Welche KI gibt es online – Überblick für Profis

Category: KI & Automatisierung
geschrieben von Tobias Hager | 29. Dezember 2025

Welche KI gibt es online 2025: Der brutale Profi-Überblick ohne Buzzword-Bullshit

Du willst wissen, welche KI es online gibt, ohne dich durch Marketing-Nebel und Influencer-Romantik zu kämpfen? Gut. Hier kommt der ungeschönte, technische, produktionsreife Überblick über Modelle, APIs, Preise, Latenzen, Compliance, Orchestrierung und Evaluierung. Keine Hypes, nur belastbare Fakten und klare Handlungsempfehlungen. Wenn du nach "Welche KI gibt es online" suchst, bekommst du hier die Antwort – nicht die weichgespülte Version, sondern die, mit der Profis ihren Stack bauen und skalieren.

- Kompletter Marktüberblick: Welche KI gibt es online – von LLMs über

Bild- und Audio-Generatoren bis zu Agent-Systemen und Retrieval-Stacks
- Technische Kennzahlen, die zählen: Kontextfenster, Tokenpreise, Durchsatz, Latenz, Rate Limits, KV-Cache, Quantisierung und GPU-Anforderungen
- Top-Provider realistisch verglichen: OpenAI, Anthropic, Google, Mistral, Meta, Cohere, Stability, Midjourney, ElevenLabs, OpenRouter und Co.
- Wie du den richtigen Dienst auswählst: Modelleigenschaften, Sicherheitsanforderungen, Datenresidenz, Compliance und Kostenkontrolle
- Praxis-Patterns für Profis: RAG, Function Calling, Tool-Use, Orchestrierung mit LangChain, LlamaIndex, Haystack und Worker-Queues
- Echte Produktionsreife: Observability, Offline-Evaluation, Human-in-the-Loop, Guardrails, Promptleaks und Jailbreak-Resilienz
- Welche KI gibt es online aktuell für Bilder, Audio, Video – und welche Workloads lokal oder on-prem sinnvoller sind
- Benchmarks ohne Buzzwords: HELM, MMLU, HumanEval, MT-Bench, V-Bench und warum dein Use Case wichtiger ist als jeder Score
- Schritt-für-Schritt-Checklisten für Auswahl, Integration und Monitoring, damit “Welche KI gibt es online” nicht zur Zeitverschwendung wird
- Skalierung ohne Bauchschmerzen: Multi-Provider-Fallback, Caching-Strategien, Kostenbudgets, QoS-Policies und SLA-Realität

Die Frage “Welche KI gibt es online” klingt harmlos, ist aber in der Praxis ein Minenfeld aus Featuritis, Preismodellen und halbgaren Versprechen. Welche KI gibt es online, die nicht nur Demos füttert, sondern produktiv 24/7 liefert? Welche KI gibt es online, die mit deinem Datenschutz, deiner Latenz, deinen Qualitätsmetriken und deinen Agent-Workflows klarkommt? Welche KI gibt es online, die du orchestrieren kannst, ohne jede Woche das halbe Prompting zu revidieren? Und welche KI gibt es online, die du morgen austauschen kannst, wenn der Anbieter plötzlich Preise verdoppelt oder die API bricht? Genau darum geht es hier – praxisnah, systematisch, gnadenlos konkret.

Bevor wir in Modelle und Provider einsteigen, klären wir die professionelle Linse, durch die du “Welche KI gibt es online” betrachten musst. Erstens: Es geht nicht um Inspiration, es geht um deterministische Outputs unter realer Last. Zweitens: Jede Entscheidung ist ein Trade-off zwischen Qualität, Kosten, Latenz und Kontrolle. Drittens: Die Architektur gewinnt, nicht das einzelne Modell – Multi-Provider, Feature-Flags, Offline-Evals und Logging sind Pflicht. Viertens: Sicherheit ist keine nachträgliche Dichtung, sondern integraler Teil des Designs. Fünftens: Roadmaps ändern sich, Verträge bleiben; plane für Austauschbarkeit. So denken Profis, wenn sie sich die Frage “Welche KI gibt es online” stellen und am Ende mit einem Stack dastehen, der Monate statt Tage überlebt.

