
TRUSTED CI
THE NSF CYBERSECURITY
CENTER OF EXCELLENCE
trustedci.org

Trusted CI:
The NSF Cybersecurity Center of Excellence

Jupyter Security

Rick Wagner
rick@globus.org
Matthias Bussonnier
bussonniermatthias@gmail.com
Ishan Abhinit
iabhinit@iu.edu
Mark Krenz
mkrenz@iu.edu

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Introduction: Agenda, etc.

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Trusted CI:
The NSF Cybersecurity Center of Excellence

Our mission: to provide the NSF community a coherent understanding of cybersecurity's role in producing trustworthy science and the information and know-how required to achieve and maintain effective cybersecurity programs.

CENTER FOR APPLIED CYBERSECURITY RESEARCH
INDIANA UNIVERSITY
Pervasive Technology Institute

NCSA

INTERNET2

WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

PITTSBURGH SUPERCOMPUTING CENTER

BERKELEY LAB

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

https://trustedci.org/

What is Cyberinfrastructure (CI)?

"The comprehensive infrastructure needed to capitalize on dramatic advances in information technology has been termed cyberinfrastructure (CI). Cyberinfrastructure integrates hardware for computing, data and networks, digitally-enabled sensors, observatories and experimental facilities, and an interoperable suite of software and middleware services and tools. "

-NSF Cyberinfrastructure Vision for 21st Century Discovery

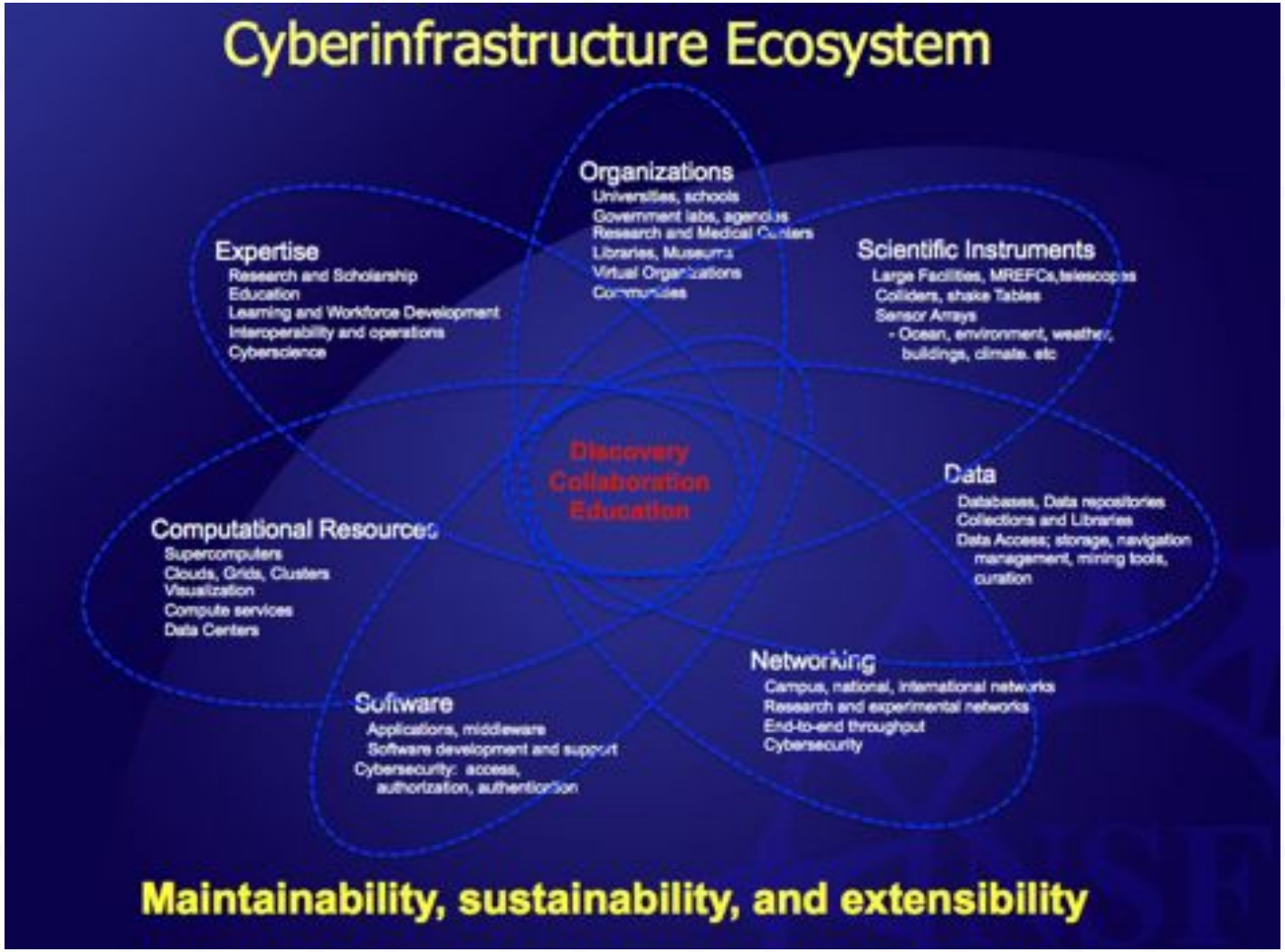


Image credit: NSF

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Today's Agenda

30 minutes:

- Taking questions: What do you want to know about Jupyter & Jupyter security?
- What are the gaps in the current documentation and understanding?
- We'll try to answer some of these questions at the end.

30 minutes:

- What is Jupyter?
- The landscape: Project Jupyter; notebooks; notebook server; IPython; JupyterHub; etc.
- Architecture, where things run and how they communicate.

90 minutes, with a 30 minute break at 4:00 p.m:

- Running Jupyter notebooks.
- Viewing un-encrypted traffic.
- Configure and run JupyterHub.
- Configure security options & recheck traffic.
- Set up external authentication

60 minutes:

- Answer questions from the beginning
- Discussion of Jupyter use cases and needs around security best practices.

Taking questions

- Who are you?
- What is your experience with Jupyter?
- What are you hoping to get out of this workshop/which questions do you have?

Introduction to Jupyter
&
Jupyter Security

Where to ask for help & contribute

- Jupyter.org/community
- Mailing list and security repositories.
  - security@ipython.org (Google Group, You can request access)
  - More concern contact core dev of relevant projects or steering council
- https://discourse.jupyter.org/
- GitHub
- gitter.im
- Jupyter at Research Facilities google groups

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Overview of Jupyter architecture & nomenclature

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Jupyter Notebook

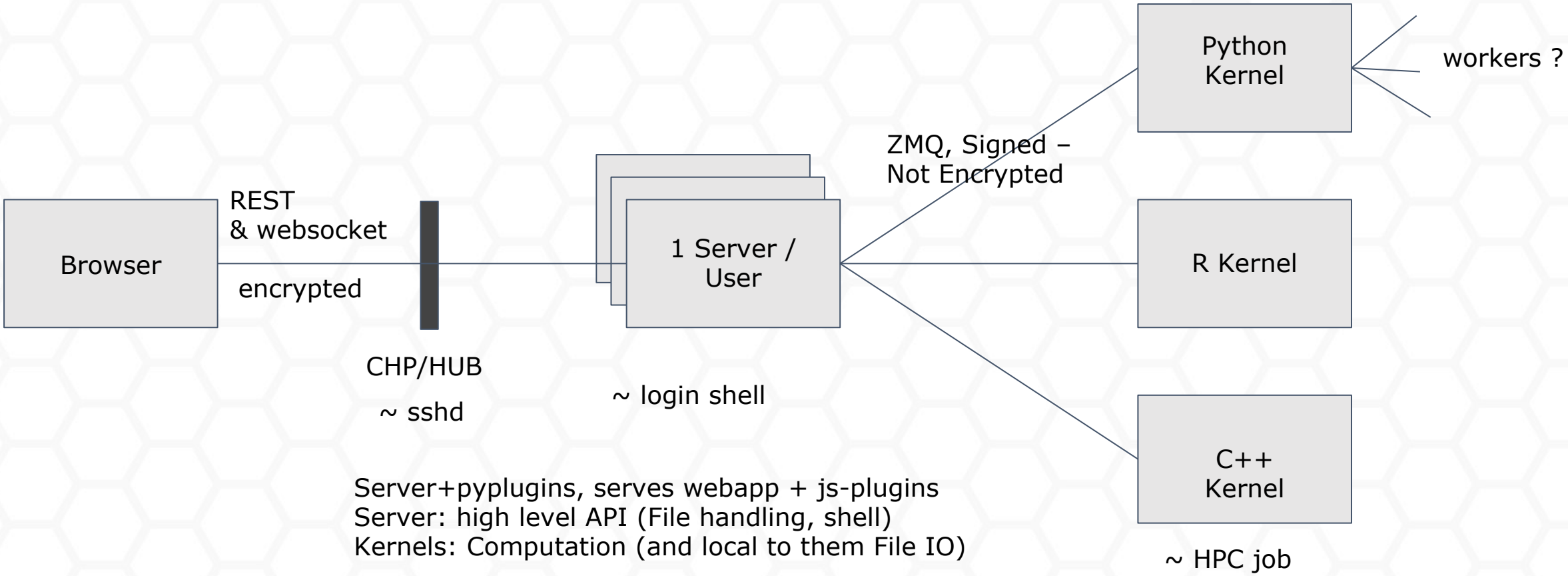
- Notebook **documents** are produced by Jupyter Notebook **App** which contain both computer code and rich text elements.
- The jupyter Notebook app is a server-client application that allows editing and running notebook documents via web browser.
- Notebook can be **executed** on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet.
- You will notice the **Jupyter** and **Notebook** terms are highly overloaded.

Running a Notebook Server

- By default, a notebook server runs locally at 127.0.0.1:8888 & accessible only from localhost
- By default use unique token authentication on filesystem to protect from other processes from other users accessing the API
- Understanding Single-User threat model is critical to understand multi-user deployment threat-model.

- Multi-user deployment should use JupyterHub

Jupyter Architecture



Server+pyplugins, serves webapp + js-plugins
Server: high level API (File handling, shell)
Kernels: Computation (and local to them File IO)

Name / Nomenclature

- **IPython** : Python package that handle executing the User-defined Python code. Historically all the **Jupyter-(.+)** where into IPython repo and named **IPython-\1.** Now only the CLI to execute Python
- **IPykernel**: Small Wrapper around IPython that talk the Jupyter ZMQ Protocol.
- **Kernel:** A process that execute user code.
- **Notebook Server:** Single user web server that serve the html/js for the browser app handles files and some plugins. Websocket/ZMQ bridge.
- **Notebook App:** JS app running in user's browser
- **Notebook document/file:** the actual json file on disk, or as loaded in the above JS APP
- **JupyterHub:**
  - Set of processes, plugin and configuration to do **Multi-User deployment**
  - **Hub itself** (jupyter aware) **+ CHP**(generic software defined networking)
- **Binder:** Authentication-less, w/ ephemeral users JupyterHub.
- **Zero-To-JupyterHub:** Hands on Deploy your own JupyterHub

Note: Many of these are *defaults* there are a number of alternatives implementations.

Threat Model vs Functionality

Avoid: Arbitrary Code Execution

Allow: Arbitrary Code Execution

Venn Diagram

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Threat Model

- **Kernel: Ensure request come from identified source (signature). Beyond that arbitrary code execution is the goal. Even modifying kernel itself. Use other mechanisms to prevent issues: Quotas, Docker, Cgroup...**

- **Server: Run as the unprivileged user. Gateway to all the functionality. Equivalent to shell. Largest attack surface. Most API need authentication (cookie). Target of enumeration attacks; bruteforce… Rarely interprets any user supplied data – though plugins can.**

- **CHP/HUB: Proxy credentials; delegate some services; allow impersonation; privileged user. Smallest attack surface; ONLY PRIVILEGED PROCESSES**

- **Browser: Arbitrary JS extension; interprete user input; opened files; Display embedded HTML/JS.**

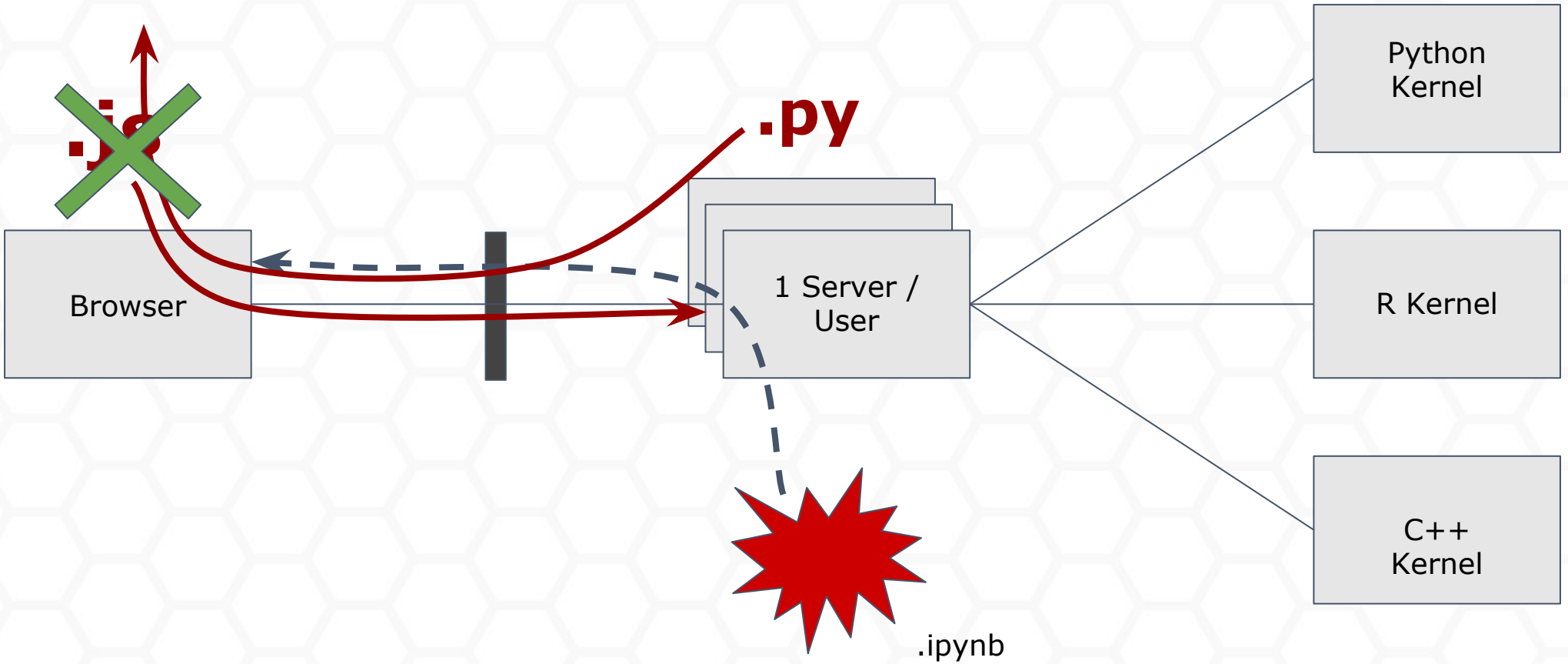
Threat Model Example of Attacks

- **Kernel:**
  - **Malicious Library (pip install panda <- no s, and import panda)**
  - **Sniff ZMQ request; user code has sensitive input (password in clear ?)**

