GSDC
Global Skill Development Council

Globally recognised certification body
190+ countries · 2,50,000+ certified professionals

IAS
ACCREDITED
Personnel Certification Body

BEST AI TOOLS FOR

Forward Deployed Engineers

www.gsdcouncil.org

Understanding AI Tools in Forward Deployed Engineering

What Does a Forward Deployed Engineer Do?

A Forward Deployed Engineer (FDE) sits at the intersection of engineering and customer success. Unlike engineers who build products in isolation, FDEs work directly inside customer environments – discovering problems, designing solutions, and deploying working AI applications that solve real business challenges. The role demands breadth: you may spend Monday mapping a customer's data architecture, Tuesday building a prototype retrieval system, and Wednesday debugging a failed API integration in a production environment.

FDEs must move fast without cutting corners. They are expected to understand both the technical landscape and the customer's business context well enough to make confident recommendations. AI tools are the leverage that makes this possible at the pace the job demands.

Why FDEs Need AI Tools

AI tools amplify an FDE's ability to work across disciplines. They compress the time required to understand unfamiliar codebases, generate working prototypes, test integrations, and evaluate AI output quality. Without them, the cognitive load of context-switching between languages, frameworks, platforms, and customer requirements would be overwhelming. With them, a single FDE can deliver the output of a small team.

Critically, the right AI tool for a given situation depends on the customer's environment, not on popularity or trending GitHub stars. A tool that is ideal in one deployment may be a liability in another. This handbook is designed to help you build judgement, not just a list of tools to install.

How AI Tools Support the FDE Workflow

01

Customer Discovery

Use AI to analyse transcripts, map data flows, and surface requirements.

02

Prototyping

Build working demos with coding assistants and rapid UI tools.

03

Integration

Connect AI applications to CRMs, databases, and internal APIs.

04

Deployment

Package and ship solutions using cloud platforms and containers.

05

Troubleshooting

Use observability and evaluation tools to find and fix problems in production.

Choose tools based on customer requirements, existing stack, data privacy needs, and integration constraints – not on what is trending.

Best AI Coding Assistants

AI coding assistants are the most immediately productive category of tools available to an FDE. They support code generation, explanation, debugging, refactoring, test creation, and navigating unfamiliar codebases – all tasks that appear daily in the FDE role.

Cursor

An AI-native code editor built on VS Code. Cursor embeds a chat interface directly into the editor and can reference your entire codebase when generating suggestions. **Best for:** FDEs who want deep, context-aware code generation within a familiar IDE. Useful when working inside large, unfamiliar customer codebases.

GitHub Copilot

Microsoft's inline code suggestion tool, integrated into VS Code, JetBrains, and other popular editors. Copilot excels at completing boilerplate, suggesting function implementations, and writing tests. **Best for:** Day-to-day code acceleration within established editor workflows. Strong enterprise adoption and SSO support.

Claude

Anthropic's large context model – capable of processing entire files, documentation sets, or codebases in a single prompt. Claude is particularly strong for code explanation, refactoring legacy systems, and producing well-structured technical documentation. **Best for:** Situations requiring careful reasoning over large amounts of code or text.

ChatGPT

OpenAI's conversational model, available via web interface and API. GPT-4 and later models handle code generation across virtually every language, debugging by pasting error traces, and rapid prototyping of logic. **Best for:** Exploratory coding tasks, quick debugging sessions, and when you need to explain code to non-technical stakeholders.

Windsurf

An agentic coding environment by Codeium that can autonomously plan and execute multi-step coding tasks – creating files, running commands, and iterating. **Best for:** FDEs who want to delegate larger coding tasks rather than just receiving inline completions. Particularly effective for scaffolding new projects quickly.

Best Tools for Building AI Applications

Building an AI-powered application requires more than a language model. You need to connect models to data, wrap them in reliable APIs, and integrate AI capabilities into existing business systems. The tools in this section fall into two distinct categories: **model providers** (who supply the AI models via API) and **application development frameworks** (which help you build around those models).

Model Providers — APIs That Supply AI Capabilities

OpenAI API

The most widely adopted LLM API. Provides access to GPT-4o and other models via a simple REST interface. Extensive documentation, broad library support, and a large community. Free tier available; paid plans scale with token usage.

Anthropic API

Provides access to Claude models. Known for large context windows, instruction-following quality, and safety-oriented design. A strong alternative to OpenAI, particularly for enterprise customers with compliance requirements.

Google Gemini API

Google's multimodal model API, capable of processing text, images, audio, and code in a single prompt. Well-suited when customers are already invested in the Google Cloud ecosystem.

Hugging Face

A model hub and inference platform hosting thousands of open-source models. Enables self-hosted deployments for customers with strict data privacy requirements who cannot send data to a third-party API.

Application Frameworks — How You Build Around Models

LangChain

A Python and JavaScript framework for chaining LLM calls together with tools, memory, and data retrievers. LangChain provides abstractions for building pipelines that combine multiple AI components. Best suited for complex, multi-step workflows requiring flexibility.

