Jason Ritenour: Hi there. I'm Jason Ritenour. I'm a cloud domain architect with Red Hat, and, today, I'm going to talk about Ansible and I'm going to show you some of the ways we can do automation against the system that we don't have access to.

Jason Ritenour: Imagine you're a systems administrator and you've got several hundred web servers running maybe some static content, maybe some dynamic content, but you need to do an update on these web servers and you can't... You don't have direct command-line access to get in and make any changes. You do, however, have access to the code repository where the content is stored, so I'm going to show you.

Jason Ritenour: As you can see on my screen, I've got this webpage up, and, if you notice, there's a misspelling. That's not how we spell index, so I want to go in and I want to make a change to that page dynamically without actually getting in and opening it up and... like a text editor and making a change that way. Especially if I've got several dozens or several hundred servers running the same content, that's obviously completely inefficient, so I'm going to go here to my GitHub code repository, and you can see the Ansible playbook I use to deploy this web server. You notice it went through and it did things like install Apache. It set this content.

Jason Ritenour: You can see the misspelling is actually here in this playbook, and then it allowed traffic through the firewall and it enabled the services appropriately, so what I'm going to go do here at this point is I'm going to edit this code repository, this playbook right here, and I'm going to change this misspelling. I'm going to correct it. I'm going to commit my changes, add a message here saying that I corrected a misspelling.

Jason Ritenour: Now, my playbook is updated at this point and the spelling is correct, but I need to actually run this playbook on this affected system in order to correct that mistake, so I'm going to go here to job templates. This is Ansible Tower, by the way. This is our graphical user interface for Ansible that allows you to manage your systems at scale. It allows you to collect systems into inventories so that they're more easily manageable. We can pull in your playbooks from source control.

Jason Ritenour: I'm going to run this install/update Apache playbook at this point, and this is going to go through. The first thing it's going to do is it's going to sync my inventory because it's getting my inventory from the virtualization platform that my web server is running on. In this case, it's Red Hat virtualization, so Ansible is going to go talk to REV to see if there's any new virtual machines that it needs to be aware of and then it's going to pull in the updated version of that playbook from source control, and then it runs through its process.

Jason Ritenour: The first thing it does is gather facts where, basically, it's talking to the system and it's finding out information about it. It's asking it things like what is your host name, what is your IP address, how much free memory do you have, how much free hard drive space, et cetera, and then you'll notice it went through and it... Some of these steps they, "Okay." Some of them say, "Changed." When it says okay, that means that it didn't actually do anything to the system because Ansible went through and it queried it and it found out that it wasn't going to actually have to do anything to get to that state described in the playbook, so, since Apache was already installed, it just said, "Okay." It skipped over that.

Jason Ritenour: It did, however, change our index file, so, if we click on this, we can get some information about what actually happened here, and it tells you the file that it went through and made the modification against. It does this information like the owner, group, the permissions on it, et cetera, and then everything else. The firewall rule was already in place. It did restart the firewall because that's something that is going to happen regardless. It's always going to restart when I say to restart the service, but if I go back over to my webpage now and I do a refresh, you'll see that my index is now correctly spelled.

Jason Ritenour: Now, this is good for just editing static content, but what if I wanted to do something that allowed some user input at the time of running the playbook? Let me go back here again. I'm going to make another change. I'm going to edit the index again, and you'll notice this variable here. Anytime you see in a playbook something enclosed in the double curly braces, that means it's a variable. Now, in this case, Ansible_hostname is an example of a fact that's gathered at the time that the playbook runs, so that's something that's dynamically going to be pulled from each system, so, that web, if you go back over, it says, "This is the index of Web." Web is the host name of this system, so, if I go through and I change this to something else like... We'll call this run message.

Jason Ritenour: Actually, let me go ahead and change all of this. Let's just eliminate all this text. Run message is going to be our variable name for the message we're going to put here on this index, so I've updated my source control. I've saved it. I'm going to go back here to my Ansible Tower, and, now, I'm going to go in and edit this job template because I want to make some changes. I want it. You'll notice, when I ran it before, it just automatically executed without any user input. This time, I'm going to go down here to my extra variables and I'm going to ask it to prompt me on launch, and I'm going to put in a variable called run message, and I'm going to make the default just a blank space. I'm going to save that.

Jason Ritenour: A couple other things I want to point out here, you'll notice I can enable things like a webhook, for example, so I can make it so that, by using web hooks, as soon as I commit the source control change, it would automatically trigger this playbook and run the... and make the appropriate changes against the systems in this inventory, but I'm not doing that in this case. Everything is being done manually, at least triggering the launch of the playbook, so, again, I'm going to go ahead and launch this job template. It will prompt me now, asking me what I want to use for this run message, so I'm going to say something like, "This site will be under maintenance Saturday, 2/8." I'm pretty sure the eighth is Saturday. Hopefully, I'm right.

Jason Ritenour: I'm going to go ahead and click next here, and then I'm going to actually launch this, so, now, again, it's going to go through this process. It's going to reach out to the... do an inventory update. It's going to pull in the most recent version of this playbook from source control, and you'll see here it's telling me now that it's running with this extra variable that I just provided, saying that the site is going to be under maintenance, so we'll give it a couple seconds for it to pull in the updated inventory in the updated playbooks, and then it will actually start executing the job, so, again, it's gathering facts. It's going to skip over the Apache install install because, again, that's already there. It's changing the index file, nothing to do with the firewall other than restart it, and it's done, so I'm going to go back over to my index again, reload it one more time, and you can see now it's running with that message I supplied, "The site will be under maintenance on Saturday, February 8th."

Jason Ritenour: That's how we can make some changes on a system using, again, changes that we do at our source control to the playbook itself or how we can do it dynamically by supplying input at runtime through Ansible Tower, and, again, imagine running this against several dozens, several hundred servers at once. It's certainly a more sane way to approach rather than going in and making manual changes to the files one by one on each individual system.

Jason Ritenour: One more thing I want to touch on before we wrap up here is a new concept we introduced in Ansible 2.8 and officially supported beginning in 2.9, and that is Ansible Collections. Now, in the past, Ansible modules were married to the Ansible release. We only updated modules when we updated the actual Ansible runtime, so, if we had some changes we made in Ansible modules, they wouldn't come out 'til Ansible 2.5 or 2.6 or what have you. Now, with Ansible Collections, we can allow vendors to make their own versions of modules, and we can certify them independent and decoupled from the Ansible release themselves.

Jason Ritenour: You'll see that here in our Ansible Automation Hub we have an example of several partners that have begun embracing this new way of distributing content. For example, Microsoft, they have several modules for Azure, and this is another. This is a great example of how having this decoupled from the Ansible runtime release has helped us, because cloud providers are constantly adding new services and new resource types, so, because we're able to allow them to release them at their own pace rather than waiting for us to update the Ansible runtime, we can get this new content out there quickly.

Jason Ritenour: Collections consist of more than just modules, however. There can also be different plugins, different actions, other forms of various collateral that could be distributed independent of the actual Ansible runtime, and, again, if you look here in the Automation Hub, this tells you how to actually install these collections. It's simply a matter of running Ansible Galaxy, collection install and then the name of the collection, so, if you want to see more collections other than these ones that we've certified here, you can go to galaxy.ansible.com, and that's a great place to start.

Jason Ritenour: If you are a current Ansible Tower subscriber, you can go here to cloud.redhat.com and check out the Ansible Automation Hub. In addition, we have plans to offer this as an on-prem version that you'll be able to run in your own data center, so look for that coming in the near future, so thanks again, and I enjoyed running this demo for you.