Ionic Reference Guide

Privitar Data Privacy Platform, version 4.5.0

Publication date September 6, 2023

Privitar Data Privacy Platform, version 4.5.0

Table of Contents

1. Introduction

The Privitar Data Privacy Platform is a data privacy solution that enables organizations to use sensitive datasets more safely.

> **Note**
>
> For ease of reference, the Privitar Data Privacy Platform is referred to as the **Privitar PlatformPrivacy Platform**, (or just **the platform**) in the rest of this manual.

Ionic Machina provides a key management system (KMS) and an access policy control software solution.

Here is some new content.

The Ionic SDK provides methods to generate a key(s) as well as for encrypting data.

The platform uses a Key Management Service (KMS) to access encryption keys used by:

- Encrypt Rule
- Derived Tokenisation
- HDFS Token Vault Encryption
- Batch jobs using JDBC connections that require secured credentials

Ionic Machina can be integrated with the platform to provide this KMS functionality.

1.1. Compatibility

Privitar v4.1 provides support for Ionic Machina. It is compatible with the Ionic Password Persistor and is tested with version 2.8.0 of the SDK.

2. Requirements

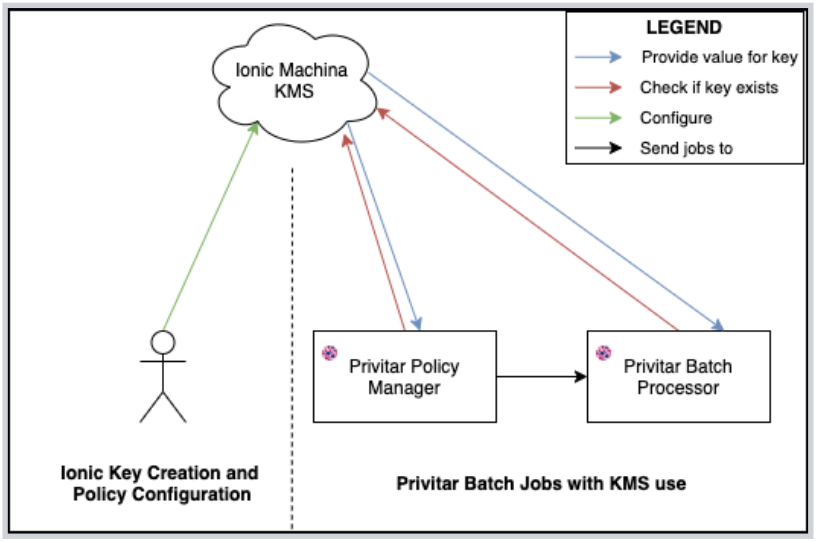
In order to integrate Ionic Machina with the Privacy Platform, the following is needed:

- The Ionic client (Ionic CLI) must be installed. This is required when enrolling devices with Ionic and generating profiles.
- The keys used by the platform must exist in Ionic.
- The appropriate policy must exist within Ionic to allow the platform to access relevant keys.

3. Architecture

The platform and cluster nodes each independently need access to the Ionic service. This is because no sensitive information is transferred to the cluster during job submission. The only information relating to Ionic, that is submitted, is the location of Ionic profiles, and the name of the environment variable storing the password.

3.1. Implementation flow



1. When the platform encounters a situation whereby secret key material is required, it checks what is configured in the Job's Policy or Environment to find the name of the key to be used.

   To maintain consistency, a PDD will always use the same version of the key.
2. The platform creates a Spark job definition and submits this to the cluster. No sensitive information is sent as part of the job submission process. The relevant information sent as part of this request is:

   a. The path of the Ionic Profile Persistor file, which must be common across all nodes
   b. The name of the environment variable containing the Persistor's password.
   c. The name and ID of the correct version of each key to be used in the job.

   In the case of *JDBC credentials*, the job definition only contains the key name (and not the ID of a particular version) of the key to be used. Both the platform node and the Spark processing nodes will request the latest version of the key from the KMS.
3. When a key is required by the Job, the spark processing nodes use the same authentication process as in step 2 using their own local Ionic Profile Persistor and `IONIC_PASSWORD` environment variable to access the KMS.

Note

The Privitar Policy Manager application reads the `IONIC_PASSWORD` environment variable containing the password for the Ionic Profile at startup time only. It does not require continued access to the environment variable after startup and is available in memory within the application. Whilst the application is running, the environment variable is no longer required to be set until the application undergoes a restart.

4. Ionic Integration Overview

This section presents an overview of the steps to follow to integrate the Privacy Platform with Ionic Machina. For each step described, a link is provided to the appropriate section in the document that describes the procedure to follow to complete that step.

The table below describes the integration process in terms of the areas of the platform that need to be integrated with Ionic.

Step	Privitar object	Ionic object	Description
1		Application	Execute an *Enrollment* in Ionic Machina to create the common device profile persistor / password files See, Create a profile.
2		Application	Deploy the profile persistor file and password to all processing nodes in your cluster See, Deploying the persistor and password.
3	Environment		Ensure Ionic SDK files are loaded into the platform Hadoop Jars Path See, Loading Ionic SDK files into Privitar.
4	Environment		Set your Cluster in the platform to use the Ionic Keystore See, Configuring the Environment.
5	Environment		Define Service User(s) in the platform for the Cluster based Environment to support 'on-behalf-of' job processing See,.Creating Service User in Privitar.
6		Application	Add the Service User(s) in Ionic. See, Creating Service User in Ionic Machina.
7		Application	Create Project /Team specific keys See, Creating Keys.

4.1. Create a profile

To enable the Privacy Platform to access the Ionic system, a *profile* (or *device credentials*) needs to be created for the platform in Ionic. A profile acts in a similar way to a user account and once set up it enables the platform to access the Ionic system.

Profiles are stored in encrypted files called *persistors* (or *profile persistors*). They store the device credentials required to connect a new device (such as the Privacy Platform) to the Ionic system. There are different types of persistors that can be created in Ionic. The Privacy Platform supports *password persistors*. This is a password protected file that can store a profile or a group of profiles - in effect one or more sets of device credentials.

The process of creating a profile in Ionic for a new device is called *enrollment*. For more information about enrolling a new device in Ionic, refer to `https://dev.ionic.com/getting-started/create-ionic-profile`.

Example 1. Example of profile creation with Ionic Machina

The following command will create a new password persistor, called "profiles.pw". This file will be password protected with the value "myPassword". This profile is created for email "my_email@domain.com", and uses keyspace identified "Keyspace_Id".

You can find more details about this command here: `https://dev.ionic.com/tools/machina/profile_enroll`

```
machina \
    --devicetype password \
    --devicepw myPassword \
    --devicefile $HOME/.ionicsecurity/profiles.pw \
    profile enroll \
    –keyspace Keyspace_Id \
    --email my_email@domain.com \
    --type idc
```

Following the creation of the profile, you should have two items:

- A password persistor file containing the profile for the Privacy Platform. The default location is `$HOME/.ionicsecurity/profiles.pw`.
- A password for the persistor.

4.2. Deploying the persistor and password

Once the credentials have been created, they need to be shared with the different environments that will need to access the Ionic Machina keystore.

4.2.1. Deploying the persistor file

The persistor must be deployed in the same path, with the same name, to the different processing locations (default: `~/.ionicsecurity/profiles.pw`). This includes:

- The machine that hosts the Privacy Platform
- Each node where the the platform jobs will run. For example, it must be installed on each Hadoop cluster data node, where Privacy Platform Spark jobs will run

4.2.2. Deploying the persistor password

The persistor password must be set in an environment variable. By default, it is called `IONIC_PASSWORD`.

Example 2. Setting the environment variable for the persistor password

```
export IONIC_PASSWORD=<password>
```

The variable will need to exist in the same environments as the persistor files.

Application Properties For Ionic Password

Privacy Platform configuration (`application.properties`) could be modified to specify a different variable name.

Name	Description
`agrotera.ionic.password.variable`	The name of an environment variable that contains the password to decrypt Ionic persistors. This will override default environment variable `IONIC_PASSWORD` .

4.3. Loading Ionic SDK files into Privitar

So the Privacy Platform can communicate with Ionic Machina, the Ionic Machina SDK, available via the Machina Tools, must be installed in the platform machine.

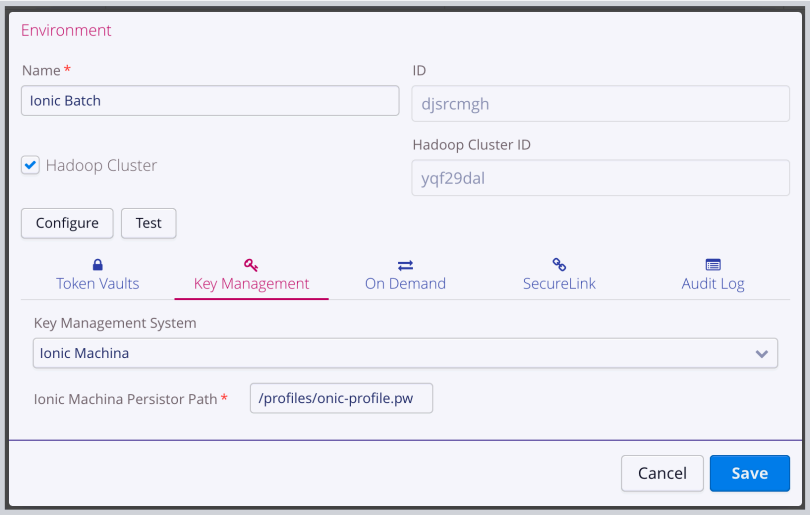
Follow these steps to install them:

1. On Privacy Platform, log in as *Superuser*.
2. Select **Cluster Types** from the navigation sidebar.
3. Locate the cluster that will use the Ionic Machina KMS feature, and click on the **Edit** button on the same row.
4. Take note of the value of **Hadoop Jars Path**.
5. Copy the *Ionic SDK* and *javax.json* from Ionic Machina Tools to the location found in the previous step.

4.4. Configuring the Environment

In order for the Privacy Platform environments to use the Ionic Machina KMS feature, follow these steps:

1. Select **Environments** from the sidebar.
2. Click on the environment that will use the Ionic Machina KMS Feature.
3. Select the **Key Management** tab.
4. Use the **Key Management System** dropdown to select *Ionic Machina*.
5. Set the value of field **Ionic Machina Persistor Path** to the file path defined in Deploying the persistor file.



4.5. Creating Service User in Privitar

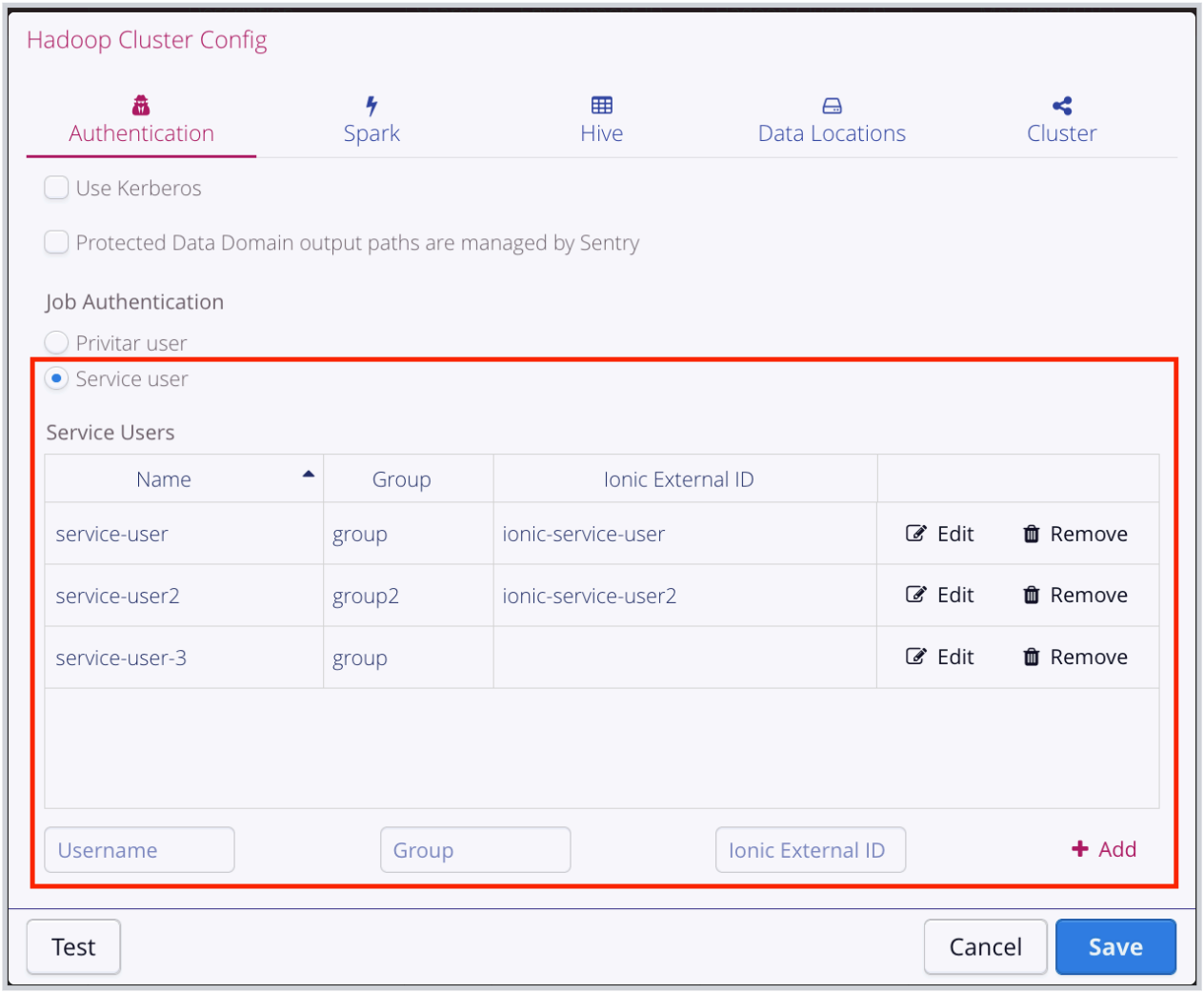
If impersonation is needed, a Privitar Service User is required for delegated requests. It needs to exist for all the Privacy Platformenvironments that will use Ionic Machina KMS. This account will be tied to the Hadoop impersonation.

To create a new *Service User* follow these steps:

1. Select **Environments**, from the navigation sidebar.
2. Click on the environment to edit.
3. Click on the **Configure** button, under the **Hadoop Cluster** check box.
4. Under the **Authentication** tab, select **Service user**.
5. Enter a **username** and a **group**, then click **Add**.
6. Click on **Save**, then **Save** again.

Note

When using service users, requests to Ionic Machina KMS default to acting on behalf of the same username as specified within Hadoop. If the username within Ionic does not match the Hadoop username, you can additionally specify an **External ID** to enable a mapping between these usernames.



4.6. Creating Service User in Ionic Machina

Please refer to your Ionic documentation to create a service user matching the one from Creating Service User in Privitar.

4.7. Creating Keys

In order to use KMS with the Privacy Platform, keys must be created in Ionic.

To proceed, follow these steps:

1. Log in with the user ID of the persistor profile, create Keys adding the Privitar 'key name' as an 'ionic_external_id' attribute. Use the Create Keys with External ID approach (`https://dev.ionic.com/sdk/tasks/create-key-with-external-id`)
2. Create a policy so the Service User configured in Creating Service User in Ionic Machina can access the Ionic keys. You can refer to the Ionic documentation to proceed: `https://dev.ionic.com/tutorials/policy/create-data-policy`

Example 3. Impersonation example

If we want a user named `privitar` to impersonate an account named `service-user` to access a key named `key`, the policy must be configured with:

- `service-user` is allowed to access `key`.
- `privitar` is allowed to access `key` when `ionic-delegated-external-id` is `service-user`.

5. Configuring Privitar to use Ionic Machina KMS

Encrypt rule, HDFS token vaults, and JDBC credentials can be configured to use keys generated with Ionic Machina.