LlamaIndex

Specialises in connecting LLMs to structured and unstructured enterprise data – documents, databases, APIs. LlamaIndex provides indexing, retrieval, and query engines that make it easier to build knowledge-grounded AI applications. Its data connectors are a particular strength.

LangChain and LlamaIndex overlap significantly. LangChain favours flexible chaining; LlamaIndex favours data-first retrieval applications. Many production applications combine both.

Best Tools for Building AI Agents

An AI agent is a system that uses a language model to make decisions, call tools, and complete multi-step tasks autonomously. Agents go beyond simple API calls: they can search the web, query databases, write and execute code, send emails, or call other services – with or without a human in the loop. Before choosing an agent framework, ask whether a simple API workflow would be sufficient. Agents add complexity; only use them when the task genuinely requires multi-step reasoning or dynamic tool selection.

LangGraph

A graph-based agent orchestration library built on LangChain. Models agent behaviour as a directed graph of nodes and edges, making complex workflows with branching logic, loops, and human approval steps easy to define and debug. Best for production-grade agents requiring controlled execution.

CrewAI

A framework for orchestrating multiple AI agents that collaborate to complete a goal. Each "crew member" has a defined role, goal, and set of tools. Best for scenarios requiring role-based task decomposition – for example, a researcher agent, a writer agent, and a reviewer agent working together.

Microsoft Semantic Kernel

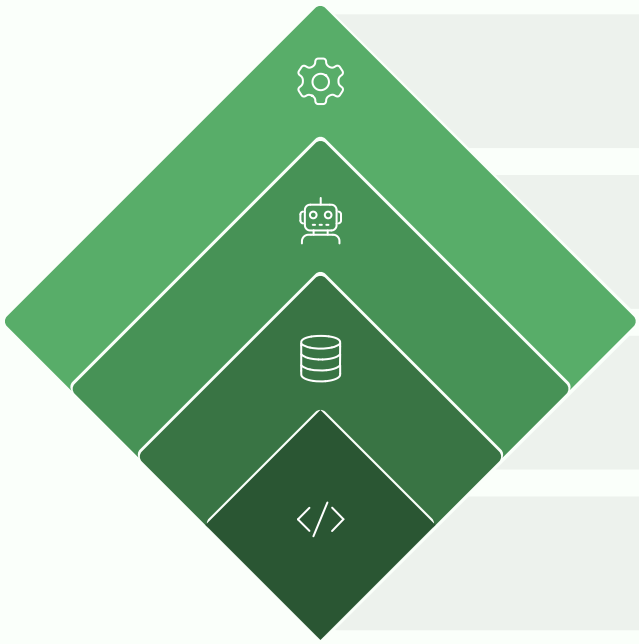
An enterprise-oriented SDK for integrating LLMs into .NET, Python, and Java applications. Semantic Kernel supports plugin-based tool calling and is designed for organisations already using Microsoft Azure and enterprise identity systems.

Model Context Protocol (MCP)

An open protocol, proposed by Anthropic, that standardises how AI models connect to tools and data sources. MCP defines a consistent interface so that any compliant model can call any compliant tool – reducing integration fragmentation across agent systems.

n8n

A low-code workflow automation platform with AI nodes built in. n8n allows FDEs to build agent-like workflows without writing all logic from scratch. Strong for integrating AI steps with CRMs, databases, and third-party APIs in a visual interface. Self-hostable for privacy-sensitive deployments.



Agent Framework

Autonomous reasoning and tools

Choose Agent

Multi-step or dynamic tool selection

APIs for Queries

Fetch data or transform inputs

Use Simple Call

Single-step or deterministic task

Best Tools for RAG and Enterprise Knowledge

Retrieval-Augmented Generation (RAG) is the technique of grounding an LLM's responses in specific documents or data sources, rather than relying solely on the model's training data. RAG is one of the most practically important patterns in enterprise AI deployment because it allows AI assistants to answer questions accurately using company-specific information – without retraining the model.

A RAG pipeline typically involves: ingesting documents, splitting them into chunks, embedding each chunk as a vector, storing those vectors in a vector database, retrieving the most relevant chunks at query time, and passing them to the LLM as context. Each stage has specific tool options.

Tool	Type	Best For	Key Strength
LlamaIndex	Framework	End-to-end RAG pipelines with connectors to 150+ data sources	Data ingestion and query engines
LangChain	Framework	Flexible retrieval chains combined with other AI components	Composability and tool ecosystem
Pinecone	Vector DB (managed)	Production vector search at scale, fully managed	Speed and managed infrastructure
Weaviate	Vector DB (managed / self-hosted)	Hybrid search combining vector and keyword retrieval	Flexible deployment, GraphQL API
Chroma	Vector DB (local / embedded)	Local development and prototyping	Zero-config, Python-native, fast to start
pgvector	PostgreSQL extension	Customers who already use PostgreSQL	No new infrastructure; familiar SQL interface
Elasticsearch / OpenSearch	Search platform	Hybrid search in existing enterprise search deployments	Mature BM25 + vector hybrid search

ⓘ For a beginner RAG prototype, start with Chroma and LlamaIndex. For production with existing PostgreSQL infrastructure, pgvector is often the path of least resistance – avoiding new infrastructure entirely.

