How does the ideal CI/CD pipeline look?

Viktor Vedmich | AWS

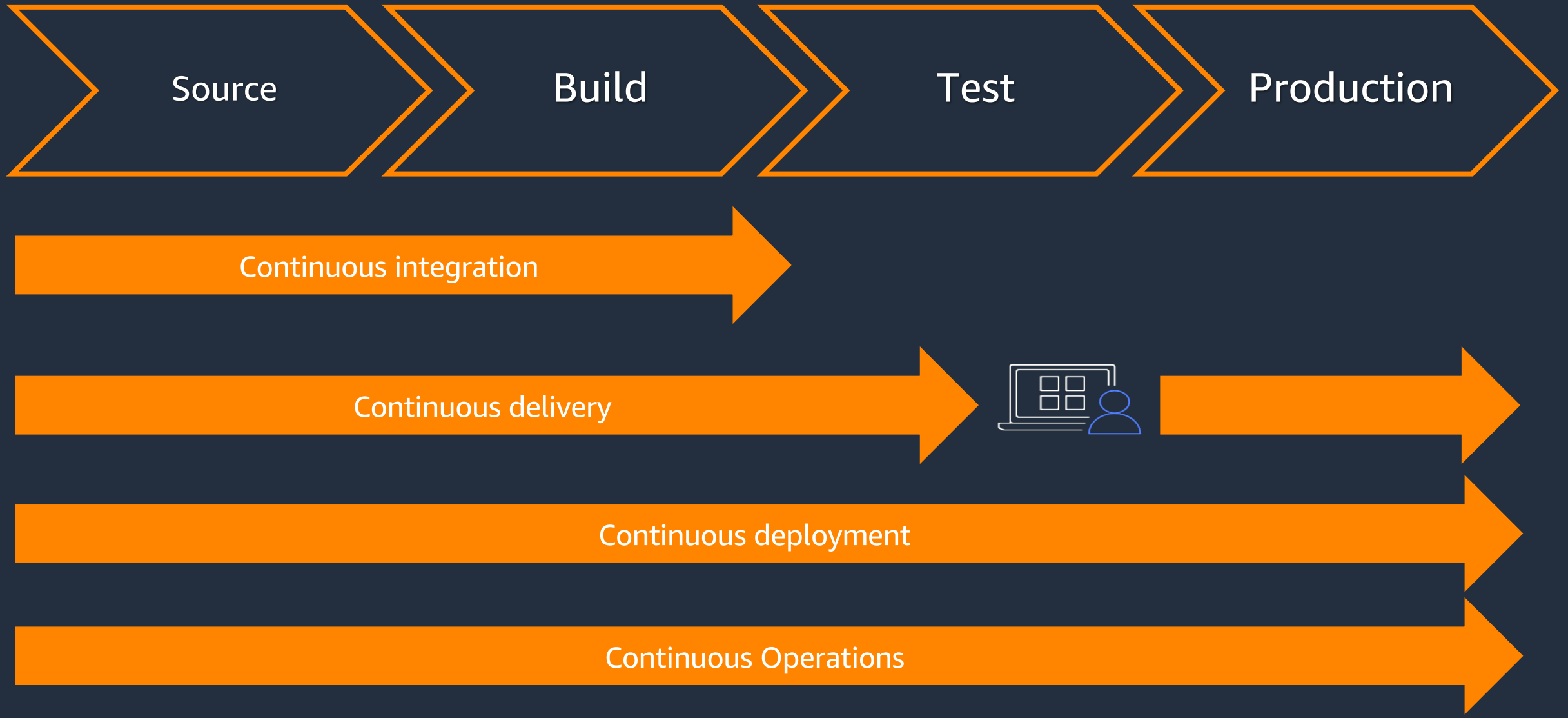
Snr Developer Advocate

Agenda

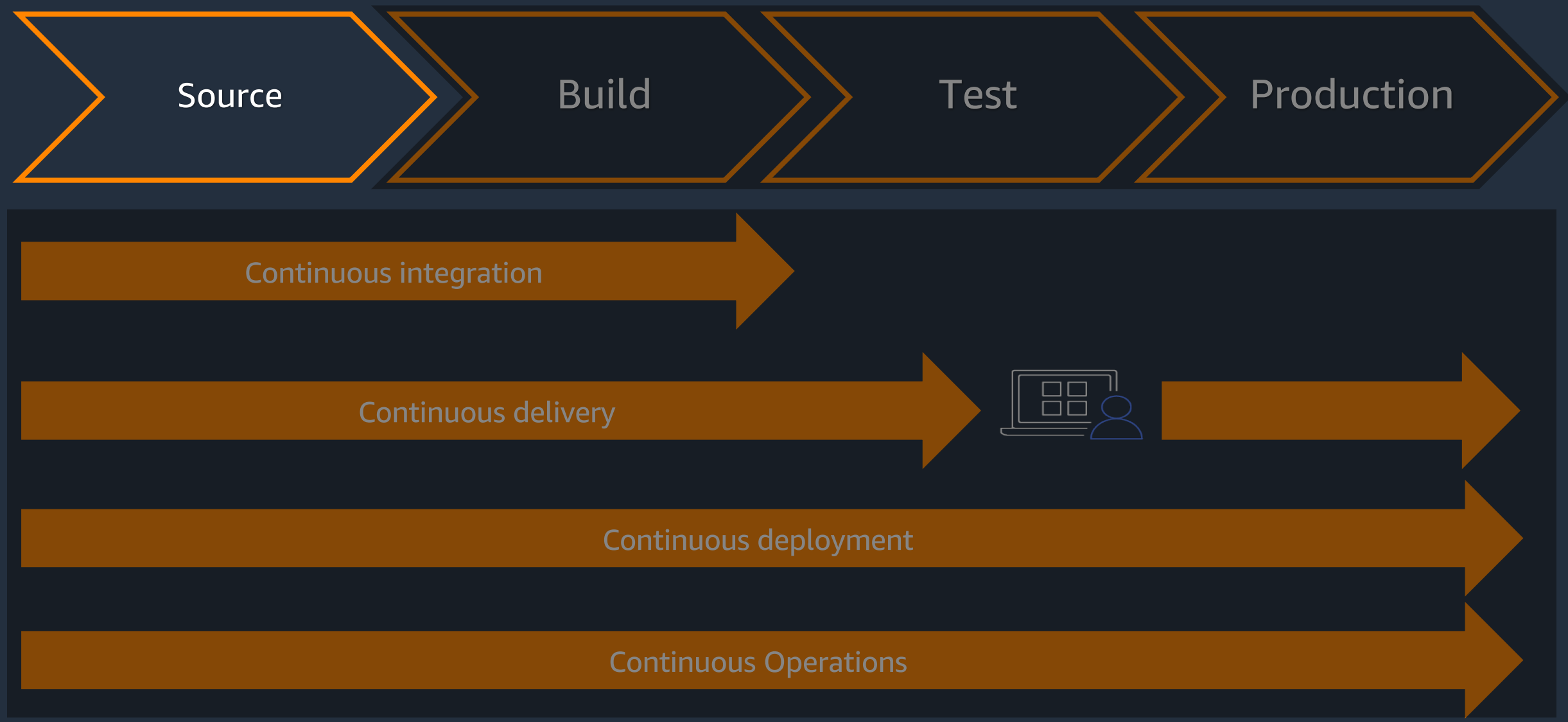
- Release Lifecycle
- Branching Strategy
- CI/CD Journey
- Quality Gates
- Metrics
- Final Blueprint

