Contents

[Objective 2](#_Toc104896105)

[Description 2](#_Toc104896106)

[Operating System 2](#_Toc104896107)

[Environment 3](#_Toc104896108)

[Files and Folders 4](#_Toc104896109)

[LOGS 6](#_Toc104896110)

[Shell Script 7](#_Toc104896111)

[Cron Job 8](#_Toc104896112)

[Monitoring script changes 10](#_Toc104896113)

[Onboarding Changes 19](#_Toc104896114)

# Objective

The document's goal is to provide a more comprehensive understanding of the monitoring script, onboarding adjustments, and all other pertinent information regarding this service.

# Description

A monitoring script is a Python and Shell script-based application. The goal of this application is to allow users to upload files or folders to specific locations, and then have the perform checks on those files or folders, such as verifying correct naming conventions, file size, and, most importantly, whether the user has uploaded the files in the correct location. After all these inspections, there are two conditions that this application must meet, namely,

* + If all the tests pass and all the conditions are met, the file will be transferred to a landing directory, and an email will be sent to that user's and circulants' support teams which contains the message of successfully transferred.
  + If all the tests fail and all the conditions are not met, the file will be deleted, and an email will be sent to the user's and circulants' support teams informing them that the file they uploaded contains one or more of the three conditions: incorrect naming convention, incorrect file size, or incorrect file location.

# Operating System

The Linux operating system is used to run the monitoring script application. We utilized AWS EC2 to create a virtual machine, running Redhat 8.0 within a VPC. To access the Virtual Machine, you'll need an AWS workspace and the appropriate credentials to access the workspace. After that, you'll need a PEM file, a username, and the server's IP address to log in to that server.

Beren Production instance :

Graphical user interface, text, application, email

Description automatically generated

Operating system details:

A screenshot of a computer

Description automatically generated with medium confidence

# Environment

Virtual environment: An isolated python environment in which the Python interpreter, libraries, and scripts are installed. They are isolated from those installed in other virtual environments, and the one that comes with the operating system.

* Installing Virtual Environment
  + $ pip install virtualenv
  + Test your installation: [virtualenv –version]
* Using Virtual Environment
  + You can create a virtualenv using the following command:
    - $ virtualenv monitor-venv
  + After running this command, a directory named monitor-venv will be created. This is the directory which contains all the necessary executables to use the packages that a Python project would need. This is where Python packages will be installed. Now after creating virtual environment, you need to activate it. Remember to activate the relevant virtual environment every time you work on the project. This can be done using the following command:
    - $ source /home/ec2-user/monitor-venv/bin/activate
    - Below is the path of Virtual environment in production server:

***“cd /home/ec2-user/monitor-env”***

Text

Description automatically generated

# Files and Folders

In this section, we mention the locations of the files and folders that our monitoring script requires. There are many important parts of the monitoring script as follows:

* Beren-monitor
  + - Mailer
    - Common
    - Main python file
* Monitor-env
* Script
* Cron job
* Logs

**Beren-monitor:**

This is the main directory which contains all the useful python file to run monitoring program. And it contains subfolders to execute some of the other functions.

Path to project directory: “cd /home/ec2-user/beren-monitor”



1. Mailer:

This mailer function contains email configuration file for vendors. It also contains a function which is written in python to send an email to vendors and circulants team members.

Path to mailer directory: “cd /home/ec2-user/beren-monitor”



Contents of mailer directory:

Text

Description automatically generated

1. Common:

It contains all the constants which has been defined in constant.py and it has been used in our main python file.

Path to common directory:

“cd /home/ec2-user/beren/monitor/common”



1. Main python file:

It is the main python file to run all the modules of monitoring script.

Edit main python file:***’vi /home/ec2-user/beren-monitor/beren-cir-monitor.py’***



**Monitor-env:**

This contains all the necessary files to activate the virtual python environment for our monitoring script.

Virtual environment path:

***“cd /home/ec2-user/minitor-venv”***



**Shell Script:**

It is used to run the respective cron job for our monitoring script.

