WHAT IS THE CANARY DEPLOYMENT & RELEASE PROCESS?

Canary release of software is a deployment method that combines characteristics of other deployment options, creating an ideal modern strategy. It uses a step-by-step rollout process with effective monitoring and rollback options to minimize the risks associated with introducing new software.

The name comes from the use of canary birds in coal mines in the past. They would usually fill the tunnels with soft bird songs, but ultimately had a more sinister purpose as toxic-gas detectors. Because canaries are more sensitive to gas than humans, they would fall ill or die at low exposure levels, signaling to the miners to evacuate before it was too late.

Today, the figurative “canaries” in canary deployment are small subsets of users who are exposed to new updates before others. They typically have a small window of time to test the rollout and determine if the software is ready for deployment to more or all users. Fortunately, the only consequences they might suffer are inconvenient user experiences from latent bugs, unlike the canaries of the coal mines in days past.

Comparison With Other Deployment Types

Software deployment methods over time have changed significantly due to various factors. Modern challenges in the field include the growing ubiquity of cloud-based and distributed applications, increasingly frequent deployments, microservice architectures, and layering of multiple development teams.

With so many factors to take into consideration, there's a lot that can go wrong at any time. Canary release helps address some of the problems in such modern architectures by cutting the process into small monitored steps. Below are two common alternative types of deployment from which canary release combines some features, providing the best of two worlds.

A/B or Blue-Green

This method is quite similar but not identical to canary deployment. It's common for people to mix up A/B testing and canary deployment, so be sure to stay aware of their differences while still appreciating how they can complement each other.

In this method, two identical production environments called Blue and Green (or A and B) run in parallel. At any given time, only one of them is live and serving all traffic while the other is idle. As the software reaches the final stages of preparation, it goes through testing in the idle environment. Once the final testing there completes, all traffic will change routes to the new software.

Rolling/Phased/Step

A rolling release staggers software deployment over multiple phases, replacing an old application with a new one in steps over time. The new and old versions temporarily coexist with the updated software added to one server (or a subset of servers) at a time, rather than an entire duplicate environment as in blue-green deployment.

Canary: The Best of Both Worlds

Canary deployment combines the two above methods into one that is even more risk-averse. It involves first deploying a new code application in part of the production infrastructure for testing, as in blue-green deployment. But rather than having a fully cloned environment, this testing environment is just a small portion of the total infrastructure.

Once the update receives approval for release, canary deployment makes use of the phased approach of rolling deployment to expose a subset of users to the new software. The number of users routed to the new application then increases over time as the software receives confirmation of functionality. In this manner, the new and old versions exist simultaneously for a little while, but the new version progressively replaces the older one.

Infrastructure and Requirements

The primary infrastructure requirement for canary deployment is a partition. Most canaries partition the user base, but an alternative is to partition based on instances. If opting for partitions by instance, you can configure a subset of the available instances to use the new software. However, it is best to partition by user because they are the final destination of the software. It is also typically easier to do so as there are several ways in which to separate users.

A user partition can group the users by simple factors such as geographical region, location, or time zone. It could also be a more randomized selection of a certain percentage of the overall user pool. For a more narrow focus, partitioning based on certain membership types or external vs. internal divisions can center the software test on specific groups.

Overall, aim to partition in such a way that the group of users has a high level of trust and/or the loss

of their trust will not have a high impact on the organization overall. Certain tooling can make it easier to manage various user partitions and can also save some development time.

Beyond infrastructure, you'll also need to be sure to have adequate tracking capabilities. Monitoring the positive and negative impact on users and applications is vital for successful canary deployment. Proper analytics such as latency, volume, error count, and memory usage take the monitored data and turn it into useful information about the release. There are some open-source options as well as paid commercial products available to help you incorporate high-quality analytics and monitoring according to your company's needs.

Canary Applications

The applicability of canary deployment varies in different circumstances. Some situations in which you want to avoid using it are when safety, critical system, or valuable assets are on the line. For example, critical systems running something like a nuclear power plant cannot tolerate even the smallest failure, thus requiring something more concrete than a canary release. Software that manipulates large or frequent financial transactions would also require extra care.

Situations in which canary is optimal are those where other forms of testing are difficult or those which require mitigation of risk. If a service depends on upstream systems that cannot receive proper testing, then canary deployment can help successfully integrate it. Additionally, if an application consists of multiple services with independent change rates, canary deployment provides a means of verifying functionality in a realistic environment. The small percentage of traffic directed to the environment via partitioning also mitigates high operational risk.

Challenges of Canary Deployment

While canary deployment provides many benefits, it still comes with some drawbacks to consider:

- Added complexity
- Difficulty in smooth rollout if the software is installed on the users' devices
- Manual release can take time and result in errors
- Requires high visibility of the user behavior, system, and application
- Challenging to manage database schema changes and API versions

Most of these challenges depend on the capabilities of the given organization as well as the type of software update itself. In general, you will need to be prepared to manage additional code, services, and components during the time of the canary release. For example, multiple versions of the software will need to run at the same time, but the brief time of overlap means that this is a short-lived problem. Try to keep the number of versions to a minimum to reduce complications.

The type of application may have a larger effect, as canary deployment is more difficult when it comes to software installed on user devices. It is still possible, but the rollout may be less smooth because the development team will have less control.

To ease the process of canary release in all cases, the development team should employ automation. Automating the deployment speeds up the process and reduces errors that often come from manual methods.

Other requirements for ease-of-use include maintaining high visibility of the system and its users

through tight monitoring, analytics, and strategy implementation. These capabilities will also make it easier for the development team to manage incompatibilities.

Benefits of Canary Deployment

Releasing software through canary deployment supports modern infrastructures in several ways:

- Phased rollout of new applications reduces user exposure to negative operational issues
- Eases rollback implementation in case of problems
- Reduces deployment cycle length, allowing deployment to production earlier and more often
- Increases customer trust by monitoring and reducing potential software problems while simultaneously supporting innovation and improvement of systems

Although some users will inevitably still experience issues, the phased rollout of canary deployment keeps the number as low as possible. And when problems occur, the ease of rollback allows rapid recovery that minimizes the overall negative experience for any given user.

Each canary deployment should optimally only take minutes or hours to complete, making it highly efficient for fast and frequent updates. The resulting shorter deployment cycles benefit organizations by reducing time to market and giving customers more product value in reduced time. It also allows customer feedback to reach the production team more rapidly, leading to a quicker response to problems.

The combination of minimized user exposure, fast recovery, and frequent deployment combine to make happy customers. They end up having access to constantly improving software systems with only minimal and short-lived negative experiences along the way. Additionally, the morale of the developer team goes up due to ease of deployment and recovery, making it a win-win for all parties.