AI-assisted app dev for the enterprise

A Red Hat guide to unlocking developer productivity through platform engineering

Red Hat

Table of contents

Introduction

Enterprise application development is in the midst of a significant evolution, led largely by the increasing application of generative AI. Emerging, gen AI-based approaches such as vibe coding and agentic coding equip developers with tools that assist in writing code, documentation, debugging, and even building entire applications, significantly boosting productivity.

These approaches are supported by other foundational practices. By consolidating application tooling and best practices, the adoption of internal developer platforms (IDPs) is effectively streamlining development processes. Platform engineering is becoming more democratized, making advanced development tools accessible to a broader range of developers. The expansion of software supply chain security is helping to establish a security focus at every stage of the application lifecycle.

Together with AI-assisted practices, these shifts not only streamline and enhance coding, but also help reduce cognitive load for developers, improving their experience and boosting efficiency and productivity. By integrating these approaches into their application development lifecycle, organizations can accelerate delivery and unlock greater potential for innovation.

This e-book explores the trends and shifts reshaping the application development landscape, and the capabilities they provide enterprise development teams. It examines the opportunities and challenges that AI-assisted application development introduces, and outlines practical ways organizations can evolve their processes, platforms, and teams. By adopting an integrated, security-first, AI-assisted approach, organizations can position themselves to realize the full value of modern enterprise application development.

The growing role of AI in application development

The use of gen AI in application development and delivery is gaining significant traction as tools mature and evidence of success emerges. Two key trends are shaping this evolution: vibe coding and agentic coding.

Vibe coding

Vibe coding refers to an emerging mode of software development in which the human programmer acts less as a direct implementer of code and more as a high-level coordinator. Coined by computer scientist Andrej Karpathy, the term describes an approach that emphasizes intuitive, human-in-the-loop interactions, where developers communicate desired outcomes (the "vibe") via natural language instructions, conceptual overviews, and progressive refinements, rather than specifying logic in syntactic detail.

Unlike traditional programming that focuses on grasping syntax and low-level operations, vibe coding emphasizes intent, architecture, and interactive debugging. Emerging alongside the advancement of foundation models and specialized large language model (LLM)-based platforms for coding such such as Replit, Windsurf, and Cursor, and integrating principles from prompt engineering, agile design, and human-AI cocreation, vibe coding abstracts away syntactic detail so developers can focus on higher-level design and rapid iteration. The result is a more fluid and collaborative form of development that allows for greater flexibility and rapid refinement. This makes vibe coding ideal for small tasks (such as generating individual functions and writing documentation), early-stage development, and exploratory contexts.

Key aspects of vibe coding include:

- **Human-led process:** The developer stays in control, guiding the AI using natural language prompts and reviewing the outputs.
- **Lightweight tools:** Tooling typically works inside integrated development environments (IDEs) or web apps with simple, stateless interfaces, in which the AI generates code but the developer handles testing, running, and debugging.
- **Human-led planning:** Developers break down tasks and write prompts, while the AI focuses only on the current request without broader planning.
- **Manual validation:** There's no built-in feedback loop, so the developer checks the AI's output through testing and refining prompts.
- **Minimal safety features:** With no built-in guardrails, developers and external tools must catch bugs and manage risks.

Agentic coding

Agentic coding is a more advanced and autonomous evolution of AI-assisted programming. Unlike the prompt-response dynamic of vibe coding, agentic coding minimizes the need for continuous human direction. It relies on agentic AI systems, which are software agents capable of independently interpreting high-level goals, breaking down tasks into subtasks, planning execution strategies, and adapting behavior based on real-time feedback and outcomes.

Equipped with intentionality, forethought, and adaptivity, these AI agents can initiate action, access tools and application programming interfaces (APIs), retrieve and process external data, and iteratively refine outputs through cycles of self-evaluation. Architecturally, agentic coding often employs reinforcement learning and modular planning, integrating specialized subagents that collaborate to complete broader missions. If vibe coding equips developers with a high-speed copilot, agentic coding provides an intelligent collaborator capable of independently steering the aircraft. This level of autonomy makes agentic coding systems suitable for complex, multistep workflows, process automation, business operations, and dynamic, data-driven environments.

