To integrate GitHub Single Sign-On (SSO) with Backstage, you need to configure the **GitHub Authentication Provider**. This will allow users to sign in to Backstage using their GitHub credentials, and no other sign-in methods (like guest access) will be available.

**Step 1: Create a GitHub OAuth App**

1. Go to your GitHub account and navigate to **Settings > Developer Settings > OAuth Apps**.
2. Click **New OAuth App**.
3. Fill in the details:
   * **Application Name**: Backstage
   * **Homepage URL**: http://localhost:3000 (or your Backstage URL).
   * **Authorization callback URL**: http://localhost:7007/api/auth/github/handler/frame (or your Backstage API URL).
4. Click **Register Application**.
5. Note down the **Client ID** and generate a **Client Secret**.

**Step 2: Update Backstage Configuration**

1. Open your Backstage project in a code editor.
2. Locate the app-config.yaml file (usually in the root directory).
3. Add the GitHub authentication provider configuration:

auth:

environment: development

providers:

github:

development:

clientId: YOUR\_GITHUB\_CLIENT\_ID

clientSecret: YOUR\_GITHUB\_CLIENT\_SECRET

Replace YOUR\_GITHUB\_CLIENT\_ID and YOUR\_GITHUB\_CLIENT\_SECRET with the values from your GitHub OAuth App.

1. Save the file.

**Step 3: Modify the Backstage App to Enable GitHub SSO**

1. Open the packages/app/src/App.tsx file.
2. Import the githubAuthApiRef and SignInPage:

import { githubAuthApiRef } from '@backstage/core-plugin-api';

import { SignInPage } from '@backstage/core-components';

1. Modify the app.createApp call to include the SignInPage component:

const app = createApp({

apis,

components: {

SignInPage: props => (

<SignInPage

{...props}

providers={[

{

id: 'github-auth-provider',

title: 'GitHub',

apiRef: githubAuthApiRef,

},

]}

/>

),

},

});

1. Save the file.

**Step 4: Disable Guest Access**

1. Open the packages/backend/src/plugins/auth.ts file.
2. Modify the createRouter function to disable guest access:

import { createRouter } from '@backstage/plugin-auth-backend';

import { Router } from 'express';

import { PluginEnvironment } from '../types';

export default async function createPlugin(

env: PluginEnvironment,

): Promise<Router> {

return await createRouter({

logger: env.logger,

config: env.config,

database: env.database,

discovery: env.discovery,

providerFactories: {

...defaultAuthProviderFactories,

guest: undefined, // Disable guest access

},

});

}

1. Save the file.

**Step 5: Restart Backstage**

1. Stop your Backstage server if it’s running.
2. Start the Backstage server:

yarn start

1. Open your browser and navigate to http://localhost:3000.

**Step 6: Test the GitHub SSO Integration**

1. When you open Backstage, you should see a **Sign In** page with only the GitHub option.
2. Click **Sign In with GitHub**.
3. You will be redirected to GitHub to authorize the app.
4. After authorization, you will be redirected back to Backstage and logged in.

**Example Code**

Here’s a summary of the key files with the changes:

**app-config.yaml**

auth:

environment: development

providers:

github:

development:

clientId: YOUR\_GITHUB\_CLIENT\_ID

clientSecret: YOUR\_GITHUB\_CLIENT\_SECRET

**packages/app/src/App.tsx**

import { githubAuthApiRef } from '@backstage/core-plugin-api';

import { SignInPage } from '@backstage/core-components';

const app = createApp({

apis,

components: {

SignInPage: props => (

<SignInPage

{...props}

providers={[

{

id: 'github-auth-provider',

title: 'GitHub',

apiRef: githubAuthApiRef,

},

]}

/>

),

},

});

**packages/backend/src/plugins/auth.ts**

import { createRouter, defaultAuthProviderFactories } from '@backstage/plugin-auth-backend';

import { Router } from 'express';

import { PluginEnvironment } from '../types';

export default async function createPlugin(

env: PluginEnvironment,

): Promise<Router> {

return await createRouter({

logger: env.logger,

config: env.config,

database: env.database,

discovery: env.discovery,

providerFactories: {

...defaultAuthProviderFactories,

guest: undefined, // Disable guest access

},

});

}

To enable multiple authentication providers (Google, Microsoft, GitHub, GitLab, Bitbucket) and guest sign-in in Backstage, you need to configure each provider in the app-config.yaml file and update the SignInPage component to include all the options. Below is a step-by-step guide to achieve this:

**Step 1: Configure Authentication Providers in app-config.yaml**

Open your app-config.yaml file and add configurations for all the providers you want to support:

auth:

environment: development

providers:

google:

development:

clientId: YOUR\_GOOGLE\_CLIENT\_ID

clientSecret: YOUR\_GOOGLE\_CLIENT\_SECRET

microsoft:

development:

clientId: YOUR\_MICROSOFT\_CLIENT\_ID

clientSecret: YOUR\_MICROSOFT\_CLIENT\_SECRET

tenantId: YOUR\_MICROSOFT\_TENANT\_ID

github:

development:

clientId: YOUR\_GITHUB\_CLIENT\_ID

clientSecret: YOUR\_GITHUB\_CLIENT\_SECRET

gitlab:

development:

clientId: YOUR\_GITLAB\_CLIENT\_ID

clientSecret: YOUR\_GITLAB\_CLIENT\_SECRET

baseUrl: https://gitlab.com # Use your GitLab instance URL if self-hosted

bitbucket:

development:

clientId: YOUR\_BITBUCKET\_CLIENT\_ID

clientSecret: YOUR\_BITBUCKET\_CLIENT\_SECRET

Replace the placeholders (YOUR\_\*\_CLIENT\_ID, YOUR\_\*\_CLIENT\_SECRET, etc.) with the actual values from your OAuth apps.

