GXFS/XFSC Deployment

- XFSC is the new name for the federated catalogue project, hosted by eclipse
- There is no documentation for helm/.yaml deployments (only for docker-compose).
- It is unclear what **other** FC components are needed to register services (Felix Schmidt from Frauenhofer has mentioned that at least a signer service is needed..)

Deployment with Helm on a Kind cluster

- Set up a ingress-compatible Kind cluster

```
kind delete clusters --all

# create a kind cluster N.B. with ingress-ready=true
cat <<EOF | kind create cluster --config=-
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
EOF
```

- Deploy ingress-nginx:

```
# This creates a ingress-nginx namespace and installs the ingress controller
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
nginx/main/deploy/static/provider/kind/deploy.yaml
```

- Checkout repository

```
git clone git@github.com:stackabletech/fc-service.git
cd fc-service/deployment/helm/fc-service
```

The Stackable repository above is a mirror of the Eclipse repository

- Switch to the `stackable` branch

```
git switch stackable
```

- Deploy charts

```
cd deployment/helm/fc-service && \
helm dependency build && \
kubectl create namespace federated-catalogue && \
helm install --namespace federated-catalogue --generate-name --set
"service.type=ClusterIP" .
```

- Add a user as described here: docker · main · Eclipse Projects / xfsc / Federated...
- Ensure /etc/hosts contains the following:

```
127.0.0.1 key-server.stackable.com
127.0.0.1 fc-server.stackable.com
127.0.0.1 demo.stackable.com
```

- the kind/nginx setup above will expose the nginx ingress controller on `127.0.0.1`

Demo App

Due to re-direct issues the demo app does not work as documented (at least as of July 2023). See these related issues: Invalid redirect (#5) · Issues · Gaia-X /... and/or this one: Login redirect error (#159) · Issues · Gaia-X /.... However, the federated catalogue service can be used/tested either with `curl` calls or with Postman.

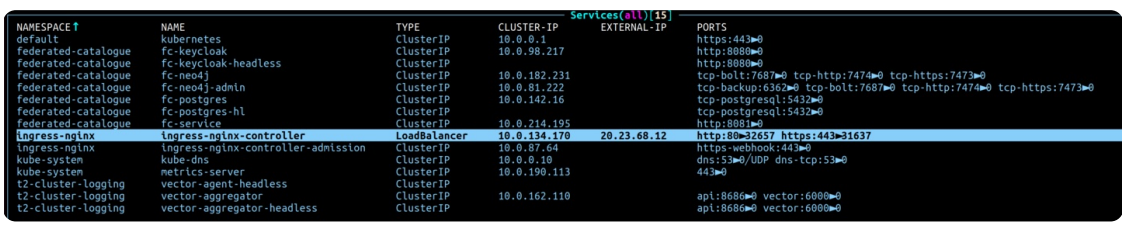
Deployment with Helm on a self-service cluster (tested with Azure & IONOS)

Follow the steps above with the exception of the following:

- Deploy `nginx` using a different deploy script:

```
kubectl create namespace ingress-nginx && \
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
nginx/controller-v1.8.1/deploy/static/provider/cloud/deploy.yaml
```

- Make a note of the nginx ClusterIP and ExternalIP e.g. (from below)
  - ClusterIP: `10.0.134.170`
  - External IP: `20.23.68.12`



- in `values.yaml` change the `hostAliases`:

```
hostAliases:
  - ip: "10.0.134.170"  # ClusterIP of Ingress Controller
    hostnames:
      - key-server.stackable.com
```

- and in the deployment `yaml` for the the demo app:

```
      hostAliases:
      - hostnames:
        - "key-server.stackable.com"
        ip: "10.0.134.170" # ClusterIP of Ingress Controller
```

- the entries in /etc/hosts must also be changed to reflect the ExternalIP:

```
20.23.68.12 key-server.stackable.com
20.23.68.12 fc-server.stackable.com
20.23.68.12 demo.stackable.com
```

Using the XFSC API

The federated catalogue service should be set up as described above. Note: for the token-exchange to be successful the `hostAliases` entry in `values.yaml` should be changed to the Ingress Controller `ClusterIP` for all cases, including a local installation using `kind`.

The examples below will be using the REST API directly.

