머신러닝 파이프라인

쿠베플로우 파이프라인 Part 1

송호연

목차

쿠베플로우 파이프라인 Part 1

학습목표

쿠베플로우 파이프라인 Part 1

01. 쿠베플로우 파이프라인에 대해 이해한다.

쿠베플로우 파이프라인이 왜 중요한 지, 어떤 기능을 갖고 있는지 이해한다.

02. 쿠베플로우 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 쿠베플로우 파이프라인의 기본 사용법에 대해 공부한다.

쿠베플로우 파이프라인 개요

01

쿠베플로우 파이프라인 개요

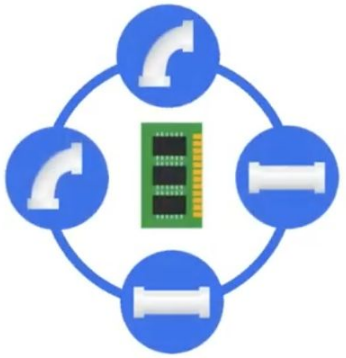
쿠베플로우 파이프라인이 필요한 이유

*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

쿠베플로우 파이프라인이 필요한 이유

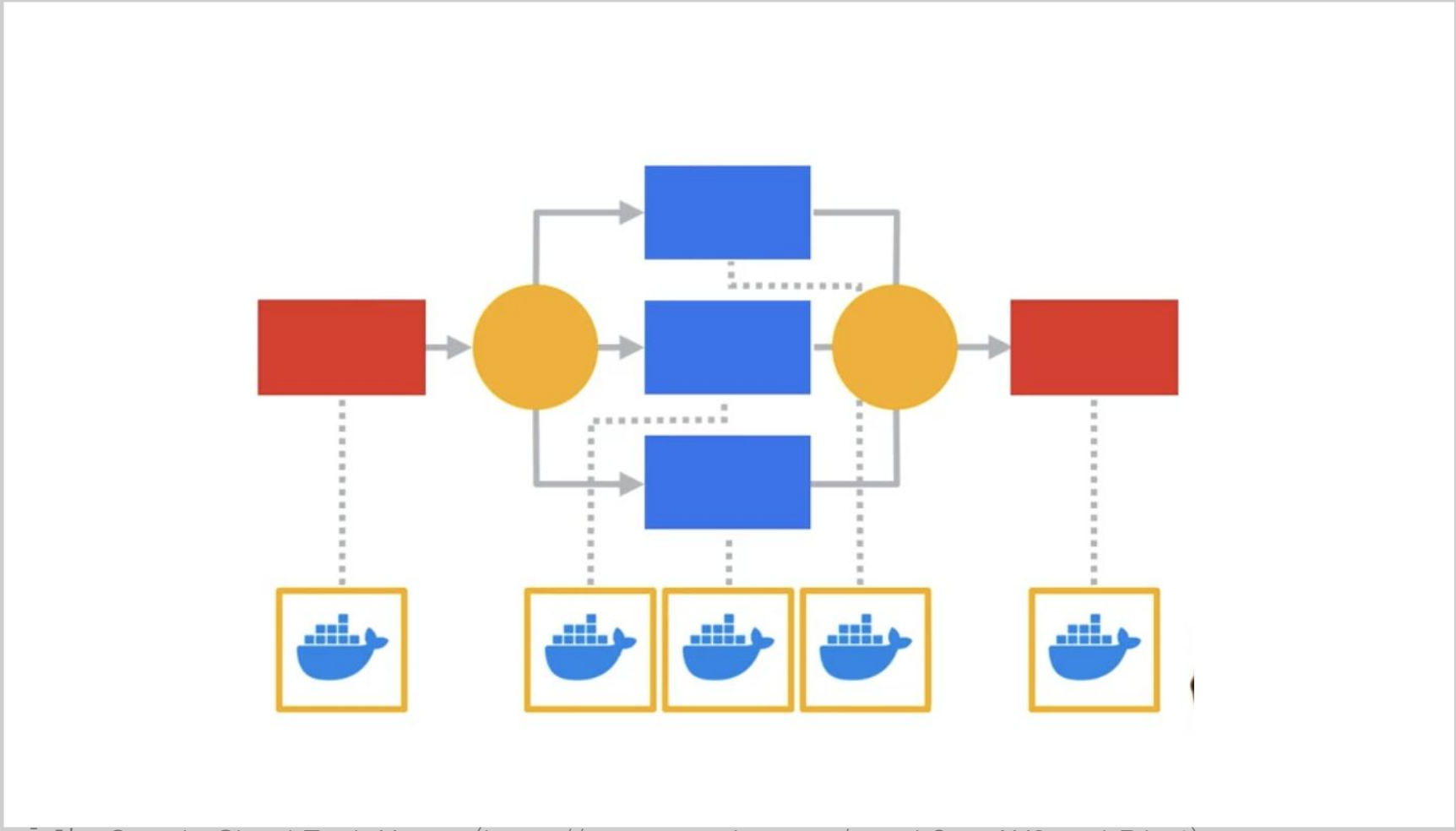
- 머신러닝 파이프라인 오케스트레이션 단순화
- 실험, 재현, 공유
- 컴포넌트를 빠르게 재사용하고 연결



*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

쿠베플로우 파이프라인이 필요한 이유



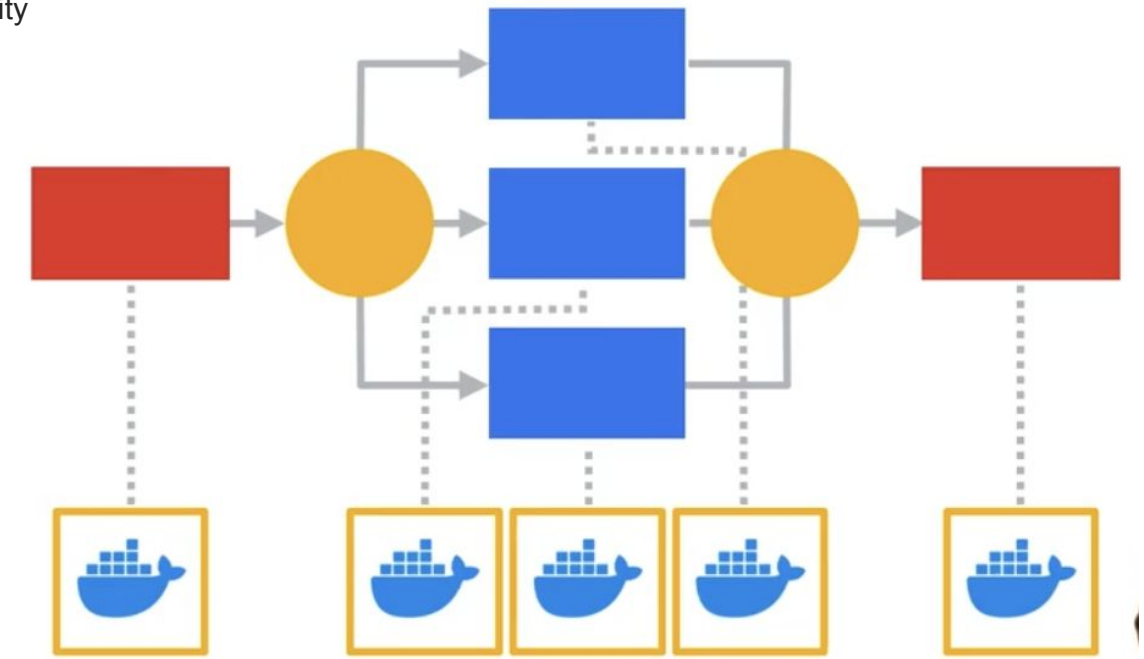
*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

