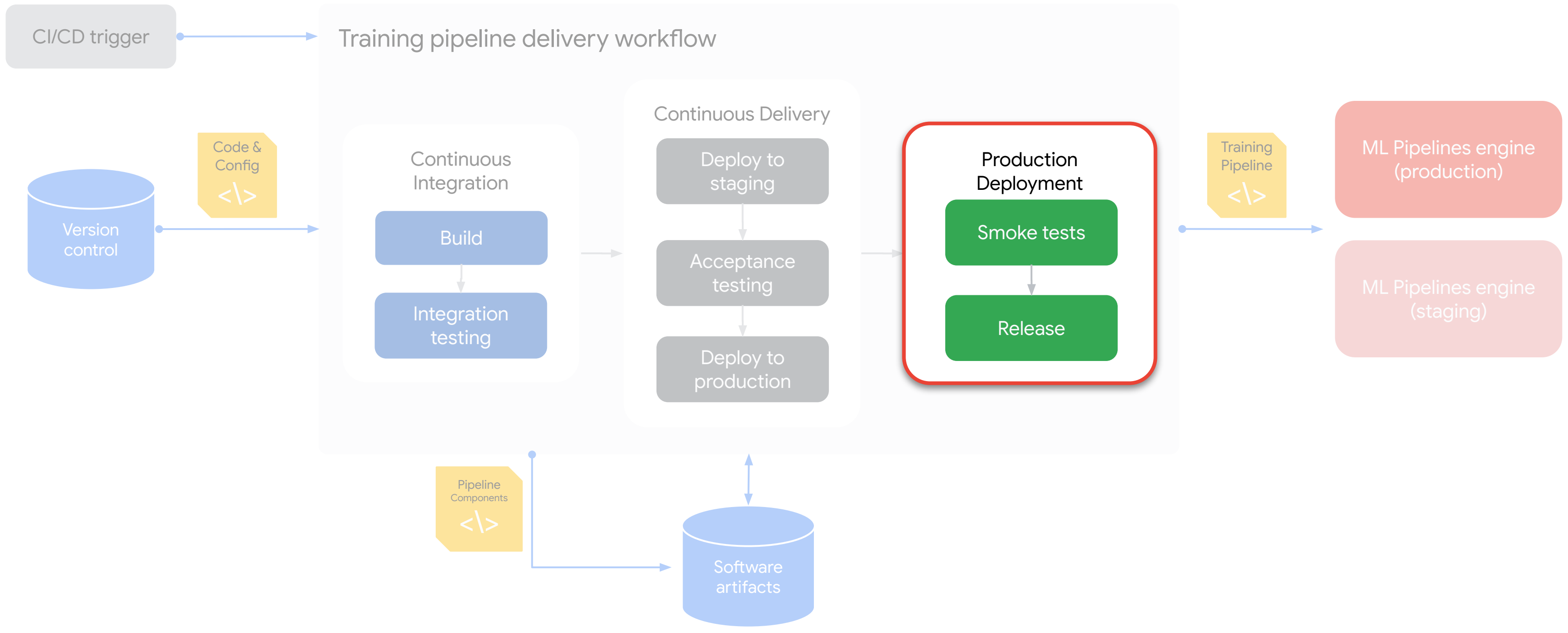
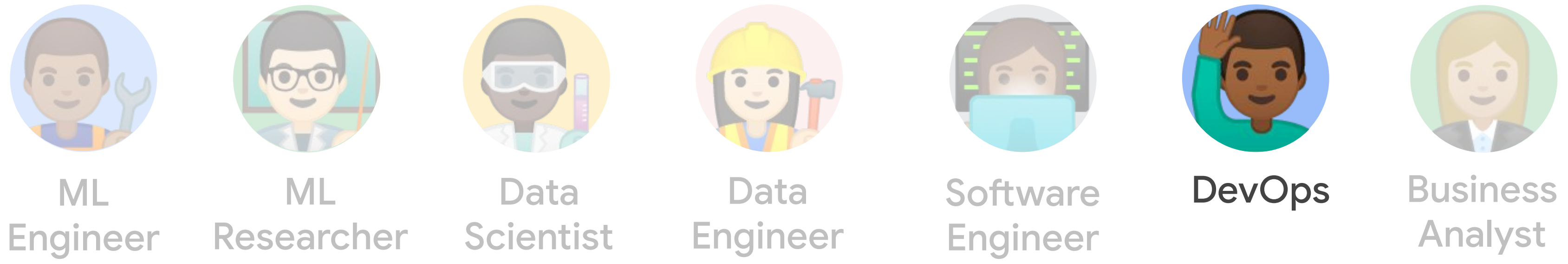
Production Deployment

MLOps: Training Operationalization



The MLOps **Personas**



Things go wrong even ***after*** testing?

Things go wrong even ***after*** testing?

1. **Assumed** testing environments is the **same** as the production environment

Things go wrong even ***after*** testing?

1. **Assumed** testing environments is the **same** as the production environment
2. **Hard to** make the test environment **match** the production environment

Things go wrong even ***after*** testing?

1. **Assumed** testing environments is the **same** as the production environment

