스터디 16주차 - ROS2 시험대비 1(입문 2주차)
홍송은

1. 액션 패키지

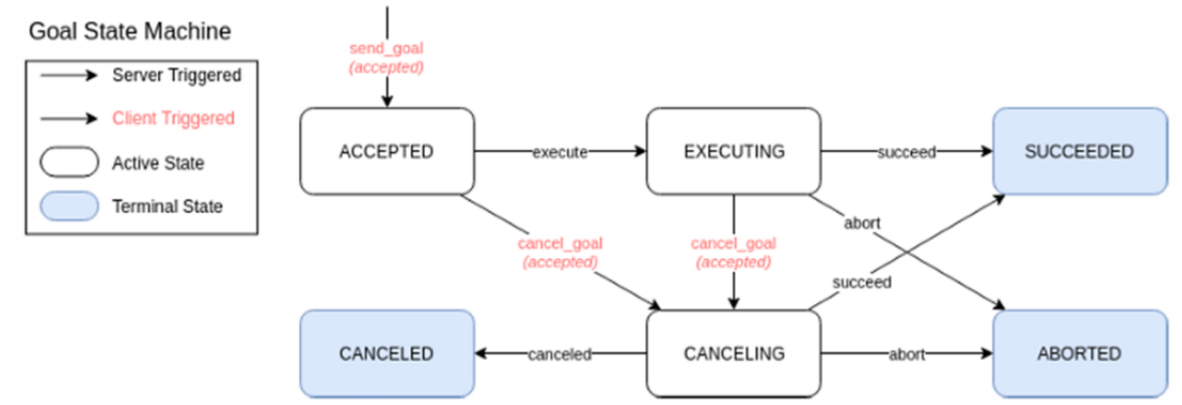
1.1 액션 패키지 생성

- Action 동작 다이어그램

- Goal State Machine
  - 비동기 방식 → 원하는 타이밍에 적절한 액션 수행 어려움
    => 목표 상태(goal state) 사용
    : 목표값을 전달한 후의 상태머신을 구동하여 액션의 프로세스를 쫓는 것

  * 상태머신은 Goal State Machine으로 액션 목표 전달 이후의 액션의 상태값을 액션 클라이언트에게 전달할 수 있어서 비동기, 동기방식이 혼재된 액션의 처리 원활하게 할 수 O

-



  - 활성 상태
    - **ACCEPTED**
      : 목표가 수락되었으면 실행을 기다리는 중
    - **EXECUTING**
      : 목표는 현재 액션 서버에서 실행 중

        - **CANCELING**
          : 클라이언트가 목표 취소를 요청했고, 액션 서버가 취소 요청을 수락.
          액션 서버가 수행해야 할 수 있는 사용자 정의 “정리” 작업에 유용
    - 최종 상태
        - **SUCCEEDED**
          : 액션 서버가 목표를 성공적으로 달성
        - **ABORTED**
          : 목표가 외부 요청 없이 Action Server에 의해 종료됨
        - **CANCELED**
          : 액션 클라이언트의 외부 요청 이후 목표가 취소됨
    - 설계된 동작에 따라 액션 서버에서 트리거되는 상태 전환
        - **execute**
          : 승인된 목표의 실행을 시작
        - **succeed**
          : 목표가 성공적으로 완료되었음을 알림
        - **abort**
          : 목표 처리 중에 오류가 발생하여 중단해야 함을 알림
        - **canceled**
          : 목표 취소가 성공적으로 완료되었음을 알림
    - 액션 클라이언트에 의해 트리거되는 상태 전환
        - **send_goal**
            - 목표가 액션 서버로 전송됨
            - 상태 머신은 액션 서버가 목표를 수락하는 경우에만 시작
        - **cancel_goal**
            - 액션 서버에 목표 처리를 중단하도록 요청
            - 액션 서버가 목표 취소 요청을 수락한 경우에만 전환 발생

