data.all - User Guide

v1.3.0

None

None

Table of contents

1. Introduction

This section defines what is data.all, what is the challenge that it is trying to overcome and the value that it can bring to your teams.

1.1 What is data.all?

A modern data workspace that makes collaboration among diverse users (like business, analysts and engineers) easier, increasing efficiency and agility in data projects ✨

1.2 Why did we built data.all?

Data teams can be diverse: analysts, scientists, engineers, business users. Diverse people, with diverse tools and skillsets — diverse "DNAs". All leading to chaos and resulting in titanic efforts spent in **Collaboration Overhead**.

Using data.all, any line of business within an organization can create their own isolated data lake, produce, consume and share data within and across business units, worldwide. By simplifying data discovery, data access management while letting more builders use AWS vast portfolio of data and analytics services, data.all helps more data teams discover relevant data and let them use the power of the AWS cloud to create data driven applications faster.

1.3 How can data.all help data teams?

Teams can easily DISCOVER AND UNDERSTAND data 💫

data.all makes all your datasets easily discoverable! No more Slack messages saying "Where's that dataset?" or long email threads for approvals. With data.all, you can simply browse the data catalog.

Key Capabilities: Discovery and Search, Data Preview & Worksheets and Notebooks

Teams can easily SHARE AND COLLABORATE with data 🤝

Data practitioners spend 30-50% of their time finding and understanding data. data.all cuts that time by 95%. Your data team will be shipping 2-3 times more projects in no time.

Key Capabilities: Data Profiling & Data Sharing and Subscriptions

Teams don't have to worry about SECURING their data 🛡️

Don't lose sleep trying to figure out if your sensitive data is secure. Build ecosystems of trust, make your team happy, and let data.all manage governance and security behind the scenes.

Key Capabilities: Granular Access Control

2. Main components

This section introduces the main components of data.all which are divided in 3 groups. This is an overview, for more details please refer to their specific sections.

- **Administrate**: used by team and data lake administrators to organise and manage teams and users inside data.all
- **Discover**: used by all users to contribute with data, search for data and share data.
- **Play**: once data is in data.all, all users can use these tools to work with data.

2.1 Administrate

2.1.1 Organizations

Organizations are high level constructs where business units can collaborate across different AWS accounts at once. An organization includes environments (see below). Organizations are abstractions, they **don't** contain AWS resources, consequently there is no CloudFormation stack associated with them.

Organizations usually correspond to whole organizations, organization divisions or a separated geographical region within an organization.

2.1.2 Environments

An environment is a workplace where a team can bring, process, analyze data and build data driven applications. This workspace is mapped to an AWS account in one region. It is possible to have more than one environment in the same AWS Account, however we recommend to stick to one environment - one account.

An environment usually corresponds to a business unit or a department. Inside an environment we add teams and assign them different levels of permissions.

2.1.3 Teams

A team corresponds to an IdP group that has been onboarded to data.all. A special case for the administration of data.all is the **Tenant**, an IdP group with high level application (tenant) permissions. As with IdP groups, users can belong to multiple teams.

Teams corresponds to real teams.

> **but really, what are teams?**
>
> Data in data.all is isolated at team level, meaning that all members of a team can access all team's datasets. Thus, a team is any group of users that can access the team's datasets. We can have bigger teams with generic data and project-based teams owning data that requires more restrictive access to only members of the project.

2.2 Discover

2.2.1 Datasets

A dataset is a representation of multiple AWS resources that helps users store data. When data owners create a dataset on data.all the following resources are created:

- Amazon S3 Bucket to store the data on AWS.
- AWS KMS key to encrypt the data on AWS.
- AWS IAM role that gives access to the data on Amazon S3.
- AWS Glue database that is the representation of the structured data on AWS.

Inside the dataset we can store structured data as tables or unstructured data in folders.

2.2.2 Catalog

data.all centralized Catalog is an inventory of datasets, tables, folders and dashboards. It contains metadata for each of the mentioned data assets and thanks to its search capabilities, users can filter based on type of data, type of asset, tags, region and on glossary terms.

We use the Catalog to search and discover data

2.2.3 Glossaries

A Glossary is a list of terms, organized in a way to help users understand the context of their datasets. For example, terms like "cost", "revenue", etc, can be used to group and search all financial datasets.

Glossaries are used to add meaning to data assets metadata facilitating and enhancing Catalog searching

2.2.4 Shares

A Share is an access request to a data asset. Users search and discover data in the catalog and for those data assets that belong to other teams, users can create a Share on behalf of a team (remember, data access: at team level!!). Then, the owners of the asset can accept or reject the share.

We use Shares to collaborate and share data with other teams.

2.3 Play

2.3.1 Worksheets

Worksheets are AWS Athena sessions that allow us to query our datasets as if we were in the AWS Athena Query editor console.

2.3.2 Notebooks

Data practitioners can experiment machine learning algorithms spinning up Jupyter notebook with access to all your datasets. data.all leverages Amazon SageMaker instance to access Jupyter notebooks.

2.3.3 ML Studio

With ML Studio Notebooks we can add users to our SageMaker domain and open Amazon SageMaker Studio

2.3.4 Pipelines

In order to distribute data processing, data.all introduces data.all pipelines where: - data.all takes care of CI/CD infrastructure - data.all offers flexible pipeline blueprints to deploy AWS resources and a Step Function

2.3.5 Dashboards

In the Dashboard window we can start Quicksight sessions, create visual analysis and dashboards.

3. Administrate

3.1 Tenant and Organizations

data.all manages teams' permissions at four levels:

1. Tenant team
2. Organization
3. Environment (next section)
4. Teams (next section)

3.1.1 Tenant

data.all has a super user's team which is a group from your IdP that has the right to manage high level application (tenant) permissions for all IdP groups integrated with data.all.

This super user's team maps to a group from your IdP that's by default named "***DAAdministrators***", any user member of this group will be able to:

- create organizations
- manage tenant permissions on onboarded teams (IdP groups) as shown below.

Manage tenant permissions

As a user part of "***DAAdministrators***" on your IdP you can access the settings menu from the profile icon.

For example, Maria Garcia is not part of "***DAAdministrators***", therefore she sees nothing

On the other hand, Tenant user is part of this group and can navigate to **Admin settings**

In *Admin Settings*, the Tenant user can manage tenant permissions. In the following picture, the user is NOT granting the *DataScienceTeam* that John belongs to permissions to create an organization.

If the tenant revokes the permission of a team to manage an object, that team won't be able to perform any action on that particular object. For the given example, assuming that John only belongs to the *DataScienceTeam*, he is not able to create organizations:

3.1.2 Organizations

Organizations are high level constructs where business units can collaborate across many different AWS accounts at once. An organization includes environments and teams (see next section). Organizations are abstractions, they **don't** contain AWS resources, consequently there is no CloudFormation stack associated with them.

Organizations usually correspond to whole organizations, organization divisions or a separated geographical region within an organization.

Create an organization

> **Organization permissions**
>
> Any user can create an organization as long as he or she belongs to a group with tenant permission "Manage Organizations" (see previous chapter, "Manage tenant permissions").

To create an organization, on the left pane select **Organization**, click **Create** and complete the following form.

organization_form

Field	Description	Required	Editable	Example
Organization name	Name of the organization	Yes	Yes	AnyCompany EMEA
Short description	Short description about the organization	No	Yes	AnyCompany EMEA region
Team	Name of the team managing the organization	Yes	No	EMEAAdmin
Tags	List of tags	No	Yes	fin,rnd,mark,sales

The next step to onboard your IdP groups is to link an environment and add teams, check Link an environment and Add a team to an environment

Edit and update an organization

On the organisation window we can check the organization metadata, as well as the environments and teams that belong to this organisation (we will come back to this in Environments and teams).

To edit the metadata of the organisation, click in **Edit** and update the information. Name, description and tags are editable, however the organisation team cannot be updated.

Delete an organization

> **Warning**
>
> Make sure that you delete the organisation environments before deleting the organisation. Otherwise, orphan environments might run into conflicts.

To archive an organisation, click on the **Archive** button next to the Edit button. A window with the previous warning will appear. If you want to go ahead and delete the organization, type *permantly archive* in the box and submit.

3.2 Environments and Teams

An environment is a **workplace** where a team can bring, process, analyze data and build data driven applications. Environments comprise AWS resources, thus when we create an environment, we deploy a CDK/CloudFormation stack to an AWS account and region. In other words, **an environment is mapped to an AWS account in one region, where users store data and work with data.**

> **One AWS account, One environment**
>
> To ensure correct data access and AWS resources isolation, onboard one environment in each AWS account. Despite being possible, **we strongly discourage users to use the same AWS account for multiple environments**.

3.2.1 Bootstrap your AWS account

*data.all*does not create AWS accounts. You need to provide an AWS account and complete the following bootstraping steps on that AWS account in each region you want to use.

1. Create AWS IAM role

data.all assumes a IAM role named **PivotRole** to be able to call AWS SDK APIs on your account. You can download the AWS CloudFormation stack from *data.all* environment creation form. (Navigate to an organization and click on link an environment to see this form)

2. Setup AWS CDK

data.all uses AWS CDK to deploy and manage resources on your AWS account. AWS CDK requires some resources to exist on the AWS account, and provides a command called `bootstrap` to deploy these specific resources.

