Intune-CaC-Automation Konzept

Ausgangslage

Das Amt für Informatik (AFI) ist das zentrale Informatikunternehmen des Kantons Zürich. Es bietet umfassende Leistungen im Bereich der zentralen Informatikinfrastruktur und kantonsweiter Anwendungen für die Direktionen, Behörden, Rechtspflege und selbstständigen Anstalten. Mit der Umsetzung der neuen Strategie Informations- und Kommunikationstechnologie (IKT) begleitet das AFI den Kanton auf seinem Weg in die digitale Welt.

Intune als zentraler Baustein der IKT-Strategie

In beiden Tenant-Umgebungen (GOV/EDU) wurde im Einklang mit der Microsoft-Strategie der Endpoint Manager Intune eingeführt. Intune vereint alle Konfigurationsprofile (früher bekannt als GPOs), Sicherheits- und Compliance-Richtlinien in einer zentralen Plattform.

Herausforderung: Datensicherung und Wiederherstellung

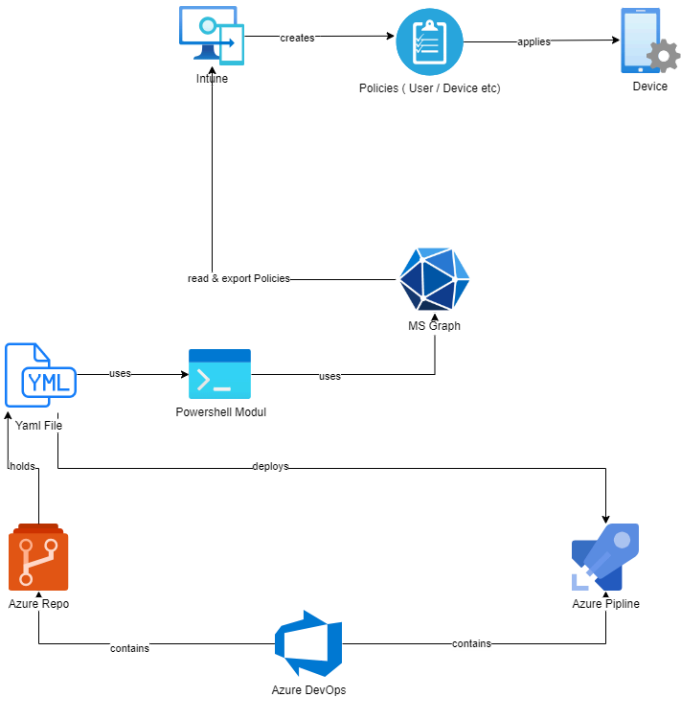
Im Falle eines Cyberangriffs oder eines Ausfalls des Schweizerischen Rechenzentrums besteht derzeit keine Möglichkeit, die verlorenen Daten wiederherzustellen. Dies würde den Verlust des Fundaments aller Einstellungen für den Dienst des Digitalen Arbeitsplatzes bedeuten.

Handlungsempfehlung: Implementierung einer umfassenden Datensicherungslösung

Um dieses Risiko zu minimieren, ist die Implementierung einer umfassenden Datensicherungslösung dringend erforderlich. Diese Lösung sollte die folgenden Anforderungen erfüllen:

- **Regelmässige Sicherung aller relevanten Daten**
- **Sichere Speicherung der Sicherungsdaten an einem externen Standort**
- **Möglichkeit zur schnellen Wiederherstellung der Daten im Falle eines Ausfalls**

Architektur



Was ist Configuration as Code (CaC)

Configuration as Code (CaC) beschreibt die Anwendung von Softwareentwicklungsbestätigungen auf die Verwaltung und Bereitstellung spezifischer Konfigurationen oder Einstellungen für eine Anwendung. Im Intune-Kontext ermöglicht CaC die Definition und Verwaltung von Intune-Einstellungen mithilfe von Code.

Was ist IntuneCD ?

IntuneCD, ausgeschrieben **Intune Continuous Delivery**, ist ein Ansatz zur Verwaltung von Konfigurationen in Microsoft Intune mithilfe von Code. Anstatt Einstellungen manuell über die Intune-Konsole vorzunehmen, ermöglicht IntuneCD die Definition und Bereitstellung dieser Einstellungen in einer automatisierten Art und Weise.

Hier ein kurzer Vergleich der beiden Herangehensweisen:

Verwaltungsmethode	Beschreibung

Manuelle Konfiguration	Intune-Einstellungen werden manuell über die webbasierte Intune-Konsole hinzugefügt, geändert und gelöscht.
IntuneCD	Intune-Einstellungen werden als Code in einer deklarativen Sprache (z.B. JSON, YAML) definiert. Dieser Code wird dann versioniert und in einem Versionskontrollsystem wie Git gespeichert. Abschließend wird die automatische Bereitstellung der Konfigurationen mithilfe von Continuous Integration- und Continuous Delivery (CI/CD)-Pipelines ermöglicht.