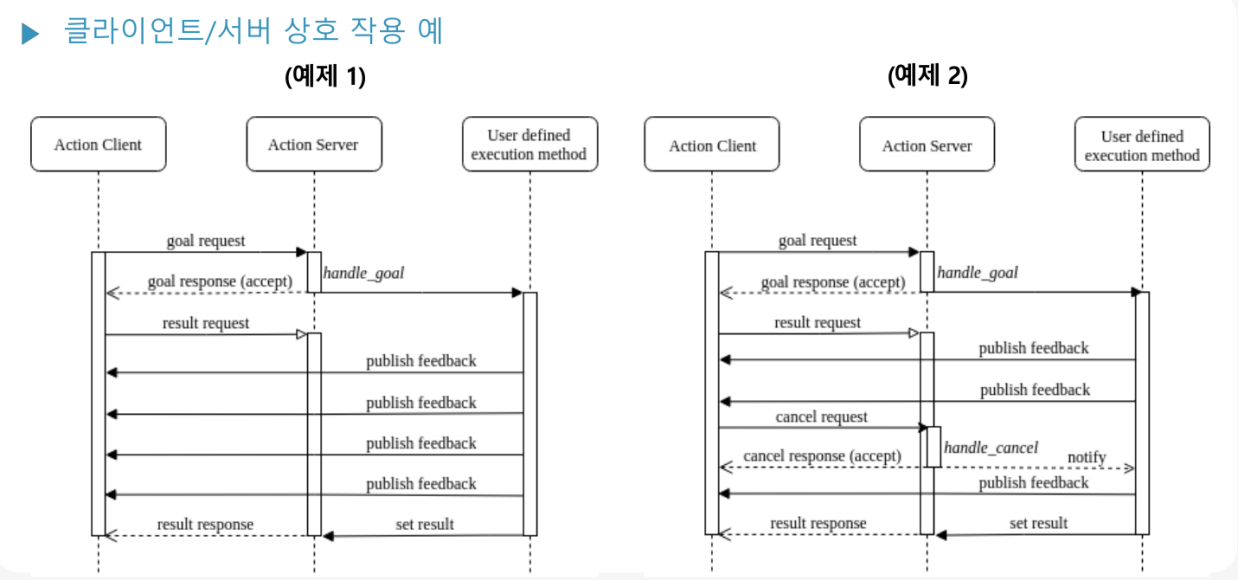
1.2 액션 인터페이스 정의

- Action?
    - ROS 2의 통신 개념 중 하나
    - 장시간 소요되는 비동기 작업을 수행
    - 진행 중 피드백과 취소 기능을 지원하는 구조

