Spider user documentation

Release 0.1

SURFsara support team

Jul 07, 2020

Contents

1 About 3

2 Getting started 9

3 Storage on Spider 15

4 Software on Spider 21

5 Compute on Spider 25

6 Workflows 31

7 Web public view 35

8 Jupyter Notebooks 37

9 Maintenances 39

10 Getting help 41

11 Documentation how-to 43

12 Cookiebeleid 49

Welcome to the SPIDER (Symbiotic Platform(s) for Interoperable Data Extraction and Redistribution) user guide! SPIDER is a feature-rich platform based at SURF and tailored for data processing and collaboration.

This guide aims to help new users getting started on SPIDER, while it also serves a useful reference for existing SPIDER members. Whether you seek information for SPIDER in general, or how to access and use the available features, or best practices for the efficient use of the resources, read on!

Table of contents:

CHAPTER 1

About

Tip: We welcome you to our new platform! In this page you will learn:

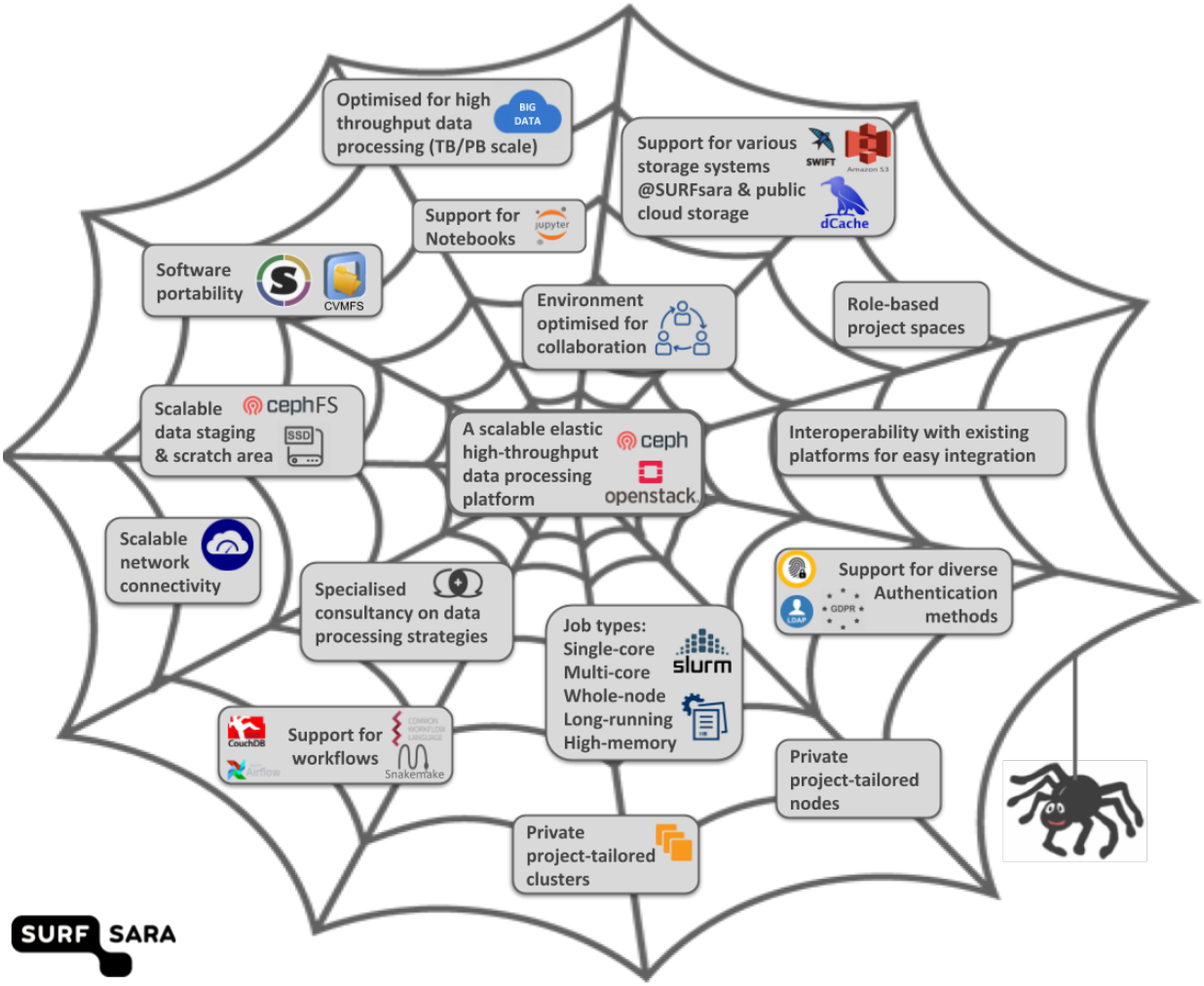
- what SPIDER is all about
- whether it is suitable for your research project
- what are the collaboration options
- how to obtain access and work within a SPIDER project

1.1 Spider at a glance

SPIDER is a versatile high-throughput data-processing platform aimed at processing large structured data sets. It runs on top of our in-house elastic Cloud. This allows for processing on SPIDER to scale from many terabytes to even petabytes. Utilizing many hundreds of cores simultaneously SPIDER can process these datasets in exceedingly short timespans. Superb network throughput ensures connectivity to external data storage systems.

Apart from scaling and capacity, SPIDER is aimed at interoperability with other platforms. This interoperability allows for a high degree of integration and customization into the user domain. SPIDER is further enhanced by specific features supporting collaboration, data (re)distribution, private resources or even private SPIDER instances.

Have a glance here of the main features offered by SPIDER:



1.2 Platform components

SPIDER is a feature-rich platform under continuous development. We keep adding new features to the platform in order to meet the needs of researcher projects working with massive datasets.

SPIDER is built on powerful infrastructure and designed with the following components:

- Batch processing cluster (based on Slurm) for generic data processing applications
- Batch partitions to enable Single-core, Multi-core, Whole-node, High-memory and Long-running jobs
- Large CephFs data staging area (POSIX-compliant filesystem) scales to PBs without loss of performance or stability
- Large and fast scratch area's (NVMe SSDs) on the worker nodes
- Fast network uplink (1200 Gbit/s) allowing for scalable parallel data transfers from other SURFsara based storage systems (e.g. dCache, SWIFT), or from external storage systems
- Role-based project spaces tailored for data-centric projects
- Scientific catalogs for cross-project collaboration
- Web access over HTTPS for public data distribution and sharing with external collaborators

- Singularity containers for software portability
- CVMFS/Softdrive support for software distribution
- Jupyter Notebooks
- Interactive jobs and direct visualization from within jobs
- Specific tooling for data-processing workflows
- Workflow management support
- Diverse authentication methods
- Private resources for special purposes (reservations, private nodes, private clusters)

1.3 Best suited cases

The best-suited cases for SPIDER are scientific projects with a requirement to process relatively large data sets. For example research projects suitable for SPIDER that deal with massive datasets are commonly in: Genomics, Proteomics, Earth observation, Astronomical observation, Climate modeling, Engineering or Physics experiments.

You would be eligible for SPIDER if your project reflects some of the following needs:

- Processing of large amount of data of many terabytes to petabytes in short time spans
- Processing of large amount of independent simulations and workflows
- Interactive processing with user-friendly interfaces for efficient data handling
- Industry standard interfaces and other interoperability features
- Co-working with your collaborators on the same project-based workspace
- Accessing external storage facilities with fast connectivity

Also SPIDER is a viable alternative for current and potential Grid users who are looking to use a more customizable system. It is a low-threshold platform, as opposed to highly complex Grid platforms that take many months of specialist development before they can start. Being built upon the exact same physical data-processing infrastructure and sharing the same scalable network connectivity as the Grid-based processing environments, SPIDER offers the same data-parallel processing capabilities as the most powerful Grid platforms.

Note though that while it's great for data-intensive applications, SPIDER is *not* really aimed at:

- HPC applications where operations per second are critical
- Processing of simulations that require multi-node execution
- Applications that cannot be ported onto Linux-based system

1.4 Collaboration

SPIDER is designed for Big Science which requires collaboration. SPIDER supports several ways to collaborate, either within your project, across projects, or to external sources.

