ABPAΞAΣ Container 설치

ABPAΞAΣ Container version

1. PosgreSql Pod 설치

1.1 PV 구성

각 VM에 공유 볼륨을 생명하고 해당 볼륨을 PV(Persistent Volumes)로 설정
각 VM에 NFS로 연계하는 부분은 여기를 참조

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: postgresql-pv-volume
  namespace: tomcat-apps
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/tomcat/postgresql"

---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgresql-pv-claim
  namespace: tomcat-apps
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

1.2 PostgreSql

postgresql pod를 만든다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-postgresql
  namespace: tomcat-apps
spec:
  replicas: 1
  selector:
    matchLabels:
      app: products
      department: tomcat-postgresql
  template:
    metadata:
      labels:
        app: products
        department: tomcat-postgresql
    spec:
      containers:
        - name: postgresql
          image: postgres:11
          ports:
            - containerPort: 5432
          volumeMounts:
            - mountPath: "/var/lib/postgresql/data"
              name: postgresql-persistent-storage
              readOnly: false
          env:
            - name: POSTGRES_PASSWORD
              value: eXXXXXXXrl  <-- root 패스워드 기입
      volumes:
        - name: postgresql-persistent-storage
          persistentVolumeClaim:
            claimName: postgresql-pv-claim

---

apiVersion: v1
kind: Service
metadata:
  name: tomcat-postgresql-service
  namespace: tomcat-apps
spec:
  type: LoadBalancer
  externalIPs:
   - 192.168.0.100
  selector:
```

```
    app: products
    department: tomcat-postgresql
  ports:
  - protocol: TCP
    port: 5432
    targetPort: 5432

---
```

http://web.joang.com:6875/books/abraksas-system/page/abraksas#bkmrk-%C2%A0 데이터베이스 생성

http://web.joang.com:6875/books/abraksas-system/page/abraksas#bkmrk---table-%EC%83%9D%EC%84%B1%C2%A0 테이블 생성

2. ΑΒΡΑΞΑΣ Dockerfile

```
FROM tomcat:9

ENV TZ=Asia/Seoul
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
COPY ./favorit.ico /usr/local/tomcat/webapps/ROOT/favorit.ico
VOLUME ["/data/tomcat", "/data/tomcat"]
COPY ./com.joang.web.filemetamanager/target/meta.war /usr/local/tomcat/webapps/
meta.war
COPY ./com.joang.web.filemetamanager/target/meta.war /usr/local/tomcat/webapps/ROOT/
meta.war
RUN mkdir -p /usr/local/tomcat/filebeat-8.6.0-linux-x86_64
COPY ./filebeat-8.6.0-linux-x86_64 /usr/local/tomcat/filebeat-8.6.0-linux-x86_64
RUN chmod go-w /usr/local/tomcat/filebeat-8.6.0-linux-x86_64/filebeat.yml
COPY ./redisson-all-3.7.5.jar /usr/local/tomcat/lib
COPY ./redisson-tomcat8.jar /usr/local/tomcat/lib
RUN rm -Rf /usr/local/tomcat/conf/context.xml
COPY ./context.xml /usr/local/tomcat/conf
RUN rm -Rf /usr/local/tomcat/conf/server.xml
COPY ./server.xml /usr/local/tomcat/conf
COPY ./redisson.conf /usr/local/tomcat/conf
COPY ./redis-data-cache.properties /usr/local/tomcat/conf
COPY ./tomcat-cluster-redis-session-manager/lib /usr/local/tomcat/lib
ENTRYPOINT ["filebeat-8.6.0-linux-x86_64/startfilebeat.sh"]
```

전체 docker 빌드 환경 파일 :

filebeat-8.6.0-linux-x86_64.tar

tomcat-cluster-redis-session-manager.tar

context.xml

redis-data-cache.properties

redisson.conf

redisson-all-3.7.5.jar

redisson-tomcat8.jar

server.xml

3. Docker Build

```
#!/bin/bash

VERSION=$1

echo ">> Build Version : " $VERSION

docker build -t tomcat-meta:$VERSION /data/tomcat/scm/dockerimage/meta

docker tag tomcat-meta:$VERSION xxx.xxxx.com:8888/tomcat-meta:$VERSION
docker tag tomcat-meta:$VERSION xxx.xxxx.com:8888/tomcat-meta:latest

docker push xxx.xxxx.com:8888/tomcat-meta:$VERSION
docker push xxx.xxxx.com:8888/tomcat-meta:latest

exit 0
```

4. Namespace 만들기

ABPAΞAΣ용 namespace를 정의합니다.

