Experience of deployment of the SWAN system on the local resources of the SPbSU

Andrey Erokhin, SPbSU
Andrey Zarochentsev, SPbSU

LUHEP is one of the laboratories in SPbSU

SPbSU

St Petersburg University Today

- 30,000 students
- 6000 academic staff
- 398 main educational programmes

© http://english.spbu.ru

Laboratory of Ultra-High Energy Physics

About 30 people (~10 students)

1st experience using SWAN+CERNbox for LUHEP; plans to extend its usage to the whole SPbSU are discussed

This work is supported by the Russian Science Foundation, GRANT 17-72-20045.

CERN SWAN (Service for Web based ANalysis)

This is my understanding and not the official definition!

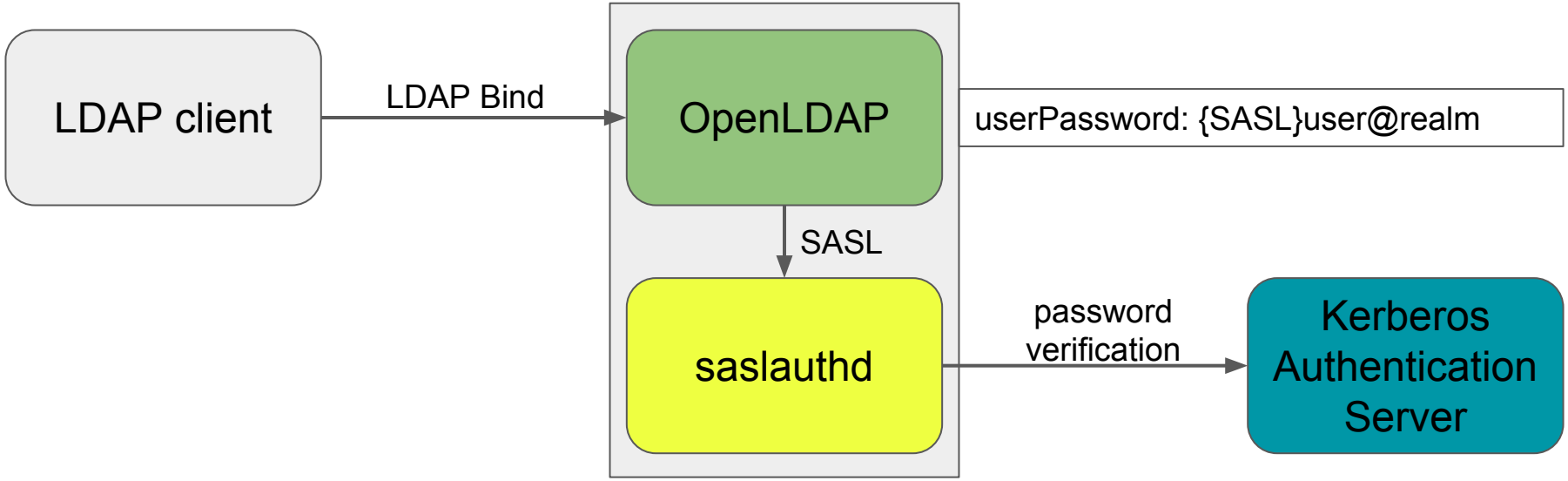
- CERN EOS storage system to store user files
- CERNBox cloud service to access them
- JupyterHub + a special Docker image for single-user Jupyter servers

EOS authentication and user mapping

- User may use Kerberos to authenticate and mount EOS directory
- EOS Manager maps username@REALM to UID
- OpenLDAP stores UIDs
- OpenLDAP stores user CPU and memory limits for JupyterHub
- OpenLDAP used by CERNBox and JupyterHub for authentication

Password duplication in OpenLDAP and Kerberos?

