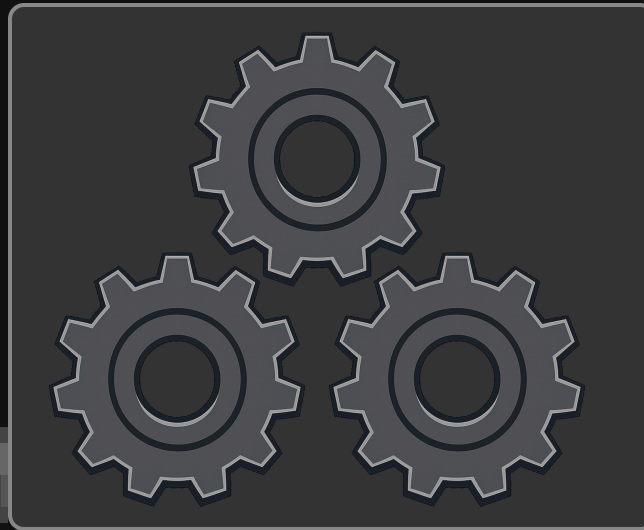
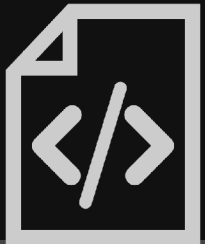


CI/CD mit Microsoft Access

David Korfkamp



Über mich



David Korfkamp

- Seit 2025 im Bereich Access und VBA
- Davor 20 Jahre C#/VB.NET, SQL Server, Analysis Services
- zuletzt TypeScript+React
- Clean Code

Leitfrage

Wie lassen sich Access-Datenbanken mit modernen Softwareentwicklungsverfahren und -ansätzen reproduzierbar entwickeln, testen und ausliefern?

- Änderungen nachvollziehbar machen git
 - Reproduzierbare Builds erzeugen GitLab
 - Tests automatisiert ausführen MSAccessVCS
 - Builds versionieren GitVersion
 - Releases paketieren Access Build Tools
- CI/CD Themen CI/CD



Was ist CI/CD?

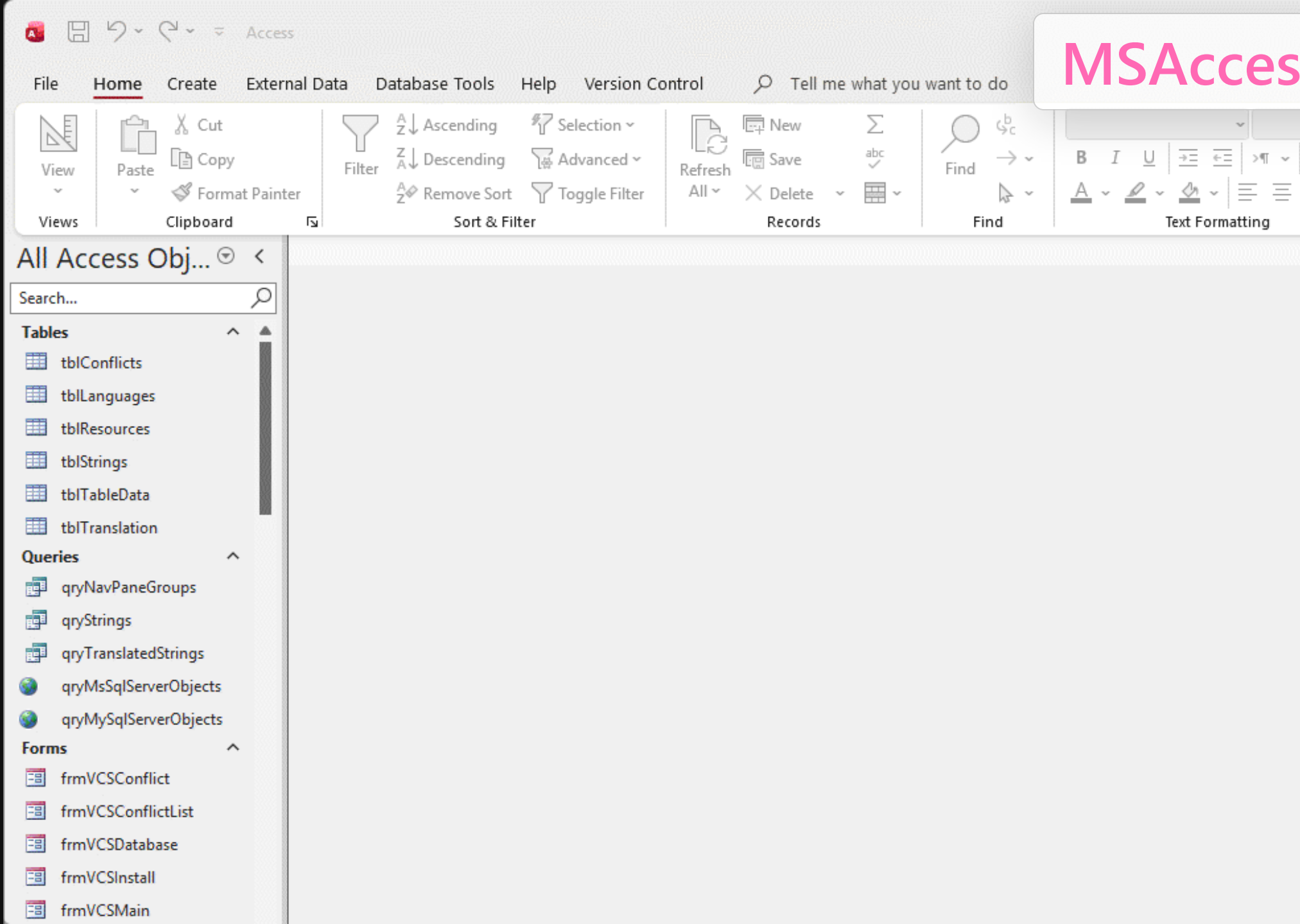
CI/CD ist eine Vorgehensweise, Software zuverlässig und reproduzierbar zu bauen, zu testen und zu deployen

- CI = Continuous Integration
 - Entwickler*innen arbeiten parallel an derselben Anwendung
 - Changes werden regelmäßig zusammengeführt
 - Code wird automatisch kompiliert und getestet
- CD = Continuous Delivery / Deployment
 - wenn alles grün ist, wird eine neue Version vorbereitet
 - diese kann automatisch veröffentlicht oder weitergereicht werden

Access ist mehr als eine Datei

- VBA-Module und Klassen
- Formulare, Berichte und Makros
- Verknüpfte Tabellen, Lokale Tabellen und Abfragen
- Referenzen
- DB-Eigenschaften





MSAccessVCS

Export als Plain Text

```
Private Sub CalculateDiscount()  
    Dim Price As Double: Price = 100  
    <<<<<<< HEAD (Current Change)  
        ' Derive discount  
        Dim Discount As Double: Discount = Price * 0.1  
        MsgBox SFormat("Discount: {0} EUR", Discount)  
    =====  
        ' Derive discount  
        Dim Discount As Double: Discount = Price * 0.15  
        MsgBox SFormat("Discount: {0} EUR", Discount)  
    >>>>>>> feature/new-discount (Incoming Change)  
End Sub
```

