


## 2차년도 주요 결과물

(과제명) 대규모 분산 에너지 저장장치 인프라의 안전한 자율운영  
및 성능 평가를 위한 지능형 SW 프레임워크 개발  
(과제번호) 2021-0-00077

- 결과물명 : 에너지 인프라 기반 SW 프레임워크 스켈레톤(SW)
- 작성일자 : 2022년 12월 1일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업  
“2차년도 주요 결과물” 로 제출합니다.

수행기관	성명/직위	확인
한국전자기술연구원	최효섭/책임연구원	

정보통신기획평가원장 귀하

## 사 용 권 한

본 문서에 대한 서명은 한국전자기술연구원 내부에서 본 문서에 대하여  
수행 및 유지관리의 책임이 있음을 인정하는 것임.

본 문서는 작성, 검토, 승인하여 승인된 원본을 보관한다.

---

작성자 :	박진원	일자 :	2022. 12. 01
-------	-----	------	--------------

---

검토자 :	김창우	일자 :	2022. 12. 02
-------	-----	------	--------------

---

승인자 :	최효섭	일자 :	2022. 12. 03
-------	-----	------	--------------

## 제 · 개정 이력

버전	변경일자	제·개정 내용	작성자
1.0	2022-12-01	최초 등록	박진원

## 목 차

1. 개요 .....	1
2. 안전 SW프레임워크 설계 .....	3
3. Kubernetes 및 Kubeflow 구축 .....	5
4. Kubeflow를 활용한 Pipeline 설계 .....	10

## 1. 개요

### □ 목적

- 본 명세서의 목적은 대규모 분산에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 “지능형 안전SW 프레임워크”를 개발하기 위해 프레임워크의 개요 및 개발 방법을 기록한다.

### □ 범위

- 본 설계서는 “지능형 안전SW 프레임워크”의 개발에 필요한 아키텍처 설계와 구현방법을 중심으로 설명한다.

### □ 시스템 개요

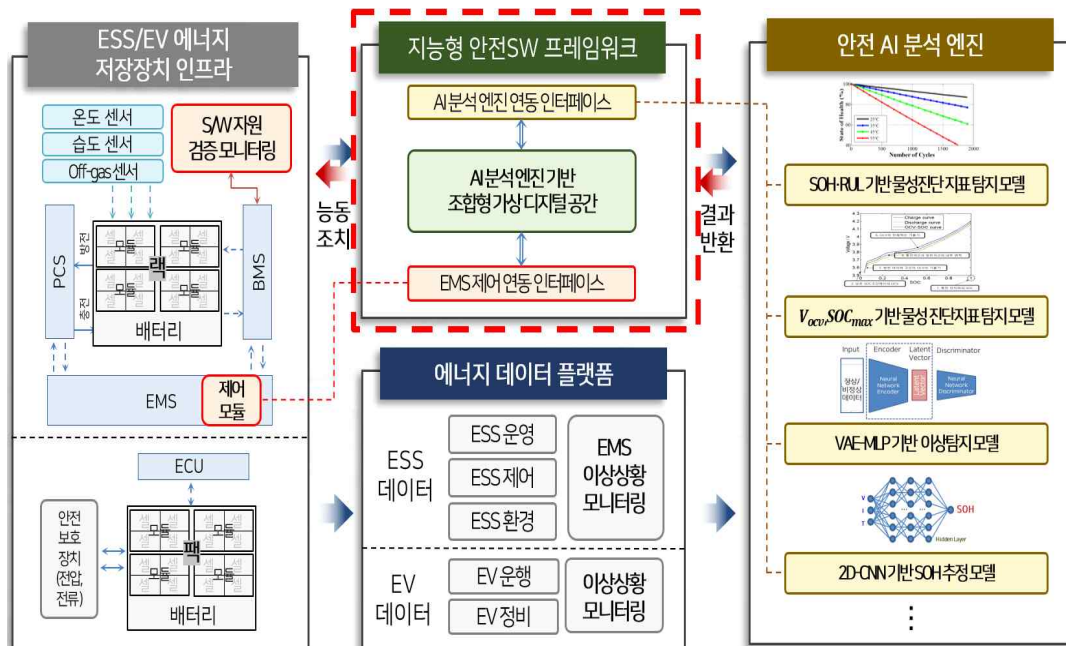


그림 1 대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 구조

- 지능형 안전SW 프레임워크는 개발 솔루션을 실제 ESS 운영 사이트와 가상 디지털 공간에 실증하여 안전 소프트웨어를 구조화하고, 에너지 인프라 운영 소프트웨어를 최적화한다. 안전SW 솔루션 활용의 손쉬운 확장 및 배포를 위해 솔루션 컨테이너 패키지를 통해 프레임워크 사용자로 하여금 솔루션을 선택 활용하도록 개발하고자 한다.

