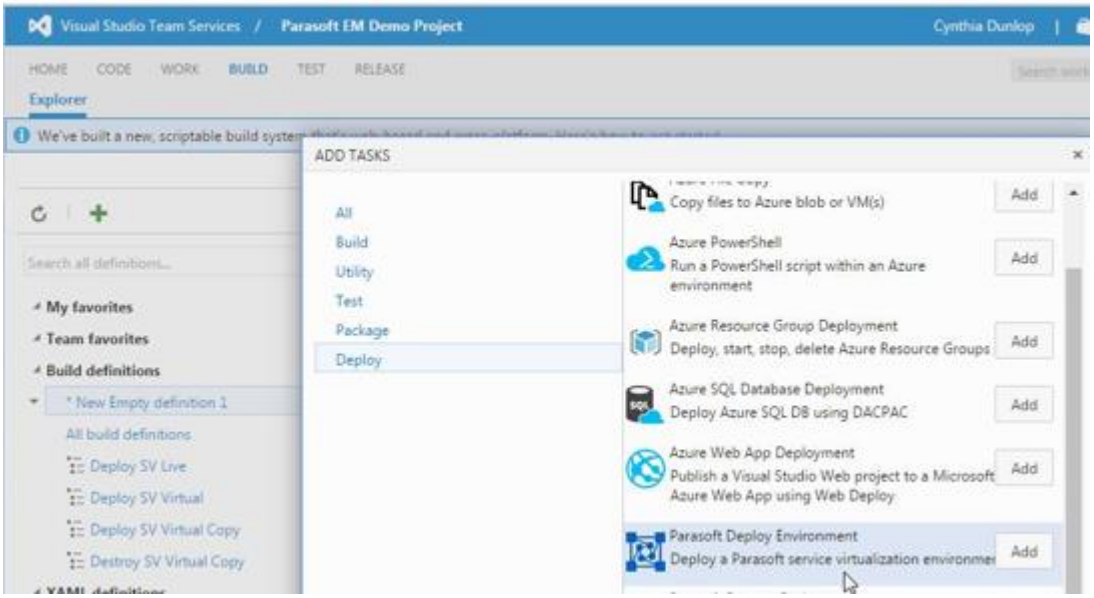
Service Virtualization for Continuous Testing in Microsoft Environments:

With Microsoft® Azure™, Microsoft® VSTS™, and Parasoft® Service Virtualization™

Today's DevTest teams are under pressure to deliver more—and more innovative—software faster than ever before. And now that most organizations are relying on software as a primary interface to the customer, compromising on quality to accelerate a release is not an option. Unfortunately, there's no silver bullet for delivering "quality @ speed" —but one essential element is to have unrestrained access to a trustworthy and realistic test environment (e.g., including the AUT and all of its dependent components). Otherwise, you can't accurately and thoroughly validate the change impacts associated with each user story, or be confident that the evolving application doesn't degrade the overall user experience. Access to a complete test environment not only helps you achieve greater velocity; it also enables you to assess the risk of a release candidate during a CI/CD process and identify high-risk release candidates early in the delivery pipeline.

However, with today's complex systems, this type of test environment access is the exception rather than the rule. Although it was once common for organizations to stand up a local physical staged test environment, the complexity of modern applications has made that approach too slow and cost-prohibitive for today's development processes. Moreover, in many cases, it is downright infeasible due to dependencies that can't be reproduced in the test environment.

With technologies like Microsoft® Azure™ (for elastic scalability), Microsoft® Visual Studio Team Services™ (for build and deployment automation), and Parasoft® Service Virtualization™ (to simulate and access dependencies), there's no reason why a complete, realistic test environment should be out of reach. By taking advantage of a Microsoft ecosystem with VSTS and Azure, organizations gain immediate access to scale and bandwidth. This provides the resources needed to enable flexible and ubiquitous access to application stacks that are under your control for DevTest purposes. But what about the dependent system components that are beyond your scope or control (e.g., third-party applications, SAP, mainframes, not-yet-implemented services, etc.)? You can simulate their behavior using Parasoft Service Virtualization—eliminating the final test environment access gap that commonly impedes teams' testing efforts.



