

# Install and Run Open Drone Map (ODM) on cPouta

Step-by-step guide  
(Volume1)

2019

## Open Geospatial Information Infrastructure for Research (oGIIR)



Consortium:



Compiled and edited by:

Augustine-Moses Gbagir (MSc.)<sup>1</sup>

Alfred Colpaert (Prof.)<sup>2</sup>

<sup>1,2</sup>Department of Geographical and  
Historical Studies, University of Eastern  
Finland.

# Contents

CHAPTER 1: SETTING UP A VIRTUAL MACHINE ON CPOUTA.....	3
cPouta virtual infrastructure.....	3
The basic skills you need to use cPouta.....	3
Basic software you need .....	3
How to set up your virtual machine (vm) on cPouta .....	4
Step 1 Create keypair for your virtual machine(vm) .....	4
Step 1.2 Modify, add password and authenticate keypair.....	6
Step 1.3 Create security group for your virtual machine (vm) .....	8
Step 1.4 Add security rules for your vm .....	9
Step 1.5 Creating our virtual machine (vm).....	11
Step 1.6 Launch your virtual machine .....	15
Step 2 Connecting to your vm.....	15
Step 2.1 To assign a public IP address:.....	15
Step 2.2 Connecting to your vm.....	16
Step 3 Create, attach and mount a volume to your VM (virtual machine) in cPouta .....	18
Step 3.1 Create a new volume.....	19
Step 3.2 Attach volume to new vm .....	20
Step 3.3 Detach volume from vm .....	20
Step 3.4 Mount new volume to vm. ....	21
CHAPTER 2: INSTALL DOCKER AND OPEN DRONE MAP(ODM) ON CPOUTA .....	24
1. Installing Docker.....	24
Step 1.0 Pre-check uninstall old docker and install new docker version.....	24
Step 1.2: Update the apt package index:.....	24
Step 1.3: Install packages to allow apt to use a repository over HTTPS: .....	25
Step 1.4: Add Docker's official GPG key and verify the key fingerprint. ....	25
Step 1.5: Set up a stable docker repository .....	26
Step 1.6: Install the <i>latest version</i> of Docker CE and containerd .....	26
Step 1.7: Verify that Docker is installed correctly .....	28
Step 1.8: Updating Docker CE.....	28
2.0 Install Open Drone Map (ODM).....	29
2.1 Build ODM docker image.....	29
2.2 Check the built docker images .....	29
3.0 Copy Files/Folders to cPouta vm.....	30
3.1 Install WinSCP.....	30
3.2 Authenticate WinSCP .....	30

4.0 Test ODM on cPouta.....	32
4.1 Log in and connect to vm.....	33
Appendix I.....	34
Appendix II .....	35
Appendix III.....	36

NOTE:

This newly released basic user guide is a work in progress and may still contain some rough edges. We are committed to improving the quality of the content of this guide. Future releases will cover aspects that are not included here in volume 1.

# CHAPTER 1: SETTING UP A VIRTUAL MACHINE ON CPOUTA

cPouta virtual infrastructure

## Separation of Responsibilities

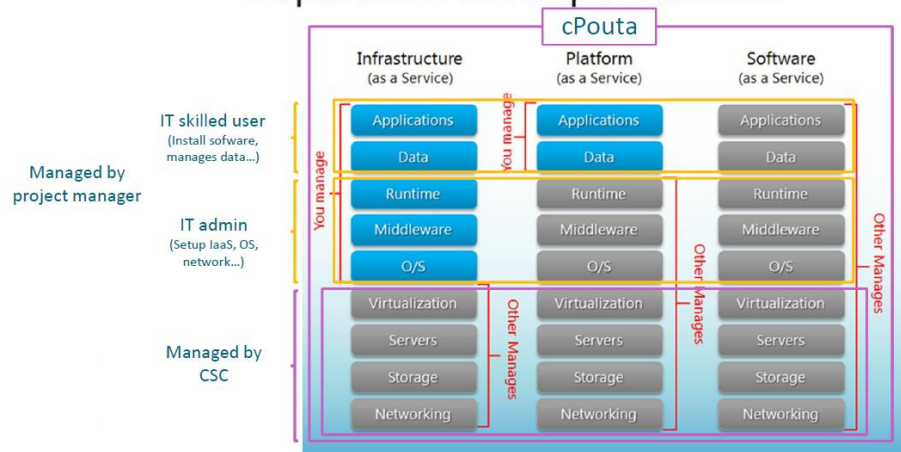


Figure 1 Shows the virtual infrastructure of cPouta [Source: Eduardo Gonzalez (CSC)].

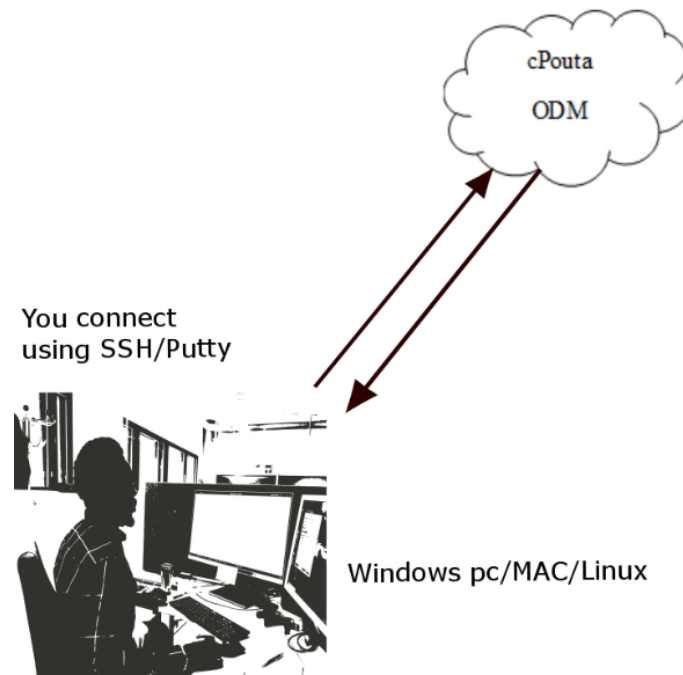


Figure 2 Shows the basic communication to the cPouta infrastructure

The basic skills you need to use cPouta

- (i) Basic knowledge of Linux commands (see table 1, Appendix I).
- (ii) Basic knowledge of CSC cloud architecture.
- (iii) Basic knowledge of UAV imagery.

Basic software you need

- 1) Puttygen
- 2) Putty, and
- 3) WinSCP

## How to set up your virtual machine (vm) on cPouta

Our focus in this section is using the cPouta user interface to set up your vm. We assume you already have a CSC account and access to cPouta. If you don't have an account on cPouta, please use this link (<https://research.csc.fi/accounts-and-projects>) and follow the instructions and get access to cPouta.

Note:

- (a) Throughout this manual, we will use a demo account in our demonstrations.
- (b) When setting up your own vm's please ensure to modify the security settings to protect your vm's and data from hacking and other un-authorized users.

### Step 1 Create keypair for your virtual machine(vm)

The key pair (password) and security set up will allow a specific IP address (that of your pc) to connect to your vm. This is necessary for accessing your vm from anywhere using and SSH connection.

#### Step 1.1 To create a key pair,

- (i) log into your account here <http://pouta.csc.fi>. When you log in, you are connected to a default project (see figure 3 below). If you wish to change the project, follow (ii) below, otherwise proceed to (iii).
- (ii) to change project, click the drop-down button (top left corner, figure 3) and select the project you want. After, this, you can proceed to creating a keypair.

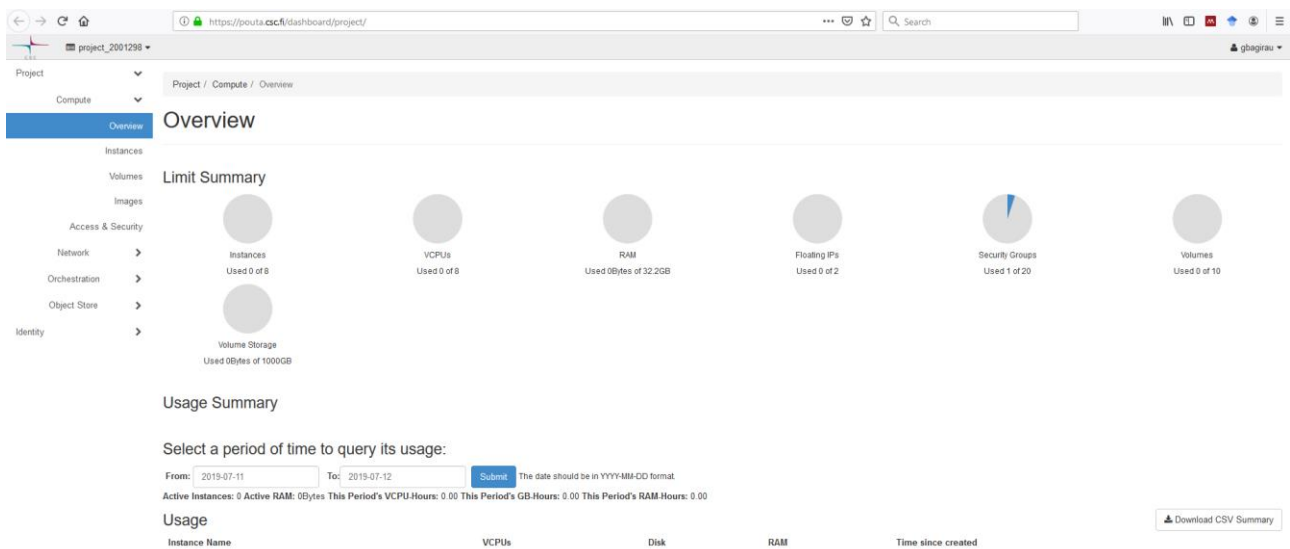


Figure 3 Log in interface of cPouta

- (iii) Click on Access and Security on the left panel (see figure 3). A new page will open (figure 4).

Note: Here you will be able to see all the security groups you create for all your vm's. This page will be blank if this is your time of doing this.

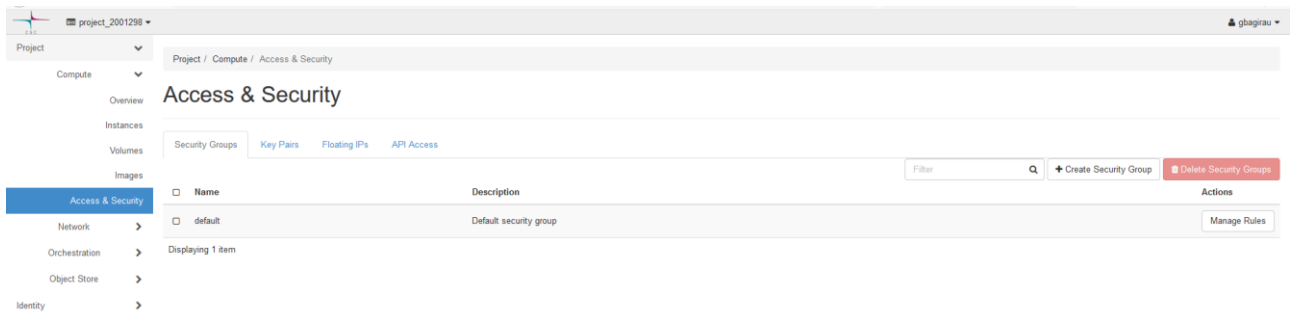


Figure 4 Shows the “Access and Security” settings of the cPouta web interface

- (iv) Next, click on “Key Pairs” displayed at the top of this page to open a pop-up window (figure 5).
- (v) Give you key pair a unique name (e.g. lastname\_key). For this tutorial, we name our key pair as “demo\_key”.
- (vi) Now click on “Create Key Pair” bottom right (figure 5). This action will automatically download the new key pair and prompt you to save it (figure 6). You can always download and re-use this key on any other pc. You will only need to authenticate the use of the key (discussed later) on this new computer.
- (vii) Now save this new key (with a “.pem” extension) to your local computer where you can remember. Make sure it is a secure location with restricted access from other users. For this tutorial, we saved our key as “demo\_key.pem.”

