# IBM Bob – Internal FAQ

**Table of Contents**

## 1) Executive Overview

In the fast-moving world of software development, urgency often overshadows clarity. Bob is an AI Development partner built for enterprise software teams that value understanding before action, security before speed, craft over chaos and need to move carefully, not just quickly.

Bob guides your entire Software Development Lifecycle (SDLC) process from planning to deployment. It works where your team already works: IDEs, command line, CI/CD pipelines, and GitHub. Bob follows a clear workflow of Plan → Execute → Validate → Govern, making changes across multiple files while keeping every step transparent, auditable, and aligned with enterprise guardrails.

Bob cultivates confidence and maintainability. It interprets intent, designs multi-step plans, executes coordinated changes across files and services, regenerates tests, and enforces compliance—all while keeping developers firmly in control.

Bob is built for the real-world complexities of enterprise codebases—Java, COBOL, IBM i, mainframe systems—and scales across thousands of developers without compromising security or sovereignty.

---

**About This FAQ**

This document is your compass for understanding Bob—its purpose, positioning, capabilities, and best practices. It is designed to help you:

- Communicate clearly about Bob's value to clients and teams.
- Navigate technical and strategic questions with confidence.
- Align messaging with IBM's principles of transparency and enterprise depth.

Think of this FAQ as a living guide, not a static artifact. Bob evolves, and so will this document. Check back often for updates as new capabilities, deployment options, and best practices emerge.

## 2) Product Positioning

### 2.1 What is Bob?

Bob is a development partner for enterprise software teams. It interprets intent, designs detailed plans, executes structured and contextually relevant changes across multiple files and services, regenerates tests, updates CI pipelines, and prepares governed PRs. Unlike "Chat in an Editor," Bob moves from snippet generation to task completion across the SDLC, with auditable steps and guardrails.

It abstracts model choice through semantic routing and evaluation, enabling predictable outcomes and cost control while keeping developers focused on tasks rather than managing models.

**Core capabilities:**

- **Deep repository understanding**: multi-repo semantics; type/interface mapping; dependency graphs; build configs and pipelines; architectural intent tracing.
- **Plan→Execute→Validate→Govern loop**: Plans with safeguards; coordinated changes; build/test runs; regression signals; PR preparation with policy enforcement.
- **Security-first architecture**: Semgrep-powered static analysis; intelligent secrets detection; PR governance; AI-aware protections (prompt normalization, sensitive data scanning, near real-time policy enforcement); AI red-teaming with risk scoring and remediation guidance.
- **Modernization depth**: Java/Jakarta upgrades, dependency audits, framework migrations; COBOL/RPG/IBM i/Z comprehension; remote SSH for Z/Power; IBM domain-trained modes.

### 2.2 Example Use cases Bob solves

- **Day-to-day SDLC developer tasks:** Everyday code authoring/refactoring; unit/integration test regeneration; code reviews and governed PR preparation; commit message and changelog generation; documentation updates; CI build/test triage and fixes; dependency updates—accelerated via Ask/Plan/Code modes, Bob-Shell recipes, context mentions, checkpoints, and approval policies.

- **Modernization**: Java upgrades; framework migrations; dependency audits; build/CI cleanup; repo-wide refactors; test regeneration; architecture documentation.

- **Mainframe & IBM i**: COBOL/RPG/IBM i comprehension; incremental refactoring; documentation; remote SSH workflows; Z/Power development consistency.

- **DevSecOps Orchestration**: Inline SAST (Semgrep), secrets detection, PR policy enforcement, automatic commit/PR descriptions; shift-left scanning embedded in authoring and CI/CD.

- **Productivity & Onboarding**: Explain unfamiliar architecture; Ask Mode for codebase Q&A; Plan Mode to design before implementing; literate programming, automation for repeatable CI/CD recipes.

## 2.3 Why does this matter to clients?

- **Modernize faster with governance**: Security and PR guardrails are embedded rather than bolted-on; teams ship safer PRs with fewer rollbacks.
- **Cost reduction & control**: ~40% savings through routing, caching, local/edge context, and deterministic operations; budgets, usage warnings, and telemetry enable transparent management.
- **Higher throughput & confidence**: 20–80% productivity gains reported across workflows; predictable, auditable changes; hybrid deployment options align with regulated environments and residency needs.

## 2.4 Pain points Bob solves today

- **Context fragmentation**: Bob orchestrates code, tests, configs, CI, and security in one plan→execute→validate→govern flow, reducing human "glue work."
- **Risky, slow modernization**: Structured plans, checkpoints, validation runs, and PR governance minimize rework and accelerate safe change.
- **Opaque AI cost & quality**: Bobalytics surfaces budgets; productivity telemetry reveals improvement areas; validation preserves quality.
- **Model orchestration complexity**: Models evolve weekly and most enterprises lack evaluation systems. Bob abstracts the model layer, continuously experiments, evaluates, and routes tasks to the best model by accuracy, performance, and cost—so teams focus on outcomes, not benchmarking or model management.