쿠베플로우 파이프라인이 필요한 이유

실행 환경과 코드 런타임을 분리

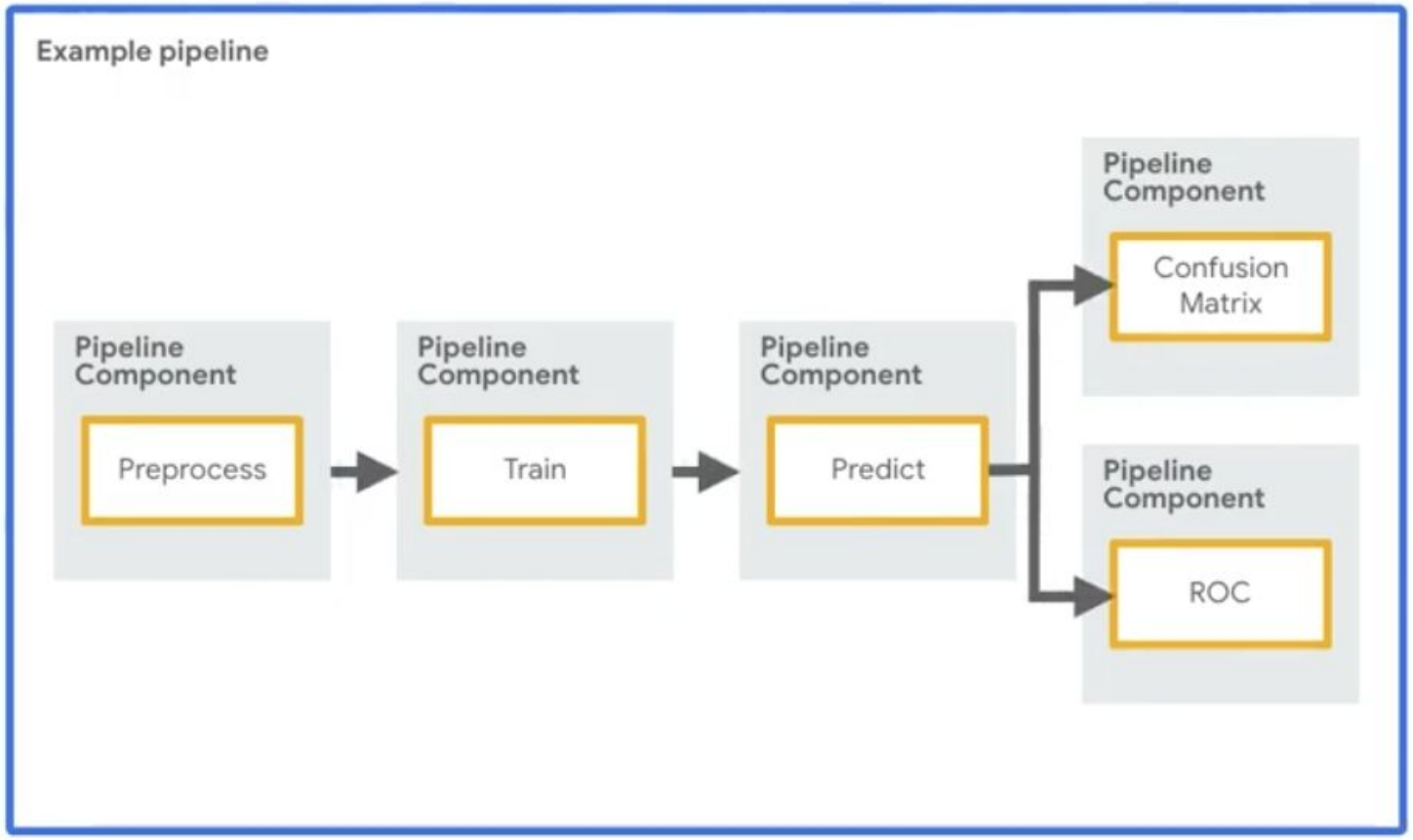
- 이식성 Portability
- 반복가능성 Repeatability
- 캡슐화 Encapsulation