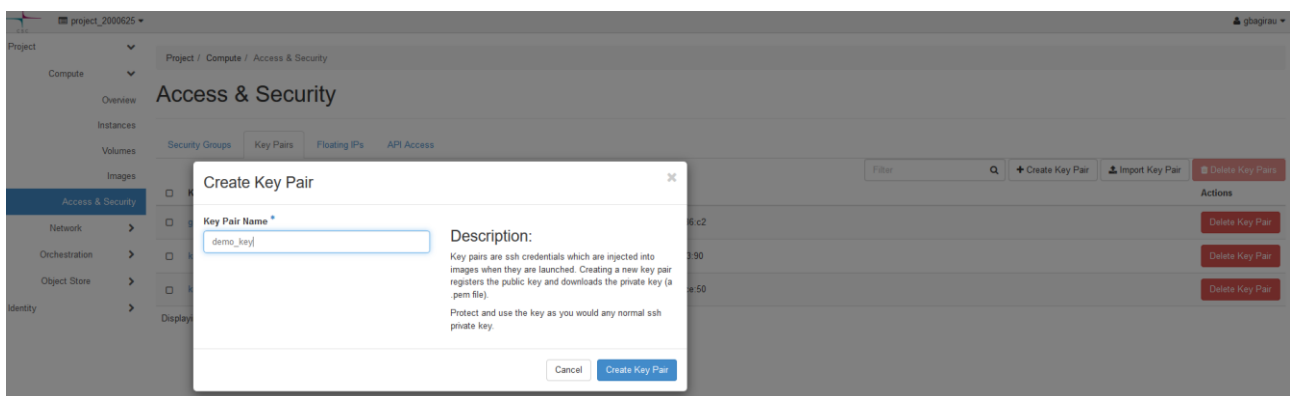


Figure 5 Creating and new key pair cPouta web interface

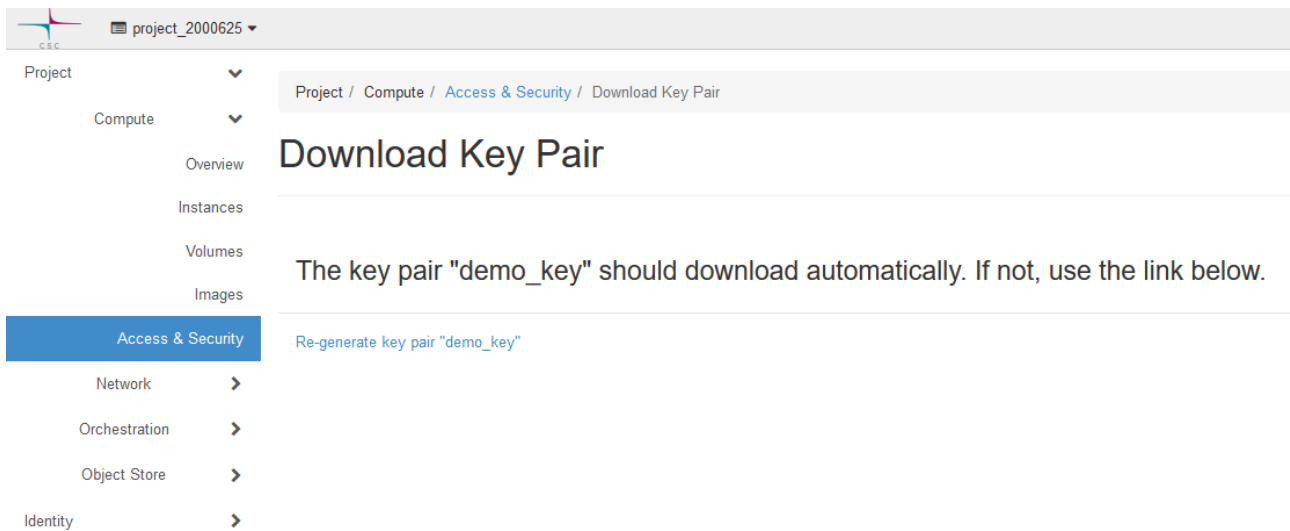


Figure 6 Shows the newly created key pair displayed on cPouta web interface. You can download and re-use this key any time.

### Step 1.2 Modify, add password and authenticate keypair

The keypair we generated in cPouta was Linux-based and Windows will not recognize it unless we convert it to a format readable by Windows. For this purpose, we will use Puttygen to convert and secure it with a password. Download Puttygen from the link below.

(<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>). Later we will use Putty SSH client to connect to our new vm.

Note:

If you are using Linux or Mac OS X, follow the instructions “1–3” below to secure your keypair (source: Eduardo Gonzales CSC).

- 1) Create .ssh directory in ~ (the users home directory) if it is not there already.

```
$ mkdir -p .ssh  
$ chmod 600 .ssh
```

- 2) Move the key into it and make it read-write only.

```
$ mv ../Downloads/lastname_firstname.pem ~/.ssh  
$ chmod 600 lastname_firstname.pem
```

- 3) Add a password to it (recommended).

```
$ ssh-keygen -p -f lastname_firstname.pem
```

- (i) From the Windows start menu (bottom left corner), navigate to Putty and double click on Puttygen to load the program (figure 7).

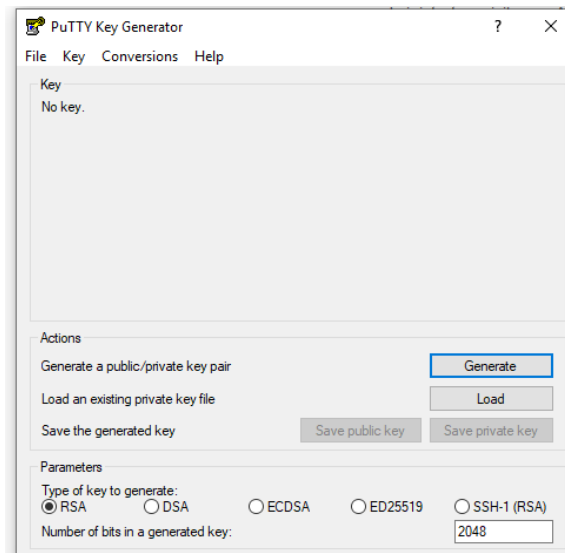


Figure 7 The Puttygen interface.

- (ii) Click on file (top left corner figure 7), navigate to where you save the keypair on your local computer. Remember to select “All Files (\*.\*)” in the file browser to see your “.pem” key file. After loading the key pair, you will have a window similar to figure 8 below. Recall, we saved our key as, “demo\_key.pem.”

Note: PuTTY can also be used to generate keypairs but since we have created already in cPouta, we just load it and use.

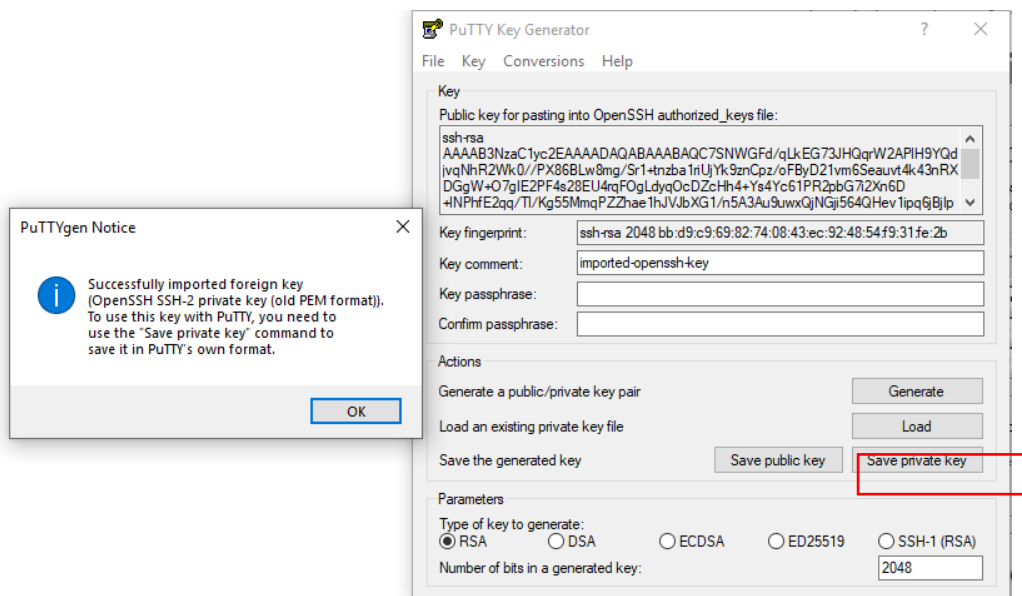


Figure 8 Shows how to save your secured key pair with PuttyGen in Windows.

- (iii) Click “ok” on the PuTTY gen Notice.



- (iv) Next, enter a secure key passphrase (password) and confirm this by re-typing the same passphrase.
- (v) Now, click on “Save private key” (figure 6 red box bottom right) and save it as a “.ppk” file. Save it to the same location as the “.pem” file. Recall that for this tutorial, we saved it as “demo\_key.ppk.”

Note: Later [step 2.2.1 page 16], we will authenticate this key using the Putty SSH client to connect to our vm.

### Step 1.3 Create security group for your virtual machine (vm)

Here, we will create a secure security group that will allow our local computer (IP address) to securely connect to our vm. For this tutorial, we will add rules to allow just a single IP address (your computer) to communicate with our vm. This will prevent snoopers (add snoop image) to access your vm.

Note:

You can add as many rules you want to allow many vm’s to use a single security group.

- (i) Go to cPouta web interface, click on Access and Security > Security Groups > Create Security Group (top right) (figure 9).

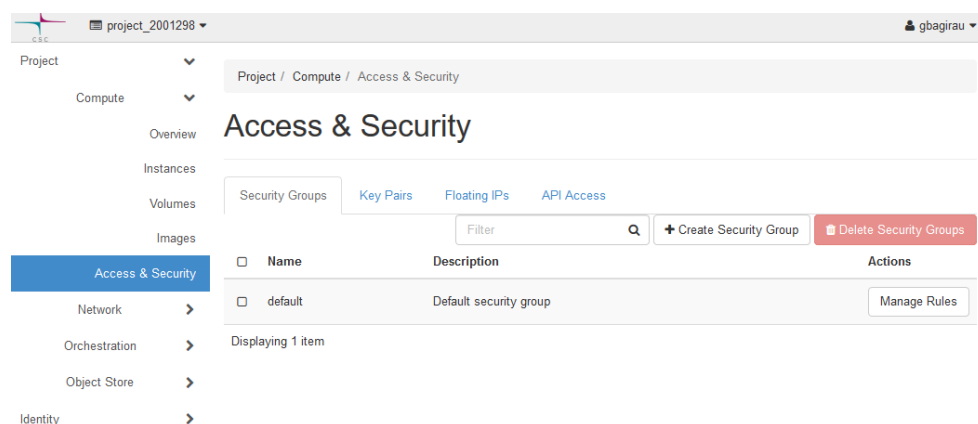


Figure 9 Shows how to create new security group for your virtual machine

Give your security group a name (e.g. lastname\_SSH) and click save (figure 10). For this tutorial, “demo\_SSH”.

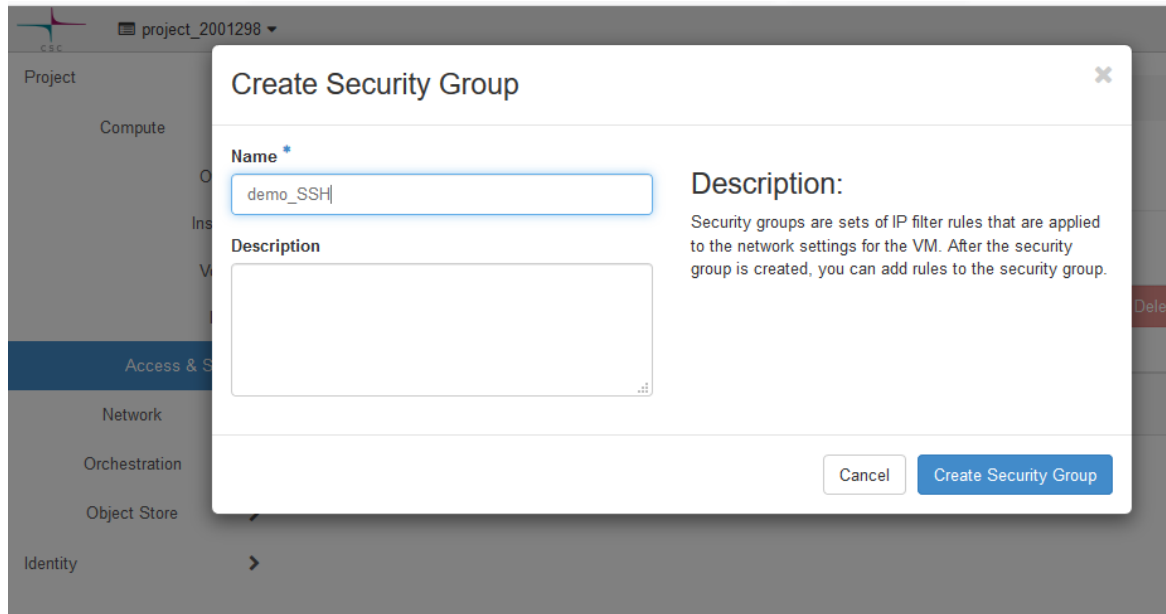


Figure 10 Shows how to create and save new security group for your virtual machine (vm).

#### Step 1.4 Add security rules for your vm

Before you can add security rules, first identify the IP address of your local computer. You will need this information to add to your security group. Click and use either of the links below to get your computer IP address: <http://v4.ident.me/> or <http://www.myipaddress.com> . The IP address will be something like: 182.345.52.2.

Now that you have your IP address, we can now proceed to set up the rules.

- i) Log into the cPouta web interface.
- ii) Select your project as before (top left corner)
- iii) Click Access and Security (figure 11)
- iv) Select the security group you created earlier. For this tutorial, we created “demo\_SSH”
- v) Click “Manage Rules” on the right and click “Add Rule” (top right corner) from the drop-down menu. This will open a pop-up window (figure 12).

