ENTERPRISE AGENTIC AI PLATFORM

Botlit

Build, govern and ship AI agents that answer from your own knowledge, run on the models you already approved, and leave a record of every single run.

BOT APPS AND AGENTS

BRING YOUR OWN MODEL

KNOWLEDGE AND RETRIEVAL

GOVERNED BY DEFAULT

PER-RUN COST LEDGER

Pre-launch. Currently accepting waitlist registrations ahead of general availability.

01 The short version

Botlit is where a company hires, trains, supervises and retires its AI workforce — and where anyone with the right permission can see exactly what those agents said, what it cost, and which version said it.

Two ideas carry the whole product. A bot app is the front door people talk to — the thing you would call a Customer Support Bot or an HR Assistant. Behind it sit agents: specialists with their own instructions, their own model and their own knowledge, such as a billing agent, a technical agent and an escalation agent. One front door, many specialists, and a router that decides who takes each message.

Agents are grounded rather than improvised. You load your documents into a knowledge base, choose how they are split up, and the platform indexes them so an agent can search them while it is composing an answer — with a standing instruction to say plainly when your documents do not cover the question. You choose the model too: register the providers your organisation has already approved, add your own keys, and bind each agent to a specific model.

Everything the agents do is written down. Every run keeps its question, its answer, its status, how long it took, which model served it, how many tokens it used and what it cost. Prompts and model choices are versioned like releases, so a change that makes answers worse is one rollback away. And the limits you set — how often an agent may run, how much it may spend, what it may never repeat — are checked before the model is called and again on what it says.

> THE PROMISE
>
> **Build agents you can trust, because you can see, version and govern everything they do.**

Botlit in four numbers

11

Model provider families

Register the providers your organisation has approved, keep your own keys and contracts, and bind each agent to a model from that list.

10

Kinds of shareable package

Apps, agents, tool packs, prompt templates, skills, retrieval recipes, policy bundles, channel templates, agent entries and knowledge templates.

7

Types of guardrail

Content, tool, data, response, budget, rate-limit and retention rules, written once per workspace and reviewable by the people who own the risk.

BYOK

Your keys, your bill

Botlit resells no inference. Agents run on your provider accounts, and the platform records what each run cost rather than marking it up.

02

Who it is for

Digital agencies and consultancies

Build one hardened assistant, then deliver it again and again. Each client gets an isolated workspace, and a package can be installed linked to your upstream improvements or forked into a copy the client owns outright. Per-client run costs are visible before the invoice goes out.

Enterprise IT and AI platform owners

You are accountable for an agent estate you did not personally build. Botlit puts permissions, model choice, guardrails, versioning and an audit record in one control plane, so the AI surface sits inside the same identity and approval model as everything else you run.

Support and service operations leaders

Deflect the repetitive questions without losing sight of what the bot is saying. Every conversation from every channel lands in one shared inbox, with the run record sitting behind each reply so a complaint about last Tuesday is a lookup rather than an investigation.

Knowledge and content owners

Your documents are the product. Choose how each corpus is split, index it, then prove retrieval quality by running the questions your users actually ask and reading back which passages come out — before anyone relies on the answers.

Compliance, risk and governance reviewers

Guardrails are objects you can read, review and version rather than paragraphs buried in a prompt. Rate, spend and content limits are evaluated in the request path, and the record of who published which agent version is added to rather than overwritten.

Technology startups and scale-ups

Ship the assistant inside your own product without operating a vector store, a credential vault, a prompt-versioning scheme and an audit log. Everything the console does is available to your code, and machine credentials pass the same permission checks a person does.

Finance and budget owners

AI spend usually surfaces four weeks late on a provider invoice. Here every run records its own tokens and cost against the agent, the app and the workspace, so the unit economics of a workload is a filter rather than a modelling exercise.

03 What breaks today

Most teams do not fail at building an AI assistant. They fail at the year afterwards — when nobody can say which prompt changed, which model answered, what it cost, or what it is allowed to do. These are the failures Botlit is shaped around.

Nobody can answer what changed

An assistant that was working starts hedging, over-apologising or getting details wrong. The prompt has been edited by four people across three months and there is no record of who moved what, or when the quality actually turned.

- Agents are versioned like code. You iterate in a draft, publish with a changelog, and restore the previous version in one step. The rollback is recorded as well, so the lineage is complete rather than convenient.

The model vendor owns your roadmap

A platform that speaks to exactly one model provider makes every future decision for you: pricing, data-handling terms, regional availability, and what happens when a better or cheaper model appears somewhere else.

- Register eleven families of model provider with your own keys, verify each with a real connection test, and bind each agent to a specific model. Moving an agent to a cheaper model is a binding change and a published version, not a migration project.

The bot invents things confidently

A fluent, plausible, wrong answer about a refund policy is worse than no answer. It is the single objection that stops customer-facing rollouts, and generic chat interfaces have no structural defence against it.

- Retrieval runs inside the agent turn. Your knowledge bases are searched as the answer is composed, the closest passages ride along with the question, and the agent is instructed to say plainly when your documents do not cover it.

03 What breaks today, continued

Guardrails exist only in a document

The policy on what an assistant may discuss, spend or touch lives in a slide deck. Nothing in the request path reads it, so the first time a rule is tested is in production, in front of a customer.

- Policies are objects the runtime reads. Rate, spend and content rules are evaluated before the model is called and again on its reply, and a blocked answer is replaced with one clear refusal rather than a leaked fragment.

AI spend is invisible until the invoice

A prompt gets longer, a model gets swapped for a better one, an integration starts retrying. Four weeks later finance asks why the line item doubled and the only honest answer is a shrug.

- Every run writes its own receipt: tokens in and out, resolved cost, duration, the model that served it, and how many tools and knowledge passages it used. Cost attribution by agent, app or workspace is a query.

Access control is bolted on afterwards

Permissions get reimplemented feature by feature, machine access becomes a shared master key, and the AI surface quietly becomes the one part of the estate that nobody can prove is governed.

- Every operation carries an access rule checked before it runs, for people and for software alike. A pipeline credential goes through the same checks as a person, so revoking it is the same operation as revoking a colleague.