DevOps Plattformen



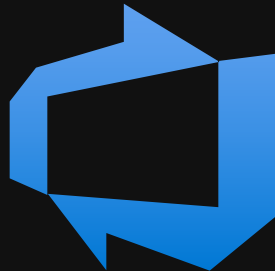
GitHub



TeamCity



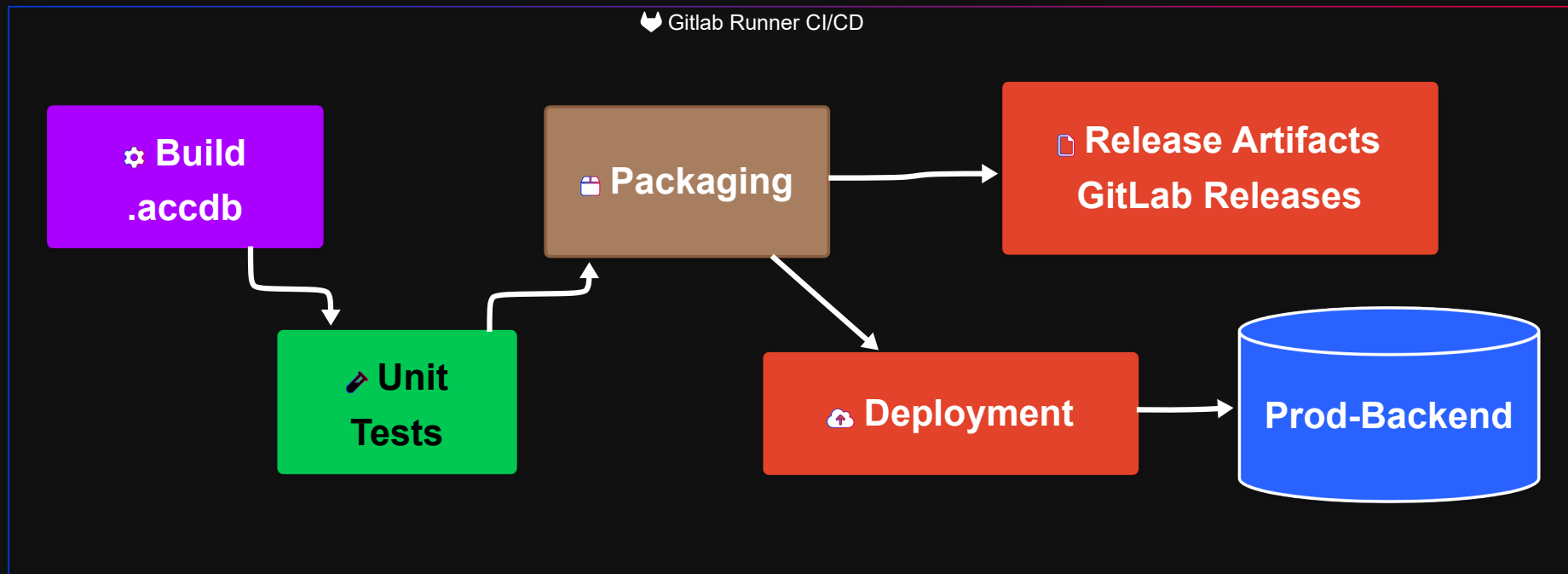
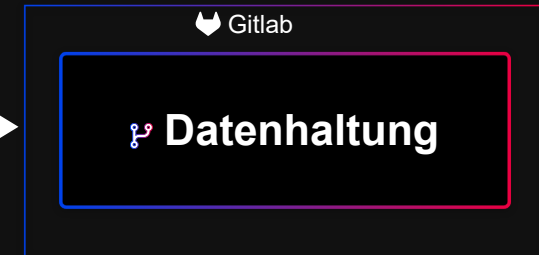
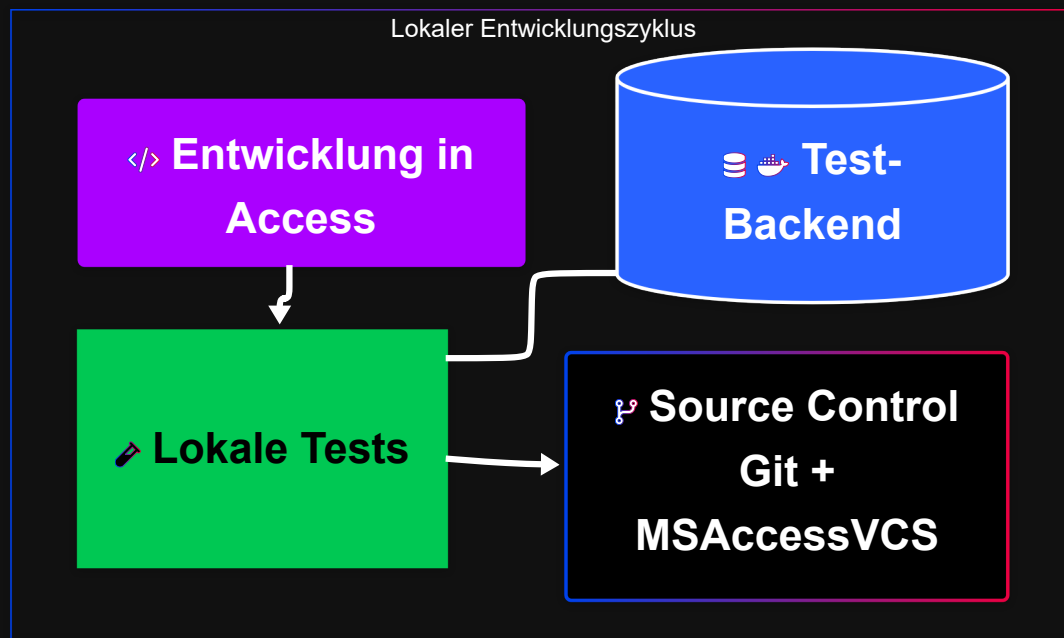
Jenkins



Azure DevOps



GitLab



Definition der Pipeline in .gitlab-ci.yml

```
stages:
  - build
  - test
  - deploy
build job:
  stage: build
  script: ...
  artifacts:
    paths:
      - my.accdb
    name: my.accdb
    expire_in: 1 day
  tags:
    - access
run tests: ...
deploy: ...
```

Access von der Kommandozeile aus builden

```
# Versionsnummer injecten
...

# Pfad zur MSAccessVCS API ermitteln
$vc$Api = ...

# Access Anwendung mit neuer DB öffnen...
$accessApp = ...

# Build im VCS AddIn triggern...
$accessApp.Run($vc$Api, "BuildHeadless", "my.accdb.src")

# Access sicher beenden...
...
```

Versionsnummern

Exkurs: Semantic Versioning

- Standard zur Versionierung von Software-Releases
- Format: *MAJOR.MINOR.PATCH* (z.B 2.3.5)
- *MAJOR*: Breaking Changes, nicht abwärtskompatibel
- *MINOR*: Neue Features, abwärtskompatibel
- *PATCH*: Fehlerbehebungen und kleine Verbesserungen, abwärtskompatibel

