Multi-User 0.5.x Documentation

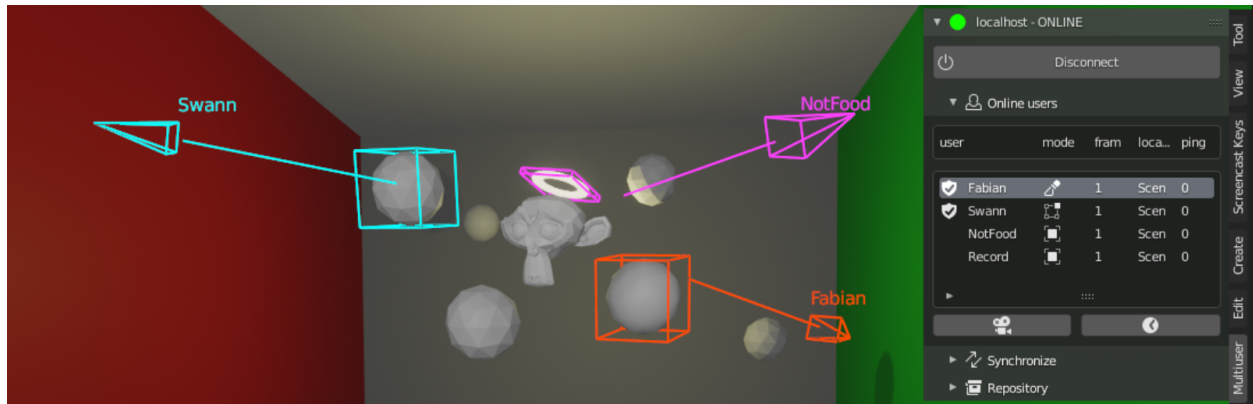# Contents

Welcome to the manual of the Multi-user, a free and open source blender addon. It tool aims to bring multiple users to work on the same .blend over the network.

Join our discord server to get help and join collaborative creation sessions.

> **Warning:** Still under development

CHAPTER 1

Getting started

Sections

2.1 About Multi-User

2.1.1 The idea

A film is an idea carved along the whole production process by many different peoples. A traditional animation pipeline involve a linear succession of tasks. From storyboard to compositing by passing upon different step, its fundamental work flow is similar to an industrial assembly line. Since each step is almost a department, its common that one person on department B doesn't know what another person did on a previous step in a department A. This lack of visibility/communication could be a source of problems which could produce a bad impact on the final production result.

Fig. 1: The linear workflow problems

Nowadays it's a known fact that real-time rendering technologies allows to speedup traditional linear production by reducing drastically the iteration time across different steps. All majors industrial CG solutions are moving toward real-time horizons to bring innovative interactive workflows. But this is a microscopic, per-task/solution vision of real-time rendering benefits for the animation production. What if we step-back, get a macroscopic picture of an animation movie pipeline and ask ourself how real-time could change our global workflow ? Could-it bring better ways of working together by giving more visibility between departments during the whole production ?

The multi-user addon is an attempt to experiment real-time parallelism between different production stage. By replicating blender data blocks over the networks, it allows different artists to collaborate on a same scene in real-time.

2.1.2 Key Features

Multi-User is a free and open source blender addon. It aims to allow multiple users to work on the same scene over the network. Based on a Clients / Server architecture, the data-oriented replication schema replicate blender data-blocks across the wire.

> **Warning:** The addon is still in development Be carefull when using it.
>
> On rare occasions, it can happen that your blender scenes become corrupted, think of making backups to avoid losing your projects.

Collaboration

Multi-User allows a strong collaborative workflow between users. Being able to collaborate in this way has opened up new opportunities:

- Being able to create together and in real time on the same 3D scene, with instant feedback.
- Being able to teach directly in the same 3D environment in real time, facilitating communication between the teacher and these students.
- To be able to experiment with several people, to make challenges or simply to have fun.
- And much more !

Easier communication

Thanks to *presence*, the overlay system that Multi-User provides, it is possible to see other users in the 3D space. The *presence* overlay is customizable to match your preferences (visibility, names, options).

Session management

The addon works on a session system. The creator of the session and the administrators have rights that allow them to easily manage the session (backups, user management). In addition, there is a management of datablock rights so that each user can collaborate as they wish.

2.1.3 Community

Discord

Feel free to join our discord server !

You will find help, a way to take part in the project, public collaborative sessions, people to create with, information about the addon's progress and much more.

Contributors

Swann, Fabian, NotFood, Poochyc, Valentin, Adrien, Tanguy, Bruno, Gorgio, Axel, Ultr-X, Wuaieyo, Softyoda, Staz, Ikxi, Kysios.

2.2 Getting started

2.2.1 Installing Multi-User

> **Warning:** Under development, use it at your own risks.

Multi-User is often updated. You can keep up to date with the latest changes through the release notes on our Discord Server.

Download

Stable Release **Recommended**. A package packed with the latest features and is considered stable without regressions.

Latest Release **Experimental**. A package updated almost daily to include the newest changes in development. These versions are not as thoroughly tested as the stable release, and might break.

Install

> **Hint:** The process is the same for linux, mac and windows.

1. Download the addon zip file
2. Run blender as administrator (to allow python dependencies auto-installation).
3. Install **multi-user.zip** from your addon preferences in *Edit → Preferences → Add-ons → Install*.

Once the addon is succesfully installed, we strongly recommend you to follow the *Quick Start* tutorial.

2.2.2 Update the Addon

Multi-User has a built-in auto-update function in its preferences.

Auto-Update

1. Enable it by clicking 'Auto-check for Update' and choose the frequency you'd like.
2. **Make sure to click the three bars in the bottom-left, and save this to your preferences**

Manual Update

Sometimes you'd like to perform manual update, or even side-grade or rollback your multi-user version. Perhaps you are trying out new features from the 'develop' branch in a test session.

1. Click on 'Check now for multiuser update'. Multi-user will now find new versions
1. Select 'Install latest master / old version'
1. In most cases, select 'master' branch for the latest stable release. The unstable 'develop' branch and older releases are available
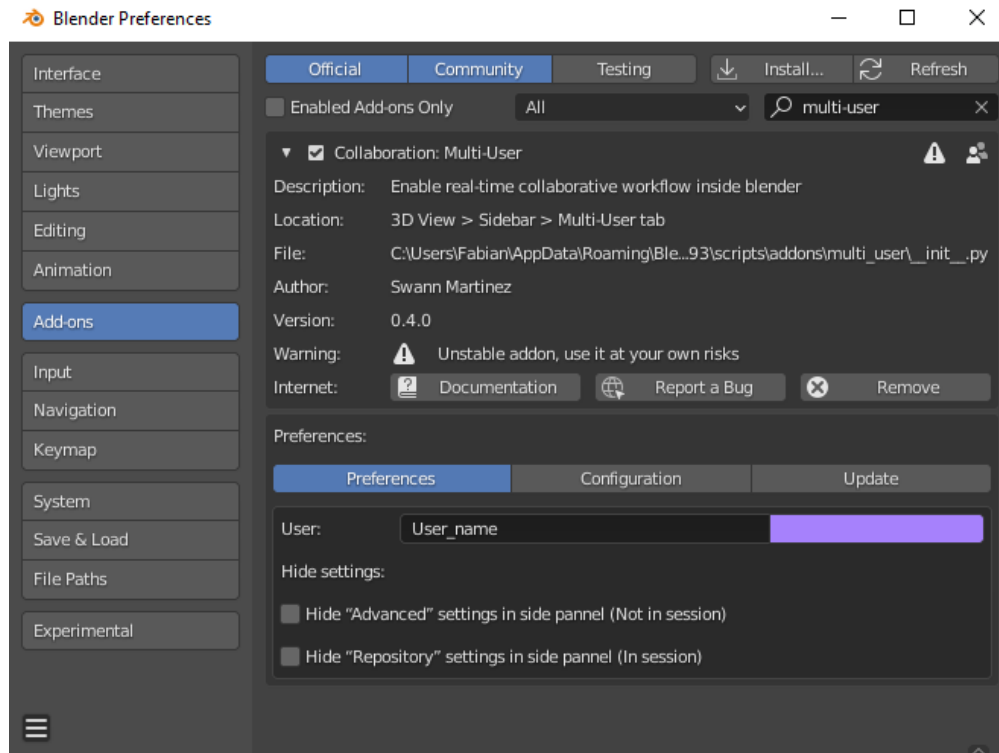
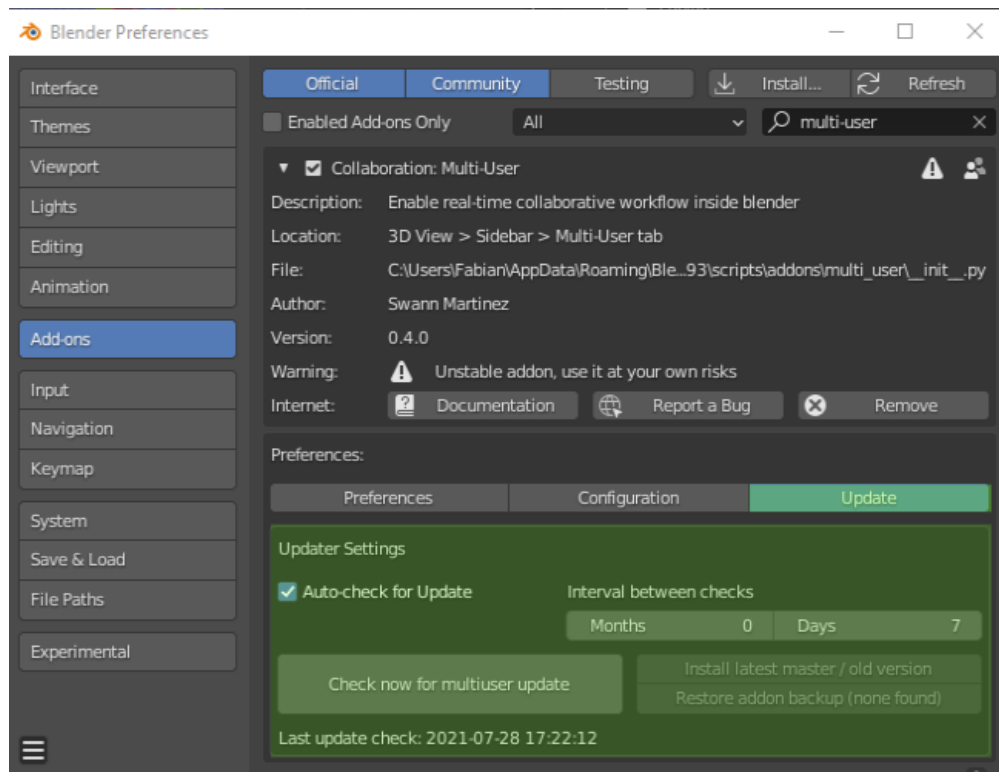Fig. 2: The Addon Preferences Pannel
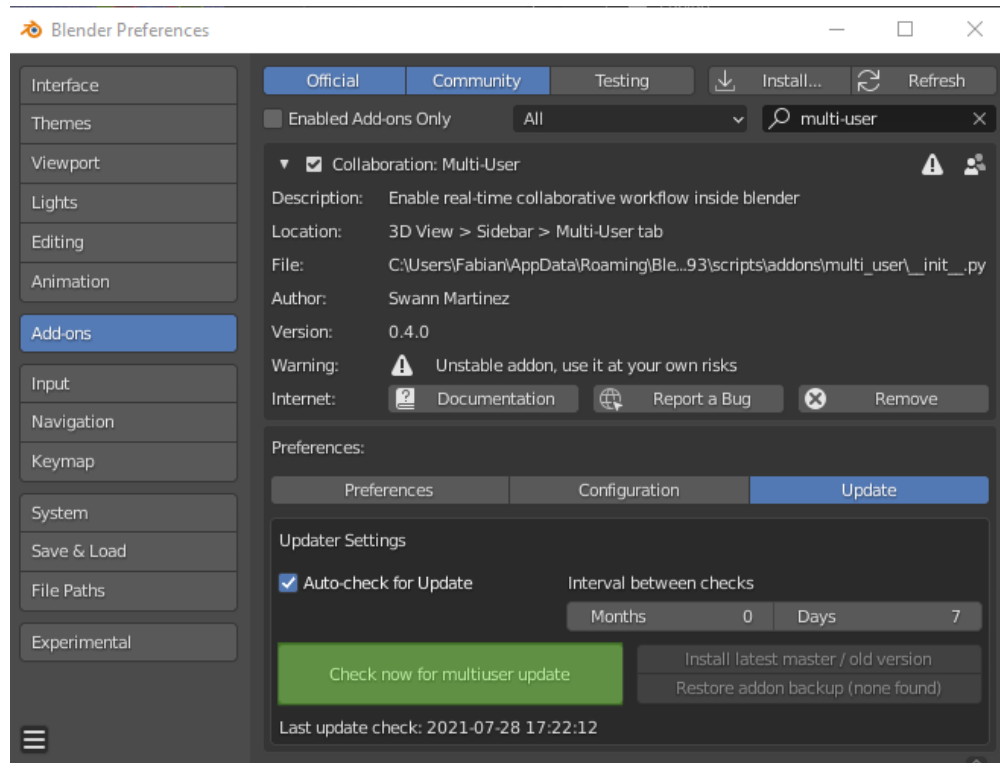