Shell script path:

***“vi /home/ec2-user/beren-monitor-scr.sh”***



**Cron job:**

A Linux scheduler which runs daily on a particular time. We can use a shortcut command to edit the cron file. But there is a one thing which we must keep in mind and that is “it is user independent”. So, while you were creating cron job and you were logged in as a root user and then if you want to edit the cron job again we must login with the same user as we were while creating that job.

Command to edit: ***“crontab -e”***

Manually editable file:

***“cd /var/spool/cron”***



File to edit cron job:

***‘vi /var/spool/cron/root’***

***‘vi /var/spool/cron/ec2-user’***

Text

Description automatically generated

**Logs:**

This file contains logs for monitoring script and cron job.

Monitoring script Log file:

***‘vi /home/ec2-user/logs/file\_placement\_logs.log’***



Cron job log file:

***‘vi /home/ec2-user/logs/batch\_cron.log’***



Log movement to Redshift Log file:

***‘vi /home/ec2-user/logs/logg.py’***



# LOGS

**File\_placement\_logs.log:**

This function is written in the main configuration python file to store the logs of file placement in user zone.

Log function in code:

***“import logging***

***logging.basicConfig(filename=common\_const.logger\_path,level=logging.INFO, format='%(asctime)s :: %(levelname)s :: %(message)s')”***

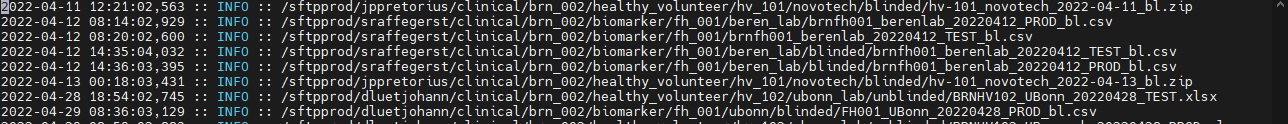


This function is written in the common folder in constant.py, it has assigned with the path of the log file which is being used in above function.

Path defined in code:

***‘Logger\_path=’/home/ec2-user/logs/file\_placement\_logs.log’***



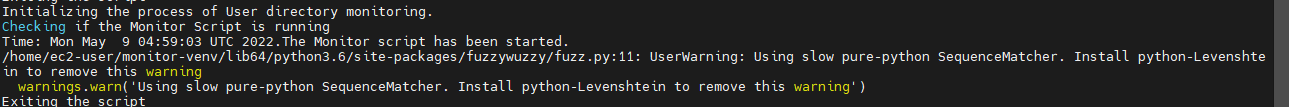
Example: 

**Batch\_cron.log:**

This function is written with the cron job to store the details about the cron job execution.



Example:



# Shell Script

Shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text. Shell Script contains file path location of monitoring script, python virtual environment and command to execute the program automatically through cron job.

Path:

***‘cd /home/ec2-user/beren-monitor-scr.sh’***

Text

Description automatically generated

To edit:

***‘vi /home/ec2-user/beren-monitor-scr.sh’***



To change location or command in script:Text

Description automatically generatedText

Description automatically generated

# Cron Job

The cron command-line utility, also known as cron job, is a job scheduler on unix-like operating systems. Users who set up and maintain software environments use cron to schedule jobs to run periodically at fixed times, dates, or intervals.

Path to cron job:

***“cd /var/spool/cron”***

Text

Description automatically generated

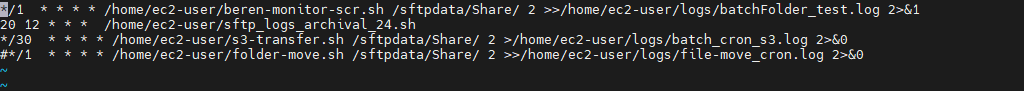
But there is one thing we must remember: "it is user independent." So, if  signed in as a root user when you created the cron job, and if wish to edit it again,  must login with the same identity  used while creation.

