

devfest

Wifi:

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

Workshop Repo:

- start - <https://bit.ly/3uh8H0s>
- completed <https://bit.ly/47wBhJK>

Slides:

<https://isemonia.github.io/oidc-workshop-devfest.pdf>



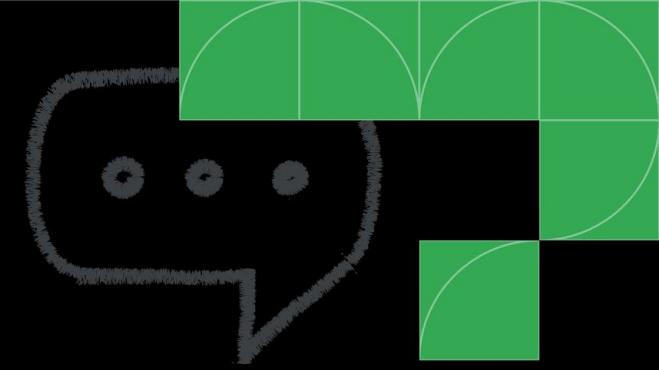
Google Developer Groups

Editable Location

You will need:

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

devfest



Build a lasting relationship
with your enterprise
customers

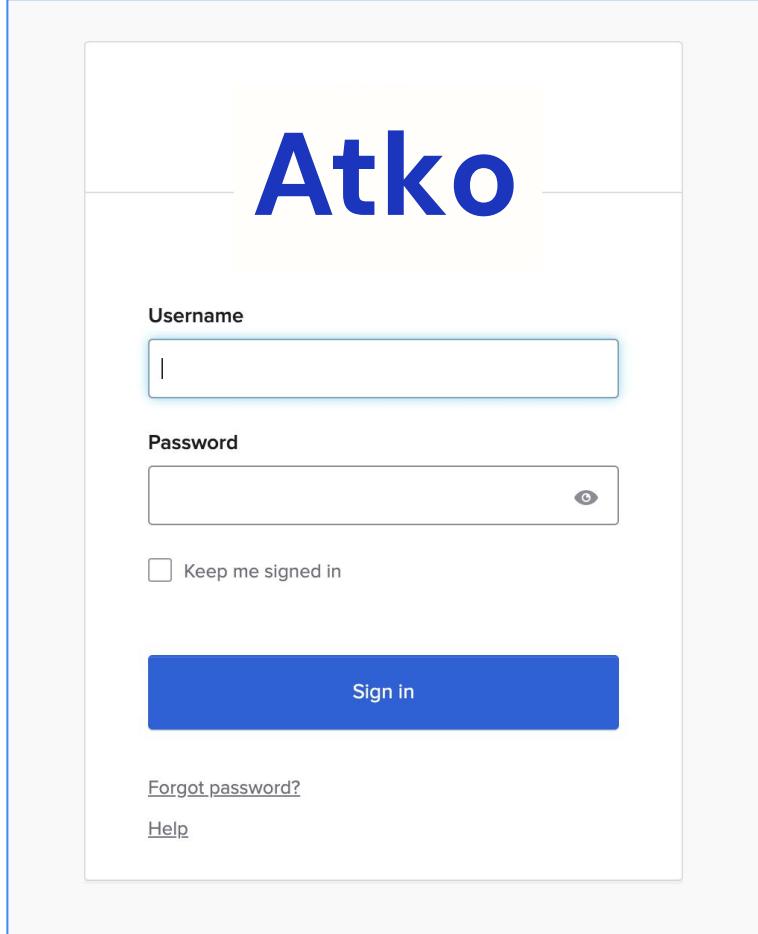


Who am I?

```
{  
  "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": "rhkhjm9oc",  
  "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)



The image shows a clean, modern login interface for 'Atko'. At the top center is the 'Atko' logo in a large, bold, blue sans-serif font. Below it is a horizontal line. The main form area has a light gray background. It contains two input fields: a 'Username' field with a placeholder 'Username' and a 'Password' field with a placeholder 'Password' and a small circular icon with a 'C' for copy. Below these fields is a 'Keep me signed in' checkbox. A large blue 'Sign in' button is centered at the bottom. At the very bottom of the form, there are two links: 'Forgot password?' and 'Help'.

Atko

Username

Password

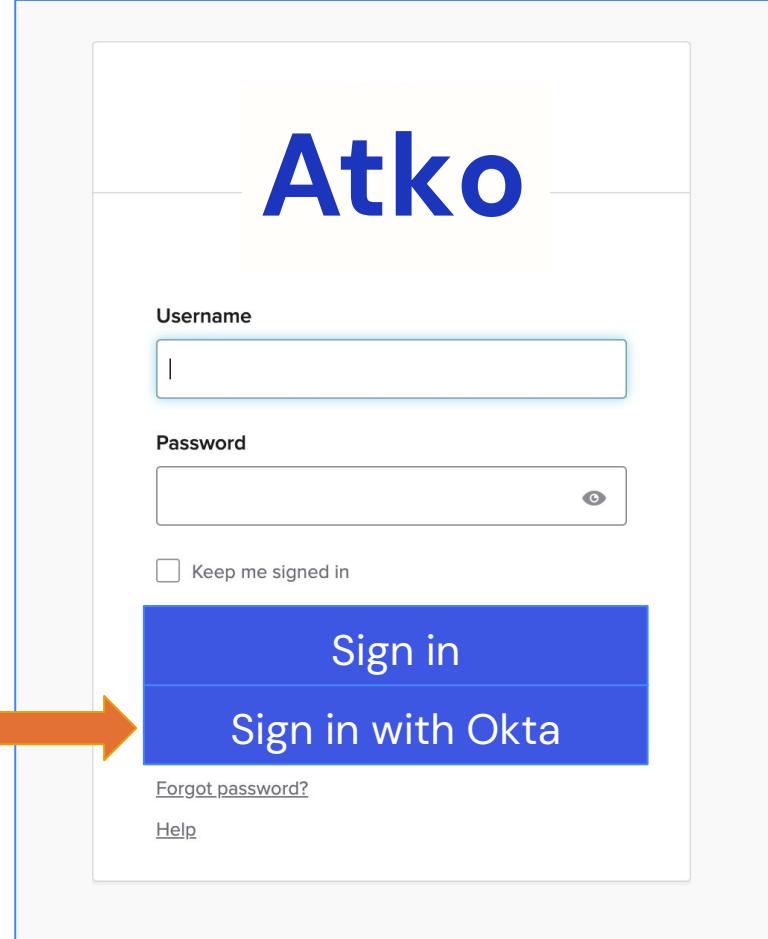
Keep me signed in

Sign in

[Forgot password?](#)

[Help](#)

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



The image shows a sign-in page for 'Atko'. At the top, the word 'Atko' is displayed in a large blue font. Below it are two input fields: 'Username' and 'Password'. Underneath the password field is a checkbox labeled 'Keep me signed in' with an unchecked state. At the bottom of the page are two large blue buttons: 'Sign in' and 'Sign in with Okta'. Below these buttons are links for 'Forgot password?' and 'Help'. The entire form is enclosed in a light gray border.

Atko

Username

Password

Keep me signed in

