BC Docker Sandbox und VS Code

Wie erstelle ich unter Windows eine BC Sandbox in Docker? Was ist notwendig, damit ich mit VS Code eine Extension in AL schreiben kann? Wie veröffentliche ich eine App?

Docker

Vorbereitung des Host

Während der Installation ist womöglich der ein oder andere Neustart des Rechners notwendig.

Eine PowerShell Konsole als Admin ausführen und das folgende Script ausführen, um Hyper-V zu aktivieren:

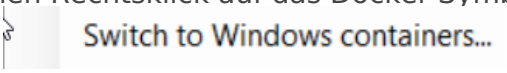
```
Enable-WindowsOptionalFeature -Online -FeatureName $("Microsoft-Hyper-V", "Containers") -All
```

Sofern noch nicht geschehen, muss nun noch einmalig das Modul **BcContainerHelper** installiert werden. In der gleichen Konsole die folgende Zeile ausführen:

```
Install-Module BcContainerHelper -force
```

Docker Installation

Die aktuellste Docker Version **hier** herunterladen und installieren. Spätestens nach dieser Installation ist ein Neustart notwendig.

Nach der Installation muss Docker für die Verwendung von Windows Containern konfiguriert werden. Dafür einen Rechtsklick auf das Docker Symbol (Taskleiste)  ausführen und anschließend auf 

klicken.

Erstellung eines BC Containers

Mit Hilfe der folgenden zwei PowerShell Scripts lassen sich jeweils BC Sandboxes erstellen. **BC 19 On-Prem** erstellt einen Container mit der aktuellsten Version, **BC specific version On-Prem** lädt das Image herunter, welches der im Script angegebenen Version am nächsten ist (hier: 17.11.30469.0).

Kurz nach dem Start des Scripts wird nach einem Benutzernamen und einem Passwort verlangt - diese Daten sind frei wählbar und dienen der Authentifizierung des Benutzers am BC Server.

Im Anschluss der Installation ist BC unter der Adresse http://*$containerName*/BC zu erreichen - also für **BC 19 On-Prem** z.B. http://bc19onprem/BC

BC 19 On-Prem

```

```

```
$containerName = 'bc19onprem'
$credential = Get-Credential -Message 'Using UserPassword authentication. Please enter credentials for the container.'
$auth = 'UserPassword'
$artifactUrl = Get-BcArtifactUrl -type 'OnPrem' -country 'de' -select 'Latest'
New-BcContainer `
    -accept_eula `
    -containerName $containerName `
    -credential $credential `
    -auth $auth `
    -artifactUrl $artifactUrl `
    -imageName 'bc19image' `
    -memoryLimit 8G `
    -isolation hyperv `
    -vsixFile (Get-LatestAlLanguageExtensionUrl) `
    -updateHosts
```

BC specific version On-Prem

```
$containerName = 'bc17onprem'
$credential = Get-Credential -Message 'Using UserPassword authentication. Please enter credentials for the container.'
$auth = 'UserPassword'
$artifactUrl = Get-BcArtifactUrl -type 'OnPrem' -version '17.11.30469.0' -country 'de' -select 'Closest'
New-BcContainer `
    -accept_eula `
    -containerName $containerName `
    -credential $credential `
    -auth $auth `
    -artifactUrl $artifactUrl `
    -memoryLimit 8G `
    -isolation hyperv `
    -vsixFile (Get-LatestAlLanguageExtensionUrl) `
    -updateHosts
```

BC specific version On-Prem (SSL)

```
$containerName = 'bc18onprem'
$credential = Get-Credential -Message 'Using UserPassword authentication. Please enter
```

```
credentials for the container.'
$auth = 'UserPassword'
$artifactUrl = Get-BcArtifactUrl -type 'OnPrem' -version '18.6.30510.0' -country 'de' -select
'Closest'
New-BcContainer `
    -accept_eula `
    -containerName $containerName `
    -credential $credential `
    -auth $auth `
    -artifactUrl $artifactUrl `
    -imageName 'bc18image' `
    -usessl -installCertificateOnHost `
    -memoryLimit 8G `
    -isolation hyperv `
    -vsixFile (Get-LatestAlLanguageExtensionUrl) `
    -updateHosts
```

VS Code und App Veröffentlichung

VS Code Installation

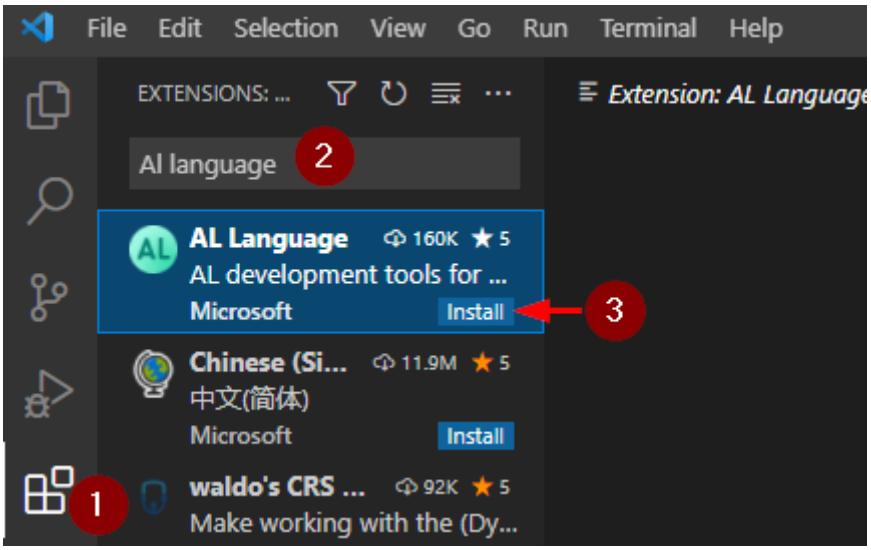
Die aktuellste Version von Visual Studio Code **hier** herunterladen und anschließend installieren.

