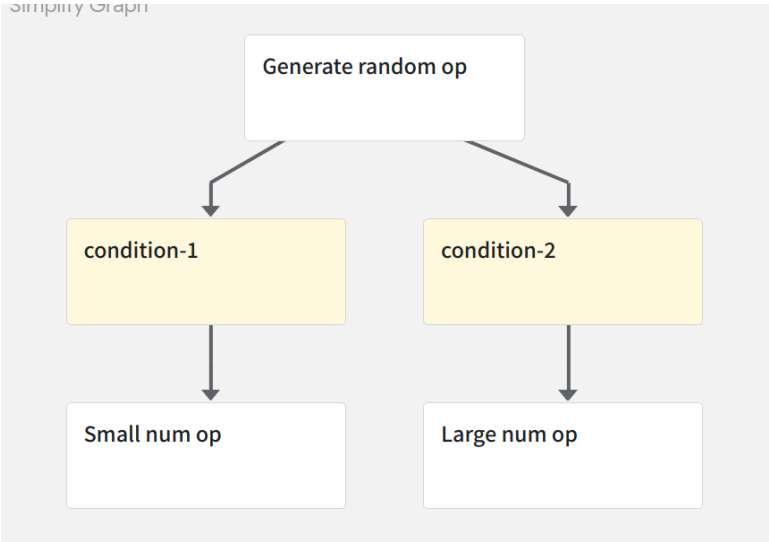
CH 4.6 Kfp 실습 (3)

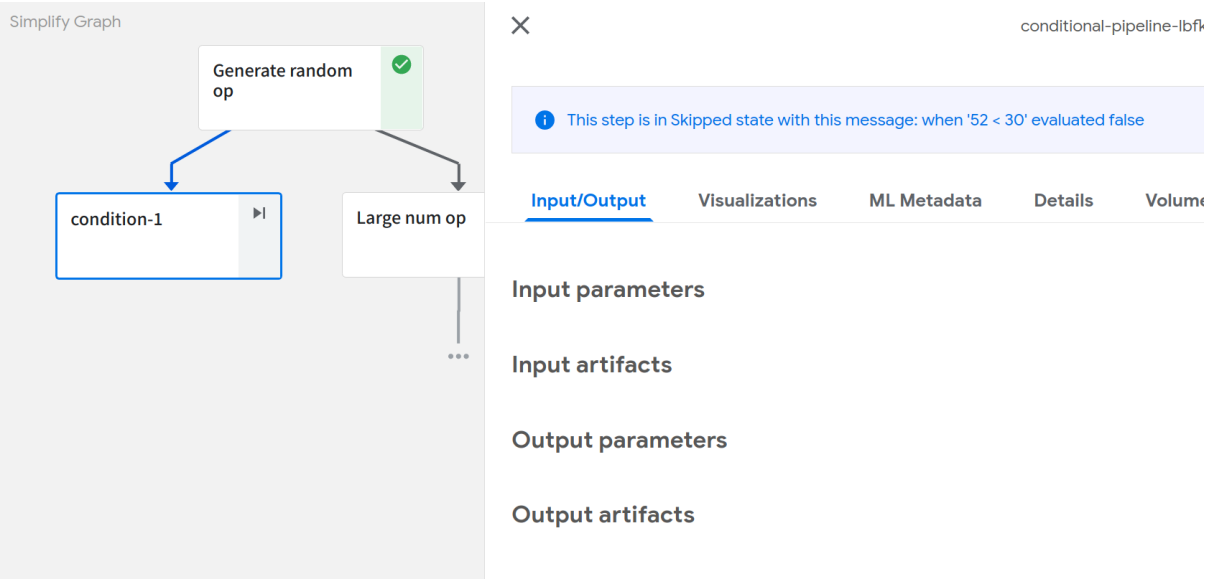
1. Conditional Pipeline
2. Parallel Pipeline

> 이번 시간에는 kfp 파이프라인을 만들 때, component 를 조금 더 디테일하게 제어할 수 있는 kfp.dsl 의 메소드에 대해서 알아보겠습니다.

1. Conditional Pipeline

> 첫 번째 컴포넌트에서 random int 를 generate 한 다음, 첫 번째 컴포넌트의 output 숫자가 30 이상인지 미만인지에 따라 이후 Component 실행 여부가 조건부로 결정되는 pipeline 예제입니다.





