devfest

Wifi:

Enterprise Workshops: https://bit.ly/47dIkav

Workshop Repo:

- start - https://bit.ly/3uh8HOs
- completed https://bit.ly/47wBhJK

Slides:
https://isemona.github.io/oidc-workshop-devfest.pdf

Google Developer Groups

Editable Location

You will need:

- a laptop with admin access
- Node v18+
- terminal access
- an IDE that supports TypeScript, such as Visual Studio Code

```
Text(
  'Simple Statement or URL',
  style: TextStyle(
    color: Colors.green[200],
  ),
),
```

devfest

```
s.star,
r: Colors.green[500],

Text('23'),
```

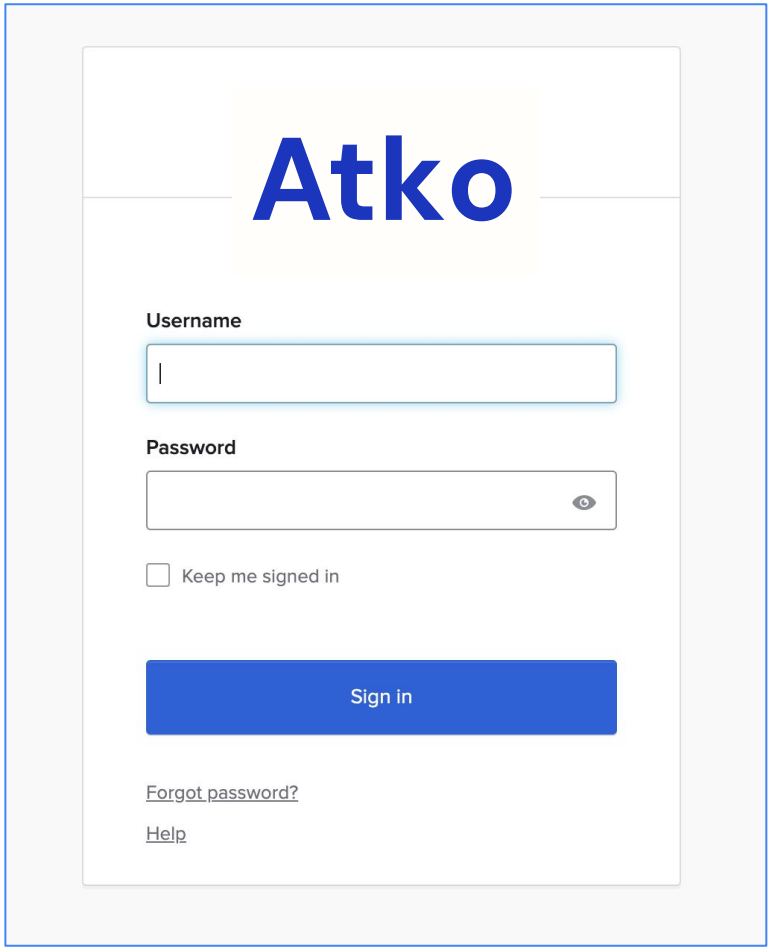
Build a lasting relationship with your enterprise customers

Who am I?

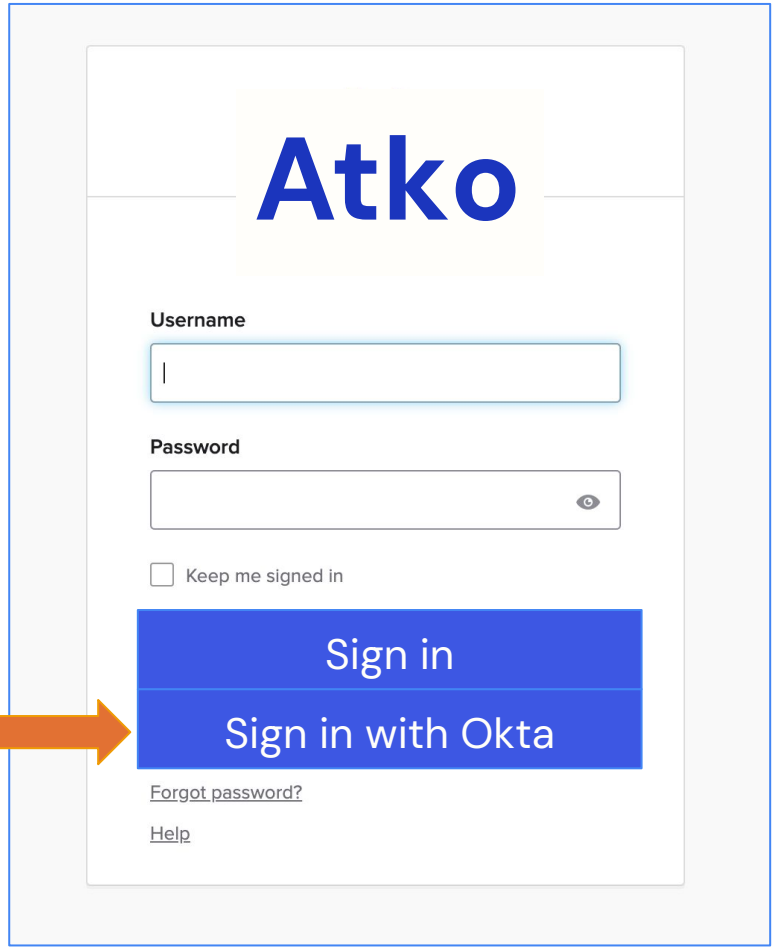

```
{
  "sub": "00upp1hf16Sn1H2fb0h7",
  "name": "Semona Igama",
  "email": "semona.igama@okta.com",
  "ver": 1,
  "iss": "https://semonav2.oktapreview.com",
  "aud": "0oaqh3fceytYR7U1I0h7",
  "iat": 1698872857,
  "exp": 1698876457,
  "jti": "ID.sGTbaSiUQUCa6SKbqZMDPfrJWjB5t-XoFzY66R0ERl4",
  "amr": [
    "pwd"
  ],
  "role": "Developer Advocate at Okta",
  "twitter_handle": "@sudomonas",
  "fav_animal": "dolphin",
  "sport": "ultimate",
  "idp": "00opp1hezxr8yhXvQ0h7",
  "nonce": "rhkhjmq9oc",
  "preferred_username": "semona.igama@okta.com",
  "auth_time": 1698872820,
  "at_hash": "acaWBJnAi1Y6ZNYLPRJteg",
  "c_hash": "0fGTQcMQWaQtVl5keiE5sQ"
}
```

What's a good login experience like?

- Follows standard security protocols
- Allows identity provider login options for Single Sign On (SSO)



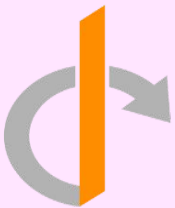
Identity Provider
(IdP e.g Okta, Auth0)
OpenID Connect (OIDC)




THIS IS
THE WAY

What is OpenID Connect (OIDC) and why use it?

Authentication →

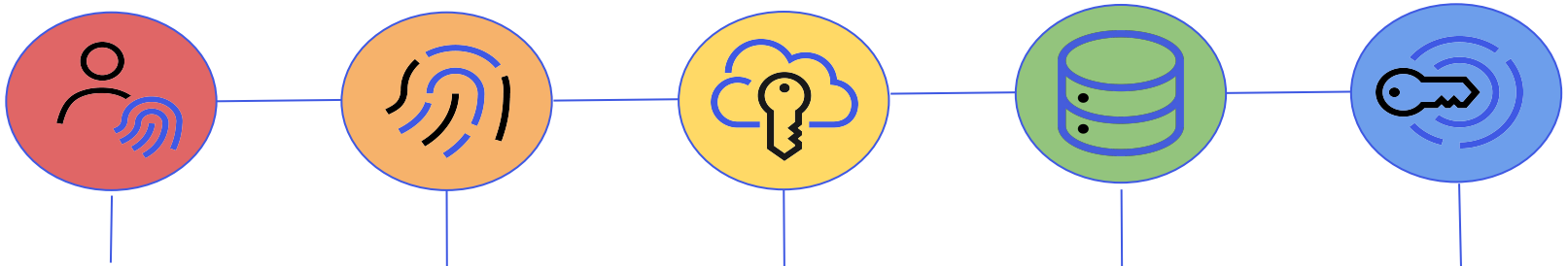


OpenID Connect

Authorization →

OAuth 2.0


OIDC
OAuth 2.0



Standardizes scopes: openid, profile, email, and address

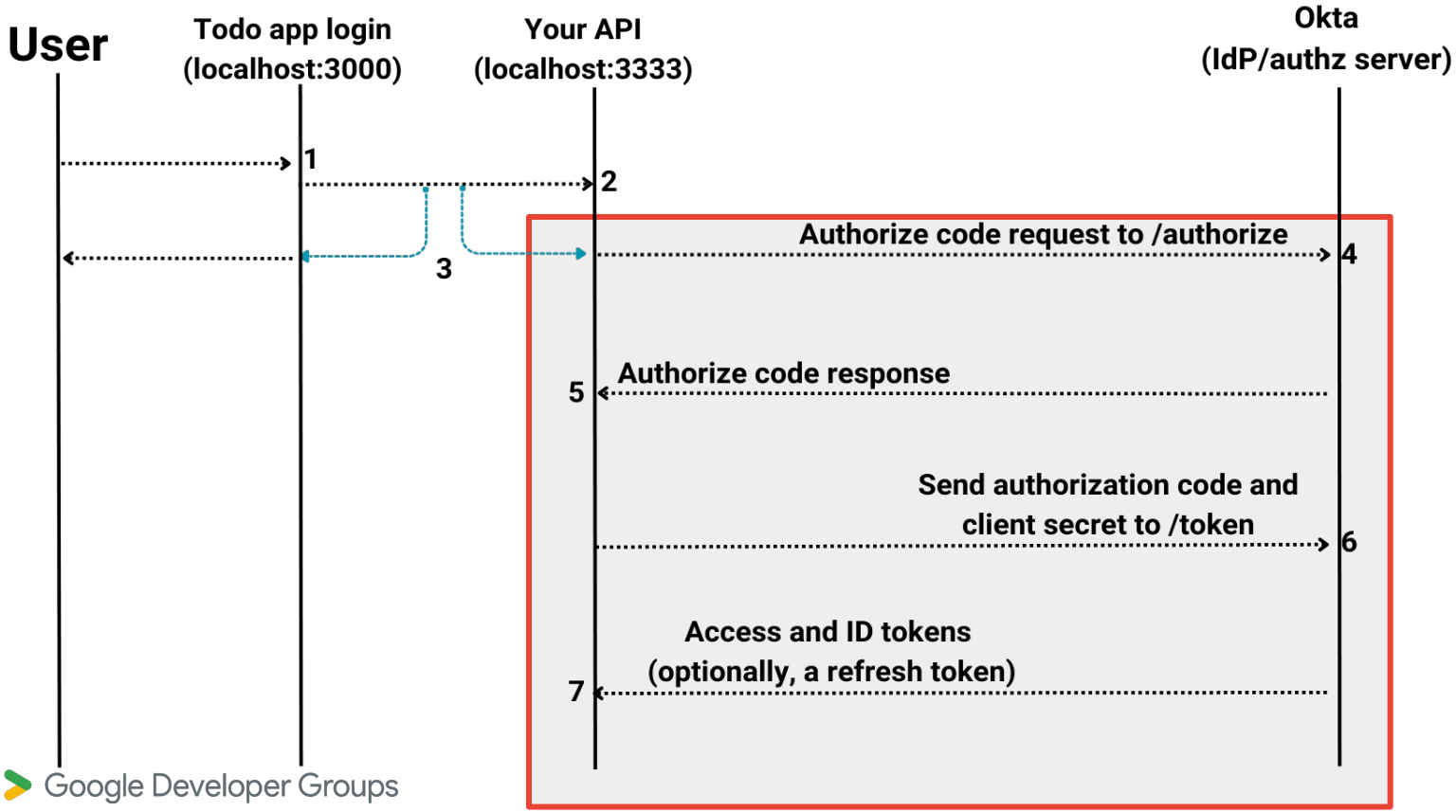
Standardizes claims for authn

User context stored in ID token to provide SSO to multiple apps

Comes with standardized endpoints i.e. /userinfo

Uses JWT (JSON Web Token) for the ID Token

IdP login via OIDC



Google Developer Groups

ID token

scopes

Authorize URI (required)

https://semonav2.oktapreview.com/oauth2/v1/authorize

Redirect URI (required)

https://oidcdebugger.com/debug

Client ID (required)