5.1. Encrypt Rule

To use the keys defined in the Ionic environment with an encrypt rule, follow these steps:

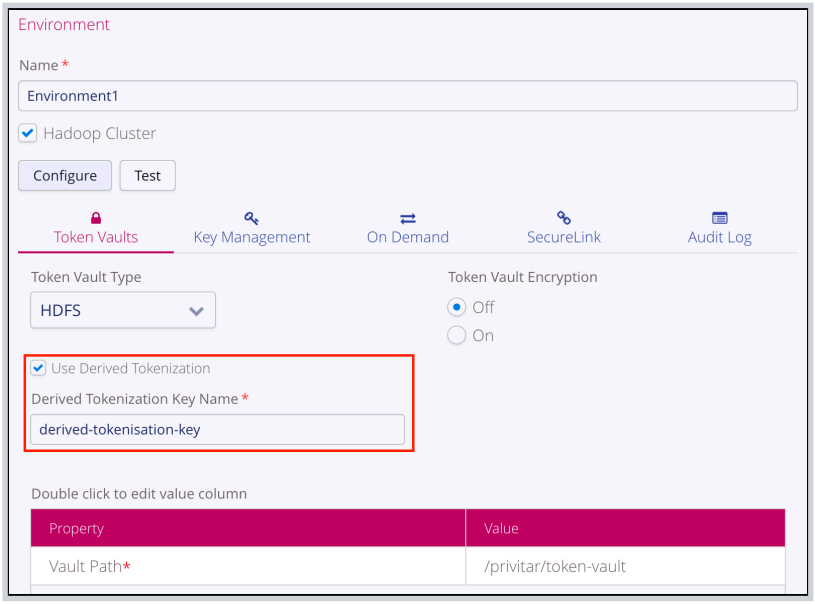
1. Select **Policies** from the navigation sidebar.
2. On the **Policies** page, click on **Rules**.
3. Click on **Create New Rule**.
4. Enter a **name** for the new rule.
5. On the **Add masking rule** page, select **Encrypt** as the **Mask Type**.
6. Under **Key Name**, enter the key defined in Ionic.
7. Click on **Save** to close the **Add masking rule** dialog box.

5.2. Derived Tokenisation

For HDFS Token Vault with KMS configured, it is possible to use an Ionic key as the derived tokenization value (See, *Token Vault Environment Configuration*, in the *Privitar User Guide* for further details about this feature).

To use a key defined in Ionic, follow these steps:

1. From the navigation sidebar, select **Environments**.
2. On the **Environment** page, click on the environment to edit.
3. Select the **Token Vaults** tab.
4. Tick **Use Derived Tokenisation** check box. Note that is only visible for HDFS token vaults.
5. Set the **Derived Tokenization Key Name**, field to the Ionic key to use.
6. Click on **Save**.

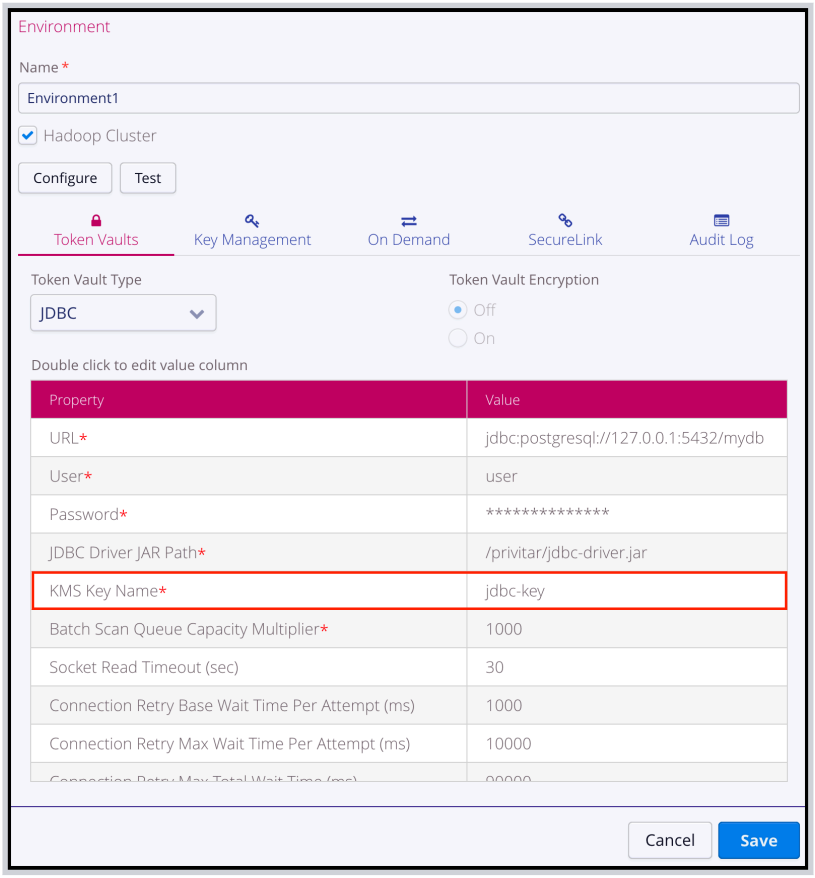


5.3. JDBC Credentials

The password of a JDBC Token Vault connection can be encrypted using an Ionic key (See, *Token Vault Environment Configuration*, in the *Privitar User Guide* for further details about this feature).

To use a key defined in Ionic, follow these steps:

1. Select **Environments** from the navigation sidebar.
2. On the **Environment** page, click on the environment to edit.
3. Select the **Token Vaults** tab.
4. Double click on the value cell in front of **KMS Key Name** .
5. Enter the Ionic key to use, then click on **Save**.
6. Click on **Save** to close the **Environment** page.

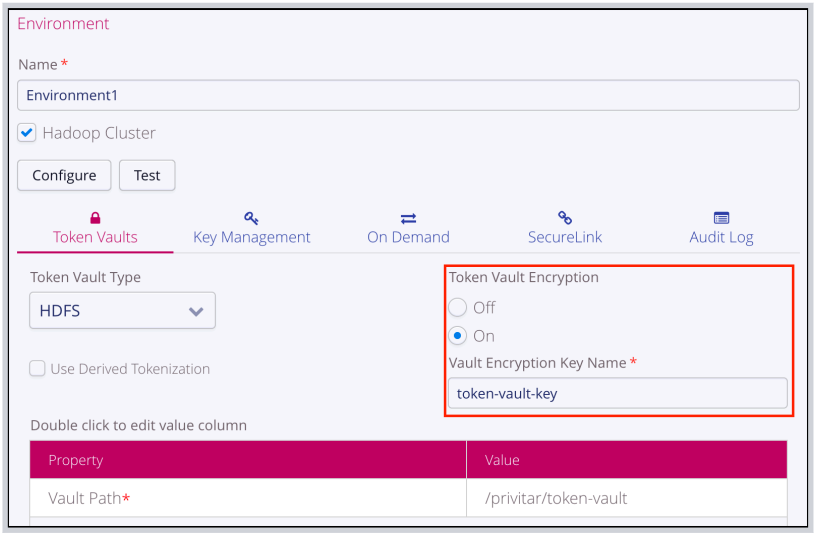


5.4. Encrypted HDFS Token Vault

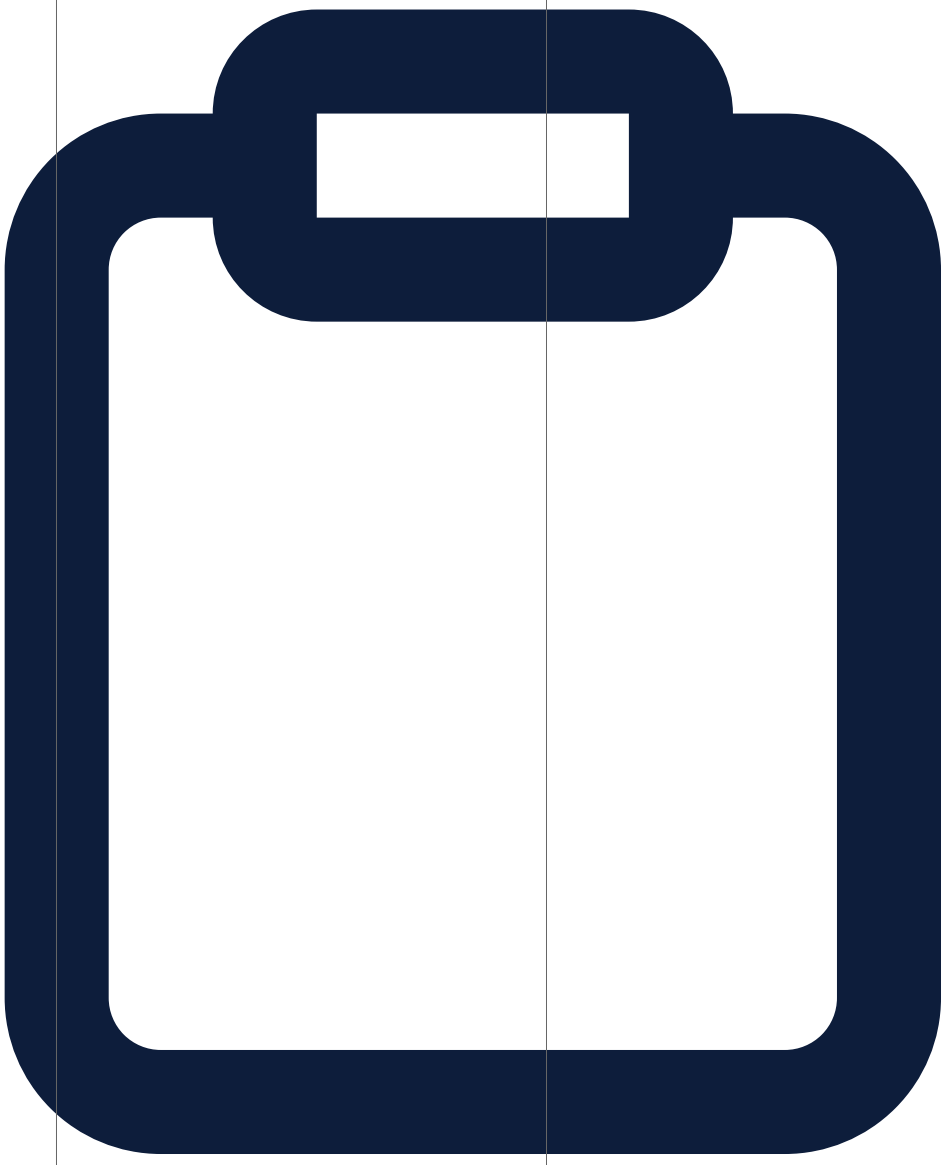
For HDFS Token Vault with KMS configured, it is possible to use an Ionic key as the derived tokenization value (See, *Token Vault Environment Configuration*, in the *Privitar User Guide* for further details about this feature).

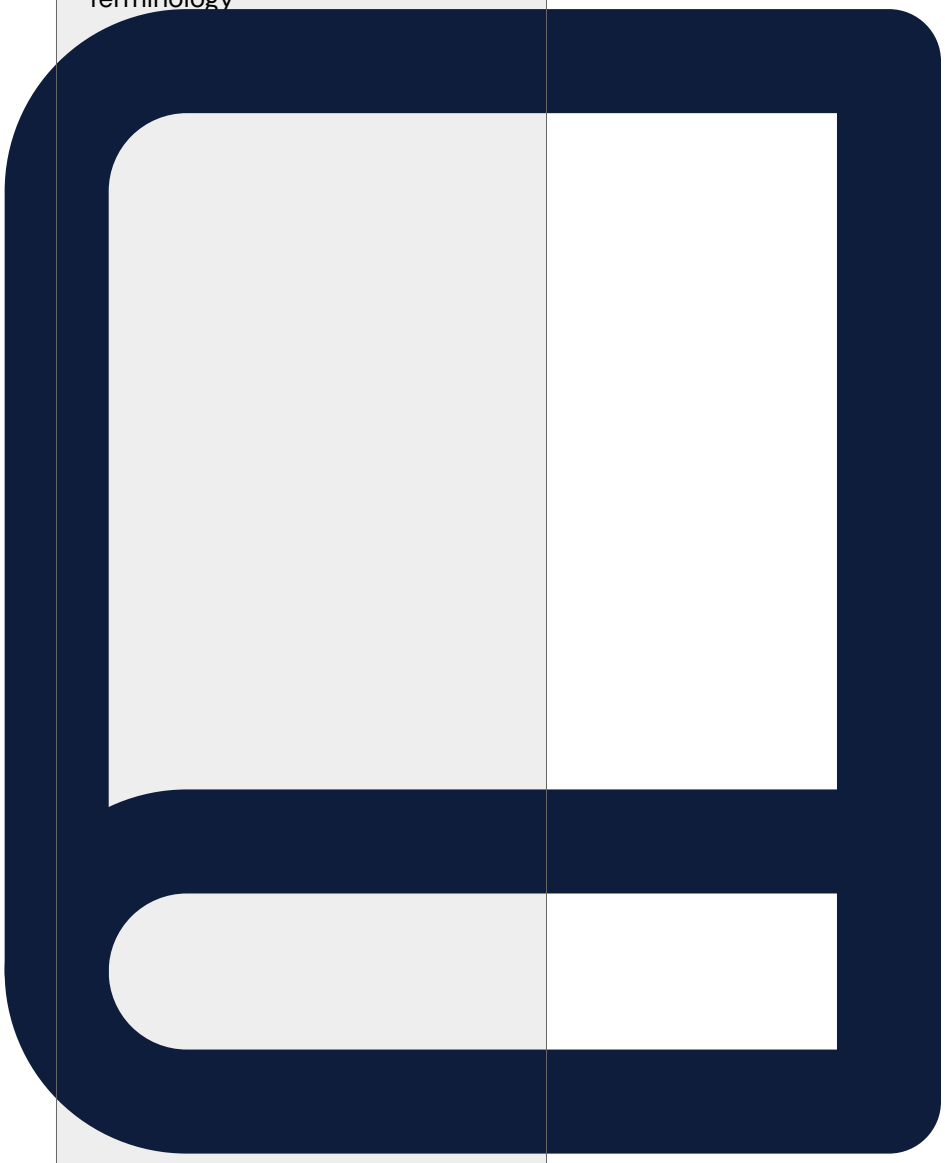
To use a key defined in Ionic, follow these steps:

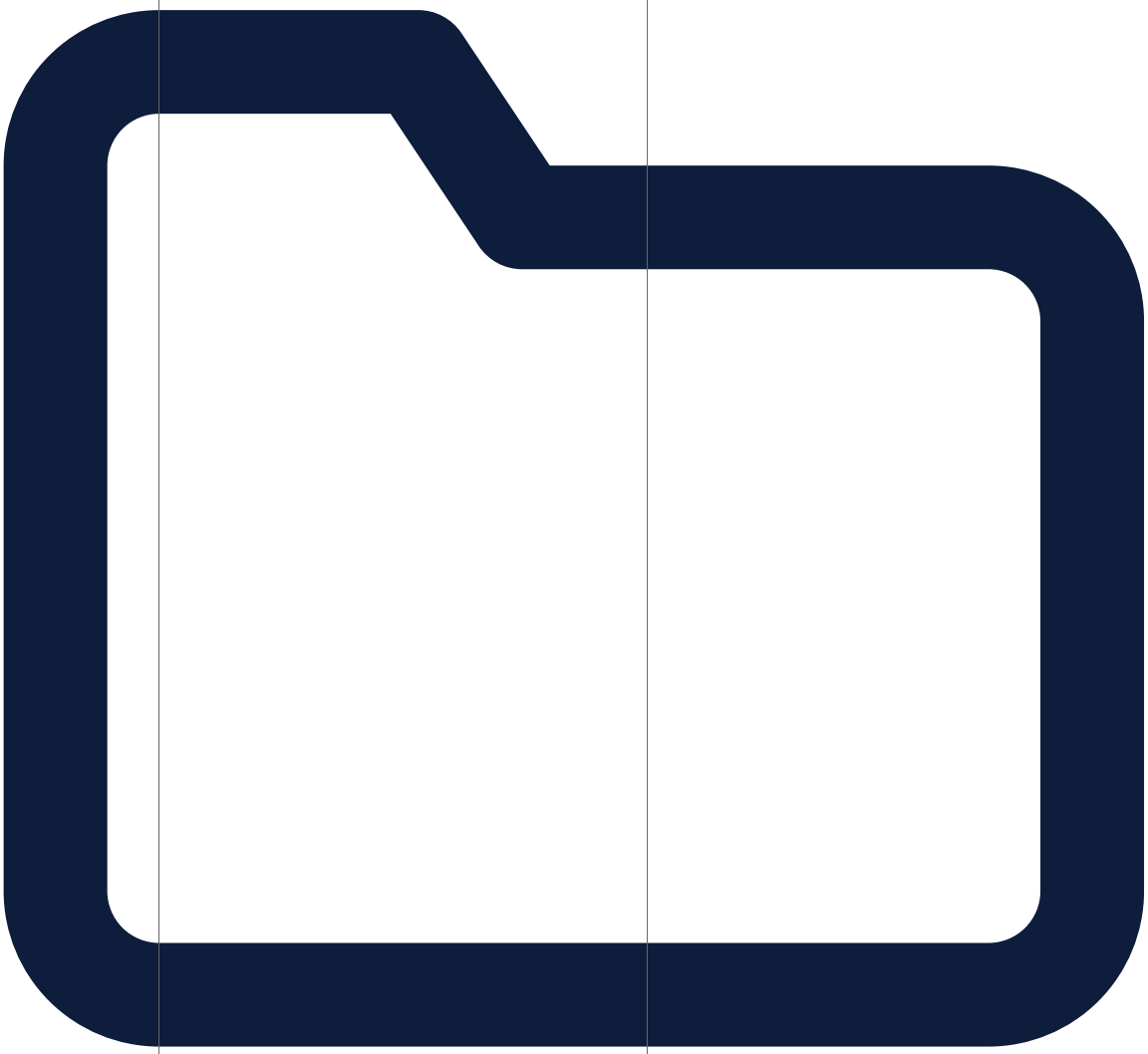
1. Select **Environments** from the navigation sidebar.
2. On the **Environment** page, click on the environment to edit.
3. Select the **Token Vaults** tab.
4. Set **Token Vault Encryption** to **On**.
5. Set the **Vault Encryption Key Name**, field to the Ionic key to use.
6. Click on **Save** to close the **Environment** page.


Environment
Name *
Environment1
Hadoop Cluster
Configure
Test
Token Vaults
Key Management
On Demand
SecureLink
Audit Log
Token Vault Type
HDFS
Use Derived Tokenization
Token Vault Encryption
Off
On
Vault Encryption Key Name *
token-vault-key
Double click to edit value column
Property
Value
Vault Path*
/privitar/token-vault

Icons for Portal Landing Page (not included in TOC)

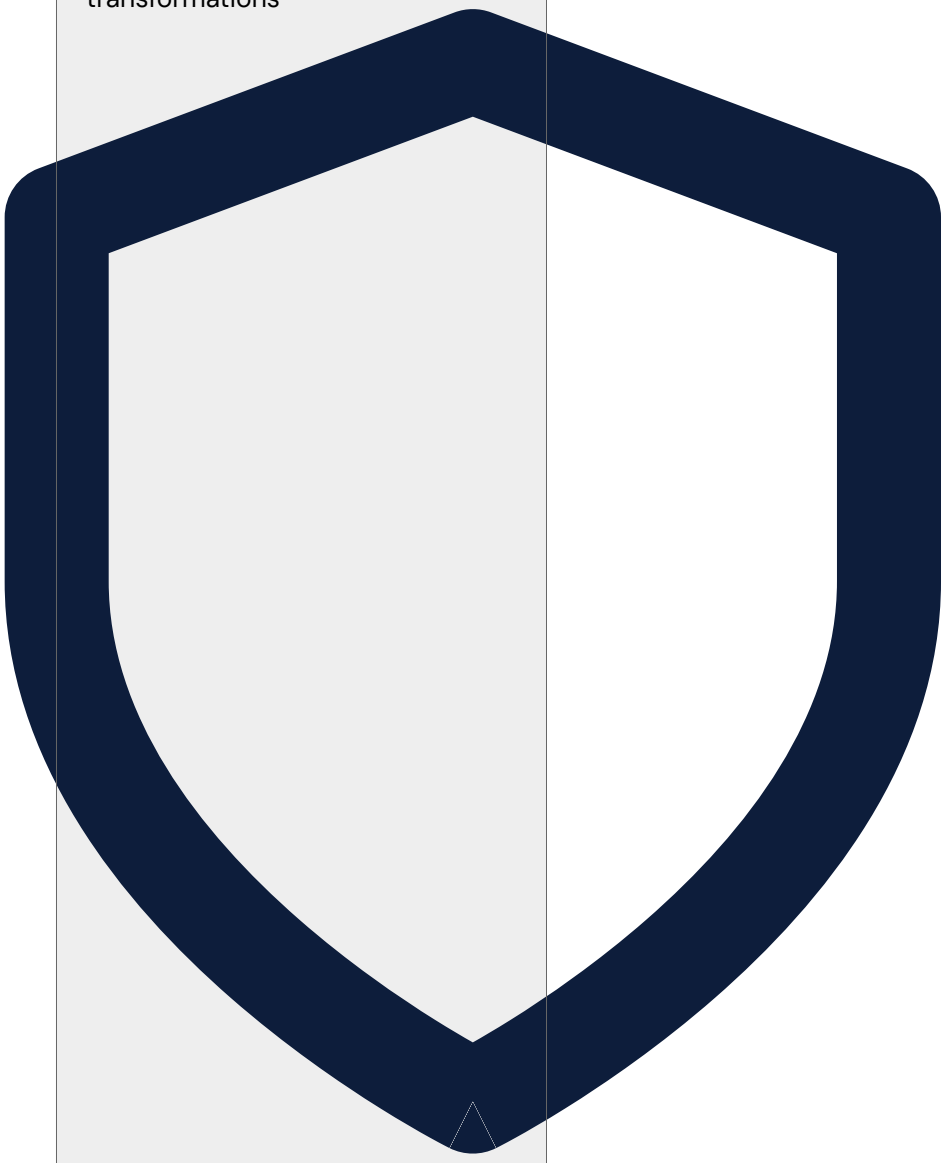
Description	Icon (76px)
Approving requests	

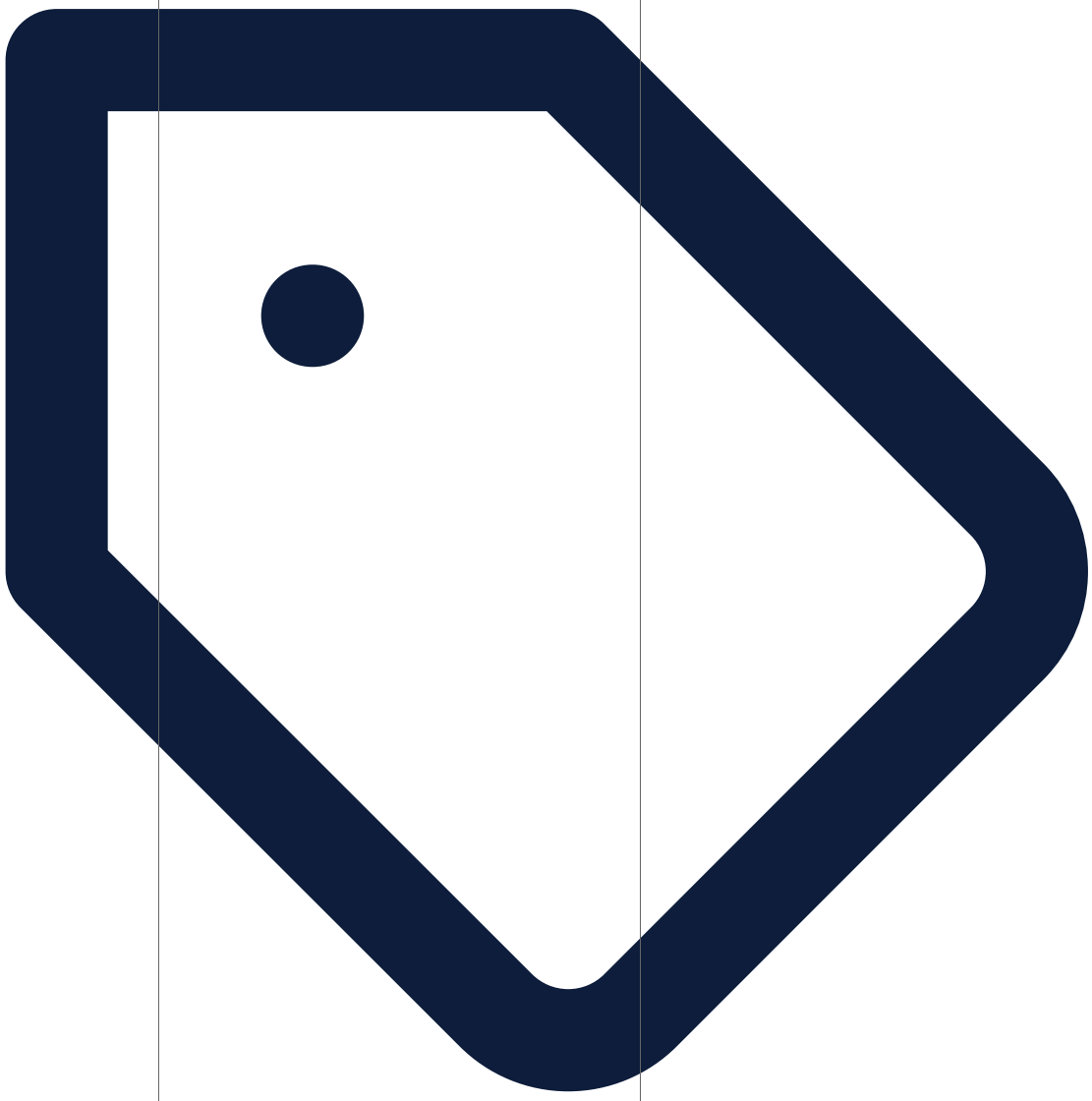
Description	Icon (76px)
Glossary of Terminology	

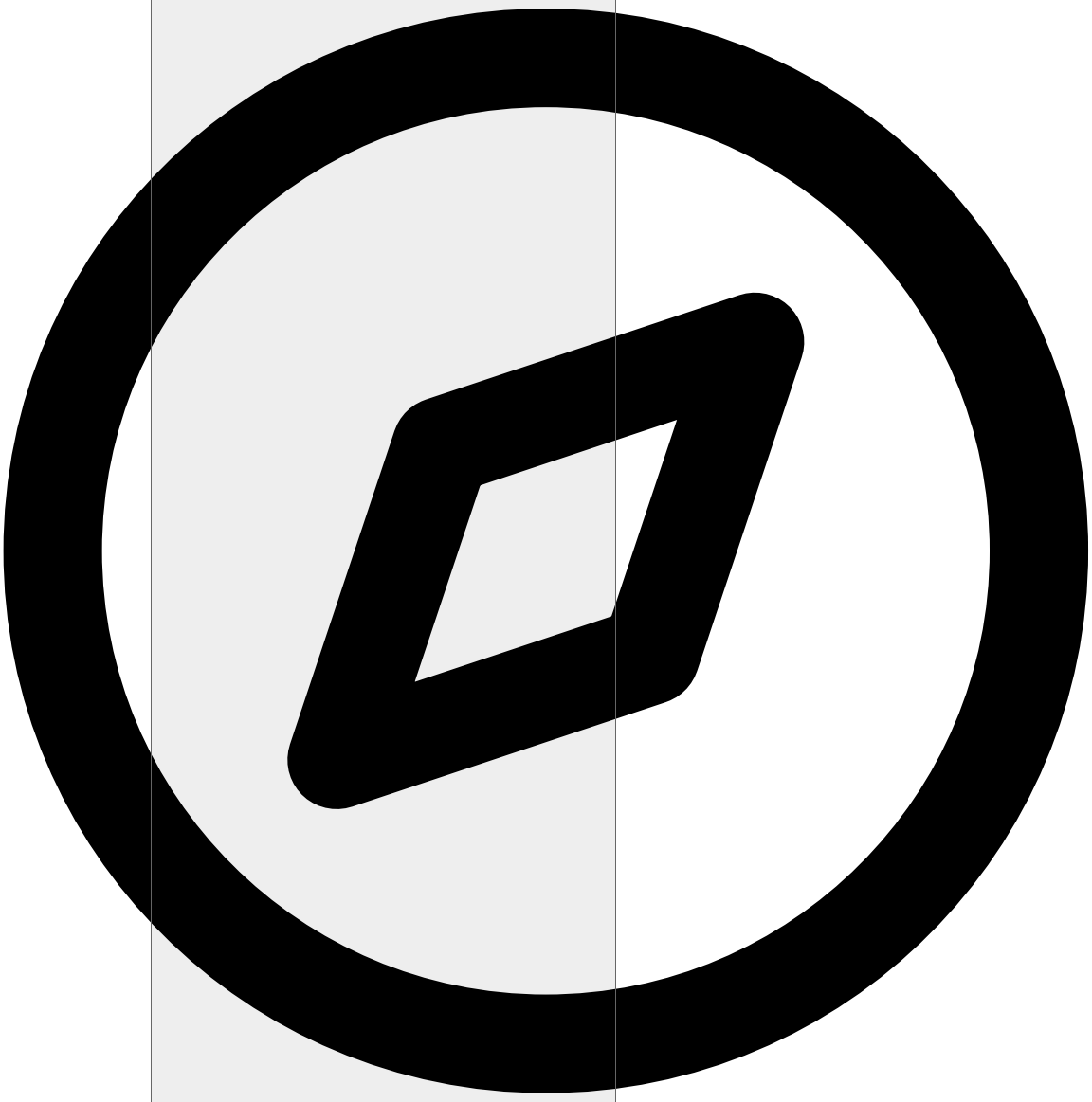
Description	Icon (76px)
Searching for and accessing data	

