Visual Flow

User Guide

August 2022

Version 0.9.19

Document Revisions

Date	Version Number	Document Changes
12/08/2020	0.9.2	Initial Draft
22/04/2021	0.9.3	Pipeline Operators
26/04/2021	0.9.4	Job Operators
07/05/2021	0.9.5	Project Name, Project Operations
25/05/2021	0.9.6	Project Name in document
09/07/2021	0.9.7	Pipeline Operators, Job Operations, Storages
24/10/2021	0.9.8	Jobs and Pipelines statuses, Custom container, Storages
13/11/2021	0.9.9	New Data Storages
22/11/2021	0.9.10	New Data Storage
15/12/2021	0.9.11	Logs levels
29/12/2021	0.9.12	Cache stage, Job Operations
21/01/2022	0.9.13	New Data Storage, Stage descriptions
04/02/2022	0.9.14	Description of STDOUT storage
18/02/2022	0.9.15	Truncate mode
04/03/2022	0.9.16	Full query mode in Transformer stage
18/03/2022	0.9.17	Sort stage
01/04/2022	0.9.18	Password component
08/08/2022	0.9.19	Verification to ensure up-to-date condition

Revision Details

Version Number	Updates	Performed by
0.9.19	P.3.1 split into 3.1. Getting Started and 3.2. Create Project	Alesya Lemeshevskaya
0.9.19	P.”4.2.. Create a job” updated to “4.2 .Create a job. Available Stages” and broken down to multiple sections per each stage	Alesya Lemeshevskaya
0.9.19	Added section 3 for Connections to p.”3.3. Manage Project Settings”	Alesya Lemeshevskaya
0.9.19	Updated screenshots for Jobs list and Pipelines list	Alesya Lemeshevskaya
0.9.19	Style and verification updates to all chapters	Alesya Lemeshevskaya

Table of Contents

1. Introduction

1.1. Terminology

ETL is an abbreviation for ***e**xtract, **t**ransform, **l**oad*, three database functions combined into one tool to pull data out of one database, transform it and place it into another database.

- **Extract** is the process of *reading data* from a database. In this stage the data is collected, often from multiple and different types of sources.

- **Transform** is the process of *converting the extracted data* from its previous form into the form needed to place it into another database.

- **Load** is the process of *writing the data* into the target database.

Job is a chain of individual stages linked together. It describes the flow of data from a data source to a data target. Usually, a stage has a minimum of one data input and/or one data output. However, some stages can accept more than one data input and output to more than one stage.

In Visual Flow the available stages are:

- Read
- Write
- Join
- Union
- Filter
- Sort
- Group By
- Remove Duplicates
- Transformer
- Change Data Capture
- Cache

Pipeline is a compound of multiple jobs and can be run. In Visual Flow, user can use such stages as:

- Job
- Notification
- Container

1.2. Scope and Purpose

Visual Flow web application is the ETL tool designed for effective data manipulation via convenient and user-friendly interface. The tool has the following capabilities:

- Can integrate data from heterogeneous sources:

✓ AWS S3
✓ DB2
✓ Cassandra
✓ Elasticsearch
✓ IBM COS
✓ Mongo
✓ MSSQL
✓ MySQL
✓ Oracle
✓ PostgreSQL
✓ Redis
✓ Redshift

- Leverage direct connectivity to enterprise applications as sources and targets
- Perform data processing and transformation
- Leverage metadata for analysis and maintenance

1.3. Process Overview

Visual Flow jobs and pipelines exist within a certain namespace (project) so the first step in the application would be to create a project or enter an existing project. Then user needs to enter Job Designer to create a job.

Job Designer is a graphical design interface used to create, maintain, execute and analyze jobs. Each job determines the data sources, the required transformations and destination of the data.

Pipeline designer is a graphical design interface aimed for managing pipelines. Designing a pipeline is similar to designing a job.

Important: When editing stages in the Configuration Panel, to save data user must click *Confirm* button, otherwise the data will be lost.

Visual Flow key functions include but not limited to:

✓ Create project which serves as a namespace for jobs and/or pipelines
✓ Manage project settings
✓ User access management
✓ Create/maintain a job in Job Designer
✓ Job execution and logs analysis
✓ Create/maintain a pipeline in Pipeline Designer
✓ Pipeline execution
✓ Import/Export jobs and pipelines

2. Roles and authorizations

The following roles are available in the application:

- ✓ Viewer
- ✓ Operator
- ✓ Editor
- ✓ Administrator

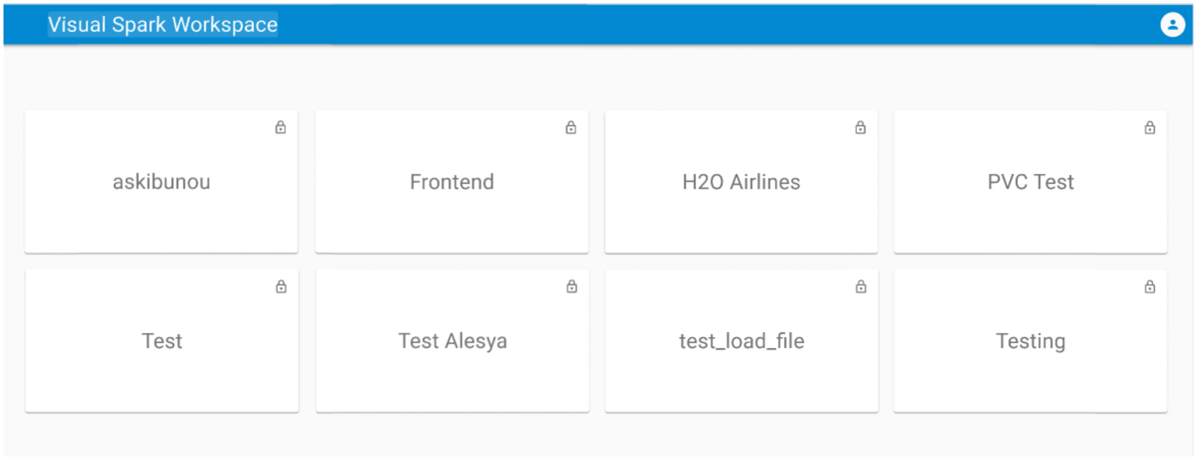
They can perform the below operations within the namespaces they are authorized to. Only a Super-admin user can create a workspace (project) and grant access to this project.

Role	Actions		
	Project Settings	Jobs	Pipelines
Viewer	View all	View all	View all
Operator	View all	View all / execute jobs	View all / execute pipelines
Editor	Edit all but Users and Roles	Edit / execute jobs	Edit / execute pipelines
Admin	Edit all	Edit / execute jobs	Edit / execute pipelines

3. Project Operations.

3.1. Getting started.

Once you first logged on to the application you see the initial screen with all existing projects:

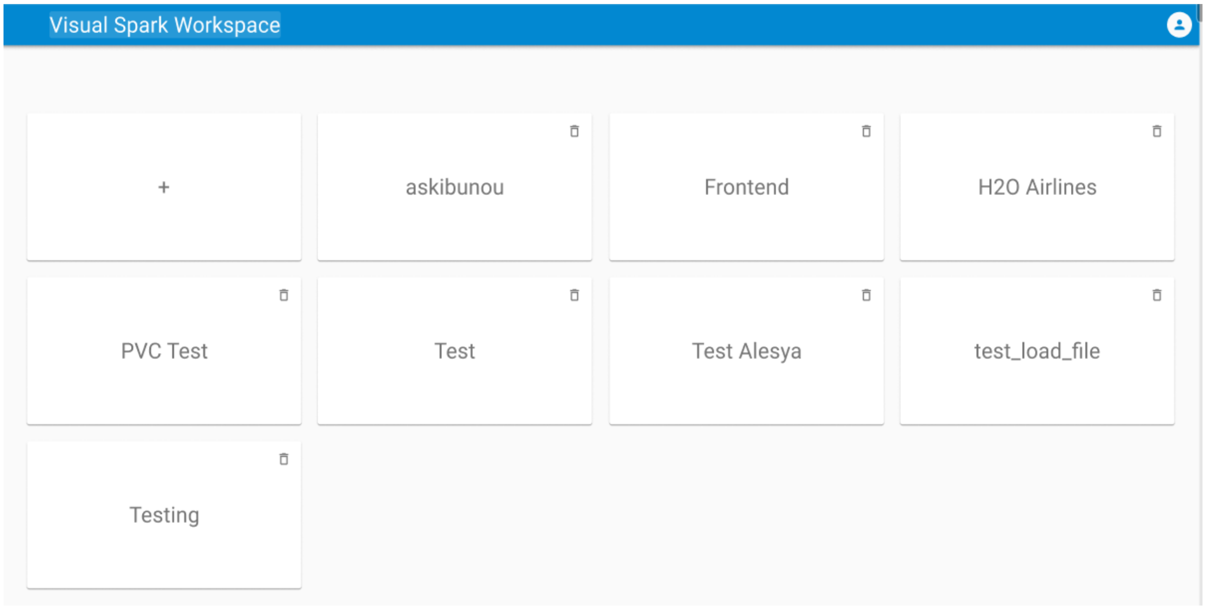


If you are not authorized to a certain project it is locked for you. So you see lock icon on its tile. Please contact project owners to get access to their projects.

3.2. Create a Project.

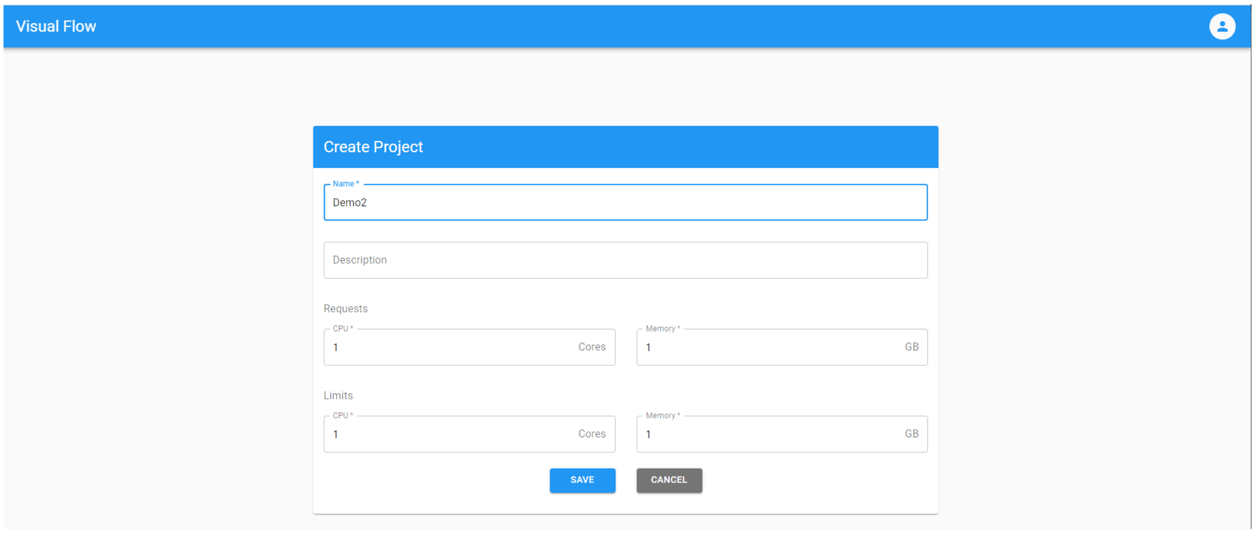
To create a project you need to push "+" button.

Note: this is the action of super-admin user only. The button is not visible for the application roles (Viewer, Operator, Editor, Admin).



