CI/CD with Microservices @ ebay

Yinon Avraham, Tech Lead @ ebay

Twitter: @yinonavraham
github.com/yinonavraham

Agenda

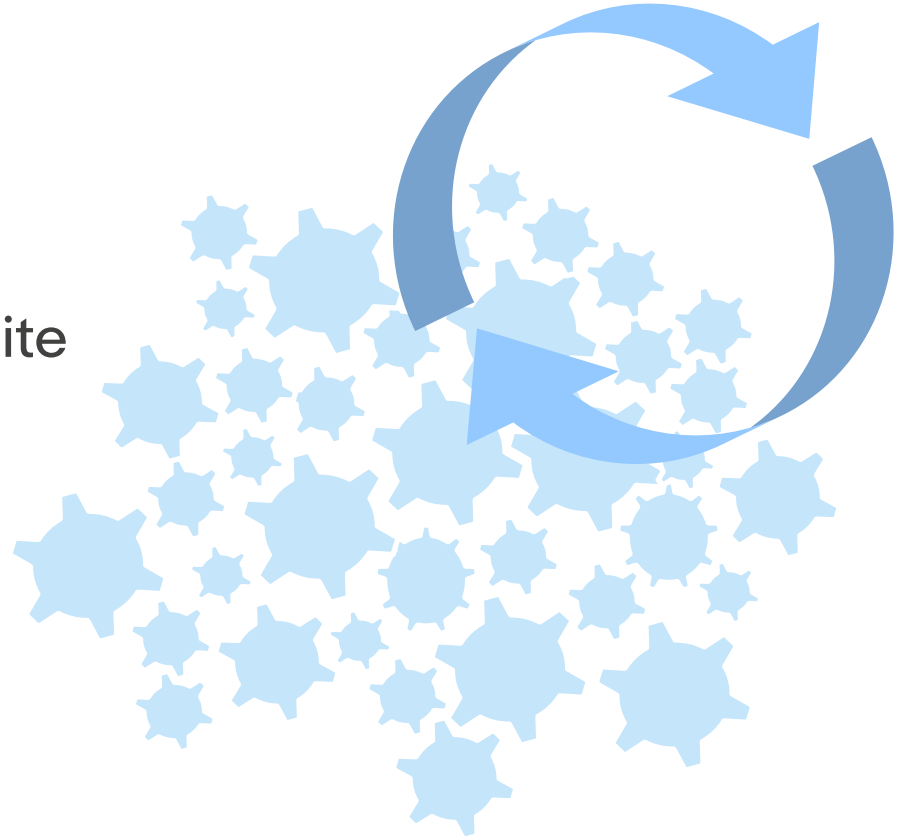
- Brief Overview on Application Development @ ebay
- CI/CD with Microservices

