DevOps

Week 10

Murtaza Munawar Fazal

Feature Toggles

- Feature Flags allow you to change how our system works without making significant changes to the code. Only a small configuration change is required.
- Feature Flags offer a solution to the need to push new code into the trunk and deploy it, but it isn't functional yet.
- Suppose you are modifying the interest calculations. You can change the code so that other users who don't have the Feature Flag set will use the original interest calculation code. The members of your team who are working on the new interest calculations can set to see the Feature Flag will have the new interest calculation code.
- The other type of Feature Flag is a Release Flag. Now, imagine that after you complete the work on the interest calculation code, you're nervous about publishing a new code out to all users at once.
- You have a group of users who are better at dealing with new code and issues if they arise, and these people are often called Canaries.

Feature Toggles

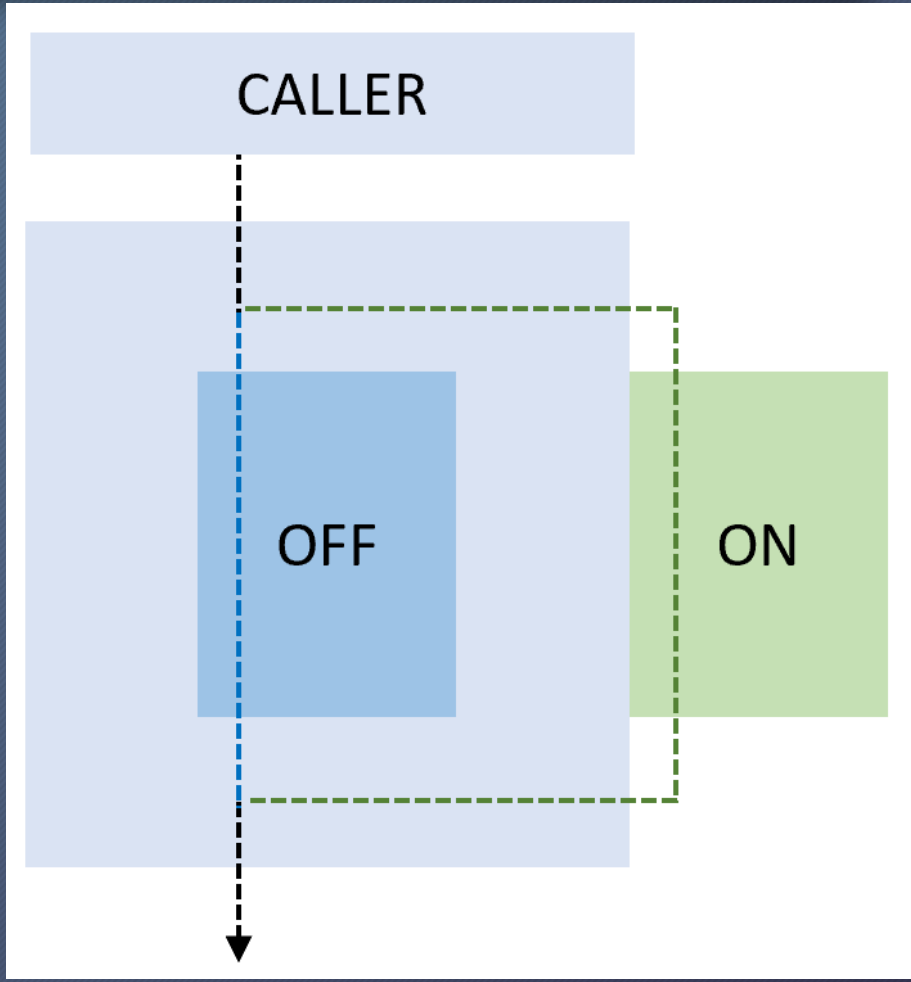
- You change the configuration so that the Canary users also have the Feature Flag set, and they'll start to test the new code as well. If problems occur, you can quickly disable the flag for them again.
- You could have half the users working with the original version of the code and the other half working with the new code version.
- You can then directly compare the outcome and decide if the feature is worth keeping. Feature Flags are sometimes called Feature Toggles instead.
- By exposing new features by just "flipping a switch" at runtime, we can deploy new software without exposing any new or changed functionality to the end-user.
- The question is, what strategy do you want to use in releasing a feature to an end-user.
  - Reveal the feature to a segment of users, so you can see how the new feature is received and used.
  - Reveal the feature to a randomly selected percentage of users.
  - Reveal the feature to all users at the same time.