1.4.1 Project space

Project spaces on SPIDER are shared workspaces given to team members that enable collaboration through sharing data, software and workflows. Within your project space there are four folders:

- Data: Housing source data from data managers
- Share: For sharing between project members
- Public: For sharing publicly through webviews
- Software: Scripts, libraries and tools

SPIDER enables collaboration for your project with granular access control to your project space through project roles, enabling collaboration for any team structure:

- *technical lead* role: the contact person for any technical matters that affect the design and execution of the project and the privileges of other members
- *data manager* role: designated data dissemination manager; responsible for the management of project-owned data
- *software manager* role: designated software manager; responsible to install and maintain the project-owned software
- *normal user* role: scientific users who focus on their data analysis

1.4.2 Scientific catalog

Collaboration is also possible across different SPIDER projects. These are cases where different user groups work on projects with different scope and goals but need to (partly) share read-only data (such as observations or biobank data). SPIDER offers a place for multiple project teams to collaborate by sharing data sets or tools. This workspace is called *scientific catalog* and it is *not* offered by default to a project.

The scientific catalog data can be either *open* to everyone on the platform or *private* to selected SPIDER project groups.

The scientific catalog has only one (but important) role:

- *scientific catalog manager*: designated data dissemination SC (scientific catalog) manager; responsible for populating the catalog and deciding which SPIDER project groups have read access to that catalog.

1.4.3 Interoperability hotspot

In contrast to many of the processing platforms already available, typically offering an all-inclusive solution within the boundaries of the their environment, SPIDER is exactly the opposite. It aims to be a connecting platform in a world that has already a lot to offer in terms of storage systems, data distribution and collaboration frameworks, software management and portability systems, and pilot job and task management frameworks. The SPIDER platform can hook them all together as an interoperability hotspot to support a variety of data processing and data collaboration use cases.

For all external services supported, even services owned by the users themselves, SPIDER offers optimized configurations and practical guidelines how to connect to these services together into a practical processing environment tailored specifically to each project.

1.5 Project lifecycle

If you decided that SPIDER sounds suitable for your research project, then you can apply to obtain access and start your project or join an existing one.

1.5.1 Starting a project

For information about the granting routes on SPIDER please see our Proposals Page.

Before applying for a new project on SPIDER we suggest you to contact *our helpdesk* to discuss your project.

1.5.2 Extending a project

You can apply for a time or resource capacity extension for an existing project on SPIDER by requesting extra resources. Please see our Proposals Page or contact *our helpdesk*.

1.5.3 Joining an existing project

If you are interested to join an existing project please contact our *our helpdesk*. Upon your request we will verify with the project PI whether we can give you access to the project and what your project role would be.

1.5.4 Ending a project

Once your project ends, all the relevant data and accounts will be removed according to the Usage Agreement terms and conditions.

See also:

Still need help? Contact *our helpdesk*

CHAPTER 2

Getting started

Tip: This is a quickstart on the platform. In this page you will learn:

- How to login Spider
- Browsing your project space
- Submitting simple jobs

2.1 Setting up your account

Access to the cluster is provided via SSH (Secure Shell) Public key authentication only. For the highest security of your data and the platform, we do not allow username/password authentication.

To use this method you will need first to configure your SSH public key on a portal provided by SURFsara. Then you can connect and authenticate to SPIDER with your SSH keys without supplying your username or password at each visit.

Please follow these steps to access SPIDER:

- **Step 1**: Login to the SURFsara portal with your SURFsara user account

As a member of a SPIDER project you shall have received a SURFsara user account. Please use the username and password sent to you and login to the SURFsara portal .

- **Step 2**: Accept the Usage Agreement in the portal

Once you login to the portal please agree to our usage terms and conditions to be able to gain access to SPIDER. You can perform this action on the "Usage Agreement" tab as shown in the image below. Please note that you will be denied access to SPIDER if you do not accept this agreement.

- **Step 3**: Upload your SSH public key to the portal

In order to access SPIDER you need to have a file on your local computer (say, your laptop) with a private SSH key, and you need to upload its matching public SSH key on the SURFsara portal. Then, when you are going to connect to SPIDER from your laptop, the SSH private and public keys will be compared and, if they successfully relate to one-another, your connection will be established. Note that uploading your key to the portal is an *one time* task.

If you already have an SSH key-pair you can proceed to upload it. Else you have to generate a key-pair in your laptop or other machine that you use to connect to SPIDER. If you need help to generate an SSH key-pair, see:

2.1.1 Generate a private/public key pair

Open a terminal and type the following command:

```
ssh-keygen
```

While interacting with *ssh-keygen* in the terminal, you will see something similar to:

```
Generating public/private rsa key pair.
Enter file in which to save the key (~/.ssh/id_rsa): ### see note 1 below
Enter passphrase (empty for no passphrase): ### see note 2 below
Enter same passphrase again:
Your identification has been saved in ~/.ssh/id_rsa
Your public key has been saved in ~/.ssh/id_rsa.pub
The key fingerprint is:
40:1f:33:78:32:51:b5:c4:51:56:99:b6:6a:3d:18:8b user@computer.surfsara.nl
The key s randomart image is:
+---[RSA 2048]----+
|                 |
|                ..|
|       .       .o.|
|        + +.o  o+ |
|       + S.B.o.+.o|
|        E =++oooo+|
|       . =oo= ++.=|
```

(continues on next page)

(continued from previous page)

```
|        *....Bo = |
|       o.o..+o..  |
+----[SHA256]-----+
```

Some notes:

[1]. You can leave the output file name blank (simply hit enter) for the default file name, or type a variation of *~/.ssh/my_chosen_name*.

[2]. You can leave the passphrase field empty but we strongly recommend you to choose and remember an easy but long passphrase. If you forget the passphrase, you need to generate a new key pair and replace the old public keys you installed on remote hosts.

The ssh-agent (see below) will assist you so that you only need to type the passphrase once per session.

Using an SSH agent

SSH-agent is a service on your computer to remember your ssh passphrase during your local session (that is, until you log out). This way, you do not have to type in that loooong passphrase every time you need to unlock your private key.

- Add the key to the local ssh-agent

Type the following on your terminal:

```
ssh-add ~/.ssh/id_rsa ### or the file name you provided to ssh-keygen
```

- While interacting with *ssh-add* in the terminal, you will see something similar to:

```
Enter passphrase for ~/.ssh/id_rsa: ### type it in
Identity added: ~/.ssh/id_rsa
```

If this fails because "Could not open a connection to your authentication agent", you need to start the ssh-agent daemon before you run *ssh-add*:

```
eval `ssh-agent -s`
```

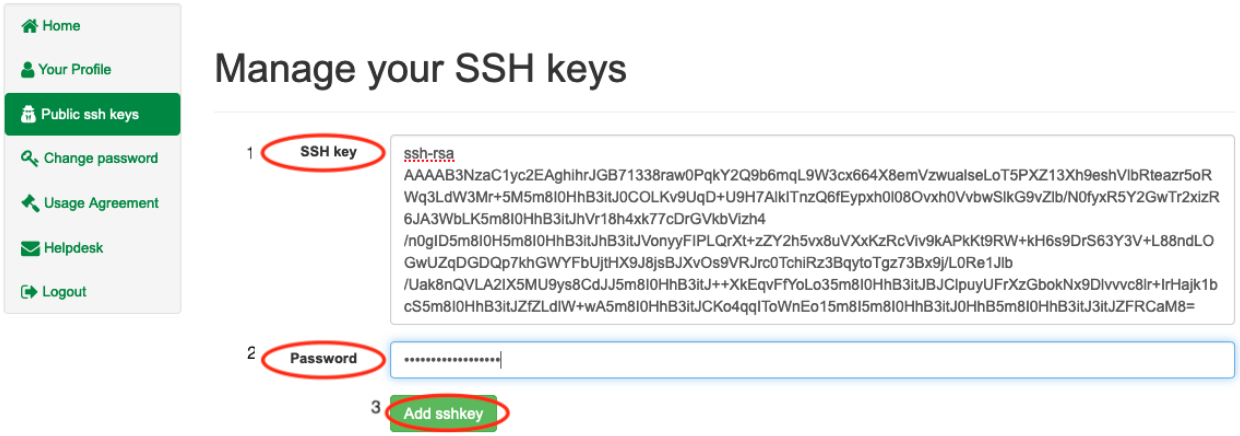
- To list the keys loaded in the *ssh-agent* type the following on your terminal:

```
ssh-add -l
```

