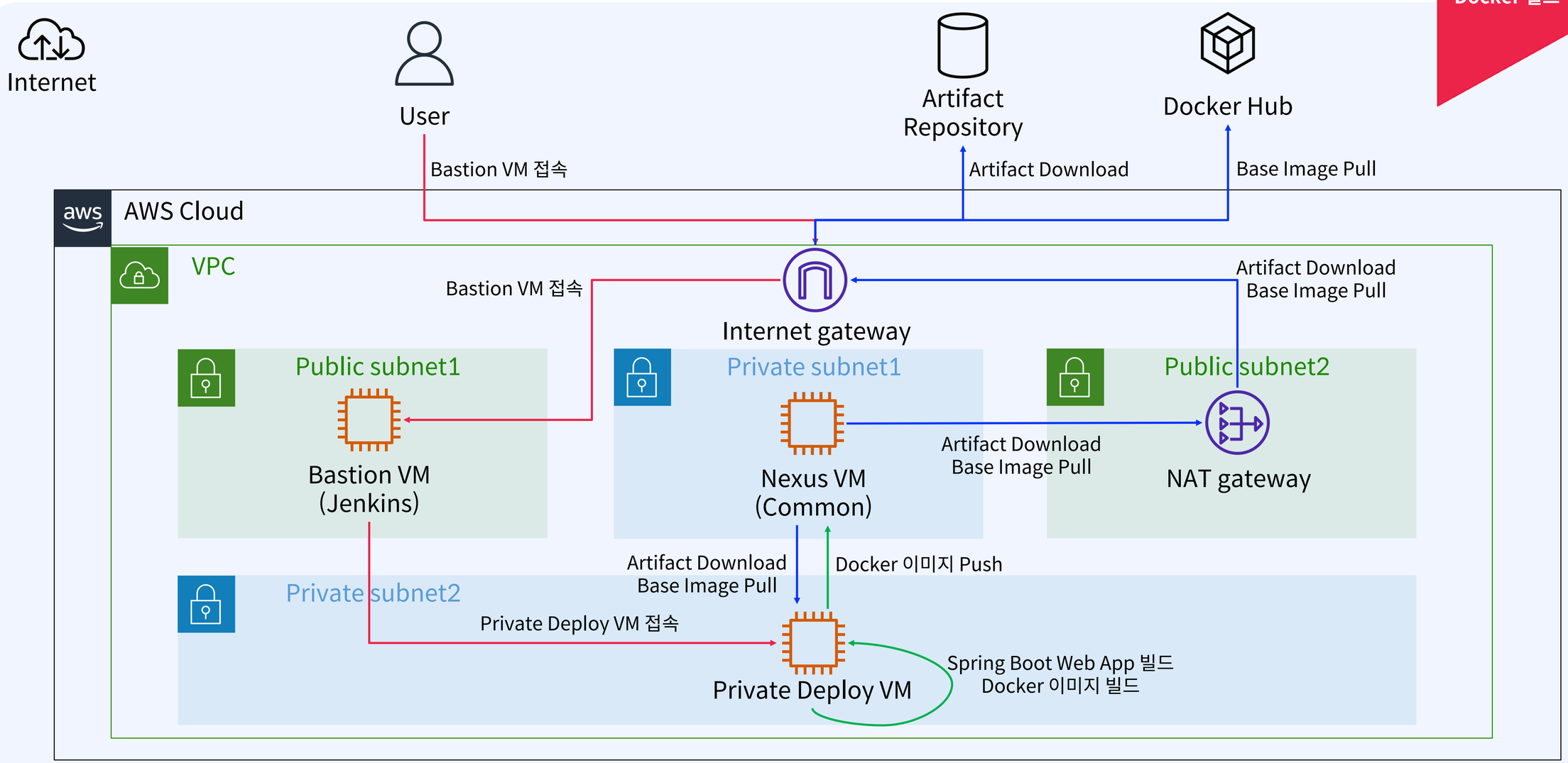
8 Docker 트러블 슈팅 방법

04 Nexus를 활용한 Private 환경 Docker 빌드

Private 환경에서 Nexus 활용



실습 진행사항

Nexus를 활용한 Private 환경 Docker 빌드 실습

1. Private 환경 구성 (NAT Gateway)
2. Nexus Repository 설정(HTTPS)
3. HTTPS로 Nexus Registry 로그인 검증
4. 마이크로서비스의 Gradle 및 Jib 설정
5. Spring App 빌드 및 Docker 이미지 빌드/배포

사전 준비사항

- AWS EC2 VM Instance 3개
  - 실습에서는 Common(Nexus용)/Private Deploy(마이크로서비스/컨테이너 빌드/배포용) 2개 VM 사용
  - Private 환경으로 변동시 Jenkins VM으로 Private Deploy VM 접속 (Bastion 서버 역할)
- AWS VPC내 4개 Subnet 구성
  - Public Subnet 2개, Private Subnet 2개

사전 준비1. AWS VPC내 4개 Subnet 구성 #1

- Public Subnet1 구성
  - Private Subnet2내 Deploy VM 접속용 네트워크
  - Bastion(Jenkins) VM 실행중
  - Internet Gateway 라우팅 추가 (0.0.0.0/0)

- Public Subnet2 구성
  - Private Subnet1내 Nexus(Common) VM에서 인터넷망에 있는 시스템 접속용
  - NAT Gateway 추가 – 실습1
  - Internet Gateway 라우팅 추가 (0.0.0.0/0) – 실습1

사전 준비1. AWS VPC내 4개 Subnet 구성 #2

- Private Subnet1 구성
- Private 환경내 Nexus 운영/관리 환경을 구현할 네트워크
- Nexus(Common) VM 실행중
- NAT Gateway 라우팅 추가 (0.0.0.0/0) – 실습1

- Private Subnet2 구성
- Private 환경내 개발/배포 환경을 구현할 네트워크
- Private Deploy VM 실행중
- Private Deploy의 경우 Spec은 다음과 같음
  - t3.medium (2Core, 4GB Mem, 8GB EBS Volume)
- 라우팅 추가 없음