Feature Toggles

- The idea of separating feature deployment from Feature exposure is compelling and something we want to incorporate in our Continuous Delivery practice.
- It helps us with more stable releases and better ways to roll back when we run into issues when we have a new feature that produces problems.
- By separating deployments from revealing a feature, you make the opportunity to release a day anytime since the new software won't affect the system that already works.
- Feature toggles are a great alternative to branching as well. Branching is what we do in our version control system. To keep features isolated, we maintain a separate branch. When we want the software to be in production, we merge it with the release branch and deploy it. With feature toggles, you build new features behind a toggle. Your feature is "off" when a release occurs and shouldn't be exposed to or impact the production software.

Feature Toggles

- When the switch is off, it executes the code in the IF, otherwise the ELSE.
- You can make it much more intelligent, controlling the feature toggles from a dashboard or building capabilities for roles, users, and so on.
- If you want to implement feature toggles, many different frameworks are available commercially as Open Source.



Canary releases

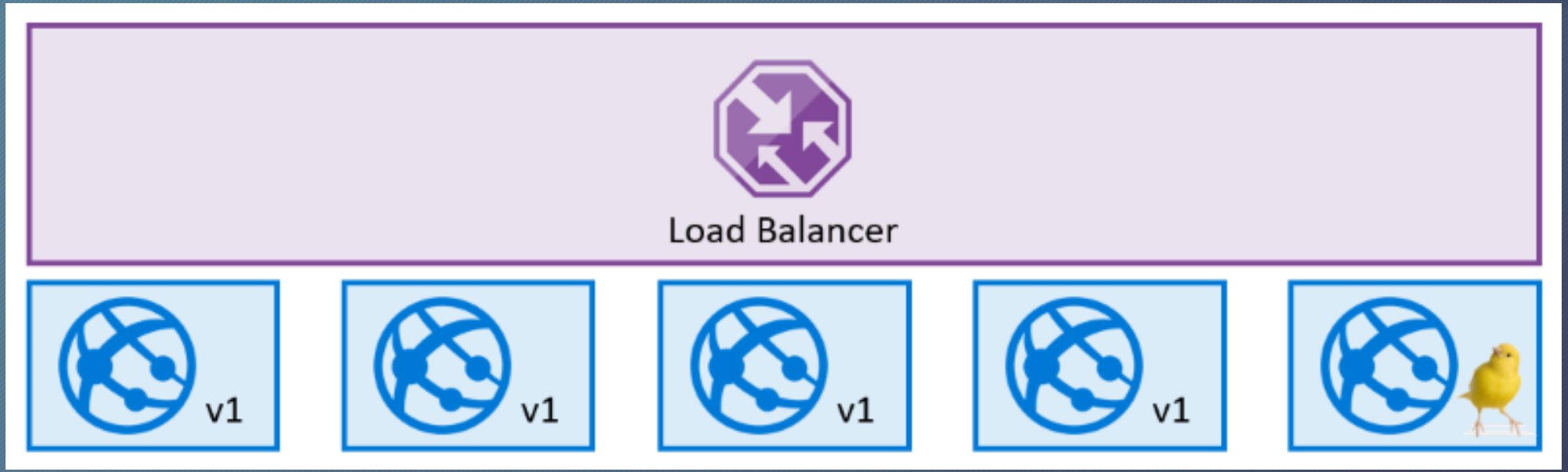
- The purpose of the canary was to identify the existence of toxic gasses. The canary would die much sooner than the miner, giving them enough time to escape the potentially lethal environment.

- A canary release is a way to identify potential problems without exposing all your end users to the issue at once. The idea is that you tell a new feature only to a minimal subset of users. By closely monitoring what happens when you enable the feature, you can get relevant information from this set of users and either continue or rollback (disable the feature).

