

自動化がもたらす「スピード」と「品質」の両立



自動化によるスピード

手動作業を極小化。コードのコミットからビルドまで、Azure Pipelinesにより一連のプロセスを完全自動化し、開発リードタイムを短縮します。



品質のゲートキーパー

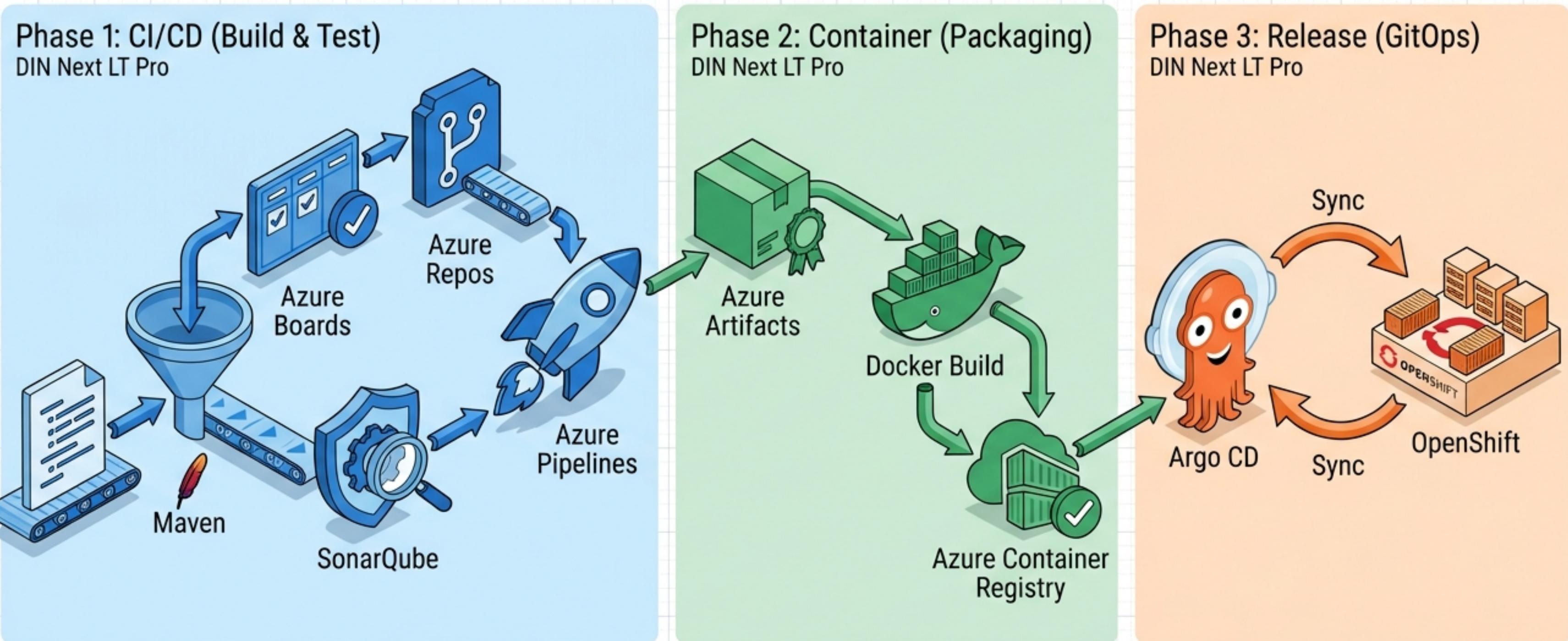
SonarQubeとJacocoによる自動検査をラインに組み込み、欠陥のあるコードが後工程に流れるのを機械的に阻止します。