사전 준비2. 예제 코드 준비

- 예제 코드 준비

- Private Deploy VM 대상
- Part2_Docker > Chapter08 > 4_pri-nexus-docker > microservice
- 개인 Repository에서 미리 예제 코드를 대상 VM에 git clone
- 예제 코드 준비할 때만 인터넷에서 git clone을 위해 NAT Gateway 라우팅을 적용

- 예제 코드 준비후 AWS Route Table 수정

- Private Subnet2가 완전한 Private 환경이 되도록 NAT Gateway 라우팅 제거
- 라우팅 제거후 Private Deploy VM은 Bastion(Jenkins) VM으로만 접속가능

실습1. Private 환경 구성 (NAT Gateway)

- Public Subnet2, Private Subnet1 라우팅 생성
  - Internet Gateway 라우팅 추가 (0.0.0.0/0)

- NAT Gateway 생성
  - Public Subnet2 내에 생성
  - Elastic IP는 자동 할당

- Private Subnet2 라우팅 생성
  - NAT Gateway 라우팅 추가 (0.0.0.0/0)

실습2. Nexus Repository 설정 #1

(1) Nexus 설정 개요

- 인증서를 적용한 HTTPS를 이용해서 Docker Registry로의 로그인을 구현
- Docker 컨테이너 Push/Pull을 암호화 해서 처리하기 위해서는 반드시 HTTPS 방식으로 처리하는게 중요

(2) Nexus 접속 포트 구분

- 5001 : Nexus Docker Hub (Docker Proxy), HTTPS 통신
- 5443 : Nexus Custom Registry (Docker Hosted), HTTPS 통신
- 8081 : Nexus Artifact Repository (Maven2 Proxy), Nexus Manager

실습2. Nexus Repository 설정 #2

(3) Nexus 재배포용 Dockerfile 및 Script 준비

- Nexus에서 인증서 적용하여 HTTPS로 Docker Registry 로그인 및 컨테이너 이미지 관리를 위해 다음과 같이 Dockerfile을 변경한다.
- 추가 Script를 Docker Build시 포함 할 수 있도록 적용한다.

실습2. Nexus Repository 설정 #3

(4) Docker 파일 경로

- Part2_Docker/Chapter08/4_pri-nexus-docker/nexus/Dockerfile

(5) Script 파일 작성

- Part2_Docker/Chapter08/4_pri-nexus-docker/nexus/start.sh

(6) Script 파일대상 실행 권한 부여 및 파일생성 현황 확인

```
$ chmod +x start.sh
$ ls -al
```

실습2. Nexus Repository 설정 #4

(7) 기존 Nexus 컨테이너 중지 및 삭제

- 기존 사용한 Nexus 컨테이너를 중지 및 삭제해도 데이터 파일의 경우는 서버(VM)에 남아있기 때문에 기존에 설정항목이나 데이터는 그대로 남아 있다.
- 이에 Nexus Docker 컨테이너를 중지하고 삭제한다.

```
$ docker ps
$ docker stop <Nexus 컨테이너 ID>
$ docker rm <Nexus 컨테이너 ID>
```

실습2. Nexus Repository 설정 #5

(8) Nexus 컨테이너 실행

- 실행시 주의해야할 점은 SAN_DNS의 변수로는 반드시 Nexus 컨테이너를 실행중인 AWS EC2 VM의 Private DNS 도메인명을 넣어야 한다는 것이다.

- Nexus 재배포 수행

$ docker build -t fastcampus-nexus3:3.32.0 <Dockerfile 포함 경로>

- 생성된 Nexus 컨테이너 이미지 확인

$ docker images

실습2. Nexus Repository 설정 #6

- Nexus 컨테이너 실행시 주의해야할 점은 SAN_DNS의 변수로는 반드시 Nexus 컨테이너를 실행중인 AWS EC2 VM의 Private DNS 도메인명을 넣어야 한다.

```
$ docker run -d -u root --net=host -e SAN_DNS=<Nexus VM의 프라이빗 IP DNS 이름>
    --name nexus -v ~/nexus-data:/nexus-data  fastcampus-nexus3:3.32.0
```