Overview

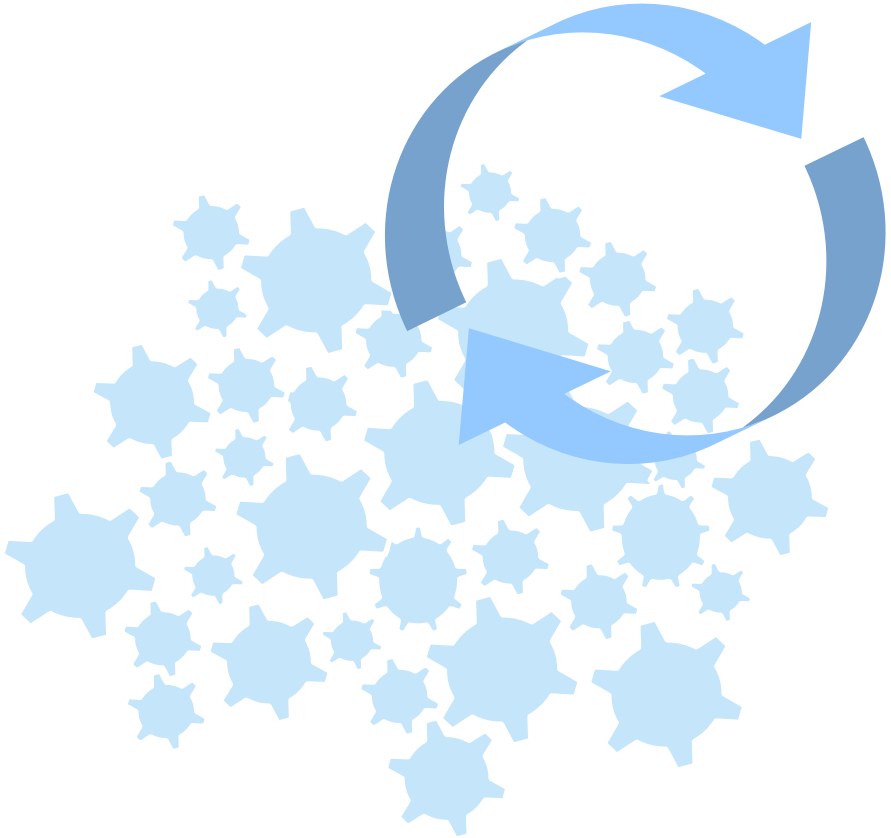
Application Development @ ebay

The ebay Ecosystem

- Thousands of applications
- Each application has many instances (1 .. thousands)
- Some are background processes, some are live-to-site
- Updates are deployed all the time



The ebay Ecosystem



- Developed and deployed from around the world

Google map, from: https://careers.ebayinc.com/join-our-team/start-your-search/find-jobs-by-location/

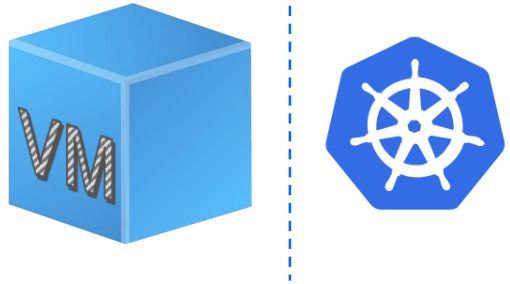
The ebay Infrastructure (examples)

- **Multi-region private cloud**

- **Application management and deployment**

  Two main options are supported:

  - In-house (VM based) application management and provisioning system
  - Kubernetes based deployment



- **CI as a Service**
- **Continuous Delivery pipeline management system**

- **System monitoring services**

ebay

Internally Hosted Tools (examples)

- **SCM** (GitHub Enterprise with HA and DR)
- **Code Analysis Tools** (e.g. SonarQube, IP checks, security checks, etc.)
- **Binary Repositories** (Maven, Docker, etc.)
- **CI as a Service** (Jenkins on Kubernetes)
- **DB as a Service** (Oracle, MySQL, Mongo, Cassandra, Couchbase, etc.)
- Etc.

