Build Forge has some advanced capabilities which can be leveraged to greatly reduce your build cycles. Those capabilities also allow developers to participate more in the build processes by delivering developer self service type capabilities.

First let's look at some of the core capabilities that will allow you to greatly reduce your build cycles. One of the significant benefits of Build Forge is the ability to accelerate build and release processes. By threading steps, tasks can be run in parallel across a pool of servers for dramatically lower run times.

Many customers have reported significant time reductions. In one customer's case, they reduced a 60-hour process just down to three hours.

Here as we look at the project for this specific project we can see that after we access our source code we're running multiple steps in parallel across multiple platforms. Build Forge orchestrates the sending of instructions to the various platforms, running those instructions as well as bringing back the results for analysis to make sure that everything is running fine within the process.

As I said, this can greatly reduce just the whole process cycle time to build a project as well as get a project all the way through to release and production.

Pooling is another capability which can greatly reduce your cycle times. For example, in many development environments a server is dedicated to a specific project. This creates choppy cycles where the hardware is over utilized during certain periods and others, sit idle.

Build Forge allows servers to be grouped into what refer to as pools so hardware can be shared across projects to increase hardware utilization and reduce hardware spend. In addition, this system provides fault tolerance. For example, if the server goes down in one of the pools, Build Forge will redirect the workload to another available server in the pool.

This feature alone can greatly get...you can get more benefit out of your build farms and your systems that are involved in your build on process, more throughput as well as deal with, you know, potentially catastrophic failures where if one of the servers goes down Build Forge will automatically send the workload to another server.

Build Forge also increases the efficiency of developers and configuration managers by providing real-time information about the state of a release. For example, when I look at one of my projects, for a specific step I can have it send the message if this step fails.

And what this message would look like as an e-mail which will go to an individual person or to a group of people, which may look something like this. You know, informing me that the project has failed where it failed, as well as given me a link to get into the system and see exactly what build it was that failed.

This level of communications gets rid of a lot of the manual interventions and the manual communications that go on in a typical build environment, thus, yet another way, compressing the build cycle into something that's more reasonable.

If teams prefer to build and release at designated times, Build Forge has a built in scheduler that may be used to schedule reoccurring builds and releases.

Otherwise, builds can set to run manually by hitting the start button. The user may also change the run-time variables as needed. This can eliminate specific steps of the process it desired.

This is useful for developers who might want to do a pre-flight build prior to doing a nightly build to reduce integration errors.

Once a process is automated in Build Forge it can be run frequently through continuous integration. As a result teams can conduct iterative code build test cycles with the push of a button.

Build Forge has developed adapters for the major source control systems for out of the box continuous integration capabilities.

With Build Forge, using the management console Web-based interface, you can give secure access to your developers thus supporting more of a self-service role. Build Forge also includes capability to plug into the developer's IDE.

Here we have Build Forge Prism, which extends the Build Manager capabilities to the developer's desktop by leveraging their IDE technology. Access to third party vendors, applications within the IDE, is typically referred to as plug-ins, and this plug-in is a lightweight set of code that provides developers seamless access to one or more external applications.

Build Forge also allows developers to view and build results within a controlled environment directly from their IDE. Capabilities are not limited to just viewing results, initiating builds and doing [assess] of currently running builds.

Build Forge...Build Forge users cannot from their IDE plug-in create new projects, delete projects or alter existing project steps. And we also leverage the system level permissions when the projects are honored so they can only view and run projects that they are granted permission for.

The Build Forge plug-in also enables developers to instruct a Build Forge project to build using a selected list of code changes and instead of extracting code from the source control system.

For example, here I have my project and when I look at the properties of that project, I have Build Forge project artifacts with all [defined]. What this will do is allow me to select which local source code do I want to use in this build and this pre check-in feature means that I'm able to test my code that I'm currently working on right out of my workspace before I check it in and potentially impact someone else by sending in bad code.

So these capabilities such as threading, pooling, notifications, intelligent log analysis, all work together to really help reduce your cycle times as well as extending self service type capabilities to the development teams.

[END OF SEGMENT]