Action Server and Client

- The limitation of Simple Action Server is that it can handle only one goal at a time.
- If you have a use-case where you want the Action Server to handle more that one goal simultaneously then use Action Server and **not** Simple Action Server.
- Action Server gives you ability to set your own Goal Policy. For eg., you can have a Goal Policy in which the Server process only one Goal at a time but stores all the incoming Goals in a queue to process them later.
- Recall, in Simple Action Server the Goal Policy is already set. You can only process one goal at a time and it won't maintain any queue for incoming goals.
- In Action Server you can have a Goal Policy in which incoming new Goal is handled by a new thread this allows Action Server to process multiple Goals at a time. This kind of behaviour is suited to process I/O Bound Tasks.
    - Consider the example in which you want a Robotic Arm connected to the internet to push the information about the package it has picked from the conveyor belt.
    - In this case you would not want your Robotic Arm to wait for the operation to push the data.
    - If the internet speed is slow it may take very long to push the data. In such scenario you would not want to Robotic operation to wait. Rather you would want your Robotic operation to continue while the data is being pushed.
    - This is where you can use ROS Actions.
        - The pushing of data to the cloud database can be handled by ROS Action Server whose job can be to do such networking tasks.
        - The Robotic Arm node can have a ROS Action Client in it which can send goals to push data on the cloud database to the Action Server.
        - After the client has sent the goal it can move on to the next operation and won't have to wait.
        - The Action Server can take this goal and create a new thread to process it.
