

2022-05-24 09:51:29

**POSCOICT**

**AI 연구개발용 PAAS 솔루션 운영 가이드**

최의규 (Syper)

# 목 차

1	용어 .....	6
2	화면 메뉴 .....	8
3	컨테이너 관리 솔루션 접속 절차 .....	11
	3.1회원가입 및 로그인 .....	11
	3.1.1 접속 URL : <a href="http://aidevs.poscoict.com">aidevs.poscoict.com</a> .....	11
	3.1.2 회원가입 (사용자가 직접 가입 신청) .....	11
	3.1.3 회원가입 (Admin이 회원 등록) .....	12
	3.1.4 Member에 대한 권한 설정(기존 생성된 권한에 Member 등록) .....	12
	3.1.5 권한 신규 생성 .....	12
	3.1.6 로그인 화면 [Admin 권한] .....	14
	3.1.7 로그인 화면 [Member 권한] .....	14
4	Apps .....	15
	4.1 Apps 개념 .....	15
	4.2 Apps 메뉴 설명 .....	15
	4.2.1 Catalog .....	15
	4.2.2 Application .....	15
	4.2.3 Repository .....	16
	4.3 Chart 사용 방법 .....	16
	4.3.1 일반적인 Chart 다운로드 .....	16
	4.3.2 Chart 구성 .....	16
	4.3.3 values.yaml .....	17
	4.3.4 Chart 다운로드 .....	19
	4.3.5 Chart 업로드 (POSCO) .....	20
	4.3.6 Chart 배포 .....	20
5	솔루션 관리절차 .....	22
	5.1K8S 마스터(Master) 관리절차 .....	22

5.1.1 설치 및 환경설정 경로.....	22
5.1.2 기동 및 종료 절차.....	22
5.1.3 모니터링 및 로그 관리.....	23
5.2 Worker 노드(Node) 관리절차 .....	23
5.2.1 설치 및 환경 설정 경로 .....	23
5.2.2 기동 및 종료 절차.....	23
5.2.3 모니터링 및 로그 관리.....	24
5.3 Private Docker Registry 관리절차 .....	24
5.3.1 설치 및 환경설정 경로.....	24
5.3.2 기동 및 종료 절차.....	24
5.3.3 모니터링 및 로그 관리.....	25
5.4 DNS 서비스 관리절차 .....	25
5.4.1 설치 및 환경설정 경로.....	25
5.4.2 기동 및 종료 절차.....	25
5.4.3 모니터링 및 로그 관리.....	26
6 MIG GPU Configuration .....	27
6.1 MIG 기능 활성화 및 삭제 .....	27
6.1.1 MIG 기능 활성화 .....	27
6.1.2 MIG 기능 비활성화 .....	27
6.1.3 nvidia-smi 옵션 설명 .....	27
6.1.4 MIG 예시 .....	27
6.1.5 MIG 기능 자동 적용 .....	28
7 Image 사용 방법.....	30
7.1 Docker Image 개념.....	30
7.2 Docker Image 검색.....	30
7.3 Docker Image 다운로드.....	30
7.4 Docker Image 업로드.....	31
7.5 표준 빌드이미지 설명 (예시).....	32
8 Kubernetes Command .....	39

8.1 Persistence Volume 관리 .....	39
<b>8.1.1 PV(Persistence Volume) 조회 .....</b>	<b>39</b>
<b>8.1.2 PV(Persistence Volume) 등록 .....</b>	<b>39</b>
<b>8.1.3 PV(Persistence Volume) 제거 .....</b>	<b>40</b>
8.2 Persistence Volume Claim 관리 .....	40
<b>8.2.1 PVC(Persistence Volume Claim) 조회 .....</b>	<b>40</b>
<b>8.2.2 PVC(Persistence Volume Claim) 등록 .....</b>	<b>40</b>
8.3 POD 관리 .....	40
<b>8.3.1 POD 조회 .....</b>	<b>40</b>
<b>8.3.2 POD 상세 조회 .....</b>	<b>41</b>
<b>8.3.3 POD 로그 조회 .....</b>	<b>41</b>
<b>8.3.4 POD 로그 Watch .....</b>	<b>41</b>
<b>8.3.5 POD 로그 저장 .....</b>	<b>41</b>
<b>8.3.6 POD 원격 접근 .....</b>	<b>41</b>
<b>8.3.7 POD Replica .....</b>	<b>41</b>
<b>8.3.8 Executing Remote Command .....</b>	<b>41</b>
9 Docker Command .....	42
9.1 Docker 명령어 .....	42
<b>9.1.1 Docker search 및 image 내려받기 .....</b>	<b>42</b>
<b>9.1.2 Docker image 목록 조회 .....</b>	<b>42</b>
<b>9.1.3 Docker image 컨테이너에 올리기 .....</b>	<b>42</b>
<b>9.1.4 Docker 내 모든 컨테이너 목록 출력 .....</b>	<b>42</b>
<b>9.1.5 Docker 컨테이너 Start / Stop / Restart / Kill .....</b>	<b>43</b>
<b>9.1.6 Docker 컨테이너 및 이미지 삭제 .....</b>	<b>43</b>
<b>9.1.7 Docker 컨테이너 내 명령어 실행 .....</b>	<b>43</b>
<b>9.1.8 Docker 새로운 이미지 생성 .....</b>	<b>44</b>
<b>9.1.9 Docker 부모 이미지와 현재 컨테이너 간 비교 .....</b>	<b>44</b>
<b>9.1.10 Docker 컨테이너 상태 정보 확인 .....</b>	<b>44</b>
<b>9.1.11 Docker 컨테이너 내 실행 중인 프로세스 보기 .....</b>	<b>44</b>
<b>9.1.12 Docker 컨테이너 또는 이미지 low-level 정보 보기 .....</b>	<b>45</b>
<b>9.1.13 Docker tar 파일 이미지 로드 방법 .....</b>	<b>45</b>
<b>9.1.14 Docker 레파지토리 내 이미지 태그 .....</b>	<b>45</b>

9.1.15 Docker 이미지 또는 레지스트리에서 레파지토리로 푸쉬 .....	45
9.1.16 Dockerfile로 이미지 생성 및 어플리케이션 실행하기 .....	46
9.1.17 Docker 명령어 요약 .....	47

최의규 (Syper)



2022-05-24 09:51:29

## 1 용어

구분	기능	내용
Brick	CS서버 (Brick Master)	사용자 로그인, 애플리케이션 관리, 도메인 생성 등 솔루션의 동작을 관리하는 Host
	API/Authentication	사용자의 인증 및 CLI(Command Line Interface), REST API를 처리.
	ETCD	0000솔루션 정보를 저장하는 데이터저장소(KEY-VALUE)
	Deployment	Build(빌드)과정에서 생성된 이미지를 heduler(스케줄러)에 요청하여 POD(컨테이너)를 배포
	Scheduler	POD(컨테이너) 요청정보를 처리하여 알맞은 노드에 할당.
	Replication Controller	POD(컨테이너) Lifecycle를 관리 예) POD(컨테이너) Down된 경우, 새로운POD(컨테이너)를 자동으로 Deployment(배포)
	Replica	POD(컨테이너)의 확장/축소 정보. 즉 유지 되어야할 POD(컨테이너)의 개수 정보.
Namespace	Namespace	서비스가 실행되는 논리적 구성으로 자원할당 및 빌드, 배포, 이미지의 관리기준
Object	Object	서비스관련 POD(컨테이너), Service(서비스), Ingress 등의 관련 컴포넌트를 정의
Node	Node	실제 물리적인 서버 또는 Virtual Machine(VM)으로 구성되는 Host이미지 기반 Docker 컨테이너들이 실제 배치되는 영역
	POD	하나의 Docker 이미지를 실행하기 위한 자원 컨테이너 논리적인 자원들(CPU, Memory)이 Resource Limits나 Quota 설정으로 할당 배치되어 사용하는 영역

	Docker Image	개발 언어, 웹 프레임워크, 데이터베이스 등의 애플리케이션 플랫폼을 이미지 형태로 패키징
	Service	서비스에 관련된 컨테이너들의 정보를 통해 연결 서비스 제공
	Ingress	HTTP/HTTPS 도메인을 통해 서비스 네트워크와 컨테이너를 연결해주는 역할
	Registry	프로젝트(0000솔루션) 빌더이미지와 소스가 빌드된 후 이미지가 저장되는 저장소.
	Persistence Volume	POD(컨테이너)에 영속적인 저장소를 제공

최의규 (Syper)

posco  
ICT

## 2 화면 메뉴

구분	기능	내용
Cluster	Dashboard	자원의 사용량 및 사용 현황 정보를 확인할 수 있는 대시보드
	Nodes	Node의 사용량 및 정보를 확인 작업 시 Unschedulable할 수 있는 기능과 Maintenance Mode를 구성할 수 있다.
Apps	Catalog	컨테이너 등 복잡한 구성의 오브젝트들을 한 번에 실행할 수 있도록 저장소에 등록되어 있는 Chart 리스트
	Application	배포되어 운영중인 Chart 리스트. 네임스페이스 별로 구분
	Repository	Chart 저장소에 대한 정보를 확인하고, 등록할 수 있다.
Container	Pod	하나의 컨테이너 이미지를 배포하는 최소 단위
	Deployment	Pod에 대해 복제본과 버전 등을 컨트롤 할 수 있는 컨트롤러 Pod를 배포하는 상위 컨트롤러에 해당
	Statefulset	Deployment와 유사하나, Stateful한 Pod를 배포하는 경우 사용. 상태 유지가 필요한 경우 사용하며, Pod 별 데이터 유지를 위해 VolumeClaimTemplates를 사용한다. 일반적으로 Active/Slave 같은 방식에 사용한다.
	DaemonSet	Deployment와 유사하나, 각 노드 별로 하나의 데몬처럼 동작해야 하는 경우 사용
	Job	Pod와 동일하지만, Pod는 항상 프로세스가 떠 있는 상태로 유지되는 반면, Job은 1회성으로 명령을 수행하고 끝낸다.
Environment	ConfigMap	Pod 실행 시 설정을 사용하기 위해 사용하는 오브젝트 예를 들어, 설정파일의 내용 등을 등록