Moreover, we need to trust data.all infrastructure account. data.all codebase and CI/CD resources are in the data.all **tooling account**, while all the resources used by the platform are located in a **infrastructure account**. From this last one we will deploy environments and other resources inside each of our business accounts (the ones to be boostraped).

To boostrap the AWS account using AWS CDK, you need :

1. to have AWS credentials configured in ~/.aws/credentials or as environment variables.
2. to install cdk : `npm install -g aws-cdk`
3. to run the following command :

```
cdk bootstrap --trust DATA.ALL_AWS_ACCOUNT_NUMBER  -c @aws-cdk/core:newStyleStackSynthesis=true --cloudformation-execution-policies arn:aws:iam::aws:policy/
AdministratorAccess aws://YOUR_ENVIRONMENT_AWS_ACCOUNT_NUMBER/ENVIRONMENT_REGION
```

> **Which account should I put in the command?**
>
> Let's check with an example: the **tooling account** is 111111111111 and data.all was deployed to the **infrastructure account** = 222222222222. Now we want to onboard a **business account** = 333333333333 in region eu-west-1. Then the cdk bootstrap command will look like: `bash`
> `cdk bootstrap --trust 222222222222  -c @aws-cdk/core:newStyleStackSynthesis=true --cloudformation-execution-policies`
> `arn:aws:iam::aws:policy/AdministratorAccess aws://333333333333/eu-west-1`

3. Enable AWS Lake Formation

data.all relies on AWS Lake Formation to manage access to your structured data. If AWS Lake Formation has never been activated on your AWS account, you need to create a service-linked role, using the following command:

```
aws iam create-service-linked-role --aws-service-name lakeformation.amazonaws.com
```

Service link creation error

If you receive: An error occurred (InvalidInput) when calling the CreateServiceLinkedRole operation: Service role name AWSServiceRoleForLakeFormationDataAccess has been taken in this account, please try a different suffix. **You can skip this step, as this indicates the Lake formation service-linked role exists.**

4. Amazon Quicksight

This is an optional step. To link environments with ***Dashboards enabled*** , you will also need a running Amazon QuickSight subscription on the bootstraped account. If you have not subscribed to Quicksight before, go to your AWS account and choose the Enterprise option as show below:

quicksight

quicksight

After you've successfully subscribed to QuickSight, we need to trust *data.all* domain on QuickSight to enable Dashboard Embedding on *data.all* UI. To do that go to:

1. Manage QuickSight
2. Domains and Embedding
3. Put *data.all* domain and check include subdomains
4. Save

quicksight_domain

3.2.2 NEW Link an environment

Necessary permissions

Environment permissions

Only organization Administrator teams can link environments to the Organization. The Organization creator team is the by default Organization Administrator team, but users of this group can now invite other teams and grant them permission to manage organization teams, and link environment to the organization.

Managing organization teams can be done through the UI or APIs. From the UI, navigate to your organizations and click on the **Teams** tab.

Invite button opens a dialog that gives the organization creators the possibility to invite one of the IdP groups they belong to, which will appear in a dropdown when we click on **Teams**. They can also invite an IdP group that they don't belong to, as long as they type the exact group name (**case sensitive**):

You can check the Organization administrators teams in the Organization's **Teams** tabs and remove a team if necessary on the icon in the Actions column.

Link environment

Once the AWS account/region is bootstraped and we have permission to link an environment to an organization, let's go! Navigate to your organization, click on the **Link Environment** button, and fill the environment creation form:

Field	Description	Required	Editable	Example
Environment name	Name of the environment	Yes	Yes	Finance
Short description	Short description about the environment	No	Yes	Finance department teams
Account number	AWS bootstraped account maped to the environment	Yes	No	111111111111
Region	AWS region	Yes	No	Europe (Ireland)
IAM Role name	Alternative name of the environment IAM role	No	No	anotherRoleName
Resources prefix	Prefix for all AWS resources created in this environment. Only (^[a-z-]*$)	Yes	Yes	fin
Team	Name of the group initially assigned to this environment	Yes	No	FinancesAdmin
Tags	Tags that can later be used in the Catalog	Yes	Yes	finance, test
VPC Identifier	VPC provided to host the environment resources instead than the default one created by *data.all*	No	No	vpc-.....
Public subnets	Public subnets provided to host the environment resources instead than the default created by *data.all*	No	No	subnet-....
Private subnets	Private subnets provided to host the environment resources instead than the default created by *data.all*	No	No	subnet-.....

Features Management

An environment is defined as a workspace and in this workspace we can flexibly activate or deactivate different features, adapting the workspace to the teams' needs. If you want to use Dashboards, you need to complete the optional fourth step explained in the previous chapter "Bootstrap your AWS account".

> **This is not set in stone!**
>
> Don't worry if you change your mind, features are editable. You can always update the environment to enable or disable a feature.

Click on Save, the new Environment should be displayed in the Environments section of the left side pane.

3.2.3 Manage your Environment

Go to the environment you want to check. You can find your environment in the Environments list clicking on the left side pane or by navigating to the environment organization. There are several tabs just below the environment name:

- Overview: summary of environment information and AWS console and credential access.
- Teams: list of all teams onboarded to this environment.
- Datasets: list of all datasets owned and shared with for this environment
- Networks: VPCs created and owned by the environment
- Warehouses: Redshift clusters imported or created in this environment
- Subscriptions: SNS topic subscriptions enabled or disabled in the environment
- Tags: editable key-value tags
- Stack: CloudFormation stack details and logs

Environment access

If **none** of the teams you belong to (IdP groups) has been onboarded to the environment, you won't be able to see the environment in the environments menu or in the organization environments list. **Check the "Manage teams" section**

Check CloudFormation stack

After linking an environment we can check the deployment of AWS resources in CloudFormation, click on the environment and then on the **Stack** tab. Right after linking an environment you should find something like the below picture.

After some minutes its status should go from "PENDING" to "CREATE_COMPLETE" and we will be able to look up the AWS resources created as part of the environment CloudFormation stack. Moreover, we can manually trigger the update in case of change sets of the CloudFormation stack with the **Update** button.

Pro Tip

If something in the creation or update of an environment fails, we can directly check the logs by clicking the logs button. No need to navigate to the AWS console to find your logs!

After being processed (not in `PENDING`), the status of the CloudFormation stack is directly read from CloudFormation.

Edit and update an environment

Find your environment in the Environments list or by navigating to the corresponding organization. Once in your selected environment, click on **Edit** in the top-right corner of the window and make all the changes you want.

Finally, click on **Save** at the bottom-right side of the page to update the environment.

Automatically updates the CloudFormation stack

Clicking on Save will update the environment metadata as well as the CloudFormation stack on the AWS account

Delete an environment

In the chosen environment, next to the Edit button, click on the **Delete** button.

orphan *data.all* resources

A message like this one: *"Remove all environment related objects before proceeding with the deletion!"* appears in the delete display. Don't ignore it! Before deleting an environment, clean it up: delete its datasets and other resources.

Note that we can keep the environment CloudFormation stack. What is this for? This is useful in case you want to keep using the environment resources (IAM roles, etc) created by *data.all* but outside of *data.all*

Create networks

Networks are VPCs created from *data.all* and belonging to an environment and team. To create a network, click in the **Networks** tab in the environment window, then click on **Add** and finally fill the following form.

I need an example!

What is the advantage of using networks from *data.all*? ....[MISSING INFO]

Create Key-value tags

In the **Tags** tab of the environment window, we can create key-value tags. These tags are not *data.all* tags that are used to tag datasets and find them in the catalog. In this case we are creating AWS tags as part of the environment CloudFormation stack. There are multiple tagging strategies as explained in the documentation.

3.2.4 Manage Teams

Environment creators have all permissions on the environment, and can invite other teams to the onboarded environment. To add an IdP group to an environment, navigate to the **Teams** tab of the environment and click on the **Invite** button.

A display will allow you to customize the AWS permissions that the onboarded group will have, adapting to different types of users (data scientists, data engineers, data analysts, management). The customizable permissions can be enabled or disabled as appears in the following picture.

When the invitation is saved, the environment CloudFormation stack gets automatically updated and creates a new IAM role for the new team. The IAM role policies mapped to the permissions granted to the invited team (e.g., a team invited without "Create Redshift clusters" permission will not have redshift permissions on the associated IAM role).To remove a group, in the *Actions* column select the minus icon.

> **Automated permission assignment**
>
> Groups retrieved from the IdP are automatically granted all application high level permissions by default to accelerate the onboarding process.

Users will only be able to see the environments where a team that they belong to has been onboarded (either as creator of the environment or invited to the environment). In the following picture, John belongs to the *DataScienceTeam* that owns the *Data Science* environment, but on top of that he can access the *Data Analysis* environment because her team has been invited by Maria.

> **Pro tip!**
>
> You know whether you are `OWNER` or `INVITED` in an environment by checking **your Role** in that environment. This information appears in the picture in each environment box in the field "Role".

