Oracle-Infrastrukturen mit Chef verwalten

Bernd Patolla, In&Out

Für die Automatisierung von Oracle-Datenbank-Infrastrukturen und -Datenbanken existieren diverse Tools. In diesem Artikel wird die Automatisierung einer Oracle-SIHA-Infrastruktur vorgestellt. Der Schwerpunkt liegt dabei auch auf der Integration in die vorhandenen Umsysteme wie ***Oracle Cloud Control, Backup-Tools wie IBM Spectrum Protect und Nagios / CheckMK Monitoring***.

Einführung in Chef

Chef ist eigentlich ein Unternehmen, das 2008 gegründet wurde und das Produkt Chef Infra entwickelt hat. Heute stellt sich das Chef-Universum wie in *Abbildung 1* dar.

Für die Automatisierung von Oracle-Infrastrukturen wird die Chef Infrastructure Automation verwendet. Diese Infrastruktur besteht aus den drei Komponenten *(siehe Abbildung 2)*.

- Chef Workstation
- Chef Server
- Clients

Auf der Workstation werden die Cookbooks mithilfe des Tools kitchen entwickelt, getestet und auf den Chef-Server hochgeladen. Auch wird auf der Workstation die Konfiguration der Server-Infrastruktur mit den Tools knife und chef verwaltet. Empfehlenswert ist das Ablegen der Konfigurations-Informationen in einem separaten Software-Repository wie zum Beispiel GIT. So kann die Konfiguration der Systeme historisiert werden. Der Chef-Server speichert die Cookbooks und die Konfigurations-Informationen der Server-Landschaft / der Clients. Die Clients holen sich die aktuelle Konfiguration vom Chef-Server und anschließend die notwendigen (und definierten) Cookbooks. Diese Cookbooks werden dann in einem Chef-Lauf ausgeführt. Nach einem Chef-Lauf meldet der Client sowohl den Status als auch seine Konfigurations-Informationen zurück an den Server.

Chef Client / Node

Ein Chef-Node repräsentiert einen Server, eine virtuelle Maschine, einen Container oder eine „Cloud-Maschine“ die unter der Kontrolle von Chef ist. Auf dem Node ist der Chef-Client installiert. Je nach Konfiguration läuft dieser in gewissen Zeitabständen und führt die definierten Cookbooks mitsamt der Konfiguration aus. Die aktuelle Konfiguration des Nodes wird vor dem Ausführen der Cookbooks mit dem Tool ohai erfasst und am Schluss an den Chef-Server geschickt. Die Konfigura-

