



DSci | Study
& Research

Data Science Journey

Mathematics
/ \
Domain Knowledge — Computer Science

4 years have passed

since the start of the journey on 3 January 2020.

Inhaltsverzeichnis

I	Artificial Intelligence	3
1	Azure ML Python Development	4
0.1	Understanding ipykernel	4
0.1.1	Overview Python Interpreter	4
	High-Level Language	4
	Difference between interpreter and a compiler	5
0.1.2	ipykernel (Jupyter Notebook)	5
	IPython (Python3)	5
	Jupyter Notebook	6
0.1.3	Interacting with different (language) Kernels	6
	Theoretical Multi languages Kernel	6
	VSCoDe Compute Cells Confusion	7
	Azure Data Studio	7
0.1.4	VEN and ipykernel	7
	Spinning Up Jupyter Notebook with Anaconda	7
	Multiple Kernel (Python)	8
0.1.5	Connecting a Notebook to a Compute Ressources	9
1	Using Azure ML API/ SDK/ CLI	9
1.1	Introduction	9
1.2	Basics	10
1.2.1	Working with conda	10
	Interacting with conda from the terminal	10
1.3	Running ON the Compute Instance	12
1.3.1	Single Jupyter Notebook Connection	12
	Problem & Fix: Unable to connect	12
1.3.2	Tunnel Connection	13
1.4	Running WITH the Compute Instance (SDK v1)	14
1.4.1	Using a local kernel/ Python Interpreter	14
II	Anhang	15
A	Abkürzungsverzeichnis	16

Teil I

Artificial Intelligence

Kapitel 1

Azure ML Python Development

0.1 Understanding ipykernel

0.1.1 Overview Python Interpreter

Reference: [Using the Python Interpreter, What Is a Python Interpreter?](#)

High-Level Language The Python language is a high-level programming language. This means, that the programming language is more similar to human language than to machine language, which is written in strings of bits - ones and zeros. The advantage of it is that it is better understood by humans, but the same can not be said for the machines. Writing commands (instruction) in lines of zeros and ones is more difficult, that using higher level concepts. The gap between this, gets closes by the **Python Interpreter**.

The Python Interpreter reads the command written by the Python programmer, evaluates them, and returns the output. If the *python.exe* application/ interpreter is opened, either through the command-line interpreter (cmd) , for example

```
1 $ python3.12
2 ///Python 3.12 (default, April 4 2022, 09:25:04)
3 ///[GCC 10.2.0] on linux
4 ///Type "help", "copyright", "credits" or "license" for more
   information.
5 >>>
```

the commands

```
1 >>> the_world_is_falt = True
2 >>> if the_world_is_flat:
3     print("Be careful not to fall off!")
4 /// Be careful not to fall off!
```

or by opening the application directly.



Abbildung 1.3: IPython interactive command-line terminal

Jupyter Notebook With the reorganization of *IPython*, the new tool **Notebook** has been created. Under the project name **Jupyter**. This Jupyter Notebook (Web Interface) is a web interface for Python. It has the same interactive interface kept. Being a web-interface, it can integrate with many of the existing web libraries for data visualization.

The concept of a *kernel* comes into play as the engine behind the web interface. The *IPython* is now the backend with the Kernel (IPython/ Jupyter) for Python. The Kernel (IPython/ Jupyter) with the advent of Jupyter Notebook (Web Interface) is able to handel *markdown* and \LaTeX text input.

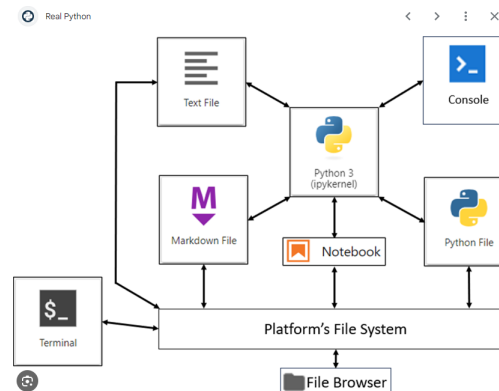


Abbildung 1.4: JupyterLab for an Enhanced Notebook, *Jupyter Lab*

Note: The interaction with the Kernel (IPython/ Jupyter) is done through the Jupyter Notebook (Web Interface). With the latest tool: **Jupyter Lab**, interaction with separate files is possible.