Fig. 3: Update menu in the addon preferences pannel
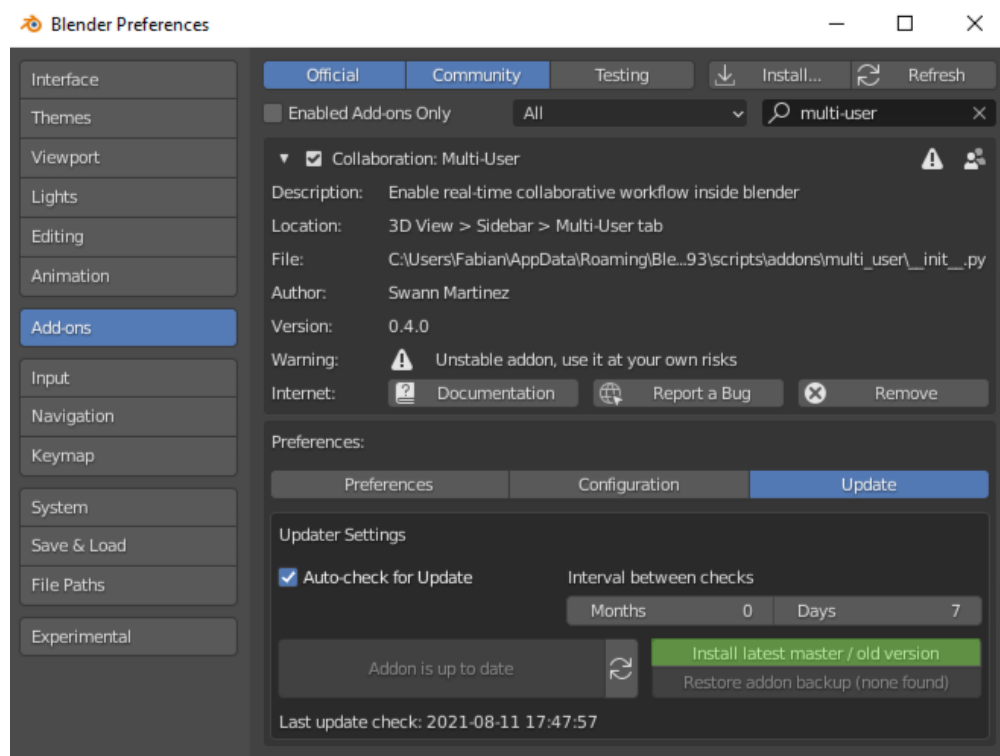
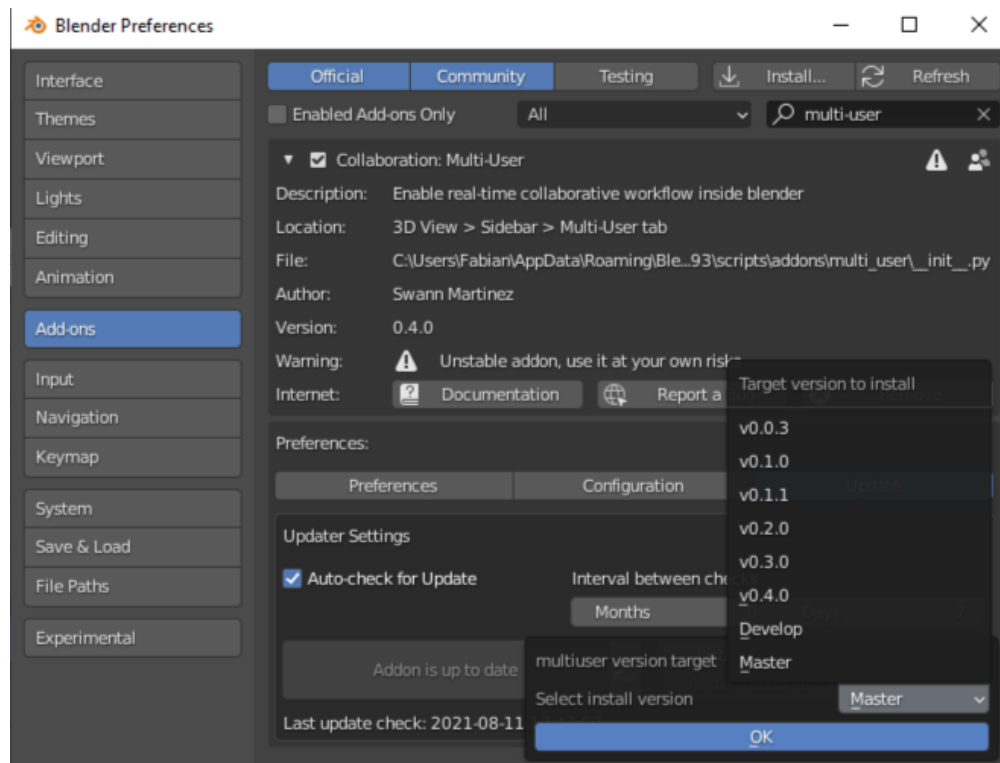Fig. 4: Check for updates

Fig. 5: Install

Fig. 6: Select version

4. Finally, restart blender to use the updated version

2.2.3 Quick Start

Welcome to the Multi-User manual. You will find here all the documentation necessary for the good use of the addon: *Multi-user 0.5.0 Reference Manual*

First of all, let's have a quick look at the Multi-User features.

Username and color

When you launch the addon for the first time you can find this panel in the Sidebar of your View3D:

1. Choose a **name** and a **color** that will be specific to you and that will allow others to identify you easily once in session. Don't worry, they can be changed at any time in *Edit → Prerecences → Add-ons → Multi-user* or in *Multi-User Pannel → General Settings*.

2. Press **Continue**

Multi-User side pannel

Once the Multi-User is launched you will arrive directly on the main menu:
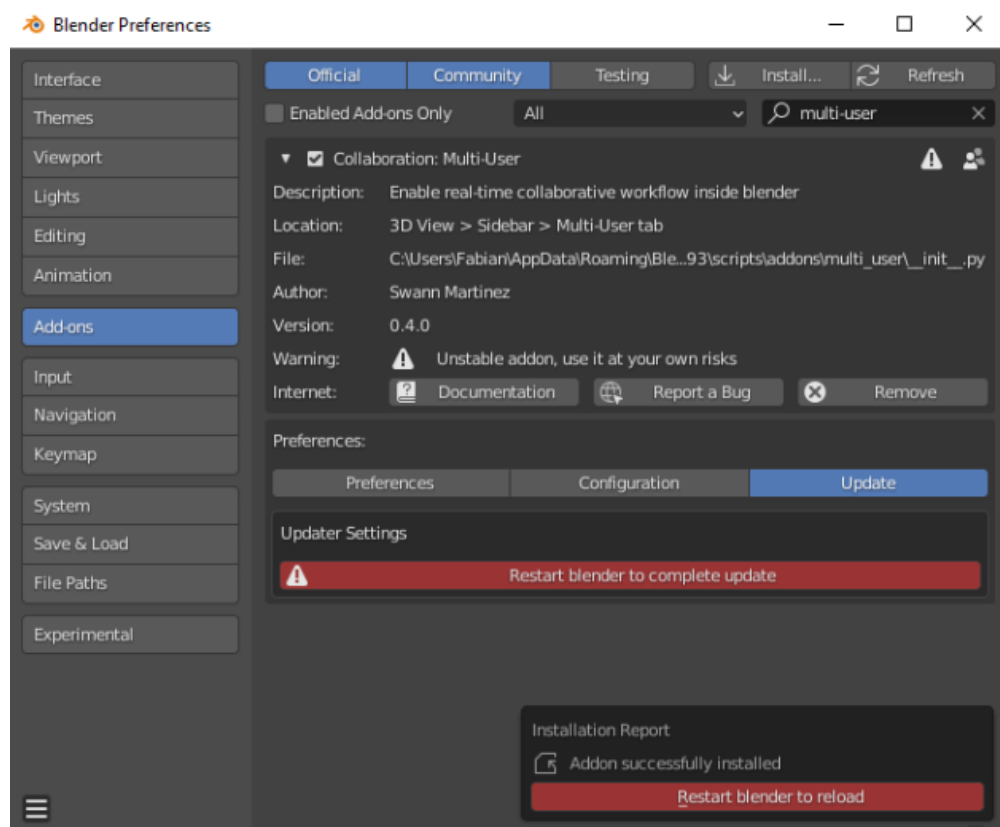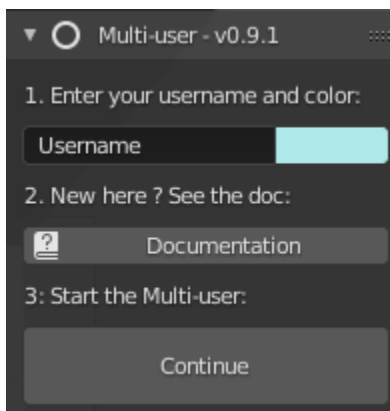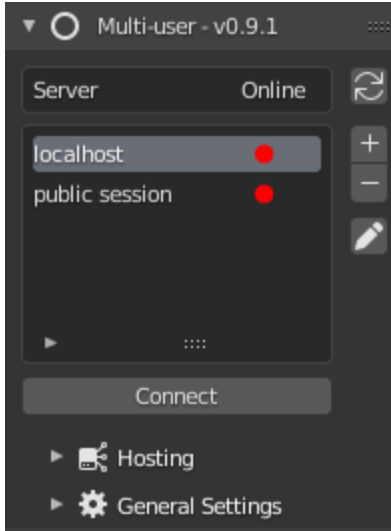
Three panels are at your disposal:

Fig. 7: Restart blender

- **Server list**: You can add, delete and edit server presets according to your preferences. At first launch two servers will already be in your preferences: *Public Session*, the public server of the Multi-User Discord, *Localhost*, to connect locally to your server.

- **Hosting**: To locally host a session with a Blender instance.

- **General Settings**: Include advanced addon settings like *user info*, *server ping*, *cache*, etc.

Session management

The multi-user addon provides a session management system. In this guide, you will quickly learn how to use the collaborative session management system in three parts:

- *How to join a session*

- *How to host a session*

- *How to manage a session*

For more details on what the addon offers:

2.2.4 How to join a session

This section describes how to join a launched session. Before starting make sure that you have access to the session **IP address**, **port number** and that you have filled in your **user information** (name and color).

Server List

The server list allows you to manage your servers:

To connect to a server, select the one you want to join in the list and click on **Connect**.

To know if the server you want to join is online, you can refresh your server list with the button on the top right corner. Online status:

- **Red**: server is offline
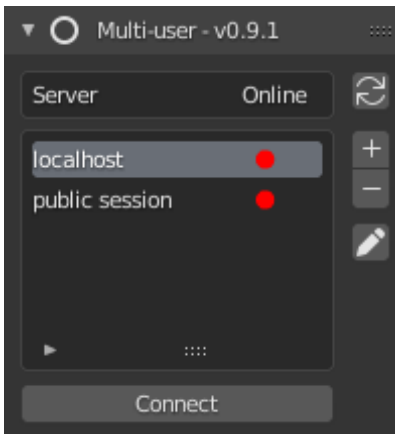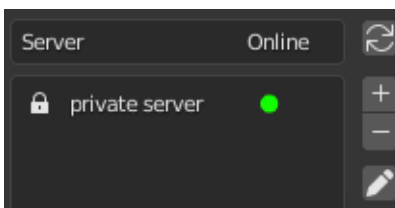
- **Green**: server is online

Fig. 8: Server List

Note: If a server is secured with a password, a lock will be displayed next to the server name. You first need to enter the password of the server in its preset to join it.