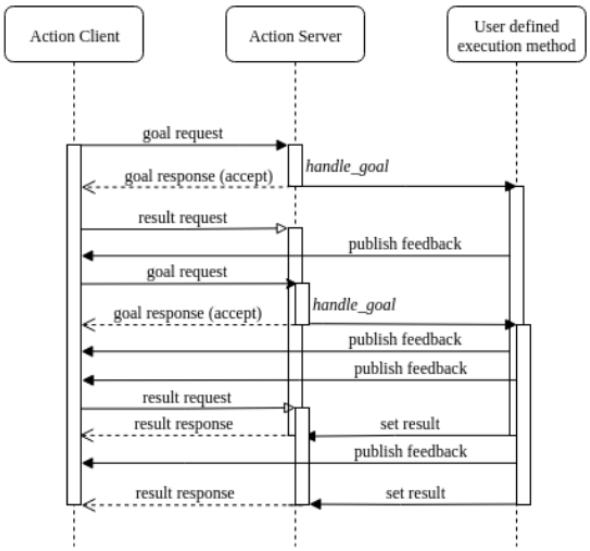
- Action Server

    - 액션 제공
    - 토픽, 서비스처럼 이름과 유형을 가짐
        - 이름: 네임스페이스 지정 가능. 모든 액션 서버 간의 고유
        - 동일한 유형을 가진 여러 액션 서버가 서로 다른 네임스페이스에서 동시에 실행될 수 O
    - 역할
        - 다른 ROS 엔터티에 작업 광고
        - 하나 이상의 액션 클라이언트로부터 목표 수락 또는 거부
        - 목표가 수신되고 수락되면 작업을 실행
        - 선택적으로 모든 실행 작업의 진행 상황에 대한 피드백 제공
        - 선택적으로 하나 이상의 작업을 취소하기 위한 요청 처리
        - 완료된 작업의 결과(성공, 실패 또는 취소 여부 포함)를 결과를 요청한 클라이언트에게 전송
- Action Client
    - 하나 이상의 Goal(수행할 액션) 전송하고 진행 상황 모니터링
    - 서버당 여러 클라이언트가 있을 수 있음. 그러나 여러 클라이언트의 Goal을 동시에 처리하는 방법은 서버가 결정
    - 역할
        - 액션 서버로 Goal 전송
        - 선택적으로 Action Server에서 Goal에 대한 사용자 정의 피드백을 모니터링
        - 선택적으로 Action Server에서 승인된 Goal의 현재 상태를 모니터링
        - 선택적으로 Action Server에서 활성 Goal를 취소하도록 요청
        - 선택적으로 Action Server에서 수신한 Goal에 대한 결과를 확인
- 클라이언트/서버 상호 작용 예

▶ 클라이언트/서버 상호 작용 예



(예제 3)



→ 액션: 응답에 시간이 걸릴 수 있을 때 유용

클라이언트가 요청 진행 상황을 추적, 최종 결과 확인, 요청이 완료되기 전에 취소할 수 있도록 해줌

- Action interface 정의

action은 ROS 메시지 IDL 형식을 사용하여 지정 → 세 섹션으로 구성되며, 각 섹션은 메시지 사양

① GOAL

- 액션이 무엇을 달성해야 하는지, 어떻게 달성해야 하는지 설명
- 액션 실행 요청을 받으면 액션 서버로 전송됨

② RESULT

- 작업의 결과 반환
- 작업 실행이 성공적으로 종료되었는지 여부와 관계없이 서버에서 클라이언트로 전송됨

③ FEEDBACK

- 작업 완료까지의 진행 상황
- 작업 실행 시작전부터 완료 전까지 서버에서 해당 작업의 클라이언트로 전송됨
- 클라이언트는 이 데이터를 사용하여 작업 실행 진행 상황 파악

이 섹션들은 모두 비어있을 수 있음

세 섹션 사이에는 세 개의 하이픈(---)이 포함된 구분자 존재

▶ action_tutorials_interfaces 의 Fibonacci.action 파일 확인

```
int32 order
---
int32[] sequence
---
int32[] partial_sequence
```

- **Action 동작 흐름**
  ① **Goal 전송: 클라이언트가 서버에 작업 요청.**
  ② **Goal 수락: 서버가 요청 수락/거절.**
  ③ **실행 및 피드백: 서버가 작업 실행하며 클라이언트에 피드백 전달.**
  ④ **결과 전송: 작업 완료 시 성공/실패/취소 결과 전송.**
  ⑤ **취소 처리: 클라이언트가 요청 시 서버는 Goal 취소 처리 가능.**