> **Difference between invited and owner**
>
> A team that has been invited to an environment has slight limitations, because well, it is not their environment! Invited teams cannot access the **Stack** tab of the environment because they should not be handling the resources of the environment. Same applies for **Tags** and **Subscriptions**. Other limitations come from the permissions that have been assigned to the team.

AWS access

data.all makes it easier to manage access to your AWS accounts. How? remember when we assigned granular AWS permissions to invited groups and this created an IAM role? (If not, check the "Manage teams" section). From the **Teams** tab of the environment we can assume our team's IAM role to get access to the AWS Console or copy the credentials to the clipboard. Both options are under the "Actions" column in the Teams table.

3.2.5 Manage Consumption Roles

Data.all creates or imports one IAM role per Cognito/IdP group that we invite to the environment. With these IAM roles data producers and consumers can ingest and consume data, but sometimes we want to consume data from an application such as SageMaker pipelines, Glue Jobs or any other downstream application. To increase the flexibility in the data consumption patterns, data.all introduces Consumption Roles.

Any IAM role that exists in the Environment AWS Account can be added to data.all. In the **Teams** tab click on *Add Consumption Role*

A window like the following will appear for you to introduce the arn of the IAM role and the Team that owns the consumption role. Only members of this team and tenants of data.all can remove the consumption role.

Existing roles only

Data.all checks whether that IAM role exists in the AWS account of the environment before adding it as a consumption role.

Data Access

- By default, a new consumption role does NOT have access to any data in data.all.
- The team that owns the consumption role needs to open a share request for the consumption role as shown in the picture below.

4. Discover

4.1 Datasets

4.1.1 Datasets

In data.all, a Dataset is a representation of multiple AWS resources that helps users store data and establish the basis to make this data discoverable and shareable with other teams.

When data owners create a dataset the following resources are deployed on the selected environment and its linked AWS account:

1. Amazon S3 Bucket to store the data on AWS.
2. AWS KMS key to encrypt the data on AWS.
3. AWS IAM role that gives access to the data on Amazon S3.
4. AWS Glue database that is the representation of the structured data on AWS.

> **AWS Champion**
>
> data.all does all the infrastructure heavy lifting for data owners using **AWS CDK** and **AWS CloudFormation** service while following AWS deployment and security best practices.

Tables and Folders

Inside a dataset we can store structured data in tables and unstructured data in folders.

- Tables are the representation of **AWS Glue Catalog** tables that are created on the dataset's Glue database on AWS.
- Folders are the representation of an **Amazon S3 prefix** where data owners can organize their data. For example, when data is loaded, it can go to a folder named “raw” then after it's processed the data moves to a folder called “silver” and so on.

Dataset ownership

Dataset ownership refers to the ability to access, modify or remove data from a dataset, but also to the responsibility of assigning these privileges to others.

- **Owners**: When you create a dataset and associate it with a team, the dataset business ownership belongs to the associated team.
- **Stewards**: You can delegate the stewardship of a dataset to a team of stewards. You can type a name of an IdP group or choose one of the teams of your environment to be the dataset stewards.

> **Note**
>
> Dataset owners team is a required, non-editable field, while stewards are optional and can be added post the dataset has been created. If no other stewards team is designated, the dataset owner team will be the only responsible in managing access to the dataset.

Dataset access

In this case we are referring to the ability to access, modify or remove data from a dataset. Who can access the dataset content? users belonging to...

- the dataset owner team
- a dataset steward team
- teams with a share request approved to dataset content

> **Note**
>
> Dataset metadata is available for all users in the centralized data catalog.

4.1.2 Create a dataset

On left pane choose **Datasets**, then click on the **Create** button. Fill the dataset form.

create_dataset

Field	Description	Required	Editable	Example
Dataset name	Name of the dataset	Yes	Yes	AnyDataset
Short description	Short description about the dataset	No	Yes	For AnyProject predictive model
Environment	Environment (mapped to an AWS account)	Yes	No	DataScience
Region (auto-filled)	AWS region of the environment	Yes	No	Europe (Ireland)
Organization (auto-filled)	Organization of the environment	Yes	No	AnyCompany EMEA
Owners	Team that owns the dataset	Yes	No	DataScienceTeam
Stewards	Team that can manage share requests on behalf of owners	No	Yes	FinanceBITeam, FinanceMgmtTeam
Confidentiality	Level of confidentiality: Unclassified, Oficial or Secret	Yes	Yes	Secret
Topics	Topics that can later be used in the Catalog	Yes, at least 1	Yes	Finance
Tags	Tags that can later be used in the Catalog	Yes, at least 1	Yes	deleteme, ds

4.1.3 Import a dataset

If you already have data stored on Amazon S3 buckets, data.all got you covered with the import feature. In addition to the fields of a newly created dataset you have to specify the S3 bucket and optionally a Glue database:

Field	Description	Required	Editable	Example
Amazon S3 bucket name	Name of the S3 bucket you want to import	Yes	No	importedBucket
AWS Glue database name	Name of the Glue database tht you want to import	No	No	anyDatabase

import_dataset

4.1.4 Navigate dataset tabs

When we belong to the dataset owner team

After creating or importing a dataset it will appear in the datasets list (click on Datasets on the left side pane). In this window, it will only ve visible for those users belonging to the dataset owner team. If we select one of our datasets we will see the following dataset window:

with

When we DON'T belong to the dataset owner team

How do we access a dataset if we don't have access to it? IN THE CATALOG! on the left pane click on Catalog, find the dataset you are interested in, click on it and if you don't have access to it, you should see only some of the tabs in comparison with the previous pic, something like:

without

4.1.5 Edit and update a dataset

Data owners can edit the dataset by clicking on the **edit** button, editing the editable fields and saving the changes.

4.1.6 Delete a dataset

To delete a dataset, in the selected dataset window click on the **delete** button in the top-right corner. As with environments, it is possible to keep the AWS CloudFormation stack to keep working with the data and resources created but outside of data.all.

4.1.7 Check dataset info and access AWS

The **Overview** tab of the dataset window contains dataset metadata, including governance and creation details. Moreover, AWS information related to the resources created by the dataset CloudFormation stack can be consulted here: AWS Account, Dataset S3 bucket, Glue database, IAM role and KMS Alias.

You can also assume this IAM role to access the S3 bucket in the AWS console by clicking on the **S3 bucket** button. Alternatively, click on **AWS Credentials** to obtain programmatic access to the S3 bucket.

overview

4.1.8 Fill the dataset with data

Tables

Quickly upload a file for data exploration

Users may want to experiment with a small set of data (e.g. a csv file). To create tables from a file, we first upload the file, then run the crawler to infer its schema, and finally, we read the schema by synchronizing the table. Upload & Crawl & Sync

1. Upload data: Go to the **Upload** tab of the dataset and browse or drop your sample file. It will be uploaded to the dataset S3 bucket in the prefix specified. By default, a Glue crawler will be triggered by the upload of a file, however this feature can be disabled as appears in the picture.

upload

1. Crawl data: the file has been uploaded but the table and its schema have not been registered in the dataset Glue Catalog database. If you have disabled the crawler in the upload, click on the **Start Crawler** button in the Data tab. If you just want to crawl one prefix, you can specify it in the Start Crawler feature.

crawl

1. Synchronize tables: Once crawled and registered in the Glue database, you can synchronize tables from your dataset's AWS Glue database by using **Synchronize** tables feature in the Data tab. In any case, data.all will synchronize automatically the tables for you at a frequency of **15 minutes**.

You can preview your small set of data right away from data.all, check Tables.

Ingest data

If you need to ingest larger quantities of data, manage bigger files, or simply you cannot work with local files that can be uploaded; this is your section!

There are multiple ways of filling our datasets with data and actually, the steps don't differ much from the upload-crawl-sync example.

- Crawl & Sync option: we can drop the data from the source to our dataset S3 bucket. Then, we will crawl and synchronize data as we did in the previous steps 2 and 3.
- Register & Sync option: we drop the data from the source to our dataset S3 bucket. However, if we want to have more control over our tables and its schema, instead of starting the crawler we can **register the tables** in the Glue Catalog and then click on Synchronize as we did in step 3.

How do we register Glue tables? There are numerous ways:

- manually from the AWS Glue console in your environment account
- Using AWS Glue API, `CreateTable`.
- In a Glue Job leveraging Glue PySpark DynamicFrame class
- With boto3
- Or with AWS Data Wrangler, Pandas on AWS.
- Also, you can deploy Glue resources using CloudFormation
- Or directly, migrating from Hive Metastore.
- there are more for sure :)

> **Use data.all pipelines to register Glue tables**
>
> data.all pipelines can be used to transform your data including the creation of Glue tables using AWS Glue pyspark extension . Visit the pipelines section for more details.

