



OpenID Connect e oidc-agent con Kubernetes

Autenticazione tramite Json
Web Token

Stefano E. Zotti

Workshop on
management of distributed resources for genomic communities

29-30 October 2024





OpenID Connect

An authentication layer on top of
the OAuth 2.0 authorization
framework

Motivazione

- Autenticazione su Kubernetes tramite Bearer Token
- Gestione degli utenti centralizzata tramite un Identity Provider (IdP) centralizzato (Keycloak)
- Autorizzazione tramite Role Based Access Control (RBAC) basata sui gruppi di appartenenza
- Ad ogni gruppo associato all'utente viene dato l'accesso al medesimo namespace in k8s (group mapping)

OpenID Connect

- OpenID connect è il layer di autenticazione del più ampio protocollo di autorizzazione OAuth 2.0
- Tecnologia ubiqua nel mondo web
- Basato sui Json Web Token
- Es. «Vuoi accedere a Spotify tramite il tuo profilo Facebook?»

Link utili

- <https://openid.net>
- <https://oauth.net/2/>

ID Token vs Access Token

- Entrambi sono Json Web Token (JWT)
- Il contenuto (*payload*) dell'ID token rappresenta «chi sei?»
- L'Access Token risponde a «cosa puoi fare?»
- Kubernetes gestisce l'autorizzazione tramite RBAC e l'unica informazione necessaria è il *claim groups*
- Per gli scopi del workshop ID Token e Access Token sono utilizzati in modo equivalente (ma non lo sono!)

Link utili

[ID Token vs Access Token](#)

Json Web Token

- Un Json Web Token è costituito da
 - Header
 - Contiene informazioni basilari come l'algoritmo di hash
 - Payload
 - Vero e proprio contenuto del token, dove le informazioni dell'utente sono riportate sottoforma di chiave-valore (*claim*)
 - Signature
 - Firma dell'Identity Provider (IdP) che ha rilasciato il token e che ne prova l'autenticità

Json Web Token

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldeIiA6ICIOQ31ZeFU1SC14Mk1KcWxZUjNZRE5WM3BpYzVvWTJmU0taOVR0Ri1GY3YwIn0..eyJleHaiOjE3Mjk4NjUwNjgsImIhdCI6MTcyOTg2MzI20CwiYXV0aF90aw11IjoxNzI50DYZMjY3LCJqdGkiOiI5NTe1NDk0NS0zOTNhLTrhMzItODdkOS1hM2IzYWYzMTVh0GEiLCJpc3MiOjodHRwczoVl2tjLXNvcnNvbGEtaW50ZWdyYXRpb24uY25hZi5pbmZuLml0L3J1Ywxtcy9zb3Jzb2xhIiwiYXVkJpbInNvcnNvbGEtYXVkJiwiYWNjb3VudCJdLCJzdWIiOjJhMDB1ZTiYzioZnMvhLTQ1NjgtYTg2NC03NzA2ZDY3ZGVhZmMiLCJ0eXAiOjJCZWfyzXXIiLCJhenAiOjJrOHMiLCJzZXNzaWuX3N0YXR1IjoiZTVjOTIyMDUtMGQyYS00ZGVilTkvNmItMTM1MzFi0WJmZGzmIiwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbiM9mZmxpbmVfYWNjZXNzIiwiZGVmYXVsdc1yb2x1cy1zb3Jzb2xhIiwidW1hX2F1dGhvcml6YXRpb24iXX0sInJlc291cmN1X2FjY2VzcyI6eyJhY2NvdW50Ijp7InJvbGVzIjpbiM1hbmFnZS1hY2NvdW50IiwbWFuYwd1LWFjY291bnQtbglua3MiLCJ2aWV3LXByb2ZpbGUiXX19LCJzY29wZSI6Im9wZW5pZCBvZmZsaW51X2FjY2VzcyBwcm9maWx1IGVtYwlslIiwig21kIjoiZTVjOTIyMDUtMGQyYS00ZGVilTkvNmItMTM1MzFi0WJmZGzmIiwizW1haWxfdmVyawZpZWQiOnRydWUsIm5hbWUiOjJKYWNvcG8gR2FzcGFyZXR0byIsImdyb3VwcyI6WyJhZG1pbisInNvcnNvbGEiXSwicHJlZmVycmVkJX3VzZXJuYW11Ijoiamdhc3BhcmV0dG8iLCJnaXZ1b19uYW11IjoiSmFjb3BvIiwiZmFtaWx5X25hbWUiOjJHYXNwYXJldHRvIiwiZW1haWwiOjJqYWNvcG8uZ2Fz cGFyZXR0b0BjbmFmluZm4uaXQi fQ.Q0QJGRzec_89iRzbk0GyhUZGYK1TwFI10TgG943IUw0wcsVoUquOnJUh8cabj8XUTcrG1lqlPpqTp9FslosFdXNvU_Z_KLFNlsZ1VWUjiwp5adDxQj2T2nyR7XEM00eu9k1RwPpoJ7cvYC88ZAmKerao5XaMkIODbWuyBfdXbc1954AEc3s6P4CFTIaZ9LOQFegQrIDSm39wtmTnW5NgUDEa_RsIDxuFC11X3bJWMP19Xjqtavp1PfA4hAU15jG1eUjArJj4NWcqwo_X6UddOKmolRzq1jEeSKBXd7acEDoQiVF_bOK2r2pEEsvdX7Bk31t5LNvzPpKknBs8b2SnPEHcRzB7dDkrcNXIusxWGhe_otUEZNg4FVwKrsDdp6T8WCK2ijzAvwDXxmCX0eY8m6DD1CWh72gDF_dPB3PN6QgNOi00Z5shI2IoQrYdRVkttUJ5Y3hym_4L1Fxv1SwiQ8Zh5P_6qKSBrr2v1c0gNN6_nB_k2Wfk1ClrSts0iIDv00rf3siIqnn1JomgdXk_WXLKyu0Phloz_0aj2b3qeN0orqoxWI2IzrJW9Z1qWbXJBd_nIBjNJKXptQ5tJQ1NyJVEx7PCjTDRd76fAEwKiDPi708D_NnSsRCUeNF0f8yjwDbJ3WeYrnA7QBXRaEwAFDXd8Mvko

