# Cloud Development Environments (CDEs) for Platform Engineers

Platform Engineering

Gitpod CODER

# Table of contents

# Executive summary

Today's software engineering landscape is defined by powerful, intersecting trends: the rise of platform engineering as a strategic priority, the accelerating impact of AI-powered development, the growing demands of cybersecurity and compliance, and the ongoing pressure to boost developer productivity. These forces add complexity, but also open the door to immense gains for organizations that can respond effectively. Cloud Development Environments (CDEs) have emerged as a key enabler, due to their intersection with all of these defining mega trends, offering a practical way to turn these challenges into opportunities.

CDEs are preconfigured development environments that automate, secure, and standardize development for software engineering teams in the cloud-native era. They integrate with version control, tooling, and infrastructure to eliminate local setup, accelerate onboarding, and standardize workflows across the development lifecycle.

They are integral to the success of a large number of platform engineering initiatives. By providing consistent environments, CDEs dramatically accelerate developer productivity and developer experience (DevEx).

Though it is their impact on developer productivity that primarily drives their adoption, their ability to solve core organizational challenges, dramatically reduce costs and channel opportunities around AI and security will be a key driver of their adoption in the future. CDEs offer a vital opportunity to proactively strengthen security and compliance posture, with Gartner predicting 40% of organizations in regulated industries will mandate CDES by 2027.

At the same time, CDEs provide the foundation required for responsible AI adoption, enabling the safe integration and scaling of AI code assistants and agents while providing necessary compute resources like GPUs. CDEs provide isolated workspaces that prevent AI tools from accessing sensitive code on local machines, provide audit and compliance trails, and enable organizations to guardrail which AI models and datasets that developers can use. This allows development teams to safely experiment with AI within defined secure boundaries to ensure enterprise level governance while unlocking AI-powered productivity gains.

CDEs thus provide the opportunity to drive measurable business impact across each of the defining trends of the current software engineering era.