GitVersion

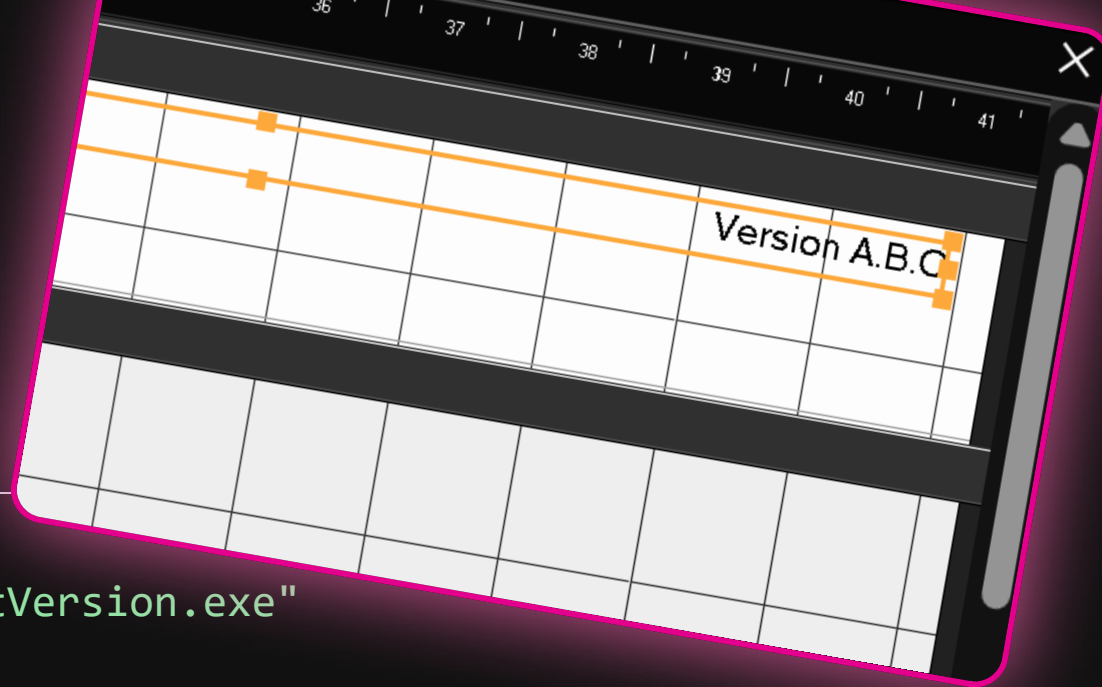
| <https://github.com/GitTools/GitVersion>

- 37e4bef (HEAD -> main) prepare release
- 2508980 add people directory
- f0b0ae3 chore: refactor
- 734c1c8 (tag: v2.0.0) new version
- 2488e9c fix export
- a8c180f add .gitignore
- 0519ac6 (tag: v1.0.0) init

v2.0.1.alpha.3
v2.0.1.alpha.2
v2.0.1.alpha.1
v2.0.0
v1.0.1.alpha.2
v1.0.1.alpha.1
v1.0.0

```
{
  "AssemblySemFileVer": "10.7.0.0",
  "BranchName": "develop",
  "CommitDate": "2026-09-04",
  "FullBuildMetaData": "Branch.develop.Sha.b5eded68...",
  "FullSemVer": "10.7.0-alpha.1",
  "Major": 10,
  "Minor": 7,
  "Patch": 0,
  "PreReleaseLabel": "alpha",
  "PreReleaseTag": "alpha.1",
  "PreReleaseTagWithDash": "-alpha.1",
  "SemVer": "10.7.0-alpha.1",
  "Sha": "b5eded68fda1f7270fb24a5766cda6b8b2d1ee14",
  "VersionSourceSemVer": "10.6.0",
  ...
}
```

GitVersion



```
# VOR dem Build....
$gitVersionTool = "$env:LOCALAPPDATA\gitVersion\gitVersion.exe"

$gitVersion = & $gitversionTool | ConvertFrom-Json
$version = $gitVersion.SemVer
$date = ([datetime]::Parse($gitVersion.CommitDate)).ToString("dd.MM.yyyy")

$versionFile = "my.accdb.src\forms\frmMain.bas"
(Get-Content $versionFile) `
    -replace 'Caption\s*=\s*"Version A\.B\.C"', "Caption =`"Version $version ($date)`" | `
    Set-Content $versionFile
```

Access von der Kommandozeile aus builden

```
#  Versionsnummer injecten
...

# Pfad zur MSAccessVCS API ermitteln
$vc$Api = ...

# Access Anwendung mit neuer DB öffnen...
$accessApp = ...

# Build im VCS AddIn triggern...
$accessApp.Run($vc$Api, "BuildHeadless", "my.accdb.src")

# Access sicher beenden...
...
```

Pfad zur MSAccessVCS API ermitteln

```
$vcsAddinFolder = (Join-Path $env:AppData "MSAccessVCS")  
$candidate = (Join-Path $vcsAddinFolder "Version Control.API")
```

Pfad zur MSAccessVCS API ermitteln

```
OperationStopped: Die Prozedur  
'C:\users\david.korfkamp\AppData\Roaming\MSAccessVCS\Version Control.API'  
wurde von Microsoft Access nicht gefunden.
```

Pfad zur MSAccessVCS API ermitteln

```
$vcsAddinFolder = (Join-Path $env:AppData "MSAccessVCS")
$candidate = (Join-Path $vcsAddinFolder "Version Control.accda")

# Download + Install (vereinfacht)
if (Test-Path $candidate) {
    return "$vcsAddinFolder\Version Control.API"
}
Invoke-WebRequest -Uri $sourceUrl -OutFile $zip
Expand-Archive -Path $zip -DestinationPath $addinFolder -Force

$process = Start-Process -FilePath "msaccess.exe" `
    -ArgumentList "" "$tmpCandidate" /cmd INSTALL SILENT `
    -WindowStyle 'Hidden' `
    -PassThru
$process.WaitForExit()
```

Access von der Kommandozeile aus builden

```
#  Versionsnummer injecten
...

#  Pfad zur MSAccessVCS API ermitteln
$vc$Api = ...

# Access Anwendung mit neuer DB öffnen...
$accessApp = ...

# Build im VCS AddIn triggern...
$accessApp.Run($vc$Api, "BuildHeadless", "my.accdb.src")

# Access sicher beenden...
...
```

Access öffnen

```
$accessApp = New-Object -ComObject "Access.Application"  
$accessApp.OpenCurrentDatabase($AccessDbPath)
```

Anmeldung



LOCALDB

Login

Kennwort

Abbrechen

Login

Einstellungen für Formular-Kaster

☒ Raster anzeigen

Rastereinheiten: Punkt

Breite: 6

Höhe: 6

☒ Am Raster ausrichten

☒ QuickInfo anzeigen

☒ Ausblenden des Projekts schließt Fenster

Bearbeiten und Fortfahren

☐ Benachrichtigen vor Zustandsände

Unterbrechen bei Fehlern

☒ Bei jedem Fehler

☐ In Klassenmodul

☐ Bei nicht verarbeiteten Fehlern

Kompilieren

☒ Bei Bedarf

☒ Im Hintergrund

OK

Abbrechen

Hilfe

Oracle ODBC Driver Connect

Service Name

ORA_LOCAL

User Name

david.korfkamp

Password

OK

Cancel

About...

Microsoft Visual Basic

Laufzeitfehler '2448':

Sie können diesem Objekt keinen Wert zuweisen.

