XIMA®

# **Synchronisation** als Entwicklungswerkzeug

TYPO3Camp Mitteldeutschland

01.04.2022

# Agenda

01 Überblick

02 Anforderungen

03 Tool(s)

04 Einsatz

05 Ausblick

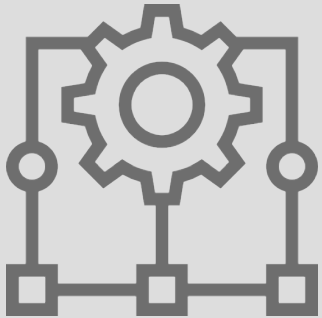
XIMA®

# Überblick

# Synchronisation

XIMA®

XIMA®



## Definition

- Synchronisation im Sinne der Informatik
- [...] das Abgleichen von Daten in einem verteilten System, [...]
- [...] Herstellung von Mehrexemplaren (Kopien) derselben Daten, meistens jedoch verbunden mit dem regelmäßigen Abgleich der Daten
- siehe auch Datenabgleich, Replikation

## Motivation – *Warum?*

„Hat jemand einen Dump vom Projekt X herumliegen?“

„Ich kann das Problem lokal nicht nachvollziehen.“

## Zielgruppe – *Wer?*

- **DevOps**
  - Backups, Abgleich von Remote Systemen
- **Entwickler*innen**
  - Abgleich lokaler Entwicklungsstand
- **Projektleitung // Kunde**
  - aktuelles Testsystem

XIMA®

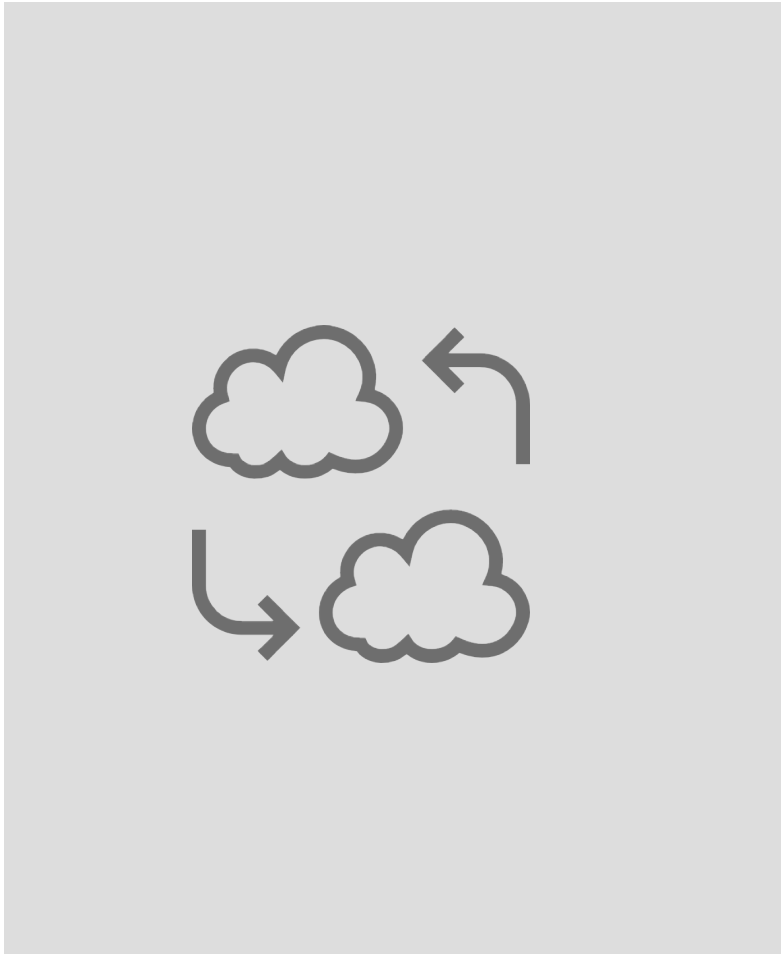
# Anforderungen

## Anforderungen – *Wie genau?*

- einfache Bedienung
- nachnutzbare Konfiguration
- flexibler Einsatz
- Möglichkeiten der Erweiterbarkeit
- Anwendungsungebunden

