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

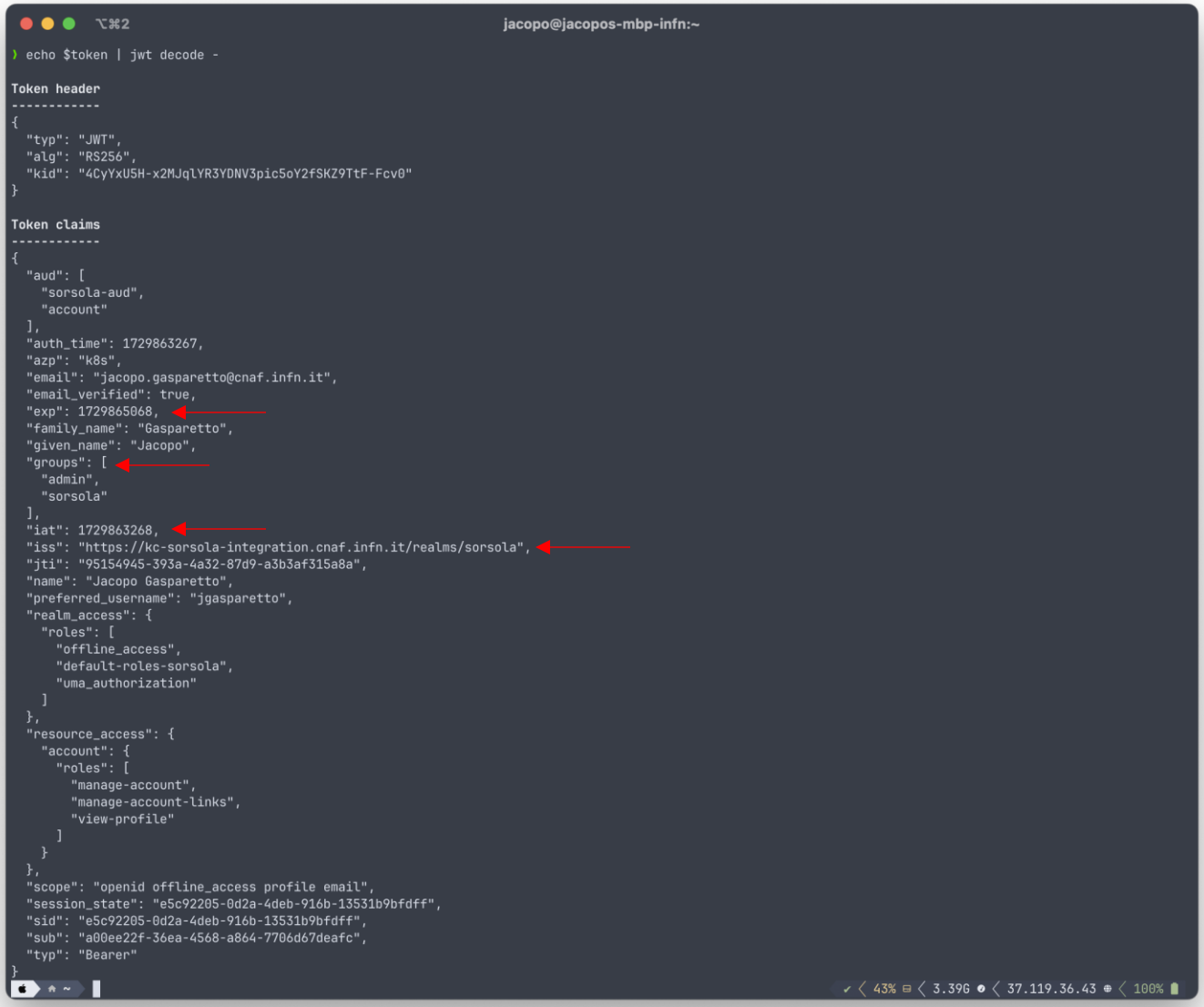

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI0Q3lZeFU1SC14Mk1KcWxZUjNZRE5WM3BpYzVvWTJmU0taOVR0Ri1GY3Yw
In0.eyJleHAiOjE3Mjk4NjUwNjgsImlhdCI6MTcyOTg2MzI2OCwiYXV0aF90aW1lIjoxNzI5ODYzMjY3LCJqdGkiOiI5NTE1NDk0NS0zOTNh
LTRhMzItODdkOS1hM2IzYWYzMTVhOGEiLCJpc3MiOiJodHRwczovL2tjLXNvcnNvbGEtaW50ZWdyYXRpb24uY25hZi5pbmZuLml0L3JlYWxt
cy9zb3Jzb2xhIiwiYXVkIjpbInNvcnNvbGEtYXVkIiwiYWNjb3VudCJdLCJzdWIiOiJhMDBlZTIyZi0zNmVhLTQ1NjgtYTg2NC03NzA2ZDY3
ZGVhZmMiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJrOHMiLCJzZXNzaW9uX3N0YXRlIjoiZTVjOTIyMDUtMGQyYS00ZGViLTkxNmItMTM1MzFi
OWJmZGZmIiwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbIm9mZmxpbmVfYWNjZXNzIiwiZGVmYXVsdC1yb2xlcy1zb3Jzb2xhIiwidW1hX2F1