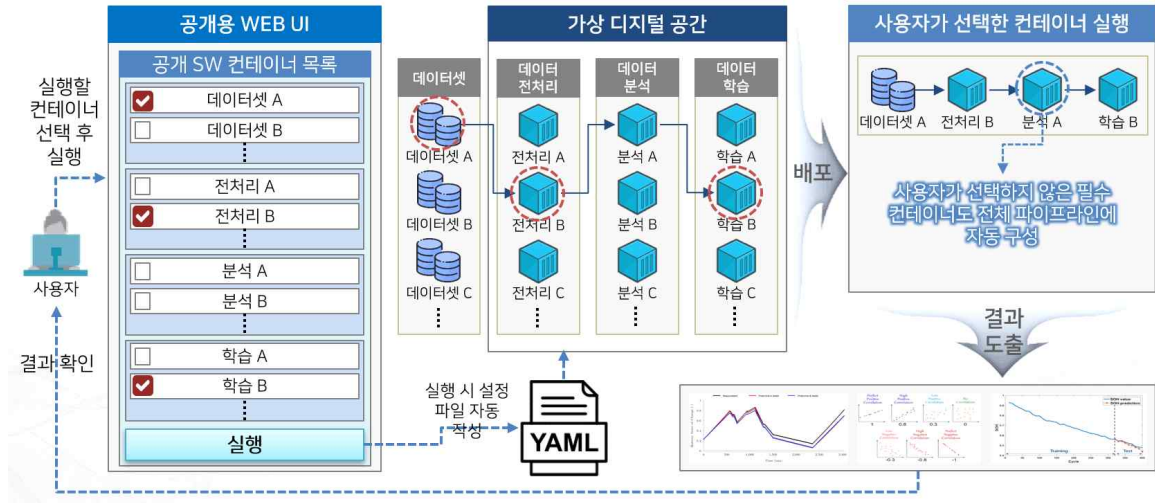



그림 2 안전 소프트웨어 솔루션의 컨테이너 기반 패키지 제공 구조

- ESS 시스템에서 나오는 데이터의 전처리부터 안전에 관한 이상감지, 진단, 예측 등을 수행하며 지표를 생성하는 분석 엔진들을 각각의 컨테이너로 개발하며, ESS 데이터를 가지고 있는 연구개발자는 “지능형 안전SW 프레임워크”를 통해 분석 모델들을 가상 공간에서 실행시킬 수 있다.

## □ 관련 계획 및 표준

- 본 설계서는 아래 계획 및 표준을 참고한다.

구분	식별자	세부 내용	설명
설계서	ISO/IEC 9126	9126-1 (품질 모델) 9126-2 (외부 품질) 9126-3 (내부 품질) 9126-4 (사용 품질)	품질 특성 및 측정기준을 제시 소프트웨어의 기능성, 신뢰성, 사용성, 효율성, 유지보수 용이성, 이식성
	ISO/IEC 14598	14598-1 (개요) 14598-2 (계획과 관리) 14598-3 (개발자용 프로세스) 14598-4 (구매자용 프로세스) 14598-5 (평가자용 프로세스) 14598-6 (평가 모듈)	ISO 9126에 따른 제품 평가 표준: 반복성, 공정성, 객관성, 재생산성
	ISO/IEC 12119	소프트웨어 패키지 -제품설명서 -사용자문서 -프로그램과 데이터	패키지 SW 품질 요구사항 및 테스트
계획서	-	지능형 안전 SW 프레임워크 요구사항 정의서	대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위해 개발된 솔루션을 가상 환경에서 시뮬레이션 분석을 도와주는 프레임워크 요구사항을 정의한 문서

	에너지 인프라 기반 SW 프레임워크 스켈레톤 (SW)	
	프로젝트	대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 개발