*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

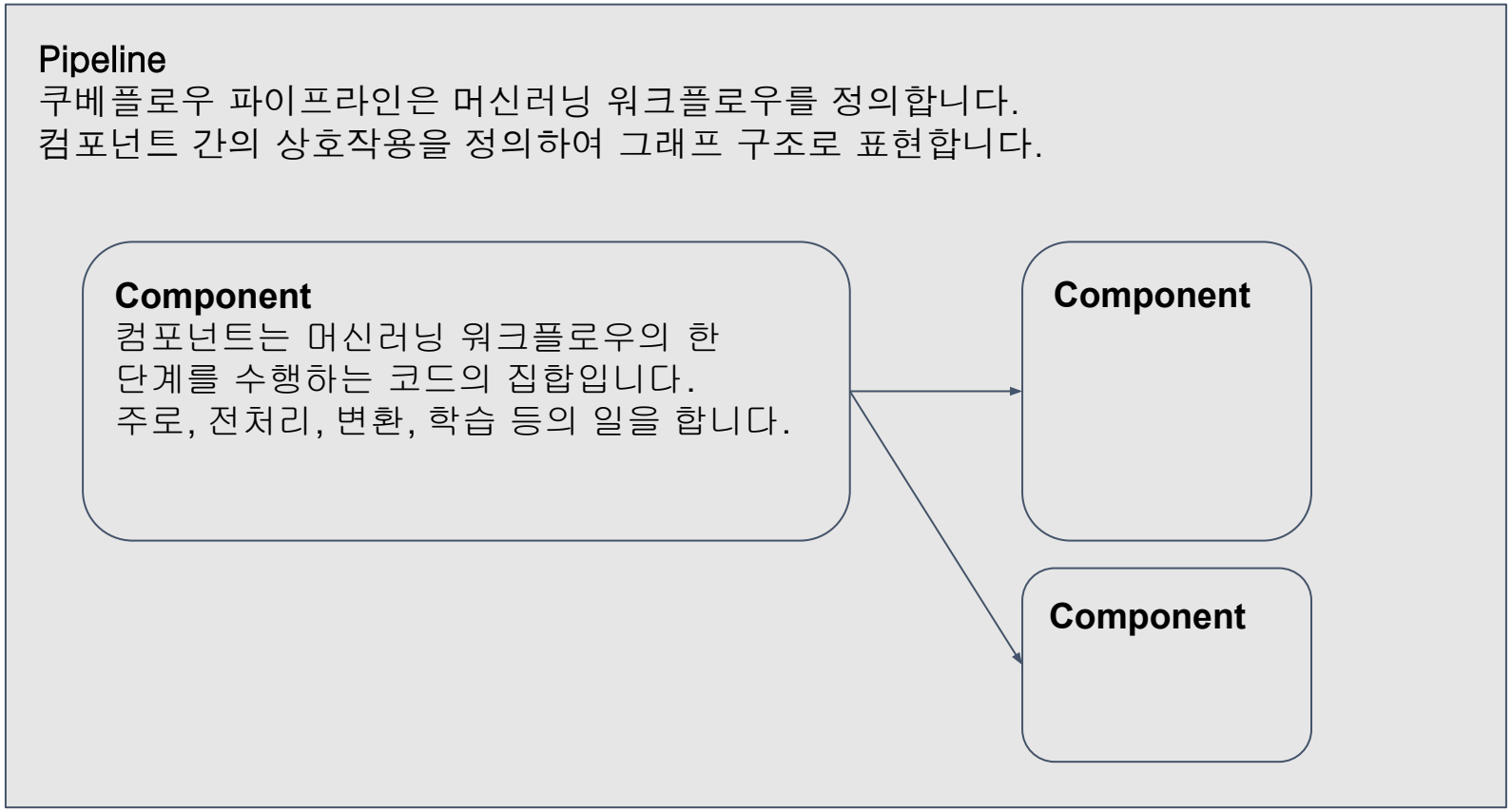
파이프라인 & 컴포넌트



*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

파이프라인 & 컴포넌트

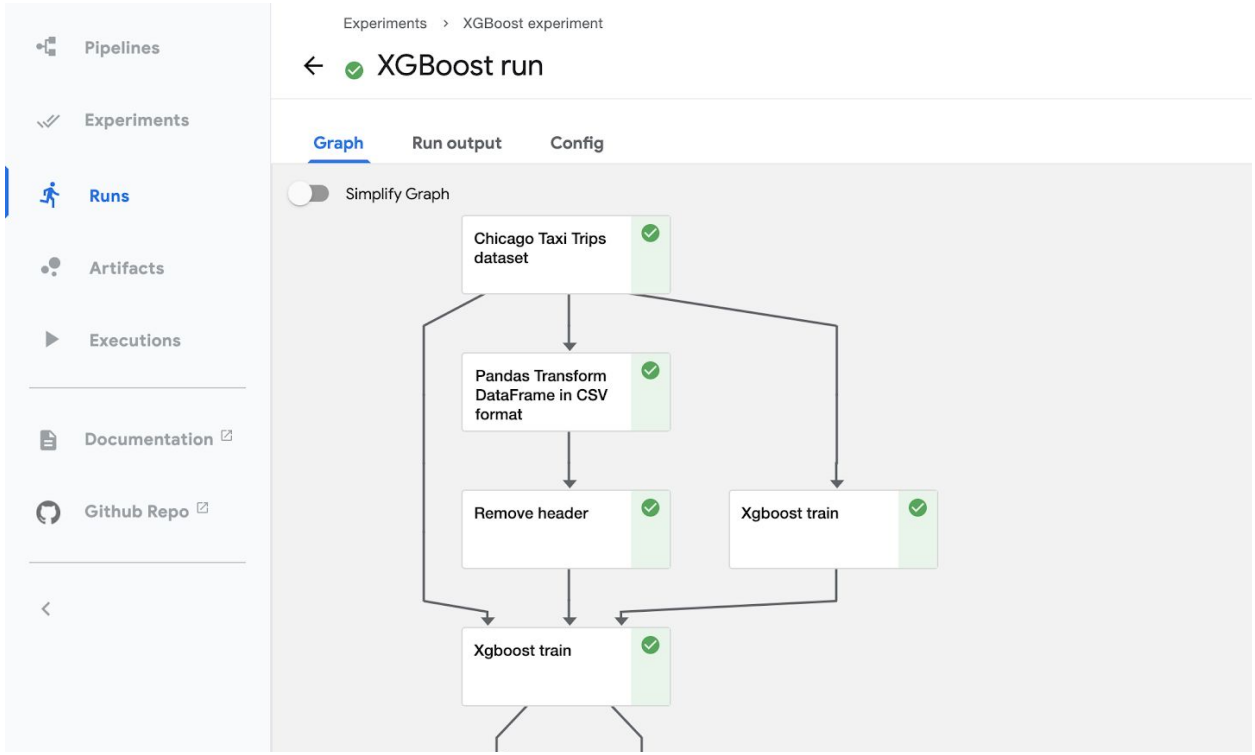


*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

그래프

그래프는 쿠베플로우 파이프라인의 시각적 표현입니다.
그래프는 각 단계를 출력하며, 컴포넌트간 의존 관계를 표현합니다.

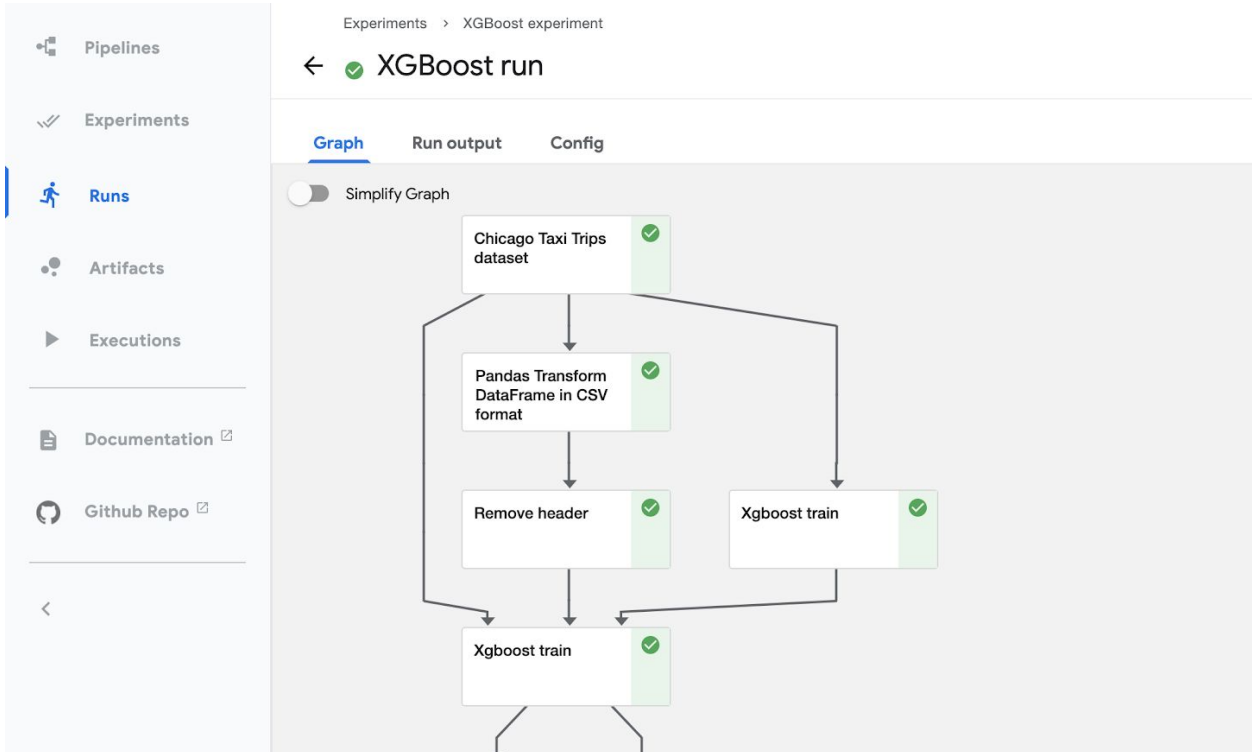


*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

실험 Experiment

실험은 파이프라인을 다양한 설정으로 돌려볼 수 있습니다.