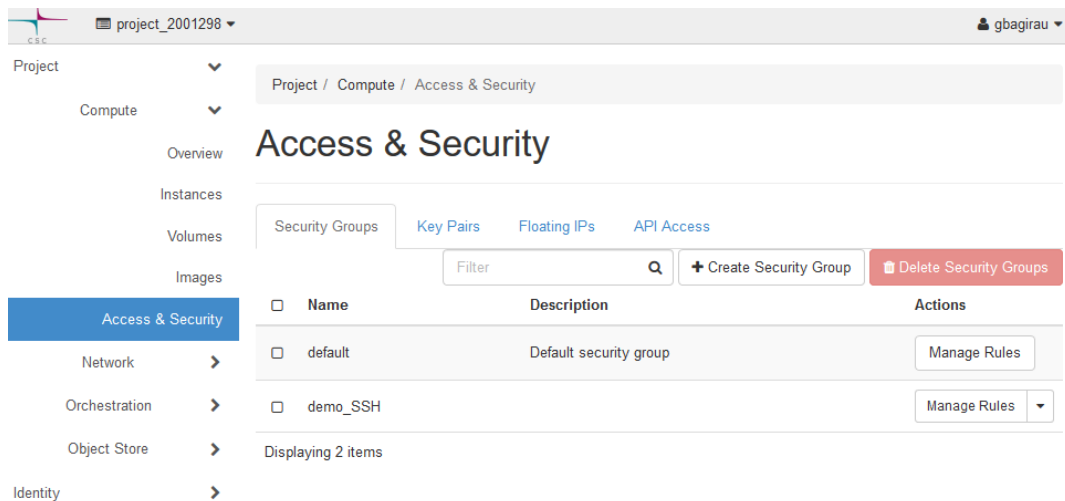


Figure 11 Shows how to modify security rules for your vm

- vi) Replace the “0.0.0.0/0” in the CIDR field with the IP address of your local computer you copied earlier. Also, use “22” in the Port field and leave other fields with the default settings (figure 12). When done, click “Add” at the bottom right corner.

**NOTE:**

- (a) You can modify the other rules based on what you need. However, you must know what you are doing otherwise you are risking the security of your vm and opening it up for hacking. Read more here: <https://research.csc.fi/pouta-getting-started>.
- (b) You can use the same keypair and security group for multiple vm’s in your project. This means, you can just do this step once and do not repeat for all your vm’s unless you choose to do so based on the sensitive nature of what you are doing.

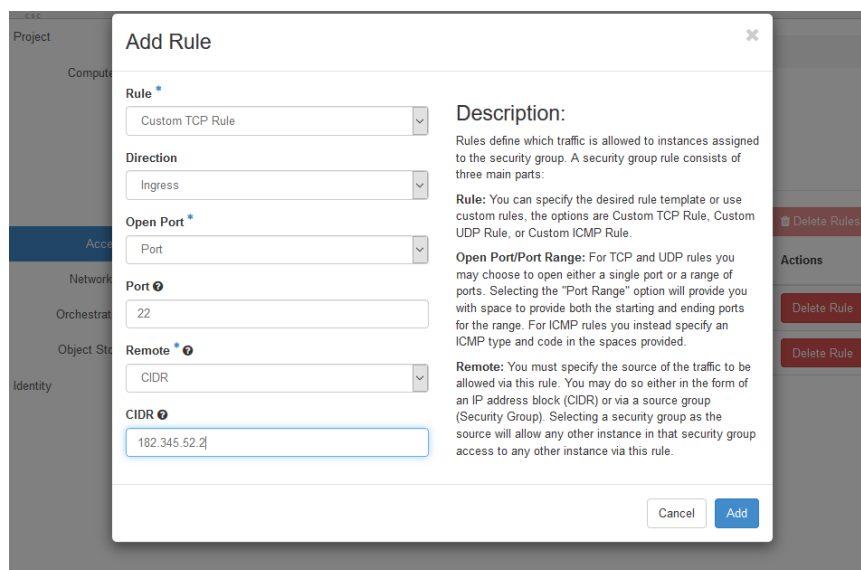


Figure 12 Shows basic rules for your vm.

After this, we are now ready to create our vm.

### Step 1.5 Creating our virtual machine (vm)

For this tutorial we will use Ubuntu-18.04 image. You can use any other image in the list or even your own image but be mindful of the reliability and maintenance of the image you are using. Depending on the image you use, there may be some small differences in their use (e.g. log in “username”, security rules, e.t.c.).

- i) Click “Images” left panel on cPouta web interface.
- ii) Look for the image named Ubuntu-18.04 and click “Launch” (bottom right corner), figure 13.

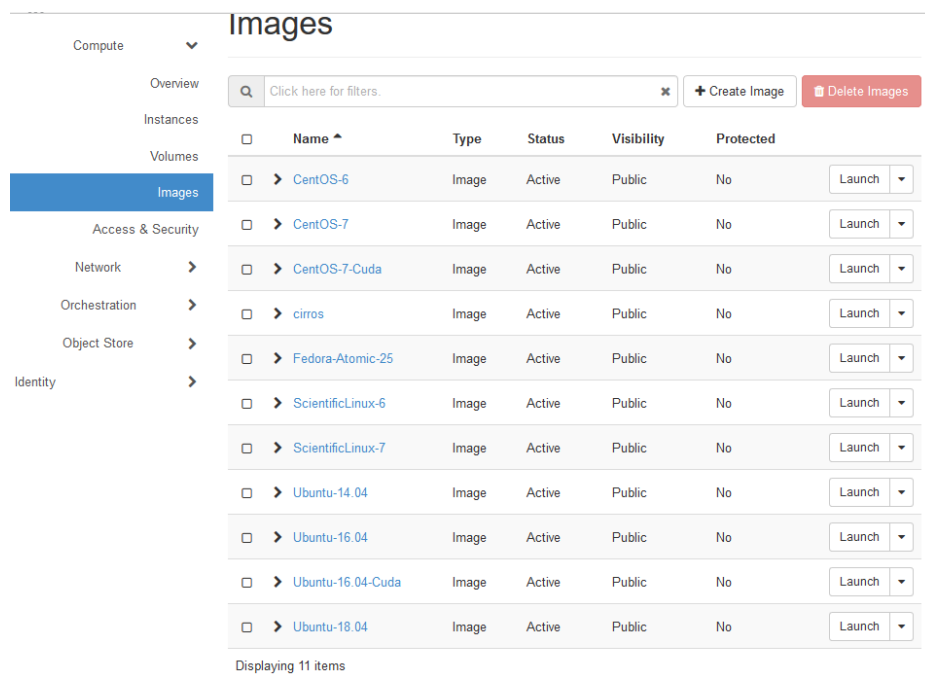


Figure 13 Shows several pre-existing CSC images to be used to create a vm.

- iii) Clicking “Launch” will open a new window (figure 14). We will now fill the boxes as shown in the figure. Click next after completing each section. At the end of the whole process, we will then launch our vm by clicking “Launch Instance” (bottom right corner). The “Launch Instance” will become active after all previous steps have been completed. for this tutorial, here are the settings we will perform: (a) Details: demo\_vm (b) Source: Create New Volume =No (c) Flavor (size of vm): select standard. tiny (d) In Security and Key Pair: Select the ones you created earlier. In this tutorial: “demo\_SSH” and “demo\_key.”
- iv) Modify these settings according to figures 14 to 21. You can launch at figure 21.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name

Availability Zone

nova

Count

1

Total Instances (8 Max)

13%

0 Current Usage

1 Added

7 Remaining

Cancel

Back

Next

Launch Instance

Figure 14 Shows the launch interface to be filled before creating your vm.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name

demo\_vm

Availability Zone

nova

Count

1

Total Instances (8 Max)

13%

0 Current Usage

1 Added

7 Remaining

Cancel

Back

Next

Launch Instance

Figure 15 Shows filling the first section “Details” before creating your vm.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Instance source is the template used to create an instance. You can use a snapshot of an existing instance, an image, or a volume (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source

Image

Create New Volume

Yes

No

Allocated

Name	Updated	Size	Type	Visibility
Ubuntu-18.04	7/1/19 1:02 PM	327.31 MB	qcow2	Public

Available 10

Select one

Click here for filters.

Name	Updated	Size	Type	Visibility
CentOS-7-Cuda	7/1/19 11:10 AM	3.13 GB	qcow2	Public
Fedora-Atomic-25	7/1/19 12:01 PM	669.38 MB	qcow2	Public
Ubuntu-16.04-Cuda	7/1/19 11:17 AM	3.10 GB	qcow2	Public

Cancel

Back

Next

Launch Instance

Figure 16 Shows filling the first section “Source” before creating your vm.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

Name	VCPUS	RAM	Total Disk	Public
standard.tiny	1	1000 MB	80 GB	Yes

Available 19

Select one

Click here for filters.

Name	VCPUS	RAM	Total Disk	Public
standard.small	2	1.95 GB	80 GB	Yes
standard.medium	3	3.91 GB	80 GB	Yes
standard.large	4	7.81 GB	80 GB	Yes
io.70GB	2	9.77 GB	90 GB	Yes
standard.xlarge	6	15.63 GB	80 GB	Yes

Cancel

Back

Next

Launch Instance

Figure 17 Shows filling the first section “Flavor” before creating your vm.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Networks provide the communication channels for instances in the cloud.

Allocated 1

Select networks from those listed below.

Network	Shared	Admin State	Status
project_2001298	No	Up	Active

Available 0

Select at least one network

Click here for filters.

Network	Shared	Admin State	Status
No available items			

Cancel

Back

Next

Launch Instance

Figure 18 Shows filling the first section “Networks” before creating your vm.

13

**Launch Instance**

Details

Source

Flavor

Networks

**Network Ports**

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Ports provide extra communication channels to your instances. You can select ports instead of networks or a mix of both.

▼ Allocated Select ports from those listed below.

Name ^	Admin State	Status
Select an item from Available items below		

▼ Available 0 Select one

Q Filter

Name ^	Admin State	Status
No available items		

✕ Cancel < Back Next > Launch Instance

Figure 19 Shows filling the first section “Network Ports” before creating your vm.

**Launch Instance**

Details

Source

Flavor

Networks

Network Ports

**Security Groups**

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Select the security groups to launch the instance in.

▼ Allocated 2

Name ^
> default
> demo_SSH

▼ Available 0 Select one or more

Q Click here for filters. ✕

Name ^	Description
No available items	

✕ Cancel < Back Next > Launch Instance

Figure 20 Shows filling the first section “Security Groups” before creating your vm.

**Launch Instance**

Details

Source

Flavor

Networks

Network Ports

Security Groups

**Key Pair**

Configuration

A key pair allows you to SSH into your newly created instance. You may select an existing key pair, import a key pair, or generate a new key pair.

+ Create Key Pair Import Key Pair

Allocated

Name
> demo_key

Displaying 1 item

▼ Available 3 Select one

Q Click here for filters. ✕

Name
------

Figure 21 Shows filling the first section “Key Pair” before creating your vm.

## Step 1.6 Launch your virtual machine

We are now ready to launch our vm.

- (i) Click on “Launch Instance” (figure 20, bottom right). This will activate your vm and is now available.
- (ii) Click on “Instance” on left panel to check if it was created successfully (figure 22).

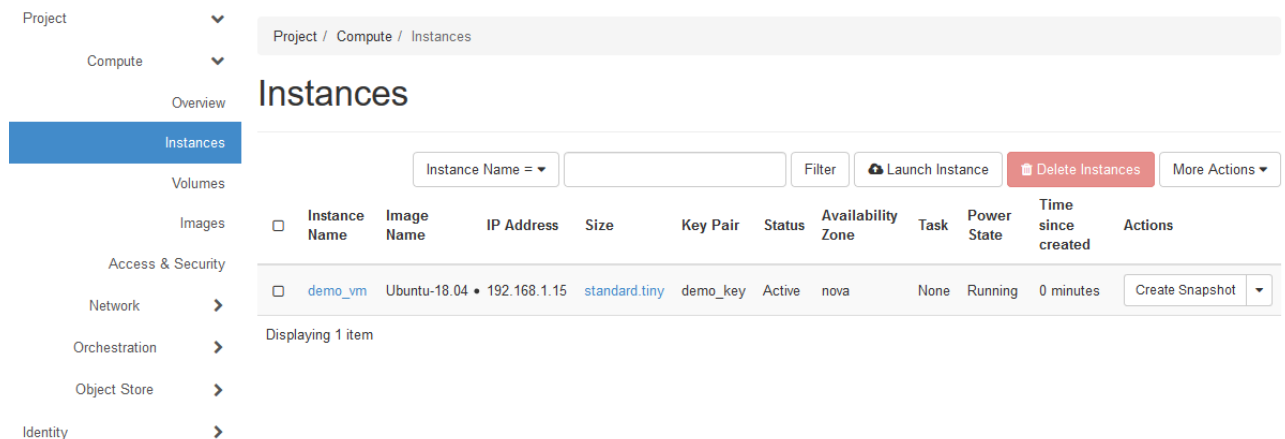


Figure 22 Shows the new created vm.

## Step 2 Connecting to your vm

To connect to your new vm, you will need to assign it a public IP address because the current IP address is a local one.

### Step 2.1 To assign a public IP address:

- (i) Click drop-down menu (right of your vm) and select “Associate Floating IP”, figure 23. A new window will open (figure 24).
- (ii) From this new window, select an IP address from the drop-down menu (if none are available, click the “+” sign).
- (iii) Finally, click “Associate” (bottom right corner) to associate this new IP to your machine. We will use this new IP to connect to your vm in the next step.