Best Tools for API Integration and Testing

FDEs spend a significant portion of their time connecting things together – AI models to business systems, internal APIs to external services, new prototypes to existing data sources. Robust API integration and testing skills, backed by the right tools, are what separate FDEs who can deliver working integrations from those who can only demonstrate them in isolation.

Postman

The industry-standard tool for API exploration and testing. FDEs use Postman to inspect API schemas, test endpoints manually, debug authentication flows, and build automated test collections. A Postman collection is also an excellent artefact to hand over to a customer's engineering team.

Swagger / OpenAPI

The OpenAPI Specification is the standard format for describing REST APIs. Swagger provides tooling to read, generate, and visualise OpenAPI schemas. When a customer hands you an API spec, Swagger UI gives you an interactive exploration interface instantly – no code required.

FastAPI

A Python web framework for building REST APIs rapidly. FDEs use FastAPI to wrap AI models and LangChain pipelines behind a standard HTTP interface – making them accessible to any frontend, integration tool, or customer system that can send an HTTP request.

Zapier

A no-code automation platform connecting thousands of apps via pre-built triggers and actions. Useful for rapid integration demos that show a customer what an automated workflow could look like – before investing in a custom integration. Best for non-technical stakeholders and low-code proofs of concept.

n8n

A self-hostable, low-code workflow automation tool with built-in AI nodes and over 400 integrations. More flexible than Zapier and suitable for production workflows. FDEs can build AI-augmented integrations – for example, classifying incoming form submissions and routing them to different systems – without writing a complete application.

Practical Integration Debugging

When an API integration fails, a methodical approach matters: check authentication headers first, then inspect the raw request and response bodies in Postman, verify the endpoint URL and method, review rate limits, and confirm the payload schema matches the API documentation. Log everything at each stage.

Best Tools for Data Preparation and Analysis

AI applications are only as good as the data that feeds them. Before building any AI solution, an FDE must inspect customer data, understand its structure and quality, clean it where necessary, and transform it into a format suitable for AI consumption. Data issues – missing values, inconsistent formats, incorrect types, duplicate records – are among the most common causes of poor AI performance in production.

1 Python and Jupyter Notebooks

The foundational environment for data work in AI engineering. Jupyter Notebooks allow FDEs to write Python code in interactive cells, see results immediately, and share reproducible analysis with customers or colleagues. Python's ecosystem – NumPy, Pandas, Matplotlib, scikit-learn – covers virtually every data task an FDE will encounter.

2 Pandas

The standard Python library for tabular data manipulation. FDEs use Pandas to load CSV files, inspect column types, find missing values, filter rows, join datasets, and reshape data for AI pipelines. It is typically the first tool applied when examining a new customer dataset. For very large datasets, Polars is a faster modern alternative worth exploring.

3 SQL and DuckDB

SQL remains the most widely understood language for querying structured data. DuckDB is a fast in-process analytical database that allows FDEs to run SQL queries directly on CSV, Parquet, and JSON files – without setting up a server. It bridges the gap between Pandas and a full data warehouse for rapid customer data exploration.

4 dbt (data build tool)

A transformation framework that allows analysts and engineers to write data transformations as SQL models with version control, testing, and documentation built in. When a customer's data pipeline is already managed by dbt, understanding its structure is essential for any AI integration that depends on transformed data.

Best Tools for Rapid Prototyping

One of the most valuable skills an FDE can demonstrate is the ability to convert a customer requirement into a working, testable prototype within hours or days – not weeks. Rapid prototyping tools eliminate the need to build frontend infrastructure from scratch, allowing you to focus on the AI logic and the user experience that matters to the customer.

Streamlit

A Python library for building interactive web applications with minimal code. Write Python, get a web app. FDEs use Streamlit to wrap LangChain chains or LlamaIndex pipelines in a simple chat or query interface. Deployed in minutes to Streamlit Community Cloud or any cloud platform.

Gradio

Similar to Streamlit but optimised for machine learning interfaces – file uploads, image inputs, audio, model comparison. Best for demos involving non-text inputs or side-by-side model comparison. Directly integrated with Hugging Face Spaces for one-click hosting.

Replit

A cloud-based development environment requiring no local setup. FDEs can spin up a Python or Node.js project, install dependencies, and share a live URL with a customer in minutes. Ideal when working on a customer's machine or in a locked-down environment.

Vercel

A frontend deployment platform that makes deploying Next.js and React applications instant. Combined with Vercel's AI SDK, it is a strong choice for FDEs building customer-facing AI interfaces that need to look polished and load quickly.

