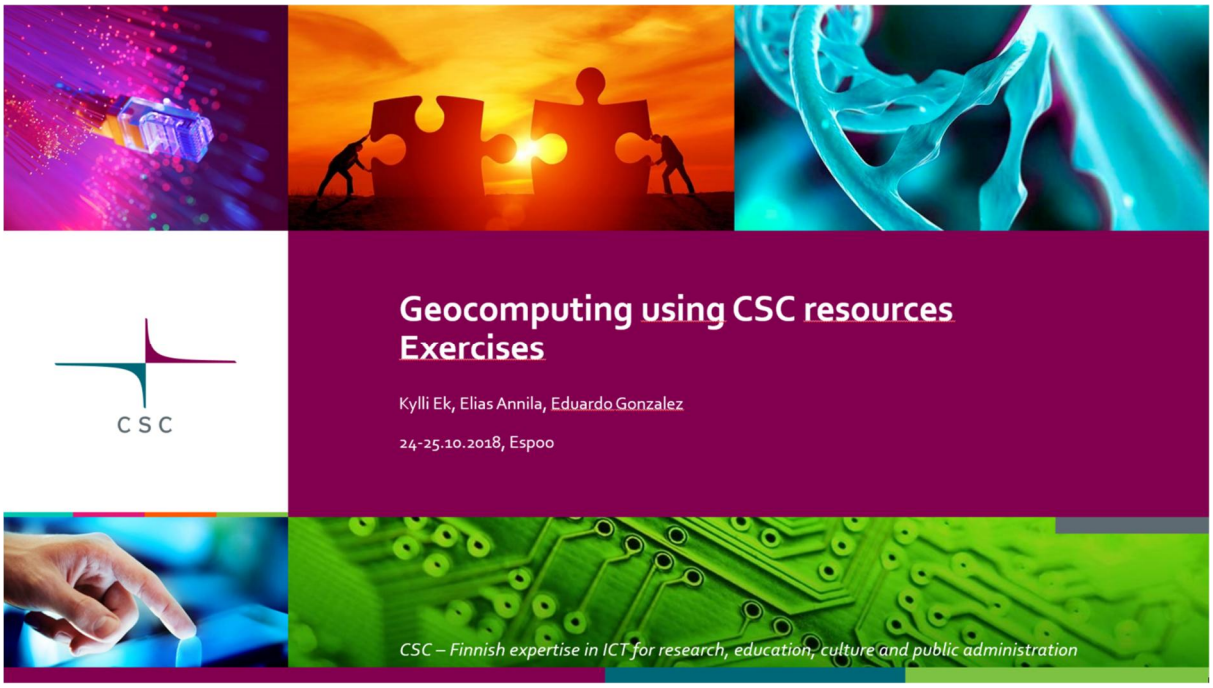

1. Move data to Taito work directory

a) Move course exercise scripts

1) Move the course exercise scripts from your local directory to the "work" directory in Taito: /wrk/trngXX
   - For Windows users we recommend using WinSCP
2) Open a ssh connection to Taito. Go to the working directory and unzip the exercises file:

```
$ cd $WRKDIR
$ unzip exercise-file-name.zip
```

Note. The not bolded dollar sign "$" at the beginning of some lines is not to be typed, but marks the command prompt. What follows is to be typed on the command prompt. Text in *italics* is not a command and you can choose what to put there (but you need to be consistent later).

➢ How many folders are in the training materials?

b) Download data from Internet

You may choose yourself also some other data sources, will not use these files later.

1) Download a single file with wget:

```
$ wget http://wwwd3.ymparisto.fi/d3/gis_data/spesific/valuma.zip .
$ unzip valuma.zip
```

   - Please notice the dot in the end of first row, it means that save the file to the current location, you may give there also some other folder.

2) Download a folder from FTP service with wget:

```
$ wget -r --no-parent -nd
ftp.funet.fi/pub/sci/geo/geodata/mml/hallintorajat_milj_tk/2017/ .
```

   - -r download all files in this folder recursively
   - --no-parent does not download files above the given folder
   - -nd does not make similar folders to Taito

3) Download a lot of data from rsync service with rsync (actually just 1 file in this case):

```
$ rsync -a rsync://rsync.nic.funet.fi/ftp/pub/sci/geo/geodata/mml/orto/etrs-
tm35fin/mara_vv_25000_50/2016/N61/10m/1/ .
```

   - wget always downloads everything, with rsync you can choose files based on name or type.