- Nexus 재배포 수행 결과 확인

```
$ docker ps
```

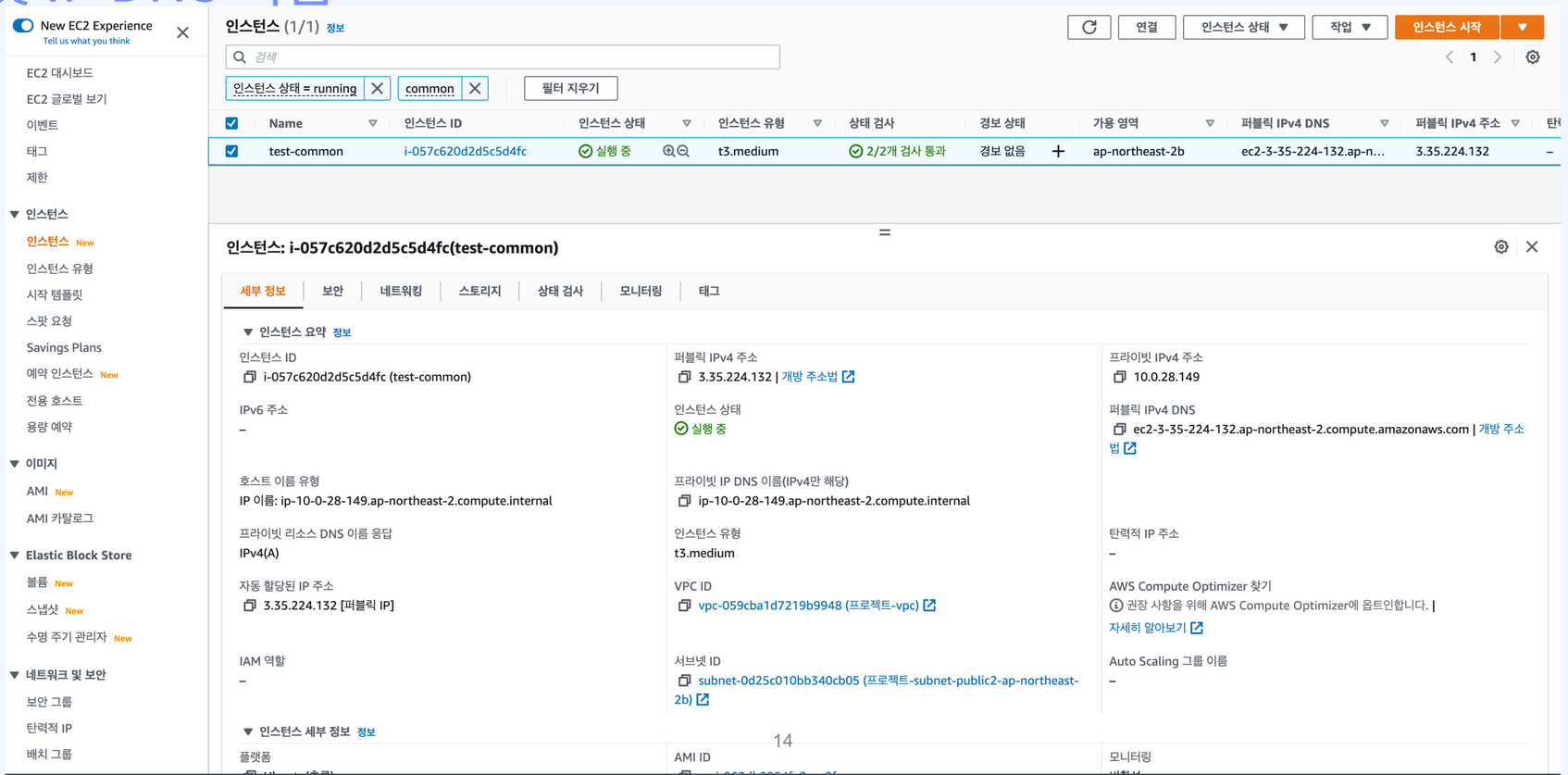
- Nexus 재배포 수행 결과 확인

```
$ docker logs -f <Nexus 컨테이너 ID>
```

실습2. Nexus Repository 설정 #7

- AWS EC2 VM의 Private DNS 도메인명 확인 방법

AWS EC2 > 인스턴스 메뉴 > (Nexus 컨테이너가 실행중인 VM 선택 > 세부정보 > 프라이빗 IP DNS 이름



실습2. Nexus Repository 설정 #8

(9) 인증서 Export

- Nexus 컨테이너에서 HTTPS용도로 사용할 인증서를 생성하고 Export 해야한다.
- 해당 인증서는 오직 현재 서버(VM)위에서 기동중인 Nexus 컨테이너의 Docker Registry를 연결할 때만 사용해야 한다.
- Nexus에서 내장되어 있는 keytool을 이용해서 nexus.crt라는 Nexus 인증서를 생성후 Export 한다.(인증서명은 nexus.crt로 생성된다.)

```
$ docker exec nexus keytool -printcert -sslserver 127.0.0.1:8443 -rfc | tee nexus.crt
```

실습2. Nexus Repository 설정 #9

(10) 인증서 Import

- 해당 인증서를 서버(VM) 내에 Import해서 인증서를 활성화 하고, Nexus에 적용할 수 있도록 nexus.crt파일을 설정한다.

```
$ sudo cp -av nexus.crt  /usr/local/share/ca-certificates/nexus.crt
$ sudo update-ca-certificates
```

실습2. Nexus Repository 설정 #10

- Docker Registry 접속할 때에도 인증서를 참조할 수 있도록 Docker 디렉토리 아래 서버(VM)의 프라이빗 ip dns 이름 및 5443 포트의 디렉토리를 생성한다.
- nexus.crt 인증서를 복사해 CA인증서로 등록한다.

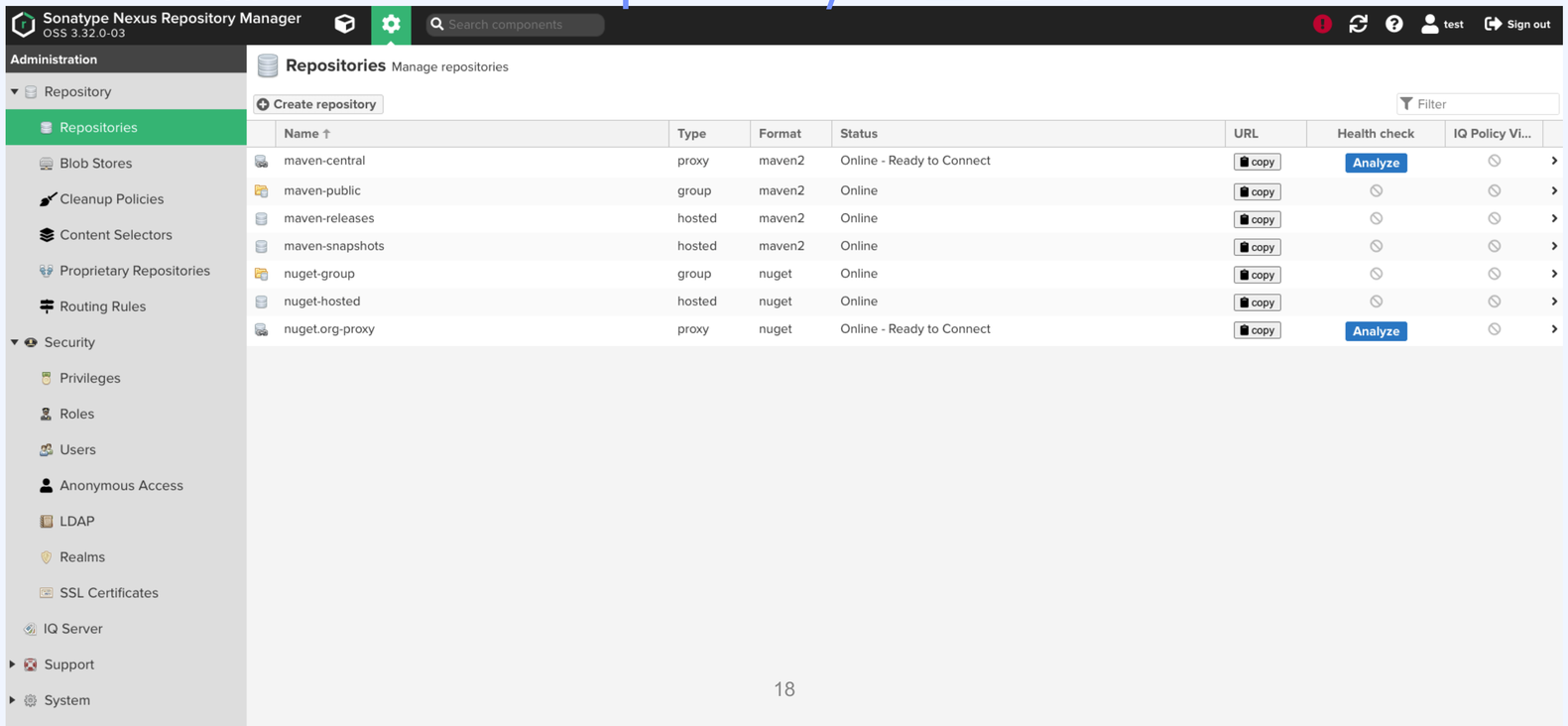
```
$ sudo mkdir /etc/docker/certs.d/<Nexus VM의 프라이빗 ip dns 이름>\:5443/ -p
$ sudo cp -av nexus.crt /etc/docker/certs.d/<Nexus VM의 프라이빗 ip dns
    이름>\:5443/ca.crt
```

