Continuous Integration and Continuous Delivery

Continuous Integration (CI) is the engineering practice of frequently committing code in a shared repository, ideally several times a day, and performing an automated build on it. These changes are built with other simultaneous changes to the system, which enables early detection of integration issues between multiple developers working on a project. Build breaks due to integration failures are treated as the highest priority issue for all the developers on a team and generally work stops until they are fixed.

Paired with an automated testing approach, continuous integration also allows us to also test the integrated build such that we can verify that not only does the code base still build correctly, but also is still functionally correct. This is also a best practice for building robust and flexible software systems.

Continuous Delivery (CD) takes the **Continuous Integration (CI)** concept further to also test deployments of the integrated code base on a replica of the environment it will be ultimately deployed on. This enables us to learn early about any unforeseen operational issues that arise from our changes as quickly as possible and also learn about gaps in our test coverage.

The goal of all of this is to ensure that the main branch is always shippable, meaning that we could, if we needed to, take a build from the main branch of our code base and ship it on production.

If these concepts are unfamiliar to you, take a few minutes and read through Continuous Integration and Continuous Delivery.

Our expectation is that CI/CD should be used in all the engineering projects that we do with our customers and that we are building, testing, and deploying each change we make to any software system that we are building.

For a much deeper understanding of all of these concepts, the books Continuous Integration and Continuous Delivery provide a comprehensive background.

Why CI/CD

- We want to have an automated build and deployment of our software
- We want automated configuration of all components
- We want to be able to quickly re-build the environment from scratch in case of disaster

- We want the latest version of the code to always be deployed to our dev/test environments
- We want a reliable release strategy, where the policies for release are well understood by all

The Fundamentals

- We run a quality pipeline (with linting, unit tests etc.) on each PR/update of the main branch
- All cloud resources (including secrets and permissions) are provisioned through infrastructure as code templates – ex. Terraform, Bicep (ARM), Pulumi etc.
- All release candidates are deployed to a non-production environment through an automated process (ex Azure DevOps or Github pipelines)
- Releases are deployed to the production environment through an automated process
- Release rollbacks are carried out through a repeatable process
- Our release pipeline runs automated tests, validating all release candidate artifact(s) end-to-end against a non-production environment

Tools

Azure Pipelines

Our tooling at Microsoft has made setting up integration and delivery systems like this easy. If you are unfamiliar with it, take a few moments now to read through Azure Pipelines (Previously VSTS) and for a practical walkthrough of how this works in practice, one example you can read through is CI/CD on Kubernetes with VSTS.

Jenkins

Jenkins is one of the most commonly used tools across the open source community. It is well-known with hundreds of plugins for every build requirement. Jenkins is free but requires a dedicated server. You can easily create a Jenkins VM using this template

TravisCI

Travis CI can be used for open source projects at no cost but developers must purchase an enterprise plan for private projects. This service is ideal for validation of PR's on GitHub because it is lightweight and easy to set up with no need for dedicated server setup. It also supports a Build matrix feature which allows accelerating the build and testing process by breaking them into parts.

CircleCI

CircleCI is a free service for open source projects with no dedicated server required. It is also ideal for validation of PR's on GitHub. CircleCI also allows workflows, parallelism and splitting your tests across any number of containers with a wide array of packages pre-installed on the build containers.

AppVeyor

AppVeyor is another free CI service for open source projects which also supports Windows-based builds.

Last update: August 26, 2024