Machine Learning - Pipelines

- Open-source Version Control System for ML (DVC)
- Curious Containers (CC)

cbmi Centrum für biomedizinische Bild- und Informationsverarbeitung
Forschung. Innovation. Inkubation

Jonas Annuscheit

htw Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

09.04.2019
CBMI-Journal-Club

Inhalt

- Vokabular
- Problemstellung
- Data Version Controll (DVC)
- Curious Containers (CC)
- DVC + CC
- Zusammenfassung

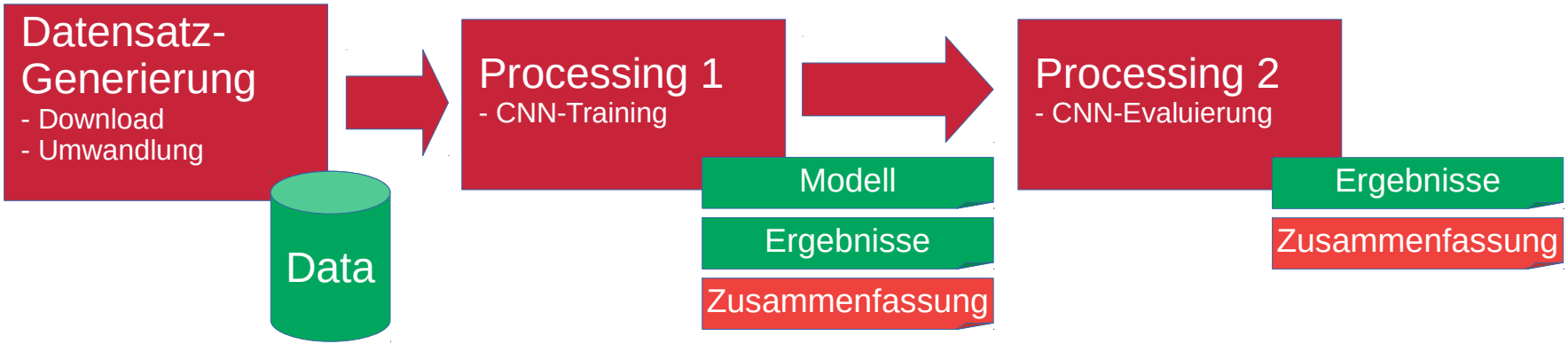
Inhalt

- Vokabular
- Problemstellung
- Data Version Controll (DVC)
- Curious Containers (CC)
- DVC + CC
- Zusammenfassung

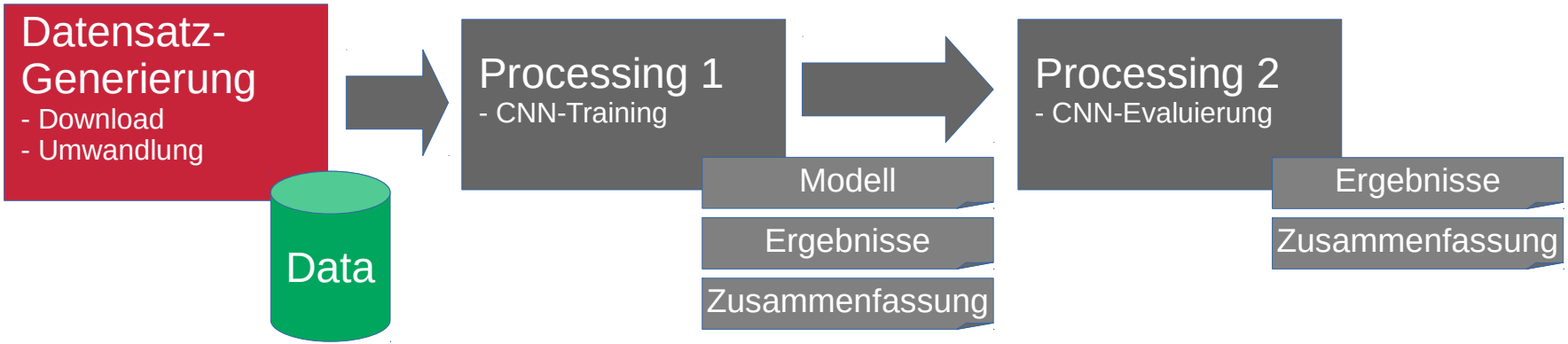
Vokabular

- Projekt: z.b.: Finden der besten Acc. für den CIFAR10-Datensatz.
- Experiment: An einer Stelle schrauben (z.B.Verschiedene Aktivierungsfunktion)
- Sub-Experiment: z.B.: Testen von 3, 5, 7 und 9 Schichten
- Ungültiges Experiment: Beinhaltet Programmierfehler

Problemstellung



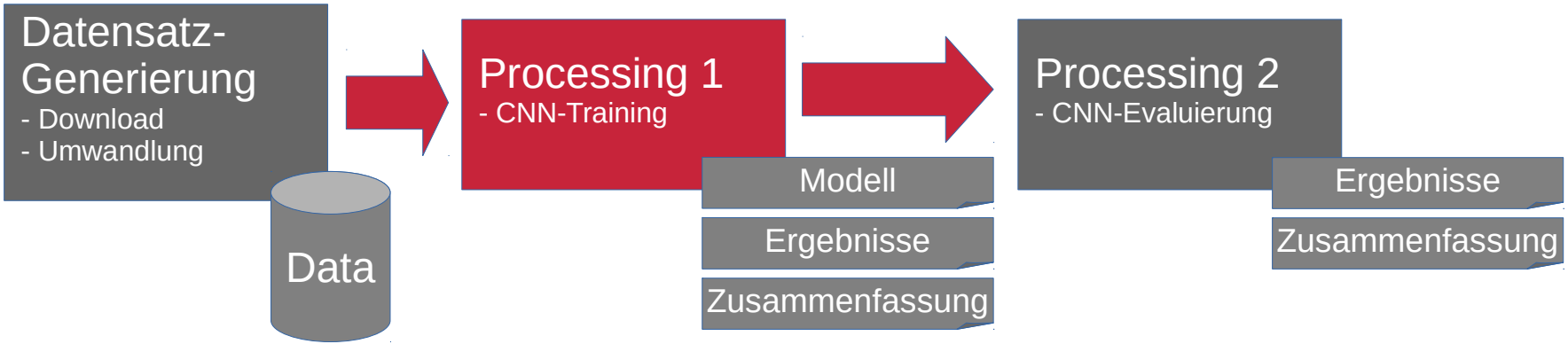
Problemstellung



Probleme:

- Verwerfung der Informationen,
  die Zum Datensatz geführt haben.