03

Two more, and the fix

One bot cannot be everything

A single assistant that handles billing, technical faults and escalation ends up with a system prompt too long to maintain and a personality that is wrong for every one of those conversations.

- A bot app fronts several specialist agents, and the router decides who takes each message: always one, in rotation, by rules you author, or by matching the keywords and phrasings you gave each specialist, with a fallback when it is unclear.

Good work cannot be reused

The support bot that finally felt right took six weeks. The next client, department or region starts from an empty page because there is no unit of distribution beyond copying and pasting a prompt.

- Ten kinds of package can be installed into another workspace, either linked so they keep taking your upstream improvements, or forked into a copy that team fully owns. A linked install can be forked later without losing its configuration.

04 What you get

Botlit is a control plane, not a framework. These are the objects you actually work with, and each one is a first-class thing you can version, permission, package and audit.

FRONT DOOR

Bot apps

The assistant your customers or employees talk to. It carries a type, a lifecycle from draft to live to paused, a visibility setting controlling who in your organisation can see it, and published versions.

SPECIALISTS

Agents

The workers behind the front door. Each has its own instructions, its own model binding and declared capabilities — retrieval, tools, delegation, memory — plus versioning with publish and rollback, and its own run statistics.

DISPATCH

App-to-agent routing

Links that connect one app to many agents with a declared strategy, per-agent priority, trigger keywords, intent patterns and a fallback, so the front door reaches the right specialist without custom code.

GROUNDING

Knowledge bases

Managed retrieval corpora. Documents are split with the strategy you choose, embedded through your own provider, and moved through an explicit index lifecycle so you always know whether a corpus is actually searchable.

What you get, continued

TUNING

Retrieval profiles

A reusable recipe for how retrieval should behave — mode, how many passages to pull, score thresholds. Attach one to an agent, or distribute it so a configuration that works for legal documents stops being tribal knowledge.

MODELS

Provider and model registry

Workspace-scoped registrations of the model providers your organisation approved, with encrypted credentials, a real connection test before anything depends on them, and per-model parameters.

REACH

Channel connectors and routes

A connector binds your workspace to a platform; a route maps a specific channel or thread to a specific bot app. One workspace can run a customer bot in one place and an internal helpdesk in another.

GUARDRAILS

Policies, skills and prompt templates

Seven kinds of policy with per-tool risk levels, named reusable skills, and versionable prompt templates with variable placeholders — so the approved disclaimer lives in one place rather than pasted into forty agents.

EVIDENCE

Executions and analytics

A record for every run, successful or not: status, input, output, duration, tokens, cost, model, tool calls and retrieved passages, rolled up per agent and per app, with live updates available to your own monitoring.

05 Put to work

Sixteen concrete jobs teams bring to Botlit, in the order a real deployment tends to grow: get something live, ground it, choose the model, put it where people are, then govern what you have built.

OPERATIONS LEAD

From signing up to a live bot app in one sitting

- A persisted, listed bot app ready for its first agent — no infrastructure to provision and no configuration files.

AGENCY BUILDER

Start from something that already works

- A working starting point in minutes, with a clean uninstall because a package knows what it brought with it.

AGENCY FOUNDER

Build once, deliver many, and price it properly

- A productised service line with per-client run economics visible from one seat, so pricing is a decision rather than a guess.

BOT BUILDER

Treat prompts and model choices like releases

- Prompt changes stop being irreversible, and the rollback itself is on the record.

BOT BUILDER

Test in chat, audit the same act

- Testing and auditing become one action, so comparing two prompt candidates is comparing two run records.

KNOWLEDGE CURATOR

Prove retrieval quality before anyone relies on it

- A corpus you have actually validated, and a retrieval recipe saved as a reusable profile you can attach elsewhere or share.

05 Put to work, continued

KNOWLEDGE LEADER

Grounded answers, or an honest admission

The bind. A model left to its own parameters will produce a confident, fluent, wrong answer about your policy — and that single behaviour is what stops customer-facing rollouts.

- Answers shaped by your documents, with a bot that degrades to a slightly worse answer rather than an error in front of a customer.

ENTERPRISE IT OWNER

Your model strategy stays yours

The bind. Which provider, under whose contract, with what data-processing terms is a procurement decision — and most agent platforms make it for you.

- Your keys, your contracts and a record of exactly which model served every single run.

SUPPORT OPERATIONS LEAD

The router that decides who takes the message

The bind. A real support surface is a front door with several specialists behind it, and most platforms make the dispatch decision a configuration field nothing ever reads.

- Most messages routed at no inference cost, ambiguous ones sent to a destination you configured rather than a coin flip.

SUPPORT LEAD

Meet people where they already are

The bind. Nobody wants another destination app. The questions are already in Slack, in Teams, in the community server and on the phone.

- Agents reachable on Slack, Microsoft Teams, Discord, WhatsApp and plain webhooks, with the same limits and the same refusal wording as in-app chat.

SUPPORT LEAD

One shared inbox for everything the agents said

The bind. The first question after switching on an AI agent is what it is actually saying to people, and if answering it means logging into four platforms the rollout stalls there.

- Oversight with receipts, so the bot said something odd on Tuesday becomes a thread and a run record rather than a reconstruction.

BUDGET OWNER

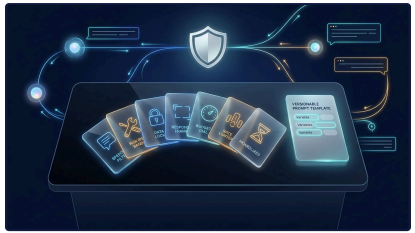
Know what each agent costs to run

The bind. What does the bot cost us is the question that decides whether an agent programme grows or gets quietly shut down, and most platforms answer it a month late.

- Unit economics you can price a retainer against or defend a budget with, backed by the run ledger rather than an estimate.

05

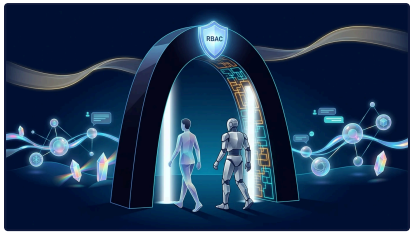
Put to work, and governed



COMPLIANCE OFFICER

Write the rules of engagement down where they bind

A reviewable rulebook that the request path actually reads, plus policy bundles you can ship to every workspace you oversee.