- Header
- Payload
- Signature

Json Web Token

Alcuni claim importanti

- **iss (issuer)**: chi ha rilasciato il token
- **iat (issued at)**: a che ora è stato rilasciato il token
- **exp (expire date)**: quando il token non sarà più accettato
- **groups**: gruppi a cui appartiene l'utente e che permetteranno a k8s di gestire l'authZ in base a le regole RBAC definite dall'admin

```

● ● ○ ℗%2                                         jacopo@jacopos-mbp-infn:~
> echo $token | jwt decode -
Token header
-----
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "4CyYxUSH-x2MJqlYR3YDNV3pic5oY2fSKZ9TtF-Fcv0"
}

Token claims
-----
{
  "aud": [
    "sorsola-aud",
    "account"
  ],
  "auth_time": 1729863267,
  "azp": "k8s",
  "email": "jacopo.gasparetto@cnaf.infn.it",
  "email_verified": true,
  "exp": 1729865068, ←
  "family_name": "Gasparetto",
  "given_name": "Jacopo",
  "groups": [ ←
    "admin",
    "sorsola"
  ],
  "iat": 1729863268, ←
  "iss": "https://kc-sorsola-integration.cnaf.infn.it/realms/sorsola", ←
  "jti": "95154945-393a-4a32-87d9-a3b3af315a8a",
  "name": "Jacopo Gasparetto",
  "preferred_username": "jgasparetto",
  "realm_access": {
    "roles": [
      "offline_access",
      "default-roles-sorsola",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "openid offline_access profile email",
  "session_state": "e5c92205-0d2a-4deb-916b-13531b9bfdff",
  "sid": "e5c92205-0d2a-4deb-916b-13531b9bfdff",
  "sub": "a00ee22f-36ea-4568-a864-7706d67deafc",
  "typ": "Bearer"
}

```

oidc-agent

oidc-agent è un tool opensource che permette di interrogare l'IdP e farsi rilasciare token a nome dell'utente attraverso command line.

Per ottenere un token è necessario

- Avviare il servizio (se non già avviato)
- Associare un client registrato in Keycloak (issuer) (una tantum)
- [Ri-autenticarsi qualora la sessione sia scaduta]
- Richiedere un token

oidc-agent: associazione del client

- Avviare il servizio (se non già avviato)
 - eval \$(oidc-agent-service start)
- Associazione di un client (pre-configurato in Keycloak)
 - oidc-gen **sorsola** --client-id **k8s** --iss **https://kc-sorsola-integration.cnaf.infn.it/realms/sorsola** --pub -w **device**
 - **friendly name** da dare al client (sorsola)
 - **--client-id k8s** ID del client registrato in Keycloak
 - **--iss https://kc-sorsola-integration.cnaf.infn.it/realms/sorsola** issuer (Keycloak)
 - **--pub** indica che stiamo associando un client pubblico (senza client_secret)
 - **-w--flow device** rappresenta il Device Grant Flow, ovvero un flusso di autenticazione per dispositivi che non hanno un browser

