Coder

Cloud Development Environment Maturity Model

Table of contents

Cloud Development Environment Maturity Model

Software development is iterative and constantly evolving. To modernize a development environment an organization must:

- Understand the current state and set an overarching goal
- Align technology, developer experience, and processes
- Define incremental milestones to achieve the goal

This document defines how organizations can use a Cloud Development Environment to achieve those goals. CDEs are a rapidly growing technology; the segment is at the top of the Innovation Trigger phase of the Gartner Hype Cycle for Platform Engineering, 2024. A mature CDE drives innovation, efficiency, governance, and growth through a consistent and low-friction developer experience. Achieving maturity requires a thorough understanding of the platform stages and a dedication to refining practices and strategies tailored to the objectives of the organization.

What is a Cloud Development Environment?

A CDE is a platform that provides developers with:

- Predefined workspaces that are decoupled from their physical workstation
- A comprehensive and consistent set of development tooling
- High performance remote resources to develop and build on

A mature CDE embodies the tactical synergies of the Development and Platform Engineering teams. It enables teams to self-govern enterprise-wide development practices and utilize stable, scalable, and secure cloud resources. Product and leadership stakeholders can leverage a CDE to improve governance and drive the strategic direction of the organization. The flexible and iterative nature of a CDE fulfills today's requirements and future proofs the organization's development practices.

Target audience

CTOs, VPs of Engineering

Transform business capabilities into scalable products

Engineering Managers, Development Teams

Streamline innovation-focused development processes

Enterprise Architects, Platform Engineers

Optimize Developer Experience with a frictionless multi-cloud approach.

