Visual Flow
User Guide

March 2022

Version 0.9.15

Document Revisions

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

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, various stages user can use are:

- Read
- Write
- Join
- Union
- Filter
- 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 an 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
  - ✓ Elastic Search
  - ✓ 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.

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
- ✓ Run custom code
- ✓ 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.

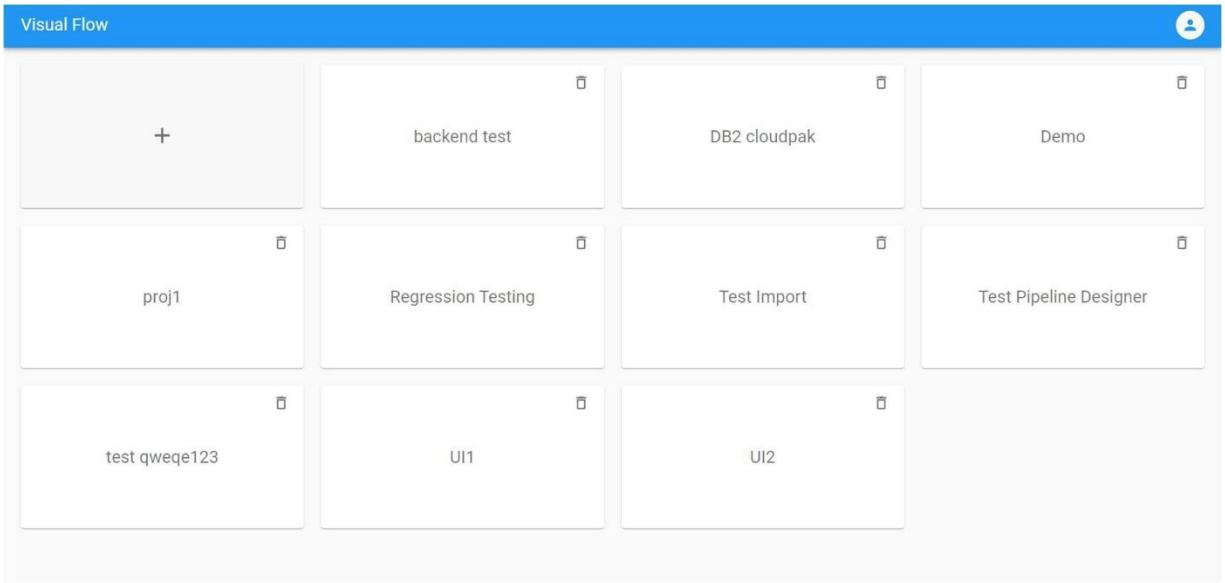
Role	Actions		
	Project Settings	Jobs	Pipelines
Viewer	View all	View all	View all
Operator	View all	View all / execute jobs	View all / execute pipelines
Editor	Edit all but Users and Roles	Edit / execute jobs	Edit / execute pipelines
Admin	Edit all	Edit / execute jobs	Edit / execute pipelines

3. Project operations

3.1. Create a Project

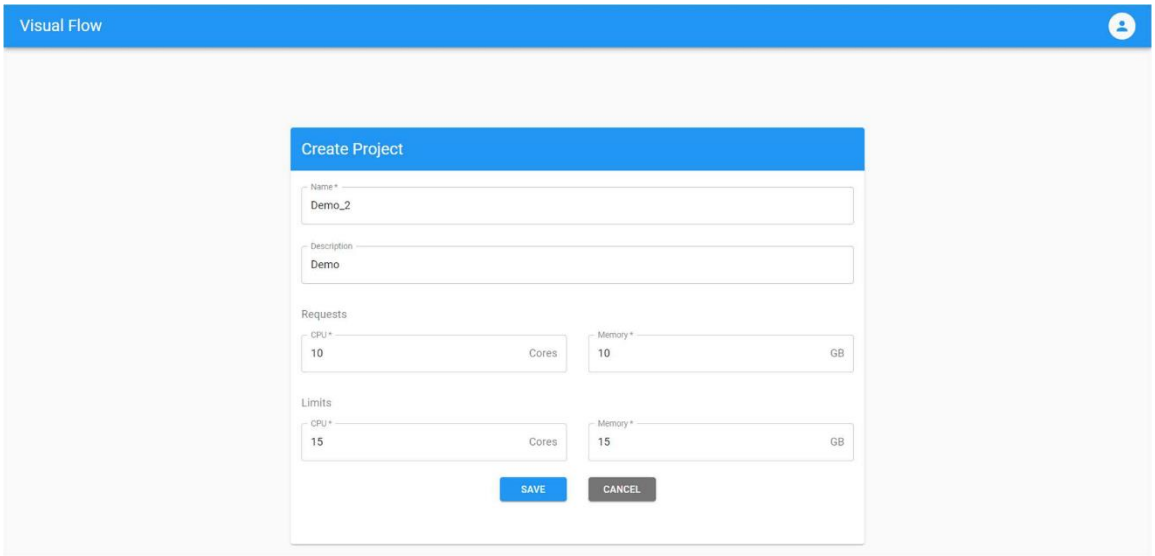
To create a project, user needs to push "+" button on the initial screen.

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

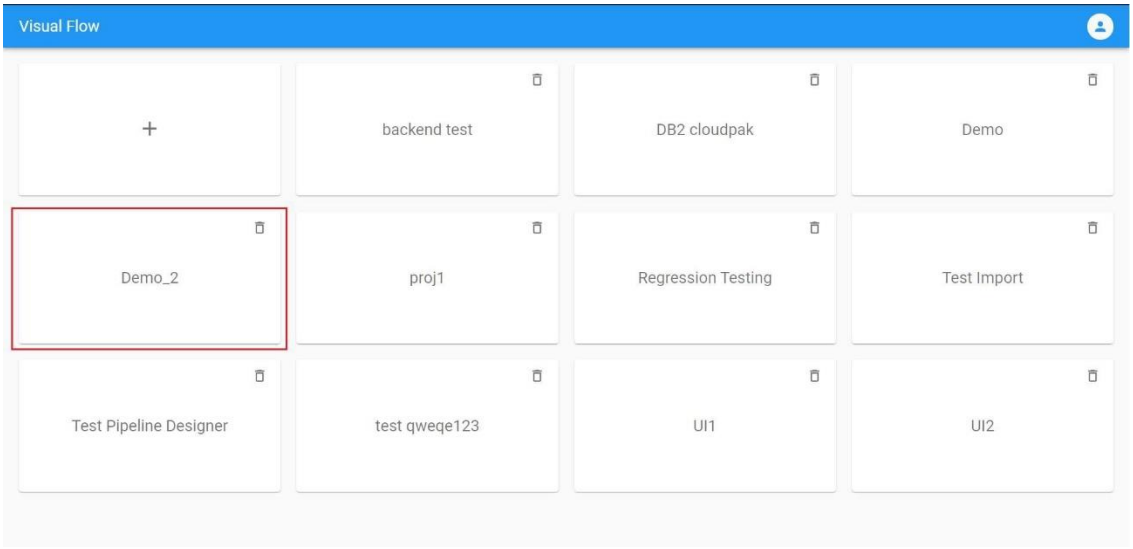


With "*+*" button pushed, user will get to *Create Project Form* to enter project basic settings:

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

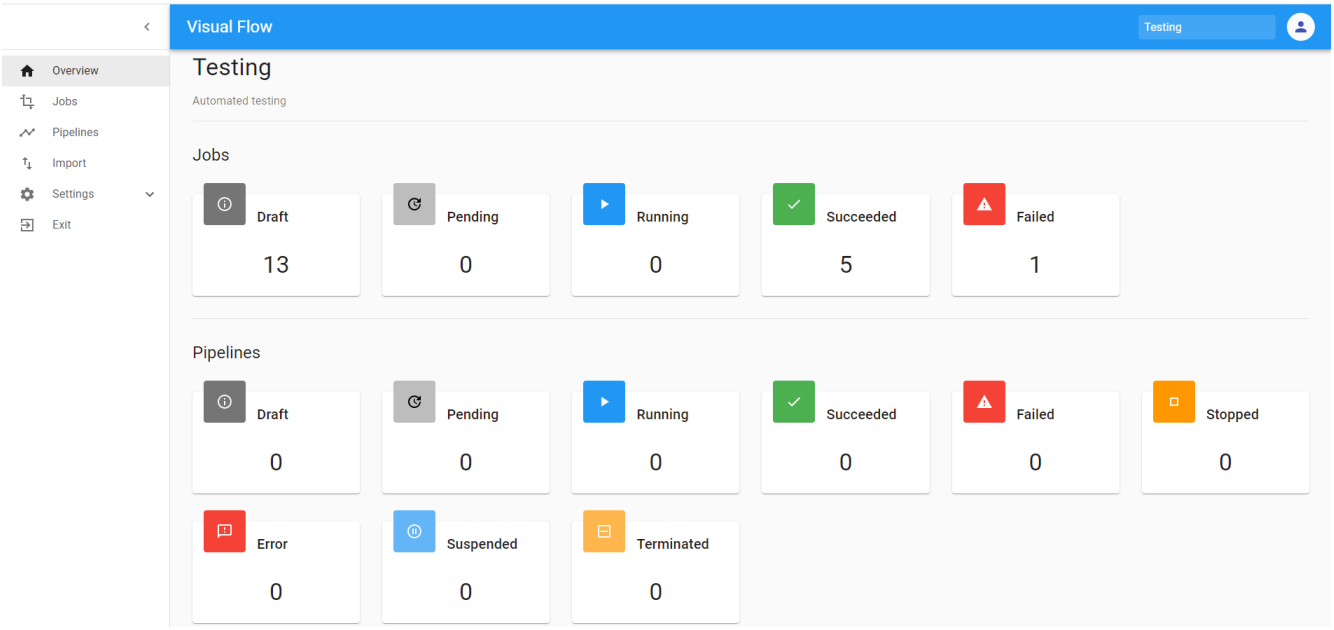


After saving *Create Project Form,* the project created under the given name and then can be found on the initial screen:



3.2. Project Overview

Click the project card to enter the newly created project, and user will get to the *ProjectOverview Screen*:



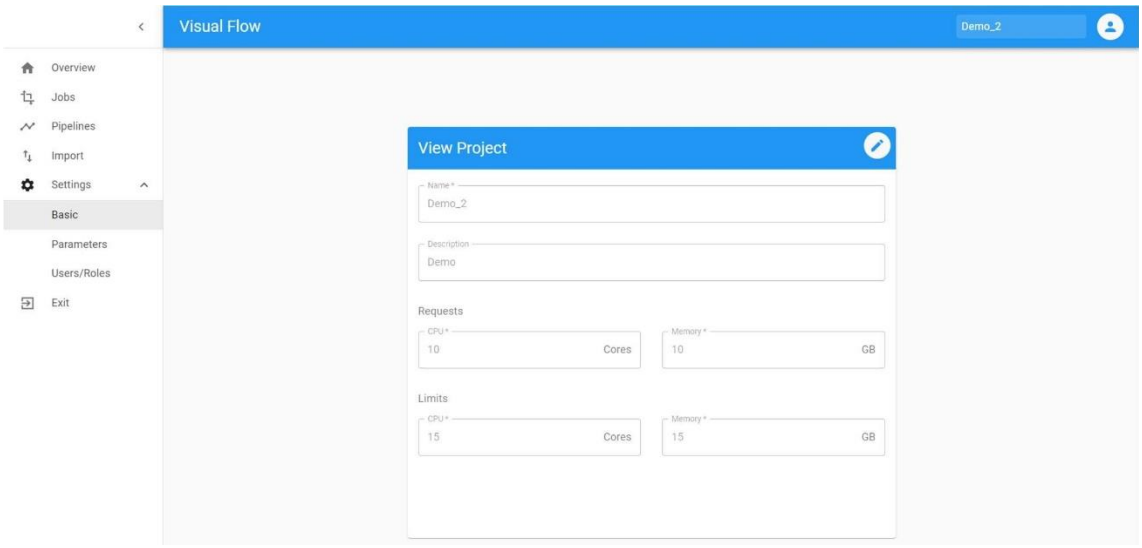
The screen contains project left menu and displays information about the project jobs, pipelines and their resource utilization (applicable for running jobs).

3.3. Manage Project Settings

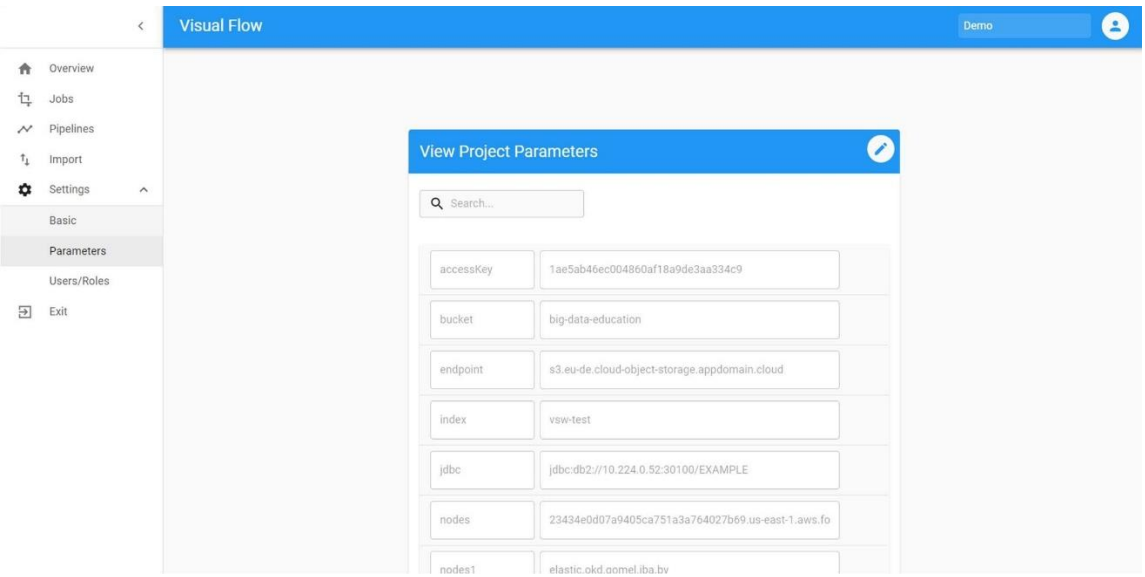
Settings submenu contains:

- Basic
- Parameters
- Users and Roles

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



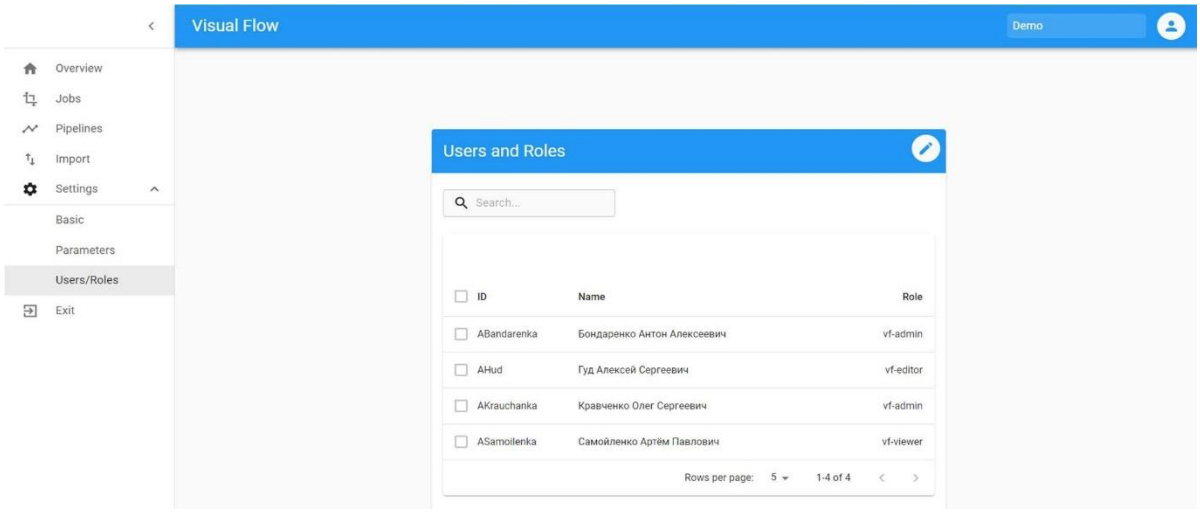
Parameters serve to store values required for the entire project, e.g. JDBC connection, DB2 credentials or table schemas can be the same for all jobs within the project and therefore stored at the project level. *Edit* button turns on the edit mode for updates.



2) *User and Roles* allows user access management or view user access depending onauthorization.

The user cannot change his role, this operation can be done by an Admin or a Super-admin. If the user tries to change his role, the error will occur «You cannot change your role".

Edit button and therefore Edit mode is only available for admin within the project or super-admin.