It is possible to **add**, **delete** and even **modify** a **server preset** with the buttons located on the top right of the server list:
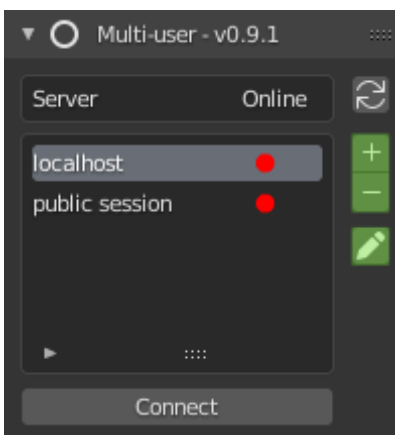

Fig. 9: Add, Remove, Edit Server Preset

Note: Two server presets are already present when the addon is launched:

- The 'localhost' preset, to join a local session quickly

- The 'public session' preset, to join the public sessions of the multi-user server (official discord to participate : https://discord.gg/aBPvGws)

Add a Server Preset

To add a server, you must first register it in the server list. Click on the **+** icon and fill in the window with the server settings:
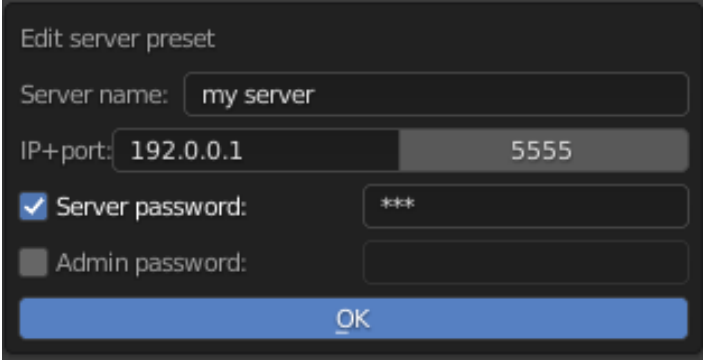

Fig. 10: Server Preset pop-up

- **Server name**: the name of the server.
- **IP**: the host's IP address.
- **Port**: the host's port number.
- **Server password**: (*optional*) the server password.
- **Admin password**: (*optional*) the session administration password.

Once you've configured every field, you can save the server preset by clicking **OK**. You can now select it in the server list to join the session !

Warning: Be careful, if you don't rename your new preset, or if it has the same name as an existing preset, the old preset will be overwritten.

Joining a server

CONNECT

When joining a server that have already be initialise, the session status screen will be **CONNECT**. You are now connected and can start creating.

During an online session, various actions are available to you. Go to *How to manage a session* to learn more about them.

LOBBY

When starting a **dedicated server**, the session status screen will take you to the **LOBBY** (see side-panel header).
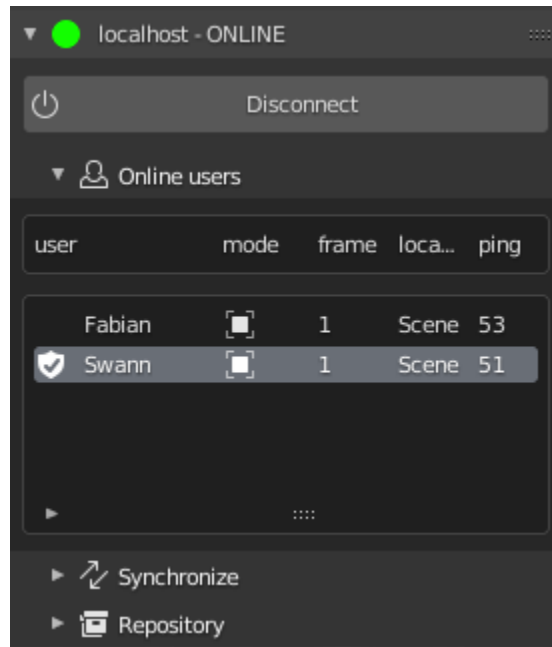
Fig. 11: In session

If the session status is set to **LOBBY** and you are a regular user, you need to wait for the admin to launch the scene (admins have shield next to their names). If you are the admin, you just need to initialise the session to start it (see image below).

2.2.5 How to host a session

Local server

The multi-user add-on relies on a Client-Server architecture. The server is the heart of the collaborative session. It is what allows user's blender instances to communicate with each other. In simple terms, *Hosting a session* means *run a local server and connect the local client to it*. When we say **local server** we mean a server which is accessible from the LAN (Local Area Network) without requiring an internet connection.

When the hosting process starts, the multi-user addon will launch a local server instance. In the **Hosting** panel configure your server according to:

- **Init the session from**: the session initialisation method.

  - **current scenes**: start with the data loaded in the current blend file.

  - **an empty scene**: clear the blend file's data and start over.

- **Port**: port on which the server is listening.

- **Server password**: (*optional*) the server password.

- **Admin password**: (*optional*) the session administration password.

Once everything is set up, you can hit the **Host** button to launch the session!

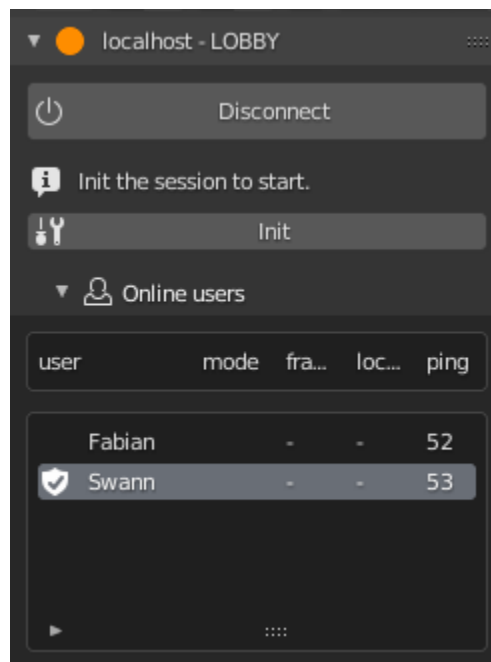This will do two things:

- Start a local server

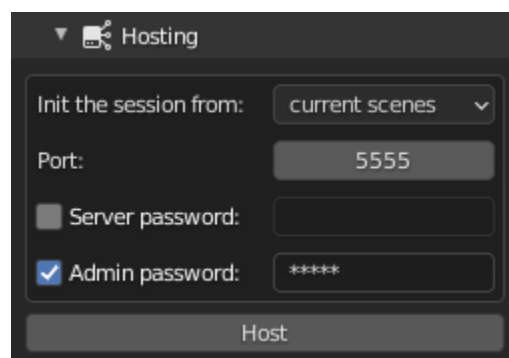Fig. 12: Session initialisation for dedicated server

Fig. 13: Hosting panel

- Connect you to it as an admin

> **Danger:** By starting from an empty scene, all of the blend data will be removed! Be sure to save your existing
> work before launching the session.

Online server

However, there are times when you will need to host a session over the internet. In this case, we strongly recommend
that you read the *Hosting on Internet* tutorial.

During an online session, various actions are available to you, go to *How to manage a session* section to learn more
about them.

2.2.6 How to manage a session

The quality of a collaborative session directly depends on the quality of the network connection, and the communication between the users. This section describes various tools which have been made in an effort to ease the communication between your fellow creators. Feel free to suggest any ideas for communication tools here .

Monitor online users

One of the most vital tools is the **Online user panel**. It lists all connected users' information including your own:

- **Role** : admin/regular user.
- **Username** : name of the user.
- **Mode** : user's active mode (object, sculpt, paint,etc.).
- **Frame**: on which frame the user is working.
- **Location**: where the user is actually working.
- **Ping**: user's connection delay in milliseconds.

By selecting a user in the list you'll have access to different users' related **actions**. Those operators allow you to
experience the selected user's state in two different dimensions: **SPACE** and **TIME**.

Snapping in space

The **CAMERA button** (Also called **snap view** operator) allow you to snap to the user's viewpoint. To disable the
snap, click on the button once again. This action serves different purposes such as easing the review process, and
working together on a large or populated world.

> **Hint:** If the target user is located in another scene, the **snap view** operator will send you to their scene.

Snapping in time

The **CLOCK button** (Also called **snap time** operator) allows you to snap to the user's time (current frame). To disable
the snap, click on the button once again. This action helps various multiple creators to work in the same time-frame
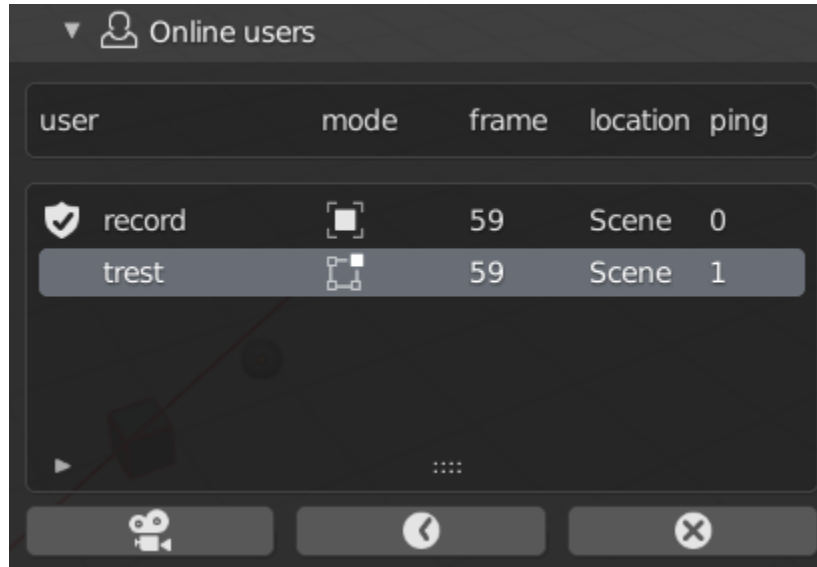(for instance multiple animators).

Fig. 14: Online user panel

Fig. 15: Snap view in action

Kick a user

> **Warning:** Only available for admin !

The **CROSS button** (Also called **kick** operator) allows the administrator to kick the selected user. This can be helpful if a user is acting unruly, but more importantly, if they are experiencing a high ping which is slowing down the scene. Meanwhile, in the target user's world, the session will properly disconnect.

Change replication behavior

During a session, multi-user will replicate all of your local modifications to the scene, to all other users' blender instances. In order to avoid annoying other users when you are experimenting, you can flag some of your local modifications to be ignored via various flags present at the top of the panel (see red area in the image below). Those flags are explained in the replication section.

Manage data

In order to understand replication data managment, a quick introduction to the multi-user data workflow is in order. The first thing to know: until now, the addon relies on data-based replication. In simple words, it means that it replicates the resultant output of a user's actions. To replicate datablocks between clients, multi-user relies on a standard distributed architecture:

- The server stores the "master" version of the work.
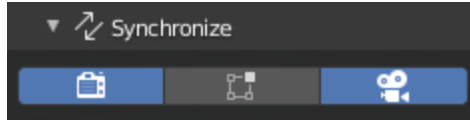
Fig. 16: Snap time in action

Fig. 17: Session replication flags

- Each client has a local version of the work.

When an artist modifies something in the scene, here is what is happening in the background:

1. Modified data are **COMMITTED** to the local repository.

2. Once committed locally, they are **PUSHED** to the server

3. As soon as the server receives updates, they are stored locally and pushed to every other client

At the top of this data management system, a rights management system prevents multiple users from modifying the same data at the same time. A datablock may belong to a connected user or be under common-right rights.

Note: In a near future, the rights management system will support roles to allow multiple users to work on different aspects of the same datablock.

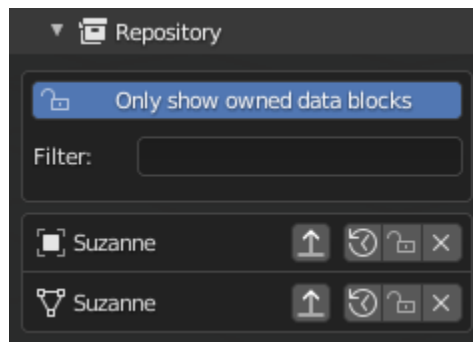The Repository panel (see image below) allows you to monitor, change datablock states and rights manually.

Fig. 18: Repository panel

The **show only owned** flag allows you to see which datablocks you are currently modifying.

Warning: If you are editing a datablock not listed with this flag enabled, it means that you have not been granted the rights to modify it. So, it won't be updated to other clients!

Here is a quick list of available actions:

icon	Action	Description
▲	**Push**	push data-block to other clients
▼	**Pull**	pull last version into blender
🕘	**Reset**	Reset local change to the server version
🔒	**Lock/Unlock**	If locked, does nothing. If unlocked, grant modification rights to another user.
✕	**Delete**	Remove the data-block from network replication

2.3 User Interface

2.3.1 General Pannels

To understand the UI of the addon's pannels, see: *Quick Start* (+how to join/host/manage). In addition to presenting the UI, it explains how to use it.

