

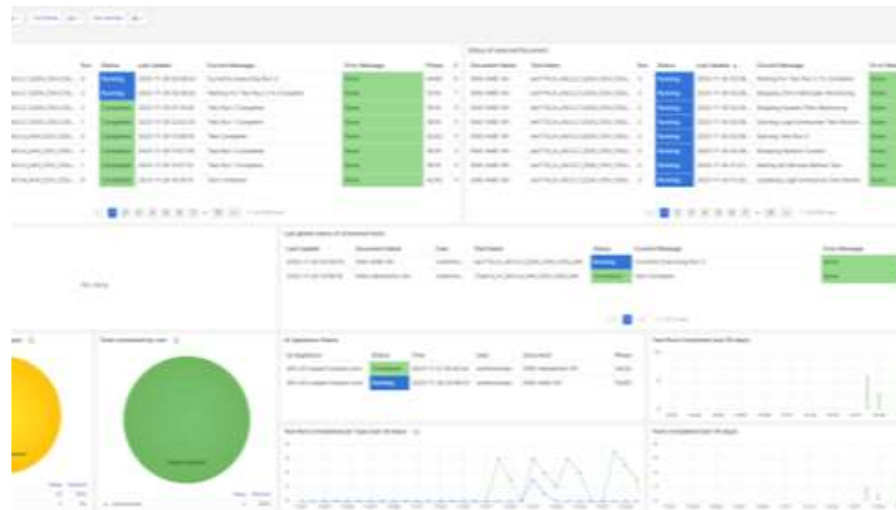
Solutions and Performance - EUC > Pages > EUC Blog

Nutanix Solutions Engineering EUC 2023 Hackathon



James Kindon

Published Dec 4, 2023 (Edited Dec 4, 2023) • 8 min read



During November of 2023, we engaged in an internal hackathon project to improve our automation around performance testing and address a growing list of needs for increasing efficiency and capability.

This hackathon was a chance to address outstanding requests that had been piling up after many different tests and an ever-changing set of requirements. We needed to improve on an already heavily automated process which of course, had grown over

time, resulting in code that needed revising to allow us the ability to scale and adapt as our requirements change.

We use a combination of tooling to baseline and compare our tests, ultimately resulting in collateral that is released for both internal and public consumption. To achieve these results, we use the following tools:

- GitHub (for code management)
- PowerShell (for execution of all components)
- [Grafana](#) (for modelling data)
- [InfluxDB](#) (for storing time-series data)
- [Telegraf](#) (for sending data to Influx)
- [Login Enterprise](#) (to execute the load test)

Our automation processes allow us to prepare a cluster for testing, be it AHV or ESXi, execute the testing itself, capture and aggregate metrics, and even create custom reports based on Grafana comparisons.

What We Did

We revisited every aspect of our automation and refactored it with the following logic:

- Where possible, control and configuration options were defined in JSON format and consumed by control scripts.
- Failure and *fail-fast* logic:
 - Failure is expected. *Unhandled* failure scenarios are **not** expected.
 - Think about failure upfront and build validation logic into any component we think has a chance of failure.
 - If we fail fast, we can fix it fast. Efficiency is king.
- There is no such thing as too much output. Log everything and log it out loud.
- Clean up redundant, repeated, or irrelevant code. Removal of unused code helped to keep the framework lean.

With the above logic, we refactored our existing code into a PowerShell module. **Nutanix.EUC** was born.

With the restructuring of the code complete, we were able to enhance and build. Immediately we were able to see the fruits of our labour, as tasks in our list flagged as *"investigate and scope effort"* suddenly became easy *enhancements*. Some key examples:

1. An enhancement to investigate the ability to ***monitor, collect, upload and analyze data sets from two clusters at the same time*** (think a workload cluster hosting 1000 virtual machines, and a corresponding dedicated cluster hosting **Nutanix Files**) easily moved from *"investigate"* to *"implemented"* because of our modular approach. Win.
2. An enhancement to implement **in-guest monitoring of specific infrastructure servers** with unique metrics became a very simple additional module that we could turn on and off with a simple true/false value, a list of servers, and a start/stop logic. Win again.

These are just a couple of basic examples of why a restructure was important.

Throughout the hackathon, we completed:

- 15 tasks associated with infrastructure uplift, upgrades, and configuration changes.
- 8 tasks for dashboard and reporting improvements.
- 30 tasks around automation improvements, enhancements, and critical changes.
- Github tells us there were 518 files changed with 42,047 additions and 375 deletions.

What Nutanix.EUC Does

Our primary focus is a combination of Citrix Virtual Apps and Desktops (including DaaS) and VMware Horizon testing using Login Enterprise. We regularly test across multiple Nutanix Clusters using several different hypervisors and configuration types. We test everything from changes in Provisioning technologies to Windows release versions, through the different profile management solutions and their impacts on performance. We manage and test against many different cluster configurations.