- **Server:**
  - **Rogue Plugin can access all files / executions.**
  - **Token with in World-Writable location, impersonate user.**

- **CHP/HUB:**
  - **MITM Admin user**

- **Browser:**
  - **XSS, Phishing incite user to “trust”**



TRUSTED CI
THE NSF CYBERSECURITY
CENTER OF EXCELLENCE

Jupyter core security model

From https://jupyter-notebook.readthedocs.io/en/latest/security.html

Our security model

- Untrusted HTML is always sanitized
- **Untrusted** Javascript is never executed
- HTML and Javascript in Markdown cells are never trusted
- **Outputs** generated by the user are trusted
- Any other HTML or Javascript (in Markdown cells, output generated by others) is never trusted
- The central question of trust is “Did the current user do this?”

Trusted JS/HTML

HTML which is or **has been** explicitly requested by the user

Example:

- **Shift Enter**
- **Run All**
- **Trust This notebook**
  - **Once trusted, hash of the file stored on disk in LRU cache with random eviction.**

The rest of the security is pluggable

Example:

- Authentication PAM/LDAP
- Direct CHP or NGINX
- Kubernetes Spawner/Slurm Spawner/SUDO spawner.
- Resource limitation by cgroup.
- Single managed environment or per-user environment.

Securing an Installation

STEP -1

- If you don’t provide system wide install user will use ssh-tunnel and old versions. You want **control** and **monitoring** of how users use Jupyter.

- I’m assuming your users are not actively hostile toward each other

Securing an Installation

STEP 0

- You should use **JupyterHub**, and google for JupyterHub for any security related questions in **multi-user** scenario

“Secure Config Jupyter == Secure Config Bash”

“Secure Config Jupyter**Hub** == Secure Config sshd”

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Securing an Installation

STEP 1

- Mailing list and security repositories.
  - security@ipython.org (Google Group, You can request access)
  - More concern contact core dev of relevant projects or steering council
- https://discourse.jupyter.org/

Securing an Installation

STEP 2

- Securing Privileged processes. CHP/HUB
  - Run behind Nginx/Apache
  - Run as unprivileged (opt-in users)
    - Sudo Spawner (limit process privileges)
    - Slurm, Kubernetes Spawner (limit/isolate servers)

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

```
Cmnd_Alias JUPYTER_CMD = /home/jupyterhub/miniconda3/bin/sudospawner

jupyterhub      ALL=(%jupyter)      NOPASSWD:JUPYTER_CMD
```

Securing an Installation

STEP 2

- None of Jupyter(Hub) processes are privileged
- Known SSL termination/vetted endpoint.
- per-user server isolated potentially even more.

- ADMIN/Sudoers weak link to elevated privileges
- Can run them in container/VM if you wish.

Securing an Installation

STEP 3

- Use OAUTH/LDAP authenticator/2FA

Securing an Installation

STEP 3

- Reduce User friction and possibility for Phishing.
- Reduce possibility to use social engineering to get access.
- Directly hook into monitoring, fail2ban...

Securing an Installation

STEP 3

- SSL all the way (CHP to single-user servers)
- and/or , Spawn in containers...

Securing an Installation

STEP 3

- Reduce possibility for attacker that got access to 1 account to potentially see traffic between CHP/Single-users

Not worse than SSHD at that point

At that point we are about as secure as we can be than SSH.

Admin users still sensitive targets, but Oauth/LDAP passwd != System password (hopefully).

Admin could still impersonate and access some services other can’t.

Securing an Installation

STEP 4

- Restrict and Configure Jupyter itself.

- By default we assume users are not malicious once logged-in but you might want to control environment.

STEP N+

- Spawn kernels on remote machine (Enterprise Gateway)
- Use non-file based storage backend (can control “trusted” state of files).
- Monitor/Cull idle kernels, Servers.
- …

At that point Hub is **more restrictive** than user install + ssh tunnel

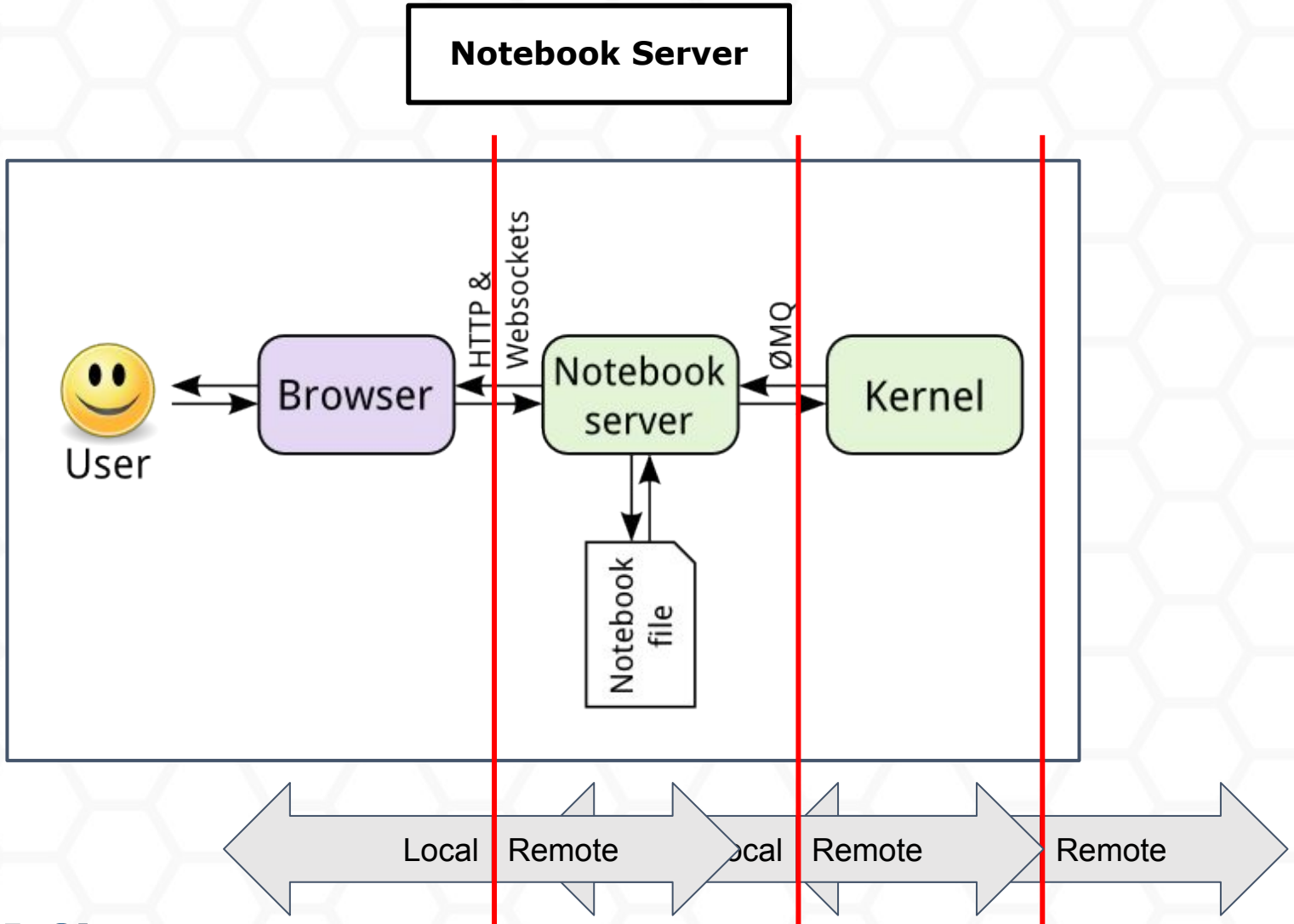
- Control/Limit plugin installation
- Monitor
- (Auto) start and stop user’s servers

Risk awareness

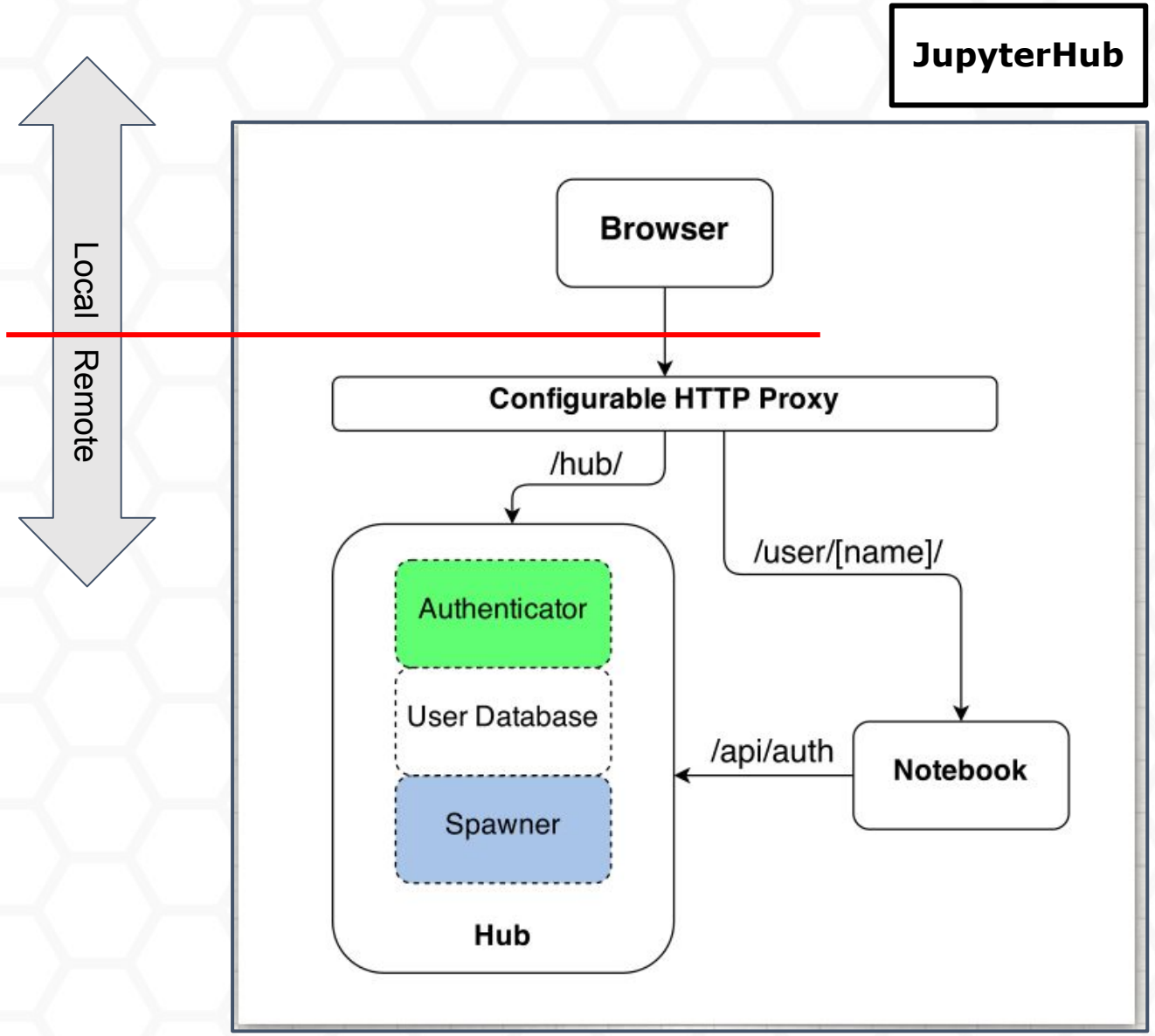
- Notebooks allow arbitrary code to be run by those you provide access to.
- Notebooks are often shared between colleagues.
- It is important to restrict access to the notebook server.
- Jupyter 4.3+ has token based auth enabled by default. Token passed in URL, Header or via login form.
- Projects are setting up the Jupyter server to be run in novel ways, such as on a supercomputer cluster.
-

