# AWS S3 Powershell file uploader

## Steps for Configuring and running Powershell script to upload files in AWS S3

Author: **Muthuvel Periyasamy**

# 1. Introduction

AWS S3 file uploder is developed using Powershell Scripts , this script can run in windows , linux and Mac machines. This Script will scan a pre-configured input directory, it will take the files arrived in the last 24 hours and send to the s3 bucket configured. Once a file is uploaded to s3 , it will move the file to the output directory , to prevent multiple upload of the same file.

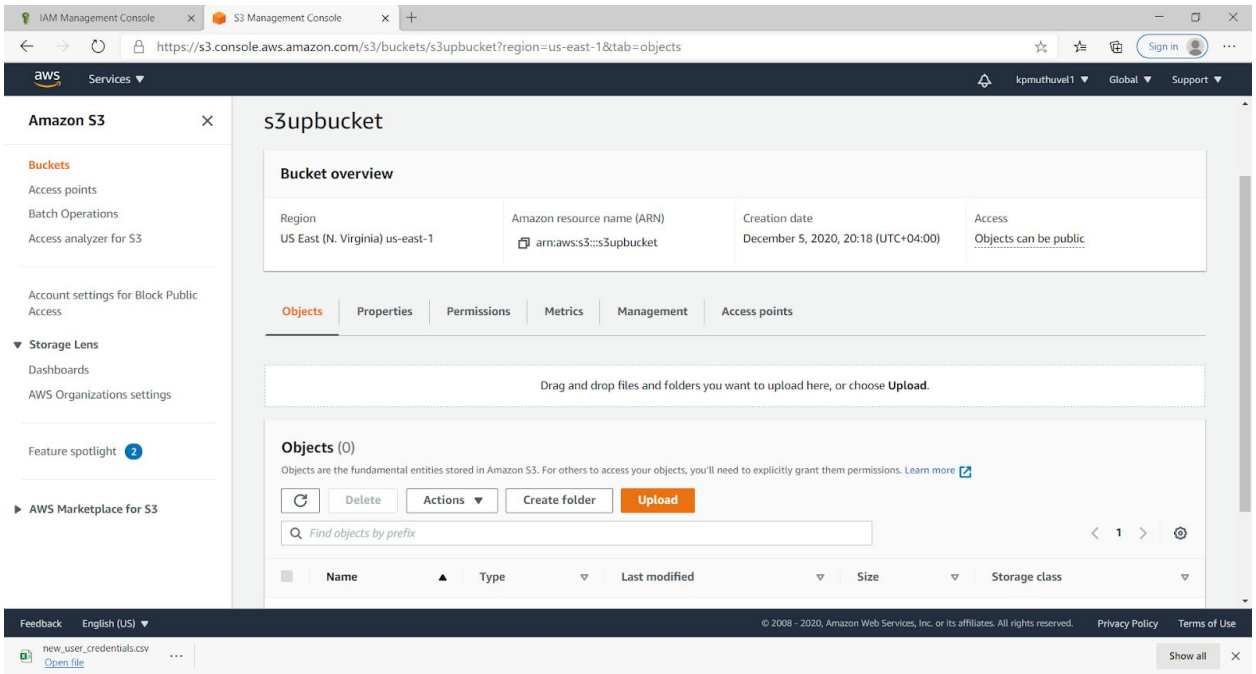
This project ensures that only secure users upload files into S3 bucket.

The following pre-request must be ready before using this powershell based s3 file uploader project.

1) Powershell 7 or later must be installed
2) AWSToolsAndSDKForNet_sdk-3.5.65.0_ps-4.1.5.0_tk-1.14.5.0.msi and AWS CLI must be installed for windows and equivalent software must be installed from linux and MacOS.
3) S3 bucket name must be created.
4) AWS user API Secret Key value.
5) S3 bucket policy to allow Powershell script to send files to it.