➢ What was the download speed?
➢ In which folder did you save the downloaded files?

2. Run a R script in Taito-shell interactively

We'll calculate contours based on a GeoTIFF image with RSAGA. Think that you have a R script that is ready on your laptop, and now you would like to run that in Taito.

First we will move your script to Taito and edit it to the new environment. Normally you will to edit at least the paths to files and folders. Then we will run the script interactively from Taito-shell. And finally we do the same/similar calculation as a batch job.

This example script's main tasks are:

- Create folders.
- Change the format of .tif files to SAGA format, because RSAGA accepts only files in that format.
- The file names of the .tif files to be processed are given as list inside the mapsheet.txt file.
- Calculate the contours and save them in Shape format

Open a graphical connection to Taito-shell and review and run the script:

1) Log in to Taito-shell either with your training or CSC user account, either from a terminal (with X11 forwarding) or using NX client.
    - NoMachine installation guide: https://research.csc.fi/csc-guide-connecting-the-servers-of-csc#1.3.3
2) Open RStudio and QGIS from right mouse menu: Applications -> Taito-shell -> GIS
    - Note that rspatial-env / geo-env modules are loaded automatically.
3) In QGIS, open one or a few DEM files to see the data. No need to download data, you can use NLS 10m DEM already available in Taito's GIS data folder: /wrk/project_ogiir-csc/mml/dem10m/etrs-tm35fin-n2000/
4) In RStudio, open the script: ./R/01_serial/Contours_simple.R
5) Use the R console to check that needed R libraries are available in Taito. Which libraries are used in this script? Check whether those libraries are available in RStudio, with (change to correct packages): require(rgdal)
6) In RStudio, change the file paths in the R script to fit your Taito environment.
7) Run your script from RStudio.
8) Results are saved to the 2_shape directory (temporary SAGA format files are saved to 1_grid directory).

3. Simple batch job script in Taito.

Now we will run the previous script as batch job.

1) In RStudio, open batch_job_serial.sh, edit the path to your R script and save the file.
    - Note that you have to load the rspatial-env module
2) Open SSH connection and move to your work directory. Submit the job to Taito

```
$ sbatch batch_job_serial.sh
```

3) See the status of your job, see the ID from output of the previous command

```
$ seff [ID_of_your_job]
```

4) Check your results with QGIS
5) Modify the batch job so, that you would receive an e-mail when the job is finished.

➢ Check with sacct how much memory and time was consumed?

4. Array job

You have another R script skeleton for this exercise, where the for loop from the previous exercise has been remove so that only a single file is calculated.

The idea is that this script will be run by one process for every given input file as opposed to running a for loop within the script. The R script takes the file to be calculated as a parameter, which is defined inside the batch job file. The mapsheets.txt file includes the file names for the files that should be processed.

1) Open the mapsheets.txt. How many files there is?
2) In RStudio, add to the R script the part to the beginning needed for reading in the input file path from the arguments.
3) In RStudio, check the file paths.
4) In RStudio, create a new batch job file by copying the one from previous exercise. Edit it so that:
    a. It will read the mapsheets.txt file contents and give each job one file to process.
    b. Fix the paths to mapsheets.txt file and to your R script.
    c. Pass the file name as a variable
5) Submit the job to Taito
6) Check your results with QGIS

➢ Check with sacct how much memory and time was consumed?

5. Parallel job with snow

You have another R script skeleton for this exercise. We use snow package for multiprocessing.

1) In RStudio, check the file paths in the R script.
2) In RStudio, add to the R script the part for starting and stopping the snow cluster and giving it work, see comments for right places.
3) In gedit/RStudio, make a batch job script. See the R page, how the batch job file for snow has to be written:
4) Submit the job to Taito
5) Check your results with QGIS

➢ Check with sacct how much memory and time was consumed?