Step-by-Step: Building a Simple AI PoC with Streamlit

01

Define the scope

Agree with the customer on one specific question the prototype must answer – for example, "query our product FAQ using natural language."

02

Prepare data

Load the customer's documents into a LlamaIndex index backed by Chroma for local vector storage.

03

Build the query engine

Wire a LlamaIndex query engine to the OpenAI API with a system prompt that constrains the assistant to answer from retrieved context only.

04

Build the interface

Wrap the query engine in a Streamlit chat interface. Ten lines of Python. Run it with `streamlit run app.py`.

05

Demo and iterate

Share the URL with the customer. Collect feedback. Adjust the prompt, chunk size, or retrieval strategy based on what is and is not working.

Best Tools for Cloud, Deployment, and DevOps

Turning a working prototype into a solution a customer can actually use requires deployment – packaging the application, choosing a hosting environment, connecting services securely, and ensuring the system can be updated and monitored. FDEs are not typically responsible for managing production infrastructure long-term, but they must be able to deploy prototypes convincingly and hand over deployable artefacts to a customer's engineering team.

1

AWS / Azure / GCP

The three major cloud providers. AWS is the most widely adopted globally. Azure dominates in Microsoft-heavy enterprise environments. Google Cloud is preferred for teams using Vertex AI and Gemini. Match the cloud to the customer's existing footprint.

2

Docker

Containerises your application and all its dependencies into a portable image. A Docker container runs identically on your laptop, a CI server, and a cloud VM – eliminating "it works on my machine" problems. Essential for any deployment handover.

3

GitHub Actions

A CI/CD platform built into GitHub. Automates testing, building, and deploying your application on every code push. FDEs use it to create repeatable deployment pipelines that a customer's team can maintain independently after handover.

4

Vercel / Serverless

For frontend-heavy AI applications, Vercel provides zero-config deployment with automatic scaling. Serverless functions on AWS Lambda or Google Cloud Functions are useful for lightweight AI API endpoints that do not need persistent compute.

When deploying AI applications, always separate your API keys from your application code. Use environment variables and a secrets manager (AWS Secrets Manager, Azure Key Vault, or HashiCorp Vault) – never commit credentials to a repository.

Best Tools for AI Testing and Evaluation

AI applications behave differently from deterministic software. The same input can produce different outputs across runs, model versions, or prompt changes. Systematic evaluation is therefore essential before deploying an AI application into a customer's environment. Testing must cover not just functional correctness but also output quality, retrieval accuracy, consistency, and resistance to edge cases.

Evaluation Tools Compared

LangSmith

Built by LangChain, LangSmith provides tracing, evaluation, and dataset management for LLM applications. It records every prompt and response in a chain, making it possible to replay inputs, score outputs, and compare performance across model or prompt changes. Free tier available.

Ragas

An open-source framework specifically designed for evaluating RAG pipelines. Ragas measures faithfulness (does the answer reflect the retrieved context?), answer relevancy, and context recall – the three dimensions most critical to RAG quality assessment.

DeepEval

A comprehensive LLM evaluation framework with built-in metrics for hallucination, bias, toxicity, and task-specific performance. Integrates with pytest, making AI evaluation feel like standard unit testing. Good for teams that want CI/CD-integrated quality gates.

Promptfoo

A command-line tool for comparing prompts and models systematically. Define a set of test cases and expected outputs, run them across multiple prompts or model versions, and see a comparison matrix. Ideal for prompt optimisation during development.

Evaluating an AI Customer Support Assistant

Before deploying an AI customer support assistant, an FDE should construct an evaluation dataset from real historical support tickets – covering common queries, edge cases, and known difficult scenarios. For each test case, define what a good response looks like.

Run the full dataset through the assistant and measure: **answer correctness** (does it resolve the query?), **faithfulness** (does it only use information from the retrieved knowledge base?), **response consistency** (does the same question always get an equivalent answer?), and **escalation accuracy** (does it correctly identify when to hand off to a human?).

Use Ragas to score retrieval quality, LangSmith to trace failures back to the specific retrieval step or prompt that caused them, and Promptfoo to test whether an improved prompt degrades performance on other test cases. Document the evaluation results as a formal quality report – this is the artefact that justifies deploying the assistant to real users.

> Never deploy an AI application to end users without a documented evaluation baseline. If you cannot measure it, you cannot defend it.

Best Tools for Monitoring and Debugging

Deploying an AI application is not the end of an FDE's responsibility – it is the beginning of a new phase. Production AI systems experience failed requests, slow responses, unexpected model output, rising token costs, and regression when models are updated. Effective monitoring and debugging tools give you visibility into what is actually happening inside your application.

LangSmith

Provides full trace visibility for LangChain applications – every prompt, retrieval step, and model call is logged with latency and token cost. When a user reports an incorrect response, you can find the exact trace and see what was retrieved and what was sent to the model.

Langfuse

An open-source observability platform for LLM applications. A strong alternative to LangSmith, particularly for teams who want to self-host their observability stack for data privacy reasons. Supports multi-step tracing, evaluation scoring, and user feedback collection.