0oaqh3fceytYR7U1l0h7

Scope (required)

openid email profile

State

aacq5u8l5bu

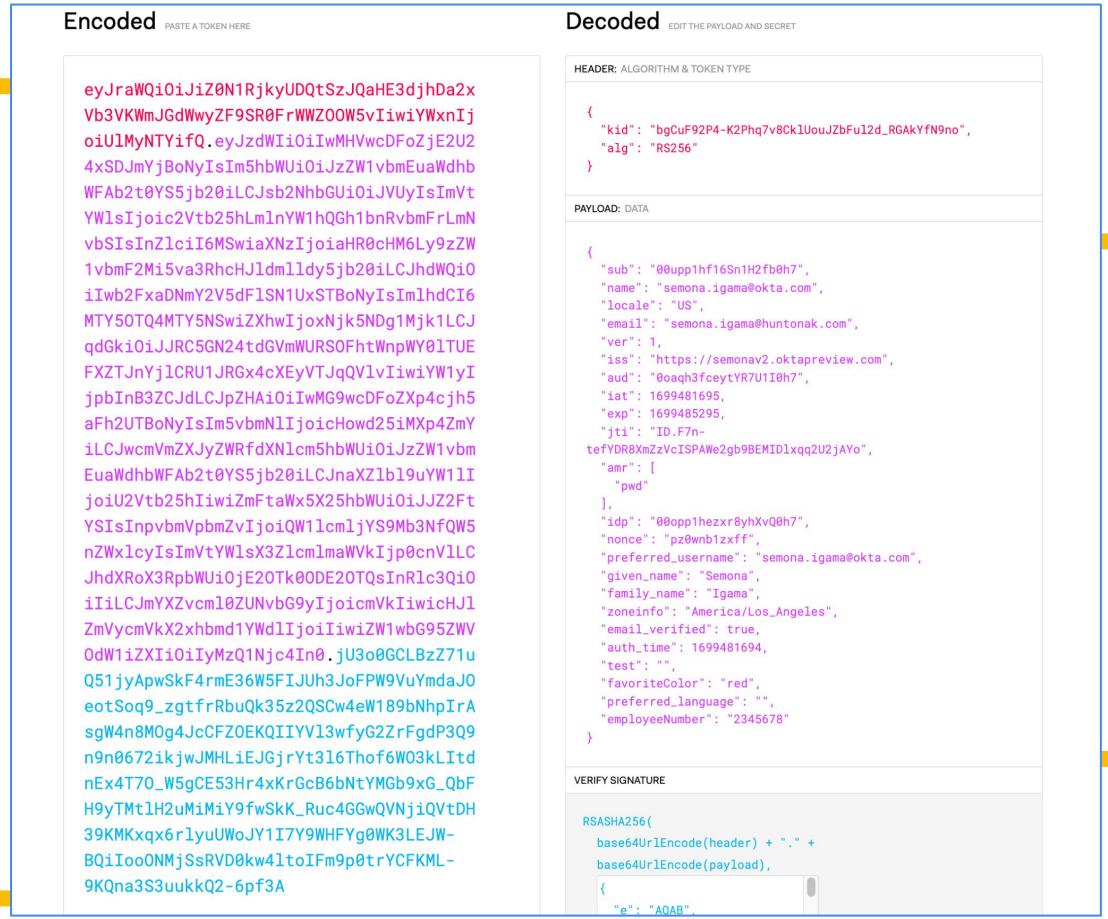
Nonce

hxnkydlebb

Response type (required)

☑ code ☐ token ☑ id_token

ID token
JWT



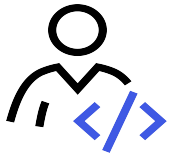
claims

Google Developer Groups

Workshop goal

Support OIDC login in the Todo App:

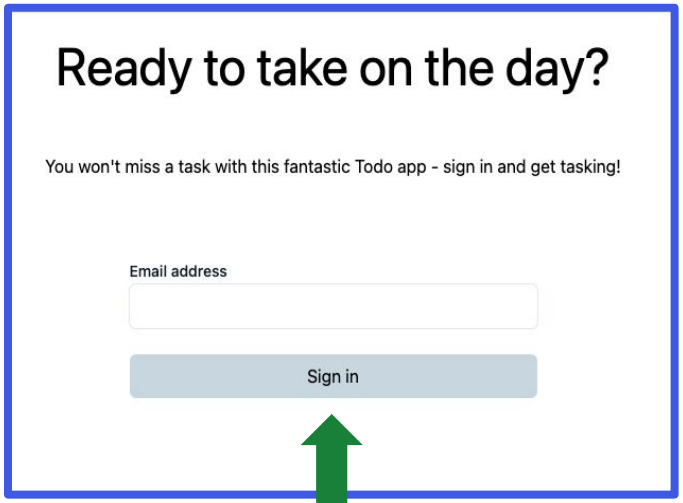
- Modify backend to enable OIDC flow
- Connect with an Identity Provider (IdP)
- Test user login with the configured IdP



SaaS developer

IdP (Okta) login using via OIDC

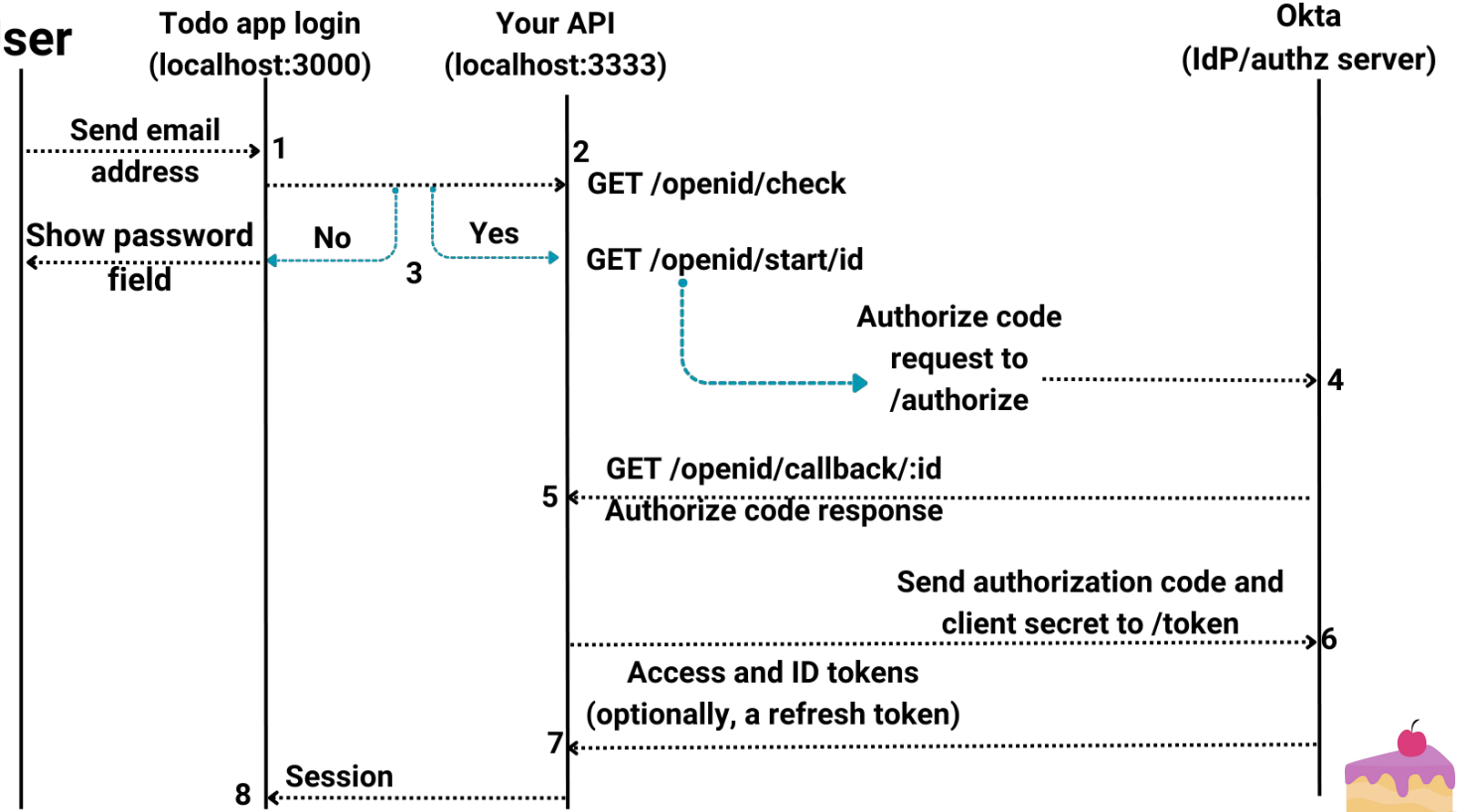
+



Enterprise customers'users

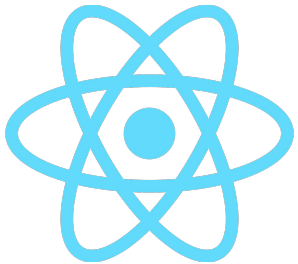
Google Developer Groups

How will we accomplish adding OIDC in our Todo App?



Walk through adding OIDC to our Todo sample application

Sample Todo App



Prisma

Express

prisma/schema.prisma

```
model User {
  id       Int    @id @default(autoincrement())
  email    String
  password String?
  name String
  Todo     Todo[]
  org       Org?    @relation(fields: [orgId], references: [id])
  orgId     Int?
  externalId String?
  @@unique([orgId, externalId])
}
```

prisma/schema.prisma

```
model Org {
  id        Int     @id @default(autoincrement())
  domain    String  @unique
  issuer                 String @default("")
  authorization_endpoint String @default("")
  token_endpoint         String @default("")
  userinfo_endpoint      String @default("")
  client_id              String @default("")
  client_secret          String @default("")
  apikey                 String
  Todo     Todo[]
  User     User[]
}
```