*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

쿠베플로우 파이프라인 코드 예시

```
In [14]: import kfp.dsl as dsl

         def my_pipeline_step(step_name, param1, param2, ...):
             return dsl.ContainerOp(
                 name = step_name,
                 image = '<path to my container image>',
                 arguments = [
                     '--param1', param1,
                     '--param2', param2,
                     ...
                 ],
                 file_outputs = {
                     'output1': '/output1.txt',
                     'output2': '/output2.json',
                     ...
                 }
             )
```

*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

쿠베플로우 파이프라인 코드 예시

```
In [14]: import kfp.dsl as dsl

def my_pipeline_step(step_name, param1, param2, ...):
    return dsl.ContainerOp(
        name = step_name,
        image = '<path to my container image>',
        arguments = [
            '--param1', param1,
            '--param2', param2,
            ...
        ],
        file_outputs = {
            'output1': '/output1.txt',
            'output2': '/output2.json',
            ...
        }
    )
```

input

*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

쿠베플로우 파이프라인 코드 예시

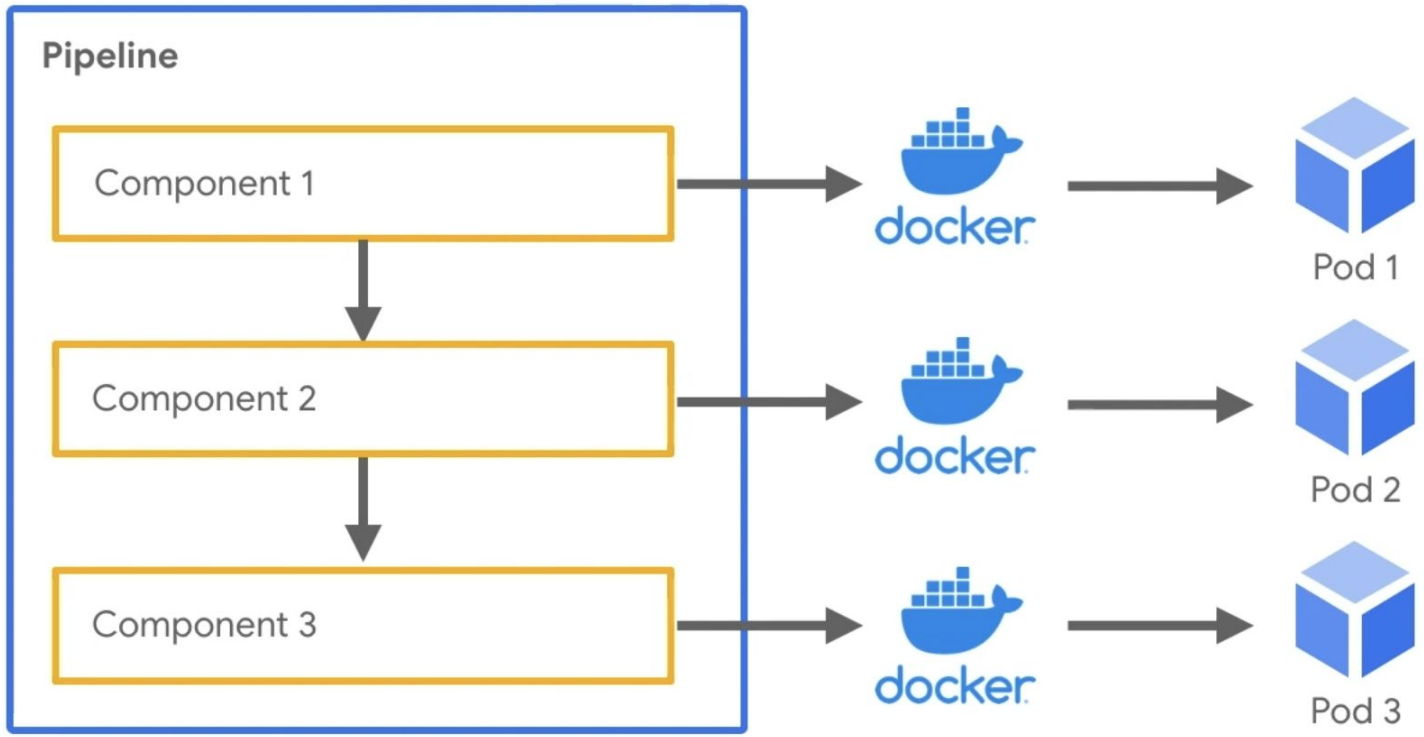
```
In [14]: import kfp.dsl as dsl

def my_pipeline_step(step_name, param1, param2, ...):
    return dsl.ContainerOp(
        name = step_name,
        image = '<path to my container image>',
        arguments = [
            '--param1', param1,
            '--param2', param2,
            ...
        ],
        file_outputs = {
            'output1': '/output1.txt',
            'output2': '/output2.json',
            ...
        }
    )
```

output

*출처 : Google Cloud Tech Youtue(https://www.youtube.com/watch?v=_AY8mmbR1o4)