Sign in

Sign in with Okta

[Forgot password?](#)

[Help](#)

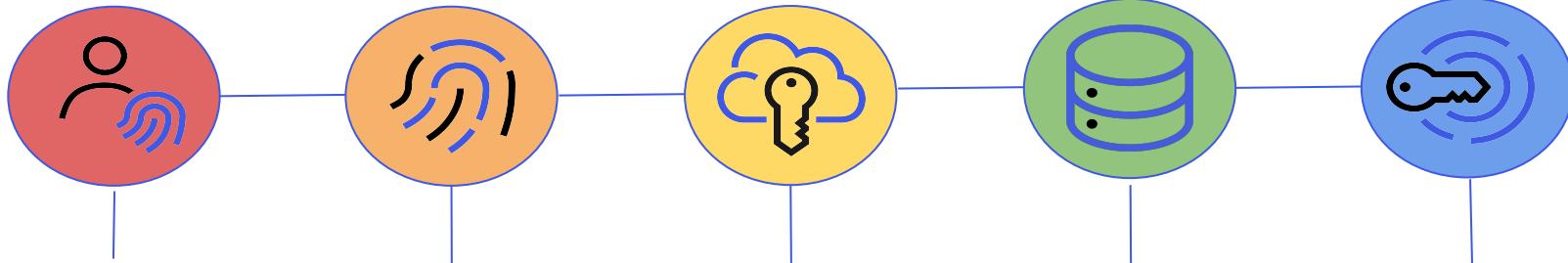


What is OpenID Connect (OIDC) and why use it?

Authentication → OpenID Connect

Authorization → OAuth 2.0





Standardizes
scopes:
openid,
profile,
email, and
address

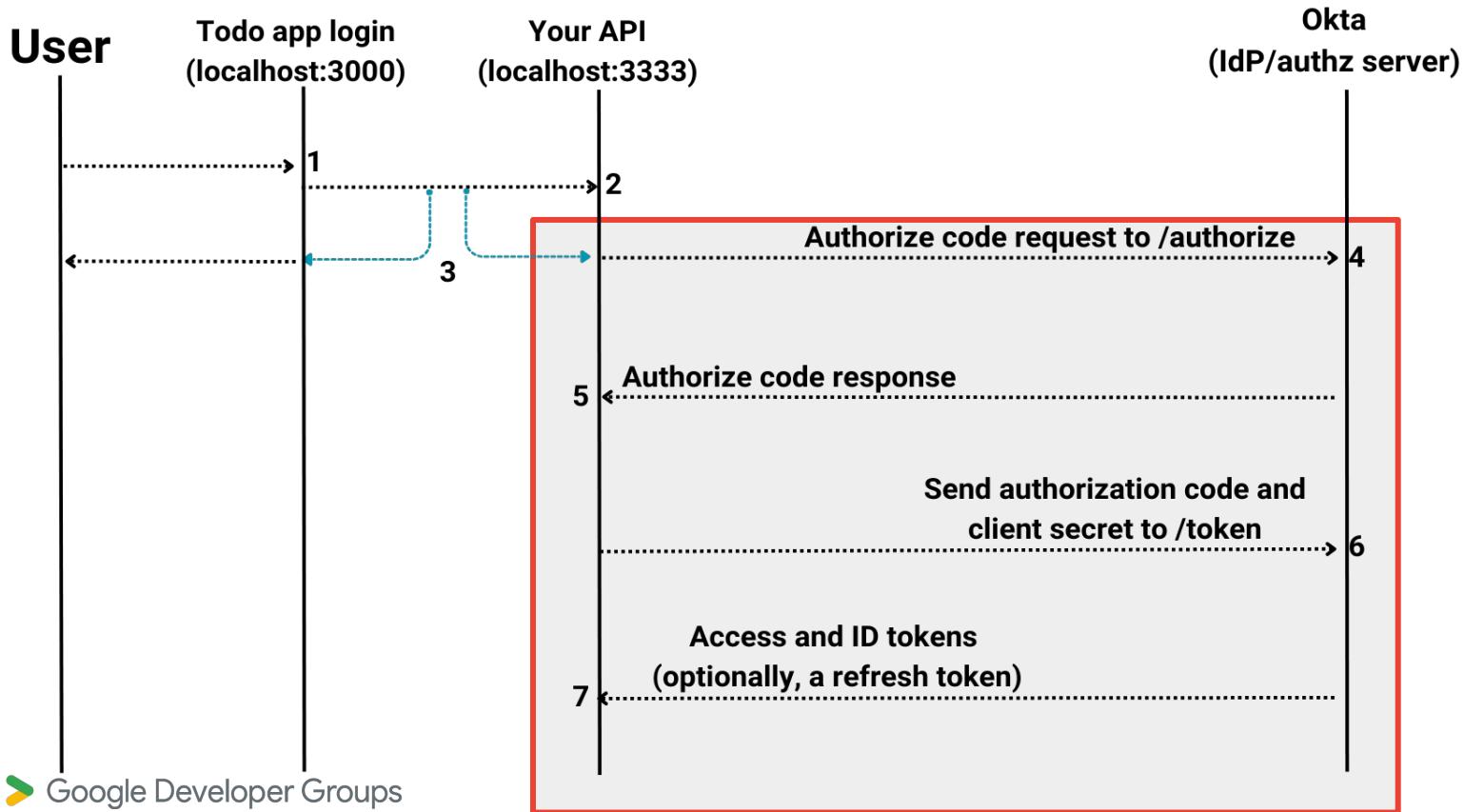
Standardizes
claims
for authn

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



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
hxnkyllebb

Response type (required)

code token id_token

ID token
JWT

Encoded

PASTE A TOKEN HERE

```
eyJraWQiOjIz0N1RjkyUDQtSzJQaHE3djhDa2x
Vb3VKWmJGdWwyZF9SR0FrWWZ00W5ViwiYWxnIj
oiu1MyNTYifQ.eyJzdWIiOiIwMHVwcDFoZjE2U2
4xSDJmYjBoNyIsIm5hbWUiOjIzZW1vbmcuaWdhb
WFAb2t0YS5jb20iLCJsb2NhbGUoIjVUyIsImVt
YWlsIjoic2Vtb25hLmlnYW1hQGh1bnRvbmFrLmN
vbsISisInZlcil6MSwiaXNzIjoiaHR0cHM6Ly9zZW
1vbmf2Mi5va3RhchJldm1ldy5jb20iLCJhdWQiO
iIwb2FxuDNmY2V5dF1SN1UxSTBoNyIsImlhcdI6
MTY50TQ4MTY5NSwiZXhwIjoxNjk5NDg1Mjk1LCJ
qdGkiOjIjRC5GN24tdGVmWURSOHtWnpWY01TUE
FXZTJnYj1CRU1JRGx4cXEyVTJqQV1vIwiYW1yI
jpBInB3ZCJdLCJpZHAi0iIwMG9wcDFoZXp4cjh5
aFh2UTBoNyIsIm5vbmlNIjoicHowd25iMxp4ZmY
iLCJwcmVmZXJyZWRfdXNlcm5hbWUiOjJzZW1vbmc
EuaWdhbWFAb2t0s5jb20iLCJnaZ1b19uYW11I
joiU2Vtb25hIwiZmfTaWx5X25hbWUiOjJJZFT
YSIIsInpvbpmZvIjoiQW1lcmljYs9Mb3NfQW5
nZWxlcycIsmVtYWlsX3ZlcmlmaWVkJp0cnVlC
JhdXRoX3RpBwUi0jE20Tk0DE20TqsInRlc3Qi0
iIiLCJmYXZvcm1oZUNvbG9yIjoiwmkIiwiChJ1
ZmVycmVkJxhbmd1YwdlIjoiIwiZW1wbG95ZWV
OdWiZXIi0IyMzQ1Njc4In0.jU3o0GCLBzZ71u
Q51jyApwSkF4rmE36W5FIJUh3JoFPW9VuYmdaJ0
eotSoq9_zgtfrRbuQk35z2QSCw4eW189bNhpIrA
sgW4n8M0g4JcCFZ0EKQIIYV13wfyG2ZrFgdP3Q9
n9n0672ikjwJMHLiEJGjrYt316Thof6W03kLItd
nEx4T70_W5gCE53Hr4xKrGcB6bNtYMGb9xG_QbF
H9yTmt1H2uMiMiY9fwSkK_Ruc4GGwQVNj1QVtDH
39KMkxqx6rlyuUWoJY1I7Y9WHFYg0WK3LEJW-
BQiIooONMjSsRVd0kw4ltoIFm9p0trYCFKML-
9KQna3S3uukkQ2-6pf3A
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "kid": "bgCuF92P4-K2Phq7v8Ck1UouJZbFu12d_RGAkYfN9no",
  "alg": "RS256"
}
```

PAYOUT: DATA