	Secret	ConfigMap과 유사하나, 공개되지 말아야 할 암호 등에 사용. ConfigMap은 내용 조회시 모든 내용이 표시되는 반면, Secret의 경우 base64로 암호화되어 있다.
	HorizontalPodAuto scaler	Pod의 자동 확장 기능. Deployment나 StatefulSet인 경우 사용 가능 지정한 사용량 임계치를 초과하는 경우 자동으로 확장
Networking	Service	Pod는 생성 시 동적으로 IP를 할당받게 되는데, 해당 Pod를 찾아갈 수 있도록 하는 오브젝트
	Ingress	외부에서 연결 시 도메인으로 접속할 수 있도록 하는 설정 (nginx-ingress라는 컨트롤러에 의해 실행)
	NetworkPolicies	특정 Pod에 대한 접속 IP나 네임스페이스 제한 등을 설정
Volume	PersistentVolumeClaim	영구볼륨을 Pod에서 호출하여 사용할 수 있도록 bind하는 오브젝트
	PersistentVolume	영구 볼륨. 컨테이너의 내부는 휘발성 디스크로, 데이터를 저장 유지하기 위해서는 영구볼륨이 필요
	StorageClass	클러스터에서 내부적으로 스토리지를 사용하기 위해 Software-defined Storage를 생성하고 class를 지정하여 사용
Resource	ResourceQuotas	네임스페이스 별로 사용할 수 있는 자원 제한 설정
	LimitRanges	지정한 네임스페이스의 Pod들이 사용할 수 있는 자원의 제한을 설정
Logs	History	사용자의 작업에 대한 히스토리
	Events	클러스터 내부에서 발생하는 이벤트에 대한 정보

Utility (상단 메뉴)	Namespace	Namespace의 사용량 및 정보를 확인 네임스페이스(프로젝트)를 생성.삭제.관리 할 수 있음.
	QuickStart	간단한 단일 컨테이너를 간편하게 실행할 수 있는 기능
	사용자 관리	사용자에 대한 정보와 권한을 관리
	알람	노드에 문제가 있을 경우, 해당 정보를 알람기능을 통해 표시

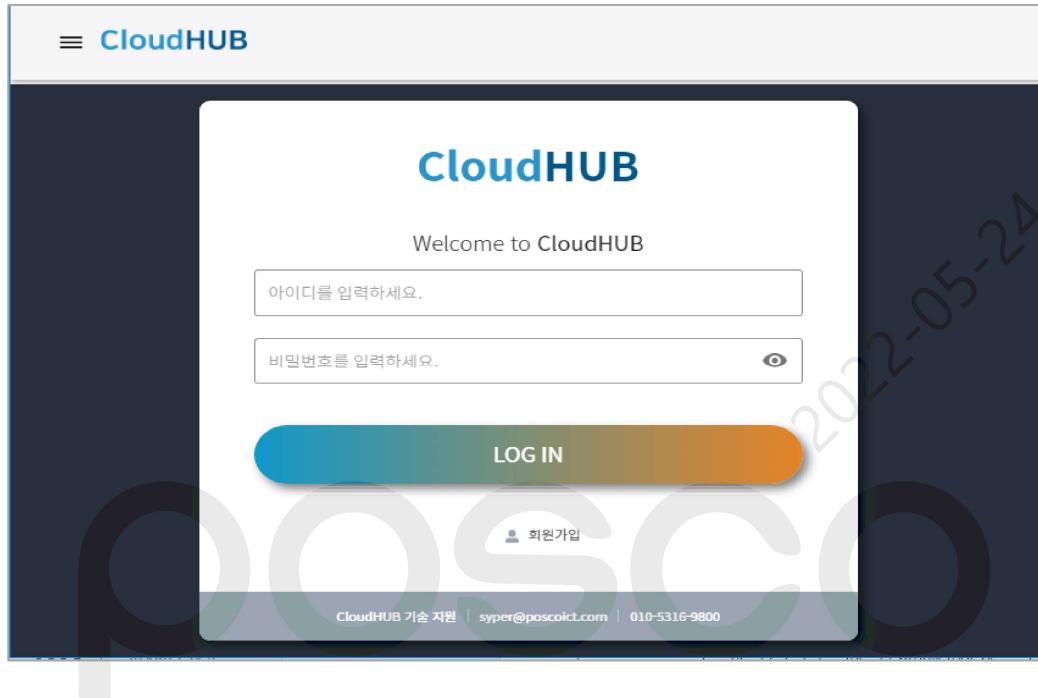
최의규 (Syper)

posco  
ICT

### 3 컨테이너 관리 솔루션 접속 절차

#### 3.1 회원가입 및 로그인

##### 3.1.1 접속 URL : aidevs.poscoict.com



##### 3.1.2 회원가입 (사용자가 직접 가입 신청)

- 아이디 및 이메일은 사용자 관리를 위해 EP ID로 등록 권고

The screenshot shows the CloudHUB sign-up form. The title 'CloudHUB' is at the top, followed by the heading '회원가입'. The form consists of several input fields with validation messages:
 

- 아이디:** 포스코 EP ID  
계정명은 최소 6자 이상 20자 이하이고 -를 포함하지 않은 영문 소문자와 숫자이어야 합니다.
- 이름:** 풍길동
- 연락처:** 010-1234-5678
- 부서:** 포항 EIC기술부
- 이메일:** 포스코 EP ID@posco.com  
이메일 형식이 둘립니다.
- 비밀번호:** 비밀번호 입력  
패스워드를 확인해주세요
- 비밀번호 확인:** 비밀번호 입력 확인  
패스워드 확인을 확인해주세요

 At the bottom of the form are two buttons: '취소' (Cancel) and '확인' (Confirm).

### 3.1.3 회원가입 (Admin이 회원 등록)

- Admin 로그인 이후 우측상단 Member  클릭 → “Create” 클릭

**Member registration**

**회원 등록**

아이디	포스코 EP ID	9 / 100
제정명은 최소 6자 이상 20자 이하이고 -를 포함하지 않은 영문 소문자와 숫자이어야 합니다.		
이름	홍길동	3 / 100
이메일	포스코 EP ID@posco.com	19 / 100
비밀번호	비밀번호를 입력	0 / 100
패스워드를 확인해주세요		
비밀번호 확인	비밀번호를 입력 확인	0 / 100
패스워드 확인을 확인해주세요		
부서	포항 EIC기술부	9 / 100
전화번호	010-1234-5678	13 / 100
<input checked="" type="radio"/> Admin <input type="radio"/> Member Admin 계정에게는 모든 Namespace에 접근이 가능합니다.		
<a href="#">취소</a> <a href="#">등록</a>		

### 3.1.4 Member에 대한 권한 설정(기존 생성된 권한에 Member 등록)

클러스터 권한	<input type="radio"/> Admin <input checked="" type="radio"/> Member
Admin 계정에게는 모든 Namespace에 접근이 가능합니다.	
Namespace별 권한	2egl
<input type="button" value="test"/> 	

### 3.1.5 권한 신규 생성

오른쪽 상단 그룹사용자 표시 아이콘을 선택하면 사용자 및 권한을 부여할 수 있다.

- 그룹사용자 표시를 선택하면 권한 리스트가 표시
- Create를 선택하고 필요한 권한을 선택 (화면 메뉴와 동일)

Quick Start				
<a href="#">회원 관리</a> <a href="#">권한 관리</a>				

## Member

[Create](#)

Name	Verb	Menu
roletest	READ	<a href="#">Catalog</a> <a href="#">Application</a> <a href="#">Pods</a> <a href="#">Deployments</a> <a href="#">StatefulSets</a> <a href="#">DaemonSets</a> <a href="#">Jobs</a>

## Create Role

Role Name :  [Create](#)

Read  Read/Write  All

- Apps
  - Catalog
  - Application
  - Container
    - Pods
    - Deployments
    - StatefulSets
    - DaemonSets
    - Jobs
- Environment
  - ConfigMaps
  - Secrets

[Cancel](#) [Create](#)

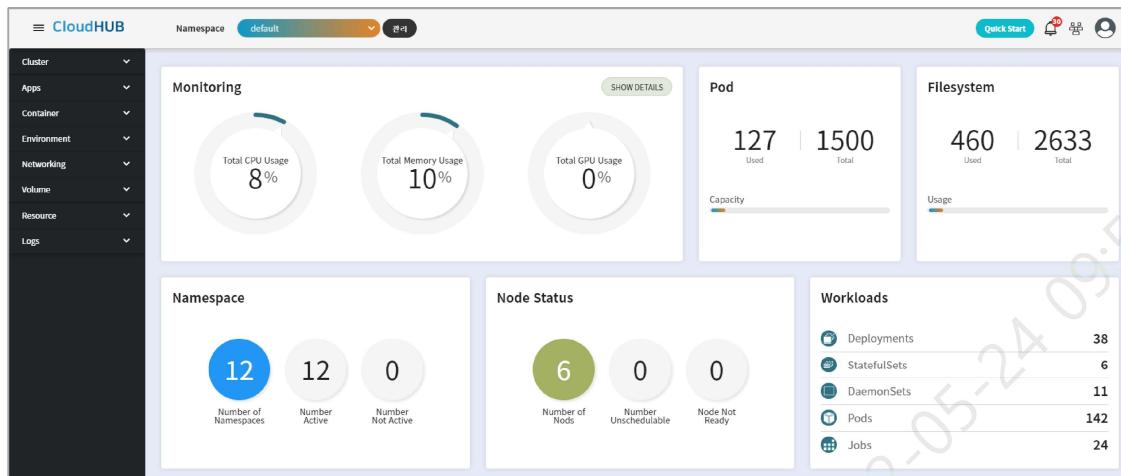
## Member

[Create](#)

Name	Verb	Menu
roletest	READ	<a href="#">Catalog</a> <a href="#">Application</a> <a href="#">Pods</a> <a href="#">Deployments</a> <a href="#">StatefulSets</a> <a href="#">DaemonSets</a> <a href="#">Jobs</a>
basic	READ	<a href="#">Catalog</a> <a href="#">Application</a> <a href="#">Pods</a> <a href="#">Deployments</a> <a href="#">StatefulSets</a> <a href="#">DaemonSets</a> <a href="#">Jobs</a>

### 3.1.6 로그인 화면 [Admin 권한]

- Login 후 Dashboard 화면 : 전체 Cluster에 대한 정보 전체 보여짐



### 3.1.7 로그인 화면 [Member 권한]

- 권한을 할당받은 Namespace와 Role에 대한 정보만 보임



## 4 Apps

### 4.1 Apps 개념

- Apps는 Kubernetes의 패키지 배포매니저인 Helm을 활용하는 방식으로, 복잡한 어플리케이션 구성은 Helm Chart를 활용하여 배포하는 방식. Kubernetes에 배포할 Object를 YAML 형태의 template을 활용하여 만들고 그 template에 사용할 변수를 values.yaml에 지정하여 컨테이너 등 오브젝트를 배포

### 4.2 Apps 메뉴 설명

#### 4.2.1 Catalog

- Catalog는 사전에 Repository에 등록한 Chart 리스트로, 사용자는 Catalog에서 간단히 배포하고자 하는 Chart를 선택하여 배포할 수 있다. 모든 네임스페이스에서 공통의 Chart 리스트를 사용한다.