The output will be one line for each key stored in the ssh-agent, similar to:

```
2048 SHA256:ajAxT3T3ZKl2rALBGGmMqufU0n6XAU15lj+fObZEvrI ~/.ssh/id_rsa (RSA)
```

Once you have generated your SSH key-pair, upload your public key to our SURFsara portal. Click on the tab "Public ssh keys" on the left pane of the portal and add your public key by copying the contents of your public key file (e.g. `cat ~/.ssh/id_rsa.pub`) as shown below:



Field [1] SSH key: here you paste your public key

Field [2] Password: here you enter your password for your account

Field [3] Add sshkey: press the key for the changes to take effect

From now on you can login to SPIDER with your SSH keys from your laptop (or other computer where your SSH key was generated/transferred). See next, *how to login*.

2.2 Logging in

The login node is your entry and access point to SPIDER. From this node you can submit jobs, transfer data and prototype your application. It has a software environment very similar to the worker nodes where your submitted jobs will run.

In order to login to SPIDER you must have already uploaded your SSH public key on the SURFsara portal (see *Setting up your account*)

If you already completed this step once, you are ready to login!

- Login to SPIDER via a terminal with the following command:

```
ssh [USERNAME]@[Spider UI HOSTNAME]
```

- For example, the user *homer* will login as:

```
ssh homer@spider.surfsara.nl
#[homer@htp-ui ~]$  # this is the first prompt upon login
```

Congrats! You've just logged in to Spider.

Note: In case that you have multiple keys in your `.ssh/` folder, you would need to specify the key that matches the .pub file you uploaded on the SURFsara portal, i.e. `ssh -i ~/.ssh/surfsarakey homer@spider.surfsara.nl`

2.3 Getting around

As a user on SPIDER you are a member of a project, and each project member gets access to the following directories:

2.3.1 Home directory

- `/home/$USER`: each project member in a project has her/his personal home space. Only the account owner can read and write data in this directory

2.3.2 Project spaces directories

Project space is a POSIX storage place allocated to each SPIDER project. It includes the following shares:

- `/project/[PROJECTNAME]/Data`: any project-specific data. Any member of the project can read data in this directory, but only the data manager(s) can write data
- `/project/[PROJECTNAME]/Software`: any project-specific software. Any member of the project can read/execute software in this directory, but only the software manager(s) can install software
- `/project/[PROJECTNAME]/Share`: any data to be shared among the project members. Any member of the project can read and write data in this directory
- `/project/[PROJECTNAME]/Public`: Any member of the project can write in this directory. Any data stored here will be read-only by all users on SPIDER and exposed publicly via http (see *how*)

The summary table below gives a quick overview of your project space permissions ('r'-read/'w'-write/'x'-execute):

Directories vs. Access Roles	`/project/[PROJECTNAME]/Data`	`/project/[PROJECTNAME]/Software`	`/project/[PROJECTNAME]/Share`	`/project/[PROJECTNAME]/Public`
Project Data manager(s)	rwx	r-x	rwx	rwx
Project Software manager(s)	r-x	rwx	rwx	rwx
Project normal user(s)	r-x	r-x	rwx	rwx
Other Spider project user	—	—	—	r–
Outside Spider user	—	—	—	r– (via the *web views*)

2.4 Submitting a job

On SPIDER you will use Slurm to schedule, manage and execute your jobs. Slurm (short for Simple Linux Utility for Resource Management) is an open source, fault-tolerant, highly scalable, cluster management and job scheduling system for Linux clusters. Further information can be found at the Slurm documentation page. You can see the currently installed version of Slurm by typing `sinfo --version` on the command line.

Let's run our first job on SPIDER. Download the sample job script to your `home` account on the SPIDER login and inspect the file before submitting it to the cluster:

```
wget https://raw.githubusercontent.com/sara-nl/spiderdocs/master/source/scripts/
↪welcome-to-spider.sh
chmod u+x welcome-to-spider.sh
```

- Submit your job to the cluster:

```
sbatch welcome-to-spider.sh
#Submitted batch job [JOBID]
```

- Check the status of your submitted and not completed job(s):

```
squeue --job [JOBID] # status of [JOBID]
squeue -u $USER # status of all my jobs
scontrol show jobid [JobID] # detailed info of [JOBID]
```

- Check your job output:

```
cat slurm-[JOBID].out
```

- Once your job has completed, you can get job statistics and accounting:

```
sacct -j [JOBID] --format=JobID,JobName,AveCPU,MaxRSS,Elapsed
```

More examples of how to use SPIDER Slurm can be found in section *Compute on Spider* and more generic info can be found at the Slurm documentation page.

2.5 Common commands

Slurm has many commands with many options, here you have a list with the most common ones. For more information please checkout the Slurm documentation page.

Command	What it does
`sinfo`	displays the nodes information
`sbatch`	submits a job to the batch system
`squeue`	displays the state of all submitted jobs
`scancel`	cancels a submitted job
`scontrol`	shows detailed job information (useful for debugging)
`sacct`	shows detailed accounting information for jobs

See also:

Still need help? Contact *our helpdesk*

CHAPTER 3

Storage on Spider

Tip: Spider is meant for processing of big data, thus it supports several storage backends. In this page you will learn:

- which internal and external storage systems we support
- best practices to manage your data

3.1 Internal storage

3.1.1 Transfers within Spider

To transfer data between directories located within SPIDER we advise you to use the unix commands `cp` and `rsync`. Other options may be available, but these are currently not supported by us.

Help on these commands can be found by (i) typing `man cp` or `man rsync` on the command line after logging into the system, or (ii) by contacting *our helpdesk*.

3.1.2 Spider filesystems

Using Home

SPIDER provides to each user with a globally mounted home directory that is listed as `/home/[USERNAME]`. This directory is accessible from all nodes. This is also the directory that you as a user will find yourself in *upon first login* into this system. The data stored in the home folder will remain available for the duration of your project.

Using scratch

Each of SPIDER worker nodes has a large scratch area on local SSD. These scratch directories enable particularly efficient data I/O for large data processing pipelines that can be split up into many parallel independent jobs.

Please note that you should only use the scratch space to temporarily store and process data for the duration of the submitted job. The scratch space is cleaned regularly in an automatic fashion and hence can not used for long term storage.

Using project spaces

Similarly to home folders Spider's project spaces are also available on all worked nodes, the following paths are available on your SPIDER UI:

- `/project/[Project Name]/Data`
- `/project/[Project Name]/Public`
- `/project/[Project Name]/Share`
- `/project/[Project Name]/Software`

This allows you to easily access your software, data and output from the worker nodes from the project spaces. See below for an example of a command that could be executed from a script on a worker node:

```
sh /project/[Project Name]/Software/[script].sh /project/[Project Name]/Data/[input␣
↪file(s)] /home/[USER]/[output]
```

Using scientific catalogs

Scientific catalogs allow for you to share software and data repositories accross projects. For example if you would like to share a large biobank of data with other research projects you could request access to upload to the scientific catalogue. Then it will be accessible from the worker nodes similarly to the `/home` and `/project` folders.

To request access to add a shared catalogue please reachout to *our helpdesk*.

3.2 External storage

3.2.1 Transfers from own system

If you are logged in as a user on SPIDER then we support `scp`, `rsync`, `curl` or `wget` to transfer data between SPIDER and your own Unix-based system. Other options may be available, but these are currently not supported by us.