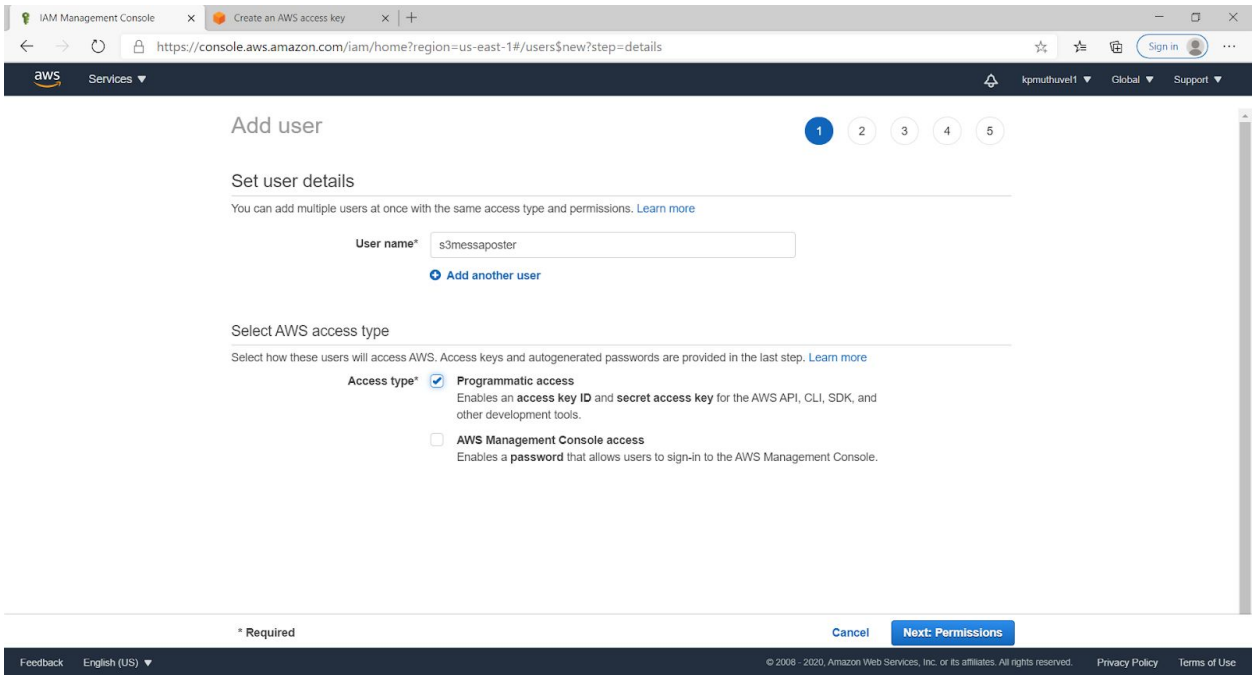
# 2. Configure S3 bucket

Login into AWS console and create a S3 bucket as shown below

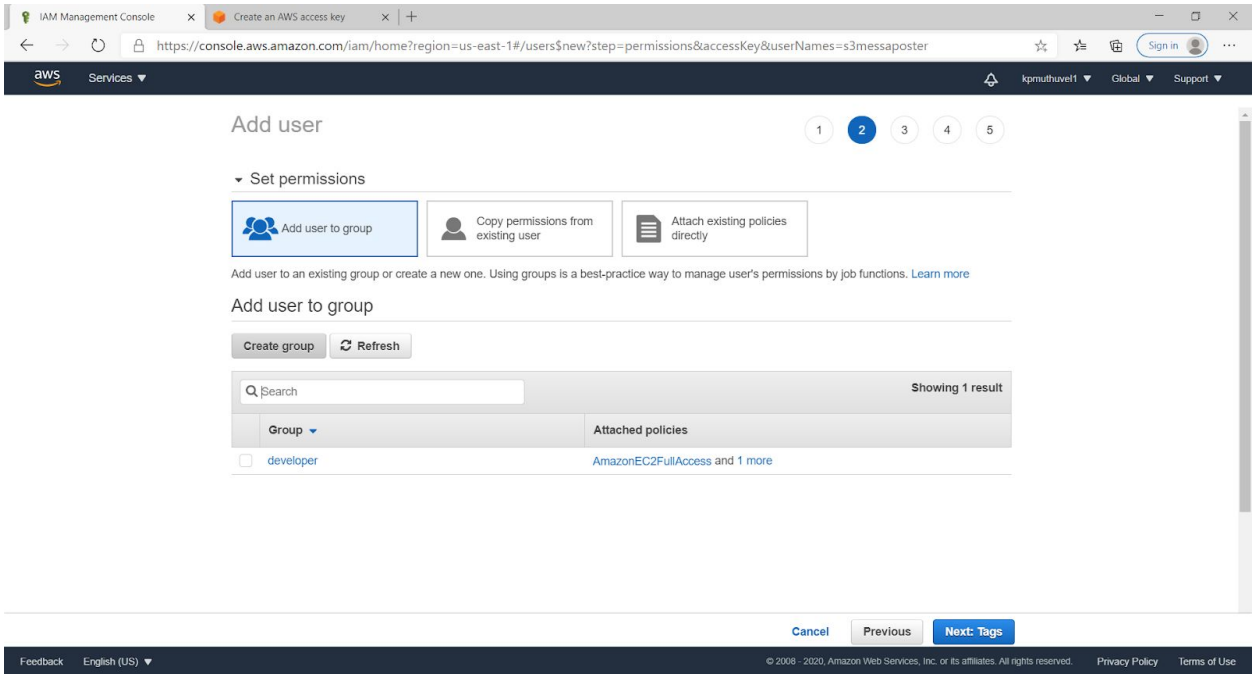



# 3. Create User with API access key

We need a user to load files from powershell script locations.
Login into AWS console and create a user  and select only Programmatic access as shown below

