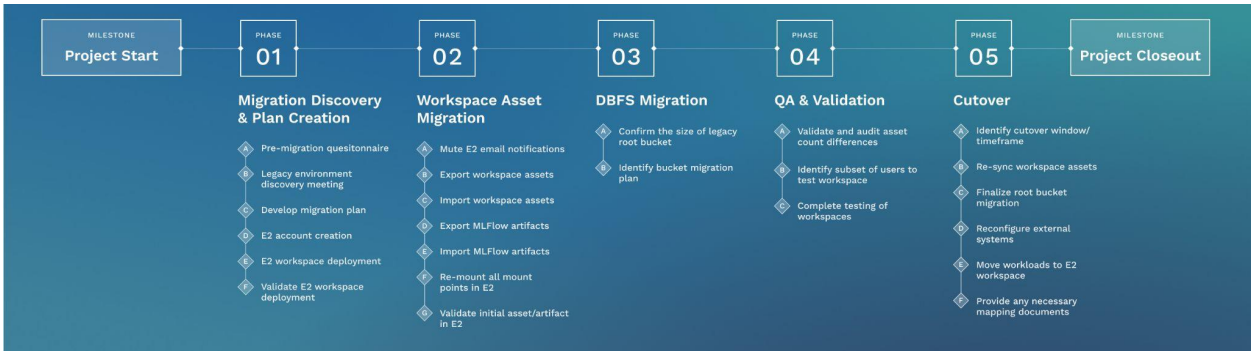
E2 Migration Playbook

This E2 Migration playbook aims to provide the reader with a summary of the migration tasks to be performed when moving a Databricks workspace from Legacy to E2 workspaces. This document contains the comprehensive steps necessary for a successful workspace migration.

Useful tools:

- Databricks Labs Migration Tool (Github)
- Databricks Labs MLFlow Migration (Github)

General Migration Timeline and Phases



1. Migration Discovery and Plan Creation
    a. Pre-migration questionnaire
    b. Legacy environment discovery
    c. Develop migration plan
    d. E2 account creation
    e. E2 workspace deployment
    f. Validate E2 workspace deployment
2. Workspace Asset Migration
    a. Mute E2 email notifications
    b. Export workspace assets
    c. Import workspace assets
    d. Export MLFlow artifacts
    e. Import MLFlow artifacts
    f. Re-mount all mount points in E2
    g. Validate initial assets/artifacts in E2
3. DBFS Migration
    a. Confirm the size of legacy root bucket
    b. Identify bucket migration plan
4. QA and Validation
    a. Validate and audit asset count differences
    b. Identify subset of users to test workspace
    c. Complete testing of workspace
5. Cutover
    a. Identify cutover weekend/timeframe
    b. Re-sync workspace assets
    c. Finalize root bucket migration
    d. Reconfigure external systems
    e. Move workloads to E2 workspace
    f. Provide any necessary mapping documents [to end users]

Migration Planning

Pre-migration Questionnaire

Assessing the current state of the legacy workspaces through the following questions can help uncover most considerations that are necessary in a migration.

1. How many legacy workspaces need to be migrated?
2. How many target E2 workspaces need assets migrated to?
3. Does the E2 workspace need to be HIPAA, PCI, or FedRAMP Compliant?
4. Does the E2 workspace need to be deployed with Terraform?
5. What external systems are integrated with the legacy workspace? (i.e. Airflow, Tableau, etc)?
6. What is the size of the legacy workspace root bucket (DBFS) and how many tables are managed or unmanaged?
7. Does MLflow need to be migrated from Legacy to E2 workspace?

Legacy Environment discovery

The migration team or migration stakeholders can benefit from a dedicated legacy environment discovery meeting. Many data teams within the organization may currently use Databricks and it is an important step to meet with all relevant stakeholders to gather the full understanding of how Databricks is used and what will need to be migrated to E2. For example, Data Science teams often will know specific tools (e.g. MLflow) or requirements (e.g. libraries) that will need to be migrated.

Desired E2 Workspace Setup

Before proceeding with the E2 migration, it's important to review and understand the current legacy environment and architecture, while planning and agreeing upon a desired end state for the E2 workspace architecture. Identifying the total number of desired workspaces and identifying where end users and workflows need to be migrated to is an important milestone before migrating any assets. Work with the Databricks Account team to plan and identify the end goal for the number of workspaces and use of each workspace. This migration opportunity gives organizations a chance to update and make changes to the overall architecture of the platform based on growth and other business needs.

Migration Plan Development

It is highly recommended that a dedicated team is identified to own and drive the success of the migration. This team will need to consist of infrastructure/cloud team members that can create and provision AWS resources and deploy the new E2 workspace. Platform owners to help identify end state goals of the migration and ensure validation and end user communication is delivered. An engineer or multiple engineers that have admin access, so these individuals can run the migration tool and validate the recreations of assets. Once the team is gathered, meeting and

understanding the end goal of the migration and developing a detailed project or action item tracker will help ensure an efficient and timely migration completion. Also, the team may need support of the IAM or IT team to set up SSO in E2 if required.

Account and Workspace Setup

E2 Account Setup

If you already have an E2 Account, please skip this section.

Create your E2 account by going to the following link. Make sure that you select the correct subscription tier, and note the Account ID.

IMPORTANT: If you require regulatory compliance including HIPAA, PCI, or FedRAMP, reach out to your account team with the Account ID of your new account, and ask them to enable Enhanced Security and Compliance for the standard you need to meet.