실습2. Nexus Repository 설정 #11

(11) Nexus Container Registry 생성

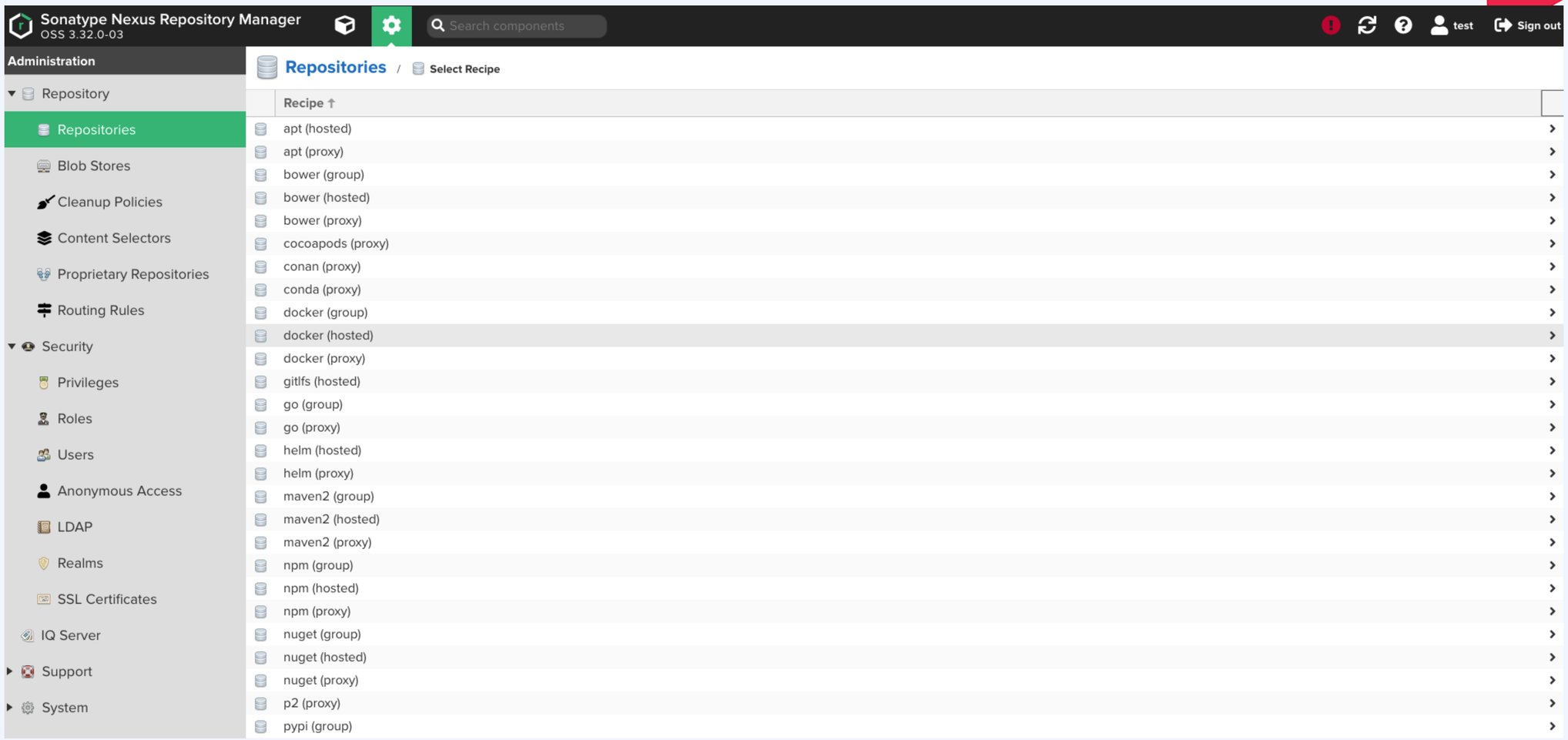
- Nexus Management Console에서 Container Registry 설정을 수행한다.

