# Autofac injecten

Da mich das eben ein paar Minuten gekostet hat, will ich das einmal festhalten.

Folgendes NuGet muss im Projekt inkludiert sein: **Autofac.Extensions.DependencyInjection**

HostApplication:
[crayon-6766ea611b493763443885/]
WebApplication:
[crayon-6766ea611b499506261198/]

---

# Docker auf Windows installieren ohne Docker Desktop

Die DockerCLI und DockerEngine sind Freeware und OpenSource, im Business Bereich muss die Docker Desktop Anwendung erworben werden. Um Docker nutzen zu können, braucht man nicht unbedingt Docker Desktop. Auch wenn Docker Desktop einem bei der Einrichtung vieles einfacher macht. Hier soll eine Schritt für Schritt Anleitung folgen

Alle Schritte nutzen PowerShell oder meine Empfehlung „Windows Terminal Canary“

# WSL (Windows-Subsystem für Linux )

Über WSL läuft eine Linux distro unter Windows.

hat man noch kein WSL installiert, kann über diesen Befehl eine Distro ausgewählt werden. Standard ist Ubuntu, ich nutze hier debian.
[crayon-6766ea611b943611601853/]
um WSL zu installieren gibt man folgenden Befehl ein
[crayon-6766ea611b947570003269/]
Da das Linux System CPU und RAM Resourcen verbraucht, kann man diese einschränken. Dazu wird eine .wslconfig Datei im Userverzeichnis erstellt
[crayon-6766ea611b948861471098/]
In Windows Explorer %UserProfile% eingeben und die erstellte .wslconfig optional bearbeiten:
[crayon-6766ea611b949391973411/]
jetzt **wsl** eingeben und wir befinden uns in der Kommandozeile von Linux

Als erstes sollte man die distro aktualisieren
[crayon-6766ea611b94b110027218/]

## Optional: WSL Image verschieben

Es kann Sinnvoll sein, die WSL Images nicht im Standard Verzeichnis zu haben. Daher kann die WSL Image jederzeit umziehen.

1. Name des WSL-Images bestimmen. In meinem Fall ist es „Debian“:
   [crayon-6766ea611b94c863536238/]
2. als tar exportieren:
   [crayon-6766ea611b94d043834897/]
3. In WSL das image lösen:

[crayon-6766ea611b94e085472146/]

4. 2 Verzeichnisse erstellen. Mein neues Verzeichnis ist D:\Docker\wsl
   [crayon-6766ea611b950969361273/]
5. WSL herunterfahren:
   [crayon-6766ea611b951110672280/]
6. Import starten: wsl –import Debian
   [crayon-6766ea611b952915562323/]
7. Neues Image starten:
   [crayon-6766ea611b953388840655/]
8. Optional:
   [crayon-6766ea611b954461497717/]

# Docker installieren und einrichten

Zunächst werden benötigte Repositories für das Generieren von Zertifikaten installiert und dann docker selbst. Wichtig: Die Zeilen hier unten so wie die sind nacheinander, Zeile für Zeile kopieren. Nicht mehrere Zeilen miteinander in das Terminal einfügen. Aufforderungen mit Y und Enter bestätigen
[crayon-6766ea611b955446675588/]
[crayon-6766ea611b956615015604/]
Docker Daemon starten. Wenn der nicht weiter geht, Terminal schließen und neuen Terminal mit **wsl** starten
[crayon-6766ea611b957702184355/]
Docker müsste jetzt installiert sein. (optional) Das testen wir, indem wir ein hello-world container laufen lassen:
[crayon-6766ea611b958682224361/]
Damit unser aktueller User nicht immer sudo eingeben muss, können wir ihn zu der Gruppe der Docker Administratoren hinzufügen. Evtl. ist die Gruppe bereits vorhanden, dann Meldung ignorieren.
[crayon-6766ea611b959707702414/]

# Um sicherzustellen, dass Docker bei jeden Neustart läuft, folgende Befehle ausführen:

[crayon-6766ea611b95a248068865/]
Damit auch der dockerd nach dem Start läuft, muss man ein script einfügen. systemctl kann man mit WSL leider nicht nutzen.

Wir legen fest, dass wie sudo bei dem aktuellen Benutzer Benutzer gehandhabt werden soll. Nämlich soll für dockerd kein Passwort erfragt werden:
[crayon-6766ea611b95b923223115/]
ganz unten folgendes hinzufügen und Benutzer gegeben falls anpassen:
[crayon-6766ea611b95d744947131/]
Jetzt müssen wir die bash bearbeiten, damit nach reboot das Skript ausgeführt wird
[crayon-6766ea611b95e852960194/]
mit „Bild runter“-Taste kann man bis ganz unten blättern und dort folgenden Code einfügen. Mit Strg+X, dann Y und Enter wieder raus:
[crayon-6766ea611b95f519660065/]
Shell neustarten:
[crayon-6766ea611b960097436032/]
WSL beenden:
[crayon-6766ea611b961749118097/]
und WSL reboot:
[crayon-6766ea611b962850192577/]
Falls das nicht geht, kann ein Windows Job eingerichtet werden, der wsl automatisch bei Windows Start ausführt
[crayon-6766ea611b963969338655/]
**Unter Aufgabenplanung in Windows diesen Task als „Unabhängig von der Benutzeranmeldung“ konfigurieren. Sonst funktioniert dieser Task nicht**

# Linux Docker, Windows bekannt machen

Docker läuft nun auf unserer WSL Linux Maschine. Das nützt uns wenig, da wir in PowerShell nicht immer auf die WSL Linux Maschine zugreifen wollen. Wir wollen einfach docker eingeben und damit die docker Instanz auf der Linux Maschine meinen. Daher erstellen wir jetzt die Brücke.

Zuerst müssen wir das in Linux erlauben. Folgender Befehl öffnet einen Editor:
[crayon-6766ea611b964236700764/]
dort mit Strg+V folgendes einfügen:
[crayon-6766ea611b965465700782/]
Mit Strg+X , dann Y und Enter speichern

Mit **exit** beenden wir die WSL Kommandozeile

Jetzt muss unter Windows eine Umgebungsvariable für diesen Docker Host angelegt werden (Terminal muss als Admin laufen):
[crayon-6766ea611b966910313641/]

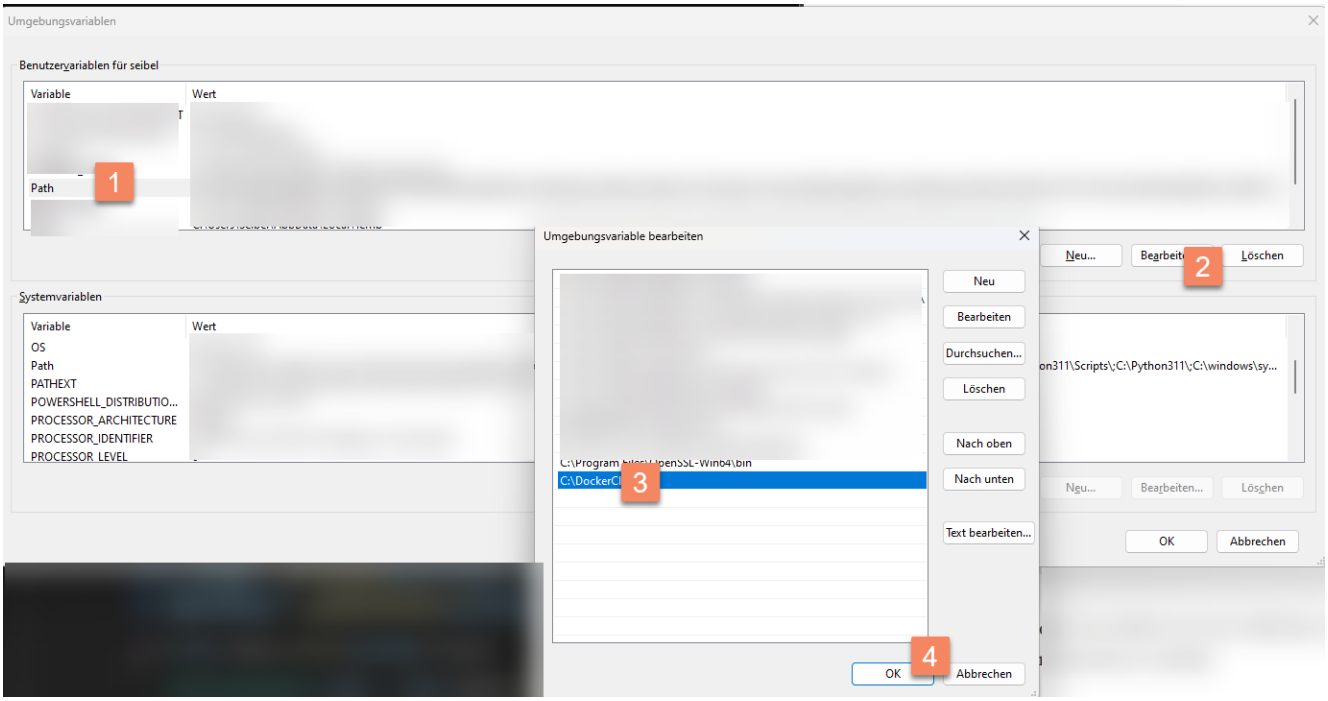
# Docker CLI unter Windows installieren

Wenn man bereits choco nutzt, geht das easy mit

[crayon-6766ea611b967031516205/]
Ansonsten hier herunterladen:

https://download.docker.com/win/static/stable/x86_64/

und unter einem Verzeichnis z.B. C:\DockerCli ablegen. Nun muss dieses Verzeichnis in den Umgebungsvariablen unter Path eingetragen werden




über folgenden Befehl müsste Docker nun aufrufbar sein
[crayon-6766ea611b968831999176/]

> *Man muss jetzt wissen, dass es nun 3 Ebenen gibt.*
>
> *Windows -> WSL Linux -> Docker Container.*
>
> *WSL hat für Windows automatisch ein mount in dem Ordner /mnt/ eingerichtet*
>
> *Möchte man nun, dass der Container auf das Windows Verzeichnis C:\Temp bindet, so muss man stattdessen /mnt/c/Temp eingeben*

Referenz:

Docker and WSL2 without Docker Desktop | by Romain Bruyère | Medium

files folders – Change the storage location of a WSL2 – Super User

How to automatically start the Docker daemon on WSL2 – NillsF blog

# Portainer installieren mit SSL Zertifikat

Portainer über docker installieren
[crayon-6766ea611b969874472404/]
! Funktioniert noch nicht. Das Zertifikat wird vom Browser nicht anerkannt !

Zertifikat auf der Linux WSL erzeugen und das Zertifikat nach D:\Docker kopieren:
[crayon-6766ea611b96a890263798/]

---

# dotNet Docker veröffentlichen

1. Wenn ein eigener nuget Server mit http verwendet wird, dann muss dort wo die sln liegt, eine NuGet.config angelegt werden:
   [crayon-6766ea611bc8e026975959/]
2. die csproj bearbeiten:
   [crayon-6766ea611bc92517086803/]

3. Docker Desktop muss vorinstalliert sein und mit wsl2 laufen
   [crayon-6766ea611bc93678147722/]
4. Https Zertifikat für Entwicklung erstellen: https://learn.microsoft.com/en-us/aspnet/core/security/docker-https?view=aspnetcore-8.0#running-pre-built-container-images-with-https
   [crayon-6766ea611bc94431980990/]

# Docker

# Docker Desktop (AMD64)

Docker Desktop: The #1 Containerization Tool for Developers | Docker

In der Regel reicht auch nur die cli
[crayon-6766ea611be02625224146/]
und Rancher Desktop
[crayon-6766ea611be05318642046/]

# Portainer

https://docs.portainer.io/start/install-ce/server/docker/wsl
[crayon-6766ea611be06810128757/]
WebAdmin: https://localhost:9443/

# Microsoft Sql Server

microsoft/mssql-server – Docker Image | Docker Hub
[crayon-6766ea611be07920573990/]
Passwort setzen nicht vergessen

Im Container, unter volumes als bind, ein Verzeichnis einrichten, wo die backups liegen und wo die volumes liegen sollen

# RabbitMQ

[crayon-6766ea611be08122839453/]
WebAdmin: https://localhost:15672/

Benutzername: guest

Password: guest

Hashicorp/Vault
[crayon-6766ea611be0a730496115/]
hashicorp/vault – Docker Image | Docker Hub

Weboberfläche: http://localhost:8200/

---

# Discord Installation „a fatal javascript error occurred“

Das war die einzige Lösung, die funktionierte, um Discord zu installieren über choco:

I have a problem with discord, and its no the logo. : r/discordapp (reddit.com)

Dazu das Powershell-Script. Ausprobiert und getestet.
[crayon-6766ea611bfbe123530255/]

# Jupyter unter Windows installieren

Mit Jupyter (Julia + Python) kann man interaktive Notebooks erstellen.

Ein Notebook besteht aus Cells, die entweder als Text (markdown) oder Code (Python, etc.) nacheinander erstellt werden können.

Der Code kann im Notebook ausgeführt werden und visuell dargestellt werden. Variablen können Code-Zelle übergreifend genutzt werden.

Damit der Code ausgeführt werden kann, wird ein Kernel benötigt, der wie ein Interpreter den Code verarbeitet und ausgibt.

Um Jupyter nutzen zu können, muss dieser zunächst installiert werden.

# Installation Jupyter

Jupyter selbst benötigt Python und für die Installation den Python package manager, welcher in Python inkludiert ist.
[crayon-6766ea611c12a587552464/]
Über den Befehl

[crayon-6766ea611c12d312208413/]
wird der JupyterLab Server lokal gestartet. Dieser ist erreichbar unter http://127.0.0.1:8888

In der Konsole ist auch der Token hinterlegt, den man für die

Ersteinrichtung benötigt.

# Editoren

## JupyterLab Web

Über http://127.0.0.1:8888 kommt man auf die Schaltzentrale von JupyterLab.

Dort kann man auch die Notebooks erstellen und bearbeiten

## Azure Data Tool

My experience working with notebooks in Azure Data Studio – Data on Wheels – Kristyna Ferris & Steve Hughes (wordpress.com)

JupyterLab unterstützt von Haus aus nur Python als Kernel. Weitere Kernels müssen nachinstalliert werden.

Leider gibt es kein Kernel für SQL. Aber Azure Data Tools hat ein eingebautes Kernel für SQL Server.

Die Nutzung inkl. Copilot ist wirklich komfortabel.

## VS Code

Für VsCode müssen die Jupyter Extensions heruntergeladen werden.

Über Strg + Shift + P kann eine neue Jupyter Datei erstellt werden.

Oben rechts in der Ecke wählt man „Select Kernel“ aus und gibt als Adresse http://127.0.0.1:8888 und das Passwort ein.

Danach kann man die Kernels aus JupyterLab nutzen

## nteract

nteract ist ein weiterer Editor um Jupyter Notebooks zu erstellen. Dieser kann mit CodeMirror und Monaco arbeiten.

nteract: write your next code-driven story.

# Weitere Kernel

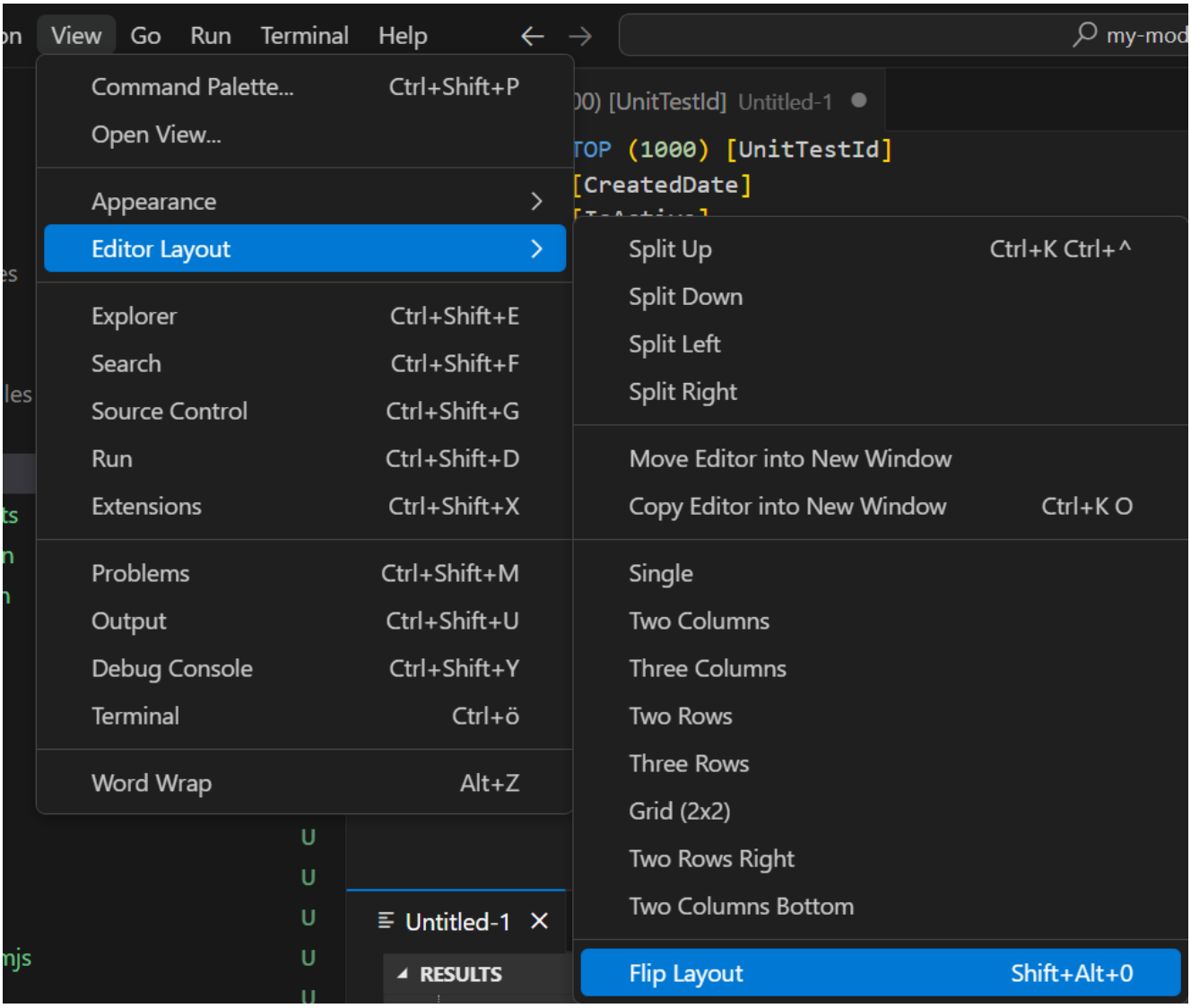
[crayon-6766ea611c12e500350417/]

## .NET Interactive (C#, F#, PowerShell)

[crayon-6766ea611c130016686451/]

---

# VS Code SQL Fenster horizontal ausrichten




---

# Entity Framework EF – Tabellen Models generieren lassen

Alt + T -> „NU“ eingeben

[crayon-6766ea611c2ba474310651/]

# Windows Powershell Dienst einrichten

[crayon-6766ea611c42f281427991/]

# Nuxt 3 mit Keycloak autorisieren

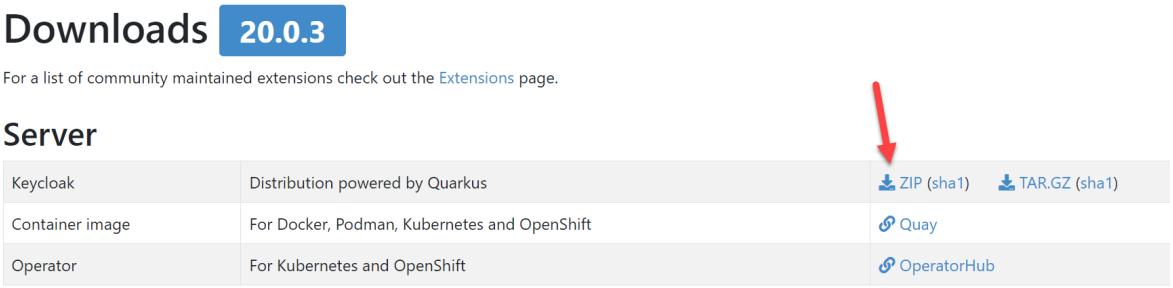
In diesem Beitrag will ich eine Schritt für Schritt Anleitung geben, wie man Nuxt 3 mit Keycloak autorisiert.

Die Thematik um Autorisierung und Authentifizierung bedarf einer soliden Kenntnis in der Thematik. Ich empfehle daher unbedingt sich die Zeit zu nehmen und das folgende Video anzuschauen und sicher zu gehen, dass alles verstanden wurde:

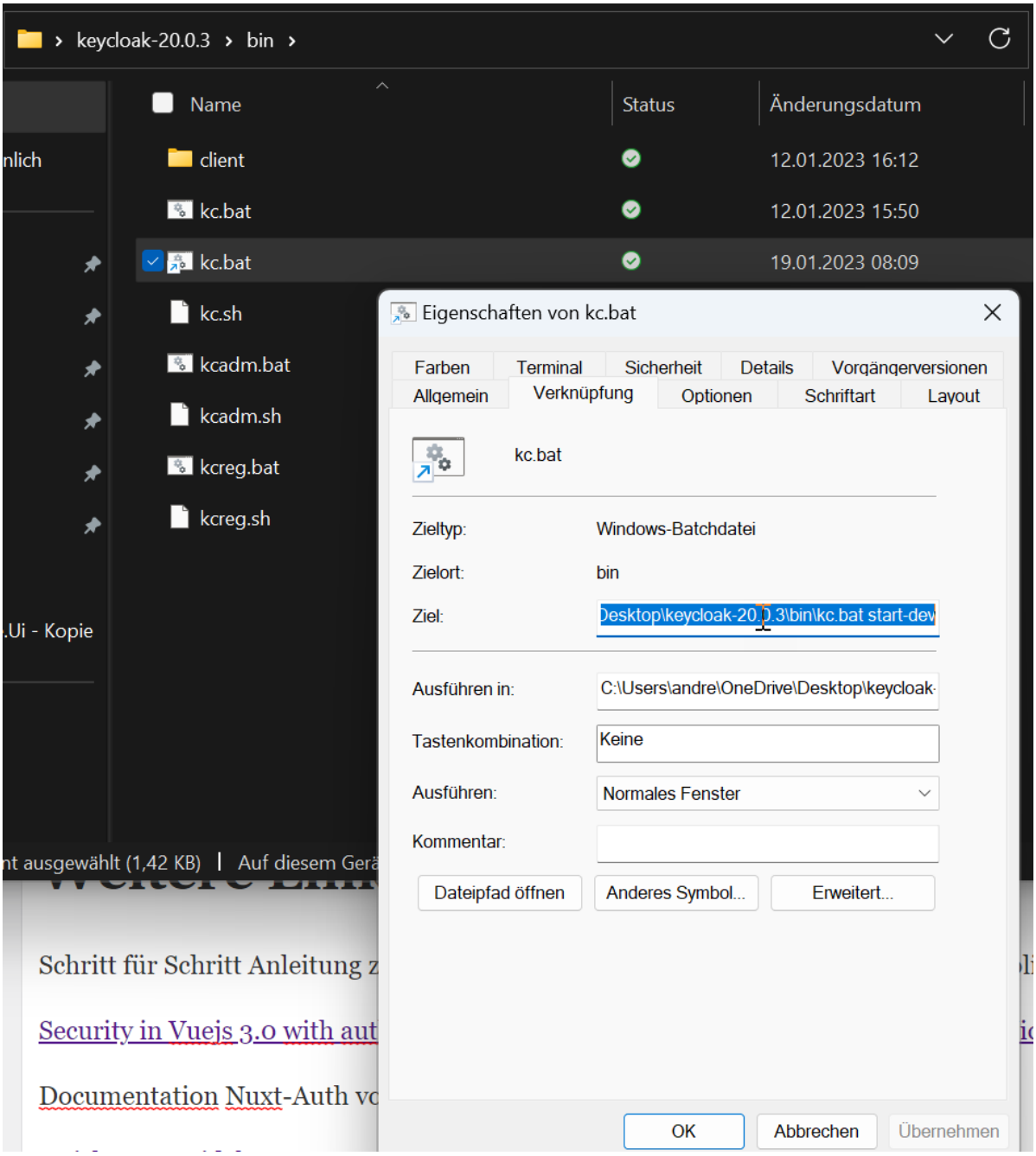
Zum Ende füge ich weitere Lohnenswerte Links hinzu.

# 1. Keycloak installieren

1. Keycloak herunterladen: downloads – Keycloak




2. Zip-Datei entpacken und in das **bin** Verzeichnis wechseln. Dort eine Verknüpfung vom **kc.bat** erstellen, dann auf Eigenschaften und beim Ziel **start-dev** anhängen

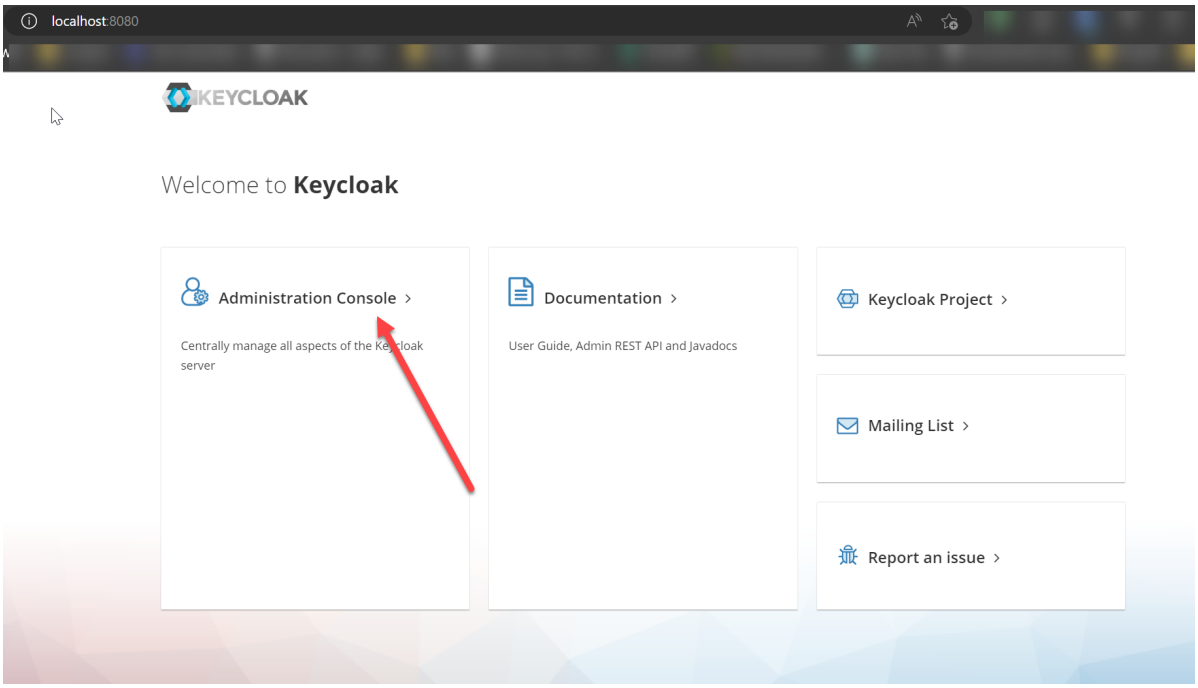



3. Verknüpfung starten und keycloak ist unter der Adresse http://localhost:8080 erreichbar. Dort einen neuen Admin

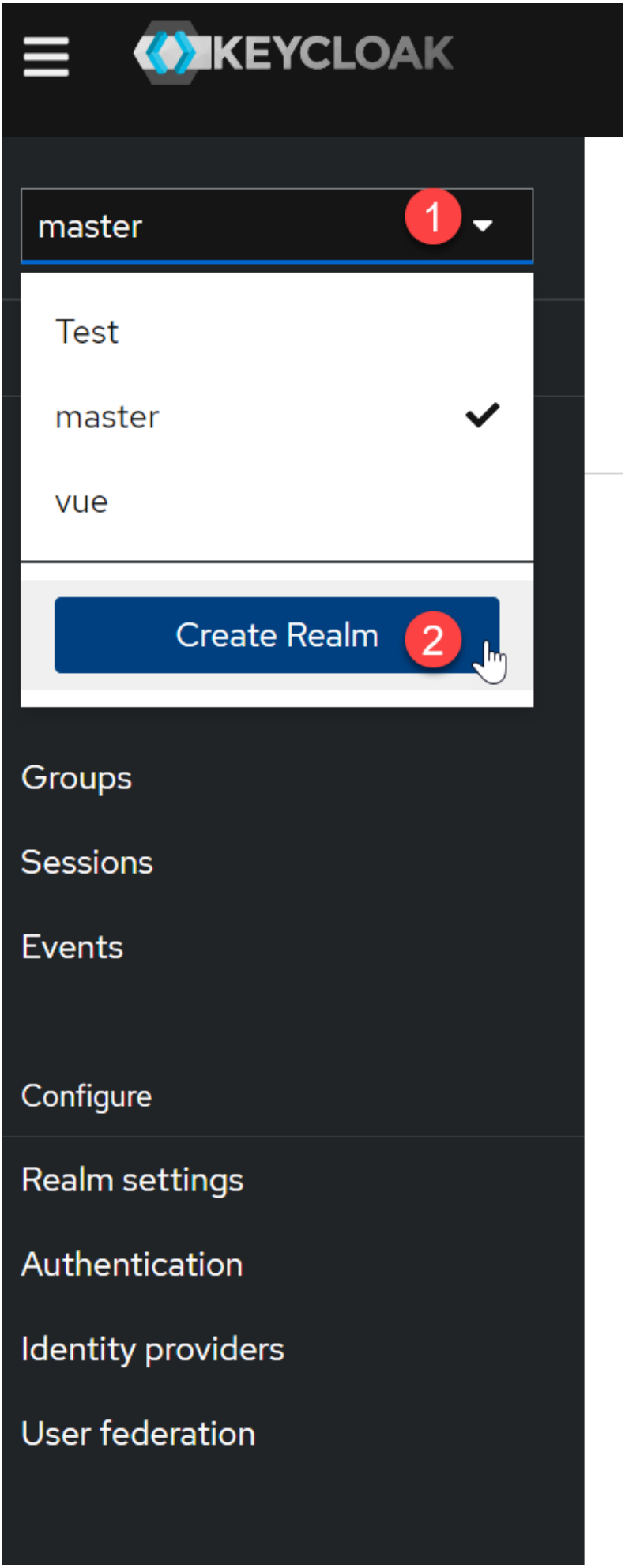
User anlegen

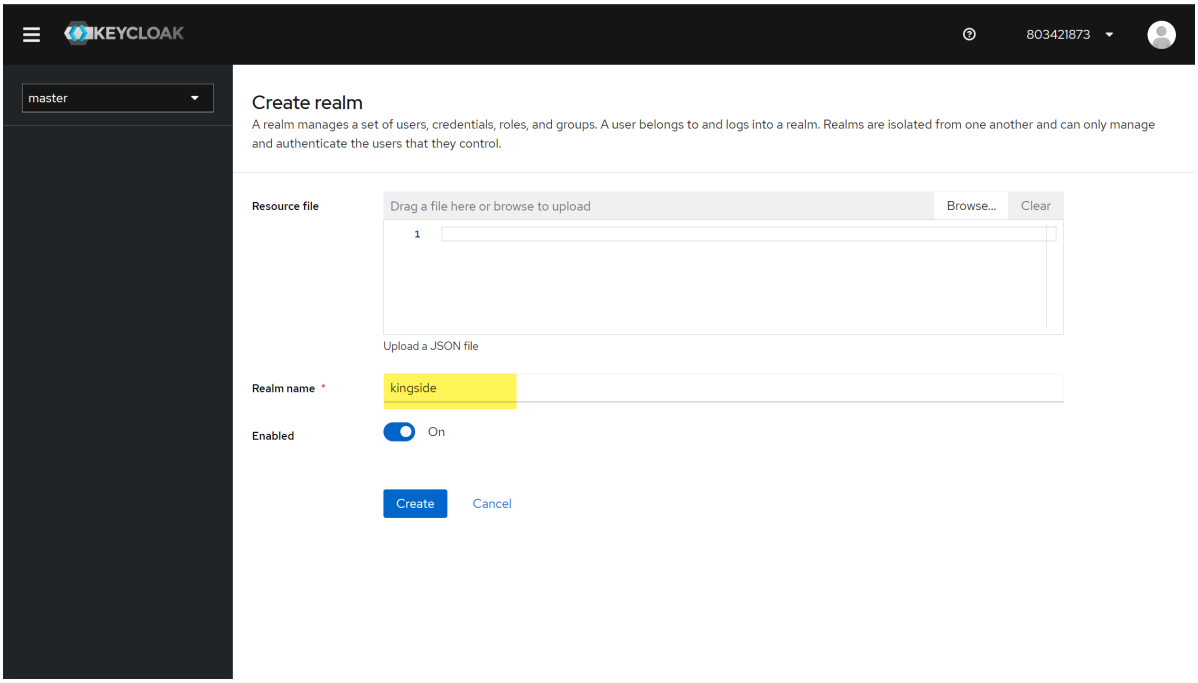
# 2. Keycloak einrichten

1. Keycloak Administarion console aufrufen über http://localhost:8080

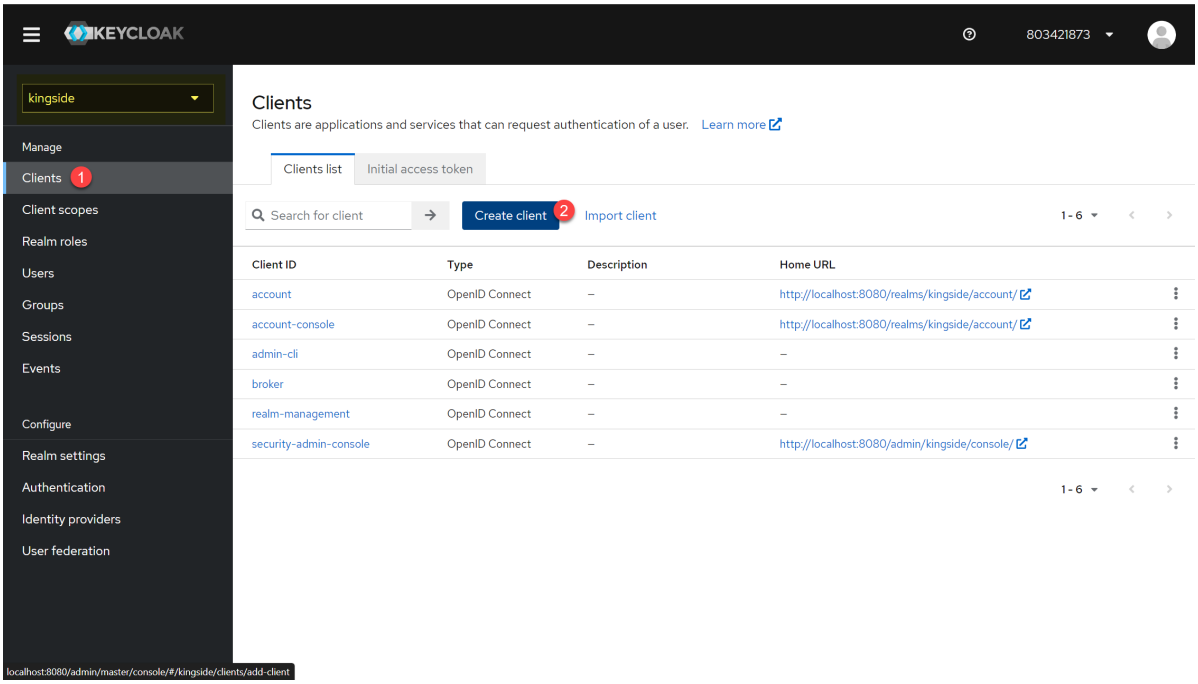


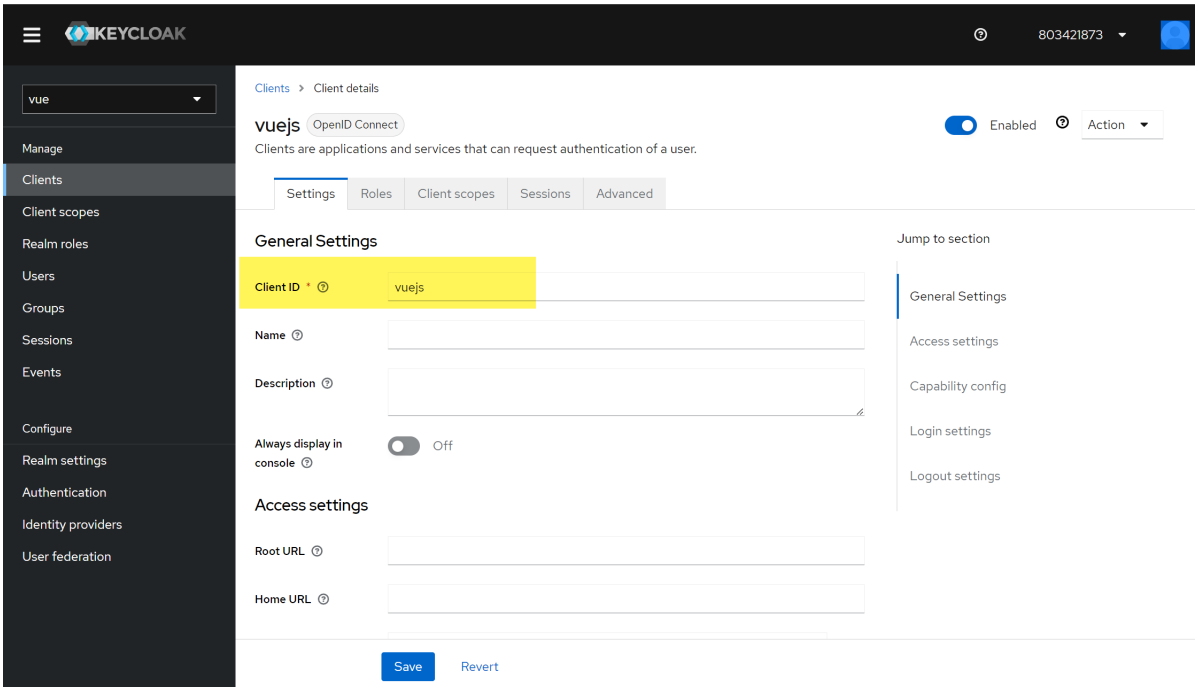

2. Neuen realm erstellen und nur **Realm name** eintragen. Wichtig! Keine Sonderzeichen benutzen

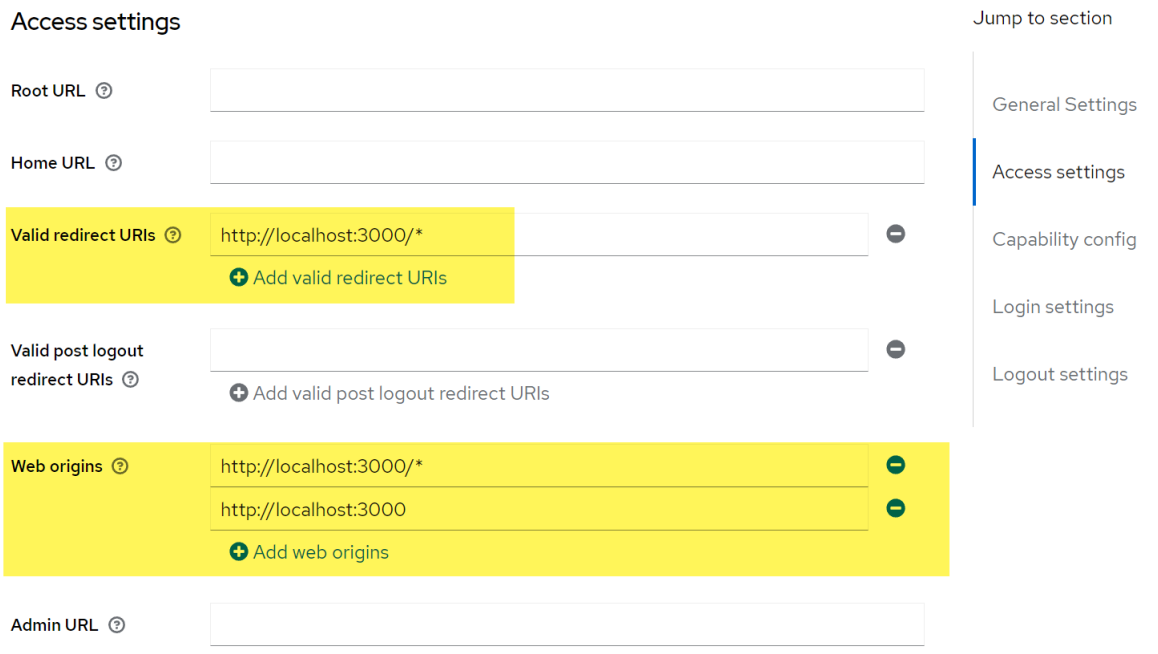

KEYCLOAK
master
1
Test
master
vue
Create Realm
2
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation




3. Jetzt in dem angelegten realm einen neuen Client erstellen




KEYCLOAK
803421873
vue
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details
vuejs
OpenID Connect
Clients are applications and services that can request authentication of a user.
Enabled
Action
Settings
Roles
Client scopes
Sessions
Advanced
General Settings
Client ID *
vuejs
Name
Description
Always display in console
Off
Access settings
Root URL
Home URL
Save
Revert
Jump to section
General Settings
Access settings
Capability config
Login settings
Logout settings


Access settings
Root URL
Home URL
Valid redirect URIs
http://localhost:3000/*
Add valid redirect URIs
Valid post logout redirect URIs
Add valid post logout redirect URIs
Web origins
http://localhost:3000/*
http://localhost:3000
Add web origins
Admin URL
Jump to section
General Settings
Access settings
Capability config
Login settings
Logout settings

## Capability config

Client authentication: Off

Authorization: Off

Authentication flow:
- [x] Standard flow
- [ ] Direct access grants
- [ ] Implicit flow
- [ ] Service accounts roles
- [ ] OAuth 2.0 Device Authorization Grant
- [ ] OIDC CIBA Grant

## Login settings

Login theme: Choose...

Consent required: Off

Display client on screen: Off

## Logout settings

Front channel logout: On

Front-channel logout URL:

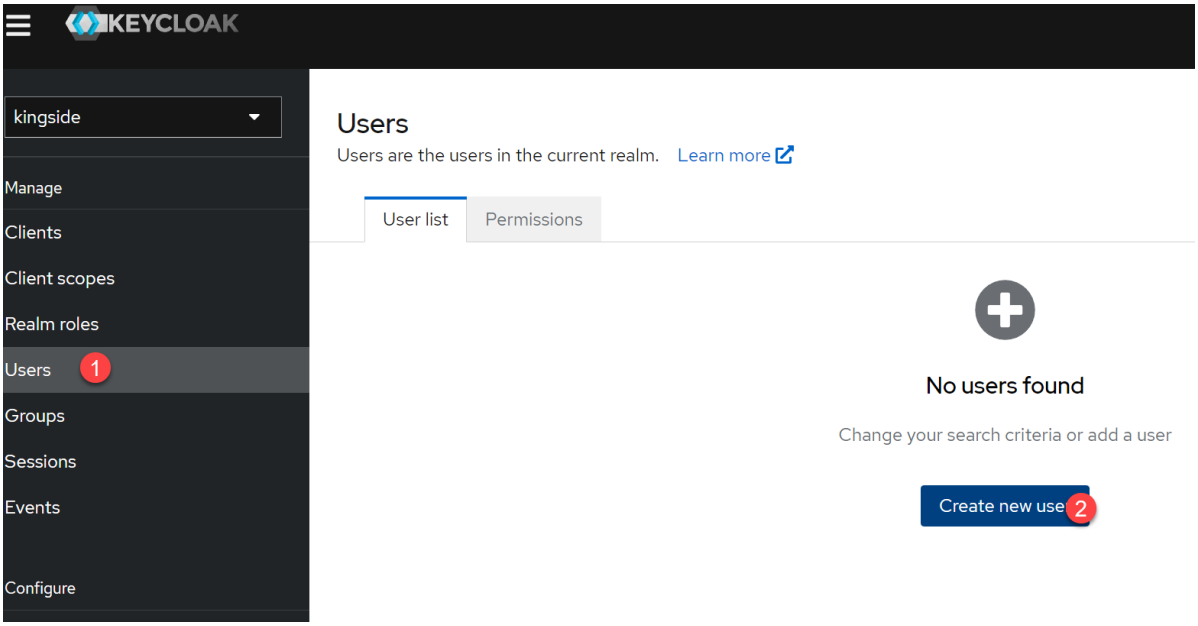
Backchannel logout URL:

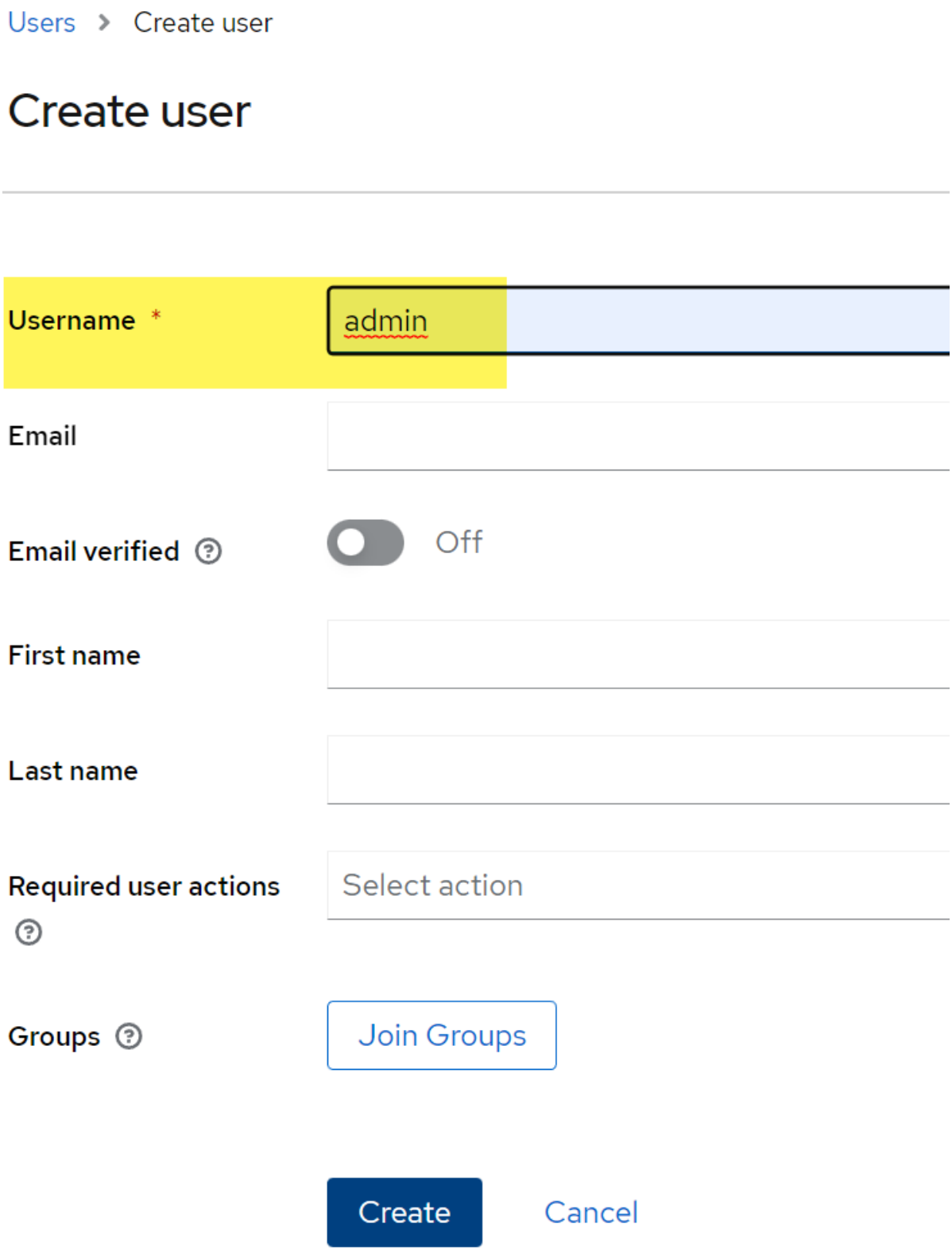
Backchannel logout session required: On

Backchannel logout revoke offline sessions: Off

Save Revert

4. User erstellen. Der **Username** reicht






5. Zu dem User wechseln und folgende Sachen um **Credentials** ergänzen

Users > User details

# admin

Details | Attributes | Credentials | Role mappir

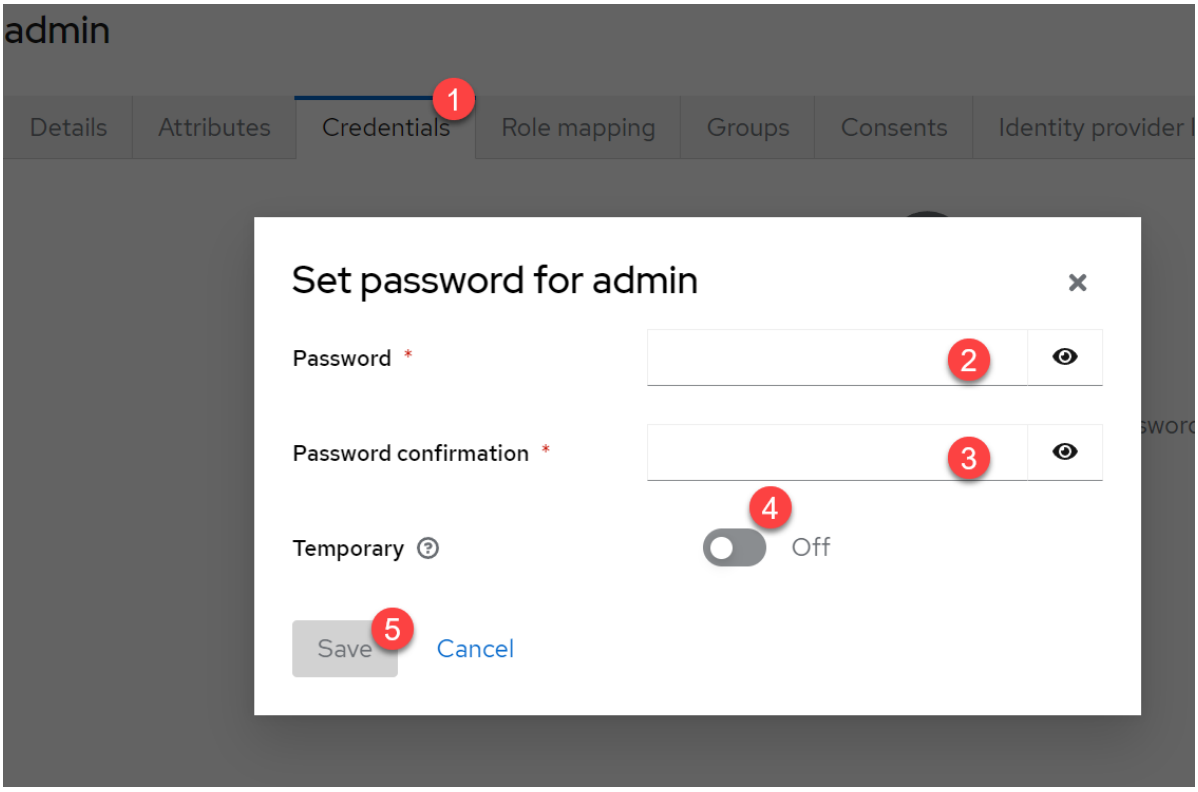
ID *: f6c242e6-cab5-4323-8d76-96

Created at *: 1/30/2023, 10:50:24 AM

Username *: admin

Email:

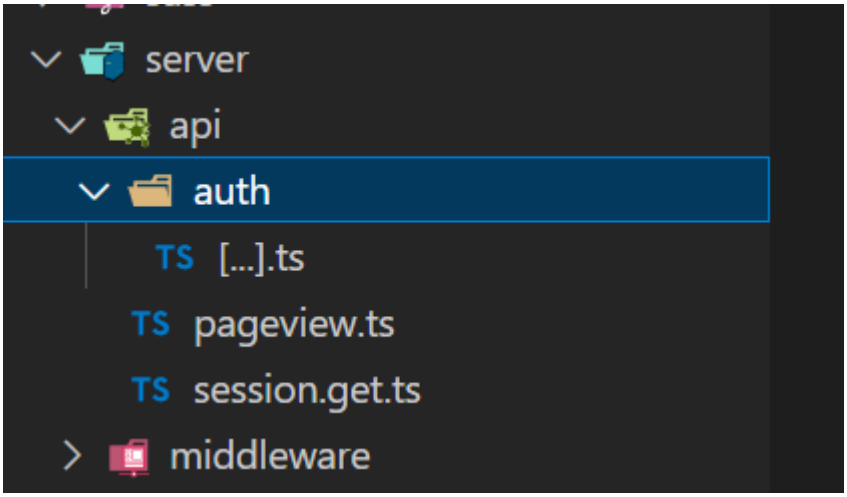
Email verified: On




# 3. nuxt 3 App aufsetzen

Installiertes nodeJs und pnpm sind Voraussetzung

1. Visual studio Code starten und das prime-vue starter repository clonen: https://github.com/sfxcode/nuxt3-primevue-starter.git
2. Verzeichnis auswählen wo die repo hin soll und mit **öffnen** bestätigen
3. Wir wechseln zum Terminal und installieren sidebase nuxt-auth. Zur Zeit des Schreibens existiert auch eine 0.4.0 alpha Version. Diese Version verursacht aber Fehler mit Keycloak.: [crayon-6766ea611c5e2218819453/]
4. nun in der nuxt.config.ts die modules ergänzen und nuxt mitteilen, dass Autorisierung auf jeder Seite gelten soll: [crayon-6766ea611c5e6379224176/]
5. Jetzt muss unter server 2 Dateien erstellt werden:




1. **server\auth\[…].ts**
   [crayon-6766ea611c5e7287054125/]
   und 2. **server\api\session.get.ts**
   [crayon-6766ea611c5e9016823455/]
6. Eine weitere Datei muss angelegt werden für die Umgebungsvariablen: **.env**
   [crayon-6766ea611c5ea654531886/]
   **Lasst euch ein Secret irgendwo generieren. Dieser String ist nur für Demozwecke!!!**

   Wichtig ist, dass die lokale Adresse mit 127.0.0.1 angegeben wird. localhost oder 0.0.0.0 kann nicht aufgelöst werden. In den Url ist der real name enthalten.

# Weitere Links

Schritt für Schritt Anleitung zur lokalen Einrichtung von Keycloak und einer Vue Applikation:

Security in Vuejs 3.0 with authentication and authorization by KeyCloak Part 1 | by Nicolas Barlatier | Dev Genius

Documentation Nuxt-Auth von Sidebase:

Quick Start · sidebase

Nuxt-Auth wurde von Next-Auth portiert. Fehlende Dokumentation kann hier nachgelesen werden:

Introduction | NextAuth.js (next-auth.js.org)

Ich nutze das prime-vue starter Paket, welches nuxt 3 inkludiert hat. Ich kann das Paket jeden nur empfehlen!

sfxcode/nuxt3-primevue-starter: Build your VUE.js App with Nuxt3 . First Class PrimeVUE support. Formkit Validation included. (github.com)

Für OAuth und OpenId Connectt gibt es Debugger. Der Author ist Nathon aus dem YouTube Video weiter oben.

OAuth 2.0 debugger (oauthdebugger.com)

OpenID Connect debugger (oidcdebugger.com)

Keycloak Dokumentation:

Documentation – Keycloak