ENTERPRISE IT ADMIN

Machine access that is governed, not exceptional

No shadow authentication path and no bypass lane, so revoking a pipeline is the same operation as revoking a colleague.

AUTOMATION ARCHITECT

Compose steps and agents on a visual canvas

Orchestration inside the same identity, credential and audit model as the agents it drives, with no second credential store.

PLATFORM ENGINEER

Approve the tool surface before agents can act on it

A reviewable inventory — these servers, these tools, this status — instead of tool access discovered after the fact.

06

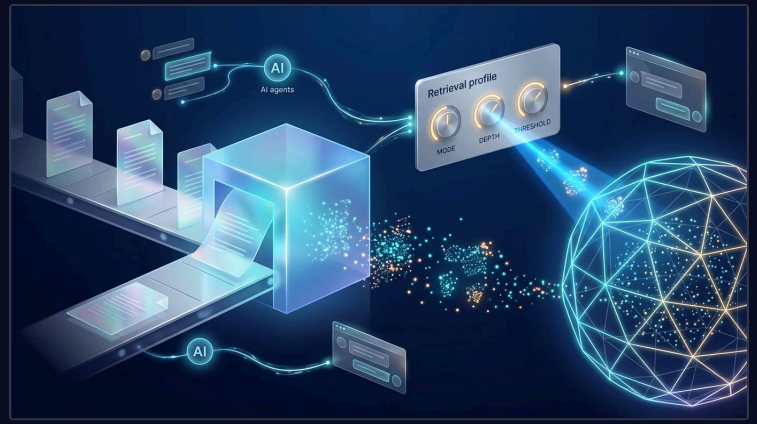
Built AI-native

There is no non-AI half of this product. The agent turn is the whole thing.

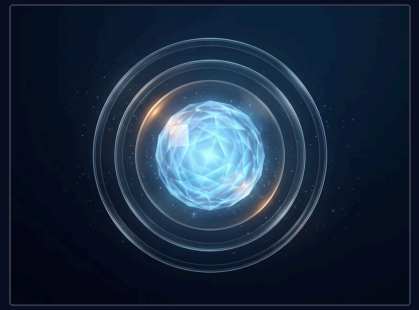
Most software adds a chat box to something else. Botlit's entire job is the agent: the model it runs on, the knowledge it reads, the tools it can reach, the rules it obeys and the record it leaves. That is why the interesting engineering is not in the interface — it is in what happens between somebody's question and the answer that comes back.

That turn is built as a sequence you can inspect rather than a single opaque call. Your limits are evaluated first, cheapest check to most expensive, so a runaway integration is stopped before it consumes any inference at all. Then the question is routed to the right specialist. Then your knowledge is searched and the closest passages ride along with the question. Then the agent may call your tools or hand the work to a colleague agent. Then the reply is checked again against your content rules before anyone sees it.

The last step is the one buyers underrate. Every run — succeeded, failed, timed out or cancelled — becomes a record carrying the question, the answer, the duration, the tokens, the resolved cost, the model that served it, how many tools it called and how many knowledge passages it read. Because that record is written by the same code path that runs the agent, there is no instrumentation to forget and no sampling. If it ran, it is in the ledger.

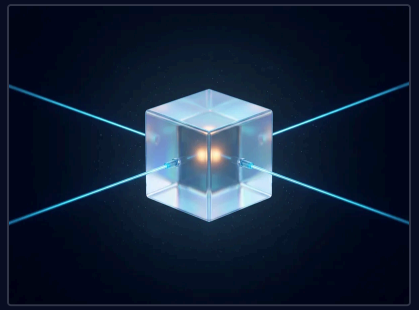


Four pillars



Agents are the whole product

Bot apps, agents, knowledge, tools, rules and run history are not features bolted onto something else. There is no legacy half to keep in step, and no non-AI path that quietly behaves differently.

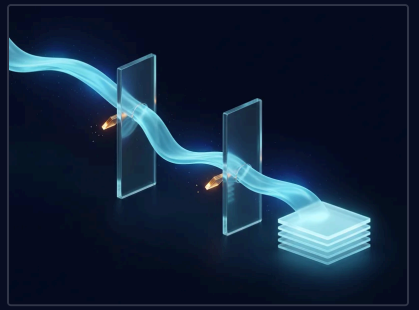


Your models, your keys, your knowledge

Agents run on providers your organisation already approved, with keys you add and keep. No inference is resold and no agent is routed through a shared model service. A model server you host yourself connects the same way any other provider does.

Grounded before it is generative

Your documents are searched while the answer is being written, not in a separate search box, and the agent is told to admit a gap rather than invent something plausible. Only corpora that have finished indexing are consulted.



Every answer is checked and recorded

Your limits apply before a model is called and again to what it says, and a blocked reply becomes one clear refusal rather than a leaked fragment. Either way the run leaves behind its time, cost, model and tool usage.

06

What the AI can actually do

Retrieval inside the turn

A retrieval-enabled agent searches every knowledge base linked to it as part of answering, and the closest passages are numbered and placed in front of the model with the question.

Grounding a live answer is a different thing from having a search feature, and it is the behaviour that decides whether a bot is safe to put in front of customers.

Honest ignorance

The retrieved context arrives under a standing instruction: use this where it is relevant, and if it does not contain the answer, say so rather than fabricating one.

A confident wrong answer costs more trust than a plain admission, and the instruction is part of the runtime rather than something each builder has to remember.

Graceful retrieval degradation

A corpus still indexing is skipped rather than half-searched. A corpus that cannot be queried at all logs the reason and the turn continues ungrounded instead of failing.

A bot that answers slightly worse is better than a bot that returns an error to a customer mid-conversation.

Ingestion you can configure

Documents are split by fixed window, recursive splitting, semantic boundaries, Markdown structure or code structure, embedded in batches through your own provider, with dimensions validated at ingest.