With "+" button pushed you get to *Create Project Form* to enter project basic settings:

- Project Name
- Project Description
- Requests (CPU/Memory)
- Limits (CPU/Memory)



Create Project

Name * Demo2

Description

Requests

CPU * 1 Cores

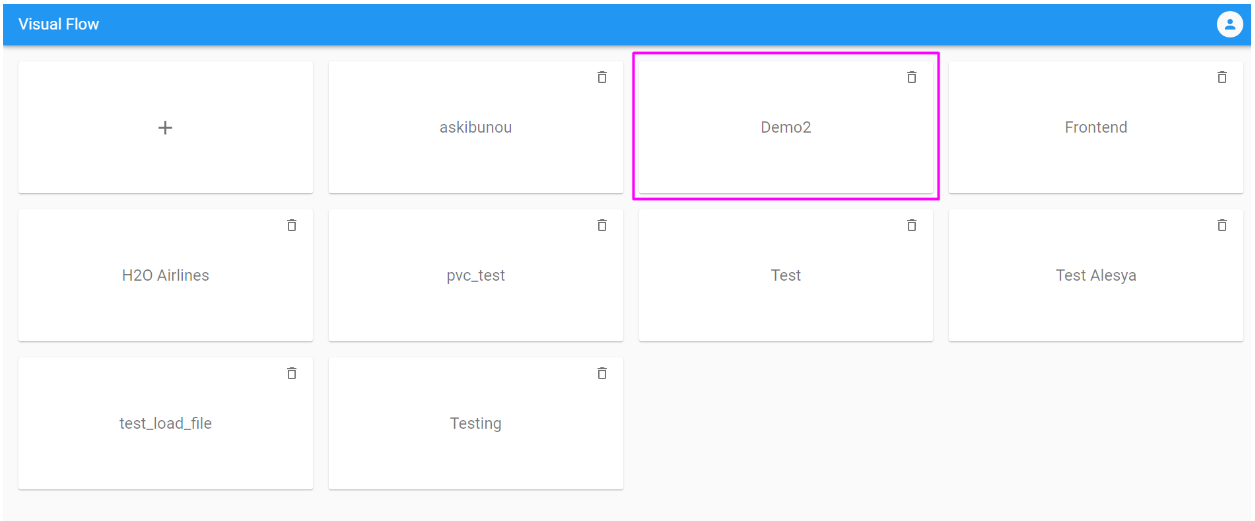
Memory * 1 GB

Limits

CPU * 1 Cores

Memory * 1 GB

After saving *Create Project Form* the project is created under the given name and then can be found on the initial screen:



askibunou

Demo2

Frontend

H2O Airlines

pvc_test

Test

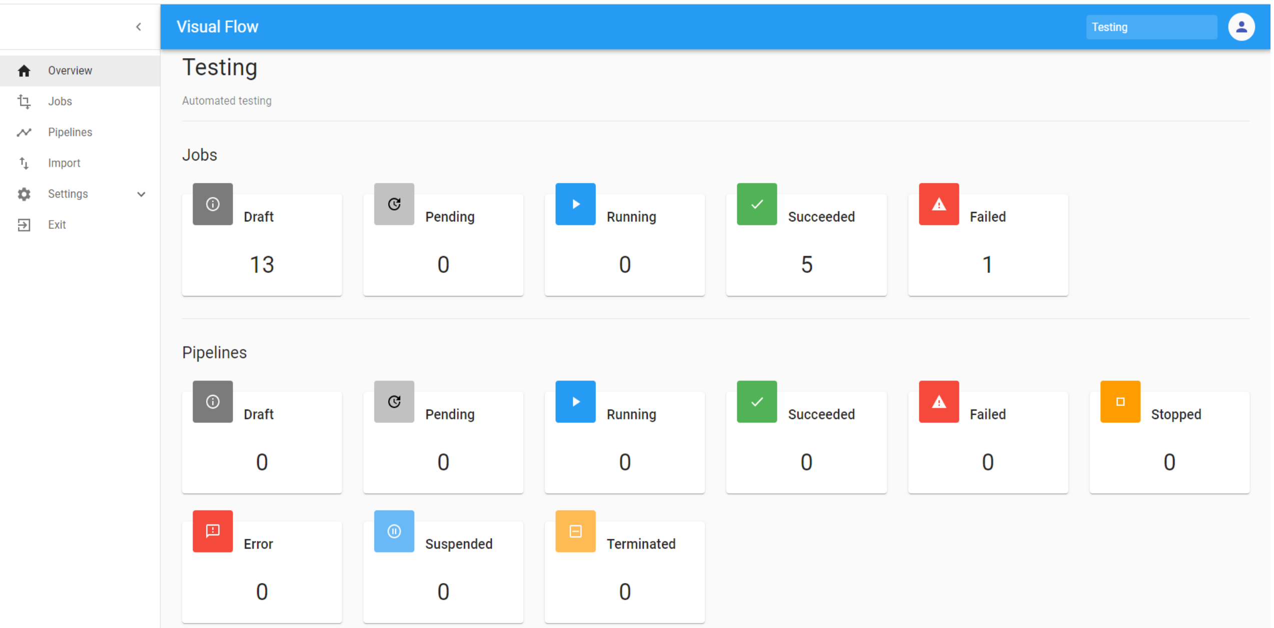
Test Alesya

test_load_file

Testing

3.3. Project Overview

The screen contains project left menu and displays information about the project jobs, pipelines and their resource utilization (applicable for running jobs).



Testing

Automated testing

Jobs

Draft 13

Pending 0

Running 0

Succeeded 5

Failed 1

Pipelines

Draft 0

Pending 0

Running 0

Succeeded 0

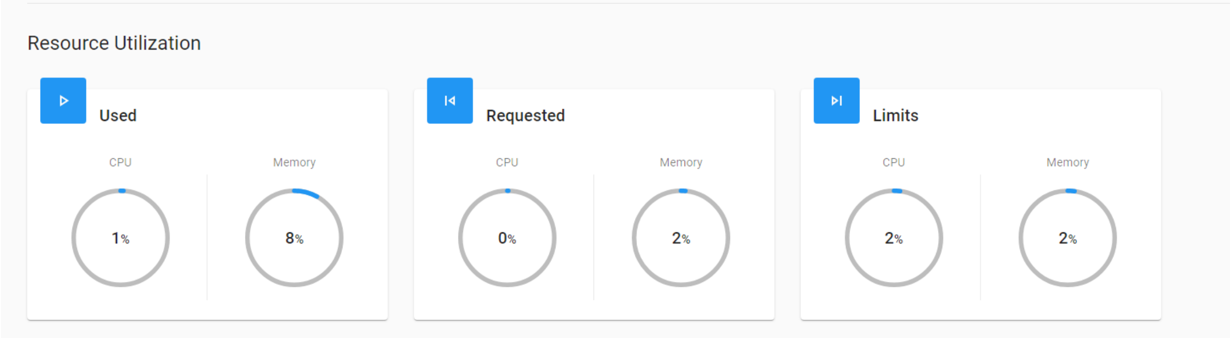
Failed 0

Stopped 0

Error 0

Suspended 0

Terminated 0



Resource Utilization

Used: CPU 1%, Memory 8%

Requested: CPU 0%, Memory 2%

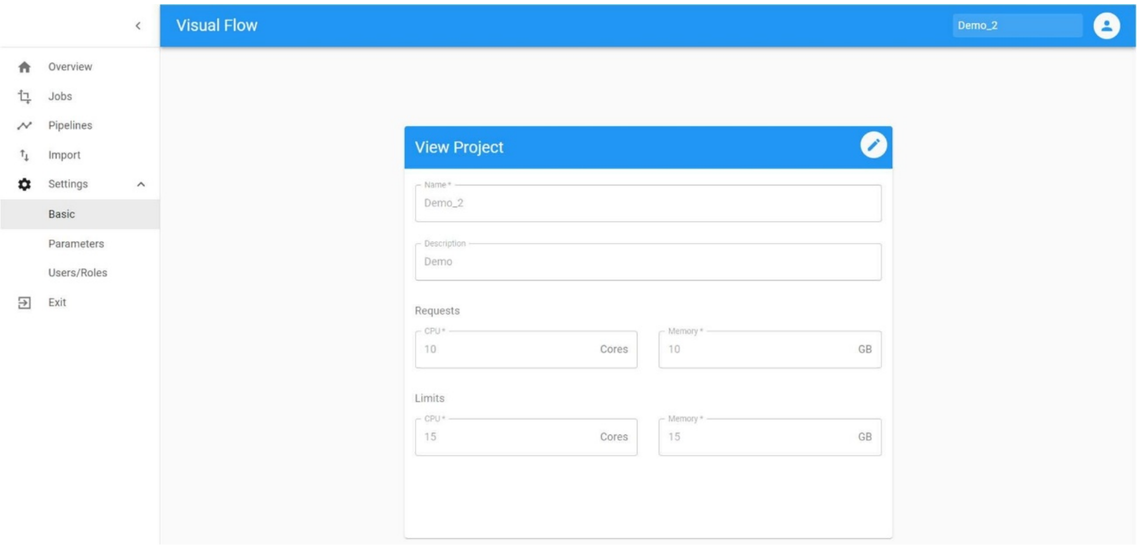
Limits: CPU 2%, Memory 2%

3.4. Manage Project Settings

Settings submenu contains:

- Basic
- Parameters
- Connections
- Users and Roles

1) *Basic* is already there after project creation. *Edit* button turns on the edit mode for updates.



View Project

Name * Demo_2

Description Demo

Requests

CPU * 10 Cores

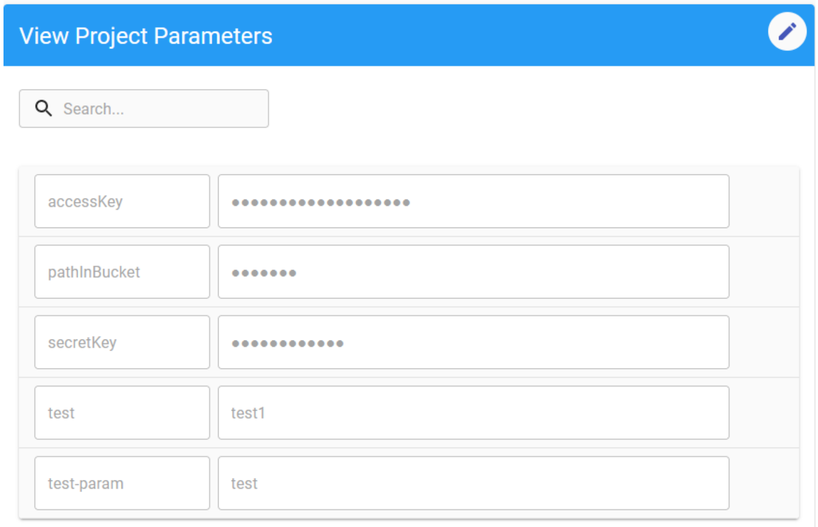
Memory * 10 GB

Limits

CPU * 15 Cores

Memory * 15 GB

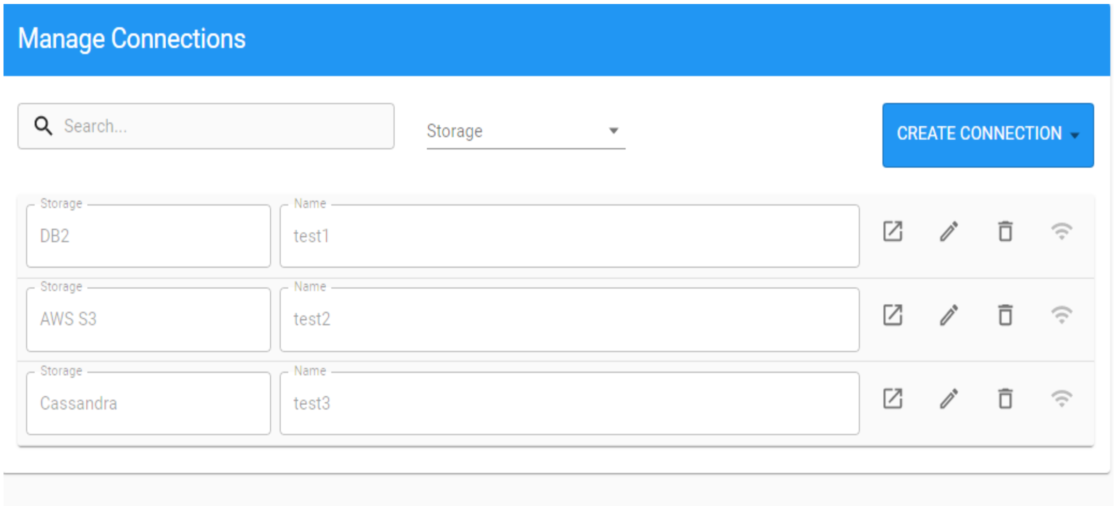
2) Parameters serve to store values required for the entire project, e.g. JDBC connection, DB2 credentials or table schema can be the same for multiple jobs within a project and therefore stored at the project level. *Edit* button turns on the edit mode for updates.



