De techniek – Een eenvoudige Azure DevOps Extension bouwen

In de vorige blogpost is er beschreven hoe je in theorie een Azure DevOps extension maakt, maar hoe werkt het nu echt? In dit artikel zullen we een simpele extension bouwen en publiceren op de Azure Devops Marketplace.

Als eerste maken we een organization aan in de Azure Devops Marketplace. (https://app.vsaex.visualstudio.com/profile/account). Dit is gratis. Je kunt hier al je extensions aan koppelen, maar pas zodra Microsoft de organization heeft geverifieerd, kunnen de extensions ook beschikbaar worden gemaakt voor het grote publiek.

De Code

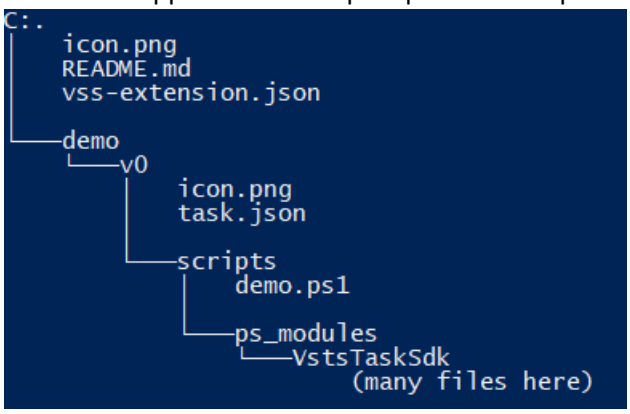
We beginnen met een simpel Powershell script en slaan deze op als Demo.ps1 :

```
[CmdletBinding()]
Param ()

#Set variable
$input = Get-VstsInput -Name Input -Require

#Display output
write-Host "Hello $input."
```

Om dit script naar een extension om te bouwen volgen we de volgende stappen.
Bouw de mappenstructuur op en plaats demo.ps1 in de scripts directory:



De laatste versie van de VstsTaskSdk is hier te vinden:
https://www.powershellgallery.com/packages/VstsTaskSdk/

We beginnen met de task.json file. Hierin wordt bepaald welke velden in Azure DevOps worden getoond en waar de script(s) staan die uitgevoerd moeten worden.

```
{
  "id": "DemoTask",
  "name": "Demo",
  "friendlyName": "Demo",
  "description": "Demo Extension.",
  "author": "Delta-N",
  "helpMarkDown": "",
  "category": "Utility",
  "demands": [],
  "version": {
    "Major": "0",
    "Minor": "0",
    "Patch": "1"
  },
  "minimumAgentVersion": "1.95.0",
  "instanceNameFormat": "Demo task",
  "inputs": [
    {
      "name": "Input",
      "type": "string",
      "label": "What is your name?",
      "defaultValue": "",
      "required": true
    }
  ],
  "execution": {
    "PowerShell3": {
      "target": "scripts/demo.ps1"
    }
  }
}
```

In de vss-extension.json staat de informatie die de marketplace nodig heeft om te bepalen welke taken bij de extension horen en welke organization deze gemaakt heeft.

```
{
  "manifestVersion": 1,
  "id": "30b61236-5eeb-4d8c-bd67-f8fb7ae01bf8",
  "version": "0.0.1",
  "name": "Demo extension",
  "publisher": "Delta-N",
  "description": "A simple demo extension",
  "tags": [
    "Extension",
    "Delta-N"
  ],
  "public": true,
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ],
  "icons": {
    "default": "icon.png"
  },
  "scopes": [
    "vso.build"
  ],
  "categories": [
    "Azure Pipelines"
  ],
  "content": {
    "details": {
      "path": "readme.md"
    }
  },
  "contributions": [
    {
      "id": "DemoTask",
      "type": "ms.vss-distributed-task.task",
      "description": "Demo Task",
      "targets": [
        "ms.vss-distributed-task.tasks"
      ],
      "properties": {
        "name": "demo"
      }
    }
  ],
  "files": [
    {
```