```
{
  "sub": "00upp1hf16Sn1H2fb0h7",
  "name": "semona.igama@okta.com",
  "locale": "US",
  "email": "semona.igama@huntonak.com",
  "ver": 1,
  "iss": "https://semonav2.oktapreview.com",
  "aud": "0aqh3fcy7YR7U10h7",
  "iat": 1699481695,
  "exp": 1699485295,
  "jti": "ID_F7n-
tefYDR8XmZzVcISPAw2gb9BEMID1xqq2U2jAYo",
  "amr": [
    "pwd"
  ],
  "idp": "00opp1hezxr8yhXv0h7",
  "nonce": "pzewnb1zxff",
  "preferred_username": "semona.igama@okta.com",
  "given_name": "Semonia",
  "family_name": "Igama",
  "zoneinfo": "America/Los_Angeles",
  "email_verified": true,
  "auth_time": 1699481694,
  "test": "",
  "favoriteColor": "red",
  "preferred_language": "",
  "employeeNumber": "2345678"
}
```

VERIFY SIGNATURE

```
RSASHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
{
  "e": "AQAB"
}
```

claims

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



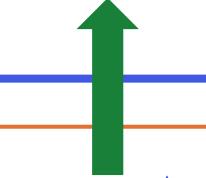
IdP (Okta) +
login using
via OIDC

Ready to take on the day?

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

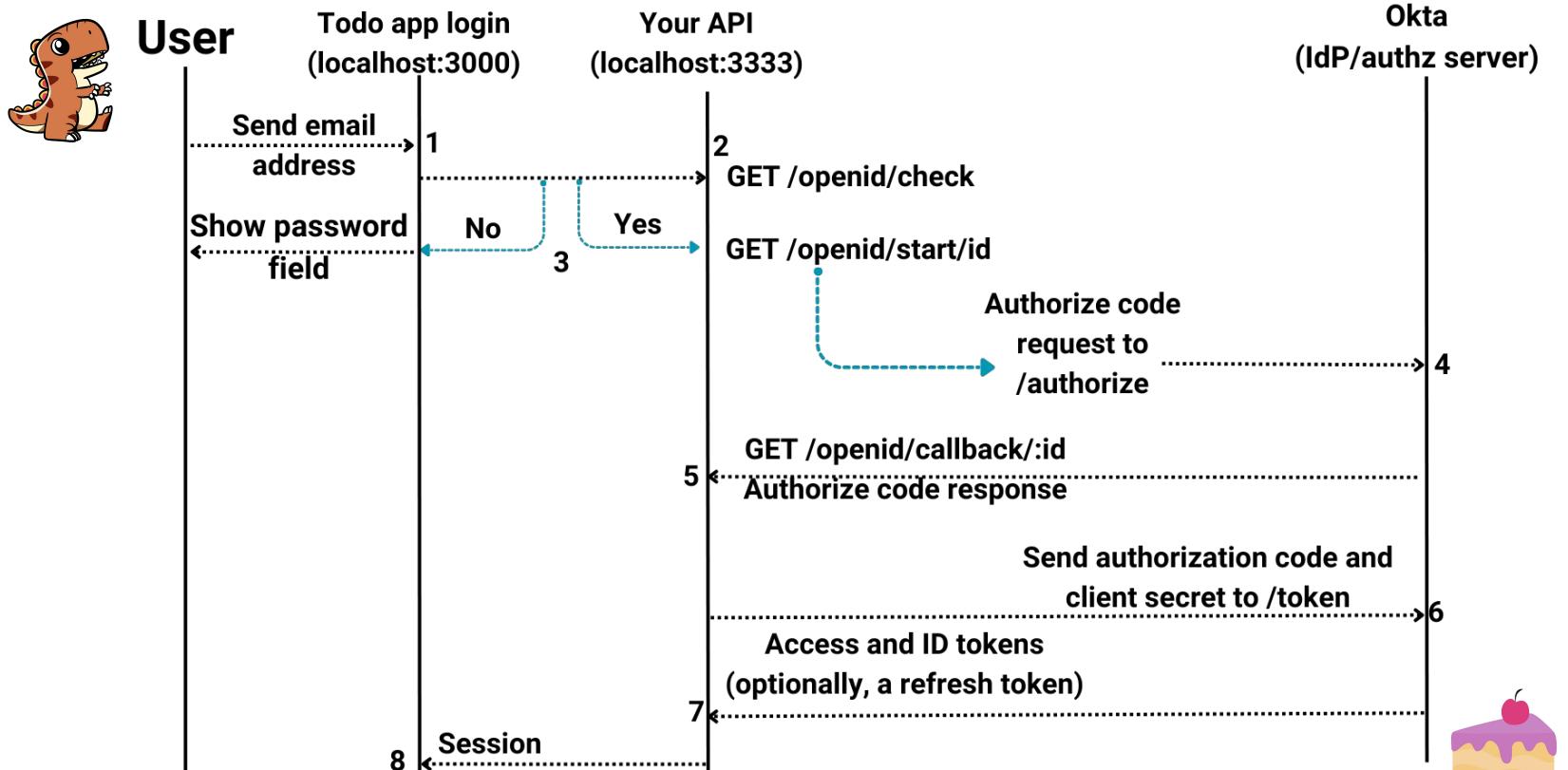
Email address

Sign in



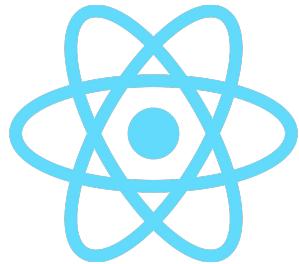
Enterprise
customers' users

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) => { ...
});

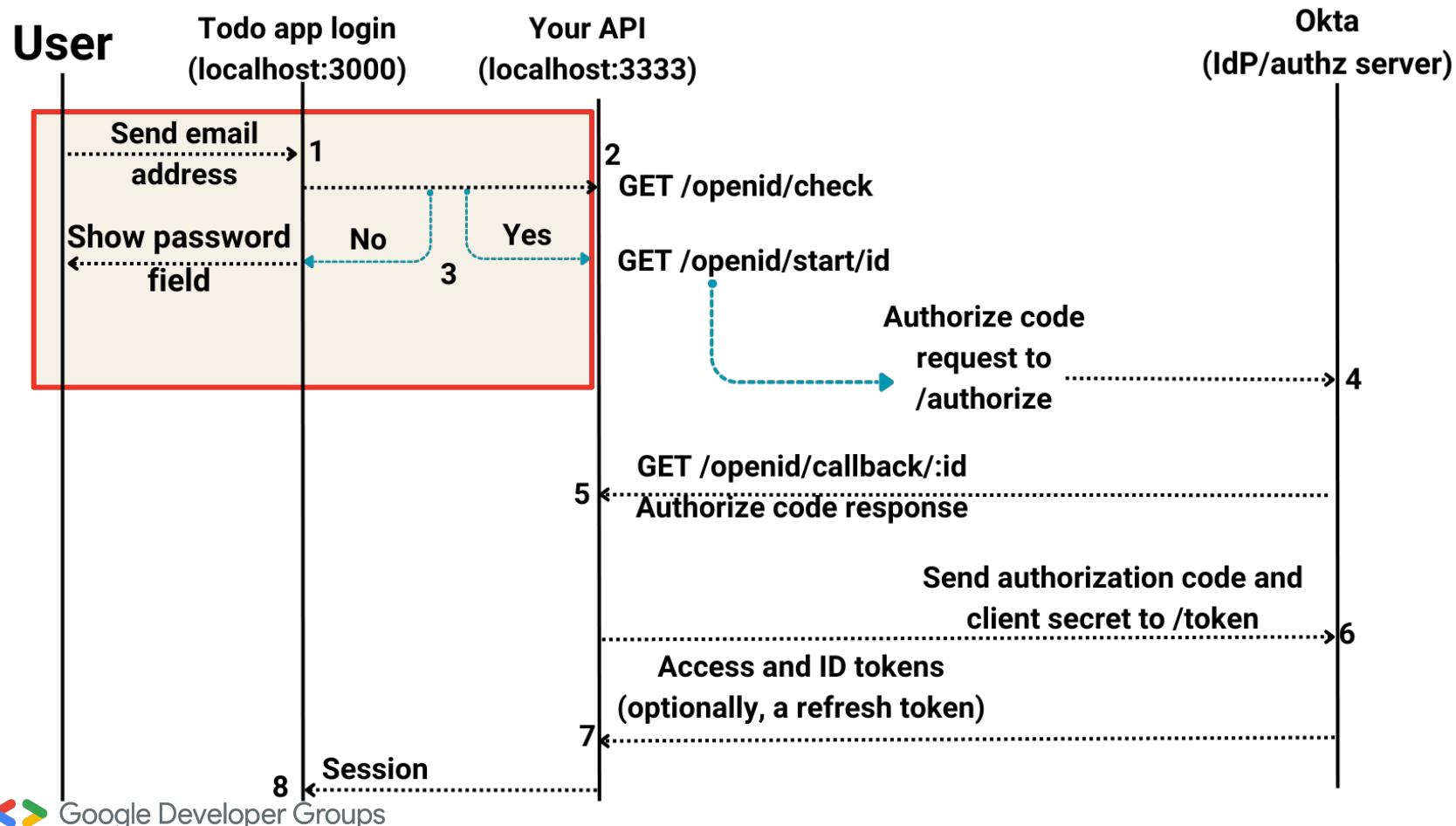
function createStrategy(org) {
>   return new OpenIDConnectStrategy({ ...
},  

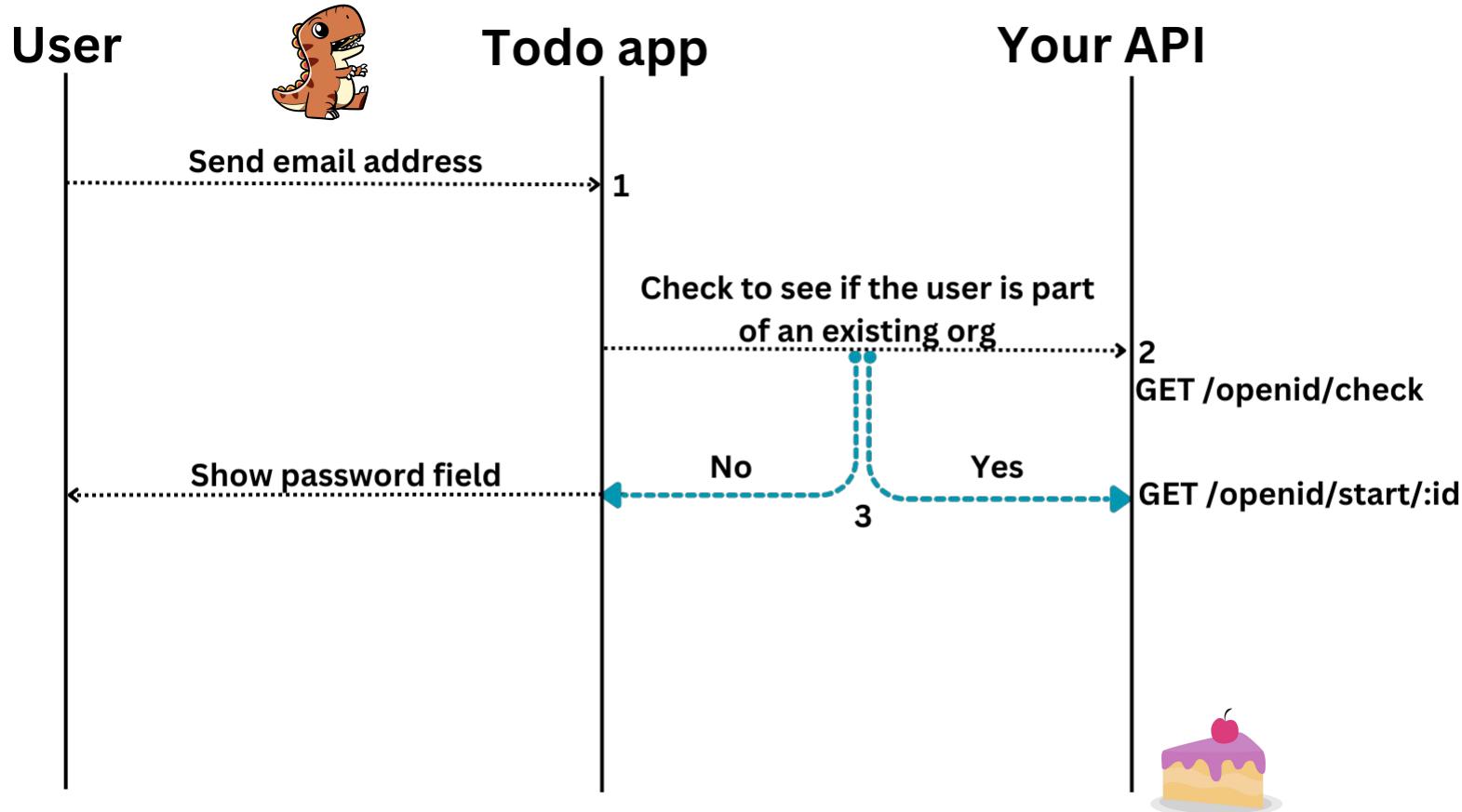
>   async function verify(issuer, profile, cb) { ...
  })
}

> // 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





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

Ready to take on the day?

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

Email address

Password

[Sign in](#)