- Example of transferring data from SPIDER to your own Unix-based system:

```
scp /home/[USERNAME]/transferdata.tar.gz [own-system-user]@own_system.nl:/home/[own-
↪system-user]/
rsync -a -W /home/[USERNAME]/transferdata.tar.gz [own-system-user]@own_system.nl:/
↪home/[own-system-user]/
```

- Example of retrieving data from own Unix-based system on Spider:

```
scp [own-system-user]@own_system.nl:/home/[own-system-user]/transferdata.tar.gz /home/
↪[USERNAME]/
rsync -a -W [own-system-user]@own_system.nl:/home/[own-system-user]/transferdata.tar.
↪gz /home/[USERNAME]/
```

3.2.2 SURFsara dCache

dCache is our large scalable storage system for storing and processing huge volumes of data fast. The system runs on dCache software that is designed for managing scientific data. You can use dCache for disk or tape or address both types of storage under a single virtual filesystem tree. Our dCache service is a remote storage with extremely fast network link to Spider. You may use the storage if your data does not fit within the storage allocation on SPIDER project space or if your application is I/O intensive.

There are several protocols and storage clients to interact with dCache. On SPIDER we support two main methods to use dCache, ADA and Grid interfaces:

ADA interface

Coming soon ..

Our ADA (Advanced dCache API) interface is based on the dCache API and the webdav protocol to access and process your data on dCache from any platform and with various authentication methods.

Grid interface

The Grid interface is recommended in cases that your project data and/or processing is tight to the Grid authentication and authorisation. To use the supported Grid clients on SPIDER you need to have an X509 Grid certificate installed into your .globus directory and be a part of a Virtual Organisation (VO). Please refer to our Grid documentation page for instructions on how to get a certificate and join a (VO).

Grid authentication

To be able to interact with dCache using a storage client, you need to create a proxy. A proxy is a short-lived certificate/private key combination which is used to perform actions on your behalf without using passwords.

Create a proxy with the following command:

```
voms-proxy-init --voms [YOUR_VO]
```

On SPIDER your proxy is generated by default in your `$HOME/.proxy` location such that it is accessible from anywhere on Spider. You can check this with `echo X509_USER_PROXY`.

The default lifetime of a proxy is 12h. If your application runs longer then you can create a proxy that is valid up to 7 days with the following command:

```
voms-proxy-init --voms [YOUR_VO] --valid 168:00
```

Grid clients

There are many Grid clients to interact with dCache. On SPIDER we support `globus-url-copy` and `gfal`.

In the examples below, a user who is a member of the VO e.g., lsgrid, has the certificate installed on to the SPIDER login node and will copy data from dCache to/from your home directory on Spider.

Globus client

Please note that you need a valid proxy to run the following commands.

- Listing directories on dCache:

```
globus-url-copy -list gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/
↪lsgrid/
```

- Copy file from dCache to Spider:

```
globus-url-copy \
    gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-your-
↪data/your-data.tar \
    file:///`pwd`/your-data.tar
```

- Copy file from SPIDER to dCache:

```
globus-url-copy \
    file:///$HOME/your-data.tar \
    gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-your-
↪data/your-data.tar
```

- Copy directory from dCache to Spider:

  First create the directory locally, e.g. testdir.

```
globus-url-copy -cd -r \
 gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-
↪your-data/testdir/ \
 file:///$HOME/testdir/
```

The `globus-*` client does not offer an option to create/delete directories or delete files. For this purpose you may use the gfal client as described below.

gfal client

Please note that you need a valid proxy as described above to run the following commands.

- Listing directories on dCache:

```
gfal-ls -l gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/
```

- Create directory on dCache:

```
gfal-mkdir gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-
↪your-data/newdir/
```

- Copy file from dCache to Spider:

```
gfal-copy \
    gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-your-
↪data/your-data.tar \
    file:///`pwd`/your-data.tar
```

- Copy file from SPIDER to dCache:

```
gfal-copy \
    file:///$HOME/your-data.tar \
    gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-your-
↪data/your-data.tar
```

- Remove a file from dCache:

```
gfal-rm gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-your-
↪data/your-data.tar
```

- Remove a whole (non empty) directory from dCache:

```
gfal-rm -r gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/path-to-
↪your-data/
```

Recursive transfer of files (transferring a directory) is not supported with the gfal-copy command. For this purpose you may use globus-url-copy.

Grid data processing

Below we show an example for I/O intensive applications. In this example you submit a job on SPIDER that performs the following steps:

- Creates a runtime directory on local `scratch` (or `$TMPDIR`)
- Retrieves the input data from dCache
- Runs the analysis
- Stores the output data on dCache

Here is a job script template for local `scratch` usage;

```
#!/bin/bash
#SBATCH -N 1       #request 1 node
#SBATCH -c 1       #request 1 core and 8GB RAM
#SBATCH -t 5:00    #request 5 minutes jobs slot

mkdir "$TMPDIR"/myanalysis
cd "$TMPDIR"/myanalysis
gfal-copy gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/path-to-your-data/
↪your-data.tar file:///`pwd`/your-data.tar

# = Run you analysis here =

#when done, copy the output to dCache
tar cf output.tar output/
gfal-copy file:///`pwd`/output.tar gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.
↪nl/data/path-to-your-data/output.tar
echo "SUCCESS"
exit 0
```

Please note that in the above example, it is assumed that the data is present on the disk storage on dCache. If the data is stored on Tape, it may need to be copied to disk first (called as staging).

Our Grid interface is based on the Grid computing technology and the gridftp protocol to access and process your data on dCache from Grid compliant platforms and with X509 certificate authentication.

3.2.3 SURFsara SWIFT

Coming soon ..

3.2.4 SURFsara Central archive

For long-term preservation of precious data SURFsara offers the Data Archive. Data ingested into the Data Archive is kept in two different tape libraries at two different locations in The Netherlands. The Data Archive is connected to all compute infrastructures, including SPIDER.

Access on Data Archive is *not* provided by default to the SPIDER projects. To request for Data Archive access, please contact our *our helpdesk*.

If you already have access on Data Archive, then you can use it directly from SPIDER by using `scp` and `rsync` to transfer data between SPIDER and Data Archive:

- Transfer data from SPIDER to Data Archive:

```
scp /home/[USERNAME]/transferdata.tar.gz [ARCHIVE_USERNAME]@archive.surfsara.nl:/home/
↪[ARCHIVE_USERNAME]/
rsync -a -W /home/[USERNAME]/transferdata.tar.gz [ARCHIVE_USERNAME]@archive.surfsara.
↪nl:/home/[ARCHIVE_USERNAME]/
```

- Retrieve data from Data Archive on Spider:

```
scp [ARCHIVE_USERNAME]@archive.surfsara.nl:/home/[ARCHIVE_USERNAME]/transferdata.tar.
↪gz /home/[USERNAME]/
rsync -a -W [ARCHIVE_USERNAME]@archive.surfsara.nl:/home/[ARCHIVE_USERNAME]/
↪transferdata.tar.gz /home/[USERNAME]/
```

In case that the file to be retrieved from Data Archive to SPIDER is not directly available on disk then the scp/rsync command will hang until the file is moved from tape to disk. Data Archive users can query the state of their files by logging into the Data Archive user interface and performing a `dmls -l` on the files of interest. Here the state of the file is either on disk (REG) or on tape (OFL). The Data Archive user interface is accessible via `ssh` from anywhere for users that have a login account and an example is given below:

```
ssh [ARCHIVE_USERNAME]@archive.surfsara.nl
      touch test.txt
      dmls  -l test.txt
      -rw-r--r--  1 homer    homer    0 2019-04-25 15:24 (REG) test.txt
```

Best practices for the usage of Data Archive are described on the Data Archive page.

3.3 Quota policy

Each SPIDER is granted specific compute and storage resources in the context of a project. For these resources there is currently **no hard quotas**. However, we monitor both the core-hour consumption and storage usage to prevent that users exceed their granted allocation.

3.4 Backup policy

The data stored on CephFS (home and project spaces) is disk only, replicated three times for redundancy. For disk-only data there is **no backup**. If you cannot afford to lose this data, we advise you to copy it elsewhere as well.

