MVA Microsoft Virtual Academy

PowerShell DSC Einsteigerkurs

Microsoft

Haiko Hertes
Head of IT
Automotive Process Institute GmbH

Jan-Henrik Damaschke
Senior Consultant Security and Cloud
Bright Skies GmbH

MVA Microsoft Virtual Academy

01 | Einleitung

Microsoft

Haiko Hertes
Head of IT
Automotive Process Institute GmbH

Jan-Henrik Damaschke
Senior Consultant Security and Cloud
Bright Skies GmbH

Haiko Hertes | @HHertes

- Head of IT, Automotive Process Institute
  - Verantwortlich für eine gemischte Microsoft- und Linux-Umgebung in einem dynamischen Unternehmen der Automobil-Branche
- Mehr als 4 Jahre Erfahrung als MCT und IT-Trainer
  - Vermittlung von praxisnahem Wissen rund um Windows Server, System Center und Virtualisierung
- Seit April 2016 als MVP ausgezeichnet
- Blog: www.hertes.net | YouTube: Kanal von Haiko Hertes
- Weitere Kanäle: https://about.me/haiko.hertes

Jan-Henrik Damaschke | @jandamaschke

- Senior Consultant Security and Cloud, Bright Skies GmbH
  - Verantwortlich für (Hybrid) Cloud Security
- Viele Jahre Erfahrung mit Microsoft Produkten
  - Mehrere MCSE's
  - CCNA Security
- Seit Juli 2016 als MVP ausgezeichnet
  - Unterkategorien Enterprise Security und PowerShell
- Blog: blog.jdamaschke.com

Kursthemen

PowerShell – Desired State Configuration

01 | Einleitung u. Kursvorstellung

02 | Aufbau einer Configuration u. OnBoard-Mittel

03 | Enacting einer Configuration via Push

04 | Weitere Ressourcen aus der Gallery

Zielgruppe des Kurses

- IT-Administratoren
- Skripter
- Alle PowerShell-Fortgeschrittenen

Hilfreich ist, wenn bereits erste Erfahrungen mit der Windows PowerShell vorhanden sind.

Vorbereitendes Material

- Bisher wenig mit der PowerShell gearbeitet?
  - MVA-Kurs PowerShell-Einführung von Sebastian Klenk: https://mva.microsoft.com/de-de/training-courses/einfhrung-in-windows-powershell-40-14344
  - Meine YouTube-Playlist zur PowerShell https://www.youtube.com/playlist?list=PLPK8RW8p4Ok91JXn8f78ko_i3lfReYa7k

Hilfreiche Tools und PS-Bestandteile

- Für jegliche Arbeit mit der PowerShell:
  - Integrated Scripting Environment (ISE)
    - Ist seit PowerShell 2.0 bzw. Windows 7/Server 2008 R2 von Anfang an dabei
    - IntelliSense
    - Syntaxhighlighting
    - Zoom
    - …
  - -WhatIf-Switch
    - Cmdlets mit Änderungswirkung haben idR einen –whatIf Switch als Parameter
    - Gibt aus, was wäre, wenn man ihn weglassen würde

Hilfreiche Tools und PS-Bestandteile

- Für PowerShell DSC im Speziellen:
  - MSDN - https://msdn.microsoft.com/en-us/powershell/dsc/overview
  - GitHub - https://github.com/PowerShell/

MVA

Join the MVA Community!

- Microsoft Virtual Academy
  - Kostenlose Online-Trainings für Entwickler und IT Professionals
  - Mehr als 3.3 Millionen registrierte Nutzer
  - Regelmäßig neue Trainings zu aktuellen Microsoft Produkten und Technologien
  - Live- und on-demand-Kurse

MVA Microsoft Virtual Academy

PowerShell DSC Einführung

Microsoft

Zweck der PowerShell DSC

- Desired State Configuration
  - „Gewünschter Zustand Konfiguration“
- Beschreiben der Zielkonfiguration eines oder mehrerer Systeme
- Statt der „sturen“ Abarbeitung imperativer Befehle: Deklarative Sprache
- Vergleichbar mit Puppet, Chef, Ansible, …
- Muss nicht zwingend auch die Konfiguration herstellen, auch „nur“ prüfen ist möglich