Overview Release Lifecycle

Release lifecycle

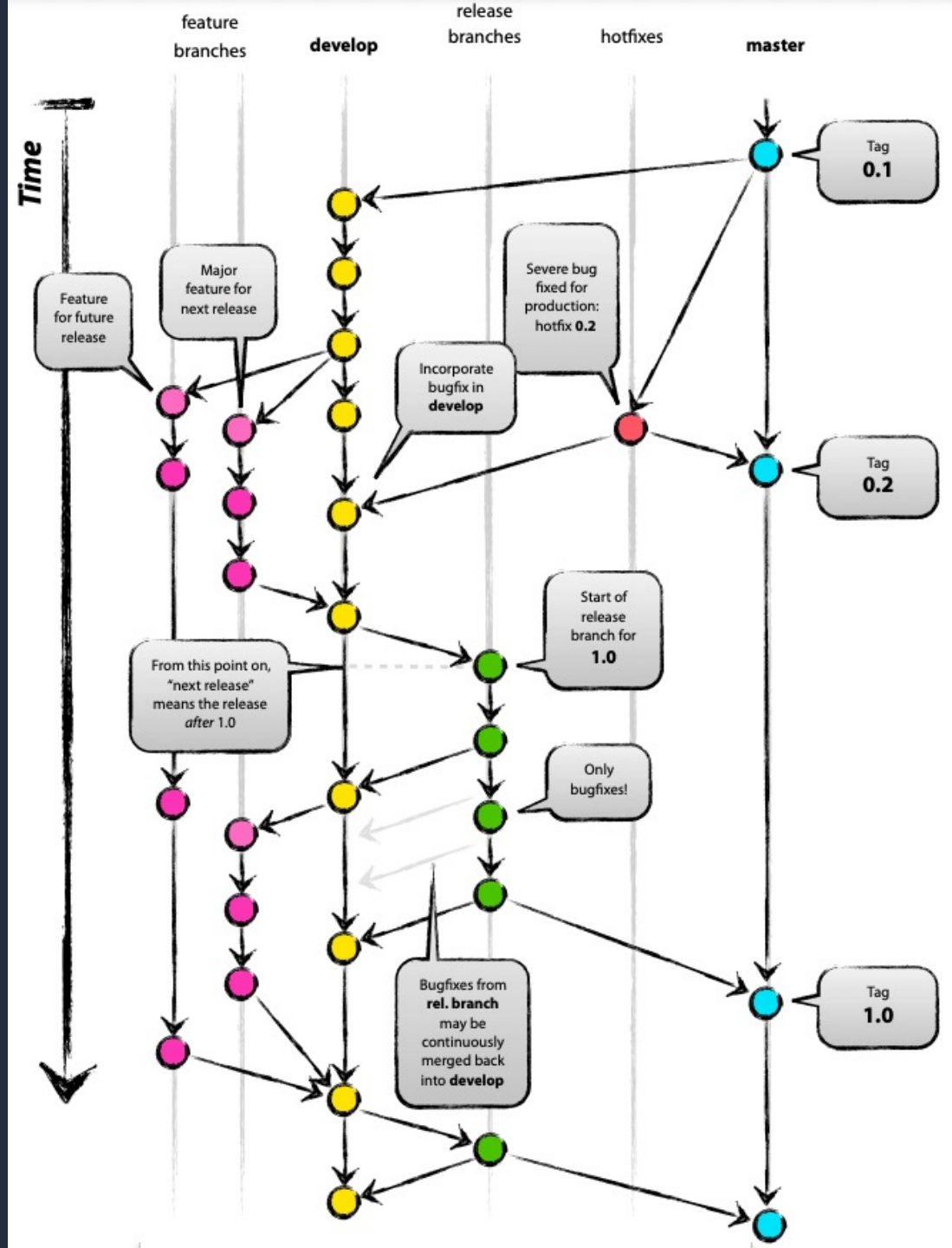


Release lifecycle



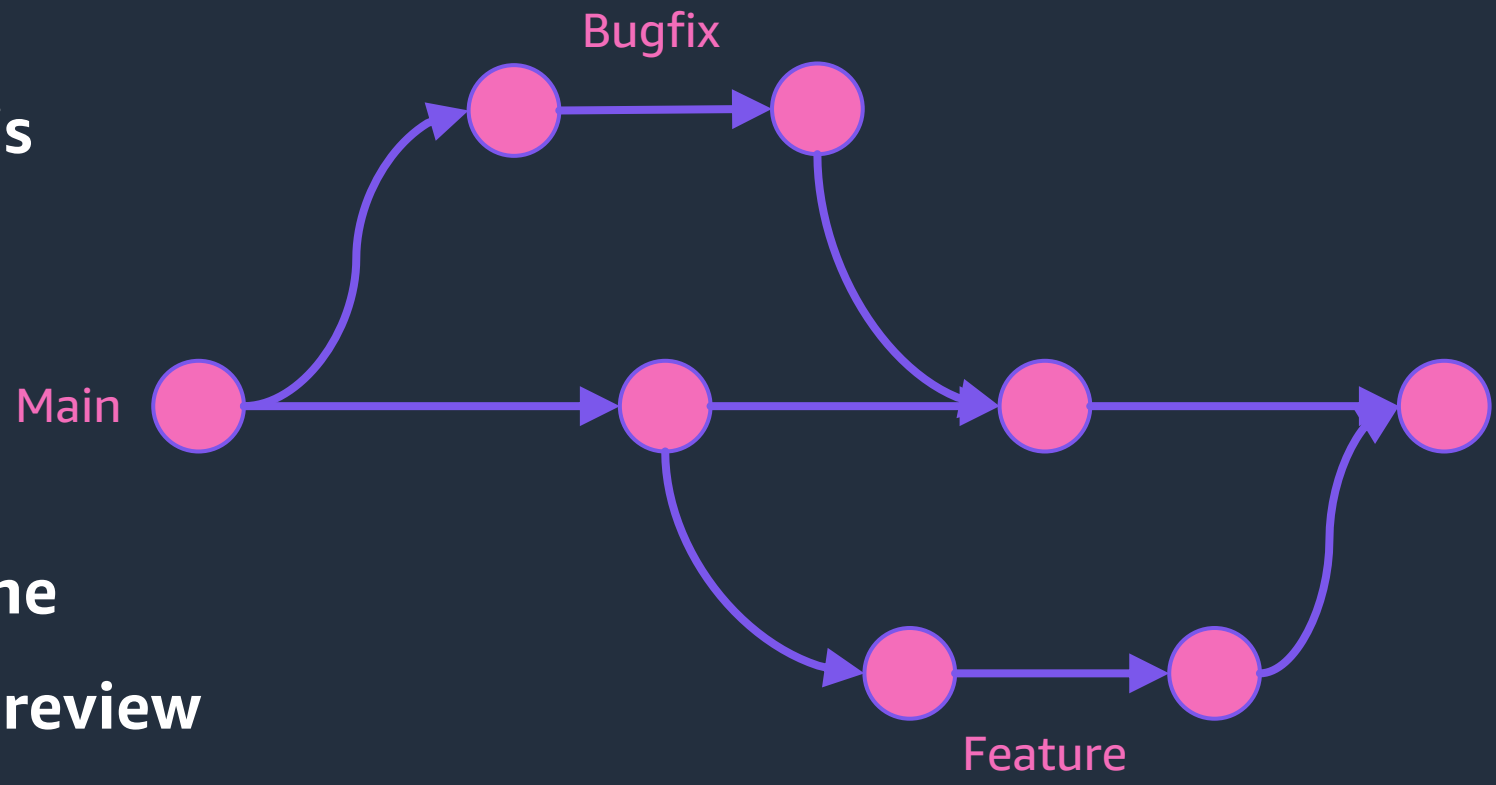
Branching Strategy step 0 for CI/CD

GitFlow created by Vincent Driessen 2010!



