



SURF RESEARCHDRIVE

Tom Wezepoel
SURFsara

CS3 Rome 2019

SURF SARA

Intro



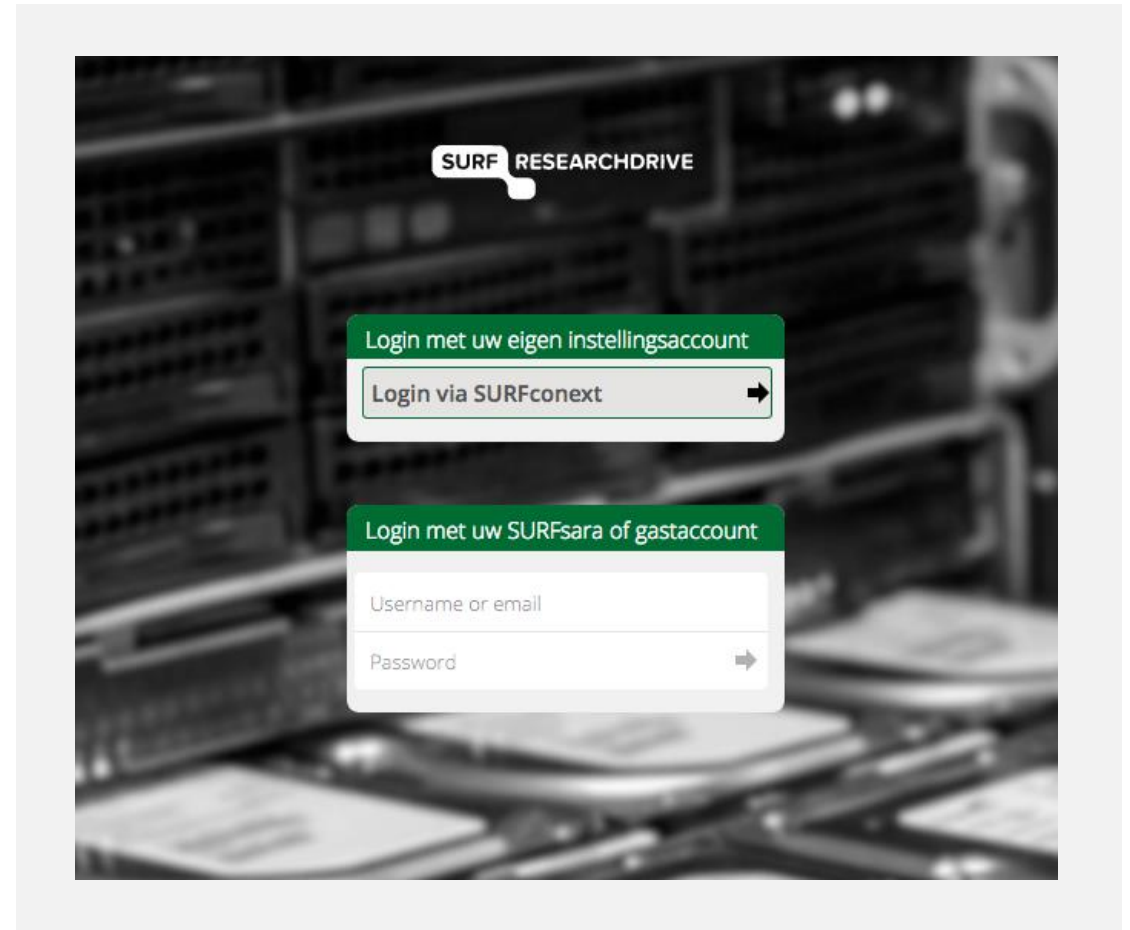
- Personal cloud storage service for Dutch education and research
- Today: > 41000 users, 300.000.000 files & 470TiB of storage
- Storage quota of 250GB
- Login through SAML (institutional account)

SURF RESEARCHDRIVE

- Similar to SURFdrive but different
- A shared-storage environment focused on/for Research (Teams) who require large storage quota's.
- Possibility for them who need to work collaboratively with other Educational & Governmental institutions and external private companies/partners.