1.3 액션 패키지 생성

- ros2 action 명령어

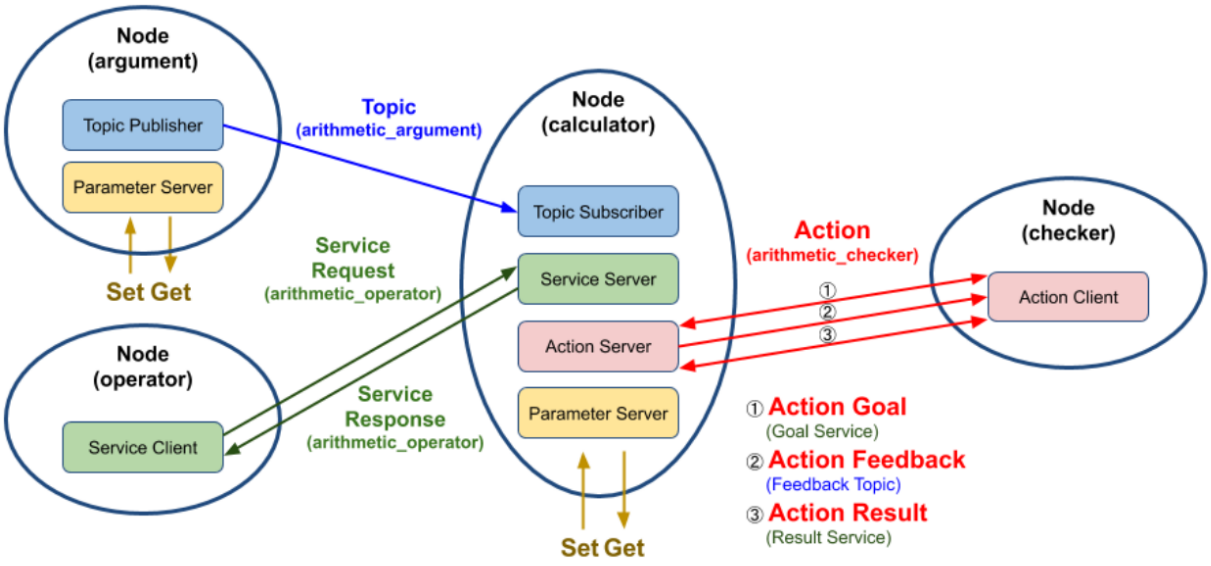
명령어	설명
ros2 action list	액션 서버/클라이언트 목록
ros2 action info	액션 타입 조회
ros2 action show	액션 메시지 구조 확인
ros2 action send_goal	액션 실행 및 피드백/결과 확인
ros2 action status	Goal 상태 모니터링
ros2 action feedback	실시간 피드백 출력

2. 인터페이스 프로그래밍(응용_1)

2.1 인터페이스 프로그래밍(ex_calculator)

- 패키지 설계
  - 계산기 개발
  - 현재 시간과 변수 a, b를 받아 연산 결과값 도출
  - 연산 결과값 누적하여 목표치에 도달했을 때 이 결과값 표시

▶ 패키지 구성



2.2 토픽 프로그래밍

- 토픽 publisher/subscriber

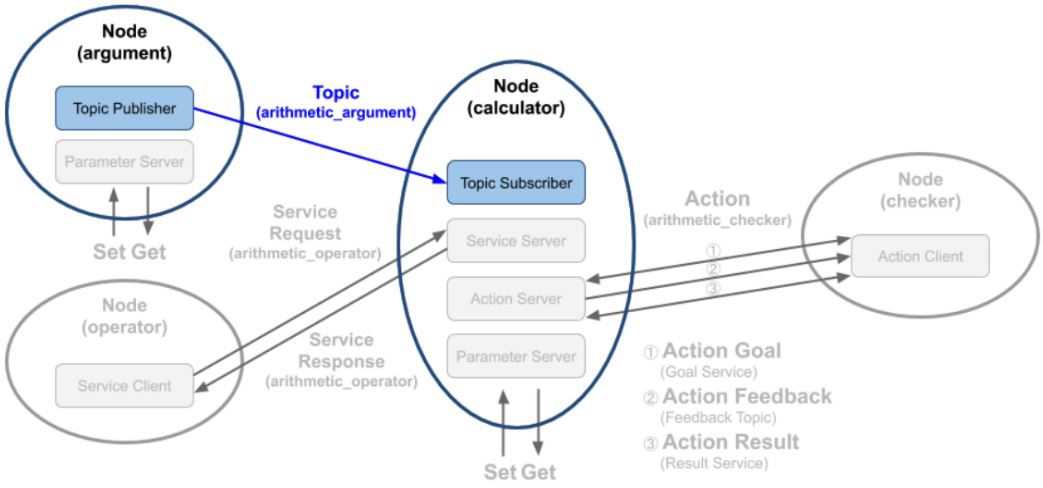
▶ 토픽 퍼블리셔
(데이터를 송신하는 프로그램)

