A Complete Guide to OpenID Connect with OpenStack

Kristi Nikolla

MOC Alliance

Open Infra Summit 2023 Vancouver

About me

- Senior Software Engineer at the MOC Alliance (massopen.cloud)
  - We've been running OpenStack with OpenID Connect using Keycloak as an IdP since 2017.
- Core Reviewer on Keystone, Former Keystone PTL (Victoria & Wallaby)
- Chair of the OpenStack Technical Committee

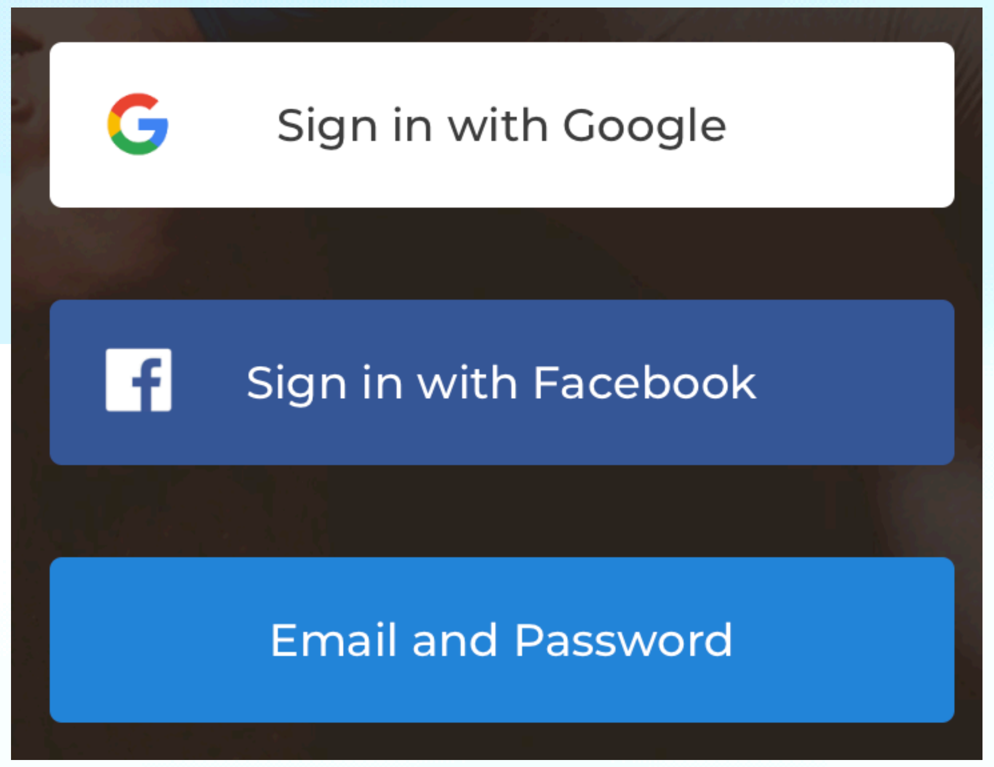
Glossary

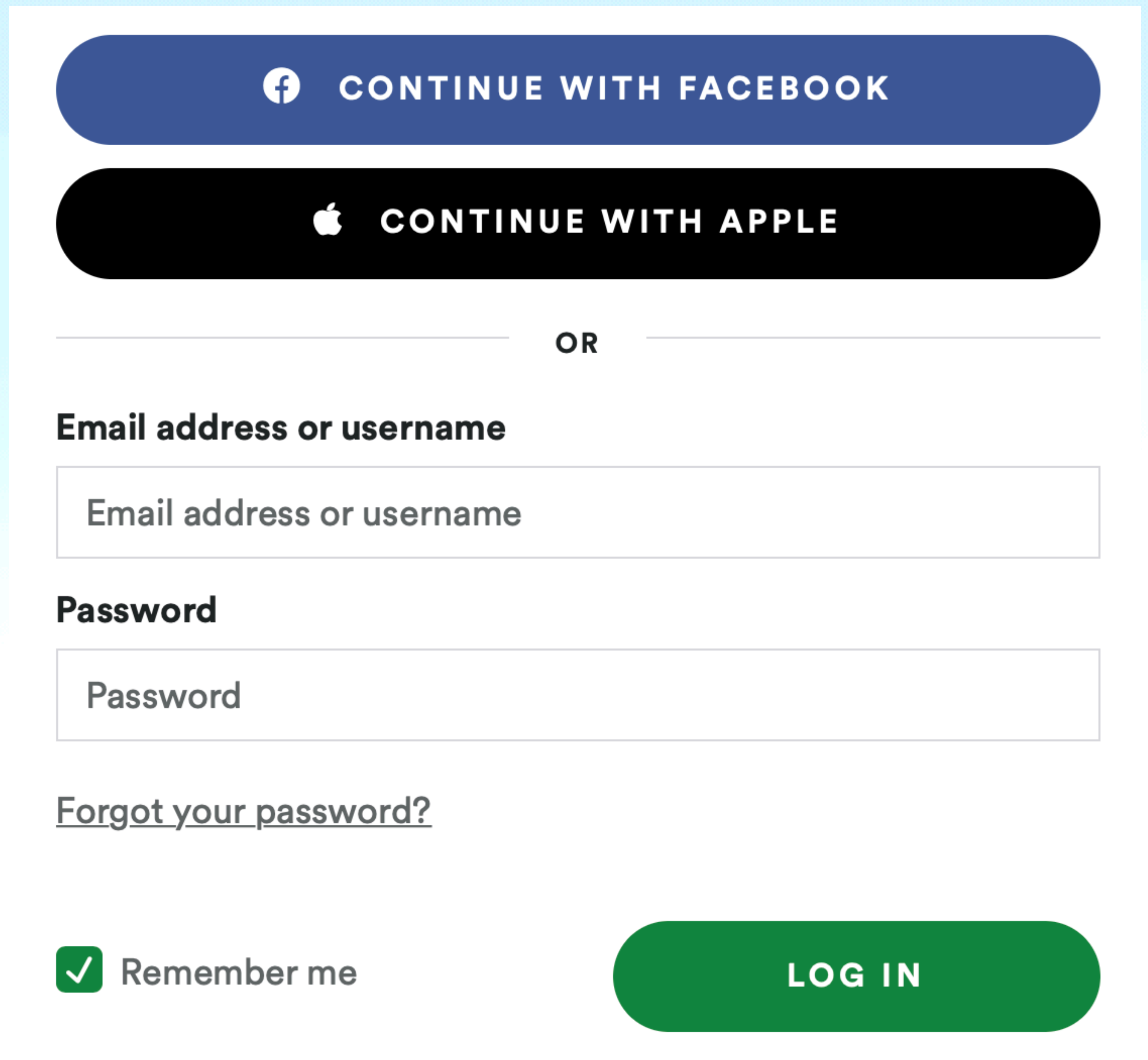
- **Authentication** - the act of verifying a user's identity.
- **Authorization** - the act of granting access to specific resources.
- **Token** - string that carries a user's authorization, may be opaque or may be signed and encrypted.
- **Identity** - represents a user (be it human or machine).

