Getting Your Model to Production with AWS Sagemaker

Dr. Richard Sieg

HOW.FM

rs@how.fm

HOW.FM

Multi-language training solutions for manual workers

DHL · KUEHNE+NAGEL · INGRAM MICRO · LOXXESS logistics & fulfilment · BOHNEN LOGISTIK · FRESSNAPF · BLOOM & WILD

Training and ongoing support in >30 languages — on autopilot

Turning any global talent into an expert — on autopilot

+ talent pool
- effort and cost

Training off-the-job

how.fm Theory

100% compliance
-5% churn

Training on-the-job

how.fm Practice

-15% staff ramp time
+30% quality

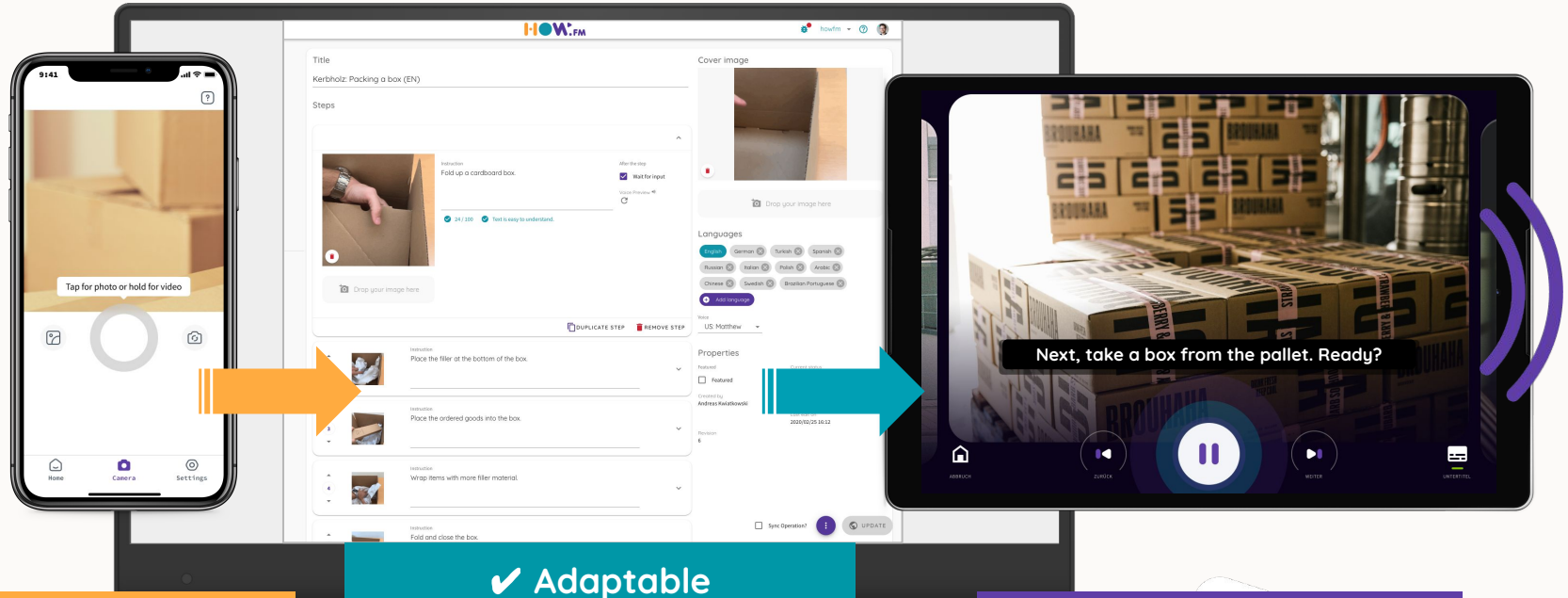
Always-on live support

how.fm Live

-50% interruptions
Real-time insights

HOW.FM

One SaaS platform to define, implement, and improve standards



✔ Modular media capturing

✔ Adaptable video-like sequences with synthesized voice-over

✔ Off- and on-the-job, with voice control

ML Projects @how.fm

HowBERT

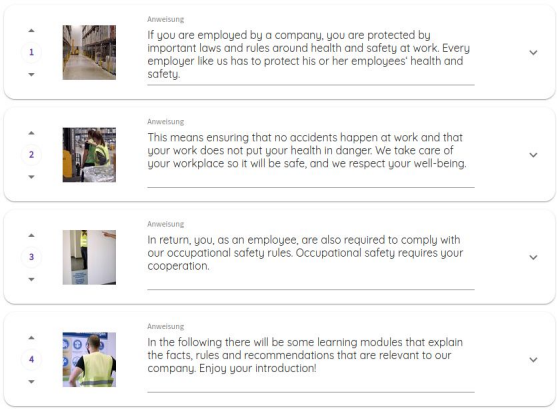
3	Fold the right side of the business shirt to the edge of the frame and keep a small distance.	Fold · the right side of t… · keep · a small distance
4	Tuck the sleeve down to the right side of the shirt.	Tuck · the sleeve · down
5	Repeat the folds on the left side of the shirt.	Repeat · the folds
6	Fold the bottom of the business shirt so that it reaches the collar. Fold the end nicely.	Fold · the bottom of th… · Fold · the end
7	Done. Put the business shirt in the box.	Done · Put · the business shirt

Detect all *actions* in natural text.

And for every action their *parameters.*
(object, location, duration etc)

HOW.FM

Translation Matching



Check whether translations are still in sync in 109 languages.

Current Machine Translation models are not good enough for how.fm crucial languages (🇧🇬🇺🇦🇷🇸...)

👉 Our process requires human translation.

👉 Only re-translate edited sections.

Document Extraction

1.4 RESPONSIBILITIES
It is the responsibility of the tea maker to use this SOP when making a cup of tea for a research study in the TBBR

1.5 PROCEDURE
Equipment and consumables:

Consumables:
Tetley tea bag
Water
Semi-skimmed milk

Equipment:
Kettle
Mug
Cylinders
Teaspoon

1. Take care during the tea making process as hot water can cause scalds and burns.
2. Put the required amount of water in an electric kettle and switch on to boil.
3. Use a mug that has been selected for use on the study that is of a standard shape.
