Here’s a full draft of documentation content designed for both Python and npm SDKs, formatted for Mintlify (using MDX/Markdown).

———

Introduction

---
title: "Introduction"
description: "Welcome to Agnitra AI — runtime optimization for AI models"
---
# Welcome to **Agnitra AI**

Agnitra AI is the agentic runtime optimisation engine for your AI models.
Whether you’re running large language models, diffusion, or other tensor-heavy workloads — Agnitra helps you:

- Run your model **faster** (higher tokens/sec, lower latency)
- Use **less memory** and compute
- Support **any hardware** (NVIDIA, AMD, Tenstorrent, etc.)
- Do it with **no code rewrites** — just integrate via SDK or CLI

> “One-line integration. Runtime patching. Cross-chip.”

---

## Why this matters
Modern AI models are massive and expensive to operate. Many users face:

- Manual tuning of kernels & performance bottlenecks
- Vendor-locked optimisation stacks
- Hard to scale or port across different accelerators
- Slow iteration cycles when trying to tune performance

Agnitra addresses these by embedding runtime telemetry, LLM–/RL-based optimisation, and kernel patching under the hood — letting you focus on your model, not the infrastructure.

---

## What this docs cover
This documentation covers:

- Installing and integrating via **Python SDK**, **npm/TS SDK**, or **CLI**
- Using our REST API if you choose the SaaS or hosted path
- Walk-throughs for use-cases (LLMs, diffusion, edge)
- Backend architecture overview
- Enterprise deployment & hardware targets
- FAQ and troubleshooting

Let’s get started.

———

Getting Started

~~~~markdown
---
title: "Getting Started"
sidebarTitle: "Getting Started"
description: "Install and run your first model optimisation in minutes."
---
## Quickstart

### Prerequisites
- Node.js version 16+ for the npm path
- Python 3.8+ for the Python SDK
- A model file (e.g., `llama.pt`, `model.onnx`) or telemetry log
- Access to your target hardware (GPU/accelerator)

### Install

#### Python
```bash
pip install agnitra

npm / TypeScript

npm install agnitra
# or
yarn add agnitra
~~~~

---

First optimisation

Python SDK

```python
from agnitra import optimize_model
model = load_model("llama.pt")  # load your model however you prefer
optimized = optimize_model(model, target="A100")
```

npm / TypeScript SDK

```typescript
import { optimizeModel } from "agnitra";
const model = await loadModel("llama.pt");
const optimized = await optimizeModel(model, { target: "A100" });
```

✅ That's it — after optimisation you can run optimized in place of your original model for inference.

---

Using the CLI

```bash
agnitra optimize --model llama.pt --target A100
```

This will profile, optimise, generate kernels, and patch at runtime.

---

Next steps

- Explore the SDK Reference for full API usage
- Check the CLI Reference for advanced options
- Review the Use-Cases section to see how others use Agnitra

---

# SDK Reference
~~~mdx
---
title: "SDK Reference"
sidebarTitle: "SDK Reference"
description: "Detailed API for Python and npm SDKs."
---
# Python SDK

## `optimize_model(model, *, target: str, mode?: str) -> model`
Optimise a loaded model for the given hardware target.

**Parameters**
- `model`: your loaded model (PyTorch, ONNX, etc.)
- `target`: e.g., `"A100"`, `"H100"`, `"MI250"`
- `mode` (optional): `"throughput"` | `"memory"` etc.

**Returns**: Optimised model instance.

## `benchmark_model(original_model, optimized_model) -> Dict[str, float]`
Compare before/after metrics: tokens/sec, latency, memory.

---

# npm / TypeScript SDK

## `optimizeModel(model: Model, options: { target: string; mode?: string }) : Promise<Model>`
Same functionality as Python, adapted for JS/TS.

```ts
import { optimizeModel } from "agnitra";

const optimized = await optimizeModel(myModel, { target: "A100", mode: "throughput" });

benchmarkModel(original: Model, optimized: Model) : Promise<BenchmarkResult>

Returns object such as { tokensPerSecGain: 0.28, latencyReduction: 0.21, memoryReduction: 0.18 }.

---

CLI Reference
~~~

(Insert detailed CLI flags, sub-commands, examples)