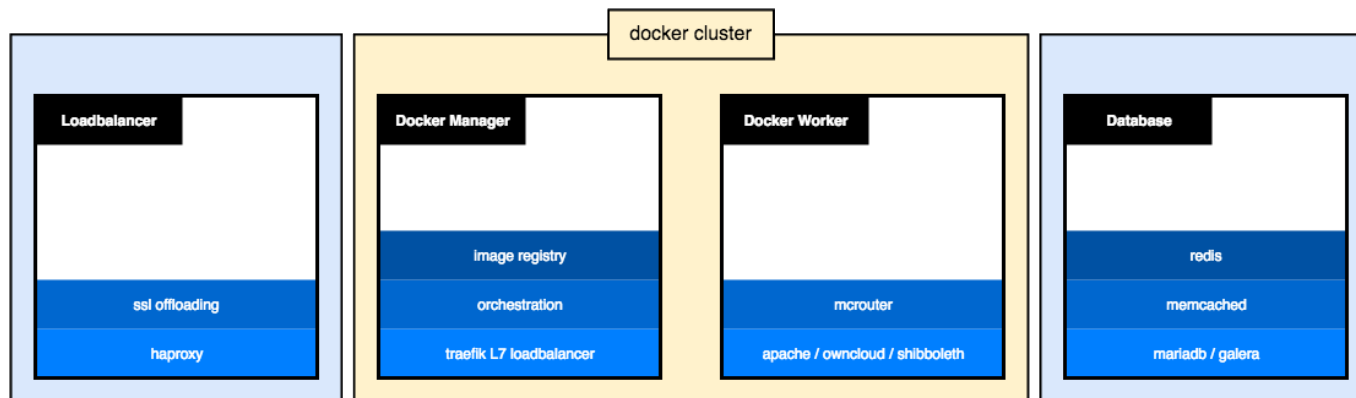
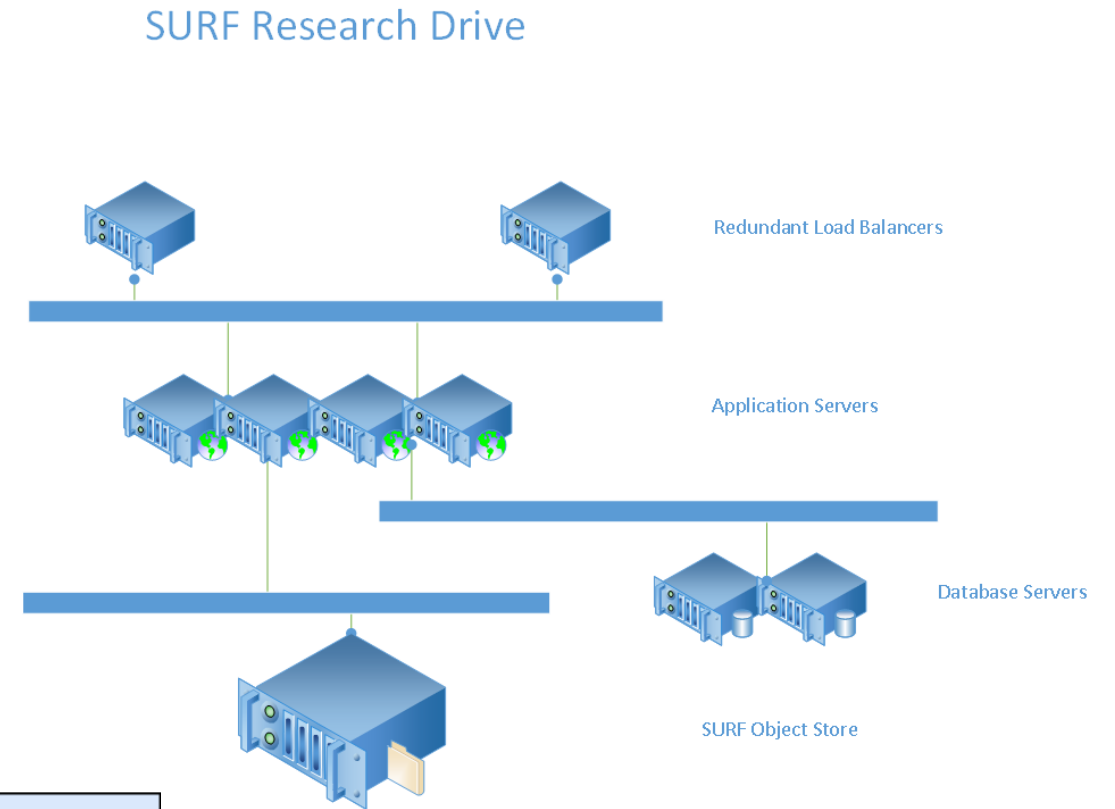
Our design