- If the canary release shows potential performance or scalability problems, you can build a fix for that and apply that in the canary environment.

- After the canary release has proven to be stable, you can move the canary release to the actual production environment.

Canary releases



- Canary releases can be implemented using a combination of feature toggles, traffic routing, and deployment slots.
  - You can route a percentage of traffic to a deployment slot with the new feature enabled.
  - You can target a specific user segment by using feature toggles.

Controlling your canary release

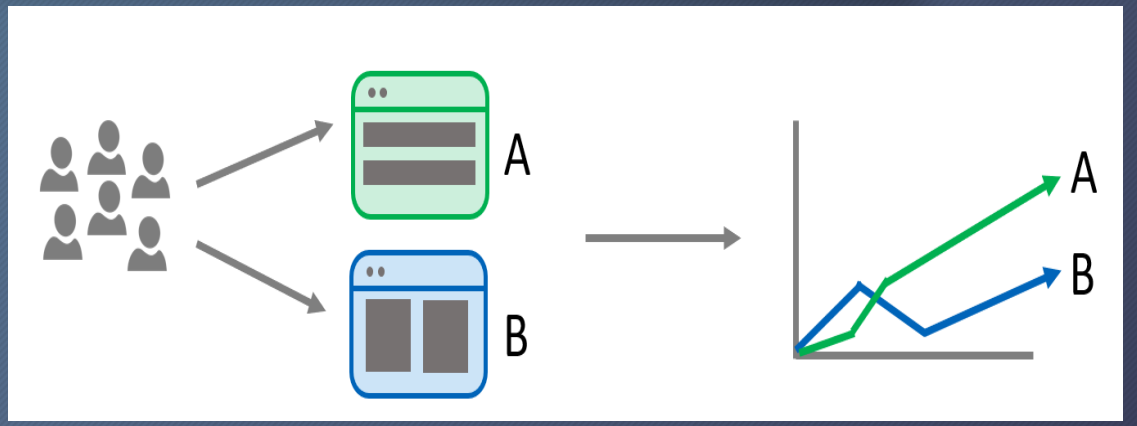
- Using a combination of feature toggles, deployment slots, and Traffic Manager, you can achieve complete control over the traffic flow and enable your canary release.
- You deploy the new feature to the new deployment slot or a new instance of an application, and you enable the feature after verifying the deployment was successful.
- Next, you set the traffic to be distributed to a small percentage of the users.
- You carefully watch the application's behavior, for example, by using application insights to monitor the performance and stability of the application.

Dark launching

- Dark launching is in many ways like canary releases. However, the difference here's that you're looking to assess users' responses to new features in your frontend rather than testing the performance of the backend.
- The idea is that rather than launch a new feature for all users, you instead release it to a small set of users. Usually, these users aren't aware they're being used as test users for the new feature, and often you don't even highlight the new feature to them, as such the term "Dark" launching.
- A Company builds and launches rockets to launch satellites. When they have a new version of a sensor, they install it alongside the old one. All data is measured and gathered both by the old and the new sensor. Afterward, they compare the outcomes of both sensors. Only when the new one has the same or improved results the old sensor is replaced.

