Inside the output.tar.gz following are the contents:

config.json
model.safetensors
processor_config.json
special_tokens_map.json
tokenizer.json
tokenizer_config.json
training_args.bin

Those files indicate a Hugging Face–style transformer checkpoint, not an ONNX export, so you do **not** currently have an ONNX model.[1] [2]

What you actually have

- `model.safetensors`: model weights in SafeTensors format (commonly used by Transformers).[1]
- `config.json`: model architecture/config (hidden size, num layers, etc.).[1]
- `tokenizer.json`, `tokenizer_config.json`, `special_tokens_map.json`, `processor_config.json`: tokenizer / processor definitions.[1]
- `training_args.bin`: Trainer args and metadata.[1]

This is the standard layout produced by `transformers.Trainer` or similar, not ONNX.[2] [1]

How to get ONNX from this

You need to export ONNX yourself using Hugging Face tooling, either by:

1. **Using `optimum` / transformers ONNX export** (recommended):
   - In Python: load the model from this folder and run `optimum.onnxruntime.export` or `transformers.onnx.export`.[1]
   - Point it at a dummy input and export to a `.onnx` file.
2. **Integrating export into your SageMaker training script**:
   - After training completes and you have `model.safetensors` etc. in `/opt/ml/model`, load the model and run the ONNX export step there so the *next* `output.tar.gz` includes `model.onnx`.[3] [1]

Without that explicit export, SageMaker will never “magically” add ONNX; it only tars whatever your script wrote to `/opt/ml/model`.[4] [3]

Options for your Spring Boot service

Given your current artifact:

- **If you can modify the Python code / re-run in SageMaker**:
  - Add an ONNX export step, regenerate `output.tar.gz`, then consume `model.onnx` with ONNX Runtime Java as we discussed.[3] [1]
- **If you cannot retrain but can run a one-off conversion**:
  - Download `output.tar.gz`, extract locally, use a small Python script (on your laptop/EC2) to load the HF model from that directory and export ONNX, then upload the `.onnx` to S3 for your Spring Boot service.[5] [6] [1]
- **If you cannot run any Python at all**:
  - You'll have to serve inference in a Python service (using these HF artifacts directly) and call it from Spring Boot, or reverse-engineer weights, which is not practical.

If you share which exact model class it is (e.g., `AutoModelForSequenceClassification` + which checkpoint), I can outline the exact Python export snippet you should add.

⁂

1. https://aws.plainenglish.io/deploying-transformers-onnx-models-on-amazon-sagemaker-7689e8710328
2. http://www.nbertagnolli.com/jekyll/update/2023/06/05/Deploy_ONNX_Transformer_to_Sagemaker.html
3. https://aws.amazon.com/blogs/machine-learning/host-ml-models-on-amazon-sagemaker-using-triton-onnx-models/
4. https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-training-algo-output.html
5. https://stackoverflow.com/questions/73153452/how-to-convert-sklearn-model-using-pipeline-to-onnx-format-for-real-time-inferen
6. https://calmcode.io/course/scikit-save/onnx-sklearn