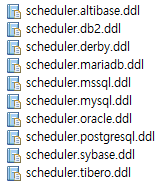
1. **배치스케줄러 소개**

작업 자동화와 모니터링을 위한 통합 시스템이며, Agent는 작업 실행과 스케줄 관리, Server는 UI 및 API를 통한 작업 상태 모니터링을 지원한다.   
다양한 데이터베이스와 유연한 설정을 통해 확장성과 안정성을 제공하며, 복잡한 배치 환경에서도 효율적이고 신뢰성 높은 운영이 가능하다.

1. **배치스케줄러 프로젝트 구조**
2. 실행환경

* JDK 버전 : 권장사항은 jdk 1.6이며, 아래 설명은 jdk 23으로 진행하였음.
* 데이터 베이스

지원하는 데이터베이스는 다음과 같습니다. 초기 DB 구축에 대한 DDL을 참조하여, 아래 DB는 지원하는 것으로 간주함.



1. 구성요소

* nexcore.scheduler.agent: 작업 실행 엔진으로, 배치 작업을 스케줄에 따라 실행합니다. 해당 모듈은 server에서 lib로 참조하게 된다.
* nexcore.scheduler.server: UI 및 API 관리 서버로, 작업 상태를 확인하고 모니터링합니다.
* nexcore.scheduler.common: 공통 모듈로, 데이터베이스와의 상호작용 및 작업 정의를 지원합니다. 해당 모듈은 server, agent에서 lib로 참조하게 된다.

1. 배치스케줄러 주 프로세스

* Job 등록

1. 관리자는 Server의 UI(JSP 기반) 또는 REST API를 통해 작업을 등록합니다.

JSP 파일: src/web/jobmon/view\_jobdef.jsp (작업 정의 화면)

Controller 클래스: ControllerServiceImpl.java

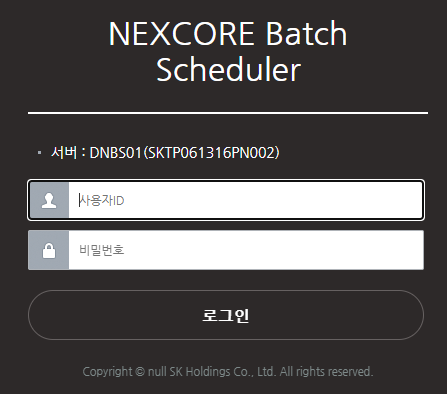
1. 작업 정의는 데이터베이스에 저장되며, SQL 매핑 파일에서 정의된 작업 정보(.xsql)로 관리됩니다.

* 배치스케줄러 작업 스케줄링 (Quartz 사용)

1. 작업이 등록되면, JobDefinitionManager 클래스가 데이터베이스에서 작업 스케줄 정보를 가져옵니다.
2. 작업 실행 시점(nextrundate)과 반복 주기(repeat\_interval)을 기반으로 작업 대기열에 추가됩니다.

* 작업실행

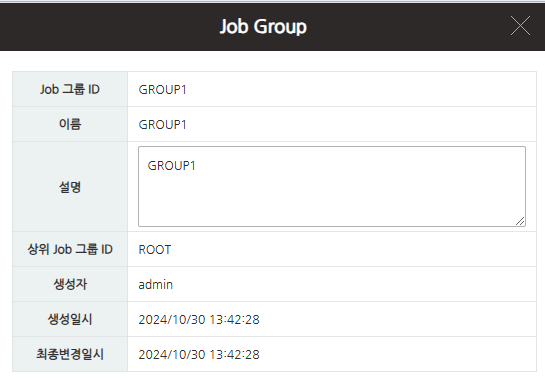
1. 작업실행은 실제 agent에서 수행하며 nextrundate를 참조하여 job이 실행되는 시점이 되었을 경우 수행한다.
2. **Batch 스케줄러 사용법 (실제 JOB 등록 및 테스트)**
3. 로그인 : 초기 비밀번호는 admin / nexcore 이다. (user 테이블에 초기 INSERT하는 쿼리를 수행해줘야한다. – 소스 설명에서 참조)



1. agent가 이미 서버에 설치가 되어있다는 가정하에 서버를 등록(메뉴 : 서버) 하여 주고, 서버와 통신이 원할하게 되고 있는지 확인한다. (녹색 : 통신중)



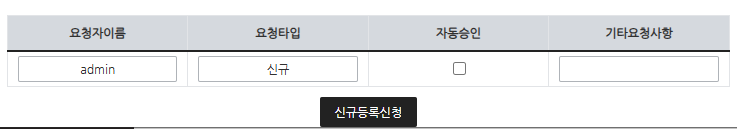
1. Job은 그룹단위로 관리가 되고 있으며, 상위개념인 group을 먼저 만들어 준다. (메뉴 : Job 그룹)



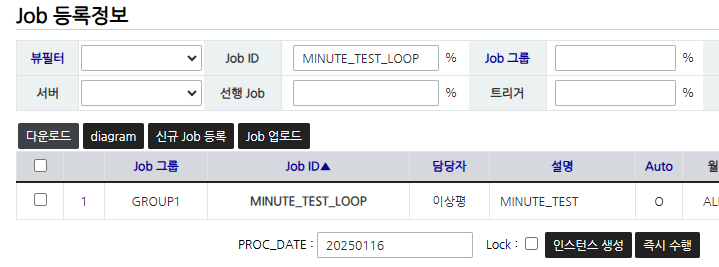
1. Job 등록 정보에서 Job을 등록하여 준다.



* 해당 배치는 실행을 sleep 10을 실행시키는 배치이며, 00:00-23:59가 유효시간이고 반복주기가 10초 이므로, 매일 00:00부터 10초마다 sleep 명령어를 실행시키게 된다.

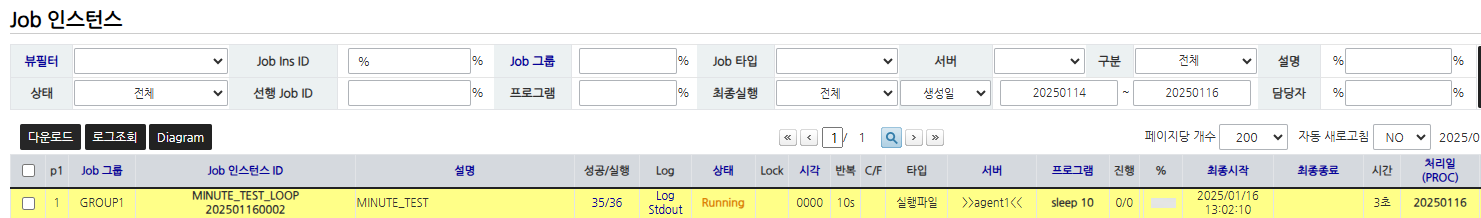


* 등록시에 ‘자동승인’을 체크하지 않으면, 별도 관리자가 승인 메뉴(Job등록요청현황 메뉴)에서 승인 해줘야 등록이 완료된다. ‘자동승인’을 체크하면 승인 필요없이 바로 job등록이 완료된다.