What is Service Virtualization?

Service Virtualization provides access to the dependencies that are beyond your control, still evolving, or too complex to configure in a test lab. It involves simulating the behavior of software components to remove dependency constraints on development and testing teams. Such constraints occur when "dependent components" connected to the application under test (e.g., APIs, 3rd-party services, databases, applications, and other endpoints) are:

- Not yet completed or still evolving
- Controlled by a third-party or partner
- Available for testing only in limited capacity or at inconvenient times
- Difficult to provision or configure in a test environment
- Needed for simultaneous access by different teams with varied test data setup and other requirements
- Restricted or costly to use for load and performance testing

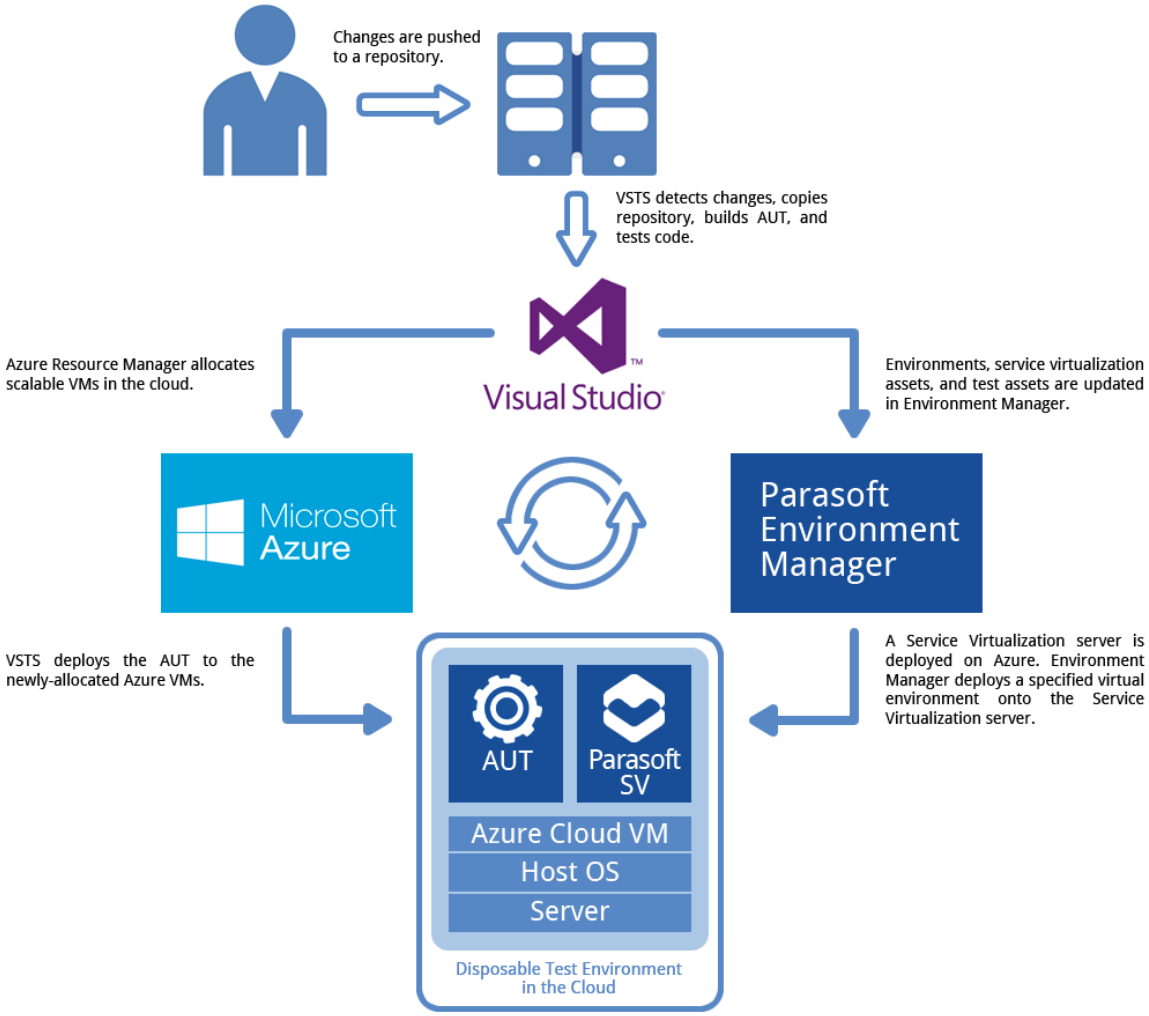
Rather than simulating entire systems, service virtualization simulates only the specific dependent component behavior that's critical for development and testing purposes. Simulation is typically achieved by recording interactions with a live application at the message/protocol level, but it can also be accomplished by other means (creation from Swagger or other definitions, from traffic files, and so on). If desired, the resulting "virtual assets" can be extended and enhanced with additional data, performance profiles, response logic, etc. With the behavior of the dependent components "virtualized," testing and development can proceed without accessing the actual live dependent components.