See: GitHub - Digital-Ecosystems/gx-catalogue-ionos

Retrieve an access token

Add a user in the keycloak frontend and then access the access token that references this user (replace `<user>` and `<password>` in the example below, with the actual user credentials)

```
ACCESS_TOKEN=$(
    curl -v -k \
    -d "client_id=federated-catalogue" \
    -d "client_secret=<client secret>" \
    -d "username=<user>" \
    -d "password=<password>" \
    -d "grant_type=password" \
    "http://key-server.stackable.com/realms/gaia-x/protocol/openid-
connect/token" | jq '.access_token' | tr -d '"'
)
echo $ACCESS_TOKEN
```

Retrieve users

The access token can now be used in API calls e.g. to query existing users:

```
curl -v -k -H "Authorization: Bearer $ACCESS_TOKEN" http://fc-
server.stackable.com/users | jq
```

The output should be something like the following, displaying the user details:

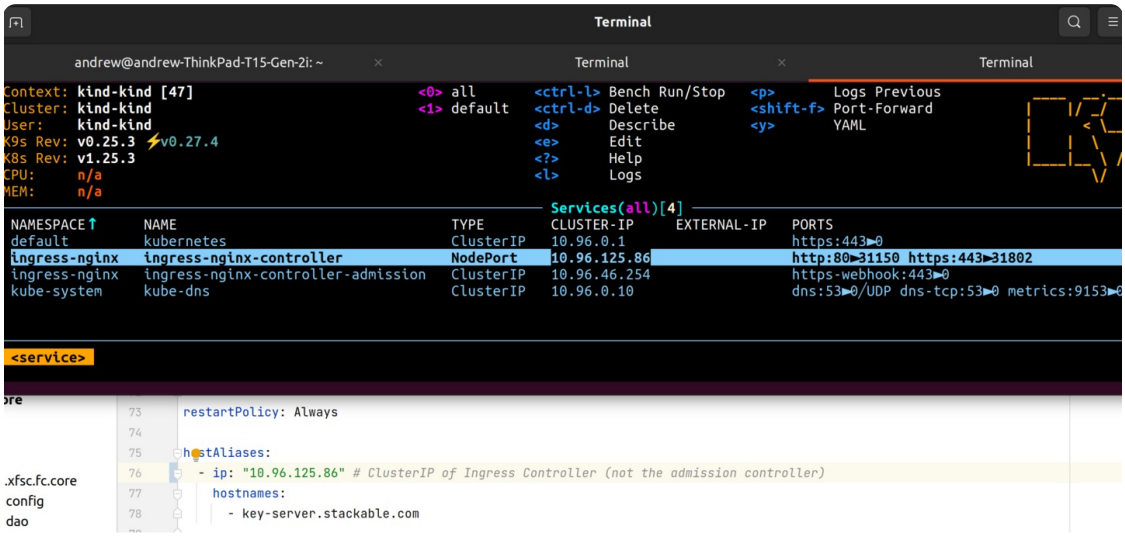
```
{
  "totalCount": 1,
  "items": [
    {
      "participantId": null,
      "firstName": "<firstname>",
      "lastName": "<lastname>",
      "email": "<email>",
      "roleIds": [ # these are the roles assigned in the keycloak UI
        "Ro-SD-A",
        "Ro-PA-A",
        "uma_protection",
        "Ro-MU-CA",
        "Ro-MU-A"
      ],
      "id": "1c501835-3c6d-479a-ab77-5efd4a54575b",
      "username": "<full name>"
    }
  ]
}
```

Verifying a Self-Description (this section should be self-enclosed)

