SHIP WITH CONFIDENCE

FEATURE FLAGS

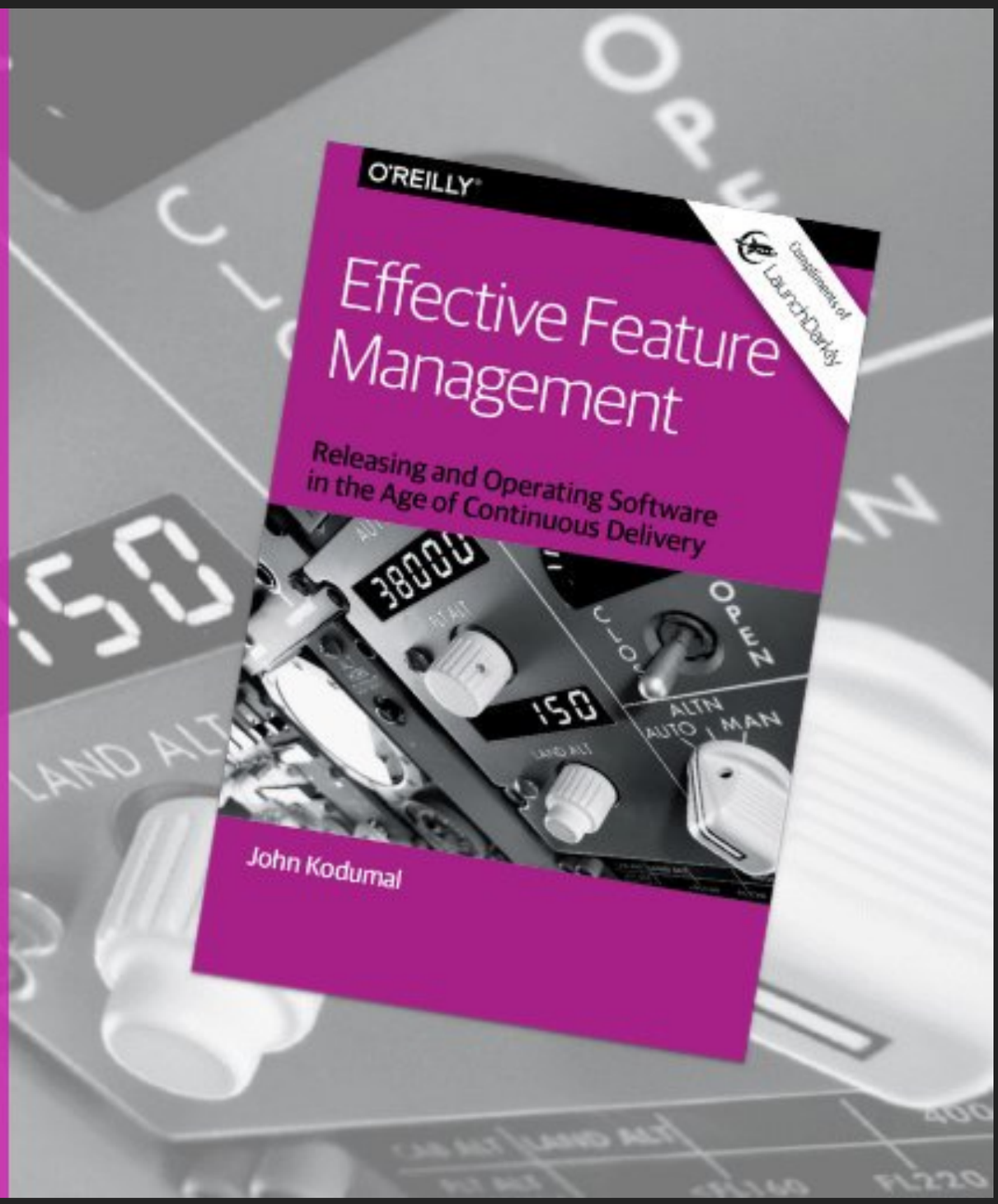
WHO AM I?