Wir teilen die Landschaft entlang der Workloads auf: Text-LLMs für Generation, Analyse und Tool-Use; Multimodale Modelle für Vision, OCR und Audio; Generative Medien für Bilder, Audio und Video; Agent-Systeme für mehrschrittige Aufgaben; Retrieval-Komponenten für Faktenstabilität; und Governance-Schichten für Sicherheit und Compliance. Für jeden Block prüfen wir API-Design, Kontextfenster, Tokenisierung, Sampling-Parameter, Throughput, Preisgestaltung und Betriebsrisiken. So wird aus “Welche KI gibt es online” kein Bastelprojekt, sondern ein messbarer, revisionssicherer Produktionsdienst. Das klingt trocken, ist aber genau der Unterschied

zwischen Showcase und Umsatz.

Welche KI gibt es online: Kategorien, Use-Cases und Pro-Terminologie

Online-KI gliedert sich für Profis zunächst in klare Kategorien, weil Use-Cases sonst diffus bleiben. Erstens gibt es Large Language Models für Textverständnis, Textgenerierung, Tool-Use und Steuerlogik, die als generelles Steuerhirn dienen. Zweitens existieren multimodale Modelle, die Text, Bild, Audio und in Teilen Video simultan verarbeiten und Antworten über Modalitäten hinweg integrieren. Drittens stehen spezialisierte Generatoren für Medien bereit, etwa Diffusionsmodelle für Bilder, Vocoder und TTS-Pipelines für Sprache sowie Motion-Modelle für Video. Viertens sind Agent-Frameworks zu nennen, die planen, entscheiden und externe Tools orchestrieren, um Aufgabenketten robust abzuarbeiten. Fünftens gehören Retrieval- und Wissensschichten dazu, insbesondere Vektordatenbanken und Indizes, damit Modelle faktenfest bleiben. Sechstens runden Sicherheits- und Governance-Layer das Ganze ab, damit nichts brennt, wenn reale Daten und Prozesse drankommen.

Damit die Diskussion nicht in Buzzwords untergeht, brauchen wir präzise Begriffe mit technischer Bedeutung. Kontextfenster bezeichnet die maximale Anzahl von Tokens, die das Modell gleichzeitig verarbeiten kann, und es bestimmt, wie viel Prompt, Historie oder Dokumenteninhalt du einspeisen kannst. Tokenisierung ist der Prozess, Eingabetext in Subworteinheiten zu zerlegen, typischerweise per Byte-Pair Encoding, und beeinflusst Kosten, Latenz und Genauigkeit. KV-Cache meint das Zwischenspeichern der Key/Value-Matrizen von Transformerschichten, um bei langen Konversationen oder Streaming die Generationszeit pro Token zu reduzieren. Funktionales Tool-Use oder Function Calling beschreibt die Fähigkeit, strukturierte Funktionsaufrufe auszugeben, die dann von deiner Anwendung mit echten Tools ausgeführt werden. Und RAG, also Retrieval-Augmented Generation, bezeichnet das Nachladen passender Wissensschnipsel aus Indizes, um Halluzinationen zu reduzieren und aktuelle Fakten einzuspeisen.

Use-Cases lassen sich entlang von Komplexität und Risiko priorisieren, was harte Anforderungen an die Provider stellt. Klassische Content-Augmentation, etwa Zusammenfassen, Umschreiben und Übersetzen, ist latenzsensibel, aber relativ fehlertolerant und gut mit Standard-LLMs abbildbar. Code-Assistenz, SQL-Generierung und Datenanalysen verlangen hingegen präzise Steuerung, deterministischere Decoding-Parameter und strenge Tests, insbesondere mit HumanEval-ähnlichen Benchmarks. Wissensarbeit mit Compliance-Bezug, also Rechts-, Finanz- oder Medizintexte, benötigt RAG, Audit-Logs, Guardrails und Datenflüsse, die DSGVO-konform sind. Agentische Automatisierung wie E-Mail-Triage, Ticket-Bearbeitung oder Onboarding erfordert robuste Tool-Use-Ketten, Re-tries und Observability, damit Fehlentscheidungen schnell abgefangen

werden. Und überall gilt: Ohne Telemetrie, Kostenkontrolle und Rollback-Kapazität spielt man Feature-Roulette statt Produktentwicklung.