N.B. make sure the `stackable` branch is the active one!

```
# /etc/hosts should look like this with kind:
127.0.0.1 key-server.stackable.com
127.0.0.1 fc-server.stackable.com
```

```
kind delete clusters --all

cat <<EOF | kind create cluster --config=-
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    #listenAddress: "127.0.0.1"
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    #listenAddress: "127.0.0.1"
    protocol: TCP
EOF

kubectl create namespace ingress-nginx && kubectl apply -f \
  https://raw.githubusercontent.com/kubernetes/ingress-
nginx/main/deploy/static/provider/kind/deploy.yaml

# navigate to deployment folder:
# cd /{path-to-git-repo}/fc-service/deployment/helm/fc-service

# Change the cluster IP of the ingress controller in values.yaml (see above),
# Edit /etc/hosts as needed (see above)
```




```
# Then, create FC (the fc-service requires 2-3 restarts until fc-keycloak is
ready)...

kubectl create namespace federated-catalogue && helm install \
```

```
  --namespace federated-catalogue \
  --generate-name \
  --set "service.type=ClusterIP" .

# Open http://key-server.stackable.com/admin console,
# go to *gaia-x realm* and create new test user:
# login with: admin/admin
# user: stackable
# password: Adminadmin1! (turn off "Temporary")
# role mappings: Ro-MU-CA, Ro-MU-A (although I think one contains the other)

# GET schemas (should find 2 ontologies and one default schema)
# For this and the following examples, we will use
#   client_secret=jakobsgold2023
#   username=stackable
#   password=Adminadmin1!
# N.B. client_secret has to match the secret defined in
# fc-service/deployment/helm/fc-service/templates/keycloak-client-secret.yaml

ACCESS_TOKEN=$(
    curl \
    -d "client_id=federated-catalogue" \
    -d "client_secret=jakobsgold2023" \
    -d "username=stackable" \
    -d "password=Adminadmin1!" \
    -d "grant_type=password" \
    "http://key-server.stackable.com/realms/gaia-x/protocol/openid-
connect/token" | jq '.access_token' | tr -d '"'
)
curl -v -H "Authorization: Bearer $ACCESS_TOKEN" -H 'Content-Type:
application/json' \
  -XGET http://fc-server.stackable.com/schemas | jq

# returning e.g.
#{
#  "ontologies": [
#    "https://w3id.org/gaia-x/gax-trust-framework#",
#    "https://w3id.org/gaia-x/core#"
#  ],
#  "shapes": [
#    "1729e65cd0b800aee432aaae3dcc19198f3f7d7c3b7875c346530007e4a79a15"
#  ],
#  "vocabularies": null
#}

# Use this SD as an example:
# https://gitlab.eclipse.org/eclipse/xfsc/cat/fc-service/-
/blob/addTrustedCloudExample/examples/Trusted_Cloud_Examples/IonosCloud_ServiceOff
ring-instance_valid.json

# Use this schema as an example:
# https://gitlab.eclipse.org/eclipse/xfsc/cat/fc-service/-
/blob/addTrustedCloudExample/examples/Trusted_Cloud_Examples/trusted-
cloud_generated_ttl.rdf
```

```
# Go to http://fc-server.stackable.com/verification page and ensure the example
SD is *not* valid.
```

https://gitlab.eclipse.org/eclipse/xfsc/cat/fc-service/-/blob/addTrustedCloudExample/examples/Trusted_Cloud_Examples/trusted-cloud_generated_ttl.rdf

Verify Self-Description

Please upload a file OR copy / paste in textarea:

Choose File IonosCloud_...ce_valid.json


```
{
  "@id": "http://example.edu/verifiablePresentation/self-description19",
  "proof": {
    "created": "2023-03-01T15:40:19Z",
    "jws":
"eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJQUzI1NiJ9..aFjeX1Ppmvum5_wEFNh_Gp3yxBeLML
mss97fChMHbJ1WLzak88AU_oqK3pCbz_e6C_kj7vc5yz7M3_RjunSJuP1GvM4b7cvYXaIjksJaAcxRIcTouTM7c
4ZMvjr7nFo1wvQpU6xWSgsfC2WZelYIsX1lXs7zP3BOFDVBKPHQa1g7sZkHKTPGuQ0QyA5USlpmCUv_rfar4
8DJmQWtUkxZmGCLpuPuiULSBpkNxWiA4931UJY_sxuNDmurQyPAoNCD7d2auytoeV3N52GUG_IrPetwnnK
DYO0KLqjaFZisgNuk0aL5NbF3kYCs05RfW4N2pWe_QmgCWdrswapsG7MF9PXusaFe6brwx2SqdiBwl9GZIko
ykgu3H-om1-Nw4iTURr5LASxViXPTxMNKdDnB0i7m9fjqtC3Nxhe04rW4b1wts15puOD_4sa-
T83EkQ8GTi5wee1wRAHq10d8WDzxyCNVaoZv4STqKbq3e0aU_2qlWnEwxYyoea-
8hz2YQfXbCe_74KZBZU7vHyl_srqDCi6gpRdzI-mZq-7bIVb9uXP26ngQln-r-
eExKmjkehz2zIlW6sLW9TtU0LVuW2MYzfbI5L1mhtic-
LAEckcFn4crjA02nQByeFJZlJqM_DFIXI8Xj6RNti8X1lhqN36f9XrvL7yumSCtk16WHJuj9uE",
    "proofPurpose": "assertionMethod",
    "type": "JsonWebSignature2020",
    "verificationMethod": "did:web:compliance.lab.gaia-x.eu"
  },
  "type": ["VerifiablePresentation"],
  "@context": ["https://www.w3.org/2018/credentials/v1"],
```