Benötigte und nützliche Extensions für VS Code

AL Language

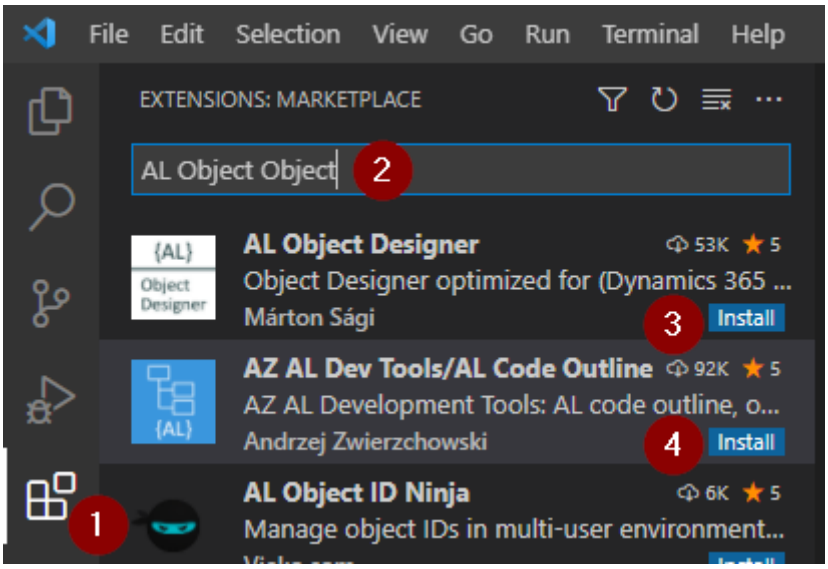
Nach dem ersten Start von Visual Studio Code ist es notwendig, die AL Language Erweiterung zu installieren. Dazu bitte die folgenden Schritte in VC Code ausführen:

1. Extension Menü öffnen
2. Nach "AL Language" suchen
3. Und das entsprechende Paket installieren



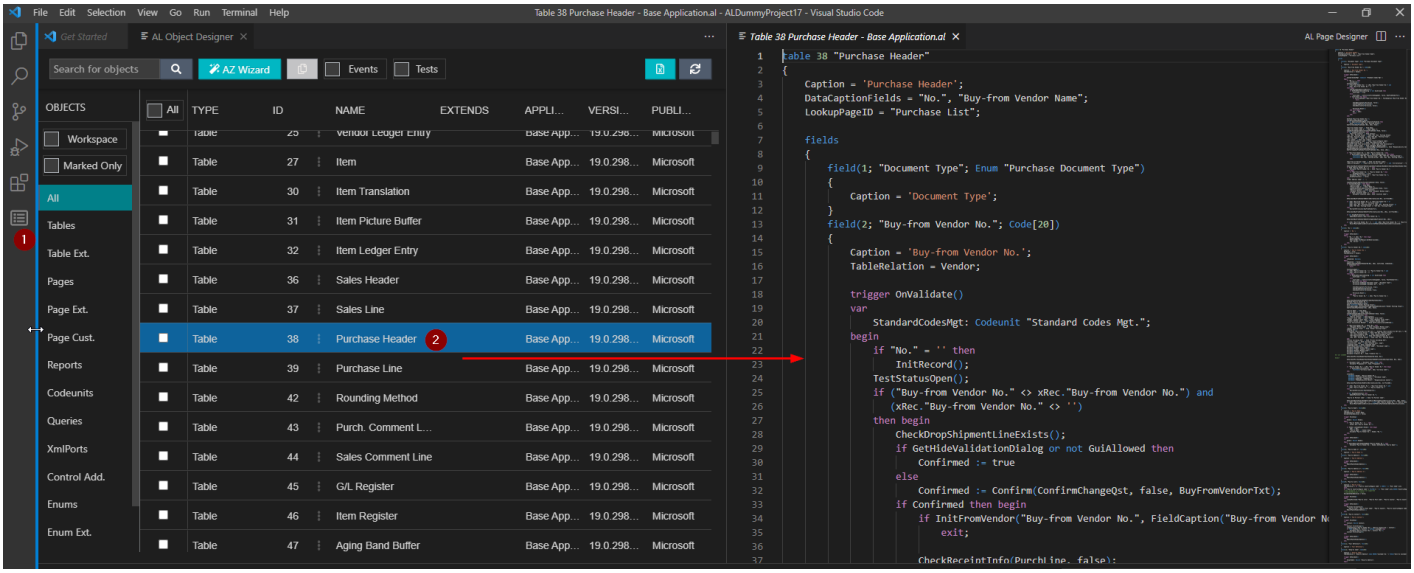
AL Object Designer und AL Dev Tools

Weitere, praktische Erweiterungen sind z.B. der **AL Object Designer** von Márton Sági und die **AL Dev Tools** von Andrzej Zwierzchowski.



AL Object Designer

Mit dieser Erweiterung lassen sich, unter anderem, alle Objekte des Projektes auflisten und der entsprechende Code einsehen.



