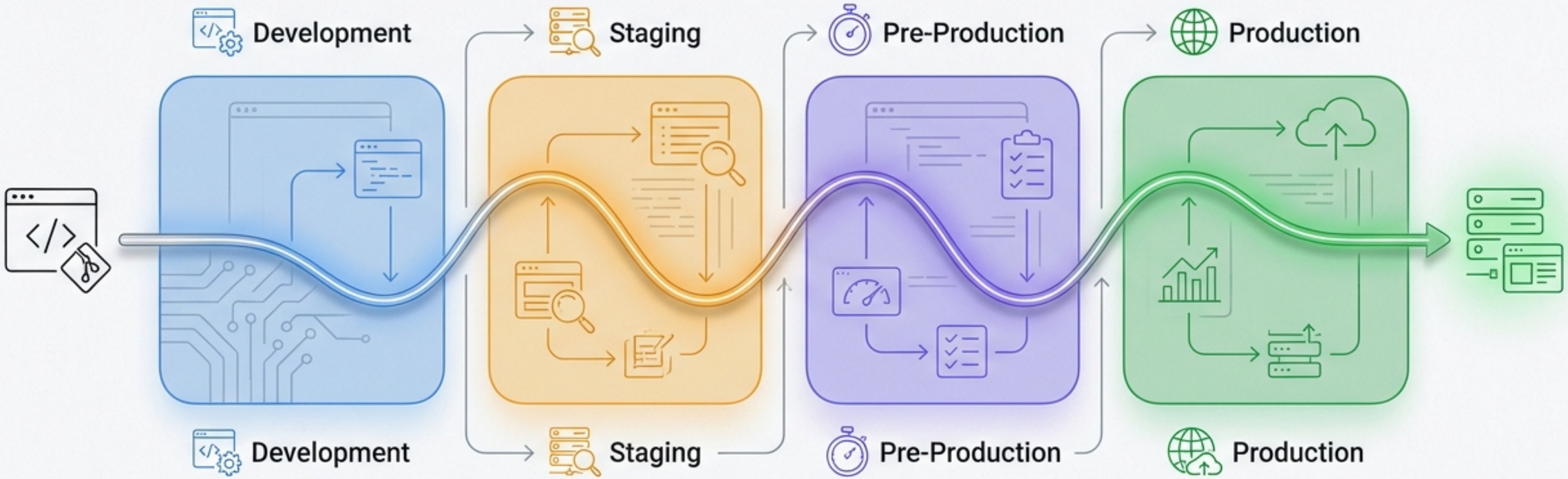
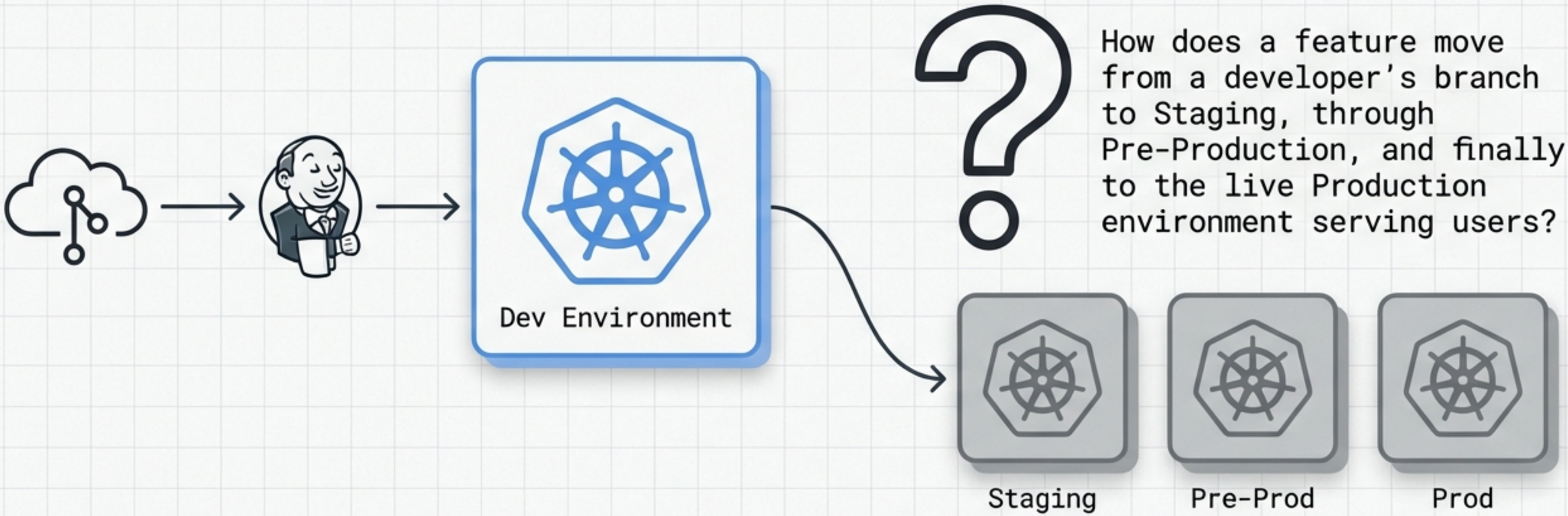
The Journey of a Commit: A Modern CI/CD Workflow