4. Job Operations

4.1. Jobs Overview

Clicking *Jobs* menu item will lead user to *Jobs Overview Screen,* which allows user to see a list of jobs existing within a project. Some of the jobs can be used in pipelines, this is indicated by the icon.

Jobs Overview Screen displays the following information:

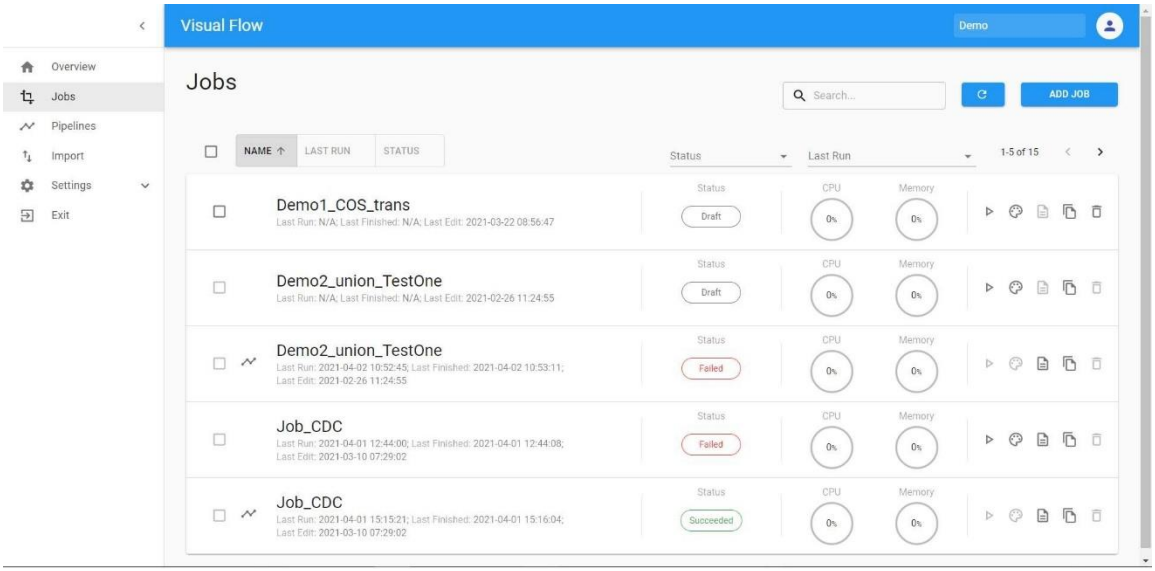
- Job Name
- Job Last run/Last finished/Last edit
- Resource Utilization (CPU/Memory)
- Available Actions (Run/Job Designer/Logs/Copy/Delete)

Job has a certain status at various phases of execution:

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

Notes:

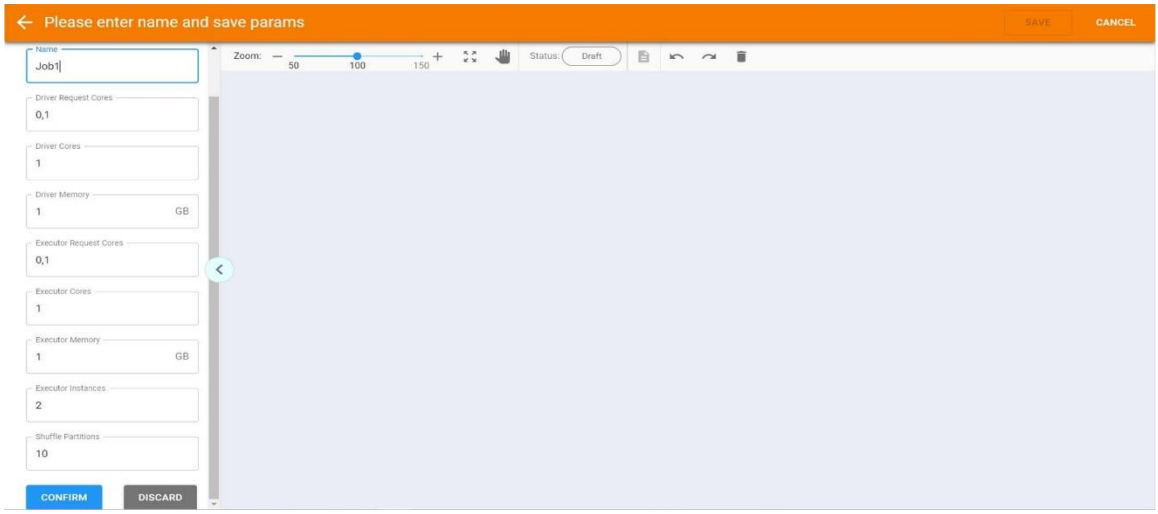
- The actions availability and therefore visibility is depending on user authorizations
- The user cannot delete job that is used in pipeline

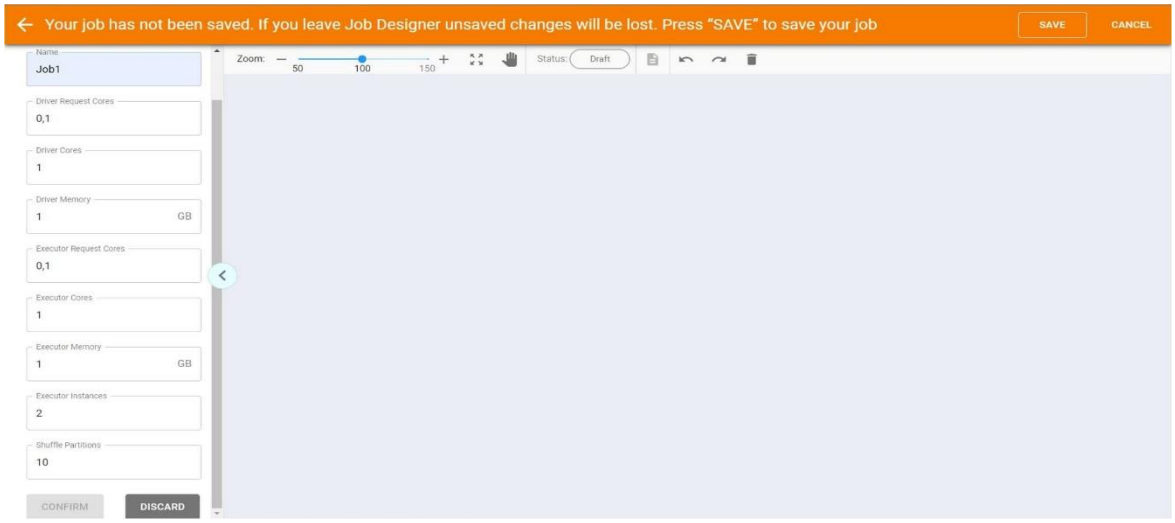


4.2. Create a Job

With *Add Job* button pushed, user will get to *Job Designer* for creating a new job.

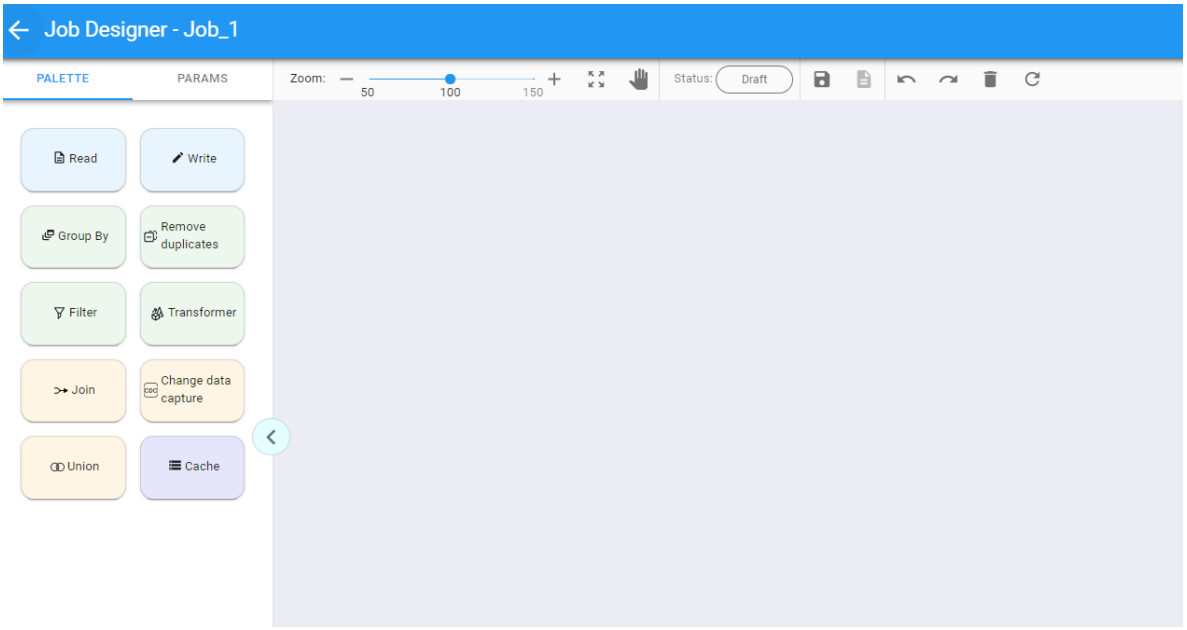
1) On the left configuration panel, user will need to give job a name, update parameters or keep their default values and then push *Confirm* on the panel:



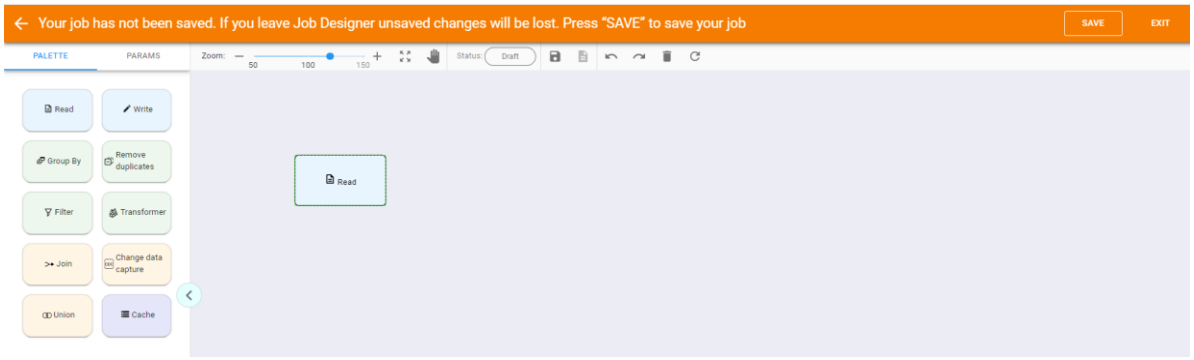


2) Save the job by pushing *Save* button on the *Job Designer* header.

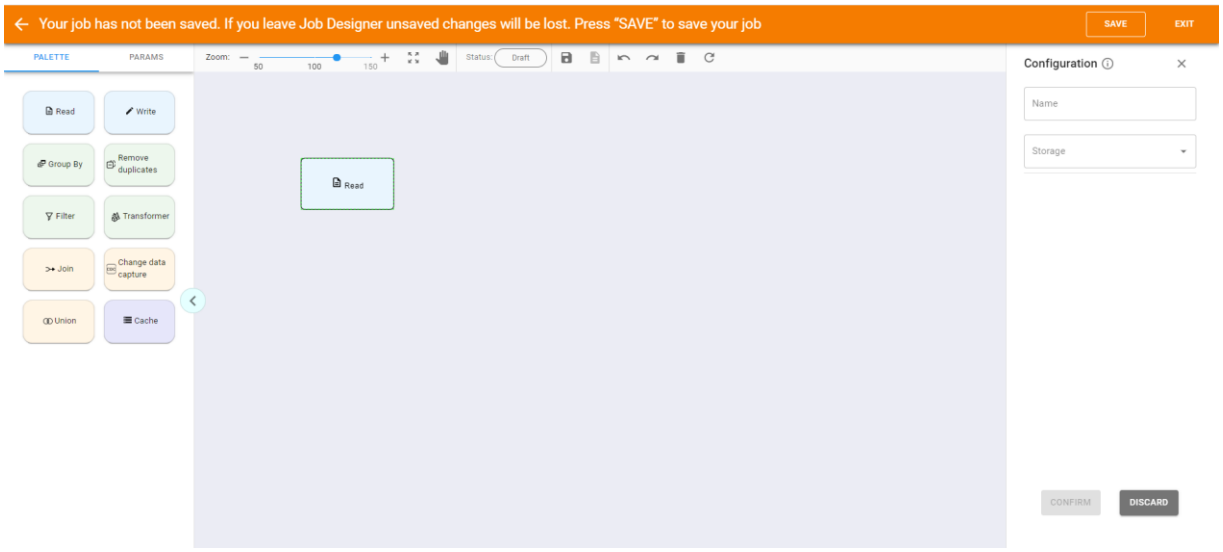
3) Go to Palette tab to see all available stages:



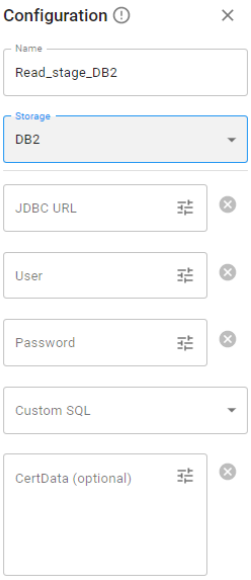
4) User can start creating a job by dragging a stage to the canvas, e.g. user can drag *Read* stage:



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



Enter name for the stage and select *Storage* DB2, if user wants to read data from DB2 table.

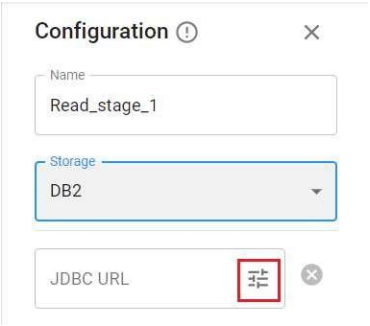


Available *Storage* values for Read stage are:

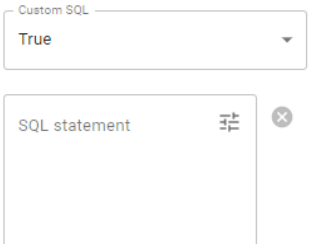
- ✓ AWS S3
- ✓ DB2
- ✓ Cassandra
- ✓ Elastic Search
- ✓ IBM COS
- ✓ Mongo
- ✓ MSSQL
- ✓ MySQL
- ✓ Oracle
- ✓ PostgreSQL
- ✓ Redis
- ✓ Redshift

6) Fill required parameters for DB2 *Storage.*

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

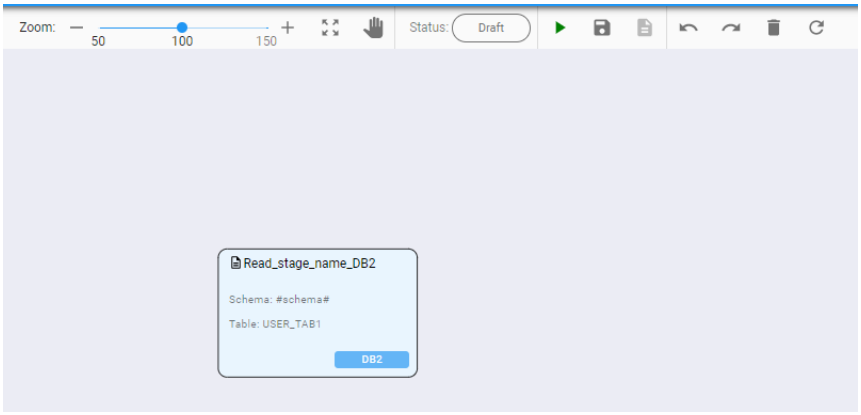


For the DB2 storage, user can use *Custom SQL* only Read stage (e.g. *select * from table where field = value*). Displays the schema and the table fields, if user chooses false. If user chooses true, user will be able to write his own SQL code in the provided field.