AL Dev Tools

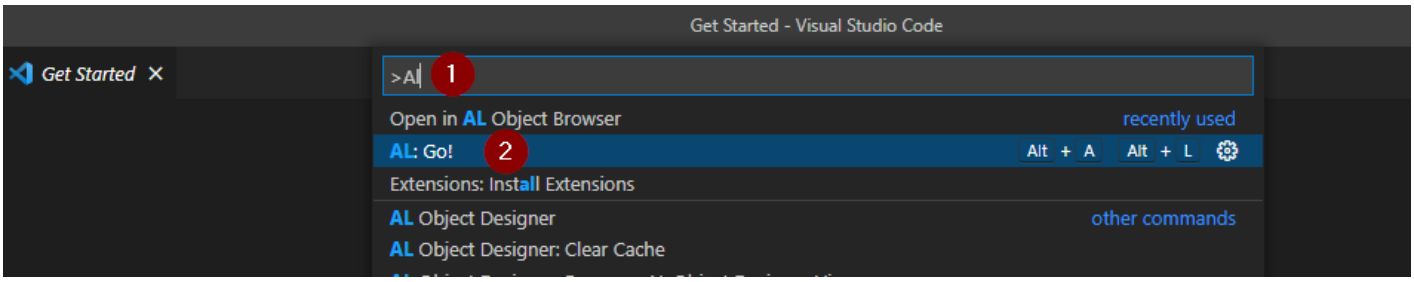
Diese Erweiterung bietet, unter anderem, verschiedene Wizards und andere praktische Tools.

Erstellen und veröffentlichen einer ersten App aus VS Code heraus

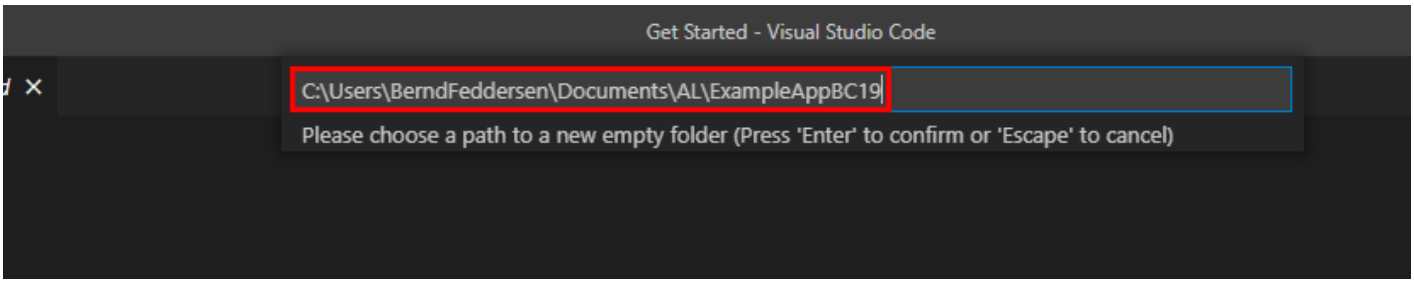
Im Folgenden wird die Erstellung eines neuen Beispielprojektes in Visual Studio Code beschrieben sowie die Veröffentlichung der daraus resultierenden BC App gezeigt. Wir benutzen hierfür die aktuellste BC Version (Runtime Version 8, bzw. Anwendungsversion 19).

Projektinitialisierung

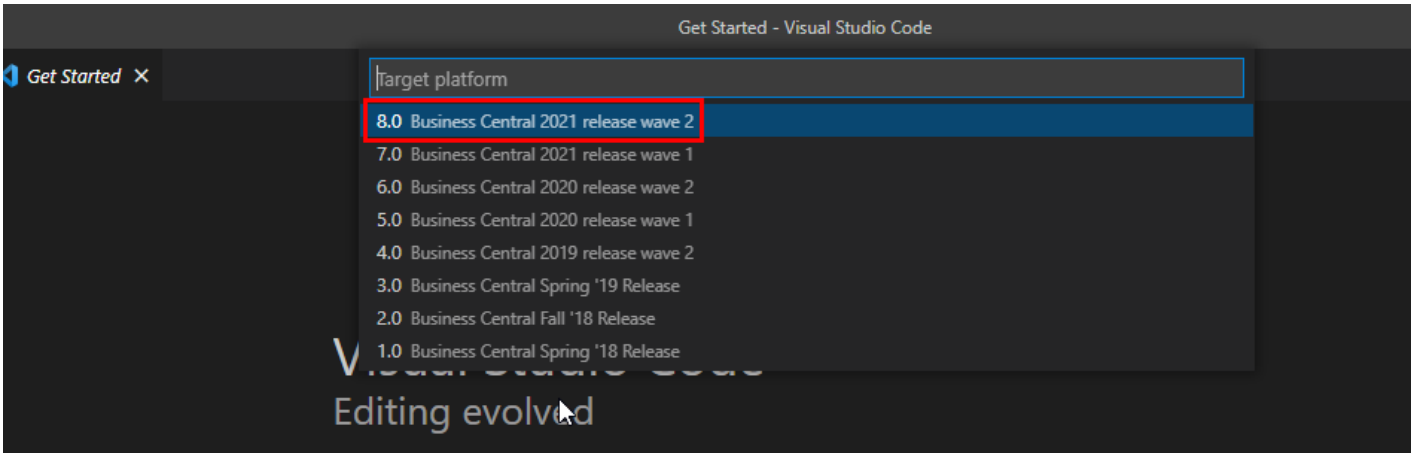
In VS Code F1 oder STRG+Shift+P drücken, um die Befehlszeile zu öffnen. Anschließend das Kommando **AL: Go!** ausführen.