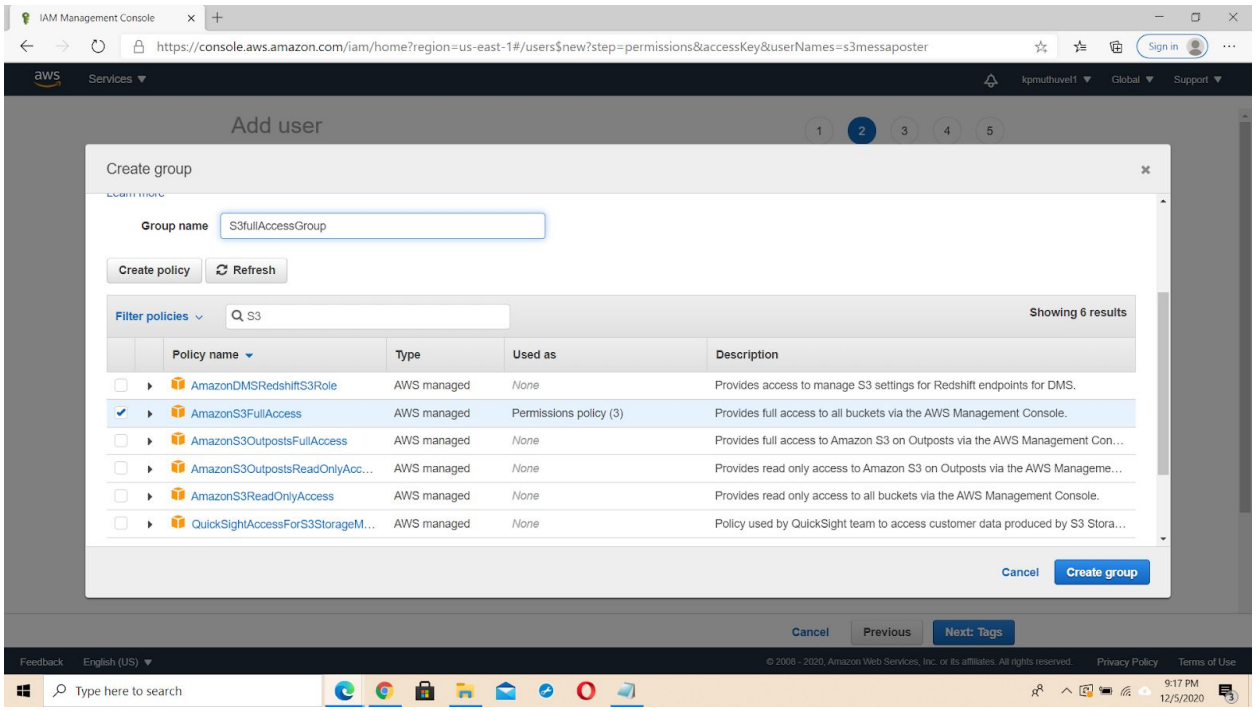


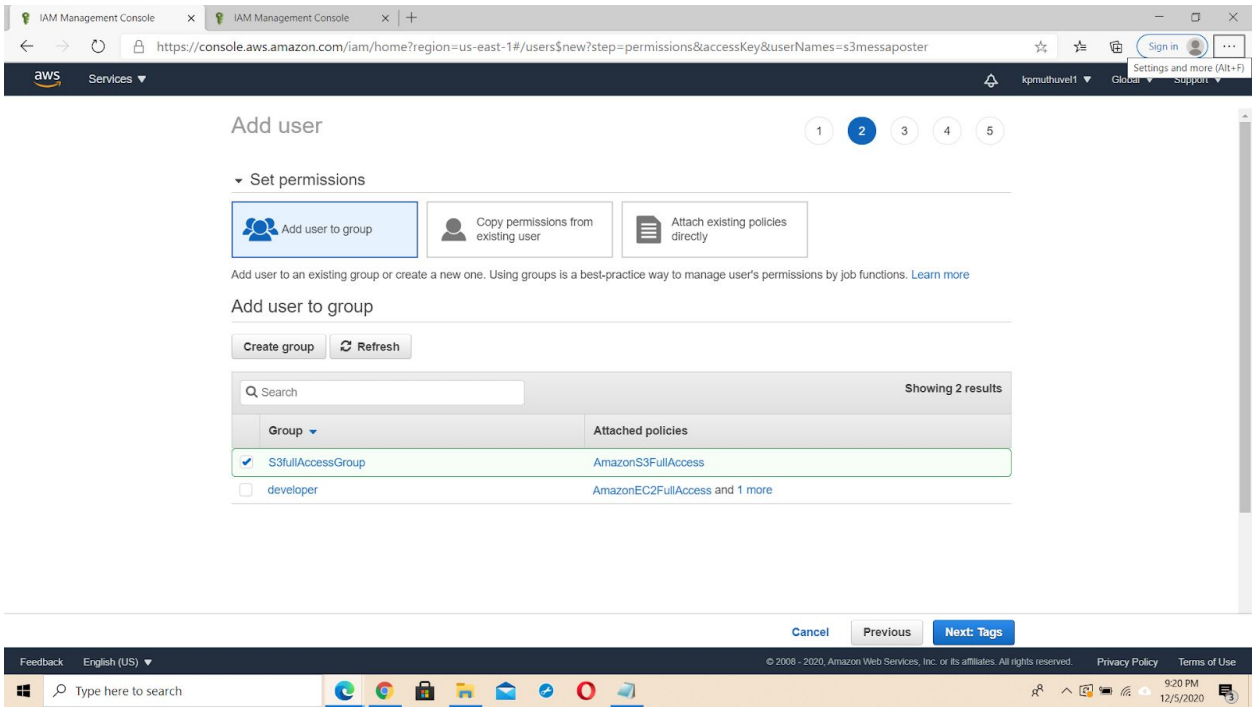

Click on Permissions button




Click on Create Group , if you want to create a new Group.

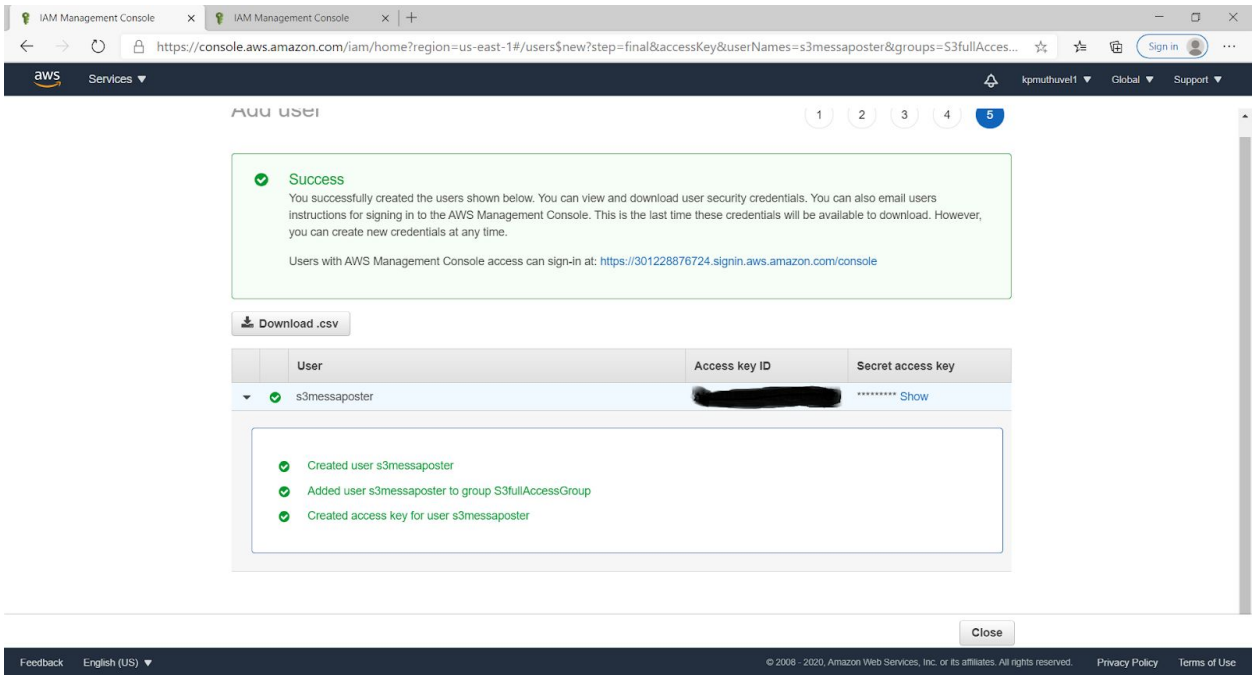
Select **AmazonS3FullAccess**




Click Next

Copy the Accesskey Id and Secret Access Key , you need these values in the powershell scripts.




# 4. Prepare S3 permission JSON script

Copy the user ARN , S3 ARN and update the below policy and update the S3 permission as shown below.

User ARN : arn:aws:iam::11111111:user/testuser
S3 ARN : arn:aws:s3:::s3update

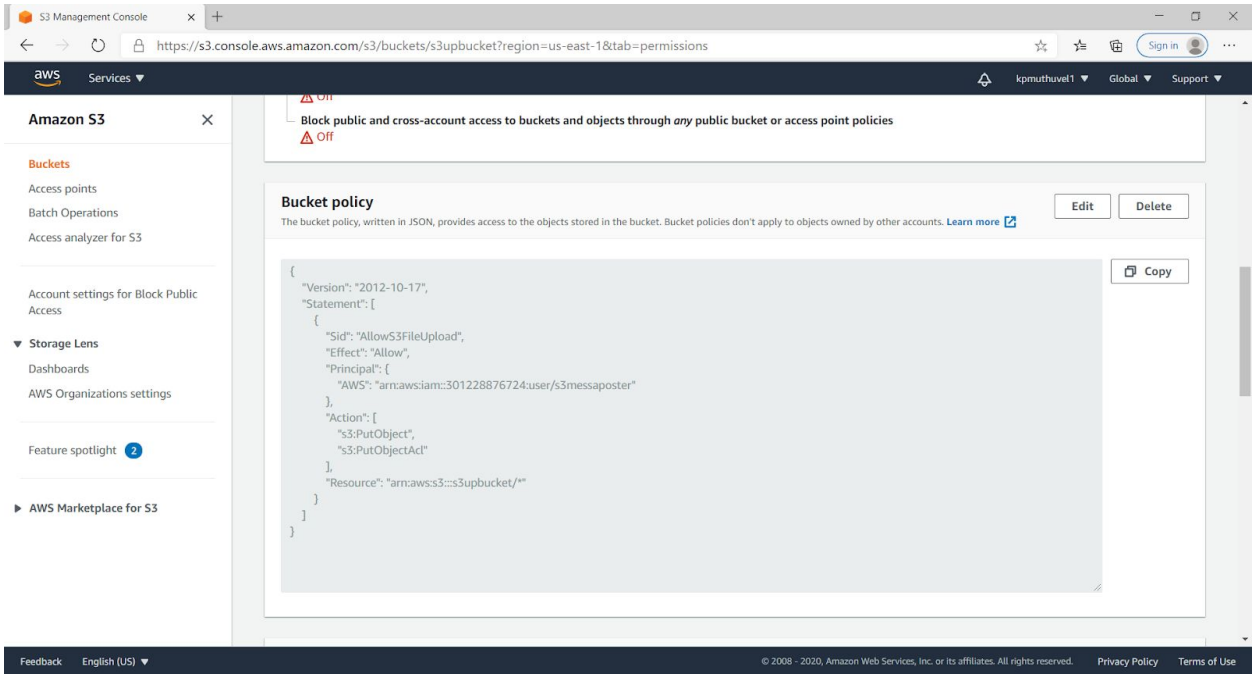
Update the highlighted and save in S3 permissions