View Project Parameters

accessKey	•••••••••••••••••••
pathInBucket	•••••••
secretKey	••••••••••••
test	test1
test-param	test

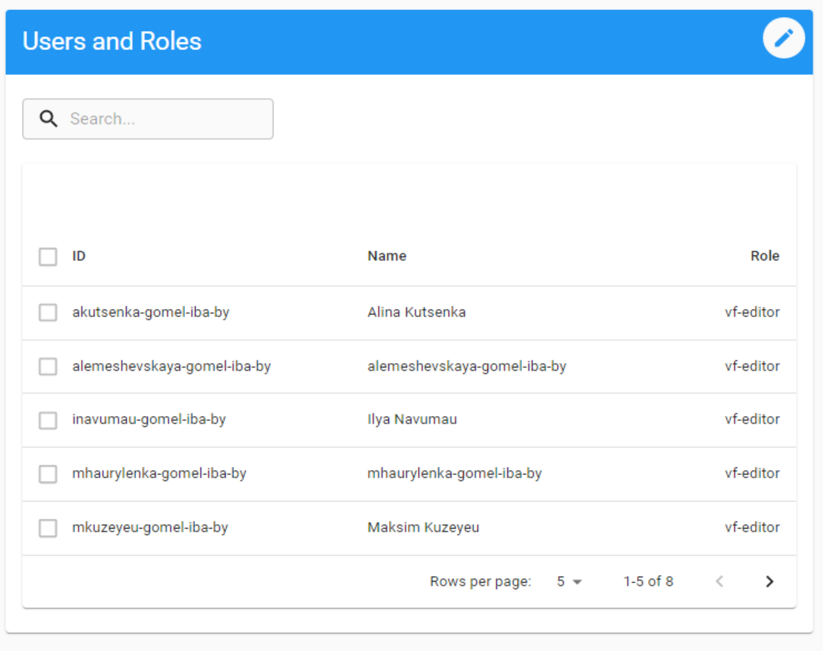
3) Connections option enables user to manage connections to a storage.
Here you see a list of all existing connections with their name/storage type and available actions (view, edit, delete, ping). Also you can create a new connection with *Create Connection* button.



Storage	Name
DB2	test1
AWS S3	test2
Cassandra	test3

4) User and Roles is meant for user access management or to view user access depending on your authorization.

Users cannot change their own roles, this operation can be done by *Admin* or *Super-admin* only.
If you try to change your role you will get error message: “You cannot change your role".
Edit button and therefore Edit mode is only available for admin within the project or super-admin.



□	ID	Name	Role
□	akutsenka-gomel-iba-by	Alina Kutsenka	vf-editor
□	alemeshevskaya-gomel-iba-by	alemeshevskaya-gomel-iba-by	vf-editor
□	inavumau-gomel-iba-by	Ilya Navumau	vf-editor
□	mhaurylenka-gomel-iba-by	mhaurylenka-gomel-iba-by	vf-editor
□	mkuzeyeu-gomel-iba-by	Maksim Kuzeyeu	vf-editor

Rows per page: 5 1-5 of 8

4. Job operations

4.1. Jobs Overview

Clicking *Jobs* menu item leads to *Jobs Overview Screen* which allows you to see a list of jobs existing within a project. Some of the jobs can be used in pipelines, this is indicated by ↗ icon. Jobs Overview Screen displays the following information:

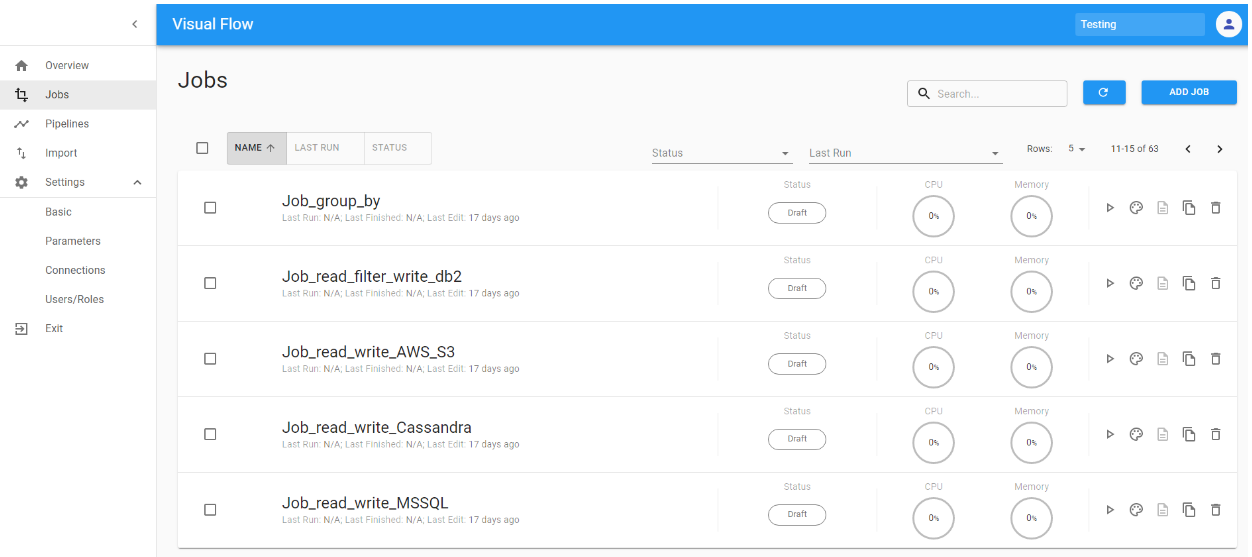
- Job Name
- Job Last run/Last finished/Last edit
- Job Status
- Resource Utilization (CPU/Memory)
- Available Actions (Run/Job Designer/Logs/Copy/Delete)

Job has a certain status at various phases of execution:

- Draft
- Pending
- Running
- Succeeded
- Failed
- Unknown (This status appears very rarely in case of undefined error)

Notes:

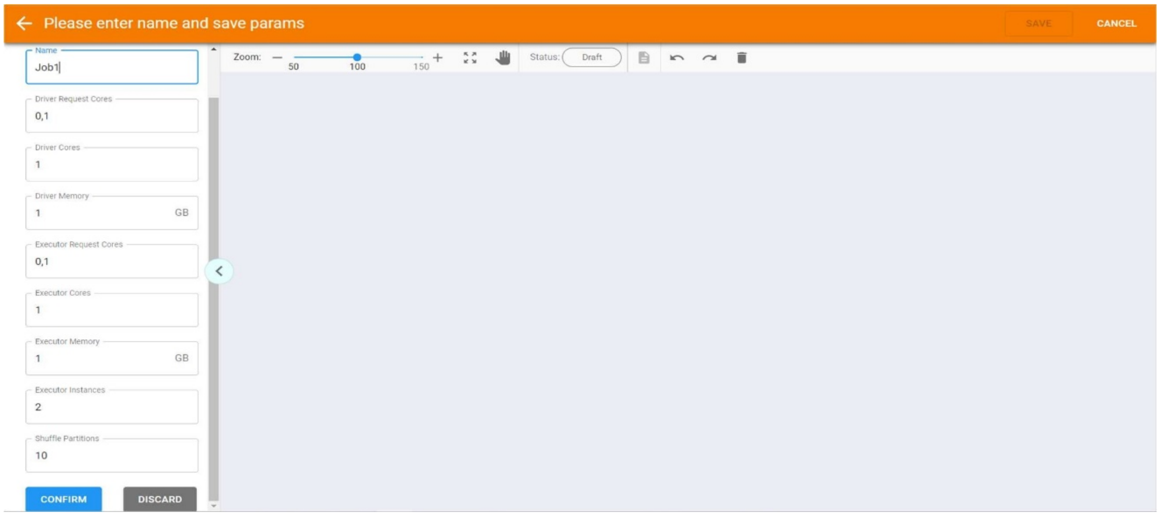
- The actions availability and therefore visibility is depending on user authorizations
- The user cannot delete a job which compounds a pipeline



	Status	CPU	Memory
☐ Job_group_by Last Run: N/A; Last Finished: N/A; Last Edit: 17 days ago	Draft	0%	0%
☐ Job_read_filter_write_db2 Last Run: N/A; Last Finished: N/A; Last Edit: 17 days ago	Draft	0%	0%
☐ Job_read_write_AWS_S3 Last Run: N/A; Last Finished: N/A; Last Edit: 17 days ago	Draft	0%	0%
☐ Job_read_write_Cassandra Last Run: N/A; Last Finished: N/A; Last Edit: 17 days ago	Draft	0%	0%
☐ Job_read_write_MSSQL Last Run: N/A; Last Finished: N/A; Last Edit: 17 days ago	Draft	0%	0%

4.2. Create a Job. Available Stages.

With *Add Job* button pushed user gets to *Job Designer* for creating a new job.



Please enter name and save params

Name: Job1

Driver Request Cores: 0,1

Driver Cores: 1

Driver Memory: 1 GB

Executor Request Cores: 0,1

Executor Cores: 1

Executor Memory: 1 GB

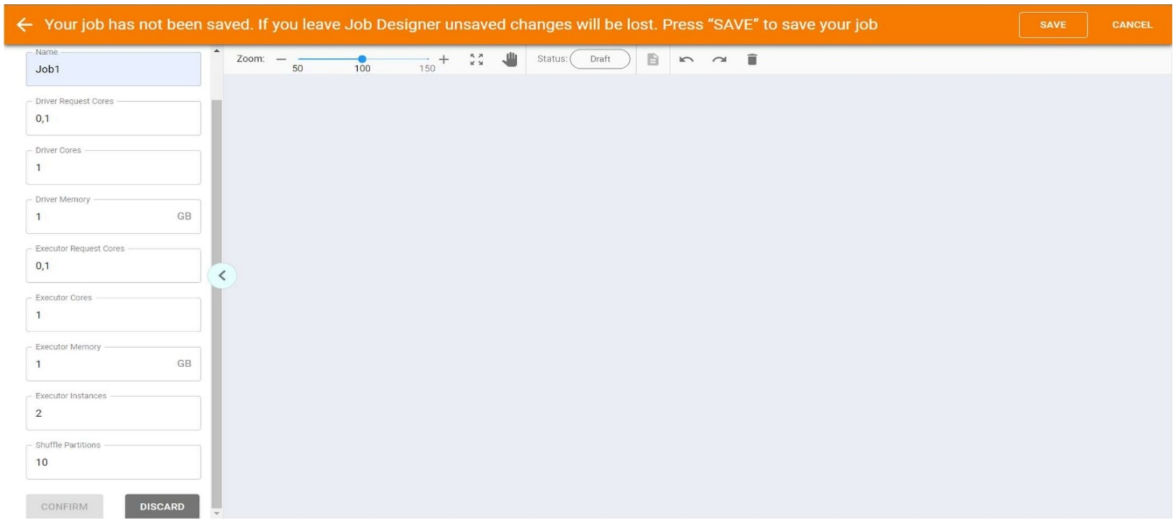
Executor Instances: 2

Shuffle Partitions: 10

Status: Draft

On the left configuration panel a name for the job must be provided.