Folders

As previously defined, folders are prefixes inside our dataset S3 bucket. To create a folder, go to the **Data** tab and on the folders section, click on Create. The following form will appear. We will dive deeper in how to use folders in the folders section.

create_folder

4.1.9 Leave a message in Chat

In the **Chats** button users can interact and leave their comments and questions on the Dataset Chat.

feed

4.1.10 Create key-value tags

Same as in environments. In the **Tags** tab of the dataset window, we can create key-value tags. These tags are not data.all tags that are used to tag the dataset and find it in the catalog. In this case we are creating AWS tags as part of the dataset CloudFormation stack. There are multiple tagging strategies as explained in the documentation.

4.2 Tables and Folders

4.2.1 **Tables**

In this section we will go through the different tabs in the Table window. We can reach this view:

1. by selecting a table from the data Catalog

2. or in the dataset view, in the **Tables** tab clicking on the arrow in the *Actions* column for the chosen table.

Check table metadata

Also in the table window, go to the **Overview** tab where you will find the following information:

- URI: unique table identifier
- Name: name of the registered table in the Glue Catalog
- Tags
- Glossary terms
- Description
- Organization, Environment, Region, Team: inherited from the dataset
- Created: creation time of the table
- Status: `INSYNC`

> **Description, Tags and Glossary terms are not inherited!**
>
> If a dataset is tagged with Tags and Glossary terms, the child tables do not inherit these tags and terms. In the Overview tab, by clicking on **Edit** is where you can add them. Same applies for the description. Adding tags and terms to your tables will make them more discoverable in the Catalog.

Add or edit table metadata

Edit your table metadata by clicking on the **Edit** button.

Preview data

Data preview gives you the ability to preview a subset of the data available on data.all. Preview feature is available for data you own or data that's shared with you.

Just select a table and in the **Preview** tab you will find the results of an SQL select subset of the table.

Leave a message in Chat

As with datasets, in the **Chats** button users can interact and leave their comments and questions on the Table Chat.

Add column description

Metadata makes more sense when columns description fields are not empty. With data.all you can add columns description and avoid the pain of figuring out fields purpose.

Select one table and in the **Columns** tab, directly type the description in the Description column as shown in the picture.

Profile data

Data profiling refers to the process of examining, analyzing, and reviewing the data available in the source by collecting statistical information about the data set's quality and hygiene. This process is called also data archaeology, data assessment, data discovery, or data quality analysis. Data profiling helps in determining the accuracy, completeness, structure, and quality of your data.

Data profiling in data.all involves:

- Collecting descriptive statistics like minimum, maximum, mean, median, and standard deviation.
- Collecting data types, along with the minimum and maximum length.
- Determining the percentages of distinct or missing data.
- Identifying frequency distributions and significant values.

By selecting the **Metrics** tab of your data table you can run a profiling job (click in the **Profile** button) , or view the latest generated data profiling metrics:

Delete a table

Deleting a table means deleting it from the data.all Catalog, but it will be still available on the AWS Glue Catalog. Moreover, when data owners delete a table, they are **not** deleting its data from the dataset S3 bucket. Teams with shared access to the dataset cannot delete tables or folders, even if they are shared.

It is possible to delete a table from the dataset **Tables** tab with the trash can icon next to each of the tables in the *Actions* column.

Another option is to go to the specific table (on the above picture click on the arrow icon next to the trash can icon). Click on the **Delete** button in the top right corner and confirm the deletion.

> **An error occurred (ResourceShared) when calling DELETE_DATASET_TABLE operation: Revoke all table shares before deletion**
>
> To protect data consumers, if the table is shared you cannot delete it. The share requests to the table need to be revoked before deleting the table. Check the Shares section to learn how to grant and revoke access.

4.2.2 Folders

To open the Folder window you can either find your chosen folder in the Catalog or navigate to the dataset and then in the **Folders** tab click on the arrow in the *Actions* column of your folder:

Check folder and S3 metadata

The **Overview** tab of the folder contains folder metadata: - URI: unique folder identifier - Name: name of the folder, it is made out of the dataset name concatenated with the S3 prefix - Tags - Glossary terms - Description - Organization, Environment, Region, Team: inherited from the dataset - Created: creation time of the table

Add or edit table metadata

Edit your folder metadata by clicking on the **Edit** button.

> **Description, Tags and Glossary terms are not inherited**
>
> Careful, those 3 fields are not synced with their dataset metadata. Just click on the **Edit** button of the folder to complete any missing information. This is especially useful to improve Catalog search of your folders.

Check the content of your folder

To check what kind of files does our prefix content, we can access the AWS S3 console on the **S3 Bucket** button of the Folder **Overview** tab.

Leave a message in Chat

Exactly the same as with tables. Allow your teams to discuss directly on the Folder Chat.

Delete a folder

Deleting folders is analogous to deleting tables. Deletion means deletion from the data.all Catalog and the content of the S3 prefix remains in the dataset S3 bucket. Only dataset owners can delete dataset folders.

The steps to delete a folder are exactly the same as with tables. You can either go to the dataset and in the **Folders** tab click on the can trash icon on the *Actions* column of the selected folder; or you can navigate to the Folder and click on the **Delete** button.

> **An error occurred (ResourceShared) when calling DELETE_DATASET_FOLDER operation: Revoke all folder shares before deletion**
>
> To protect data consumers, if the table is shared you cannot delete it. The share requests to the table need to be revoked before deleting the table. Check the Shares section to learn how to grant and revoke access.

4.3 Centralized Catalog and glossaries

4.3.1 **Catalog**

In the Catalog we have a record with metadata for each dataset, table, folder and dashboard in data.all. Users come to this centralized Catalog to search and find data owned by other teams. Once users find a data asset they are interested in, they will create a Share request.

How do users find the data that they need?

Data needs to be discoverable, for this reason data.all Catalog offers a variety of filters that use business context to improve your search:

- **Type of data**: `dataset`, `table`, `folder` and/or `dashboard`
- **Tags**: tags of the data asset.
- **Topics**: filter by general topics created by the user.
- **Region**: AWS region where the data asset is located.
- **Classification**: `unclassified`, `official` and/or `secret`
- **Glossary**: filter datasets by the glossary terms created by users. This helps in two ways: It lets you narrow down results quickly using granular glossary terms like "sales", "profit", etc. Traditionally, a data glossary is just used to organize data. However, data.all uses it to power its search. This further encourages users to enrich and maintain the glossary regularly.

4.3.2 **Glossaries**

A Glossary is a list of terms, organized in a way to help users understand the context of their datasets. For example, terms like "cost", "revenue", etc, can be used to group and search all financial datasets.

The use of familiar terminology helps in quickly understanding the data and its background. It is a crucial element of data governance as it helps in bringing the business understanding closer to an organization's data initiatives.

On data.all, glossary terms can be attached to any dataset and can be leveraged to power quick and ease data discovery in the Catalog.

> **Spotlight**
>
> Glossaries are built hierarchically. They are made of categories and terms. This structure allows for glossaries from multiple domains to co-exist.

Term:

- A term is the lowest unit which is unique inside each glossary.
- It describes the content of the data assets in the most useful and precise way.
- It can exist independently, without belonging to any particular category or sub-category.

Category:

A category is used to group the terms of a similar context together. it is just a way of organizing terms.

Create a new glossary

1. Go to **Glossaries** menu on the elft side pane.
2. Click on **Create**.
3. Fill the form and add a new glossary.

create_glossary

Add a category inside a Glossary

1. Click on the button "Add category" to add a new category.
2. Add a name and description to your category for better understanding.

add_category

Add terms to a category

1. Click on the button "Add term" to add a new term to the category.
2. Give it an appropriate name and description.

add_term

> **Remember!**
>
> The term will be used to recognize and filter the datasets. Hence, keep it short and precise.

Link your data with appropriate glossary terms

You can associate a glossary term to a dataset or a table. Go to a dataset click on "edit" and update the glossary terms field as shown below

link_term

Approve and Check all data related to a glossary

To see a list of all datasets and tables that have been linked with terms of a specific glossary, go to Glossaries and select the glossary. In the **Associations** tab it is possible to check the related data assets (target name), their types (e.g. dataset) and the specific term that they have used.

Important: Glossary owners need to approve the association. If it is not approved it won't be used as filter in the catalog.

relatedterm

4.4 Shares

Teams can browse data.all catalog and request access for data assets. data.all shares data between teams securely within and environment and across environments without any data movement.

Datasets can contain tables and folders. Tables are Glue Tables registered in Glue Catalog. data.all uses (and automates) Lake Formation sharing feature to create access permissions to tables, meaning that no data is copied between AWS accounts.

Under-the-hood, folders are prefixes inside the dataset S3 bucket. To create sharing of folders in data.all, we create an S3 access point per requester group to handle its access to specific prefixes in the dataset.

Concepts

- Share request or Share Object: one for each dataset and requester team.
- Share Item refers to the individual tables and folders that are added to the Share request.