dGhvcml6YXRpb24iXX0sInJlc291cmNlX2FjY2VzcyI6eyJhY2NvdW50Ijp7InJvbGVzIjpbIm1hbmFnZS1hY2NvdW50IiwibWFuYWdlLWFj
Y291bnQtbGlua3MiLCJ2aWV3LXByb2ZpbGUiXX19LCJzY29wZSI6Im9wZW5pZCBvZmZsaW5lX2FjY2VzcyBwcm9maWxlIGVtYWlsIiwic2lk
IjoiZTVjOTIyMDUtMGQyYS00ZGViLTkxNmItMTM1MzFiOWJmZGZmIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsIm5hbWUiOiJKYWNvcG8gR2Fz
cGFyZXR0byIsImdyb3VwcyI6WyJhZG1pbiIsInNvcnNvbGEiXSwicHJlZmVycmVkX3VzZXJuYW1lIjoiamdhc3BhcmV0dG8iLCJnaXZlbl9u
YW1lIjoiSmFjb3BvIiwiZmFtaWx5X25hbWUiOiJHYXNwYXJldHRvIiwiZW1haWwiOiJqYWNvcG8uZ2FzcGFyZXR0b0BjbmFmLmluZm4uaXQi
fQ.QOQJGRzec_89iRzbkOGyhUZGYKlTWfI10TgG943IUw0wcsVoUquOnJUh8cabj8XUTcrG1lqLaPpqTp9FslosFdXNvU_Z_KLFNlsZlVWUj
iwp5adDxQj2T2nyR7XEM00eu9klRwPpoJ7cvYC88ZAmKerao5XaMkIODbWuyBfdXbci954AEc3s6P4CFTIaZ9LOQFegQrIDSm39wtmTnW5Ng
