Instructions for the Cortex Agent Slack Hands-on-Lab

v1.0

INTRODUCTION

This hands-on lab introduces Snowflake's Cortex Agent, demonstrating its value in streamlining data analysis. You'll explore Cortex Search for intelligent document review and Cortex Analyst for extracting insights from structured data. The lab focuses on integrating pre-written Python code, but requires setting up Snowflake and Slack trials, configuring a Slack app, and building out a Python environment.

Through this "final assembly" process and a focus on the underlying technology, you'll gain practical experience in how Cortex Agent combines these powerful functionalities, showcasing its ability to efficiently retrieve and synthesize information from diverse data sources, ultimately accelerating your data-driven decision-making.

Key Value Propositions

By better understanding what's possible when integrating Snowflake's Cortex Agent into Slack, users will gain significant value:

- **Real-time Insights**: Get instant answers to data-related questions directly within Slack conversations, eliminating context switching between applications and speeding up access to critical information.
- **Democratized Data Access**: Empower non-technical users to query and understand data using natural language, removing the need for specialized SQL expertise and fostering data literacy across teams.
- **Streamlined Decision-Making**: Facilitate quicker, more data-informed decisions by making insights readily available precisely where team discussions and collaborative work happen.
- **Improved Efficiency**: Drastically reduce the time spent manually searching for information across diverse platforms by leveraging the combined power of Cortex Search and Analyst directly within Slack.
- **Proactive Problem Solving**: Identify and address issues faster by having Cortex Agent surface relevant information from documents or structured data proactively, allowing for timely interventions.

ARCHITECTURAL OVERVIEW

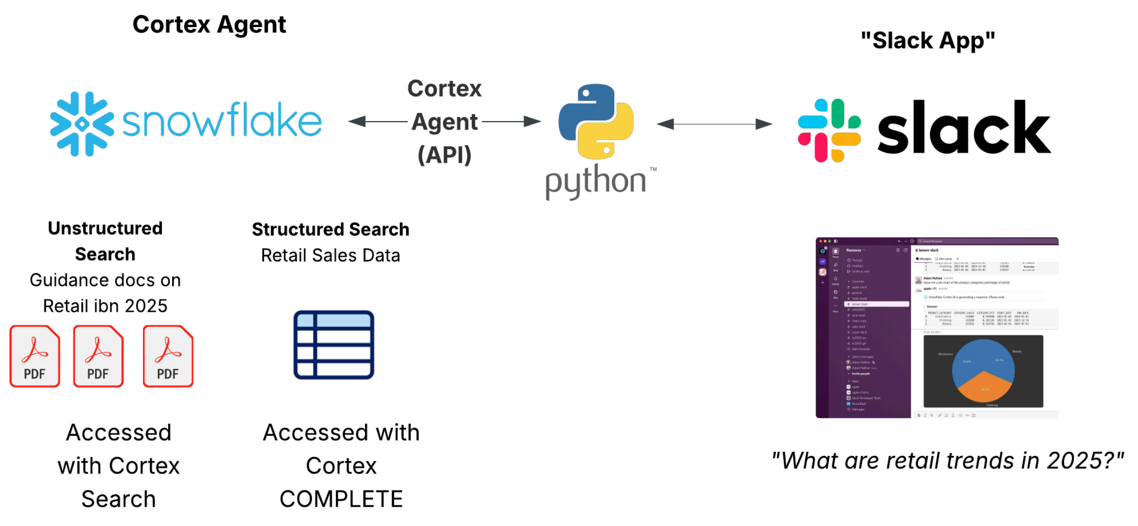
Solution Architecture: Three Layers

This solution employs a streamlined three-layer architecture for conversational data analysis in natural language:

1. Presentation Layer (Slack App): This is the user's interface. The Python-based Slack bot receives queries and presents formatted results, charts, or text answers directly within Slack, also managing dynamic interactive elements like viewing a dataset or chart's underlying SQL.

2. Application/Orchestration Layer (Python Bot & Cortex Agent): The Python bot receives and transmits prompts from Slack and orchestrates communication with the Snowflake Cortex Agent API. It manages the creation of charts, the display of query results, and the display of datasets.