Update parameters or keep their default values and then push *Confirm* on the panel:



Your job has not been saved. If you leave Job Designer unsaved changes will be lost. Press "SAVE" to save your job

Name: Job1

Driver Request Cores: 0,1

Driver Cores: 1

Driver Memory: 1 GB

Executor Request Cores: 0,1

Executor Cores: 1

Executor Memory: 1 GB

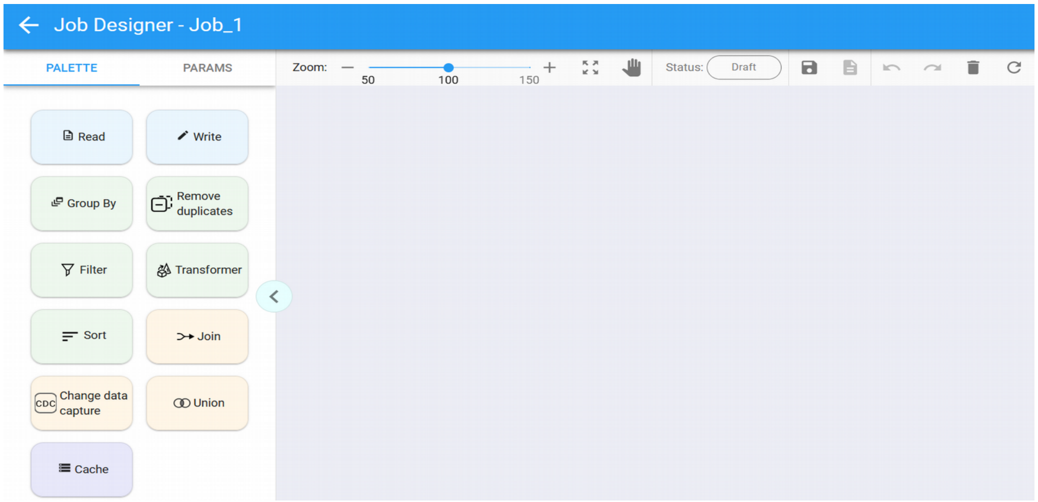
Executor Instances: 2

Shuffle Partitions: 10

Status: Draft

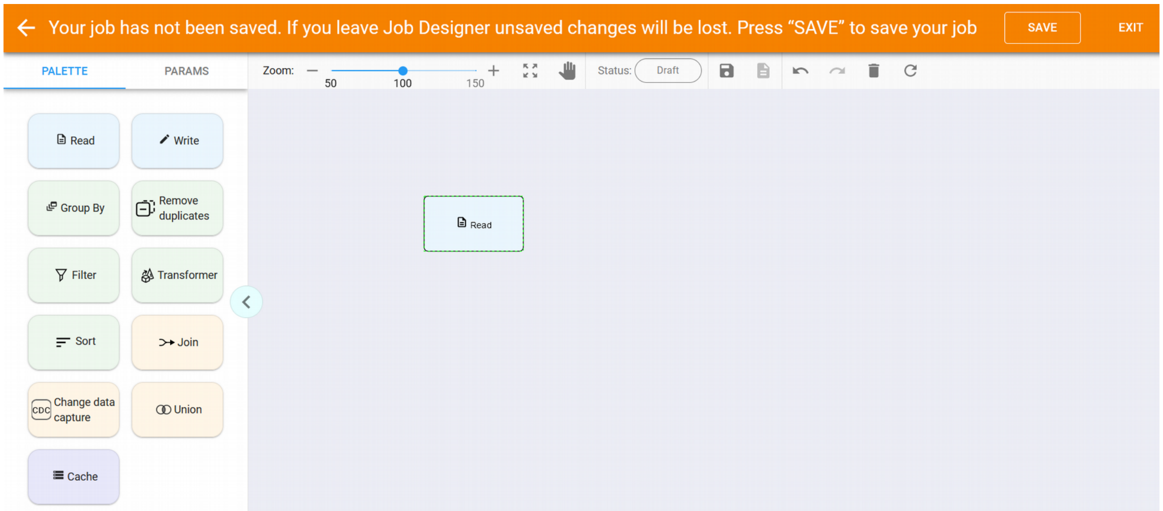
Save the job by pushing *Save* button on the *Job Designer* header.

You will see *Palette* tab then with all available stages:

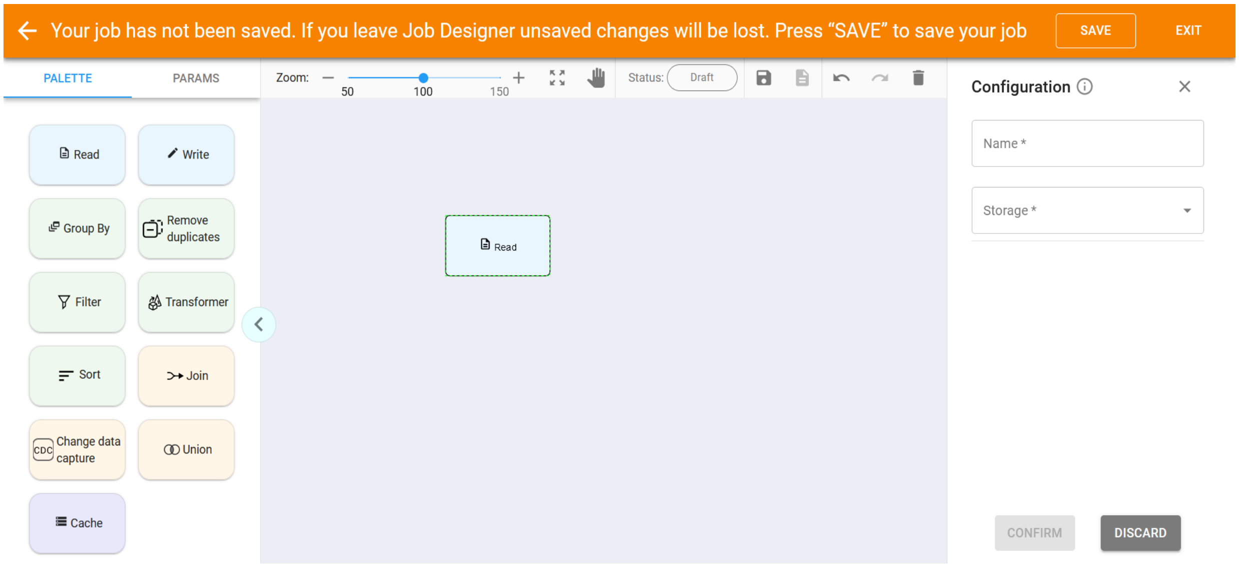


4.2.1. Read Stage.

You can start creating a job by dragging a stage to the canvas, e.g. you can drag *Read* stage:


Your job has not been saved. If you leave Job Designer unsaved changes will be lost. Press “SAVE” to save your job

Double-click on the stage opens the configuration panel on the right:



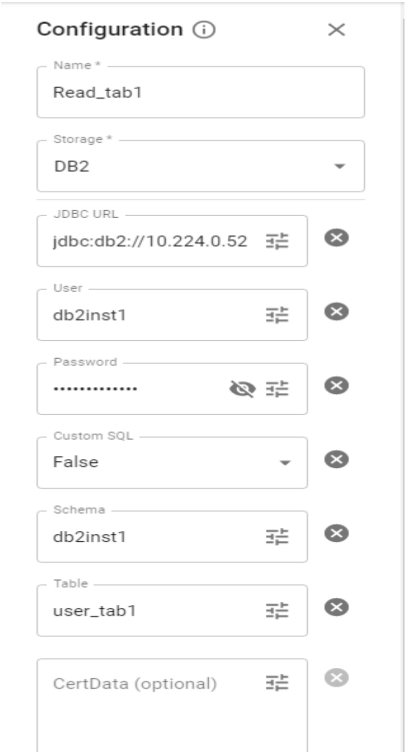
Your job has not been saved. If you leave Job Designer unsaved changes will be lost. Press “SAVE” to save your job

Enter name for the stage and select *Storage* DB2 if you want to read data from DB2 table.

Available *Storage* values for Read stage are:

- ✓ AWS S3
- ✓ DB2
- ✓ Cassandra
- ✓ Elasticsearch
- ✓ IBM COS
- ✓ Mongo
- ✓ MySQL
- ✓ MSSQL
- ✓ Oracle
- ✓ PostgreSQL
- ✓ Redis
- ✓ Redshift
- ✓ Redshift-jdbc

Fill required parameters for DB2 *Storage:*



Configuration

Name *: Read_tab1

Storage *: DB2

JDBC URL: jdbc:db2://10.224.0.52

User: db2inst1

Password: ••••••••••••

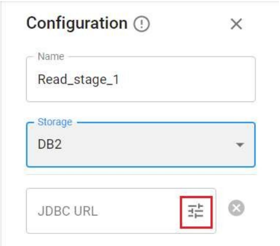
Custom SQL: False

Schema: db2inst1

Table: user_tab1

CertData (optional)

Important: you can pick up a parameter value with *Parameters* button on the right panel if it is previously created as a project parameter.



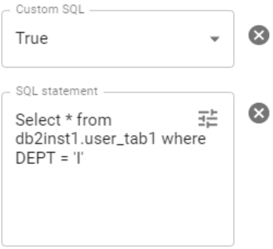
Configuration

Name: Read_stage_1

Storage: DB2

JDBC URL

Read Stage for DB2 storage has an option *Custom SQL* to read data with SQL statement (e.g. *select * from table where field = value*). If you select *Custom SQL* - *True* you need to enter the SQL statement and specify schema:



Custom SQL: True

SQL statement: Select * from db2inst1.user_tab1 where DEPT = 'I'

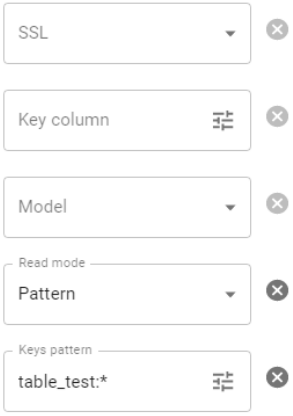
For *Redis* source you need to define *Key column*, *Model, SSL, Read mode, Keys pattern or Table* fields in the Configuration of Read stage.

✔ *Key column*. Column name to store the hash key.

✔ *Model (binary, hash)* defines the *Redis* model used to persist DataFrame. By default it is hash.

✔ *Read mode* (key, pattern) defines the way the read operation is handled. If “*key”* is selected, then the read is done based on "*table*" field. In case of "*pattern*" the provided pattern (option "keys Pattern") dictates what Redis keys will be read.

Keys pattern. If pattern ends with * (e.g., "table: *"), all keys from the pattern will be read. If one pattern is defined (e.g. "table: first value") then only one key will be read.