- Self-regulated data stewardship (Manage / create folder structures, user access and determining quotas.)
- Flexible Storage Space limits, starting from 1TB
- Multiple Authentication methods;
 - SURFsara LDAP
 - Local Guest Account
 - SAML (SURFconext / institute account)
- Possibility for functional accounts
- Possibility to link external storage



Technical Design

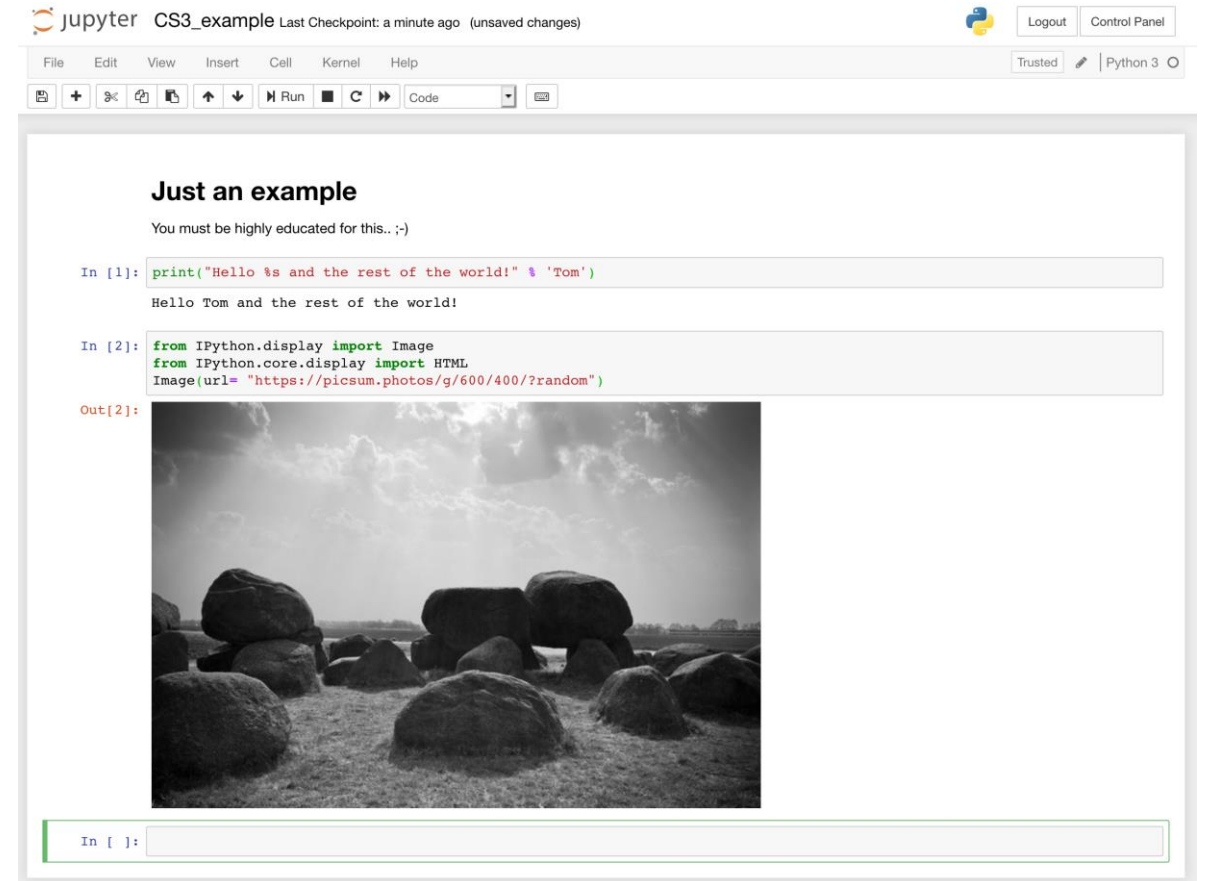
- Haproxy / keepalived front-end
- Docker cluster, OwnCloud running in Containers
- Storage on Swift S3 Object Storage
- MariaDB Galera Database Cluster