## 2. 안전SW 프레임워크 설계

### □ 프레임워크 아키텍처 구성도

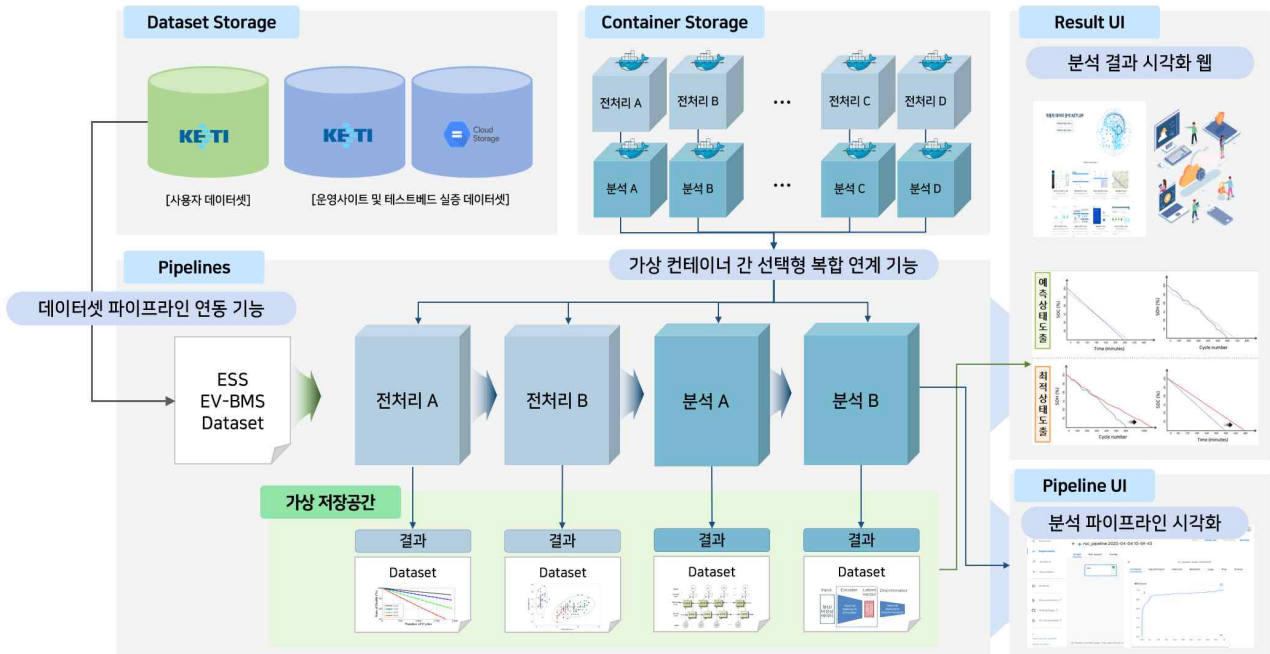


그림 3 지능형 안전 SW 프레임워크 아키텍처

### □ 프레임워크 아키텍처 구성 요소

#### o 데이터 저장소


- (기본 데이터 저장소) 운영사이트의 ESS 데이터와 테스트베드에서 실험을 통해 생성한 다양한 ESS 위험 상황의 데이터셋 제공
- (사용자 데이터 저장소) 사용자가 분석을 원하는 임의의 데이터를 이용하여 시뮬레이션을 진행할 수 있게 데이터를 업로드하는 공간입니다. 분석에 필요로 하는 필수 속성, 데이터 포맷, 파일명 등의 양식을 제공

#### o 컴포넌트 저장소

- 에너지 데이터 플랫폼에서 제공하는 다양한 전처리 엔진과 분석 엔진이 저장되어 있으며, 가상 컨테이너 이미지의 형태로 패키징된 독립적인 코드 조각으로 공개 SW의 도커 허브에 존재
- 가상화 컨테이너는 Json 포맷의 데이터셋을 읽고 반환할 수 있으며, 동일한 데이터 입출력 방법과 실행 방법을 컴포넌트 스펙 YAML 파일에 정의하여 'Kubeflow Pipeline' 을 통해 유연하게 실행가능

#### o Kubeflow

- 머신러닝 모델 학습부터 배포 단계까지 모든 작업에 필요한 도구와 환경을 쿠버네티스(Kubernetes) 위에서 kubeflow 컴포넌트를 제공

	에너지 인프라 기반 SW 프레임워크 스켈레톤 (SW)	
	프로젝트	대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 개발

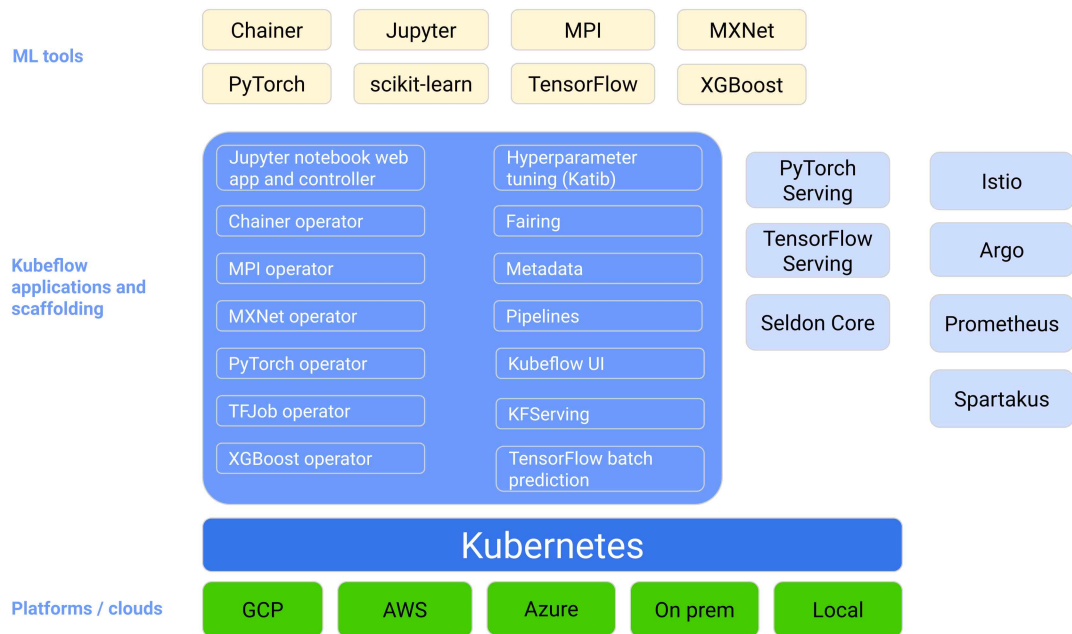


그림 4 Kubeflow 구조

### 3. Kubernetes 및 Kubeflow 구축

#### □ 개요

- 지능형 안전 SW 프레임워크를 설계하기 위해 Kubernetes와 Kubeflow 구축이 필요하다. 본 장에서는 두 가지를 구축하는 방법에 대해서 기술할 것이다.
- Kubernetes 클러스터를 구축하기 위해 GCP(Google Cloud Platform)의 3개의 인스턴스를 만들어 사용하였다.