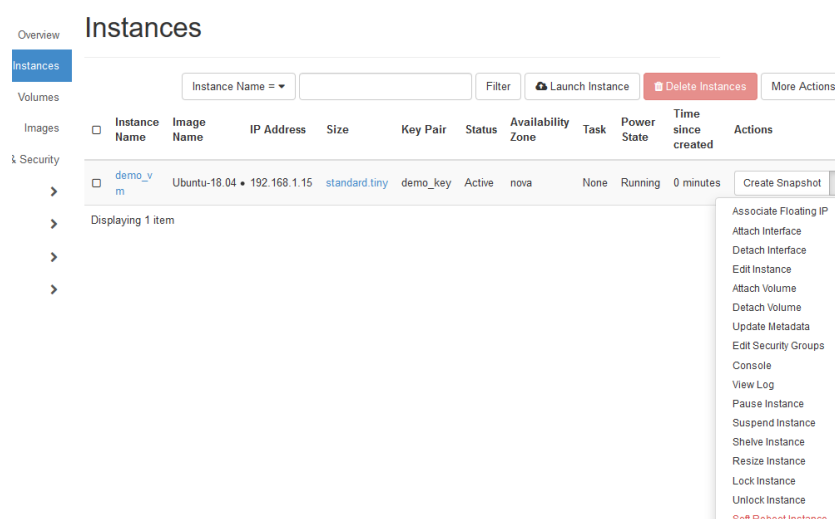


Figure 23 Shows how to select and associate a “Floating IP Address” to your new vm.

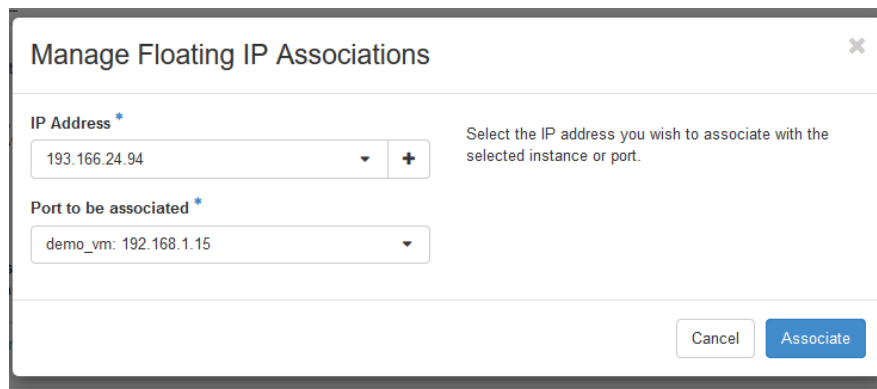


Figure 24 Shows the associated new floating ip address to our new vm.

## Step 2.2 Connecting to your vm

We will now connect to our vm using Putty (Windows users). For Linux, this can be done using SSH commands. But before we connect, we MUST first authenticate the “Key Pair” located on our local pc. This will enable us to connect to our vm located on the “Remote cPouta Server.”

NOTE:

- 1) You are the administrator of your VM with total control.
- 2) You are responsible for installing and maintaining your VM software’s.
- 3) You are responsible for the **security and use** of your VM.
- 4) You are responsible for configuring and setting all security and firewalls to enable you connect to your VM and host services (e.g. http, databases, map servers e.t.c.).

### Step 2.2.1 Authenticate your key pair using PuTTY SSH

- a) Launch Putty and add the public IP of your VM to box “Host Name or IP address. You can also give both username and IP at the same time; e.g. ubuntu@x.x.x.x.
- b) Click “Connection” under Category on the left panel of the Putty window.

- c) Click ‘+’ sign beside “SSH” to open up additional commands.

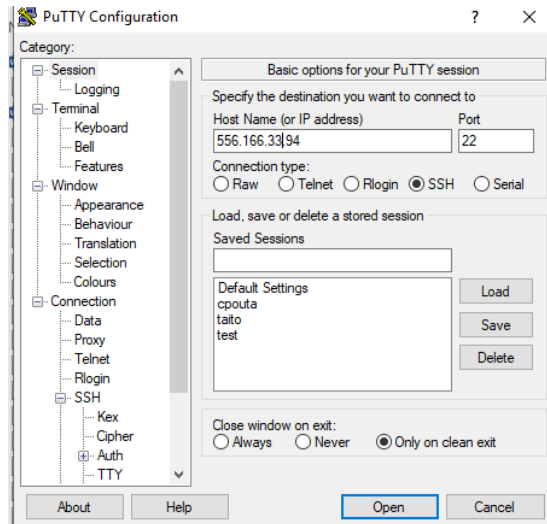


Figure 25 Shows the Putty SSH configuration and authentication interface. Here we input the “IP Address of our new vm” as the host name.

- (d) Click ‘on’ “Auth” to open up
- (e) Click on ‘Browse...’ under “Private key file for authentication” and navigate to where you saved your “.ppk” file and load it.

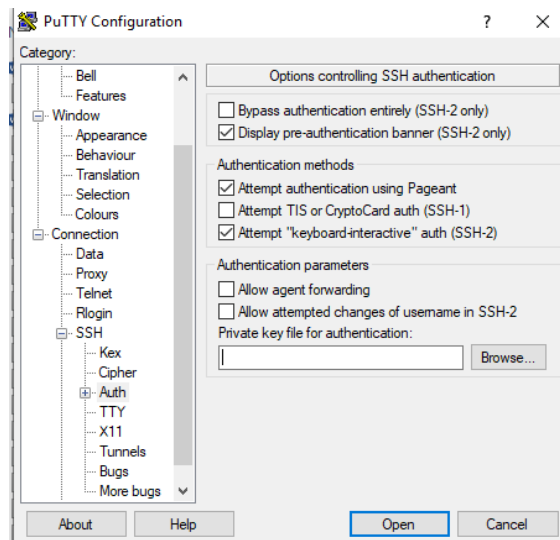


Figure 26 Shows the beginning of the authentication process. Here we browse to where we stored our “Key Pair” on our local pc.

- (f) You will see the following message below (figure 27), click ‘yes’ to continue to log in page.

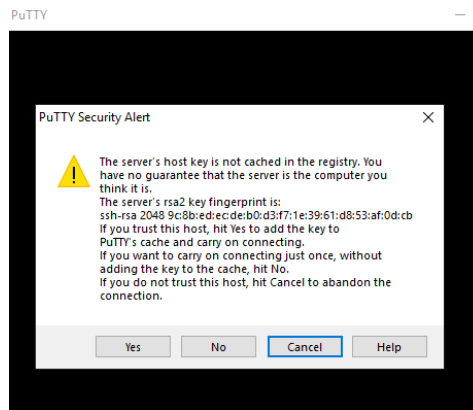


Figure 27 Shows the security alert and final confirmation window.

(g) Provide your username and key-phrase (Fig 28) and you are now logged in and good to go (Fig 29).

Note: Depending on the image you used, the username maybe different. We used “Ubuntu-18.04” in this tutorial and the user is ‘ubuntu.’

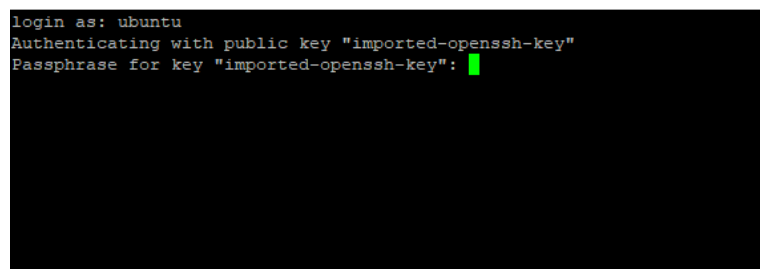


Figure 28 Shows log in window to your vm via Putty SSH.

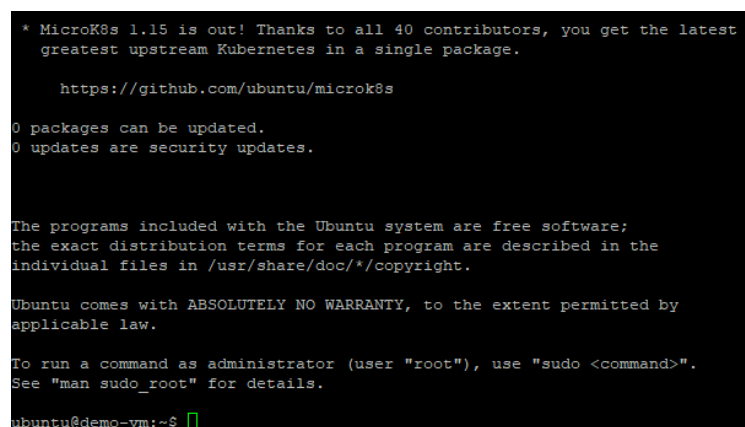


Figure 29 Shows we are logged into the home directory of our vm.

Step 3 Create, attach and mount a volume to your VM (virtual machine) in cPouta  
To use your new vm, it is advisable to attach a volume (similar to a hard drive on your local pc), as the basic system has only 80 GB of disk space. This is where we will store data, install software’s

and run analysis. We will now create this new volume. If you already have an existing volume, skip this part and proceed to “Step 3.2” and attach this volume to your vm.

### Step 3.1 Create a new volume

We will use the cPouta web interface to create this new volume because it is much easier for those who are not familiar with **command line interface**. But if you are very comfortable using the “command line interface”, you will find instructions [→here](https://research.csc.fi/pouta-persistent-volumes)

From the cPouta web interface,

- (1) Click on “Volumes” on the left panel (figure 30).
- (2) Click on “Create Volume” on top middle right (figure 30). What you must specify here is the size of the volume (in gigabit), the minimum being 1GB. (Source: CSC <https://research.csc.fi/pouta-persistent-volumes>)

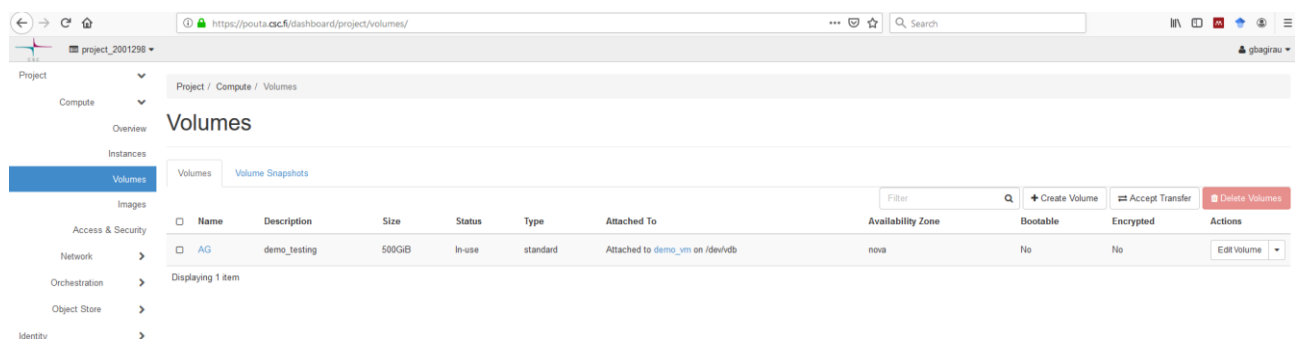


Figure 30 Shows the cPouta web interface to create a new volume.

You can select an old image as a source to create your volume. If so, click the drop down button under “Volume Source” figure 31 and select Image, otherwise just leave this field empty and proceed to “2”.

The screenshot shows the 'Create Volume' form. It has two columns. The left column contains: 'Volume Name' (text input with 'demo\_test'), 'Description' (text area), 'Volume Source' (dropdown menu with 'No source, empty volume' selected), 'Type' (dropdown menu with 'standard' selected), 'Size (GiB)' (text input with '1'), and 'Availability Zone' (dropdown menu with 'nova' selected). The right column contains: 'Description:' (text) with a sub-description 'Volumes are block devices that can be attached to instances.', 'Volume Type Description: standard' (text) with a sub-description 'Volumes stored on a Ceph backend', and 'Volume Limits' (text) with two progress bars: 'Total Gigabytes (500 GiB)' showing 500 GiB used out of 1,000 GiB available, and 'Number of Volumes (1)' showing 1 out of 10 available. At the bottom right are 'Cancel' and 'Create Volume' buttons.

Figure 31 Shows the basic requirements for creating a new volume using the cPouta web interface.

- (3) Select the type of volume (tiny, large, e.t.c.) and set the size (figure 31). Give your volume a name and description (optional) and click on “Create Volume” bottom right (figure 31). This will create a new volume and is now available and displayed. In this tutorial, we created a volume named “demo\_test.”

### Step 3.2 Attach volume to new vm

To attach the newly created volume,

- (1) Click on “Edit Volume” under **Actions** (top right figure 30).
- (2) Click on “Manage Volume Attachments” from the drop-down menu. This will open a new window (figure 32). All available instances will appear here. Recall that we created “demo\_vm” (see figure 23).
- (3) Select the instance you created earlier and click “Attach Volume” bottom right (figure 32).

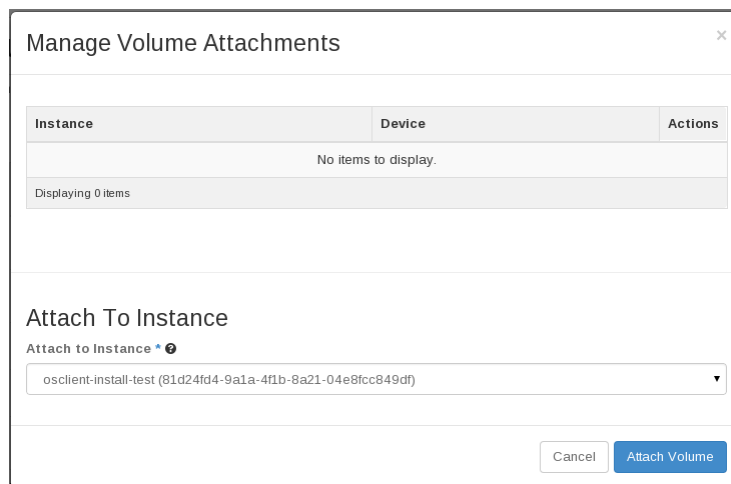


Figure 32 Shows how to select and attach a volume to your vm on cPouta web interface.