From a Developer's IDE to a Live Production Environment



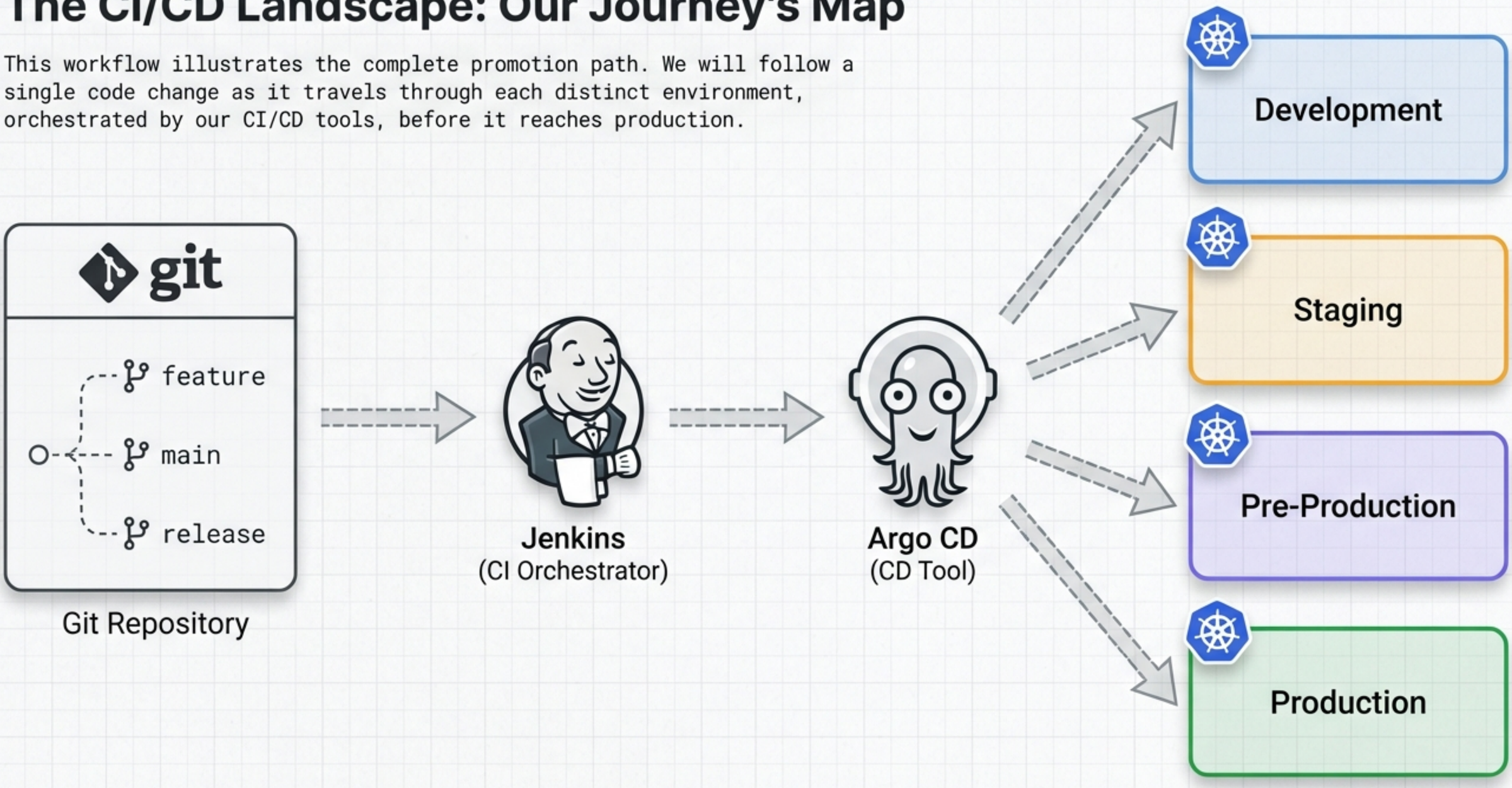
Moving Beyond a Single Environment

A standard CI/CD pipeline successfully automates deployment to a single environment, like a development cluster. However, real-world software delivery requires a structured process to promote code safely across multiple stages.