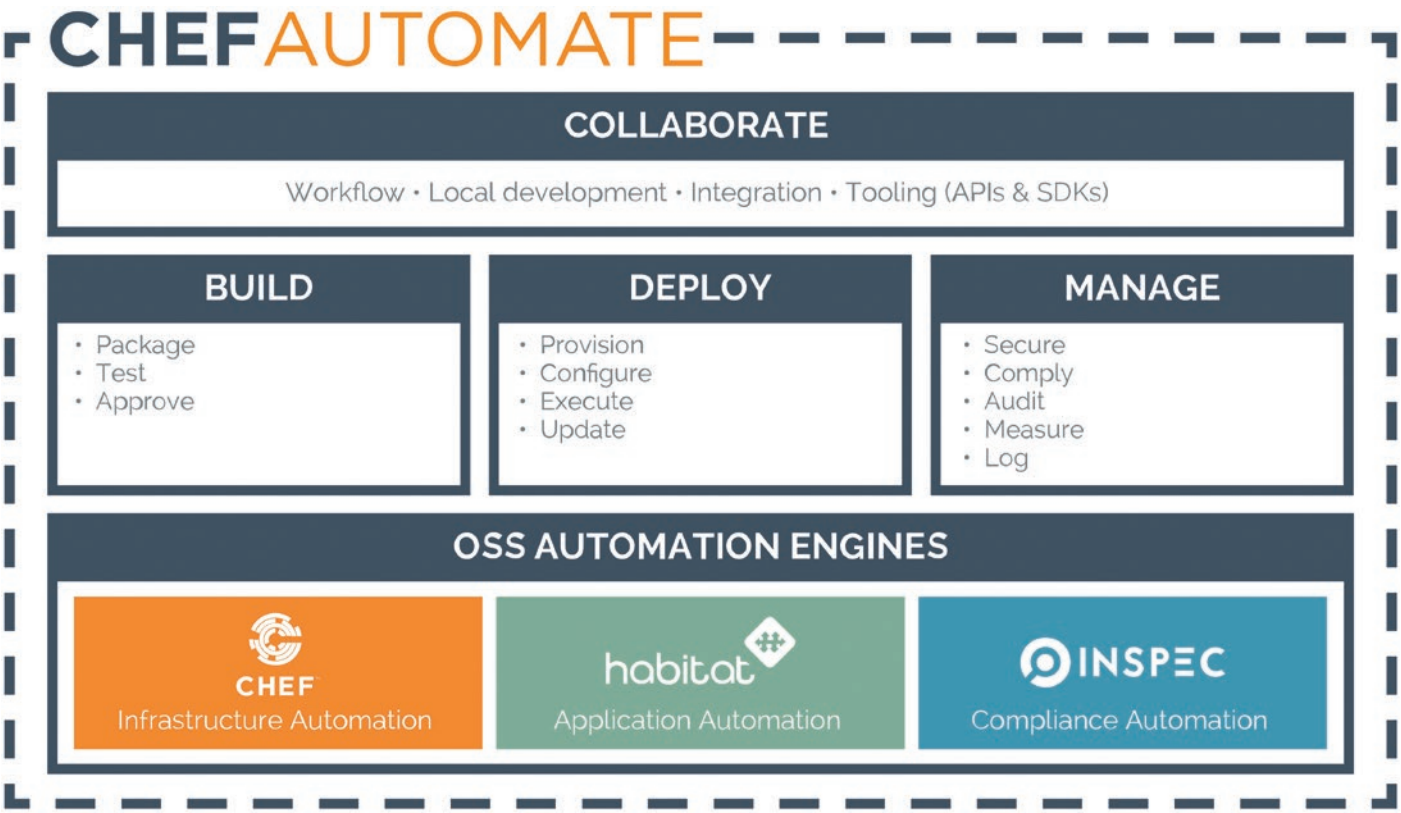

Abbildung 1: Chef Automate Universum (Quelle: https://docs.chef.io/platform_overview)

tion eines „node" besteht aus verschiedenen Teilen:

- Eine Liste von abzuarbeitenden und versionierten „cookbooks" (run-list)
- Zuweisung einer „role" zu dem Node
- Zuweisung eines „environment" zu dem Node
- Definition von „attributes"

Cookbook

Ein Cookbook beschreibt, wie etwas ausgeführt wird und welcher Endzustand vorherrschen soll. Cookbooks besitzen neben dem Code Metadaten wie eine Versions-Nummer und ein Readme. In den Metadaten werden Abhängigkeiten von anderen Cookbooks, teilweise bis auf die Versions-Nummer, beschrieben. Daneben besitzt ein Cookbook verschiedene Komponenten:

- Recipes
- Attributes
- File Distribution
- Templates
- Ressourcen und/oder Libraries

Recipes

Ein Recipe ist grundsätzlich in der Sprache Ruby geschrieben. Eine Beschreibung von Ruby ist unter *https://www.ruby-lang.org/en/documentation/* zu finden. Mit der neuesten Chef-Version gibt es inzwischen die Möglichkeit, YAML-Code zu schreiben. Ein Recipe besteht aus einer Reihenfolge von Ressourcen und kann andere Recipes aufrufen. Ein Recipe definiert so punktuell den gewünschten Zustand des Systems. Recipes werden im Subverzeichnis „recipes" eines Cookbook gespeichert.

File Distribution

Ein File in einem Cookbook kann durch eine Ressource auf den Node kopiert werden. Die in der Ressource definierten Files werden in dem Verzeichnis „files" vorgehalten.

Abbildung 2: Chef-Infrastruktur-Übersicht (Quelle: https://docs.chef.io/platform_overview)

Templates

Templates grenzen sich gegenüber Files durch Platzhalter ab, die während der Laufzeit eines Cookbook mit dynamischen Werten gefüllt werden. In Templates kann auch Ruby-Code zum Erzeugen des entsprechenden Files auf dem Node benutzt werden.

Ressourcen / Libraries

Mit Ressourcen oder Library-Funktionen kann die Funktion von Chef erweitert werden. In einer Ressource können andere Chef-Ressourcen aufgerufen oder Ruby-Code verwendet werden.