シンプルな運用

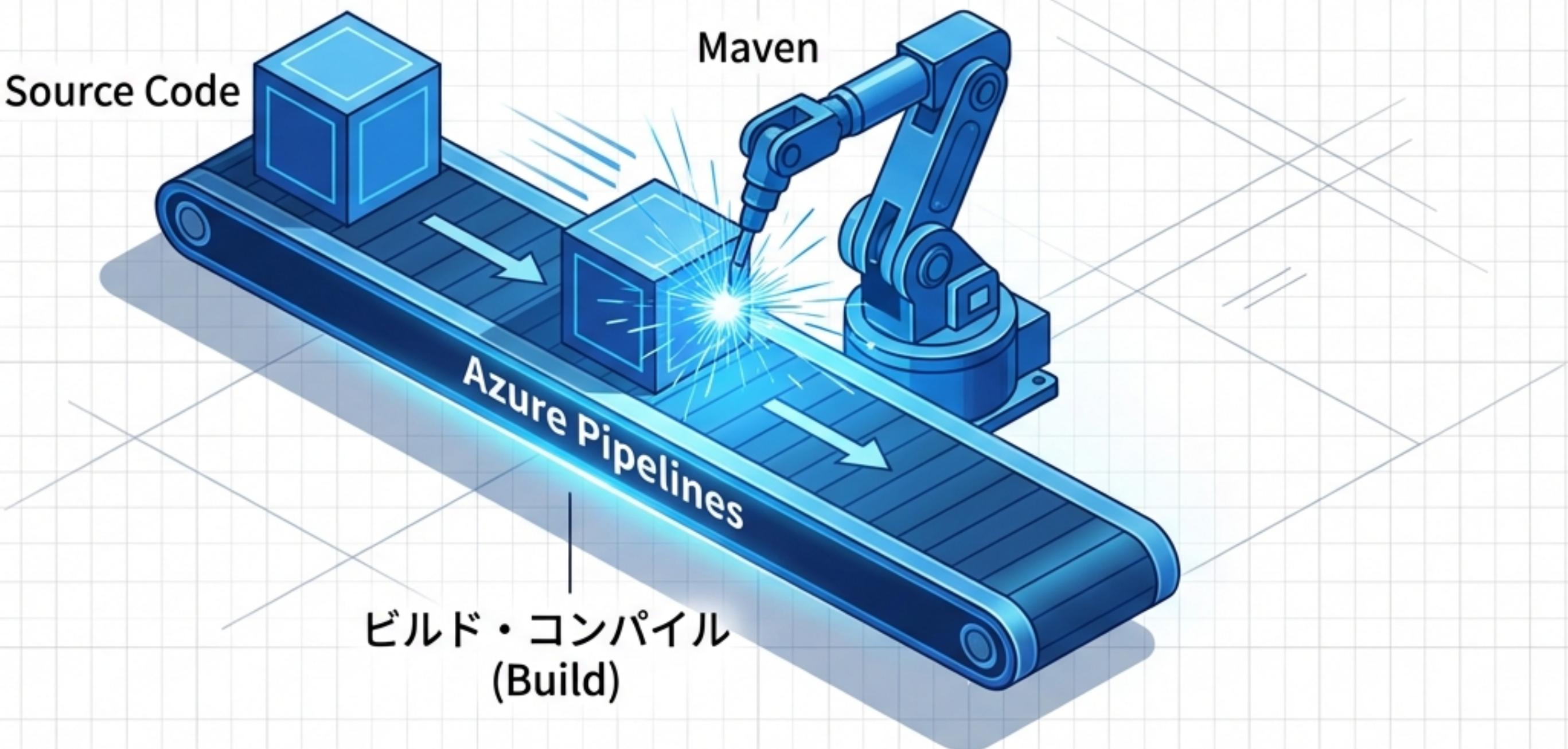
複雑なマージ作業やコンフリクト解消を回避するため、メインブランチを中心としたシンプルな開発フロー（トランクベース）を採用します。

