ABPAΞAΣ 설치

[설치] 데이터베이스 설치(PostgreSql)

데이터베이스 설치(PostgreSql)

- PostgreSql 설치

- sudo apt install postgresql

- PostgreSql 외부 접근 허용

- sudo vi /etc/postgresql/13/main/postgresql.conf
  -
```
listen_addresses = '*'                  # what IP address(es) to listen on;
#listen_addresses = 'localhost'         # what IP address(es) to listen on;
```

- sudo vi /etc/postgresql/13/main/pg_hba.conf
  -
```
# IPv4 local connections:
#host    all             all             127.0.0.1/32            md5
host    all             all             0.0.0.0/0               md5
# IPv6 local connections:
#host    all             all             ::1/128                 md5
host    all             all             ::/0                    md5
```

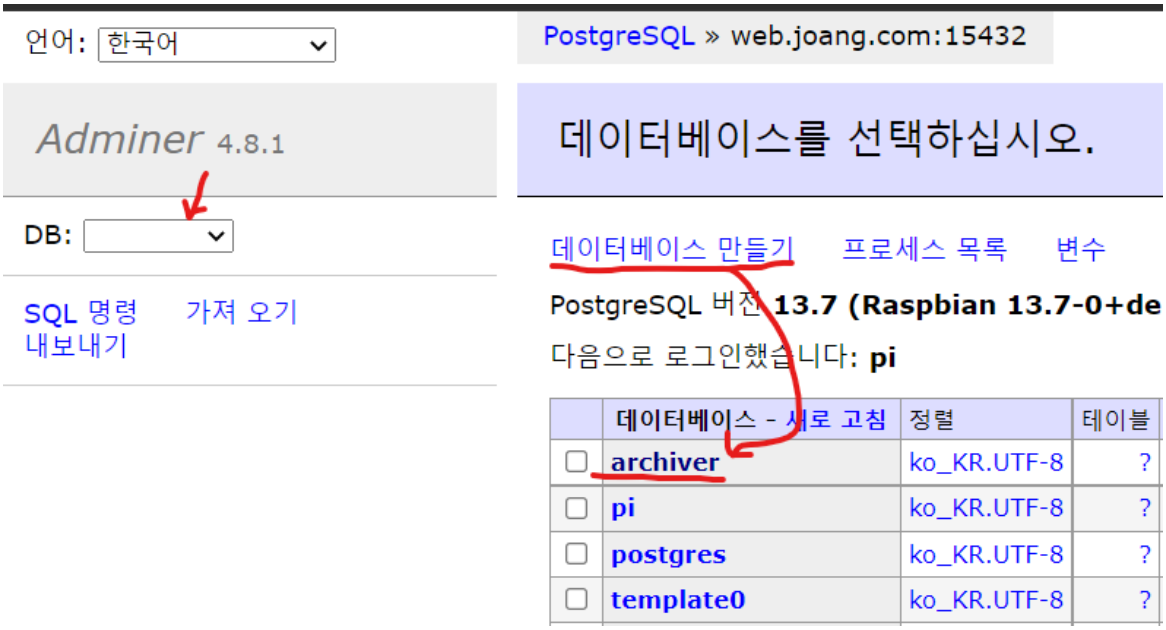
- 계정만들기

sudo -u postgres psql -c "ALTER ROLE postgres WITH password '여기패스워드'" postgres 계정으로 로그인 패스워드는 '여기패스워드'로 정한 패스워드