A policy PDF and a source repository need different treatment, and a mismatched configuration is better rejected on upload than discovered as poor recall six months later.

Exact similarity search

Stored passages are compared with the question exactly rather than approximately, and the store sits behind a driver interface so a dedicated vector engine can be substituted without touching the agent runtime.

At workspace and department scale, exactness beats approximation — and the product states the ceiling rather than quietly truncating past it.

Tool calls mid-conversation

Botlit opens a real session against your Model Context Protocol servers, lists their tools, calls what the model asks for, feeds the results back and lets the agent finish — with a hard cap on iterations.

An agent that can only talk is a search box with better manners; the cap is what stops a tool-hungry loop becoming an unbounded bill or a hung request.

06 Tools, delegation, routing, limits

Network transports only

Streamable HTTP and the legacy event-stream transport are supported. Local command connections are refused outright, and every outbound address is screened before anything is sent to it.

Tool configuration is customer-authored data in a shared backend; honouring a local command field would be a remote code execution primitive rather than a feature.

Delegation that really runs

An agent can hand work to a specialist, which executes its own turn with its own model, knowledge and tools and returns a real answer into the same conversation.

Multi-agent is the most over-claimed phrase in the category; in most products it means a table of records and a message row nothing ever picks up.

Bounded hand-offs

Delegation depth is capped and cycles are rejected, so an agent already in the active chain cannot be delegated to again. The delegate option is only offered to the model while depth budget remains.

The bounds are what make agent-to-agent traffic safe to switch on rather than an interesting way to exhaust a timeout.

Routing as a runtime

Four dispatch strategies decide which specialist takes each message, scoring keywords and intent patterns first and escalating to a model classifier only when that is inconclusive.

Most messages get routed at zero inference cost and identically on every run, which is a reliability decision as much as a cost one.

Limits before inference

Rate limits are evaluated first, then spend against the day's consumption, then the message itself against blocked keywords, patterns you supply and an opt-in set of conservative personal-data heuristics.

Cheapest check first means a runaway integration is stopped before it costs anything, and an over-the-line request is refused rather than billed.

Output checked too

The model's reply is evaluated against the same content rules, and a violation returns one explicit refusal — identical wording in the app and on a channel — instead of a partial answer or a raw error.

Half a blocked answer is still a leak, and two different failure messages across two surfaces is how an incident becomes a support ticket storm.

06

Models, embeddings and the ledger

Per-agent model binding

Each agent binds to a specific model with its own parameters, so a reasoning-heavy escalation agent and a routine triage agent need not share a model or a price point.

Matching model cost to task complexity is the single largest lever on agent economics, and here it is a version bump rather than a rewrite.

Embeddings on your provider

Knowledge indexing and search use the embedding model you configured, per knowledge base, through your own provider account.

The corpus that represents your business should not be embedded by a vendor you never chose, on terms you never read.

The run ledger

Every run writes status, input, output, duration, token counts split by direction, resolved cost, the model, tool-call count and retrieved-passage count, aggregated per agent and per app.

It answers the questions that follow a bad week: which agent's cost tripled and after which publish, and which runs answered with zero retrieved passages.

Usage counted where it happens

The units a plan prices are counted as they occur. A bot that fans a message out to its own specialists is counted once rather than once per agent.

Counting a single logical bot several times for its own internal orchestration is the kind of metering that quietly destroys trust in a bill.

06

And what keeps it in bounds

Permission check on every operation

Including asking an agent a question. The AI path gets no shortcut of its own, and machine callers pass the same checks as people.

The workspace is the wall

Agents, documents, keys, conversations and run records belong to exactly one workspace, taken from who is asking rather than what the caller claims. Retrieval never leaves it.

Your limits, not our defaults

Rate, spend and content rules come from your workspace. A rule you set is genuinely binding; one you never set changes nothing, so enforcement never starts blocking traffic a tenant did not ask to have blocked.

Tool risk classified in the open

Each tool a server exposes can be classified low, medium or high risk in a policy, so the conversation about what an agent may do happens before it can do it.

Trust levels on agent traffic

Agents in the registry carry a trust level — unverified, verified publisher or internal — and tool policies can require a minimum level before a connection is made or a message is sent.

Credentials that never come back

Provider keys are encrypted at rest, resolved only to make the call, redacted from responses and kept out of logs. They are never stored inside an agent definition.

06

In bounds, continued

Destinations screened

Every address you point the platform at — a model endpoint or a tool server — must be public and secure. Private, loopback, link-local and metadata addresses are refused with a readable message.

Versions are added, not overwritten

Publishes and rollbacks accumulate with their changelogs, so which version said that is a lookup rather than an excavation across chat exports.

Failures are recorded too

A run that failed, timed out or was cancelled still writes a record with structured error detail. The ledger is not a success log.

Cost is recorded, not resold

Agent execution runs on your own provider keys, so what the ledger reports is your own inference bill, shown for visibility rather than charged as platform credit. The two are kept separate on purpose.

Audit belongs to the platform

Configuration changes and agent operations flow into the shared activity trail rather than a private log table, so the AI surface is reviewed in the same place as the rest of the estate.

Reachable from, and in numbers

REACHABLE FROM

- In-app chat with any agent, showing which model answered and what it used
- Slack, Microsoft Teams, Discord, WhatsApp and plain inbound webhooks
- The shared team inbox, where every conversation from every channel lands together
- Your own code, through the full API that the console itself uses
- Automated pipelines and product backends, via machine credentials that pass the same permission checks
- The embedded visual workflow builder, where a step can drive an agent
- Your Model Context Protocol tool servers, which the agent reaches out to mid-conversation
- Other agents, internal or registered, reached as delegate options inside the same turn

IN NUMBERS

- Eleven families of model provider can be registered per workspace, each with its own key and connection test
- Five chunking strategies: fixed window, recursive, semantic, Markdown-aware and code-aware
- Four dispatch strategies decide which specialist answers: single, rotation, rule-based and keyword-and-classifier
- Seven kinds of policy: content, tool, data, response, budget, rate limit and retention
- Three trust levels on registered agents: unverified, verified publisher and internal
- Nine channel types, including a custom type for anything not on the list
- Ten kinds of package can be installed linked to upstream or forked into a copy you own
- Every run records tokens split by direction, resolved cost, duration, model, tool calls and retrieved passages