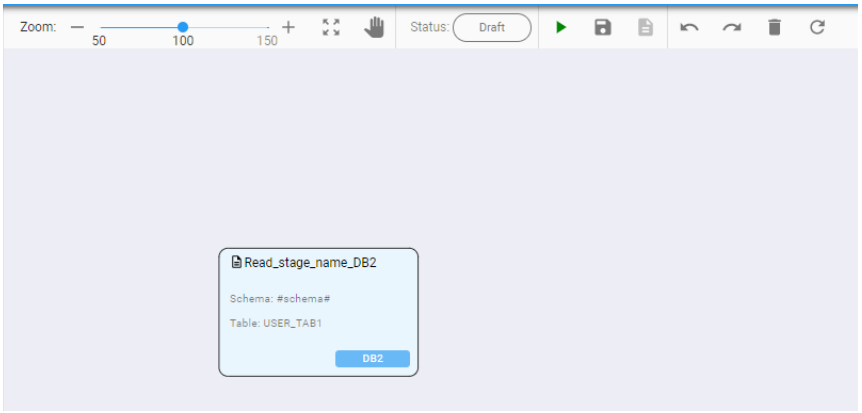


Read mode: Pattern

Keys pattern: table_test:*

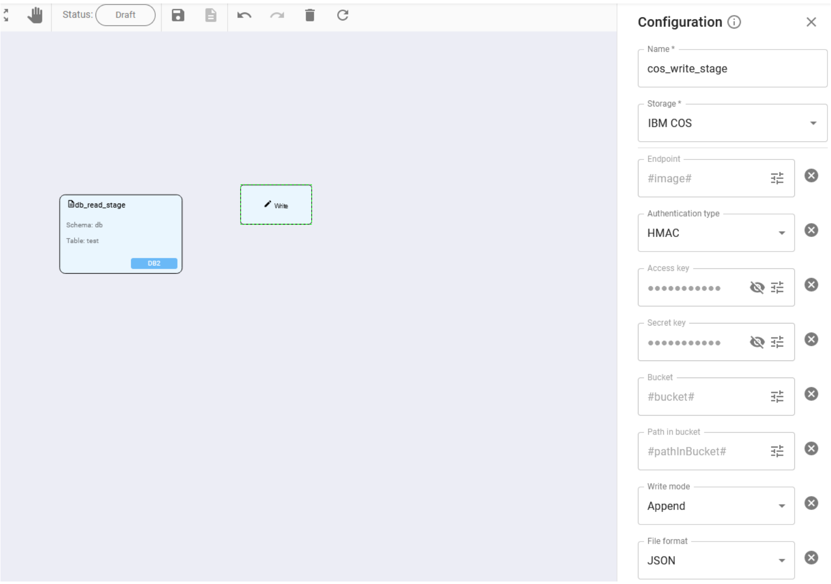
Save the stage by pushing *Confirm* button on the configuration panel. Push *Save* button on the header If you want to save the job at this step.

When the first stage of the job is configured the canvas looks like this:



4.2.2. Write Stage.

Now drag another stage, e.g. Wr*ite* stage:



Configuration

Name * cos_write_stage

Storage * IBM COS

Endpoint #image#

Authentication type HMAC

Access key ●●●●●●●●●●●

Secret key ●●●●●●●●●●●

Bucket #bucket#

Path in bucket #pathInBucket#

Write mode Append

File format JSON

Enter a name for the stage and select *Storage IBM COS* if you want to post data from DB2 table to Cloud Object Storage file. Fill required parameters for IBM COS *Storage.*

Available *Storage* values for Write stage are:

- ✓ AWS S3
- ✓ DB2
- ✓ Cassandra
- ✓ Elasticsearch
- ✓ IBM COS
- ✓ Mongo
- ✓ MSSQL
- ✓ MySQL
- ✓ Oracle
- ✓ PostgreSQL
- ✓ Redis
- ✓ Redshift
- ✓ STDOUT

IBM COS Storage has two options of *Authentication type: HMAC* and *IAM.*
If *HMAC* is selected you should fill accessKey and secretKey.
If *IAM* is selected iamApiKey and iamServiceId should be entered.

For *IBM COS* and *AWS S3* storages function *Partition By* can be used in Write stage. It partitions the output on the file system by given columns. If specified, the output is laid out on the file system similar to Hive's partitioning scheme.

As an example, when we partition a dataset by year and then month, the directory layout will look like:
- year=2016/month=01/
- year=2016/month=02/

In a case of importing table data with *Write* stage to *Cassandra* source from another storage the table layout for output must be previously created in Cassandra. Columns, key fields, data types of the fields must be specified in the table.

Important: All the above points must match the imported table.

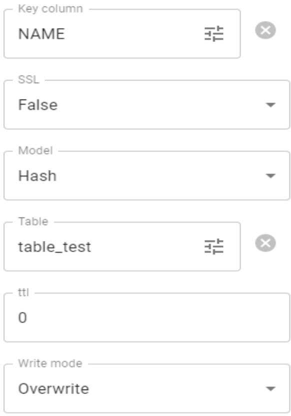
If the column names have uppercase characters in the imported table when data is output to *Cassandra*, the job will be failed as in *Cassandra* column names are stored in lowercase only. The issue can be resolved with *Transformer* stage.

The output recorded to *STDOUT* storage can be seen in Logs. The number of records to be shown in Logs can be specified in *Quantity* field. Available range is from 1 to 2147483631 records.

For *Redis* source in *Write* stage the following fields must be specified: *Key column*, *Model, SSL,*
TTL, *Table and Write mode.*

✓ *Key column*. For writing it specifies the unique column used as Redis key. By default the key is auto-generated.

✓ *TTL*. Data expiration time in seconds. Data doesn't expire if TTL is negative or 0. By default it is 0. Positive value of *TTL* means number of seconds in which the data will be removed.



Key column: NAME

SSL: False

Model: Hash

Table: table_test

ttl: 0

Write mode: Overwrite

Important:

Write mode field defines how data will be posted to its destination. Available values are:

- ✓ Overwrite
- ✓ Append
- ✓ Error if Exists

With '*Overwrite*' Write mode *Truncate mode* can be used for DB2, Oracle, MySQL, PostgreSQL, MSSQL, Redshift*:*

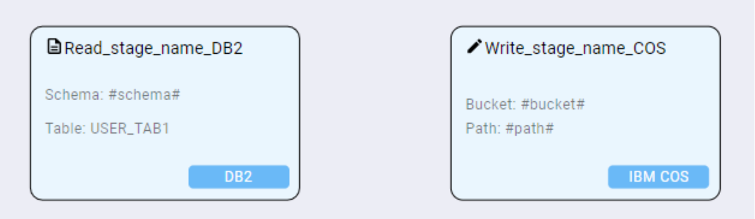
- ✓ *None*. No truncation will occur, but the target table will be deleted and recreated. Note that all the indexes, constraints, etc. defined for this table will be lost.
- ✓ *Simple*. The standard truncation that deletes data from the target table but will keep the indexes, constraints and other modifiers intact. However note that if the target table has a primary key referenced as a foreign key in other tables, the truncation will fail.

  To resolve this either use Cascade mode instead or drop constraints manually (outside of VF) prior to accessing the table with VF.
- ✓ *Cascade* (only for *Oracle* and *PostgreSQL*). The cascade truncation does not only delete the data from the target table, but also from other tables that use the target table's primary key as a foreign key constraint.

File format is to choose a format of destination file. Available formats are:

- ✓ CSV
- ✓ JSON
- ✓ Parquet
- ✓ ORC
- ✓ Text
- ✓ Avro

Confirm the stage by pushing *Confirm* on the panel.
Now there are two stages to connect to each other.

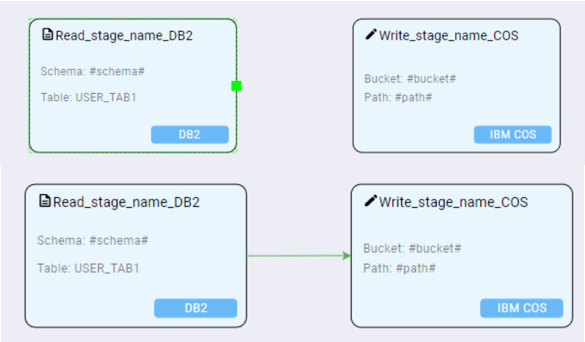


Read_stage_name_DB2
Schema: #schema#
Table: USER_TAB1

Write_stage_name_COS
Bucket: #bucket#
Path: #path#

Important:

To connect stages, hover his mouse on a stage edge until user sees a green rectangle. Click it and drag it to the border of another stage and its green rectangle. When you reach it a green arrow should appear.



Read_stage_name_DB2
Schema: #schema#
Table: USER_TAB1

Write_stage_name_COS
Bucket: #bucket#
Path: #path#

Read_stage_name_DB2
Schema: #schema#
Table: USER_TAB1

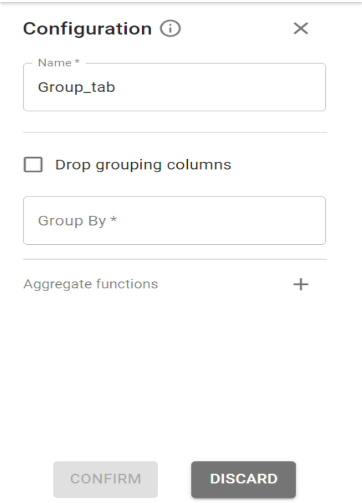
Write_stage_name_COS
Bucket: #bucket#
Path: #path#

Other stages available:

- ✓ Group By
- ✓ Remove duplicates
- ✓ Filter
- ✓ Transformer
- ✓ Sort
- ✓ Join
- ✓ Change data capture
- ✓ Union
- ✓ Cache

4.2.3. Group By Stage.

The stage allows grouping by column which must be specified in the Configuration panel. There is an option *Drop grouping columns* for removing grouping columns from the output. Also aggregate function can be added e.g., Count, Avg etc.



Configuration

Name * Group_tab

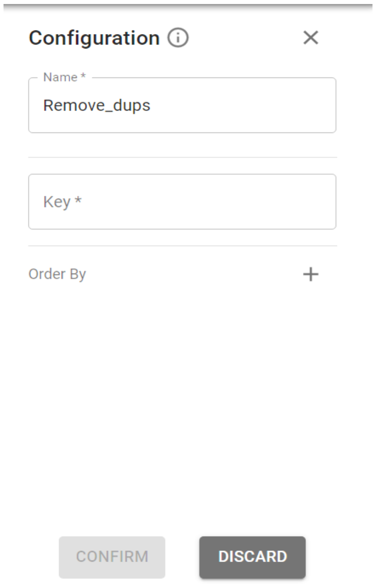
□ Drop grouping columns

Group By *

Aggregate functions

4.2.4. Remove Duplicates Stage.

Specify a key column for the operation. To specify more than one key, use comma or Enter. For Order By operation you need to specify column to sort by and sort order Asc or Desc. Default is Asc.



Configuration

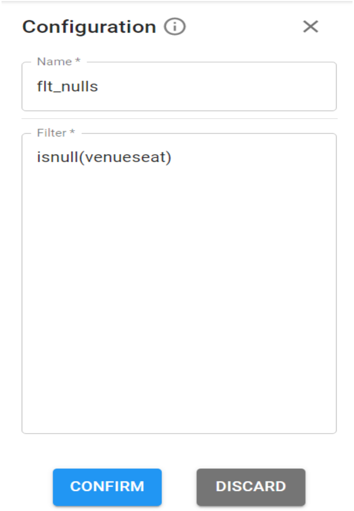
Name * Remove_dups

Key *

Order By

4.2.5. Filter Stage.

Enter any boolean expression. Two or more expressions can be combined using logical operators (AND, OR).
Examples: (column1 < 10) and (column2 between 10 and 25).



Name *
flt_nulls

Filter *
isnull(venueseat)

4.2.6. Transformer Stage.

Transformer stage gives you ability to modify columns that will be written to some data storage later.
Transformer mode defines the type of the SQL query (Spark SQL dialect) that is accepted and executed.

Simple mode only allows you to specify the part between SELECT and FROM. You can do things like these:

- col1, concat(col1, col2)
- count(*) as count
- a, b, row_number() OVER (PARTITION BY a ORDER BY b)
- col, exists(col, x -> x % 2 == 0)
- col, collect_list(col)

Syntax: <column_name_1> as <alias_1>, function(<column_name_2>) as <alias_2>

Full SQL mode allows you to write a full-blown Spark SQL query. In this case you have to specify table name manually or reference a table name from parameter.

Table name. The name of the table that you should use within the Spark SQL query. Applicable for *Full SQL* transformer mode only.

4.2.7. Sort Stage.

There are two types of sorting available: *Full sort* or *Sort within partitions*.

Full sort sorts DataFrame by the specified column(s).

Sort within partitions sorts each DataFrame partition by the specified column(s). In this case the order of the output data is not guaranteed because the data is ordered at partition level.
Select column(s) to sort by and sort order (default value is Asc).