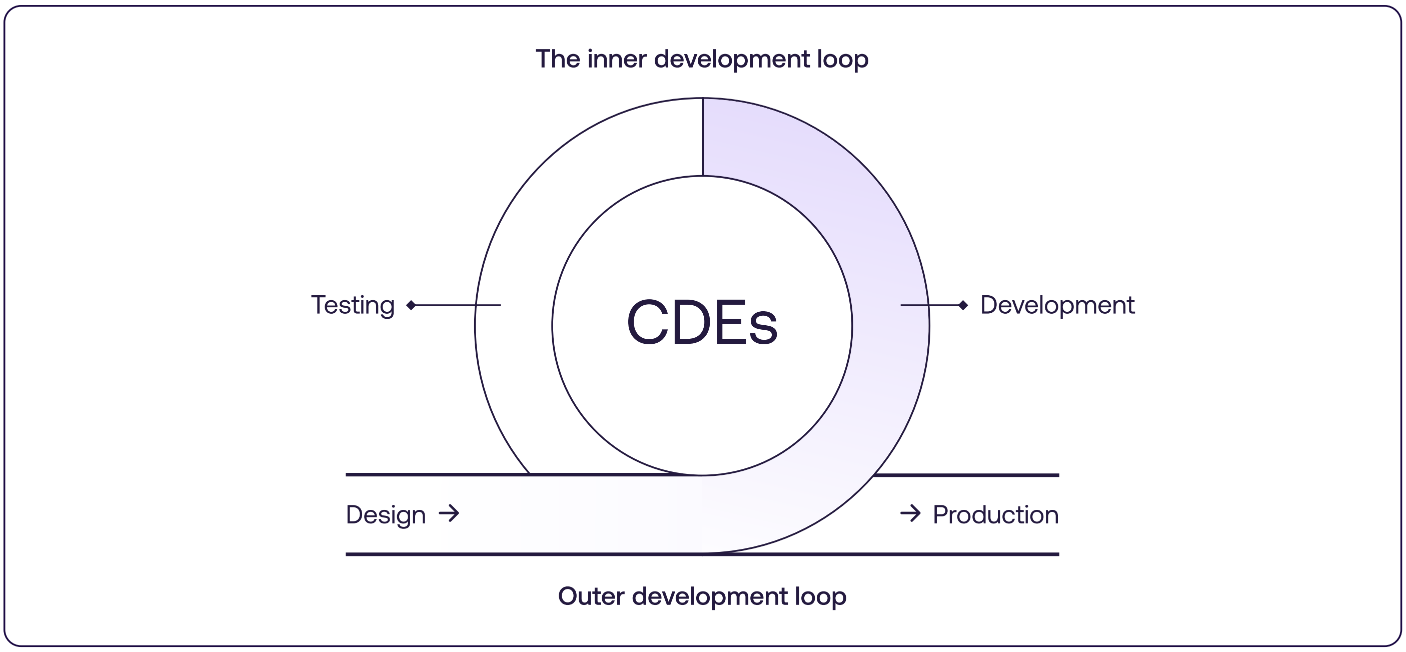
# Why CDEs matter

## Developer productivity

87% of engineering decision makers say developer productivity is a top priority. But turning that priority into measurable gains remains a persistent challenge. Developers often spend a disproportionate amount of time on work unrelated to coding, with estimates suggesting just 30-40% of their time is spent actually writing code. The rest is lost to maintenance tasks, waiting on builds, or troubleshooting environment issues. Bottlenecks like context switching, slow setup, and dependency conflicts disrupt focus and kill momentum. While CI/CD workflows have matured, the local development loop remains largely manual and inconsistent, dragging down iteration speed. This fragmented DevEx can reduce engineering output by 20–50% and is a major factor in team burnout and churn.

Traditional development setups heavily contribute to this friction. Environments differ across teams and individual machines, leading to inconsistency, frustration, and delays. This variability makes it difficult to replicate issues or share workflows, often resulting in the dreaded “works on my machine” problems.

These challenges weaken the developer's inner loop, the critical rapid cycle of coding, testing, and iterating, as developers must constantly troubleshoot inconsistent environments, manage access manually, and recover from configuration issues instead of staying focused on writing and shipping code. When this inner loop is slow and inconsistent, innovation is stifled, feature velocity decreases, and time-to-market is extended.



CDEs offer a practical way to address these challenges. By providing a standardized, preconfigured workspace, they reduce the time developers spend on setup, configuration, and troubleshooting. This helps eliminate common issues like dependency conflicts and entirely removes “works on my machine” issues. With consistent environments across teams, collaboration and debugging become easier and faster. Overall, CDEs create a more stable foundation for development work, helping developers focus more on writing code and less on managing their tools.

## Security concerns

The growing negative impact of unmanaged local development machines becomes even more apparent when focusing on cybersecurity. As cyber threats escalate globally, with AI-powered attack scans reaching 36,000 per second, and the average cost of a data breach rising to $4.88 million, tech leaders increasingly recognize cybersecurity as a strategic business focus. When sensitive source code and credentials are stored on individual laptops, organizations face heightened risks of exfiltration, misconfiguration, and noncompliance. These decentralized setups lack built-in policy enforcement and often operate outside centralized oversight, contributing to “shadow IT” and increasing the attack surface massively.

Organizations have traditionally turned to Virtual Desktop Infrastructure (VDI) to address these security concerns by centralizing desktop environments within the data center. While VDI solutions provide granular access controls and prevent sensitive data from leaving the corporate network, they come with significant overhead in terms of infrastructure management, licensing costs, and performance limitations that can hinder developer productivity.

CDEs represent a more targeted and efficient evolution of this security approach. Rather than virtualizing entire desktop environments, CDEs focus specifically on development workflows while delivering superior security outcomes. They enforce identity-based access, isolate secrets within each workspace, and eliminate the need to store source code locally. Their ephemeral nature limits the impact of vulnerabilities by design, while built-in audit trails, centralized controls, and consistent policy enforcement create a secure-by-default development foundation.

## AI adoption in the enterprise

The rapid advancement of AI, particularly in areas such as code assistants, autonomous software agents, and large language models (LLMs), is fundamentally reshaping software development workflows. While these technologies promise significant productivity gains and innovation, they also introduce complex infrastructure, security, and governance challenges that today's developer environments are not equipped to handle.