Answers:

- Full code and batch job files are in Github: https://github.com/csc-training/geocomputing/tree/master/R

- If you want, you can open a new project to RStudio and clone that repository.

6. Calculate depressions and high points from DEM using Python and rasterio

In this exercise we locate depressions and high points from an elevation model. The way we'll do this is by applying a focal mean to the elevation model and then calculating the difference between the smoothed raster (where each cell contains a mean of surrounding area) and the actual elevation model. This results in a raster where cells that are higher than surrounding cells get a positive value and cells below their neighbors get a negative value.

If we wanted to process large areas split across multiple files we could do this in a couple of ways. In this exercise, you are provided with a serial script that processes a few files and your task is to perform the same task in parallel using array jobs as well as python multiprocessing module.

One advantage of multiprocessing approach over array jobs is that if we wanted to for example combine our resulting files and then do some further processing with the combined file we could easily do this. Also with multiprocessing approach, it would be easy to split one large file into smaller chunks and then process those in parallel rather than operating with multiple files.

Basic idea behind the script is to:

- Read elevation model as a *numpy* array with *rasterio*
- Create a suitable kernel that results in mean of surrounding area being computed
- Apply sliding mean to your elevation array using your kernel
- Subtract the resulting array from array that contains the original elevation model
- Save output with *rasterio*

Above functionality is provided in *high_low()* function in the serial script. This function remains the same for array and parallel jobs and it is sufficient to modify the main() function. There is a for loop in the main() function that processes the files listed in mapsheets.txt which you need to replace with suitable code for array jobs and multiprocessing.

Serial job

Think that you have a Python script that is ready on your laptop, and now you would like to run that in Taito. In this exercise the script is already provided in the exercise zip and your task is to just to get it running as a batch job in taito.

1) Move your data and script to Taito, but that we already did in Exercise 2. Just find the folder again with WinSCP or with shell.
2) Open Spyder and check that needed Python libraries are available in Taito. Which libraries are used in this script? Check that they are available in Taito, with something like (change to correct packages):

   ```
   >>> import gdal
   ```

3) In Spyder, change the file paths in the Python script.

4) In Spyder, make a batch job script.
    - Load the geoconda module
5) Submit the job to Taito
6) Check your results with QGIS

➢ Check with sacct how much memory and time was consumed?

Array job

Use the serial job script as a base for an array job script. The idea is that the batch job script gives each array job one raster file to be calculated. In Python script the file names are read from arguments.

1) In Spyder, replace the for loop from serial job with code that reads a single mapsheet filepath from arguments and calls high_low() function with it.
2) In Spyder, check the file paths.
3) In Spyder, make a batch job script. The batch job script should read the mapsheets.txt file and give each job one file to process. (hint: see https://research.csc.fi/taito-array-jobs or slides on how to use sed to read a row given by SULRM_ARRAY_TASK_ID)
4) Submit the job to Taito
5) Check your results with QGIS

➢ Check with sacct how much memory and time was consumed?

Parallel job with multiprocessing

Use the serial job script as a skeleton for parallel job script. We use multiprocessing package.

1) In Spyder, Modify the script so that instead of using a for loop, you create a multiprocessing pool and map your high_low function to your list of files.
2) In Spyder, check the file paths.
3) In Spyder, make a batch job script, make sure to set number of cpus per task to match number of processes you want to use in your multiprocessing pool.
4) Submit the job to Taito
5) Check your results with QGIS

➢ Check with sacct how much memory and time was consumed?

Answers:

- Full code and batch job files are in example_solutions folder.

7. Calculating TWI with TAUDEM as serial and parallel job

Some applications can be run in parallel to speed them up. In this example you run the TAUDEM software to calculate topographic wetness index TWI in both serial and parallel to see if the jobs speed up.

We are using 10m DEM as input that already is available in Taito.

1) Connect to Taito and create a new *taudem* folder for this exercise.