If you have issues creating the E2 account please reach out to your Databricks Account team or create a Databricks Support Ticket by emailing help@databricks.com.

Set Up Workspace Infrastructure

IMPORTANT: If you do not need custom workspace infrastructure, feel free to follow our workspace creation quickstart to provision all required infrastructure through a CloudFormation template. Afterwards skip to the 'Configure Workspace' section to customize your newly created workspace.

- ☐ To use Terraform to provision the workspace and underlying infrastructure, please see the provider documentation.
- ☐ If you require a Customer-Managed VPC, create the prerequisites for deployment in your AWS account. These include:
  - ☐ A VPC of size /25 or larger (recommended is at least /20).
  - ☐ At least 2 private subnets in different AZs.
  - ☐ If not using Private Link, a NAT Gateway and Internet Gateway.
  - ☐ If not using Private Link, a route table that includes a default route to the NAT.
  - ☐ A Security Group that complies with the requirements here.
  - ☐ Subnet NACLs that comply with the requirements here.

- [ ] If using Private Link, PL endpoints are deployed according to this section. Ensure that Private DNS is enabled at the endpoint and VPC level.
- [ ] In general, customers will also require regional endpoints, per this section. Make sure that S3, Kinesis Streams, and STS endpoints are created.
- [ ] **IMPORTANT:** Do not re-use a Databricks-Managed VPC from another workspace.

- [ ] If using a Customer-Managed VPC, record the IDs of the PL endpoints (if used), security group, subnets, and VPC. These will be used to launch the workspace.
- [ ] Create a cross-account role and S3 root bucket that will be used for deployment. For the cross-account role, make sure that you select the appropriate version of the policy (i.e., if they are using Customer-Managed VPC, it must be "Your VPC, Default" or "Your VPC, Custom"). As a general rule, most customers use "Your VPC, Custom", especially coming from a legacy workspace.
- [ ] Record the ARN of the cross-account role and the name of the S3 bucket. These will be used to launch the workspace.
- [ ] Once all infrastructure is ready, determine if the deployment requires an API-based deployment, or if it can use the Account Console UI (UI recommended).
  - [ ] As of Q1 FY23, PrivateLink deployments require API-based deployment.
  - [ ] For all other deployments, use the UI.

Deploy a Workspace via the UI (Recommended)

For UI-based deployments, follow these steps. If using APIs instead, see the next section.

- [ ] Navigate to the Workspaces page on accounts.cloud.databricks.com.
- [ ] Click "Create Workspace", and then "Custom AWS configuration".
- [ ] Fill in the workspace name, and confirm that the workspace URL looks as desired
- [ ] Confirm that the workspace tier is correct.
- [ ] Select the desired workspace region (the same region as any AWS infrastructure that was deployed earlier).
- [ ] In the "Credential configuration" box, scroll down and select "Add a new credential". Enter a user-friendly credential name and the role ARN from above.
- [ ] In the "Storage configuration" box, scroll down and select "Add a new storage configuration". Enter a user-friendly storage name and the S3 bucket name.
- [ ] If a Customer-Managed VPC is being used, click on "Advanced configurations" and select "Add a new network configuration" in the box; enter a user-friendly name, and then add the VPC, subnets, and security group created earlier.
- [ ] For workspace deployment troubleshooting, see the Troubleshooting section below.

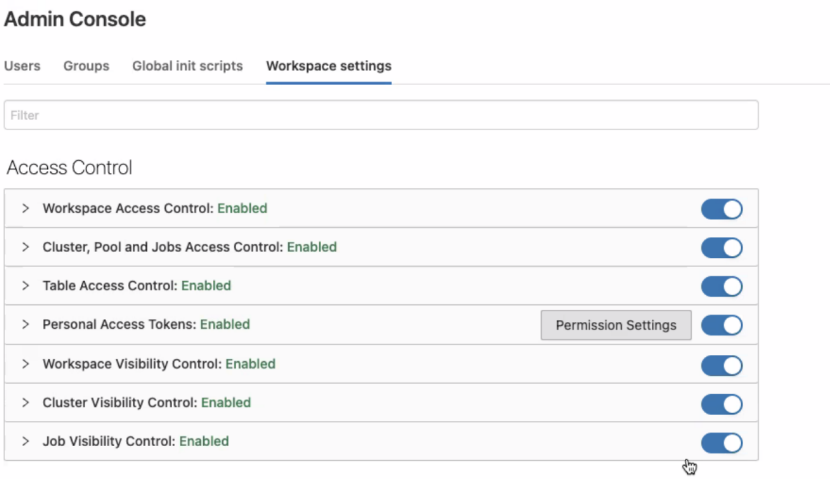
(Optional) Deploy a Workspace via API

For API-based deployments, follow the steps below. Please see the master Account API documentation for API specifics.

- [ ] Confirm the E2 Account ID, username, and password that will be used to call the APIs.
- [ ] Call the Create Credentials API using the cross-account role ARN. Record the credentials object ID from the response.
- [ ] Call the Create Storage Configuration API using the S3 bucket created as the root bucket. Record the storage configuration object ID from the response.
- [ ] Register the PrivateLink endpoints by calling the VPC Endpoint API. This must be done once for each endpoint (REST, Back-End, and if used, Front-End). Record each response separately and note the use case for each (as these are easily mixed up).
- [ ] Call the Create Network API using the AWS IDs for the VPC, Subnets, and Security Group; use the Databricks endpoint IDs (i.e., the responses from the previous step) for the back-end and REST endpoints. Do not use the front-end endpoint ID in this step.
- [ ] Call the Create Private Access Settings API. Note that this requires adjustment based on your intended setup; see this page for more information on the various settings.
- [ ] If required, call the Create Encryption Key API. You may need to do this separately for Managed Services and Storage if you wish to use separate keys for each. If not, you can set the `"use_cases"` value to `"MANAGED_SERVICES","STORAGE"`, which will use the same key for both.
- [ ] Call the Create Workspace API. All objects referenced in this API should be Databricks IDs, not AWS IDs.
- [ ] Call the Get Workspace API to check the status of the workspace deployment. If it returns `RUNNING`, you may continue to workspace validation. If not, see the Troubleshooting section below.

