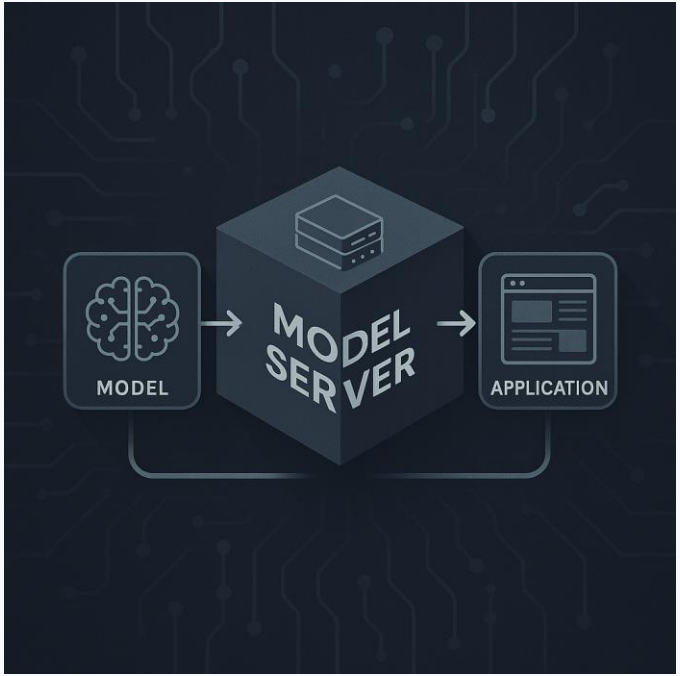
Model Server for Product Integration

Seamlessly deploy and monitor AI models in your products

July 31, 2025



TensorFlow Serving

Production-grade serving for TensorFlow models

High-performance serving system designed for production

- Flexible architecture: load, version & serve multiple models
- Easy algorithm experiments with stable APIs
- Integrates seamlessly with TensorFlow and extends to other models

TorchServe

Flexible serving for PyTorch models

Model Archive

Package models & code together for deployment

Multi model

Serve multiple models concurrently

Metrics

Built-in logging & Prometheus export

Customization

Custom handlers for preprocessing & postprocessing

[2]

REST APIs

Expose your model with a simple HTTP interface

```
from flask import Flask, request, jsonify
app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json['text']
    result = model_server.infer(data)
    return jsonify({ 'prediction': result })

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Why REST?

- Stateless communication simplifies scaling
- Widely adopted & language agnostic
- Integrates easily with single-page applications & microservices

[3]

Monitoring & Observability

Ensure reliability with metrics and dashboards

Prometheus

Open-source monitoring system that collects metrics via a pull model. Instrument your model server and export metrics for alerting and analysis.

Grafana

Leading open-source platform for visualizing and alerting on metrics. Create dashboards to track latency, throughput, error rates and uptime.

[4][5]

API Versioning

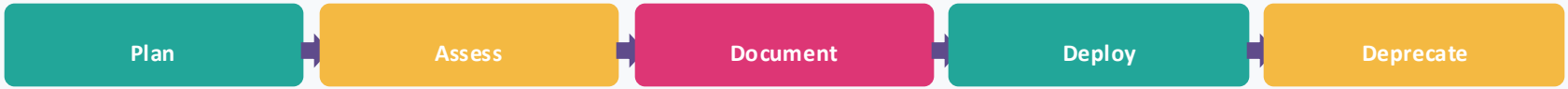
Maintain stability while evolving your APIs

Why version?

- Keeps producers and consumers in sync
- Minimizes impact of breaking changes
- Provides clear communication & trust

Strategies

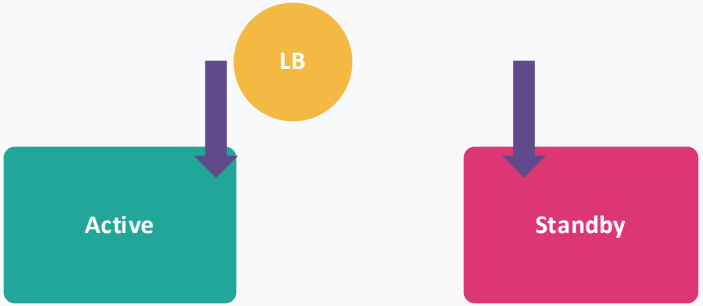
- URL versioning (/v1/resource)
- Query param (?version=v1)
- Header version (Accept-Version: v1)
- Consumer-based: sticky to first call



[6][7]

Reliability & Uptime

Design for continuity and resilience



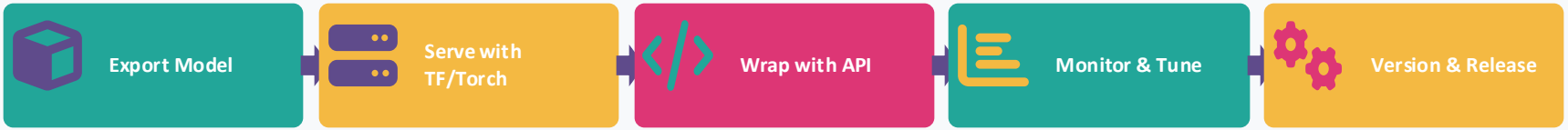
High availability = continuous operation

- Eliminate single points of failure with redundancy
- Use failover clusters and load balancing
- Implement automatic failure detection & recovery
- Protect against data loss with replication and backups

[8][9]

Let’s Build!

Deploy a sentiment analysis model server



- Train and export your model to a SavedModel or MAR file
- Deploy using TensorFlow Serving or TorchServe
- Build a REST API around it using Flask or FastAPI
- Instrument with Prometheus and visualise in Grafana
- Version and roll out updates gracefully

[10][11][12][13][14][15]