## Abgrenzung – *Was nicht?*

- kein Content Staging
- kein Clustering
- keine reine TYPO3 Extension



## Synchronisationsobjekte – *Was genau?*

- Datenbank
    - TYPO3 relevante Tabellen // Inhalte
- Dateien
    - fileadmin, uploads, etc.
    - Extension: File Fill

XIMA®

Tool(s)

## Evaluation – *Wie?*

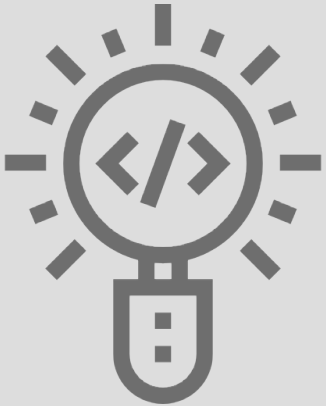
Welche Tools unterstützen die Synchronisation für TYPO3 bereits?

- **Content Staging**
  - in2publish Core - Extension
  - Netresearch Sync - Extension
  - content_sync - Extension
- **Deployment**
  - deployer-extended-typo3 - PHP Bibliothek
- **Backup**
  - TYPO3-backup - Bash script

***Welche Tools nutzt ihr?***

XIMA®

# db-sync-tool



## db-sync-tool

- https://github.com/jackd248/db-sync-tool
- Python Bibliothek zur Synchronisation von verteilten Systemen
- einfacher Einsatz als Command Line Tool
- angelehnte Funktionalität von drush sql:sync (Drupal)

**Features**

- Synchronisation von MySQL/MariaDB Datenbanken
- automatische Extraktion der Datenbankzugänge für unterstützte Systeme
- wiederverwendbare Konfiguration
- einfache Erstellung und Verwaltung von Backups
- ausführliches Logging
- zahlreiche Anpassungsmöglichkeiten

# db-sync-tool

**Inside the tool**

- Funktionsweise
- Einblick in automatische Extraktion
- Synchronisationsmodi
- Beispiel Setup

XIMA®

"Prod"



"Stage"

XIMA®

"Prod"

"Stage"

ORIGIN

TARGET

XIMA®

"Prod"

"Stage"

ORIGIN

TARGET

1 extract

php

XIMA®

"Prod"

"Stage"

ORIGIN

TARGET

1 extract

php

2 export

mysqldump

XIMA®

"Prod"

"Stage"

ORIGIN



TARGET

1 extract

`php`

2 export

`mysqldump`


3 compress

`tar`

XIMA®

"Prod"

"Stage"