oidc-agent: associazione del client

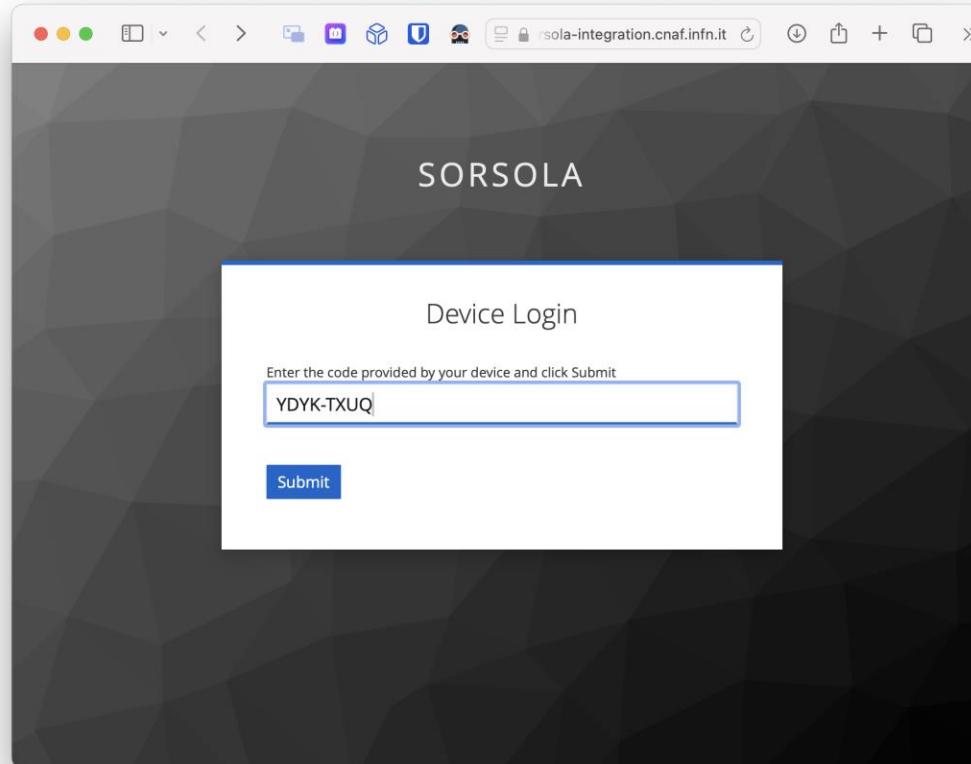
- Dopo aver lanciato il comando, alla domanda relativa a quali scope richiedere, digitare max per ottenerli tutti
- Copiarsi il codice generato
- Incollare il codice nell'URL indicato dopo essersi autenticati

```
● ● ● ℗%2
oidc-gen sorsola --client-id k8s --iss --pub -w device
> oidc-gen sorsola --client-id k8s --iss https://kc-sorsola-integration.cnaf.infn.it/realm...sorsola --pub -w device
No account exists with this short name. Creating new configuration ...
The following scopes are supported: offline_access profile roles email aud openid
Scopes or 'max' (space separated) [openid profile offline_access]: max ← 1. Digitare 'max'
Redirect_uris (space separated):
Generating account configuration ...
accepted

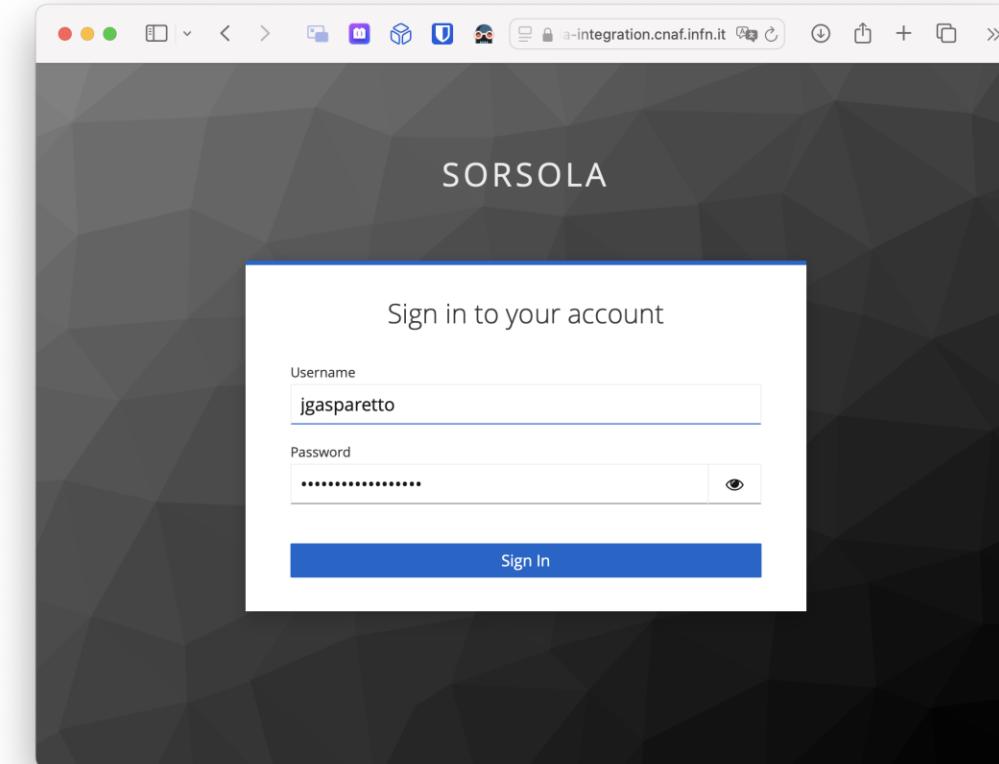
Using a browser on any device, visit:
https://kc-sorsola-integration.cnaf.infn.it/realm...sorsola/device ← 3. Aprire il link ed incollare il codice
And enter the code: YDYK-TXUQ ← 2. Copiare il codice
Alternatively you can use the following QR code to visit the above listed URL.