Fortfahren

Beenden

Debuggen

Hilfe

Kontrollierter Access-DB Start

Variante 1: **SHIFT** gedrückt halten simulieren

funktioniert nur, wenn der Gitlab Runner NICHT als Service ausgeführt wird



```
public class KeyboardUtils {  
    ...  
    public static void HoldShift() { keybd_event(VK_SHIFT, 0, KEYEVENTF_KEYDOWN, 0); }  
    public static void ReleaseShift() { keybd_event(VK_SHIFT, 0, KEYEVENTF_KEYUP, 0); }  
}
```

```
[KeyboardUtils]::HoldShift()  
$accessApp.OpenCurrentDatabase($AccessDbPath)  
[KeyboardUtils]::ReleaseShift()
```

Kontrollierter Access-DB Start

Variante 2: AutoExec Makro + Umgebungsvariable

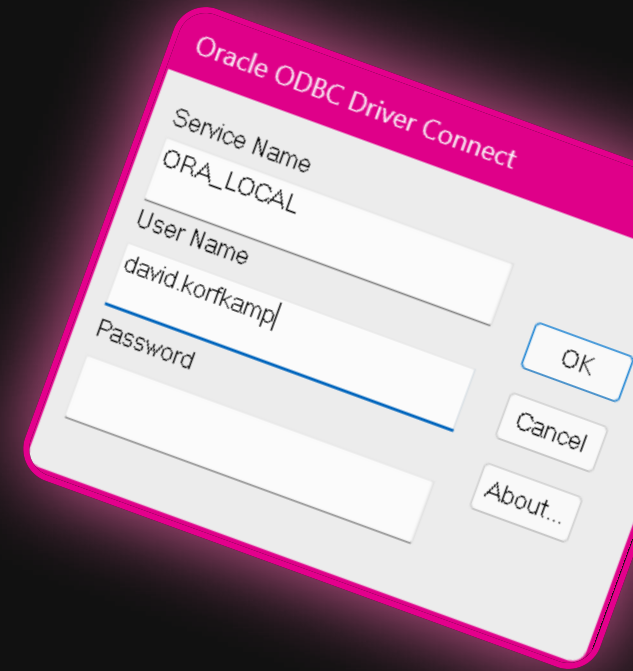
funktioniert nur, wenn kein StartUp Form festgelegt ist. Kann stattdessen über AutoExec Makro nachgebaut werden.



```
Public Function BootstrapApp()  
    If Environ$("ACCESS_NO_AUTOEXEC")="1" Then Exit Function  
    If Not Environment.Initialize Then Exit Function  
  
    DoCmd.OpenForm "frmMain", acNormal, WindowMode:=acWindowNormal  
  
    ...  
End Function
```

Backend Zugriffe steuern

- ODBC-Login-Dialoge blockieren ebenfalls
- Dialog wird getriggert, wenn
 - Form geöffnet wird, das Daten aus ODBC-DB zieht
 - Code ausgeführt wird, der auf ODBC-DB zugreift
 - ODBC-gebundene Objekte (Passthrough-Queries, Linked Tables) refreshed oder neu angelegt werden
 - MSAccessVCS macht das während des Builds



Backend Zugriffe steuern

Lösungsansätze

- MSAccessVCS `.env` Dateien nutzen
 - Arbeitsverzeichnis wird in Pipeline vor jedem Build geleert 🤔
 - Kennwort wird in Access-DB gespeichert 🙅
- Kennwort in ODBC DSN auf Gitlab-Runnern speichern
 - muss auf allen Gitlab-Runnern eingerichtet werden
 - geht nicht bei allen ODBC-Treibern (Oracle z.B.) 🙅
- Kennwort in ODBC-Connection String in Access speichern
 - Kennwort wird in Klartext in git gespeichert 🙅



DB Connection Cache Warm-up

Temporäres Modul in Ziel-DB anlegen

Oder einfach:
`$accessApp.DBEngine.Workspaces(0).OpenDatabase($connStr)` 🧑

```
Public Sub ConnectionCacheWarmup(ByVal pDsn As String, ByVal pUser As String, ByVal pPass As String)
    Dim connStr As String: connStr = "ODBC;DSN=" & pDsn & ";UID=" & pUser & ";PWD=" & pPass
    DBEngine.Workspaces(0).OpenDatabase "", dbDriverCompleteRequired, False, connStr
End Sub
```

In GitLab Pipeline aufrufen

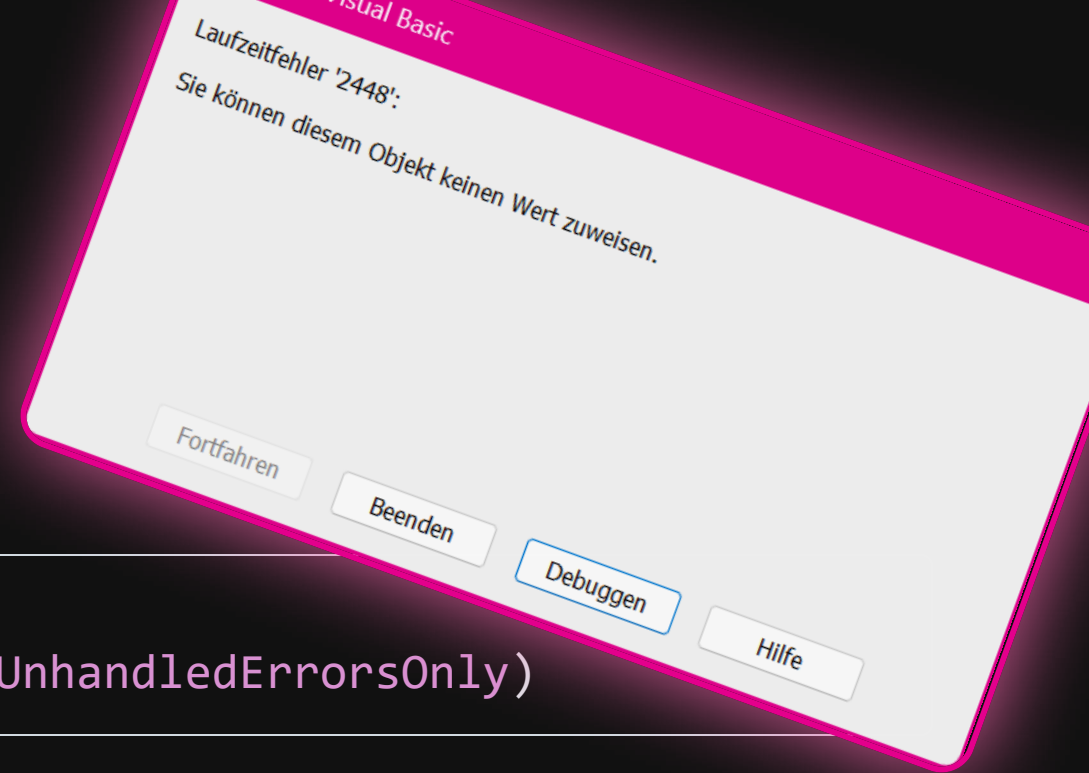
```
$accessApp.Run("ConnectionCacheWarmup", $env:GL_DSN, $env:GL_USER, $env:GL_PASS)
```

Error Trapping kontrollieren

```
$breakOnUnhandledErrorsOnly = 2  
$accessApp.SetOption("Error Trapping", $breakOnUnhandledErrorsOnly)
```

In der Entwicklerversion von MSAccessVCS wird diese Option mittlerweile korrekt gesetzt.

