Creatio Academy

Containerized components

Bulk duplicate search

Version 8.0

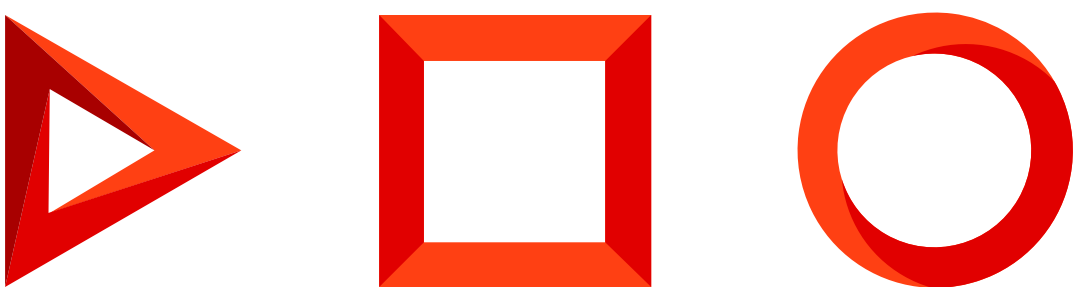

Table of Contents

Bulk duplicate search

PRODUCTS: ALL CREATIO PRODUCTS

Use bulk duplicate search to deduplicate Creatio section records.

Attention. Set up the global search service in ElasticSearch to ensure correct operation of the bulk duplicate search. Learn more in a separate article: Global search. You need repository access to set up the current version of the global search service. Contact Creatio support to verify your license and gain access to the repository.

Basic knowledge of kubernetes or docker-compose and Linux administration is required to set up bulk duplicate search service.

You can deploy the components of global duplicate search using **Kubernetes orchestrator and Helm package manager** or **Docker**.

Use the requirements calculator to check the server requirements.

Set up the bulk duplicate search service using Kubernetes

To set up the service, download the source files. Download files.

To install the service:

1. Set up the target environment:
   a. **Kubernetes cluster**. Learn more about setting up and managing the cluster in the Kubernetes documentation.
   b. **Helm package manager**. Learn more about installing the package manager in the Helm documentation.
   c. **Global search service** via Kubernetes. Learn more in a separate article: Global search.
2. Unpack the **values-onsite.yaml** file from the source file archive and save the file to the same directory as the archive.
3. Open the file. View the main parameters the file uses in the table below. To optimize the load on Redis and RabbitMQ, we recommend using the same service instances for both global search and bulk duplicate search. Bulk duplicate search always uses the same ElasticSearch service as global search.

   Note. By default, the Mongodb service is deployed together with the duplicate search service. If you need to change the Mongodb deployment parameters, unpack the source files from the archive and change the parameters in the mongodb section of the values-onsite.yaml file.
4. Set up the connection to services you installed previously, for example, when setting up global search, in the **values-onsite.yaml** source file.
   - For **Redis**:

Set up the connection to the previously installed Redis. To do this, specify the connection parameters in the `global.redis` section of the **values-onsite.yaml** file.

```
# Multi pods global parameters
global:
  # Redis server connection parameters
  redis:
    host: [host]
    port: [port]
    database: [database]
```

Where

[host] is the address of the Redis server.

[port] is the connection port of the Redis server.

[database] is the name of the Redis database.

- For **RabbitMQ**:

  Set up the connection to the previously installed RabbitMQ. To do this, specify the connection parameters in the `global.rabbitmq` section of the **values-onsite.yaml** file.

  ```
  # Multi pods global parameters
  global:
    # RabbitMQ connection parameters
    rabbitmq:
      host: [host]
      vhost: [vhost]
      port: [port]
      user: [user]
      password: [password]
  ```

  Where

  [host] is the address of the RabbitMQ service.

  [vhost] is the virtual host address of the RabbitMQ service.

  [port] is the amqp connection port of RabbitMQ.

  [user] is the RabbitMQ user.

  [password] is the RabbitMQ user password.

- For **Mongodb**:

  - Disable the Mongodb installation. To do this, specify `enabled: false` in the `mongodb` section of the **values-onsite.yaml** file.

```
mongodb:
  enabled: false
```

- Set up the connection to external mongodb. To do this, specify the connection parameters in the `global.mongodb` section of the **values-onsite.yaml** file.

```
# Multi pods global parameters
global:
  # Deduplication database connection parameters
  mongodb:
    host: [host]
    port: [port]
    user: [user]
    password: [password]
```

Where

[host] is the address of the Mongodb service.

[port] is the port to connect to the service.