GitHub Flow

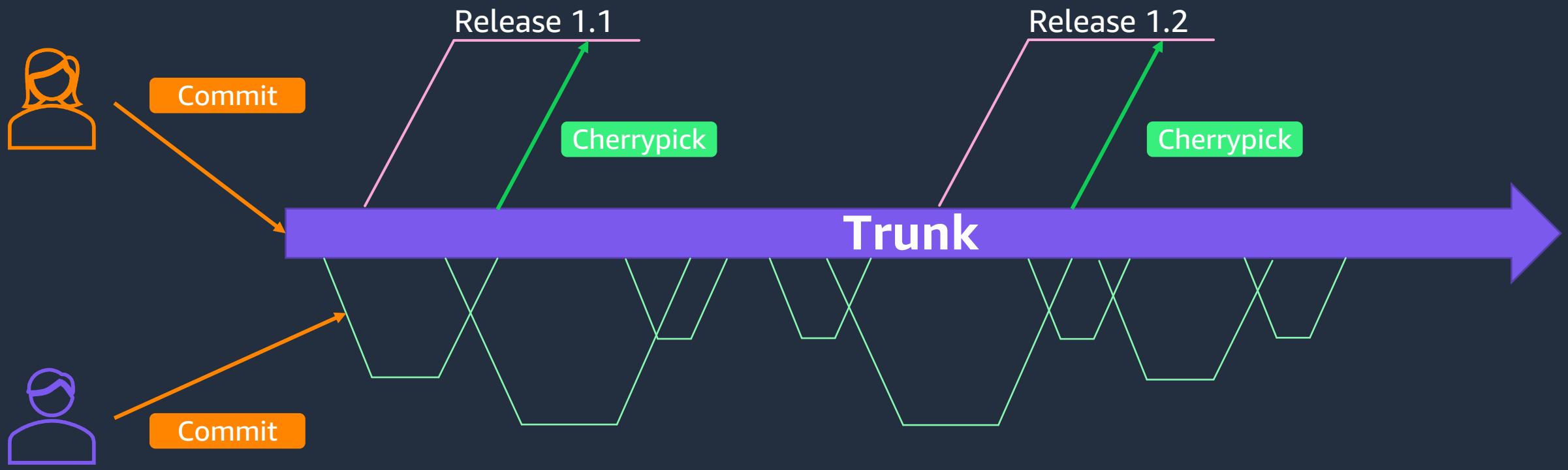
1. Anything in the main branch is deployable
2. Short lived branches
3. Code review
4. Open a pull request at any time
5. Merge only after pull request review



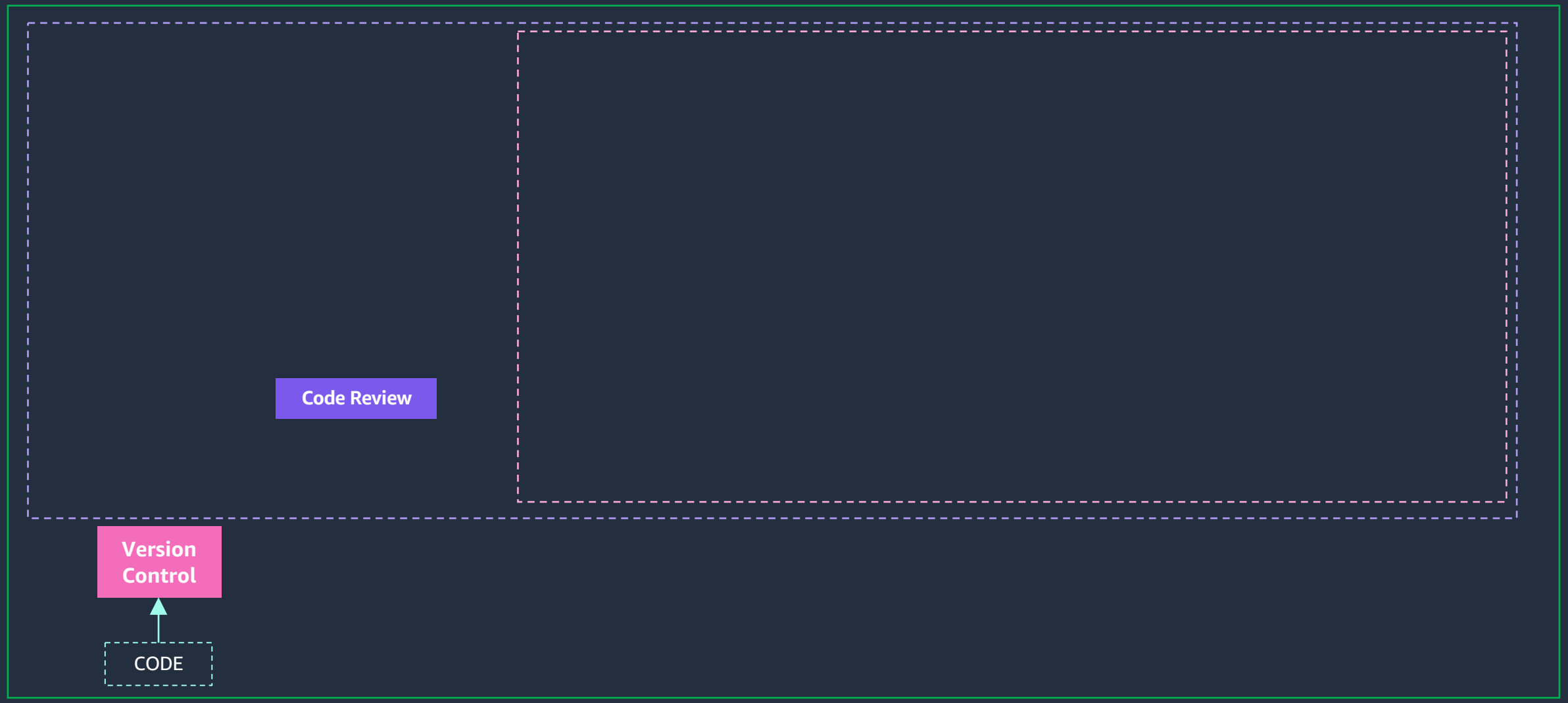
Branching strategy: Trunk-Based Development

- Commit in trunk at least once a day
- Trunk is always is a releasable state

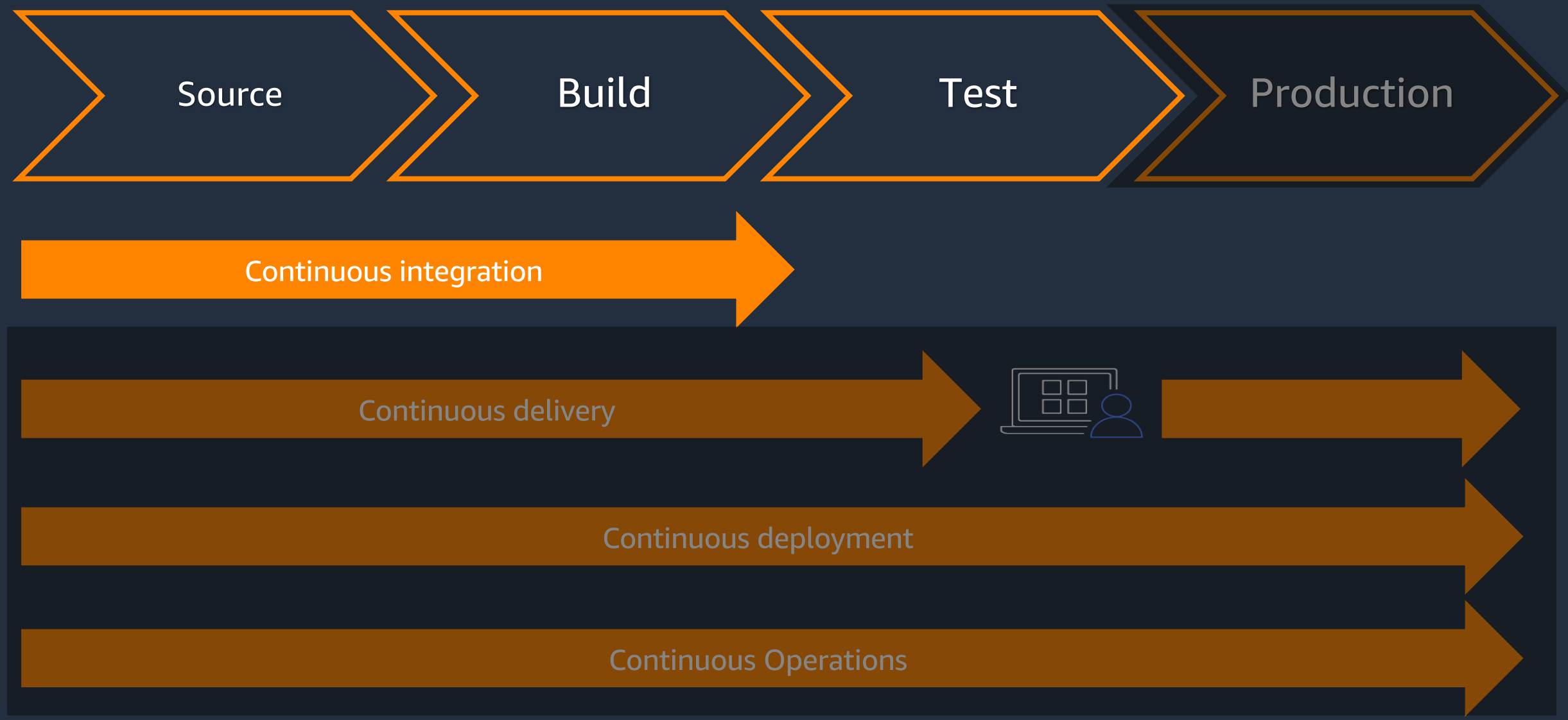
- Hide unfinished code with feature toggle
- Refactor with branch by abstraction



