Continuous Integration and Continuous Delivery (Building Real-World Cloud Apps with Azure)

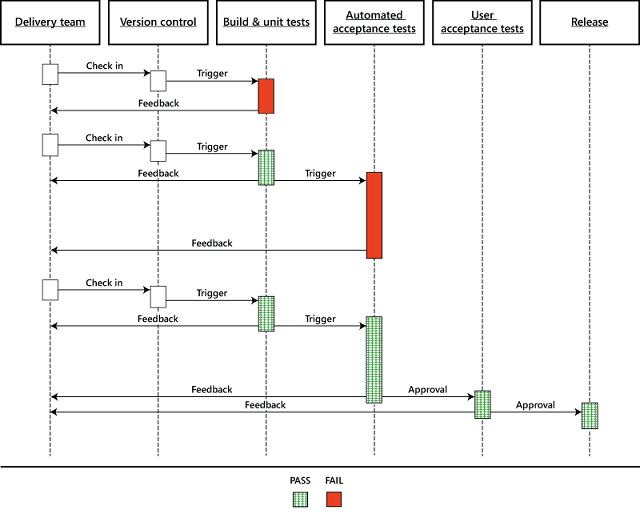
The first two recommended development process patterns were [Automate Everything](https://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/automate-everything) and [Source Control](https://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/source-control), and the third process pattern combines them. Continuous integration (CI) means that whenever a developer checks in code to the source repository, a build is automatically triggered. Continuous delivery (CD) takes this one step further: after a build and automated unit tests are successful, you automatically deploy the application to an environment where you can do more in-depth testing.

The cloud enables you to minimize the cost of maintaining a test environment because you only pay for the environment resources as long as you're using them. Your CD process can set up the test environment when you need it, and you can take down the environment when you're done testing.

Continuous Integration and Continuous Delivery workflow

Generally we recommend that you do continuous delivery to your development and staging environments. Most teams, even at Microsoft, require a manual review and approval process for production deployment. For a production deployment you might want to make sure it happens when key people on the development team are available for support, or during low-traffic periods. But there’s nothing to prevent you from completely automating your development and test environments so that all a developer has to do is check in a change and an environment is set up for acceptance testing.

The following diagram from [a Microsoft Patterns and Practices e-book about continuous delivery](http://aka.ms/ReleasePipeline) illustrates a typical workflow.  Click the image to see it full size in its original context.

[](http://msdn.microsoft.com/en-us/library/dn449955.aspx)

How the cloud enables cost-effective CI and CD

Automating these processes in Azure is easy. Because you’re running everything in the cloud, you don’t have to buy or manage servers for your builds or your test environments. And you don’t have to wait for a server to be available to do your testing on. With every build that you do, you could spin up a test environment in Azure using your automation script, run acceptance tests or more in-depth tests against it, and then when you’re done just tear it down. And if you only run that server for 2 hours or 8 hours or a day, the amount of money that you have to pay for it is minimal, because you’re only paying for the time that a machine is actually running. For example, the environment required for the Fix it application basically costs about 1 cent per hour if you go one tier up from the free level. Over the course of a month, if you only ran the environment an hour at a time, your testing environment would probably cost less than a latte that you buy at Starbucks.

Visual Studio Online

One of the things we’re working on with Visual Studio Online (VSO), formerly known as Team Foundation Service, is to make it work really well with continuous integration and delivery. Here are some key features of VSO:

* It supports both Git (distributed) and TFVC (centralized) source control.
* It offers an elastic build service, which means it dynamically creates build servers when they’re needed and takes them down when they’re done. You can automatically kick off a build when someone checks in source code changes, and you don’t have to have allocate and pay for your own build servers that lie idle most of the time. The build service is free as long as you don’t exceed a certain number of builds. If you expect to do a high volume of builds, you can pay a little extra for reserved build servers.
* It supports continuous delivery to Azure.
* It supports automated load testing. Load testing is critical to a cloud app but is often neglected until it's too late. Load testing simulates heavy use of an app by thousands of users, enabling you to find bottlenecks and improve throughput —before you release the app to production.
* It supports team room collaboration, which facilitates real-time communication and collaboration for small agile teams.
* It supports agile project management.

For more information about the continuous integration and delivery features of VSO, see [Visual Studio Lab Management](http://www.visualstudio.com/en-us/explore/lab-management-vs.aspx) and [Visual Studio Release Management](http://www.visualstudio.com/en-us/explore/release-management-vs.aspx). An application monitoring feature, [Application Insights for Visual Studio Online](http://msdn.microsoft.com/en-us/library/dn481095.aspx), is in preview (available to try but not released for production use yet).

If you’re looking for a turn-key project management, team collaboration, and source control solution, check out VSO. The service is free for up to 5 users, and you can sign up for it at [http://www.visualstudio.com](http://www.visualstudio.com/). Summary

The first three cloud development patterns have been about how to implement a repeatable, reliable, predictable development process with low cycle time. In the [next chapter](https://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/web-development-best-practices) we start to look at architectural and coding patterns.