Admin Guide v25.10

See all our docs at docs.snorkel.ai

PDF generated on December 5, 2025

Welcome to Snorkel Admin

The Snorkel **Admin guide** provides comprehensive guidance for Snorkel administrators responsible for deploying, configuring, and maintaining Snorkel. Whether you're setting up a new installation, managing user access, or troubleshooting issues, these guides will help you efficiently manage your Snorkel environment.

Installation and deployment

Get your Snorkel AI Data Development Platform up and running:

- Installation overview and conventions: Understand deployment options and requirements.
- Snorkel Kubernetes installation overview: How to deploy Snorkel on Kubernetes.
- Adding a License Key: Activate your Snorkel installation.
- SDK installation: Set up the Python SDK for developers.

Integrations

- External S3 bucket storage: Connect an Amazon S3 bucket.
- Connect external models: Configure external models for your deployment.

Authentication and user management

Configure secure access to Snorkel:

- Platform authentication and user roles: Understand authentication methods.
- Snorkel user roles: Understand user roles and permissions.
- Configure SAML SSO: Set up SAML-based single sign-on.
- Configure OIDC SSO: Set up OpenID Connect single sign-on.
- Active Directory Roles synchronization: Synchronize with Active Directory roles.

Access control

Manage permissions and data access:

- Feature access management: Control access to specific platform features.
- Create RBAC-Based workspaces: Set up workspaces with role-based access.
- Configure file upload and download controls: Manage data import/export permissions.
- Manage RBACs for dataset connectors: Control access to data sources.

Security and compliance

Ensure your Snorkel deployment meets security standards:

- Data security: Learn about data encryption, storage, and transfer.
- Secret storage and encryption: Understand how sensitive information is protected.
- Audit logging in Snorkel: Monitor system activity and user actions.

Administration and maintenance

Day-to-day administration tasks:

- User and Admin settings: Configure platform settings.
- Credential management: Manage API keys and credentials.
- Debugging technical issues: Troubleshoot problems with job logs and support bundles.
- Supported languages: Review language support for your deployment.

Installation Overview and Conventions

We're excited to get you set up with the Snorkel AI Data Development Platform! There are several ways to install Snorkel. The Snorkel AI Data Development Platform can be installed with Kubernetes for collaborative development and long-term scalability or configured with a single, standalone Linux server for lab environments. If you have any questions, contact a Snorkel AI team member.

You'll find the following content in the Help Center:

- **Prior to Installing:** an overview of installation options, as well as client-side browser requirements for installing the Snorkel AI Data Development Platform.
- **Deployment Options:** instructions to install Snorkel AI Data Development Platform on a Kubernetes-based system.
- **Post-Installation:** post-installation instructions, such as uploading a license key for the Snorkel AI Data Development Platform.

Conventions Used in These Installation Documents

`represents a block of code to be run`

```
Represents longer
blocks of code
to be run
```

Code blocks like `<this>` should be replaced with items specific to your environment.

Client-side requirements

Device specifications

Refer to these specifications for requirements of the client machine used to access the Snorkel AI Data Development Platform on a supported web browser.

- RAM: 8GB RAM, with at least 3GB of available RAM during usage of the Snorkel AI Data Development Platform
- CPU: 4-core CPU, or 6-core CPU if no GPU acceleration is available
- GPU access recommended

Browser Support

Fully supported:

The Snorkel AI Data Development Platform currently supports the following browser configurations:

- MacOS
  - Google Chrome: version *128* or later
- Windows
  - Google Chrome: version *128* or later

The Snorkel AI Data Development Platform supports the five latest versions of Google Chrome.

Partial support:

- Microsoft Edge
- Arc browser
- Other browsers that are built on Chromium, the open-source framework that powers Google Chrome

Unsupported:

- Firefox
- Safari
- Other browsers not built on Chromium and not listed as supported
- Accessing Snorkel through any browser on mobile devices or tablets

Snorkel Kubernetes installation overview

This topic is your starting point to deploy the Snorkel AI Data Development Platform on Kubernetes. Installing the Snorkel AI Data Development Platform is a two-step process:

1. Preparing Infrastructure Resources
2. Installing Snorkel into the Kubernetes Cluster

Preparing infrastructure resources

At a high level, the Snorkel AI Data Development Platform requires the following pieces of infrastructure for deployment:

- An operational Kubernetes (K8s) cluster
- NFS or equivalent (NAS drive, etc.)
- A domain to create URLs for the platform

If you are using a major cloud provider (AWS, GCP, and Azure) for Snorkel, you should deploy into a net-new K8s cluster. However, deploying into an existing cluster and on-prem options are possible. Snorkel offers Terraform and manual methods of spinning up all required cloud infrastructure pieces for the three major cloud providers in preparation for installing the Snorkel AI Data Development Platform.

Assumptions and requirements

The Snorkel AI Data Development Platform can run on any Kubernetes installation as long as it satisfies our requirements on Cluster Specifications and Storage and Ingress.

Cluster Specifications

Category	Requirement
Minimum Kubernetes version	1.25+
Recommended Kubernetes distribution	AWS EKS, GCP GKE, Azure AKS, OpenShift 4.9+
Node Processor	X86_64
Namespace total CPU	64+
Namespace total RAM	360 GB+
Namespace total GPU	4+ T4 or better
Available per-pod CPU	16+
Available per-pod RAM	64 GB+

Category	Requirement
Storage volumes	768 GB+ (use-case dependent) NFS-equivalent with read/write access

Storage and ingress

To run the Snorkel AI Data Development Platform in your K8s cluster, whether it's running in a private datacenter or public cloud, you must meet these requirements:

- A **StorageClass** that supports the **ReadWriteOnly** access mode.
- A **StorageClass** that supports the **ReadWriteMany** access mode (e.g., NFS, or other filesystems).
  - For AWS, Elastic File System (EFS) is a good option.
- An **IngressController** that is running.
  - For AWS, AWS Load Balancer Controller is a good option.
- A **DNS domain/zone** that is ready for the Snorkel AI Data Development Platform to use. Snorkel requires five subdomains.
- A **TLS certificate** with a publicly-recognized cert.

Major cloud infrastructure scripts

For the three major cloud providers, we provide Terraform and manual instructions to spin up required infrastructure that satisfy the requirements above.

AWS

- AWS Infrastructure Setup - Terraform (Recommended)
- AWS Infrastructure Setup - Manual

GCP

- GCP Infrastructure Setup - Terraform (Recommended)
- GCP Infrastructure Setup - Manual

Azure

- Azure Infrastructure Setup - Terraform (Recommended)
- Azure Infrastructure Setup - Manual

Install Snorkel into the Kubernetes cluster

Once all of the required infrastructure resources are provisioned, you're ready to deploy the Snorkel AI Data Development Platform into the Kubernetes cluster. Choose from these methods of installation into the cluster:

- Deploying with an Admin Console (Recommended)
- Deploying with Helm

Setup and Installation

Snorkel Python SDK

The `snorkelflow` SDK can be provided in several formats for installation on the Snorkel host server or remote nodes:

- A wheel file, e.g., `snorkelflow-xx.yy.zz-py3-none-any.whl`
- A Docker image containing the wheel file

Installation instructions for each distribution method can be found below.

> **TIP**
> In all cases, we recommend using a Python or Anaconda virtual environment.

> **NOTE**
> The `snorkelflow` SDK requires Python 3.8+ but recommends Python 3.10+

> **NOTE**
> The `snorkelflow` wheel requires `pip >= 19.0`,
> run `python3 -m pip install --upgrade pip` to upgrade pip.

Installing from a wheel file

Install the `snorkelflow` package by running:

```
python3 -m pip install path/to/snorkelflow-xx.yy.zz-py3-none-any.whl
```

replacing `path/to/snorkelflow-xx.yy.zz-py3-none-any.whl` with the correct path and file name. There are a number of `extras` defined for the `snorkelflow` SDK which specifies external libraries needed for extra functionality.

- `data`: dependencies needed for the dataset loading utilities in `snorkelflow`
- `debug`: dependencies needed to generate support bundles for debugging performance
- `kubernetes`: dependencies needed to generate Kubernetes configurations
- `sdk`: dependencies needed for the software development kit interface

For example, to install the `sdk` extras, run:

```
python3 -m pip install 'path/to/snorkelflow-xx.yy.zz-py3-none-any.whl[sdk]'
```

replacing `path/to/snorkelflow-xx.yy.zz-py3-none-any.whl` with the correct path and file name. Installing from a Docker image The `snorkelflow` SDK will be bundled in a Docker image called `snorkelai/snorkelflow-whl`. Extract the `snorkelflow` SDK by running:

```
VERSION=<snorkelflow version>
LOCAL_WHL_PATH=/tmp/whl
mkdir -p $LOCAL_WHL_PATH
CONTAINER_ID=$(docker create snorkelai/snorkelflow-whl:$VERSION bash)
docker cp $CONTAINER_ID:/ $LOCAL_WHL_PATH
```

replacing `<snorkelflow version>` with the provided version of `snorkelflow`. Once the wheel file is extracted, follow the instructions for Installing from a wheel file using the wheel file located in `$LOCAL_WHL_PATH`.

Authenticating your SDK Client

SDK usage outside the Snorkel AI Data Development Platform requires authentication via an API key. Instructions on SDK authentication can be found in Snorkel API Authentication.

Installing from a standalone Anaconda channel

Instructions for installation from a standalone Anaconda channel are provided separately by Snorkel AI.

Snorkel AI Data Development Platform

Installing the Snorkel AI Data Development Platform

Follow the Snorkel Kubernetes Installation overview to begin installing the platform.

AWS Infrastructure Setup - Terraform (Recommended)

Overview

This document will guide you through creating and deploying a new Kubernetes cluster to your existing AWS account, including the creation of all required resources. This process will be completed through the AWS web interface and will involve running our Terraform configuration.

Assumptions/Prerequisites

By default, the Snorkel AI Data Development Platform is deployed into an existing AWS VPC and subnets, and configures new infrastructure in your AWS account. We start off with a couple assumptions:

- You have access to an AWS account
- You have access to a VPC, along with 3 or more subnets within that VPC
- You have a domain that you own, as well as a certificate for TLS
  - The domain is configured in Route53
  - The certificate is configured in AWS Certificate Manager
- The individual executing these setup tasks has admin access to the VPC

Subnet Tagging

To ensure the kubernetes cluster created later in this document recognizes the 3 or more subnets provided, we need to tag them each with 2 tags:

- Navigate to subnets in the AWS Console
  - VPC -> Subnets
- For each of the 3 subnets, add a tag for elb (for the aws-load-balancer-controller later in this document) and for the cluster
  - Subnet -> Manage tags
  - Add an elb tag
    - For private subnets
      - Key: kubernetes.io/role/internal-elb
      - Value: 1
    - For public subnets
      - Key: kubernetes.io/role/elb
      - Value: 1
  - Add a cluster tag
    - Key: kubernetes.io/cluster/snorkel-flow-aws
    - Value: shared

Resource Creation

In this step, we use the terraform package provided by the Snorkel team to create the required infrastructure needed to run the Snorkel AI Data Development Platform:

- Download and extract the AWS terraform files, and store it somewhere accessible on the machine that has the required admin permissions to the VPC

- `cd` into the top-level directory with the `variables.tf` file as well as the `terraform.tfvars` file. The `terraform.tfvars` file is used to inject your configurations into the `variables.tf` file.
- Inside terraform.tfvars, note the required variables and fill them out in the file
  - vpc_id: ID of the existing VPC to deploy into
  - subnets: a list of existing subnets in the vpc (3 or more) to use for the Snorkel AI Data Development Platform
  - domain_filter: the name of an existing hosted zone in route53, to use as the domain for the Snorkel AI Data Development Platform urls

Once you are happy with the inputted variables, run terraform and wait for the resources to spin up, which can take around ~15-20 minutes:

- ```
  terraform init
  ```
- ```
  terraform plan
  ```
- ```
  terraform apply
  ```

Once terraform finishes, we should have a fresh k8s cluster to deploy the Snorkel AI Data Development Platform into.
  ```

AWS Infrastructure Setup - Manual

Overview

This document covers the steps required to deploy a new Kubernetes cluster to your existing AWS account, including the creation of all required resources. This process will be completed through the AWS CLI as well as the web interface.

Assumptions/Prerequisites

By default, the Snorkel AI Data Development Platform is deployed into an existing AWS VPC and subnets, and configures new infrastructure in your AWS account. We start off with a couple assumptions:

- You have access to an AWS account
- You have access to a VPC, along with 3 or more subnets within that VPC
- You have a domain that you own, as well as a certificate for TLS
  - The domain is configured in Route53
  - The certificate is configured in AWS Certificate Manager
- The individual executing these setup tasks has admin access to the VPC

Part 1: Subnet Tagging

To ensure the kubernetes cluster created later in this document recognizes the 3 or more subnets provided, we need to tag them each with 2 tags:

- Navigate to subnets in the AWS Console
  - VPC -> Subnets
- For each of the 3 subnets, add a tag for elb (for the aws-load-balancer-controller later in this document) and for the cluster
  - Subnet -> Manage tags
  - Add an elb tag
    - For private subnets
      - Key: kubernetes.io/role/internal-elb
      - Value: 1
    - For public subnets
      - Key: kubernetes.io/role/elb
      - Value: 1
  - Add a cluster tag
    - Key: kubernetes.io/cluster/snorkel-flow-aws

      > NOTE
      >
      > `snorkel-flow-aws` is the default eks cluster name. If you use a different cluster name during EKS setup, replace `snorkel-flow-aws` with your cluster name.

    - Value: shared

Part 2: Security Group Setup

Create a security group for Snorkel inside of your existing VPC:

- Navigate to Security Groups in the AWS Console
  - EC2 -> Network and Security -> Security Groups
- Create a new Security Group with name `snorkel-flow-aws` in the existing VPC, allowing all outbound traffic and TCP inbound traffic
  - Create security group
    - Security group name: snorkel-flow-aws
    - Description: snorkel-flow-aws
    - VPC: Existing VPC ID
    - Inbound traffic -> Add rule
      - Type: All traffic
      - Source: Anywhere-IPV4
    - Outbound traffic -> Add rule
      - Type: All traffic
      - Source: Anywhere-IPV4
    - Outbound traffic -> Add rule
      - Type: All traffic
      - Source: Anywhere-IPV6

Part 3: Create an EFS Instance

Create an EFS instance for Snorkel inside of your existing VPC:

- Navigate to EFS in the AWS Console
- Create a new EFS instance with name `snorkel-flow-aws` in the existing VPC, with existing subnets as the mount points
  - Create file system -> Customize
    - File system settings
      - Name: snorkel-flow-aws
      - Storage class: standard
      - Enable automated backups
      - Enable encryption at rest
      - Throughput mode: Bursting
    - Network access
      - VPC: your existing VPC
      - Mount targets
        - Your 3 or more subnets, along with their associated availability zone
        - Security group: add `snorkel-flow-aws` created in Part 2
    - Create

Create an access point at the root directory of your EFS volume, with POSIX and creation permissions for user 99:

- Navigate to the new EFS in the AWS Console
  - EFS -> snorkel-flow-aws -> Access points
- Create a new access point
  - Create access point
    - Name: snorkel-flow-aws

    - Root directory path: /snorkel-flow/data
    - POSIX User ID: 99
    - POSIX Group ID: 99
    - POSIX Secondary group IDs: 99
    - Root directory owner user ID: 99
    - Root directory owner group ID: 99
    - Root directory access point permissions: 775
  - Create access point

Part 4: Create an EKS cluster

Create an IAM role for the EKS cluster backplane:

- Navigate to IAM roles in the AWS Console
  - IAM -> Roles
- Create a new role snorkel-flow-aws-cluster with AmazonEKSClusterPolicy and AmazonEKSVPCResourceController attached
  - Create role
    - Select trusted entity
      - Custom trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EKSClusterAssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": "eks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- Add permissions
  - Add the `AmazonEKSClusterPolicy` and `AmazonEKSVPCResourceController` policies
- Name, review, and create
  - Role name: snorkel-flow-aws-cluster
  - Create Role

Create an EKS cluster for Snorkel inside of your existing VPC:

- Navigate to EKS (Elastic Kubernetes Service) in the AWS Console
- Create a new EKS cluster with name `snorkel-flow-aws` in the existing VPC
  - Add cluster -> Create
    - Configure cluster
      - Name: snorkel-flow-aws
      - Kubernetes version: default

    - Cluster service role: snorkel-flow-aws-cluster
  - Specify networking
    - VPC: your existing VPC
    - Subnets: your existing 3 or more subnets
    - Security Groups: snorkel-flow-aws
  - Select addons
    - Amazon VPC CNI
    - Kube-proxy
    - CoreDNS
  - Create

Part 5: Add a node group to the EKS cluster

Create an IAM role for the EKS node group:

- Navigate to IAM roles in the AWS Console
  - IAM -> Roles
- Create a new role snorkel-flow-aws-cluster-node with AmazonEC2ContainerRegistryReadOnly, AmazonEKS_CNI_Policy, and AmazonEKSWorkerNodePolicy attached
  - Create role
    - Select trusted entity
      - Custom trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EKSNodeAssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- Add permissions
  - Add the `AmazonEC2ContainerRegistryReadOnly`, `AmazonEKS_CNI_Policy`, and `AmazonEKSWorkerNodePolicy` policies
- Name, review, and create
  - Role name: snorkel-flow-aws-cluster-node
  - Create Role

Create a node group for the EKS cluster:

- Navigate to the EKS cluster created in **Part 4** in the AWS Console
  - EKS (Elastic Kubernetes Service) -> Clusters -> snorkel-flow-aws -> Compute
- Add a new node group

- Add node group
  - Configure node group
    - Name: snorkel-flow-aws
    - Node IAM role: snorkel-flow-aws-cluster-node
  - Set compute and scaling configuration
    - Node group compute configuration
      - AMI type: Amazon Linux 2 (AL_x86_64)
      - Capacity type: On-Demand
      - Instance Types: m5.8xlarge
      - Disk Size: 1024GiB
    - Node group scaling configuration
      - Desired size: 2 nodes
      - Minimum size: 2 nodes
      - Maximum size: 2 nodes
  - Specify Networking
    - Subnets: your 3 or more existing subnets
  - Create

Part 6: Create aws-efs-csi-driver, aws-load-balancer-controller, external-dns, and aws-ebs-csi-driver operators via helm

Create IAM policies for aws-efs-csi-driver, aws-load-balancer-controller, external-dns, and aws-ebs-csi-driver operators.

- Navigate to IAM policies in the AWS Console
  - IAM -> Policies
- Create a new policy lb-controller-irsa-aws-load-balancer-controller
  - Create policy
    - Policy editor -> JSON

```
{
    "Statement": [
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:DescribeSSLPolicies",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeListenerCertificates",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeTags",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeInstances",
                "ec2:DescribeCoipPools",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeAddresses",
                "ec2:DescribeAccountAttributes"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "wafv2:GetWebACLForResource",
                "wafv2:GetWebACL",
                "wafv2:DisassociateWebACL",
                "wafv2:AssociateWebACL",
                "waf-regional:GetWebACLForResource",
```

```
                "waf-regional:GetWebACL",
                "waf-regional:DisassociateWebACL",
                "waf-regional:AssociateWebACL",
                "shield:GetSubscriptionState",
                "shield:DescribeProtection",
                "shield:DeleteProtection",
                "shield:CreateProtection",
                "iam:ListServerCertificates",
                "iam:GetServerCertificate",
                "cognito-idp:DescribeUserPoolClient",
                "acm:ListCertificates",
                "acm:DescribeCertificate"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:RevokeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:CreateSecurityGroup",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:CreateTags",
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
                },
                "StringEquals": {
                    "ec2:CreateAction": "CreateSecurityGroup"
                }
            },
            "Effect": "Allow",
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Action": [
                "ec2:DeleteTags",
                "ec2:CreateTags"
            ],
            "Condition": {
                "Null": {
```

```
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "true",
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            },
            "Effect": "Allow",
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Action": [
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DeleteSecurityGroup",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Condition": {
                "Null": {
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:CreateLoadBalancer"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:CreateListener"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticloadbalancing:RemoveTags",
                "elasticloadbalancing:AddTags"
            ],
```

```
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "true",
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            },
            "Effect": "Allow",
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/net/*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/app/*/*"
            ]
        },
        {
            "Action": [
                "elasticloadbalancing:RemoveTags",
                "elasticloadbalancing:AddTags"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:listener/net/*/*/*",
                "arn:aws:elasticloadbalancing:*:*:listener/app/*/*/*",
                "arn:aws:elasticloadbalancing:*:*:listener-rule/net/*/*/*",
                "arn:aws:elasticloadbalancing:*:*:listener-rule/app/*/*/*"
            ]
        },
        {
            "Action": "elasticloadbalancing:AddTags",
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
                },
                "StringEquals": {
                    "elasticloadbalancing:CreateAction": [
                        "CreateTargetGroup",
                        "CreateLoadBalancer"
                    ]
                }
            },
            "Effect": "Allow",
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/net/*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/app/*/*"
            ]
        },
        {
            "Action": [
                "elasticloadbalancing:SetSubnets",
```

```
                "elasticloadbalancing:SetSecurityGroups",
                "elasticloadbalancing:SetIpAddressType",
                "elasticloadbalancing:ModifyTargetGroupAttributes",
                "elasticloadbalancing:ModifyTargetGroup",
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DeleteLoadBalancer"
            ],
            "Condition": {
                "Null": {
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*"
        },
        {
            "Action": [
                "elasticloadbalancing:SetWebAcl",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyRule",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:AddListenerCertificates"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

- Policy name: lb-controller-irsa-aws-load-balancer-controller
- Create policy
- Create a new policy external-dns-irsa-external-dns
  - Create policy
    - Policy editor -> JSON

```
{
    "Statement": [
        {
            "Action": "route53:ChangeResourceRecordSets",
            "Effect": "Allow",
            "Resource": "arn:aws:route53:::hostedzone/*",
            "Sid": "ChangeResourceRecordSets"
        },
        {
            "Action": [
                "route53:ListTagsForResource",
                "route53:ListResourceRecordSets",
                "route53:ListHostedZones"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Sid": "ListResourceRecordSets"
        }
    ],
    "Version": "2012-10-17"
}
```

- Policy name: external-dns-irsa-external-dns
- Create policy
- Create a new policy efs-csi-controller-aws-efs-csi-driver
  - Create policy
    - Policy editor -> JSON

```
{
    "Statement": [
        {
            "Action": [
                "elasticfilesystem:DescribeMountTargets",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:DescribeAccessPoints",
                "ec2:DescribeAvailabilityZones"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "elasticfilesystem:CreateAccessPoint",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/efs.csi.aws.com/cluster": "true"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "elasticfilesystem:TagResource",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "elasticfilesystem:DeleteAccessPoint",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

- Policy name: efs-csi-controller-aws-efs-csi-driver

- Create policy
- Create a new policy ebs-csi-controller-aws-ebs-csi-driver
  - Create policy
    - Policy editor -> JSON

```
{
    "Statement": [
        {
            "Action": [
                "ec2:ModifyVolume",
                "ec2:DetachVolume",
                "ec2:DescribeVolumesModifications",
                "ec2:DescribeVolumes",
                "ec2:DescribeTags",
                "ec2:DescribeSnapshots",
                "ec2:DescribeInstances",
                "ec2:DescribeAvailabilityZones",
                "ec2:CreateSnapshot",
                "ec2:AttachVolume"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:CreateTags",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateVolume",
                        "CreateSnapshot"
                    ]
                }
            },
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ]
        },
        {
            "Action": "ec2:DeleteTags",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ]
        },
        {
            "Action": "ec2:CreateVolume",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/ebs.csi.aws.com/cluster": "true"
                }
```

```
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:CreateVolume",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/CSIVolumeName": "*"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:CreateVolume",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/kubernetes.io/cluster/*": "owned"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:DeleteVolume",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:DeleteVolume",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/CSIVolumeName": "*"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:DeleteVolume",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/kubernetes.io/cluster/*": "owned"
```

```
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:DeleteSnapshot",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/CSIVolumeSnapshotName": "*"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "ec2:DeleteSnapshot",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
                }
            },
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

- `Policy name: ebs-csi-controller-aws-ebs-csi-driver`
- Create policy

Create associated IAM roles for aws-efs-csi-driver, aws-load-balancer-controller, external-dns, and aws-ebs-csi-driver operators.

- Navigate to IAM roles in the AWS Console
  - IAM -> Roles
- Create a new role lb-controller-irsa-aws-load-balancer-controller that refers to the lb-controller-irsa-aws-load-balancer-controller policy
  - Create role
    - Select trusted entity
      - Custom trust policy
        - **Note:** `${OIDC_URL}` is `OpenID Connect provider URL` of the EKS cluster created in **Part 4** without the `https://` and can be found on the EKS cluster's page (e.g. https://oidc.eks.us-west-2.amazonaws.com/id/64A3B40CAAD5443BD95F6A8B7889F272)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${ACCOUNT_ID}:oidc-
provider/${OIDC_URL}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${OIDC_URL}:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
    ]
}
```

- Add permissions
  - Add the `lb-controller-irsa-aws-load-balancer-controller` policy created in the prior steps
- Name, review, and create
  - Role name: lb-controller-irsa-aws-load-balancer-controller
  - Create Role
- Create a new role external-dns-irsa-external-dns that refers to the external-dns-irsa-external-dns policy
  - Create role
    - Select trusted entity
      - Custom trust policy
        - **Note:** `${OIDC_URL}` is `OpenID Connect provider URL` of the EKS cluster created in **Part 4** without the `https://` and can be found on the EKS cluster's page (e.g. https://oidc.eks.us-west-2.amazonaws.com/id/64A3B40CAAD5443BD95F6A8B7889F272)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${ACCOUNT_ID}:oidc-
provider/${OIDC_URL}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${OIDC_URL}:sub": "system:serviceaccount:kube-
system:external-dns"
                }
            }
        }
    ]
}
```

- Add permissions
  - Add the `external-dns-irsa-external-dns` policy created in the prior steps
- Name, review, and create
  - Role name: external-dns-irsa-external-dns
  - Create Role
- Create a new role efs-csi-controller-aws-efs-csi-driver that refers to the efs-csi-controller-aws-efs-csi-driver policy
  - Create role
    - Select trusted entity
      - Custom trust policy
        - **Note:** `${OIDC_URL}` is `OpenID Connect provider URL` of the EKS cluster created in **Part 4** without the `https://` and can be found on the EKS cluster's page (e.g. https://oidc.eks.us-west-2.amazonaws.com/id/64A3B40CAAD5443BD95F6A8B7889F272)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${ACCOUNT_ID}:oidc-
provider/${OIDC_URL}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${OIDC_URL}:sub": "system:serviceaccount:kube-system:efs-
csi-controller-sa"
                }
            }
        }
    ]
}
```

- Add permissions
  - Add the `efs-csi-controller-aws-efs-csi-driver` policy created in the prior steps
- Name, review, and create
  - Role name: efs-csi-controller-aws-efs-csi-driver
  - Create Role
- Create a new role ebs-csi-controller-aws-ebs-csi-driver that refers to the ebs-csi-controller-aws-ebs-csi-driver policy
  - Create role
    - Select trusted entity
      - Custom trust policy
        - **Note:** `${OIDC_URL}` is `OpenID Connect provider URL` of the EKS cluster created in **Part 4** without the `https://` and can be found on the EKS cluster's page (e.g. https://oidc.eks.us-west-2.amazonaws.com/id/64A3B40CAAD5443BD95F6A8B7889F272)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${ACCOUNT_ID}:oidc-
provider/${OIDC_URL}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${OIDC_URL}:sub": "system:serviceaccount:kube-system:ebs-
csi-controller-sa"
                }
            }
        }
    ]
}
```

- Add permissions
  - Add the `ebs-csi-controller-aws-ebs-csi-driver` policy created in the prior steps
- Name, review, and create
  - Role name: ebs-csi-controller-aws-ebs-csi-driver
  - Create Role

Create aws-efs-csi-driver, aws-load-balancer-controller, external-dns, and aws-ebs-csi-driver operators via helm charts.

- Connect to your cluster via kubectl
  - ```
    aws eks --region us-west-2 update-kubeconfig --name snorkel-flow-aws
    ```
  - ```
    kubectl get pods --all-namespaces
    ```
  - Should return coredns, aws-node, and kube-proxy pods
- Deploy aws-efs-csi-driver
  - ```
    helm repo add aws-efs-csi-driver https://kubernetes-sigs.github.io/aws-
    efs-csi-driver/
    ```
  - ```
    helm repo update aws-efs-csi-driver
    ```

    - ```
      helm install aws-efs-csi-driver aws-efs-csi-driver/aws-efs-csi-driver -n kube-system --set controller.serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::${ACCOUNT_ID}:role/efs-csi-controller-aws-efs-csi-driver" --version 2.4.7
      ```
    - ```
      kubectl get pods -n kube-system
      ```
    - Confirm that this now has the new efs-csi pods
- Deploy aws-load-balancer-controller
    - ```
      helm repo add eks [https://aws.github.io/eks-charts](https://aws.github.io/eks-charts)
      ```
    - ```
      helm repo update eks
      ```
    - ```
      helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=snorkel-flow-aws --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::${ACCOUNT_ID}:role/lb-controller-irsa-aws-load-balancer-controller" --version 1.5.5
      ```
- Deploy external-dns
    - create an OIDC provider for the cluster first (follow instructions in https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html)
    - ```
      helm repo add external-dns [https://kubernetes-sigs.github.io/external-dns/](https://kubernetes-sigs.github.io/external-dns/)
      ```
    - ```
      helm repo update external-dns
      ```
    - ```
      helm install external-dns external-dns/external-dns -n kube-system --set provider=aws --set txtOwnerId=snorkel-flow-aws --set txtPrefix=txt- --set domainFilters={$DOMAIN} --set extraArgs={--aws-prefer-cname} --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::${ACCOUNT_ID}:role/external-dns-irsa-external-dns" --version 1.13.0
      ```
    - Note: $DOMAIN is the domain that you own
- Deploy aws-ebs-csi-driver
    - ```
      helm repo add aws-ebs-csi-driver [https://kubernetes-sigs.github.io/aws-ebs-csi-driver](https://kubernetes-sigs.github.io/aws-ebs-csi-driver)
      ```
      ```

- ```
  helm repo update aws-ebs-csi-driver
  ```
- ```
  helm install aws-ebs-csi-driver aws-ebs-csi-driver/aws-ebs-csi-driver -n kube-system --set controller.serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::${ACCOUNT_ID}:role/ebs-csi-controller-aws-ebs-csi-driver" --version 2.21.0
  ```

Part 7: AWS-Specific Configurations for Snorkel

While deploying the Snorkel AI Data Development Platform in AWS (link), there are a couple specific configurations that are needed to be set for helm to work with the infrastructure set up above:

- For annotations on the UI ingress, we'll need the following:

```
alb.ingress.kubernetes.io/scheme: internal
external-dns.alpha.kubernetes.io/ttl: "60"
alb.ingress.kubernetes.io/certificate-arn: $AWS_CERTIFICATE_ARN
alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS":443}]'
alb.ingress.kubernetes.io/ssl-redirect: "443"
alb.ingress.kubernetes.io/load-balancer-attributes:
idle_timeout.timeout_seconds=600
alb.ingress.kubernetes.io/success-codes: 200-302
alb.ingress.kubernetes.io/target-type: "ip"
external-dns.alpha.kubernetes.io/hostname: $SUBDOMAIN.$DOMAIN
alb.ingress.kubernetes.io/group.name: $SUBDOMAIN.$DOMAIN
alb.ingress.kubernetes.io/healthcheck-path: /health
```

- $AWS_CERTIFICATE_ARN is the certificate configured for your domain in AWS Certificate Manager
- $SUBDOMAIN is the subdomain chosen for the Snorkel AI Data Development Platform's url
- $DOMAIN is a domain configured in route53
- Note: a similar set of annotations are required for each ingress that the Snorkel AI Data Development Platform uses
- For the data volume driver, we'll need the following:

```
csi:
  driver: efs.csi.aws.com
  volumeHandle: $EFS_ID::$ACCESS_POINT_ID
```

- $EFS_ID is the id of the EFS created earlier for the Snorkel AI Data Development Platform
- $ACCESS_POI

Azure Infrastructure Setup - Terraform (Recommended)

This page walks through the steps that are required to deploy a new Kubernetes cluster to your existing Azure account, including the creation of all required resources. This process requires both the Azure CLI and the web interface.

Pre-requisites and Azure features

In this section, we walk through the pre-requisites for creating a new cluster in Azure, and enable any Azure features that the Snorkel AI Data Development Platform requires to run.

To begin, you will need a few command line tools. Install the current versions of the following tools if you do not have them already installed.

- Instructions to install az
- Instructions to install helm
- Instructions to install kubectl

Now you can run `az login` to login with an Azure account or service principal with appropriate admin permissions. Ensure that you specify the `--tenant` flag.

We require Azure features that are currently only available in Preview. You must ensure that they are activated and registered for your account. You can register for the required Azure Preview features with the following commands.

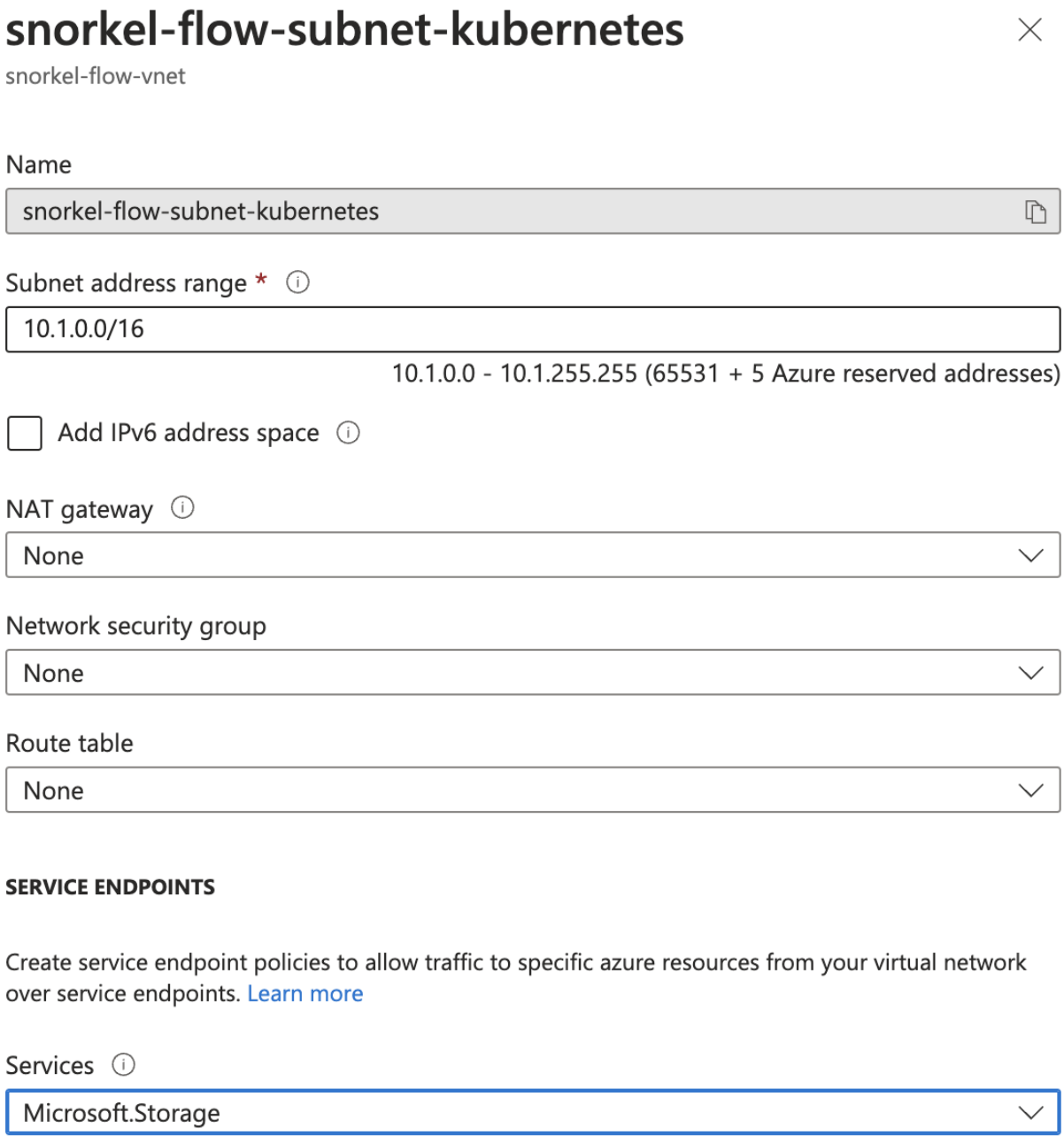
- Azure Preview
  - ```
    az extension add --name aks-preview
    ```
  - ```
    az extension update --name aks-preview
    ```
- Azure Files NFS mounting in AKS
  - ```
    az feature register --name AllowNfsFileShares --namespace
    Microsoft.Storage
    ```
  - ```
    az provider register --namespace Microsoft.Storage
    ```
  - Wait around 15 minutes, and ensure that outputs are "Registered"

    ```
    az feature show --name AllowNfsFileShares --namespace Microsoft.Storage -
    -query properties.state
    ```

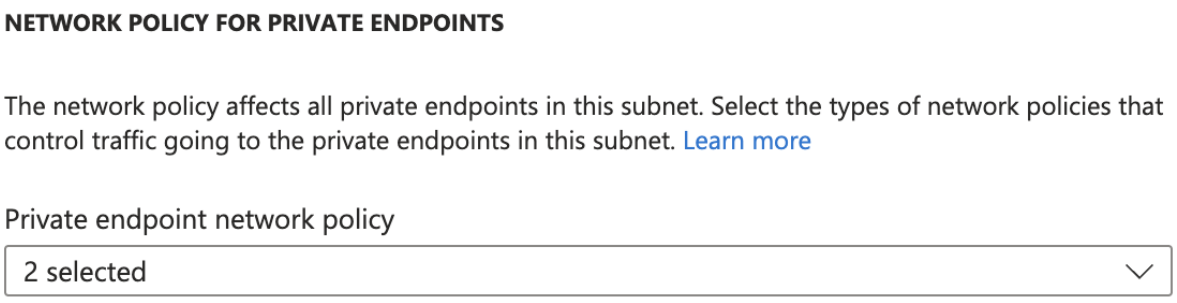
Provision the required infrastructure and run Terraform

Once you have set up the required pre-requisites, you can create the cluster that the Snorkel AI Data Development Platform runs in, alongside any required resources.

1. Create a new resource group where your account / service principal that you previously logged in with is an **Owner** of the resource group.
2. Ensure that an Azure AD group where members will have admin access to the newly created cluster exists (your account / service principal is an owner and/or member of this group).
3. Create a new virtual network with a subnet (suggest at least /18 address space) in the resource group.
4. After creating the subnet, go to the Azure UI and find the subnet in the Virtual Networks page. Ensure that **Microsoft.Storage** is checked under **Service Endpoints** in the subnet configuration (see the screenshot below).



5. Select both options under **Network policy for private endpoints** (i.e., **Network security groups** and **Route tables**).



6. You can use your existing Azure DNS zone, or create a new one for the Snorkel AI Data Development Platform (learn how to create an Azure DNS with a delegated domain).

7. [OPTIONAL] If you would like to configure TLS, then create a new Azure Key Vault and generate and/or import certificates:

```
az keyvault create -g snorkel-flow-rg -l <Location> -n <KeyVaultName> --
enable-rbac-authorization
```

8. Gather the required variables:

   - Open variables.tf to see the resource groups that are defined. Note that getting the subnet ID may require you to use the CLI if it is not available in the UI.

     ```
     az network vnet subnet list --resource-group snorkel-flow-rg --vnet-name
     snorkel-flow-vnet | grep id
     ```

   - Open terraform.tfvars and fill in the exposed variables. Here are a few important ones:
     - subnet_id: The ID of an existing subnet to deploy into.
     - admin_group_id: The ID of the group that will be granted admin access to the new cluster.
     - vm_size_node: The VM size of the cluster node.
     - gpu_support: Enable GPU support for the cluster.
     - vm_size_gpu_node: The VM size / type of the GPU node. This requires gpu_support to be enabled.