Hands-On Exercises

Jupyter Security Boundaries



Jupyter Security Boundaries



TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

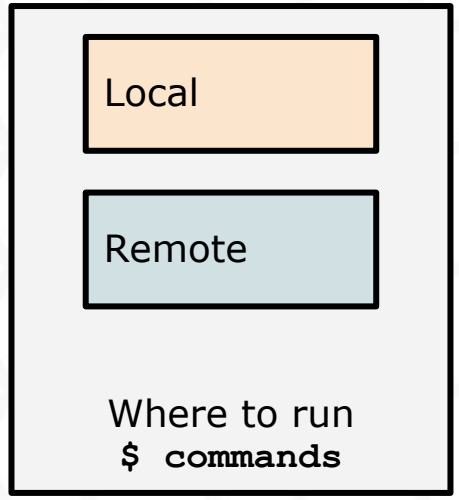
Hands on Exercises

Hosts:

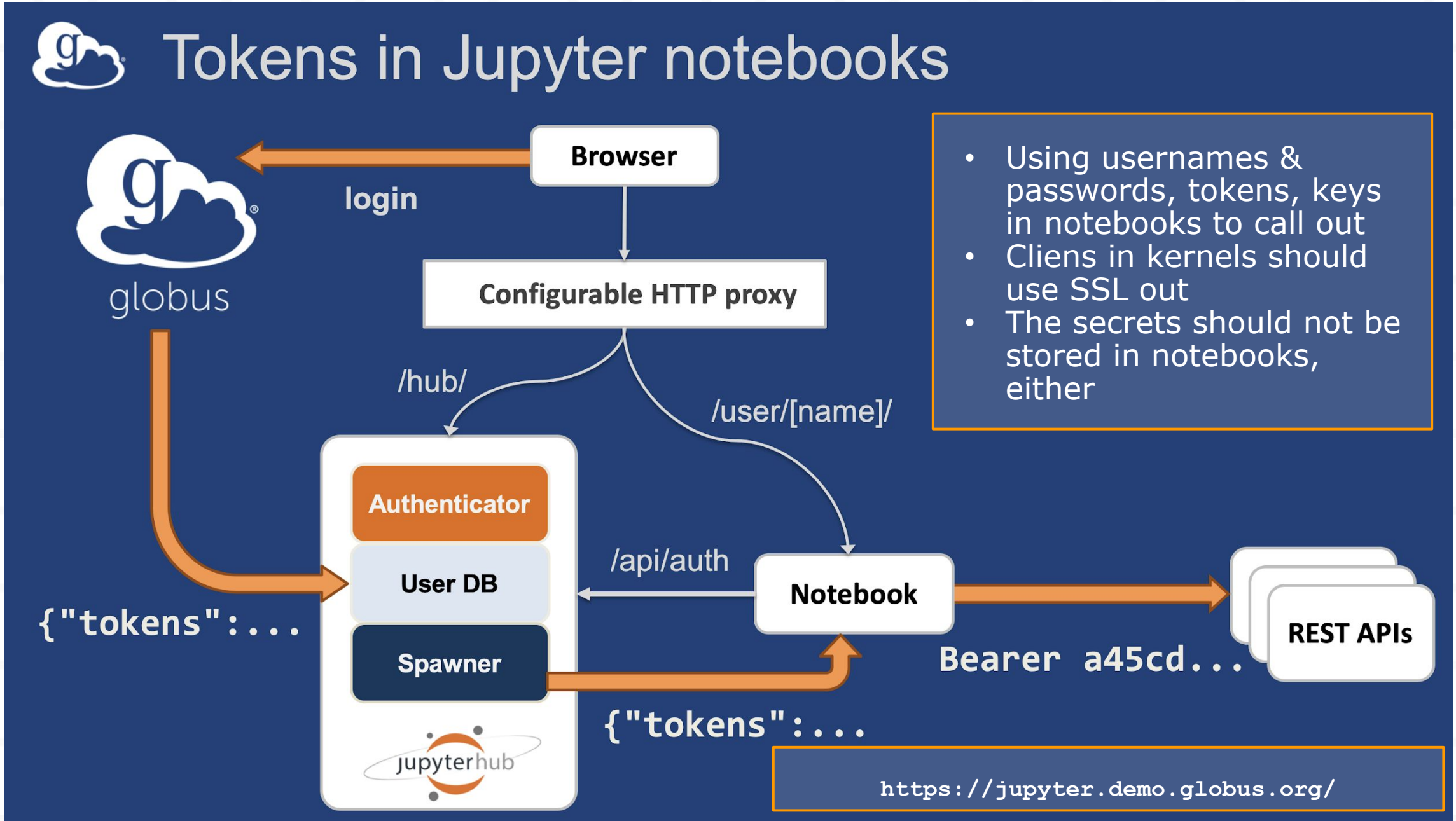
- Go to spreadsheet and pick a hostname and IP, put your name down
- Ubuntu 18.04 LTS
- Hostnames are `trusted-ci[1-N].globus-training.net`

User accounts:

- "`campusadmin`" (passwordless sudo permissions) is for configuring Jupyter
- "`researcher`" (unprivileged) is intended to demonstrate user access
- Password will be on the whiteboard

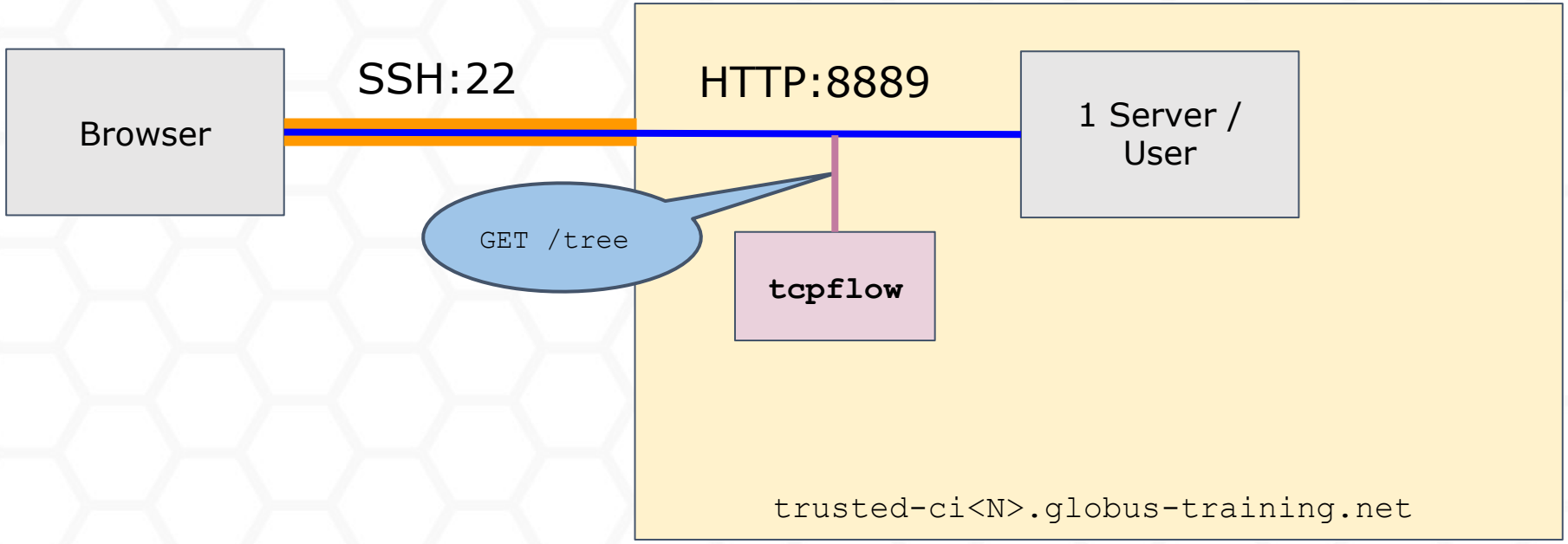


Driving Security Motivation: Calling APIs



- Using usernames & passwords, tokens, keys in notebooks to call out
- Cliens in kernels should use SSL out
- The secrets should not be stored in notebooks, either

Single User Notebook Server



Access Remote Notebook Server

SSH to the server

```
local:~$ ssh campusadmin@trusted-ci<N>.globus-training.net
```

Start a single user Jupyter notebook server

```
campusadmin@trusted-ci<N>:~$ jupyter notebook --no-browser --port=8889
```

Run `tcpflow`

"**`tcpflow`** is a program that captures data transmitted as part of TCP connections (flows), and stores the data in a way that is convenient for protocol analysis and debugging."

It's like `tcpdump` with readable text output. --Rick

Basic command

```
campusadmin@trusted-ci<N>:~$ sudo tcpflow -p -c -i lo port <port>
```

I don't recommend leaving this running

Let's run this on the HTTP port

```
campusadmin@trusted-ci<N>:~$ sudo tcpflow -p -c -i lo port 8889 | grep 'password='
```


https://github.com/simsong/tcpflow

Access Remote Notebook Server

Port forward from your laptop the server

```
local:~$ ssh -N -f -L localhost:8888:localhost:8889 \
  campusadmin@trusted-ci<n>.globus-training.net
```

Look for this line in your Jupyter server output:

```
http://localhost:8889/?token=a0fac14de491e7eb80ba8ab0e6e1ee16ad6ef4532d1316ca
```

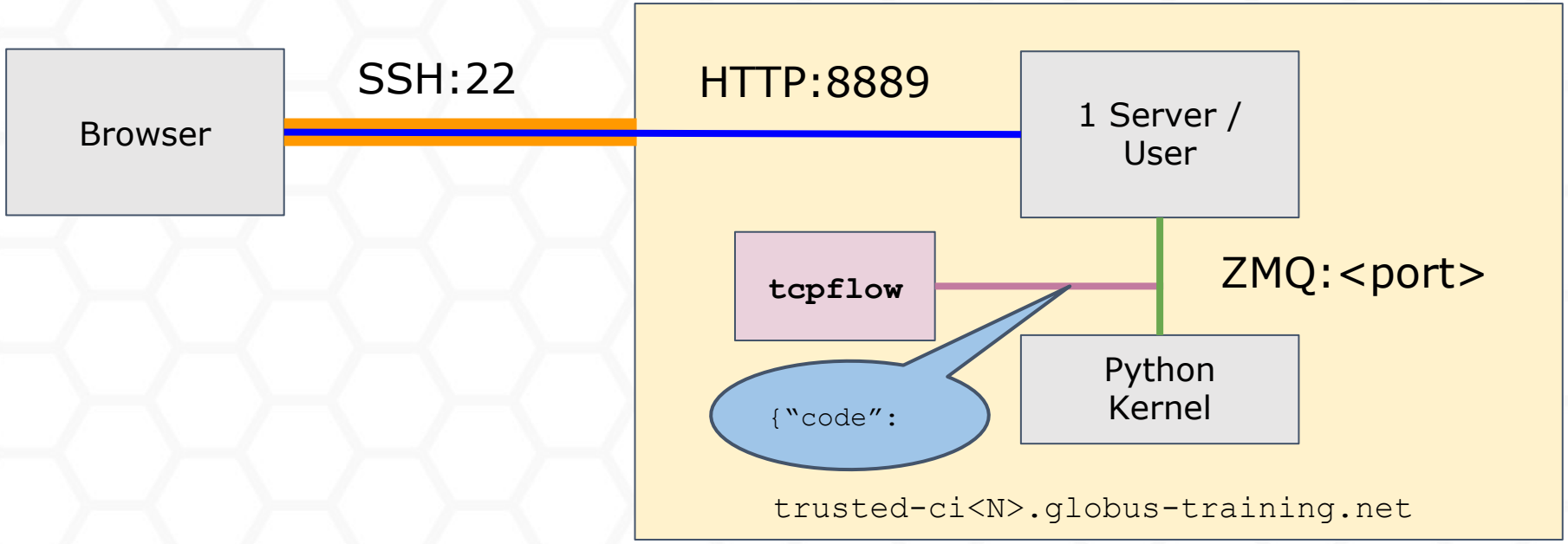
Change 8889 to 8888 & put that line in a browser

