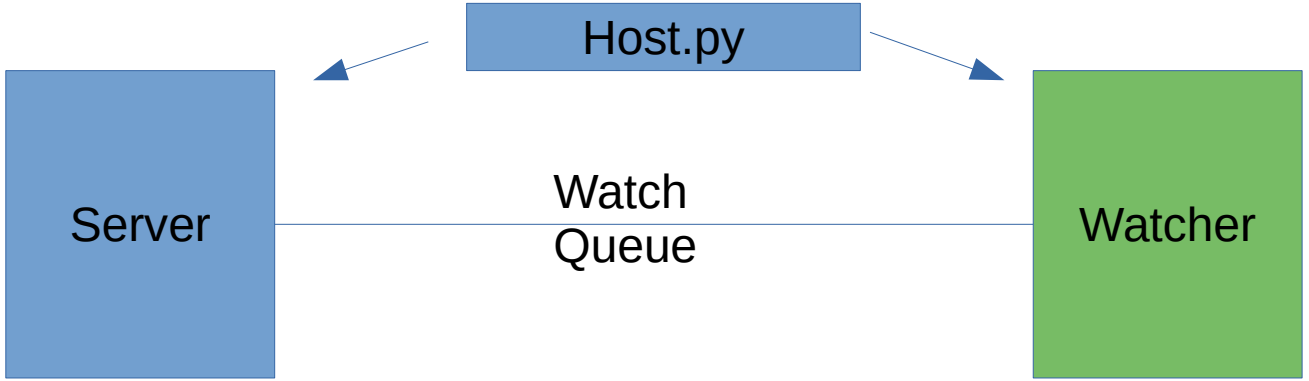


MongoDB

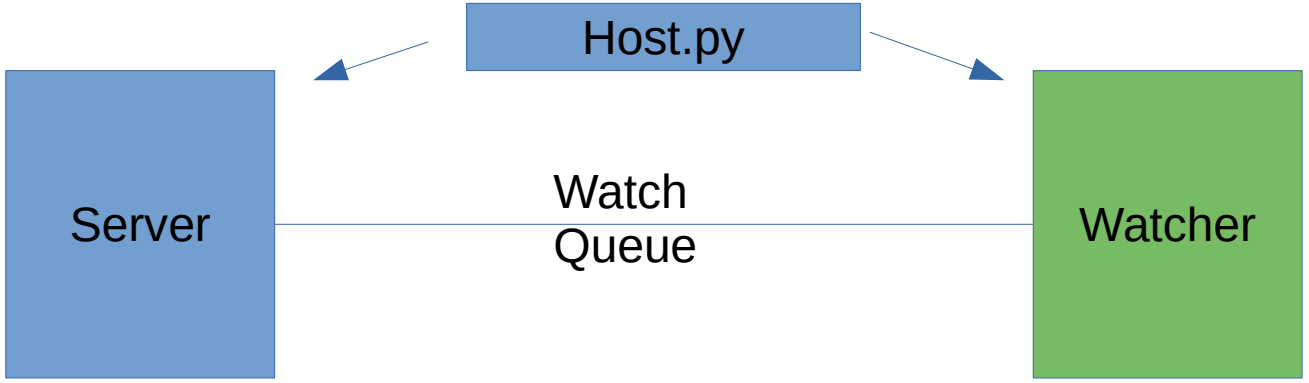
$MMEDS

- $MMEDS is the mmeds base directory (mmeds_server_data) on whatever filesystem

- Rectangles are Objects

- Objects other than MMEDSDocs are processes

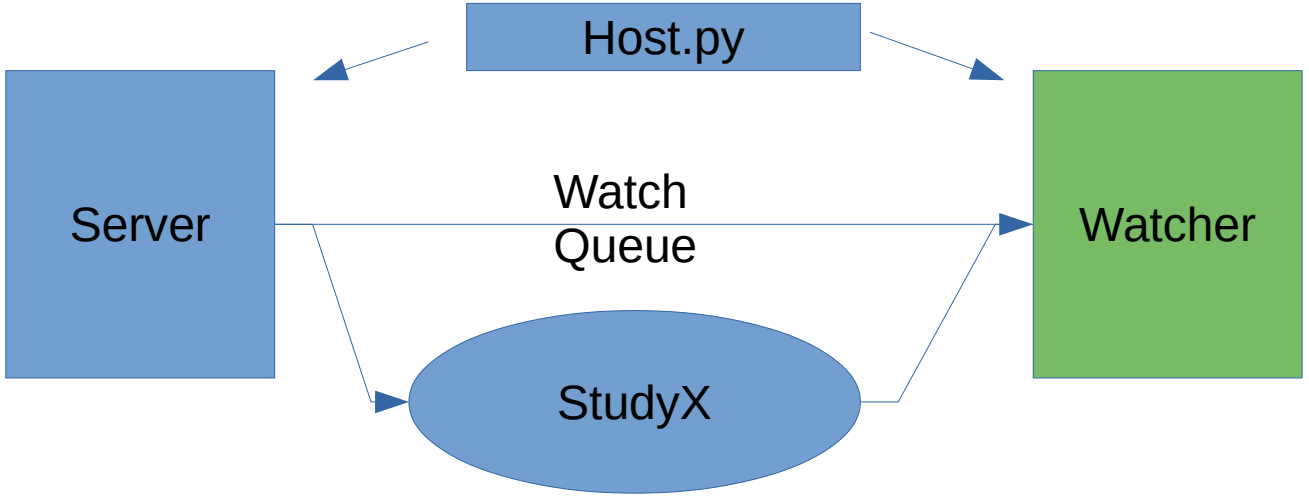
- Arrows are pointers
  i.e. The source of an arrow has knowledge of the object the arrow points to



MongoDB

$MMEDS

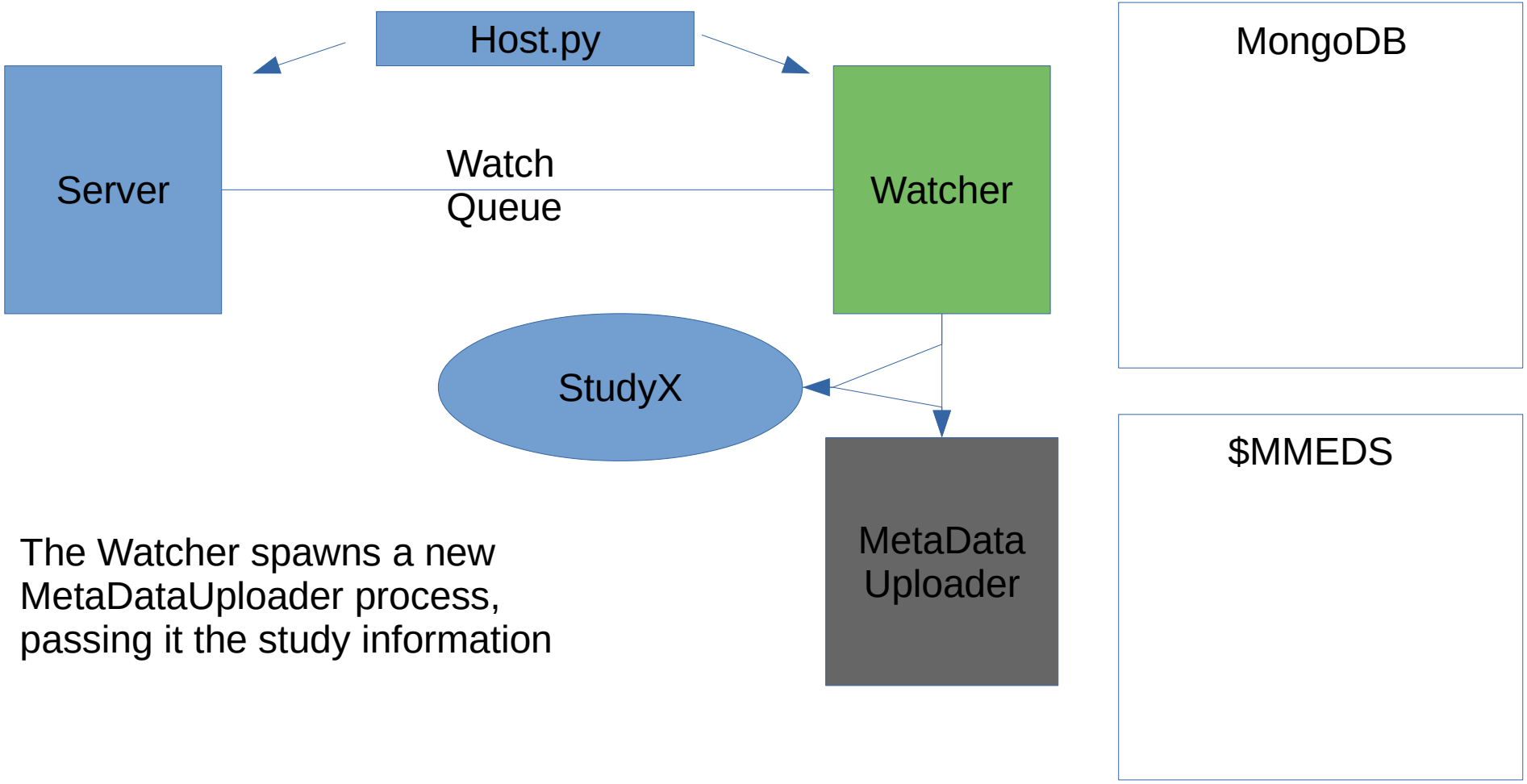
- $MMEDS is the mmeds base directory (mmeds_server_data) on whatever filesystem
- Rectangles are Objects
- Objects other than MMEDSDocs are processes
- Arrows are pointers
    i.e. The source of an arrow has knowledge of the object the arrow points to