LLMs, Multimodalität und Text-APIs: Modelle, Kontext, Kosten und Latenzen

Die großen Namen im LLM-SaaS-Bereich sind etabliert, aber ihre Stärken unterscheiden sich im Detail. OpenAI bietet mit GPT-4-Varianten und multimodalen Ablegern starke Reasoning-Fähigkeiten, breite Tooling-Unterstützung und flächendeckende SDKs, jedoch mit proprietärer Abschottung. Anthropic setzt mit der Claude-3-Familie auf ausgedehnte Kontextfenster, vorsichtige Sicherheitsmechanismen und starke Instruktionsbefolgung, was in regulierten Umgebungen oft punktet. Google liefert mit Gemini-Modellen tiefe Multimodalität und solide Integration in Cloud-Ökosysteme, was für Teams mit GCP-Stack attraktiv ist. Mistral und Cohere fokussieren auf effiziente APIs, europäische Datenhaltung oder Enterprise-Kontrollen, was in Compliance-getriebenen Umfeldern entscheidend sein kann. Meta Llama-Modelle sind offen lizenziert und via Hoster oder Self-Hosting nutzbar, wodurch Kontrolle und Kostenersparnis bei stabilen Workloads möglich werden.

Technisch entscheidet oft nicht der Markenname, sondern Metriken, die selten im Marketing stehen. Kontextlänge beeinflusst Prompt-Design und RAG-Strategien, denn jenseits einiger hundert K Tokens nehmen Qualität und Kosten nichtlinear zu. Sampling-Parameter wie Temperatur, Top-p, Top-k und Repetition Penalty steuern Kreativität und Determinismus und gehören versioniert in dein Prompt-Repo. Kosten werden pro 1.000 Tokens abgerechnet, getrennt für Eingabe und Ausgabe, und Streaming kann Latenz für die UI deutlich senken. Rate Limits und Durchsatz pro Minute limitieren Skalierung, weshalb Batch- und Prefill-Strategien mit KV-Cache in hochfrequenten Pipelines wertvoll sind. Für sensible Workloads sind Datenretention-Policies, Log-Opt-Outs und On-Request-Deletion essenziell, andernfalls riskierst du Trainingsleaks oder Audit-Probleme. Und ganz banal: SLA, Statusseiten-Historie und Incident-Reaktionszeit schlagen jedes Whitepaper.

Auch Multimodalität ist kein Selbstzweck, sondern ein Architekturthema mit Kosten- und Qualitätsfolgen. Vision-Inputs benötigen vernünftige Aufbereitung, etwa OCR-Vorverarbeitung, Caching von Embeddings und sinnvolles Chunking für Dokumente mit Tabellen. Audio-Eingaben per Speech-to-Text profitieren von Domain-Adaptionen, Vocab-Hints und robustem Postprocessing, weil Nummern, Namen und Codes sonst entgleisen. Tool-Use in multimodalen Modellen ist mächtig, setzt aber strenge JSON-Schemata, Retry-Logik und idempotente Tools voraus. Für viele Anwendungen ist die Kombination aus schlankem Text-LLM und spezialisierter Vision- oder STT-Komponente günstiger und genauer als ein “Alleskönner”. Und wenn dein Workload konstant und wiederholbar ist, lohnt sich Feintuning oder Adapter-Layer wie LoRA auf offenen Modellen, um Kosten und Latenz dauerhaft zu drücken.

Welche KI gibt es online für Bilder, Audio und Video: Diffusion, Vocos, TTS und Motion

Im Bildbereich dominieren Diffusionsmodelle und Transformer-Hybride, die sich in Qualität, Prompt-Treue und Lizenzlage unterscheiden. Anbieter wie Midjourney bieten starke Ästhetik und Konsistenz, aber eingeschränkte Steuerung und Enterprise-Transparenz. Stability AI liefert mit Stable Diffusion und Varianten eine umfangreiche Open-Source-Ökologie, die sich über Hoster bequem betreiben und anpassen lässt. Adobe und Co. punkten mit Lizenzsicherheit und Integrationen in Kreativ-Workflows, was in Markenumfeldern oft den Ausschlag gibt. Technisch wichtig sind Prompt-Kontrolle, Sampler-Parameter, Guidance-Scale, Auflösung, ControlNet-Module für Layouts und Inpainting/Outpainting für iterative Kreativarbeit. Für Produktionspipelines spielen außerdem Inferenzzeit, Batch-Generierung und Kosten pro Ergebnis eine größere Rolle als schicke Showcase-Bilder.