Für andere Build-Verfahren aber möglicherweise relevant.



Access von der Kommandozeile aus builden

```
# ☒ Versionsnummer injecten
...

# ☒ Pfad zur MSAccessVCS API ermitteln
$vc$Api = ...

# ☒ Access Anwendung mit neuer DB öffnen...
$accessApp = ...

# ☒ Build im VCS AddIn triggern...
$accessApp.Run($vc$Api, "BuildHeadless", "my.accdb.src")

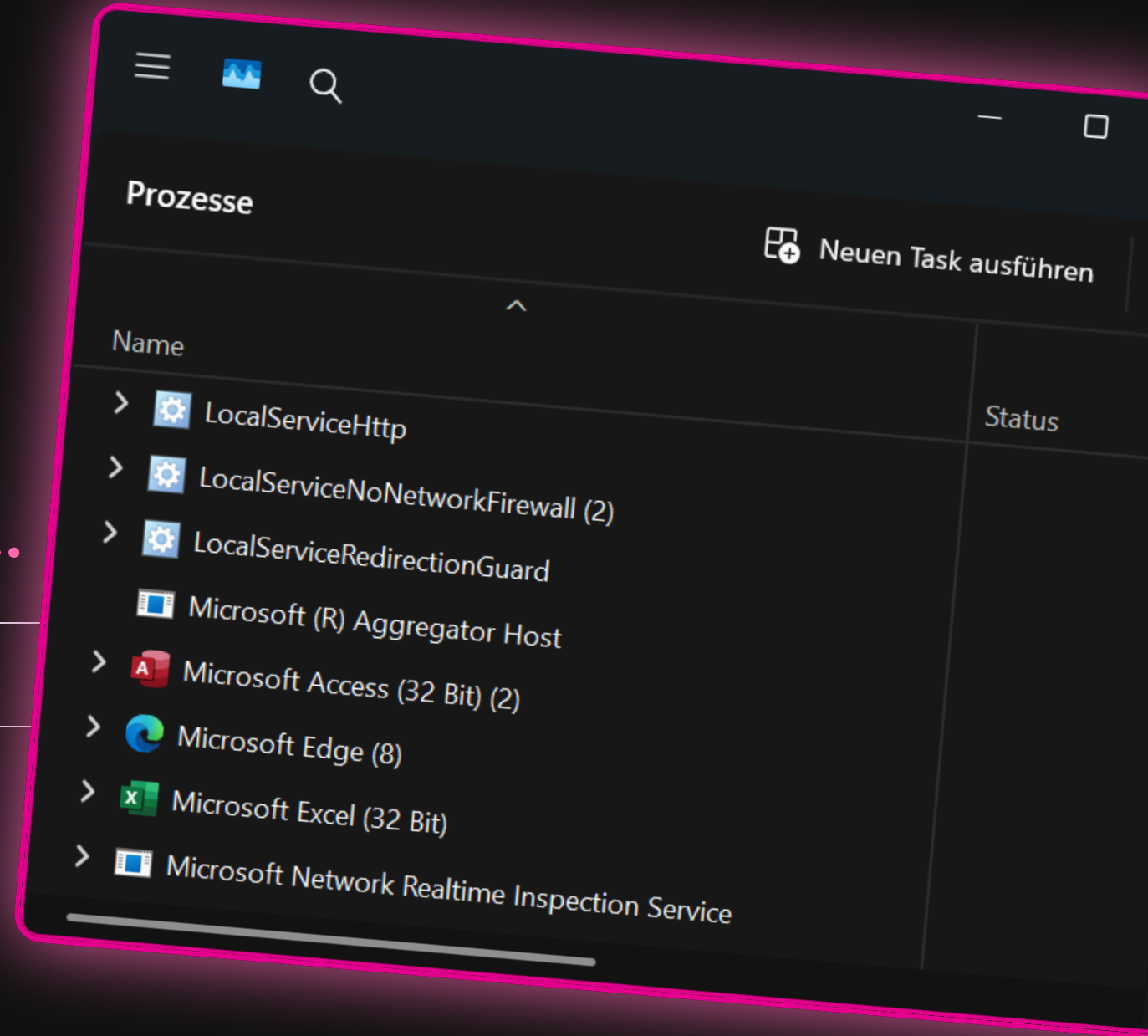
# Access sicher beenden...
...
```

Jetzt nur noch beenden...

```
$accessApp.Quit()
```

Jetzt nur noch beenden...

```
$accessApp.Quit()
```



Jetzt nur noch beenden... (2)

```
$AccessApp.Quit()  
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($AccessApp) | Out-Null  
Remove-Variable AccessApp -ErrorAction SilentlyContinue  
[GC]::Collect()  
[GC]::WaitForPendingFinalizers()
```

Jetzt nur noch beenden... (3)

```
build job:
  stage: build
  script:
    - Build-Accdb -SourceFolder "my.accdb.src"
  after_script:
    - |
      Get-Process MSACCESS -ErrorAction SilentlyContinue | `
      Stop-Process -Force -ErrorAction SilentlyContinue
```