The combination of Microsoft Azure, Microsoft VSTS, and Parasoft Service Virtualization—operating natively within the Microsoft environment—enables organizations rapidly deploy a complete test environment on demand. The most realistic dependent components available *at that specific point in time* are aggregated from a central repository and then provisioned automatically. The "most realistic set of dependent components" is often a combination of both real components and simulated components delivered via service virtualization.

These simulated test environments are lightweight and Azure-compatible so that when you need to scale (e.g., for performance testing), you can do that on demand. They are also disposable. A test environment can be instantly provisioned from a golden template, used and dirtied, then simply destroyed. There's no need to spend time resetting the environment or test data to its original state. The exact same environment can be instantly spun up whenever it's needed (e.g., for reproducing or verifying defects).

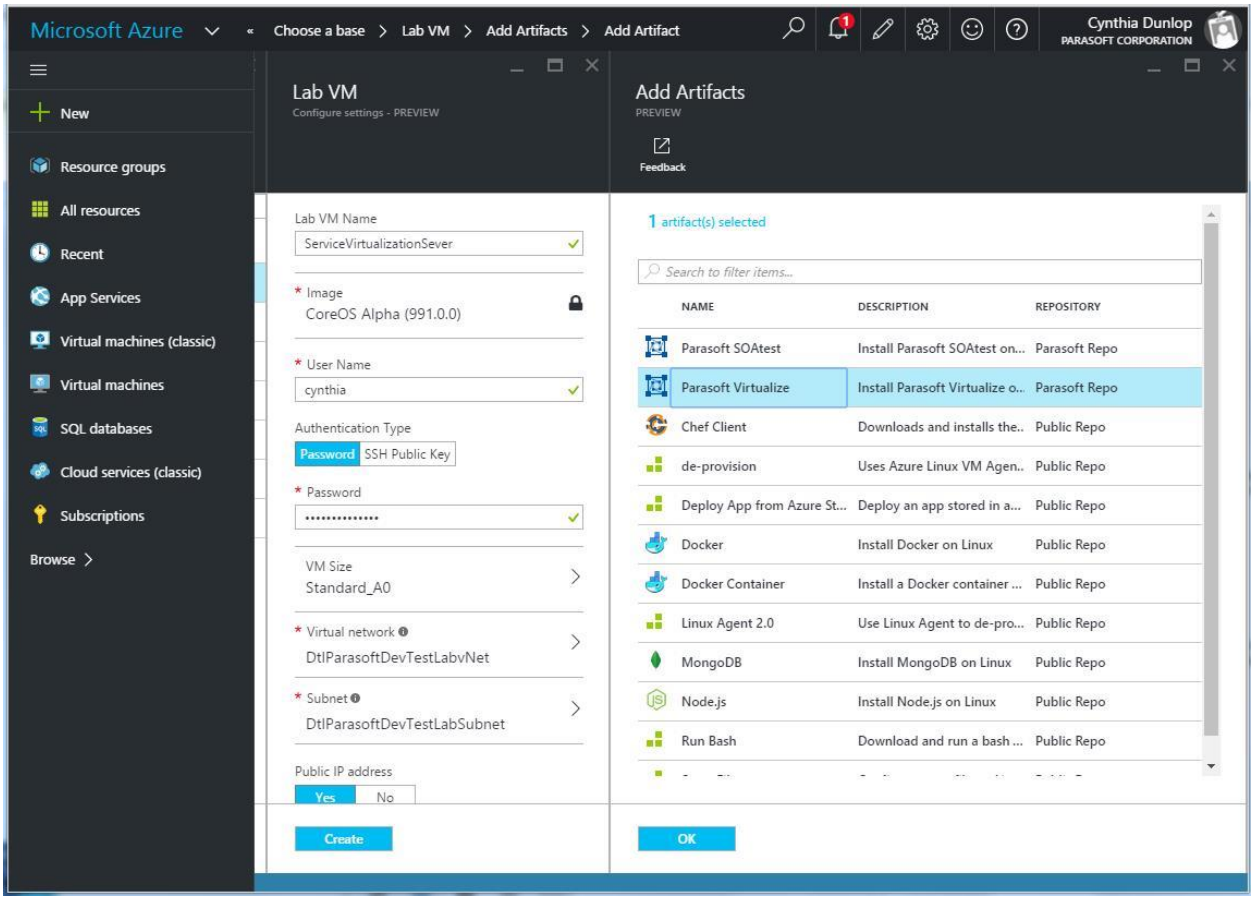
A Technical Look

To streamline and accelerate the provisioning process, you can take advantage of Microsoft Azure and Microsoft VSTS to automatically deploy disposable test environments onto servers running in the cloud—making complete test environments available in a matter of seconds. The following diagram shows one way that you can use Microsoft Azure and Microsoft VSTS to rapidly deploy a complete test environment in less than 10 minutes:

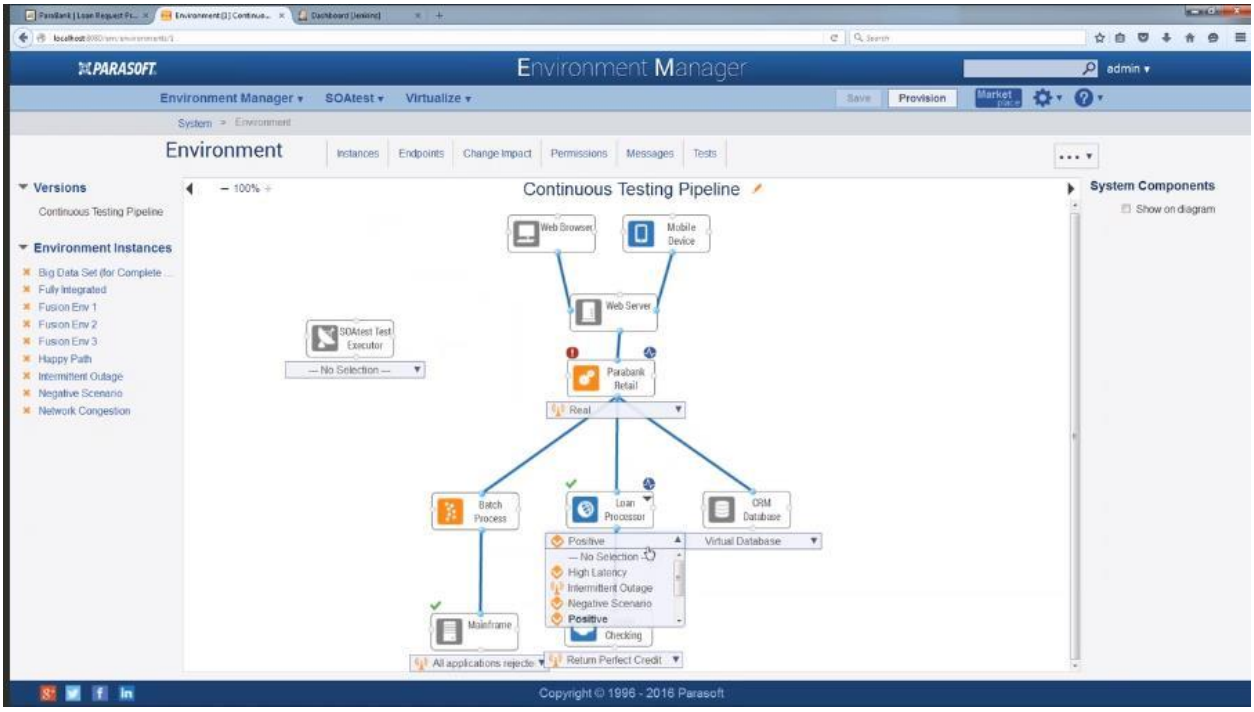


Here's a quick overview of how the service virtualization and environment-focused steps can be configured (we're assuming you already understand how to deploy your AUT).

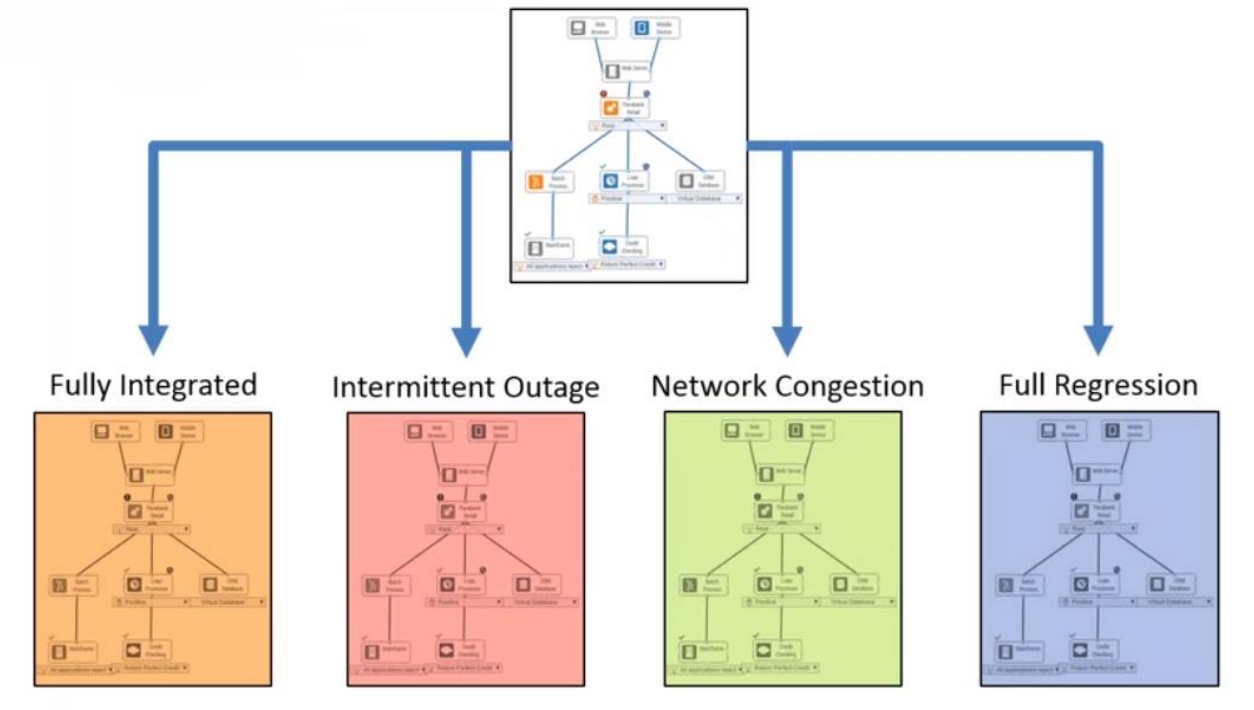
Service Virtualization servers can be automatically deployed to cloud-based Azure VMs that have been allocated by Azure Resource Manager. This not only streamlines installation, but also provides elasticity and scalability.