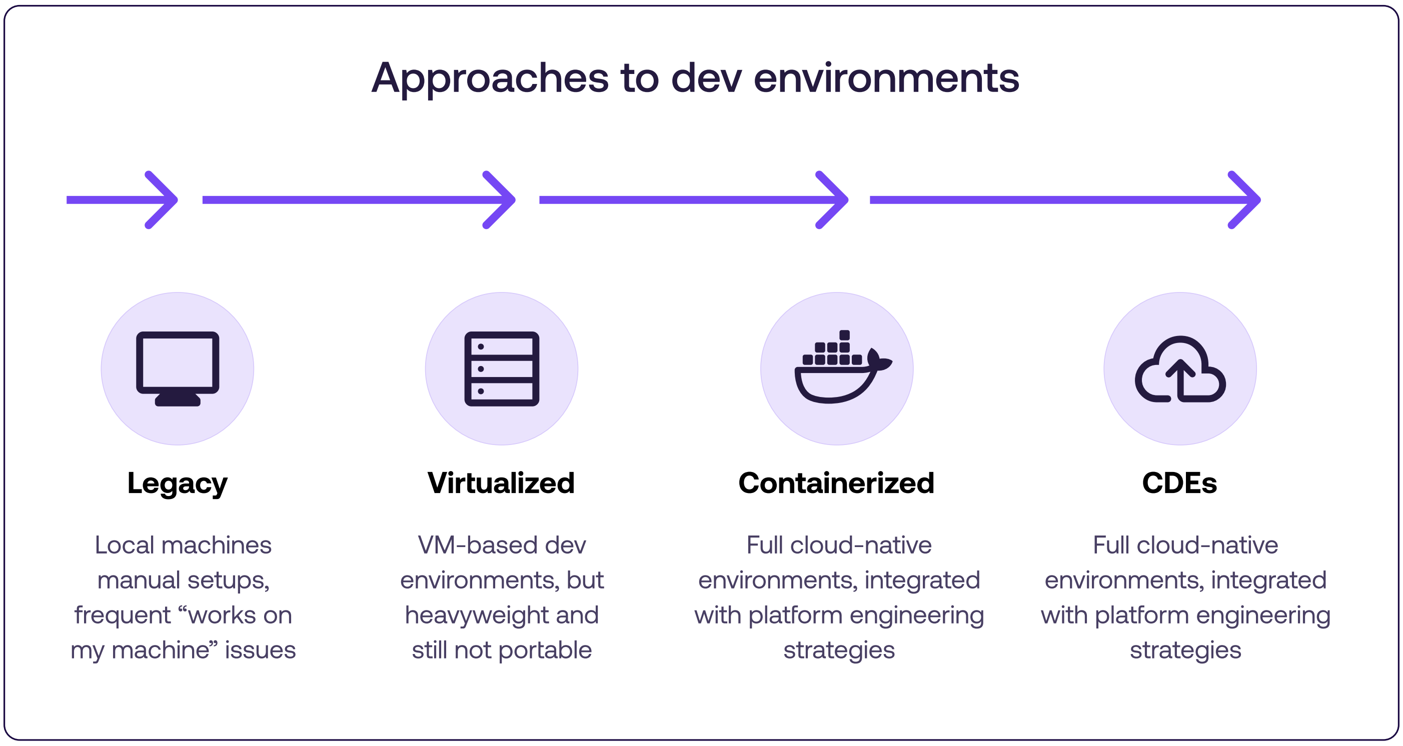
When AI tools and agents operate on unmanaged local machines, organizations face the emergence of "shadow AI" or the uncontrolled usage of AI tools and models operating outside IT governance, similar to how "shadow IT" creates blind security gaps today. Without centralized control, it's nearly impossible to enforce policies, audit usage, or monitor agent behavior. Supporting AI workloads also requires access to GPU-backed compute, difficult and costly to provision consistently across a fleet of developer laptops.