- Node 설정
- QoS 설정
- create_publisher 설정
- 퍼블리시 함수 작성

▶ 토픽 서브스크라이버
(데이터를 수신하는 프로그램)

- Node 설정
- QoS 설정
- create_subscription 설정
- 서브스크라이브 함수 작성

▶ 토픽 퍼블리셔 / 토픽 서브스크라이버

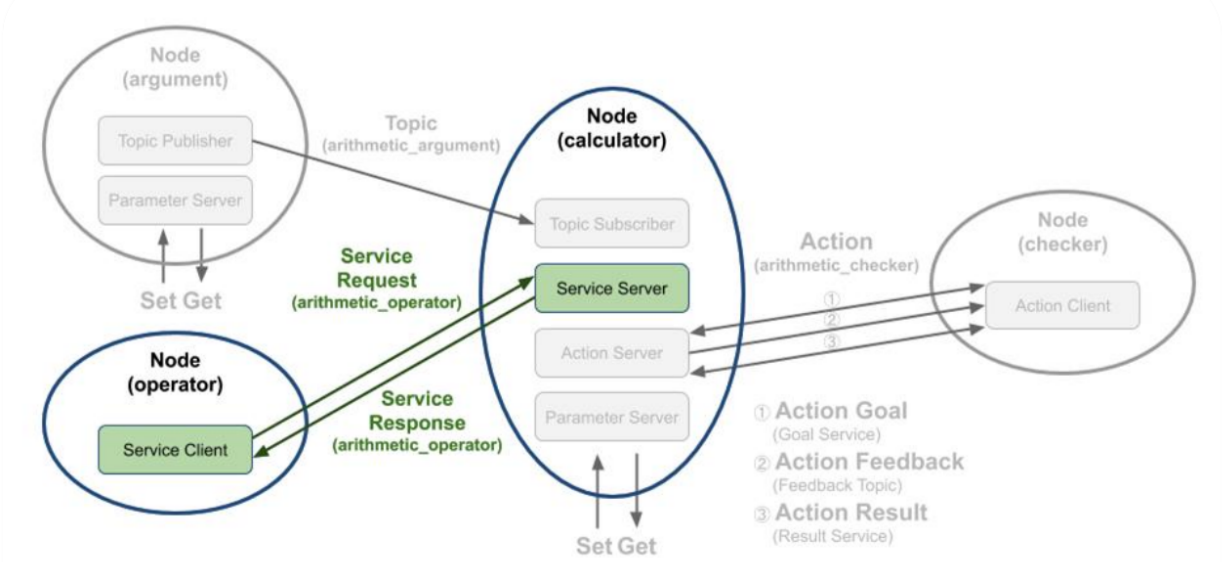


  - calculator
    : 토픽 서브스크라이버 노드
  - argument
    : 토픽 퍼블리셔 노드
  - operator node
    : arithmetic_operator이라는 서비스 이름으로 calculator 노드에게연산자(+, -, *, /)를 서비스 요청값으로 보냄
  - calculator node
    : 서브스크라이브하여 저장하고 있는 변수 a, b와 operator 노드로부터 요청 값으로 받은 연산자를 이용하여 계산(a 연산자 b)하고 operator 노드에게 연산의 결과값을 서비스 응답(Response) 값으로 보냄