”Inside” the Kernel (IPython/ Jupyter) lives the Python Interpreter. Another way of saying this is that the Kernel (IPython/ Jupyter) is the interface to the Python Interpreter.

Note: *Jupyter Lab* allows for a more interactive and simultaneous way to code.

0.1.3 Interacting with different (language) Kernels

Theoretical Multi languages Kernel As said before the Kernel (IPython/ Jupyter) is for interacting with the Python Interpreter and the other functionality of a Jupyter Notebook (Web Interface).

The community around, *Jupyter Notebook*, the application, developed more Kernel (General Jupyter) for the Jupyter Notebook (Web Interface). Those Kernel (General Jupyter) allows to interact with different languages like *Ruby*, *Scale*, *R*.

It would be possible (Unclear how) to create a Kernel (General Jupyter) which allows to interact with all those languages in one Jupyter Notebook (Web Interface). The current research for this chapter could not found one. For example the Kernel (IPython/ Jupyter) allows us to interact for example with Python, Markdown, Shell Script. This does not conclude for example Structured Query Language (SQL).

VSCode Compute Cells Confusion If a Jupyter Notebook (Web Interface) is open, for each cell it is possible to select the language for this cell. This those two things. First it changes the *IntelSense* syntax highlights. Secondly, it provides the Kernel (General Jupyter) information about the language which is used in this cell.

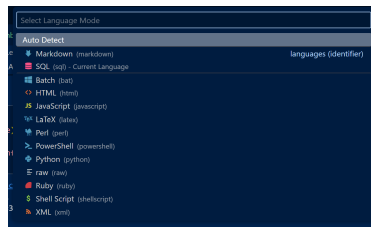


Abbildung 1.5: Cell Language Mode selection.

However, this does not mean that all languages are supported by the selected Kernel (General Jupyter).

Azure Data Studio In Azure Data Studio you can connect to different Kernel (General Jupyter), *Azure Data Studio*:

- SQL Kernel,
- PySpark3 and PySpark Kernel,
- Spark Kernel,
- Python Kernel (for local development)

0.1.4 VEN and ipykernel

Spinning Up Jupyter Notebook with Anaconda

- First a conda ven gets created.
- The Jupyter Notebook "application" gets installed

This in turn will install the package for the web interface, *IPython* and the default Kernel (IPython/ Jupyter) for Python.

```
1 pip install jupyter
```

Listing 1.1: pip commands to get ready for Jupyter Notebook

This command will import all dependence to work with a Kernel (IPython/ Jupyter). An example of package environment see below.

```

1 PS C:\Users\PaulJulitz\iCloudDrive\TexMaker\GitHub_Notizen_DSci\
   Notizen_DSci> conda list
2 # packages in environment at C:\Users\PaulJulitz\anaconda3\envs\VSCode:
3 #
4 # Name                               Version                               Build    Channel
5 ...
6 azure-common                        1.1.28                               pypi_0   pypi
7 azure-core                          1.27.1                               py38haa95532_0
8 azure-identity                      1.12.0                               pypi_0   pypi
9 azure-keyvault-secrets              4.6.0                               pypi_0   pypi
10 azureml                             0.2.7                               pypi_0   pypi
11 ...
12 ipykernel                          5.3.4                               py38h5ca1d4c_0
13 ipython                             7.27.0                               py38hd4e2768_0
14 ipython_genutils                   0.2.0                               pyhd3eb1b0_1
15 ...
16 jupyter_client                     7.0.1                               pyhd3eb1b0_0
17 jupyter_core                       4.8.1                               py38haa95532_0
18 jupyter_server                     1.4.1                               py38haa95532_0
19 jupyterlab                         3.2.1                               pyhd3eb1b0_1
20 jupyterlab_pygments                0.1.2                               py_0
21 jupyterlab_server                  2.8.2                               pyhd3eb1b0_0
22 ...

```

Listing 1.2: Example Jupyter Notebook conda ven

Multiple Kernel (Python) If the setup is done through the Anaconda interface (conda). The complete installation is done by Anaconda if the IDE Jupyter Notebook (Web Interface) is installed. By default *IPython* is installed with the Kernel (IPython/ Jupyter).

