

How to Connect to the SQL Server from the Windows 10 Enclave Virtual Desktop Using RStudio

Introduction

This document offers tips for connecting to SQL server from the Windows Enclave Virtual Desktop using RStudio, Version 4.0.4. Note that this solution has been validated using the Windows 10 Enclave Virtual Desktop connection to a MS Windows SQL Server with either ODBC, RODBC or JDBC.

Requirements or Prerequisites

1. Access to Enclave Platform should already be granted and your workspace provisioned. If you do not have approval to access the Enclave Platform, complete and submit the [Access Request Form](#).
2. You should be logged on to your Enclave Windows Virtual Desktop. You cannot access the Enclave Platform resources from outside the platform.

Step-by-Step Instructions

Depending on how you want to run R scripts to connect to the SQL server, there are three drivers you can use. These are ODBC, RODBC or JDBC drivers.

Step 1. Install ODBC, RODBC or JDBC Drivers

Installing ODBC, RODBC or JDBC drivers should be as easy as this:

ODBC Driver

Download ODBC Driver

The Microsoft ODBC Drivers for SQL Server are stand-alone ODBC drivers which provide an application programming interface (API) implementing the standard ODBC interfaces to Microsoft SQL Server.

The Microsoft ODBC Driver for SQL Server can be used to create new applications. You can also upgrade your older applications which currently use an older ODBC driver. The ODBC Driver for SQL Server supports connections to Azure SQL Database, Azure Synapse Analytics, and SQL Server.

Version	Features Supported
Microsoft ODBC Driver 18 for SQL Server	<ul style="list-style-type: none">• Support for TDS 8.0• Extensions to SQLGetData• Option to send SQL_LONG_* types as (max)-types
Microsoft ODBC Driver 17 for SQL Server	<ul style="list-style-type: none">• Always Encrypted support for BCP API• New connection string attribute UseFMTONLY causes driver to use legacy metadata in special cases requiring temp tables
Microsoft ODBC Driver 13.1 for SQL Server	<ul style="list-style-type: none">• Always Encrypted• Azure AD Authentication• Always On Availability Groups (AG)
Microsoft ODBC Driver 13 for SQL Server	<ul style="list-style-type: none">• Internationalized Domain Name (IDN)
Microsoft ODBC Driver 11 for SQL Server	<ul style="list-style-type: none">• Driver-Aware Connection Pooling• Connection Resiliency• Asynchronous execution (Polling Method)

Summary

- **RODBC Driver**

RODBC is simple to install, and binary distributions are available for Windows from CRAN. <https://cran.r-project.org/web/packages/RODBC/index.html>

To install R packages in RStudio, simply run the following R script:

```
install.packages("RODBC")
```

- **JDBC Driver**

JDBC creates a new DBI driver that can be used to start JDBC connections.

JDBC function has two purposes. One is to initialize the Java VM and load a Java JDBC driver (not to be confused with the JDBCdriver R object which is actually a DBI driver). The second purpose is to create a proxy R object which can be used to call dbConnect which actually creates a connection.

JDBC requires a JDBC driver for a database-backend to be loaded. Usually, a JDBC driver is supplied in a Java Archive (jar) file. The path to such a file can be specified in classPath. The driver itself has a Java class name that is used to load the driver (for example the MySQL driver uses com.mysql.jdbc.Driver), this has to be specified in driverClass.

```
JDBC (driverClass = "", classPath = "", identifier.quote = NA)
```

The RJDBC package is an implementation of R's DBI interface using JDBC as a backend. This allows R to connect to any DBMS that has a JDBC driver.

```
install.packages("DBI", dep=TRUE)
install.packages("rJava", dep=TRUE)
install.packages("RJDBC", dep=TRUE)
```