Just as CDEs are foundational for secure software development, they are a **prerequisite** for responsible AI adoption in the enterprise. Within a CDE, organizations can enforce access controls, isolate secrets, and implement guardrails for AI agent actions, mitigating risks like credential leakage or unintended code changes. CDEs also provide secure access to essential AI/ML resources such as GPU-backed compute, approved models, and governed datasets. They ensure consistent environments for AI engineers, bridging skill gaps and enabling reproducible workflows with full auditability of how AI is used throughout development. Recognizing these benefits, many organizations are already **mandating secure CDEs** as the foundation for any internal AI experimentation.

As established in a recent article in the platform engineering community blog, platform engineering and AI are symbiotically driving the future of software. CDEs operate at this critical intersection, offering the infrastructure needed to scale AI usage securely and responsibly across an enterprise. They enable organizations to move beyond local experimentation toward enterprise-grade deployments, with the controls, compliance, and reliability required in regulated industries. By integrating CDEs into Internal Developer Platforms (IDPs), teams can streamline every layerof the AI workflow, from provisioning compute to using code assistants to deploying agents, within a governed, auditable, and secure environment.

# What are CDEs (and what are they not)?

As established above, CDEs provide preconfigured, ready-to-use development environments accessed on demand for human and AI collaboration. These environments are secure, consistent, and equipped with the dependencies, SDKs, security policies, and tooling necessary for software engineering teams in the cloud-native era. CDEs integrate deeply with version control systems, developer tools, infrastructure, and CI/CD pipelines, standardizing workflows across the development lifecycle.