2) With gedit/RStudio, create a bash script that calculates TWI using TAUDEM commands:

```
#!/bin/bash
pitremove -z /wrk/project_ogiir-csc/mml/dem10m/etrs-tm35fin-n2000/W3/W33/W3333.tif -fel fel.tif
dinfflowdir -fel fel.tif -slp slp.tif -ang ang.tif
areadinf -ang ang.tif -sca sca.tif
twi -sca sca.tif -slp slp.tif -twi twi.tif
```

3) Save the above script to taudem.sh file and make the file executable:

```
chmod 700 taudem.sh
```

Serial job

Let's first run the job with just one core.

Copy one of the old batch scripts to current directory, and change / add the following items in it:

1) Output to out_%j.txt
2) Error to err_%j.txt
3) Run time: 2 minutes
4) Set number of tasks to 1 (#SBATCH -n 1)
5) Load the geo-env and taudem modules
6) Make a new folder for the output files and move to that folder
7) Run commands: srun taudem.sh

Submit the job with:

```
$ sbatch jobscript.sh
```

Submitting the job echoes the SLURM job id number to the screen, but that is also shown in the output and error filenames (out_<SLURM_JOBID>.out). Check if the job is running with

```
$ squeue -u <your username>
```

Or

```
$ squeue -j <SLURM_JOBID>
```

The job should take about 25 seconds to finish. After that it won't appear in the queue.

Once the job is finished, you can check how much memory and time it used:

```
$ sacct -j <SLURM_JOBID> -o elapsed,TotalCPU,reqmem,maxrss,AllocCPUS
```

- elapsed – time used by the job
- TotalCPU – time used by all cores together
- reqmem – amount of requested memory
- maxrss – maximum resident set size of all tasks in job.
- AllocCPUS – how many CPUs were used

➢ Did you reserve a good amount of memory?

Parallel job

Now let's run the same with 4 cores. Change the number of tasks to 4 in the batch job script and change the folder for outputs. Submit the job. Check with the *sacct* command how long it took to run the Taudem job on 4 cores and how did the memory usage change and try to answer these questions:

- Does it make sense to use 4 cores instead of 1?
- Was the memory reservation ok?

Check the efficiency of your jobs with seff. Compare the results of 1 and 4 core jobs.

```
$ seff <SLURM_JOBID>
```

- How efficient was your job?

8. Creating virtual rasters

Create a virtual raster from 2m DEM that covers the area of the provided polygon shapefile (polygon.shp). Use the /proj/ogiir-csc/mml/karttalehtijako/vrt_creator.py script in Taito-shell.

Before running the script, the geo-env module should be loaded. Calling the script with -h flag gives you info on how to use the script.

Use the script to also build overviews for your .vrt file and open it in QGIS to see that you indeed managed to create what you wanted.

- How many tiles does your virtual raster include?

9. Change coordinate system of many files with GDAL.

GDAL/OGR provide many useful tools. In this exercise, we will reproject the coordinate system of multiple files in a folder, and add overviews to the same files. We do not use R nor Python, but a simple Linux bash file.

1) Open the gdal.sh file with gedit/RStudio.
2) Fix the paths and save the file.
3) As there is not many files in the given folder and running the script takes just some seconds, we can run it in Taito-shell.
4) Make sure that you can use GDAL tools, ie. have appropriate modules loaded (you should have from previous exercise). Easiest option is to execute gdalinfo without any parameters, it should provide help how to use it.
   ```
   $ gdalinfo
   ```
5) Check the original file with gdalinfo. What is the coordinate system? Are the files tiled? Do they have overviews?

   ```
   $ gdalinfo /wrk/project_ogiir-csc/mml/dem10m/etrs-tm35fin-n2000/W3/W33/W3333.tif
   ```

6) Change the permissions of gdal.sh, so that it can be executed.

   ```
   $ chmod 700 gdal.sh
   ```

7) Run the script.

```
$ ./gdal.sh
```

➢ Check the result file with gdalinfo. What is the coordinate system? Are the files tiled? Do they have overviews?

If you would have more files you should not use Taito-shell for such work, rather write also a SLURM batch job script and use the same script in Taito.

10. Pouta – Setting up a virtual machine

When you set up a new virtual machine, you are creating a new "cloud computer" with a specific hardware (Pouta flavors) and an operating system (provided by an image you select). The end result is something similar to what your own desktop computer is, just it is running in the cloud and usually does not have a graphical user interface (but you can install one if needed).