2.3.2 Presence

Presence is the multi-user module responsible for displaying user presence. During the session, it draw users' related information in your viewport such as:

- Username
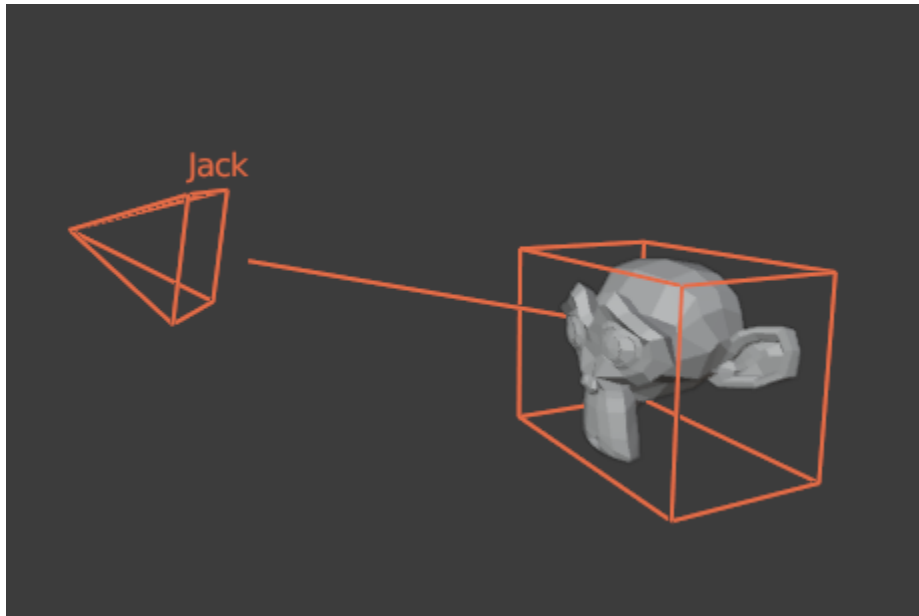- User point of view
- User active mode
- User selection

Fig. 19: Presence overlay

Presence settings

The presence overlay panel allows you to enable/disable various drawn parts via the following flags:

- **Presence Overlay**: display presence overlay
- **Selected objects**: display other users' current selections
- **Users camera**: display users' current viewpoint
- **Users mode**: display users' current mode
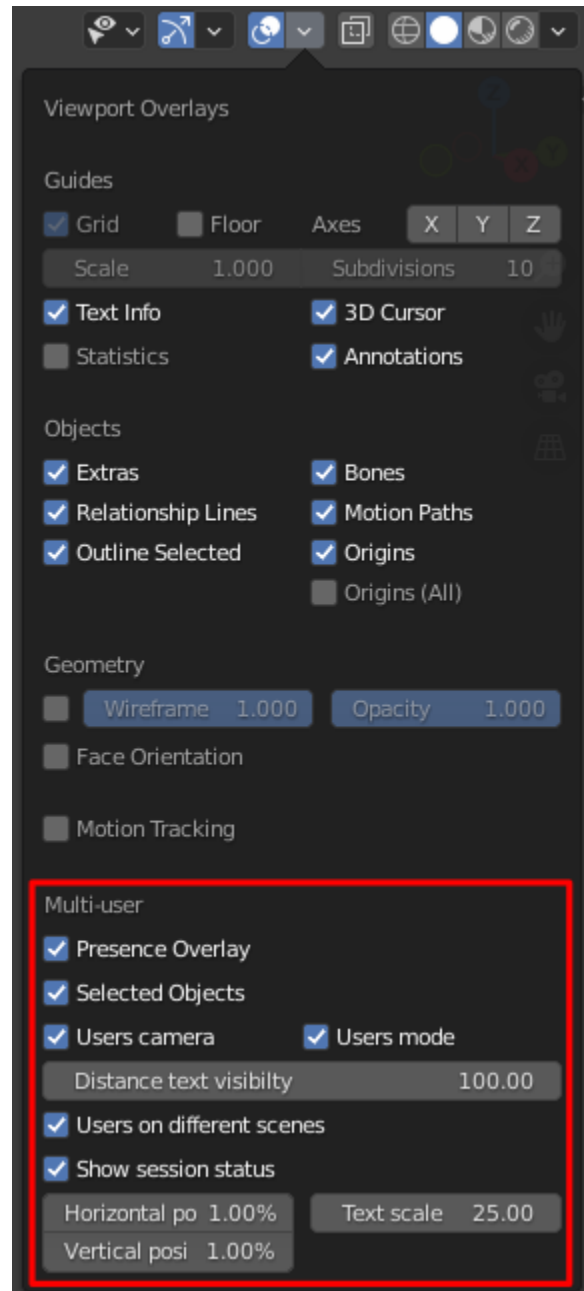- **Distance text visibility**: display text of the overlay at this maximal distance

Fig. 20: Presence overlay settings

- **Users on different scenes**: display users working on other scenes
- **Show session status**: display the session status in the viewport
  - **Vertical/Horizontal position**: session position in the viewport
  - **Text scale**: session status text size

2.4 General Settings

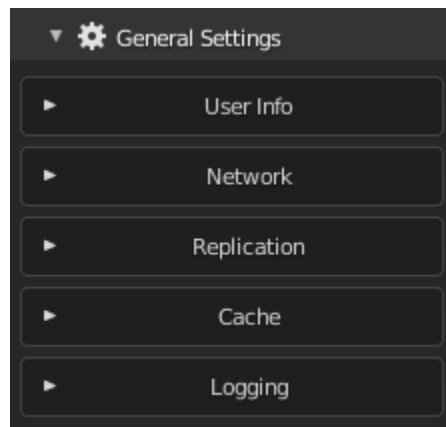This section contains optional settings to configure before a session.
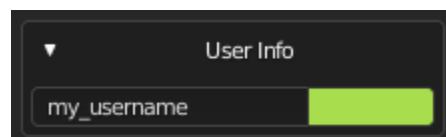

Fig. 21: General Settings pannel

2.4.1 User info

Fig. 22: User Info settings

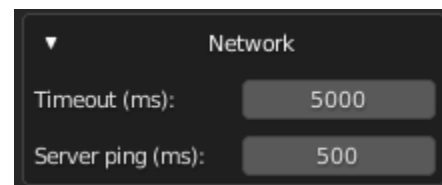The **User Info** pannel is here to change your user name and use color.

2.4.2 Network

Fig. 23: Network settings

Timeout (in milliseconds) is the maximum ping authorized before auto-disconnecting. You should only increase it if you have a bad connection.
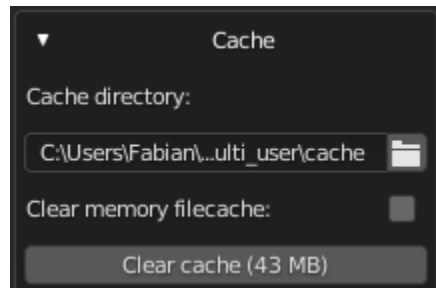
2.4.3 Cache

Fig. 24: Cache settings

Multi-user allows you to replicate external dependencies such as images (textures, hdris, etc. . .), movies, and sounds. On each client, the files will be stored in the multi-user cache folder.

Cache directory choose where cached files (images, sound, movies) will be saved.

Clear memory filecache will save memory space at runtime by removing the file content from memory as soon as it has been written to the disk.

Clear cache will remove all files from the cache folder.

> **Warning:** Clearing the cache could break your scene images/movies/sounds if they are used in a blend file! Try saving the blend file and choosing 'Pack all into blend' before clearing the cache.
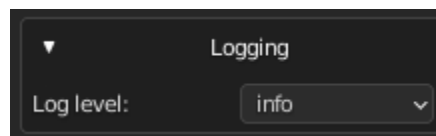
2.4.4 Logging

Fig. 25: Advanced log settings

log level allows you to set the level of detail captured in multi-user's logging output. Here is a brief description on the level of detail for each value of the logging parameter:

Log level	Description
ERROR	Shows only critical errors
WARNING	Shows only errors (of all kinds)
INFO	Shows only status-related messages and errors
DEBUG	Shows all possible information

2.4.5 Save session data

> **Danger:** This is an experimental feature, it is still recommended to use regular .blend save.

The save session data allows you to create a backup of the session data.

When you hit the **save session data** button, the following popup dialog will appear. It allows you to choose the destination folder and if you want to run an auto-save.
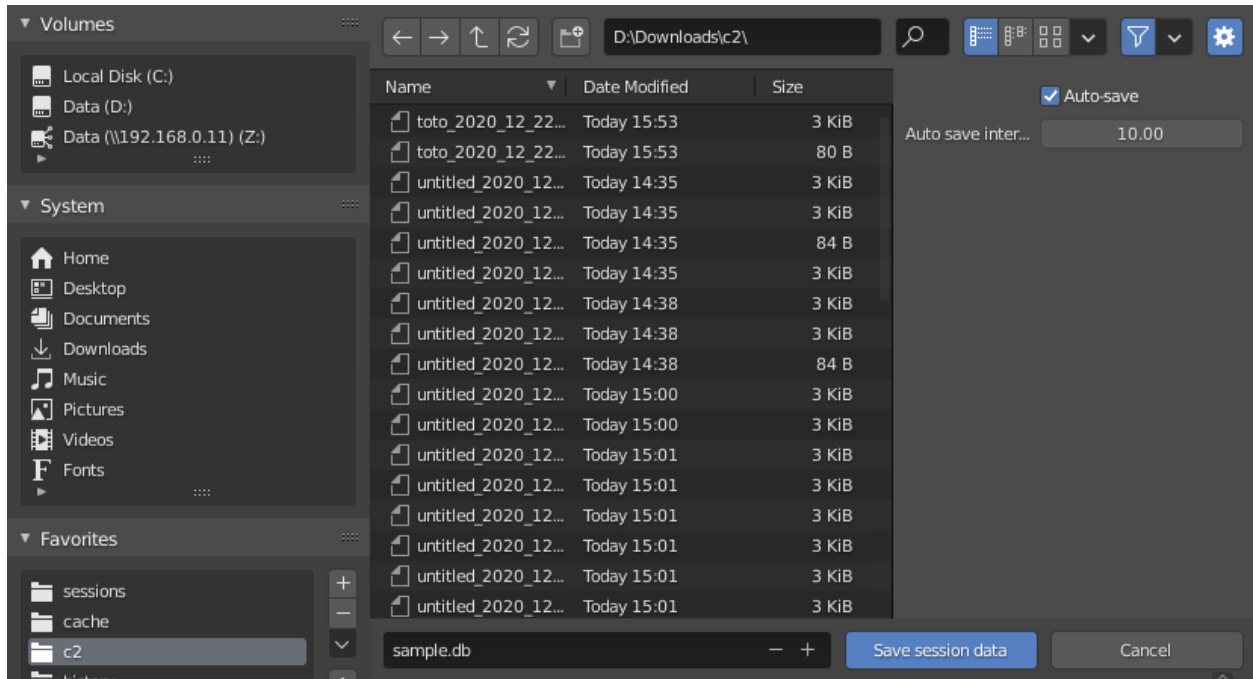

Fig. 26: Save session data dialog.

If you enabled the auto-save option, you can cancel it from the **Cancel auto-save** button.

To import session data backups, use the following **Multiuser session snapshot** import dialog

> **Note:** It is not yet possible to start a session directly from a backup.

2.5 Hosting on Internet

> **Warning:** Until now, those communications are not encrypted but are planned to be in a mid-term future (status).