STATED PLAINLY

What it does not do yet

What it does not do yet, stated plainly. Retrieved passages shape an answer but are not yet returned as a structured source list, so nothing can display citations underneath a reply — that is the next step on this path. Tool calling and agent-to-agent delegation run on the OpenAI-compatible provider path only; agents on the other provider families still get normal single-turn responses until each one's native tool-calling format has its own adapter. Replies do not stream yet. Text arriving from your documents, your tools and other agents still enters the prompt unfiltered, and a destructive tool call is not yet held for human confirmation — both are designed and ticketed, not shipped. Per-agent tool allow-lists, a spend view sliced by app, agent and model, and anomaly alerts when an agent drifts from its own baseline are on the same list. Botlit publishes no accuracy figure, because the evaluation set that would make such a number mean anything is still being built.

07

Who does what

Botlit ships a role for each person who touches an agent estate, so a knowledge curator, a compliance reviewer and an operations lead can all work in the same workspace without any of them holding more access than their job needs.

Platform administrator

MODEL STRATEGY, TOOL RISK, TENANCY, CREDENTIAL HYGIENE

The IT or AI platform owner who decides what the workspace's agent estate is allowed to do at all — which models, which tools, which rules and which outside agents are trusted.

Bot builder

BEHAVIOUR QUALITY, ITERATION SPEED, SAFE ROLLBACK

The maker who designs the front door and the specialists behind it, writes the instructions, wires the routing and tests in chat before anything is published.

Knowledge curator

RETRIEVAL QUALITY, INDEX FRESHNESS, SOURCE COVERAGE

The content owner who makes sure agents answer from correct, current material — and who proves it rather than assuming it.

Bot operations lead

LIVE BEHAVIOUR, CHANNEL HEALTH, INTERVENTION SPEED

The service leader who runs published bots day to day, watches live conversations, and steps in when a run fails or a channel misbehaves.

Governance auditor

EVIDENCE, REVIEWABILITY, SEPARATION OF DUTIES

The compliance or risk reviewer who inspects configuration, policies, provider bindings and run evidence to prove how the estate is governed — without changing anything.

Developer and integrator

API PARITY, CREDENTIAL SCOPE, COST PER ACTIVE USER

The engineer embedding Botlit inside another product or pipeline, treating the console as a debug surface and the API as the actual interface.

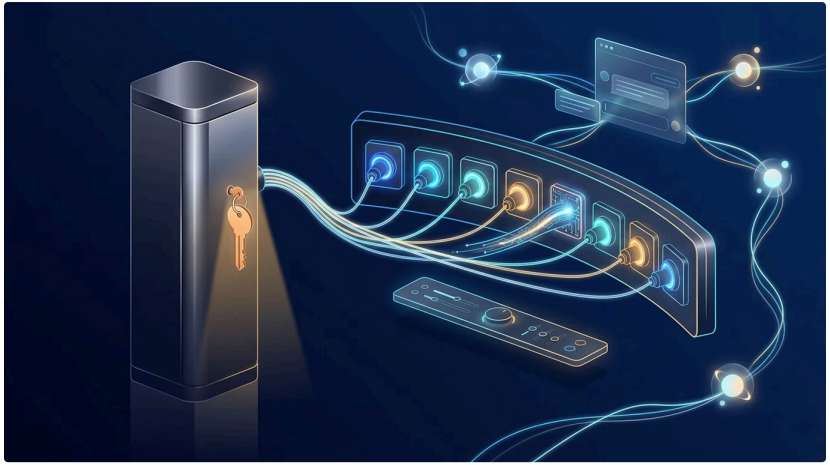
08

Under the hood

Botlit is a managed control plane with a real agent runtime underneath it. The console, your own code and an inbound channel message all travel the same route: an API edge that establishes who is asking and what they are allowed to do, then a service that owns bot apps, agents, knowledge, providers, channels, guardrails, packages and the run ledger. There is no second, privileged path for the AI features.

The agent turn itself is a defined sequence rather than one opaque call: evaluate the workspace's limits, dispatch to the right specialist, retrieve from the linked knowledge bases, call the bound model, execute any tool calls or delegations the model asks for within a bounded loop, evaluate the reply, then write the record. Each stage is independently testable, and the routing logic in particular is written as a pure function with no database or network dependency so every strategy is exercised deterministically.

Nothing about inference is hidden. Credentials resolve at run time through a vault rather than living in an agent definition, the request goes to the provider endpoint you registered, and the tokens and cost that come back are recorded against the run. Swapping a model is a binding change and a published version — which is the whole point of separating what Botlit owns from what your model provider owns.



THE SPLIT THAT MATTERS

Botlit owns the agents, the knowledge, the rules, the channels and the record. Your model providers own the inference, on your contracts and your keys. That line is deliberate: it is why there is no house model to be locked into, why a self-hosted model server is a first-class citizen, and why moving an agent to a cheaper model is a version bump rather than a migration.

Capability by area

Agent runtime

A defined turn: limits, dispatch, retrieval, model call, bounded tool and delegation loop, output check, record. Runs identically for in-app chat, a channel message, a delegated task and a workflow-triggered run.

Knowledge and retrieval

Five chunking strategies, batched embedding through your own provider, per-corpus collections with an explicit index lifecycle, dimension validation at ingest, exact similarity search behind a pluggable driver interface, and reusable retrieval profiles.

Model layer

Eleven provider families with encrypted, run-time-resolved credentials, per-model parameters, real connection tests, and a resolution cache so credential lookup does not dominate response latency.

Routing

Four dispatch strategies with per-link priority, trigger keywords, intent patterns and a fallback agent. Rotation state lives on the link so it survives restarts; the classifier is a fallback, not the first move.

Tools and delegation

A genuine Model Context Protocol client over network transports, with live tool discovery, a bounded call loop, delegation depth caps, cycle rejection, and outbound endpoint screening on every address.

Channels

Connectors and routes for nine channel types. Inbound webhooks are acknowledged before the agent runs; outbound messages are stored and delivered in parallel so the audit trail and the delivery never drift apart.