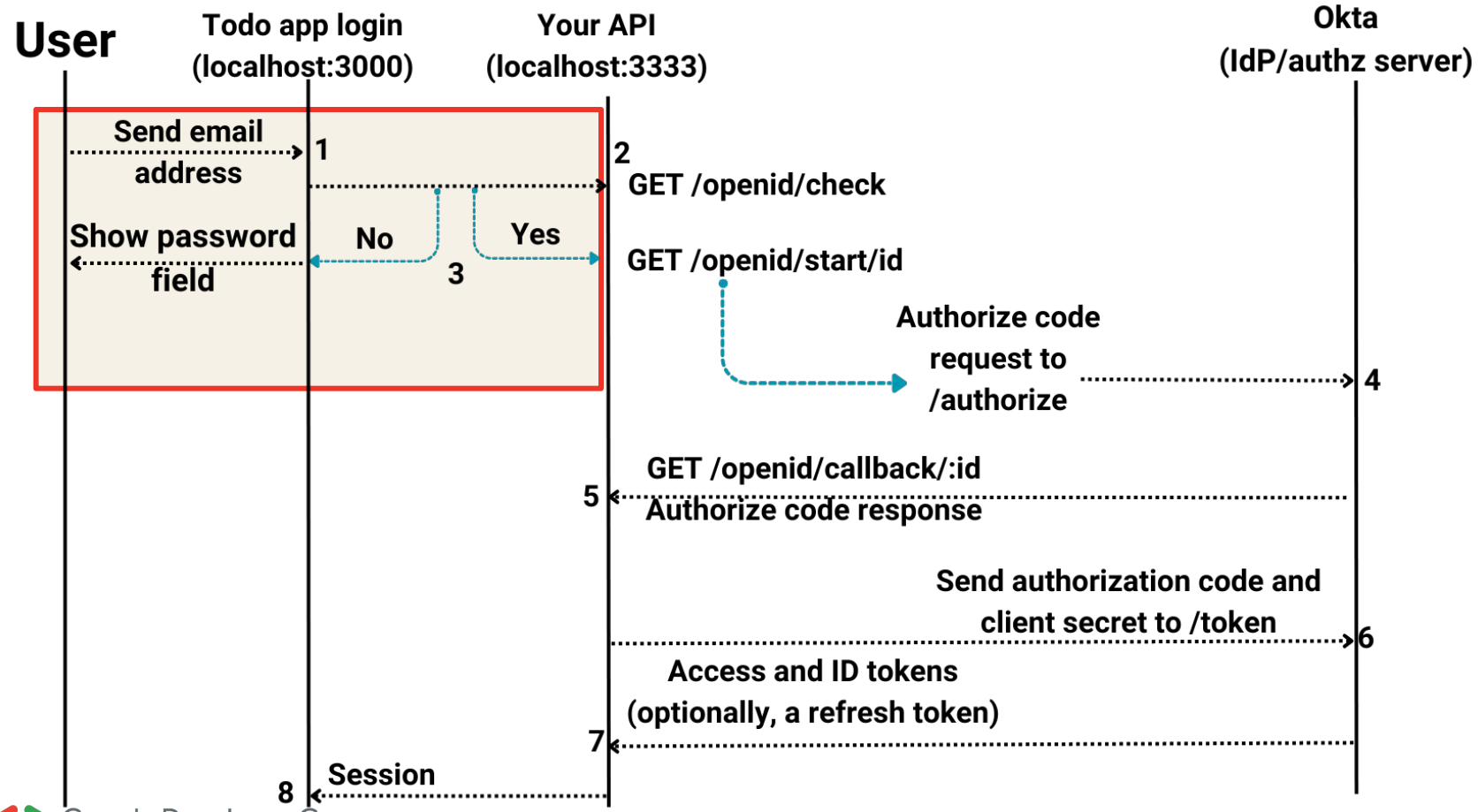
prisma/schema.prisma

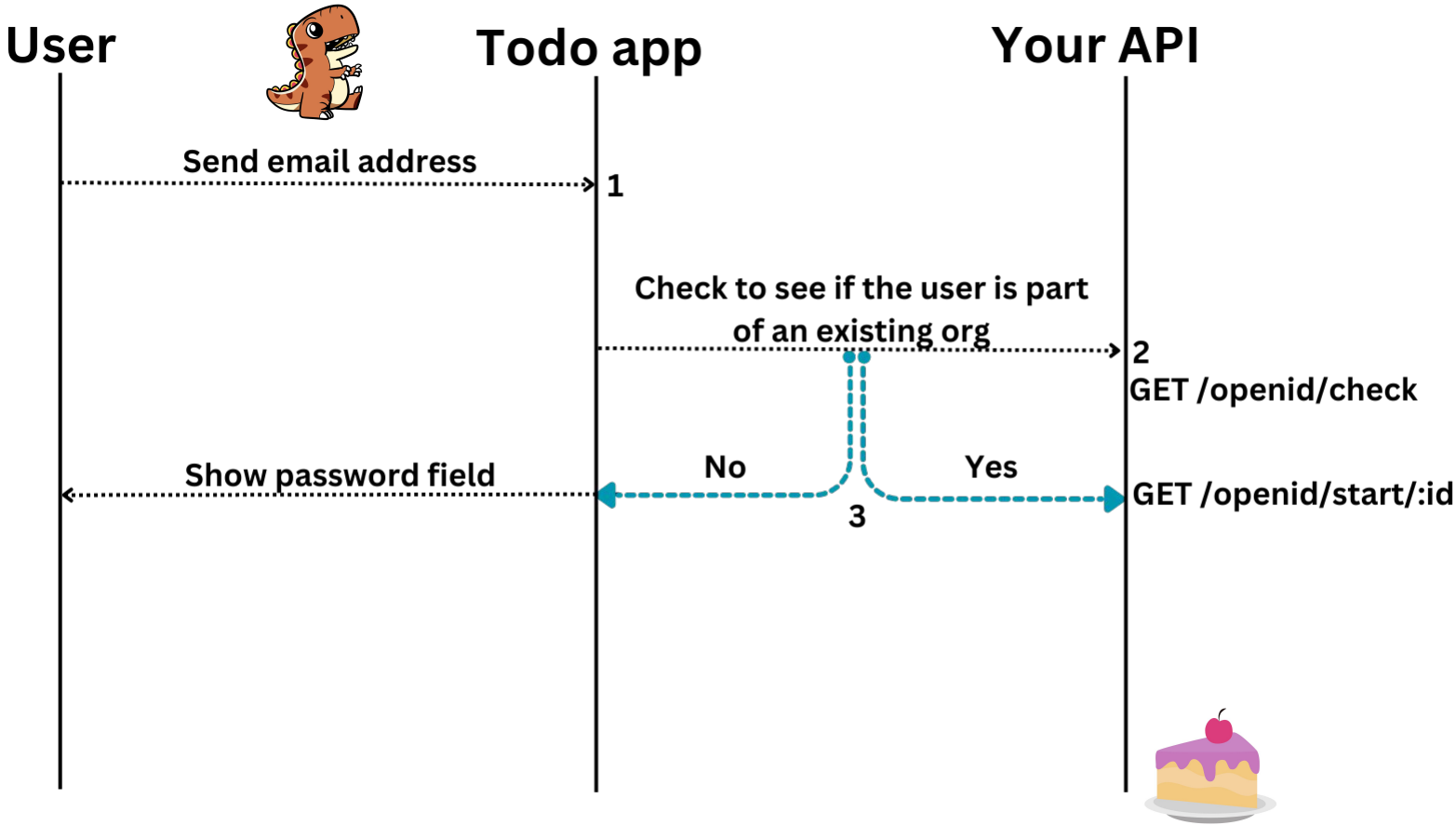
```
model Todo {
  id        Int     @id @default(autoincrement())
  task      String
  completed Boolean @default(false)
  createdAt DateTime @default(now())
  completedAt DateTime?
  user      User?   @relation(fields: [userId], references: [id])
  userId    Int
  org       Org?    @relation(fields: [orgId], references: [id])
  orgId     Int?
}
```

apps/api/src/main.ts

```
////////// OIDC helper functions
async function orgFromId(id) {…
}

function getDomainFromEmail(email) {…
}

app.post('/api/openid/check', async (req, res, next) => {…
});

function createStrategy(org) {
  return new OpenIDConnectStrategy({…
  },
  async function verify(issuer, profile, cb) {…
  })
}

// The frontend then redirects here to have the backend start the OIDC flow.…
// to avoid revealing how many enterprise customers you have.)
app.get('/openid/start/:id', async (req, res, next) => {…
});

app.get('/openid/callback/:id', async (req, res, next) => {…
});
```

Frontend





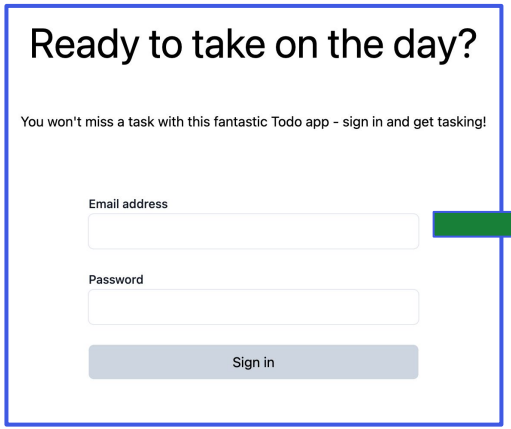
Google Developer Groups

Frontend modifications already done:

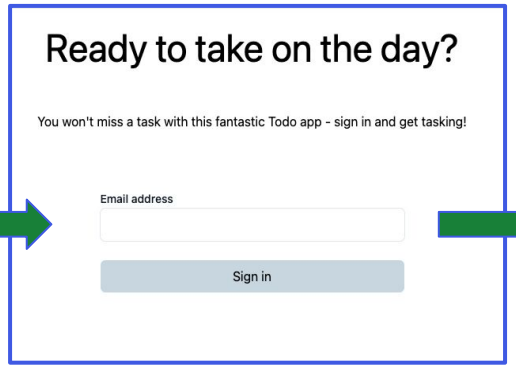
1. Hide the password field
2. Submit the email address to server
3. Confirm the user's organization

Review completed code - https://bit.ly/3MELxHW

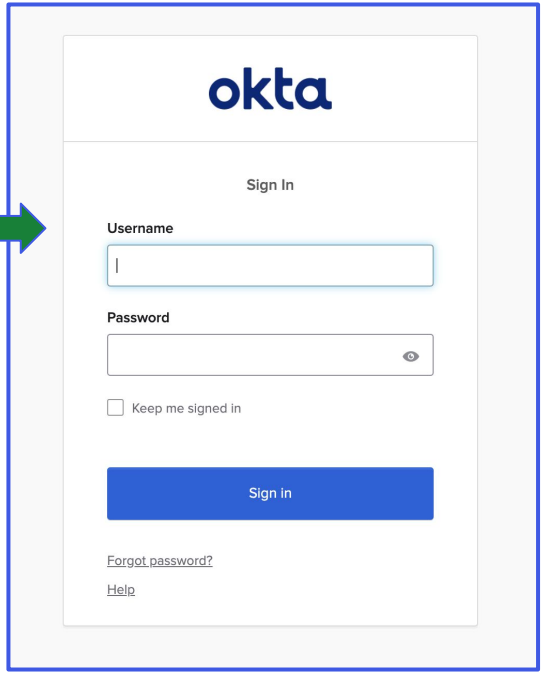
Before



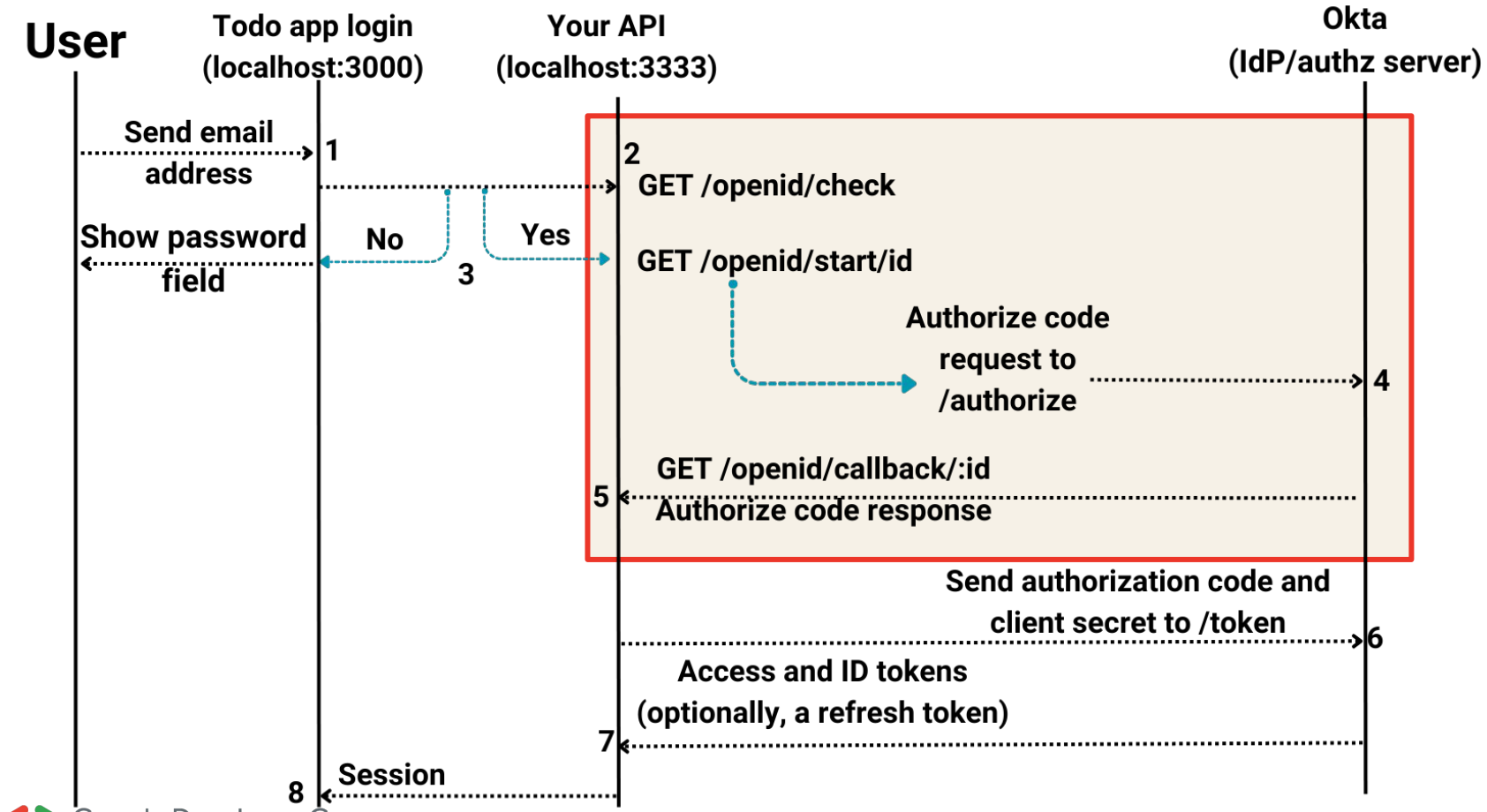
After

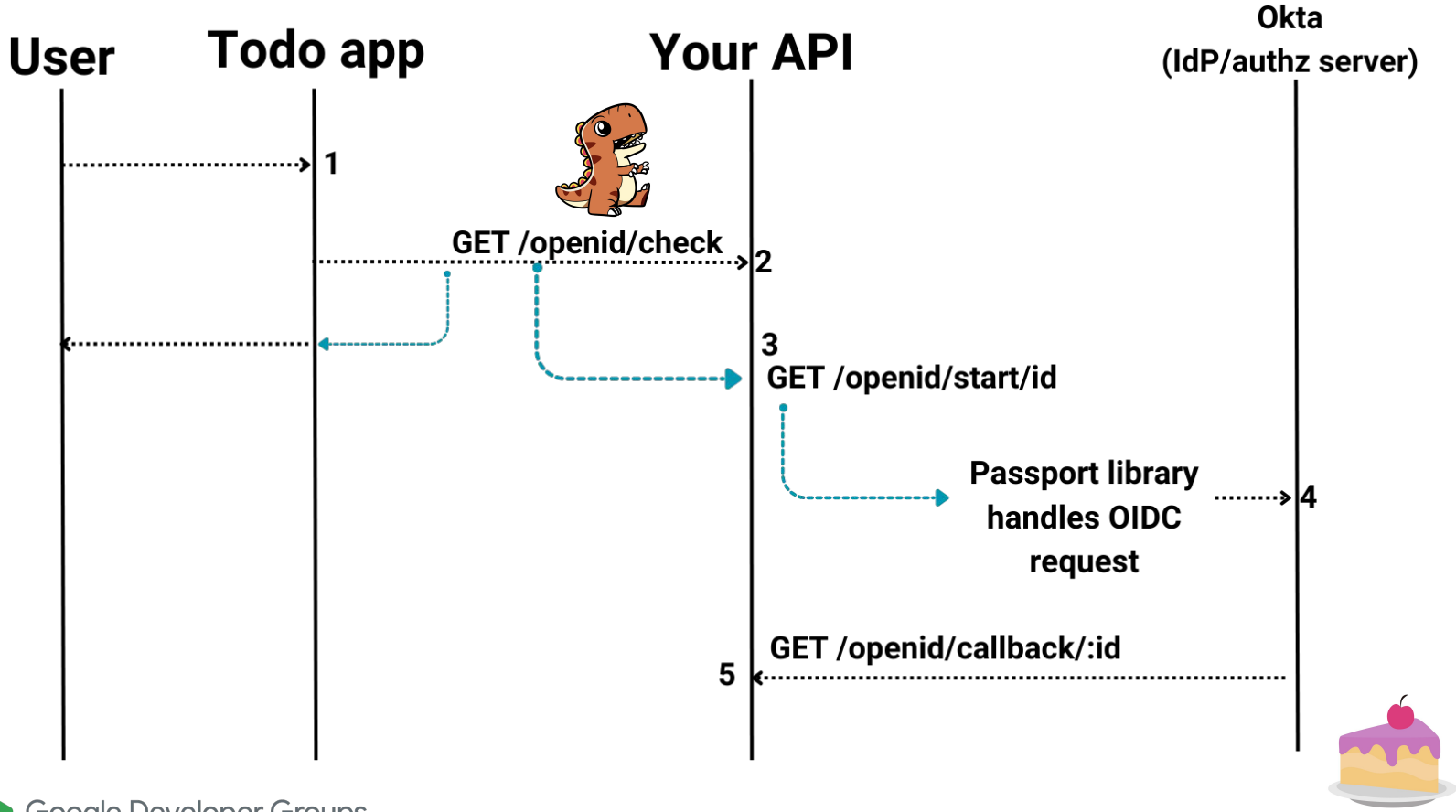


Redirect to IdP



Backend





Google Developer Groups

Update the Express API to support OIDC

Steps:

1. In your terminal, Install the Passport OIDC Library

➔ `npm install --save passport-openidconnect`

2. In `apps/api/src/main.ts` add the helper functions `orgFromId` and `getDomainFromEmail`

```
////////// OIDC helper functions
async function orgFromId(id) {
  const org = await prisma.org.findFirst({
    where: {
      id: parseInt(id)
    }
  })
  return org
}

function getDomainFromEmail(email) {
  let domain;
  try {
    domain = email.split('@')[1];
  } catch(e) {
    return null;
  }
  return domain;
}
```

completed code - https://bit.ly/3uaKZTH

Add route to check the OpenID org

Steps:

1. In apps/api/src/main.ts create the /api/openid/check endpoint which will use getDomainFromEmail to return the numeric ID of the org that the user's email domain belongs to or otherwise return null

```
app.post('/api/openid/check', async (req, res, next) => {
  const { username } = req.body;

  const domain = getDomainFromEmail(username);
  if(domain) {
    var org = await prisma.org.findFirst({
      where: {
        domain: domain
      }
    });
    if(!org) {
      org = await prisma.org.findFirst({
        where: {
          User: {
            some: {
              email: username
            }
          }
        }
      })
    }
    if(org && org.issuer) {
      return res.json({ org_id: org.id });
    }
  }

  res.json({ org_id: null });
});
```

completed code - https://bit.ly/3QBwcJj

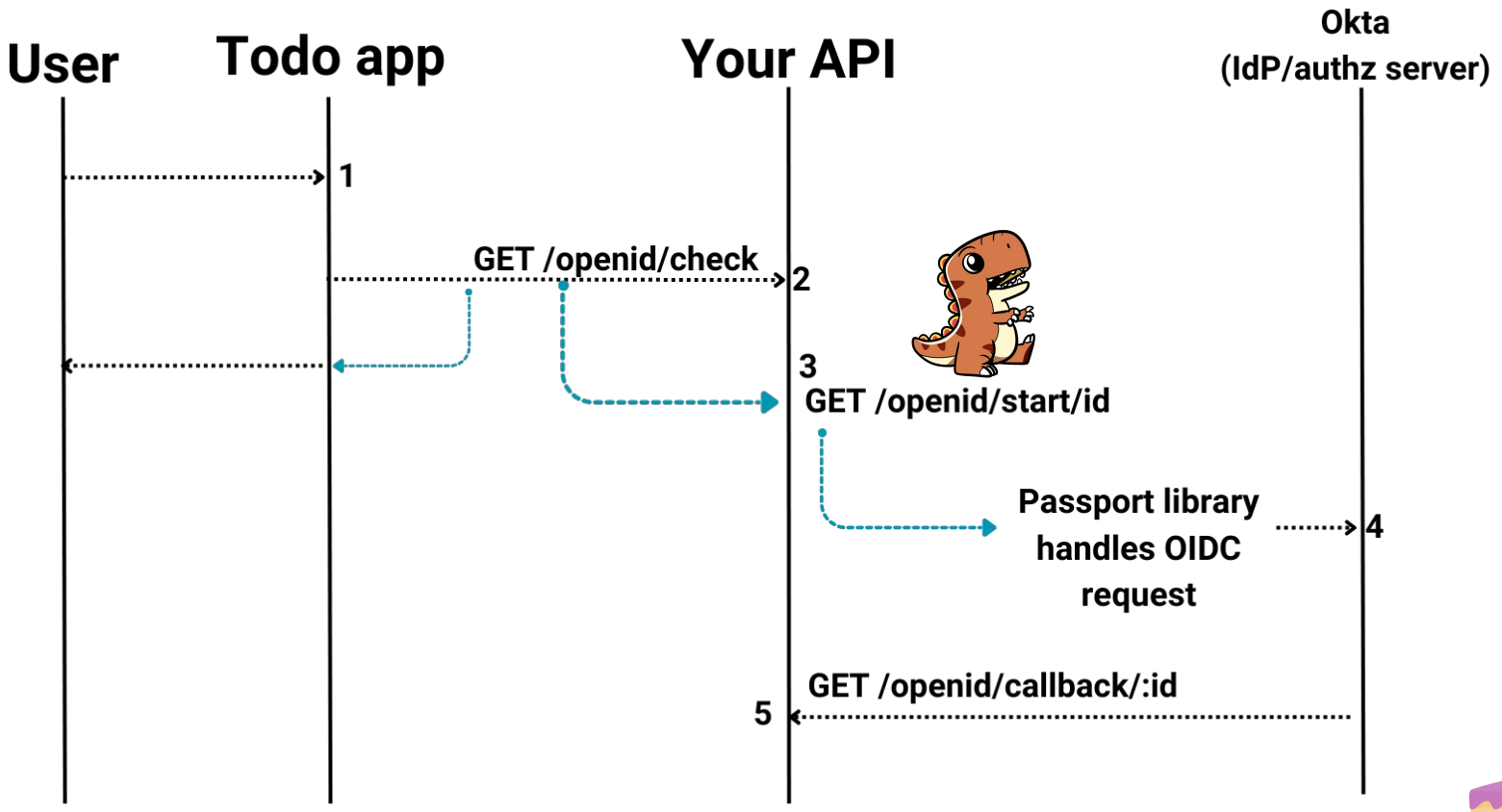
Enable the OIDC flow

Steps:

1. The frontend will redirect the browser to the backend's /openid/start/${org_id} endpoint to initiate the OIDC login flow when a user belongs to an existing org

```
// The frontend then redirects here to have the backend start the OIDC flow.
// (You should probably use random IDs, not auto-increment integers
// to avoid revealing how many enterprise customers you have.)
app.get('/openid/start/:id', async (req, res, next) => {

  const org = await orgFromId(req.params.id);
  if(!org) {
    return res.sendStatus(404);
  }

  const strategy = createStrategy(org);
  if(!strategy) {
    return res.sendStatus(404);
  }

  passport.authenticate(strategy)(req, res, next);

});
```

completed code -https://bit.ly/3SFL5x5



Google Developer Groups

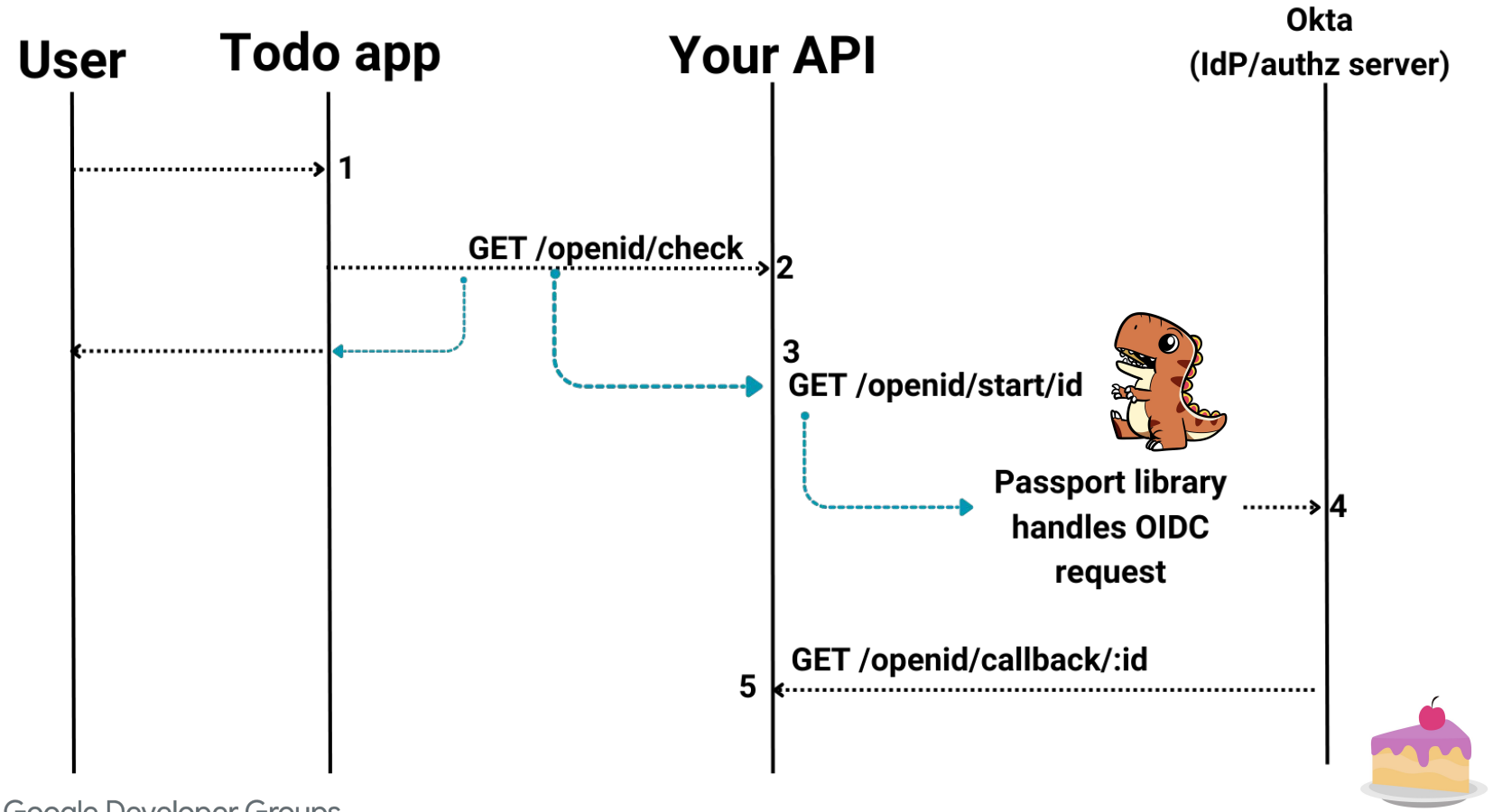
Use the Passport OIDC library

Steps:

1. Add the import for the OIDC passport after the import for passport-local at the top of apps/api/src/main.ts
   ➔ `import passportOIDC from 'passport-openidconnect';`
2. Define an variable for the OIDC strategy so you can utilize it within the API
   ➔ `const OpenIDConnectStrategy = passportOIDC.Strategy;`
3. Create a strategy for each org

```
import express from 'express';
import { PrismaClient, Todo, User } from '@prisma/client';
import passportLocal from 'passport-local';
import passport from 'passport';
import passportOIDC from 'passport-openidconnect';
import session from 'express-session';

interface IUser {
  id: number;
}

const prisma = new PrismaClient();
const LocalStrategy = passportLocal.Strategy;
const OpenIDConnectStrategy = passportOIDC.Strategy;
```

completed code - https://bit.ly/3MKc4no



Google Developer Groups

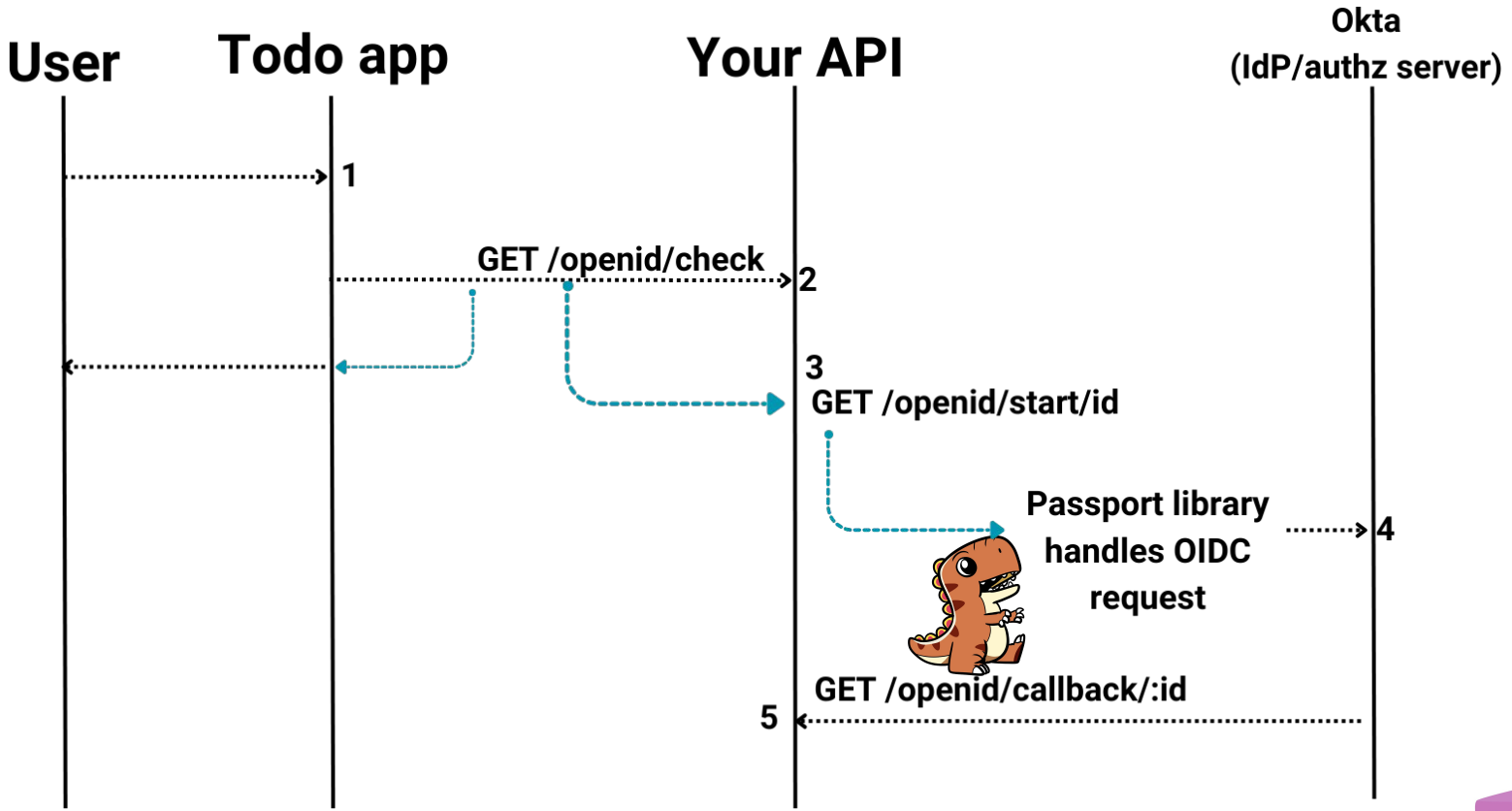
Receive callback after authentication

Steps:

1. Build a callback route to the end of apps/api/src/main.ts in order to confirm that the ID of the request corresponds to an existing org in the database

```
app.get('/openid/callback/:id', async (req, res, next) => {

  const org = await orgFromId(req.params.id);
  if(!org) {
    return res.sendStatus(404);
  }

  const strategy = createStrategy(org);
  if(!strategy) {
    return res.sendStatus(404);
  }

  passport.authenticate(strategy, {
    successRedirect: 'http://localhost:3000/'
  })(req, res, next);

});
```

completed code - https://bit.ly/3QZuRO7



Google Developer Groups

Loosen the cookie policy for the OIDC redirect authentication flow

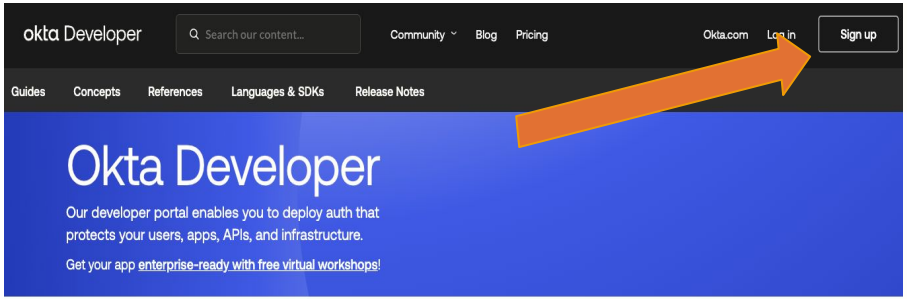
Steps:

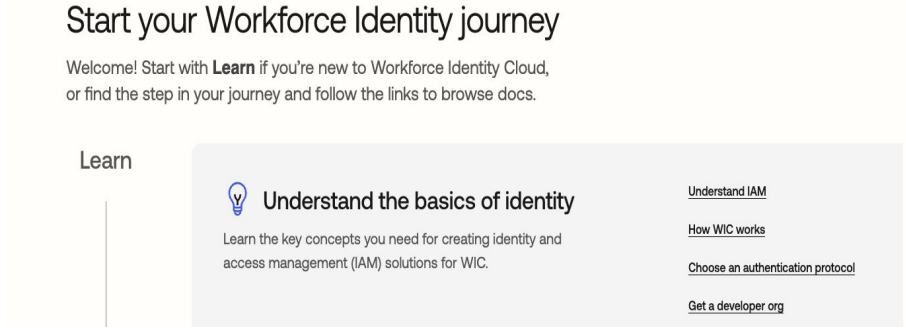
1. In apps/api/src/main.ts, change the session cookie policy from sameSite: 'strict' to sameSite: 'lax'

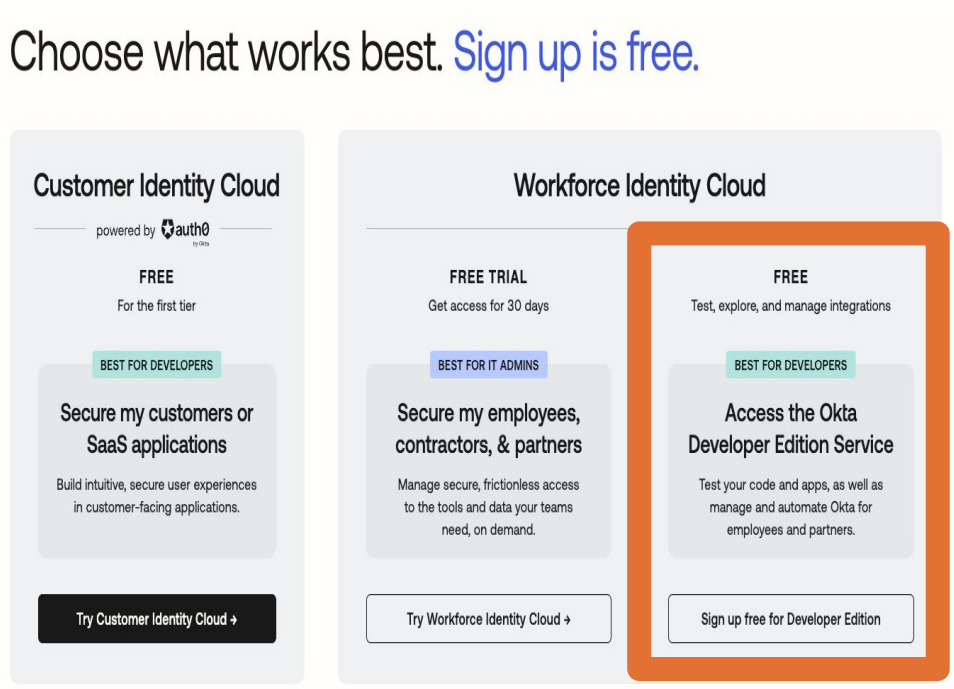
```
const prisma = new PrismaClient();
const LocalStrategy = passportLocal.Strategy;

const app = express();
app.use(express.json())
app.use(session({
  resave: false,
  saveUninitialized: false,
  secret: 'top secret',
  cookie: {
    http: false,
    sameSite: 'lax'
  }
}));
```