Available sort options:

- asc
- asc nulls first
- asc nulls last
- desc
- desc nulls first
- desc nulls last

4.2.8. Join Stage.

These are available types of join:

- *Inner* join. Transfers records from input data sets whose key columns contain equal values to the output data set. Records whose key columns do not contain equal values are dropped.
- *Left outer* join. Transfers all values from the left data set but transfers values from the right data set only where key columns match. The stage drops the key column from the right data set.
- *Right outer* join. Transfers all values from the right data set and transfers values from the left data set and intermediate data sets only where key columns match. The stage drops the key column from the left and intermediate data sets.
- *Full outer* join. Transfers records in which the contents of the key columns are equal from the left and right input data sets to the output data set. It also transfers records whose key columns contain unequal values from both input data sets to the output data set.
- *Cross* join. Returns a result data set where each row from the first table is combined with each row from the second table.

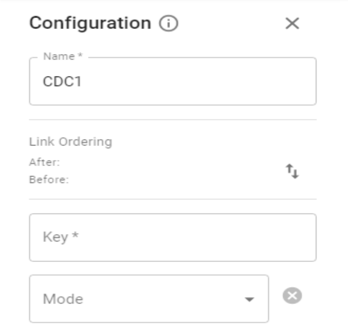
Left semi join. Returns values from the left side of the relation that has a match with the right.*Left anti* join. Returns values from the left relation that has no match with the right.

Link Ordering option allows you to specify which input link is regarded as the left link and which link is regarded as the right link. By default, the first link added is regarded as the left link, and the last one as the right link.

4.2.9. Change Data Capture Stage.

This stage is intended to find all differences between before (old) and after (new) datasets. Based on differences, CDC produces an additional column 'Operation', which indicates the state of the row from the old dataset considering it's presence/absence in the new one. CDC compares each row of the new and the old datasets based on key and columns to compare values and sets Operation value.

Note: old and new datasets must not contain duplicates (rows with the same key) based on key column(s). Old and new datasets columns to compare and key columns must be presented in both datasets with the same names. If there are duplicated rows at least in one of the dataset, results of the CDC will be unpredictable.



Configuration

Name *
CDC1

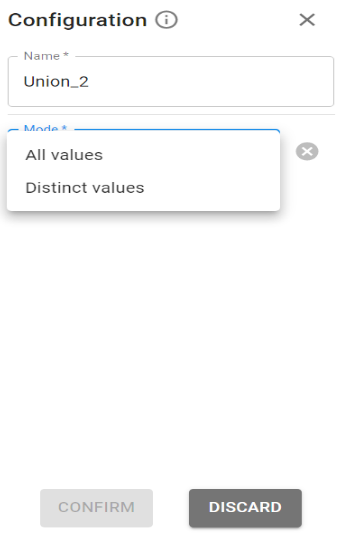
Link Ordering
After:
Before:

Key *

Mode

4.2.10. Union Stage.

You can union two datasets.
Note: Column's sequence, names, types are important for union operation.



Configuration

Name *
Union_2

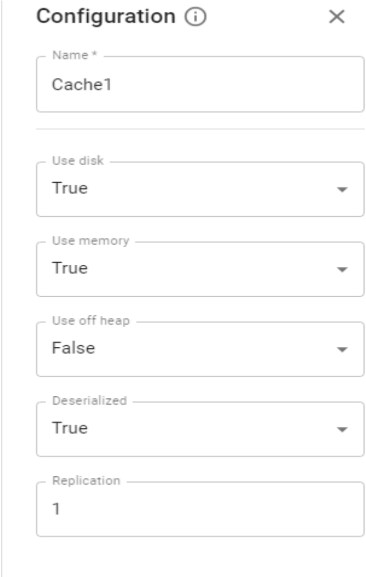
Mode *

Mode contains 2 options: *All values* and *Distinct values*.
Distinct values which is default will eliminate duplicate records from the second dataset.
All values needs to be specified explicitly, and it tolerates duplicates from the second dataset.

4.2.11. Cache Stage.

Cache stage persists dataset in some storage. The storage type can be tweaked by specifying/combining parameters. The configuration gives you ability to define:

- ✔ Whether to use memory.
- ✔ Whether to drop the RDD to disk if it falls out of memory.
- ✔ Whether to keep the data in memory in a serialized format.
- ✔ Whether to replicate the RDD partitions on multiple nodes.



Configuration ⓘ ✕

Name *: Cache1

Use disk: True

Use memory: True

Use off heap: False

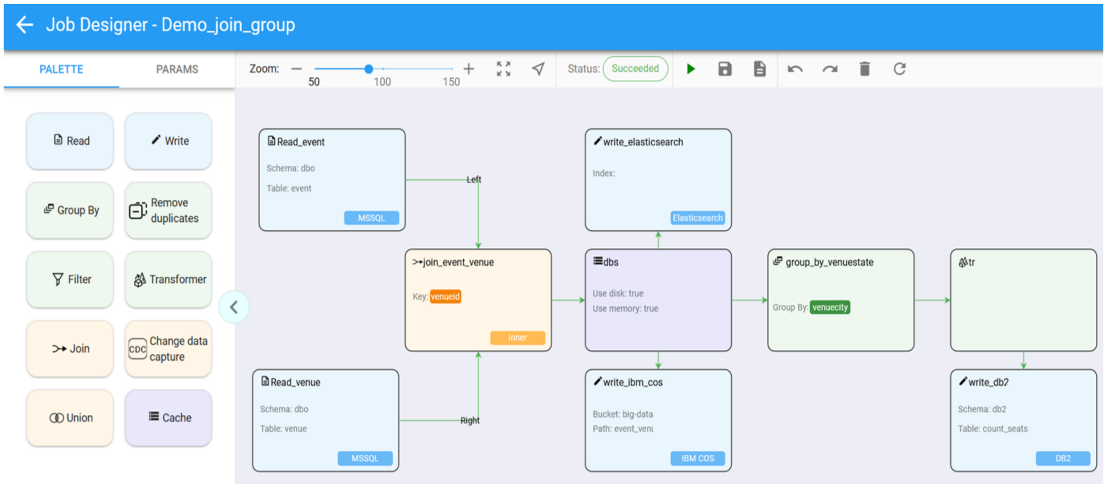
Deserialized: True

Replication: 1

Save the job by pushing *Save* on the *Job Designer* header.

For newly created job as long as it is not yet run its status is *Draft*: Status: Draft

Drag other stages according to the flow of user job from source to destination. See the job with more stages as example:



Job Designer - Demo_join_group

Zoom: 50 100 150 — Status: Succeeded

Read_event — Schema: dbo — Table: event — MSSQL

Read_venue — Schema: dbo — Table: venue — MSSQL

join_event_venue — Key: venueid — inner

dbs — Use disk: true — Use memory: true

write_elasticsearch — Index: — Elasticsearch

write_ibm_cos — Bucket: big-data — Path: event_venı — IBM COS

group_by_venuestate — Group By: venuecity

tr

write_db2 — Schema: db2 — Table: count_seats — DB2

4.3. Job Designer functions overview

The following functions are available in *Job Designer*:

- ✓ Zoom operations: Zoom: — 50 100 150 +
- ✓ Show job status: Status: Succeeded
- ✓ Run job ▶ / Stop job ■ (for running)

✓ Save job

✓ See job logs

✓ Undo / Redo operation on canvas

✓ Remove element from canvas

✓ Refresh

4.4. Job Execution

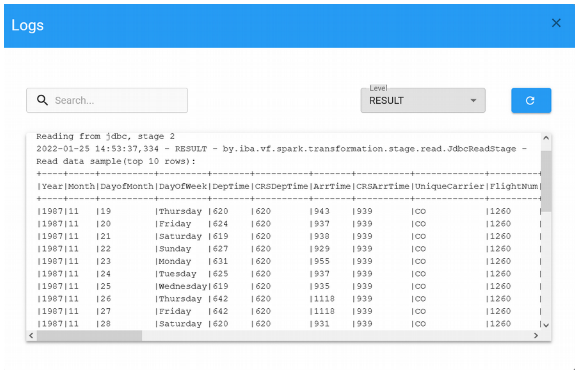
Push *Play* button ▶ to run the job:

Its status changes from *Draft* to *Pending* Status: Pending

Push Refresh to update the status. It should turn to *Running* Status: Running

While running, it can be interrupted with *Stop* button. ■ When job completed its status is *Succeeded* or *Failed*

Use *Logs* button to analyze job logs. It gets you to *Logs Screen*:



```
Reading from jdbc, stage 2
2022-01-25 14:53:37,334 - RESULT - by.iba.vf.spark.transformation.stage.read.JdbcReadStage -
Read data sample(top 10 rows):
+----+-----+----------+---------+-------+----------+-------+----------+-------------+---------
|Year|Month|DayofMonth|DayOfWeek|DepTime|CRSDepTime|ArrTime|CRSArrTime|UniqueCarrier|FlightNum|
+----+-----+----------+---------+-------+----------+-------+----------+-------------+---------
|1987|11   |19        |Thursday |620    |620       |943    |939       |CO           |1260     |
|1987|11   |20        |Friday   |624    |620       |937    |939       |CO           |1260     |
|1987|11   |21        |Saturday |619    |620       |938    |939       |CO           |1260     |
|1987|11   |22        |Sunday   |627    |620       |929    |939       |CO           |1260     |
|1987|11   |23        |Monday   |631    |620       |955    |939       |CO           |1260     |
|1987|11   |24        |Tuesday  |625    |620       |937    |939       |CO           |1260     |
|1987|11   |25        |Wednesday|619    |620       |935    |939       |CO           |1260     |
|1987|11   |26        |Thursday |642    |620       |1118   |939       |CO           |1260     |
|1987|11   |27        |Friday   |642    |620       |1118   |939       |CO           |1260     |
|1987|11   |28        |Saturday |620    |620       |931    |939       |CO           |1260     |
```

Logs Screen has several levels:

✓ WARNING
✓ INFO
✓ ERROR
✓ DEBUG
✓ RESULT

5. Pipeline Operations

5.1. Pipelines Overview

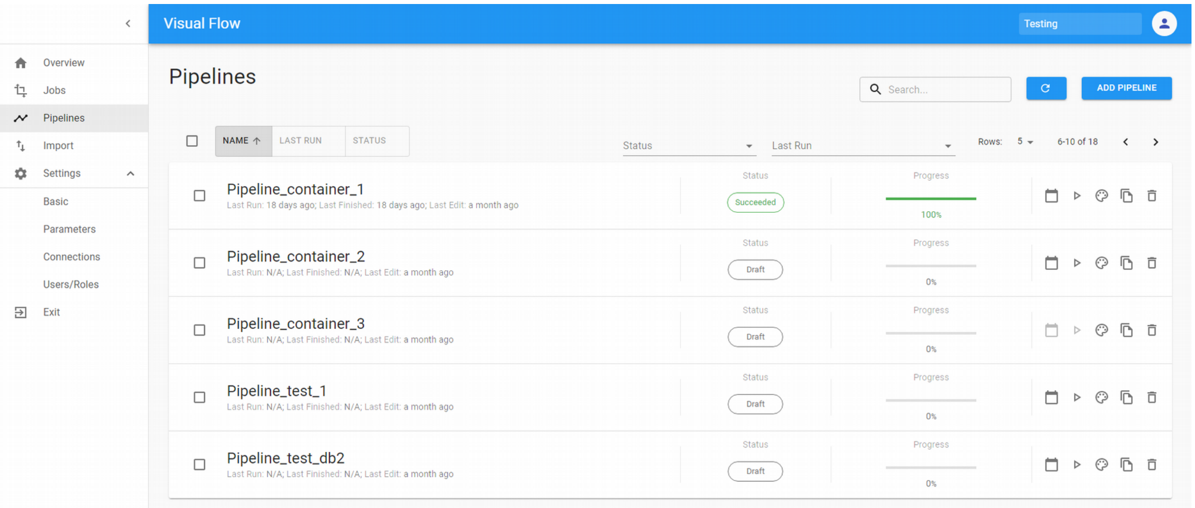
Clicking *Pipelines* menu item takes you to *Pipelines Overview Screen* which shows a list of pipelines existing within a project.