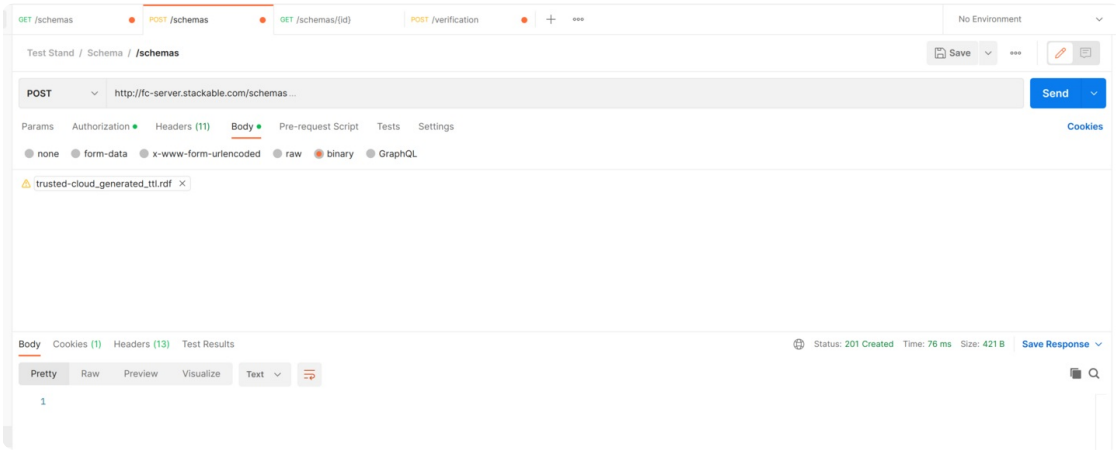

Verify Semantics Verify Schema Verify Signature

Verify

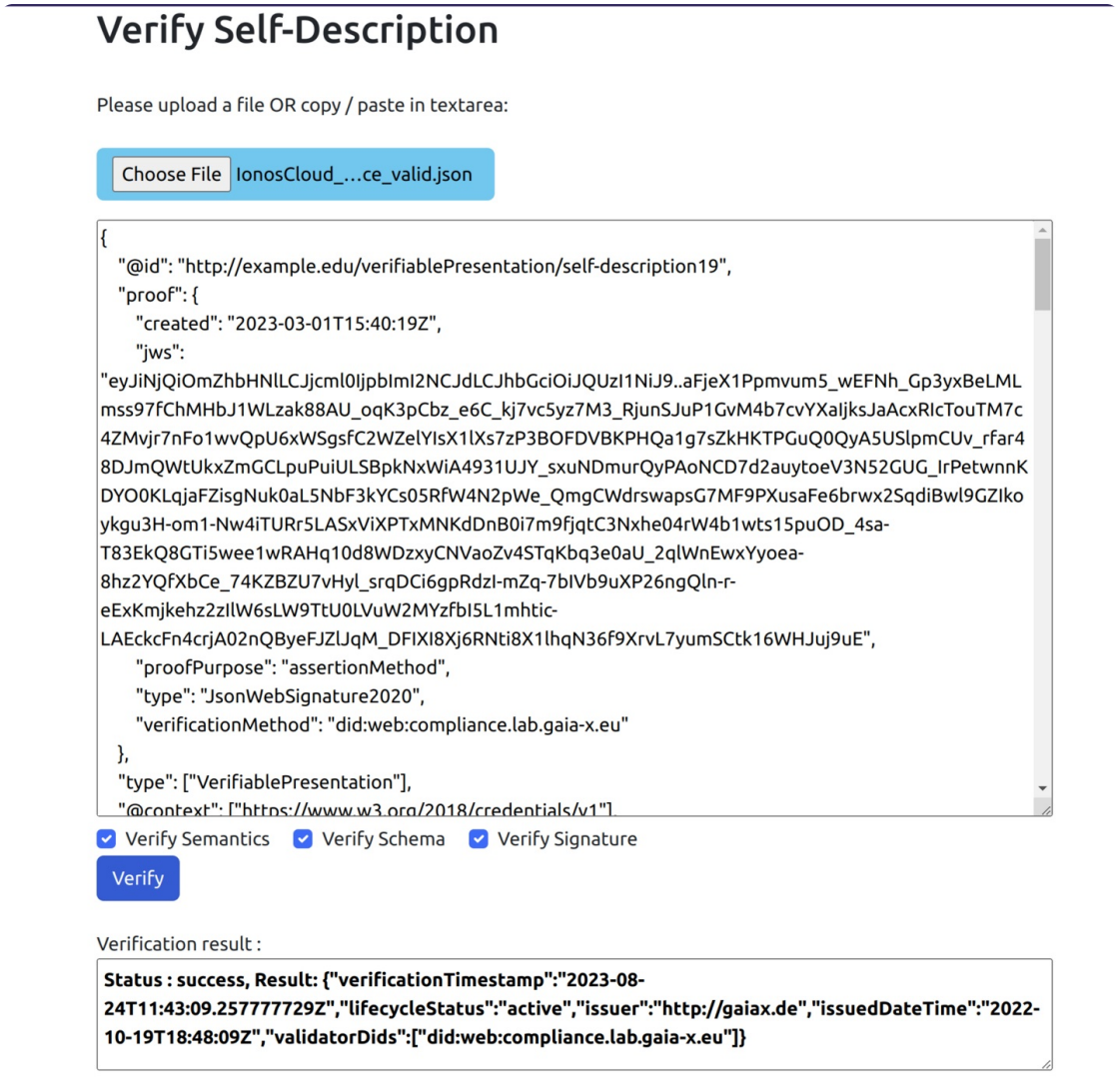
Verification result :

Status : error, Error : {"code":"verification_error","message":"Semantic Error: no proper CredentialSubject found"}

```
# Use this postman collection:
# https://gitlab.com/gaia-x/data-infrastructure-federation-services/gxfs-workshop/-/blob/main/Catalogue/Basics/Test%20Stand.postman_collection.json
# Get token:

ACCESS_TOKEN=$(
    curl \
    -d "client_id=federated-catalogue" \
    -d "client_secret=jakobsgold2023" \
    -d "username=stackable" \
    -d "password=Adminadmin1!" \
    -d "grant_type=password" \
    "http://key-server.stackable.com/realms/gaia-x/protocol/openid-connect/token" | jq '.access_token' | tr -d '"'
)
echo $ACCESS_TOKEN

# POST the example schema and then re-verify:
```




```
# The schema can also be posted with curl (curl command can be exported from Postman: see code
# snippet menu on in the right-hand margin)

# N.B. the Content-Type must be application/rdf+xml and use --data-binary!

ACCESS_TOKEN=$(
    curl \
    -d "client_id=federated-catalogue" \
    -d "client_secret=jakobsgold2023" \