Was sind Azure Devops Piplines ?

Azure DevOps Pipelines ist eine Cloud-basierte Plattform für die Continuous Integration (CI) und Continuous Delivery (CD) von Softwareanwendungen. Sie ermöglicht Entwicklern und IT-Teams die Automatisierung von Aufgaben wie dem Kompilieren, Testen und Bereitstellen von Code. Dies führt zu einer schnelleren Entwicklung, höheren Qualität und verbesserten Zuverlässigkeit von Softwareanwendungen.

Die wichtigsten Funktionen von Azure DevOps Pipelines sind:

- **CI/CD-Pipelines:** Erstellen Sie automatisierte Pipelines, die Code kompilieren, testen und bereitstellen.
- **Testen:** Führen Sie verschiedene Arten von Tests durch, z. B. Unit-Tests, Integrationstests und Funktionstests.
- **Bereitstellung:** Stellen Sie Code in verschiedene Umgebungen bereit, z. B. Entwicklung, Test und Produktion.
- **Überwachung:** Überwachen Sie den Status Ihrer Pipelines und den Fortschritt Ihrer Bereitstellungen.
- **Zusammenarbeit:** Arbeiten Sie mit anderen Teammitgliedern zusammen, um Pipelines zu erstellen, zu verwalten und zu optimieren

Vorteile von Configuration as Code

CaC bietet folgende Vorteile:

- **Verbesserte Konsistenz und Wiederholbarkeit:** CaC stellt sicher, dass Intune-Einstellungen konsistent und wiederholbar über verschiedene Umgebungen hinweg angewendet werden. Dies reduziert das Risiko menschlicher Fehler und vereinfacht die Verwaltung komplexer Umgebungen.
- **Vereinfachte Zusammenarbeit:** CaC ermöglicht die gemeinsame Nutzung von Intune-Konfigurationen zwischen Teams und Abteilungen. Dies fördert die Zusammenarbeit und erleichtert die schnelle und effiziente Implementierung neuer Richtlinien.
- **Verbesserte Sichtbarkeit und Kontrolle:** CaC bietet eine zentrale Quelle der Wahrheit für Intune-Konfigurationen. Dies ermöglicht es Administratoren, den Status ihrer Umgebung schnell und einfach zu überwachen und Änderungen nachzuverfolgen.
- **Automatisierte Bereitstellung:** CaC kann mit Continuous Integration- und Continuous Delivery (CI/CD)-Pipelines integriert werden, um Intune-Einstellungen automatisch bereitzustellen. Dies ermöglicht eine schnelle und effiziente Aktualisierung von Richtlinien und die Reduzierung von Ausfallzeiten.

Vorteile von IntuneCD

IntuneCD ist eine Methode zur kontinuierlichen Bereitstellung und Verwaltung von Intune-Einstellungen mithilfe von Code. Sie ermöglicht die Automatisierung der Konfiguration und Bereitstellung von Intune-Richtlinien, wodurch Unternehmen die folgenden Vorteile erzielen können:

- **Verbesserte Effizienz:** IntuneCD automatisiert manuelle Prozesse, wodurch Zeit und Ressourcen eingespart werden.
- **Reduzierte Fehlerquote:** Menschliche Fehler werden durch die Automatisierung minimiert, was zu einer konsistenteren und zuverlässigeren Umgebung führt.
- **Erhöhte Agilität:** IntuneCD ermöglicht die schnelle und einfache Implementierung neuer Richtlinien, wodurch Unternehmen schneller auf sich ändernde Geschäftsanforderungen reagieren können.
- **Verbesserte Sichtbarkeit und Kontrolle:** IntuneCD bietet eine zentrale Quelle der Wahrheit für Intune-Einstellungen, wodurch Administratoren den Status ihrer Umgebung schnell und einfach überwachen und Änderungen nachverfolgen können.

Vorteile von Azure Devops Piplines

Azure DevOps Pipelines ist ein leistungsstarkes Tool, mit dem Unternehmen aller Grössenordnungen die Qualität ihrer Software verbessern, Kosten senken und die Zusammenarbeit optimieren können.