This tutorial aims to guide you toward hosting a collaborative multi-user session on the internet. Hosting a session can be achieved in several ways:

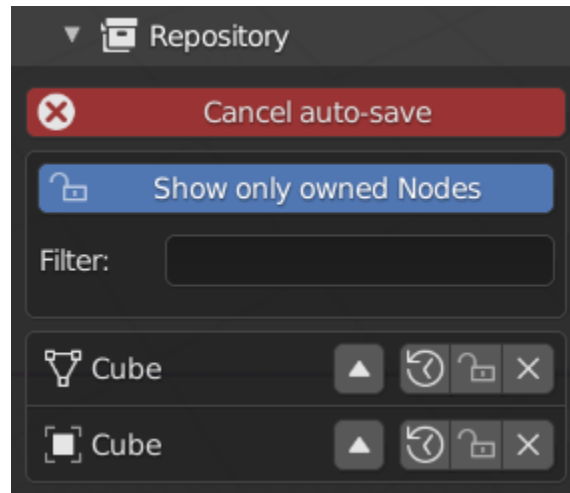- *From blender*: hosting a session directly from the blender add-on panel.
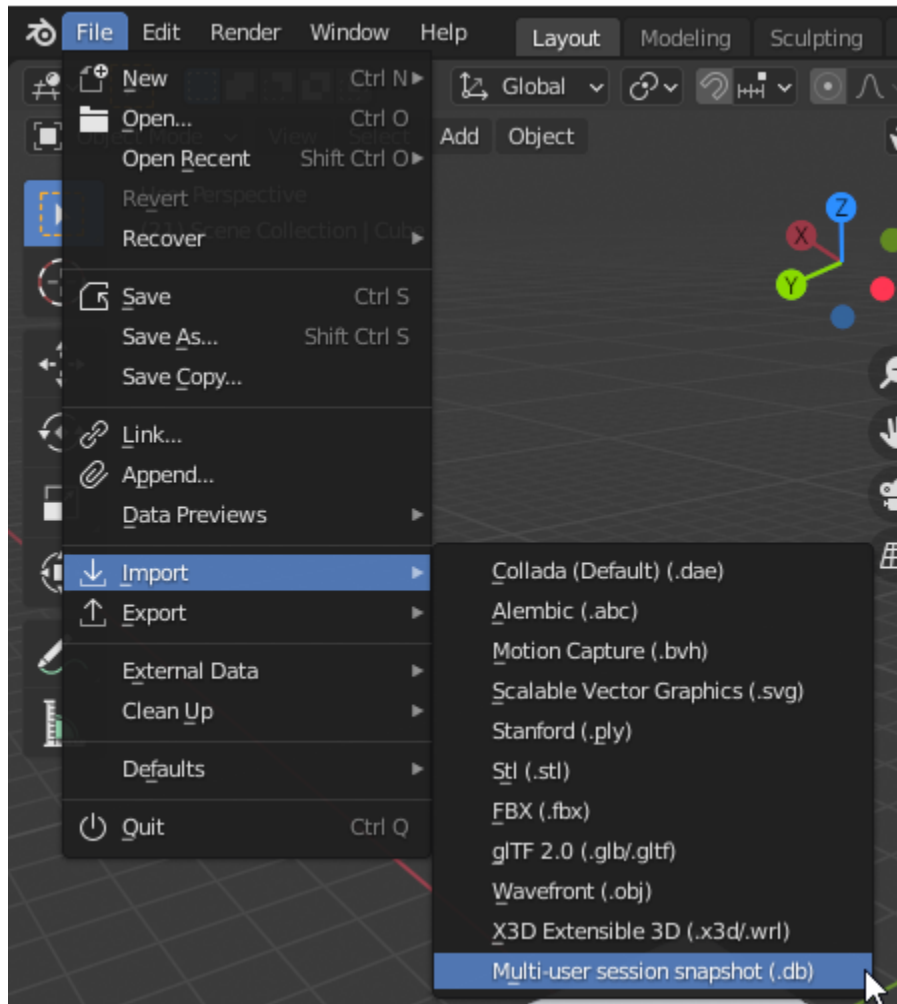
Fig. 27: Cancel session autosave.

Fig. 28: Import session data dialog.

- *From the dedicated server*: hosting a session directly from the command line interface on a computer without blender.

- *Cloud Hosting Walkthrough*: hosting a session on a dedicated cloud server such as Google Cloud's free tier.

2.5.1 From blender

By default your router doesn't allow anyone to share you connection. In order grant the server access to people from internet you have two main option:

- The *Using a connection sharing solution*: the easiest way.

- The *Using port-forwarding*: this way is the most unsecure. If you have no networking knowledge, you should definitely follow *Using a connection sharing solution*.

Using a connection sharing solution

You can either follow Pierre Schiller's excellent video tutorial or jump to the *text tutorial*.

Many third party software like ZEROTIER (Free) or HAMACHI (Free until 5 users) allow you to share your private network with other people. For the example I'm gonna use ZeroTier because it's free and open source.

1. Installation

Let's start by downloading and installing ZeroTier: https://www.zerotier.com/download/

Once installed, launch it.

2. Network creation

To create a ZeroTier private network you need to register a ZeroTier account on my.zerotier.com (click on **login** then register on the bottom)

Once you account it activated, you can connect to my.zerotier.com. Head up to the **Network** section (highlighted in red in the image below).

Hit 'Create a network'(see image below) and go to the network settings.

Now that the network is created, let's configure it.

In the Settings section(see image below), you can change the network name to what you want. Make sure that the field **Access Control** is set to **PRIVATE**.

Hint: If you set the Access Control to PUBLIC, anyone will be able to join without your confirmation. It is easier to set up but less secure.

That's all for the network setup ! Now let's connect everyone.

3. Network authorization

Since your ZeroTier network is Private, you will need to authorize each new user to connect to it. For each user you want to add, do the following step:
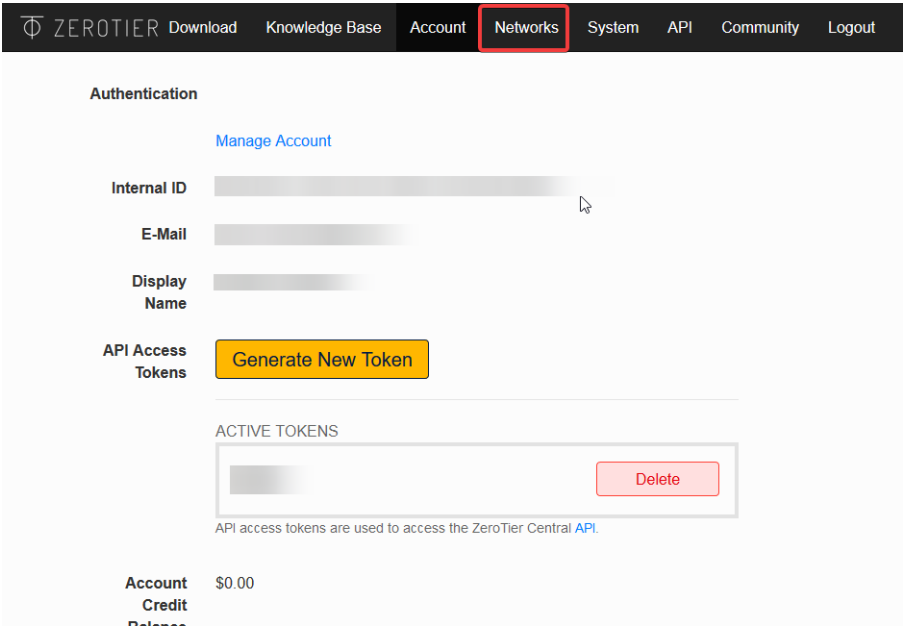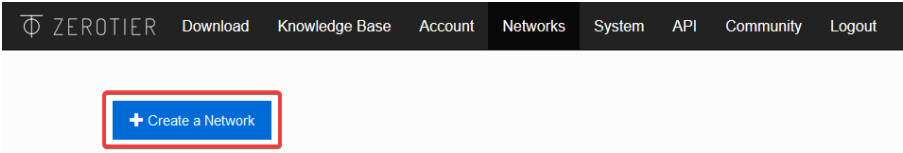

Fig. 29: ZeroTier user homepage

Fig. 30: Admin password

Fig. 31: Network settings

1. Get the client **ZeroTier id** by right clicking on the ZeroTier tray icon and click on the *Node ID*, it will copy it.
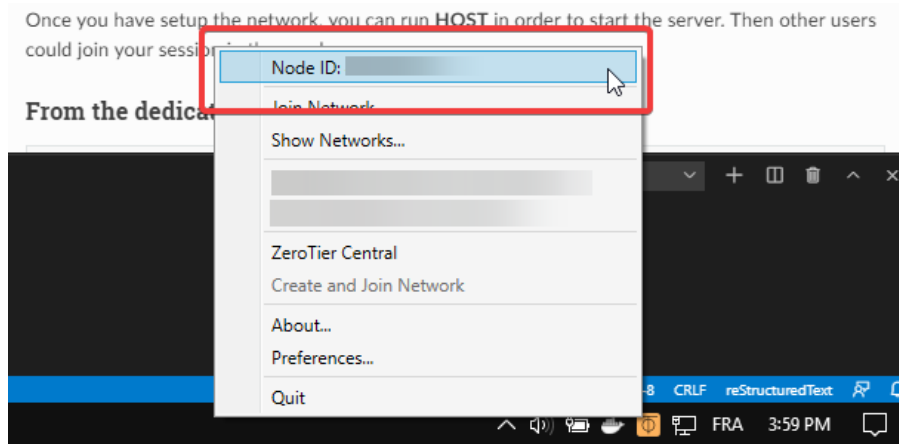

Fig. 32: Get the ZeroTier client id

2. Go to the network settings in the Member section and paste the Node ID into the Manually Add Member field.

4. Network connection

To connect to the ZeroTier network, get the network id from the network settings (see image).

Now we are ready to join the network ! Right click on the ZeroTier tray icon and select **Join Network** !

Past the network id and check `Allow Managed` then click on join ! You should be connected to the network.

Let's check the connection status. Right click on the tray icon and click on **Show Networks...**.

The network status must be **OK** for each user(like in the picture above) otherwise it means that you are not connected to the network. If you see something like **ACCESS_DENIED**, it means that you were not authorized to join the network. Please check the section *3. Network authorization*

This is it for the ZeroTier network setup. Now everything should be setup to use the multi-user add-on over internet ! You can now follow the *Quick Start* guide to start using the multi-user add-on !

Using port-forwarding

The port forwarding method consists of configuring your network router to deny most traffic with a firewall, but to then allow particular internet traffic (like a multiuser connection) through the firewall on specified ports.

In order to know which ports are used by the add-on, please check the *Port setup* section. To set up port forwarding for each port you can follow this guide for example.

Once you have set up the network you can follow the *Quick Start* guide to begin using the multi-user add-on !
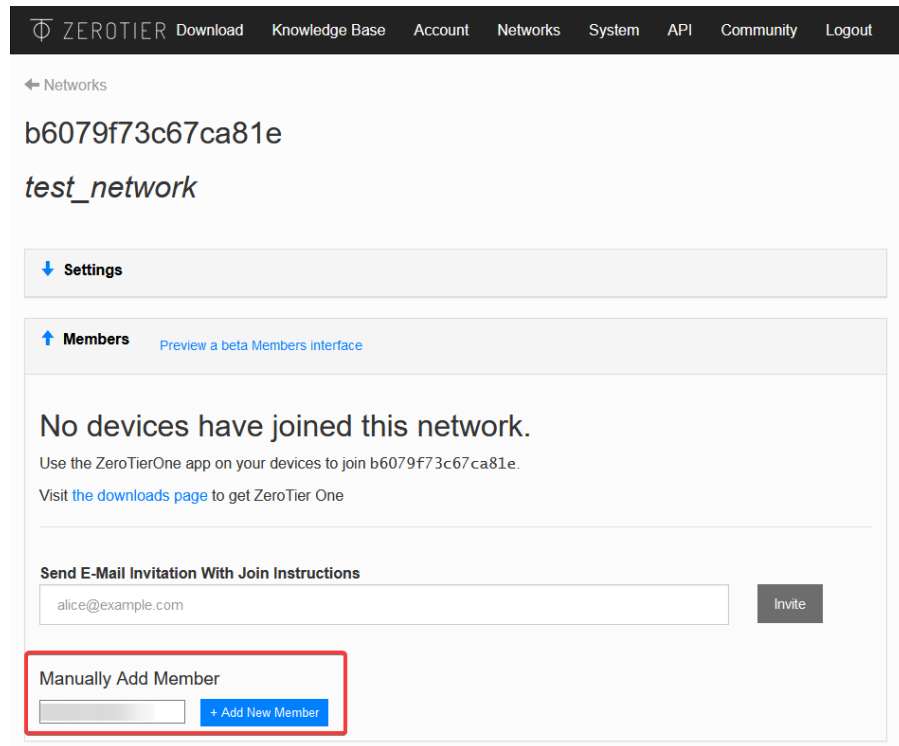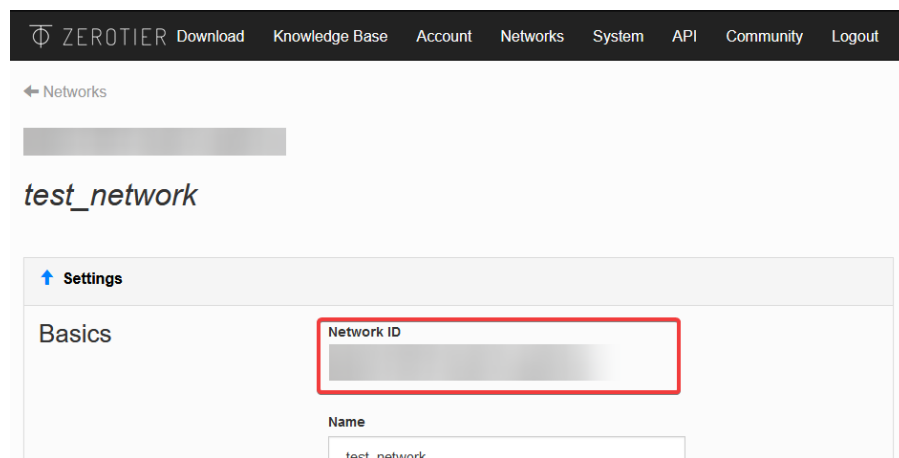
2.5.2 From the dedicated server