The screenshot shows the 'Catalog' page of a cloud-based application management interface. The title 'Catalog' is at the top left. Below it, there are six chart entries arranged in two columns of three:

- egl2-coater**: A Helm chart for EGL Service. Provides task(tcp,edit), data base(mysql), python, etc. Version 1. CloudHub icon.
- egl2-zn**: A Helm chart for EGL Service. Provides task(tcp,edit), data base(mysql), python, etc. Version 1. CloudHub icon.
- mysql**: Fast, reliable, scalable, and easy to use open-source relational database system. Version 5.7.30. CloudHub icon.
- tensorflow-notebook**: A Helm chart for tensorflow notebook and tensorboard. Version 1.6.0. CloudHub icon.
- tensorflow-serving**: TensorFlow Serving is an open-source software library for serving machine learning models. Version 1.14.0. CloudHub icon.
- tomcat**: Deploy a basic tomcat application server with sidecar as web archive container. Version 7.0. CloudHub icon.

#### 4.2.2 Application

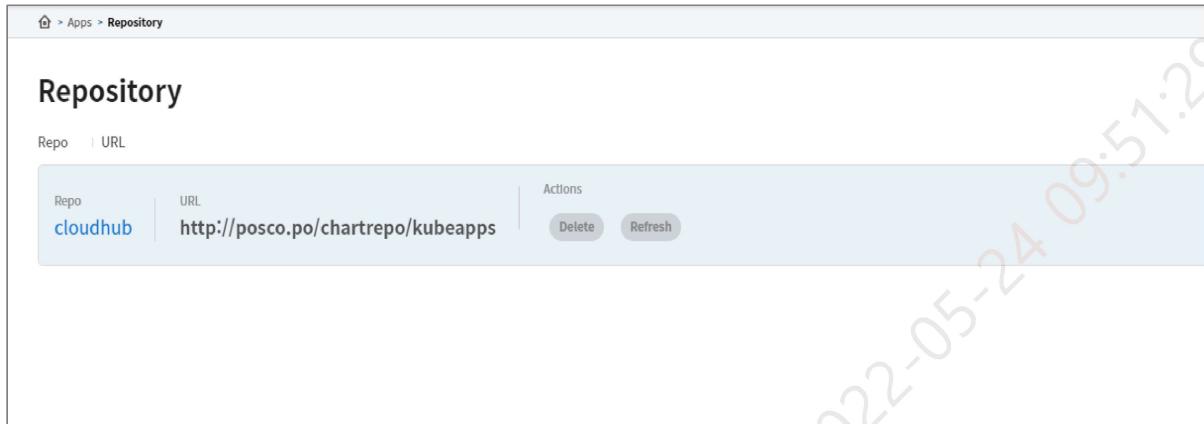
- Application은 실제 배포되어 실행되는 Chart의 리스트로, 현재 네임스페이스 별로 어떤 Chart가 실행중인 상태인지 확인 할 수 있다.

The screenshot shows the 'Application' page of the same interface. The title 'Application' is at the top left. Below it, there are two chart entries:

- egl2-coater**: Version egl2-coater 1. Status: deployed. CloudHub icon.
- egl2-zn**: Version egl2-zn 1. Status: deployed. CloudHub icon.

### 4.2.3 Repository

- Chart의 정보를 가져오는 저장소 위치를 지정  
<http://registry.poscoict.com/chartrepo/kubeapps>로 Private Registry의 Chart 저장소가 지정되어 있다. 참고로 해당 도메인은 DNS에 등록되어 있지 않고 hosts 파일에 등록



## 4.3 Chart 사용 방법

### 4.3.1 일반적인 Chart 다운로드

- 기본적으로 주로 사용되는 어플리케이션의 Chart는 인터넷에서 구할 수 있다. 그 외의 Chart는 기존 차트를 수정하여 구성하거나 별도로 생성하여 구성할 수 있다.

1. <https://artifacthub.io/> 사이트
2. hub.helm.sh
3. charts.bitnami.com/bitnami 등

### 4.3.2 Chart 구성

- Chart.yaml : 차트에 대한 이름 등 기본적인 설정
- templates : 실제 실행이 되는 Kubernetes Object의 YAML 템플릿이 저장되는 공간
- values.yaml : Chart를 배포할 때 필요한 설정값들을 저장하는 파일로, 해당 설정은 templates의 YAML을 실행할 때 변수로 등록되어 실행한다.
- files : 표준 차트를 실행할 때 사용할 서비스를 등록할 수 있도록 하기 위해 구성. supervisor에서 기동 및 모니터링할 서비스를 등록.

### 4.3.3 values.yaml

- 표준 Chart는 global, service, persistence 3부분으로 구성

```

# Default values for standard-project
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

# !Important: If you don't need to set a value,
# just use curly braces instead of commenting it out.

global:
  # Access VIP
  accessIp: 203.238.192.132

  # Hostname of Container, It does not mean container name
  hostname: standard

  extraHosts: {}
    # 192.168.0.1: test1.testdomain.com
    # 192.168.0.2: test2.testdomain.com

  password: "Passw0rd!"

  image: "registry.poscoict.com/library/tensorflow1.15-py36-novnc:cuda10-cudnn7"

  resources: {}
    # limits:
    #   cpu: 1
    #   memory: 1Gi
    #   nvidia.com/gpu: 1

  # Choose which services to enable.
  # Supported Service is jupyter notebook, vnc(xrdp), tensorboard, sshd
  # you also should configure supervisor configuration
service:
  notebook:
    enabled: true
    ingress:
      hostname: standard-notebook.aidevs.poscoict.com

  webvnc:
    enabled: true
    ingress:
      hostname: standard-vnc.aidevs.poscoict.com

  tensorboard:
    enabled: false
    ingress:
      hostname: standard-tensorboard.aidevs.poscoict.com

  sshd:
    enabled: true
    port: 31234

  # Volume Configuration
  # !Important: Changing the value has no effect while the container is running.

```

```

persistence:
  data1:
    type: nfs
    server: 203.238.192.170
    # To use a different path, set the path value below
    # path: /volume1/repo

    size: 1000Gi

    # mountPath value means mountpoint in container
    # The default value is the project name.
    # Specify a value to set a different path
    # mountPath: /standard

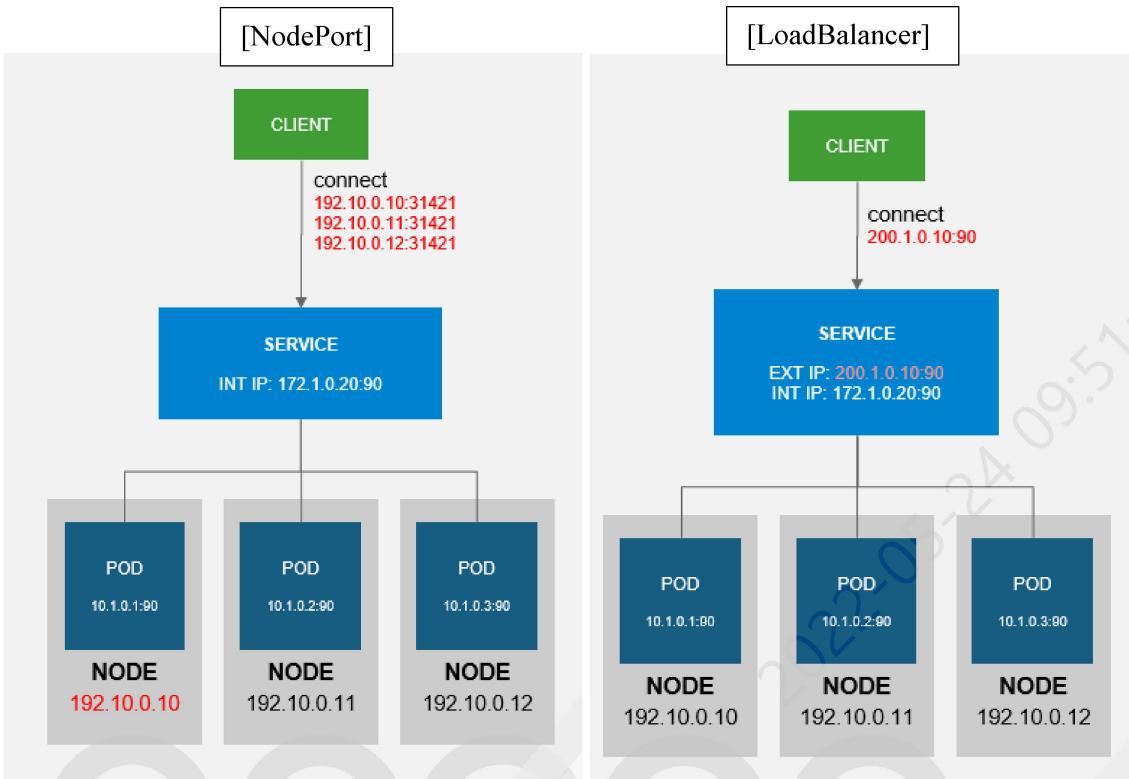
    # Setting subPath will create the directory on the volume path
    # eg, value "standard" creates the /volume1/repo/standard directory.
    # Without a value, no directory is created.
    subPath: standard

  data2: {}
    # type: nfs
    # server: 203.238.192.170
    # size: 1000Gi
    # mountPath: /standard2
    # subPath: standard2

nodeSelector: {}
  # type: nvidiagpu

```

- **accessIP**: ssh 처럼 포트로 오픈하는 서비스에 접속할 때 사용할 IP (실제 서비스 동작에 무관)
- **hostname**: 컨테이너의 호스트네임. (로그인 시 출력되는 호스트네임)
- **extraHosts**: hosts 파일에 도메인 등록하는 것과 같음.
- **password**: 컨테이너에서 실행한 어플리케이션에 접속할 때 사용되는 암호
- **image** : 실행하는 컨테이너의 image 명으로 repository와 tag로 구성되어 있으며, 실제 컨테이너 실행시에는 repository:tag 형태
- **resources**는 컨테이너에 대한 자원을 제한할 때 사용하는 설정으로 **requests**는 자원의 경합이 발생했을 때 보장하는 자원 설정이며, **limits**는 최대한 사용할 수 있는 자원 설정
- **service**: Pod에 연결할 때 사용되는 서비스에 대한 설정 부분. 사용할 서비스를 **enable**하면 되며, 도메인은 중복되지 않도록 적절하게 수정해 주어야 한다. 이와 함께 실제 **enable**할 서비스에 대해 files/svc.conf (supervisor configuration)에 설정해 주어야 한다.
- **port**: 해당 서비스에 대해 외부에서 access하기 위한 port