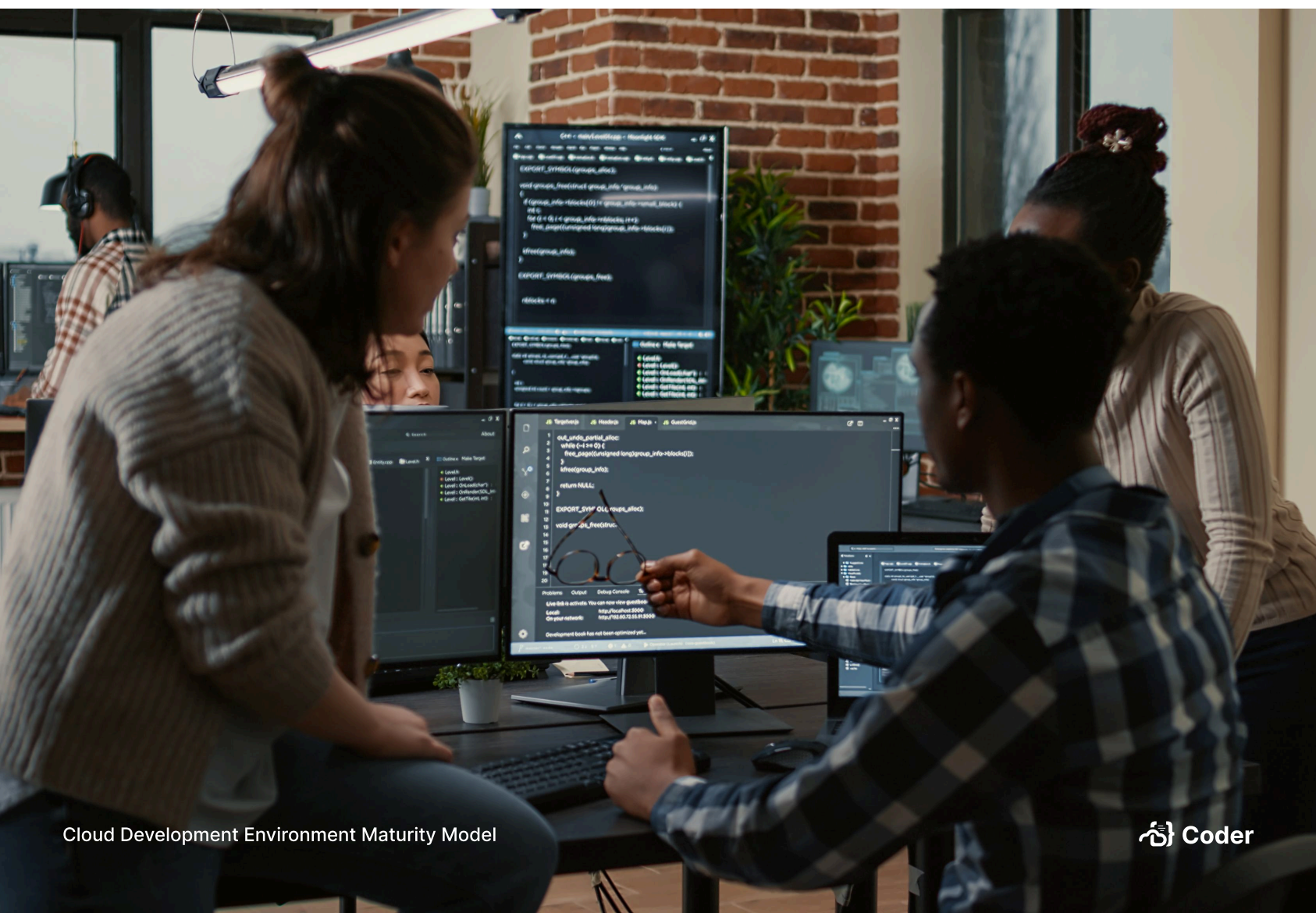
Security Engineers

Ensure adherence to security protocols and establish guardrails

Cloud Development Environment Maturity Model

This model is based on the Software Engineering Institute's Capability Maturity Model Integration. It addresses the ideas and questions many organizations have when considering, deploying, or expanding a CDE. It outlines five distinct stages of maturity and identifies the tooling, infrastructure, people, and process dimensions for each. It is imperative to understand that these dimensions are interconnected; progress – or lack thereof – in one dimension affects neighboring dimensions.

It also identifies risks with stagnation and offers guidance to advance through each stage. It provides a structured framework to assess current state and a roadmap to plan future state of a CDE. This comprehensive, iterative, and incremental approach highlights specific areas of focus for strategic growth and improved competitive advantage. The model is defined by the following stages:



Cloud Development Environment Maturity Model

Stage	Tooling	Infrastructure	People	Process
STAGE 4 Optimized	Development teams build and maintain their own templates	Infrastructure chosen by developer and additional platform teams	Local development is now an edge case, teams using templates to build out development infrastructure	Platform engineering provides pipeline for templates, infrastructure, and maintains enterprise integrations
STAGE 3 Refined	Development teams fulfill own requirements, additional use cases added	Tiered, multi-cloud, multi-platform infrastructure controlled by Platform Engineering Team	CDE only teams approaching the majority	Platform engineering still maintains use cases, but focuses on scalability
STAGE 2 Defined	Standardized CDE conversion patterns emerge. Platform team maintains developer requirements.	Software defined, scalable cloud. Integrated with enterprise IDP, monitoring, security.	Multiple teams CDE only	Production grade CDE deployed and managed by enterprise platform engineering
STAGE 1 Foundational	Pilot use cases enabled on CDE	Minimally viable, supported infrastructure	Pilot group CDE only	Deploy and govern a pilot CDE, iterate and refine.
STAGE 0 Ad-hoc	No standards	Laptop or shadow IT	Inconsistent developer experience	Practices, automation and abstraction vary from team to team

STAGE 0

Ad-hoc

This is the widest ranging stage of this model since it encompasses all teams at a pre-CDE organization. At worst teams are mired in a non-standardized and poorly controlled environment. At best they have begun to abstract their environments, standardize some processes, and – most importantly – are aware of the organization-wide issues of this stage. The goal of this stage is to catalog existing processes and identify a pilot group to work on the Foundational stage. The Ad-hoc stage is generally characterized by these dimensions:

Tooling

Each team chooses foundational components such as libraries, frameworks, and compilers as they see fit. Leads to inconsistent builds, bugs, and regressions.

Infrastructure

Coding and builds take place on laptops, workstations, and unsanctioned virtual machines. Permission issues, access issues and hardware contention waste valuable time. Non-standard architectures lead to hardware related bugs. Unsanctioned resources are risky and expensive.

People

Manual onboarding and enablement is lengthy and error prone, time to first commit ranges from days to weeks. Morale and productivity suffer from delays, lost work, bugs, and regressions due to an inconsistent environment. Developer experience rating ranges from extremely poor to good: teams with best practices, abstraction, and some automation have the best DevEx in this phase.