Nexus Management Console 로그인 > Administration(톱니바퀴) > Repositories 메뉴 > Create Repository 버튼 클릭



실습2. Nexus Repository 설정 #12

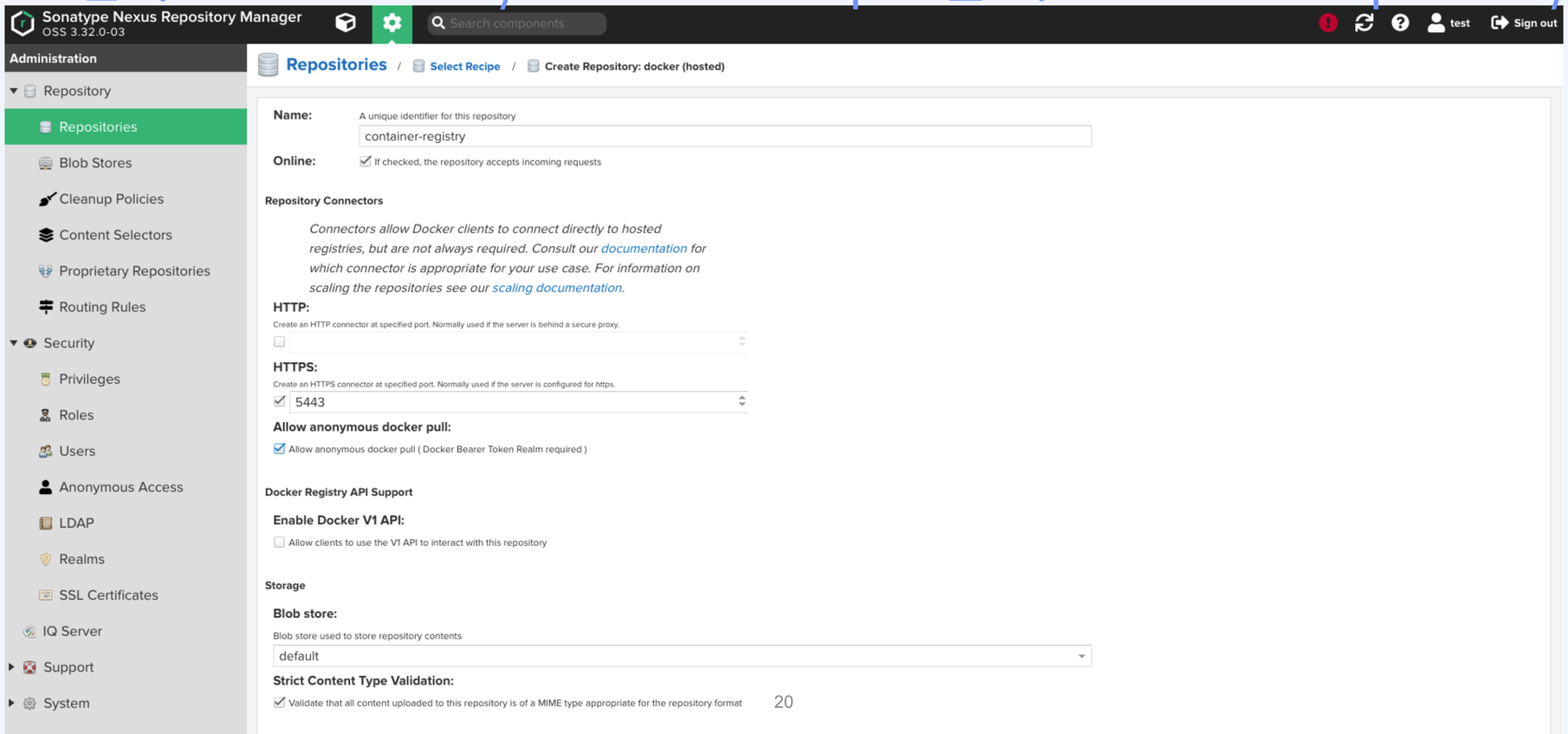
- Docker Hosted를 선택



실습2. Nexus Repository 설정 #13

- Docker Registry관련 정보 입력

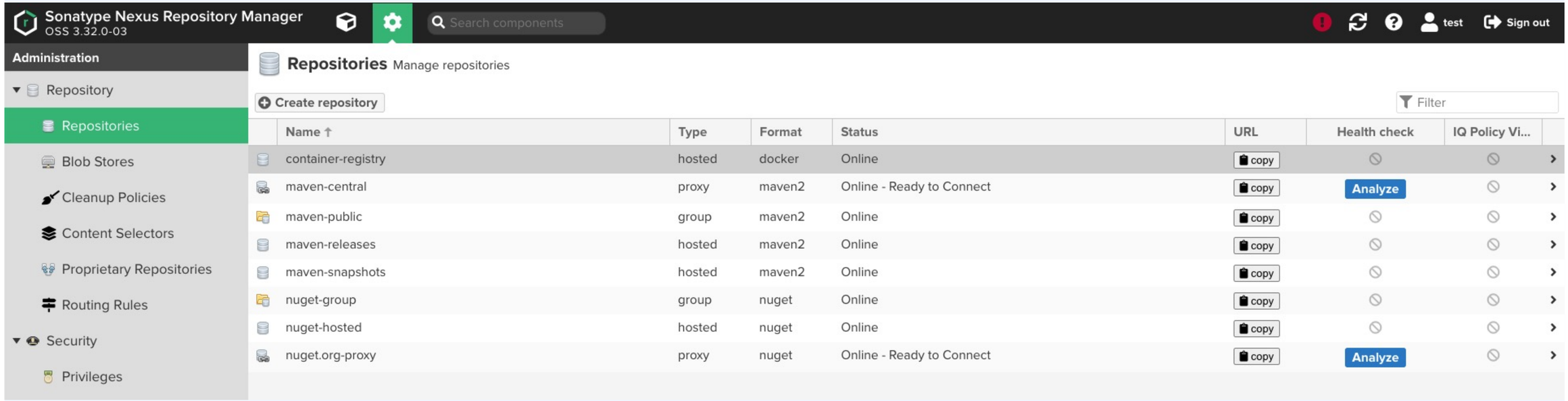
Docker Registry명을 "container-registry"로 입력한다. > https를 체크, 포트를 5443으로 입력 > Allow anonymous docker pull을 체크 > Create repository 버튼 클릭