CI/CD blueprint



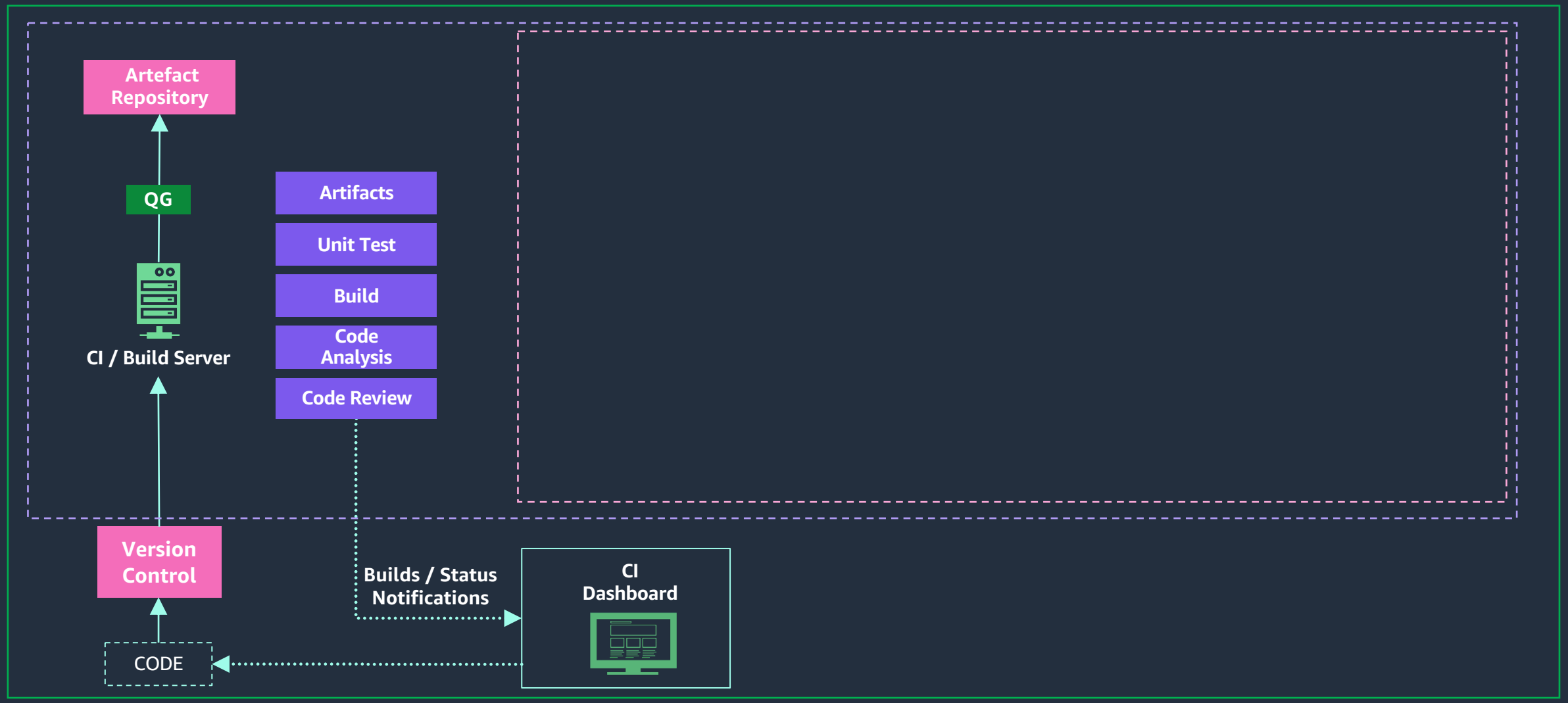
Release lifecycle



Continuous Integration goals

1. Automatically kick off a new build when new code is checked in
2. Build and test code in a consistent, repeatable environment
3. Continually have an artifact ready for deployment
4. Continually close feedback loop when build fails

Continuous Integration



aws

Quality gateway

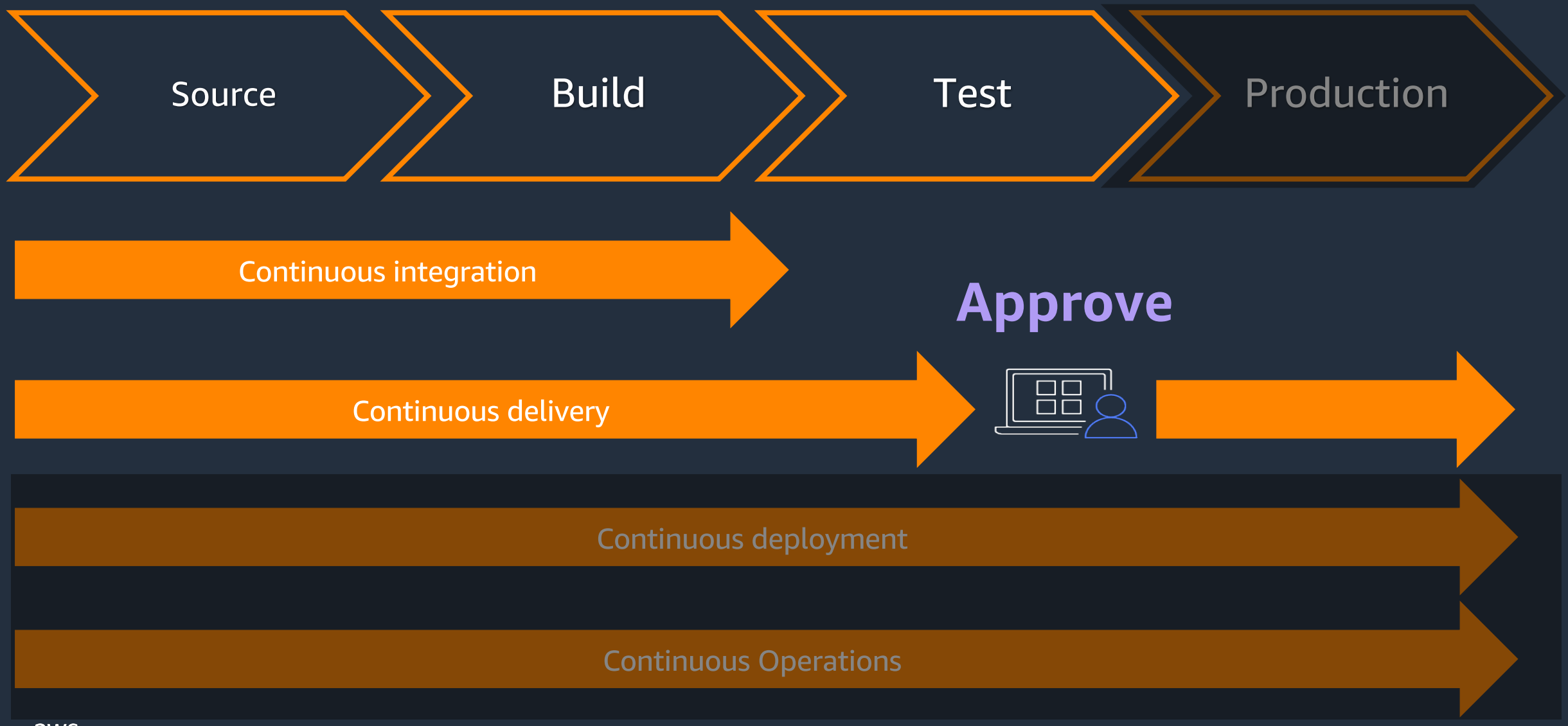
Semantic Versioning 2.0.0

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

PATCH version can be your hash of the commit in Git.