- 콜백 병렬 처리 설정 및 QoS 설정

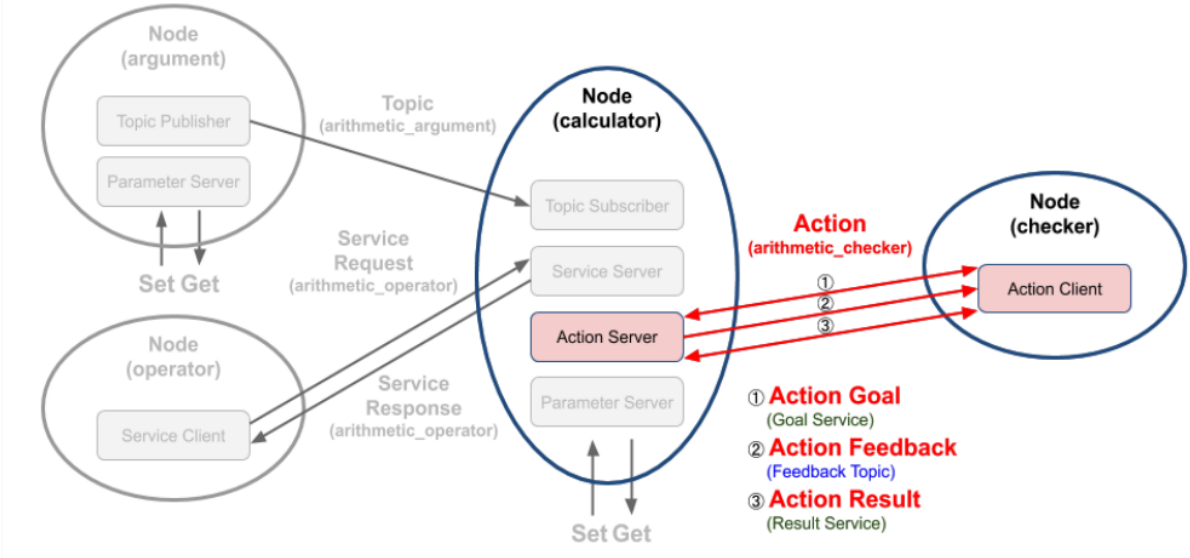
- calculator 노드는 토픽, 서비스, 액션 요청을 병렬로 처리하기 위해 'ReentrantCallbackGroup'을 사용
- 이를 위해 'MultiThreadedExecutor'를 설정하고 콜백 그룹과 함께 병렬 실행 가능
- 퍼블리셔와 서브스크라이버 모두 동일한 QoS 설정 사용:
  - Reliability: RELIABLE
  - History: KEEP_LAST
  - Depth: 10
  - Durability: VOLATILE

