1.       Above all first we have to save the model checkpoint and make sure that we have our final checkpoint saved after training.

Hint: you find the name of checkpoint -12345.meta(- followed by numbers) other files with. Ckpt extension are the intermediate check points of training process.

2. Since our saved\_model is from training it does not have the tensors and nodes that we use in testing. So, we have to save a new graph that has test graph in it. Save a small graph that can be used for serving. To do this we have to run Spoken\_gpt2\_test.py which does a local test along with saving the model with small test graph.

3. Adding signature definition. Finalize input and output tensors that we are going to use in the graph and then add tags to the graph. We can use Serve\_support\_gpt-3.py to add\_tags or print tesnorlist to know names of input and output tensor names. Once model is saved it is ready to be served either on docker or aws sagemaker.

4. **model; to be hosted as a Docker image with TensorFlow Serving**

Refer to Run\_commands.txt and get the commands to run a saved model on docker and use client\_gpt-3.py to communicate with docker image. If model is working fine with docker image then it works same way with aws too.

4.**If model; to be hosted on AWS SageMaker**

save the model save it in a specific format so that aws can read it. (export/serve/model version/variables)

Note- entry.py is an entry point, since this is just a predict request it is a empty python file.

Once the end point is created, we can use it and data should be parsed in json format. In aws we have a lambda function (sagemaker\_endpoint) as of now this works fine.