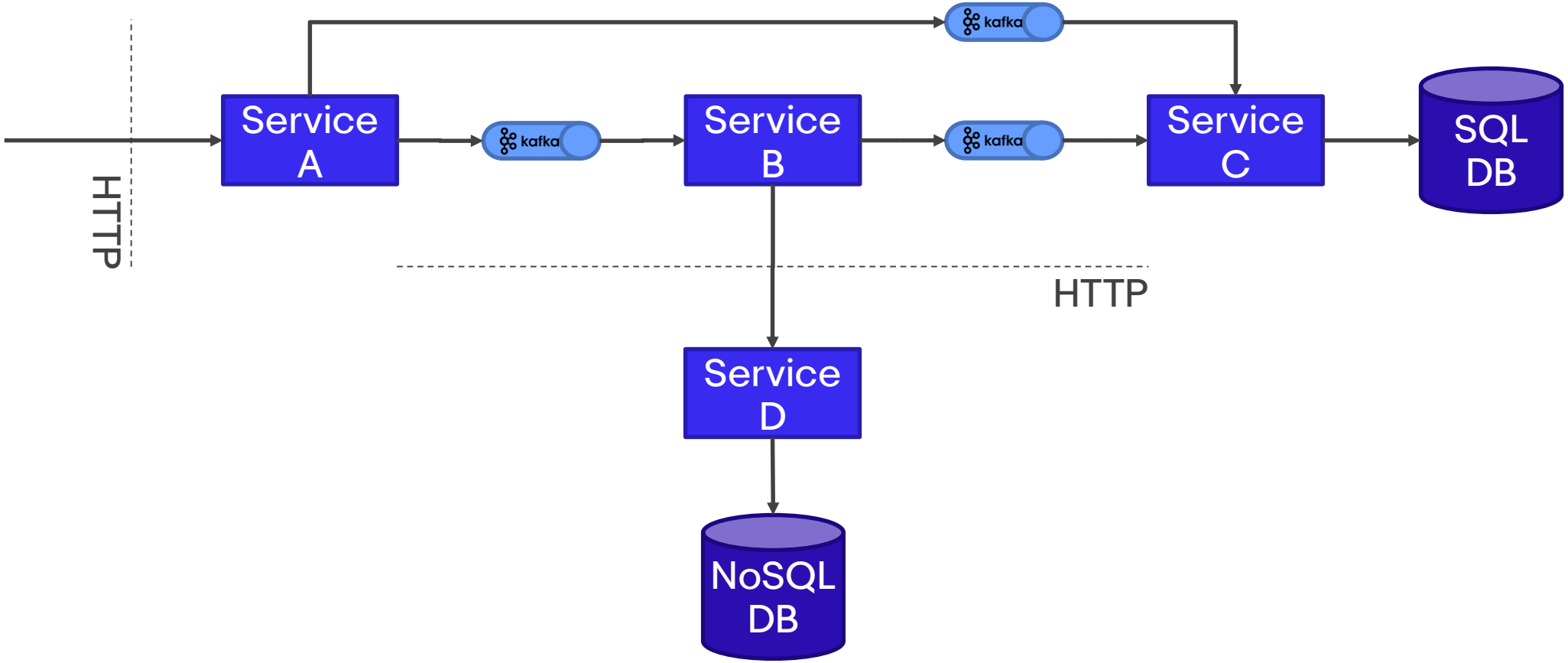
ebay

CI/CD with Microservices

Motivation for CI/CD

- Continuously update and improve code in production
- Development velocity
- Software metrics visibility
- Reduce probability of errors in production
- Reduce (manual) friction points
- Reduce development costs, e.g. by detecting errors early (by early integration)
- Etc.