Process

Ranges from completely manual to team-based best practices. On the low end of this range, developers are selecting, installing, and manually configuring each tool individually. Inconsistently configured or patched environments proliferate “works on my machine” issues. Productivity halts when developers perform maintenance tasks. On the high end of the range, there are team-based abstractions in the form of golden images plus automation in the form of scripts and declarative configurations.

Adapting rating models like Net Promoter Score (NPS) and System Usability Scale (SUS) to measure Developer Experience (DevEx) allows organizations to assess and improve the tools, environments, and processes that developers interact with daily.

Net Promoter Score (NPS):

Ask developers how likely they are to recommend their development environment or tools to others. This captures the overall satisfaction and loyalty developers feel towards their work environment. By analyzing the feedback from promoters, passives, and detractors, organizations can identify strengths and areas for improvement in the developer experience.

System Usability Scale (SUS):

Developers rate statements about the ease of use, efficiency, and intuitiveness of the tools they use. The results provide a quantifiable measure of the usability of the development environment, helping to identify pain points and prioritize enhancements.

By leveraging these adapted rating models, organizations can gain valuable insights into the developer experience, driving continuous improvements that enhance productivity, satisfaction, and overall development efficiency.

This scenario is not ideal for any organization that depends on its developers. It ranges from poorly controlled and unpredictable to abstracted but inconsistent.

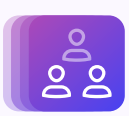
The risks of stagnation are severe:

- High rate of developer attrition
- Longer development cycles and increased costs
- Negative effects on innovation and competitive advantage
- Lost incremental revenue and customer churn

No matter where an organization stands in the range outlined above, transcending this stage is crucial for any growing organization, it is simply not scalable and in some cases unsustainable. It is not necessary to build consensus or have a final goal at this point. This is the time to identify which team to work with on a CDE pilot. Focus on the following dimensions to progress to the Foundational stage:

Tooling & Infrastructure

Assess the current situation. Catalog and grade variations in tooling and infrastructure. Identify any current best practices.



People

Discuss best practices with each team, identify teams that have or are willing to follow a best practice model. Target teams that are using the same base operating system as the CDE (most likely Linux), this ensures minimal friction when enabling their tooling in a workspace. Establish a DevEx rating model and establish an acceptable baseline. Assure developers that the goal is simply to assess, not to eliminate any processes; there are always edge cases in software development that necessitate a local environment.

Process

Determine which team to invite to the CDE pilot. Ideally the assessor finds a team that has an abstracted and semi-automated process and an acceptable baseline rating for developer experience. That is most likely the team to invite to the pilot.

STAGE 1

Foundational

The Foundational stage has a narrower focus, the goal of this stage is to deploy a CDE that:

- Runs on supported infrastructure
- Supports the pilot team's development projects without carrying any technical debt over from Stage 0
- Improve pilot team's DevEx rating

The Foundational stage is characterized by these dimensions:

Tooling

Work with the pilot team to define requirements and select a CDE. Enable pilot team's workspaces. It is crucial to settle any workspace related technical debt at this point, build the workspaces according to best practices on the CDE's base operating system.

Infrastructure

Involve the platform and security teams at the start of this phase to evaluate the chosen CDE and determine the best minimally viable, secure, and supported infrastructure that is representative of the organization's production systems.

Most CDE vendors/providers offer a free-to-use tier that is fully functional and usable for smaller teams and projects. These offerings are a great way to introduce CDE functionality to your team, but as your CDE usage increases you'll need the advanced scaling and governance features that the paid enterprise tiers offer.

If your CDE could grow to 50 or more developers, considering enterprise features from the start will ease any scaling issues you may see in the future.