4. Place a round caffeinated Tetley tea bag into the mug chosen for use on the study.

Extract **all paragraphs from customer documents.**

And *classify* which paragraphs contain instructions.

👉 Auto-Import of customer documents

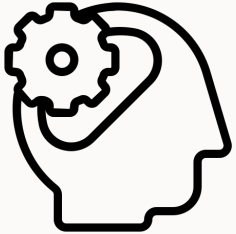
HOW.FM

Model Hosting

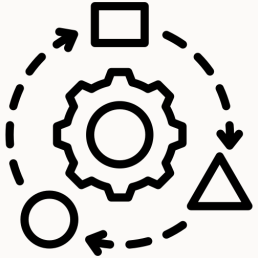
Example API - usable as guideline & template to host NLP models


LANGUAGE MODELS
ML COMMUNITY
CODING
DEPLOYMENT
imgflip.com

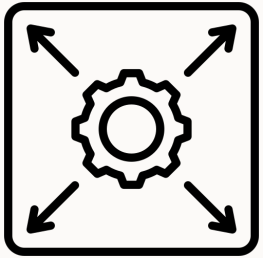
Requirements



Simple



Adaptable



Scalable

Why not use the GUI Console?

Best Practice…

- Reproducibility
- Stage management (Develop, Production etc…)
- Don’t touch Production
- Code Review
- Continuous integration

The Key Players

AWS SageMaker:
Platform for training & deploying ML models

🆕 **Serverless Deployments**

AWS Cloud Development Kit (CDK):
Framework to model cloud applications programmatically

HuggingFace:
Hub for ML/NLP models & tools for NLP applications

The Key Components

Model Code

- SageMaker API
- Load 🤗 Model
- Transform predictions

API Code

- Call SageMaker Endpoint
- Logging
- Auxiliary API Code

Infrastructure

- Define cloud application
- Deployment
- Stage management

Model Code

Model Handler

```python
class ModelHandler(object):
    def __init__(self):
        self.initialized = False
        self.model = None
        self.tokenizer = None

    def initialize(self):
        self.tokenizer =
      AutoTokenizer.from_pretrained(TOKENIZER)
        self.model =
      AutoModel.from_pretrained(MODEL)
        self.model = self.model.eval()
        self.initialized = True

    def preprocess(self, data):
        return data["inputs"]
```

🤗Model

```python
    def inference(self, sentences):
        inputs = self.tokenizer(sentences,
            return_tensors="pt", padding=True)
        with torch.no_grad():
            logits = self.model(**inputs).logits
        id2label = self.model.config.id2label
        predicted_classes = [id2label[x.item()] for x in
                logits.argmax(dim=1)]
        return predicted_classes

    def handle(self, data):
        instances = self.preprocess(data)
        predictions = self.inference(instances)
        return {"predictions": model_out}
```

Transformation

Adjust this for different kind of models!