Log in to the Pouta web interface

Although there are other ways to manage your Pouta project (command line, Ansible…) you will use the Pouta web interface for this exercise.

- Open a web browser and go to http://pouta.csc.fi
- Log in with your course account

Set up basic access components

Since cloud virtual machines are available via the internet, one very important difference to using a desktop machine is that you must configure different access and security rules.

Before you start creating virtual machines, you need to create these security components:

- a key pair, which plays the role of a password. Note that you can reuse this key pair for any virtual machine you create.
- a security group, which is a set of rules to specify what communication is allowed from which specific IP addresses (as opposed to allowing anyone in the world to attempt breaking into your virtual machine).

Create a key pair

In Access & Security > Key Pairs, click on Create Key Pair

- Name it: *lastname_firstname_key*
- save the key pair file to your local computer

You need to make a few modifications to the key pair to work with it. Follow the instructions for your operating system.

In Windows machines, you will use Putty to connect to your virtual machines. Because the key pair created in Pouta is in Linux format, to use your key in Windows you need to convert it first:

1) Use Puttygen tools to convert your key (if not installed in your computer, download Putty tools from: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)
2) In Puttygen, go to File > Load private key, and load your *lastname_firstname_key.pem* key (note that you have to select All Files (*.*) in the file browse window to see it)
3) Add a Key passphrase and click on Save private key, save as *lastname_firstname_key.ppk*

To use your key in Linux and Mac OS X:

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
```

Create a security group

In Access & Security > Security Groups:

- Click on Create Security Group.
- Name it: *lastname_firstname_SSH.*
- Click on Create Security Group.

You will add security to your future virtual machines by setting your current IP address as the only one allowed to connect to them. This actually means that nobody else will be able to even try to connect to it.

- First, find out your local computer's IP address by opening a new web browser window and going to http://v4.ident.me/, the IP address being shown is the one you will add to your security group.

Now you can modify your security group to accept connections only from your current local computer's IP address:

- Go back to the Pouta web interface.
- Find the security group you created previously and click on Manage Rules on its right.
- Add a new rule with Add Rule, then select SSH from the Rule drop down.
- Leave Remote as CIDR.
- In the CIDR field, you should change the default value (0.0.0.0/0) to the IP address you got above (from v4.ident.me).

Remember that you don't need to redo these steps for every virtual machine you create. You can reuse these key pair and security group for all your virtual machines in your project!

Create a virtual machine

To create a new virtual image, you will use an Ubuntu 16 image (from the existing Pouta images) and the access settings you just created.

Go to the Images page

- Find the image named Ubuntu-16.04 and click on Launch.
- In Details > Instance Name: *lastname_firstname_vm*
- In Flavor, select *standard.tiny*.
- In Security Groups and Key Pair, select the ones you just created.
- You can leave the rest as defaults and click on Launch Instance.

Your instance should be visible in the Instances view, wait until it has started. You can check its details by clicking on the name of the instance.

Connect to your virtual machine

A brand new virtual machine is automatically given a local IP address but to be able to connect to your new instance you need to assign it a public IP address.

Go to the Instances view and:

- Find your VM's name and from the drop down on the right, select Associate Floating IP.
- Select an IP from the drop down (if there are not available IPs, click on the "+" sign).
- You can see the IP you assigned to your VM in the Instances page, next to the name of your virtual machine.
- To connect to the instance, you will use this public-IP address (floating IP) you just assigned.

Important note to log in to VMs created from CSC images:

- the CSC images have only one user by default cloud-user. This user has no password so the only way to connect to this virtual machine is via SSH and using this user.

FROM WINDOWS: use Putty to connect to your VM:

- Open Putty and add the public-ip you assigned to your VM as the Host Name (or IP address).
- Go to Connection > SSH > Auth then in Private key file for authentication add your key pair file in ppk format (*lastname_firstname_key.ppk*).
- You can also connect to your instance with WinSCP for transferring files.

FROM LINUX: use ssh commands to connect to your VM:

```
# use these commands to add your key to your keys archive
```

```
$ ssh-agent /bin/bash
$ ssh-add lastname_firstname_key.pem
$ ssh -A cloud-user@public-ip
```

You can notice that the terminal in your very own virtual machine is very similar to what you have seen when using Taito, which is also a Linux based machine.

Remember that you are the administrator of your own virtual machines and as such: 1) you have full control (and responsibilities) to install and maintain the software; 2) you have the responsibility to configure security and firewalls for connections from the internet to you VM to your hosted services (http, databases, map servers…).

Optional exercise: create a snapshot of your VM

In order to back up a VM state (or to s save billing units when a VM is not used) you can create a snapshot from it, so that you can continue use it to start new virtual machines later.

Create a snapshot

In the Pouta web interface:

- Shut down your instance: Compute > Instances > instance_name > Shut Off Instance
- Then, Compute > Instances > instance_name > Create Snapshot
- Name it: *lastname_firstname_vm_date*
- This creates a new Image in Compute > Image

A snapshot can be thought of a version of your virtual machine that can be launch later on as an instance (this instance will be the same as that you have in the moment of taking the snapshot). The snapshots are stored in Pouta as Images.

To review that the snapshot was created properly:

- Go to Compute > Images
- Click on the name of your image (snapshot) to see its details

If you want to reuse your snapshot to create a new virtual machine:

- Go to Compute > Images
- Find your image snapshot and click on Launch

11. cPouta – Installing GIS software using an existing distribution

The easiest way to get GIS software running on a Pouta VM is to use an existing GIS distribution.

In this exercise you will use a ready-made Pouta image of the popular OSGeoLive open GIS distribution. You can see the steps needed to create the Pouta image from the downloaded OSGeoLive virtual disk from: https://research.csc.fi/osgeolive.

Create an OSGeoLive virtual machine

Create a VM (launch an instance) in the same way you did before, with this differences:

- Go to Images, find the image named osgeolive11 and click on Launch.
- Name it: *lastname_firstname_osgeolive*.
- As Flavor, select *standart.small*.
- In Security Groups and Key Pair, select the ones you created earlier.

This is a large machine so it will take a couple of minutes to create it.

After the virtual machine has been created, associate a floating IP to it, so that you can connect to it later Putty/ssh.

Disclaimer: Note that this is a general distribution not specifically prepared for cloud. You must change the only user's (named *user*) password (by default it is *user*). Some web services installed in the OSGeoLive distribution (such as PostgreSQL or GeoServer) may represent a security risk if you don't change their default passwords and take care of firewalls for your virtual machine.

Connect to your OSGeoLive virtual machine

You can access this virtual machine from the console in the Pouta web interface:

- Go to Instances, then select Console from your VM's dropdown on the right (if you mouse or keyboard is not working in the you might need to click on the gray bar above the console)
- You can use this console normally as a GUI (note that the keyboard layout is in English, so for best compatibility, change your local computer's layout to English too).

IMPORTANT security considerations

Passwords

- Check the passwords in the file passwords.txt (in Desktop)
- Notice that these passwords are known to anyone who uses OSGeoLive. You must change these passwords if you plan to use any of those services.

  Change now at least the password for user (old password is user)

  ```
  $ sudo passwd user
  ```

Security groups

- Note that the Security Groups created in Pouta should be properly set up for your virtual machines.
- Security groups work together with your SSH keys and the services passwords to keep your machine from being accessed by unauthorized people.
- Ask your IT support for advice!

Enable SSH service in your OSGeoLive virtual machine

You need to install SSH service to be able to connect via ssh. To set it up:

Open a terminal via the Pouta web interface console and run this code:

```
$ sudo apt update
$ sudo apt install openssh-server
```

Now you can connect to your virtual machine with Putty/ssh as you have done in previous exercises, but use the user name user instead of cloud-user.

Make sure you have read the IMPORTANT security considerations part above!

Mounting the share GIS data folder to your OSGeoLive virtual machine

You can mount a Taito folder to your VM using a SSH file system mount. Follow these steps:

- Connect to your machine using Putty/ssh as you did before (or continue using the terminal via the Pouta web interface).
- See instructions here: https://research.csc.fi/csc-guide-remote-disk-mounts
  - The folder in Taito is: /proj/ogiir-csc/
  - Use your training credentials (trngXX)
  - For ex:

```
$ mkdir ~/gisdata
$ sudo apt install sshfs
$ sshfs trng27@taito.csc.fi:/proj/ogiir-csc/ ~/gisdata
```

- Note that you can connect to your own personal/project folders in the same way.

Now you can access those files from your OSgeoLive virtual machine.

Use the graphical interface form the Pouta web interface to open QGIS to view some files you just mounted from Taito:

- Open QGIS from the lower left icon in the OSGeoLive Desktop > Geospatial > Desktop GIS > QGIS
- Then open a raster layer with Layer > Add Layer > Add Raster Layer…
- Browse to the folder where you mounted the Taito disk and open for example some files from the mml/dem2m/ folder.

Optional exercise - Test some of the GIS tutorials in OSGeoLive

You can test some of the GIS tutorials in OSGeoLive. They are a great way to get to know different software.

- Open the Firefox browser, the homepage is set to the OSGeoLive documentation
- Go to Contents
- Select a tutorial and test how different software are running in your VM

Optional exercise – Install GeoServer in cPouta

If you would like to set a basic GeoServer installation in cPouta, you can follow our instructions in our Github repository.

Go to: https://github.com/csc-training/geocomputing/blob/master/pouta/geoserver/basic_geoserver_jetty.md

There you can find some general details to the type of virtual machine you would need plus some details about the security details you need to take care of. Finally, you can find the installation instructions needed to get a GeoServer up and running.

12. BONUS exercises for Taito

Copying files to hpc_archive

You can access hpc_archive only from Taito and Sisu. You can access IDA also from your computer after installing the required tools.

1) Log in to Taito. Check the contents of your hpc_archive:

```
$ ils
```

2) Show the directory that you're in in hpc_archive:

```
$ ipwd
```

3) Show the directory that you're in in taito:

```
$ pwd
```

4) Create a directory in hpc_archive

```
$ imkdir test
```

5) Move to test directory in hpc_archive

```
$ icd test
```

6) Confirm where you are in hpc_archive

```
$ ipwd
```

7) Copy (put) a file to the test directory in hpc_archive:

```
$ iput <filename>
```

8) Confirm that the file is in hpc_archive

```
$ ils
```

9) Copy the file back from hpc_archive, but to a local folder called *localtest*

```
$ mkdir localtest
$ cd localtest
```

```
$ iget <filename>
```

Archive a file

Make a new directory in hpc_archive home directory (not under test) called *mysafe*. Copy there your files, e.g., *test_hostname.sh* and *R_array.sh* from previous exercises. Compress and archive a file, e.g., *fit.R* using a command tar -zcvf , and copy it to *mysafe*.

GRASS GIS in batch jobs through a python script.

Why would you want to use GRASS through Python you may ask? Well, GRASS provides a wide variety of raster and vector tools and the Python scripting interface also allows you to run these tools in serial or parallel fairly easily. GRASS also provides a graphical model builder that is quite handy tool for creating python code that utilizes GRASS.

1) Load modules needed for GRASS:
2) Launch grass in gui mode and set up a location and a mapset:
    - Start grass with command: grass72
    - You will see a window prompting you to select GRASS GIS database directory. Set this to /wrk/USERNAME/grass_db (create if doesn't exist).
    - Create a new location. Set GIS Data Directory to grass_db folder you just created and give name for the location (can be anything). When asked to choose a method for creating a new location choose the "Read projection and datum terms from georeferenced file" option and select the vrt file you just created in the last exercise. After this you will be asked if you want to import your vrt, if you'd like to set default region and if you'd like to create a mapset. Answer no to all as we don't want to import our vrt with r.in.gdal but rather link it with r.external, setting region is easier later on and PERMANENT mapset is fine for us.
    - After this you can launch grass gui.
3) Link the vrt file you created in the previous exercise to GRASS raster map with r.external
    - file->link external data-> link external raster data
4) Set computational region to match your raster with g.region
    - settings->region->set region
    - select your vrt file from set region to match raster map menu.

    If you prefer to use commands (write to terminal you started GRASS from), for steps 3 and 4 you can do so. Relevant commands are:

```
r.external input=your_vrt output=name_of_grass_raster_map
g.region raster=name_of_grass_raster_map
```

5) Open GRASS graphical modeler and create a simple model that performs a raster analysis of your choice to your linked raster map. A couple options would be to create a hillshade or calculate contours (or both) (note: r.viewshed analysis has a bug in it so it won't work with

external rasters). Also export your resulting grass rasters to tiff format by adding r.out.gdal tool to your model. Export your model as python code.

- Note: GRASS doesn't store raster and vector data in tiffs or shapefiles. rather the data is stored in your grass_db folder split across multiple files. For this reason you shouldn't try to give file path as output to grass tools (besides r.out.*, or v.out.*) and instead simply give a name for the map like so output=hillshade_map.

The Python code should look something like this:

```
#First import grass.script
import grass.script as gscript

#Then grass commands can be called with run_command function
gscript.run_command('g.region', raster='your_raster_map')
```

If you want to, you can modify the Python script. If (when) you don't know what parameters the command you're running needs (or what that command might be), you can either refer to documentation at , or check parameters from GRASS gui.

You can test your python script from within grass either by typing python your_saved_script.py in the grass console or by hitting play button in the graphical modeller.

6) Make batch job file and run your saved Python script as a serial job.

   Start grass with --exec option ie.

   ```
   grass72 --exec python your_script.py args.
   ```

   This launches grass without ui and executes your python script from within grass.

It is possible to run GRASS functions with plain Python if you set up grass environment in your Python script. We're not going to do it in this exercise, but it's nice to know you can do it.

See https://grass.osgeo.org/grass70/manuals/libpython/script.html#module-script.setup

- Do you know that also QGIS and ArcGIS Desktop have similar graphical model builder?

BONUS: Make your Python script run a couple of raster analysis in parallel using Array jobs or ParallelModuleQueue from pygrass.

See:
https://grass.osgeo.org/grass73/manuals/libpython/pygrass.modules.interface.html?highlight=parallelmodulequeue#pygrass.modules.interface.module.ParallelModuleQueue

Links to important support pages

General

- User accounts: https://research.csc.fi/accounts-and-projects
- Trainings, inc materials of past events: https://www.csc.fi/web/training
- Geocomputing: https://research.csc.fi/geocomputing
- Virtual rasters: https://research.csc.fi/virtual_rasters
- Linux: https://research.csc.fi/csc-guide-linux-basics-for-csc
    - Unix cheat sheet: https://research.csc.fi/csc-guide-appendixes

Taito

- Taito user guide: https://research.csc.fi/taito-user-guide
- NoMachine: https://research.csc.fi/csc-guide-connecting-the-servers-of-csc#1.3.3
- Directories and data storage: https://research.csc.fi/csc-guide-directories-and-data-storage-at-csc
- Batch jobs: https://research.csc.fi/taito-batch-jobs
- Software (and modules): https://research.csc.fi/software -> Geosciences
    - geo-env: https://research.csc.fi/-/geo-env
    - rspatial-env: https://research.csc.fi/-/rspatial-env
    - GDAL: https://research.csc.fi/-/gdal-o-1
    - GeoPython: https://research.csc.fi/-/geopython
    - GRASS: https://research.csc.fi/-/grass-gis
    - LasTools: https://research.csc.fi/-/lastools
    - QGIS: https://research.csc.fi/-/qgis
    - SagaGIS: https://research.csc.fi/-/saga-gis
    - Taudem: https://research.csc.fi/-/taudem
    - proj4: https://research.csc.fi/-/proj4
- Taito GIS data: https://research.csc.fi/gis_data_in_taito
- Code examples for geocomputing: https://github.com/csc-training/geocomputing

cPouta

- cPouta user guide: https://research.csc.fi/pouta-user-guide
- OSGeoLive installation guide to cPouta: https://research.csc.fi/osgeolive
- ArcPy installation guide to cPouta: https://research.csc.fi/arcpy-in-pouta