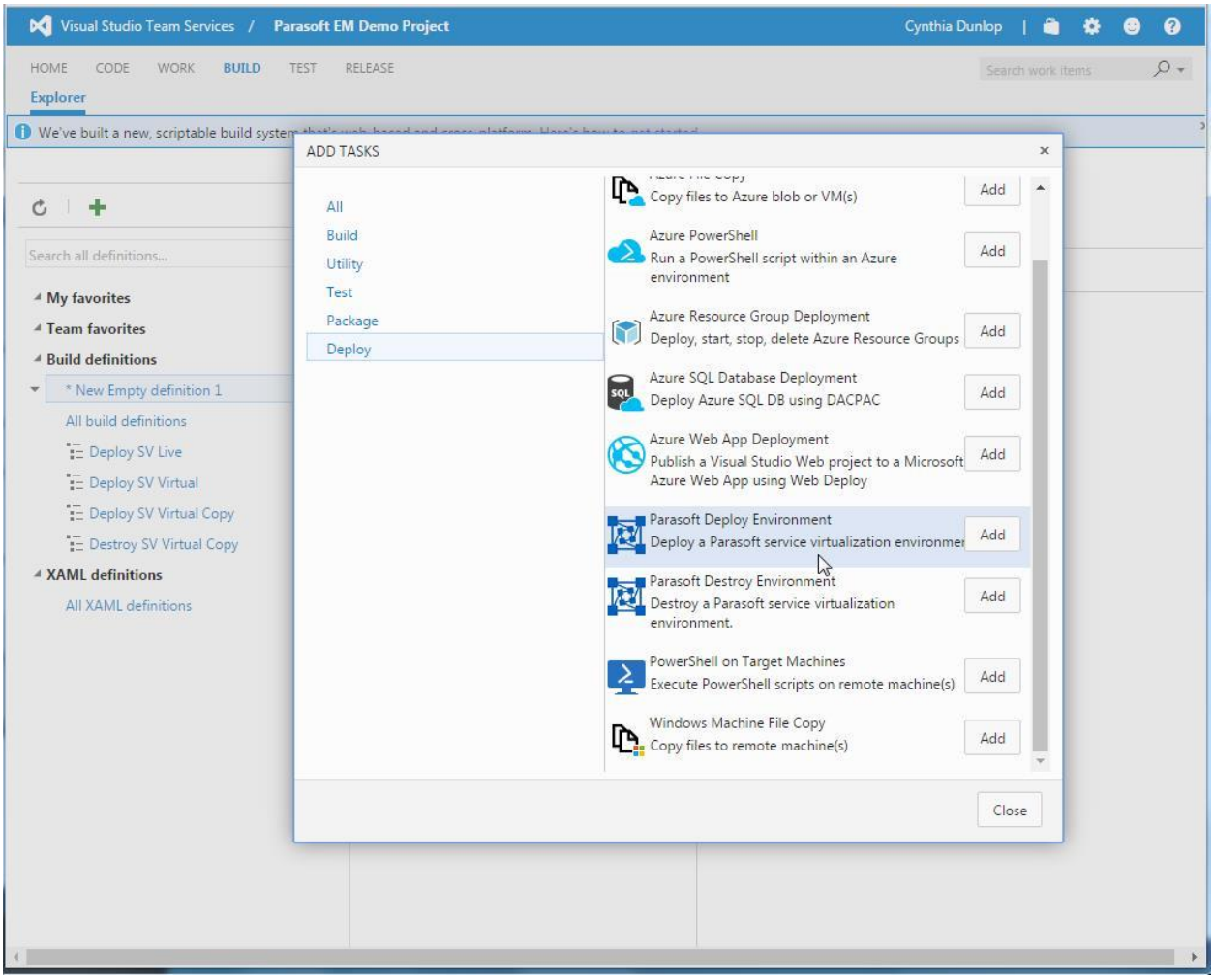
"Golden copies" of simulated test environments are defined using Parasoft Environment Manager, which provides a browser-based interface. A system diagram helps you define a complete environment, including all dependent components. You then leverage simulation technologies to coach the environment to behave in certain ways. For example, assume you have the following environment for a sample banking application:



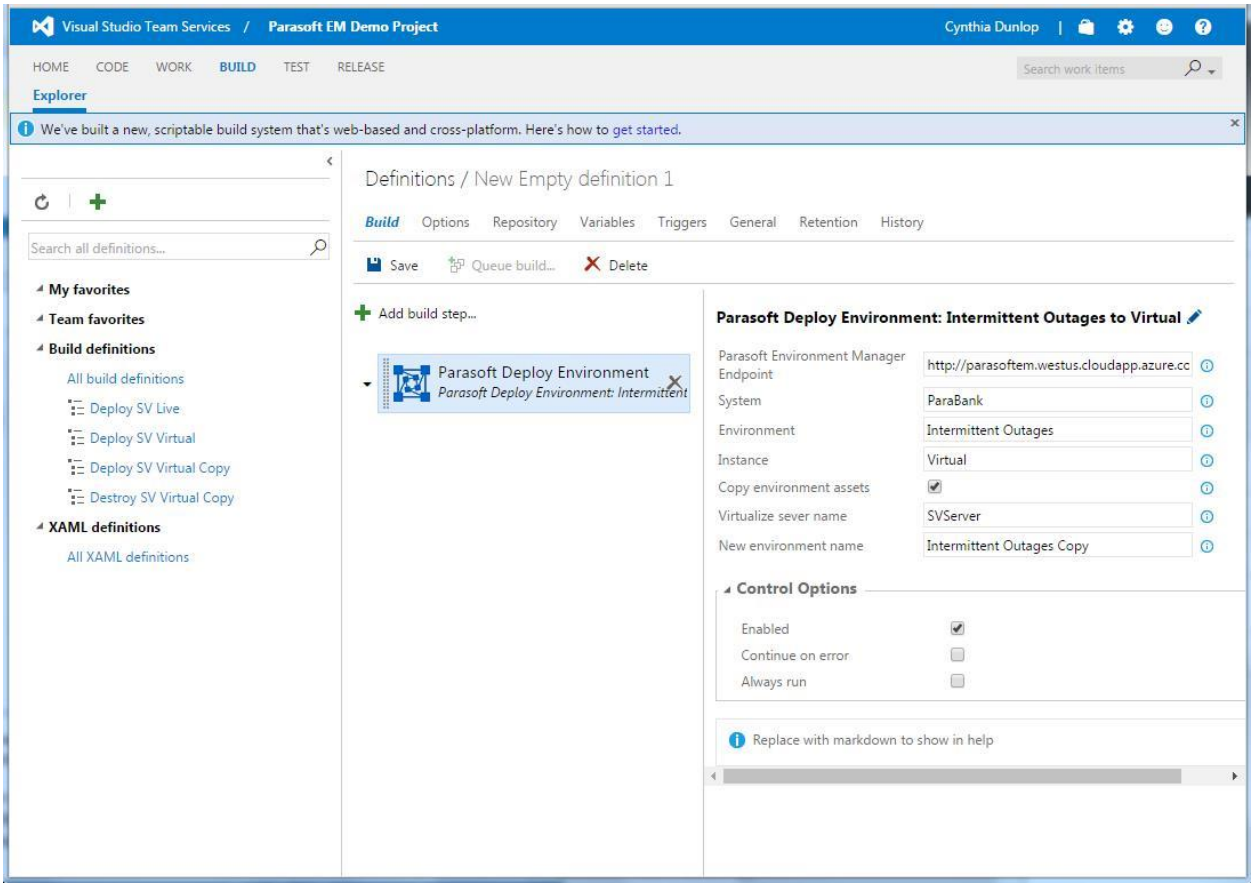
From this single environment, you can leverage simulation technology to configure the behavior of the system environments. It would be highly time-consuming to achieve this with the actual applications.



Any of the various "golden" environment instances can then be automatically deployed to your Azure VMs at any phase of the Microsoft VSTS pipeline. For example, the following screenshot shows how you could deploy a simulated test environment as a build step.



