Coder

carahsoft.

Introducing CDEs to your enterprise

Report

Thank you for downloading this Coder report. Carahsoft is the master aggregator for Coder DevSecOps solutions.

To learn how to take the next step toward acquiring Coder's solutions, please check out the following resources and information:

For additional resources:
carah.io/CoderResources

For upcoming events:
carah.io/CoderEvents

For additional Coder solutions:
carah.io/CoderSolutions

For additional DevSecOps solutions:
carah.io/CoderDevSecOps

To set up a meeting:
Coder@carahsoft.com
877-742-8468

Coder

Introducing CDEs to your enterprise

Improve onboarding, developer happiness, and outcomes with a managed cloud development environment

JANUARY 2024

Introduction

Hobbyists, students, and small- to medium-sized businesses are well-acquainted with cloud development environments (CDEs). They use platforms like Replit or StackBlitz to get the flexibility and simplicity of using a cloud-connected or cloud-based IDE. They don't have to deal with the complexities and delays of configuring software and hardware before they can even edit the first file

On the other hand, enterprise development teams use popular IDEs like Microsoft VS Code or JetBrains IntelliJ IDEA, but don't make the leap to CDE. Of course, enterprise developers can also benefit from adopting a CDE. And with opportunities for economies of scale, large enterprise teams can get bigger benefits than smaller teams.

But there isn't a lot of guidance out there for rolling out a managed CDE platform at enterprise scale. Coder is in a unique place for CDEs in the enterprise. We have helped roll out CDEs to thousands of developers across our vast portfolio of customers from finance to industrial drones.

Here's a best practices guide for how to measure value and roll out a CDE in your enterprise. Of course, every enterprise has different needs, so we've drawn from the different use cases we've worked with.

Step 1: Find the shadow CDEs

Acknowledge the existing “shadow CDEs” in your enterprise. Just like “shadow” IT, shadow CDEs are the disparate, unofficial CDEs that different teams have adopted out of necessity. There are infrastructure challenges that developers needed to overcome. What are those challenges?

Some of the consequences of shadow CDEs:

- High costs of continually running compute and storage.
- No economy of scale for on-maintaining, troubleshooting, or cost-saving.

Some shadow CDE setups you’ll probably encounter:

- Virtual machine (VM) or container accessed via SSH
- VM or container accessed via a browser-based IDE like code-server
- Coder open source (as opposed to Coder Enterprise)