3. Data & Intelligence Layer (Snowflake with Cortex Agent): Residing within Snowflake, this foundational layer retrieves query results from both structured data and unstructured documents. Cortex Analyst performs natural language to SQL conversion for structured queries, while Cortex Search intelligently retrieves vector-embedded information from your documents (in this lab, a collection of PDFs). All data processing and AI inferencing occur within Snowflake's security perimeter.



LAB PREREQUISITES

Here are the key skills and resource requirements a participant should possess prior to beginning this hands-on lab:

Skills

- **Command Line / Terminal Proficiency:** Ability to open and navigate a command line or terminal, and execute basic commands (e.g., cd, git clone, python, pip).
- **Python Fundamentals:** Some familiarity with Python syntax, the importing of libraries, understanding how to execute Python scripts, and the concept of virtual environments. Participants should already have **Python installed** on their machine.
- **SQL:** A basic understanding of SQL queries and the difference between structured and unstructured data.
- **Familiarity with LLMs and Generative AI:** A conceptual understanding of what Large Language Models (LLMs) are and how generative AI works.
- **Text Editor Proficiency:** Ability to open, edit, and save plain text files (like `.env` files).
- Slack: the user should have a basic familiarity with Slack to understand its basic function and have the ability to create a channel.

Tools

- A computer with python3 installed
- This lab requires the use of two trial licenses for free software: a Snowflake Trial and a Slack Trial. They are straightforward to acquire and configure. Ensure that you have enterprise permission to configure these applications either under your enterprise email or a personal email account.
- This guide is developed for Mac. The majority of the instructions here will apply to other environments as well, but you may need to look at slight variations in code to execute, e.g. the creation and activation of Python environments

BUILDING THE APPLICATION

The development of this lab contains the following steps:

1. **Get Your Snowflake Trial Account**
    a. Acquire a trial Snowflake account suitable for this lab.
2. **Clone a GitHub repository**
    a. Access and clone a GitHub repository to download the lab's application files
3. **Configure Python**
    a. Configure a Python virtual environment with required libraries
4. **Setup Key-Pair Authentication**
    a. Configure secured access with Snowflake
5. **Configuring Snowflake**
    a. Build out the database, schema and stage objects to hold project files
6. **Load Retail Sales Data**
    a. Upload a .csv file through the Snowsight interface
7. **Load Semantic View**
    a. Upload a completed .yaml file, the semantic layer to describe the retail data.
8. **Load PDF Content & Create Cortex Search Service**
    a. Upload the unstructured PDFs that will be used for our Cortex search service
    b. Configure Cortex Search to parse, chunk and load the unstructured PDF data
9. **Build, Secure and Integrate Your Slack Application**
    a. Leverage a pre-configured manifest .json file to simplify Slack app creation.
10. **Testing Your Application End-to-End**
    a. Validate the end-to-end connectivity and functionality of the application
11. **Clean Up**
    a. If desired, the scripts to drop the lab elements

Get Your Snowflake Trial Account

It is mandatory to use a Snowflake trial account for this lab. Not all Snowflake cloud providers and regions have the full set of LLM models and functionality available for the Cortex features you'll be using here.

Additionally, a trial account lets you act as an ACCOUNTADMIN for your dedicated environment during the lab, simplifying setup with no security implications for your organization's production data.

The Snowflake trial comes with $400 of free credits.

Obtaining a Snowflake Trial

1. Navigate to the Snowflake trial signup page: **https://signup.snowflake.com/**
2. Fill in straightforward personal and company details.
3. On the next screen, for the configuration options, be sure to select:
   - **Region:** United States
   - **Cloud Provider:** Amazon Web Services (AWS)
   - **Edition:** Enterprise
4. Finally, click the **"Get started"** button.
5. Validate the trial through the email you will receive.
6. Log in to verify connectivity

Retrieve Your Account/Server URL

Once your Snowflake trial account is active, you'll need to retrieve your unique **Account/Server URL** from Snowsight, Snowflake's web interface. This URL is essential as it's how your Python bot will connect to your specific Snowflake instance.