It displays the following information:

- Pipeline Name
- Checkbox for deleting/exporting multiple pipelines
- Pipeline Last run/Last finished/Last edit
- Pipeline Status
- Pipeline Progress
- Available Actions (Cron Scheduling/Run/Pipeline Designer/Copy/Delete)

Pipeline has a certain status at various phases of execution:

- Draft
- Error (This status appears e.g. due to incorrectly entered data)
- Failed
- Pending
- Running
- Stopped
- Succeeded
- Suspended (This status can be reproduced via the API)
- Terminated



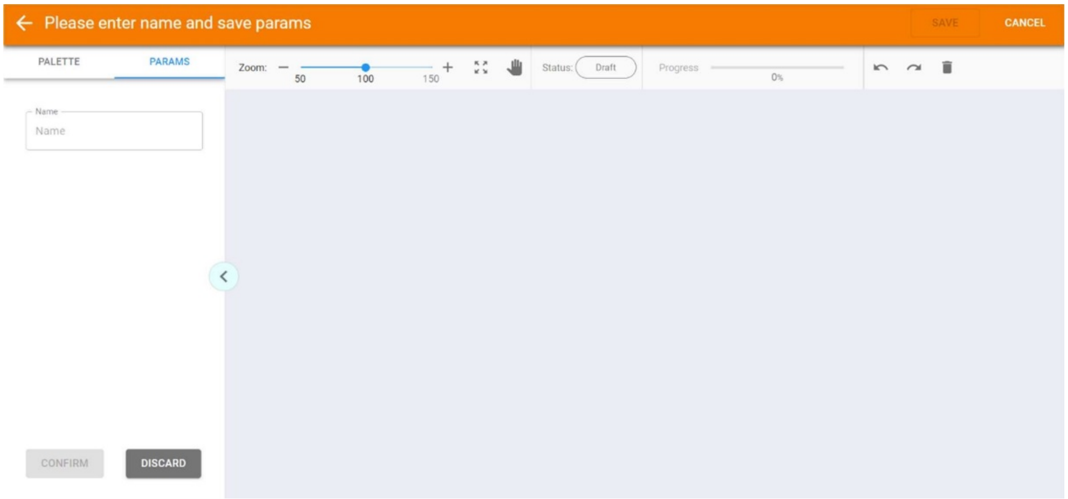
	Name	Status	Progress
□	Pipeline_container_1 Last Run: 18 days ago; Last Finished: 18 days ago; Last Edit: a month ago	Succeeded	100%
□	Pipeline_container_2 Last Run: N/A; Last Finished: N/A; Last Edit: a month ago	Draft	0%
□	Pipeline_container_3 Last Run: N/A; Last Finished: N/A; Last Edit: a month ago	Draft	0%
□	Pipeline_test_1 Last Run: N/A; Last Finished: N/A; Last Edit: a month ago	Draft	0%
□	Pipeline_test_db2 Last Run: N/A; Last Finished: N/A; Last Edit: a month ago	Draft	0%

Note: the actions availability and therefore visibility is depending on user authorizations.

5.2. Create a Pipeline

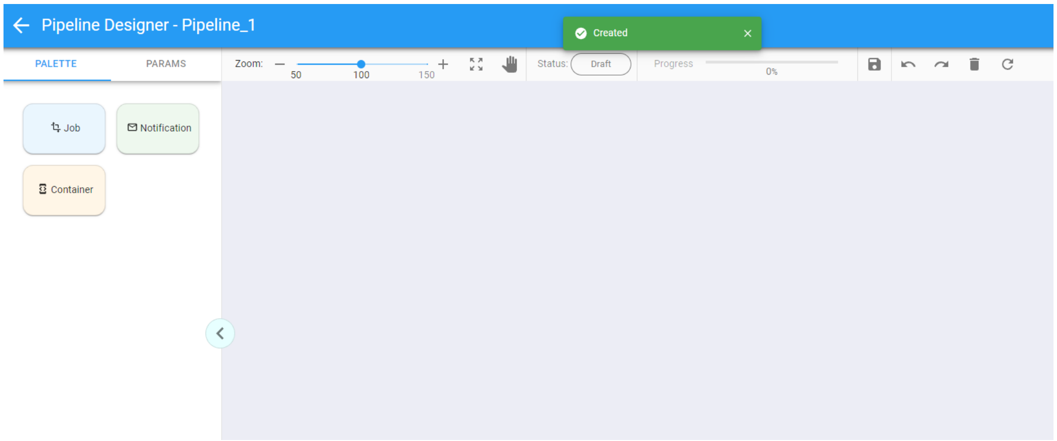
With *Add Pipeline* button pushed you get to *Pipeline Designer* for creating a pipeline.

1) On the left configuration panel *Params* tab is opened by default, where you can enter pipeline name and push *Confirm* button on the panel:


Please enter name and save params

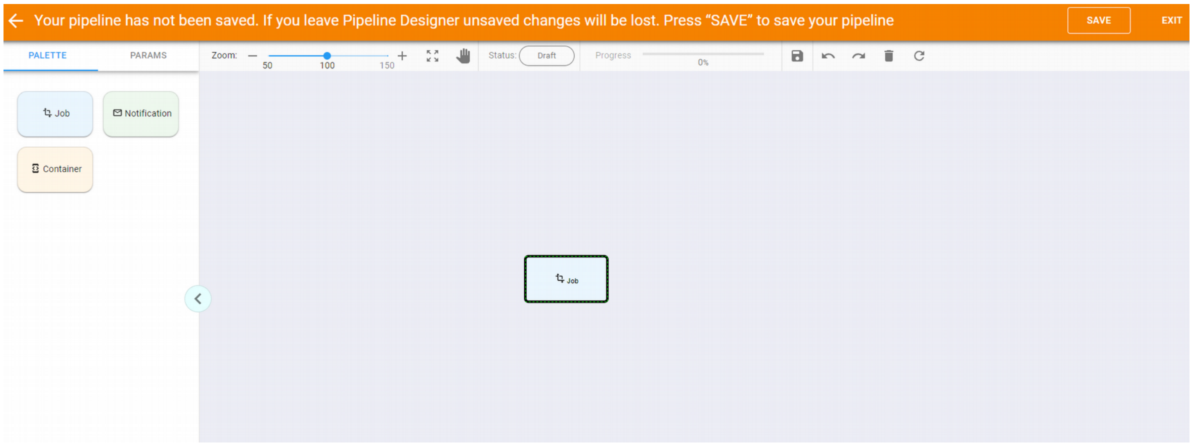
2) Save the pipeline by pushing *Save* button on the *Pipeline Designer* header.

3) After saving the pipeline Palette tab with all available stages is opened by default:

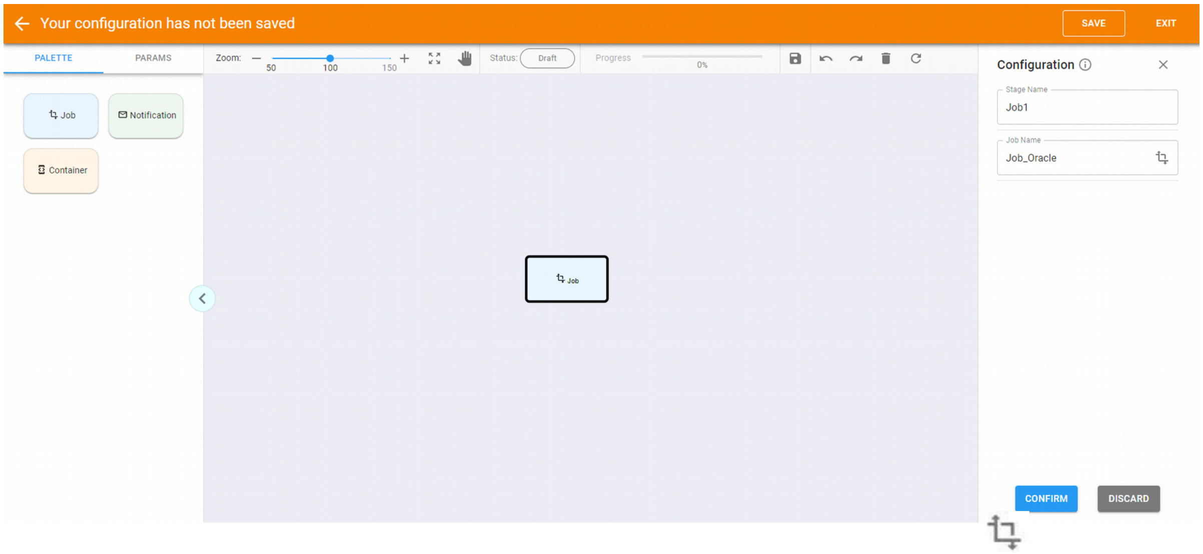

Pipeline Designer - Pipeline_1

Created

4) Pipeline is a combination of existing jobs stages and/or notification stages and/or container stages. Notification stage is most often added to configuration in the case of job stage failure/success. Start creating a pipeline by dragging *Job* stage to the canvas:


Your pipeline has not been saved. If you leave Pipeline Designer unsaved changes will be lost. Press "SAVE" to save your pipeline

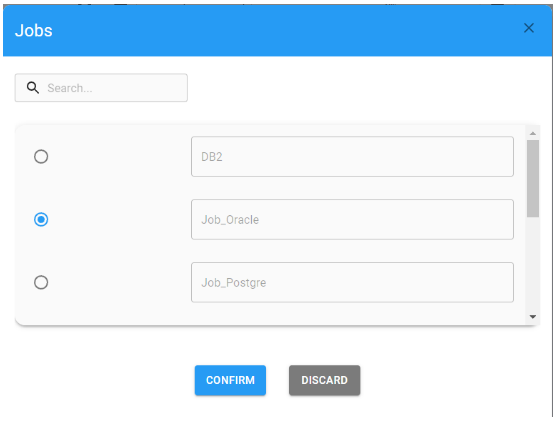
5) Double-click on the stage opens the configuration panel on the right:



Stage Name: Job1

Job Name: Job_Oracle

Enter a name for the stage and select a job from the list and push *Confirm* button



○ DB2

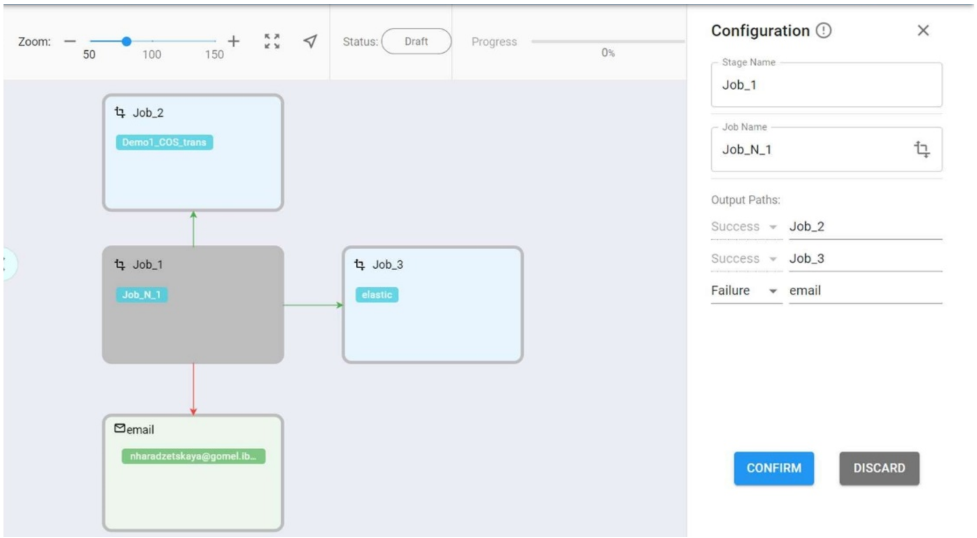
● Job_Oracle

○ Job_Postgre

6) Save the stage by pushing *Confirm* button on the panel. Push *Save* button on header If you want to save the pipeline at this step.