- adminer-4.8.1.php 설치

- sudo apt-get install adminer
- /var/www/html 에 adminer-4.8.1.php 복사

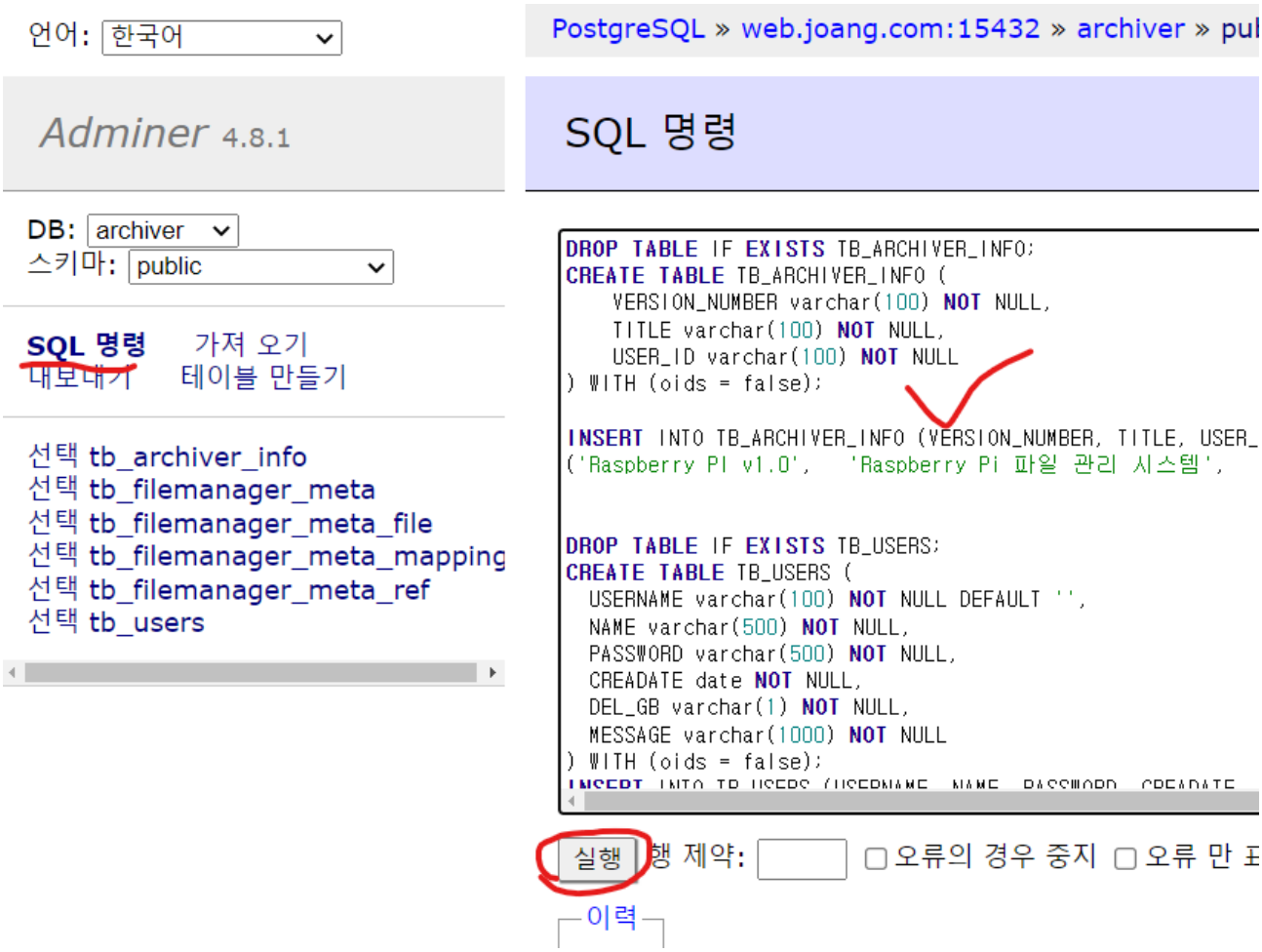
- Database 생성



- 사용자 계정 생성

- CREATE USER pi PASSWORD 'archiverpw123!@#' SUPERUSER;
  <-- 계정이나 패스워드를 바꾸면 업그레이드 시에 war를 배포 후 tomcat 폴더의 webapps/archiver/WEB-INF/spring/root-context.xml의 DB 접속 정보를 매번 바꾸어야 한다.