Key aspects of agentic coding include:

- **Delegated autonomy:** The AI agent takes on more responsibility, while the developer sets high-level goals and reviews the results.
- **Automated execution:** Agents work through full pipelines on their own, using tools like compilers, APIs, and version control in sandboxed environments without constant human input.
- **AI-driven planning:** The agent breaks down goals into smaller tasks and figures out the steps to reach the objective, all within set boundaries.
- **Built-in feedback loops:** Agents learn from test results, logs, or human feedback, adjusting their behavior and re-planning if needed.
- **Context awareness:** Agents can handle larger projects by remembering past interactions, documentation, and code to manage complex, multistep workflows.
- **System-level generation:** Suitable for creating full features or systems, including code, tests, configuration files, and documentation, with minimal manual stitching by developers.
- **Safety and oversight:** These systems include sandboxing, access controls, and logging for traceability. Agents monitor their own performance, analyze errors, and can automatically roll back or retry failed actions.

The future of AI-assisted software engineering lies in a hybrid workflow that combines the creative agility of vibe coding with the automated power of agentic systems. This creates a balanced model of human-AI collaboration, where the developer's role shifts to that of a supervisor who defines objectives and validates outcomes, while intelligent agents independently execute complex tasks.

The foundations of AI-assisted application development

The successful adoption of AI-assisted application development is not a standalone effort; it's based on a solid foundation of flexible, automated, and security-focused practices. IDPs and platform engineering are essential for creating the clear, consistent, and well-governed environments that generative AI tools require to operate effectively and safely.

By establishing a unified developer experience, automating infrastructure management, and integrating comprehensive software supply chain security, organizations create the necessary structure and confidence to safely incorporate AI-based code suggestions, testing, and deployment processes.

IDPs and internal developer portals

IDPs simplify and automate software development by consolidating all the necessary tools and services. They consist of 2 essential layers: the application platform, which provides the foundational infrastructure and governance, and the internal developer portal, which provides the user-friendly frontend. The internal developer portal serves as the access point to the underlying application platform, creating a unified development ecosystem that supports self-service application delivery.

IDPs are designed to solve key developer challenges by providing centralized access to all developer needs, including onboarding, resource provisioning and utilization, documentation, and a catalog of tools and technologies. They improve collaboration through shared workflows and increase visibility across teams. Most importantly, IDPs boost productivity by automating repetitive tasks and offering self-service capabilities. This is especially helpful during onboarding, when an effective IDP can help a new developer get up and running quickly, helping them deliver value in less time.

IDPs are vital for organizations seeking to maximize developer agency and speed while managing an increasingly complex technology stack. IDPs aim to address the complexities of a sophisticated technology landscape, coordinate numerous development teams, and enhance developer productivity.

Platform engineering

Platform engineering is a discipline focused on making developers more efficient and productive. By providing common tools and capabilities via an IDP, organizations equip developers with everything they need to safely get their code to the production environment and roll back quickly when required, without needing to manage the underlying complexity. In platform engineering, developers aren't required to have deep knowledge of Kubernetes or other complex systems. Instead, platform engineers can create easily replicable solutions and expose them as templates that free developers from unnecessary toil. This approach allows developers to concentrate on writing code and delivering features, rather than managing infrastructure.

Most successful platform teams take a "platform as a product" approach, where they view their organization's IDP as a product and developers as their customers. These teams focus on improving the developer experience across the organization on an ongoing basis by reducing or eliminating repetitive work and making sure developers feedback is heard and implemented. The ultimate goal of platform engineering is to provide a smooth and security-focused path for code to reach production. By automating and simplifying everything from provisioning, observability, and monitoring to compliance and security, the platform removes obstacles and empowers developers to deliver value to customers in less time.