1. In Snowsight, locate and click your **username** in the lower-left corner to open the user menu.
2. From the menu, navigate to the **Account** submenu.
3. Click on **View account details** next to your active account.
4. An **Account Details** page will open.
5. Find and make a note of
   a. **Your User Name**
   b. **Account Identifier**
   c. **Account/Server URL**
   d. **Account Locator**

Clone a GitHub Repository

Clone Repository and Set Up Python Environment

Next, you'll set up your local Python environment and clone the lab code from GitHub.

1. **Create a Project Directory:** Create a folder
2. **Clone the Repository:**
    a. On your machine, open a terminal or command prompt.
    b. Navigate to a designated folder where you want to store the lab files.
    c. Then, clone the lab repository using the following command:

```
None
      git clone
      https://github.com/sfc-gh-anathan/saks-cortex-agents-with-slack.git
```

3. **Validate** you have a new folder named **saks-cortex-agents-with-slack** containing a full suite of project files.

Configure Python

Update Environment Variables Part I

You will configure program variables to access Snowflake in Part I. In Part II, further down, you will enter the variables to enable connection to Slack.

1. Locate the **fill-in-the-env.txt** file in the root directory.
2. **Rename** this file to .env (There is no file name, only the ".env" extension)
3. Open the .env file in a text editor. You will receive a warning about a file name starting with a dot. Ignore.
    a. Validate that there is no extension (like .txt) that is hidden from site.
    b. To do this: right click on the file and validate that the Name & Extension has ONLY **.env**
4. You will update the following placeholder values in the **.env** file with your specific details:
    a. **Snowflake Account Details:** Update
        - ACCOUNT
            - Using the Account/Server URL you noted during the Snowflake Trial install above, enter it in the pattern displayed in the **.env** file
        - HOST
            - Account/Server URL
        - AGENT_ENDPOINT
            - Add the Account Locator before the ".us-east-1"
        - DEMO_USER (the name you created to log in to the Snowflake Trial
        - DEMO_USER_ROLE (use ACCOUNTADMIN)
        - SLACK TOKENS: Leave them as placeholders for now.

Set Up Python Virtual Environment

1. Open a Terminal
2. Ensure your system has python3 installed before proceeding (see code below)
3. In the **saks-cortex-agents-with-slack** directory, execute the commands below. They will create an isolated Python environment, install all necessary libraries, while preventing conflicts with other Python projects on your machine.
4. *Note: These instructions are for running the hands-on lab on a Mac. The source .venv/bin/activate command might differ slightly for Windows users (e.g., .\.venv\Scripts\activate).*

```
None
# Verify you are running Python3. (You will see a version number if you are.)

python3 --version

# Create a virtual python3 environment named .venv

cd saks-cortex-agents-with-slack

python3 -m venv .venv

# Activate the virtual environment

source .venv/bin/activate

# Verify that you are using the virtual environment's Python

which python3

# you should see: saks-cortex-agents-with-slack/.venv/bin/python3

# Install the required Python libraries

pip install -r requirements.txt
```

Setup Key-Pair Authentication

You will generate the RSA public and private key pair required for securely authenticating your Python application with Snowflake. These keys will be stored directly in the root directory of your project (the saks-cortex-agents-with-slack folder).

1. **Generate Private Key:** Open your terminal or command prompt (ensure you are still in your activated virtual environment within the project root directory). Execute the following command to generate your private key (rsa_key.p8). This command generates a 2048-bit RSA private key and saves it as rsa_key.p8. The -nocrypt option is used for simplicity in this lab to avoid password protection on the private key. In a production environment, you would typically encrypt this file with a passphrase.

```
None
openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out rsa_key.p8 -nocrypt
```

2. **Generate Public Key:** Now, use the private key to generate its corresponding public key **rsa_key.pub**. This command extracts the public key from your newly created private key and saves it in a file named **rsa_key.pub**. You will need the content of this public key file for the next step in Snowflake.

```
None
openssl rsa -in rsa_key.p8 -pubout -out rsa_key.pub
```

3. **Validate file creation:** After executing these commands, you should see **rsa_key.p8** and **rsa_key.pub** files in your project's root directory.

Configuring Snowflake

This section guides you through the essential configurations within your Snowflake trial account to enable the lab's functionalities.