- Table 생성



- 실행 sql 파일 createTable.sql 열어서 창에 위와 같이 붙여서 실행

```
-- Adminer 4.8.1 PostgreSQL 13.11 (Raspbian 13.11-0+deb11u1) dump

\connect "archiver";

-- public.tb_archiver_sys_info definition

-- Drop table

DROP TABLE public.tb_archiver_sys_info;

CREATE TABLE public.tb_archiver_sys_info (
        sys_link varchar(500) NOT NULL,
        sys_context_root varchar(500) NOT NULL,
        sys_title varchar(100) NOT NULL,
        sys_name varchar(100) NOT NULL,
        sys_version varchar(100) NOT NULL,
        sys_owner varchar(100) NOT NULL,
        sys_date varchar(100) NOT NULL,
```

```
        sys_manual_link varchar(100) NOT NULL,
        sys_etc varchar(4000) NULL,
        sys_message varchar(4000) NULL,
        sys_storage varchar(1000) NULL,
        sys_backup_location varchar(4000) NULL,
        sys_backup_yn varchar(2) NOT NULL DEFAULT 'N'::character varying,
        sys_backup_date timestamp NULL
);


INSERT INTO "tb_archiver_sys_info" ("sys_link", "sys_context_root",
"sys_title", "sys_name", "sys_version", "sys_owner", "sys_date",
"sys_manual_link", "sys_etc", "sys_message", "sys_storage",
"sys_backup_location", "sys_backup_yn", "sys_backup_date") VALUES
('http://web.joang.com:8082',   'archiver',     'ABPAΞAΣ',      '아브라삭
스',     '0.5',     '류현수',   '2023.05.28',      'http://web.joang.com:6875/books/
abraksas-system',       '관리자 류현수 ',    '관리자 화면 완료 ',   '/media/pi/RaspRefp/
data',  '/media/pi/RaspRefp/backup/ArchiverBackup',     'N',    '2023-06-06
07:39:02.571868');

DROP TABLE IF EXISTS "tb_filemanager_meta";
DROP SEQUENCE IF EXISTS seq_meta;
CREATE SEQUENCE seq_meta INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807
CACHE 1;

CREATE TABLE public.tb_filemanager_meta (
        idx int4 NOT NULL DEFAULT nextval('seq_meta'::regclass),
        user_id varchar(100) NOT NULL,
        meta varchar(1000) NOT NULL,
        crea_dtm date NOT NULL,
        crea_id varchar(100) NOT NULL,
        del_gb varchar(1) NOT NULL DEFAULT 'N'::character varying
);


DROP TABLE IF EXISTS "tb_filemanager_meta_file";
DROP SEQUENCE IF EXISTS seq_file;
CREATE SEQUENCE seq_file INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807
CACHE 1;

-- DROP TABLE public.tb_filemanager_meta_file;


CREATE TABLE public.tb_filemanager_meta_file (
        idx int4 NOT NULL DEFAULT nextval('seq_file'::regclass),
        file_title varchar(1000) NOT NULL,
        file_detail text NOT NULL,
        original_file_name varchar(500) NOT NULL,
        stored_file_name varchar(1000) NOT NULL,
        file_size int8 NOT NULL,
```

```
        file_type varchar(10) NOT NULL,
        crea_dtm date NOT NULL,
        crea_id varchar(100) NOT NULL,
        del_gb varchar(1) NOT NULL DEFAULT 'N'::character varying,
        down_yn varchar(1) NULL,
        error varchar(10000) NULL,
        star_level numeric NOT NULL
);


DROP TABLE IF EXISTS "tb_filemanager_meta_mapping";
DROP SEQUENCE IF EXISTS seq_meta_mapp;
CREATE SEQUENCE seq_meta_mapp INCREMENT 1 MINVALUE 1 MAXVALUE
9223372036854775807 CACHE 1;

-- DROP TABLE public.tb_filemanager_meta_mapping;

CREATE TABLE public.tb_filemanager_meta_mapping (
        idx int4 NOT NULL DEFAULT nextval('seq_meta_mapp'::regclass),
        user_id varchar(100) NOT NULL,
        meta_idx int4 NOT NULL,
        file_idx int4 NOT NULL,
        crea_dtm date NOT NULL,
        crea_id varchar(100) NOT NULL,
        del_gb varchar(1) NOT NULL DEFAULT 'N'::character varying
);


DROP TABLE IF EXISTS "tb_filemanager_meta_ref";
CREATE TABLE public.tb_filemanager_meta_ref (
        from_idx int4 NOT NULL,
        to_idx int4 NOT NULL,
        meta_ref int4 NOT NULL,
        crea_dtm date NOT NULL,
        crea_id varchar(100) NOT NULL,
        del_gb varchar(1) NOT NULL DEFAULT 'N'::character varying,
        updt_dtm date NULL
);


DROP TABLE IF EXISTS "tb_users";
CREATE TABLE public.tb_users (
        username varchar(100) NOT NULL DEFAULT ''::character varying,
        "name" varchar(500) NOT NULL,
        "password" varchar(500) NOT NULL,
        userrole varchar(3) NOT NULL,
        creadate date NOT NULL,
        del_gb varchar(1) NOT NULL,
        message varchar(1000) NOT NULL,
        main_image_path varchar NOT NULL DEFAULT '/images/
```

```
ABPAΞAΣ.jpg'::character varying,
        logindate timestamp NULL
);


DROP TABLE IF EXISTS "tb_users_mapping";
DROP SEQUENCE IF EXISTS seq_file_user_mapping;
CREATE SEQUENCE seq_file_user_mapping INCREMENT 1 MINVALUE 1 MAXVALUE
9223372036854775807 CACHE 1;

CREATE TABLE public.tb_users_mapping (
        idx int4 NOT NULL DEFAULT nextval('seq_file_user_mapping'::regclass),
        username varchar(100) NOT NULL,
        file_idx int4 NOT NULL,
        crea_dtm date NOT NULL,
        crea_id varchar(100) NOT NULL
);

COMMENT ON TABLE "public"."tb_users_mapping" IS '파일 공유 사용자 정보';


-- 2023-06-07 22:24:57.676591+09
```

- ```
  INSERT INTO public.tb_users
  (username, "name", "password", userrole, creadate, del_gb, message)
  VALUES('guest', 'guest', 'guest', 'gst', '2023-07-17', 'N', 'GUEST 입니다.');


  INSERT INTO public.tb_users
  (username, name, "password", userrole, creadate, del_gb, message)
  VALUES('archiveradmin', '관리자','17xxxx21q', 'adm', '2023-09-04', 'N', '관리자님 안녕하세요 ');
  ```

  - id : archiveradmin password : archiveradmin <-- 패스워드는 바꾸자

[설치] Web Application Server 설치 (Tomcat)