Software supply chain security

Keeping up with evolving security threats and safeguarding the software supply chain requires a security focus be implemented throughout the software development lifecycle. The shift-left security approach aims to solve this by incorporating security focus and practices as early as possible, and making it the responsibility of developers. This inadvertently creates significant cognitive overload and makes it difficult for developers to effectively manage security on top of their core coding duties, proving that simply shifting the burden of security isn't a sustainable solution.

A more effective strategy is the shift-down approach. Instead of asking developers to handle more security tasks, the platform itself provides security by default. This is achieved by building a trusted software supply chain directly into the IDP. The platform automatically enforces a security focus across the entire lifecycle by detecting malicious code, making sure only signed software is deployed, and continuously monitoring for threats. By embedding security from the ground up, this platform-centric model reduces both enterprise risk and the heavy burden on developers.

Navigating the challenges of building applications with AI assistance

While AI-assisted development approaches are accelerating innovation, their rapid adoption by enterprises can introduce significant challenges:

- **The rise of shadow AI**
  Shadow AI refers to the unsanctioned use of public AI coding tools by developers. These tools often bypass corporate security and legal reviews, increasing the risk of intellectual property (IP) leakage, security vulnerabilities, and untraceable code decisions.
- **Technology sprawl and debt**
  Without proper governance, AI tools can rapidly introduce a messy sprawl of unvetted and unsupported libraries. This inconsistent codebase creates long-term technical debt, making applications difficult and costly to maintain over time.
- **Lack of clear ownership and accountability**
  When an AI agent can autonomously write, test, and commit code, the lines of ownership and responsibility are blurred and it raises the question of who is accountable. Without comprehensive policies for human review and approval, there is a risk of creating systems that no one fully understands or can confidently maintain.

To fully realize the benefits of AI-assisted development while managing risks, enterprises must take a structured, accountable, and security-conscious approach. The following strategies can help organizations address these key challenges effectively.

Maintain developer accountability and oversight

AI coding tools are assistants, not replacements for human developers. Developers remain accountable for code and should use AI primarily for tasks such as ideation and prototyping. It is also important to recognize that agentic coding is not yet mature for end-to-end enterprise software development lifecycle due to lack of explainability, traceability, and contextual awareness. Human oversight is crucial until these gaps are addressed, as reinforced by guidance provided by leading cybersecurity bodies, such as the UK National Cyber Security Centre (NCSC).[1]

1 "Guidelines for secure AI system development." National Cyber Security Center, 23 Nov. 2023.

Apply security-focused development lifecycle practices across all phases

AI tools can introduce new risks through unvetted dependencies or insecure code patterns. Enterprises must make sure that guardrails are in place across all stages of the development lifecycle:

- **During coding**, software composition analysis (SCA) should be enforced to detect vulnerable or unapproved open-source packages introduced by AI.
- **During builds**, automated vulnerability scanning should be used to identify known security flaws.
- **During deployment**, security policy enforcement should be applied using Infrastructure as Code (IaC) validation and environment hardening.
- **During runtime**, compliance monitoring and threat detection should be integrated to catch anomalies early.

Adopt platform engineering and a platform-as-a-product mindset

Integrating a platform engineering approach and treating their internal platform as a product that evolves with developer needs and business priorities can help enterprises scale AI responsibly. This strategy provides a standardized, security-focused environment that mitigates the risk of shadow AI.

For platform engineering to support AI-assisted development, it's important to avoid traditional platform-first thinking, which often leads to overbuilding and rigid, upfront designs. Instead, the focus must be on creating flexible, "innovation zone" environments (safeguarded sandboxes) where teams can rapidly test, learn, and adapt. Within these environments, developers and data scientists can safely experiment with new models and workflows without affecting production systems, and with oversight from the platform engineering team. When a new tool or workflow proves its value, it is reviewed and added to the platform's official product backlog. This makes sure that the platform evolves to include capabilities that teams have proven they need—not what leaders guess they might want.

Within this framework, the platform supports modern AI coding methods, including vibe coding and agentic coding.