Integrating Jupyter Hub

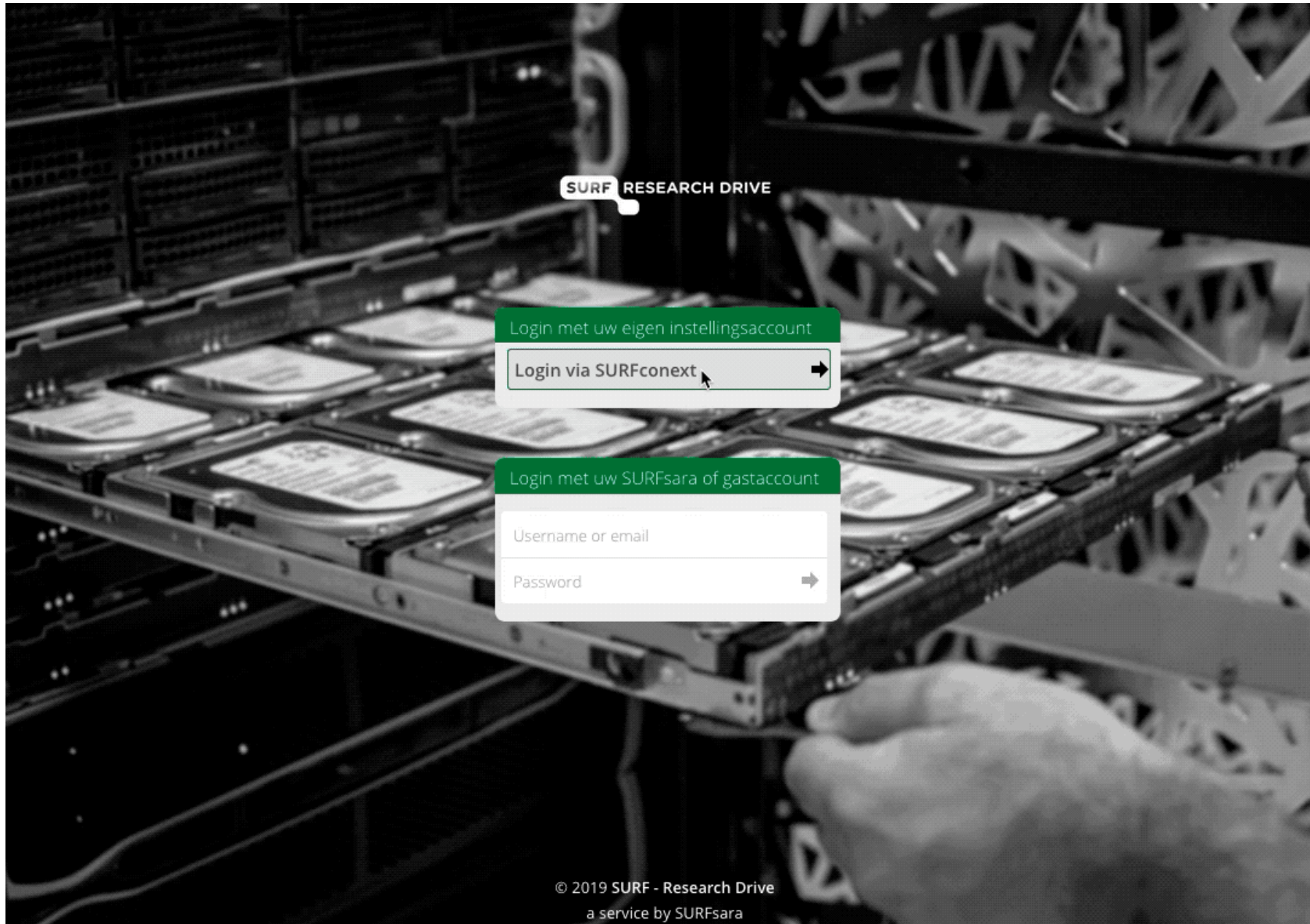
- Web application for running and sharing notebook documents
- A notebook document can contain code and other elements that are executable via a kernel.
- A notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.
- Makes it possible to serve a pre-configured data science environment to any user in the world. It is customizable and scalable, and is suitable for small and large teams, academic courses, and large-scale infrastructure.