▼ conditional.py

```
import kfp
from kfp import dsl
from kfp.components import create_component_from_func


@create_component_from_func
def generate_random_op(minimum: int, maximum: int) -> int:
    import random

    result = random.randint(minimum, maximum)

    print(f"Random Integer is : {result}")
    return result


@create_component_from_func
def small_num_op(num: int):
    print(f"{num} is Small!")


@create_component_from_func
def large_num_op(num: int):
    print(f"{num} is Large!")


@dsl.pipeline(
    name='Conditional pipeline',
    description='Small or Large'
)
def conditional_pipeline():
    # generate_random_op 의 결과를 number 변수에 할당
    number = generate_random_op(0, 100).output

    # if number < 30, execute small_num_op
```

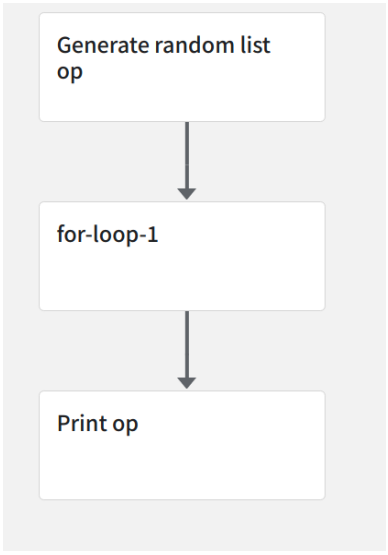
```
    with dsl.Condition(number < 30):
        small_num_op(number)
    # if number >= 30, execute large_num_op
    with dsl.Condition(number >= 30):
        large_num_op(number)


if __name__ == "__main__":
    kfp.compiler.Compiler().compile(
        conditional_pipeline,
        "./conditional_pipeline.yaml"
    )
```

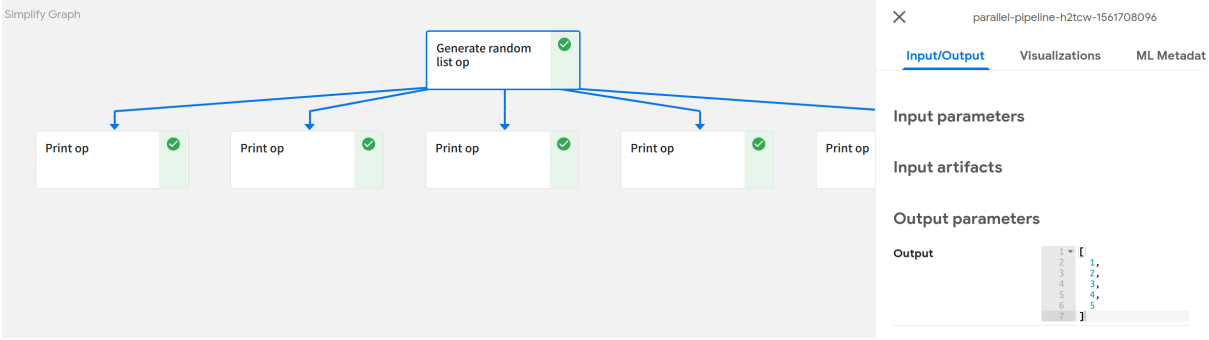
- `dsl.Condition` 메소드를 사용하여 컴포넌트의 실행 여부를 분기처리할 수 있습니다.
- UI 를 통해 pipeline 업로드 후, 실행해봅니다. Generate random op 의 결과에 따라 condition-1, condition-2 컴포넌트의 실행 결과가 달라지는 것을 확인합니다.

---

2. Parallel Pipeline

> 다수의 동일한 컴포넌트 병렬로 실행하는 pipeline 예제입니다.





▼ parallel.py

```
import kfp
from kfp import dsl
from kfp.components import create_component_from_func


@create_component_from_func
def generate_random_list_op() -> list:
    import random

    total = random.randint(5, 10)
    result = [i for i in range(1, total)]

    return result


@create_component_from_func
def print_op(num: int):
    print(f"{num} is Generated!")


@dsl.pipeline(
    name='Parallel pipeline',
)
def parallel_pipeline():
    random_list = generate_random_list_op().output

    # ParallelFor 의 argument 로 [1,2,3] 과 같은 형태의 constant list 를 입력해도 되지만,
    # 이전 component 에서 random 하게 generate 한 list 를 넘겨주는 예시입니다.
    with dsl.ParallelFor(random_list) as item:
        print_op(item)


if __name__ == "__main__":
    kfp.compiler.Compiler().compile(
        parallel_pipeline,
        "./parallel_pipeline.yaml"
    )
```

- `dsl.ParallelFor` 메소드를 사용하여 컴포넌트를 병렬로 실행할 수 있습니다.
- UI 를 통해 pipeline 업로드 후, 실행해봅니다. Generate random op 의 결과에 따라 condition-1, condition-2 컴포넌트의 실행 결과가 달라지는 것을 확인합니다.