Model API

```python
handler = ModelHandler()
app = Flask(__name__)
@app.route("/invocations", methods=["POST"])
def invoke():
    body = request.get_json(force=True)
    output = handler.handle(body)
    return output

@retry(stop_max_delay=1000 * 30, retry_on_exception=_retry_if_error)
def _start_serve():
    if not handler.initialized:
        handler.initialize()
    serve(app, host="0.0.0.0", port=8080)
```

Flask API

Endpoint for SageMaker

Initialize

Dockerfile

```
<...>
EXPOSE 8080
RUN mkdir -p /opt/ml/model
RUN mkdir /app
WORKDIR /app
RUN pip --no-cache-dir install -r requirements.txt

COPY src /app/src
COPY api.py api.py
RUN chmod +x api.py

ENTRYPOINT ["python", "api.py"]
CMD ["serve"]
```

The Docker image is then registered in ECR and used by SageMaker

API Code

SageMaker Helper

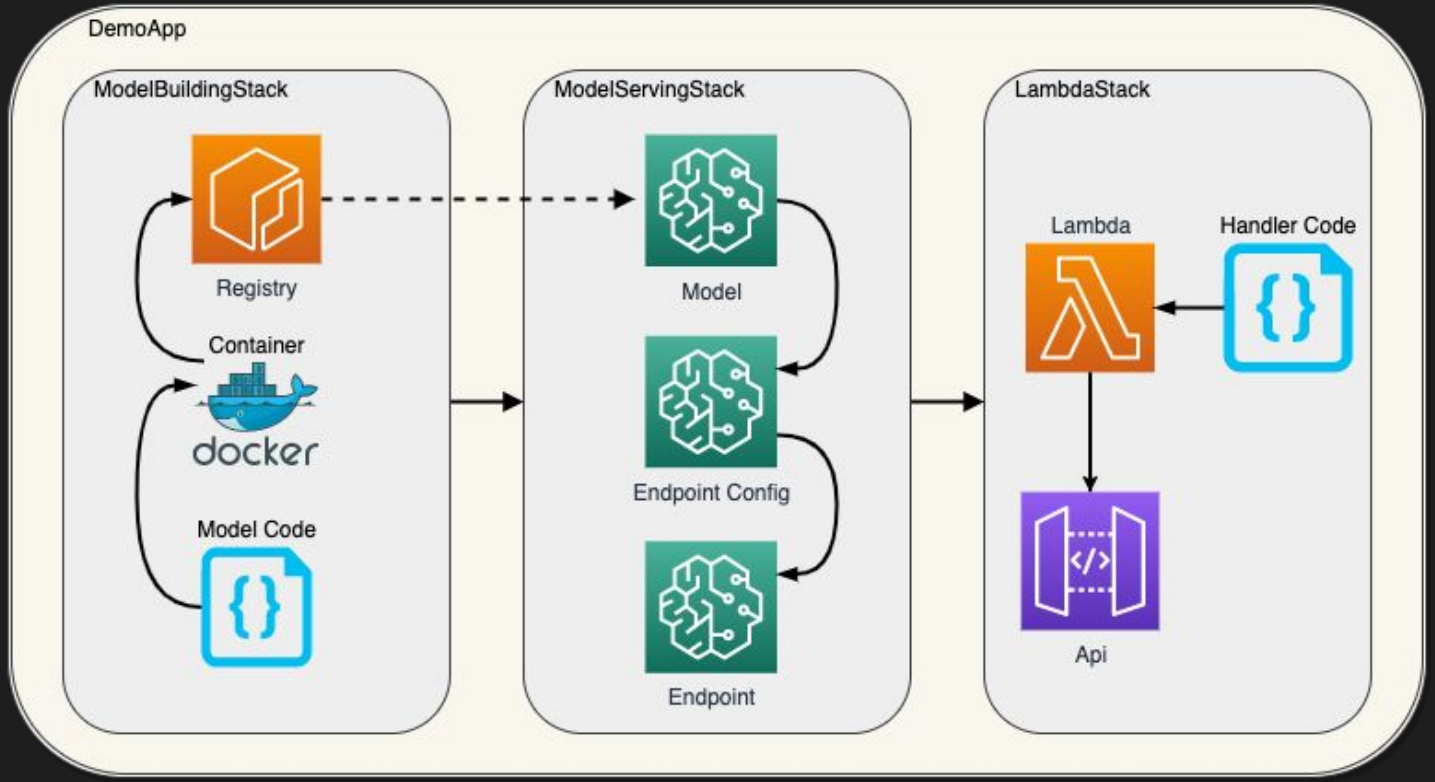
```
const getSagemakerPredictions = async (
    inputs: string[], endpointName: string,
    sagemakerClient: SageMakerRuntimeClient,
): Promise<number[][]> => {
    const sagemakerBodySource = { inputs: inputs };
    const params = {
        Body: new TextEncoder().encode(JSON.stringify(sagemakerBodySource)),
        EndpointName: endpointName,
    };
    const command = new InvokeEndpointCommand(params);
    const sagemakerData = await sagemakerClient.send(command);
    const sagemakerResponse =
            JSON.parse(new TextDecoder().decode(sagemakerData.Body));
    return sagemakerResponse["predictions"];
};
```