```
graph LR; A[Before] --> B[After]; B --> C[Redirect to IdP]
```

After

Ready to take on the day?

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

Email address

[Sign in](#)



```
graph LR; A[Before] --> B[After]; B --> C[Redirect to IdP]
```

Redirect to IdP

okta

Sign In

Username

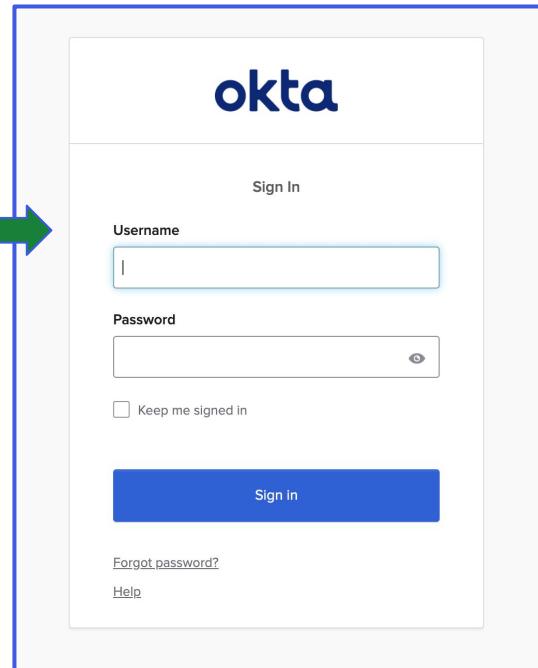
Password

Keep me signed in

[Sign in](#)

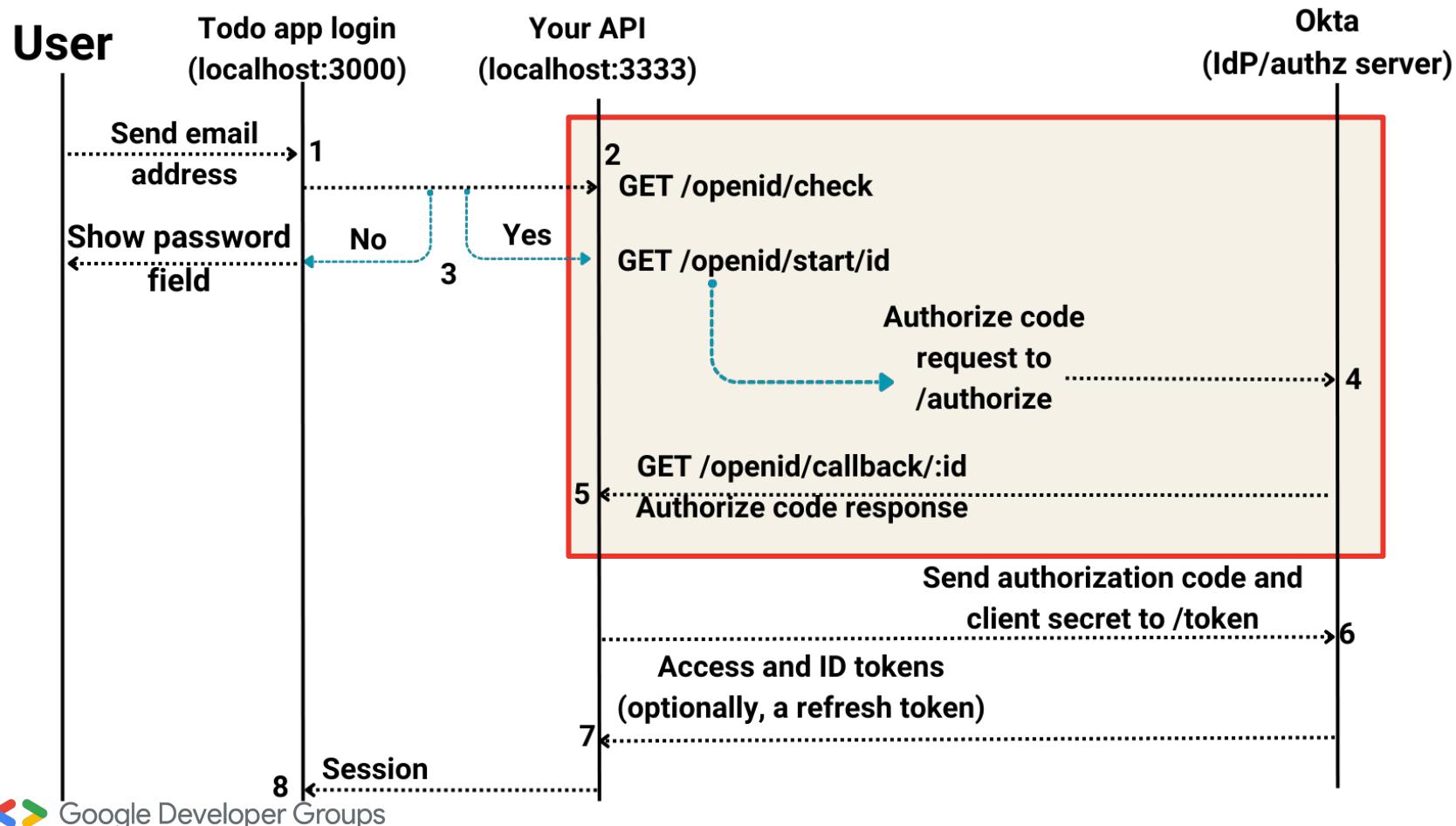
[Forgot password?](#)

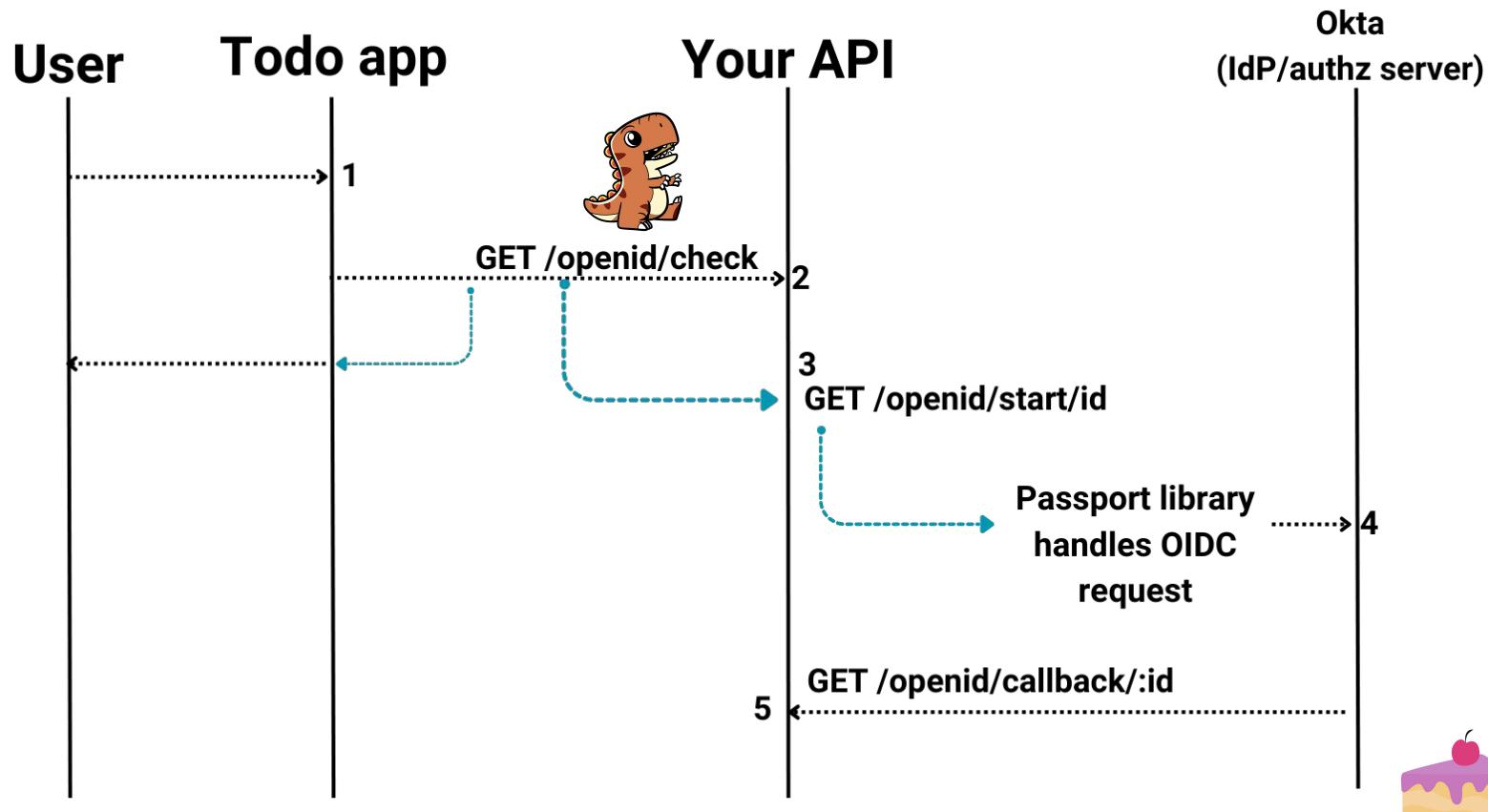
[Help](#)