- Java 설치

- sudo apt install default-jdk

- Tomcat

- sudo mkdir -p /app
- sudo chown -c pi.pi /app <-- 권한을 pi 계정에 주기
- apache-tomcat-9.0.64.tar.gz를 Download에 받는다
- mkdir -p /home/pi/Downloads/tomcat <-- 압축을 풀 폴더를 만든다.
  ```

- tar zxvf ./apache-tomcat-9.0.64.tar.gz -C ~/Downloads/tomcat <-- ~/Downloads/tomcat 폴더에 압축을 푼다.
- cp -Rf ./apache-tomcat-9.0.64 /app <-- 모두 복사

- 자동실행 등록

- 재시작 시에 자동 시작 등록
  sudo vi /etc/systemd/system/tomcat.service

```
[UNIT]
Description=apache-tomcat-9.0.64
After=syslog.target network.target

[Service]
Type=forking

Environment="CATALINA_HOME=/app/apache-tomcat-9.0.64"
Environment="CATALINA_BASE=/app/apache-tomcat-9.0.64"
Environment="CATALINA_OPTS=-Xms2048M -Xmx2048M -server -XX:+UseParallelGC"

ExecStart=/app/apache-tomcat-9.0.64/bin/startup.sh
ExecStop=/app/apache-tomcat-9.0.64/bin/shutdown.sh

User=pi
Group=pi
UMask=0000
RestartSec=10

[Install]
WantedBy=multi-user.target
```

- 서비스 등록
  sudo systemctl enable tomcat.service
- catalina.sh를 /app/apache-tomcat-9.0.64/bin 에 복사, 환경 설정을 추가함 아래 내용 참고

```
- UMASK="0027"를 UMASK="0000"

  # Achiver ADD 2022.06.30
  JAVA_OPTS="$JAVA_OPTS -Xmx2048m -Xms2048m"
```

재시작

- sudo reboot

[설치] Application 설치

Application 설치

http://web.joang.com:8082/archiver.war 를 받아서 app/apache-tomcat-9.0.64/webapps에 war 파일 복사

재시작을 하지 않아도 되지만 재시작 필요 시에 sudo systemctl restart tomcat.service 명령어 수행

[설치] 배치 구성하기 (Youtube, 백업)

Download YouTube

/home/pi/pythonDownloadCrontab/KKYouTuBeDownloader.py 생성

```
from postgresqlDatabase import Databases
import yt_dlp
import datetime
import os
import re
import argparse, logging, logging.config, conf

today = datetime.date.today()
y = today.year
m = today.month
d = today.day
stored_file_name = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
base_dir = ""
dir = "/{year}/{month}/{day}/".format(year=y,month=m,day=d)
downloadfilename = ""
metadata = ""

# Log
logging.config.dictConfig(conf.dictConfig)
logger = logging.getLogger(__name__)

class CRUD(Databases):
    def insertDB(self,schema,table,colum,data):
        sql = " INSERT INTO {schema}.{table}({colum}) VALUES
('{data}') ;".format(schema=schema,table=table,colum=colum,data=data)
        try:
            self.cursor.execute(sql)
        except Exception as e :
            logger.debug(" insert DB err, e="+str(e))

    def readDB(self,table,colum, condition):
        sql = " SELECT {colum} from {table} where 1=1
```

```
{condition}".format(colum=colum,table=table, condition=condition)
        try:
            self.cursor.execute(sql)
            result = self.cursor.fetchall()
        except Exception as e :
            result = (" read DB err",str(e))

        return result

    def readDBbySQL(self,sql):
        try:
            self.cursor.execute(sql)
            result = self.cursor.fetchall()
        except Exception as e :
            result = (" read DB err",str(e))

        return result

    def updateDB(self,table,colum,value,condition):
        sql = " UPDATE {table} SET {colum}='{value}' WHERE 1=1 and
{condition}::integer ".format(table=table ,
colum=colum ,value=value,condition=condition )
        try :
            self.cursor.execute(sql)
        except Exception as e :
            logger.debug(" update DB err , e="+str(e))

    def deleteDB(self,schema,table,condition):
        sql = " delete from {schema}.{table} where {condition} ;
".format(schema=schema,table=table,
        condition=condition)
        try :
            self.cursor.execute(sql)
        except Exception as e:
            logger.debug( "delete DB err , e="+str(e))

    def commit(self):
        try :
            self.db.commit()
        except Exception as e:
            logger.debug( "delete DB err , e="+str(e))

    def downLoadMovieFileByURL(self,downloadurl, idx, file_title):
        global downloadfilename
        logger.debug( ">> Download Movie URL:" + downloadurl)
        ydl_opts = {
             'verbose': True,
             'format': 'bestvideo[ext=mp4]+bestaudio[ext=m4a]/bestvideo+bestaudio/
best',
             'merge_output_format': 'mp4',
```