OpenTelemetry

The open standard for distributed tracing, metrics, and logging. Many AI frameworks now emit OpenTelemetry traces, allowing AI application spans to be correlated with the rest of the application stack in tools like Grafana, Jaeger, or Datadog.

Sentry & Grafana

Sentry captures application errors and exceptions with full stack traces. Grafana visualises time-series metrics – request volume, latency percentiles, error rates, and token usage – in dashboards that give an operational view of AI system health over time.

Troubleshooting Quick Reference

Problem	Likely Cause	Tool to Use
Incorrect or hallucinated answer	Retrieval returning irrelevant chunks; weak prompt grounding	LangSmith trace, Ragas faithfulness score
Slow response times	Large context windows; inefficient retrieval; no caching	LangSmith latency trace, Grafana metrics
Unexpected exceptions	API rate limits; malformed requests; schema changes	Sentry, application logs, Postman
Rising API costs	Token usage growing; unnecessary retrieval; no caching	LangSmith token tracking, Grafana cost dashboards
Model output regression	Model version update; prompt drift	Promptfoo regression tests, DeepEval

Best Tools for AI Security

AI applications introduce a new category of security risks that traditional application security does not fully address. Prompt injection – where a user manipulates the model's instructions through their input – is a critical concern. Sensitive data exposure, insecure tool permissions, and inadequate API credential management are equally important issues that FDEs must understand and address before handing over any AI solution to a customer.

Security Categories in AI Applications

OWASP LLM Top 10

The OWASP guidance for LLM application security defines the most critical risk categories – prompt injection, insecure output handling, training data poisoning, model denial of service, and others. Every FDE should be familiar with this framework before deploying any AI application to a customer environment.

Microsoft Presidio

An open-source data protection and anonymisation SDK. Presidio identifies and redacts PII (personally identifiable information) – names, email addresses, phone numbers, credit card numbers – from text before it is sent to an AI model. Essential when customer data contains sensitive personal information.

Garak

An LLM vulnerability scanner that probes models for prompt injection susceptibility, jailbreaks, data leakage, and toxic output generation. Use Garak to systematically test an AI application's resilience before it is exposed to end users.

Infrastructure and Credential Security

Beyond model-level vulnerabilities, AI applications require standard security hygiene applied consistently. API keys for OpenAI, Anthropic, or other model providers should never appear in code repositories. Use environment variables in development and a dedicated secrets manager – AWS Secrets Manager, Azure Key Vault, or HashiCorp Vault – in production. Rotate credentials regularly.

AI agents that call tools – querying databases, sending emails, writing files – should operate under the principle of least privilege. Grant each tool call only the permissions it strictly requires. An agent that can read a customer database should not also have write permissions unless explicitly necessary.

Promptfoo, beyond its role in prompt comparison, includes adversarial testing capabilities that simulate prompt injection attacks against your AI application. Use it alongside Garak to ensure your safety guardrails hold against realistic attack patterns.