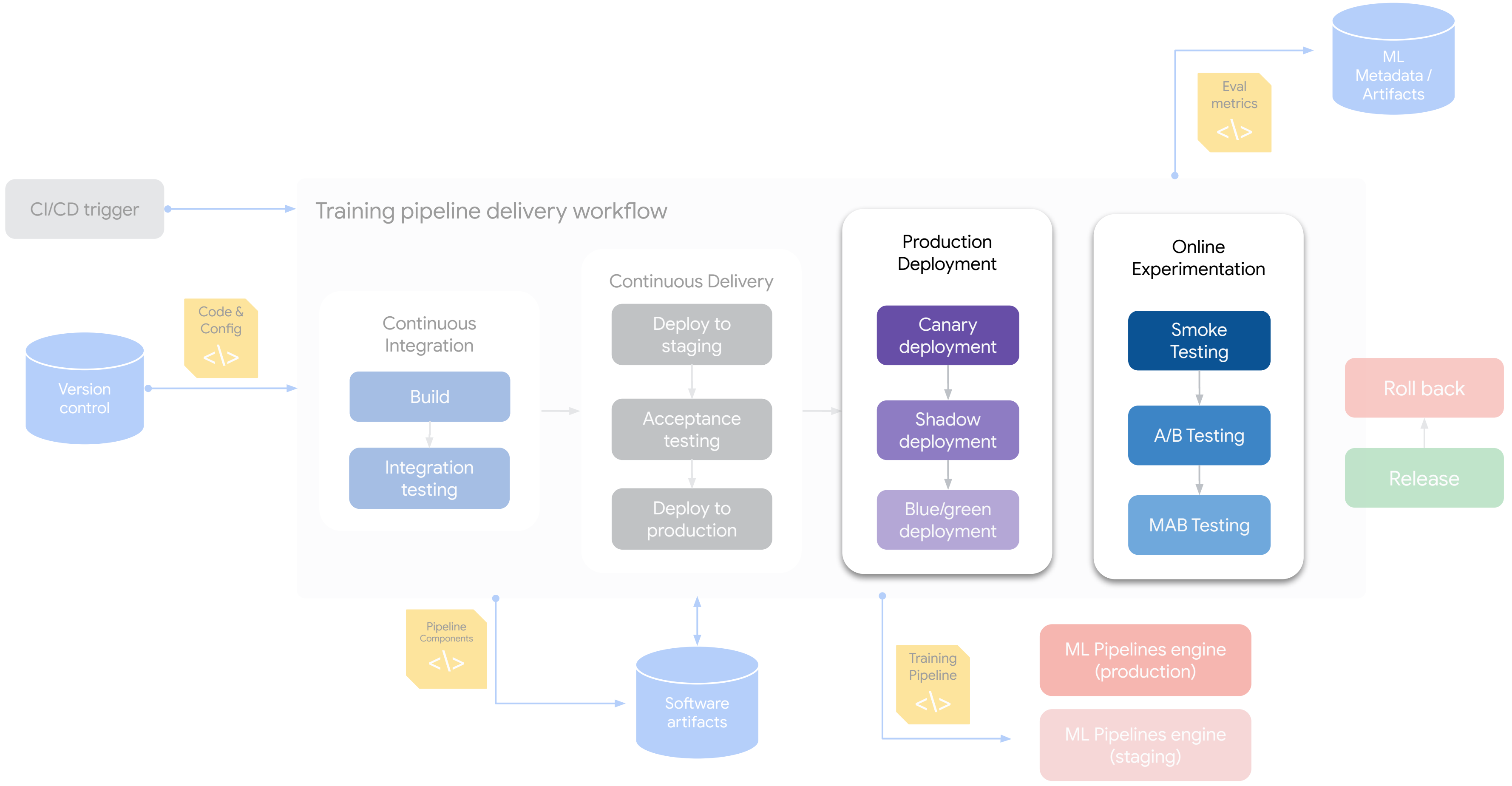
2. **Hard to** make the test environment **match** the production environment

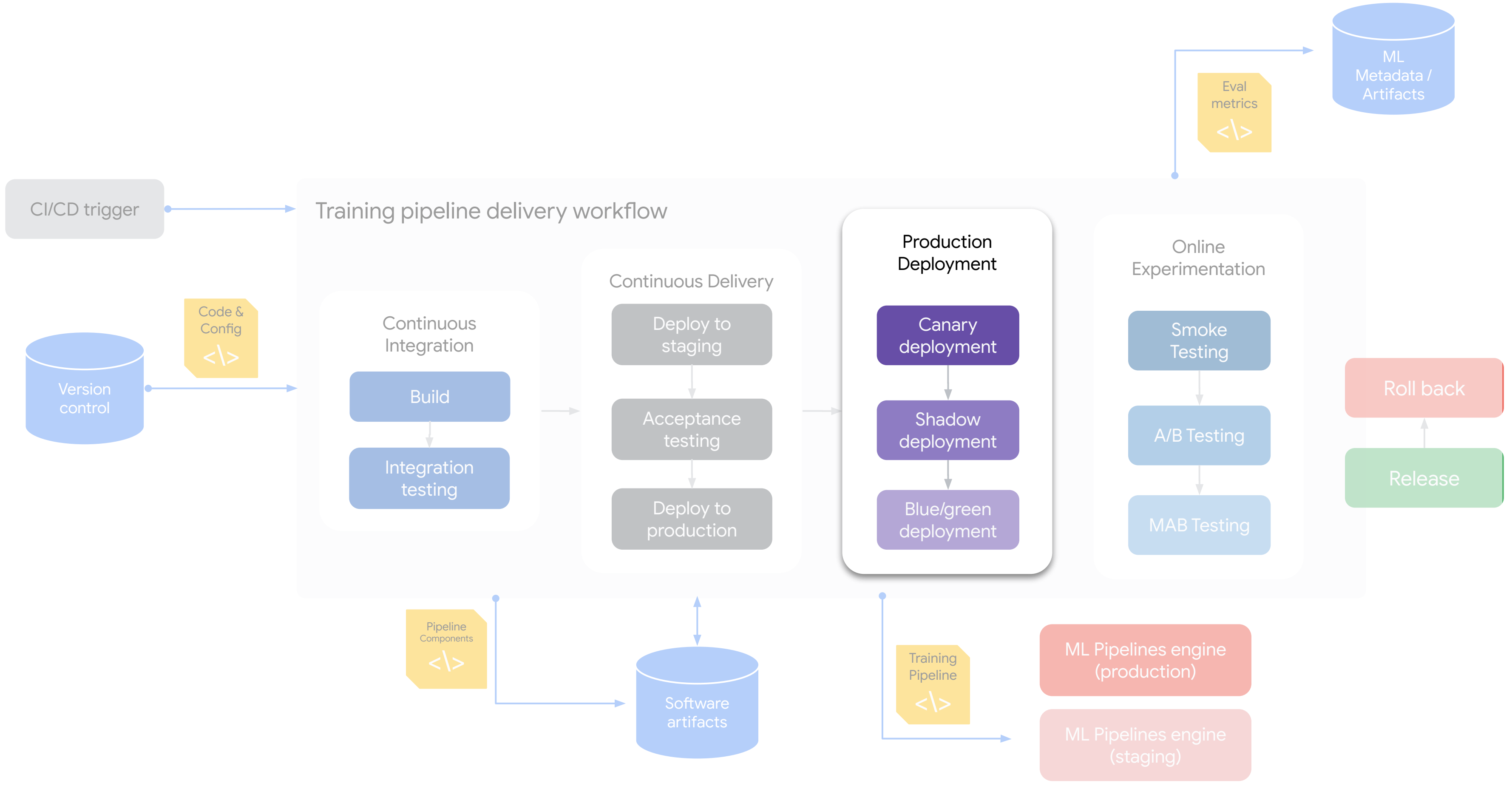
3. **Impossible to test all** the different **conditions** and scenarios in practice

Things go wrong even ***after*** testing?