Microservices



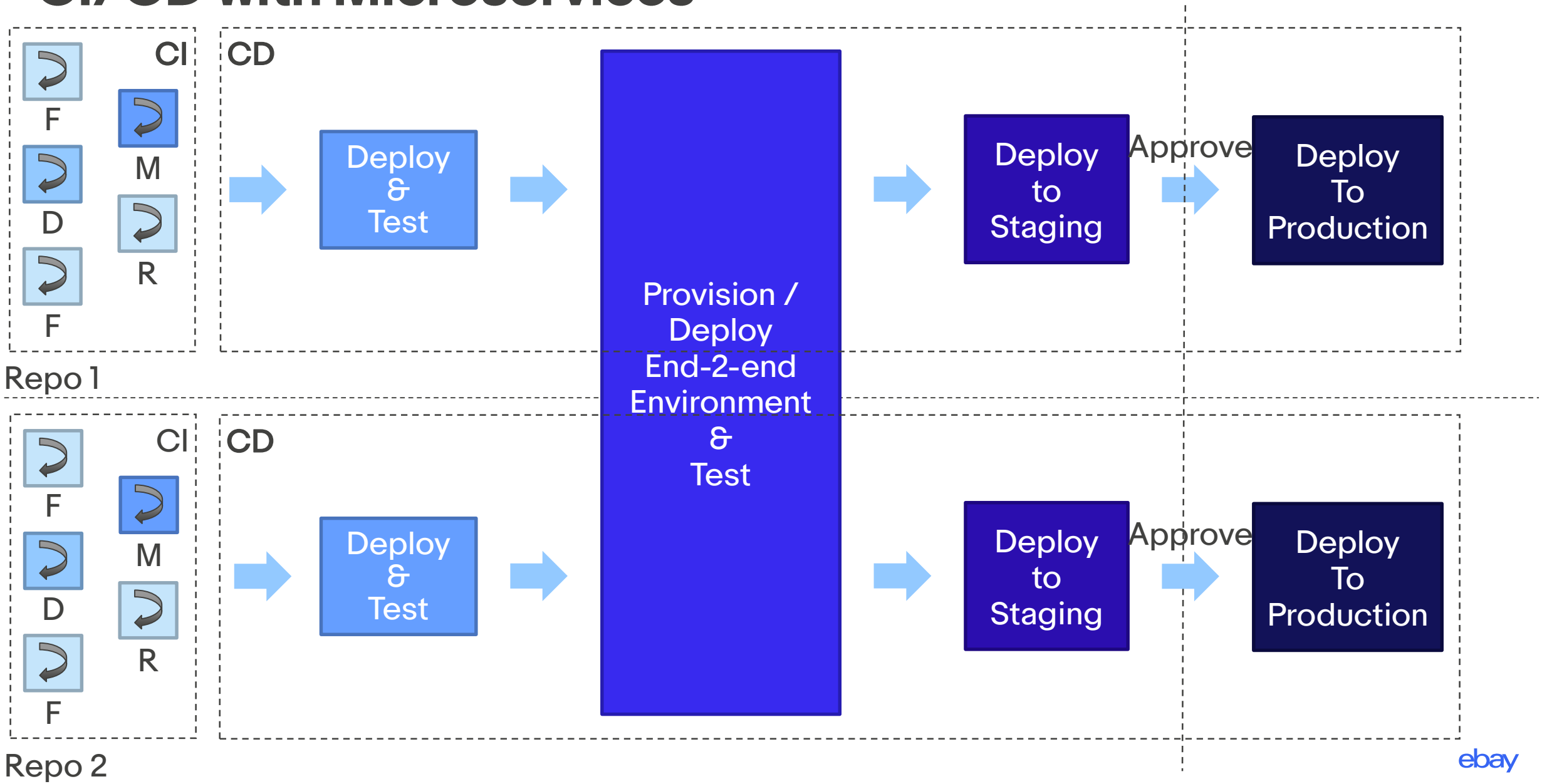
ebay

Some Basics

- **Multiple git repositories** (1 per application)
- **Similar code stack** (Java, Spring, etc.)
- **Shared common functionalities** (Kafka produce/consume, REST, etc.)
- **Branching models**
  - **GitHub Flow** – release (master) branch / feature branches
  - **Git Flow** - release (master) branch / integration (develop) branch / feature branches

Bottom line - multiple concurrent live branches

CI/CD with Microservices



ebay

DEV

Local Development

- **Requirements**
  - Developer should be able to run the services on his development machine
  - There should be isolation between development environments – no shared resources
  - Developers shall use the same dependencies (DB version, etc.)
- **How**
  - Running dependencies locally using Docker (databases, services, etc.)
  - Use docker-compose files with preconfigured settings (e.g. DB credentials, etc.)
  - Custom CLI tool to ease developers' daily work (automate common tasks)
  - Development environment related files are managed in a dedicated git repository

CI

Continuous Integration Pipeline

- **Requirements**
  - Quick feedback
  - Protected branches
  - Standard Pipeline: Build → Test → Publish
  - D.R.Y
  - Isolation
  - Visibility