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowS3FileUpload",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::11111111:user/testuser"
            ]
```

```
        },
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl"
        ],
        "Resource": "arn:aws:s3:::s3updatet/*"
    }
  ]
}
```

# 5. Update S3 permission with JSON

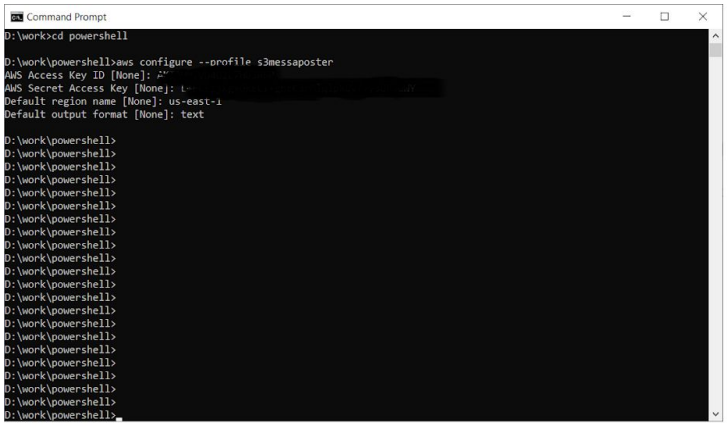
Copy the JSON policy created in the above steps and update S3 permission tab Bucket policy as shown below as shown below.




# 6. Configure AWS API key and Secret key in you local machine.

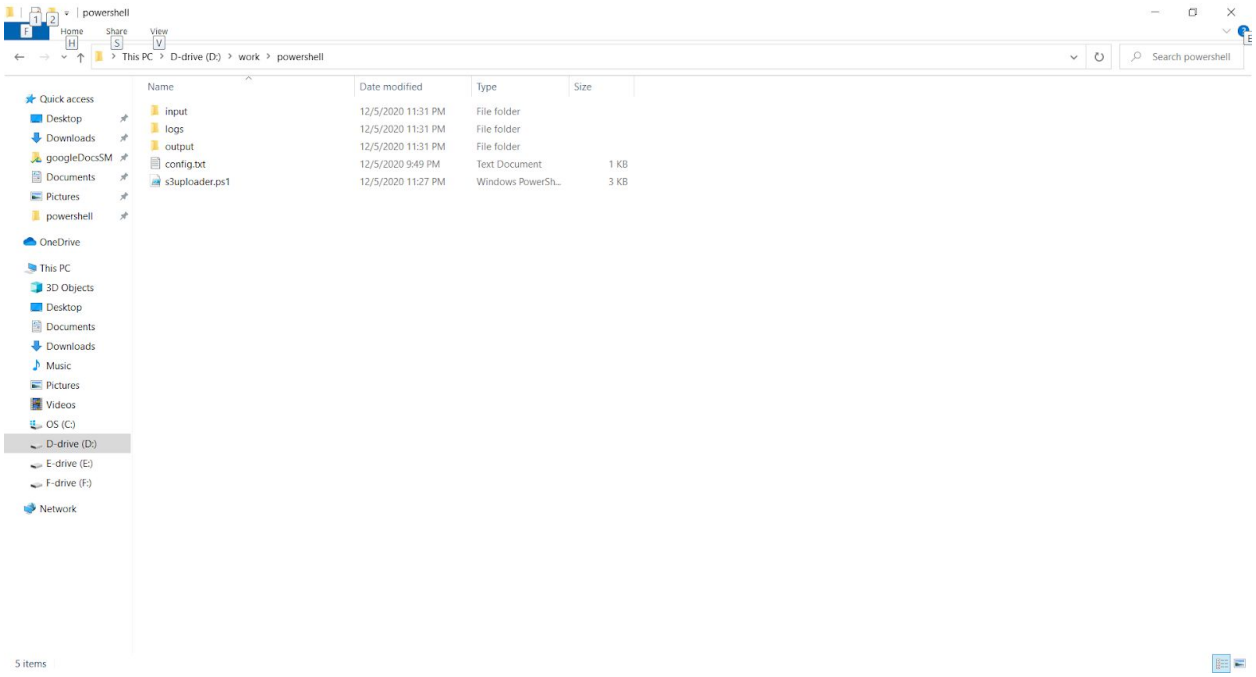
Enter below command in the location where you are going to run the script and enter the API KEY id and Secret Key as shown below.

aws configure --profile s3messaposter




# 7. Configure Deliverable.

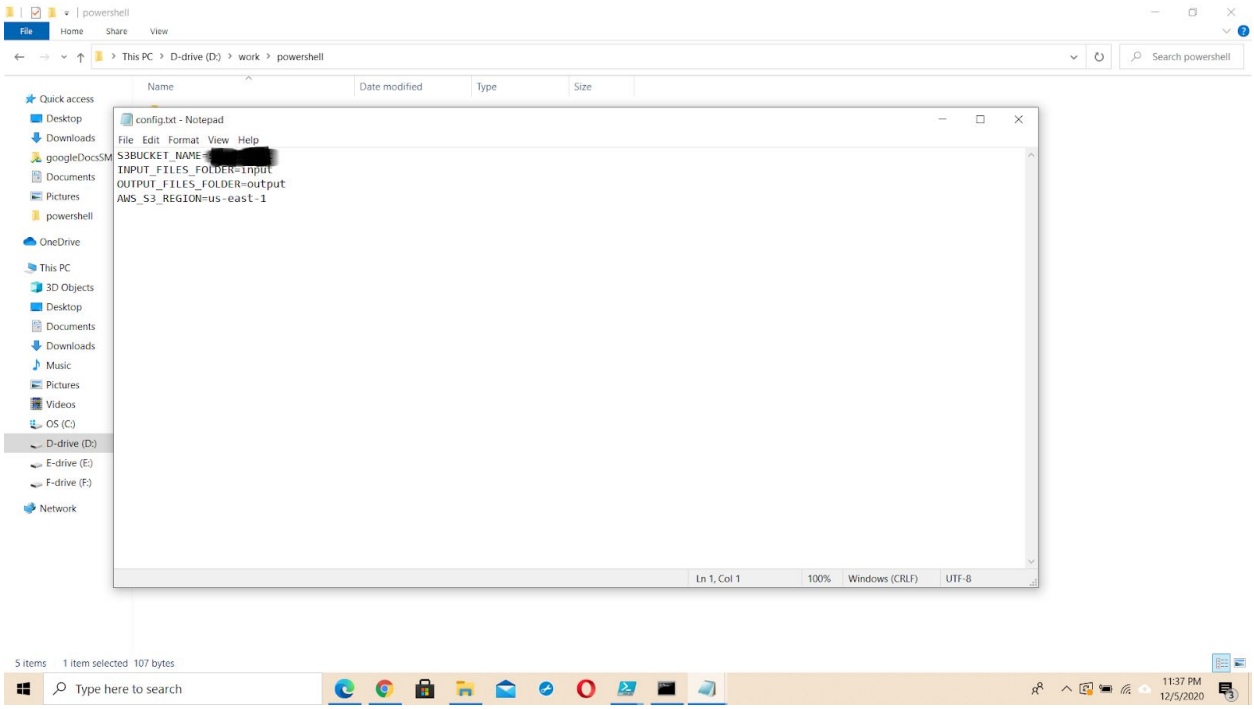
Unzip the deliverable , you will get a folder as shown below




