One platform for your organization's internal applications

Users submit requests inside the application. Developers and an AI build agent implement them with a preview environment. Changes are reviewed, merged and deployed on your infrastructure.

23	84	3	AGPL-3.0
infrastructure layers documented	alternative components: built-in or integration options	cloud providers deployed end to end: AWS, Azure, DigitalOcean	open source license for the platform and the forge

Request workflow

The request and the change that resolves it stay linked in one system. The forge implements the complete workflow today; a platform deployment runs it from the issue onward.

01

Request

A user submits a request from inside the application, signed in or anonymously.

02

Issue

The request becomes an issue in the application's repository, labeled with its source.

03

Change

A developer or the AI build agent opens a pull request. Each pull request gets a preview environment running the changed application.

04

Checks and review

Automated checks run on the change and again on the merged result. An AI reviewer and a person review it.

05

Merge and deploy

The change is merged under the repository's merge rules and deployed. A failed deployment leaves the previous version running.

06

Status

The user who submitted the request sees its status in the application, through to the release that contains it.

Application template

Most internal application code is business logic. The application template provides authentication, authorization, data access, deployment, backups, monitoring and audit logging. Settled rules become configuration that changes through review. The platform team maintains the template, and an improvement reaches every application built on it.

Governance

- **Existing systems stay in place.** Each infrastructure layer uses a built-in option or an integration with your existing system.
- **Configurable merge policy.** Review and passing checks can be required before merge. Whether AI-authored changes require a person to merge is a setting.
- **Controlled AI usage.** The AI build agent uses the model provider or gateway you configure. The application template contains no AI code.
- **MCP access.** AI assistants authenticate with OAuth as a user and receive that user's permissions. The platform logs every MCP tool call.
- **Open source.** Licensed under AGPL-3.0.

Engagement

An engagement consists of a platform deployment and a forge for each developer who needs one. Before the design review, we complete the bill of materials for your environment. EspoDev deploys Open Platform, implements the selected integrations and provides support.

Bill of materials

23 layers and 84 alternatives. Integration points and source references for every layer are at www.open-platform.sh/bill-of-materials.

Built-in options Selected through an existing setting, mode or deployment module.

Integration options Require integration work. EspoDev implements them at the documented integration point.

LAYER	DEFAULT	BUILT-IN OPTIONS	INTEGRATION OPTIONS
INFRASTRUCTURE			
Location and provisioning	Your own servers	AWS, Azure, DigitalOcean	Google Cloud, VMware vSphere, Your existing network
Kubernetes	k3s	AKS	RKE2, EKS, GKE, OpenShift
Node operating system	Ubuntu 24.04 LTS	The AKS default node image	Red Hat Enterprise Linux, Your hardened image
Pod networking (CNI)	Flannel	Azure CNI Overlay powered by Cilium	Azure CNI Pod Subnet, Cilium, Calico
Ingress and edge	Traefik	The node's public address (VMs), Azure load balancer (AKS), Cloudflare Tunnel	Azure Front Door, in front, Azure Application Gateway, in front, NGINX, F5
DNS and certificates	cert-manager	Azure DNS	Your enterprise CA, Amazon Route 53
IDENTITY AND ACCESS			
Sign-in (identity provider)	Built-in accounts	Microsoft Entra ID, Any OpenID Connect provider: Okta, Ping, Keycloak, ADFS	Your directory as the only identity source
Roles and access	Teams and repository permissions	—	Groups from your directory
Secrets	SOPS with age keys	—	Azure Key Vault, AWS KMS, HashiCorp Vault
SOURCE AND DELIVERY			
Git hosting	Forgejo	—	GitHub Enterprise, Azure DevOps, GitLab, Mirror to or from any git remote
Continuous integration	Actions-compatible runners	GitHub Actions workflow syntax	Azure Pipelines, GitLab CI, Jenkins
Container registry	Built-in OCI registry	Azure Container Registry, for platform images (AKS)	Azure Container Registry, for app images, Amazon ECR, Harbor, JFrog Artifactory
Deployment (GitOps)	Flux	—	Argo CD
Policy and admission	Kyverno	—	Azure Policy for AKS, OPA Gatekeeper