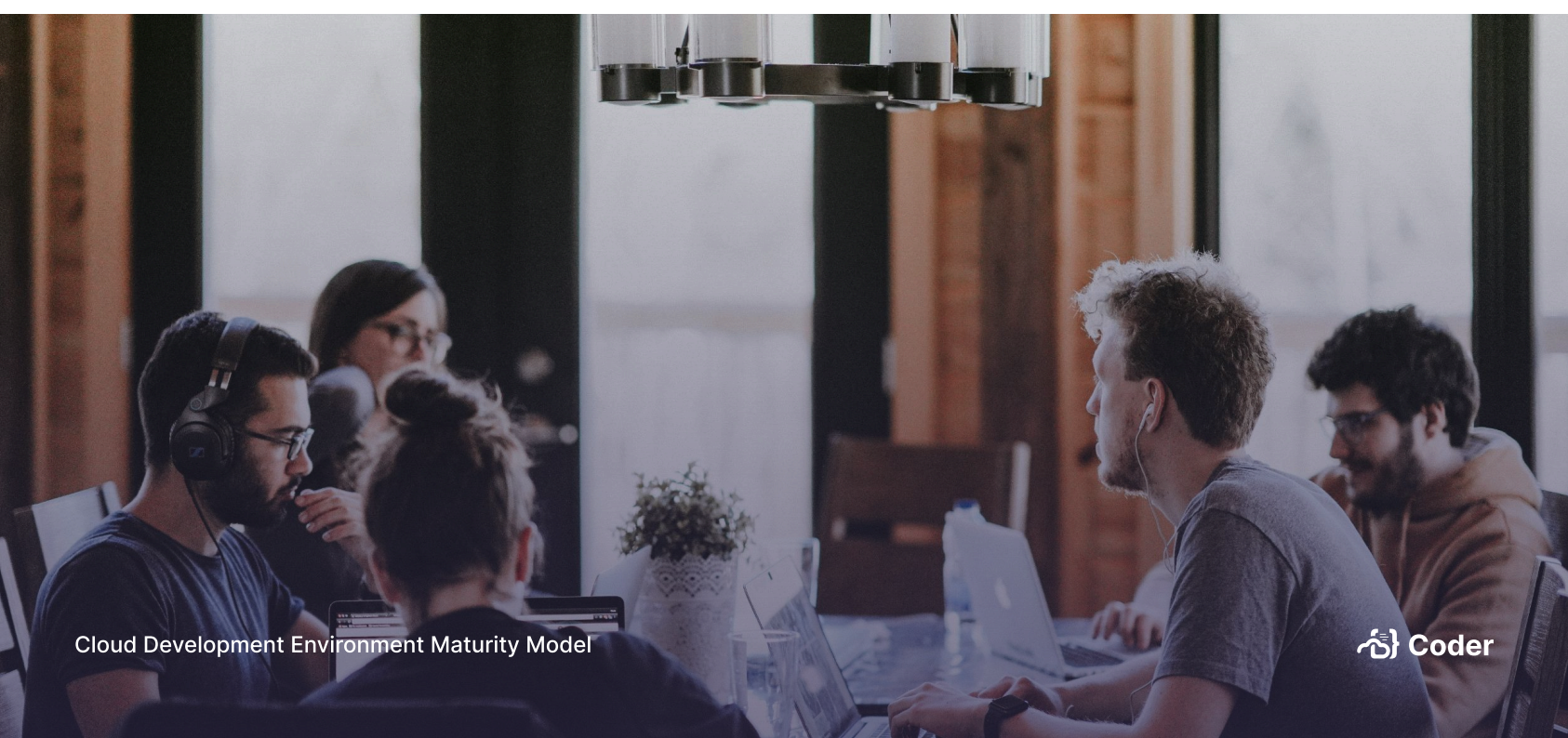
People

Pilot team readies their projects and moves the first project to CDE. Iterate until DevEx rating for that project is measurably better than their Ad-hoc baseline. Once a DevEx rating gain is established, build a workspace for the pilot team's next project, and repeat until the pilot team is CDE-only.

Process

Small team of platform engineers and pilot team work in conjunction to deploy and govern CDE, iterate and refine workflow.

There are two types of risk at this stage. The first is simply stopping at this point. If the pilot team is the only user of the CDE, the organization effectively regresses to Stage 0; the CDE becomes just another team-based abstraction among many. The second risk is expanding too quickly. Other teams will take notice and may want to start using the CDE at this point. However, a foundational CDE is not enterprise ready and adding workloads at this point creates technical debt. Focus on the deliverable of this stage: a CDE that supports one team that can be used as a baseline for an enterprise CDE.



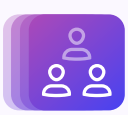
Enterprise-wide goals and use cases become apparent as this phase draws to a close. Focus on the following dimensions to progress to the Defined stage:

Tooling

Catalog and prioritize use cases to enable during the Defined stage.
Create technical supportability collateral that define requirements to convert to a workspace pattern.