Connect to the Identity Provider

developer.okta.com

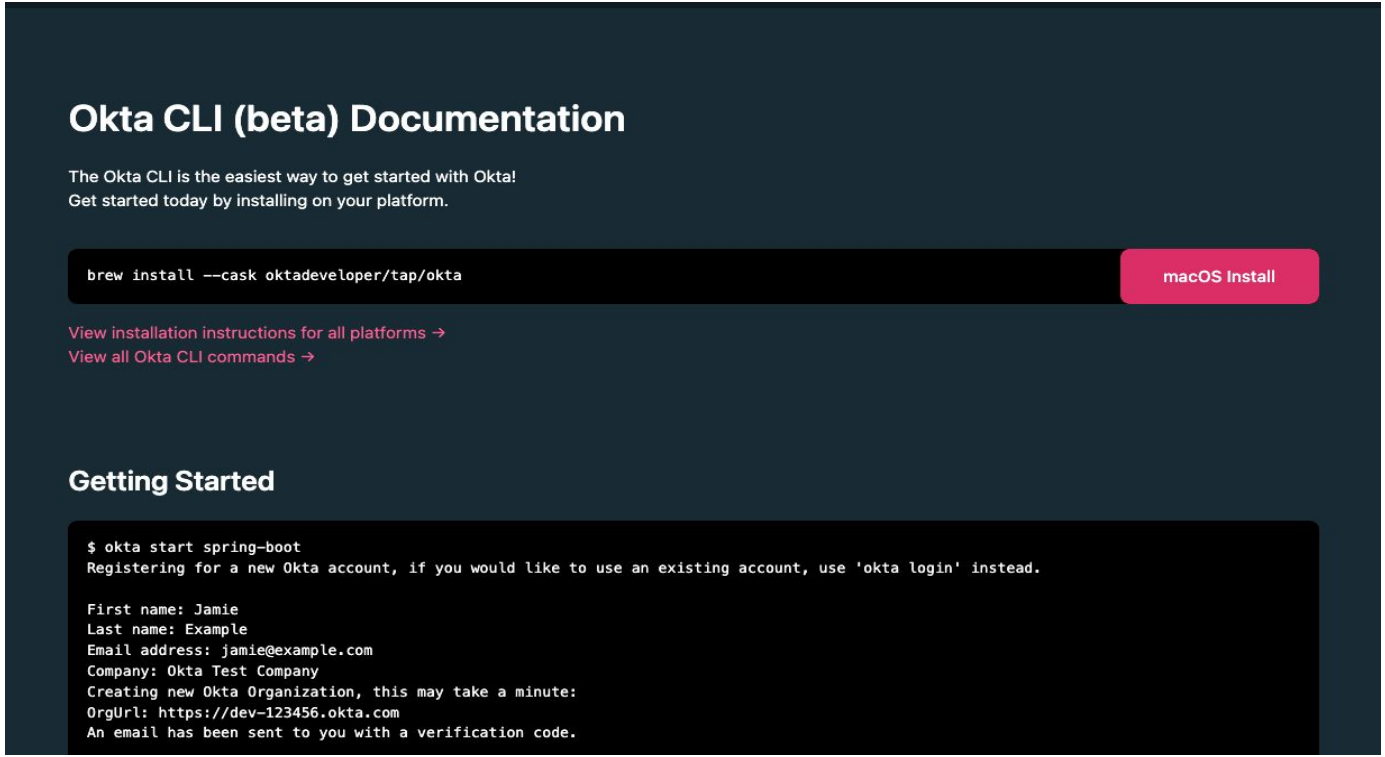




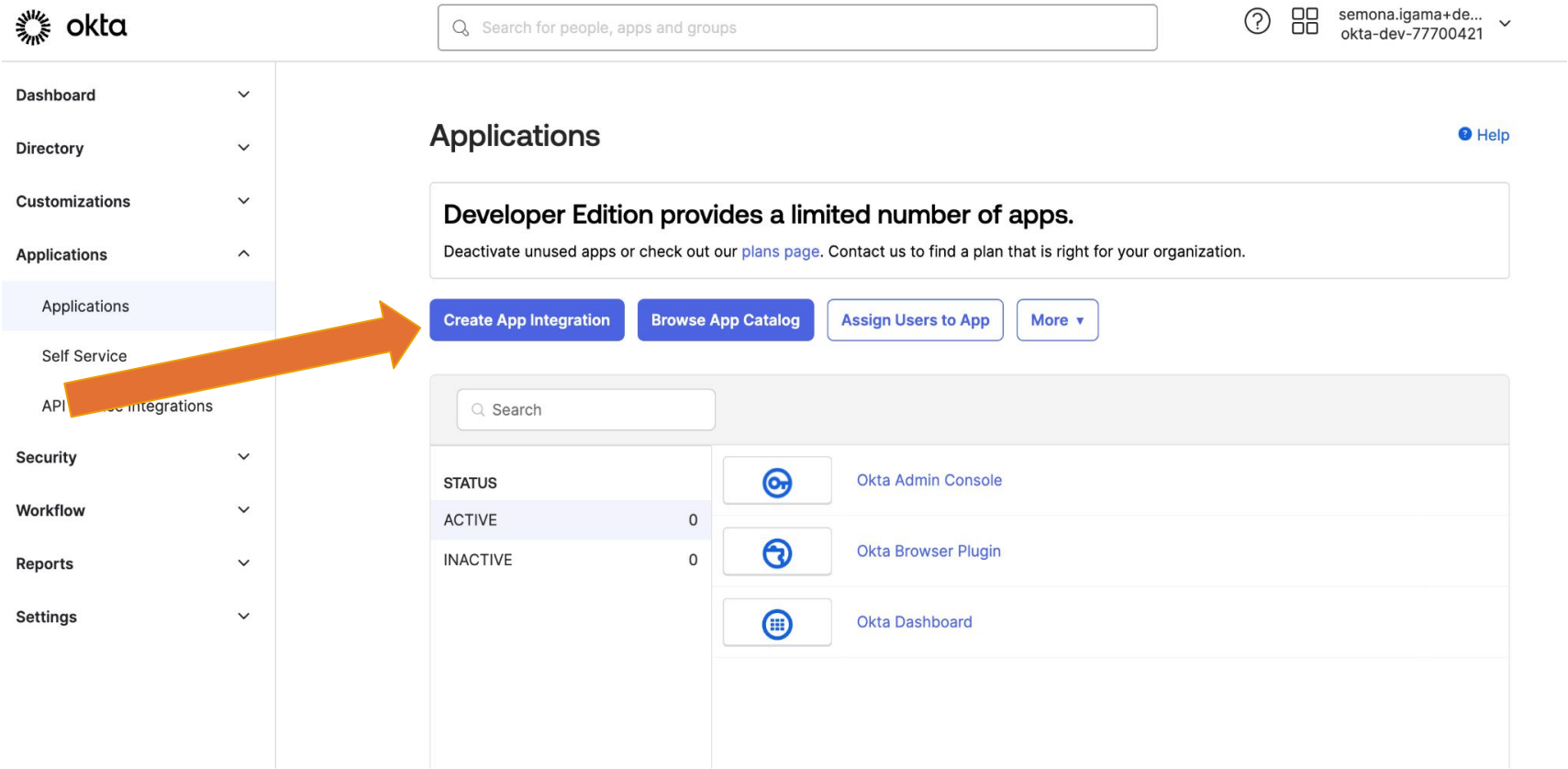


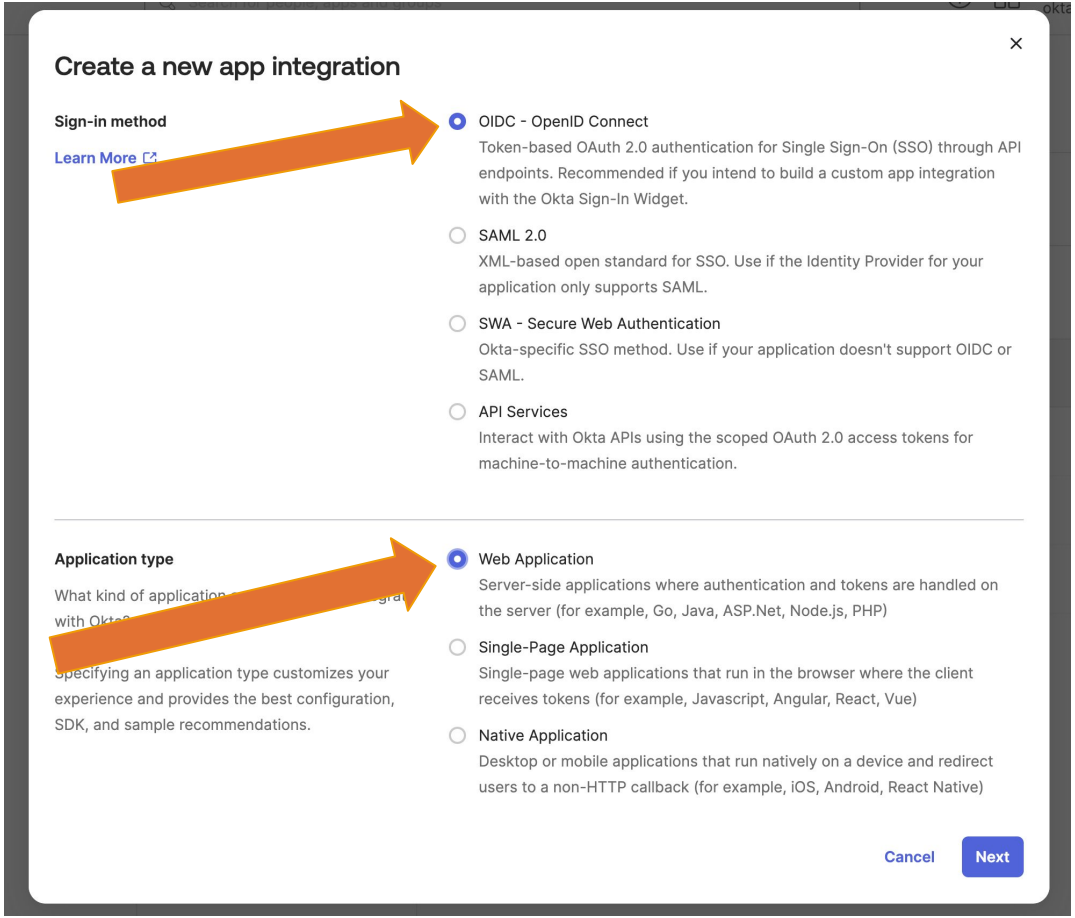
Google Developer Groups

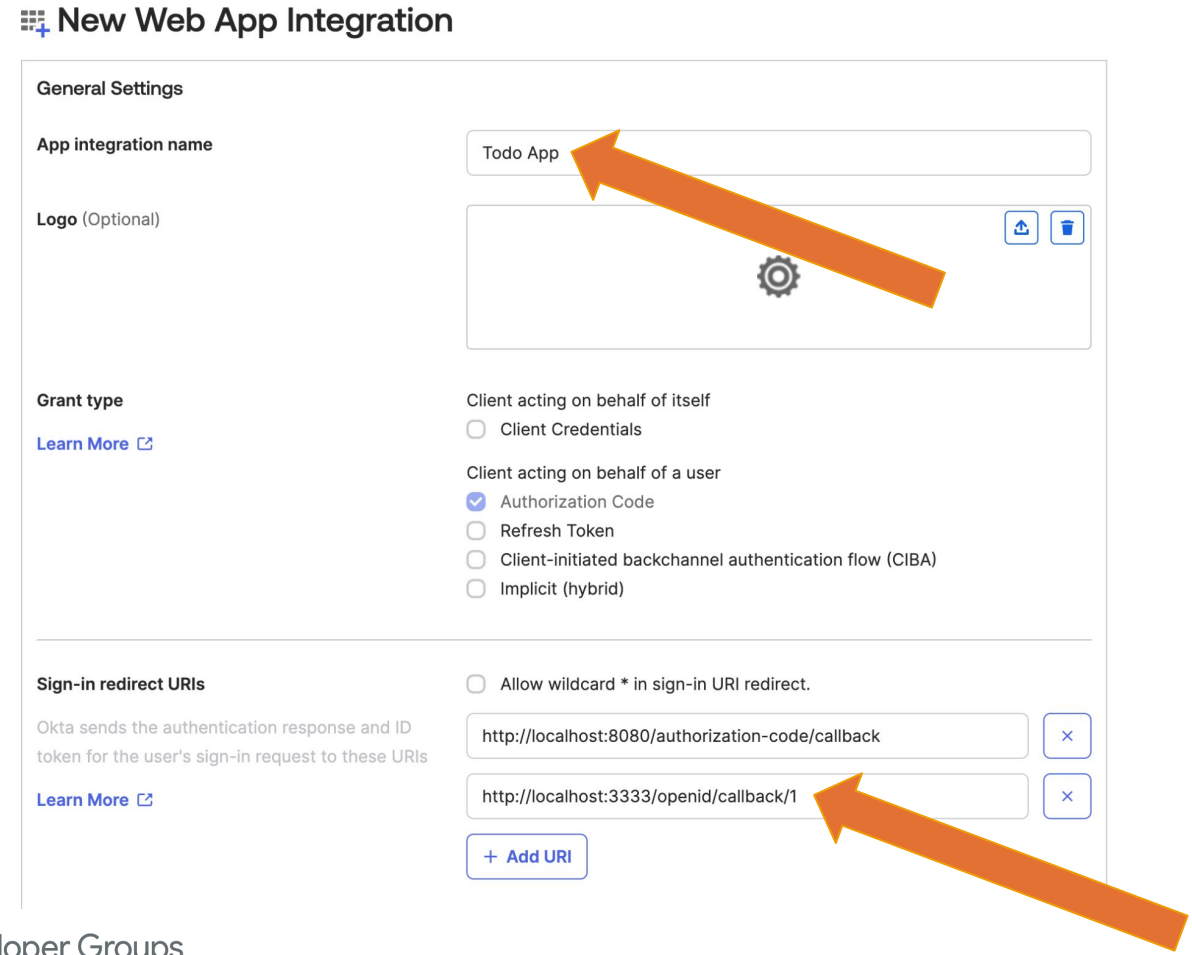
Okta CLI - cli.okta.com



Create an OIDC Application in Okta


okta
Search for people, apps and groups
semona.igama+de...
okta-dev-77700421
Dashboard
Directory
Customizations
Applications
Applications
Self Service
API
Integrations
Security
Workflow
Reports
Settings
Applications
Help
Developer Edition provides a limited number of apps.
Deactivate unused apps or check out our plans page. Contact us to find a plan that is right for your organization.
Create App Integration
Browse App Catalog
Assign Users to App
More
Search
STATUS
ACTIVE 0
INACTIVE 0
Okta Admin Console
Okta Browser Plugin
Okta Dashboard


Create a new app integration
Sign-in method
Learn More
OIDC - OpenID Connect
Token-based OAuth 2.0 authentication for Single Sign-On (SSO) through API endpoints. Recommended if you intend to build a custom app integration with the Okta Sign-In Widget.
SAML 2.0
XML-based open standard for SSO. Use if the Identity Provider for your application only supports SAML.
SWA - Secure Web Authentication
Okta-specific SSO method. Use if your application doesn't support OIDC or SAML.
API Services
Interact with Okta APIs using the scoped OAuth 2.0 access tokens for machine-to-machine authentication.
Application type
What kind of application
with Okta
Specifying an application type customizes your experience and provides the best configuration, SDK, and sample recommendations.
Web Application
Server-side applications where authentication and tokens are handled on the server (for example, Go, Java, ASP.Net, Node.js, PHP)
Single-Page Application
Single-page web applications that run in the browser where the client receives tokens (for example, Javascript, Angular, React, Vue)
Native Application
Desktop or mobile applications that run natively on a device and redirect users to a non-HTTP callback (for example, iOS, Android, React Native)
Cancel
Next


New Web App Integration
General Settings
App integration name
Todo App
Logo (Optional)
Grant type
Learn More
Client acting on behalf of itself
Client Credentials
Client acting on behalf of a user
Authorization Code
Refresh Token
Client-initiated backchannel authentication flow (CIBA)
Implicit (hybrid)
Sign-in redirect URIs
Allow wildcard * in sign-in URI redirect.
Okta sends the authentication response and ID token for the user's sign-in request to these URIs
Learn More
http://localhost:8080/authorization-code/callback
http://localhost:3333/openid/callback/1
+ Add URI

Assignments

Controlled access

Select whether to assign the [illegible] to everyone in your org, [illegible] selected group(s), or skip assignment until after app creation.

- (•) Allow everyone in your organization to access
- ( ) Limit access to selected groups
- ( ) Skip group assignment for now

Enable immediate access (Recommended)

Recommended if you want to grant access to everyone without pre-assigning your app to users and use Okta only for authentication.

- [x] Enable immediate access with **Federation Broker Mode**

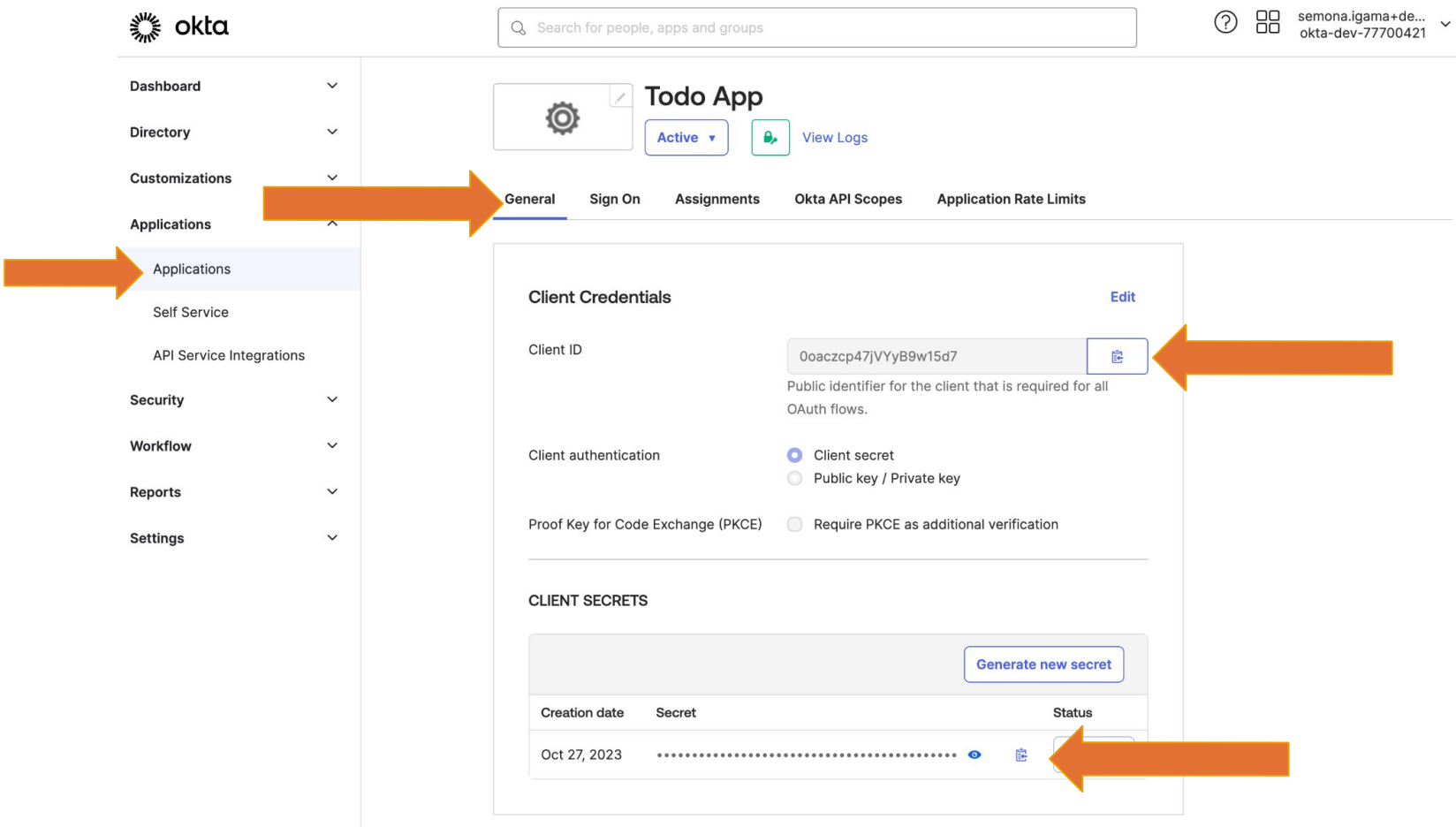
To ensure optimal app performance at scale, Okta End User Dashboard and provisioning features are disabled. Learn more about Federation Broker Mode.