Im Audiobereich unterscheiden wir STT, TTS und Voice-Cloning, die jeweils eigene Tücken haben. Für Speech-to-Text zählen Wortfehlerrate, Latenz, Sprechertrennung, Sprachenabdeckung und Domänenvokabular, weshalb Vocab-Hints und Post-Korrektur mit regulären Ausdrücken oder LLM-Checks Gold wert sind. Text-to-Speech erfordert natürliche Prosodie, korrekte Betonung und Latency unter UI-Schmerzgrenzen, vor allem bei Dialogsystemen. Voice-Cloning ist rechtlich heikel und muss rechtssicheres Consent-Management, Wasserzeichen oder Stimmverschleierung beinhalten. Anbieter wie Deepgram, Whisper-basierte Services, AssemblyAI, ElevenLabs und AWS/Google/Microsoft-Stacks decken unterschiedliche Preis-Leistungs-Sweet-Spots ab. Und wie immer gilt: Inferenzkosten, Rate Limits und Ausfallstrategien entscheiden darüber, ob dein Bot um 9 Uhr morgens produktiv ist oder stottert.

Videogenerierung ist die neue glänzende Kugel, aber mit realen Produktionsgrenzen. Modelle für Text-zu-Video liefern beeindruckende Clips, kämpfen jedoch mit Konsistenz, Physik und feinen Details über mehrere Sekunden. Für Marketing reicht das oft, für Produktdemos oder Education braucht es Struktur, Storyboards und Postproduktion. Realistische Workflows splitten in statische Assets, animierte Sequenzen, TTS und Schnitt, statt alles in ein Wunderprompt zu pressen. Video-Latenz ist ungleich höher, weshalb Queues, Vorproduktion und Caching obligatorisch sind. Und wenn Rechte, Musik und Markenrichtlinien im Spiel sind, sind Audit-Trails und Render-DNA wichtiger als der Wow-Effekt im ersten Frame.

Agents, Tool-Use und Orchestrierung: RAG, Function Calling, LangChain und Co.

Agentische Systeme sind kein magisches Bewusstsein, sondern deterministische Steuerlogik mit stochastischem Kern. Der Agent plant, ruft Tools über definierte Schemata auf, validiert Ergebnisse und entscheidet, was als nächstes passiert. Function Calling mit strengen JSON-Schemas, robuste Parser, Zeitouts, Circuit Breaker und idempotente Tools sind die Lebensversicherung gegen Kettenchaos. RAG senkt Halluzinationen, funktioniert aber nur mit sauberem Index, sinnvollem Chunking, passenden Embeddings und Relevanz-Scoring, das dein Domänenwissen abbildet. Multi-Step-Reasonsing braucht Checkpoints, Re-tries und fehlerbewusste Policies, etwa Self-Consistency oder Toolformer-ähnliche Muster. Ohne Observability siehst du nur Prompts und Hoffnungen – mit strukturierter Telemetrie erkennst du Dead-Ends, Kostentreiber und verbesserst systematisch.

Orchestrierungsframeworks beschleunigen, aber verschleiern gern technische Schulden. LangChain bietet Bausteine für Chains, Agents, Memory und Tools, doch ohne Disziplin entsteht schnell Komplexität mit magischen Side Effects. LlamaIndex fokussiert Retrieval-Qualität, Indizes, Reranker und Evaluierung für dokumentengestützte Systeme. Haystack ist in vielen Enterprise-Stacks erprobt und spielt besonders gut mit Open-Source-Komponenten. Für schwere Lasten solltest du Worker-Queues, Feature-Flags, A/B-Routing, Canary-Deployments und Circuit Breaker außerhalb der LLM-Library betreiben. Und ja, eine simple, eigene Orchestrierung über klaren Code, Message-Bus und Telemetrie ist oft wartbarer als zehn Schichten Metaframework.

