and interact with ecFlow suites running at the Centre.

By default, proxychains tool sends network traffic to `localhost:9050`. If your SOCKS proxy is listening on a different port, adjust /usr/local/etc/proxychains.conf configuration file (MacOS) accordingly.

3. Accessing restricted ecFlow servers

3.1. Servers with username-based access control

Many ecFlow servers at the Centre use a whitelist to only allow authorized users in. This creates a problem when connecting remotely - typically your username on the laptop will be different from your authorized ECMWF username.

There are two possible ways to access these servers:

- you can ask the ecFlow server administrator to add your laptop username to the server's whitelist, or
- you can create a new user account on your laptop, setting username to match the ECMWF username.

If you create a new user account for ecflow_ui but want to keep using your regular account, here is a handy script for running ecflow_ui as another user. It uses Dynamic Port Forwarding method, but can be easily adapted to use Local Port Forwarding.

A script for starting SSH tunnel + ecflow_ui on a laptop as another user

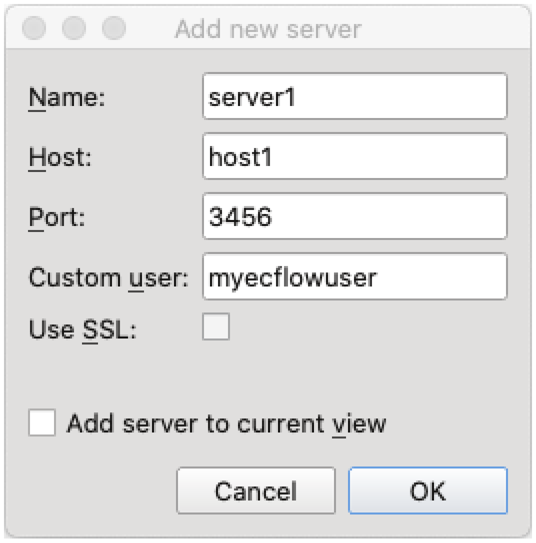
```
#!/bin/bash
set -e
# --------------------------------------------------------------
# A script for starting SSH SOCKS proxy and ecFlow UI as another user.
# Prerequisites:
#   * User named "myecuser" must exist on the laptop
#   * "myecuser" has set up their Teleport client to access ECMWF.
# --------------------------------------------------------------

ECMWF_USER=myecuser              # your ECMWF username
ECMWF_HOST=myecworkstation       # your ECMWF workstation name

xhost + || :
sudo -i -u "$ECMWF_USER" -- sh << SUDO

    set -e

    # Starting ssh-agent for $ECMWF_USER.
    # Teleport service requires ssh-agent running.

    ssh-agent -- sh << SSH_AGENT

        set -e

        # "tsh login" will fetch SSH certificate from
        # jump.ecmwf.int and load it onto the ssh-agent.

        tsh login -d

        # "ssh -f" will start SSH tunnel in the background.
        # The trap will terminate the tunnel on exit.

        trap 'pkill -f ssh.*-f' 0 1 2 3 15
        ssh -f -N -v -C -D9050 -J "$ECMWF_USER@jump.ecmwf.int" "$ECMWF_USER@$ECMWF_HOST"

        # finally, start the local ecFlow UI
        /Volumes/Macintosh\ HD/opt/miniconda3/bin/ecflow_ui
SSH_AGENT
SUDO
```

3.2. Servers with password-based access control

As an alternative to username-based access control, ecFlow offers password-based access control. See: Security(custom user).

The password-based access control must be enabled on the ecFlow server side. Discuss with your server administrator to set it up.

When the server-side setup is done by the administrator, configure the connection settings in your ecflow_ui:



Add new server

Name: server1

Host: host1

Port: 3456

Custom user: myecflowuser

Use SSL: ☐

☐ Add server to current view

Replace "host1" with a real ECMWF ecFlow server hostname and "myecflowuser" with your authorised ecFlow username, for which you have the password.