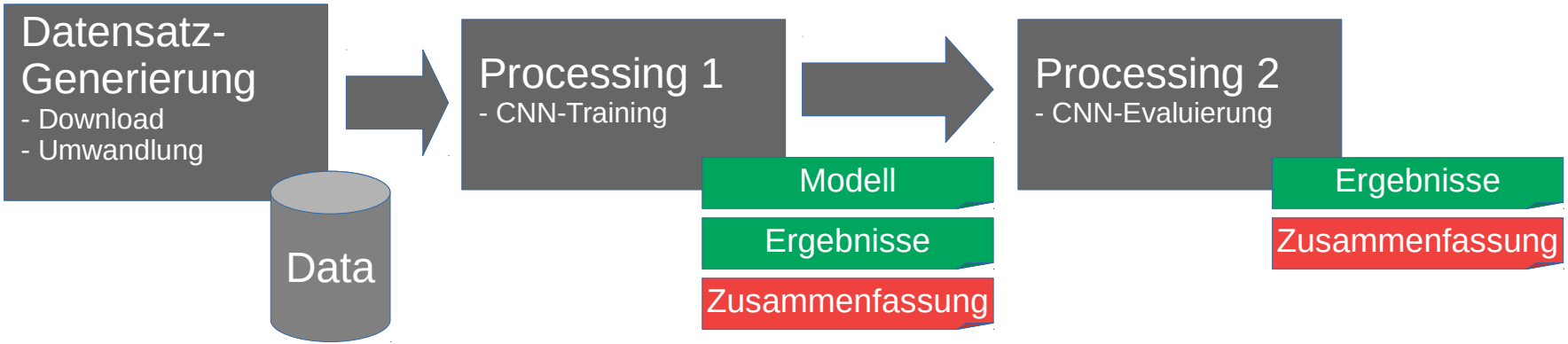
Problemstellung



Probleme:

- Verwerfung der Informationen, die Zum Datensatz geführt haben.
- Verwendete Parameter unbekannt
- Welche Datei zu welchen Ergebniss geführt hat, ist nicht nachvollziehbar.

Problemstellung



Probleme:

- Verwerfung der Informationen, die Zum Datensatz geführt haben.
- Verwendete Parameter unbekannt
- Welche Datei zu welchen Ergebniss geführt hat, ist nicht nachvollziehbar.
- Überschreibung der Dateien
- Nichts sagende Benennung

Problemstellung

- Reproduzierparkeit: Jeder kann die Ergebnisse Reproduzieren
- Klarheit: Leichte Verständlichkeit des Experiment-Aufbaues
- Skalierbarkeit: Mehrere Experimente parallel laufen lassen
- Einfachheit: Kein oder nur wenig extra Aufwand

Open-source Version Control System for ML (DVC)