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

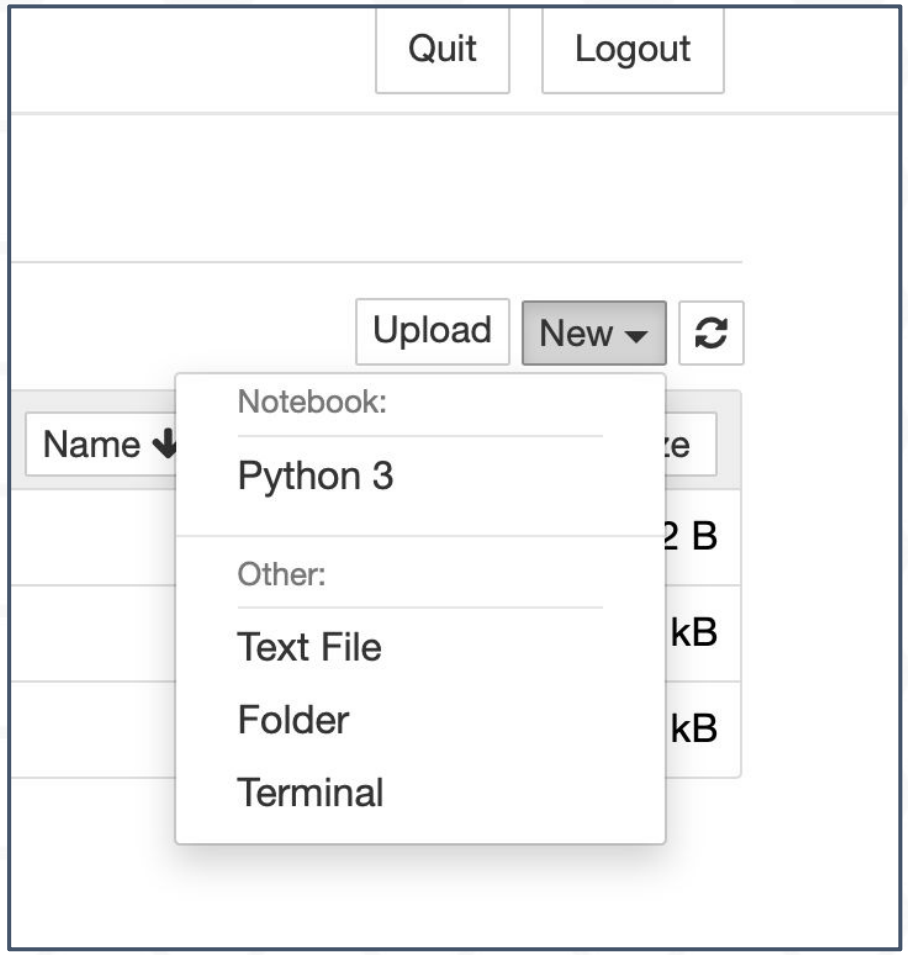
tcpflow HTTP Capture

```
campusadmin@trusted-ci1:~$ sudo tcpflow -p -c -i lo port 8889
tcpflow: listening on lo
127.000.000.001.43882-127.000.000.001.08889: GET
/?token=1297a2b4dc6e3d7ec8a95f6443acff930f1f1920f3172ac8 HTTP/1.1
Host: localhost:8888
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Cookie:
username-localhost-8888="2|1:0|10:1571099165|23:username-localhost-8888|44:MzlkZTk1NjNhYjBmNDU2MWEyYmV
mOWE5MjhlNzlmOTE=|8a9b04bf9af42ef7d84da55c9f3cc06ac1b91a330608ca2378a1743d7ed617cb";
_xsrf=2|25101594|37f123c20fef78ecf3cb7bb75bb7b5f1|1569445998
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15) AppleWebKit/605.1.15 (KHTML, like Gecko)
Version/13.0.2 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive


127.000.000.001.08889-127.000.000.001.43882: HTTP/1.1 302 Found
Server: TornadoServer/6.0.3
Content-Type: text/html; charset=UTF-8
Date: Tue, 15 Oct 2019 05:04:56 GMT
Location: /tree?token=1297a2b4dc6e3d7ec8a95f6443acff930f1f1920f3172ac8
Content-Length: 0
```

Notebook Server <-> Kernel Traffic



Start a Kernel & Notebook
Find its Ports



Connection information is in the notebook

```
In [1]: %connect_info
{
  "shell_port": 33479,
  "iopub_port": 49461,
  "stdin_port": 40731,
  "control_port": 37079,
  "hb_port": 49495,
  "ip": "127.0.0.1",
  "key": "f81be3e1-ac51af8526582e50f89670f0",
  "transport": "tcp",
  "signature_scheme": "hmac-sha256",
  "kernel_name": ""
}
```

Start a Kernel & Notebook Find its Ports

On host, connect info also in file.

Keep file ACLs limited to user

If “you” get connect info and **key**

“you” can connect and execute code as user

Connection information is also in the user home directory

```
campusadmin@trusted-ci1:~$ cd .local/share/jupyter/runtime/
campusadmin@trusted-ci1:~$ ls -1tr
notebook_cookie_secret
nbserver-8485.json
nbserver-8485-open.html
nbserver-9738.json
nbserver-9738-open.html
kernel-e6757c17-0699-4e15-8ac7-4596c7178844.json
campusadmin@trusted-ci1:~$ cat kernel-*
{
  "shell_port": 33617,
  "iopub_port": 56051,
  "stdin_port": 52759,
  "control_port": 55021,
  "hb_port": 50873,
  "ip": "127.0.0.1",
  "key": "1a917676-c05a83835fc282317c41975d",
  "transport": "tcp",
  "signature_scheme": "hmac-sha256",
  "kernel_name": ""
}
campusadmin@trusted-ci1:~$
```

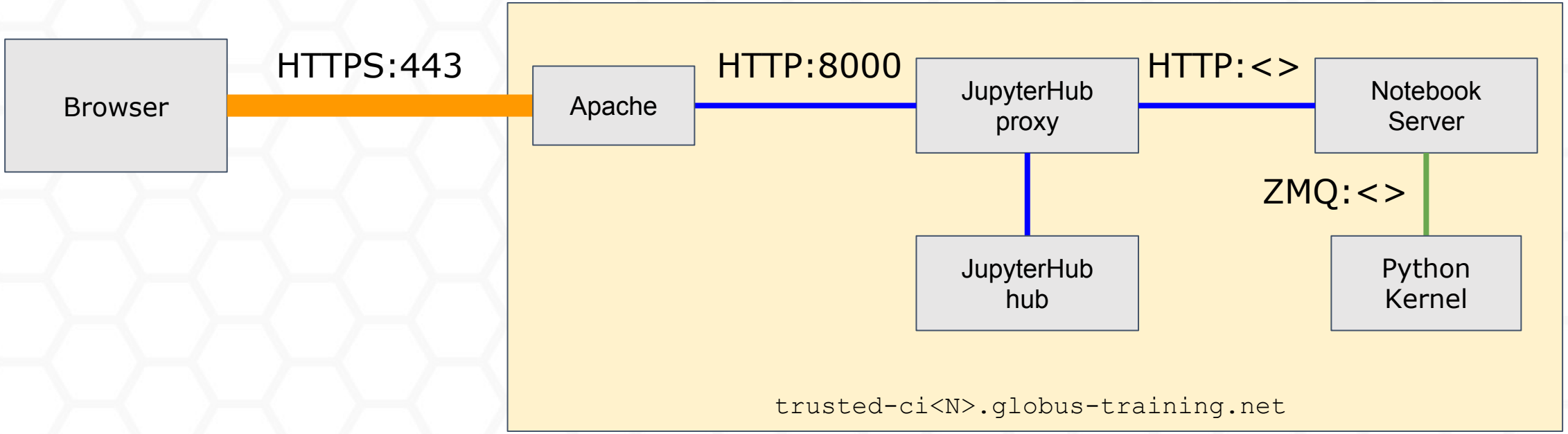
Listen to Kernel Input and Output

```
In [9]: import os
        print('because no one ever stores secrets in the environment')
        for k in ('USER', 'PWD', 'HOME'):
            print('{}: {}'.format(k, os.environ[k]))

        because no one ever stores secrets in the environment
        USER: campusadmin
        PWD: /home/campusadmin
        HOME: /home/campusadmin
```

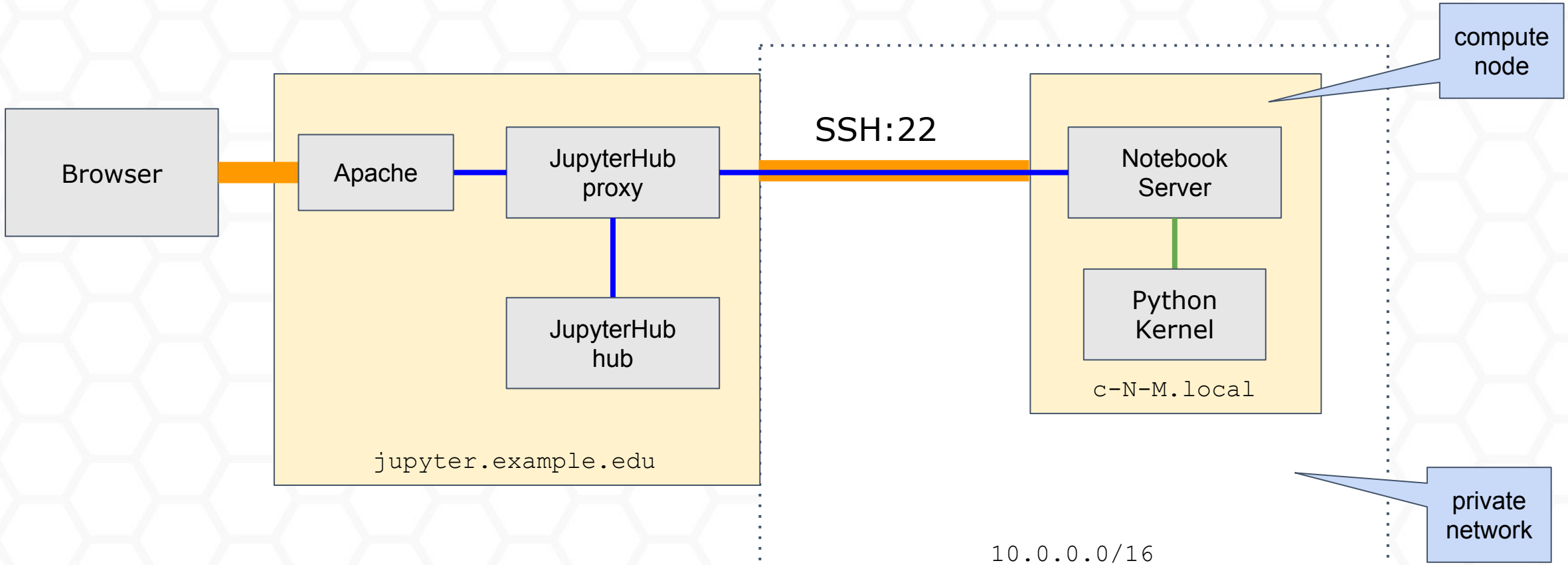
```
$ sudo tcpflow -p -c -i lo port 56051
tcpflow: listening on lo

127.000.000.001.56051-127.000.000.001.43522:
... .snip. ...
{"code":"import os\nprint('because no one ever stores secrets in the environment')\nfor k
in ('USER', 'PWD', 'HOME'):\n    print('{}: {}'.format(k,
os.environ[k]))","execution_count":10}
... .snip. ...
{"name":"stdout","text":"because no one ever stores secrets in the environment\nUSER:
campusadmin\nPWD: /home/campusadmin\nHOME: /home/campusadmin\n"}
```

JupyterHub Starting Point

<> = “random” ports



JupyterHub HPC/Cluster Model



Setting Up JupyterHub

Going to set up JupyterHub available at

`https://trusted-ci<N>.globus-training.net/jhub/`

These hosts have Let’s Encrypt certs enabled

Apache config /etc/apache2/sites-enabled/000-default-le-ssl.conf

Apache Rewrites & Proxies

Add to `/etc/apache2/sites-enabled/000-default-le-ssl.conf`
Add to

```
RewriteEngine On
RewriteCond %{HTTP:Connection} Upgrade [NC]
RewriteCond %{HTTP:Upgrade} websocket [NC]

RewriteRule /jhub/(.*) ws://127.0.0.1:8000/jhub/$1 [P,L]
RewriteRule /jhub/(.*) http://127.0.0.1:8000/jhub/$1 [P,L]

<Location "/jhub/">
  # preserve Host header to avoid cross-origin problems
  ProxyPreserveHost on
  # proxy to JupyterHub
  ProxyPass         http://127.0.0.1:8000/jhub/
  ProxyPassReverse  http://127.0.0.1:8000/jhub/
</Location>
```

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Bounce Apache

```
sudo systemctl restart apache2.service
```

JupyterHub Config & Start

SSH to the server

```
local:~$ ssh campusadmin@trusted-ci<N>.globus-training.net
```

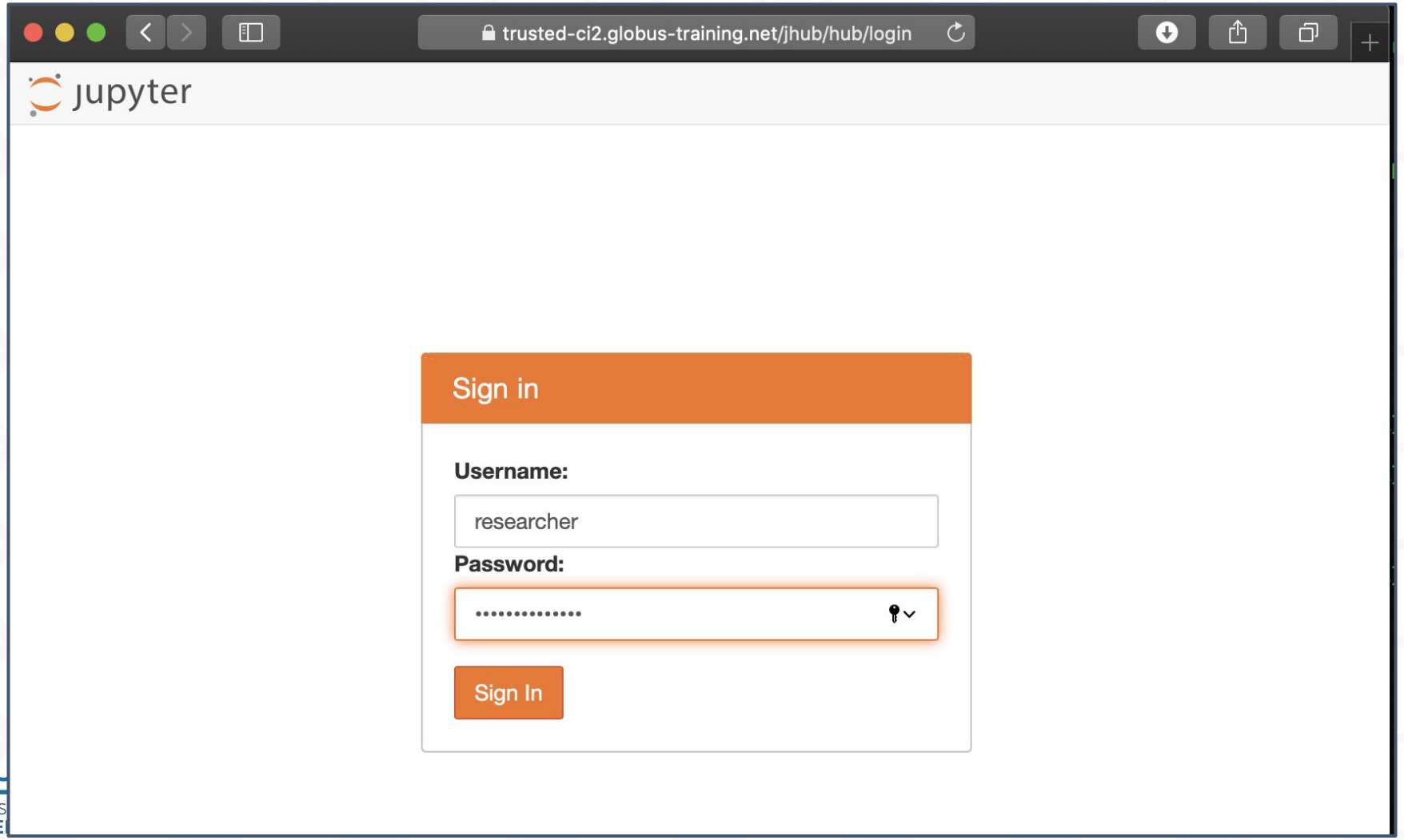
Create default JupyterHub config and start JupyterHub

```
$ cd /etc/jupyterhub
$ sudo jupyterhub --generate-config -f \
   /etc/jupyterhub/jupyterhub_config.py
$ sudo jupyterhub --ip 127.0.0.1 --port 8000 --base-url \
  '/jhub/' --config=/etc/jupyterhub/jupyterhub_config.py
```