- **Automatisierung:** Automatisierung von Builds, Tests und Bereitstellungen für schnellere und effizientere Softwareentwicklung.
- **CI/CD:** Implementierung von Continuous Integration und Continuous Delivery (CI/CD) für schnellere und zuverlässigere Bereitstellungen.
- **Verbesserte Qualität:** Integration von automatisierten Tests zur frühzeitigen Fehlererkennung und -Behebung.
- **Kosteneinsparungen:** Zeit- und Ressourceneinsparung durch Automatisierung.
- **Transparenz:** Umfassende Einblicke in den Build-, Test- und Bereitstellungsprozess.
- **Verbesserte Zusammenarbeit:** Erleichterung der Zusammenarbeit von Teams an der Softwareentwicklung.
- **Integration:** Integration mit anderen Tools wie GitHub, Jira und Azure Key Vault für einen nahtlosen Workflow.
- **Skalierbarkeit:** Anpassung an die Bedürfnisse kleiner Teams bis hin zu großen Unternehmen.

Key Concepts

In diesem Abschnitt werden die wichtigsten Konzepte und Komponenten vom Backup Service für Intune vorgestellt.

IntuneCD

IntuneCD ist ein Ansatz zur Verwaltung von Konfigurationen in Microsoft Intune mithilfe von Code. Anstatt Einstellungen manuell über die Intune-Konsole vorzunehmen, ermöglicht IntuneCD die Definition und Bereitstellung dieser Einstellungen in einer automatisierten Art und Weise. Dies bietet mehrere Vorteile, darunter:

- Verbesserte Konsistenz und Wiederholbarkeit
- Minimierung von Fehlern
- Automatisierte Bereitstellung
- Einfache Nachverfolgung von Änderungen

Azure DevOps Pipeline:

Azure DevOps Pipeline ist eine Plattform für die Automatisierung von Aufgaben und Workflows. Sie ermöglicht die Erstellung von Pipelines, die aus mehreren Aufgaben bestehen, die nacheinander ausgeführt werden. Dies kann für die Automatisierung von Aufgaben wie der Sicherung Ihrer Intune-Umgebung verwendet werden.

Sicherungsprozess:

Der Sicherungsprozess mithilfe von IntuneCD und Azure DevOps Pipeline umfasst die folgenden Schritte:

1. **Exportieren Sie Ihre Intune-Einstellungen, Rollen, Richtlinien, Gruppen und Apps** Dies wird mithilfe von PowerShell-Cmdlets aus der Azure DevOps-Erweiterung für Intune durchgeführt.
2. **Speichern Sie die exportierten Daten in Azure Repositorie** Azure Repo ist eine skalierbare und sichere Cloud-Speicherlösung für Code.
3. **Erstellen Sie eine Azure DevOps-Pipeline** Die Pipeline definiert die Aufgaben, die zur Sicherung Ihrer Intune-Umgebung ausgeführt werden müssen.
4. **Konfigurieren Sie einen Trigger für Ihre Pipeline** Der Trigger startet die Pipeline automatisch in regelmäßigen Abständen.
5. **Testen Sie Ihre Pipeline** Stellen Sie sicher, dass die Pipeline wie erwartet funktioniert.

Wiederherstellungsprozess:

Der Wiederherstellungsprozess mithilfe von IntuneCD und Azure DevOps Pipeline umfasst die folgenden Schritte:

1. **Stellen Sie die exportierten Daten aus Azure Repo wieder her**
2. **Importieren Sie die wiederhergestellten Daten in Intune** Dies wird mithilfe von PowerShell-Cmdlets aus der Azure DevOps-Erweiterung für Intune durchgeführt.
3. **Verifizieren Sie Ihre Intune-Umgebung** Stellen Sie sicher, dass die Wiederherstellung erfolgreich war.

Vorteile:

Die Sicherung Ihrer Intune-Umgebung mit IntuneCD und Azure DevOps Pipeline bietet mehrere Vorteile, darunter:

- **Automatisierte Sicherung:** Die Sicherung Ihrer Intune-Umgebung wird automatisch mithilfe von Azure DevOps Pipeline durchgeführt, wodurch manueller Aufwand und Fehlerquellen reduziert werden.
- **Regelmässige Sicherungen:** Sie können festlegen, dass Ihre Intune-Umgebung in regelmäßigen Abständen gesichert wird, um sicherzustellen, dass Sie immer über eine aktuelle Sicherung verfügen.
- **Sichere Speicherung:** Ihre Sicherungen werden in Azure Repository gespeichert, einer skalierbaren und sicheren Cloud-Speicherlösung.
- **Einfache Wiederherstellung:** Im Falle eines Ausfalls oder eines Cyberangriffs können Sie Ihre Intune-Umgebung schnell und einfach aus Ihrer Sicherung wiederherstellen.