실습2. Nexus Repository 설정 #14

- Nexus에서 Docker Registry 생성 결과 확인

* Docker Proxy도 동일하게 HTTPS로 Repository를 설정



실습2. Nexus Repository 설정 #15

- Nexus Docker Hub 생성
  - Docker Proxy 타입, 접속 포트 : 5001
  - https://registry-1.docker.io (Docker Hub 연결)

- Nexus Artifact Repository 생성
  - Maven2 Proxy 타입, 접속 포트 : 8081
  - https://plugins.gradle.org/m2/ (Gradle Artifact Central Repository)

- Nexus Repository 설정후 AWS Route Table 수정
  - Private Subnet1이 Private 환경이 되도록 Internet Gateway 라우팅 제거
  - NAT Gateway 라우팅 등록(0.0.0.0/0)

실습2. Nexus Repository 설정 #16

(12) Nexus 서버 로컬에서 Nexus Docker Registry로의 로그인 테스트

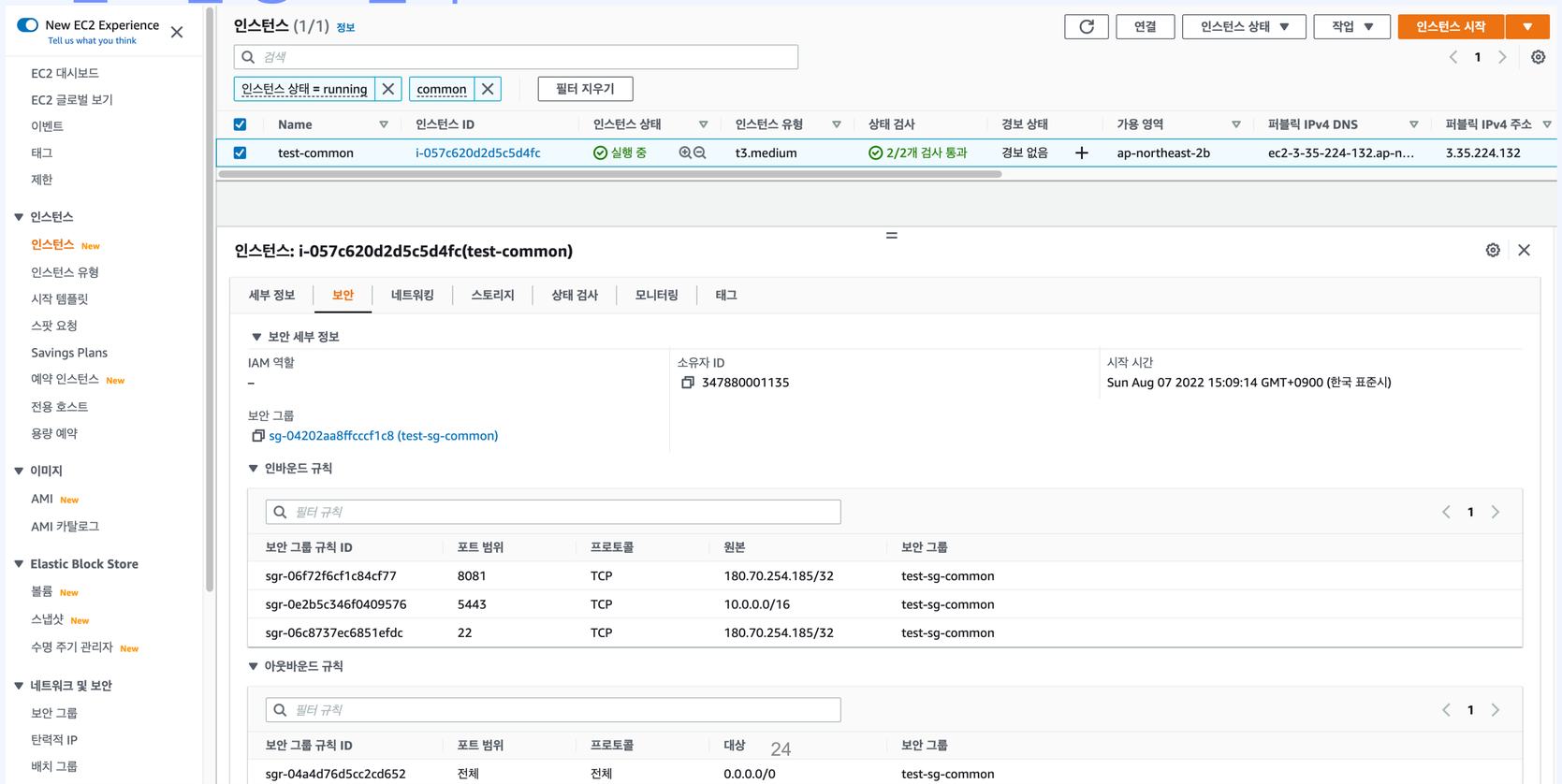
- URL 입력시 반드시 Nexus VM의 Private DNS 도메인명으로 입력한다.
- 포트는 5443으로 설정한다.
- test 계정의 ID 및 Password를 입력한다.
- Login Succeeded가 출력되면 정상적으로 Nexus Docker Registry에 로그인이 된것이다.