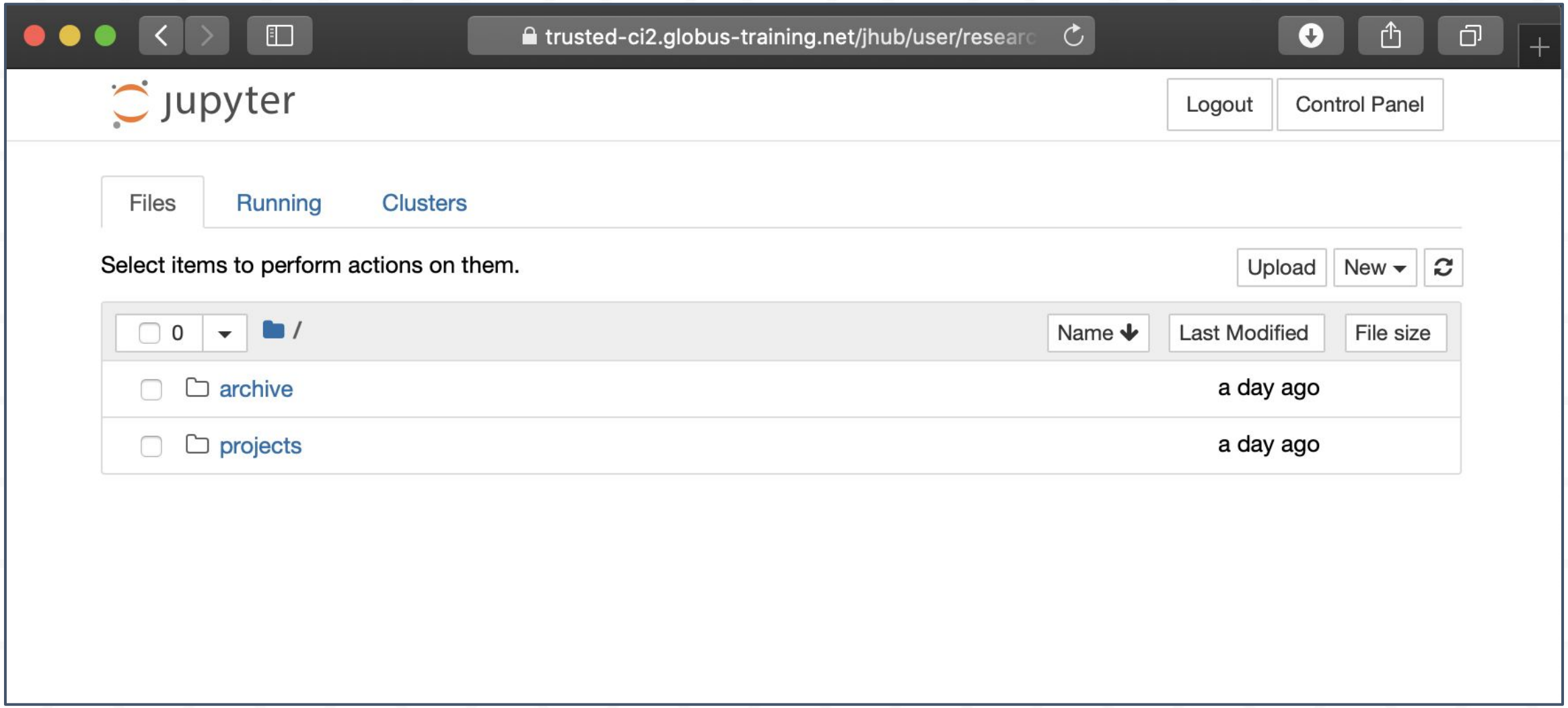
TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

- We're going to be changing the JupyterHub config a lot.
- Should go into a service script.

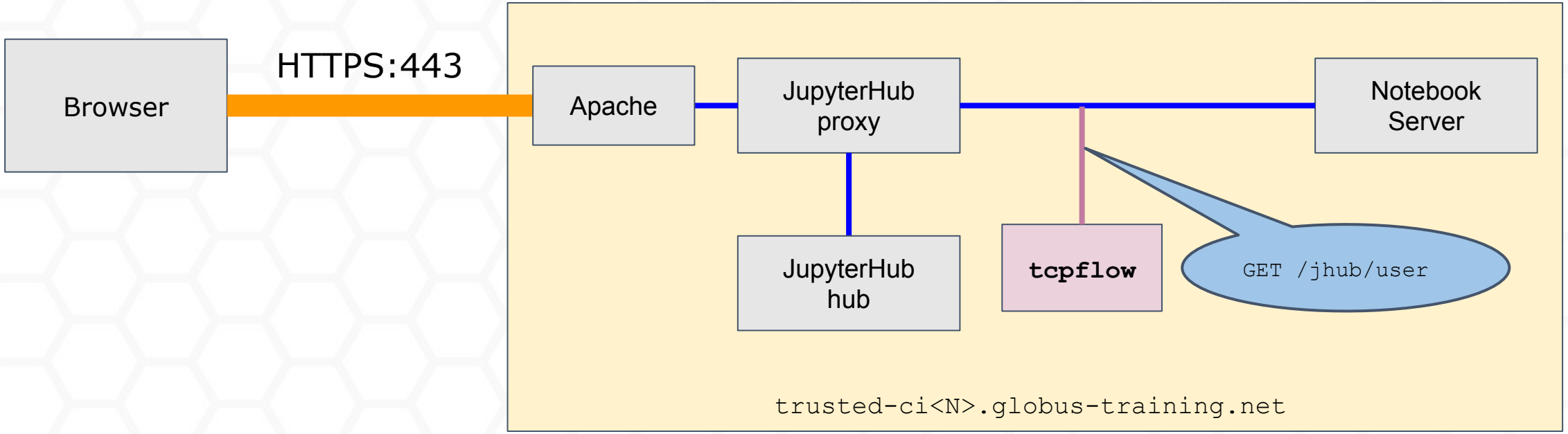
Login As Researcher



Login As Researcher



JupyterHub Internal Traffic



<> = “random” ports

Find the Proxy Port

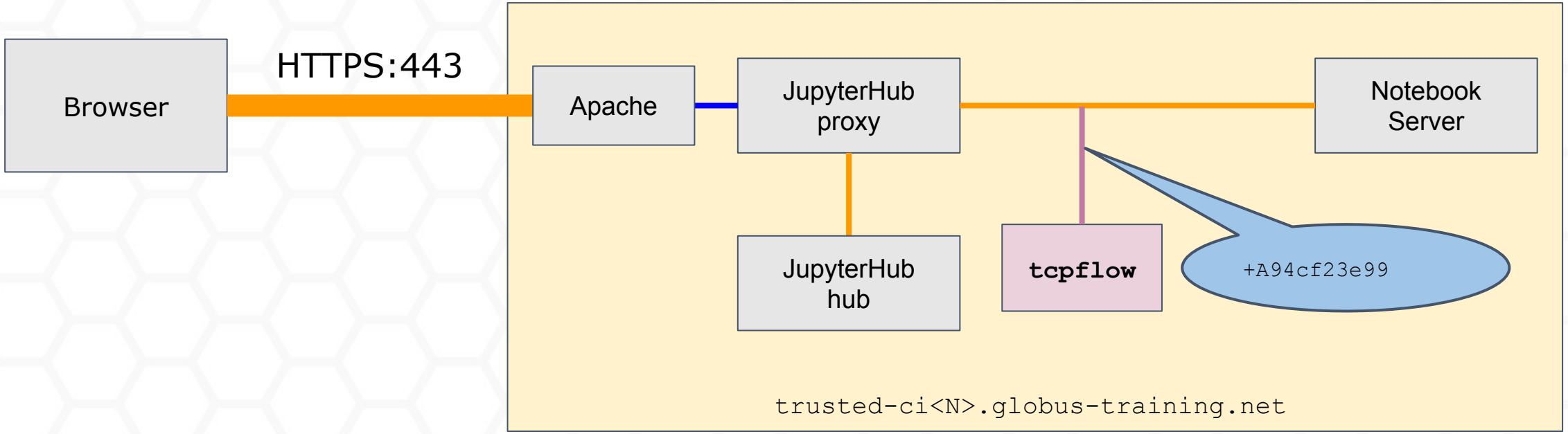
```
[I 2019-10-15 01:49:24.982 JupyterHub proxy:261] Adding user researcher to proxy /jhub/user/researcher/
=> http://127.0.0.1:41009
01:49:24.984 [ConfigProxy] info: Adding route /jhub/user/researcher -> http://127.0.0.1:41009
01:49:24.985 [ConfigProxy] info: Route added /jhub/user/researcher -> http://127.0.0.1:41009
```

Listen...

```
campusadmin@trusted-ci2:~$ sudo tcpflow -p -c -i lo port 41009
tcpflow: listening on lo

127.000.000.001.50114-127.000.000.001.41009: GET /jhub/user/researcher/api/sessions?_=1571122288275
HTTP/1.1
x-forwarded-proto: http
x-forwarded-port: 80
```

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

JupyterHub Internal SSL



<> = “random” ports

Update Config and Create Certs

Add to `/etc/jupyterhub/jupyterhub_config.py`

```
c.JupyterHub.trusted_alt_names = ['DNS:trusted-c<N>.globus-training.net', 'DNS:trusted-ci<N>']
c.JupyterHub.internal_certs_location = '/etc/jupyterhub/internal-ssl'
c.JupyterHub.internal_ssl = True
```

In `/etc/jupyterhub/`

```
$ sudo jupyterhub --ip 127.0.0.1 --port 8000 --base-url '/jhub/' \
    --config=/etc/jupyterhub/jupyterhub_config.py --generate-certs
[I 2019-10-15 02:16:07.663 JupyterHub app:1363] Adding CA for hub-internal:
IP:127.0.0.1;DNS:localhost;DNS:ec2-54-67-48-138.us-west-1.compute.amazonaws.com;DNS:trusted-c2.globus-tr
aining.net;DNS:trusted-ci2
[I 2019-10-15 02:16:07.889 JupyterHub app:1383] Generating signed pair for proxy-api:
IP:127.0.0.1;DNS:localhost;DNS:trusted-c2.globus-training.net;DNS:trusted-ci2
[I 2019-10-15 02:16:07.978 JupyterHub app:1383] Generating signed pair for proxy-client:
IP:127.0.0.1;DNS:localhost;DNS:trusted-c2.globus-training.net;DNS:trusted-ci2
[I 2019-10-15 02:16:08.143 JupyterHub app:2301] Certificates written to directory
`/etc/jupyterhub/internal-ssl`
```

Update Config and Create Certs

Start JupyterHub

```
$ cd /etc/jupyterhub
$ sudo jupyterhub --ip 127.0.0.1 --port 8000 --base-url \
  '/jhub/' --config=/etc/jupyterhub/jupyterhub_config.py
```

Browse to

```
https://trusted-ci<N>.globus-training.net/jhub/
```

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Me or JH Bug? User Cert Di rship

```
    'certfile "%s" does not exist' % self
ValueError: certfile "/home/resea                    does not exist
```

```
$ sudo ls -al /ho
total 40
drwxr-xr-x 7 resea
drwxr-xr-x 6 root
-rw-r--r-- 1 resear
-rw-r--r-- 1 researc
drwx------ 2 researc                                  ache
drwx------ 3 root                                  .27 .jupyterhub
drwx------ 3 research                              15 01:49 .local
-rw-r--r-- 1 researche                            807 Oct 13 22:45 .profile


$ sudo chown researcher:researcher /home/researcher/.jupyterhub
```

FIXED!

chown jupyterhub dir in user home #2785

Merged minrk merged 1 commit into jupyterhub:master from rpwagner:master 14 hours ago

https://github.com/jupyterhub/jupyterhub/pull/2785

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Find the Proxy Port

```
[I 2019-10-15 02:31:20.224 JupyterHub proxy:261] Adding user researcher to proxy /jhub/user/researcher/
=> https://127.0.0.1:36193
02:31:20.231 [ConfigProxy] info: Adding route /jhub/user/researcher -> https://127.0.0.1:36193
02:31:20.231 [ConfigProxy] info: Route added /jhub/user/researcher -> https://127.0.0.1:36193
```

Listen...