Security Testing Tools

Garak, Promptfoo – probe for vulnerabilities, injection, and jailbreaks

Privacy Tools

Microsoft Presidio – detect and redact PII before it reaches the model

Security Frameworks

OWASP LLM Top 10 – risk taxonomy and mitigation guidance

Infrastructure Controls

Secrets managers, IAM policies, least privilege tool permissions

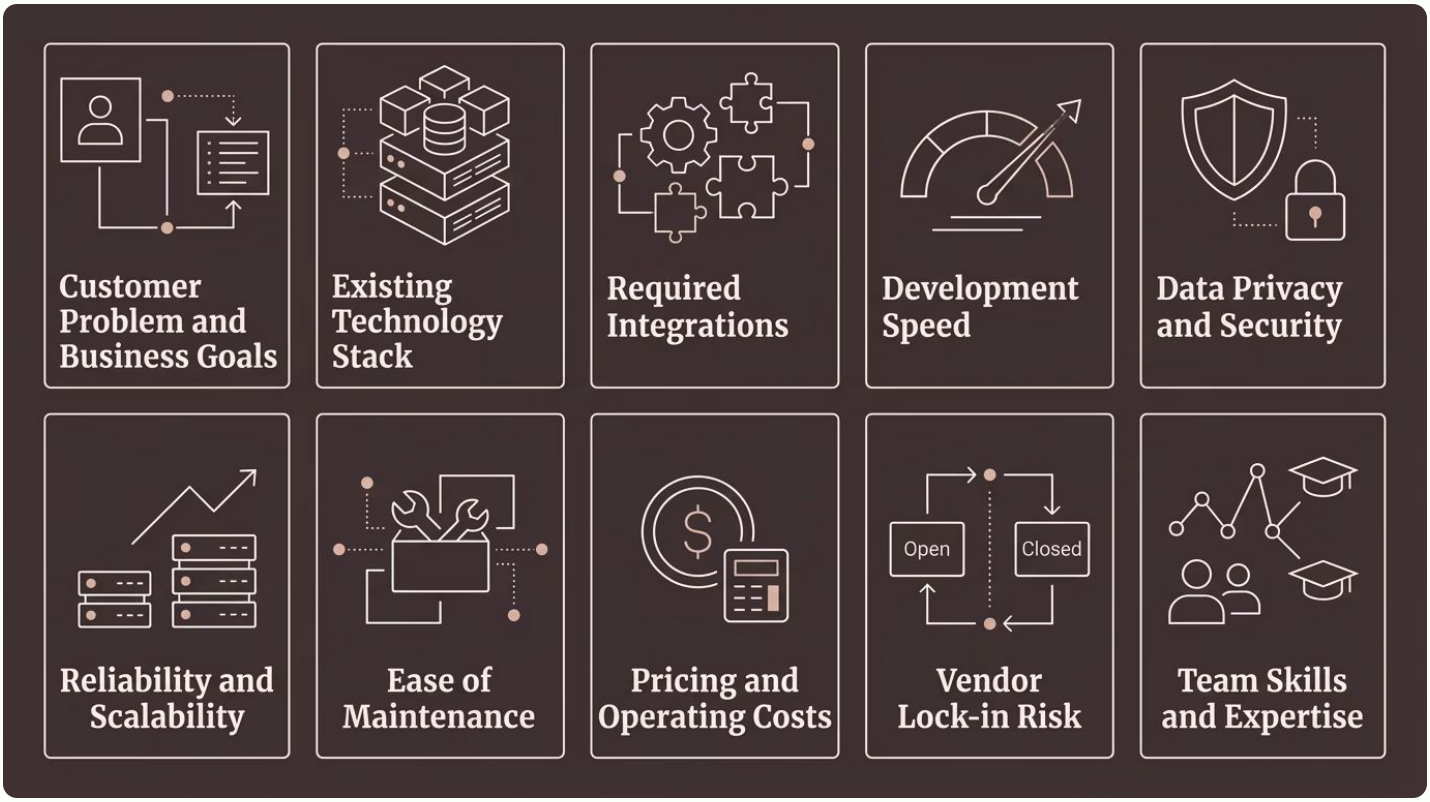
AI Tools Comparison Matrix

The table below groups the most important tools by category. Use it as a quick reference – not as a ranking. No tool is universally best. The right choice always depends on the customer's environment, existing stack, data requirements, security constraints, and budget. Free tiers, paid plans, and feature sets change; verify current pricing and capabilities with official documentation before recommending a tool to a customer.

Tool	Category	Main Purpose	Strengths	Limitations	Learning Priority
Cursor	Coding Assistant	AI-native code editor	Codebase context, inline chat	Subscription cost	High
GitHub Copilot	Coding Assistant	Inline code suggestions	IDE integration, enterprise SSO	Less context-aware than Cursor	High
OpenAI API	Model Provider	LLM via REST API	Ecosystem, documentation, adoption	Cost at scale; data privacy	High
LangChain	App Framework	LLM pipeline orchestration	Ecosystem, flexibility	Complexity; evolving API surface	High
LlamaIndex	App Framework	RAG and data retrieval	Data connectors, query engines	Less flexible outside RAG	High
LangGraph	Agent Framework	Graph-based agent orchestration	Controlled execution, branching	Steeper learning curve	Medium
n8n	Automation / Agent	Visual workflow automation	Self-hostable, 400+ integrations	Less flexible for complex logic	Medium
Pinecone	Vector DB	Managed vector search	Speed, scalability, managed	Vendor lock-in, cost	Medium
Chroma	Vector DB	Local vector storage	Zero-config, fast prototyping	Not production-grade at scale	High (for prototyping)
Streamlit	Prototyping	Python web app builder	Speed, simplicity, Python-native	Limited UI customisation	High
LangSmith	Evaluation / Monitoring	LLM tracing and evaluation	Deep LangChain integration	Tied to LangChain ecosystem	High
Ragas	Evaluation	RAG pipeline evaluation	Purpose-built RAG metrics	Narrow scope	Medium
Postman	API Testing	API exploration and testing	Industry standard, rich UI	Limited for automated CI testing	High
Docker	Deployment	Application containerisation	Portability, reproducibility	Operational complexity	High
Garak	AI Security	LLM vulnerability scanner	Systematic injection testing	Specialist, narrow scope	Medium

How to Choose the Right AI Tools

Selecting AI tools for a customer engagement is a decision that should be driven by the customer's problem and environment – not by what is newest or most discussed in the AI community. The following framework provides a structured way to evaluate options. Apply it before every engagement to ensure your recommendations are defensible and grounded in the customer's actual situation.



Decision Tree: Narrowing Your Options

01

Start with the problem

Is this a retrieval problem, a generation problem, an automation problem, or an integration problem? The answer narrows the category of tools before you consider specific products.

02

Check existing stack

What cloud, databases, identity systems, and programming languages does the customer already use? Favour tools that integrate naturally. Avoid introducing unnecessary new infrastructure.