Open the **config.txt** , you will get entries as shown below, enter your S3BucketName , InputFileDirectory path [Where you are going to store the input files needs to be uploaded into s3 bucket] , OutputFileDirectory path [once files are send to S3 , move that file to another directory, when the job runs again, it should not process the file already send to S3]

S3BUCKET_NAME=yourS3bucketName
INPUT_FILES_FOLDER=input
OUTPUT_FILES_FOLDER=output
AWS_S3_REGION=yourS3bucketRegion

Save the confi.txt file entries.




# 8. Run the PowerShell Script.

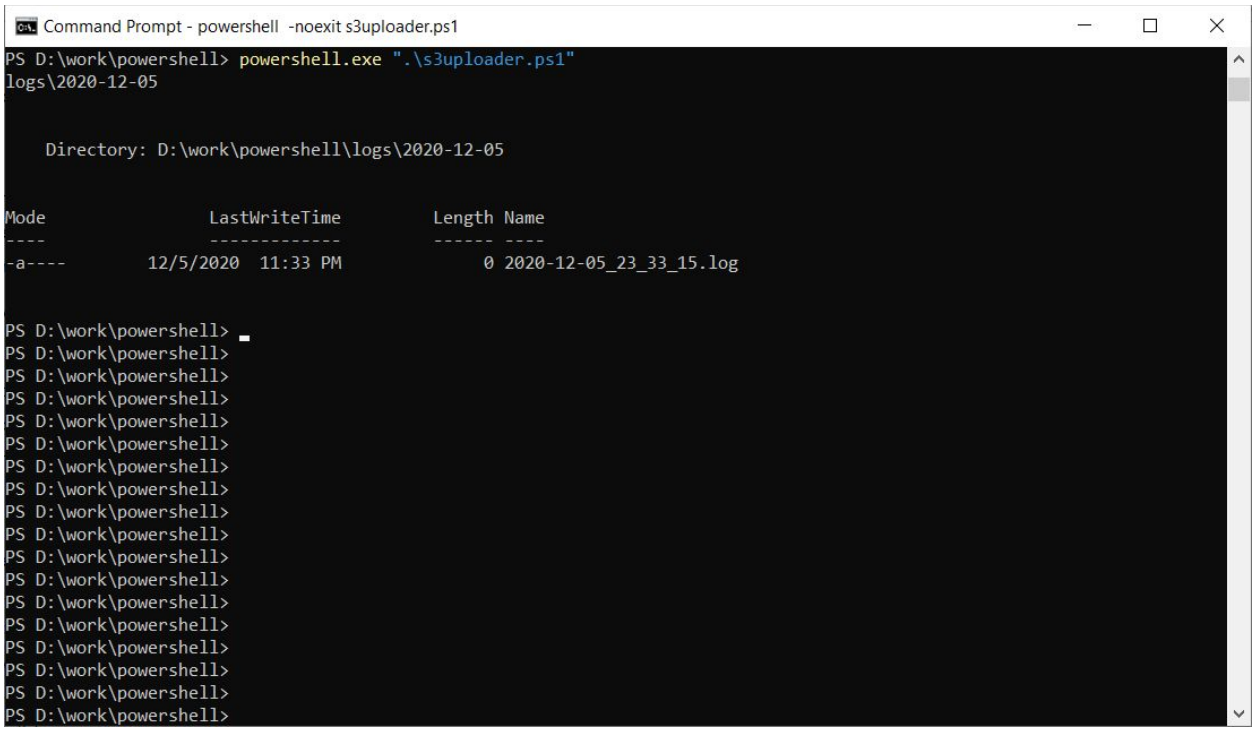
Goto the folder where you have unzipped the deliverables and run the script using below command

For Windows

**powershell.exe ".\s3uploader.ps1"**

**In case MacOS**

**pwsh ".\s3uploader.ps1"**



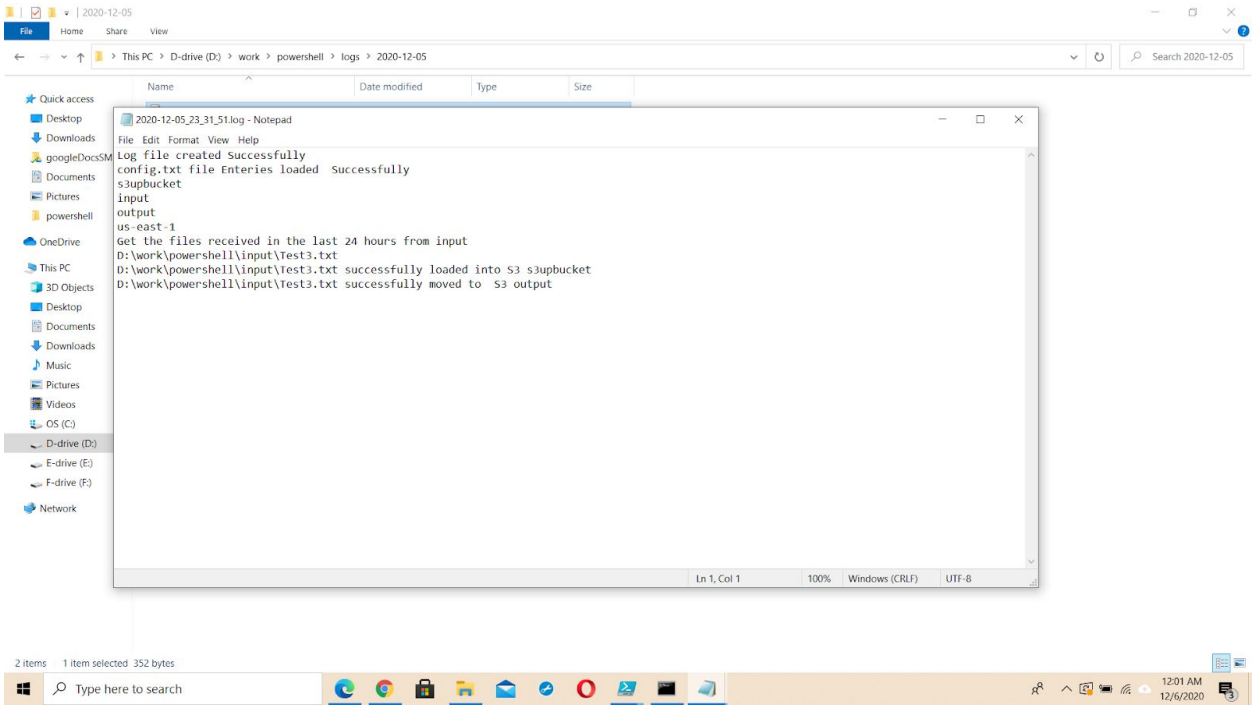

# 9. Schedule the Script.

You can use Windows Task Scheduler Standard procedure to Schedule this job to be run any interval you like.

Also you can use crontab in linux and MacOS to run the program any interval you want.

# 10. Logs

Scripts generate a log file on every single execution ,this file will help to identify the issues , files uploaded successfully into S3 and other vital informations.