See also:

Still need help? Contact *our helpdesk*

CHAPTER 4

Software on Spider

Tip: There are several ways to setup and use your software on Spider. In this page you will learn:

- which software is provided by default on the system
- how to install your software on the local filesystem
- using containerization for making your software portable across different platforms
- distributing your software on different platforms

4.1 System software

The standard supported login shell on SPIDER is bash. The standard supported software setup is identical on all nodes. Basic unix functionality is installed system-wide:

- software compilers (e.g., gcc, g++, f95)
- editors (e.g., vi, vim, emacs, nano and edit).
- graphical tools are supported via X11 ssh forwarding on the login node.
- operating system (OS on Spider is CentOS 7.5.1804 (Core) on login and worker nodes.

4.2 User installed software

Software that does not require root privileges can be setup in user-space. The software that is installed on the local CephFS will be made available on all nodes for your jobs because the local directories are globally mounted.

4.2.1 Software on home

Home can be used to install software that you don't want to share with other members in your SPIDER project. This can be placed in the location `/home/$USER`.

4.2.2 Software on project space

For software that you want to install and share with other project members, we advice you to use the `/project/[PROJECTNAME]/Software` in your *Project spaces directories*.

Only dedicated software managers have permissions to write in this directory, and all members in the project have read and execute permissions in this space.

The project members can use the software installed by the software manager, simply by exporting the right path in `$HOME/.bashrc`.

Take a look into our examples below for installing miniconda and samtools in the project space software directory by building from source without root privileges:

Installing Miniconda

- Download and install the latest Miniconda2 package, which contains the conda package manager and python 2.7. The path shown below points to a path in the Software space of the project.

```
wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
chmod +x Miniconda2-latest-Linux-x86_64.sh
./Miniconda2-latest-Linux-x86_64.sh -b -p /project/[PROJECTNAME]/Software/Miniconda2
```

- Add the following line to export the following path in your $HOME/.bashrc:

```
export PATH=/project/[PROJECTNAME]/Software/Miniconda2/bin:$PATH
```

- Logout and login for the changes to take effect. Run the following to install samtools with conda:

```
conda install samtools
```

All the members in your project can now use this software by exporting the paths as shown in the step above.

Installing samtools

Here is an example of installing samtools by building it from the source instead of *using Miniconda2*.

- Download the samtools packages in the Software space of the project:

```
wget https://github.com/samtools/samtools/releases/download/1.3.1/samtools-1.3.1.tar.
↪bz2
tar -xf samtools-1.3.1.tar.bz2
cd samtools-1.3.1
./configure --prefix=/project/[PROJECTNAME]/Software/samtools
make
make install
```

- Add the following line to export the following path in your $HOME/.bashrc:

```
export PATH=/project/[PROJECTNAME]/Software/samtools/bin:$PATH
```

- Logout and login for the changes to take effect.

All the members in your project can now use this software by exporting the paths as shown in the step above.

4.3 Singularity containers

On SPIDER we support Singularity. Singularity is a container solution for building software stacks in the form of images. Singularity enables these images to be run in user space. We dot not provide a space for building Singularity images, but we do support the execution of these images by users on Spider.

The currently supported version of Singularity on SPIDER can be found by typing `singularity --version` on the command line after *login* into the system. Additional information can be found on Singularity SURFsara page and more generic info can be found at the Sylabs documentation.

4.3.1 Upload your image

Your Singularity image can be viewed as a single file containing all the necessary software for your purpose. When compared to traditionally compiled software it is similar to a binary file containing the executable software. The image can be placed anywhere on Spider, as long as the location is accessible to your processing jobs. However, we strongly recommend that you place your Singularity images in one of the dedicated locations for user space software that are described on the *User installed software* page.

4.3.2 Singularity in batch jobs

Regular commands and Singularity based commands are very similar. In many cases for your job submission script you simply add `singularity exec` in front of the commands to be executed within your job. However, please note that in some cases you may need to also use directory binding via the `--bind` option (see *Binding directories*). Below we provide an example comparing a regular command in a job with a Singularity command.

- Regular job on Spider:

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 10:00
#SBATCH -c 1
echo "Hello I am running a regular command using the python version installed on the␣
↪host system"
echo "I am running on " $HOSTNAME
python /home/[USERNAME]/hello_world.py
```

- Singularity command on SPIDER (in this example the image is placed in the home directory of the user):

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 10:00
#SBATCH -c 1
echo "Hello I am running a singularity command using the my own python version␣
↪installed in my image"
echo "I am running on " $HOSTNAME
singularity exec --pwd $PWD /home/[USERNAME]/my-singularity-python-image.simg python /
↪home/[USERNAME]/hello_world.py
```

Please note that that the `--pwd $PWD` is recommended for use. This is because by default, Singularity makes the current working directory within the container the same as on the host system (Spider). For resolving the current

working directory, Singularity looks up the physical absolute path (see `man pwd` for more info). However, some directories on Spider may be symbolic links and the current working directory would then resolve differently than expected. This would then result in your files not being where you expected them to be (combined with some warning messages).

4.3.3 Binding directories

By default Singularity does not *see* the entire directory structure on Spider. This is because by default the file system overlap between the host system and the image is only partial. Additional directories can be made available by the user in severals ways:

(i) Create the directories within the image, see e.g. Singularity SURFsara (note that this requires sudo rights and thus needs to be done outside of Spider)

(ii) Bind new directories at the time of execution via the `--bind` option. For binding directories it is only necessary to specify the top directory.

Below we provide an example for binding the `cvmfs` directory. This is necessary if your Singularity image is distributed via *Softdrive*.

- Singularity command on SPIDER (in this example the image is placed in the Softdrive directory):

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 10:00
#SBATCH -c 1
echo "Hello I am running a singularity command using the software installed in my␣
↪image on Softdrive"
echo "I am running on " $HOSTNAME
singularity exec --bind /cvmfs --pwd $PWD /cvmfs/softdrive.nl/[USERNAME]/my-
↪singularity-image.simg python /home/[USERNAME]/hello_world.py
```

Please note that it is possible to bind several directories by providing a comma separated list to the `--bind` option, e.g. `--bind /cvmfs,/project`. Additional information can be found in the Sylabs documentation.

4.4 Softdrive

With Softdrive SURFsara it is possible to install your software in a central place and distribute it *automagically* across any compute cluster, including SPIDER. Simply put, systems with the CernVM-FS installed have instant access to the Softdrive SURFsara software repositories via the command line. This is very handy when you work on multiple platforms to solve the problem of installing and maintaining the software in different places.

Access on Softdrive is *not* provided by default to the SPIDER projects. To request for Softdrive access, please contact our *our helpdesk*.

If you already have access on Softdrive, then you can use it directly from SPIDER, simply by exporting the `/cvmfs/softdrive.nl/$USER` software paths into your SPIDER scripts or your `.bashrc` file.

On SPIDER nodes, your Softdrive files will be available under:

```
/cvmfs/softdrive.nl/[SOFTDRIVE_USERNAME]/
```

Please note that your [SOFTDRIVE_USERNAME] can be different than your [SPIDER_USERNAME].

See also:

Still need help? Contact *our helpdesk*

CHAPTER 5

Compute on Spider

Tip: This is a quickstart on the platform. In this page you will learn:

- how to prepare and run your workloads
- about job types, partitions and Slurm constraints

5.1 Prepare your workloads

When you submit jobs to the batch system, you create a job script where you specify the resources that your programs need from the system to execute successfully.

Before submitting your jobs, it is a good practice to run a few tests of your programs locally (on the login node or other system) and observe:

i) the time that your programs take to execute

ii) the amount of cores that your software needs to execute these tasks

iii) the maximum memory used by the programs during execution

We suggest you, where possible, first debug your job template on the login node. In doing so, please take into account that the login node is a shared resource and hence any job testing should consume the least demanding set of resources. If you have high resource demands please contact *our helpdesk* for support in testing your jobs.

Once you get a rough estimate of the resources above, you are set to go. Create your job script to request from the scheduler the estimated resources.

In the current setup of Slurm on SPIDER, we ask you to specify at least the following attributes:

SBATCH directive	Functionality	Usage example
`-N <number>`	the number of nodes	`#SBATCH -N 1` (the job will run on a single node)
`-c <number>`	the number of cores	`#SBATCH -c 2` (the job will use 2 cores couple to 16000 MB memory)
`-t HH:MM:SS`	the wall-clock time	`#SBATCH -t=1:00:00` (the job will run max for 1 hour)
`-p <partition>`	partition selection	`#SBATCH -p normal` (the job will run max for 120 hours)
`-p <partition>`	partition selection	`#SBATCH -p infinite` (the job will run max for 720 hours)

5.2 Run your jobs

5.2.1 Running a local Job with *srun*

The `srun` command creates an allocation and executes an application on a cluster managed by Slurm. It comes with a great deal of options for which help is available by typing `srun --help` on the login node. Alternatively, you can also get help at the Slurm documentation page.

The `srun` command when used on the command line is executed locally by Slurm, an example of this is given below. A python script, `hello_world.py`, has the following content;