Attributes

Attribute können in verschiedenen Stufen definiert werden:

- auf dem Node, um diesen zu beschreiben (z.B. Hostname, IP-Adresse, OS, ...),
- über das CLI oder in einem JSON-File beim Aufruf vom Chef-Client,
- in einem Cookbook,
- in einer Rolle,
- in einer Umgebung / Environment und
- in einem Policy-File.

Attribute sind aufsteigend priorisiert:

- Default: Ein Default-Attribut wird bei jedem Chef-Lauf neu gesetzt und hat die niedrigste Priorität in der Hierarchie.
- Force_default: Ein als Force_default definiertes Attribut in einem Cookbook hat eine höhere Priorität als ein Default-Attribut einer Rolle oder einer Umgebung.
- Normal: Ein normales Attribut bleibt in der Node-Konfiguration konsistent gespeichert.
- Override: Ein Override-Attribut wird beim Starten eines Chef-Laufs automatisch resettet. Es wird oft in einem Recipe oder einem Attribute File definiert. Override-Attribute sollten nur dann benutzt werden, wenn unbedingt notwendig.
- Force_Override: Wird benutzt, um sicherzustellen, dass ein Override-Attribut übersteuert wird.
- Automatic: Automatic-Attribute werden durch das Tool ohai gesetzt. Ein automatisches Attribut kann nicht geändert werden.

Attribute können an verschiedenen Stellen definiert werden:

- in einem JSON-File auf dem Node; es wird durch die Option „-j“ dem Chef-Client bekannt gegeben,
- auf dem Node durch das Tool ohai, das vor jedem Chef-Client-Lauf ausgeführt wird,
- in einem Attribut-File in einem Cookbook,
- in einem Recipe eines Cookbook,
- in Environment-Definitionen,
- iRollen-Definitionen oder
- in einem Policy-File.

Weitere Informationen zu Attributen und deren Hierarchie können der Webseite *https://docs.chef.io/attributes/* entnommen werden.

Roles

Rollen beschreiben eine Reihenfolge von Cookbooks und oder Recipes. Auch können in Rollen Attribute überschrieben werden. Rollen eignen sich beispielsweise für die funktionale Einteilung von Servern. So kann man zum Beispiel für alle Oracle-DB-Server eine Rolle oracle-db erstellen. In einer Rolle können auch für diese Rolle gültige Attribute definiert werden.

Environments

In Environments definiert man normalerweise die unterschiedlichen Funktionsstufen wie Entwicklung, Integration und Produktion einer Umgebung. Diesen Environments können dann Cookbooks mitsamt Versionen zugewiesen werden. So kann etwa eine neuere Version eines Cookbook zuerst in dem Environment „Entwicklung“ eingesetzt, anschließend in das Environment „Integration“ deployt und zum Schluss erst in der „Produktion“ aktiviert werden.

Oracle in Chef

Im Folgenden wird die Installation einer Oracle-19c-Datenbank mittels Chef Infra auf einem Red-Hat-Enterprise-Linux-7-System vorgestellt. Dabei findet eine Integration in die Umsysteme Oracle Cloud Control 13.3, Tivoli Spectrum Protect 8.1.9 und CheckMK (basierend auf Nagios) statt. Die Datenfiles inklusive Control-Files, Online-Redo-Log-Files, Archived-Redo-Log-Files sind in zwei ASM-Diskgruppen abgelegt, nur das spfile, pfile und Password File sind im Filesystem im ORACLE_HOME vorhanden. Der Ablauf der Installation als Überblick über die einzelnen Schritte ist im *Listing 1* aufgeführt.

Die Notation ‘::<name>’ bedeutet in Chef, dass ein anderes Recipe im gleichen

```
# Basis
include_recipe '::oracle-users'
include_recipe '::lvm-fs'
include_recipe '::dep-pkg-install'
include_recipe '::kernel-params'

# Grid / ASM Installation und Integration
include_recipe '::asm-sw-install'
include_recipe '::crs-setup'
include_recipe '::create-asm-instance'

# Cloud Control Agent Installation und Integration
include_recipe '::emcli-install'
include_recipe '::emcc-agent-install'
include_recipe '::emcli-reg-lsnr-asm'

# DB Software Installation
include_recipe '::db-sw-install'
include_recipe '::systemctl-odb'
clear_oraInventory 'clear OraInventory' do
  action :clear
end

# Backup Integration
include_recipe '::tdpo-install'
```

Listing 1: Übersicht über die Oracle-Installation

Cookbook aufgerufen wird. Im Recipe '::oracle-users' werden zuerst die OS-User und Gruppen für die Datenbank als auch die Grid-Infrastruktur angelegt. Sowohl für Gruppen als auch für User existieren in Chef entsprechende Ressourcen *(siehe Listing 2)*.