# 11. Running the file uploaded in Linux and MacOS

Scripts can run in Linux and MacOS without any changes. Config.txt file entries needs to be changed to match with linux and MacOS path.

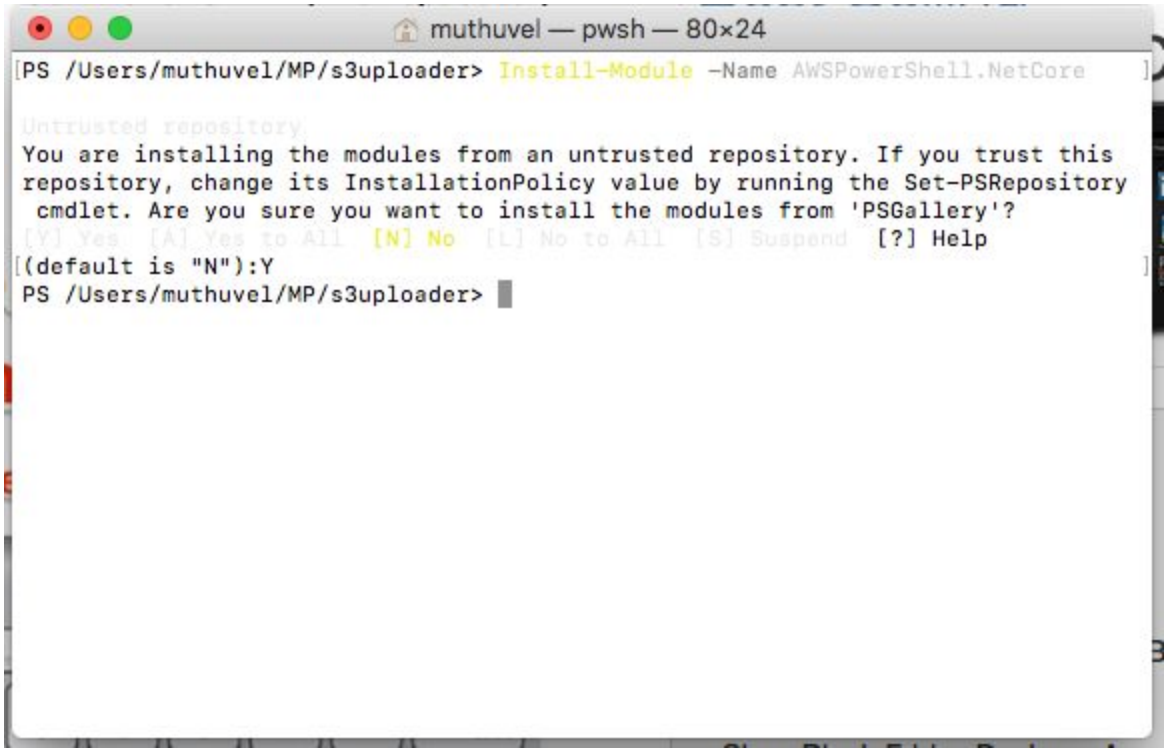
Following steps for installing PowerShell in MacOS

brew install --cask powershell

https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-macos?view=powershell-7.1

Install AWSPowerShell.NetCore using below command

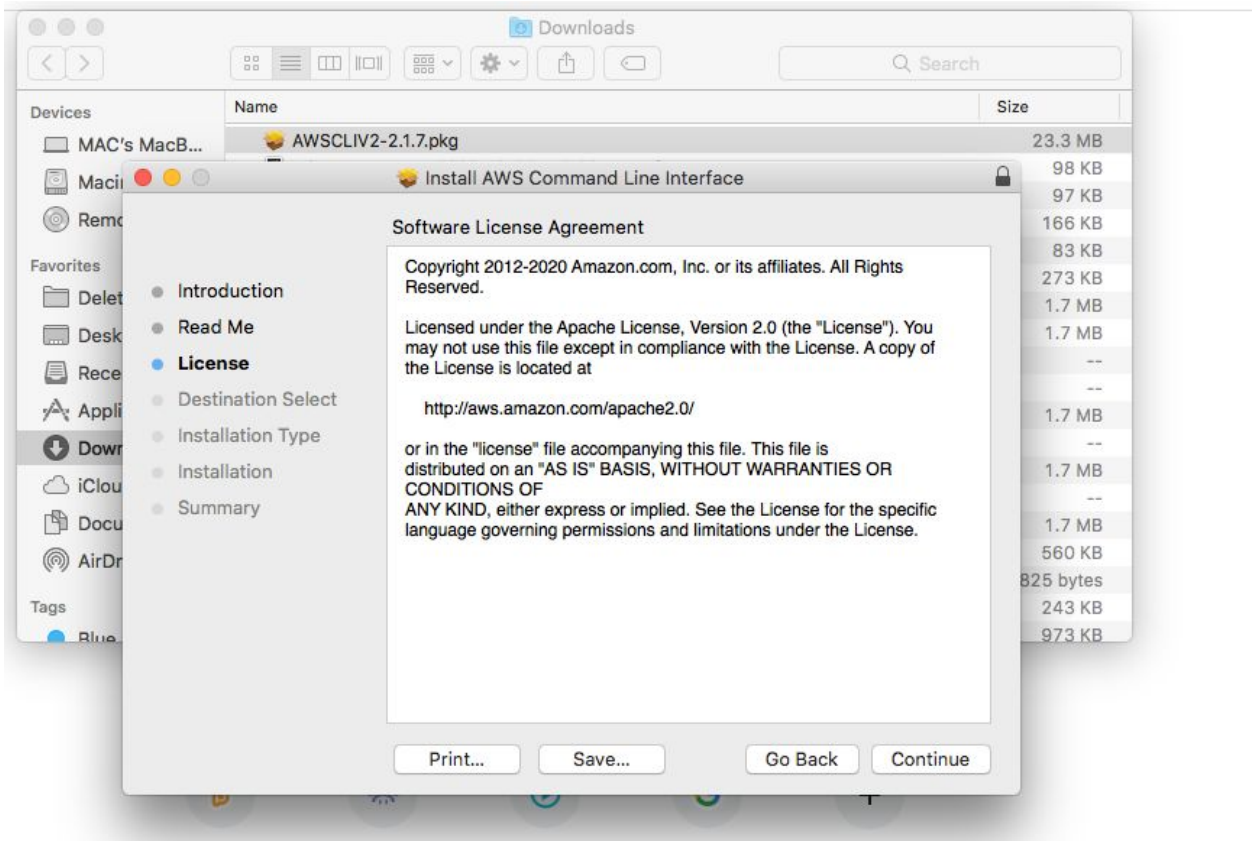
Install-Module -Name AWSPowerShell.NetCore

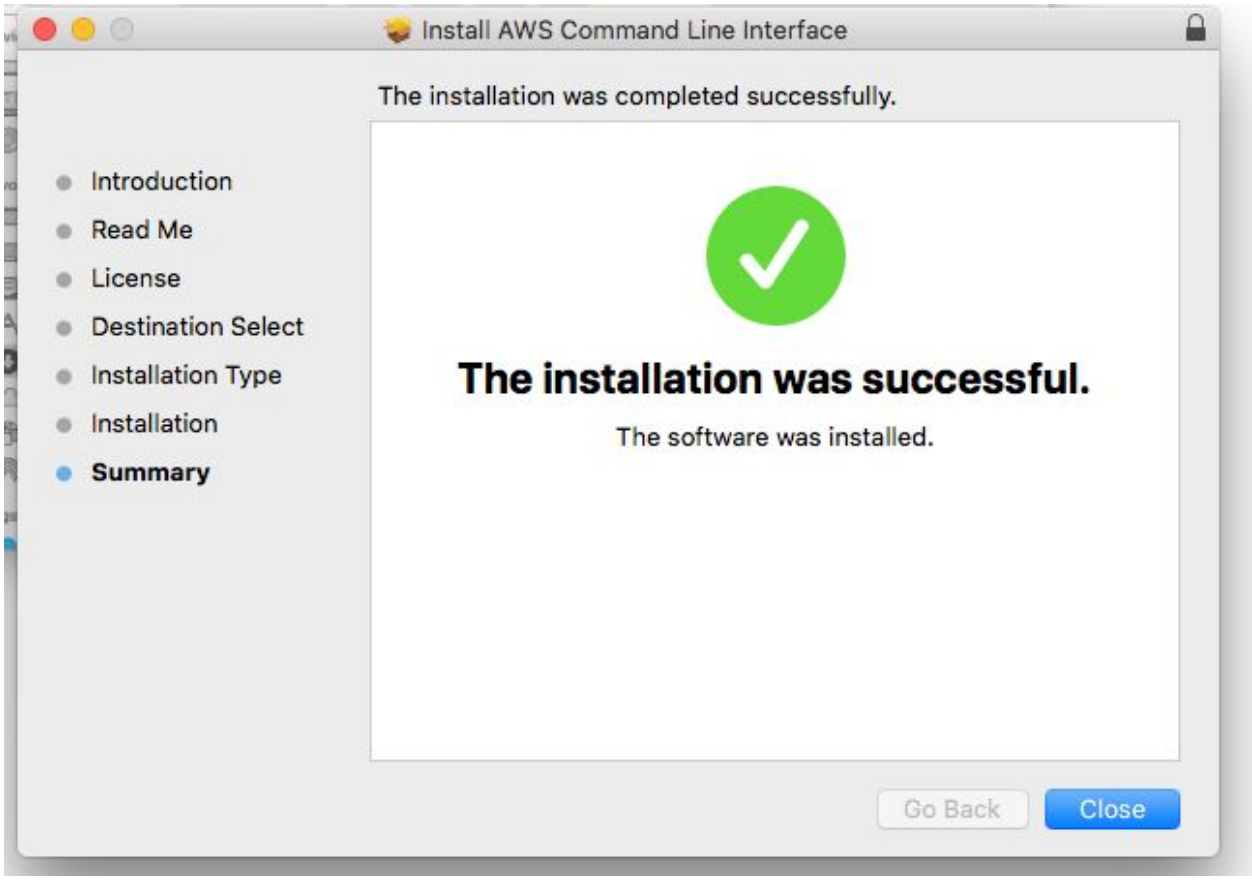



Download AWSCLIV2 using below link

https://awscli.amazonaws.com/AWSCLIV2-2.1.7.pkg

And Install AWSCLIV2 by double clicking the file.


Downloads
Search
Devices
MAC's MacB...
Favorites
Tags
Name
Size
AWSCLIV2-2.1.7.pkg
23.3 MB
Install AWS Command Line Interface
Software License Agreement
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
Copyright 2012-2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at
http://aws.amazon.com/apache2.0/