1. **Assumed** testing environments is the **same** as the production environment
2. **Hard to** make the test environment **match** the production environment
3. **Impossible to test all** the different **conditions** and scenarios in practice

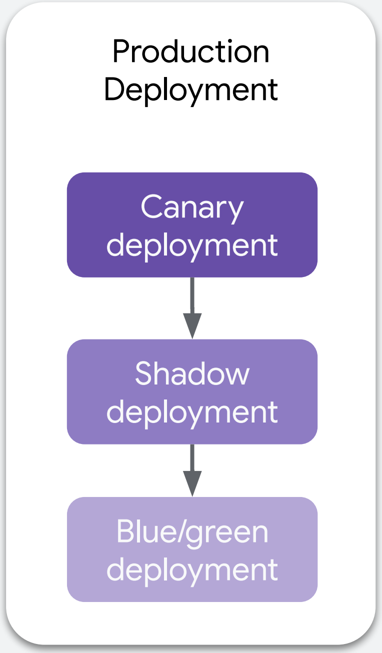
⇒ *“Test” live in production environment with real-traffic*


ML Metadata / Artifacts
Eval metrics
CI/CD trigger
Training pipeline delivery workflow
Continuous Delivery
Production Deployment
Online Experimentation
Code & Config
Version control
Continuous Integration
Build
Integration testing
Deploy to staging
Acceptance testing
Deploy to production
Canary deployment
Shadow deployment
Blue/green deployment
Smoke Testing
A/B Testing
MAB Testing
Roll back
Release
Pipeline Components
Software artifacts
Training Pipeline
ML Pipelines engine (production)
ML Pipelines engine (staging)

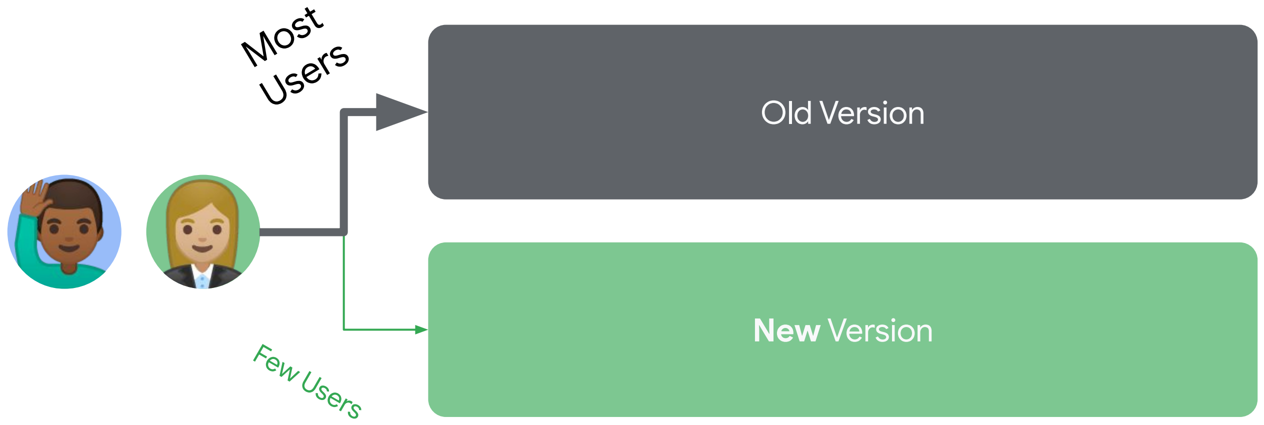

ML Metadata / Artifacts
Eval metrics
</>
CI/CD trigger
Training pipeline delivery workflow
Production Deployment
Online Experimentation
Continuous Delivery
Code & Config
</>
Continuous Integration
Deploy to staging
Canary deployment
Smoke Testing
Roll back
Version control
Build
Acceptance testing
Shadow deployment
A/B Testing
Release
Integration testing
Deploy to production
Blue/green deployment
MAB Testing
Pipeline Components
</>
Training Pipeline
</>
ML Pipelines engine (production)
Software artifacts
ML Pipelines engine (staging)

Production Deployment Strategies

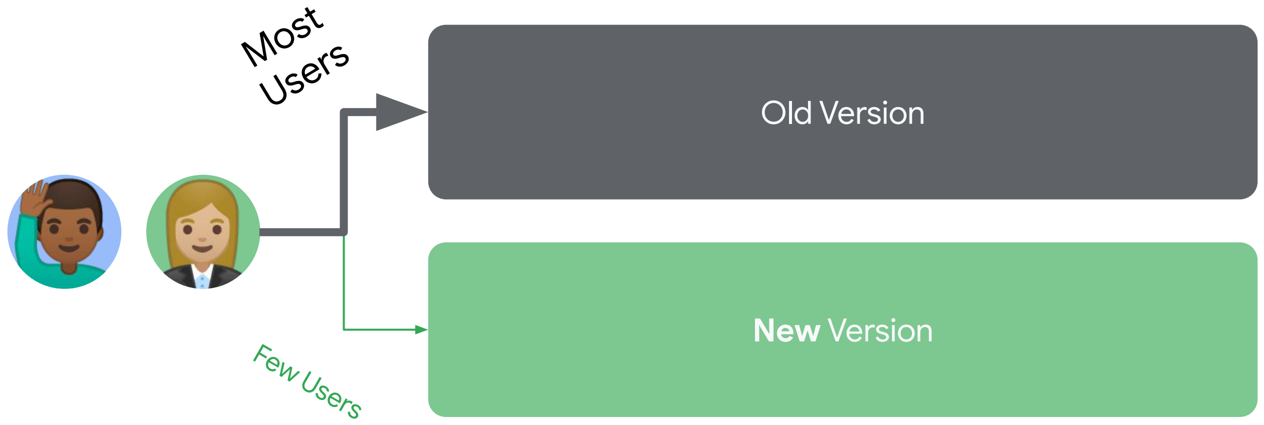
1. Canary Deployment
2. Shadow Deployment
3. Blue/Green Development