全体構成：コードから本番環境への自動化された旅路



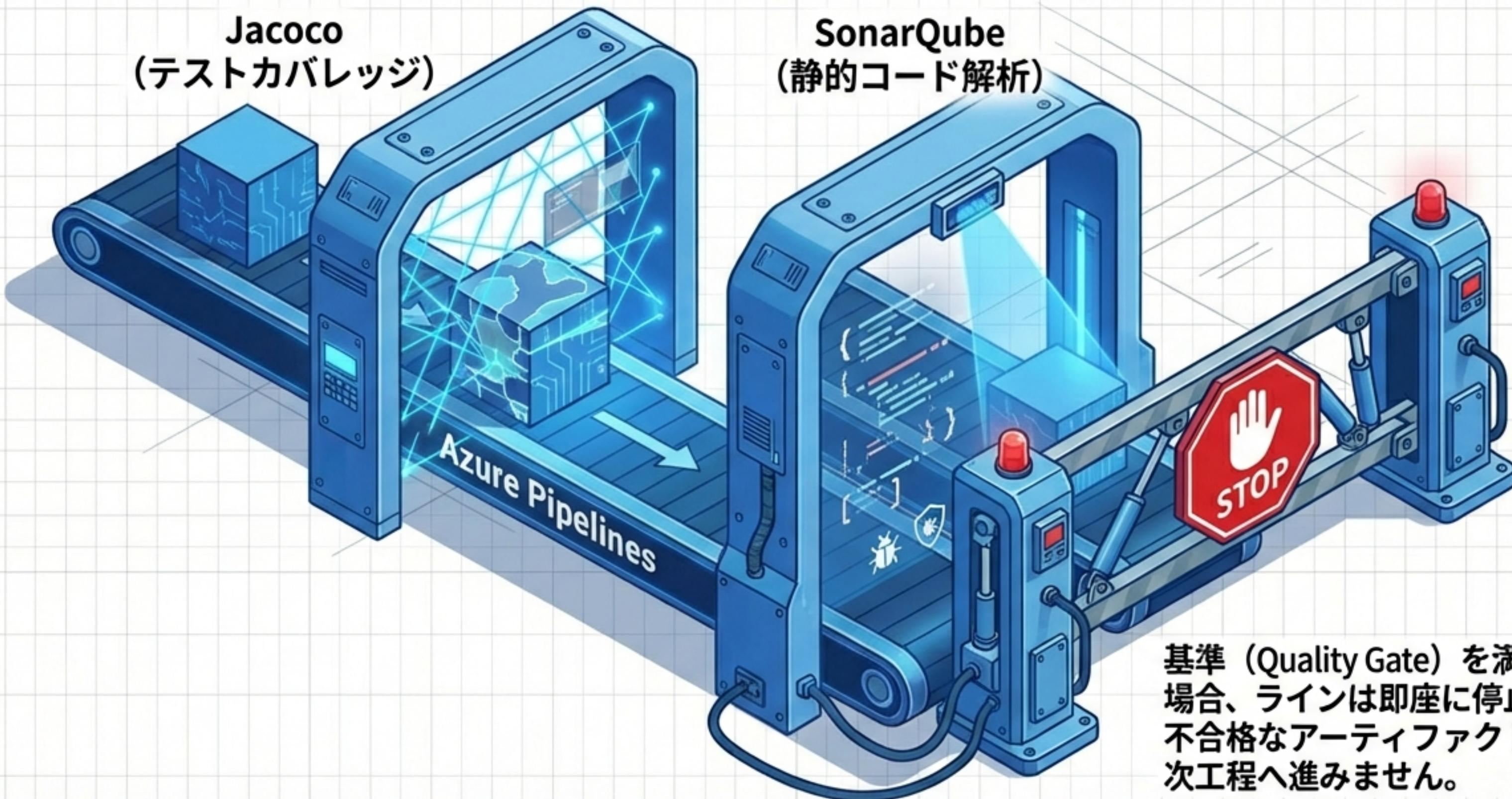
ソースコードから本番稼働まで、3つの自動化ステージを経てデリバリーされます。

Phase 1：自動化された組立ライン（CI）



開発者がコードをAzure Reposにプッシュした瞬間、Azure Pipelinesが組立ラインを始動させます。Mavenロボットが自動的にビルドを行い、手動操作によるミスを排除します。