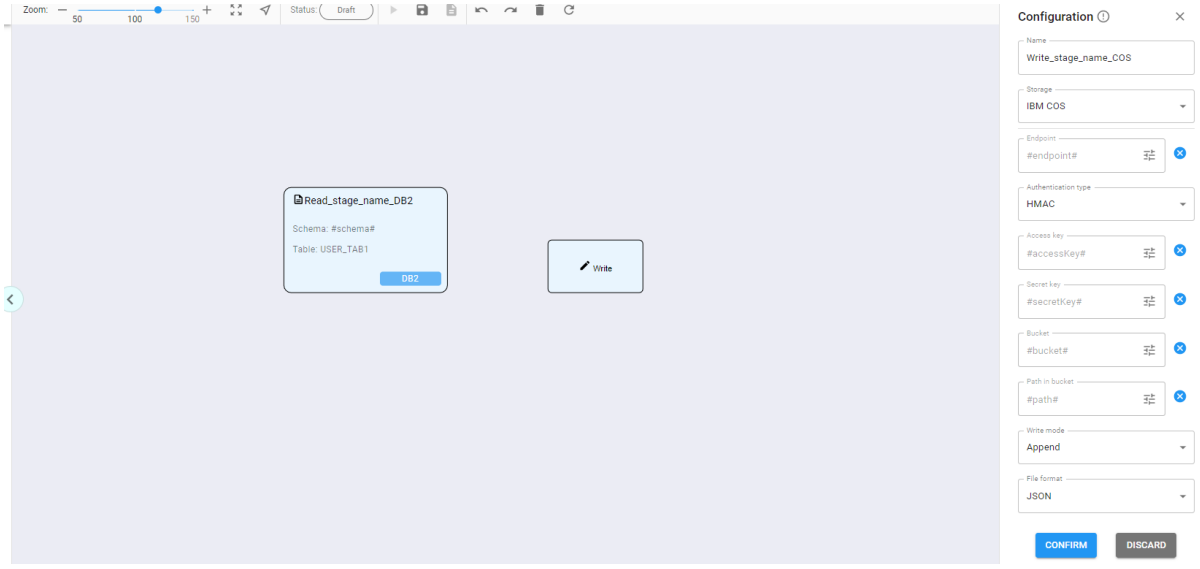


7) Save the stage by pushing Confirm button on the configuration panel. If user wants to save his job at this step, user should press *Save* button on the header.

User has configured the first stage of the job, and it now looks like this:



8) Now drag another stage, e.g. W*rite* stage:

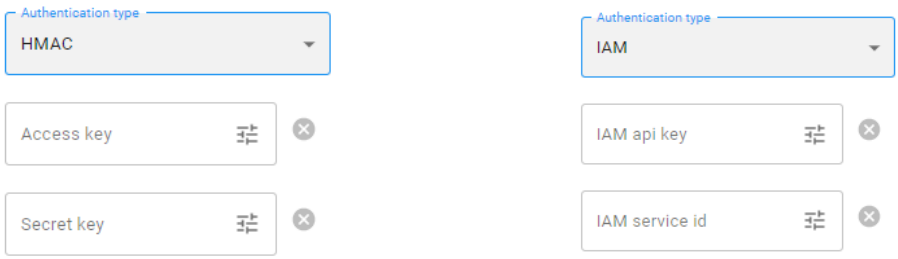


9) Enter a name for the stage and select *Storage IBM COS* if user wants to post data from the DB2 table to Cloud Object Storage file. Fill required parameters for IBM COS *Storage.*

Available *Storage* values for Write stage are:

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

For *IBM COS* Storage, user can use *Authentication type.* Authentication type displays accessKey and secretKey, if user chooses HMAC, or iamApiKey and iamServiceId, if user chooses IAM.



For *IBM COS* and *AWS S3* storages, user can add function *Partition By* in Write stage. Partitions the output by the given columns on the file system. 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 would look like:

- year=2016/month=01/
- year=2016/month=02/

In order to import table data to *Cassandra* source with *Write* stage from another database, at first, user needs to create a layout of the table in Cassandra that he wants to output. Create columns, define a key field, correctly specify the data type of the fields of the future table.

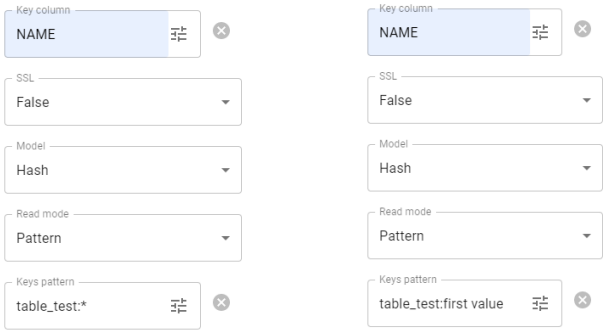
Important: All the above points must match the imported table.

If the column names have uppercase characters in the imported table, when data is output to *Cassandra*, the job will be failed. The reason is that in *Cassandra*, the column names are stored only as lowercase characters. This problem can be solved using a *Transformer* stage.

The results that will be recorded in the *STDOUT* storage, user can view in the Logs. It is also possible to specify the number of records that will be displayed in the Logs. This can be specified in the Quantity field. Can be shown from 1 to 2147483631 records.

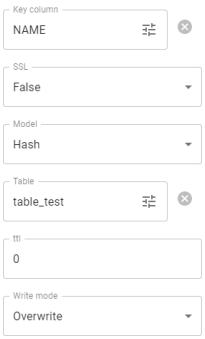
For *Redis* source, the user needs to define *Key column*, *Model, SSL, Read mode, Keys pattern or Table* fields in the Configuration of the Read stage.

- ✓ Key column field. For Read stage, the user specifies a column name to store the hash key.
- ✓ Model (binary, hash) field defines the *Redis* model used to persist DataFrame. By default, it's hash.
- ✓ Read mode (key, pattern) defines the way that the read operation would be handled. If "key" is selected, then the read would be done based on the "table" field. In case of the "pattern", a provided pattern (option "keys Pattern") will dictate, what Redis keys will be read.
- ✓ Keys pattern. If the pattern ends with * (e.g., "table: *"), all keys from the pattern would be read. If the user defines one pattern (e.g., "table: first value"), then only one key will be read.



For Write stage, the user defines such fields as *Key column*, *Model, SSL, TTL*, *Table and Write mode* fields.

- ✓ Key column field. For writing, specifies unique column used as a Redis key. By default, a key is auto-generated.
- ✓ TTL field. Data time to live in seconds. Data doesn't expire if TTL is less than 1. By default, it's 0.



Important:

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

- ✓ Overwrite
- ✓ Append
- ✓ Error if Exists

In '*Overwrite*' Write mode*, user can use Truncate mode* for DB2, Oracle, MySQL, PostgreSQL, MSSQL, Redshift*:*

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

To overcome 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 that would not only delete the data from target table, but also from other tables that use target table's primary key as a foreign key constraint.

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

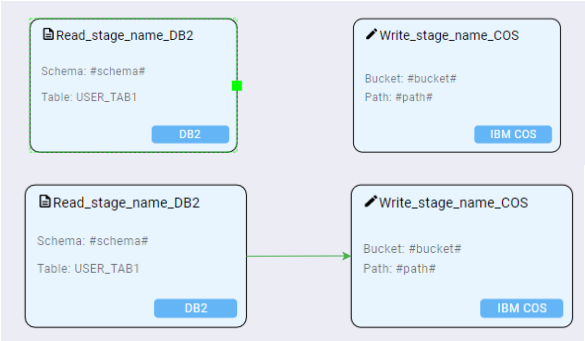
- ✓ CSV
- ✓ JSON
- ✓ Parquet
- ✓ ORC
- ✓ Text

10) Save the stage by pushing *Save* on the panel.

11) Now user has two stages to connect to each other.

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 user reach it, a green arrow should appear.



12) Several stages are also available for the user:

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

Group By stage. In the Configuration panel, user defines key columns for grouping. There is operation *Drop grouping columns*. It needs to remove grouped columns from the output. Or add aggregate function, for example, Count or Avg and others.

Remove duplicates stage. Specify the key column for the operation. To specify more than one key, use a comma or Enter. For Order By operation, user needs to specify column what will be sorted in Asc and Desc.

Filter stage. User enters any boolean expression. Two or more expressions may be combined using the logical operators (AND, OR). Examples: > ((column1 < 10) and column2 between 10 and 25).

Transformer stage. Transformer stage gives the user an ability to modify columns that will be written to some data storage later. The customization allows for (and not limited to):

1. Specify only needed columns.
2. Provide alias for columns.
3. Use Spark-SQL functions/procedures to modify (or create new) columns.

Join stage. There are a lot of types of join:

1. *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.

2. *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.

3. *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.

4. *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.

5. *Cross* join. Returns a result data set where each row from the first table is combined with each row from the second table.

6. *Left semi* join. Returns values from the left side of the relation that has a match with the right.

7. *Left anti* join. Returns values from the left relation that has no match with the right.

Link Ordering option allows user 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 user adds is regarded as the left link, and the last one as the right link.

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.

Union stage. User can union two datasets. NOTE: Column's sequence, names, types are important for union operation.

