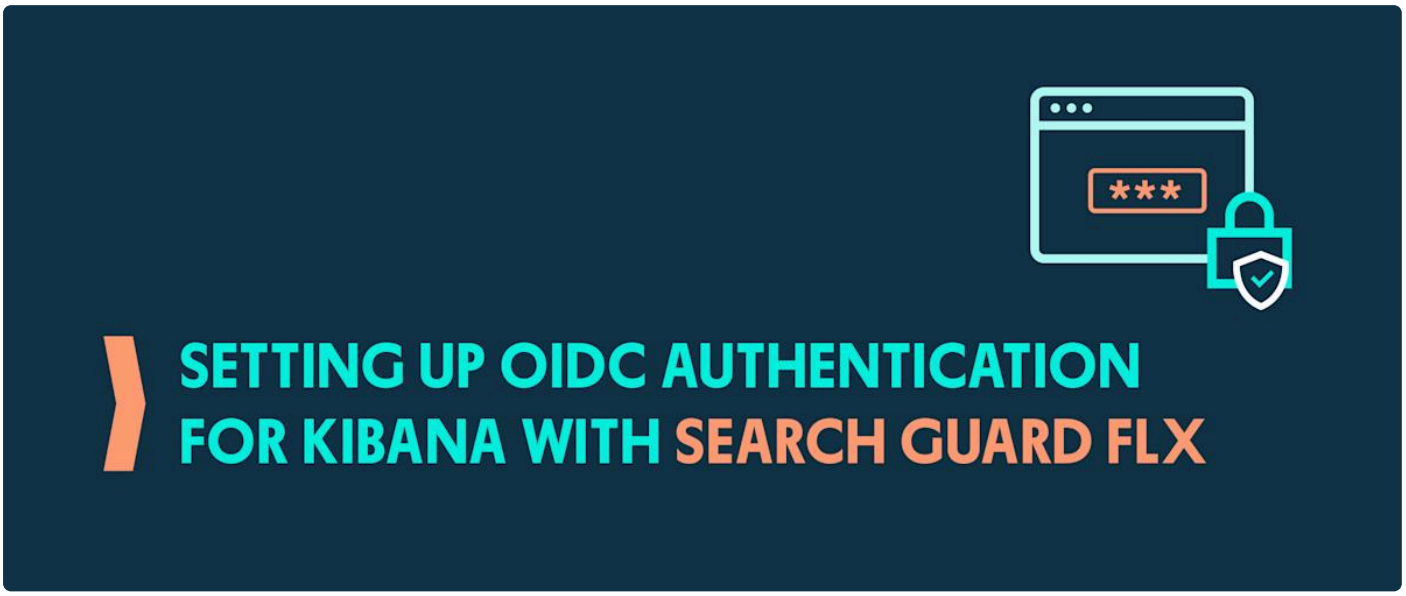
# Setting Up OIDC Authentication for Kibana with Search Guard FLX

Daniel Gleim · 2026-02-25




## Configure Keycloak Authentication for Kibana with Search Guard FLX

This guide explains how to configure **Keycloak** as an OpenID Connect (OIDC) provider for **Kibana** secured by the **Search Guard FLX** plugin. You will set up Keycloak, configure Search Guard for both **backend** (REST API) and **frontend** (Kibana SSO) authentication, map Keycloak roles to Search Guard roles, and test the integration.

### Prerequisites

- Elasticsearch cluster with the **Search Guard FLX** plugin installed
- Kibana with the matching Search Guard Kibana plugin version
- `sgctl` available for uploading Search Guard configuration
- A running **Keycloak** server
- Valid TLS certificates so Kibana and Elasticsearch can trust Keycloak

### Create a Keycloak Client for Kibana

- Log in to the **Keycloak Admin Console**.
- Choose or create a **realm** (e.g. `your-realm`).
- Go to **Clients → Create** and set:
  - **Client ID:** `my-kibana-client` (or another name)
  - **Client Type:** `openid-connect`

- **Client Authentication: ON** (confidential client)
- **Standard Flow Enabled: ON**
- **Set the Root URL to your Kibana URL:** (e.g., `https://kibana.example.com:5601/`)
- **Valid Redirect URIs:** `https://kibana.example.com:5601/auth/oidc/login` (replace with your public Kibana URL; avoid wildcards in production)

- Save and copy the **client secret** from the *Credentials* tab.

## Configure Search Guard Backend Authentication (`sg_authc.yml`)

The `sg_authc.yml` file defines REST API authentication for Elasticsearch. For Kibana, create a **basic authentication domain** for the internal server user, ensuring it can always connect even if OIDC is unavailable:

```
auth_domains:
  - type: basic/internal_users_db
```

Assign the server user the `SGS_KIBANA_SERVER` role. Upload the configuration using `sgctl`:

```
./sgctl.sh connect --cluster https://elasticsearch:9200 --user admin --password  <admin-password>
--ca-cert /path/to/ca.pem
./sgctl.sh update-config /path/to/sg_authc.yml
```

**Note:** Frontend SSO (OIDC/SAML) is configured separately in `sg_frontend_authc.yml`.

## Configure Kibana Frontend OIDC (`sg_frontend_authc.yml`)

End-user single sign-on is defined in **`sg_frontend_authc.yml`**.

```
# sg_frontend_authc.yml
default:
  auth_domains:
    - type: oidc
      id: keycloak_oidc
      label: "Keycloak SSO"

      oidc.client_id: "<my-kibana-client>"
      oidc.client_secret: "<client-secret-from-keycloak>"

      oidc.idp.openid_configuration_url: "https://keycloak.example.com/realms/your-realm/.well-known/openid-configuration"

      oidc.get_user_info: true
      oidc.use_pkce: true

      # Map roles based on how they appear in your token
      # If you added the optional mapper:
      user_mapping.roles.from_comma_separated_string: "oidc_id_token.roles"
      # Or map directly from default Keycloak paths:
      # user_mapping.roles.from: "oidc_id_token.realm_access.roles"
      # user_mapping.roles.from: "oidc_user_info.resource_access.kibana.roles"

    - type: basic
      label: "Local login (fallback)"
```

Upload:

```
./sgctl.sh update-config /path/to/sg_frontend_authc.yml
```

## Configure Kibana ( `kibana.yml` )

```
server.name: kibana
server.host: "0.0.0.0"
server.publicBaseUrl: "https://kibana.example.com:5601"

elasticsearch.hosts: ["https://elasticsearch:9200"]
elasticsearch.username: "kibanaserver"
elasticsearch.password: "<kibanaserver-password>"
elasticsearch.ssl.verificationMode: none
elasticsearch.requestHeadersWhitelist: ["Authorization", "sgtenant"]

searchguard.cookie.password: "<random-32+char-secret>"
searchguard.cookie.isSameSite: None
searchguard.cookie.secure: true
```

**Security tip:** Store passwords and secrets in the **Kibana keystore** or a secret manager instead of plain text.

## Map Roles ( `sg_roles_mapping.yml` )

```
sg_kibana_user:
  backend_roles:
    - "kibana_user"

sg_kibana_admin:
  backend_roles:
    - "kibana_admin"
```

The strings under `backend_roles` must match the values in the Keycloak token (for example the `roles` claim or the nested Keycloak paths you chose).

Upload:

```
./sgctl.sh update-config /path/to/sg_roles_mapping.yml
```

## Restart and Test

1. Restart Elasticsearch (if you updated its configs) and Kibana.
2. Browse to `https://kibana.example.com`.
3. You should be redirected to Keycloak.
4. Log in with a user who has the `kibana_user` or `kibana_admin` role.
5. You'll be redirected back to Kibana with the appropriate privileges.

## Troubleshooting Checklist

- Check Elasticsearch and Kibana logs for error messages
- Verify that all URLs and client credentials are correct
- Ensure that the Keycloak roles are correctly mapped to Search Guard roles
- Check that the Search Guard FLX plugin is properly installed and configured in both Elasticsearch and Kibana
- Check our documentation and Kibana Troubleshooting.

## Security Best Practices

- Always use **confidential clients** and store secrets securely.
- Limit valid redirect URIs to exact values—avoid wildcards.
- Keep TLS enabled and provide trusted CA certificates instead of disabling verification.
- Enable `oidc.use_pkce: true` for additional protection.
- Rotate client secrets regularly and monitor logs for authentication anomalies.

## Conclusion

You have successfully configured **Keycloak authentication for Kibana** using the **Search Guard FLX** security plugin:

- REST API auth (Kibana server user) lives in `sg_authc.yml`.
- Kibana single sign-on lives in `sg_frontend_authc.yml`.
- Keycloak roles map to Search Guard roles in `sg_roles_mapping.yml`.

This configuration provides a robust, single sign-on experience for your users while maintaining the security of your Elasticsearch cluster and Kibana instance.

Remember to keep your configurations and plugins updated, and regularly review your security settings to ensure ongoing protection of your data and applications.

---

This article was published on the Search Guard blog: https://search-guard.com/blog/kibana-openid-keycloak-flx/