- **Ingress**: 외부에서 도메인을 통해 연결할 때 사용하기 위한 설정으로 해당 hostname으로 접속할 수 있다. 단, 외부에서 접속할 수 있는 도메인 \*.aidevs.poscoict.com이어야 한다.
- **persistence**: 데이터 유지가 필요한 경우, 외부의 Persistent Volume(영구볼륨)을 이용하여 데이터를 저장할 수 있다. 예를 들어, DB, 분석 모델 등 데이터 저장이 필요한 경우 설정
- **mountPath**: 컨테이너 내에 마운트할 포인트
- **subpath**: subpath를 설정하면 subpath 이름으로 디렉토리가 생성이 되며, 지정하지 않으면 디렉토리가 생성되지 않는다.
- **nodeSelector**: 특정 레이블을 가진 Worker Node에만 배포할 수 있도록 설정

#### 4.3.4 Chart 다운로드

- Catalog 차트 저장소로부터 Chart를 다운로드하여 Chart를 수정하고, 새로운 Chart 이름으로 Catalog에 등록할 수 있다. (Control/Bastion 서버)

```
$ helm repo list
NAME URL
poscoict https://registry.poscoict.com/chartrepo/kubeapps

$ helm search repo poscoict //search
NAME          CHART VERSION APP VERSION DESCRIPTION
poscoict/ad-djlee 1           1           A Helm chart for ML Development Service.
poscoict/ad-hsse 1           1           A Helm chart for ML Development Service.
poscoict/ad-jhy  1           1           A Helm chart for ML Development Service.
```

poscoict/ad-kdw	1	1	A Helm chart for ML Development Service.
poscoict/ad-singasong	1	1	A Helm chart for ML Development Service.

```
$ helm fetch poscoict/ad-kdw --version=1
// download, 버전을 지정하지 않으면 자동으로 최신버전을 다운로드
```

#### 4.3.5 Chart 업로드 (POSCO)

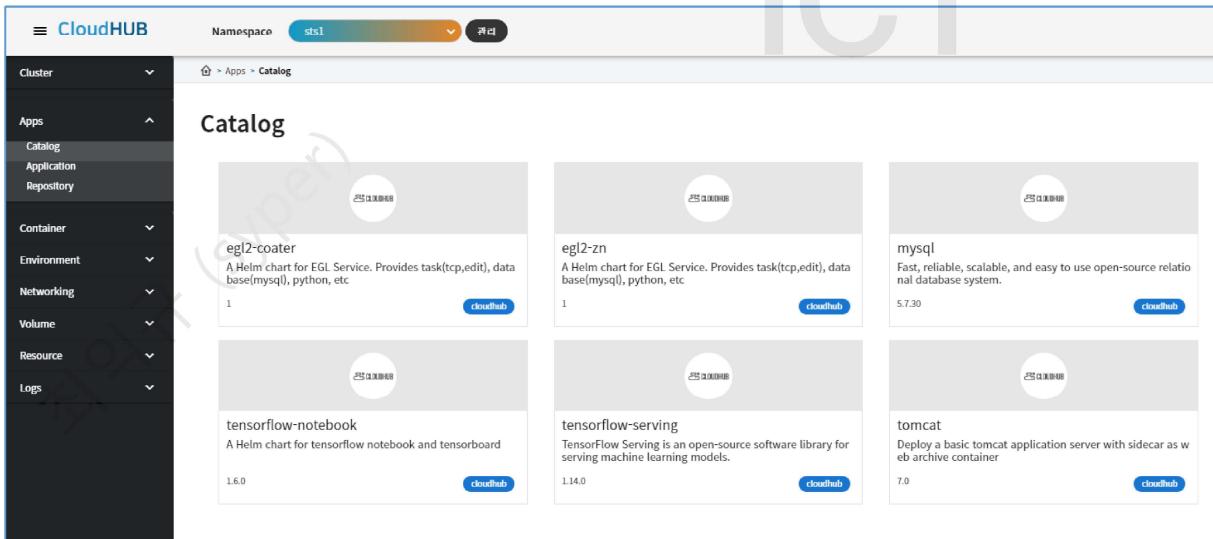
- Chart를 수정 후에 Catalog 차트 저장소에 업로드하여 활용할 수 있다. (컨트롤 서버, 마스터 서버)

```
$ ls -l
total 8
drwxr-xr-x. 2 root root 6 Sep 21 15:27 charts
-rw-r--r--. 1 root root 337 Oct 10 16:42 Chart.yaml
drwxr-xr-x. 5 root root 81 Oct 10 16:32 templates
-rw-r--r--. 1 root root 1874 Oct 10 17:29 values.yaml
```

```
$ helm push . poscoict
// upload, 다른 경로에서 업로드하려면 Chart.yaml 파일이 위치하는 경로를 지정
```

#### 4.3.6 Chart 배포

- 솔루션 화면에 로그인한 후, Apps메뉴에서 Catalog화면으로 진입



- 배포하고자 하는 Chart를 선택

Catalog

**cloudhub/egl2-zn**

1 - cloudhub  
A Helm chart for EGL Service. Provides task(tcp,edit), database(mysql), python, etc.

Chart Versions | App Version | Home

Deploy

- Deploy 버튼을 클릭하면, 배포 관련 상세 설정 화면으로 진입
- 배포할 이름과 네임스페이스, 그리고 여러 개 인 경우 버전을 선택한 후 Submit

Deploy

**cloudhub/egl2-zn**

Name: egl2test

Namespace: egl2

Version: 1

```

1 db:
2   config:
3     dbname: AIMODEL
4     password: posco
5     name: egl2zn-db
6     port: 3306
7     server: 172.16.185.34
8     size: 50Gi
9   type: nfs
10  resources: {}
11  storage: {}
12  clusterIP: None
13  port: 3306
14  type: ClusterIP
15  name: egl2zn-model
16  persistence:
17    mountPath: /data/egl2
18    persistence:
19      mountPath: /data/egl2
20      server: 172.16.185.34
21

```

정상적으로 실행이 되면, Ready 상태로 되며, AccessURL을 통해 서비스에 접속할 수 있다.

application Detail

**sts1**

A Helm chart for AI/ML Edge Service. Provides task(tcp,edit), database(mysql), python, etc.

App Version: 1 | Chart Version: 0.1.0 | Up to date

Ready

Pods: 3

Access URLs

RESOURCE	TYPE	URL
project-sts1-analysis-jupyter	Ingress	<a href="http://example-jupyter.paisc.posco.co.kr/">http://example-jupyter.paisc.posco.co.kr/</a>

Notes

```

1. You get the Jupyter Application URL by running these commands:
http://example-jupyter.paisc.posco.co.kr

2. You get the Rstudio Application URL by running these commands:
http://example-rstudio.paisc.posco.co.kr

3. You get the MySQL Application URL by running these commands:
http://example-mysql.paisc.posco.co.kr

```

Deployments

NAME	AvailableReplicas	Replicas	UpdatedReplicas
project-sts1-analysis	1	1	1
project-sts1-db	1	1	1

## 5 솔루션 관리절차

### 5.1 K8S 마스터(Master) 관리절차

#### 5.1.1 설치 및 환경설정 경로

디렉토리	설명	비고
/var/lib/containers	Container 정보가 저장되는 디렉토리	
/var/lib/etcd	etcd정보를 가지고 있는 디렉토리	

#### 5.1.2 기동 및 종료 절차

##### 5.1.2.1 K8S 마스터 기동 절차

항목	세부 절차	명령어	비고
서비스 기동	마스터 로그인	마스터 서버에 root 계정으로 로그인	
	kubelet	자동 기동 (systemctl start kubelet)	
	api	자동 기동 (컨테이너 - kube-apiserver)	
	controller	자동 기동 (컨테이너 - kube-controller-manager)	
	scheduler	자동 기동 (컨테이너 - kube-scheduler)	
	컨테이너 런타임	자동 기동 (systemctl start crio)	
기동 여부 확인	마스터 로그인	마스터 서버에 root 계정으로 로그인	
	kubelet	systemctl status kubelet	
	api/controller/scheduler	kubectl -n kube-system get pods   grep kube	
	컨테이너 런타임	systemctl status crio	

##### 5.1.2.2 K8S마스터 종료 절차

- 노드 작업 시 3대 중 2대 이상이 기동이 된 상태로 유지하며 1대씩 작업하시기 바랍니다. 2/3(3대 중 2대)가 정지되면, 기존 운영중인 서비스에 문제는 없으나, 데이터 유지를 위해 Read-only 모드로 동작합니다. (신규 작업 불가)

- 시스템 종료 또는 재기동 시, Brick 솔루션 > Cluster > Node 메뉴에서 해당 노드를 선택한 후 “Maintenance mode”로 설정하거나 CLI 상에서 “kubectl drain [노드명]”을 수행하게 되면, 해당 노드에서 동작중인 컨테이너들이 다른 (Master) 노드들로 마이그레이션되는 작업이 수행됩니다(마이그레이션 시 일정 시간 소요). 그 이후 시스템 종료/재기동 작업을 진행하는 절차로 진행하면 됩니다.

하지만, “Maintenance mode” 설정 없이 시스템 종료/재기동 작업을 하더라도, kubernetes에서는 45초(default) 이후 서버가 종료된 것을 인지하게 되며, 이후부터 동일하게 컨테이너들이 다른 노드들로 마이그레이션되는 작업이 수행됩니다.

### 5.1.3 모니터링 및 로그 관리

#### 5.1.3.1 로그 모니터링

로그 파일은 /var/log 디렉토리에 rotation되어 저장되므로 디렉토리 저장공간을 확인하여 주기적으로 정리하시기 바랍니다. /var/log 영역이 포함된 파일시스템을 주기적으로 모니터링하여 일정량 이상의 여유 공간을 확보하세요.

```
# /var/log 영역의 파일시스템 및 사용량 확인
```

```
$ df /var/log
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/mapper/rhel-root 63747676 30157768 33589908 48% /
```

## 5.2 Worker 노드(Node) 관리절차

### 5.2.1 설치 및 환경 설정 경로

디렉토리	설명	비고
/var/lib/containers	Container 정보가 저장되는 디렉토리	

### 5.2.2 기동 및 종료 절차

#### 5.2.2.1 Worker 노드 기동 절차

항목	세부 절차	명령어	비고
서비스 기동	kubelet	자동 기동 (systemctl start kubelet)	
	노드 로그인	각 노드 서버에 root 계정으로 로그인	
	컨테이너 런타임	자동 기동 (systemctl start crio)	
기동 여부 확인	kubelet	systemctl status kubelet	
	노드 로그인	각 노드 서버에 root 계정으로 로그인	
	컨테이너 런타임	systemctl status crio	

#### 5.2.2.2 노드 종료 절차

- Worker Node는 시스템 종료 또는 재기동 작업 시에, Brick 솔루션 > Cluster > Node 메뉴에서 해당 노드를 선택한 후 “Maintenance mode”로 설정하거나 CLI 상에서 “kubectl drain [노드명]”을 수행하여, 해당 노드에서 동작중인 컨테이너들이 다른 (Master) 노드들로 마이그레이션한 후에 작업하시기 바랍니다.