Lambda Handler

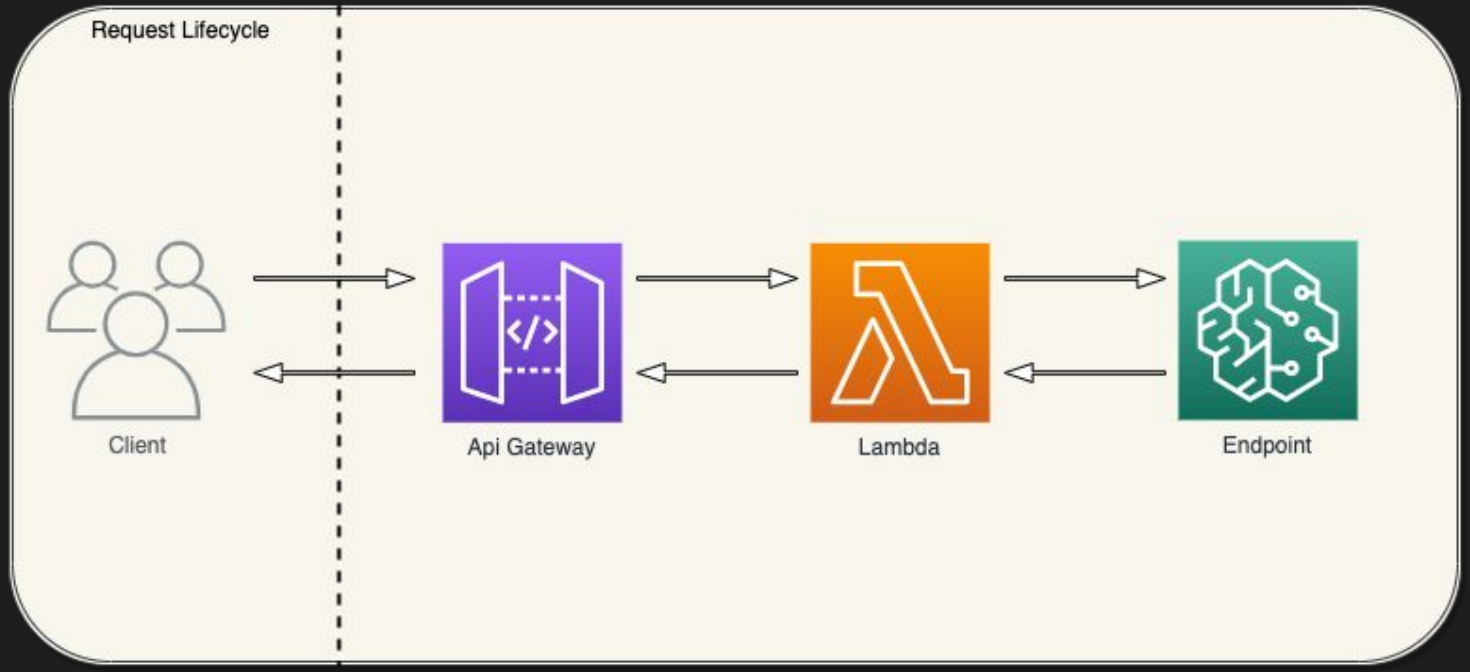
```
const sagemakerClient = new SageMakerRuntimeClient({ region: region });
const handler = async (
   event: APIGatewayProxyEventV2, _context: Context,
): Promise<APIGatewayProxyStructuredResultV2> => {
   const requestBody = JSON.parse(event.body);
   const inputs = requestBody.inputs;
   const predictions =
         await getSagemakerPredictions(inputs, endpointName, sagemakerClient);
   const response = {
       msg: "Request successful.",
       body: { predictions: predictions },
   };
   return jsonResponse(200, response);
};
```