Jonas Annuscheit

cbmi Centrum für biomedizinische Bild- und Informationsverarbeitung

Forschung. Innovation. Inkubation

09.04.2019
CBMI-Journal-Club

htw Hochschule für Technik und Wirtschaft Berlin

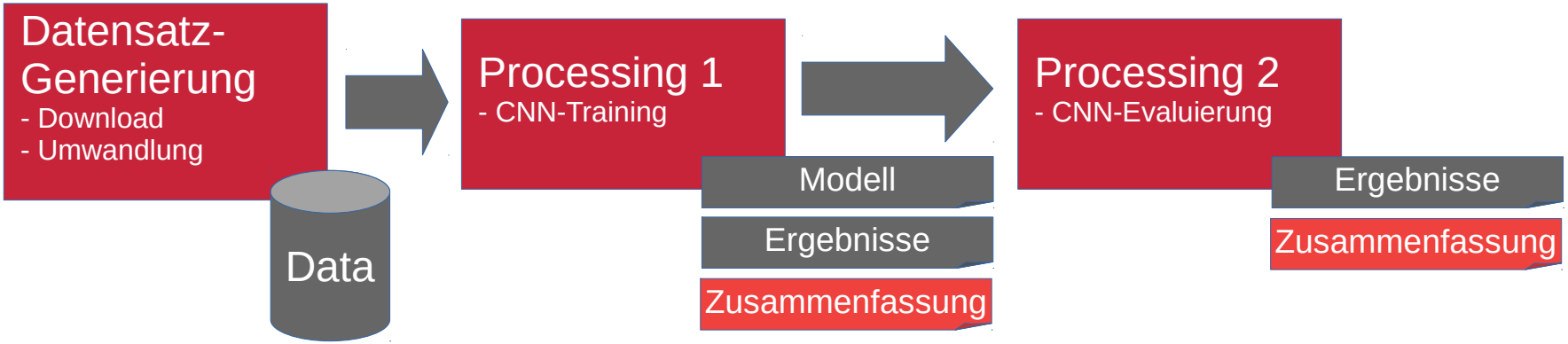
University of Applied Sciences

DVC

https://www.youtube.com/watch?v=4h6I9_xeYA4

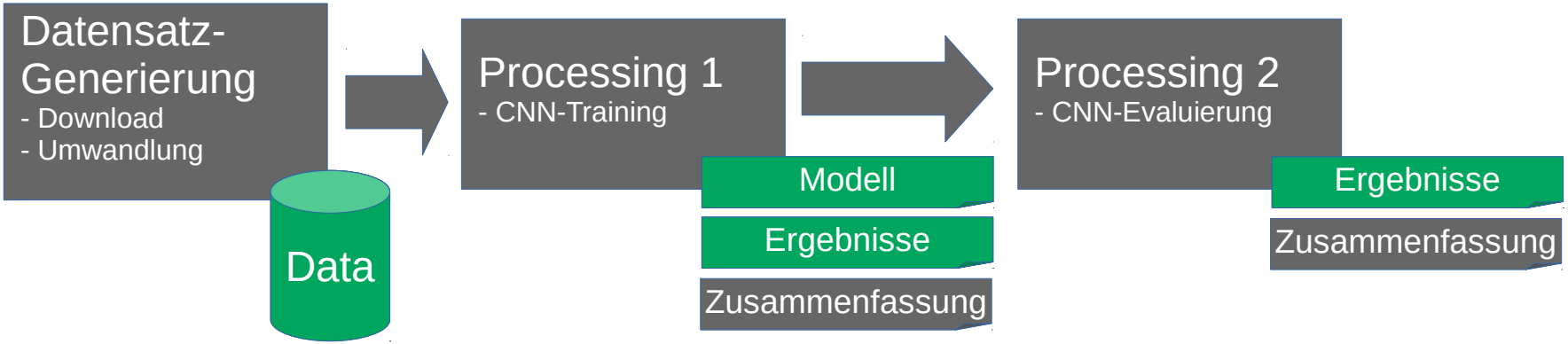
Erste Annahme

- Git ist ein gute Möglichkeit Source-Code zu verwalten



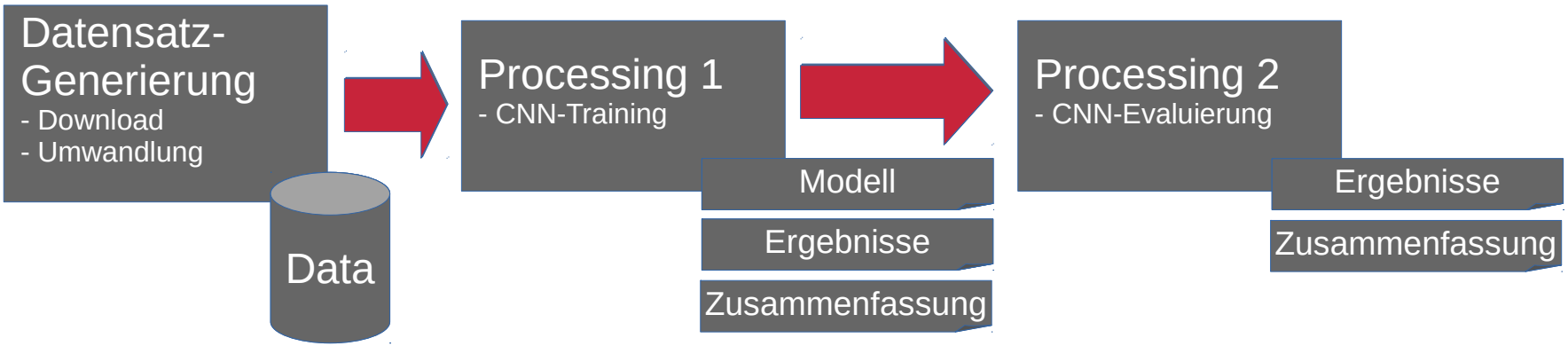
Zweite Annahme

- Große Dateien müssen extra gehändelt werden



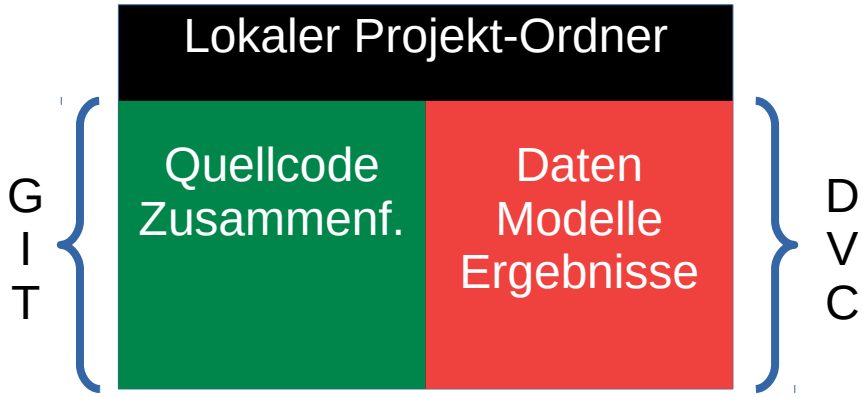
Dritte Annahme

- Die Abfolge der Abrufe muss gespeichert werden.