[user] is the Mongodb user on whose behalf to connect.

[password] is the Mongodb user password.

- For **ElasticSearch**:

Set up the connection to the previously installed ElasticSearch. To do this, specify the connection parameters in the `global.elasticsearch` section of the **values-onsite.yaml** file.

```
# Multi pods global parameters
global:
  # Elastic search connection parameters
  elasticsearch:
    protocol: [protocol]
    host: [host]
    port: [port]
    path: [path]
    user: [user]
    password: [password]
```

Where

[user] is the ElasticSearch user.

[password] is the ElasticSearch user password.

[port] is the port to connect to ElasticSearch.

[path] is the path parameter of the ElasticSearch service (by default, \).

[protocol] is the ElasticSearch connection protocol (by default, http).

[host] is the address of the ElasticSearch service.

5. Run the `helm install gs -f values-onsite.yaml deduplication.tgz` command. As a result, Helm will install the bulk duplicate search service and selected dependencies.

> **Note.** By default, Helm deploys services with NodePort type.

The main parameters of the global search service the **values.yaml** file uses.

Parameter	Parameter description
duplicatesSearchWorker.maxDuplicatesPerRecord	The maximum allowable number of duplicates for a single record.
log4Net	Logging settings.
global.elasticsearch	ElasticSearch connection parameters.
global.rabbitmq	RabbitMQ connection settings.
global.mongodb	The connection settings of the duplicate search service's internal base.
global.db	The connection settings of the global search service's internal service database.
global.redis	Redis connection settings.

Set up the bulk duplicate search service in Docker

Components of the bulk duplicate search service

Prerequisites:

1. Global search components. View the index in a separate article: Global search.
2. Components of the bulk duplicate search service. View the component index below.

Mongodb. Document-oriented DBMS.

dd-web-api. Web service for communication in Creatio.

dd-data-service. Internal service for communication with mongodb.

dd-duplicates-search-worker. Duplicate search component.

dd-duplicates-deletion-worker. Component for targeted deletion of duplicates.

dd-duplicates-confirmation-worker. Component that groups and filters found duplicates, taking into account their

uniqueness.

dd-duplicates-cleaner. Component that clears duplicates.

dd-deduplication-task-worker. Component that sets the deduplication task.

dd-deduplication-preparation-worker. Component that prepares the deduplication process. Generates duplicate search queries according to the rules.

dd-deduplication-task-diagnostic-worker. Component that controls the execution of the duplicate search task.

To set up the components, download the source files. Download files.

1. Deploy and set up Creatio global search.
2. Download and unpack the necessary source files. Copy them to the computer that has docker, docker-compose software installed. Download files.
3. Set up the environment variables.
4. Launch the containers.
5. Verify that containers are running successfully.
6. Verify logging.
7. Enable the bulk duplicate search functionality in Creatio.

Set up the environment variables

The compose/.env file stores the environment variables. Edit this file to set the variables.

Variable name	Details	Default value
ELASTICSEARCH_URI	The IP address of the server where you deployed ElasticSearch as part of Creatio global search setup.	http://user:password@external.elasticsearch-ip:9200/

Launch the containers

To launch the containers, run the following command:

```
cd compose # go to the compose directory
docker-compose up -d
```

Verify that containers ran successfully

To view the list of all running containers, run the following command at the console:

```
docker ps --filter "label=service=dd" -a --format "table {{.Names}}\t{{.Ports}}\t{{.Status}}\t{{
```

The running containers have an “Up” status.

Verify logging

By default, logging is performed during the “stdout” container command execution. To view the last 100 records from the dd-data-service container, run the following command:

```
docker logs --tail 100 dd-data-service
```

Update the bulk duplicate search version

To update the bulk duplicate search version 2.0 to version 3.0, take the following steps.

1. Back up Creatio duplicate data. To do this, run the `/api/snapshot/backup/gzip/{indexName}` command via the http://[server IP address]:8086/api swagger web-api. {indexName} is the name (last 64 characters) of the index in the “Global search url address” (“GlobalSearchUrl” code) system setting.
2. Delete the version 2.0 of docker volumes. To do this, open the docker-compose directory that contains the version 2.0 files and run the docker-compose down -v command.
3. Install the version 3.0 of the bulk duplicate search service.
4. Upload the duplicate data retrieved on step 1 to the service. To do this, run the `/api/snapshot/restore/gzip` command in the new service version via swagger.

Enable the bulk duplicate search functionality in Creatio

Take the following steps in Creatio.