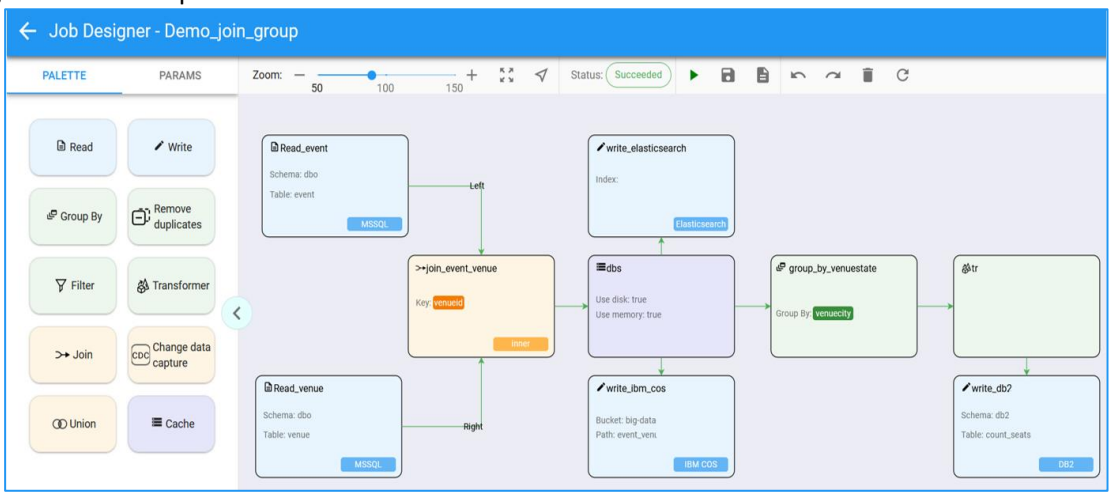
Cache stage. Persists the data set in some storage. The storage type can be tweaked by specifying/combining parameters. Overall, the configuration gives the ability to define:

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

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

User has created a job reading data from the DB2 table and posting it to the IBM COS file. For newly created job, before he run it the status will be *Draft*: 

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



4.3. Job Designer functions overview

The following functions are available in *Job Designer*:

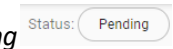
- ✓ Zoom operations: 

- ✓ Show job status: 

- ✓ 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

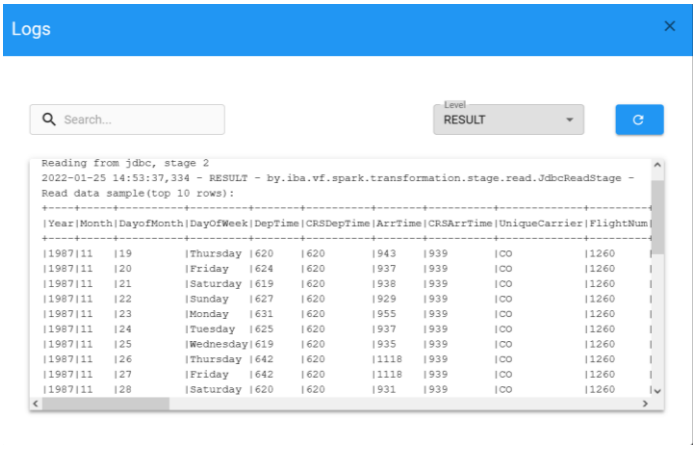
Push *Play* button  to run the job:

User will see its status changed from *Draft* to *Pending* 

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

While running, it can be interrupted with *Stop* button.  When job completed the status will be *Succeeded* or *Failed*

Use *Logs* button  to analyze job logs. User will get to *Logs Screen*:



Logs Screen has several levels:

- ✓ WARNING
- ✓ INFO
- ✓ ERROR
- ✓ DEBUG
- ✓ RESULT

5. Pipeline Operations

5.1. Pipelines Overview

Clicking *Pipelines* menu item will take user to *Pipelines Overview Screen,* which allows user to see a list of pipelines existing within a project.

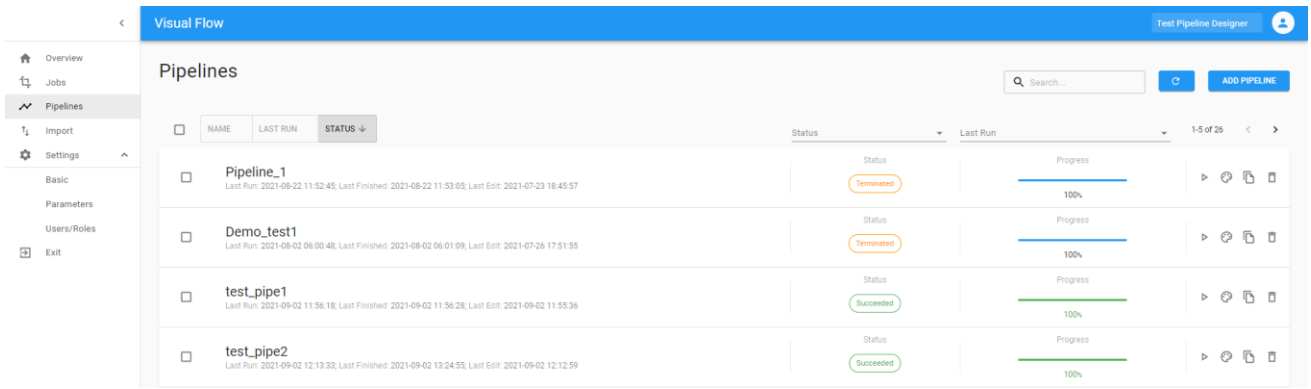
It displays the following information:

- Pipeline Name
- Checkbox for deleting/exporting the pipeline
- Pipeline Last run/Last finished/Last edit
- Pipeline Status
- Pipeline Progress
- Available Actions (Run/Pipeline Designer/Copy/Delete)

Pipeline has a certain status at various phases of execution:

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

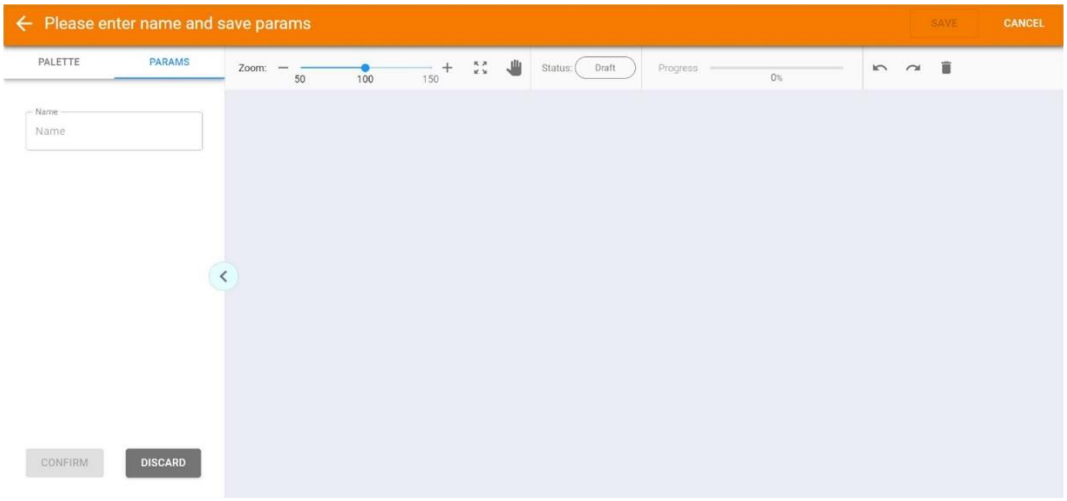
Note: the actions availability and therefore visibility is depending on user authorizations.



5.2. Create a Pipeline

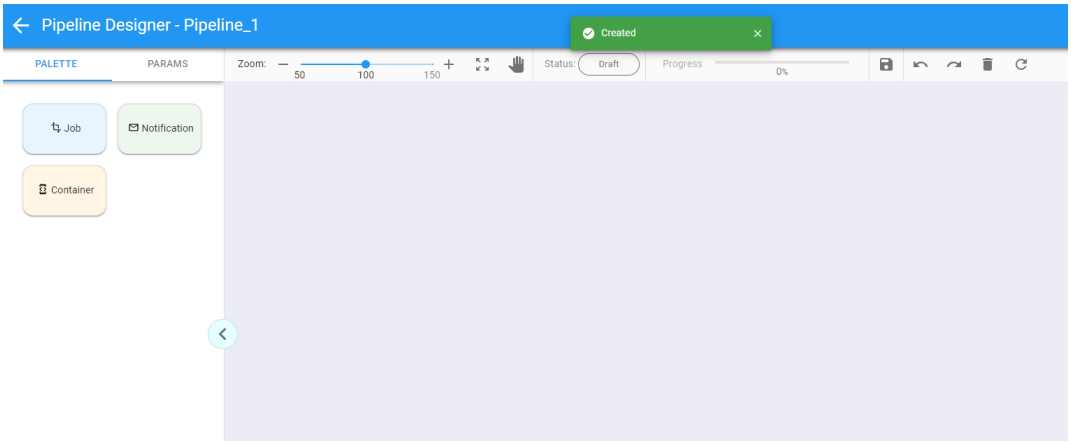
With *Add Pipeline* button pushed, user will get to *Pipeline Designer* for creating a pipeline.