### Step 3.3 Detach volume from vm

If you need to detach this volume, just follow the same procedure you used for attaching volume in step 3.2. In this case you will click on “Detach Volume” bottom right (figure 33).

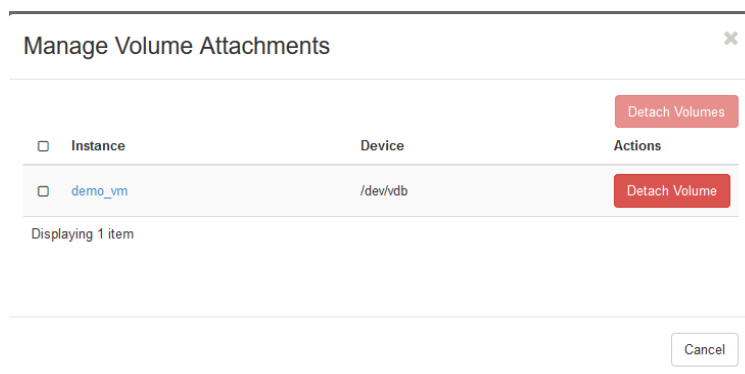


Figure 33 Shows how to detach a volume from vm in cPouta web interface.

### Step 3.4 Mount new volume to vm.

Before using this new volume, you must first create a file system and mount your vm onto this file system.

Note:

- If this is the first time you are using the attached new volume with your vm, this volume must be initialized. **This should ONLY be done the FIRST time you use it**, otherwise you will overwrite all data on the volume.
- If you are going to use a volume created by others, **please** verify if the initialization has been carried out or not.

Now let's continue with mounting the volume to our vm.

#### Step 3.4.1 Create file system on new volume

After logging into your vm,

- (a) List all available volumes (output should be similar to that shown in figure 34).

```
sudo parted -l
```

*“You should be able to identify the volume based on its size. For this exercise, let's say that it is /dev/vdb. We are going to use xfs because we know that it works well in cPouta: (Source: CSC <https://research.csc.fi/pouta-persistent-volumes> )*

```
ubuntu@demo-vm:~$ sudo parted -l
Error: /dev/vdb: unrecognised disk label

Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 537GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:

Model: Virtio Block Device (virtblk)
Disk /dev/vda: 85.9GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 14     1049kB 5243kB  4194kB                bios_grub
 15     5243kB 116MB   111MB   fat32         boot, esp
 1      116MB  85.9GB  85.8GB   ext4
```

Figure 34 Shows the list of available volume for my vm

- (b) Create a file system (figure 35).

```
sudo mkfs.xfs /dev/vdb
```

```

ubuntu@demo-vm:~$ sudo mkfs.xfs /dev/vdb
meta-data=/dev/vdb          isize=512    agcount=4, agsize=327
68000 blks
        =                       sectsz=512   attr=2, projid32bit=1
        =                       crc=1        finobt=1, sparse=0, r
mapbt=0, reflink=0
data      =                       bsize=4096   blocks=131072000, ima
xpct=25   =                       sunit=0      swidth=0 blks
naming    =version 2           bsize=4096   ascii-ci=0 ftype=1
log        =internal log      bsize=4096   blocks=64000, version
=2        =                       sectsz=512   sunit=0 blks, lazy-co
unt=1      =                       extsz=4096   blocks=0, rtextents=0
realtime  =none

```

Figure 35 Shows the working file system created.

### Step 3.4.2 Create a new directory, path and mount your new vm

In this tutorial we want to mount our vm to the directory “docker”. We will install docker later in this directory.

#### (a) Create new directory

```
sudo mkdir -p /demo/docker
```

#### (b) Mount vm to this directory

```
sudo mount /dev/vdb /demo/docker
```

#### (c) Check that directory has been created successfully

```
df -h
```

#### (d) Navigate and list content of docker directory (figure 36). This directory should be empty now because we have not installed docker yet.

```
cd/demo/docker/ls -al
```

```

ubuntu@demo-vm:~$ mkdir /demo
mkdir: cannot create directory '/demo': File exists
ubuntu@demo-vm:~$ sudo mkdir /demo
mkdir: cannot create directory '/demo': File exists
ubuntu@demo-vm:~$ sudo mount /dev/vdb /demo
ubuntu@demo-vm:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            469M   0  469M   0% /dev
tmpfs           97M   652K   96M   1% /run
/dev/vda1       78G   1.2G   77G   2% /
tmpfs           481M   0  481M   0% /dev/shm
tmpfs           5.0M   0   5.0M   0% /run/lock
tmpfs           481M   0  481M   0% /sys/fs/cgroup
/dev/vda15      105M   3.6M  101M   4% /boot/efi
tmpfs           97M   0   97M   0% /run/user/1000
/dev/vdb        500G   543M  500G   1% /demo
ubuntu@demo-vm:~$

```

Figure 36 Displays the file systems, contents and directories of our vm.

#### (e) Make volume available whenever you reboot your vm. You will need to add it to the **/etc/fstab** configuration file. You can use the label that you previously created for the partition and modify

the below code to match. If you don't do this, any time you reboot your vm, you will need to re-attach and mount the volume all over.

#MODIFY THIS CODE HERE#

####

```
sudo sh -c 'echo
"/dev/vdb      /media/volume    xfs      defaults,nofail    0      2" >>
/etc/fstab'
```

#####

# We modified the above code by replacing “/media/volume” with “/demo”, the directory we created previously.

```
sudo sh -c 'echo "/dev/vdb      /demo      xfs      defaults,nofail    0      2" >>
/etc/fstab'
```

NOTE:

- In the future, if you want to attach this volume to any other vm, you only need to run two lines of code (REMEMBER THE PATH TO THE DIRECTORY OF YOUR VM):

```
sudo mkdir -p /demo/docker
sudo mount /dev/vdb /demo/docker
```

- If at this stage you want to detach this volume from the vm, you **MUST** first unmount.

```
sudo umount /dev/vdb
```

Now we are ready to install applications, copy data and perform data analysis.



## CHAPTER 2: INSTALL DOCKER AND OPEN DRONE MAP(ODM) ON CPOUTA

Install docker and ODM on cPouta

This chapter is divided into four parts: (1) Install docker (2) Install ODM (3) Copy data to cPouta using WinSCP, and (4) Test ODM.

### 1. Installing Docker

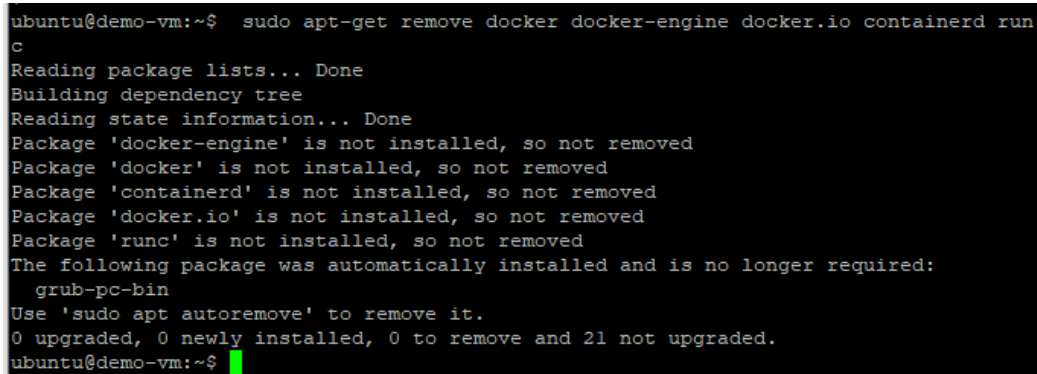
Step 1.0 Pre-check uninstall old docker and install new docker version.

It is good practice to always perform this step even if you have not previously installed docker. This will ensure you have installed the latest and current docker version (figure 1).

#Run this code

YOU CAN COPY THIS CODE USING “COPY AND PASTE”. Paste in the Putty VM window by pressing the *right mouse button*.

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

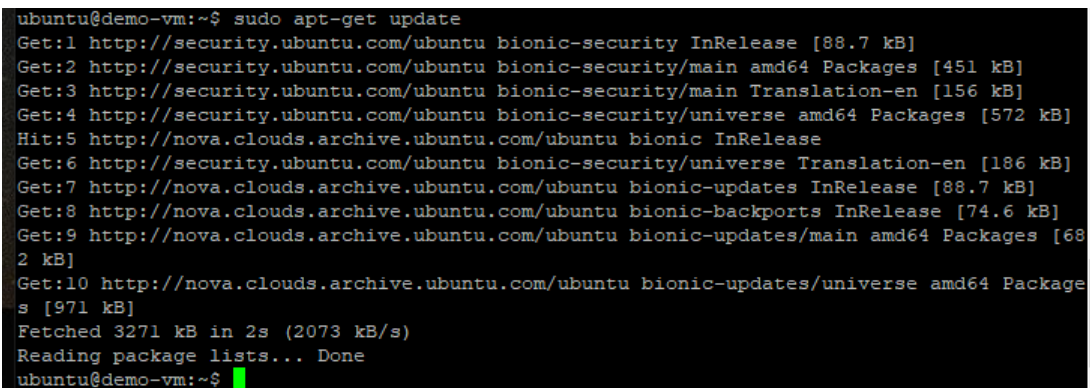


```
ubuntu@demo-vm:~$ sudo apt-get remove docker docker-engine docker.io containerd runc
c
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package 'docker-engine' is not installed, so not removed
Package 'docker' is not installed, so not removed
Package 'containerd' is not installed, so not removed
Package 'docker.io' is not installed, so not removed
Package 'runc' is not installed, so not removed
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 21 not upgraded.
ubuntu@demo-vm:~$
```

Fig 1 Shows output indicating that any old version of docker has been removed successfully

Step 1.2: Update the apt package index:

```
sudo apt-get update
```



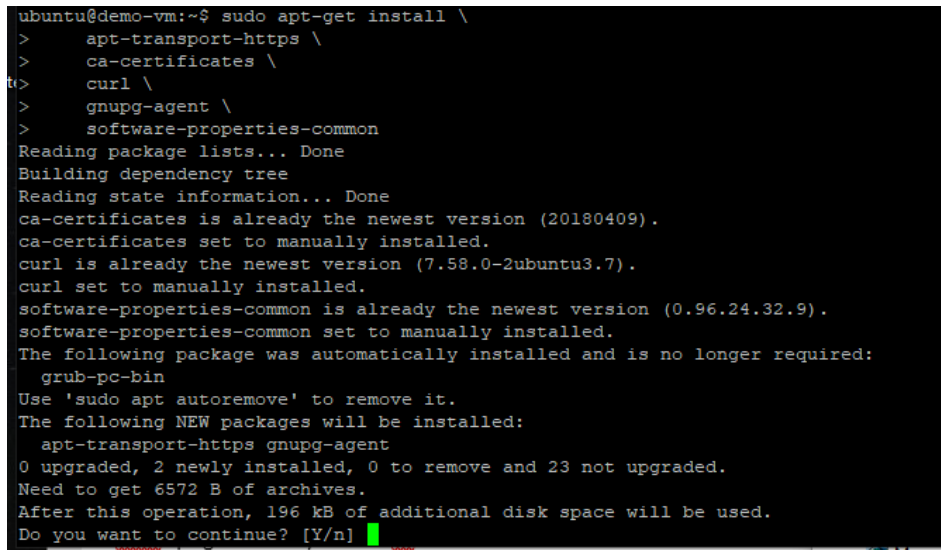
```
ubuntu@demo-vm:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [451 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [156 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [572 kB]
Hit:5 http://nova.clouds.archive.ubuntu.com/ubuntu bionic InRelease
Get:6 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en [186 kB]
Get:7 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:8 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:9 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [682 kB]
Get:10 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [971 kB]
Fetched 3271 kB in 2s (2073 kB/s)
Reading package lists... Done
ubuntu@demo-vm:~$
```

Fig 2 Indicates all packages updated successfully.

Step 1.3: Install packages to allow apt to use a repository over HTTPS:

```
#
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

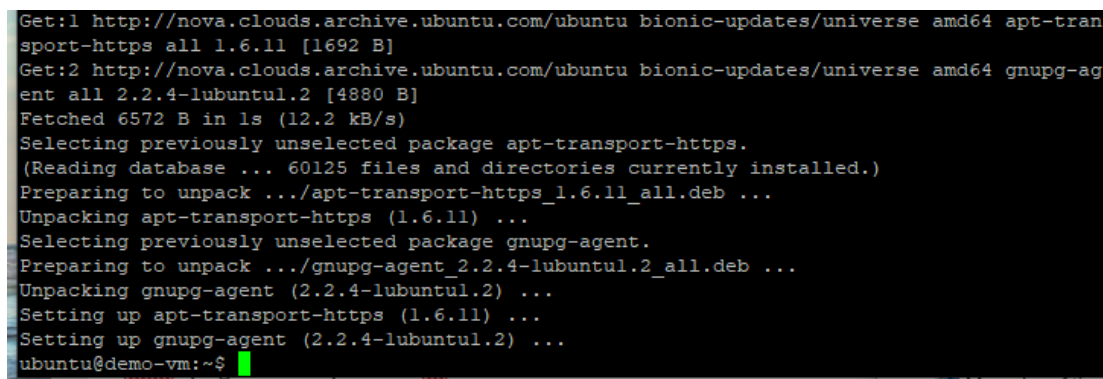
After running this command, you will be prompted to confirm the installation of docker (figure 3). Type Yes/Y and press 'enter' on your keyboard to continue and complete the installation.