The command for the installation of the Kernel (IPython/ Jupyter) is

```

1 pip install ipykernel

```

Listing 1.3: pip to install ipykernel

For example to use Scala, [Link](#):

```

1 # Install the package
2 pip install spylon-kernel
3 # To select the kernel in the notebook, create a kernel spec
4 python -m spylon_kernel install
5 # Start Jupyter Notebook
6 ipython notebook

```

Listing 1.4: Using Scala for a Notebook

In the notebook the kernel can be selected. The command is to list the available kernels is

```

1 $ jupyter kernelspec list
2 >>>Available kernels:
3 >>>python2.7      /Users/jakevdp/.ipython/kernels/python2.7
4 >>>python3.3      /Users/jakevdp/.ipython/kernels/python3.3
5 >>>python3.4      /Users/jakevdp/.ipython/kernels/python3.4
6 >>>python3.5      /Users/jakevdp/.ipython/kernels/python3.5
7 >>>python2        /Users/jakevdp/Library/Jupyter/kernels/python2
8 >>>python3        /Users/jakevdp/Library/Jupyter/kernels/python3

```

Listing 1.5: pip to install ipykernel

See, [Running Multiple Kernel](#) for understand how to select a Kernel (General Jupyter) through the command line.

To see which Kernel (General Jupyter) is running, run the following code

```
1 import sys
2 print(sys.executable)
3 print(sys.version)
4 print(sys.version_info)
```

0.1.5 Connecting a Notebook to a Compute Ressources

To use a Jupyter Notebook (Web Interface), the web interface creates either a

- local server,
- or provides the possibility to connect to a Jupyter hosted server.

Note: The local server is still your own machine.

The packages for connecting to a server are installed in the `venv`. The `spinningup` is basically the configuration to either a local port on your machine or to a hosted server. The former is used for computing.

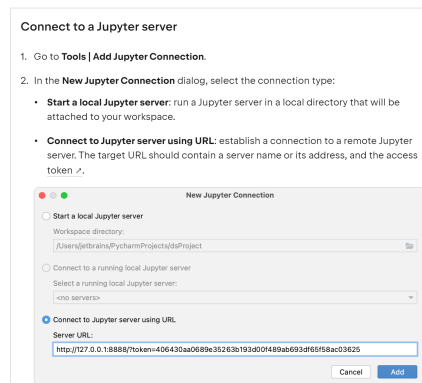


Abbildung 1.6: Mac Setup of Jupyter Notebook Server

If the notebook should be connected to the compute cluster, this is done by a SSH Tunnel.

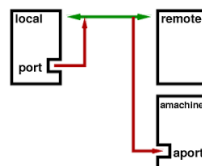


Abbildung 1.7: Tunnel Traffic, [SSH Tunnel](#)

1 Using Azure ML API/ SDK/ CLI

1.1 Introduction

The Python `azureml-SDK v2/ azure-ai-ml (AMLSDKv2)` with the package library

azure-ai-ml

and the azureml-SDK v1/ azureml-core (AMLSDKv1) with

azureml-core

are Application programming interface (API) s for the Azure ML (Azure ML) workspace and it's services. The Command Line Interface (CLI) package for AML contains a more compact command line style workflows, that need to be executed. The format in v2 is normally *<noun><verb><option>*.

The Software Development Kit (SDK) is more directed to the development, while the CLI is more convient for a Continous Integration (CI) / Continous Development (CD) process. The later is due to the more compact way of executing commands.¹

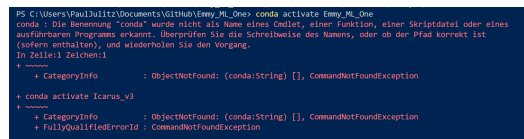
1.2 Basics

1.2.1 Working with conda

Interacting with conda from the terminal Here are three ways to interact with the python interpreter:

conda comand Starting the *Anaconda Navigator* and start the IDE from there. This then allows to interact with conda throug the **conda Cmdlet** from the terminal.

PowerShell Extension If the local machine has some restriction, that prohibist the use of the command **conda** from the powershell terminal.