Configure Workspace

Before starting the object migration, toggle all the desired configurations by going to the Admin Console -> Workspace settings tab. Note that the Access Control configurations **must be** set as desired before starting the migration.



Validate E2 Workspace Deployment

When the workspace is deployed it is recommended that few tests are conducted on components of the workplace. If the steps below fail, you may need to check your AWS service endpoints, ensure connectivity to the metastore, or update your egress rules.

- Log into the workspace; the initial user will be the same username and password as you used to log into accounts.cloud.databricks.com. This user will be the workspace admin
- Start a cluster to ensure it spins up properly and runs without issues
- Install a pip library on the cluster, such as pandas or numpy
- From the Workspace homepage, click on "Explore the Quickstart Tutorial", which will create a new notebook for you. Attach the notebook to the cluster you just created, and click "Run All"
- Upload a json/csv file to the DBFS
- Create an empty database or use the default database. Then create a delta table using spark and the uploaded files
- Drop the created delta table
- Create and run one sample workflow using a job cluster

Turn off new email notifications

By default, users added to Databricks workspaces will receive an email invitation to join the workspace. Generally the migration process takes place behind the scenes to not impact end users day to day, until the new workspace is ready for them. To mitigate invitations to join a workspace that is not yet ready for end users, the Databricks Account team can opener contact help@databricks.com to mute email notifications. This process will only stop the email notification from going out when users are recreated in the E2 workspace with the migration tool. Please provide the E2 workspace ID and workspace URL to the Databricks Account team, so they can fill out the necessary ES Ticket. These requests are typically turned around within 24-48 hours by the Databricks team.

Users added directly to the workspace via the UI will still be sent the workspace invitation and the muting of email notifications only affects users added through the SCIM API (and therefore the migration tool).

Legacy And E2 Workspace Access

Admin access is needed to ensure developers can see all assets in the environment. The personal access token created to set up the Migration tool will therefore need to be an Admin PAT.

Set up SSO in E2 Workspace

Upon completion of the asset migration to the E2 Workspace, SSO will need to be enabled in the E2 workspace if it is not already done. The SSO can be set up by following the steps from this documentation.

Migration Tool Setup

Dependencies

- E2 Account and Workspace Setup
- Migration Developer Admin Access to Legacy and E2 Workspace(s)

Prerequisites

- An environment running Linux or Mac with python 3.6+, pip, git, and the databricks CLI installed.

Migration Tool Capabilities

Below is an overview of what assets can migrate, don't migrate, and change during a migration.

Component	Export	Import
Users/Groups/Instance Profiles	Supported	Supported
Clusters (w/ ACLs), Pools, Policies	Supported	Supported
Notebooks	Supported	Supported
Notebook ACLs	Supported	Supported
Repos	Supported	Supported
Metastore	Supported	Supported
Jobs (w/ACLs)	Supported	Supported
Libraries	Supported	Unsupported
Secrets	Supported	Supported
Table ACLs	Supported	Supported
ML Models	Supported*	Supported*
Access Tokens	Unsupported	Unsupported
DBFS mount points	Supported	Unsupported

*MLFlow objects use a separate export/import tool than the other general migration tool

Below is a list of components that change during a migration.

Component	Legacy vs. E2
Object IDs (e.g. Job IDs)	Change
Clusters JDBC Connections	Change

Regarding the migration of libraries

There are generally 3 types of libraries based on where the libraries are added/installed:

1. Workspace Libraries: This is not supported using the migrate tool, if needed workspace UI libraries can be added manually.
2. Cluster Libraries: This is not supported using the migrate tool, if needed cluster libraries can be installed manually.
3. Job Libraries: This is supported using the migrate tool as Job libraries paths are included in the Job configuration.

Note: To use any of the libraries uploaded to DBFS, DBFS Migration should happen first.

Setup

1. Generate Personal Access Token

   Before setting up any migration tools, Personal Access Tokens need to be created for every workspace associated with the migration, legacy and E2. Use the following reference article to help set up the access tokens.

2. Databricks-CLI Profiles

   In order to run the migration tool from your Linux shell, create a profile for the old workspace by typing:

   ```
   databricks configure --token --profile oldWS
   ```

   In this case `oldWS` is the profile name you'll refer to for running the migration tool export pipeline.

   When you use the databricks cli configure command, you'll be prompted for 2 things:

1. Databricks Host (should begin with https://): When this happens, enter the old databricks workspace URL that you captured in your file above. The Databricks Host should look something like this: *https://<instance_name>.databricks.com*
2. Token: When this happens, paste in the token you generated for the old databricks account.

Repeat the steps above for the new databricks account and change the oldWS profile name to something like `newWS` in order to keep track of which account you're exporting FROM and which account you're importing TO.

```
databricks configure --token --profile newWS
```

In this case `newWS` is the profile name you'll refer to for running the migration tool import pipeline.