```
ubuntu@demo-vm:~$ sudo apt-get install \
> apt-transport-https \
> ca-certificates \
t> curl \
> gnupg-agent \
> software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20180409).
ca-certificates set to manually installed.
curl is already the newest version (7.58.0-2ubuntu3.7).
curl set to manually installed.
software-properties-common is already the newest version (0.96.24.32.9).
software-properties-common set to manually installed.
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  apt-transport-https gnupg-agent
0 upgraded, 2 newly installed, 0 to remove and 23 not upgraded.
Need to get 6572 B of archives.
After this operation, 196 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Fig 3 Shows confirmation page before repository packages

After a successful installation, you will be returned to the default root user that is displayed any time you log into your virtual machine (figure 4)



```
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 apt-transport-https all 1.6.11 [1692 B]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 gnupg-agent all 2.2.4-lubuntu1.2 [4880 B]
Fetched 6572 B in 1s (12.2 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 60125 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_1.6.11_all.deb ...
Unpacking apt-transport-https (1.6.11) ...
Selecting previously unselected package gnupg-agent.
Preparing to unpack .../gnupg-agent_2.2.4-lubuntu1.2_all.deb ...
Unpacking gnupg-agent (2.2.4-lubuntu1.2) ...
Setting up apt-transport-https (1.6.11) ...
Setting up gnupg-agent (2.2.4-lubuntu1.2) ...
ubuntu@demo-vm:~$
```

Fig 4 Indicates all packages installed correctly

Step 1.4: Add Docker's official GPG key and verify the key fingerprint.

This step is required to ensure that all your downloads are valid. This action will add the key for the official Docker Repository to your system. Also, the Docker repository will be added to the APT sources (figure 5).

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

sudo apt-key fingerprint 0EBFCD88
```

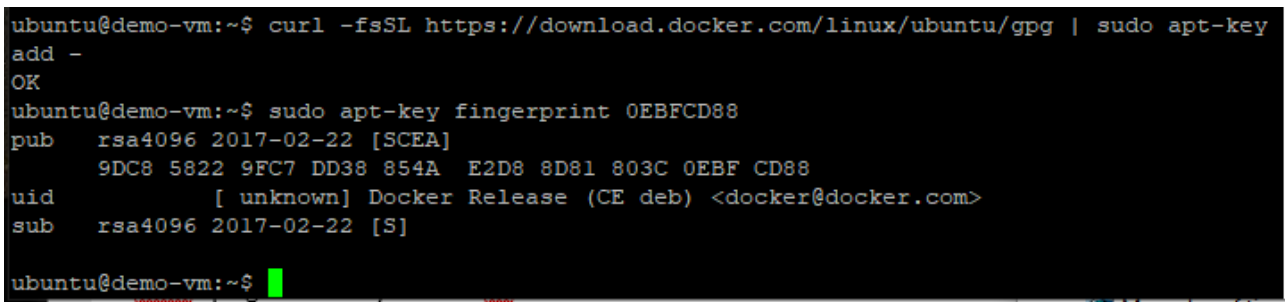
A terminal window showing the execution of commands to add the Docker repository key. The prompt is 'ubuntu@demo-vm:~\$'. The first command is 'curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -', which returns 'OK'. The second command is 'sudo apt-key fingerprint 0EBFCD88', which displays the key details: 'pub rsa4096 2017-02-22 [SCEA] 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88', 'uid [ unknown] Docker Release (CE deb) <docker@docker.com>', and 'sub rsa4096 2017-02-22 [S]'. The prompt returns to 'ubuntu@demo-vm:~\$'.

Fig 5 Indicates docker repository and fingerprint have been added successfully

#### Step 1.5: Set up a stable docker repository

```
sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
```

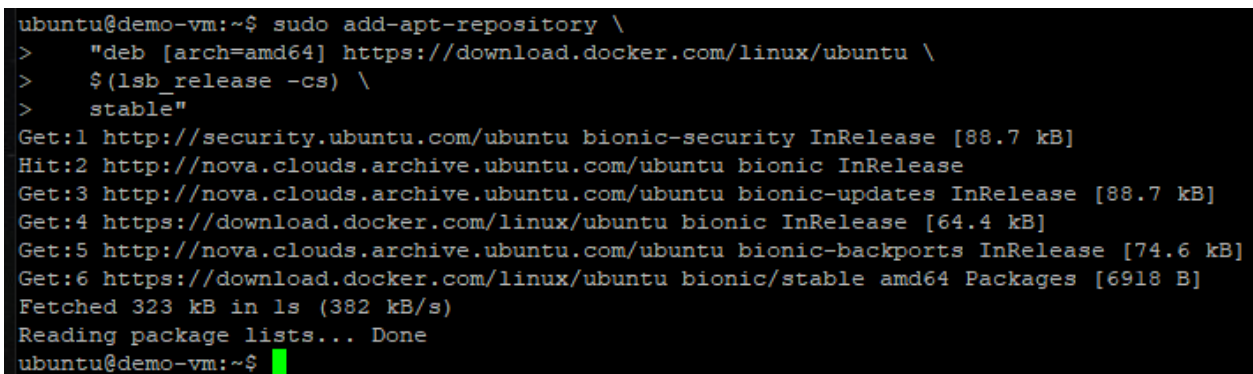
A terminal window showing the execution of the 'add-apt-repository' command. The prompt is 'ubuntu@demo-vm:~\$'. The command is 'sudo add-apt-repository \> "deb [arch=amd64] https://download.docker.com/linux/ubuntu \> \$(lsb\_release -cs) \> stable"'. The output shows the repository being added: 'Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]', 'Hit:2 http://nova.clouds.archive.ubuntu.com/ubuntu bionic InRelease', 'Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]', 'Get:4 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]', 'Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]', 'Get:6 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [6918 B]', 'Fetched 323 kB in 1s (382 kB/s)', 'Reading package lists... Done'. The prompt returns to 'ubuntu@demo-vm:~\$'.

Fig 6 Shows stable repository successfully unpacked and installed.

#### Step 1.6: Install the *latest version* of Docker CE and containerd

If you need to install a specific Docker version go to **step 1.6.1**.

# Run this command below. When prompted to confirm, type Yes/Y and hit 'enter' to complete installation of Docker (figure 7). After installation, you be returned to the default root user log-in interface (figure 8).

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

```

ubuntu@demo-vm:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  aufs-tools cgroupfs-mount libltdl7 pigz
The following NEW packages will be installed:
  aufs-tools cgroupfs-mount containerd.io docker-ce docker-ce-cli libltdl7 pigz
0 upgraded, 7 newly installed, 0 to remove and 23 not upgraded.
Need to get 53.4 MB of archives.
After this operation, 253 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Fig 7 Shows confirmation message before installing docker

```

systemd/system/containerd.service.
Processing triggers for ureadahead (0.100.0-21) ...
Setting up cgroupfs-mount (1.4) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.23) ...
Setting up libltdl7:amd64 (2.4.6-2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up docker-ce-cli (5:18.09.7~3-0~ubuntu-bionic) ...
Setting up pigz (2.4-1) ...
Setting up docker-ce (5:18.09.7~3-0~ubuntu-bionic) ...
update-alternatives: using /usr/bin/dockerd-ce to provide /usr/bin/dockerd (dockerd) in a
uto mode
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd
/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/sy
stem/docker.socket.
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.23) ...
ubuntu@demo-vm:~$

```

Fig 8 Indicates docker successfully installed

Step 1.6.1 If you want to install a specific version of Docker,

**SKIP** and go to step 1.7 if not necessary.

- first display a list of available versions (figure 9).

# Run the code below

apt-cache madison docker-ce

```

ubuntu@demo-vm:~$ apt-cache madison docker-ce
docker-ce | 5:18.09.7~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.6~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.5~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.4~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.3~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.2~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.1~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 5:18.09.0~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bioni
c/stable amd64 Packages
docker-ce | 18.06.3~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stab
le amd64 Packages
docker-ce | 18.06.2~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stab
le amd64 Packages
docker-ce | 18.06.1~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stab
le amd64 Packages
docker-ce | 18.06.0~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stab
le amd64 Packages
docker-ce | 18.03.1~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stab
le amd64 Packages
ubuntu@demo-vm:~$

```

Fig 9 Shows the list of available docker version you can select and install

- Install a specific version by replacing the <VERSION\_STRING> in the code below. In the example below, we presume we want to install the docker version 5:18.09.7~3-0~ubuntu-bionic .

# Modify and run this code e.g. with the strings

```
sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-  
cli=<VERSION_STRING> containerd.io
```

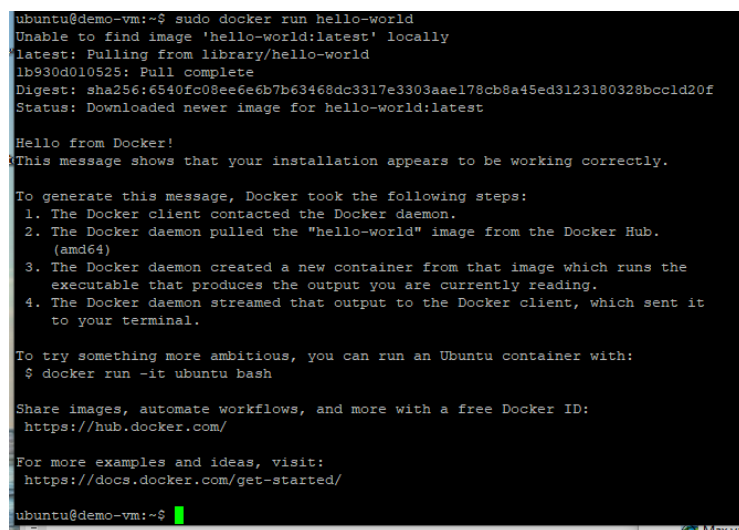
```
sudo apt-get install docker-ce=<5:18.09.7~3-0~ubuntu-bionic  
> docker-ce-cli=<5:18.09.7~3-0~ubuntu-bionic  
> containerd.io
```

Step 1.7: Verify that Docker is installed correctly

We can verify using the 'hello-world' Docker image (figure 10).

#Run this command

```
sudo docker run hello-world
```

A terminal window showing the output of the command 'sudo docker run hello-world'. The output indicates that the 'hello-world:latest' image was pulled from the Docker Hub. It then shows the container running and printing 'Hello from Docker!' followed by a message stating that the installation appears to be working correctly. It also lists the steps Docker took to generate this message: 1. The Docker client contacted the Docker daemon. 2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64) 3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading. 4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal. Finally, it suggests trying something more ambitious by running 'docker run -it ubuntu bash' and provides links for more information: 'https://hub.docker.com/' and 'https://docs.docker.com/get-started/'. The prompt 'ubuntu@demo-vm:~\$' is visible at the bottom.

```
ubuntu@demo-vm:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:6540fc08ee6e6b7b63468dc3317e3303aae178cb9a45ed3123180328bccld20f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

ubuntu@demo-vm:~$
```

Fig 10 Shows verification that docker installed successfully and running.

**CONGRATULATIONS!** You have now installed Docker on your cPouta vm.

Step 1.8: Updating Docker CE

To update Docker in the future, just run the command below.

#Run

```
sudo apt-get update
```

You can find detailed information about docker using these links below

<https://docs.docker.com/install/linux/docker-ce/ubuntu/>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04>

This basic installation will allow you to process about 200 images. For more images you must move and start docker on your attached volume.

instructions: (see appendix II).

## 2.0 Install Open Drone Map (ODM)

### 2.1 Build ODM docker image

#Run

```
git clone https://github.com/OpenDroneMap/OpenDroneMap.git
```

Note:

Sometimes it might be necessary to replace old versions of ODM before installing a newer one. In such an instance you can run the code below.

#Run

```
sudo docker pull opendronemap/opendronemap
```

### 2.2 Check the built docker images

#Run

```
sudo docker images -a
```

We observe three images available (figure 11). This is because, we ran the above command three times. So, it means that anytime you run the command above (step 2.1), you build and store a new image in your container.

```
ubuntu@demo-vm:~/OpenDroneMap$ sudo docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu@demo-vm:~/OpenDroneMap$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
ubuntu@demo-vm:~/OpenDroneMap$ sudo docker ps -a
CONTAINER ID        NAMES              IMAGE               COMMAND             CREATED             STATUS              PORTS
dd6b594a80d8       eloquent_shirley   hello-world         "/hello"            18 minutes ago      Exited (0)          18 minutes ago
173c1b6671d5       serene_wilson      hello-world         "/hello"            16 hours ago        Exited (0)          16 hours ago
f87b302c9144       goofy_jang         hello-world         "/hello"            17 hours ago        Exited (0)          17 hours ago
ubuntu@demo-vm:~/OpenDroneMap$
```

Fig 11 Shows ODM Docker images built successfully.

Now, make a copy (snapshot) of your working VM for later use. To do this, open your VM instance and click on “Create Snapshot” on the far right (figure 12).