```
apiVersion: v1
kind: Namespace
metadata:
  name: tomcat-apps
  labels:
    app.kubernetes.io/name: tomcat-apps
    app.kubernetes.io/part-of: tomcat-apps
```

5. ΑΒΡΑΞΑΣ Yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-meta
  namespace: tomcat-apps
spec:
  selector:
    matchLabels:
      app: products
      department: tomcat-meta
  replicas: 2
  minReadySeconds: 20
  template:
    metadata:
      labels:
        app: products
        department: tomcat-meta
    spec:
#      hostNetwork: true
      containers:
      - name: tomcat-meta
        image: xxx.xxxx.com:8888/tomcat-meta:latest
        imagePullPolicy: Always
        volumeMounts:
        - name: tomcat-volume
          mountPath: /data/tomcat
        env:
        - name: "PORT"
          value: "8080"
      volumes:
      - name: tomcat-volume
        hostPath:
          path: /data/tomcat
          type: Directory

---

apiVersion: v1
kind: Service
metadata:
  name: tomcat-meta-service
  namespace: tomcat-apps
spec:
  type: LoadBalancer
  externalIPs:
   - 192.168.0.100
  selector:
```

```
    app: products
    department: tomcat-meta
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
  sessionAffinity: ClientIP

---

apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: meta-ingress
  namespace: tomcat-apps
  annotations:
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/proxy-body-size: 5000m
    nginx.ingress.kubernetes.io/session-cookie-hash: sha1
    nginx.ingress.kubernetes.io/session-cookie-name: route
    nginx.ingress.kubernetes.io/use-regex: 'true'
spec:
  rules:
  - host: web.joang.com
    http:
      paths:
      - path: /meta/*
        backend:
          serviceName: tomcat-meta-service
          servicePort: 8080
      - path: /*
        backend:
          serviceName: tomcat-meta-service
          servicePort: 8080


---
```

6. Kubernetes ΑΒΡΑΞΑΣ Pod

```
kubectl apply -f /data/tomcat/scm/tomcat-meta.yaml
```

```
docker image prune -f
```

```
kubectl -n tomcat-apps rollout restart deployment tomcat-meta
```

7. 배치 (Python batch)

백업 프로그램

배치 startBackUpBatch.sh

```
#!/bin/bash

python /apps/KKBackupFiles.py > /apps/backup.log



exit 0
```

```
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")
base_dir = "/media/pi/RaspRefp/data"
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()
```

Youtube 다운로드 프로그램

```
#!/bin/bash

python /apps/KKYouTuBeDownloader.py > /apps/youtube.log

exit 0
```

```
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):
        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 + '%(title)s.%(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(">> 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):
        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 + '%(title)s.%(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', '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]
        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)):
                if(db.updateAndRename(idx, downloadFileType)):
                    resultYN = 'Y'
                else:
                    resultYN = 'E'
            else:
                resultYN = 'E'
        elif(downloadFileType == "muc"):
            logger.debug("> MUSIC !")
            if(db.downLoadMusicFileByURL(downloadUrl, idx)):
                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()
```

공통

```
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,
        },
    }
}
```

```
import psycopg2

class Databases():
    def __init__(self):
        self.db = psycopg2.connect(host='xxx.xxxx.com',
dbname='archiver',user='pi',password='password',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()
```

8. 배치 (Kubernetes Job)

```
FROM python:3.9-slim

ENV PYTHONUNBUFFERED=1
ENV TZ=Asia/Seoul
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
RUN mkdir -p /apps
VOLUME ["/data/tomcat", "/data/tomcat"]
RUN mkdir -p /apps/filebeat-8.6.0-linux-x86_64
COPY ./filebeat-8.6.0-linux-x86_64 /apps/filebeat-8.6.0-linux-x86_64
RUN chmod go-w /apps/filebeat-8.6.0-linux-x86_64/filebeat.yml
RUN apt update
RUN apt install ffmpeg -y
RUN pip install --upgrade pip
RUN pip install psycopg2-binary
RUN pip install yt_dlp
RUN pip install ffmpeg-python
COPY ./python-programes /apps
ENTRYPOINT ["/apps/filebeat-8.6.0-linux-x86_64/startfilebeat.sh"]
```

```
#!/bin/bash

VERSION=$1

echo ">> Build Version : " $VERSION

docker build -t tomcat-meta-batch:$VERSION /data/tomcat/scm/dockerimage/meta-batch

docker tag tomcat-meta-batch:$VERSION xxx.xxxx.com:8888/tomcat-meta-batch:$VERSION
docker tag tomcat-meta-batch:$VERSION xxx.xxxx.com:8888/tomcat-meta-batch:latest

docker push xxx.xxxx.com:8888/tomcat-meta-batch:$VERSION
docker push xxx.xxxx.com:8888/tomcat-meta-batch:latest

exit 0
```

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: tomcat-meta-batch
  namespace: tomcat-apps
spec:
  schedule: "*/15 * * * *"
  concurrencyPolicy: Forbid
  jobTemplate:
    spec:
      template:
```

```
        spec:
          containers:
          - name: tomcat-meta-batch
            image: web.joang.com:8888/tomcat-meta-batch:latest
            command:
            - /bin/sh
            - -c
            - /apps/startBackUpBatch.sh > /apps/batch.log
            volumeMounts:
            - name: tomcat-volume
              mountPath: /data/tomcat
          volumes:
          - name: tomcat-volume
            hostPath:
              path: /data/tomcat
              type: Directory
          restartPolicy: OnFailure
  successfulJobsHistoryLimit: 3
  failedJobsHistoryLimit: 1
```

사용법 : ABPAΞAΣ 사용하기

Revision #21
★Created 2023-06-07 09:20:08 UTC by Hyeon Su Ryu
Updated 2023-06-13 07:56:12 UTC by Hyeon Su Ryu