3. Clone the Git Repo from Databricks Labs Github
4. Open Linux shell and navigate to the migrate repository.
5. Install package dependencies

```
python3 setup.py install
```

Workspace Asset Migration

Dependencies

- Setup of Databricks CLI profiles and migration tool complete

Prerequisites

- An environment running Linux with python, pip, git, and the databricks CLI installed.
- Admin access to both the old and new databricks accounts in the form of a Personal Access Token.

Export the Workspace

To export a workspace, run:

```
python3 migration_pipeline.py --profile $SRC_profile
--export-pipeline --user-checkpoint [--session $SESSION_ID]
```

Where `$SRC_profile` is the Databricks profile for the source workspace, as configured during Setup, and $SESSION_ID is an optional session identifier used for subsequent checkpoint runs. All data is exported to a folder named according to the `$SESSION_ID` value under the logs folder - "logs/`$SESSION_ID`". If `$SESSION_ID` is not specified, a random value will be generated.

Also, the `—notebook-format SOURCE` parameter should be specified as while overwriting the notebooks during the cutover or the last sync the script needs the exported workspace assets to be in the SOURCE format.

Skip tasks that keeps failing

To skip certain tasks/assets that are either not required or throwing various errors, run:

```
python3 migration_pipeline.py --profile $SRC_profile
--export-pipeline --user-checkpoint [--session $SESSION_ID]
--skip-tasks $TASK_LIST
```

In the above command, `$TASK_LIST` is a space separate list of tasks and task can be anything from `instance_profiles, users, groups, workspace_item_log, workspace_acls, notebooks, secrets, clusters, instance_pools, jobs, metastore, metastore_table_acls, mlflow_experiments, mlflow_runs`. Skipping `mlflow_experiments` and `mlflow_runs` is a good idea as there is a separate tool for MLFlow migration.

Example syntax:

```
python3 migration_pipeline.py --profile $SRC_profile
--export-pipeline --user-checkpoint [--session $SESSION_ID]
--skip-tasks mlflow_experiments mlflow_runs
```

Recommended parameters and checkpointing

As a starting point, we recommend using the following parameter values:

- retry-total=30
- num-parallel=8
- retry-backoff=1.0

These can be adjusted per your scenario if needed; in general, if API limits are being hit, you can increase retry-backoff, decrease num-parallel, or both.

If script failure occurs, you can safely rerun the same command with `-–use-checkpoint` and `--session $SESSION_ID` to let the migration pick up from the previous checkpoint and rerun.

Updating the Instance-profiles/AWS Account ID

If your source and destination workspaces are in different accounts, you will need to update the Instance Profile ARN accordingly during the migration. To do this, run the following command after exporting the workspace assets:

```
python3 export_db.py --profile $SRC_PROFILE --use-checkpoint
--old-account-id $OLD_AWS_ACCT_ID --update-account-id
$NEW_AWS_ACCT_ID --set-export-dir $EXPORT_DIR/$SESSION_ID
```

Where `$SRC_profile` is the Databricks profile for the source workspace, as configured during Setup, and $SESSION_ID is an optional session identifier used for subsequent checkpoint runs. `OLD_AWS_ACCT_ID` is the source AWS account ID and `NEW_AWS_ACCT_ID` is the destination AWS account ID. Note that this will only update the ARN in the Instance Profiles; the same instance profiles names must still exist in the destination AWS account.

Updating Email IDs

If the user's email ids need to be changed during the migration process, it can be easily done using the below command once the export of the workspace is completed

```
python3 export_db.py --profile $SRC_PROFILE --replace-email
old_email@abc.com:new_email@abc.com --session $SESSION_ID
```

For multiple email_ids you can use following syntax which has email_id mapping separated by , (comma)

```
python3 export_db.py --profile $SRC_PROFILE --replace-email
old_email@abc.com:new_email@abc.com,old_email2@abc.com:new_email2@abc
.com --session $SESSION_ID
```

The above commands will update logs to use the new email_id so running import-pipeline after running the above command will migrate assets using new email_id.

Import the Workspace

To import into a target workspace, run:

```
python3 migration_pipeline.py --profile $DST_PROFILE
--import-pipeline --use-checkpoint [--session $SESSION_ID]
```

The same recommended parameters as above apply in the import workflow, and similarly, if a failure occurs, `--use-checkpoint` can be used to rerun from the last checkpoint.

Validation

Simple workspace object validation can be performed once the import is completed by first exporting the contents of the target workspace:

```
python3 migration_pipeline.py --profile $DST_PROFILE
--export-pipeline --use-checkpoint --cluster-name
```

And then running the validate_pipeline.sh script:

```
./validate_pipeline.sh $SRC_EXPORT_SESSION_ID $DST_EXPORT_SESSION_ID
```

Once this completes, check the console summary, as well as the logs folder (where a new folder should be generated).

MLFlow Migration

Dependencies

- Migration of workspace assets complete (users, groups, directories, etc. created by the workspace asset migration will be needed for reference during the MLFlow migration).

Prerequisites

- An environment running Linux with Python 3.7.6+, pip, and git.
- Admin access to both the old and new databricks accounts in the form of a Personal Access Token.

Options to migrate MLflow objects

1. Cloning/installing the tool on your machine

   This option is recommended when the size of MLFlow artifacts is pretty low (<= 100 GB). Follow the Local setup instructions to set up the migration tool on your local machine.