Voraussetzungen

Diese Lösung erfordert mehrere Komponenten:

- **Azure DevOps**: Eine Cloud-Plattform für die Zusammenarbeit und Automatisierung von Softwareentwicklungsprozessen. Sie beinhaltet auch ein **Git-Repository** zur Versionskontrolle des Codes.
- **Azure DevOps Pipeline**: Eine Funktion innerhalb von Azure DevOps, mit der Sie automatisierte Skripte (Code) ausführen können. Diese Pipeline wird den eigentlichen Sicherungsvorgang durchführen.
- **Azure Active Directory (Azure AD) Workloadidentität**: Auch bekannt als **Azure Service Principal**. Dies ist eine Identität, die Anwendungen (in diesem Fall die Azure DevOps Pipeline) verwenden können, um sich bei Azure-Diensten wie Microsoft Graph API zu authentifizieren und auf die Intune-Ressourcen zuzugreifen.

Zuständigkeiten

DevOps Pipelines

Die **Verantwortung** für die DevOps-Pipelines liegt beim Workplace-Team. Benachrichtigungen über den Erfolg oder Misserfolg eines Pipeline-Durchlaufs können auf Wunsch auf Personen aus dem SAS-Team erweitert werden.

Update Management

Die **Verantwortung** für die Aktualisierung des Backups (YAML-Datei) liegt beim Workplace-Team.

Die Aktualisierung des Backups erfolgt täglich.

Wiederherstellen

Die Wiederherstellung von einer früheren Intune-Instanz muss durch einen **Change im ITSM-System** genehmigt werden. Die **Umsetzung des Changes** erfolgt durch einen Workplace Engineer. Die **Kommunikationsrichtlinie** für diesen Prozess muss noch **festgelegt werden** (GOV/EDU).

Betriebliche Aufgaben

Die operativen Aufgaben sind noch zu bestimmen.

Task	Beschreibung

Billing & Cost Management

Tenant	Kosten
GOV	**kostenlos**
EDU	**kostenlos**
Test-Tenant	**kostenlos**

Security

Graph API Applikationsberechtigung

- `DeviceManagementApps.Read.All`
- `DeviceManagementConfiguration.ReadWrite.All` (Read**Write** because of this "feature")
- `DeviceManagementManagedDevices.Read.All`
- `DeviceManagementServiceConfig.Read.All`
- `Group.Read.All`
- `Policy.Read.All`
- `Policy.Read.ConditionalAccess`
- `DeviceManagementRBAC.Read.All`

Applikation - Intune-Backupper

Code

```yaml
trigger: none
schedules:
  - cron: "0 1 * * *"
    displayName: "1am every day"
    branches:
      include:
        - main
    always: true
variables:
  - name: BACKUP_FOLDER
    value: prod-backup
  - name: TENANT_NAME
    value: dce0a2f3-3f15-406f-8e61-70888a8ba255
  - name: SERVICE_CONNECTION_NAME
    value: intune_backup_connection
  - name: USER_EMAIL
    value: IT_Intune_Backupper@contoso.com
  - name: USER_NAME
    value: IT_Intune_Backupper

jobs:
  - job: backup_intune
    displayName: Backup & commit Intune configuration
    pool:
      vmImage: ubuntu-latest
    continueOnError: false
    steps:
      - checkout: self
        persistCredentials: true

      - task: Bash@3
        displayName: Remove existing prod-backup directory
        inputs:
          targetType: "inline"
          script: |
            rm -rfv "$(Build.SourcesDirectory)/$(BACKUP_FOLDER)"
          workingDirectory: "$(Build.SourcesDirectory)"
```

```yaml
        failOnStderr: false

    - task: Bash@3
      displayName: Install IntuneCD
      inputs:
        targetType: "inline"
        script: |
          pip3 install IntuneCD
        workingDirectory: "$(Build.SourcesDirectory)"
        failOnStderr: true

    - task: AzurePowerShell@5
      displayName: "Get Graph Token for Workload Federated Credential"
      inputs:
        azureSubscription: $(SERVICE_CONNECTION_NAME)
        azurePowerShellVersion: "LatestVersion"
        ScriptType: "inlineScript"
        Inline: |
          $accessToken = (Get-AzAccessToken -ResourceTypeName MSGraph -ErrorAction Stop).Token
          Write-Host "##vso[task.setvariable variable=accessToken;issecret=true]$accessToken"

    # Backup the latest configuration, using the current directory
    - task: Bash@3
      displayName: Create Intune backup
      inputs:
        targetType: "inline"
        script: |
          mkdir -p "$(Build.SourcesDirectory)/$(BACKUP_FOLDER)"

          BACKUP_START=`date +%Y.%m.%d:%H.%M.%S`
          # set BACKUP_START pipeline variable
          echo "##vso[task.setVariable variable=BACKUP_START]$BACKUP_START"

          IntuneCD-startbackup \
              -t $(accessToken) \
              --mode=1 \
              --output=json \
              --path="$(Build.SourcesDirectory)/$(BACKUP_FOLDER)" \
              --exclude CompliancePartnerHeartbeat ManagedGooglePlay VPPusedLicenseCount\
              --append-id \
              --ignore-omasettings
        workingDirectory: "$(Build.SourcesDirectory)"
        failOnStderr: true

    # Commit changes and push to repo
    - task: PowerShell@2
      displayName: Find change author & commit the backup
      name: commitAndSetVariable
      inputs:
        targetType: "inline"
        script: |
          # $verbosePreference = 'continue'

          $root = "$(Build.SourcesDirectory)"

          Set-Location $root

          # configure GIT defaults
          git config --global user.name 'unknown'
          git config --global user.email 'unknown@unknown.com'
          # to avoid 256 limit on Windows
          git config --global core.longpaths true
          # to support UNICODE
          git config --global core.quotepath off
          # to avoid 'CRLF will be replaced by LF the next time Git touches it'
          git config --global core.eol lf
          git config --global core.autocrlf false

          # get changed config files
          $untrackedFile = git ls-files --others --exclude-standard --full-name
          $trackedFile = git ls-files --modified --full-name
          $changedFile = $untrackedFile, $trackedFile | % { $_ } | ? { $_ }

          # "status"
          # git --no-pager status

          # "diff"
          # git --no-pager diff

          if ($changedFile) {
              # set CHANGE_DETECTED pipeline variable
              echo "##vso[task.setVariable variable=CHANGE_DETECTED;isOutput=true;]1"