Release lifecycle

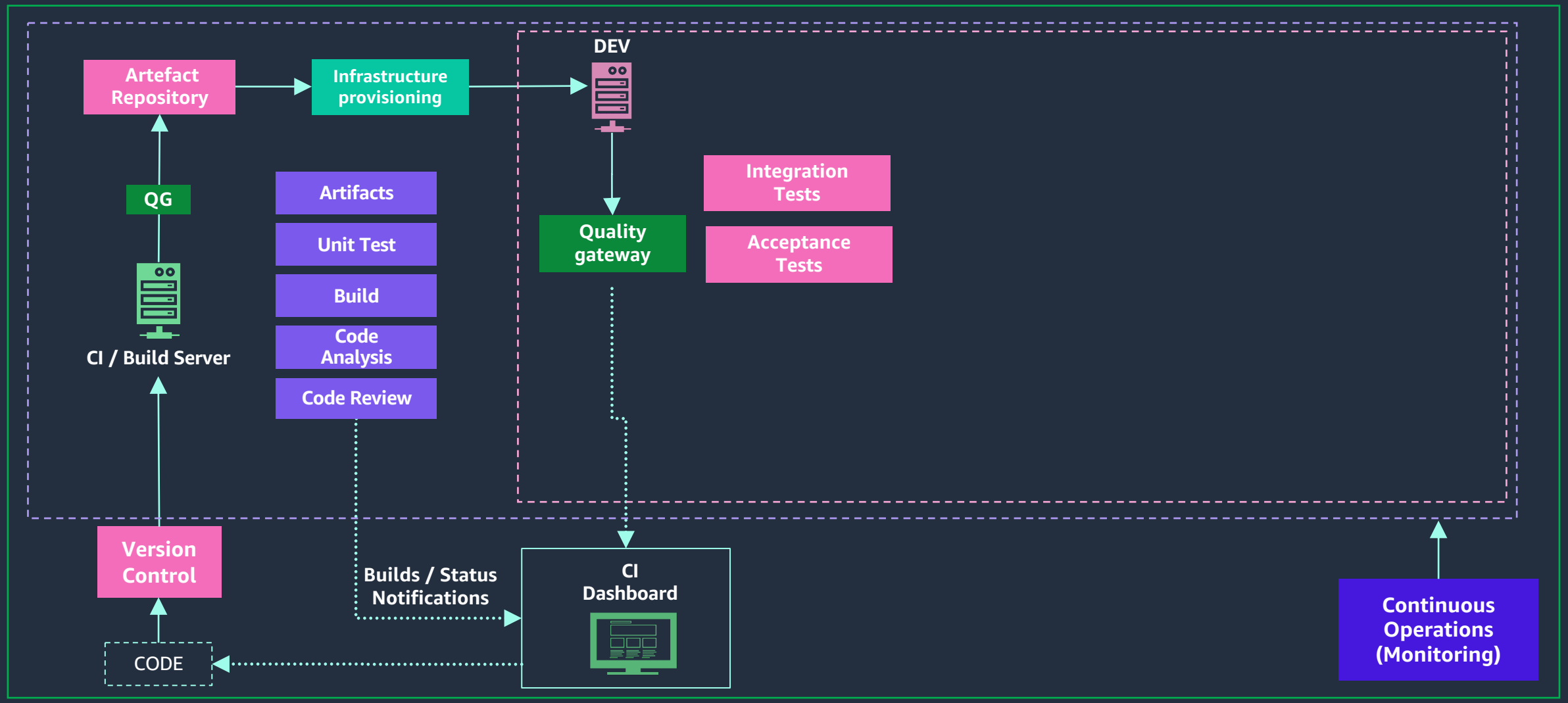


Continuous Delivery

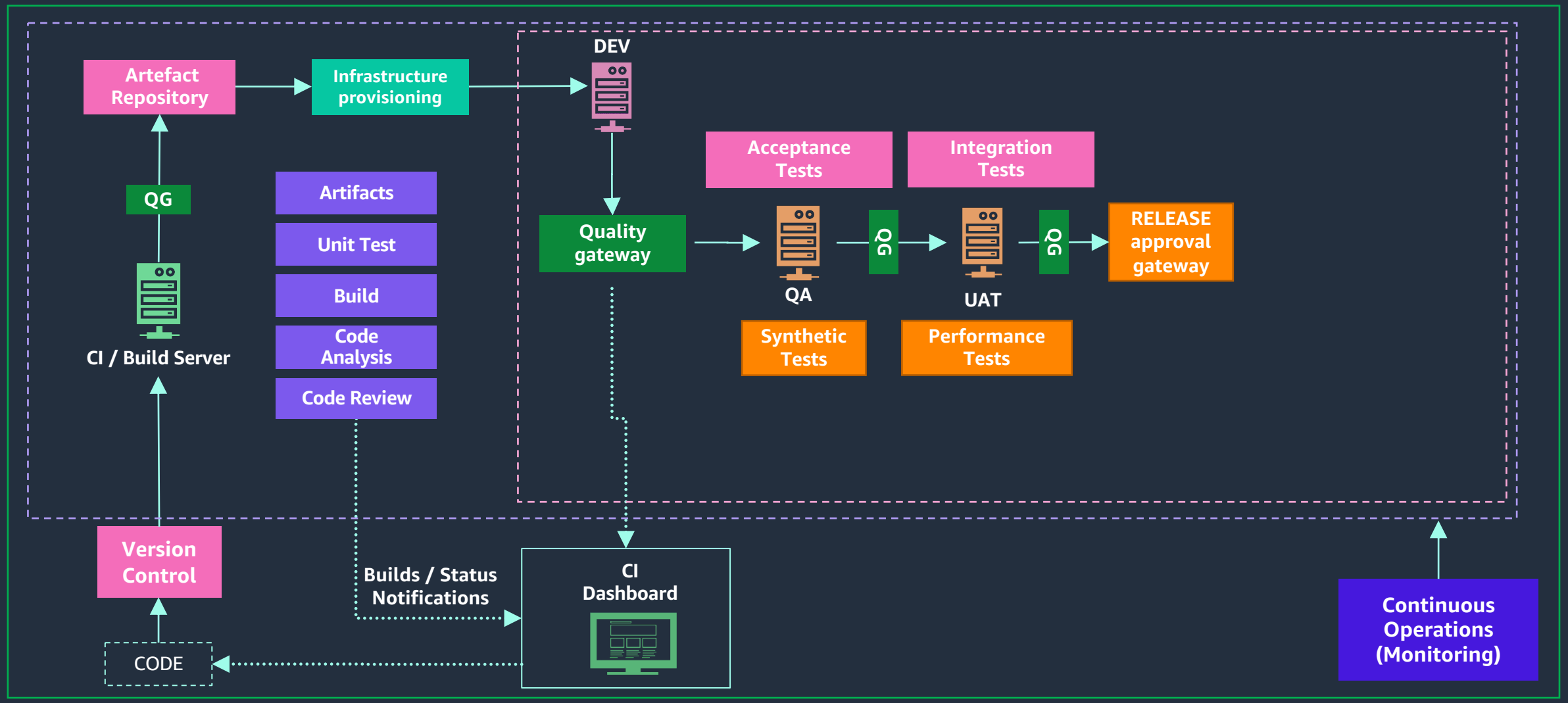
Continuous Integration +

- Automated pipeline that handles deployment activities
- Each change is a release candidate
- Automated deployment pipeline promote build **artifact**
- Production deployments triggered manually