쿠베플로우 파이프라인 개요

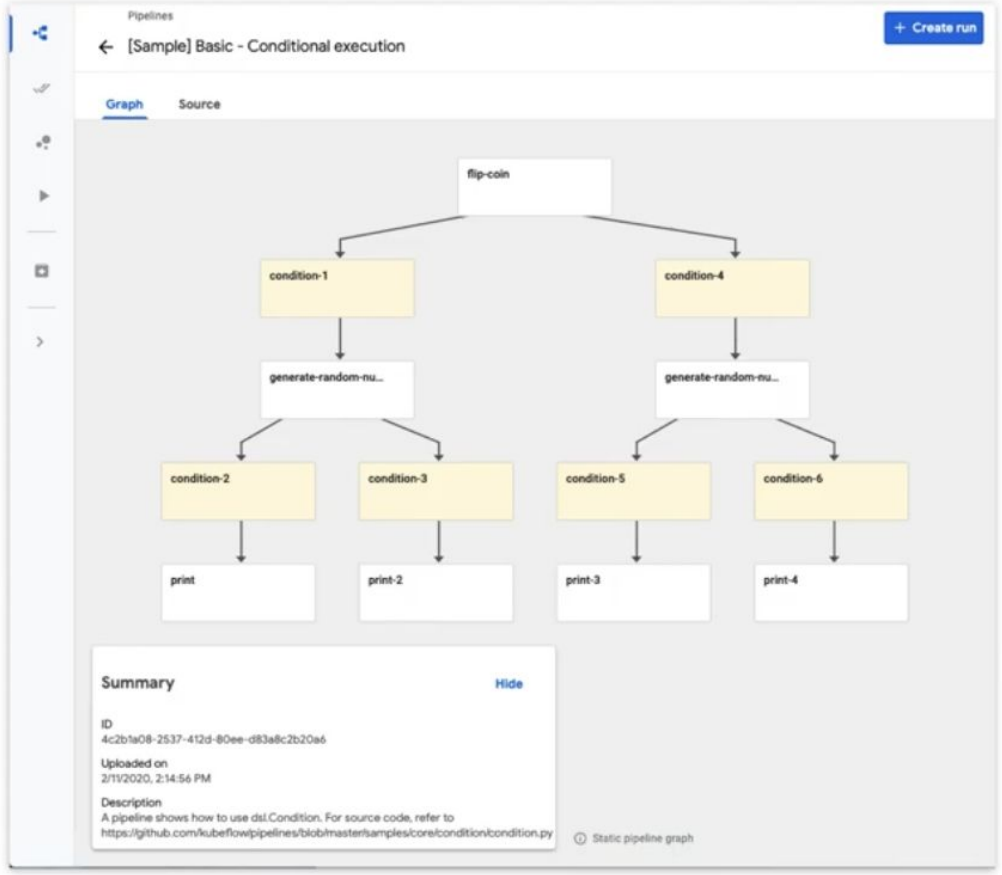
쿠베플로우 파이프라인이 필요한 이유



*출처 : 출처 작성

쿠베플로우 파이프라인 개요

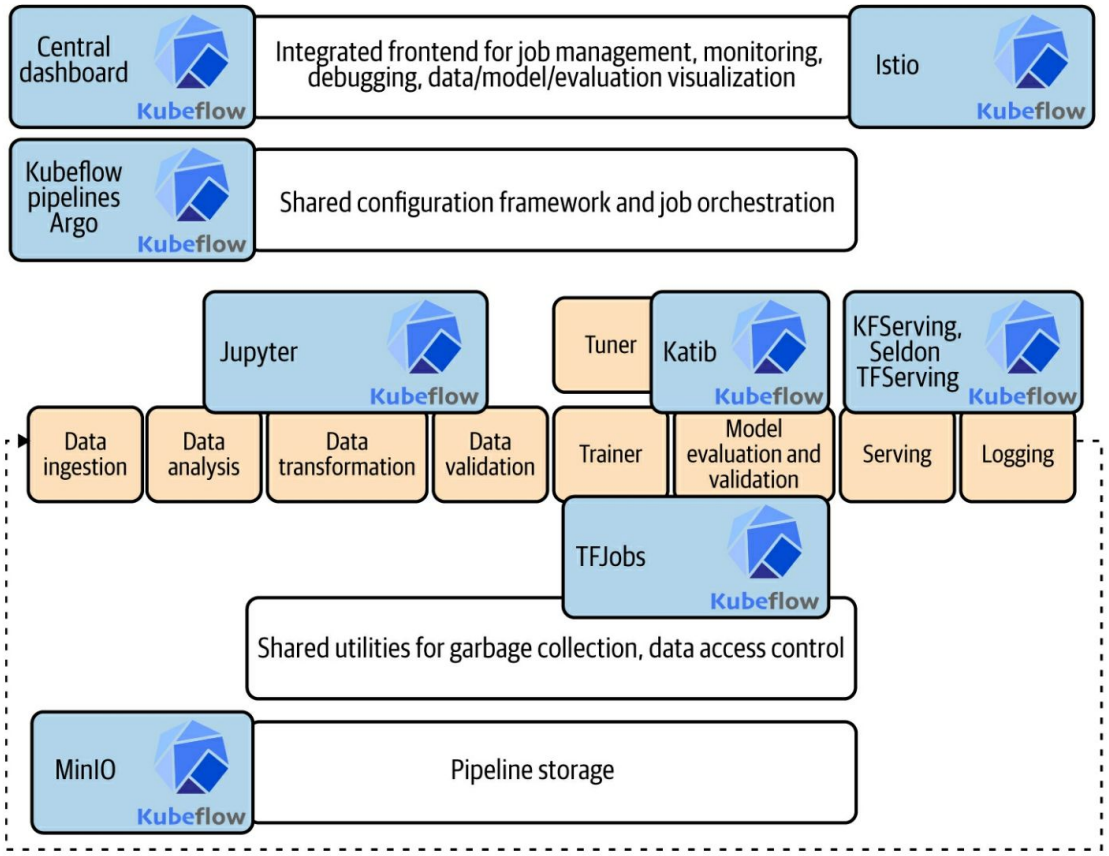
쿠베플로우 파이프라인이 필요한 이유



*출처 : 출처 작성

쿠베플로우 파이프라인 개요

쿠베플로우 구성요소



*출처 : 출처 작성

실습 1

Hello World

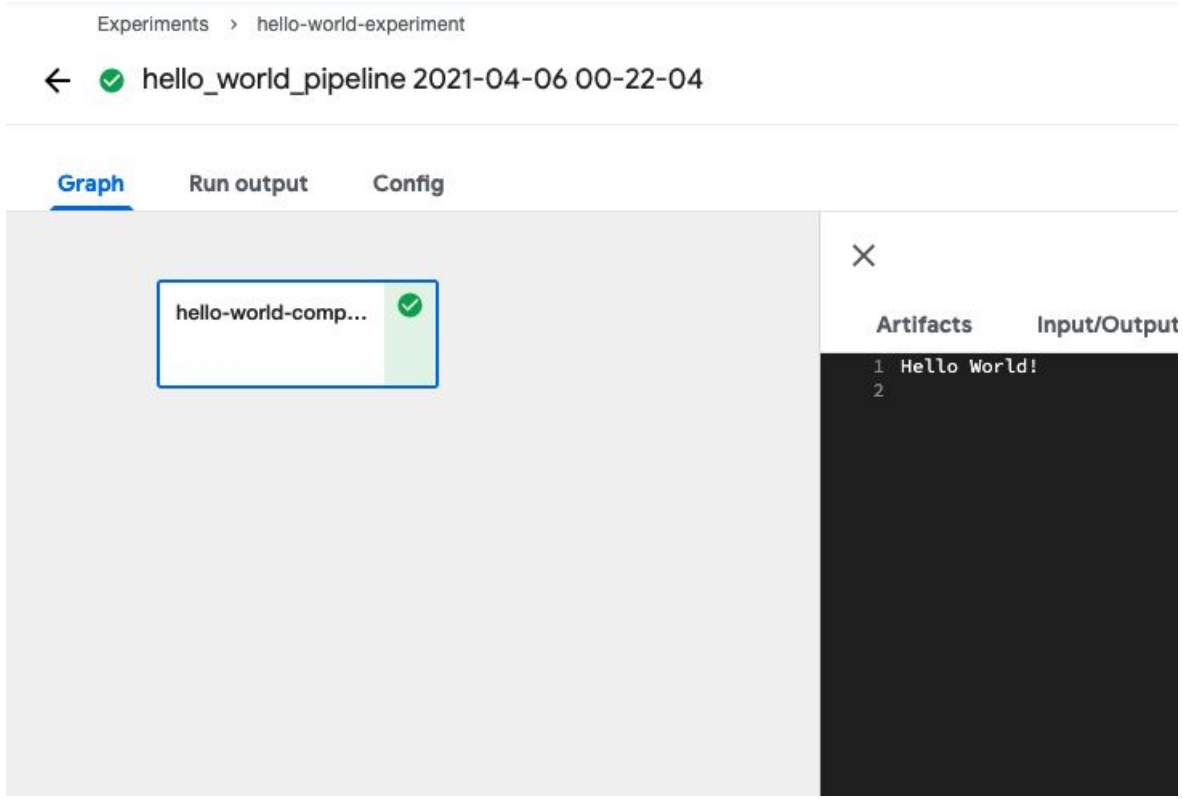
Add

Parallel

실습 1

Hello World

https://github.com/chris-chris/kubeflow-tutorial/tree/master/lesson2_hello_world



실습 1

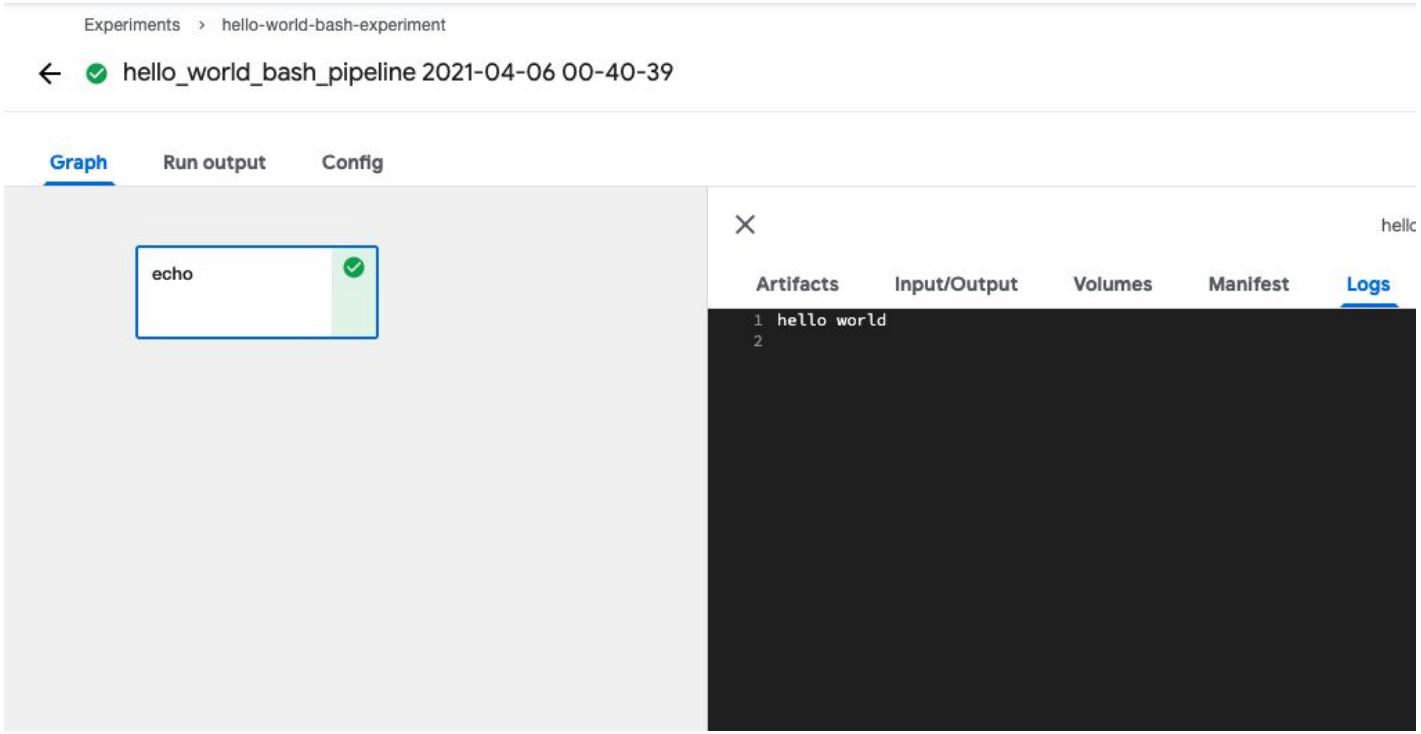
Hello World

```
import kfp

KUBEFLOW_HOST = "http://127.0.0.1:8080/pipeline"

def hello_world_component():
    ret = "Hello World!"
    print(ret)
    return ret

@kfp.dsl.pipeline(name="hello_pipeline", description="Hello World Pipeline!")
def hello_world_pipeline():
    hello_world_op = kfp.components.func_to_container_op(hello_world_component)
    _ = hello_world_op()

if __name__ == "__main__":
    kfp.compiler.Compiler().compile(hello_world_pipeline, 'hello-world-pipeline.zip')
    kfp.Client(host=KUBEFLOW_HOST).create_run_from_pipeline_func(
        hello_world_pipeline,
        arguments={},
        experiment_name="hello-world-experiment")
```