Canary Deployment

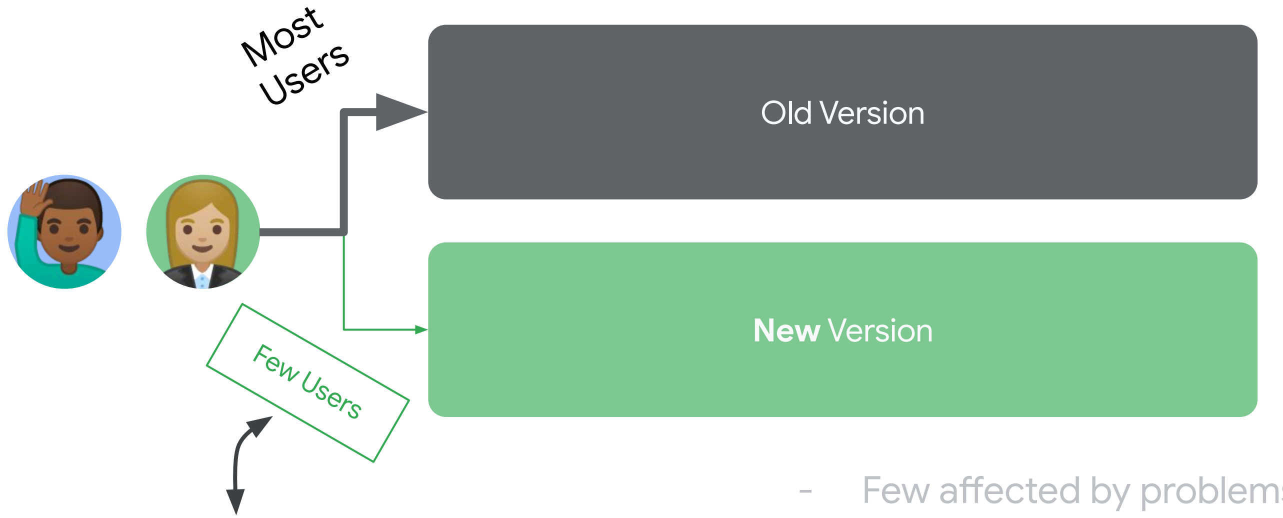


Canary Deployment



- Few affected by problems

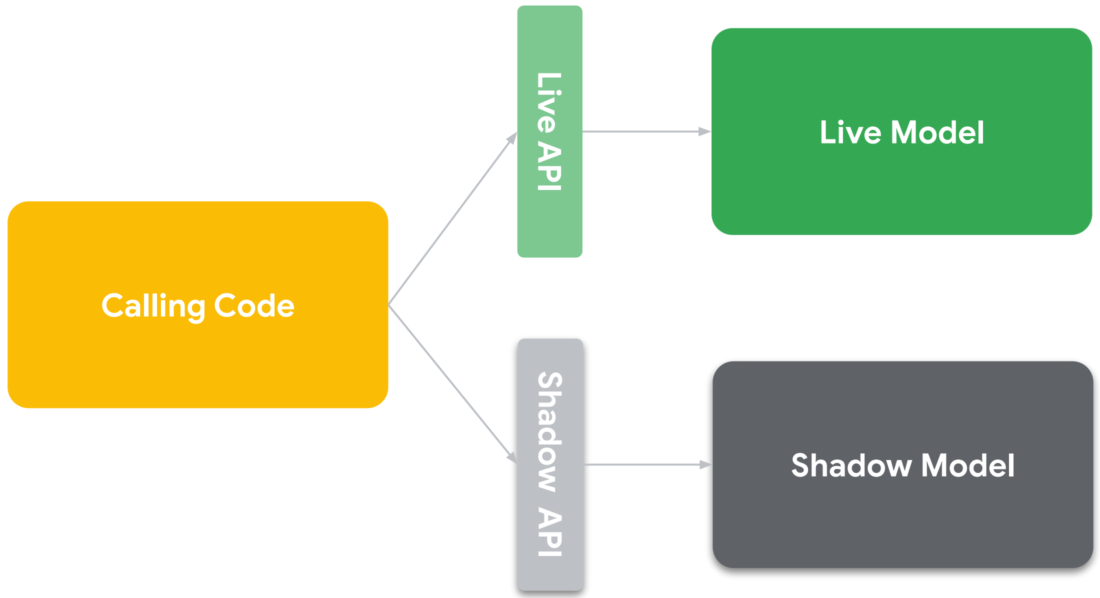
Canary Deployment



- Few affected by problems

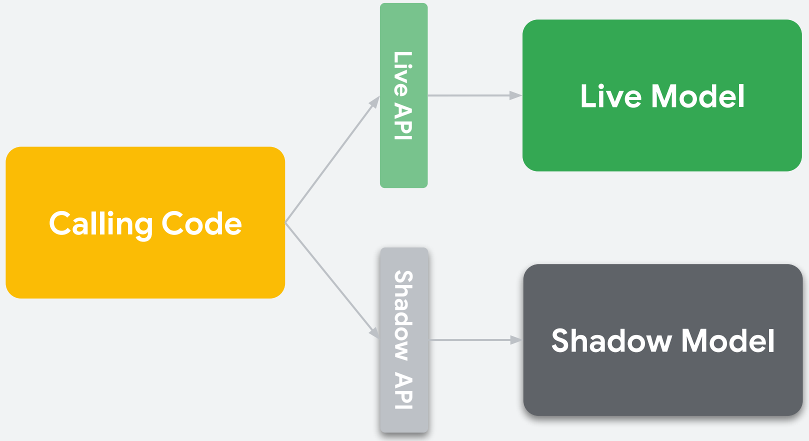
Is this a **representative** sample?

Shadow Deployment



Shadow Deployment

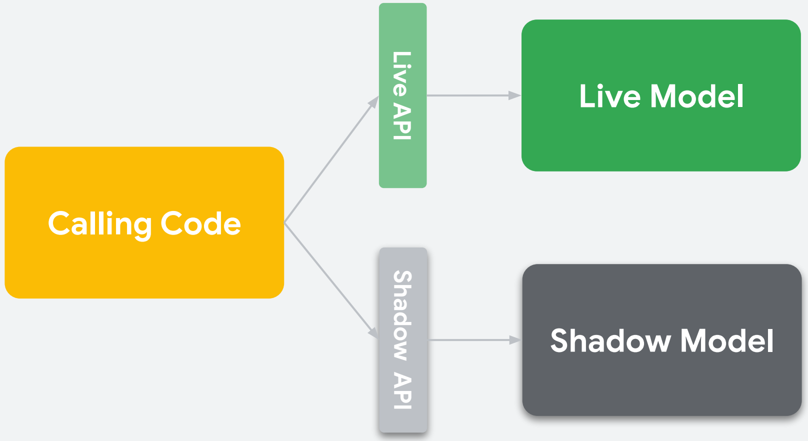
+ **No impact to production** traffic (customer)
+ **Check stability** (requirements)



Shadow Deployment

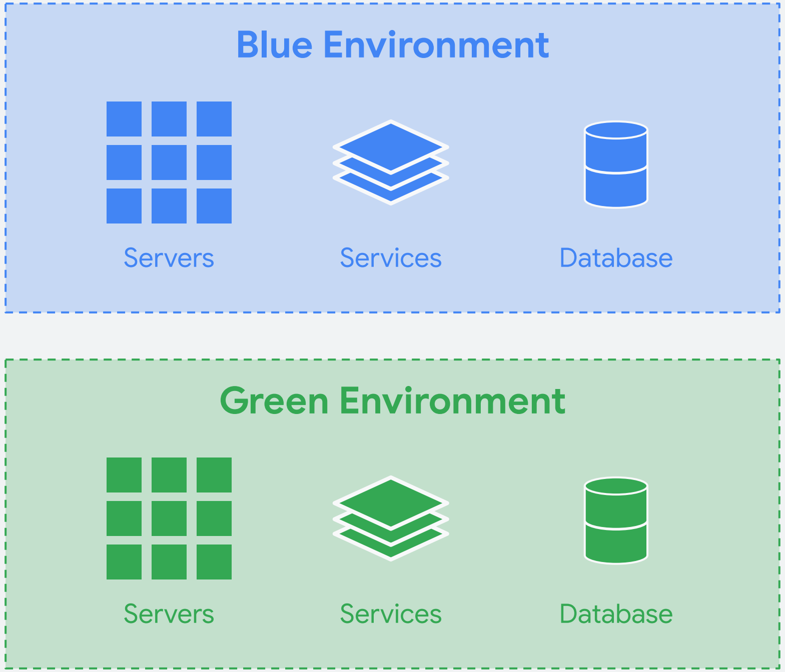
+ **No impact to production** traffic (customer)
+ **Check stability** (requirements)

- Increased **cost** (parallel)
- Complex **update-in-place**
- Need to carefully design **performance testing**



Blue/Green Deployment

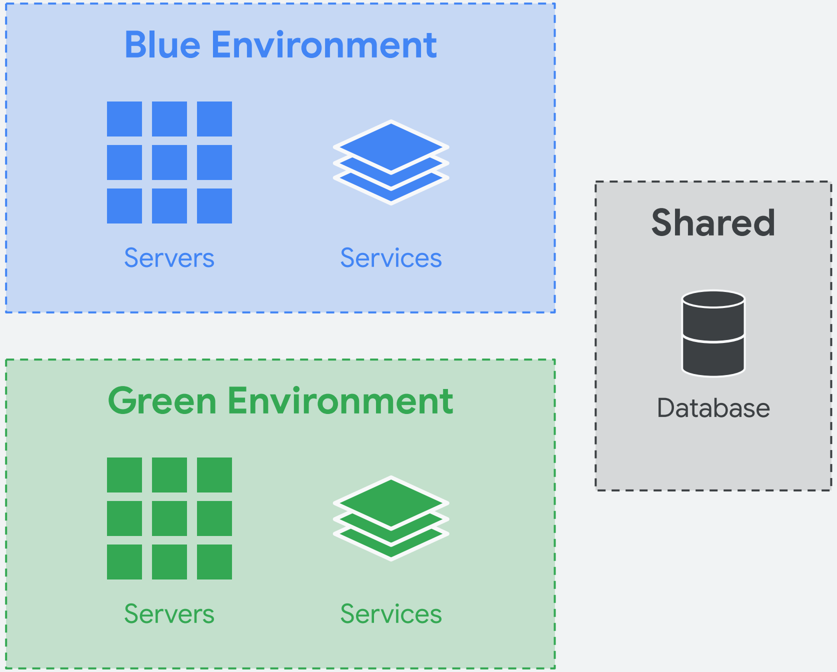
In its purest form, blue-green asks us to **duplicate** every resource our application leverages.



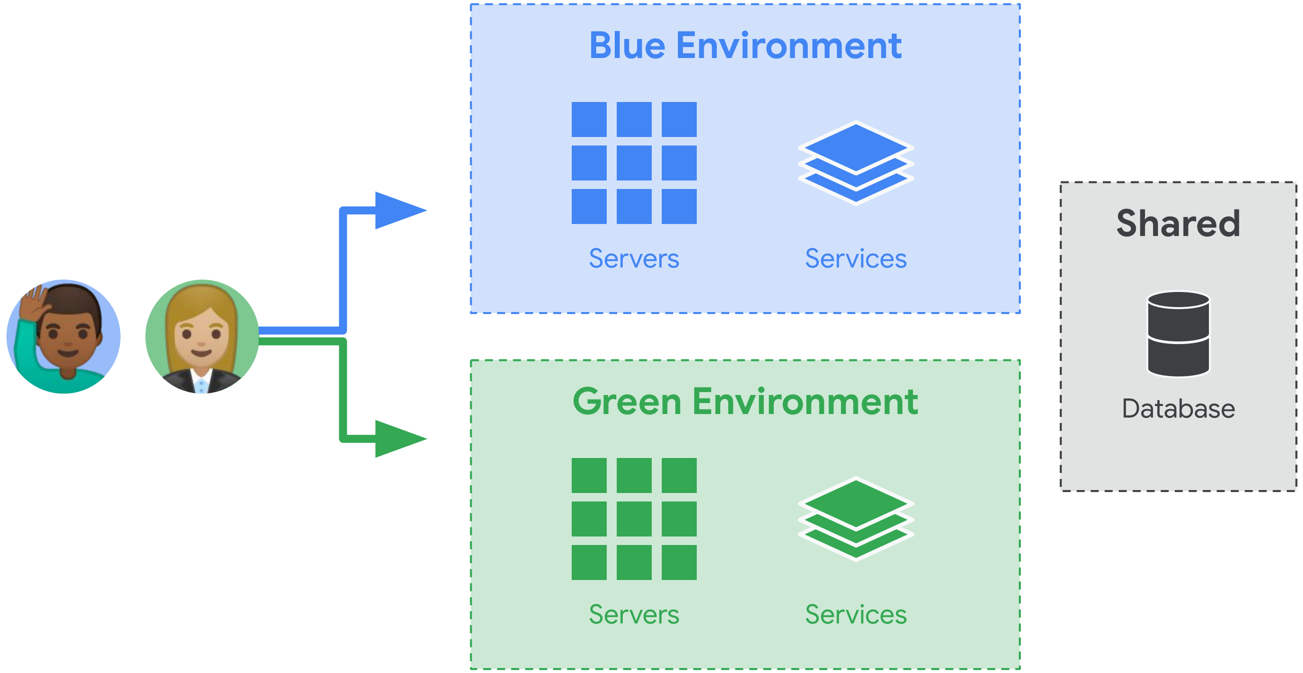
Blue/Green Deployment

In its purest form, blue-green asks us to **duplicate** every resource our application leverages.

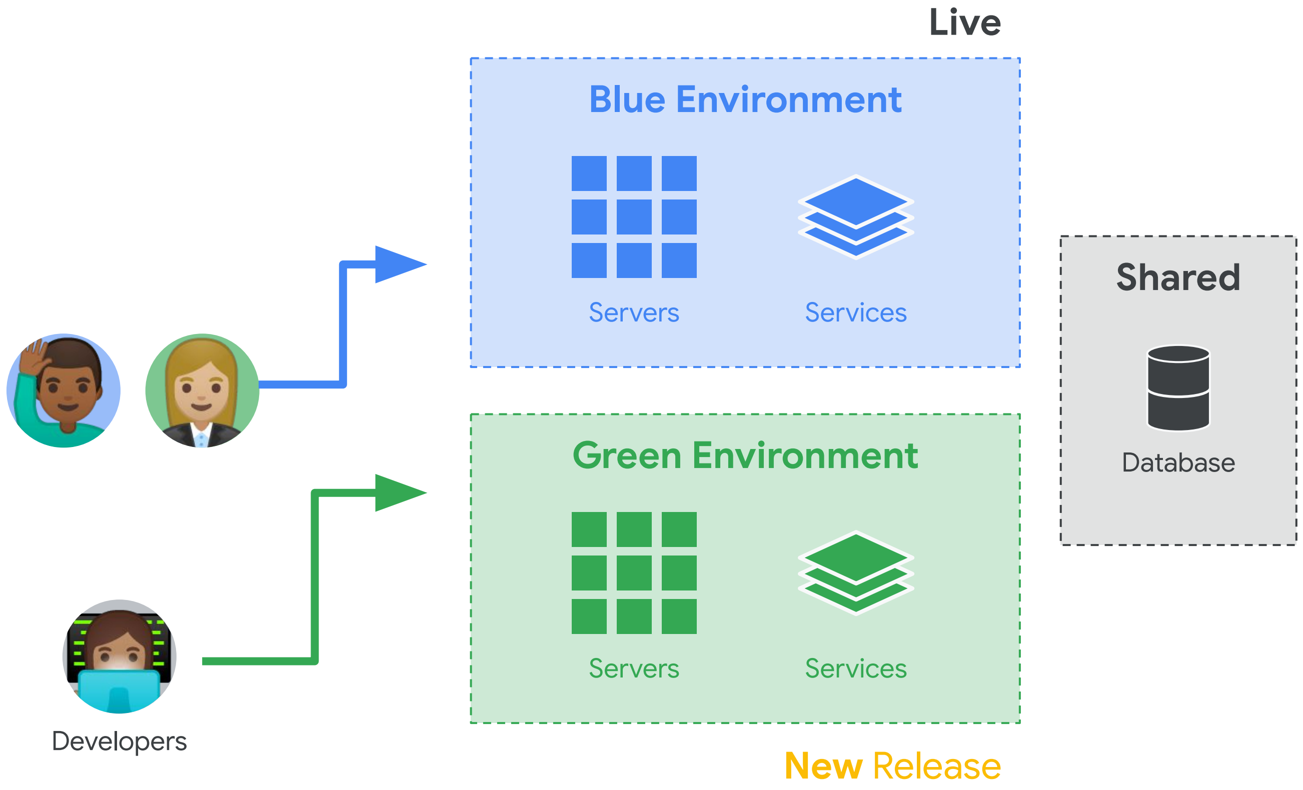
But, we often see blue-green deployments with **shared** components.