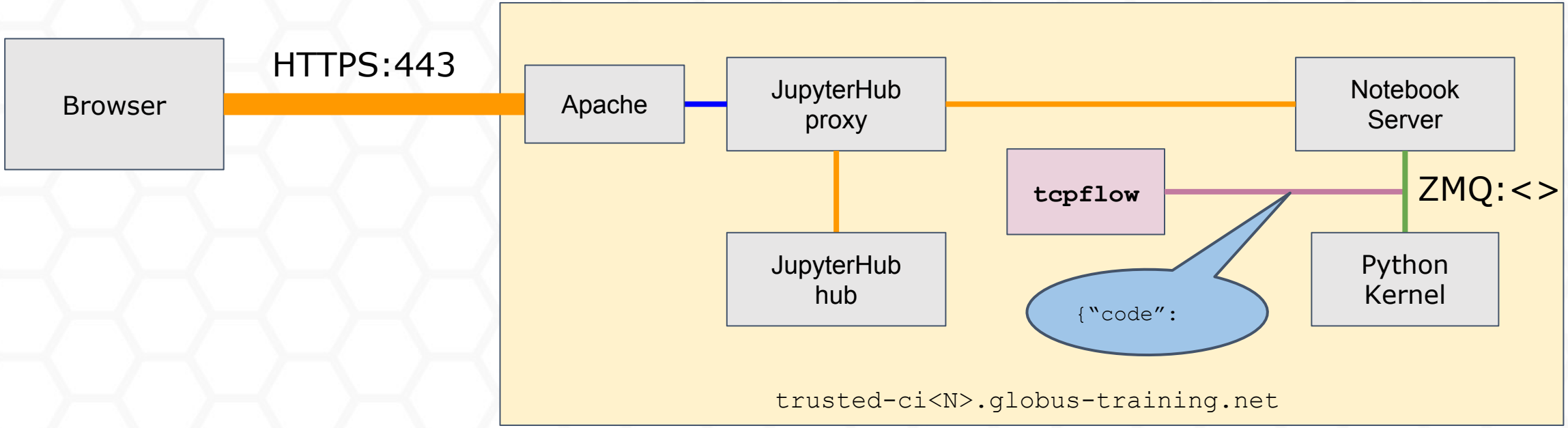
```
$ sudo tcpflow -p -c -i lo port 36193
tcpflow: listening on lo
```

Success!

```
$ sudo tcpflow -p -c -i lo port 36193
tcpflow: listening on lo
127.000.000.001.47846-127.000.000.001.36193: ]!='3|DqyUn/+0,'g(k$
jih98762.*&=5#@?>32101-)%</i$"trusted-ci2.globus-training.net
 3t
127.000.000.001.36193-127.000.000.001.47846: =9w*=b;g
0H1013073119Z010Uuser-researcher0"0
vWF%i<
EVK-z$FN}
-/qB/)~F5/|~S@|[HbUxRd`zj694cQUf'L|=Z}]#g/C3I|4FRL@<o*fTUj=P]_[O`UsHMgx4}4v\Ky?of%g]Ne?n0l0
U%0++0HUA0?localhosttrusted-ci2.globus-training.net:0%Ied`0V,;?.i^akNLp[?B.$rh$7IWc4Sus~=BOaMIAV74mdb=e>
4W|Ai^^=8;q1KFztPiPOk>{r:[<^3;5b5e8M9k%
6@.o:g[wGa-3*#mLkYc#pecT>`qQ#E;x9rhq_Z*!.#1>x.O3),N[Z=BZi])7DFm$;9{`6%W2G3"$T8C-A=`#C!Go:

0H1013072429Z010Uproxy-client0"00.001.36193: #000
B#\ <=LWBX3?lyW=v}jT-C@[&SxUnFl+zk\zw|q[PFlkMk$Xvk=#Th=-s_?d{8w0w^}$k@ck0W?]h-"mT4.SX`+S#wQ9C
```

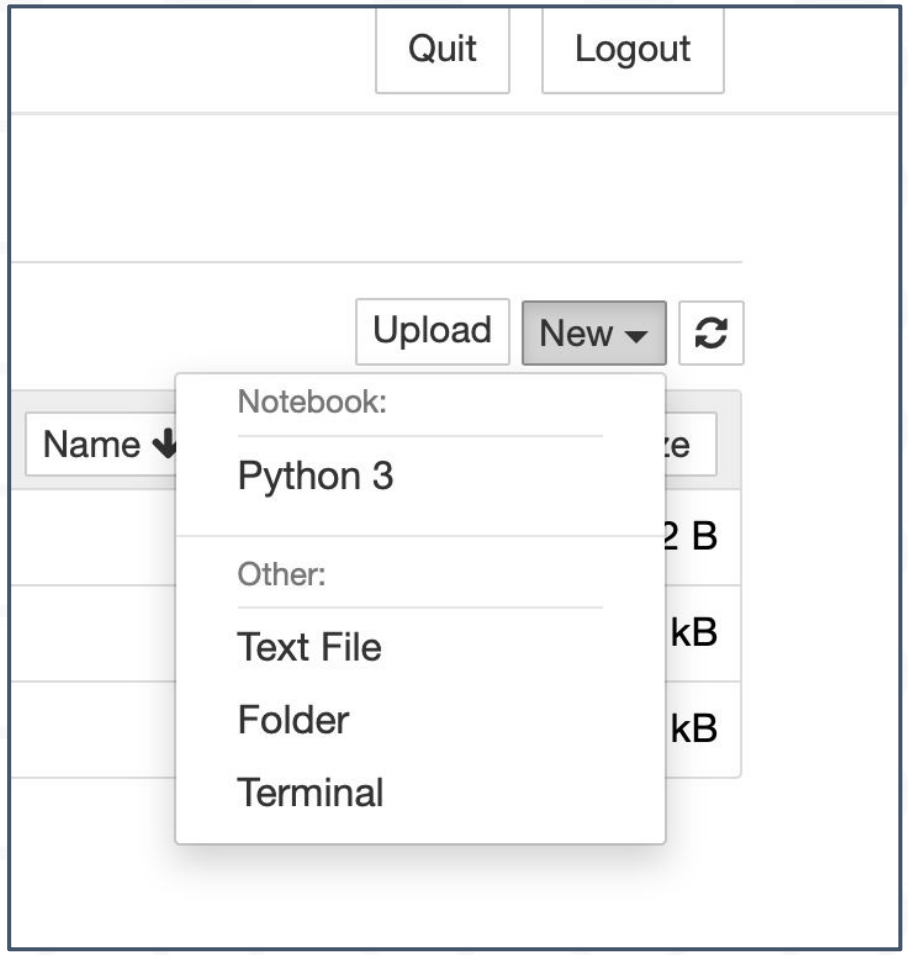
TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

JupyterHub Internal SSL



<> = "random" ports

Start a Kernel & Notebook
Find its Ports




```
$ sudo ls -1tr /home/researcher/.local/share/jupyter/runtime/
nbserver-8485.json
nbserver-8485-open.html
nbserver-9738.json
nbserver-9738-open.html
kernel-e6757c17-0699-4e15-8ac7-4596c7178844.json
$ sudo cat \
  /home/researcher/.local/share/jupyter/runtime/kernel-...
{
  "shell_port": 45317,
  "iopub_port": 54279,
  "stdin_port": 47461,
  "control_port": 58485,
  "hb_port": 46187,
  "ip": "127.0.0.1",
  "key": "92b993fc-759de490904af0c952884cbd",
  "transport": "tcp",
  "signature_scheme": "hmac-sha256",
  "kernel_name": ""
}
```

{"code":

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

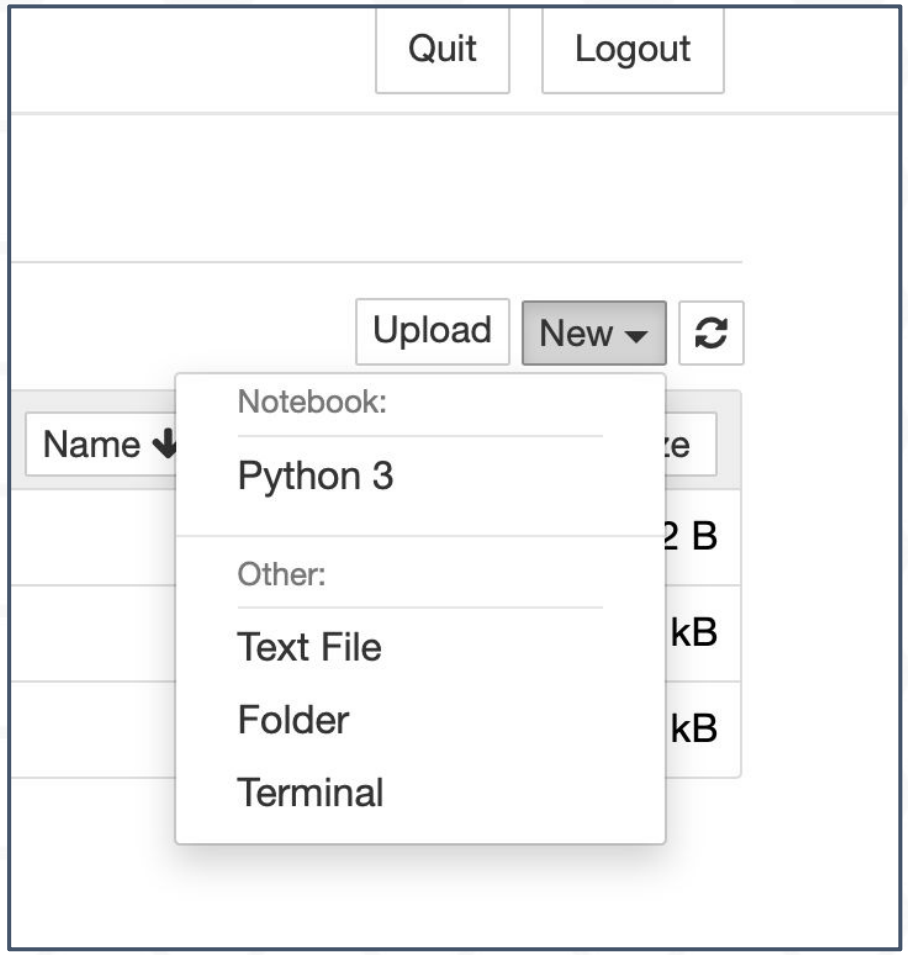
Change Kernel Communication to IPC

Add to /etc/jupyterhub/jupyterhub_config.py

```
c.Spawner.args = ['--transport="ipc"']
```

- IPC uses file descriptors to pass data
- Hard to intercept traffic
- Does not work between hosts
- ZeroMQ config for TLS available, in Jupyter soon
- Also good for single user notebook server

Start a Kernel & Notebook
Find its Ports



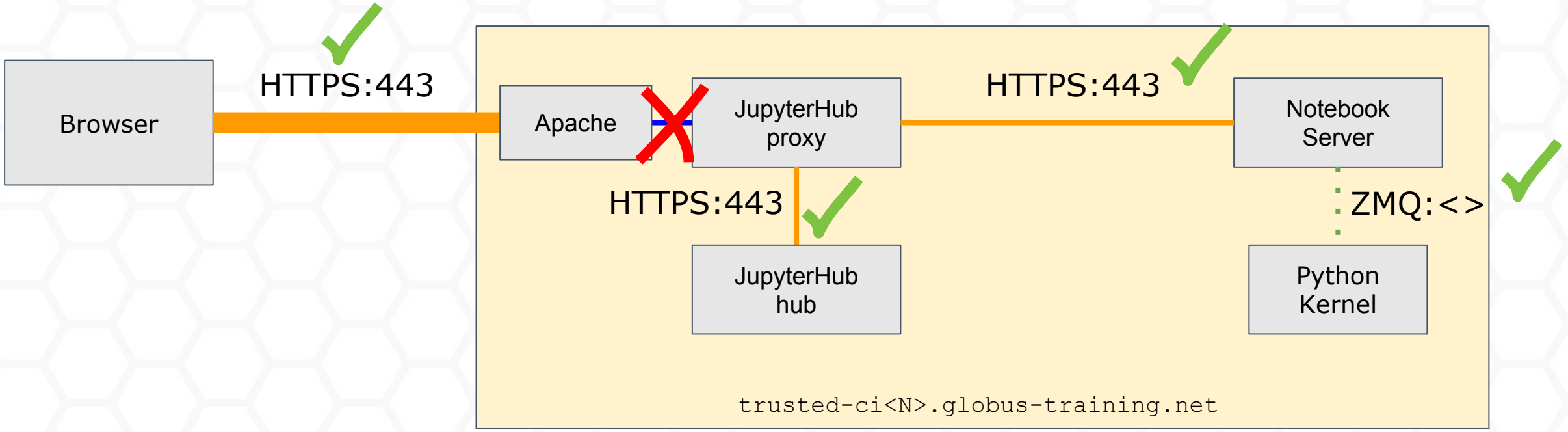

```
$ sudo ls -1tr /home/researcher/.local/share/jupyter/runtime/
nbserver-11237.json
nbserver-11237-open.html
kernel-0746a014-a85a-4959-8572-637bfcaef64d-ipc-4
kernel-0746a014-a85a-4959-8572-637bfcaef64d-ipc-3
kernel-0746a014-a85a-4959-8572-637bfcaef64d-ipc-2
kernel-0746a014-a85a-4959-8572-637bfcaef64d-ipc-1
kernel-0746a014-a85a-4959-8572-637bfcaef64d.json
kernel-0746a014-a85a-4959-8572-637bfcaef64d-ipc-5
$ sudo cat
/home/researcher/.local/share/jupyter/runtime/kernel-...
{
  "shell_port": 1,
  "iopub_port": 2,
  "stdin_port": 3,
  "control_port": 4,
  "hb_port": 5,
  "ip":
"/home/researcher/.local/share/jupyter/runtime/kernel-0746a014-
a85a-4959-8572-637bfcaef64d-ipc",
  "key": "1bc70c05-808f5bec9272a64f66967ac3",
  "transport": "ipc",
  "signature_scheme": "hmac-sha256",
  "kernel_name": ""
}
```