You will set up secure key pair authentication for your Snowflake user, create foundational database objects and stages, and then load the necessary semantic model and PDF documents.

Finally, you will deploy a Cortex Search Service, allowing your application to leverage Snowflake's AI capabilities for both structured data analysis and unstructured document review.

Install Your Public Key into Snowflake

To allow your Python bot to securely connect to Snowflake, you'll associate the public key you generated with your Snowflake user.

1. Open a SQL worksheet in Snowsight
    a. You can find Worksheets in the left hand pane:
        - **Command Line Icon** >> **Project** >> **Worksheets**
2. Open your **rsa_key.pub** file in a plain text editor. This is the one you created earlier.
    a. The file will be in the root of your python project
3. Carefully copy **only the multi-line string of characters** that sits *between* the -----BEGIN PUBLIC KEY----- and -----END PUBLIC KEY----- lines.
4. Paste this content *between the single quotes* in the RSA_PUBLIC_KEY parameter of the ALTER USER statement below.
5. Remember to replace **YOUR_USER_NAME** with your actual Snowflake username in both the **ALTER** statement and the **SELECT** statement (this will match your **DEMO_USER** in .env and is also the one you use to login to Snowflake..
6. Execute the following SQL command in Snowsight to associate the key to a user
7. Run the **SHOW USERS** and **SELECT** statements. You should see **'true'**.

```
None
ALTER USER YOUR_USER_NAME SET RSA_PUBLIC_KEY =
'MIIBIgkqh...YOUR_BASE64_PUBLIC_KEY_CONTENT_GOES_HERE_ONLY...';


SHOW USERS; -- this must be executed directly before the SELECT statement below


SELECT "has_rsa_public_key" FROM TABLE(RESULT_SCAN(LAST_QUERY_ID())) where
"name" = 'YOUR_USER_NAME';
```

Create Foundational Objects

Now, you will set up the basic database, schema, warehouses, and stages required for the lab. The provided SQL will create these objects and select them for use.

- **Execute Setup SQL:** Using the same SQL worksheet paste the code from the **setup.sql** file in the root of your project. Run all statements from top to bottom.
- **Validate Environment:** In Snowsight, navigate through the **Data** section in the left hand menu pane.
  - **Database Icon >> Databases >> SLACK_DEMO**
    - You may need to refresh the list of databases
- Fully open all items below **SLACK_DEMO >> SLACK_SCHEMA**
- Visually confirm that you see:
  - **SLACK_DEMO** database,
  - **SLACK_SCHEMA** schema
  - **SLACK_SEMANTIC_MODELS** stage has been created
  - **SLACK_PDFS** stage has been created

Load Retail Sales Data

This step loads the structured sales data into a Snowflake table that Cortex Analyst will use.

- In Snowsight's left hand menu, navigate to Data
    1. **Database Icon >> Add Data >> Load data into a Table**
    2. Click "Browse" and select the **retail_sales_dataset.csv** file from your project (it will be in the root folder)
    3. Select **SLACK_DEMO.SLACK_SCHEMA** for "Select or create a database and schema"
    4. For "Table name", enter **RETAIL_SALES_DATASET**.
    5. Review the file format options and click "**Next**" and then "**Load**".
- Validate: Click through to "**Query Data**". You should see various transaction records. The query path should be:
    1. **"SLACK_DEMO"."SLACK_SCHEMA"."RETAIL_SALES_DATASET"**

Load Semantic View

The semantic model (**retail_sales_data.yaml**) is crucial for Cortex Analyst to understand your structured data in natural language queries.

- **Upload Semantic Model:**

    1. In Snowsight, navigate to
        - **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA >> Stages** >> **SLACK_SEMANTIC_MODELS**.
    2. Click the **"+ Files"** button (or similar upload icon) to upload files.
    3. Select the **retail_sales_data.yaml** file from your project folder and **Upload**.
    4. Visually confirm you see the **retail_sales_data.yaml** file in the stage.

Load PDF Content & Create Cortex Search Service

For Cortex Search to review documents, you'll need to load sample PDF documents into the **SLACK_PDFS** Snowflake stage.

These documents will then be parsed and chunked by the Snowflake Cortex service. This will first extract the text from the PDFs and then create small chunks of text that are necessary for effective embedding, the underlying process of Cortex Search.