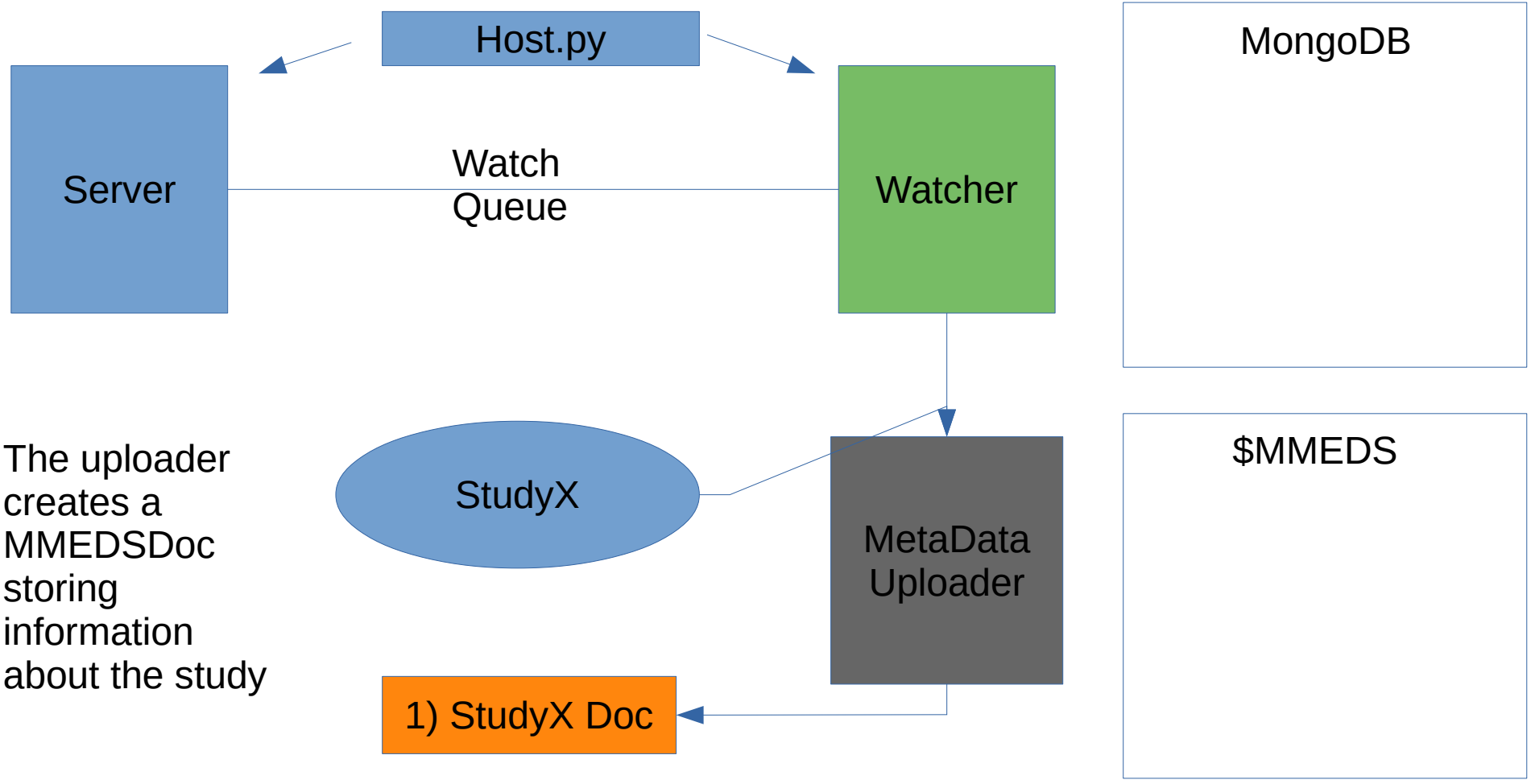
MongoDB

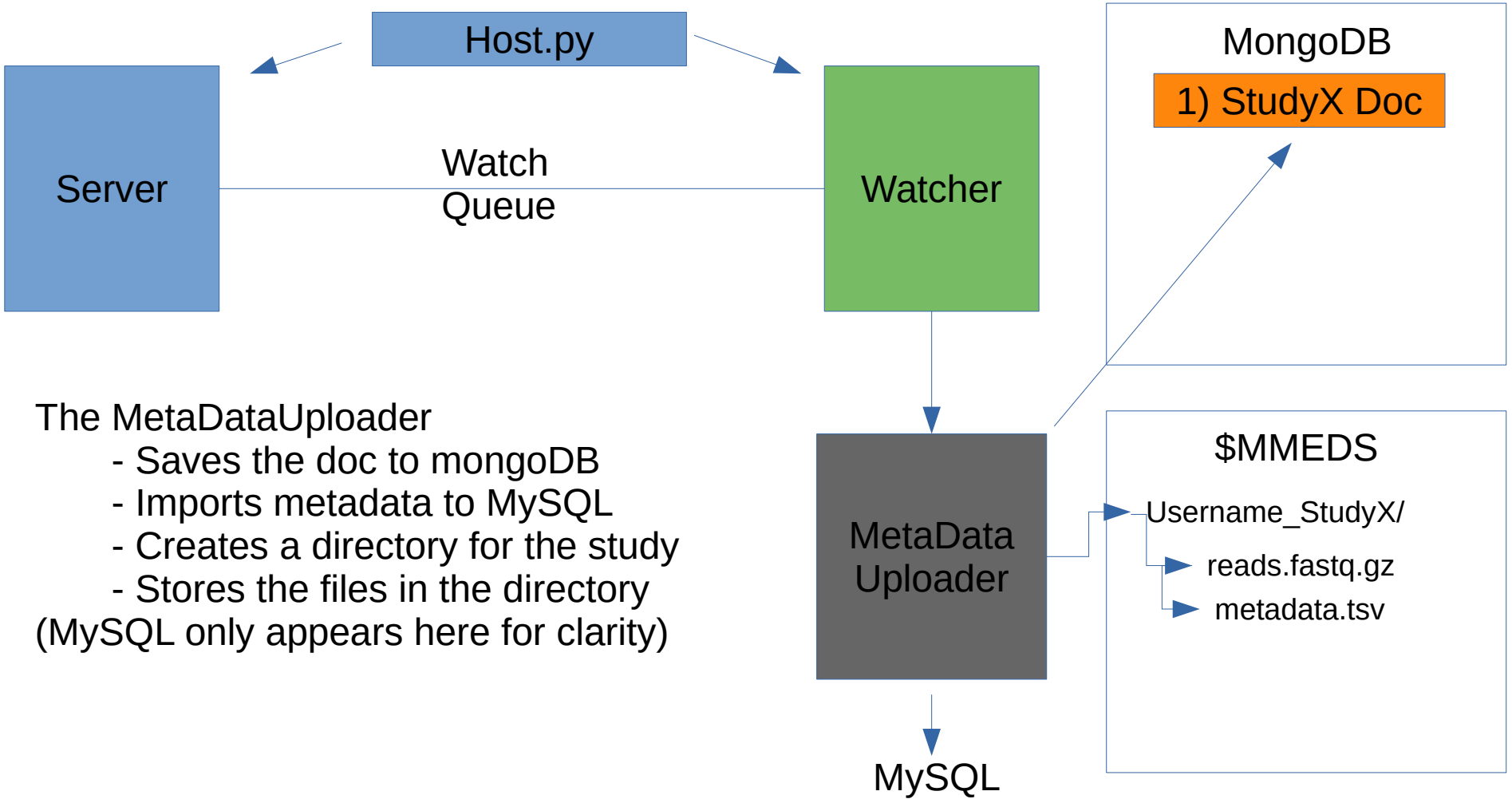
$MMEDS

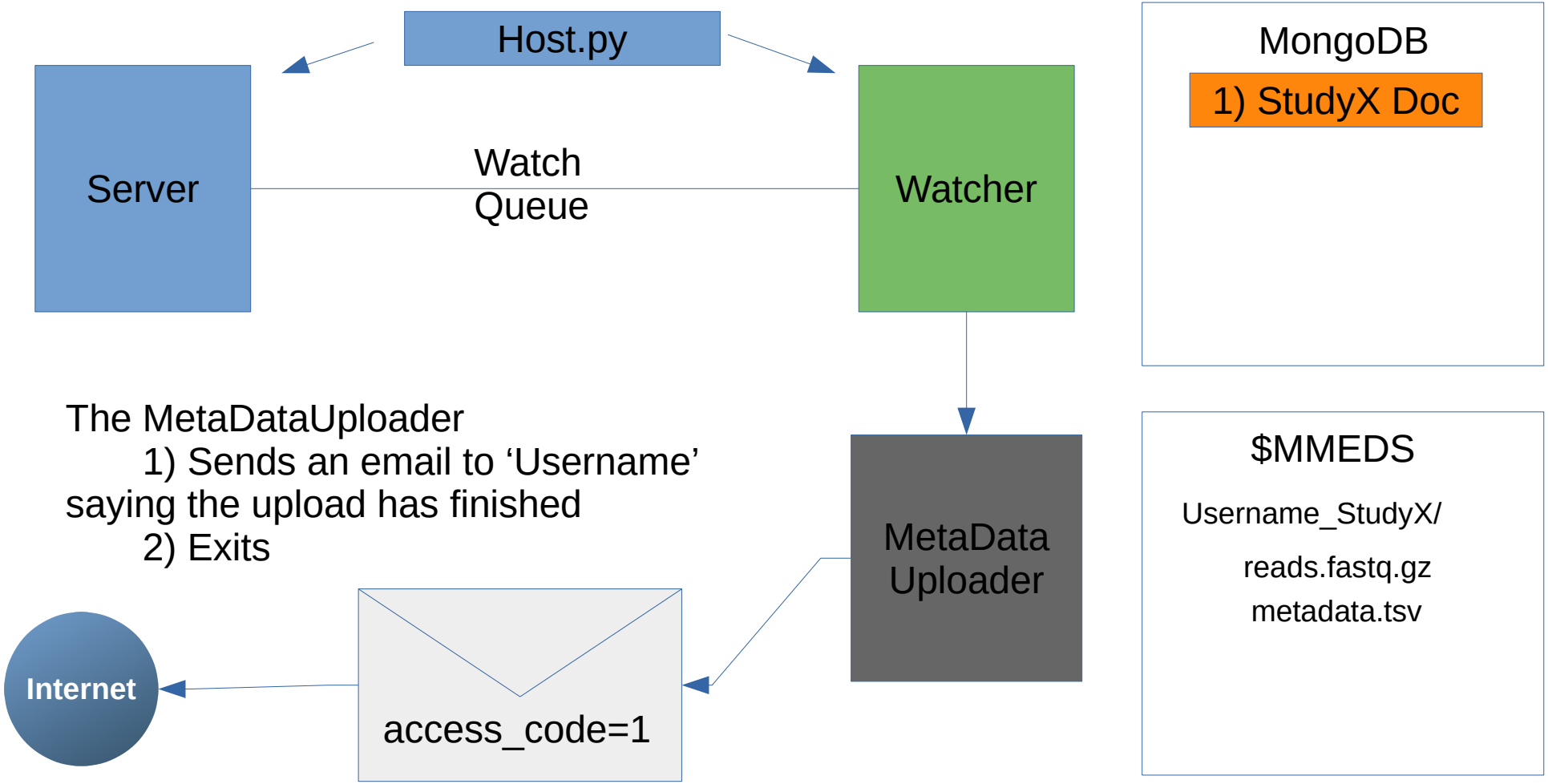
A study is uploaded to the server and passed to the Watcher via the Watch Queue

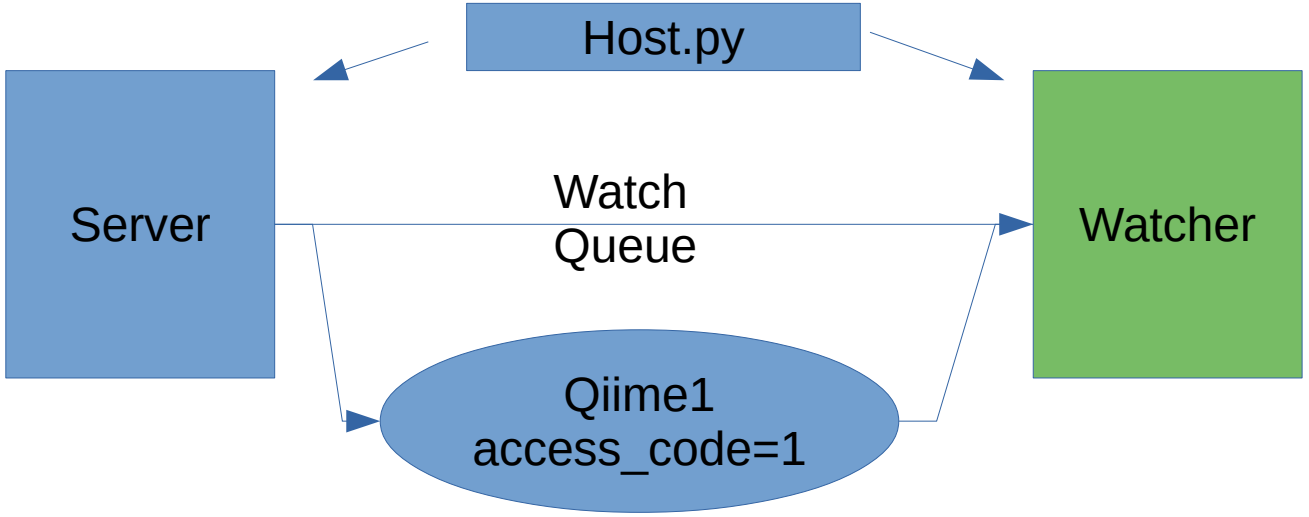


The Watcher spawns a new MetaDataUploader process, passing it the study information