{"code":

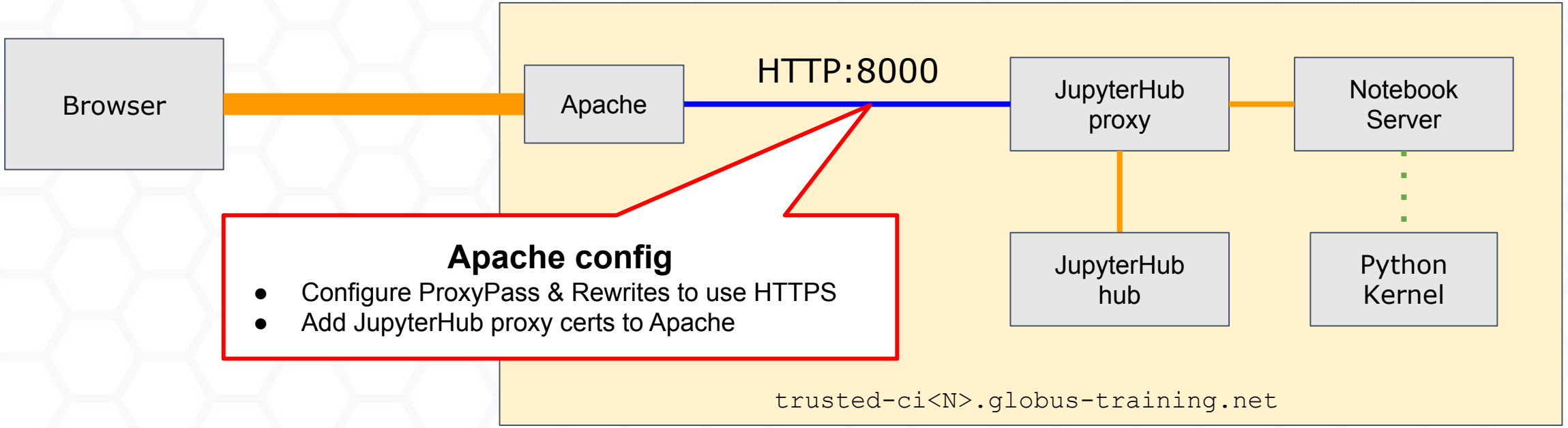
{"code":

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

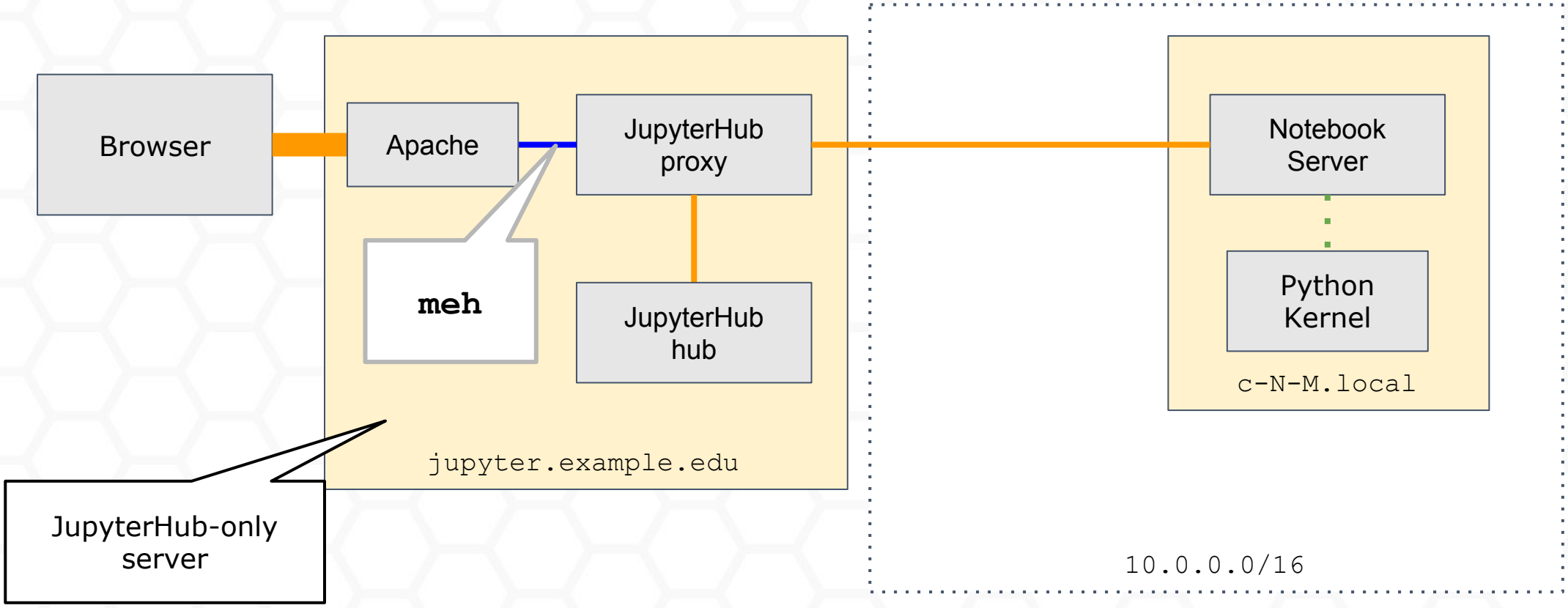
JupyterHub Internal SSL



Homework: Apache <-> JupyterHub



JupyterHub HPC/Cluster Model



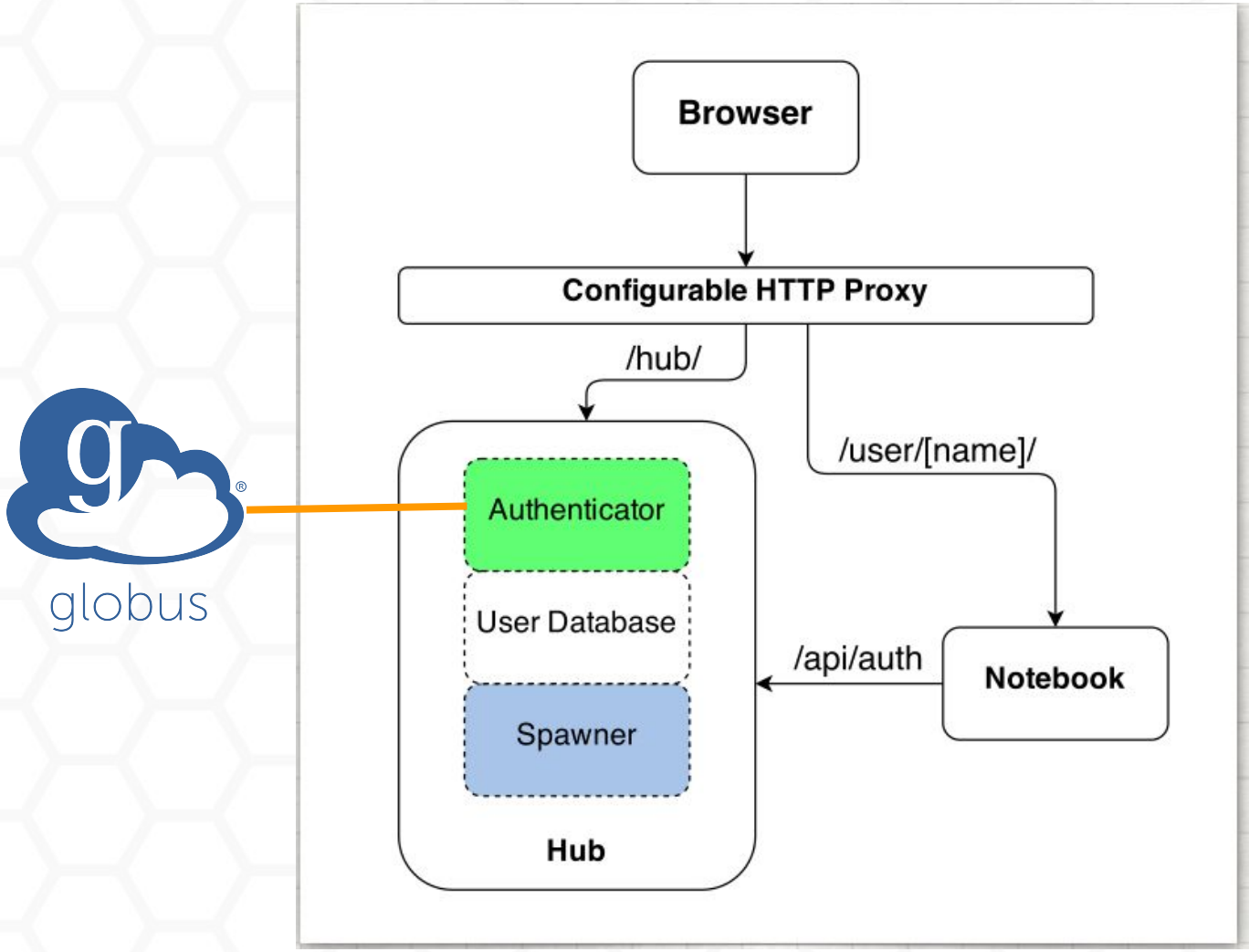
TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

JupyterHub Authenticators

Default is PAM (system username & pass)

Outsource authentication to OAuth

- Auth0
- Bitbucket
- CILogon
- GitHub
- GitLab
- Globus
- Google
- MediaWiki
- Okpy
- OpenShift
- Generic (campus)