```
#!/usr/bin/env python
print("Hello World")
```

This python script can be locally executed as;

```
srun python hello_world.py
#Hello World
```

Typically `srun` should only be used with a job script that is submitted with `sbatch` to the Slurm managed job queue.

5.2.2 Submitting a Job Script with *sbatch*

The `sbatch` command submits `batch script` or `job description script` with 1 or more `srun` commands to the batch queue. This script is written in bash, and requires SBATCH header lines that define all of your jobs global parameters. Slurm then manages this queue and schedules the individual `srun` jobs for execution on the available worker nodes. Slurm takes into account the global options specified with `#SBATCH <options>` in the job description script as well as any local options specified for individual `srun <options>` jobs.

Below we provide an example for `sbatch` job submission with options. Here we submit and execute the above mentioned `hello_world.py` script to the queue via `sbatch` and provide options `- N 1` to request only 1 node, `-c 1` to request for 1 core and 8000 MB memory (coupled) and `-t 1:00` to request a maximum run time of 1 minute. The job script, `hello_world.sh`, is an executable bash script with the following code;

```
#!/bin/bash
#SBATCH -N 1
#SBATCH -c 1
#SBATCH -t 1:00
srun python /home/[USERNAME]/[path-to-script]/hello_world.py
```

You can submit this job script to the Slurm managed job queue as;

```
sbatch hello_world.sh
#Submitted batch job 808
```

The job is scheduled in the queue with `jobid 808` and the stdout output of the job is saved in the ascii file `slurm-808.out`.

```
more slurm-808.out
#Hello World
```

More information on `sbatch` can be found at the Slurm documentation page.

5.2.3 Using local `scratch`

If you run jobs that require intensive IO processes, we advise you to use `scratch` because it is local SSD on every compute node of the the SPIDER. This is a temporary storage that can be used only during the execution of your job and will be arbitrarily removed at any point once your job has finished running.

In order to access the `scratch` filesystem within your jobs, you should use the `$TMPDIR` variable in your job script. We advise you the following job workflow:

- At the start of your job, copy the necessary input files to `$TMPDIR`
- Run your analysis and produce your intermediate/output files on `$TMPDIR`
- Copy the output files at the end of the job from `$TMPDIR` to your home directory

Here is a job script template for `$TMPDIR` usage;

```
#!/bin/bash
#SBATCH -N 1       #request 1 node
#SBATCH -c 1       #request 1 core and 8000 MB RAM
#SBATCH -t 5:00    #request 5 minutes jobs slot

mkdir "$TMPDIR"/myanalysis
cp -r $HOME/mydata "$TMPDIR"/myanalysis
cd "$TMPDIR"/myanalysis

# = Run your analysis here =

#when done, copy the output to your /home storage
tar cf output.tar output/
cp "$TMPDIR"/myanalysis/output.tar $HOME/
echo "SUCCESS"
exit 0
```

5.3 Job types

- For regular jobs we advise to always only use 1 node per job script i.e., `-N 1`. If you need multi-node job execution, consider better an HPC facility.
- On SPIDER we provide **8000 MB RAM per core**.
  - This means that your memory requirements can be specified via the number of cores *without* an extra directive for memory
  - For example, by specifying `-c 4` you request 4 cores and 32000 MB RAM

5.4 Slurm partitions

We have configured two partitions on SPIDER as shown in the table above:

- If no partition is specified, the jobs will be scheduled on the normal partition which has a maximum walltime of 120 hours and can run on any worker nodes.
- Infinite queues can run only on two worker nodes with a maximum walltime of 720 hours. Please note that you should run on this partition at your own risk. Jobs running on this partition can be killed without warning for system maintenances and we will not be responsible for data loss or loss of compute hours.

5.5 Slurm constraints

5.5.1 Regular constraints

The Slurm scheduler will schedule your job on any compute node that can fulfil the constraints that you provide with your `sbatch` command upon job submission.

The minimum constraints that we ask you to provide with your job are given in the example above.

Many other constraints can also be provided with your job submission. However, by adding more constraints it may become more difficult to schedule and execute your job. See the Slurm manual (https://slurm.schedmd.com) for more information and please note that not all constraint options are implemented on SPIDER. In case you are in doubt then please contact *our helpdesk*.

5.5.2 Spider-specific constraints

In addition to the regular `sbatch` constraints, we also have introduced a number of Spider-specific constraints that are tailored to the hardware of our compute nodes for the SPIDER platform.

These specific constraints need to be specified via constraint labels to `sbatch` on job submission via the option `--constraint=<constraint-label-1>,<constraint-label-2>,...,<constraint-label-n>`

Here a comma separated list implies that all constraints in the list must be fulfilled before the job can be executed.

In terms of Spider-specific constraints, we support the following constraints to select specific hardware:

1) cpu architecture constraint labels : 'skylake', 'broadwell'
2) local scratch constraint labels ; 'ssd'

As an example we provide below a bash shell script `hello_world.sh` that executes a compiled C script called 'hello'. In this script the #SBATCH line specifies that this script may only be executed on a node with 2 cpu-cores where the node must have a skylake cpu-architecture and ssd (solid state drive) local scratch disk space.

```
#!/bin/bash
#SBATCH -c 2 --constraint=skylake,ssd
echo "start hello script"
/home/[USERNAME]/[path-to-script]/hello
echo "end hello script"
```

From the command line interface the above script may be submitted to Slurm via: `sbatch hello_world.sh`

Please note that not all combinations will be supported. In case you submit a combination that is not available you will receive the following error message:

'sbatch: error: Batch job submission failed: Requested node configuration is not available'

See also:

Still need help? Contact *our helpdesk*

CHAPTER 6

Workflows

Tip: This is a quickstart on the platform. In this page you will learn:

- creating a job dependency with Slurm
- using Picas pilot framework from Spider

6.1 Slurm job dependencies

A job can be given the constraint that it only starts after another job has finished. Lets say that you have two Slurm jobs, A and B. You want job B to start after job A has successfully completed. Here are the steps:

- Submit job A and keep the returned job ID:

```
sbatch <jobA.sh>
```

- Submit job B with a condition to start after job A, by providing its assigned job ID. This way job B only starts after Job A has successfully completed:

```
sbatch --dependency=afterok:<jobID_A> jobB.sh
```

- We can also tell Slurm to run job B, even if job A fails:

```
sbatch --dependency=afterany:<jobID_A> jobB.sh
```

- If you want job B to start after several other jobs have completed, use the delimiter ':' as:

```
sbatch --dependency=afterok:<jobID_A:jobID_C:jobID_D> jobB.sh
```

See also:

For more information on job dependencies, see also the `-d, --dependency` section in the man page of the sbatch command.

6.2 PiCaS

When you run many jobs on SPIDER it can be difficult to keep track of the state of these jobs, especially when you start running hundreds to thousands of jobs. Although Slurm offers some functionality for tracking the status of the jobs, via the Slurm job ID, in many cases a Pilot job framework, such as PiCaS, is necessary for this purpose.

Access on PiCaS is *not* provided by default to the SPIDER projects. To request for PiCaS access, please contact our *our helpdesk*.

If you already have access on PiCaS, then you can use it directly from SPIDER, i.e. you can establish a connection to your CouchDB database and use the python PiCaS client either from the login node or the worker nodes.

To connect with your PiCaS database you need to provide your credentials (username, password, database name). It is possible to specify the password on the command line, however for security reasons this should be avoided on shared systems (like the login node) because it can allow other local users to read the password (e.g. with the `ps` command). Also to avoid having to type these credentials every time your client connects to your database or using them within your jobs, we advice you to authenticate to PiCaS with the steps below.

- Create a PiCaS configuration directory in your home directory. Here we will call this directory `picas_cfg`, but you are free to give it any other name.

