**Step 1 — Create required AWS resources**

* **Prereqs**
* AWS account with permissions for S3, IAM, Lambda, OpenSearch, Textract, Bedrock.
* Bedrock model access approved in your region (e.g., us-east-1).
* AWS CLI configured: aws configure (or SSO).
* **Option A (recommended): Automated setup script**

1) Install deps



  cd D:\RKS\AWS\Prj\AIAgent\intelligent-doc-processing-ai-agent

  python -m venv .venv

  .\.venv\Scripts\Activate.ps1

  pip install -r requirements.txt

  ```

  2) Run setup

  ```powershell

  python examples\setup\_and\_deployment.py

  ```

  3) What it does

  - Checks credentials and Bedrock access

  - Creates S3 bucket (`S3\_BUCKET\_NAME` from config)

  - Creates IAM roles/policies (Lambda exec, service permissions)

  - Creates OpenSearch domain (small, single node)

  - Creates Lambda function(s)

  - Writes `.env` with values/placeholders to update

  4) Edit `.env` after run

  - Set `OPENSEARCH\_ENDPOINT` to your actual domain endpoint (from AWS console or `aws opensearch describe-domain`).

  - Confirm `S3\_BUCKET\_NAME`, `AWS\_DEFAULT\_REGION`, `OPENSEARCH\_INDEX\_NAME`, `BEDROCK\_MODEL\_ID`, `BEDROCK\_EMBEDDINGS\_MODEL`, `LAMBDA\_FUNCTION\_NAME`.

  - Leave `AWS\_ACCESS\_KEY\_ID`/`AWS\_SECRET\_ACCESS\_KEY` blank if you use CLI/SSO; the app uses your default credentials.

- \*\*Option B: Manual CLI (if you prefer explicit control)\*\*

  - Create S3 bucket (adjust bucket/region):

  ```powershell

  aws s3api create-bucket --bucket ai-agent-documents --create-bucket-configuration LocationConstraint=us-east-1 --region us-east-1

  ```

  - Create IAM role for Lambda and attach policies:

    - Trust policy: service `lambda.amazonaws.com`

    - Attach:

      - `AWSLambdaBasicExecutionRole`

      - Access to S3 (bucket-level read/write)

      - `AmazonTextractFullAccess` (or scoped)

      - Bedrock invoke permissions (bedrock:InvokeModel, bedrock:InvokeModelWithResponseStream)

      - OpenSearch permissions (es:ESHttp\* for your domain, or scoped index perms)

  - Create OpenSearch domain:

  ```powershell

  aws opensearch create-domain --domain-name ai-agent-opensearch --cluster-config InstanceType=t3.small.search,InstanceCount=1 --ebs-options EBSEnabled=true,VolumeType=gp3,VolumeSize=20 --region us-east-1

  ```

  - Note the domain “Endpoint” and put it in `OPENSEARCH\_ENDPOINT`.

  - Create a Lambda (optional for local demo; required for serverless orchestration):

  ```powershell

*# after zipping your code*

  aws lambda create-function --function-name document-processor --runtime python3.11 --role arn:aws:iam::<account-id>:role/<lambda-exec-role> --handler orchestration.lambda\_functions.lambda\_handler --zip-file fileb://lambda-deployment.zip --timeout 120 --memory-size 1024 --region us-east-1

  ```

- \*\*Verify resources\*\*

  ```powershell

  aws sts get-caller-identity

  aws s3 ls

  aws opensearch describe-domain --domain-name ai-agent-opensearch

  aws lambda get-function --function-name document-processor

  ```

*### Step 2 — Run the demos locally*

1) Ensure deps installed and venv active (see above).

2) Create `.env`:

```powershell

copy .\config\env\_example.txt .\.env

```

Edit `.env`:

- `AWS\_DEFAULT\_REGION=us-east-1`

- `S3\_BUCKET\_NAME=ai-agent-documents` (or your bucket)

- `OPENSEARCH\_ENDPOINT=https://<your-domain>.<region>.es.amazonaws.com`

- `OPENSEARCH\_INDEX\_NAME=document-embeddings`

- `BEDROCK\_MODEL\_ID=anthropic.claude-3-sonnet-20240229-v1:0`

- `BEDROCK\_EMBEDDINGS\_MODEL=amazon.titan-embed-text-v1`

- `LAMBDA\_FUNCTION\_NAME=document-processor`

3) Run demos:

```powershell

python .\examples\knowledge\_agent\_demo.py

python .\examples\legal\_assistant\_demo.py

```

4) Optional: Deploy/update Lambda code

```powershell

Compress-Archive -Path \* -DestinationPath lambda-deployment.zip -Force

aws lambda update-function-code --function-name document-processor --zip-file fileb://lambda-deployment.zip

```

Tips

- OpenSearch domain creation can take ~10–20 minutes.

- Bedrock access must be granted per region and model family.

- Prefer using AWS CLI/SSO auth instead of placing static keys in `.env`.

2) Run setup



  python examples\setup\_and\_deployment.py

3) What it does

* Checks credentials and Bedrock access
* Creates S3 bucket (S3\_BUCKET\_NAME from config)
* Creates IAM roles/policies (Lambda exec, service permissions)
* Creates OpenSearch domain (small, single node)
* Creates Lambda function(s)
* Writes .env with values/placeholders to update

4) Edit .env after run

* Set OPENSEARCH\_ENDPOINT to your actual domain endpoint (from AWS console or aws opensearch describe-domain).
* Confirm S3\_BUCKET\_NAME, AWS\_DEFAULT\_REGION, OPENSEARCH\_INDEX\_NAME, BEDROCK\_MODEL\_ID, BEDROCK\_EMBEDDINGS\_MODEL, LAMBDA\_FUNCTION\_NAME.
* Leave AWS\_ACCESS\_KEY\_ID/AWS\_SECRET\_ACCESS\_KEY blank if you use CLI/SSO; the app uses your default credentials.
* **Option B: Manual CLI (if you prefer explicit control)**
* Create S3 bucket (adjust bucket/region):



  aws s3api create-bucket --bucket ai-agent-documents --create-bucket-configuration LocationConstraint=us-east-1 --region us-east-1

* Create IAM role for Lambda and attach policies:
* Trust policy: service lambda.amazonaws.com
* Attach:
* AWSLambdaBasicExecutionRole
* Access to S3 (bucket-level read/write)
* AmazonTextractFullAccess (or scoped)
* Bedrock invoke permissions (bedrock:InvokeModel, bedrock:InvokeModelWithResponseStream)
* OpenSearch permissions (es:ESHttp\* for your domain, or scoped index perms)
* Create OpenSearch domain:



  aws opensearch create-domain --domain-name ai-agent-opensearch --cluster-config InstanceType=t3.small.search,InstanceCount=1 --ebs-options EBSEnabled=true,VolumeType=gp3,VolumeSize=20 --region us-east-1

* Note the domain “Endpoint” and put it in OPENSEARCH\_ENDPOINT.
* Create a Lambda (optional for local demo; required for serverless orchestration):



*# after zipping your code*

  aws lambda create-function --function-name document-processor --runtime python3.11 --role arn:aws:iam::<account-id>:role/<lambda-exec-role> --handler orchestration.lambda\_functions.lambda\_handler --zip-file fileb://lambda-deployment.zip --timeout 120 --memory-size 1024 --region us-east-1

* **Verify resources**



  aws sts get-caller-identity

  aws s3 ls

  aws opensearch describe-domain --domain-name ai-agent-opensearch

  aws lambda get-function --function-name document-processor

**Step 2 — Run the demos locally**

1) Ensure deps installed and venv active (see above).2) Create .env:



copy .\config\env\_example.txt .\.env

Edit .env:

* AWS\_DEFAULT\_REGION=us-east-1
* S3\_BUCKET\_NAME=ai-agent-documents (or your bucket)
* OPENSEARCH\_ENDPOINT=https://<your-domain>.<region>.es.amazonaws.com
* OPENSEARCH\_INDEX\_NAME=document-embeddings
* BEDROCK\_MODEL\_ID=anthropic.claude-3-sonnet-20240229-v1:0
* BEDROCK\_EMBEDDINGS\_MODEL=amazon.titan-embed-text-v1
* LAMBDA\_FUNCTION\_NAME=document-processor

3) Run demos:



python .\examples\knowledge\_agent\_demo.py

python .\examples\legal\_assistant\_demo.py

4) Optional: Deploy/update Lambda code



Compress-Archive -Path \* -DestinationPath lambda-deployment.zip -Force

aws lambda update-function-code --function-name document-processor --zip-file fileb://lambda-deployment.zip

Tips

* OpenSearch domain creation can take ~10–20 minutes.
* Bedrock access must be granted per region and model family.
* Prefer using AWS CLI/SSO auth instead of placing static keys in .env.