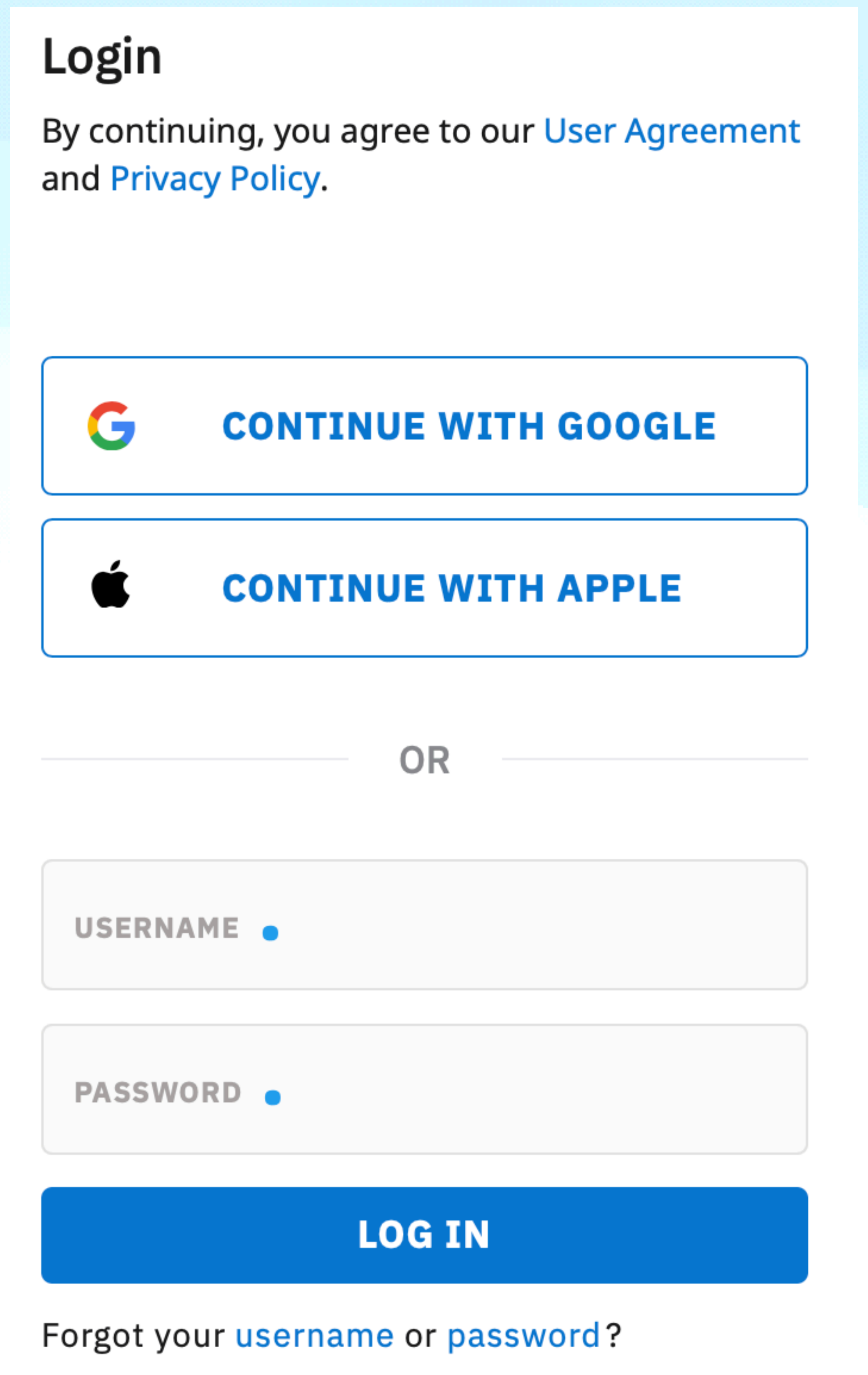
What is identity federation

- Identity federation is the ability to **share identity information** across multiple identity management systems.
  - These may be across organizational boundaries.
- Enables Single Sign-On
  - User authenticates directly with the identity source rather than having multiple sets of credentials.
- The most common protocols are **OpenID Connect** and **SAML2**.

Examples






Login

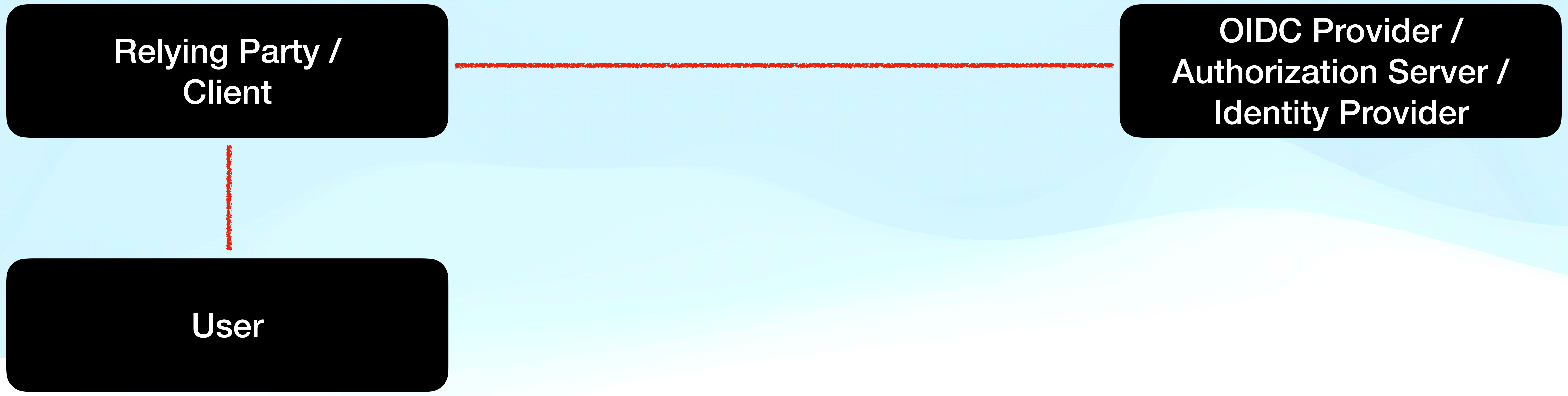
By continuing, you agree to our User Agreement and Privacy Policy.