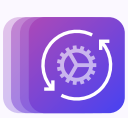
Infrastructure

Begin preliminary design work on an enterprise grade infrastructure. It must conform to organizational security standards, include at least one scalable cloud platform, and integrate with existing enterprise systems; e.g. a hybrid cloud that runs federated instances of the same platform both on-prem and in a public cloud and integrates with the organization's identity provider and monitoring systems. Organizations that are already cloud-native hold an advantage here.



People

Generate buzz around the CDE; talk with teams who expressed interest.
Create a list of ideal teams to convert in the next stage. Define enterprise wide DevEx scoring model.



Processes

A great CDE is started as a ground-up effort that becomes fully backed by leadership. Identify a process champion in leadership and present the findings from this stage. Work with the champion to define goals for the Defined stage and beyond. Model how productivity and DevEx scoring gains are multiplied in the next stages. Foster preliminary discussions about the organization’s visions for a Golden Path for developers.

STAGE 2

Defined

The goal of the Defined stage is to deploy an enterprise grade CDE that is stable, secure, and scalable. This stage has a clearly defined, yet expansive scope including the following deliverables:

- Build out and integrate a long-lived production infrastructure
- Migrate the pilot group's workspaces
- Expand CDE usage to ideal teams
- Decommission Foundational infrastructure
- Establish a staging environment

The Defined stage is characterized by these dimensions:

Tooling

Fully migrate the pilot team's workspaces from the Foundational CDE to the production CDE. Then incrementally enable other ideal teams' workloads. Follow the same pattern from the Foundational stage: settle technical debt and build workspaces according to best practices on the CDE's base operating system. Standardize local-to-CDE conversion patterns. All workspace requirement requests flow through the platform engineering team in the form of a request/enablement cycle; i.e. developers request a change to their environment, the platform engineers update the workspace template, and finally developers update from the template.

Technical Debt:

The accumulation of suboptimal configurations, quick fixes, or shortcuts that are taken to meet immediate deadlines or reduce upfront costs. These compromises often result in increased complexity, fragility, and reduced maintainability. Over time, this "debt" incurs "interest" in the form of additional effort required to address issues, refactor, and implement new features, leading to slower iterations and higher long-term costs.

By proactively addressing technical debt, organizations can streamline their development processes, improve code quality, and reduce the long-term costs associated with maintenance and rework. Moreover, minimizing technical debt frees up resources, allowing teams to focus on innovation and quickly adapt to new challenges or market demands, ultimately enhancing the organization's competitive edge.

Infrastructure

CDE is deployed on a software defined, scalable cloud platform. This could include on-prem, hybrid on-prem/public, or public cloud. It is fully integrated with enterprise identity provider, monitoring, and security systems. It is crucial to decommission the infrastructure used in the Foundational stage and establish a staging environment that is representative of production. Use the staging environment to vet all potentially breaking changes to the CDE.

People

Groups for each team and/or function are created in an enterprise identity provider. Developers from multiple teams ready their projects to migrate to CDE. As with the Foundational stage, iterate on a project until its DevEx rating is inline with the rating from the pilot team. Keep in mind that this is a larger jump (stage 0 to stage 2) for these developers. Once an acceptable DevEx rating is achieved, move on to the next project and repeat until these teams are CDE-only.

Process

Production-grade CDE is deployed, managed, and observed by the enterprise platform engineering team. Staging system is in place to vet all changes.

At this stage the CDE has gained significant traction and both tangible and intangible benefits are realized. The journey is far from complete. Stagnating at this stage risks losing continual incremental gains, losing expansion opportunities, and – most critically – waning enthusiasm from the organization's most advanced development teams. If the advanced teams don't experience progressive functionality and constantly reduced friction they could abandon the CDE. Which leads the organization on a regressive path back to Stage 0.

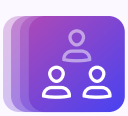
As the organization moves through this stage, points of both refinement and expansion become apparent. To progress to the Refined stage focus on the following dimensions:

Tooling

Research the use of self-serve technologies such as the development container specification. Also research converting some alternate use cases such as VDI and workspaces for non-developers: data scientists, system operators, SREs, etc.

Infrastructure

Consider alternate cloud providers, computing platforms and hardware tiering. Developers may need features and capabilities not available on the current platform. Advanced development use cases may require additional resources while alternate use cases may run perfectly on less powerful nodes.



People

Continue to generate buzz around the CDE. Form two peer teams composed of developers; one for optimization and one for expansion. The optimization team focuses on accelerating innovation by reducing friction. The expansion team focuses on identifying alternate use cases and setting DevEx rating goals for each.

Processes

Work with platform and security teams to discover or implement internal systems that make self-service possible such as container registries, software catalogs and software repositories. Continue to work with CDE process champion to refine goals, and boost the visibility of the CDE.

STAGE 3

Refined

The Defined stage provided a CDE that is useful to a significant number of developers. The Refined stage aims to expand that CDE in both capabilities and user population. The strategic importance of the CDE becomes apparent in this stage. The goals of this stage are:

- Grow CDE usage
- Reduce developer friction
- Expand the computing platform

The Refined stage is characterized by these dimensions:

Tooling

The development container specification is introduced. The CDE is granted access to additional enterprise systems such as software catalogs, secure container registries, and secure software repositories to enable developer self-service processes.

At least one alternate use case is enabled via the platform team's request/enablement cycle. An alternate use case is defined as either a development use case that uses a different operating system than the platform (such as Windows) or a non-development use case such as data science. Windows development may require additional tooling and configurations such as secure browsers, RDP clients, and the enterprise policies that go along with them.

Infrastructure

Authentication and authorization to at least one alternate cloud provider or platform are federated. This could include multiple public cloud providers and/or multiple on-prem platforms spanned across multiple datacenters. Multiple compute platforms are available for workspace deployment (Docker, Kubernetes, VMs). Hardware tiering is enabled and determined per use case.

This stage aims to meet the CDE users where they are, with the hardware they need. For example, providing an AI/ML developer on a different continent a workspace that is deployed in-region with the required memory and GPU cores for the task. Or providing 100s of Windows workspaces in each of the organization's large offices.

People

Up to this point, the platform team has been responsible for fulfilling the developers' requirements with a request/enablement cycle. This stage introduces developer self-service with the developer container specification. It allows the platform team to maintain control over the templates and infrastructure, but allows developer teams to fulfill their workspace dependencies in a secure and scalable manner. This is a significant milestone in creating a Golden Path for developers. Development teams work with the optimization team (as established in the previous stage) to adopt the development container specification.

The **Golden Path** is a set of best practices, tools, and processes that guide developers to efficiently and effectively use a CDE. It is a curated, opinionated approach designed to minimize friction, reduce errors, and ensure that developers can focus on writing code rather than dealing with environment setup and maintenance. It provides a clear, consistent workflow that all developers in the organization can follow, leading to a more predictable and productive development process.

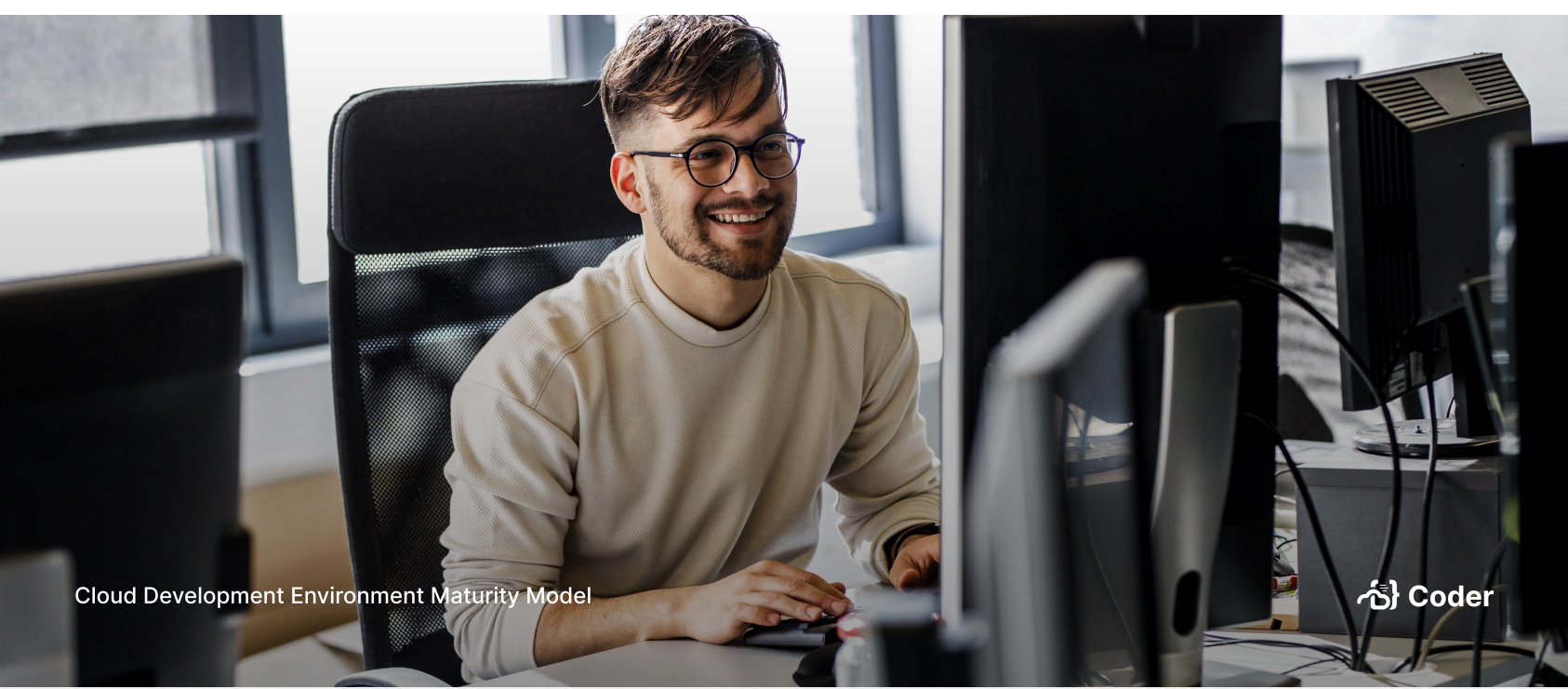
It is continuously refined based on feedback from developers and advancements in technology. It empowers developers by providing a robust, low-friction experience, allowing them to spend more time on innovation and less on troubleshooting or configuring their environments.

Use cases so far have been narrowly scoped: teams of developers working on the native operating system of the CDE. This stage introduces workspaces to enable developers who use Windows systems or enable non-developer use cases such as data science and analysis. Alternate use cases do not move towards self-service at this stage, they rely on the platform team's request/enablement cycle. These teams work with the expansion team (as established in the previous stage) and enterprise security to enable alternate use cases.

Process

The platform team still owns the templating and infrastructure placement for workspaces; placement and platform selection is determined by either performance or economic factors. The platform team focuses on solving – and future proofing against – scaling issues related with self-serve, multi-cloud, hardware tiering, and alternate use cases. Platform, security and leadership teams work together to create governance, observability and security guidelines.

At this point the CDE is truly an enterprise-wide standard platform with room for continual growth and improvement. The platform is solid and growth has become organic; teams are requesting access then working with the expansion team to onboard with minimal overhead. Stagnation at this stage poses a significant risk to platform growth, centered around scalability and delegation. If these are not continuously improved it will lead to waning enthusiasm for the platform and a regressive path for the organization.



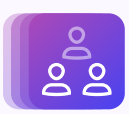
To progress to the Optimized stage focus on the following dimensions:

Tooling

Catalog as many unconverted workloads as possible, grade, and rank as conversion candidates.

Infrastructure

Ensure any scaling issues with the increased growth and multi-cloud capabilities are solved. Identify ideal uses for various cloud providers, computing platforms and hardware tiers.



People

Engage the optimization team to identify developers to delegate template maintenance. Shift devcontainer enablement from optimization team to expansion team. Engage the expansion team to identify additional self-service users and alternate use cases.

Process

Define guidelines for template delegation operations. Establish CI/CD pipeline for templates. The CDE is a major enterprise resource at this point but there is always room for improvement. Continue to work with CDE process champion to pave the Golden Path and boost the visibility of the CDE.

STAGE 4

Optimized

During the Refined stage, the CDE met many of the organization's tactical requirements and matured into a strategic system with a clear process.

The Optimized stage is a major inflection point for the CDE; the organization's strategy begins to center around the CDE rather than the CDE simply meeting the organization's requirements. This is the final stage of the maturity model so any efforts are continuous and iterative.

The goals of this stage are:

- Continue to scale the platform and reduce developer friction
- Enable tiered self-service
- Introduce a clearly defined Golden Path for the organization's development staff

The Optimized stage is characterized by the following dimensions:

Tooling

As with the Refined stage, automation and alternate use cases are continually vetted and added to the CDE. A CI/CD pipeline has been deployed by the platform team to allow developers to create, test, and maintain their own templates, enabling full self-service. This creates a tiered self-service model where teams can mature to full self-service.

Tiered Self-Service:

A CDE can offer developers varying levels of control over their environments, allowing for flexibility and scalability based on their expertise and needs. This approach helps balance the ease of use for developers with the governance and oversight required by the organization.

Tier 0:

Developers rely entirely on the platform engineering team to set up and maintain their development environments. Any changes or updates to the environment require a request from the developer, which the platform team then implements. This tier is ideal for alternate use cases and non-developer users

Tier 1:

Developers use pre-defined templates to define the workspace hardware and networking while using the devcontainer spec to define the software dependencies. This allows developers to adjust their workspace to meet specific project needs while still maintaining the consistency and security of the core environment. It strikes a balance between control and flexibility, enabling faster iterations without sacrificing governance.

Tier 2:

Developers have complete control over their development environment templates. This level of autonomy accelerates development and innovation, as developers can rapidly adapt their environments to new tools or workflows without waiting for platform team approval. However, this tier also requires a higher level of expertise and responsibility from developers to ensure that their environments remain secure and aligned with organizational standards.

Infrastructure

The underlying infrastructure is the same as the Refined stage, although continuously and incrementally improved. The major infrastructure change in this stage is developers are now able to choose where to deploy their workspaces in a secure manner without intervention from the platform team.

People

Developers are free to fully self-serve and smooth out any points of friction in their enablement process; this accelerates innovation and competitive advantage. The optimization team's scope narrows to enabling full self-service. The expansion team continues to find and enable new use cases and encourage progression through the self-service tiers.

Process

Tiered enablement, fewer teams are fully dependent on the platform team's request/enablement process. Most developers self-serving with either devcontainers or templates. The organization's Golden Path for developers is clearly defined: a process for teams wanting to onboard to the CDE and progress to fully self-service. Platform team focuses on running and maintaining CDE as just another platform.

Even though this is the final stage there are risks with stagnation. As with any strategic system, the organization must ensure it is properly maintained, improved, and governed. Continual incremental improvements ensure that the CDE is seen as the go-to platform.

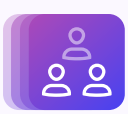
To continuously improve the CDE, focus on the following dimensions:

Tooling

Continually enable new use cases. Continually iterate on self-service processes.

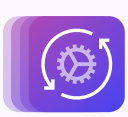
Infrastructure

Integrate ongoing CDE optimization with strategic organizational goals. For example, if the organization needs to balance spending across multiple providers – or even completely move away from a provider – utilize CDE's multi-cloud capabilities to move workloads. Ensure the latest tech is available to CDE users at the appropriate tiers. Continually iterate on automation.



People

Task the expansion and optimization teams to continuously find new use cases and encourage self-service. Periodically assess DevEx rating for all users of the CDE.



Process

Utilize DevEx rating data to drive incremental improvements at all levels of the CDE. Ensure CDE stays in compliance with all security and governance requirements. Work with process champion to continuously adjust and improve the strategic goals of the CDE and Golden Path.

Conclusion:

The Cloud Development Environment Maturity Model offers a comprehensive framework for organizations aiming to modernize and optimize their development environments. By progressing through the maturity stages—Ad-hoc, Foundational, Defined, Refined, and Optimized—organizations can systematically enhance their development processes, infrastructure, and tooling. This journey not only drives efficiency and innovation but also aligns with broader organizational goals, ensuring that the development environment remains scalable, secure, and adaptable to future challenges.

Achieving a mature CDE requires a clear understanding of the current state, a commitment to incremental improvements, and active engagement from all stakeholders, including developers, platform engineers, and leadership. As the CDE evolves, it becomes a strategic asset that empowers teams, reduces friction, and fosters a culture of continuous improvement. By following the guidelines and best practices outlined in this model, organizations can create a robust, self-service development platform that not only meets today's demands but also anticipates and adapts to future needs, securing a competitive advantage in an ever-changing technological landscape.