ORIGIN



TARGET

1 extract

php

2 export

mysqldump


3 compress

tar


4 transfer

rsync // sftp

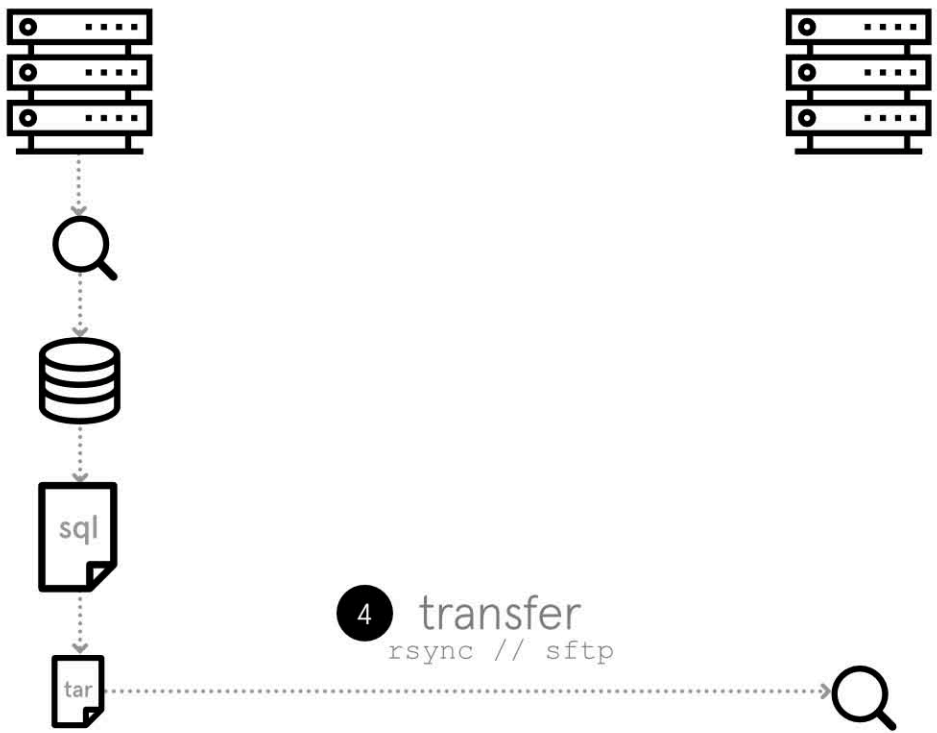
XIMA®

"Prod"

"Stage"

ORIGIN

TARGET




1 extract
php

2 export
mysqldump

3 compress
tar

4 transfer
rsync // sftp

5 extract
php

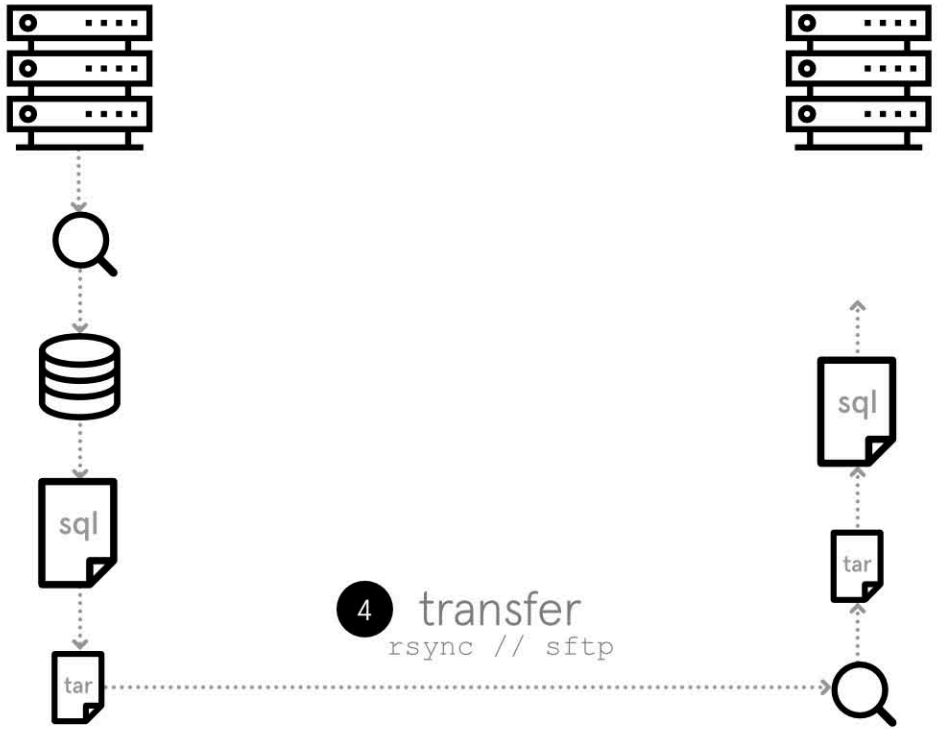
XIMA®

"Prod"

"Stage"

