In this demonstration, we're going to talk about how small frequent releases can reduce the error rate and help make things easier and better to ship.

And that's one of the core fundamentals of continuous delivery. I'm gonna take a look here at the Parts Unlimited Team Services site that I have and this won't be any of your labs, but you see that I have many user stories here. One of them has raised the discount to 15%, the look and feel should be consistent, create an automated build.

All of these are relatively small adjustments to the website. There are things that shouldn't take too long to implement. They are not very large scale bulky items. And they all may add up to something wonderful. But we're incrementing, we're delivering them in small incremental pieces.

Here's a very small example, and this is just a tweak to the change, but raise the discount to 15%. You can see that there's been code changes made, and it's one of the other videos that you can watch if you like. And we change the discount to 15%, that’s the code that we created in order to do that. I'm gonna actually take a look at that code.

If we click on it, it's going to bring us to the pull request that this was implemented in and we can see the delta. Now with this was the only thing that's changed. We have a very small change, we changed the index.chtml file and it was merged successfully and it was committed to the code branch.

So now we have a small change that was pushed to the dev environment, and I do want to show you the dev environment. The dev environment itself has, let's get there, the 15% discount. But if we go out further, we'll note that in the staging environments, we're still at the 3.14159% discount. So we've automatically pushed that out to the dev environment, where we're getting confidence.

Now let's say that for some reason something broke. If something broke, it's very easy to go back and find what broke. In this case it was just one line of code that changed, or if I'd accidentally mistyped something or if I had forgotten a beginning paragraph symbol, something like that, we would immediately know what the problem was because it's such a small change.

And that's why it is rolling these very fast changes continually out into environments is very, very valuable. Now if we wanted to, we could also go out and deploy our app. We can go to our release phase and I want to note a couple things.

One is, we're going to the dev environment at every single release. So if I look at just our dev build, you'll see that that dev build kicked off automatically and it deployed successfully to our dev environment. And that's that fast, continuous delivery.

Now, I'd like to see this go even further, because it can reduce our risks when we bring it out to production as well. There might be things in production that don't exist in dev. So to do so, we can then go promoting it through the various things. I can approve and we can approve and push it through that pipeline.

The way we have it set up here, it's a manual to step which is fine. You could also have it set up as an automated step as part of your continuous delivery pipeline if that made sense for your organization.

So key takeaway, small units of work that can be delivered rapidly into an environment, preferably production. That helps reduce risk because we're shipping less code and we automate all of it to make sure that we don't r**un into errors as part of that delivery process**