해당 노드에서 기동되는 컨테이너 서비스들이 다른 노드로 마이그레이션되는 데 일정 시간이 소요됩니다. 이후 시스템 종료/재기동 작업을 진행하는 절차로 진행하면 됩니다.

CLI상에서 해당 노드를 drain하게 되면, 먼저 해당 서버는 컨테이너가 배포되지 않는 unschedulabe 상태로 전환되며, 그 이후 컨테이너(POD)들의 Eviction(축출)이 시작됩니다. 축출된 컨테이너들은 다른 노드들에서 기동이 됩니다. (단 마이그레이션이 가능한 조건이어야 합니다.)

참고로 daemonset의 경우 마이그레이션이 제외되며, Pod Disruption Policy 가 적용된 경우 다른 노드로 마이그레이션 되지 않을 수 있습니다.

- CLI 상에서 drain할 때, daemonset이나 각자가 고유한 스토리지 영역을 사용하는 StatefulSet 방식의 컨테이너 관련해서 경고메시지가 발생하며 진행되지 않을 수 있습니다. 이 경우 --delete-local-data 옵션과 -ignore-daemonsets 옵션을 사용하여 진행할 수 있습니다.

### 5.2.3 모니터링 및 로그 관리

#### 5.2.3.1 로그 모니터링

항목	명령어	비고
노드 서비스	journalctl -f -l -u	

로그 파일은 /var/log 디렉토리에 rotation되어 저장되므로 디렉토리 저장공간을 확인하여 주기적으로 정리하시기 바랍니다. /var/log 영역이 포함된 파일시스템을 주기적으로 모니터링하여 일정량 이상의 여유 공간을 확보하세요.

## 5.3 Private Docker Registry 관리절차

### 5.3.1 설치 및 환경설정 경로

디렉토리	설명	비고
/var/lib/docker	Docker Container 정보가 저장되는 디렉토리	
/registry	Private Docker Registry 디렉토리	

### 5.3.2 기동 및 종료 절차

#### 5.3.2.1 Private Registry 기동 절차

항목	세부 절차	명령어	비고
----	-------	-----	----

서비스 기동	로그인	Private Docker Registry 서버에 root 계정으로 로그인	
	Docker Registry 서비스	자동 기동 (systemctl start brick-registry)	
기동 여부 확인	로그인	Private Docker Registry 서버에 root 계정으로 로그인	
	Docker Registry 서비스	systemctl status brick-registry	

### 5.3.2.2 Private Registry 종료 절차

항목	세부 절차	명령어	비고
서비스 종료	로그인	Private Docker Registry 서버에 root 계정으로 로그인	
	Docker Registry 서비스	systemctl stop brick-registry	
기동 여부 확인	로그인	Private Docker Registry 서버에 root 계정으로 로그인	
	Docker Registry 서비스	systemctl status brick-registry	

### 5.3.3 모니터링 및 로그 관리

#### 5.3.3.1 로그 모니터링

항목	명령어	비고
Docker Registry 서비스	journalctl -f -l -u brick-registry	

로그 파일은 /var/log/ 디렉토리에 rotation되어 저장되며 디렉토리 저장공간을 확인하여 주기적으로 정리합니다

## 5.4 DNS 서비스 관리 절차

#### 5.4.1 설치 및 환경설정 경로

디렉토리	설명	비고
/etc/named.conf	Named 서버의 설정 정보	
/etc/named.rfc1912.zone	Named 서버의 Zone 설정 정보	
/var/named/<도메인명>.zone	Named 서버의 등록된 호스트정보	

#### 5.4.2 기동 및 종료 절차

##### 5.4.2.1 Named기동 절차

항목	세부 절차	명령어	비고
서비스 기동	로그인	DNS 서버에 root 계정으로 로그인	
	Named 서비스	systemctl start named	
기동 여부 확인	로그인	DNS 서버에 root 계정으로 로그인	
	Named 서비스	systemctl status named	

#### 5.4.2.2 Named 종료 절차

항목	세부 절차	명령어	비고
서비스 종료	로그인	DNS 서버에 root 계정으로 로그인	
	Named 서비스	systemctl stop named	
기동 여부 확인	로그인	DNS 서버'에 root 계정으로 로그인	
	Named 서비스	systemctl status named	

#### 5.4.3 모니터링 및 로그 관리

##### 5.4.3.1 로그 모니터링

항목	명령어	비고
DNS 서비스	journalctl -f -l -u named	

로그 파일은 /var/log/ 디렉토리에 rotation되어 저장되며 디렉토리 저장공간을 확인하여 주기적으로 정리합니다.

최의규 (Syper)

## 6 MIG GPU Configuration

### 6.1 MIG 기능 활성화 및 삭제

#### 6.1.1 MIG 기능 활성화

기본적으로 MIG 기능을 사용하기 위해서는 먼저 MIG 기능 활성화가 필요합니다.

```
# 전체 GPU MIG Enable
$ nvidia-smi -mig 1

# 특정 GPU MIG Enable
$ nvidia-smi -i [GPU ID] -mig 1
```

수행 후 Pending 상태로 재부팅이 필요하며, 재 부팅 후 nvidia-smi 명령어로 MIG Enable 상태 확인이 가능

#### 6.1.2 MIG 기능 비활성화

```
# 전체 GPU MIG Disable
$ nvidia-smi -mig 0

# 특정 GPU MIG Disable
$ nvidia-smi -i [GPU ID] -mig 0
```

단, MIG 기능을 비활성화 하려면, 사전에 DCI(Compute Instance) 및 DGI(GPU Instance)를 먼저 삭제한 후 MIG를 비활성화해야 합니다.

#### 6.1.3 nvidia-smi 옵션 설명

cgi : GPU Instance를 생성 (create)

cci : Compute Instance를 생성 (create)

dgi : GPU Instance를 삭제 (delete)

dci : Compute Instance를 삭제 (delete)

--default-compute-instance : 기본 Compute Instance 사용 (-C 옵션과 동일)

#### 6.1.4 MIG 예시

- 1) MIG Enable
 

```
$ nvidia-smi -i [GPU ID] -mig 0
```

## 2) GPU Instance 생성

\$ nvidia-smi mig -i [GPU ID] -cgi [Profile ID]

※ nvidia-smi mig -i [GPU ID] -lgip 명령으로 생성 가능한 GPU Instance Profile을 조회

## 3) Compute Instance 생성

\$ nvidia-smi mg -i [GPU ID] -gi [GPU Instance ID] -cci [Compute Profile ID]

※ nvidia-smi mig -i [GPU ID] -gi [GPU Instance ID] -lcip 명령으로 생성 가능한 Compute Instance Profile을 조회

※ GPU Slicing에 대해서는 아래 URL 참고

<https://docs.nvidia.com/datacenter/tesla/mig-user-guide/>

<https://kyumdoctor.tistory.com/130>

The table below shows the supported profiles on the A100-SXM4-40GB product. For A100-SXM4-80GB, the profile names will change according to the memory proportion - for example, 1g.10gb, 2g.20gb, 3g.40gb, 4g.40gb, 7g.80gb respectively.

**Table 7. GPU Instance Profiles on A100**

Profile Name	Fraction of Memory	Fraction of SMs	Hardware Units	L2 Cache Size	Number of Instances Available
MIG 1g.5gb	1/8	1/7	0 NVDECs /0 JPEG /0 OFA	1/8	7
MIG 1g.5gb+me	1/8	1/7	1 NVDEC /1 JPEG /1 OFA	1/8	1 (A single 1g profile can include media extensions)
MIG 2g.10gb	2/8	2/7	1 NVDECs /0 JPEG /0 OFA	2/8	3
MIG 3g.20gb	4/8	3/7	2 NVDECs /0 JPEG /0 OFA	4/8	2
MIG 4g.20gb	4/8	4/7	2 NVDECs /0 JPEG /0 OFA	4/8	1
MIG 7g.40gb	Full	7/7	5 NVDECs /1 JPEG /1 OFA	Full	1

**Note:** The 1g.5gb+me profile is only available starting with R470 drivers.

### 6.1.5 MIG 기능 자동 적용

MIG 활성화는 Permanent하게 적용이 되어 시스템 재기동 시에도 적용이 되나, GPU Slicing은 서버가 재기동되면 모두 초기화가 됩니다. 따라서 시스템 재기동 시에도 적용되도록 하기 위해서는 rc-local 서비스를 활용하거나 별도의 서비스를 만들어 시스템 기동 시 적용되도록 해야만 합니다.

아래는 rc-local이라는 시스템 서비스를 활용하는 방법입니다.

#### 6.1.5.1 /etc/rc.d/rc.local 에 실행할 내용 등록

```
$ vi /etc/rc.d/rc.local
...
# NVIDIA Mig Configuraion
# For further information, please visit the website below
# https://docs.nvidia.com/datacenter/tesla/mig-user-guide/#supported-profiles
# Slice 0,1 GPU to 40GB, 40GB
nvidia-smi mig -cgi 9,9 -i 0,1 -C

# Slice 2,3 GPU to 40GB, 20GB, 20GB
nvidia-smi mig -cgi 9,14,14 -i 2,3 -C
```

-C 옵션을 주지 않고 만든다면, -cci 옵션으로 다시 Compute Instance를 만들어 주는 작업이 필요합니다.

#### 6.1.5.2 rc-local 서비스 활성화

```
$ chmod +x /etc/rc.d/rc.local

// 서비스 확인
$ systemctl is-enabled rc-local.service
$ systemctl status rc-local.service

// 적용 테스트
$ systemctl enable rc-local.service
$ systemctl start rc-local.service
```

단, MIG 기능을 비활성화 하려면, 사전에 DCI(Compute Instance) 및 DGI(GPU Instance)를 먼저 삭제한 후 MIG를 비활성화해야 합니다.

