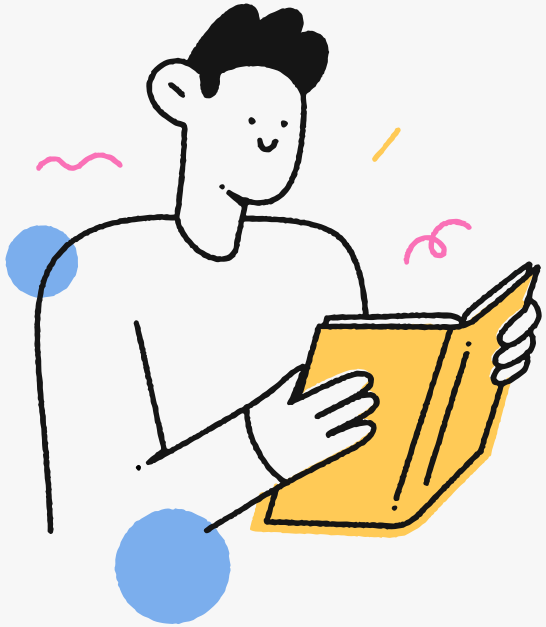
Progressive Delivery

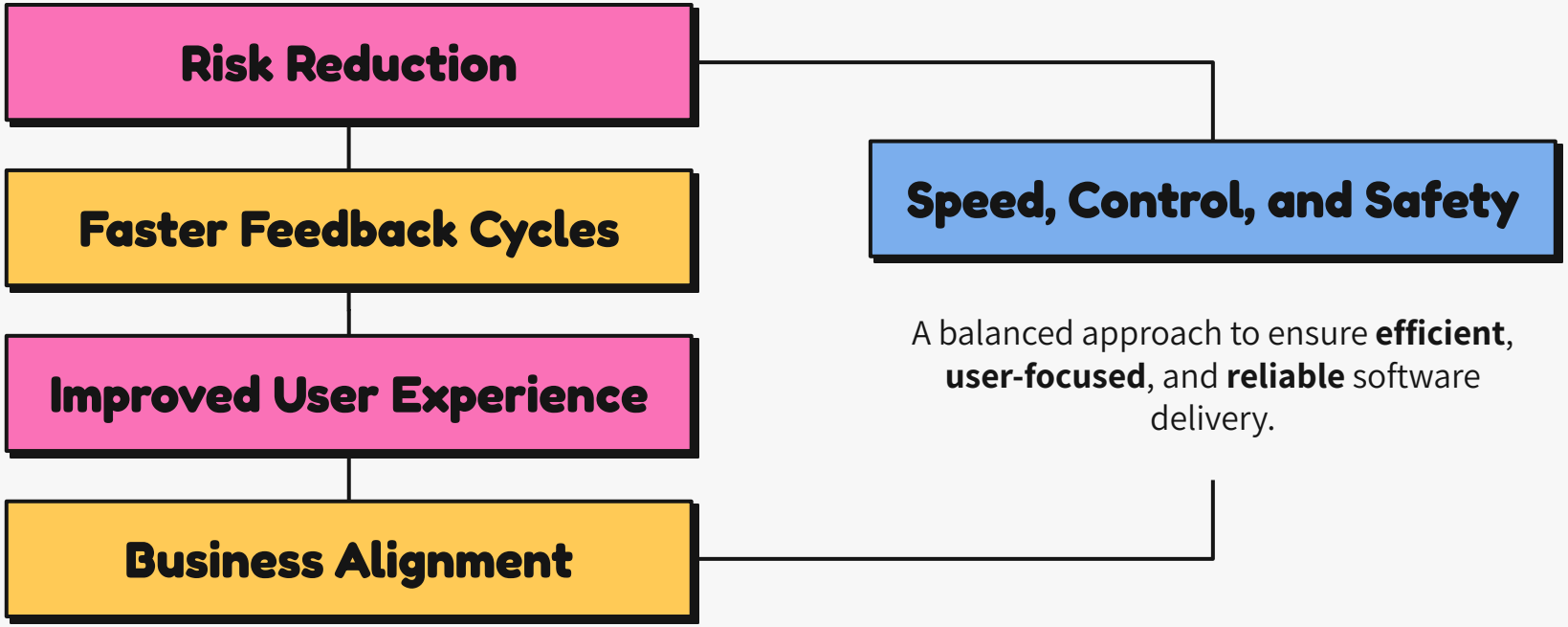
Ben Khalfallah Héla



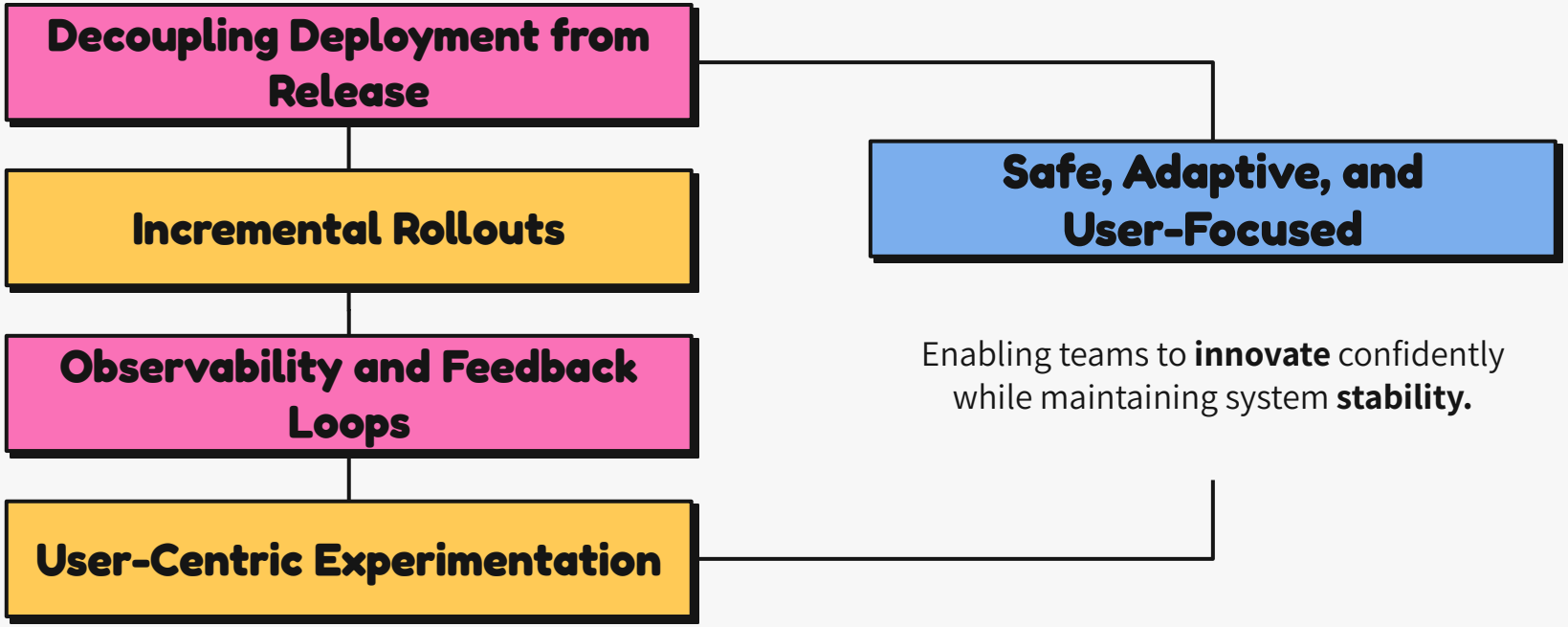
“Progressive delivery is the **next step after continuous delivery**, where new versions are deployed to a **subset of users** and are **evaluated** in terms of correctness and performance **before rolling them to the totality of the users** and **rolled back** if not matching some key **metrics**.”