Access von der Kommandozeile aus builden

```
# ☒ Versionsnummer injecten
...

# ☒ Pfad zur MSAccessVCS API ermitteln
$vc$Api = ...

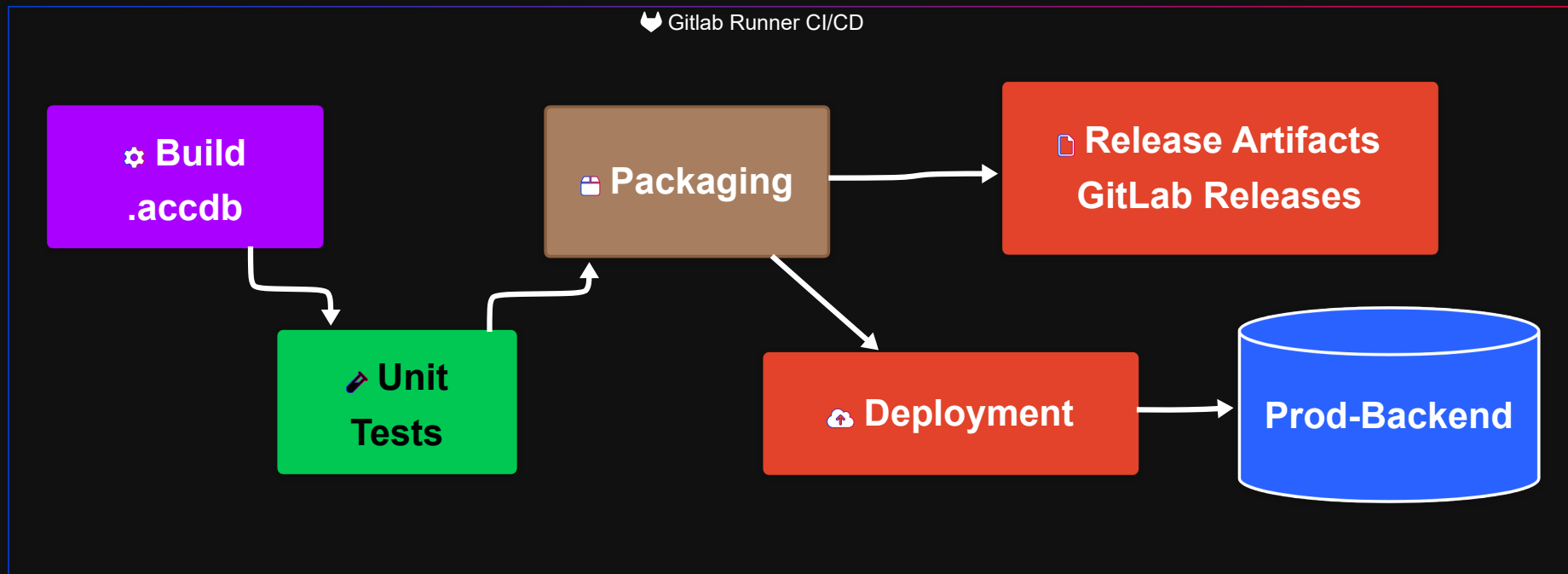
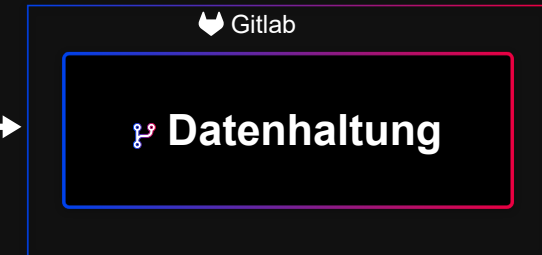
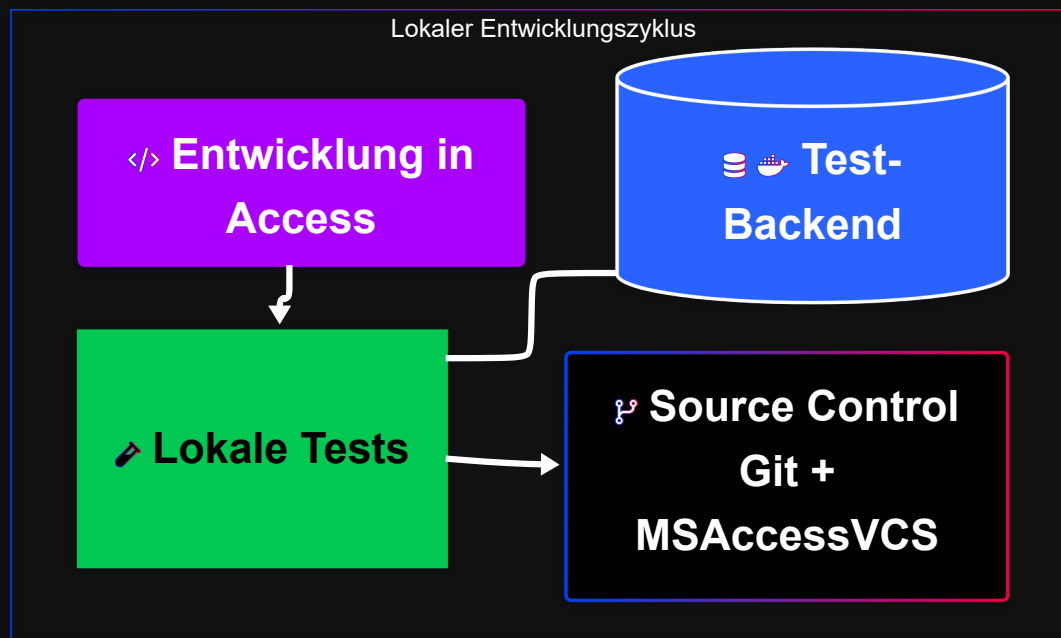
# ☒ Access Anwendung mit neuer DB öffnen...
$accessApp = ...

# ☒ Build im VCS AddIn triggern...
$accessApp.Run($vc$Api, "BuildHeadless", "my.accdb.src")

# ☒ Access sicher beenden...
...
```

Unit Tests





Unit Test Plattformen



RubberDuck

AccUnit

MSAccessVCS

VBUNIT3

vbUnit3

AccessUnit(Fork)

Unit Tests schreiben

```
'@Folder("Tests")

...

Public Sub ParseJsonTest()
    'Arrange
    Dim expected As String: expected = "test@test.de"
    Dim actual As Scripting.Dictionary

    'Act
    Set actual = JsonUtils.Deserialize("{\"email\":\"test@test.de\"}")

    'Assert
    ' NICHT modAssert.TestAssert
    TestAssert expected = actual.item("email")
End Sub
```