```

```
                # install required Graph modules (for authentication and getting audit logs)
                if (!(Get-Module "Microsoft.Graph.DeviceManagement.Administration" -ListAvailable)) {
                  Install-Module Microsoft.Graph.DeviceManagement.Administration -AllowClobber -Force -AcceptLicense
                }

                #region authenticate to Graph API using service principal secret
                Write-Host "Authenticating to Graph API"
                $secureToken = ConvertTo-SecureString -String $(accessToken) -AsPlainText -Force
                Connect-MgGraph -AccessToken $secureToken -NoWelcome
                #endregion authenticate to Graph API using service principal secret

                #region helper functions
                # function to be able to catch errors and all outputs
                function _startProcess {
                    [CmdletBinding()]
                    param (
                        [string] $filePath = ''
                        ,
                        [string] $argumentList = ''
                        ,
                        [string] $workingDirectory = (Get-Location)
                        ,
                        [switch] $dontWait
                        ,
                        # lot of git commands output verbose output to error stream
                        [switch] $outputErr2Std
                    )

                    $p = New-Object System.Diagnostics.Process
                    $p.StartInfo.UseShellExecute = $false
                    $p.StartInfo.RedirectStandardOutput = $true
                    $p.StartInfo.RedirectStandardError = $true
                    $p.StartInfo.WorkingDirectory = $workingDirectory
                    $p.StartInfo.FileName = $filePath
                    $p.StartInfo.Arguments = $argumentList
                    [void]$p.Start()
                    if (!$dontWait) {
                        $p.WaitForExit()
                    }

                    $result = $p.StandardOutput.ReadToEnd()
                    if ($result) {
                        # to avoid returning of null
                        $result
                    }
                    if ($outputErr2Std) {
                        $p.StandardError.ReadToEnd()
                    } else {
                        if ($err = $p.StandardError.ReadToEnd()) {
                            Write-Error $err
                        }
                    }
                }

                function _getResourceId {
                  [CmdletBinding()]
                  param (
                    [string] $filePath
                  )

                  $fileName = [System.IO.Path]::GetFileNameWithoutExtension($filePath)

                  # some files are just additional content for an existing config JSON and IntuneCD author decided to not put ResourceId in their name (a.k.a. reso
                  # some files just don't have ResourceId in their name because of the IntuneCD author decision
                  if ($filePath -like "*Device Configurations/mobileconfig/*") {
                    $parentFolderPath = Split-Path (Split-Path $filePath -Parent) -Parent
                    $fileName = Get-ChildItem $parentFolderPath -File | ? {
                      (ConvertFrom-Json -InputObject (Get-Content $_.FullName -Raw)).payloadFileName -eq [System.IO.Path]::GetFileName($filePath)
                      } | select -expand BaseName
                    if (!$fileName) {
                      #FIXME throw az budu umet vytahnout parent file i pri DELETE operaci
                      Write-Warning "Unable to find 'parent' config file for $filePath"
                      return
                    }
                  } elseif ($filePath -like "*/Managed Google Play/*") {
                    return ($modificationEvent | ? { $_.Category -eq 'Enrollment' -and $_.ActivityType -eq "Patch AndroidForWorkSettings" }).Resources.ResourceId
                  }

                  # parse resource ID from the file name
                  # file name is in format <policyname>__<ID>
                  # beware that it doesn't have to be GUID! For example ESP profile, Apple configurator profile etc uses as ID <guid>_guid>, <guid>_string
                  $delimiter = "__"
                  if ($fileName -like "*$delimiter*") {
```

```
                    $resourceId = ($fileName -split $delimiter)[-1]
                    # just in case file name contains more than two following underscores in a row which would lead to ID starting with underscore(s)
                    $resourceId = $resourceId -replace "^_*"