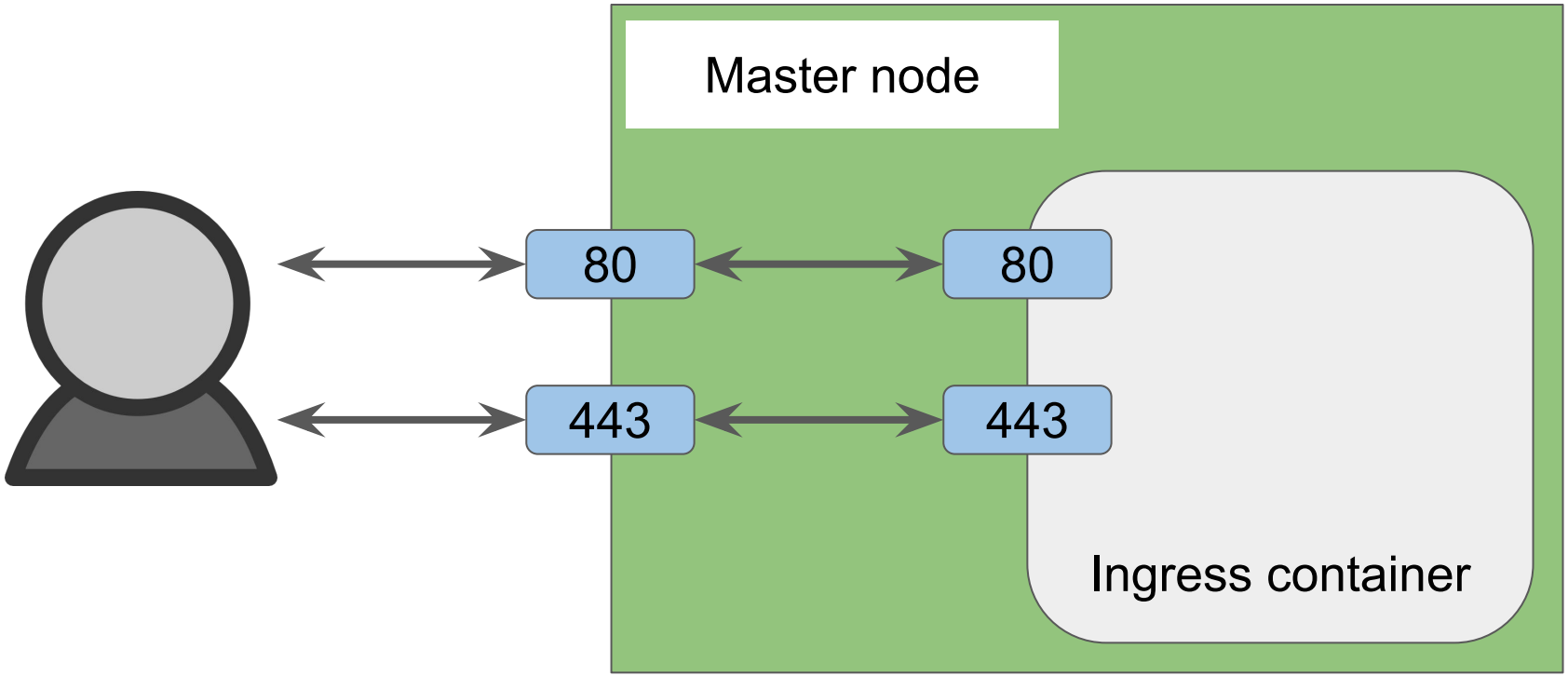
OpenLDAP SASL Pass-Through authentication