---

# Use Cases
~~~mdx
---
title: "Use Cases"
sidebarTitle: "Use-Cases"
description: "Real-world model optimisation examples"
---
## LLaMA-7B on A100
We profiled LLaMA-7B with baseline tokens/sec = 320. After optimisation we achieved ~410
(+28%) on A100, latency dropped ~20%, memory ~18%.

#### Python SDK example
```python
from agnitra import optimize_model
opt = optimize_model(model, target="A100", mode="throughput")

npm SDK example

import { optimizeModel } from "agnitra";
const opt = await optimizeModel(model, { target: "A100", mode: "throughput" });

Stable Diffusion XL (SDXL) on H100

…

---

# Architecture
```mdx
---
title: "Architecture"
sidebarTitle: "Architecture"
description: "How Agnitra works under the hood"
---
## Pipeline Overview
1. Telemetry Collector – captures GPU time, memory, tensor shapes
2. IR Graph Extraction – builds intermediate representation (e.g., torch.fx)
3. LLM + RL Agent – suggests kernel strategies; learns over time
4. Kernel Generator & Runtime Patcher – builds Triton/CUDA kernels, injects at runtime
5. Feedback Loop & Benchmarking – captures results, refines intelligence

## Hardware Support
List of supported chips, versioning details…

## Data Flow Diagram
(Insert architecture diagram here)
~~~

FAQ

---
title: "FAQ"
sidebarTitle: "FAQ"
description: "Common questions and answers"
---
### What kinds of models are supported?
We support PyTorch, ONNX, TensorFlow in most cases. Telemetry ingestion is flexible.

### What hardware is supported?
Any GPU/accelerator that has accessible profiling and kernel generation capabilities (NVIDIA-A100/H100, AMD-MI series, Tenstorrent etc.)

### Is the optimisation safe?
Yes — we generate kernels tested in sandbox; your original model remains unchanged until you choose to swap in the optimised version.

### What about enterprise/on-prem?
See our *Enterprise Deployment* section for VPC, private GPU clusters, compliance and security.

Enterprise Deployment

---
title: "Enterprise Deployment"
sidebarTitle: "Enterprise"
description: "On-prem, VPC, OEM licensing"
---
## Deployment Models
- **Hosted SaaS** — easy to adopt
- **On-Prem SDK** — installs in your environment
- **OEM Embedding** — integrate with hardware vendors

## Compliance & Security
Details on data privacy, telemetry anonymisation, and certs.

## Custom Domain & Domain-Path Setup
We support publishing docs at `docs.yourcompany.com` or `/docs` path.

## Support & SLA
Enterprise customers receive dedicated SLAs, support, custom model tuning, annual training etc.

CLI Reference

---
title: "CLI Reference"
sidebarTitle: "CLI"
description: "Command-line usage and flags"
---
## `agnitra optimize`

agnitra optimize –model  –target  [–mode throughput|memory] [–telemetry ]

### Flags
- `--model` : path to your model file
- `--target` : hardware target (e.g., A100)
- `--mode` : optimisation mode
- `--telemetry` : (optional) supply pre-collected telemetry JSON

### Examples

```bash
agnitra optimize --model llama.pt --target A100
agnitra optimize --model sdxl.onnx --target H100 --mode memory
```

---

agnitra benchmark

agnitra benchmark --before model.pt --after optimized_model.pt

Generates a report, e.g., tokens/sec, latency, memory.

---

### Configuration File — `docs.json`
This file configures your Mintlify docs site layout, themes, nav structure.

```json
{
  "name": "Agnitra AI Docs",
  "logo": {
    "light": "/logo/light.svg",
    "dark": "/logo/dark.svg"
  },
  "colors": {
    "primary": "#00C4A7",
    "background": "#111827"
  },
  "sidebar": [
    { "title": "Getting Started", "items": ["introduction", "getting-started"] },
    { "title": "SDK Reference", "items": ["sdk-reference"] },
    { "title": "CLI Reference", "items": ["cli-reference"] },
    { "title": "Use-Cases", "items": ["use-cases"] },
    { "title": "Architecture", "items": ["architecture"] },
    { "title": "Enterprise", "items": ["enterprise-deployment"] },
    { "title": "FAQ", "items": ["faq"] }
  ],
  "editUrl": "https://github.com/yourorg/docs/edit/main/docs/",
  "favicon": "/favicon.ico"
}
```

---