```
mkdir /home/[USERNAME]/picas_cfg
chmod go-rwx /home/[USERNAME]/picas_cfg
```

- Check the settings of your directory with `ls -la`. The output should be similar to:

```
ls -la /home/homer/picas_cfg
drwx------ 1 homer homer  3  May  7 08:33 picas_cfg
```

- Create a new file called `picasconfig.py` inside the `picas_cfg` directory:

```
cd /home/[USERNAME]]/picas_cfg
touch picasconfig.py
```

- Add the following lines to the `picasconfig.py` file:

```
PICAS_HOST_URL="https://picas.surfsara.nl:6984"
PICAS_DATABASE="[YOUR_DATABASE_NAME]"
PICAS_USERNAME="[YOUR_USERNAME]"
PICAS_PASSWORD="[YOUR_PASSWORD]"
```

- Storing cleartext passwords in any medium is dangerous, so we need to make sure it is not readable by others. Save the `picasconfig.py` file and for additional security set it to read-write (rw) access for you only:

```
chmod go-rw /home/[USERNAME]/picas_cfg/picasconfig.py
```

- Check the permissions of your `picasconfig.py` file with `ls -la`. The output should be similar to:

```
ls -la /home/homer/picas_cfg/picasconfig.py
-rw------- 1 homer homer  126 May  7 08:33 picasconfig.py
```

- Finally, add the `picas_cfg` directory to your PYTHONPATH environment variable so that python can locate it. We recommend that you set this variable in your /home/[USERNAME]]/.bashrc file by adding the following lines to it:

```
PYTHONPATH=/home/[USERNAME]/picas_cfg:$PYTHONPATH
export PYTHONPATH
```

You are now ready to start using your PiCaS credentials without having to type them each time you or your jobs need to connect to the PiCaS server. Good practices to build worflows with PiCaS can be found in PiCaS example.

See also:

Still need help? Contact *our helpdesk*

CHAPTER 7

Web public view

Tip: Web public views are meant for open data distribution. In this page you will learn:

- how to distribute public data through a web view
- about risks when publishing data

SPIDER allows for public data sharing in a fast and easy way, which is often a huge problem in other platforms. In a web-based front-end, the so called *web public view*, you can give direct access to external collaborators in selected directories containing designated data products.

But how does this work?

Let's say that you have run some analysis on SPIDER and you want to redistribute some scientific products or intermediate results to collaborators that are not members of your SPIDER project, thus they don't have access to your *project space*. Here is what you need to do:

- Login to SPIDER, select the files that you want to publish and copy them to your project space `/project/[PROJECTNAME]/Public` directory, e.g.:

```
#the user [USERNAME: surfadvisors-homer] belongs to the project [PROJECTNAME:␣
↪surfadvisors] and wants to publish the file 'hello-world.sh'
ssh surfadvisors-homer@spider.surfsara.nl
cp $HOME/hello-world.sh /project/surfadvisors/Public/
ls -l /project/surfadvisors/Public/hello-world.sh
#-rwxrwxrwx  1  surfadvisors-homer  surfadvisors-homer  192 Jul  1 08:53 hello-world.
↪sh
```

- That's it! Any data located in your `/project/[PROJECTNAME]/Public` directory is exposed to the web under the domain: `https://public.spider.surfsara.nl/project/[PROJECTNAME]/`. This means that the example file can be downloaded by anyone from link below:

https://public.spider.surfsara.nl/project/surfadvisors/hello-world.sh

Warning: Be careful with the data you place under your project space `/project/[PROJECTNAME]/Public` directory. This data is automatically exposed to the web and anyone that knows the URL can download it!

See also:

Still need help? Contact *our helpdesk*

CHAPTER 8

Jupyter Notebooks

Tip: Jupyter notebooks are very popular in science for interactive work. In this page you will learn:

- how to use Jupyter notebooks on Spider
- which available flavors to choose

Coming soon ..

See also:

Still need help? Contact *our helpdesk*

CHAPTER 9

Maintenances

Tip: To keep the system healthy we need sometimes to interrupt the platform for maintenance. In this page you will find:

- the current status of the platform & the maintenances log
- our downtime policy

9.1 Maintenance windows

9.1.1 Current status

```
Up & Running
```

9.1.2 Future maintenances

Status (Scheduled/Unscheduled)	Date (start / end)	comment

9.1.3 Past maintenances

(most recent first)

Status (Scheduled/Unscheduled)	Date (start / end)	comment

9.2 Policy

All SPIDER users are notified in advance for the scheduled maintenance windows. Exceptions to this policy are made in the following cases:

- Urgent changes are needed for the continuity of the platform
- Patches for high-risk security issues

In these cases, the changes will be applied as soon as needed, even outside maintenance windows.

See also:

Still need help? Contact *our helpdesk*

CHAPTER 10

Getting help

Tip: Perhaps you are puzzled with something and seek for help on SPIDER. In this page you will find:

- our Helpdesk details
- how to get expert advice for project challenges
- the most commonly asked questions and workarounds for known issues
- some tutorial as prerequisites to use the platform

10.1 Helpdesk

If you run into problems and cannot find an answer in our *Spider User Guide* or the *FAQ*, we encourage you to contact our experts who are there to support the SPIDER users. Send us an email with your question at: helpdesk@surfsara.nl

Find more about SURFsara support availability and response times here.

10.2 Expertise

Our support team is specialized in advising on data processing strategies and collaborative data analysis. Over the years the team has gained experience in many unique projects, all requiring a unique combination of generic components to implement their ideal data processing setup. Our team is available to support a range of proven solutions for each problem:

- off-the-shelf data staging processes from a variety of (external) storage systems
- off-the-shelf processing orchestration processes
- specialized consultancy for automated production pipes

10.3 FAQ

10.3.1 I cannot login after adding/changing my ssh keys

If you added or changed your SSH key on the SURFsara portal, access to the login node may not be immediately possible. It can take *up to 15 minutes* to be able to login to SPIDER. In case that it takes longer time, please contact us at *helpdesk*.

10.4 Tutorials

- Tutorial for Introduction to Unix
- Tutorial for Introduction to batch scheduling systems
- Tutorial for Slurm Quickstart

See also:

Still need help? Contact *our helpdesk*

Other info:

CHAPTER 11

Documentation how-to

Tip: Do you have ideas for corrections and improvements of our user documentation? In this page you will learn how to:

- contribute to our documentation
- edit the docs with Sphinx language
- build the docs on your laptop

Our documentation is hosted on Github and is written in Sphinx *restructed text*. Behind the scenes we use ReadThe-Docs to publish it automatically. You can contribute either directly to our Spider Github repo or send an email to *our helpdesk* with your remarks and we will change the documentation ourselves. Any contribution is welcome!

The rest of this page explains how to to submit your changes directly through Github.

11.1 Contribute through GitHub

In case that you have a GitHub account and a little knowledge of git (see GitHub's git cheat sheet), you can try submitting your changes directly to our repository. Here is what you have to do:

1. Fork our Spider Github repo
2. Git pull your fork
3. *Edit with Sphinx language* the files with your changes
4. *Build the documentation locally* to preview the changes
5. Commit and push your changes back to your fork
6. Create a pull request to inform us of your changes
7. After we've reviewed and accepted your work, we will merge your commits and the documentation will be updated automatically

11.2 Edit with Sphinx language

When you contribute directly to our Github repo we ask you to write the changes in Sphinx language. The philosophy of Sphinx documentation is that content is stored in files that can be easily read *and* edited by humans, in a format called *restructured text*, with the file extension `.rst`. Using a simple grammar, text can be styled. The document is structured using special tags; using these tags, documentation can be split into multiple files, and you can cross-reference between files and build indexes.

Although Sphinx is quite intuitive, we have created a simple Sphinx cheatsheet to help you use the Sphinx syntax:

11.2.1 Cheatsheet

Look at the raw version of this file to compare the source and build version: `sphinx_cheatsheet.rst`

The first lines of this page will also be explained later on in this document (see *Links* and *Titles*)

Moderate emphasis

Strong emphasis

