

# Answer Key — `.env.example` Scoping

Reference answers · one section per variable

The point is not matching these answers word-for-word. The point is that for every variable you either (a) named a concrete narrower scope and a concrete rotation cadence, or (b) decided it does not belong in agent context at all. "Scope it down" without naming a mechanism is not an answer.

## 1. `GITHUB_TOKEN`

**Recommended scope:** a GitHub fine-grained personal access token (or, preferably, a GitHub App installation token) scoped to a single repository — read contents, and write contents only on a single branch if the agent is allowed to write. No org admin, no workflows, no secrets, no other repos.

**Rotation:** 7-day TTL on the fine-grained token, with per-run rotation preferred where the tooling supports it (mint at job start, revoke at job end).

**Belongs in agent?** Yes, when scoped. A single-repo token is the canonical agent credential for code workflows — just never the org-admin version shown in the example.

## 2. `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`

**Recommended scope:** never long-lived admin keys. Use STS `AssumeRole` with a role whose policy is scoped to specific resources (specific S3 prefixes, specific DynamoDB tables, specific Lambda functions), and attach a session policy that narrows it further for the run.

**Rotation:** 1-hour session credentials (or shorter). STS sessions expire on their own; there is no manual rotation step.

**Belongs in agent?** Only as an assumed-role session with a short TTL. Never static admin keys. The example values ( `AKIAFAKEADMINKEY123` plus a "full admin" secret) are exactly the shape that should never enter an agent's environment.

## 3. `DATABASE_URL`

**Recommended scope:** the `admin:admin@prod-db` URL in the example is wrong on every axis — admin user, admin password, production database. Use a dedicated read-only database role scoped to specific schemas (or specific tables) that the agent actually needs to query, connecting to a non-production replica if one exists.

**Rotation:** rotate per run, minting a fresh database user or a scoped JWT-style connection token at job start and revoking at job end.

**Belongs in agent?** Only as a read-only role, and ideally only against non-prod. A direct admin connection string to a prod database is a credential that ends careers, not a credential you put in `.env`.

#### 4. `SLACK_BOT_TOKEN`

**Recommended scope:** an org-wide `xoxb` bot token is wrong. Create a dedicated Slack app scoped to a single channel (or a single workspace) with narrow OAuth scopes — for example, `chat:write` in one channel and nothing else. No `channels:history` on private channels, no `users:read.email`, no admin scopes.

**Rotation:** quarterly on a calendar, or immediately on any incident. Slack bot tokens are not meant to rotate per run.

**Belongs in agent?** Only if the agent has a legitimate, explicit need to post in a specific channel. If the only use case is "notify me when the run finishes," that is better handled by a webhook URL scoped to one channel, not a bot token.

#### 5. `OPENAI_API_KEY`

**Recommended scope:** the org default key is wrong. Create a per-project key with an explicit spend cap (via the OpenAI dashboard's project feature), scoped to only the model families and endpoints the agent actually uses. A project key with a \$20/month cap cannot drain the org budget even if it leaks.

**Rotation:** monthly, or immediately on any suspicion of compromise.

**Belongs in agent?** Yes, per-project with a spend cap. This is one of the credentials that an AI agent obviously needs; the discipline is making sure it is the scoped version, not the org default.

#### 6. `STRIPE_SECRET_KEY`

**Recommended scope:** `sk_live` in an agent is never correct. For any non-production use, use `sk_test` against Stripe's test mode — no real money, no real customers. For the rare case where a production-facing agent genuinely needs Stripe access, use `sk_restricted` (Stripe's restricted API keys) with a specific permission set — for

example, "read customers" only, or "read charges" only — never full access.

**Rotation:** monthly for restricted keys, immediately on any incident. Test keys can be rotated less often because they carry no financial risk.

**Belongs in agent?** Almost never. If it absolutely must, `sk_restricted` with a minimal permission set. The example value (`sk_live_fake_full_access`) is a credential that under no circumstances should reach an agent's context.

## 7. `JWT_SIGNING_SECRET`

**Recommended scope:** there is no correct scope for this variable in an agent. A JWT signing secret is the key that proves "this token was issued by our service." If an agent has it, the agent can forge tokens for any user. The correct handling is to keep the signing secret inside the service that issues and validates JWTs, and to never export it to any other process — human or agent.

**Rotation:** on a cadence set by your service's key-rotation policy (typically quarterly with overlap windows), and immediately on suspected compromise.

**Belongs in agent? No.** Signing secrets are a service-internal concern. If an agent needs to call an authenticated API, the agent should use its own scoped credentials to obtain short-lived JWTs from the service, not forge them itself.

## Pattern summary

Out of eight variables, zero belong in an agent in their original form. Three (GitHub, OpenAI, maybe Slack) belong in an agent in a properly scoped form. Two (AWS, Database) belong in an agent only through an indirection — assumed-role sessions, read-only roles. Two (Stripe live, JWT signing) almost never belong in an agent at all. The default answer for any credential handed to you as "just put it in the `.env`" should be "which narrower version of this should I mint instead?" — not "ok."