Daneben werden noch die User-Limits, sudo-Regeln und E-Mail-Forwarding für die beiden User oracle und grid eingestellt. Im Recipe '::lvm-fs' werden eine Volume Group mit Logical Volumes und Filesysteme für die Mount-Points /u01 (späteres Oracle-Software-Verzeichnis), /u10 für das Diag-Verzeichnis und /u80 für Dumps, ... angelegt. Die Verzeichnis-Namen sind natürlich anpassbar und als Attribute definiert. Das Recipe '::dep-pkg-install' installiert die notwendigen OS-Packages für eine erfolgreiche Oracle-Installation. Die entsprechenden Packages sind als Attribute-Array definiert und können so dynamisch angepasst und erweitert werden. Auch wird das Kernel-Package oracleasm konfiguriert. '::kernel-params' installiert ein Tuned-Profile für Oracle inklusive HugePages. Alle Kernel-Parameter und -Werte sind auch als Attribute definiert. Wichtig ist hier, dass nur bei Änderungen des Tuned-Profile dieses auch aktiviert wird: Der Tuned resettet temporär alle Werte auf die Defaults zurück, wenn ein Profile aktiviert wird. Wenn genau in diesem Augenblick in der Datenbank beispielsweise eine neue Session eröffnet wird, kann das zu Fehlern führen (durch nicht vorhandene HugePages, Shared-Memory-Segmente oder fehlende Semaphore). Die bisherigen Schritte sind teilweise auch in den Preinstallation-RPMs von Oracle enthalten. Die Preinstallation-RPMs stehen allerdings nur Oracle Linux Support Customers zur Verfügung.

```
%w(oinstall dba asmdba asmoper asmadmin).each do |group_name|
  group group_name do
      gid     node['odb']['group'][group_name]
      action :create
      only_if { node['etc']['group'][group_name.to_s].nil? }
  end
end
user 'oracle' do
  uid            node['odb']['user']['oracle']['uid']
  gid            node['odb']['user']['oracle']['gid']
  shell          node['odb']['user']['shell']
  comment        'oracle admin user'
  manage_home    true
  password       node['odb']['user']['pwd']
  action         :create
end
```

Listing 2: Oracle-User und Gruppen erstellen

```
template "#{node['odb']['asm']['ora_base']}/asm_swonly_193.rsp" do
  source 'asm_swonly_193.rsp.erb'
  owner  'grid'
  group  'oinstall'
  mode   '0640'
  variables(hostname: node['hostname'])
end
```

Listing 3: Erzeugen eines Response-Files

GRID-Software

Anschließend fängt die eigentliche Software-Installation an. Dazu wird als Erstes ein Response-File auf dem Node erzeugt *(siehe Listing 3)*.

Im Cookbook ist ein Template mit einem Platzhalter hostname vorhanden, das nun unter $ORACLE_BASE auf dem Filesystem abgelegt wird. Anschließend wird mithilfe einer bash-Ressource das (Golden-) Image-File der Grid-Software ausgepackt und das Skript gridSetup.sh unter anderem mit dem oben erzeugten Response-File für die Installation der Software aufgerufen. Das Package cvuqdisk wird aus dem Grid-Home-Verzeichnis installiert.

Im nächsten Recipe '::crs-setup' werden mit dem roothas-Perl-Skript (resp. Shell-Skript) die Cluster Ready Services (CRS) konfiguriert.

Jetzt kann die ASM-Instanz erzeugt werden (im Recipe '::create-asm-instance'). Dazu werden zuerst ein init+ASM.ora-File, das Password-File und das Audit Dump Directory erzeugt. Anschließend wird mit dem srvctl Utility die ASM-Instanz zu den CRS hinzugefügt und dann gestartet. Um unnötig viele Audit-Trace-Files zu vermeiden, wird das Check-Interval in den CRS auf 60 Sekunden hochgesetzt. Für das Management und Monitoring mittels Cloud Control werden der User asmsnmp erzeugt und die notwendigen Rechte zugewiesen. Ebenso wird zum Schluss noch ein spfile im Filesystem erzeugt. Damit ist die Installation und Konfiguration der Grid-Software und der CRS abgeschlossen.

