Canary deployments are often combined with *feature toggles*, in which you wrap all new features in an if-statement. By default, the if-statement defaults to false, so the new feature is toggled off when you initially deploy the code. Because all new functionality is off, when you deploy the canary server, it should behave identically to the control, and any differences can be automatically flagged as a problem and trigger a rollback. If there were no problems, later on you can enable the feature toggle for a portion of your users via an internal web interface. For example, you might initially enable the new feature only for employees; if that works well, you can enable it for 1% of users; if that's still working well, you can ramp it up to 10%; and so on. If at any point there's a problem, you can use the feature toggle to ramp the feature back down. This process allows you to separate *deployment* of new code from *release* of new features.

Deployment server

You should run the deployment from a CI server and not from a developer's computer. This has the following benefits:

Fully automated

To run deployments from a CI server, you'll be forced to fully automate all deployment steps. This ensures that your deployment process is captured as code, that you don't miss any steps accidentally due to manual error, and that the deployment is fast and repeatable.

Consistent environment

If developers run deployments from their own computers, you'll run into bugs due to differences in how their computer is configured: for example, different operating systems, different dependency versions (different versions of Terraform), different configurations, and differences in what's actually being deployed (e.g., the developer accidentally deploys a change that wasn't committed to version control). You can eliminate all of these issues by deploying everything from the same CI server.