```

oidc-agent: associazione del client

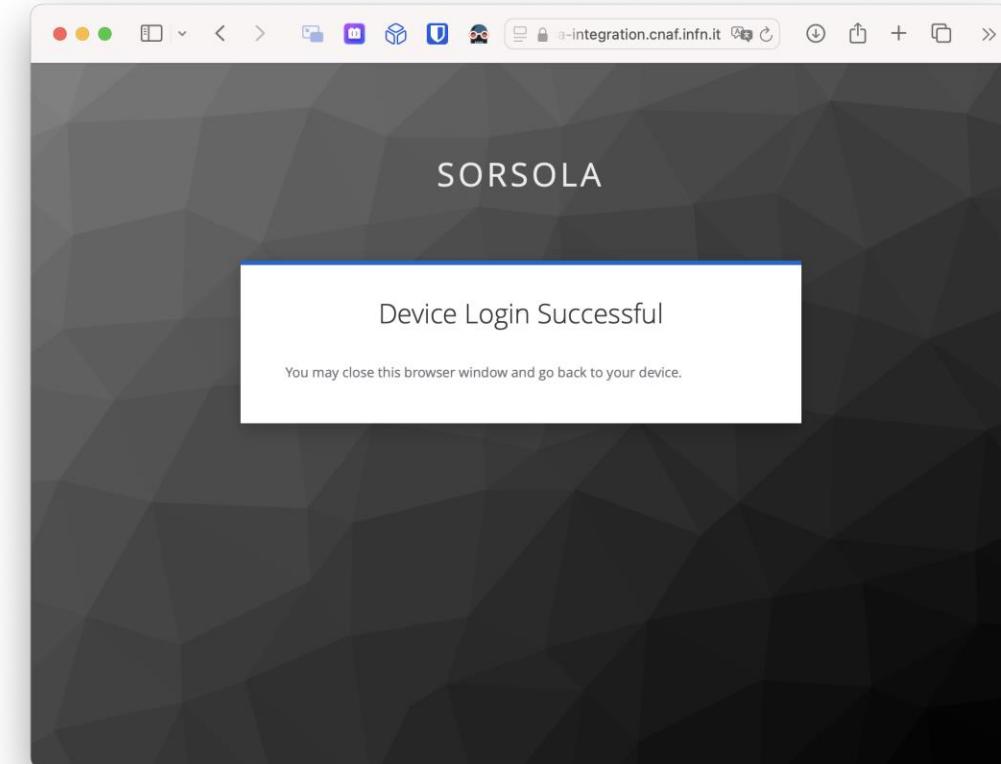
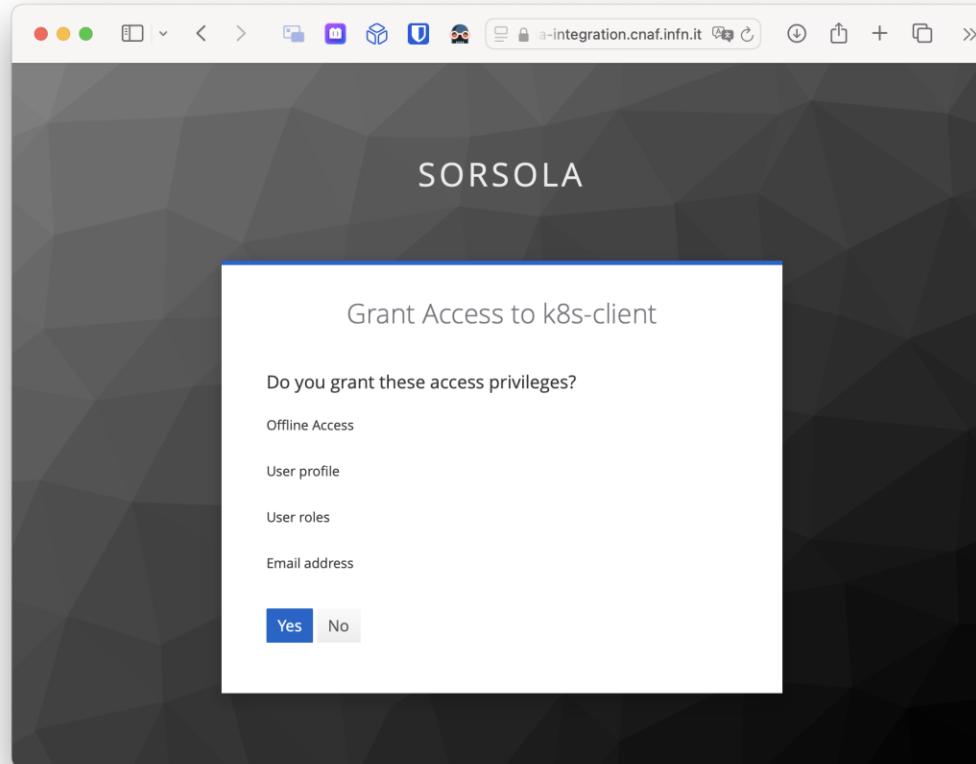