## 7 Image 사용 방법

### 7.1 Docker Image 개념

- Docker는 가장 범용적이고 폭넓게 사용되는 컨테이너 런타임이다.
- 기본적으로 인터넷에서 공식적으로 배포되는 docker 컨테이너 이미지를 사용할 수 있으나, 일반적으로 기업에서 사용하는 어플리케이션은 기존 인터넷에 공개된 공식 이미지를 커스터마이징하거나 새롭게 만드는 등 별도의 이미지 빌드가 필요하다. 공개된 이미지는 hub.docker.com에서 확인하거나 docker search 커맨드를 사용해 확인할 수 있다.

### 7.2 Docker Image 검색

- “STARS” 가 많은 이미지는 그만큼 많은 사용자에게서 검증이 된 것을 의미하며, “OFFICIAL”에 OK 표시가 있는 이미지는 해당 어플리케이션 업체 또는 Docker 사이트에서 공식적으로 인증받은 공식 이미지를 의미한다. 참고로 자세한 버전 정보는 hub.docker.com 웹사이트에서 검색해야 한다.
- Docker image 이름의 규칙은 repository:tag 형식으로 되어 있다. 예를 들어, 이미지 이름이 pivotaldata/centos-gpdb-dev:1.0 이라고 한다면, 앞 부분 pivotaldata/centos-gpdb-dev는 repository이며, 1.0이 tag에 해당한다. repository부분에 단순히 어플리케이션명만 들어가 있는 경우에는 앞에 docker.io가 생략되어 있는 것으로 생각하면 된다.

```
$ docker search [문자열]
```

```
$ docker search centos
```

NAME	DESCRIPTION	STARS	OFFICIAL
AUTOMATED			
centos	The official build of CentOS.	6232	[OK]
ansible/centos7-ansible	Ansible on Centos7	132	
[OK]			
consol/centos-xfce-vnc	Centos container with "headless" VNC session...	122	
[OK]			
jdeathe/centos-ssh	OpenSSH / Supervisor / EPEL/IUS/SCL Repos - ...	115	
[OK]			
centos/systemd	systemd enabled base container.	86	
[OK]			
...			

### 7.3 Docker Image 다운로드

- Docker Image를 인터넷으로부터 다운로드할 수 있다. 자세한 버전 정보는 Docker 사이트에서 확인할 수 있으며, 다운로드할 때, 뒤에 TAG를 붙이지 않으면 기본적으로 가장 최신의 latest 태그 이미지를 다운로드 한다.

```
$ docker pull [문자열]
```

```
$ docker pull centos
```

```
Using default tag: latest
```

```
latest: Pulling from library/centos
3c72a8ed6814: Pull complete
Digest: sha256:76d24f3ba3317fa945743bb3746fbaf3a0b752f10b10376960de01da70685fb
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest...
```

\$ docker images		TAG	IMAGE ID
REPOSITORY	SIZE		
CREATED			
centos	203MB	centos7	7e6257c9f8d8 2 months
centos	215MB	latest	0d120b6ccaa8 2 months

- 현재 로컬 서버에 다운로드되어 저장된 이미지는 docker images 명령으로 확인할 수 있다.

## 7.4 Docker Image 업로드

- 다운로드하거나 새롭게 빌드한 Docker Image를 AI 클라우드의 Private Docker Registry에 업로드할 수 있다. 업로드하기 위해서는 Docker Image의 이름의 repository로 적절하게 변경해주어야 인식이 된다. 그렇지 않으면 docker hub (docker.io) 사이트로 업로드를 시도하게 된다.
- 아래와 같이 docker image의 이름을 적절하게 변경하고, 업로드한다.

\$ docker images		TAG	IMAGE ID
REPOSITORY	SIZE		
CREATED			
centos	203MB	centos7	7e6257c9f8d8 2 months
centos	215MB	latest	0d120b6ccaa8 2 months

**\$ docker tag centos:latest registry.poscoict.com/hub/centos:latest**

```
$ docker push registry.poscoict.com/hub/centos:latest
The push refers to repository [registry.poscoict.com/hub/centos]
291f6e44771a: Pushed
latest: digest: sha256:fc4a234b91cc4b542bac8a6ad23b2ddcee60ae68fc4dbd4a52efb5f1b0baad71
size: 529
```

- Private Docker Registry에 웹으로 접속하여 정상적으로 업로드가 되었는지 확인할 수 있다. 로그인하여 hub 프로젝트에서 방금 업로드한 centos 이미지를 확인할 수 있다.

Harbor

admin  
\*\*\*\*\*  
 Remember me [Forgot password](#)

[LOG IN](#)

More info...

Projects hub System Admin

Summary Repositories Helm Charts Members Labels Logs Robot Accounts Tag Retention Webhooks Configuration

Name	Tags	Pulls
hub/anis-python	2	147
hub/anis-r	1	84
hub/bats	1	0
hub/busybox	1	3
hub/centos	1	0
hub/centos7	1	0
hub/code-server	1	4
hub/configurable-http-proxy	1	0
hub/dduportal/bats	1	0
hub/edit-task	1	49
hub/egi2-analysis	2	24
hub/egi2-dbmanager	2	31

PUSH IMAGE  EVENT

## 7.5 표준 빌드이미지 설명 (예시)

```
# Copyright (c) Cloudbub.

# Ubuntu 18.04 (bionic)
# https://hub.docker.com/_/ubuntu/?tab=tags&name=bionic
# Reference : https://hub.docker.com/r/jupyter/base-notebook/dockerfile
# OS/ARCH: linux/amd64

##### Configuration #####
# Base Container Image
FROM nvidia/cuda:10.2-cudnn7-devel-ubuntu18.04 as stage-ml

# Python Version Configuration(Anaconda, Miniconda)
#ARG ANACONDA_VERSION=5.1.0
#ARG MINICONDA_VERSION=4.5.4
#ARG CONDA_VERSION=4.5.4
#ARG PYTHON_VERSION=3.6.9

# Tensorflow version config - comment below if you don't want to install tensorflow
#ARG TENSORFLOW_ENABLE="yes"
```

```

ARG TENSORFLOW_VERSION=1.15.4

# Pytorch version config - comment below if you don't want to install pytorch
ARG PYTORCH_ENABLE="yes"

##### Configuration #####
LABEL maintainer="cloudhub <support@brickcloud.co.kr>"
ARG NB_USER="poscoict"
ARG NB_UID="1000"
ARG NB_GID="100"

# Fix DL4006
SHELL ["/bin/bash", "-o", "pipefail", "-c"]

USER root

# Install all OS dependencies for notebook server that starts but lacks all
# features (e.g., download as all possible file formats)
ENV DEBIAN_FRONTEND noninteractive

# For NVIDIA Repository Certificates Issues
RUN echo "deb [by-hash=no] http://developer.download.nvidia.com/compute/cuda/repos/ubuntu180
4/x86_64 /" > /etc/apt/sources.list.d/cuda.list && \
    echo "deb [by-hash=no] http://developer.download.nvidia.com/compute/machine-learning/repos/u
buntu1804/x86_64 /" > /etc/apt/sources.list.d/nvidia-ml.list

RUN apt-get update \
&& apt-get install -yq --no-install-recommends \
    apt-utils \
    build-essential \
    wget \
    bzip2 \
    ca-certificates \
    vim \
    tzdata \
    supervisor \
    openssh-server \
    gnupg \
    git \
    sudo \
    locales \
    fonts-liberation \
    pandoc \
    run-one \
&& apt-get clean && rm -rf /var/lib/apt/lists/* \
&& echo "en_US.UTF-8 UTF-8" > /etc/locale.gen && locale-gen

# Configure environment
ENV CONDA_DIR=/opt/conda \
    SHELL=/bin/bash \
    NB_USER=$NB_USER \
    NB_UID=$NB_UID \
    NB_GID=$NB_GID \
    LC_ALL=en_US.UTF-8 \
    LANG=en_US.UTF-8 \
    LANGUAGE=en_US.UTF-8
ENV PATH=$CONDA_DIR/bin:$PATH \
    HOME=/home/$NB_USER \

```

```

TZ=Asia/Seoul

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime

# Copy a script that we will use to correct permissions after running certain commands
COPY fix-permissions /usr/local/bin/fix-permissions
RUN chmod a+rx /usr/local/bin/fix-permissions

# Enable prompt color in the skeleton .bashrc before creating the default NB_USER
RUN sed -i 's/^#force_color_prompt=yes/force_color_prompt=yes/' /etc/skel/.bashrc

# Create NB_USER wtih name jovyhan user with UID=1000 and in the 'users' group
# and make sure these dirs are writable by the 'users' group.
RUN echo "auth requisite pam_deny.so" >> /etc/pam.d/su && \
    sed -i.bak -e 's/^%admin/#%admin/' /etc/sudoers && \
    sed -i.bak -e 's/^%sudo/#%sudo/' /etc/sudoers && \
    useradd -m -s /bin/bash -N -u $NB_UID $NB_USER && \
    usermod -aG sudo $NB_USER && \
    echo "%sudo ALL=NOPASSWD: ALL" >> /etc/sudoers && \
    mkdir -p $CONDA_DIR && \
    chown $NB_USER:$NB_GID $CONDA_DIR && \
    chmod g+w /etc/passwd && \
    fix-permissions $HOME && \
    fix-permissions $CONDA_DIR

USER $NB_UID
WORKDIR $HOME

# Setup work directory for backward-compatibility
RUN mkdir /home/$NB_USER/work && \
    fix-permissions /home/$NB_USER

ENV MINICONDA_VERSION=$MINICONDA_VERSION \
    CONDA_VERSION=$CONDA_VERSION \
    PYTHON_VERSION=$PYTHON_VERSION

# Install Anaconda/Miniconda
WORKDIR /tmp
RUN wget --quiet https://repo.anaconda.com/miniconda/Miniconda3-$MINICONDA_VERSION \
    -Linux-x86_64.sh && \
    /bin/bash Miniconda3-$MINICONDA_VERSION-Linux-x86_64.sh -f -b -p $CONDA_DIR & \
& \
    rm -f /tmp/Miniconda3-$MINICONDA_VERSION-Linux-x86_64.sh && \
    echo "conda ${CONDA_VERSION}" >> $CONDA_DIR/conda-meta/pinned && \
    conda config --set ssl_verify false && \
    conda config --system --prepend channels conda-forge && \
    conda config --system --set auto_update_conda false && \
    conda config --system --set show_channel_urls true && \
    conda update -y -n base -c defaults conda && \
    if [ ! $PYTHON_VERSION = 'default' ]; then conda install --yes python=$PYTHON_VERSION; \
fi && \
    conda list python | grep '^python' | tr -s '' | cut -d '.' -f 1,2 | sed 's/$.*/' >> $CONDA_DIR/conda- \
meta/pinned && \
    #conda install --quiet --yes conda && \
    conda install --quiet --yes pip && \
    #conda update --all --quiet --yes && \
    conda clean --all --yes && \
    rm -rf /home/$NB_USER/.cache/yarn /tmp/* && \
    fix-permissions $CONDA_DIR && \