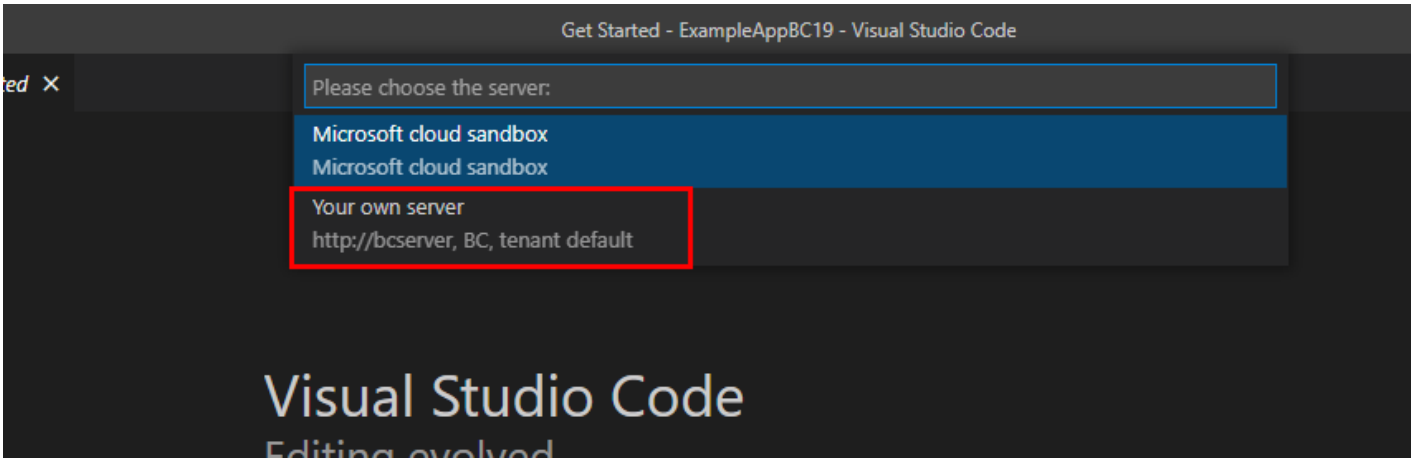
Daraufhin wird der Pfad zu einem leeren Verzeichnis benötigt. Der Ordner (hier: *ExampleAppBC19*) muss im Vorfeld nicht erstellt werden.



Damit VS Code die Konfigurationsdatei **app.json** mit den korrekten Parametern erstellt, muss nun die Business Central Version ausgewählt werden, für die eine App entwickelt werden soll. In diesem Fall ist es die *Version 8.0 (Runtime)*, bzw. *Anwendungsversion 19.0.0.0*.



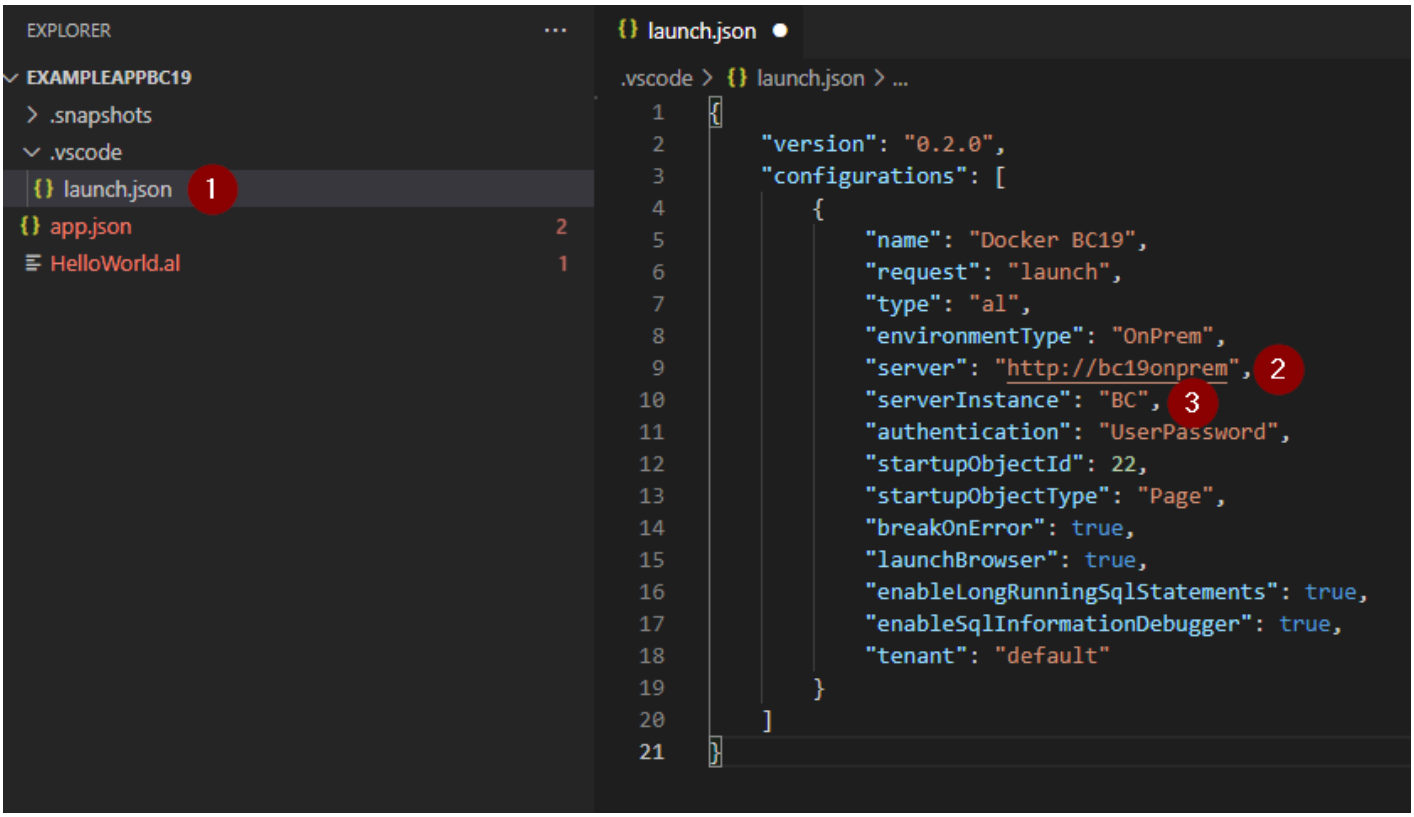
Die **launch.json**, also die Konfigurationsdatei mit den Startparametern für die Veröffentlichung der App aus Visual Studio heraus, wird im nächsten Schritt entsprechend der Auswahl des Servers erstellt. In diesem Fall wird mit einem On-Prem Server gearbeitet (in Docker).