- **Pre-requisite:** define boundaries

Continuous Integration Pipeline

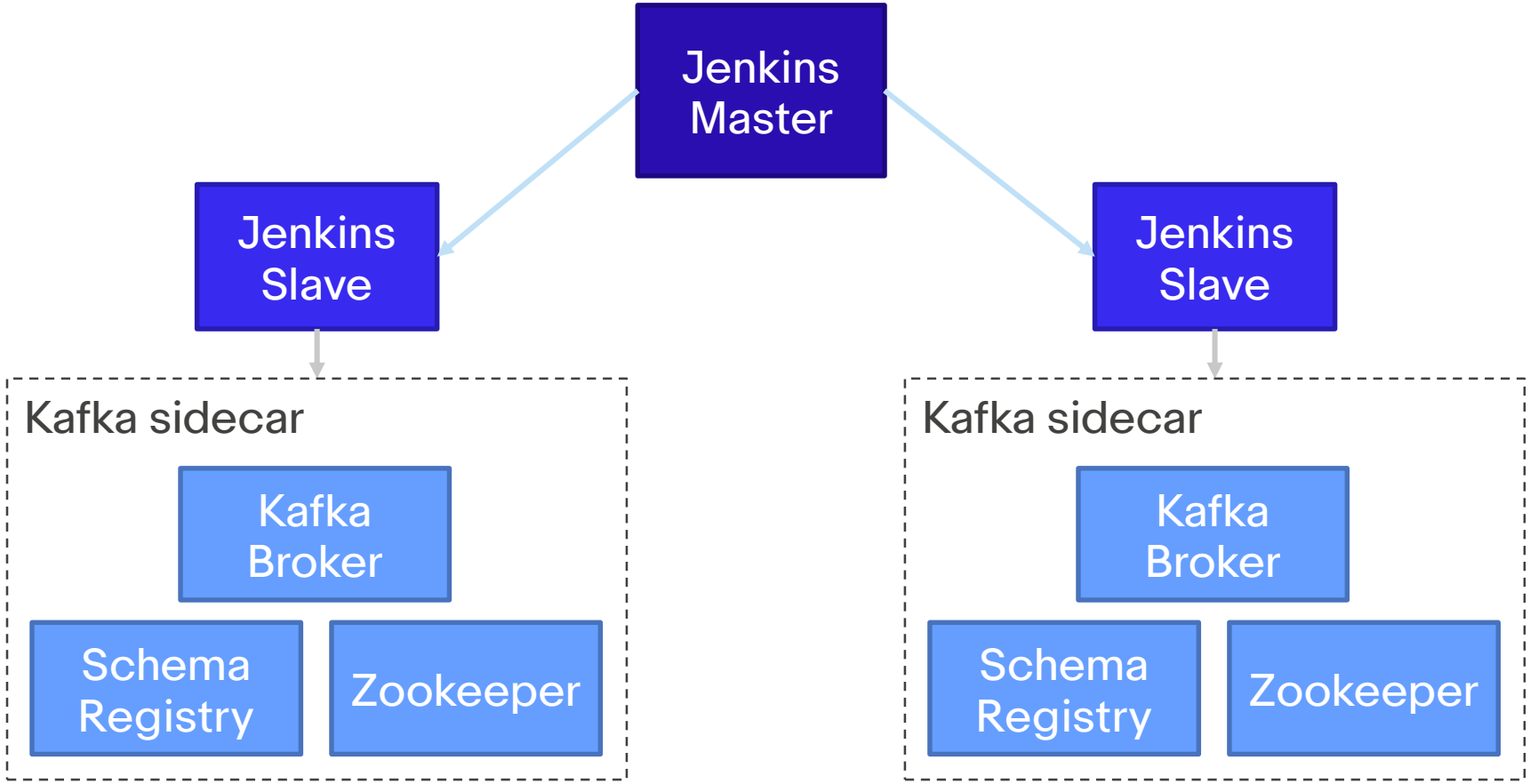
Requirements:

- Quick Feedback
- Protected Branches
- Build → Test → Publish
- D.R.Y
- Isolation
- Visibility

- **How**
  - Detailed notifications over email and Slack (where / who / what / ...)
  - Verbose output during the build
  - Pull request validation + GitHub integration (trigger validation job, update the pull request, etc.)
  - Use Jenkins Multibranch Pipeline jobs
    - Pipeline as code (job configuration should be part of the codebase)
    - Auto detects branches (new and deleted)
    - Triggered on push (using GitHub webhook)
    - Use Jenkins Shared Pipeline Libraries for code reuse (https://github.com/ebay/Jenkins-Pipeline-Utils »)
    - Spin-up dependencies using Docker (Jenkins is running on Kubernetes...)

Continuous Integration Pipeline

- Isolation using the “side-car” pattern (using Docker Compose)



ebay

Continuous Integration Pipeline

Tips

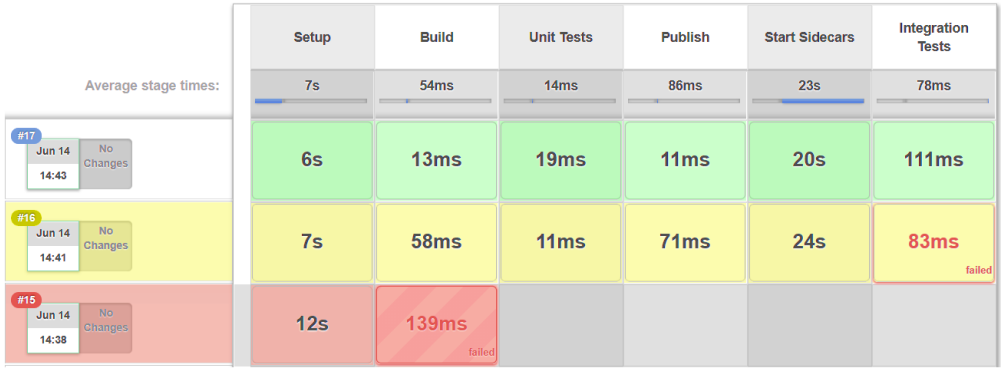
- **Pipeline code reuse**
  - Put commonly used utilities in a shared library
  - Version your pipeline library as any other library (major version- no breaking changes, minor version- features and fixes…)
  - When using your own shared libraries - depend on the latest minor version of a constant major version (e.g. “1.x”)