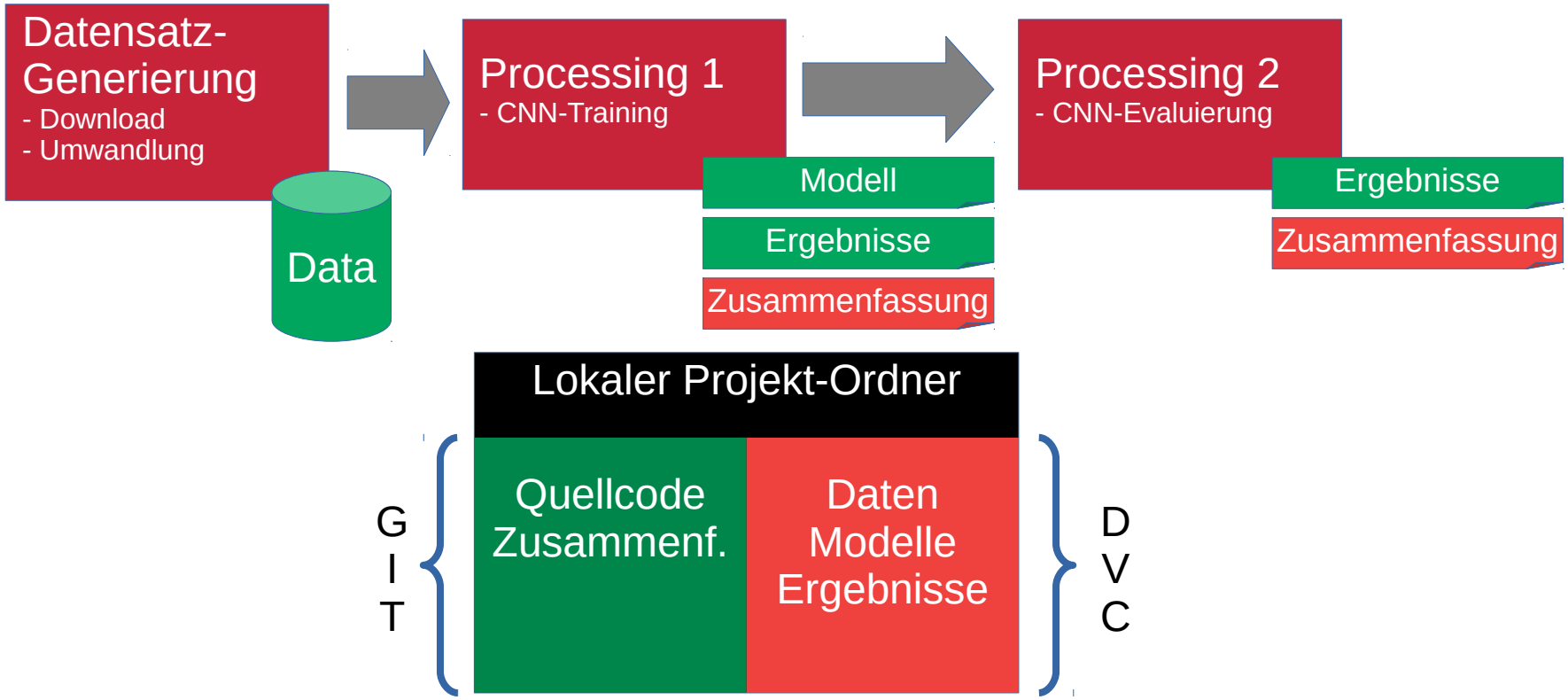


Zusammenfassung

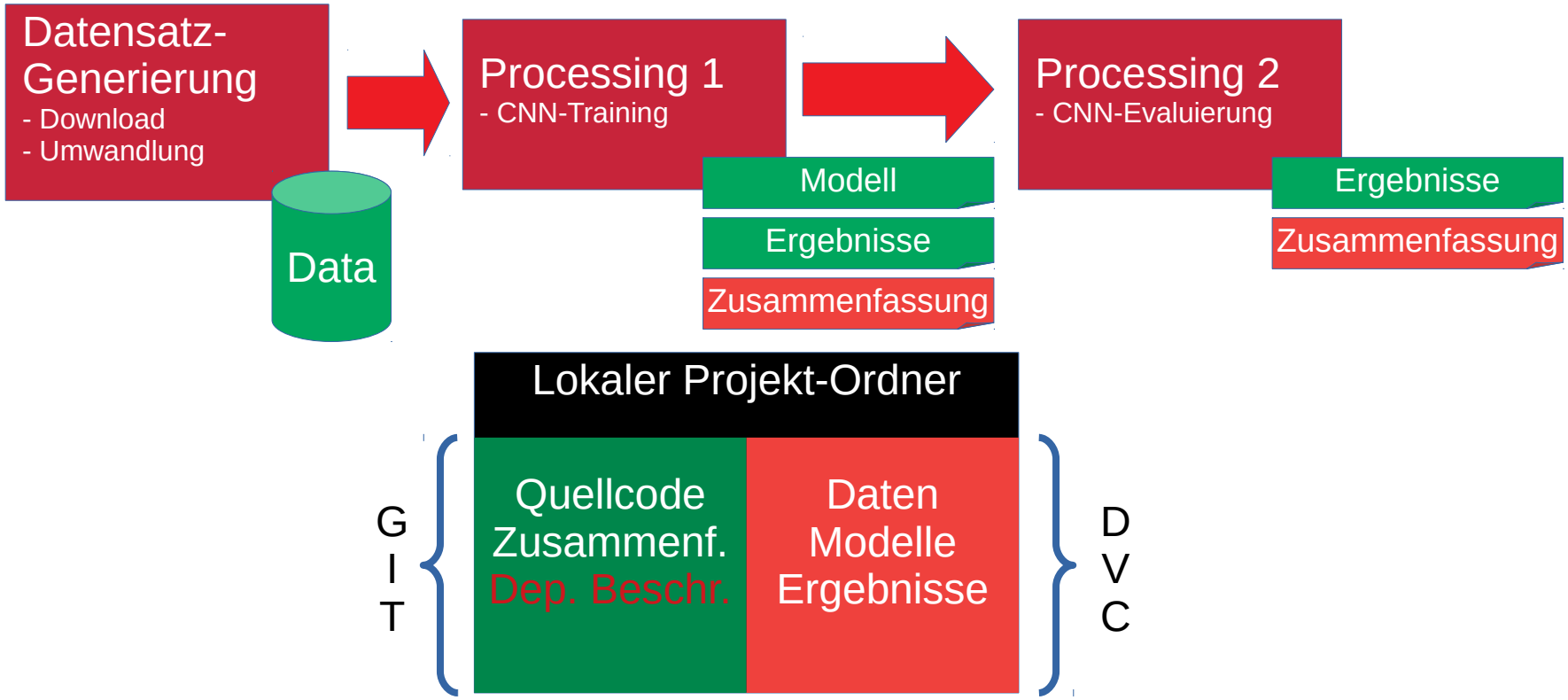
Zusammenfassung



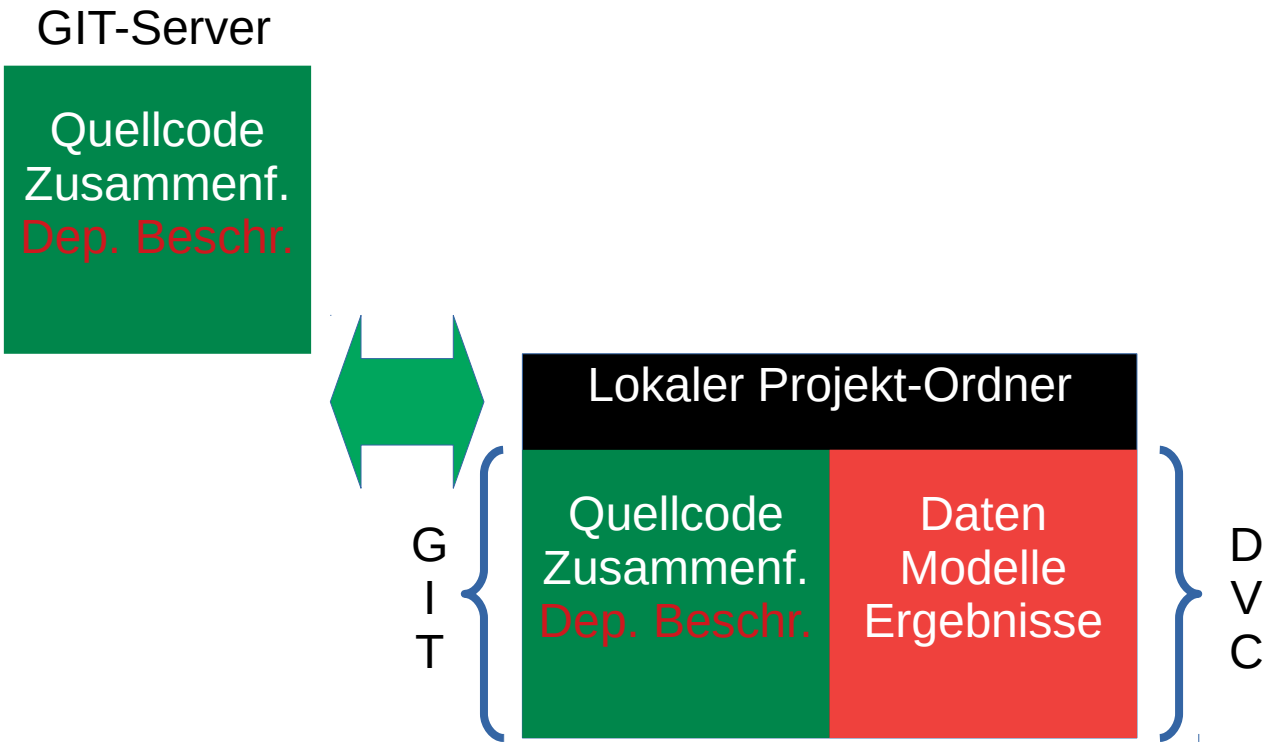
Zusammenfassung



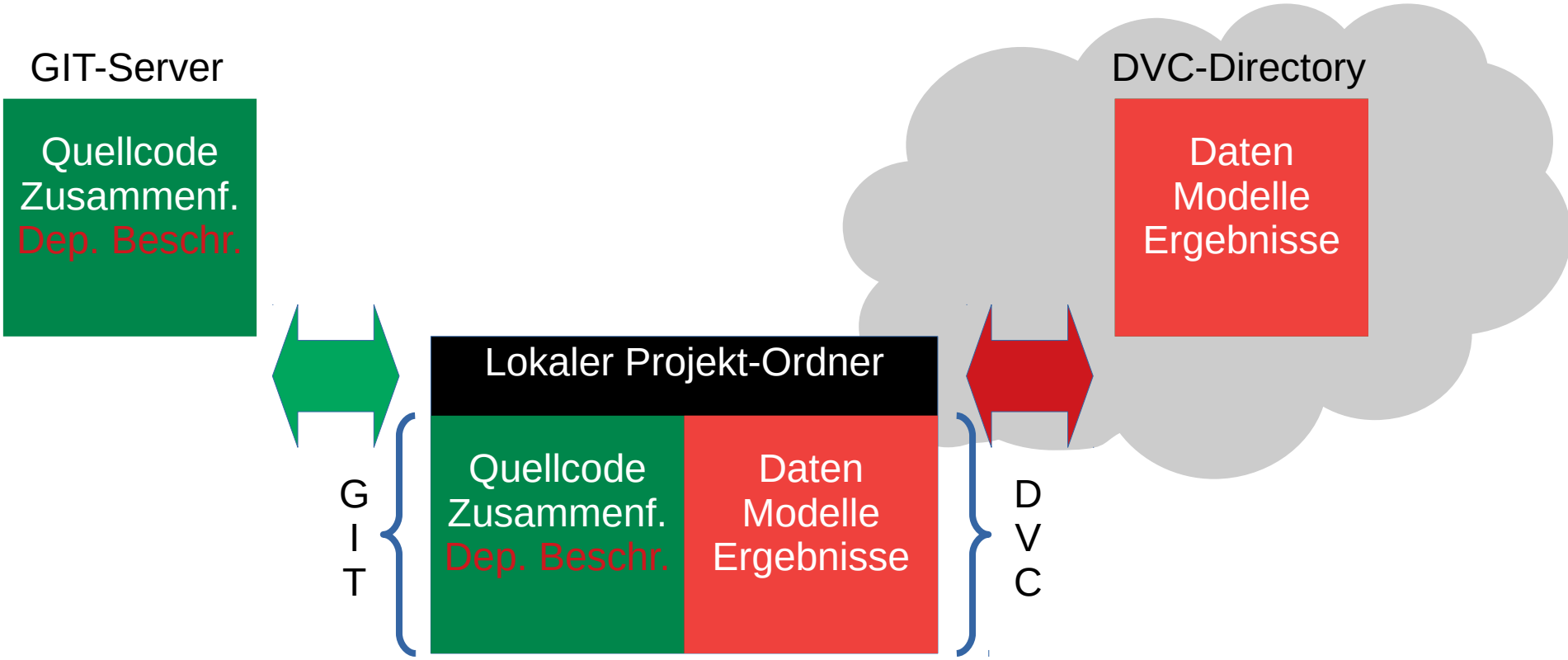
Zusammenfassung



Zusammenfassung



Zusammenfassung



DEMO: DVC

Komandos

```
# First you should install Anaconda and create an environment
conda create --name the_name_of_your_env pip git numpy appdirs decorator -y

# activate the environment
conda activate the_name_of_your_env   # in older versions: source activate the_name_of_your_env

# download source code
git clone https://github.com/mastaer/dvc-cc.git

# install the software that you need to run the code
pip install dvc-cc/dep/cc_core-7.0.0-py3-none-any.whl --ignore-installed   # soon this will be:  pip install cc-core
pip install dvc-cc/dep/cc_faice-7.0.0-py3-none-any.whl --ignore-installed  # soon this will be:  pip install cc-faice
pip install dvc-cc/dist/dvc_cc-0.1.0-py3-none-any.whl --ignore-installed

# GO to github or gitlab and create a new project: i.e. the project name: dvc_demo

# Clone the git: here is it the following link
git clone https://git.tools.f4.htw-berlin.de/annusch/dvc_demo.git

# Go the the project folder
cd dvc_demo
```

```
# CONVERT TO A DVC-Project,
# dvc init # This will be called by dvc-cc project create

# Create simple project
dvc-cc project create --mini_project

# instead of just calling "python create_some_data.py", we define with
# we use the --no-exec command, so that we can demonstrate later the dvc repro -P
dvc run -d create_some_data.py \