Host.py
Server
Watch
Queue
Watcher
MongoDB
The uploader creates a MMEDSDoc storing information about the study
StudyX
MetaData Uploader
$MMEDS
1) StudyX Doc


Host.py
Server
Watch
Queue
Watcher
MongoDB
1) StudyX Doc
The MetaDataUploader
- Saves the doc to mongoDB
- Imports metadata to MySQL
- Creates a directory for the study
- Stores the files in the directory
(MySQL only appears here for clarity)
MetaData
Uploader
$MMEDS
Username_StudyX/
reads.fastq.gz
metadata.tsv
MySQL


Host.py
Server
Watch
Queue
Watcher
MongoDB
1) StudyX Doc
The MetaDataUploader
1) Sends an email to ‘Username’ saying the upload has finished
2) Exits
MetaData
Uploader
$MMEDS
Username_StudyX/
reads.fastq.gz
metadata.tsv
Internet
access_code=1



The user returns to the server and requests an analysis

MongoDB

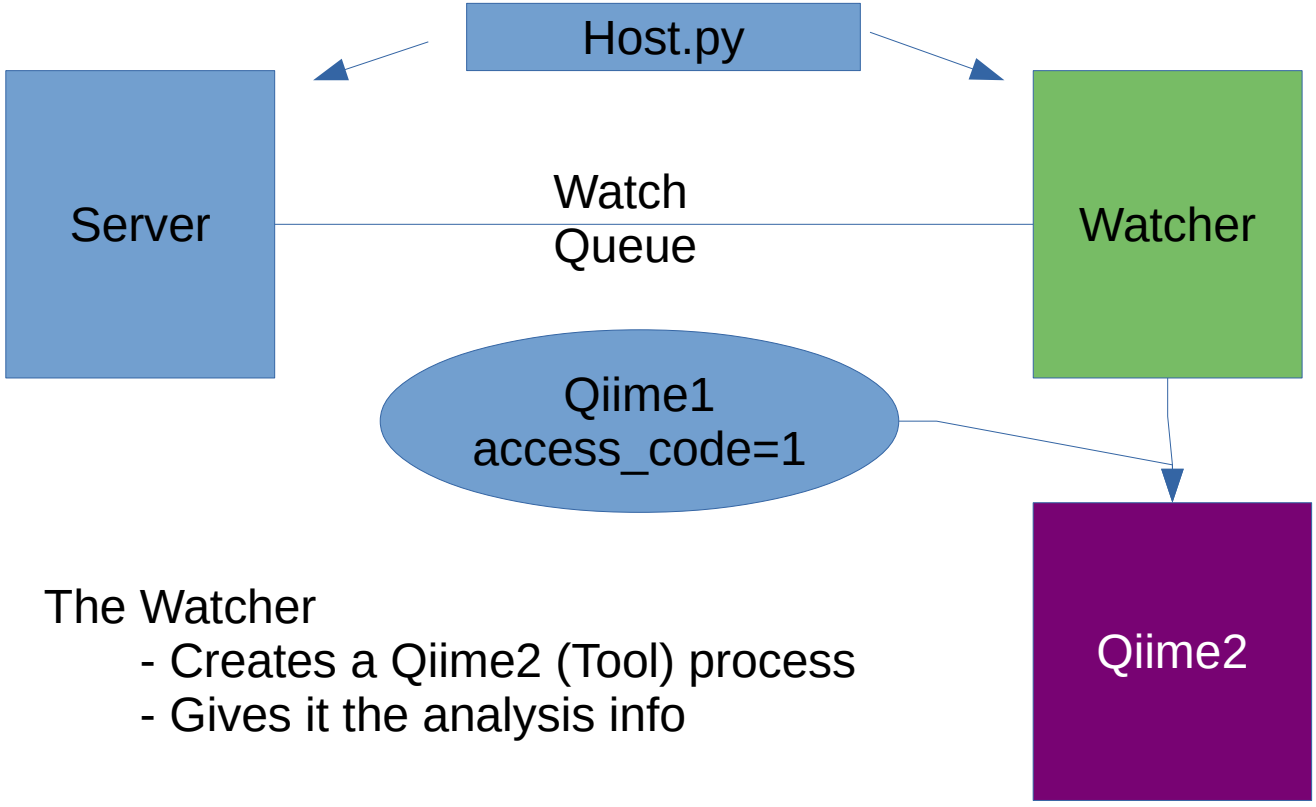
1) StudyX Doc

$MMEDS

Username_StudyX/

reads.fastq.gz

metadata.tsv



The Watcher

- Creates a Qiime2 (Tool) process
- Gives it the analysis info

MongoDB

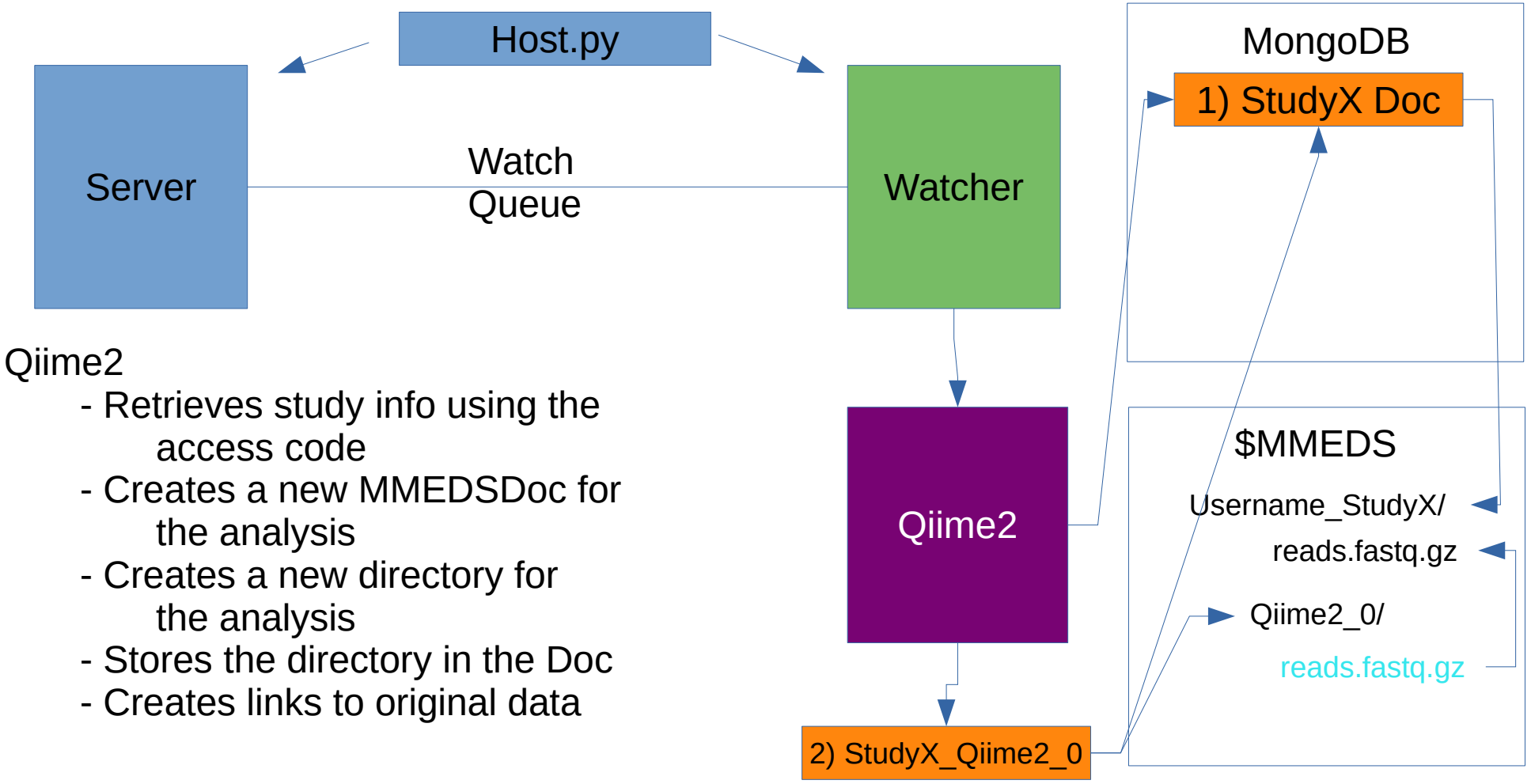
1) StudyX Doc

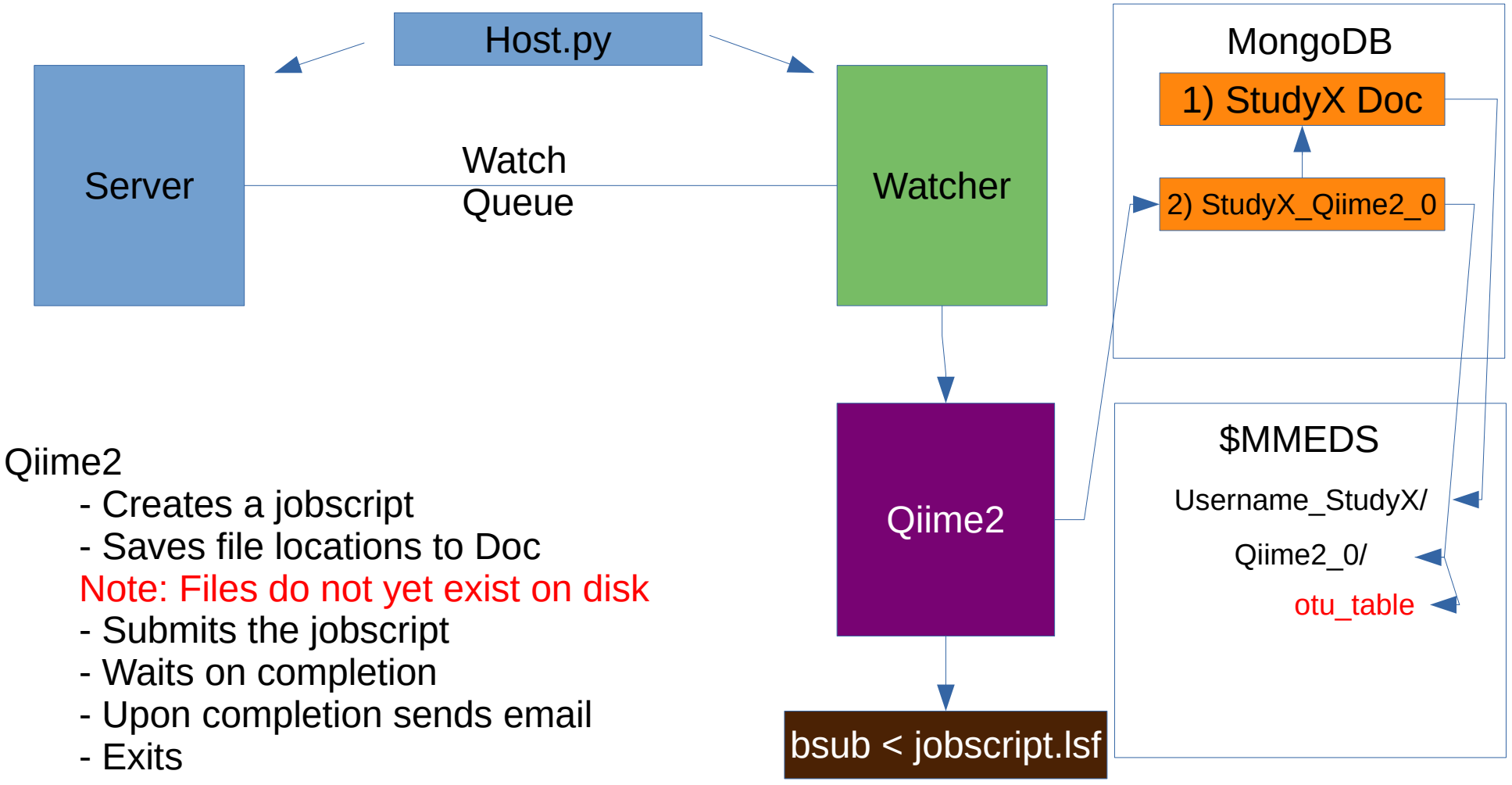
$MMEDS

Username_StudyX/

reads.fastq.gz

metadata.tsv


Host.py
Server
Watch
Queue
Watcher
MongoDB
1) StudyX Doc
Qiime2
- Retrieves study info using the
access code
- Creates a new MMEDSDoc for
the analysis
- Creates a new directory for
the analysis
- Stores the directory in the Doc
- Creates links to original data
Qiime2
$MMEDS
Username_StudyX/
reads.fastq.gz
Qiime2_0/
reads.fastq.gz
2) StudyX_Qiime2_0