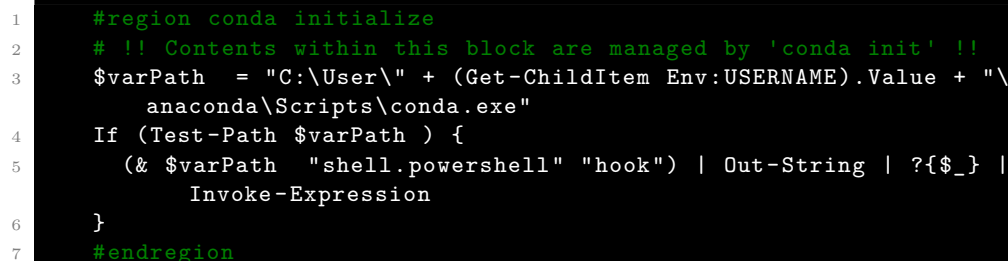


```
PS C:\Users\PaulJulitt\Documents\Github\Emy_M_One> conda activate Emy_M_One
conda : Die Bezeichnung "conda" wurde nicht als Name eines Cmdlet, einer Funktion, einer Skriptdatei oder eines
ausführbaren Programms erkannt. Überprüfen Sie die Schreibweise des Namens, oder ob der Pfad korrekt ist
(sodass er enthalten), und wiederholen Sie den Vorgang.
In Zeile:1 Zeichen:1
+ ~~~~~
+ CategoryInfo          : (ObjectNotFound: (conda:String) [], CommandNotFoundException)
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\PaulJulitt\Documents\Github\Emy_M_One> conda activate Icarus_v3
conda : Die Bezeichnung "conda" wurde nicht als Name eines Cmdlet, einer Funktion, einer Skriptdatei oder eines
ausführbaren Programms erkannt. Überprüfen Sie die Schreibweise des Namens, oder ob der Pfad korrekt ist
(sodass er enthalten), und wiederholen Sie den Vorgang.
In Zeile:1 Zeichen:1
+ ~~~~~
+ CategoryInfo          : (ObjectNotFound: (conda:String) [], CommandNotFoundException)
+ FullyQualifiedErrorId : CommandNotFoundException
```

Abbildung 1.8: Terminal Response by using conda command

Still, the following powershell script allows when execute to use a *powershell extension*, that bypass this problem.



```
1 #region conda initialize
2 # !! Contents within this block are managed by 'conda init' !!
3 $varPath = "C:\User\" + (Get-ChildItem Env:USERNAME).Value + "
   anaconda\Scripts\conda.exe"
4 If (Test-Path $varPath ) {
5     (& $varPath "shell.powershell" "hook") | Out-String | ?{$_} |
       Invoke-Expression
6 }
7 #endregion
```

Listing 1.6: .ps1 script

This powershell extension allows you with the conda program.

¹Reference: [Difference CLI, SDK, MLOps Python AML](#)

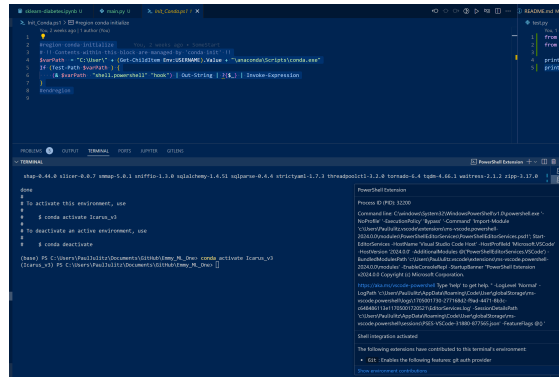


Abbildung 1.9: PowerShell Extension

One restriction still exists. In Visual Studio Code (vscode) the option to run a program requires to create a python terminal in which the **conda** command can be access. This will not work, because a new terminal will be created in which the **conda** command can't be access. However, the python interpreter in a specific conda ven can still be access by providing the direct path to it.

```
(base) PS C:\Users\PaulJulitz\Documents\GitHub\Emmy_ML_One> conda activate Icarus_v3
(Icarus_v3) PS C:\Users\PaulJulitz\Documents\GitHub\Emmy_ML_One> & C:/Users/PaulJulitz/anaconda3/envs/Icarus_v3/python.exe c:/Users/PaulJulitz/Documents/GitHub/Emmy_ML_One/test.py
None
(Icarus_v3) PS C:\Users\PaulJulitz\Documents\GitHub\Emmy_ML_One>
```

Abbildung 1.10: PowerShell Extension and Python Interpreter

```
1 (base) PS C:\Users\PaulJulitz\Documents\GitHub\Emmy_ML_One>
2 conda activate Icarus_v3
3 (Icarus_v3) PS C:\Users\PaulJulitz\Documents\GitHub\Emmy_ML_One>
4 & C:/Users/PaulJulitz/anaconda3/envs/Icarus_v3/python.exe c
5 :/Users/PaulJulitz/Documents/GitHub/Emmy_ML_One/test.py
Hello
None
(Icarus_v3) PS C:\Users\PaulJulitz\Documents\GitHub\Emmy_ML_One>
```

Listing 1.7: Commands

Pointing to the interpreter If the conda command can't be used. Then the python interpreter it self can be selected.