— Carlos Sanchez, CloudBees

Aims and Benefits of Progressive Delivery



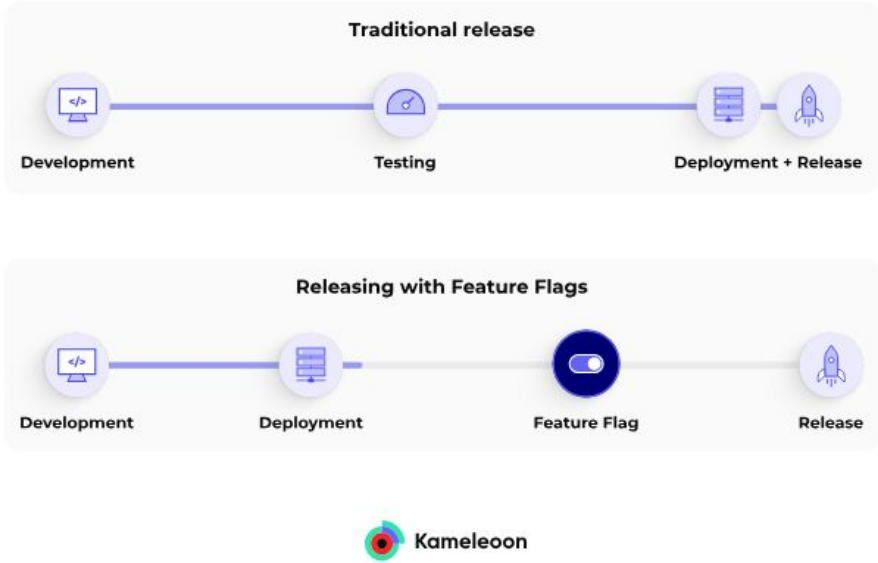
Core Principles of Progressive Delivery



Building Blocks of Progressive Delivery

1. **Feature Flags** allow teams to **enable, disable, or hide specific features** in the user interface **without** redeploying code.
2. **Canary Deployment** involves creating an updated version of an application and **directing a small percentage of user traffic** to it.
3. **Blue-Green Deployments**, **two separate environments**, blue and green, are maintained to ensure seamless deployment.
4. **A/B Testing,** an experimentation technique where **two or more variations** of a feature are shown to different user groups to compare their impact on behavior and performance.
5. **Observability** stands on three pillars – **Logs**, **Traces**, and **Metrics**.

Feature Flags



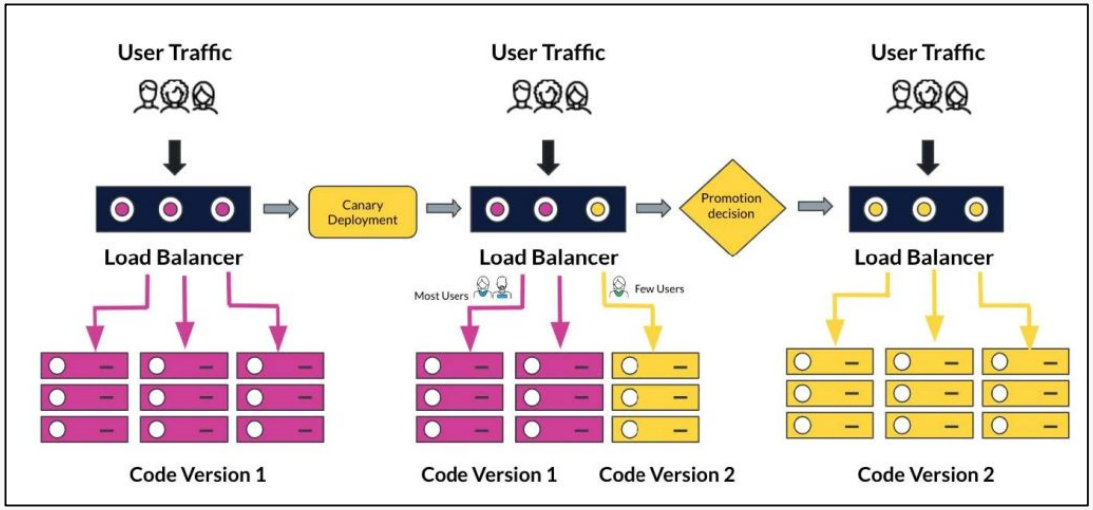
As a key element of Progressive Delivery, feature flags **enable testing changes** and **new features with a select group of users**, ensuring controlled rollouts before broader availability.

```
let treatment =
flags.getTreatment("search-algorithm");

if (treatment === "v1") {
    // Use the first version of the new search
algorithm
    useSearchAlgorithmV1();
} else if (treatment === "v2") {
    // Use the second version of the new search
algorithm
    useSearchAlgorithmV2();
} else {
    // Use the existing search algorithm
    useCurrentSearchAlgorithm();
}
```