        -o mydata.npy \
        --no-exec \
        -f _create_data.dvc \
        python create_some_data.py

# we do this also for the train.py and for the eval.py
dvc run -d mydata.npy \
        -d train.py \
        -o train_model.npy \
        -m train_metric.json \
        --no-exec \
        -f _train.dvc \
        python train.py

dvc run -d mydata.npy \
        -d train_model.npy \
        -d eval.py \
        -m test_metric.json \
        -f eval.dvc \
        --no-exec
        python eval.py
```

```
# NOW we can run the experiment
dvc repro -P

# If we delete one file, for example the test_metric.json and call dvc repro -P again, it will only run the stage that is needed
rm test_metric.json
dvc repro -P


# show the dependencies
dvc pipeline show --ascii eval.dvc

# show a metric that
dvc metrics show -t json -x L1_Test -a



# if you are finish with testing, and this is nothing for you, you can remove the env with:
conda deactivate
conda remove -n the_name_of_your_env --all -y
```

Problemstellung (DVC)

- Reproduzierparkeit: (fast) jeder kann die Ergebnisse reproduzieren
- Klarheit: Leichte Verständlichkeit des Experiment-Aufbaues
- Skalierbarkeit: Mehrere Experimente parallel laufen lassen
- Einfachheit: Kein oder nur wenig extra Aufwand

Curious Containers (CC)

Jonas Annuscheit

cbmi Centrum für biomedizinische Bild- und Informationsverarbeitung
Forschung. Innovation. Inkubation