```
            'outtmpl': base_dir + dir + stored_file_name + file_title + '.%(ext)s',
            'extractor_args': {
               'youtube': {
                   'player_client': ['default', '-tv_simply'],
                   'player_js_version': ['actual'],
               },
            }
        }
        try:
            yt_dlp.YoutubeDL(ydl_opts).cache.remove()
            ydl = yt_dlp.YoutubeDL(ydl_opts).extract_info(downloadurl, download=True)
            downloadfilename = ydl["title"]
            metadata = ydl['description']
            logger.debug(">> Movie file name : " + downloadfilename)
            logger.debug(">> Movie metadata : " + metadata )
            db.updateDB('tb_filemanager_meta_file', 'file_detail', downloadurl+"\n\r"+
metadata.replace("'","\""), 'idx='+idx)
            return True
        except Exception as e:
            logger.debug("Error e="+str(e))
            db.updateDB('tb_filemanager_meta_file', 'down_yn', 'E', 'idx='+idx)
            db.updateDB('tb_filemanager_meta_file', 'error',
'@downLoadMovieFileByURL:'+str(e), 'idx='+idx)
            return False

    def downLoadMusicFileByURL(self,downloadurl, idx, file_title):
        global downloadfilename
        logger.debug( ">> Download Music URL:" + downloadurl)
        ydl_opts = {
             'verbose': True,
            'format': 'bestaudio/best',
            'postprocessors': [{
                   'key': 'FFmpegExtractAudio',
                   'preferredcodec': 'mp3',
                   'preferredquality': '192',
             }, {'key': 'FFmpegMetadata'},],
            'outtmpl': base_dir + dir + stored_file_name + file_title + '.%(ext)s'
        }
        try:
            yt_dlp.YoutubeDL(ydl_opts).cache.remove()
            ydl = yt_dlp.YoutubeDL(ydl_opts).extract_info(downloadurl, download=True)
            downloadfilename = ydl["title"]
            metadata = ydl['description']
            logger.debug(">> Music file name : " + downloadfilename )
            logger.debug(">> Music metadata : " + metadata )
            db.updateDB('tb_filemanager_meta_file', 'file_detail', downloadurl+"\n\r"+
metadata.replace("'","\""), 'idx='+idx)
            return True
        except Exception as e:
            logger.debug("Error e=" + str(e))
```

```
            db.updateDB('tb_filemanager_meta_file', 'down_yn', 'E', 'idx='+idx)
            db.updateDB('tb_filemanager_meta_file', 'error',
'@downLoadMusicFileByURL:'+str(e), 'idx='+idx)
            return False

    def updateAndRename(self, idx, downloadFileType):
        try:
            logger.debug("> Download Original file name = " + base_dir + dir +
downloadfilename + ", type=" + downloadFileType)
            changeOrgFileName = re.sub("'", "", re.sub("[/]", "_",
re.sub('[\/:*?"<>|]','',downloadfilename)))
            logger.debug("> Change Original file name = " + base_dir + dir +
changeOrgFileName + ", type=" + downloadFileType)
            logger.debug("> Change Store file name = " + base_dir + dir +
stored_file_name + ", type=" + downloadFileType)

            if(downloadFileType == "mov"):
                final_changeOrgFileName = changeOrgFileName+'.mp4'
                final_stored_file_name = stored_file_name+'.mp4'
            elif(downloadFileType == "muc"):
                final_changeOrgFileName = changeOrgFileName+'.mp3'
                final_stored_file_name = stored_file_name+'.mp3'

            for filename in os.listdir(base_dir + dir):
                logger.debug(">> filename = " + filename + " , stored_file_name=" +
stored_file_name)
                if filename.startswith(stored_file_name):
                    logger.debug("> Change file name ("+idx+")= " + base_dir + dir +
filename + " to " + base_dir + dir + final_stored_file_name)
                    os.rename(base_dir + dir + filename, base_dir+dir +
final_stored_file_name)
                    logger.debug("> Changed file stored name=" +
final_stored_file_name)
                    logger.debug("> Changed file Org name=" + final_changeOrgFileName)

            logger.debug("original_file_name ("+idx+")= " + final_changeOrgFileName)
            db.updateDB('tb_filemanager_meta_file', 'original_file_name',
final_changeOrgFileName, 'idx='+idx)
            logger.debug("stored_file_name ("+idx+")= " + dir +
final_stored_file_name)
            db.updateDB('tb_filemanager_meta_file', 'stored_file_name', dir +
final_stored_file_name, 'idx='+idx)
            logger.debug("file_size ("+idx+")= " + base_dir + dir +
final_stored_file_name)
            logger.debug("file_size ("+idx+")= " + str(os.path.getsize(base_dir + dir
+ final_stored_file_name)))
            db.updateDB('tb_filemanager_meta_file', 'file_size',
os.path.getsize(base_dir + dir + final_stored_file_name), 'idx='+idx)
            logger.debug("crea_dtm ("+idx+") = NOW()")
            db.updateDB('tb_filemanager_meta_file', 'crea_dtm', 'NOW()', 'idx='+idx)
```