Zweck der PowerShell DSC

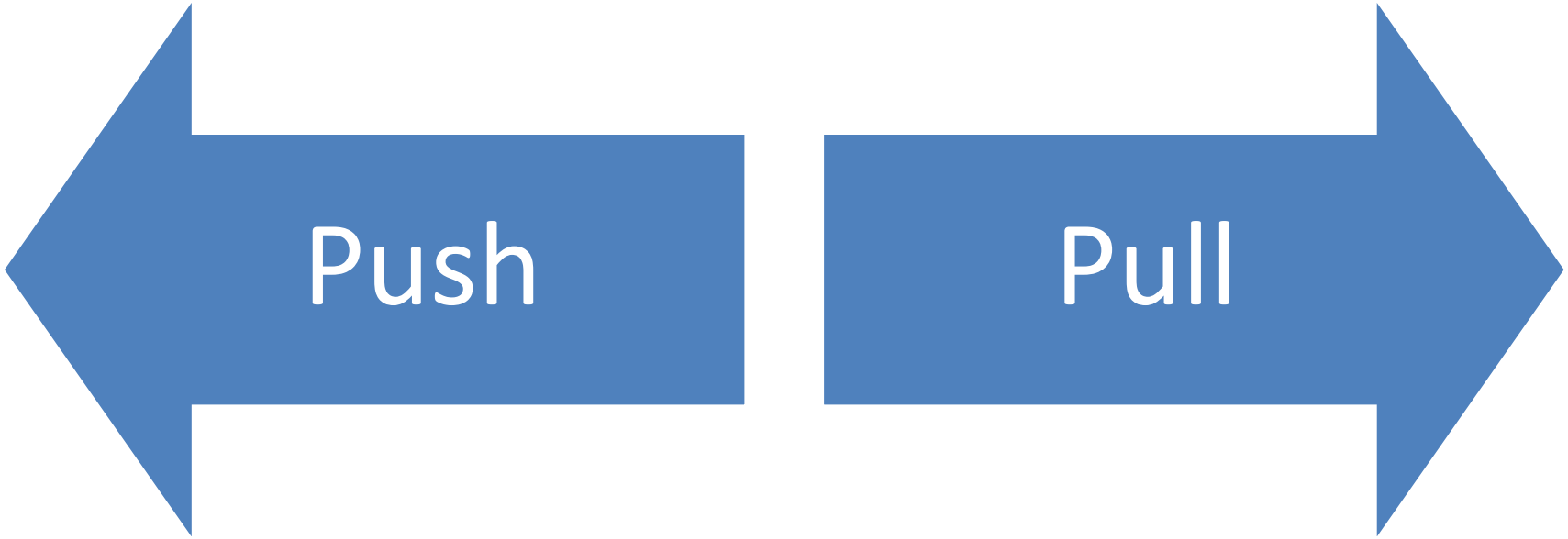
- Bei Bedarf können mehrere Server parallel konfiguriert werden
- Benötigt min. WMF / PowerShell 4.0
- Seit Mitte 2016 auch für Linux verfügbar!
- Auf den Zielsystemen setzt ein LCM (Local Configuration Manager) die Konfigurationen um