Step 2: Measure the current situation

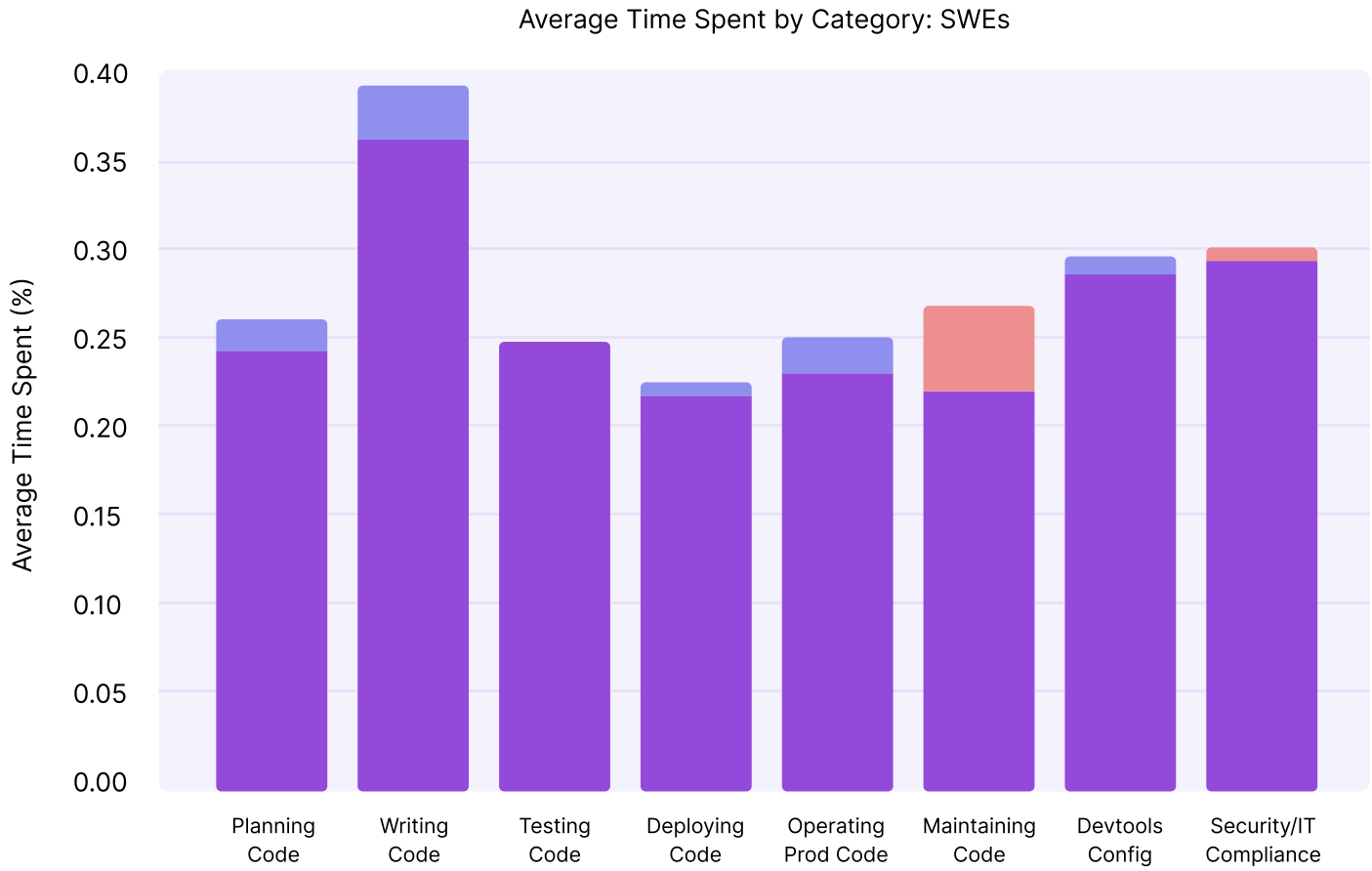
With an understanding of the tooling your developers are using, you’re ready to measure what they do today. Send a survey to your organization’s developers with questions like these:

- What percentage of your time do you spend on actual development vs administrative activities?
- How often do you set up a new development environment?
- How long does it take you to set up your development environment?
- How long do you spend waiting on local clones, builds, and tests?
- How satisfied are you with the speed and flexibility of your development environment?
- Do you work remotely?
- Where is your primary development environment, local or cloud?
- Which tools and platforms do you use?
- Which tools and platforms have you evaluated but didn’t adopt?
- Is your team organized by product, division, tech stack, client, or geography?

Step 3: Build a business case

You've collected pivotal metrics from your teams. Now you can use this data to align with your CTO or developer experience leaders to carve out the business case for an enterprise-wide, managed CDE platform. Take a look below at some business cases that our customers have successfully pitched.

You'll surely uncover low-hanging fruit that make an obvious case for an enterprise-wide managed CDE platform. For example, we conducted a survey among 223 developers, devops, managers, and platform engineers. They report spending as much or more time on configuration and compliance than development activities like planning, coding, testing, devops, and maintenance.



There are advantages to adopting a single CDE platform. But in the previous steps you might have uncovered smaller, but still promising, opportunities that don't necessarily require full adoption of an official CDE. For example, one of our customers achieved improvements with a hybrid approach to using Coder. For developing hotfixes, Coder was ideal. For developing on the main branch, they preferred working locally on their laptops.

BUSINESS CASE:

Business Case: Faster Developer Onboarding

Shorten the time it takes a developer to make their first commit when they join a team. CDEs expedite the initial journey from a new hire to a contributor, significantly reducing the time to the first commit.

Key metric: Time to first commit

BUSINESS CASE:

Business Case: Enhanced Developer Experience

Accelerate the inner development loop, courtesy of superior compute resources, network access, and lower latency. These advantages are a hallmark of CDEs. Also, CDEs unlock a suite of tools erstwhile unavailable in traditional setups.

Key metric:

- Build time
- Repository clone time
- File indexing time
- Number of steps to set up an environment

BUSINESS CASE:

Business Case: Security And IT Compliance

Reduce the time spent on complying with security and IT. And at the same time, get more consistent and higher compliance. A managed CDE platform consolidates these policies in one place and lets the experts, security and IT teams, manage them. Enterprise CDE platforms, Coder included, follow industry-standard security practices and give administrators useful tools like access control and auditing

Key metric:

- Time spent on security and IT compliance
- Compliance rates

BUSINESS CASE:

Business Case: Digital Transformation

CDEs are a conduit for cloud-native development best practices. Adopting a CDE can encourage developers to leverage the advantages of the cloud to build their applications.

Key metric:

- Percent of cloud-native developers
- Percent of applications moved to the cloud
- Percentage of customer-facing services in the cloud

BUSINESS CASE:

Business Case: Legacy System Access

CDEs provide consistent environments from which you can standardize access to legacy systems. Remove antiquated development environments such as standalone VMs and improve developer experience when working on legacy apps.

Closely related to this is developer context switching. Assigning a developer to another project means they have to acquaint themselves with the project again. We've heard stories of developers spending up to a week just to "context switch" to another project. This adds up for several projects per year.

Key metric:

- Time to first access
- Developer context switching

Step 4: Pick the alpha users

Identify a cadre of alpha users. These users are the first ship to explore the CDE waters. A pilot phase is essential for garnering initial feedback and making requisite adjustments before a broader roll-out.

For a successful alpha, it's best to keep it simple. This will make it easier for you to support complex setups later.

These aren't requirements, but here are some green flags for choosing alpha users:

- Web app developers (frontend or backend) working on applications with minimal external dependencies such as networks, filesystems or tooling.
- Developers using Microsoft VS Code: This IDE embraces the cloud, so it has fewer obstacles for your pilot project.
- Developers who already use CDEs or devcontainers.
- Cutting edge tinkerers who may want to use a new language (like Rust) or technology (like Docker) or work on side-projects.

For the pilot project, we recommend that you avoid users in these situations, if possible:

- Windows, mobile, embedded developers: Environments for GUI development can be laggy or tricky.
- Developers with setups that require conventional desktop IDEs like Eclipse.

Step 5: Measure your market

You've already gathered metrics and selected alpha users. Now you need to measure the entire market of developers in your enterprise and their requirements.

Run a survey to understand how many developers would potentially use a CDE. From what we've seen, about 80% of developers have use cases that would work with a CDE. We've even seen Windows, mobile, and embedded development work well with some tuning.

Step 6: Plan your development infrastructure

Work with engineering leaders to understand the infrastructure requirements around the largest, or most beneficial, codebases to run on CDEs.

We strongly believe enterprise CDEs should be self hosted, either on-prem or in a hyperscaler cloud provider like AWS, Azure, or GCP. Enterprise codebases have complex security and infrastructure requirements, which make third party SaaS solutions a non-starter.

Even enterprises with substantial on-prem data centers should still consider running CDEs both on-prem and on a hyperscaler cloud provider. We have a couple of customers who are benefiting from this combined approach:

- For many development teams, access to the on-prem network in their CDE will be critical for filesystem access, network access, databases, datasets, etc.
- The cloud helps developers build up new skills and can help with a digital transformation effort.

Understand whether you need to run CDEs on Kubernetes (each CDE is a Kubernetes deployment/volume) or VMs (each CDE is a new VM). Reassure your CTO or developer experience leader that you can always do both, but it's best to start simple with targeted use cases.

Step 7: Prepare for the pilot project

Prepare for your pilot project with these tasks:

- Setting up cloud infrastructure
- Creating the base and other images
- Supporting a seamless transition for the alpha group

You'll use the insights that you glean from this step to refine your CDE setup and ensure that it meets developer's needs and enterprise goals.

Keep in mind that teams will undoubtedly discover use cases down the road that require different infrastructure, dependencies, or often both. It's great to roll out slowly, with focus, but keep this in mind when you are shopping for a platform for CDEs

Tip: Coder lets platform teams bring their own infrastructure (templates) while allowing developers to bring their own project requirements (images or devcontainers).

If you're using devcontainers, consider envbuilder. Envbuilder is an open source project for handling CDE infrastructure on Kubernetes. Developers can bring their own dependencies via their git repository. For example, they might need Java 11 and Python 3 in their environment.

Task: Prepare The Infrastructure

Specify a CDE that fits into the existing infrastructure patterns and paradigms of your enterprise. Take advantage of your cloud development platform's templating features. For example, Coder uses Terraform-based templates. Coder also runs a scale test and health checks to ensure everything is running properly.

Task: Make Images For Development Environments

Make a base image. This is the lowest common denominator that all, or most, of your enterprise teams can start from. Include any standard tools (e.g. git) and internal necessities, like homegrown utilities, that developers expect in any workspace.

Make 5-6 golden path images. A golden path is a task-specific development environment for users to tinker with a project for a specific technology or purpose, like Spring Boot or Angular. Consider making a "kitchen sink" image: Many CDE platforms, like Gitpod and GitHub Codespaces, have a kitchen sink image with many languages and tools pre-installed. There are benefits to this as it can be a "one size fits most". It's often good to do this if you already manage developer desktops, like legacy VDI, and understand requirements

Task: Add Documentation

Write getting started guides and use cases. That means a FAQ, an internal wiki, Slack, and so on. Also, make sure there's a clear point of contact. Give your developers a way to reach out if they need help!

Step 8: Run the pilot project

Get your alpha group started by sending out log-in credentials. It's important to get insights that show who is using it, and what IDEs, languages, and projects they work on. You can gather feedback by identifying end users and power users. Work closely with your teams to ensure that the pilot project is working for their use cases.

Continue gathering metrics like time to onboarding, build times, repo clone times, and adoption. Use these measurements to confirm your decisions and adapt your CDE workflows to your alpha users.

Step 9: Roll out to the masses

This step is about fine-tuning the setup, monitoring the usage, and ensuring a feedback loop for continuous improvement. Post-alpha, the journey towards mass adoption entails aligning with engineering leaders for onboarding modifications, expanding use cases, and infusing more automation to enhance the allure of CDEs.

Keep your users informed and leverage word of mouth. Email current and prospective users with carrots and metrics that encourage CDE adoption.

From the previous step, you've probably discovered new opportunities. Here are some of the improvements our customers have made in their enterprises:

- Introduce new technologies and new languages to your organization, previously restricted due to security or IT concerns.
- Use envbuilder to run devcontainers. We've seen our customers reduce support tickets while observing massive growth of CDEs inside their organization.
- Some teams will want to bring their own infrastructure. Luckily, your CDE platform lets you do this.
- Consider introducing functionality for collaboration, monitoring, and automation, like Slack Me, NFS shares, JFrog, and Grafana.
- Re-consider introducing another compute type, like VMs or containers.
- Don't ignore other infrastructure targets. You might find that a mix of on-prem and cloud will let you support more use cases. CDEs are not one-size-fits all.

Step 10: Measure the adoption

The goal here is, of course, to make sure you're on the path to success and make the necessary corrections to stay on that path. It's also the way to identify quick wins and sustainable productivity gains, like speeding up the developer's inner loop and reducing build times. You can also gain insights to understand what use cases best fit your CDEs.

More opportunities with platform engineering

The move to DevOps for the past few years gives developers more responsibilities, "You build it, you own it." Platform engineering is the next phase of DevOps where developers who were previously overwhelmed with complexity, especially those who are unfamiliar with infrastructure concepts, can still own and manage their applications in the cloud.

The added benefit is that platform engineering can happen sooner in the DevOps workflow. While there are tools further along in the pipeline, like security scans in CI, best practices from day 1 reduce time waiting for a build to deploy and time switching to secure languages/ artifacts.

Like platform engineering, CDEs open compelling opportunities to give enterprise developers best-practice environments with many of the complexities of their environment abstracted away from them.

Examples:

- AI tools, like GitHub Copilot, can be pre-installed to help developers write less cumbersome code.
- Artifact management tools, like JFrog Artifactory, ensure developers download from a trusted source.

Another benefit is that formalizing CDEs forces you to ask difficult questions, such as "how are developers spending their time?" This likely unearths complex IT policies and other environmental challenges that ultimately hold developers back.

Looking forward, we expect that CDE platforms will be further integrated with AI/LLMs to offer engineers and platform teams, not only when writing new code, but by providing unique insights and metrics around bottlenecks and optimizations within the development environment. We believe CDEs will provide the metrics that can help engineering teams justify a major refactor, invest in new infrastructure, or even identify quick wins in build scripts that reduce waiting time.

The future of CDEs

In the realm of software development, the clamor for enhanced operational efficiencies, shorter onboarding times, and enriched developer experience has motivated the rise of Cloud Development Environments (CDEs). The venture into the enterprise frontier is not only feasible but laden with a myriad of benefits waiting to be unearthed.

At the end of the day, CDEs are just one of many tools to help engineers write clean code, avoid distractions, and reduce long feedback loops. With the right leadership, the CDE presents a unique opportunity compared to other developer experience tools. After all, CDEs help you understand and support developers where they spend the majority of their time: their IDE.

Coder stands at the frontier of this future, spearheading the enterprise migration to CDEs with a proven track record.