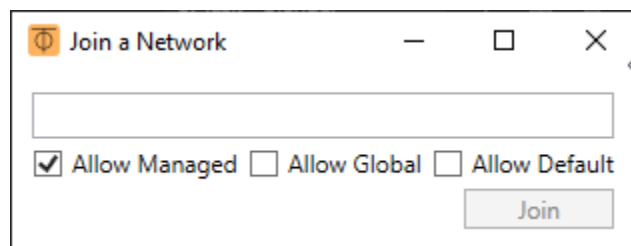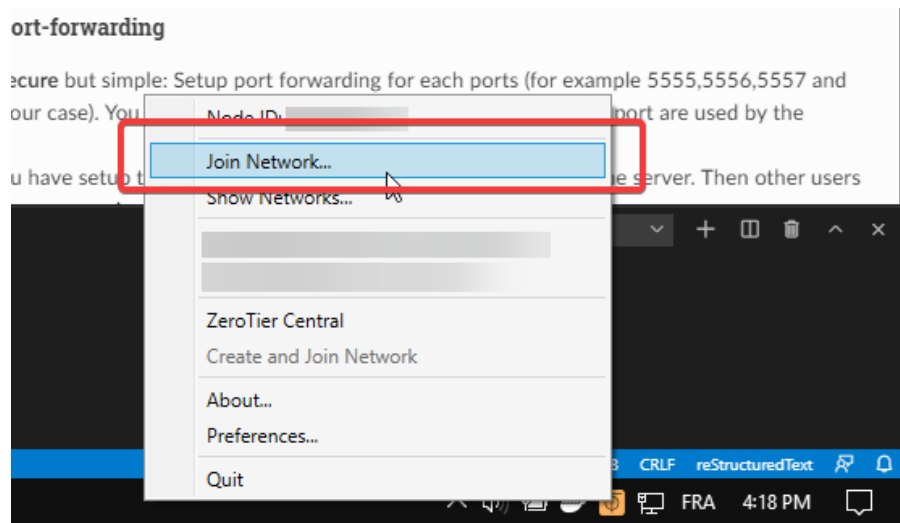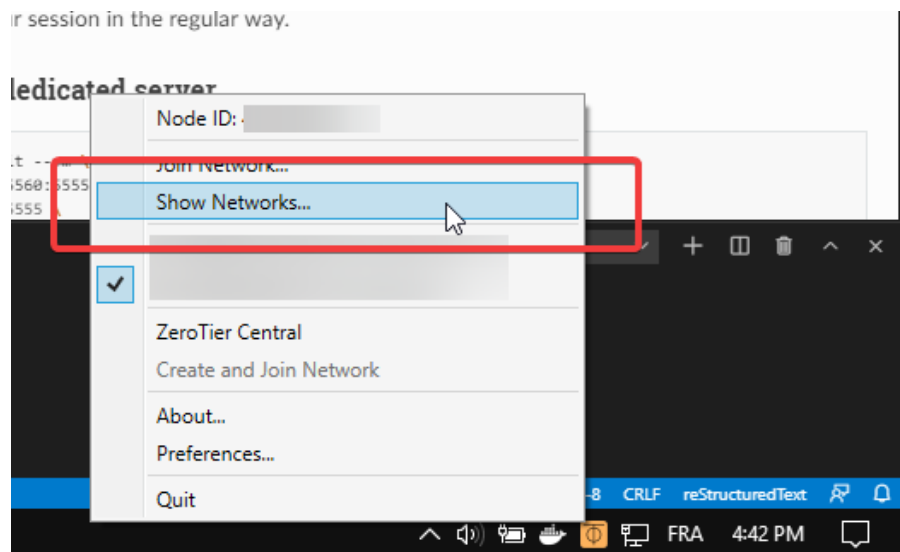Fig. 33: Add the client to network-authorized users

Fig. 34: Joining the network
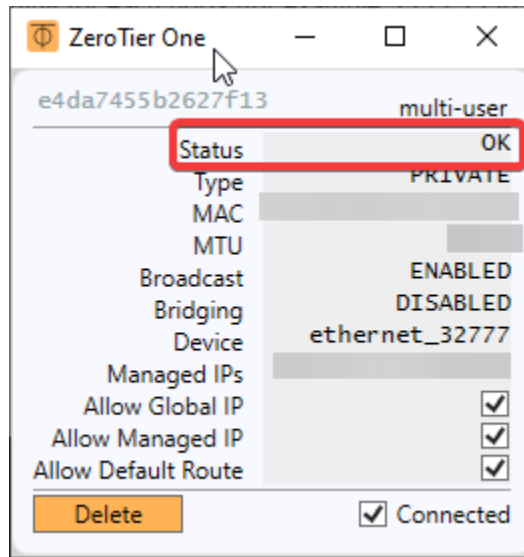

Fig. 35: Show network status

Fig. 36: Network status.

> **Warning:** The dedicated server is developed to run directly on an internet server (like a VPS (Virtual Private Server)). You can also run it at home on a LAN but for internet hosting you need to follow the *Using port-forwarding* setup first. Please see *Cloud Hosting Walkthrough* for a detailed walkthrough of cloud hosting using Google Cloud.

The dedicated server allows you to host a session with simplicity from any location. It was developed to improve internet hosting performance (for example poor latency).

The dedicated server can be run in two ways:

- *Using a regular command line*
- *Using a pre-configured image on docker engine*

> **Note:** There are shell scripts to conveniently start a dedicated server via either of these approaches available in the gitlab repository. See section: *Server startup scripts*

Using a regular command line

You can run the dedicated server on any platform by following these steps:

1. Firstly, download and intall python 3 (3.6 or above).

2. Install the latest version of the replication library:

```
python -m pip install replication
```

3. Launch the server with:

```
replication.serve
```

Hint: You can also specify a custom **port** (-p), **timeout** (-t), **admin password** (-pwd), **log level (ERROR, WARN-ING, INFO or DEBUG)** (-l) and **log file** (-lf) with the following optional arguments

```
replication.serve -p 5555 -pwd admin -t 5000 -l INFO -lf server.log
```

Here, for example, a server is instantiated on port 5555, with password 'admin', a 5 second timeout, and logging enabled.

As soon as the dedicated server is running, you can connect to it from blender by following *How to join a session*.

Hint: Some server commands are available to enable administrators to manage a multi-user session. Check *Dedicated server management* to learn more.

Using a pre-configured image on docker engine

Launching the dedicated server from a docker server is simple as running:

```
docker run -d \
    -p 5555-5560:5555-5560 \
    -e port=5555 \
    -e log_level=DEBUG \
    -e password=admin \
    -e timeout=5000 \
    registry.gitlab.com/slumber/multi-user/multi-user-server:latest
```

Please use the :latest tag, or otherwise use the URL of the most recent container available in the multi-user container registry. As soon as the dedicated server is running, you can connect to it from blender by following *How to join a session*.

You can check that your container is running, and find its ID and name with:

```
docker ps
```

Viewing logs in a docker container

Logs for the server running in a docker container can be accessed by outputting the container logs to a log file. First, you'll need to know your container ID, which you can find by running:

```
docker ps
```

Then, output the container logs to a file:

```
docker logs your-container-id >& dockerserver.log
```

Note: If using WSL2 on Windows 10 (Windows Subsystem for Linux), it is preferable to run a dedicated server via regular command line approach (or the associated startup script) from within Windows - docker desktop for windows 10 usually uses the WSL2 backend where it is available.

Downloading logs from a docker container on a cloud-hosted server

If you'd like to pull the log files from a cloud-hosted server to submit to a developer for review, a simple process using SSH and SCP is as follows:

First SSH into your instance. You can either open the VM Instances console and use the browser terminal provided by Google Cloud (I had the best luck using the Google Chrome browser)... or you can see here for how to set up your instance for SSH access from your local terminal.

If using SSH from your terminal, first generate SSH keys (setting their access permissions to e.g. chmod 400 level whereby only the user has permissions) and submit the public key to the cloud-hosted VM instance, storing the private key on your local machine. Then, SSH into your cloud server from your local terminal, with the following command:

```
ssh -i PATH_TO_PRIVATE_KEY USERNAME@EXTERNAL_IP_ADDRESS
```

Use the private key which corresponds to the public key you uploaded, and the username associated with that key (visible in the Google Cloud console for your VM Instance). Use the external IP address for the server, available from the VM Instances console e.g.

```
ssh -i ~/.ssh/id_rsa user@xxx.xxx.xxx.xxx
```

Once you've connected to the server's secure shell, you can generate a log file from the docker container running the replication server. First, you'll need to know your container ID, which you can find by running:

```
docker ps
```

If you're cloud-hosting with e.g. Google Cloud, your container will be the one associated with the registry address where your Docker image was located. e.g. registry.gitlab.com/slumber/multi-user/multi-user-server:latest

To view the docker container logs, run:

```
docker logs your-container-name
```

OR

```
docker logs your-container-id
```

To save the output to a file, run:

```
docker logs your-container-id >& dockerserver.log
```

Now that the server logs are available in a file, we can disconnect from the secure shell (SSH), and then copy the file to the local machine using SCP. In your local terminal, execute the following:

```
scp -i PATH_TO_PRIVATE_KEY USERNAME@EXTERNAL_IP_ADDRESS:"dockerserver.log" LOCAL_PATH_
↪TO_COPY_FILE_TO
```

e.g.

```
scp -i ~/.ssh/id_rsa user@xxx.xxx.xxx.xxx:"dockerserver.log" .
```

This copies the file dockerserver.log generated in the previous step to the current directory on the local machine. From there, you can send it to the multi-user maintainers for review.

Note: See these notes for how to check server logs on Google Cloud using other tools.

Server startup scripts

Convenient scripts are available in the Gitlab repository: https://gitlab.com/slumber/multi-user/scripts/startup_scripts/

Simply run the relevant script in a shell on the host machine to start a server with one line of code via replication directly or via a docker container. Choose between the two methods:

```
./start-server.sh
```

or

```
./run-dockerfile.sh
```

Hint: Once your server is up and running, some commands are available to manage the session *Dedicated server management*

Dedicated server management

Here is the list of available commands from the dedicated server:

- `help` or `?`: Show all commands. Or, use `help <command>` to learn about another command
- `exit` or `Ctrl+C` : Stop the server.
- `kick username`: kick the provided user.
- `users`: list all online users.

Also, see *How to manage a session* for more details on managing a server.

Managing a docker server from the command line

If you want to be able to manage a server running within a docker container, open the terminal on the host machine (or SSH in, if you are using cloud hosting), and then run

```
docker ps
```

to find your container id, and then

```
docker attach your-container-id
```

to attach to the STDOUT from the container. There, you can issue the server management commands detailed in *Dedicated server management*. Type `?` and hit return/enter to see the available commands. Also, see *How to manage a session* for more details on managing a server.

2.5.3 Port setup

The multi-user network architecture is based on a client-server model. The communication protocol uses four ports to communicate with clients:

- Commands: command transmission (such as **snapshots**, **change_rights**, etc.) [user-nominated port]
- Subscriber : pull data [Commands port + 1]
- Publisher : push data [Commands port + 2]

- TTL (time to leave) : used to ping each client [Commands port + 3]

To know which ports will be used, you just have to read the port in your preferences.
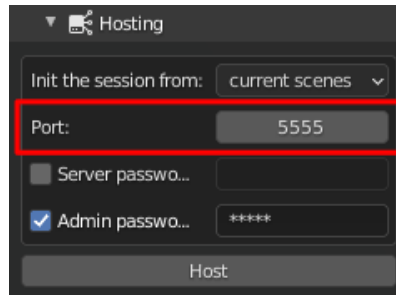

Fig. 37: Port in host settings

In the picture below we have setup our port to **5555** so the four ports will be:

- Commands: **5555** (5555)

- Subscriber: **5556** (5555 +1)

- Publisher: **5557** (5555 +2)

- TTL: **5558** (5555 +3)

Those four ports need to be accessible from the client otherwise multi-user won't work at all !

2.5.4 Cloud Hosting Walkthrough

The following is a walkthrough for how to set up a multi-user dedicated server instance on a cloud hosting provider - in this case, Google Cloud. Google Cloud is a powerful hosting service with a worldwide network of servers. It offers a free trial which provides free cloud hosting for 90 days, and then a free tier which runs indefinitely thereafter, so long as you stay within the usage limits. ^^Thanks to community member @NotFood for the tip!