```
graph LR; A[Before] --> B[After]; B --> C[Redirect to IdP]
```

Backend





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>

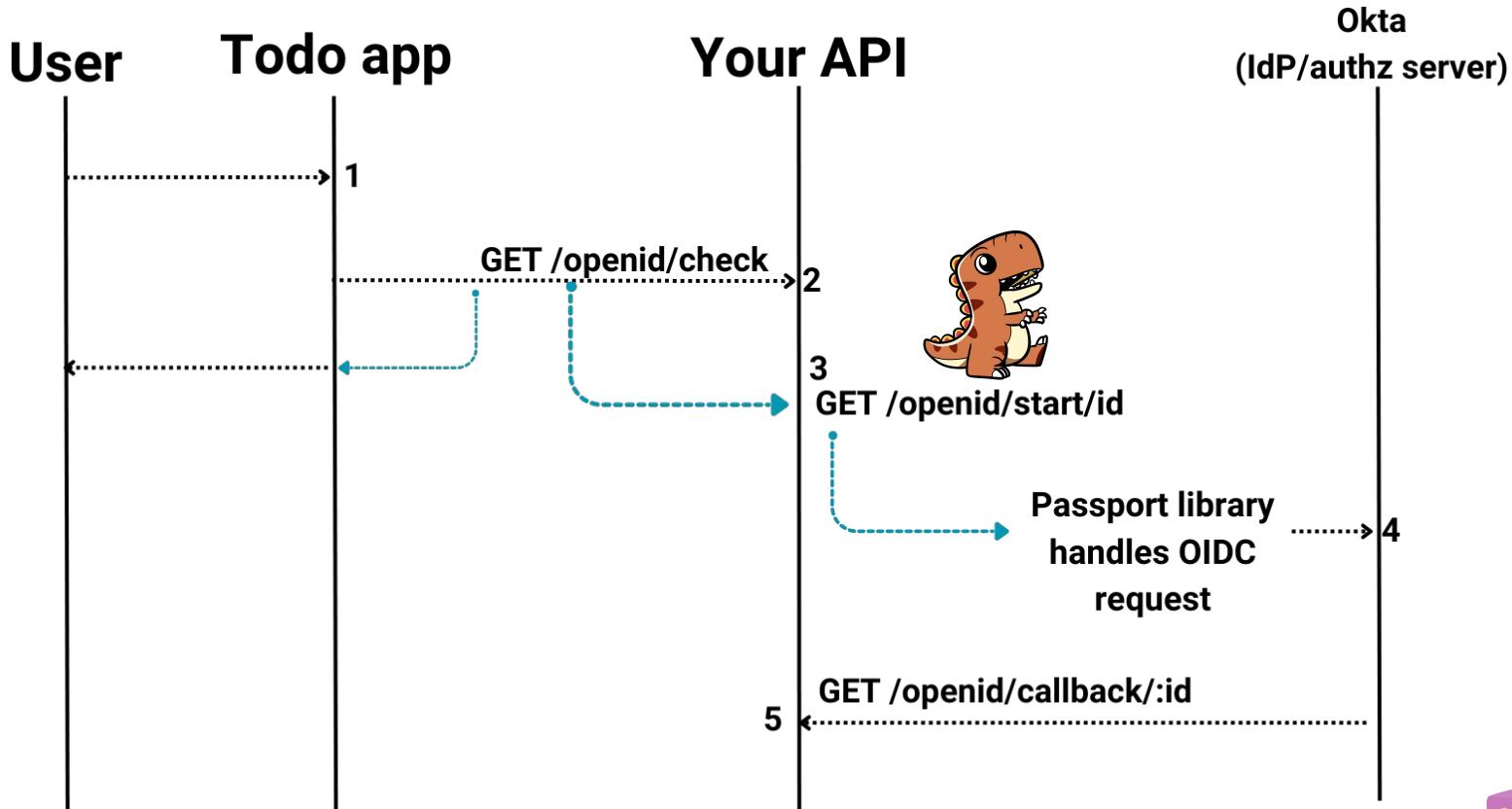
Enable the OIDC flow

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>



Use the Passport OIDC library

Steps:

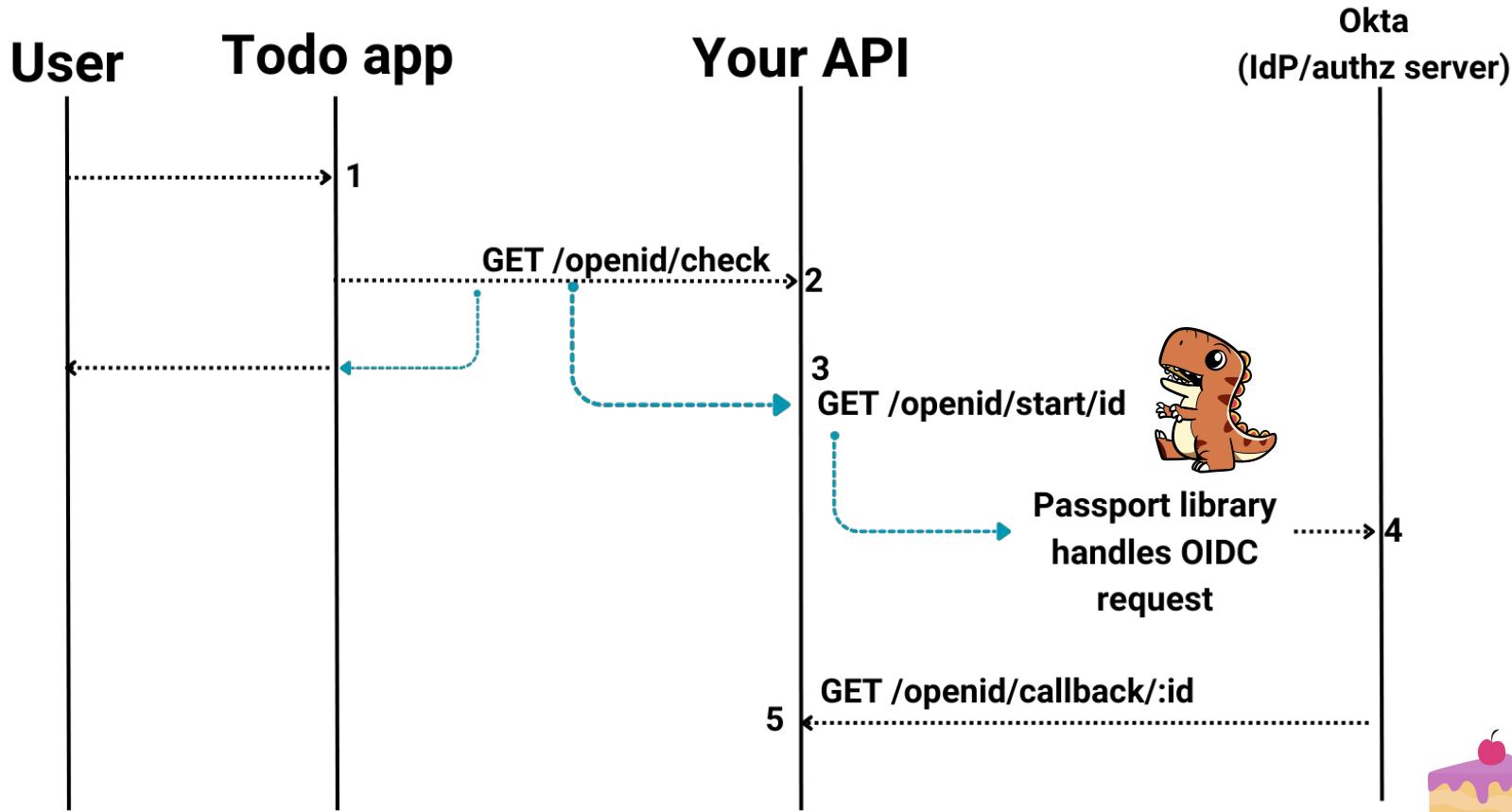
1. Add the import for the OIDC passport after the import for passport-local at the top of `apps/api/src/main.ts`
→ `import passportOIDC from
'passport-openidconnect';`
2. Define an variable for the OIDC strategy so you can utilize it within the API
→ `const OpenIDConnectStrategy =
 passportOIDC.Strategy;`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



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) => {

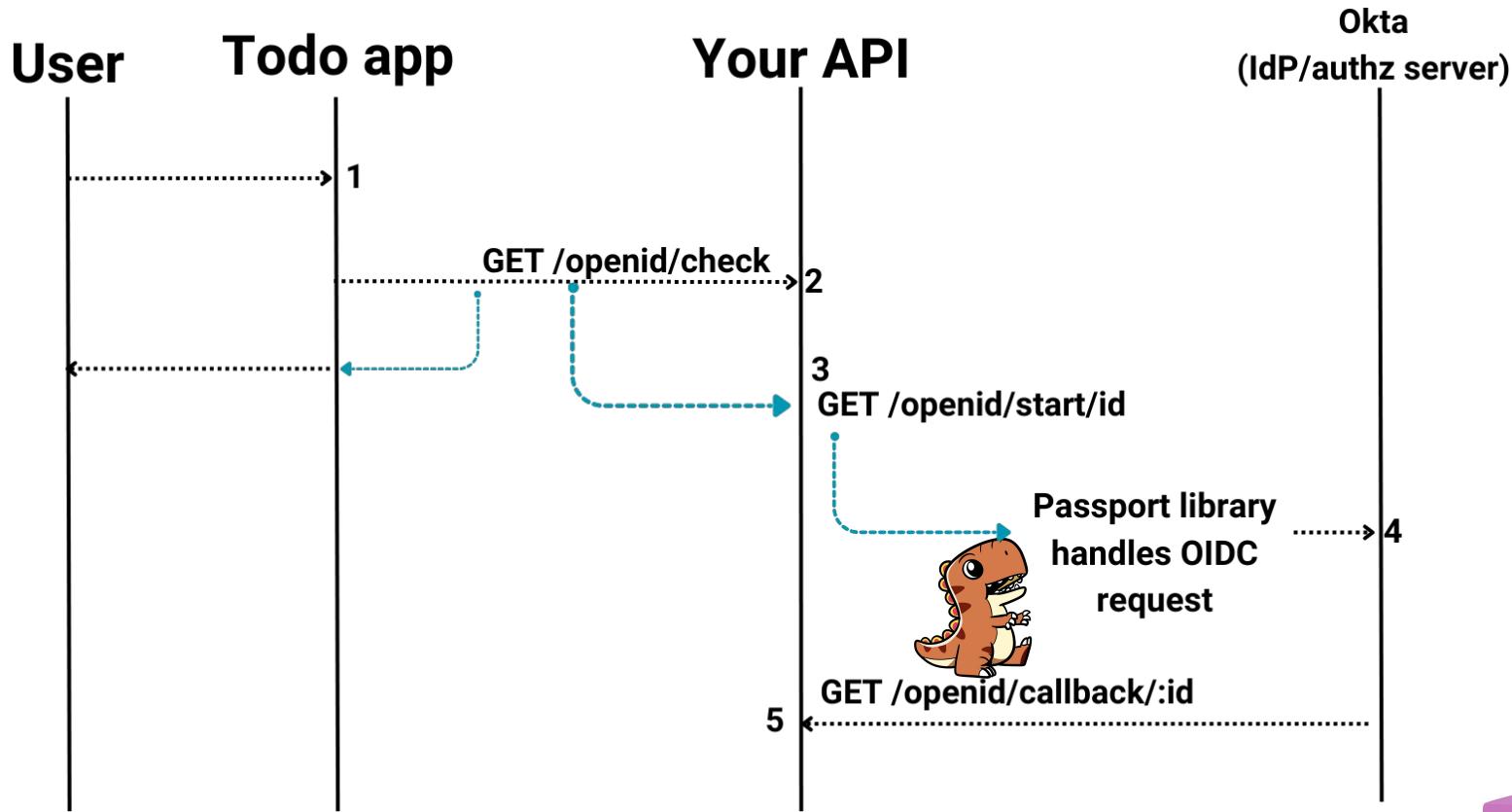
  const org = await orgFromId(req.params.id);
  if(!org) {
    return res.sendStatus(404);
  }

  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/3QZuR07>



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

The screenshot shows the top navigation bar of the developer.okta.com website. It includes links for 'Community', 'Blog', 'Pricing', 'Okta.com', 'Log in', and a prominent 'Sign up' button. An orange arrow points from the text 'Choose what works best. Sign up is free.' to the 'Sign up' button.

okta Developer

Community ▾ Blog Pricing

Okta.com Log in Sign up

Guides Concepts References Languages & SDKs Release Notes

Okta Developer

Our developer portal enables you to deploy auth that protects your users, apps, APIs, and infrastructure.

Get your app [enterprise-ready](#) with [free virtual workshops!](#)

Start your Workforce Identity journey

Welcome! Start with **Learn** if you're new to Workforce Identity Cloud, or find the step in your journey and follow the links to browse docs.

Learn



Understand the basics of identity

Learn the key concepts you need for creating identity and access management (IAM) solutions for WIC.

[Understand IAM](#)

[How WIC works](#)

[Choose an authentication protocol](#)

[Get a developer org](#)

Choose what works best. **Sign up is free.**

Customer Identity Cloud

powered by auth0

FREE

For the first tier

BEST FOR DEVELOPERS

Secure my customers or SaaS applications

Build intuitive, secure user experiences in customer-facing applications.

[Try Customer Identity Cloud →](#)

Workforce Identity Cloud

FREE TRIAL

Get access for 30 days

BEST FOR IT ADMINS

Secure my employees, contractors, & partners

Manage secure, frictionless access to the tools and data your teams need, on demand.

[Try Workforce Identity Cloud →](#)

FREE

Test, explore, and manage integrations

BEST FOR DEVELOPERS

Access the Okta Developer Edition Service

Test your code and apps, as well as manage and automate Okta for employees and partners.

[Sign up free for Developer Edition](#)

Okta CLI - cli.okta.com

Okta CLI (beta) Documentation

The Okta CLI is the easiest way to get started with Okta!
Get started today by installing on your platform.

```
brew install --cask oktadeveloper/tap/okta
```

[macOS Install](#)

[View installation instructions for all platforms →](#)
[View all Okta CLI commands →](#)

Getting Started

```
$ okta start spring-boot
Registering for a new Okta account, if you would like to use an existing account, use 'okta login' instead.

First name: Jamie
Last name: Example
Email address: jamie@example.com
Company: Okta Test Company
Creating new Okta Organization, this may take a minute:
OrgUrl: https://dev-123456.okta.com
An email has been sent to you with a verification code.
```

Create an OIDC Application in 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	Okta Admin Console
INACTIVE	0	Okta Browser Plugin

Okta Dashboard

Create a new app integration

Sign-in method

[Learn More !\[\]\(f751fb5266fcde85b8e494ae0908d01e_img.jpg\)](#)

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

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)

[Learn More](#)

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 application to everyone in your org, ~~only selected group(s)~~, or skip assignment until after app creation.

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.



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



- 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!!

Dashboard

Directory

Customizations

Applications

Applications

Self Service

API Service Integrations

Security

Workflow

Reports

Settings

Todo App

Active



View Logs

General

Sign On

Assignments

Okta API Scopes

Application Rate Limits



Client Credentials

Edit

Client ID

0oaczcp47jVYyB9w15d7

Public identifier for the client that is required for all OAuth flows.

Client authentication

 Client secret Public key / Private key

Proof Key for Code Exchange (PKCE)

 Require PKCE as additional verification

CLIENT SECRETS

Generate new secret

Creation date	Secret	Status
Oct 27, 2023	*****	



General

HealthInsight

Authenticators

Authentication Policies

Global Session Policy

Profile Enrollment

Identity Providers

Delegated Authentication

Networks

Behavior Detection

Device Assurance Policies

Device Integrations

Administrators

API

Workflow

[← Back to Authorization Servers](#)

default

[Help](#)[Active ▾](#)

Default

[Settings](#)[Scopes](#)[Claims](#)[Access Policies](#)[Token Preview](#)

Settings

[Edit](#)Name defaultAudience api://defaultDescription Default Authorization Server for your ApplicationsIssuer Okta URL (<https://dev-77700421.okta.com/oauth2/default>)

Authorization Servers

An authorization server defines your security boundary, and is used to mint access and identity tokens for use with OIDC clients and OAuth 2.0 service accounts when accessing your resources via API. Within each authorization server you can define your own OAuth scopes, claims, and access policies. Read more at [help page](#)

Metadata URI <https://dev-77700421.okta.com/oauth2/default/.well-known/oauth-authorization-server>Signing Key Rotation [?](#) AutomaticLast Rotation Oct 27, 2023

```
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	issuer	authorization_endpoint	token_endpoint	userinfo_endpoint	client_id	client_secret	apikey	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.

Dashboard

Directory

People

Groups

Devices

Profile Editor

Directory Integrations

Profile Sources

Customizations

Applications

People

Add person

More actions ▾

Search for users by first name, primary email or username



Advanced search ▾

Status

All

Showing

Person & username

Primary email

S

semona igama

semona.igama+devtest@okta.com

A

semona.igama+devtest@okta.com

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

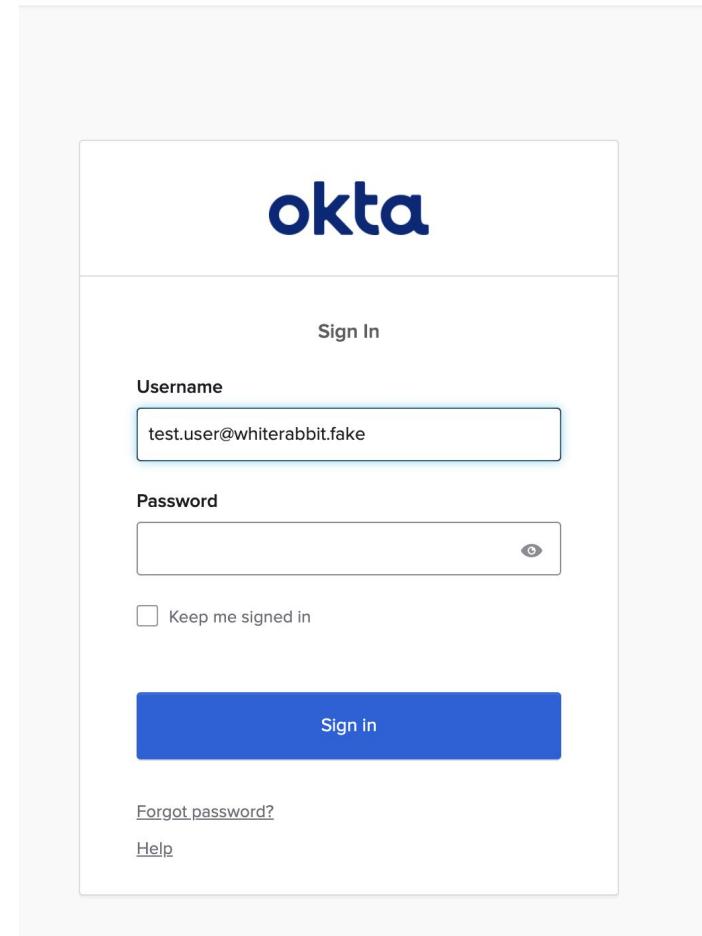
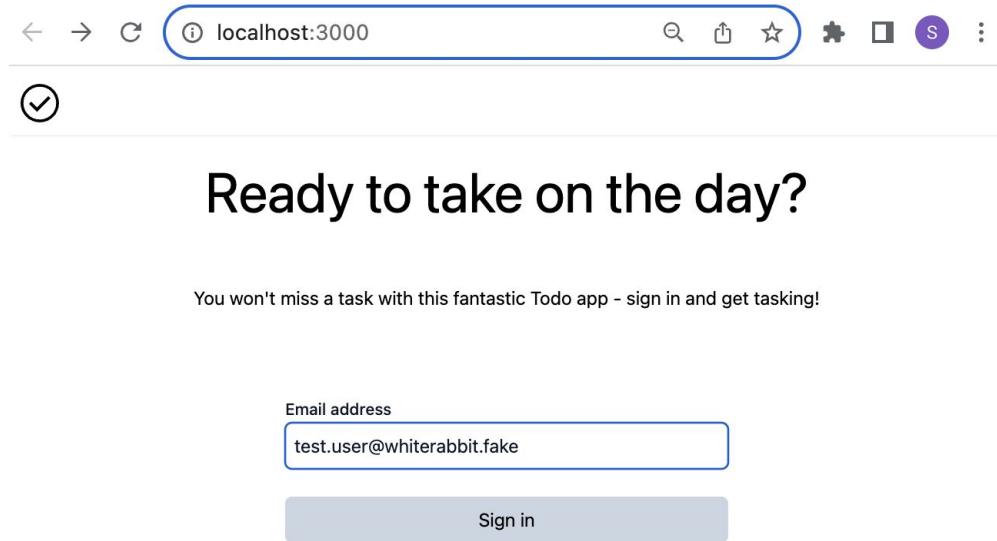
User must change password on first login

Do not send unsolicited or unauthorized activation emails. [Read more](#)

Save

Save and Add Another

Cancel





Welcome, Test User! [Profile](#) [Sign out](#)

Ready to take on the day?

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

[Where's my todos?](#)

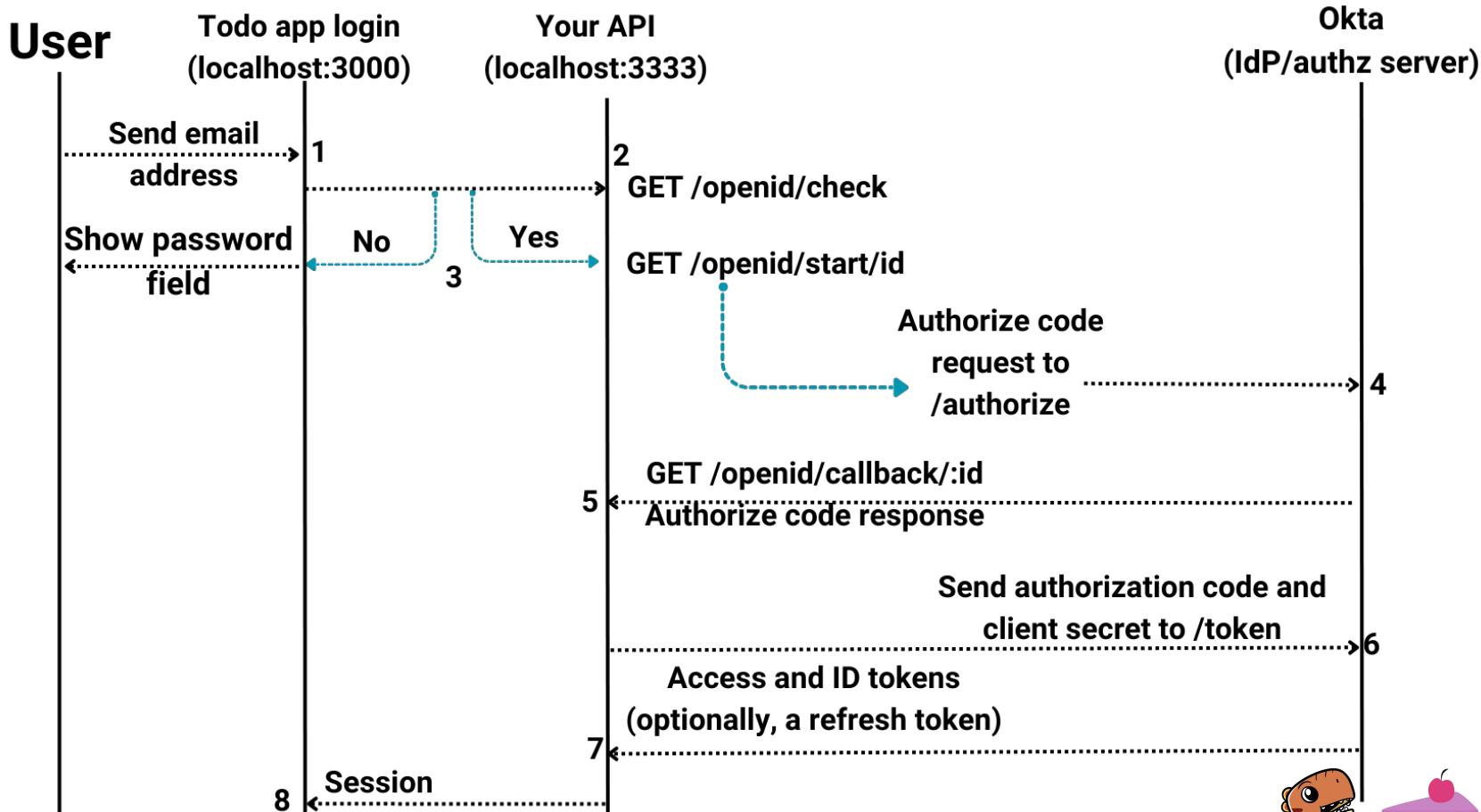
localhost:5555

User Org +

Filters None Fields All Showing 2 of 2 Add record

	id #	email A?	password A?	name A?	Todo []	org ()?	orgId #?	externalId A?
	1	semona.igama+devtest@...	null	semona igama	0 Todo	Org	1	00ucz8s1y4qGmS6F25d7
	2	test.user@whiterabbit...	null	Test User	0 Todo	Org	1	00ud0i9yglooMcfNY5d7

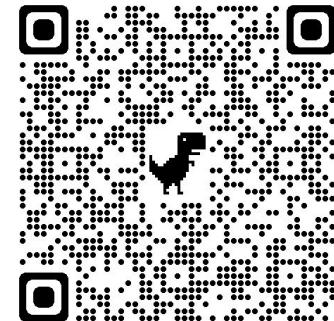
Recap



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!



Feedback
Survey

- Semona Igama @sudomonas
- @0ktaDev on YouTube, X
- developer.okta.com/blog
- #securedbyOIDC



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>