2. Using Databricks notebooks to run the export on databricks workspace

   This option is recommended when the size of MLFlow artifacts is pretty big (> 100 GB). The reason is during the export the artifact will be downloaded on the machine on which the tool is running. So if the tool is running on a local machine, it will occupy a large amount of space there. This option provides a solution to that problem by running the notebooks to export MLFlow artifacts to an external S3 bucket mounted to the workspace. Also, you can utilize databricks jobs to run the export-import as it will be a cost efficient solution.

   The notebooks can be accessed from here.

MLFlow Migration Tool Setup

Local setup

First create a virtual environment.

```
python3 -m venv mlflow-export-import

source mlflow-export-import/bin/activate
```

There are two different ways to install the package.

1. Install from github directly

```
pip3 install
git+https://github.com/amesar/mlflow-export-import/#egg=mlflow-export-import
```

2. Install from github clone (Recommended)

```
git clone https://github.com/amesar/mlflow-export-import

cd mlflow-export-import

pip3 install -e .
```

3. To run the tools externally (from your laptop) against a Databricks tracking server (workspace) set the following environment variables.

```
export MLFLOW_TRACKING_URI=databricks

export DATABRICKS_HOST=https://mycompany.cloud.databricks.com

export DATABRICKS_TOKEN=MY_TOKEN
```

For full details see Access the MLflow tracking server from outside Databricks.

Databricks setup

There are two different ways to install the package.

1. Install package in notebook

Install notebook-scoped libraries with %pip.

```
pip3 install
git+https:///github.com/amesar/mlflow-export-import/#egg=mlflow-export-import
```

2. Install package as a wheel on cluster

Build the wheel artifact, upload it to DBFS and then install it on your cluster.

```
python3 setup.py bdist_wheel

databricks fs cp dist/mlflow_export_import-1.0.0-py3-none-any.whl
{MY_DBFS_PATH}
```

Export / Import using Bulk Tools

The bulk import/export tool is the recommended way to migrate your MLFlow assets. Once completing all of the requirements mentioned above, run the following command to export all MLFlow assets. This command will export all MLflow objects of the tracking server (Databricks workspace) - all models, experiments and runs as well as a run's Databricks notebook (best effort).

```
export-all --output-dir out
```

While there is not an import-all tool, the import-models command below will import all MLFlow assets from above, except for databricks notebooks. The following command will import models and their runs and experiments from the above exported directory.

```
import-models --input-dir out
```

Export / Import specific models and associated experiments

This command is recommended when you want to migrate only certain models and all the associated experiments. Once completing all of the requirements mentioned above, run the following command to export specified models in --models parameter.

```
export-models --output-dir out --models 'model1,model2'
```

More details can be found here: Export-registered-models Documentation

The following command will import models and their runs and experiments from the above exported directory. The --input-dir should be the exported path of the model.

```
import-models --input-dir out
```

More details can be found here: Import-registered-models Documentation

Export / Import specific experiments

This command is recommended when you want to migrate only certain experiments. Once completing all of the requirements mentioned above, run the following command to export specified experiments in --experiments parameter. The parameter can take comma separated list of experiment names or list of ids such as '1,2,3'

```
export-models --output-dir out --experiments 'exp_name1,exp_name2'
```

More details can be found here: Export-experiments Documentation

The following command will import experiments from the above exported directory. The --input-dir should be the exported path of the experiments.

```
import-experiments --input-dir out
```

More details can be found here: import-experiments Documentation

MLFlow Export / Import Limitations

General Limitations

- Nested runs are only supported when you import an experiment.
- If the run linked to a registered model version does not exist (has been deleted) the version is not exported since when importing MLflowClient.create_model_version requires a run ID.

Exporting Notebook Revisions

- The notebook revision associated with the run can be exported. It is stored as an artifact in the notebooks folder under the run's artifacts root.
- You can save the notebook in the supported SOURCE, HTML, JUPYTER and DBC formats.

Importing Notebooks

- Partial functionality due to Databricks API limitations.
- The Databricks API does not support:
    - Importing a notebook with its entire revision history.
    - Linking an imported run with a given notebook revision.
- When you import a run, the link to its source notebook revision ID will not exist and thus the UI will point to a dead link.
- As a convenience, the import tools allows you to import the exported notebook into Databricks. For more details, see:
    - README_point - Import run
    - README_point - Import experiment
- The imported notebook cannot be attached to the run that created it.
- If you have several runs that point to different revisions of the same notebook, each imported run will be attached to a different notebook.
- You must export a notebook in the SOURCE format for the notebook to be imported.

User ID

- When importing a run or experiment, for open source MLflow you can specify the user owner. For Databricks import you cannot - the owner will be based on the personal access token (PAT) of the import user.

DBFS Migration

Please contact your account team or help@databricks.com to discuss this phase of the migration. Below is a high-level overview of how your team could move the data yourself.

Sizing

To initiate the migration, the DBFS root bucket of the legacy workspace should be cataloged and sized. At this time it's recommended to remove any top level directories or subdirectories that are no longer needed, this will help reduce the size of the overall bucket.

Using Spark + Delta

Once the cataloging/analysis is completed there are a few approaches to migrating the data. Table data can be migrated by using DEEP CLONES to move managed tables to unmanaged tables. Those unmanaged tables can then easily be re-registered in the E2 workspace. Any non-table data can be copied by using dbutils.fs.cp() or by copying files with an S3 Batch or S3 Sync within your AWS Account.