2.3 서비스 프로그래밍



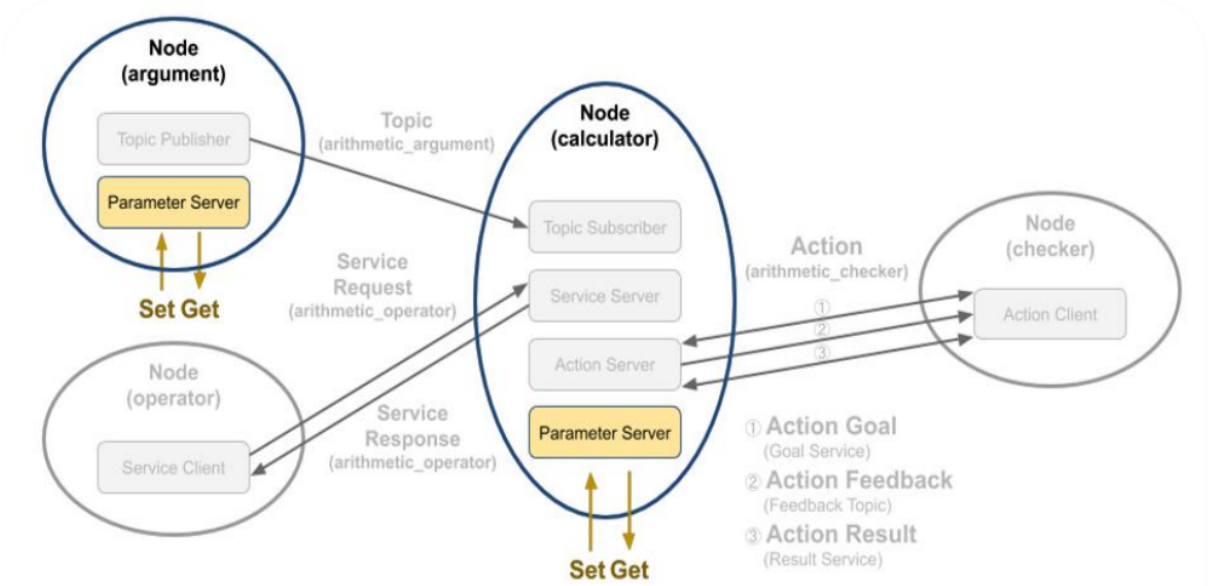
- operator 노드는 사칙연산자 중 하나를 선택하여 `arithmetic_operator` 서비스 요청으로 calculator 노드에 전송
- calculator 노드는 미리 저장한 a, b 값과 받은 연산자를 기반으로 계산 수행 후, 결과값을 서비스 응답 값으로 되돌려줌
- 이때 `create_service()`와 `get_arithmetic_operator()` 콜백 함수가 사용됨

2.4 액션 프로그래밍



- 액션 목표(action goal)를 지정하는 액션 클라이언트와 액션 목표를 받아 특정 태스크를 수행하면서 중간 결과값에 해당되는 액션 피드백(action feedback)과 최종 결과값에 해당되는 액션 결과(action result)를 전송하는 액션 서버

2.5 파라미터 프로그래밍



argument 노드는 QoS 설정과 랜덤으로 생성되는 변수 a, b의 랜덤 생성 범위를 파라미터 calculator 노드는 QoS 설정을 사용

- argument 와 calculator 노드에 Parameter Server가 내장되어 있음
- 서버를 통해 외부에서 해당 노드의 설정을 동적으로 조회 하거나 변경 할 수 있다는 것을 보여줌
- 시스템을 유연하게 조정하고 런타임 중 동작을 제어하는 데 매우 유용한 기능

2. 6 실행 인자 프로그래밍

- 프로그램 실행 시 추가로 입력되는 인수로, main 함수의 매개변수로 사용됨
- 실행 명령어와 함께 전달되어 프로그램 동작에 영향을 줌

```
# -g 100은 "GOAL_TOTAL_SUM" 값을 100으로 설정
& ros2 run ex_calculator checker -g 100
```

- 실행 인자 구문
  ① 파서 만들기 (parser = argparse.ArgumentParser)
  : ArgumentParser 객체를 만들어 인자 파싱 준비를 함
  ② 인자 추가하기 (parser.add_argument)
  : add_argument() 로 실행 시 사용할 인자 이름, 타입, 기본값, 설명 등을 설정
  ③ 인자 파싱하기 (args = parser.parse_args())
  : parse_args() 를 호출해 실제로 입력된 인자들을 파싱
  ④ 인자 사용하기 (args.xxx)
  : 파싱된 인자는 args.인자이름 형식으로 접근하여 사용할 수 있음