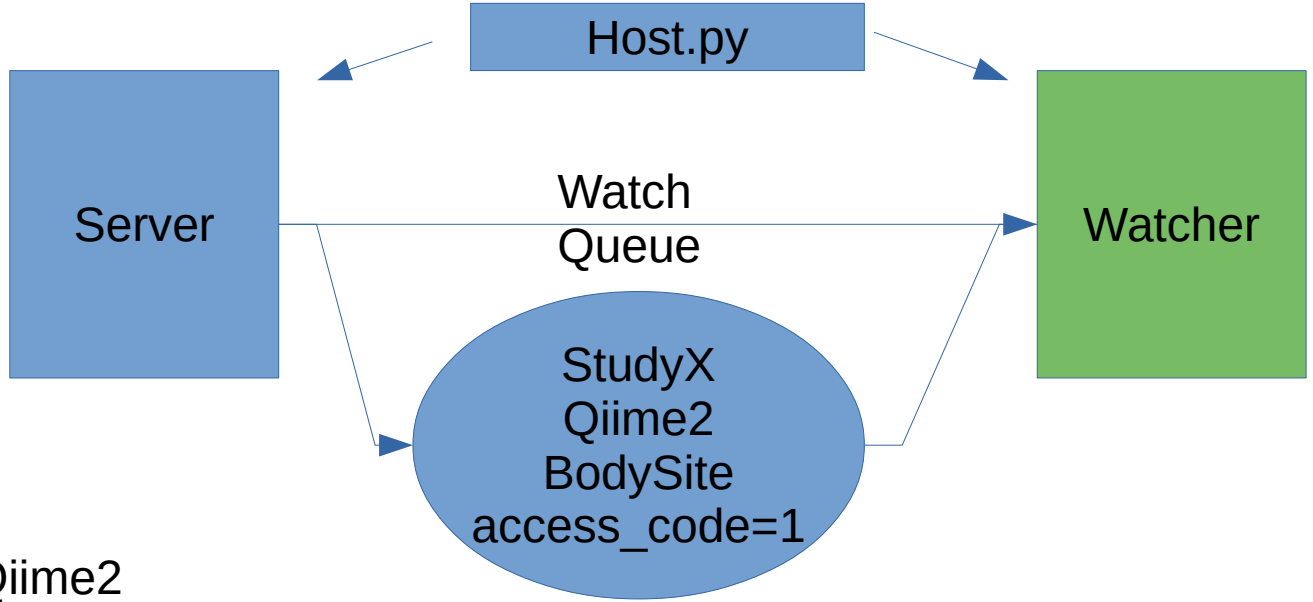


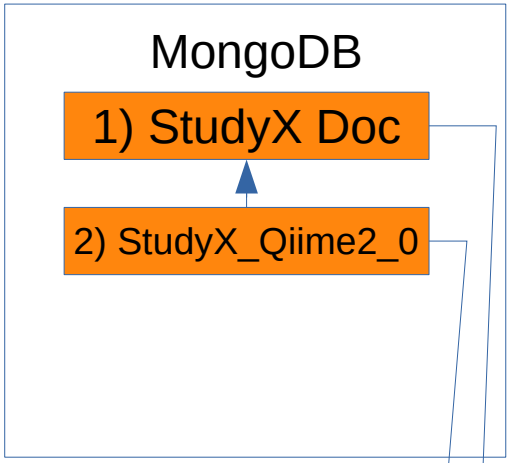
Qiime2

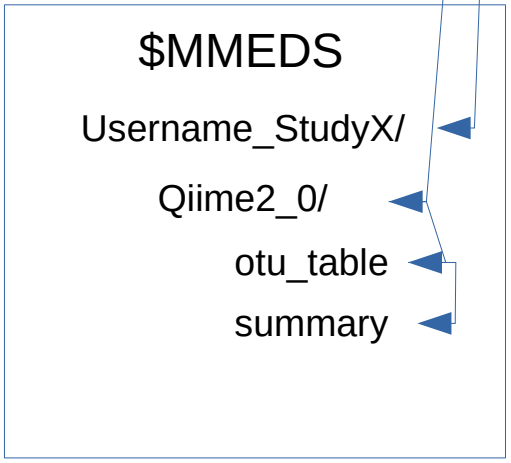
- Creates a jobscript
- Saves file locations to Doc

Note: Files do not yet exist on disk

- Submits the jobscript
- Waits on completion
- Upon completion sends email
- Exits





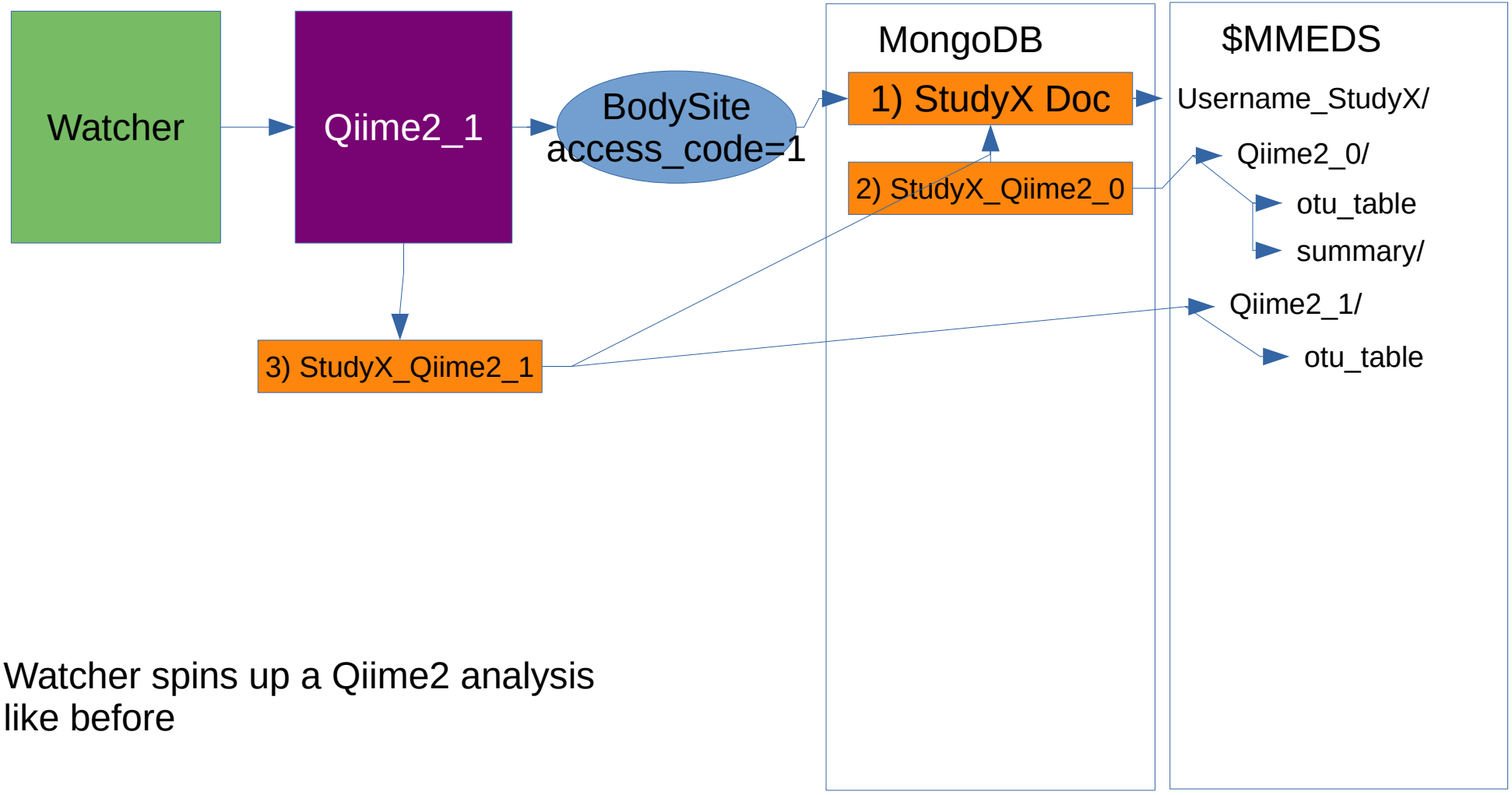


Qiime2

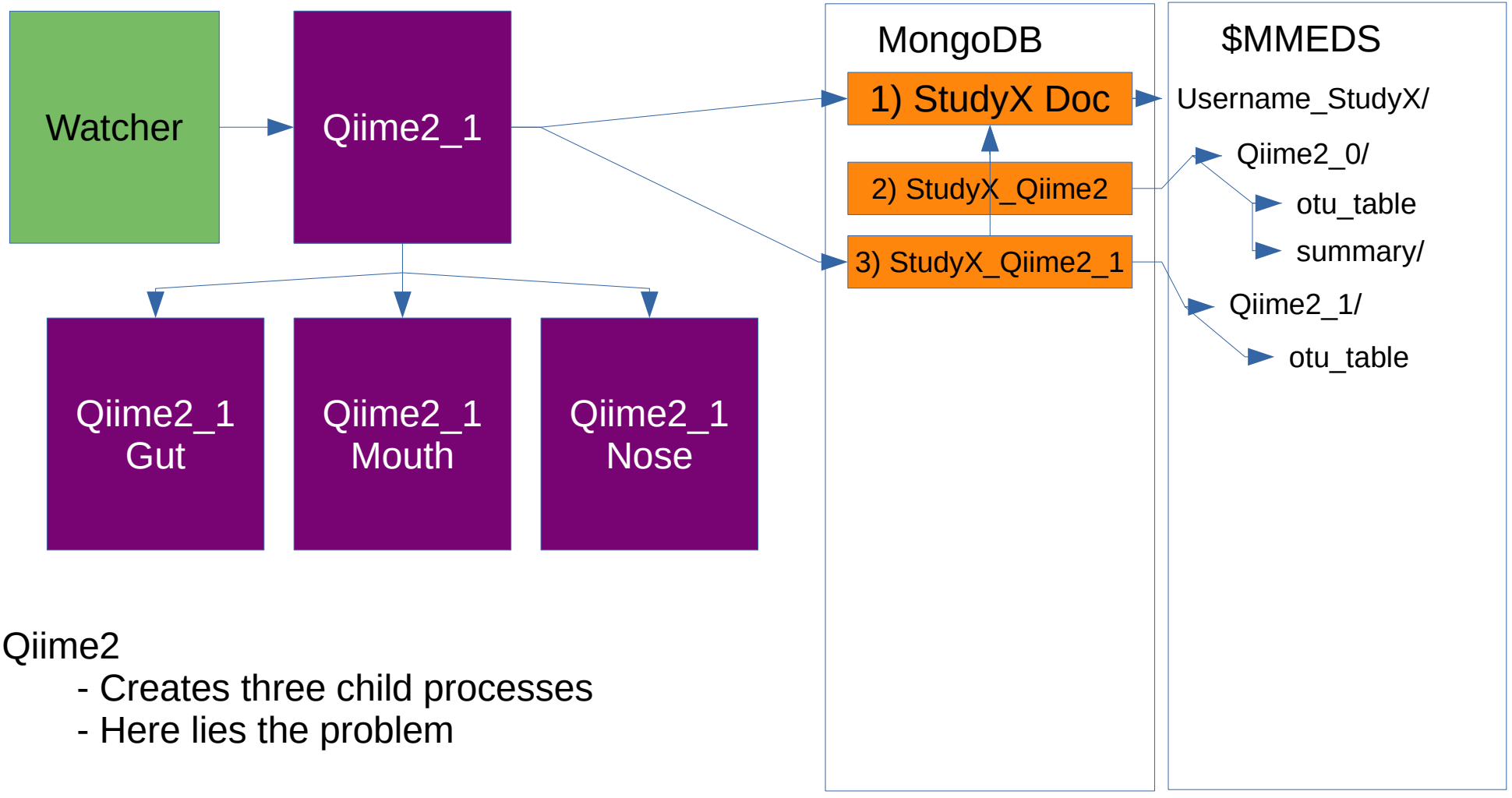
- User requests another analysis
- This time with a subset analysis