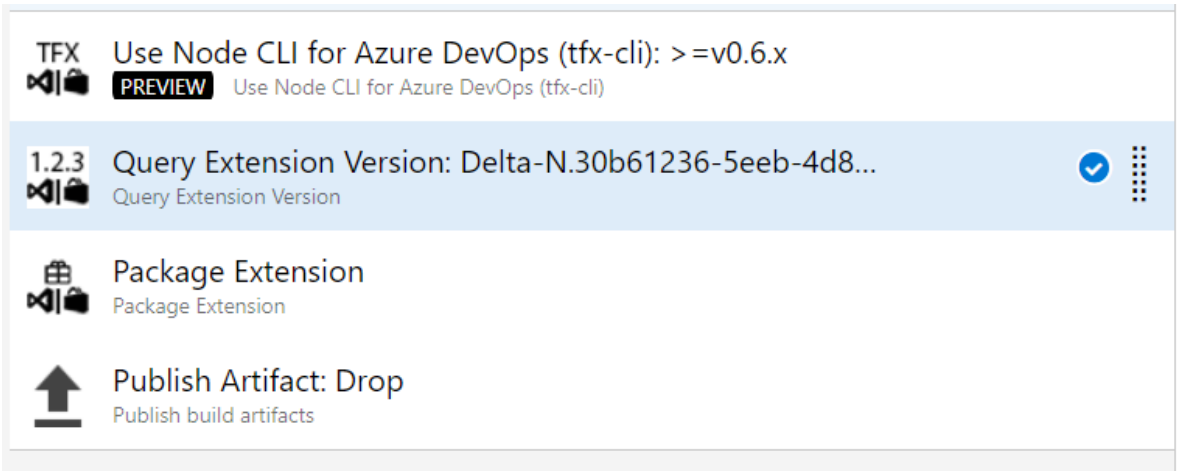
```
      "path": "demo"
    }
  ]
}
```

De volgende velden zijn belangrijk:

- **Id:** Deze waarde is uniek. Hier kan een naam of een GUID voor gebruikt worden.
- **Publisher:** Dit is de naam van de organization die je hiervoor aangemaakt hebt.
- **Contributions / id:** Hier komt het id te staan die gelijk is aan het id in de task.json.

De build pipeline

Nu code klaar is, is het tijd om de build pipeline op te zetten. We hebben 4 taken nodig om de extension te creëren.



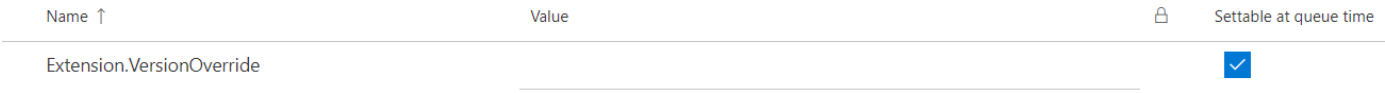
Use Node CLI for Azure DevOps (tfx-cli)

Deze taak haalt de NPM module TFX op die vereist is om de extension te compileren. Er hoeft na het toevoegen van deze taak geen aanpassing gedaan te worden aan de instellingen.