or in the "license" file accompanying this file. This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF
ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
Print...
Save...
Go Back
Continue



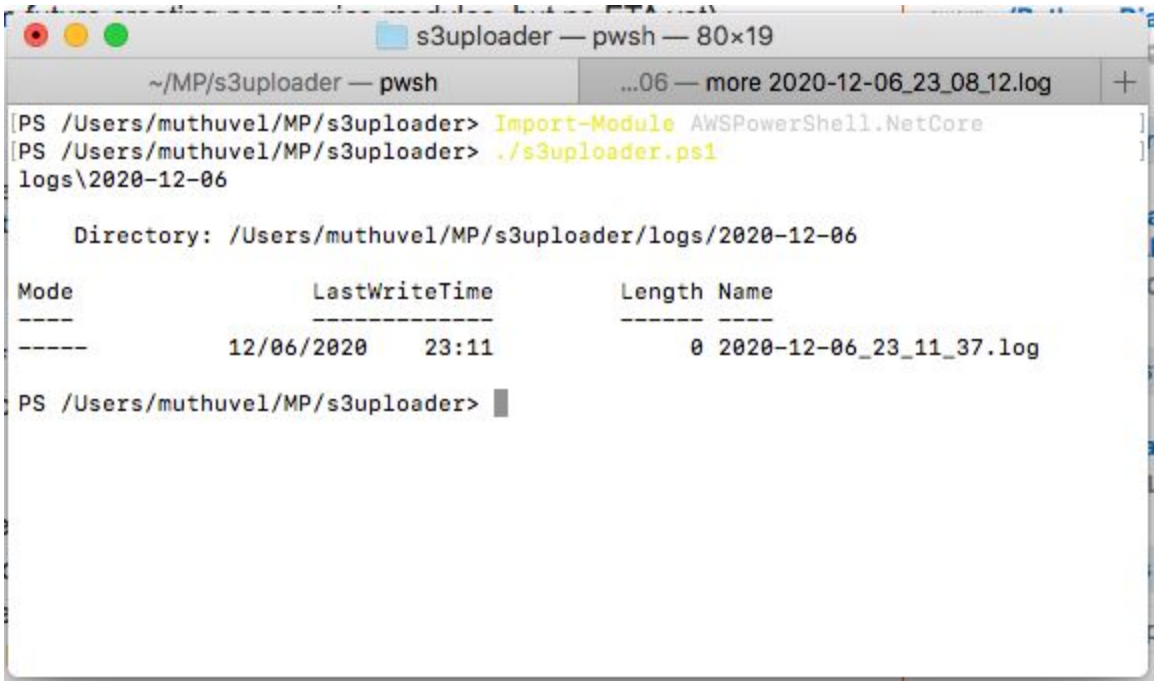

Cd to the directory where this powershell script is located and

Enter the below line command

**Import-Module AWSPowerShell.NetCore**

Run the powershell script using below command

**pwsh ./s3uploader.ps1**




# 12. Schedule the job to be run at every 15 minutes

Make the below entry in the **crontab -e**
**Please change the path to the location you unzip the deliverables**

***/15 * * * * cd /MP/s3uploader && pwsh ./s3uploader.ps1**

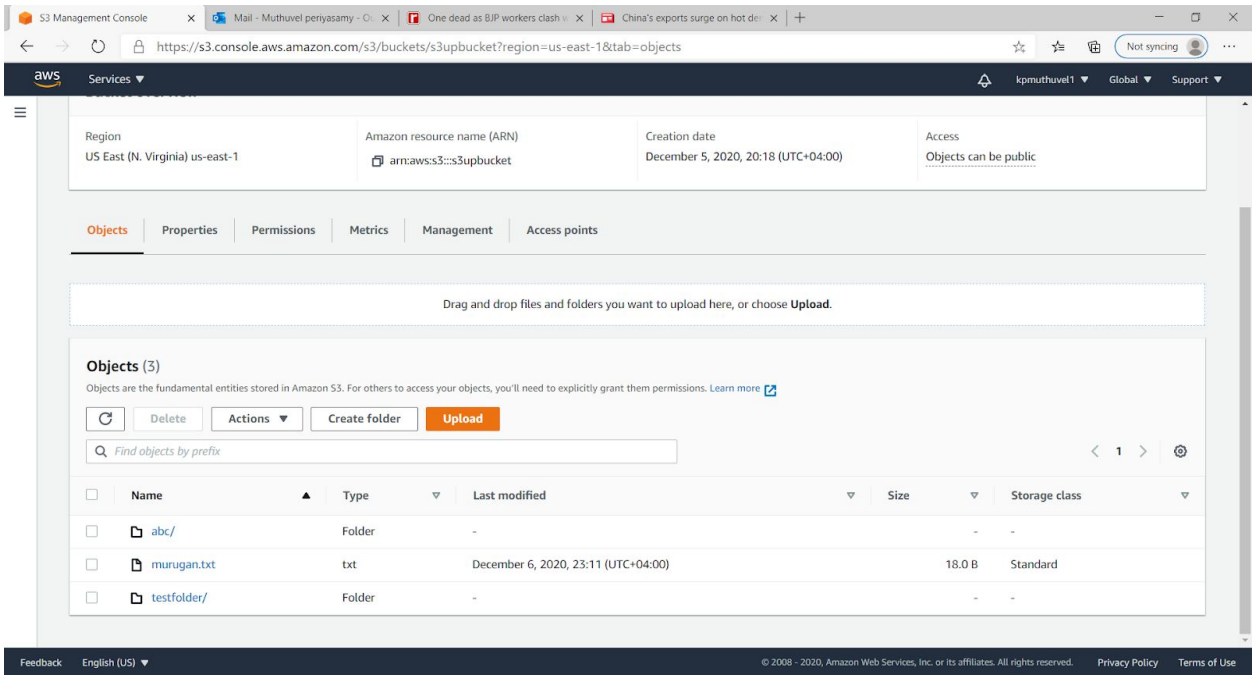
Save the above changes, it is will run automatically for every 15 minutes and if there is any new file it will upload in the server.

crontab -l

Will show your crontab entries.

# 13. Loading folders and its files

This script will load all the folders and files residing in the input directory, as show below.




# 14. Script source file