Using S3 Sync or S3 Batch

The entire DBFS root bucket can be migrated using AWS S3 tooling if desired. This has the benefit of being the simplest option, but may be slow, and will also break Delta table transaction logs. For this reason, it is recommended at a minimum to migrate Delta tables using DEEP CLONE as above. For data that you do wish to move using S3 tooling, note that the target prefix will differ from the source prefix (i.e., the target S3 prefix will include the target workspace ID). You should also ensure that the folder hierarchy within the top-level prefix does not change when migrating data, as this will break some Databricks functionality.

Additional E2 Configurations

Re-mounting Mount Points

First, identify all mount points to the current legacy workspace. Determine which mount points, if not all, to move to the E2 workspace. To list out the mount points in the legacy workspace, use the below command.

```
python3 export_db.py --profile DEMO --mounts
```

From a notebook, the mount points can be re-mounted to the E2 workspace. This article can also be used for more details.

Mount a bucket using an AWS instance Profile

Make sure to configure the cluster with the associated instance profile needed for access to the S3 bucket.

```
aws_bucket_name = "<aws-bucket-name>"

mount_name = "<mount-name>"

dbutils.fs.mount("s3a://%s" % aws_bucket_name, "/mnt/%s" %
mount_name)
```

Mount a bucket using AWS key

Make sure to use Databricks secrets to store the keys. When an S3 bucket is mounted using keys, *all users* have read and write access to *all the objects* in the S3 bucket.

```
access_key = dbutils.secrets.get(scope = "aws", key =
"aws-access-key")

secret_key = dbutils.secrets.get(scope = "aws", key =
"aws-secret-key")

encoded_secret_key = secret_key.replace("/", "%2F")

aws_bucket_name = "<aws-bucket-name>"

mount_name = "<mount-name>"
```

```
dbutils.fs.mount("s3a://%s:%s@%s" % (access_key, encoded_secret_key,
aws_bucket_name), "/mnt/%s" % mount_name)
```

QA and Validation

Dependencies

- Migration of all assets complete, except for jobs
- Setup of SSO in E2 Workspace
- DBFS Root Migration Plan Identified

Below are the recommended steps to take when conducting QA and Validation of the migration of assets. These steps are not limited to the QA and Validation that can be performed and the exact steps should be discussed further to satisfy the QA requirements or needs users have.

Validate the Number of Assets

Once users have tested the workspace, ensure the number of assets match 1:1 for the Legacy workspace and the E2 workspace. In some cases it will not be a 1:1 match, but in those cases, confirm why it's not a 1:1 match and. To easily validate these numbers, run a workspace sizing notebook on the workspaces, below are the numbers that should be validated in this:

1. Number of Users
2. Number of Groups
3. Number of Clusters
4. Number of Notebooks
5. Number of Databases
6. Number of Tables
7. Number of Jobs

Aside from the above list, ensure that the number of instance profiles match between workspaces for this, navigate to Settings>Admin Console>Instance profiles. Take count of the number of profiles.

Workspace Access and Actions

The end goal of this phase of the migration is to ensure that when users log into their new workspace they can continue their usual workflow, uninterrupted. Users should be able to log into the new workspace and see all of the assets and develop as they usually would.

Identify subset of users to test workspace

To begin the validation process, identify power users and admin users that are available to test the new workspace. Once time is set aside, meet and test the workspace with the users by completing the following tasks:

1. Ask users to login to E2 workspace (Through SSO if enabled, but otherwise through the workspace login UI)
2. Ask users to look for their notebooks and account for them
3. Ask users to open a notebook of their's and attach a typical cluster they use, to the notebook
4. Ask users to query external data they would normally query
    a. For this step, assist users in identifying external data they can query
    b. Reminder, the DBFS root will not have been migrated, therefore this data can't be queried
5. Ask users to create a new job from scratch
6. Ask admin users, ask them to create a new cluster from scratch
7. Ask MLFlow users, to login and validate that their artifacts are there

Complete Testing of the Workspace

After initial testing, complete testing of the workspace by verifying that the workspaces and appropriate assets are consistent. This can vary between workspaces and depend mostly on what assets are predominantly used in the workspace. Here are a few things worth looking into:

1. Check the configuration and metadata of sample databases and tables. Specifically, type of the table (EXTERNAL / MANAGED) and the Location of the table
2. Ensure the configuration of all the clusters is the same and
    a. Manually add libraries if needed to the cluster and test whether or not libraries are properly installed during cluster start up
3. Add appropriate init scripts and test whether or not clusters are starting up properly Check the configuration of Jobs. Most notably:
    a. Schedule
    b. Tasks
    c. Libraries
    d. Job cluster Runtime
4. Run production level jobs to see if the run is successful or not
5. Check the connection of external databases such as Redshift, external s3 location (especially if the instance-profiles have been changed/updated)
6. Confirm that the ACLs of notebooks, clusters, jobs and MLflow objects are consistent between workspaces

.

Cutover Planning

Dependencies

- QA + Validation Complete to satisfy QA requirements
- DBFS Root Migration Scheduled and Confirmed

Discuss Cutover Expectations

Moving towards the end of the migration, it's important to discuss and develop an action plan for migrating users and production jobs to the new workspace. Strategizing on how and when workloads will begin running in the E2 workspace and paused in the legacy workspace. For this to begin, QA + Validation needs to be complete and a plan for the DBFS Root Migration needs to be finalized and agreed upon. The below points need to be discussed when evaluating and planning the cutover date from legacy to E2.