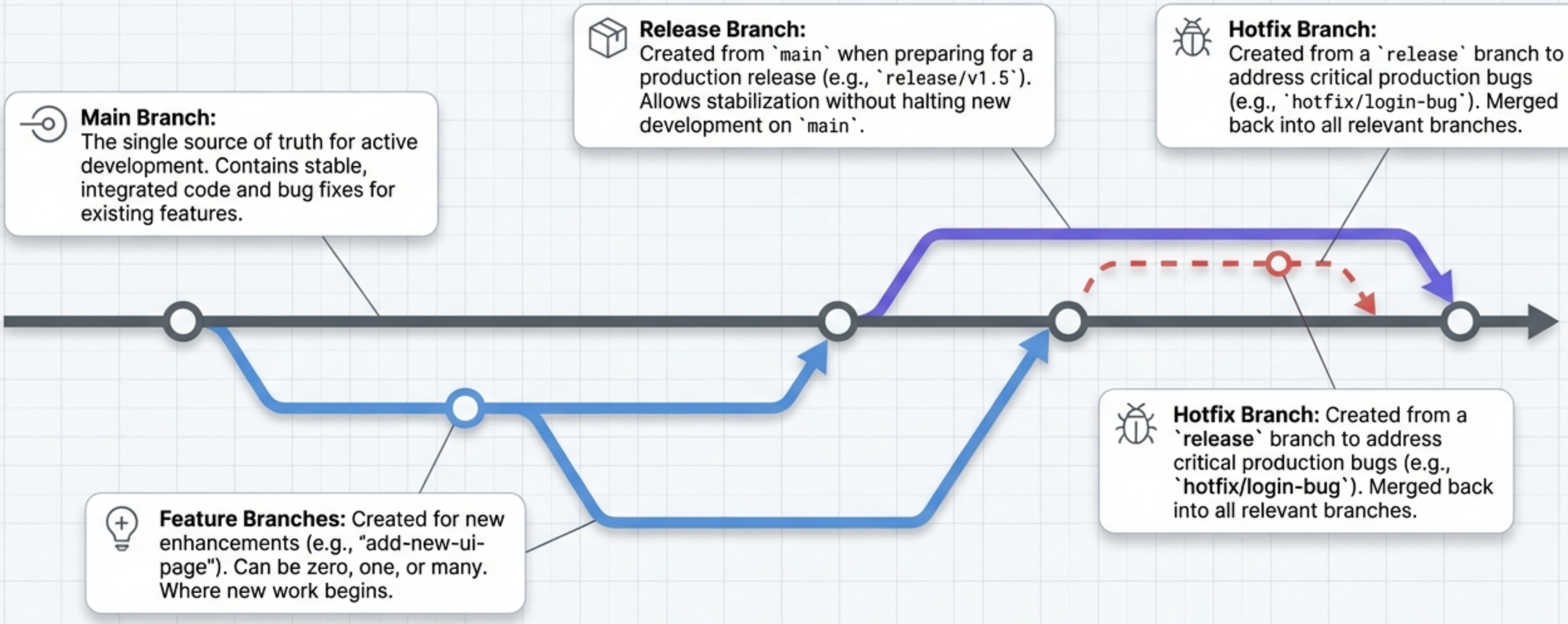
The CI/CD Landscape: Our Journey's Map

This workflow illustrates the complete promotion path. We will follow a single code change as it travels through each distinct environment, orchestrated by our CI/CD tools, before it reaches production.



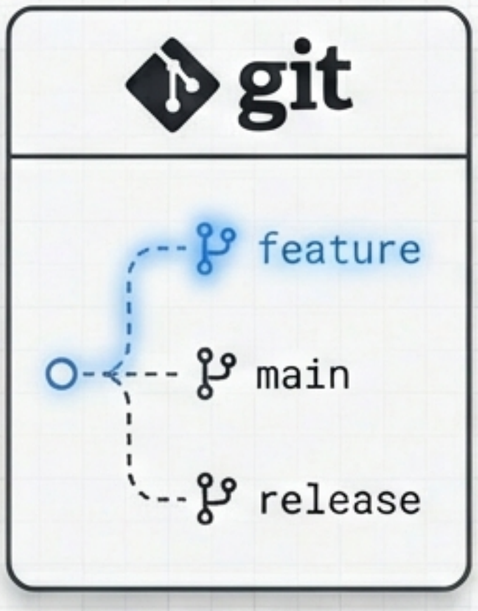
The Rules of the Road: Our Branching Strategy

A well-defined branching strategy is the blueprint for our promotion workflow. It ensures code is managed cleanly, features are developed in isolation, and production remains stable.



Stage 1: The Developer Commits to a Feature Branch

The journey starts when a developer pushes a code change for a new feature to its dedicated branch in the Git repository. A webhook configured on this branch instantly triggers our CI pipeline.



