**User Guide: Automated Deployment Pipeline with Snowflake and Git Integration**

**Overview:**

This Python script automates the deployment process of code changes from a development branch (dev) to a testing branch (test), and then to a user acceptance testing branch (uat), and finally to a production branch (prod). It captures code changes, uploads them to Git, deploys the changes to Snowflake databases in different environments, captures deployment metadata, checks deployment status, and notifies stakeholders via email.

**Prerequisites:**

**Snowflake Account:** Ensure you have access to Snowflake databases in the specified environments (test, uat, prod).

**Git Repository:** Set up a Git repository where your code changes will be pushed and pulled.

**SnowSQL:** Install SnowSQL and configure connection profiles for test and uat environments.

Use the following parameters for the dev environment:

**user**: ‘user\_name’ ,

**password**: ‘password’ ,

**account**: ‘account\_url’ ,

**warehouse**: ‘warehouse\_name’ ,

**database**: ‘database\_name’ ,

**schema**: ‘schema\_name’

**note**:- give required parameter to the above field

**Python Environment:** Make sure you have Python installed on your system.

**Required Python Packages**: Install the necessary Python packages using ‘**pip install snowflake-connector-python pandas GitPython’**

**Configuration**:

**Connection Profiles in .snowsql config :**

**[connections.testcli]**

**#Can be used in SnowSql as #connect example**

**accountname = ‘**account\_url’

**username** = ‘user\_name’

**password =** ‘password’

**dbname =** ‘database\_name’

**schemaname =**  ‘schema\_name’

**[connections.uatcli]**

Same as testcli

**Connection Profiles:**

**connection\_profile\_test**: SnowSQL connection profile for the test environment.

**connection\_profile\_uat**: SnowSQL connection profile for the user acceptance testing environment.

**sf\_params\_dev**: Snowflake connection parameters for the development environment.

**Git Configuration:**

**git\_url**: URL of the Git repository.

**git\_local\_repo\_path**: Local path to the Git repository.

**dev\_branch**: Name of the development branch.

**test\_branch**: Name of the test branch.

**uat\_branch**: Name of the user acceptance testing branch.

**prod\_branch**: Name of the production branch.

**Email Configuration:**

**sender\_email**: Email address from which notifications will be sent.

**receiver\_email**: Email address to which notifications will be sent.

**password:** Password for the sender email account.

**email\_subject**: Subject of the notification email.

**email\_body**: Body of the notification email.

**Usage:**

**Capture Code Changes:**

The script uses the capture function from snowflake development environment, to capture code changes from the dev branch and upload them to the test branch in the specified Git repository. It also sends an email notification about the code sync.

**First Deployment (dev -> test -> uat):**

* The script deploys the code changes from the ‘**test**’ branch to the Snowflake test environment specified in the ‘**connection\_profile\_test’** variable.
* Deployment logs are captured and stored in a CSV file for reference.

**Check Deployment Status and Notify:**

* The script checks the deployment status in the test environment and notifies particular manager via email.
* If the deployment is successful, it proceeds to deploy the changes to the user acceptance testing environment (uat branch) using the deploy function.

**Second Deployment (uat -> prod):**

* The script deploys the code changes from the ‘**uat’** branch to the Snowflake user acceptance testing environment specified in the ‘**connection\_profile\_uat’** variable.
* Deployment logs are captured and stored in a CSV file for reference.
* The script then checks the deployment status in the user acceptance testing environment and notifies particular managers via email.

**Notes:**

* Ensure that the Snowflake connections, Git repository, and email configurations are correctly set up before running the script.
* Regularly check the email notifications for deployment status updates.
* Make sure to handle sensitive information (like email passwords) securely and consider using environment variables or other secure methods for storing sensitive data.
* In case of deployment failures, review the logs and Snowflake connection parameters to diagnose the issue.