Cloud hosting is a little more complicated to set up, but it can be valuable if you are trying to host a session with multiple friends scattered about planet earth. This can resolve issues with data replication or slowdowns due to poor latency of some users (high ping). This guide may seem technical, but if you follow the steps, you should be able to succeed in hosting an internet server to co-create with other multi-user creators around the world.
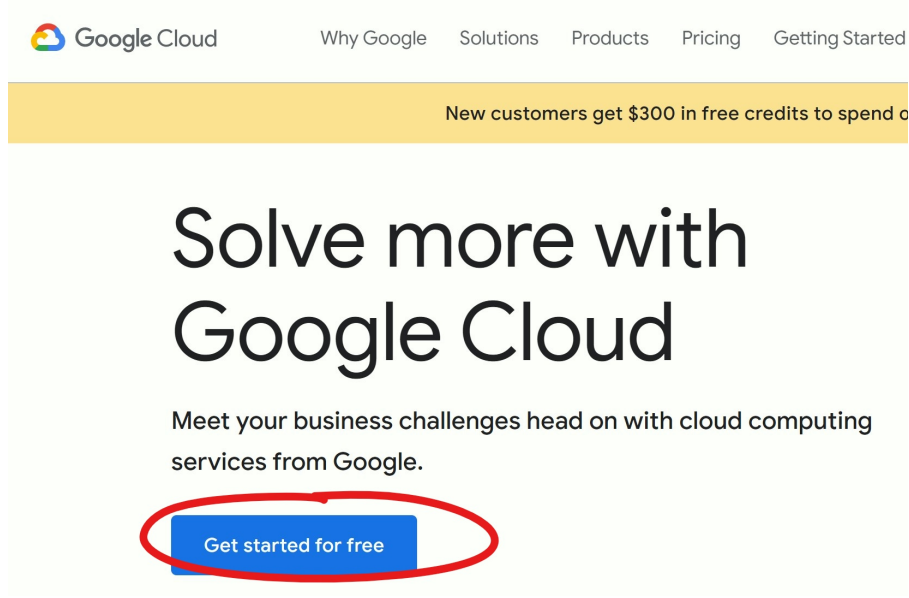
Setup Process

1. Sign Up for Google Cloud

Let's start by activating an account with Google Cloud. Go to https://www.cloud.google.com and click 'Get Started For Free'

Google will ask you to login/signup, and to set up a billing account (Don't worry. It will not be charged unless you explicitly enable billing and then run over your free credit allowance). You will need to choose a billing country (relevant for tax purposes). You will choose your server location at a later step.

2. Enable Billing and Compute Engine API

From here on, we will mostly stick to the instructions provided here. Nevertheless, the instructions for multi-user specifically are as follows.

In order to set up a Virtual Machine (VM) to host your server, you will need to enable the billing account which was created during your signup process. From your console, click on 'Go to Checklist' and then 'Create a Billing Account', following the prompts to choose the billing account that was created for you upon signup.
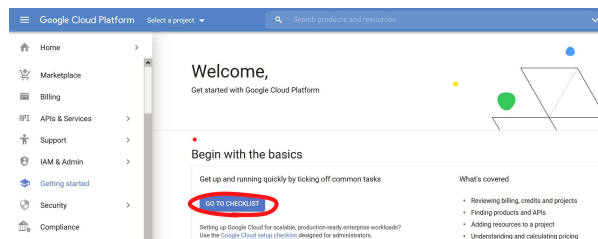
Now hit 'Set Account', and go back to your console.

Now enable the Compute Engine API. Click here to enable.

3. Create a Linux Virtual Machine Instance

Continue following the instructions to create a VM instance. However, once you've finished step 2 of 'Create a virtual machine instance', use the settings and steps for multi-user as follows.

3.1 Choose a Server Location

The most important settings which you will need to choose for your specific case are the server Region and Zone. You must choose a location which will provide the best ping for all of your fellow creators.

All you need to know is that you'll probably want to choose a location near to where most of your collaborators are located. If your friends are spread out, somewhere in the middle which distributes the ping evenly to all users is best.

You can use this map to make a rough guess of the best server location, if you know your friends' locations.

A much better approach is to have your users run a ping test for Google Cloud's servers at https://www.gcping.com/

←

Starter checklist

Get up and running quickly by ticking off common tasks

- **Create a billing account**   GET STARTED
  Track any free trial credits that you may have, and any charges
  accrued if you've upgraded your account

- **Create a project**   GET STARTED
  Use this project to organise your resources as you get started

- **Find products and APIs**   GET STARTED
  Discover popular solutions or explore all Google Cloud
  Platform products and services

- **Start working on your project**   GET STARTED
  Add resources for your cloud infrastructure

- **Understand and calculate pricing**   GET STARTED
  Estimate usage costs for various architectures and
  requirements

You can come back to this checklist at any time.

CLOSE

✕   **Create a billing account**

A billing account was created for you during signup.

About billing accounts

Each project is associated with a single billing account. Multiple projects can have their
resource usage billed to the same account.

GO TO BILLING

Billing
This project has no billing account

This project is not linked to a billing account

LINK A BILLING ACCOUNT   MANAGE BILLING ACCOUNTS
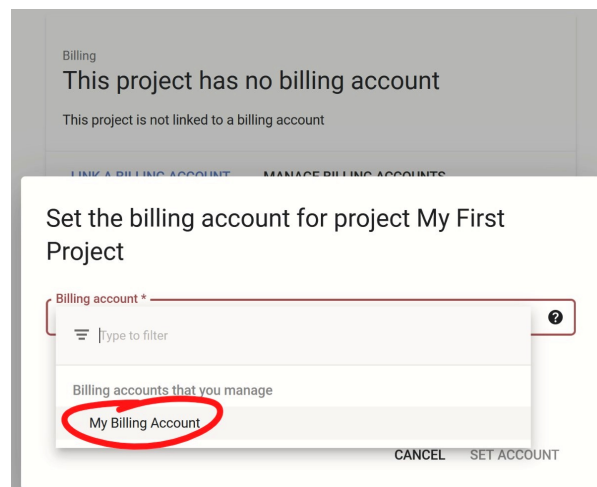
Set the billing account for project My First Project

Billing account *

≡ Type to filter

Billing accounts that you manage

My Billing Account

CANCEL   SET ACCOUNT
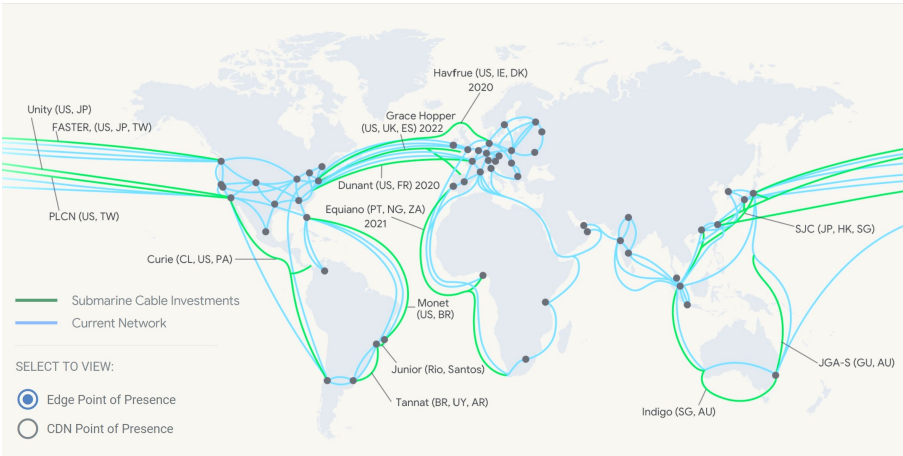
Have your collaborators open this webpage from their fastest browser, and press the play button. The play button turns to a stop icon while the ping test is running. When it is complete, the play button returns. You may need to refresh your browser to get this to work. You can replay the test to add more server locations to the scan, and stop when you are satisfied that the results are consistent.

Now, gather your friends' data, and work down each user's list from the top, until you find the first location which gives roughly the same ping for all users.

In general, global (using load balancing) will provide the best results, but beyond that, the US Central servers e.g. IOWA generally turn out best for a globally distributed bunch of creators. When in doubt, choose between the servers offered under the free tier

- Oregon: *us-west1*

- Iowa: *us-central1*

- South Carolina: *us-east1*

For the following example, the server which gave the most balanced, and lowest average ping between two friends based in Europe and Australia was in Iowa. Salt Lake City would also be an excellent choice.

Fig. 38: Left - European User | Right - Australian User

Now, input this server location in the 'Region' field for your instance, and leave the default zone which is then

populated.

Note: You can read here for a deeper understanding about how to choose a good server location.

3.2 Configure the VM

You can deploy the replication server to your VM in either of the ways mentioned at *From the dedicated server*. That is, you can set it up *Using a regular command line* or *Using a pre-configured image on docker engine*. We will go through both options in this walkthrough. See *Should I deploy a Docker Container or launch a server from Linux VM command-line?* for more details on how to choose. Deploying a container is the recommended approach.

Option 1 - Deploy a container

If you are familiar with Docker, you'll appreciate that it makes life a little simpler for us. While configuring your instance, you can check **Deploy a container to this VM instance** and copy in the URL of the latest docker image available from the multi-user container registry to the *Container image* field, or use the tag `:latest`

Make sure to choose the amount of memory you'd like your server to be able to handle (how much memory does your blender scene require?). In this example, I've chosen 4GB of RAM.

Click on **Advanced container options** and turn on *Allocate a buffer for STDIN* and *Allocate a pseudo-TTY* just in case you want to run an interactive shell in your container.

Optional server parameters

The default Docker image essentially runs the equivalent of:

```
replication.server -pwd admin -p 5555 -t 5000 -l DEBUG -lf multiuser_server.
→log
```

This means the server will be launched with 'admin' as the administrator password, run on ports 5555:5558, use a timeout of 5 seconds, verbose 'DEBUG' log level, and with log files written to 'multiuser_server.log'. See *Using a regular command line* for a description of optional parameters.

Note: If you'd like to configure different server options from the default docker configuration, you can insert your options here by expanding 'Advanced container options'

For example, I would like to launch my server with a different administrator password than the default, my own log filename, and a shorter 3-second (3000ms) timeout. I'll click *Add argument* under **Command arguments** and paste the following command with options into the "command arguments" field:

```
replication.serve -pwd supersecretpassword -p 5555 -t 3000 -l DEBUG -lf
→logname.log
```

Now, my configuration should look like this:

The rest of the settings are now complete. Hit **Create** and your instance will go live. If you've taken this approach, you're already almost there! Skip to *4. Setting up Firewall and opening Ports*.

Name ❓
Name is permanent

instance-multiuser-docker

Labels ❓ (Optional)

➕ Add label

Region ❓   **Zone** ❓
Region is permanent   Zone is permanent

us-central1 (Iowa)   ⯆   us-central1-a   ⯆

Machine configuration

Machine family

| General-purpose | Compute-optimised | Memory-optimised |

Machine types for common workloads, optimised for cost and flexibility

Series

E2   ⯆

CPU platform selection based on availability

Machine type

e2-medium (2 vCPU, 4 GB memory)   ⯆

	vCPU	Memory	GPUs
	1 shared core	4 GB	-

⯅ CPU platform and GPU

Confidential VM service ❓
☐ Enable the Confidential Computing service on this VM instance.

Container ❓
☑ Deploy a container image to this VM instance. Learn more

Container image ❓

registry.gitlab.com/slumber/multi-user/multi-user-server:latest

Restart policy ❓

Always   ⯆

☐ Run as privileged ❓

☑ Allocate a buffer for STDIN ❓

☑ Allocate a pseudo-TTY ❓

Command ❓

Command arguments ❓

➕ Add argument

Environment variables ❓

Name ❓
Name is permanent

instance-multiuser-docker

Labels ❓ (Optional)

➕ Add label

Region ❓
Region is permanent

us-central1 (Iowa) ▼

Zone ❓
Zone is permanent

us-central1-a ▼

Machine configuration

Machine family

| General-purpose | Compute-optimised | Memory-optimised |

Machine types for common workloads, optimised for cost and flexibility

Series

E2 ▼

CPU platform selection based on availability

Machine type

e2-medium (2 vCPU, 4 GB memory) ▼

	vCPU	Memory	GPUs
	1 shared core	4 GB	-

⌄ CPU platform and GPU

Confidential VM service ❓
☐ Enable the Confidential Computing service on this VM instance.

Container ❓
☑ Deploy a container image to this VM instance. Learn more

Container image ❓

registry.gitlab.com/slumber/multi-user/multi-user-server:latest

Restart policy ❓

Always ▼

☐ Run as privileged ❓

☑ Allocate a buffer for STDIN ❓

☑ Allocate a pseudo-TTY ❓

Command ❓

Command arguments ❓

python3 -m replication.server -pwd supersecretpassword -p 5555 -t 3000 - ✕

➕ Add argument

Hint: You can find further information on configuration options here. Also, see these notes for other options when deploying your server inside a container, including how to access the server's logs.

Option 2 - Over SSH

Otherwise, we can run the dedicated server ourselves from the command-line over SSH.

While creating your instance, keep the default settings mentioned in the guide, however at step 4, choose Debian version 10. Also, there is no need to enable HTTP so skip step 6.

Make sure to choose the amount of memory you'd like your server to be able to handle (how much memory does your blender scene require?). In this example, I've chosen 4GB of RAM.

Now, finally, click 'Create' to generate your Virtual Machine Instance.