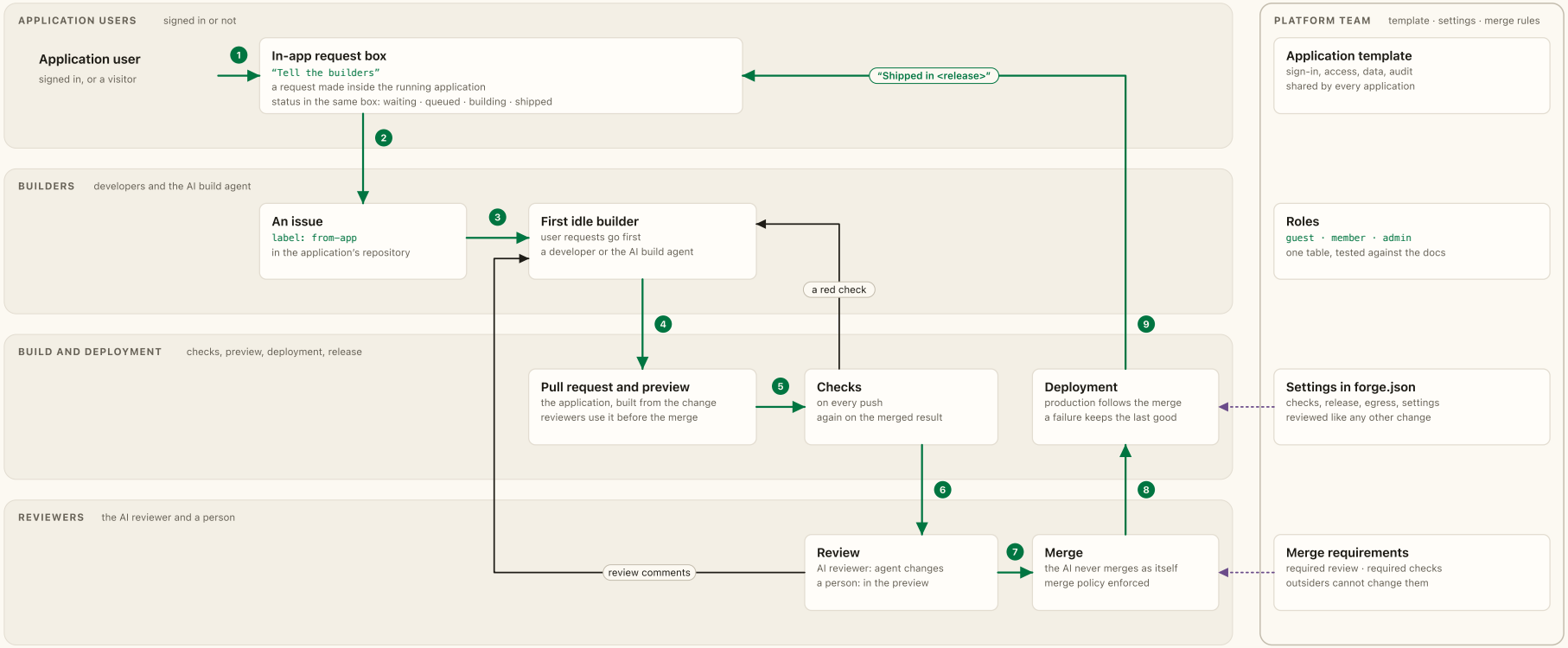
LAYER	DEFAULT	BUILT-IN OPTIONS	INTEGRATION OPTIONS
DATA			
App database	PostgreSQL	—	Azure Database for PostgreSQL, Amazon RDS for PostgreSQL, Your managed PostgreSQL, Microsoft SQL Server
Object and file storage	MinIO (S3-compatible)	Any S3-compatible store, for backups	Amazon S3, Azure Blob Storage, through an S3 gateway
Backups and recovery	Nightly app database dumps, off-site copy, weekly restore drill	Your S3-compatible off-site target	Veeam Kasten, Azure Backup for AKS, alongside
OPERATIONS			
Monitoring and dashboards	Prometheus and Grafana	Azure Monitor managed Prometheus and Container Insights (AKS)	Datadog, Splunk, Dynatrace, New Relic
Alerts and email	Push notifications through a relay	—	Microsoft Teams, Slack, PagerDuty, SMTP or Exchange
Audit trail	Git history, reviews, and a log line per MCP call	—	Microsoft Sentinel, Splunk
AI			
AI model provider	A commercial model provider, or a model gateway	OpenRouter, per organization	Your private model gateway, Azure AI Foundry, Amazon Bedrock, Google Vertex AI, Open models on your hardware, through a gateway
AI assistants and agents (MCP clients)	Remote MCP with OAuth	Assistants that support remote MCP with OAuth, Your own agents	—
APPS			
App template	TypeScript: Bun, Fastify, React	—	Your organization's template, .NET services, Java services, Python services

Request workflow

Lanes show who acts. Each step is a code path in the forge; a platform deployment runs the same steps from the issue onward.

Open Platform — request workflow

A request made inside an application becomes a reviewed, deployed change, and the requester sees its status in the same place. The diagram shows the forge, which runs the complete workflow; a platform deployment runs it from the issue on.



