**Interview Assignment for TechOps**

A customer is interested in a secure, automated labeling pipeline.  They have chosen AWS as the services provider, and Sagemaker Ground Truth (SMGT) as the labeling system.

The requirements are for a system that creates a single labeling job with three tasks in it on a daily basis.  In each unique task, a labeler is asked to identify whether a single sentence is talking about sushi or pizza.

If two workers have 100% the same answer for a task, then the result is assumed to be correct.  However, for all tasks where the answer differs 50/50 between the two workers, the task must be re-submitted as a new “TieBreaker” job with one worker.

A single file combined output file (aka, manifest) for each job must be created, where it contains all 100% consensus answers from the initial job run, and for the answers that were not 100%, those answers replaced with the answers from the TieBreaker run.

For example, in the initial SMGT run, the output manifest includes the following results:   (file format provided is just an example for simplicity)

{  
    "sentence 1": [{  
            "worker 1": "pizza"  
        },  
        {  
            "worker 2": "pizza"  
        }  
    ]  
}  
  
{  
    "sentence 2": [{  
            "worker 1": "sushi"  
        },  
        {  
            "worker 2": "pizza"  
        }  
    ]  
}  
  
{  
    "sentence 3": [{  
            "worker 1": "sushi"  
        },  
        {  
            "worker 2": "sushi"  
        }  
    ]  
}

Since sentence 2 was not unanimous, sentence 2 content is created as a new TieBreaker job with one labeler, and that labeler responds with:

{  
    "sentence 2": [{  
            "worker 1": "sushi"  
        }  
    ]  
}

The final combined output manifest would look like:

{  
    "sentence 1": [{  
            "worker 1": "pizza"  
        },  
        {  
            "worker 2": "pizza"  
        },  
        "majorityAnswer": "pizza"  
    ]  
}  
  
{  
    "sentence 2": [{  
            "worker 1": "sushi"  
        },  
        "majorityAnswer": "sushi"  
    ]  
}  
  
{  
    "sentence 3": [{  
            "worker 1": "sushi"  
        },  
        {  
            "worker 2": "sushi"  
        },  
        "majorityAnswer": "sushi"  
    ]  
}

* How would you solve this issue?  Provide an architectural diagram with details on what each component does to accomplish the end result.
* Provide some example Python code that would take the inputs provided in the example above, and output the final example result into S3.  Upon writing it to S3, send an SNS message to signal the job is complete.  A Python Lambda should receive this SNS message and simply return “Hello World”.
* For bonus points, implement #1 fully or partially into a real AWS-based running system using Python and AWS Python Boto3 library.