Tool-Use reift erst mit starker Typisierung, Fehlerklassen und Eingabevalidierung. JSON-Schemata brauchen Versionierung, Defaultwerte, Enums und Grenzen, damit der Agent nicht in Unfug rennt. Antworten aus Tools müssen validiert, harmonisiert und bei Bedarf durch ein Validierungslayer mit Grammar-Constrained Decoding oder Structured Outputs geschleust werden. Für RAG sind Evaluation und Offline-Experimente Pflicht: NDCG, MRR, Recall@k, Passage-Anteil im finalen Output und Query-Klassifizierung zählen. Hybrid-Retrieval mit dichten und spärlichen Indizes, Reranking mit Cross-Encodern und Query-Rewriting sind keine Kür, sondern Standard. Und wenn du wirklich automatisierst, brauchst du Rollen, Berechtigungen, Audit-Logs und ein Revisionskonzept, sonst baut dein Agent dir in fünf Minuten 500 falsche Rechnungen.

- Tool-Use stabilisieren: JSON-Schema definieren, Parser erzwingen, Fehlerklassen abbilden, idempotente Funktionen bauen, Timeouts setzen, Retries begrenzen.
- RAG hartmachen: Quellen normalisieren, sinnvolles Chunking, Embeddings testen, Retrieval evaluieren, Reranking aktivieren, Output mit Zitatstellen versehen.

- Orchestrierung produktiv machen: Message-Queues, Circuit Breaker, zentralisierte Logs, Kostenmetering, Feature-Flags, Canary-Routing, Fallback-Provider.
- Beobachtbarkeit sichern: Prompt- und Tool-Traces, Token-Statistiken, Qualitätsmetriken, Fehlertaxonomie, Dashboards und Alerting.

Compliance, Sicherheit und Governance: EU AI Act, DSGVO, PII und Promptleaks

Ohne Governance ist jede Online-KI eine Haftungsfalle mit freundlicher API. Der EU AI Act und bestehende Regelwerke verlangen Risikoanalyse, Dokumentation, Transparenz und in Teilen strenge Qualitäts- und Aufsichtspflichten. DSGVO macht klar: Personenbezogene Daten brauchen Rechtsgrundlage, Zweckbindung, Minimierung und Löschkonzepte. Datenresidenz, Auftragsverarbeitung, Subprozessoren und Transfer in Drittländer sind Vertrags- und Architekturthemen, nicht Fußnoten. BYOK und eigene KMS-Schlüssel sind bei sensiblen Daten Standard, ebenso Verschlüsselung at-rest und in-transit. Und Promptleaks sind real: Ohne Sanitization, PII-Filter, Secret-Scanning und Kontext-Reduktion landen vertrauliche Informationen schneller im Modellkontext als dir lieb ist.

Security beginnt vor dem Prompt und endet nicht mit einem hübschen Output. Eingaben brauchen Validierung und Red Teaming gegen Prompt Injection, Jailbreaks und Indirekte Injection via Dokumente oder Links. Output muss auf Policy-Verstöße geprüft werden, idealerweise mit separaten Klassifizierern, Regeln oder Sicherheits-Layer wie Guardrails. Tool-Use verlangt strikte Sandboxing-Strategien, begrenzte Berechtigungen, Least Privilege und Monitoring jeder Aktion. Für Audits benötigst du vollständige Traces, die Promptversion, Parameter, Modellversion, verwendete Tools, Datenquellen und Nutzerkontext enthalten. Und du brauchst Prozesse: Rollende Reviews, Change Logs, Notfallplan, Verantwortlichkeiten und Schulungen, sonst bringt die beste Technik nichts.

Provider-seitig sind Datenschutz-Features nicht verhandelbar, wenn es ernst wird. Keine Trainingsnutzung deiner Daten, klare Retentionszeiten, Logging-Opt-Out und regelmäßige Pen-Tests sind Mindeststandard. Private Endpunkte, regionale Rechenzentren, vertragliche SLAs und Support-Reaktionszeiten gehören in den Vertrag, nicht ins Wunschdenken. Für besonders sensible Workloads ist Self-Hosting oder VPC-Hosting mit offenen Modellen eine valide Option, gepaart mit MLOps-Standards, Secrets-Management und Observability-Stacks. Und last but not least: Dokumentation ist Compliance, schreibe auf, was dein System tut, warum es das tut, mit welchen Risiken – und wie du sie kontrollierst.