Versioning and records

Draft, published and superseded versions with changelogs on agents and apps, and a run record for every execution including failures, with metrics rolled up per agent and per app and streamed as live updates.

Automation

A full visual workflow builder embedded in the product — the same engine used across the wider platform — with versioned canvases, step-level run history and credential adapters that hand vaulted model keys to an agent step.

Standards and integrations

OPEN STANDARDS IT SPEAKS

- Model Context Protocol — Botlit connects as a client to the tool servers you run
- Agent-to-Agent protocol for cross-framework agent interoperability
- A complete GraphQL API — everything the console does is available to your own code
- The OpenAI-compatible chat completions format, so any endpoint that speaks it plugs straight in
- Streamable HTTP and event-stream transports for tool servers; local command transports refused by design
- OAuth 2.0, API-key and basic authentication for third-party integrations
- Inbound and outbound webhooks for event-driven flows
- Common document formats for ingestion, including PDF, Word, Markdown, plain text, CSV and HTML
- Open embedding models of your choosing, configured per knowledge base
- TLS 1.3 in transit and AES-256 at rest

PLUGS INTO

- OpenAI, Anthropic, Google, Cohere, Mistral, Groq, Together and HuggingFace as model providers
- AWS Bedrock and managed enterprise OpenAI deployments for teams with a cloud data boundary
- A custom provider type for any self-hosted model server that speaks the OpenAI wire format
- Slack and Microsoft Teams for internal helpdesk and employee-facing assistants
- Discord and WhatsApp for community and customer-facing conversations
- Generic inbound webhooks, plus a custom channel type for anything not on the list
- Your own Model Context Protocol servers, whatever they wrap behind them
- Third-party services registered under the platform integration vault by category — CRM, communication, productivity, cloud storage, development, calendar, email, payment, analytics and accounting
- The embedded FluidGrids visual workflow engine, for orchestration around the agents
- The wider Burdenoff Workspaces platform for identity, permissions, files, notifications, analytics and billing

How it deploys

Managed, multi-tenant

Botlit runs as a hosted service. There is no agent runtime for you to host, monitor or scale, and no vector infrastructure to operate before the first agent answers.

Workspaces as the boundary

Every object belongs to exactly one workspace, so a multi-team enterprise or an agency running many clients gets isolation by construction rather than by naming convention.

Inference wherever you want it

Public provider accounts, an enterprise cloud deployment under your own contract, or a model server on your own infrastructure reached as an OpenAI-compatible endpoint. Content that must not leave your network does not have to.

API-first from day one

The console is a client of the same API you get. Machine credentials pass the same permission checks as a person, so pipelines and product backends are governed rather than excepted.

Regional footprint

Service state currently resides in a single region in India. Multi-region residency and cross-region disaster recovery are on the roadmap and are not claimed today.

09

Security & trust

Botlit agents handle your customers' conversations and your internal knowledge, so the security posture is written for the person who reads the questionnaire before they read the brochure. Here is what is in place, and further down, what is not.

Authorisation on every operation

Every query and every mutation carries an access rule that is checked before it runs. There is no per-feature authorisation code to get subtly wrong and no unguarded path into the AI surface.

Identity established at the edge

Authentication happens once, centrally, and the caller's identity — person, application or automated service — arrives with the request. The product's own services never validate tokens themselves.

Workspace tenancy from the caller

The workspace is derived from who is asking rather than from what the request claims. A caller cannot widen its own scope by naming a different workspace in a filter.

Encryption in transit and at rest

TLS 1.3 protects data in transit and AES-256 protects it at rest, across conversations, documents, run records and configuration alike.

Credential vaulting

Model provider keys and integration credentials live in a managed secret store, are resolved only at execution, are never written into an agent definition, and are redacted from API responses and logs.

Outbound endpoint screening

Every address the platform is pointed at must be public and secure. Private, loopback, link-local and cloud metadata addresses, and internal-only hostnames, are refused with a readable error rather than silently ignored.

Local command execution refused

Tool server connections that would run a local command are rejected outright, because connection configuration is customer-authored data in a shared backend and honouring it would be a code execution primitive.

09

Security controls, continued

Input and output content checks

A configured content policy is evaluated on the incoming message and again on the model's reply, covering blocked keywords, patterns you supply, and an opt-in set of conservative personal-data heuristics.

Spend and rate ceilings

Rate-limit and budget policies are evaluated before any model is called, so a runaway integration is stopped before it consumes inference rather than after the invoice arrives.

Trust-gated agent traffic

Connections and messages between agents can be gated on a minimum trust level, so a workspace that only wants its own agents talking to each other says so once as policy.

Append-only history

Run records, publishes and rollbacks accumulate rather than being overwritten, so which version said that, on which model, at what cost is answered by a lookup.

Supply-chain scanning in the build

Container and dependency vulnerability scanning, infrastructure-as-code checks and secret scanning run as gates in the build pipeline, alongside image scanning in the registry.

Documents through the platform file service

Knowledge documents are stored and served through the shared platform file service rather than an ad-hoc bucket, inheriting its access controls and lifecycle handling.

Coordinated vulnerability disclosure

Security reports are accepted through the published contact channel with a prompt acknowledgement and coordinated disclosure.

STATED PLAINLY

Where the compliance posture stands

Botlit holds no security certification today, and says so rather than implying otherwise. Controls are being aligned against SOC 2 criteria, but no accredited auditor has been engaged and no report exists; ISO 27001 has not been scoped. A HIPAA business associate agreement is not available — and that is a commercial blocker as much as an engineering one, because the chain would have to include whichever model provider a customer chooses. The GDPR baseline is real and shipped: data subject access, erasure and portability, with a published data processing agreement, acceptable use policy, responsible AI policy and subprocessor list. CCPA is covered by the same posture. Single sign-on with SAML, advanced role administration and audit exports are Enterprise-tier commitments that are not yet built. On the AI side specifically: text arriving from documents, tools and other agents still enters the prompt unfiltered, and a destructive tool call is not yet held for confirmation. Every enterprise competitor in this category has at least SOC 2 today. If a certification is a hard prerequisite for your procurement, Botlit is not the right choice this quarter, and pretending otherwise would waste your evaluation cycle.