Sharing workflow

Requesters create a share request and add items to it. Both requesters and approvers can work on this `DRAFT` of the request. Then requesters submit it and wait until approvers pick it up from the `PENDING APPROVAL` and approve or reject the request which will go to `APPROVED` or `REJECTED` status correspondingly. If at a later state an approved share needs to be rejected, approvers can revoke access. Finally, approvers and requesters can always edit the share request which will send it back to a `DRAFT` state.

wf

Create a share request to a table (requester)

On left pane choose **Catalog** then **Search** for the table you want to access. Click on the lock icon of the selected data asset.

catalog_search

The following window will open. Choose your target environment and team and optionally add a *Request purpose*.

share_request_form

If instead of to a team, you want to request access for a Consumption role, add it to the request as in the picture below.

share_request_form

Finally, click on **Send Request**. This will create a share request or object for the corresponding dataset and if you have requested a table or folder it will add those items to the request. The share needs to be submitted for the request to be sent to the approvers.

Submit a share request (requester)

A created share request needs to be filled with requested tables and/or folders and then submitted. For the previous example, where we requested access to a table directly from the share request form, the requester can click on **Submit** from the Shares menu on the *Sent* tab.

submit_share_2

If we have created a share request for a dataset or we want to edit the tables and folders that we want to access to, select **Learn more** to open the request. Is *supermarket_sales* the only table you want to get access to? yes? then click on **Submit** and wait for the approver to approve or reject the share request.

Approve/Reject a share request (approver)

> **Notifications**
>
> When a share request is submitted, the approvers receive a notification with the user that is requesting access and the dataset. And viceversa, when a request is approved, the requesters get a notification with the approver's name and the approved dataset.

shares As a dataset **owner** or **steward** you can approve or reject a dataset access request. To do that, you have 2 options. First, you can go from the Shares menu and in the *Received* tab approve or reject the request that is pending approval.

If you want to check the content of the request before approving or rejecting, click on **Learn more** and the following will open. Here you can see the tables and folders added to the dataset share request.

accept_share

If a dataset share request has been accepted, the requester should see the dataset on his or her screen. The role with regards to the dataset is `SHARED`.

Add/delete items to/from a share request (requester/approver)

When you create a share request for a dataset, you still need to add the items (tables or folders) that you want to get access to. Or if you have an existing share request with some shared tables and folders, you can add more items to the request or delete shared ones.

It will be more clear with an example. As appears in the picture, go to the share request and click on **Add Item** to add a table or folder to the request. On the contrary, if you want to remove a shared item click on the trash icon next to it.

If you are adding a new item to the request, the following window will open to let you choose a specific table or folder in the dataset.

add_share

> **You have to submit the request again!!**
>
> Note that after you have added or removed an item, the share request status has gone from `APPROVED` to `DRAFT`. Since the request has changed it needs to be resubmitted. **Both, approvers and requesters, can add or delete items from a share request, but ONLY requesters can submit the new draft of the request.** So after adding or deleting items requesters have to follow the steps for resubmitting explain above.

Revoke a share request (approver)

What happens if you granted access (you are an approver) to a dataset/tables/folders and now you want to revoke that access? In case you made a mistake, or your data cannot be shared any longer; go to the Shares menu and look for the specific share request in the *Received* tab. In the top right corner there is a **Revoke** button that will behave as a rejection of a pending approval share request. In fact the status of the request changes to `REJECTED`.

4.4.1 Check your sent/received share requests

Anyone can go to the **Shares** menu on the left side pane and look up the share requests that they have received and that they have sent.

4.4.2 Consume shared data

Data.all tables are Glue tables shared using AWS Lake Formation, therefore any service that reads Glue tables and integrates with Lake Formation is able to consume the data. Permissions are granted to the team role or the consumption role that has been specified in the request.

For the case of folders, the underlying sharing mechanism used is S3 Access Points. You can read data inside a prefix using the IAM role of the requester (same as with tables) and executing get calls to the S3 access point.

Use data subscriptions

data.all helps data owners publish notification updates to all their data consumers. It also helps data consumers react to new data shared by the owners.

STEP 1: ENABLE SUBSCRIPTIONS ON THE ENVIRONMENT

Check the environment documentation for the steps to enable subscriptions.

> **AWS SNS Topics**
>
> When subscriptions are enabled, **as a data producer you can publish a message** to the producers SNS topic. You can also **subscribe to data consumers SNS topic** to be aware of the latest data updates from the producers.

STEP 2: PUBLISH NOTIFICATION UPDATE

IMPORTANT

This feature is disabled at the moment

5. Play

5.1 Worksheets

data.all offers a rich editor to write SQL queries and explore data. It is Athena on the backend that runs our queries on environments where our teams have been onboarded.

5.1.1 Create a Worksheet

On the left pane under **Play** click on **Worksheets** to go to the Worksheet menu. Here you will find all Worksheets owned by your teams.

> **Shared queries = Seamless Collaboration**
>
> Check, learn from and collaborate with other members of your team to improve your analyses and get insights from your data, directly from data.all worksheets. No need to send queries by email, no need to create views :)

To create a new worksheet click on the **Create** button in the top right corner and fill the Worksheet form:

worksheets

Field	Description	Required	Editable	Example
Worksheet name	Name of the worksheet	Yes	Yes	PalmDor
Short description	Short description about the worksheet	No	Yes	Query used to retrieve Palm D'or winners
Team	Team that owns the worksheet	Yes	No	DataScienceTeam
Tags	Tags	No	Yes	adhoc

> **No AWS resources**
>
> When we are creating a worksheet we are NOT deploying AWS resources. We don't provision clusters, we are not creating tables or views. We simply store the query in data.all database and we run it serverlessly on AWS Athena.

5.1.2 Edit worksheet metadata

Select a worksheet and click on the pencil icon to edit the metadata of the worksheet. This includes worksheet name, description and tags. The ownership of the worksheet, its team, is not editable.

5.1.3 Delete a worksheet

Next to the edit button, there are 2 other buttons. To delete a worksheet click on the trash icon one. Worksheets are not AWS resources, they are a data.all construct whose information is stored in the data.all database. Thus, when we delete a worksheet we are not deleting AWS resources or CloudFormation stacks.

5.1.4 Write and save your queries

Select your worksheet and choose any of the environments, datasets and tables of your team to list column information. In the query editor write your SQL statements and click on **Run Query** to get your query results. Error messages coming from Athena will pop-up automatically.

worksheets

If you want to save the current query for later or for other users, click on the **save** icon (between the edit and the delete buttons).

More than just SELECT

Worksheets can be used for data exploration, for quick ad-hoc queries and for more complicated queries that require joins. As far as you have access to the joined datasets you can combine information from multiple tables or datasets. Check the docs for more information on AWS Athena SQL syntax.

5.2 Notebooks

Data practitioners can experiment machine learning algorithms spinning up Jupyter notebook with access to all your datasets. data.all leverages Amazon SageMaker instance to access Jupyter notebooks.

5.2.1 Create a Notebook

> **Pre-requisites**
>
> To use Notebooks you need to introduce your own VPC ID or create a Sagemaker Studio domain inside a VPC (read the docs). Provisioning the notebook instances inside a VPC enables the notebook to access VPC-only resources such as EFS file systems.

To create a Notebook, go to Notebooks on the left pane and click on the **Create** button. Then fill in the following form:

notebooks

Field	Description	Required	Editable	Example
Sagemaker instance name	Name of the notebook	Yes	No	Cannes Project
Short description	Short description about the notebook	No	No	Notebook for Cannes exploration
Tags	Tags	No	No	deleteme
Environment	Environment (and mapped AWS account)	Yes	No	Data Science
Region (auto-filled)	AWS region	Yes	No	Europe (Ireland)
Organization (auto-filled)	Organization of the environment	Yes	No	AnyCompany EMEA
Team	Team that owns the notebook	Yes	No	DataScienceTeam
VPC Identifier	VPC provided to host the notebook	No	No	vpc-.....
Public subnets	Public subnets provided to host the notebook	No	No	subnet-....
Instance type	[ml.t3.medium, ml.t3.large, ml.m5.xlarge]	Yes	No	ml.t3.medium
Volume size	[32, 64, 128, 256]	Yes	No	32

If successfully created we can check its metadata in the **Overview** tab. Unlike other data.all resources, Notebooks are non-editable.

notebooks

5.2.2 Check CloudFormation stack

In the **Stack** tab of the Notebook, is where we check the AWS resources provisioned by data.all as well as its status. As part of the Notebook CloudFormation stack deployed using CDK, data.all will deploy:

1. AWS EC2 Security Group
2. AWS SageMaker Notebook Instance
3. AWS KMS Key and Alias

5.2.3 Delete a Notebook

To delete a Notebook, simply select it and click on the **Delete** button in the top right corner. It is possible to keep the CloudFormation stack associated with the Notebook by selecting this option in the confirmation delete window that appears after clicking on delete.

5.2.4 Open JupyterLab

Click on the **Open JupyterLab** button of the Notebook window to start writing code on Jupyter Notebooks.

buttons

5.2.5 Stop/Start instance

As we briefly commented, data.all uses AWS SageMaker instances to access Jupyter notebooks. Be frugal and stop your instances when you are not developing. To do that, close the Jupyter window and click on **Stop Instance** in the Notebook buttons. It takes a couple of minutes, just refresh and check the Notebook Status in the overview tab. It should end up in `STOPPED`.

> **Save money, stop your instances**
>
> This feature allows users to easily manage their instances directly from data.all UI.