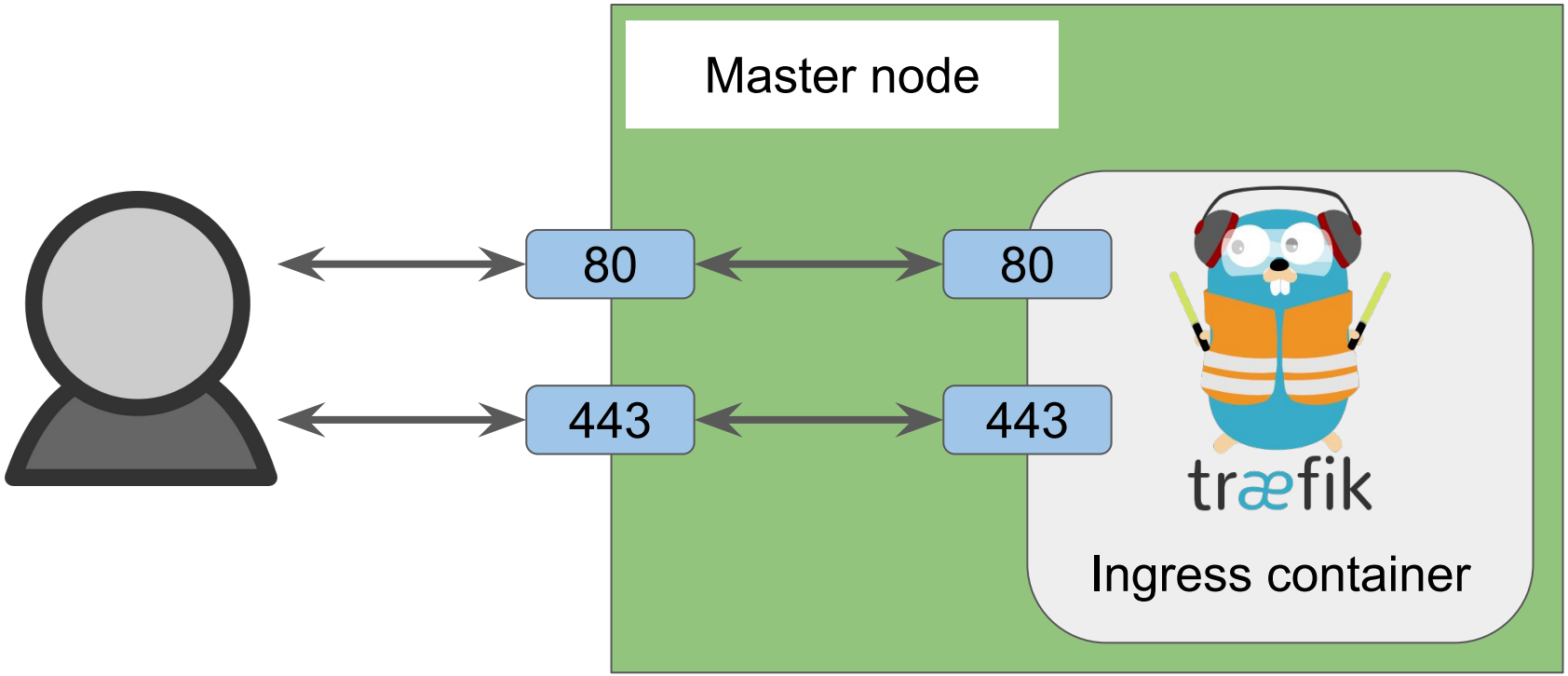
Our Kubernetes cluster

- One node with public IP — master node
- A few nodes with private IPs
- Cluster configured with **`kubeadm`**:
  - Easy to use
  - Secure: enforces **RBAC**, secures communication between Kubernetes components
- **Canal** pod network

