TDI – Using Test Driven Development Methodologies for Infrastructure

For those who have had exposure to development environments, you are probably familiar with TDD, or Test Driven Development. If not, then I highly recommend that you dig into the topic as it may be one of the best tools to have in your toolkit: http://en.wikipedia.org/wiki/Test-driven_development

The concept is simple: Design a test for an upcoming feature. Run the test to fail. Implement the feature. Run the test with success. The key to the whole process is designing the test for the upcoming feature and allowing it to fail with the understanding as to why it has done so.

More often than not, we find ourselves designing and building infrastructure systems quickly and we frequently build before and during design. What begins as a proof of concept becomes a production system. The issue here is that we will build for success. Who wouldn't right?

To truly design an end-to-end system with the ability to fully understand it's moving parts and linkages we have to do the opposite. We have to design for failure. It's a difficult concept to understand for many.

A famous quote which is often attributed to IBM is "fail early, fail often". There are hundreds of references to it and similar phrases. It's probably as hard to trace the true origin as it is the urban legend of pop rocks and cola. Regardless, the idea is one we should latch on to. Nobody likes to fail, but if we don't fail during design, we will inevitably fail in production and we will not understand the reasons. The pressure will be on so it will not be the time to break out the design documents and reverse engineer.

The role of the Systems Architect is to be able to design with the future and current state in mind. As designers and implementers we need to expand our understanding to not just the pie in the sky, but the inner workings. Much like Douglas Adams' character Dirk Gently the holistic detective, we should understand the "fundamental interconnectedness of all things".

Where the application development teams have TDD, we as infrastructure designers have to look to TDI: Test Driven Infrastructure.

Designing for failure is one thing, but being able to test for failure is the holy grail of successful implementation. It is as challenging in systems and infrastructure design as it is in application development. There are limited tools available to do end-to-end testing of systems like VDI, Virtual Server environments, cloud orchestration, auto deploy, Business Continuity and many other environment frameworks.

What we need to do is to leverage the tools available to us such as management and monitoring tools. Sprinkle in some scripting and automation, add a dash of log aggregation and we have ourselves the recipe for a nice hybrid infrastructure test method. The advantage to growing your own testing infrastructure is that you have effectively created your long term production monitoring

tool along the way. It is truly a win-win.

In a previous article on Installation versus Implementation I wrote about how we can install a product very quickly, but the true value comes in designing for growth, performance and ultimately for managing failure.

Look to what you have available for tools to bring into your TDI design. I make use of SCOM, vCenter Operations Manager, SolarWinds Orion, PowerCLI, PowerShell and some other home grown application tools. All of these help me work to understand the linkages between my systems. During design and build the plan we will include purposeful failure and recovery to best understand how each system will act in production. Not just in a pure successful implementation, but we will gain an understanding of how and why it will fail, and what the recovery steps are.

Raise your glasses to failure and celebrate that we found it before it found us!