Infrastructure

Stack Overview



Request Lifecycle



CDK App

```
const main = async () => {
    const app = new App();
    const { config: { env, ...sagemakerOptions } } = getStage(app);

    await prepareEcrRepository(modelDockerImage);

    new ModelBuildingStack(app, "DemoModelBuildingStack", {
        env, modelDockerImage });

    new ModelServingStack(app, "DemoModelServingStack", {
        env, endpointName, modelDockerImage, sagemakerOptions });

    new LambdaStack(app, "DemoLambdaStack", {
        env, endpointName });
};
```

CDK App

Helper to create ECR Repository

Stages

serverless.json

```
{
 "env": {
   "account": "111111111111",
   "region": "eu-west-1"
 },
 "modelName": "serverless",
 "endpointMemorySize": 1024,
 "endpointMaxConcurrency": 1,
 "endpointType": "serverless",
 "huggingFaceTokenizer": "distilbert-base-uncased-finetuned-sst-2-english",
 "huggingFaceModel": "distilbert-base-uncased-finetuned-sst-2-english"
}
```

serverless or *instance*

Can also be S3 location

```
make new_stage
make deploy
```

DEMO

Interested in the repo?

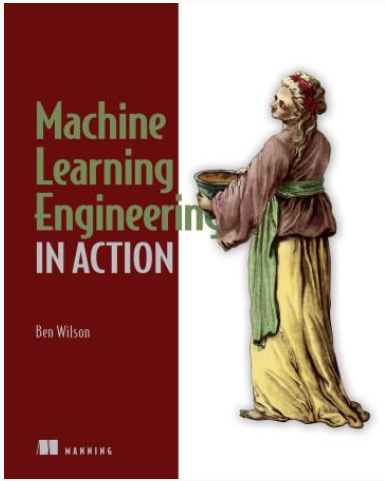
📧 rs@how.fm

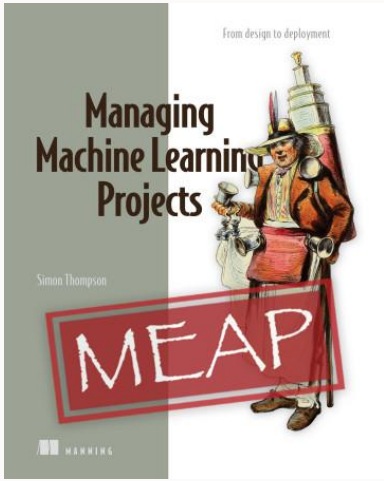
What remains...

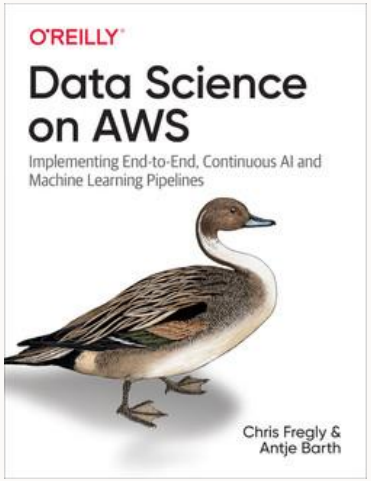
- Authorization
- Domain Routing
- Continuous Integration/Development
- Testing & Linting
- Caching
- GPU Tweaks
- OpenAPI Specifications
- ML Monitoring (we use Fiddler)

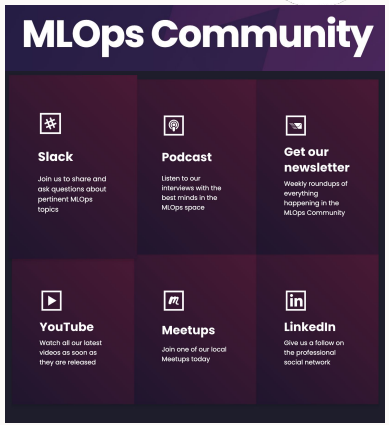
? Ask me about this!

Recommendations









We’re looking for Working Student(s)