                  } else {
                    $resourceId = $null
                  }

                  return $resourceId
                }
                #endregion helper functions

                # get date of the last config backup commit, to have the starting point for searching the audit log
                # because of shallow clones, I need to fetch more data before calling git log
                $gitCommitDepth = 30
                git fetch --depth=$gitCommitDepth
                $commitList = _startProcess git "--no-pager log --no-show-signature -$gitCommitDepth --format=%s%%%%%%%cI" -outputErr2Std -dontWait
                $lastCommitDate = $commitList -split "`n" | ? {$_} | % {
                  $commitName, $commitDate = $_ -split "%%%"
                  if ($commitName -match "^\d{4}\.\d{2}\.\d{2}_\d{2}\.\d{2} -- ") {
                    # config backup commit name is in a format '2023.10.08_01.01 -- ...'
                    $commitDate
                  }
                }
                if ($lastCommitDate) {
                  # pick the newest and convert it to datetime object
                  $lastCommitDate = Get-Date @($lastCommitDate)[0]
                } else {
                  Write-Warning "Unable to obtain date of the last backup config commit. ALL Intune audit events will be gathered."
                }

                # array where objects representing each changed file will be saved with information like who made the change etc
                $modificationData = New-Object System.Collections.ArrayList

                #region get all Intune audit events since the last commit
                # it is much faster to get all events at once then retrieve them one by one using resourceId
                #region create search filter
                $filter = "activityResult eq 'Success'", "ActivityOperationType ne 'Get'"

                if ($lastCommitDate) {
                  # Intune logs use UTC time
                  $lastCommitDate = $lastCommitDate.ToUniversalTime()
                  $filterDateTimeFrom = Get-Date -Date $lastCommitDate -Format "yyyy-MM-ddTHH:mm:ss"
                  $filter += "ActivityDateTime ge $filterDateTimeFrom`Z"
                }

                $backupStart = [DateTime]::ParseExact('$(BACKUP_START)', 'yyyy.MM.dd:HH.mm.ss', $null)
                $backupStart = $backupStart.ToUniversalTime()
                $filterDateTimeTo = Get-Date -Date $backupStart -Format "yyyy-MM-ddTHH:mm:ss"
                $filter += "ActivityDateTime le $filterDateTimeTo`Z"

                $eventFilter = $filter -join " and "
                #endregion create search filter

                "`nGetting Intune event logs"
                "`t- from: '$lastCommitDate' (UTC) to: '$backupStart' (UTC)"
                "`t- filter: $eventFilter"
                # Get-MgDeviceManagementAuditEvent requires DeviceManagementApps.Read.All scope
                $modificationEvent = Get-MgDeviceManagementAuditEvent -Filter $eventFilter -All
                #endregion get all Intune audit events since the last commit

                "`nProcessing changed files"
                # try to find out who made the change
                foreach ($file in $changedFile) {
                  $resourceId = _getResourceId $file

                  # get author of the resource change
                  if ($resourceId) {
                    "`t- $resourceId ($file)"

                    $resourceModificationEvent = $modificationEvent | ? { $_.Resources.ResourceId -eq $resourceId }

                    # list of change actors
                    $modificationAuthorUPN = @()

                    $resourceModificationEvent.Actor | % {
                      $actor = $_

                      if ($actor.UserPrincipalName) {
                        # modified by user
                        $modificationAuthorUPN += $actor.UserPrincipalName
                      } elseif ($actor.ApplicationDisplayName) {
                        # modified by service principal
                        $modificationAuthorUPN += ($actor.ApplicationDisplayName + " (SP)")
```

```
                    }
                  }

                  $modificationAuthorUPN = $modificationAuthorUPN | select -Unique | Sort-Object
                } else {
                  if ($file -like "*/Assignment Report/report.json") {
                    # assignment report has no ID because it is generated by IntuneCD
                  } elseif ($file -like "*/Managed Google Play/*" -or $file -like "*Device Management Settings/settings.json" -or $file -like "*/Apple Push Notif