Request workflow: one request, from the application back to it · Returned to the builder · Configuration the platform team maintains · Required role · Optional

1. a user writes in the application's "Tell the builders" box, signed in or not
2. the request becomes an issue in the application's repository, labelled from-app, with the page and environment attached
3. a signed-in user's request goes to the first idle builder, ahead of the AI build agent's own queue; a visitor's request waits until a repository writer adopts it
4. the builder opens a pull request; the forge serves a live preview of the application, built from the change
5. checks run on every push, and again on the pull request merged with its base before the merge
6. the AI reviewer reads every pull request the AI build agent writes (a second reviewer reads changes to security paths); a person comments and follows guided review steps in the preview
7. the merge follows the repository's merge policy: the AI never merges as itself; a repository can require an approving review and passing checks, enforced inside the merge; in an autonomous repository, a green, approved pull request merges in the name of the person who turned autonomy on
8. the merge deploys production; a failed deployment keeps the last good version serving, and a direct push runs the tests before the switch
9. the requester sees "Shipped in <release>" in the same box

Lanes show who acts; the numbers follow one request. Each box and arrow is traced to the code in docs/architecture/README.md.

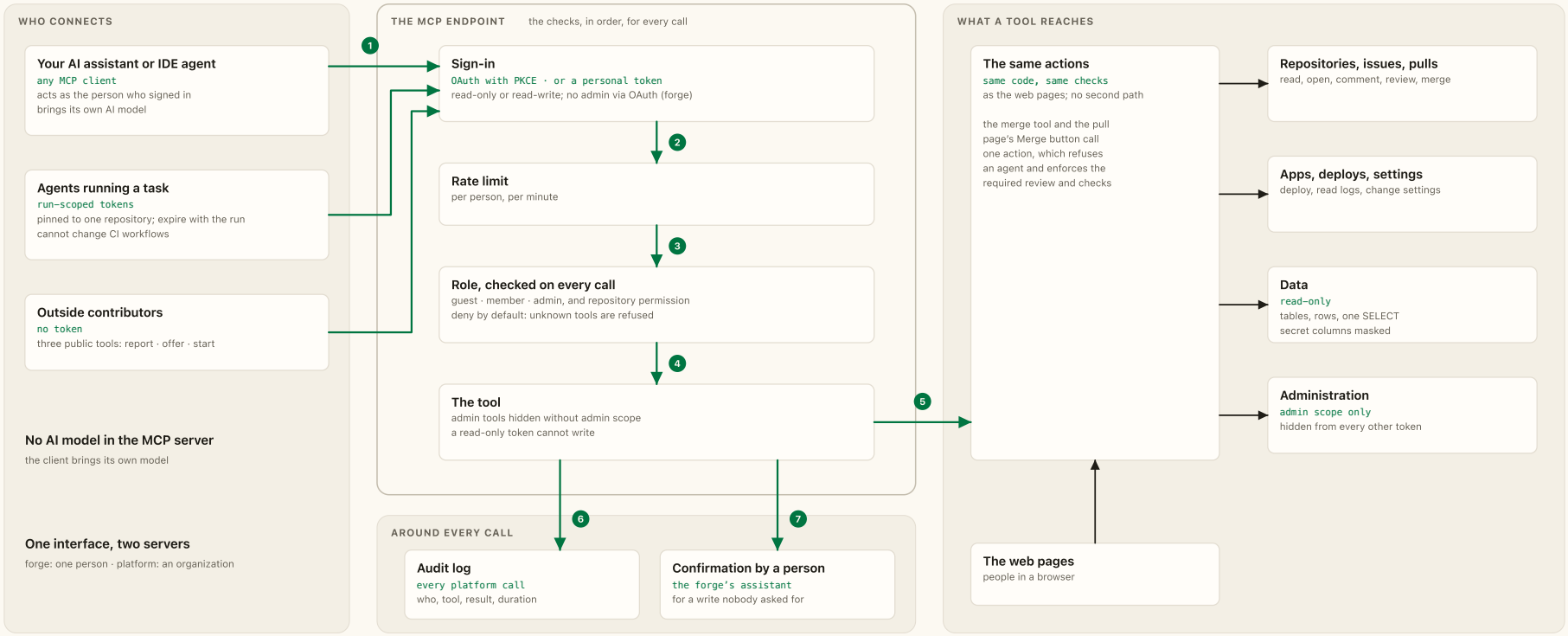
www.open-platform.sh · request workflow · 2026-11

MCP request handling

Who connects, the checks each call passes in order, and what a tool can reach.

Open Platform — MCP request handling

An AI assistant, an agent running a task and an outside contributor reach Open Platform through one Model Context Protocol (MCP) endpoint and the same checks. The MCP server calls no AI model; the client brings its own.