Same when you are coming back to work on your Notebook, click on **Start instance** to start the SageMaker instance. In this case the Status of the notebook should first be `PENDING` and once the instance is ready, `INSERVICE`.

5.2.6 Create Key-value tags

In the **Tags** tab of the notebook window, we can create key-value tags. These tags are not *data.all* tags that are used to tag datasets and find them in the catalog. In this case we are creating AWS tags as part of the notebook CloudFormation stack. There are multiple tagging strategies as explained in the documentation.

5.3 ML Studio

With ML Studio Notebooks we can add users to our SageMaker domain and open Amazon SageMaker Studio

5.3.1 Create a ML Notebook

To create a new Notebook, go to ML Studio on the left side pane and click on Create. Then fill in the creation form with its corresponding information.

notebooks

Field	Description	Required	Editable	Example
Sagemaker Studio profile name	Name of the user to add to SageMaker domain	Yes	No	johndoe
Short description	Short description about the notebook	No	No	Notebook for Cannes exploration
Tags	Tags	No	No	deleteme
Environment	Environment (and mapped AWS account)	Yes	No	Data Science
Region (auto-filled)	AWS region	Yes	No	Europe (Ireland)
Organization (auto-filled)	Organization of the environment	Yes	No	AnyCompany EMEA
Team	Team that owns the notebook	Yes	No	DataScienceTeam

5.3.2 Check CloudFormation stack

In the **Stack** tab of the ML Studio Notebook, is where we check the AWS resources provisioned by data.all as well as its status. As part of the CloudFormation stack deployed using CDK, data.all will deploy some CDK metadata and a SageMaker User Profile.

5.3.3 Delete a Notebook

To delete a Notebook, simply select it and click on the **Delete** button in the top right corner. It is possible to keep the CloudFormation stack associated with the Notebook by selecting this option in the confirmation delete window that appears after clicking on delete.

notebooks

5.3.4 Open Amazon SageMaker Studio

Click on the **Open ML Studio** button of the ML Studio notebook window to open Amazon SageMaker Studio.

notebooks

5.4 Pipelines

Different business units might have their own data lake and ingest and process the data with very different tools: Scikit Learn, Spark, SparkML, AWS SageMaker, AmazonAthena… The diversity of tools and use-cases result in a wide variety of CICD standards which discourages development collaboration.

In order to distribute data ingestion and processing, data.all introduces data.all pipelines:

- data.all takes care of CICD infrastructure
- data.all integrates with AWS DDK, a tool to help you build data workflows in AWS
- data.all allows you to define development environments directly from the UI and deploys data pipelines to those AWS accounts

> **Focus on value-added code**
>
> data.all takes care of the CICD and multi-environment configuration and DDK provides reusable assets and data constructs that accelerate the deployment of AWS data workflows, so you can focus on writing the actual transformation code and generating value from your data!

5.4.1 Multi-environment Pipelines

In some cases, enterprises decide to separate CICD resources from data application resources, which at the same time, need to be deployed to multiple accounts. Data.all allows users to easily define their CICD environment and other infrastructure environments in a flexible, robust way.

Let's see it with an example. In your enterprise, the Research team has 3 AWS accounts: Research-CICD, Research-DEV and Research-PROD. They want to ingest data with a data pipeline that is written in Infrastructure as Code (IaC) in the Research-CICD account. The actual data pipeline is deployed in 2 data accounts. First, in Research-DEV for development and testing and once it is ready it is deployed to Research-PROD.

Pre-requisites

As a pre-requisite, Research-DEV and Research-PROD accounts need to be bootstrapped trusting the CICD account (`-a` parameter) and setting the stage of the AWS account, the environment id, with the `e` parameter. Assuming 111111111111 = CICD account the commands are as follows:

- In Research-CICD (111111111111): `ddk bootstrap -e cicd`
- In Research-DEV (222222222222): `ddk bootstrap -e dev -a 111111111111`
- In Research-PROD (333333333333): `ddk bootstrap -e prod -a 111111111111`

In data.all we need to link the AWS accounts to the platform by creating 3 data.all Environments: Research-CICD Environment, Research-DEV Environment and Research-PROD Environment.

Creating a pipeline

data.all pipelines are created from the UI, under Pipelines. We need to fill the creation form with the following information:

- Name, Description and tags
- CICD Environment: AWS account and region where the CICD resources will be deployed.
- Team, this is the Admin team of the pipeline. It belongs to the specified CICD Environment where the pipeline is defined as IaC
- Development strategy:

Finally, we need to add **Development environments**. These are the AWS accounts and regions where the infrastructure defined in the CICD pipeline is deployed.

> **environment ID = data.all environment stage**
>
> When creating the pipeline and adding development environments, you define the stage of the environment. The bootstrap `e` parameter needs to match the one that you define in the data.all UI. In our example, we bootstraped with the parameters "dev" and "prod" and then we defined the stages as "dev" and "prod" correspondingly.

`create_pipeline`

CDK pipelines - Trunk-based

This CodePipeline pipeline is based on the CDK Pipelines library. As stated in the documentation, CDK Pipelines is an opinionated construct library. It is purpose-built to deploy one or more copies of your CDK applications using CloudFormation with a minimal amount of effort on your part.

CODECOMMIT REPOSITORY

When a pipeline is created, a CloudFormation stack is deployed in the CICD environment AWS account. It contains an AWS CodeCommit repository with the code of an AWS DDK application set up for a multi-account deployment, as explained in its documentation.

In the deployed repository, data.all pushes a `ddk.json` file with the details of the selected development environments:

```
{
    "environments": {
        "cicd": {
            "account": "111111111111",
            "region": "eu-west-1"
        },
        "dev": {
            "account": "222222222222",
            "region": "eu-west-1",
            "resources": {
                "ddk-bucket": {"versioned": false, "removal_policy": "destroy"}
            }
        },
        "prod": {
            "account": "333333333333",
            "region": "eu-west-1",
            "resources": {
                "ddk-bucket": {"versioned": true, "removal_policy": "retain"}
            }
        }
    }
}
```

In addition, the `app.py` file is also written accordingly to the development environments selected in data.all UI.

```
# !/usr/bin/env python3

import aws_cdk as cdk
from aws_ddk_core.cicd import CICDPipelineStack
from ddk_app.ddk_app_stack import DDKApplicationStack
from aws_ddk_core.config import Config

app = cdk.App()

class ApplicationStage(cdk.Stage):
    def __init__(
            self,
            scope,
            environment_id: str,
            **kwargs,
    ) -> None:
        super().__init__(scope, f"dataall-{environment_id.title()}", **kwargs)
        DDKApplicationStack(self, "DataPipeline-PIPELINENAME-PIPELINEURI", environment_id)

config = Config()
(
    CICDPipelineStack(
        app,
        id="dataall-pipeline-PIPELINENAME-PIPELINEURI",
        environment_id="cicd",
        pipeline_name="PIPELINENAME",
    )
        .add_source_action(repository_name="dataall-PIPELINENAME-PIPELINEURI")
        .add_synth_action()
        .build().add_stage("dev", ApplicationStage(app, "dev", env=config.get_env("dev"))).add_stage("prod", ApplicationStage(app, "prod",
env=config.get_env("prod")))
        .synth()
)

app.synth()
```

CICD DEPLOYMENT

data.all backend performs the first deployment of the CICD stack defined in the CodeCommit repository. The result is a CloudFormation template deploying a CICD pipeline having the aforementioned CodeCommit repository as source. This CodePipeline pipeline is based on the CDK Pipelines library.

create_pipeline

CodePipeline pipelines - Trunk-based or GitFlow

For cases in which we need more control over the CICD pipeline, instead of using CDK Pipelines library we can use aws-codepipeline construct library.

CODECOMMIT REPOSITORY AND CICD DEPLOYMENT

When a pipeline is created, a CloudFormation stack is deployed in the CICD environment AWS account. It contains:

- an AWS CodeCommit repository with the code of an AWS DDK application where we made some modifications to allow cross-account deployments.
- CICD CodePipeline(s) pipeline that deploy(s) the application

In the first run of the pipeline we will perform some initialization actions from the pipeline itself (you don't need to do anything). In short, we initialize the DDK application by running `ddk init` and we push the code back to our repository.

This is the original repository:

created_pipeline

This is the repository once it has been initialized in the commit "First Commit from CodeBuild - DDK application":

created_pipeline

We added the `Multiaccount` configuration class that allows us to define the deployment environment based on the `ddk.json`. Go ahead and customize this configuration further, for example you can set additional `env_vars`.

Trunk-based pipelines append one stage after the other and read from the main branch of our repository:

created_pipeline

Gitflow strategy uses multiple CodePipeline pipelines for each of the stages. For example if you selected `dev` and `prod`:

created_pipeline

The `dev` pipeline reads from the `dev` branch of the repository:

created_pipeline

5.4.2 Which development strategy should I choose?

CDK pipelines - Trunk-based

1. The `CDK-pipelines` construct handles cross-account deployments seamlessly and robustly. It synthesizes CDK stacks as CloudFormation stacks and performs the deployment cross-account. Which means that we don't manually assume IAM roles in the target accounts, all is handled by CDK :)
2. It also allows developers to modify the CICD stack as it is self-mutating. It is easy to customize having several typical CodePipeline stages out-of-the-bix. For example, developers can add monitoring, tests, manual approvals directly in the repository with single-line changes.