ORIGIN

TARGET




1 extract
php

2 export
mysqldump

3 compress
tar

4 transfer
rsync // sftp

uncompress 6
tar

extract 5
php

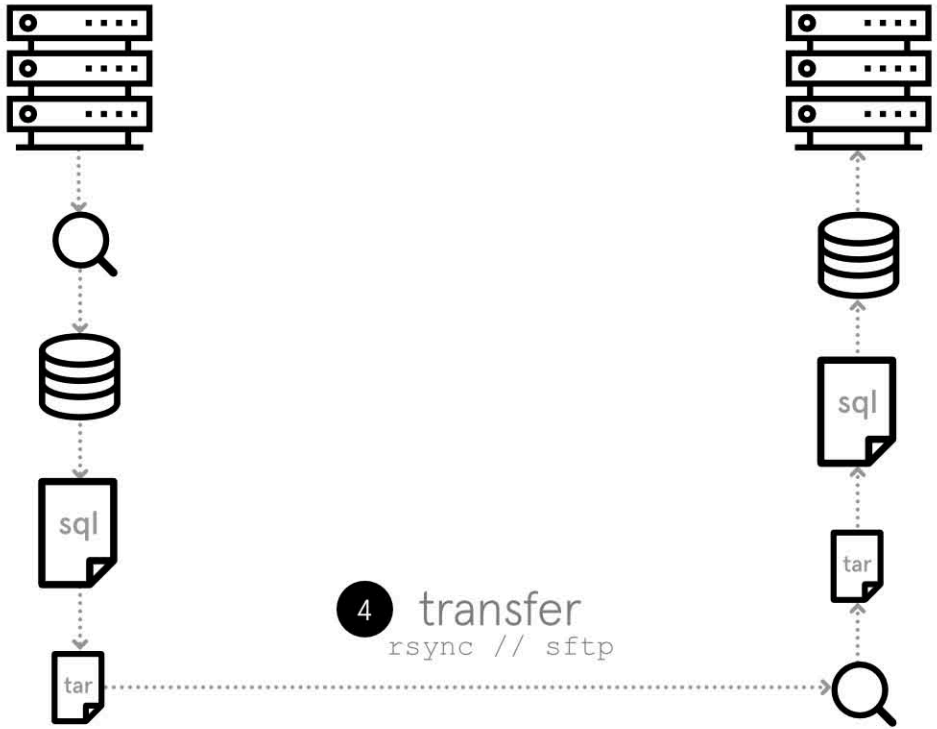
XIMA®

"Prod"

"Stage"

ORIGIN

TARGET




1 extract
php

2 export
mysqldump

3 compress
tar

4 transfer
rsync // sftp

7 import
mysql

6 uncompress
tar

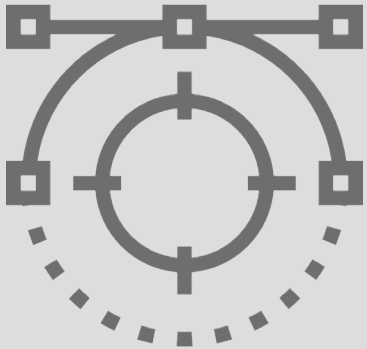
5 extract
php

## Funktionsweise

- Einsatz als einfacher **Command Runner**

  (am Beispiel einer einfachen Synchronisation)

  - (1) Zugangsdaten extrahieren (Origin)
  - (2) Datenbank Dump exportieren
  - (3) Datenbank Dump komprimieren
  - (4) Datenbank Dump transferieren
  - (5) Zugangsdaten extrahieren (Target)
  - (6) Datenbank Dump dekomprimieren
  - (7) Datenbank Dump importieren
  - (8) Aufräumen
- damit ist keine zusätzliche Software auf den Zielsystemen notwendig



## Extraktion

- Datenbankzugänge können anhand der in der Anwendung liegenden Konfigurationsdateien automatisch ausgelesen werden
  (zB mittels einfachen Konsolenbefehle: regex, php, sed, grep, etc.)
- aktuell unterstütze Anwendungen
  - TYPO3
  - Symfony
  - Drupal
  - Wordpress
  - Laravel
- Datenbank Konfiguration im db-sync-tool kann auch manuell vorgenommen werden

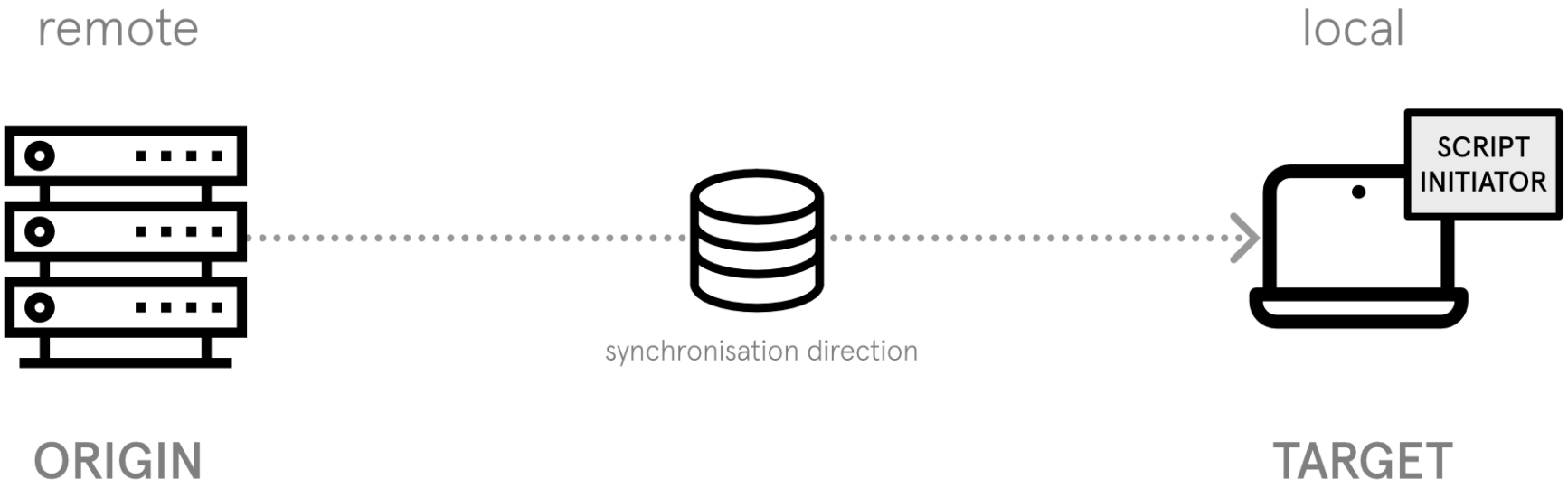
```php
<?php
return [
    'DB' => [
        'Connections' => [
            'Default' => [
                'charset' => 'utf8',
                'dbname' => 'db',
                'driver' => 'mysqli',
                'host' => 'db',
                'password' => 'db',
                'port' => 3306,
                'user' => 'db',
            ],
        ],
    ],
];
```

LocalConfiguration.php

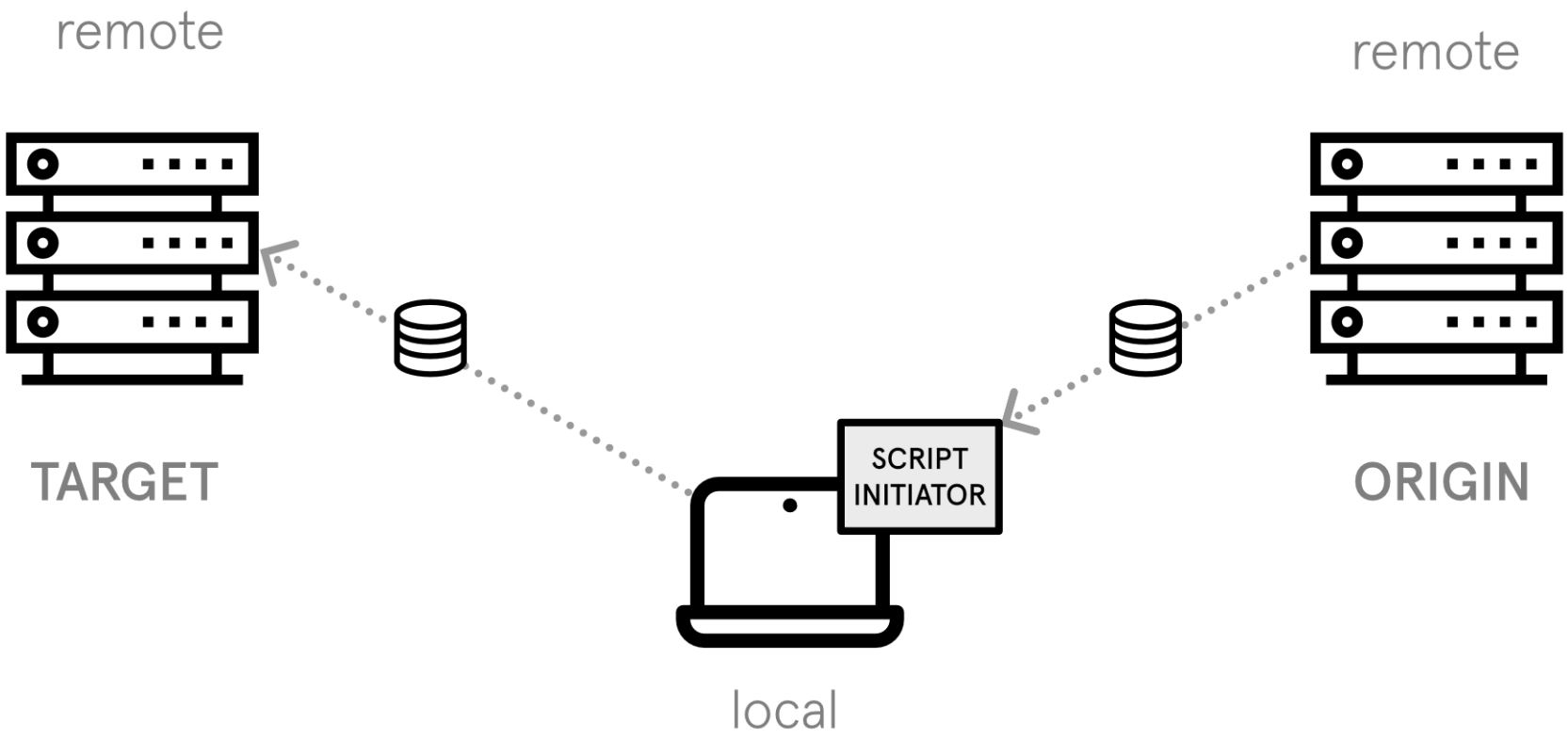
XIMA®

## Synchronisationsmodi

- verschiedene Modi für die Synchronisationsrichtung und das Zusammenspiel der verteilten Systeme
- Definition des Modus erfolgt implizit über Konfiguration von Synchronisationsquelle (ORIGIN) und -ziel (TARGET) sowie ausführendem Skript Initiator
- Folgende Modi:
    - Receiver
    - Sender
    - Proxy
    - Dump Local
    - Dump Remote
    - Import Local
    - Import Remote
    - Sync Local
    - Sync Remote




**Receiver** – Abgleich lokaler Entwicklungsstand




**Proxy** - Abgleich Remote Systeme


**Dump Remote** - Backup

## Setup

- Vorbedingung (Skript Initiator)
    - die Bibliothek benötigt Python 3.5 oder höher
- Installation via pip (alternativ auch packagist)

```
pip3 install db-sync-tool-kmi
```

- Einsatz für lokale Synchronisation in einem TYPO3 Projekt
    - Beispielkonfiguration `config.yml`

```yaml
type: TYPO3
origin:
    host: 192.87.33.123
    user: ssh_demo_user
    path: /var/www/html/shared/typo3conf/LocalConfiguration.php
    name: Demo Prod
target:
    path: /var/www/html/htdocs/typo3/web/typo3conf/LocalConfiguration.php
ignore_table:
    - be_users
    - sys_domain
    - cf_cache_*
```

config.yml

XIMA®

XIMA®

## Setup

- Vorbedingung
    - Die Bibliothek benötigt Python 3.5 oder höher
- Installation via pip (alternativ auch packagist)

```
pip3 install db-sync-tool-kmi
```

- Einsatz für lokale Synchronisation in einem TYPO3 Projekt
    - Beispielkonfiguration `config.yml`
- Ausführung der vorbereiteten Synchronisation

```
db_sync_tool -f config.yml
```

```
root@4c566339836b:/var/www/html#
```

XIMA®

XIMA®

# Einsatz

```
app/
  .ddev/
     commands/
       web/
          sync
   deployment/
      db-sync-tool/
         dump-prod.yml
         dump-stage.yml
         sync-prod-to-stage.yml
         sync-prod-to-local.yml
         sync-stage-to-local.yml
  src/
  .gitlab-ci.yml
```

Projektstruktur

XIMA®

## Einsatz

- Einsatz des **db-sync-tools** in unseren Projekten
- Entwickler*innen
    - Abgleich lokaler Entwicklungsstand
    - Umgebung: DDEV
        - Installation des db-sync-tools über erweitertes Dockerfile
    - einfache Nutzung eines DDEV Commands

```
ddev sync [prod|stage]
```

```
app/
  .ddev/
     commands/
       web/
          sync
   deployment/
      db-sync-tool/
         dump-prod.yml
         dump-stage.yml
         sync-prod-to-stage.yml
         sync-prod-to-local.yml
         sync-stage-to-local.yml
  src/
  .gitlab-ci.yml
```

Projektstruktur

XIMA®

```
#!/bin/bash

## Description: Executes db_sync_tool
## Usage: sync
## Example: "ddev sync prod" or "ddev sync stage"

db_sync_tool -f ./deployment/db-sync-tool/sync-{$1}-to-local.yml
```

.ddev/commands/web/sync

XIMA®

# Einsatz

- Einsatz des **db-sync-tools** in unseren Projekten
- DevOps
    - Umgebung: GitLab CI
        - Installation des db-sync-tools in GitLab CI Runner Dockerfile
    - Abgleich Remote Systeme
        - eigener Job **sync:stage**
        - bspw. mittels regelmäßigen Schedule
    - Backup
        - innerhalb des Jobs **deploy:prod**
        - Ausführung vor jedem Deployment

```
app/
  .ddev/
     commands/
       web/
          sync
   deployment/
      db-sync-tool/
         dump-prod.yml
         dump-stage.yml
         sync-prod-to-stage.yml
         sync-prod-to-local.yml
         sync-stage-to-local.yml
  src/
  .gitlab-ci.yml
```

Projektstruktur

XIMA®

XIMA®

```yaml
sync:stage:
  stage: sync
  only:
        - schedules
  script:
        ## Sync prod database to stage database
        - db_sync_tool -f deployment/db-sync-tool/sync-prod-to-stage.yml -y


deploy:prod:
    stage: deploy
    only:
        - master
    when: manual
    script:
        ## Create backup before deployment
        - db_sync_tool -f deployment/db-sync-tool/dump-prod.yml
        ## Run deployer deployment
        - php bin/dep deploy prod
```

.gitlab-ci.yml

XIMA®

# Ausblick

# Ausblick

**Weiterführender Einsatz**

- lokale Projektinitialisierung
    - einfaches Aufsetzen der kompletten lokalen Entwicklungsumgebung mit einem DDEV Command
- Remote Projektinitialisierung
    - Feature Branch Deployment
    - Automatische Erstellung von Projektinstanzen auf einem Staging System
- file-sync-tool
    - nachnutzbare Konfiguration für Dateisynchronisation

**Weiterführende Themen**

- Concurrency // Nebenläufigkeit von Daten
- Verschlüsselung bzgl. sensibler Daten

XIMA®

*Wie synct ihr?*

XIMA®

# Quellen

- Icon-Pack: Web design & development | Lineal (flaticon)
- https://unsplash.com/photos/1ScMQPmZOPw
- https://unsplash.com/photos/IsxaFsXi2rs
- https://unsplash.com/photos/HEEvYhNzpEo
- https://unsplash.com/photos/84R6ipsqaxo
- https://unsplash.com/photos/ze_aQiu3ZFU
- https://unsplash.com/photos/-IAS_N85adA
- https://unsplash.com/photos/5EUh-tq31eA
- https://unsplash.com/photos/9pOXS0ZGPDM