Subset Analysis Definition:

- An analysis performed on subsets of the samples available in a study. Separated by the value in some column (Body Site in this case)

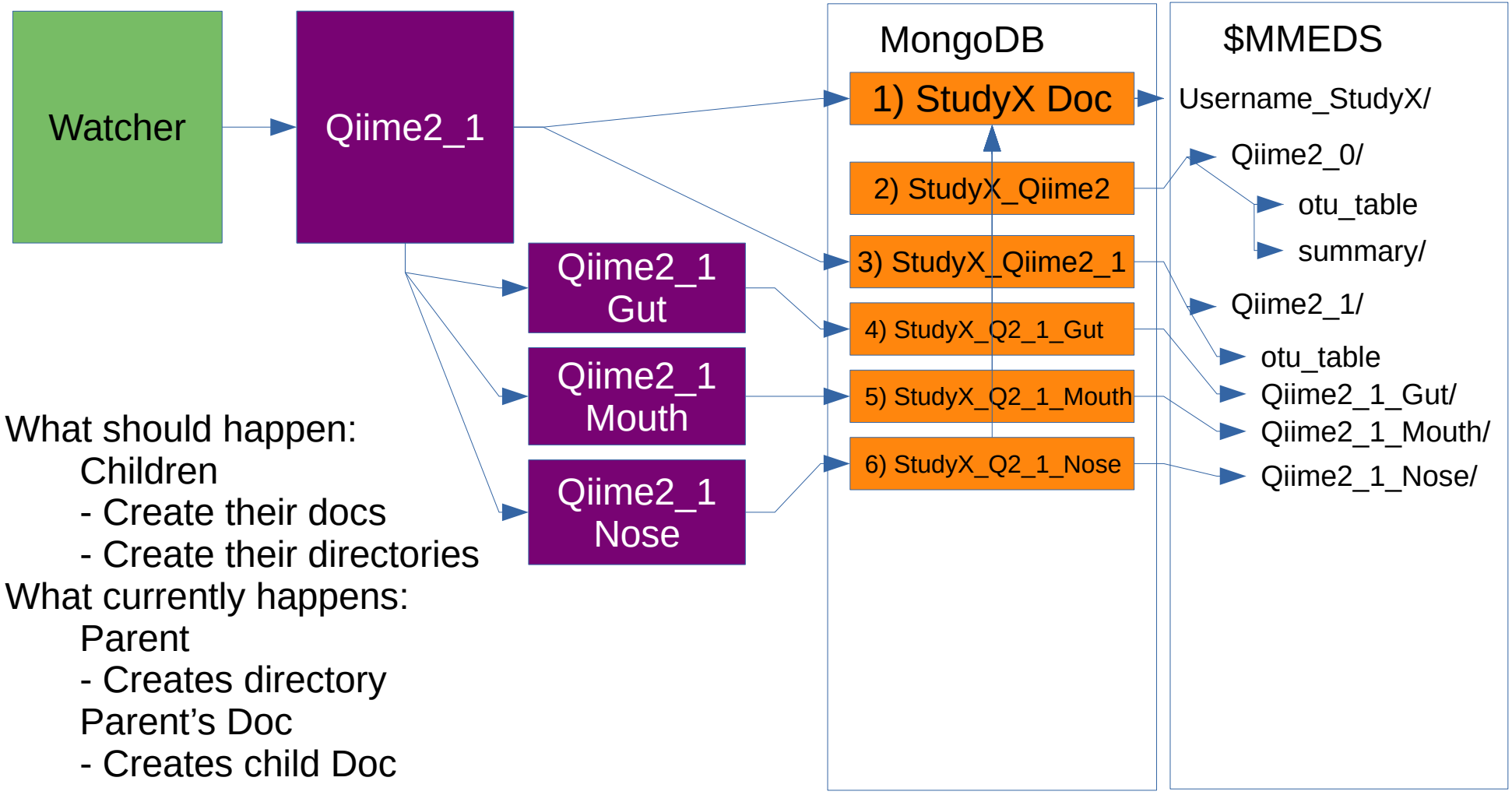


Watcher spins up a Qiime2 analysis like before



Qiime2

- Creates three child processes
- Here lies the problem



What should happen:

Children

- Create their docs
- Create their directories

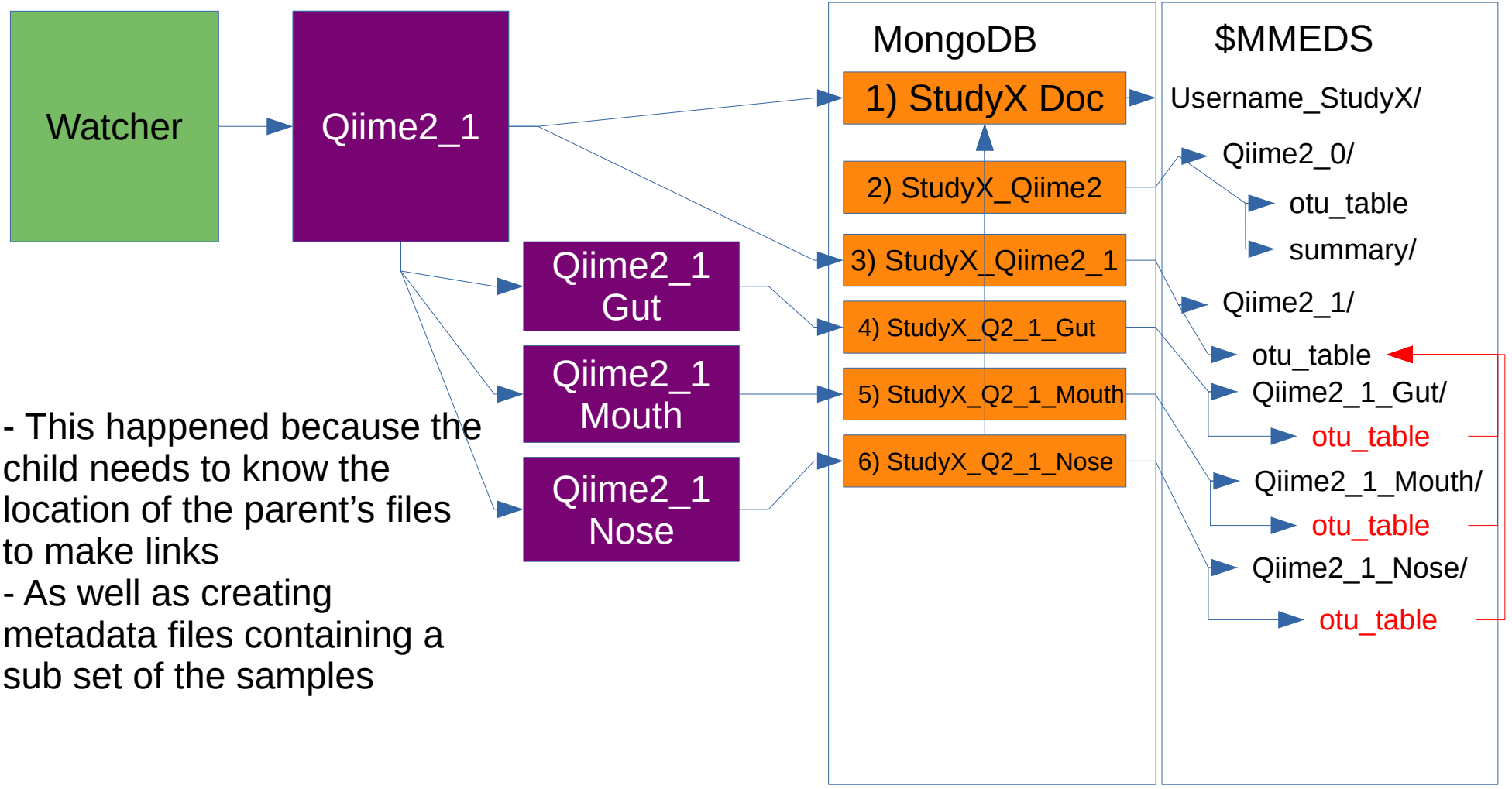
What currently happens:

Parent

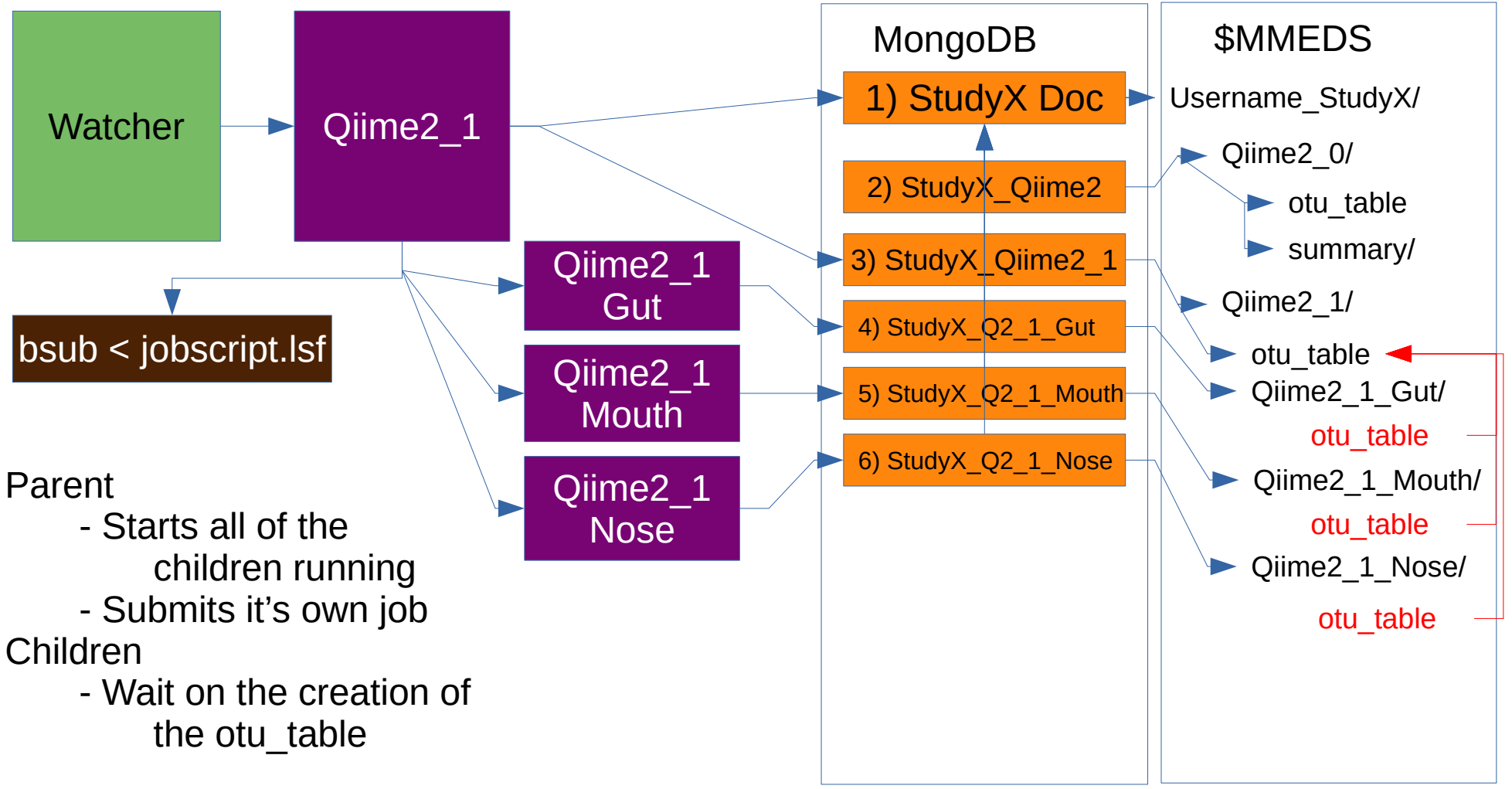
- Creates directory

Parent's Doc

- Creates child Doc

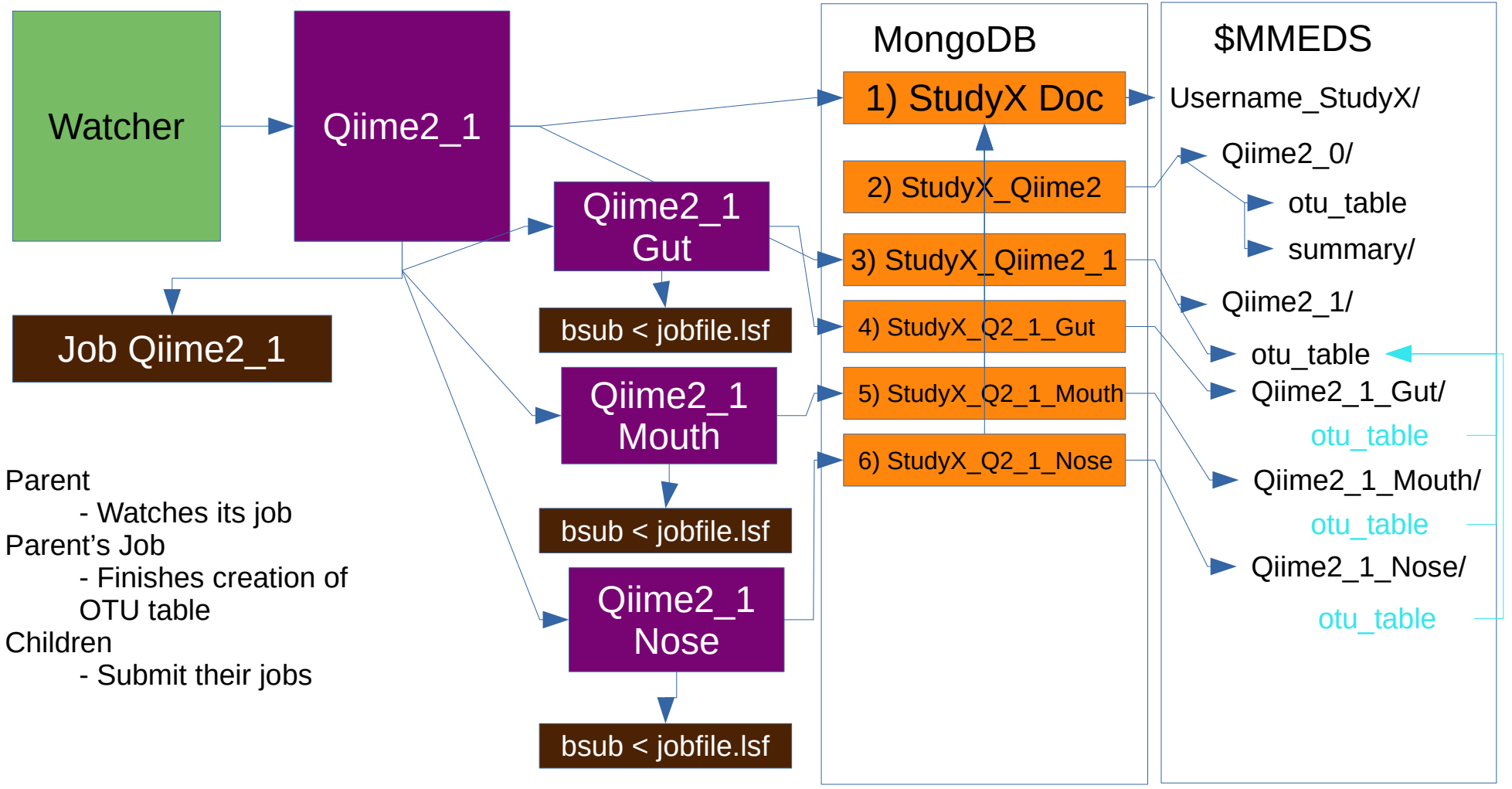


- This happened because the child needs to know the location of the parent's files to make links
- As well as creating metadata files containing a sub set of the samples



Parent
- Starts all of the children running
- Submits it's own job

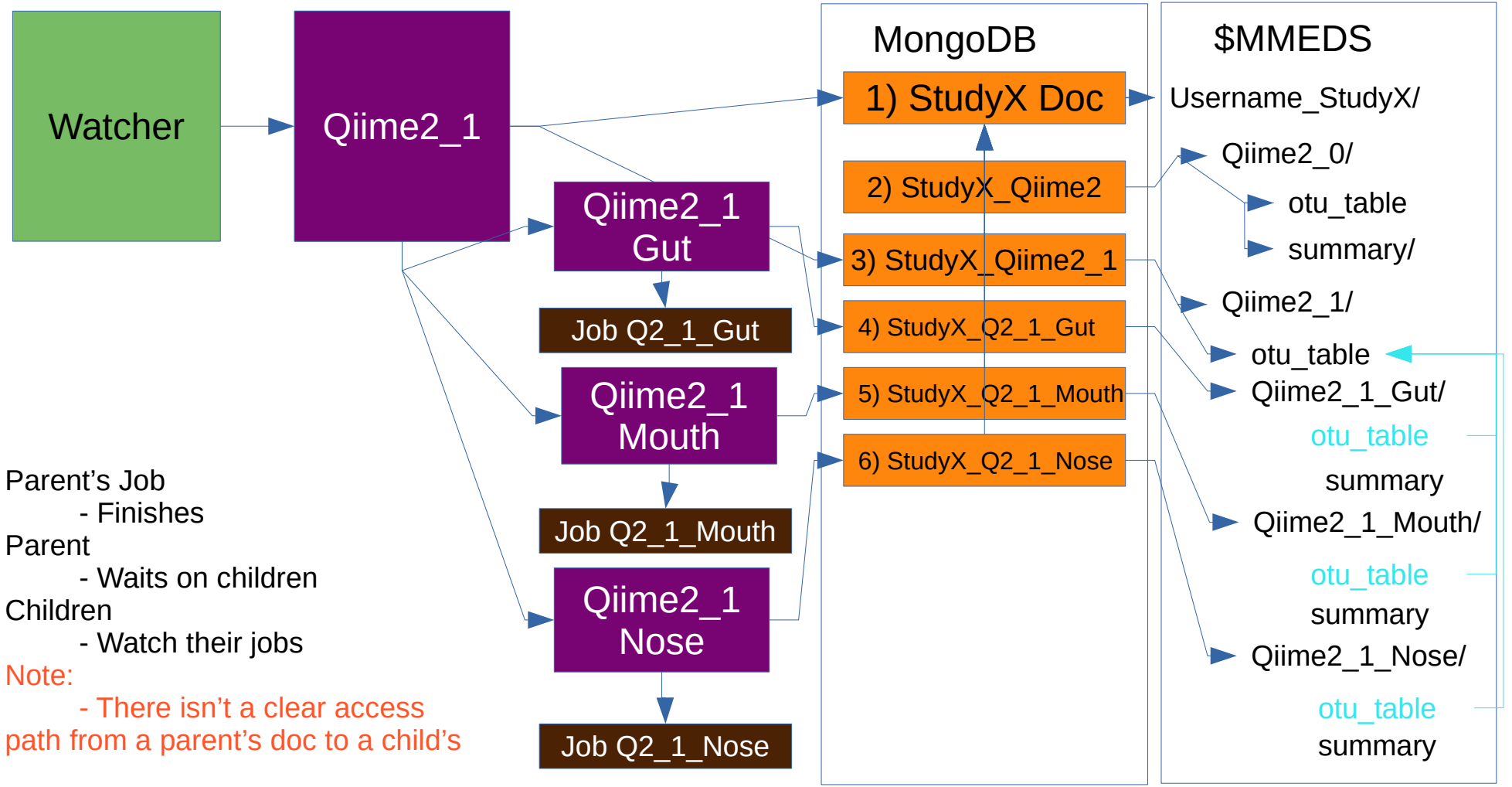
Children
- Wait on the creation of the otu_table