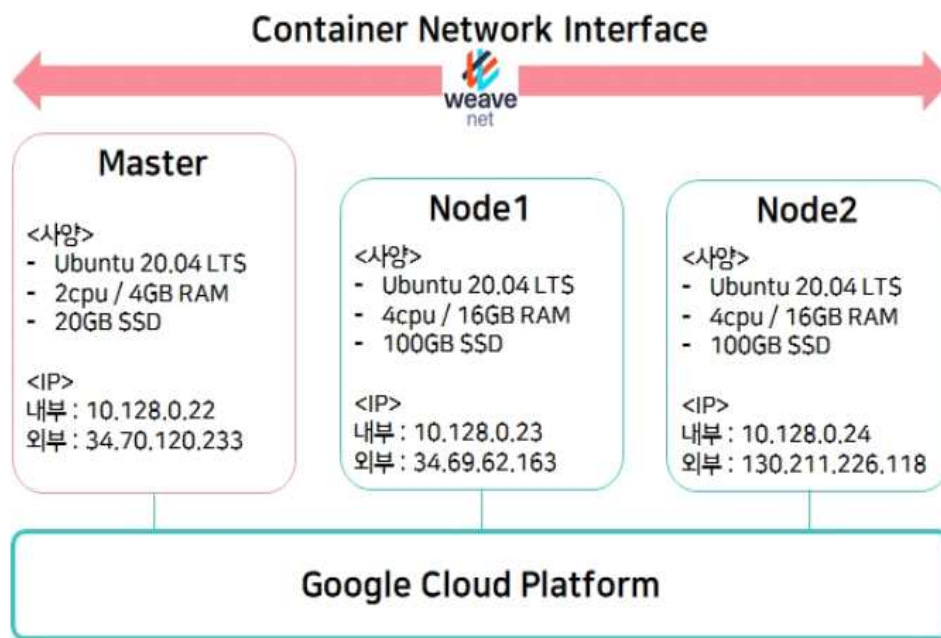


그림 5 Kubernetes 구성 환경

#### □ Kubernetes 구축

- 설치 이전에 master node와 worker node에 환경설정을 해줘야한다.

Command
<pre> swapoff -a &amp;&amp; sed -i '/swap/s/^\s*#/' /etc/fstab  cat &lt;&lt;EOF   sudo tee /etc/sysctl.d/k8s.conf net.bridge.bridge-nf-call-ip6tables = 1 net.bridge.bridge-nf-call-iptables = 1 EOF  sudo sysctl -system  systemctl stop firewalld  systemctl disable firewalld </pre>



## o Kudeadm, kubectl, kubelet 설치

### Command (master & worker node)

```
apt-get update

apt-get install -y apt-transport-https ca-certificates curl

curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg \
    https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] \
    https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

apt-get update

sudo apt-get install -qy kubelet=1.19.0-00 kubectl=1.19.0-00 kubeadm=1.19.0-00

apt-mark hold kubelet kubeadm kubectl

systemctl start kubelet

systemctl enable kubelet
```

## o Control-plane 구성(master node)

### Command (master & worker node)

```
#Master node에서만 진행
kubeadm init

cat > token.txt #kubeadm init으로 나온 결과를 복사해서 넣기


#여기서, kubectl get nodes 커맨드를 사용하면 실행이 안됨. 따라서, 아래 과정 진행
mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

#위 명령어를 실행시킨뒤 kubectl get nodes 커맨드를 실행시키면, 정상적으로 출력
#STATUS가 NotReady인데 이것은 Container Network Interface(CNI)때문임. 따라서, CNI를 설치해야 함

kubectl apply -f https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n' )
```

	에너지 인프라 기반 SW 프레임워크 스켈레톤 (SW)	
	프로젝트	대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 개발

#### o Worker node 구성

- 위 과정에서 master node 구성 시 생성된 token 값을 모든 Worker node에 붙여넣은 후 커맨드를 실행 후, node 확인 명령어를 입력하게 되면 아래 그림과 같은 결과를 확인할 수 있다.

```
root@master-k8s:/home/jwpark9010# kubectl get nodes
NAME          STATUS    ROLES          AGE    VERSION
master-k8s    Ready     control-plane, 19h    v1.23.1
node1-k8s     Ready     <none>         19h    v1.23.1
node2-k8s     Ready     <none>         19h    v1.23.1
root@master-k8s:/home/jwpark9010#
```

그림 6 설치가 완료된 kubernetes 화면

### □ Kubeflow 설치

#### o Dynamic volume provisioner 설치

- kubeflow를 쉽게 설치하기 위해서는 동적 볼륨 프로비저너(dynamic volume provisioner)가 필요
- kubeflow는 기본 스토리지 클래스를 사용하기 때문에, local-path 스토리지 클래스를 기본 클래스로 설정해야함