9. Initialize Terraform.

```
terraform init
```

10. Check that the resources to be created are correct.

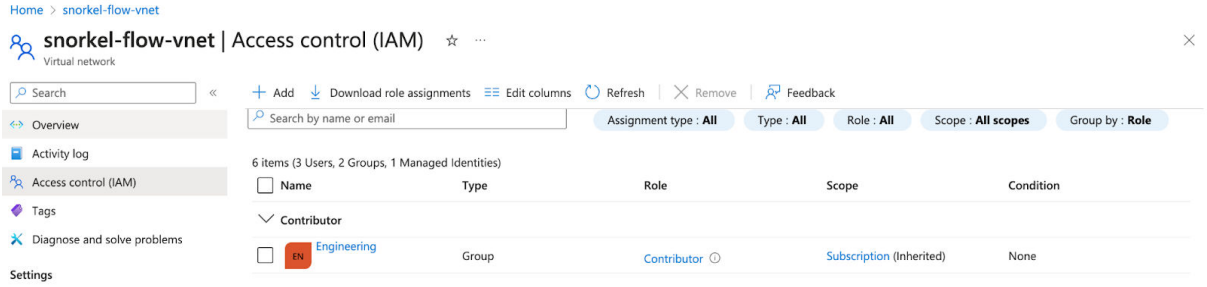
```
terraform plan
```

11. If everything looks good, then you can apply the configuration.

```
terraform apply
```

12. Configure the Virtual Network Role Assignment for the manually provisioned resources:

13. Go to the overview page of the virtual network that you created in step 3, and click **Access control (IAM)**



14. Click **Add**, then click **Add role assignment**.

15. Select **Network Contributor**, then click **next**.

16. Check **Managed identity** (this is next to **Assign access to**).

17. Click **Select members**, then select **Kubernetes service** (this is under **Managed identity**).

18. Select the **snorkel-flow-cluster** (ensure it is the cluster that you created by terraform in the previous step), then click **Select**.

19. Click **Review and assign** to finish.

20. Set up kubectl access.

```
az aks get-credentials --resource-group snorkel-flow-rg --name snorkel-flow-cluster --admin
```

    If this command doesn't work:

    1. Go to the cluster overview page.
    2. Click the **cluster configuration** tab on the left-hand side.
    3. Ensure that the **Kubernetes local accounts** checkbox is checked.

21. Enable the ingress controller add-on.

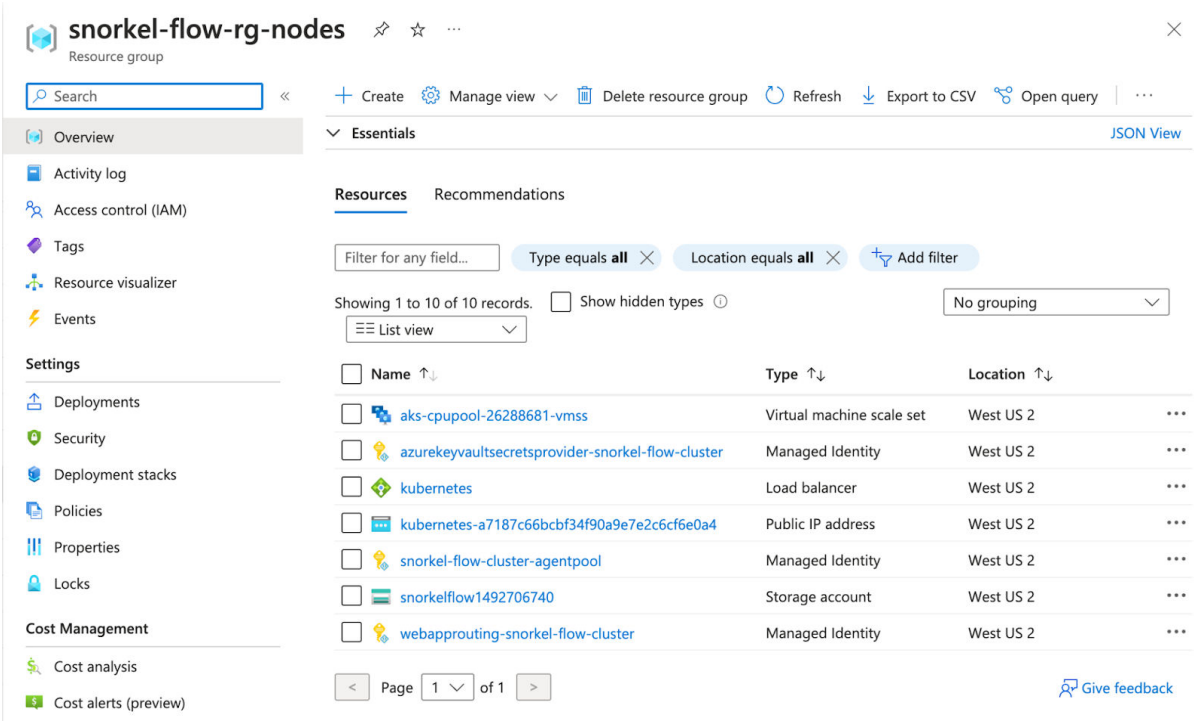
```
az aks enable-addons -g snorkel-flow-rg -n snorkel-flow-cluster --addons azure-keyvault-secrets-provider,web_application_routing --enable-secret-rotation
```

22. Get the object (principal) ID of the managed identity of the web-app-routing add-on from the Azure UI and save it under the name MANAGEDIDENTITY_OBJECTID.

23. From the Azure console, go to **Resource Groups**.

24. Select the resource group that was created by the Snorkel AI Data Development Platform installation. This is not the one that is defined in variables.tf, but rather the other automatically created one that ends in **-nodes** (e.g., **snorkel-flow-rg-nodes**).

25. Search for the appropriate managed identity. This should start with **webapprouting-** (e.g., **webapprouting-snorkel-flow-cluster** at the bottom of the list below).



26. To enable automatic management of DNS records, connect the ingress controller add-on to Azure DNS. Get the resource ID of the Azure DNS zone that you created in step 5 from the Azure UI (select the DNS Zone, then click **Properties**), then save it under the name ZONEID.
27.
```
az role assignment create --role "DNS Zone Contributor" --assignee $MANAGEDIDENTITY_OBJECTID --scope $ZONEID
```
28.
```
az aks addon update -g snorkel-flow-rg -n snorkel-flow-cluster --addon web_application_routing --dns-zone-resource-id=$ZONEID
```
29. Connect the ingress controller add-on to Azure Key Vault (for automatic management of TLS certs).
30.
```
KEYVAULTID=$(az keyvault show --name <KeyVaultName> --query "id" --output tsv)
```
31.
```
az role assignment create --role "Key Vault Secrets User" --assignee $MANAGEDIDENTITY_OBJECTID --scope $KEYVAULTID
```

Azure Infrastructure Setup - Manual

Overview

This document covers the steps required to deploy a new Kubernetes cluster to your existing Azure account, including the creation of all required resources. This process will be completed through the Azure CLI as well as the web interface.

Prerequisites and Azure Features

In this step, we will go over the prerequisites for creating a new cluster in Azure, and enable any Azure features that the Snorkel AI Data Development Platform requires to run.

To begin, you will need a few command line tools. Install the current versions of these tools if they are not already installed.

1. Instructions to install az
   - https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt#option-1-install-with-one-command
2. Instructions to install helm
   - https://helm.sh/docs/intro/install/#from-script
3. Instructions to install kubectl
   - https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/#install-kubectl-binary-with-curl-on-linux
4. Run

   ```
   az login
   ```

   to login with an Azure account or service principle with appropriate admin permissions (ensure to specify the --tenant flag)
5. We require Azure features that are currently only available in Preview. Ensure they are activated and registered for your account, you can register for the required Azure Preview features with the following commands.
   - Azure Preview
     - ```
       az extension add --name aks-preview
       ```
     - ```
       az extension update --name aks-preview
       ```
   - Azure Files NFS mounting in AKS
     - ```
       az feature register --name AllowNfsFileShares --namespace Microsoft.Storage
       ```
     - ```
       az provider register --namespace Microsoft.Storage
       ```

- Wait around 15 minutes, and ensure

```
az feature show --name AllowNfsFileShares --namespace
Microsoft.Storage --query properties.state
```

outputs "Registered"

Manually provision required infrastructure

In this step, we will be creating the cluster that the Snorkel AI Data Development Platform runs in, alongside any required resources.

1. Create a new resource group where your account / service principal you previously logged in with is an "Owner" of the resource group)
2. Make sure that an Azure AD group exists whose members will have admin access to the newly created cluster (where your account / service principal is an owner and/or member)
3. Create a new virtual network with a subnet (suggest at least /18 address space) in the resource group created above.
4. After creating the subnet, go to the Azure UI, find the subnet under the **Virtual Networks** page, and ensure **Microsoft.Storage** is checked under **Service Endpoints** in the subnet configuration.
5. Select both options under **Network policy for private endpoints** (**Network security groups** and **Route tables**).
6. You can use your existing Azure DNS zone, or create a new one for the Snorkel AI Data Development Platform.
    1. Azure DNS with a delegated domain -> https://learn.microsoft.com/en-us/azure/dns/dns-delegate-domain-azure-dns
7. [OPTIONAL] If you would like to configure TLS, then create a new Azure Key Vault
    1.
```
az keyvault create -g snorkel-flow-rg -l <Location> -n <KeyVaultName> --
enable-rbac-authorization
```
    2. Generate and/or import certificates (https://medium.com/@jibinpb/lets-encrypt-certificate-with-azure-dns-b9ed32ae5aee)
8. Gather required variables
    1. snorkel_rg_name
        1. from step 4a
    2. vm_size_node
        1. we recommend D32ds_v5
    3. admin_group_name
        1. from step 4b
    4. vnet_name and subnet_name
        1. from step 4c

9. Go to the Azure web portal, and click through to provision the infrastructure (replace the variables in `{{ }}` with their appropriate values from 5)
    1. Cluster
        1. Navigate to the **Kubernetes services** page from the top-level search and click **Create** then **Create a Kubernetes cluster**
        2. Select the resource group as `{{ snorkel\_rg\_name }}`
        3. For **Kubernetes cluster name**, put `snorkel-flow-cluster`
        4. For **Primary node pool**, set **Scale method** to **Manual** and **Node count** to **1**
        5. Click **Next: Node pools**
        6. Under **Node pools** Click **Add node pool**
        7. For **Node pool name**, put cpupool
        8. Check all availability zones boxes available
        9. For **Node size**, search for `{{ vm\_size\_node }}` , click the option displayed, and click **Select**
        10. Set **Scale method** to **Manual** and **Node count** to **2**
        11. Click **Next: Access**
        12. For **Authentication and Authorization**, select **Azure AD authentication with Azure RBAC**
        13. Click **Next: Networking**
        14. For **Network configuration**, select **Azure CNI**
        15. Select `{{ vnet\_name }}` and `{{ subnet\_name }}`.
        16. Set **DNS name prefix** to be snorkel-flow-cluster
        17. Click **Next: Integrations**
        18. Click **Next: Advanced**
        19. Set **Infrastructure resource group** to be snorkel-flow-cluster-nodes
        20. Click **Review + create**
        21. Click **Create** to finish
    2. Storage Account
        1. Navigate to the **Storage accounts** page from the top-level search and click **Create**
        2. Select the resource group as what was defined in the **Infrastructure resource group** from the cluster creation step
        3. Type in a unique storage account name (can only have lowercase letters and numbers)
        4. Select the region to be the same as the `{{ snorkel\_rg\_name }}` resource group
        5. Select the performance to be **Premium**
        6. Select the premium account type to be **File shares**
        7. Select the redundancy to be **ZRS**
        8. Click **Next: Advanced**
        9. Uncheck **Require secure transfer for REST API operations**
        10. Click **Next Networking**
        11. Under **Network access**, check **Enabled from selected virtual networks and IP addresses**
        12. Select the virtual network name and subnet name
        13. Click **Review**
        14. Click **Create** to finish

3. Cluster Role Assignment
    1. Go to the overview page of the newly created cluster, and click **Access control (IAM)**.
    2. Click **Add**, then click **Add role assignment**
    3. Search for **Azure Kubernetes Service Cluster Admin Role**, and select it then click next
    4. Click **Select members**, then search by `{{ admin\_group\_name }}` and select it then click **Next**.
    5. Click **Review and assign** to finish
4. Virtual Network Role Assignment
    1. Go to the overview page of the manually created virtual network from step 4, and click **Access control (IAM)**.
    2. Click **Add**, then click **Add role assignment**
    3. Search for **Network Contributor**, and select it then click next
    4. Check **Managed identity** next to **Assign access to**
    5. Click **Select members**, select **Kubernetes service** under **Managed identity**, then select the `snorkel-flow-cluster` (ensure it is the right cluster in the previously created resource group), and click **Select**
    6. Click **Review and assign** to finish

10. Set up kubectl access
11. ```
    az aks get-credentials --resource-group {{ snorkel_rg_name }} --name snorkel-flow-cluster --admin
    ```

If this command doesn't work, you should go to the cluster overview page, go to the **cluster configuration** tab on the left hand side, and ensure the **Kubernetes local accounts** checkbox is checked.

11. Complete cluster setup by creating and applying the following yaml files to the cluster.
    Replace the variables in `{{ }}` with their appropriate values - the storage account name can be seen from the Storage Account **Overview** tab, and the storage account access key can be seen from the Storage Account **Access keys** tab.
12. Create and apply namespace.yaml to create the namespace for the Snorkel AI Data Development Platform
    1. ```
       kubectl apply -f namespace.yaml
       ```

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    meta.helm.sh/release-name: snorkel-flow
    meta.helm.sh/release-namespace: snorkel-flow
  labels:
    app.kubernetes.io/managed-by: Helm
  name: snorkel-flow
```

2. Create and apply storageaccountsecret.yaml

1. ```
   kubectl apply -f storageaccountsecret.yaml
   ```

```
apiVersion: v1
stringData:
  azurestorageaccountkey: {{ STORAGE_ACCOUNT_KEY }}
  azurestorageaccountname: {{ STORAGE_ACCOUNT_NAME }}
kind: Secret
metadata:
  annotations:
    meta.helm.sh/release-name: snorkel-flow
    meta.helm.sh/release-namespace: snorkel-flow
  labels:
    app.kubernetes.io/managed-by: Helm
  name: secret-storage-account
  namespace: snorkel-flow
type: Opaque
```

3. Create and apply storageclass.yaml
   1. ```
      kubectl apply -f storageclass.yaml
      ```

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: snorkel-flow-sc
parameters:
  protocol: nfs
  secretName: secret-storage-account
  secretNamespace: snorkel-flow
  storageAccount: {{ STORAGE_ACCOUNT_NAME }}
provisioner: file.csi.azure.com
reclaimPolicy: Retain
volumeBindingMode: Immediate
```

12. Enable the ingress controller add on (https://learn.microsoft.com/en-us/azure/aks/web-app-routing?tabs=without-osm)
13. ```
    az aks enable-addons -g snorkel-flow-rg -n snorkel-flow-cluster --addons azure-keyvault-secrets-provider,web_application_routing --enable-secret-rotation
    ```
14. get the object (principal) ID of the managed identity of the web-app-routing add on from the Azure UI and save it under the name MANAGEDIDENTITY_OBJECTID
    1. From the Azure console, go to **Resource Groups**, select the resource group created by the Snorkel AI Data Development Platform installation (not the one defined in variables.tf, but rather the other
    ```

automatically created one that ends in “-nodes”, for example “snorkel-flow-rg-nodes”), and search for the appropriate managed identity (should start with “webapprouting-”, e.g. “webapprouting-snorkel-flow-cluster”).

15. Next, in order for automatic management of DNS records we will connect the ingress controller add-on to Azure DNS. Get the resource ID of the Azure DNS zone you created in step 4 from the Azure UI (select the **DNS Zone**, click **Properties**) and save it under the name ZONEID
16.
    - ```
      az role assignment create --role "DNS Zone Contributor" --assignee $MANAGEDIDENTITY_OBJECTID --scope $ZONEID
      ```
    - ```
      az aks addon update -g snorkel-flow-rg -n snorkel-flow-cluster --addon web_application_routing --dns-zone-resource-id=$ZONEID
      ```
17. Connect the ingress controller add-on to Azure Key Vault (for automatic management of TLS certs)
    - ```
      KEYVAULTID=$(az keyvault show --name <KeyVaultName> --query "id" --output tsv)
      ```
    - ```
      az role assignment create --role "Key Vault Secrets User" --assignee $MANAGEDIDENTITY_OBJECTID --scope $KEYVAULTID
      ```

GCP Infrastructure Setup - Terraform (Recommended)

Overview

This document will guide you through creating and deploying a new Kubernetes cluster to your existing Google Cloud account, including the creation of all required resources. This process will be completed through the gcloud CLI, the Google Cloud web interface and will involve running the Snorkel AI Data Development Platform's Terraform configuration.

Prerequisites and Google Cloud Platform Features:

Complete these prequisites for creating a new cluster in GCP, and enable any GCP features that the Snorkel AI Data Development Platform requires to run.

To begin, you will need a few command line tools. Install the current versions of these tools if they are not already installed.

1. Instructions to install Helm on your local machine:
   - https://helm.sh/docs/intro/install/
2. Instructions to install Terraform on your local machine:
   - https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli
3. Create a new project on Google Cloud with associated billing account
4. The user account that will perform the next steps will also require certain user permissions
5. List of permissions that will most likely be encompassing, but could also contain some that we don't require.
   - https://docs.gcp.databricks.com/administration-guide/cloud-configurations/gcp/permissions.html
6. Create a DNS zone with an associated domain that you own, or if there is a pre-existing DNS zone then you can skip this step.
7. Install the gcloud CLI and configure access to your project via gcloud init:
   - https://cloud.google.com/sdk/docs/downloads-interactive
8. Enable the required gcloud APIs

Cluster Deployment (Terraform)

Next, execute a Terraform configuration to create the cluster that the Snorkel AI Data Development Platform runs in, alongside any required resources.

1. Download and extract the terraform files, you should be in the directory containing the various `.tf` files and the `variables.tf` file.
2. At this time, configure `provider.tf` inside of the services folder to point towards a Cloud Storage bucket of your choice - note that the name for the bucket has to be globally unique. If this GCS bucket is set up, then the state of the terraform installation will be stored inside this bucket, allowing it to persist.
3. Inside `variables.tf`, note the required variables and fill them out in the file.
   - `project`: The name of the project on Google Cloud
   - `region`: The region the cluster will be located, ex: us-central1

- `zone`: The specific zone the cluster will be located, ex: us-central1-c
- `cluster_name`: Name of the created cluster
- `node_count`: Number of compute nodes that will be created (default 4)
- `machine_type`: Type of compute node that will be added (default e2-standard-32)
- `domain`: The domain you own, in order to set up DNS, ex: snorkel.ai
- `subnet_id`: ID of an existing subnet to deploy into, this can be empty (default null)
- `vpc_id`: ID of an existing VPC to deploy into, this can be empty (default null)

4. Note that upon creation of a project, GCP will automatically provision a default subnet unless the `compute.skipDefaultNetworkCreation` option is set. This will also provision a default subnet for each region that GCP offers. If no `subnet_id` or `vpc_id` is provided to the terraform files, then the default will be used.
5. Once you are happy with the inputted variables, we can initialize Terraform with
   - ```
     terraform init
     ```
6. See what the planned resources to be created are with
   - ```
     terraform plan
     ```
7. If everything looks as expected, we can continue to apply the configuration with
   - ```
     terraform apply
     ```
8. Wait for the cluster to spin up, this can take around 10 minutes. The status of the cluster spin-up can be monitored in the Google Cloud web interface.
     ```

GCP Infrastructure Setup - Manual

Overview

This document will guide you through creating and deploying a new Kubernetes cluster to your existing Google Cloud account, including the creation of all required resources. This process will be completed through the gcloud CLI and the GCP web interface.

Pre-Requisites and Google Cloud Platform Features:

In this step, we will go over the pre-requisites for creating a new cluster in GCP, and enable any GCP features that the Snorkel AI Data Development Platform requires to run.

To begin, you will need a few command line tools. Install the current versions of these tools if they are not already installed.

1. Instructions to install Helm on your local machine:
   - https://helm.sh/docs/intro/install/
2. Instructions to install Terraform on your local machine:
   - https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli
3. Create a new project on Google Cloud with associated billing account
4. The user account that will perform the next steps will also require certain user permissions
5. List of permissions that will most likely be encompassing, but could also contain some that we don't require.
   - https://docs.gcp.databricks.com/administration-guide/cloud-configurations/gcp/permissions.html
6. Create a DNS zone with an associated domain that you own, or if there is a pre-existing DNS zone then you can skip this step.
7. Install the gcloud CLI and configure access to your project via gcloud init:
   - https://cloud.google.com/sdk/docs/downloads-interactive
8. Enable the required gcloud APIs

```
gcloud services enable dns.googleapis.com --project= snorkel-ai-gcp-
standardization
```

```
gcloud services enable compute.googleapis.com --project=snorkel-ai-gcp-
standardization
```

```
gcloud services enable container.googleapis.com --project=snorkel-ai-gcp-
standardization
```

```
gcloud services enable file.googleapis.com --project=snorkel-ai-gcp-
standardization
```

Cluster Deployment

In this step, we will be creating the cluster that runs the Snorkel AI Data Development Platform itself, as well as any required resources for the cluster.

1. Create a service account for the cluster
   - ```
     gcloud iam service-accounts create gke-cluster-admin-account --display-name "GKE Cluster Admin Account" --project=snorkel-ai-gcp-standardization
     ```
2. Get the email output from
   - ```
     gcloud iam service-accounts describe gke-cluster-admin-account@snorkel-ai-gcp-standardization.iam.gserviceaccount.com
     ```
3. Grant the roles to the email found in the output, here it is listed as gke-cluster-admin-account@snorkel-ai-gcp-standardization.iam.gserviceaccount.com but you will be using the output from the previous step. This will allow the cluster service account to manage resources required during the the Snorkel AI Data Development Platform deploy.
   - ```
     gcloud projects add-iam-policy-binding snorkel-ai-gcp-standardization --member=serviceAccount:gke-cluster-admin-account@snorkel-ai-gcp-standardization.iam.gserviceaccount.com --role=roles/container.admin
     ```
   - ```
     gcloud projects add-iam-policy-binding snorkel-ai-gcp-standardization --member=serviceAccount:gke-cluster-admin-account@snorkel-ai-gcp-standardization.iam.gserviceaccount.com --role=roles/compute.instanceAdmin.v1
     ```
   - ```
     gcloud projects add-iam-policy-binding snorkel-ai-gcp-standardization --member=serviceAccount:gke-cluster-admin-account@snorkel-ai-gcp-standardization.iam.gserviceaccount.com --role=roles/dns.admin
     ```
   - ```
     gcloud projects add-iam-policy-binding snorkel-ai-gcp-standardization --member=serviceAccount:gke-cluster-admin-account@snorkel-ai-gcp-standardization.iam.gserviceaccount.com --role=roles/file.editor
     ```
4. Create the cluster using the gcloud CLI

- ```
  gcloud container clusters create snorkel-flow-gcp \
  --zone=us-central1 \
  --num-nodes=4 \
  --network=[NETWORK] \ # VPC ID if needed
  --subnetwork=[SUBNETWORK] \ # Subnet ID if needed
  --addons=GcpFilestoreCsiDriver,HttpLoadBalancing \
  --enable-ip-alias \
  --service-account=gke-cluster-admin-account@snorkel-ai-gcp-
  standardization.iam.gserviceaccount.com \ # From step B
  --machine-type="e2-standard-32" \
  --scopes=https://www.googleapis.com/auth/cloud-platform \
  --enable-autoupgrade \
  --enable-autorepair \
  --project=snorkel-ai-gcp-standardization
  ```

5. Once the cluster is up and running, install the helm chart for ExternalDNS
   - ```
     helm repo add external-dns https://kubernetes-sigs.github.io/external-
     dns/
     ```
   - ```
     helm repo update
     ```
   - ```
     helm install external-dns external-dns/external-dns \
     --namespace kube-system \
     --set provider=google \
     --set google.project=snorkel-ai-gcp-standardization \
     --set "domainFilters[0]"={DOMAIN YOU OWN} \
     --set "sources[0]"=ingress \
     --set "sources[1]"=service
     --version
     ```
6. Set up kubectl access to the newly created cluster: (https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl)
   - ```
     gcloud components install kubectl
     ```
   - ```
     gcloud container clusters get-credentials CLUSTER_NAME --
     region=COMPUTE_REGION
     ```

Deploying with an Admin Console

Prerequisites

Before installing the Snorkel AI Data Development Platform, make sure the following steps have been completed

- You have access to a Kubernetes cluster with specs outlined in the Snorkel Kubernetes Installation Overview.
- You have privileges to schedule pods, deployments, services, ingresses, and secrets, as well as to create namespaces, within the Kubernetes (K8s) cluster. When we refer to "the K8s cluster" in the remainder of this document, we will be referring to this cluster.
- You have access to a workspace with your K8s environment. This guide assumes an Ubuntu virtual machine is available.
- You have been added to the Vendor Portal under the Stable release by your Snorkel AI contact and you have been provided with a customer license for the installation. There will be another license needed for the Snorkel AI Data Development Platform later in the process.

These instructions assume that you will install the Snorkel AI Data Development Platform using Amazon's Elastic Kubernetes Service. Snorkel can run on any Kubernetes installation that follows the specs in Snorkel Kubernetes Installation Overview, but for this tutorial, we will focus on EKS. If you have specific questions about how Snorkel is configured, please contact Snorkel support. For other general management questions regarding Kubernetes, or EKS, please refer to their documentation.

System Dependencies

From your workspace, install the command line tools required to set up Kubernetes resources:

- Install kubectl
- Install helm

Install into an Existing Cluster

All resources created by the Snorkel utilities will live in a single namespace within the EKS cluster.

```
$ curl https://kots.io/install | bash
$ export PATH=$HOME/bin:$PATH
$ kubectl kots install snorkel-flow
```

The above commands downloads the kots plugin and sets it up for use. You will be asked to provide a directory to save the kubectl-kots file. In the above example, the directory is $HOME/bin.

Follow the steps below to complete the installation:

1. Provide a namespace to install to (default is snorkel-flow).
2. Enter a password for the Admin Console and save it for future use.
3. At this point, you would need to hit Control-C to get your prompt back and run `kubectl port-forward -n <NAMESPACE> svc/kotsadm --address 0.0.0.0 8805:3000`
4. Visit `http://<IP_ADDRESS>:8805` to access the Admin Console and provide the password.

5. Upload your Replicated YAML license file.
6. Configure the platform for your use by filling in the configuration items with assistance from your Snorkel technical representative.
7. Save the configuration and click the "Deploy" button
8. Check the k8s environment for the pods to come online
9. Access the platform ingress page and provide the the Snorkel AI Data Development Platform license to activate the platform.

At this point the Snorkel AI Data Development Platform should be successfully installed into your cluster!

Deploying with Helm

Prerequisites

Before installing the Snorkel AI Data Development Platform, make sure the following have been done.

- You have sent Snorkel AI a **Docker ID** that you'll use for the initial installation and updates.
- You have access to a Kubernetes cluster with specs outlined in the Snorkel Kubernetes Installation Overview.
- You have privileges to schedule pods, deployments, services, ingresses, and secrets, as well as to create namespaces, within a specific AWS EKS cluster. When referring to "the EKS cluster" in the remainder of this document, we will be referring to this cluster.
- A workspace where you will have access to EKS, as well as the ability to save your the Snorkel AI Data Development Platform configuration file, which will be generated during this setup, to a permanent location. This guide assumes an Ubuntu virtual machine is available.
- You have a set of the Snorkel AI Data Development Platform **helm charts** sent over by Snorkel AI

Note: These instructions assume that you will install the Snorkel AI Data Development Platform using Amazon's Elastic Kubernetes Service. Snorkel can run on any Kubernetes installation that follows the specs in Snorkel Kubernetes Installation Overview, but for this tutorial, we will focus on EKS. If you have specific questions about how the Snorkel AI Data Development Platform is configured, please contact Snorkel support. For other general management questions regarding Kubernetes, or EKS, please refer to their documentation.

System Dependencies

From your workspace, install the command line tools required to set up Kubernetes resources:

- Install kubectl
- Install helm

Ensure Cluster Access

Next, you will need to ensure that your kubectl is configured to access the cluster that the Snorkel AI Data Development Platform will be deployed into. Run the following to ensure the pods returned is what you expect:

```
kubectl get pods ––all–namespaces
```

If the output of this command is what you expect to be running in the cluster already, we can move on!

Creating the Kubernetes Namespace

All resources created by the Snorkel utilities will live in a single namespace within the EKS cluster. Create this namespace, and set your kubectl context to point to this namespace.

```
PROJECT_NAME=<A short, unique, alphanumeric name for this instance of Snorkel,
such as “snorkeldemo” or “snorkelproduction”>
kubectl create namespace $PROJECT_NAME
kubectl config set-context --current --namespace $PROJECT_NAME
```

We set the current namespace to the result - for the remainder of the setup, we will be operating in this namespace unless otherwise noted.

Accessing Docker Hub In Kubernetes

In order for your EKS cluster to download Snorkel images, you must give the cluster a registry credential that it can use as an image pull secret. This can be done by running the following:

```
DOCKER_USERNAME=<username for accessing the Docker registry>
DOCKER_PASSWORD=<password for the associated Docker registry user>
DOCKER_EMAIL=<Email used for Docker hub>
kubectl create secret docker-registry regcred \
  -n $PROJECT_NAME \
  --docker-server=index.docker.io \
  --docker-username=$DOCKER_USERNAME \
  --docker-password=$DOCKER_PASSWORD \
  --docker-email=$DOCKER_EMAIL
```

Install Snorkel with Helm

Finally, we'll use the set of Snorkel helm charts to install the full Snorkel AI Data Development Platform. With helm, installation comes in 2 steps:

- Edit the **values.yaml** file for your specific installation
- Use the helm CLI to deploy Snorkel

Tailor the values.yaml File

The following is a guide for fields within the values.yaml file. Documentation is also provided in the values.yaml file itself.

Name	Type	Default
projectName	string	snorkelflow
version	string	[YOUR_SNORKEL_FLOW_VE
image.imageNames	map	`{}`

Name	Type	Default
pagerduty_key	string	
affinity.binPackAlwaysPinnedPods	boolean	false
affinity.tolerations	list	
affinity.nodeAffinity	map	
autoscaling.worker_autoscaling	string	"0"

autoscaling.cluster_autoscaling	map	`{"pod_disruption_bud`
		`"0", "business_hour_s`
		`11, "business_hour_en`

traffic.basePath	string	
traffic.istio.enabled	boolean	false
traffic.istio.mtls	map	`{"enabled": false}`
traffic.istio.gateway	map	`{"create": true}`
traffic.ingresses.domain	string	"snorkel-ai.com"
traffic.ingresses.ingressClassName	string	null
traffic.ingresses.serviceType	string	"ClusterIP"

Name	Type	Default
traffic.ingresses.cloudProvider	string	
traffic.ingresses.tlsHosts	map	`{"enabled": false}`
traffic.ingresses.annotations	map	`{}`
traffic.ingresses.services.[SERVICE]	map	`{\[SERVICE\]: {"enab "urlPrefix": \ [SERVICE_URL_PREFIX\] "annotations": }}`
traffic.tls	map	`{"key_secret_name": proxy-envoy-tls-key-p "cert_secret_name": " proxy-envoy-tls-cert-`
traffic.allowAllInboundTrafficOnKeyServices	boolean	true
traffic.allowInternetAccess	boolean	true
traffic.networkPolicies.enabled	boolean	false
traffic.networkPolicies.ingresses	map	
gpu.enabled	boolean	false

Name	Type	Default
gpu.gpu_config.tolerations	list	
gpu.gpu_config.node_selectors	map	
gpu.gpu_config.schedulerName	string	
gpu.separate_gpu_pods	boolean	false
prefect.enabled	boolean	true
namespace.enabled	boolean	true
services.env	map	{}
services.labels	map	{}
services.[SERVICE].resources	map	{}
services.[SERVICE].env	map	{}
services.[SERVICE].labels	map	{}
services.[SERVICE].min_replicas	int	0
services.[SERVICE].max_replicas	int	VARIES
services.db.shared_buffers	string	"2GB"
services.jupyterhub.enabled	boolean	true

Name	Type	Default
services.jupyterhub.singleUserNotebook.serviceAccountName	string	"snorkelflow-jupyterhub-u
services.jupyterhub.singleUserNotebook.startTimeout	int	300
services.jupyterhub.singleUserNotebook.gpu	boolean	false
services.jupyterhub.singleUserNotebook.resources	map	`{"cpu_guarantee": 1, 1, "memory_gurantee" "memory_limit", "8G"}`
services.jupyterhub.singleUserNotebook.storage	map	`{"dynamicClass": "nu "dynamic"}`
services.secretGenerator.enabled	boolean	false
volumes.[VOLUME].storageClass	string	VARIES
volumes.[VOLUME].storageRequest	string	VARIES
volumes.[VOLUME].volumeName	string	
volumes.[VOLUME].persistentVolume.enabled	boolean	false
volumes.[VOLUME].persistentVolume.driver	map	`{}`
authorization.adRoles.enabled	boolean	false
authorization.adRoles.oidc	map	`{}`

Name	Type	Default
authorization.adRoles.saml	map	`{}`
authentication.jwt	map	`{"enabled": false}`
authentication.oidc	map	
authentication.role	map	`{"key": null, "value`
pretrained_models.enabled	string	false

Deploy Snorkel with the Helm CLI

To see the output of the templated charts with the values in the values.yaml file, run the following:

```
$ helm template --values [PATH_TO_VALUES_FILE] [PATH_TO_CHART_DIRECTORY]
```

If things look good, install Snorkel by running the following:

```
$ helm install --values [PATH_TO_VALUES_FILE] [PATH_TO_CHART_DIRECTORY]
```

At this point the Snorkel AI Data Development Platform should be successfully installed into your cluster!

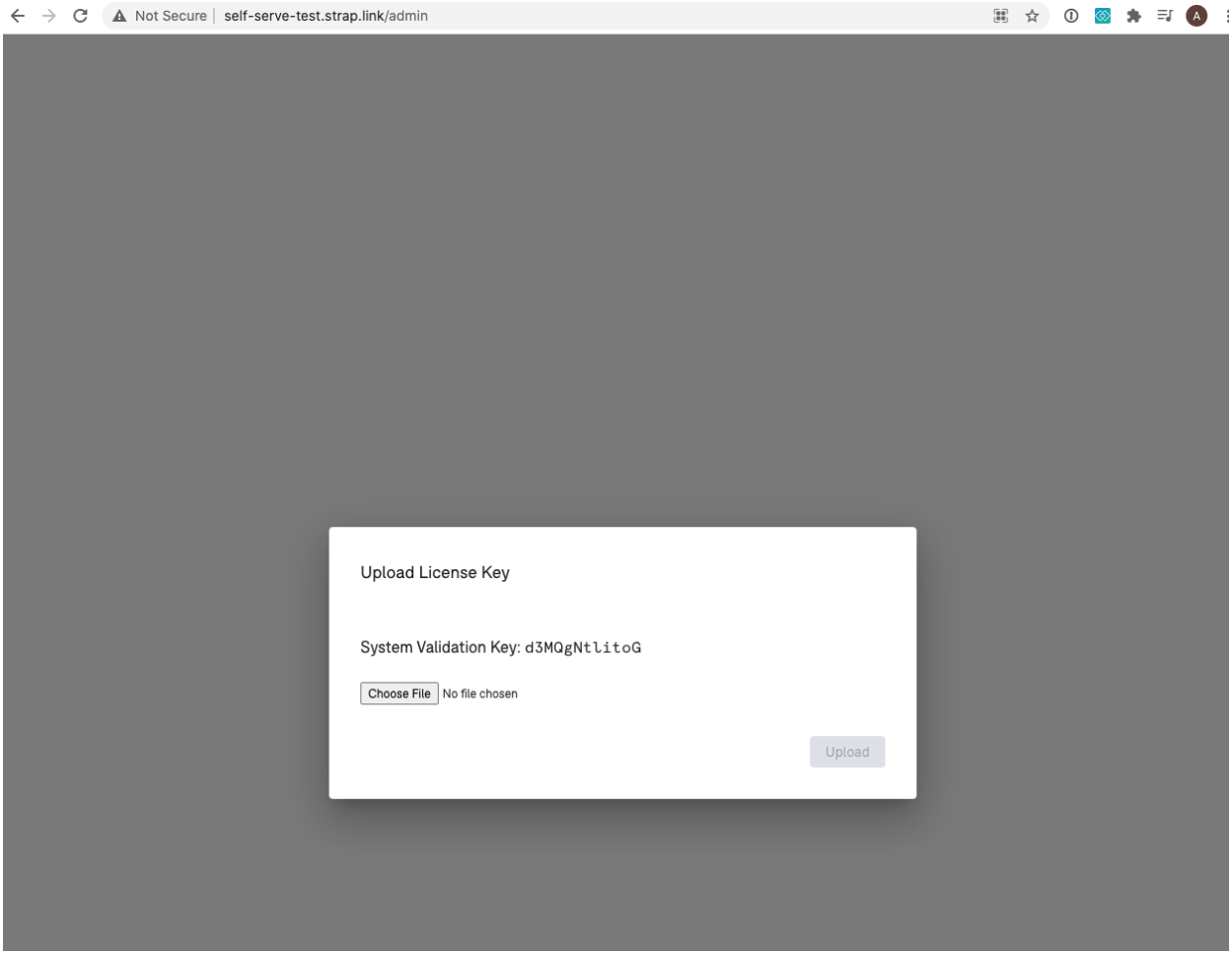
Adding a License Key

Adding a License Key

To make your Snorkel AI Data Development Platform operational, you will need to add a license key furnished by your Snorkel support contact.

To proceed: go to the the Snorkel AI Data Development Platform on your newly created instance at `http://<HOST-IP>`.

You should see a screen requesting that you upload a license key, as well as a System Validation key followed by a string of letters and numbers. It should look like this:



From here, you can select the validation key, copy it, and send it to your Snorkel support contact. They will respond with a file you can upload using the Upload button. Once that is done, your Snorkel AI Data Development Platform will be ready.

Next steps

Now that your Snorkel AI Data Development Platform is active, you can continue configuring it.

You may want to:

- Generate an API key for SDK access

- Add user roles

External S3 bucket storage

Snorkel provides the ability to use Amazon S3 as external storage for datasets and related objects (datasources, uploaded files, etc.). When configured, Snorkel performs in-platform authorization checks to verify user access to files stored in S3.

Prerequisites

Before configuring external S3 storage, ensure you have:

- A Snorkel instance that is either Snorkel-hosted or running on Amazon EKS
- An S3 bucket with appropriate permissions
- AWS credentials and permissions to create IAM roles

Configuration

On-premises instances

For on-premises deployments, configure external storage in your Snorkel configuration:

```
external_storage:
  enabled: false
  # bucket: "s3://my-company-snorkel-storage-bucket"
  # roleArn: "arn:aws:iam::123456789012:role/SnorkelStorageRole"
  # region: "us-west-2"
```

Note: For on-premises installations, follow the AWS EKS IAM roles for service accounts documentation to set up proper IAM configuration.

Contact Snorkel support for detailed installation guidance specific to your environment.

Snorkel-hosted instances

For Snorkel-hosted instances, follow these steps to configure cross-account S3 access:

Step 1: Obtain OIDC issuer URL

Contact Snorkel support to receive the OIDC issuer URL from your Snorkel cluster.

Step 2: Create IAM OIDC provider.

Follow the AWS cross-account access documentation to:

1. Create an IAM OIDC provider for your cluster.
2. Assign IAM roles to Kubernetes service accounts using the issuer URL obtained from Snorkel in the previous step.

Step 3: Configure IAM role permissions

The created IAM role must have the following S3 permissions:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:CreateBucket",
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListBucket",
    "s3:DeleteObject"
  ],
  "Resource": [
    "arn:aws:s3:::your-bucket-name",
    "arn:aws:s3:::your-bucket-name/*"
  ]
}
```

Important: Replace `your-bucket-name` with your actual S3 bucket name.

Step 4: Provide role ARN to Snorkel

After creating the IAM role, provide the role ARN to Snorkel support to complete the configuration.

Final configuration

Once configured with the S3 bucket, role ARN, and AWS region, Snorkel can store and manage datasets in your external S3 bucket while maintaining proper access controls and authorization.

Connect external models

Snorkel supports connecting third-party foundation models, like OpenAI and Claude, to run inference-related workflows like evaluation, on the Snorkel AI Data Development Platform.

To connect to external models, follow these steps:

1. Create an account with the third-party service provider.
2. Configure a connection to that service within your Snorkel instance.