Forgot your username or password?

What is OpenID Connect

- OpenID Connect provides a standard protocol that is built on top of the OAuth 2.0 Authorization Framework.
- **Standardizes** on
    - an **id_token** - in JWT format which contains identity information and the format of that information (**claims**).
    - User Information Endpoint
- Different flows define the interaction between the entities
    - Most common is the **Authorization Code** flow.

Glossary



A **client** will act on behalf of a **user**.

A **client** will authenticate with an **authorization server** and receive an **access token** and an **id token**.

A **client** will verify a **token** (potentially with the **authorization server**).

Example identity information

```
{
  "sub": "32f28601-ac39-4a5b-9edf-422ccc526f1a",
  "email_verified": true,
  "name": "Kristi Nikolla",
  "groups": [
    "devops",
    "staff"
  ],
  "preferred_username": "knikolla@bu.edu",
  "given_name": "Kristi",
  "family_name": "Nikolla",
  "email": "knikolla@bu.edu"
}
```

Why integrate with OpenID Connect

- To bring your existing identity and access management into OpenStack.
  - Allowing users to authenticate with their already existing accounts.
- To architect an identity and access management system that isn't centered around OpenStack.
- To use tools that aren't built for OpenStack.

Keystone as a client

- Allows you to use your existing identity provider with OpenStack.
  - Ex. Keycloak, Dex, Azure AD, etc.
- Users can use their existing accounts and credentials.
- Keystone never sees the user's credentials.

Red Hat OpenStack Platform

If you are not sure which authentication method to use, contact your administrator.

Keystone Credentials [✓ OpenID Connect]

Red Hat
Connect

CILogon

Consent to Attribute Release

MSS Keycloak requests access to the following information. If you do not approve this request, do not proceed.

- Your CILogon user identifier
- Your name
- Your email address
- Your username and affiliation from your identity provider