CodePipeline pipelines - Trunk-based or GitFlow

1. The `aws-codepipelines` construct uses AWS CodePipelines directly. We are able to define any type of CICD architecture, such as in this case Trunk-based and GitFlow.
2. Developers working on the pipeline cannot modify the CICD pipeline
3. Cross-account deployments require specific definition of the environment in the code.

Summary

CDK pipelines are recommended for flexibility and for a robust cross-account application deployment, whereas CodePipeline pipelines are recommended if you need to provide an immutable pipeline architecture or if you want to implement a GitFlow strategy.

5.4.3 Cloning the repository

Pre-requisites:

1. Install git: `sudo yum install git`
2. Install pip: `sudo yum -y install python-pip`
3. Install git-remote-codecommit: `sudo pip install git-remote-codecommit`

Clone the repo:

1. Get the AWS Credentials from the AWS Credentials button in the Pipeline overview tab.
2. Clone the repository with the command in the overview tab.

created_pipeline

5.5 Dashboards

Data.all connects with Amazon Quicksight to allow users to quickly visualize and analyse their data.

5.5.1 **Start Quicksight session**

qs

5.5.2 **Import a dashboard**

6. Security

Details on data.all security-first approach to building a modern data workspace

6.1 Data and metadata on data.all

6.1.1 **Data virtualization**

data.all is a fully virtualized solution that does not involve moving data from existing storage layers.

Any queries run on the data.all are pushed to existing processing layers (e.g. directly to your database, warehouse, or a processing layer such as Athena or Presto on top of S3).

6.1.2 **Data and metadata storage**

Data and metadata collected and created by data.all are stored in applications and databases within the customer's VPC (virtual private cloud). This includes information for data previews and queries, data quality, metadata, and user data.

6.1.3 **Data previews and queries**

data.all gives users the ability to see sample data previews for a datasets and results for any queries run on data.all.

In both cases, the request is pushed upstream to the original data source, and a 100-row sample of the result is provided to data.all users.

6.1.4 **Data quality profile**

Users can generate data quality metrics with the click of a button on data.all. Once generated, these metrics are stored in PostgreSQL on the customer's VPC.

6.1.5 **Metadata**

Dataset metadata, including metadata generated on data.all, is stored across Elasticsearch and Aurora PostgreSQL.

Elasticsearch is used to optimize search on the product, and Aurora PostgreSQL acts as a persistence backend.

6.1.6 **User data**

Data on users, roles, and IdP groups is stored in a PostgreSQL database.

Any user data transmitted over the internet is SSL-encrypted over HTTPS.

6.1.7 **Authentication**

The data.all authentication process is based SAML 2.0–based login. data.all can also integrate into organizations' existing SAML 2.0–based SSO authentication systems.

7. Platform Monitoring

As an administrator of data.all I want to know the status of data.all. In this section we will focus on the following aspects of monitoring:

- Platform observability
- Platform usage

7.1 Observability

It refers to the infrastructure of data.all, the frontend and backend.

7.1.1 AWS CloudWatch

As part of the deployment, data.all deploys observability AWS resources with CDK and ultimately in CloudFormation. These include AWS CloudWatch Alarms on the infrastructure: on Aurora DB, on the OpenSearch cluster, on API errors… Operation teams can subscribe to a topic on Amazon SNS to receive near real time alarms notifications when issues are occurring on the infrastructure.

7.1.2 AWS CloudWatch RUM

Additionally, if we enabled CloudWatch RUM in the config.json file when we deployed data.all we will be able to collect and view client-side data about your web application performance from actual user sessions in near real time.

7.2 Platform usage

I want to know how my teams are using the platform. Inside this category we answer questions such as *"how many environments or datasets are in data.all?"*.

7.2.1 RDS Queries

The first option is to query the RDS metadata database that contains all the information regarding environments, datasets and other data.all objects. You need access to the data.all infrastructure account, in which you will: 1) Navigate to RDS Console 2) Connect with secrets manager ARN 3) Get this ARN from AWS Systems Manager Parameter Store (search for "aurora") 4) Run SQL statements to extract insights about the usage of the platform

7.2.2 Quicksight enabled monitoring

When we deployed data.all, we can configure optional monitoring of Quicksight, this is the `enable_quicksight_monitoring` parameter. If enabled, we allow AWS Quicksight to establish a VPC connection with our RDS metadata database in that account. We modify the security group of our Aurora RDS database to communicate with Quicksight, then we can use AWS Quicksight to create rich dynamic analyses and dashboards based on the information on RDS. Once the deployment is complete you need to follow the next steps:

1) Pre-requisite: Quicksight Enterprise Edition We need to subscribe to Quicksight and allow data.all domain to embed dashboards, follow the instructions in the step 4 of the Linking environment section.

2) Create Quicksight VPC connection

Follow the steps in the documentation and make sure that you are in the same region as the infrastructure of data.all. For example, in this case Ireland region.

quicksight

To complete the set-up you will need the following information:

- VPC_ID of the RDS Aurora database, which is the same as the data.all created one. If you have more than one VPC in the account, you can always check this value in AWS SSM Parameters or in the Aurora database as appears in the picture:

quicksight

- Security group created for Quicksight: In the VPC console, under security groups, look for a group called `<resource-prefix>-<envname>-quicksight-monitoring-sg` For example using the default resource prefix, in an environment called prod, look for `dataall-prod-quicksight-monitoring-sg`.

3) Create Aurora data source We have automated this step for you! As a tenant user, a user that belongs to `DAADministrators` group, sign in to data.all. In the UI navigate to the **Admin Settings** window by clicking in the top-right corner. You will appear in a window with 2 tabs: Teams and Monitoring. In the Monitoring tab, introduce the VPC connection name that you created in step 2 and click on the *Save* button. Then, click on the *Create Quicksight data source* button. Right now, a connection between the RDS database and Quicksight has been established.

quicksight

4) Customize your analyses and share your dashboards Go to Quicksight to start building your analysis by clicking on the *Start Quicksight session* button. First, you need to create a dataset. Use the **dataall-metadata-db** data source, this is our connection with RDS.

quicksight

Use this dataset in an analysis (check the docs customization of analyses) and publish it as a dashboard (docs in publish dashboards)

> **Not only RDS**
>
> With Quicksight you can go one step further and communicate with other AWS services and data sources. Explore the documentation for cost analyses in AWS with Quicksight or AWS CloudWatch Logs collection and visualization with Quicksight.

5) Bring your dashboard back to data.all Once your dashboard is ready, copy its ID (you can find it in the URL as appears in the below picture)

quicksight

Back in the data.all Monitoring tab, introduce this dashbaord ID. Now, other tenants can see your dashboard directly from data.all UI!

quicksight

8. Labs

8.1 Hands-on Lab: Data Access Management with data.all teams

This document is a step-by-step guide illustrating some functionalities of the data.all "Teams" feature. This guide is far from exhaustive and mainly focuses on how users can share data across environment and teams. After completing it, you are free to continue exploring data.all and the functionalities it provides.

8.1.1 🎯 Scope of this guide

To follow this guide, you will need:

- An AWS account (#111111111111) where data.all is deployed. Your version of data.all must support the "Teams" feature
- An AWS account that will be used as a data.all environment for the data platform team (#222222222222)
- An AWS account that will be used as a data.all environment for the data science team (#333333333333)

The scenario you will implement in this guide is the following. The data platform team owns a dataset. Data scientists are interested by the content of this dataset for their analysis. There are however two different types of data scientists, that are members of two different teams: data science team A and data science team B.

A data scientist from team A will request access to the data platform dataset for its team. A data platform user will then accept the request, thus granting team A read-only access to the dataset. Team B does not have access to the dataset from the data platform team.

Then a user in team A creates a dataset in the data science environment. We will check that users in team B does not have access to this data.

You will go through the following steps to implement this scenario:

1. Create users and groups in Cognito
2. Create the Organisation and the Environment for the data platform team
3. Create the Dataset for the data platform team and upload some data
4. Create the Organisation and the Environment for the data science team
5. Invite team A and team B to the data science environment
6. Share data platform data with team A in the data science environment
7. Create a Dataset managed by team A in the data science account

Here is an illustration of the scenario:

1. Create users and groups in Cognito

First, you need to create users and groups from the Cognito console. This happens in the account where the infrastructure of data.all is deployed (#111111111111). You will later use these users to connect to data.all. Go to the AWS console and create five groups and four users as follow:

Cognito group

- **DAAdministrators**: group for data.all administrators
- **DataPlatformaAdmin**: group for data platform admin team
- **DataScienceAdmin**: group for data science admin team
- **TeamA**: first category of data scientists
- **TeamB**: second category of data scientists

Cognito users

- **data.alladmin**: create this user and add it to both in the DAAdministrators and DataPlatformAdmin groups. This user will be able to manage permissions of all teams in data.all (tanks to the DAAdministrators group membership) and will be able to create resources for the data platform team
- **ds-admin**: create this user and add it to the DataScienceAdmin group. This user will create resources for the data science team
- **ds-a**: create this user and add it to TeamA
- **ds-b**: create this user and add it to TeamB

> **Warning**
>
> When creating users, you will need to **provide both the name of the user and its email.**

After creating users and assigning them to groups in Cogntio, you end-up with the following situation:

2. Create the Organisation and the Environment for the data platform team

We will start by creating the resources for the data platform team. Log into data.all with the user in the **DataPlatformAdmin** group. Create an Organisation for the Data Platform team. Make sure that the DataPlatformAdmin team manages this organization.