Command
kubectl apply -f <a href="https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml">https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml</a>
kubectl get storageclass #조회
kubectl patch storageclass local-path -p '{"metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
kubectl get sc #기본 클래스가 설정된 것을 확인

#### o Kubeflow 1.4v 다운로드 및 환경설정

- kubeflow 1.3v 이후로 kustomize를 이용하여 간단하게 설치할 수 있음

Command
git clone <a href="https://github.com/kubeflow/manifests.git">https://github.com/kubeflow/manifests.git</a>
git checkout v1.4-branch #1.4v 설치하기 위한 브랜치 변경

- kubeflow 설치전에 접속 계정 수정

Command
# 1. manifests/common/user-namespace/base/params.env 파일수정
# before
user=user@example.com

profile-name=kubeflow-user-example-com

# after

user=jwpark9010@gmail.com

profile-name=jinwon

### Command

#2. manifests/common/dex/base/config-map.yaml 파일수정

# before

staticPasswords:

- email: user@example.com

hash: \$2y\$12\$4K/VkmDdlq1Orb3xAt82zu8gk7Ad6ReFR4LCP9UeYE90NLiN9Df72

# https://github.com/dexidp/dex/pull/1601/commits

# FIXME: Use hashFromEnv instead

username: user

userID: "15841185641784"

# after

staticPasswords:

- email: user@example.com

hash: \$2y\$12\$4K/VkmDdlq1Orb3xAt82zu8gk7Ad6ReFR4LCP9UeYE90NLiN9Df72

# https://github.com/dexidp/dex/pull/1601/commits

# FIXME: Use hashFromEnv instead

username: user

userID: "15841185641784"

- email: jwpark9010@gmail.com

hash: \$2b\$10\$XloPMZeB4FBgLdVUApM3b.buKil58p6f4Eq9hPyvNHNwM.QY/K8Hy

username: jinwon

userID: jinwon

o kustomize 다운로드

- kubeflow 1.4v 설치하기 위해서는 kustomize가 필요
- kustomize 파일은 이전에 설치된 manifests에 다운

### Command

wget https://github.com/kubernetes-sigs/kustomize/releases/download/v3.2.0/kustomize\_3.2.0\_linux\_amd64

mv kustomize\_3.2.0\_linux\_amd64 kustomize

o Kustomize 이용하여 kubeflow 설치

### Command

# manifests 폴더로 이동한뒤 아래의 명령어를 실행


while ! kustomize build example | kubectl apply -f -; do echo "Retrying to apply resources"; sleep 10; done

o 서비스 활용을 위한 포트포워딩 작업

- 아래 명령어를 사용하여 나오는 PORT들에 대해서 포트포워딩 작업을 해줘야함

### Command

kubectl get svc -n istio-system istio-ingressgateway

	에너지 인프라 기반 SW 프레임워크 스켈레톤 (SW)	
	프로젝트	대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 개발

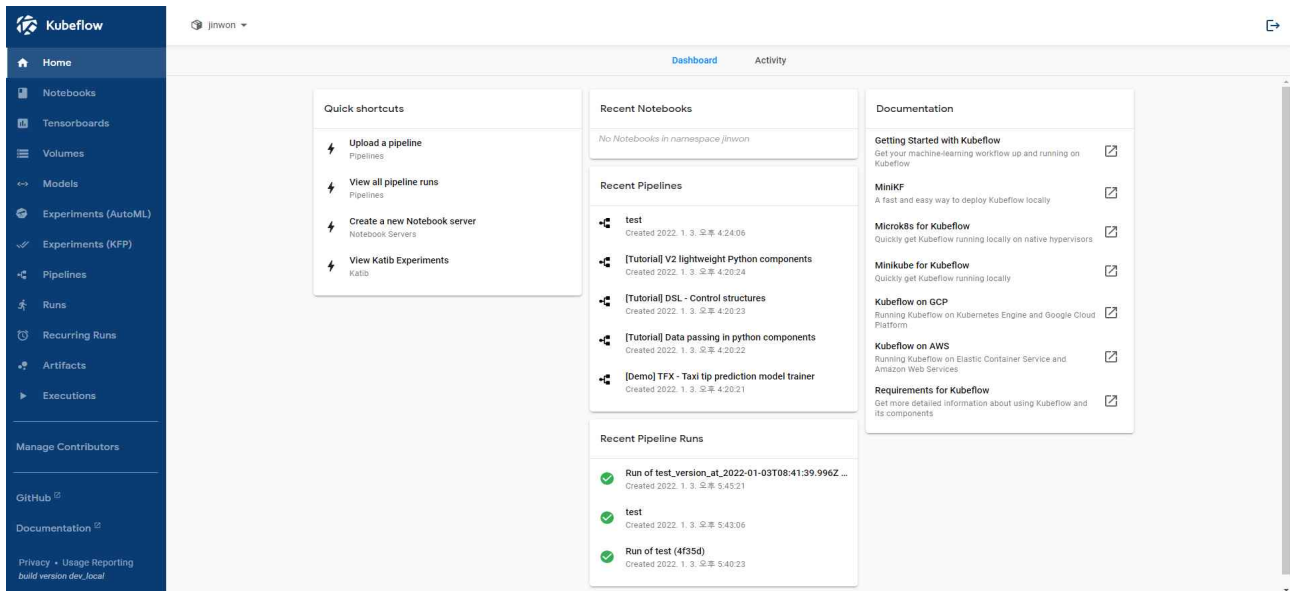


그림 7 구축된 Kubeflow 메인화면

## 4. Kubeflow를 활용한 Pipeline 설계

### □ 개요

- 위 장에서 구축된 MLops 프레임워크인 Kubeflow 기반으로 Pipeline 설계방법을 소개한다.