- Eingaben härten: PII-Maskierung, Secret-Scanner, Content-Filter, Kontextbegrenzung, Link-Sandboxing, Indirekte-Injection-Checks.
- Abläufe regeln: Verantwortliche benennen, Änderungsprozesse definieren,

Red-Teaming planen, Vorfälle dokumentieren, Notfallmaßnahmen testen.
- Verträge klären: AVV, Subprozessorenliste, Datenresidenz, SLA, Support, Audit-Rechte, Exit-Strategie und Datenlöschung.

Evaluation, Benchmarks und Monitoring: Qualität messen statt hoffen

Benchmarks sind hilfreich, aber dein Use Case schlägt jeden Score. Öffentliche Tabellen wie MMLU, Big-Bench, HumanEval, GSM8K, MBPP oder MT-Bench geben eine Tendenz, erklären aber nicht, ob dein Spezialjargon korrekt verstanden wird. Baue deshalb einen domänenspezifischen Evalsuite mit realen Prompts, Gold-Labels und klaren Metriken, die Geschäftsziele abbilden. Für Generationen nutzt du automatische Metriken wie BLEU, ROUGE, BERTScore und Semantik-Ähnlichkeit, ergänzt durch menschliche Bewertungen. Für RAG misst du Retrieval-Qualität mit Recall@k, NDCG, MRR, Passage-Abdeckung und Zitiergenauigkeit. Für Tool-Use brauchst du Erfolgsquote pro Schritt, Fehlerklassen, End-to-End-Completion-Rate und Median-Latenz unter Last.

Monitoring ist mehr als “hat geantwortet”. Du brauchst Telemetrie pro Anfrage: Modell, Version, Parameter, Tokenkosten, Latenz, Fehlertyp, Toolpfad, Quellen, Nutzerkontext. Caching-Raten, Fallback-Häufigkeiten und Abbruchgründe zeigen dir, wo Geld verbrannt wird. Für Kostenkontrolle setzt du Budgets, Quoten und Kosten-Alerts pro Team, Projekt und Feature-Flag. Live-Quality kann mit LLM-as-a-Judge, Heuristiken und stichprobenbasierter Human-in-the-Loop-Prüfung begleitet werden. Drift-Erkennung warnt dich, wenn Antworten abgleiten, etwa nach Provider-Updates oder Datenänderungen. Ohne diese Infrastruktur ist jede Optimierung ein Glücksspiel und jeder Ausfall ein Drama.

Offline-Experimente verhindern, dass dein Produktivsystem zum Labor wird. Du versionierst Prompts, Chain-Logiken, RAG-Parameter und Tool-Schemata und testest Varianten systematisch gegen deine Eval-Sets. Eine saubere CI-Pipeline führt Evals automatisch aus und blockiert Rollouts bei Qualitätsrückfall. Canary-Deployments und A/B-Tests validieren Effekte in der Realität, bevor die gesamte Nutzerschaft betroffen ist. In Verbindung mit Feature-Flags kannst du riskante Änderungen gezielt und reversibel ausrollen. So verwandelst du LLM-Entwicklung von Alchemie in Ingenieurwesen.

- Evals aufsetzen: Use-Case definieren, Gold-Daten kuratieren, Metriken wählen, Benchmarks implementieren, CI integrieren, Review-Prozess etablieren.
- Monitoring bauen: Tracing, Metriken, Logs, Dashboards, Alerts, Kostenlimits, Rate-Limits, Fallback-Regeln, Incident-Playbooks.
- Kontinuierlich verbessern: Drift erkennen, Ursachen analysieren, Hypothesen testen, Small-Batch-Rollouts, Regressions verhindern.