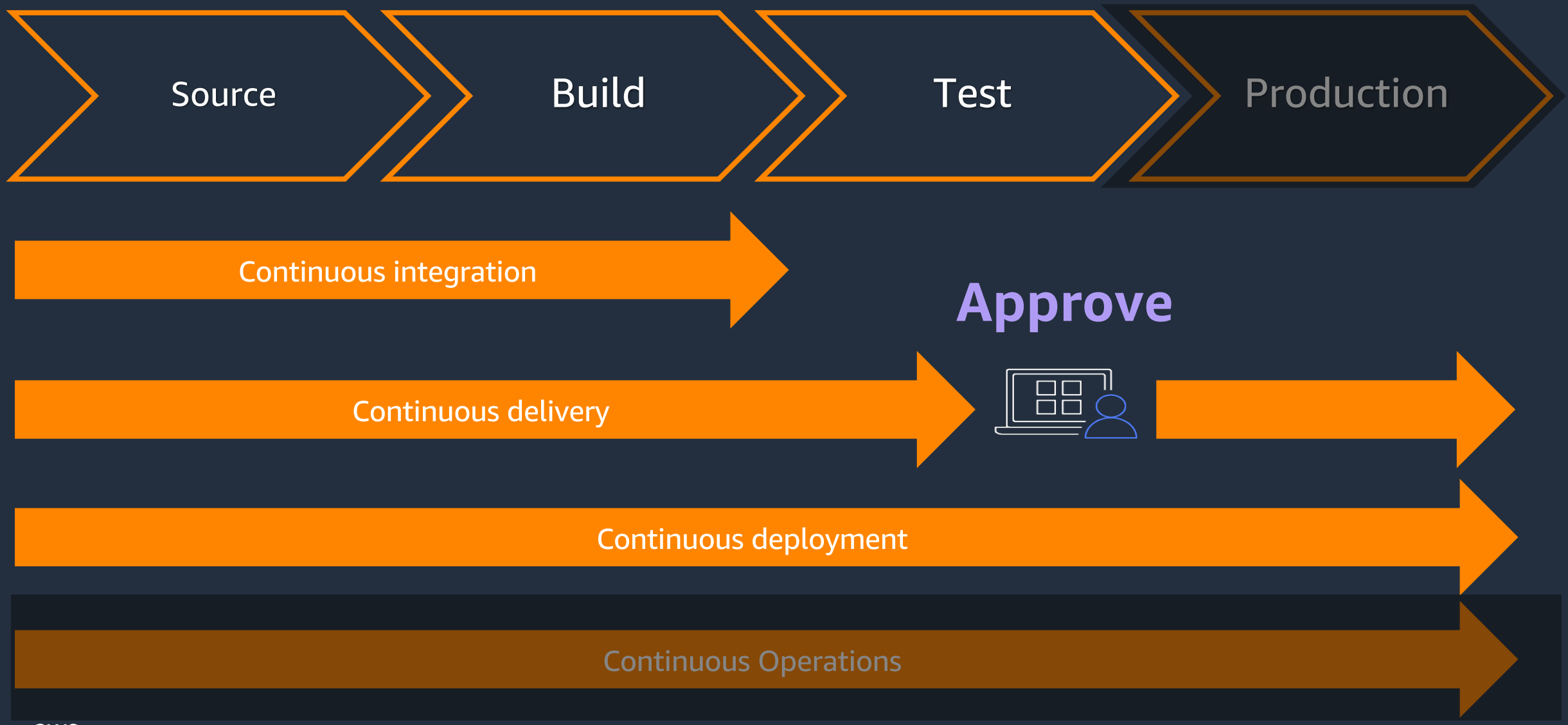
Continuous Delivery



Continuous Delivery



Release lifecycle



Continuous Deployment

1. Automatically deploy new changes to staging environments for testing
2. Deploy to production safely without impacting customers
3. Deliver to customers faster: Increase deployment frequency and reduce change lead time and change failure rate

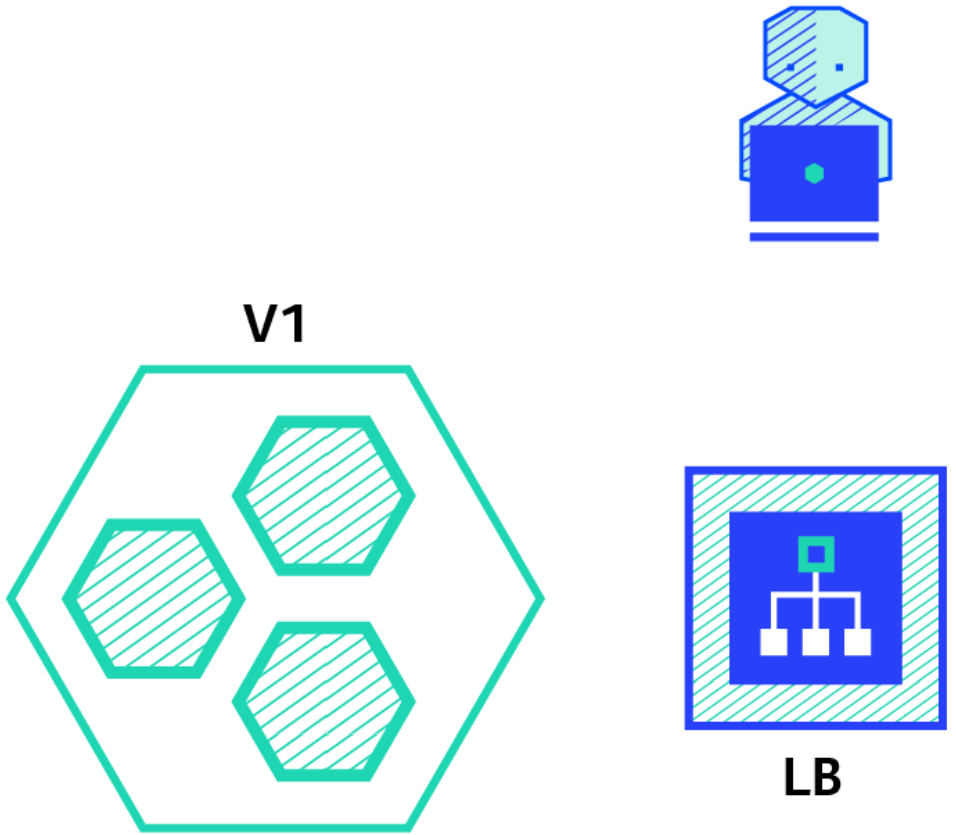
Deployment Strategies

Deployment Strategies

Strategy	Zero Downtime	Real traffic testing	Targeted users	Cloud cost	Rollback duration	Negative impact on user	Complecity of setup
RECREATE roll out new application code to an existing fleet of servers	✗	✗	✗	■□□	■■■	■■■	□□□
ROLLING version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■◪□	■■□
CANARY version B is released to a subset of users, then proceed to full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■