실습 1

Hello World

https://github.com/chris-chris/kubeflow-tutorial/tree/master/lesson2_hello_world



실습 1

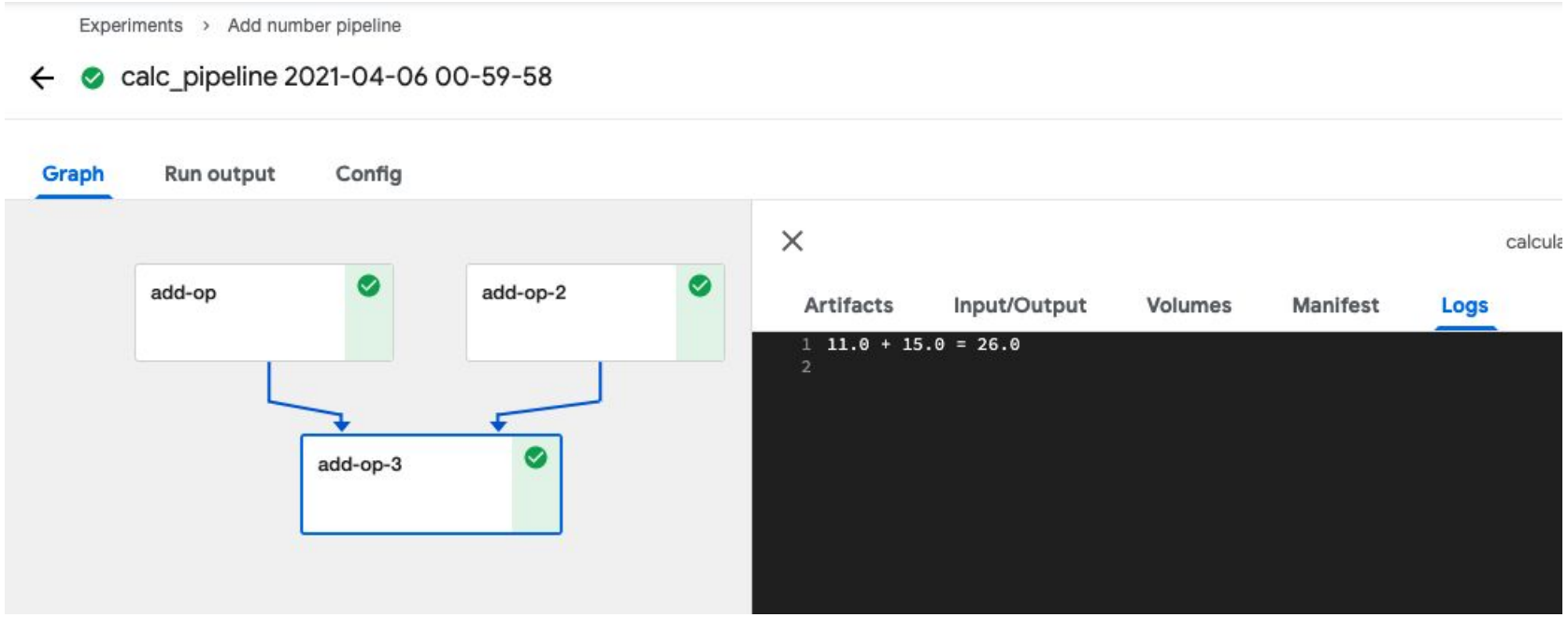
Hello World

```
import kfp
from kfp import dsl

BASE_IMAGE = "library/bash:4.4.23"
KUBEFLOW_HOST = "http://127.0.0.1:8080/pipeline"

def echo_op():
    return dsl.ContainerOp(
        name="echo",
        image=BASE_IMAGE,
        command=["sh", "-c"],
        arguments=['echo "hello world"'],
    )

@dsl.pipeline(name="hello_world_bash_pipeline", description="A hello world pipeline.")
def hello_world_bash_pipeline():
    echo_task = echo_op()

if __name__ == "__main__":
    kfp.compiler.Compiler().compile(hello_world_bash_pipeline, __file__ + ".zip")
    kfp.Client(host=KUBEFLOW_HOST).create_run_from_pipeline_func(
        hello_world_bash_pipeline,
        arguments={},
        experiment_name="hello-world-bash-experiment",
    )
```

실습 1

Add

https://github.com/chris-chris/kubeflow-tutorial/tree/master/lesson3_add



실습 1

Add

```
import kfp
from kfp import components
from kfp import dsl


EXPERIMENT_NAME = 'Add number pipeline'        # Name of the experiment in the UI
BASE_IMAGE = "python:3.7"
KUBEFLOW_HOST = "http://127.0.0.1:8080/pipeline"


@dsl.python_component(
    name='add_op',
    description='adds two numbers',
    base_image=BASE_IMAGE  # you can define the base image here, or when you build in the next step.
)
def add(a: float, b: float) -> float:
    '''Calculates sum of two arguments'''
    print(a, '+', b, '=', a + b)
    return a + b
```

실습 1

Add

```
# Convert the function to a pipeline operation.
add_op = components.func_to_container_op(
    add,
    base_image=BASE_IMAGE,
)

@dsl.pipeline(
    name='Calculation pipeline',
    description='A toy pipeline that performs arithmetic calculations.'
)
def calc_pipeline(
        a: float = 0,
        b: float = 7
):
    #Passing pipeline parameter and a constant value as operation arguments
    add_task = add_op(a, 4) #Returns a dsl.ContainerOp class instance.

    #You can create explicit dependency between the tasks using xyz_task.after(abc_task)
    add_2_task = add_op(a, b)

    add_3_task = add_op(add_task.output, add_2_task.output)
```

실습 1

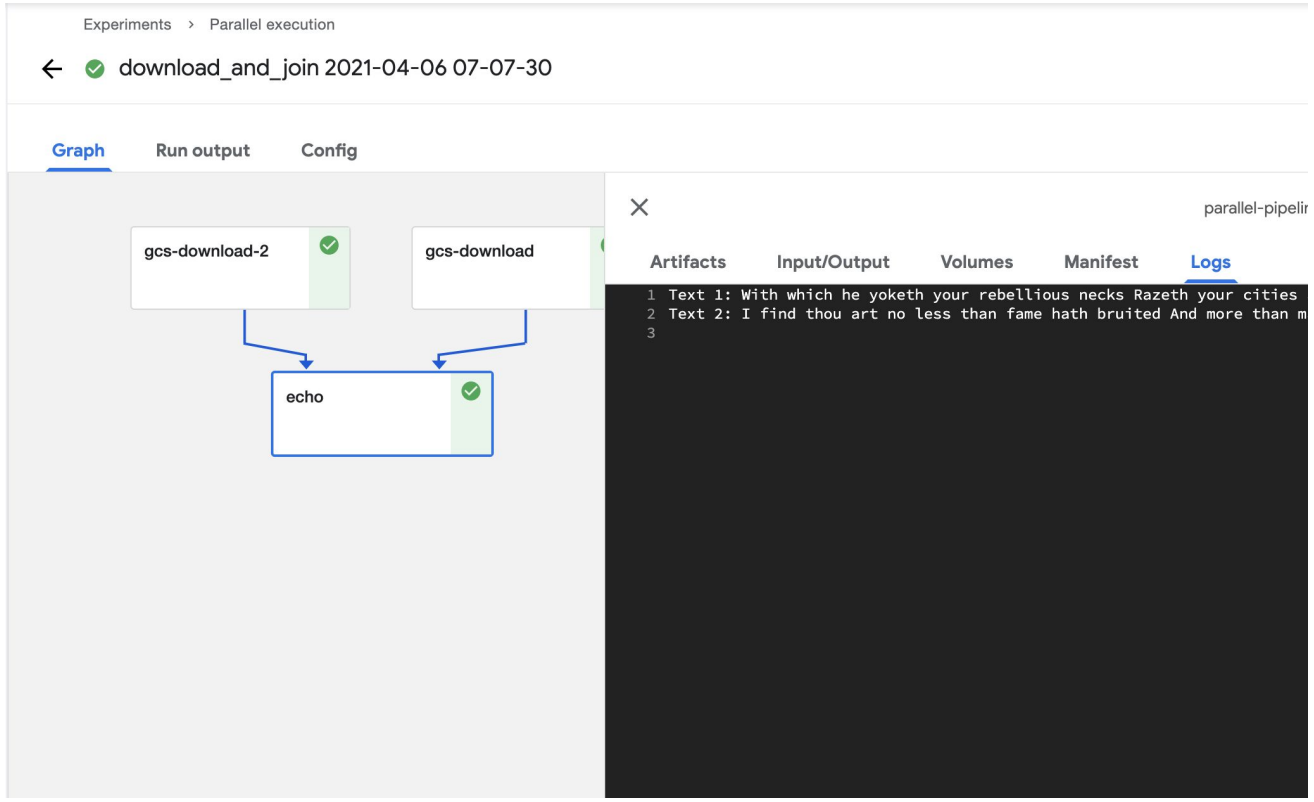
Add

```
if __name__ == "__main__":
    # Specify pipeline argument values
    arguments = {'a': '7', 'b': '8'}
    # Launch a pipeline run given the pipeline function definition
    kfp.Client(host=KUBEFLOW_HOST).create_run_from_pipeline_func(
        calc_pipeline,
        arguments=arguments,
        experiment_name=EXPERIMENT_NAME)
    # The generated links below lead to the Experiment page and the pipeline run details page, respectively
```