Adopting AI application development in the enterprise

How organizations integrate AI-assisted application development typically unfolds over three stages, from initial experimentation to full, customized implementation.

Stage 0: The experimentation phase

Here, organizations explore the benefits of AI-assisted development with low commitment. Developers test easy-to-use, cloud-based tools like GitHub Copilot, Cursor, or Lovable to validate productivity gains in generating code, documentation, and tests. This approach is quick and requires minimal upfront investment.

Stage 1: Moving from experimentation to adoption

At this stage, AI becomes a standard part of the development workflow. Companies typically choose from 2 paths:

- **Option A: Expand cloud services.** Continue using hosted AI tools. This is simple to scale but introduces risks like unpredictable subscription costs, data privacy concerns, and potential IP exposure.
- **Option B: Deploy locally.** Bring powerful, open-weight AI models in-house. This requires an upfront investment in hardware and AI inference software but gives the organization complete control over data, security, and long-term costs.

Regardless of the choice, integrating software supply chain security becomes critical at this stage.

Stage 2: Customizing and personalizing for the enterprise

This is the most mature stage, when AI is tailored to the organization's unique environment. It involves:

- **Standardized integration:** AI tools are built into developer templates, making them a default part of every new project.
- **Fine-tuning models:** Using retrieval-augmented generation (RAG), the company trains or augments its locally deployed open-weight AI models on its own codebase.

How Red Hat can help organizations adopt modern development practices

Red Hat provides an open, scalable foundation for building, deploying, and managing enterprise applications across environments, and a comprehensive AI platform that supports the entire lifecycle of AI models, from development to deployment and monitoring.

With solutions such as Red Hat® OpenShift®, Red Hat Advanced Developer Suite, Red Hat AI, and a suite of security offerings, Red Hat positions organizations to efficiently implement modern application development practices—including platform engineering, software supply chain security, and AI-assisted approaches—while maintaining the security and reliability they require.

Platform engineering

To make developers more efficient and productive, platform engineering focuses on creating a robust self-service IDP. The complete IDP is formed by 2 essential layers: the application platform (the technical foundation and infrastructure layer) and the user-friendly internal developer portal (the front end interface). Together this unified ecosystem streamlines the entire software lifecycle, helping developers to work with greater agency and speed.

Red Hat OpenShift: The application platform foundation of the IDP

An IDP centralizes an organization's tools and automates key processes, allowing developers to work with greater speed and efficiency. Red Hat OpenShift serves as the core application platform of the IDP by providing the following capabilities:

- **Consistent development environments**
  Red Hat OpenShift empowers developers through Red Hat OpenShift Dev Spaces and a dedicated Developer perspective in the Red Hat OpenShift web console. This self-service approach removes bottlenecks and frees developers from waiting on operations teams for resources.

  OpenShift Dev Spaces provides consistent, preconfigured development environments that run in a browser, making sure every developer works with the correct tools and dependencies. It also supports integrations for those who prefer their existing IDEs, such as Visual Studio Code, JetBrains IDE (e.g., IntelliJ), Eclipse IDE, and Azure DevOps.

  The Developer perspective in the OpenShift web console offers a simplified graphical interface for deploying, managing, and monitoring applications, making complex cloud-native tasks more accessible.

- **Consistent infrastructure with GitOps**
  Red Hat OpenShift GitOps, which is built on the popular open source project Argo CD, automates infrastructure and application deployment. It uses Git as the single source of truth for all configurations. This approach makes sure that development, staging, and production environments are consistent, which prevents configuration drift and makes deployments highly reliable and repeatable.

- **Efficient software delivery with continuous integration and continuous delivery (CI/CD)**
  Red Hat OpenShift Pipelines, which uses the cloud-native Tekton framework, automates the entire software lifecycle. By creating repeatable pipelines for building, testing, and deploying applications, it removes manual steps and reduces the risk of human error. This leads to faster, safer, and more consistent release cycles.

- **Monitoring and observability**
  Red Hat OpenShift Observability is an all-encompassing solution designed to provide extensive insights into the performance and health of applications and infrastructure running on Red Hat OpenShift—whether deployed in the public cloud, on premise, or at the edge. With OpenTelemetry support and advanced troubleshooting tools, it offers real-time visibility, monitoring, and analysis of system metrics, logs, traces, and events, allowing for rapid diagnosis and resolution of issues before they affect applications or users.

- **Security with role-based access control (RBAC)**
  Security is a core component of the Red Hat OpenShift platform, managed through built-in RBAC. The platform operates on a "deny-by-default" security model, which means that users and teams are only granted the specific permissions they absolutely need to perform their roles. This principle of least privilege is fundamental to minimizing security risks.

It's important to note that Red Hat OpenShift GitOps, Red Hat OpenShift Pipelines, and Red Hat OpenShift Observability are included in subscriptions for most variants of Red Hat OpenShift, with the exception of OpenShift Virtualization Engine (OVE) and OpenShift Kubernetes Engine.

Red Hat Developer Hub: The internal developer portal

An internal developer portal is the central hub where an organization's developers can access all the tools, documentation, and resources they need to build software efficiently. A popular open source framework for building these portals is Backstage; however, it lacks features such as comprehensive support and RBAC, which are required for enterprise use. Red Hat Developer Hub, part of Red Hat Advanced Developer Suite, solves these issues, providing an enterprise-grade portal built on Backstage.

Key benefits of Red Hat Developer Hub include:

- **Enterprise support:** The hub is a fully supported and curated platform, providing a stable foundation and expert assistance for long-term developer productivity.
- **Built-in security:** It includes comprehensive, enterprise-ready RBAC that integrates with Red Hat OpenShift for centralized user management and security.
- **Accelerated development:** A central software catalog makes resources easy to find, while software templates allow developers to start new projects quickly using approved best practices.
- **Dynamic plug-ins:** Teams can safely extend the portal's functionality by adding, updating, or removing Red Hat-verified plug-ins without causing downtime.
- **Streamlined Red Hat OpenShift integration:** The hub is fully integrated with the Red Hat OpenShift platform, providing direct access to essential cloud-native technologies.

Software supply chain security

Protecting the software supply chain is more critical than ever due to growing threats and regulations. To address this, Red Hat offers a suite of security products designed to help teams reduce risk throughout the entire development lifecycle. Key products in this suite include:

- **Red Hat Trusted Portfolio Analyzer**
  This tool scans and analyzes an organization's software components, including custom code, third-party libraries, and open source dependencies. It manages Software Bill of Materials (SBOMs) and identifies known vulnerabilities (CVEs), offering a clear view of an organization's risk profile directly within developer IDEs like Visual Studio Code and IntelliJ without slowing down the workflow.
- **Red Hat Trusted Artifact Signer**
  This solution signs and verifies software artifacts, such as container images, to confirm their authenticity and integrity. This process, which helps meet compliance standards like Supply-chain Levels for Software Artifacts (SLSA), creates a permanent record of all signatures, boosting confidence in a software's origin.

When combined with security features in Red Hat OpenShift, such as the Red Hat Quay container registry and Red Hat Advanced Cluster Security for Kubernetes, these tools provide a complete software supply chain security solution suitable for enterprise use.

AI-assisted application development

Red Hat provides a trusted, open source platform for building and deploying AI applications anywhere, from private datacenters to the public cloud. This approach is built on flexibility, underpinned by both Red Hat technologies and a broad partner ecosystem.

Here's how Red Hat supports enterprise AI adoption:

- **Unified AI platform**
  Red Hat OpenShift AI provides the core infrastructure and tools to build, deploy, and manage AI models consistently across any environment. It includes support for hardware accelerators and the necessary AI libraries to provide performance and scale.

- **Optimized models and tools**
  Red Hat Enterprise Linux® AI offers enterprise-ready gen AI models, along with tools for organizations to fine-tune them using their own data. This allows organizations to create security-focused, custom AI solutions that align with their specific needs.

