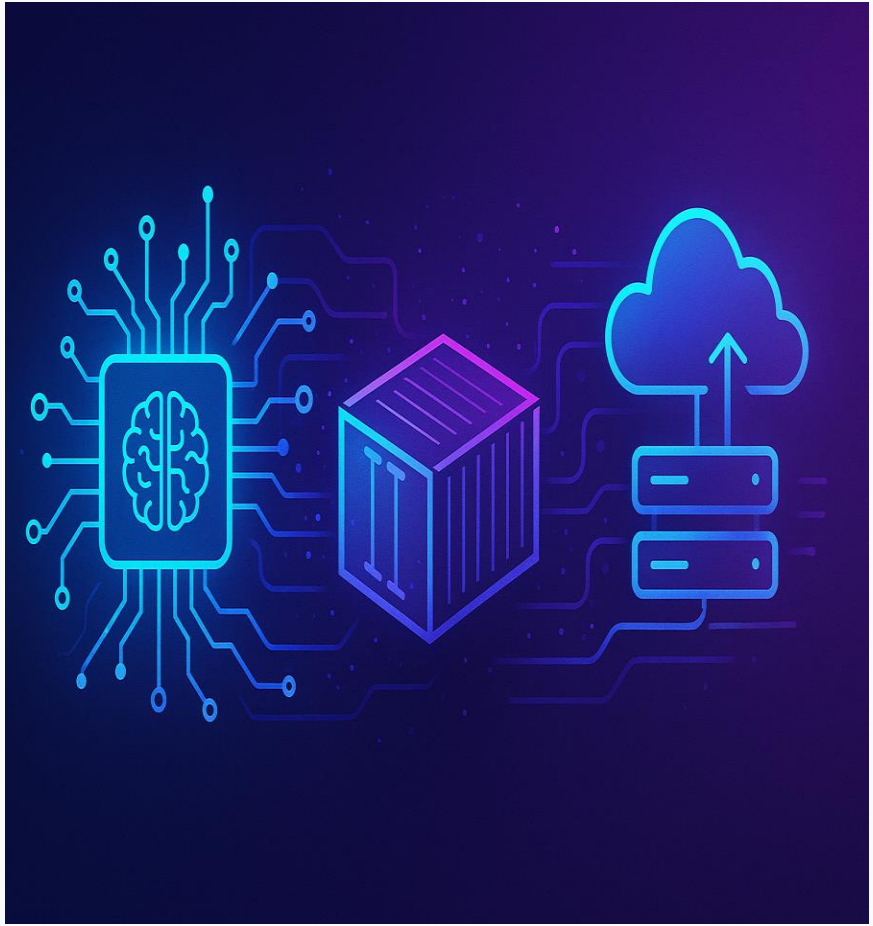
Serving Efficiency for Product Scalability

Optimize model deployment for scalable, high-performance products.
Master optimization, containerization, cloud strategies and cost considerations.

July 30, 2025



Model Optimization

Reduce model size and latency without sacrificing accuracy.

Pruning
Slim models by pruning low-impact connections.

Quantization
Convert 32-bit floats to 8-bit ints for smaller & faster models.

Distillation
Train a small model to mimic a larger one and retain accuracy.

Batch Inference
Run inference on multiple inputs together to maximise throughput.

API Development

Expose your optimized model through a modern web API.

- **High performance**
- Comparable to NodeJS & Go
- **Fast to code & fewer bugs**
- Increase dev speed by ~200–300%
- **Intuitive & easy to learn**
- Standards-based (OpenAPI & JSON Schema)

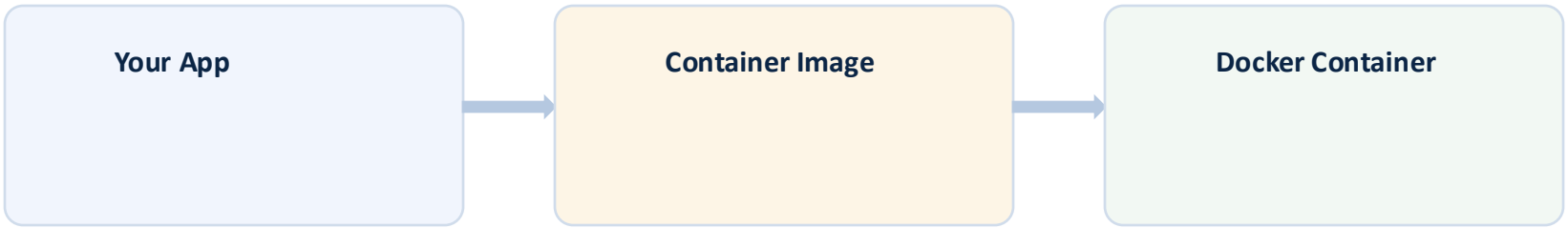
```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def read_root():
    return {"Hello": "World"}
```