03

Evaluate data privacy constraints

Can data be sent to a third-party API, or does it need to stay on-premises or within a specific cloud region? This constraint immediately rules out or requires certain model providers.

04

Consider development speed vs. maintainability

Managed services are faster to start with but create vendor dependencies. Self-hosted tools require more setup but give the customer full control. Discuss this trade-off explicitly.

05

Confirm team skills

The best tool is one the customer's team can operate and maintain after you leave. Prioritise tools that align with existing skills, and identify what training might be needed.

Key Questions to Ask Before Recommending a Tool

Cost and Licensing

What are the realistic costs at the customer's expected usage volume? Are there free tier limits that would be breached immediately in production? Is the pricing model predictable?

Vendor Lock-in

How difficult would it be to switch to a different tool if the vendor changes pricing, discontinues the product, or fails to meet requirements? Open-source tools with standard APIs reduce this risk.

Security and Compliance

Does the tool have the certifications the customer's industry requires – SOC 2, ISO 27001, HIPAA, GDPR compliance? Where is data processed and stored?

Community and Support

Is there an active community, good documentation, and accessible support? Tools with small communities and poor documentation create risk for the customer's team long after the engagement ends.

Recommended AI Tool Stack for an FDE

The following three stacks are illustrative examples – not mandatory configurations. The right combination for any customer engagement depends on the factors described in the previous section. Use these as starting points and reference points for conversations, not as prescriptions.

🟢 Beginner Stack

Coding: GitHub Copilot or Cursor
LLM API: OpenAI API
Prototyping: Streamlit
API Testing: Postman
Vector Store: Chroma (local)
Framework: LlamaIndex

This stack gets a beginner FDE from zero to a working RAG prototype with minimal configuration. All tools have strong documentation and active communities. Start here if you are new to the FDE role or preparing for certification. Costs are manageable on free tiers during learning.

🔵 AI Application Stack

Coding: Cursor
LLM API: OpenAI API + Anthropic API
Framework: LangChain + LlamaIndex
Vector Store: Weaviate or Pinecone
Deployment: FastAPI + Docker
Evaluation: LangSmith + Ragas
Automation: n8n

This stack is appropriate for building a full AI application with retrieval, integrations, and evaluation. It introduces more moving parts and is suitable for FDEs who are comfortable with Python and REST APIs. The combination of LangChain and LlamaIndex is powerful but requires understanding where each framework is most appropriate.

🔴 Production-Oriented Stack

Coding: Cursor + GitHub Copilot
LLM API: OpenAI / Anthropic / Azure OpenAI
Framework: LangGraph + LlamaIndex
Vector Store: Pinecone or pgvector
Cloud: AWS / Azure / GCP + Docker
CI/CD: GitHub Actions
Monitoring: LangSmith + Grafana + Sentry
Evaluation: DeepEval + Promptfoo
Security: Presidio + Garak + Secrets Manager

This stack reflects what a mature AI application deployment looks like. It prioritises observability, security, and maintainability alongside capabilities. Suitable for FDEs preparing production handovers or supporting enterprise-grade deployments.

FDE AI Tools Learning Roadmap

The following eight-stage learning path is designed to be completed progressively. Each stage builds on the previous one. You do not need to master every tool before moving forward – the goal is to develop enough working familiarity to use each tool effectively in a real customer engagement and to know when to reach for it.

1

Stage 1

AI Coding Assistant

Install Cursor or GitHub Copilot. Use it to generate a Python script that reads a CSV and prints a summary. Practise asking it to explain code line by line.

2

Stage 2

LLM APIs and Prompt Design

Call the OpenAI API directly in Python using the `openai` library. Write three different system prompts and observe how they change the output for the same user query.

3

Stage 3

API Integration and Debugging

Use Postman to call a public API (for example, a weather API or a public REST dataset). Build a FastAPI endpoint that calls that API and returns transformed data.

4

Stage 4

Basic AI Application

Build a Streamlit app that accepts a user question and returns an answer from an LLM. Add a simple system prompt that constrains the assistant to a specific topic.

5

Stage 5

RAG and Enterprise Data

Take 10 PDF documents, ingest them with LlamaIndex into a Chroma vector store, and build a query interface that answers questions using the document content.

6

Stage 6

AI Agents and Automation

Build a LangGraph agent that can call two tools – for example, a web search tool and a calculator – and decide which to use based on the user's question.

7

Stage 7

Evaluation and Security

Use Ragas to score the retrieval quality of the RAG application from Stage 5. Run Promptfoo against your Streamlit app from Stage 4 to test prompt injection resistance.

8

Stage 8

Deploy and Monitor

Containerise the application from Stage 5 with Docker. Deploy it to a cloud platform. Configure LangSmith to capture traces. Review the first 20 traces and identify one improvement.

Aim to complete one stage per week while working through real exercises. The practical exercises are more valuable than reading alone – learning by building is the fastest path to FDE readiness.

