Zcrafter

Technical Overview

November 2026

Abstract

Zcrafter builds two products for IBM z/OS mainframes. **Cobots** is a desktop application in which a team of AI agents works on a live mainframe. Agents read on their own, every change waits for a person to approve it, and every action is recorded. **Bench** is a set of mainframe tasks, each of which can be reset, used to train and test AI agents. It scores an agent on what actually happened on the system, not on what the agent says it did. Cobots installs nothing on the mainframe: it uses z/OSMF, the REST interface that IBM ships with z/OS, and acts with the permissions of the person signed in.

1. Introduction

This document describes what each product is, how it is built and how it runs. Table 1 compares the two. Section 2 covers Cobots in detail, Section 3 covers Bench, and Section 4 lists the open interfaces both are built on. It is written for two kinds of reader, people who know mainframes and people who know AI, so Table 2 explains the terms from each side.

Table 1. The two products side by side.

	Cobots	**Bench**
What it is	A desktop app with a team of AI agents that work on z/OS	Mainframe tasks that train and score AI agents
Who uses it	Mainframe engineers, operators and the teams modernizing their systems	AI labs and the teams that build models
What comes out	Finished mainframe work, with an approval and an audit record for each change	A score for each attempt and a full record of how the agent got there

Table 2. Terms used in this document.

z/OS	IBM's mainframe operating system	**z/OSMF**	The REST interface IBM ships with z/OS
Data set	A file on z/OS	**3270**	The text screen used to work on a mainframe
Member	One item inside a library data set	**Model**	The AI system that reads a request and replies
Job	One run of a batch program	**Agent**	A model that takes actions, step by step
JCL	The script that describes a job	**Tool**	One action an agent may ask for
Condition code	A job step's result; 0000 means normal	**MCP**	An open standard for giving agents tools
Abend	A job that ends abnormally, such as S0C7		

2. Cobots

2.1. What it does

A person chats with an agent, or with the whole team in one room. Each agent has a role, such as z/OS Systems, Batch Ops, COBOL Dev or Security, and a lead agent passes work between them.

Three rules shape everything an agent does:

- **Reads run at once.** Listing data sets, browsing members, checking job status and reading job output change nothing, so agents do them without asking.
- **Changes wait for a person.** An edit, a job submission or a delete stops at an approval card that shows the exact change (Figure 1). Console and TSO commands need approval every time.
- **Everything is visible.** Each agent has a view-only 3270 screen that shows every action it takes on the mainframe, in order (Figure 4).

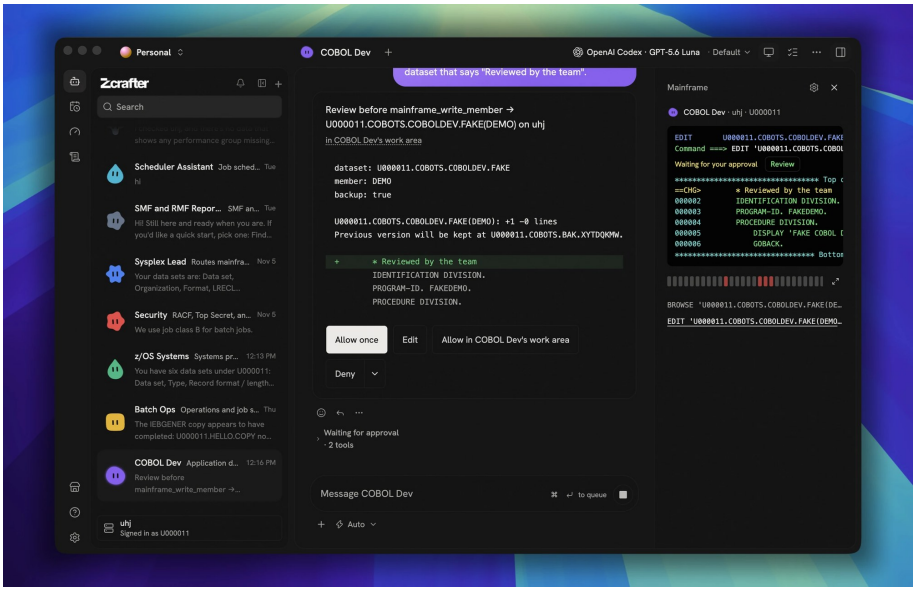


Figure 1. An approval card in Cobots. The agent wants to add one line to a member. The card shows the line, where the previous version will be kept, and the choices: allow once, edit, or deny.

2.2. Architecture

Cobots is one application made of five parts, all on the engineer's own machine (Figure 2). It connects out to two places and no others: the mainframe, and the AI model the customer chose.

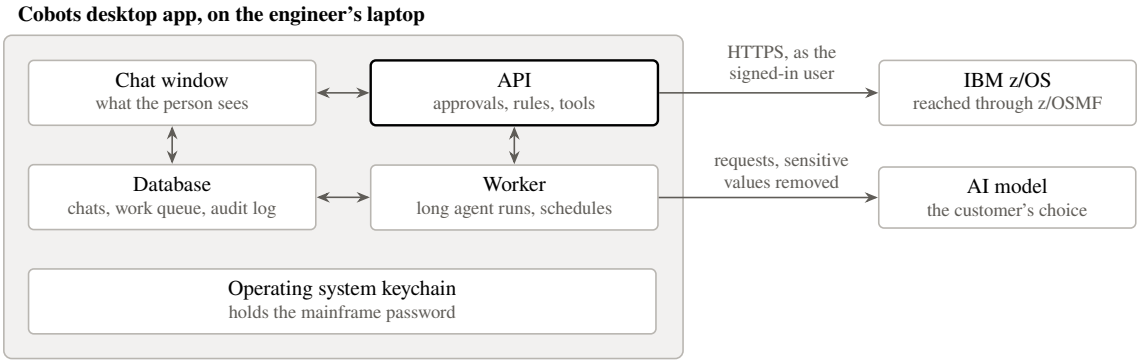


Figure 2. The parts of Cobots. Everything in the shaded box runs on one machine inside the company network. Nothing is installed on the mainframe.

- **Chat window.** Shows chats, approval cards and each agent's mainframe screen.
- **API.** Checks every action against the approval rules before anything reaches the mainframe.
- **Worker.** Runs long agent tasks and scheduled jobs in the background.
- **Database.** A built-in PostgreSQL database: conversations, the work queue and the audit log.
- **Keychain.** Holds the mainframe password. It is read for one action at a time and never reaches the model.

2.3. How a request becomes mainframe work

Figure 3 follows one request. The model can only ask to use a tool. The API, not the model, decides whether that tool runs. Table 3 lists the kinds of tool, and Figure 4 shows the end of one approved change.

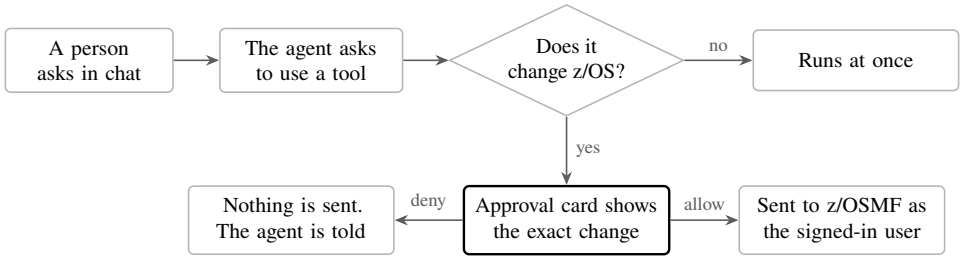


Figure 3. The path of one request. A read goes straight through; a change stops until a person allows it. Either way the result returns to the chat, and the action is added to the audit log and to the agent's 3270 screen.

Table 3. Kinds of tool and what happens when an agent asks for one.

Kind	Examples	What happens
Read	List data sets and members, read a member or a USS file, search, job status, job output	Runs at once
Change	Write a member; create, copy, rename or delete a data set; submit or cancel a job	Waits on an approval card
Command	Console and TSO commands	Needs approval every time
Plan	Several steps in order, each checked against the result of the step before	One card for the whole plan

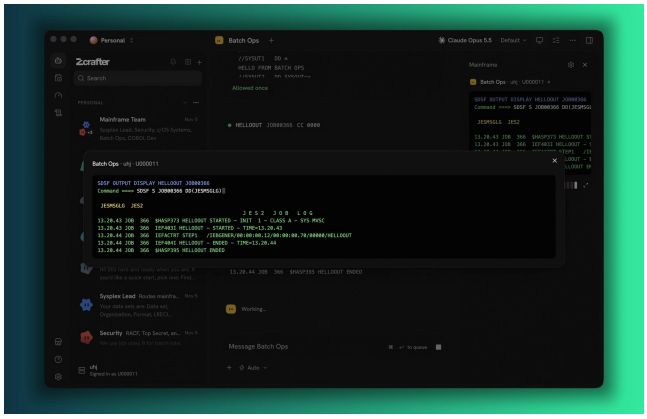


Figure 4. An agent's mainframe screen, opened large: the log of a job the agent submitted after approval, which ended with condition code 0000.

2.4. Security

- **Permissions.** An agent signs in to z/OSMF as the person using the app. The site's own security rules decide what it can reach, exactly as they do for that person.
- **Passwords.** Kept in the operating system keychain, never in the database, the logs or anything sent to the model.
- **Approvals.** Every change needs a person. There is one exception, and the person has to turn it on: an agent may write inside its own work area without a card. It can never submit jobs or run commands there.
- **Audit.** Every action on the mainframe, reads included, is written to the audit log. Log lines never contain the contents of a data set.
- **Model privacy.** Before a request leaves for the model, Social Security, card and account numbers and IP addresses are swapped for placeholders. The real values are put back in the reply, on the laptop.
- **Network.** The app accepts connections only from its own machine. Each mainframe connection is checked against the certificate saved at sign-in.

2.5. How it is hosted

- **On a laptop (the default).** A signed installer for macOS and Windows. The app carries its own database and services, so there is no server, no container and no cloud account to set up. Each engineer's password stays on their own machine, and nothing central holds everyone's passwords. Updates are signed and install in the app.
- **On a team server (optional).** The same code runs as containers inside the company network, with real sign-in, shared agents, shared approvals and one audit log. The API stores nothing itself, so more copies can run side by side. Per-user mainframe sign-in on the server is in development; today the laptop is the complete product.
- **The model.** The customer connects their own: Anthropic, OpenAI, Google Gemini (including through Vertex AI), Amazon Bedrock, OpenRouter, or a model they run themselves. No Zcrafter service sits between the app, the mainframe and the model.
- **On the mainframe.** z/OSMF switched on, and a user ID. Nothing else.

3. Bench

Bench answers one question: how good is an AI agent at real mainframe work? A useful agent has to do more than write COBOL. It has to act inside a system where a job can finish normally and still produce the wrong result. Bench gives an agent a task on a z/OS system, limits what it may touch, checks the result on the system, and resets for the next attempt (Figure 5). It can be used to test a model, by scoring it on a fixed set of tasks, or to train one, by learning from the scores and the records of each attempt. Table 4 describes the four parts.

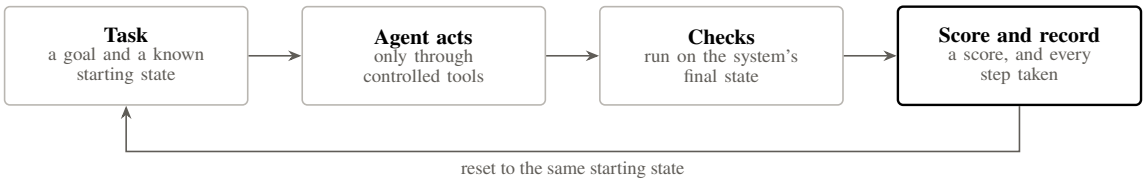


Figure 5. The Bench loop. Every attempt starts from the same state, so two attempts can be compared.

Table 4. The four parts of Bench.

Task environments	Separate tasks with COBOL, JCL, data sets, jobs and operational failures. Each one starts from a known state.
Controlled tools	TN3270 screens, the Model Context Protocol (MCP) and APIs. The tools themselves enforce what the agent may read and change, so the limits do not depend on the model behaving.
Deterministic rewards	Separate checks for correctness, safety and efficiency, taken from the system's final state. The same outcome always gets the same score.
Agent trajectories	What the agent saw, each action it took, each change and the outcome. They show how a result was reached, for evaluation and for training.

An example task: a payroll job abends with S0C7. The agent may read the program, the input file and the job output, and may edit only the test library. The attempt passes when the program compiles, the job ends with condition code 0000, the report matches the expected records, and the protected data sets are unchanged. All four must hold.

4. Built on open interfaces

Zcrafter's tools use the Zowe SDK, the open-source framework from the Open Mainframe Project. They reach data sets, jobs and job output through z/OSMF, the same interfaces Zowe CLI and Zowe Explorer use, instead of a proprietary agent on the host. An agent's actions therefore stay inside the access controls a mainframe team already runs. Two smaller tools share this base: **Bridge**, an MCP server that lets any AI agent work with z/OS behind an approved plan, and the **Zcrafter CLI** for larger mainframe projects.