Zweck der PowerShell DSC

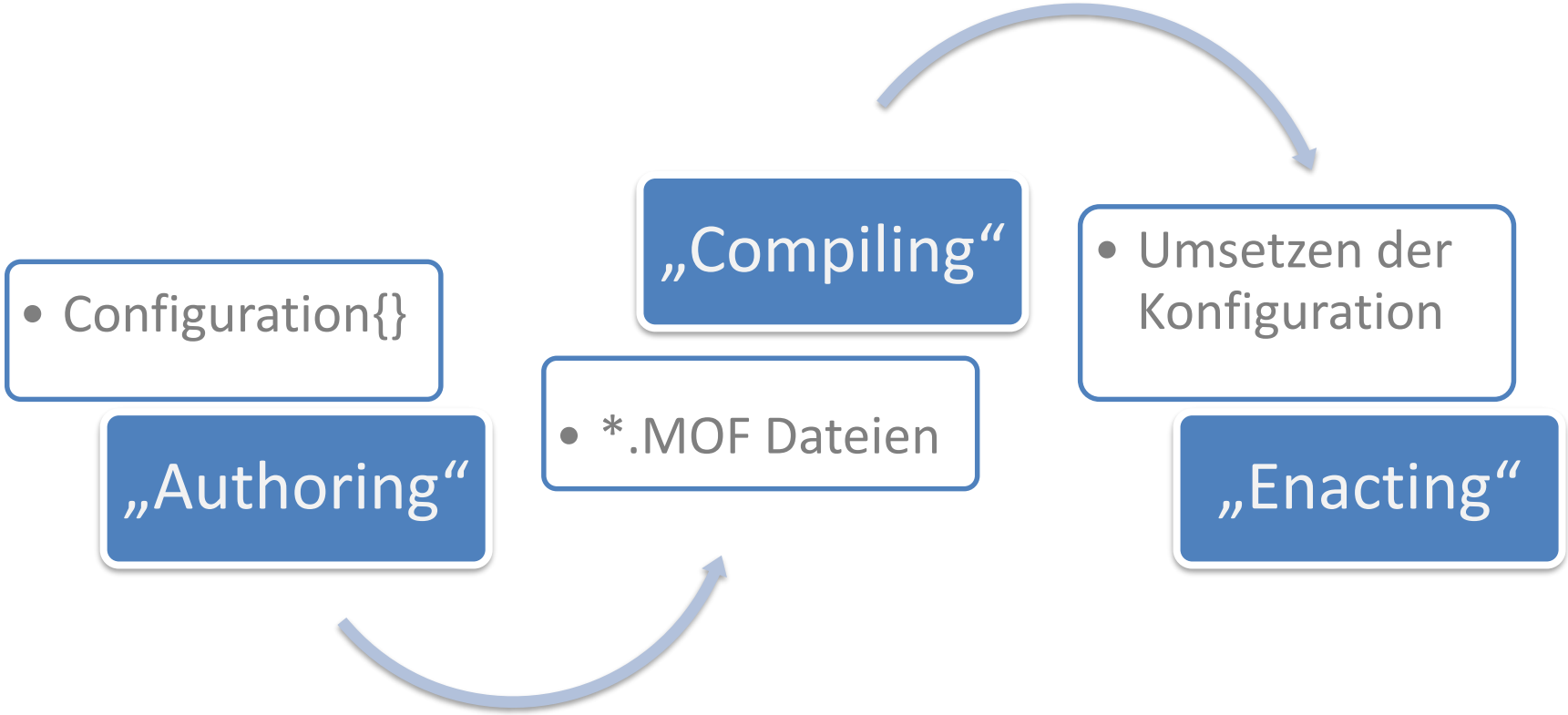
- Automatisierung großer Umgebungen
- Vereinfachen komplexer Setups
- Definieren von Standards
- Sicherstellen von nötigen Konfigurationen (z.B. Sicherheit, Compliance, …)

Architektur von PowerShell DSC

- 2 Möglichkeiten für das Umsetzen („Enacting“) der Konfiguration:



Der DSC-Workflow:



Authoring-Phase

- Schreiben des bzw. der Configuration{}-Files (i.d.R. als *.ps1 Datei)
- Eigene DSC-Syntax mit vielen Elementen der „normalen“ PowerShell

Compiling-Phase

- Ausführen des Configuration-Blockes (z.B. mit F8 in der ISE)
  - Wichtig ist, dass der Block nicht nur deklariert, sondern auch aufgerufen wird (wie bei einer function{})
- Dabei entstehen *.MOF-Dateien
  - Diese enthalten die Konfiguration in einer anderen Syntax
  - Sprache bereits aus CIM bekannt, von „Distributed Management Task Force“ entwickelt