Approaches to dev environments

**Legacy**

Local machines manual setups, frequent “works on my machine” issues

**Virtualized**

VM-based dev environments, but heavyweight and still not portable

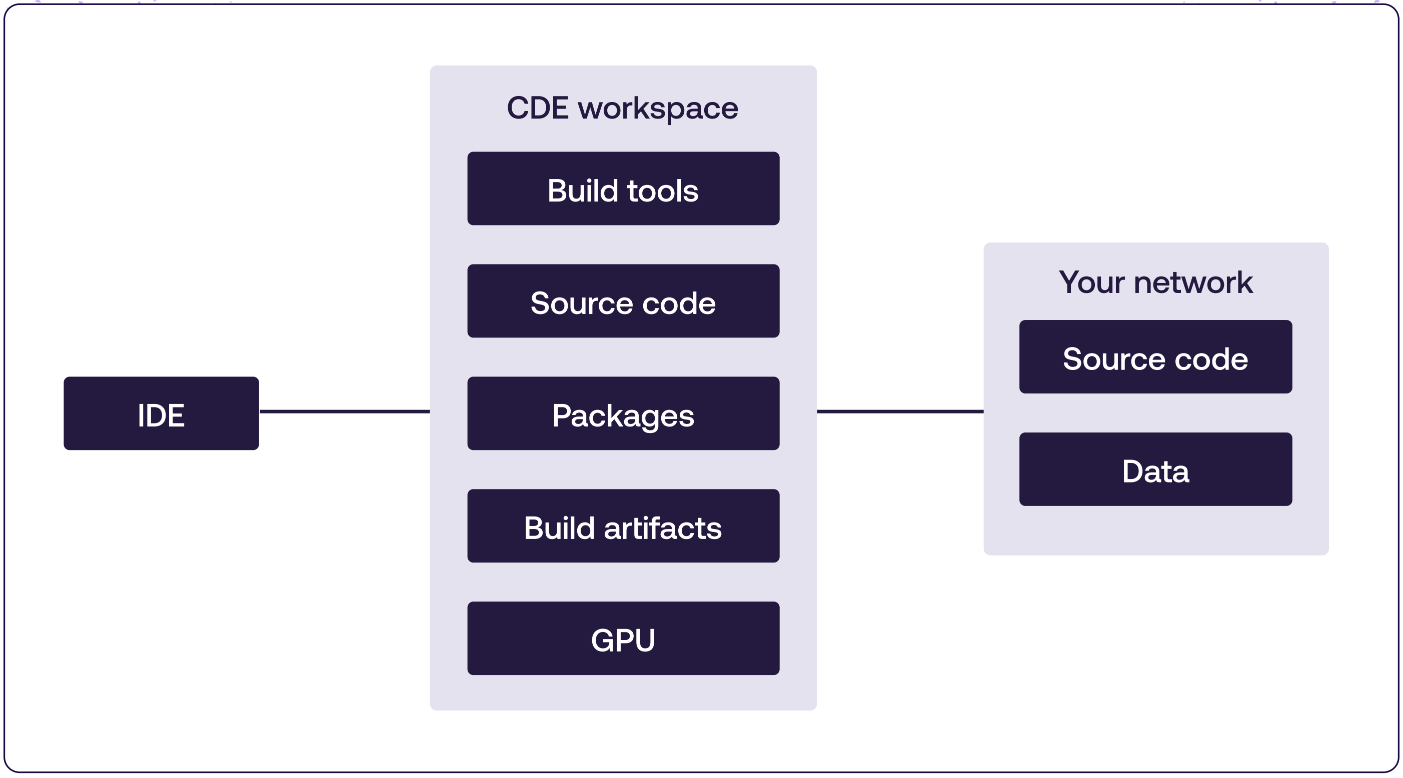
**Containerized**

Full cloud-native environments, integrated with platform engineering strategies

**CDEs**

Full cloud-native environments, integrated with platform engineering strategies

It is crucial to distinguish CDEs from simpler or legacy approaches that they might resemble. A CDE is emphatically not simply a cloud-hosted IDE. While many CDE platforms offer browser-based code editors, they primarily provide a full development environment running in the cloud that developers can connect to using their preferred local IDEs, such as VS Code or IntelliJ. Nor are CDEs just a wrapper around VMs or containers. Although they utilize underlying infrastructure like VMs or containers, CDEs provide orchestration and integration capabilities that tie the environment into organizational workflows, policies, and security frameworks.



Critically, CDEs are not simply general-purpose Virtual Desktop Infrastructure (VDI) solutions. While VDI can provide some similar security benefits like preventing local code download, CDEs are specifically tailored for the software development workflow, offering superior performance, native IDE integration, support for specialized hardware like GPUs, and granular access controls optimized for developers. They also eliminate the poor developer experience and latency often associated with traditional VDI.

Comparing CDEs to traditional development models highlights their unique advantages. Legacy local setups are characterized by manual installations, inconsistent tooling, and frequent "works on my machine" issues caused by dependency conflicts. While virtualized or containerized environments improved portability and isolation, they often introduced their own complexities for developers to manage at scale and could still suffer from performance or integration challenges. CDEs move beyond these limitations by providing a consistent, preconfigured environment that is fully managed by the platform team, abstracting away the underlying complexity of dependencies and infrastructure setup for the developer.

# Where do CDEs fit into your platform engineering initiative?

CDEs serve as a foundational pillar and a critical developer-facing layer within modern platform engineering initiatives. While Internal Developer Platforms (IDPs) typically provide a unified interface for discovering and managing the broader developer ecosystem and "outer loop" workflows, CDEs focus squarely on optimizing the developer's "inner loop", the core activities of coding, testing, and iterating within the development environment itself.

**By translating platform engineering principles directly into the workspace, CDEs give platform teams a powerful lever for:**

### ◆ Standardization and golden paths

Teams can define preconfigured, policy-compliant environments that promote best practices and reduce cognitive load.

### ◆ Developer self-service

Engineers can provision ready-to-use workspaces on demand, eliminating delays and abstracting infrastructure complexity.

### ◆ Reduced support burden

Centralized environment management minimizes one-off tooling requests and support tickets, freeing platform teams to focus on scaling core capabilities.

### ◆ The secure foundation for AI

CDEs provide the controlled, auditable environment necessary for integrating AI-powered development tools while maintaining data governance and preventing inadvertent exposure of proprietary code to external AI services.

Because they're both highly visible and tightly scoped, CDEs are often a smart starting point for Minimum Viable Platform (MVP) initiatives. They deliver fast time-to-value, enable secure experimentation (including with AI), and offer a tangible improvement to developer experience, making them a foundational building block in modern platform strategy.

## Real-world adoption and case studies

Enterprises are deploying CDEs to achieve measurable business outcomes and overcome long-standing developer pain points. Palantir, for example, dramatically accelerated developers' onboarding time to new projects from 15 days to just 1 hour. Across global financial services, a VP of Developer Experience at a F500 investment bank reported that developers' time spent coding increased from less than 5% to 20%, yielding a 224% ROI on labour costs alone. While another successfully reduced project onboarding time to minutes and, crucially for security, centralised developer data off laptops onto their secure infrastructure, achieving over 90% adoption among developers. Plaid's Developer Efficiency Team transitioned to a remote development environment backed by standardized EC2 instances, addressing scaling challenges and improving consistency across development workflows. Beyond finance, several Federal agencies eliminated source code from thousands of decentralized laptops, significantly mitigating data exfiltration risks, and a manufacturing startup reported successfully cutting AWS costs by 90% ($3M to $300K) for developer environments by consolidating workspaces and automating VM shutdowns.

Zooming into one specific case study, Slack, the cloud-based team communication platform, proactively addressed the complexities of their local development workflow, which involved installing dependencies and managing resource-intensive software across operating systems. Leveraging remote development environments on AWS EC2, Slack enabled engineers to provision a fresh, isolated environment on demand, ready within minutes. This move dramatically improved developer experience, standardised setups, eliminated local overhead, and allowed engineers to work on multiple branches concurrently. A particularly impactful result was the reduction in onboarding time for new hires from approximately an hour to mere minutes. By early 2022, over 90% of Slack engineers had adopted this workflow, reporting improved productivity and reduced performance burden on local machines.

## Analyst validation

At the same time, analyst validation underscores the benefits of CDEs. Gartner pins CDEs in the Hype Cycle for Platform Engineering 2025, identifying them as a rapidly emerging solution for enabling platform engineering, developer enablement, and secure AI adoption. Meanwhile, the DORA report highlights how traditional local development remains a bottleneck even in otherwise mature engineering organizations. Their findings show that AI-augmented workflows thrive in standardized, cloud-based environments like CDEs, which help mitigate risks like credential leakage and untraceable model activity. Together, these insights confirm what adopters have already proven: CDEs deliver tangible value across productivity, security, and innovation.

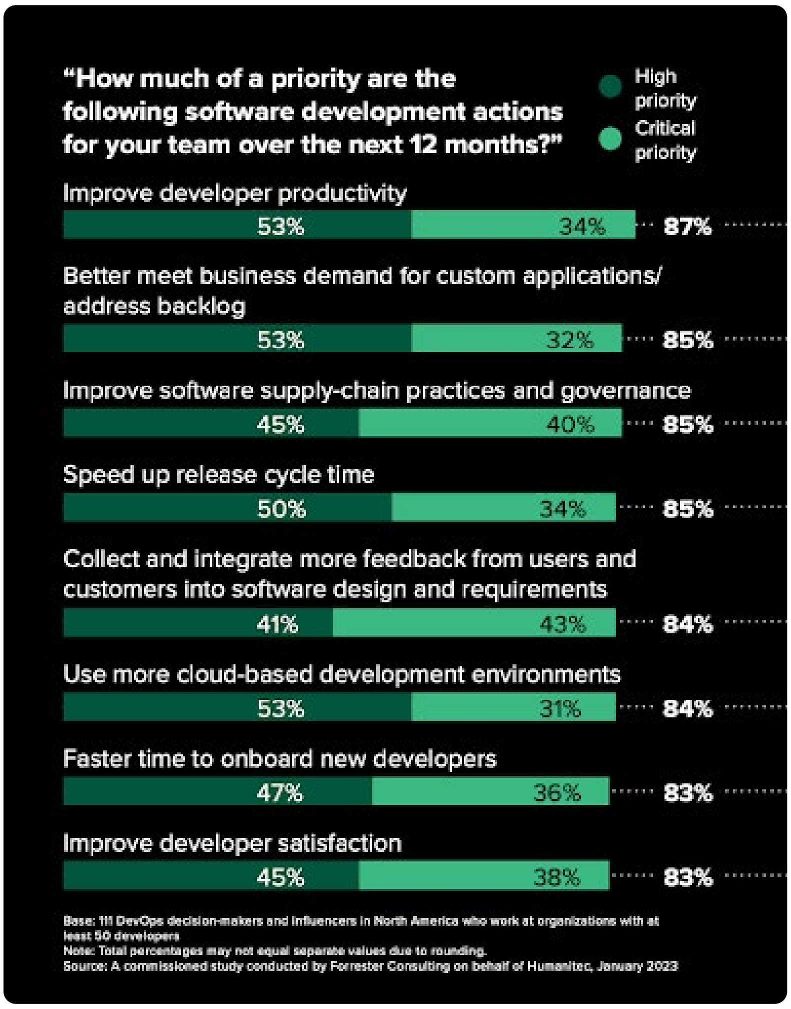
# The business case for CDEs

Turning the operational and security benefits of CDEs into a clear financial case is key to driving strategic adoption and showing they're more than just a developer tool, they're a real investment in business value. CDEs offer a clearly quantifiable economic rationale across multiple cost categories, directly impacting the bottom line through enhanced productivity, infrastructure optimization, and improved risk management.

## Why do enterprises adopt CDEs

- Regulated industries need a way to securely roll out AI.
- Security is pushing for VDI, but it's slowing developers down.
- Inconsistent environments lead to bugs, frustration, and delayed GTM.
- Onboarding new developers still takes days, or even weeks, and local hardware failures leave developers unable to access code.
- Keeping dev environments consistent requires full-time effort.
- You need to guarantee code stays inside secure infrastructure.
- Collaboration breaks down on complex, multi-service projects.
- You need to evaluate or deploy autonomous coding agents.

Quantifiable benefits for CDEs are most evident in developer productivity and onboarding efficiency. Developers notoriously spend a large portion of their time on environment setup and maintenance, not coding. CDEs mitigate this, potentially saving 10-20% of developer time per week, or up to 10 hours. This reclaimed time translates directly into increased feature velocity and faster time-to-market. Further, onboarding new developers, mitigating local hardware failures, or enabling transitions between projects is dramatically accelerated. For organizations with significant hiring or internal mobility, the cost savings from accelerated ramp-up are immense. Improved developer experience also boosts satisfaction and retention, reducing the high costs associated with employee churn. These gains flow directly into the developer productivity cost category.



Forrester Opportunity Snapshot: A Custom Study Commissioned By Humanitec, February 2023

Beyond direct developer efficiency, CDEs drive tangible cost optimization in infrastructure and IT support, alongside critical savings in security and compliance. Ephemeral environments that automatically scale or shut down based on usage significantly reduce cloud infrastructure waste and costs. This can lead to savings, such as a 90% reduction in VM spend highlighted above. As established, CDEs also present a superior, developer-optimized alternative to expensive, general-purpose Virtual Desktop Infrastructure (VDI), potentially yielding a 50% reduction in VDI-related costs while improving user experience. Centralized controls, audit trails, and compliance readiness built into CDEs reduce ongoing security and compliance costs.

When you add up the impact from saving valuable developer time and speeding up onboarding, to cutting infrastructure costs and reducing security and AI-related risks, the business case for CDEs is clear. They're not just tools; they're a strategic investment that helps teams move faster, stay secure, and build for the future. For any organization looking to boost engineering output, streamline operations, and tackle modern software challenges with confidence, CDEs are a clear strategic opportunity.

## ROI calculation

The benefits of CDEs as highlighted in this report might be clear to see. However, it is challenging to secure buy-in with decision makers without quantifiable measures. This can often be difficult for many teams to articulate to leadership however, as teams might focus on technical language, rather than the business narrative that might more effectively convince decision makers. Here is an example of how you could convert technical language around developer productivity into a business narrative when pitching a buy case for a CDE:

### 01 Highlight the cost of troubleshooting time

Let's break down what it actually costs to lose 5 hours a week to setup issues or environment bugs. Developers are among the highest-paid employees, and if they're only spending around 70% of their time coding, each hour of lost productivity ends up costing about $80 (after factoring in benefits, taxes, and time off). That adds up to around $1,500 per developer per month or $1.8 million per year for a team of 100. And that's not even counting the cost of delayed launches or missed revenue opportunities.

### 02 Onboarding time is expensive

As shown above, several real-world examples show that CDEs can reduce onboarding time from days or weeks to just hours. If your current onboarding process takes 10 days, it's realistic to reduce that to just 2 or 3. That's a 60-hour time savings per hire. For a company onboarding 30 developers annually, including hires, contractors, and team transfers, that adds up to over $140,000 in lost time due to inefficient onboarding alone.

### 03 The bigger picture

When you zoom out, the combined impact is massive. A 100-person engineering org with 30 annual joiners could reclaim over 24,000 hours a year by using a CDE. That's the equivalent of a 19% increase in engineering capacity or roughly $2 million in added value every year.

## Conclusion

CDEs are a foundational layer in modern software delivery, not just as a tool for individual developers, but as strategic infrastructure that aligns tightly with platform engineering, security and AI goals. By standardizing workspaces and abstracting away complexity, CDEs help organizations streamline the entire development lifecycle, unlocking measurable improvements in each of these defining mega trends.

Because they sit at the heart of the developer's "inner loop", the day-to-day flow of writing, testing, and iterating, CDEs offer a unique opportunity to drive impact where it matters most. For platform teams, they translate core platform engineering principles into tangible outcomes: enabling developer self-service, enforcing golden paths by default, and dramatically reducing the operational burden of managing local environments.

At the same time, CDEs create the secure and governable foundation needed to adopt AI coding assistants and agents at scale, while ensuring that sensitive code and intellectual property remain within controlled infrastructure.

As platform engineering matures into a key business strategy, adopting CDEs becomes a clear strategic step toward building a scalable, secure, and future-ready engineering organization.

Want to get started? Join our free course: Cloud Development Environments for Platform Engineers, created alongside leading CDE providers Gitpod and Coder to cover everything from CDE fundamentals to real-world case studies and common pitfalls to avoid.

## Resources

**This whitepaper draws upon extensive research, expert insights, and enterprise case studies. For more information on Cloud Development Environments and platform engineering best practices, explore the following resources:**

Anderson, Tim. Why Stripe embraced remote coding – and fixed Ruby with Sorbet. DevClass.

Badmaev, Andrey. How to build a business case for purchasing a cloud development environment. Gitpod Blog.

Bichard, Lou. Why CDEs should be prioritized before IDPs. Gitpod Blog.

Bichard, Lou. Make developer experience in regulated industries fun again. Gitpod Blog.

Coder. Palantir's Journey to the Cloud: Leveraging Coder for Improved Development. Coder Success Stories.

Coder. Cloud Development Environment Maturity Model. Coder Reports.

DeBellis, Derek, et al. Accelerate State of DevOps 2024: A Decade with DORA. Google Cloud

Galante, Luca. AI and Platform Engineering. Platform Engineering Blog.

Gartner. Hype Cycle for Platform Engineering, 2024.

Gartner. Market Guide for Cloud Development Environments.

Gartner. A Software Engineering Leader's Guide to Improving Developer Experience.

George, Sylvestor. Remote Development at Slack. Slack Engineering Blog.

Forrester Opportunity Snapshot: A Custom Study Commissioned By Humanitec.

FortiGuard Labs. 2025 Global Threat Landscape Report. Fortinet.

IBM Security and Ponemon Institute. Cost of a Data Breach Report 2024. IBM.

Kelly, Don. The Journey to Cloud Development: How Shopify Went All-in on Spin. Shopify Engineering Blog.

Moyal, Talia. Driving platform adoption through development environments. Gitpod Blog.

Moyal, Talia. The AI security gap: a CTOs & CISOs guide to making their first AI investment. Gitpod Blog.

Paquette, Marc. What Makes a Great CDE for Enterprise Users 2025. Coder Blog.
Platform Engineering Community. State of Platform Engineering Report, Volume 3.

Potter, Ben. Building the IDE Golden Path. PlatformCon 2024 Talk. Coder.

Puri, Arjun; Dunne, Jarrod; & Dashevskii, Oleg. Scaling Plaid's internal developer experience with a remote development environment. Plaid Engineering Blog.

Stack Overflow. Stack Overflow Developer Survey 2023. Stack Overflow.

Sundaram, Senthil. Supercharging Remote Development with the Cloud. Rippling Engineering Blog.

Whiteley, Rob. The AI-Native Developer Stack: Rethinking Code From Ideation to Production in Minutes. Coder Blog.