1. Configure the “Deduplication service api address” system setting.
2. Set up the “Duplicates search” operation permissions.
3. Enable the bulk duplicate search functionality in Creatio. Note that this setting is DBMS-specific.
4. Restart the Creatio application.

Configure the “Deduplication service api address” system setting

Go to the [*System settings*] section, open the “Deduplication service api address” (“DeduplicationWebApiUrl” code) system setting, and specify the URL to dd-web-api. Use a string of the following type: http://external.deduplication-web-api:8086.

Set up the “Duplicates search” operation permissions

Go to the [*Operation permissions*] section, open the “Duplicates search” (“CanSearchDuplicates” code) system operation, and, on the [*Operation permission*] detail, grant permissions to the necessary users/roles, who can search for duplicates.

Enable the bulk duplicate search functionality in Creatio

Toggle the bulk duplicate search (Deduplication, ESDeduplication, BulkESDeduplication) functionality by running an SQL script. The script depends on the DBMS: Microsoft SQL, Oracle, or PostgreSQL.

For Microsoft SQL DBMS

```
DECLARE @DeduplicationFeature NVARCHAR(50) = 'Deduplication';
DECLARE @DeduplicationFeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE Code = @D

DECLARE @ESDeduplicationFeature NVARCHAR(50) = 'ESDeduplication';
DECLARE @ESDeduplicationFeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE Code =

DECLARE @Bulk_ES_DD_Feature NVARCHAR(50) = 'BulkESDeduplication';
DECLARE @Bulk_ES_DD_FeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id
FROM Feature WHERE Code =@Bulk_ES_DD_Feature);

DECLARE @allEmployeesId UNIQUEIDENTIFIER = 'A29A3BA5-4B0D-DE11-9A51-005056C00008';
IF (@DeduplicationFeatureId IS NOT NULL)
BEGIN
      IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @DeduplicationFeatureId)
       UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId =@DeduplicationFeatureI
      ELSE
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allE
@DeduplicationFeatureId)
END;
ELSE
BEGIN
      SET @DeduplicationFeatureId = NEWID()
      INSERT INTO Feature (Id, Name, Code) VALUES
(@DeduplicationFeatureId, @DeduplicationFeature, @DeduplicationFeature)
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allE
END;

IF (@ESDeduplicationFeatureId IS NOT NULL)
BEGIN
      IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @ESDeduplicationFeatureId
     UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId = @ESDeduplicationFeature
      ELSE
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allE
END;
ELSE
BEGIN
      SET @ESDeduplicationFeatureId = NEWID()
      INSERT INTO Feature (Id, Name, Code) VALUES (@ESDeduplicationFeatureId, @ESDeduplicationFe
      INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEm
END;
```

```
IF (@Bulk_ES_DD_FeatureId IS NOT NULL)
BEGIN
      IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @Bulk_ES_DD_FeatureId)
       UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId =@Bulk_ES_DD_FeatureId
      ELSE
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState,FeatureId) VALUES (@allEm
END;
ELSE
BEGIN
      SET @Bulk_ES_DD_FeatureId = NEWID()
      INSERT INTO Feature (Id, Name, Code) VALUES (@Bulk_ES_DD_FeatureId, @Bulk_ES_DD_Feature, @
      INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEm
END;
```