### □ Kubeflow Pipeline

- kubeflow pipeline(KFP)은 python으로 작성할 수 있게 SDK를 지원한다. 아래 코드는 KFP SDK를 사용하여, 데이터 읽어오기, 데이터 전처리, 데이터 변환, 모델 학습 등의 일련의 과정을 수행한다.

#### keti\_mlops\_tutorial.ipynb

```
import os
import time
import requests
from random import randrange
from datetime import datetime
import yaml
import pprint
import logging

try:
    import kfp
    import kfp.components as comp
    from kfp import dsl
    from kfp import onprem
    from kubernetes import client as k8s_client
except ImportError:
    print("Trying to install required module : kfp \n")
    os.system('python -m pip install kfp')
    import kfp
    import kfp.components as comp
    from kfp import dsl
    from kfp import onprem
    from kubernetes import client as k8s_client

logger = logging.getLogger()

def create_persistent_volume_func(pvol_name="keti-shared-volume"):
    vop = kfp.dsl.VolumeOp(
        name = pvol_name,
        resource_name = pvol_name,
        #volume_name = pvol_name,
```

```

        size = '10Gi',
        #modes = kfp.dsl.VOLUME_MODE_RWM,
        #generate_unique_name = False
    ).set_display_name("[0] Import Creating persistent volume")
    return vop

def data_selection_func(pvc_args, load_data_args):#prev_cont, load_data_args):
    data_selection_cont = dsl.ContainerOp(
        name="data_selection",
        image="kjoohyu/data_selection:0.11",#"ketidp/ess:data_selection_v0.1",
        arguments=[
            '--selected_data', load_data_args["selected_data"],
            '--type', load_data_args["ess_type"],
            '--start_date',load_data_args["start_date"] ,
            '--end_date', load_data_args["end_date"] ,
            '--Bank', load_data_args["Bank"],
            '--Rack', load_data_args["Rack"],
            '--Bank_num', load_data_args["Bank_num"],
            '--Rack_num', load_data_args["Rack_num"],
            '--Bank_columns', load_data_args["Bank_columns"],
            '--Rack_columns', load_data_args["Rack_columns"],
        ],
        command=['python', 'load_data.py'],
        #pvolumes={"/data": prev_cont.volume} # volume container
    ).set_display_name("[1] Load ESS battery data") \
    .apply(onprem.mount_pvc(pvc_args.get("pvc_name"),
                            volume_name=pvc_args.get("volume_name"),
                            volume_mount_path=pvc_args.get("volume_mount_path")))

    return data_selection_cont

def split_train_test_func(prev_cont, pvc_args, split_train_test_args, label_column):
    split_train_test_cont = dsl.ContainerOp( # train, test 데이터 분리
        name="split train test data",
        image="kjoohyu/split_train_test:0.12",#"ketidp/ess:split_train_test_v0.1",
        arguments=[
            ' - - l o a d _ d a t a _ p a t h ' ,
            pvc_args.get("volume_mount_path"),#dsl.InputArgumentPath(data_selection_cont.outputs['data']),
            '--save_data_path', pvc_args.get("volume_mount_path"),
            '--split_method', split_train_test_args,
            '--label_column', label_column
        ],
        command=['python', 'split_data.py'],
        #pvolumes={"/data": prev_cont.pvolume}

```

```

        ).set_display_name("[2] Split raw data to train them").after(prev_cont) \
        .apply(onprem.mount_pvc(pvc_args.get("pvc_name"),
                                volume_name=pvc_args.get("volume_name"),
                                volume_mount_path=pvc_args.get("volume_mount_path")))

    return split_train_test_cont

class PreProcessing:
    def anomaly_func(prev_cont, pvc_args, anomaly_detection_args):
        anomaly_cont = dsl.ContainerOp(
            name="preprocessing-anomaly",

image="kjoohyu/preprocessing_anomaly:0.15",#ketidp/ess:preprocessing_anomaly_v0.1",
            arguments=[
                ' - - s p l i t _ X _ t r a i n ',
pvc_args.get("volume_mount_path")+'/X_train.csv',
                ' - - s p l i t _ Y _ t r a i n ',
pvc_args.get("volume_mount_path")+'/Y_train.csv',
                ' - - s p l i t _ X _ t e s t ',
pvc_args.get("volume_mount_path")+'/X_test.csv',
                ' - - s p l i t _ Y _ t e s t ',
pvc_args.get("volume_mount_path")+'/Y_test.csv',
                '--anomaly_args', anomaly_detection_args,
                '--save_data_path', '/data'
            ],
            command=['python', 'preprocessing_anomaly_detection.py'],
            #pvolumes={"data": prev_cont.pvolume}
        ).set_display_name("[3-1] Preprocessing : Anomaly
detection").after(prev_cont) \
        .apply(onprem.mount_pvc(pvc_args.get("pvc_name"),
                                volume_name=pvc_args.get("volume_name"),

volume_mount_path=pvc_args.get("volume_mount_path")))
    return anomaly_cont

    def scaler_func(prev_cont, pvc_args, scaler_method, scaler_args):
        scaler_cont = dsl.ContainerOp(
            name="preprocessing-scaler",

image="kjoohyu/preprocessing_scaler:0.12",#ketidp/ess:preprocessing_scaler_v0.1",
            arguments=[
                ' - - s p l i t _ X _ t r a i n ',
pvc_args.get("volume_mount_path")+'/X_train.csv',
                ' - - s p l i t _ X _ t e s t ',

