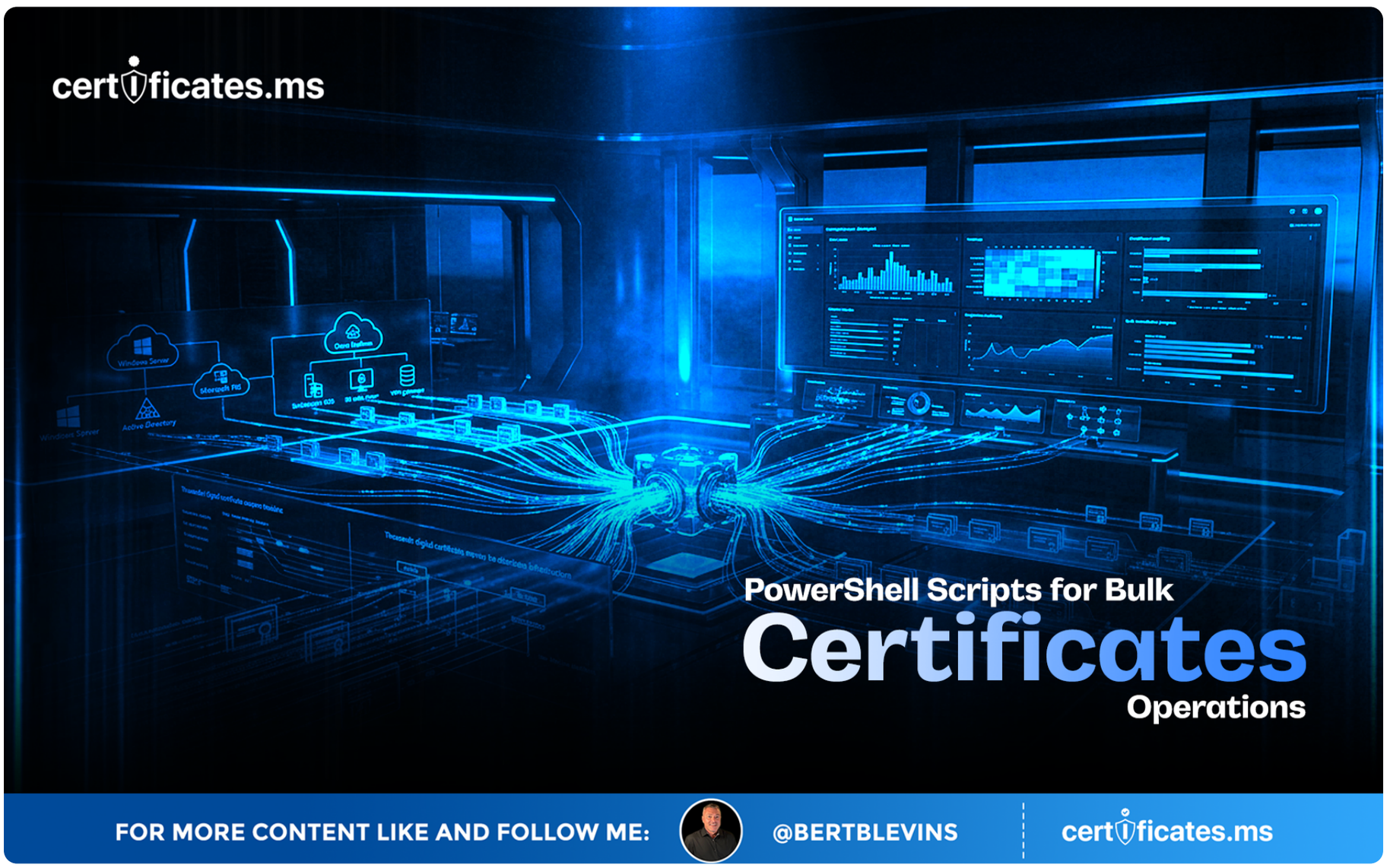
PowerShell Scripts for Bulk Certificate Operations



Windows-centric environments often accumulate certificates across dozens or hundreds of servers, and reaching into each machine's certificate store individually to check, renew, or clean up certificates does not scale past a handful of systems. PowerShell offers a genuinely powerful toolkit for handling these operations in bulk, and this article covers the practical patterns for using it effectively.

Inventorying Certificates Across a Fleet

PowerShell's PKI module allows querying the local certificate store directly, and combined with PowerShell remoting, the same query can be run across an entire fleet of servers simultaneously, returning a consolidated inventory of every certificate installed, its expiration date, its issuer, and which store it lives in. Building a scheduled script that performs this inventory regularly and exports the results to a central location, whether a simple spreadsheet or a proper database, is often the first meaningful step an organization takes toward genuine certificate visibility, well before adopting a dedicated CLM platform.