Installation Cloud Control Agent

Für die Installation und Konfiguration des Oracle Cloud Control Agent wird das Cloud Control CLI benötigt. Beide Software-Komponenten stellt der Cloud Control Server über eine Web-Schnittstelle zur Verfügung. Das CLI steht über die URL *https://cc-server:port/public_lib_download/emcli/kit/emcliadvancedkit.jar* zum Download mittels wget zur Verfügung. Es wird noch ein JRE oder JDK zum Auspacken und für die Runtime benötigt. Anschließend muss noch die Verbindung zum Cloud Control Server in einem Aufruf 'emcli setup' definiert werden. Je nach vorhandenen Security-Richtlinien können dabei die Optionen -autologin, -certrans=yes, -licans=yes und -trustall benutzt werden. Damit sind die Vorbereitungen für die Installation des Agent geschaffen. Ist auf dem Linux-System der firewalld aktiv (Abfrage durch 'firewall-cmd –state'), so sollten die benutzten Ports freigeschaltet werden. Auch der Agent (resp. ein Shell-Skript für die Installation des Agent) kann vom Cloud Control Server über die URL *https://cc-server:port/install/getAgentImage* direkt heruntergeladen werden. Für die Installation des Agent wird dynamisch ein Response-File angelegt. Das heruntergeladene File wird mit den Parametern RSPFILE_LOC und AGENT_BASE_DIR aufgerufen. Nach der Installation und Konfiguration werden dem Agent noch die Default-Targets definiert *(siehe Listing 4)*.

Nach der Konfiguration können nun die CRS-Services (als Type „has"), die ASM-Ins-

```
# add internal targets to agent
bash 'add_internal_targets' do
    user          'oracle'
    group         'oinstall'
    environment   node['odb']['dba'][dba_env.to_s]
    code <<-EOC
        cd #{node['odb']['emcc']['orahome']}/agent_#{node['odb']['emcc']['version']}.0.0.0/bin
        ./emctl config agent addinternaltargets
    EOC
end
```

Listing 4: Definition der Default Targets

```
# register the has service
bash 'EM CC register has service' do
   user          'oracle'
   group         'oinstall'
   environment   'JAVA_HOME' => '/usr/java/latest'
   code <<-EOC2
       cd #{node['odb']['emcli']['home']}
       HAS_CNT=`./emcli get_targets -targets=has_#{node['fqdn']}:has -noheader | wc -l`
       [[ ${HAS_CNT} -eq 0 ]] && ./emcli add_target \
           -name="has_#{node['fqdn']}" \
           -type="has" \
           -host=#{node['fqdn']} \
           -properties="OracleHome:#{node['odb']['asm']['ora_home']};eonsPort:2016"
       exit 0
   EOC2
end
```

Listing 5: Registrierung der CRS im Cloud Control

tanz und der Datenbank-Listener im Cloud Control registriert werden. Dazu wird als Erstes mit „get_targets" überprüft, ob die Komponente schon registriert sein sollte (von einem vorherigen Chef-Run). Als Beispiel werden im *Listing 5* die CRS registriert.

Oracle-Datenbank-Software

Auch bei der Installation der Oracle-Datenbank-Software wird ein Response-File für den runInstaller erzeugt und anschließend der runInstaller in einer bash-Ressource aufgerufen (mit einigen Parametern). Dann werden die nicht gewünschten Optionen (wie OLAP, RAT, ...) aus dem Oracle Binary entfernt und die benötigten (lizenzierten) (wie z.B. Unified Auditing, Partitioning) hinzugefügt.

Als Abschluss wird noch die Integration der zukünftigen Datenbanken in den System-Start-Prozess systemd erstellt.

Backup Software

Im Folgenden wird die Integration, Installation und Konfiguration der Backup-Software Tivoli Spectrum Protect for Oracle beschrieben. Entsprechende Automatisierungs-Hilfsmittel und Vorgehensweisen können auch für andere Backup-Software benutzt werden. Von einem (lokalen) Software-Repository wird die Backup-Software heruntergeladen. Die silent-Installation benötigt ein Response-File sowie eine einfache User-Interaktion *(siehe Listing 6)*.