To edit:

***“crontab -e”***



Output of command:



# Monitoring script changes

In this section, we’ll talk about the modifications that need to be considered when a new DTA or Mailer function is allocated. In this monitoring script we are monitoring the two types of data which is:

1. Clinical
2. Bioinformatics

We will be discussing the changes which need to be considered in these two data.

**Clinical:**

File Path:

Text

Description automatically generated

Main file:

Text

Description automatically generated

To edit:



Chronology:

The first function which will be going to execute is the main function.

Text

Description automatically generated

This will provide the location of file and then it will call the function with the path:



And then this function will be called, and this function will call another function where we have written all the important program for clinical data.

Text

Description automatically generated

Text

Description automatically generated

Below is the screenshot of clinical function which is going to find out that if the data is clinical, it must check all the assigned DTA.

Text

Description automatically generated

Here we need to make changes if new DTA has been assigned for clinical data:

***Code:***

***def check\_file\_name(filename):***

***try:***

***#Hv001\_genwiz\_ataseq\_abc001\_R1.fastq***

***file\_name\_temp = filename.split("/")***

***file\_name=file\_name\_temp[-1]***

***file\_aa=file\_name.split(".")***

***file\_name\_final=file\_aa[0]***

***file\_name\_final=file\_name\_final.lower()***

***file\_type=file\_aa[1]***

***file\_type\_configs=["zip","xlsx","csv","xpt"]***

***for i in range(len(file\_name\_temp)):***

***if file\_name\_temp[i] == 'unblinded':***

***file\_name\_temp[i] = 'ub'***

***elif file\_name\_temp[i] == 'blinded':***

***file\_name\_temp[i]='bl'***

***else:***

***pass***

***#print(file\_name\_temp)***

***#print(file\_type)***

***if "\_" in file\_name\_final:***

***file\_name = file\_name\_final.split("\_")***

***elif "-" in file\_name\_final:***

***file\_name = file\_name\_final.split("-")***

***else:***

***file\_name=file\_name\_final***

***#print(file\_name)***

***file\_name\_string = file\_name\_temp[4]+file\_name\_temp[5]+file\_name\_temp[6]***

***for line in file\_name:***

***if re.search(r'\d{4}-\d{2}-\d{2}',line):***

***file\_name.remove(line)***

***for line in file\_name:***

***if file\_type in file\_type\_configs:***

***if re.findall(r"^\w+",line) :***

***if line in file\_name\_temp:***

***rete\_val = 1***

***else:***

***rete\_val = 0***

***else:***

***rete\_val = 0***

***file\_name = ''.join(file\_name)***

***#print(file\_name)***

***a = fuzz.ratio(file\_name\_string, file\_name)***

***#print('Score ---->',a)***

***if a > 50:***

***flag = 1***

***else :***

***flag = 0***

***if rete\_val == 1 :***

***return True***

***else:***

***return False***

***except:***

***print("Hi")***

Text

Description automatically generated

**Bioinformatics:**

File Path:

Text

Description automatically generated

Main file:

Text

Description automatically generated

To edit:



Chronology:

***The first function which will be going to execute is the main function.***

***def main():***

***try:***

***if sys.argv[1]!="" and sys.argv[1]!=None:***

***file\_paths=sys.argv[1].split("\n")***

***file\_paths=set(file\_paths)***

***for file\_path in file\_paths:***

***if ("filepart" not in file\_path) and (common\_const.arwruser not in file\_path) and ("berensftpuser" not in file\_path):***

***#print(file\_path)***

***try:***

***logging.info(file\_path)***

***sendmail\_movefile(file\_path)***

***except:***

***continue***

***except:***

***traceback.print\_exc()***

***if \_\_name\_\_ == '\_\_main\_\_':***

***main()***

Text

Description automatically generated

This will provide the location of file and then it will call the function with the path:

Code:

sendmail\_movefile(file\_path)



And then this function will be called, and this function will call another function where we have written all the important program for Bioinformatics data.