htw Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

09.04.2019
CBMI-Journal-Club

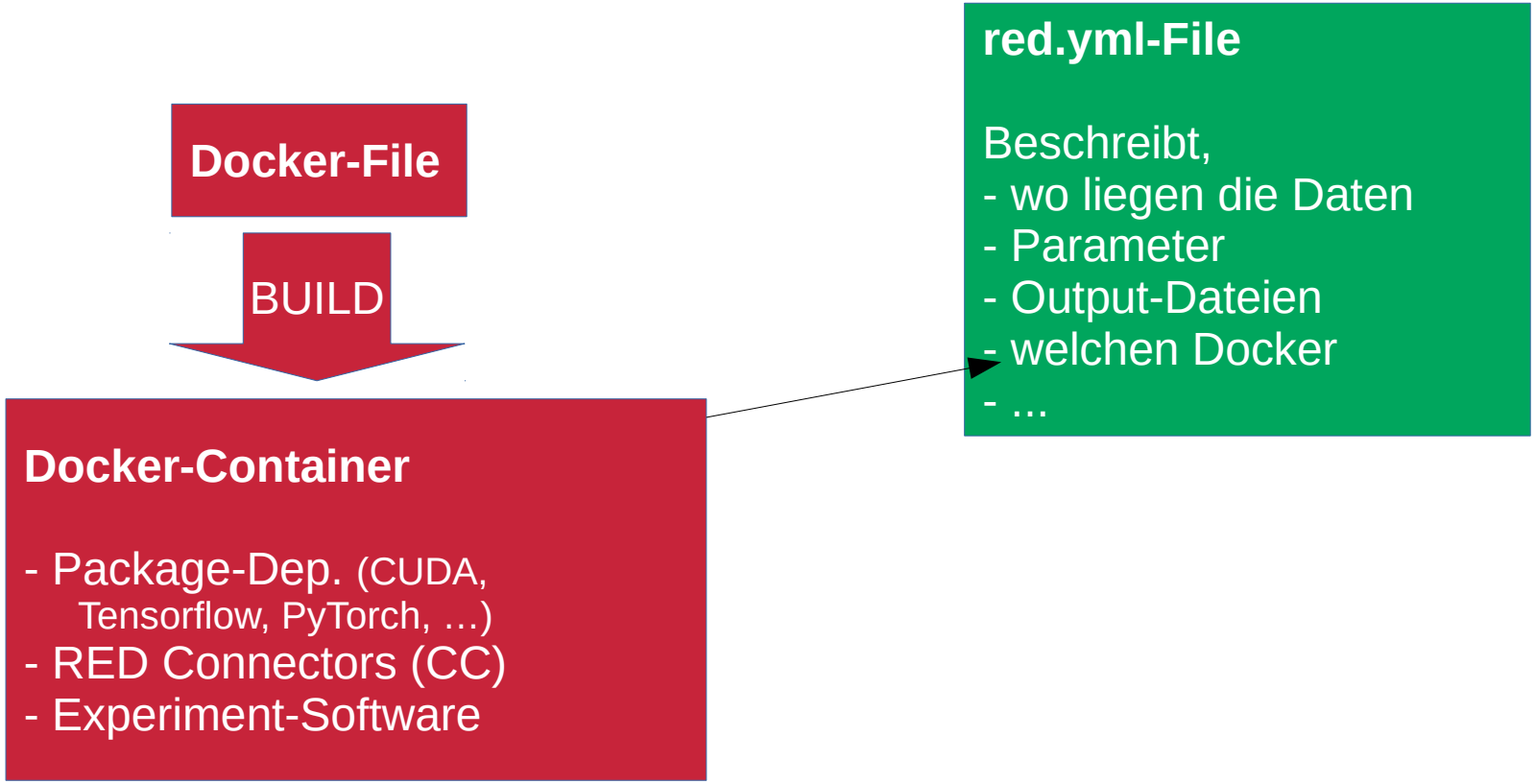
Informationen

- Autoren: Christoph Jansen & Bruno Schilling
- Ziele: Reproduzierparkeit & Skalierbarkeit
- Paper: Christoph Jansen at al. Reproducibility and Performance of Deep Learning Experiments for Cancer Detection in Pathological Images

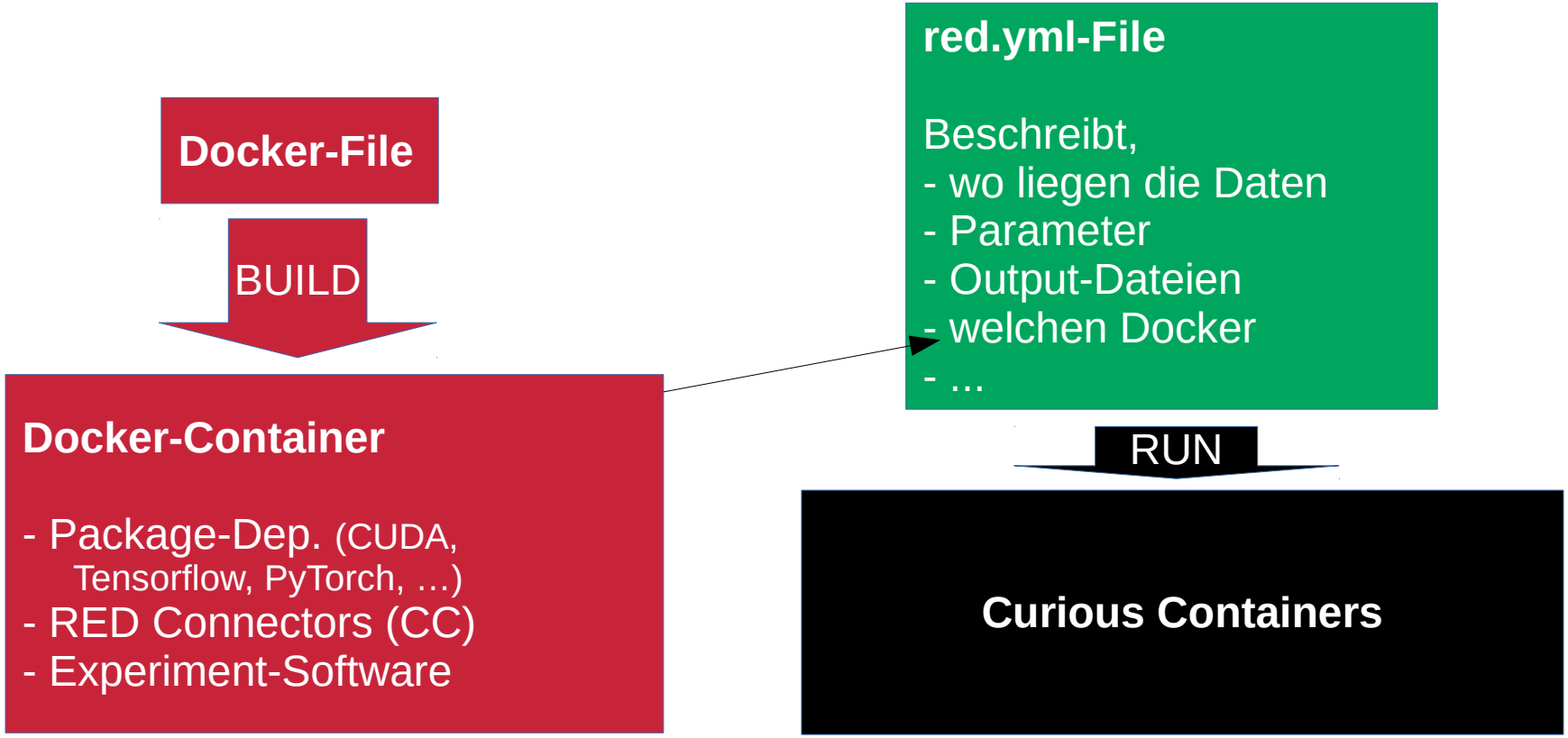
Informationen

- Autoren: Christoph Jansen & Bruno Schilling
- Ziele: Reproduzierparkeit & Skalierbarkeit
- Paper: Christoph Jansen at al. Reproducibility and Performance of Deep Learning Experiments for Cancer Detection in Pathological Images

Die Idee: Ein Experiment = Ein Docker-Container + red.yml-File



Die Idee: Ein Experiment = Ein Docker-Container + red.yml-File



Problemstellung (CC)

- Reproduzierparkeit: Jeder kann die Ergebnisse reproduzieren
- Klarheit: Leichte Verständlichkeit des Experiment-Aufbaues
- Skalierbarkeit: Mehrere Experimente parallel laufen lassen
- Einfachheit: Kein oder nur wenig extra Aufwand