```
This is text in Courier
```

This is plain text.

- Use `literal markup` for:
  - pieces of code
  - commands
  - arguments
  - file names, hostnames
  - a specific term, when emphasis is on its name
  - configuration file statements and values
- Acronyms:

In general: avoid acronyms. When you want to use them, the first occurrence on a page should explain them: CephFS (Ceph Filesystem). Sphinx supports a :abbr: tag, see http://www.sphinx-doc.org/en/stable/markup/inline.html#other-semantic-markup. Here's a test: CEPHFS (Ceph Filesystem)

- This is a bulleted list.
- ...

1. This is a numbered list.
2. It has two items.

- this is
- a list
  - with a nested list
  - and some subitems
- and here the parent list continues

Python

or

SURFsara website (see bottom of the document; that is were we tell Sphinx were SURFsara website should point to)

This is an implicit link to title:

Sample H2

Internal wiki link:

Reference tag: place above a title: .. _my-reference-label:

Then refer to it from another page as. For example, for this cheatsheet: cheatsheet or ref:*other label <cheatsheet>*

11.2.2 H1: document title

Sample H2

Sample H3

Sample H4

Sample H5

Sample H6

And some text.

Header 1	Header 2	Header 3
body row 1	column 2	column 3
body row 2	Cells may span columns.	
body row 3	Cells may span rows.	• Cells • contain • blocks.
body row 4		

or

Column1	Column2
value1	40
value2	41
value3	42

Warning: This is a **warning** box.

Note: This is a **note** box.

Tip: This is a **tip** box.

Error: This is an **error** box.

See also:

This is a simple **seealso** note.

Your Topic Title

Subsequent indented lines comprise the body of the topic, and are interpreted as body elements.

Sidebar Title

Optional Sidebar Subtitle

Subsequent indented lines comprise the body of the sidebar, and are interpreted as body elements.

`path/to/myfile.txt`

`A file for download`

- When you want to display commands, output and comments, use `.. code-block:: bash`. Prefix commands with a `$` and prefix comments with `##`, otherwise they are marked up as a command. Example:

```
$echo 'Hello World!'
Hello World!
## Comments should be prefixed with a double ``#``.
```

- When you want to display commands and comments, use `.. code-block:: bash`. Don't prefix commands. Example:

```
# [homer@htp-ui ~]$ is the first prompt upon login
ssh [USERNAME]@[Spider HOSTNAME]
```

- You can add line numbers to code examples with the :linenos: parameter.

```
1 # [homer@htp-ui ~]$ is the first prompt upon login
2 ssh [USERNAME]@[Spider HOSTNAME]
```

- To display the contents of a shell script, use `.. code-block:: bash`.
- To display configuration files, use `.. code-block:: cfg`.

11.3 Build the documentation locally

Because the syntax of the files is human readable, you can edit the files using your favorite text editor. Once you are done editing, you can generate documentation in various formats, such as HTML or epub. While you can edit the pages on virtually any system, it is recommended to preview your changes before publishing them.

There are different ways to generate the HTML documentation from source and review your changes:

- *Docker image*
- *Sphinx local installation*
- *GitHub edit/preview*

Note that you only need to use one of the options mentioned above. Using Docker is the preferred way, as this mimics the ReadTheDocs build system closest. GitHub edit/preview on the other hand is good enough for minor, textual changes, but is otherwise the least preferred option.

Below you will find information for each of the methods.

11.3.1 Docker image

This is the preferred option to build and test your changes. It tries to build the documentation the same way as readthedocs.org.

Setup the readthedocs Docker image

Preparation

Before you start with the Docker image, make sure that:

- You have Docker Engine installed
- Docker is running on your host

Build image

The official Docker image can be found in the readthedocs repository: https://hub.docker.com/r/readthedocs/build/

To pull the image to your laptop, run:

```
docker pull readthedocs/build
```

- Once the Docker image is ready, find the following script inside your Github fork and run it to build your documentation:

  ```
  ./build.sh
  ```

Optionally you can provide an output location (default: ./build) and the Docker image name (default: readthedocs/build):

```
./build.sh /alternative/output/path/ docker_image_alternative_name
```

Note: For Mac OS X, use `./build_mac.sh` instead.

11.3.2 Sphinx local installation

For the Sphinx documentation setup locally you will need to:

Install Sphinx

Spider documentation itself is written in *restructured text*. This page explains how to install the Sphinx software on your local machine.

Linux

On recent CentOS distributions, *Sphinx* is included in epel. Fedora has it included in the standard distribution. Install *sphinx* with:

```
sudo yum install python-sphinx
```

On Ubuntu, install with:

```
sudo apt-get install python-sphinx
```

Mac OS X

On Mac OS X, you will need macports, so get that if you have not done so.

Next, follow the instructions on the sphinx installation page.

Warning: the standard terminal in Mac OS X has locale settings incompatible with sphinx; to change it once for a terminal session, type:

```
export -n LC_CTYPE
```

To change it permanently:

1. open the terminal preferences
2. go to the tab 'advanced'
3. uncheck "Set locale environment variables on startup"

- To generate HTML documentation, use the command:

```
make html
```

which will generate static pages in the `build`-directory as long as you have the software Sphinx installed locally.

11.3.3 Github edit/preview

For small changes you can edit a page directly from your GitHub fork webview. The *preview* button does not give a fully compatible *rst* overview, but is sufficient for textual changes.

See also:

Still need help? Contact *our helpdesk*

CHAPTER 12

Cookiebeleid

12.1 Wat is een cookie?

Wij maken op deze website gebruik van cookies. Een cookie is een eenvoudig klein bestandje dat met pagina's van deze website wordt meegestuurd en door uw browser op uw harde schrijf van uw computer wordt opgeslagen. De daarin opgeslagen informatie kan bij een volgend bezoek weer naar onze servers teruggestuurd worden.

12.2 Google Analytics

Via onze website wordt een cookie geplaatst van het Amerikaanse bedrijf Google, als deel van de "Analytics"-dienst. Wij gebruiken deze dienst om bij te houden en rapportages te krijgen over hoe bezoekers de website gebruiken. Google kan deze informatie aan derden verschaffen indien Google hiertoe wettelijk wordt verplicht, of voor zover derden de informatie namens Google verwerken. Wij hebben hier geen invloed op. Wij hebben Google niet toegestaan de verkregen analytics informatie te gebruiken voor andere Google diensten. De informatie die Google verzamelt wordt zo veel mogelijk geanonimiseerd. Uw IP-adres wordt nadrukkelijk niet meegegeven. De informatie wordt overgebracht naar en door Google opgeslagen op servers in de Verenigde Staten. Google stelt zich te houden aan de Safe Harbor principles en is aangesloten bij het Safe Harbor-programma van het Amerikaanse Ministerie van Handel. Dit houdt in dat er sprake is van een passend beschermingsniveau voor de verwerking van eventuele persoonsgegevens.

12.3 In- en uitschakelen van cookies en verwijdering daarvan

Meer informatie omtrent het in- en uitschakelen en het verwijderen van cookies kan je vinden in de instructies en/of met behulp van de Help-functie van jouw browser.

12.4 Meer informatie over cookies?

Op de volgende websites kan je meer informatie over cookies vinden:

- Cookierecht.nl
- Consumentenbond: “Wat zijn cookies?”
- Consumentenbond: “Waarvoor dienen cookies?”
- Consumentenbond: “Cookies verwijderen”
- Consumentenbond: “Cookies uitschakelen”
- Your Online Choices: “A guide to online behavioural advertising”

Happy Spidering!..

Tip: Need help? Contact *our helpdesk*