$ docker login -u '<Nexus 로그인 계정 ID>' -p '<Nexus 로그인 계정 Password>' https://<Nexus VM의 프라이빗 IP DNS 이름>:5443

실습2. Nexus Repository 설정 #17

(13) Nexus Docker Registry의 HTTPS 포트를 Nexus VM의 Security Group에

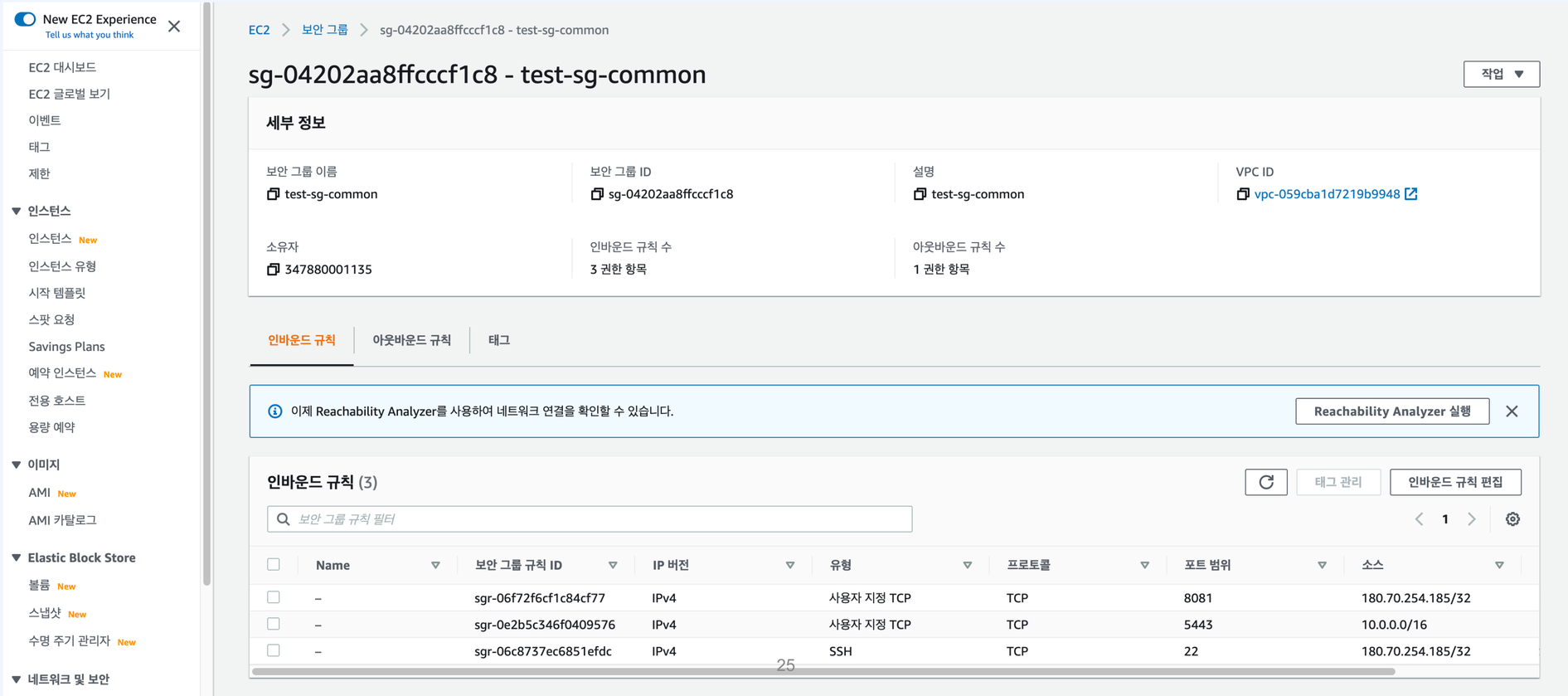
AWS Management Console > EC2 > 인스턴스 > test-common(Nexus VM) 클릭 > 보안 탭 > 보안 그룹 링크 클릭



실습2. Nexus Repository 설정 #18

- 인바운드 규칙 편집을 눌러 다음의 정보를 입력하여 5443 포트 등록

- 유형 : 사용자 지정 TCP, 프로토콜 : TCP, 포트 범위 : 5443, 소스 : <Nexus VM의 VPC의 IP 대역/CIDR 등록>



실습2. Nexus Repository 설정 #19

(14) Private Deploy VM 로컬에서 Nexus Docker Registry로의 로그인 테스트

- Nexus 서버에 있는 nexus.crt 인증서를 복사하여 Private Deploy VM 서버에 붙여넣기하여 파일을 생성한다.

$ vi nexus.crt

- Docker Registry 접속할 때에도 인증서를 참조할 수 있도록 Docker 디렉토리 아래 서버(VM)의 프라이빗 IP DNS 이름 및 5443 포트의 디렉토리를 생성한다.
- nexus.crt 인증서를 복사해 CA인증서로 등록한다.

$ sudo mkdir /etc/docker/certs.d/<Nexus VM의 프라이빗 ip dns 이름>\:5443/ -p

$ sudo cp -av nexus.crt /etc/docker/certs.d/<Nexus VM의 프라이빗 ip dns 이름>\:5443/ca.crt

실습2. Nexus Repository 설정 #20

- Nexus Docker Registry로의 로그인을 다음과 같이 수행한다.
- URL 입력시 반드시 Nexus VM의 Private DNS 도메인명으로 입력한다.
- 포트는 5443으로 설정한다.
- test 계정의 ID 및 Password를 입력한다.
- Login Succeeded가 출력되면 정상적으로 Nexus Docker Registry에 로그인이 된것이다.

  $ docker login -u '<Nexus 로그인 계정 ID>' -p '<Nexus 로그인 계정 Password>' https://<Nexus VM의 프라이빗 IP DNS 이름>:5443

실습3. Nexus Registry (Docker Hub Proxy) 로그인 #1

Private Deploy VM에만 적용

- Nexus Docker Hub Proxy에 로그인

  $ docker login https://<Nexus Private IP>:5001

- Nexus test계정의 ID/Password 입력

실습4. Gradle 및 Jib 설정 #1

settings.gradle 설정

```
pluginManagement {
 repositories {
  maven {
   url "http://<Nexus Private IP>:8081/repository/<Nexus에서 적용한 Repository명>"
   allowInsecureProtocol true
  }
 }
}
rootProject.name = 'test-docker-spring-boot'
```

실습4. Gradle 및 Jib 설정 #2

build.gradle 설정

```
repositories {
  maven {
    url "http://<Nexus Private IP>:8081/repository/<Nexus에서 적용한 Repository명>"
    allowInsecureProtocol true
  }
.. 중략 ..
jib {
  from {
    image = "https://<Nexus Private IP>:5001/<Base Image명>:<이미지 TAG>"
  }
```