Finally, you'll initiate a Cortex Search Service to enable high speed retrieval of the text within the PDFs.

- **Upload PDF Documents (Using the Stage Interface):**

    1. In Snowsight, navigate to **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA >> Stages >> SLACK_PDFS**.
    2. Click the **"+ Files"** button (or similar upload icon) to upload files.
    3. Navigate to the **Data** folder within the root of your project directory and select all PDF files and **Upload** them to the stage.
- **Parse and Chunk PDFs:** Now, cut, paste and run all SQL commands in the **cortex_search_service.sql** file in the root of the project directory. This SQL will use Cortex functions to load the Cortex Search service with your PDF data.
    1. NOTE: THIS MAY TAKE FROM 2 - 5 MINUTES
- **Validate Creation of Chunked PDFs:**
    1. In Snowsight, navigate to **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA** >> **Tables**. Visually confirm that the **PARSE_PDFS** and **PARSED_PDFS** tables have been created and contain data
    2. Examine the content within each by selecting the table name and then selecting "**Data Preview**" in the header menu for the table.
- **Create Cortex Search Service**
    1. This step deployed the Cortex Search service, allowing your Cortex Agent to perform intelligent searches over your loaded PDF content.
- **Validate**
    - Navigate to **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA** >> **Cortex Search Services**.
    - Visually confirm that **INFO_SEARCH** is listed
    - Select **INFO_SEARCH**, then "View Details" at upper right
    - Verify the status is "active" for both Serving and Indexing.
- **Explore**
    - Select "Data Preview" and note how the page content is broken into chunks and the title of the source document is listed along with the file's path. At far right, you'll see an "embedding" (not human readable).
    - Select **Playground** at upper right. You can search against the data in the UI.

Build, Secure and Integrate Your Slack Application

This section guides you through integrating your bot with Slack. You will create a dedicated Slack channel, provision your Slack App using a manifest file with pre-configured settings for application creation, install the application into your workspace, and retrieve the essential API tokens.

Finally, you'll add (“invite”) your bot to the Slack channel you have created and perform a quick check to confirm the integration is successful.

Create Your Slack Trial Account

To set up your Slack trial account:

1. Go to **https://slack.com/get-started#/createnew**.
2. Follow the prompts to create a new Free Trial Slack workspace.

Create Your Slack App from a Manifest

You'll create the Slack App using a provided manifest.json file, which pre-configures your application’s features and permissions.

1. Go to the Slack API dashboard: **https://api.slack.com/apps**.
2. Click the **"Create New App"** button.
3. Select **"From a manifest"**.
4. Choose the **workspace** where you want to install this app (your trial workspace). Then click **Next**.
5. Open your **manifest.json** file in the root of your project folder in a text editor. **Copy its entire content** and paste it into the provided JSON editor on the Slack API page.
6. Click **"Next"**, then **"Create"** to finish creating the app.
7. You will navigate to a “**Basic Information**” page for the application itself.

Install the App and Retrieve Tokens

After creating the app, you need to install it to your workspace and obtain the necessary authentication tokens. Here’s how to find the two tokens that are critical for your Python middle-tier code to connect to Slack

1. On the app's settings page, navigate to **Install App** >> **Install to Workspace** in the left sidebar.
2. Review the requested permissions and click **"Allow"**.
3. A Bot User OAuth Token will be displayed. Copy it

4. Upon successful installation, you will be redirected to the "OAuth & Permissions" page. Here in **Settings >> Install App**, you'll find your **Bot User OAuth Token** (starts with xoxb-)
5. Open your .env file and paste it after **SLACK_BOT_TOKEN**
6. In Slack navigate to **Settings >> Basic Information >> App-Level Tokens**
7. Select “**Generate Token and Scopes**”
8. Call your Token Name “slack-app-token”
9. Add a “Scope” of **connections:write**
10. Select **Generate**
11. Copy the Token (will start with ‘xapp’) and paste into your **.env** file for **SLACK_APP_TOKEN**

Create the Lab Channel

First, create a new public channel in your Slack workspace where your bot will interact.