We aren't limited to these tests, we can (and do) test different technologies and different performance-impacting scenarios. For example, [Parallels RAS has been baselined and tested](#) using our existing tooling. We also previously performed [large-scale testing on Citrix Session Recording](#) which required disparate metric collection and analysis in the old framework. With the new frameworks, our data will look much more consistent and present in a much more consumable fashion.

It is very simple to extend our logic and capability to new platforms and solutions. If the solution can be automated, it can be pulled into our structures.

This module currently allows us to execute tasks such as:

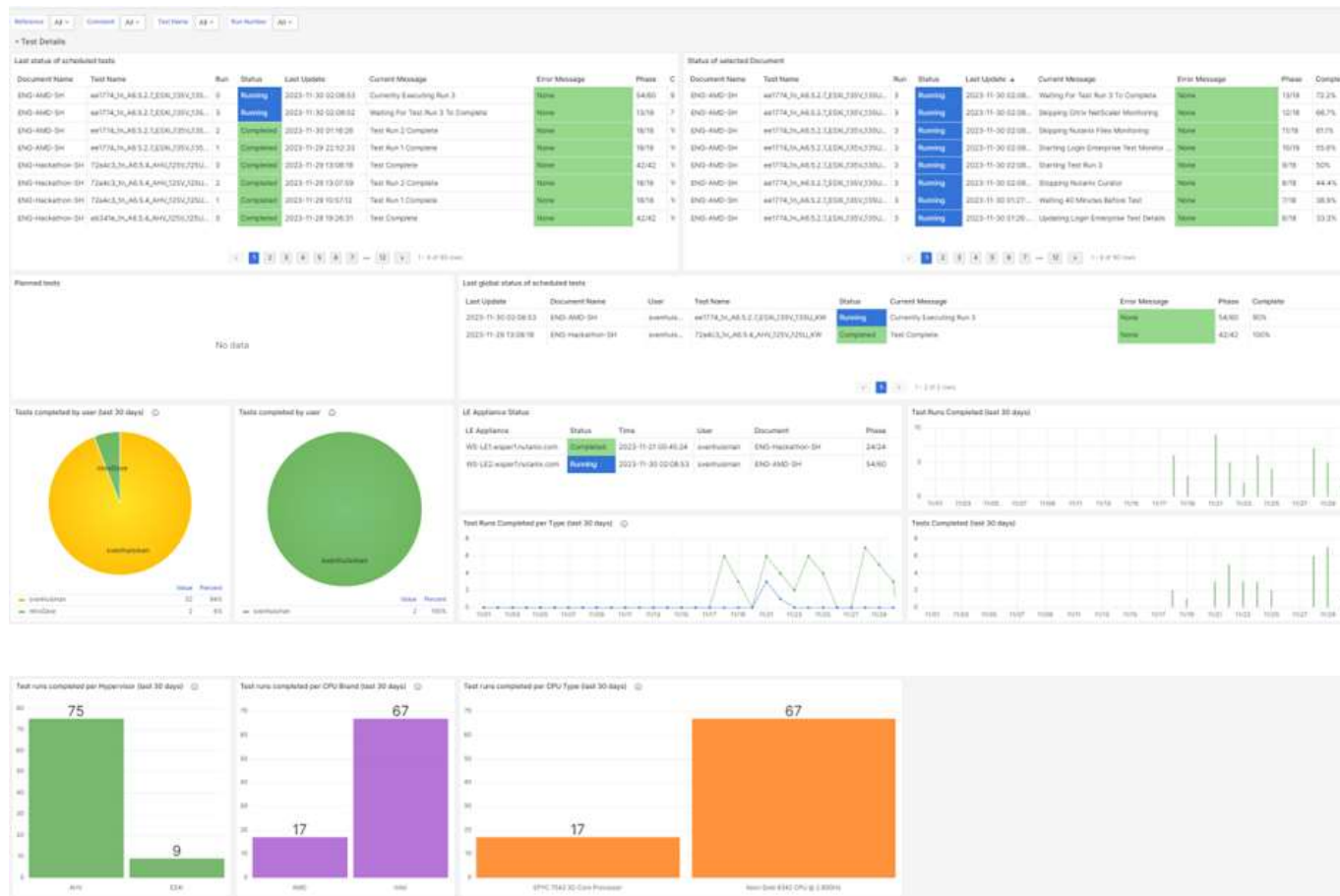
- Configuring and managing **Active Directory** accounts.
- Configuring and managing a **Login Enterprise Appliance** and associated test configurations.
- Configuring and monitoring **Nutanix AOS** and **Nutanix Files** metrics via API calls.
- Configuring and managing **Citrix Virtual Apps and Desktops** and **Citrix DaaS** deployments via PowerShell Snapins.
- Configuring and managing **VMWare Horizon** deployments via PowerShell Modules.
- Configuring and monitoring of Windows Infrastructure via **Telegraf**.
- Managing data manipulation for ingestion into **InfluxDB**.
- Download of images from **Grafana** dashboards created from **InfluxDB** data.
- Integration with **Slack** for status reporting.
- Integration with **Grafana** for real-time test monitoring and status, along with real-time metric reporting.

