Archive Sync different region testing

Thinking about the entire implementation:

-Entire cache file would have to individual to different users

-Config file can be used to actually differentiate each cache file

-Cache implementation has to be changed entirely, so to access and change the cache file related to the config file.

Config File:

Requires the following data to be saved:

AWS_ACCESS_KEY
AWS_SECRET_KEY
DEFAULT_REGION
AMOUNT_DATA_RETRIEVED (for this month, resets every day and in KB)
LAST_DATA_RETRIEVED_DATE (for knowing which dat data was retrieved)
TOTAL_DATA_UPLOADED (entire storage size)
ONLY_FREE (default = yes)
MAXIMUM_RETRIEVAL_TIME_ALLOWANCE (default = 30 days)
(TO BE ADDED AS NEEDED)

Sync Time Notification

-tells the last time the sync was done for each vault, when archives are listed

-Technically this is not so hard as i just have to save the inventory_date variable retrieved from job object within vault_sync_reconcile function.

-However the annoying part is that i would need to create a different cache table (sql) than the one that saves the archives.

-Another information to note is that sync date doesn't necessarily guarantee the existence of data due to the delayed nature of Glacier, but can give some idea of the time that the vault has been synced.

Data Retrieved/Data Uploaded Notification

-These data will be saved in config file and be displayed when user calls the function.

-All these data will not be exact and the user should acknowledge that this has a discrepancy

Stopping Mechanism

-DATA_UPLODADED and DATA_RETRIEVED will be saved in a config file and be used to determine whether the free price tier has exceeded or not.

-Therefore whenever archive_retrieve function is called, price will be calculated based on the information stored in config file and notify the user regarding the price and let the user decide whether the retrieval would should be done or not.

-Implementation:

-The command would be same as the archive retrieval command with extra options:
retrieve regardless of price or retrieve only if it is free
maximum retrieval time allowed

-With these options, functions will decide whether to retrieve this or to reject according to the options (Default values will be stored in config file)

-Functions to be implemented

-CheckPrice(amount of data to be retrieved, time allowed for this retrieval)
-returns the expected price of this retrieval job

-RetrieveOrNot(price,arguments received)
-decide whether to retrieve or not
-if conditions are met, call queueing mechanism for retrieval
-if conditions are not met, notify the user for rejecting retrieval

Queuing Mechanism

-Queuing mechanism will plan out which retrieval job is needed to be done at which hour.

-Then the function will make the use of ‘at’ command in linux to actually run the retrieval job requests. This means that all the retrieval job queues will be ‘saved’ in ‘at’ command with the corresponding time. AT will run each retrieval job at the designated time.

-Cache table with the future retrieval job will be saved for planning purposes.

-However, the computer has to be on for the duration of the time that the program makes the retrieval requests.

-Also it would be more favorable for each retrieval request to be done on each archive rather than a range of an archive. This would be also good for error handling as the users will notice which file has not been retrieved successfully compared to when using range retrieval method. Therefore thinking of archive as the minimal unit would be the best for everyone.

-Implementation

-The functions will decide how many archives should be retrieved every four hours.

-Then timeline will be planned based on how many archives are needed to be retrieved and time allowed.

-Functions will formulate lines of command input with ‘at’, so that retrieval would happen at designated hour.

-Error should be notified, if retrieval fails at some time

-Functions to be implemented

-PlanRetrieval(archive_size, archive_name)

return hour of the day that the retrieval should be done

-FormulateAtSyntax(archive_name,time)

-return line of the command (at) that will be executed on linux