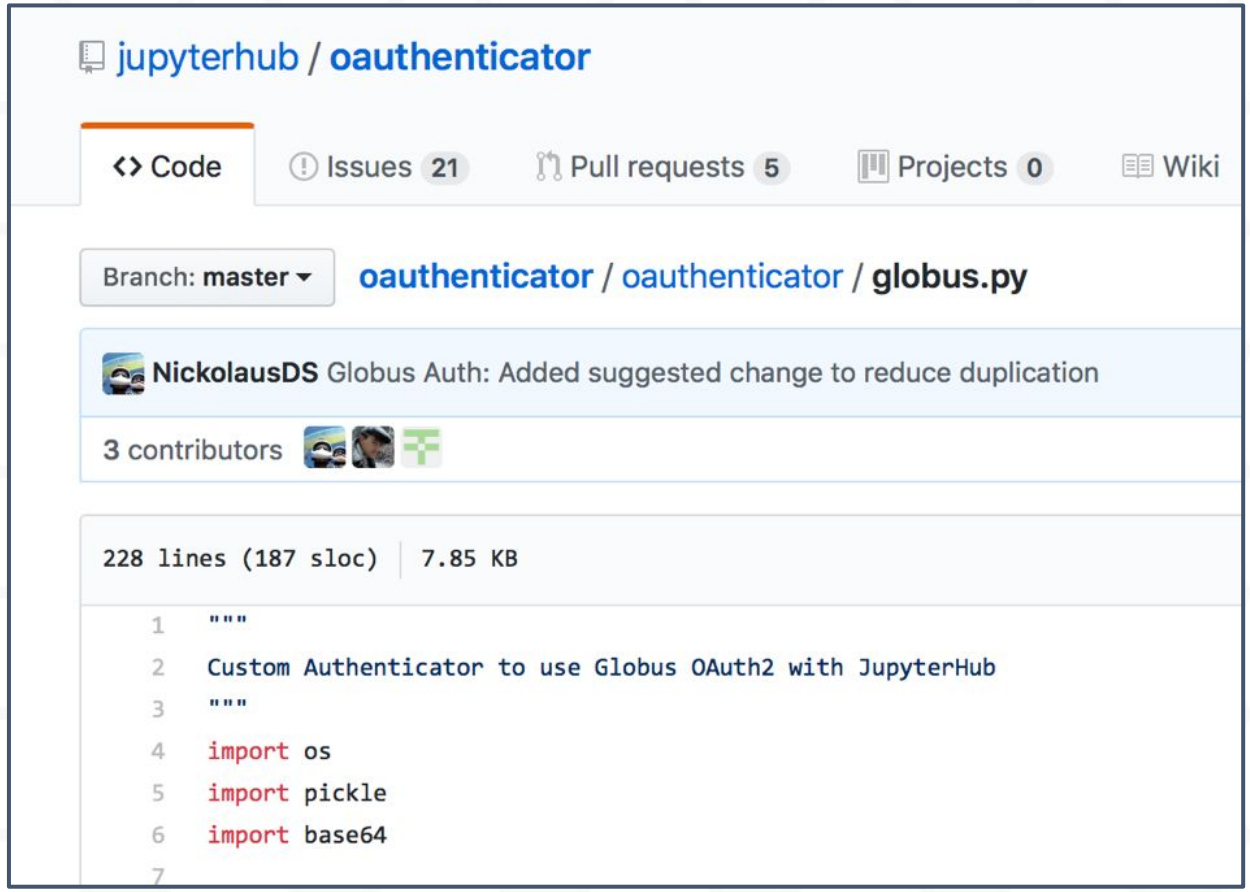


JupyterHub Globus OAuthenticator

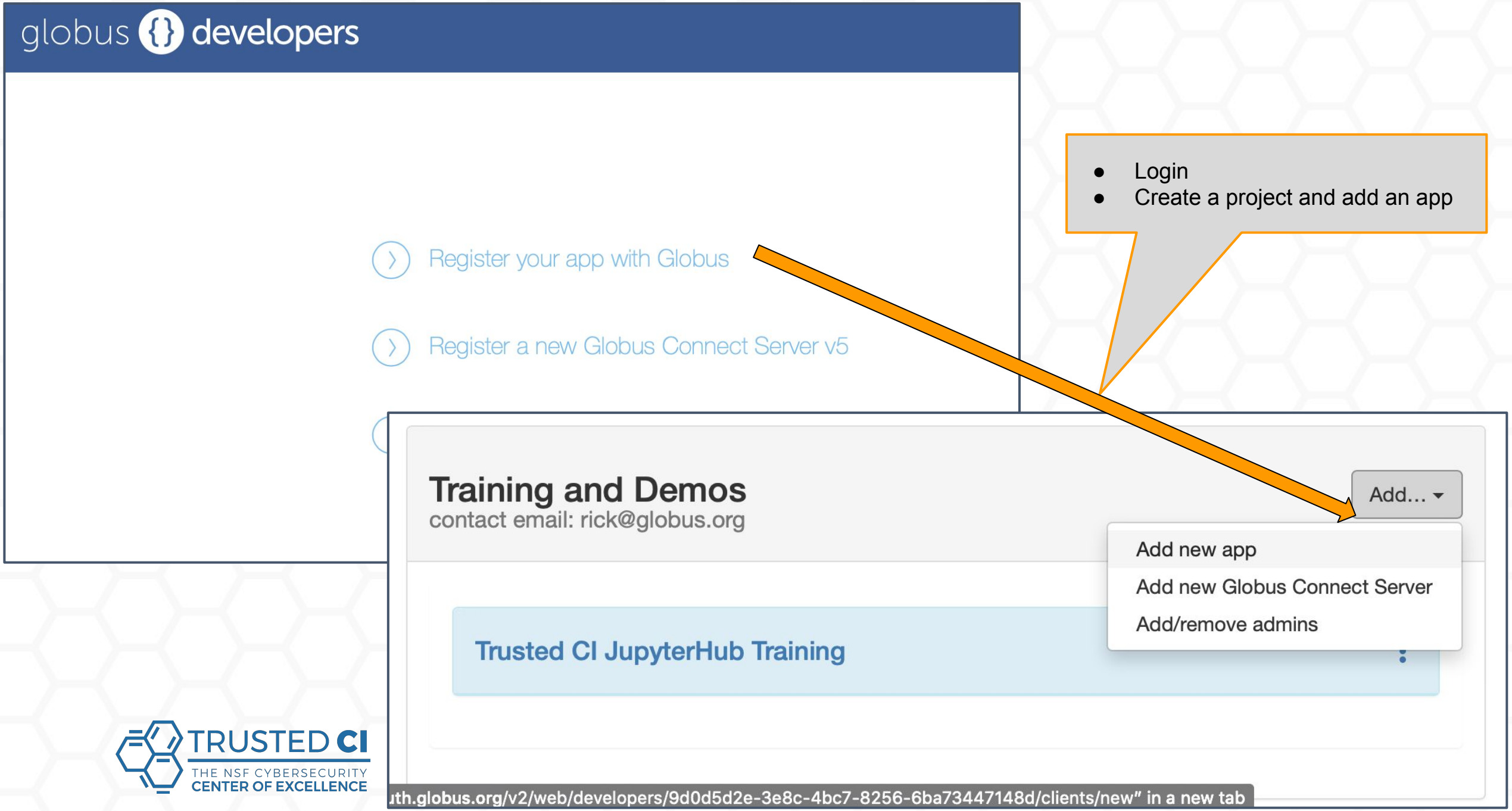
- Steps are very similar for all OAuth providers:
- Register app, get client ID and secret

For this demo:

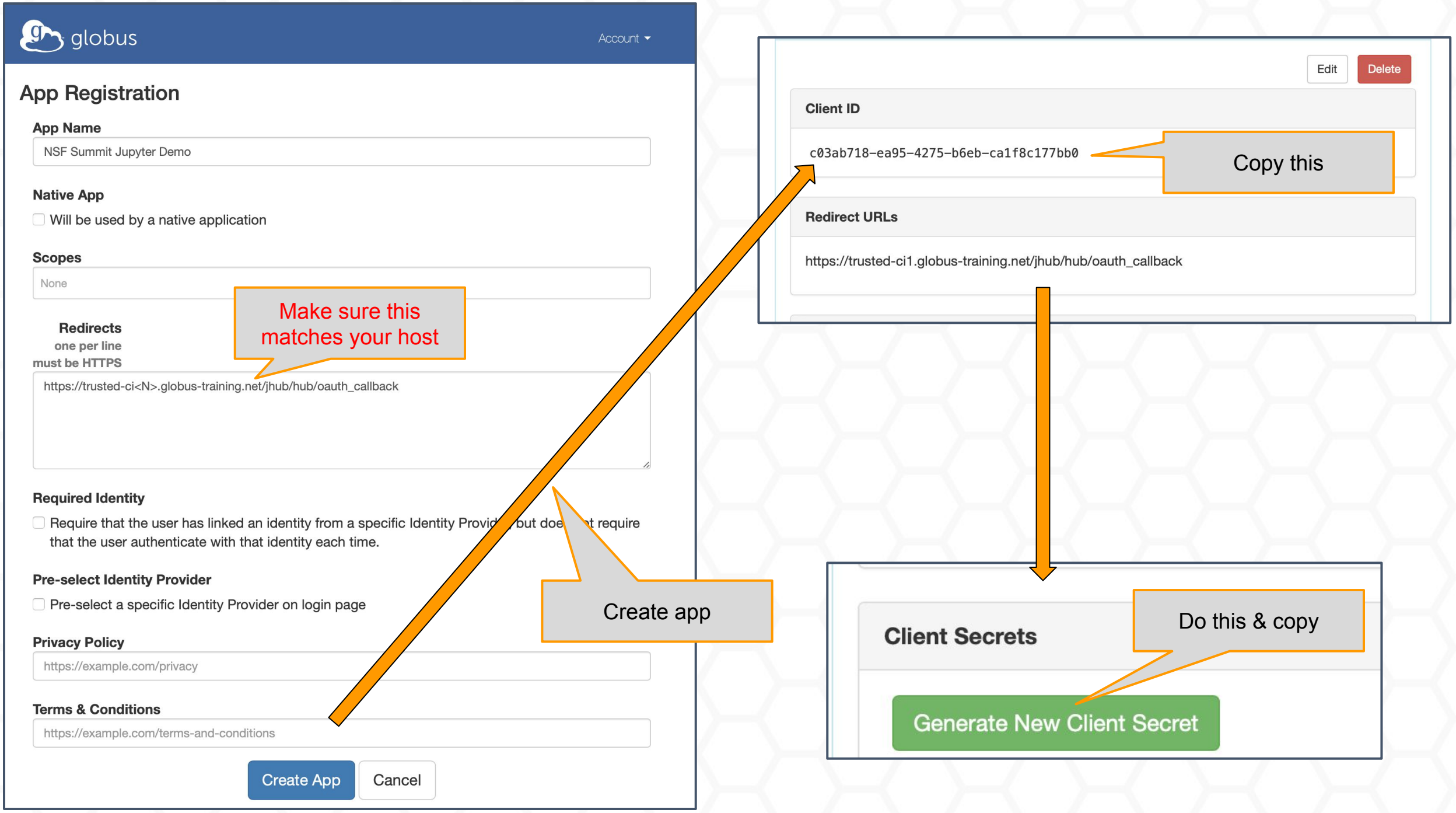
- Go to https://developers.globus.org
- Set name, callback URL
- Get client ID, client secret



TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE



TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE


globus
Account
App Registration
App Name
NSF Summit Jupyter Demo
Native App
Will be used by a native application
Scopes
None
Redirects
one per line
must be HTTPS
https://trusted-ci<N>.globus-training.net/jhub/hub/oauth_callback
Make sure this matches your host
Required Identity
Require that the user has linked an identity from a specific Identity Provid but doe t require that the user authenticate with that identity each time.
Pre-select Identity Provider
Pre-select a specific Identity Provider on login page
Privacy Policy
https://example.com/privacy
Terms & Conditions
https://example.com/terms-and-conditions
Create App
Cancel
Create app
Edit
Delete
Client ID
c03ab718-ea95-4275-b6eb-ca1f8c177bb0
Copy this
Redirect URLs
https://trusted-ci1.globus-training.net/jhub/hub/oauth_callback
Client Secrets
Do this & copy
Generate New Client Secret

JupyterHub Globus OAuthenticator

Add to /etc/jupyterhub/jupyterhub_config.py

```
from oauthenticator.globus import LocalGlobusOAuthenticator
c.JupyterHub.authenticator_class = LocalGlobusOAuthenticator
c.LocalGlobusOAuthenticator.enable_auth_state = True
c.LocalGlobusOAuthenticator.oauth_callback_url = \
     'https://trusted-ci<N>.globus-training.net/jhub/hub/oauth_callback'
c.LocalGlobusOAuthenticator.client_id = '<>'
c.LocalGlobusOAuthenticator.client_secret = '<>'
c.LocalGlobusOAuthenticator.create_system_users = True
```

We're going to create a new system account for each user
Can also explicitly map users to system accounts

```
c.Authenticator.username_map  = {
  'service-name': 'localname'
}
```

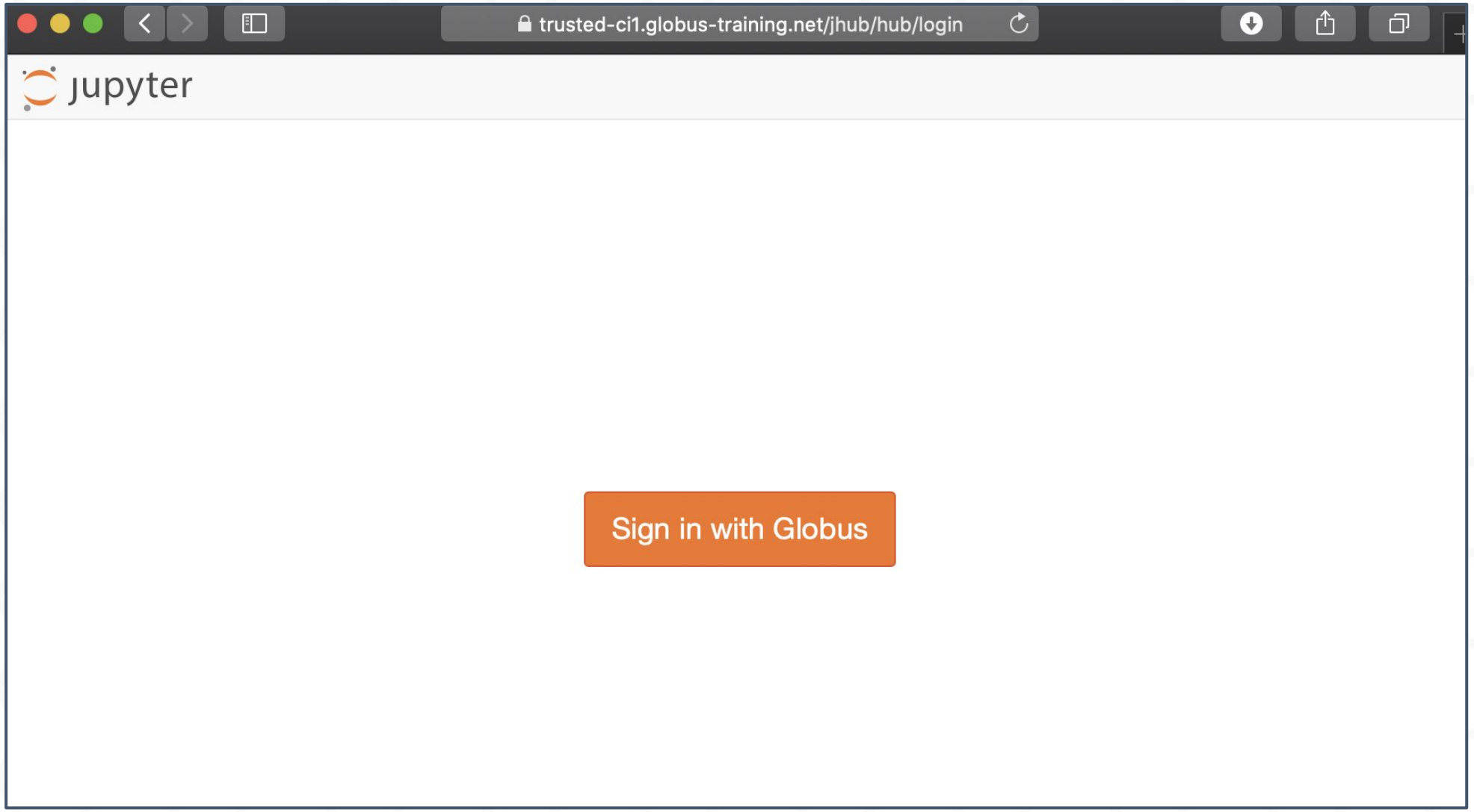
New start command to pass key for user secrets

```
sudo JUPYTERHUB_CRYPT_KEY=$(cat /etc/jupyterhub/crypt.key) jupyterhub \
   --ip 127.0.0.1 --port 8000 --base-url '/jhub/' \
  --config=/etc/jupyterhub/jupyterhub_config.py
```

JupyterHub database will get an encrypted area

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

https://trusted-ci<N>.globus-training.net/jhub/



Wrap Up:
Answers to Earlier Questions
Defining Jupyter Security Best Practices

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

What can be done to improve security in the Jupyter Community?

- Who are the stakeholders?
- What are the use cases?
- Let's start with you...
  - Why are you running Jupyter?
  - How do you run Jupyter?
  - What are the challenges?