Final AI Tools Cheat Sheet

Use this section as a quick reference throughout your FDE work. It summarises the most practical guidance from this handbook in a format designed for fast consultation – not for reading end to end.

Essential Tools Every Aspiring FDE Should Explore

Cursor or GitHub Copilot

AI coding assistant – your daily driver

OpenAI API

The most accessible LLM API to start with

LlamaIndex + Chroma

RAG prototyping, zero-config vector store

Streamlit

Fastest path from Python to a working demo

Postman

API testing, debugging, and documentation

Docker + GitHub Actions

Portable deployment and CI/CD pipelines

Tool Selection Checklist for a Customer Project

- ◯ Define the customer problem before selecting any tool
- ◯ Identify existing stack – cloud, databases, languages, identity
- ◯ Confirm data privacy requirements (third-party API or self-hosted?)
- ◯ Assess realistic cost at the expected usage volume
- ◯ Check security and compliance certifications
- ◯ Evaluate vendor lock-in risk and available alternatives
- ◯ Confirm the customer's team can operate the tool after handover
- ◯ Build a prototype to validate the tool works in this specific context
- ◯ Set up evaluation baselines before any user-facing deployment
- ◯ Configure monitoring and alerting from day one in production

Common Mistakes When Choosing AI Tools

- Choosing tools based on trending status rather than customer fit
- Building AI agents for tasks that a simple API call would handle
- Deploying without an evaluation baseline or monitoring in place
- Ignoring vendor lock-in and assuming pricing will stay constant
- Skipping security testing before exposing AI applications to end users
- Attempting to learn every available tool rather than achieving depth in a core set

Tools to Learn Next — Based on Career Stage

Early Career

OpenAI API → LlamaIndex → Streamlit → Postman → Docker

Mid-Career

LangChain → LangGraph → LangSmith → Ragas → FastAPI → Weaviate

Senior / Production Focus

LangGraph → DeepEval → Garak → Presidio → Grafana → GitHub Actions

The FDE Principle: Right Tools, Real Problems

> Effective Forward Deployed Engineers succeed by choosing and combining the right tools to solve real customer problems – not by trying to learn every tool available.

The AI tooling landscape is vast, fast-moving, and often overwhelming. New frameworks, model releases, and platforms appear every week. It is impossible – and counterproductive – to try to stay current with every development. The FDE who can evaluate an unfamiliar tool quickly, integrate it where it adds value, and recommend against it where it does not, is far more valuable than one who has simply installed everything.

The tools in this handbook are not a curriculum to be completed in sequence. They are a reference to be returned to as you encounter real customer problems. Some tools you will use every week. Others you will reach for once a year when the specific situation calls for them. Depth in a core set – coding assistant, LLM API, RAG framework, deployment tools, evaluation tools – combined with broad awareness of what else exists, is the configuration that serves an FDE best.

Customer problems are the best teacher. Build things. Break them. Debug them. Evaluate them honestly. Deliver something a real user can touch. Each engagement will sharpen your judgement about which tools to trust and which to approach with caution. That judgement, built on practical experience rather than theory, is what the best FDEs have in common.

What This Handbook Covered

01

The FDE Role

What FDEs do and why AI tools matter

02

Coding Assistants

Cursor, Copilot, Claude, ChatGPT, Windsurf

03

Building AI Applications

APIs, frameworks, and data connectors

04

AI Agents

LangGraph, CrewAI, MCP, n8n

05

RAG and Enterprise Data

Vector databases and retrieval pipelines

06

Testing, Monitoring, Security

Evaluation, observability, and safety controls

07

Real-World Scenarios

Five practical FDE engagement examples

08

Learning Roadmap

Eight progressive stages with exercises

FORWARD DEPLOYED ENGINEERING | GSDC FDE CERTIFICATION | PRACTICAL AI HANDBOOK

GSDC
Global Skill Development Council

Globally Recognised Certification Body
190+ countries · 2,50,000+ Certified Professionals

IAS ACCREDITED Personnel Certification Body

Certified Forward Deployed Engineer

ABOUT GSDC CERTIFICATION

EBOOK

Extensive and exclusive Ebook created by world's experts to help you with understanding core concepts.

LEARNING MATERIALS

Get access to learning materials such as videos, ebooks, templates, and practice exams, which will help you clear the certification exam.

CREATED BY EXPERTS

GSDC certifications are created and authored by world's leading experts in the field.

LEARNING OBJECTIVE

- **Gain a strong understanding of Forward Deployed Engineering concepts, roles, tools, technologies, and development practices.**
- **Identify business and customer challenges and design AI-driven solutions aligned with real-world requirements.**
- **Apply modern AI frameworks, APIs, data integration, and deployment practices to build and implement solutions within existing business systems.**
- **Apply testing, evaluation, security, monitoring, and optimization practices to deliver reliable, scalable, and production-ready AI solutions.**

Enroll now with the code **LEARN20** To avail **20%** discount

Enroll Now

www.gsdcouncil.org