Parent
- Watches its job

Parent's Job
- Finishes creation of OTU table

Children
- Submit their jobs

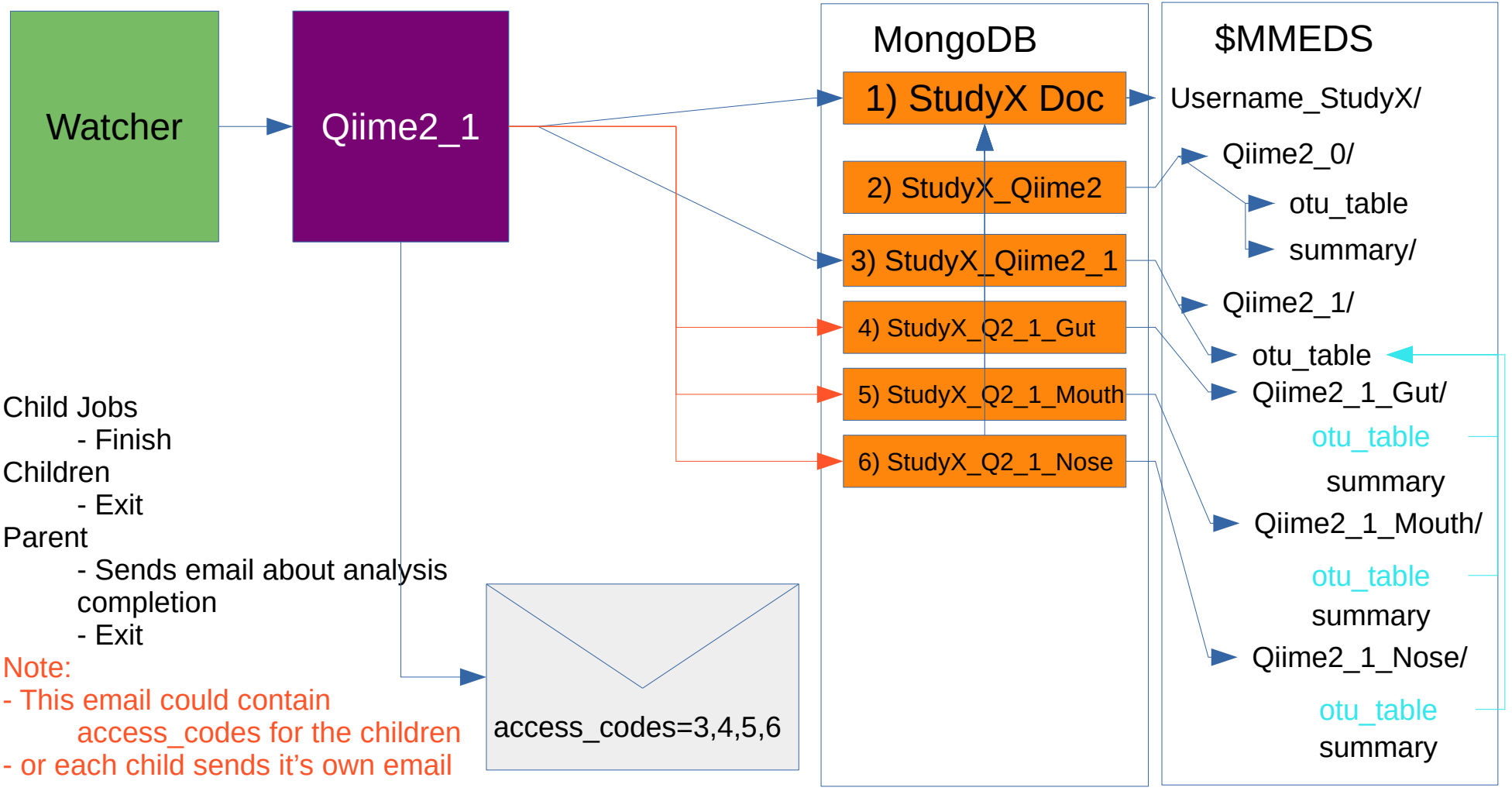


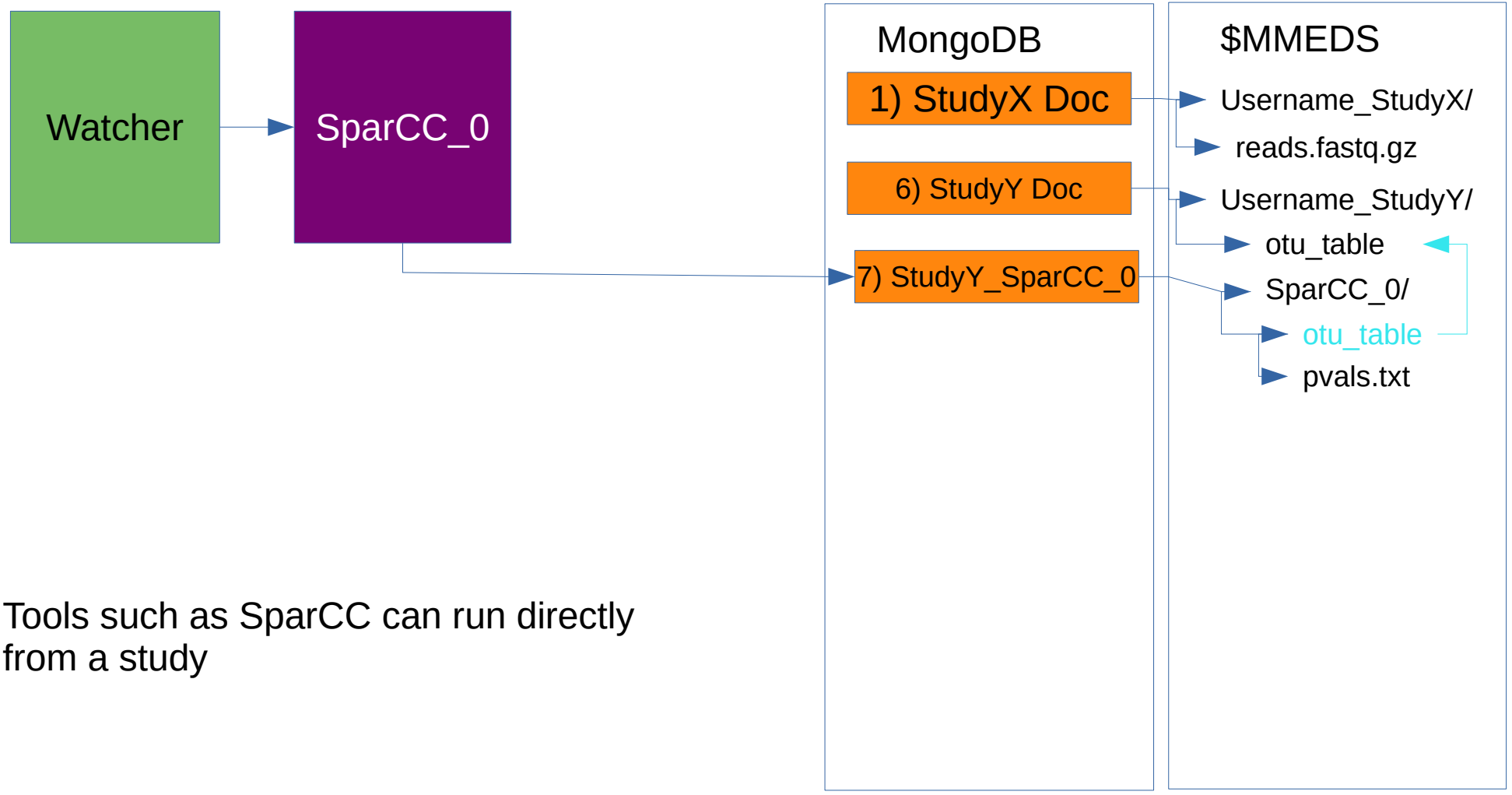
Parent's Job
- Finishes

Parent
- Waits on children

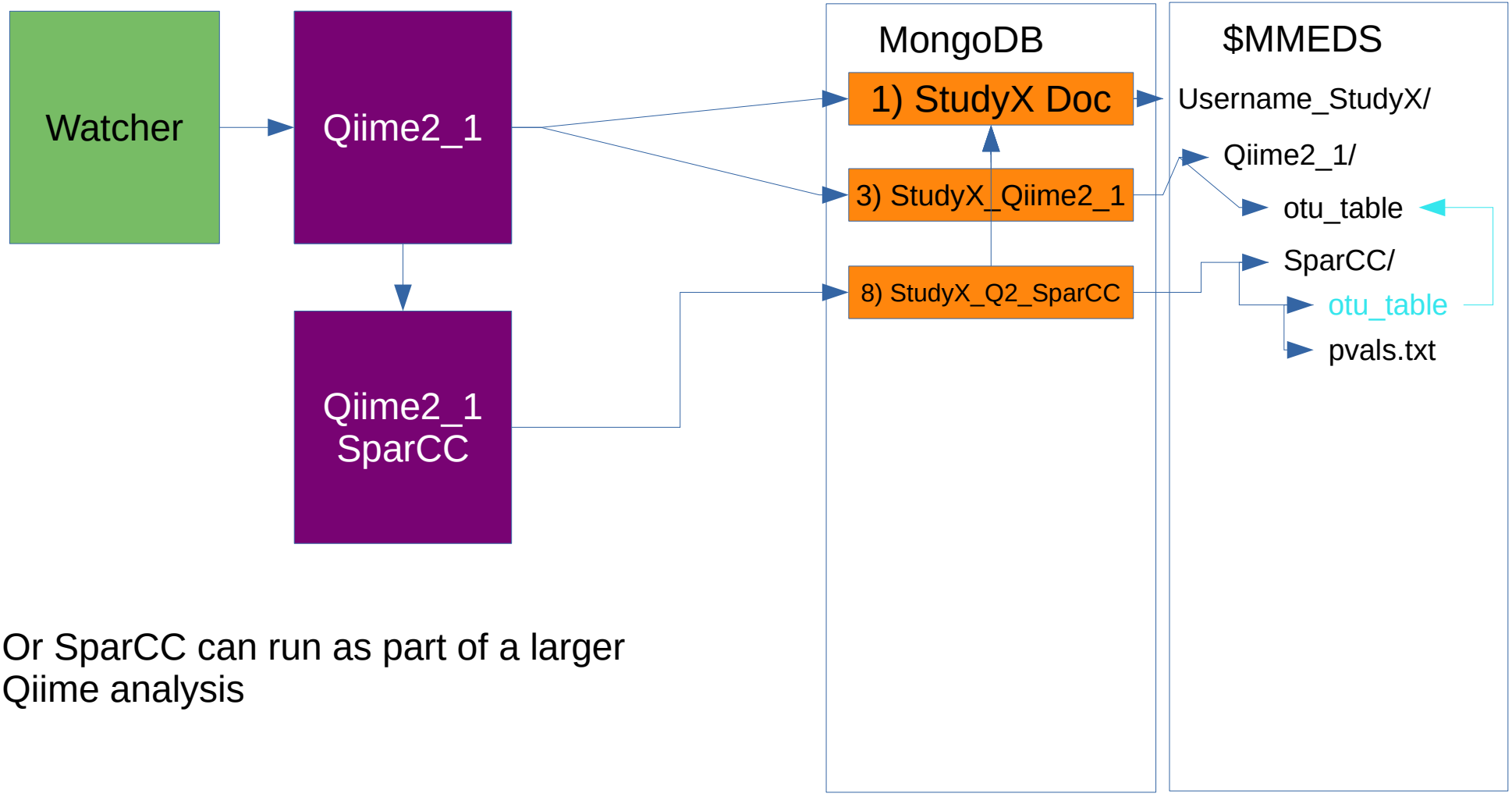
Children
- Watch their jobs

Note:
- There isn't a clear access path from a parent's doc to a child's


Watcher
Qiime2_1
MongoDB
1) StudyX Doc
2) StudyX_Qiime2
3) StudyX_Qiime2_1
4) StudyX_Q2_1_Gut
5) StudyX_Q2_1_Mouth
6) StudyX_Q2_1_Nose
$MMEDS
Username_StudyX/
Qiime2_0/
otu_table
summary/
Qiime2_1/
otu_table
Qiime2_1_Gut/
otu_table
summary
Qiime2_1_Mouth/
otu_table
summary
Qiime2_1_Nose/
otu_table
summary
Child Jobs
- Finish
Children
- Exit
Parent
- Sends email about analysis completion
- Exit
Note:
- This email could contain access_codes for the children
- or each child sends it's own email
access_codes=3,4,5,6