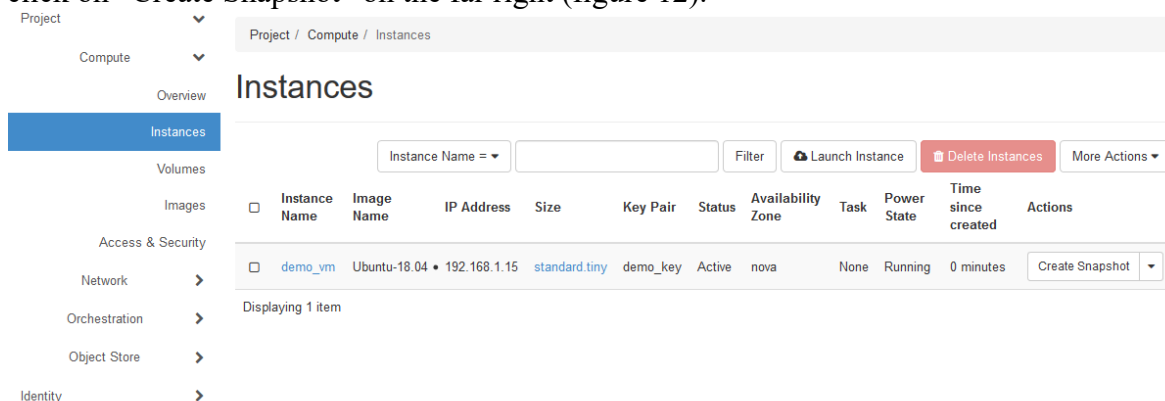


Fig 12 Displays VM instance.

## Additional ODM Resources

<https://hub.docker.com/r/opendronemap/opendronemap/>

<https://odmbook.com/>

<https://github.com/OpenDroneMap/ODM>

<https://docs.opendronemap.org/ODMinstallation.html>

### 3.0 Copy Files/Folders to cPouta vm

Here, we will demonstrate using the WinSCP graphical user interface (GUI) application.

Note:

- There are other applications you can use also
- You can use SSH command line to accomplish the same tasks

### 3.1 Install WinSCP

You can download and install WinSCP here: <https://winscp.net/eng/index.php>

If you already WinSCP installed, skip and move to 3.2.

### 3.2 Authenticate WinSCP

You **must authenticate** WinSCP with the same key phrase of your virtual machine (vm), otherwise you cannot connect to your vm. If you attempt to connect to your vm without authentication, you will receive the error shown in figure 13 below.

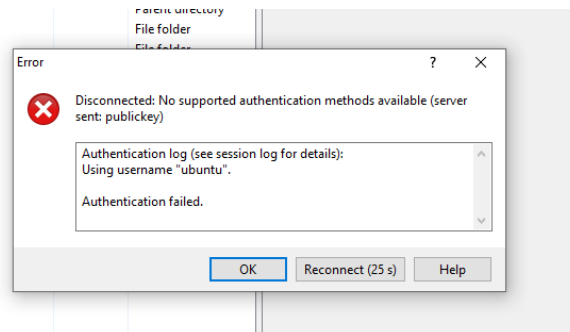
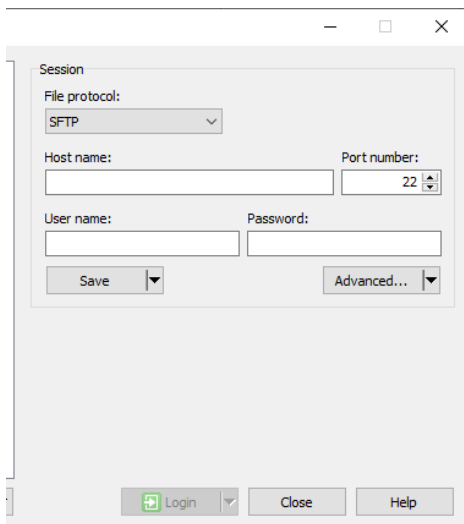
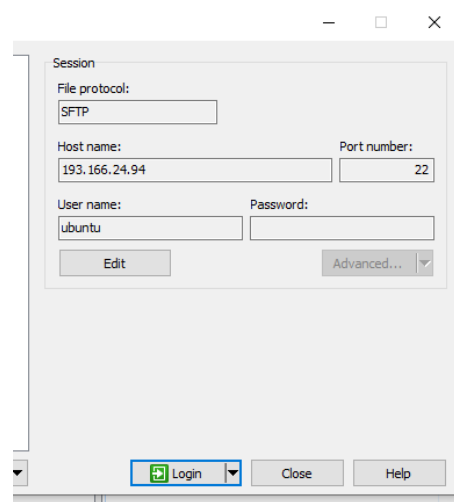


Fig 13 Shows WinSCP error message when attempting to connect to a password protected vm without authentication.

3.2.1: Open WinSCP (figure 14 a) and enter the host name of your vm (this will be the I.P address of your vm), the user name (this depends on the image you used in cPouta). For our case we used **ubuntu image** and user name is ubuntu (figure 14 b). Leave the password field empty because we have an '.SSH' passphrase already setup.



(a)



(b)

Fig 14 Shows the WinSCP log-in interface. In “b” we display the host name and username. The host name is the ip address of your virtual machine. The username will depend on the image you used for your vm in cPouta. For ubuntu, the username is ubuntu.

3.2.2: Click on the Advanced tab on the right (figure 14 a). From the dropdown, click **Advanced** to open the authentication page (figure 15).

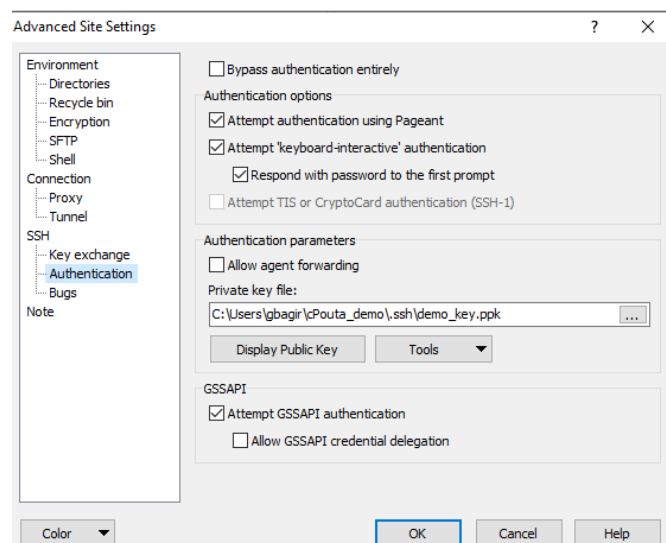


Fig 15 Shows WinSCP authentication interface.

3.2.3 From this open window, click ‘Authentication’ under SSH on the left panel.

**Click on the three dots “...”** on the right under “Private key file” under the box “**Authentication parameters**”. Navigate to the “.ssh folder” were you stored your private key for your vm. Load the file and click on ‘ok’. Click save to save the information. Now you are ready to log into your vm using WinSCP.

3.3.4 Log-in and connect to your vm on cPouta

Open WinSCP and enter your login details (figure 16). This also indicates that WinSCP has been successfully authenticated.



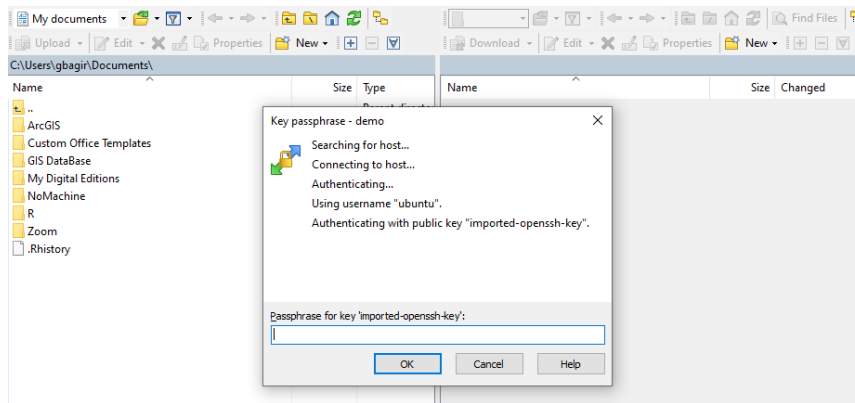


Fig 16 Shows WinSCP log-in interface when attempting to log in.

3.3.5 Enter your key phrase and press enter (figure 17). You can now copy files from your local pc (left panel) to your vm (right panel). In our case here, we are in the ODM directory. This is the directory from which we will run and process our images.

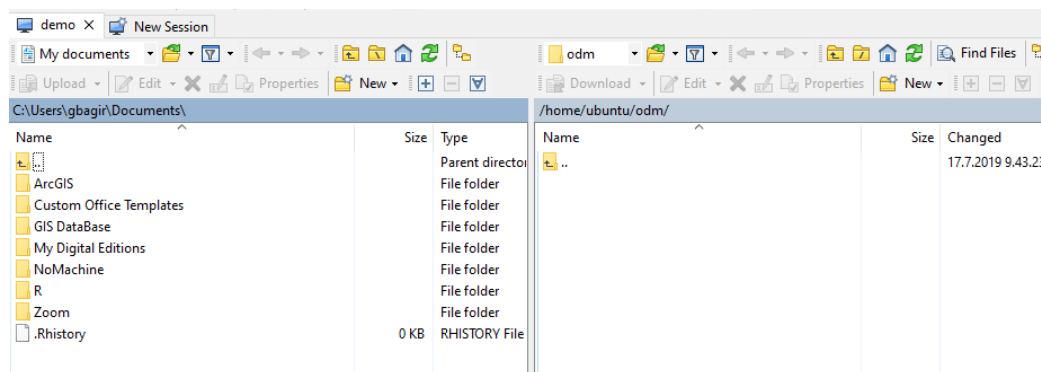


Fig 17 Shows the connection to our vm using WinSCP. From here we can copy files from our local pc (left panel) to our virtual machine (right panel).

**CONGRATULATIONS** for making it this far!

#### 4.0 Test ODM on cPouta

Now that we have installed ODM successfully copied some test data to our vm (figure 18), let's test that all is working correctly.

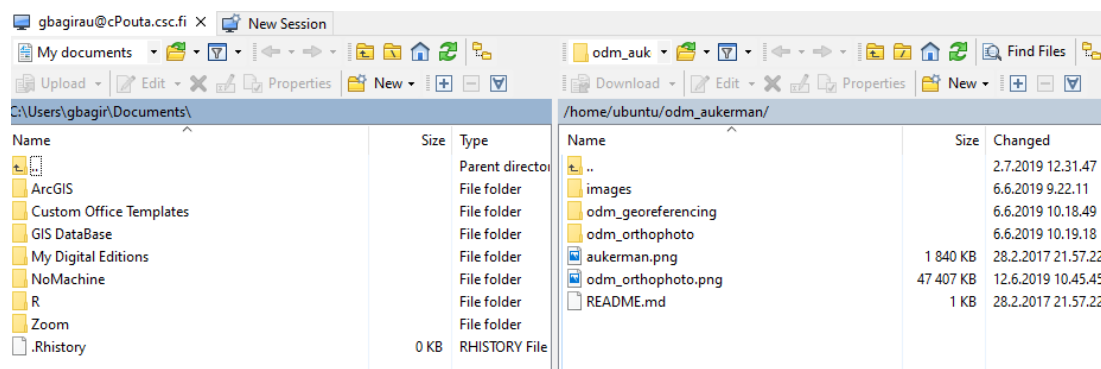


Fig 18 Shows the data in the ODM directory of our vm (right panel).

#### 4.1 Log in and connect to vm

Open PuTTY SSH, enter your username and key phrase (password) of your vm (figure 19).

Note: When you type in your key phrase it will not be visibly displayed, so make sure you type correctly. To run ODM, you must be in the ODM directory and one level below the “images” folder (figure 18).

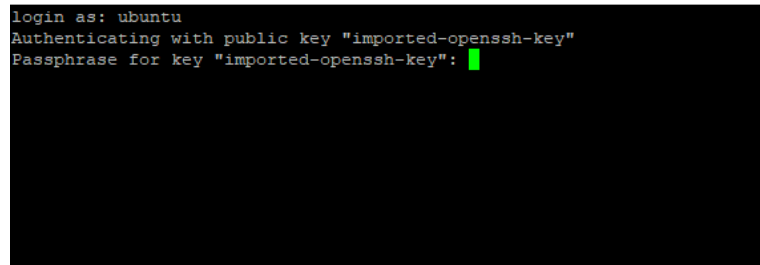


Fig 19 Shows log in window to your vm via Putty SSH.

4.1.2 Navigate to one folder below that which contains the images you want to process. In this tutorial we have our “images” in folder “odm\_aukerman”.

#Run

```
cd odm_aukerman
```

4.1.2 Process ODM images.

#Run

```
time sudo docker run -it --rm \  
  -v $(pwd)/images:/code/images \  
  opendronemap/opendronemap \  
  --dsm --dem-resolution 2 \  
  --dsm --dem-resolution 2 --smrf-threshold 0.4 --smrf-window 24\
```

After complete processing, the output will be stored in different folders (figure 18) depending on the parameters you have defined.

Note:

The processing time depends on;

- The number of images
- The type of parameters you have defined
- Other factors like memory e.t.c.

The code above is just few parameters. See Appendix III for additional parameters.

