Cloning a Realm to a New Cluster or Region

In scenarios where high availability, regional redundancy, or environment separation (e.g., staging to production) is required, cloning an entire Keycloak realm to a new cluster or region becomes essential. This process involves exporting the realm’s configuration and optionally user data, transferring it securely, and importing it into a fresh Keycloak instance. This guide covers all the required steps, including command-line tooling, configuration handling, and validation checks to ensure a seamless realm replication.

Pre-Cloning Preparation

Before initiating the cloning process, ensure that both the source and target Keycloak clusters are accessible and running compatible versions of Keycloak. This avoids schema mismatches and import errors. Install the Keycloak Admin CLI (kcadm.sh) and Keycloak Configuration CLI (keycloak-config-cli) on your local system or CI/CD pipeline.

Deploy a new Keycloak instance in the target cluster or region. This can be done using Docker, Kubernetes, or a managed hosting provider. Example Docker command to spin up a development instance:

```
docker run -d --name keycloak \
  -p 8080:8080 \
  -e KEYCLOAK_ADMIN=admin \
  -e KEYCLOAK_ADMIN_PASSWORD=admin \
  quay.io/keycloak/keycloak:24.0.3 \
  start-dev
```

Ensure that network connectivity exists between your machine and the target Keycloak instance. Also, create an admin user for the new instance.

Exporting Realm Configuration from the Source Cluster

To begin the cloning process, export the realm’s full configuration (including clients, roles, groups, and optionally users) using the Keycloak Admin CLI or built-in export tools. If using the Keycloak start command with --export flag, execute:

```
/opt/keycloak/bin/kc.sh export --dir /opt/keycloak/data/export \
  --realm myrealm \
  --users skip
```

To include users in the export:

```
/opt/keycloak/bin/kc.sh export --dir /opt/keycloak/data/export \
  --realm myrealm \
  --users all
```

If the Keycloak instance is running in Docker:

```
docker exec -it keycloak /opt/keycloak/bin/kc.sh export \
  --dir /opt/keycloak/data/export \
  --realm myrealm \
  --users all
```

The exported realm will be saved as a JSON file, e.g., myrealm-realm.json.

Transferring Exported Data

Once exported, securely transfer the generated realm export directory or file (myrealm-realm.json) to the target cluster or region. Depending on your infrastructure, use one of the following methods:

- SCP or SFTP for VM-to-VM transfer:

```
scp myrealm-realm.json user@target-host:/tmp/
```

- AWS S3, Azure Blob Storage, or GCS for multi-cloud environments.
- Git repositories or artifact registries for CI/CD pipelines.

Ensure that the target Keycloak container or pod has access to the file location.

Importing Realm into the Target Cluster

To import the realm into the new Keycloak instance, use the same kc.sh tool on the target side. The import must be triggered before the Keycloak server starts. If using a container:

```
docker run -d --name keycloak-new \
  -p 8080:8080 \
  -e KEYCLOAK_ADMIN=admin \
```

```
  -e KEYCLOAK_ADMIN_PASSWORD=admin \
  -v $(pwd)/export:/opt/keycloak/data/import \
  quay.io/keycloak/keycloak:24.0.3 \
  start-dev --import-realm
```

This command will initialize the new Keycloak instance with the cloned realm, including users if they were part of the original export.

Alternatively, if the server is already running, use kcadm.sh or the keycloak-config-cli for post-start import. The keycloak-config-cli is suitable for GitOps-style deployments:

```
docker run --rm \
  -e KEYCLOAK_URL=http://localhost:8080 \
  -e KEYCLOAK_USER=admin \
  -e KEYCLOAK_PASSWORD=admin \
  -v "$(pwd)/myrealm:/config" \
  adorsys/keycloak-config-cli:latest
```

Post-Cloning Validation

After the import is complete, validate the integrity of the cloned realm with the following checks:

- Confirm that the new realm appears in the admin console and is accessible at /realms/myrealm.
- Inspect clients to verify that client IDs, redirect URIs, and secrets have been preserved.
- Validate that roles, groups, and permissions are correctly replicated.
- Test login using a few sample user accounts if users were exported.
- Decode access tokens to confirm the correctness of claims, issuer, and audience.
- Check identity provider connections (e.g., Google, LDAP) and test federated logins.
- Enable auditing under **Events > Settings** to monitor realm activity in the new instance.
- Update baseUrl settings for clients if moving across DNS regions.
- Ensure SMTP settings, themes, and custom scripts are present and functioning.
- Enable TLS and update public frontend URLs if applicable.
- Verify realm-specific settings like session timeout, brute force detection, and required actions.

Revision #2
Created 2025-06-16 06:56:05 UTC by kaiwalya
Updated 2025-06-16 06:57:48 UTC by kaiwalya