Source: <https://jupyter-notebook.readthedocs.io>



The screenshot shows a Jupyter Notebook interface. At the top, it says "jupyter CS3_example Last Checkpoint: a minute ago (unsaved changes)". There are "Logout" and "Control Panel" buttons in the top right. Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". To the right of the menu bar are "Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for file operations, a "Run" button, and a "Code" dropdown menu. The main content area has a heading "Just an example" and a message "You must be highly educated for this.. :-)". There are two code cells. The first cell contains the code `print("Hello %s and the rest of the world!" % 'Tom')` and its output is "Hello Tom and the rest of the world!". The second cell contains the code `from IPython.display import Image`, `from IPython.core.display import HTML`, and `Image(url="https://picsum.photos/g/600/400/?random")`. The output of the second cell is a black and white photograph of large, rounded rocks on a grassy field under a cloudy sky. At the bottom of the notebook, there is an empty code cell with the prompt "In []:".

Faster demo!



The ingredients

- Container based on the JupyterHub docker image
- Apache HTTP Reverse-Proxy from OwnCloud front-end to Jupyter Websocket
- User authentication
- Some fine tuning..



Apache HTTP Reverse-Proxy

- Jupyter Hub is running in its own container, within the Research Drive Docker Service Stack
- User requests arrive at the OwnCloud containers and forwarded to the Jupyter Websocket via Reverse Proxy

```
<Location /jupyter>  
  Satisfy Any  
  Allow from all  
  AuthType None  
  Require all granted  
</Location>
```

```
ProxyPreserveHost On  
ProxyPass /jupyter http://jupyterhub:8000/jupyter  
ProxyPassReverse /jupyter http://jupyterhub:8000/jupyter
```

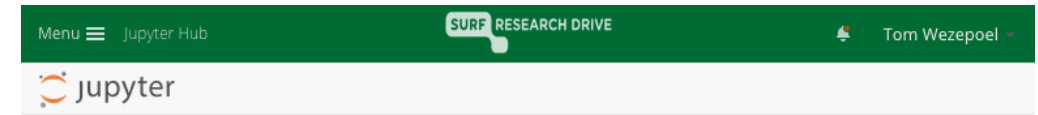
```
# Use RewriteEngine to handle websocket connection upgrades  
RewriteEngine On  
RewriteCond %{HTTP:Connection} Upgrade [NC]  
RewriteCond %{HTTP:Upgrade} websocket [NC]  
RewriteRule /(.*) ws://jupyterhub:8000/$1 [P,L]
```

User authentication

- Jupyter User authentication can be done in various ways
 - SAML / Shibboleth¹
 - OAuth token²
- Requirement; Jupyter Hub only available for certain group
 - Extended ownCloud External Sites app with a group option
 - Extended Jupyter Hub with a database check to the ownCloud database