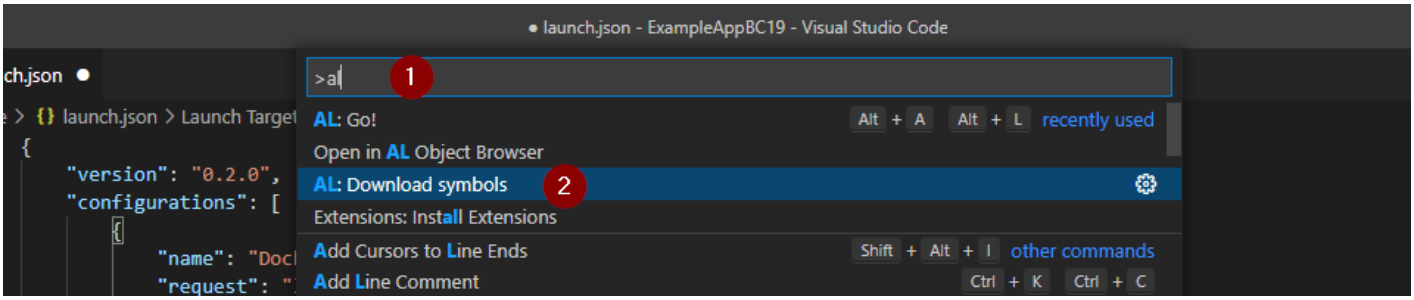
Sofern noch eine Aufforderung zur Eingabe von Benutzername und Passwort folgt, kann diese mit ESC abgebrochen werden.

Download von Symbolen

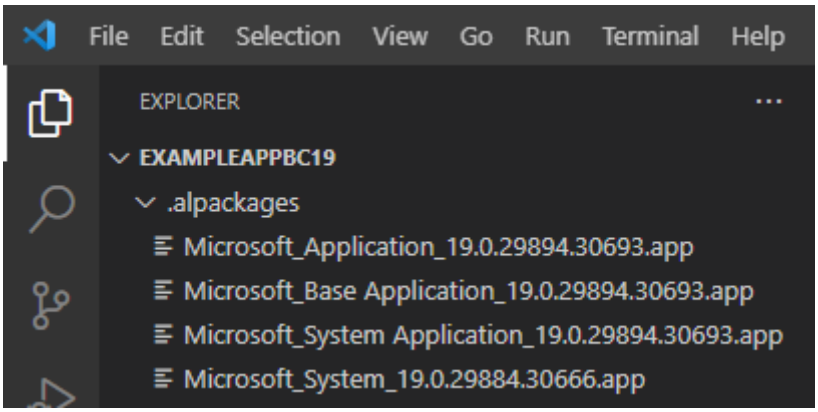
Visual Studio Code benötigt nun noch Informationen zu den Objekten und Extensions, die in der BC Instanz existieren. Damit der Download der sogenannten Symbole möglich ist, müssen die Verbindungsdaten in der Projektdatei **launch.json** entsprechend der Konfiguration in Docker angepasst werden:



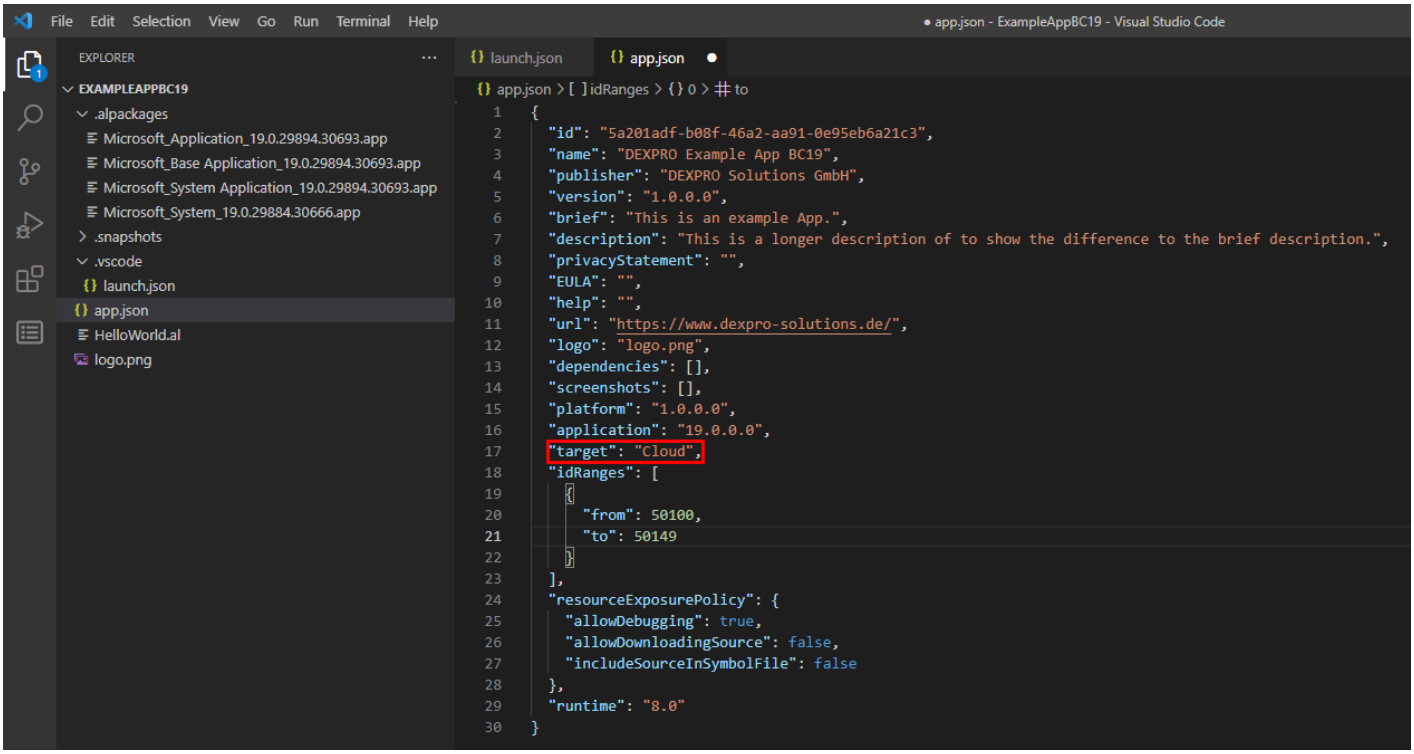
Anschließend in der Befehlszeile (F1) *AL: Download symbols* ausführen.