Abbildung 1.11: Selecting the python interpreter in vscode

Running a python script will use the python interpreter inside the selected conda ven.

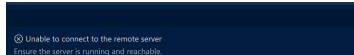


Abbildung 1.15: Error message after selecting Azure ML Compute Instance

For the moment, rollback the Jupyter Notebook (Web Interface) extension to a version v2023.10.xxx helped.²

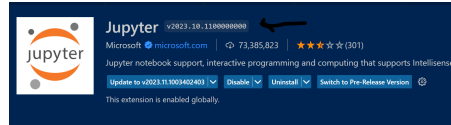


Abbildung 1.16: Rollback version

1.3.2 Tunnel Connection

With the option to connect to the workspace it self, the blobstorage, where the notebooks of each user are stored, can be access. To to this `vscode` connected via tunnel to the workspace with the compute instance in the web browser or with the desktop application.

Deutsche Bahn > Azure Machine Learning 55Bn München > Compute

Compute

The "Kubernetes clusters" tab is now where you can access previous versions of "Inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with any previously created compute targets using those types. [Learn more about Kubernetes clusters.](#)

Compute instances Compute clusters Kubernetes clusters Attached computes

Choose from a selection of CPU or GPU instances preconfigured with popular tools such as VS Code, JupyterLab, Jupyter, and RStudio, ML packages, deep learning frameworks, and GPU drivers. [Learn more about compute instances](#)

[+ New](#) [Refresh](#) [Start](#) [Stop](#) [Restart](#) [Schedule and idle shutdown](#) [Delete](#) [View options](#)

Search

Name	☆	State	Idle shutdown	Applications	Size	Created on	Assigned to
ReginaSiegers02		Create failed	---		STANDARD_DS3_V2	Dec 13, 2023 9:39 AM	Regina Siegers
AmineSassi02		Stopped	30 minutes	JupyterLab Jupyter VS Code (Web)	STANDARD_DS3_V2	Dec 12, 2023 5:30 PM	Amine Sassi
PauPaMayer02		Stopped	30 minutes	JupyterLab Jupyter VS Code (Web)	STANDARD_DS3_V2	Dec 12, 2023 5:18 PM	Paul Pa Mayer
PaulJulitz02	✓	Running	30 minutes	JupyterLab Jupyter VS Code (Web)	STANDARD_DS3_V2	Dec 12, 2023 5:04 PM	Paul Julitz

Abbildung 1.17: Create a tunnel connection

On the bottom left (1) the tunnel connection is displayed. The internal file system gets linked as the storage (2), where the notebook can be access. Note: Locally stored notebooks can be used by the previous section.