A call, through the checks in order · The same actions, from MCP or the web pages · Required role · Optional

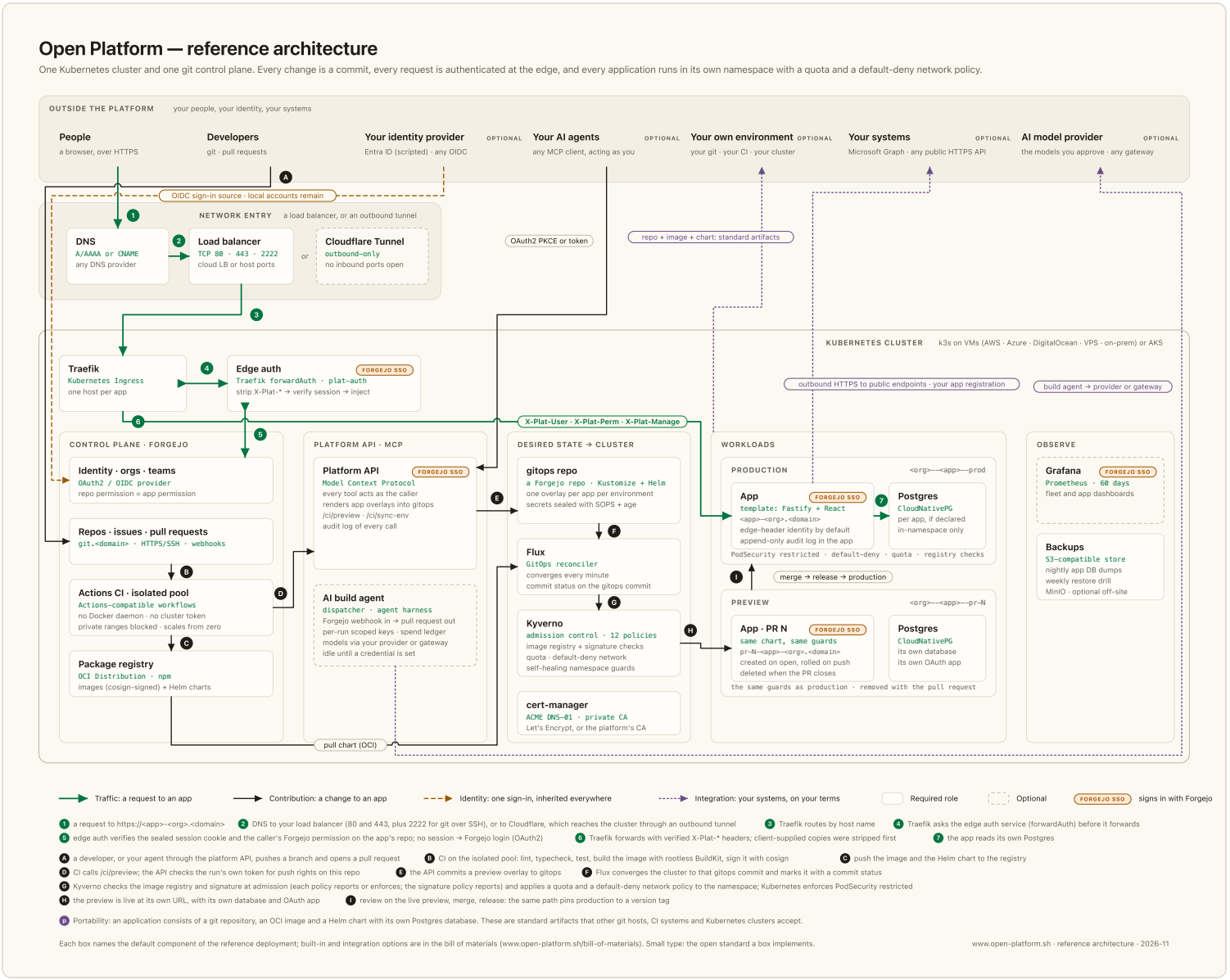
1. sign-in with OAuth and PKCE, or a personal token; the person chooses read-only or read-write; on the forge an OAuth grant never carries admin rights; on the platform an administrator can cut off a person's MCP access at once
2. a rate limit per person, applied before any tool runs
3. the caller's role, and on the platform their repository permission, is read on every call; anything it does not allow is refused, and an unknown tool is denied
4. admin tools are hidden from a token without admin scope; a read-only token cannot write
5. the tool calls the same action the web page calls, with the same checks
6. the platform writes every call to its audit log: who, which tool, what result, how long
7. on the forge, a write by its own assistant that follows outside text, and that the person did not ask for, waits for the person's confirmation

The forge (a single-user workspace) and the platform (an organization's deployment) each serve this interface; they are separate servers. Each box and arrow is traced to the code in docs/architecture/README.md.

www.open-platform.sh · MCP request handling · 2026-11

Reference architecture

Each component is the default for its layer. www.open-platform.sh/bill-of-materials lists the alternatives.



www.open-platform.sh · Deployed, integrated and supported by EspoDev · www.espodev.com