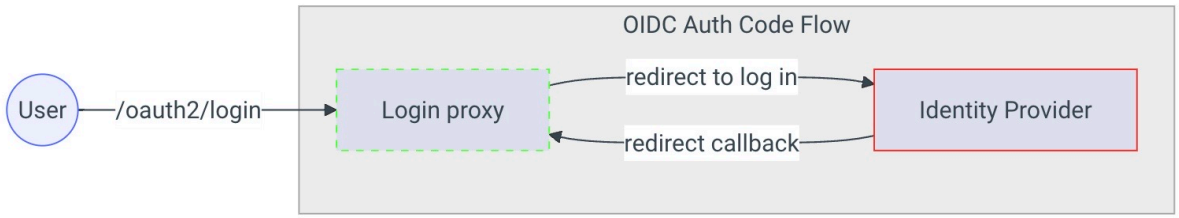
Installing Wonderwall in your own namespace.

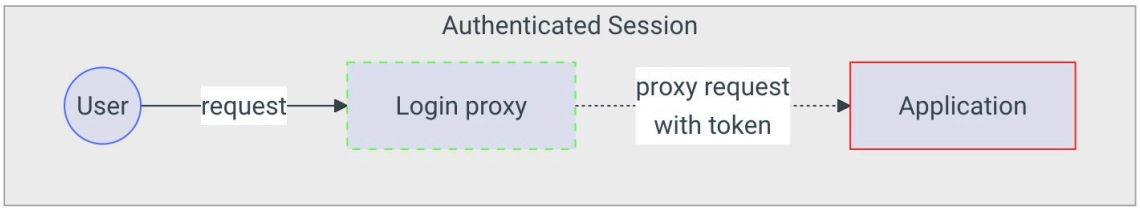
Background

The NAIS platform simplifies the installation of a login proxy as a sidecar to applications requiring user authentication in either ID-Porten or Entra ID (aka. Azure AD). This is done declaratively through entries in the NAIS manifest. With the required configuration in place, NAIS performs a pre-registration of the application with the ID provider (IDP). The client ID and JWK (among other things) are set as environment variables, allowing the application to use these to securely authenticate with the IDP in order to request tokens.

The proxy can be configured to pass unauthenticated requests straight through, only kicking in if special endpoints for login *(/oauth2/login)* or logout *(/oauth2/logout)* are accessed, or to ensure that **all** requests are authenticated. It is the latter configuration mode we are interested in here. Correctly configured, the login proxy intercepts all calls to the actual application, ensuring that the user is authenticated, forcing a login sequence using the OIDC Authorization Code Flow if not.



After this flow is complete, a cookie is set in the browser, and this cookie is used as a key to look up the token from the session in subsequent requests, passing it on in an Authorization header. The session information is typically held in Redis, which needs to be enabled for the login proxy.



Since there is no platform support (yet) in NAIS for HelseID, some more work is required:

Manually register the application with the HelseID self-service application

Note that access to this application is limited to pre-registered users in the Nav organization. It is probably best to keep it this way until we see the extent of delegated access required. An API for automation does indeed exist. The required scopes must also be registered at this point.

Download a file containing the Client JWK and the Client ID

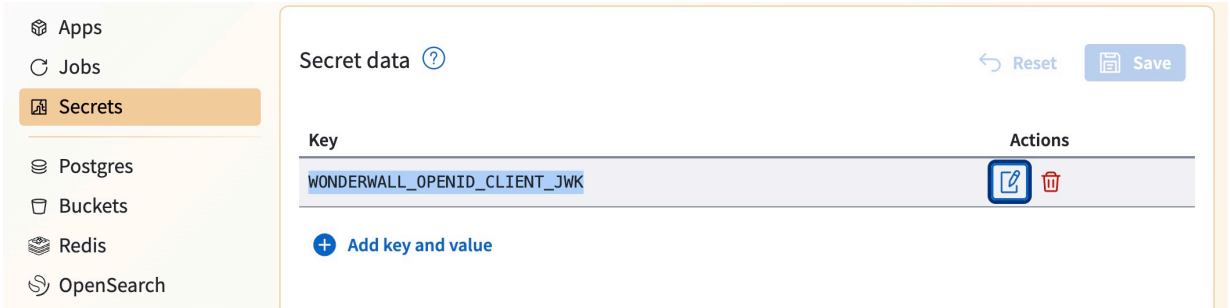
Same comment as above.

OPen/decrypt the file using the password sent by SMS.

Same comment as above

Register the JWK as a secret using the NAIS console for your team.

(Here we assume that whoever performed the client registration manually has transferred the file to you by secure means)



The name of the secret can be anything, but the name of the key must be exactly as above. Remember/write down the secret name and the client id. You may have to reformat the JWK to be a valid JSON. You may have to add the following key to the JWK, it is not a required field, but Wonderwall still requires in (may be fixed in later versions)

BAT

```
"alg": "RS256",
```

Create a NAIS manifest file for the Wonderwall instance

Note that this app is the one that is exposed to the outside world. The upstream app that does the real work should not have an ingress.

YAML

```yaml
apiVersion: "nais.io/v1alpha1"
kind: "Application"
metadata:
  name: <your wonderwall instance name>
  namespace: <your namespace>
  labels:
    team: <your team>
spec:
  redis:
    - instance: sessions
      access: readwrite
  envFrom:
    - secret: <your wonderwall secret name>
  accessPolicy:
    outbound:
      rules:
       - application: <your upstream app, ie the app that does the actual work>
      external:
        - host: helseid-sts.test.nhn.no <omit test for prod, obviously>
  image: ghcr.io/nais/wonderwall:latest
  liveness:
    path: /oauth2/ping
    initialDelay: 60
  readiness:
    path: /oauth2/ping
    initialDelay: 60
  prometheus:
    enabled: false
  replicas:
    min: <up to you>
    max: <up to you>
    cpuThresholdPercentage: <up to you>
  ingresses:
    - <your ingress>
  resources:
    limits:
      memory: <up to you>
    requests:
      cpu: <up to you>
      memory: <up to you>
  env:
    - name: WONDERWALL_OPENID_SCOPES
      name:  profile, offline_access,
helseid://scopes/identity/assurance_level,helseid://scopes/identity/pid,helseid://scopes/id
entity/security_level, helseid://scopes/hpr/hpr_number <whatever is required>
    - name: WONDERWALL_REDIS_PASSWORD
      value: $(REDIS_PASSWORD_SESSIONS)
    - name: WONDERWALL_REDIS_USERNAME
      value: $(REDIS_USERNAME_SESSIONS)
    - name: WONDERWALL_REDIS_URI
      value: $(REDIS_URI_SESSIONS)
    - name: WONDERWALL_UPSTREAM_HOST
      value: <your upstream app>
    - name: WONDERWALL_BIND_ADDRESS
      value: $(BIND_ADDRESS)
    - name: WONDERWALL_INGRESS
      value: <same as ingress above>
    - name: WONDERWALL_OPENID_WELL_KNOWN_URL
      value: https://helseid-sts.test.nhn.no/.well-known/openid-configuration
    - name: WONDERWALL_AUTO_LOGIN
      value: "true"
    - name: WONDERWALL_OPENID_CLIENT_ID
      value: <your client id>
```

Deploy the wonderwall app

The easy way is

BAT

```
kubectl -f wonderwall.yml apply
```

But you should probably create a Github action for this.

Deploy your upstream app the usual way

Note that the standard zero trust rules apply, you must allow the Wonderwall instance to be an inbound application.

Hit your your Wonderwall instance with a path accepted by the upstream app.

You will then be forced to authenticate, then profit.