Save Cancel

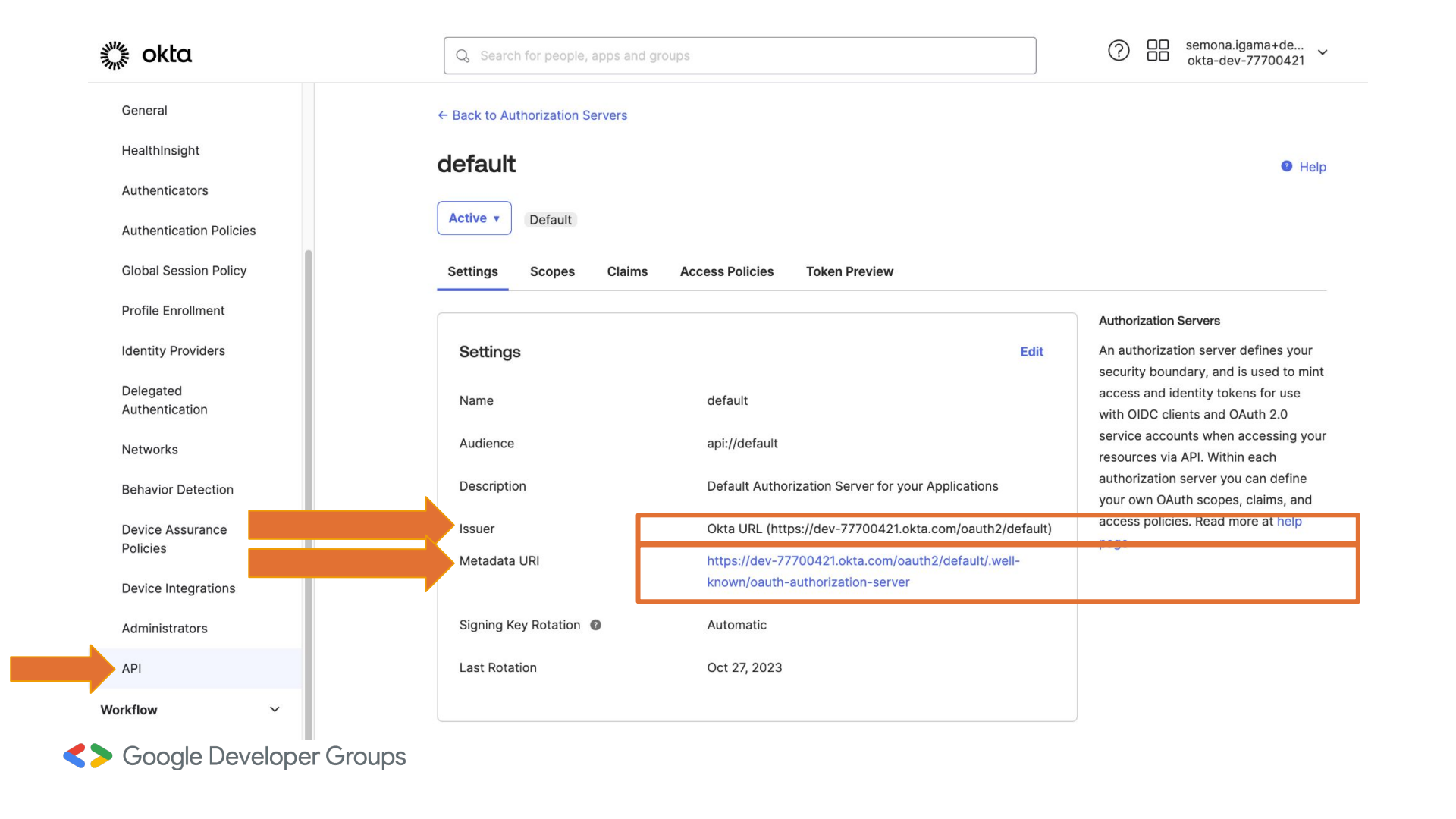
Add org authorization server configuration to the database

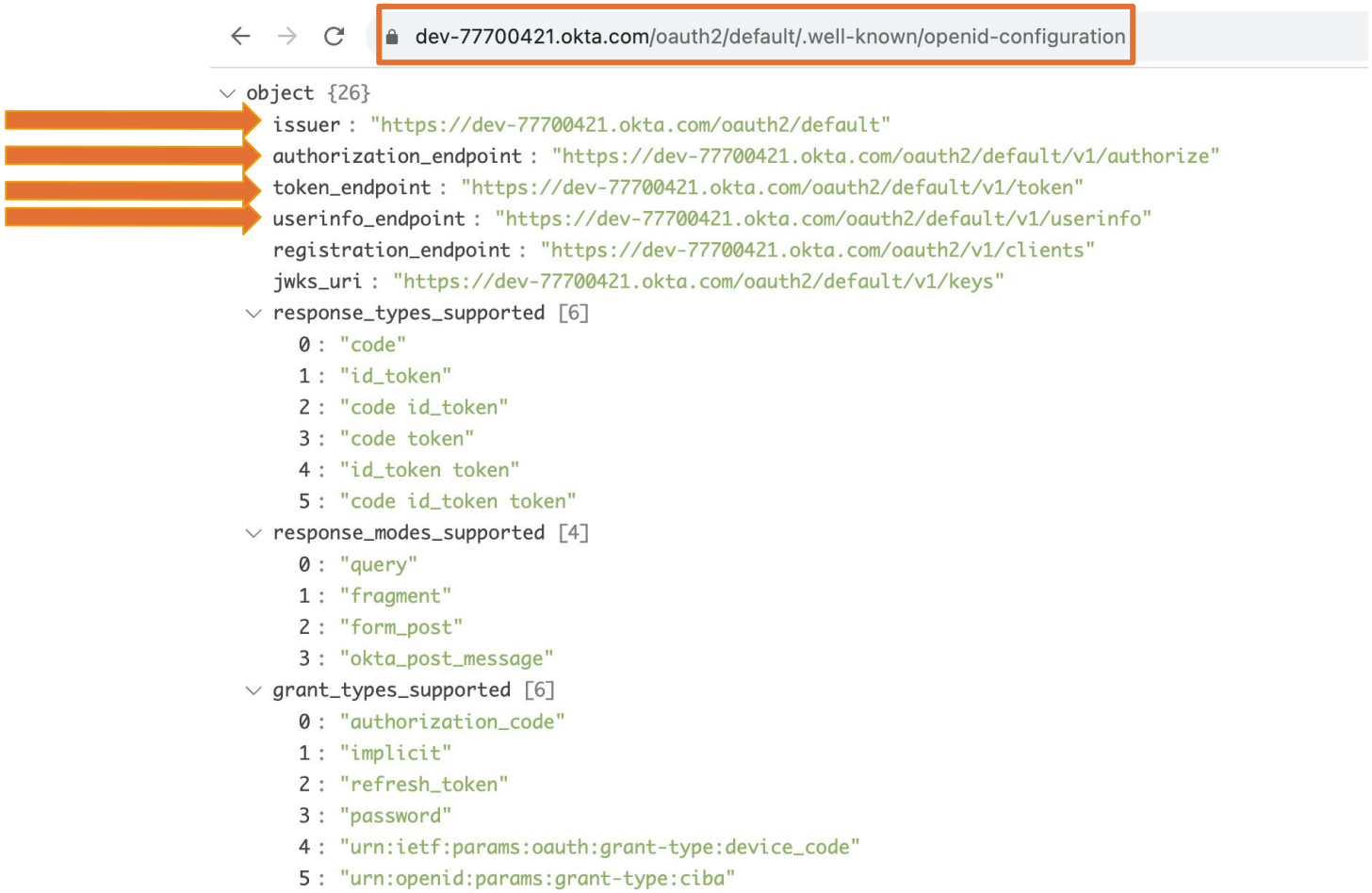
Steps:

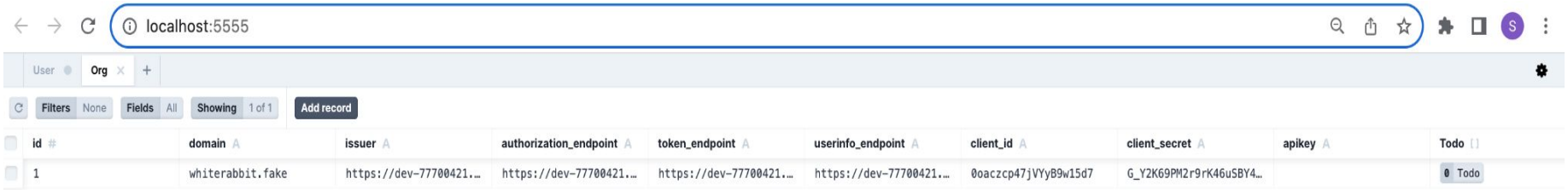
1. Edit the database and add a new org using Prisma Studio
   ➔ `npx prisma studio`
2. Enter the domain name of this organization i.e. `whiterabbit.fake`
3. Fill out the `client_id` and `client_secret` for the org with ID 1, using the values from Okta
4. In Okta navigate to the Security tab > API and locate the following:
   - Issuer info
   - Metadata URL - replace `oauth-authorization-server` in the metadata URL with `openid-configuration`
5. Save changes!!



Google Developer Groups



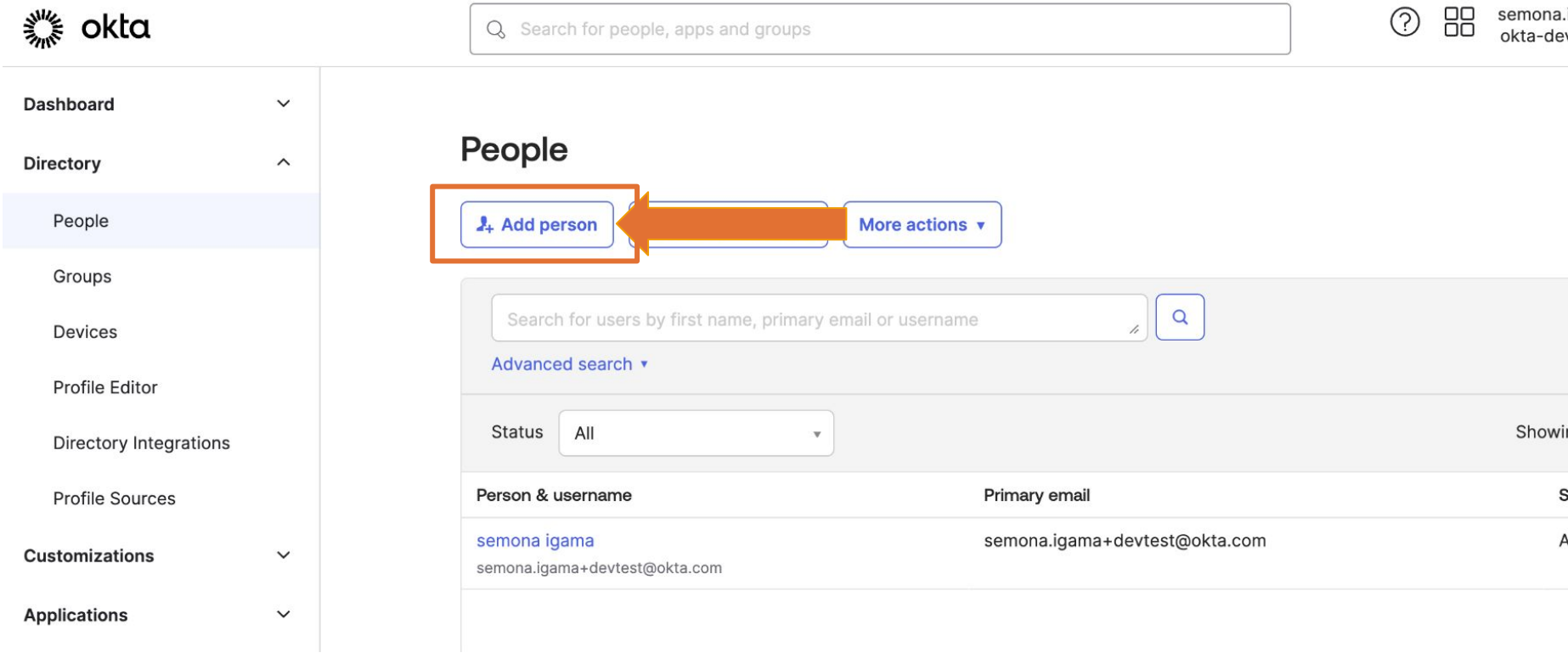

dev-77700421.okta.com/oauth2/default/.well-known/openid-configuration
object {26}
issuer : "https://dev-77700421.okta.com/oauth2/default"
authorization_endpoint : "https://dev-77700421.okta.com/oauth2/default/v1/authorize"
token_endpoint : "https://dev-77700421.okta.com/oauth2/default/v1/token"
userinfo_endpoint : "https://dev-77700421.okta.com/oauth2/default/v1/userinfo"
registration_endpoint : "https://dev-77700421.okta.com/oauth2/v1/clients"
jwks_uri : "https://dev-77700421.okta.com/oauth2/default/v1/keys"
response_types_supported [6]
0 : "code"
1 : "id_token"
2 : "code id_token"
3 : "code token"
4 : "id_token token"
5 : "code id_token token"
response_modes_supported [4]
0 : "query"
1 : "fragment"
2 : "form_post"
3 : "okta_post_message"
grant_types_supported [6]
0 : "authorization_code"
1 : "implicit"
2 : "refresh_token"
3 : "password"
4 : "urn:ietf:params:oauth:grant-type:device_code"
5 : "urn:openid:params:grant-type:ciba"


localhost:5555
User
Org
Filters None
Fields All
Showing 1 of 1
Add record
id #
domain A
issuer A
authorization_endpoint A
token_endpoint A
userinfo_endpoint A
client_id A
client_secret A
apikey A
Todo []
1
whiterabbit.fake
https://dev-77700421.…
https://dev-77700421.…
https://dev-77700421.…
https://dev-77700421.…
0oaczcp47jVYyB9w15d7
G_Y2K69PM2r9rK46uSBY4…
0 Todo