- When and how production jobs will be unpaused in the E2 workspace and paused in the legacy workspace
    - When do all job templates need to be migrated in the E2 workspace?
    - Who will be pausing and unpausing the jobs?
    - (If applicable) Who will be reconfiguring airflow or other job orchestration tools?
- When and how users notified of the new workspace and granted full access to the new workspace
    - How will users be told to log into the new workspace?
    - When will the Legacy workspace be no longer accessible to users?
    - When can users log into the E2 workspace?
- When and how external systems will be integrated with the new E2 workspace
    - Who needs to be informed of these things changing?
    - When do the integrations need to be updated and modified?
- When and how will the DBFS migration occur?
    - Confirmed scheduled date for DBFS migration
    - Align cutover date (when users move workspaces) to the expected completion of the DBFS migration

**Note: During the re-import of assets will be re-synced but not overwritten except for notebooks (if --overwrite-notebooks is provided). So if there is a requirement to overwrite the asset e.g. cluster (to update the cluster config) that was migrated during the initial migration phase, the asset will need to be deleted and re-imported.

Identify cutover weekend / timeframe

Cutover refers to the steps required to end usage of the legacy Databricks workspace and to begin solely using the E2 Databricks workspace(s) deployment. There are two methods of

performing cutover: (1) performing the cutover over the course of a weekend, with a “go-live” date of the new workspace at the start of the following weekend, or (2) a phased cutover that can migrate and “go-live” sub-segments of workflows and users.

Asset Mapping

During the migration of workspaces some assets change based on the import to a new workspace. This is not something that can be modified or controlled. To help in these scenarios, documentation can be created to assist end users with the migration transition. Below is a list of documents that we suggest to be provided to impacted end-users:

- Job ID Mapping
- JDBC Mapping
- Repo Listings

These mappings can be created using the REST API to get asset details from both the Legacy and E2 workspaces, comparing asset names and other configs if needed between workspaces.

Reconfigure External Systems

In preparation of the migration close out, be prepared to reconfigure any external tools/systems that are integrated with Databricks. Most impacts and changes should be covered in the above documents, but make sure to understand the impact of any other 3rd party tools integrated with Databricks to make sure things are reconfigured properly.

Final Migration/Cutover Steps

Dependencies

- Cutover Date Discussed and Scheduled
- Asset Mapping Documentation provided

Finalized DBFS Migration

The DBFS migration will occur based on the process discussed and planned for earlier in the migration. Once the DBFS Migration is complete users should not be writing to the old DBFS root bucket and users should be prepared to move to the new E2 workspace.

Workspace Catch Up

While QA and Validation were being conducted on the E2 workspace, users will be working in the Legacy workspace. This means the assets migrated to E2 will become stale over time. In this case, the workspace assets will need to be re-migrated and overwritten in the new workspace. All of the export and import actions will need to be conducted again using the steps provided previously in this document.

Production Job Migration

Before moving any production jobs to the E2 workspace, it is important to assess when production jobs need to be moved and how they can be paused in the Legacy workspace and unpaused in the E2 workspace. Make sure to have a back up plan to back fill the data for business critical production jobs, in the case production jobs fail or can't be paused and unpaused in a timely manner.

Final Validation

To ensure everything has migrated over properly, run the workspace sizing notebook one more time in the Legacy and E2 workspaces to make sure all assets are accounted for.

User Migration

Once the entire E2 workspace is caught up and prepped, users can log into the new workspace. Make sure it is clearly stated to users how and when they can begin using the new E2 workspace. Explain to users what will be happening to the Legacy workspace and when it will no longer be accessible. Clearly state that any changes they make in the Legacy workspace after the cutover date, will not be reflected in the E2 workspace.

Shutting Down the Legacy Workspace

How To

Once you have completed the migration and validation, shut down the workspace by following these steps to cancel your legacy Databricks subscription.

Appendix

Troubleshooting a Deployment

If a workspace fails to launch, or launched correctly but does not pass validation, this section provides some general guidelines on troubleshooting. Note that much of this content is also publicly available on this page.

Workspace fails to launch

- ☐ If the failure occurs because of the Network Object, run the Get Network API to determine what failed (or use the Inspect function on Chrome and view the network API calls while looking at the UI). The most common causes are incorrect Security Group and/or NACL rules; make sure they follow the rules outlined in our documentation.
- ☐ If the failure occurs because of the Storage configuration, make sure that the root bucket policy is set properly.
- ☐ If the failure occurs because of the Credentials configuration, make sure that the role follows our documentation.

Workspace launches, but is inaccessible

- ☐ If you are using front-end PrivateLink, ensure that the traffic to the workspace is flowing through the appropriate endpoint. Also validate the Private Access Setting; if you are not using a front-end endpoint, this should be set to `ANY` and `public_access_enabled` should be set to `true`.
- ☐ If the workspace URL does not resolve (i.e., if `dig my-workspace.cloud.databricks.com` fails), you may need to delete the workspace, wait 5 minutes, and recreate the workspace. This is due to occasional transient DNS registration errors.

Workspace launches, but clusters fail

- ☐ Causes for failed clusters vary, and can be difficult to troubleshoot. The most common failures are due to network connectivity errors. Connectivity errors can occur if you have restricted outbound access, but did not set up regional endpoints, or if PrivateLink was set up incorrectly. Verify that the security groups attached to all endpoints all the cluster nodes to reach them, and that NACLs are not disallowing communication.
- ☐ Bootstrap failure can also occur due to S3 connectivity errors. Make sure that an S3 gateway endpoint exists in the worker VPC, and that a route exists in the worker subnet route tables.

- ☐ Further diagnosis can be done by looking at the AWS EC2 instance logs; this will provide an error message the step that failed in bootstrapping and/or launch.