For Oracle DBMS

```
CREATE OR REPLACE FUNCTION
generate_uuid return varchar2 is
       v_uuid varchar2(38);
       v_guid varchar2(32);
BEGIN
       v_guid := sys_guid();
       v_uuid := lower(
'{' ||
       substr(v_guid, 1,8) || '-' ||
       substr(v_guid, 9,4) || '-' ||
       substr(v_guid, 13,4) || '-' ||
       substr(v_guid, 17,4) || '-' ||
       substr(v_guid, 21) ||
       '}'
       );
       RETURN v_uuid;
END;
/
DECLARE
      DeduplicationFeature VARCHAR(50) := 'Deduplication';
      DeduplicationFeatureId VARCHAR(38) := NULL;
      DeduplicationFeatureId_GUID VARCHAR(38) := generate_uuid();
      ESDeduplicationFeature VARCHAR(50) := 'ESDeduplication';
      ESDeduplicationFeatureId VARCHAR(38) := NULL;
      ESDeduplicationFeatureId_GUID VARCHAR(38) := generate_uuid();
      BulkESDeduplicationFeature VARCHAR(50) := 'BulkESDeduplication';
      BulkESDeduplicationFeatureId VARCHAR(38) := NULL;
      Bulk_ES_DD_GUID VARCHAR(38) := generate_uuid();
      allEmployeesId VARCHAR(38) := '{7F3B869F-34F3-4F20-AB4D-7480A5FDF647}';
      State_Deduplication VARCHAR(1) := NULL;
      State_ESDeduplication VARCHAR(1) := NULL;
```

```
      State_BulkESDeduplication VARCHAR(1) := NULL;
BEGIN
      SELECT MAX("Id") INTO DeduplicationFeatureId FROM "Feature" WHERE "Code" = DeduplicationFe
      SELECT MAX("Id") INTO ESDeduplicationFeatureId FROM "Feature" WHERE "Code" = ESDeduplicati
      SELECT MAX("Id") INTO BulkESDeduplicationFeatureId FROM "Feature" WHERE "Code" = BulkESDed
      SELECT MAX("FeatureState") INTO State_Deduplication FROM "AdminUnitFeatureState" WHERE "Fe
      SELECT MAX("FeatureState") INTO State_ESDeduplication FROM "AdminUnitFeatureState" WHERE "
      SELECT MAX("FeatureState") INTO State_BulkESDeduplication FROM "AdminUnitFeatureState" WHE
      IF (DeduplicationFeatureId IS NULL) THEN
       INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (DeduplicationFeatureId_GUID, Dedupli
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       ELSE
       IF (State_Deduplication IS NOT NULL) THEN
       UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = DeduplicationFe
       ELSE
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       END IF;
END IF;
IF (ESDeduplicationFeatureId IS NULL) THEN
       INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (ESDeduplicationFeatureId_GUID, ESDed
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       ELSE
       IF (State_ESDeduplication IS NOT NULL) THEN
       UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = ESDeduplicatior
       ELSE
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       END IF;
END IF;
IF (BulkESDeduplicationFeatureId IS NULL) THEN
       INSERT INTO "Feature" ("Id", "Name", "Code") VALUES(Bulk_ES_DD_GUID, BulkESDeduplicationF
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       ELSE
       IF (State_BulkESDeduplication IS NOT NULL) THEN
       UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = BulkESDeduplica

       ELSE
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       END IF;
      END IF;
END;
```

For PostgreSQL DBMS

```
DO $$

DECLARE
```

```
 DeduplicationFeature VARCHAR(50) := 'Deduplication';
 DeduplicationFeatureId uuid;

 ESDeduplicationFeature VARCHAR(50) := 'ESDeduplication';
 ESDeduplicationFeatureId uuid;

 Bulk_ES_DD_Feature VARCHAR(50) := 'BulkESDeduplication';
 Bulk_ES_DD_FeatureId uuid;

  allEmployeesId uuid := 'A29A3BA5-4B0D-DE11-9A51-005056C00008';

BEGIN

 SELECT "Id" INTO DeduplicationFeatureId FROM "Feature"
 WHERE "Code" = DeduplicationFeature
 LIMIT 1;
 IF (DeduplicationFeatureId IS NOT NULL)
  THEN
   IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = DeduplicationFeatureId)
    UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = DeduplicationFeatu
   ELSE
    INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (
   END IF;
 ELSE
  DeduplicationFeatureId := uuid_generate_v4();
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (DeduplicationFeatureId, DeduplicationFeat
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (al
 END IF;

 SELECT "Id" INTO ESDeduplicationFeatureId FROM "Feature"
 WHERE "Code" = ESDeduplicationFeature
 LIMIT 1;
 IF (ESDeduplicationFeatureId IS NOT NULL)
  THEN
   IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = ESDeduplicationFeatureId
    UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = ESDeduplicationFea
   ELSE
    INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (
   END IF;
 ELSE
  ESDeduplicationFeatureId := uuid_generate_v4();
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (ESDeduplicationFeatureId, ESDeduplicatior
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (al
 END IF;

 SELECT "Id" INTO Bulk_ES_DD_FeatureId FROM "Feature"
 WHERE "Code" = Bulk_ES_DD_Feature
 LIMIT 1;
 IF (Bulk_ES_DD_FeatureId IS NOT NULL)
```

```
   THEN
    IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = Bulk_ES_DD_FeatureId) TH
     UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = Bulk_ES_DD_Feature
    ELSE
     INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (
    END IF;
  ELSE
   Bulk_ES_DD_FeatureId := uuid_generate_v4();
   INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (Bulk_ES_DD_FeatureId, Bulk_ES_DD_Feature,
   INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (al
  END IF;
 END $$;
```

Restart the Creatio application

Clear redis, restart the Creatio application and log in.

Recommended operations for the service functioning

We recommend performing mongodb backup once a day to support the functionality of the service and enable restoring of data, e. g., in case of electricity breakdowns.