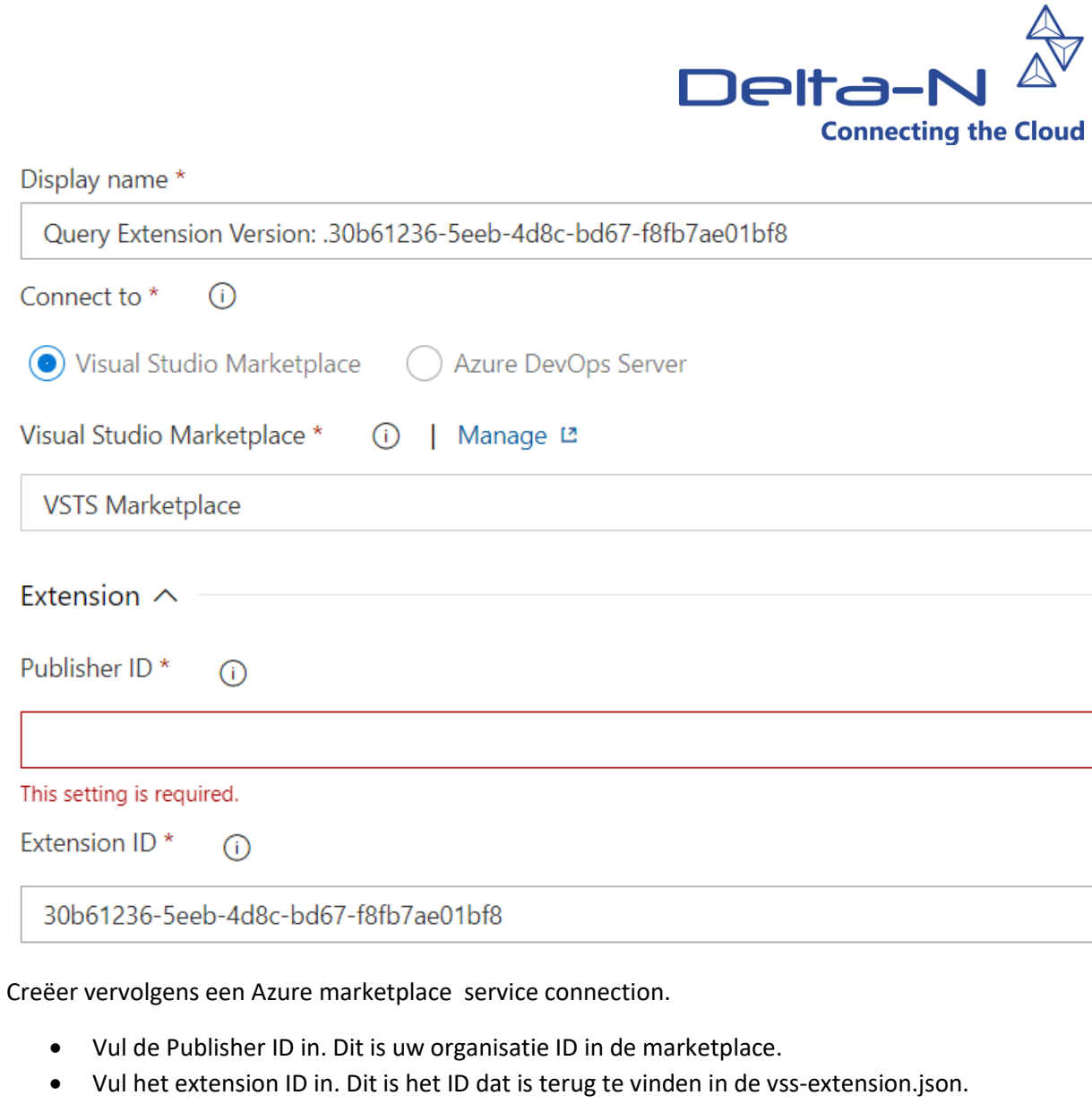
Query Extension Version

Deze taak haalt in de marketplace het huidige versienummer van de extension op. Vervolgens kan worden aangegeven of de major, minor of patch versie verhoogd moet worden. Aangezien we een nieuwe extension gaan uploaden, maken we gebruik van de variable 'Extension.VersionOverride. Deze variable geeft de mogelijkheid om zelf een versienummer op te geven en query over te slaan.

Maak de volgende Variable aan:

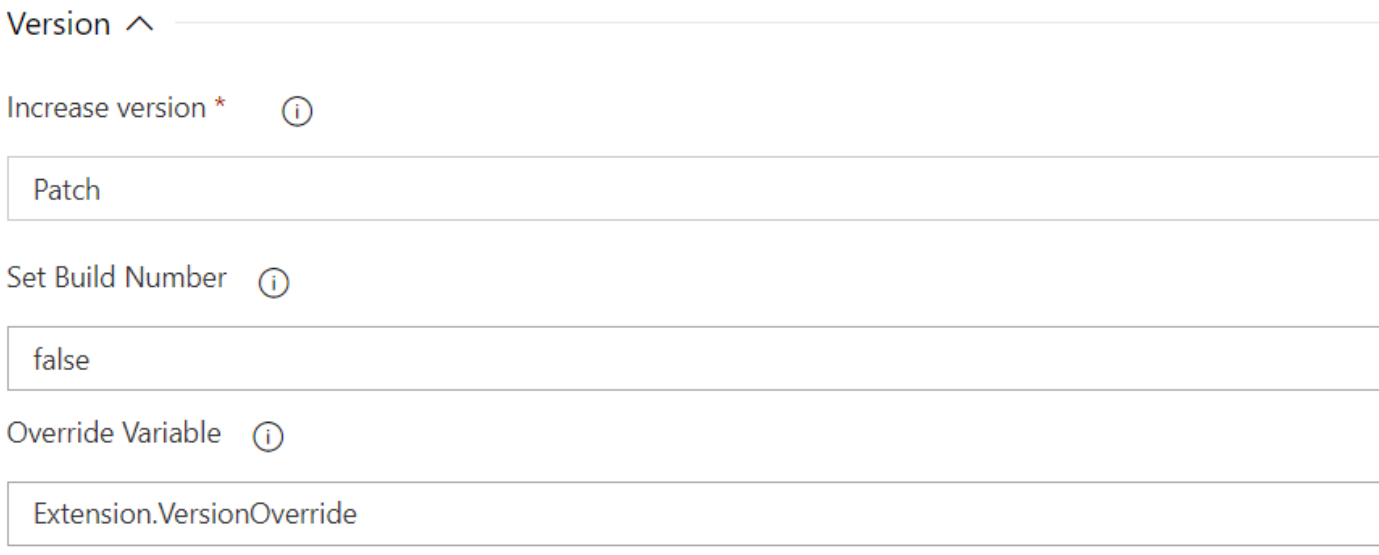


Laat de value leeg en zet het vinkje "Settable at queue time" aan, zodat er tijdens de eerste build een versienummer opgegeven kan worden.



Creëer vervolgens een Azure marketplace service connection.

- Vul de Publisher ID in. Dit is uw organisatie ID in de marketplace.
- Vul het extension ID in. Dit is het ID dat is terug te vinden in de vss-extension.json.



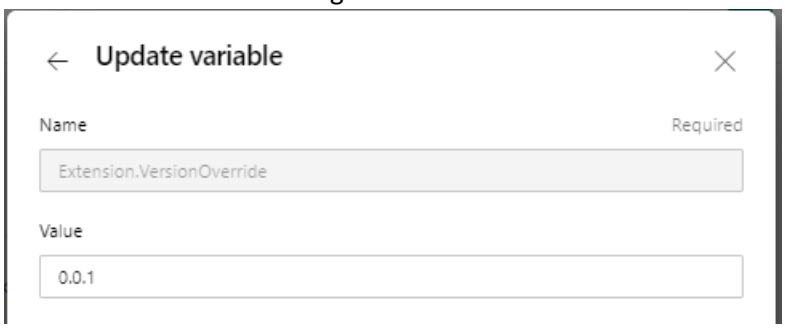
Neem bovenstaande instellingen over.

Package Extension

Deze taak compileert de code tot een VSIX bestand.

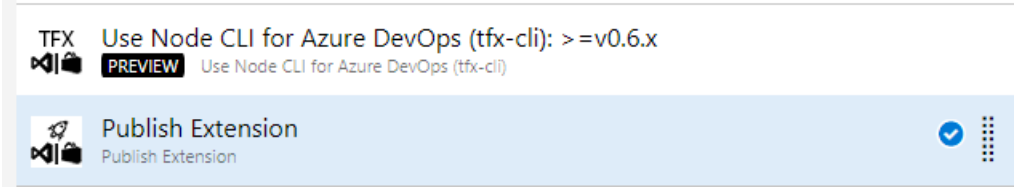
Neem bovenstaande instellingen over. De $(output.Extension.Version) is een variable die wordt gevuld door de Query Extension Version task.

Nu de pipeline helemaal gevuld, kunnen we de build starten. Vergeet niet de variable in te vullen met de eerste versie. In dit geval wordt het 0.0.1.



Release Pipeline

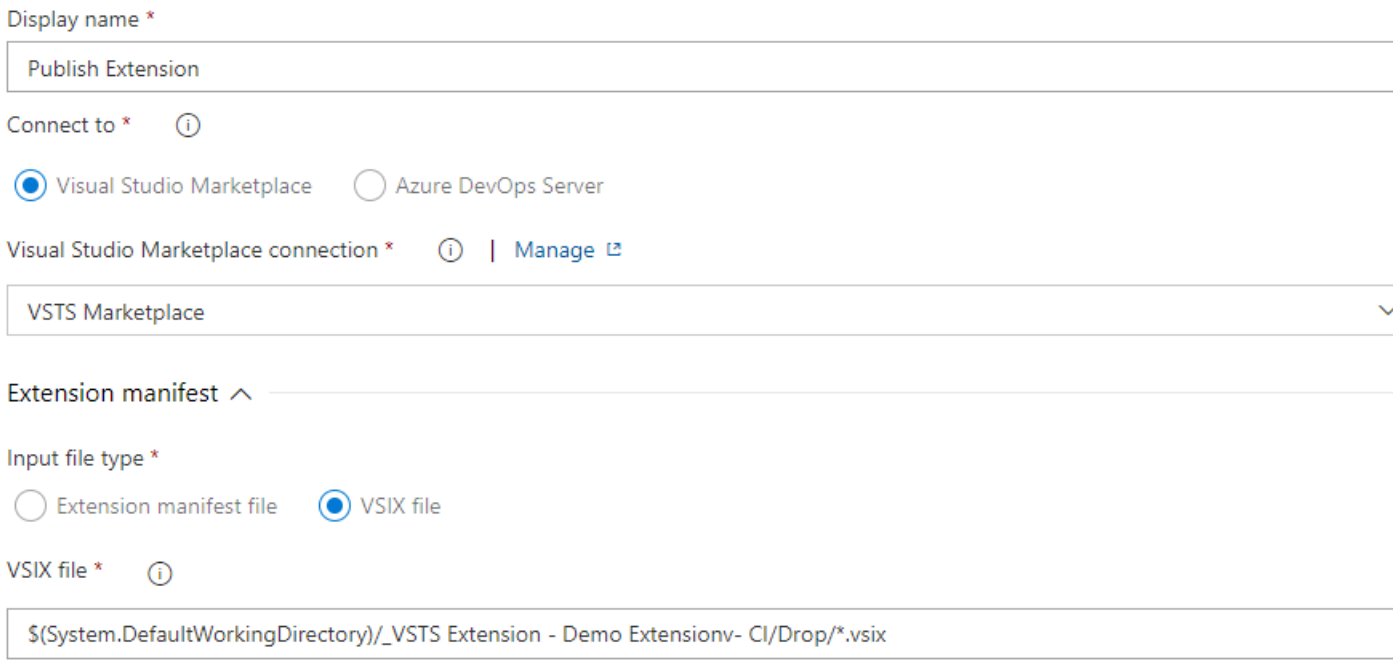
Nu we een build artifact hebben, kunnen we de extension in de marketplace uploaden. Selecteer na het aanmaken van de pipeline bij artifacts de build pipeline van de vorige stappen.
Vervolgens voegen we de volgende taken toe aan de stage.



De ‘Use Node CLI for Azure DevOps’ task is al uitgelegd in de build pipeline, dus deze slaan we over.

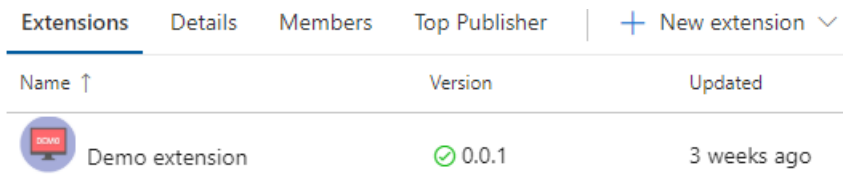
Publish Extension

In deze taak geven we aan welk VSIX bestand naar de marketplace gestuurd moet worden.



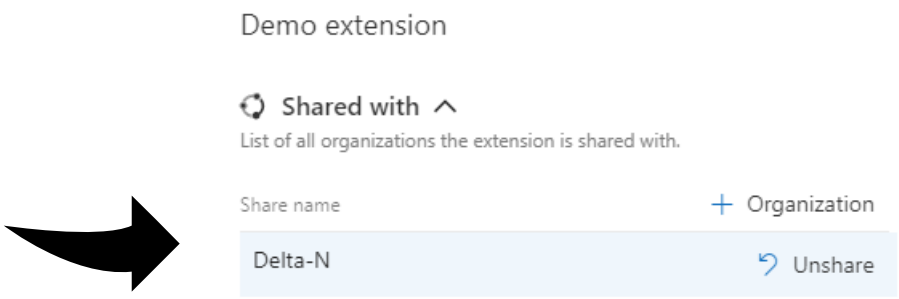
Kies de file locatie door middel van de … en vervang vervolgens de bestandsnaam.vsix voor *.vsix. Hierdoor hoeft bij de volgende versie van de release, de release pipeline niet aangepast te worden.

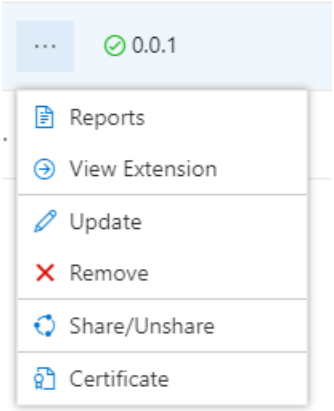
Na het runnen van de release pipeline, kan je de extension terug vinden in de marketplace. Ga hiervoor naar https://marketplace.visualstudio.com/manage.



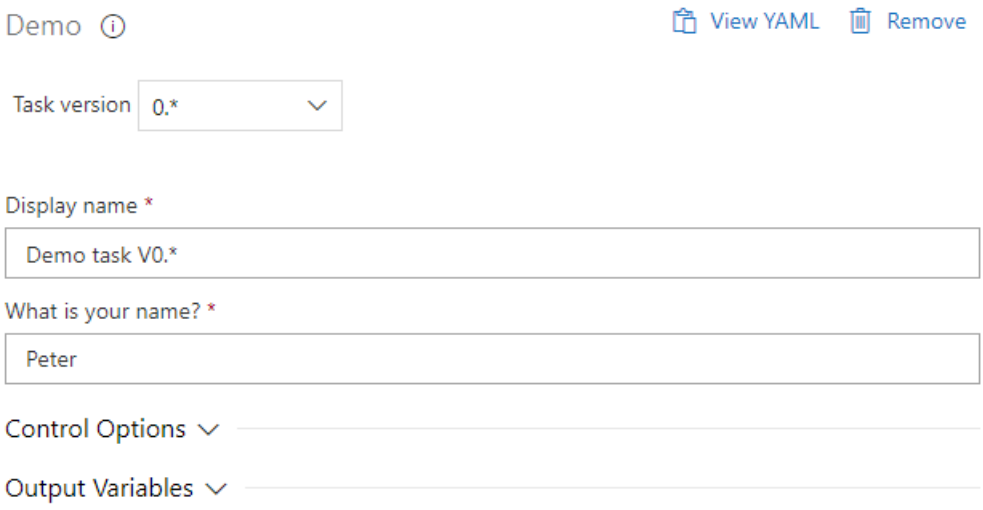
Deel en installeer de extension

Om de extension te testen, moeten we de deze delen met de organisatie. Klik hiervoor op de puntjes achter de extension.





Als we nu de extension installeren en toevoegen in de pipeline, dan hebben we een veld om een naam in te vullen.

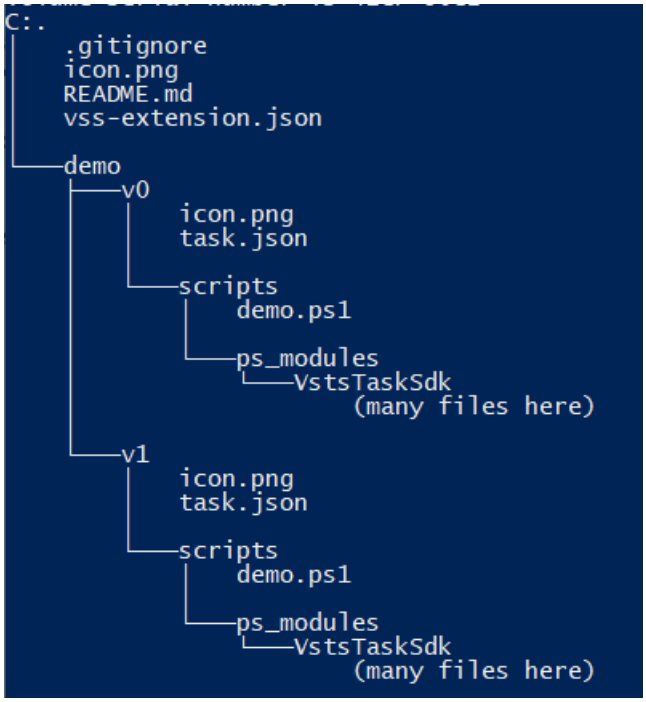


Als we de pipeline draaien, krijgen dit resultaat:

Versie beheer

Wat nu als er een aanpassing gedaan moet worden die de werking van de huidige extension in de weg zit? Zoals in de vorige blogpost beschreven kunnen meerdere versies van een task in de extension worden opgenomen. Voor versie 1 van deze task, gaan we een veld toevoegen.

Als eerste maken we een copy van de folder V0 en noemen we deze V1.



Vervolgens voegen we een input toe aan task.json binnen de V1 map

```
"inputs": [
    {
      "name": "Input",
      "type": "string",
      "label": "What is your name?",
      "defaultValue": "",
      "required": true
    },
    {
      "name": "Color",
      "type": "string",
      "label": "What is your favorite color?",
      "defaultValue": "",
      "required": true
    }
  ],
```

En passen we de versie aan naar 1.0.0

```
"version": {
    "Major": "1",
    "Minor": "0",
    "Patch": "0"
```

```
  },
```

Daarna passen we het Powershell script aan, zodat we dit nieuwe veld ook kunnen uitlezen

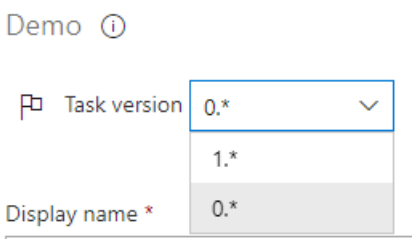
```
[CmdletBinding()]
Param ()

#Set variable
$input = Get-VstsInput -Name Input -Require
$color = Get-VstsInput -Name Color -Require

#Display Output
write-Host "Hello $input. $Color is a great Color!"
```

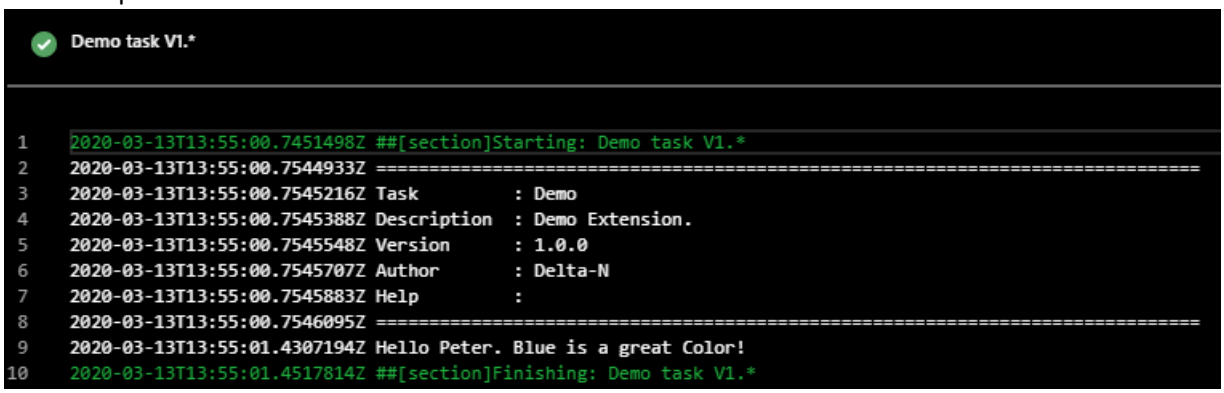
We hoeven nu niets meer aan de pipelines te doen, dus zodra je de code commit, wordt door de buildpipeline automatisch de patch versie van de extension met 1 verhoogd.

Als we nu in Azure DevOps gaan kijken, kunnen we kiezen uit 2 versies. 0.* en 1.*.



Bij task version 1.* zien we nu een extra veld.

En de output is



Op deze manier is het mogelijk om grote wijzigingen door te voeren, zonder dat bestaande pipelines hier hinder van ondervinden.

Disclaimer

Microsoft update Azure DevOps met grote regelmaat. Hierdoor kan het zijn dat niet alle gebruikte screenshots overeenkomen met de werkelijkheid.

Peter Barendse, Devops Engineer