- Brian Abston
  - Born and raised in OKC
  - Married for 24 years
  - 3 sons (16, 13, 8)
- Platform Architect at Sonic Drive In

FREE EBOOK

Effective Feature Management

Releasing and Operating Software in the Age of Continuous Delivery



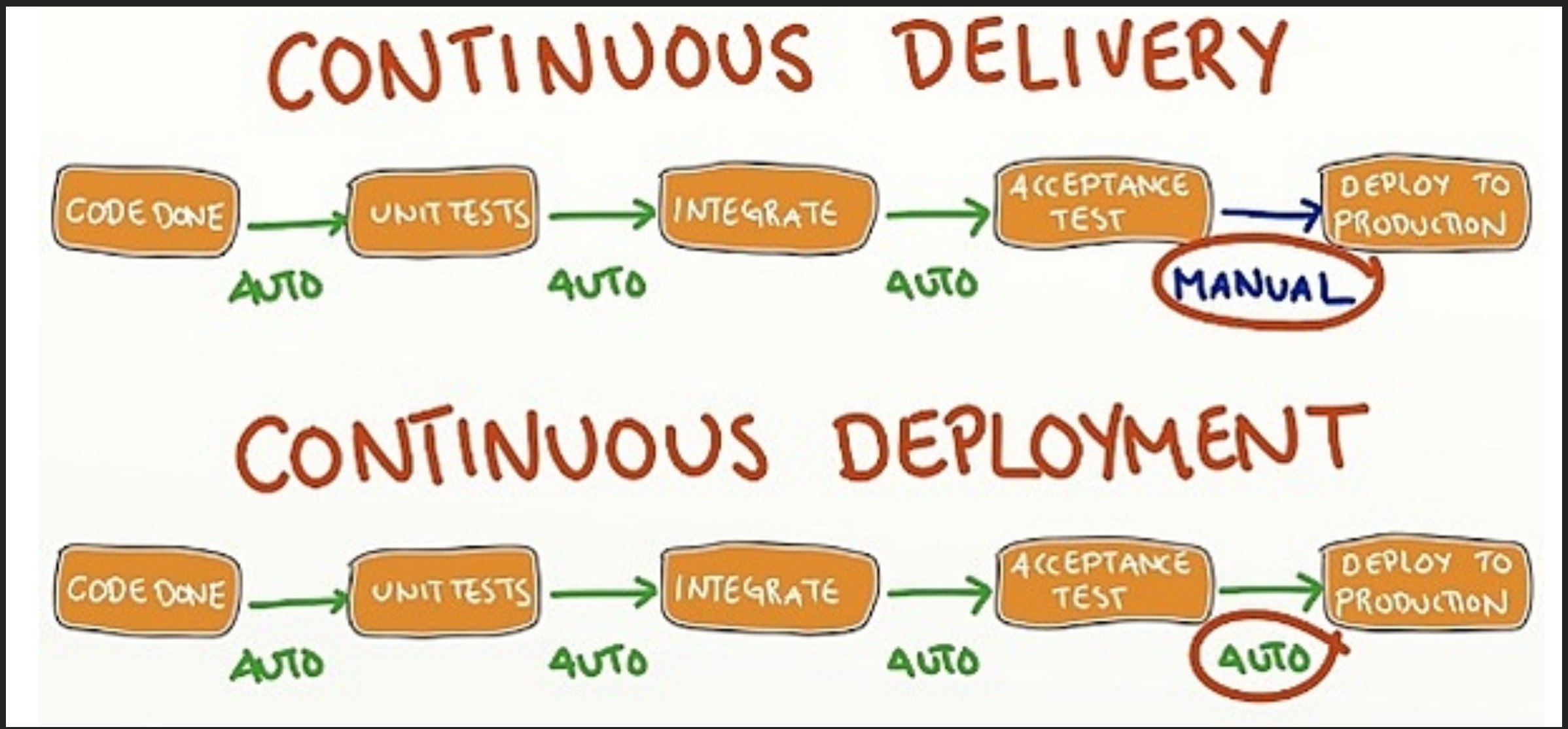
- **Continuous Integration** is the practice of integrating changes from different developers in the team into a trunk as early as possible, in best cases several times a day. This makes sure the code individual developers work on doesn't divert too much. When you combine the process with automated testing, continuous integration can enable your code to be dependable.
- Who is doing Continuous Integration?

- **Continuous Delivery** is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by **clicking** on a button.
- Who is doing Continuous Delivery?