1. <https://github.com/HarryKodden/JupyterHub-SAML>

2. <https://github.com/jupyterhub/oauthenticator>



Sign in with your Research Drive account

```
# Check in the OC database if the user is a member of the Jupyter group
def getJupyterUserFromDB(self, user):
    query_users = self.DBQuery("SELECT uid FROM "+ self.db_name + ".oc_group_user
                                WHERE gid = 'jupyter' and uid = '"+ str(user) +"'")

    if query_users:
        return True
    else:
        return False
```

User authentication – OAuth2

- Add a new OAuth 2.0 client for Jupyter in ownCloud admin interface
- Define GenericOAuthenticator in the Jupyter Config

```
# cat /etc/jupyterhub/jupyterhub_config.py
```

```
# Import OAuth2 Authenticator
```

```
from oauthenticator.generic import LocalGenericOAuthenticator
```

```
c.JupyterHub.authenticator_class = LocalGenericOAuthenticator
```

```
c.GenericOAuthenticator.username_key = "sub"
```

```
c.GenericOAuthenticator.create_system_users = True
```

```
c.GenericOAuthenticator.login_service = "your ownCloud account"
```

```
c.GenericOAuthenticator.token_url = "https://< your-ownCloud-url >/index.php/apps/oauth2/api/v1/token"
```

```
c.GenericOAuthenticator.userdata_url = "https://<your-ownCloud-url>/index.php/apps/oauth2/api/v1/userinfo"
```

```
c.OAuthenticator.client_id = "<ownCloud OAuth Client Identifier>"
```

```
c.OAuthenticator.client_secret = "<ownCloud OAuth Client Secret>"
```

```
c.OAuthenticator.oauth_callback_url = "https://://<your-ownCloud-url>/jupyter/hub/oauth_callback"
```

OAuth 2.0

Registered clients

Name	Redirection URI	Client Identifier
Desktop Client	http://localhost:*	xdXOt13JKxym1B1Qc
Android	oc://android.owncloud.com	e4rAsNUSIU01F4nl
iOS	oc://ios.owncloud.com	mxd50QDk6es5LzOzF
Jupyter Hub	https://researchdrive.surfsara.nl/jupyter/hub/oauth_callback	zoGiLt30CqzteHox8

Add client

Allow subdomains

Integrate user's storage

- Mounting of the Object Storage bucket isn't possible. WebDAV?
- Mounting of WebDAV with an oAuth token isn't possible.. Ask for the WebDAV password?
- How do we get that WebDAV storage in the container?

Menu Jupyter Hub SURF RESEARCH DRIVE Tom Wezepoel

jupyter Home Token Logout

Spawner Options

Login information:

To use Jupyter Notebooks, you must provide your Research Drive OwnCloud WebDAV credentials.
[Click here to generate a webdav password.](#)

WebDAV username
tom.wezepoel@surfsara.