Wenn im Anschluss `[2021-10-11 10:36:58.09] All reference symbols have been downloaded.` angezeigt wird, hat alles geklappt. VS Code hat nun auch einen neuen Ordner mit Packages erstellt:



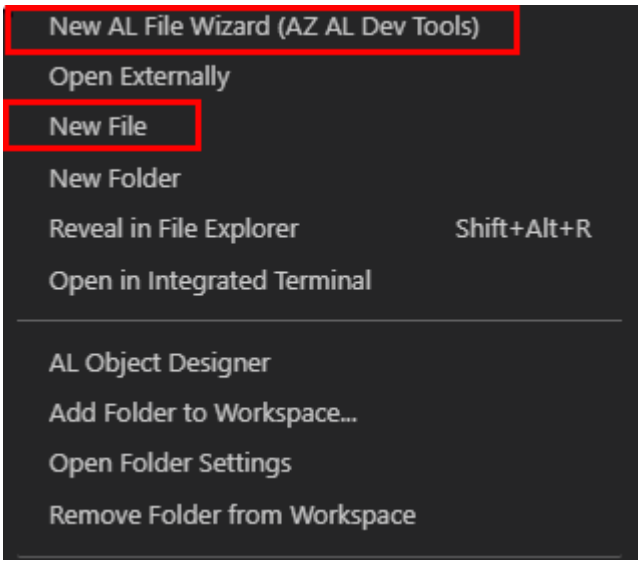
Je nachdem, wie die App später in BC bzw. im Store angezeigt werden soll, können nun noch Anpassungen an der **app.json** vorgenommen werden. Darüber hinaus kann hier festgelegt werden, für welches Ziel (Cloud oder On-Prem) entwickelt werden soll - dazu wird der Parameter **target** entsprechend gefüllt (hier: Cloud).



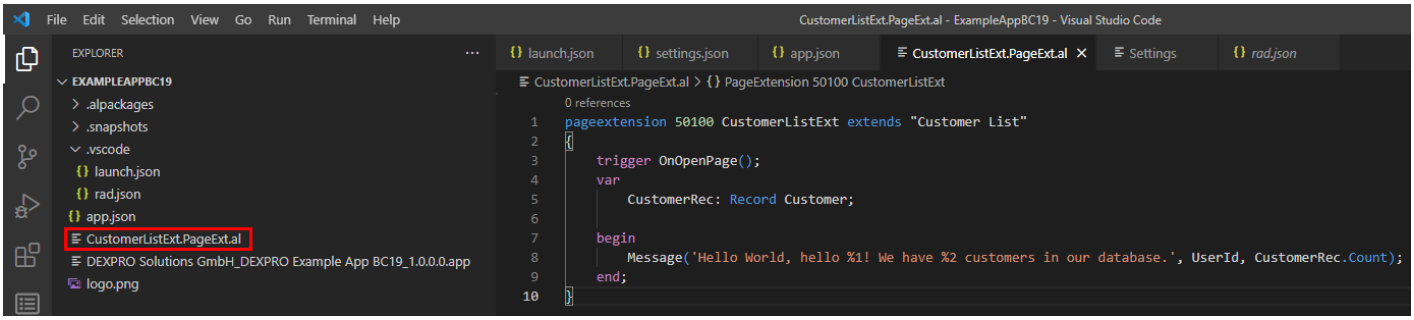
Erstellen und Kompilieren einer Extension

VS Code hat bereits eine HelloWorld.al Datei erstellt. Die Datei kann einfach gelöscht werden. Stattdessen bietet es sich an, eine neue, den best practices von Microsoft entsprechende, Datei zu erstellen. In diesem Fall eine Extension der Page "Customer List". Die Bezeichnung der Datei ist dann *CustomerListExt.PageExt.al.* Sinnvoll ist auch die Benutzung von Unterordnern (z.B. src, pages, pageextension etc.) - in diesem Beispiel sparen wir uns das mal.

Eine neue Datei wird erstellt, indem man an den Ort, wo die Datei hinterlegt werden soll, einen Rechtklick macht (hier: root) und anschließend **New File** wählt - alternativ kann auch der **New AL File Wizard** auf den AZ AL Dev Tools benutzt werden.