## 2.5 Who are Bob's customers?

### Primary Customer Segments

<u>1. Enterprise Organizations with Legacy Systems</u>
Characteristics:
- Large, existing codebases with history and constraints
- Decades-old systems requiring modernization (Java, COBOL, RPG, IBM i, mainframe)
- Complex, interconnected architectures across multiple repos
- Technical debt that needs systematic reduction

Why Bob:
- Built for the messy reality of production codebases
- Deep expertise in enterprise languages competitors don't support
- Cross-repo understanding for complex dependencies
- Proven modernization capabilities at scale

<u>2. Regulated Industries Requiring Compliance</u>
Characteristics:
- FedRAMP, HIPAA, PCI compliance requirements

- Need for auditable, predictable outcomes
- Hybrid deployment needs (SaaS, on-prem, sovereign cloud, air-gapped)
- Security-first development practices

Why Bob:
- Shift-left security embedded in workflows
- AI runtime security (Palo Alto Prisma AIRS)
- Hybrid deployment flexibility
- Delivers secure, production-ready code every time
- Compliance expertise (FedRAMP, HIPAA, PCI) (currently available in Bob marketplace)

### 3. Security-Conscious Enterprises

Characteristics:
- Deploying AI in production with security concerns
- Need protection against AI-specific threats (prompt injection, jailbreaks, data poisoning)
- Require AI-aware security integrated into developer workflows
- Traditional firewalls/scanners insufficient for language-based threats

Why Bob:
- AI development tool with integrated AI runtime security
- Catches threat traditional tools miss
- Real-time policy enforcement across model calls and APIs
- Continuous validation as models and data evolve

### 4. Professional Development Teams at Scale

Characteristics:
- Teams working on real, long-lived systems
- Need for consistency across large developer organizations
- Require measurable productivity improvements
- Focus on long-term maintainability over short-term output

Why Bob:
- Proven at scale (9,000+ IBM developers, growing to 20,000+)
- 95% activation rate, 65% daily usage
- 20-80% productivity gains validated
- Augments workflows without disruption
- Cost transparency through Bobalytics

**Technical Decision Makers:**

| Persona | Needs | Bob’s Value |
|---|---|---|
| **CIO/CTO** | Budget control, governance, enterprise strategy, measurable ROI | 20-80% productivity gains, ~40% cost savings, enterprise-scale validation |
| **CISO/Security Lead** | AI runtime security, shift-left security, compliance, vulnerability management | Integrated AI security, FedRAMP/HIPAA/PCI expertise, automated compliance |

| | | |
|---|---|---|
| **Architect** | Technical architecture, reasoning-focused support, critical decisions | Deep context awareness, understanding before action, repo-wide analysis, ability to read and create diagrams, as well as pulling in context from external sources via MCP |
| **Dev Manager** | Team productivity, tooling decisions, workflow integration | Augments existing workflows, keeps developers in control, measurable gains |
| **Platform Lead** | Infrastructure, deployment models, hybrid flexibility | SaaS, on-prem, sovereign cloud, Z/Power native deployments |

**End Users:**

| **Persona** | **Needs** | **Bob's Value** |
|---|---|---|
| **Software Developers** | Navigate complex codebases, understand before changing, confidence in decisions | Development partner, proactive insights, context-aware support |
| **DevOps Engineers** | Infrastructure code understanding, deployment pipelines, system configurations | Bob-Shell for CLI workflows, CI/CD integration, automation support |
| **Security Engineers** | SAST scanning, vulnerability remediation, compliance enforcement | IDE-native security, intelligent secrets detection, automated PR reviews |
| **Developers (Onboarding to Unfamiliar Codebases)** | Get up to speed quickly, understand legacy systems, reduce learning curve | Deep context analysis, helps understand before modifying, reasoning-focused |

## 3) How Bob Works

### 3.1 What can you do with Bob?

**Write and modify code**

- Help you write code from natural language descriptions
- Refactor existing code to improve readability, performance, or structure
- Debug issues with intelligent error analysis and suggested fixes
- Create new files, functions, and entire projects with proper structure

The more specific your request, the better the results. "Create a React component that displays a sortable table of user data" works better than "Make me a table component."

**Understand your codebase**

Bob helps you understand unfamiliar or complex code before modifying it through deep context awareness—analyzing your entire codebase, dependencies, environment, and project structure.

- Get answers to questions about your code's functionality and architecture
- Receive explanations of complex code segments in plain language
- Navigate large codebases more efficiently with contextual understanding
- Learn new frameworks and libraries through interactive guidance

**Automate development tasks**

Bob helps you streamline routine development tasks while maintaining quality and consistency.

- Help you generate commit messages based on your staged changes
- Help you create pull request descriptions that clearly explain your changes
- Support you in automating repetitive coding tasks with natural language instructions
- Assist with writing and updating documentation that stays in sync with your code

**Key features**

Specialized modes
Bob adapts to your specific needs with purpose-built modes that optimize behavior for different development scenarios:

- **Code mode**: Write, modify, and refactor code with precision
- **Ask mode**: Get answers and explanations about your codebase
- **Plan mode**: Plan and design before implementation
- **Advanced mode**: Access extended capabilities for complex tasks

Each mode is designed for specific development scenarios, allowing you to work efficiently without adjusting your communication style. In addition, Bob also allows you to create your own modes that meet your business needs/workflows and share them. (See here for more details.)

Powerful tools
Bob comes with a comprehensive set of tools that extend capabilities beyond text generation:

- **File operations**: Read, write, and modify files in your project
- **Command execution**: Run commands directly from the chat interface
- **Browser control**: Navigate and interact with web resources
- **Code analysis**: Search codebases and analyze definitions
- **MCP integration**: Connect to external tools and services

These tools work together seamlessly, allowing you to accomplish complex tasks without switching between applications.

MCP (Model Context Protocol)
MCP extends Bob's capabilities with custom tools:

- Connect to databases and APIs
- Access specialized development resources
- Integrate with your organization's internal systems
- Create custom workflows for your specific needs

Bob-Shell
Take Bob's capabilities beyond your editor with Bob-Shell, bringing the same AI assistance to your terminal. Bob-Shell provides intelligent support for command-line tasks with the same context awareness and reasoning-focused approach. To get started, see Installing and setting up Bob-Shell.

Further reference on Bob's features

https://pages.github.ibm.com/code-assistant/bob-docs/

## 3.2 Models

### 3.2.1 What LLM models does Bob use?

Bob uses a model orchestration layer rather than a single model. This layer dynamically selects from multiple families of large language models (LLMs), including (**subject to change**):

- Claude
- Mistral/Devstral
- Granite
- Specialized fine-tuned models for code reasoning, security, and next-edit prediction

Bob will also support sovereign-deployable models and IBM-trusted models for on-premises installations.

### 3.2.2 How does Bob use them?

- **Automatic Routing:** Bob evaluates each task and routes it to the most suitable model based on:
  - Accuracy (quality of output for the task)
  - Performance (speed and efficiency)

  - Cost (optimized resource usage)

- **Continuous Evaluation:** Bob experiments and benchmarks models internally, so customers don't need to manage model selection or updates.

- **Model-Agnostic Experience:** Developers interact with **capabilities** (Ask, Plan, Code, Advanced) rather than choosing models. Bob abstracts the complexity and ensures predictable outcomes.

### 3.2.3 Why should we NOT talk about models with customers?

- **Focus on Outcomes, Not Internals:** Customers care about **business results**—modernization speed, security posture, cost transparency — not which LLM is behind the scenes.

- **Avoid Confusion:** Discussing models can lead to unnecessary debates about "which model is better," which is irrelevant because Bob handles orchestration automatically.

- **Future-Proof Messaging:** Models evolve weekly. Naming specific models creates a **point-in-time comparison** that ages quickly and undermines credibility.

- **Enterprise Value Proposition:** Bob's differentiation is **not model choice**; it's:
  - Enterprise depth (Java, COBOL, IBM i, Z/Power)
  - Security-first architecture
  - Hybrid deployment flexibility
  - Governance and cost transparency

- **Strategic Principle:** IBM's official communication guidelines emphasize **capabilities and outcomes**, not underlying models. The message should be:

  *"Bob abstracts the model layer and continuously routes tasks to the best model for accuracy, performance, and cost—so you focus on tasks, not models."*

## 3.3 Deployment and installation

**Deployment options:**

- Clients – macOS/Windows/Linux

- Server – SaaS (Q1); on-prem (Q3); sovereign cloud (TBD)
  - At GA, Bob backend service will be deployed on IBM Cloud (Washington). Other data centers' / hyperscalers' availability will be announced later.
  - Bob backend service accesses models hosted on IBM Cloud (e.g., Granite, Mistral) and AWS (e.g., Claude).

- SSH for Z/Power; Bob-Shell for CI/CD (Q1)

**Installation – IBM Bob**

- System requirements: macOS, Linux, Windows; 4GB+ RAM (8GB recommended); 500MB disk; access to public internet.
- Installers: macOS (.dmg), Linux (AppImage/.deb), Windows (.exe); authentication via API key; verify via "About" and basic chat/tool tests.
- Post-install: review security guidance; set workspace defaults; configure MCP servers; add custom instructions/modes; set keyboard shortcuts.

