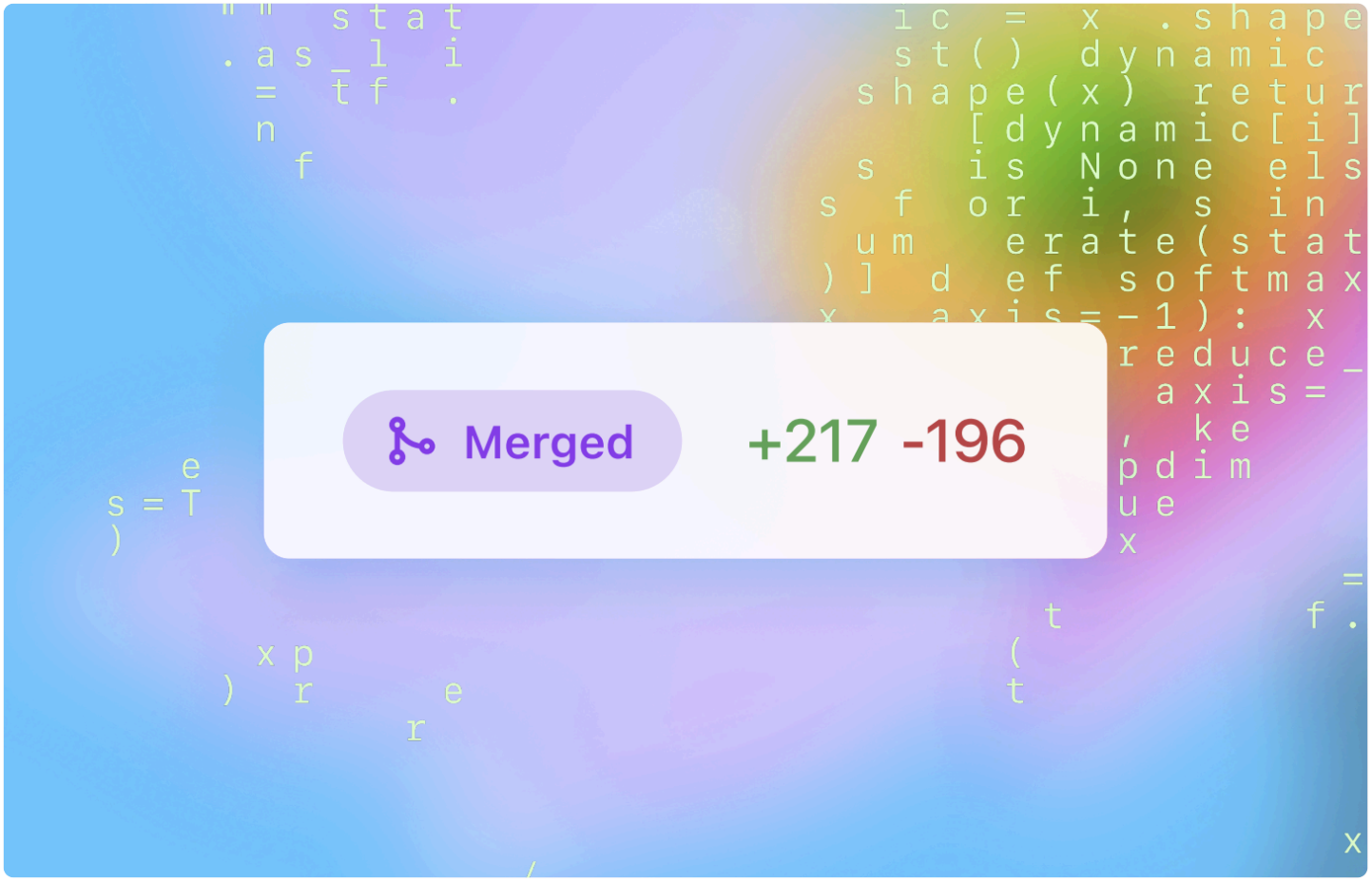
OpenAI

Codex Security Whitepaper



Contents

Overview

Codex is OpenAI's agentic software engineering product. Enterprise customers using Codex maintain ownership and control over their data, and also receive our comprehensive commitment to product security. Developers can use Codex in two environments: locally on their computer and by launching tasks in OpenAI's cloud-hosted environment. This security whitepaper focuses on these two environments separately as they handle data in different ways.

Administrative Controls

Environment Permissions

Codex's granular environment permissions determine which Codex environments are enabled for your organization. Administrators can toggle features on or off in ChatGPT's workspace settings. The following permissions are available:

- **Codex Local:** Administrators can enable or disable signing into Codex CLI, IDE extension, and Codex App with ChatGPT (on-device agent-assist coding tools)
- **Codex Cloud:** Administrators can enable or disable access to Codex Cloud (remote cloud environment)
- **Github Connector:** Administrators can enable or disable access to the Codex Github Connector (used to read and write to Github repositories)

These controls allow you to determine which Codex environments can be used by your team. For example, enterprises may choose to only allow Codex CLI, IDE and App usage.

Role-Based Access Control (RBAC)

Additionally, each permission is supported by ChatGPT's RBAC groups. This allows administrators to provision access to a subset of users for any combination of permissions available. RBAC groups may also be populated by SCIM, allowing permissions to be dictated by your SSO system's user groups.

Admins can also delegate limited Codex administration to a subset of users (for example, viewing workspace analytics and managing Codex environments), aligned with existing workspace RBAC.

Governance and observability

Codex supports three layers of visibility depending on the governance need: the Analytics Dashboard for self-serve adoption and code review impact, the Analytics API for programmatic daily metrics exports, and the Compliance API for detailed audit logs used in investigations and compliance workflows. See https://developers.openai.com/codex/enterprise/governance for more detail.

Managed configuration

Enterprise admins can centrally govern local Codex behavior using two managed layers: admin enforced requirements and managed defaults. Requirements are hard constraints that users cannot override (even via config.toml, CLI flags, profiles, or in session UI), while managed defaults set starting values on launch that users can change during a session but are re applied the next time Codex starts. See https://developers.openai.com/codex/security for more detail.

Using requirements.toml, managed_config.toml, and on macOS, MDM delivered preferences, admins can centrally lock in guardrails such as:

- **Authentication and login policy:** Control which login methods Codex CLI, IDE, and App may use by setting “forced_login_method” to “chatgpt” or “api”, and optionally pin Codex to a specific ChatGPT workspace using “forced_chatgpt_workspace_id”. If the active credentials do not match the managed configuration, Codex signs the user out and exits. This allows security teams to prevent use of personal API keys or personal ChatGPT accounts and ensure all usage flows through the organization’s SSO-protected workspace.
- **Sandboxing and filesystem access:** Set the default sandbox mode (for example “workspace-write” or “read-only”) and constrain writable paths so that Codex can only modify files inside approved project directories. These policies apply consistently across CLI, IDE, and App usage, reducing the risk of unintended edits outside the developer’s workspace.
- **Command-execution approvals:** Configure organization-wide approval policies (such as “untrusted” or “on-request”) so that Codex must obtain explicit user approval before running certain commands, even if a developer attempts to relax these settings locally. Admins can also enforce restrictive command rules via a [rules] table in requirements.toml, where decisions must be “prompt” or “forbidden”, and the most restrictive rule wins when merged with regular .rules files.
- **Network and environment policy:** Define whether Codex may access the network for local runs and which environment variables are forwarded into the sandboxed command environment via “shell_environment_policy”, helping prevent unintentional exposure of secrets or internal endpoints. Admins can also constrain web search mode in requirements.toml, and Codex defaults to cached web results to reduce exposure to prompt injection from arbitrary live content.
- **MCP servers and integrations:** Standardize which MCP servers, tools, and external integrations Codex can talk to by configuring an approved mcp_servers list in requirements.toml. Codex enables a server only when both its name and identity match an approved entry (matching command for stdio servers and url for HTTP servers).
- **Telemetry and observability defaults:** Codex supports opt-in OpenTelemetry (OTel) so teams can audit usage and investigate issues without weakening local security defaults. Telemetry is off by default, and the recommended posture is to keep prompt logging redacted (log_user_prompt = false) and treat tool arguments and outputs as sensitive when routing events to collectors you control.
- **Cloud requirements**: When users sign in with ChatGPT on a Business or Enterprise plan, Codex can fetch admin enforced requirements from the Codex service and apply them across Codex surfaces. This layer fills unset requirement fields, with precedence below macOS MDM managed preferences and above /etc/codex/requirements.toml.

Teams can standardize Codex setup through Team Config to share defaults (config.toml), shared command rules (rules/), and shared Skills (skills/) across developers without per-user duplication.

Quick Reference: Data Handling By Codex Environment Type

Control / Feature	**Codex CLI, IDE, App (local) accessed via sign-in-with-ChatGPT**	**Codex Cloud (remote)**
Admin control	Enable or disable sign in with ChatGPT	Enable/disable Codex Cloud toggle
Training	No training on enterprise data.	No training on enterprise data.
Data retention	Codebase remains in the local development environment. Compliance API audit logs are stored for up to 30 days. No other data is retained.	Codebase cached temporarily on OpenAI's infrastructure. Compliance API audit logs are stored for up to 30 days.
Data residency	Codex CLI, IDE extension, and App run locally on the developer's computer. Execution/inference processed temporarily in the US. Compliance API audit logs are stored regionally in accordance with ChatGPT Enterprise settings.	Persistent conversation data and Compliance API audit logs stored regionally in accordance with ChatGPT Enterprise settings Execution/inference processed temporarily in U.S. Temporary caching only.
Data egress	Codebase remains in the local development environment. Prompts/metadata sent to OpenAI for inference.	Codebase is downloaded into an OpenAI managed container. Prompts/metadata sent to OpenAI for inference.
User Group RBAC	Supported	Supported

Note: when accessing CLI, IDE, App with an API key, the data retention, residency and sharing settings defined by your API organization's administrators will be used. Compliance API audit logs do not cover API usage. API organization owners can choose to opt-in to share API data with OpenAI. This setting is not available to certain organizations, including Enterprise and customers with Zero Data Retention enabled.

Codex CLI, IDE Extension, and Codex App

Codex CLI, IDE extension, and Codex App run locally on the developer's computer. The IDE extension is built for any VS Code-based editor and runs on top of the CLI's agent harness. The Codex App is a standalone application on the user's computer. When developers use these tools, requests are sent to OpenAI so that our models can provide responses with code completions. Data sent to OpenAI may include user prompts, code

snippets, and other file data that the model accesses to conduct the coding task. Only data in scope for the Codex Compliance API audit logs are retained for up to 30 days and stored in accordance with ChatGPT data residency settings.

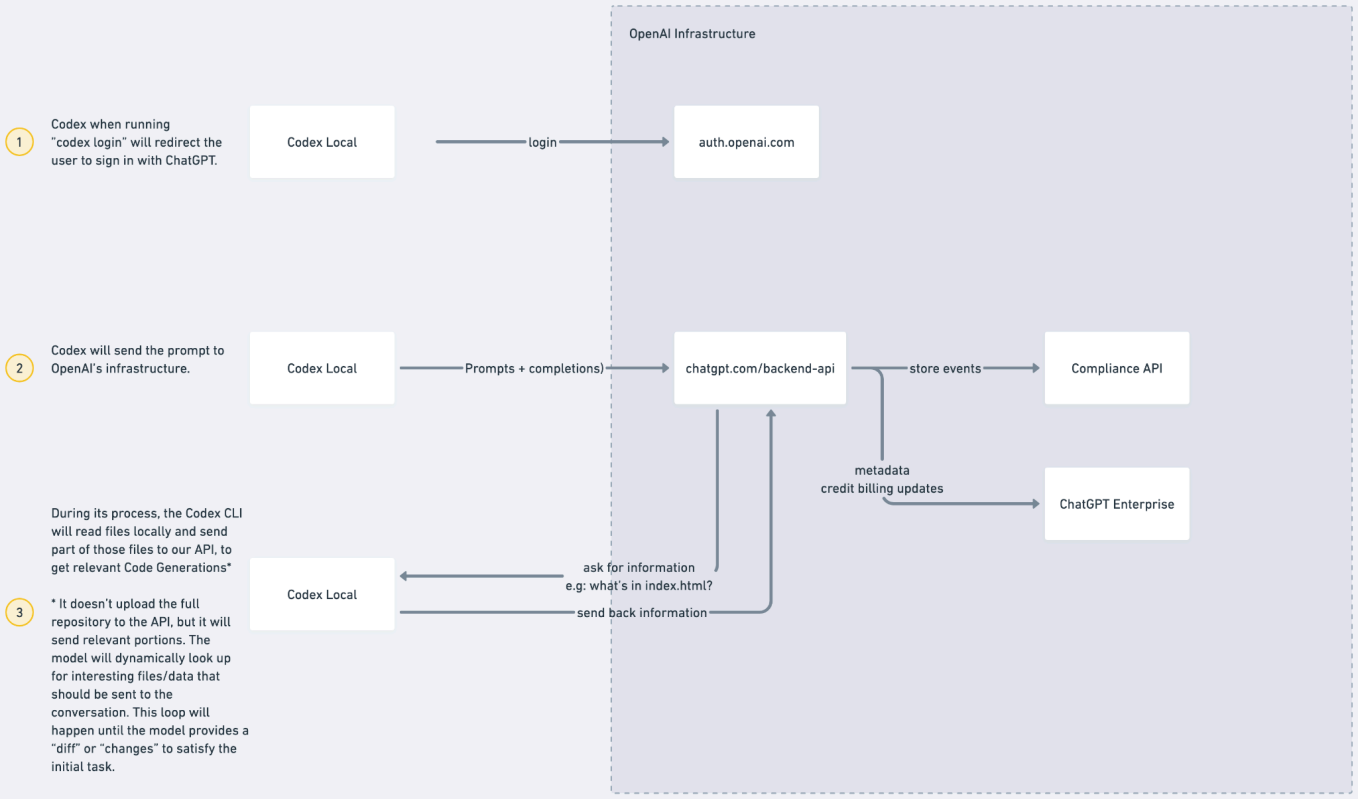
Login and Authorization

Codex CLI accepts two methods of authentication which determine how model responses may be stored: sign-in-with-chat and API key authentication.

- **Sign in with ChatGPT:** model responses are handled by OpenAI's dedicated endpoint for Codex. The primary use-case for this authentication method is to allow developers to sign into Codex CLI, IDE extension, and Codex App with their ChatGPT account. Data for the Compliance API audit logs is retained in accordance with your ChatGPT settings.
- **API key authentication:** model responses are handled by the OpenAI API Organization that you configure. The primary use-case for this authentication method is for programmatic execution of Codex CLI (e.g. in CI/CD pipelines). Data is retained in accordance with the settings in your API Organization. Only API Organization members may create API keys.

Sign in with ChatGPT is the default authentication method for Codex CLI. When a user starts Codex without a valid session, they are guided through the sign-in-with-chat OAuth flow, and follow the same login policy that is defined for your workspace users. Permissions applied via workspace settings or RBAC are enforced based on user entitlements.

Codex CLI, IDE Extension, and App login and data flow diagram using Sign in with ChatGPT authentication.



When a user successfully completes the login flow, their credentials are stored locally in their user profile (`~/.codex/auth.json`). Access token lifetime is 90 days, but the token is refreshed before expiration as usage occurs. For each request, we validate the signed-in identity, entitlements, and available credits before execution.

Data Storage

Codex CLI, IDE extension, and Codex App store data in the following ways:

Completions data storage: For developers using Sign in with ChatGPT, data is never trained on and is only stored in our Compliance API infrastructure. For developers signing in with an API key, data is retained in accordance with your API Organization's settings. Compliance API exports detailed activity logs including prompt text, and generated outputs (timestamp, model, token usage).

Local log storage: By default, Codex writes logs of all session history to a directory on the developer's computer. This data includes all prompts sent to OpenAI, as well as all interactions with our models. Retaining this data on the developer's laptop allows for session resume functionality. Local transcript retention can be limited or disabled by configuration if you do not want Codex to save session transcripts under CODEX_HOME.

Billing metadata storage: While prompts and completions are never stored, OpenAI logs credit usage in order to provide accurate spending information. For developers logged in with sign-in-with-chat, this is reflected in the ChatGPT Enterprise usage dashboard. For API key authentication, this is shown in the API Platform dashboard.

Cloud Push from the CLI, IDE, and App

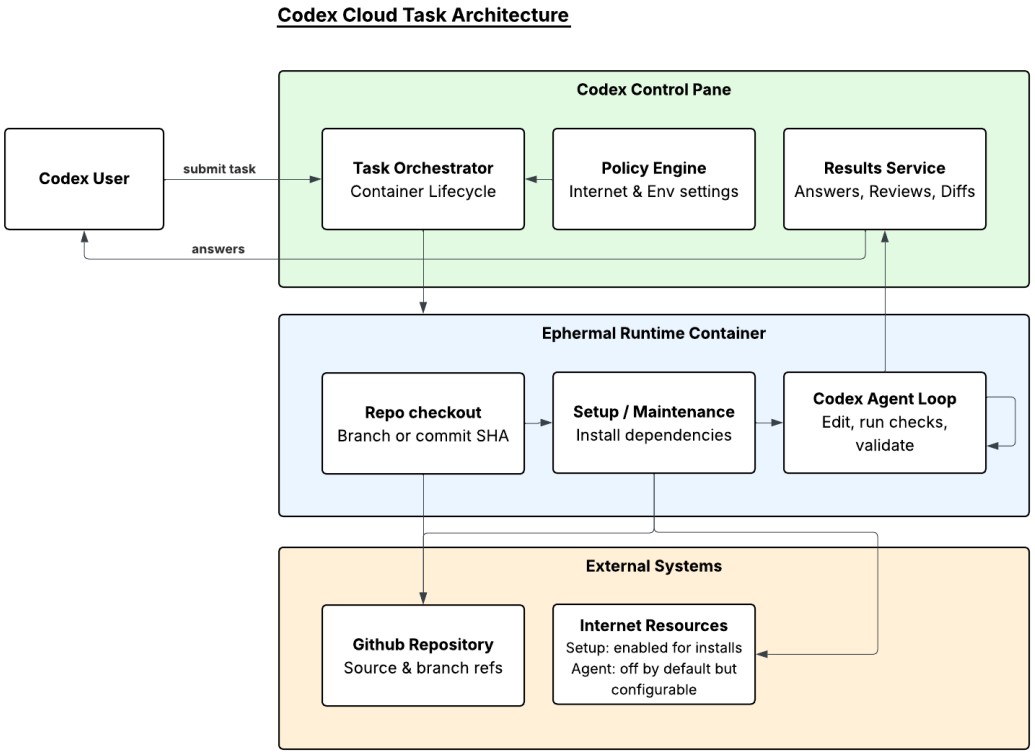
For users that are authorized to use Codex Cloud, all Codex local surfaces allow cloud tasks to be created by selecting the remote environment. The task runs in an ephemeral container following the Codex Cloud model (two-phase networking, run phase offline by default, secrets removed before execution).

Programmatic CLI Use

The Codex CLI can also be used in a programmatic mode that allows it to run in CI/CD use-cases. We recommend using an API key for authorization in this mode. **We strongly recommend against exposing programmatic use in untrusted environments** (e.g. a public code repository where Codex can be run by any user) without thorough security review and red-teaming.

Codex Cloud

Codex Cloud lets the agent execute tasks in the cloud and submit pull requests to your repositories. It runs in isolated containers and integrates with your organization's controls for identity, repo access, and retention.



Customers can connect Codex to their existing GitHub Account to grant access to their public and private repositories. Organizations that require approvals to connect GitHub Apps can limit Codex access to only a set of approved repositories. Codex can then propose code edits based on user requests, iterate on feedback, and submit pull requests.

Codex Cloud comes with the following commitments to user data security and privacy:

- Encryption: All customer data is encrypted at rest (AES-256) and in transit (TLS 1.2+ wherever available)
- Restricted Access: Authorized OpenAI employees will only ever access your content for the purposes of resolving incidents, recovering end user conversations with your explicit permission, or where required by applicable law.
- No Training: We do not train our models on the content you provide to us (e.g., inputs or outputs of our services). Your code will not be used to improve our models
- Data Segregation: OpenAI hosts data in a multi-tenant environment, and tenancy is enforced at our storage layers based on user identity and workspace membership
- Data Retention: Content follows ChatGPT Enterprise retention policies (see trust.openai.com)

Execution Environment

Codex Infrastructure

Codex runs workloads on OpenAI's CaaS with Kubernetes orchestration and a secure container runtime providing sandboxing and enforced maximum lifetimes. Nodes are monitored and regularly replaced as defense-in-depth. Tasks use encrypted transport and data-at-rest.

Network

Networking follows a two-phase model: environment setup may access the internet to install dependencies; execution runs offline by default. Ingress is protected at the internet edge with WAF; all egress flows through a Layer-7 proxy with logging, rate limits, and enforced deny-lists, plus external secure DNS for domain-level blocking. Agent Internet Access can be enabled by admins.

See the Agent Architecture diagram at the end for a visual of this setup vs. run phase and how artifacts return through the GitHub API.

Secrets

Secrets are stored encrypted under customer-specific keyrings; the root keyring is in a hardened KMS and accessible only to the service using a service identity. Decryption requires a cryptographic assertion of the logged-in user and proof of access. During task setup, secrets are used and then removed before execution. Secrets are never written to container logs or PR diffs; access attempts are denied once the execution phase begins.

Data & Control

All traffic enters via a secure edge with WAF; internal orchestration/storage uses private networking with TLS in transit and AES-256 at rest. On completion, Codex pushes a branch and creates a PR via the GitHub API. Diffs and artifacts follow ChatGPT Enterprise retention settings controlled by workspace admins.

Identity

User identity is asserted via existing login/enterprise SSO; internal services require that cryptographic assertion for any access to secrets or secure content.

GitHub Connector and ChatGPT Codex Connector

The ChatGPT GitHub Connector is a read-only integration that lets ChatGPT answer questions using repository contents and metadata. It connects via a GitHub App installed by admins, mirrors live GitHub permissions, and enforces immediate revocation on uninstall. Current usage is read-only.

The Codex GitHub Connector requires write privileges because it performs tasks such as creating branches, committing code, opening and editing pull requests, labeling issues, and posting comments. Codex mints short-lived installation tokens (e.g. pull_requests:write, metadata:read) to carry out these actions while still enforcing GitHub's real-time access controls.

The connector is installed and scoped as a standard GitHub App. An organization owner or admin must approve the installation and can restrict it to specific organizations and repositories. Codex can only read from or write to repositories that the app is installed on and where the requesting user already has access; it does not bypass or weaken GitHub's existing RBAC or branch protection rules.

For each operation, Codex obtains a short-lived, least-privilege installation token scoped to the minimum permissions required. Tokens are stored encrypted, rotated automatically, and become unusable if the app is uninstalled or if repository access is revoked. All access is performed over TLS and subject to the same network and egress controls described elsewhere in this document.

Actions taken through the Codex GitHub Connector are auditable. Changes appear in GitHub as commits, comments, and pull-request updates authored by the Codex GitHub App, and they are also reflected in GitHub's native audit logs. Within ChatGPT Enterprise, Codex additionally records high-level job metadata (such as timestamp, repository, and operation type) so security teams can trace which user requested which Codex-initiated changes.

If your organization uses IP allowlisting for GitHub, configure allow lists using both the ChatGPT egress IP ranges and Codex container egress range, so admins should periodically re-check and update allow lists.

Integrations

Codex Cloud integrates with other tools like Slack, and other third-party apps. When Codex is invoked through mentioned @Codex, the relevant content (for example, the message, thread history, or file metadata) is sent to Codex so it can understand the request and create or update a task.

All such data is handled in accordance with OpenAI's Privacy Policy, Terms of Use, and other applicable enterprise agreements, and access is constrained by the scopes and RBAC of the underlying platform.

Each user needs to individually connect and authorize Codex to apps from the Codex Connectors page.

For Slack, admins can further restrict data sharing by disabling Codex's ability to post full answers back into Slack. When this setting is turned off, Codex only posts a link to the task in Codex Cloud, keeping potentially sensitive environment details out of Slack while still allowing users to monitor task status.