Enacting-Phase

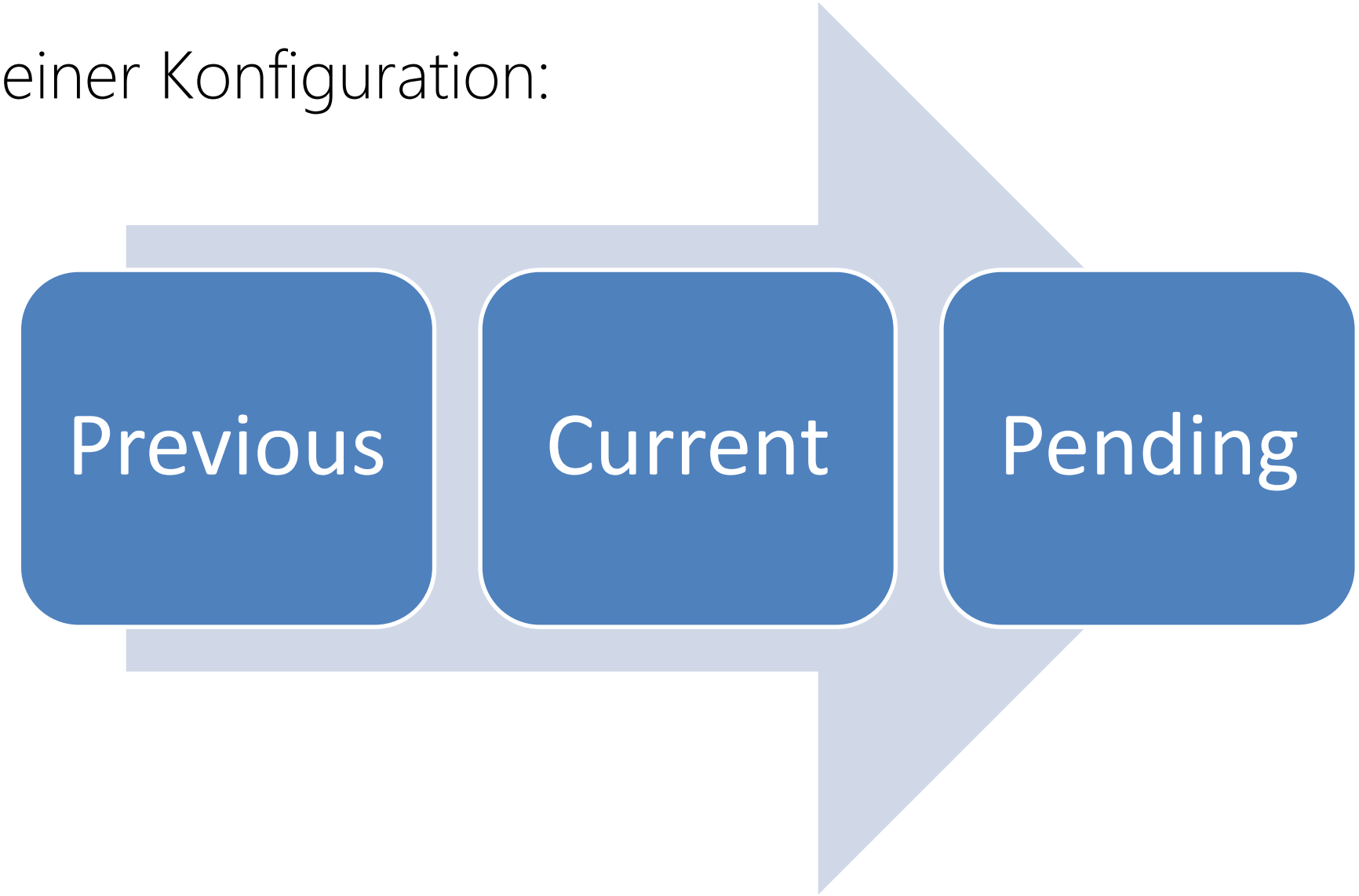
- Für das Umsetzen der Konfiguration(en) gibt es zwei Möglichkeiten:
  - Push
    - Zielserver bekommen die Konfiguration manuell zugewiesen
  - Pull
    - LCM auf den Zielservern laden die Konfiguration in regelmäßigen Abständen von einem Pull-Server

MOF-Dateien

- Managed Object Format
- Standard-Format der Distributed Management Task Force (DMTF) für CIM (Common Interface Model)

Current, Previous, Pending Configuration

- 3 Stati einer Konfiguration:



Versionen

- PowerShell DSC steht ab WMF / PS 4.0 zur Verfügung
  - Vieles geht schon, aber z.T. nur aufwändig
- Mit PowerShell 5.0 kamen viele Neuerungen hinzu
  - Benutzung wurde auch wesentlich einfacher
  - Einige Verbesserungen u. Neuerungen:
    - Verschlüsselte MOF-Dateien
    - Partial Configurations
    - Bessere LCM Verwaltung
    - Neue Debugging-Möglichkeiten

Versionen

- Mit PowerShell 5.1 kommen einige weitere Verbesserungen hinzu:
  - Weitere Debugging-Möglichkeiten
  - Mehr Invest in die Sicherheit von DSC

Wie geht es weiter?

- Modul 2: Aufbau einer Configuration
  - Wie ist eine Configuration aufgebaut
  - Welche Möglichkeiten bietet PowerShell 5 bereits ab Werk für DSC


Microsoft