https://www.kameleoon.com/feature-flagging

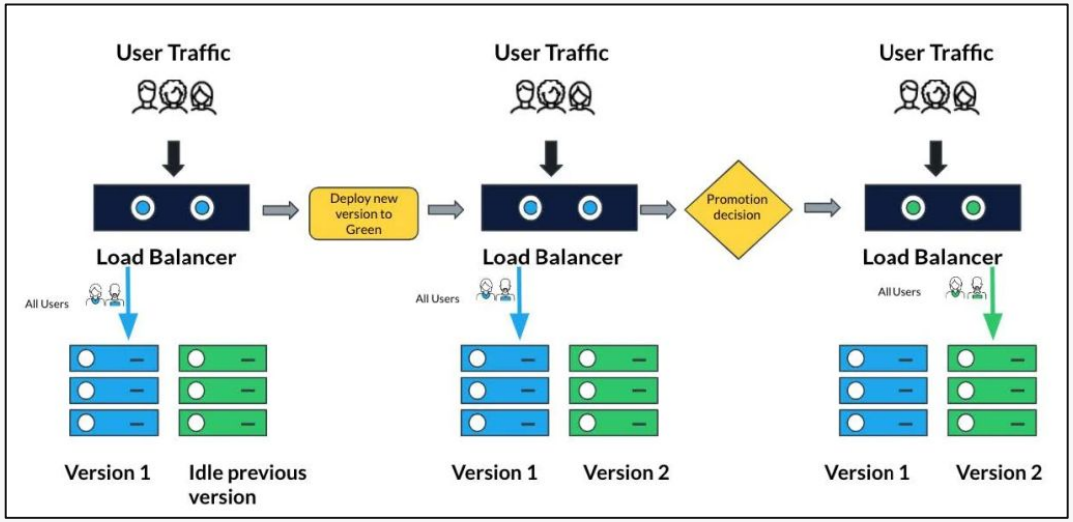
Canary Deployment



https://www.cloudbees.com/blog/progressive-delivery-a-detailed-overview#canary-testing

- This approach allows teams to **test the updated code in a live production environment with a subset of users**.
- If the **changes perform as expected**, the updated version is **gradually rolled out to the entire user base**.
- Else, **if issues arise**, the **impact is limited to the small group**, reducing the “blast radius” of potential failures.
- To ensure accurate results, it’s essential that the **canary group is representative of the overall user base**.

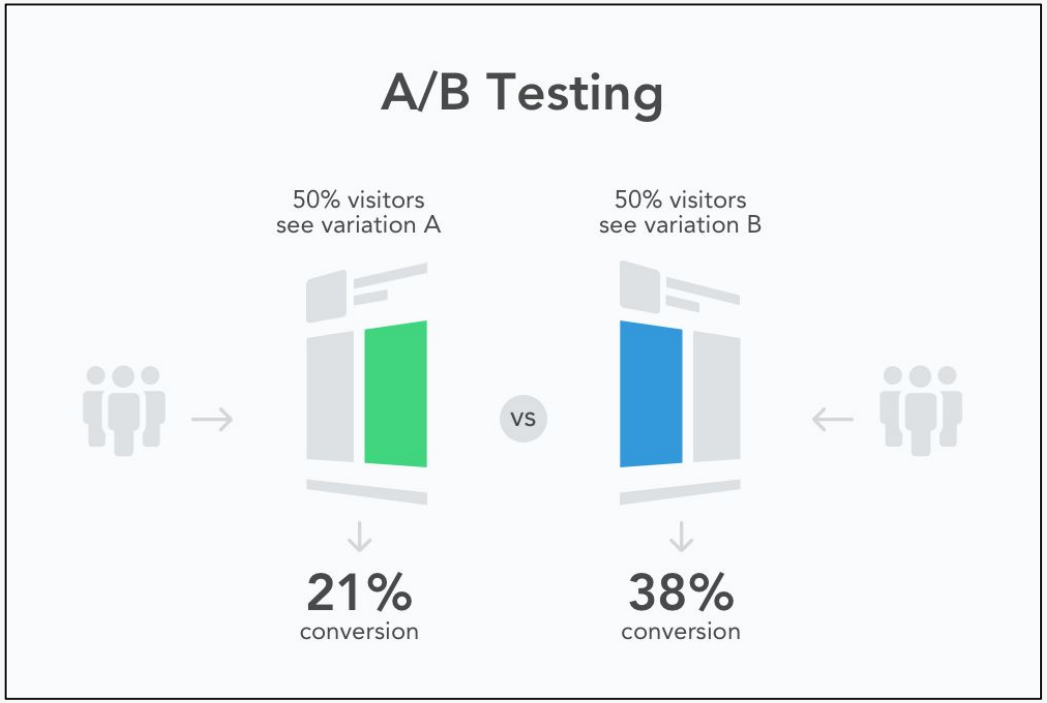
Blue-Green Deployments



https://www.cloudbees.com/blog/progressive-delivery-a-detailed-overview#blue-green-deployments

- **One environment (e.g., blue) is live** and serving user traffic, while **the other (e.g., green) is used for testing and staging** the next version of the software.
- When it's time to release a new version, **traffic is gradually or instantly redirected** from the live (blue) environment to the updated (green) environment.
- This approach minimizes downtime and provides a quick rollback option in case of issues.

