

AIS Workspace

Technische Architektur und Betrieb

Philipp Elhaus · November 2026

Auftragssteuerung, Ausführungsumgebungen und kontrollierte Veröffentlichung für projektbezogene Softwarearbeit mit Agenten.

Inhaltsverzeichnis

1 Systemarchitektur 3

2 API, Identität und Berechtigungen 4

3 Zustand, Daten und Suche 5

4 Agentenlaufzeiten 6

5 Orchestrierung und Wiederaufnahme 7

6 Coding, Testing und Artefaktfreigabe 8

7 Auslieferung in Projektumgebungen 9

8 Plattformrelease 10

9 Betriebsüberwachung und Störungsfälle 11

Quellen und Abbildungen 12

1 Systemarchitektur

Die Workspace-API steuert Aufträge für Browser und API-Clients und führt die Werkzeuge der Prompt-Agenten aus. Coding und Testing laufen in isolierten Foundry-Sitzungen. Freigegebene Änderungen gelangen über Publisher und Pipeline in die Zielumgebung. [1] [7]

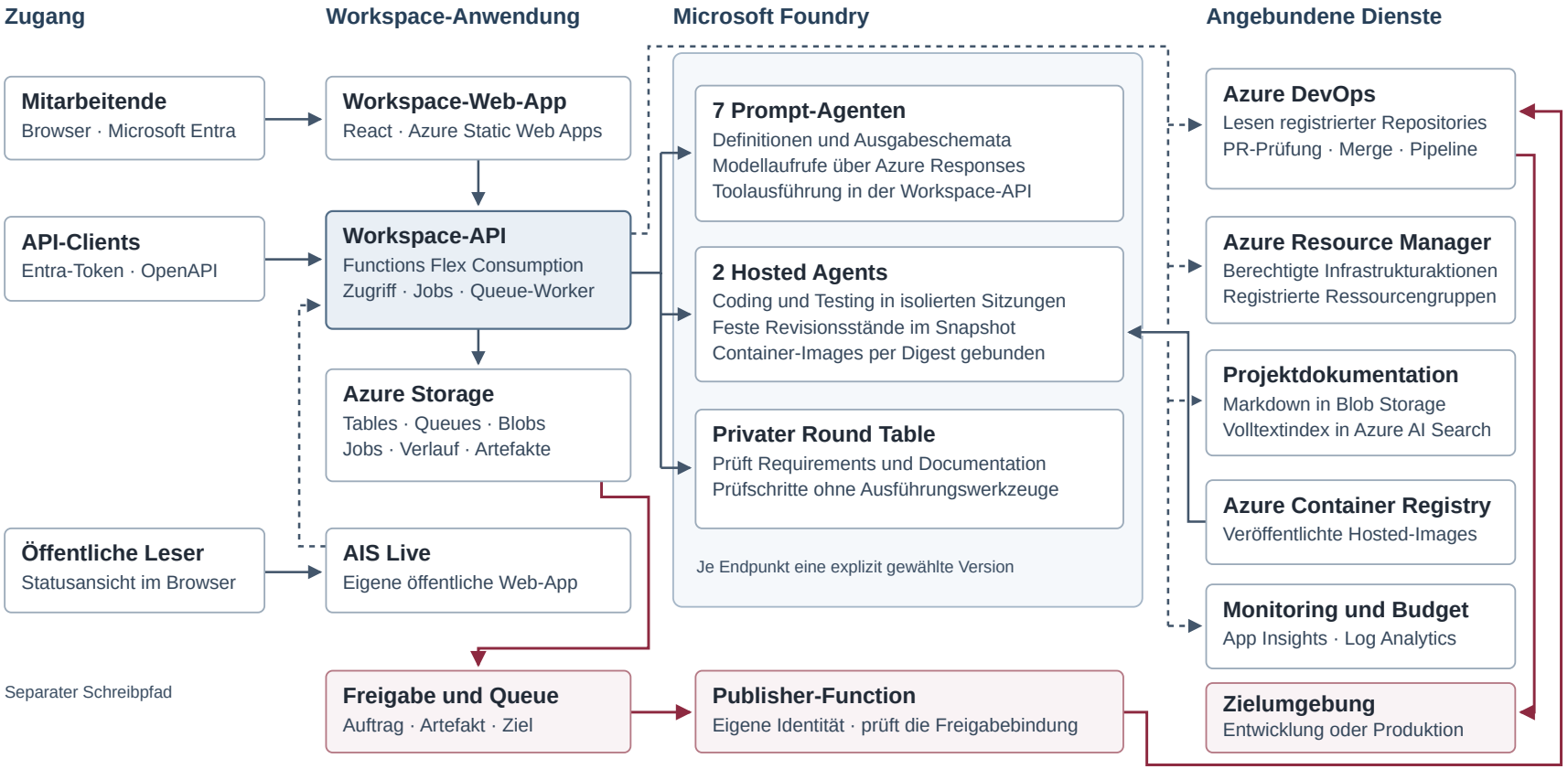


Abb. 1 · Systemarchitektur von AIS Workspace. [1] [3] [5] [7]

Durchgezogene Linien zeigen Aufrufe und Image-Bereitstellung. Gestrichelte Linien markieren Statusabfrage, Werkzeuge und Telemetrie. Rot ist der Auslieferungspfad bis zur Zielumgebung.

2 API, Identität und Berechtigungen

Anmeldung und Projektzugriff

Das React-Frontend läuft als Azure Static Web App. Die Node-API auf Azure Functions Flex Consumption nimmt Browser- und API-Aufrufe entgegen. App Service Authentication prüft das Entra-Token. Die Anwendung verifiziert Aussteller, Zielgruppe und Identitätsbindung für die zugelassenen Mandanten. Bestehende Konten und nachweislich verknüpfte Unternehmenskonten verwenden die direkte Workspace-Rolle an der vorgesehenen Ressourcengruppe. Separat zugelassene Unternehmenskonten verwenden eine private Anwendungsfreigabe. [1] [2] [3] [10]