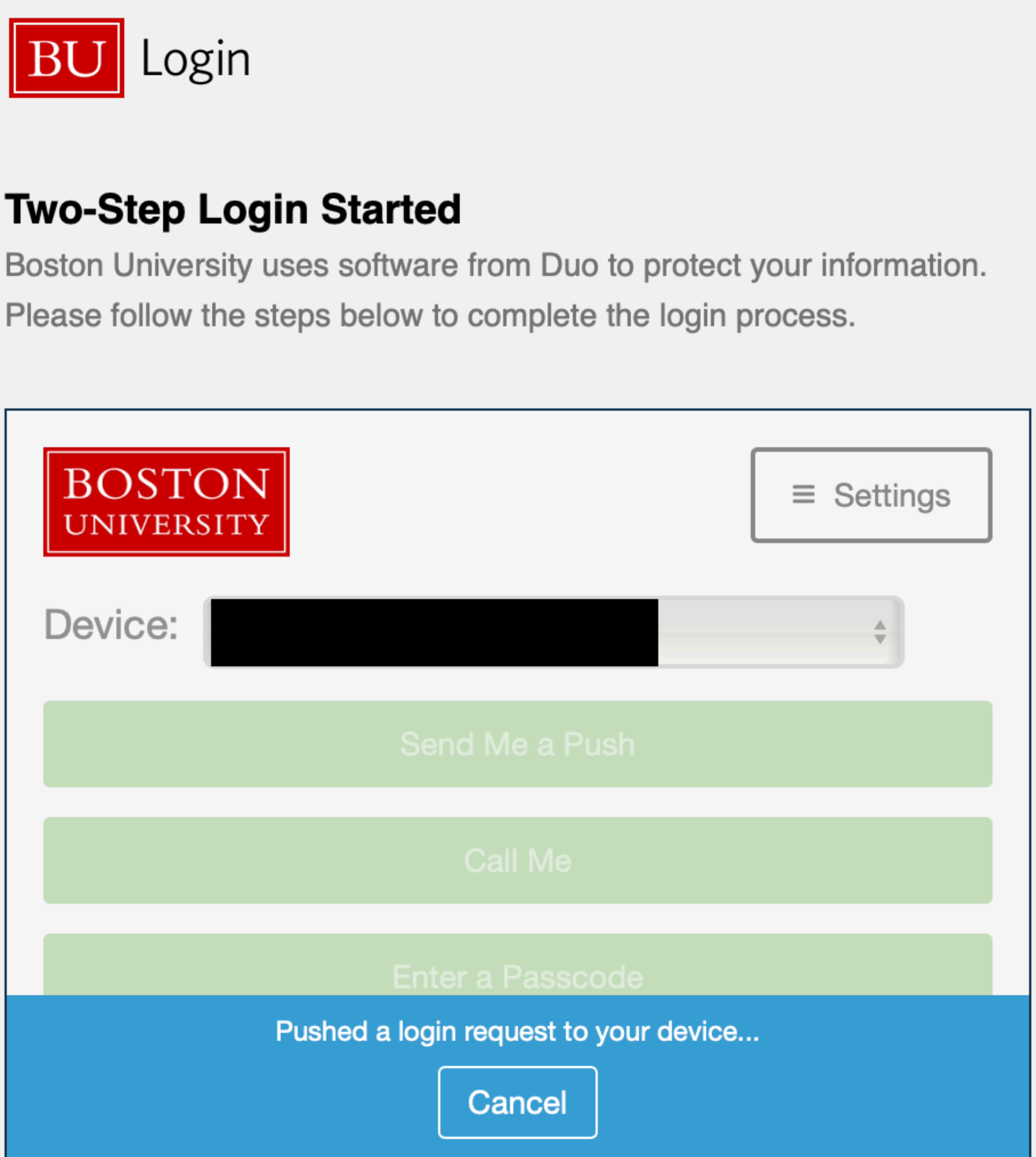
Selected Identity Provider

Boston University
Remember this selection

By selecting "Log On", you agree to the privacy policy.

BU Login
BU login name
password
Continue

You have asked to login to CILogon


BU Login

Two-Step Login Started

Boston University uses software from Duo to protect your information.
Please follow the steps below to complete the login process.

BOSTON UNIVERSITY

≡ Settings

Device: [redacted]

Send Me a Push

Call Me

Enter a Passcode

Pushed a login request to your device...

Cancel

Red Hat OpenStack Platform | Project | Identity | Project | Help | knikolla@b

ject | Compute | Volumes | Network | Orchestration | Object Store

erview | Instances | Images | Key Pairs | Server Groups

ect / Compute / Overview

verview

mit Summary

mpute

Instances	**VCPUs**	**RAM**
Used 1 of 1	Used 2 of 2	Used 4GB of 4GB

ume

Volumes	**Volume Snapshots**	**Volume Storage**
Used 2 of 2	Used 0 of 10	Used 50GB of 100GB

work

Floating IPs	**Security Groups**	**Security Group Rules**	**Networks**	**Ports**	**Routers**
Allocated 1 of 50	Used 2 of 10	Used 7 of 100	Used 1 of 100	Used 3 of 500	Used 1 of 10

Authentication

- User authenticates in the identity provider and is sent back...
  1. Apache, using **mod_auth_oidc**, does the OpenID Connect magic and passes the **claims** to Keystone.
  2. **Locate** or **Create** the correct user and update its information based on the received claims.
  3. Keystone redirects the user to Horizon with an **OpenStack token**.
- This is done through **mappings**.
  - To associate a mapping with an **identity provider**, we specify a **federation protocol**. For OpenID Connect use **openid** or **mapped**.

CLI Operations

```
$ openstack identity provider create \
  --remote-id https://keycloak.mss.mghpcc.org/auth/realms/mss \
  --domain default \
  keycloak

$ openstack mapping create \
  --rules mapping.json \
  sso_oidc_mapping

$ openstack federation protocol create \
  --identity-provider keycloak
  --mapping sso_oidc_mapping \
  openid
```

Add **openid** to /etc/keystone.conf [auth] methods

Mappings

```
{
    "rules": [
        {
            "local": [
                {
                    <user>
                    [<group>]
                    [<project>]
                }
            ],
            "remote": [
                {
                    <match>
                    [<condition>]
                }
            ]
        }
    ]
}
```