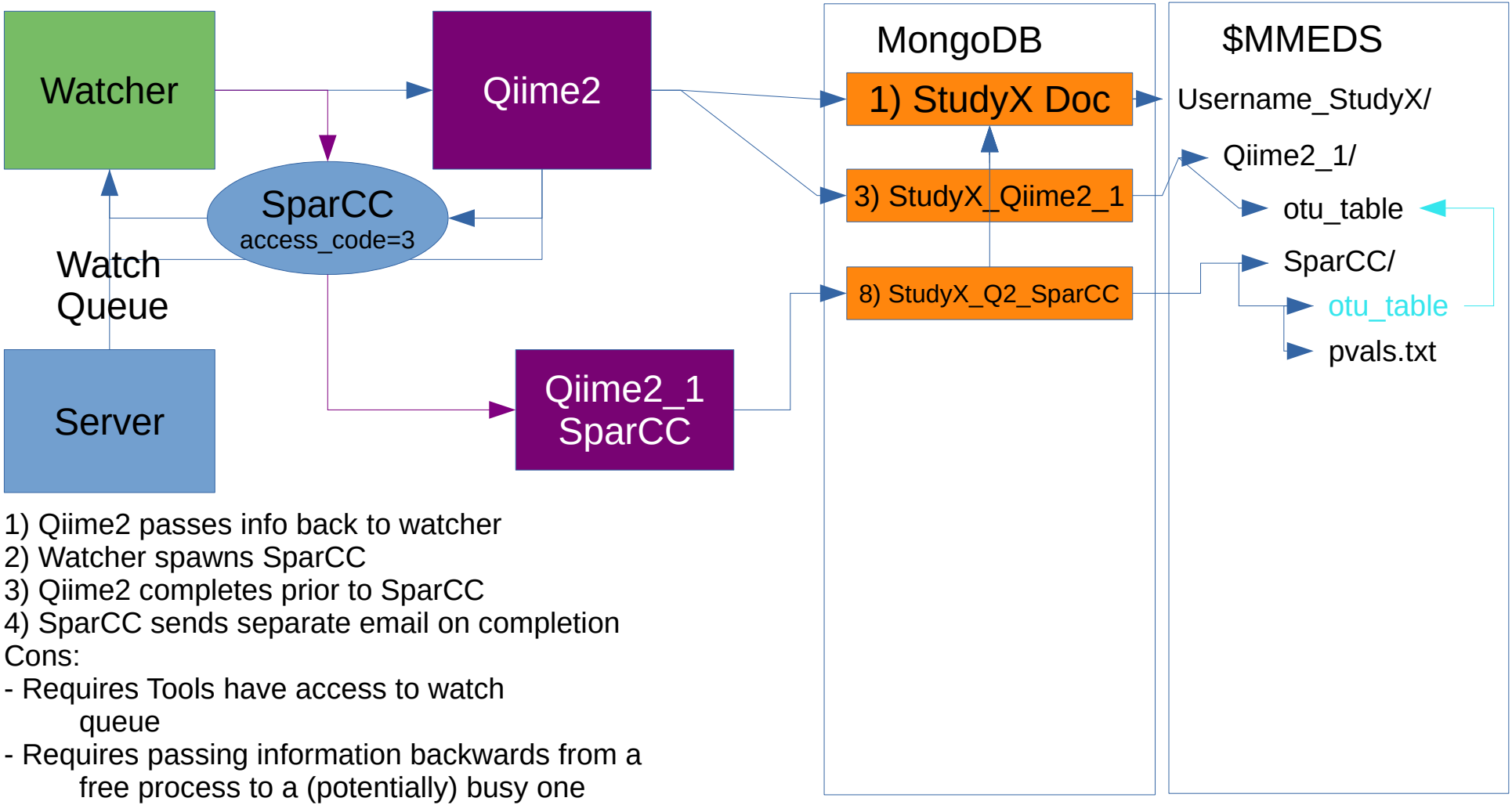


Tools such as SparCC can run directly from a study



Or SparCC can run as part of a larger Qiime analysis

Alternatives

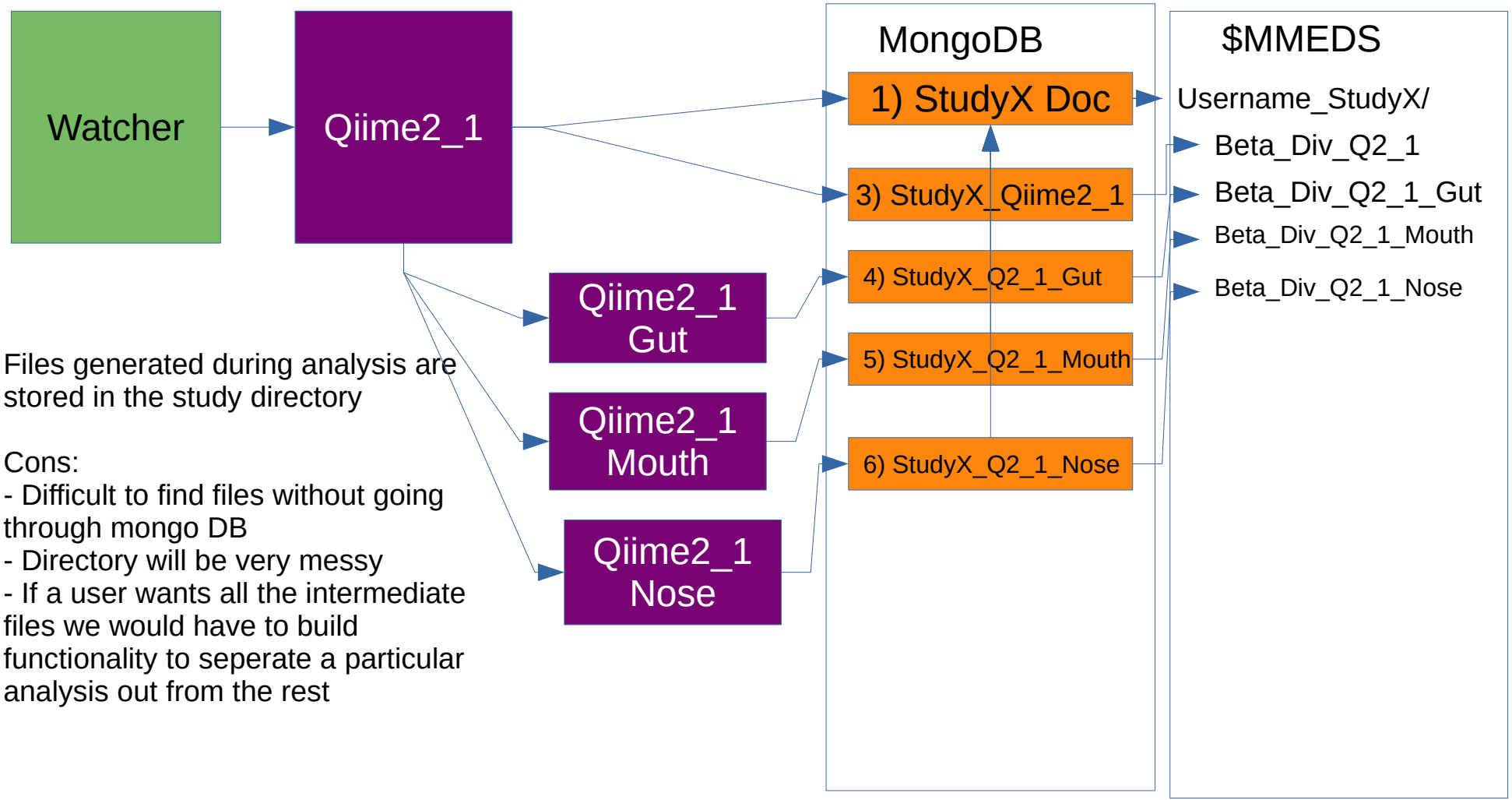


1) Qiime2 passes info back to watcher
2) Watcher spawns SparCC
3) Qiime2 completes prior to SparCC
4) SparCC sends separate email on completion

Cons:

- Requires Tools have access to watch queue
- Requires passing information backwards from a free process to a (potentially) busy one

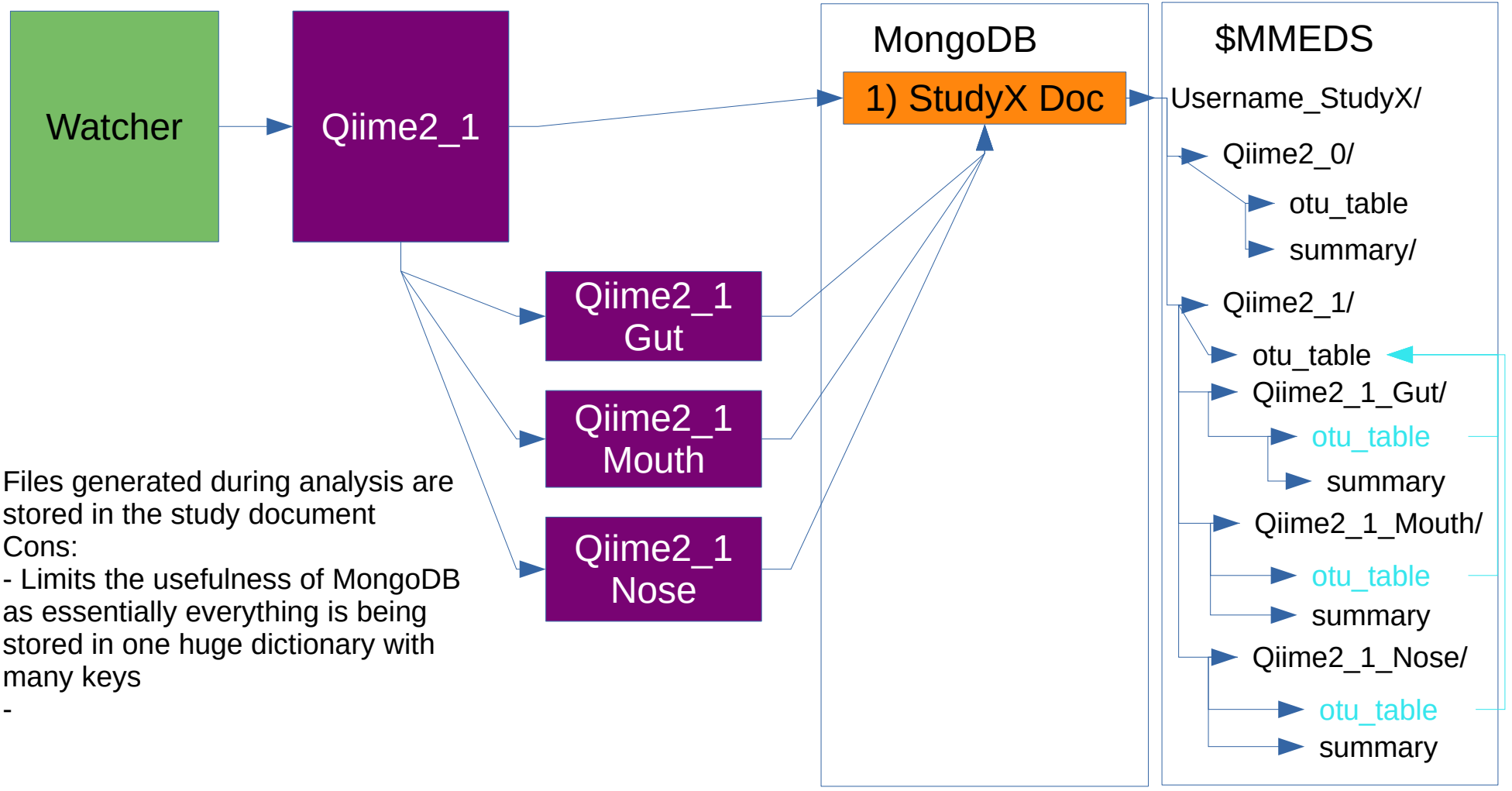
Alternatives



Files generated during analysis are stored in the study directory

Cons:
- Difficult to find files without going through mongo DB
- Directory will be very messy
- If a user wants all the intermediate files we would have to build functionality to seperate a particular analysis out from the rest

Alternatives



Files generated during analysis are stored in the study document
Cons:
- Limits the usefulness of MongoDB as essentially everything is being stored in one huge dictionary with many keys
-