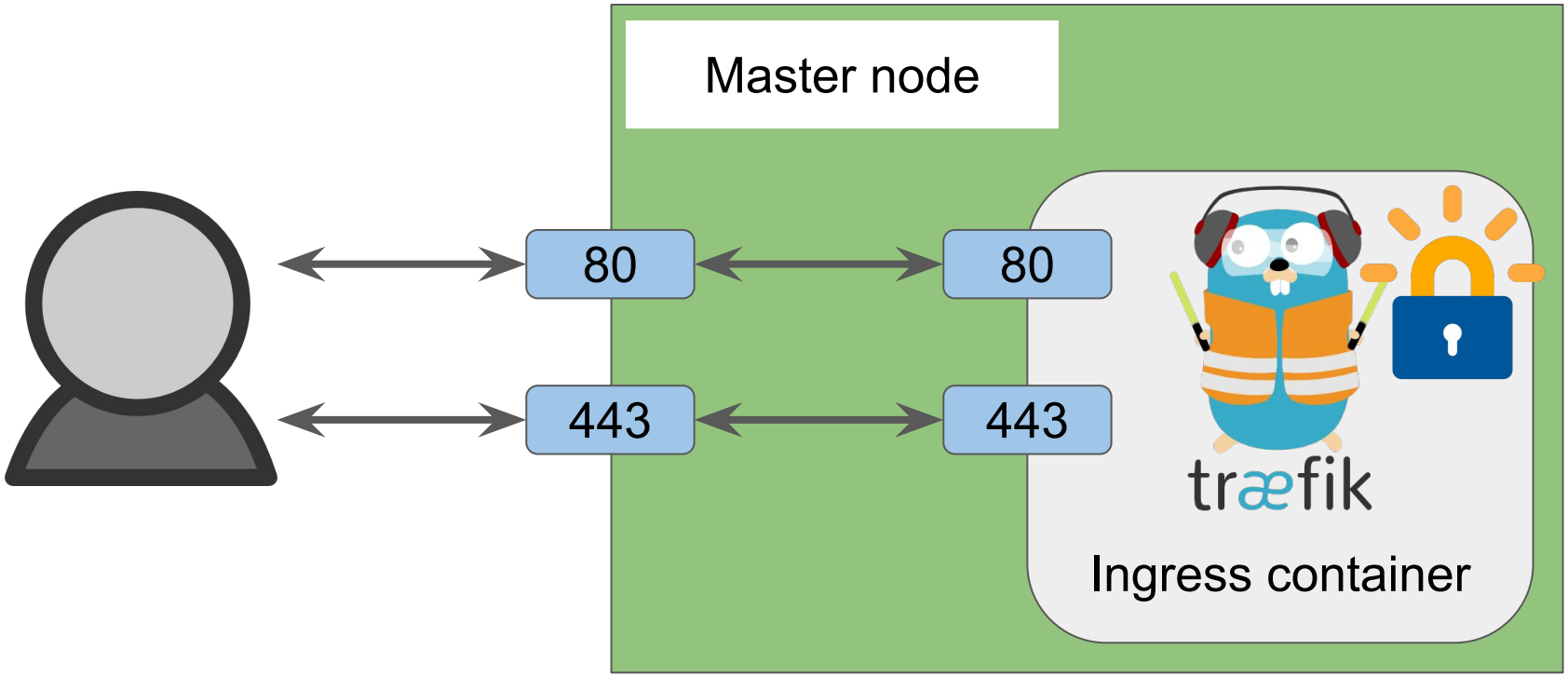
Ingress controller: hostPort



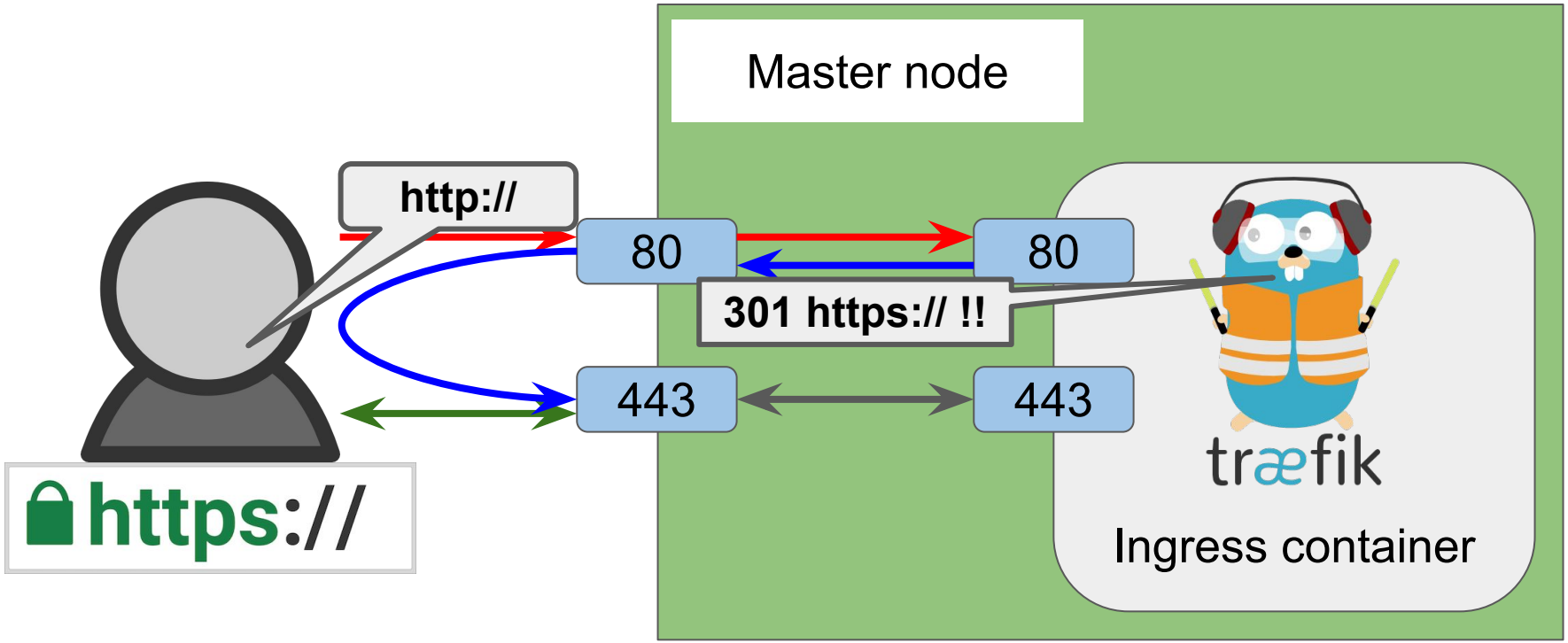