**Step 2: Create OAuth Apps for Each Provider**

1. **Google**:
   * Go to the [Google Cloud Console](https://console.cloud.google.com/).
   * Create credentials for an OAuth 2.0 Client ID.
   * Set the authorized redirect URI to http://localhost:7007/api/auth/google/handler/frame.
2. **Microsoft**:
   * Go to the [Azure Portal](https://portal.azure.com/).
   * Register a new application under **App registrations**.
   * Set the redirect URI to http://localhost:7007/api/auth/microsoft/handler/frame.
3. **GitHub**:
   * Go to your GitHub account and navigate to **Settings > Developer Settings > OAuth Apps**.
   * Create a new OAuth app with the callback URL http://localhost:7007/api/auth/github/handler/frame.
4. **GitLab**:
   * Go to your GitLab account and navigate to **User Settings > Applications**.
   * Create a new application with the redirect URI http://localhost:7007/api/auth/gitlab/handler/frame.
5. **Bitbucket**:
   * Go to your Bitbucket account and navigate to **Workspace settings > OAuth consumers**.
   * Create a new OAuth consumer with the callback URL http://localhost:7007/api/auth/bitbucket/handler/frame.

**Step 3: Update the SignInPage Component**

Open the packages/app/src/App.tsx file and modify it to include all the authentication providers:

import {

googleAuthApiRef,

microsoftAuthApiRef,

githubAuthApiRef,

gitlabAuthApiRef,

bitbucketAuthApiRef,

SignInPage,

} from '@backstage/core-plugin-api';

const app = createApp({

apis,

components: {

SignInPage: props => (

<SignInPage

{...props}

providers={[

{

id: 'google-auth-provider',

title: 'Google',

apiRef: googleAuthApiRef,

},

{

id: 'microsoft-auth-provider',

title: 'Microsoft',

apiRef: microsoftAuthApiRef,

},

{

id: 'github-auth-provider',

title: 'GitHub',

apiRef: githubAuthApiRef,

},

{

id: 'gitlab-auth-provider',

title: 'GitLab',

apiRef: gitlabAuthApiRef,

},

{

id: 'bitbucket-auth-provider',

title: 'Bitbucket',

apiRef: bitbucketAuthApiRef,

},

{

id: 'guest',

title: 'Guest',

apiRef: undefined, // Guest access

},

]}

/>

),

},

});

**Step 4: Enable Guest Access**

If you want to allow guest access, ensure the guest provider is included in the SignInPage component (as shown above). Guest access does not require any additional configuration in the app-config.yaml.

**Step 5: Restart Backstage**

1. Stop your Backstage server if it’s running.
2. Start the Backstage server:

yarn start

1. Open your browser and navigate to http://localhost:3000.

**Step 6: Test the Authentication Providers**

1. When you open Backstage, you should see a **Sign In** page with all the configured providers (Google, Microsoft, GitHub, GitLab, Bitbucket, and Guest).
2. Test each provider by clicking on the respective button and completing the OAuth flow.
3. For guest access, click the **Guest** option to sign in without credentials.

**Example Code**

Here’s a summary of the key files with the changes:

**app-config.yaml**

auth:

environment: development

providers:

google:

development:

clientId: YOUR\_GOOGLE\_CLIENT\_ID

clientSecret: YOUR\_GOOGLE\_CLIENT\_SECRET

microsoft:

development:

clientId: YOUR\_MICROSOFT\_CLIENT\_ID

clientSecret: YOUR\_MICROSOFT\_CLIENT\_SECRET

tenantId: YOUR\_MICROSOFT\_TENANT\_ID

github:

development:

clientId: YOUR\_GITHUB\_CLIENT\_ID

clientSecret: YOUR\_GITHUB\_CLIENT\_SECRET

gitlab:

development:

clientId: YOUR\_GITLAB\_CLIENT\_ID

clientSecret: YOUR\_GITLAB\_CLIENT\_SECRET

baseUrl: https://gitlab.com

bitbucket:

development:

clientId: YOUR\_BITBUCKET\_CLIENT\_ID

clientSecret: YOUR\_BITBUCKET\_CLIENT\_SECRET

**packages/app/src/App.tsx**

import {

googleAuthApiRef,

microsoftAuthApiRef,

githubAuthApiRef,

gitlabAuthApiRef,

bitbucketAuthApiRef,

SignInPage,

} from '@backstage/core-plugin-api';

const app = createApp({

apis,

components: {

SignInPage: props => (

<SignInPage

{...props}

providers={[

{

id: 'google-auth-provider',

title: 'Google',

apiRef: googleAuthApiRef,

},

{

id: 'microsoft-auth-provider',

title: 'Microsoft',

apiRef: microsoftAuthApiRef,

},

{

id: 'github-auth-provider',

title: 'GitHub',

apiRef: githubAuthApiRef,

},

{

id: 'gitlab-auth-provider',

title: 'GitLab',

apiRef: gitlabAuthApiRef,

},

{

id: 'bitbucket-auth-provider',

title: 'Bitbucket',

apiRef: bitbucketAuthApiRef,

},

{

id: 'guest',

title: 'Guest',

apiRef: undefined,

},

]}

/>

),

},

});