Python Script Integration Guide

This document provides an overview of integrating Python scripts from the `pythonscripts` folder into the AWS Migration App to automate migration steps. It includes a mapping of migration steps to their corresponding scripts, instructions for integrating these scripts as API endpoints, and references for AWS boto3 API documentation for the relevant services. This guide is intended for developers extending the app's functionality by adding new migration steps.

Last Updated: June 26, 2025

Overview

The `pythonscripts` folder contains custom Python scripts designed to automate various AWS migration steps. These scripts encapsulate the logic and AWS operations required for tasks such as creating KMS keys, deploying Control Tower, configuring SSO, and moving accounts. By integrating these scripts into the app's backend, developers can expose them as API endpoints to support the migration process, ensuring automation, maintainability, and scalability.

Step-to-Script Mapping

The following table maps each migration step to its corresponding Python script in the `pythonscripts` folder and the associated step slug used in the app. These scripts can be integrated into `app/services/aws_services.py` to create new API endpoints.

Migration Step	Step Slug	Script File
Create KMS key	`create-kms`	`create_kms.py`
Deploy Control Tower	`deploy-control-tower`	`deploy_control_tower.py`
Raise AWS Organizations service limits	`raise-limits`	`raise_limits.py`
Configure SSO with external IdP	`configure-sso`	`configure_sso.py`
Create new OUs in new Organization	`create-ous`	`create_ous.py`
Replicate SSO Configs - Permission Sets	`replicate-sso`	`replicate_sso.py`

Remove CloudTrail account trail	`remove-cloudtrail`	`remove_cloudtrail.py`
Move non-Prod accounts and register OUs	`move-nonprod`	`move_nonprod.py`
Move accounts to new OUs	`move-accounts`	`move_accounts.py`
Register OU in Control Tower	`register-ou`	`register_ou.py`
Remove SecOps CloudTrail	`remove-secops`	`remove_secops.py`
Duplicate SSO config and test access	`test-sso`	`test_sso.py`
Migrate smaller accounts	`migrate-small`	`migrate_small.py`
Migrate larger accounts	`migrate-large`	`migrate_large.py`
Migrate old payer account	`migrate-payer`	`migrate_payer.py`
Move Production accounts to new Organization	`migrate-prod`	`migrate_prod.py`
Delete CUR report in old payer's S3 bucket	`delete-cur`	`delete_cur.py`
Import existing accounts into Control Tower	`import-accounts`	`import_accounts.py`

Note: If additional scripts are added to the `pythonscripts` folder, update this mapping table accordingly.

How to Integrate a Script as an API Endpoint

To integrate a Python script as a new API endpoint in the AWS Migration App, follow these steps:

Step 1: Locate the Script

- Identify the relevant script in the `server/Backend/pythonscripts` folder that corresponds to the migration step (e.g., `create_kms.py` for the "Create KMS key" step).

Step 2: Move or Reference the Script

- **Recommended Approach**: Move the script's logic into a function in `app/services/aws_services.py` for consistency and maintainability.
    - Example: Convert `create_kms.py` into a function `create_kms_key` in `aws_services.py`.
- **Alternative**: If the script is complex or standalone, import and call it directly from `aws_services.py`.

Step 3: Create an API Route

- In `app/api/routes/steps.py`, import the new function from `aws_services.py`.
- Add a new FastAPI route (typically GET or POST) to call the function, handle input/output, and return results in the expected format (e.g., `StepResponse` model).