Visibility and Tracking

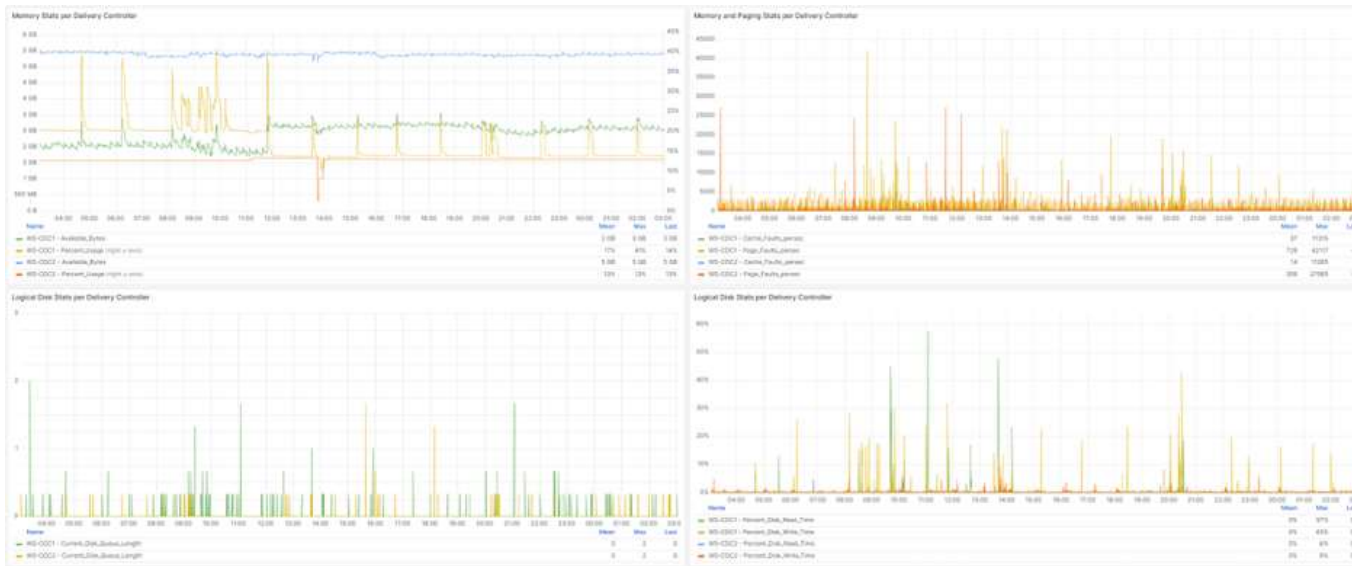
When we manage multiple tests, multiple clusters, multiple documents, and multiple humans, having visibility into what is going on and where becomes important. When we have a global team where we can hand off work in a follow-the-sun style setup, this visibility becomes *critical*.

One of the coolest enhancements that came out of our hackathon was a Grafana-fronted, InfluxDB-backed testing status dashboard.

This dashboard allows our team to view the exact status of any test at any time, in real-time. We can see what is running, what has finished, what is scheduled (some of our tests have multiple runs), where everything is running and who is running it.



We can gain insight into which clusters are busy and by which tests, or which ones are idle and what their current configurations are.



Learnings

Throughout the process, we learned some good lessons:

- Each person in a team brings different skills and perspectives, code-based or not. Blending them can result in a very cool set of outcomes. Within our team, some have an incredible ability to manipulate data and present information, while others can figure out the fine-grained code requirements to get that data into a place where it can be presented nicely.
- Sometimes the solution to what appears a complex problem is far more obvious than one, two, or three sets of eyes may think. Having multiple viewpoints and avoiding a rabbit hole early can pay dividends.
- Keeping on target/focused often needs someone who is not staring at the code to bring perspective.
- Being a global team, A focused few weeks and appropriate communication, updates and working sessions for handover, meant we could be extremely efficient and productive, without taxing any one person.
- Don't try and solve everything at once, be willing to re-prioritize and adapt quickly. Doing so means we can tick more boxes and gain more from our time.

What is next?

The EUC world never sits still. New technologies and solutions are constantly being introduced, each of which will likely need to be validated and baselined on Nutanix infrastructure at some stage.


Our goal is to continue the growth of the module where it makes sense and where we see repeatable tasks. We want to reduce dependencies (where possible) on limiting components such as PowerShell Snapins that tie us to specific PowerShell versions which reduces efficiency and introduces complexity.

Once we can move, for example, Citrix automation to an API-based approach, we can say goodbye to PowerShell 5.1 and operate on PowerShell 7. This might seem insignificant as you read this but catering for two different versions with very different functionalities and capabilities is a significant tax on development time and testing.

We might even have different load testing engines that we can bring into the fold. Who knows. Bring on Solutions Engineering EUC Hackathon 2024.

Topics

[performance testing](#)[Hackathon Project](#)[automation processes](#)

Send feedback 

Related content