Example mappings

```
$ openstack mapping show keycloak -f json -c rules
{
  "rules": [
    {
      "local": [
        {
          "user": {
            "name": "{0}",
            "email": "{1}",
            "domain": {
              "id": "default"
            }
          }
        }
      ],
      "remote": [
        {
          "type": "OIDC-preferred_username"
        },
        {
          "type": "OIDC-email"
        }
      ]
    }
  ]
}
```

```
$ openstack user show 9907dfdb494a467595be2317300a8d55 -f json
{
  "domain_id": "default",
  "email": "knikolla@bu.edu",
  "enabled": true,
  "federated": [
    {
      "idp_id": "keycloak",
      "protocols": [
        {
          "protocol_id": "openid",
          "unique_id": "knikolla%40bu.edu"
        }
      ]
    }
  ],
  "id": "9907dfdb494a467595be2317300a8d55",
  "name": "knikolla@bu.edu",
  "options": {},
  "password_expires_at": null
}
```

Mapping conditions

- Conditions to filter users based on the value of the claim
  - **any_one_of**
  - **not_any_of**
- Conditions to filter values in claims
  - **blacklist**
  - **whitelist**
- Regex is also supported.

Mapping conditions example

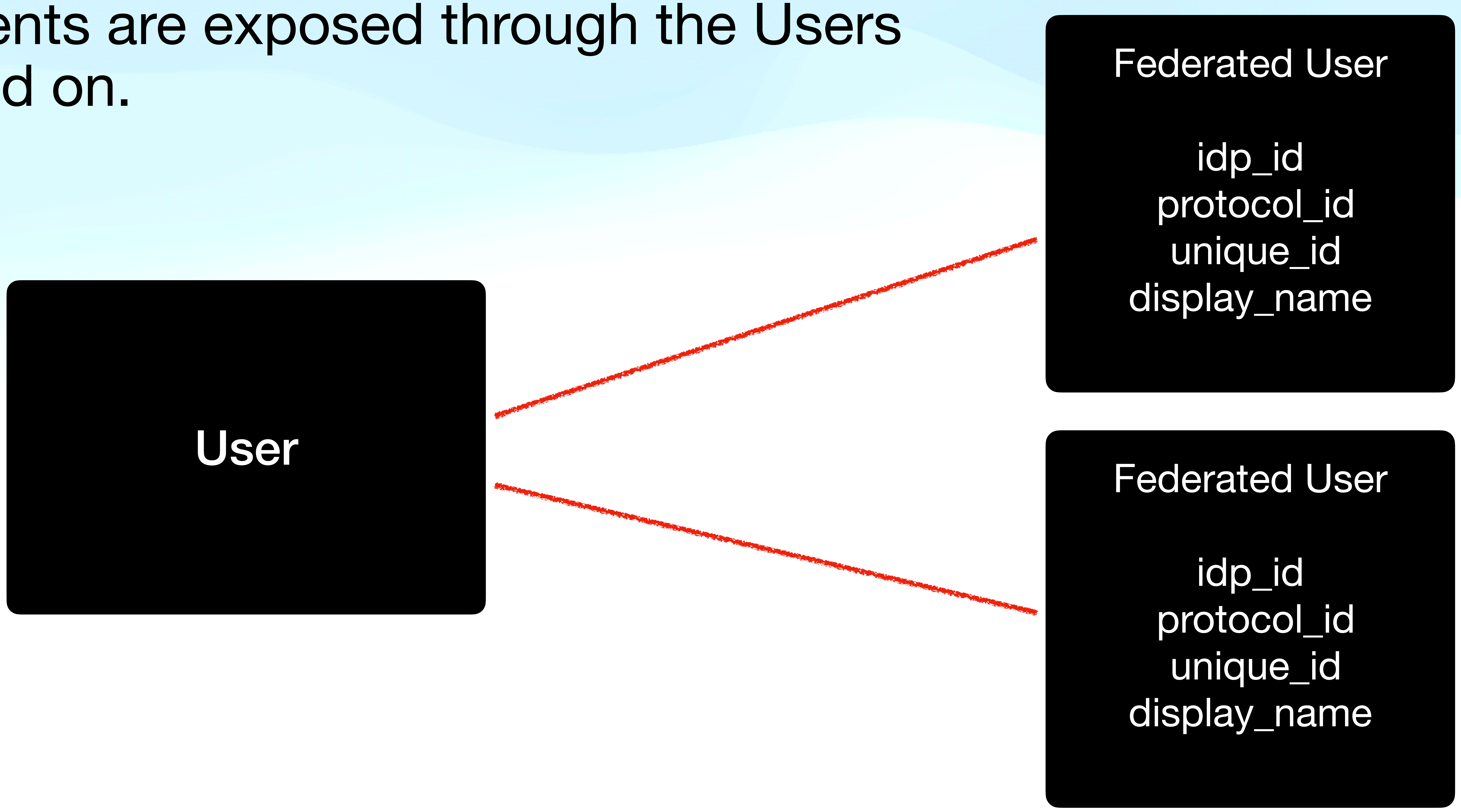
```
{
    "rules": [
        {
            "local": [
                {
                    "user": {
                        "name": "{0}"
                    }
                },
                {
                    "groups": "{1}",
                    "domain": {
                        "name": "Default"
                    }
                }
            ],
            "remote": [
                {
                    "type": "OIDC-preferred_username"
                },
                {
                    "type": "OIDC-groups",
                    "whitelist": [
                        "Developers",
                        "OpsTeam"
                    ]
                }
]}]}
```