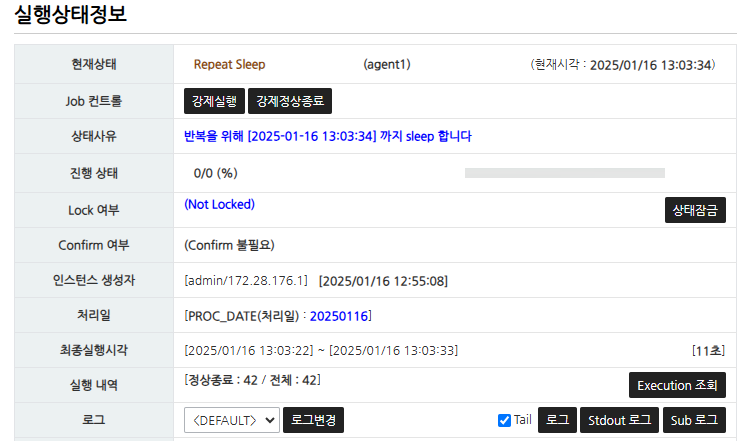


* 실제 agent에서 구동할 수 있게 하려면 job을 통해서 ‘인스턴스’를 생성하여야 한다. 그러므로 해당 job을 체크하고 ‘인스턴스 생성’을 하여 준다. (Lock : Lock이 선택되면 인스턴스가 생성은 되지만 실행은 안되고, Lock을 해제하게 되면 인스턴스 생성과 동시에 바로 실행이 되어버린다.)

1. Job 인스턴스 메뉴에서 등록한 job이 잘 동작하는지 모니터링할 수 있다.



(인스턴스 전체 현황 모니터링)



(인스턴스 세부항목 별로 모니터링을 할 수 있다)

1. **Nexcore Batch Scheduler Source 소개**
2. Batch-scheduler : 스케줄러의 백엔드와 UI 부분를 모두 관리하며, 해당 폴더는 아래와 같습니다.

UI는 자체 Jetty Server를 사용하며 6버전대를 사용하며 해당 버전은 java 1.6 버전에java를 상위버종속되어 있어서 상위 사용하기 위해서는 1.6버전에 종속되어 있어서 업데이트 부분이 필요하며, 현재 1.8 버전에서 compile은 가능한 수준이며, WEB은 compile이 되지 않아 향후 조정 필요.

비즈니스 로직 부분은 아래와 같으며, 패키지간에 상위 패키지를 lib로 사용 중이며, 주 소스 server, agent는 아래와 같이 lib를 참조하여 사용한다.

nexcore-scheduler-server : nexcore-scheduler-common, nexcore-scheduler-agent 참조

nexcore-scheduler-agent : nexcore-scheduler-common 참조

1. 주요 설정 정보

* nexcore-scheduler-server.properties

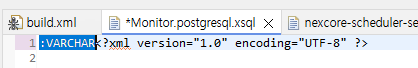
scheduler.db.vendor : 사용할 DB를 선택한다.

scheduler.jdbc.\* : DB 접속 정보를 입력한다.

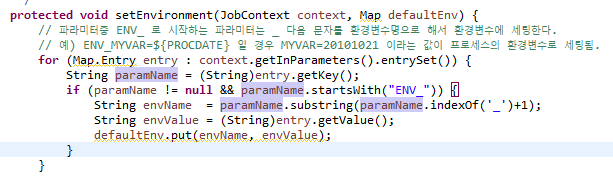
1. **Nexcore Batch Scheduler 분석 및 수정 작업 (진행내용)**
2. 기존 소스 내에 오타 수정 – 관리가 안된 소스로 간주하여, 일부 소스간의 오타 및 자료형 타입을 잘못 쓰고 있음

* 수정 내용

1. Monitor.postgresql.xsql 내 상단 VARCHAR 구문 삭제



1. Nexcore-scheduler-server-core.xml.notouch 파일내 Jetty 속성 ‘httpsEnable’ 삭제 (다) 부분과 연관이 있으며, 임시방편으로 사용 안하는 것으로 처리
2. JettyStarter.java 내 httpsEnable = false 로 수정
3. Batch-agent 내에 null체크 추가 (paramName이 null로 오는 경우 agent 오류 발생)



1. 라이선스 체크 삭제

: 현재 아래 체크 로직을 삭제한 상태로 서비스 이상없음을 확인하였음.



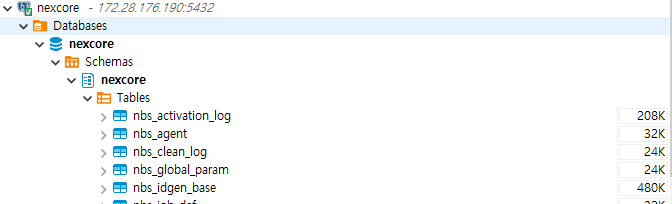
1. Build.xml 파일 작성

(nexcore.scheduler.server, nexcore.scheduler.agent, nexcore.scheduler.common 모두 해당)

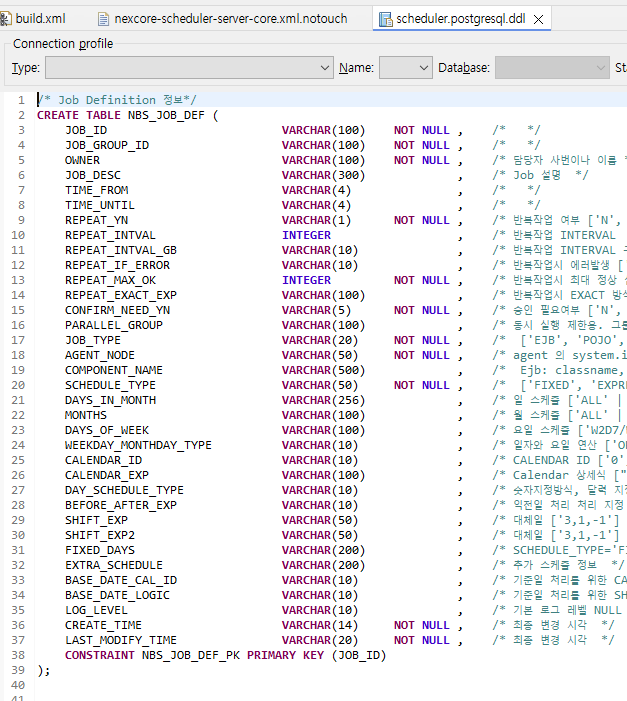
: build 시에 /dist 경로에 아래 정보로 jar 파일이 생성이 될 수 있도록 작성.