Auswahl-Framework: So findest du die richtige Online-KI für dein Team

Wähle nie ein Modell, wähle eine Architektur mit Exit-Strategie. Beginne mit der Problemdefinition, nicht mit dem Logo, und übersetze Anforderungen in messbare Metriken für Qualität, Latenz, Kosten und Sicherheit. Erstelle eine Shortlist aus 3–5 Providern, die deine Mindestanforderungen abdecken, etwa Kontextlänge, Tool-Use, Datenresidenz und Preisbereich. Teste alle Kandidaten auf deiner Eval-Suite, nicht auf Demo-Prompts, und vergleiche robuste Kennzahlen unter identischen Parametern. Plane Fallbacks und Multi-Provider-Routing von Anfang an, damit du bei Incidents, Preissprüngen oder Qualitätsdrift handlungsfähig bleibst. Und schreibe Policies, wie neue Modelle oder Versionen in deine Landschaft kommen, sonst wird dein Stack zum Zoo.

Kostenbetrachtung ist mehrstufig und beginnt auf Token-Ebene, endet aber in Architekturkosten. Rechne Input- und Output-Token mit realistischen Promptgrößen, Antworthöhen und Retries, nicht mit Schönwetterzahlen. Berücksichtige Latenz- und Durchsatzbedarfe, denn Skalierung kostet mehr als nur Tokenpreise; Queueing, Parallelisierung und Caching machen den Unterschied. Prüfe Rate Limits, Priorisierungsmodelle und Reservierungen beim Provider, damit Peak-Lasten nicht zur Lotterie werden. Denke auch an Personalkosten: Eine API mit reifem SDK, klarer Doku und stabilen Semantiken spart Monate gegenüber hippen, brüchigen Lösungen. Und dokumentiere deine Annahmen; Budgetkontrolle ohne Annahmen ist eine Fiktion.

Technischer Zuschnitt entscheidet, ob du morgen noch ruhig schläfst. Prüfe, ob Structured Outputs stabil sind, ob Function Calling Schema-Treue hält und ob Validierungsfehler sauber behandelbar sind. Schau dir Observability-Features an: Request-IDs, korrelierbare Logs, Statusseitenhistorie, dedizierte Supportkanäle. Verifiziere Sicherheitsmechanismen, Datenflüsse, Retention-Optionen, BYOK, regionale Endpunkte und Vertragsklarheit. Und verhandle Off-Ramps: Datenexport, Migrationshilfen, Preisanpassungsklauseln, Notfallkontakte. Gute Anbieter scheuen das nicht; schlechte schon.

1. Use-Case klären: Ziel, Risiken, Metriken, Constraints definieren und schriftlich fixieren.
2. Shortlist bilden: Funktionale Mindestkriterien, Compliance, Preisrahmen, regionale Anforderungen.
3. Eval-Suite ausrollen: Einheitliche Prompts, Parameter, Messung, Regressionstests.
4. Architektur planen: Multi-Provider, Fallbacks, Caching, Observability, Kosten-Policies.
5. Verträge schließen: AVV, SLAs, Datenresidenz, Support, Exit-Strategie, Eskalationspfade.
6. Rollout steuern: Feature-Flags, Canary, Monitoring, Incident-Playbooks,

Postmortems.

Zum Schluss: Welche KI gibt es online ist eine falsche Einzahlfrage in einem Mehrzahlmarkt. Du brauchst bewusst mehrere Bausteine, die du austauschen kannst, wenn Anforderungen, Preise oder Qualität sich verschieben. Ein starker Stack ist modular, beobachtbar, evaluiert und durch Policies geschützt. So baust du keine Abhängigkeit, sondern einen Wettbewerbsvorteil. Und du ersparst dir den peinlichen Satz “wir warten gerade auf die Antwort vom Modell”, wenn die Produktion brennt.

Wenn du bis hierher gelesen hast, hast du verstanden, dass “Welche KI gibt es online” nur der Anfang ist. Die eigentliche Kunst ist, aus den Bausteinen einen verlässlichen, sicheren und bezahlbaren Produktionsdienst zu bauen. Konzentriere dich auf Messbarkeit statt Magie, auf Architektur statt Anekdoten, auf Prozesse statt Bauchgefühl. Dann lieferst du Ergebnisse, nicht nur Demos. Willkommen in der Realität der Profis. Willkommen bei 404.