**Installation – Bob-Shell (CLI)**

- Requirements: Node.js v22.15+; OS: macOS, Linux, Windows, z/OS, Linux on Z.
- Install via Command Palette in Bob or terminal scripts (curl/bash; PowerShell). Authentication via API key.
- Safety: non-destructive by default; use --yolo to enable writes; respects start directory boundaries; use @file/@folder mentions; consider markdown tagging for downstream processing.

## 3.4 Security and governance

### 3.4.1 Data protection & privacy

- **Encryption at rest (storage)**
  Bob supports encryption at rest for customer data and artifacts when deployed in SaaS or on-prem/sovereign environments—aligned to IBM security baselines and enterprise KMS/HSM options depending on the deployment model.

- **Encryption in transit (TLS)**
  All service-to-service communication and developer interactions are protected with TLS (HTTPS) for IDE, CLI/Bob-Shell, CI/CD, and service endpoints.

- **Data minimization & redaction**
  Bob applies prompt normalization and sensitive data scanning to detect/remove secrets and sensitive strings from prompts/outputs before model calls; this reduces leakage risk and enforces least-privilege data principles.

- **Onclient context & RAG (privacy preserving)**
  Where feasible, Bob uses local/edge context (onclient RAG, cached embeddings) to limit what leaves the developer workstation, lowering exposure.

- **Customer data** will not be used for training purposes.

### 3.4.2 Application & code security (shift left)

- **Inline SAST (Semgrep) in IDE/PR**
  Bob runs Semgrep-powered static analysis during authoring and again at PR time, surfacing findings inline with remediation guidance.

- **Intelligent secrets detection**
  LLM-assisted secrets detection (reduced false positives) is applied on edited files, staging changes, and PRs.

- **PR policy enforcement & governance**
  Bob enforces enterprise PR rules (required checks, reviewers, blocking conditions, compliance banners) and automates PR descriptions/changelogs so merges meet organizational policy.

- **CI/CD guardrails & continuous validation**
  Plans and changes are validated via CI builds/tests with policy gates (security scans, coverage, license checks) before merge/roll-out.

### 3.4.3 AI-aware protections (runtime & model layer)

- **Prompt Injection blocking & jailbreak prevention**
  Bob normalizes prompts, filters hostile instructions, and blocks jailbreak patterns; detections trigger policy actions (warn, quarantine, block).

- **Data Poisoning & Model vulnerability detection**
  Bob's AI aware runtime applies near real-time policy enforcement on model calls/APIs and flags anomalous behavior or known model CVEs.

- **Model orchestration with governance**
  Bob's semantic routing chooses the best model per task (accuracy/performance/cost), with auditability of routing decisions and usage budgets exposed via Bobalytics.

### 3.4.4 Access controls, isolation & governance

- **.bobignore boundaries**
  Developers and teams define file/folder access boundaries with .bobignore (read/write tools respect these; exceptions are documented and mitigated via VCS/checkpoints).

- **Approval policies (manual/auto/hybrid)**
  Risky actions (writes, remote commands, resource access) are gated by approvals; actions and diffs are explainable and logged.

- **Project rules & mode scoped policies**
  Per project and per mode **rule files** enforce coding standards, documentation, and merge criteria.

- **Audit logging & telemetry (Bobalytics)**
  Budgets, routing transparency, warnings, and task throughput per developer provide leadership visibility and audit trails.

### 3.4.5 Deployment security & compliance

- **Hybrid deployment (SaaS, on-prem, sovereign, air-gapped)**
  Bob supports SaaS GA, on-prem/sovereign, Z/Power native, and air-gapped instances—enabling strict residency and isolation requirements.

- **Compliance alignment**
  Built-in expertise in compliances (initially provided through Bob marketplace); automated standards enforcement is integrated at PR/CI steps for predictable, auditable outcomes.

## 3.5 How Bob enables org-wide sharing of rules, commands, and modes

- **Shared, versioned Rule Packs**: Bob supports rules defined globally or per project in .bob/, and mode specific rules under /.bob/rules{modeslug}/. SMEs can publish rule bundles (coding standards, doc requirements, PR policies) that teams apply consistently in IDE/PR/CI/CD, making governance auditable across repos.

- **Teamwide Custom Modes**: Beyond built-in modes (Ask/Plan/Code/Advanced), SMEs can create and share custom modes with tailored prompts and tool permissions, standardizing specialized tasks (e.g., security reviews, documentation) across the org.

- **Curated BobShell Recipes (Commands)**: Developers can use BobShell to run repeatable, safebydefault CLI workflows (e.g., modernization steps, security scans, build/CI cleanups). Commands require explicit approval unless auto-approval is configured; this supports central policy gating while scaling SME defined recipes.

- **Marketplace/Catalog Distribution**: The Bob Marketplace (a catalog of community or org-curated assets) lets teams browse, install, and share custom modes and configurations—including MCP servers—so SMEs can publish once and let the wider org adopt responsibly.

- **File Access Boundaries with .bobignore**: Org rule packs are complemented by .bobignore patterns (gitignore style) to prevent reads/writes to sensitive files or build artifacts, define workspace scope, and show blocked access notifications in the UI—ensuring shared workflows never overreach file boundaries.

- **Approval Policies & Safety Controls**: Teams can configure manual/auto/hybrid approvals for actions and commands, keeping high risk steps gated while allowing low risk automation to flow—ideal for org wide recipes that need governance without friction.

- **Operational Guidance & Context Mentions**: Shared guidance includes prompting best practices, checkpoints (to roll back), and context mentions (e.g., @/folder, @problems) so developers execute SME workflows consistently and provide precise context when running shared modes and commands.

## 3.6 Best practices for success

- **Mode strategy**: Start in Plan → implement in Code → Ask for explanations → Advanced for MCP/browser automation.

- **Checkpoints**: use to roll back to safe states; more effective than trying to correct flawed output with new prompts.

- **Version control discipline**: frequent small commits; branches for experiments; push regularly.

- **Prompting**: be specific; provide examples; use context mentions (@/path/to/file, @/folder, @problems, @terminal); use Enhance Prompt for structure.

- **Context management**: start new tasks for distinct goals; avoid dumping entire repo; break complex work into focused subtasks.

## 3.7 What are the "risks" of using Bob?

Bob is a powerful tool, and it's important to use it responsibly. Here are some things to keep in mind:

- **Bob can make mistakes** - Always review Bob's proposed changes carefully before approving them
- **Bob can execute commands** - Be cautious about allowing Bob to run commands, especially if you're using auto-approval
- You can use Bob to **control the changes** and help you fully understand the risks of any changes.

## 4) Market Positioning (AI for Code) & Why Pick Bob

### 4.1 Competitive disclaimer

Bob's differentiation is rooted in enterprise depth and real-world scale, not in point-in-time feature comparisons. As a principle, we avoid comparing Bob directly against named competitors, because static comparisons age quickly and often obscure what matters most to enterprise buyers: credibility, outcomes, and operational reality.

### 4.2 Market positioning & differentiators

Bob is an enterprise-grade AI SDLC partner built for long-lived, complex systems. Where typical AI coding tools focus on autocomplete or single-surface chat, Bob delivers end-to-end orchestration across IDE, CLI (Bob-Shell), CI/CD, and GitHub—with security-first guardrails, hybrid deployment options, and modernization depth for real production codebases. Bob emphasizes understanding before action, partnership over automation, and auditable outcomes that improve governance, security, and total cost of ownership.

Positioning statement: Bob is built for the real-world complexity of enterprise software—modernization, governance, and security at scale—not just writing code faster. It turns AI from novelty into a trusted, auditable partner for delivery teams.

| Differentiator | What it delivers | Why it matters |
|---|---|---|
| **1) Understanding Before Action** | Reads repo(s), dependencies, build/CI configs, and architecture before proposing changes; explains impact. | Prevents blind edits and hallucinations; reduces regressions and rework; respects architectural intent. |
| **2) Partnership Over Automation** | Structured Plan → Execute → Validate → Govern loop with transparent diffs and permissioned actions; developer stays in control. | Predictable results with auditable steps; avoids runaway behaviors and hidden decisions. |
| **3) Enterprise-Grade Scale & Depth** | Native depth in Java, COBOL, RPG, IBM i, mainframe; multi-repo semantics; remote SSH for Z/Power; domain-trained modes. | Handles real enterprise systems and complex interdependencies that span repos. |
| **4) Security-First Architecture** | Inline SAST (Semgrep), intelligent secrets detection, PR policy enforcement, prompt normalization, jailbreak protections, Prisma AIRS runtime security across IDE → PR → CI/CD. | Shift-left controls catch AI-specific and code issues early; improve trust, compliance, and merge quality. |
| **5) Authentic Adoption at Scale** | Client-zero telemetry: 20K+ onboarded, ~95% activation, ~65% daily usage, 20–80% | Demonstrated value and durability in a demanding enterprise. |

| | | |
|---|---|---|
| | productivity gains, ~40% cost savings. | |
| **6) Built for Real Codebases** | Excels with large, messy, long-lived code; multi-repo dependencies; long-term maintainability focus. | Works where most assistants struggle—production complexity, tech debt, interconnected systems. |
| **7) Cost Optimization & Transparency** | Semantic routing, aggressive caching, local/edge context; Bobalytics exposes budgets, routing decisions, and productivity telemetry. | Deliver ~40% AI cost savings and measurable ROI; leaders manage spend and outcomes. |
| **8) Hybrid Deployment Flexibility** | SaaS, on-prem, sovereign, air-gapped; Z/Power native; consistent UX across environments. | Meets residency, sovereignty, and isolation requirements for regulated industries. |
| **9) Compliance Built-In** | Built-in expertise for FedRAMP, HIPAA, PCI; automated enforcement at PR/CI steps. | Auditable outcomes and predictable governance. |
| **10) Multi-Agent Architecture** | Coordinates tasks across IDE/CLI/CI/CD/GitHub; Bob-to-Bob handoffs; headless/embedded flows. | Enables cross-surface automation and continuous validation. |
| **11) Model Intelligence, Not Pass-Through** | Abstracts model choice; continuous evaluation; semantic routing to best model by accuracy, performance, cost; on-client RAG. | Future-proofs teams against fast-changing models; keeps focus on outcomes not model names. |

## 5) Proof & ROI (Client Zero)

**Adoption & Engagement (last 6–9 months):**

- 20,000+ professional developers onboarded across enterprise product teams
- ~95% activation rate post-onboarding; ~65% daily active usage, indicating sustained workflow integration
- 1M+ structured SDLC interactions; 200K+ tool executions; 11K+ BobShell sessions embedded in CI/CD

**Productivity Impact:**

- 20–40% cycle-time reduction for complex tasks (multi-repo refactors, architectural changes)
- 50–80% acceleration for structured workflows (dependency upgrades, test regeneration, CI fixes)
- 90%+ time savings on repetitive SDLC tasks

**Cost & Efficiency:**

- ~40% AI cost reduction via semantic routing, caching, and local/edge context
- $1M+ in avoided costs from reduced rework and faster delivery cycles. These results were achieved in real production environments — across regulated, long-lived enterprise codebases — not greenfield or demo projects.

## 6) Branding & Communication Guidelines

Refer to these guidelines to maintain consistent communication about Bob across all documentation, articles, and content. Use them to ensure accurate representation of Bob's capabilities and market positioning.

### 6.1 Referring to Bob

When writing about Bob:

- Use IBM Bob once at the beginning of your content
- Use Bob throughout the rest of the document

Example:
*"IBM Bob is an AI SDLC partner that helps developers work confidently with real codebases. Bob augments your existing workflows and provides proactive insights..."*

### 6.2 Visual identity

When creating content that includes Bob's image or icon:

- Use the official IBM Bob icon provided by the design team
- Avoid using generative AI to create custom versions or variations

Using the official icon ensures consistent brand identity across all Bob-related materials. Download Bob Logo (SVG)

### 6.3 Language guidelines

| **Category** | **Recommended Phrases** |
|---|---|
| **Relationship** | Development partner; Works with you; Augments your workflow; Fits into your process |
| **Capability** | Helps you understand; Improves clarity and confidence; Designed for real codebases; Offers proactive insights |
| **Control** | Keeps you in control; You make the decisions; Transparent and predictable |
| **Approach** | Understanding first; Reasoning-focused; Context-aware |

Avoid these terms:

| **Avoid** | **Why** | **Use instead** |
|---|---|---|
| **Agent** | Implies autonomy | Partner |
| **Autonomous** | Contradicts developer control | Supportive, collaborative |

| | | |
|---|---|---|
| **Coding assistant** | Implies simple code generation | Development partner;<br>AI-powered coding assistant |
| **Writes code for you** | Passive developer role | Helps you write better code |
| **Takes over your code** | Threatening | Helps you manage your code |
| **Magic** | Undermines transparency | Intelligent, reasoning-focused |
| **Fully automatic** | Hands-off implication | Interactive, collaborative |
| **Bob the builder** | Copyright issues | Bob, your development partner |

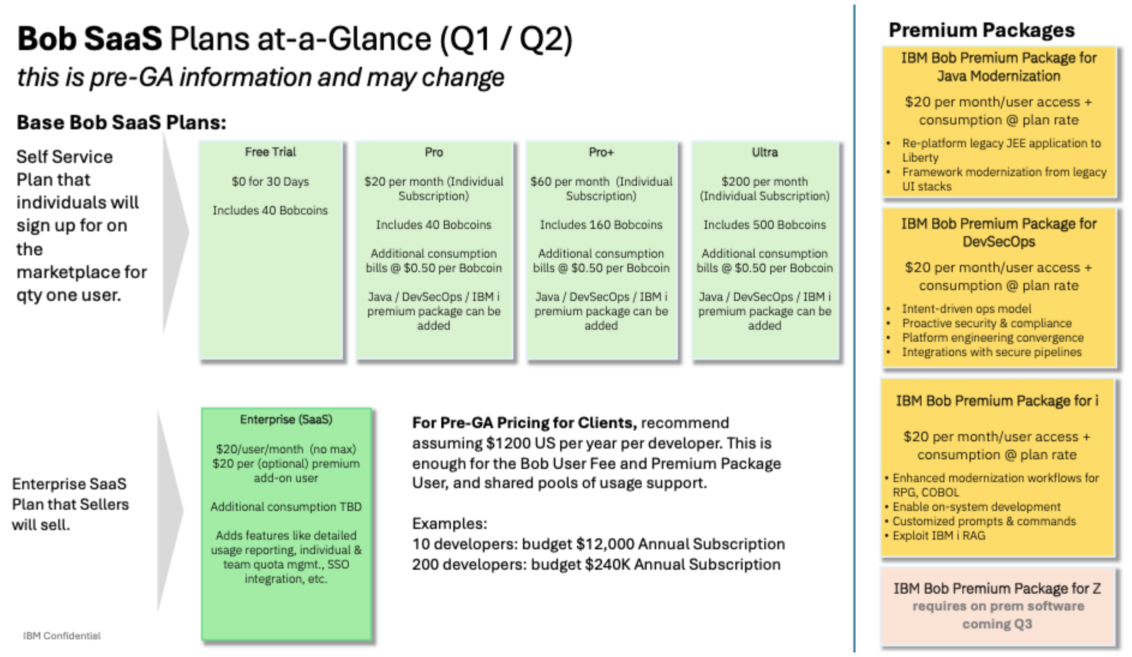
## 7) Pricing and Packaging

[Work in progress. Subject to change. Do NOT share with anyone outside of IBM, even under an NDA.]

**Objectives:**

| | |
|---|---|
| Win in Select | • PLG, SaaS-only<br>• Easy purchase and quick time to value (<5 minutes) with credit card<br>• Allow flexibility for plan changes |
| **Be the first choice for Enterprise** | • SLG<br>• Offer multiple license options<br>• Simplify pricing compared to current WCA/WCAA/WCAZ combination of parts<br>• Consider typical discounts<br>• Offer multiple deployment types (SaaS, On-Prem, Hybrid)<br>• Leverage traditional IBM sales motions |
| **Protect existing WCA investments** | • Avoid reversals<br>• Offer a path for existing clients to Bob |

**Packaging:** (This is proposed pricing, subject to change.)



**Bob SaaS** Plans at-a-Glance (Q1 / Q2)

*this is pre-GA information and may change*

**Base Bob SaaS Plans:**

Self Service Plan that individuals will sign up for on the marketplace for qty one user.

| Free Trial | Pro | Pro+ | Ultra |
|---|---|---|---|
| $0 for 30 Days<br>Includes 40 Bobcoins | $20 per month (Individual Subscription)<br>Includes 40 Bobcoins<br>Additional consumption bills @ $0.50 per Bobcoin<br>Java / DevSecOps / IBM i premium package can be added | $60 per month (Individual Subscription)<br>Includes 160 Bobcoins<br>Additional consumption bills @ $0.50 per Bobcoin<br>Java / DevSecOps / IBM i premium package can be added | $200 per month (Individual Subscription)<br>Includes 500 Bobcoins<br>Additional consumption bills @ $0.50 per Bobcoin<br>Java / DevSecOps / IBM i premium package can be added |

Enterprise SaaS Plan that Sellers will sell.

| Enterprise (SaaS) |
|---|
| $20/user/month (no max)<br>$20 per (optional) premium add-on user<br>Additional consumption TBD<br>Adds features like detailed usage reporting, individual & team quota mgmt., SSO integration, etc. |

**For Pre-GA Pricing for Clients,** recommend assuming $1200 US per year per developer. This is enough for the Bob User Fee and Premium Package User, and shared pools of usage support.

Examples:
10 developers: budget $12,000 Annual Subscription
200 developers: budget $240K Annual Subscription

IBM Confidential

**Premium Packages**

IBM Bob Premium Package for Java Modernization

$20 per month/user access + consumption @ plan rate

- Re-platform legacy JEE application to Liberty
- Framework modernization from legacy UI stacks

IBM Bob Premium Package for DevSecOps

$20 per month/user access + consumption @ plan rate

- Intent-driven ops model
- Proactive security & compliance
- Platform engineering convergence
- Integrations with secure pipelines

IBM Bob Premium Package for i

$20 per month/user access + consumption @ plan rate

- Enhanced modernization workflows for RPG, COBOL
- Enable on-system development
- Customized prompts & commands
- Exploit IBM i RAG

IBM Bob Premium Package for Z
requires on prem software
coming Q3

**Bob Software (GA 2026Q3)**

- Working with PM, Pricing, Finance and Accounting towards approval for WCA Products (Z, WCA, WCA Ansible) to migrate to Bob software parts with no Tradeup Required
- Customers can purchase / deploy WCA from Catalogs with expectation to evolve to Bob SW
- Working on alternative to allow Software entitlement on SaaS.

**Seller Call to Action:**

- Jan-June: Sell WCA
- Use Early Access, POCs /Client Bobathons to get client interest
- Include WCA in ELAs
- Once GA we will sell Bob
- Entitled Bob Customers to migrate automatically to Bob parts

**Messages For existing WCA customers**

*IBM ensures continuity and value for your investment with us. Customers currently deploying or adopting WCA should continue with confidence, as the solution delivers immediate productivity gains and accelerates progress toward modernization objectives. Both technical and commercial paths to IBM Bob will be defined once it GA's. At this time, IBM Bob or any commitments related to IBM Bob should not be in any contractual agreement with a client.*

*Detailed materials on future statements*

## 8) Roadmap

[Work in progress.  Subject to change. Do NOT share with anyone outside of IBM, even under an NDA.]

**Q1 2026**

GA: IBM Bob on SaaS
GA: IBM Bob Premium Add-ons

- Java Modernization
- IBM i Application Modernization

**Q2 2026**

GA: IBM Bob Premium Add-ons

- DevSecOps

Public Preview: IBM Z Modernization Premium Add-ons

**Q3 2026**

GA: IBM Bob on-prem
GA: IBM Bob Premium Add-ons

- IBM Z Modernization

## Appendix: IBM Watsonx Code Assistant for Z

**IBM Watsonx Code Assistant for Z statement and positioning for existing customers and potential 4Q 2025/Q1+Q2 2026 customers**

- IBM Watsonx Code Assistant for Z (WCA4Z) is IBM's leading AI-based mainframe application modernization developer solution. It goes beyond an IDE to provide proven capabilities in support of the entire mainframe application development life cycle.
- IBM Bob is a next generation IDE based code assistant from IBM that combines the latest in frontier models, agents, and IDEs/UX to deliver key capabilities across the entire SDLC to enhance developer productivity. Bob was showcased at TechXchange as a Technical Preview with plans to make available in 2026.
- IBM announced at TechXchange the technology preview of Bob. Customers that bought WCA4Z and/or are looking to invest on WCA4Z are questioning the future direction and roadmap of WCA4Z.
- Bob brings strong value proposition and technical capabilities to our customers that substantially boost the value proposition of WCA and WCA4Z – frontier LLMs, agentic IDE, cost optimized experience – this will be a game changer for IBM customers.
- IBM has decided that the WCA4Z product and technologies WILL converge to a mainframe-specific Bob offering to provide a next generation mainframe application modernization and developer experience.

**CONSIDERATIONS**

- Mainframe application modernization has unique requirements, which highly differentiated WCA4Z from general-purpose AI assistants. For example, it requires z/OS application analysis at scale, z/OS runtime information, z/OS middleware knowledge and complex logic refactoring. Continuing to deliver and innovate for these key value propositions will be central to the convergence with Bob.
- Bob is not intended to be a mainframe re-platforming assistant.
- Bob can assist with modernization projects on mainframe, e.g., migrating from WebSphere to Liberty, Java 8 to 21, etc.

**COMMITMENT TO CLIENTS: REQUEST – Please confirm if we can commit to this**

- As customers move forward in the journey of mainframe application modernization, IBM Bob plans to incorporate for its mainframe use case:
    - The feedback and inputs from our mainframe customers over the past 2 years that have driven the WCA4Z roadmap. This will drive the requirements for the converged Bob based offering
    - The converged product strategy – The key value propositions for mainframe application understanding and modernization (as described by the wheel) will

remain the basis for the core functionality and use cases, and IBM will build the converged offering accordingly.

- We will add support for VS code to support the ongoing IDE modernization journey of IBM Z customers and IBM Z tools ecosystem, in addition to Bob IDE
- s390x and Spyre support will be part of the converged product strategy
- GA timeline of the converged offering will be announced in 2026. There will be both commercial and technical migration details announced at that time
- The converged offering will be available SaaS (initially) and on prem (follows)
- IBM will protect the investment of all customers that have bought WCA4Z

**CLOSING**

- *[existing customer statement]* Customers in the process of deploying and adopting WCA4Z should continue to do so with confidence given the value of the solution and the ability to realize productivity gains and make progress towards modernization objectives. The planned converged offering will be built with key components of WCA4Z to accelerate and simplify future adoption of the new offering
- *[Pipeline customer statement]* Customers that are considering investing in WCA4Z should continue to do so with confidence given the value of the solution and the ability to realize productivity gains and make progress towards modernization objectives. The planned converged offering will be built with key components of WCA4Z to accelerate and simplify future adoption of the new offering