실습 1

Parallel

https://github.com/chris-chris/kubeflow-tutorial/tree/master/lesson4_parallel



실습 1

Parallel

```
import kfp
from kfp import dsl


EXPERIMENT_NAME = 'Parallel execution'        # Name of the experiment in the UI
BASE_IMAGE = "python:3.7"
KUBEFLOW_HOST = "http://127.0.0.1:8080/pipeline"


def gcs_download_op(url):
    return dsl.ContainerOp(
        name='GCS - Download',
        image='google/cloud-sdk:272.0.0',
        command=['sh', '-c'],
        arguments=['gsutil cat $0 | tee $1', url, '/tmp/results.txt'],
        file_outputs={
            'data': '/tmp/results.txt',
        }
    )
```

실습 1

Parallel

```
def echo2_op(text1, text2):
    return dsl.ContainerOp(
        name='echo',
        image='library/bash:4.4.23',
        command=['sh', '-c'],
        arguments=['echo "Text 1: $0"; echo "Text 2: $1"', text1, text2]
    )
@dsl.pipeline(
    name='Parallel pipeline',
    description='Download two messages in parallel and prints the concatenated result.'
)
def download_and_join(
        url1='gs://ml-pipeline-playground/shakespeare1.txt',
        url2='gs://ml-pipeline-playground/shakespeare2.txt'
):
    """A three-step pipeline with first two running in parallel."""

    download1_task = gcs_download_op(url1)
    download2_task = gcs_download_op(url2)

    echo_task = echo2_op(download1_task.output, download2_task.output)
```

실습 1

Parallel

```
if __name__ == '__main__':
    # kfp.compiler.Compiler().compile(download_and_join, __file__ + '.zip')
    kfp.Client(host=KUBEFLOW_HOST).create_run_from_pipeline_func(
        download_and_join,
        arguments={},
        experiment_name=EXPERIMENT_NAME)
```

실습 2

Control

실습 2

Control

https://github.com/chris-chris/kubeflow-tutorial/tree/master/lesson5_control_structure

실습 2

Control

```
EXPERIMENT_NAME = 'Control Structure'       # Name of the experiment in the UI
BASE_IMAGE = "python:3.7"
KUBEFLOW_HOST = "http://127.0.0.1:8080/pipeline"

import kfp
from kfp import dsl
from kfp.components import func_to_container_op, InputPath, OutputPath


@func_to_container_op
def get_random_int_op(minimum: int, maximum: int) -> int:
    """Generate a random number between minimum and maximum (inclusive)."""
    import random
    result = random.randint(minimum, maximum)
    print(result)
    return result
```

실습 2

Control

```
@func_to_container_op
def flip_coin_op() -> str:
    """Flip a coin and output heads or tails randomly."""
    import random
    result = random.choice(['heads', 'tails'])
    print(result)
    return result


@func_to_container_op
def print_op(message: str):
    """Print a message."""
    print(message)
```

실습 2

Control

```
@dsl.pipeline(
    name='Conditional execution pipeline',
    description='Shows how to use dsl.Condition().'
)
def flipcoin_pipeline():
    flip = flip_coin_op()
    with dsl.Condition(flip.output == 'heads'):
        random_num_head = get_random_int_op(0, 9)
        with dsl.Condition(random_num_head.output > 5):
            print_op('heads and %s > 5!' % random_num_head.output)
        with dsl.Condition(random_num_head.output <= 5):
            print_op('heads and %s <= 5!' % random_num_head.output)

    with dsl.Condition(flip.output == 'tails'):
        random_num_tail = get_random_int_op(10, 19)
        with dsl.Condition(random_num_tail.output > 15):
            print_op('tails and %s > 15!' % random_num_tail.output)
        with dsl.Condition(random_num_tail.output <= 15):
            print_op('tails and %s <= 15!' % random_num_tail.output)
```

실습 2

Control

```
# %%
@func_to_container_op
def fail_op(message):
    """Fails."""
    import sys
    print(message)
    sys.exit(1)
```

실습 2

Control

```
@dsl.pipeline(
    name='Conditional execution pipeline with exit handler',
    description='Shows how to use dsl.Condition() and dsl.ExitHandler().'
)
def flipcoin_exit_pipeline():
    exit_task = print_op('Exit handler has worked!')
    with dsl.ExitHandler(exit_task):
        flip = flip_coin_op()
        with dsl.Condition(flip.output == 'heads'):
            random_num_head = get_random_int_op(0, 9)
            with dsl.Condition(random_num_head.output > 5):
                print_op('heads and %s > 5!' % random_num_head.output)
            with dsl.Condition(random_num_head.output <= 5):
                print_op('heads and %s <= 5!' % random_num_head.output)
```

실습 2

Control

```
        with dsl.Condition(flip.output == 'tails'):
            random_num_tail = get_random_int_op(10, 19)
            with dsl.Condition(random_num_tail.output > 15):
                print_op('tails and %s > 15!' % random_num_tail.output)
            with dsl.Condition(random_num_tail.output <= 15):
                print_op('tails and %s <= 15!' % random_num_tail.output)

        with dsl.Condition(flip.output == 'tails'):
            fail_op(message="Failing the run to demonstrate that exit handler still gets executed.")


if __name__ == '__main__':
    # Compiling the pipeline
    kfp.compiler.Compiler().compile(flipcoin_exit_pipeline, __file__ + '.zip')
    kfp.Client(host=KUBEFLOW_HOST).create_run_from_pipeline_func(
        flipcoin_exit_pipeline,
        arguments={},
        experiment_name=EXPERIMENT_NAME)
```

짚어보기

쿠베플로우 파이프라인 Part 1

01. 쿠베플로우 파이프라인에 대해 이해한다.

쿠베플로우 파이프라인이 왜 중요한 지, 어떤 기능을 갖고 있는지 이해한다.

02. 쿠베플로우 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 쿠베플로우 파이프라인의 기본 사용법에 대해 공부한다.

머신러닝 파이프라인

쿠베플로우 파이프라인 Part 1

송호연

감사합니다.

THANKS FOR WATCHING