The screenshot shows a web browser window with the URL "sola-integration.cnaf.infn.it". The main header says "SORSLA". A modal dialog box titled "Device Login" is displayed. It contains the instruction "Enter the code provided by your device and click Submit". Below this is a text input field containing the code "YDYK-TXUQ". At the bottom is a blue "Submit" button.



The screenshot shows a web browser window with the URL "sola-integration.cnaf.infn.it". The main header says "SORSLA". A modal dialog box titled "Sign in to your account" is displayed. It has two input fields: "Username" with the value "jgasparetto" and "Password" with a masked value. At the bottom is a blue "Sign In" button.

oidc-agent: associazione del client



oidc-agent: associazione del client

- Una volta dato il consenso sulla pagina di Keycloak, e dopo che quest'ultimo ha dato esito positivo, tornare sul terminale
- Inserire una password per proteggere il client o lasciare il campo vuoto per non usare alcuna password

```

● ● ● 7%2
jacopo@jacopos-mbp-infn:~

The following scopes are supported: offline_access profile roles email aud openid
Scopes or 'max' (space separated) [openid profile offline_access]: max
Redirect_uris (space separated):
Generating account configuration ...
accepted

Using a browser on any device, visit:
https://kc-sorsola-integration.cnaf.infn.it/realmssorsola/device

And enter the code: YDYK-TXUQ
Alternatively you can use the following QR code to visit the above listed URL.



```

oidc-agent: ri-autenticazione

- Dopo un riavvio della VM, riavviare il servizio con

```
eval $(oidc-agent-service start)
```

e caricare il profilo sorsola

```
oidc-add sorsola
```

- Qualora la sessiona sia scaduta (oidc-agent dovrebbe indicarlo) e/o oidc-agent rilascia token che non vengono più accettati di Kubernetes, sarà necessario effettuare una nuova autenticazione tramite il comando

```
oidc-gen sorsola --reauthenticate -w device
```

- La procedura di autenticazione sarà simile a quella effettuata in fase di associazione. Sarà dunque necessario copiare il codice mostrato e incollarlo all'indirizzo indicato

oidc-agent: richiesta di un token

- Per richiedere un Access Token

```
oidc-token sorsola
```

- Per salvarsi il token (che ricordiamo avere una durata limitata, es. 60 minuti)

```
token=$(oidc-token sorsola)
```

- Per utilizzare il token appena ottenuto con kubernetes

```
kubectl --token=$token get pod
```

oppure

```
alias k="kubectl --token=$(oidc-token sorsola)"  
k get pod
```

Nota: ogni volta che si richiede un nuovo token sarà necessario rieffettuare l'alias.



Ringrazio Jacopo Gasparetto
per l'aiuto con queste slide e
tutto il team del CNAF