    -d "username=stackable" \
    -d "password=Adminadmin1!" \
    -d "grant_type=password" \
    "http://key-server.stackable.com/realms/gaia-x/protocol/openid-connect/token" | jq '.access_token' | tr -d '"'
)
curl -v -H "Authorization: Bearer $ACCESS_TOKEN" -H 'accept: application/json' \
  -H 'Content-Type: application/rdf+xml' \
  -XPOST http://fc-server.stackable.com/schemas \
  --data-binary @/path-to/trusted-cloud_generated_ttl.rdf
```




```
# Or, using curl:
ACCESS_TOKEN=$(
    curl \
    -d "client_id=federated-catalogue" \
    -d "client_secret=jakobsgold2023" \
    -d "username=stackable" \
    -d "password=Adminadmin1!" \
    -d "grant_type=password" \
    "http://key-server.stackable.com/realms/gaia-x/protocol/openid-
connect/token" | jq '.access_token' | tr -d '"'
)
curl -v -H "Authorization: Bearer $ACCESS_TOKEN" -H \
  'Content-Type: application/json' \
  -XPOST http://fc-server.stackable.com/verification \
  -d @/{path-to}/IonosCloud_ServiceOffering-instance_valid.json

 # e.g.
 # {"verificationTimestamp":"2023-08-
24T11:45:48.628741348Z","lifecycleStatus":"active","issuer":"http://gaiax.de","iss
edDateTime":"2022-10-19T18:48:09Z","validatorDids":
["did:web:compliance.lab.gaia-x.eu"]}
```