Step 2. Specifying DSNs

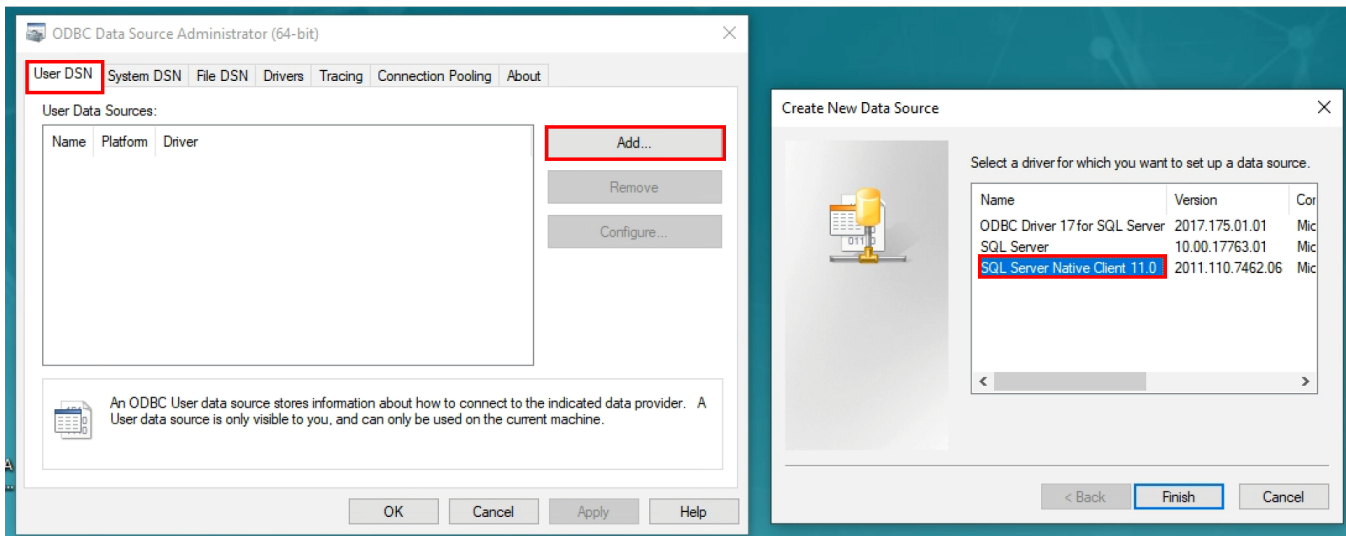
The ODBC driver managers have 'User DSNs' and 'System DSNs': these differ only in where the information is stored, the first on a per-user basis and the second for all users of the system.

Windows has a GUI you can use to set up the DSNs. For example, to set up the ODBC Data Source, go to the **Start** menu and type "ODBC". When "**Microsoft ODBC Administrator**" or "**Data Sources (ODBC)**" appears under **Programs**, click on it. Alternatively, go to the **Start** menu and select "**Administrative Tools**". In this step, you can add, remove and edit ('configure') DSNs as illustrated in the screenshots below.

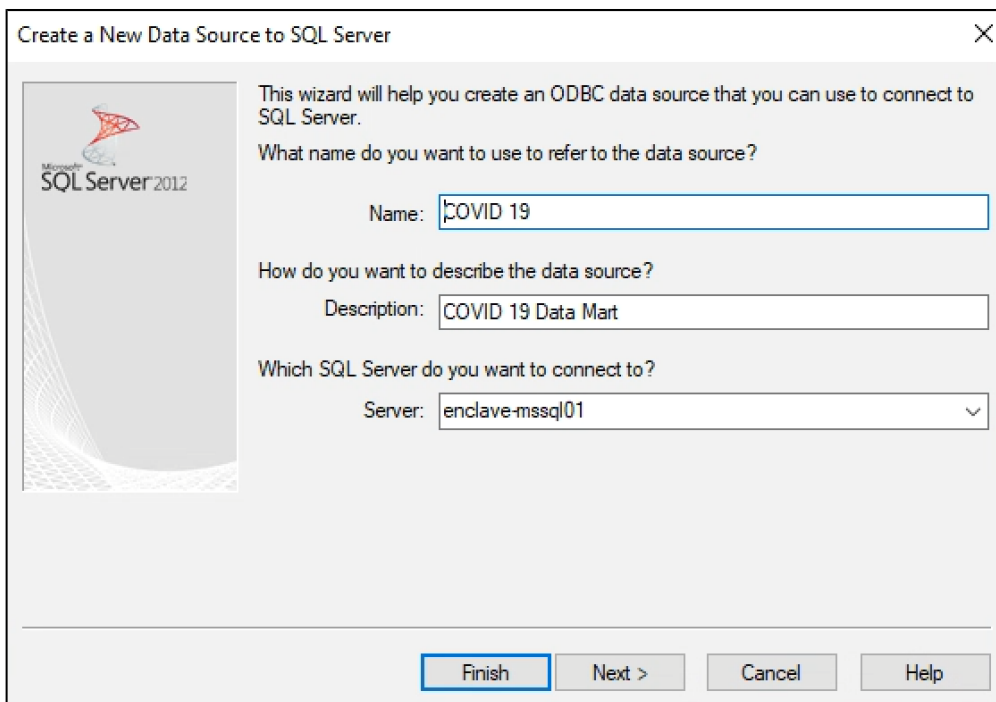
You need to repeat these steps for each database you are planning to use

