Contents

Objective:

Sample Application to watch a folder in local file system and upload/download to/from S3. Upon successful upload to S3, a message is queued to SQS for other applications (consumer) to download the files from S3 and process further.

Technology Used:

Python is used as programming language along with following packages & external services.

Programming language	Python 2.7	
Python packages	boto3	AWS
	botocore	AWS
	ConfigParser	In-built
	log4py	https://github.com/maihde/log4py
AWS Services	S3	AWS
	SQS	AWS

Prerequisites:

The application requires Python 2.7 and packages [boto3, botocore, ConfigParser].

Configuration:

Application reads configurations from "*Application.properties*" during start-up, below keys need to be configured before executing application. The key names are self-explanatory.

Configuration Section	Notes

`[General]`
`logFilePath=s3uploadTest-Atanu.log`
`folderToStoreTempFiles=TEMP/Atanu`
`folderToWatch``=Watch`
`folderQueuedForUpload=QueuedForUpload`
`folderForUploadedFiles=UploadedToS3`
`folderForDownloadedFiles``=DownloadedFromS3`
	`folderToStoreTempFiles`: the root directory relative to "main.py" directory, for keeping temporary files.
	`folderToWatch`: Directory relative to folderToStoreTempFiles to provide files for upload to S3.
	`folderForDownloadedFiles`: Directory relative to folderToStoreTempFiles for storing files downloaded from S3.

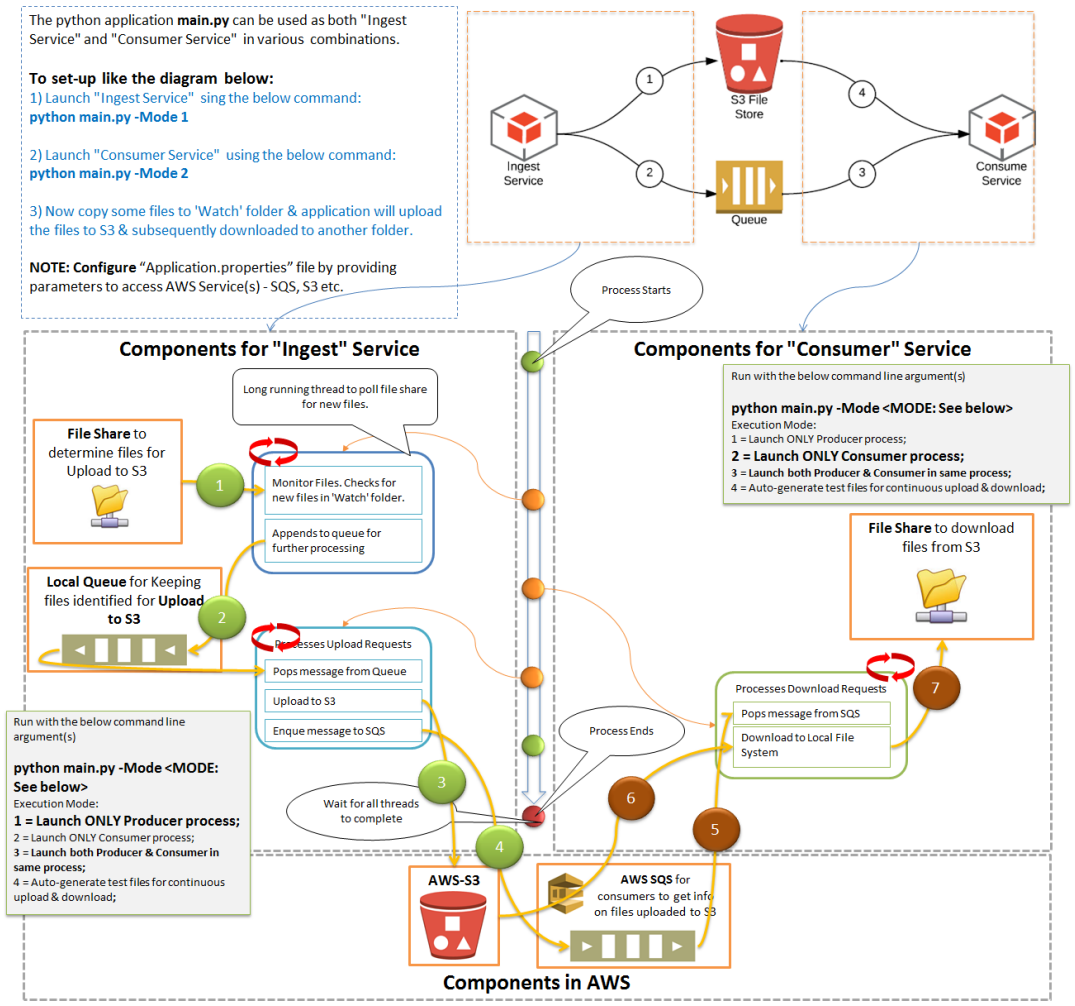
`[AWS]`
`aws_s3_bucket=icd-users`
`aws_s3_folder=atanu.banik/s3uploadTest`
`aws_access_key_id=`
`aws_secret_key=`
`aws_region=us-west-2`
`aws_sqs=atanu_banik`
	The **highlighted** keys are Mandatory
	
	NOTE:
	1. Application tries to create SQS if not present. But it requires admin privileges for SQS service in absence of this the application will crash.
	2. Application doesn't try to create S3 bucket if not already present. Application will crash if the S3 bucket is not present.
	3. Access & Secret Keys are optional. Logged in user's credential will be used if kept empty.