Falls auf dem Backup-Server noch ein Passwort für den DB-Server definiert wurde, muss auf dem Client ein sogenanntes Passwort-File erstellt werden. Dies bedingt bei Tivoli Spectrum Protect die interaktive Eingabe des Passworts. Eine einfache Ein- und Ausgabe-Umleitung wie im *Listing 6* ist hier nicht mehr ausreichend. Daher wird für die Automatisierung auf das Tool expect zurückgegriffen. Dieses sucht in der Ausgabe nach definierten Strings (oder regulären Ausdrücken) und liefert dann die definierte Antwort zurück. In Chef ist das Vorgehen im *Listing 7* aufgeführt.

Nagios-/CheckMK-Integration

Als Basis für das Monitoring der Oracle-Datenbanken wird das Skript check_oracle_health von Gerhard Lauser verwendet. Allerdings wurde das Monitoring um eine dynamische Komponente erweitert: In der zu beobachtenden Datenbank werden in einer Tabelle die Monitoring-Checks und in einer zweiten die ange-

```
# install agent
bash 'install tdpo software' do
   code <<-EOC
       #{tdpo_bin_file} -i silent -f /var/tmp/tdpo_response.txt <<EOF
           2
       EOF
   EOC
end
```

Listing 6: Installation Backup Client Software

```
# create password file
expect_script 'create tdpo password file' do
  user 'root'
  group 'root'
  code <<-EOC
      spawn /bin/tdpoconf PASSWord -tdpo_opt=/opt/tech/scripts/oracle/rman_scripts/tdpo.opt
      expect {
          -regexp "Please enter current password:" {
              exp_send "#{node['hostname']}-bkp-ora\r"
              exp_continue
          }
          -regexp "Please enter new password:" {
              exp_send "#{node['hostname']}-bkp-ora\r"
              exp_continue
          }
          -regexp "Please reenter new password for verification:" {
              exp_send "#{node['hostname']}-bkp-ora\r"
              exp_continue
          }
      }
  EOC
  not_if { File.exist?('/home/oracle/scripts/rman_scripts/TSM.IDX') }
end
```

Listing 7: Konfiguration des TDPO-Password-Files

```
  # create sql script:
  template "#{node['odb']['check_mk']['script_dir']}/create_nagios.#{sid}.sql" do
      source 'check_mk.sql.erb'
      owner  'oracle'
      group  'oinstall'
      mode   '0640'
      action :create
      not_if File.exist?("#{node['odb']['check_mk']['script_dir']}/create_nagios.#{sid}.sql")
  end
```

Listing 8: Erzeugen des SQL-Skripts für das Monitoring

passten Thresholds gespeichert. Dementsprechend wird beim Erzeugen der Datenbank ein SQL-Skript zum Erstellen des Monitoring-Users, der Tabellen und das Befüllen derselben mit den Default-Checks und Thresholds ausgeführt. Das Skript ist als Template vorhanden *(siehe Listing 8)*, das anschließend mittels sqlplus ausgeführt wird.

Damit wäre dann die Installation der Grid- und Datenbank-Software mit Integration in Cloud Control, der Backup-Software und CheckMK/Nagios automatisiert. Diese Mechanismen können nun auch für weitere Upgrades der Software benutzt werden, wie etwa für ein Out-Of-Place-Patching.

Genauso kann auch das Erstellen einer Datenbank-Instanz automatisiert werden. Inzwischen wurde auch ein Cookbook für das automatisierte In-Place-Patching erstellt, das natürlich auch vorab Applikationen stoppen und Blackouts sowohl im Cloud Control als auch in anderen Monitoring-Applikationen setzen kann.

Zusammenfassung

Oracle-Infrastrukturen können mit Chef automatisiert betrieben werden. Dabei werden die Installation und Konfiguration sowohl der Datenbank als auch einer GRID-Infrastruktur unterstützt. Es können auch Umsysteme wie Cloud Control, Backup-Systeme sowie weitere Monitoring-Systeme automatisch eingebunden werden. Manuelle und fehlerträchtige Tasks können auf ein Minimum reduziert werden.

Über den Autor

Bernd Patolla ist bei der Firma In&Out AG in der Schweiz als Senior IT Consultant tätig. Die Schwerpunkte bilden Oracle-Datenbanken, Server- und Storage-Infrastrukturen. Seit mehreren Jahren arbeitet Bernd Patolla in diversen Projekten im Bereich Automatisierungen im Chef-Umfeld.

Bernd Patolla
bernd.patolla@inout.ch