Ingress controller: træfik



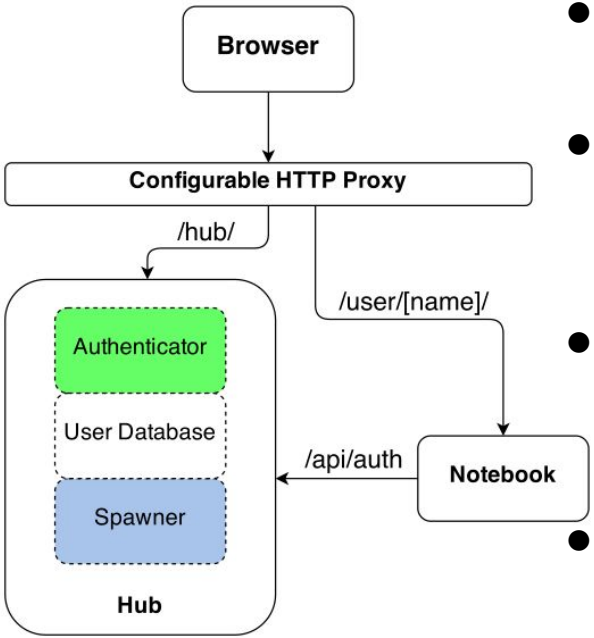
Ingress controller: Let’s Encrypt support