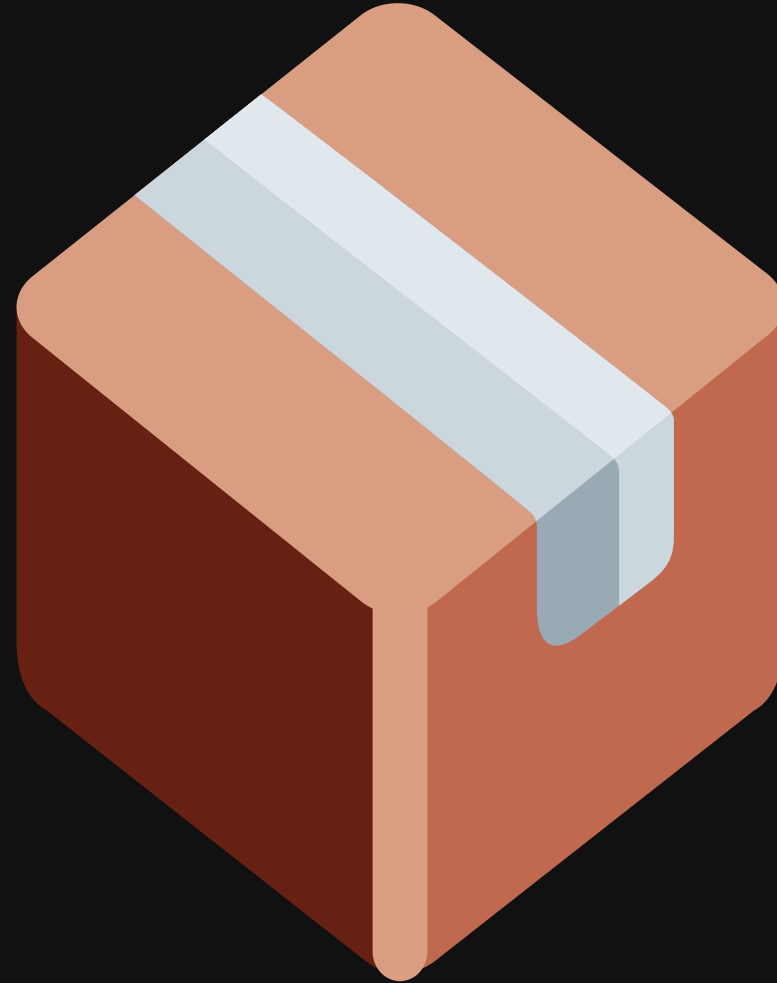
Unit Tests in Pipeline ausführen

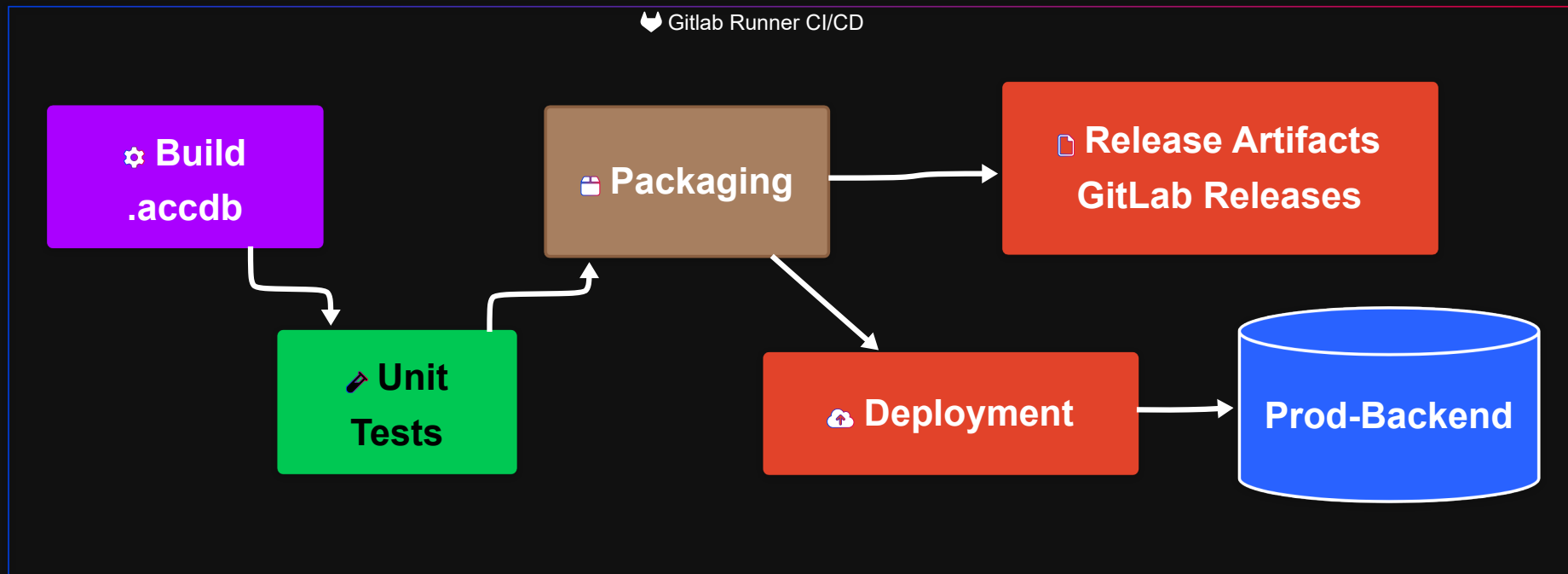
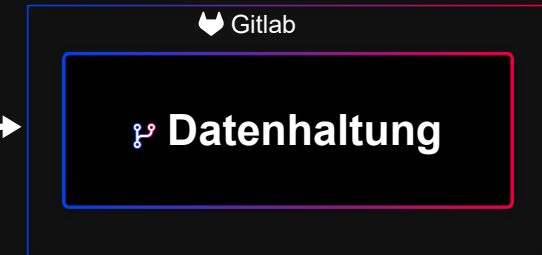
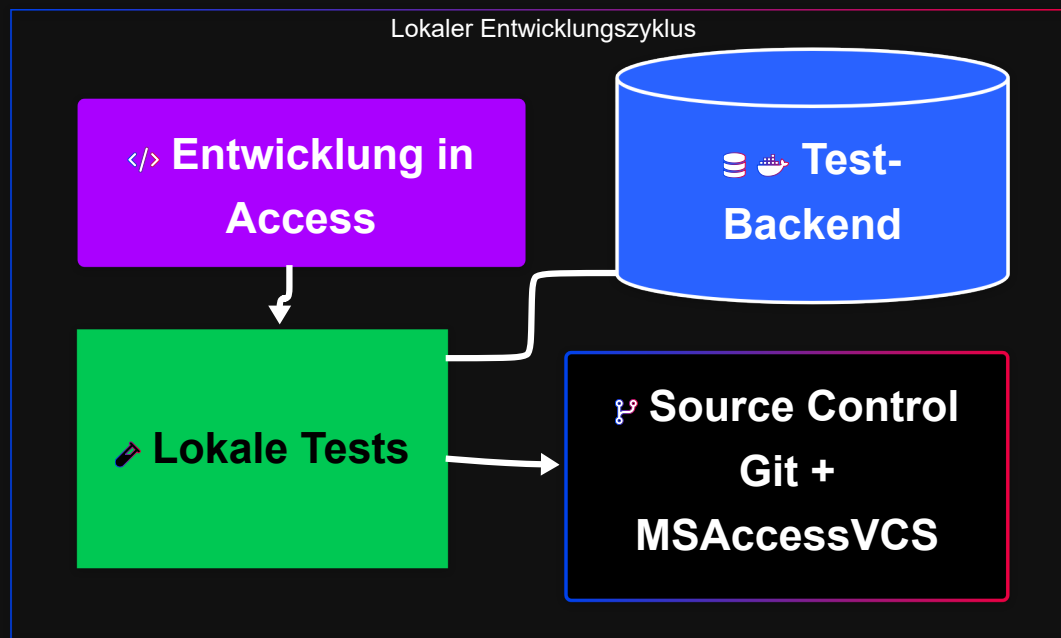
```
test job:
  dependencies:
    - build job
  stage: test
  script:
    - |
      $accessApp = Open-AccessWithoutStartupCommands -AccessDBPath "my.accdb"

      $result = $accessApp.Run(GetVcsApi, "RunTestsHeadless") | ConvertFrom-Json
      if (-not $result.allPassed) {
        $failCount = $result.summary.failed + $result.summary.errorred
        Write-Error "$failCount unit tests from $AccessDBPath finished with errors."
      }
  artifacts:
    reports:
      junit: test_results.xml
```