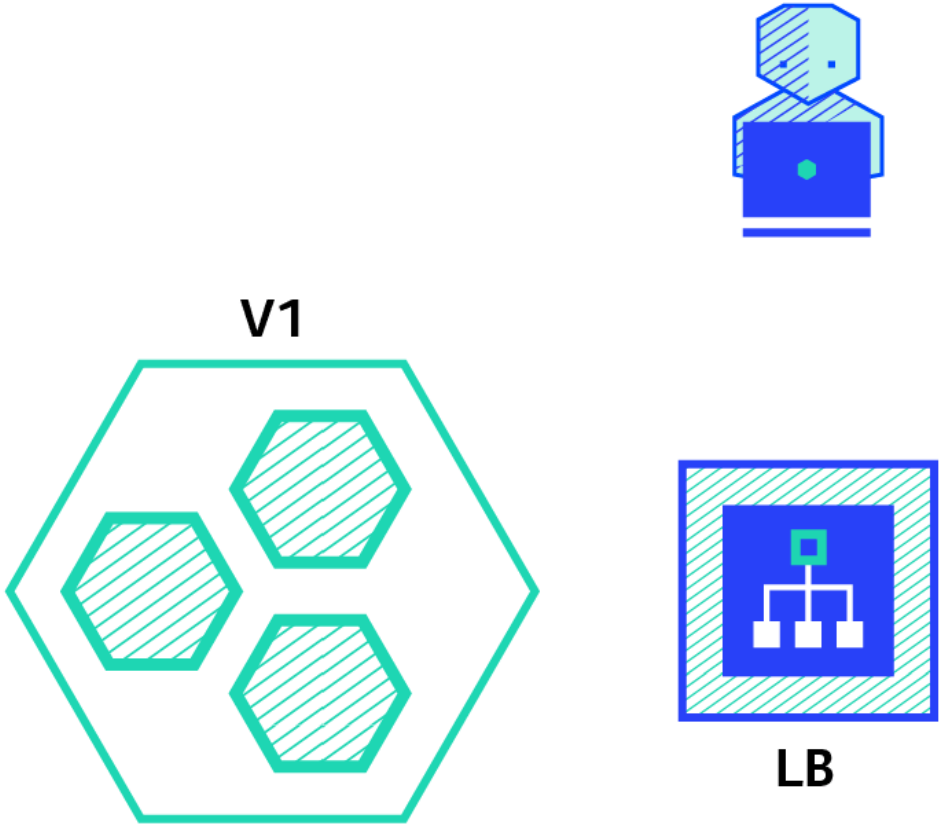
aws

Deployment Strategies: Recreate



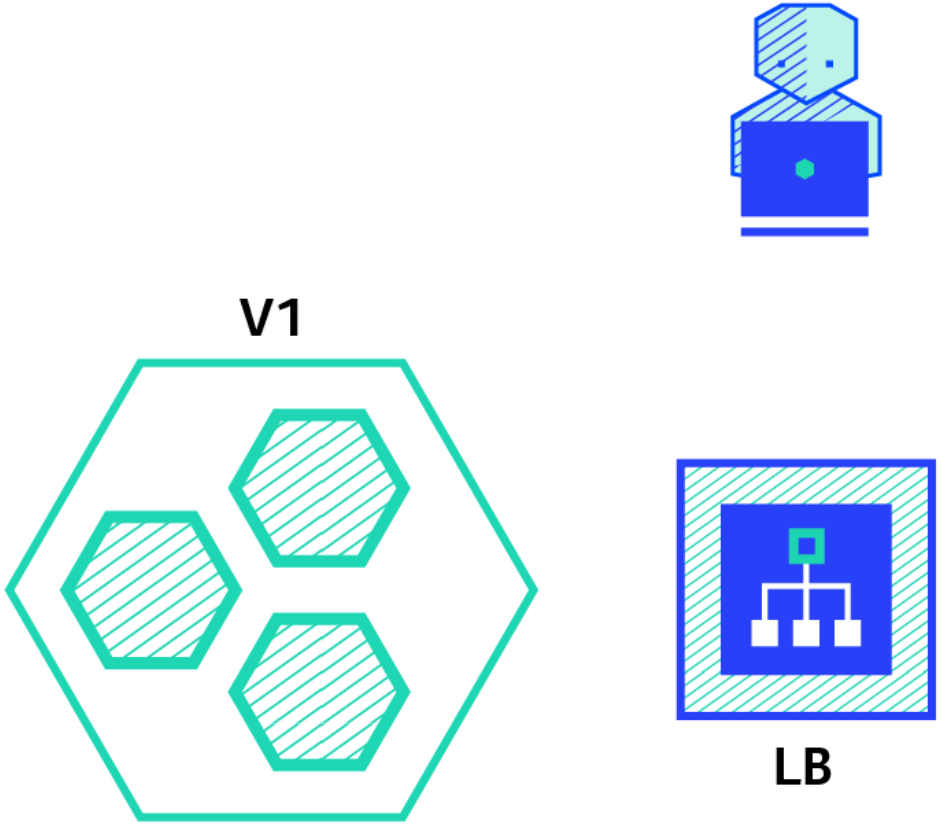
Six Strategies for Application Deployment

Deployment Strategies: Rolling



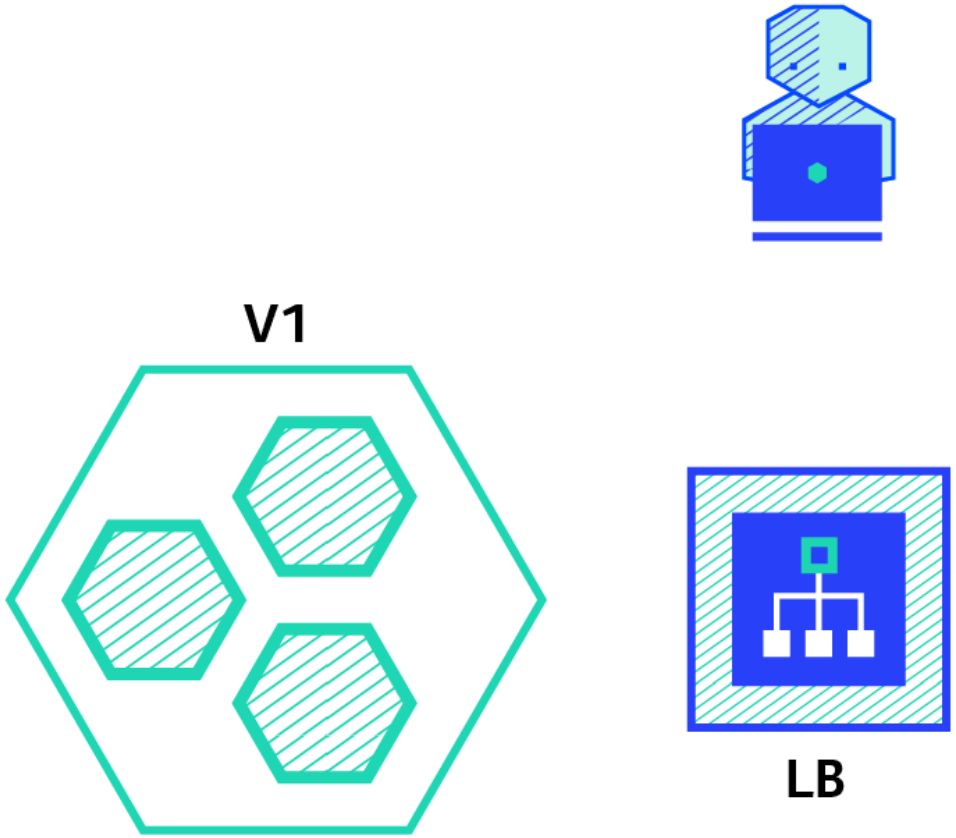
Six Strategies for Application Deployment

Deployment Strategies: Blue/Green



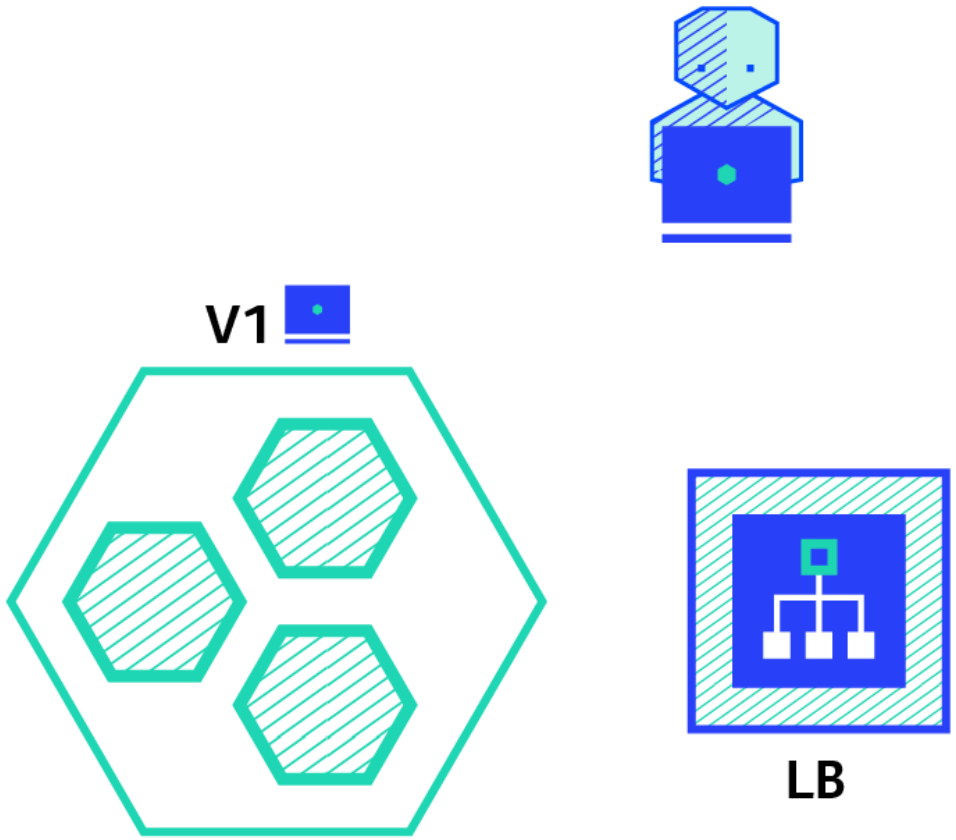
Six Strategies for Application Deployment