Creating a Stackable Self-Description

Use this as an example:


```
{
    "@context": [
      "https://www.w3.org/2018/credentials/v1"
    ],
    "@id": "https://delta-dao.com/.well-known/participantStackable.json",
    "type": [
      "VerifiablePresentation"
    ],
    "verifiableCredential": [
      {
        "@context": [
          "https://www.w3.org/2018/credentials/v1"
        ],
        "@id": "https://delta-dao.com/.well-known/participantStackable.json",
        "@type": [
          "VerifiableCredential"
        ],
        "issuer": "http://gaiax.de",
        "issuanceDate": "2022-10-19T18:48:09Z",
        "credentialSubject": {
          "@context": {
            "gax-core": "https://w3id.org/gaia-x/core#",
            "gax-trust-framework": "https://w3id.org/gaia-x/gax-trust-
framework#",
            "xsd": "http://www.w3.org/2001/XMLSchema#",
            "vcard": "http://www.w3.org/2006/vcard/ns#"
          },
          "@id": "gax-core:Participant1",
          "@type": "gax-trust-framework:LegalPerson",
          "gax-trust-framework:registrationNumber": "DE334447979",
          "gax-trust-framework:legalAddress": {
            "@type": "vcard:Address",
            "vcard:country-name": "Germany",
            "vcard:locality": "Wedel",
            "vcard:postal-code": "22880",
            "vcard:street-address": "Thomas-Mann-Straße 8"
          },
          "gax-trust-framework:headquarterAddress": {
            "@type": "vcard:Address",
            "vcard:country-name": "Germany",
            "vcard:locality": "Wedel",
            "vcard:postal-code": "22880",
            "vcard:street-address": "Thomas-Mann-Straße 8"
          },
          "gax-trust-framework:termsAndConditions": {
            "@type": "gax-trust-framework:TermsAndConditions",
            "gax-trust-framework:content": {
              "@type": "xsd:anyURI",
              "@value":
"https://docs.stackable.tech/home/stable/release_notes"
            },
```

```
            "gax-trust-framework:hash": "1234"
          },
          "gax-trust-framework:legalName": "Stackable GmbH"
        }
      }
    ]
  }
```

Compile and run this class from the Stackable mirror:

```
# N.B. create a key pair using the following command:
openssl req -x509 -newkey rsa:4096 -keyout prk.ss.pem -out cert.ss.pem -sha256 -days 365 -nodes

mvn clean package
java -jar ./target/gxfsTest-0.1.0-jar-with-dependencies.jar
```

N.B. you will have to change these paths used as constants in Main:

```
36
37  public class Main {
38      //openssl req -x509 -newkey rsa:4096 -keyout prk.ss.pem -out cert.ss.pem -sha256 -days 365 -nodes
39      private static final String PATH_TO_PRIVATE_KEY = "src/main/resources/prk.ss.pem";
40      private static final String PATH_TO_PUBLIC_KEY = "src/main/resources/cert.ss.pem";
41      private static String PATH_TO_SELF_DESCRIPTION = "src/main/resources/vc.json";
42      private static String PATH_TO_SIGNED_SELF_DESCRIPTION = "src/main/resources/sd.signed.json";
43
44      static String readFile (String path) throws IOException {
```

The resulting signed json file can be pasted into the box at the top of http://fc-server.stackable.com/verification and then verified:

- **Without** `"Verify Signature"` everything should be fine
- **With** `"Verify Signature"` there will be an error, like the following. It is not clear why. The signer project also checks the credential ("proof") before exiting, but as it uses different versions of some libraries this ***may*** be the cause.

```
Status : error, Error : {"code":"verification_error",
"message":"Signatures error;
com.danubetech.verifiablecredentials.VerifiablePresentation
does not match with proof"}
```