1. In your Slack workspace, select the “**Home**” icon at the top of the left-hand menu bar
2. Select the dropdown for the **Channels section header**
3. Select “**Create**” and then **"Create a channel"**.
4. Name the channel “**<COMPANY_NAME> SLACK”** and ensure it is set to **Public**.
5. Click **"Create"**.

You can skip inviting others for now.

Invite the Bot to Your Channel

Even after installation, the bot needs to be explicitly invited into channels where you want it to participate.

1. Go to your newly created “**<COMPANY_NAME> SLACK”** channel in Slack.
2. In the Slack message box, type “**/invite @COMPANY_NAME> Slack”**.
3. Select your bot from the autocomplete list and press Enter.

Testing Your Application End-to-End

Test Slack Connection

This step verifies that your Python environment can successfully connect to your Slack App via Socket Mode and receive/send messages. First we need to add tokens in Python so that it can secretly connect to Slack. These are stored in the .env file. You saw them earlier when setting values in the file.

Activate the Python environment (if it is not already)

Open your terminal and activate the virtual machine

```
# Activate the virtual environment


source .venv/bin/activate
```

Execute the Slack Connection

1. Execute the test-slack-connection.py script. You'll find it in the root of the project directory.

```
None
python test-slack-connection.py
```

2. You should see a debug section and the values for your two tokens
3. This section will be followed by **"⚡ Bolt app is running!"**
4. Success

Validate Connectivity in Slack

1. Make sure you see output in your terminal indicating that the "⚡️ `Bolt` app is running!"
2. Go to your **#slack-bot** channel in Slack.
3. Type "hello" (exact case)
4. Your bot should respond with: "**Hey there @your_username**!"

If your bot responds, your Slack integration is successful, and your environment is ready to proceed with the main lab application!

If you are not successful:

1. Double-check your .env tokens for accuracy
2. Validate that the tokens were visible after running test-slack-connection.ppy.
   a. If no, it may be that the .env file cannot be accessed because it has a hidden extension like ".txt" Remove this through the File Info context menu.
3. Make sure you've added the application to the channel.
4. Make sure that @SAKS SLACK has been invited to the channel.

Running the Solution

Now that your environment is fully configured, it's time to launch the application and begin exploring its capabilities.

This solution showcases the power of Snowflake's Cortex Agent in combining structured data analysis with unstructured document review. You'll interact with the bot in natural language, asking questions that leverage **Cortex Analyst** to generate insights and potentially various graph types from structured data, or tapping into **Cortex Search** to retrieve answers directly from your loaded PDF documents.

1. **Launch the Application:** Ensure your Python virtual environment is still activated in your terminal (if not, run source .venv/bin/activate). Then, execute the app.py script:

```
None
python app.py
```

2. As above, you should see output in your terminal indicating that the "⚡`Bolt` app is running!"

3. **Explore in Slack:**

    ○ Go to your #slack-bot channel in Slack.
    ○ Begin by asking natural language questions. Try queries related to the **retail_sales_dataset** (structured data) to see how Cortex Analyst generates results and charts, for example:
        ■ "Show me total sales by month."
        ■ "What were the top 3 product categories by unit sales is 2023?"
        ■ "What is the difference between sales for men and women? Show by age range."
        ■ "Are older customers buying more expensive items than younger customers"
    ○ Then, ask questions about the content of the PDF documents you uploaded (unstructured data) to see Cortex Search in action, for instance:
        ■ "Is the retail environment bullish?"
        ■ "What can you tell me about Saks?"
        ■ "Which companies are focused on sustainability?"
    ○ Don't forget to click the "Show SQL Query" button that appears with structured data results to see the underlying SQL generated by Cortex Analyst!

Clean Up

To clean up the trial environment, run **cleanup.sql**.

This script will drop the **SLACK_DB** database, the objects that it contains and the **SLACK_S** warehouse we created for this project.