Bulk Expiration Checking and Alerting

Once an inventory script exists, extending it to flag certificates approaching expiration within a defined threshold is a natural next step, and this kind of lightweight, custom alerting script has saved plenty of organizations from an entirely preventable outage. A script that runs daily, checks every certificate across the fleet, and sends an alert for anything within a defined number of days of expiring closes a large share of the gap between having no monitoring at all and adopting a full commercial platform, at a fraction of the implementation cost and time.

Bulk Certificate Requests Through PowerShell

PowerShell's certificate request cmdlets can generate CSRs and submit requests to an internal Active Directory Certificate Services CA programmatically, making it practical to script bulk certificate requests for a large batch of servers or services at once rather than requesting each certificate individually through a manual console workflow. This is particularly useful during a migration or a large-scale certificate refresh, where dozens or hundreds of certificates need to be reissued within a compressed timeframe.

Bulk Installation and Store Management

Installing certificates into the correct certificate store, whether the local machine store, the current user store, or a specific application's dedicated store, is a common source of manual error when done one server at a time. A well-built PowerShell script can install a certificate into the correct store across every target server consistently, verify the installation succeeded, and log the result, removing the variability that comes from different administrators manually clicking through slightly different steps on different machines.

Cleaning Up Expired and Orphaned Certificates

Over time, certificate stores across a fleet tend to accumulate expired certificates, duplicate entries, and orphaned certificates left behind by decommissioned applications, none of which are actively dangerous but all of which clutter the store and make genuine certificate management harder. A scheduled PowerShell cleanup script that identifies and, after appropriate review, removes clearly expired and unused certificates keeps the certificate store itself a more reliable source of truth for whatever inventory and monitoring processes depend on it.

Integrating PowerShell Scripts Into Broader Automation

While custom PowerShell scripts are genuinely useful for bulk certificate operations, especially in smaller or Windows-heavy environments, organizations should be honest about when a growing collection of custom scripts has outgrown ad hoc scripting and would be better served by a dedicated CLM platform or a more structured automation framework like the Ansible and Terraform patterns discussed elsewhere in this series. PowerShell scripting and dedicated CLM tooling are not mutually exclusive; many organizations use PowerShell for Windows-specific bulk operations while relying on broader platforms for cross-platform certificate governance.

PowerShell Automation for AI-Adjacent Windows Infrastructure

Windows Server environments increasingly host AI-related infrastructure, including on-premises model-serving components and the Windows-based automation supporting AI-driven business processes, and the same bulk PowerShell certificate management patterns described above apply directly to these systems. Extending existing PowerShell certificate inventory and renewal scripts to cover new AI-related Windows infrastructure, rather than standing up an entirely separate process for it, keeps certificate governance consistent across an organization's full Windows estate as that estate grows to include AI workloads.

The Countdown Is Already Running: 200 Days, 100 Days, 47 Days

Every certificate conversation in 2026 eventually arrives at the same clock, and it is worth closing on it here. The CA/Browser Forum's Ballot SC-081v3 is not a proposal under discussion; it is an approved, already-in-motion schedule. Maximum public TLS certificate lifetimes fall from 398 days to 200 days on March 15, 2026. They fall again to 100 days on March 15, 2027. By March 15, 2029, they drop to just 47 days, with domain validation itself needing to be re-proven roughly every 10 days.

01

Translate that into operational terms and the picture gets stark quickly. An organization currently renewing certificates a few times a year will be handling renewal events on the order of every couple of weeks by the end of this countdown, across every endpoint it operates. Manual tracking, calendar reminders, and a spreadsheet somebody checks once a month will not survive contact with that cadence. What has always been an occasional chore is becoming a continuous, automated operation, whether an organization plans for it or not.

02

PowerShell scripts built for occasional bulk operations today will need to run considerably more often as the schedule below compresses renewal cycles toward every 47 days, making it worth investing in genuinely robust, well-tested automation now rather than scripts that were only ever meant to handle an occasional manual bulk task.

03

The 200-day, 100-day, and 47-day milestones are not distant hypotheticals; the first has already arrived. Organizations that build the automation loop now, generating keys, vaulting them securely, brokering issuance across Certificate Authorities through APIs, and rebinding certificates to live endpoints without manual intervention, will meet each deadline without disruption. Organizations that wait will be rebuilding their certificate operations under deadline pressure, with far less room for error and far less time to get it right. The countdown is the call to action. The only real decision left is whether to automate on your own schedule, or on the CA/Browser Forum's.