```

```

pvc_args.get("volume_mount_path")+'/X_test.csv',
        '--prep_method', scaler_method,
        '--prep_args', scaler_args,
        '--save_data_path', '/data'
    ],
    command=['python', 'preprocessing_scaler.py'],
    #pvolumes={"/data": prev_cont.pvolume}
).set_display_name("[3-2] Preprocessing : Scale Up &
Down").after(prev_cont) \
        .apply(onprem.mount_pvc(pvc_args.get("pvc_name"),
                                volume_name=pvc_args.get("volume_name"),

volume_mount_path=pvc_args.get("volume_mount_path")))
    return scaler_cont

class MachineLearning:
    def regression_func(prev_cont, pvc_args, reg_method, reg_args):
        regression_cont = dsl.ContainerOp(
            name="ml_model_regresstion",

image="kjoohyu/ml_regresstion:0.15",#ketidp/ess:ml_regresstion_v0.1",
        arguments=[
            '--X_train', pvc_args.get("volume_mount_path")+'/X_train.csv',
            '--Y_train', pvc_args.get("volume_mount_path")+'/Y_train.csv',
            '--X_test', pvc_args.get("volume_mount_path")+'/X_test.csv',
            '--Y_test', pvc_args.get("volume_mount_path")+'/Y_test.csv',
            '--regression_method', reg_method,
            '--regression_args',reg_args,
            '--loss_function', 'mse',
            '--save_data_path',
pvc_args.get("volume_mount_path")+'/result.csv'
        ],
        command=['python', 'ml_regresstion.py'],
        #pvolumes={"/data": prev_cont.pvolume}
).set_display_name("[4] MachineLearning : Regression
method").after(prev_cont) \
        .apply(onprem.mount_pvc(pvc_args.get("pvc_name"),
                                volume_name=pvc_args.get("volume_name"),

volume_mount_path=pvc_args.get("volume_mount_path")))
    #.set_gpu_limit(1)
    #ml_regression_cont.add_node_selector_constraint('nvidia')
    return regression_cont

```



```
def classification_func(prev_cont, pvc_args, cls_method, cls_args):
    classification_cont = dsl.ContainerOp(
        name="ml_model_classification",
        image="kjoohyu/ml_classification:0.1", #ketidp/ess:ml_classification_v0.1",
        arguments=[
            '--X_train', pvc_args.get("volume_mount_path")+'/X_train.csv',
            '--Y_train', pvc_args.get("volume_mount_path")+'/Y_train.csv',
            '--X_test', pvc_args.get("volume_mount_path")+'/X_test.csv',
            '--Y_test', pvc_args.get("volume_mount_path")+'/Y_test.csv',
            '--classification_method', cls_method,
            '--classification_args', cls_args,
            ' - - s a v e _ d a t a _ p a t h ' ,
            pvc_args.get("volume_mount_path")+'/result.csv'
        ],
        command=['python', 'ml_classification.py'],
        #pvolumes={"data": prev_cont.pvolume}
        ).set_display_name("[4] MachineLearning : Classification
method").after(prev_cont) \
        .apply(onprem.mount_pvc(pvc_args.get("pvc_name"),
                                volume_name=pvc_args.get("volume_name"),
                                volume_mount_path=pvc_args.get("volume_mount_path")))
        return classification_cont

##### SAMPLE PIPELINES
#####
@kfp.dsl.pipeline(
    name = 'Sample Pipeline 1',
    description = 'sample pipeline case 1'
)
# Sample pipeline case #1
def sample_pipeline_1():
    target_pl = kf_params_dict.get('target_pl')
    pvc_args = target_pl.get("pvc_args")
    #vop = create_persistent_volume_func(pvol_name=target_pl.get('persist_volume'))
    data_selection_cont = data_selection_func(pvc_args=pvc_args,
                                              load_data_args=target_pl.get('load_data_args'))
    split_train_test_cont = split_train_test_func(prev_cont=data_selection_cont,
                                                    pvc_args=pvc_args,
```

```

split_train_test_args=target_pl.get('split_train_test_args'),
                                label_column=target_pl.get('label_column'))
    anomaly_cont = PreProcessing.anomaly_func(prev_cont=split_train_test_cont,
                                              pvc_args=pvc_args,

anomaly_detection_args=target_pl.get('anomaly_detection_args'))
    scaler_cont = PreProcessing.scaler_func(prev_cont=anomaly_cont,
                                           pvc_args=pvc_args,
                                           scaler_method=target_pl.get('scaler_method'),

scaler_args=kf_params_dict.get(target_pl.get('scaler_args'))
    regression_cont = MachineLearning.regression_func(prev_cont=scaler_cont,
                                                       pvc_args=pvc_args,
                                                       reg_method=target_pl.get('reg_method'),

reg_args=kf_params_dict.get(target_pl.get('reg_args'))
    # No caching parts
    data_selection_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
    split_train_test_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
    anomaly_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
    scaler_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
    regression_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'

@kfp.dsl.pipeline(
    name = 'Sample Pipeline 2',
    description = 'sample pipeline case 2'
)
# Sample pipeline case #2
def sample_pipeline_20:
    target_pl = kf_params_dict.get('target_pl')
    pvc_args = target_pl.get("pvc_args")
    #vop = create_persistent_volume_func(pvol_name=target_pl.get('persist_volume'))
    data_selection_cont = data_selection_func(pvc_args=pvc_args,
                                              load_data_args=target_pl.get('load_data_args'))
    split_train_test_cont = split_train_test_func(prev_cont=data_selection_cont,
                                                  pvc_args=pvc_args,

split_train_test_args=target_pl.get('split_train_test_args'),
                                label_column=target_pl.get('label_column'))
    anomaly_cont = PreProcessing.anomaly_func(prev_cont=split_train_test_cont,
                                              pvc_args=pvc_args,