```

```

fix-permissions /home/$NB_USER

# Install Tini
RUN conda install --quiet --yes 'tini=0.18.0' && \
    conda list tini | grep tini | tr -s '' | cut -d '' -f 1,2 >> $CONDA_DIR/conda-meta/pinned && \
    conda clean --all --yes && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

# Install Jupyter-Notebook
RUN pip install --no-cache-dir \
    notebook==5.4.0 \
    jupyterlab=='0.35.6' && \
    jupyter notebook --generate-config && \
    rm -rf /home/$NB_USER/.cache && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

# Install Jupyterhub
RUN conda install -c conda-forge --quiet --yes \
    jupyterhub=2.2.1 && \
    rm -rf /home/$NB_USER/.cache/yarn && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

# Install Tensorflow
RUN pip install --no-cache-dir --upgrade pip setuptools requests && \
    conda config --set ssl_verify false && \
    conda install --quiet --yes tensorflow=1.15.0 && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

# Install Pytorch
RUN if [ "$PYTORCH_ENABLE" = "yes" ]; then \
    conda config --set ssl_verify false && \
    conda install --quiet --yes pytorch==1.4.0 torchvision==0.5.0 cudatoolkit=10.1 -c pytorch && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER; \
fi

# Install Other packages
RUN conda config --set ssl_verify false && \
    conda install --quiet --yes \
    pillow \
    pandas \
    scikit-learn \
    matplotlib && \
    # to resolve jupyter notebook print bug
    conda install --quiet --yes tornado=5.1.1 && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

EXPOSE 8888 22

# Configure container startup

```

```

ENTRYPOINT ["tini", "-g", "--"]
CMD ["start-notebook.sh"]

# Copy local files as late as possible to avoid cache busting
COPY start.sh start-notebook.sh start-singleuser.sh /usr/local/bin/
COPY jupyter_notebook_config.py /etc/jupyter/

# Fix permissions on /etc/jupyter as root
USER root
RUN fix-permissions /etc/jupyter/

# Switch back to jovyan to avoid accidental container runs as root
USER $NB_UID

WORKDIR $HOME

FROM stage-ml as stage-ide

ARG PYCHARM_VERSION=2021.3.2

USER root

RUN wget --no-check-certificate https://download.jetbrains.com/python/pycharm-community-${PYCHARM_VERSION}.tar.gz && \
    tar -xvf pycharm-community-${PYCHARM_VERSION}.tar.gz -C /opt && \
    ln -s /opt/pycharm-community-${PYCHARM_VERSION} /opt/pycharm && \
    rm -f pycharm-community-${PYCHARM_VERSION}.tar.gz

FROM stage-ide as stage-display

ENV NO_VNC_HOME=/usr/share/novnc

USER root

# Install xfce
RUN apt-get update \
# && apt-get install -yq --no-install-recommends \
&& apt-get install -yq \
    xfce4 \
    xfce4-terminal \
    mousepad \
    net-tools \
    curl \
    gettext \
    libnss-wrapper \
    && apt-get purge -y pm-utils xscreensaver* \
    && apt-get clean && rm -rf /var/lib/apt/lists/*

# Install vnc tools
RUN apt-get update \
    && apt-get install -yq \
        python-numpy \
        && wget -qO- https://sourceforge.net/projects/tigervnc/files/stable/1.10.1/tigervnc-1.10.1.x86_64.tar.gz | tar xz --strip 1 -C / && \
        mkdir -p ${NO_VNC_HOME}/utils/websockify && \
        wget -qO- https://github.com/novnc/noVNC/archive/v1.2.0.tar.gz | tar xz --strip 1 -C ${NO_VNC_HOME} && \

```

```

wget -qO- https://github.com/novnc/websockify/archive/v0.9.0.tar.gz | tar xz --strip 1 -C ${NO_VNC_HOME}/utils/websockify && \
chmod +x ${NO_VNC_HOME}/utils/*.sh && \
rm -rf /var/lib/apt/lists/*
FROM stage-display as stage-final

ARG VNC_BLACKLIST_THRESHOLD
ARG VNC_BLACKLIST_TIMEOUT
ARG VNC_PW
ARG VNC_RESOLUTION
ARG NO_VNC_HOME
ARG STARTUPDIR

ENV \
    DISPLAY=:1 \
    NO_VNC_PORT="6901" \
    VNC_BLACKLIST_THRESHOLD=${VNC_BLACKLIST_THRESHOLD:-20} \
    VNC_BLACKLIST_TIMEOUT=${VNC_BLACKLIST_TIMEOUT:-0} \
    VNC_COL_DEPTH=24 \
    VNC_PORT="5901" \
    VNC_PW=${VNC_PW:-123456} \
    VNC_RESOLUTION=${VNC_RESOLUTION:-1360x768} \
    VNC_STARTUPDIR=${HOME}/vnc_startup \
    VNC_CONFDIR=${HOME}/.vnc

ADD --chown=$NB_UID:$NB_GID startup.sh ${VNC_STARTUPDIR}/
ADD --chown=$NB_UID:$NB_GID Desktop/ ${HOME}/Desktop/
ADD --chown=$NB_UID:$NB_GID test ${HOME}/work/test
ADD --chown=$NB_UID:$NB_GID xfce4-config/ ${HOME}/.config/xfce4/

RUN chown -R ${NB_UID}:${NB_GID} ${HOME}

USER root

WORKDIR ${HOME}

EXPOSE ${VNC_PORT} ${NO_VNC_PORT}

CMD [ "supervisord", '-n" ]

```

- ARG: Dockerfile 작성 시에 자주 사용하는 이름을 변수로 등록하여 사용하기 위해 설정
- FROM: 빌드에 사용할 기본 베이스 이미지를 지정하는 부분이다.
- LABEL: 생성하는 image에 Label을 추가하는 부분. 예를 들면, 컨테이너 이미지에 대한 maintainer 같은 레이블 정보를 설정할 수 있으며, 간혹 다른 용도로 사용하기도 한다. (기동 시에 특정 레이블을 조회하여 라이브러리를 기동할 수 있도록 확인하는 용도)
- USER : 빌드하는 시점에서의 사용자를 선택하는 부분이다. 예를 들어, 패키지 설치 등은 root로 실행해야 하고, 특정 application은 보안을 위해 특정한 사용자로 실행할 수 있도록 한다.

- RUN : 이미지를 빌드할 때 실행해야 할 명령은 RUN으로 처리. Docker 이미지는 명령 하나가 하나의 레이어가 되며, 이것이 지속적으로 쌓여서 생성이 되기 때문에 생성 이후 불필요한 파일들이 유지되면 이미지의 용량이 커지게 된다. 이를 위해서 RUN 명령으로 패키지 설치를 한 이후에는 같은 명령 줄 라인에 남은 파일을 지워주는 작업도 같이 진행해줘야 한다.
- ENV : 컨테이너가 실행이 될 때, 컨테이너에 환경변수를 적용
- COPY : COPY는 이미지를 빌드할 때, 외부의 파일을 컨테이너로 복사.삽입하기 위해 사용
- ADD: COPY와 동일하지만, 좀 더 확장된 기능을 가지고 있다. (COPY를 개선하기 위해 나옴)
- WORKDIR : 이미지를 빌드할 때 작업 디렉토리를 지정할 수 있다. 특정 위치에서 어떤 작업이 진행되어야 하는 경우 사용
- EXPOSE : 컨테이너가 기동될 때 외부로 오픈해야 할 PORT를 정의하는 것으로, 단지 선언적인 의미이다. 이미지를 확인해서 해당 이미지가 어떤 포트를 사용하는지 나타내기 위해 사용하며, 실제 외부로 포트를 오픈하는 것은 별도의 옵션 사용이 필요하다.
- ENTRYPOINT : 컨테이너를 기동할 때, 반드시 실행되는 명령으로, 이미지에 해당 옵션이 사용되면 무조건 해당 명령이 기본적으로 처음에 실행이 된다. 명령이 고정적인 경우에 사용. CMD와 달리 override할 수 없음. Kubernetes에서는 command가 동일한 역할
- CMD 컨테이너를 기동할 때 실행되는 명령이나, 명령줄에서 재지정하여 override할 수 있으며, ENTRYPOINT가 지정되어 있는 경우, ENTRYPOINT 이후 인수처럼 실행이 된다.Kubernetes에서는 args가 동일한 역할

ICT

최익규 (Syper)

## 8 Kubernetes Command

### 8.1 Persistence Volume 관리

#### 8.1.1 PV(Persistence Volume) 조회

PV(Persistence Volume) 리스트를 조회합니다.

```
$ kubectl get pv
```

#### 8.1.2 PV(Persistence Volume) 등록

PV(Persistence Volume)을 등록합니다.

```
$ kubectl create -f pv-xxx.yaml
```

```
-----
{
  "apiVersion": "v1",
  "kind": "PersistentVolume",
  "metadata": {
    "name": "{PV명}"
  },
  "spec": {
    "capacity": {
      "storage": "100Gi"
    },
    "accessModes": ["ReadWriteMany"],
    "nfs": {
      "path": "<nfs-mount-path>",
      "server": "<nfs-server>"
    }
  }
}
```

#### Access Modes

CONDITION	CLI	DESCRIPTION
ReadWriteOnce	RWO	Mount Volume에 대해서 하나의 POD에서 read/write
ReadOnlyMany	ROX	Mount Volume에 대해서 다수의 POD에서 read
ReadWriteMany	RWX	Mount Volume에 대해서 다수의 POD에서 read/write

### 8.1.3 PV(Persistence Volume)제거

프로젝트의 POD빌드정보를 조회합니다.

```
$ kubectl delete pv <pv-name>
```

## 8.2 Persistence Volume Claim관리

### 8.2.1 PVC(Persistence Volume Claim)조회

프로젝트PVC(Persistence Volume Claim)리스트를 조회합니다.

```
$ kubectl get pvc -n <project-name>
```