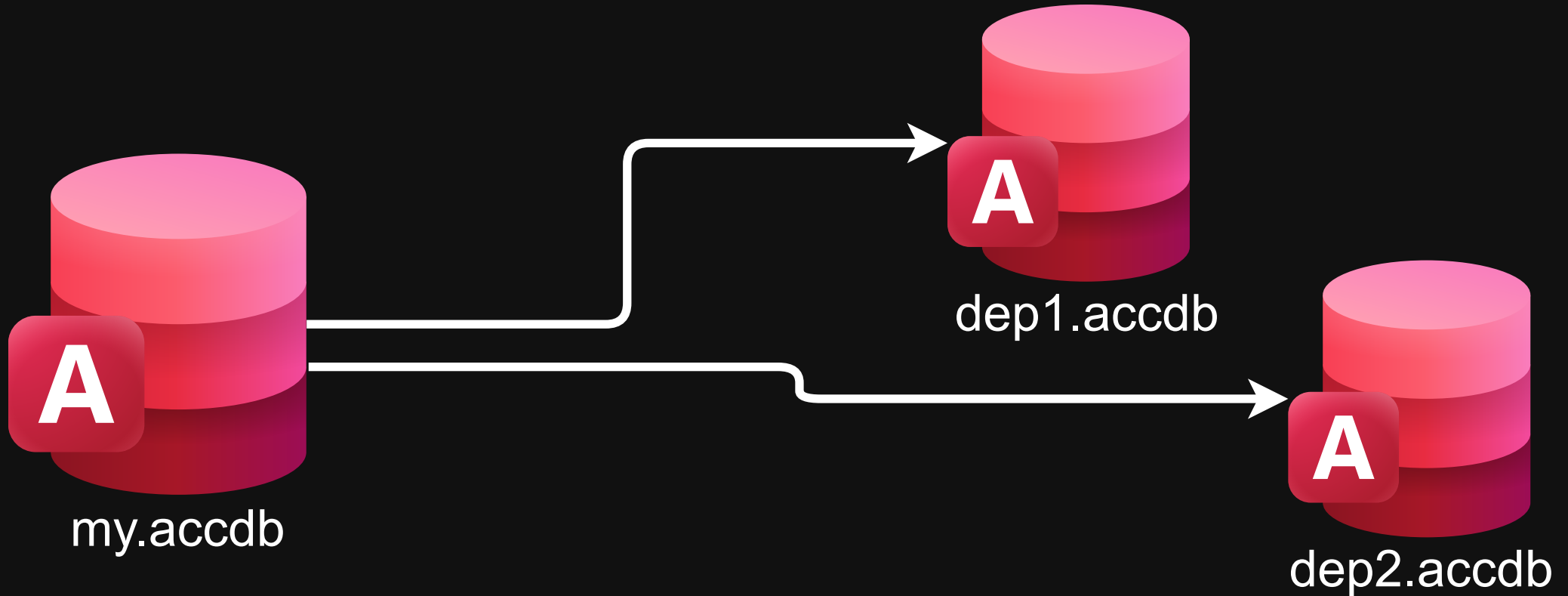
Tests	CollExcept_With_Larger_Coll_Returns_Empty			0.00ms	View details
Tests	Abc_StartsWith_a_is_false_Test			0.00ms	View details
Tests	CollExcept_With_Empty_Coll_Returns_Same			0.00ms	View details
Tests	CollContains_On_Filled_With_Same_Numbers_Returns_True			0.00ms	View details
Tests	CredentialManager_ReadUser			0.00ms	View details
Tests	MailCompose_with_only_one_recipient			4.00ms	View details

Packaging





Packaging und Abhängigkeiten



Problem

- ~~Referenzen sind immer absolute Pfade~~

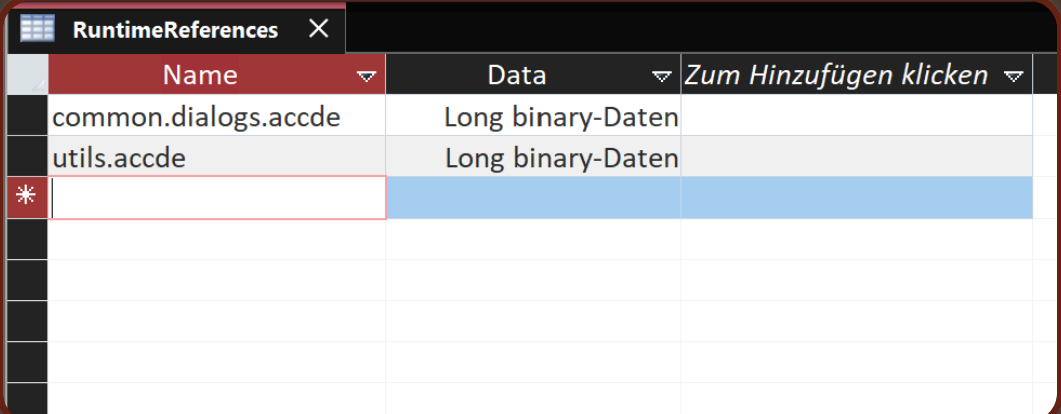
Hinweis: Das ursprünglich beobachtete Verhalten ließ sich mit aktuellem Access 365 nicht mehr reproduzieren 🤖. Packaging vereinfacht die Verteilung dennoch erheblich, da nur eine Datei kopiert werden muss.

Lösungsansätze

- Auf allen Clients und Dev-PCs Verzeichnis unter gleichem Namen anlegen
 - Abhängigkeiten darin referenzieren
- zentrales Netzlaufwerk
 - für *READONLY* Referenzen möglich
 - funktioniert nur innerhalb eines Netzwerks

Referenzen in Access-DB speichern

- Beim Packaging: Referenzen als BLOB in Access-Tabelle speichern
- Beim Start: BLOB in Temp-Verzeichnis entpacken und referenzieren, anschließend AutoExec-Makro bzw. Startformular laden
- Veröffentlichung nur als `.accdb` (kein `.accde` Support)



	Name	Data	Zum Hinzufügen klicken
	common.dialogs.accde	Long binary-Daten	
	utils.accde	Long binary-Daten	
*			

```
Set rs = db.OpenRecordset( _  
    "SELECT Name, Data FROM RuntimeReferences", _  
    dbOpenDynaset)  
While Not rs.EOF  
    Dim tmpRefPath As String  
    tmpRefPath = GetTempRefPath(rs!Name)  
    WriteToDisk tmpRefPath, rs!Data  
    AddReference tmpRefPath  
    rs.MoveNext  
End While  
  
'AutoExec Macro ausführen  
ContinueStartup
```



Access Build Tools

Überblick über die Cmdlets

```
Build-Accdb # Versionsnummer injizieren, Accdb builden  
Convert-ToAccde # Kompilieren  
Invoke-UnitTests # Unittests ausführen und Ergebnis als JUnit XML speichern  
Publish-Accdb # Referenzierte accdb's embedden und Startupcode injizieren
```

Zusammenfassung

- MSAccessVCS-Add-in macht Access-Objekte versionierbar
- Access Build Tools kapseln Versionierung, Build, Tests und Packaging
- GitLab Runner führen dieselben Schritte reproduzierbar aus
- eine gepackte Version kann überall, wo Access Runtime vorhanden ist, ausgeführt werden

Vielen Dank für die Aufmerksamkeit

Fragen und Diskussion

david.korfkamp@hotmail.de

Referenzen

- MS Access VCS Add-in: <https://github.com/joyfullservice/msaccess-vcs-addin>
- GitVersion: <https://github.com/GitTools/GitVersion>
- RubberduckVBA: <https://rubberduckvba.ca/>
- vbUnit3: <https://vbunit.org>
- AccUnit: <https://github.com/AccessCodeLib/AccUnit>
- AccessUnit(Fork): <https://codekabinett.com/rdumps.php?Lang=2&targetDoc=accessunitfork-access-vba-unit-test>
- Access Build Tools: <https://github.com/lagwagon667/Access-Build-Tools>

Bildquellen

- Code Document Bild auf 1. Folie: [Flaticon.com](https://flaticon.com)
- MSAccessVCS Demo GIF: <https://github.com/joyfullservice/msaccess-vcs-addin>
- Unit Test Bild mit Reagenzgläsern: KI
- Rubberduck Logo auf Unit Test Plattformen: <https://rubberduckvba.ca/>
- vbUnit3 Logo auf Unit Test Plattformen: <https://vbunit.org>
- alle anderen: eigene Aufnahmen oder ohne Attribution