```

```

anomaly_detection_args=target_pl.get('anomaly_detection_args'))
    scaler_cont = PreProcessing.scaler_func(prev_cont=anomaly_cont,
                                           pvc_args=pvc_args,
                                           scaler_method=target_pl.get('scaler_method'),

scaler_args=kf_params_dict.get(target_pl.get('scaler_args'))
    classification_cont = MachineLearning.classification_func(prev_cont=scaler_cont,
                                                             pvc_args=pvc_args,

cls_method=target_pl.get('cls_method'),

cls_args=kf_params_dict.get(target_pl.get('cls_args'))

# No caching parts
data_selection_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
split_train_test_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
anomaly_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
scaler_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'
classification_cont.execution_options.caching_strategy.max_cache_staleness = 'P0D'

if __name__ == "__main__":

    # Load My KubeFlow Client info using colab env
    USERNAME = ""
    PASSWORD = ""
    NAMESPACE = ""
    HOST = ""
    #USERNAME = os.getenv("USERNAME")
    #PASSWORD = os.getenv("PASSWORD")
    #NAMESPACE = os.getenv("NAMESPACE")
    #HOST = os.getenv("HOST") # istio-ingressgateway's Node Port IP:PORT

    session = requests.Session()
    response = session.get(HOST)
    headers = {
        "Content-Type": "application/x-www-form-urlencoded",
    }
    user_data = {"login": USERNAME, "password": PASSWORD}
    session.post(response.url, headers=headers, data=user_data)
    session_cookie = session.cookies.get_dict()["authservice_session"]

```

```
# Connect My KubeFlow Client
client = kfp.Client(host=f"{HOST}/pipeline",
                    namespace=f"{NAMESPACE}",
                    cookies=f"authservice_session={session_cookie}",
)

# User's KF pipeline file written by YAML
#from google.colab import drive
#drive.mount('/content/gdrive')
#!more /content/gdrive/MyDrive/Colab_Notebooks/kf_params_v1.yaml
#kf_params_file_path = "/content/gdrive/MyDrive/Colab_Notebooks/kf_params_v1.yaml"
!curl -L -O https://github.com/keti-dp/OpenESS/raw/main/MLOps/kf_params_v1.yaml
kf_params_file_path = "/content/kf_params_v1.yaml" # To use the uploaded file
try:
    with open(kf_params_file_path) as f:
        kf_params_dict = yaml.load(f, Loader=yaml.FullLoader)
        logger.info("Load KubeFlow Pipeline values from {}", kf_params_file_path.split("/")[-1])
        # pprint.pprint(kf_params_v1)
except FileNotFoundError as e:
    logger.error("Failed to load KubeFlow Pipeline values from {}",
kf_params_file_path.split("/")[-1])
    raise e

# (example) pipeline selection
# load_data + split_train + anomaly_detection + reg(xgboost) + result
pipeline_func_name = kf_params_dict.get('user_pipeline_func')
pipeline_func = eval(pipeline_func_name)
logger.info("User pipeline fuction name : ", pipeline_func_name)

now = datetime.now()
dt_string = now.strftime("%d/%m/%Y_%H:%M:%S")
print("[KF_pipeline_log] Start time : ", dt_string)

run_name = pipeline_func.__name__ + '-run-' + dt_string

# Submit pipeline directly from pipeline function
#arguments = {}
run_result = client.create_run_from_pipeline_func(pipeline_func,
                                                  run_name=run_name,
                                                  namespace=NAMESPACE,
                                                  arguments={})
```