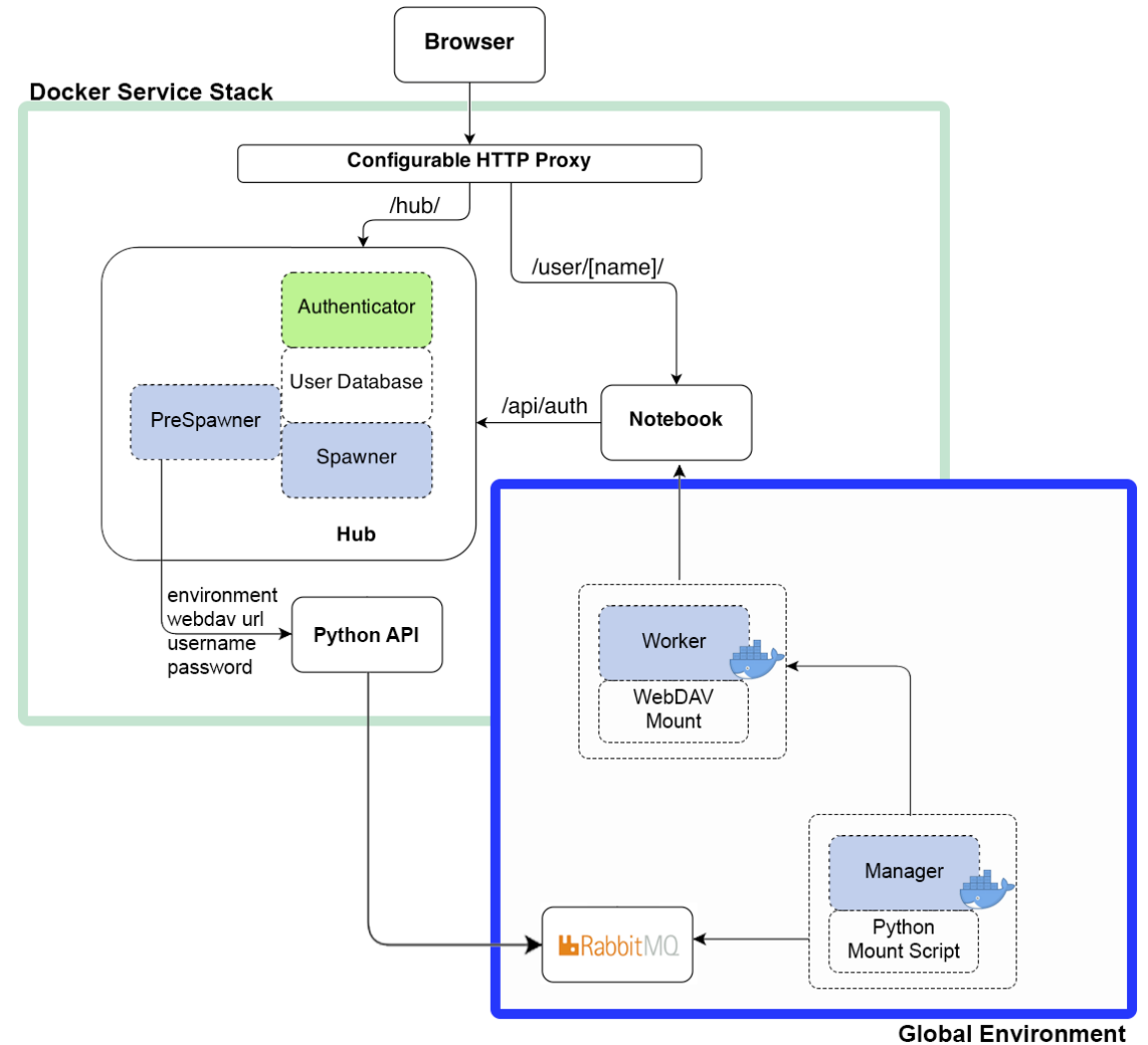
WebDAV password

Select your notebook
DataScience Notebook

Create your notebook

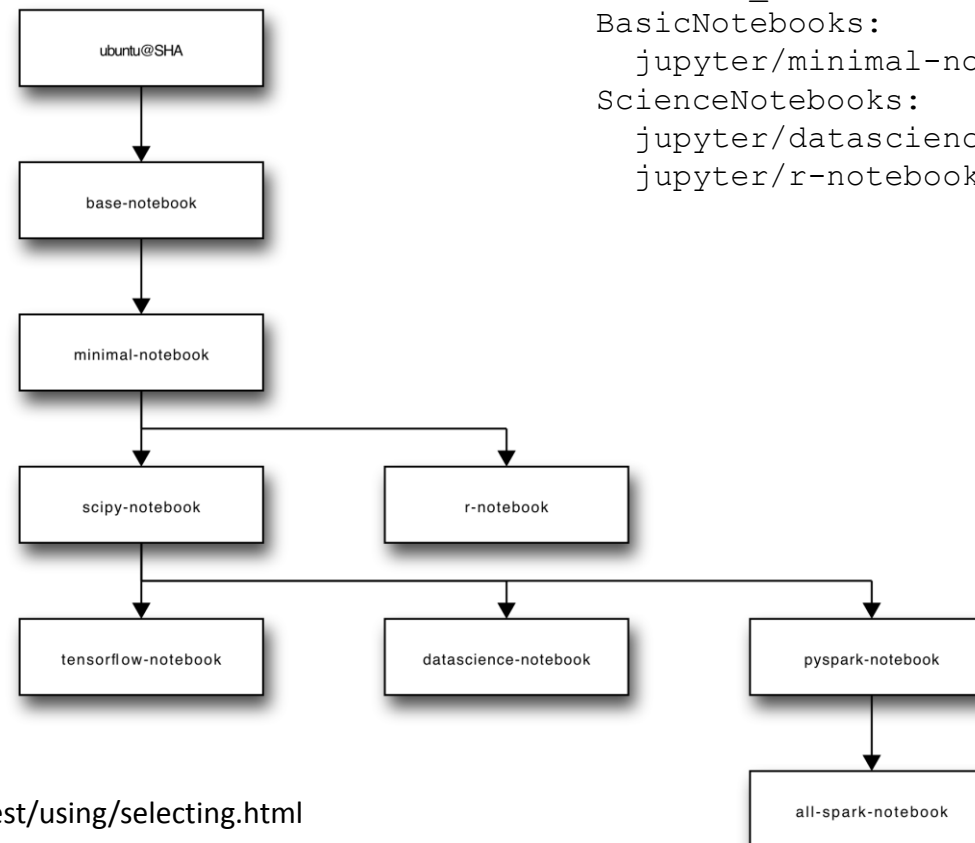
Integrate user's storage

- Jupyter Hub launches a proxy
- The proxy forwards all requests to Jupyter Hub
- The Hub handles user authentication and spawns a user notebook in a private container on demand
- PreSpawner is triggered and sends the user's credentials to RabbitMQ via a Python API
- The Python WebDAV mount script on the Docker Managers read the RabbitMQ mount and create the WebDAV mount on Docker Worker where the notebook is running.
- On the Docker Worker the mount will be connected to the notebook container



Supported notebooks

- Currently we only support the Jupyter default defined notebooks



```
#####  
# Jupyter values
```

```
jupyterhub_images:  
  BasicNotebooks:  
    jupyter/minimal-notebook      : Minimal Notebook  
  ScienceNotebooks:  
    jupyter/datascience-notebook : DataScience Notebook  
    jupyter/r-notebook            : R Notebook
```

<https://jupyter-docker-stacks.readthedocs.io/en/latest/using/selecting.html>

Challenges

- 500 : Internal Server Error
DNS timing issue by a fast respawn of a notebook
- Jupyter Hub container naming and escape sequence redefined
- Fix applied for "bad" usernames due to non-alphanumeric characters (Username can contains an @-sign)

```
### HTTP 500 - Internal Server Error fix ###
```

```
def get_ip_and_port(self):  
    """Queries Docker daemon for service's IP and port."""  
    if self.use_internal_ip:  
        ip = self.service_name  
        port = self.port  
  
        # SURFsara patch  
        # Return IP address instead of hostname  
        # Jupyter still tries to connect to previous container IP  
        # Retry mechanism which try it up to 30 times,  
        # before to hung up.  
        import time  
        import socket  
        time.sleep(3)  
        for attempt in range(30):  
            try:  
                time.sleep(1)  
                ip = socket.gethostbyname(ip)  
                self.log.info("Jupyter environment '%s'  
                    ip address is: %s",self.service_name,ip)  
                break  
            except:  
                self.log.info("Jupyter environment '%s' is  
                    still unknown, retrying %s..",  
                    self.service_name,str(attempt))  
                continue  
        else:  
            self.log.error("Jupyter environment '%s' is still  
                unknown, please check docker logs.", self.service_name)
```

Future plans

- Move the WebDAV mount to an extra WebDAV container for each user
- Add more docker workers, to handle the demand. 😊
- Extend login page, with the option to choose the amount of memory & CPU's





**LET'S
COLLABORATE!**

 Tom Wezepoel

 tom.wezepoel@surfsara.nl

 www.surf.nl

SURF SARA