Then, just indicate what environment instances you want copied to what servers, and the entire process can be automated.



Whenever the simulated test environment is no longer needed (e.g., after automated tests are completed), it can be "destroyed" with a similar build definition.

There is no need to share test environments or resources across teams or test phases; the exact environment that's needed is instantly spun up whenever it's needed, then destroyed as soon as it has served its purpose. Both deploy and destroy definitions can be integrated at any point in the Microsoft VSTS pipeline.

Benefits

This process of using Parasoft Service Virtualization within the Microsoft environment offers a number of benefits to DevTest teams.

Facilitates Early and Continuous Testing

With instant and simultaneous access to realistic, flexible test environments, the team can execute a broader array of end-to-end tests. Testing can begin as soon as each user story is completed, and myriad user stories can be validated simultaneously during Continuous Testing—each with the very specific test environment required for the associated tests.

More Flexible than the Real Thing

Simulation allows you to provision a test environment with more flexibility than a staged test environment. With a physical staged test environment, it's difficult to manipulate the behavior or performance of many dependent applications—for example, mainframes or ERPs. You'd have to deal with hardware, configuration settings associated with the allocation of memory or the allocation of VM configuration resources, and so on. However, using simulation technologies such as service virtualization, you can easily replicate conditions that would be impossible to configure in a production environment or even a staged test environment. This lets you expand your test horizon to test extreme cases (e.g., for concurrency issues or outages, resiliency and failover, load balancer issues, and so on).

More Realistic than the Real Thing

In many cases, a simulated test environment is actually more realistic than a physical staged test environment. It's not uncommon to face hardware constraints in physical staged test environments because these environments often exhibit slower performance than you'd encounter in production. Using service virtualization, you can have greater control over how dependencies respond. You can ensure production-level response times, thereby increasing the fidelity of the test environment. Service virtualization can also compensate for errors or outages in staged test environments. For example, assume some of the system dependencies that are usually available for testing happen to be down at the time of test execution. With traffic automatically routed to a simulated version of those dependencies, your tests can proceed as if the dependencies were actually available.

Eliminate Risks Associated with Data Privacy and Security

Since you're in a simulation environment, you can automatically access data sets that are cleansed and masked. Moreover, you don't need to worry about security concerns—if someone is able to penetrate or attack a simulated asset, it doesn't matter…it's fake, and it will be destroyed as soon as the test is completed.

Ensure Consistency and Accuracy

Everyone is always on the same page, accessing the latest and greatest data, virtual assets, test assets, etc. Each time a test environment is provisioned (e.g., from the various stages of the VSTS pipeline), it is created from scratch using the master set of data, virtual assets, test artifacts, etc. stored in the repository. This makes it impossible for tests to be executed against "dirtied" test environments, using outdated test data, versus an unauthorized version of a virtual asset, etc.

About Parasoft

Parasoft researches and develops software solutions that help organizations deliver defect-free software efficiently. To combat the risk of software failure while accelerating the SDLC, Parasoft offers a Development Testing Platform and Continuous Testing Platform. Parasoft's enterprise and embedded development solutions are the industry's most comprehensive—including static analysis, unit testing, requirements traceability, coverage analysis, API testing, dev/test environment management, service virtualization and more. The majority of Fortune 500 companies rely on Parasoft in order to produce top-quality software consistently and efficiently as they pursue agile, lean, DevOps, compliance, and safety-critical development initiatives.

Contacting Parasoft

Headquarters

101 E. Huntington Drive, 2nd Floor
Monrovia, CA 91016
Toll Free: (888) 305-0041
Tel: (626) 305-0041
Email: info@parasoft.com

Global Offices

Visit www.parasoft.com/contact for contacting Parasoft in EMEAI, APAC, and LATAM.

Author Information

This paper was written by:

- Wayne Ariola (wayne.ariola@parasoft.com), Chief Strategy Officer at Parasoft
- Cynthia Dunlop (cynthia.dunlop@parasoft.com), Lead Technical Writer at Parasoft