SEARCH GUARD

CONFIGURATION BASICS

01.

CONFIGURATION TYPES

Static configuration

→ Stored in elasticsearch.yml

→ Changes require a node restart

→ All low-level security configuration is static

→ Shields against hiding attack evidence

Dynamic configuration

→ Stored in the Search Guard configuration index

→ Hot reloadable, changes take effect immediately

→ Changes require admin TLS certificate

02.

STATIC CONFIGURATION

- TLS configuration, REST and transport
- Node- and admin certificate configuration
- Audit logging configuration
- Compliance logging configuration
- REST API access permissions
- Special features

→ e.g., auto-initializing the Search Guard index

→ e.g., snapshot/restore privileges

EXAMPLE

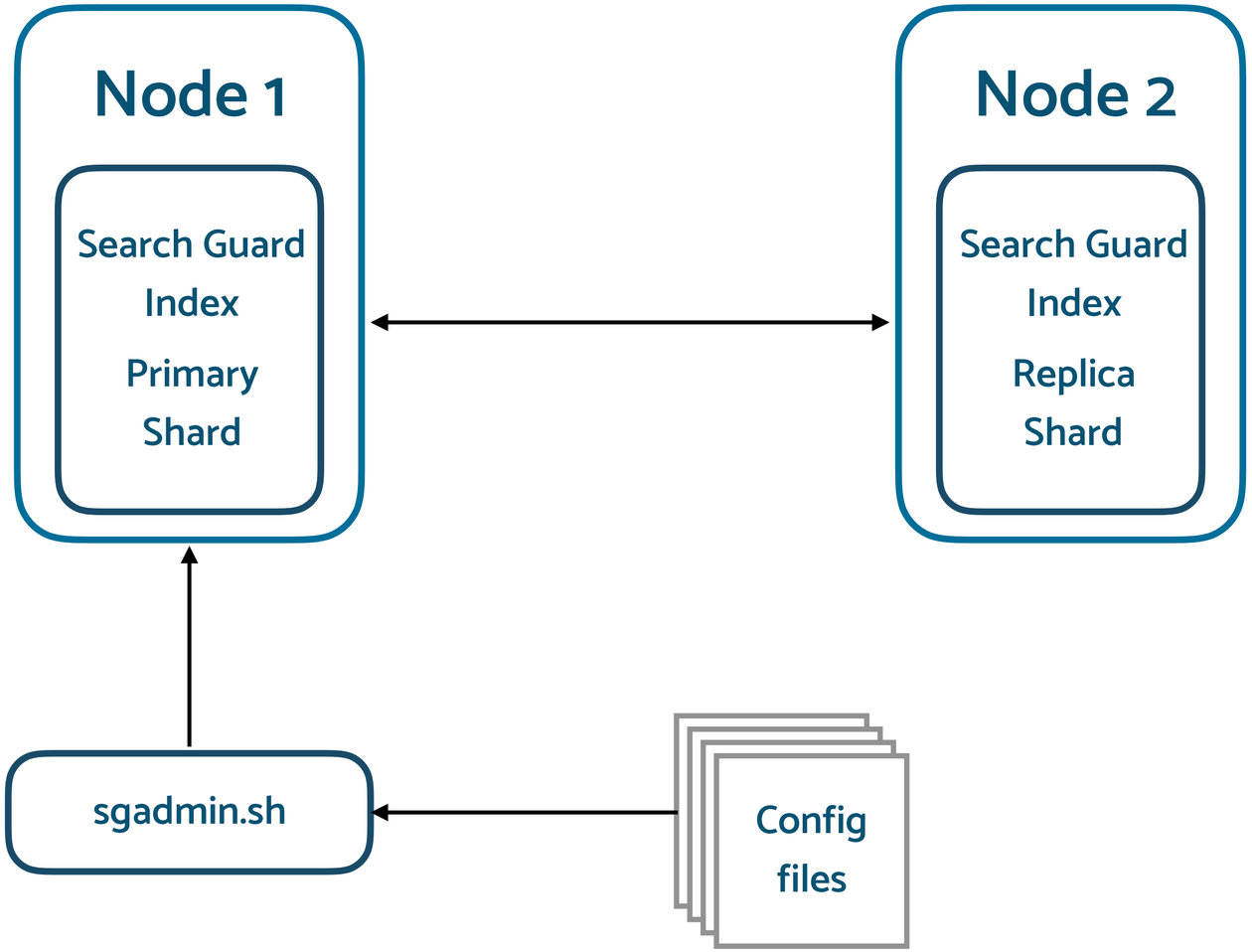
```
searchguard.ssl.transport.pemcert_filepath: esnode.pem
searchguard.ssl.transport.pemkey_filepath: esnode-key.pem
searchguard.ssl.transport.pemtrustedcas_filepath: root-ca.pem
searchguard.ssl.transport.enforce_hostname_verification: false

searchguard.ssl.http.enabled: true
searchguard.ssl.http.pemcert_filepath: esnode.pem
searchguard.ssl.http.pemkey_filepath: esnode-key.pem
searchguard.ssl.http.pemtrustedcas_filepath: root-ca.pem

searchguard.allow_default_init_sgindex: true

searchguard.authcz.admin_dn:
  - CN=kirk,OU=client,O=client,L=test, C=de
```

03.

DYNAMIC CONFIGURATION

- Authentication / Authorization: sg_config.yml
- Search Guard roles: sg_roles.yml
- Internal users: sg_intern_users.yml
- Role mapping: sg_roles_mapping.yml
- Action groups: sg_action_groups.yml

SEARCH GUARD CONFIGURATION INDEX



04.

AUTH/AUTH: SG_CONFIG.YML

- **Three main definitions**
- **How are user credentials passed to Search Guard?**

→ REST and/or transport

- **How should these credentials be verified?**

→ Authentication backend(s)

- **Should Search Guard fetch additional roles for the user?**

→ Authorization backend(s)

→ E.g., defined in a backend system like LDAP

STRUCTURE

```
searchguard:
  dynamic:
      <Proxy definitions, Kibana settings>
    authc:
      <authentication domain name>
        enabled: <true | false>
        order: <n>
        http_authenticator:
          ...
        authentication_backend:
           ...
    authz:
      <authorisation domain name>
       enabled: false
        authorization_backend:
          ...
```

EXAMPLE: BASIC AUTH / INTERNAL USERS

```
searchguard:
  dynamic:
    ...
    authc:
      basic_internal_auth_domain:
        enabled: true
        order: 0
        http_authenticator:
          type: basic
          challenge: true
        authentication_backend:
          type: intern
```

05.

INTERNAL_USERS: SG_INTERNAL_USERS.YML

- “Internal” authentication backend
- A simple list of users, roles and (hashed) passwords
- Uploaded to Search Guard index
- Can be used if no other authentication backend is available / needed
- Can be used for service users, like internal Kibana / Logstash server user

EXAMPLE

```
# The hash value is a bcrypt hash and can be generated with plugin/tools/hash.sh
admin:
  hash: $2a$12$VcCDgh2NDk07JGN0rjGbM.Ad41qVR/YFJcgHp0UGns5JDymv..TOG
  roles:
    - root
johndoe:
  hash: $2a$12$GI9JXffO3WUjTsU7Yy3E4.LBxC2ILo66Zg/rr79BpikSL2IIRezQa
roles:
    - manager
    - humanresources
janedoe:
  hash: $2a$12$xZOcnwYPYQ3zIadnIQIJ0eNhX1ngwMkTN.oMwkKxoGvDVPn4/6XtO
 roles:
    - manager
    - finance
```

06.

ROLES: SG_ROLES.YML

- **Defines permissions per role**
  - → On cluster level
  - → On index level
  - → On type level (deprecated - will be removed in Search Guard 7)
- **Defines document- and field-level security**
- **Defines anonymized fields**
- **Defines tenants for Kibana multi-tenancy**

STRUCTURE

```
<sg_role_name>:
  cluster:
    - '<permission>'
  indices:
    '<indexname or alias>':
      '<type>':
        - '<permission>'
      _dls_: '<querydsl query>'
      _fls_:
        - ‘<field>'
      _masked_fields_:
        - ‘<field>’
  tenants:
    <tenantname>: <RW|RO>
```

07.

CLUSTER LEVEL PERMISSIONS

- **For actions that**
  - → apply on cluster level, e.g., cluster health
  - → multi-search/get API: mget, msearch
  - → bulk API
- **Pre-defined action groups available**
  - → e.g., CLUSTER_MONITOR
  - → e.g., CLUSTER_COMPOSITE_OPS
  - → e.g., MANAGE_SNAPSHOTS

08.

INDEX LEVEL PERMISSIONS

- **For actions that**

→ affect one or more indices

- **Support for wildcards and regular expressions**
- **Support for index aliases**
- **Support for date/math based index names**
- **Dynamic index names (variable substitution)**
- **Alias handling**

→ Any index is resolved to concrete index/indices name(s)

→ Aliases and their real index names are treated equally

EXAMPLE

```
sg_role:
  cluster:
    - CLUSTER_COMPOSITE_OPS
  indices:
    'humanresources':
      '*':
        - CRUD
    ‘logs-*’:
      '*':
        - READ
    ‘kpi-{attr.ldap.department}’:
      '*':
        - READ
    ‘notes-{user.name}’:
      '*':
        - INDICES_ALL
```

09.

ROLES MAPPING

- **How are Search Guard roles assigned to a request?**
- **Map users, backend roles and hosts to Search Guard roles**

→ Username depends on the authentication process

→ E.g., username, DN of a client certificate, LDAP DN

- **Backend roles returned by authorization process, e.g.**

→ Active Directory Groups

→ JWT claims

→ Internal users backend roles

- **Wildcards and regular expressions supported**

10.

ROLES MAPPING

- **Using backend roles**

→ flexible, requires an authentication domain that supports the notion of roles/groups

→ Supported by: LDAP, AD, SAML, OpenID, JWT, Internal users, Proxy

→ Unsupported: Kerberos, PKI (but: combination possible, e.g., Kerberos / AD)

- **Using usernames**

→ Easier to set up, but less flexible

→ All new users must be added explicitly

- **Using hostnames**

→ E.g., allow everything from local machine

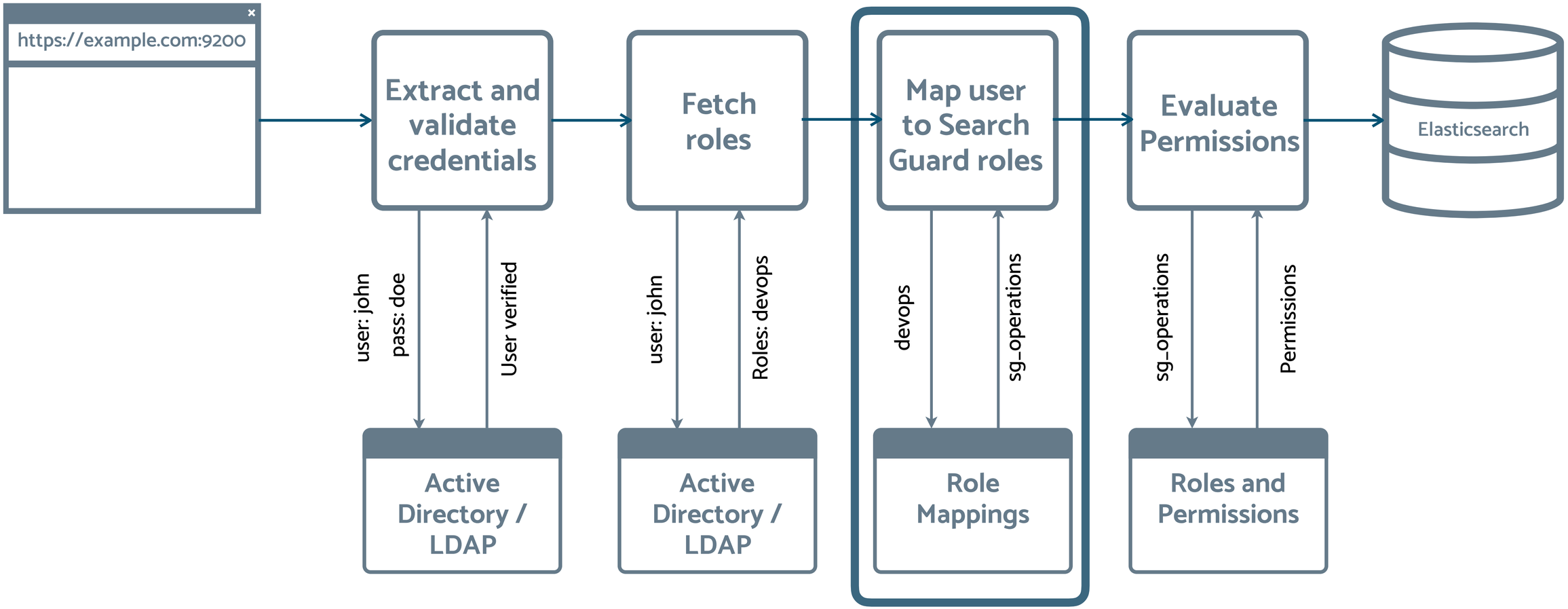
STRUCTURE

```
<sg_role_name>:
  users:
    - 'username'
    - 'username'
  backendroles:
     - 'backendrole'
     - 'backendrole'
  hosts:
     - 'hostname|IP'
     - 'hostname|IP'
```

EXAMPLE

```
sg_it_administrators:
  users:
    - ‘it_admin’
  backendroles:
     - ‘admins'
     - ‘it_admin_group’
  hosts:
     - ‘127.0.0.1’
```

ROLES MAPPING



11.

ELASTICSEARCH PERMISSIONS

- **Elasticsearch actions**

→ each Elasticsearch action (e.g. “search”)

→ has an assiated names (e.g. “indices:data/read/search”)

- **Internal Elasticsearch naming format**
- **Search Guard permissions are based on action names**

→ called “single permissions”

→ because they only apply to a single action

→ allows for fine-grained permission schemas

EXAMPLE

```
sg_role:
  cluster:
    - "indices:data/read/mget"
    - "indices:data/read/msearch"
    - "indices:data/read/mtv"
   - ...
  indices:
    'humanresources':
      '*':
        - "indices:data/read/search*"
        - “indices:data/read/suggest*"
        - ...
```

12.

ACTION GROUPS

- **Single permissions definitions can be verbose**
- **Permission definitions need to be repeated across roles**
- **Action groups define a bunch of permissions under a telling name**
- **Action groups for most use-cases shipped with Search Guard**

→ E.g., READ, WRITE, DELETE, CRUD

- **Users can define their own action groups**

→ Action groups can reference each other

→ Wildcards supported

EXAMPLE

```
CLUSTER_COMPOSITE_OPS:
   - "indices:data/write/bulk"
    - "indices:admin/aliases*"
WRITE:
  - "indices:data/write*"
READ:
  - "indices:data/read*"
CRUD:
  - READ
  - WRITE
```

USAGE IN ROLE DEFINITIONS

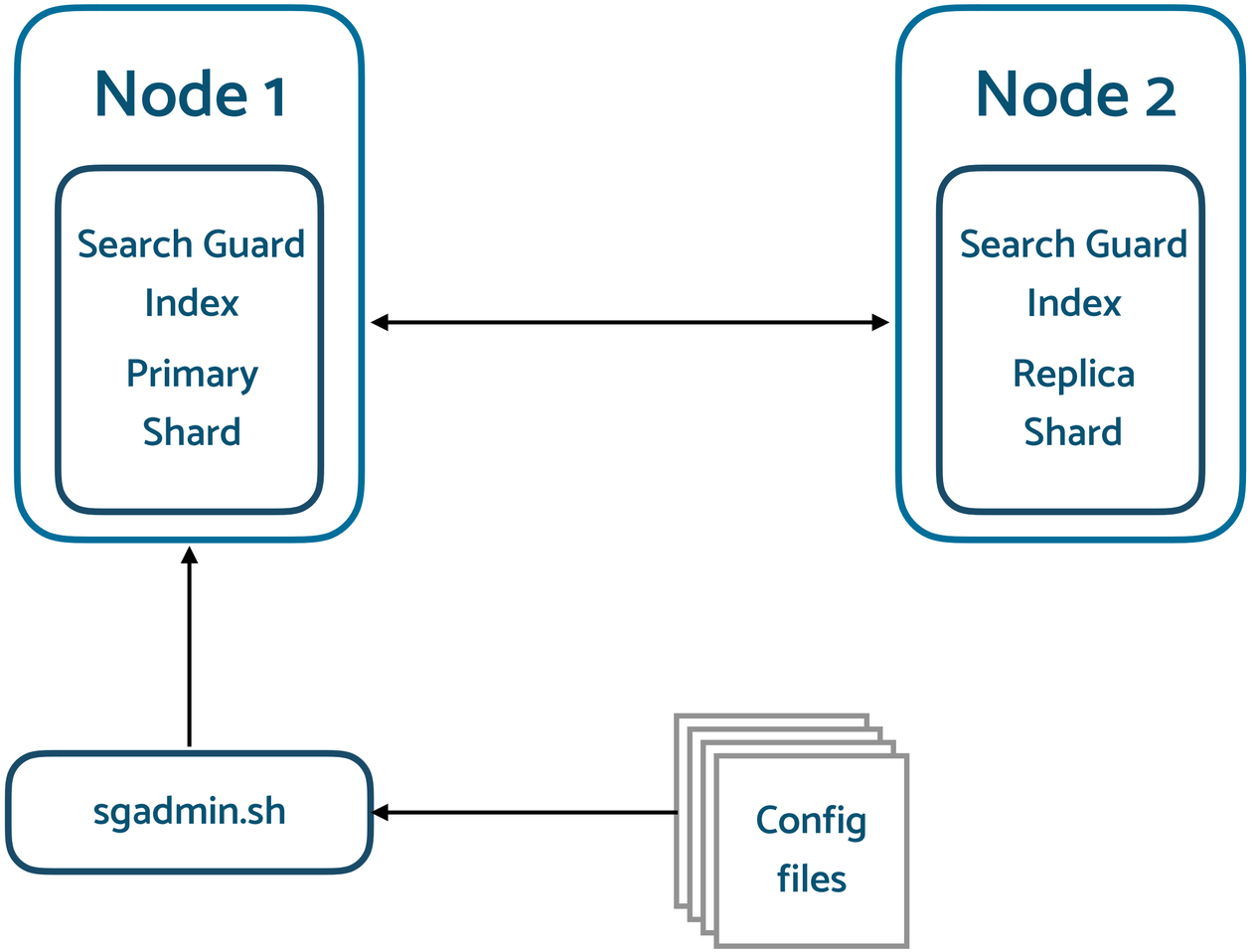
```
sg_role:
  cluster:
    - CLUSTER_COMPOSITE_OPS
   indices:
    'humanresources':
      '*':
        - CRUD
        - indicies:aliases*
```

13.

SEARCH GUARD CONFIGURATION INDEX

- **Search Guard configuration is stored in an Elasticsearch index**
  - → Hot-reloadable
  - → Changes take effect immediately
  - → No configuration files necessary on nodes
- **Accessible only with TLS admin certificate**
- **Configuration changes can be applied by**
  - → sgadmin command line tool
  - → REST API
  - → Kibana configuration GUI

SEARCH GUARD CONFIGURATION INDEX



14.

RESOURCES

- **Search Guard website**

→ https://search-guard.com/

- **Documentation**

→ https://docs.search-guard.com

- **Community Forum**

→ https://groups.google.com/d/forum/search-guard

- **GitHub**

→ https://github.com/floragunncom

SEARCH GUARD

SEND US A MESSAGE:

info@search-guard.com

Search Guard

floragunn GmbH
Tempelhofer Ufer 16
D-10963 Berlin, Germany

E-Mail: info@search-guard.com
Web: search-guard.com

Managing Directors: Claudia Kressin, Jochen Kressin
Registergericht: Amtsgericht Charlottenburg
Registernummer: HRB 147010 B
E-Mail: info@floragunn.com