Other Symptoms

- ☐ If DBFS mounts are not working correctly, make sure the appropriate Spark configurations are set via the documentation.
- ☐ If mounted buckets can be listed but not read, make sure that if the bucket is encrypted using SSE, it is configured appropriately in Databricks (see documentation here).
- ☐ If libraries fail to install, make sure that traffic is appropriately whitelisted. You may be blocking the traffic using a proxy or firewall, for example; they may need to whitelist addresses like pypi.python.org.

Migration Script Error Handling (Debug)

While running the WM script, there can be some errors that may be generic or specific to the workspace you are migrating. While it is also impossible to capture every single error case, we are hoping to list out as many error cases as possible in the table below so that the script invokers can self-mitigate the issues. Please refer to the below table for possible solutions.

Stage that causes the issue	Error Logs / Problems	How to mitigate
ANY	#0 Migration paused due to invalid or expired token. Trying to renew... No new token found. Please renew token by running: $ databricks configure --token --profile staging_src Will try again in 20 seconds. #1 Migration paused due to invalid or expired token. Trying to renew... No new token found. Please renew token by running: $ databricks configure --token --profile staging_src Will try again in 20 seconds.	This happens because the PAT token used for the migration has expired. Simply regenerate the PAT token from the workspace and reconfigure the ~/.databrickscfg in another terminal with the new PAT token.
Export		
Notebook	cluster_id: 1122-193247-saw00005 ERROR : Notebook run failed: https://ES-170495.dev.databricks.com#job/90420709366465/run/36749102081393	Go to the job run url printed in the error message and triage what has gone wrong.

	... http_status code 200 ... notebook execution result_state: FAILED Internal Notebook error, while executing ACL Export	
Table ACL	Stuck at: Get: https://dbc-dcc9b651-86e5.staging.cloud.databricks.com/api/2.0/jobs/runs/get polling for job to finish: https://dbc-dcc9b651-86e5.staging.cloud.databricks.com#job/21667049 1981771/run/997376634255277	Table ACL export is known to take a very long time (depending on the size of the workspace). But feel free to go to the url printed in the message to make sure the job is still running.
Metastore	Failed to execute SHOW CREATE TABLE against table `xxx_db`.`xxx`, which is created by Hive and uses the following unsupported server configuration...	Very likely because of this issue. Workaround: If users run into this, they can use a cluster using a Spark 2.x runtime until we add support for this.
Import		
Cluster	{'error_code': 'INVALID_PARAMETER_VALUE', 'message': 'Workspace restricted to instance types that encrypt in transit. Please specify one such driver node type', 'http_status_code': 400}	This happens when the cluster types used in source workspace are not supported in the destination workspace. (e.g. i3.xlarge is being used in source workspace, but not supported in dst workspace) In this case, you need to work with customers to find out which cluster types to use instead and make the change in clusters.log.
Cluster	{'error_code': 'INVALID_PARAMETER_VALUE', 'message': 'The instance profile arn	This means the IAM role used to deploy destination workspace does not have access to the instance-profile

	(arn:aws:iam::997819012307:instance-profile/shard-demo-s3-access) is not registered in Databricks, or you do not have sufficient permissions to use the instance profile. Please contact your administrator.', 'http_status_code': 400}	arn that is used for one of the source workspace clusters.
Cluster	ValueError: Cluster id must exist in new env for cluster_name: cluster_policy_tests. Re-import cluster configs.	It usually means the cluster it is trying to set the ACL on does not exist in the dst workspace. Look at the log and see if the particular cluster indeed failed to be imported to the dst workspace. For example, a cluster may fail to be imported with the following error message: Creating cluster: cluster_policy_tests post: https://eng-cal-wm-1.cloud.databricks.com/api/2.0/clusters/create {'error_code': 'INVALID_PARAMETER_VALUE', 'message': 'The instance profile arn (arn:aws:iam::997819012307:instance-profile/shard-demo-s3-access) is not registered in Databricks, or you do not have sufficient permissions to use the instance profile. Please contact your administrator.', 'http_status_code': 400} One must manually fix the setting or can remove this cluster from the clusters.log to proceed.
Notebook	File "/Users/kevinkim/migrate/pipeline/pipeline.py", line 71, in _run_task task.run() • File "/Users/kevinkim/migrate/tasks/tasks.py", line 157, in	Retry seems to mitigate the issue.

	run • ws_c.import_all_workspace_items(archive_missing=self.args.archive_missing) • AttributeError: 'Namespace' object has no attribute 'archive_missing'	
Workspace ACL	Fail to import ACLs for directories that doesn't exist. This will print error messages in failed_import_acl_dir.log and terminate the migration pipeline. {"error_code": "RESOURCE_DOES_NOT_EXIST", "message": "Path (/Repos/<email>) doesn't exist.", "http_status_code": 404}	This is happening because the directory is not imported in the destination workspace. Manual investigation is needed to understand why it's not imported. If this is expected, skip the workspace ACL step using "--skip-steps workspace_acls" while kicking the migration-pipeline again.
Metastore	File "/home/ec2-user/dbc-dcc9b651-86e5/migrate/dbclient/TableACLsClient.py", line 89, in copy_files_to_dbfs_path raise Exception(f"not a valid source path: '{source_local_path}'") Exception: not a valid source path: 'logs/20211206035344/table_acls/'	Very likely because your Metastore export failed. Check if the "table_acls" directory exists in your logs/$sessionId directory. If not, it probably means the metastore export failed.