Example for "Create KMS key":

```
@router.get("/prepare-new/create-kms", response_model=StepResponse)
async def create_kms(account_id: str = Query(None), db: Session = Depends(get_db)):
    step_id = 101  # Assign a unique step ID
    result = create_kms_key(db, account_id)
    # Save execution to database (using PG_queries.py)
    execution = save_execution(db, step_id, account_id, result)
    return {
        "step_id": step_id,
        "status": result["success"],
        "result": result["data"],
        "logs": [],
        "execution_time": execution.execution_time
    }
```

Step 4: Update Step Mappings

Update the `STEP_IDS` dictionary in `app/api/routes/steps.py` to include the new step:

```
STEP_IDS = {
    # Existing mappings
    "create-kms": 101
}
```

Update the `PHASE_STEPS` dictionary to map the step to the appropriate phase (e.g., `prepare-new`):

```
PHASE_STEPS = {
    "prepare-new": [101],  # Add new step ID
    # Other phases
}
```

- Ensure the frontend uses the correct step slug (e.g., `create-kms`) to call the new endpoint.

Step 5: Test the Integration

- Call the new API endpoint from the frontend (e.g., via `migrationApi.ts`).
- Verify that the script executes correctly and returns the expected results (e.g., check logs and data in the `aws_migration` database via pgAdmin).
- Test the endpoint using FastAPI's Swagger UI at `http://localhost:8000/docs`.

Example Integration: Create KMS Key

To integrate the "Create KMS key" step:

1. **Locate the Script**: Find `create_kms.py` in `pythonscripts`.

Move Logic: Add a function to `app/services/aws_services.py`:

```
def create_kms_key(db: Session = None, account_id: str = None):
    session = get_aws_session(account_id)  # From aws_client_helper.py
    kms_client = session.client('kms')
    response = kms_client.create_key(Description='Migration KMS Key')
    return {"success": True, "data": response}
```

2. **Add API Route**: In `app/api/routes/steps.py`, add the route as shown above.
3. **Update Mappings**: Add `create-kms: 101` to `STEP_IDS` and include `101` in `PHASE_STEPS["prepare-new"]`.
4. **Update Frontend**: Add a button or action in a relevant page (e.g., `MigrationJourney.tsx`) to call `/prepare-new/create-kms` and display results using components like `StepCard.tsx`.

References

For additional details and resources, refer to the following:

1. **Excel Reference**:
    - Check the project's Excel file for a detailed mapping of migration steps, scripts, and additional notes. This file includes step-specific requirements, configurations, or dependencies not covered in this document.
2. **AWS boto3 API Documentation**:
   The scripts in `pythonscripts` use boto3 to interact with various AWS services. Below are the boto3 API references for the services involved in the listed migration

steps:

- **KMS** (Create KMS key): boto3 KMS Documentation
  - Key methods: `create_key`, `describe_key`
- **Control Tower** (Deploy Control Tower, Register OU, Import accounts): boto3 Control Tower Documentation
  - Key methods: `create_landing_zone`, `enable_control`
- **Organizations** (Raise limits, Create OUs, Move accounts): boto3 Organizations Documentation
  - Key methods: `create_organizational_unit`, `move_account`, `list_accounts`
- **SSO Admin** (Configure SSO, Replicate SSO, Test SSO): boto3 SSO Admin Documentation
  - Key methods: `create_permission_set`, `attach_managed_policy_to_permission_set`
- **CloudTrail** (Remove CloudTrail, Remove SecOps CloudTrail): boto3 CloudTrail Documentation
  - Key methods: `delete_trail`, `get_trail_status`
- **S3** (Delete CUR report): boto3 S3 Documentation
  - Key methods: `delete_object`, `delete_bucket`

3. **boto3 General Documentation**:
   - For a complete reference to boto3 and its capabilities: boto3 Documentation
   - Ensure AWS credentials used in `server/Backend/.env` have permissions for these services (configured via AWS IAM).

Final Notes

- **Script Integration**: The scripts in `pythonscripts` are ready to be integrated as API endpoints. Follow the pattern above to move logic into `aws_services.py`, expose it via `steps.py`, and update mappings.
- **Maintainability**: Centralizing logic in `aws_services.py` ensures consistency and easier maintenance compared to calling standalone scripts.
- **Extensibility**: New migration steps can be added by creating new scripts, integrating them as APIs, and updating the frontend to trigger them.
- **Verification**: Always test new endpoints to ensure they interact correctly with AWS services and store results in the `aws_migration` database.