```
Function log([string]$fileName ,[string]$Message) {

   $Stamp = (Get-Date).toString("yyyy/MM/dd HH:mm:ss")
   $Line = "$Stamp $Message"
   Add-Content $fileName -Value $Line

}


$datestr = "logs\$((Get-Date).ToString('yyyy-MM-dd'))"
#Write-Host($datestr)
if (-not (Test-Path -LiteralPath $datestr)) {
   New-Item -ItemType Directory -Path $datestr
}

$fileName = $datestr+"\$((Get-Date).toString('yyyy-MM-dd_HH_mm_ss')).log"
if (-not (Test-Path -LiteralPath $fileName)) {
```

```
   New-Item -ItemType File -Path $fileName
}

Add-Content $fileName -Value "Log file created Successfully"

#$configval = Get-Content -Path config.json
Get-Content "config.txt" | foreach-object -begin {$h=@{}} -process { $k = [regex]::split($_,'=');
if(($k[0].CompareTo("") -ne 0) -and ($k[0].StartsWith("[") -ne $True)) { $h.Add($k[0], $k[1]) } }

Add-Content $fileName -Value "config.txt file Enteries loaded  Successfully"


$bucketName = $h.Get_Item("S3BUCKET_NAME")
$inputDir = $h.Get_Item("INPUT_FILES_FOLDER")
$outputDir = $h.Get_Item("OUTPUT_FILES_FOLDER")
#$awsKey = $h.Get_Item("AWS_S3_KEY")
$awsRegion = $h.Get_Item("AWS_S3_REGION")


Add-Content $fileName -Value $bucketName
Add-Content $fileName -Value $inputDir
Add-Content $fileName -Value $outputDir
#Add-Content $fileName -Value $awsKey
Add-Content $fileName -Value $awsRegion


$bucketName = $h.Get_Item("S3BUCKET_NAME")
#log(  $fileName, $bucketName )

$msg0 = 'Get the files received in the last 24 hours from '+$inputDir
Add-Content $fileName -Value $msg0

# for getting files/folders created the last 24 hours of the script running time
$files = Get-ChildItem $inputDir | Where-Object { $_.CreationTime -gt (Get-Date).AddDays(-1) }

# for getting files/folders created in the last 1 hour of the script running time
#$files = Get-ChildItem $inputDir | Where-Object { $_.CreationTime -gt (Get-Date).AddHours(-1)
}

# for getting files/folders created in the last 1 hour of the script running time
#$files = Get-ChildItem $inputDir | Where-Object { $_.CreationTime -gt
(Get-Date).AddMinutes(-30) }
```

```
for( $i=0;$i -lt $files.Count ; $i++) {

    Add-Content $fileName -Value $files[$i].FullName

    try {

        $isDir = (Test-Path -Path $files[$i].FullName -PathType Container)

        if($isDir){

            Write-S3Object -BucketName $bucketName -region $awsRegion $files[$i].Name -Folder
$files[$i].FullName -ProfileName s3messaposter -Recurse

            $msg1 = " folder "+$files[$i].FullName+" and its content uploaded successfully to
"+$bucketName

            Add-Content $fileName -Value $msg1

        } else {

            #Write-s3Object -BucketName $bucketName -Key $awsKey -region $awsRegion -File
$files[$i].FullName -ServerSideEncryption AES256 -ProfileName snippy
            Write-S3Object -BucketName $bucketName -region $awsRegion -File
$files[$i].FullName -ProfileName s3messaposter
            $msg1 = $files[$i].FullName+' successfully loaded into S3 '+$bucketName
            Add-Content $fileName -Value $msg1
            #Move-item –path $files[$i].FullName –destination $outputDir
            #$msg2 = $files[$i].FullName+' successfully moved to  S3 '+$outputDir
            #Add-Content $fileName -Value $msg2

        }

    }catch {
        $errMsg = $files[$i].FullName+' S3 upload failed:'+ $PSItem.ToString()
        Add-Content $fileName -Value $errMsg
    } finally {
    }
}
```