Deployment Strategies: Canary



Six Strategies for Application Deployment

Deployment Strategies: A/B testing



Six Strategies for Application Deployment

Deployment Strategies

Strategy	Zero Downtime	Real traffic testing	Targeted users	Cloud cost	Rollback duration	Negative impact on user	Complecity of setup
RECREATE roll out new application code to an existing fleet of servers	✗	✗	✗	■□□	■■■	■■■	□□□
ROLLING version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■◪□	■■□
CANARY version B is released to a subset of users, then proceed to full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■

aws

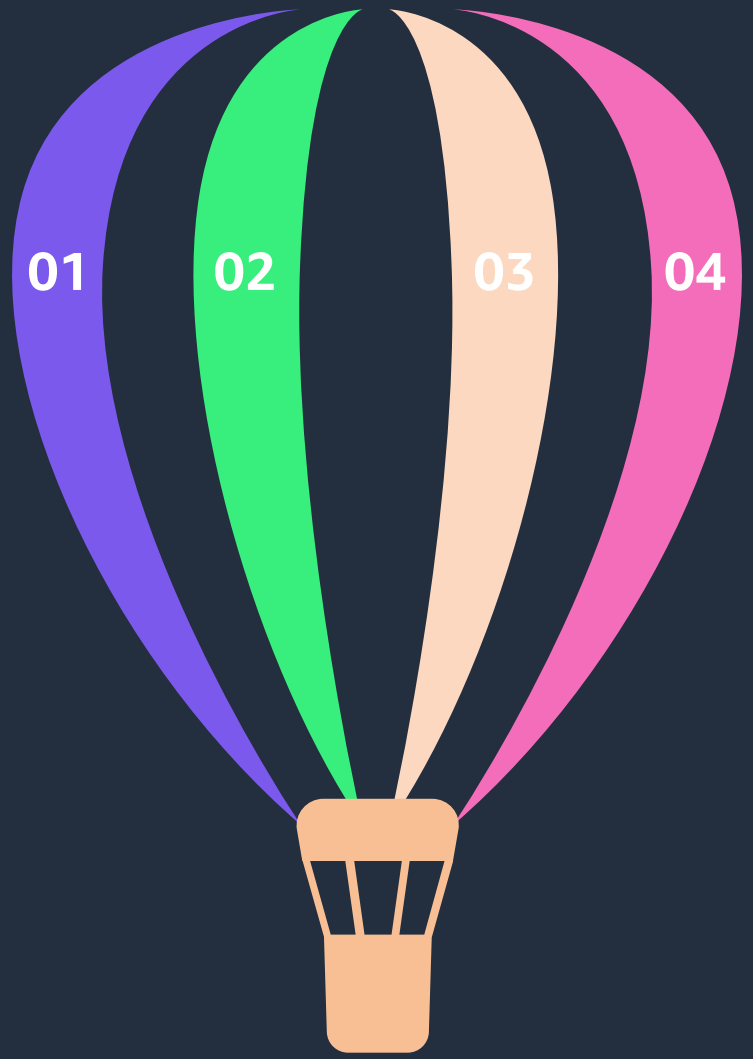
How to measure your CI/CD process

DORA metrics

DevOps Research and Assessment (DORA) Metrics

Deployment Frequency (DF)

How often an organization successfully releases to production



Change Failure Rate (CFR)

The percentage of deployments causing a failure in production

Lead Time for Changes (LT)

The amount of time it takes a commit to get into production

Time to Restore Service (TRS)

How long it takes an organization to recover from a failure in production

aws

Effects of CI/CD

Deployment frequency	Weekly–monthly	→	On-demand
Change lead time	One–six months	→	Less then one hour
Change failure rate	16–30%	→	0–15%

26% of software teams

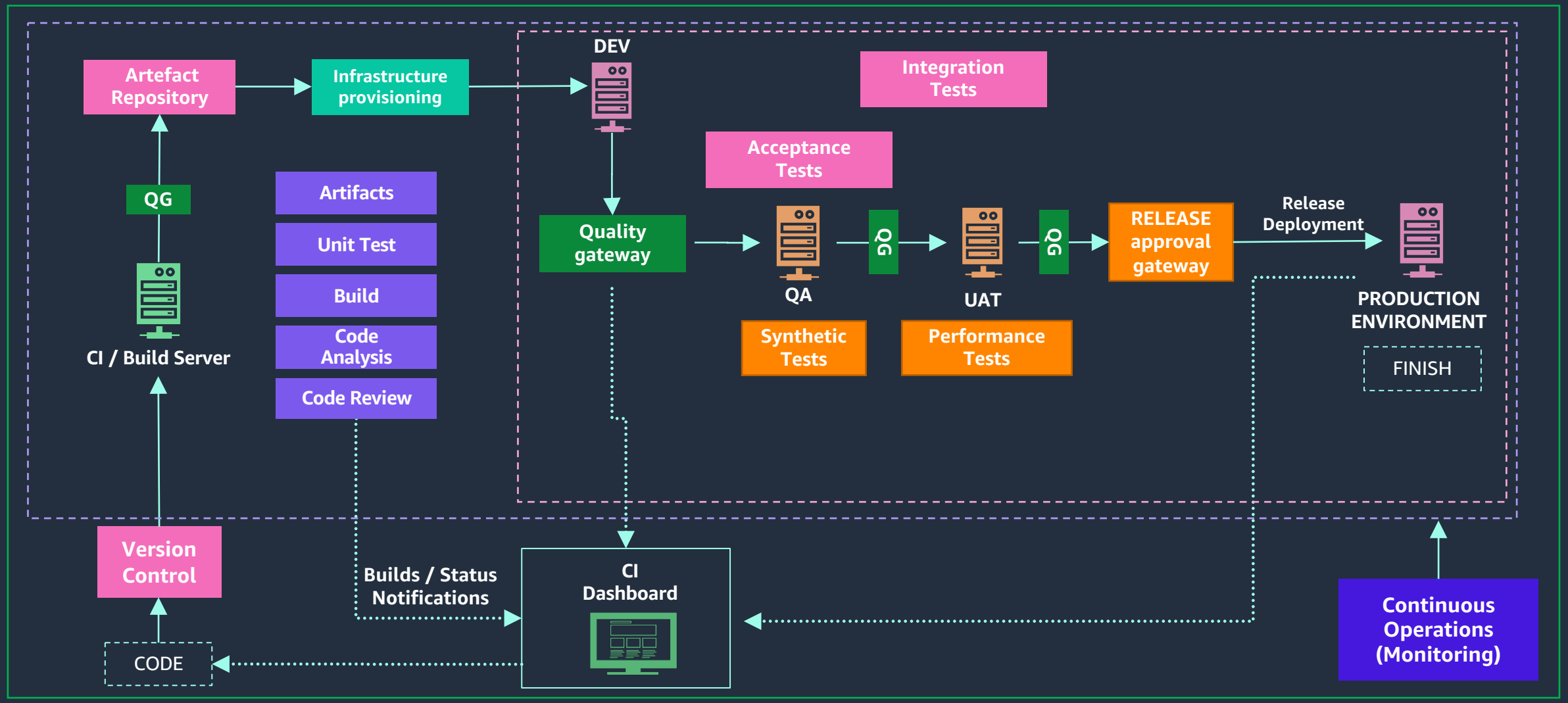
Source: 2021 DORA State of DevOps Report

Continuous Integration Metrics (KPIs)

Acronym	Metrics	Description	Process objectives under control	Recommended value
BFP	**B**uild **F**ailed **P**ercentage	The percent of builds failed due to infrastructure reason per day vs all builds	Infrastructure problems and progress in resolving them	5-10%
T2F	**T**ime**2F**eedback	Average duration of PR build.	Fluctuations of feedback time over period	<30 min
PUK	**P**ercentage of **U**nknown to **K**nown	The percent of unknown errors to known errors	New infra problems (e.g., git updates)	>90%
MBSQ	**M**aximum **B**uild**S**erver **Q**ueue	Maximal BuildServer queue on working hours	Quality/performance/insufficiency of build process	<5
AQT	**A**verage **Q**ueue **T**ime	Avg build wait time in Jenkins queue	Quality/performance/insufficiency of build process	<10m
PRC	**P**ull **R**equest **C**ount	Calculate count of pull (or merge) request per day	If all PR created were reviewed/built in measured period.	

CI/CD Blueprint

CI/CD blueprint



Thank you!

Viktor Vedmich