Now link a new environment to this organisation. You can do this by clicking on **Environment** and then **Link Environment**

When onboarding a new AWS account as an environment in data.all, you usually need to make some operations in the account first. The UI lists these operations for you: bootstrapping the AWS account and creating the data.allPivotRole notably. You will have to go through these operations if it is the first time you use this AWS account to create an environment in data.all. Then, create the environment by providing a name, the account ID (#**222222222222**) and the Team managing it (**DataPlatformAdmin**).

Wait until the stack is deployed successfully. You can check the status of the stack in the **stack** tab of the environment. Once the status is **create_complete**, create a new dataset in this environment. You can do it from the **Contribute** window

Deploy this dataset in the environment you have just created. Also make sure that the **DataPlatformAdmin** team owns this dataset (Governance section):

Wait until the dataset is created successfully. You can check the status in the **stack** tab of the dataset. Once the status is **create_complete**, you can start uploading some data from the **upload** tab:

From this window, you are able to upload files in your dataset. When uploading files, you can ask for a crawler running automatingly in your dataset, thus populating a glue database. To make sure the crawler will work, please upload a csv file of your choice. Insert any name you want in the **prefix** section. This will be the name of your Glue table.

Click on the **upload** button. This puts your file in the S3 bucket related to your data.all dataset. It also launches the Glue crawler populating the Glue database. Leave some time for the crawler to run and click on the **Tables** tab. If the crawler ran successfully, clicking on the **synchronize** button will display your table. At this point, feel free to explore your table from the data.all user interface.

We have completed all the tasks on the data platform side. This included the creation of the organisation, the environment, the dataset and the upload of a csv file. This is an illustration of where we are in the process:

Note: As you may have already noted down at the beginning of this guide, the data platform user is also part of the **DAAdministrators** group. Being part of this group enables this user to manage the permissions of all the other teams in data.all. To do so, click **Setting**. This provides the list of teams for which you can manage the permissions.

Click on the icon next to the team's name to manage its permissions

This opens a new window from where you can manage all permissions of the team.

3. Create the Organisation and the Environment for the data science team

You will now create data.all resources for the data science team. Log into data.all with the user in the **DataScienceAdmin** group. Create an Organisation for the data science team. Make sure that the **DataScienceAdmin** team manages this organization.

Now link a new environment to this organisation. Provide a name for this environment, the AWS account ID (**#333333333333**), and the team owning it (**DataScienceAdmin**).

You now have an organisation and an environment managed by the **DataScienceAdmin** team. The next step is to invite team A and team B to this data science environment. This will enable data scientists from team A and team B to access the environment.

4. Invite Team A and Team B to the data science environment

With the user in DataScienceAdmin team, select the data science environment and click on the **Teams** tab. You can invite other teams in your environment with the **invite** button.

This opens a new window asking you to indicate the name of the team you want to invite. You can also manage the permissions this team will have in your environment. Use this **invite** button to invite **TeamA** and **TeamB** in your data science environment.

Users from team A and team B now have access to your environment

5. Share data platform data with Team A in the data science environment

Log into data.all with user in **TeamA**. This user does not own any data, but wants to access data of the data platform team. Go on data.all **Data Catalog** tab. This shows all the datasets and tables you can request access to. There are different tools you can use in order to find the data you are looking for (you can find more information about these in the data.all documentation):

- Directly typing the name of the dataset or the table in the search bar
- Use tags or topics associated to the datasets
- Use data.all Glossary

In this case, data scientist in team A wants to access **mydpdata** uploaded by the data platform team. Use the search bar to find the data. When you see the table you want, click on **Request access**.

This button opens a new window where you can configure your request. When you share a dataset or a table in data.all, the share occurs at an environment and team level. You therefore need to indicate for which environment and for which team you make the request. In this case, the user in team A wants to access data in the data science environment. Fill the request accordingly.

When you click on **Send Request**, this does not directly send the request to the data latform team. It rather creates a Draft that you can still edit in the **Collaborate** tab, under **Sent**. Click on the **Submit** button to send the request

Now re-open a new data.all window connected as the user in the **DataPlatformAdmin** team. This team owns the dataset and is therefore responsible of accepting access requests. It is possible to delegate this right to other teams using **Data Stewards** but we did not set this up in this guide. Under **Collaborate** and **Received**, you can find all the access requests received by the data platform team. Locate the request you just made with the user in Team A. If you want to know more about this request (who is making it, for which table in the dataset,...), click on **Learn More**. If you agree to grant access, click on **Approve**.

This action triggers an ECS task that updates the permissions of the table in Lake Formation. Users in Team A are now able to access the data platform data. Let us verify it.

Re-open data.all connected as the user in **TeamA**. You can first visit the **Contribute** tab where you will see the dataset that has been shared with team A.

Quick reminder: The data platform team agreed to share the **mydpdata** table with TeamA in the data science environment called **DSENV**.

As a conclusion, the table **mydpdata** is accessible from the environment **DSENV**, through a role only team A can assume. Team A users can assume this role directly from the data.all user interface. Select the data science environment and go under the **Teams** tab. You will then see all the teams that have access to the environment. Find TeamA line and click on the AWS logo.

This opens a new window in the AWS console. The AWS account is the one you associated to the data science environment earlier in this guide (#333333333333). Also note that you are assuming a role specific to your team. Use the search bar to get to the Athena console. In the Athena Query editor, you will be able to see under **Database** the dataset shared by the data platform team. The name of this database is a concatenation of "dh" (for data.all), the name of the dataset (dpdataset) and random characters to ensure unicity. Under **Tables**, you can now see **mydpdata** which you can query using with Athena.

Explanation: When the data platform team uploaded the csv file under the **mydpdata** prefix, the crawler created a new Glue table called **mydpdata** in the AWS account associated to the data platform environment (#333333333333). When the data platform team accepted to share **mydpdata** with team A in the data science environment, it triggered an ECS task that updated the Lake Formation (AWS service managing data access) settings in both the data platform and data science environments. It updated the settings in a way that allows the IAM role of team A in the data science environment to read the **mydpdata** table stored in the data

platform environment. In short, only the role of team A in the data science environment is able to read **mydpdata** table (in addition to data platform team of course). This is a **read-only** access, and the data is not moved from the data platform environment to the data science environment.

You can repeat the same thing to check that team B does not have access to the data. Log into data.all with a user in **TeamB** and select the data science environment. Under the **Teams** tab, click on the AWS logo to connect to the AWS console assuming the role of **TeamB**. Go to the Athena Query Editor and you will see that you won't be able to see data shared with team A.

At this stage of the guide, you should better understand how data sharing cross account works. The graph below illustrates where we are in the original scenario.

6. Create a Dataset managed by team A in the data science account

In the previous section of this guide, you went through an example of how you can share data across environment and teams. In this section, we will focus on the creation of datasets in a single account. Team A will create a dataset in the data science environment. We will make sure that other teams invited to the data science environment (t eamB) are not able to access the dataset of team A.

Open data.all with a user in **TeamA**. In the **Contribute** section, create a new dataset in the data science environment. Make sure that **TeamA** owns this dataset.

When the dataset is fully created, upload a csv file from the **upload** tab of the dataset. Upload this file under a prefix named **datateama** to create a new Glue table with the same name. After uploading the file, wait a few minutes to let the crawler do its job. In the **Tables** section, click on **Synchronize** to display your new table.

Now that the data is uploaded, team A is able to access the data as it is registered as the owner of the dataset. However, team B is not able to read the data even if it has access to the environment. If you log into data.all with the user in team B, you won't be able to see the **TeamADataset** in the **Contribute** section. In addition, you will find below two screenshot of the Athena console. In the first screenshot, we assume the role of **TeamA** in the data science environment (process already explained in the previous section). In the second screenshot, we assume the role of **TeamB** in the data science environment. When assuming the role of team A, we can see the team A dataset in the **database** section, and also the **datateama** table. We can then query the data with Athena. However, when assuming the role of team B in the data science environment, we are not able to see any dataset. This is because in this guide, we have not created or shared any dataset with team B. Team B is thus unable to query the data of team A.

This last section illustrated how you can use teams to manage data access in a single environment. You have reached the end of the guide that illustrated some capabilities that data.all brings. Now that you got the basis, fell free to explore all the other things you can do with your data.

Cleanup

When you are done with this guide, you delete your data.all resources (dataset, environment, organization). This also automatically deletes the Cloudformation stacks created in your AWS accounts.