This connection setup allows all workflows in your instance to access the third-party models for inference operations.

Prerequisites

- An account and credentials with the providers you want to connect

How to connect external services

You can connect external integrations to your Snorkel instance. The Snorkel secrets service securely manages stored API keys and secrets.

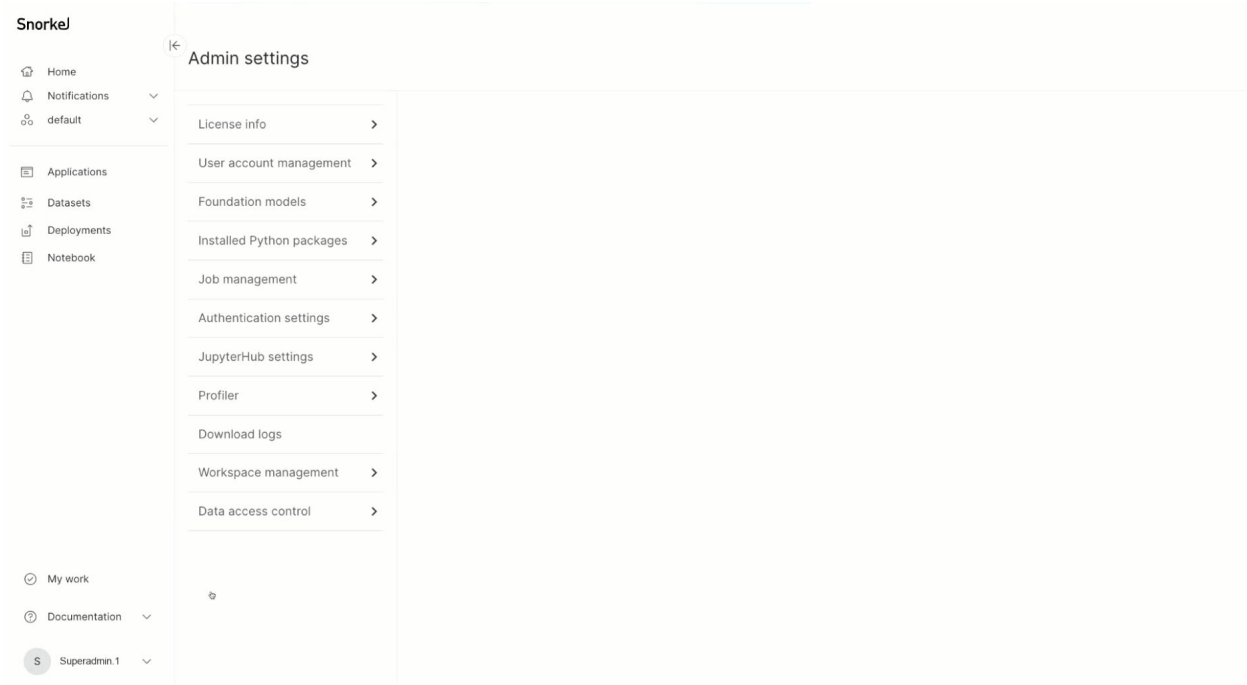
Connect to model providers using the Snorkel GUI

Prerequisites:

- Log in to Snorkel with superadmin permissions

1. In the Snorkel GUI, select your profile.
2. Select **Admin Settings** > **Foundation Models.**
3. Select **Connect** for the integration you want to add.
4. Enter the required details from your external integration provider and select **Save.**
5. Configure available models for the provider you added.
6. (Optional) To make changes, select the **Edit** or **Delete** button for your integration.

> ⚠ WARNING
>
> Deleting an integration also deletes all models associated with that integration.



Connect to model providers using the SDK

1. To list the integrations that are currently connected, use this SDK command:

   ```
   sai.list_integrations()
   ```

2. To check the operational status of an integration, use the `get_model_provider_status` command for your associated provider:

   ```
   sai.get_model_provider_status(ExternalLLMProvider.OPENAI)
   ```

3. To add a connection to a new integration, use the `set_secret` command for your supported services:

   - Hugging Face

     ```
     sai.set_secret("huggingface::inference::api_token", "**YOUR API KEY**")
     ```

     For more, see Hugging Face token settings.

   - OpenAI

     ```
     sai.set_secret("openai_api_key", "**YOUR API KEY**")
     ```

     For more, see OpenAI API keys.

   - Azure OpenAI

     ```
     sai.set_secret("azure_openai_api_key", "**YOUR API KEY**")
     ```

- Azure Machine Learning

```
sai.set_secret("azure::ml::api_key", "**YOUR API KEY**")
```

- Vertex AI

```
sai.set_secret("vertexai_lm_location", "**YOUR PROJECT LOCATION**")
sai.set_secret("vertexai_lm_project_id", "**YOUR PROJECT ID**")
sai.set_secret("vertexai_lm_credentials_json", "**YOUR CREDENTIALS
JSON**")
```

- Amazon SageMaker

```
sai.set_secret("aws::finetuning::region", "**YOUR REGION**")
sai.set_secret("aws::finetuning::access_key_id", "**YOUR ACCESS KEY
ID**")
sai.set_secret("aws::finetuning::secret_access_key", "**YOUR SECRET
ACCESS KEY**")
sai.set_secret("aws::finetuning::sagemaker_execution_role", "**YOUR
EXECUTION ROLE**")
```

- Amazon Bedrock

```
sai.set_secret("aws::bedrock::region", "**YOUR REGION**")
sai.set_secret("aws::bedrock::access_key_id", "**YOUR ACCESS KEY ID**")
sai.set_secret("aws::bedrock::secret_access_key", "**YOUR SECRET ACCESS
KEY**")
sai.set_secret("aws::bedrock::bedrock_execution_role", "**YOUR EXECUTION
ROLE**")
```

- Custom inference service: see Custom Inference Service API

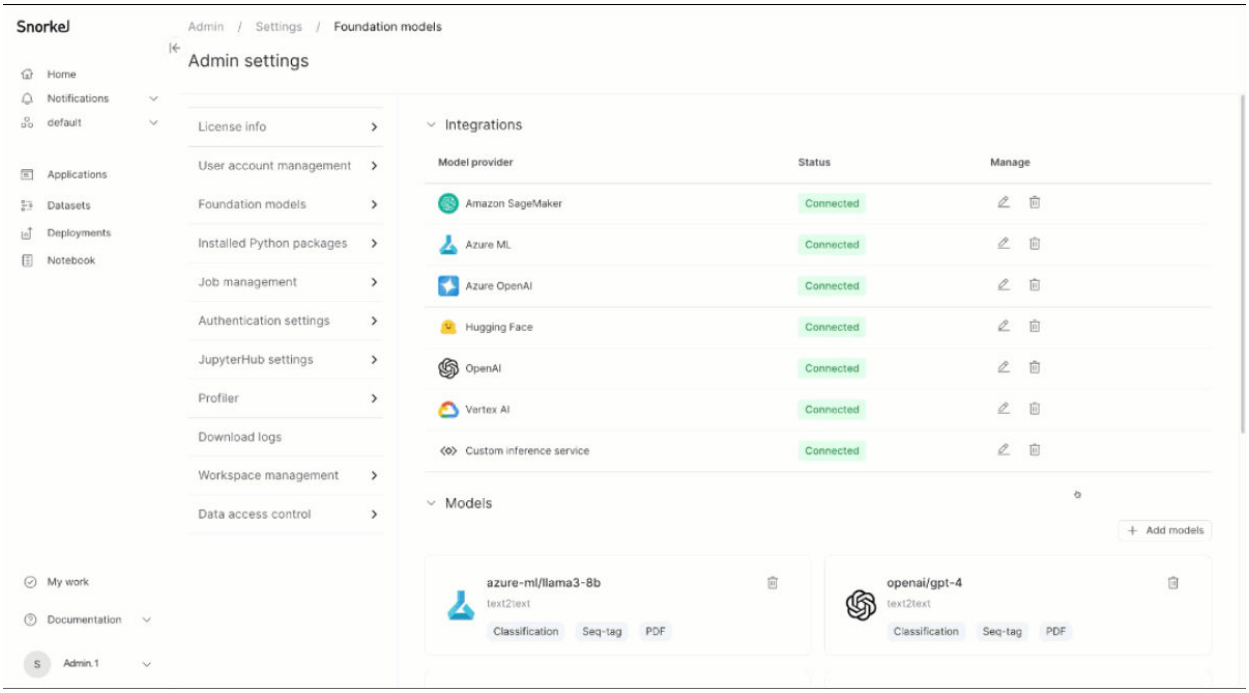
How to add models in Snorkel

After you connect an external service to Snorkel, you can add the models from that service to Snorkel.

Add models using the Snorkel GUI

1. In the Snorkel GUI, select your profile.
2. Select **Admin Settings** > **Foundation Models.**
3. Select the **Add models** button.
4. From the dropdown, choose the model provider you set up in the previous step. Enter the details for **Model type**, **Model name**, and **Endpoint URL**.

The model you added is now visible in the **Models** section and is available for use in Snorkel.



Add models using the SDK

To view the models that are currently connected, use this SDK command:

```
sai.get_external_model_endpoints()
```

You can view detailed information about configured models using the `detail` parameter:

```
sai.get_external_model_endpoints(detail=True)
```

You can inspect configuration information for a particular model using the `model_name` parameter:

```
sai.get_external_model_endpoints(model_name="openai/gpt-4o", detail=True)
```

To add models to Snorkel, include the following information for each model:

- `model_name`: The name of the provider followed by the name of the model. For example, `openai/gpt-4`.
- `endpoint`: The endpoint to perform inference requests.
- `model_provider`: The provider serving the model. For example, Hugging Face, OpenAI, Vertex AI, or a custom inference service.
- `fm_type`: The associated task type for the model.

  This example shows how to add the `openai/gpt-4o` model to Snorkel:

```
from snorkelai.client.tdm.models import ExternalLLMProvider, FMType

sai.set_external_model_endpoint(
    model_name="openai/gpt-4o",  # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions",  # Chat
completions endpoint
    model_provider=ExternalLLMProvider.OPENAI,  # Model provider
    fm_type=FMType.TEXT2TEXT,  # The task type of the model
)
```

To delete a configured model, use this SDK command:

```
sai.delete_external_model_endpoint("openai/gpt-4o")
```

Hugging Face 🤗 inference endpoints

To add a Hugging Face model, launch your chosen Hugging Face model on their servers using Hugging Face endpoints. You will be given an endpoint URL for that model.

This example shows how to set up a Hugging Face Text2Text model for Prompt Dev applications:

```
sai.set_external_model_endpoint(
    model_name="**MODEL NAME**",  # Set this to the model you have created the
endpoint for
    endpoint="**ENDPOINT URL**",  # Set this to the URL for the model found in
the Inference Endpoints Dashboard
    model_provider=ExternalLLMProvider.HUGGINGFACE, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model; in this case,
text2text
)
```

OpenAI API

To add an Open AI model, set up your API token and ensure that you are using the appropriate API endpoint. Specify a chat completions endpoint for chat models or a completions endpoint for language models. For more information about what API endpoint a model belongs, see OpenAI's Text generation models documentation.

This example shows how to configure an OpenAI model with the chat completions API endpoint:

```
sai.set_external_model_endpoint(
    model_name="openai/o1-mini", # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions", # Chat completions
endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

This example shows how to configure an OpenAI model with the legacy completions API endpoint:

```
sai.set_external_model_endpoint(
    model_name="openai/gpt-3.5-turbo-instruct", # Compatible completions model
    endpoint="https://api.openai.com/v1/completions", # Completions endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Azure OpenAI API

To add an Azure OpenAI model, set up your API token and ensure that you are using the appropriate API endpoint.

This example shows how to configure a supported Azure OpenAI chat model.

```
sai.set_external_model_endpoint(
    model_name="azure_openai/your-deployment-name", # Compatible chat
completions model
    endpoint="https://your-instance-name.openai.azure.com/chat/completions", #
Chat completions endpoint
    model_provider=ExternalLLMProvider.AZURE_OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Azure Machine Learning API

To add an Azure Machine Learning AI model, set up your API token and ensure that you are using the appropriate API endpoint.

This example shows how to configure a supported Azure Machine Learning model.

```
from snorkelai.models.prompts.prompts_services.azure import
AzureDataInferenceInterfaceTypes

sai.set_external_model_endpoint(
    model_name="your-model-name", # Name of your deployed Azure ML model
    endpoint="https://<deployment-name>.westus2.inference.ml.azure.com/score",
# Chat completions endpoint
    model_provider=ExternalLLMProvider.AZURE_ML, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
    azure_task_type=AzureDataInferenceInterfaceTypes.Llama.value,
)
```

Bedrock Claude API

To add a Bedrock Claude model endpoint:

```
sai.set_external_model_endpoint(
    model_name="bedrock/anthropic.claude-3-5-sonnet-20241022-v2:0",  # Model
name
    endpoint="bedrock-runtime.us-west-2.amazonaws.com",  # Endpoint
    model_provider=ExternalLLMProvider.Bedrock,  # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Supported Claude models

The following models are currently supported via AWS Bedrock:

- Claude 3.5 Sonnet v2
- Claude 3.7

View the Model IDs here.

Vertex AI language models API

To add a Vertex AI model, set up your Vertex AI location, project ID, and credentials JSON.

This example shows how to configure a Vertex AI model:

```
sai.set_external_model_endpoint(
    model_name="vertexai_lm/gemini-1.5-pro-002",  # Model name
    endpoint="https://cloud.google.com/vertex-ai",  # Endpoint
    model_provider=ExternalLLMProvider.VERTEXAI_LM,  # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Amazon SageMaker API

To add an Amazon Sagemaker model endpoint for predictive use cases:

```
sai.set_external_model_endpoint(
    model_name="sagemaker/jumpstart-dft-meta-textgeneration-llama-3-8b-
instruct", # Model name
    endpoint="https://runtime.sagemaker.<region-
name>.amazonaws.com/endpoints/jumpstart-dft-meta-textgeneration-llama-3-8b-
instruct/invocations", # Endpoint
    model_provider=ExternalLLMProvider.SAGEMAKER, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Custom inference service API

The custom inference service enables users to configure custom endpoints and additional foundation model providers.

OpenAI-compatible services

The custom inference service supports endpoints from foundation model providers that follow the OpenAI API schema, such as Together AI, Groq, and others. To integrate a supported model: OpenAI API specification.

1. Set API secrets

```
sai.set_secret("custom_inference_api_key", "**YOUR API KEY**")

# Optional: Set default HTTP headers. Note: if authentication is is not via
sai.set_secret("custom_inference_optional_headers", {
  "**HEADER_NAME**": "**HEADER_VALUE**"
})
```

NOTE: By default, `custom_inference_api_key` is used as a Bearer token in the `Authorization` header for requests (i.e. `Authorization: Bearer <custom_inference_api_key>`). If your service uses a different or additional authentication method, set the correct headers using `custom_inference_optional_headers`. However, you still need to provide an arbitrary value for `custom_inference_api_key`.

2. Register model endpoints

After setting up your custom inference API key, add a model with custom inference service that conforms to the OpenAI API specification. Like the OpenAI setup, specify a chat completions endpoint for chat models or a completions endpoint for language models. This inference service is also extensible to other foundation model providers that conform to OpenAI's API specification, such as Together AI.

This example shows how to set up a supported model with the chat completions API endpoint:

```
sai.set_external_model_endpoint(
    model_name="meta-llama/Llama-3.2-3B-Instruct-Turbo", # Chat model
    endpoint="https://api.together.xyz/v1/chat/completions", # Inference
service chat endpoint
    model_provider=ExternalLLMProvider.CUSTOM_INFERENCE_SERVICE, # Model
provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

This example shows how to set up a supported model with the completions API endpoint:

```
sai.set_external_model_endpoint(
    model_name="meta-llama/Meta-Llama-3-70B", # Language model
    endpoint="https://api.together.xyz/v1/completions", # Inference service
completions endpoint
    model_provider=ExternalLLMProvider.CUSTOM_INFERENCE_SERVICE, # Model
provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Non-OpenAI-compatible services

If your external inference service does not conform to the OpenAI API spec, you can still integrate it using a **transform configuration**. This allows Snorkel to map requests and responses between OpenAI-compatible schemas and your custom API.

Steps:

1. Set the `custom_inference_api_key` and `custom_inference_optional_headers` secrets as above.
2. Create a transform configuration: The transform configuration is a JSON object with the following required fields:

- **`/chat/completions`**:
  The root key representing the OpenAI endpoint you wish to transform. This key defines which OpenAI endpoint's requests/responses will be mapped.

- **`custom_path`**:
  The path on your custom LLM API that corresponds to the OpenAI `/chat/completions` endpoint. Snorkel routes requests to this path instead of the default OpenAI endpoint.

- **`openai_to_custom_request`**:
  An object that defines how to transform the OpenAI request JSON into the format expected by your custom LLM API.
  - **Keys**: Field names as expected by your custom API. - **Values**: JMESPath expressions describing how to extract or map values from the OpenAI request schema.

- **`custom_to_openai_response`**:
  An object that defines how to transform your custom LLM API's response into the OpenAI response schema.
  - **Keys**: Field names as expected by the OpenAI API/SDK.
  - **Values**: JMESPath expressions describing how to extract or map values from your custom API's response.

  When configuring a Custom Inference Services with a transform spec and associated models, set the **base URL** for the custom inference service `endpoint` (**do not** include the chat completions path). Instead add the chat completion path to the `custom_path` field in the transform configuration.

  Example transform configuration:

```
import json

transform_config = {
  "/chat/completions": {
    "custom_path": "/v1/generate",
    "openai_to_custom_request": {
      "useCase": "'text-generation'",
      "contextId": "model",
      "preSeed_injection_map": {
        "system": "messages[?role=='system'].content | join(' ', @)",
        "user": "messages[?role=='user'].content | join(' ', @)"
      },
      "parameters": {
        "temperature": "temperature",
        "maxOutputTokens": "max_tokens",
        "topP": "top_p",
        "responseMimeType": "'application/json'"
      }
    },
    "custom_to_openai_response": {
      "id": "responseMetadata.vegasTransactionId",
      "model": "responseMetadata.llmResponsePayload[0].modelVersion",
      "choices": "[{index: to_number('0'), message: {role: 'assistant',
content:
responseMetadata.llmResponsePayload[0].candidates[0].content.parts[0].text},
finish_reason: responseMetadata.llmResponsePayload[0].finishReason == 'STOP'
&& 'stop' || 'length'}]",
      "usage": {
        "prompt_tokens":
"responseMetadata.llmResponsePayload[0].usageMetadata.promptTokenCount ||
to_number('0')",
        "completion_tokens":
"responseMetadata.llmResponsePayload[0].usageMetadata.candidatesTokenCount
|| to_number('0')",
        "total_tokens":
"responseMetadata.llmResponsePayload[0].usageMetadata.totalTokenCount ||
to_number('0')"
      }
    }
  }
}


sai.set_secret("custom_inference_transform_spec",
json.dumps(transform_config))
```

How to specify model hyper-parameters

Snorkel supports the setting of arbitrary model hyper-parameters when you configure a model in Snorkel. Some example parameters that this include are:

- `temperature`
- `top_p`
- `max_input_length`
- `max_output_length`

When setting a model hyper-parameter, please ensure it appears exactly as documented by the model's provider. For example, when configuring the temperature hyper-parameter, some models leverage the keyword `temp`, others use `t`, and some require the full word `temperature`.

Once you have confirmed the hyper-parameter names and values you'd like to set, provide them as keyword arguments to `sai.set_external_model_endpoint`. For example, to set `temperature` and `max_tokens` on `openai/gpt-4o`, run:

```
sai.set_external_model_endpoint(
    model_name="openai/gpt-4o", # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions", # Chat completions
endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
    temperature=0.5,
    max_tokens=500,
)
```

How to set rate limits

Snorkel supports custom rate limits for the OpenAI, Azure OpenAI, Bedrock, and custom inference service integrations.

To enable this feature, use the SDK function `sai.set_external_model_endpoint()` to configure the `requests_per_sec` and `tokens_per_sec` parameters. Snorkel consumes as much as the provided quota when computing previews of prompt LFs for the whole Snorkel instance.

When setting these values, get the correct values from your organization's usage tier and provide the values in per-second level. If you provide incorrect values, Snorkel might hit rate limits for OpenAI or underutilize the provided quota.

To determine OpenAI usage limits, visit the OpenAI organization limits page. Find the **token limits** and **request limits** per minute for your desired model and divide it by 60 to compute the per-second value. For example, when the model `gpt-4o-mini` shows 10,000 RPM for request and other limits and 30,000,000 TPM for token limits, provide `requests_per_sec=166` and `tokens_per_sec=500000` as extra keyword arguments in the `set_external_model_endpoint()` function.

Here is an example for configuring `gpt-4o-mini` with rate limits:

```
# delete an existing model endpoint, only if already registered
sai.delete_external_model_endpoint("openai/gpt-4o-mini")
sai.set_external_model_endpoint(
    model_name="openai/gpt-4o-mini", # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions", # Chat completions
endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
    requests_per_sec=166, # request limit per second
    tokens_per_sec=500000, # token limit per second
)
```

Overview

This article lists the different authentication methods supported by the Snorkel AI Data Development Platform and introduces the **User Role** feature for different platform user personas. It can point you to an authentication setup suitable for your organization, and help map the correct permissions from your roles/groups to Snorkel User Roles.

Platform Authentication

The Snorkel AI Data Development Platform offers two primary methods of authentication that are described below. For details on SSO integration, please visit the links mentioned.

1. Basic authentication
2. **[Recommended]** Single Sign-On (SSO)
   - SAML integration
   - OIDC integration

After initial authentication, the capabilities of each user is defined by the concept of workspaces and the Role-Based Access Controls (RBAC) roles within each workspace.

Basic Authentication

This is authentication users with a username/password pair. This is the **default** authentication method for freshly installed Snorkel AI Data Development Platform and can be disabled once alternative authentication method is set up.

For a freshly installed Snorkel AI Data Development Platform, you will be prompted to create a first user, which will inherit the **superadmin** role. The superadmin role is the most elevated role within Snorkel and is leveraged for administrative tasks within the Snorkel AI Data Development Platform (such as setting up SSO, creating new users), and should not be used for day to day activities.

Single Sign-on (SSO)

This is the recommended authentication method for Snorkel. We support enterprise-grade SSO integration with SAML and OIDC. Please go to relevant section for your integration.

- SAML integration
- OIDC integration

Snorkel User Roles

A user role represent a collection of permissions to perform actions on Snorkel AI Data Development Platform (e.g. creating application, uploading dataset).

Currently we have the following roles (details in user roles definition):

- **Superadmin/System Admin**: Overall manager for the entire instance: control over all workspaces and applications in the instance
- **Admin/Workspace Admin**: Full admin control over all app development for a specific workspace.
- **Developer**: Core persona developing application in Snorkel: uploading data, pre-processing, LF and model development, deployment

- **Reviewer**: Manage all annotators and annotation batches
- **Annotator**: Creator of annotations for batches assigned to them

> ℹ️ Except for superadmin, all user roles are **workspace-scoped**. This means the same user can have different roles in different workspaces. (e.g. **Developer** in workspace 1, but **Reviewer** in workspace 2)

For more detailed permission breakdown and persona, please reference user roles definition

Snorkel User roles

The table below summarizes the Snorkel AI Data Development Platform capabilities available for each role. Each role has a description, a suggested user type, and product capabilities available for that role.

Role and permission overview

> ℹ️ Except for superadmin, all user roles are **workspace-scoped**. This means the same user can have different roles in different workspaces. (e.g. **Developer** in workspace 1, but **Reviewer** in workspace 2)

If you do not know what role you belong to, please ask your System Administrator for assistance. They can look up this information in the **Admin Settings > User Management** tab.

Role	Description	Suggested user	Capabilities
Superadmin/System Admin	Overall manager for the entire instance: control over all workspaces and applications in the instance	IT Manager	Inherits **Admin** Capabilities and access to all Admin Settings across workspaces: Manage Users (Create user, Deactivate user, Create invite links for account creation, Manage user's role in each workspace) Manage Licenses Ability to add and remove Workspaces Manage dataset and application limits for each workspace SSO/Authentication settings Job Management Manage DataConnector permissions
Admin/Workspace Admin	Full admin control over all app development for a specific workspace	Principal Data Scientist, Product Manager	Inherits **Developer** Capabilities and Admin Capabilities for specific workspaces: Access to admin capabilities for specific workspaces Manage Users and their roles for the workspaces they are admins for
Developer	Core persona developing application in Snorkel: uploading data, pre-	Data Scientists	Inherits **Reviewer** Capabilities and:

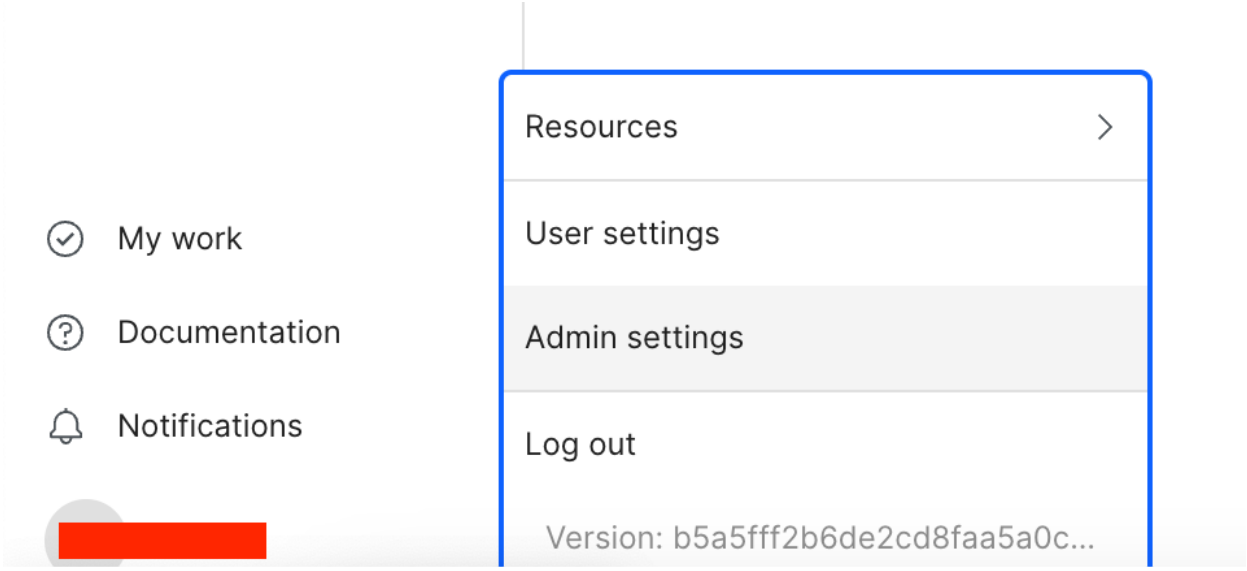
Role	Description	Suggested user	Capabilities
	processing, LF and model development, deployment		Access to the Development Mode (i.e., DAG + Model Studio) Ability to Create Labeling Functions Ability to Run models Ability to Export and Deploy models Access Notebooks under their individual account
Reviewer	Manage all annotators and annotation batches	Subject Matter Expert, Manager	Inherits **Annotator** Capabilities and: Access to Annotation Suite metrics dashboard (e.g., inter-annotator agreement, overall batch progress) Assign batches to Annotators View annotations from all Annotators Add annotations to any batch
Annotator	Creator of annotations for batches assigned to them	Subject Matter Expert	Access to only the Annotation Mode View batches assigned to them and their previous annotations Add annotations to their assigned batches

User permission management

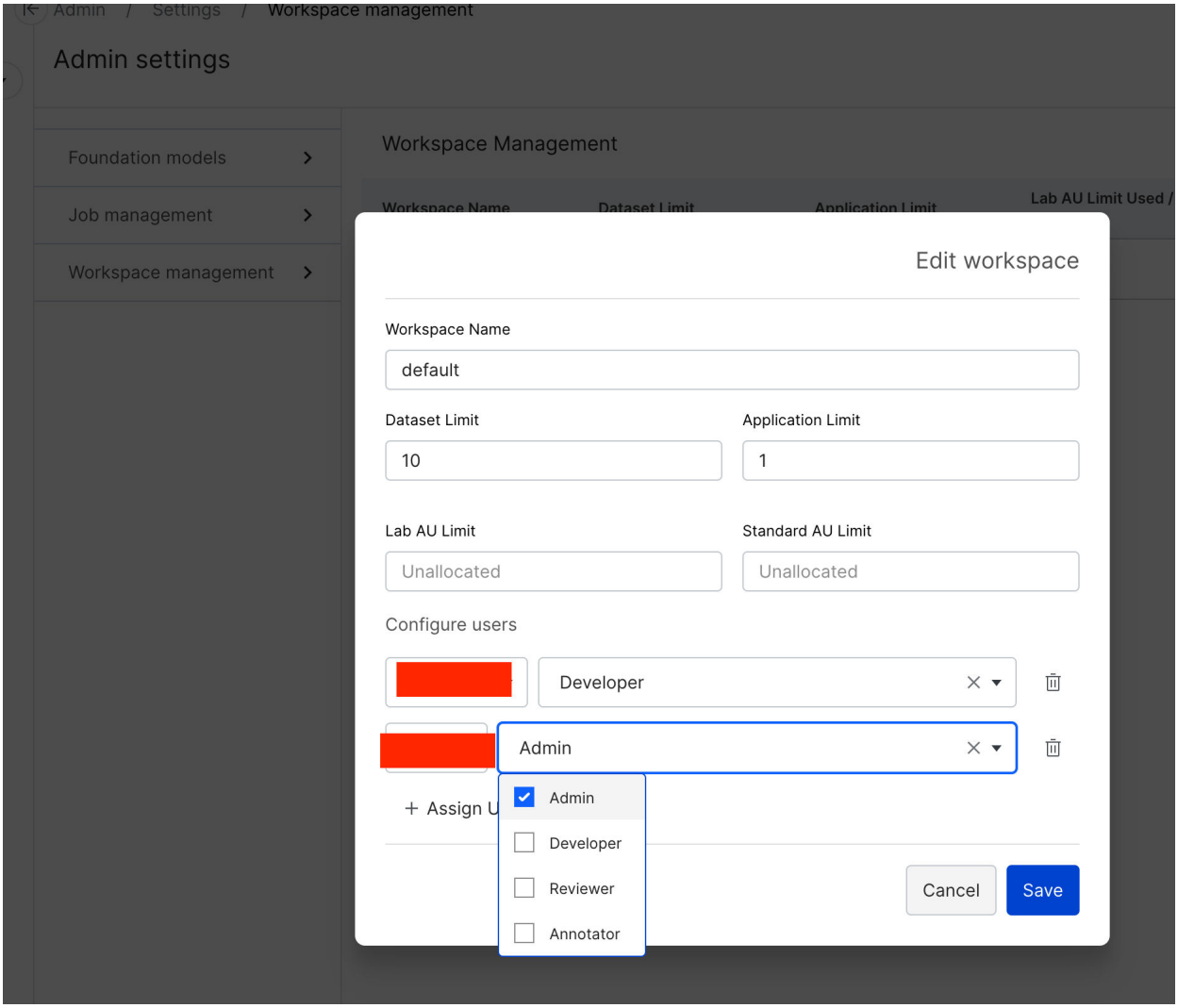
From the **User account management** panel, administrators can create new users, deactivate users, and assign a different role to each user. This allows fine-tuning of access control to users, based on their role and need for access to certain data.

If you are an administrator, you can update users and roles in the **Workspace management** panel:

1. Select your username in the bottom left corner of your screen.
2. Select **Admin settings**.



3. Select the edit icon ( ) for your specific workspace.
4. Add the users and roles in the **Configure users** section.



5. Select **Save** to apply these users and roles to your workspace.

Deep Dive: Permissions for annotation workflows

This section goes deeper into permissions related to annotation workflow, and what the Snorkel user roles mentioned above can perform.

	Annotator	Reviewer	Developer	Admin
See assigned batches	X	X	X	X
Annotate on assigned batches	X	X	X	X
See all batches	X	X	X	X
See all assignees	X	X	X	X
Create new batches			X	X
Assign batches to new assignees		X	X	X
Rename batches		X	X	X
Delete batches		X	X	X
Set an assignee as expert		X	X	X
See other assignees' annotations		X	X	X
Commit annotations to ground truth			X	X
Manage users and jobs				X

Configure SAML SSO

This document will guide you through the process of setting up SAML-based authentication for single sign-on (SSO) in the Snorkel AI Data Development Platform. By the end of the step-by-step guide, you will be able to log in to Snorkel using your identity provider (IDP) mediated by SAML 2.0.

Prerequisites

- An existing Snorkel AI Data Development Platform
- Superadmin access to the Snorkel AI Data Development Platform
- A SAML 2.0-compatible Identity Provider (e.g. Okta, Google, or PingIdentity)

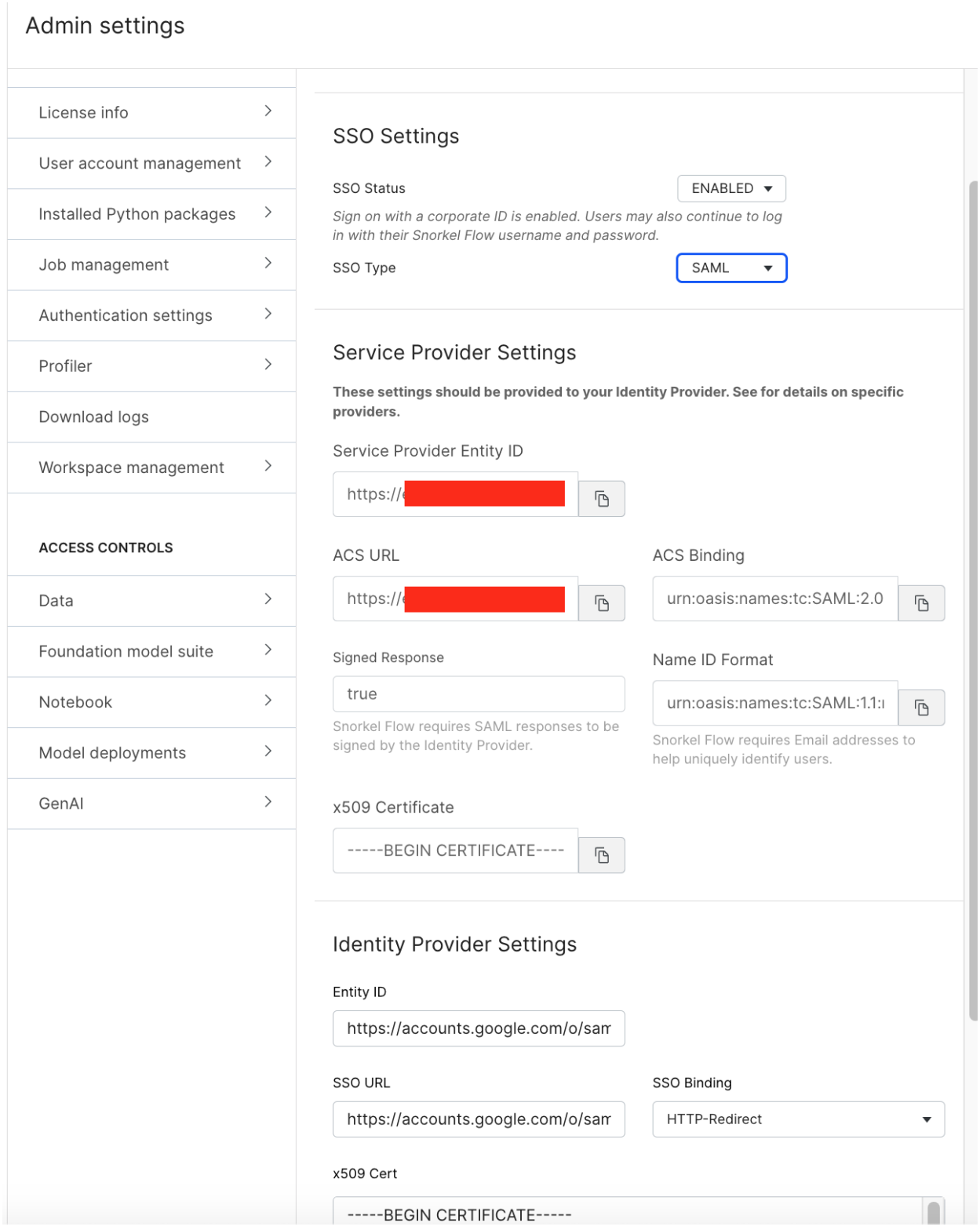
Configuring your identity provider

We recommend having three windows open: one for Snorkel, another for your IDP, and a third with these instructions. You will be asked to copy and paste metadata between the two systems.

Initial configuration

Open the **Admin Settings** page in Snorkel, and navigate to the **Authentication Settings** tab. Change SSO Status to *Enabled* and SSO Type to *SAML*.

> ⚠ WARNING
>
> **DO NOT** set SSO Status to *Required* until you have successfully verified your setup, otherwise you risk being locked out of the instance. Once the SSO Status is set to *Required*, you will no longer be able to log in with username/password.



Registering Snorkel with your identity provider

In order for your IDP to provide authentication for Snorkel, you will need to provide information about your Snorkel AI Data Development Platform to your IDP.

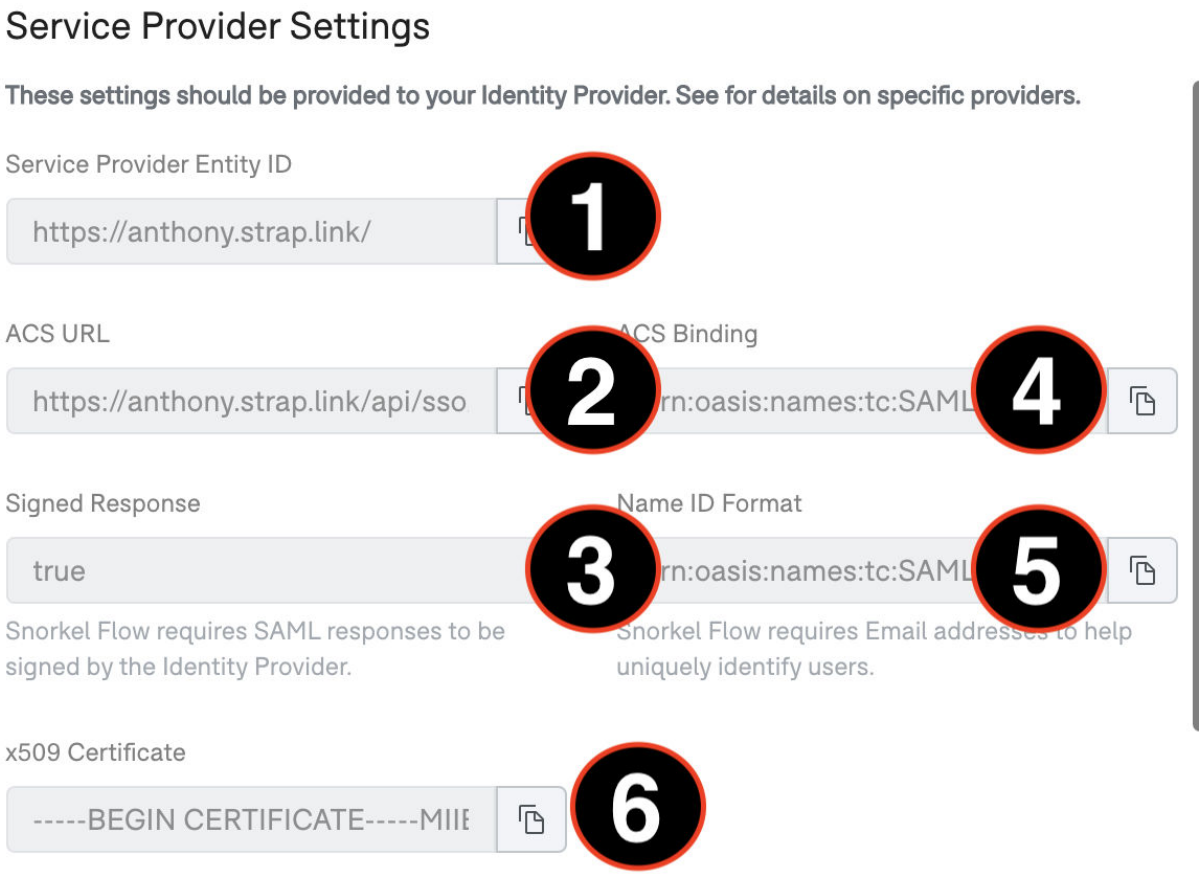
First, in your IDP management console, add a new service provider. We recommend naming the service provider after your Snorkel AI Data Development Platform. Each IDP has their own name for service

providers, such as “Applications” or “Connections.” Note that if you have multiple Snorkel AI Data Development Platforms, you will need to repeat this for each instance.