UDEa_RsIDxuFCl1X3bJWMP19XjqtavP1PfA4hAUl5jG1eUjArJj4NWcqwO_X6UddOKmolRzq1jEeSKBXd7acEDoQiVF_bOK2r2pEEsvdX7Bk
31t5LNvzPpKknBs8b2SnPEHcRzB7dDkrcNXIusxWGhe_otUEZNg4FVwKrsDdpy6T8WCk2ijzAvwDXxmCX0eY8m6DD1CWh72gDF_dPB3PN6Qg
NOiO0Z5shI2IoQrYdRVkttUJ5Y3hym_4L1Fxv1SwiQ8Zhz5P_6qKSbRr2v1cOgNN6_nB_k2WfklClrSts0iIDvOORf3siIqnnlJomgdXk_WX
LKyu0Phloz_0aJ2b3qeNOorqoxWI2IZrJW9Z1qWbXJBd_nIBjNJKXptQ5tJQ1NyJVEy7PCjTDRd76fAEwKiDPi708D_NnSsRCUeNF0f8yjwD
bJ3WeYrnA7QBXRaEwAFDXd8Mvko
```


- 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



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)

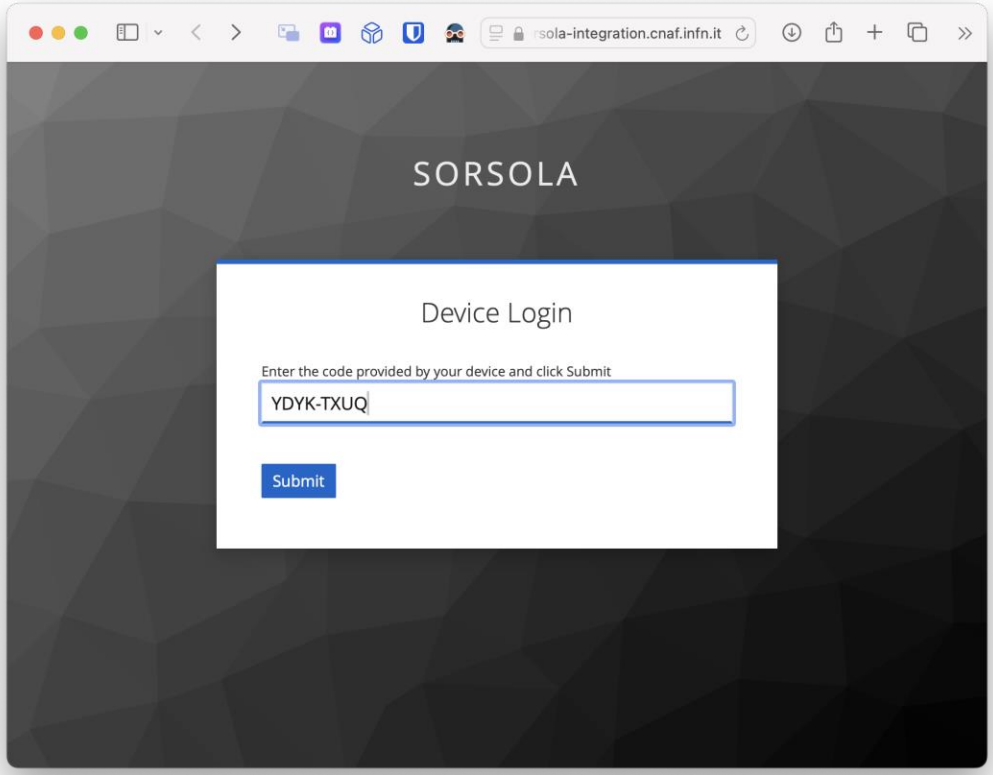
  - `oidc-gen sorsola --client-id k8s --iss https://kc-sorsola-integration.cnaf.infn.it/realms/sorsola --pub -w device`

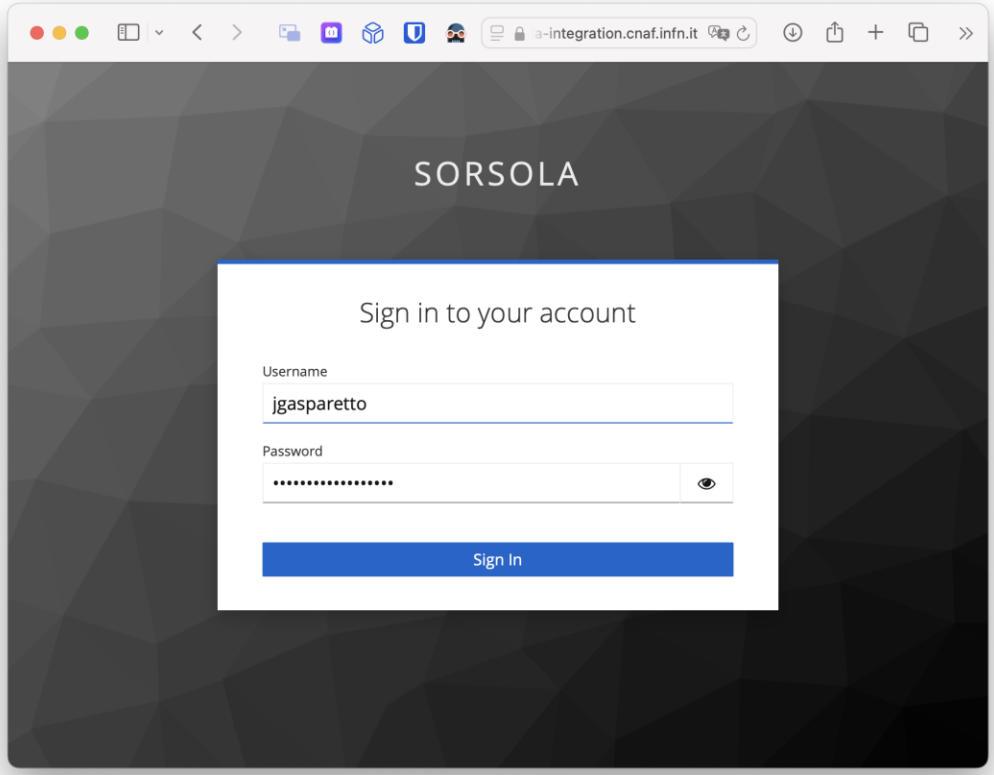
    - `friendly name` da dare al client (sorsola)
    - `--client-id k8s` ID del client registrato in Keycloak
    - `--iss https://kc-sorsola-integration.cnaf.infn.it/realms/sorsola` issuer (Keycloak)
    - `--pub` indica che stiamo associando un client pubblico (senza client_secret)
    - `-w/--flow` device rappresenta il Device Grant Flow, ovvero un flusso di autenticazione per dispositivi che non hanno un browser

oidc-agent: associazione del client

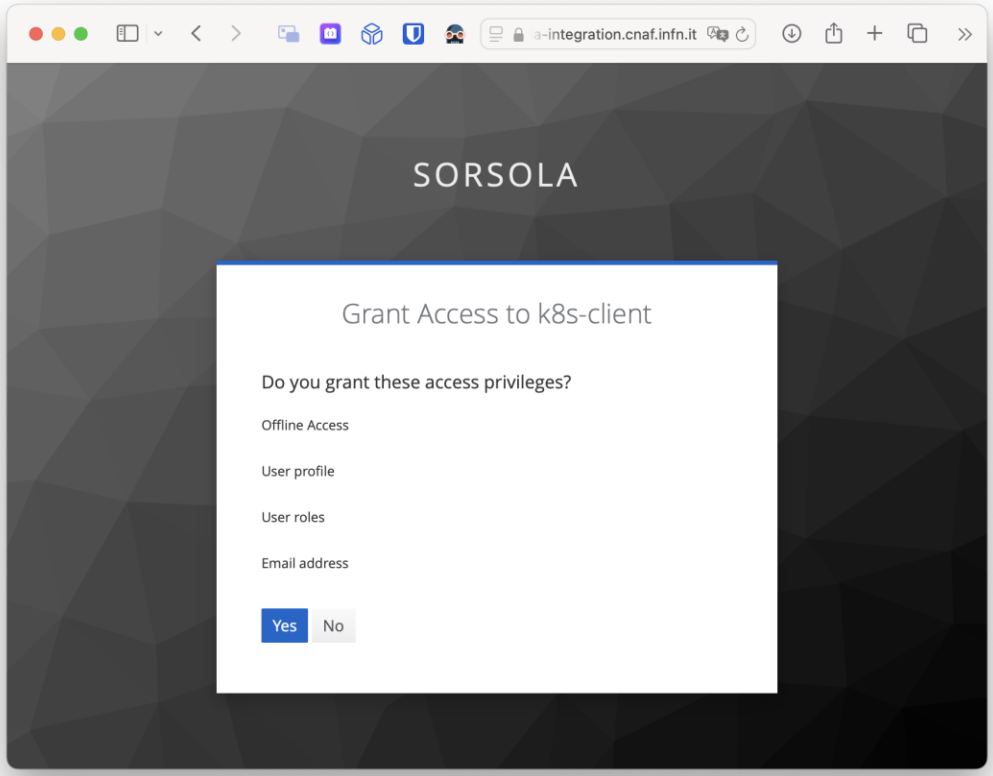
- Dopo aver lanciato il comando, alla domanda relativa a quali scope richiedere, digitare max per ottenerli tutti
- Copiarsi il codice generato
- Incollare il codice nell'URL indicato dopo essersi autenticati

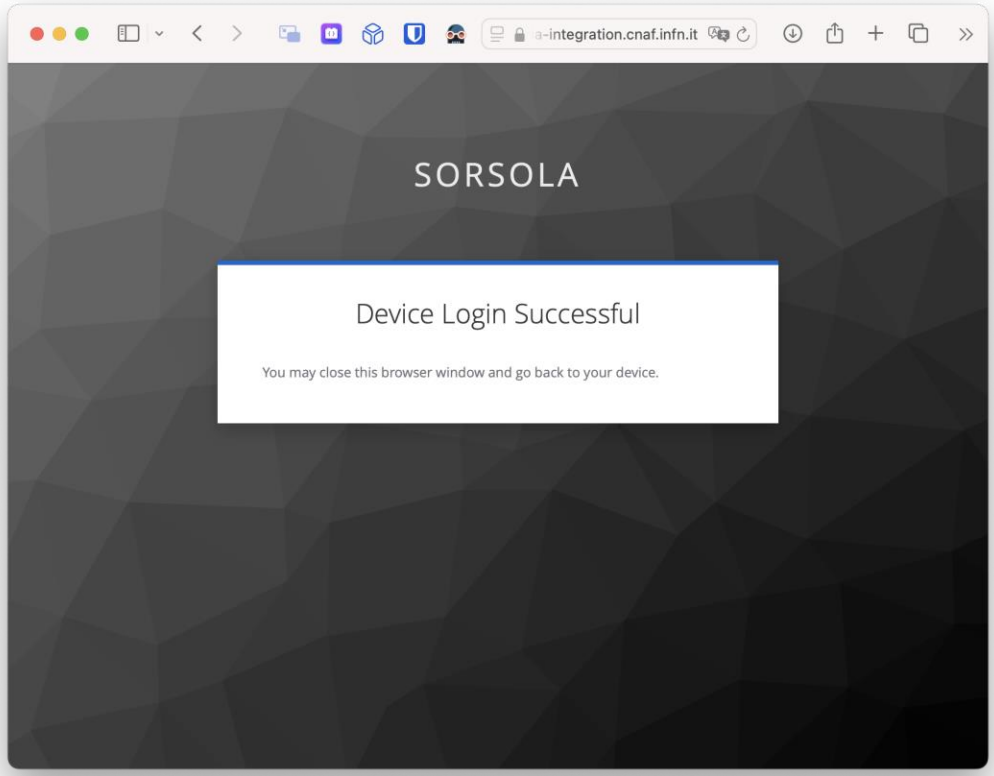
oidc-agent: associazione del client





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

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