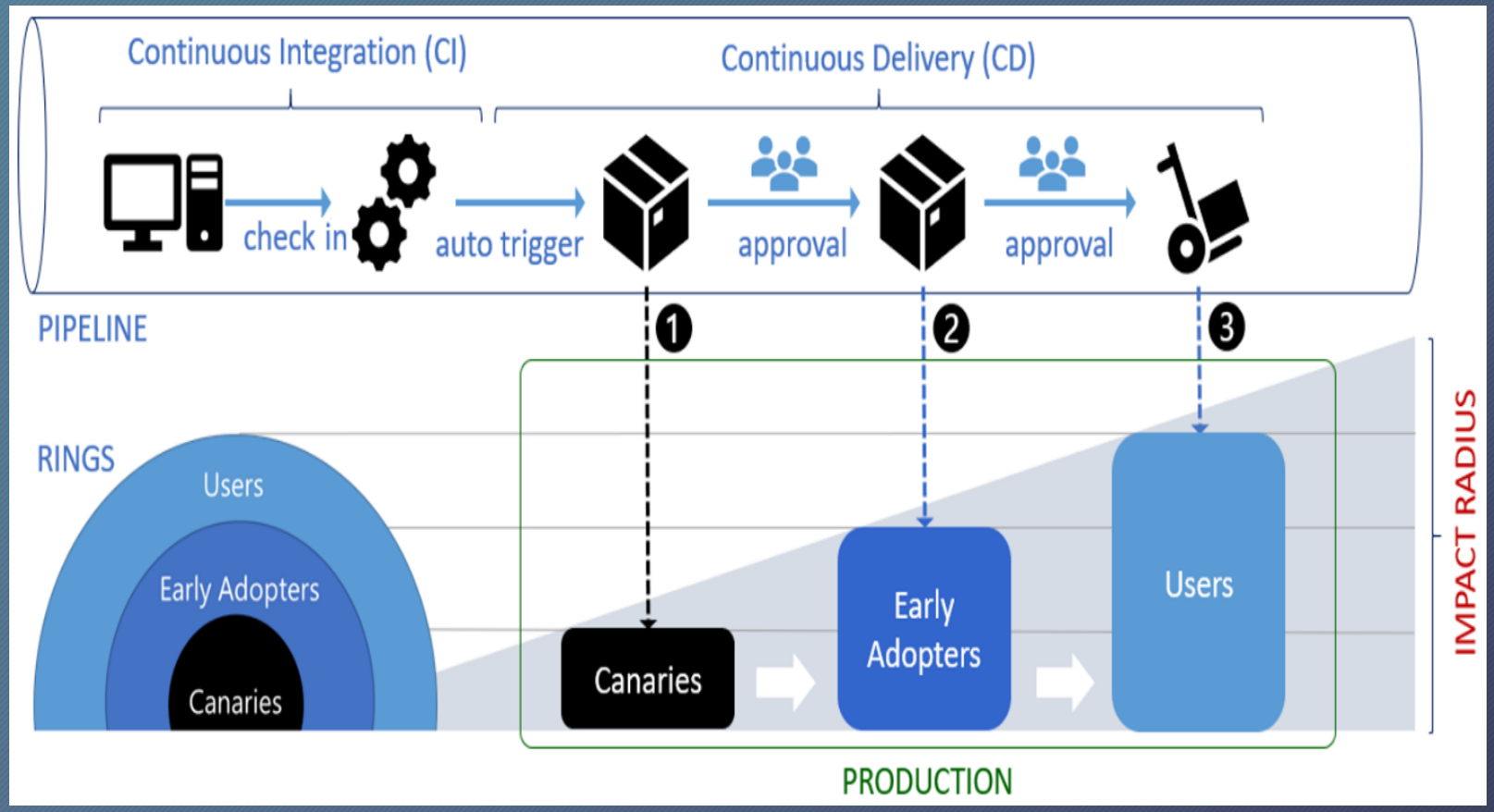
A/B testing

- A/B testing (also known as split testing or bucket testing) compares two versions of a web page or app against each other to determine which one does better.
- A/B testing is mainly an experiment where two or more page variants are shown to users at random.



- Also, statistical analysis is used to determine which variation works better for a given conversion goal.
- A/B testing isn't part of continuous delivery or a pre-requisite for continuous delivery. It's more the other way around.
- Common aims are to experiment with new features, often to see if they improve conversion rates.

Deployment Rings



Deployment Rings

- Progressive exposure deployment, also called ring-based deployment
- They support the production-first DevOps mindset and limit the impact on end users while gradually deploying and validating changes in production.
- Impact (also called blast radius) is evaluated through observation, testing, analysis of telemetry, and user feedback.
- In DevOps, rings are typically modeled as stages.
- Rings are, in essence, an extension of the canary stage. The canary release releases to a stage to measure impact. Adding another ring is essentially the same thing.
- With a ring-based deployment, you first deploy your changes to risk-tolerant customers and progressively roll out to a more extensive set of customers.
- The Microsoft Windows team, for example, uses these rings.

Deployment Rings