```
            logger.debug("crea_dtm ("+idx+") = NOW()")
            return True
        except Exception as e:
            logger.debug("Error e=" + str(e))
            db.updateDB('tb_filemanager_meta_file', 'down_yn', 'E', 'idx='+idx)
            db.updateDB('tb_filemanager_meta_file', 'error',
'@updateAndRename:'+str(e), 'idx='+idx)
            return False



    def existDirectory(self,directory):
        logger.debug(">>" + directory)
        isDir = os.path.isdir(directory)
        if(isDir):
            logger.debug(">>> Exist !")
        else:
            logger.debug(">>> Make Dir !")
            os.makedirs(directory, exist_ok=True)
        return isDir



if __name__ == "__main__":
    db = CRUD()
    result = db.readDB('tb_archiver_sys_info','sys_storage', 'and
sys_title=\'ABPAΞAΣ\'')
    if( len(result) == 0 ):
        logger.debug(">> sys_storage result size 0 " )
    else:
        logger.debug(">> sys_storage result size " + str(len(result)) )
        base_dir = result[0][0]
        logger.debug(">> base_dir = " + base_dir)

    result = db.readDB('tb_filemanager_meta_file','idx, original_file_name, file_type,
file_title', 'and down_yn=\'N\'')
    resultYN = 'N'

    if( len(result) == 0 ):
        logger.debug(">> result size 0 " )
    else:
        logger.debug(">> result size " + str(len(result)) )
        row=result[0]
        logger.debug(">> Row detail " + str(row) )
        db.existDirectory(base_dir+dir)
        idx = str(row[0])
        downloadUrl = row[1]
        downloadFileType = row[2]
        file_title = row[3]
        db.updateDB('tb_filemanager_meta_file', 'down_yn', 'D', 'idx='+idx)
```

```
        db.commit()
        logger.debug(">>"+idx+" , "+downloadUrl+" , "+downloadFileType)
        if(downloadFileType == "mov"):
            logger.debug("> MOVIE !")
            if(db.downLoadMovieFileByURL(downloadUrl, idx, file_title)):
                if(db.updateAndRename(idx, downloadFileType)):
                    resultYN = 'Y'
                else:
                    resultYN = 'E'
            else:
                resultYN = 'E'
        elif(downloadFileType == "muc"):
            logger.debug("> MUSIC !")
            if(db.downLoadMusicFileByURL(downloadUrl, idx, file_title)):
                if(db.updateAndRename(idx, downloadFileType)):
                    resultYN = 'Y'
                else:
                    resultYN = 'E'
            else:
                resultYN = 'E'
        else:
            logger.debug("> ERROR FILE TYPE !")
            db.updateDB('tb_filemanager_meta_file', 'down_yn', 'E', 'idx='+idx)
            db.updateDB('tb_filemanager_meta_file', 'error', '@DownloadType',
'idx='+idx)
        db.updateDB('tb_filemanager_meta_file', 'down_yn', resultYN, 'idx='+idx)
        if(resultYN == ""):
            db.updateDB('tb_filemanager_meta_file', 'error', 'download done',
'idx='+idx)
        logger.debug(">> "+idx+" UPDATE=" + resultYN)
    db.commit()
```

Backup :실제 파일이름으로 변환해서 백업

/home/pi/pythonDownloadCrontab/KKBackupFiles.py 파일을 생성

```
from postgresqlDatabase import Databases
import yt_dlp
import datetime
import os
import re
import argparse, logging, logging.config, conf
from shutil import copyfile

today = datetime.date.today()
y = today.year
m = today.month
d = today.day
```

```
stored_file_name = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
dir = "/{year}/{month}/{day}/".format(year=y,month=m,day=d)

# Log
logging.config.dictConfig(conf.dictConfig)
logger = logging.getLogger(__name__)

class CRUD(Databases):

    def readDB(self,table,colum, condition):
        sql = " SELECT {colum} from {table} where 1=1
{condition}".format(colum=colum,table=table, condition=condition)
        try:
            self.cursor.execute(sql)
            result = self.cursor.fetchall()
        except Exception as e :
            result = (" read DB err",str(e))

        return result

    def updateDB(self,table,colum,value,condition):
        sql = " UPDATE {table} SET {colum}='{value}', sys_backup_date=NOW() WHERE 1=1
and {condition} ".format(table=table , colum=colum ,value=value,condition=condition )
        try :
            self.cursor.execute(sql)
        except Exception as e :
            logger.debug(" update DB err , e="+str(e))

    def backupFile(self,sys_storage, sys_backup_location):
       sql = " SELECT original_file_name, stored_file_name from
tb_filemanager_meta_file order by idx  "
       try:
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
       except Exception as e :
        result = (" read DB err",str(e))
       for row in result:
        toLocation = row[1].rindex("/", 0)
        #logger.debug(">> row = " + row[1][:toLocation+1] + " = " + str(toLocation))
        #logger.debug(">> row = " + sys_storage + row[1] + " --> TO --> " +
sys_backup_location + "/" + row[1][:toLocation]  + row[0])
        self.copyBackupFile(sys_storage + row[1], sys_backup_location + row[1]
[:toLocation] , row[0])

    def copyBackupFile(self, orgFile, destDir, destFile):
        if not os.path.exists(destDir + "/" + destFile):
            logger.debug(">> file not exist = " + destFile)
      # 디렉토리 만들기
            if not os.path.exists(destDir):
                logger.debug(">> Make Dir = " + destDir)
```