7) Drag and configure other stages. Connect them with the same manner as in Job Designer.

You can link stages based on success or failure of each stage. After connecting stages to each other you can choose Success or Failure link on configuration panel. There can be only one connection for failure. See the example of configured pipeline:



Configuration

Stage Name: Job_1

Job Name: Job_N_1

Output Paths:

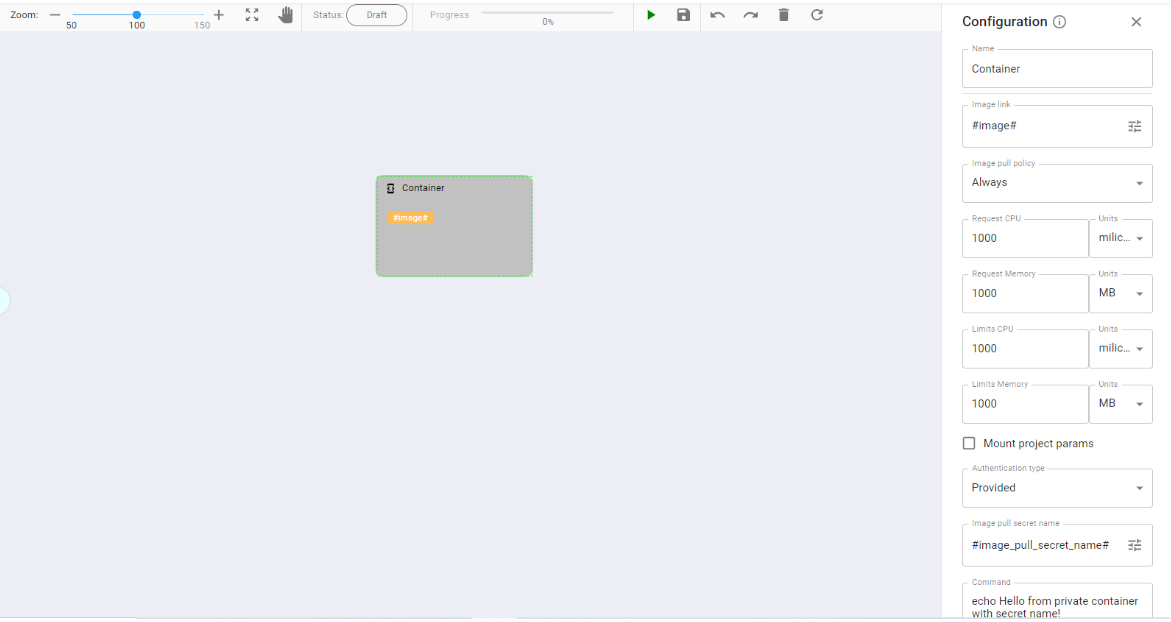
Success – Job_2

Success – Job_3

Failure – email

A *Custom container* stage is required to run custom commands to execute any logic in a pipeline. You can use docker image instead of custom commands.

1) Start creating a pipeline by dragging *Container* stage to the canvas and enter parameters in the Configuration panel:



Configuration

Name: Container

Image link: #image#

Image pull policy: Always

Request CPU: 1000 Units: milic...

Request Memory: 1000 Units: MB

Limits CPU: 1000 Units: milic...

Limits Memory: 1000 Units: MB

☐ Mount project params

Authentication type: Provided

Image pull secret name: #image_pull_secret_name#

Command: echo Hello from private container with secret name!

Container stage has the following fields in the Configuration:

✓ *Image link*. Docker image path.

Examples:

mysql,

mysql:latest,

bitnami/argo-cd:2.1.2,

localhost:5000/bitnami/argo-cd:2.1.2,

registry.redhat.io/rhel7:latest.

✓ *Image pull policy*. Defines when the image will be pulled (downloaded). Possible values:

- *If not present* – is downloaded only if it does not exist locally;
- *Always* – is downloaded before each start;
- *Never* – is not downloaded, local copy is used.

✓ *Requests and Limits CPU*

✓ *Requests and Limits memory*

✓ *Mount project parameters.* Defines whether to mount all project parameters as environment variables inside the Pod.

✓ *Authentication type*

✓ *Authentication mode* can be one of these:

- *Not applicable* : image pull secrets are not required as the image is pulled from the public registry;
- *New:* create a new image pull secret on the fly by providing all necessary information;
- *Provided* : use existing image pull secret by providing its name (Image pull secret name).

✓ *Image pull secret name*. Name of the secret to pull the image. Note that it must exist within the same k8s namespace as the current pipeline.

✓ *Username*

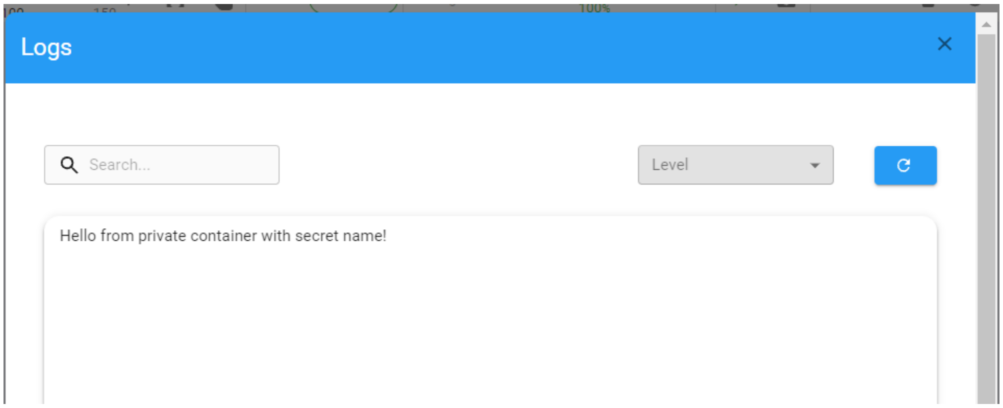
✓ *Password*

✓ *Registry.* Name of the registry for authentication.

✓ *Command*. Command that will be executed once Pod is created.

Important:

Container stage has a *Logs* button 📄 leading to Logs window.
If the pipeline completed successfully the logs display the message contained in *Command* field in the configuration of the *Container* stage.



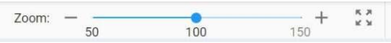
Hello from private container with secret name!

Before the first run or after updating the status of the pipeline is *Draft* Status: Draft . See each stage border painted in ***Grey*** color, which stands for *Draft.*

5.3. Pipeline Designer Functions Overview

The following functions are available in *Pipeline Designer*:

✓ Zoom func

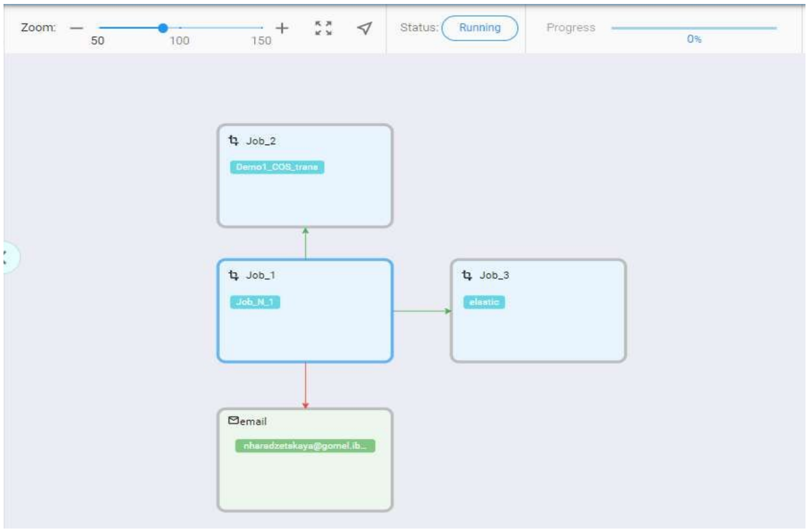


- ✓ Move elements: ➤
- ✓ Move elements/screen: ✋
- ✓ Show pipeline status: Status: Succeeded
- ✓ Show pipeline progress: Progress 100%
- ✓ Run pipeline ▶ / Stop pipeline ■ (for running)
- ✓ Save pipeline 💾
- ✓ Undo / Redo operation on canvas ↶ ↷
- ✓ Remove element from canvas 🗑
- ✓ Create cron schedule for pipeline 📅
- ✓ Refresh ⟳

5.4. Pipeline Execution

If you run a pipeline e.g. from the above example its status will change from *Draft* to *Pending* and then to *Running.* Push Refresh to update the status.

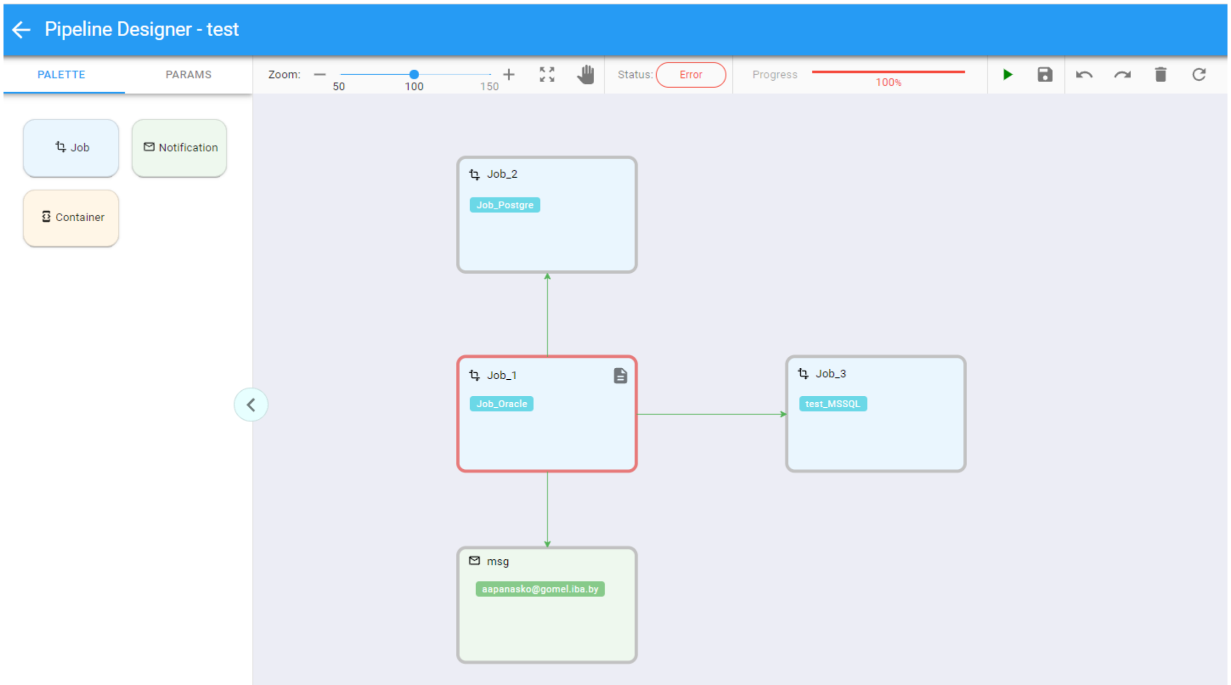
The border of the stage currently running will be painted in *Blue*:



If a pipeline succeeded, all completed stages are painted in *Green* indicating success.

The stages configured for failure scenario (red arrow) will remain *Grey* as *Draft* as they have not been executed.

If a pipeline fails, then *Red* border will indicate the failed stage:



Failed pipeline can be re-run from the point of failure with button Error ↻ located on the Pipelines Overview Screen.

Important:

Job stage has a *Logs* button 🗎 for analyzing logs of a certain job.