In the **ODBC Data Source Administrator** window, select the **User DSN** tab, click the **Add** button.

Select **SQL Server Native Client 11.0** from the list of drivers and then click **Finish**.




Type the name you want to use for your data source, add a description of the data source, and the name of the Server you want to connect to (enclave-mssql01 in this example). Click **Next** to continue.



Verify that "With Integrated Windows authentication" option is selected. Click **Next** to continue.

Microsoft SQL Server DSN Configuration



How should SQL Server verify the authenticity of the login ID?

With Integrated Windows authentication.

SPN (Optional):

With SQL Server authentication using a login ID and password entered by the user.


Login ID:

Password:

< Back Next > Cancel Help

Check "Change the default database to" box and then choose the name of the database to which you want to connect from the drop-down (COVID19_Mart in this example). Click Next to continue.

Microsoft SQL Server DSN Configuration



Change the default database to:

Mirror server:

SPN for mirror server (Optional):

Attach database filename:

Use ANSI quoted identifiers.

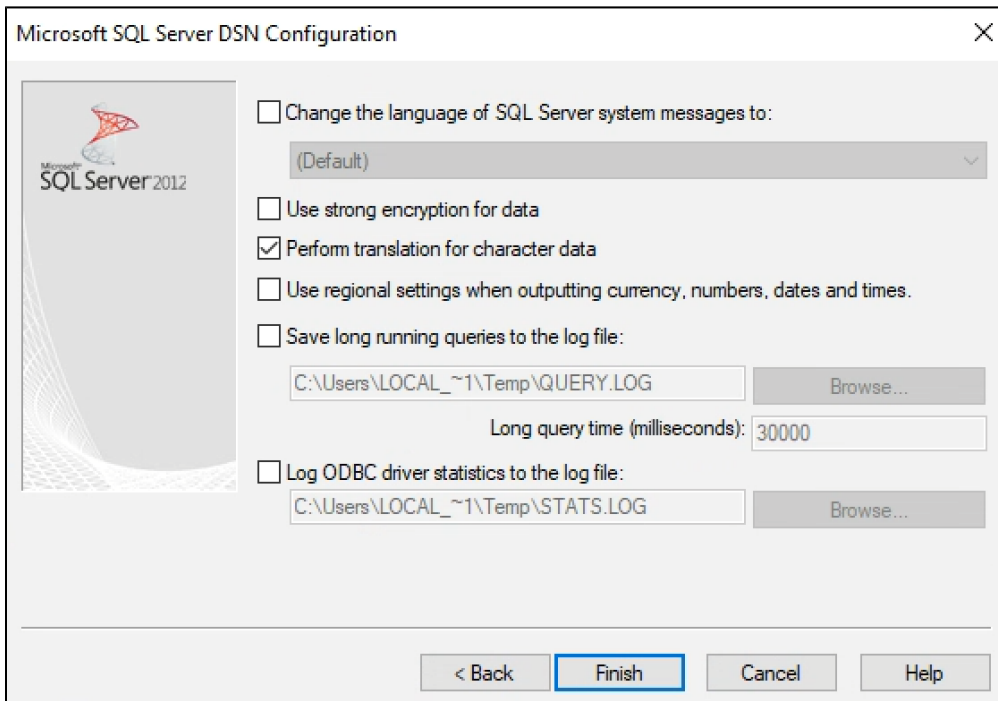
Use ANSI nulls, paddings and warnings.