[Concurrency] sleepWatchLocalFolder=6.0 sleepUploadToS3=3.0 sleepPollSQS=2.0	Used by the following threads/processes to sleep after every iteration of processing a request or polling for request.

[Environment]
# Manual File Upload == 0 	
# Automatic File Upload == 1 	
operationMode=0
	Not in use!
	Command line parameters are used for this. An example is given below:
	
	Run with the below command line argument(s)
	
	python main.py -Mode <MODE: See below>
	Execution Mode:
	1 = Launch ONLY Producer process;
	2 = Launch ONLY Consumer process;
	3 = Launch both Producer & Consumer in same process;
	4 = Auto-generate test files for continuous upload & download;

Design

The following diagram depicts Architecture, Design and Data Flow Diagram.



Dedicated Threads (multiprocessing in Python):

1) **Process: Watch for new files for upload**
   A dedicated thread (multi-process in python) monitors '*Watch'* folder for new files. New files are picked up for S3-Upload and moved to the next folder named *'QueuedForUpload'*. The files are also queued for S3-Upload in a *local queue* shared across processes.
2) **Process: Upload to S3**
   A dedicated thread consumes files from the *local queue* and uploads the file to S3. Upon successful upload the S3Key of uploaded file is queued in SQS for other applications to download. Upon successful file upload to S3, the candidate files are moved to this folder.
3) **Process: Poll SQS to download files to S3**
   A dedicated thread consumes messages from SQS & downloads the files from S3.

Getting Started with the Application

Execute Application

The python application **main.py** can be used as both "Ingest Service" and "Consumer Service" in various combinations. Refer to above diagram for detail.

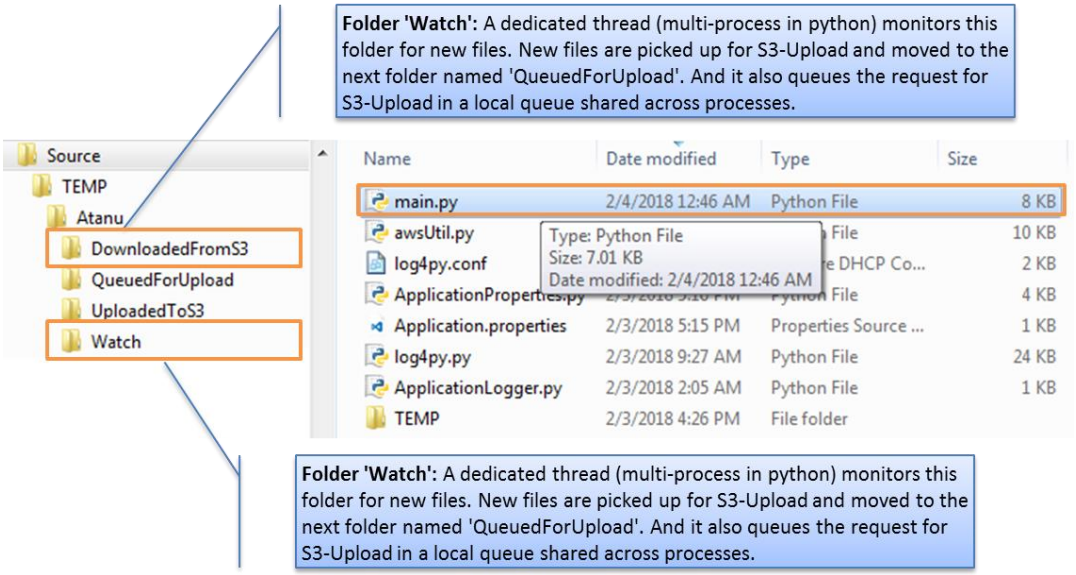
To set-up like the diagram above:

1) Launch "Ingest Service" sing the below command:
python main.py -Mode 1
2) Launch "Consumer Service" using the below command:
python main.py -Mode 2
3) Now copy some files to 'Watch' folder & application will upload the files to S3 & subsequently downloaded to another folder.

NOTE: Configure "*Application.properties*" file by providing parameters to access AWS Service(s) - SQS, S3 etc.

Upload files

During start-up, application creates the below sub-folders under the root folder of "main.py". The folder relative to the root folder of "main.py" can be set by configuration key: '*folderToStoreTempFiles'*.



Temporary Folder Configuration	Remarks
`logFilePath`=s3uploadTest-Atanu.log	Application creates log file with this name under directory: ***folderToStoreTempFiles***
`folderToStoreTempFiles`=TEMP/Atanu	The directory relative to "**main.py**" directory, for keeping the following files: a) Application log file b) files for uploaded to S3 – User needs to copy files here for upload c) files downloaded from S3.
`folderToWatch`=Watch	Directory for user to copy files for upload to S3.
`folderQueuedForUpload`=QueuedForUpload	Directory relative to ***folderToStoreTempFiles*** for application to move files from 'folderToWatch' folder, which are picked up for upload but not uploaded yet.
`folderForUploadedFiles`=UploadedToS3	Directory relative to ***folderToStoreTempFiles*** for application to move files from 'folderQueuedForUpload' folder, which are successfully uploaded to S3.
`folderForDownloadedFiles`=DownloadedFromS3	Directory relative to ***folderToStoreTempFiles*** for application to store the files downloaded from S3.