- **Integrated operations and products**
  Red Hat helps organizations efficiently integrate AI into existing DevOps and MLOps workflows. It also embeds AI directly into its own products. Red Hat Lightspeed, for example, adds AI-powered features into solutions like Red Hat Enterprise Linux, Red Hat OpenShift, and Red Hat Ansible® Automation Platform, to automate tasks and boost user productivity.

Red Hat OpenShift AI

Built on the enterprise-grade foundation of Red Hat OpenShift, Red Hat OpenShift AI provides a consistent and trusted environment for teams to develop, deploy, and manage AI models.

- **Model development**
  Organizations can either develop custom models or use Red Hat's repository of popular, pre-optimized models like Llama, Gemma, and Granite. This flexibility with the integration of various AI/ML libraries helps accelerate development and reduce GPU costs.

- **Model serving and monitoring**
  The platform allows you to deploy models across any hybrid cloud environment and centrally monitor their performance. It uses vLLM as the inference runtime for the hybrid cloud. It also offers model optimization capabilities with tools like LLM Compressor to reduce compute requirements and costs, while preserving model accuracy.

- **Full lifecycle management**
  OpenShift AI integrates MLOps practices to manage the entire AI workflow. This allows organizations to reliably manage a model's lifecycle from the initial experiment all the way through to production.

- **Resource optimization and management**
  The platform is designed to scale efficiently for all AI workloads. It allows teams to effectively share expensive resources like GPUs, projects, and models across the organization.

A standout feature of OpenShift AI is its flexibility. It is platform- and infrastructure-agnostic, giving enterprises the freedom to choose the right hardware, cloud, and tools for their specific AI journey.

Validated and optimized AI models

Red Hat allows organizations to fine-tune open source models with their own private data. This approach is crucial for meeting regulatory requirements, protecting IP, and managing computing resources like GPUs.

Red Hat delivers these capabilities in 2 primary ways:

1. **A library of trusted AI models**
   Red Hat provides a repository of popular, security-focused, and efficient models ready for enterprise use. This includes the IBM family of Granite models, which are fully open source under the Apache 2.0 license and cover both general purpose and code generation tasks. The library also features a collection of third-party models that are validated (tested for performance and accuracy on Red Hat's platform) and optimized (compressed for greater speed and efficiency while maintaining quality).

2. **Community-driven fine-tuning with InstructLab**
   An open source project from IBM and Red Hat, InstructLab allows organizations to enhance and customize LLMs with their specific knowledge. It empowers developers and specialists to contribute improvements, keeping AI development open, collaborative, and tailored to real world needs.

These features are delivered through Red Hat Enterprise Linux AI, which is included with an OpenShift AI subscription.

Consistent devOps and MLOps operations

Successfully deploying AI at scale requires close collaboration between data engineers, data scientists, and developers. The challenge for operations teams is managing the complex lifecycle of creating, deploying, and monitoring AI models.

The solution is MLOps, which applies the proven principles of DevOps to the machine learning lifecycle. Just as DevOps provides a consistent workflow for application code, MLOps does the same for AI models. By utilizing OpenShift GitOps for model deployment and runtime monitoring capabilities provided by Red Hat OpenShift, organizations can build a consistent and standardized MLOps and DevOps delivery mechanism, effectively productionizing their AI efforts.

Conclusion

Through its suite of powerful, security-focused, open hybrid cloud solutions, Red Hat can help your organization adopt—and advance—modern, AI-assisted application development. With Red Hat as your trusted partner, you can confidently navigate the future of application development and unlock its full potential, helping you to accelerate delivery and boost innovation.

Contact a Red Hat sales specialist to learn more about how Red Hat solutions can help.

facebook.com/redhat
twitter.com/RedHat
linkedin.com/company/red-hat

North America	Europe, Middle East, and Africa	Asia Pacific	Latin America
1 888 REDHAT1	00800 7334 2835	+65 6490 4200	+54 11 4329 7300
www.redhat.com	europe@redhat.com	apac@redhat.com	info-latam@redhat.com

redhat.com