```
                os.makedirs(destDir)
      # 파일 옮기기
            logger.debug(">> Copy Files = " + orgFile + " --> " + destDir + "/" +
destFile )
            os.system('cp ' + orgFile + ' "' + destDir + "/" + destFile + '"')

        else:
            logger.debug(">> file exist ! -> " + destDir + "/" + destFile)



if __name__ == "__main__":
    db = CRUD()
    result = db.readDB('tb_archiver_sys_info','SYS_BACKUP_YN, sys_storage,
sys_backup_location', 'and SYS_TITLE=\'ABPAΞAΣ\' and SYS_BACKUP_YN=\'Y\'')

    if( len(result) == 0 ):
        logger.debug(">> Backup Skip " )
    else:
        logger.debug(">> result size " + str(len(result)) )
        row=result[0]
        logger.debug(">> Row detail " + str(row) )
        if( str(row[0]) == "Y" ):
            logger.debug(">> BACKUP START  ! ")
            db.updateDB('tb_archiver_sys_info', 'SYS_BACKUP_YN', 'P',
'SYS_TITLE=\'ABPAΞAΣ\'')
            db.commit()
            sys_storage = str(row[1])
            sys_backup_location = str(row[2])
            db.backupFile(sys_storage, sys_backup_location)
        db.updateDB('tb_archiver_sys_info', 'SYS_BACKUP_YN', 'N',
'SYS_TITLE=\'ABPAΞAΣ\'')
    db.commit()
```

Python 공통

/home/pi/pythonDownloadCrontab/conf.py 생성

```
from pathlib import Path

p = Path("logs")
if not p.exists():
    p.mkdir()

dictConfig = {
    'version': 1,
    'disable_existing_loggers': True,
    'formatters': {
        'standard': {
```

```
            'format': '%(asctime)s [%(levelname)s] %(name)s:: %(message)s',
        },
    },
    'handlers': {
        'default': {
            'level': 'DEBUG',
            'formatter': 'standard',
            'class': 'logging.StreamHandler',
            'stream': 'ext://sys.stdout',
        },
        'file': {
            'class': 'logging.handlers.RotatingFileHandler',
            'level': 'DEBUG',
            'formatter': 'standard',
            'filename': 'logs/logfile.log',
            'mode': 'a',
            'maxBytes': 5_242_880,
            'backupCount': 3,
            'encoding': 'utf-8',
        },
    },
    'loggers': {
        '__main__': {
            'handlers': ['default','file'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'camera': {
            'handlers': ['default', 'file'],
            'level': 'DEBUG',
            'propagate': False,
        },
    }
}
```

/home/pi/pythonDownloadCrontab/postgresqlDatabase.py 생성

```
import psycopg2

class Databases():
    def __init__(self):
        self.db = psycopg2.connect(host='localhost',
dbname='archiver',user='pi',password='archiverpw123!@#',port=5432)
        self.cursor = self.db.cursor()

    def __del__(self):
        self.db.close()
        self.cursor.close()

    def execute(self,query,args={}):
```

```
        self.cursor.execute(query,args)
        row = self.cursor.fetchall()
        return row

    def commit(self):
        self.db.commit()
```

Crontab 설정

```
*/10 * * * * /usr/bin/python /home/pi/pythonDownloadCrontab/KKYouTuBeDownloader.py > /
home/pi/logs/sysout.log
*/13 * * * * /usr/bin/python /home/pi/pythonDownloadCrontab/KKBackupFiles.py > /home/
pi/logs/backup.log
```

10 분마다 Youtube 다운로드 찾아 다운로드 실행

13분마다 백업 여부를 확인하고 백업이 있을 때 배치로 수행

Backup shell

/home/pi/backupMetaFilesToRemoteDitectory.sh /meta/data /meta/sshf > /home/pi/backupMetaFilesToRemoteDitectory.log

```
        #!/bin/sh

        # ssh-keygen -t rsa
        # ssh-copy-id -i ~/.ssh/id_rsa.pub xxx@xxx.xxx.xxx.xxx
        #오류 -->
        #/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
xxx/.ssh/id_rsa.pub"
        #/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are
already
           installed
        #
        #/usr/bin/ssh-copy-id: ERROR:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        #/usr/bin/ssh-copy-id: ERROR:
#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
        #ERROR: @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
        #ERROR: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
        #ERROR: IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
        # ...
        # ===> ssh-keygen -f "/home/xxx/.ssh/known_hosts" -R "xxx.xxx.xxx.xxx"
```

```
        export _SDATE=$(date +%Y%m%d%H%M%S)
        OrgDirectory=$1
        BackupDirectory=$2
        DirectoryFound=$(find $OrgDirectory -type d )
        countfiles=0