DVC + CC

Jonas Annuscheit

cbmi Centrum für biomedizinische Bild- und Informationsverarbeitung

Forschung. Innovation. Inkubation

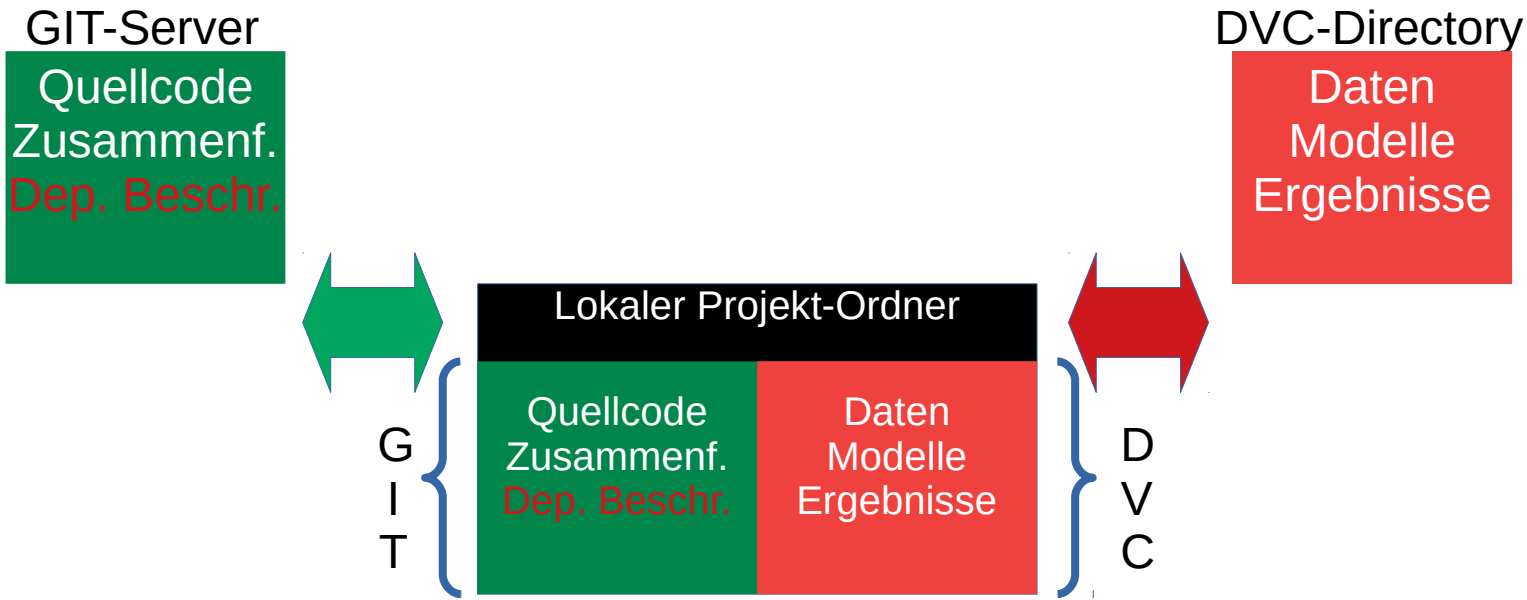
09.04.2019
CBMI-Journal-Club

htw Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Idee: Schritt 1

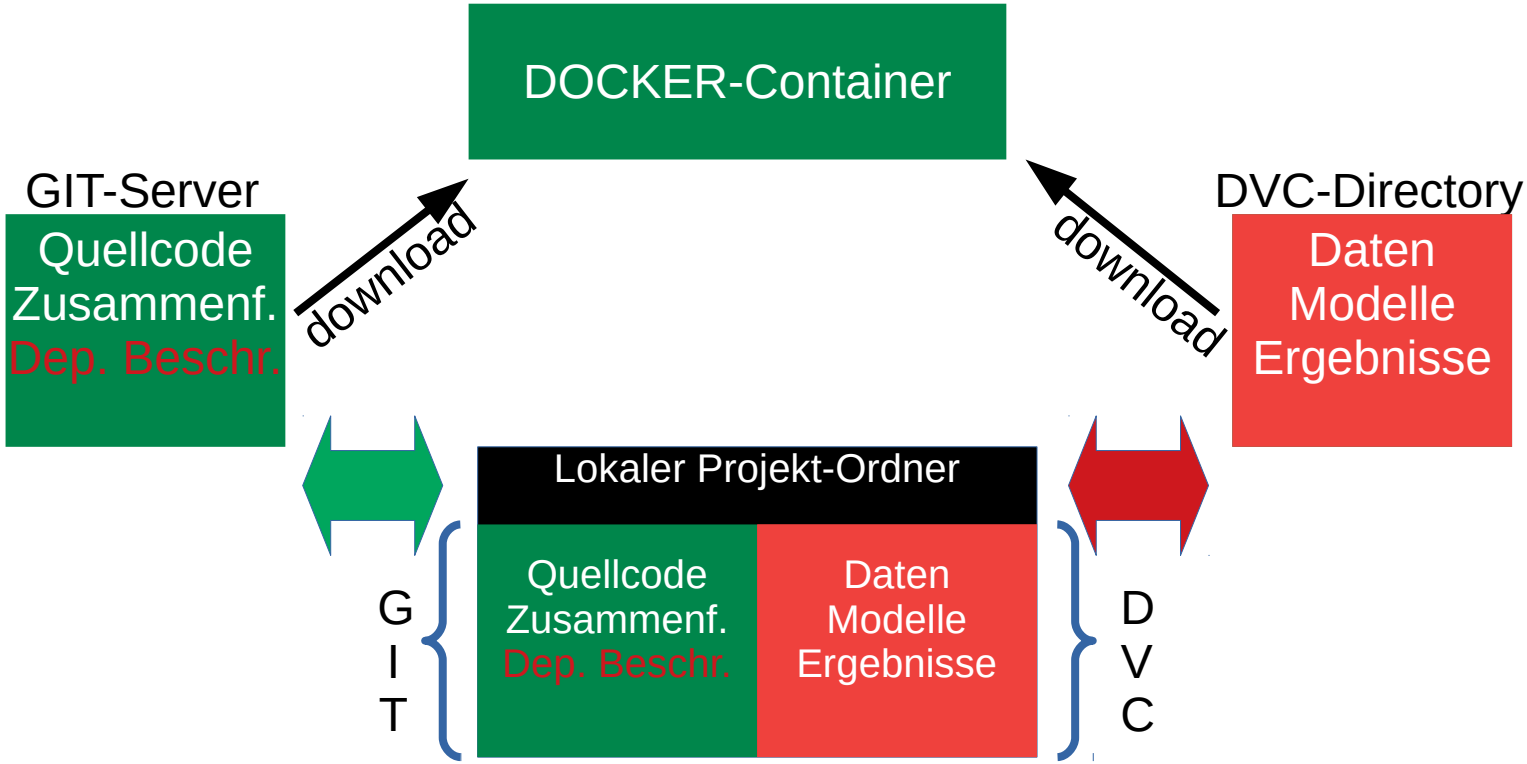
- Verwendung von DVC



Idee: Schritt 1.5

- Selber Docker-Container für alle Projekte
- Docker beinhaltet statt Source-Code vom Experiment, GIT und DVC-Repro.-Downloader

Idee: Schritt 1.5



Idee: Schritt 2

- red.yml-File aus den Informationen des GIT bzw. DVC automatisiert generieren

DEMO: DVC

Komandos

```
# First you should install Anaconda and create an environment
conda create --name the_name_of_your_env pip git numpy appdirs decorator -y

# activate the environment
conda activate the_name_of_your_env   # in older versions: source activate the_name_of_your_env

# download source code
git clone https://github.com/mastaer/dvc-cc.git

# install the software that you need to run the code
pip install dvc-cc/dep/cc_core-7.0.0-py3-none-any.whl --ignore-installed   # soon this will be:  pip install cc-core
pip install dvc-cc/dep/cc_faice-7.0.0-py3-none-any.whl --ignore-installed  # soon this will be:  pip install cc-faice
pip install dvc-cc/dist/dvc_cc-0.1.0-py3-none-any.whl --ignore-installed

# GO to github or gitlab and create a new project: i.e. the project name: dvc_demo

# Create a Folder on a server for the extern dvc directory

# Clone the git:
git clone https://git.tools.f4.htw-berlin.de/annusch/demo_dvc.git

# Go the the project folder
cd demo_dvc
```