Your IDP will then ask you to enter standard metadata about the newly created service provider. These can be retrieved from the Service Provider Settings section in the Snorkel AI Data Development Platform.

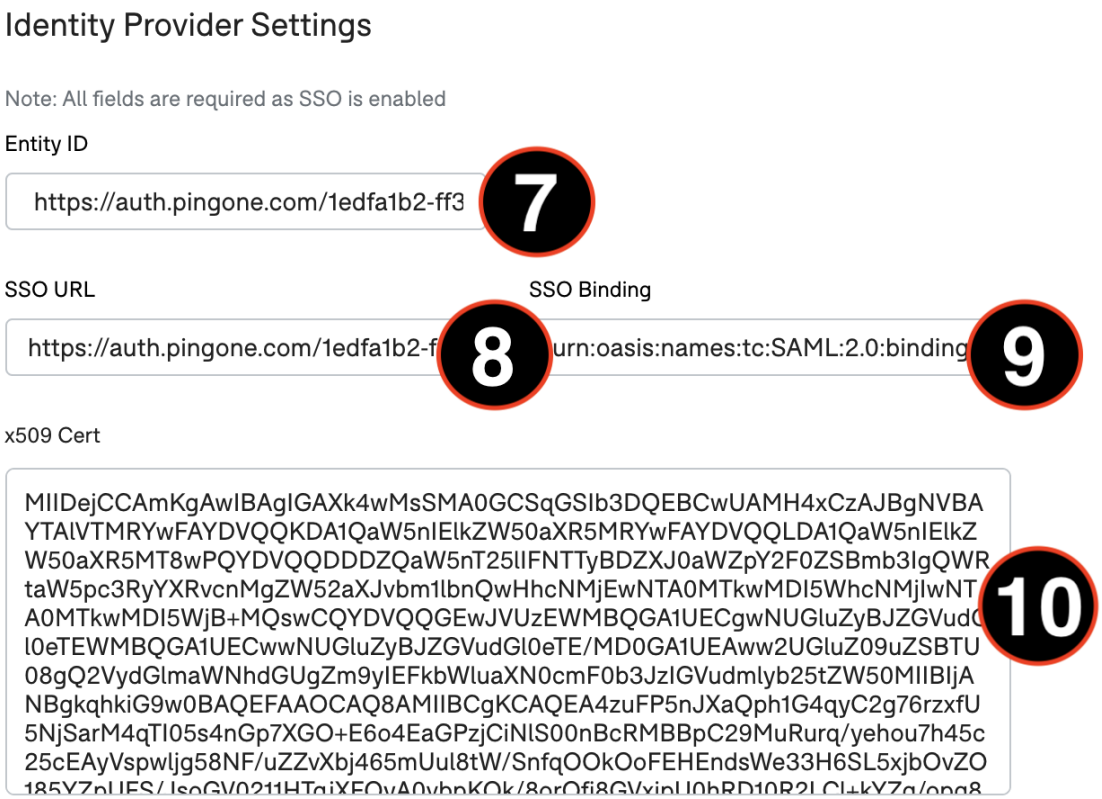
1. Service Provider Entity ID: Unique identifier that names your Snorkel AI Data Development Platform.
2. ACS URL: URL where requests are sent after successful authentication with your IDP.
3. Signed Response: Look for a checkbox that says “Sign Assertion & Response” or “Sign Response” and check it.
4. ACS Binding: In many cases, this value is optional. Paste the value as given in the Snorkel settings or look for an option called Binding and select Redirect.
5. Name ID Format: Snorkel uses each user’s email address for SSO login. Paste the value as given in Snorkel settings or look for an option called Name ID Format and select “Email” or “Email Address”.
6. X509 Certificate: Certificate provided by Snorkel, often referred to as the Verification Certificate by IDPs. Paste the value for this certificate directly into this field in your IDP or upload file containing this content.

> INFO
> In the diagram below, the numbered fields correspond to the descriptions above.



Registering your identity provider with Snorkel

7. **Identity Provider Entity ID**: This may also be called the Issuer ID, and also tends to be a URL. Paste the value from your IDP here.
8. **SSO URL**: This may also be called the SSO Service URL, and frequently will end with a code like `start` or `startsso`. Paste the value from your IDP here.
9. **SSO Binding**: This is commonly the string `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`, but if your IDP specifies a different value, paste it here.
10. **X509 Certificate**: This is frequently a file downloaded from your IDP called the Signing Certificate. You may also see a body of text to copy and paste. Download this file, copy the contents and paste them into this field.



Once you've finished the configuration above, click the Save button.

Testing the connection

You can test the connection to your IDP by using the **Test SSO** button on the **Admin Settings** page. This will attempt to perform an authentication handshake between your IDP and your Snorkel AI Data Development Platform. If the connection is successful you will be redirected back to the Admin Settings page with a success message. If not, you will be presented an error message.

If you run into an error, review your steps through the guide above to ensure everything was configured correctly. If you need additional assistance, contact Snorkel AI support.

Requiring SSO login

As an administrator, you can require that all Snorkel authentication use your SSO provider. This is enabled by changing the SSO Status from **Enabled** to **Required**.

As a prerequisite, all users must have email addresses assigned to their profile in Snorkel. Emails can be provided by Snorkel administrators on the **User Management** tab on the **Admin Settings** page.

> ⚠ WARNING
>
> **DO NOT** set SSO Status to *Required* until you have successfully verified your setup, otherwise you risk being locked out of the instance. Once the SSO Status is set to *Required*, you will no longer be able to log in with username/password.

Provider-specific guides

PingIdentity

Initial configuration

1. Add a Connection in the PingOne console.
2. Select “Web App” as the application type.
3. Select “Manually Enter” under app metadata.

Registering Snorkel with your identity provider

1. Copy ACS URL from Snorkel into the information pane.
2. Download the signing certificate and open the file in a text editor. Copy and paste the content into the “x509 Cert” field under Identity Provider Settings on Snorkel.
3. Select “Sign Assertion and Response”.
4. Leave the default Signing Algorithm in place.
5. Do not enable encryption.
6. Under the field Entity ID, copy the field called “Entity ID” under Service Provider Settings in Snorkel. Paste this value into this field.
7. Leave SLO Endpoint and SLO Response Endpoint blank.
8. Set the field “Assertion Validity Duration (In Seconds)” to 300.
9. Set the field “Target Application URL” to the URL of your Snorkel AI Data Development Platform. This is generally the same as the URL you pasted for the Entity ID.
10. Select “Enforce Signed Auth Request”.
11. Under the “Verification Certificate”, first copy the contents of the x509 Cert under the Service Provider Settings and paste it into a file. Save this file, then choose to import this file as the verification certificate on PingIdentity.
12. Click Save and Continue.

Registering your identity provider with Snorkel

1. From the list of Connections on PingIdentity, select the Snorkel connection you just created.
2. Click the menu button on the right side of the application.

3. Click the “Configuration” tab in the Snorkel application details.
4. Copy the Issuer ID beginning with `https://auth.pingone.com` and paste it as the Entity ID under Identity Provider Settings in Snorkel.
5. Copy the Single Sign-on Service URL beginning with `https://auth.pingone.com` and paste it as the SSO URL under Identity Provider Settings in Snorkel.
6. Enter `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect` as the SSO Binding under Identity Provider Settings in Snorkel.
7. Click Save in Snorkel.

Configure OIDC SSO

This document will guide you through the process of setting up OpenID Connect-based (OIDC) authentication for single sign-on (SSO) in the Snorkel AI Data Development Platform. By the end of the step-by-step guide, you'll be able to log in to the Snorkel AI Data Development Platform using your identity provider mediated by OpenID Connect.

Prerequisites

- An existing Snorkel AI Data Development Platform
- Superadmin access to the Snorkel AI Data Development Platform
- An OpenID Connect identity provider (e.g Okta, Google, or PingIdentity)

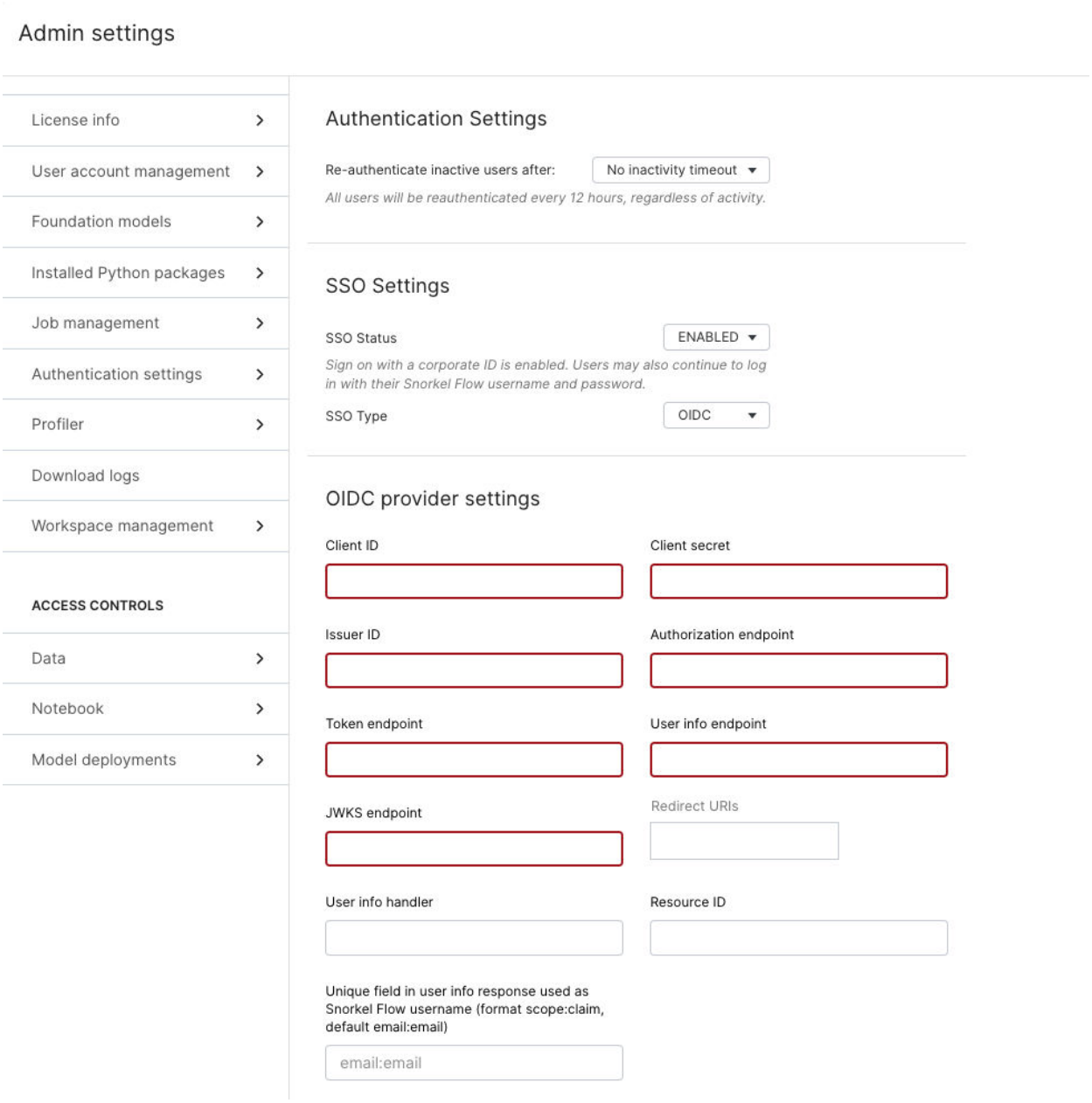
Configuring your identity provider

We recommend having three windows open: one for the Snorkel AI Data Development Platform, another for your identity provider, and a third with these instructions. You'll be asked to copy and paste metadata between the two systems.

Initial configuration

Open the **Admin Settings** page in Snorkel, and navigate to the **Authentication Settings** tab. In SSO Settings section, change SSO Status to *Enabled* and SSO Type to *OIDC*.

> ⚠ WARNING
>
> **DO NOT** set SSO Status to *Required* until you have successfully verified your setup, otherwise you risk being locked out of the instance. Once the SSO Status is set to *Required*, you will no longer be able to log in with username/password.



Continue to the next section to read a description for each of these settings.

Registering your identity provider with Snorkel

Fill out the **OIDC provider settings** according to the descriptions in the table below.

Field	Description
Client ID	The client ID created by your identity provider.
Client secret	The client secret created in pair with the client ID by your identity provider.
Issuer ID	An ID used by the Identity provider, this may be a URL.

Field	Description
Authorization endpoint	A URL provided from your identity provider for Authorization.
Token endpoint	A URL provided from your identity provider for generation of tokens.
User info endpoint	A URL provided from your identity provider for user info.
JWKS endpoint	A URL provided from your identity provider for JWKS info. This may also be referred to as a cert endpoint.
Redirect URIs	The URIs that are allowed for redirection for this OIDC token. These must match what is set in the identity provider settings. The default path used by the Snorkel AI Data Development Platform is `/api/sso/oidc/callback`.
User info handler	Leave field blank
Resource ID	Leave field blank
Unique field in user info response	If your OIDC provider uses a different field or format for user identification, you can customize this field to match the desired claim. The format is `scope:claim`. By default, this is set to `email:email`

Once you've finished the configuration above, select **Save** at the bottom of the page.

Testing the connection

You can test the connection to your identity provider by using the **Test SSO** button on the Admin Settings page. This will attempt to perform an authentication handshake between your identity provider and your Snorkel AI Data Development Platform. If the connection is successful you'll be redirected back to the Admin Settings page with a success message. If not, you'll be presented an error message.

If you run into an error, review your steps through the guide above to ensure everything was configured correctly. You may need additional configuration on your identity provider side. If you need additional assistance, contact Snorkel AI support.

Requiring SSO login

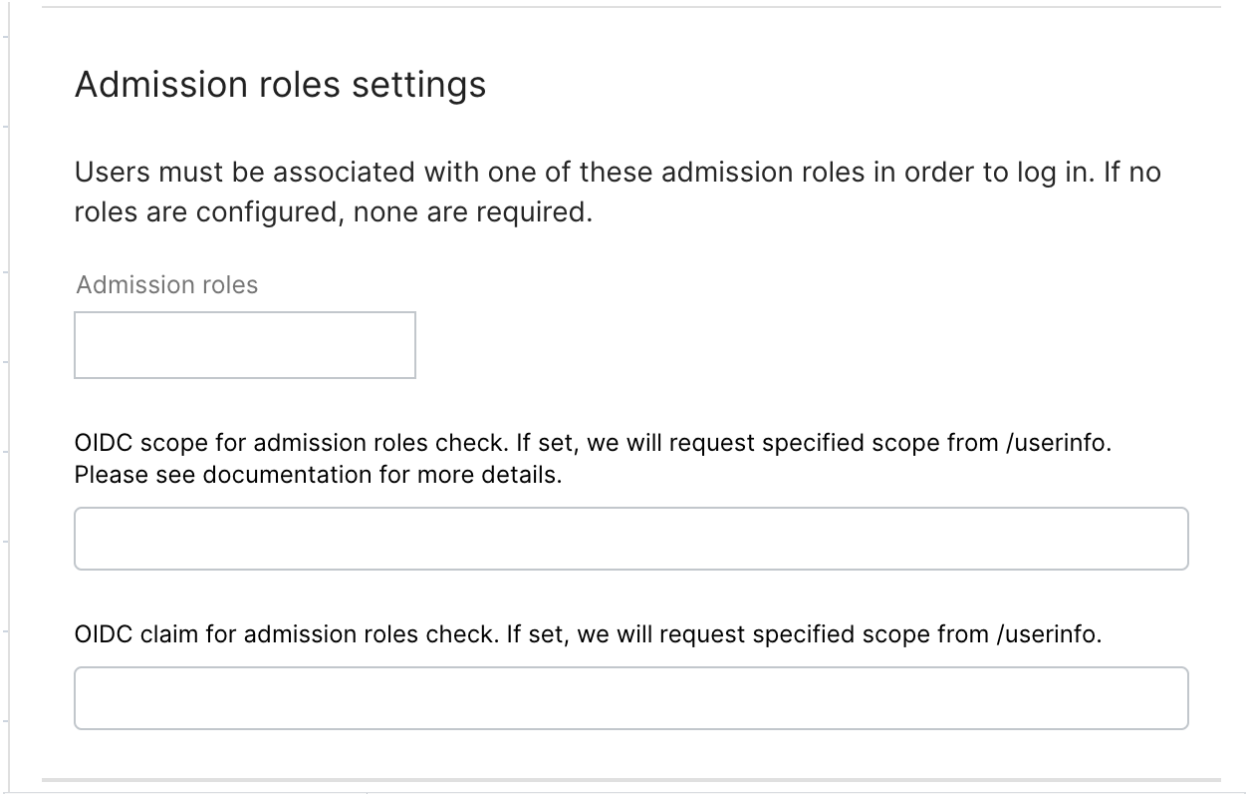
As an administrator, you can require that all Snorkel authentication use your SSO provider. This is enabled by changing the SSO Status from *Enabled* to *Required*.

As a prerequisite, all users must already have email addresses assigned to their profile in the Snorkel AI Data Development Platform. Emails can be provided by Snorkel administrators on the User Account Management tab on the Admin Settings page.

> ⚠ WARNING
>
> **DO NOT** set SSO Status to *Required* until you have successfully verified your setup, otherwise you will risk being locked out of the instance. Once the SSO Status is set to *Required*, you will no longer be able to log in with username/password.

Admission Roles

You can allow only specific users from your OIDC provider to connect to the Snorkel AI Data Development Platform by using the **Admission Roles** feature. This field lets you set a list of the roles that a user must have at least one of to be allowed to login via OIDC.



Field	Description
Admission Roles	(**Required**) The roles allowed to sign in with snorkel. User must be *one of* these to pass admission.
OIDC scope for admission role check	(**Optional**) If set, Snorkel will call `/userinfo` with the specified scope.
OIDC claim for admission role check	(**Optional**) If set, Snorkel will use the content of this claim after calling `userinfo` for admission checks.

Considered the following scenario:

> You want only employees with the `SnorkelUser` role to access the Snorkel AI Data Development Platform using SSO. This role information is stored in `roles` scope under `thirdparty` claim. (In this example, both `roles` and

> `thirdparty` are specific to your company and not part of the spec for OIDC)

Then your settings should be:

Field	Description
Admission Roles	`SnorkelUser`
OIDC scope for admission role check	`roles`
OIDC claim for admission role check	`thirdparty`

After the initial handshanke, the user specified scope (`roles`) will be requested from the `/userinfo` endpoint, and the user specified claim (`thirdparty`) will be checked to get the list of roles from the user info response. If the user does not have the specified role (`SnorkelUser`), they will be denied access for the log in.

However, if your OIDC provider provides the admission role as part of the initial request, no second request to `/userinfo` is needed. In this case, you should omit the scope and claim field above and the platform will check the claim on the `id_token` provided by the OIDC provider.

Active Directory Roles synchronization

AD sync is a process where user information, workspaces, and roles are synchronized between your organization's Active Directory (AD) and your Snorkel AI Data Development Platform. This synchronization ensures that user identities and permissions are consistent and secure across both on-premises Snorkel-hosted environments.

Here are some benefits to using AD sync:

- Security and compliance: AD sync helps enforce consistent access policies and ensures that only authorized users have access to specific resources, reducing the risk of data breaches and ensuring compliance with regulations such as GDPR.
- Centralized identity management: By centralizing identity management, user management processes such as provisioning and deprovisioning accounts are streamlined. In addition, it reduces the administrative overhead associated with managing multiple user directories.
- Improved user experience: By synchronizing user identities, users will only need a single set of credentials.
- Efficient resource allocation: You can dynamically assign and revoke access to resources based on users' roles and responsibilities within your organization. This ensures that users will have the appropriate level of access to perform their job functions without granting unnecessary privileges.

If you want to implement AD roles sync on your instance, reach out to your Snorkel representative.

Considerations for AD roles

When users are provisioned to the default workspace without any AD role, they receive the minimum Annotator permissions. Each user can have only one role within the same workspace, but can have different roles for each workspace.

If the superadmin sets up an admission role, the admission role is required for every user to gain access to the workspace, even if a user has other AD roles.

When the AD roles for a user changes, the new role overwrites previous roles for that user.

The Snorkel AI Data Development Platform completely ignores any roles that are not formatted correctly. This includes, but is not limited to these incorrect formats:

- Nonexistent roles and workspaces
- Prefixes that do not match
- Roles that do not meet these formats: `<PREFIX><SEPARATOR><WORKSPACE><SEPARATOR><SF-ROLE>` or `<WORKSPACE><SEPARATOR><SF-ROLE>`

Setting up Active Directory Roles sync with SAML SSO

Active Directory (AD) synchronization ensures that user identities and permissions are consistent and secure across on-premises Snorkel-hosted environments.

Prerequisites

- Working single sign-on (SSO) install. See Configure SAML SSO. If SSO is configured to be required, no user should be able to log in via username/password and bypass the AD roles sync.
- Use Active Directory as an identity provider (IdP).
- Have ability to configure additional role information and provide these roles via SAML attributes.
- Snorkel AI Data Development Platform installed via Helm on Kubernetes.

Set up AD roles in identity provider

Set up the identity provider (IdP) so that when a certain scope is requested, the `userinfo` response returns a claim that lists the AD roles or a comma-separated set of AD roles.

1. In **Admin settings**, select **Authentication Settings**.
2. **Optional:** Enter a specific admissions role.
3. Enter the scope and claim for your role.
4. Assign the specific permissions to the defined roles. Users have access to the resources based on their assigned roles.
   AD roles from the claim are a comma-separated string containing all roles or a list of strings. The list of AD roles should be formatted as `<PREFIX><SEPARATOR><WORKSPACE><SEPARATOR><SF-ROLE>` or `<WORKSPACE><SEPARATOR><SF-ROLE>`:

- `PREFIX`: Set in Helm template. If not defined, can be left empty.
- `SEPARATOR`: Character(s) to separate components of AD role. For example, `_` or `-`.
- `WORKSPACE`: Specific workspace to which role applies. For **superadmin** roles, define as `ALL_WORKSPACE`. You cannot set a role in the default workspace.
- `SF-ROLE`: The Snorkel role, such as `ANNOTATOR`.

Enabling AD Roles sync

To enable AD Roles sync for SAML, enable `authorization.adRoles` like the following:

```
authorization:
  adRoles:
    enabled: 1
    saml:
      attributeName: csgroups
    prefix: DTCA_CFG_DTI_EA_AIDLP
    separator: _$
```

- `authorization.adRoles.enabled`: Enables the feature. Notice that this is 1 and not true.
- `authorization.adRoles.saml.attributeName`: Specifies which custom attribute to look for the roles in the SAML response.
- `authorization.adRoles.prefix`: Specifies the expected prefix of the role.
- `authorization.adRoles.separator`: Specifies the expected separator between the prefix, workspace, and user role.

All of these parameters except `authorization.adRoles.prefix` must be set if AD roles sync is enabled.

Setting up Active Directory Roles sync with OIDC SSO

Active Directory (AD) synchronization ensures that user identities and permissions are consistent and secure across on-premises Snorkel-hosted environments.

Prerequisites

- Working single sign-on (SSO) install. See Configure OIDC SSO. If SSO is configured to be required, no user should be able to log in via username/password and bypass the AD roles sync.
- Use Active Directory as an identity provider (IdP).
- Have ability to configure additional role information and provide these roles via the `/userinfo` endpoint.
- Snorkel AI Data Development Platform installed via Helm on Kubernetes.

Set up AD roles in identity provider

Set up the identity provider (IdP) so that when a certain scope is requested, the `userinfo` response returns a claim that lists the AD roles or a comma-separated set of AD roles.

1. In **Admin settings**, select **Authentication Settings**.
2. **Optional:** Enter a specific admissions role.
3. Enter the scope and claim for your role.
4. Assign the specific permissions to the defined roles. Users have access to the resources based on their assigned roles.
   AD roles from the claim are a comma-separated string containing all roles or a list of strings. The list of AD roles should be formatted as `<PREFIX><SEPARATOR><WORKSPACE><SEPARATOR><SF-ROLE>` or `<WORKSPACE><SEPARATOR><SF-ROLE>`:

- `PREFIX`: Set in Helm template. If not defined, can be left empty.
- `SEPARATOR`: Character(s) to separate components of AD role. For example, `_` or `-`.
- `WORKSPACE`: Specific workspace to which role applies. For **superadmin** roles, define as `ALLWORKSPACE`. You cannot set a role in the default workspace.
- `SF-ROLE`: The Snorkel role, such as `ANNOTATOR`.

Enabling AD Roles sync

To enable AD Roles sync for OIDC, in the Helm template `values.yaml`, enable `authorization.adRoles` like the following:

```
authorization:
  adRoles:
    enabled: 1
    oidc:
      claim: csgroups
    prefix: DTCA_CFG_DTI_EA_AIDLP
    separator: _$
```

- `authorization.adRoles.enabled`: Enables the feature.
- `authorization.adRoles.oidc.claim`: Specifies which claim to look for the roles in the userinfo response.
- `authorization.adRoles.prefix`: Specifies the expected prefix of the role.
- `authorization.adRoles.separator`: Specifies the expected separator between the prefix, workspace, and user role.

All of these parameters except `authorization.adRoles.prefix` must be set if AD roles sync is enabled.

Configure access for file upload and data download controls

By default, all roles have access to upload files and download data from anywhere in the Snorkel AI Data Development Platform. You can update access controls to limit file uploads and data downloads.

> NOTE
>
> Fine-grained, role-based access controls (RBACs) are not yet available for file uploads and for data downloads. Any changes made to access controls impact all roles.

Prerequisites

To update access controls for file upload and data download, you must meet these requirements:

- An existing Snorkel AI Data Development Platform.
- An account with **Superadmin** permissions.

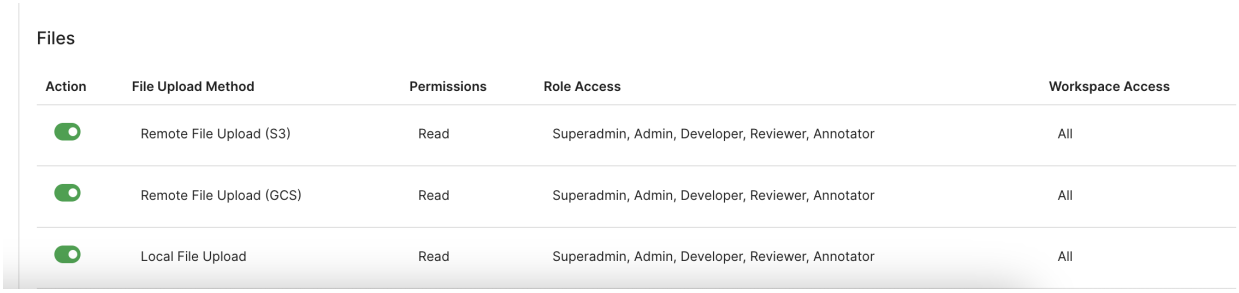
To open dataset access control

1. Click your user name on the bottom-left corner of your screen.
2. Click **Admin settings**.
3. Click **Data** under **Access Controls**.

To update access controls for file upload

In **Files** in the **Data** tab, select the toggle **Action** to enable or disable access these permissions:

- Remote file upload (S3 and GCS are separated)
- Local file upload



Files

Action	File Upload Method	Permissions	Role Access	Workspace Access
	Remote File Upload (S3)	Read	Superadmin, Admin, Developer, Reviewer, Annotator	All
	Remote File Upload (GCS)	Read	Superadmin, Admin, Developer, Reviewer, Annotator	All
	Local File Upload	Read	Superadmin, Admin, Developer, Reviewer, Annotator	All

To update access controls for data downloads

The **Download Controls** tab lists all areas in the Snorkel AI Data Development Platform where data can be exported and downloaded locally.

Click the toggle under the **Access** column to enable or disable access for these resources:

- **Node Auth Resource Type**
  - Export Annotation Batches
  - Export Studio Dataset

    - Export Studio Training Set Labels
    - Export Trained Model as CSV (Application)
    - Export Trained Model as CSV (Studio)
- Application Resource Type
    - Export Populator (Application +/- Dataset)
- Deployment Resource Type
    - Download Model (Deployment Package)
- Logs
    - Job logs
    - Download encrypted logs
    - download logs
- Dataset Resource Type
    - Multi-task Annotation

Manage RBACs for dataset connectors

This page walks through the process of controlling access to dataset upload connectors:

- **Disable** a data connector type (e.g., Local File Upload) for the **entire Snorkel AI Data Development Platform.**
- **Share** data connector credentials **securely** in-platform **within the same workspace**.
- **Restrict** data connector type usage based on **user role** and/or **workspace**.

Prerequisites

To enable dataset access controls, we require the following:

- An existing Snorkel AI Data Development Platform.
- An account with **Superadmin** permissions.

Default rules

By default, Snorkel AI Data Development Platform allows full access to each dataset upload method:

- Cloud Storage
- File Upload
- SQL DB
- Databricks SQL
- Google BigQuery
- Snowflake

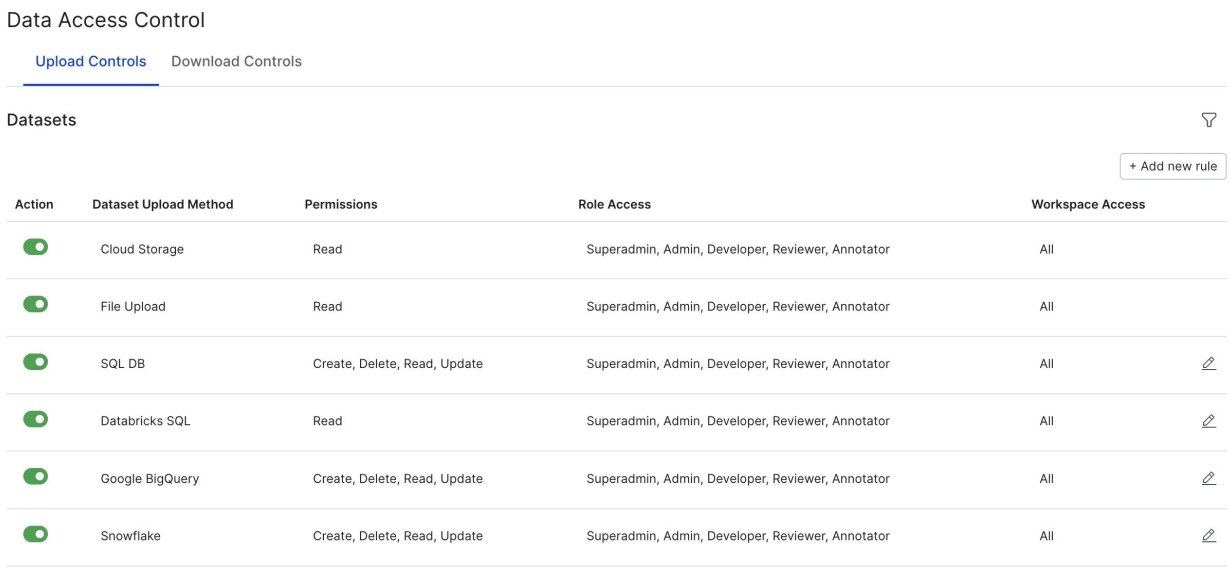
The permissions that are available differ based on the dataset upload method:

- For Cloud Storage and File Upload, the only available permission is **Read**.
- For SQL DB, Databricks SQL, Google BigQuery, and Snowflake, the permissions are **Create, Delete, Read, and Update**.

Access the dataset access control page

Follow these steps to access the Dataset Upload - Access Control page:

1. Click your user name on the bottom-left corner of your screen.
2. Click **Admin settings**.
3. Click **Data** under **Access Controls**.



Manage data connectors

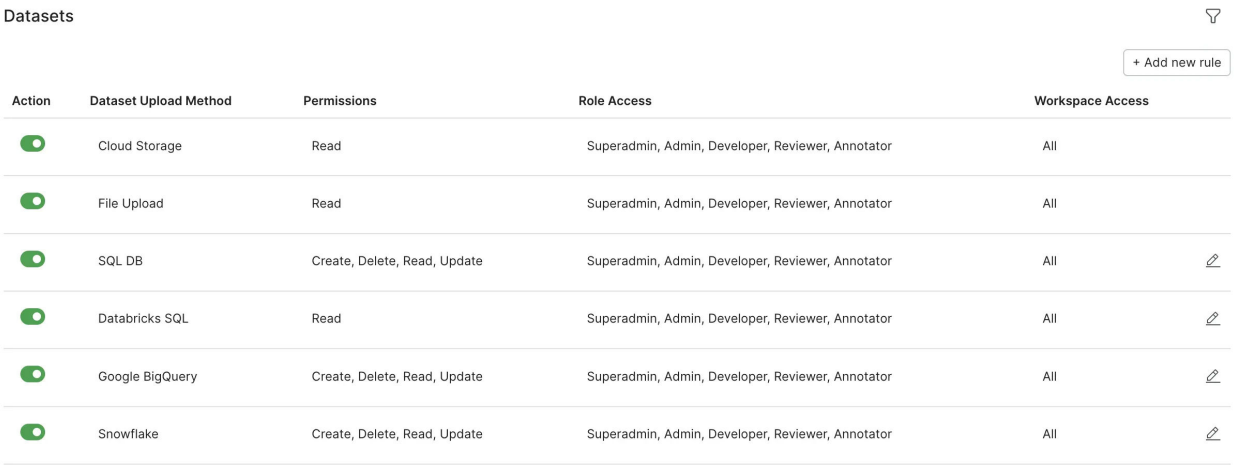
This section walks through how you can:

- Disable dataset upload methods across your Snorkel AI Data Development Platform
- Enable dataset upload methods across your Snorkel AI Data Development Platform

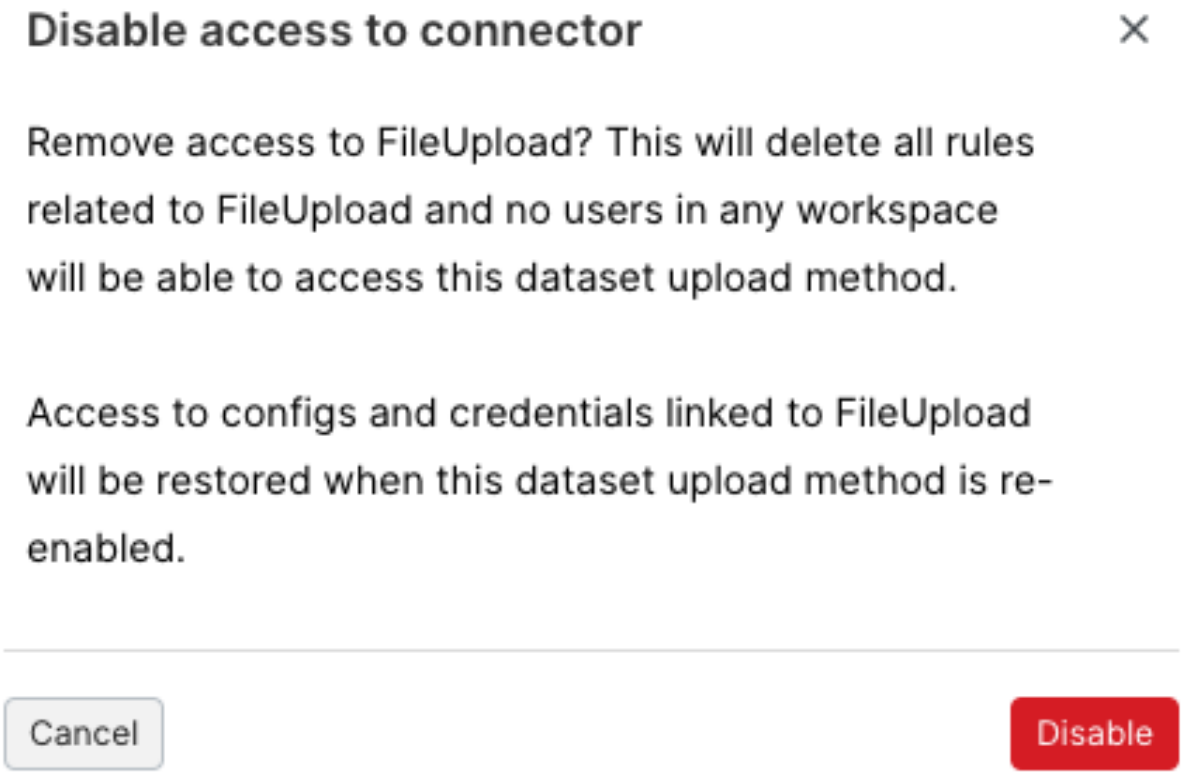
Disable a dataset upload method for entire Snorkel AI Data Development Platform

Let's say that we want to disallow local data upload for the platform. Follow these steps to disable the File Upload dataset upload method:

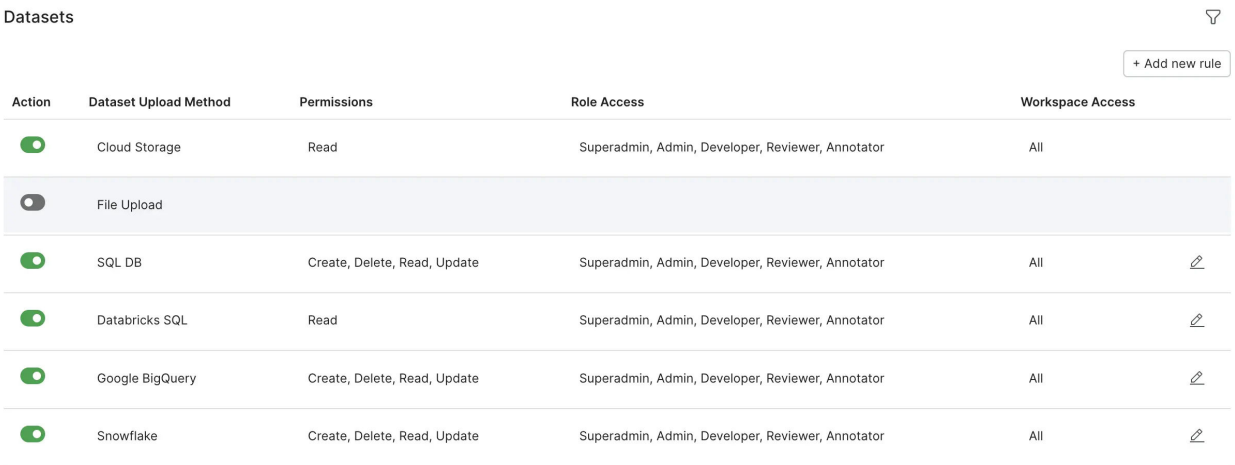
1. Navigate to the Dataset table in Upload Controls: click your user name on the bottom-left corner of your screen, click **Admin settings**, then click **Data**.



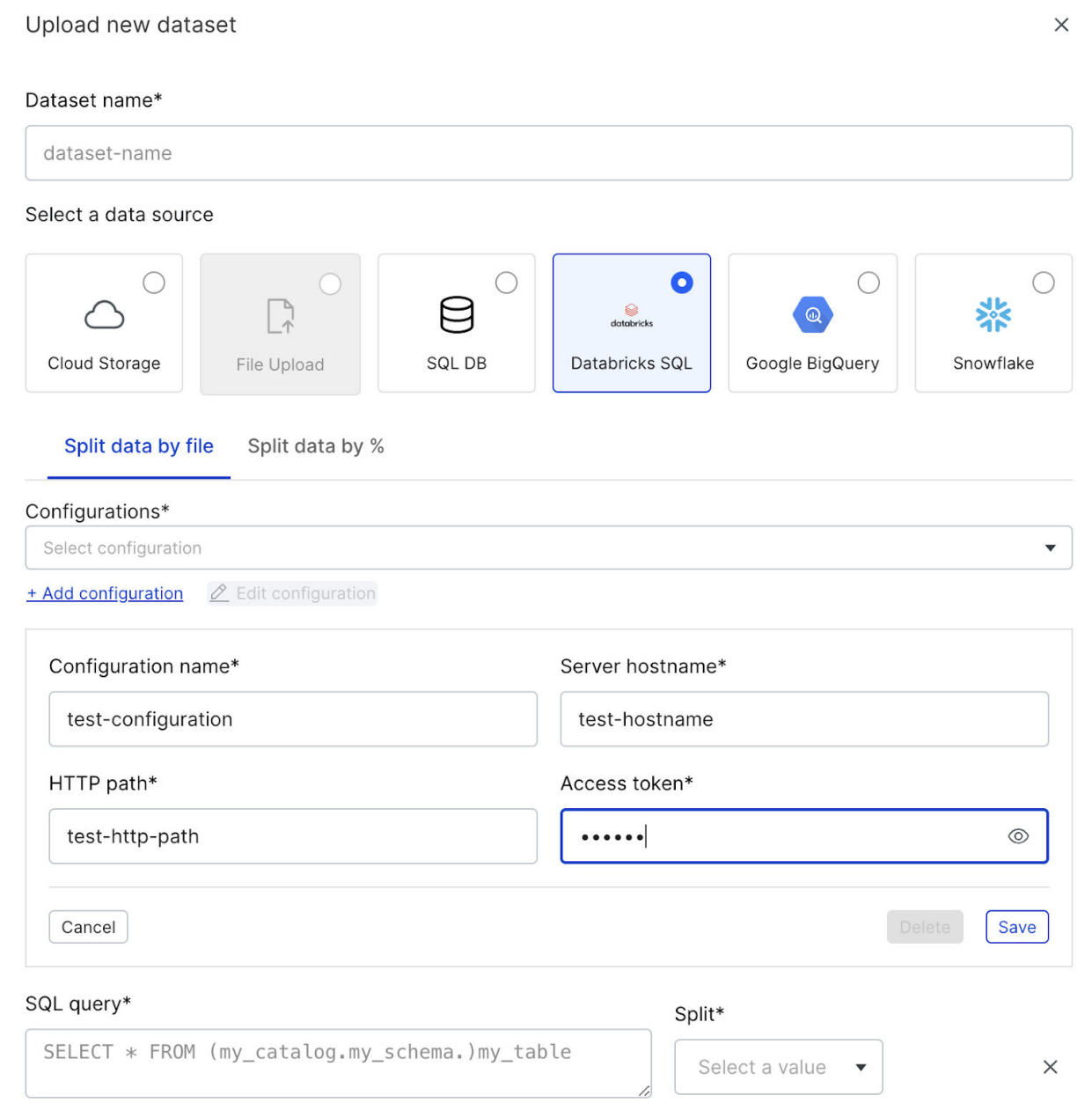
2. Click the toggle in the **Action** column for File Upload. This will bring up the following confirmation dialog.



3. Click **Disable** to disable the specified data connector (in this example, File Upload).



The disabled data connector will be reflected during dataset creation. As you can see below, the **File Upload** option is grayed out.

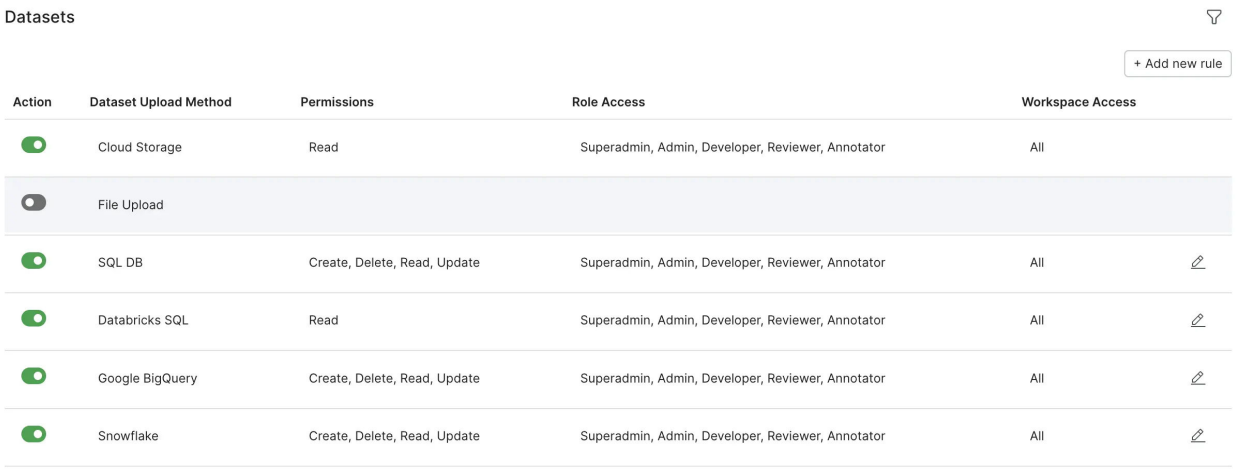


> **NOTE**
>
> Disabling a data connector will delete all existing rules that have been created. When reactivating a dataset upload method, the default role is populated again (i.e., full permissions for all roles for all workspaces).

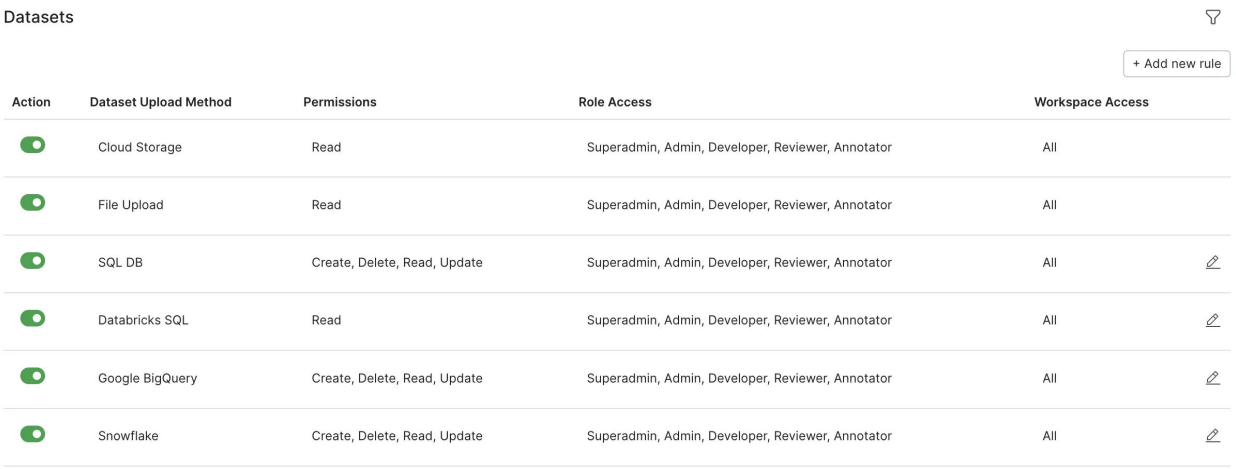
Enable a dataset upload method for entire Snorkel AI Data Development Platform

Let's say that we've disabled the File Upload dataset upload method, but now want to re-enable it. Follow these steps enable local data upload:

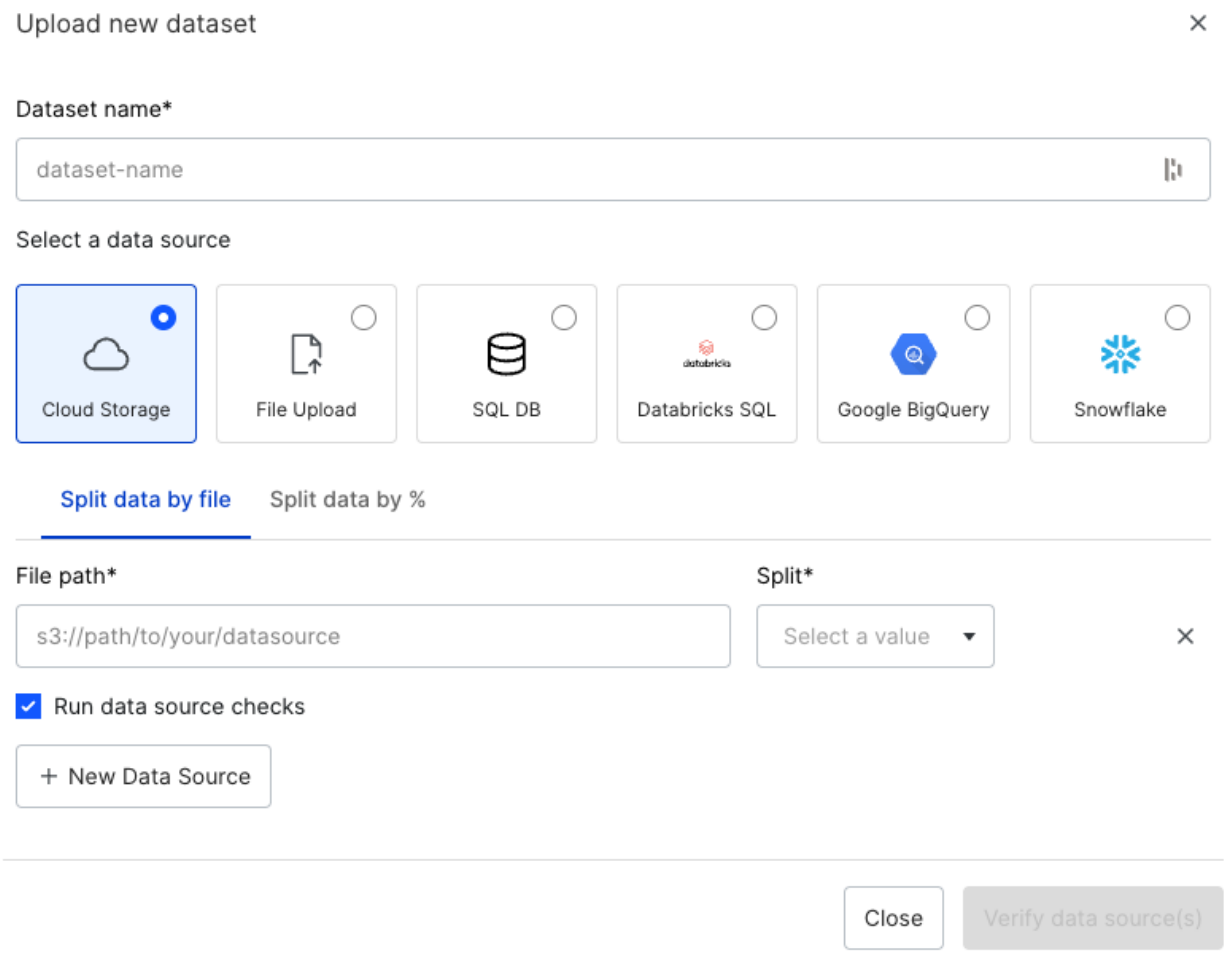
1. Navigate to the Dataset Upload - Access Control page: click your user name on the bottom-left corner of your screen, click **Admin settings**, then click **Data**.



2. Click the toggle in the **Action** column for File Upload.



The enabled data connector will be reflected during dataset creation. As you can see below, the **File Upload** option is not grayed out.



Share data connector credentials securely in-platform

This section walks through how you can:

- Create a data connector configuration
- Ingest an existing data connector configuration

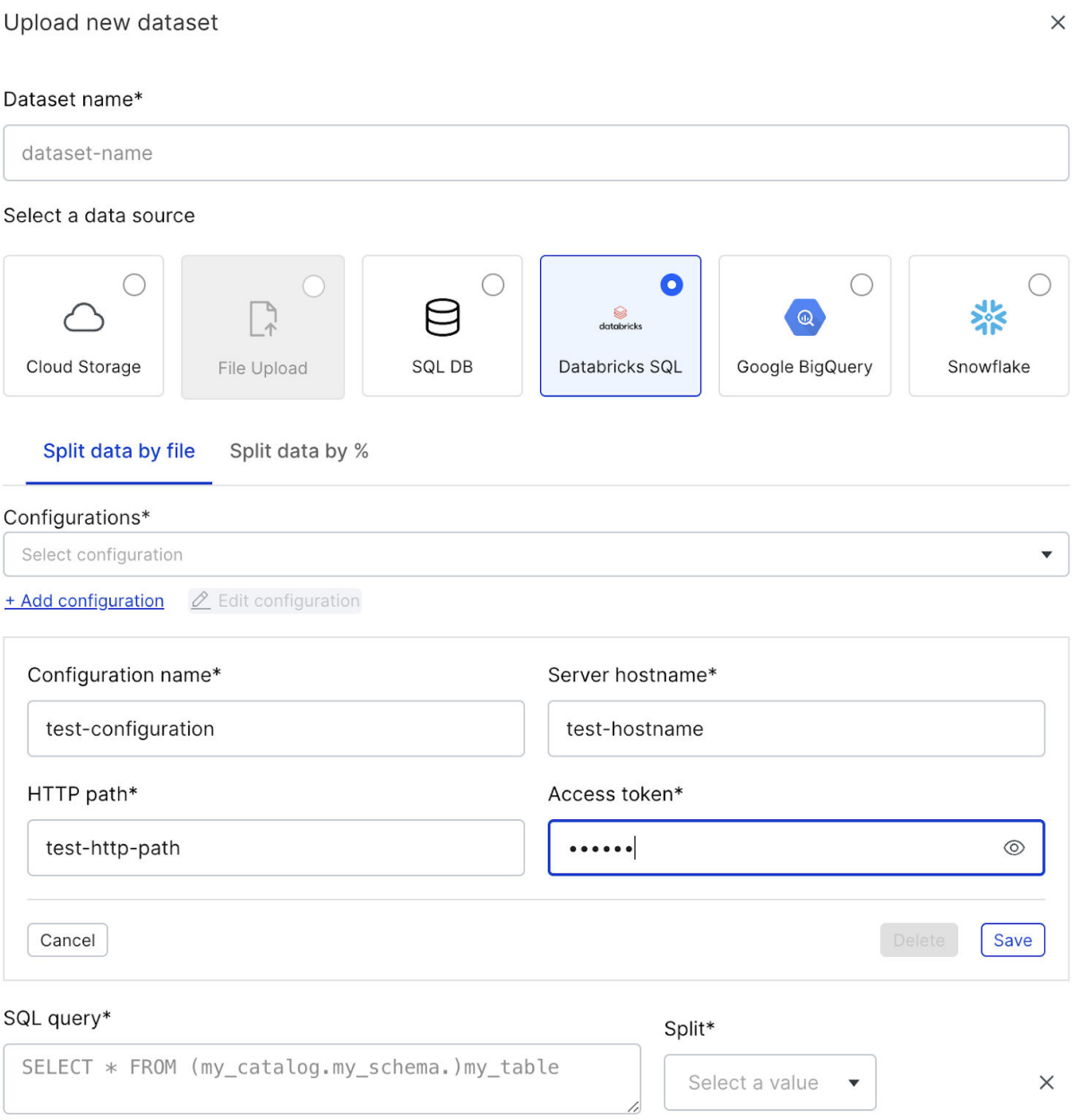
Create a data connector configuration

Follow these steps to create a data connector configuration:

1. In the left-side menu, click **Datasets**.
2. In the top left corner of the Datasets page, click **+ Upload new dataset**. Currently, we support storing connector credentials for the following connectors:
   - SQL DB
   - Databricks SQL
   - Google BigQuery
   - Snowflake
3. Pick one of the dataset upload methods, then click **+ Add configuration**. You will see configuration fields pop up on your screen. Different connector configs will have different fields that are relevant for the connector. The example below shows the configuration options for the Databricks SQL connector.

4. Fill out the configuration fields, then click **Save**.
5. Once saved, the configuration will be available for everyone with **Read** access to configs in that particular workspace.

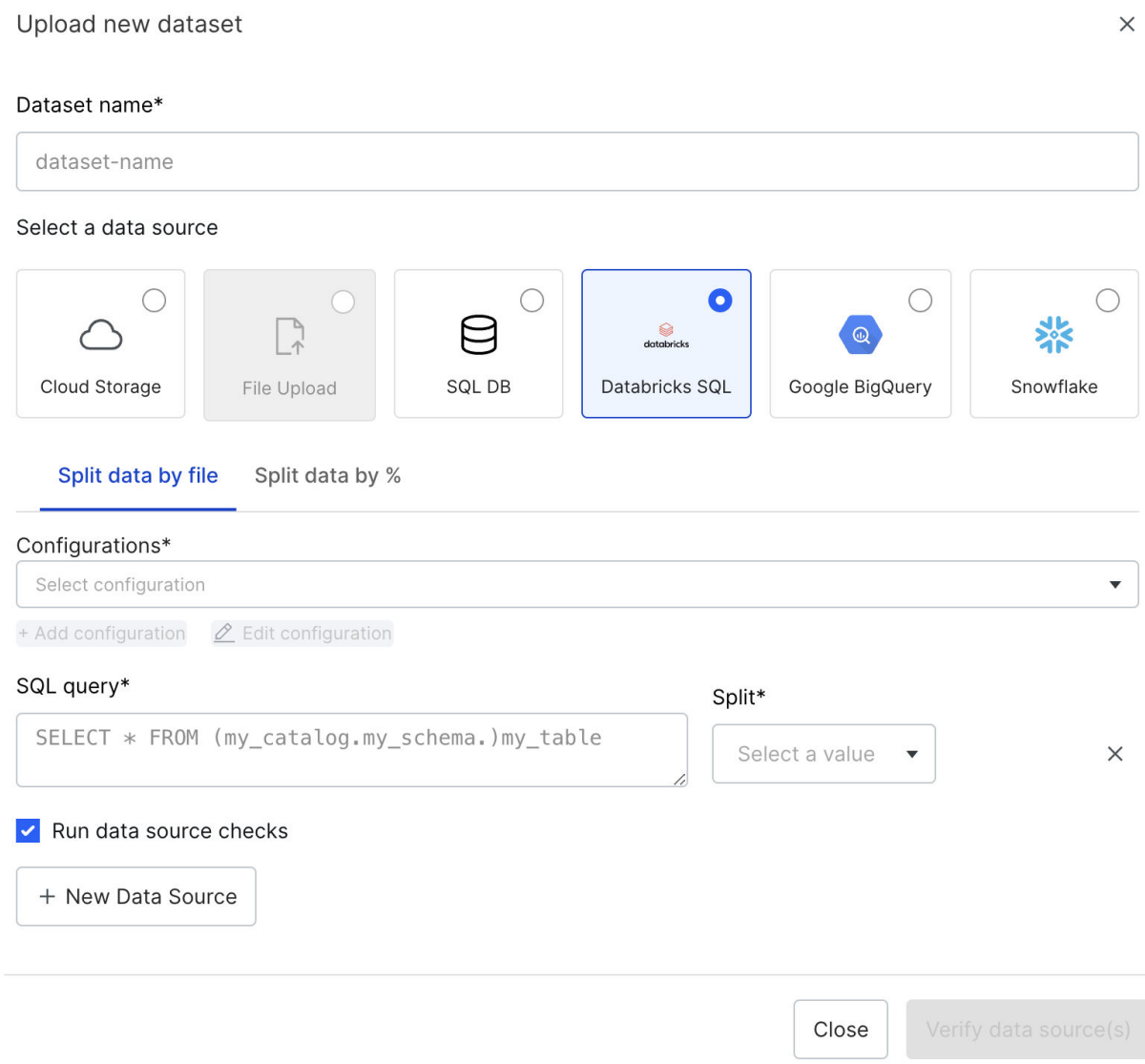
> **NOTE**
>
> Users with **Create, Delete, Read, Update** permissions will be able to modify the configuration, but will never be able to retrieve the **secrets** field from the configuration (e.g., the **Access token** field for DatabricksSQL).



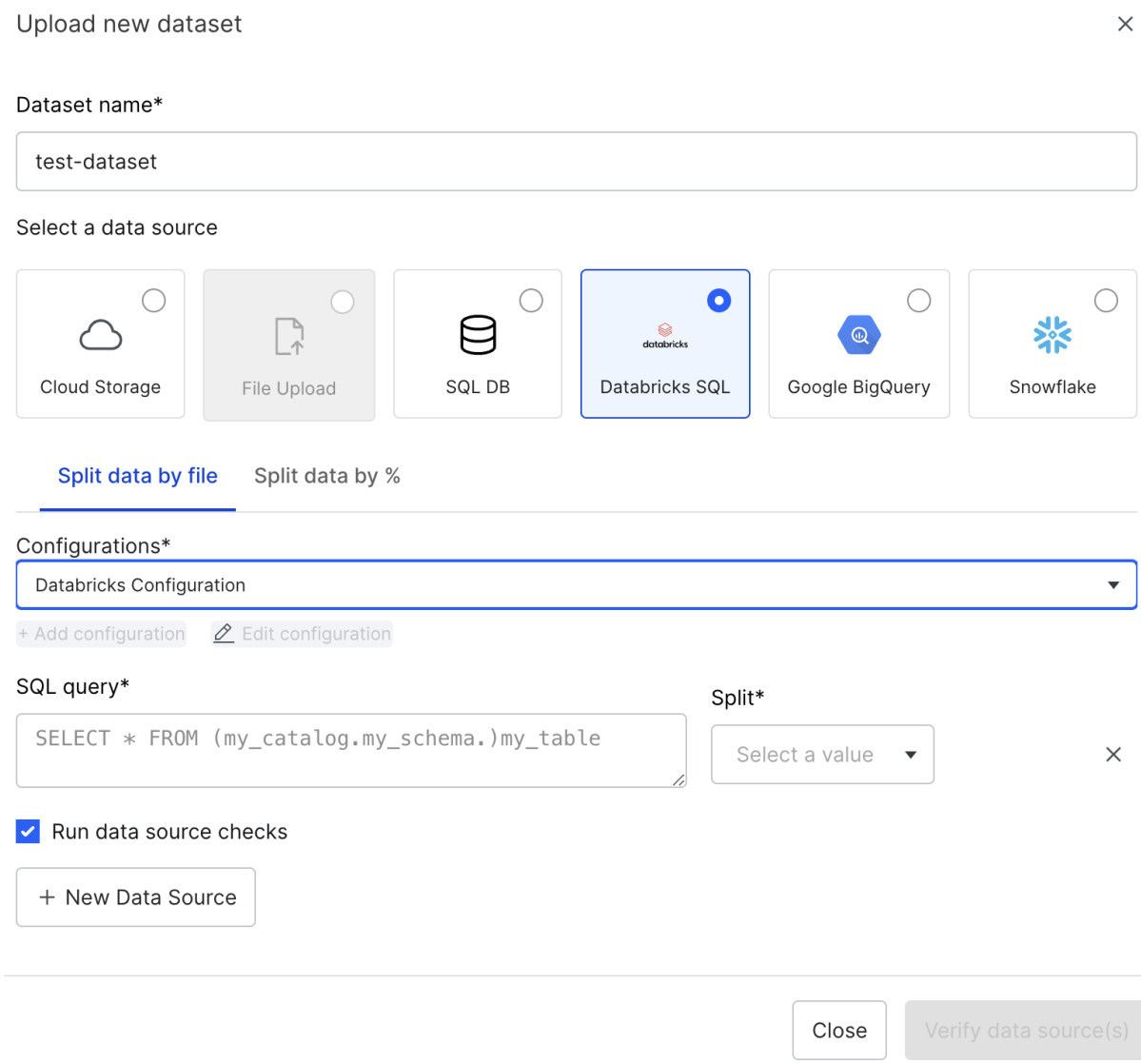
Ingest a data source with an existing configuration

Users with **Read** access cannot create a new configuration, but can ingest any existing configurations. For example, let's say that we are a user that wants use an existing Databricks SQL configuration.

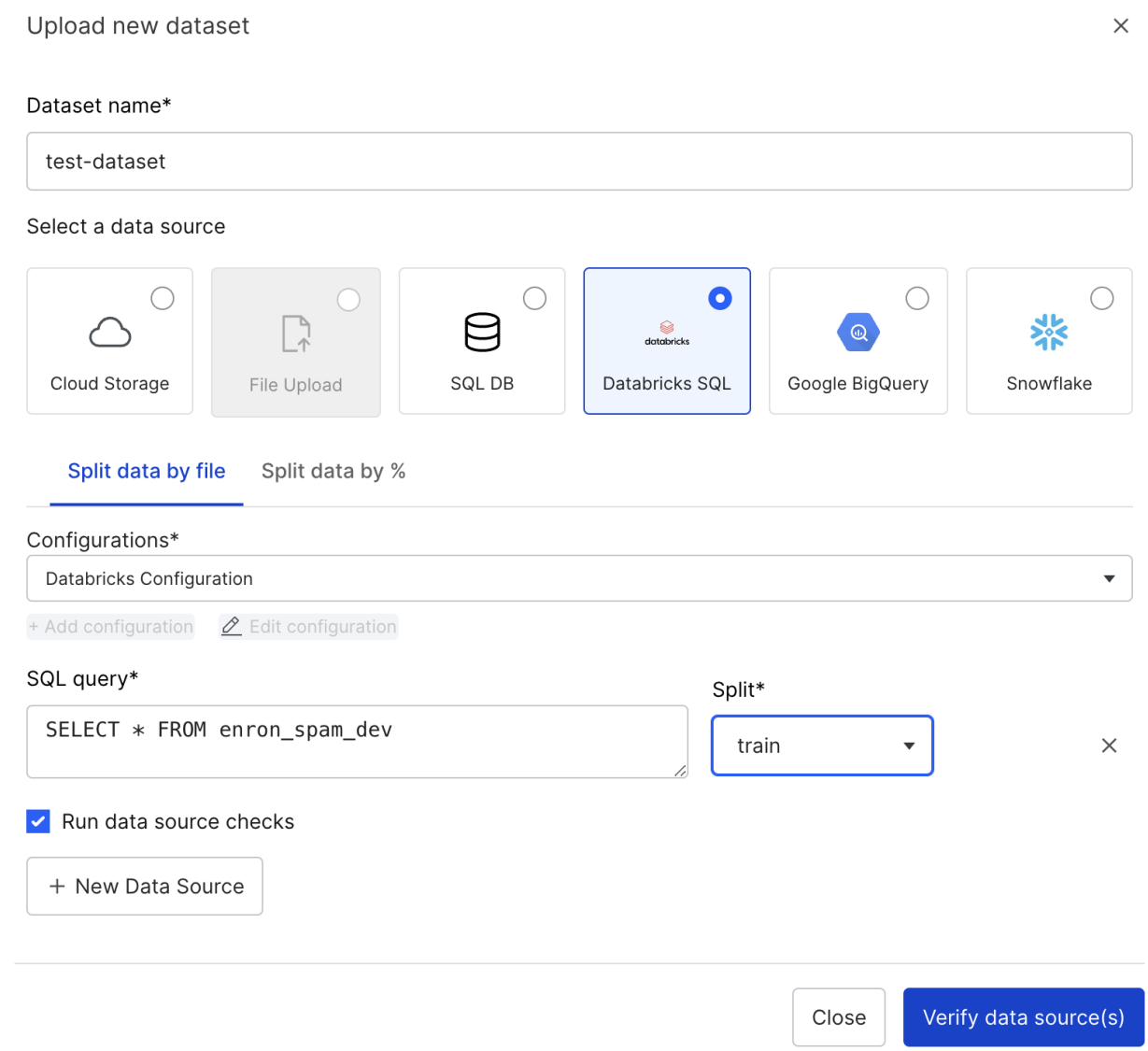
1. Click **Datasets** in the left-side menu to open the Datasets page.
2. Click **+ Upload new dataset** to begin data creation.
3. Under **Select a data source**, click **Databricks SQL**.



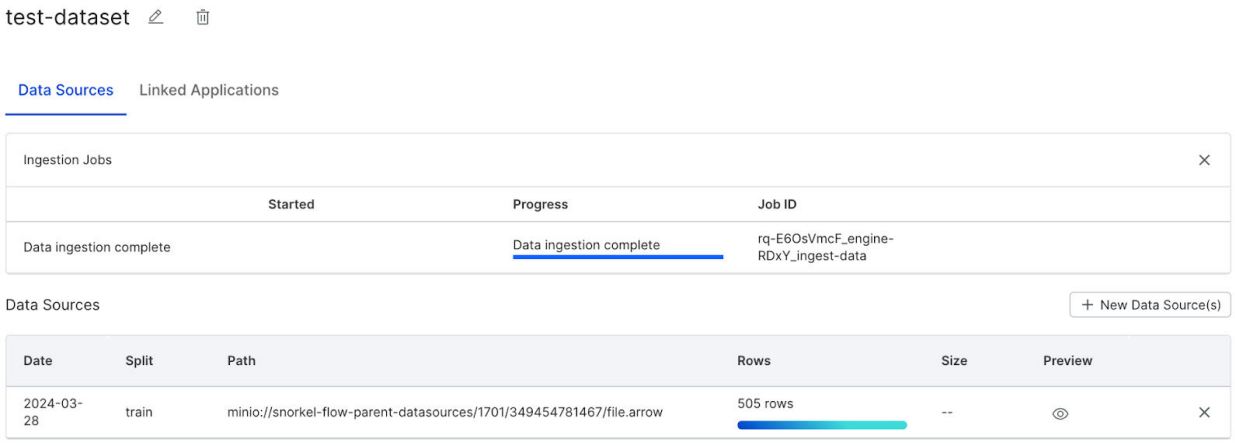
4. Give the dataset a name (in this example, test-dataset).
5. Select **Databricks Configuration** from the **Configurations** drop down.
   - In the **Configurations** drop down, you will see all configs that are available to you in the workspace.
   - Notice that the **Edit configuration** button is grayed out as you don't have permission to see or edit it.



6. Finish the standard flow for data ingestion (e.g., enter a **SQL Query** and specify a data Split), then click **Verify data source(s)**.



Notice that the data has been successfully ingested from Databricks, without the need to provide server details or access credentials.



Fine grained data connector rules for different roles

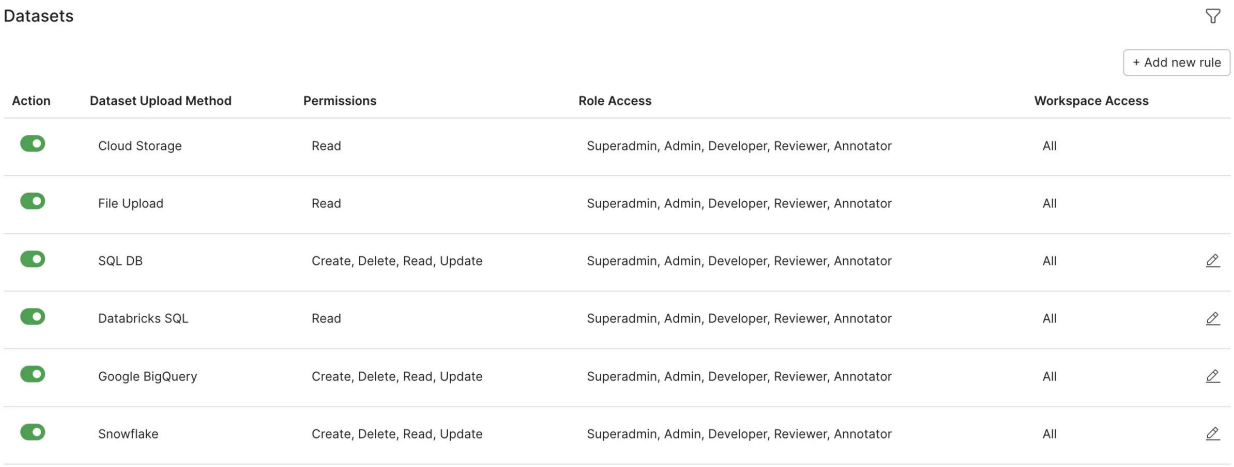
This section walks through how you can:

- Create or update rules
- Delete rules

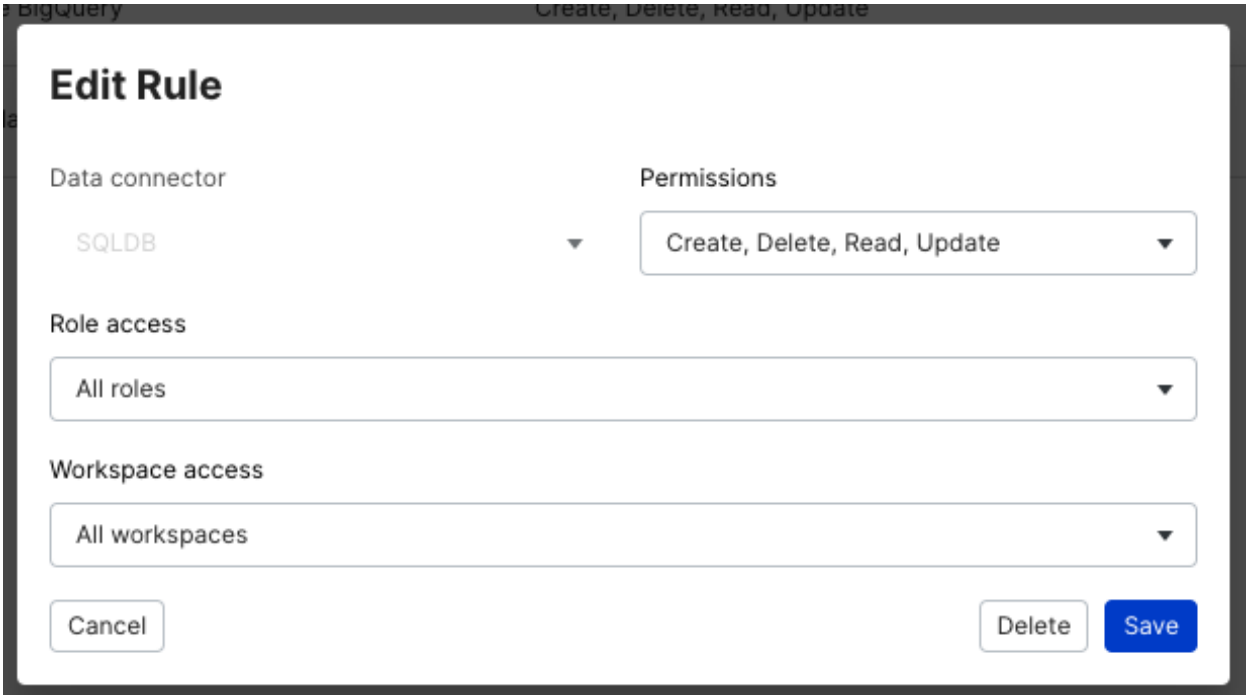
Create or update rules

There may be situations in which you'll want to assign different levels of permissions to different roles. For example, let's say that superadmins and admins should be the only ones able to create, delete, and update configurations for SQL DB. Annotators and reviewers should still be able to read and use the configurations for data ingestion, and developers should not be able to read or use the configurations. Follow these steps to create the necessary rules:

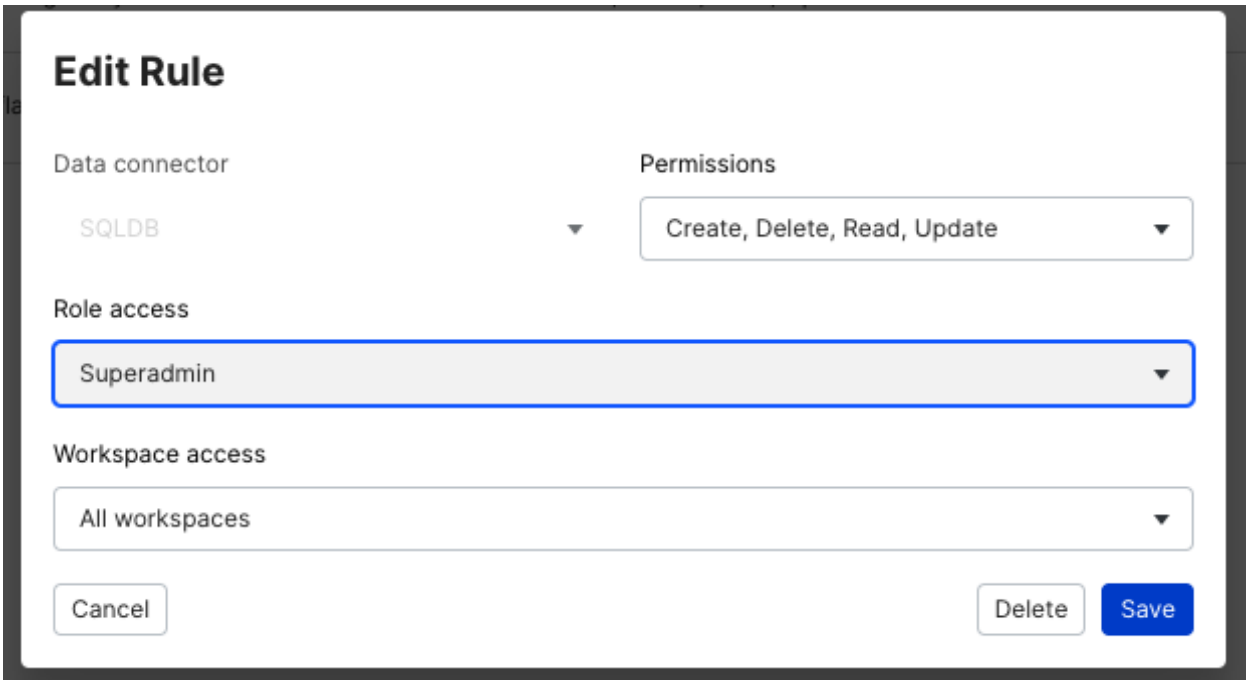
1. Navigate to the Dataset Upload - Access Control page: click your user name on the bottom-left corner of your screen, click **Admin settings**, then click **Data**.



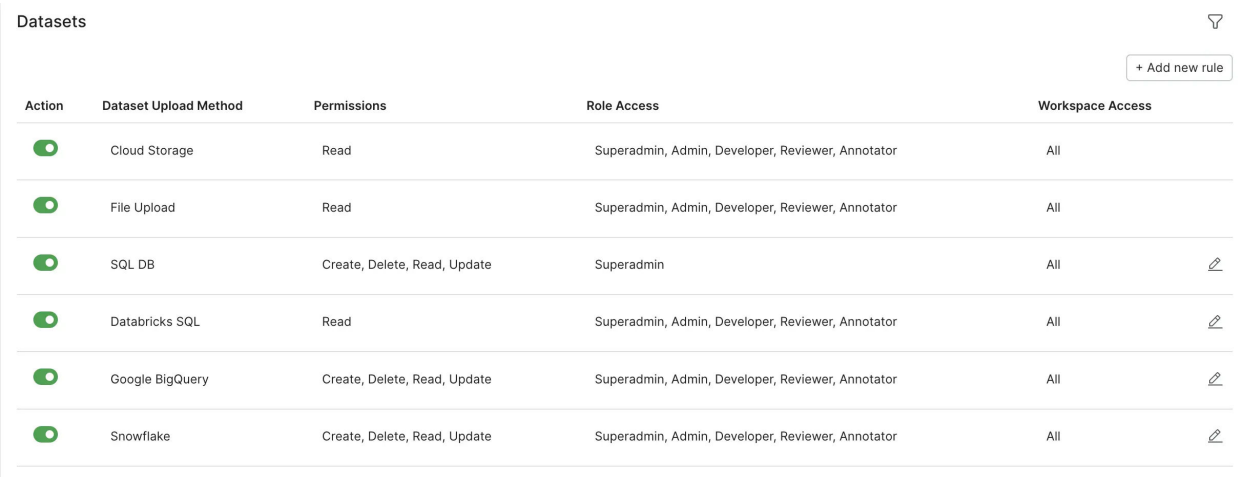
2. In this example, we want to edit the initial, pre-populated rule for SQL DB. To do this, click the pencil icon for the SQL DB rule.



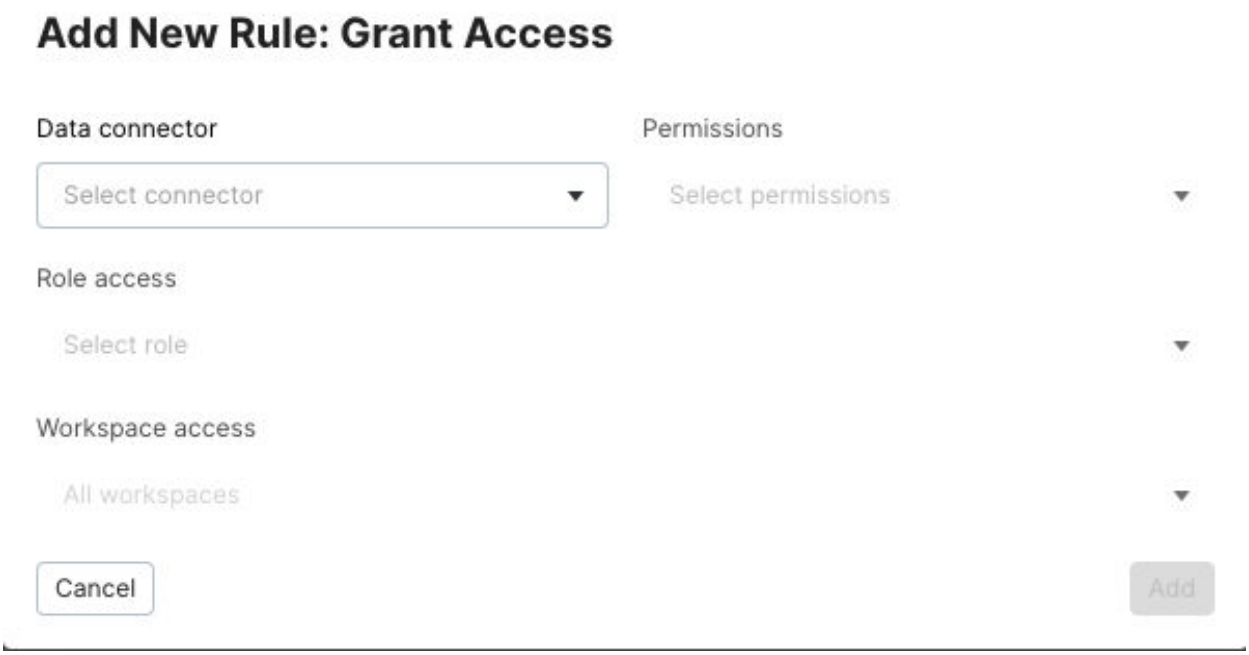
3. Select **Superadmin** in the **Role access** drop down.



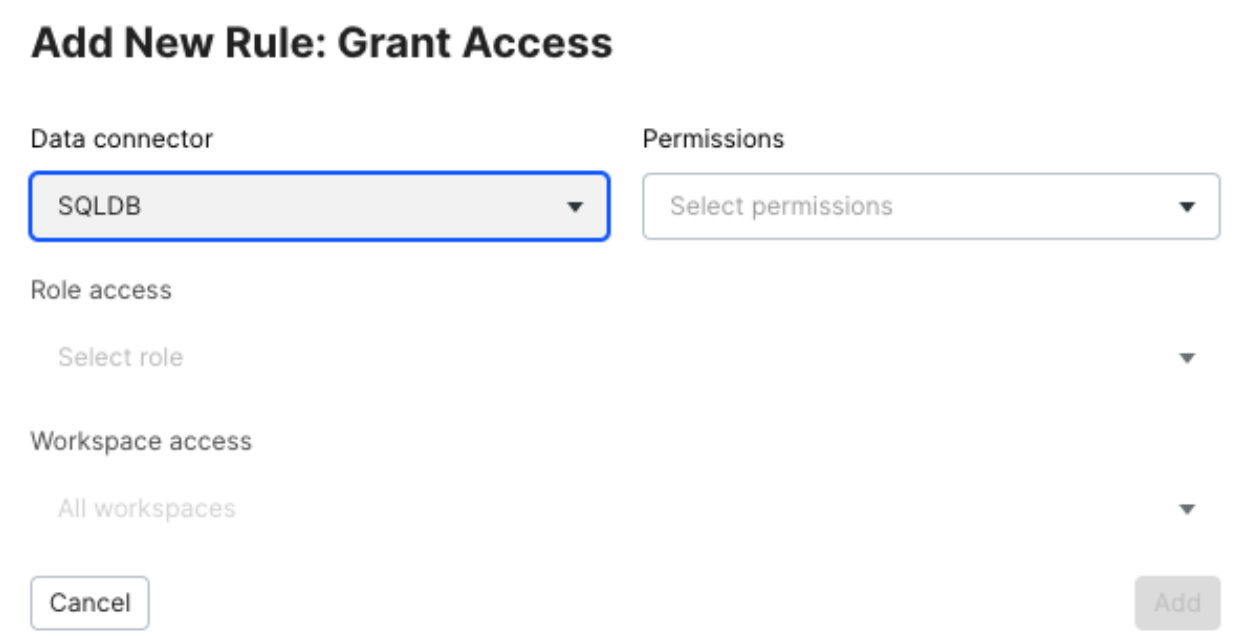
4. Click **Save** to save the rule. This will automatically bring you back to the Data Access Control page.



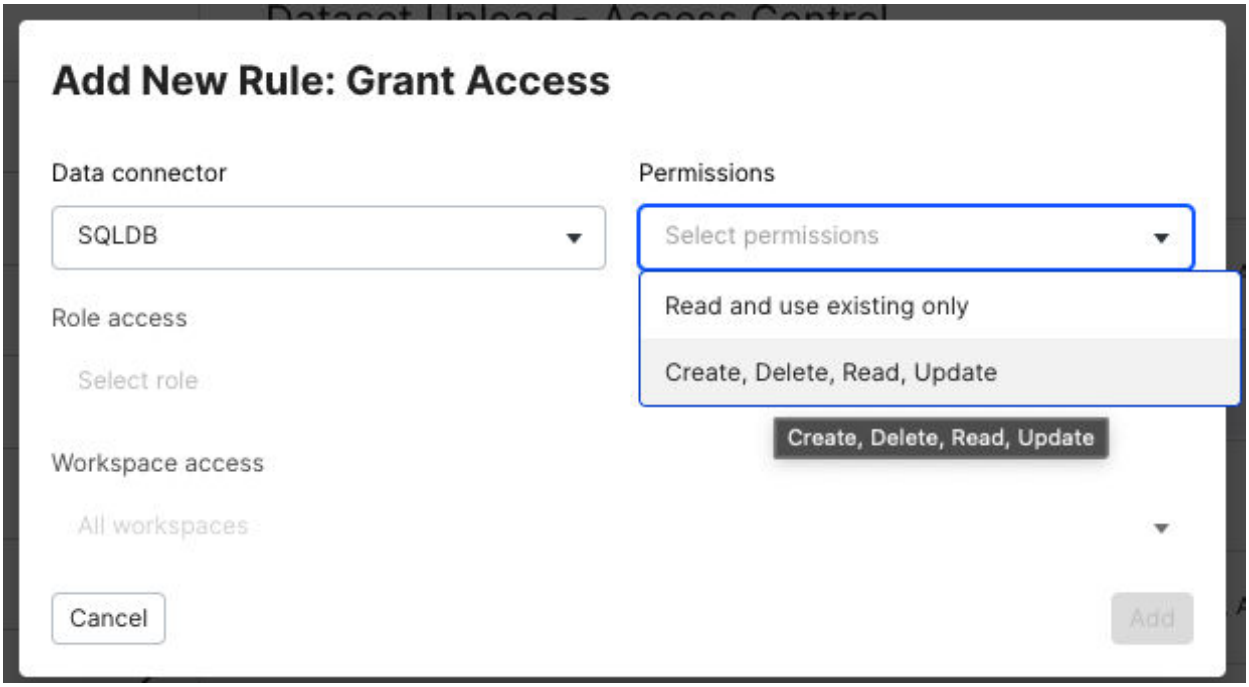
5. Next, let's create a new rule to allow create, read, update, and delete permissions for admins. Click **+ Add new rule** in the top-right corner of the Dataset Upload - Access Control page. This will bring up a dialog to create a new access rule.



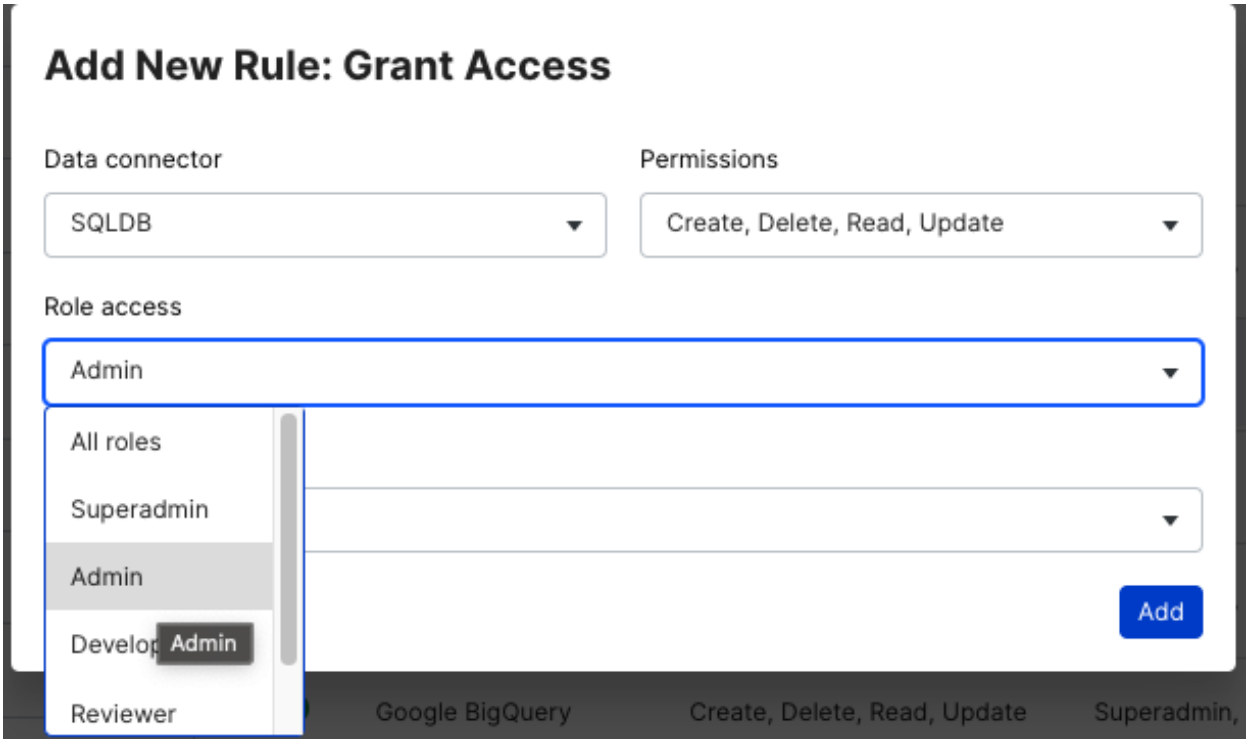
6. Select **SQLDB** in the **Data connector** drop down.



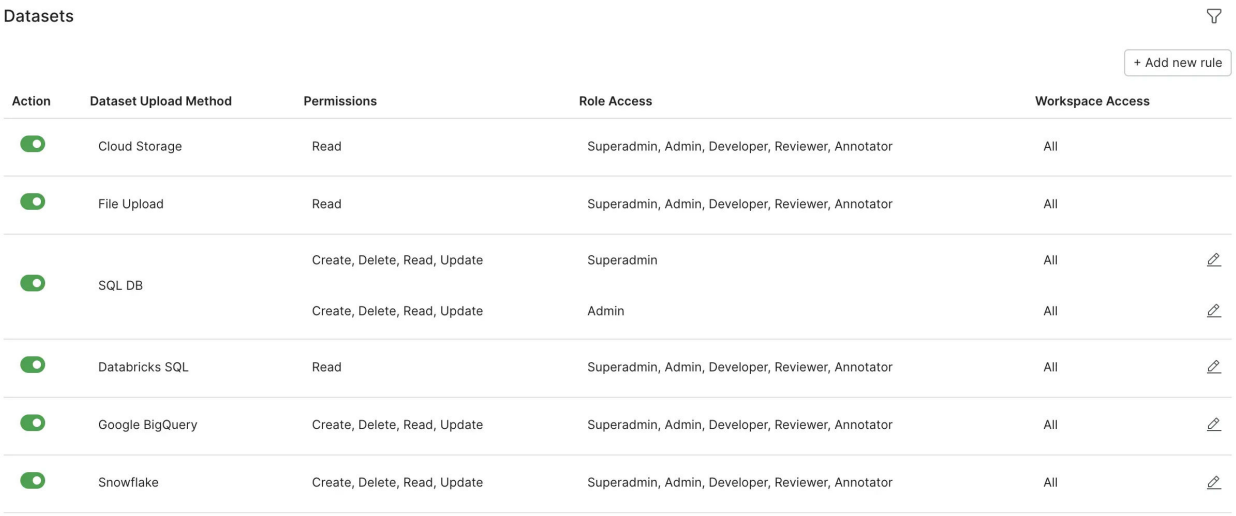
7. Select **Create, Delete, Read, Update** in the **Permissions** drop down.



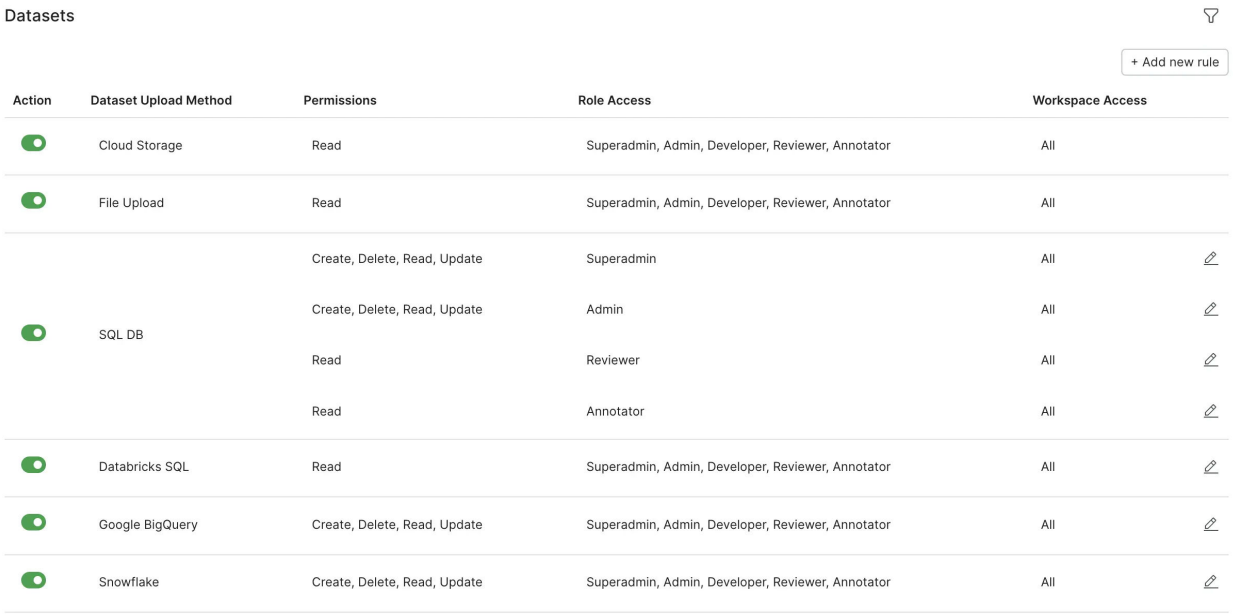
8. Select **Admin** in the **Role access** drop down.



10. Click **Add** to save the new rule. This will automatically bring you back to the main page.



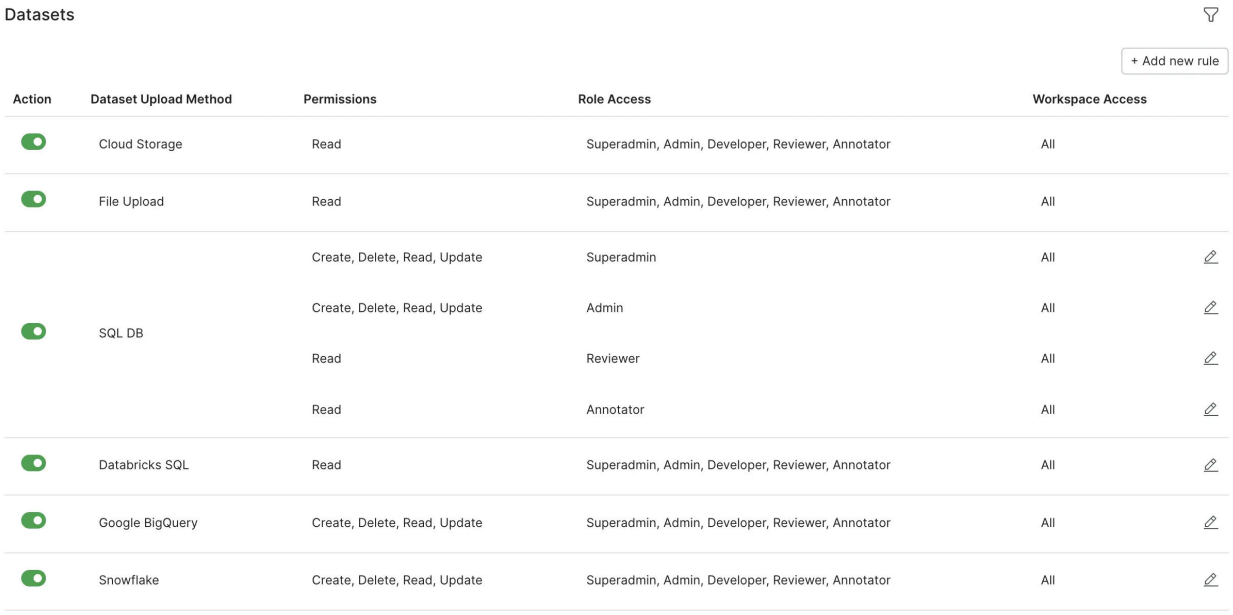
Similarly, you can create rules for **Read** permissions for annotators and reviewers. Note that if there is no rule for the developer role for a particular workspace, then developers will not have access to the data connector. This means that the data connector option will be grayed out for them during dataset creation.



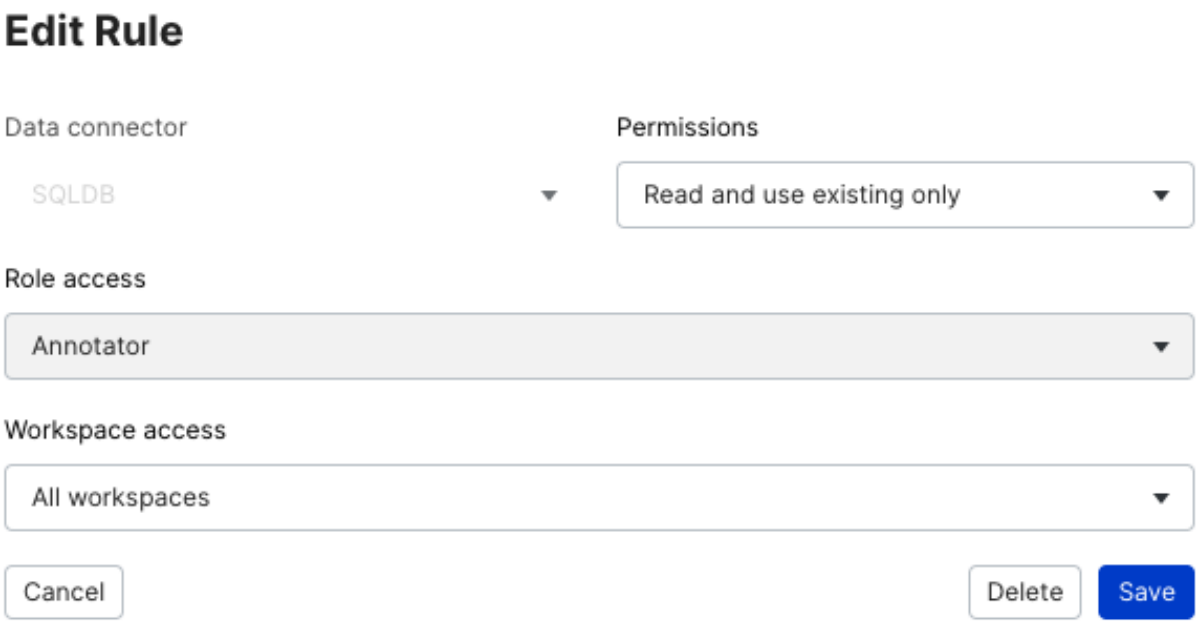
Delete rules

Let's say that we no longer want a rule that enables annotators **Read** access to the SQL DB. Follow these steps to delete the rule:

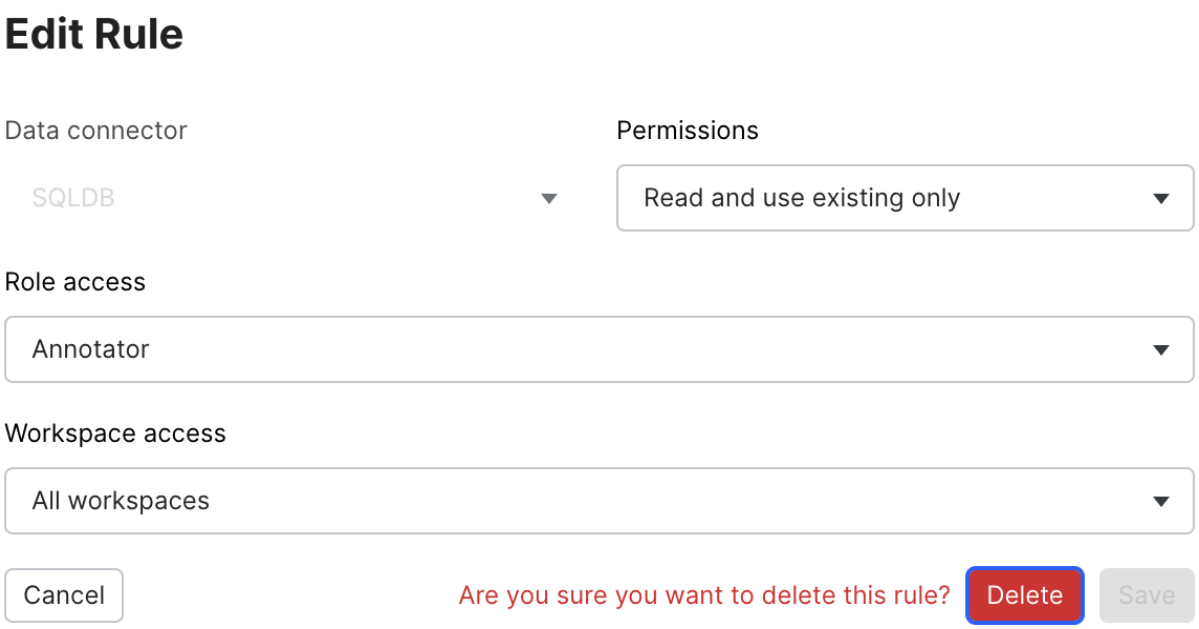
1. Navigate to the Dataset Upload - Access Control page: click your user name on the bottom left corner of your screen, click **Admin settings**, then click **Data**.



2. Click the pencil icon next to the rule you wish to remove. In this example, we will select the SQL DB Annotator Rule with **Read** Permissions.



3. Click **Delete**. This will bring up a confirmation message.



4. Click **Delete** one more time to delete the rule.

Datasets

+ Add new rule

Action	Dataset Upload Method	Permissions	Role Access	Workspace Access	
ON	Cloud Storage	Read	Superadmin, Admin, Developer, Reviewer, Annotator	All	
ON	File Upload	Read	Superadmin, Admin, Developer, Reviewer, Annotator	All	
		Create, Delete, Read, Update	Superadmin	All	✎
ON	SQL DB	Create, Delete, Read, Update	Admin	All	✎
		Read	Reviewer	All	✎
ON	Databricks SQL	Read	Superadmin, Admin, Developer, Reviewer, Annotator	All	✎
ON	Google BigQuery	Create, Delete, Read, Update	Superadmin, Admin, Developer, Reviewer, Annotator	All	✎
ON	Snowflake	Create, Delete, Read, Update	Superadmin, Admin, Developer, Reviewer, Annotator	All	✎

Create RBAC-based Workspaces for the Snorkel AI Data Development Platform

In this walkthrough, we demonstrate how to use workspaces in the Snorkel AI Data Development Platform. Resources such as applications and datasets in one workspace will be isolated from other workspaces.

Create a Workspace

> NOTE
> You have to be a superadmin to perform workspace creation.

1. Click on the **Admin settings** page on the sidebar menu and click **Workspace Management**.
2. On the management page, click on the **Create Workspace** button, and you will see **Create Workspace** dialog.
3. Enter the relevant information for creating a workspace:

- **Workspace Name** - Any name you'd like for the workspace (50-character limit).
- **Dataset Limit** - Number of datasets you'd like to allocate to the workspace.
- **Application Limit** - (Optional) Number of applications you'd like to allocate to the workspace. The default is unlimited.
- **Assign User** - (Optional) Choose the users you want to add and their corresponding roles in this workspace. You can assign users after workspace creation if you want.

> NOTE
> The dataset limit and application limit are subject to the license limit.

Once you have entered all the information, click the **Save** button.

Edit a Workspace

> NOTE
> You must be a superadmin or admin of the workspace to edit a workspace.

1. Click on the **Admin settings** page on the sidebar menu and click **Workspace Management**.
2. On the management page, click on the **edit** icon of the workspace, and you will see the **Edit Workspace** dialog.

3. Edit the fields you want to change.

- **Workspace Name** - Change to any name you'd like for the workspace (50-character limit).
- **Dataset Limit** - Number of datasets you'd like to allocate to the workspace.
- **Application Limit** - Number of applications you'd like to allocate to the workspace. The default is unlimited.
- **Assign User** - Add users to or remove users from the workspace or change the roles of existing users.

Once you have changed the fields you want, click the **Save** button.

Switch Workspaces

On the sidebar menu, click on the **Workspace** dropdown to see the available workspaces to choose from. Click on the workspace you want to switch to. You will only see resources (applications, datasets, etc.) in that workspace.

> NOTE
>
> You can only see workspaces you have access to.

Feature Access Management

This document explains how to control access to features. By the end of this guide, you will be able to customize users' access to different features in the platform, including notebooks and deployments.

Prerequisites

- An existing Snorkel AI Data Development Platform
- An account with Superadmin permissions

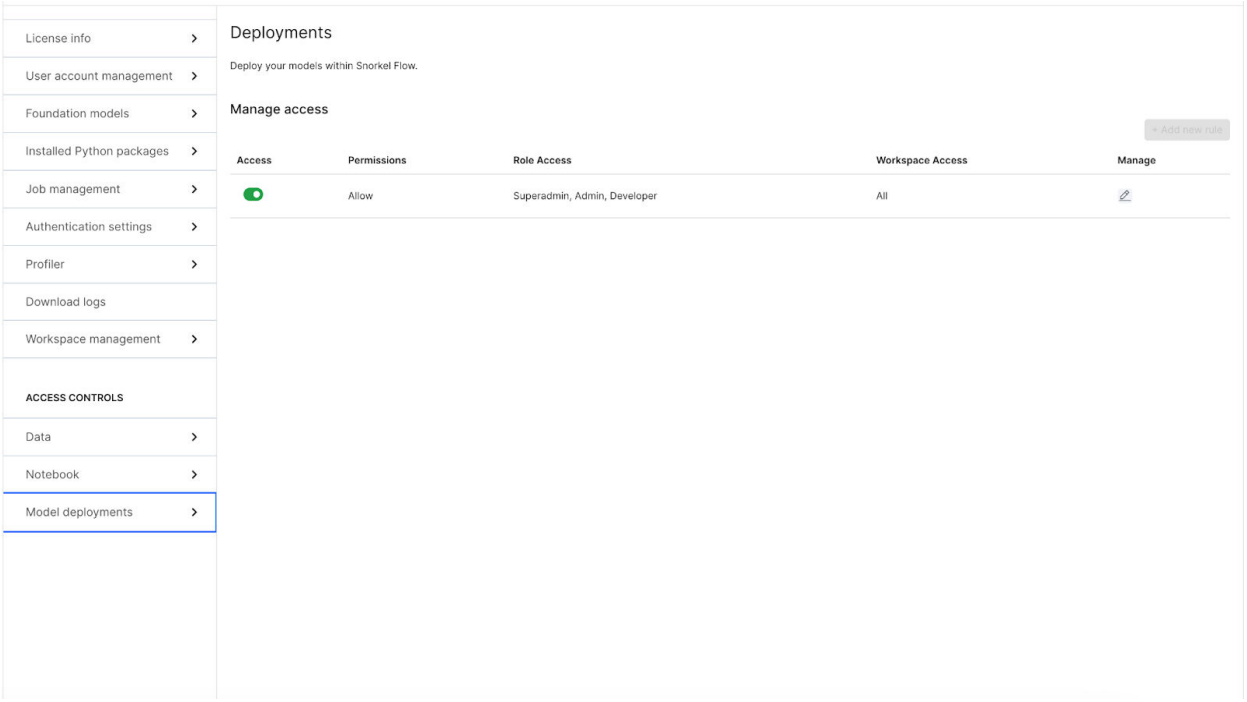
Use Cases

This document describes the process of:

- Enabling and disabling access to model deployments for the entire Snorkel AI Data Development Platform
- Enabling and disabling access to single-user notebooks in Jupyterhub for the entire Snorkel AI Data Development Platform

Feature Access Control Page

This example shows the feature for model deployments.

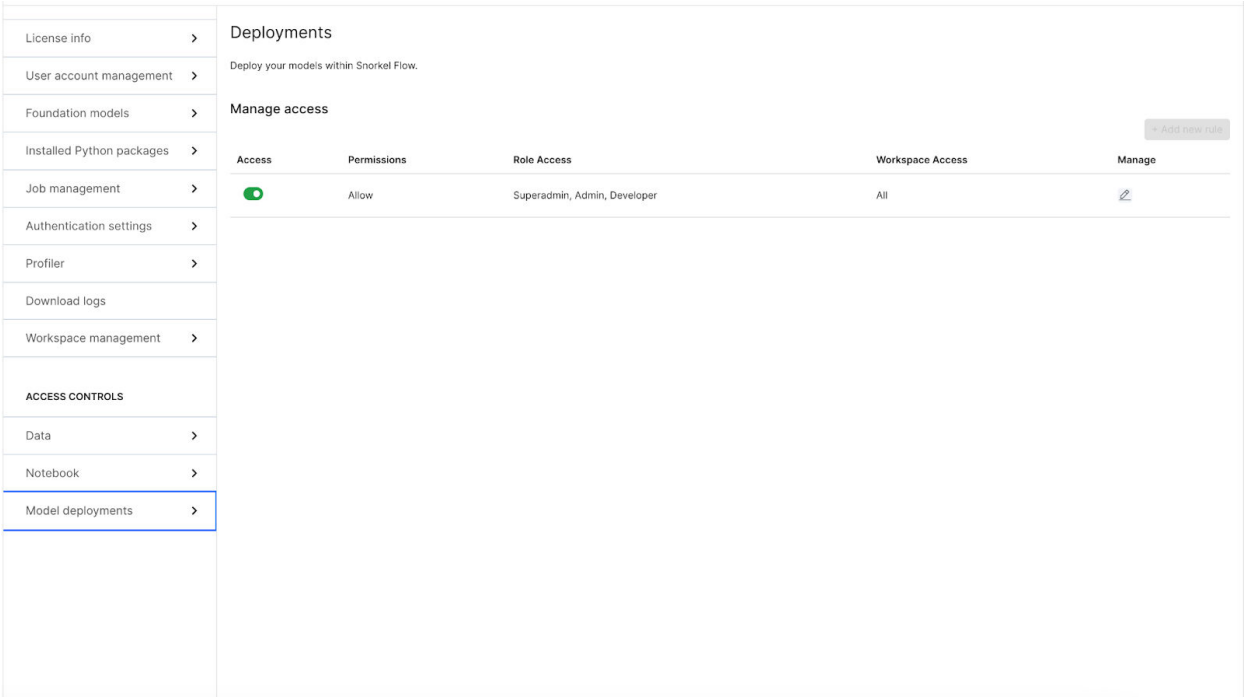


When a feature is enabled, it can be accessed by roles specified in the Role Access column. When a feature is disabled, no user in the instance will be able to access the feature. Currently, there is no way to modify the role access or workspace access attributes.

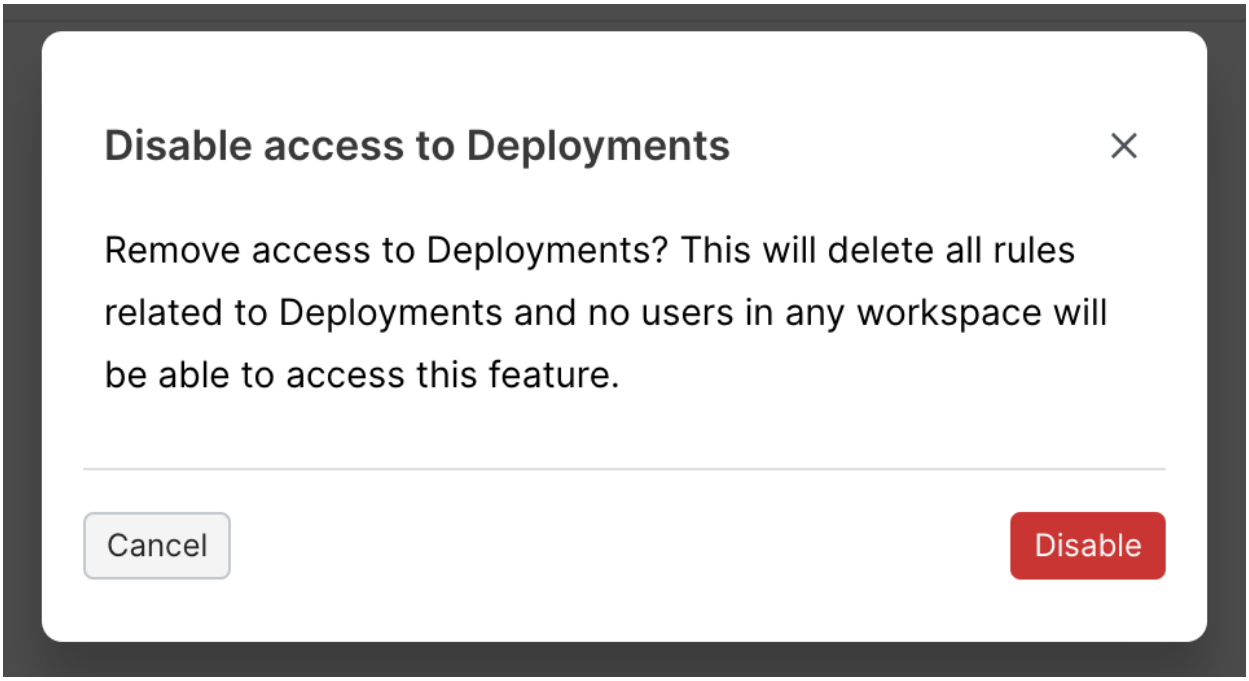
> Note: If the "Notebook" feature is enabled, users with the "developer" role or above in any workspace will be able to use notebooks in all workspaces the user has access to. This is a technical requirement in Jupyterhub and not the Snorkel AI Data Development Platform.

Disabling a Feature

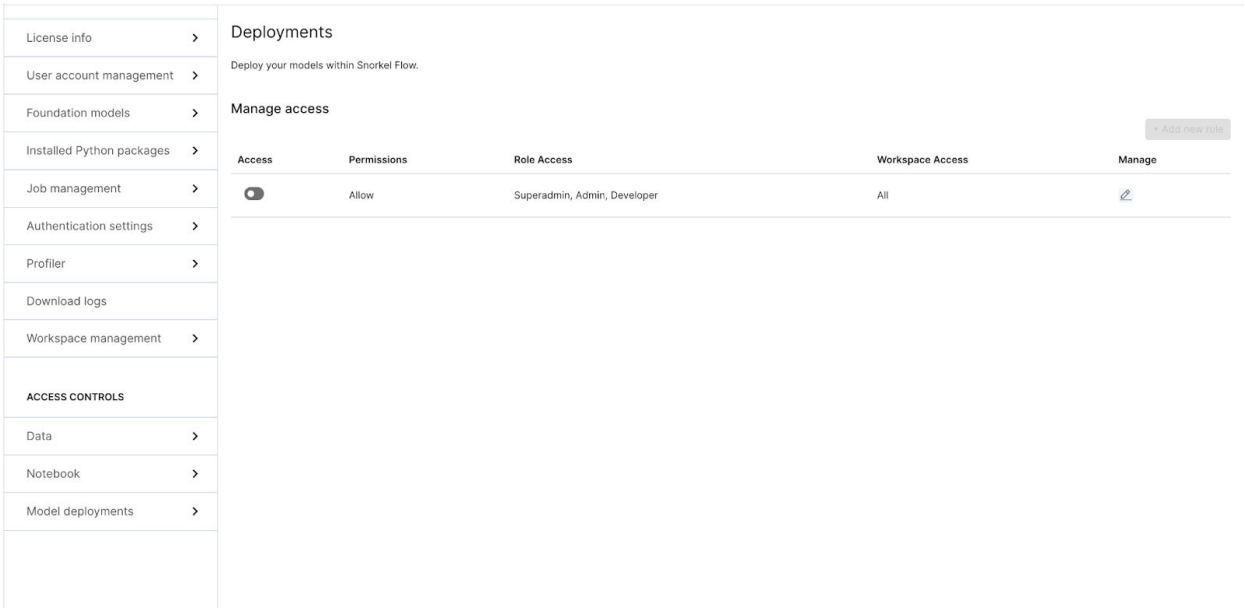
1. Open the **Admin Settings** page and navigate to a feature page under **Access Controls**. This example uses **Model deployments**.



2. Switch the toggle button under **Access** to disallow deployment functionality for everyone in all workspaces. This brings up a confirmation dialog.



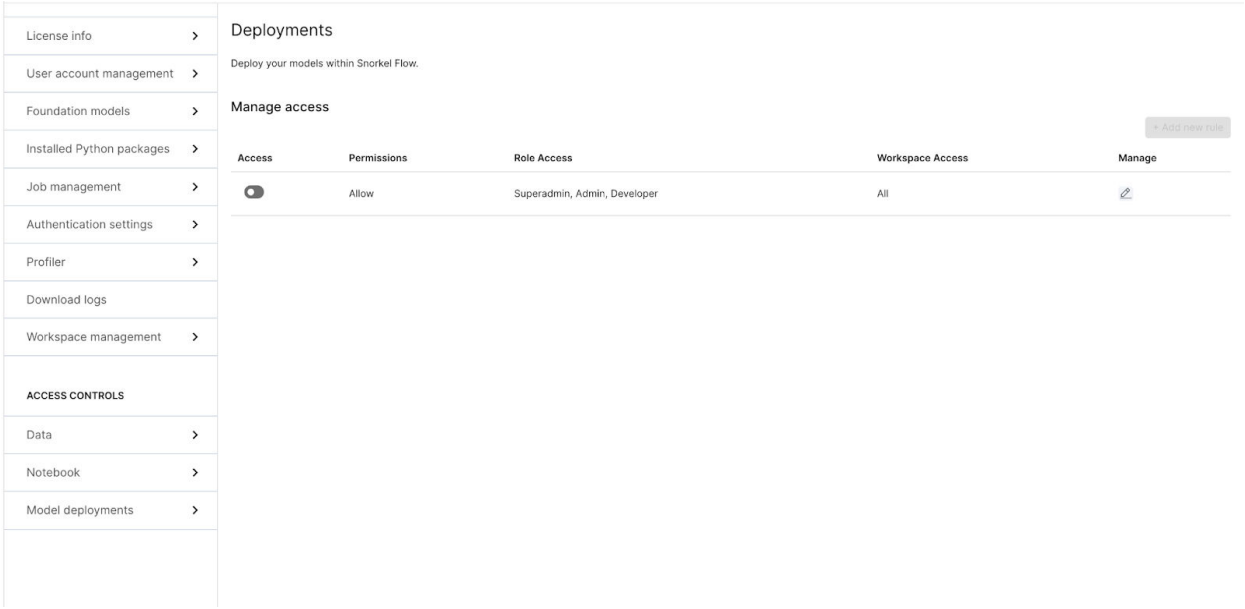
3. Now you can confirm that you wish to disable deployment functionality by selecting the **Disable** button.



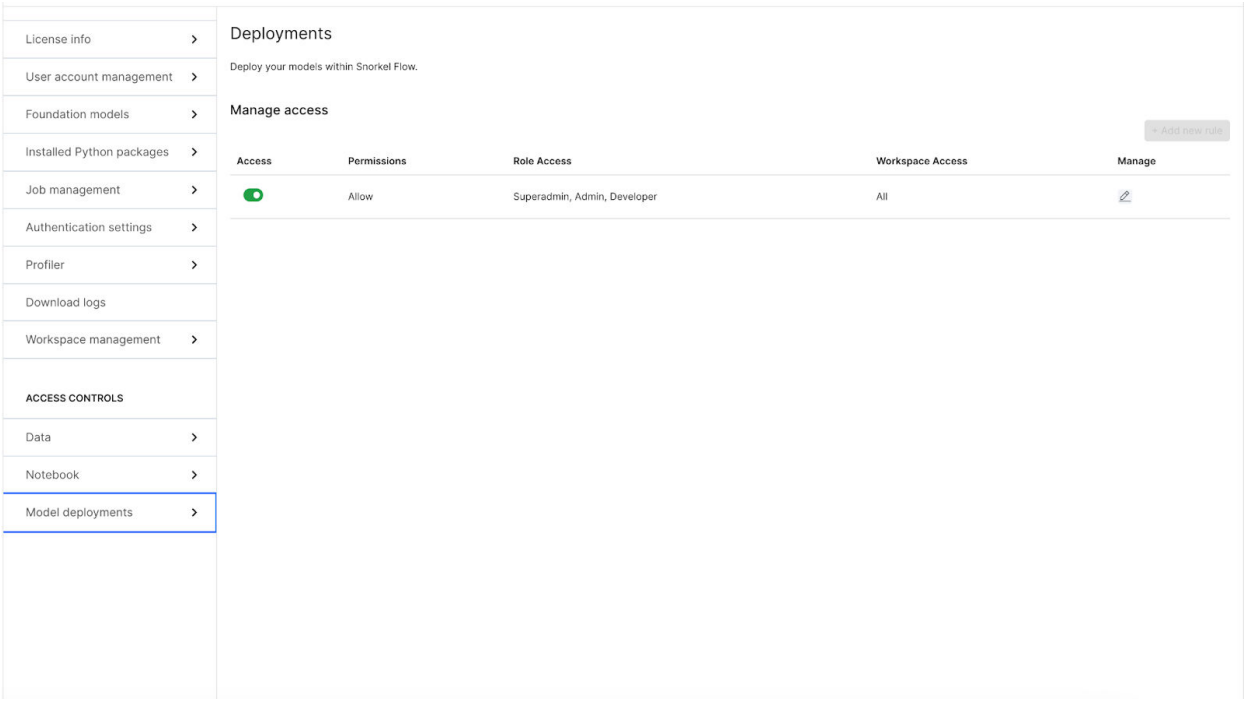
Once disabled, this will be reflected in the UI. You can no longer see the **Deployments** in the sidebar menu.

Enabling a Feature

1. Open the **Admin Settings** page and navigate to the **Deployments** tab under **Access Controls**.



2. Switch the toggle in the **Access** column to enable model deployments for superadmins, admins, and developers.



When enabled, users with the appropriate role specified in **Role Access** will be able to see the **Deployments** tab.

Credential management

Proper credential management is essential for authentication and interaction with 3rd-party services. Integrating with services like OpenAI, HuggingFace, AWS Bedrock, or Snowflake requires users to enter secret keys associated with their accounts. Storing these keys in plain text, however, poses a security risk.

In the Snorkel AI Data Development Platform, **superadmin** users can leverage the SDK to encrypt and store sensitive credentials/keys in the **secret store.** These tools enable superadmin users to register credentials to a specific **reference key** and allow all users to reference that key throughout the platform. Each key is scoped to a workspace.

Set secrets

Superadmins can use sai.set_secret to add secrets to the secret store:

```
import snorkelai.sdk.client as sai

sai.set_secret(key, value, secret_store='local_store', workspace_uid=1)
```

Where:

- **key**: The key to reference the secret in the store.
- **value**: The secret that is being added.
- **secret_store**: The secret store that you can register secrets to. Currently, only the 'local_store' option is supported.
- **workspace_uid**: The workspace that the secret is registered to. The default value is 1.

Delete secrets

Superadmins can use sf.delete_secret to delete the secret that is associated with the key from the secret store:

```
sf.delete_secret(key, secret_store='local_store', workspace_uid=1)
```

List secrets

Superadmins can use sf.list_secrets to list all the keys (i.e., references) for a workspace. This does NOT list the secrets:

```
sf.list_secrets(secret_store='local_store', workspace_uid=1)
```

Data security

This page provides information about how Snorkel stores, encrypts, retains, and transfers customer data. This document applies to Snorkel-hosted instances, which leverage AWS infrastructure. The Snorkel AI Data Development Platform's data encryption approach is similar for all major cloud providers, including AWS, Azure, and GCP.

Data storage

Customer data is stored exclusively on our SOC-2 Type II certified AWS infrastructure. All customer data is isolated from other customer data.

Data encryption

Production data is encrypted in-transit and at-rest. At-rest encryption includes encrypted block storage volumes, NFS volumes, and any object buckets. All services use AES-256 encryption.

Data retention

Data is retained subject to the customer's usage of Snorkel AI products (datasets can be deleted any time if desired), or to meet legal, regulatory, or contractual requirements.

Data transfer

Snorkel can provide a secure bucket pre-signed URL, such as AWS S3, to transfer data securely.

Supported languages

This page provides information about the languages that are supported in the Snorkel AI Data Development Platform.

> NOTE
>
> The Snorkel AI Data Development Platform UI is only available in English. However, we can ingest training data from other languages.

The following languages are supported in the Snorkel AI Data Development Platform:

- Germanic/Roman languages (e.g., English, Spanish, and Italian).
- Beta: Slavic languages (e.g., Russian).
- Beta: Asian languages (e.g., Chinese and Japanese).

Debugging technical issues: How to retrieve job logs and support bundles

This topic explains the options you have to retrieve diagnostic job logs and the support bundle. Workspace Admin permissions or higher are required for each of these tasks.

You might occasionally encounter problems while using the Snorkel AI Data Development Platform that make it difficult to complete your desired workflow. Snorkel makes it easy for you to address these issues without help from Snorkel Support.

1. Workspace Admin can help you download **diagnostic job logs** to identify and understand why jobs are failing.
2. If you still cannot identify and resolve the problem or are having trouble accessing the Snorkel Application altogether, ask your System Admin to send a **support bundle** to Snorkel. This will provide a full system health report to Snorkel, which will enable Snorkel Support to help you address the problem quickly.

Retrieve diagnostic job logs

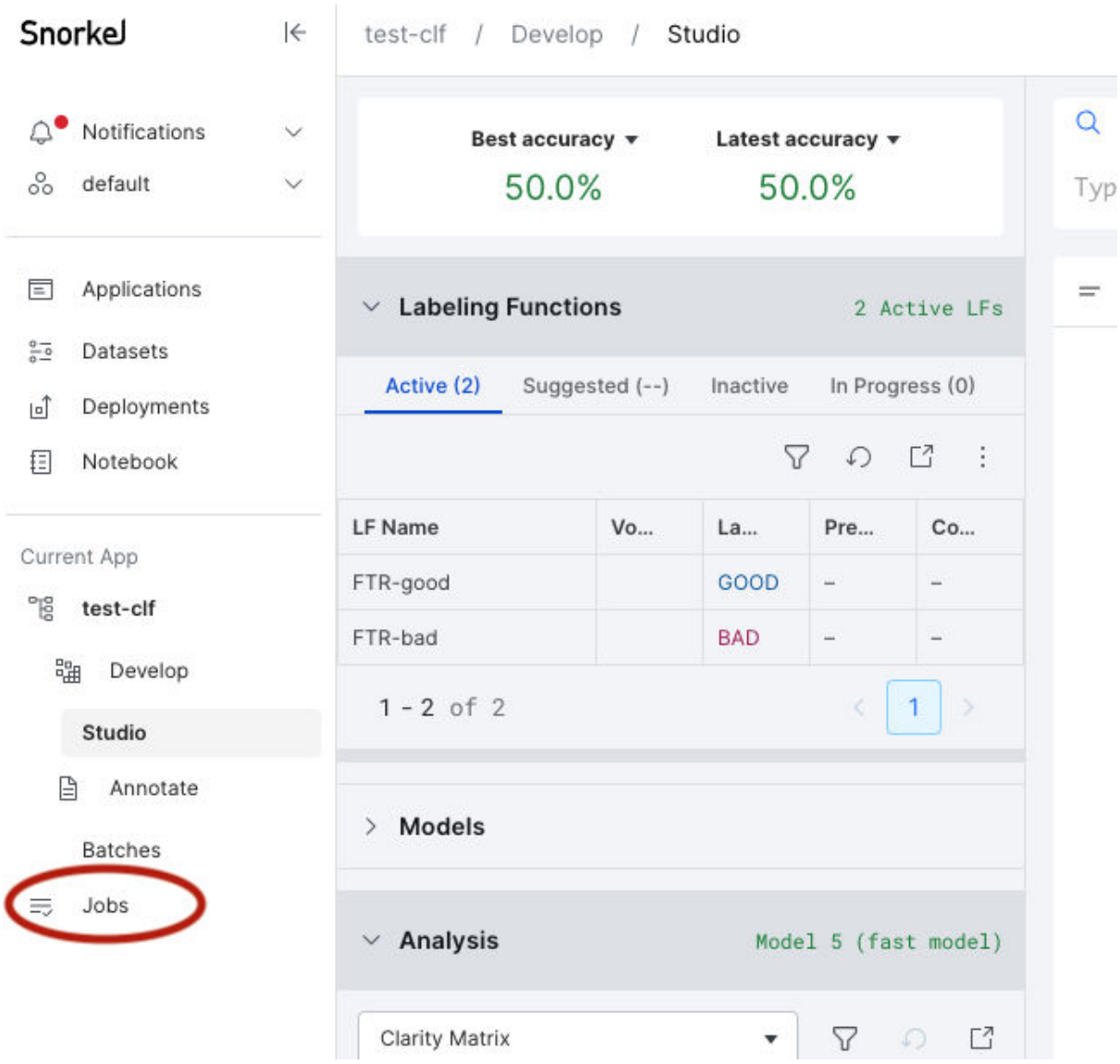
There are several ways Workspace Admins can retrieve diagnostic logs from the Snorkel AI Data Development Platform. These options depend on your level of access to system data. These options are restricted to Workspace Admins or higher. Users without any administrative privileges must request help from a Workspace Admin or System Admin.

For more information on role responsibilities, see Snorkel User roles overview and capabilities

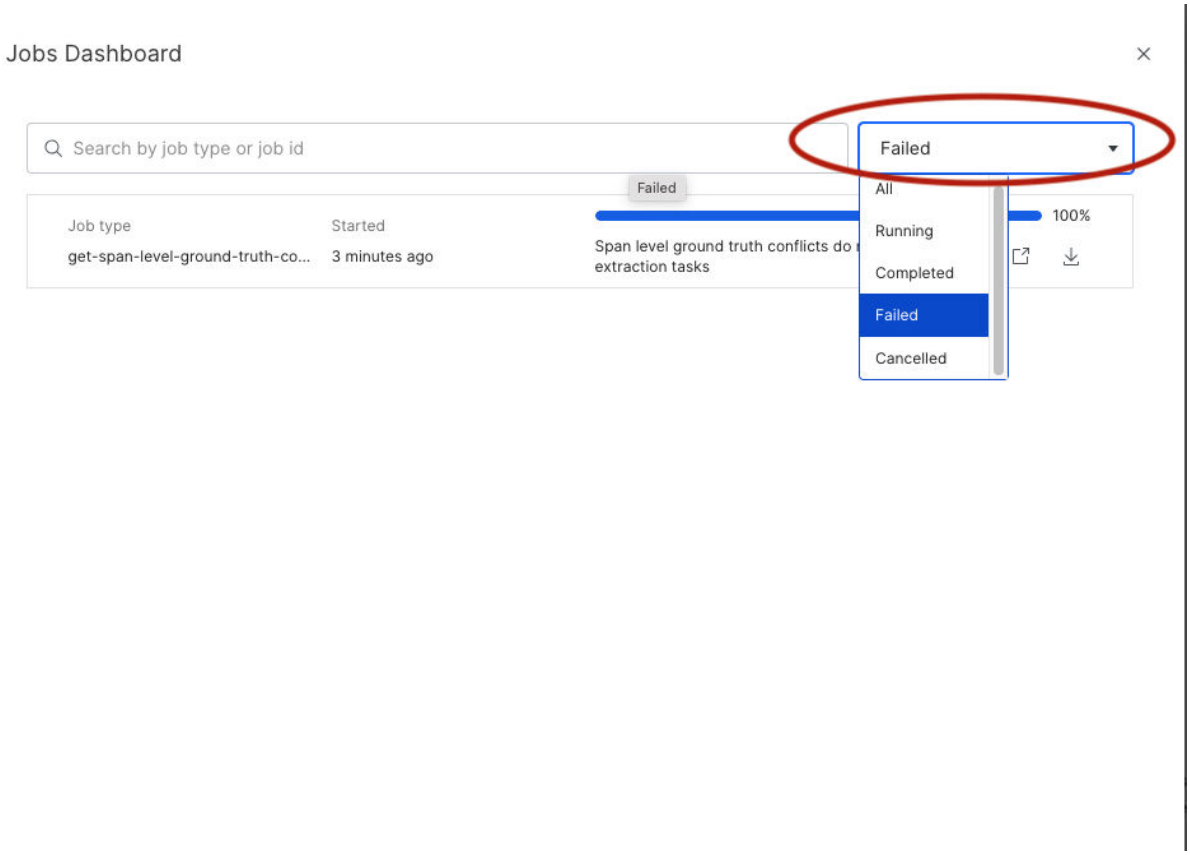
Accessing an Application's Job Dashboard

If you encountered an error while performing an operation in an application, access the **Jobs** in the left-hand menu and collect any relevant error logs there. You must have Workspace Admin privileges or higher to view these logs.

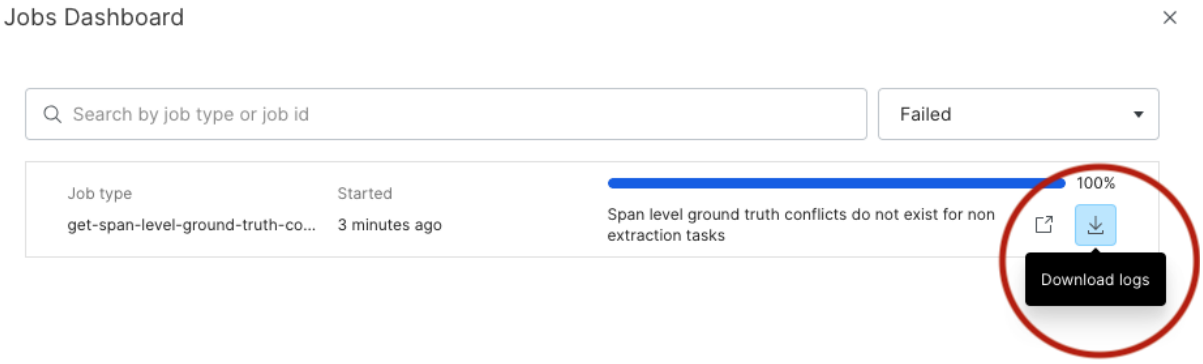
1. Click on the "Jobs" link on the lefthand navigation bar.



2. This will load the Jobs Dashboard that list all jobs that are in progress have completed. Filter the view by which jobs have failed recently. This can be done by using the job status dropdown on the right.



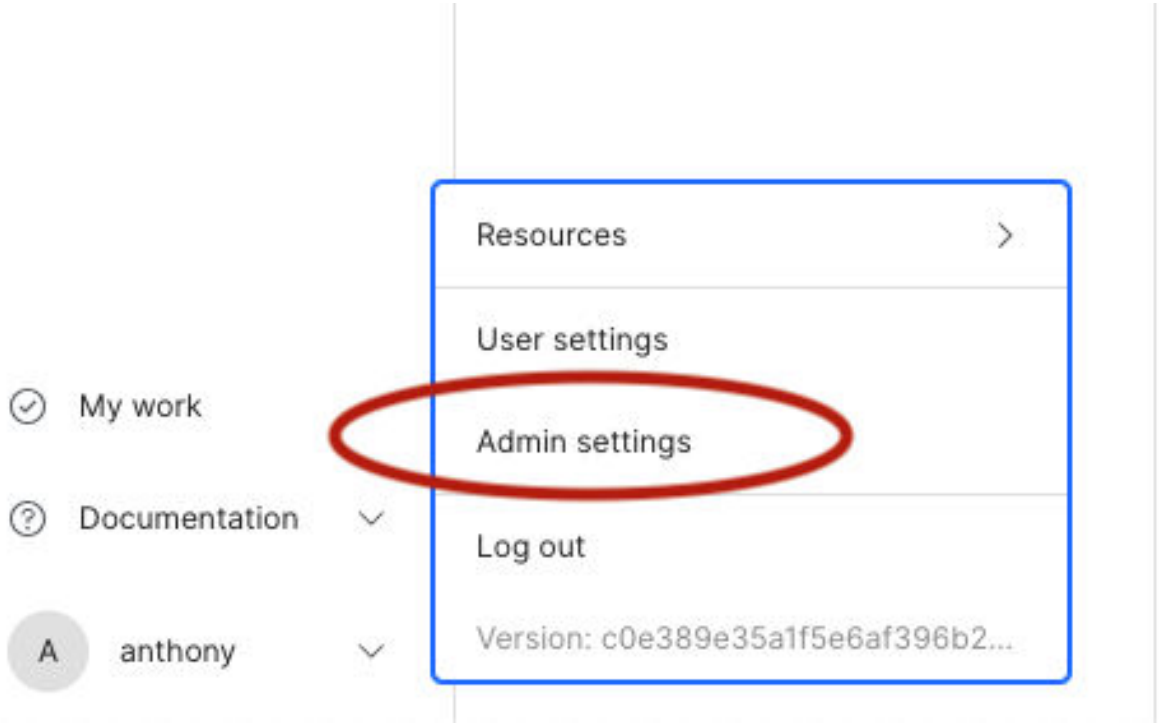
3. If any failed tasks show up here and the "Started" column's time matches when you encountered the error, use the download link associated with the job to retrieve the system diagnostic logs.



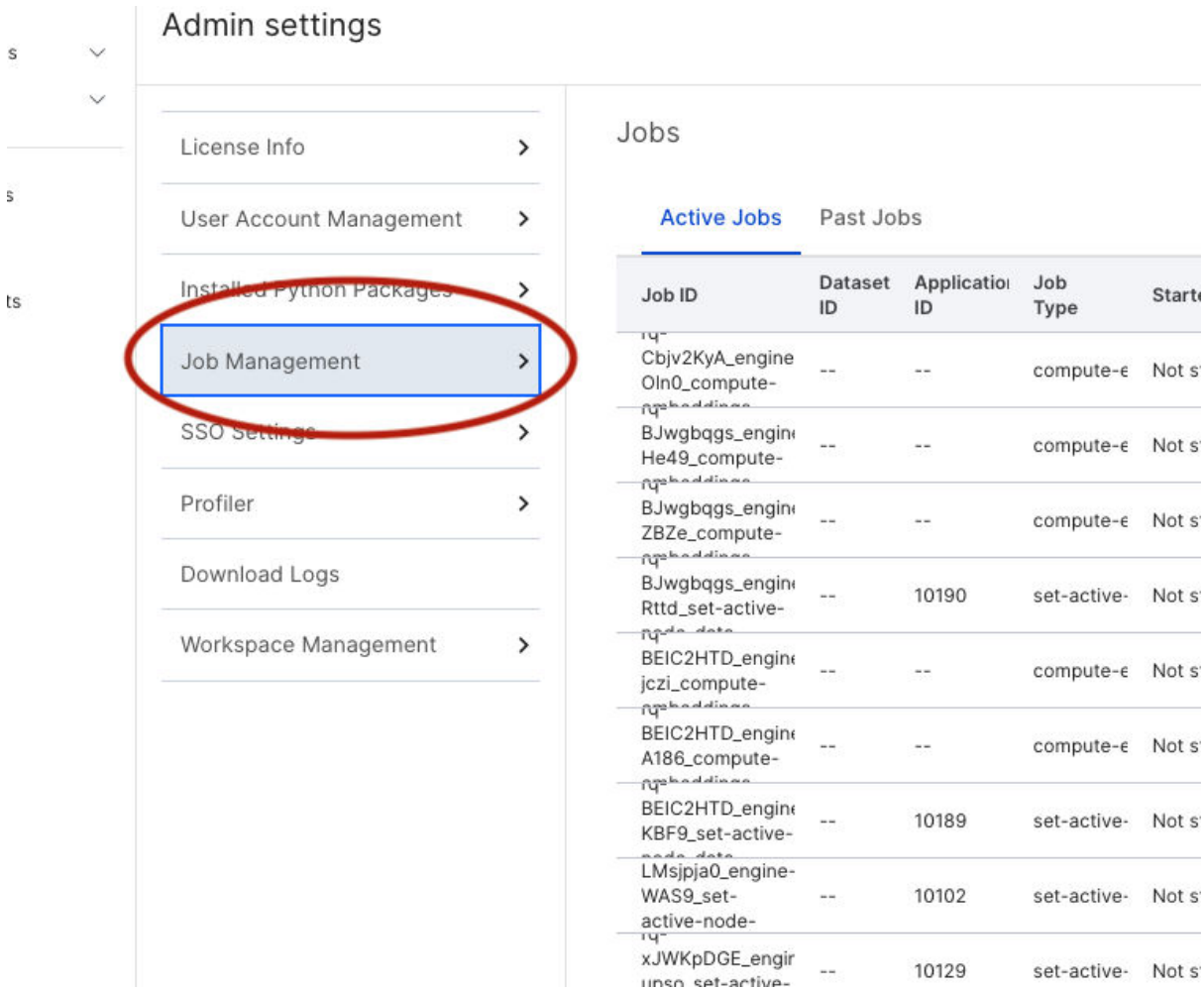
Accessing the Global Job Dashboard

Failed job logs can also be retrieved via the **Administrative Settings** for all applications on the platform. You must have Workspace Admin privileges or higher to access this view.

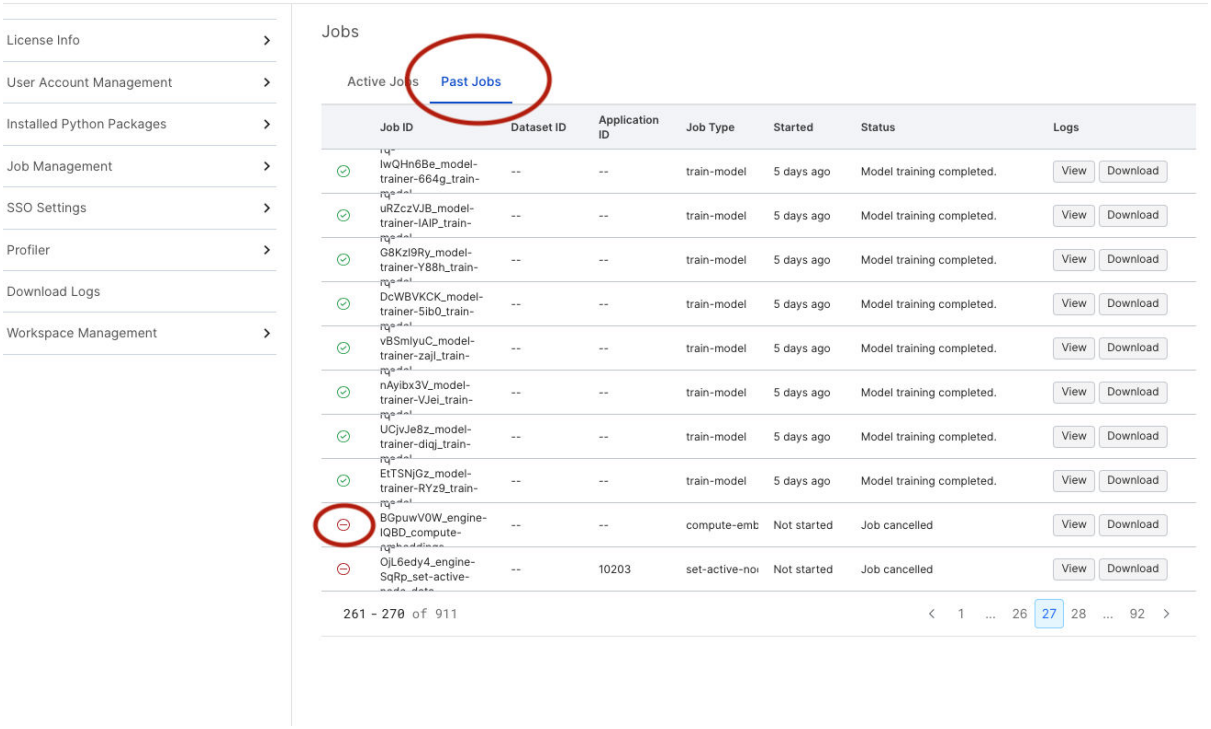
1. Select on your username in the lower left-hand corner, then select **Admin Settings**.



2. In **Admin Settings**, select **Job Management**.



3. Select **Past Jobs** to see all jobs are complete. This page includes past jobs from all applications. You might need to tab through to find the job that presented the error. You can find jobs that had an error by looking for a red **X** on the far left of the job's row.



4. Download as many potentially relevant log files as needed. If you cannot determine which job is responsible for the issue, escalate the issue to your System Admin.

Retrieve a support bundle

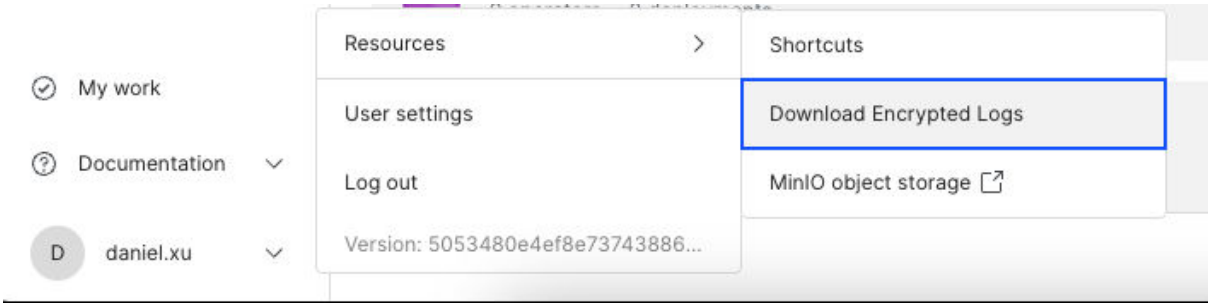
System Admins administer and upgrade the Snorkel AI Data Development Platform. The options below are intended for ONLY Snorkel users with system access.

Export the in-app encrypted logs

This option requires the Snorkel Application to be responsive.

> ⚠ WARNING
>
> The in-app encrypted log export might not function correctly if the platform is experiencing systemwide issues. If you begin the download but there is little to no progress, try one of the alternative approaches to retrieve the support bundle.

1. From the main page, select your username in the lower left-hand corner.
2. Select **Resources** and then Download Encrypted Logs. The downloaded logs contain sensitive data that include work that was executed across the Snorkel AI Data Development Platform. Snorkel encrypts this bundle to prevent exposure.
3. Send the downloaded bundle to Snorkel for decryption and diagnosis.

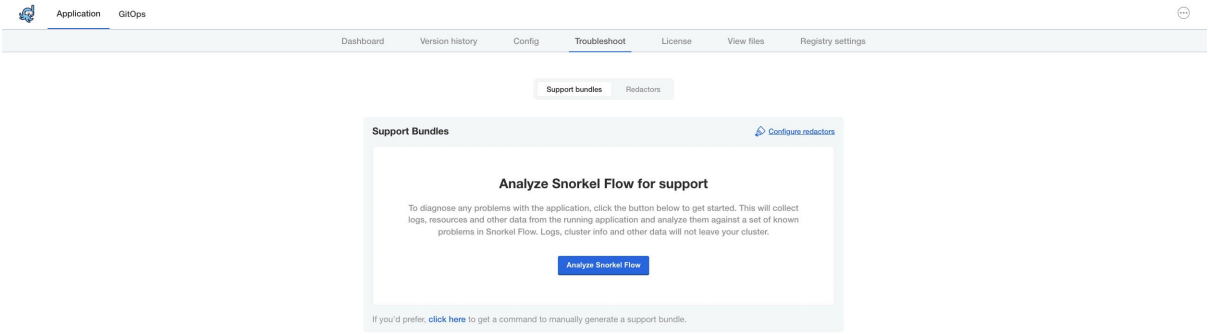


Using the Replicated user interface

This option can be used to extract Support Bundles while the application is down.

For Snorkel customers using Replicated to manage their Snorkel AI Data Development Platform, System Admins can download the support bundle.

1. In the Replicated admin console, select **Troubleshoot** in the navigation bar.



2. Select **Analyze Snorkel** to create a support bundle that can be sent to Snorkel directly. System Admins can customize the **SupportBundle** via the SupportBundle custom object. For more details, contact Snorkel Support.

Use script to gather support bundle

This option can be used to extract Support Bundles while the application is down.

A System Admin can use the `python` script gather_support_bundle.py to generate a support bundle.

From a workstation with cluster access, start the script:

```
NAMESPACE="snorkelflow"
python3 gather_support_bundle.py --namespace ${NAMESPACE}
```

This script creates a collection of log files in an archive, as well as some Kubernetes documents if relevant to the installation. These can be sent to Snorkel Support directly.

Manually collecting log files

This option can be used to extract Support Bundles while the application is down.

The Snorkel AI Data Development Platform stores all log files for all jobs and applications in a well-known `\.logs\` path in the primary data volume mount.

1. Compress the `\.logs\` directory as an archive.
2. Send the archive to Snorkel Support to diagnose the issues.

Audit Logging in Snorkel

This documentation provides an overview of audit logging in the Snorkel AI Data Development Platform, including what it encompasses, the parts of the platform covered, and how to access audit logs.

Overview

Audit logging in the Snorkel AI Data Development Platform is designed to track and record critical actions and events performed within the platform. The audit logs capture essential details about each activity to provide comprehensive visibility and accountability. Audit logs support security, compliance, and operational monitoring.

Prerequisites

To access and review audit logs, you need the following prerequisites:

- An existing Snorkel AI Data Development Platform with access to a notebook server.
- Administrative access to the Snorkel AI Data Development Platform.

What are Audit Logs?

Audit logs provide a way to see which users have performed what actions within the platform. Specific actions and their metadata are logged in the platform, capturing:

- **What activity was performed?** - The specific operation or action executed (e.g., user creation, dataset deletion).
- **Who or what performed the activity?** - The identity of the user or service responsible for executing the action.
- **Where or on what system the activity was performed from?** - The source system or IP address from which the action originated (partially captured).
- **What was the activity performed on?** - The target entity or object affected by the action (e.g., dataset, application, user).
- **When was the activity performed?** - The timestamp indicating when the action occurred.
- **What was the status, outcome, or result of the activity?** - The result or status of the action, such as success or failure (partially captured).

What Do Audit Logs Encompass?

Audit logs in the Snorkel AI Data Development Platform cover a broad range of actions across various components of the platform, including user management, data management, and application management. The following operations are currently audited:

Access Management

- **Access Attempts**: Tracks user access to the system, including successful and failed attempts.
- **API Authorization Failure**: Records details of failed API authorization attempts.
- **API Key Generation**: Audits creation and regeneration of API keys.

User Management

- **User Create/Update/Delete**: Audits actions involving user accounts, including creation, modification, and deletion.
- **Invite - Create**: Tracks the creation of invitations for new users.
- **Role - CRUD**: Logs create, read, update, and delete operations on user roles.
- **RBAC Create/Update/Delete**: Audits actions related to Role-Based Access Control (RBAC) changes.
- **Password Changes**: Tracks all user password change activities.

Data Management

- **Datasource Create/Update/Delete**: Logs operations involving data source management, such as adding, updating, or deleting data sources (only through the UI).
- **Dataset - Delete**: Tracks deletion of datasets.
- **Static Asset Upload**: Logs uploads of static assets.
- **File Operations**: Audits the start of data creation operations, including file uploads to Minio and downloads from remote locations.

Application Management

- **Application Create/Update/Delete**: Records actions related to the creation, updating, or deletion of applications.
- **Workspace Create/Update/Delete**: Logs operations involving workspaces, such as their creation, updating, or deletion.

Python Environment Management

- **Custom Pip Install/Wheel Install/Reset**: Audits all changes made to the Python environment, including custom package installations and resets.

Snorkel Actions

- **Annotation Create/Update/Delete/Bulk Delete/Import**: Logs all actions related to annotation management, including bulk operations and imports.
- **Annotation Transfer**: Records details of transferring annotations between different entities or users.
- **Annotation Commit**: Logs the committing of annotations to the system.
- **Batch Creation**: Audits the creation of data batches.
- **Label Schema Create/Delete**: Logs operations involving the creation or deletion of label schemas.
- **Training Set Create**: Tracks the creation of training sets.
- **External LLM Configurations**: Audits operations involving configuration changes for external language models.
- **Label Function (LF) CRUD Operations**: Logs create, read, update, and delete actions on label functions.

Super Admin Actions

All actions performed by super admin users are fully audited, including:

- Differentiation between actions performed through the user interface (UI) versus those executed through the software development kit (SDK).

How to Access Audit Logs

You can access audit logs via the `/audit-logs` endpoint using the Snorkel SDK. This can be executed inside a notebook server if you have SuperAdmin privileges. Here's a sample code snippet:

```
# Inside Snorkel notebook server
import snorkelai.sdk.client as sai

# Configure client context for Snorkel AI Data Development Platform
ctx = sai.SnorkelSDKContext.from_endpoint_url()
resp = ctx.tdm_client.get('/audit-logs?limit=100')

print(resp)
```

This will return a JSON formatted list of values. An example of this is show below.

```
[
  {
    "event_id": 1,
    "event_time": "2021-08-17T20:51:47.268843",
    "event_name": "api_key",
    "event_type": "create",
    "event_details": {
      "censored_key":
"************************************************************E4YO"
    },
    "user_uid": 2
  }
]
```

The meaning of these fields is as follows:

- **event_id**: A unique ID of each event that has occurred on the platform.
- **event_time**: The time the event took place.
- **event_name**: The name of the type of event. Usually this points to the resource it's related to (e.g. lf, application, workspace, user)
- **event_type**: The type of event that has occurred. (e.g. create/read/update/delete).
- **event_details**: A JSON object that is unique for each event type. Information in this field is only relevant for the specific event_name, and event_type. (e.g. for a **login** event, you will have `{'user_uid': 22, 'username': 'john_wick'}`)
- **user_uid**: The uid of the user who took the action.

Endpoint: Get Audit Logs

This endpoint allows you to fetch a list of audit logs from the system, which can be used for tracking and monitoring events related to the application's activities.

URL

GET `YOUR_SNORKEL_INSTANCE_URL/audit-logs`

Headers

- **Authorization**: This header must contain the API key for authorization. For example, `"Authorization: key API_KEY"`.
- **Content-Type**: This should be set to `application/json`.

Query Parameters

- **limit** (optional, integer): The maximum number of audit log events to retrieve.
  - Default: 200
  - Maximum: 1000
  - Minimum: 1
- **last_id** (optional, integer): The ID of the last audit log event from a previous request. Used for pagination to fetch the next set of audit logs.

Request Example

```
curl -L -H "Authorization: key API_KEY" -H "Content-Type: application/json"
"https://edge-tdm-api.k8s.g498.io/audit-logs?limit=100&last_id=50"
```

Response

The response is a JSON object containing:

- **events** (array of AuditEvent objects): A list of audit events.
- **last_id** (integer): The ID of the last audit log event retrieved, which can be used for pagination in subsequent requests.

Secret storage and encryption

NOTE: This topic applies to Snorkel AI Data Development Platform 0.91.24 and later.

Snorkel AI Data Development Platform sometimes requires storing client credentials or other secret data. For example, Snorkel AI Data Development Platform might need to store an OIDC client secret or a third-party API key on the platform. Snorkel stores these secrets with the following configuration:

- **Storage:** Secrets are stored in the Snorkel Postgres database.
- **Encryption:** Snorkel encrypts these secrets before storing them in the Postgres database. Snorkel uses AES-256 encryption with randomized initialization vector (IV) and an encryption key.
- **Key:** Snorkel recommends providing a user-created encryption key or relies on a generated key if no user key is present.

Best practice: User-created encryption key

If your installation of the Snorkel AI Data Development Platform is running on Kubernetes, create your own encryption key in the project's namespace.

Your Snorkel installation stores this key separately from the Snorkel Postgres database, using the Kubernetes security primitive `Secrets` object.

The advantage of this approach is that if someone gains access to the database or its backups, they cannot decrypt the stored secrets. The Snorkel AI Data Development Platform does not mount keys stored this way into the Kubernetes environment. Instead, the Snorkel AI Data Development Platform calls the Kubernetes API for encryption and decryption when needed.

WARNING: Do not rotate this key after it is set. If you change this value, you risk losing all encrypted data.

The secret looks like this:

```
apiVersion: v1
data:
  root-key: <YOUR_ENCODED_KEY>
kind: Secret
metadata:
  name: root-key-secret
  namespace: <YOUR_NAME_SPACE>
type: Opaque
```

These are the key parameters in this Kubernetes `.yaml` file:

- `root-key`: The base64-encoded key.
- `name`: Unique name for this secret.
- `namespace`: Shared namespace for the Snorkel project.

For added security, use the secret manager and Kubernetes secrets integration provided by your cloud provider:

- For GCP, use the Secret Manager add-on with Google Kubernetes Engine.
- For AWS, use the AWS Secrets Manager secrets with Kubernetes.

Default: Generated encryption key

If no user-provided key is available or the installation is not running on Kubernetes, the Snorkel AI Data Development Platform generates a default key for each Snorkel AI Data Development Platform.

The Snorkel AI Data Development Platform generates this key deterministically, using the Snorkel Key Component, which is known only to Snorkel AI, and a Customer Key Component, which Snorkel generates securely for each installation.

However, Snorkel highly recommends that you provide their own encryption key to ensure the keys can be decrypted only with access to the database contents and the secret store.

User and Admin settings

Browser support

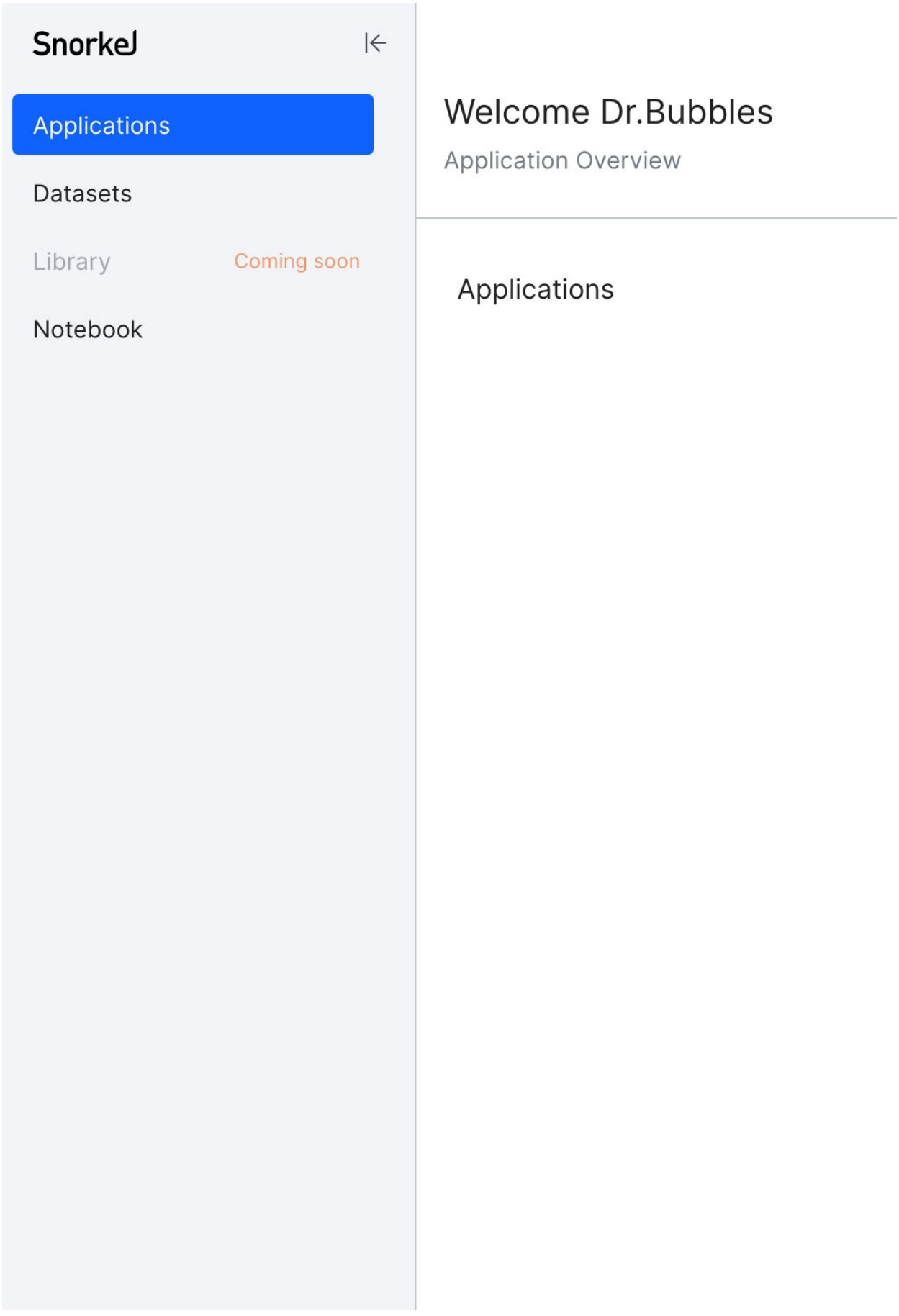
The Snorkel AI Data Development Platform currently supports the following browser configurations:

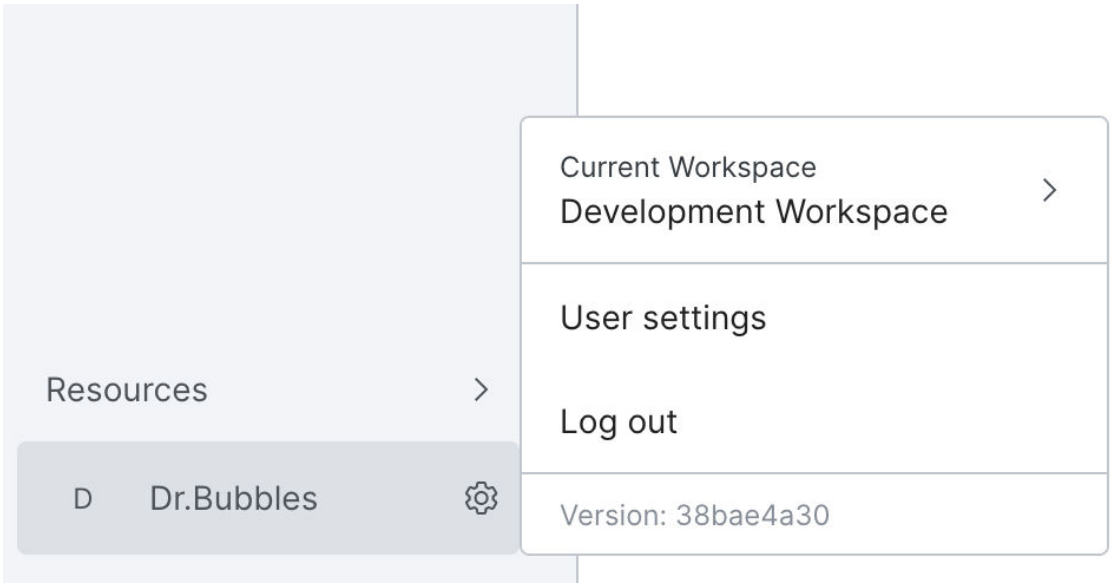
- MacOS
  - Google Chrome: version *109* or later
- Windows
  - Google Chrome: version *109* or later

Mobile devices are not recommended for accessing the Snorkel AI Data Development Platform.

User settings

User settings are accessible from the user menu at the bottom of the sidebar menu, under the heading User Settings. All users of the Snorkel AI Data Development Platform are able to access the settings page.



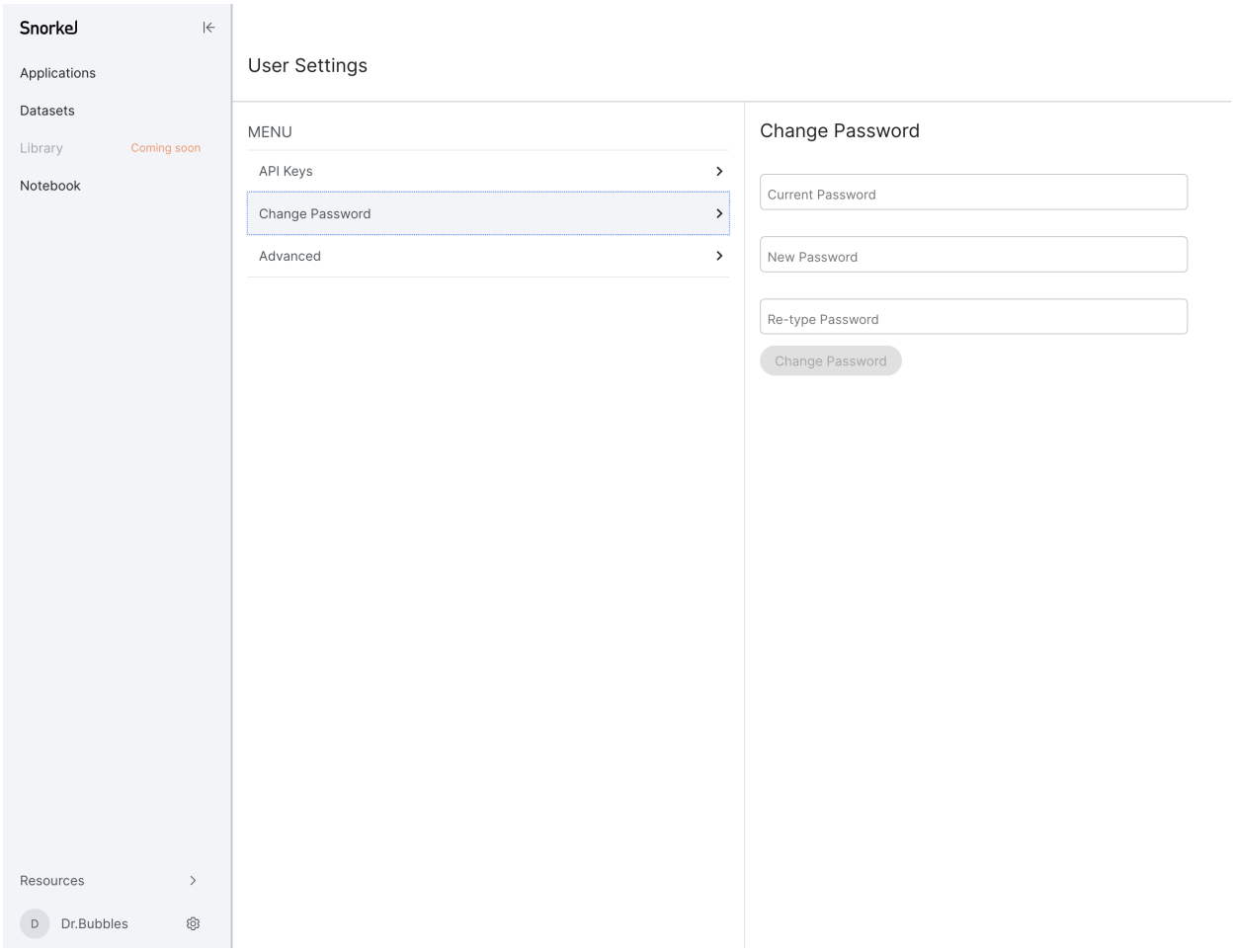


Password settings

The **User Settings** page allows you to change your password, as long as you know your current password.

Warning

If you've forgotten your password, please have an admin use the **reset_password** SDK function to do so.



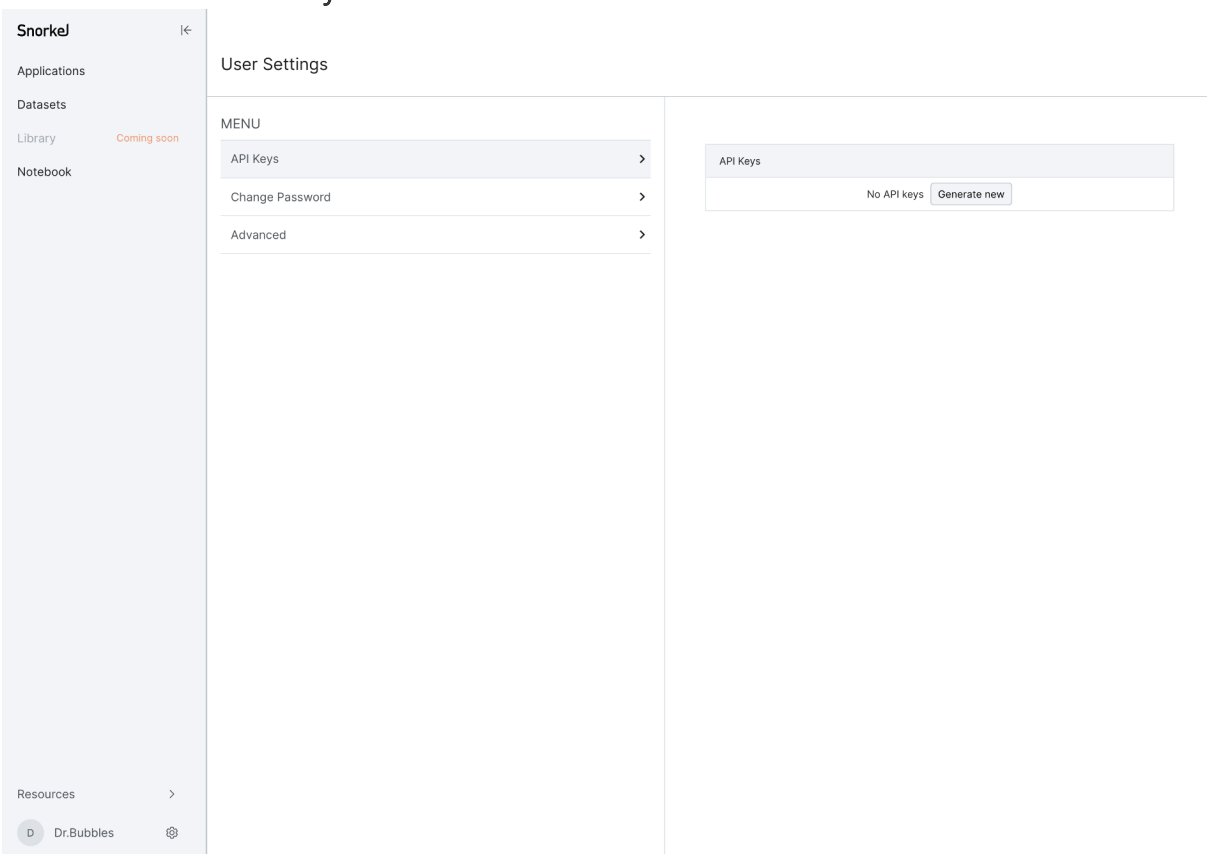
Using API keys

The **API Keys** page allows you to generate, view, and revoke API Keys. API Keys can act as a replacement for your username and password when using the SDK or API.

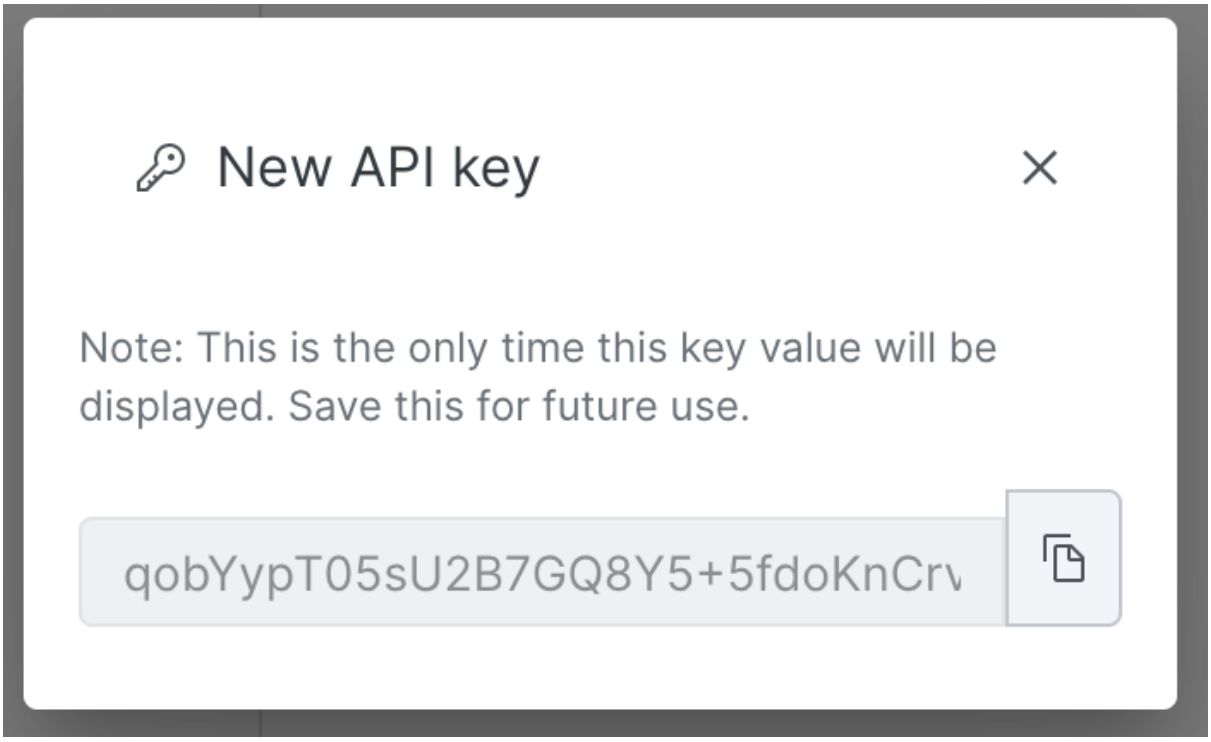
> **NOTE**
>
> API Keys should be handled with the same care as a username and password.

Generating and using API keys

1. Select your username from the bottom left navigation.
2. Select **User settings**.
3. Select **API Keys** from the **User Settings** menu.
4. Select **Generate new API key**.



5. Copy your new API key. Your API Key will only be shown in full once.

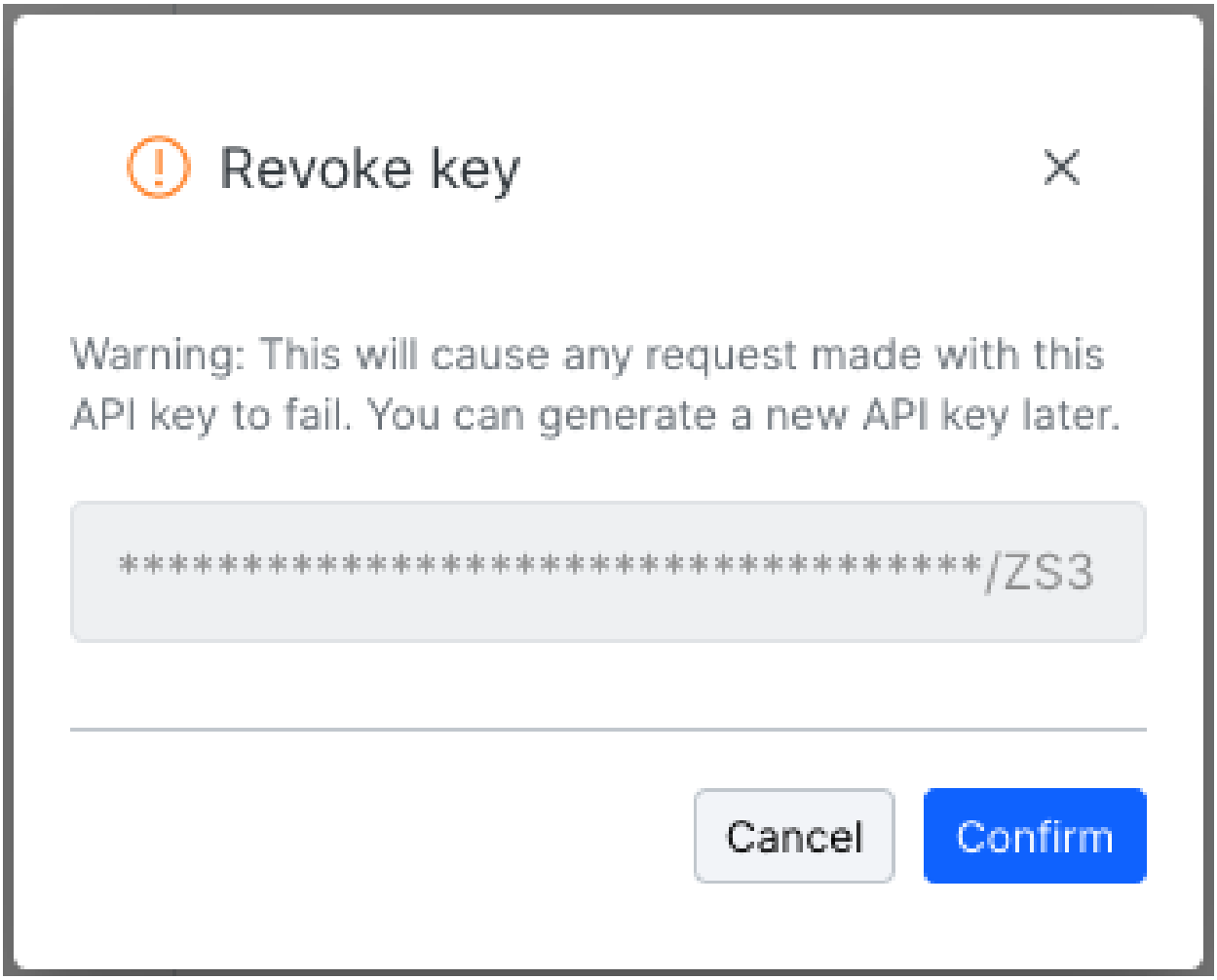


Currently only a single API key can be active per user.

Revoking an API key

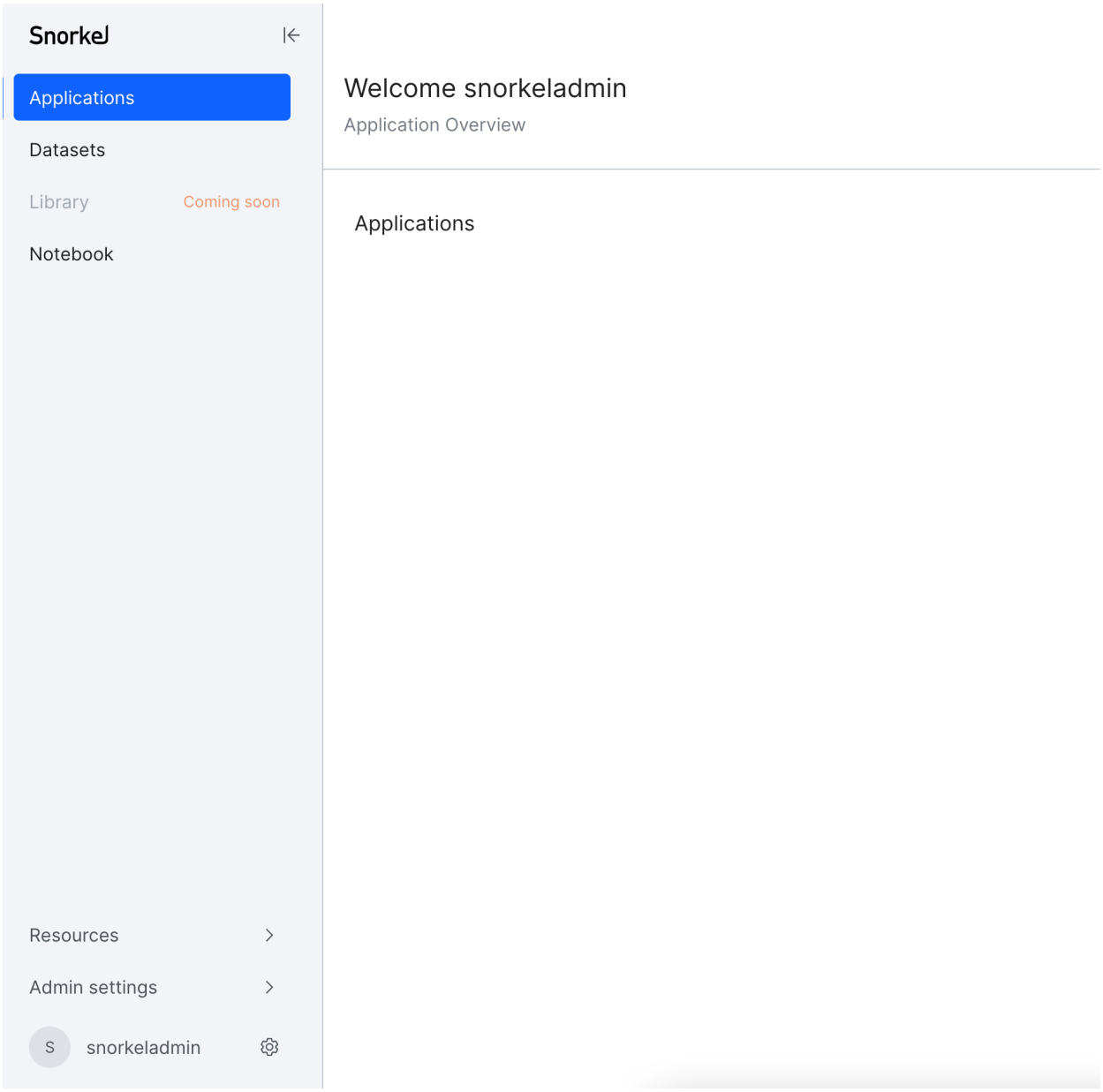
To revoke an API Key, click the **Revoke** button next to the key.

A sample of the last few characters of the key will be displayed to confirm it is the correct key. Click the **Confirm** button to revoke the key.



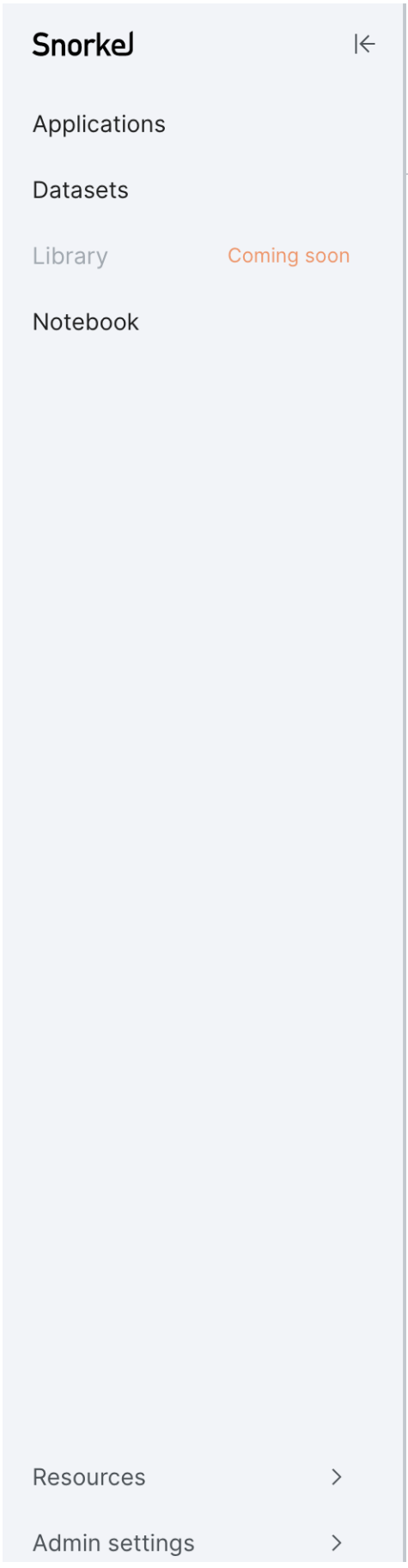
Access Admin settings

The Admin Settings page is accessible from the bottom of the sidebar menu, under the heading **Admin Settings**. Only admin users of the Snorkel AI Data Development Platform are able to access Admin Settings.



Admin settings menu

The **Admin Settings** menu gives you access to show and update licensing information, manage user accounts, manage installed python packages, manage running jobs, and manage sso settings.

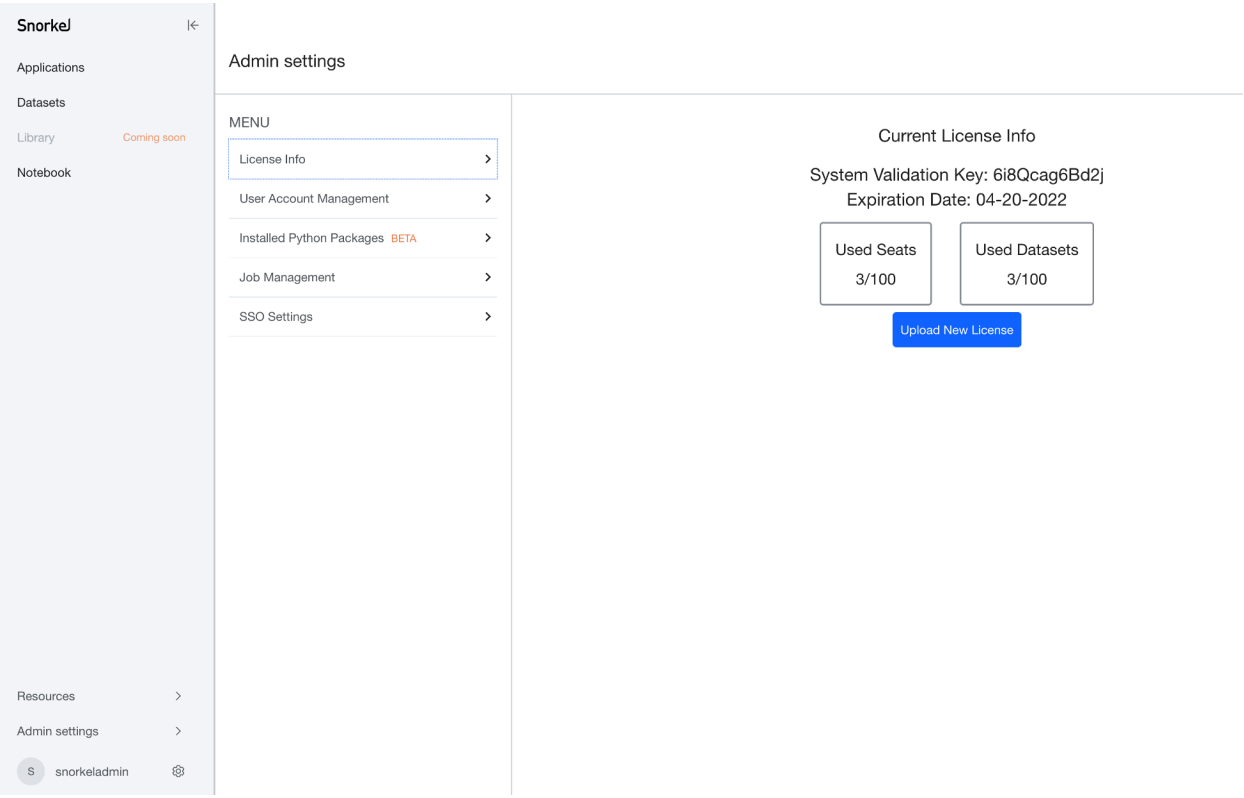


View and update licensing Information

The **License Info** section of the Admin Settings provides information on the current license key, along with the option to upload a new license key.

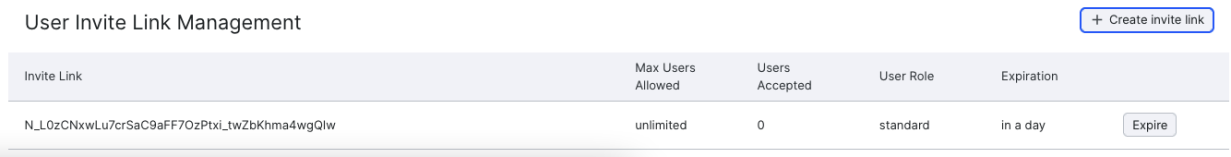
Create and deactivate users

You can create and deactivate user accounts via the User Account Management section of Admin Settings. User accounts can be created by an admin manually, or by generating an invite link. See Using Invite Links for more info.

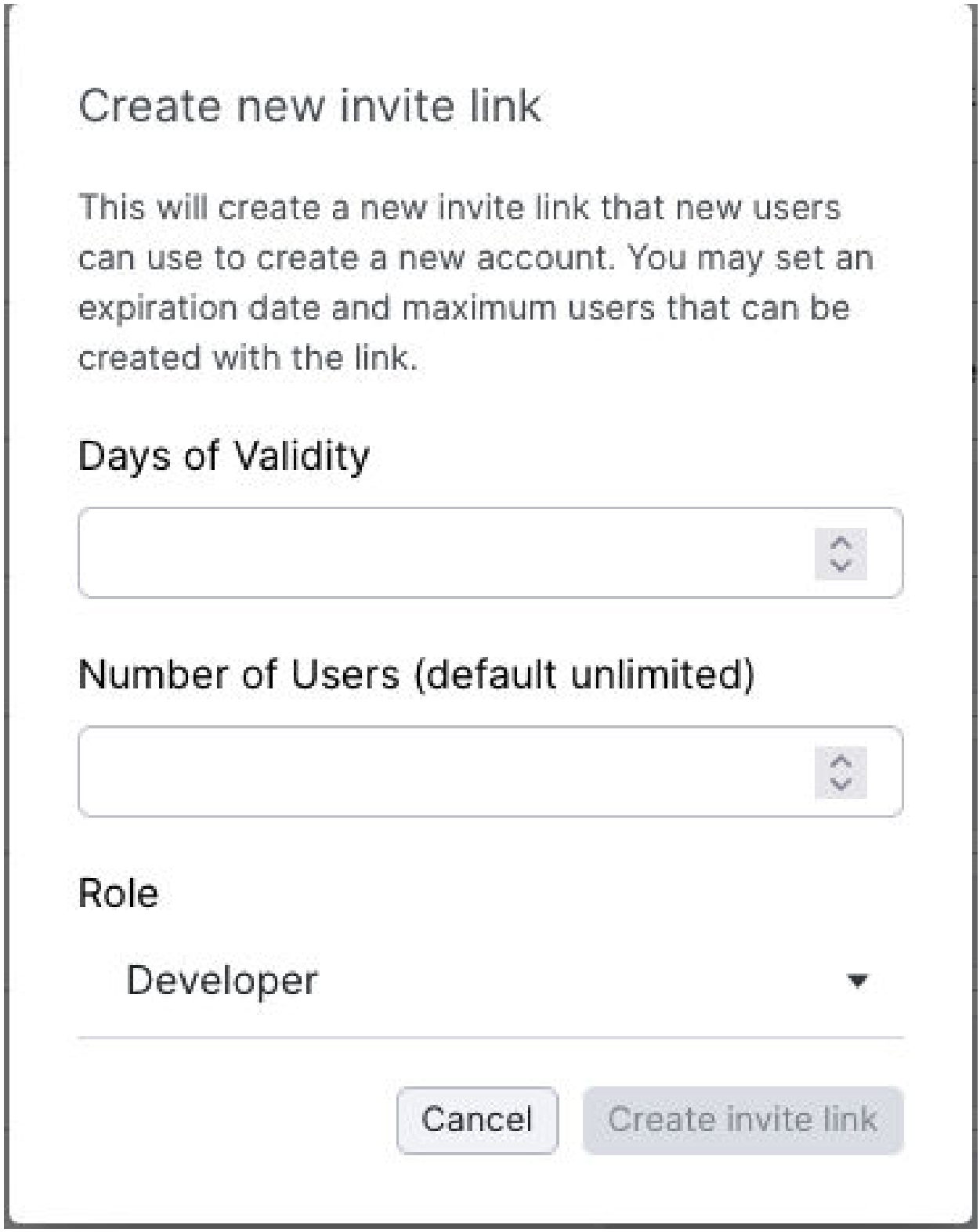


Using invite links

Invite links allow one or more users to register themselves into the Snorkel AI Data Development Platform. An Admin creates these by clicking the create invite link located next to the User Invite Link Management table.



This will show a modal with the following options: * the number of days the link is valid for * the maximum number of users that are allowed to use the link * the role these users will be assigned



Once a link has been created, it should be given to user(s) to allow them to create their own account. A generated invite link can be expired manually by clicking the expire button next to the corresponding invite.

Resetting another user's password

As an admin, you can reset a users password, by using the SDK function **reset_password**.

Reset_password lets an admin set a users password to a new password, allowing a user to access their account if they have forgotten their password.

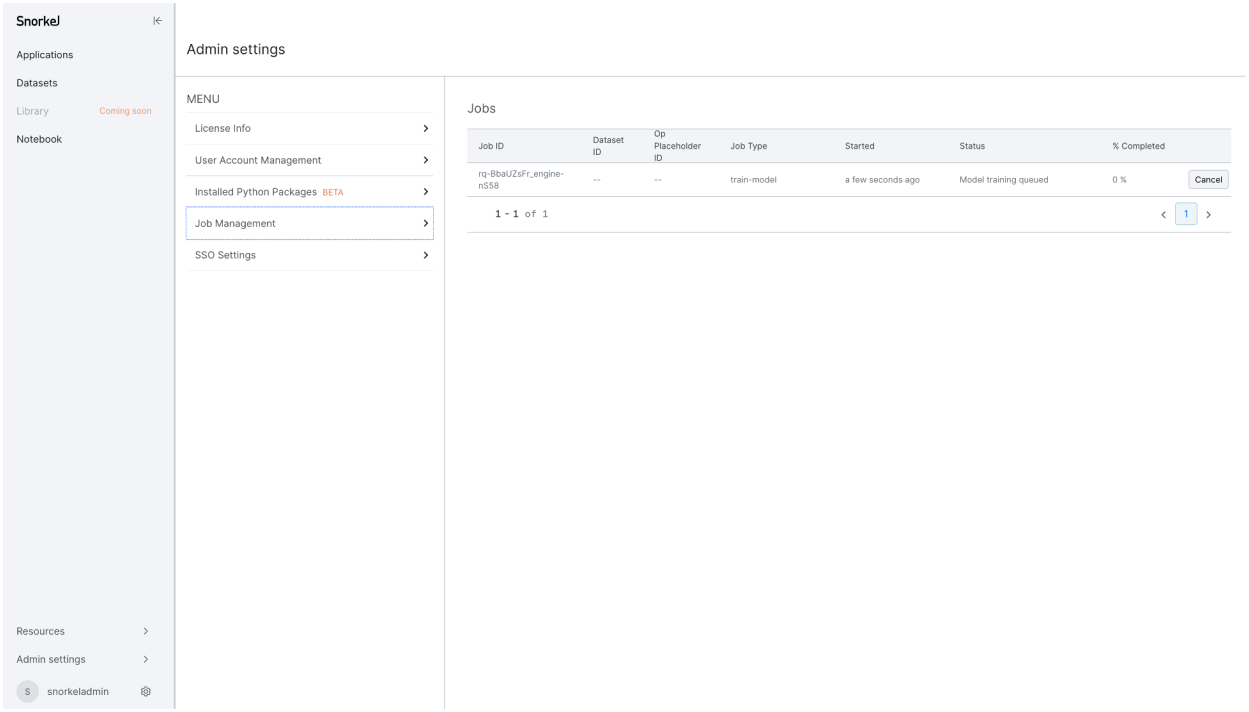
As this method is restricted to admin usage only, one will need to supply an API key belonging to an admin account in order to successfully reset a password.

```
reset_password(username="<username here>", password="<new password here>")
```

Managing running jobs

The Job Management section of Admin Settings lets administrators view current jobs running in the Snorkel AI Data Development Platform, and cancel them. Jobs that are running will show the Job ID, along with metadata about the running job including when it was started, it's current status, and it's percentage completed. You can also view logs of each job by clicking the **Logs** button.

To cancel a running job, click the **Cancel** button the right side next to the job you wish to cancel.



Authentication settings

In **Authentication Settings**, admins can customize the session time-out. If a session is inactive, users are forced to re-authenticate after the specified length of time, which can be 15 minutes, 30 minutes, or 1 hour. The default inactivity setting is one hour.

Per the NIST requirement for AAL3, all users are forced to re-authenticate after 12 hours. This includes token authentication.

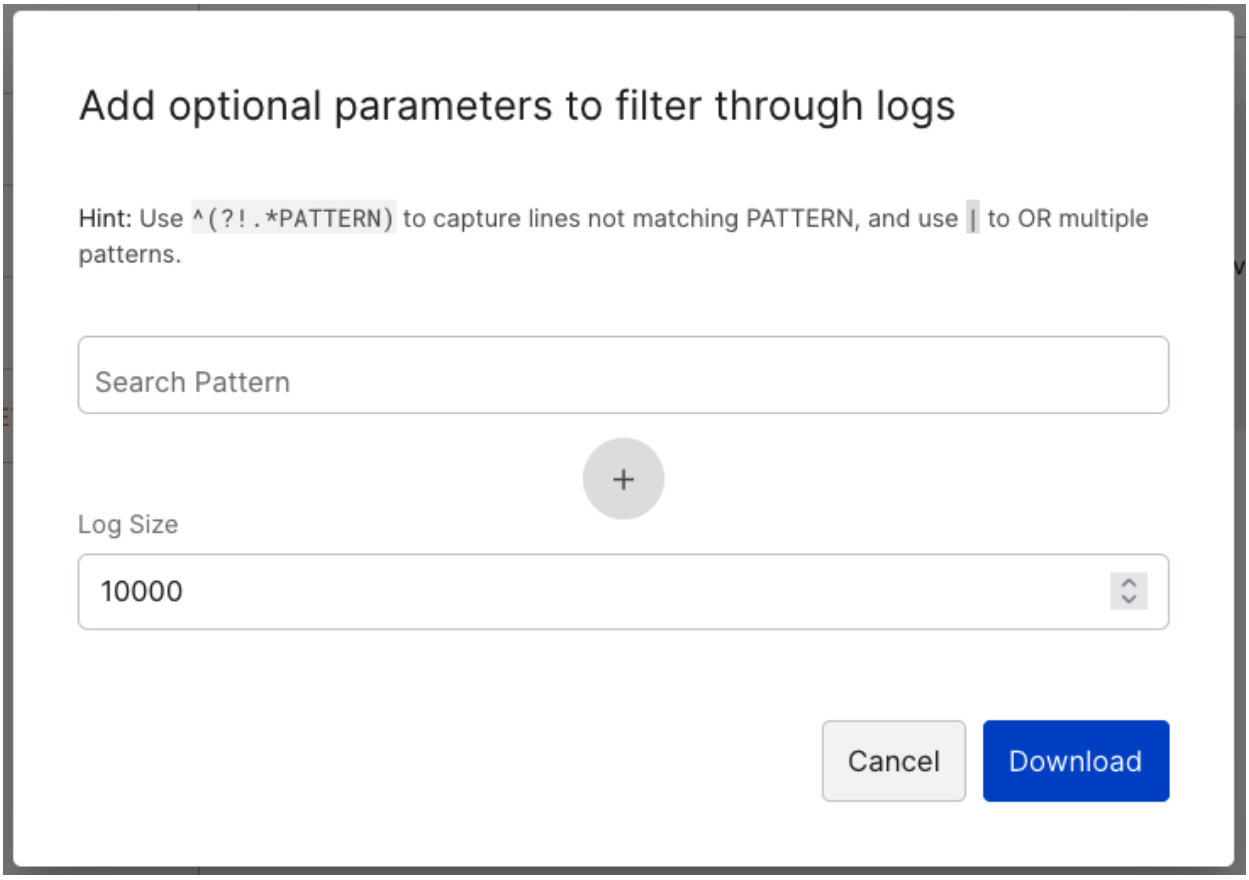
SSO settings

For setting up and managing Single Sign-on settings, you can either use SAML or OpenID Connect.

- For SAML setup, see our SAML SSO setup guide.
- For OpenID Connect, see our OIDC SSO setup guide.

Download logs

The **Download Logs** section of **Admin Settings** allows administrators to download logs from the platform. These logs allow debugging of issues that may occur during platform usage.



The two options for downloading logs are the Search Pattern and the Log Size.

Search Pattern: A regular expression of what to capture when searching through the logs. If nothing is specified, all logs will be captured.

> NOTE
>
> Using very complicated regular expressions can slow down the process of searching through logs. It is recommended to start with a simple regular expression, such as a single word match, and add complexity over time.

Log Size: The number of lines to capture for each Snorkel service that is running.