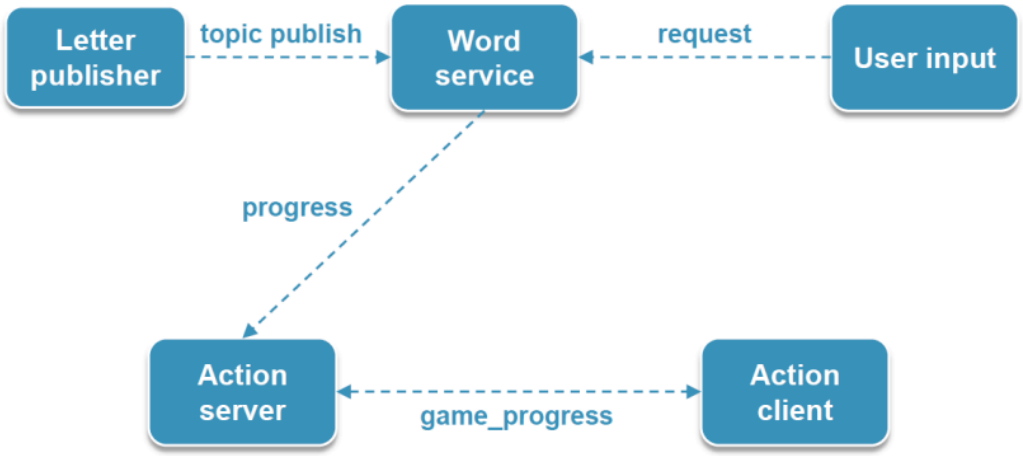
2.7 런치 파일 프로그래밍

- 런치 파일을 통해 여러 노드(argument, calculator 등)를 한 번에 실행 가능

- 실행 시 함께 파라미터 파일(.yaml)을 참조하여 초기값 자동 설정
- `ros2 launch ex_calculator arithmetic.launch.py` 명령으로 실행

3. **인터페이스 프로그래밍(응용_2)**

3.1 인터페이스 프로그래밍(hangman)



- Letter Publisher
  : a부터 z까지의 알파벳을 순서대로 publish
- Word Service
  : 임의의 단어를 선택, 행맨 게임 진행, 진행 상황 publish
  - letter_topic 수신 → 현재 글자로 CheckLetter 서비스 요청 응답 처리
  - 게임 상태는 Progress 메시지로 퍼블리시
- Action Client
  : Goal 설정, action server와의 상호작용을 통해 행맨 게임 진행 상태 업데이트
  - send_goal: 목표를 서버로 전송
  - feedback_callback: 서버로부터 진행 피드백 수신
  - get_result_callback: 최종 게임 결과 수신 (승/패 여부 확인)
- Action Server
  : 사용자의 진행 상황을 관리, 임의 진행 상태를 추적, 게임 결과를 클라이언트에게 전달
  - progress 토픽을 구독하여 실시간 게임 상태를 추적
  - 목표 수신 시 execute_callback 실행, 현재 게임 상태를 주기적으로 피드백으로 전송
  - game_over 또는 won 상태에 도달하면 최종 결과를 Result로 반환
- User Input
  : 사용자로부터 입력을 받고, 해당 문자를 CheckLetter 서비스로 요청
  - Enter 입력 시 현재 글자가 선택된 단어에 있는지 확인 요청
  - 결과 메시지는 Word Service로부터 받아옴

3.2 실행 순서

① letter_publisher 실행
② word_service 실행
③ progress_action_server 실행

④ progress_action_client 실행
⑤ user_input 실행

4. **ROS2 응용**

4.1 ROS2 bag

- ROS2 BAG은 시스템의 토픽에 게시된 데이터를 기록하기 위한 플랫폼 기능
- 각 토픽에서 수집된 데이터를 데이터베이스에 저장하며, 이를 재생하여 테스트 및 실험 결과를 재현할 수 있음
- ROS2 BAG(Turtlesim)
  - Turtle의 움직임을 BAG 파일에 기록
  - 기록된 BAG 파일을 이용하여 turtle 움직이기
- 주요 명령어
  - 기록: ros2 bag record -o <bag_name> <topic_name>
  - 재생: ros2 bag play <bag_name>
  - 정보 확인: ros2 bag info <bag_name>