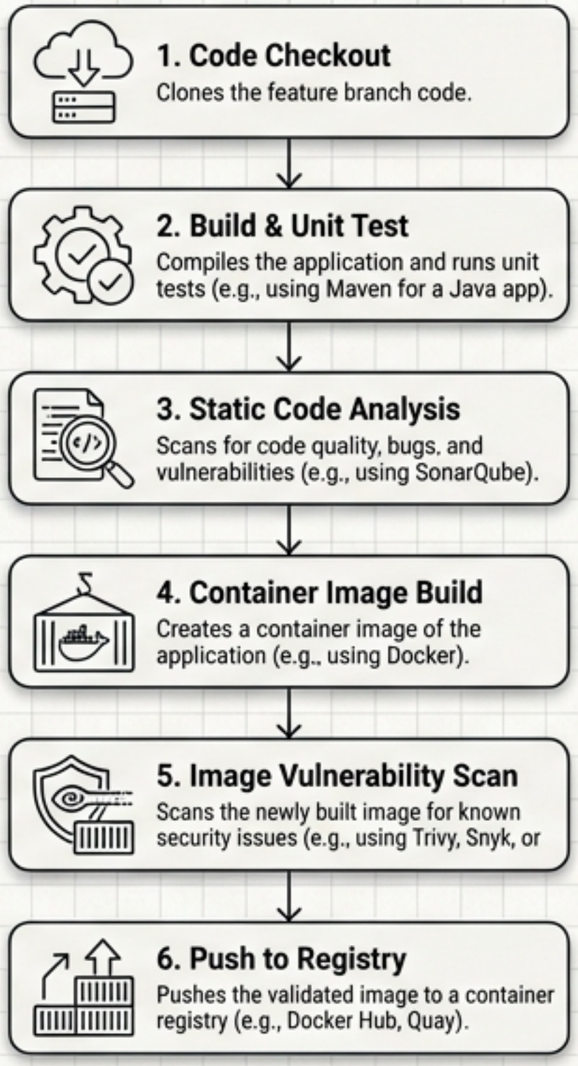
Git Repository

Jenkins
(CI Orchestrator)

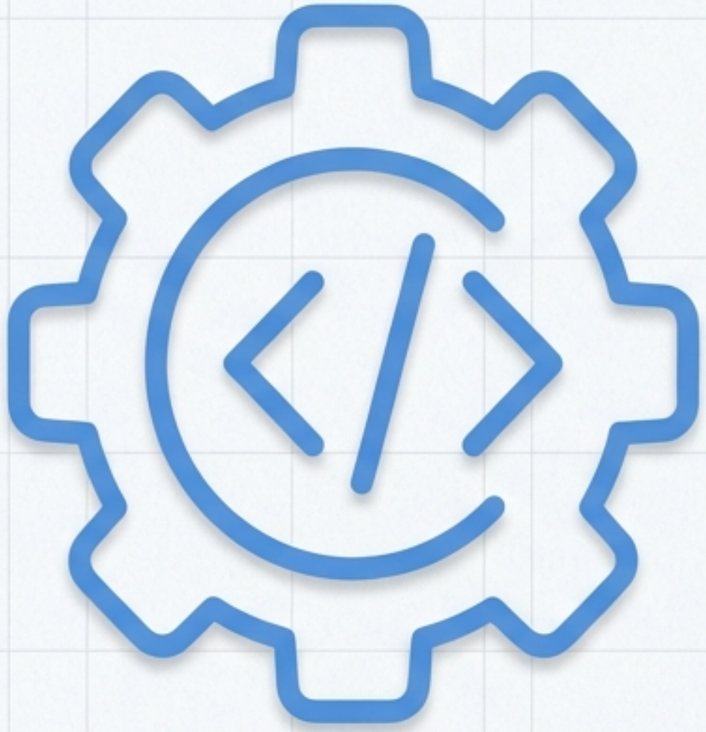
Argo CD
(CD Tool)

The Feature Branch CI Pipeline: Building with Confidence

This initial pipeline validates the new code in isolation, ensuring it meets quality and security standards before it can be integrated with other changes.



Environment Profile: Development



Purpose

To provide developers with a dedicated, isolated environment to deploy and test their specific feature changes immediately after a commit.

Key Activities

Live testing of new code, rapid iteration, debugging in a Kubernetes-like environment. The deployment is fully automated by the feature branch pipeline.