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



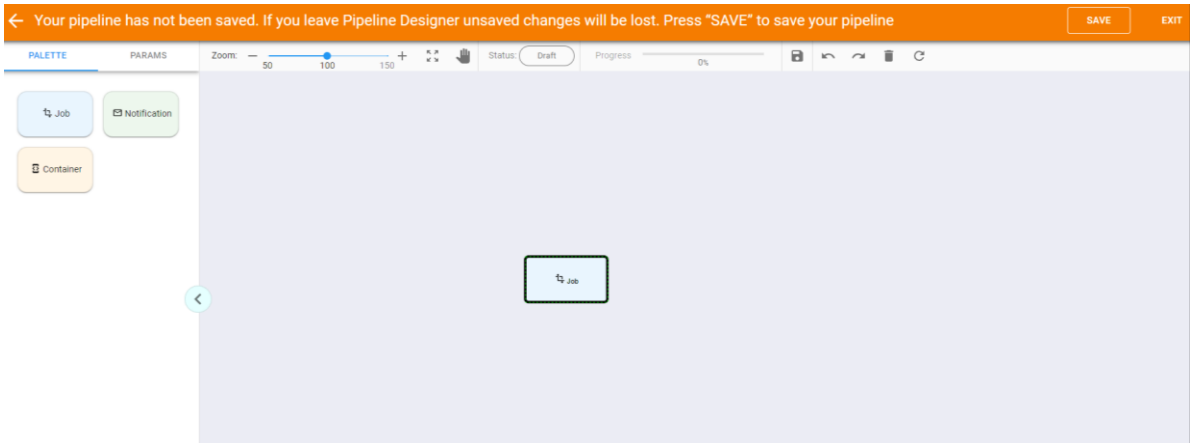
2) Save the pipeline by pushing *Save* button on the *Pipeline Designer* header.

3) After saving the pipeline, Palette tab is opened by default, at this tab user can see all available stages:

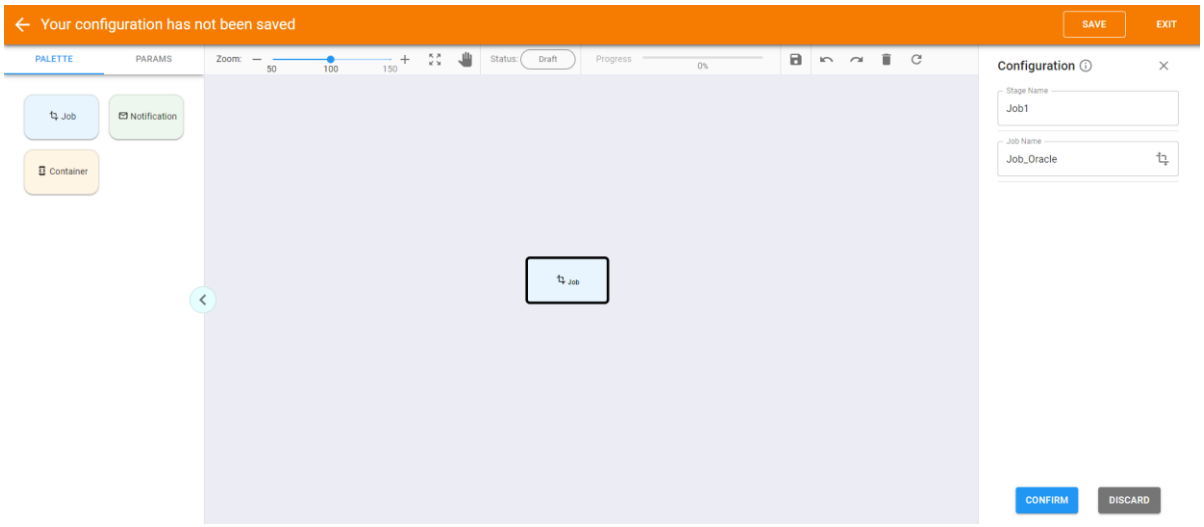


4) Pipeline is a combination of existing jobs stages and/or notification stages and container stages. Notification stage most often added to configuration in the case of job stage failure/success.

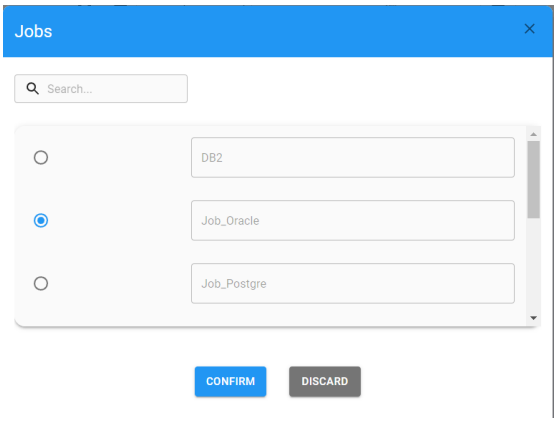
Start creating a pipeline by dragging *Job* stage to the canvas:



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



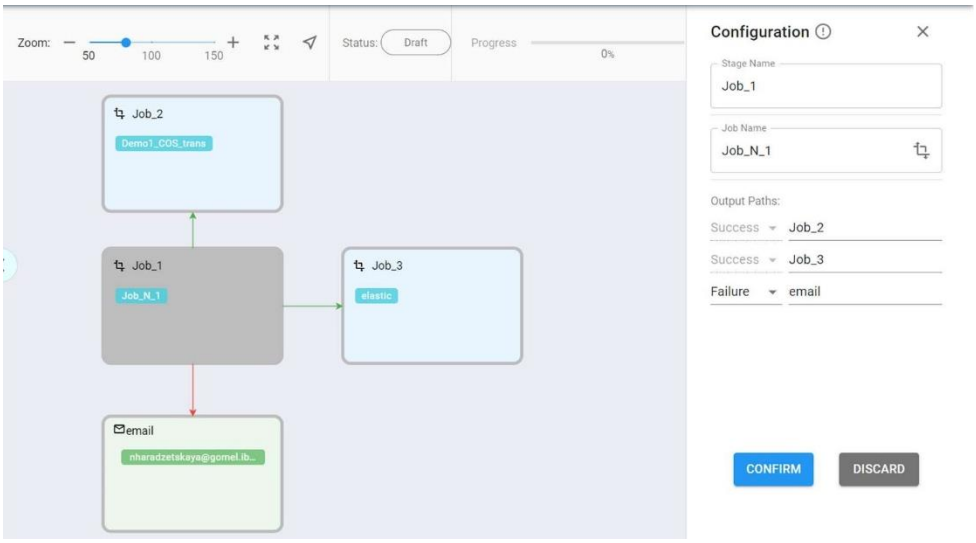
Enter a name for the stage and select a job from the list by pushing *Job* button.



6) Save the stage by pushing *Confirm* button on the panel. If user wants to save his pipeline at this step, user should press *Save* button on the header.

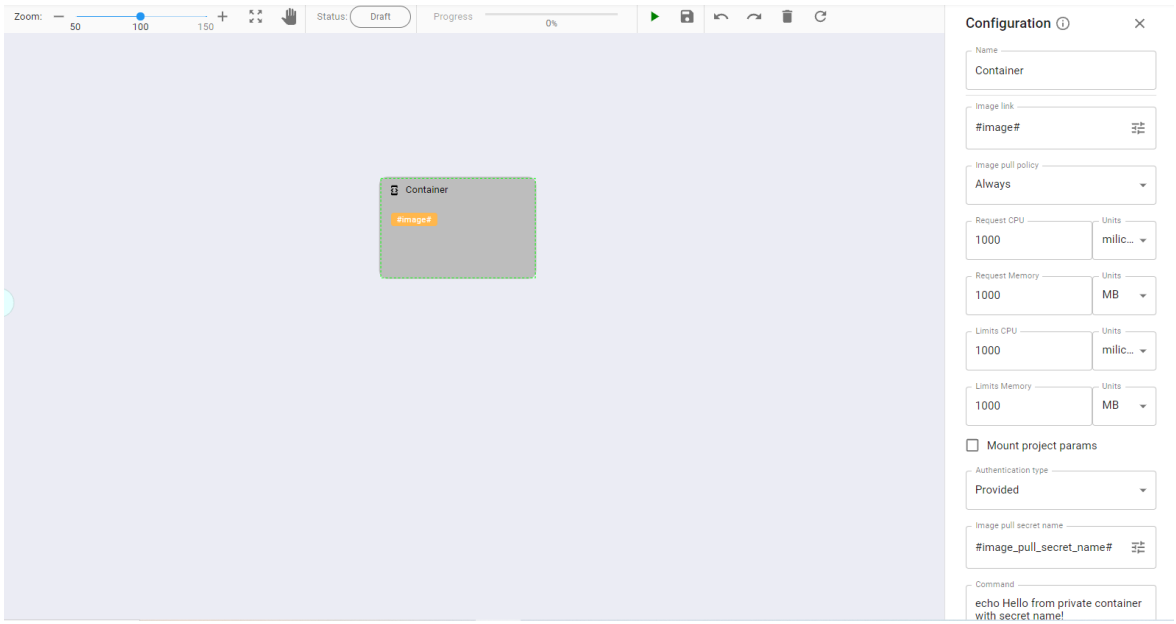
7) Drag and configure other stages. Connect them with the same manner user did in Job Designer.