                    # IntuneCD don't gather those resources ID
                  } elseif ($file -like "*Device Configurations/mobileconfig/*") {
                    # IntuneCD gather those resources ID in their "parent" JSON, but when DELETE operation occurs, there is no "parent" to gather such data (at l
                    #FIXME zrusit az budu umet tahat ID i pri DELETE operaci
                  } else {
                    throw "Unable to find resourceId in '$file' file name. Pipeline code modification needed, because some changes in IntuneCD were made probably
                  }

                  $modificationAuthorUPN = $null
                }

                if ($modificationAuthorUPN) {
                  "`t`t- changed by: $($modificationAuthorUPN -join ', ')"
                } else {
                  "`t`t- unable to find out who made the change"
                  $modificationAuthorUPN = 'unknown@unknown.com'
                }

                $null = $modificationData.Add(
                    [PSCustomObject]@{
                        resourceId            = $resourceId
                        file                  = Join-Path $root $file
                        modificationAuthorUPN = $modificationAuthorUPN
                    }
                )
              }

              #region commit changes by author(s) who made them
              "`nCommit changes"
              # tip: grouping by created string, otherwise doesn't work correctly (probably because modificationAuthorUPN can contains multiple values)!
              $modificationData | Group-Object { $_.modificationAuthorUPN -join '&'} | % {
                  $modificationAuthorUPN = $_.Group.ModificationAuthorUPN | Select-Object -Unique
                  $modificationAuthorName = $modificationAuthorUPN | % { $_.split('@')[0] }
                  $modifiedFile = $_.Group.File

                  $modifiedFile | % {
                    "`t- Adding $_"
                    $gitResult = _startProcess git -ArgumentList "add `"$_`"" -dontWait -outputErr2Std
                    if ($gitResult -match "^fatal:") {
                      throw $gitResult
                    }
                  }

                  "`t- Setting commit author(s): $($modificationAuthorName -join ', ')"
                  git config user.name ($modificationAuthorName -join ', ')
                  git config user.email ($modificationAuthorUPN -join ', ')

                  # in case of any change in commit name, you have to modify retrieval of the $lastCommitDate too!!!
                  $DATEF = "$(Get-Date $backupStart -f yyyy.MM.dd_HH.mm)"
                  $commitName = "$DATEF` -- $($modificationAuthorName -join ', ')"

                  "`t- Creating commit '$commitName'"
                  $null = _startProcess git -ArgumentList "commit -m `"$commitName`"" -dontWait

                  $unpushedCommit = _startProcess git -ArgumentList "cherry -v origin/main"
                  if ([string]::IsNullOrEmpty($unpushedCommit)) {
                    # no change detected
                    # this shouldn't happen, it means that detection of the changed files isn't working correctly
                    Write-Warning "Nothing to commit?! This shouldn't happen."
                    # set CHANGE_DETECTED pipeline variable
                    echo "##vso[task.setVariable variable=CHANGE_DETECTED;isOutput=true;]0"
                  } else {
                    "`t`t- Commit was created"
                    # save commit date to pipeline variable to use it when creating TAG
                    echo "##vso[task.setVariable variable=COMMIT_DATE;isOutput=true;]$DATEF"
                    # save modification author(s) to use when creating TAG
                    echo "##vso[task.setVariable variable=MODIFICATION_AUTHOR;isOutput=true;]$(($modificationData.modificationAuthorUPN | select -Unique | Sort-O
                  }
              }
              #endregion commit changes by author(s) who made them

              "`nPush changes to upstream"
              $result = _startProcess git -argumentList "push origin HEAD:main" -dontWait -outputErr2Std
            } else {
              "No change detected"
```

```
              # set CHANGE_DETECTED pipeline variable
              echo "##vso[task.setVariable variable=CHANGE_DETECTED;isOutput=true;]0"
            }

    # Create markdown documentation & commit
    # - task: Bash@3
    #   displayName: Generate markdown document & commit
    #   inputs:
    #     targetType: 'inline'
    #     script: |
    #       if [ "$(commitAndsetVariable.CHANGE_DETECTED)" -eq 1 ]
    #       then