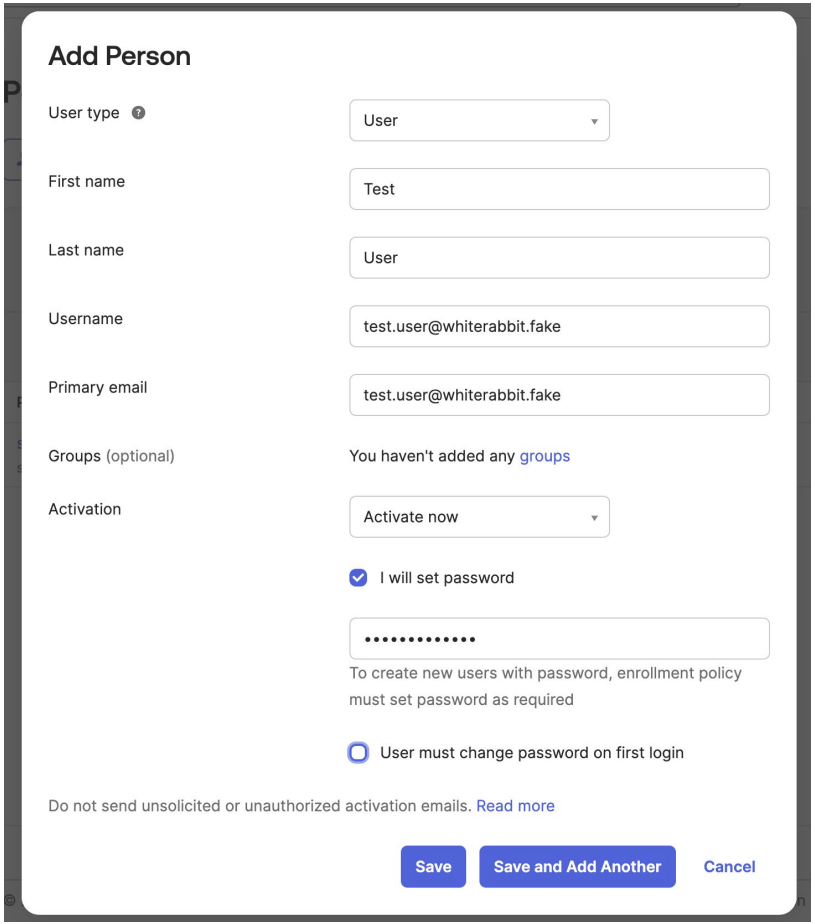
Use OIDC authentication in the application

Steps:

1. Create some user accounts in your Okta Admin Console, and try logging into the Todo app as those users! Use Prisma (`npx prisma studio`) to see how each user's database record is created the first time they log in.



Google Developer Groups


Add Person
User type
User
First name
Test
Last name
User
Username
test.user@whiterabbit.fake
Primary email
test.user@whiterabbit.fake
Groups (optional)
You haven't added any groups
Activation
Activate now
I will set password
••••••••••••
To create new users with password, enrollment policy must set password as required
User must change password on first login
Do not send unsolicited or unauthorized activation emails. Read more
Save
Save and Add Another
Cancel

Ready to take on the day?

You won't miss a task with this fantastic Todo app - sign in and get tasking!

Email address

test.user@whiterabbit.fake

Sign in

Connecting to

Sign in with your account to access Todo App

okta

Sign In

Username

test.user@whiterabbit.fake

Password

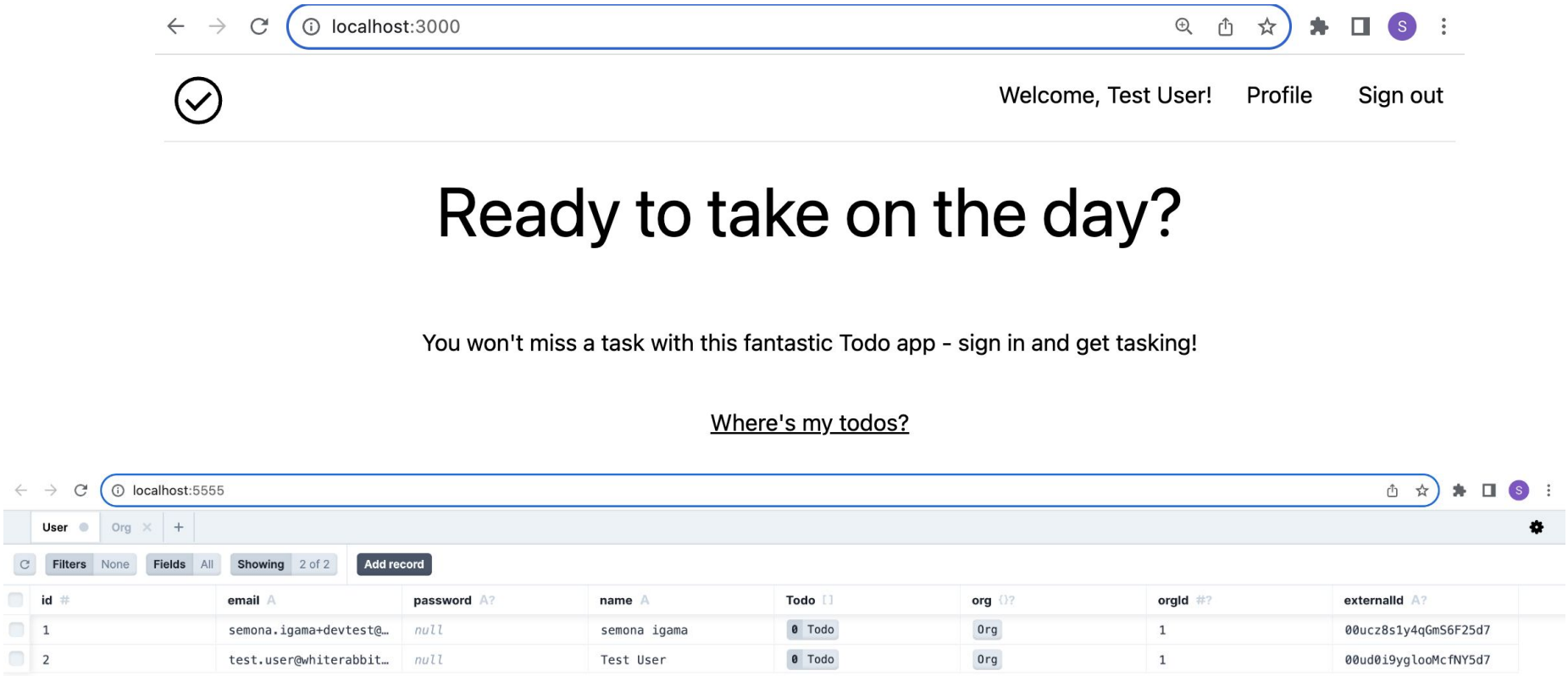
Keep me signed in

Sign in

Forgot password?

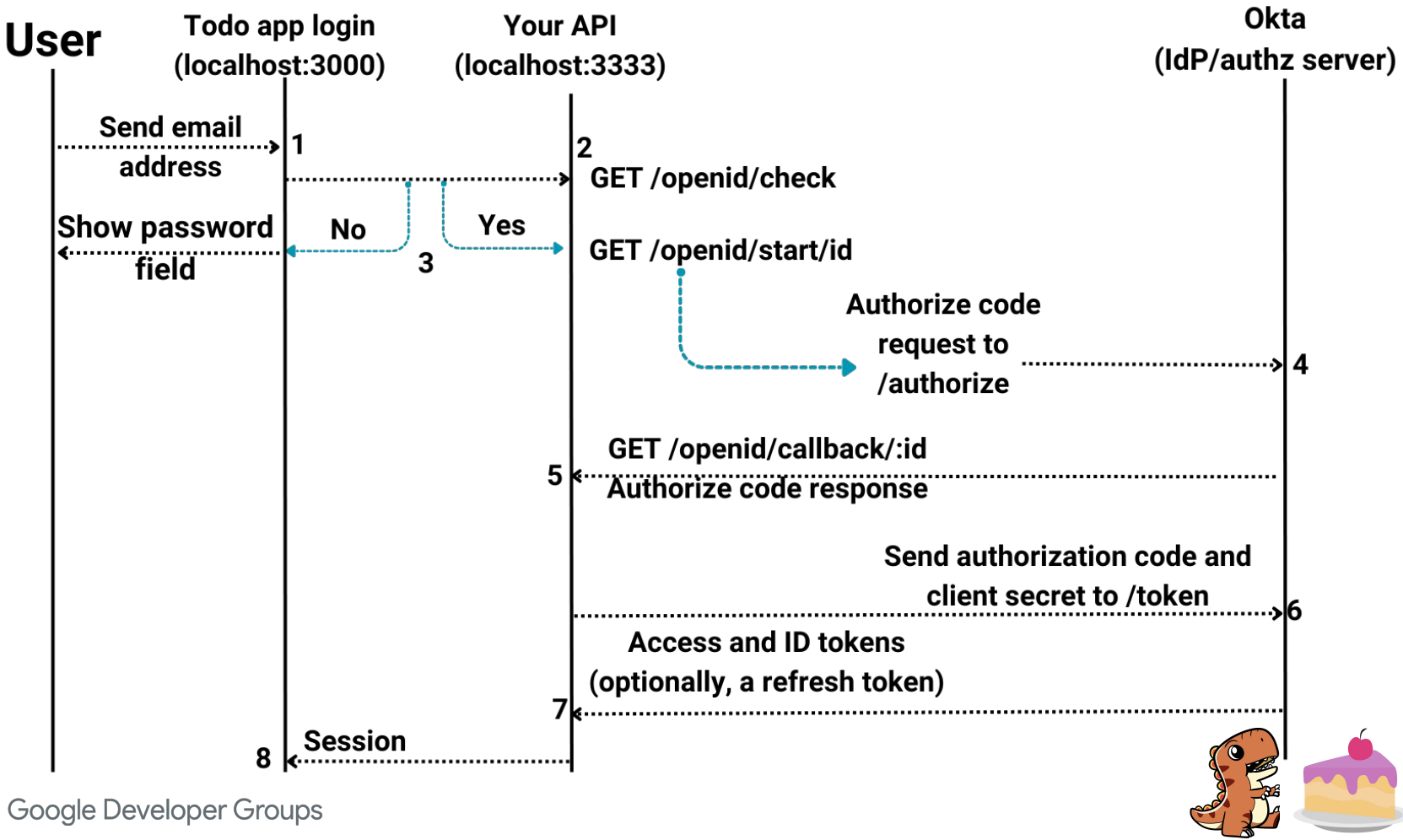
Help

Google Developer Groups



Google Developer Groups

Recap


User
Todo app login
(localhost:3000)
Your API
(localhost:3333)
Okta
(IdP/authz server)
Send email address
1
2
GET /openid/check
Show password field
No
Yes
3
GET /openid/start/id
Authorize code request to /authorize
4
GET /openid/callback/:id
5
Authorize code response
Send authorization code and client secret to /token
6
Access and ID tokens
(optionally, a refresh token)
7
8
Session

Recap

1. Redirected the user to the backend's /openid/start/ endpoint with an org's ID
2. Passport to redirect the user to their org's OpenID server
3. The user will prove their identity to their org's OpenID server, and that server will redirect the user to the app's backend
4. Passport will start a session to the Todo app
5. Only the app's backend will see the information from the OIDC server, so frontend tampering cannot intercept the user's session

Friends don't let friends build auth. Stay in touch!

- Semona Igama @sudomonas
- @OktaDev on YouTube, X
- developer.okta.com/blog
- #securedbyOIDC

Feedback Survey

🙏

Resources

- https://auth0.com/intro-to-iam/what-is-openid-connect-oidc
- https://openid.net/specs/openid-connect-core-1_0.html
- https://www.okta.com/openid-connect/
- https://jwt.io/
- https://jwt.io/introduction
- Tenor GIF - https://tenor.com/view/mando-way-this-is-the-way-mandalorian-star-wars-gif-18467370
- regionalevents.okta.com/enterprisereadyworkshops
- An Illustrated Guide to OAuth and OpenID Connect - https://www.youtube.com/watch?v=t18YB3xDfXI