***Code:***

***def sendmail\_movefile(event\_path):***

***try:***

***#print("The event path"+event\_path)***

***#print("here")***

***staging\_path=check\_user\_dir(event\_path)***

Text

Description automatically generated

***Code:***

***def check\_user\_dir(event\_path):***

***try:***

***#absolute\_parent\_path\_landing=common\_const.watch\_parent\_dir***

***vendore\_lst=os.listdir("/sftpprod")***

***#print("The vendor lst "+str(vendore\_lst)) if enable\_print else None***

***event\_loc\_for\_vendor=event\_path.replace("/sftpprod","")***

***dir\_last\_name=event\_path.split("/")[-2]***

***#print("dir last name is :"+dir\_last\_name) if enable\_print else None***

***loc\_for\_legacy=event\_path.split("/")[4]***

***testing=event\_path.split("/")***

***leg\_final\_path=event\_path.replace(testing[2],"berenprod\_landing")***

***loc\_for\_bin=event\_path.split("/")[3]***

***#print(leg\_final\_path)***

***#print('here')***

***if loc\_for\_legacy in ["legacy"]:***

***shutil.move(event\_path,leg\_final\_path)***

***elif loc\_for\_bin=='clinical':***

***ret\_val=check\_file\_name(event\_path)***

***if ret\_val is False:***

***os.remove(event\_path)***

***sendmail.sendO365Email\_fn(event\_loc\_for\_vendor,common\_const.placement\_error)***

***return False***

***elif ret\_val==True:***

***for vend in vendore\_lst:***

***if vend in event\_path:***

***dest\_path=event\_path.replace(vend,common\_const.arwruser)***

***return dest\_path***

***elif loc\_for\_bin=='bioinformatics':***

***ret\_val=file\_siz\_check(event\_path)***

***ret\_val=True***

***if ret\_val == False:***

***os.remove(event\_path)***

***sendmail.sendO365Email\_fn(event\_loc\_for\_vendor,common\_const.placement\_error)***

***return False***

***elif ret\_val==True:***

***for vend in vendore\_lst:***

***if vend in event\_path:***

***dest\_path=event\_path.replace(vend,common\_const.arwruser)***

***return dest\_path***

Text

Description automatically generated

Below is the screenshot of bioinformatics function which is going to find out that if the data is bioinformatics, for now we don’t have any DTA for Bioinformatics data. But further if you want to make changes. Here we need to edit:

***Code:***

***elif loc\_for\_bin=='bioinformatics':***

***ret\_val=file\_siz\_check(event\_path)***

***ret\_val=True***

***if ret\_val == False:***

***os.remove(event\_path)***

***sendmail.sendO365Email\_fn(event\_loc\_for\_vendor,common\_const.placement\_error)***

***return False***

***elif ret\_val==True:***

***for vend in vendore\_lst:***

***if vend in event\_path:***

***dest\_path=event\_path.replace(vend,common\_const.arwruser)***

***return dest\_path***

Text

Description automatically generated

# Onboarding Changes

This function has been written to send out email to vendors and circulants team members.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

The main function is “sendmail.py”, but while onboarding new vendors we need to make changes in “constant.py”.

Text

Description automatically generated

If we need to change SMTP server then we need to edit “sendmail.py”.

***Code:***

***def get\_study\_type(file\_path,mailer\_dict):***

***try:***

***#print(file\_path)***

***file\_path=file\_path.replace("/sftpdata/Share","") if "/sftpdata/Share" in file\_path else file\_path***

***path\_first\_part=file\_path.split("/")[1]***

***#print(path\_first\_part)***

***for vals in mailer\_dict.keys():***

***if vals.lower() in path\_first\_part:***

***lst\_email=mailer\_dict[vals]***

***return lst\_email***

***final\_email\_id=["circulants\_beren\_support@circulants.com"]***

***return final\_email\_id***

***except:***

***final\_email\_id=["circulants\_beren\_support@circulants.com"]***

***return final\_email\_id***

Text

Description automatically generated