### 8.2.2 PVC(Persistence Volume Claim)등록

프로젝트 PVC(Persistence Volume Claim)을 등록합니다..

```
$ kubectl create -f pvc-xxx.yaml
-----
{
  "apiVersion": "v1",
  "kind": "PersistentVolumeClaim",
  "metadata": {
    "name": "{PVC명}"
  },
  "spec": {
    "accessModes": ["ReadWriteMany"],
    "resources": {
      "requests": {
        "storage": "100Gi"
      }
    }
  }
}
```

## 8.3 POD관리

### 8.3.1 POD조회

프로젝트의 POD리스트를 조회합니다.

```
$ kubectl get pod -n <project-name>
```

### 8.3.2 POD상세조회

프로젝트의 POD상세정보를 조회합니다.

```
$ kubectl describe pod <pod-name> -n <project-name>
```

### 8.3.3 POD로그조회

프로젝트의 POD의 로그를 조회합니다.

```
$ kubectl logs <pod-name> -n <project-name>
```

### 8.3.4 POD로그 Watch

프로젝트의 POD의 로그를 watch설정하여 지속적으로 조회합니다.

```
# POD로그 watch(tail)
$ kubectl logs -f <pod-name> -n <project-name>

# POD로그 watch
$ kubectl logs -f <pod-name> --tail=500 -n <project-name>
```

### 8.3.5 POD로그저장

프로젝트의 POD의 로그를 로컬시스템에 저장합니다.

```
$ kubectl logs <pod-name> -n <project-name> > xxxxx.log
```

### 8.3.6 POD원격접근

프로젝트의 POD배포정보를 조회합니다.

```
$ kubectl -exec -ti <pod-name> -n <project-name> -- sh
```

### 8.3.7 POD Replica

프로젝트의 POD의 개수를 Manual로 조정합니다.

```
$ kubectl scale --replicas=<pod-replicas-number> dc <project-dc-name> -n <project-name>
```

### 8.3.8 Executing Remote Command

프로젝트의 POD에 shell명령어를 수행합니다.

```
$ kubectl exec <pod-name> -- <shell-command>
```

## 9 Docker Command

### 9.1 Docker 명령어

#### 9.1.1 Docker search 및 image 내려받기

##### Docker search 및 image 내려받기

1. Docker Hub를 통한 이미지 검색

- 명령어 : docker search [이미지명]  
ex) docker search ubuntu

2. Docker image 내려받기

- 명령어 : docker pull [이미지명]:[태그]  
ex) docker pull ubuntu:lastest

#### 9.1.2 Docker image 목록 조회

##### Docker image 목록 조회

1. Docker 이미지 목록 조회

- 명령어 : docker images

#### 9.1.3 Docker image 컨테이너에 올리기

##### Docker image 컨테이너에 올리기

1. Docker image 컨테이너 Run

- 명령어 : docker run [옵션] [이미지 이름] [실행할 파일]  
ex) docker run -i -t --name hello ubuntu /bin/bash

#### 9.1.4 Docker 내 모든 컨테이너 목록 출력

##### Docker 내 모든 컨테이너 목록 출력

1. Docker에서 실행 중인 컨테이너 보기

- 명령어 : docker ps

## 2. Docker 전체 컨테이너 목록 출력

- 명령어 : docker ps -a

### 9.1.5 Docker 컨테이너 Start / Stop / Restart / Kill

#### Docker 컨테이너 Start / Stop / Restart / Kill

##### 1. Docker 컨테이너 시작

- 명령어 : docker start [컨테이너명]

##### 2. Docker 컨테이너 정지

- 명령어 : docker stop [컨테이너명]

##### 3. Docker 컨테이너 재시작

- 명령어 : docker restart [컨테이너명]

##### 4. Docker 컨테이너 강제종료

- 명령어 : docker kill [컨테이너명]

### 9.1.6 Docker 컨테이너 및 이미지 삭제

#### Docker 컨테이너 및 이미지 삭제

##### 1. Docker 컨테이너 삭제

- 명령어 : docker rm [컨테이너명]

##### 2. Docker 컨테이너 이미지 삭제

- 명령어 : docker rmi [컨테이너명]

### 9.1.7 Docker 컨테이너 내 명령어 실행

#### Docker 컨테이너 내 명령어 실행

##### 1. Docker 컨테이너 내 명령어 실행

- 명령어 : docker exec [컨테이너명] [명령어] [매개변수]  
 ex) docker exec e2af61 echo "portal"

## 2. Docker 컨테이너 내 원격 접속

- 명령어 : docker exec -it [컨테이너명]  
 ex) docker exec -it e2af61

### 9.1.8 Docker 새로운 이미지 생성

#### Docker 새로운 이미지 생성

##### 1. Docker 컨테이너 내 접속 변경사항 적용

##### 2. Docker 이미지 레지스터에 적용

- 명령어 : docker commit [컨테이너명]:[태그]  
 ex) docker commit e2af61 ubuntu:git

### 9.1.9 Docker 부모 이미지와 현재 컨테이너 간 비교

#### Docker 부모 이미지와 현재 컨테이너 간 비교

##### 1. Docker 부모 이미지와 현재 컨테이너 간 비교

- 명령어 : docker diff [컨테이너명]  
 ex) docker diff e2af61

### 9.1.10 Docker 컨테이너 상태 정보 확인

#### Docker 컨테이너 상태 정보 확인

##### 1. Docker 컨테이너 상태 정보 확인

- 명령어 : docker stats [컨테이너명]  
 ex) docker stats e2af61

### 9.1.11 Docker 컨테이너 내 실행 중인 프로세스 보기

#### Docker 컨테이너 내 실행 중인 프로세스 보기

### 1. Docker 컨테이너 내 실행 중인 프로세스 보기

- 명령어 : docker top [컨테이너명]

ex) docker top e2af61

### 9.1.12 Docker 컨테이너 또는 이미지 low-level 정보 보기

#### Docker 컨테이너 또는 이미지 low-level 정보 보기

### 1. Docker 컨테이너 또는 이미지 low-level 정보 보기

- 명령어 : docker inspect [컨테이너명]

ex) docker inspect e2af61

### 9.1.13 Docker tar 파일 이미지 로드 방법

#### Docker tar 파일 이미지 로드 방법

### 1. Docker tar 파일 이미지 로드 방법

- 명령어 : docker load -i [tar파일]

ex) docker load -i XXX.tar

### 9.1.14 Docker 레파지토리 내 이미지 태그

#### Docker 레파지토리 내 이미지 태그

### 1. Docker 레파지토리 내 이미지 태그

- 명령어 : docker tag [이미지명:[태그명]] [레지스트리호스트/] [사용자명/] [이미지명[:태그명]]

ex) docker tag registry.access.redhat.com:5000/0000솔루션3/XXX:v0.1

pasregistry.lotte.cloud:5000/0000솔루션3/XXX:v0.1

### 9.1.15 Docker 이미지 또는 레지스트리에서 레파지토리로 푸쉬

#### Docker 이미지 또는 레지스트리에서 레파지토리로 푸쉬

1. Docker 이미지 또는 레지스트리에서 레파지토리로 푸쉬
  - 명령어 : docker push [옵션] [이미지명[:태그명]]
  - ex) docker push pasregistry.lotte.cloud:5000/0000솔루션3/XXX:v0.1

### 9.1.16 Dockerfile로 이미지 생성 및 어플리케이션 실행하기

#### Dockerfile로 이미지 생성 및 어플리케이션 실행하기

명령어 내용

FROM: 원본 Docker Image를 지정

MAINTAINER: 작성자 정보

RUN: 명령어 실행

ADD: File, Directory 추가

CMD: Container의 실행 명령어 1

ENTRYPOINT: Container의 실행 명령어 2

WORKDIR: 작업 Directory 지정

ENV: 환경변수 지정

USER: 실행 User 지정

EXPOSE: Port export

VOLUME: Volume mount

#### 1. Dockerfile로 이미지 생성 샘플

# FROM은 어떤 이미지로부터 새로운 이미지를 생성할지를 지정

```
FROM registry.access.redhat.com/0000솔루션/jboss-eap64-0000솔루션
```

# MAINTAINER는 이 Dockerfile을 생성-관리하는 사람을 입력

```
MAINTAINER LDCC <admin@l-cloud.co.kr>
```

# RUN은 직접 쉘 명령어를 실행하는 명령어

```
RUN yum -y update
```

# 환경변수 설정

```
ENV JAVA_HOME /usr/lib/jvm/java-1.8.0
```

# EXPOSE는 컨테이너에 오픈할 포트를 지정.

```
EXPOSE 8080
```

```
# CMD는 컨테이너에서 실행될 명령어를 지정
```

```
CMD ["/opt/eap/bin/0000솔루션-launch.sh"]
```

## 2. Dockerfile을 빌드 샘플

docker build 명령어는 -t 플래그를 사용해 이름과 태그를 지정하며 마지막은 빌드 대상 디렉토리를 가리킨다

```
$ docker build -t pasregistry.lotte.cloud:5000/0000솔루션/jboss-eap64-0000솔루션-ldcc .
```

## 3. Dockerfile을 빌드 후 실행

```
docker run -ti --name <container-name> --user <container-user> pasregistry.lotte.cloud:5000 /0000솔루션/jboss-eap64-0000솔루션-ldcc
```

### 9.1.17 Docker 명령어 요약

#### Docker 명령어 요약

##### 실행 중인 컨테이너 보기

- docker ps

##### 전체 컨테이너 보기

- docker ps -a

##### 새 컨테이너 실행

- docker run [options] <image> [command]

##### 중지된 컨테이너 실행

- docker start [options] <container>

##### 컨테이너 재시작

- docker restart [options] <container>

##### 컨테이너 중지

- docker stop [options] <container>

### 컨테이너 강제 중지

- docker kill [options] <container>

### 컨테이너 삭제

- docker rm [options] <container>

### 컨테이너 / 이미지 상세 정보 조회

- docker inspect [options] <container|image>

### 컨테이너 로그 확인

- docker logs [options] <container>

### 컨테이너 프로세스 확인

- docker top [options] <container>

### 컨테이너 파일시스템 변경사항 보기

- docker diff [options] <container>

### 실행 가능한 이미지 목록 보기

- docker images

### docker 허브에서 이미지 검색

- docker search [options] <term>

### docker 허브에서 이미지 내려 받기

- docker pull [options] <image>

### docker 허브에 이미지 올리기

- docker push [options] <image>

### Dockerfile로 부터 이미지 만들기

- docker build [options] <path>

### 컨테이너로 부터 백업 이미지 만들기

- docker commit [options] <container> <image>

### 이미지 삭제

- docker rmi [options] <image>

### 컨테이너 내 명령어 실행

- docker exec <container> [command]

컨테이너 원격 접속

- docker exec -it <container> /bin/bash



최의규 (syper)

2022-05-24 09:51:29