²Posted Problem, Pull Request and Workaround

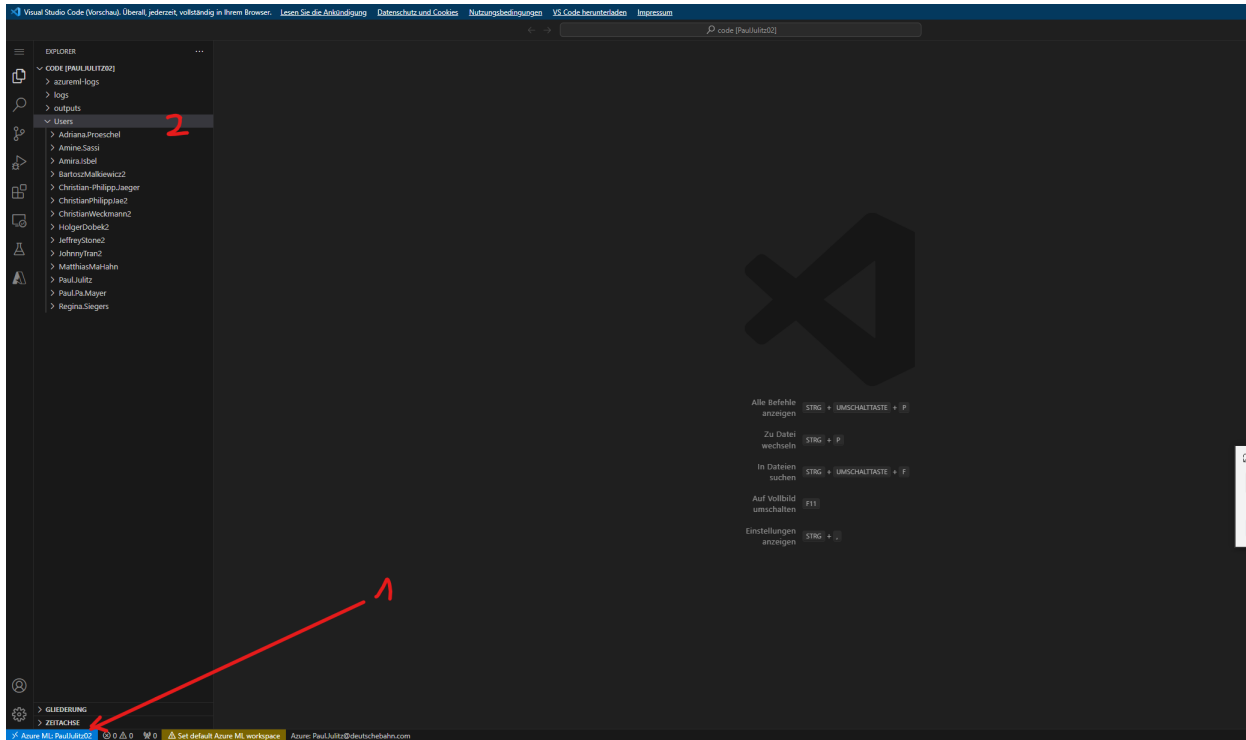


Abbildung 1.18: Create a tunnel connection

1.4 Running WITH the Compute Instance (SDK v1)

This section will explain how to use AML with your local computer as a compute resources.

1.4.1 Using a local kernel/ Python Interpreter

- Create a Python ven with conda on you local computer
- Install the newest AMLSDKv1. ³

```
1 pip install azureml-core
```

Listing 1.8: Installing Azure ML SDK core package

The SDK contains many more optional packages, see [Optional Packages for Azure ML](#)

- Installing Jupyter Notebook (Web Interface) required packages.

```
1 conda
```

Listing 1.9: Installing Azure ML SDK core package

³Because the more extensive example is setup with AMLSDKv1, we will start from here to. The idea is, that more functionally are provides by AMLSDKv2 then this will be incrementally be used. [Installation Guide](#)

Teil II

Anhang

Anhang A

Abkürzungsverzeichnis

Symbole

- .ipynb** Jupyter Notebook file format which is interoperable across many platforms. The name also refers to the user-facing web interface called Jupyter Notebook.. *Glossar:* Jupyter Notebook (Web Interface)
- .json** JavaScript Object Notation. *Glossar:* JSON

A

- AML** Azure ML. 10, 14
- AMLSDKv1** The azureml-SDK v1 with the library package azureml-core. 10, 14
- AMLSDKv2** The azureml-SDK v2 with the library package azure-ai-ml. 9, 14
- API** Application programming interface. 10

C

- CD** Continuous Development. 10
- CI** Continuous Integration. 10
- CLI** Command Line Interface. 10
- cmd** command-line interpreter. 4

E

- EC2** Amazon Elastic Computing Cloud. *Glossar:* Amazon Elastic Computing Cloud

H

- HTTP** Hypertext Transfer Protocol. *Glossar:* HTTP

I

- IDE** Integrated development environment. 5, 8, 10

J

- JSON** JavaScript Object Notation. *Glossar:* JSON

N

- Na** Not available. *Glossar:* Na (R)
- NaN** Not a Number. *Glossar:* NaN

O

O(OKR) Objective form the OKR logic. *Glossar*: Objective (OKR)

ODBC Open Database Connectivity (Connection). *Glossar*:

S

SDK Software Development Kit. 10, 14

SQL Structured Query Language. 7, *Glossar*: SQL

U

URL Uniform Resource Locator. *Glossar*: URL

V

ven Virtuel Environment first. 7, 9, 11, 14

vscode Visual Studio Code is and IDE for source-code editing and developing.. 11–13