品質のゲートキーパー：欠陥品の流出を阻止する



バージョン管理指針：「スナップショット」と「不变のリリース」

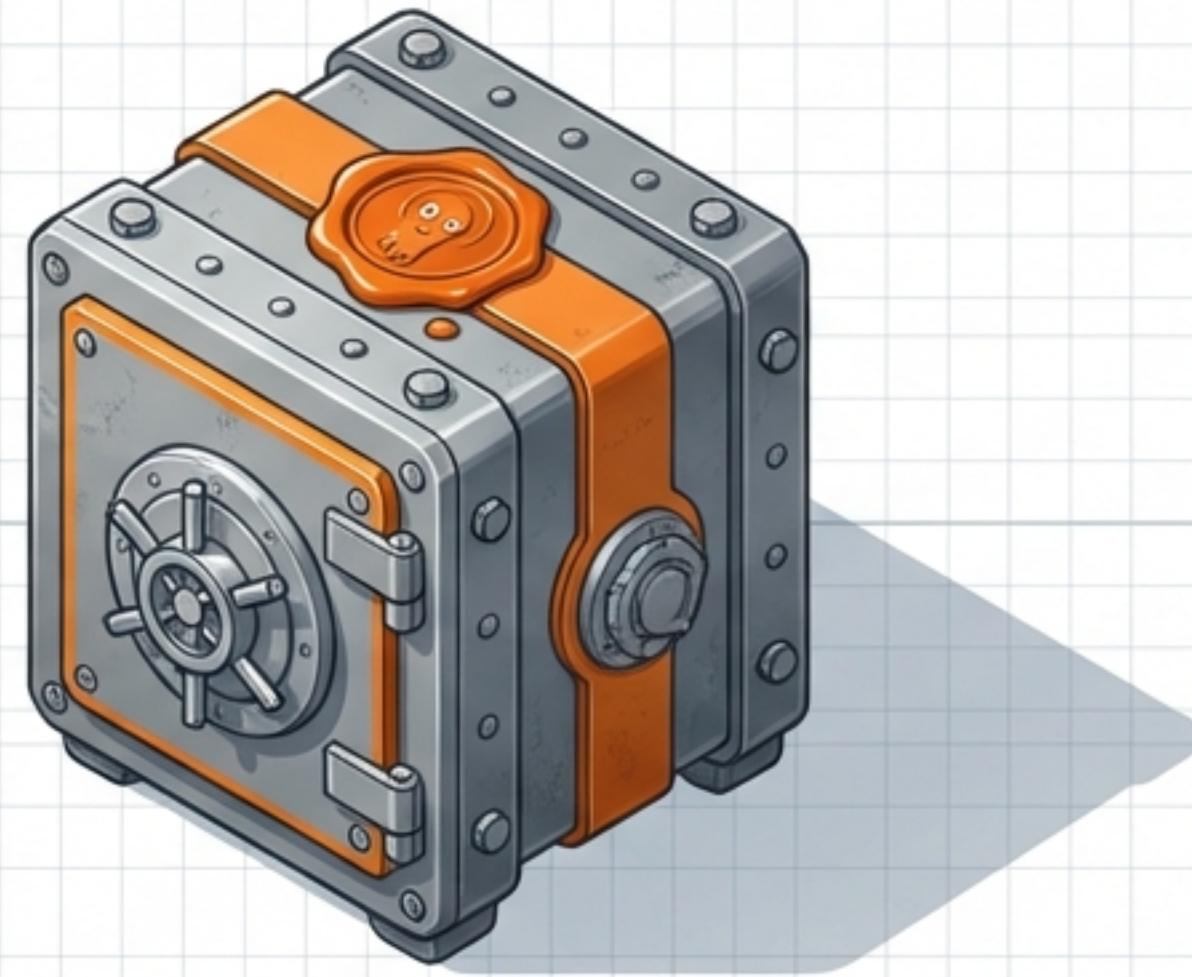
開発用：スナップショット (Snapshot)



ver 1.0.0-SNAPSHOT

日々上書きされる開発版。

本番用：イミュータブル・リリース (Release)