Die Datei dann passend benennen und mit beliebigem Code füllen.



Die folgenden Schritte (inkl. automatischer Tests) können via CI/CD auch automatisiert ausgeführt werden (z.B. in Azure DevOps), wenn der Programmcode in einem entsprechenden Repository hinterlegt ist.

Kompilieren einer Extension via VS Code

Befehlszeile öffnen (F1) und **AL: Package** ausführen. Der Code wird dann als *.app Package* kompiliert (im obigen Screenshot bereits passiert) - der Name setzt sich aus einigen, in der *app.json* hinterlegten, Parametern zusammen.

Veröffentlichung einer Extension via VS Code

In diesem Fall reicht es, mit F5 die Veröffentlichung (mit Debugging bzw. STRG+F5 ohne Debugging) zu starten - der Kompiliervorgang ist in dem Veröffentlichungsprozess integriert. Alternativ: Befehlszeile öffnen (F1) und **AL: Publish with(out) Debugging** ausführen.

Veröffentlichung einer Extension via PowerShell

Veröffentlichung in einem Docker Container:

Das Cmdlet *Publish-BcContainerApp,* mit entsprechenden Parametern, wird benutzt, um in einer BC Containerumgebung eine App zu veröffentlichen - das Script kann **hier** eingesehen werden.

Hier für unser Beispiel:

```
Publish-BcContainerApp -containerName "bc19onprem" -appFile
"C:\Users\BerndFeddersen\Documents\AL\ExampleAppBC19\DEXPRO Solutions GmbH_DEXPRO Example App
BC19_1.0.0.0.app" -install -sync –SkipVerification
```

Veröffentlichung außerhalb eines Containers:

Die Veröffentlichung in einer On-Prem Umgebung außerhalb eines Containers wird mit dem Cmdlet *Publish-NAVApp* realisiert. Das wird **hier** ganz gut beschrieben.

Veröffentlichung in der BC Cloud (ohne AppSource):

Mit entsprechenden Rechten (**EXTEND. MGT. - ADMIN** bzw. vor BC21 Wave 1 **D365 EXTENSION MGT**) kann eine Erweiterung einfach über den Webclient als "*Per Tenant Extension*" installiert werden. Die Erweiterung muss also in diesem Fall für jeden Mandanten installiert werden. Es folgt ein Beispiel anhand unserer BC Test-Cloud.

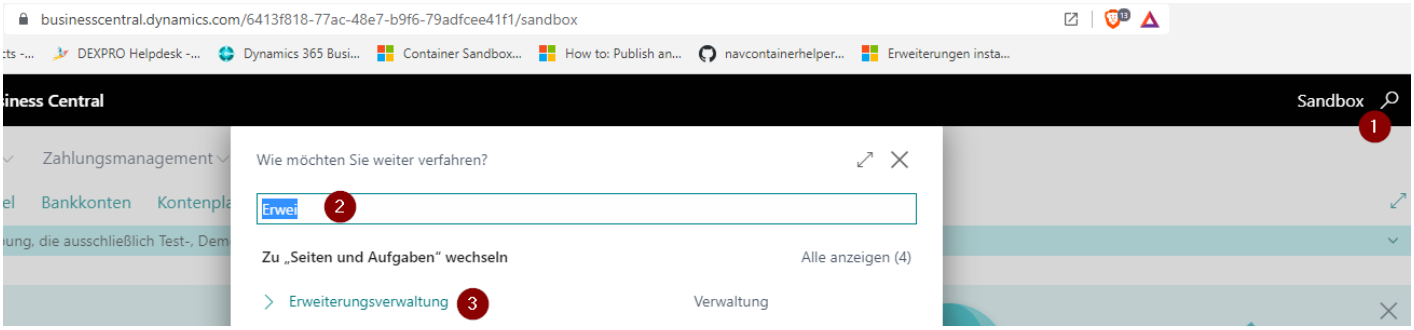
Unsere Testumgebung mit Cronus Testdaten ist unter der folgenden URL zu erreichen:

https://businesscentral.dynamics.com/6413f818-77ac-48e7-b9f6-79adfcee41f1

Als Sandbox:

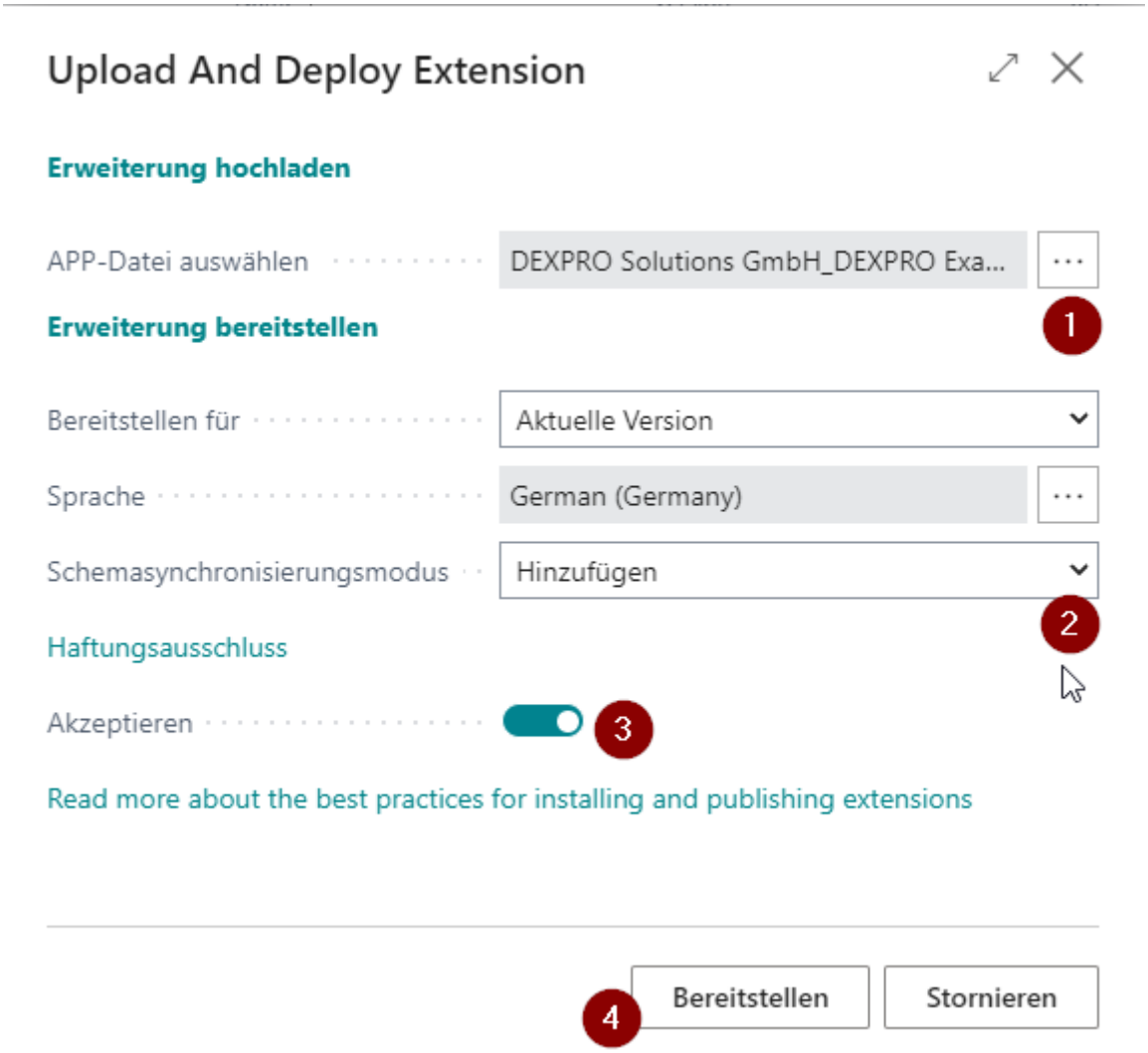
https://businesscentral.dynamics.com/6413f818-77ac-48e7-b9f6-79adfcee41f1/sandbox

In Business Central die Erweiterungsverwaltung öffnen:

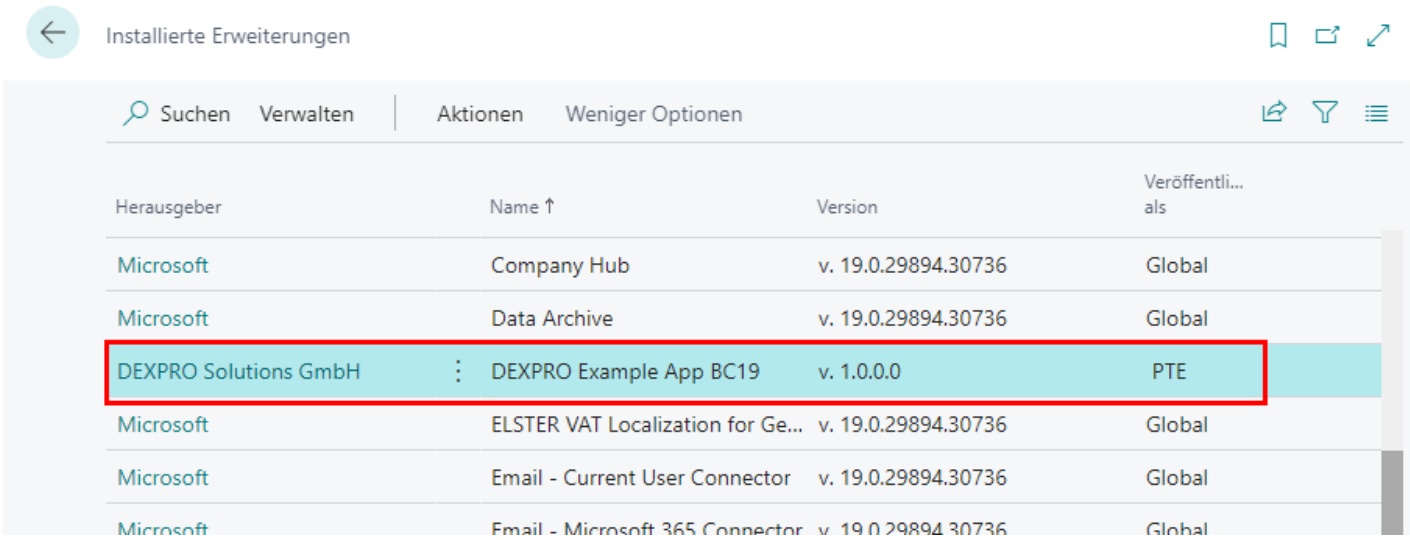


Anschließend das Menü zum Hochladen von Erweiterungen öffnen:

Daraufhin die durch VS Code kompilierte .app Datei auswählen und mit den gewünschten Einstellungen hochladen und installieren:



Nach einem kurzen Moment ist die installierte Erweiterung dann in der entsprechenden Liste sichtbar (ggfs. aktualisieren, damit die Änderung sichtbar wird):



Ergebnis:

In der Beispiel App haben wir die Page **Customer List** erweitert, sodass jedes Mal bei der

Ausführung des Triggers OnOpenPage irgendetwas passiert - dementsprechend sieht das jetzt so aus: