* Compiler –
  + Converts the code to object code which is machine readable format. Creates stand alone machine code.
  + Compile programs run faster than interpreted code.
  + Once the code is compiled and converted to objects, source code is not really necessary.
  + Eg C, C++, GO
  + Limitations 🡪
    - Platform Dependency.
    - Cold Start
* Interpreter –
  + Directly executes instructions without converting into object code.
  + Code executed line by line.
  + Eg. Python, PHP
  + Limitations 🡪
    - Execution time of interpreted language is slower
* Cross-Compiler / Retargeting –
  + A compiler which generates executable code that works not only on the machine/platform on which it runs but also other machines.
* Static Type checking –
  + Checks for any type error in the code.
* Dynamic Type Checking –
  + Checks type safety at runtime
* Go –
  + Open Source-programming language.
  + Static typing and run-time efficiency
  + Readability and usability
  + Since its new language there is a limitation on availability of Libraries
* AWS Lambda
  + Lambda allows you to run code without thinking about servers.
  + Allows continuous scaling, parallel processing.
  + Use Case -
    - Data processing – Lambdas can be triggered when there is change in data or when some action is performed by the user.
    - For eg. New file in S3, or DynamoDB
* **Event Driven** –
  + Request is send from clients to services, the server will send a response only when some event has occurred which concerns the client or any other client which is responsible
  + Git, React, Gaming servers is an Example
  + An event can be associated with timestamp. This helps in undoing the steps. For eg: 2 players at playing counter strike. Player 1 shoots player 2 at time t. by the time the request reached the server it was time t+1 and the player 2 moved by 1 position. When using an event driven architecture the server can understand at what time player 1 shot player 2 and check player 2’s position.
  + Presence of Event bus (**Event Bridge**). An event bus receives events from sources and routes it to rules associated with the evnt bus
  + Advantages – Easy Rollback, Availability,
    - Transactional guarantee –
      * At most 1 ( The message will be sent once if there is failure, no retry)
      * At least 1 (the message will defo be sent, incase there is a failure in between then the server will keep trying)
  + Disadvantages – Consistency issues(since we have multiple servers), not too much control
* Live Stream Architecture
* Publisher
* Server
* Subscriber
  + Publisher/Subscriber should be able to register, login to their channel. Web console -> get userid, password -> use lambda to authenticate -> dynamobd
    - When creating a channel people will get SNS ad SQS
  + Subscriber should be a able to subscribe to different channels. SQS will place its queue in SNS to get notifications when new videos are posted
  + So when Publisher starts a live stream a notifications is send from SNS to all SQS.
  + A Subscriber will click on Join event which will result in a web socket being created