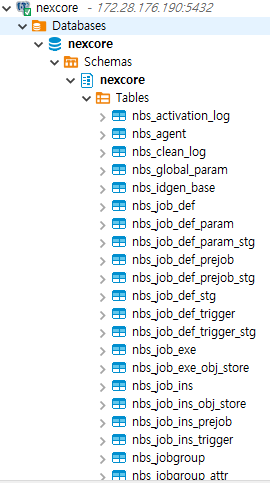


1. DB 설치 및 스키마 생성
2. postgresql을 설치를 로컬에서 진행하고, database, schema, user는 모두 nexcore로 하였습니다.



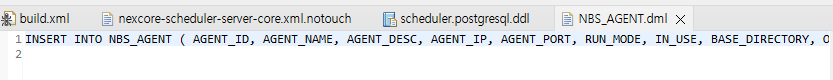
1. /script/dbsql/ddl/\* 경로에 있는 ddl 파일을 DB 종류에 맞게 수행시켜 기본 TABLE 생성





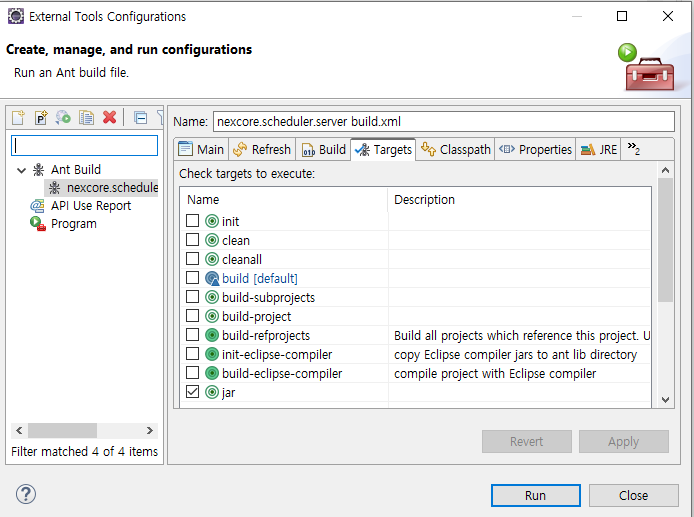
(생성완료 확인)

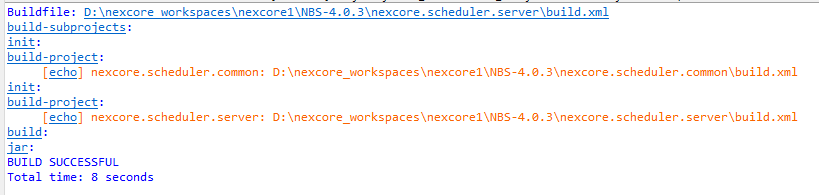
1. /script/dbsql/dml/NBS\_USER.dml 의 쿼리도 수행시켜, 관리자도 생성햐여 준다. 관리자는 별도 추가하는 기능이 없으므로 필수이다. (화면상에 관리자 생성 부분은 없다.)



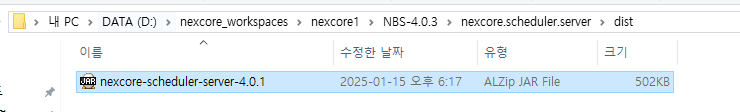
1. **batch-scheduler-server 프로젝트 빌드 및 배포**
2. 만들어 놓은 build.xml 파일을 ant로 수행시켜 준다.

수행 시킬 때 targets을 설정하여 jar 파일이 떨어지도록 수행한다.





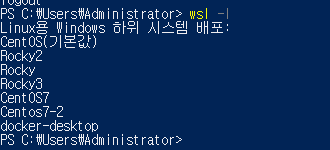
1. /dist 폴더에 프로젝트 jar 파일이 생성되었는지 확인한다.



1. 배포할 서버를 만들어야 하므로, 서버가 없으므로 wsl –d 를 통해서 Centos 7버전대의 linux를 미리 생성합니다. (wsl 관련은 구글 검색을 활용)

Jar 파일 이외에 필요한 파일들은 압축 파일 중의 reference경로로 올려 두었습니다. 해당 경로는 실제 NEXCORE가 실행되었던 폴더를 가져 온 것이니, 해당 경로의 파일들을 복제하고 lib안에 ‘nexcore-scheduler-server.jar, nexcore-scheduler-agent, nexcore-scheduler-common을 갈아 끼워서 테스트 합니다.

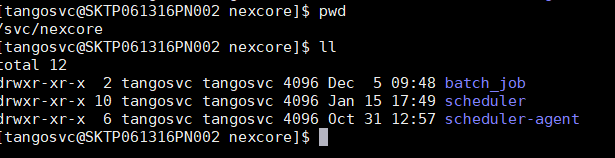
* wsl –l 로 Centos7 으로 만들어 놓은 가상 OS 확인



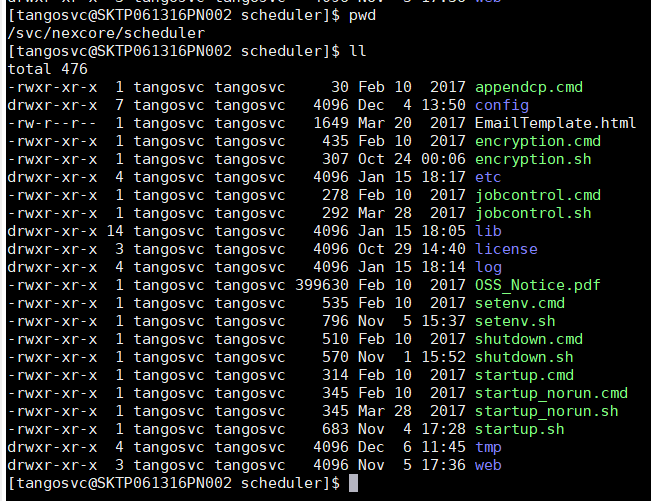
* 해당 Centos7 OS에 접속

: 사전에 wsl –d 로 접속하여, sshd를 활성화 하여주고, 일반계정(tangosvc)을 만들어 둠.

: ssh를 통해 서버에 접속해서 아래 디렉토리를 만들고 배치스케줄러 구조 복제(reference 폴더 참조)



1. Scheduler : 배치스케줄러 서버(nexcore-batch-scheduler.jar)가 실행되는 환경

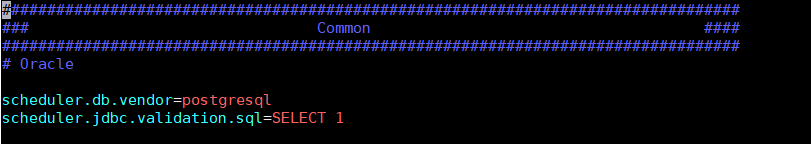


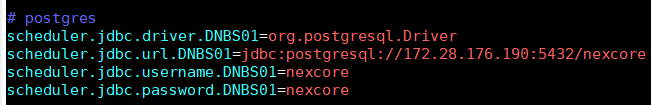
startup.sh : batch-scheduler-server를 수행시킨다.

shutdown.sh : batch-scheduler-server를 중지한다.

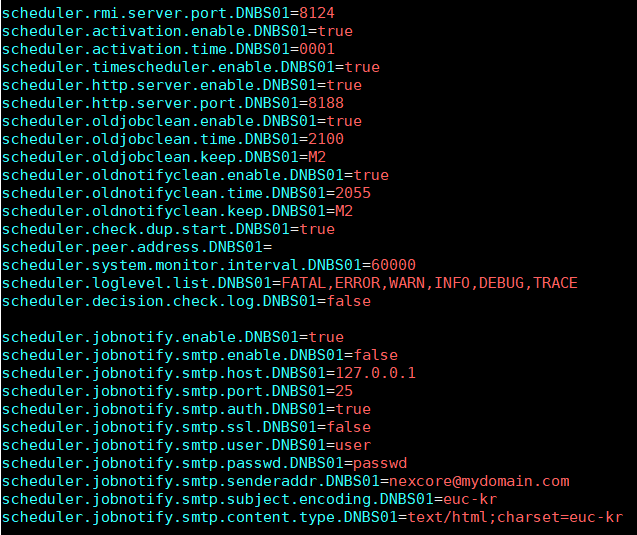
config 폴더 : scheduler-server에 대한 주 properties를 관리한다.

* config/properties/nexcore-scheduler-agent.properties (sample)

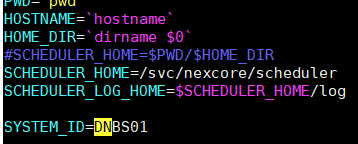


(사용할 DB에 대한 vendor를 설정하며, vendor에 대한 명칭은 소스에서 별도 분기 처리하여 정확히 기입해야 한다.)  
 

(설치한 DB 계정정보를 입력한다.)

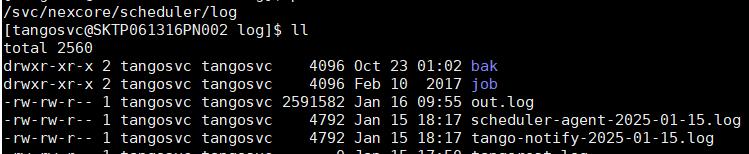


(DNBS01 은 시스템 명칭이며, 해당 명칭은 setenv.sh에서 별도 수정이 가능하다. 그러므로 변경이 되면 properties내 명칭도 변경해주어야 합니다.)



(setenv.sh)

log 폴더 : batch-schdeuler-server 동작 LOG를 관리한다.

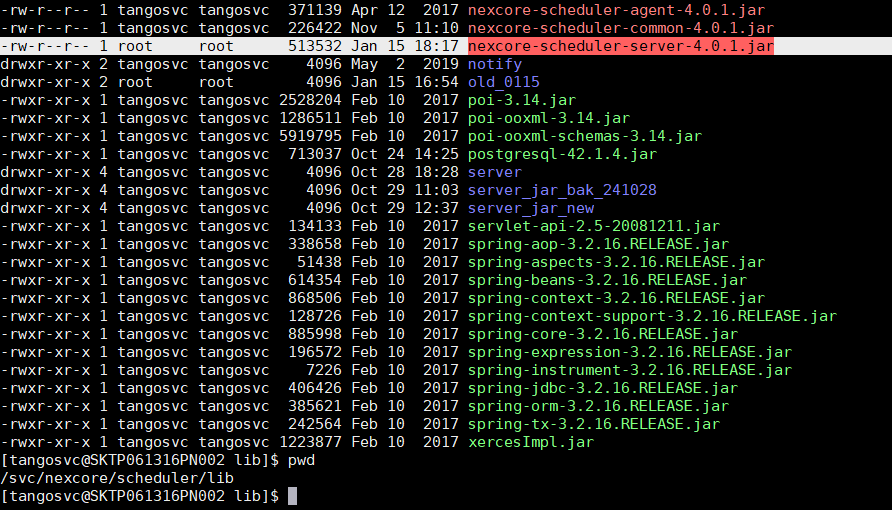


web 폴더 : 배치스케줄러 UI 관련 jsp파일을 관리한다.

tmp 폴더 : web 폴더내 jsp 컴파일된 파일을 관리한다.

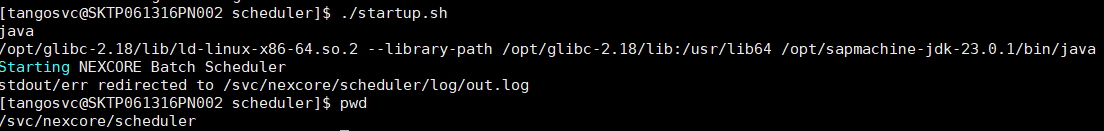
1. 배포 테스트를 위해 1)에서 생성해 둔 nexcore-batch-scheduler.jar 파일을 아래 경로에 배치한다.

* /svc/nexcore/scheduler/lib 경로에 신규 생성한 nexcore-scheduler-server-4.0.1.jar를 배치한다.

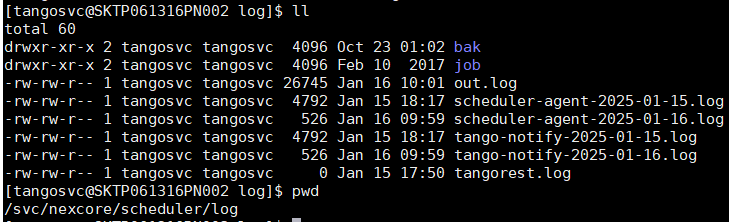


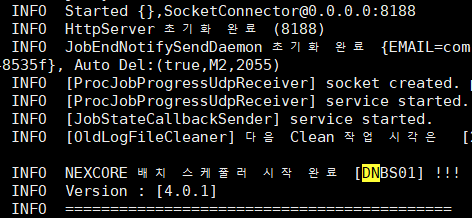
* /svc/nexcore/scheduler 경로에서 startup.sh를 수행한다.

(java 버전은 상위버전(jdk-23)으로 서버에 설치하였습니다.)

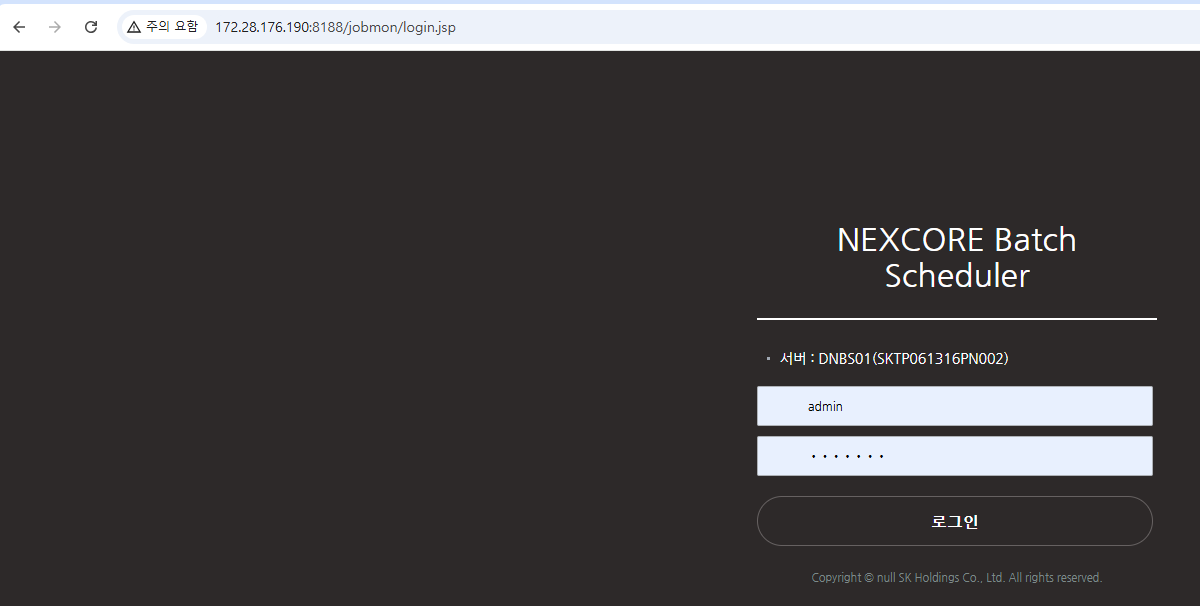


* Nexcore-batch-scheduler-server start 로그 확인 (agent가 아직 start 전이므로 agent와 통신할 수 없다는 오류가 발생할 수 있음. 현재는 agent가 활성화 중이므로 별도 에러는 안나옴)

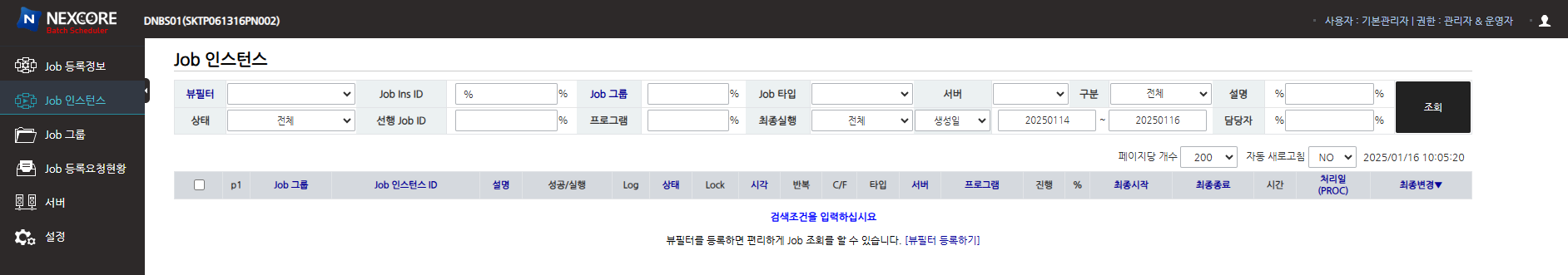




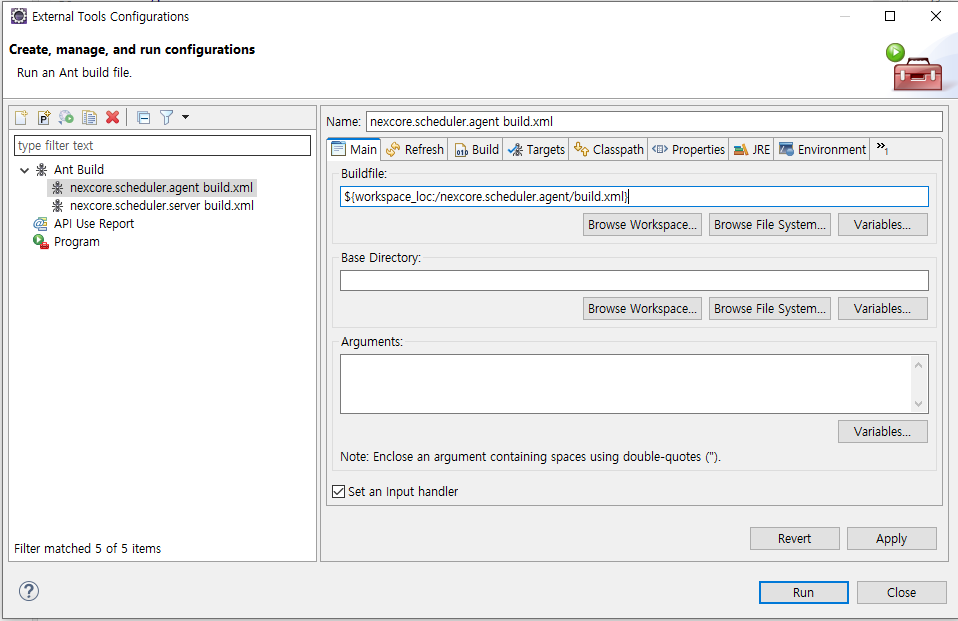
1. UI에서 LOGIN 페이지가 뜨는지 확인 (admin / nexcore)



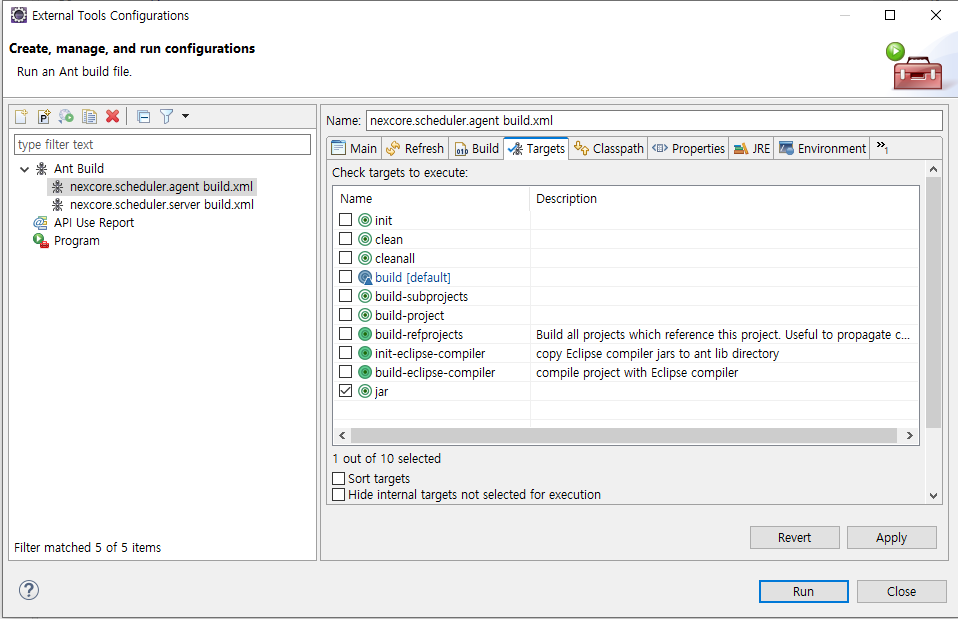
1. LOGIN 하여, 메인화면이 정상적으로 나오는지 확인한다.

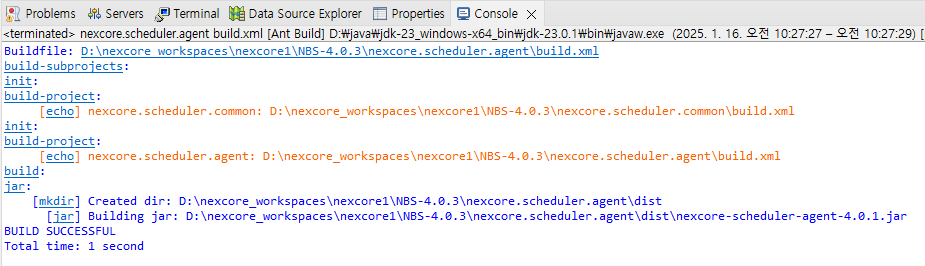


1. batch-scheduler-agent 프로젝트 빌드 및 배포
2. 만들어 놓은 build.xml 파일을 ant로 수행시켜 준다.

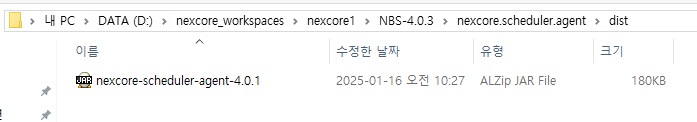


수행 시킬 때 targets을 설정하여 jar 파일이 떨어지도록 수행한다.





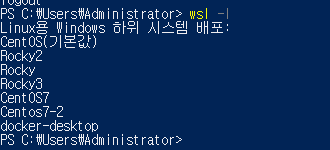
1. /dist 폴더에 프로젝트 jar 파일이 생성되었는지 확인한다.



1. 이전 단계에서 생성한 CentosOS7에 접속하여, 작업을 진행합니다.

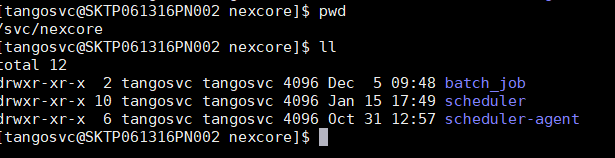
Jar 파일 이외에 필요한 파일들은 압축 파일 중의 reference경로로 올려 두었습니다. 해당 경로는 실제 NEXCORE가 실행되었던 폴더를 가져 온 것이니, 해당 경로의 파일들을 복제하고 lib안에 ‘nexcore-scheduler-agent’ 을 갈아 끼워서 테스트 합니다.

* wsl –l 로 Centos7 으로 만들어 놓은 가상 OS 확인



* 해당 Centos7 OS에 접속

: ssh를 통해 서버에 접속해서 아래 디렉토리를 만들고 배치스케줄러 구조 복제이(이미 scheduler-agent를 복제하였으므로, 참조만 하며, 별도의 에이전트서버를 생성하여 구현할 때는 scheduler-agent만 복제해야함.)(reference 폴더 참조)



1. scheduler-agent : 배치스케줄러 에이전트(nexcore-batch-agent.jar)가 실행되는 환경

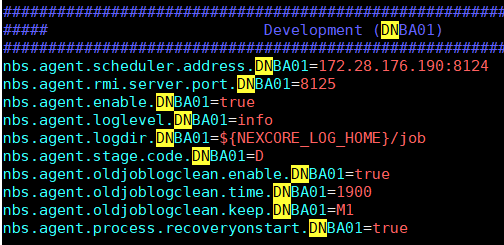


startup.sh : batch-scheduler-agent를 수행시킨다.

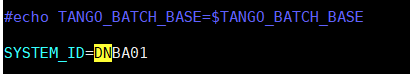
shutdown.sh : batch-scheduler-agent를 중지한다.

config 폴더 : scheduler-agent에 대한 주 properties를 관리한다.

* config/properties/nexcore-scheduler-agent.properties (sample)

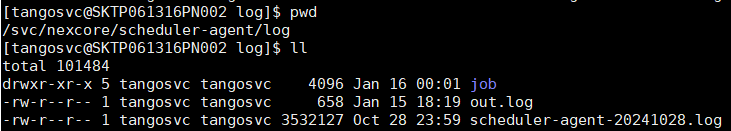


(IP 정도만 수정하여주고, DNBA01 도 setenv.sh에서 관리되기 때문에 변경 시에 해당 파일도 변경하여 준다.)



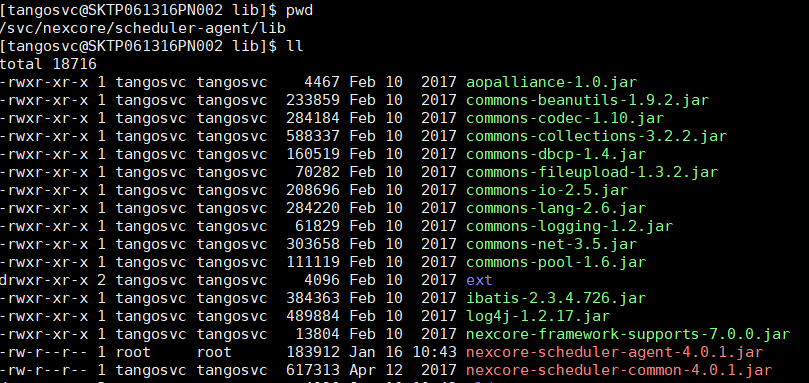
(setenv.sh)

log 폴더 : batch-schdeuler-server 동작 LOG를 관리한다.

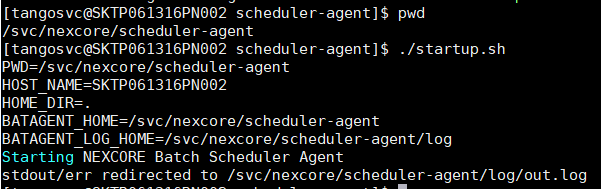


1. 배포 테스트를 위해 1)에서 생성해 둔 nexcore-batch-agent.jar 파일을 아래 경로에 배치한다.

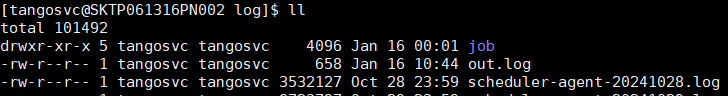
* /svc/nexcore/scheduler-agent/lib 경로에 신규 생성한 nexcore-scheduler-agent-4.0.1.jar를 배치한다.

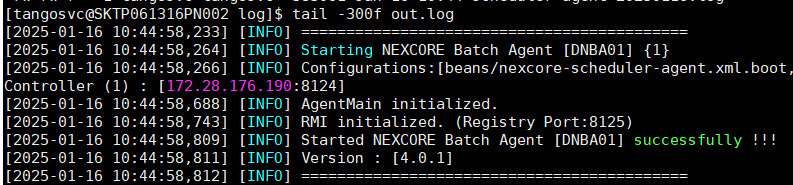


* /svc/nexcore/scheduler-agent 경로에서 startup.sh를 수행한다.

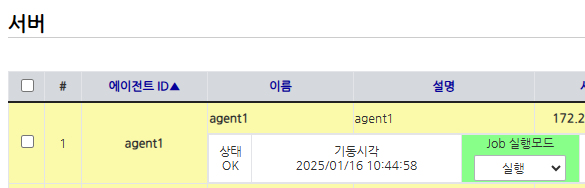


* Nexcore-batch-scheduler-agent start 로그 확인



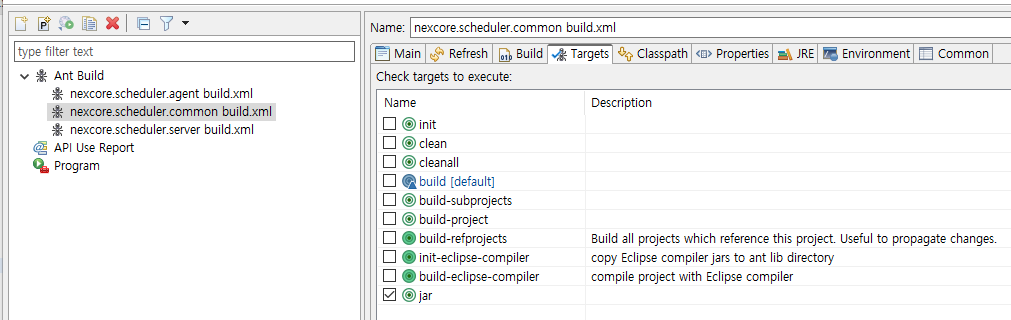


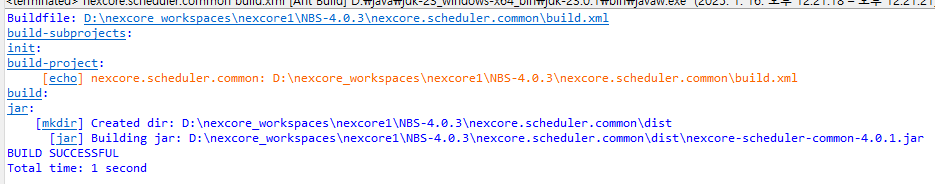
1. UI에서 서버 메뉴에서 에이전트가 감지되었는지 확인



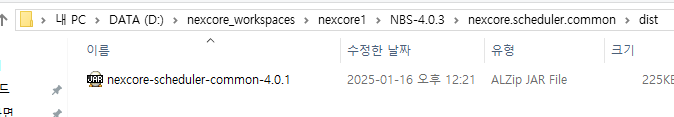
1. Nexcore-scheduler-common 프로젝트 빌드 및 배포
2. 만들어 놓은 build.xml 파일을 ant로 수행시켜 준다.

수행 시킬 때 targets을 설정하여 jar 파일이 떨어지도록 수행한다.





1. /dist 폴더에 프로젝트 jar 파일이 생성되었는지 확인한다.

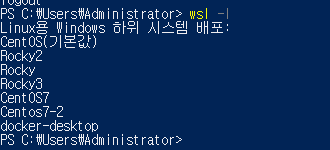


1. 이전 단계에서 생성한 CentosOS7에 접속하여, 작업을 진행합니다.

Jar 파일 이외에 필요한 파일들은 압축 파일 중의 reference경로로 올려 두었습니다. 해당 경로는 실제 NEXCORE가 실행되었던 폴더를 가져 온 것이니, 해당 경로의 파일들을 복제하고 lib안에 ‘nexcore-scheduler-common’ 을 갈아 끼워서 테스트 합니다.

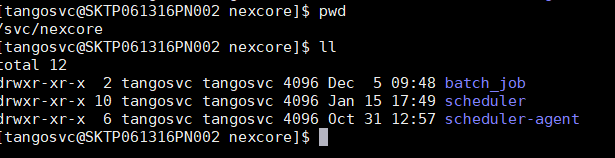
nexcore-scheduler-server, nexcore-scheduler-agent 프로젝트에 모두 반영해야 한다.

* wsl –l 로 Centos7 으로 만들어 놓은 가상 OS 확인

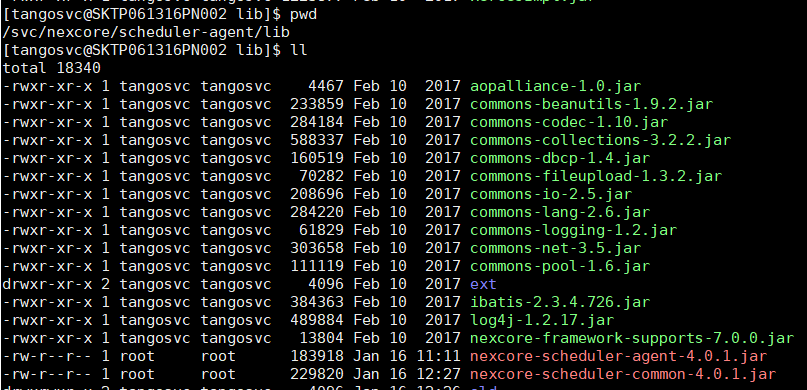


* 해당 Centos7 OS에 접속

: ssh를 통해 서버에 접속해서 아래 디렉토리를 만들고 배치스케줄러 구조 복제이(이미 scheduler, scheduler-agent를 복제하였으므로, 해당 폴더내 lib에 nexcore-batch-common.4.0.1.jar를 덮어씌워준다..)(reference 폴더 참조)



1. scheduler-agent : 배치스케줄러 에이전트 경로 내의 lib에 덮어씌워준다. 이전 nexcore-batch-common-4.0.1.jar는 백업해두는게 좋다.



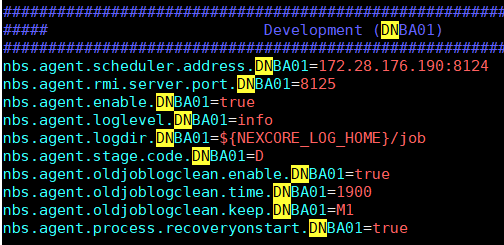
1. scheduler : 배치스케줄러 서버 경로 내의 lib에 덮어씌워준다. 이전 nexcore-batch-common-4.0.1.jar는 백업해두는게 좋다.

startup.sh : batch-scheduler-agent를 수행시킨다.

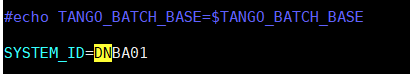
shutdown.sh : batch-scheduler-agent를 중지한다.

config 폴더 : scheduler-agent에 대한 주 properties를 관리한다.

* config/properties/nexcore-scheduler-agent.properties (sample)

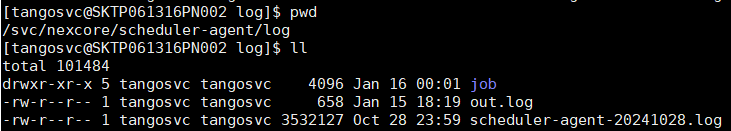


(IP 정도만 수정하여주고, DNBA01 도 setenv.sh에서 관리되기 때문에 변경 시에 해당 파일도 변경하여 준다.)



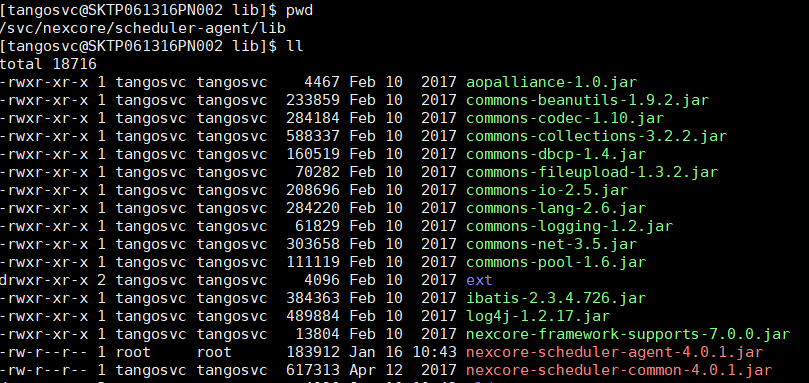
(setenv.sh)

log 폴더 : batch-schdeuler-server 동작 LOG를 관리한다.

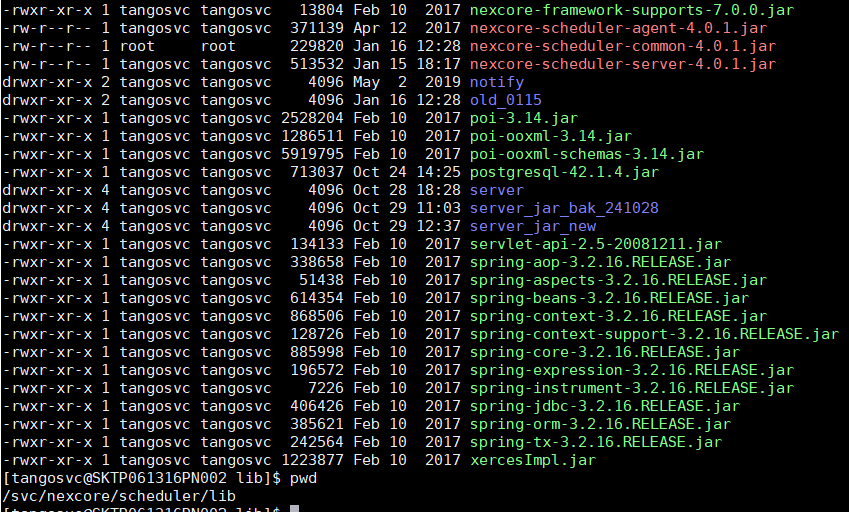


1. 배포 테스트를 위해 1)에서 생성해 둔 nexcore-batch-agent.jar 파일을 아래 경로에 배치한다.

* /svc/nexcore/scheduler-agent/lib 경로에 신규 생성한 nexcore-scheduler-agent-4.0.1.jar를 배치한다.

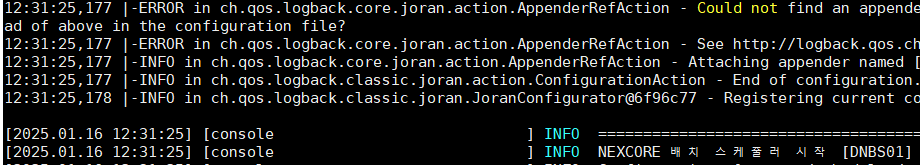


* /svc/nexcore/scheduler/lib 경로에 신규 생성한 nexcore-scheduler-agent-4.0.1.jar를 배치한다.



1. scheduler-server, scheduler-agent를 각각 shutdown.sh 하고 startup.sh한 후에, 이상이 없는지 로그를 확인한다.

(scheduler-server)



(scheduler-agent)