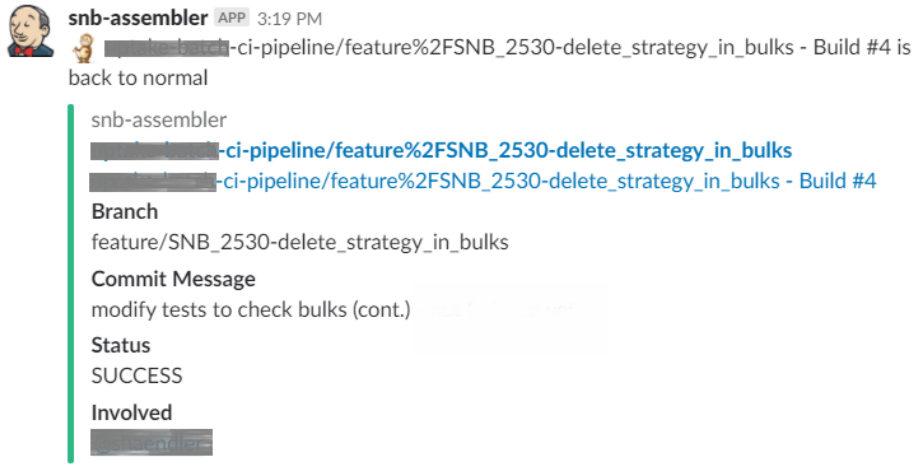
ebay

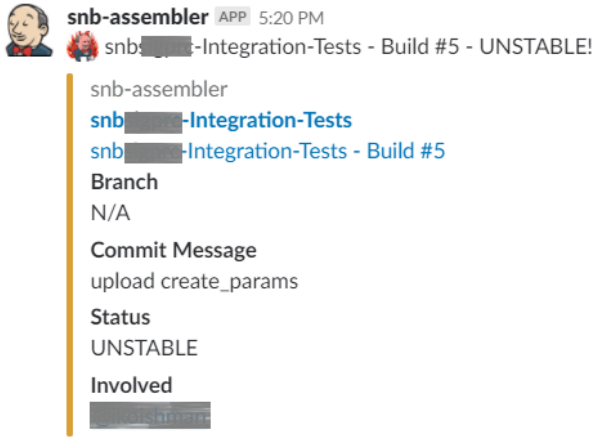
Continuous Integration Pipeline

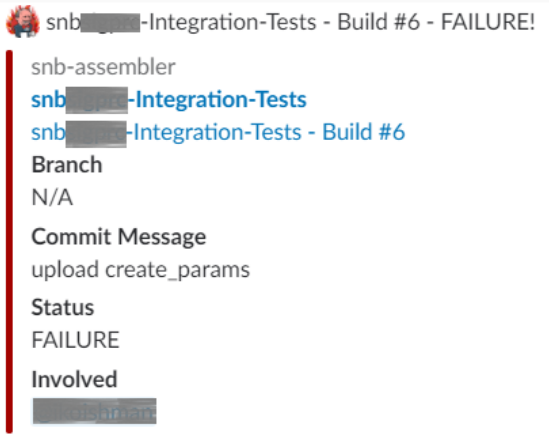
Tips

- **Visibility**
  - Use visualization where possible
  - Write everything to the build log (tools, inputs, decisions, etc.)
  - Make sure to keep sensitive data obfuscated (auth keys, etc.)
  - Emails and Slack notifications
    - Increase visibility by using colors, icons, formatting…
    - Add details which can help understand the issue (but not too much)









CD

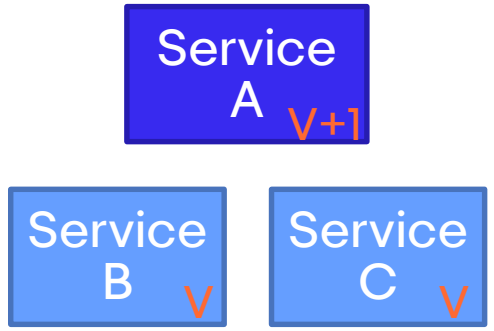
Continuous Delivery Pipeline

- **Requirements**
  - Automate the road to production (as much as possible)
  - Protect production from potential issues
  - Verify service
  - Verify cross-service integration
  - High confidence
  - Auditing