Containerization with Docker

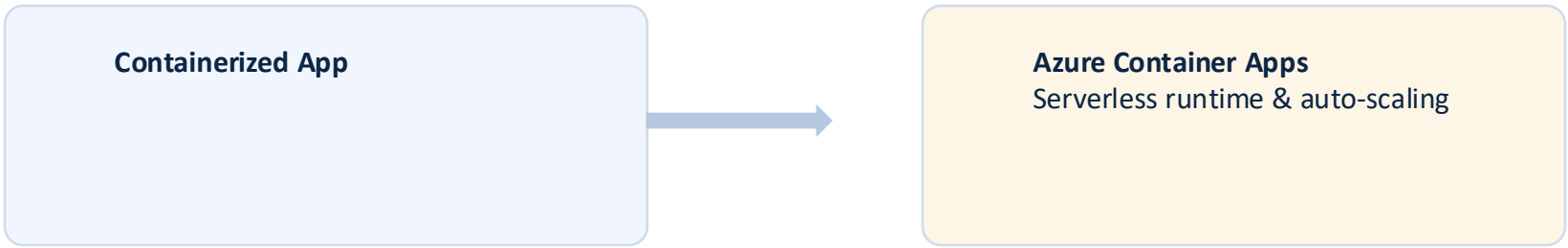
Package your model API into a portable unit that runs anywhere.



- Package code & dependencies into a single image
- Runs consistently across Linux & Windows
- Shares OS kernel → lightweight & cost-efficient
- Isolates software for uniform behaviour

Cloud Deployment on Azure

Deploy containerized models with automatic, serverless scaling.



- Serverless platform reduces operational complexity
- Autoscale on HTTP traffic, events, CPU/memory
- Scale to zero for idle workloads
- Ideal for APIs, background jobs & microservices

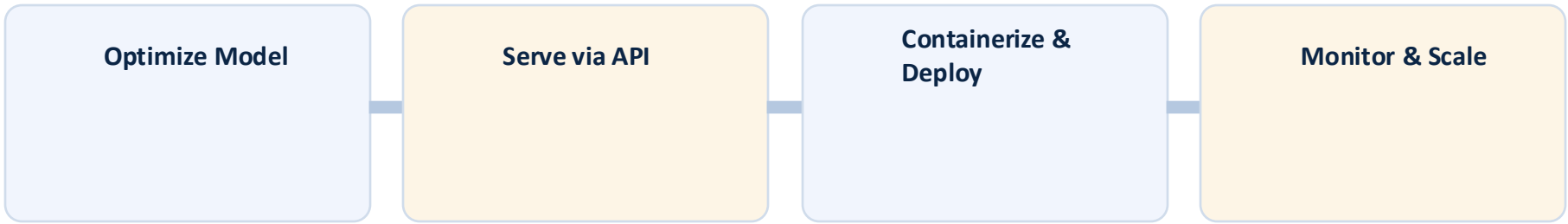
Cost-Benefit Analysis

Compare deployment strategies to choose the right fit for your needs.

Aspect	Serverless	Containers
Management	Fully managed; auto-scales	Requires orchestration & ops
Scalability	Auto-scales on demand	Manual or orchestrated
Billing Model	Pay per execution	Pay for allocated resources
Performance	Possible cold start	Consistent runtime
Control	Limited control	Full environment control
Best for	Stateless tasks	Long-running/stateful services

Planning for Scalability

Design your serving pipeline to handle growth and high user loads.



- Leverage autoscaling & load balancing
- Design for horizontal scaling & statelessness
- Implement caching & concurrency controls
- Monitor resource usage & tune thresholds

Let's Build!

Deploy an optimized model to production using ONNX Runtime, FastAPI, Docker and Azure.

- Quantize & prune your model, then export to ONNX
- Create a FastAPI endpoint for predictions
- Package the API into a Docker image
- Deploy to Azure Container Apps & configure autoscaling
- Monitor performance & adjust scaling thresholds