A/B Testing



- A/B Testing is an experimentation technique **where two or more variations of a feature are shown to different user groups** to compare their **impact on behavior and performance**.
- The goal is to determine **which variation achieves the best results based on predefined metrics**.

https://splitmetrics.com/resources/what-is-ab-testing-and-why-it-matters-for-mobile-developers/

Tools and Integration

Feature Flagging	• **LaunchDarkly**: Manage feature visibility, enable gradual rollouts, toggle features dynamically. • **Flagsmith**: Open-source solution for feature flagging and experimentation.
Advanced Deployment Strategies	• **Argo Rollouts:** Manage **canary**, **blue-green** deployments, and traffic shaping in Kubernetes. • **Istio:** Traffic management for canary deployments and splitting in service meshes.
Observability & Visualization	• **Prometheus:** Collect and query metrics from deployments and applications. • **OpenTelemetry:** Unified tracing, logging, and metrics collection. • **Grafana:** Create dashboards to monitor metrics and application performance.
A/B Testing	**Kameleoon:** Advanced experimentation platform for feature performance optimization.
Analytics	**Amplitude**: Measure user engagement and experiment impact.

Progressive Delivery: Key Case Studies

Industry Insights	
Deployment Rings at Microsoft	• Employs **deployment rings** to manage incremental rollouts. • Features are **first tested with a small, internal group (Ring 0)** before gradually being expanded to broader user bases. • Ring deployments **reduce risk and ensure high-quality rollouts** while maintaining agility.
Staff Ships and Feature Flags at GitHub	• GitHub uses **“staff ships”** as **an internal testing phase**, **deploying new features to their own employees first**. • Leveraging feature flags, GitHub **measures the feature’s performance and gathers feedback before external release**, ensuring stability and customer satisfaction.

Progressive Delivery: Key Case Studies

Industry Insights	
Atlassian: Continuous Delivery Transformation	• Atlassian adopted **LaunchDarkly** to **decouple deployments from releases**. • Teams leverage feature flags to **test features with specific user segments and iterate based on real-time feedback**. • **Impact**: Improved Mean Time to Recovery (MTTR) by **97%** and increased customer satisfaction scores.
XLNT Platform at LinkedIn	• **LinkedIn's XLNT** platform is a **robust A/B testing and experimentation solution**. • Fully **integrated with its Continuous Deployment pipeline**, XLNT supports **more than 200 experiments daily**, offering actionable insights into feature performance and user engagement.

Anti-Patterns of Progressive Delivery

"Big Bang" Releases

Deploying large-scale changes without staged rollouts increases risk.

Ignoring Feedback

Deployments without mechanisms for real-time user feedback lead to missed learning opportunities.

Anti-Patterns of Progressive Delivery

Overlooking Blast Radius

Deploying changes to all users at once amplifies the impact of failures.

Overusing Feature Flags

Accumulation of outdated or unused flags creates technical debt and confusion.

Future of Progressive Delivery

1 AI/ML-Driven Deployments

- **Trend:** AI/ML tools **predict** rollout strategies and detect risks.
- **Innovation:** **Automated** anomaly detection and proactive mitigation.
- **Example Tools:** Harness, Dynatrace.

2 Edge Computing and IoT

- **Trend:** Decentralized decision-making for localized updates.
- **Innovation:** Lightweight agents and adaptive traffic shaping for resource optimization.
- **Example Tools:** Azure IoT Hub, AWS IoT Core.

3 Serverless and Cloud-Native

- **Trend:** Enhanced GitOps workflows for real-time deployment synchronization.
- **Innovation:** Smarter rollouts, self-healing deployments, and advanced observability.
- **Example Tools:** Flux, ArgoCD, Istio.

Thanks!

Do you have any questions?

- ★ **Blog:** https://helabenkhalfallah.com/blog/
- ★ **X:** https://x.com/b_k_hela
- ★ **Linkedin:** https://www.linkedin.com/in/h%C3%A9la-ben-khalfallah-4a104014/