Application intent:

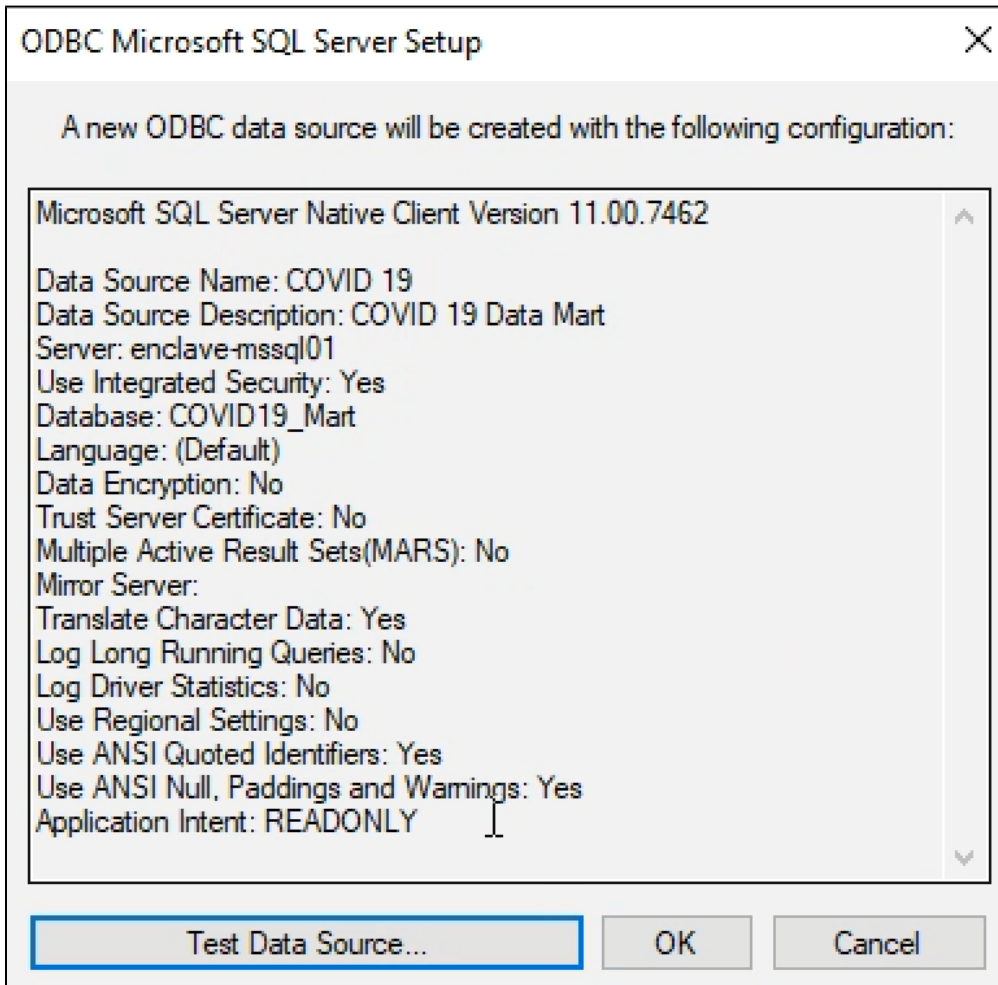
Multi-subnet failover.