10

One product, many front doors

One control plane, reached from wherever the work happens. The web console and the API are live surfaces; the distributable clients below are specified and in build, and are marked as such rather than listed as if you could install them today.

Web console

The full product in a browser: workspaces, bot apps, agents, knowledge, retrieval profiles, tool and agent registries, channels, guardrails, run records, packages, workflows and settings.

Public API

Everything the console does, available to your own code through a complete GraphQL surface with machine credentials that pass the same permission checks as a person.

Channel connectors

Slack, Microsoft Teams, Discord, WhatsApp, generic inbound webhooks and a custom type, each mapped by routes to a specific bot app.

Shared team inbox

Every conversation from every channel in one place, with both sides of each exchange and the run record sitting behind each agent reply.

Embedded workflow builder

A full visual automation canvas inside the product, with versioned workflows, step-level run history and credential adapters that feed vaulted model keys to an agent step.

Package store

Ten kinds of distributable package, installable linked to track upstream improvements or forked into a copy you own, with fork-after-install when a linked copy outgrows the standard build.

Front doors in build and planned

Templates

Reusable app and agent blueprints that live in your workspace, private, workspace-visible or public, and can themselves be packaged and distributed.

Documentation site

Getting started, core concepts, agent design, API reference, tool protocol and security sections, published as a public documentation site.

Node and Python SDKs

In build, not yet published. Typed access to agents, knowledge, runs, channels and governance for Node and Bun projects, and an async client for notebooks and data pipelines.

Command-line client

In build, not yet published. A terminal client for driving agents, knowledge bases, runs, channels and deployments across workspaces.

Tool server

Planned. A Model Context Protocol server exposing Botlit itself, so an outside AI assistant can drive your agents. Botlit is a good client of your tool servers today; the reverse direction is not built.

Editor, browser and mobile

Planned. An editor extension for building and debugging agents, a browser side panel for quick access, and native mobile apps for monitoring an estate on the move.

11

Plans

Three editions, published ahead of launch so evaluation can start now. Botlit is pre-launch: plan records exist and prices are set, but checkout is not yet open and waitlist members get the first onboarding slots.

FREE

$0

Evaluation, proofs of concept and individual builders who want to hold a real agent in their hands before committing anything.

- Free forever
- Up to 10 bots
- 1,000 runs per month
- 2,000 messages per month
- 2 knowledge bases
- Bring your own model keys
- Per-run token and cost records

PRO

$49 / month

Teams running a real workload — a support deflection bot, an internal helpdesk, an in-product assistant — that needs channels, versioning and packages.

- $468 per year, equivalent to $39 per month
- Up to 100 bots
- 50,000 runs per month
- 50,000 messages per month
- 20 knowledge bases
- Tool and agent registries, channel connectors and routes
- Package store with linked and forked installs, and priority support

ENTERPRISE

Custom

Organisations governing an agent estate at scale, and regulated teams whose procurement starts with the security questionnaire.

- Everything in Pro, at unlimited scale
- Multi-agent orchestration
- Single sign-on and SAML
- Advanced role administration and audit exports
- Custom tool servers and model endpoints
- Custom quotas and branding
- Custom service level and a named contact

What to know before you price it

What is actually metered

Bots created, runs triggered and messages answered are counted as they happen and enforced against your plan. The knowledge-base figures above are documented targets rather than hard caps today, and Botlit would rather say so than let you discover it during a rollout.

Model spend is separate, and yours

Agents run on your provider accounts, so inference is billed to you by your provider directly. Botlit records what each run cost for visibility and resells no inference, which is why there is no credit currency to buy, expire or reconcile.

Availability

Botlit is pre-launch. Plans are published and seeded, but payment processing is not yet enabled and no purchase flow is live. Waitlist registration is the route to early access and launch notification.

Enterprise shape

Single sign-on, advanced role administration, audit exports, custom quotas, branding and service-level commitments are Enterprise-tier items still being built out. Early conversations shape which land first, so tell us what your procurement blocks on.

12

Why teams choose it

Governance is structural, not a plan feature

Every operation carries an access rule enforced before it runs, for people and machines alike, and the workspace is derived from the caller rather than the request. Most competitors add role-based access as a tier upgrade or a directory integration.

A receipt for every run, not a credit dashboard

Tokens split by direction, resolved cost, duration, model, tool calls and retrieved passages, on every run including the failures. The alternative most vendors offer is an aggregate credit balance that cannot be attributed to an agent or a client.

No house model and no resold inference

Eleven provider families, your keys, your contracts, and a self-hosted model server as a first-class option. There is no margin for Botlit in which model you choose, which is exactly why the choice stays yours.

Ten kinds of forkable package

Not just app templates: prompt templates, skills, retrieval profiles, policy bundles, channel templates, tool packs, agent entries and knowledge templates all travel as packages, installable linked or forked. No comparable product treats a guardrail set as a shippable asset.

Workspace-per-client isolation as a native shape

Agencies and multi-team enterprises get tenancy walls by construction rather than folder discipline, with per-client run economics readable from one seat. Single-workspace products make this a spreadsheet exercise.

Open protocols as managed entities

The tool protocol and the agent-to-agent protocol are governed registries with connection testing, discovery and trust levels — not an integration someone wired into a configuration file where nobody can review it.

A real workflow engine inside the product

The visual builder is the same production engine used across the wider platform, with versioned canvases and step-level run history, not a lightweight flow editor bolted on for the demo.

Roles that match who actually shows up

A platform administrator, a builder, a knowledge curator, an operations lead and a read-only governance auditor each get a role shaped for their job, so a compliance reviewer can prove governance without being handed edit rights.

Status labels published on the product's own site

Every capability carries a shipped, in-progress or roadmap label in public, and the roadmap items are named rather than blurred. That is unusual enough in this category to be a differentiator in itself.