Key Documents & Resources

Here are the key documents and their specific URLs that would be most helpful for participants during this lab:

1. **The Lab's GitHub Repository:** This is your primary source for all the project code, manifest files, semantic model, and sample data.
    - **URL:** https://github.com/sfc-gh-anathan/saks-cortex-agents-with-slack
2. **Snowflake Cortex Documentation:** For a deeper understanding of Snowflake's built-in AI capabilities, including Cortex Agent, Cortex Search, and Cortex Analyst functions.
    - **URL:** https://docs.snowflake.com/en/developer-guide/cortex/
3. **Slack Bolt for Python Documentation (Getting Started):** This resource is excellent for understanding how the Python bot interacts with the Slack API, especially regarding event handling and basic bot functionality.
    - **URL:** https://tools.slack.dev/bolt-python/getting-started/

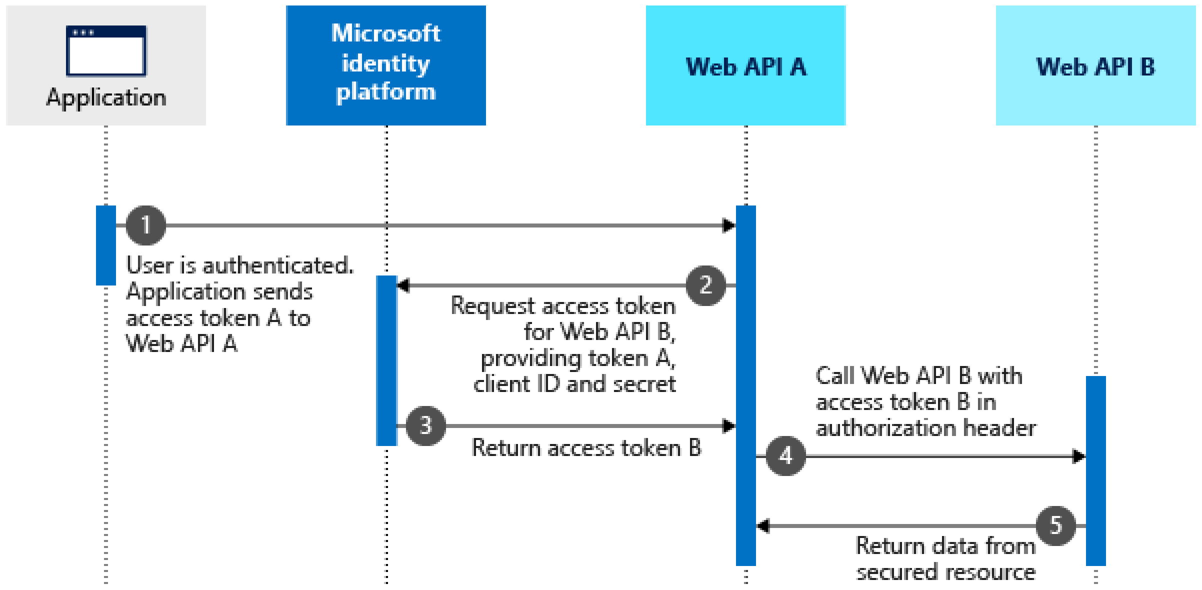
Appendix A

Securing the Solution

- Access to Snowflake is secured through key-pair/PAT authentication mapped to a service user
- The middle-tier python application interacts with the Slack application through OAuth access “bot tokens”. These are the most common and recommended token type for most Slack apps. They are associated with your app's bot user and are granted once per workspace where your app is installed. They allow your app to act independently of a specific user.
- The key security challenge is ensuring that the human user interacting with Snowflake has the correct RBAC role. This is managed through “on-behalf-of” security illustrated below. The image is taken from Microsoft “on-behalf-of” documentation, but the principles apply for OAuth authentication systems as a whole.

Security Flow

1. The client application makes a request to API A with token A (with an aud claim of API A).
2. API A authenticates to the identity platform token issuance endpoint and requests a token to access API B.
3. The identity platform token issuance endpoint validates API A's credentials along with token A and issues the access token for API B (token B) to API A.
4. Token B is set by API A in the authorization header of the request to API B.
5. Data from the secured resource is returned by API B to API A, then to the client.


Application
Microsoft identity platform
Web API A
Web API B
1
User is authenticated. Application sends access token A to Web API A
2
Request access token for Web API B, providing token A, client ID and secret
3
Return access token B
Call Web API B with access token B in authorization header
4
5
Return data from secured resource