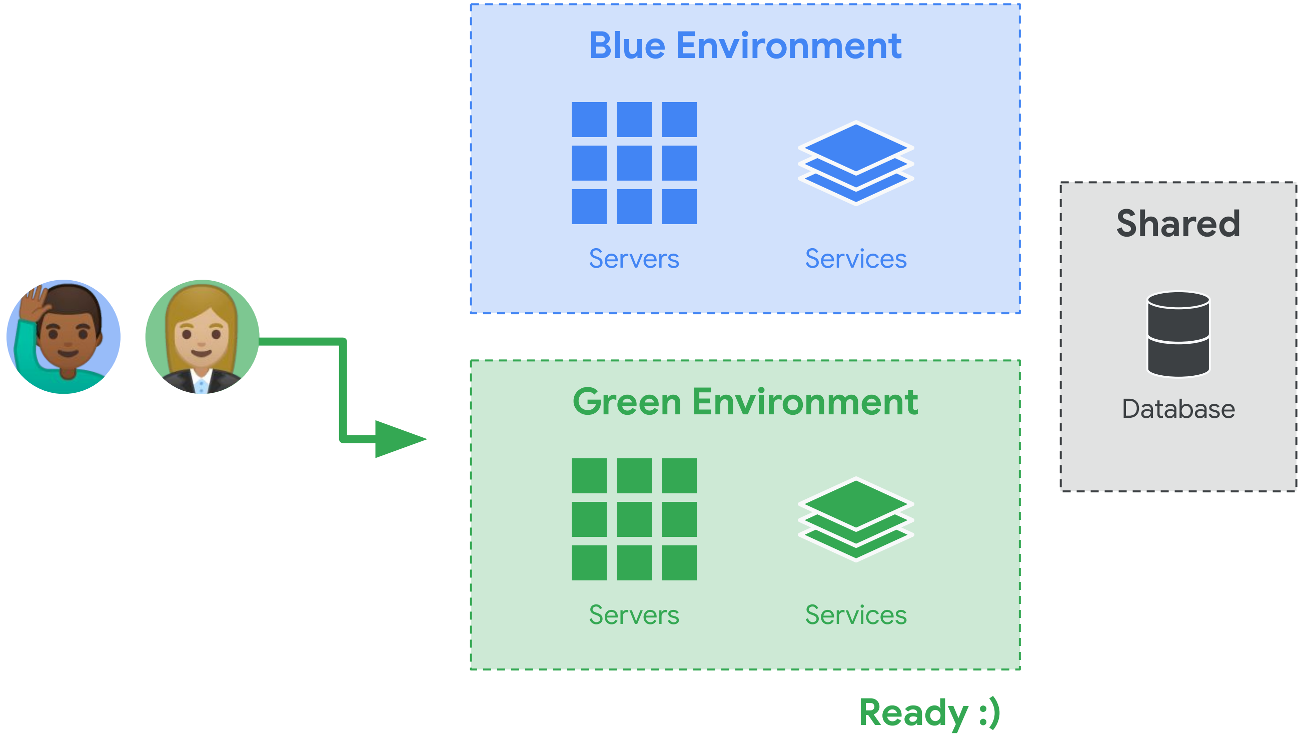
Blue/Green Deployment



Blue/Green Deployment: **Alternate Producers**

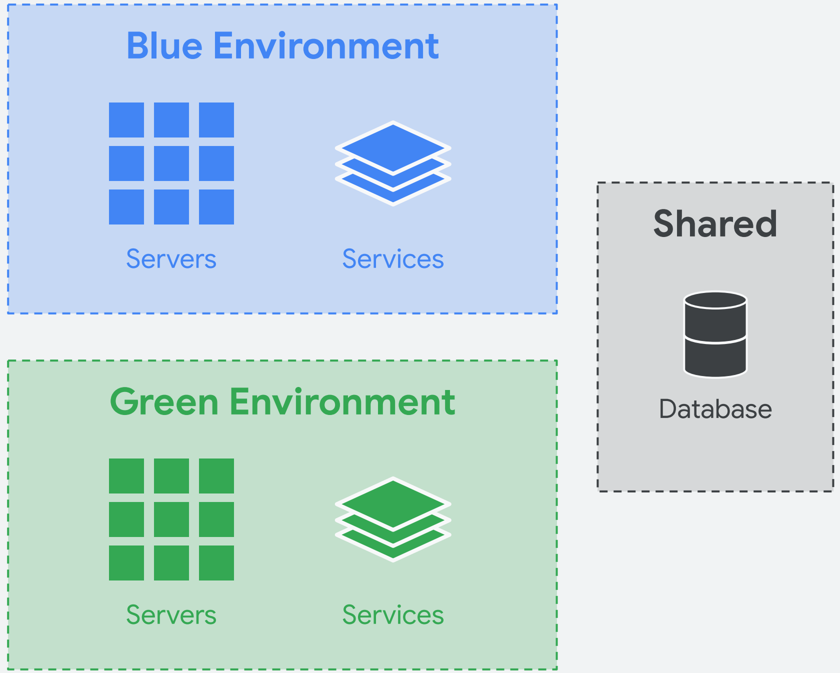


Blue/Green Deployment: **Alternate Producers**



Blue/Green Deployment

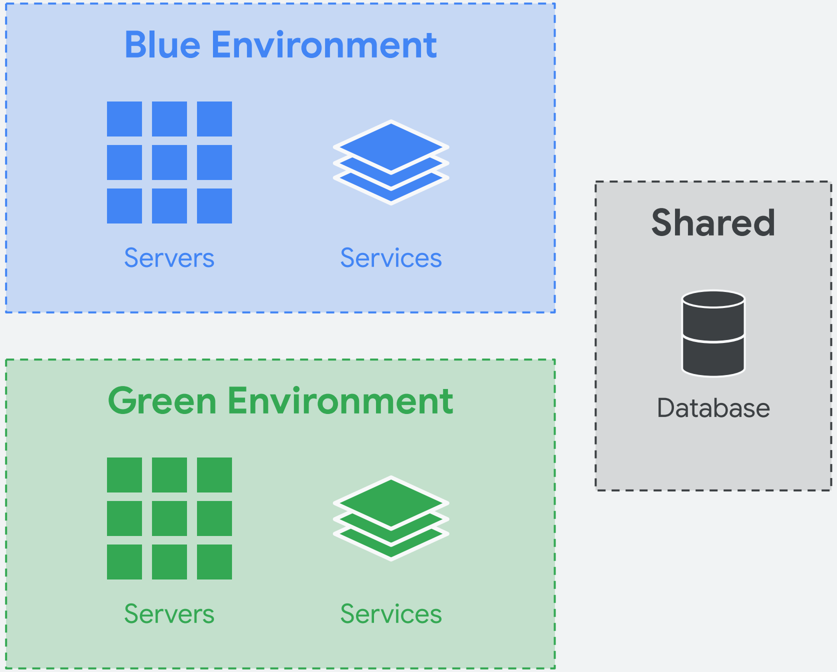
+ No system downtime
+ Accurate testing
+ Reliability testing of deployments



Blue/Green Deployment

For **TinyML**:

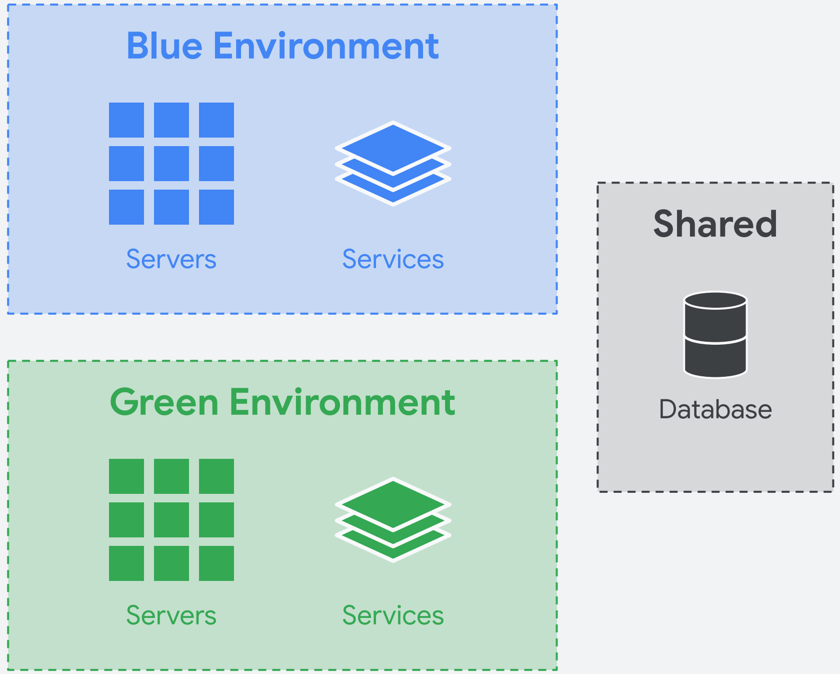
+ **When we can't make continuous updates**



Blue/Green Deployment

For **TinyML**:

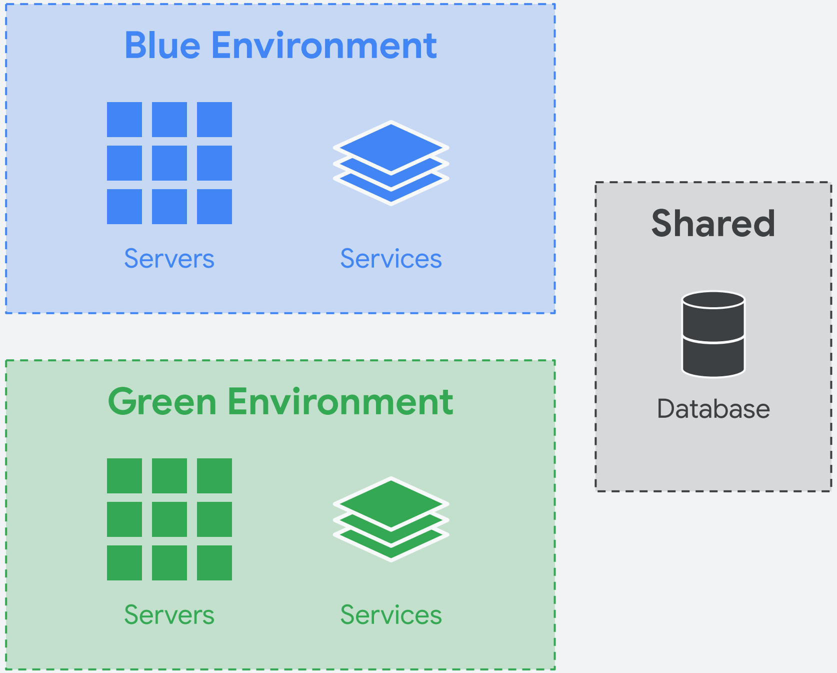
+ When we can't make continuous updates
+ **When we can't support two identical environments**

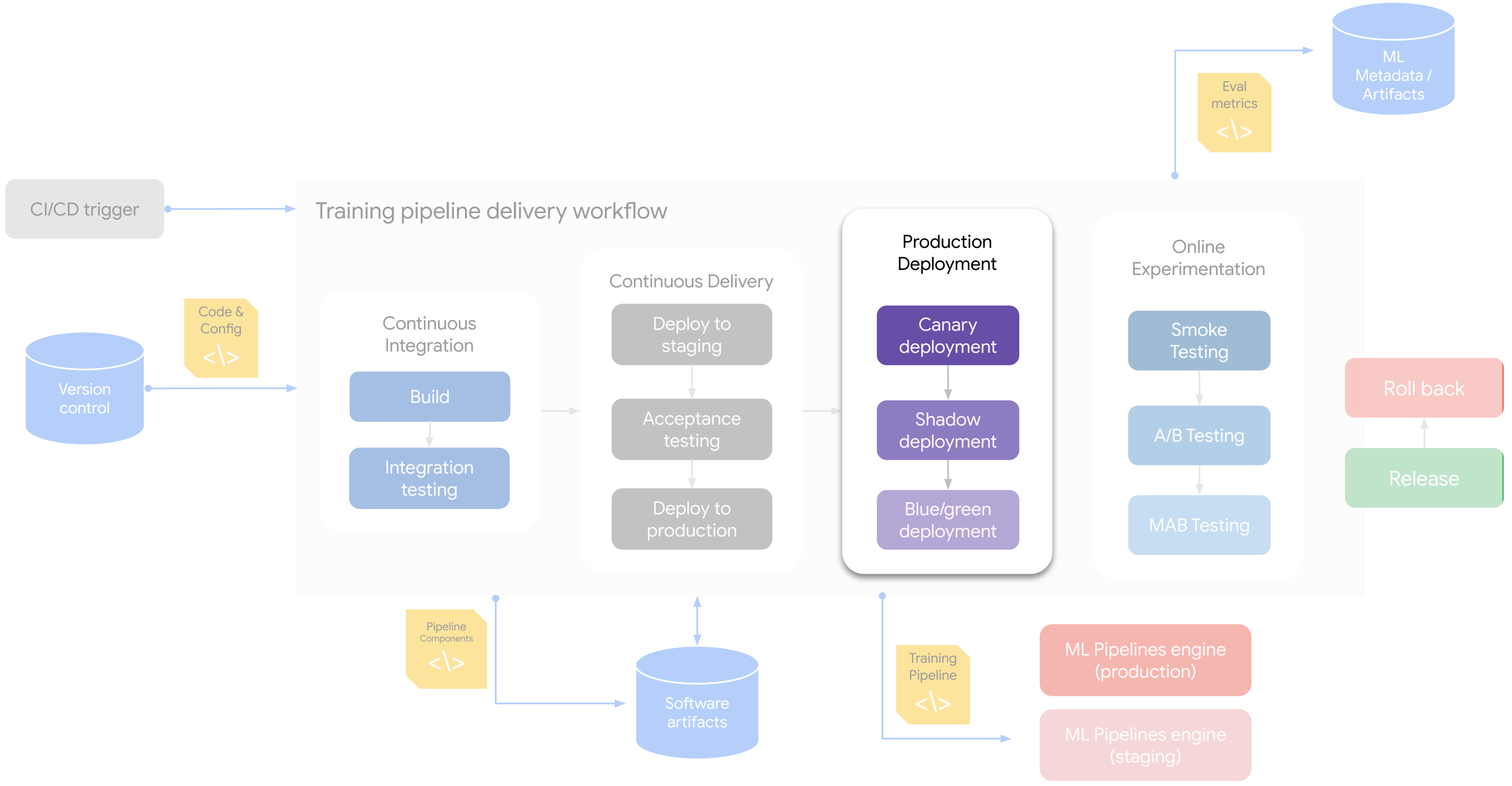


Blue/Green Deployment

For **TinyML**:

+ When we can't make continuous updates
+ When we can't support two identical environments
+ **When infrastructure does not allow a router to direct users**




CI/CD trigger
Training pipeline delivery workflow
Code & Config
Version control
Continuous Integration
Build
Integration testing
Continuous Delivery
Deploy to staging
Acceptance testing
Deploy to production
Production Deployment
Canary deployment
Shadow deployment
Blue/green deployment
Online Experimentation
Smoke Testing
A/B Testing
MAB Testing
Eval metrics
ML Metadata / Artifacts
Roll back
Release
Pipeline Components
Software artifacts
Training Pipeline
ML Pipelines engine (production)
ML Pipelines engine (staging)