- **Continuous Deployment** goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's **no human intervention**, and only a failed test will prevent a new change to be deployed to production.

- Who is doing Continuous Deployment?

While continuous deployment may be our goal it can be very scary for the business. This is the reason a lot of companies stop at continuous delivery.



WHAT IF

- Who has to get CAB/business approval before deploying to production?
- What if you could fully decouple the act of delivering software from the act of releasing features?
- What if you could deploy any time you want but release only when you are ready?
- Feature flags enable businesses to dynamically control the availability of the features to the end users.

WHAT IS A FEATURE FLAG

- Just a decision point in your code that can change the behavior of your application.
- Think of a feature flag as a powerful “if” statement.

EARLY DAYS

- Feature flags have been around:
  - Command-line argument
  - Undocumented variable in configuration file
  - Hidden value in registry
- The value of each feature flag was typically set early in the life cycle:
  - Compile
  - Deploy
  - Runtime
- To pick up a change the application would have to be recompiled, redeployed, or restarted.
- Modern feature flagging is different and done in real time.

TYPES

- Feature flags can be either temporary or permanent.
  - Temporary are used to safely deploy changes or test new behaviors.
  - Permanent flags can be used to create kill switches or reveal functionality to specific users.

FLAG OFTEN

- Does not have to be changes only visible to customers.
- Can flag backend improvements or infrastructure changes.

DEPLOY != RELEASE

- When deployment is decoupled from release, feature flags give you the opportunity to create a release strategy.
  - How will the feature be released?
  - Should anyone see it now?
  - Who will be getting this feature first?
  - Who will beta test it?
  - Will it be rolled out progressively?
  - Do I need to compare it against the old behavior?
  - Do I need to hit a particular date for an external event?
  - What do I do if something doesn't go right?
- Teams using feature flags are no longer constrained by users seeing new features the moment it hits production.


I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO I DO IT IN PRODUCTION
memegenerator.net


I SEE YOU TEST YOUR CODE IN PRODUCTION
I TOO LIKE TO LIVE DANGEROUSLY
quickmeme.com

TEST IN PRODUCTION
WHAT COULD GO WRONG
imgflip.com

TEST IN PRODUCTION

- True test environments are difficult to create.
- Micro-services have created new challenges.
  - The volume of messages, transactions, traffic, and data has increased significantly, to the point that it is often impossible to replicate in another environment.
  - Long gone are the days of realistically testing on your laptop.
- Real users are unpredictable.
- Your test data, even at high volume, will never capture every edge case.
- The closer you come to production-level traffic volumes, the more expensive testing becomes, especially when in the cloud.
- **Kill your test environments!!! Save money. Test in production.**

HOW TO TEST IN PRODUCTION

- With feature flags you can use powerful deployment strategies to control access to new features and instantly turn access off to any feature without the need to rollback or redeploy.

DEPLOYMENT STRATEGIES

- **Percentage Deployment**
  - Rollout to small number of users and increase over time if things are healthy and feedback is positive.
  - Useful if little variation in your targeted user base or are more concerned with operational impact of your change.
  - Can be automated to gradually increase if there are no monitoring alarms.
- **Ring Deployment**
  - Term coined by Microsoft.
  - Similar to percentage rollout except the groups are selected specifically to manage the risk.
  - Start with low-risk users and expand through larger, higher-risk users.
  - Typically starts with internal users, then canary group (Facebook), then beta or early adopters, and then general release.

BASELINES

- Establishing KPI baselines and monitoring those is critical to early detection of issues, resolving them, and lowering MTTR.

WORKFLOW

- Feature flags immediately add technical debt.
  - Make sure you when you create a feature flag you also create a card to go back and cleanup the flag once the feature has been completely rolled out and tested.
  - At the time of development create a branch and pull request removing the flag. Then merge when the flag is no longer needed.
- You can start with any branching strategy and work towards trunk based development.
- Don’t put flags in reusable library code. Making your library dependent on your flagging strategy could reduce reusability.
- Use naming convention like fflag.
- Name feature flag creation and deletion in the commit message.

GETTING STARTED

- Config file
- Library
  - Ff4j
  - Unleash
- Feature management service
  - Launch Darkley
  - Split