Regex example

```
{
    "type": "OIDC-email",
    "any_one_of": [
        ".*@yeah.com$"
    ],
    "regex": true
},
{
    "type": "OIDC-groups",
    "whitelist": [
        "Project.*$"
    ],
    "regex": true
}
```

What's in a Keystone user?

- A Keystone user is a collection of components.
- The federated components are exposed through the Users API and can be operated on.



Pre-creating a federated user

```
payload = {
    "user": {
        "domain_id": "default",
        "enabled": True,
        "name": "knikolla@bu.edu",
        "email": "knikolla@bu.edu",
        "federated": [
            {
                "idp_id": "keycloak",
                "protocols": [
                    {
                        "protocol_id": "openid",
                        "unique_id": "knikolla%40bu.edu"
                    }
                ]
            }
        ]
    }
}
session.post("https://keystone/v3/users", json=payload)
```

Authorization

- You can assign roles on a federated user
- Carrying groups through mappings
- Project auto-provisioning

Group membership through mappings

- By default these group assignment are only valid for the duration of the token.
- Can be persisted for a limited amount of time by setting
    - /etc/keystone.conf [federation] default_authorization_ttl = 0
    - Per identity provider -> authorization_ttl

Project auto-provisioning

- If the project doesn't exist it's created.
- Role assignments on this project aren't kept up to date.

Project auto-provisioning example

```
{
    "rules": [
        {
            "local": [
                {
                    "user": {
                        "name": "{0}"
                    }
                },
                {
                    "projects": [
                        {
                            "name": "Project for {0}",
                            "roles": [
                                {
                                    "name": "member"
                                }]}]}
            ],
            "remote": [
                {
                    "type": "OIDC-preferred_username"
                }
 ]}]}
```

API access

- Application credentials
  - Bypasses external authentication.
- OIDC Password
- **OIDC Device Authorization grant**

Get this running for yourself

- https://github.com/knikolla/devstack-plugin-oidc/
- The Keystone Devstack plugin supports setting up Keystone integrated with Keycloak.
- Keycloak running as a container using docker-compose.

```
git clone https://opendev.org/openstack/devstack.git

cp samples/local.conf .
echo "
    ENABLED_SERVICES=key,mysql,rabbit
    enable_plugin keystone https://github.com/openstack/keystone
    enable_service keystone-oidc-federation
" >> local.conf
./stack.sh
```

Keystone as an authorization server

- Keystone has limited support for some OAuth 2.0 grants
  - Client Credentials grant
  - OAuth 2.0 mTLS
- Keystonemiddleware can be configured to receive a token through "Authorization: Bearer" instead of "X-Auth-Token".
- Allows for some compatibility between your existing applications or clients and Keystone.

Using an external authorization server

- We still live in a world where OpenStack only works with tokens issued by Keystone.
- Proposal to implement support for OAuth 2.0 Token Introspection in Keystonemiddleware
  - https://review.opendev.org/c/openstack/keystone-specs/+/861554
  - Discussion in Keystone PTG sessions on Wednesday

Future Work

- Documentation for some of the features in Keystone is still under review and hasn't merged yet.
- Implement more features of a OpenID Provider / Authorization Server.

Thank you & Questions

Slides will be published on kristi.nikolla.me