Ownership

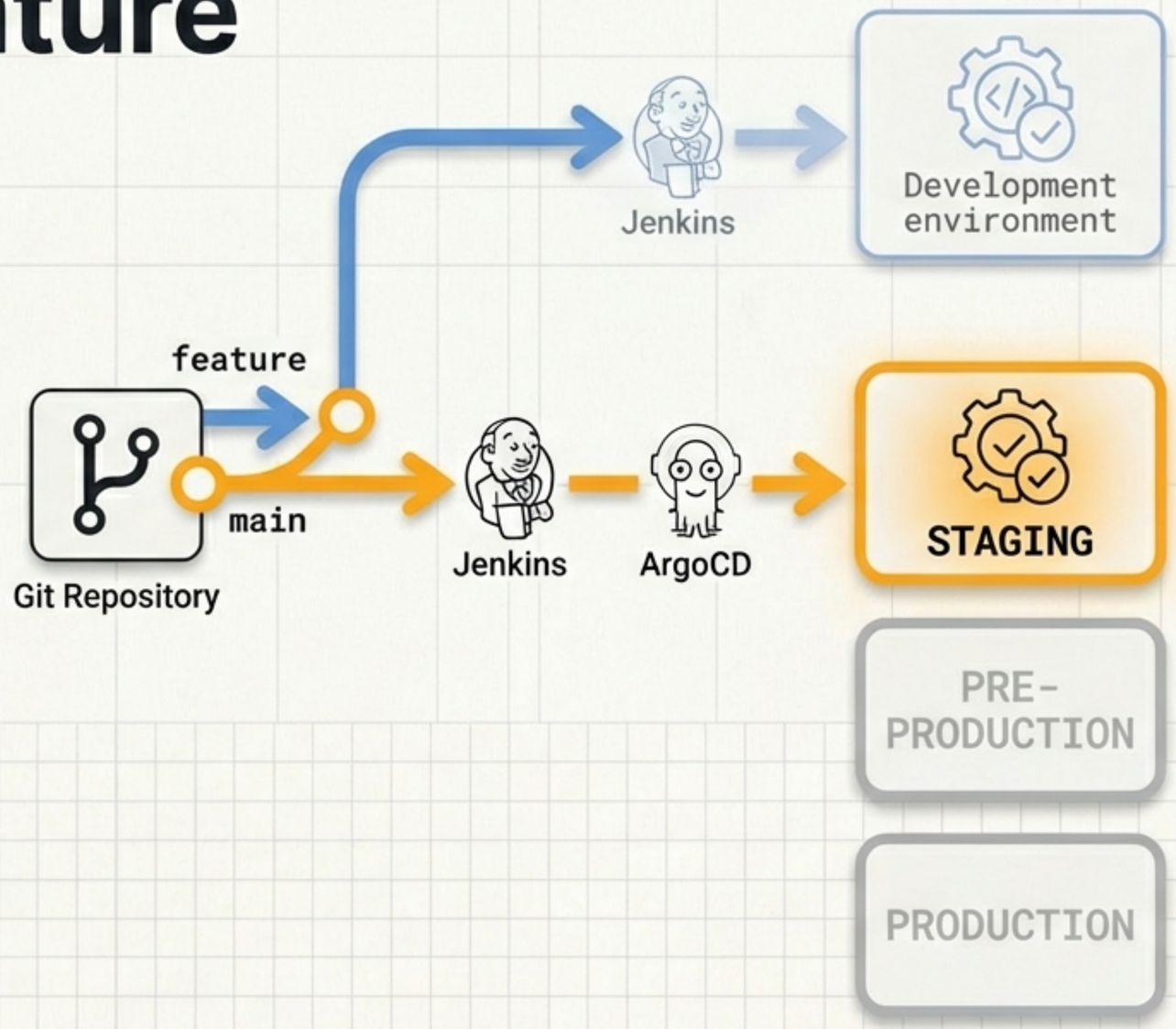
The Developer or Development Team working on the feature.

Infrastructure

Typically a dedicated Kubernetes cluster or a namespace within a shared cluster.

Stage 2: The Feature Merges into the Main Branch

Once the feature is complete and passes its CI pipeline, the developer merges the code into the `main` branch. This act of integration triggers a new, separate pipeline designed to deploy the combined codebase to the Staging environment.



Environment Profile: Staging (QA)

Purpose

To test the integrated codebase from the `main` branch. This is where we verify that the new feature works correctly with all other changes being made across the project.

Key Activities

Automated regression testing, functional tests, manual exploratory testing by the QA team. More intensive tests like performance testing and penetration testing (pen testing) may also be conducted here.

Ownership

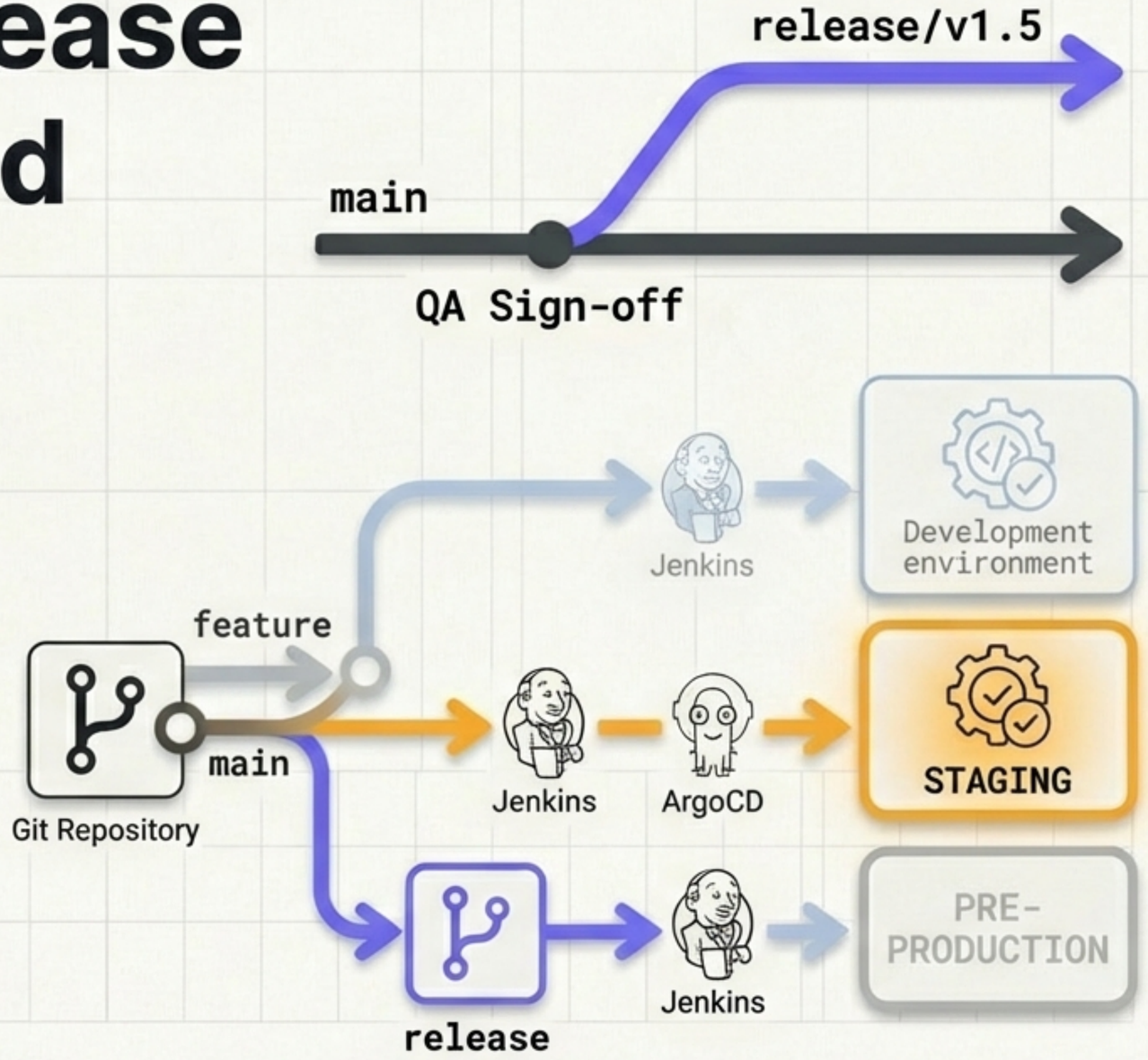
The Quality Assurance (QA) Team.

Infrastructure

A dedicated Kubernetes cluster that is more robust than the Dev environment. Developers and QA may have access, but it is more controlled.

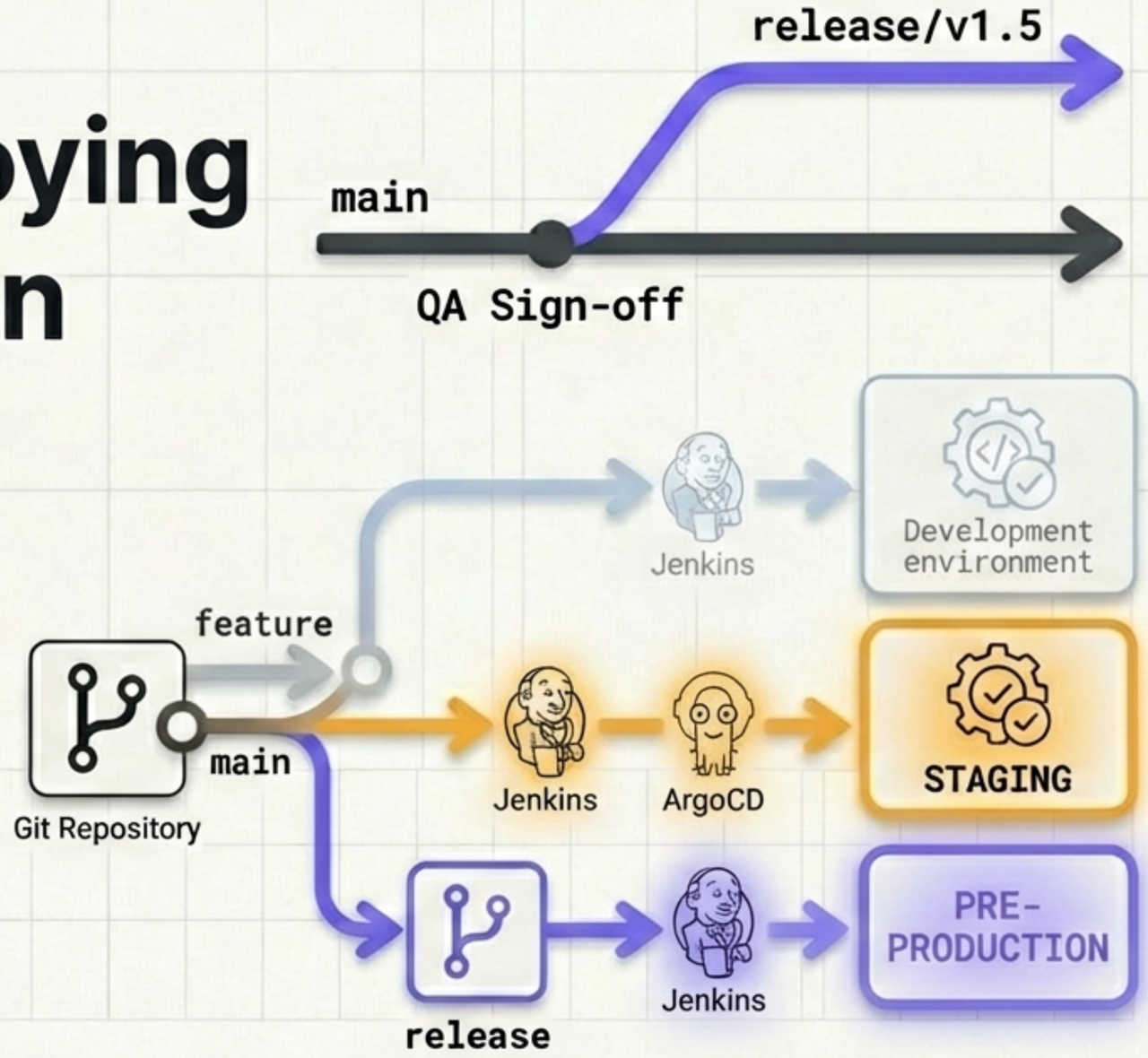
Stage 3: The Release Branch is Created Created

After the QA team provides a sign-off on the features in the Staging environment, a DevOps engineer creates a `release` branch from the `main` branch. This branch now contains a stable, feature-complete the application, ready for final validation before going to production.



The Final Dress Rehearsal: Deploying to Pre-Production

The creation of a release branch triggers the “release pipeline.” This pipeline is often the most rigorous, potentially including extensive, time-consuming automated tests (e.g., full-scale performance and security tests) that are not run on every commit. destination is the Pre-Production environment.



Environment Profile: Pre-Production (UAT)

Purpose

To serve as an exact replica of the Production environment. It's used for final validation, User Acceptance Testing (UAT), and safely reproducing/debugging production issues without touching the live system.

Key Activities

Final sign-off checks, UAT by product owners or stakeholders, validating deployment procedures.

Ownership

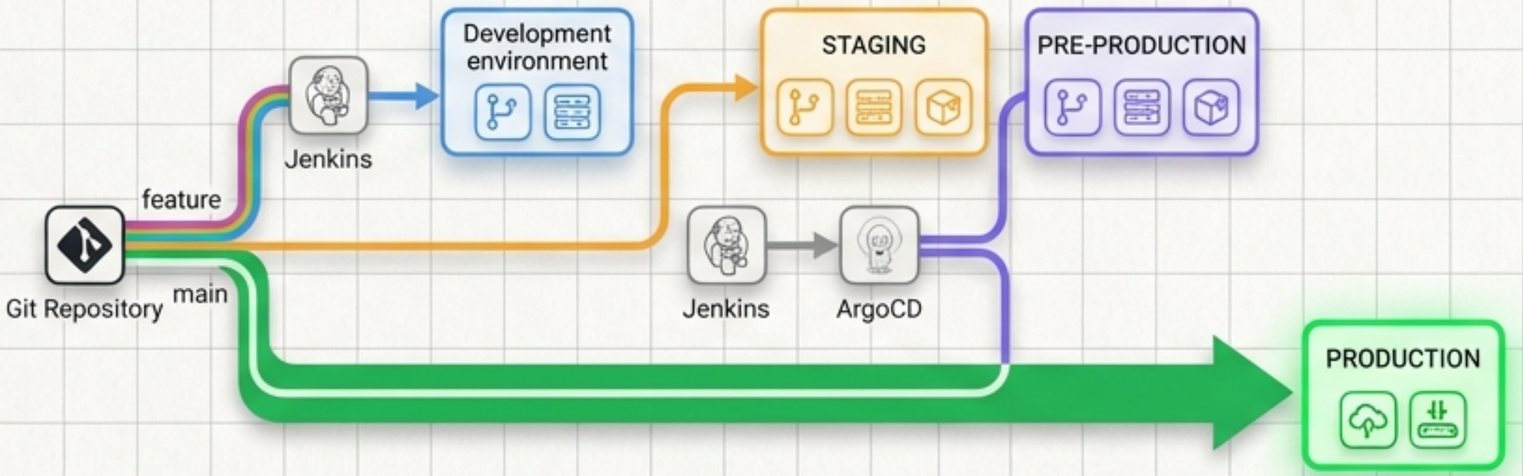
DevOps / Release Engineering Team. Access is highly restricted.

Infrastructure

A dedicated Kubernetes cluster that mirrors the production cluster's configuration, resources, and data policies as closely as possible.

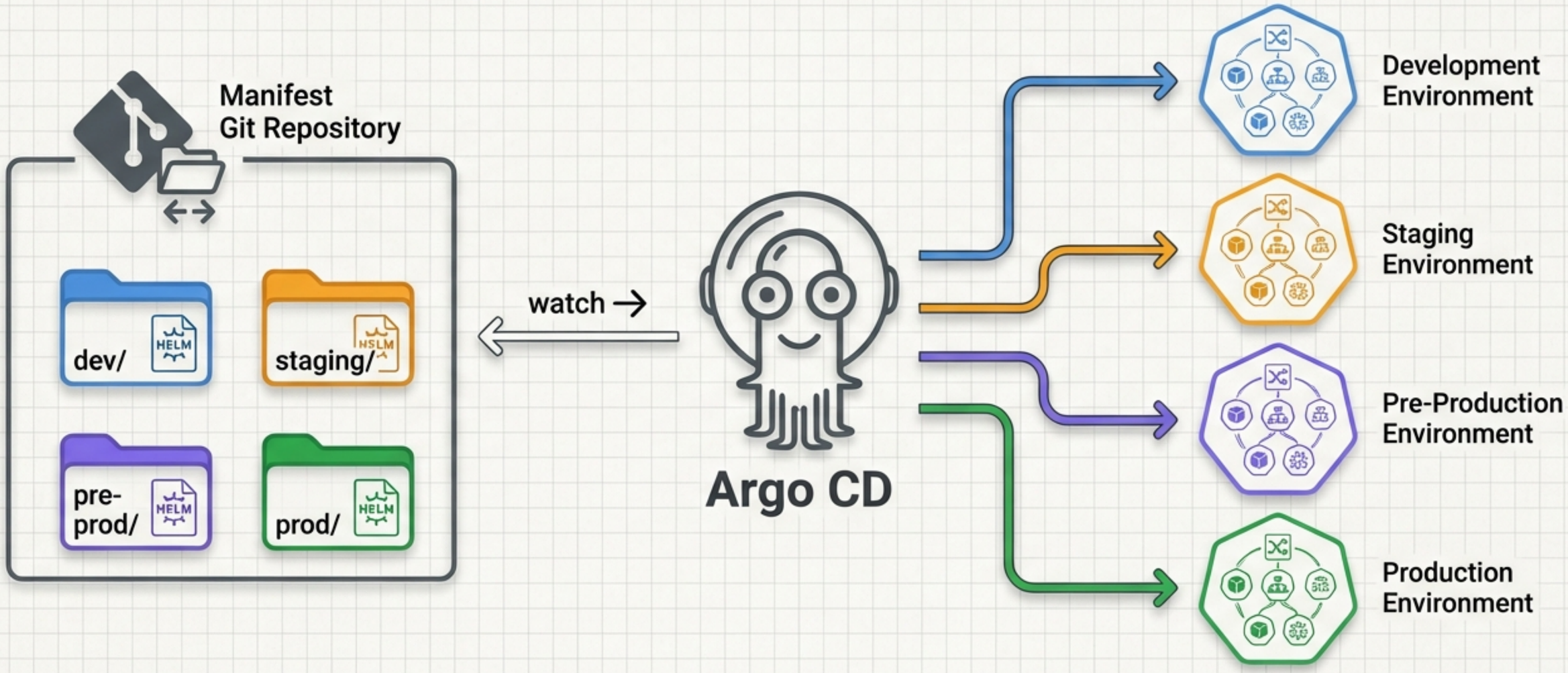
Final Stage: The Release Goes Live to Production

Once all checks in Pre-Production are green-lit, the same release pipeline is used to promote the application to the Production environment.
This final step is carefully monitored. The journey of the commit is now complete – the feature is live.



A Unified View: Managing All Environments with GitOps

While separate Jenkins pipelines trigger for each branch, a single Argo CD instance can manage deployments to all environments. It watches a dedicated manifest repository where each environment has its own configuration folder or Helm chart. This provides a single source of truth for the desired state of every environment.



The Journey's End: From Code to Customer Value

The journey of a commit is a disciplined process governed by a clear branching strategy and a series of validation gates. Each environment serves a distinct purpose, from isolated feature development to production-grade validation, ensuring that changes are integrated, tested, and deployed with increasing confidence at every step.