    #         INTRO="Intune backup and documentation generated at $(Build.Repository.Uri) <img align=\"right\" width=\"96\" height=\"96\" src=\"./logo.png\">"
    #         IntuneCD-startdocumentation \
    #             --path="$(Build.SourcesDirectory)/prod-backup" \
    #             --outpath="$(Build.SourcesDirectory)/prod-as-built.md" \
    #             --tenantname=$TENANT_NAME \
    #             --intro="$INTRO" \
    #             #--split=Y

    #         # Commit changes and push to repo
    #         DATEF=`date +%Y.%m.%d`
    #         git config user.name $(USER_NAME)
    #         git config user.email $(USER_EMAIL)
    #         git add --all
    #         git commit -m "Intune config as-built $DATEF"
    #         git pull origin main
    #         git push origin HEAD:main
    #       else
    #         echo "no configuration backup change detected in the last commit, documentation will not be created"
    #       fi
    #     workingDirectory: '$(Build.SourcesDirectory)'
    #     failOnStderr: false
    #   env:
    #     TENANT_NAME: $(TENANT_NAME)

  - job: tag
    displayName: Tag repo
    dependsOn: backup_intune
    condition: and(succeeded(), eq(dependencies.backup_intune.outputs['commitAndsetVariable.CHANGE_DETECTED'], 1))
    pool:
      vmImage: ubuntu-latest
    continueOnError: false
    variables:
      COMMIT_DATE: $[ dependencies.backup_intune.outputs['commitAndSetVariable.COMMIT_DATE'] ]
      MODIFICATION_AUTHOR: $[ dependencies.backup_intune.outputs['commitAndSetVariable.MODIFICATION_AUTHOR'] ]
    steps:
      - checkout: self
        persistCredentials: true

      # Set git global settings
      - task: Bash@3
        displayName: Configure Git
        inputs:
          targetType: "inline"
          script: |
            git config --global user.name $(USER_NAME)
            git config --global user.email $(USER_EMAIL)
          workingDirectory: "$(Build.SourcesDirectory)"
          failOnStderr: true

      - task: Bash@3
        displayName: Pull origin
        inputs:
          targetType: "inline"
          script: |
            git pull origin main
          workingDirectory: "$(Build.SourcesDirectory)"
          failOnStderr: false

      - task: PowerShell@2
        displayName: Git tag
        inputs:
          targetType: "inline"
          script: |
            # change in configuration backup folder detected, create TAG
            $DATEF= "$(COMMIT_DATE)"
            "Creating TAG '$DATEF'"
            git tag -a "$DATEF" -m "$DATEF -- Intune configuration snapshot (changes made by: $(MODIFICATION_AUTHOR))"  *> $null
            git push origin "$DATEF" *> $null # even status information goes to stderr :(
          failOnStderr: false
          pwsh: false
          workingDirectory: "$(Build.SourcesDirectory)"
```

```
# Publish PDF & HTML documents as an artifacts
# - job: publish
#   displayName: Publish as-built artifacts
#   dependsOn: tag
#   condition: and(succeeded(), eq(dependencies.backup_intune.outputs['commitAndsetVariable.CHANGE_DETECTED'], 1))
#   pool:
#     vmImage: ubuntu-latest
#   continueOnError: false
#   steps:
#   - checkout: self
#     persistCredentials: true

#   # Install md-to-pdf
#   # https://github.com/simonhaenisch/md-to-pdf
#   - task: Bash@3
#     displayName: Install md-to-pdf
#     inputs:
#       targetType: 'inline'
#       script: |
#         npm i --location=global md-to-pdf
#       workingDirectory: '$(Build.SourcesDirectory)'
#       failOnStderr: true

#   # Convert markdown document to HTML
#   - task: Bash@3
#     displayName: Convert markdown to HTML
#     inputs:
#       targetType: 'inline'
#       script: |
#         cat "$(Build.SourcesDirectory)/prod-as-built.md" | md-to-pdf --config-file "$(Build.SourcesDirectory)/md2pdf/htmlconfig.json" --as-html > "$(Build.Sour
#       workingDirectory: '$(Build.SourcesDirectory)'
#       failOnStderr: false

#   - task: PublishBuildArtifacts@1
#     inputs:
#       pathToPublish: "$(Build.SourcesDirectory)/prod-as-built.html"
#       artifactName: "prod-as-built.html"

#   # Convert markdown document to PDF
#   - task: Bash@3
#     displayName: Convert markdown to PDF
#     inputs:
#       targetType: 'inline'
#       script: |
#         cat "$(Build.SourcesDirectory)/prod-as-built.md" | md-to-pdf --config-file "$(Build.SourcesDirectory)/md2pdf/pdfconfig.json" > "$(Build.SourcesDirectory
#       workingDirectory: '$(Build.SourcesDirectory)'
#       failOnStderr: false

#   - task: PublishBuildArtifacts@1
#     inputs:
#       pathToPublish: "$(Build.SourcesDirectory)/prod-as-built.pdf"
#       artifactName: "prod-as-built.pdf"
```