Private Aufträge gehören einem kanonischen Eigentümer. Zusätzlich gelten seine Projekt- und Agentenrechte. Bei Annahme hält die API die registrierten Repositories und den erlaubten Umfang fest. Worker und Publisher prüfen aktuelle Berechtigungen vor den jeweils vorgesehenen Operationen erneut. [3] [6] [7]

Workload-Identitäten in Produktion

Komponente	Aufgabenbereich
API und Worker	Storage, Aufrufe an Foundry und Azure-DevOps-Lesezugriff für registrierte Repositories.
Infrastructure, DevOps, Documentation	Getrennte Identitäten für freigegebene ARM-Aktionen, begrenzte Boards- und Pipeline-Operationen beziehungsweise die Projektdokumentation in Blob Storage und Search.
Hosted Coding und Testing	Projektcode aus bereitgestellten Snapshots und Werkzeuge im isolierten Container.
Publisher	Eigenständige Prüfung und begrenzte Azure-DevOps-Schreibrechte für **ai/***-Branches und Pull Requests.

AIS Live hat eine eigene Web-App und Pipeline. Es bezieht eine reduzierte, öffentliche Statusprojektion über **GET /api/live**. Der öffentliche Funktionskatalog ist davon getrennt. [1] [2] [3]

3 Zustand, Daten und Suche

Die API bestätigt eine neue Agentenaufgabe mit einer Auftrags-ID. Der Client ruft den Zustand über die jeweilige Job-Route ab. Die persistierten Auftragsdaten verbinden den Status mit Eigentümer, Projekt, Anfrage und angezeigtem Ergebnis. Die Queue stößt die Verarbeitung an. [2] [4] [6]

Datenablage

Dienst	Verwendung
Azure Tables	Aufträge, Eigentümerpartitionen, Arbeitsreihenfolge, Sperren und Projektkatalog. ETags sichern bedingte Änderungen an den Datensätzen.
Azure Queues	Arbeitsaufträge für Agenten, Werkzeuge und Publisher. Der Worker beansprucht die aktuelle Auftragsversion. Für bestimmte Queues werden ausgeschöpfte Wiederholungen gesondert abgelegt.
Azure Blobs	Revisionsgebundene Anweisungen, projektbezogenes Markdown, Vorschläge und davon getrennte Checkpoints. Die Publisher-Function betreibt ihren Host-Zustand in eigenem Storage.
Azure AI Search	Volltextindex je Projekt. Die Suche nutzt den Index, der Dokumentinhalt stammt aus dem aktuellen Markdown-Blob.

Datenabgrenzung zwischen Umgebungen

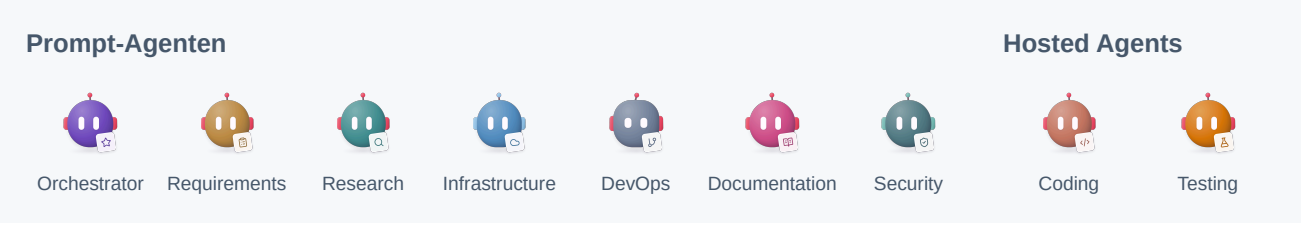
Produktions- und Entwicklungsdaten liegen in getrennten Storage-Konten. Der Server setzt für die Entwicklung eigene Markdown-Pfade und Indexpräfixe durch. Azure Search wird auf Dienstebene berechtigt. Der gemeinsam geführte Projektkatalog ist eine gezielte Ausnahme mit getrennt vergebenen Rollen für API, Entwicklungszugriff und Publisher. [1] [3] [4]

Storage verwendet Entra-Datenzugriff bei deaktiviertem Shared Key und öffentlichem Blobzugriff. Aufträge, Ergebnisse und Artefakte sind eigentümergebunden. Projektdokumentation steht den berechtigten Projektmitgliedern zur Verfügung. [3] [4]

4 Agentenlaufzeiten

Prompt-Agenten

Die API prüft die ausgewählte Foundry-Definition gegen die mit der Anwendung bereitgestellte Fassung. Sie übergibt Anweisungen, Werkzeugschemata und Ausgabeformate an Azure Responses. Werkzeugaufrufe werden in der Anwendung ausgeführt und gegen die Rechte des Agenten und des Projekts geprüft. Der Security Agent liest freigegebenen Repository-Code und gibt beratende Befunde zurück. [1] [5]



Hosted Agents

Bei Coding und Testing bereitet die API die registrierten Repositories auf exakten Quellständen vor. Sie lädt begrenzte Archive und ein serverseitiges Auftragsmanifest in eine isolierte Foundry-Sitzung. Das Manifest bindet Job, Projekt, Repository-IDs, Branches, Quellstände und Validierungsprofile. Die Laufzeit verwendet eine geprüfte Kopie des Manifests im Speicher. [5] [7]

Das per Digest festgelegte Image stammt aus Azure Container Registry. Die Sitzung bearbeitet die bereitgestellte Arbeitskopie und erhält dafür Dateisystem- und Validierungswerkzeuge. Der Zugriff auf Azure DevOps bleibt bei API und Publisher. [1] [3] [5]

Privater Round Table

Requirements und Documentation können einen Entwurf prüfen lassen. Der Worker hält den Entwurf zurück, führt die ausgewählten Beiträge und die Bewertung des Vorsitzes über einen separat versionierten Dienst aus und liefert die validierte Überarbeitung des ursprünglichen Autors. Auswahl und Protokoll bleiben dem Auftrag und seinem Eigentümer zugeordnet. Dieser Dienst arbeitet ohne Ausführungswerkzeuge. [5]

5 Orchestrierung und Wiederaufnahme

Planung und Übergaben

Der Orchestrator klärt Ziel, Projekt und offene Entscheidungen, bevor er einen Plan erstellt. Die Antwort des Auftraggebers setzt denselben Auftrag fort. Ein Plan enthält bis zu vier Spezialistenaufgaben. Der Server prüft Agentenauswahl, Projekt, Umgebung und Abhängigkeiten vor der Übergabe. Abhängige Aufgaben erhalten das Ergebnis des Vorgängers. [6]

Der freigegebene Agentenumfang und akzeptierte Eingaben bleiben an die Sitzung gebunden. Nachträgliche Hinweise können noch nicht gestartete Aufgaben beeinflussen. Für Recherche mit privaten Vorgängerergebnissen stehen ausschließlich zugelassene interne Quellen zur Verfügung. [6]

Arbeitsreihenfolge und Ausführungsschutz

Ausführungsplätze je Eigentümer

Standardmäßig läuft ein Wurzelauftrag. Die Administration kann bis zu drei zulassen. Eine Orchestrator-Familie teilt einen Ausführungsplatz und gibt ihn beim Warten auf eine menschliche Entscheidung frei.

Konkurrierende Zustellungen

Der Scheduler beansprucht die aktuelle Version atomar per ETag. Ein Worker darf Fortschritt nur für die laufende Ausführung speichern. Coding prüft dazu seine Vorschlagsrevision.

Verlorene Bestätigung

Nach einer mehrdeutigen Queue-Antwort liest die API den dauerhaften Datensatz erneut. Wurde der Auftrag bereits beansprucht, bleibt er angenommen.

Fehler und Abbruch

Fehlgeschlagene Voraussetzungen beenden noch nicht gestartete abhängige Aufgaben. Bereits vorhandene Artefakte und Publikationsbelege bleiben nachvollziehbar. Ein periodischer Abgleich übernimmt die Wiederaufnahme bei ausgefallenen Benachrichtigungen. [4] [6] [10]

6 Coding, Testing und Artefaktfreigabe

Validierung der Arbeitskopien

Das Projektprofil legt fest, welche Prüfungen und Werkzeuge für die Laufzeit zulässig sind. Coding erstellt den Änderungsvorschlag und führt gezielte Prüfungen aus. Testing führt die ausgewählten Testkategorien aus und weist fehlende Tests als übersprungen aus. Ein geänderter Repository-Stand macht vorherige Prüfergebnisse für diesen Stand ungültig. [5] [7]

Bearbeitung, Validierung, Checkpoint und Finalisierung sind je Arbeitskopie serialisiert. Die Laufzeit gibt einen finalisierten Vorschlag mit festem Basisstand, Diff und Prüfergebnissen zurück. Spätere Änderungen gehören zu einer neuen Revision. [5] [7]

Gemeinsamer Vorschlag von Coding und Testing

Für die gemeinsame Auslieferung durch Coding und Testing wählt der Auftraggeber das Ziel vor dem Start. Der Orchestrator bindet den Auftrag an die Branches der gewählten Umgebung. Coding speichert die Bearbeitung als Checkpoint. Testing übernimmt diesen in eine eigene, gepinnte Sitzung und erhält die bisherigen Änderungen bytegenau. Es ergänzt Tests und liefert einen kombinierten Vorschlag. Die Freigabe setzt bestandene Testbelege für die geänderten Repositories voraus. [6] [7]

Bindung der Freigabe

Die Freigabe gilt dem konkreten Vorschlag und seiner SHA-256-Prüfsumme. Ein kompatibles Ziel kann den geprüften Vorschlag weiterverwenden. Erfordert das Ziel einen anderen Basisstand, wird der Vorschlag neu entwickelt beziehungsweise vorbereitet, validiert und freigegeben. Checkpoints dienen der Fortsetzung und bleiben von veröffentlichbaren Artefakten getrennt. [4] [7]

7 Auslieferung in Projektumgebungen

Entwicklung oder Produktion

Der reguläre Auslieferungsweg führt in die registrierte Entwicklungs- oder Produktionsumgebung. Die Zielwahl legt Repository-Branches und zugehörige Pipelines fest. Bei begleiteter Auslieferung bestätigt der Auftraggeber die geprüfte Änderung und die Integration. Für dafür konfigurierte Ziele kann er die Auslieferung vorab freigeben. Die Plattform bindet diese Zustimmung nach erfolgreicher Validierung durch Testing an das konkrete Artefakt. [2] [7]

Integration und Rollout-Nachweis

Der Publisher prüft Artefakt, Zielbranches und Freigabe. Er erstellt die Pull Requests, prüft deren Quellrevision und führt die Integration aus, sobald die Branchrichtlinien erfüllt sind. Die registrierte Pipeline stellt den integrierten Stand bereit. Der Rollout-Observer ordnet den Pipeline-Lauf diesem Stand zu und meldet die Umgebung erst nach erfolgreicher Deployment-Stage als bereit. [2] [7] [10]

Ein Entwurfs-PR steht für die Übergabe zur weiteren Prüfung zur Verfügung. In diesem Modus endet der Auftrag mit der Veröffentlichung des Entwurfs. [2]



Abb. 2 · Zielauswahl im Orchestrator mit Beispieldaten.

8 Plattformrelease

Die Bereitstellung der Plattform verwendet eigene Pipelines und Identitäten. Sie folgt einem anderen Verfahren als die Auslieferung eines vom Workspace bearbeiteten Projekts. [7] [8]

Grundlage und Umgebungen

Bicep-Quellen beschreiben die Azure-Grundlage und erzeugen die Bereitstellungseinstiege. Entwicklung und Produktion verwenden getrennte Functions, Web-Apps, Foundry-Projekte und föderierte Deployment-Identitäten. Modellbereitstellungen und Modellquoten werden gemeinsam genutzt. [1] [3] [8]

Prüfkette

1. **Stand auswählen und prüfen.** Azure DevOps bindet Validierung und Paketierung an den ausgewählten Quellstand. Tests, Coverage-Grenzen und Infrastrukturprüfungen laufen vor der Veröffentlichung der Release-Artefakte.
2. **Pakete und Images versiegeln.** API, Publisher und das umgebungsspezifische Web-Bundle erhalten Prüfsummen. Hosted-Images werden über unveränderliche Registry-Digests referenziert.
3. **Versionen aktivieren.** Der Deploy-Lauf reserviert die Zielumgebung und stellt die ausgewählten Definitionen und Pakete bereit. Jeder verwendete Foundry-Endpunkt erhält genau eine mit 100 % ausgewählte Version. Die Wartungsanzeige begleitet den Wechsel.
4. **Ziel prüfen.** Paketidentität, Web-Oberfläche, API und ausgewählte Agentenversionen werden überprüft. Der abschließende Release-Eintrag bindet den bestätigten Zustand an die Auslieferung. [8] [9]

Rückkehr zu einer früheren Version

Ein Lauf darf nur mit seiner aktuellen, eindeutig zugeordneten Reservierung in die Umgebung zurückschreiben. Die Wiederherstellung verwendet das frühere versiegelte Artefakt. Eine verlorene Bestätigung für eine bereits erfolgreich abgeschlossene Auslieferung wird mit den gespeicherten Belegen abgeglichen. [8]

9 Betriebsüberwachung und Störungsfälle

Beobachtung

Application Insights und Log Analytics halten Korrelation, Ausführungsphase, Status, Dauer und sichere Modellmetadaten fest. Prompt-Inhalte, Quellcode, Diffs und Zugangsdaten bleiben außerhalb der Telemetrie. Die Konfiguration begrenzt Datenaufnahme und Aufbewahrung je Umgebung. Private Auftragsdaten bleiben in ihren jeweiligen eigentümergebundenen Speichern. [4] [9]

Ausführung unterbrechen und fortsetzen

Die Hosted-Laufzeit begrenzt ein Ausführungsfenster auf 40 Werkzeugrunden und 160 Aufrufe. Bleibt nach begonnenem Fortschritt drei Minuten lang das nächste Aktivitätssignal aus, pausiert der Worker und versucht einen Checkpoint zu sichern. Eine Ausführungsfrist von 18 Minuten begrenzt auch stille Antworten. Die Fortsetzung setzt einen zum Auftrag passenden Checkpoint voraus. [5] [6]

Ein Hintergrundabgleich prüft ausstehende Sitzungsbereinigungen und verarbeitet verpasste Zustandswechsel. Bei einer mehrdeutigen Antwort eines externen Systems liest der zuständige Worker vorhandene Belege erneut. Er prüft den Auftrag, bevor er eine Operation wiederholt. [4] [5] [6] [7]

Kosten und Validierung

Ein Timer aktualisiert serverseitig den sichtbaren Kostensnapshot. Das Monatsbudget meldet Schwellenwerte, und eine Monatssperre verhindert doppelte Warnungen. Zeitversetzte Azure-Buchungen bleiben über den Zeitstempel des Snapshots erkennbar. [9]

Der Emulator-Test mit 25 Eigentümern prüft konkurrierende Zustellungen, Scheduler und Isolation. Live-Prüfungen am Ziel bestätigen Erreichbarkeit und ausgewählte Versionen. Für Störungen steht eine an das aktive Release gebundene Wartungsanzeige zur Verfügung. [6] [8] [9]

Quellen und Abbildungen

Stand 24. November 2026. Die Produktreferenz dokumentiert den öffentlichen Funktionsumfang und API-Vertrag. Microsoft Learn beschreibt die verwendeten Azure-Dienste und Referenzmuster.

[1] **AIS Workspace und seine Webplattform**
Öffentlicher Funktionskatalog · Azure Static Web Apps · Functions Flex Consumption

[2] **AIS Workspace · Öffentliche Produktreferenz**
OpenAPI · AIS Live

[3] **Microsoft Learn · Identität und Zugriff**
App Service Authentication · Verwaltete Identitäten

[4] **Microsoft Learn · Persistenz und Suche**
Azure Tables und ETags · Azure Queue Storage · Azure AI Search

[5] **Microsoft Learn · Agentenlaufzeiten**
Foundry Hosted Agents · Azure Responses API

[6] **Microsoft Azure Architecture Center · Queue-Verarbeitung**
Competing Consumers · Queue-Based Load Leveling

[7] **Microsoft Learn · CI/CD für Projektumgebungen**
Azure-Pipelines-Umgebungen · Branch-basierte Pipeline-Trigger

[8] **Microsoft Learn · Infrastruktur und Release-Artefakte**
Bicep · Azure Pipeline Artifacts

[9] **Microsoft Learn · Betriebsüberwachung und Kosten**
Application Insights · Azure-Budgets

[10] **Microsoft Learn · Prüfung und Freigabe**
Branchrichtlinien · Pipeline-Freigaben und Prüfungen

Bildnachweise. Abb. 1 ist die Systemdarstellung des AIS-Workspace-Teams. Abb. 2 zeigt die Workspace-Oberfläche mit Beispieldaten. Logos und Agentensymbole entsprechen der Plattformdarstellung.