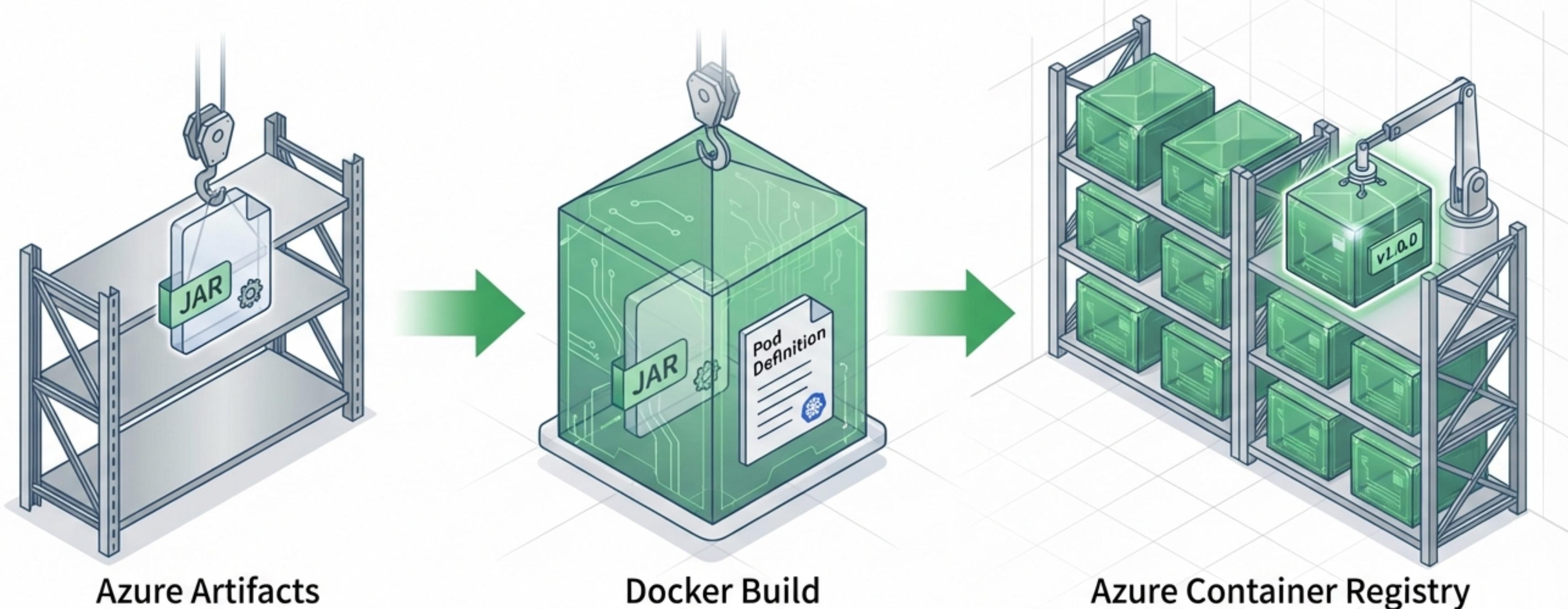


ver 1.0.0 (Immutable)

一度作成したら絶対に上書きされない不变のバージョン。

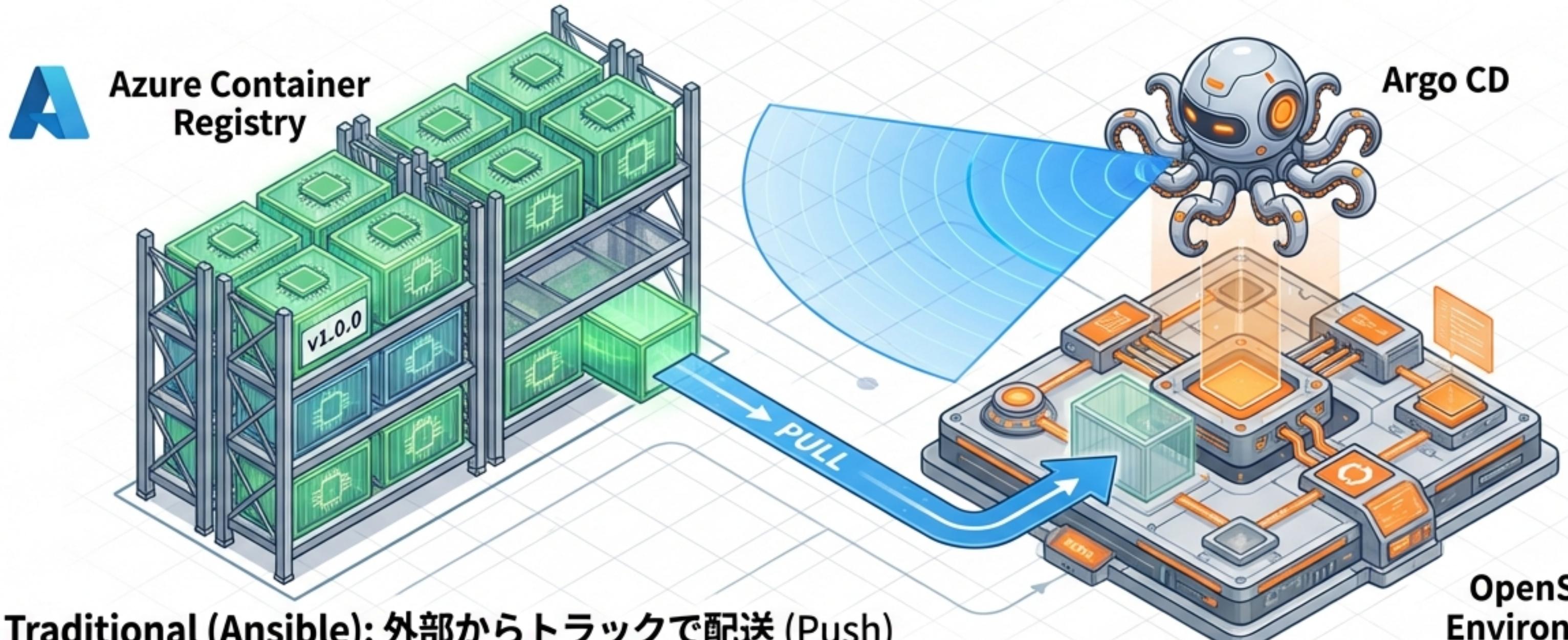
「不变のリリース」を作成することが、確実なロールバックと幂等性の条件となります。

Phase 2：コンテナ生成と格納庫への登録



アプリケーションと動作環境を一体化させることで、
「開発環境では動いたが、本番環境では動かない」という問題を根絶します。

Phase 3 : GitOpsによる継続的デリバリー (Argo CD)



- **Traditional (Ansible):** 外部からトラックで配達 (Push)
- **Modern (Argo CD):** OpenShift内のエージェントが、倉庫(Registry/Git)の変更を自動検知して同期 (Pull)

本番環境に常駐するArgo CDが、あるべき状態（マニフェスト）と現在の状態を常に監視。差異が発生した瞬間、自動的に同期を実行します。

同期メカニズム：宣言的定義による自動デプロイ



Gitを「唯一の信頼できる情報源 (Single Source of Truth)」として扱います。

確実な復旧：GitOpsが保証する「冪等性」