User can link his stages based on the success or failure of each stage. After connecting stages between themselves, user can choose Success or Failure link on configuration panel. There can be only one connection for failure. See the example of configured pipeline:



A *Custom container* stage is required to run custom commands to execute any logic in the pipeline. Instead of custom commands, can use the created docker image.

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

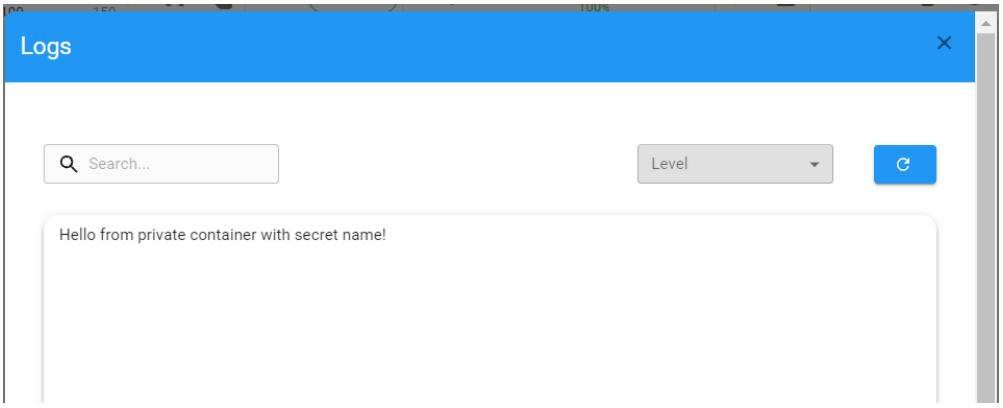


The Container stage has the following fields in the Configuration:

✓ 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* - download only if not exist locally;
– *Always* - download before each start;
– *Never* - do not download use only local copy.
✓ Requests and Limits CPU
✓ Requests and Limits memory
✓ Mount project params. Defines whether to mount all project params as environment variables inside the Pod.
✓ Authentication type
✓ Authentication mode that could be one of these:
– *Not applicable* - image pull secrets are not needed, 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 it's 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 be created.

Important:

Container stage has a *Logs* button  . In Logs window, provided that the pipeline is successfully completed, the text of the command that was previously registered in the *Configuration* of Container stage will be displayed.

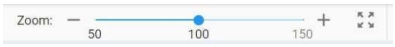


Before the first run or after updating, its status will be *Draft* 

. 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 functions: 

- ✓ Move elements: 
- ✓ Move elements/screen: 
- ✓ Show pipeline status: 

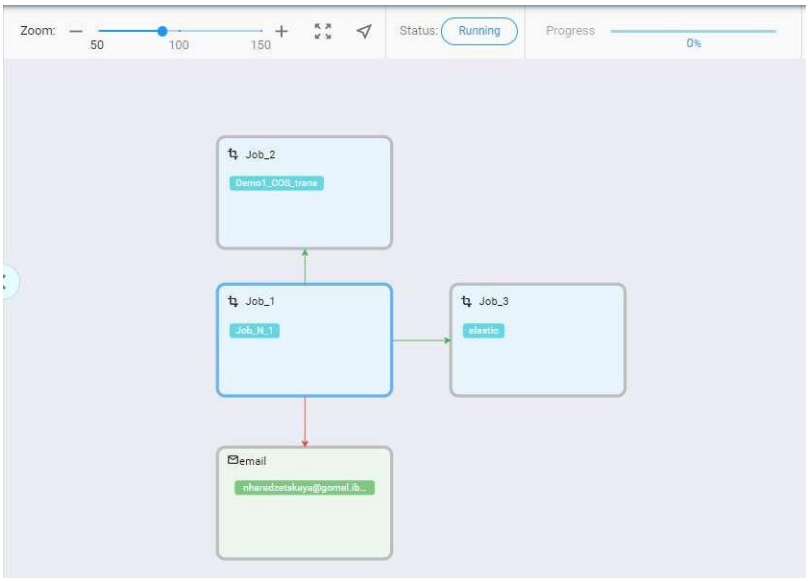
- ✓ Show pipeline progress: 

- ✓ Run pipeline  / Stop pipeline  (for running)
- ✓ Save pipeline 
- ✓ Undo / Redo operation on canvas
- ✓ Remove element from canvas 
- ✓ Refresh 

5.4. Pipeline Execution

If user runs 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.

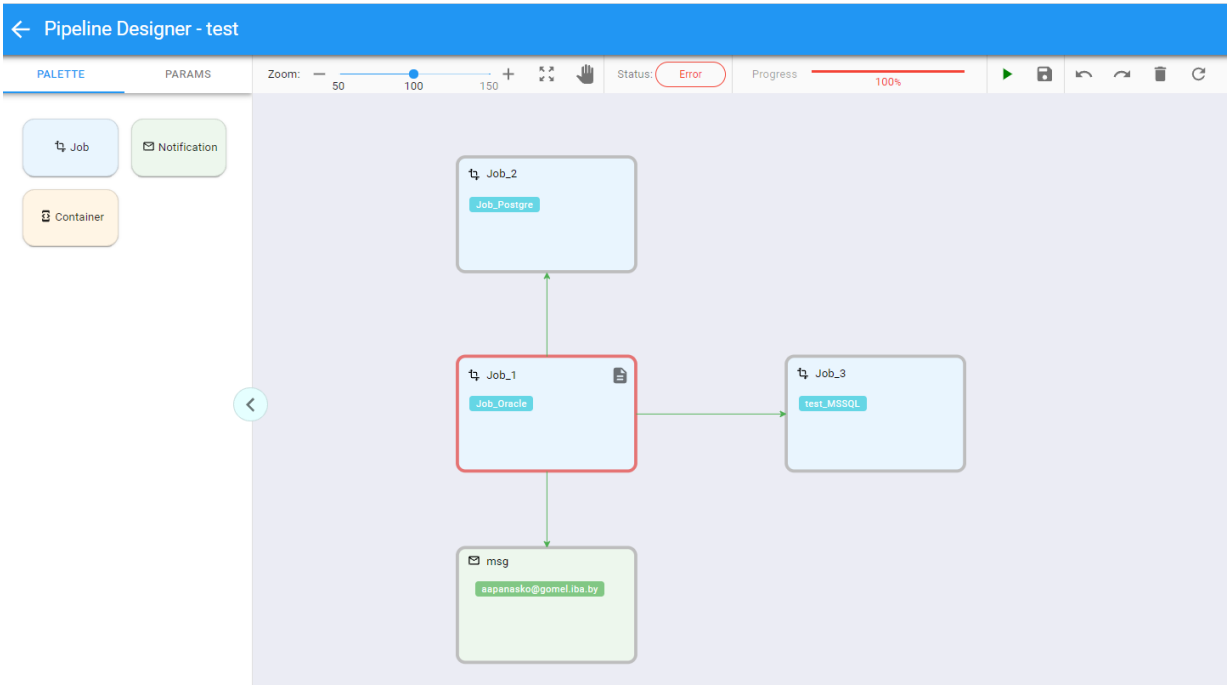
The border of the stage currently running will be painted in *Blue*:



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

The ones configured for failure scenario (red arrow) of the previous stage will remain *Grey* as *Draft* as they have not been executed.

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



Failed pipeline can be re-run from the point of failure with button 

located on the Pipelines Overview Screen.

Important:

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