Ingress controller: force https

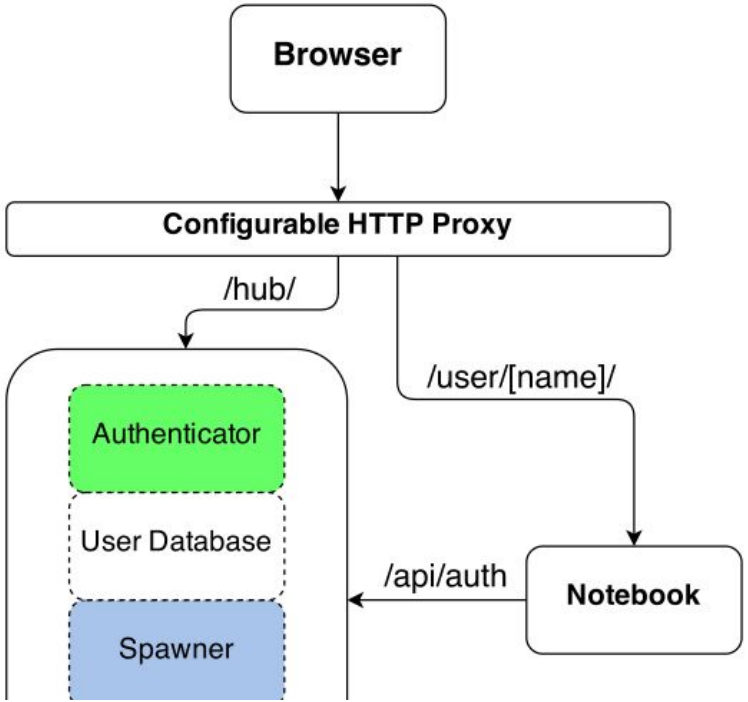


JupyterHub setup



- LUHEPLDAPAuthenticator = LDAPAuthenticator + loading CPU/MEM limits from OpenLDAP
- LUHEPKubeSpawner = KubeSpawner + dynamic **`/spawn`** page generation to let user choose CPU/MEM limits
- **`cernphsft/systemuser`** Docker image with Jupyter Notebook: ROOT C++ & Python & R kernels from **`cvmfs`**; user's home from **EOS**
- Proxy — later

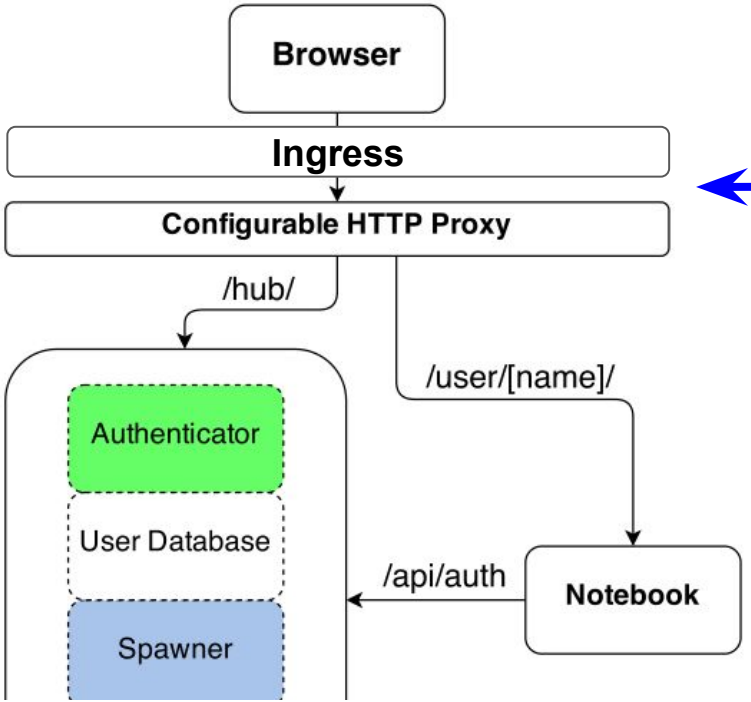
JupyterHub in the wild



Configurable HTTP Proxy = node-http-proxy + REST API to manage routes

This is JupyterHub in «default» mode

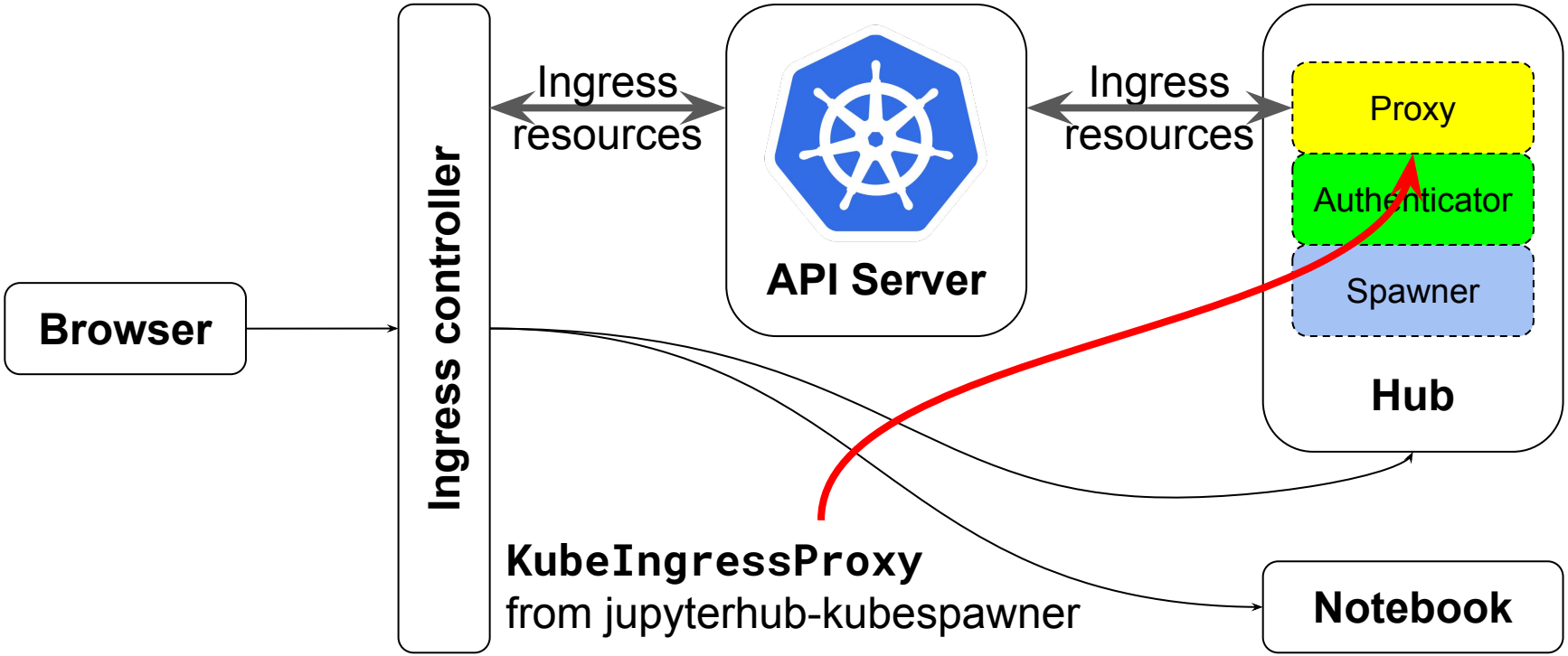
JupyterHub in k8s



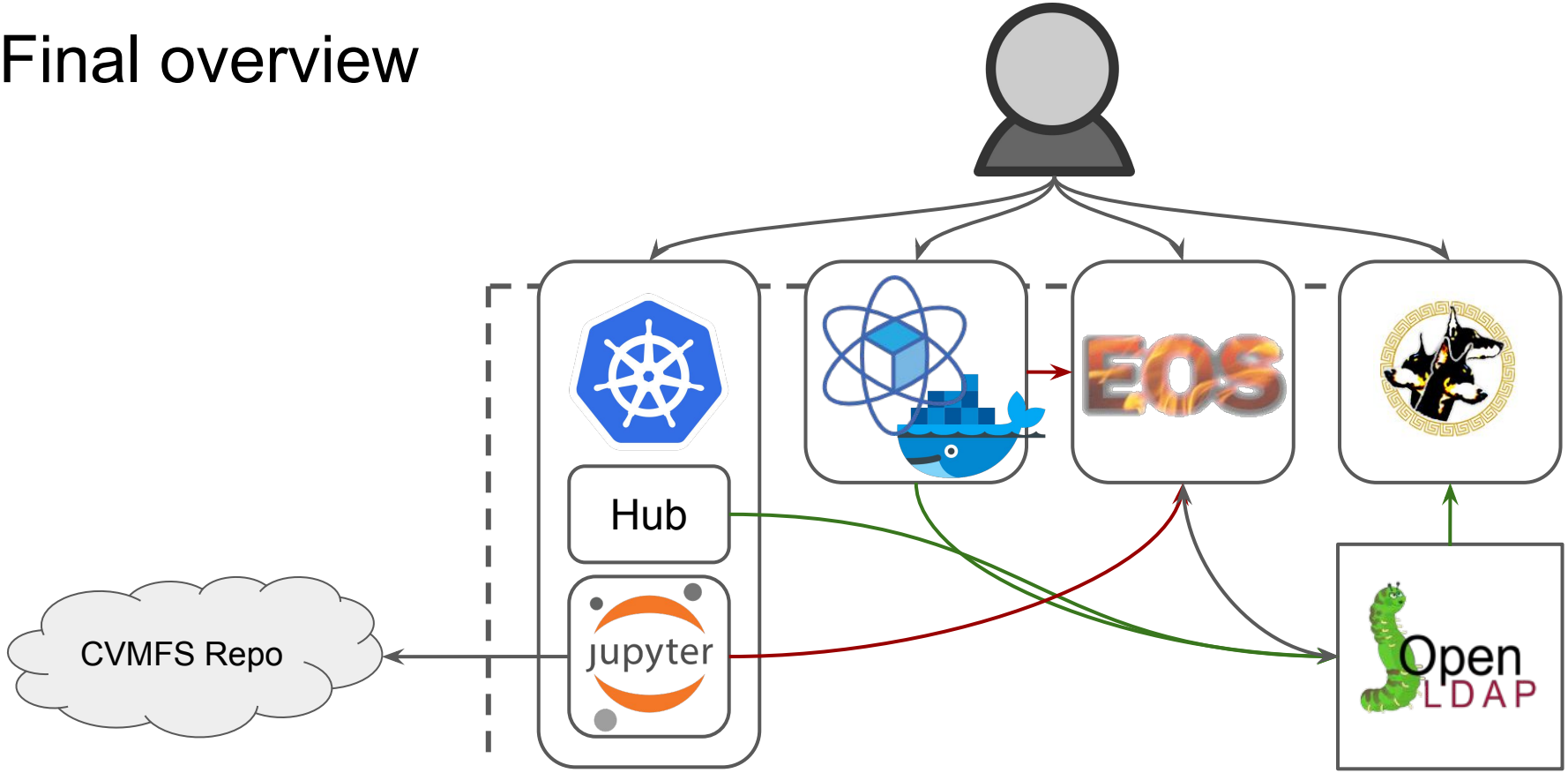
If we put JupyterHub in its «default» mode into k8s, it would look like here.

Too many proxies, isn't it?

JupyterHub 0.8 — customizable Proxy class



Final overview



Thank you!