Honest answers to hard questions

You have no security certification. How can we buy this?	You may well not be able to yet, and that is the honest answer. There is no SOC 2 report, no ISO certification and no available HIPAA agreement; controls are being aligned against SOC 2 criteria but no auditor has been engaged. The GDPR and CCPA baseline is real and published. If a certificate is a hard prerequisite for your procurement, take the evaluation to a certified vendor now and revisit Botlit when a report exists.
Half these features sound like they are still being built.	Some are, and they are labelled. Retrieval inside the answer, routing, guardrail enforcement, versioning and the run ledger are running today. Tool calling and agent-to-agent delegation run on the OpenAI-compatible provider path only. Citations under an answer, streaming replies, the drafting copilot and the scheduled fleet supervisor are designed and ticketed but not shipped. The product is pre-launch and does not pretend otherwise.
Is this actually multi-agent, or a table of agent records?	A fair suspicion, because that is what the phrase usually means. Here a delegation invokes the target agent for real — its own model, its own knowledge, its own tools — and folds the response back into the conversation, with the depth capped and cycles rejected. The limitation is the provider path, not the mechanism.
Why should we run our documents through another vendor's retrieval?	You are not. Embedding and inference run through your own provider account under your own contract, and search never leaves your workspace. What Botlit adds is the pipeline around it: configurable splitting, dimension validation at ingest, an explicit index lifecycle and the ability to prove retrieval quality before anyone relies on it.
Everything is inside one browser console. Our workflows are code.	The console is a client of the same API you get. Machine credentials pass identical permission checks, run updates are available as a live stream, and installs, publishes and rollbacks are all scriptable. The published SDKs and command-line client are in build; today the API is the integration surface.

Hard questions, continued

What happens when a model provider changes its pricing or terms?	You change the binding and publish a version. Nothing about an agent's knowledge, guardrails, channels or history is tied to a provider, and the previous version stays one rollback away. That is the whole reason the control plane and the inference are separated.
How do we know the guardrails actually do anything?	Because they run in the request path, not in a document. Rate limits are checked before spend, spend before content, and content again on the reply, with a single explicit refusal when something is blocked. Worth knowing: enforcement is deliberately fail-open, so a workspace that configures no policy behaves exactly as it did before. A policy you set binds; one you never set does not start blocking traffic by surprise.
Your interface is not as polished as the canvas-first tools.	Correct. Products built around a visual design canvas have a better canvas, and if design-led flow building is your primary need they are the stronger choice. Botlit optimises for the year after launch: versioning, permissions, cost attribution and evidence.
We would be an early customer of a pre-launch product.	You would. What that buys is influence over sequencing — the roadmap items above are ordered, not fixed — and early onboarding slots. What it costs is certifications that do not exist yet and reference clients that do not exist yet, because Botlit has neither and will not manufacture either.
Can this handle a knowledge base at real scale?	At workspace and department scale, yes, and it says where the ceiling is. Retrieval compares your question against every stored passage exactly rather than approximately, which is precise but bounded; past the stated limit the product warns instead of quietly truncating, and the store sits behind a driver interface so a dedicated vector engine can be substituted without touching the agent runtime.

Where it sits against the alternatives

VS CLOUD-SUITE AGENT BUILDERS

Botlit does not require a particular cloud tenancy, identity stack or productivity suite, and your model strategy stays negotiable rather than defaulting to one vendor's endpoint. The concession is real: those products have generally available tool and agent-protocol support across every model they offer, a mature compliance portfolio, and far more channel connectors than Botlit's nine.

VS CRM-NATIVE AGENT SUITES

If your agents need to span several systems and several model providers, a suite whose full value depends on a particular CRM edition is an expensive shape. Botlit has no CRM prerequisite and gives per-run cost receipts instead of a credit aggregate. Conceded: for deep native actions on CRM objects and cases, and for a runtime trust layer that has been enforcing personal-data masking for years, the incumbent is genuinely ahead.

VS SUPPORT-DESK AI AGENTS

Botlit is not support-only — the same platform runs internal helpdesk, HR, sales qualification and operations agents — and it does not price by resolution or lock you to one proprietary model. Conceded: for a team already living in that helpdesk, nothing beats it on time to first resolution, its outcome-based pricing genuinely aligns cost with value, and its compliance portfolio is far broader than ours.

Against the alternatives, continued

VS NO-CODE CHATBOT CANVASES

Once one bot becomes an estate, the questions change to who published what, what it costs and who is allowed to change it — which is where versioning, per-run receipts, workspace isolation and a forkable package store start earning their keep. Conceded: those products get a first bot live faster, their free tiers are more generous, and their visual builders are more polished than ours.

VS OPEN-SOURCE AGENT FRAMEWORKS

A framework hands you maximum control and the entire operational burden: hosting the runtime, building the credential vault, the versioning scheme, the permission model and the audit log yourself. Botlit is the managed control plane instead. Conceded: they have the broadest model and tool ecosystem by a wide margin, they are self-hostable, and their graph-based orchestration is a better abstraction for some problems.

VS SINGLE-VENDOR MODEL ASSISTANT APIS

An assistant toolkit from a model vendor is the fastest path to an agent on that vendor's models, and the slowest path off them. Botlit keeps the provider list open, adds workspace permissions the toolkit does not have, and ships native channel deployment rather than leaving you to build it. Conceded: their built-in retrieval is production-ready today, their reasoning models lead the field, and there is no platform fee at all.

START TODAY

Start with the honest version

Botlit is pre-launch, and the fastest way to decide whether it fits is to read the status labels on its own site rather than take a summary's word for it.

- Join the waitlist for an early onboarding slot and launch notification.
- Read the AI page, where every capability carries a shipped, in-progress or roadmap label.
- Work through the use cases that match your team and note which ones say they need the API rather than the console.
- Bring your security questionnaire to the security page first, so the certification gaps surface before the evaluation does.

JOIN THE WAITLIST
botlit.ai/waitlist

HOW THE AI ACTUALLY WORKS
botlit.ai/ai

USE CASES IN DEPTH
botlit.ai/use-cases

SECURITY POSTURE
botlit.ai/security

TALK TO A PERSON
contact@botlit.ai

Botlit is published by Algoshred Technologies Pvt. Ltd., Chennai, India. Capabilities described here reflect the product as of August 2026; items still in progress are labelled throughout.