< Back Next > Cancel Help

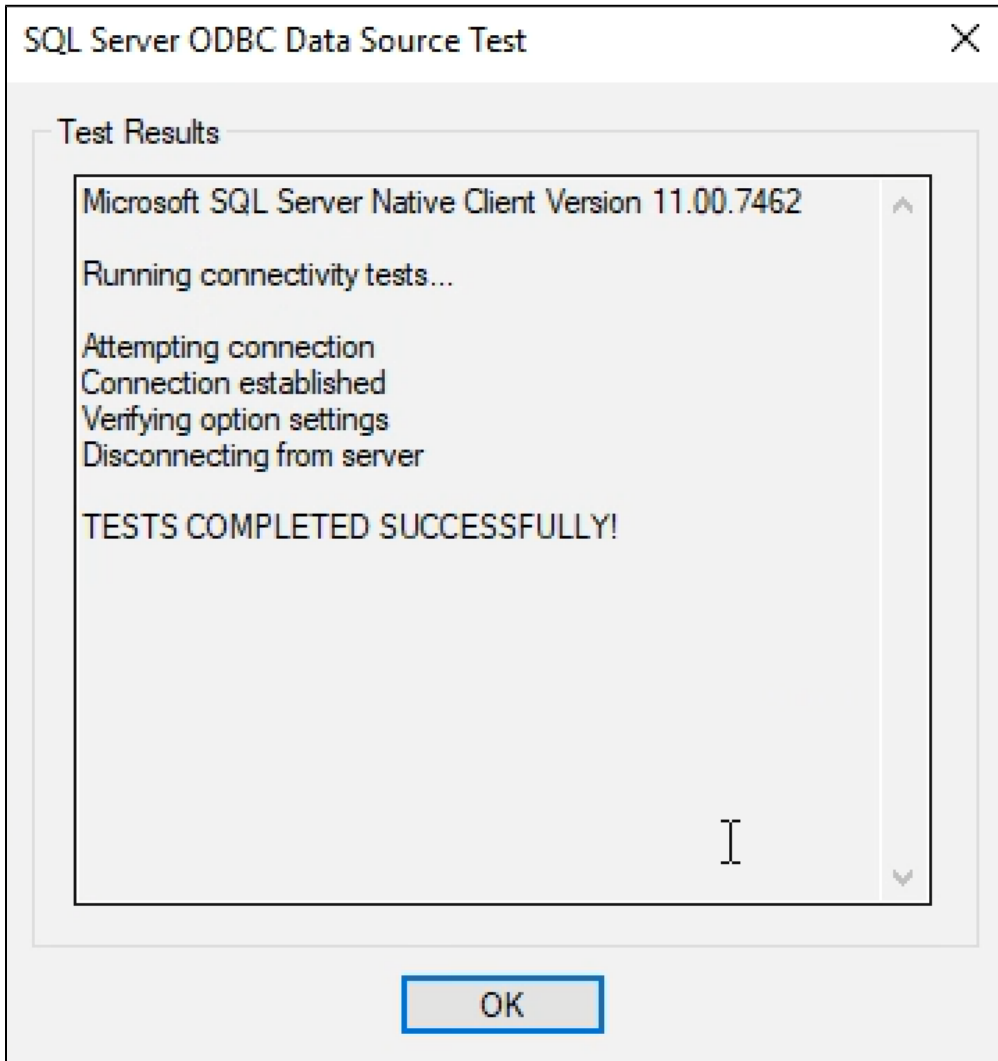
Verify general configuration settings for the ODBC connection and click **Finish**.



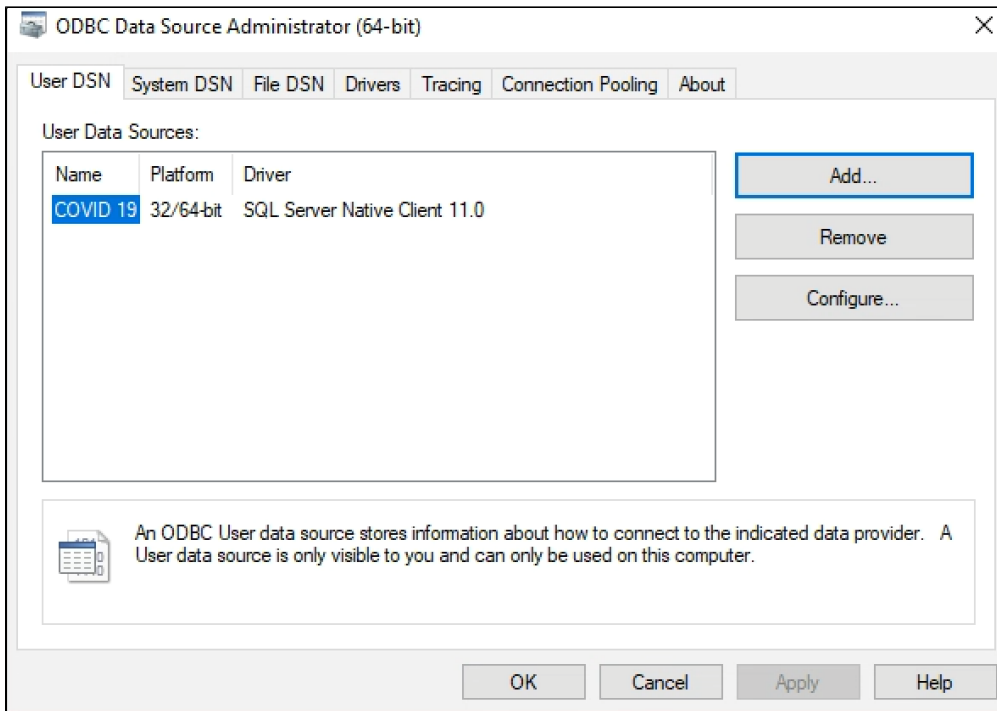
Once the summary of configuration for the new data source is created, test the data source by clicking the **Test Data Source** button.