        MountRemoteDirectory()
        {
                if [ $(mount -l | grep /meta/sshf | wc -l) -eq 1 ];then
                        echo "   >> EXIST"
                else
                        echo "   >> Mount Remote Directory"
                        /usr/bin/sshfs -o allow_other xxx@xxx.xxx.xxx.xxx:/xxx/ext/
xxx/002-xxx /xxx/xxx
                fi
        }

        MakeList()
        {
                echo "   >> Make List !"
                find $OrgDirectory -type f > $BackupDirectory/backFileList.list
        }

        checkDirectoryExist()
        {
                dirname=`dirname "$1"`
                        if [ -d "$BackupDirectory/$dirname" ]; then
                                        printf "-"
                        else
                                        echo -e "\n   >> NOT EXIST --> Makedirectory "
$BackupDirectory/$dirname
                                    mkdir -p "$BackupDirectory/$dirname"
                        fi
        }

        checkFileExist()
        {
                Orgfilename="$1"
                filename=`echo "$1" | cut -d'/' -f4-`
                        #echo ">> " $BackupDirectory/$filename
                        if [ -f "$BackupDirectory/$filename" ]; then
                                        printf "."
                        else
                                        echo -e "\n   >> NOT EXIST --> Copy "
$Orgfilename " --> " $BackupDirectory"/"$filename
                                        cp "$Orgfilename" "$BackupDirectory/$filename"
                        fi
        }
```

```
        exportMetaDB()
        {
                echo -e "\nSTART EXPORT META DB "$_SDATE " --> " $BackupDirectory "/
META-" $_SDATE ".dump"
                pg_dump -p 5432 archiver > $BackupDirectory/"METADATA-"$_SDATE".dump"
        }

        deleteOldExportDB()
        {
                FinalDbExportDumpCnt=5
                FinalDbExportDumpDueDay=60
                NOW=`date +%Y%m%d%H%M%S`
                FinalBackupDay=`date +%Y%m%d%H%M%S --date="$FinalDbExportDumpDueDay
days ago"`
                echo -e "\nSTART EXPORT META DB DELETE !! "$NOW","$BackupDirectory
                echo "   >> TODAYTIME : $NOW"
                echo "   >> FinalDbExportDumpDueDay : $FinalDbExportDumpDueDay"
                echo "   >> FinalBackupDay : $FinalBackupDay"
                echo "   >> ## Old DB export dump delete ! ##"
                TotalExportCnt=`find $BackupDirectory -name "*.dump" | wc -l`
                echo '   >> Total backup count = ' $TotalExportCnt
                echo '   >> Final backup remain count = ' $FinalDbExportDumpCnt
                TargetdeleteExportCnt=`expr $TotalExportCnt - $FinalDbExportDumpCnt`
                echo '   >> Total backup count - Final backup count ='
$TargetdeleteExportCnt

                BackupAllFiles=$(find $BackupDirectory -name "*.dump" | sort -n)
                cnt=1
                echo '   >> ' $FinalBackupDay ' day before delete target backup list'
                for i in $BackupAllFiles
                do
                  backupfileDate=`echo $i | awk -F "-" '{print $2}' | awk -F "."
'{print $1}'`
                  if [ $backupfileDate -le $FinalBackupDay ]
                  then
                        echo "    >> $backupfileDate -le $FinalBackupDay = Check Total
count $TargetdeleteExportCnt -ge $cnt"
                        if [ $TargetdeleteExportCnt -ge $cnt ]
                        then
                                echo "     >>> Delete backup $i"
                                rm $i
                        else
                                echo "     >>> skip backup : " $i
                        fi
                  else
                        echo "    >> Not yet Delete day : " $i
                  fi
                done
        }
```

```
        echo -e "\n#########################################"
        echo "START "$_SDATE
        if [ $# -ne 2 ];then
                        echo "ERROR 1.target folder, 2. backup folder!"
                        echo " ex : /home/xxx/backupMetaFilesToRemoteDitectory.sh
<<OrgDirectory>>  <<BackupDirectory>>"
                        exit 1
        fi

        # 목록 파일을 만든다.
        MountRemoteDirectory
        MakeList

        LINE_NO=$(cat $BackupDirectory/backFileList.list | wc -l) # 몇 줄 라인인지 읽음
        echo " Total source file count : " $LINE_NO " line"
        while read line
        do
                dirname=`echo $line | cut -d'/' -f4-`
                checkDirectoryExist "${dirname}"
                checkFileExist "${line}"
                countfiles=$(($countfiles+1))
        done < $BackupDirectory/backFileList.list

        echo -e "\n Total check file count : " $countfiles " line"
        exportMetaDB
        deleteOldExportDB

        _EDATE=$(date +%Y%m%d%H%M%S)
        Taketime=`expr $_EDATE - $_SDATE`
        echo -e "\n\n ### FINISH take="$Taketime "###"
        echo "#########################################"
        exit 0
```

실행

1. http://localhost:8080/archiver/login.do

Revision #25
Created 2023-06-07 10:38:53 UTC by Hyeon Su Ryu
Updated 2025-11-23 12:00:40 UTC by Hyeon Su Ryu