Shared Responsibility

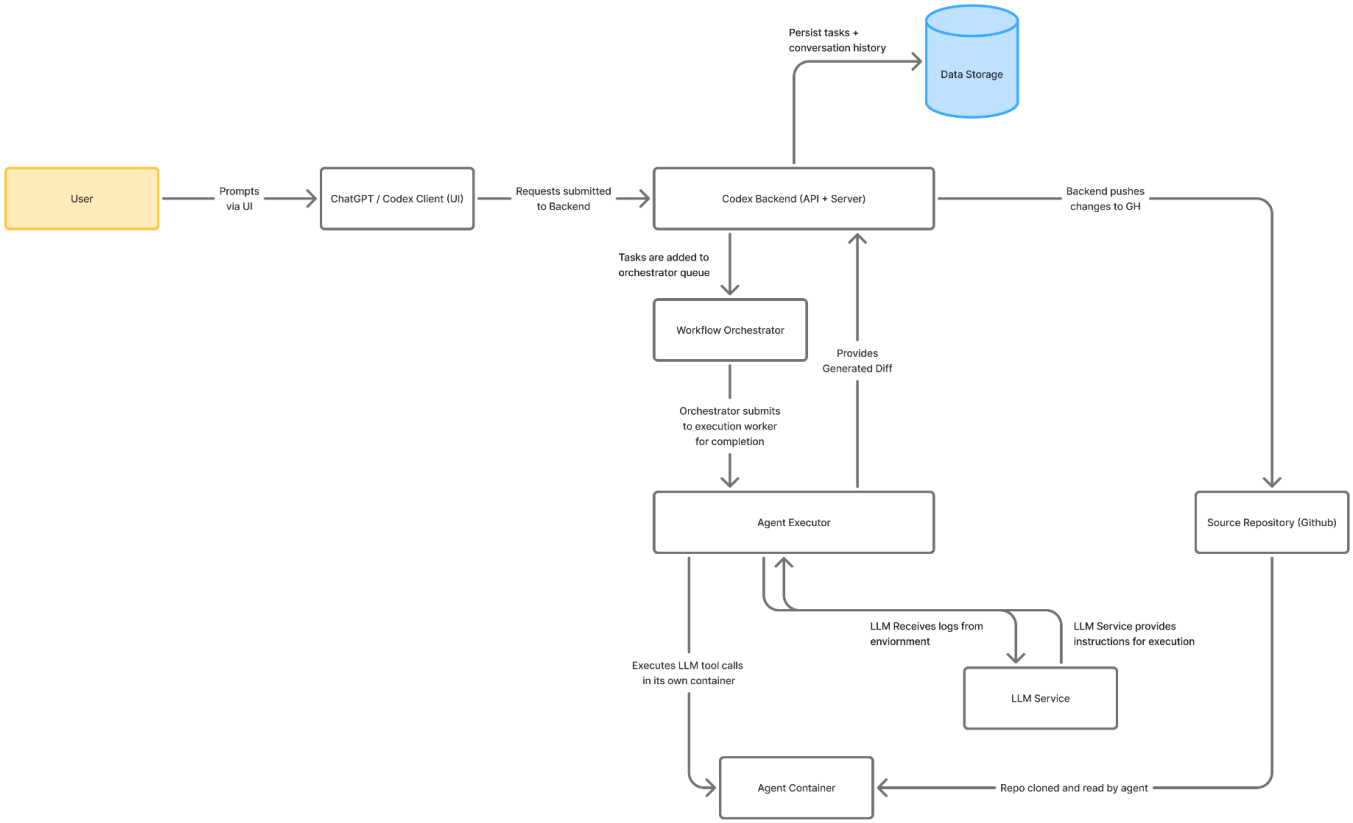
Codex security is a partnership between OpenAI and the enterprise. The Agent Architecture diagram below illustrates these trust boundaries—user → orchestrator → executor → container, with identity assertion and secrets injected only during setup, followed by offline execution.

OpenAI provides the core protections: sandboxed execution, encryption, hardened networking, identity enforcement, and secret handling. The tables below define who does what, the default posture, and where customers typically add controls. Use them as your acceptance checklist.

Area	**OpenAI**	**Customer**
Identity	Enterprise SSO; attribute requests to user	Enforce SSO/conditional access
Networking	Workspace toggle for Agent internet access	If enabling Internet Access, maintain allow-lists per your risk posture
Secrets & tokens	Short-lived tokens; no secret logging	Avoid embedding secrets in images. Rotate, scope, and never print to logs
Admin controls	Workspace toggle for CLI/IDE/App sign-in and Codex Cloud	Decide if Codex Cloud usage is allowed

Learn more about the Codex product suite at https://developers.openai.com/codex and about security and compliance at trust.openai.com. Contact your Account Director (AD) to learn more.

Codex Cloud Architecture



The diagram illustrates the user → orchestrator → executor → container flow. During the setup phase, identity is asserted and any approved secrets are injected. In the execution phase, the container runs offline by default, with no inbound access, and secrets are removed. Once execution is complete, Codex returns artifacts (e.g. a pull request) through the GitHub API and then destroys the container.