실습5. Spring App 빌드 및 Docker 이미지 빌드/배포 #1

Deploy VM에만 적용

- Spring Boot Web Application 빌드 명령어

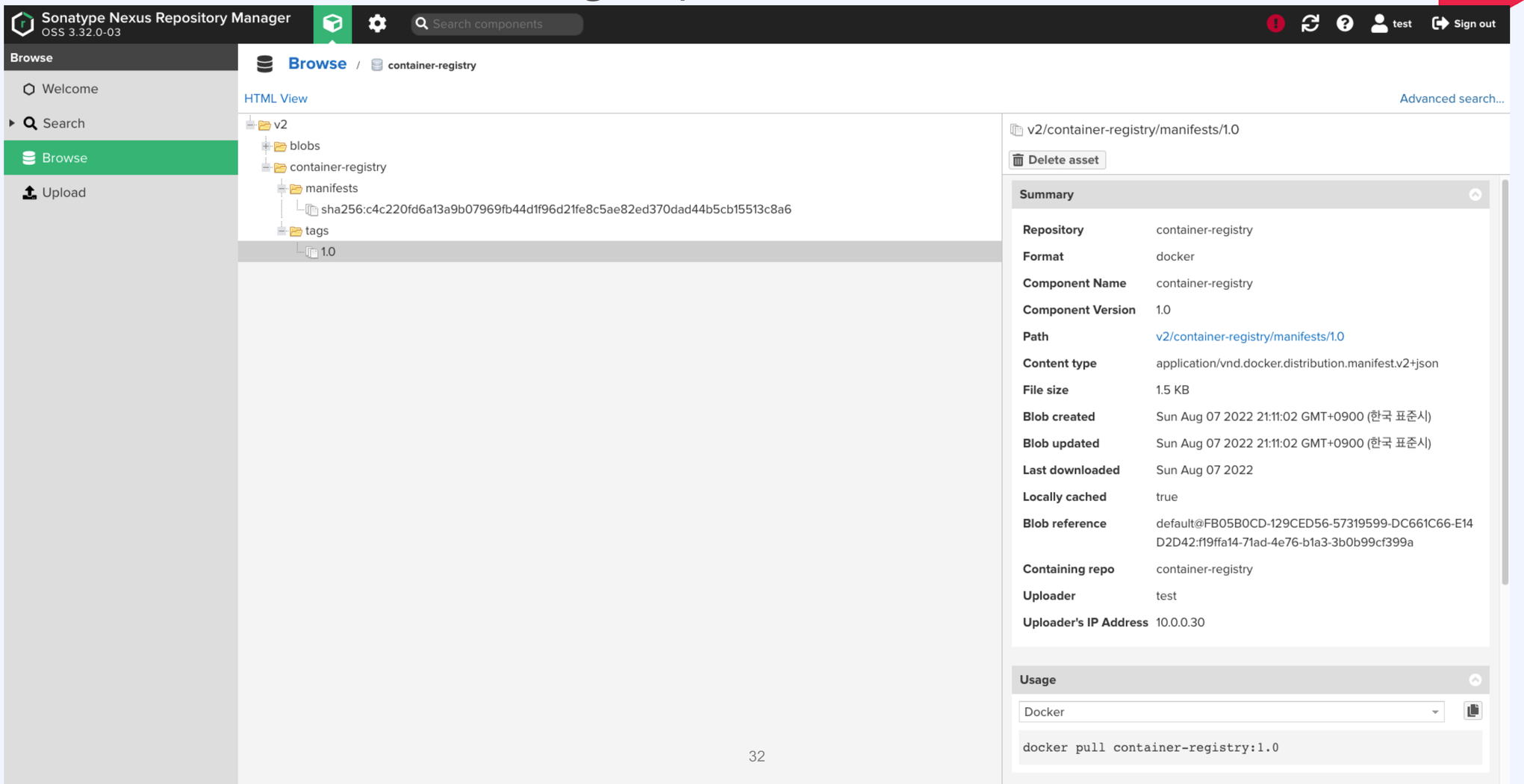
```
$ ./gradlew clean build --info
```

- Docker 이미지 빌드 및 Nexus Custom Registry로의 Push 명령어

```
$ ./gradlew jib -Djib.to.image=<Nexus Private IP>:5443/<Repository명>:<이미지TAG>
    --console=plain
```

실습5. Spring App 빌드 및 Docker 이미지 빌드/배포 #2

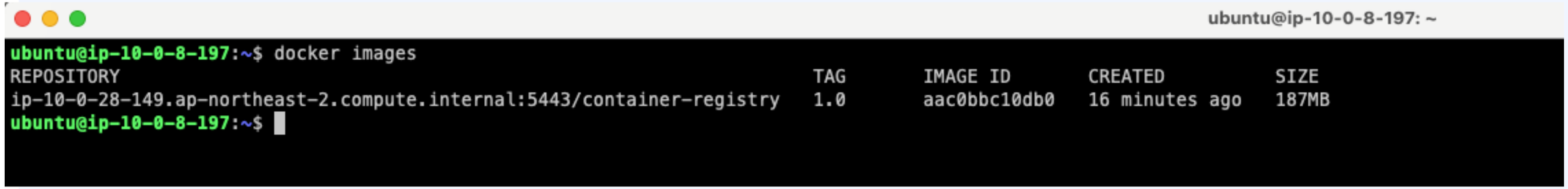
- Nexus 확인 결과, Docker Registry에 정상 Push 및 저장



실습5. Spring App 빌드 및 Docker 이미지 빌드/배포 #3

- Deploy VM에서 Nexus Docker Registry에서의 컨테이너 이미지 Pull 명령어

$ docker pull <Nexus Private IP>:5443/<Repository명>:<이미지 TAG>



실습5. Spring App 빌드 및 Docker 이미지 빌드/배포 #4

- Deploy VM에서 Pull된 컨테이너 정상 기동 실행 명령어

$ docker run -d -p 80:8080 -t <Nexus Private IP>:5443/<Repository명>:<이미지 TAG>

- Spring App 출력 결과 확인

$ curl http://localhost:80 -v