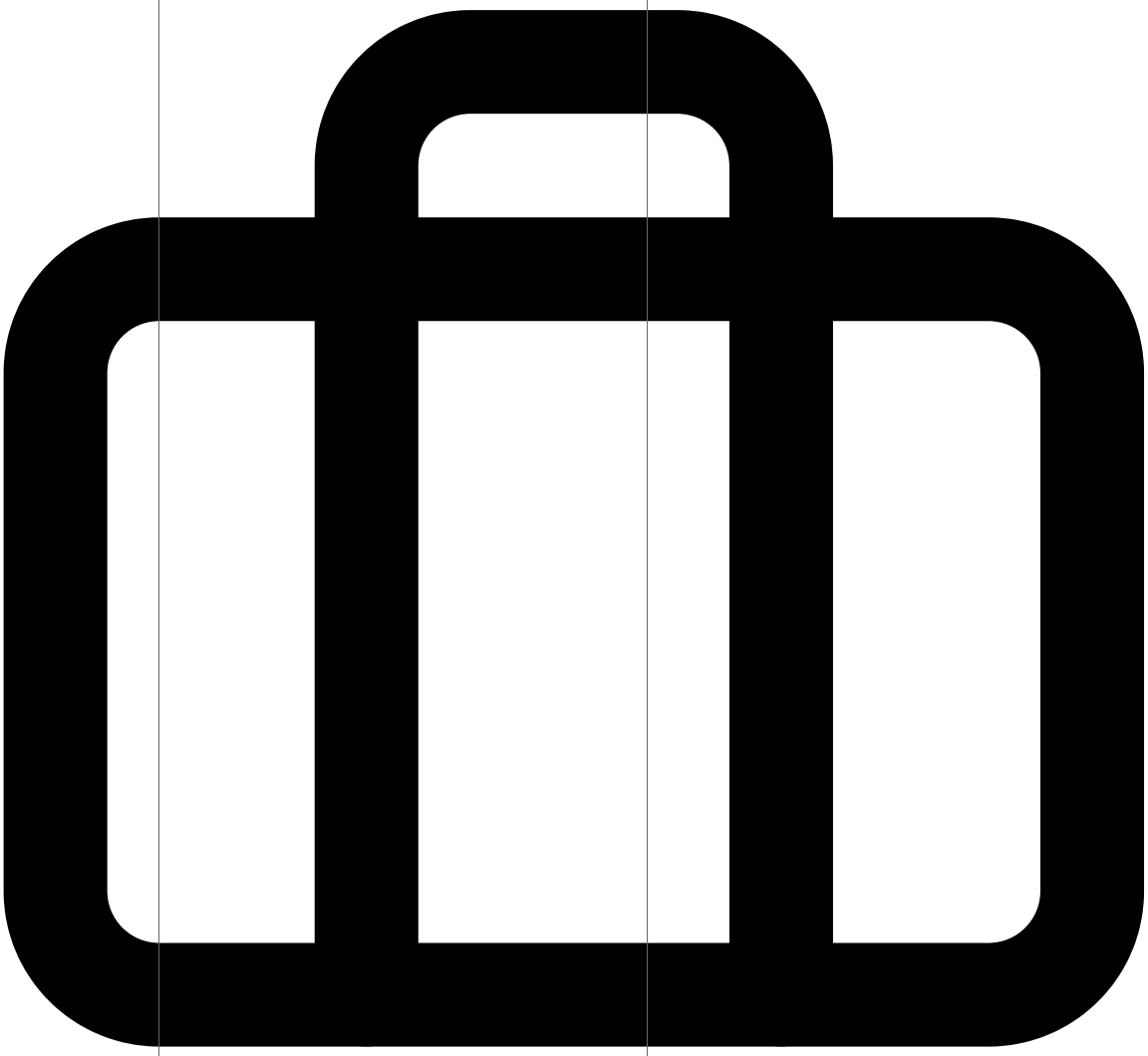
Description	Icon (76px)
Adding data	

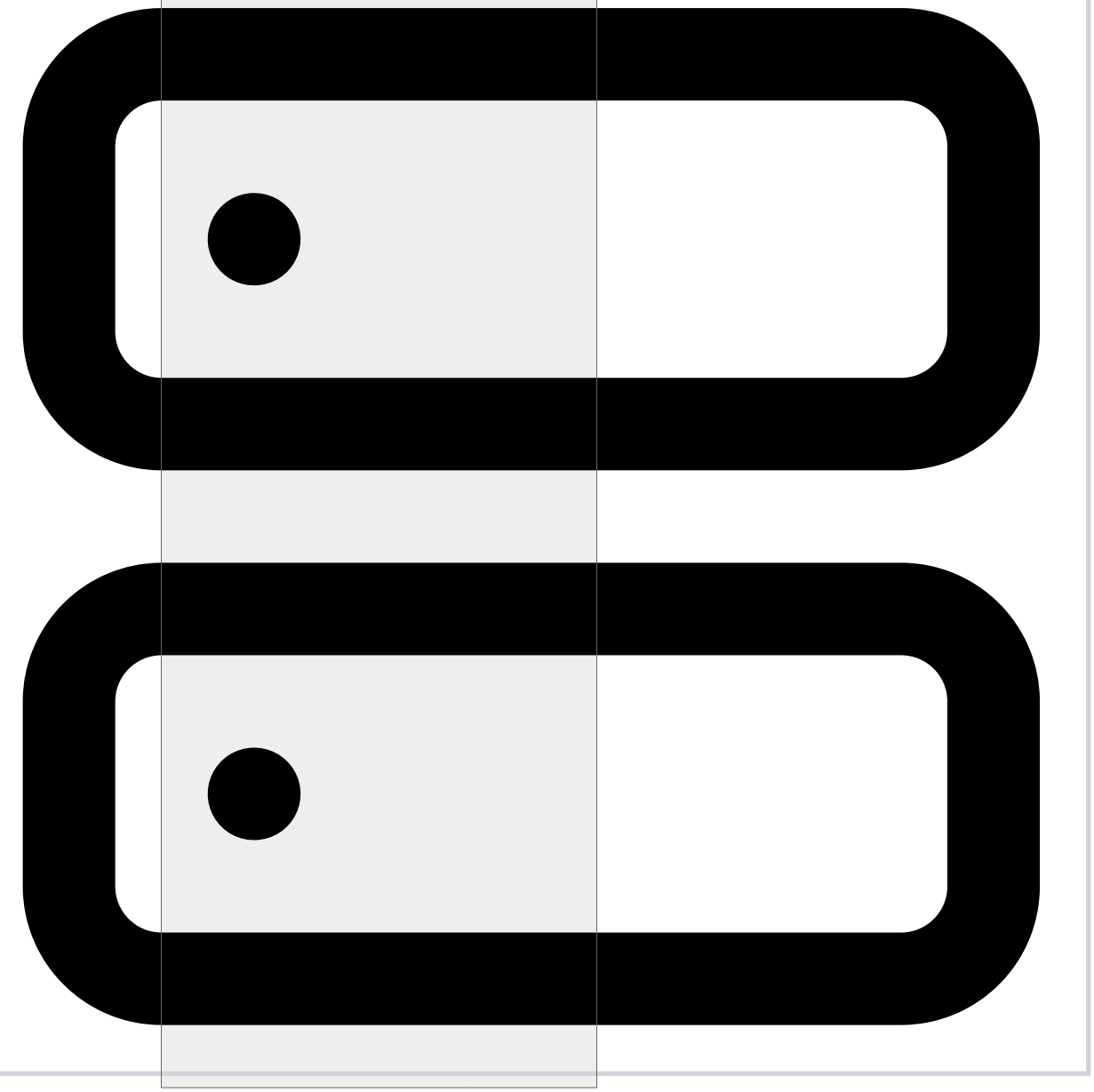
Description	Icon (76px)
Welcome	

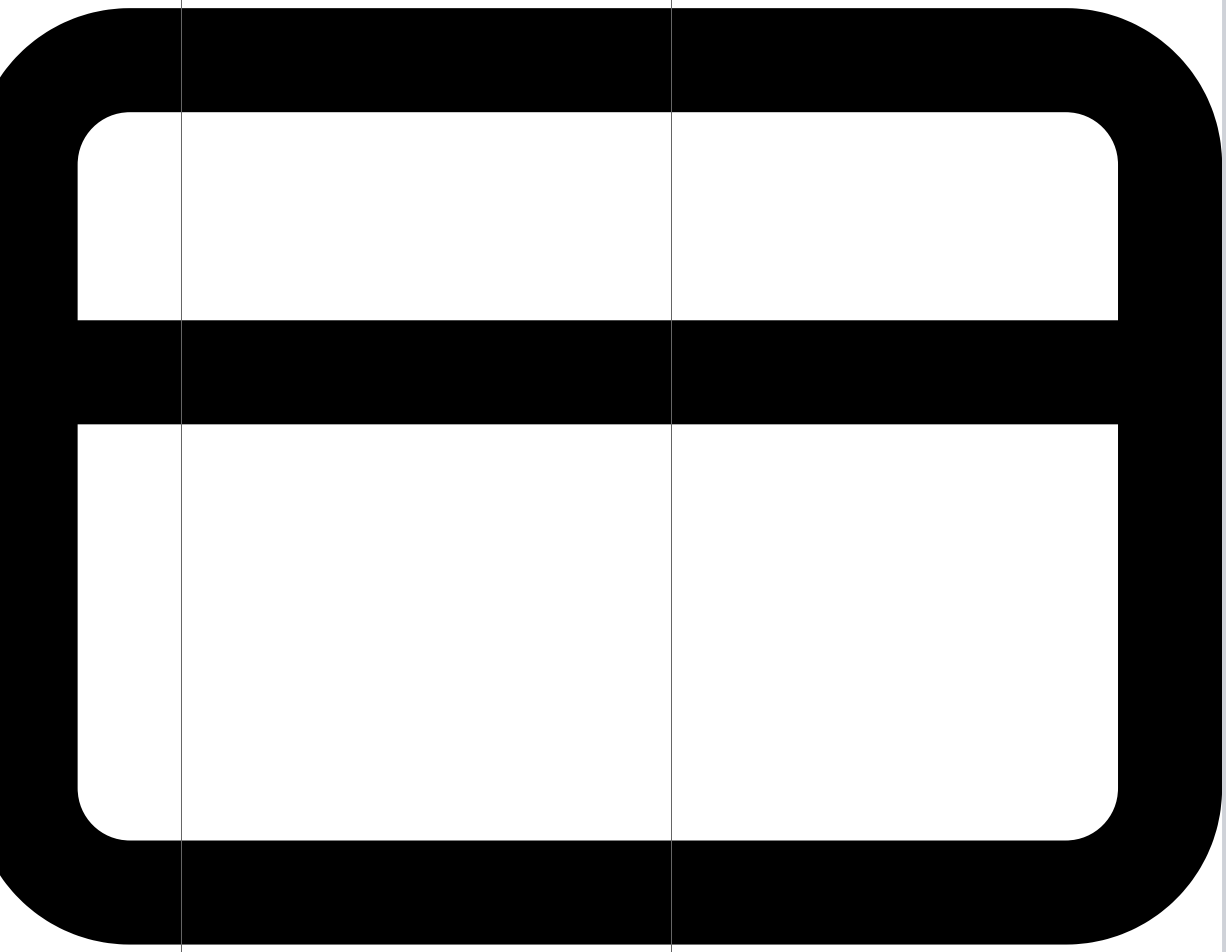
Description	Icon (76px)
Policies, rules & transformations	

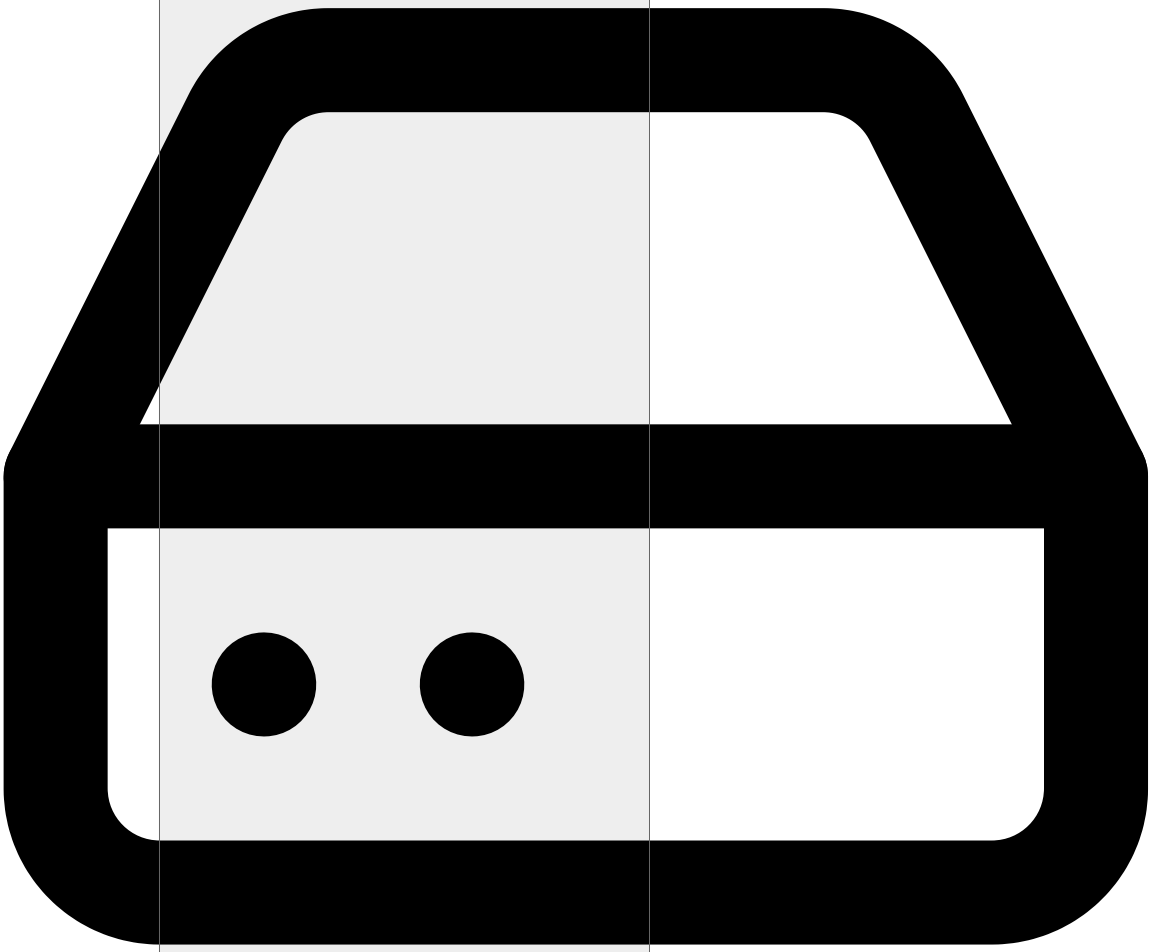
Description	Icon (76px)
Business Information	

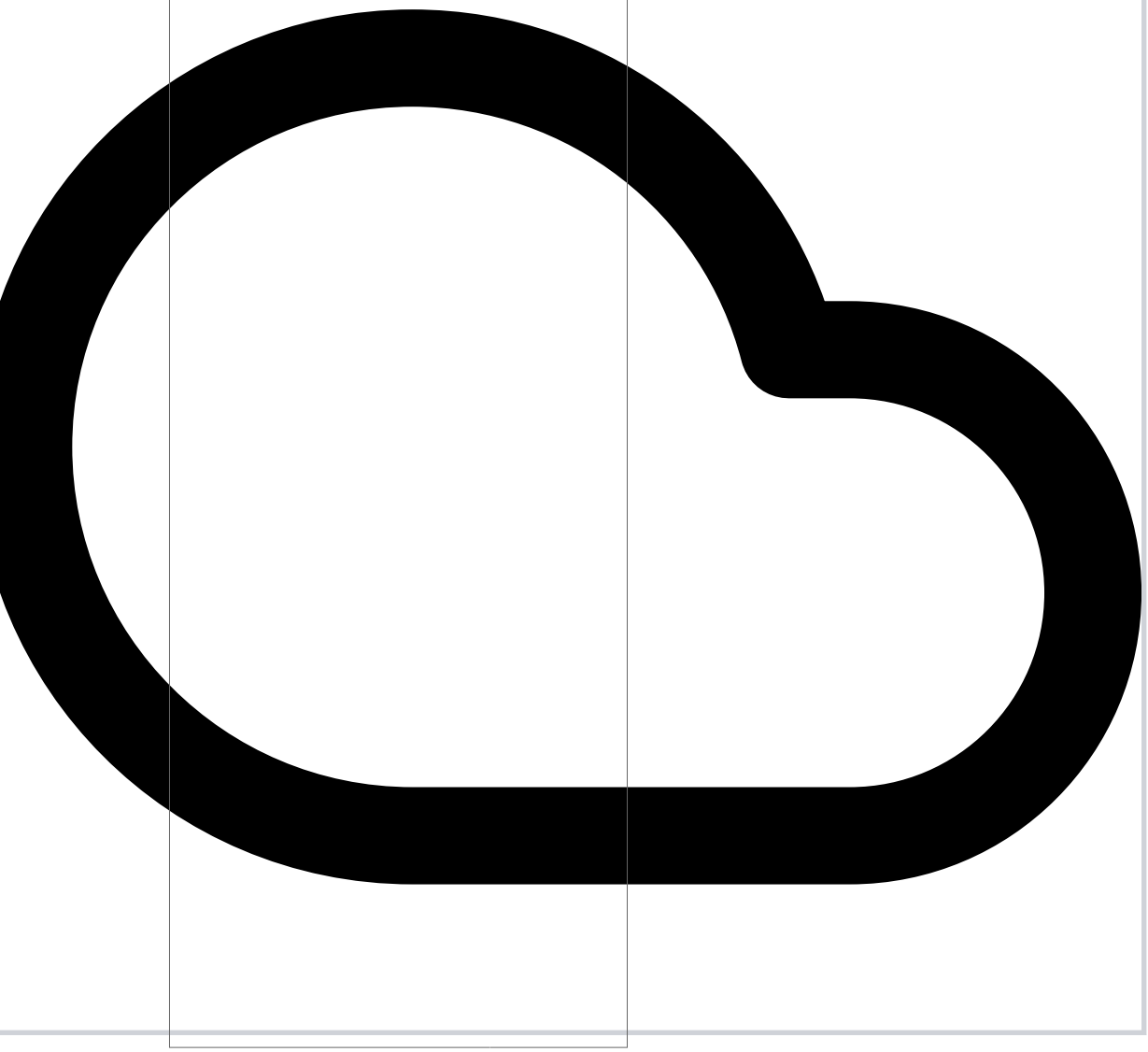
Description	Icon (76px)
Architecture	

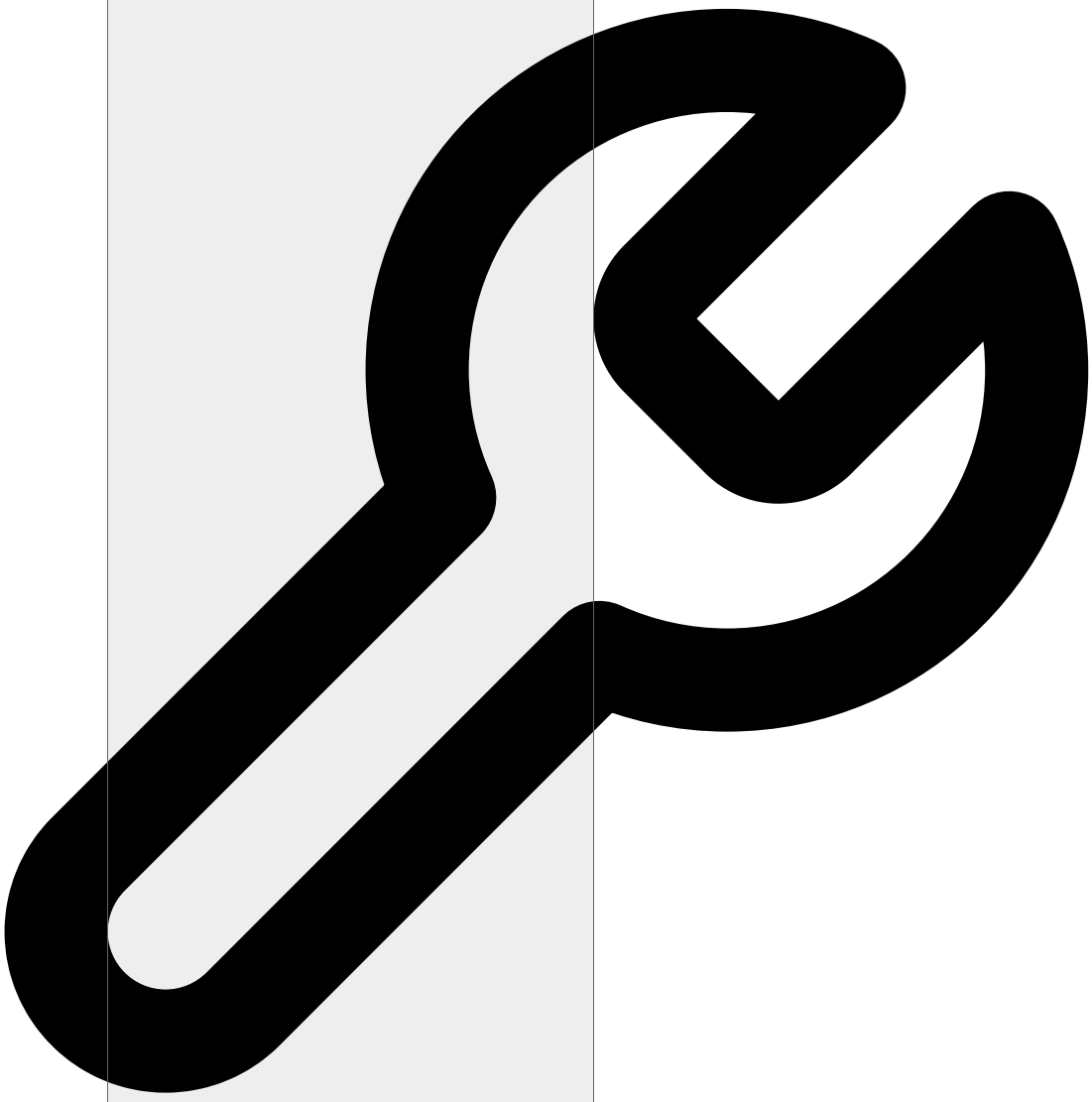
Description	Icon (76px)
Enterprise	

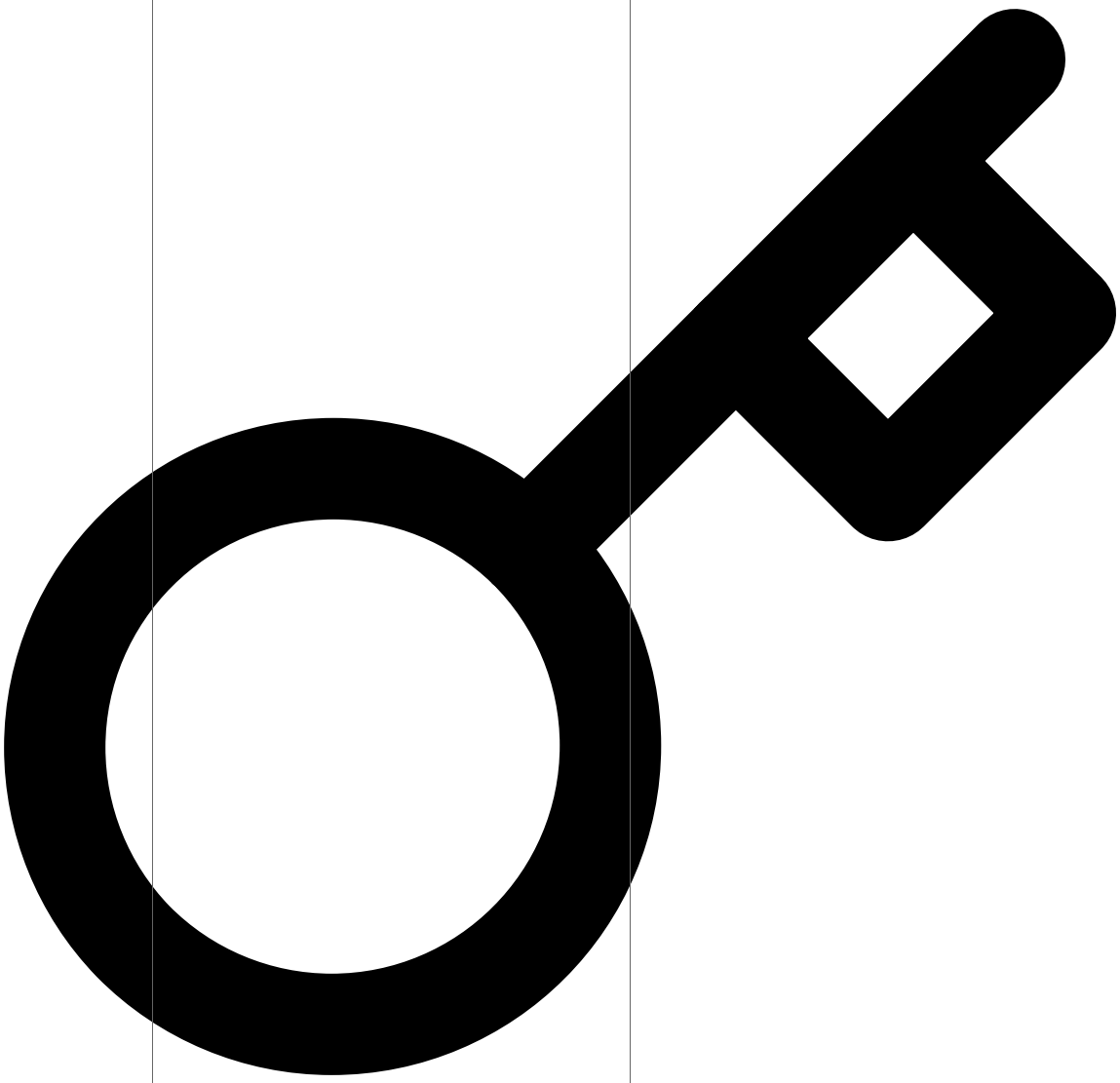
Description	Icon (76px)
Exchange	

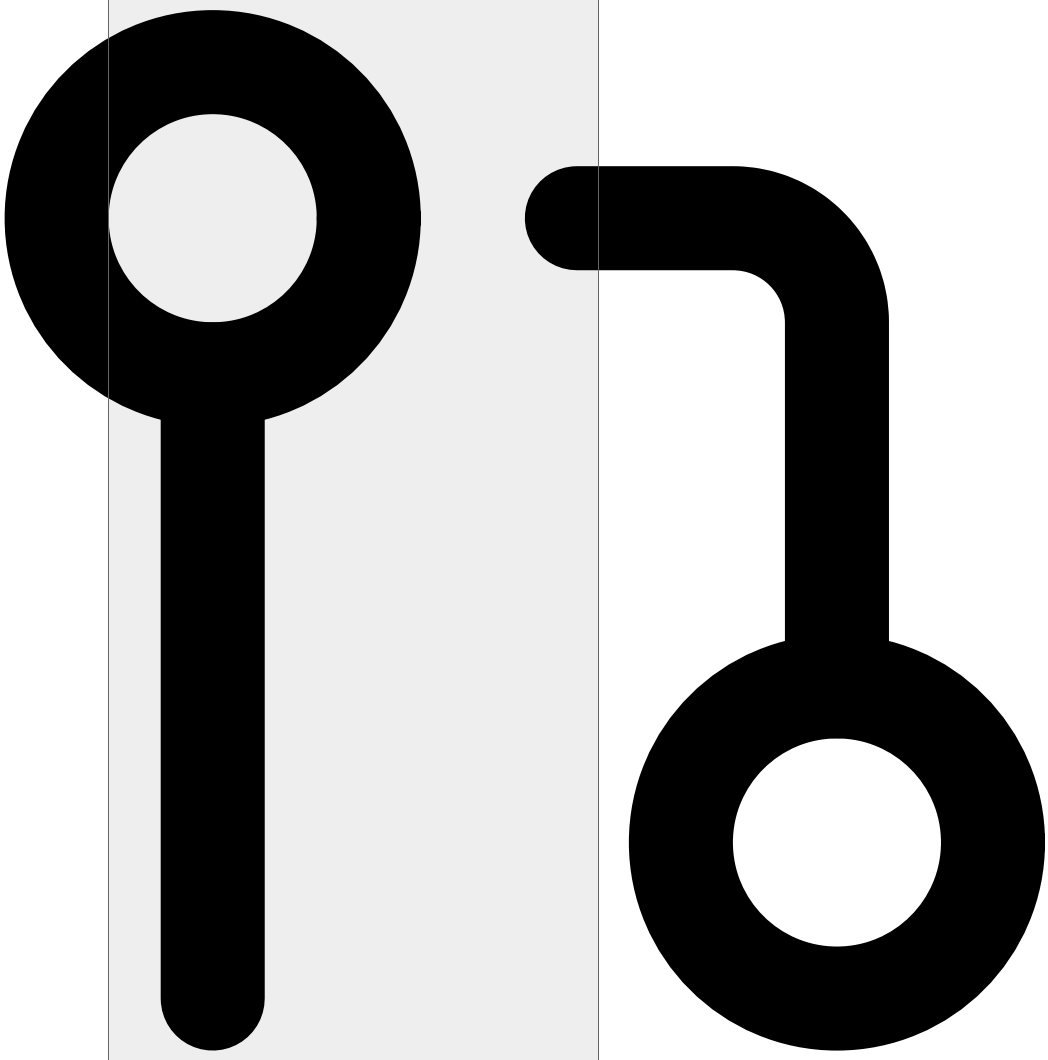
Description	Icon (76px)
Third-Party Licensing	

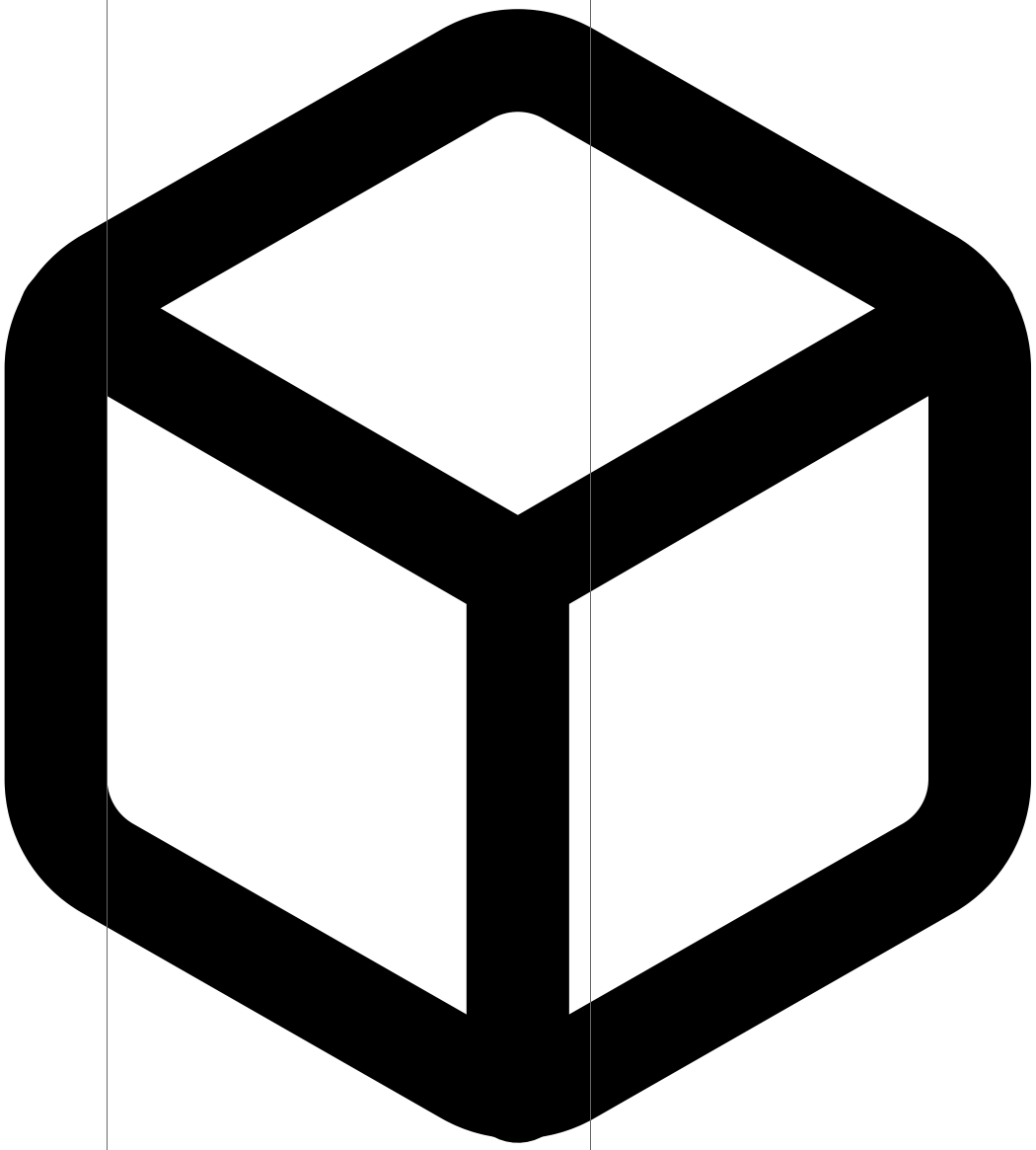
Description	Icon (76px)
Backup	

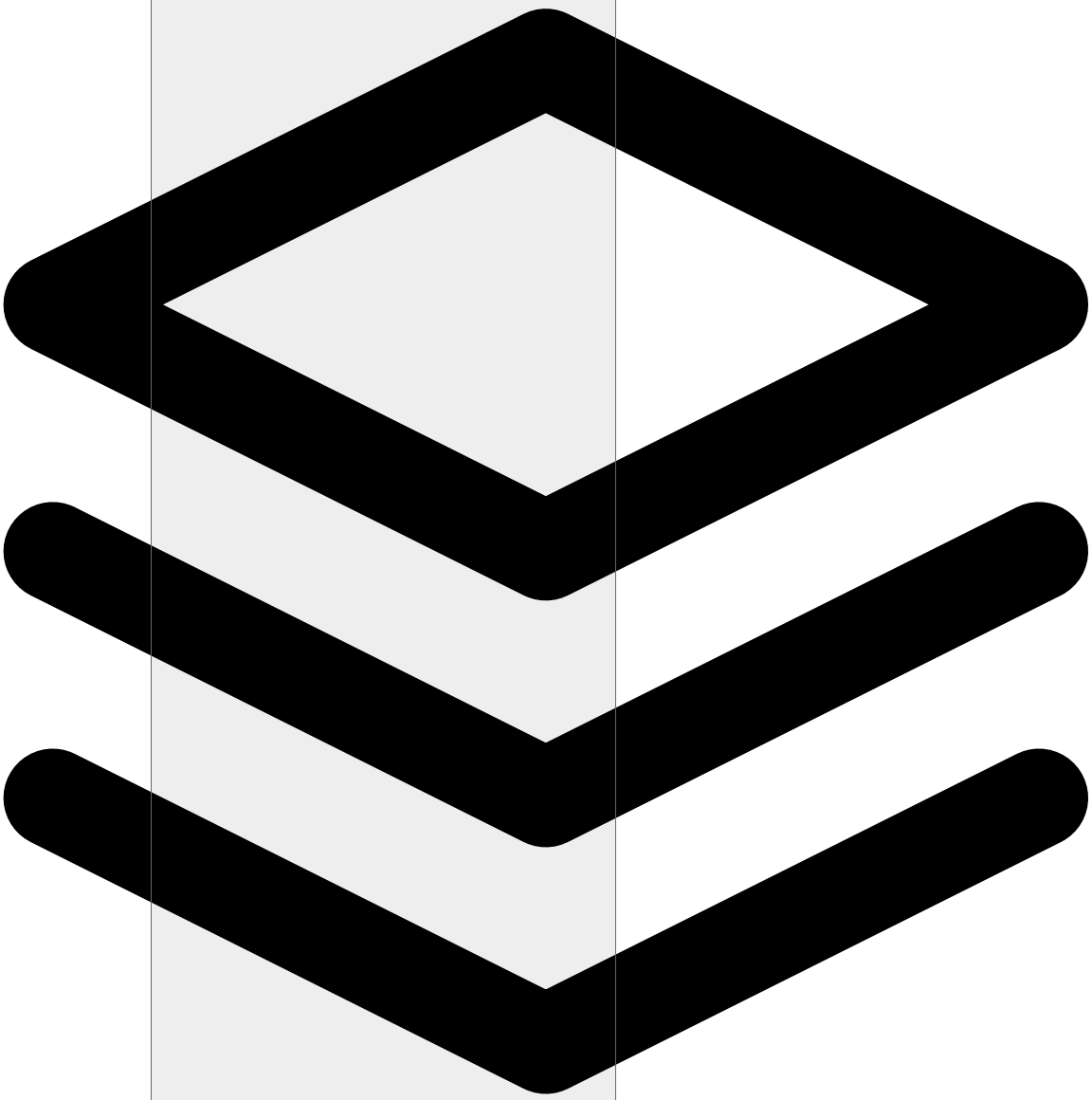
Description	Icon (76px)
Cloud	

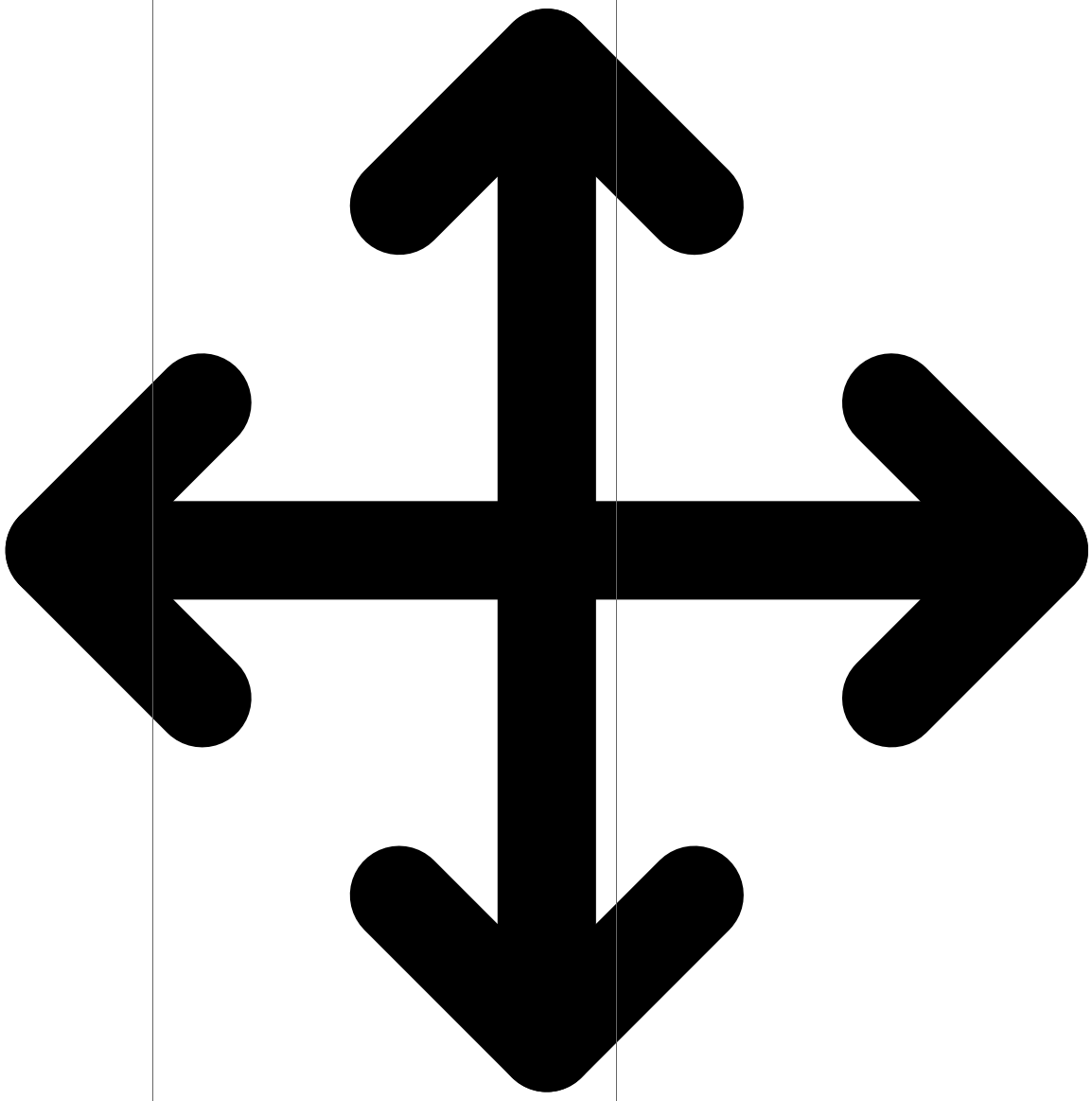
Description	Icon (76px)
Tools	

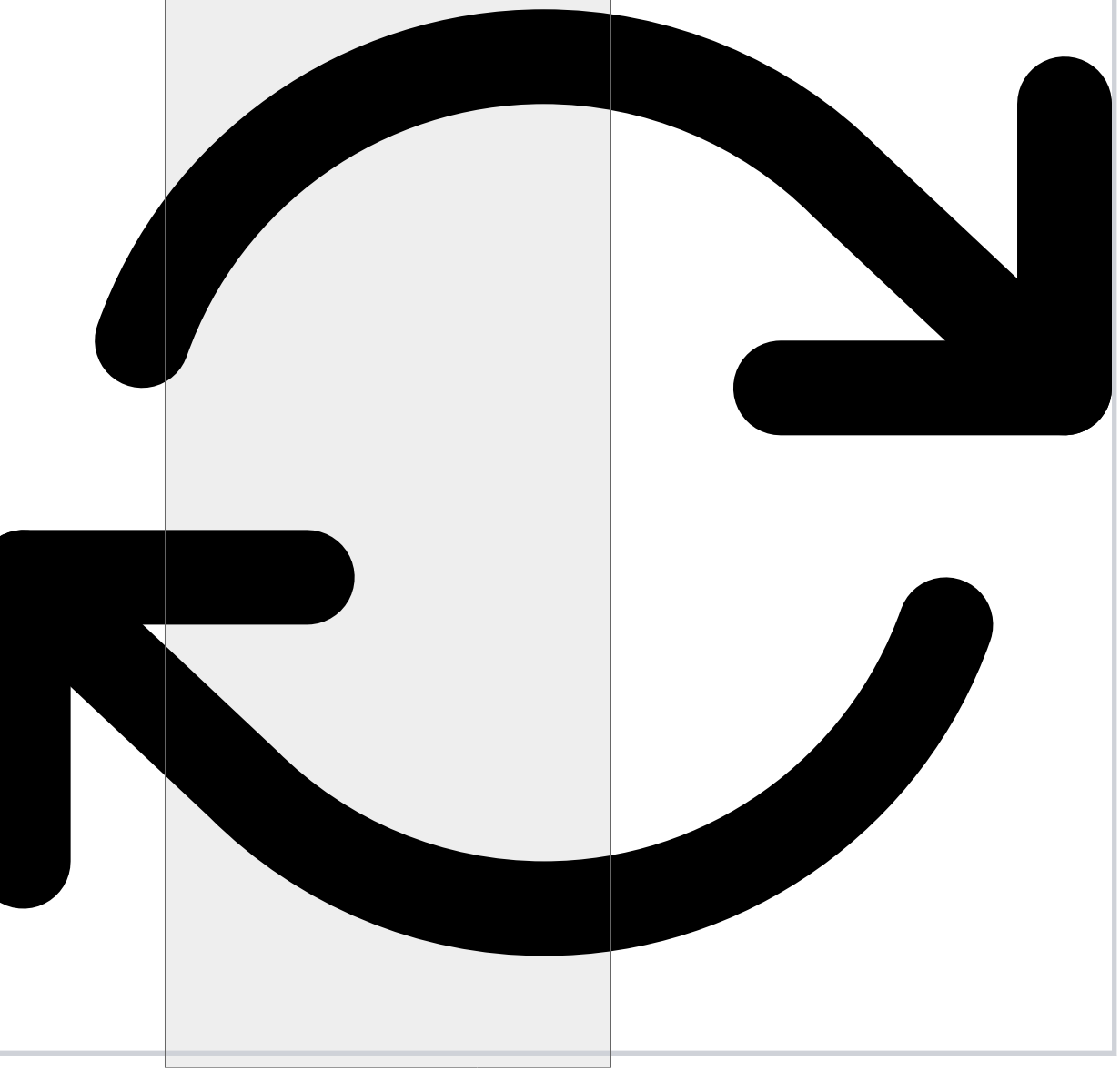
Description	Icon (76px)
Authorize	

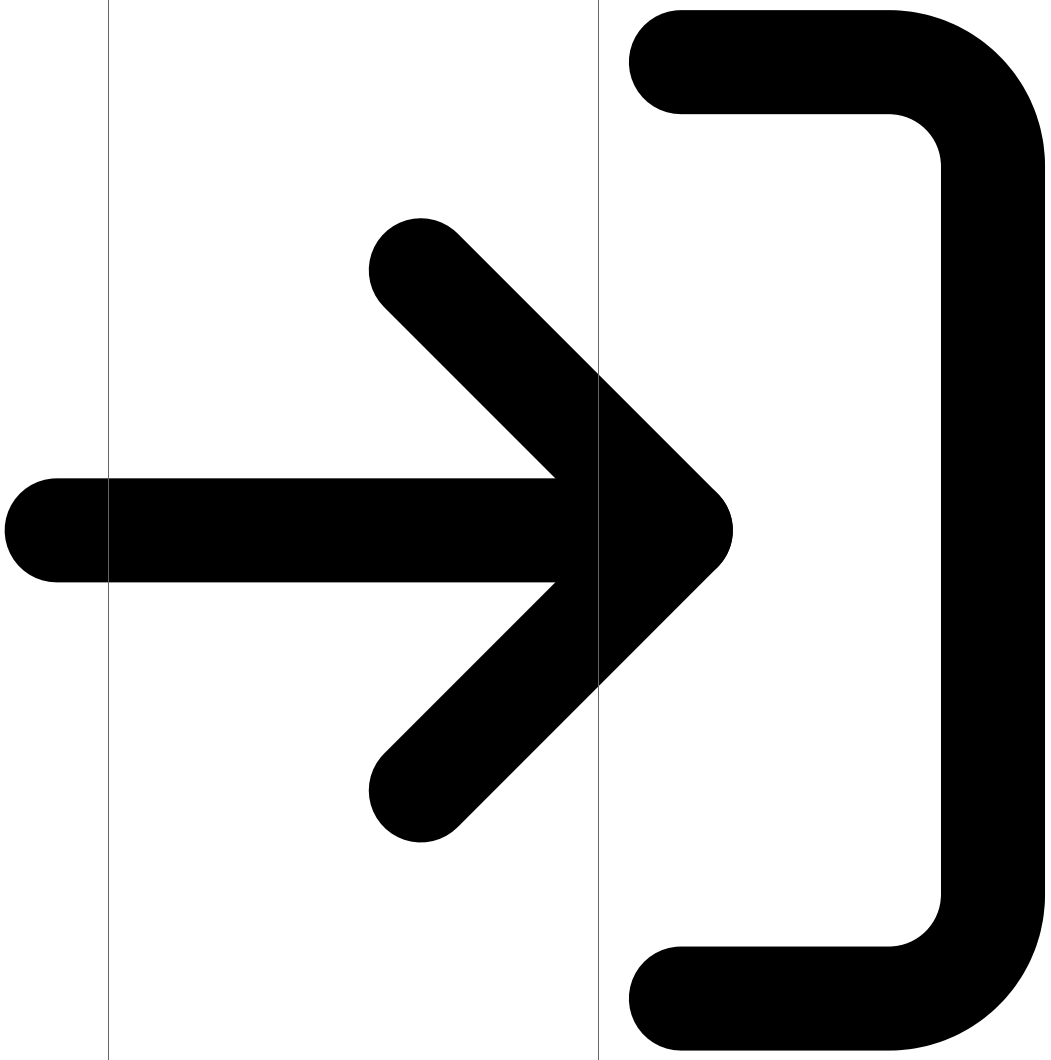
Description	Icon (76px)
API	

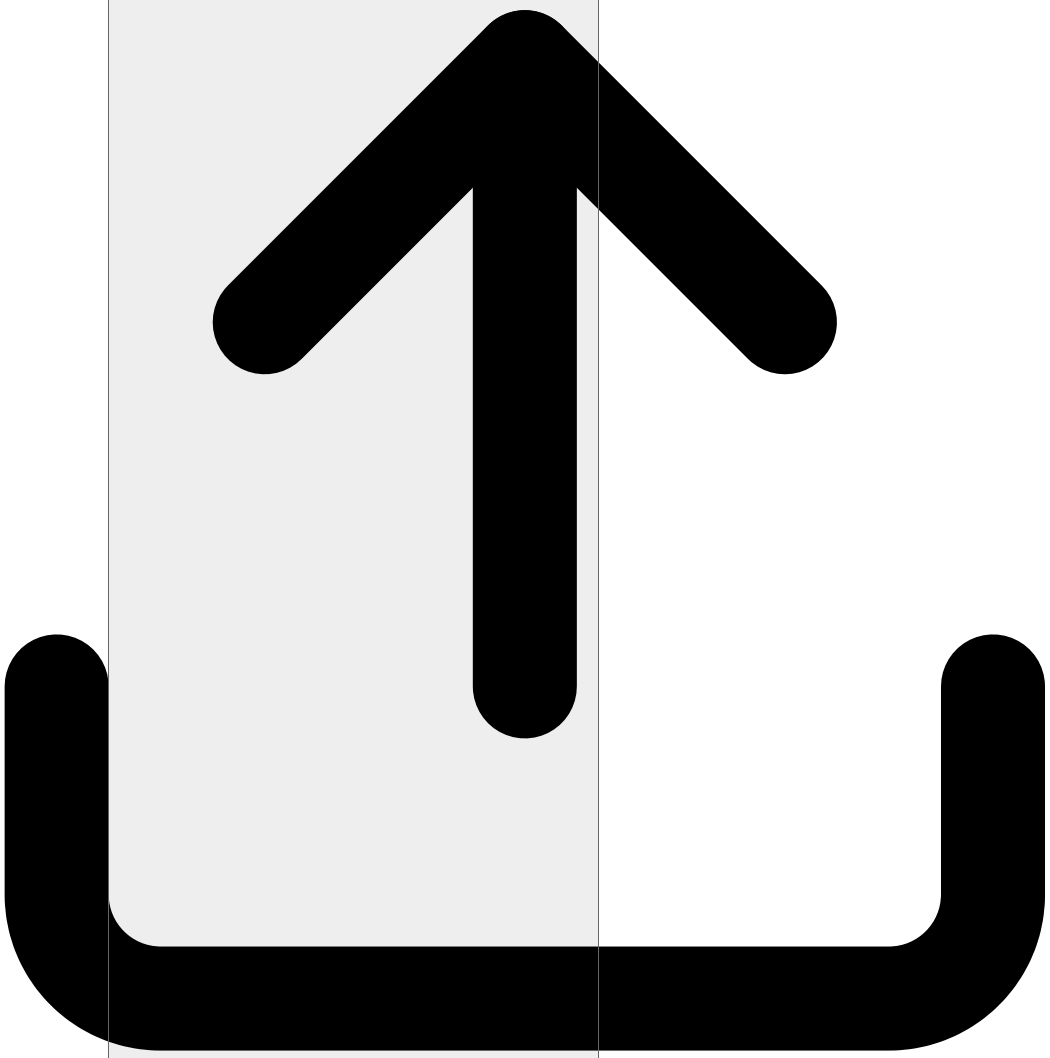
Description	Icon (76px)
AWS (box)	

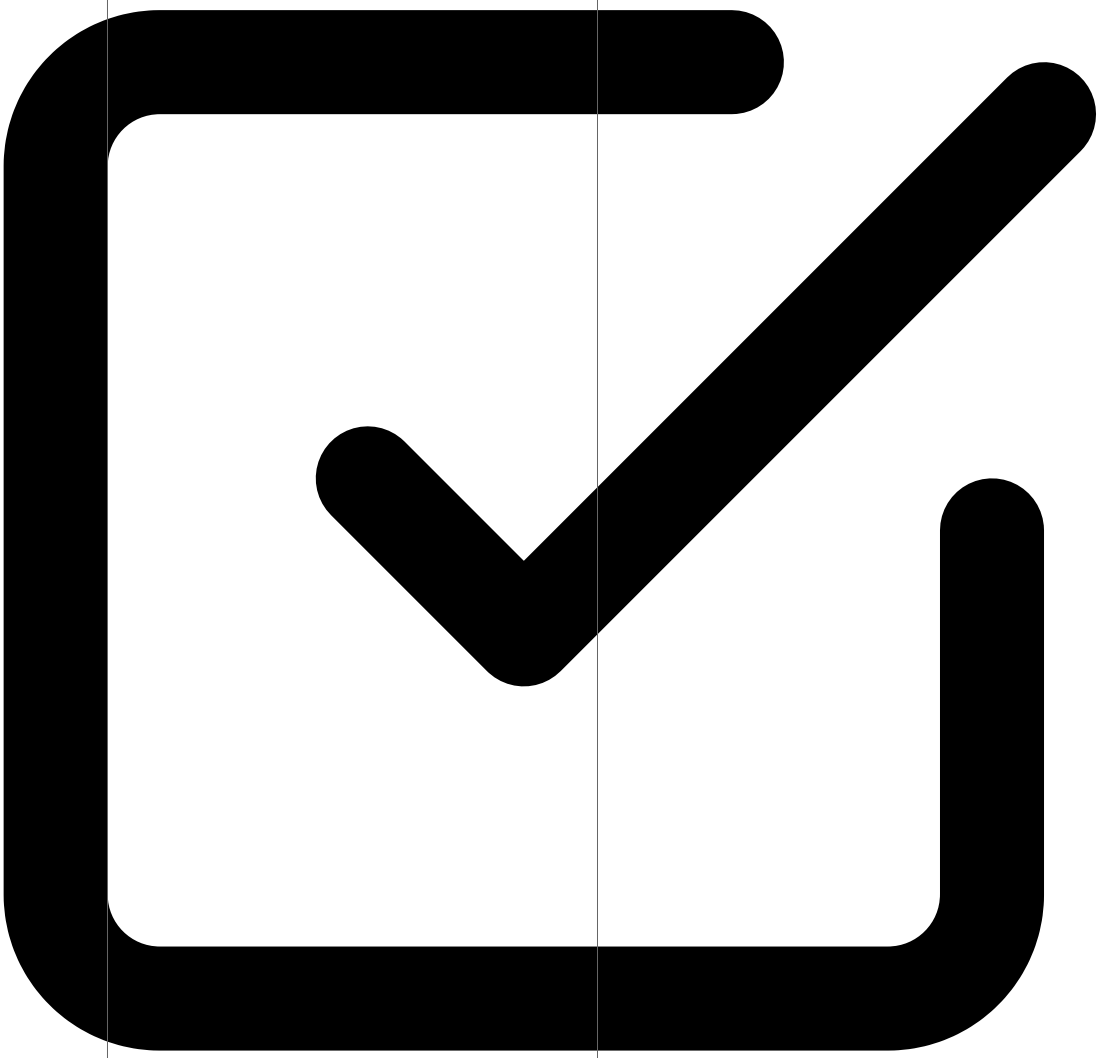
Description	Icon (76px)
Stack (layers)	

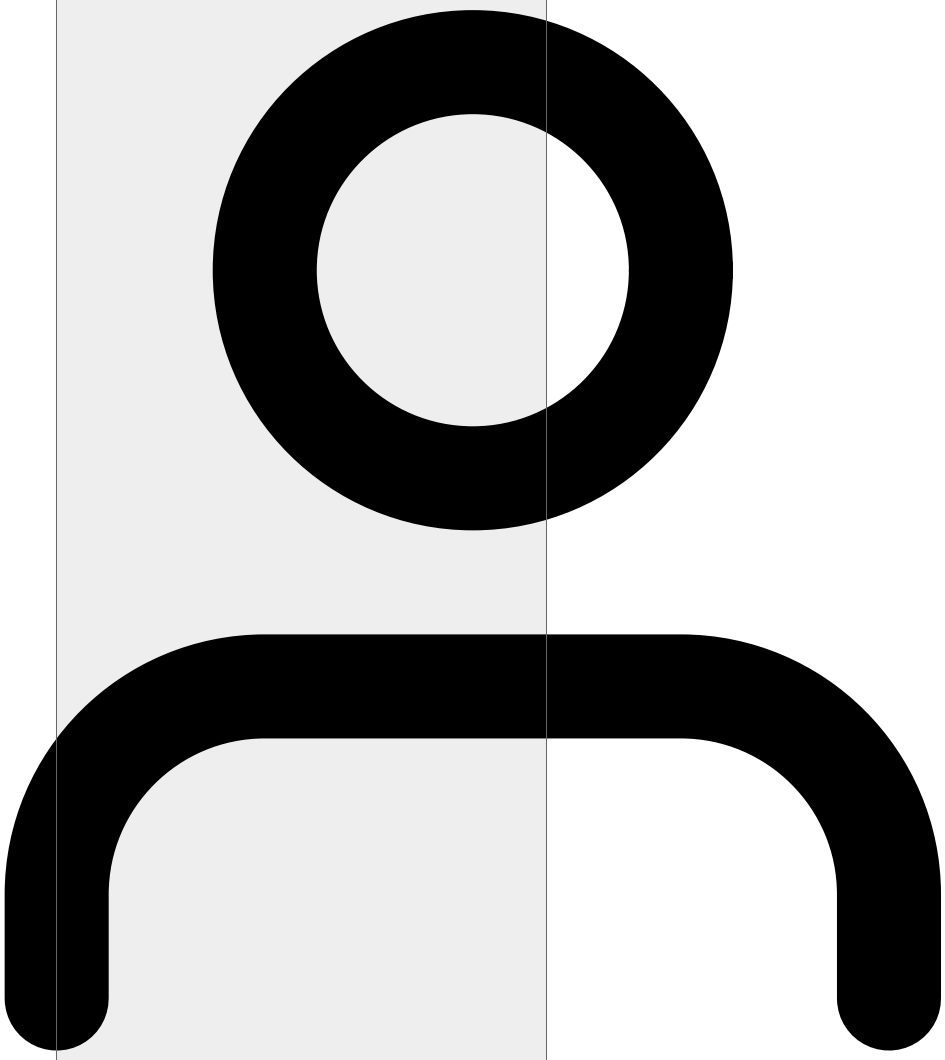
Description	Icon (76px)
Move (move)	

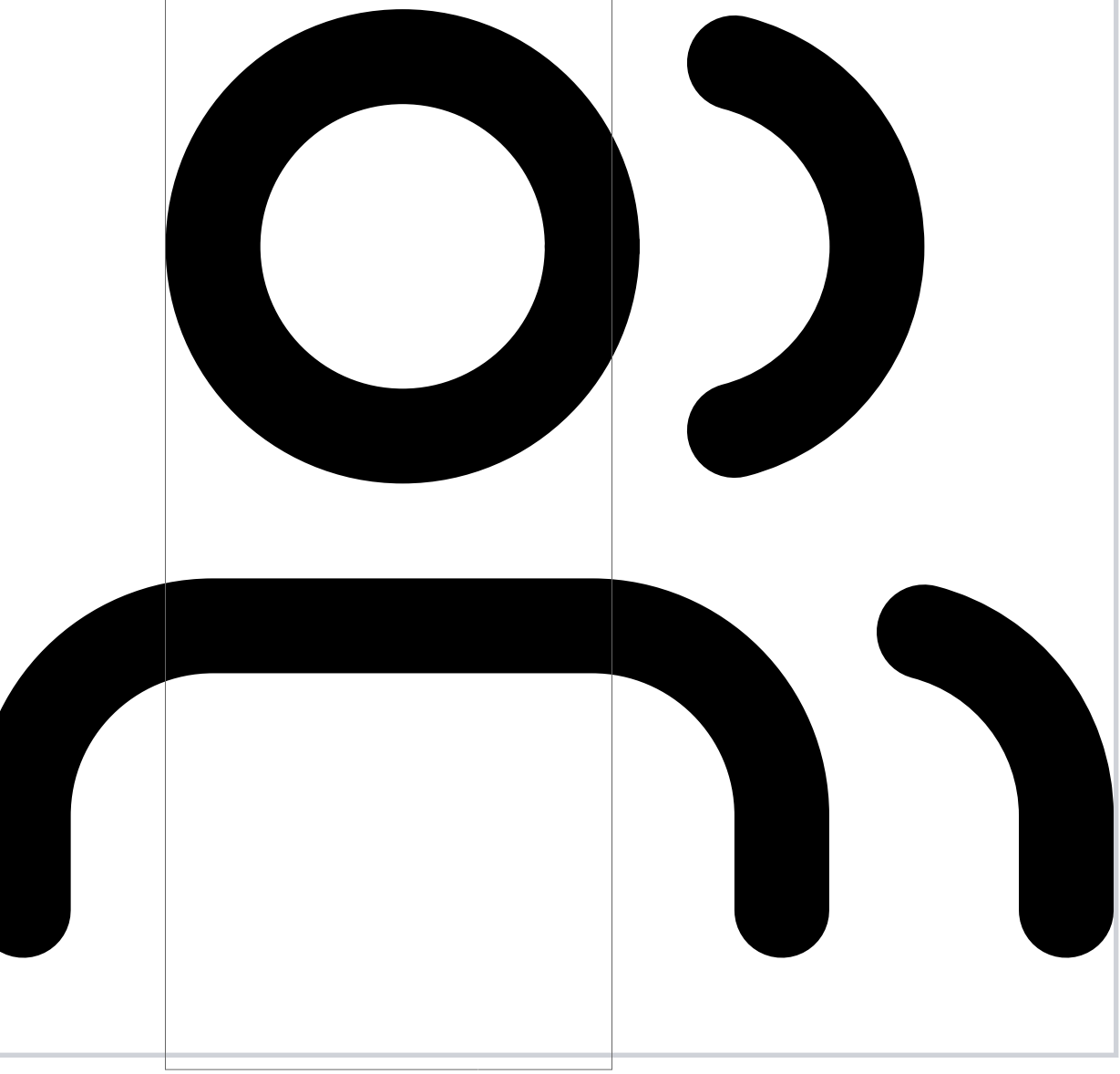
Description	Icon (76px)
Update (refresh-cw)	

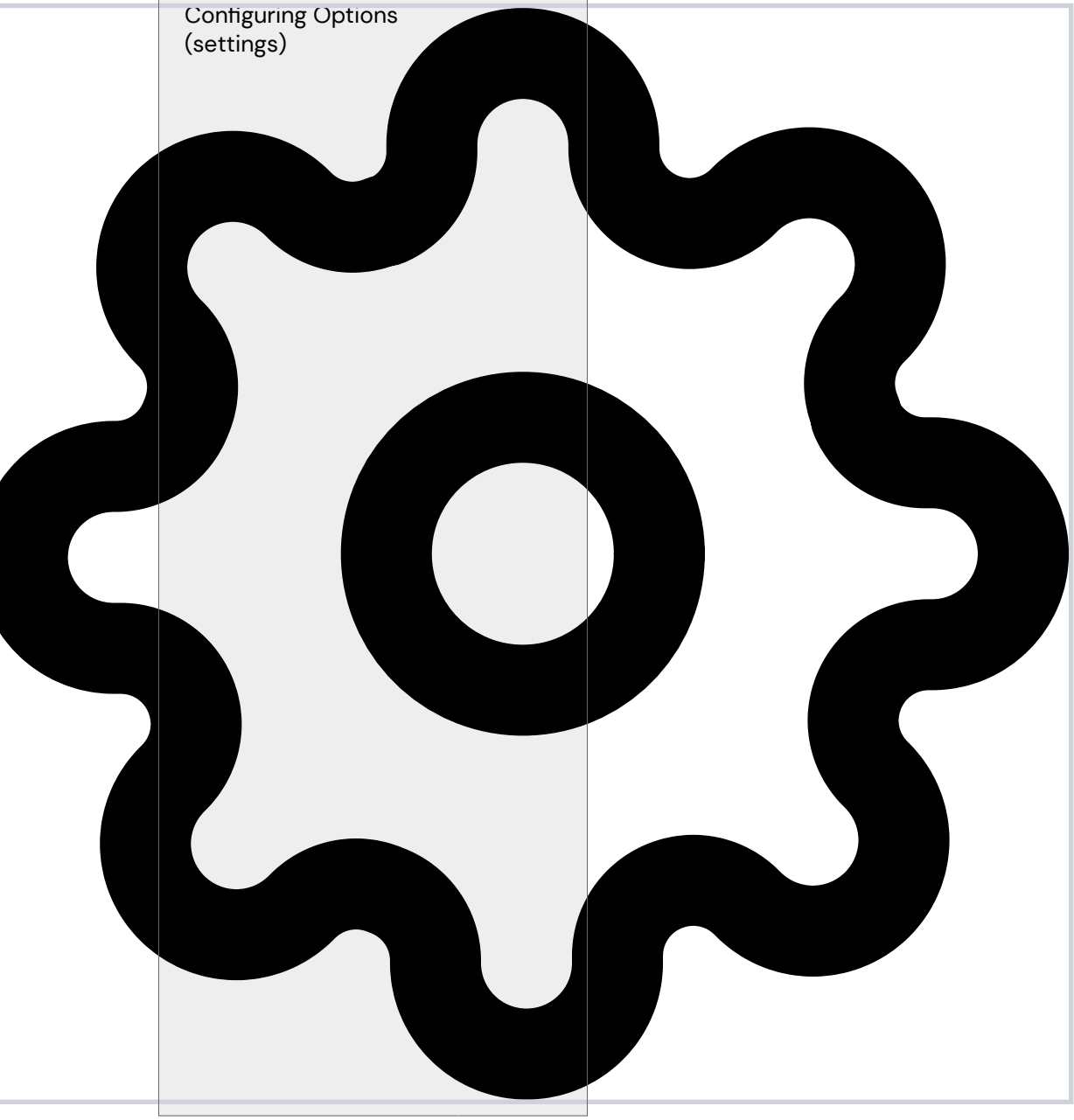
Description	Icon (76px)
Install (log-in)	

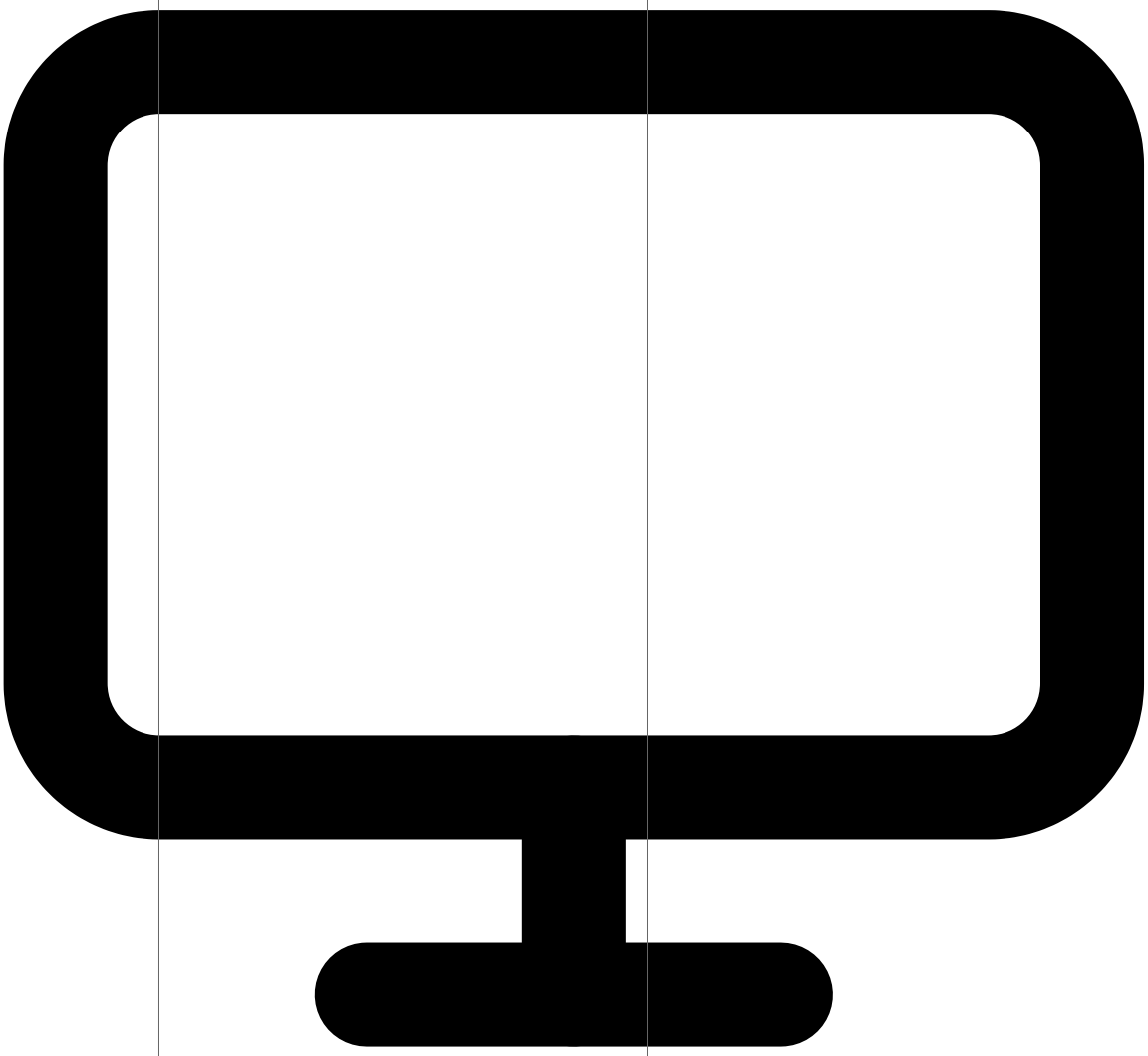
Description	Icon (76px)
Upload (upgrade)	

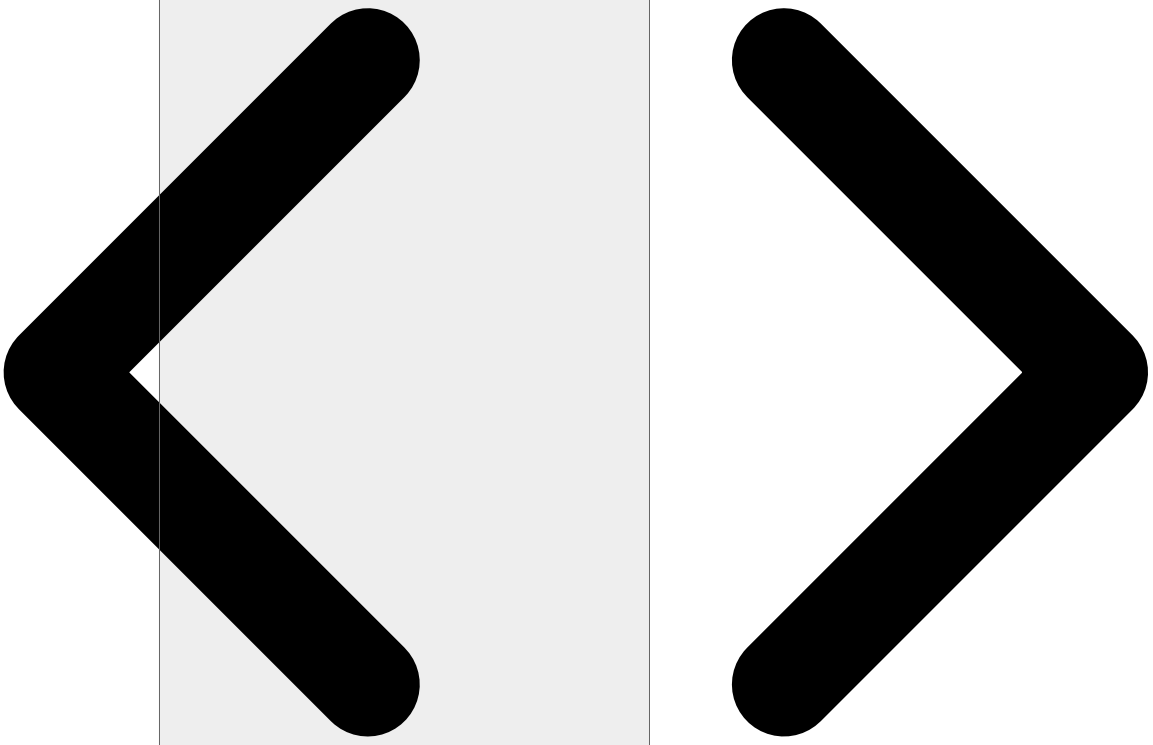
Description	Icon (76px)
Require (check-square)	

Description	Icon (76px)
User (user)	

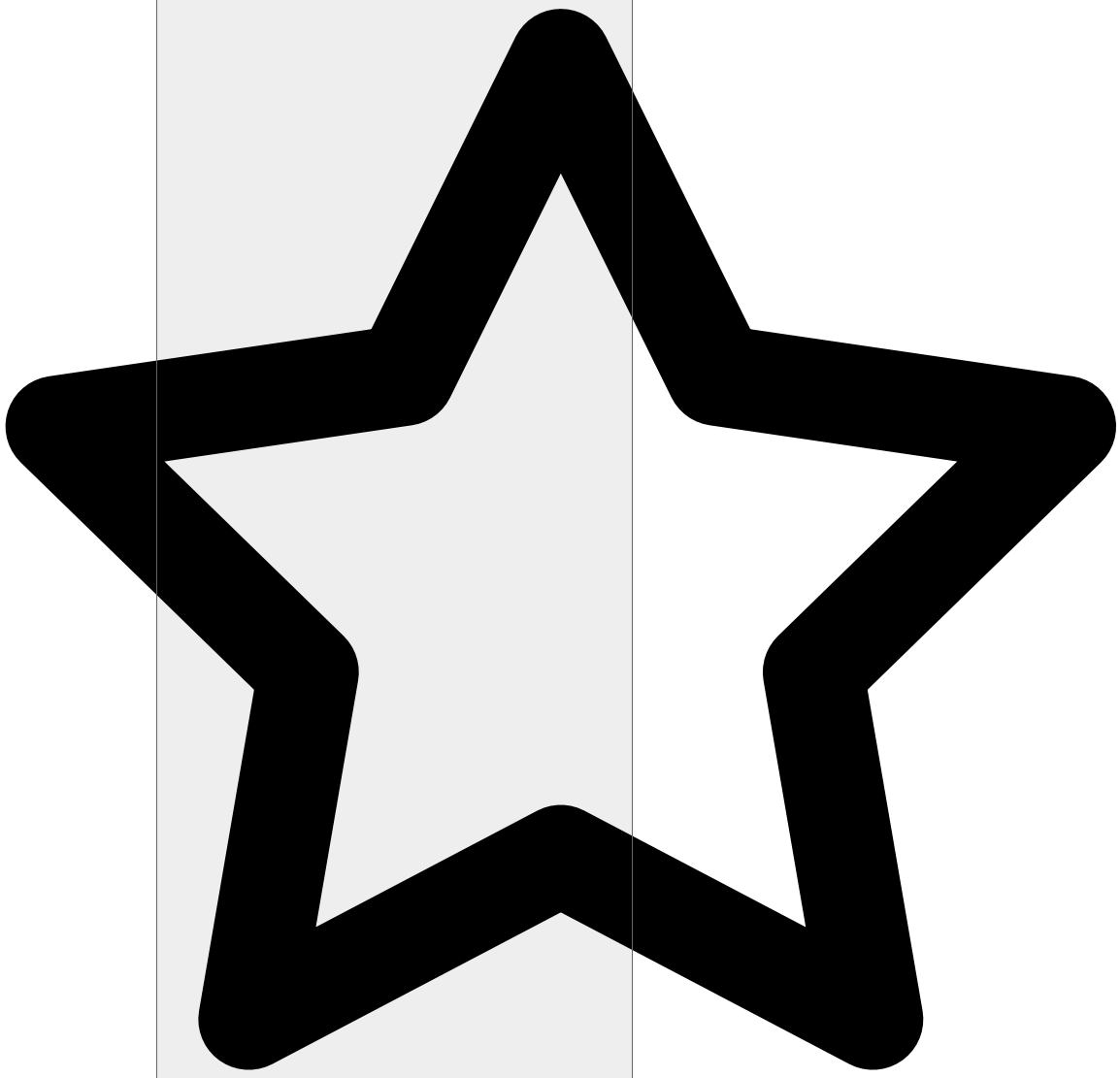
Description	Icon (76px)
Users	

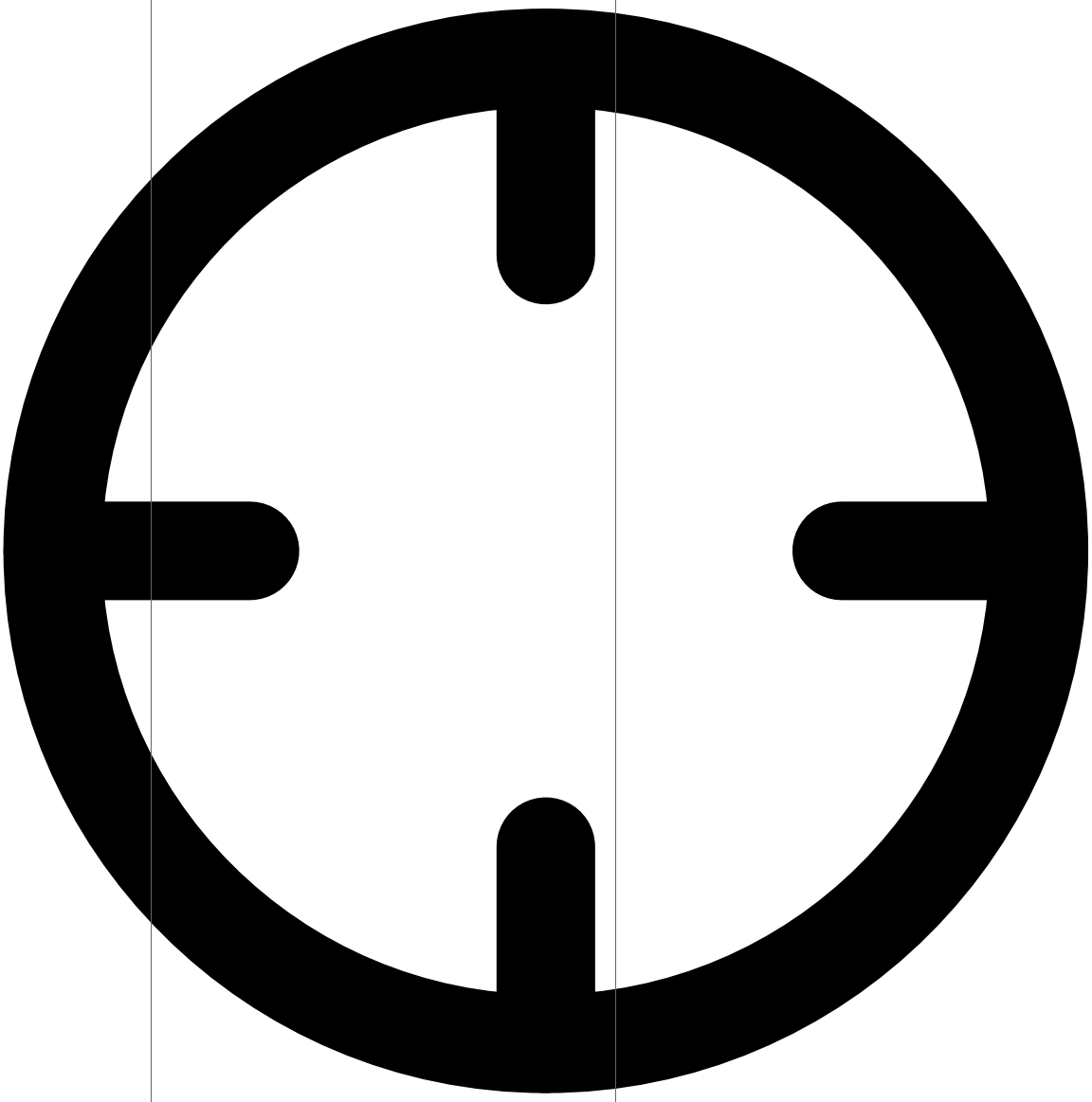
Description	Icon (76px)
Configuring Options (settings)	

Description	Icon (76px)
“Manage the Deployment” (monitor)	

Description	Icon (76px)
"Swagger" (code)	

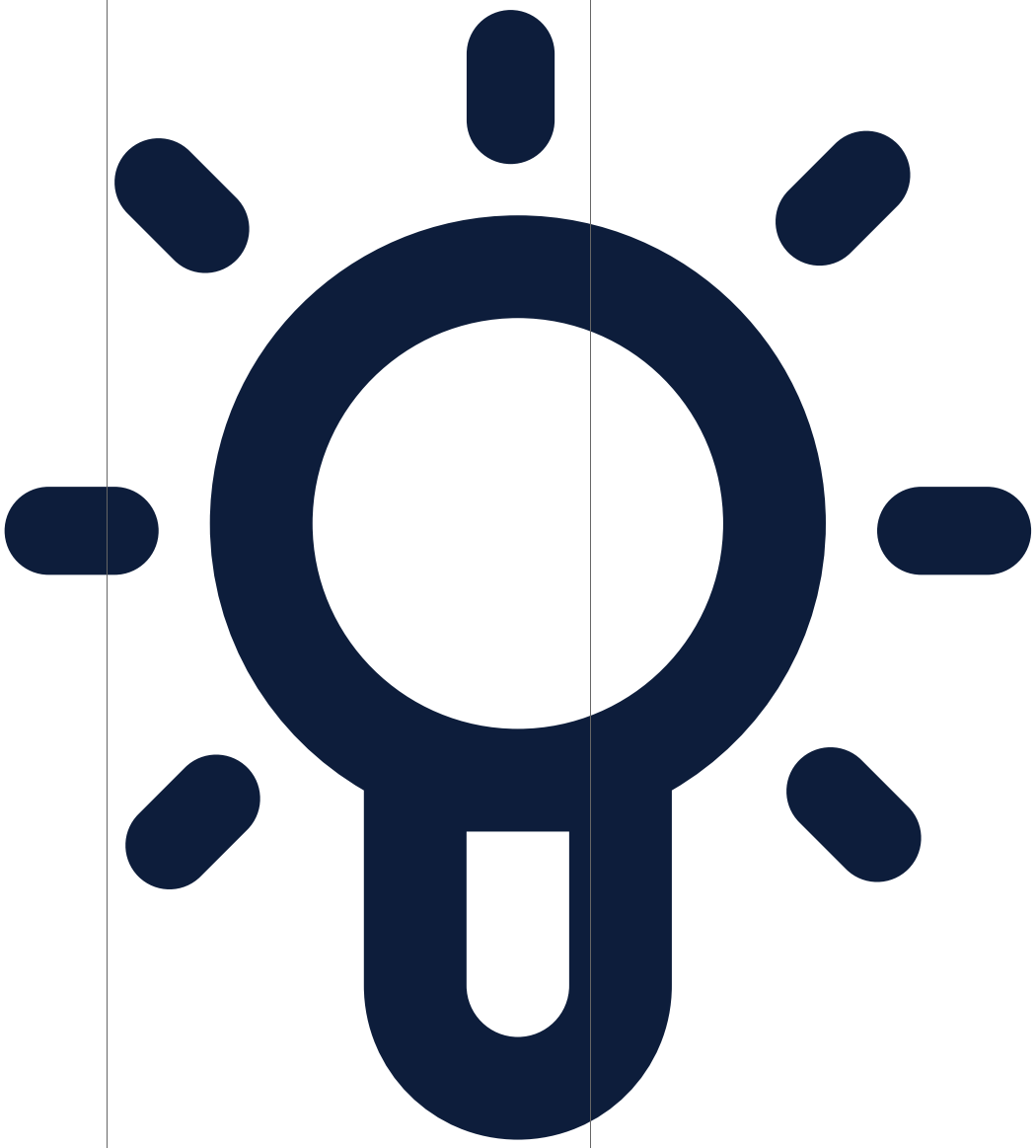
Description	Icon (76px)
Contact Support (help-circle)	

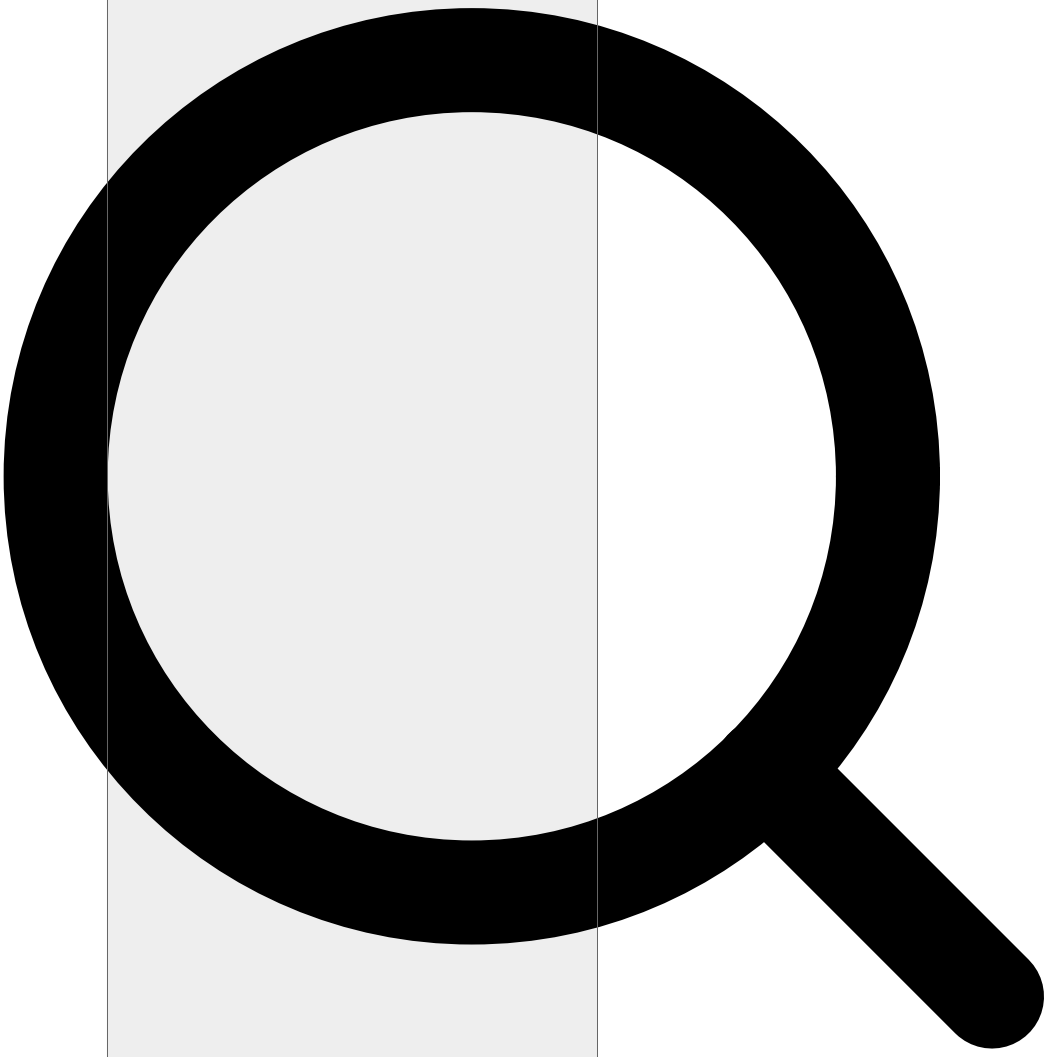
Description	Icon (76px)
(star)	

Description	Icon (76px)
Watermark (crosshair)	

Description	Icon
Important	

Description	Icon
Note (option to use FontAwesome \f15c)	

Description	Icon
Tip	

Description	Icon
Search bar icon (portal)	

Description	Icon
Warning