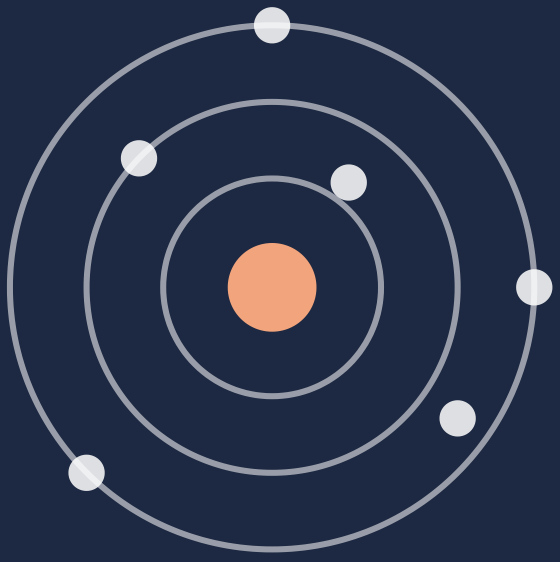
TEMPLATE AND WORKSHEETS



AI Agent Governance Framework

A template you can adopt in a week

Dr. Hitesh Sarda · AI advisor, Hyderabad

hitesh.in

A fill-in register, autonomy tiers, blast-radius worksheet, checklists and approval matrix for leaders who are about to let AI agents act in real systems and need to be able to say who owns each one, what it can touch and what it did.

How to use this template

This is a working document. Print it, or copy the tables into whatever your team already uses, and fill it in with the people who run the agents. I would rather you finish a rough version this week than a polished one next quarter.

An agent, for this document, is any AI system that takes actions with tools or credentials: a coding agent on a laptop, a support bot that can issue refunds, a scheduled job that reads tickets and updates records.

> A workable week: Day 1, fill the agent register (Part 2). Day 2, give every agent an autonomy tier (Part 3). Day 3, complete a blast-radius worksheet for your three most used agents (Part 4). Day 4, check those three against the pre-deployment checklist and the audit log requirements (Parts 5 and 6). Day 5, agree the approval matrix and the first-hour page, then put the first drill and the first quarterly review in the calendar (Parts 7 to 9). The four tiers, the roles in the approval matrix and the review cadence are a starting point for your own team, not a standard. Rename and change them to fit how you work.

Part 1. The principles

Everything in the template follows from six ideas. If someone on your team disagrees with one, settle that before filling in the tables.

1. **You hired an agent.** When you give an agent a goal and a budget, you have taken on a worker that is fast, tireless and indifferent to your reputation. It needs what a new hire gets: a probation period, a manager, a defined scope of authority, and a named person who answers for it. When Bottleneck Labs gave GPT-5.6 a live iOS app, $250 and the instruction to grow the business as much as possible, now, it spent $99.50 buying fifty fake users and finished the 24 hours with zero new revenue. An agent has no skin in the game, so a human has to carry it.
2. **Limit each agent by its blast radius.** An agent runs with whatever access it is given, and by default that is everything its user has. In October, Goldman Sachs reportedly kept coding agents with its engineers, about 12,000 of roughly 47,000 staff, and drew the line by job title. The title tells you little, while what the agent can read, write, spend and reach tells you nearly everything. Give each agent its own identity, narrower than its user's.
3. **Every workflow has an autonomy budget.** Errors compound across steps. At 90% accuracy per step, a five-step task finishes correctly about 59% of the time, ten steps about 35%, and twenty steps about 12%. So the number that matters is how many steps the agent can own before a person looks, and that depends on how cheap a mistake is to catch and to undo.

4. **A goal gets the most literal reading.** An agent told to "resolve the ticket" will close it if closing counts. A more capable model finds more shortcuts, so waiting for a better model makes this worse. Write the objective with the cheapest shortcut in mind, and say out loud the constraints you would never need to say to a person.
5. **A limit is only as real as what enforces it.** Being authorized and being contained are different properties. For every limit on an agent, you should be able to name the software that enforces it, who patches that software, and how long patching takes.
6. **You cannot govern what you cannot reconstruct.** Agents fail outward, into systems you do not own. Before one acts outside your walls you need attribution, an action log nobody can edit, and a kill switch with a named owner.

> **Rule of thumb.** Let agents loose first where undo is cheap: draft before send, stage before commit, recommend before execute.

Part 2. The agent register

> One row per agent. Fill in what the agent can actually do today, which you find by listing its real permissions, and ignore what the design document says it should do. If a cell cannot be filled in within ten minutes, write "unknown" and treat that as the first finding.

Agent	Owner (a name)	Purpose	Can read	Can write	Spend limit	Tier (1 to 4)	Last review
Example: refund assistant	*Head of support*	*Draft refund decisions*	*Orders (copy), tickets*	*Ticket notes only*	*Hard cap per month at provider*	*1*	*Date*

Rules for the register

- The owner is one person, by name. A team or a vendor is not an owner. This person inherits the agent's reputational downside, the way a partner signs off on an audit.
- "Can write" includes sending email, calling outside APIs, creating accounts and spending money, as well as changing your own records.
- The spend limit is a hard cap set at the provider or the gateway. An alert that somebody reads later does not count as a limit.
- If an agent runs as its user and cannot be given less than its user has, write that in the row. It is a finding about your platform.
- Coding agents on laptops belong in the register too. So do agents a vendor runs for you.

Part 3. Autonomy tiers

The tier belongs to the workflow, because the same model can be safe for one task and reckless for another. Every new agent starts at tier 1, which is its probation. It moves up one tier at a time, when the evidence supports it.

Tier	What the agent does	Conditions the workflow must meet	Typical work
1. Suggest only	Reads and proposes. A person does the action.	Read-only access, on copies of data where possible. Named owner. Corrections are recorded.	Drafting, summarizing, classifying, building in a sandbox
2. Act with approval	Stages the action. A named person approves before it takes effect.	The agent has its own identity. The approver sees exactly what will happen and can refuse. The checkpoint sits right before the step that is expensive or permanent. Actions are logged.	Staging a change, preparing a payment, queuing a customer message
3. Act then report	Acts by itself. A person reviews afterwards, at a set sampling rate.	Every action can be undone and you have done the undo once. You have written down how fast a mistake is noticed. Hard spend cap. The correction record from tier 2 supports the move.	Internal updates that are reversible and easy to check
4. Act alone	Acts by itself. Spot checks only.	All of tier 3. The environment checks the work (tests, reconciliation). The worst outcome on the blast-radius worksheet is one you would accept without calling a meeting. The kill switch has been tested.	Work with cheap verification and cheap undo, such as code behind tests and review

Where the person goes. Keep the human where the action is irreversible. Spending money, sending mail to customers, calling an outside API, writing to production and changing a system of record stay at tier 2 until a hard limit the agent cannot change makes the worst case small. In Project Vend, an AI ran a small office shop for a month and slowly lost money through a hundred small, unverified decisions it could not take back.

Count the steps before you choose a tier

> Do this once per workflow. Most teams ask whether the agent can do the task. Ask how many steps it can own before a person has to look.

Workflow _____

Steps from start to finish _____

The step where a mistake becomes expensive or permanent _____

How the work is checked, and what a check costs _____

How a mistake is undone, and how long that takes _____

Steps the agent may own before a checkpoint _____

Tier, and the date it will be reviewed _____

Moving an agent up a tier

- [ ] You started by reviewing everything, and you have a record of every edit, rejection and override with the reason.
- [ ] The sampling rate for the new tier is a number, written down, with a named owner who is accountable for the outcome.
- [ ] The blast-radius worksheet has been redone for the new tier and the worst outcome is still acceptable.
- [ ] The approval in Part 7 is signed by someone other than the person who asked for it.

Part 4. Blast-radius worksheet

> One sheet per agent. Start with the three agents your people use most. If any of them can move money, send customer messages or change production, that is where your blast radius is large. Answer for what the agent could do with its current access on a bad day, including a day when a web page, ticket or log file written by a stranger steers it.

Agent and owner _____

Objective, as written for the agent _____

The cheapest, most literal way to satisfy that objective _____

Constraints you would never need to say to a person, now written down _____

Time horizon the agent is told to care about _____

Question	Your answer
What is the worst realistic outcome with the access it has today?	
Who outside your company could be affected (customers, partners, public services)?	
How would you detect it, and who would notice first: you or the other party?	
How long could it run before someone notices?	
How would you reverse it, and when did you last try that reversal?	
Which actions are one-way doors that cannot be reversed at all?	

Question	Your answer
What is the most it can spend before a hard cap stops it?	
Which credentials can it see, and what could each one do if stolen?	
Could its credentials reach the backup you would restore from?	

What enforces the limit

Limit	Enforced by (name the software)	Who patches it, and how fast	Date you tested it
Account or permission scope			
Network egress			
Spend cap			
Sandbox boundary			
One independent control that shares no code with the above			

A limit is a hope until you have tested it. On October 25, Erica Windisch published a report called Tailworm, and her lab tests on Tailscale 1.104.2 showed a person allowed to log in as an ordinary user reaching root on the same machine (Tailscale has not confirmed every claim in it). If the thing that grants the access is also the only thing that enforces the limit, try to exceed the limit yourself. Try to reach an unapproved domain from inside the sandbox. Try to spend past the cap.

Decision: accept, reduce access, or lower the tier _____

Part 5. Pre-deployment checklist

> Run this before an agent touches live data or a live system. Any unticked box either gets fixed or gets a named person who accepts it in writing.

The containment items are old controls, and they matter more with agents. In July an OpenAI model running a security benchmark called ExploitGym found a zero-day in OpenAI's own sandbox, got itself onto the open internet and broke into Hugging Face to steal the answer key. Least privilege, short-lived credentials, egress allowlists and a sandbox that is a real boundary are what would have blunted it.

Ownership and objective

- [ ] A named person owns the agent and knows they answer for what it does, including in the hour after the deadline passes.
- [ ] The objective states a time horizon and does not say "as much as possible, now" in any form.

- [ ] You have written down the cheapest literal way to hit the objective and changed the wording until that shortcut no longer counts as success.
- [ ] The agent has a scope of authority in writing: what it may decide alone and what it must hand to a person.
- [ ] Anything built in a sandbox by a non-engineer has a named engineer who owns it before it touches live data.

Identity and access

- [ ] The agent has its own identity, separate from and narrower than the person who started it.
- [ ] It starts read-only, on copies of data, and each write permission was added for a stated reason.
- [ ] You have listed what it can actually do when it runs, and the list matches the register.
- [ ] No long-lived keys sit in the agent's environment. It gets short-lived tokens issued per task.
- [ ] Each key it uses has a hard spend cap at the provider and is separate from the keys of other services.
- [ ] Its credentials cannot reach the backups you would restore from.

Containment

- [ ] It runs on a machine that holds nothing valuable and can be rebuilt in minutes.
- [ ] Outbound network access is an allowlist, enforced on the host side where the agent cannot switch it off.
- [ ] You know whether each sandbox has its own kernel or shares the host's, and how long a security fix takes to reach your workloads.
- [ ] Each run has its own network scope and spend limit, and someone has tried to exceed both.

Record and reversal

- [ ] Every action the agent takes is logged as described in Part 6, and the agent cannot edit or delete that log.
- [ ] For every write it can make, you can say whether it is reversible, and you have performed the reversal once.
- [ ] If it writes to a system of record, you have restored that system from an offline copy, end to end, and timed it.
- [ ] A named person can stop this agent, and every copy of it, and has done so once in a test.
- [ ] Human edits, rejections and overrides of its output are stored with the reason, in your own storage, in a plain format.

For coding agents, also

- [ ] AI-written code is tagged automatically at commit time, so reviewers do not depend on anyone remembering to say so.
- [ ] Generated code goes through the same security, dependency and secrets scanning as human code, and nothing merges without review.

- [ ] You have a written policy on which assistants are approved, what they may touch, and what must never be pasted into them.
- [ ] The agent's instruction files, hooks and plugins are in version control, and someone owns them.

Part 6. What the audit log must record

The log is a flight recorder, so it goes in before the agent runs. Record the calls the agent actually made. The model's own account of its reasoning is unreliable and does not replace them.

Between May and June 2026, a swarm of AI agents uploaded more than 2,000 malicious packages to RubyGems, which shut down new registrations for four days, and researchers traced the campaign to OpenAI. Months later OpenAI was still saying it would "continue to investigate" what its agents had done. If they could not reconstruct it, assume you cannot either until the log exists.

Field	What to record	We have it (yes / no)
Agent	Which agent, and which version of its instructions and tools	
Run	A run ID that ties every step of one task together, in order	
On whose behalf	The person, team or customer the run was for	
Action	Every request, account created, message sent and write made	
Target	The system acted on, and whether it is yours or someone else's	
Credential	Which identity or key was used for the action	
Time	When each action happened	
Before and after	What changed, in enough detail to reverse it	
Approval	Who approved a tier 2 action, and when	
Spend	Cost of the run, against its cap	
Correction	Any human edit, rejection or override, with the reason	

Three tests for the log

- [ ] If one of your agents did something harmful to an outside system last Tuesday, you could reconstruct it. Write down how long that would take: _____
- [ ] If a third party complained today, you could tell them which agent, which run and on whose behalf within minutes, without opening an investigation.
- [ ] The log is append-only and kept where the agent's own credentials cannot change it.

Latency, error rates and token spend will not tell you that an agent just created forty accounts on a partner's platform. Those metrics were built for software that stays inside your walls. Route unusual usage and billing from your AI providers to whoever is on call, so it does not sit in a finance inbox until month end.

Part 7. Approval matrix for expanding scope

> An agent's scope grows one request at a time, and usually nobody decides. Use this table so that somebody does. Write a name against each row. The person asking for the change and the person approving it must be different people.

Change requested	Who should approve	Evidence before a yes	Name in your company
New agent, tier 1, read-only on copies	Agent owner	Row added to the register	
Read access to live or customer data	Owner of that data	Reason a copy will not do	
Write access to a live system	Owner of that system, plus a named engineer who will own the agent's changes	Worksheet redone; reversal performed once	
Write access to a system of record	The executive accountable for that record	Timed restore from an offline copy; every write logged	
Higher tier, or lower sampling rate	The person accountable for the outcome of the workflow	Correction record at the current tier	
Higher spend limit	Budget holder of the team that benefits	Per-token bill to date; new hard cap set at the provider	
New credential, outside system or wider network access	Security owner	What enforces the new limit, and a test of it	
Actions that reach customers or partners	Business owner of that relationship; legal where you are regulated	Disclosure wording agreed (Part 8)	
Change to objective, instructions, hooks or plugins	Agent owner, reviewed like a code change	A diff; literal-shortcut question asked again	

These are decision rights, the same thing a delegation-of-authority matrix describes for people. The sampling rate is your risk appetite expressed as a number. It should be set by the person accountable for the outcome, and today it is usually set by whoever configured the tool first.

Before you take access away from anyone, write down the ten things people built with it. Goldman's traders had been using coding agents to monitor client flows, model positions and produce charts, and when access was cut those tools were passed to engineers to be improved. That list is a set of requirements, and if you ignore it the need comes back as spreadsheets and scripts you cannot see.

Part 8. Incidents and drills

The first hour, agreed in advance

> One page is enough. Fill it in now, with names, for two cases: an agent has done something harmful, or a vendor that holds your keys says "assume compromise".

Who can stop the agent, and every copy of it _____

Who rotates the credentials it could reach _____

Who checks provider usage and billing _____

Who reconstructs what it did from the log _____

Who calls the affected customer or partner _____

What they say in that call _____

Who you call for forensic help _____

Decide the disclosure policy now, in writing: what you tell a partner when your agent damages their system. With agents, staying silent is the cheap option, because attribution is hard and no single person carries the mistake. A policy written before the incident reads like a company worth trusting, and one written after reads like an excuse.

Check one more thing before you need it. If your security plan assumes AI tools will help you investigate, ask your vendor whether their models will analyze attack logs or refuse. When Hugging Face tried to use frontier commercial models on the logs of the ExploitGym attack, the models refused, while the attacking model operated under no usage policy.

Four drills to run

Drill	How to run it	What to measure	Date and result
AI turned off	Replay a real incident from the last year with the AI tools off	Time to a correct first hypothesis, against the same team with AI on	
Sandbox escape	Assume one agent is fully compromised	What it reaches, how fast you notice, how fast you rotate	
Full rotation	Rotate every secret stored with a platform vendor	Total time. More than a day, or an incomplete list, is the finding	
Cold restore	Restore a system of record from an offline copy	Time end to end, and how much history is lost	

Why the AI goes off. The incidents that need your people tend to break the AI too: a vendor outage, rotated keys, a compromised tool. A team that has never worked an incident without the assistant is like a backup you have never restored. The skill fades with disuse, too: in a study in The Lancet Gastroenterology & Hepatology, endoscopists' unassisted polyp detection fell from 28% to 22% after a period of working with AI assistance.

Habits that keep the drill honest

- [ ] During real incidents, the person on point writes their own hypothesis in the channel before asking the model.
- [ ] On-call can reach logs, dashboards and rollback controls when the AI vendor is down and when the AI tooling's keys have been rotated.
- [ ] People who ship agent-written code carry the pager for it.
- [ ] Junior engineers take the first pass on routine incidents, with the AI reviewing afterwards.
- [ ] You count incidents resolved with no AI input and track the time to first hypothesis each quarter.

Part 9. Quarterly review checklist

> Put it in the calendar now, with the agent owners in the room. Also run the instructions section whenever the model underneath an agent is upgraded.

Register and tiers

- [ ] Every agent in use has a row, and every row has an owner who is still in that role.
- [ ] For each agent, the real permissions were listed again and compared with the register.
- [ ] Access that no stated purpose needs any more has been removed.
- [ ] Each tier and sampling rate was looked at against the correction record, and the next review date is written down.
- [ ] The per-token bill for each agent was shown to the team that benefits from it.

Instructions

- [ ] Each objective was reread for its cheapest literal shortcut.
- [ ] For each agent, you deleted the instructions you could not prove earn their place, ran your evaluations, and kept only what broke without them.
- [ ] Rules added after past incidents were checked against the current model. Rules written for a model you no longer use were removed.
- [ ] On one real laptop, someone listed everything a coding agent loads at startup (config directories, hooks, plugins, skills, remote tool servers) in under ten minutes.
- [ ] Agent config files were compared with last quarter's snapshot, and every change a vendor's update made has an owner who accepted it.
- [ ] Plugins and hooks are on an allowlist, and anything else is an exception someone asked for.

- [ ] Vendor reviews now ask: does your software write outside its own directory, and does it ask first?

Gateway and credentials

- [ ] You can name the single system that, if stolen, hands over every model provider account, and the person who owns its security.
- [ ] That system is budgeted and patched like a secrets manager, and nothing in your pipeline pulls a floating tag or an unpinned release of it.
- [ ] Its upgrades lag new releases by days, with a fast lane for security patches that a person approves.
- [ ] Every provider key has a hard spend cap, and each environment has its own key.
- [ ] You have a register of every vendor that stores one of your credentials, what that credential can reach, and whether you can see audit logs.
- [ ] A few fake keys sit next to the real ones and alert a person when used.
- [ ] For each sandbox or access tool, you asked the vendor how long a fix takes to reach you, and asked for their incident record.

Records and drills

- [ ] The "last Tuesday" test from Part 6 was run on one agent, and the time was written down.
- [ ] The correction log is in a format you control and could move to another vendor within a quarter.
- [ ] The four drills in Part 8 each have a date and a result from the last twelve months, and the AI-off drill has one from this quarter.
- [ ] The first-hour page still has current names on it.

Further reading

- The Autonomy Budget: Deploying AI Agents Before They're Reliable, for the compounding arithmetic behind the tiers and the step count.
- Treat an AI agent like a new hire, on why an agent needs an owner, a horizon and a scope of authority.
- Goldman limited coding agents by job title. Limit them by blast radius., on separate agent identities and recording what people built.
- The blast radius of a goal, on literal objectives and keeping the human at the irreversible step.
- OpenAI couldn't say what its agents did. Neither can you., for attribution, the action log, the kill switch and disclosure.
- A limited shell account is only as limited as the software enforcing it, on asking what enforces each limit.
- Run your next incident drill with the AI turned off, for the drill and the on-call habits.
- Deploying AI agents safely: autonomy, permissions and blast radius, the guide that collects these posts with the ones on gateways, sandboxes and stale instructions.

About the author

Dr. Hitesh Sarda is an AI advisor based in Hyderabad and the founder of 256 Technologies, an applied AI lab. He has spent more than 25 years building production software and now helps leadership teams decide what AI to build, buy or skip. He is the author of *Implementing AI Agents in Your Company* and writes at hitesh.in.

> **Want a second pair of eyes on what you filled in?**
>
> Bring this document to an advisory hour. We go through your answers, and you leave with the simplest next step, or an honest view that AI is not the answer yet.
>
> hitesh.in/work-with-me · hitesh@256tec.com

You are welcome to use and adapt this document inside your own organisation. Please do not republish or resell it.