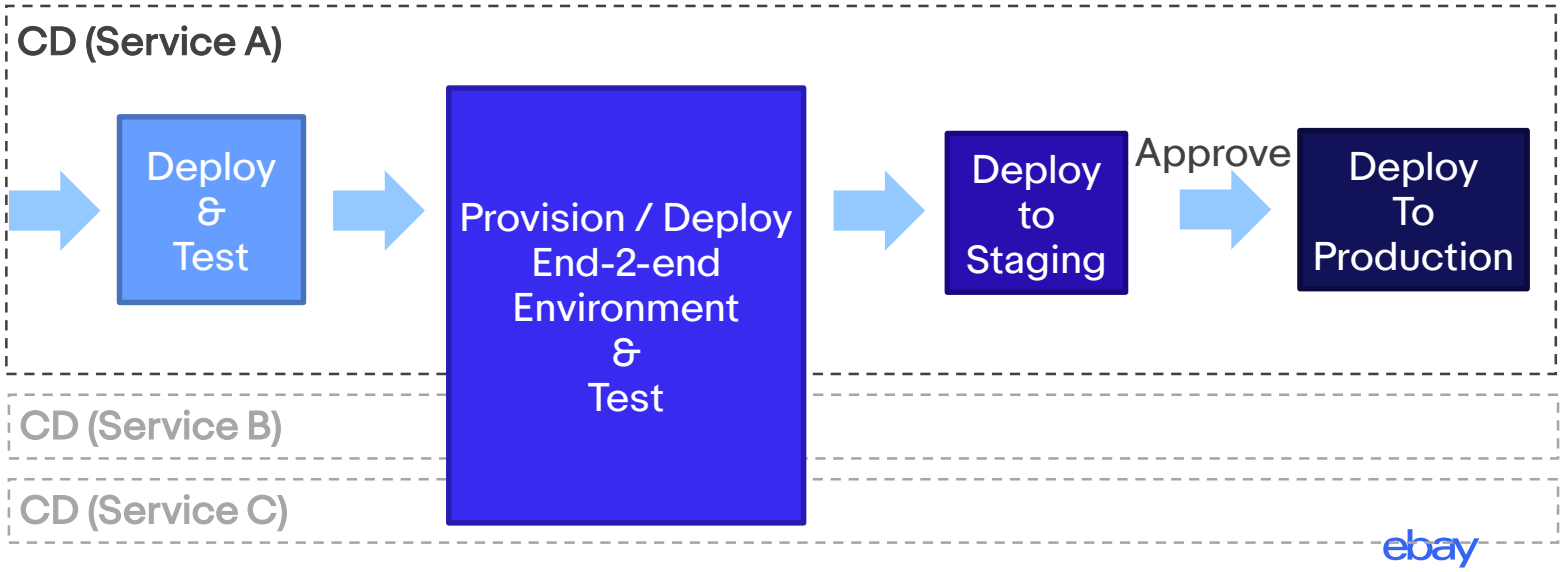
- **Remember -** boundaries...

Continuous Delivery Pipeline

- **Automatic up to Staging**
  - Service validation (Deploy and test on a “real” isolated test environment)
  - Service integration (Deploy and test on an isolated end-to-end environment)
  - Deploy to Staging & run smoke test



- **Production**
  - Manual approval and signing (Audited)
  - Deploy to production
  - Monitoring, Logging, Alerting, ...



Summary

Key Takeaways

- **Automate as much as possible**
- **Keep isolation** (developers, environments, build runs, etc.)
- **Define test boundaries**
- **Local Development Environment**
  - Should be easy and quick to setup, maintain and run
  - Should be aligned across all developers
- **CI**
  - Should provide quick feedback on all branches
  - Should help protecting important branches
- **CD**
  - Should give high confidence
  - Should be auditable

ebay

Questions?

Thank You!

Yinon Avraham, Tech Lead @ ebay

Twitter: @yinonavraham
github.com/yinonavraham