If everything works as expected, the confirmation screen will appear as indicated below. Click the **OK** button to close the test screen window. Click **OK** button again to close the creation window.



Once the ODBC data source has been successfully created, click the **OK** button.



Step 3. Making a MS SQL Server Connection

ODBC works by setting up a connection or channel from the client (here RODBC) to the DBMSs as specified in the DSN. Such connections are normally used throughout a session, but should be closed explicitly at the end of the session—however RODBC will clear up after you if you forget (with a warning that might not be seen in a GUI environment).

There can be many simultaneous connections.

The simplest way to make a connection is

```
library(RODBC)
cn <- odbcConnect("some dsn")
```

and when you are done with it,

```
close(cn)
```

or if you prefer

```
odbcClose(cn)
```

Here three sample codes. These codes provides a template for connecting to an existing database. In this example the database is i2b2.

The first example is a very simple R program: ***"odbc connect to SQL Server using DSN Covid 19.R"***

```
library(RODBC)
cn <- odbcConnect("COVID 19")
dataSQLQuery <- sqlQuery(cn, "SELECT * FROM i2b2.dbo.ACT_COVID")
View(dataSQLQuery)
```

It uses the DSN created by following Step 2 above or the PDF files that is attached.

The second R program: "**RODBC Connect to SQL Server i2b2 database from R**"

```
libPaths()
local({r <- list("cran" = "http://repo.analyticsenclave.org:8082/artifactory/cran")})
getOption("repos")
install.packages('RODBC')
library('RODBC')
cn <- odbcDriverConnect(connection="Driver={SQL Server Native Client 11.0};server=mssql01-t.analyticsenclave.
org;database=i2b2;trusted_connection=yes;")
dataSQLQuery <- sqlQuery(cn, "SELECT * FROM i2b2.dbo.ACT_COVID")
View(dataSQLQuery)
```

It uses odbcDriver Connection function with the connection strings.

The third R program: "**RJDBC_enclave - RJDBC to query i2b2 COVID table.R**"

```
install.packages("DBI",dep=TRUE)
install.packages("rJava",dep=TRUE)
install.packages("RJDBC",dep=TRUE)
library('DBI')
library('rJava')
library('RJDBC')
drv <- JDBC("com.microsoft.sqlserver.jdbc.SQLServerDriver", "c:/Program Files/sqljdbc_8.2/enu/mssql-jdbc-8.2.2.
jre8.jar", identifier.quote = "")
cn <- dbConnect(drv, "jdbc:sqlserver://mssql01.analyticsenclave.org:1433; databaseName=COVID19_Mart;
domain=analyticsenclave.org;IntegratedSecurity=true")
dataSQLQuery <- dbGetQuery(cn, "SELECT * FROM i2b2.dbo.ACT_COVID")
View(dataSQLQuery)
```

Limitations

Please Note: The above code was tested to run successfully under **R 4.0.4**

Related References

1. [Connect to an ODBC Data Source \(SQL Server Import and Export Wizard\) - SQL Server Integration Services \(SSIS\) | Microsoft Docs](#)
2. [Setting up R to connect to SQL Server – RStudio Support](#)
3. [Working with a JDBC connection - JDBC Driver for SQL Server | Microsoft Docs](#)

Additional Resources

How to connect to Oracle using ODBC

1. Start the database connection wizard.
2. Select Oracle (ODBC / JDBC), and then click Next.
3. Select ODBC.
4. Click Edit Drivers.

5. Select the Oracle drivers you wish to use (in this example, Oracle in OraClient11g_home1). ...
6. Click Back.
7. Select Create a new data source name (DSN) with the driver, and then select the Oracle driver chosen in step 4. ...
8. Click Connect.

Related documents

- [Connect R \(RStudio\) with RJDBC to Microsoft SQL Server Express 2008 R2 – Incentergy – Effizienzsteigerndes Projektmanagement](#)