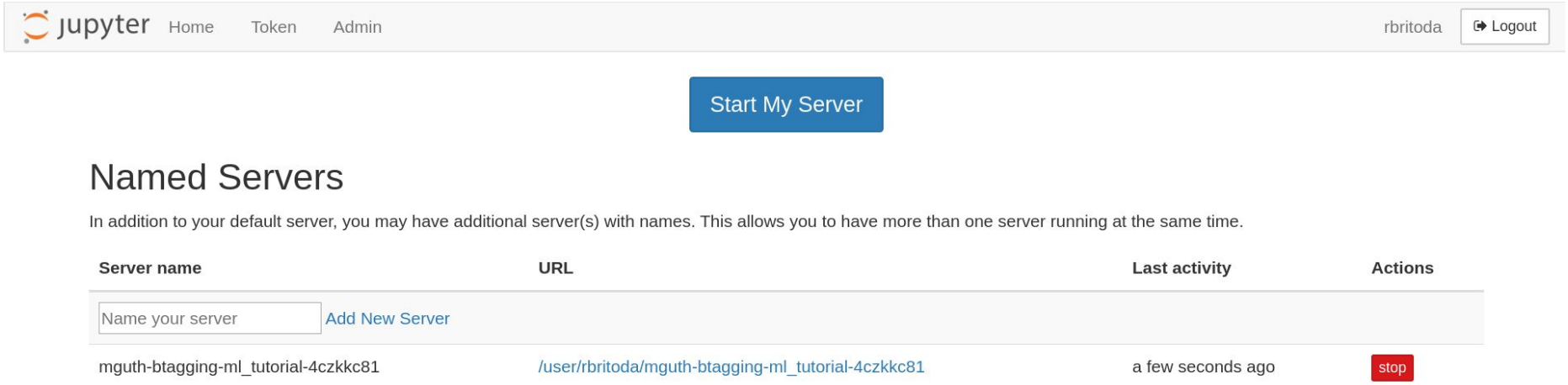
Web Application Access to EOS with OAuth2

Ricardo Rocha
CERN IT-CM-RPS

Use Case: A Notebook Service



Deployed on a Kubernetes cluster

Access to EOS via fuse, mounts managed by a container on each node

https://gitlab.cern.ch/cloud/eosd https://gitlab.cern.ch/helm/charts/cern/tree/master/eosxd

Problem

Kerberos is not a good fit for web based applications

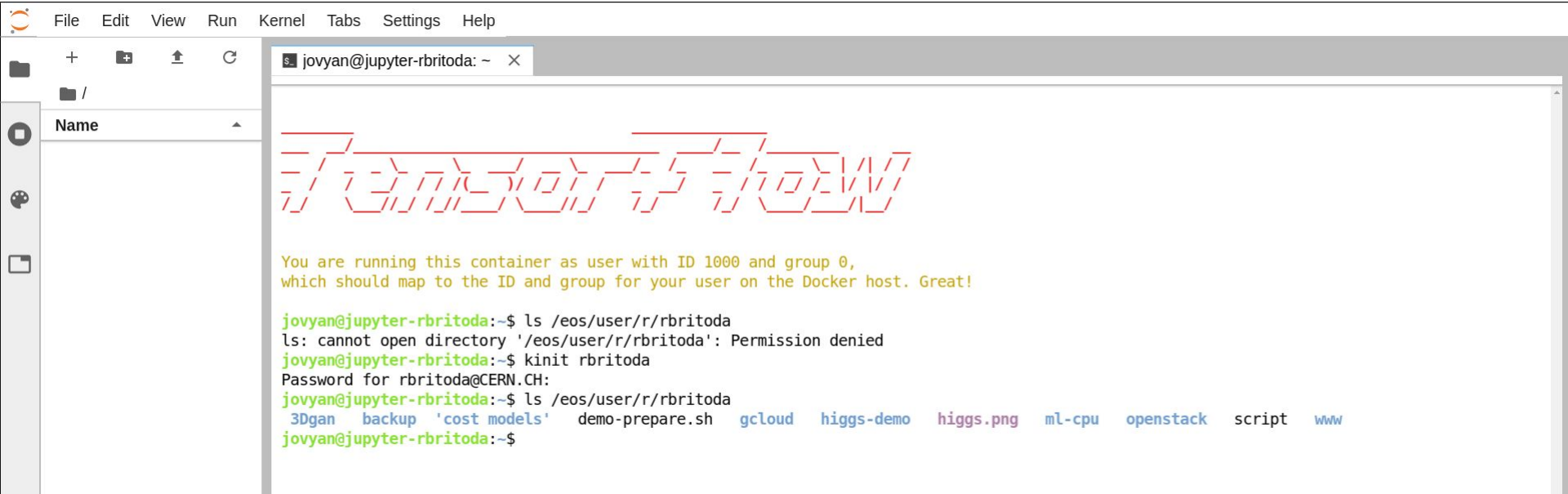
- Requires additional (and awkward) step after login to the main application
- Adds requirement for krb5 clients and setup on the client application
- Blocker if access to the *home directory* requires a valid token

Same is true for X509 certificates, particularly proxy certificates

Problem

Kerberos is not a good fit for web based applications

Requires additional (and awkward) step after login to the main application



File Edit View Run Kernel Tabs Settings Help

/

Name	Last Modified
img	3 months ago
warmStarts	3 months ago
DL1r_Variables.json	3 months ago
DL1r-extbeff-22M-ext-Zprime.npy	3 months ago
DL1r-extbeff-22M.npy	3 months ago
Dockerfile	3 months ago
params_MC16D-ext_2018-PFlow_70-8M…	3 months ago
plottingFunctions.py	3 months ago
README.md	3 months ago
Sample-preparation.ipynb	3 months ago
Sample-preparation.py	3 months ago
train_dips.ipynb	3 months ago
train_dips.py	3 months ago
train_DL1.ipynb	3 months ago
train_DL1.py	3 months ago
whatever.conf	3 months ago

train_DL1.ipynb

Markdown

Python 3

In the following the DL1 network is defined

[11]:

```
# Input layer
inputs = Input(shape=(X_train.shape[1],))
# number of nodes in the different hidden layers
l_units = [72, 57, 60, 48, 36, 24, 12, 6]
x = inputs
# loop to initialise the hidden layers
for unit in l_units:
    x = Dense(units=unit, activation="linear", kernel_initializer='glorot_uniform')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
#     x = Dropout(0.1)
# output layer, using softmax which will return a probability for each jet to be either
predictions = Dense(units=3, activation='softmax',
                    kernel_initializer='glorot_uniform')(x)

model = Model(inputs=inputs, outputs=predictions)
model.summary()
```

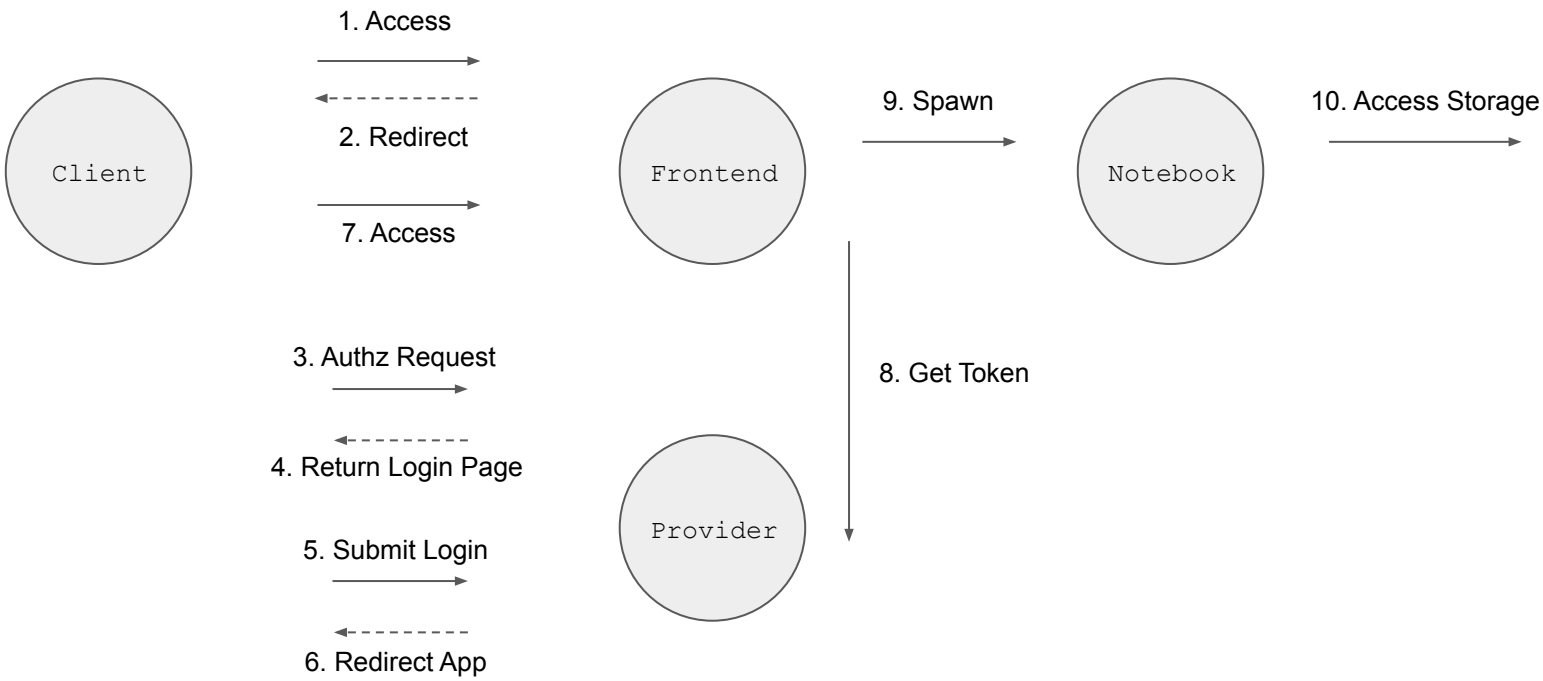
jovyan@jupyter-rbritoda-2dr

```
jovyan@jupyter-rbritoda-2dmguth-2dbtagging-2dml-5ftutorial-2d4czkkc81:~$ ls /eos/user/r/rbritoda
ls: cannot open directory '/eos/user/r/rbritoda': Permission denied
jovyan@jupyter-rbritoda-2dmguth-2dbtagging-2dml-5ftutorial-2d4czkkc81:~$ kinit rbritoda
Password for rbritoda@CERN.CH:
jovyan@jupyter-rbritoda-2dmguth-2dbtagging-2dml-5ftutorial-2d4czkkc81:~$ ls /eos/user/r/rbritoda
 3Dgan    'cost models'      gcloud       higgs.png   openstack   www
 backup    demo-prepare.sh   higgs-demo   ml-cpu      script
```

1 1

jovyan@jupyter-rbritoda-2dmguth-2dbtagging-2dml-5ftutorial-2d4czkkc81: ~

OIDC / OAuth2 to the rescue



OIDC / OAuth2 to the rescue

What's in a token?

```
echo $XrdSecsssENDORSEMENT | base64 -d

{..., "exp":1580858527, ,"iss":"https://auth.cern.ch/auth/realms/cern", "typ":"Bearer",
 "aud":"hub-staging", "azp":"hub-staging", "auth_time":1580851584,
 "allowed-origins":["https://hub-staging.cern.ch"],
 "scope":"oidc-cern-profile oidc-email oidc-cern-login-info oidc-client-id offline-access",
 "sub":"rbritoda", "cern_upn":"rbritoda",
 "resource_access":{"hub-staging":{"roles":["hep"]}},
 "cern_person_id":617758, "name":"Ricardo Brito Da Rocha", "preferred_username":"rbritoda",
 "given_name":"Ricardo", "cern_roles":["hep"], "cern_preferred_language":"EN",
 "family_name":"Brito Da Rocha", "email":"ricardo.rocha@cern.ch", ...}
```

OIDC / OAuth2 to the rescue

Server side support added by the EOS team (many thanks!)

EOS fuse client already had support to pass a token, no change required

Minimal setup and configuration on the client

```
XrdSecsssENDORSEMENT=”oauth2:<token>:auth.cern.ch/auth/realms/cern/protocol/openid-connect/userinfo”

"auth" : { "shared-mount" : 1, "krb5" : 0, "sss" : 1,
           "ssskeytab" : “/etc/eos/fuse.sss.keytab" }
```

Demo

```
rbritoda@lxplus733 ~/.../helm/releases/hub$$ kubectl -n eosxd get all
NAME              READY   STATUS    RESTARTS   AGE
pod/eosxd-46hg5   1/1     Running   0          26m
pod/eosxd-kzzz5   1/1     Running   0          26m
pod/eosxd-w5mnt   1/1     Running   0          26m

NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/eosxd   3         3         3       3            3           <none>          26m
```

```
rbritoda@lxplus733 ~/.../helm/releases/hub$$ kubectl -n eosxd get configmap -o yaml
apiVersion: v1
items:
- apiVersion: v1
  data:
    fuse.ams.conf: |
      {"name":"ams","hostport":"eosams.cern.ch","localmountdir":"/eos/ams/", "remotemountdir":"/eos/ams/"}
    fuse.atlas.conf: |
      {"name":"atlas","hostport":"eosatlas.cern.ch","localmountdir":"/eos/atlas/", "remotemountdir":"/eos/atlas/"}
    fuse.cms.conf: |
      {"name":"cms","hostport":"eoscms.cern.ch","localmountdir":"/eos/cms/", "remotemountdir":"/eos/cms/"}
    fuse.conf: |
      {
        "auth": {
          "environ-deadlock-timeout": "500",
          "forknoexec-heuristic": "1",
          "gsi-first": "0",
          "krb5": "0",
          "shared-mount": "1",
          "sss": "1",
          "ssskeytab": "/etc/eos/fuse.sss.keytab",
        },
```

Is this it? Almost ...

Token Renewal

Notebooks (and batch jobs) are often long lived

CERN gives access tokens with a lifetime of 20min (even started with 60sec)

Refresh Tokens can be used to renew the access token

> But these are shell like environments (notebook, terminal)
>
> And access tokens are passed via environment variables

WIP: rely on a file instead for these cases

https://gitlab.cern.ch/cloud/oauth2-refresh

Token Renewal Sidecar

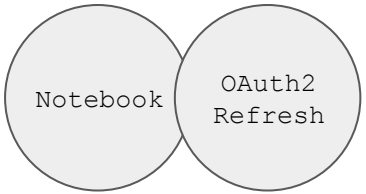
Implementation for Kubernetes as a *Sidecar*

Containers in the same Pod share the filesystem

Sidecar handles the access token renewal using the refresh token

    And updates the file expected by the EOS fuse client

Fully transparent to the main application



Questions?