4. Setting up Firewall and opening Ports

Now that your VM is instanced, you'll need to set up firewall rules, and open the ports required by multi-user. The documentation for VM firewalls on google cloud is here.

First, go to the dashboard showing your VM instances and note the 'External IP' address for later. This is the address of your server. Then, click 'Set up Firewall Rules'.

Now you will need to create two rules. One to enable communication inbound to your server (ingress), and another to enable outbound communication from your server (egress). Click 'Create Firewall'

Now create a rule exactly as in the image below for the outbound communication (egress).

Note: If you set a different port number in *Optional server parameters*, then use the ports indicated in *Port setup*

And another rule exactly as in the image below for the inbound communication (ingress).

Finally, your firewall configuration should look like this.

5. Install Replication Server into Virtual Machine

Note: Skip to *7. Initialise your Server in Blender* if you've opted to launch the server by deploying a container. Your server is already live!

Now that we have set up our Virtual Machine instance, we can SSH into it, and install the Replication Server. Open the VM Instances console once more, and SSH into your instance. It's easiest to use the browser terminal provided by Google Cloud (I had the best luck using the Google Chrome browser), but you can also see here for how to set up your instance for SSH access from your terminal.

Now, a terminal window should pop up in a new browser window looking something like this:

Remember, you had set up the VM with Debian 10. This comes with Python 3.7.3 already installed. The only dependency missing is to set up pip3. So, run:

```
sudo apt install python3-pip
```

And now lets install the latest version of replication:

Name ❓
Name is permanent

> instance-1

Labels ❓ (Optional)

> **+ Add label**

Region ❓
Region is permanent

Zone ❓
Zone is permanent

> us-central1 (Iowa) ▼

> us-central1-a ▼

Machine configuration

> **Machine family**
>
> | General-purpose | Compute-optimised | Memory-optimised |
>
> Machine types for common workloads, optimised for cost and flexibility
>
> **Series**
>
> > E2 ▼
>
> CPU platform selection based on availability
>
> **Machine type**
>
> > e2-medium (2 vCPU, 4 GB memory) ▼
>
> | | vCPU | Memory | GPUs |
> |---|---|---|---|
> | | 1 shared core | 4 GB | - |

⌄ CPU platform and GPU

Confidential VM service ❓
☐ Enable the Confidential Computing service on this VM instance.

Container ❓
☐ Deploy a container image to this VM instance. Learn more

Boot disk ❓

> New 10 GB standard persistent disk
>
> Image
>
> 🛡 Debian GNU/Linux 10 (buster) | Change |

Identity and API access ❓

> **Service account** ❓
>
> > Default compute service account ▼
>
> **Access scopes** ❓
> ● Allow default access
> ○ Allow full access to all Cloud APIs
> ○ Set access for each API

Firewall ❓
Add tags and firewall rules to allow specific network traffic from the Internet.
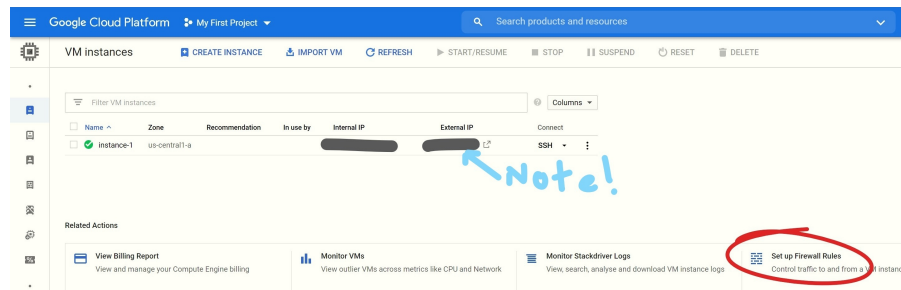
Fig. 41: Note down your External IP
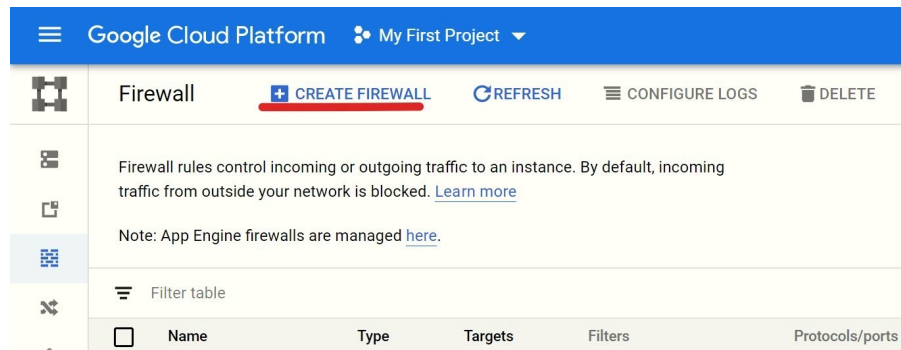

```
sudo pip3 install replication==0.1.13
```

6. Launch Replication Server on VM Instance

We're finally ready to launch the server. Simply run:

```
replication.serve -p 5555 -pwd admin -t 5000 -l INFO -lf server.log
```

See *Using a regular command line* for a description of optional parameters

And your replication server is live! It should stay running in the terminal window until you close it. Copy the external IP that you noted down earlier, available here and now you can open Blender and connect to your server!

7. Initialise your Server in Blender

Once in Blender, make sure your multi-user addon is updated to the latest version. update-version. Then, follow the instructions from *How to join a session* and connect as an admin user, using the password you launched the server with and the IP adress of the server. Then, click *connect*.

Now as the admin user, you can click on *init* and choose whether to initialise the server with a preloaded scene, or an empty scene.

Now your session is live!

If you made it this far, congratulations! You can now go ahead and share the external IP address with your friends and co-creators and have fun with real-time collaboration in Blender!

Hopefully, your cloud server setup has improved your group's overall ping readings, and you're in for a smooth and trouble-free co-creation session.

← Create a firewall rule

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked.Learn more

Name *
multiuser-allow-out

Lowercase letters, numbers, hyphens allowed

Description
Enable multiuser egress on 5555, 5556, 5557, 5558

Logs

Turning on firewall logs can generate a large number of logs; this can increase costs in Stackdriver.Learn more
○ On
◉ Off

Network *
default ▼ ❓

Priority *
1000 ❓

Priority can be 0–65535Check priority of other firewall rules

Direction of traffic ❓
○ Ingress
◉ Egress

Action on match ❓
◉ Allow
○ Deny

Targets
All instances in the network ▼ ❓

Destination filter
IP ranges ▼ ❓

Destination IP ranges *
0.0.0.0/0 ✕ for example, 0.0.0.0/0, 192.168.2.0/24 ❓

Protocols and ports ❓
○ Allow all
◉ Specified protocols and ports

☑ tcp : 5555,5556,5557,5558

☐ udp : all

☐ Other protocols
protocols, comma separated, e.g. ah, sctp

∨ DISABLE RULE

CREATE   CANCEL

←   Firewall rule details   ✏ EDIT   🗑 DELETE

multiuser-allow-in

Description
Enable multiuser ingress on 5555, 5556, 5557, 5558

Logs

Turning on firewall logs can generate a large number of logs; this can increase costs in Stackdriver. Learn more

○ On

◉ Off

Network
default

Priority *
1000   ❓

Priority can be 0–65535 Check priority of other firewall rules

Direction
Ingress

Action on match
Allow

Targets
All instances in the network ▾

Source filter
IP ranges ▾ ❓

Source IP ranges *
0.0.0.0/0 ⊗   for example, 0.0.0.0/0, 192.168.2.0/24   ❓

Second source filter
None ▾ ❓

Protocols and ports ❓

○ Allow all

◉ Specified protocols and ports

    ☑ tcp :   5555,5556,5557,5558

    ☐ udp :   all

    ☐ Other protocols

    protocols, comma separated, e.g. ah, sctp

⌄ DISABLE RULE

SAVE   CANCEL

Equivalent REST

Fig. 44: Final Firewall Configuration

Note: If you should so desire, pay attention to your credit and follow the steps here to close your instance at your discretion.

Should I deploy a Docker Container or launch a server from Linux VM command-line?

- Directly from Linux VM - This approach gives you control over your session more easily. However, your server may time out once your SSH link to the server is interrupted (for example, if the admin's computer goes to sleep).

- Deploy a Docker Container - This is the recommended approach. This approach is better for leaving a session running without supervision. It can however be more complicated to manage. Use this approach if you'd like a consistent experience with others in the multi-user community, pulling from the most up-to-date docker image maintained by @swann in the multi-user container registry.

2.6 Troubleshooting

The majority of issues new users experience when first using Multi-User can be solved with a few quick checks.

- Update the multi-user addon to the latest version

- Make sure to allow Blender through your firewall

> **Hint:** Your firewall may have additional settings like Ransomware protection, or you may need to enable both Blender and Python on private and/or public Networks

- Solve problems with your connection quality

- Minimise the use of large textures or file sizes

Use the #support channel on the multi-user discord server to chat, seek help and contribute.

2.7 Ways to contribute

Note: Work in progress

2.7.1 Testing and reporting issues

A great way of contributing to the multi-user addon is to test development branch and to report issues. It is also helpful to report issues discovered in releases, so that they can be fixed in the development branch and in future releases.

Testing development versions

In order to help with the testing, you have several possibilities:

- Test latest release
- Test development branch

Filing an issue on Gitlab

The gitlab issue tracker is used for bug report and enhancement suggestion. You will need a Gitlab account to be able to open a new issue there and click on "New issue" button in the main multi-user project.

Here are some useful information you should provide in a bug report:

- **Multi-user version** such as *lastest*, *commit-hash*, *branch*. This is a must have. Some issues might be relevant in the current stable release, but fixed in the development branch.

- **How to reproduce the bug**. In the majority of cases, bugs are reproducible, i.e. it is possible to trigger them reliably by following some steps. Please always describe those steps as clearly as possible, so that everyone can try to reproduce the issue and confirm it. It could also take the form of a screen capture.

2.7.2 Contributing code

In general, this project follows the Gitflow Workflow. It may help to understand that there are three different repositories - the upstream (main multi-user project repository, designated in git by 'upstream'), remote (forked repository, designated in git by 'origin'), and the local repository on your machine. The following example suggests how to contribute a feature.

1. **Fork the project into a new repository:** https://gitlab.com/yourname/multi-user

2. **Clone the new repository locally:**

   ```
   git clone https://gitlab.com/yourname/multi-user.git
   ```

3. **Keep your fork in sync with the main repository by setting up the upstream pointer once. cd into your git repo and then ru**

```
git remote add upstream https://gitlab.com/slumber/multi-user.git
```

4. **Now, locally check out the develop branch, upon which to base your new feature branch:**

```
git checkout develop
```

5. **Fetch any changes from the main upstream repository into your fork (especially if some time has passed since forking):**

```
git fetch upstream
```

'Fetch' downloads objects and refs from the repository, but doesn't apply them to the branch we are working on. We want to apply the updates to the branch we will work from, which we checked out in step 4.

6. **Let's merge any recent changes from the remote upstream (original repository's) 'develop' branch into our local 'develop'**

```
git merge upstream/develop
```

7. **Update your forked repository's remote 'develop' branch with the fetched changes, just to keep things tidy. Make sure yo**

```
git push origin develop
```

8. **Locally create your own new feature branch from the develop branch, using the syntax:**

```
git checkout -b feature/yourfeaturename
```

. . . where 'feature/' designates a feature branch, and 'yourfeaturename' is a name of your choosing

9. **Add and commit your changes, including a commit message:**

```
git commit -am 'Add fooBar'
```

10. Push committed changes to the remote copy of your new feature branch which will be created in this step:

```
git push -u origin feature/yourfeaturename
```

If it's been some time since performing steps 4 through 7, make sure to checkout 'develop' again and pull the latest changes from upstream before checking out and creating feature/yourfeaturename and pushing changes. Alternatively, checkout 'feature/yourfeaturename' and simply run:

```
git rebase upstream/develop
```

and your staged commits will be merged along with the changes. More information on rebasing here

---

Hint: -u option sets up your locally created new branch to follow a remote branch which is now created with the same name on your remote repository.

---

11. Finally, create a new Pull/Merge Request on Gitlab to merge the remote version of this new branch with committed updates, back into the upstream 'develop' branch, finalising the integration of the new feature. Make sure to set the target branch to 'develop' for features and 'master' for hotfixes. Also, include any milestones or labels, and assignees that may be relevant. By default, the Merge option to 'delete source branch when merge request is activated' will be checked.

---

12. Thanks for contributing!

Note: For hotfixes, replace 'feature/' with 'hotfix/' and base the new branch off the parent 'master' branch instead of 'develop' branch. Make sure to checkout 'master' before running step 8

Note: Let's follow the Atlassian Gitflow Workflow, except for one main difference - submitting a pull request rather than merging by ourselves.

Note: See here or here for instructions on how to keep a fork up to date.