## Appendix I

Table 1. Some common linux commands you can use to get up and running on cPouta

Command Name	Description
<code>pwd</code>	prints the absolute path of current working directory
<code>cd</code>	changes current working directory
<code>cd ..</code>	moves you one level up to a different directory
<code>cd ~</code>	returns to home directory irrespective of which directory you are currently in
<code>ls</code>	prints contents of current directory
<code>ls -a</code>	prints all hidden files in current working directory
<code>ls -la</code>	prints the directories showing access permission, size, date modified, e.t.c
<code>ls -t</code>	prints contents of working directory according to time created
<code>ls -S</code>	re-orders the printed contents according to file size
<code>ls -ltrh</code>	prints recently modified files
<code>mkdir</code>	creates new directory
<code>rmdir</code>	removes an empty directory
<code>sudo</code>	gives you root access/ permissions usually reserved for Admin's
<code>df</code>	shows the size, free and used space on mounted filesystems on the computer
<code>df -h</code>	gives size of disc in MB or GB instead of bytes
<code>watch -n 5 df -h</code>	monitors disk usage
<code>watch -n 5 free -m</code>	monitors memory usage
<code>exit</code>	closes a window terminal, stops a shell script or logs you out.
<code>mv</code>	allows you to move files and directories
<code>cp</code>	copies files to different location
<code>cp *</code>	copies all files within a directory to a new directory
<code>cp -r</code>	copies directory together with files and sub-directories
<code>cp -a</code>	copies a file and preserves the files attributes
<code>rsync -r</code>	after first use, subsequently only files that have been modified will be copied
<code>ps</code>	list running processes in the opened current shell
<code>ps -e</code>	list all running processes
<code>chmod</code>	Sets file/folder access permissions

<i>curl</i>	Used to retrieve information by from other locations via URL's or internet addresses
<i>echo</i>	prints information to the command window inform of text strings
<i>grep</i>	will search and display using matching patterns. This can be used for a single line of text or the contents of files.
<i>tar</i>	for zipping (compressing) and archiving files/folders

See this guide for additional linux commands:

<https://research.csc.fi/csc-guide-linux-basics-for-csc>

## Appendix II

### Extra working space for docker:

See <https://linuxconfig.org/how-to-move-docker-s-default-var-lib-docker-to-another-directory-on-ubuntu-debian-linux>

```
sudo var/lib/systemd/system/docker.service
```

Change FROM:

```
ExecStart=/usr/bin/docker daemon -H fd://
```

TO:

```
ExecStart=/usr/bin/docker daemon -g /new/path/docker -H fd://
```

```
ExecStart=/usr/bin/dockerd -g /mnt/docker -H fd:// ( /mnt/docker is large mounted disk:
remember use only the absolute form of the path)
```

/new/path/docker = new directory on larger disk

Stop docker

```
sudo systemctl stop docker
```

Check that docker is completely stopped:

```
ps aux | grep -i docker | grep -v grep
```

No output = ok

Reload system daemon

```
sudo systemctl daemon-reload
```

Create new directory

```
mkdir /new/path/docker for example docker -> /home/cloud-user/mnt/docker
```

Resync

```
Sudo rsync -aqxP /var/lib/docker/ /home/cloud-user/mnt/docker (This takes some time)
```

Restart docker

```
sudo systemctl start docker
```

For docker info: `sudo docker info`

Check docker

```
sudo ps aux | grep -i docker | grep -v grep
```

Check docker works correctly

If Docker does not run correctly, you may need to re-install Docker (follow previous instructions from chapter 2).

## Appendix III

There are many command line arguments to use to process ODM. We have listed them below and we will give some examples on how to modify the commands to process your images and output your desired results.

```
-h, --help                show this help message and exit
--images <path>, -i <path> Path to input images
--project-path <path>      Path to the project folder
--resize-to <integer>      resizes images by the largest side for opensfm. Set to
                           -1 to disable. Default: 2048
--end-with <string>, -e <string> Can be one of:dataset | split | merge | opensfm | mve
                           | odm_filterpoints | odm_meshing | mvs_texturing |
                           odm_georeferencing | odm_dem | odm_orthophoto
--rerun <string>, -r <string> Can be one of:dataset | split | merge | opensfm | mve
                           | odm_filterpoints | odm_meshing | mvs_texturing |
                           odm_georeferencing | odm_dem | odm_orthophoto
--rerun-all               force rerun of all tasks
--rerun-from <string>      Can be one of:dataset | split | merge | opensfm | mve
                           | odm_filterpoints | odm_meshing | mvs_texturing |
                           odm_georeferencing | odm_dem | odm_orthophoto
--video <string>           Path to the video file to process
--slam-config <string>     Path to config file for orb-slam
--proj <PROJ4 string>      Projection used to transform the model into geographic
                           coordinates
--min-num-features <integer> Minimum number of features to extract per image. More
                           features leads to better results but slower execution.
                           Default: 8000
--matcher-neighbors <integer> Number of nearest images to pre-match based on GPS
                           exif data. Set to 0 to skip pre-matching. Neighbors
                           works together with Distance parameter, set both to 0
                           to not use pre-matching. OpenSFM uses both parameters
                           at the same time, Bundler uses only one which has
                           value, preferring the Neighbors parameter. Default: 8
--matcher-distance <integer> Distance threshold in meters to find pre-matching
                           images based on GPS exif data. Set both matcher-
                           neighbors and this to 0 to skip pre-matching. Default:
                           0
--use-fixed-camera-params  Turn off camera parameter optimization during bundler
--max-concurrency <positive integer> The maximum number of processes to use in various
                           processes. Peak memory requirement is ~1GB per thread
```

and 2 megapixel image resolution. Default: 4

--depthmap-resolution <positive float>  
Controls the density of the point cloud by setting the resolution of the depthmap images. Higher values take longer to compute but produce denser point clouds. Default: 640

--opensfm-depthmap-min-consistent-views <integer: 2 <= x <= 9>  
Minimum number of views that should reconstruct a point for it to be valid. Use lower values if your images have less overlap. Lower values result in denser point clouds but with more noise. Default: 3

--opensfm-depthmap-method <string>  
Raw depthmap computation algorithm. PATCH\_MATCH and PATCH\_MATCH\_SAMPLE are faster, but might miss some valid points. BRUTE\_FORCE takes longer but produces denser reconstructions. Default: PATCH\_MATCH

--opensfm-depthmap-min-patch-sd <positive float>  
When using PATCH\_MATCH or PATCH\_MATCH\_SAMPLE, controls the standard deviation threshold to include patches. Patches with lower standard deviation are ignored. Default: 1

--use-hybrid-bundle-adjustment  
Run local bundle adjustment for every image added to the reconstruction and a global adjustment every 100 images. Speeds up reconstruction for very large datasets.

--mve-confidence <float: 0 <= x <= 1>  
Discard points that have less than a certain confidence threshold. This only affects dense reconstructions performed with MVE. Higher values discard more points. Default: 0.6

--use-3dmesh  
Use a full 3D mesh to compute the orthophoto instead of a 2.5D mesh. This option is a bit faster and provides similar results in planar areas.

--skip-3dmodel  
Skip generation of a full 3D model. This can save time if you only need 2D results such as orthophotos and DEMs.

--use-opensfm-dense  
Use opensfm to compute dense point cloud alternatively

--ignore-gsd  
Ignore Ground Sampling Distance (GSD). GSD caps the maximum resolution of image outputs and resizes images when necessary, resulting in faster processing and lower memory usage. Since GSD is an estimate, sometimes ignoring it can result in slightly better image output quality.

--mesh-size <positive integer>  
The maximum vertex count of the output mesh. Default: 100000

--mesh-octree-depth <positive integer>  
Oct-tree depth used in the mesh reconstruction, increase to get more vertices, recommended values are 8-12. Default: 9

--mesh-samples <float >= 1.0>  
Number of points per octree node, recommended and default value: 1.0

--mesh-point-weight <positive float>  
This floating point value specifies the importance that interpolation of the point samples is given in the formulation of the screened Poisson equation. The results of the original (unscreened) Poisson Reconstruction can be obtained by setting this value to 0. Default= 4

--fast-orthophoto  
Skips dense reconstruction and 3D model generation. It generates an orthophoto directly from the sparse

reconstruction. If you just need an orthophoto and do not need a full 3D model, turn on this option.  
Experimental.

--crop <positive float>  
Automatically crop image outputs by creating a smooth buffer around the dataset boundaries, shrinked by N meters. Use 0 to disable cropping. Default: 3

--pc-classify  
Classify the point cloud outputs using a Simple Morphological Filter. You can control the behavior of this option by tweaking the --dem-\* parameters.  
Default: False

--pc-csv  
Export the georeferenced point cloud in CSV format.  
Default: False

--pc-las  
Export the georeferenced point cloud in LAS format.  
Default: False

--pc-filter <positive float>  
Filters the point cloud by removing points that deviate more than N standard deviations from the local mean. Set to 0 to disable filtering. Default: 2.5

--smrf-scalar <positive float>  
Simple Morphological Filter elevation scalar parameter. Default: 1.25

--smrf-slope <positive float>  
Simple Morphological Filter slope parameter (rise over run). Default: 0.15

--smrf-threshold <positive float>  
Simple Morphological Filter elevation threshold parameter (meters). Default: 0.5

--smrf-window <positive float>  
Simple Morphological Filter window radius parameter (meters). Default: 18.0

--texturing-data-term <string>  
Data term: [area, gmi]. Default: gmi

--texturing-nadir-weight <integer: 0 <= x <= 32>  
Affects orthophotos only. Higher values result in sharper corners, but can affect color distribution and blurriness. Use lower values for planar areas and higher values for urban areas. The default value works well for most scenarios. Default: 16

--texturing-outlier-removal-type <string>  
Type of photometric outlier removal method: [none, gauss\_damping, gauss\_clamping]. Default: gauss\_clamping

--texturing-skip-visibility-test  
Skip geometric visibility test. Default: False

--texturing-skip-global-seam-leveling  
Skip global seam leveling. Useful for IR data. Default: False

--texturing-skip-local-seam-leveling  
Skip local seam blending. Default: False

--texturing-skip-hole-filling  
Skip filling of holes in the mesh. Default: False

--texturing-keep-unseen-faces  
Keep faces in the mesh that are not seen in any camera. Default: False

--texturing-tone-mapping <string>  
Turn on gamma tone mapping or none for no tone mapping. Choices are 'gamma' or 'none'. Default: none

--gcp <path string>  
path to the file containing the ground control points used for georeferencing. Default: None. The file needs to be on the following line format: easting northing height pixelrow pixelcol imagename

--use-exif  
Use this tag if you have a gcp\_list.txt but want to

```

                                use the exif geotags instead
--dtm                        Use this tag to build a DTM (Digital Terrain Model,
                                ground only) using a simple morphological filter.
                                Check the --dem* and --smrf* parameters for finer
                                tuning.
--dsm                        Use this tag to build a DSM (Digital Surface Model,
                                ground + objects) using a progressive morphological
                                filter. Check the --dem* parameters for finer tuning.
--dem-gapfill-steps <positive integer>
                                Number of steps used to fill areas with gaps. Set to 0
                                to disable gap filling. Starting with a radius equal
                                to the output resolution, N different DEMs are
                                generated with progressively bigger radius using the
                                inverse distance weighted (IDW) algorithm and merged
                                together. Remaining gaps are then merged using nearest
                                neighbor interpolation. Default=3
--dem-resolution <float>
                                DSM/DTM resolution in cm / pixel. Default: 5
--dem-decimation <positive integer>
                                Decimate the points before generating the DEM. 1 is no
                                decimation (full quality). 100 decimates ~99% of the
                                points. Useful for speeding up generation. Default=1
--dem-euclidean-map          Computes an euclidean raster map for each DEM. The map
                                reports the distance from each cell to the nearest
                                NODATA value (before any hole filling takes place).
                                This can be useful to isolate the areas that have been
                                filled. Default: False
--orthophoto-resolution <float > 0.0>
                                Orthophoto resolution in cm / pixel. Default: 5
--orthophoto-no-tiled
                                Set this parameter if you want a stripped geoTIFF.
                                Default: False
--orthophoto-compression <string>
                                Set the compression to use. Note that this could break
                                gdal_translate if you don't know what you are doing.
                                Options: JPEG, LZW, PACKBITS, DEFLATE, LZMA, NONE.
                                Default: DEFLATE
--orthophoto-bigtiff {YES,NO,IF_NEEDED,IF_SAFER}
                                Control whether the created orthophoto is a BigTIFF or
                                classic TIFF. BigTIFF is a variant for files larger
                                than 4GiB of data. Options are YES, NO, IF_NEEDED,
                                IF_SAFER. See GDAL specs:
                                https://www.gdal.org/frmt\_gtiff.html for more info.
                                Default: IF_SAFER
--orthophoto-cutline          Generates a polygon around the cropping area that cuts
                                the orthophoto around the edges of features. This
                                polygon can be useful for stitching seamless mosaics
                                with multiple overlapping orthophotos. Default: False
--build-overviews            Build orthophoto overviews using gdaladdo.
--verbose, -v                Print additional messages to the console Default:
                                False
--time                        Generates a benchmark file with runtime info Default:
                                False
--version                    Displays version number and exits.
--split <positive integer>
                                Average number of images per submodel. When splitting
                                a large dataset into smaller submodels, images are
                                grouped into clusters. This value regulates the number
                                of images that each cluster should have on average.
--split-overlap <positive integer>
                                Radius of the overlap between submodels. After
                                grouping images into clusters, images that are closer
                                than this radius to a cluster are added to the

```



cluster. This is done to ensure that neighboring submodels overlap.

`--sm-cluster <string>` URL to a nodeodm-proxy instance for distributing a split-merge workflow on multiple nodes in parallel.  
Default: None

`--merge <string>` Choose what to merge in the merge step in a split dataset. By default all available outputs are merged.  
Default: all