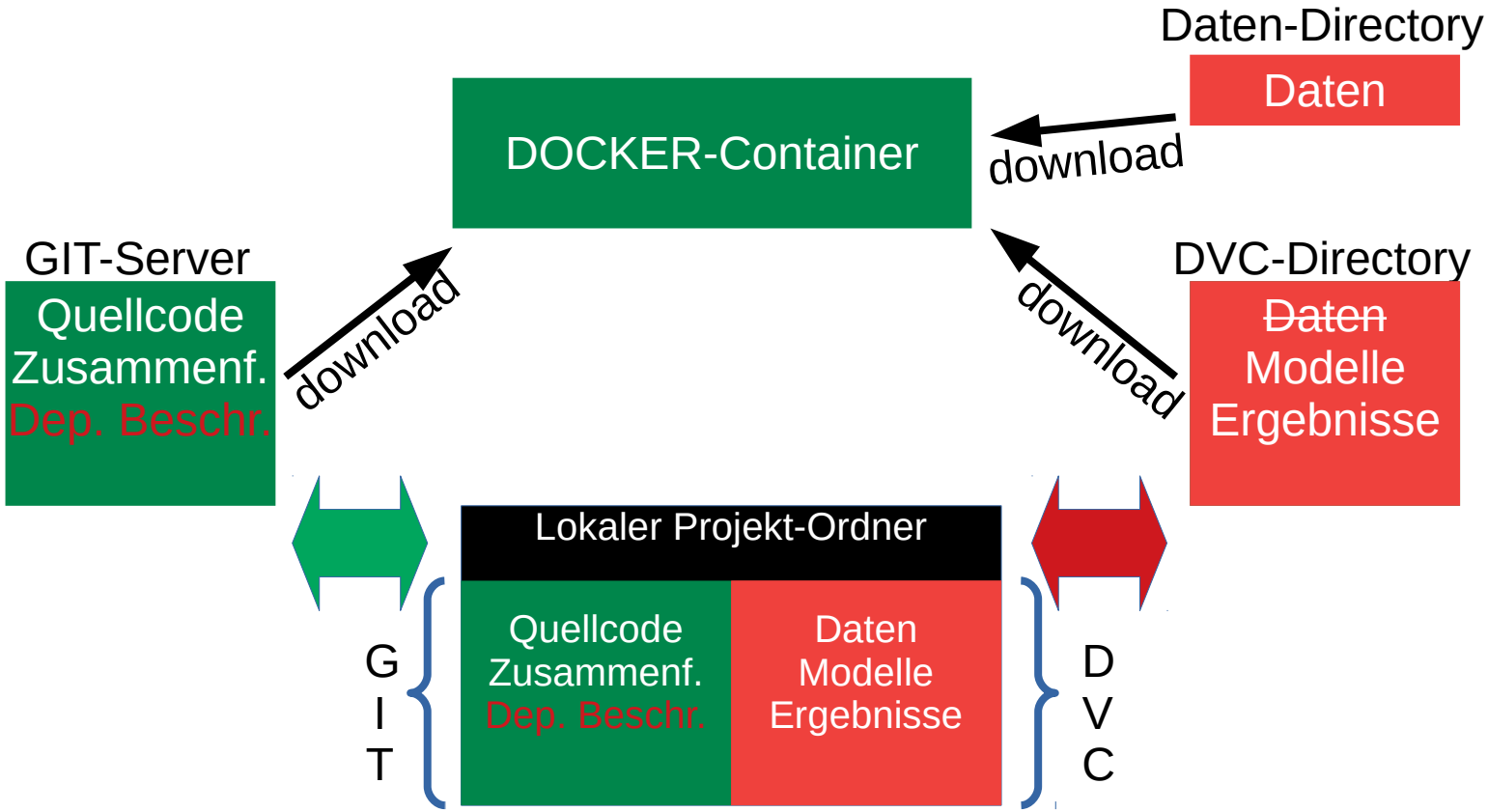
```
# CONVERT TO A DVC-Project,
# dvc init # This will be called by dvc-cc project create

# Create simple project
dvc-cc project create

# create a .dvc/config.local file: It should look like this (nano .dvc/config.local)
['remote "nas"']
url = ssh://annusch@avocado01.f4.htw-berlin.de/data/ldap/jonas/demo_dvc
ask_password = true

[core]
remote = nas

# create the dataset
dvc repro _generate_data.dvc

# push everything
dvc-cc git commit_and_push
```

```
46 # build a experiment in a branch
47 dvc-cc git branch first_experiment
48
49 # add the following line to nano code/argparser.py
50     parser.add_argument('-n','--num_of_samples', default=10)
51 # edit line 26 from code/train.py to
52 num_of_samples = int(args.num_of_samples)
53
54 # Create some sub-experiments:
55 dvc-cc project dummy -p "-n 2"
56 dvc-cc project dummy -p "-n 5"
57 dvc-cc project dummy -p "-n 10"
58 dvc-cc project dummy -p "-n 20"
59 dvc-cc project dummy -p "-n 100"
60
61 dvc-cc git commit_and_push
62
63 # create red yml file
64 dvc-cc red add_job -T
65
66 # run the job
67 dvc-cc jobs run
```

Problemstellung (DVC + CC)

- Reproduzierparkeit: (fast-fast-fast) jeder kann die Ergebnisse reproduzieren
- Klarheit: Leichte Verständlichkeit des Experiment-Aufbaues
- Skalierbarkeit: Mehrere Experimente parallel laufen lassen
- Einfachheit: Kein oder nur wenig extra Aufwand

What is next?


Daten-Directory
Daten
DOCKER-Container
download
GIT-Server
Quellcode
Zusammenf.
Dep. Beschr.
download
download
DVC-Directory
~~Daten~~
Modelle
Ergebnisse
Lokaler Projekt-Ordner
Quellcode
Zusammenf.
Dep. Beschr.
Daten
Modelle
Ergebnisse
G
I
T
D
V
C

Zusammenfassung (Vorschläge)

Jonas Annuscheit

cbmi Centrum für biomedizinische Bild- und Informationsverarbeitung
Forschung. Innovation. Inkubation

htw Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

09.04.2019
CBMI-Journal-Club

Vorschläge

- mehrere Experimenten => DVC
- Ein Experiment == Ein Branch
- Automatisierte Experiment-Namen-Vergabe

  (z.B. Branch-Name + Parameters) & Bennenung aller Dateien danach
- Verwendung von Clustern => Nutzung von CC mit DVC_CC

DANKE

Jonas Annuscheit

09.04.2019
CBMI-Journal-Club

cbmi Centrum für biomedizinische Bild- und Informationsverarbeitung
Forschung. Innovation. Inkubation

htw Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences