OSP

Otto Group Solution Provider

CI/CD & DevOps

Lernziele der Veranstaltung

Was solltest du mitnehmen?

OSP
Otto Group Solution Provider

- **Integration, Delivery, Deployment**
- **CI/CD Konzepte**
- **Sinnhaftigkeit von CI/CD**
- **Einsatzmöglichkeiten im Beleg**

Struktur der Veranstaltung

OSP
Otto Group Solution Provider

- Let‘s create a Startup
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Best Practices
- Fazit
- Noch Fragen?

OSP

Otto Group Solution Provider

Let‘s create a Startup

OSP|
Otto Group Solution Provider

Dresden Tech Designs

- Startup neben der Uni
- Maßgeschneiderte Softwareprojekte
- Von Organisation über Entwicklung bis Einführung und Betrieb
- Alles aus einer Hand

Unsere Situation

Das sind wir

OSP
Otto Group Solution Provider

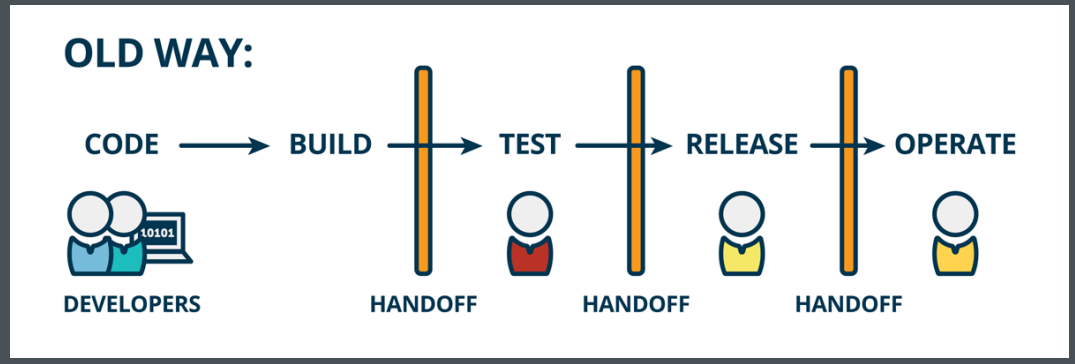
- Wir
  - sind ein großes Team
  - arbeiten kollaborativ
  - arbeiten an mehreren Funktionen gleichzeitig
  - haben mehrere Arbeitsstände, je nach Feature
  - arbeiten nach der „DevOps“ Kultur
  - wollen schnelles Feedback und kontinuierlich hohe Qualität für unsere Software

Entwickeln und Betreiben

„Gängiges“ Arbeiten

OSP
Otto Group Solution Provider

- Oft wird in einzelnen „Silos“ gearbeitet
- Entwicklungs- und Betriebsteams sind nur für ihren Bereich zuständig
- Wenig Prozesse werden automatisiert
- Bereitstellung erfolgt am Ende der Entwicklung
- Das führt möglicherweise zu
  - Langsamer Breitstellung der entwickelten Software
  - Kommunikationsproblemen
  - Ineffizienten Prozessen

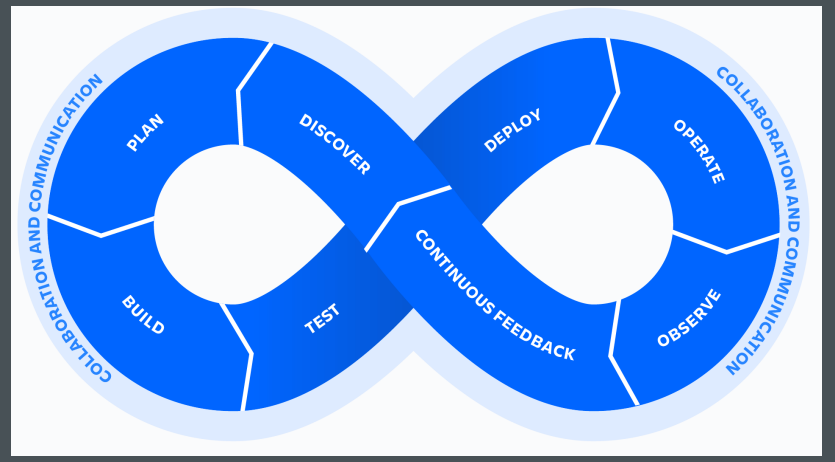


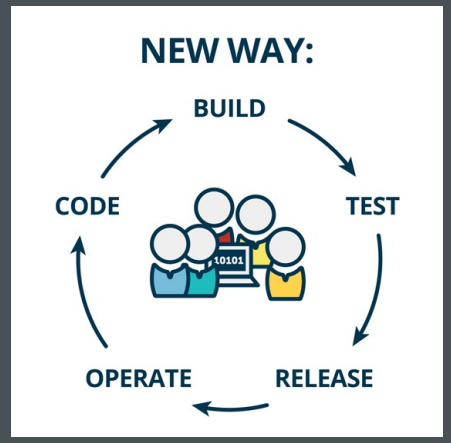
DevOps

Unser Prinzip

OSP
Otto Group Solution Provider

- Philosophie für Entwicklung und Betrieb von Software
- Entwicklung und Betrieb vereinen
- Verwendung unterschiedlicher Tools und Praktiken
- Dauerhafte Zusammenarbeit von Entwicklungs- und operativen Teams
- Wir wollen Prozesse der Entwicklung und Bereitstellung automatisieren / beschleunigen
- Etablieren von ständiger Kommunikation / Zusammenarbeit





Unsere Situation

Kundenauftrag!

OSP
Otto Group Solution Provider

- Der Kunde möchte eine API zum Verwalten von TO-Dos
- Als Programmiersprache soll Python gewählt werden
- TO-Dos sollen in einer Datenbank gespeichert werden
- TO-Dos sollen hinzugefügt, abgerufen, gelöscht und geupdatet werden
- -> CRUD-Operationen

Auf was müssen wir uns vorbereiten?

OSP
Otto Group Solution Provider

- Kollaborative Arbeit
- Schnelles Feedback zur Qualität unserer Arbeit
- Händische Arbeit minimieren
- Denken in kleinen Releases / Versionen
- Entwickeln und Testen (Integration)
- Bereitstellen (Delivery)
- Installieren / Ausliefern (Deployment)

OSP
Otto Group Solution Provider

- Was kann bei kollaborativer Arbeit an Software schief gehen?
- Welchen Herausforderungen können wir begegnen?
- Was macht uns die Arbeit leichter?

Herausforderungen

Kollaboratives Arbeiten

OSP
Otto Group Solution Provider

- Synchronisierung untereinander
- Austausch von Dateien
- Aktuellsten Stand abrufen
- Simultanes Arbeiten mit möglichst geringem Aufwand
- Wer hat wann, was geändert?
- Funktioniert mein Code noch nach Änderung?

OSP

Otto Group Solution Provider

Kick-Off!

GitHub Repository

Der Startpunkt

OSP
Otto Group Solution Provider

- Ausgangspunkt für uns ist ein Repository
- Code- und Dokumentenverwaltung
- Automatisierung
- Soll unsere Arbeit vereinfachen
- Beispiel Repository
- Wir nutzen GitHub Actions

OSP
Otto Group Solution Provider

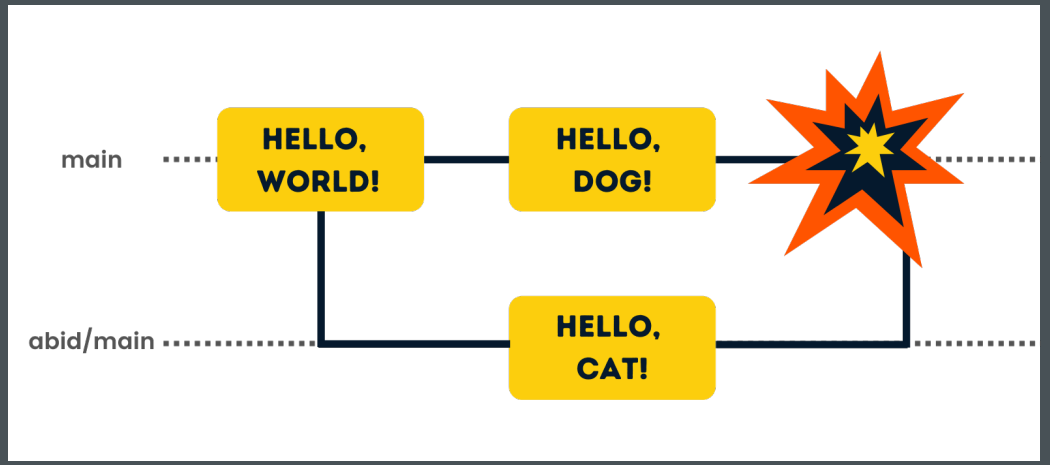
- Was kann bei kollaborativer Arbeit in einem Repository „schief“ gehen?
- Auf welche Probleme seid ihr bereits gestoßen?

GitHub Repository

Was kann passieren?

OSP
Otto Group Solution Provider

- Alle arbeiten gleichzeitig am Projekt
- Vielleicht ändern zwei Personen die gleiche Datei und pushen ihren Stand
- Oder es gab Änderungen auf einem Branch
- Dann gibt es zwei unterschiedliche Versionen des Codes
- -> Es kommt zu einem Konflikt (Merge Konflikt)
- -> Dateien im Repository passen also nicht immer “zusammen“



OSP

Otto Group Solution Provider

Continuous Integration

Continuous Integration

Unser Hilfsmittel

OSP
Otto Group Solution Provider

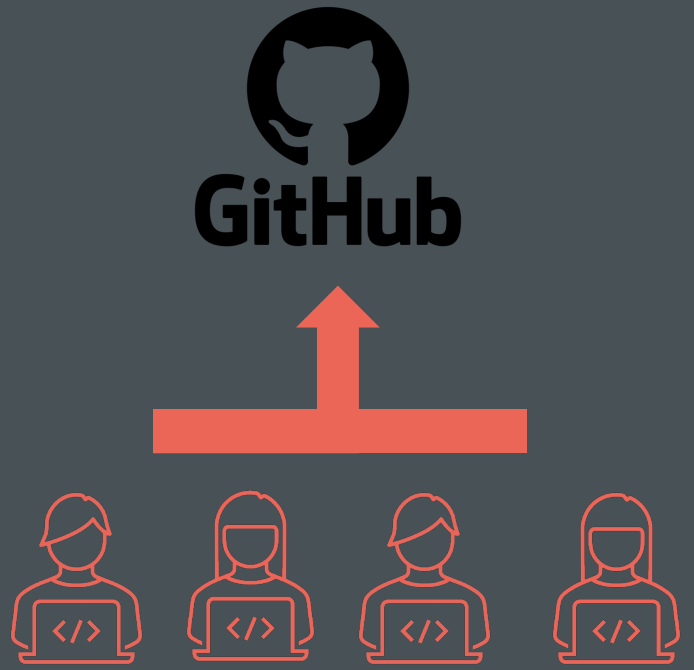
- Continuous Integration
- Zentraler Ort für Dokumente / Code -> Repository (GitHub, GitLab, BitBucket, ...), ein Versionsverwaltungssystem
- Jeder kann mitarbeiten
- Änderungen werden in zentralen Stand „integriert“
- Wir commiten unsere Änderungen regelmäßig
- Integration von Änderungen stets geprüft (-> Konflikte werden erkannt)
- Testen von Änderungen an unserem Code

Continuous Integration

Was führen wir ein

OSP
Otto Group Solution Provider

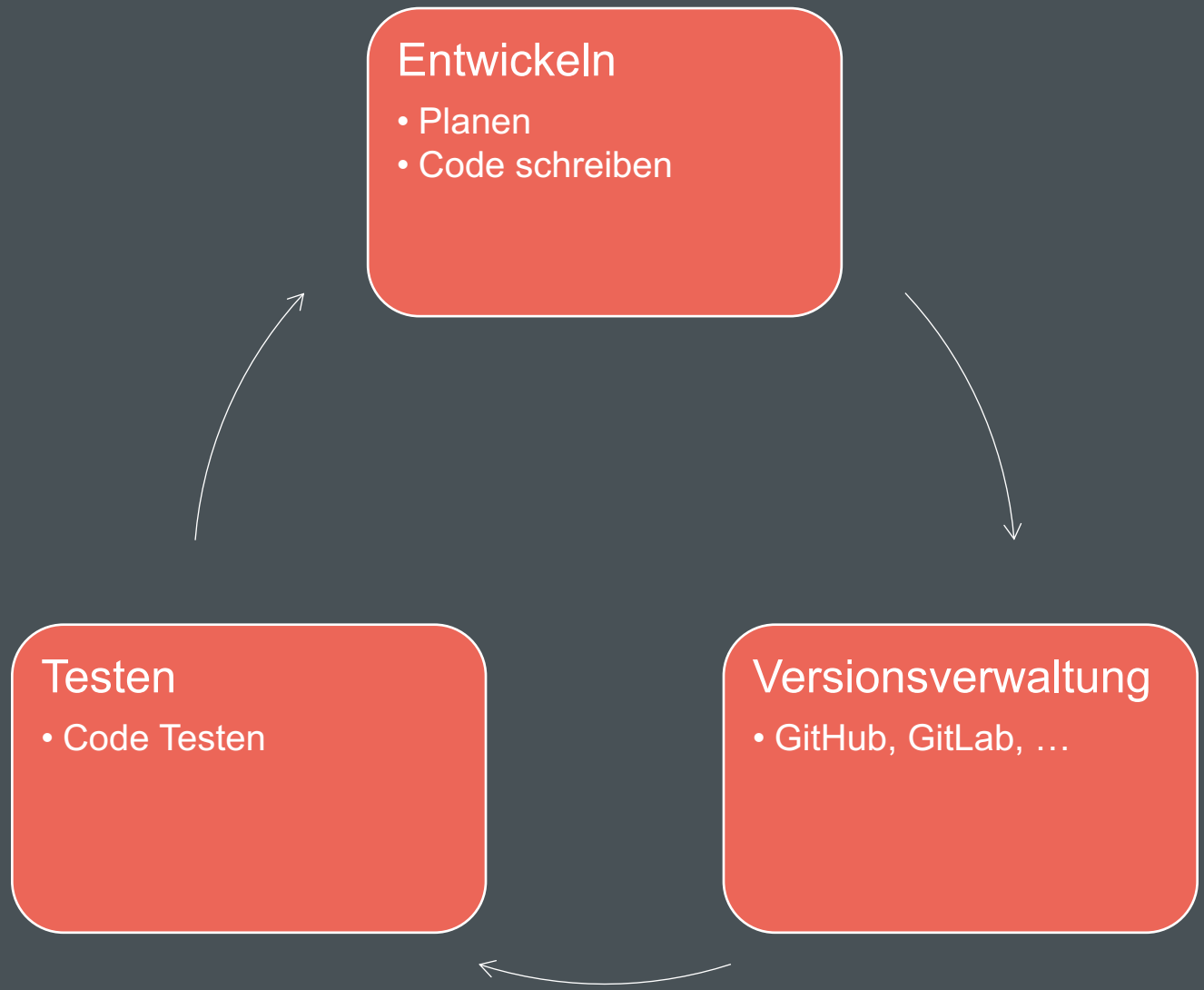
- Nutzen eines Repositorys auf GitHub (Unser Repository)
- Entwickeln von Tests für unsere Software (Tests)
- Aktualisierung unserer Dokumentation / Dokumente
- Häufige Synchronisierung unserer Änderungen



Continuous Integration

Zyklus

OSP
Otto Group Solution Provider



Continuous Integration

Zyklus - GitHub

OSP
Otto Group Solution Provider

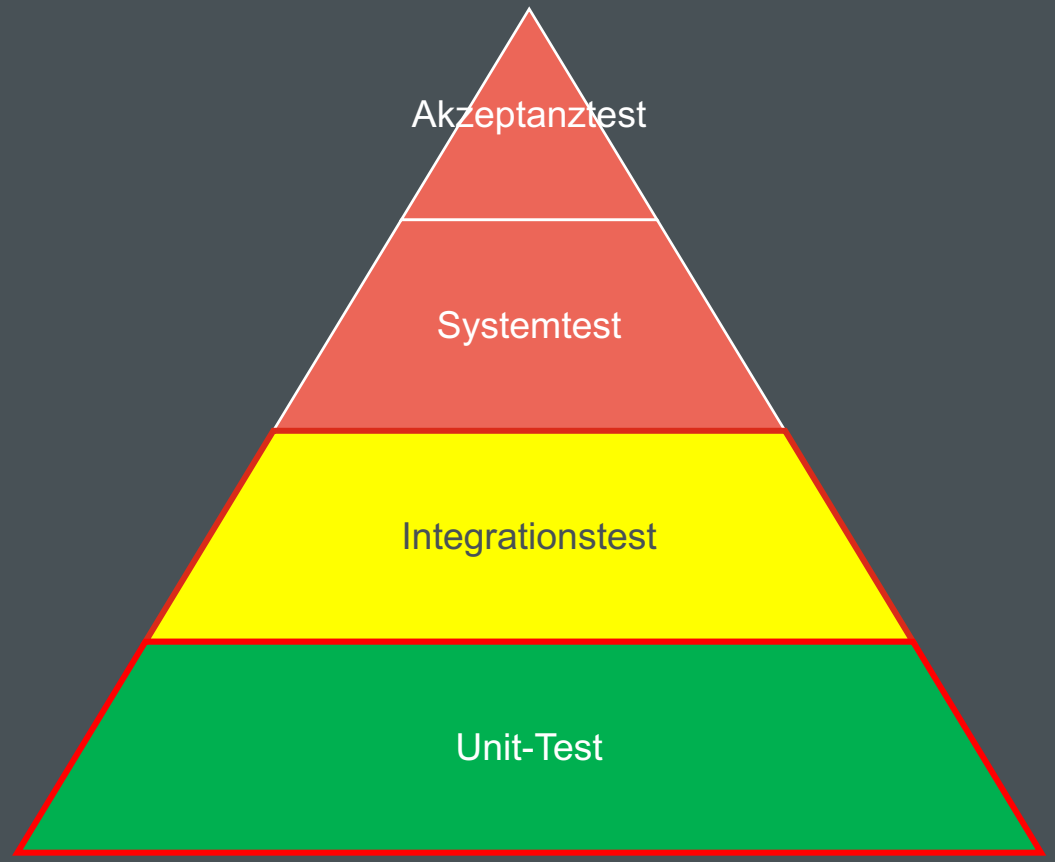
- Nutzen eines Repositorys auf GitHub (Unser Repository)
- Wir entwickeln fleißig Features
- Wir pushen unsere Änderungen ins Repository oder Mergen einen Branch
- GitHub prüft ob unser Code integriert werden kann -> Passt der Code zum bestehenden Code
- Dazu werden die Änderungen geprüft und durch Tests validiert -> geschieht automatisch

Continuous Integration

Zyklus - Tests

OSP
Otto Group Solution Provider

- Wir wollen unseren Code regelmäßig testen
- Unser Ziel ist automatisiertes Testen
- Dafür Nutzen wir
  1. Unit-Tests (Modul-Tests)
  2. Integrationstests



Continuous Integration

Zyklus – Tests – Unit-Test

OSP
Otto Group Solution Provider

- Prüfung einzelner Komponenten unseres Systems in Isolation
- Ziel: Fehler finden und Lauffähigkeit validieren
- Oft betrachten einzelner Klassen in einem Unit-Test

```python
class ToDoBase(BaseModel):
    title: str
    description: str


class ToDoCreate(ToDoBase):
    title: str
    description: str


class ToDo(ToDoBase):
    id: int
    title: str
    description: str
    is_done: bool

    class Config:
        from_attributes = True
```

Getestete Klasse

```python
def test_create_todo_with_valid_data():
    todo_data = {
        "id": 1,
        "title": "Test Todo",
        "description": "This is a test todo",
        "is_done": False
    }
    todo = ToDo(**todo_data)
    assert todo.id == 1
    assert todo.title == "Test Todo"
    assert todo.description == "This is a test todo"
    assert todo.is_done == False

```

Beispiel Unit-Test

Continuous Integration

Zyklus – Tests - Integrationstest

OSP
Otto Group Solution Provider

- Überprüfung des Zusammenspiels von Konfiguration mehrerer Module / Subsystemen
- Beispielsweise Datenbankkommunikation
- Zielstellung: Die einzelnen Teile meines Systems funktionieren miteinander
- -> Testen von Interaktionen mit anderen Modulen / (Sub)-Systemen

```python
def get_todo(db: Session, todo_id: int):
    """
    Retrieves a todo item from the database based on the specified todo_id.

    Parameters:
        db (Session): The database session.
        todo_id (int): The ID of the todo item to retrieve.

    Returns:
        ToDo | None: The todo item with the specified ID, or None if not found.
    """
    return db.query(models.ToDo).filter(models.ToDo.id == todo_id).first()
```

Getestete Klasse

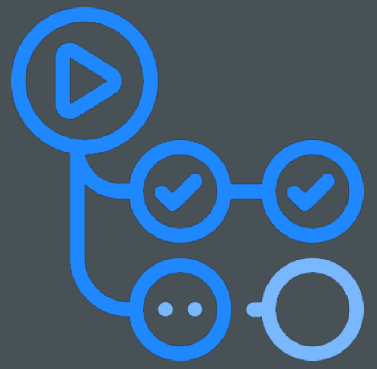
```python
def test_get_todo(client, db: Session):
    # Test getting a todo
    todo = models.ToDo(title="Test Todo", description="Test Description")
    db.add(todo)
    db.commit()

    response = client.get(f"/api/todo/{todo.id}")
    assert response.status_code == 200
    assert response.json()["title"] == "Test Todo"
    assert response.json()["description"] == "Test Description"
    assert response.json()["is_done"] is False
```

Beispiel Unit-Test

GitHub Actions

Teil der CI

OSP
Otto Group Solution Provider



- Wir Nutzen GitHub Actions zur Automatisierung von Aufgaben und Prozesse im Kontext von Softwareentwicklung
- GitHub Actions bietet uns die Plattform um Aufgaben automatisiert durchzuführen
- Prozesse / Aufgaben sind beispielsweise:
  - Generieren von Dokumentation
  - Prüfen unserer Änderungen am Code -> Tests
  - Anlegen / Ändern von Issues
  - Bauen unserer Anwendung

GitHub Actions

Funktionsweise

OSP
Otto Group Solution Provider

- Wir definieren Schritte, welche ausgeführt werden sollen
- Alle Schritte bilden einen Workflow
- Dieser wird durch Events ausgelöst
- Tritt ein Event ein werden alle oder ein Teil der Schritte ausgeführt, je nach Konfiguration
- Events sind beispielsweise:
  - Commits
  - Merge Requests
  - Issue Aktionen
- Schritte können dann erfolgreich / nicht erfolgreich sein -> direktes Feedback

Continuous Integration

Gelöste Probleme / Gewinn

OSP
Otto Group Solution Provider

- ✓ Zentraler Ort -> Keine Dateien hin und her schieben
- ✓ Neuester Stand immer sichtbar -> Keine Frage wer die neuesten Änderungen hat
- ✓ Synchronisierung der Stände immer möglich -> Jeder sieht sofort den Stand der anderen
- ✓ Gleichzeitiges Arbeiten möglich -> Jeder kann simultan Arbeiten, ohne andere zu hindern
- ✓ Änderungsverlauf (Commits/Pull Requests) -> Änderungsverlauf stehts präsent
- ✓ Funktionstest nach Änderung -> Fehler bei neuen Funktionen / Änderungen direkt erkannt

Unser Stand

GitHub Repository eingeführt, Tests

OSP
Otto Group Solution Provider

Das können wir

- ✓ Kollaborativ erfolgreich Arbeiten
- ✓ Softwareqaulität durch Tests halten
- ✓ Fehler direkt erkennen
- ✓ Dokumentation generieren

Wir wollen zusätzlich

- × Software bauen
- × Software ausliefern

OSP|
Otto Group Solution Provider

Fragen?

Continuous Delivery

OSP
Otto Group Solution Provider

- Wie kommen wir von Code zu etwas ausführbaren?
- Wie bringe ich den Code „zum laufen“?

Build Step

Wat is dat?

OSP
Otto Group Solution Provider

- Verpacken / transformieren unseres Codes in eine kleine ausführbare Einheit
- Zum Beispiel:
  - Kompilieren des Codes in Maschinencode (Java)
  - Ausführen von Transpilern zur Transformation von einer Sprache in eine andere (Typescript zu Javascript)
  - Installieren von Abhängigkeiten und Einrichten der Laufzeitumgebung (Python)
  - Verpacken des Codes in Docker Images (quasi alle Sprachen)

```
# Bauen einer NPM-Installation
npm run build
```

```
# Bauen einer Python Anwendung
python -m pip install --upgrade pip
pip install -r requirements.txt
python setup.py build
```

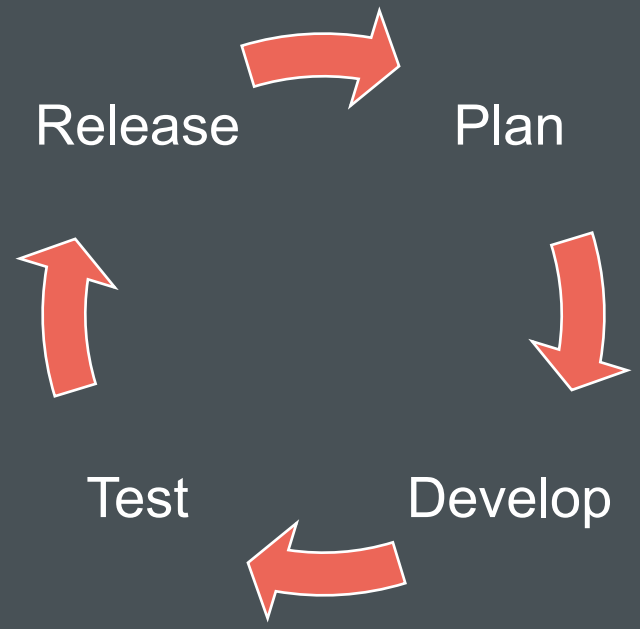
```
# Bauen einer Java Anwendung mit Maven
mvn clean install
```

Denken in Releases

Unser Ziel

OSP
Otto Group Solution Provider

- Möglichst zügig wissen, ob wir in die richtige Richtung arbeiten
- -> Schnell Feedback einholen
- Technische Schulden verringern
- Qualität je Release dauerhaft halten, am besten sogar steigern
- Schnell Funktionen „an den Mann bringen“
- Unsere Software möglichst einfach und stressfrei ausliefern

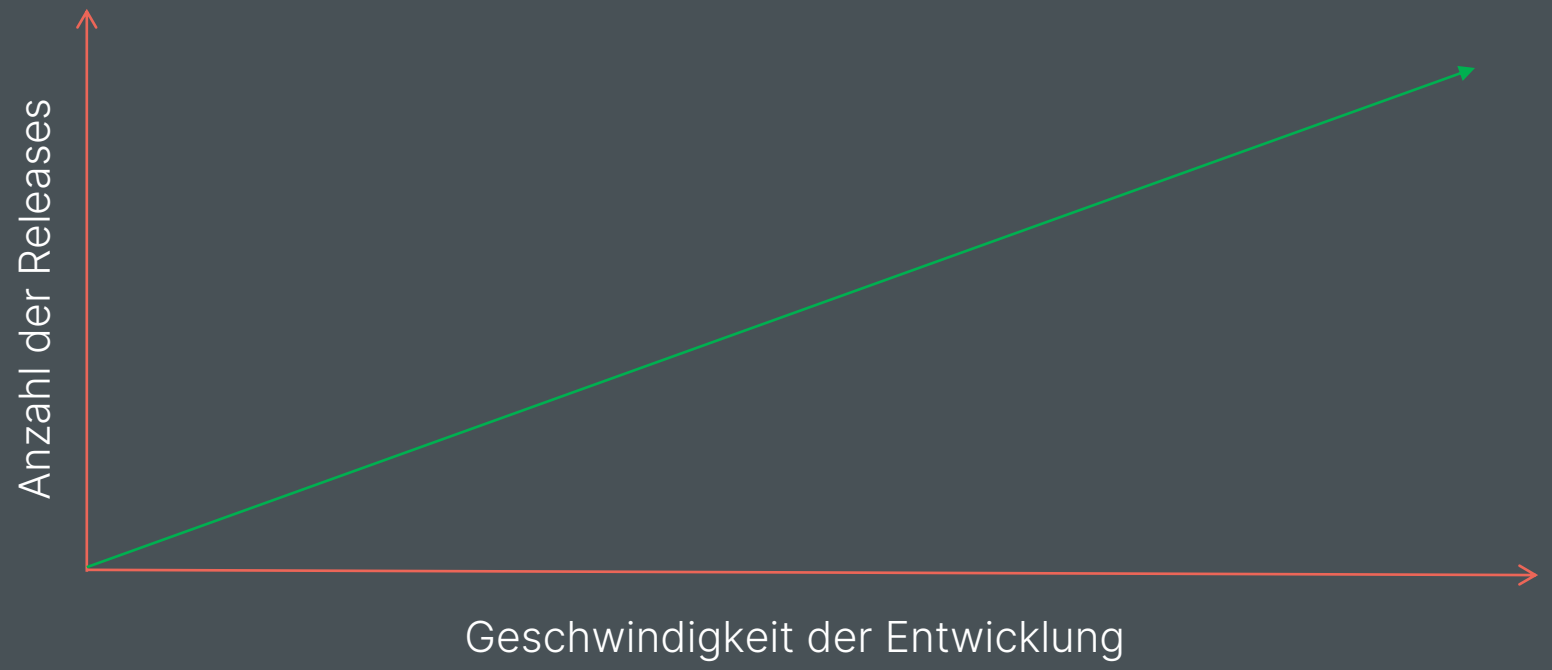


Denken in Releases

Klein aber oho!

OSP
Otto Group Solution Provider

- Wir wollen oft neue Funktionen / Verbesserungen veröffentlichen
- Wir denken in „kleinen“ Releases -> Release = neue Version der Software

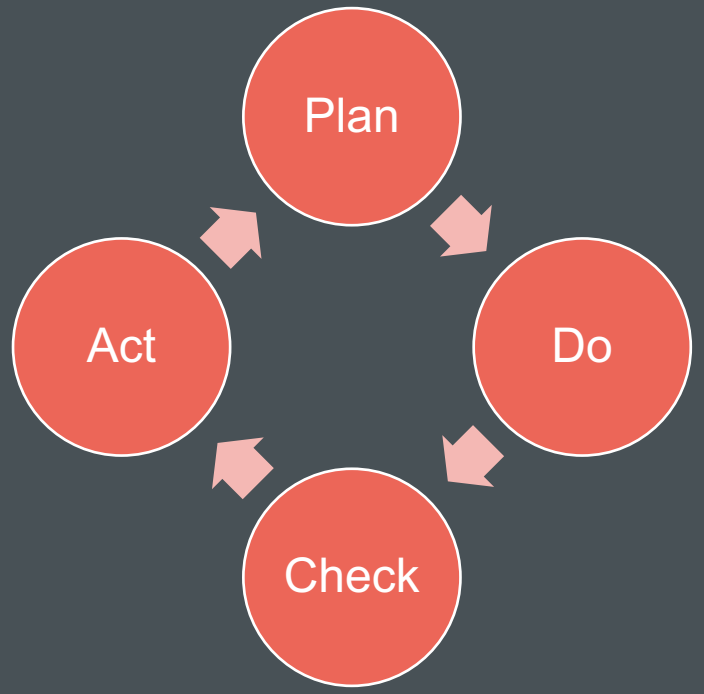


Denken in Releases

Schnelles Feedback

OSP
Otto Group Solution Provider

- Schnelles Feedback durch schnelle kleine Releases
- Der Kunde gibt uns direkt Meldung, ob wir gute Arbeit leisten
- Wir bekommen Feedback aus erster Hand
- Dadurch reduzieren wir unsere technischen Schulden

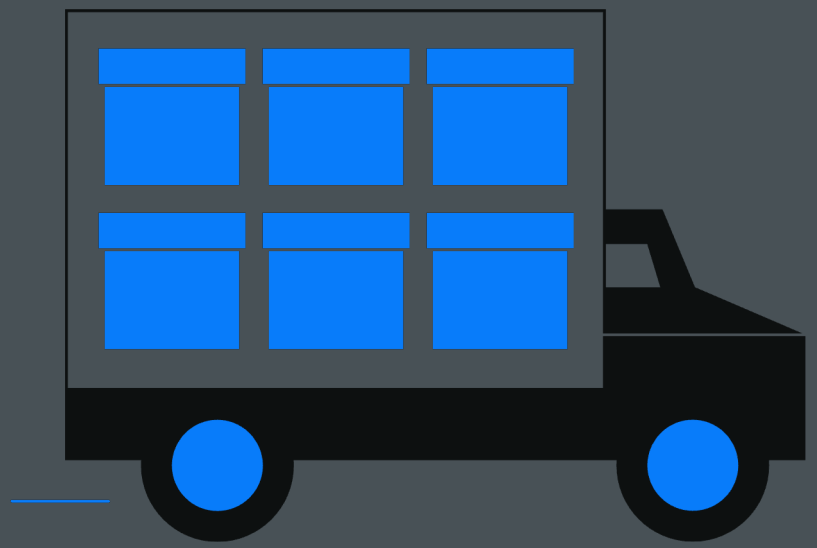


Continuous Delivery

Aufbauen auf der CI

OSP
Otto Group Solution Provider

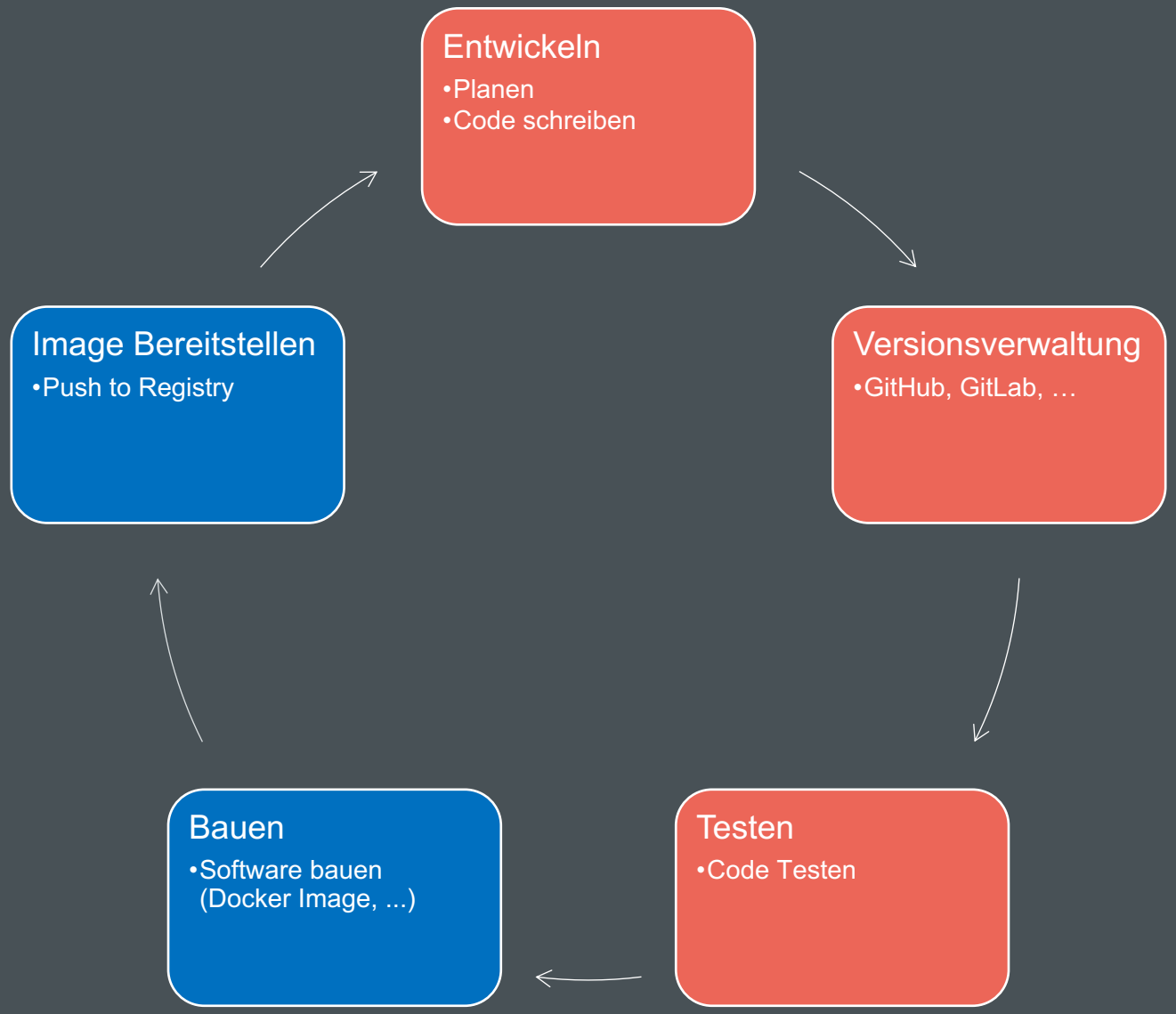
- Continuous Delivery
- Wir liefern die Software aus -> Delivery
- Wir nutzen dafür automatisierte Build Prozesse
- Wir „verpacken“ oder „bauen“ unsere Software in ein Packet
- Das Packet kann dann schnell ausgeliefert werden
- Alles geschieht automatisiert



Continuous Delivery

Zyklus

OSP
Otto Group Solution Provider



Continuous Delivery

Beispiel Build

```yaml
build-and-push-image:
  runs-on: ubuntu-latest
  needs: test
  steps:
    - name: Checkout repo
      uses: actions/checkout@v4

    - name: Set up QEMU
      uses: docker/setup-qemu-action@v3

    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v3

    - name: Login to GitHub Container Registry
      uses: docker/login-action@v3
      with:
        registry: ghcr.io
        username: ${{ github.repository_owner }}
        password: ${{ secrets.REGISTRY_TOKEN }}

    - name: Extract metadata (tags, labels) for Docker
      id: meta
      uses: docker/metadata-action@v5.5.1
      with:
        images: ghcr.io/${{ github.repository }}

    - name: Build Image and push to registry
      uses: docker/build-push-action@v5
      with:
        context: ./todo-api-python
        platforms: linux/amd64,linux/arm64
        push: true
        tags: ${{ steps.meta.outputs.tags }}
        labels: ${{ steps.meta.outputs.labels }}
```

Build Schritt

Dockerfile

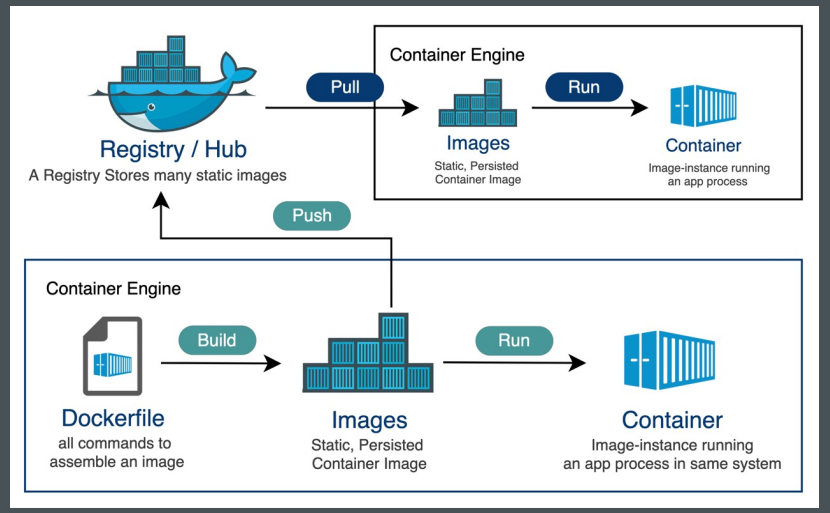
```dockerfile
#
FROM python:3.11

#
WORKDIR /code

#
COPY ./requirements.txt /code/requirements.txt

#
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

#
COPY ./app /code/app

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

#
CMD wait-for-it -s db:5432 -t 5 && uvicorn app.main:app --host 0.0.0.0 --port 8000
```

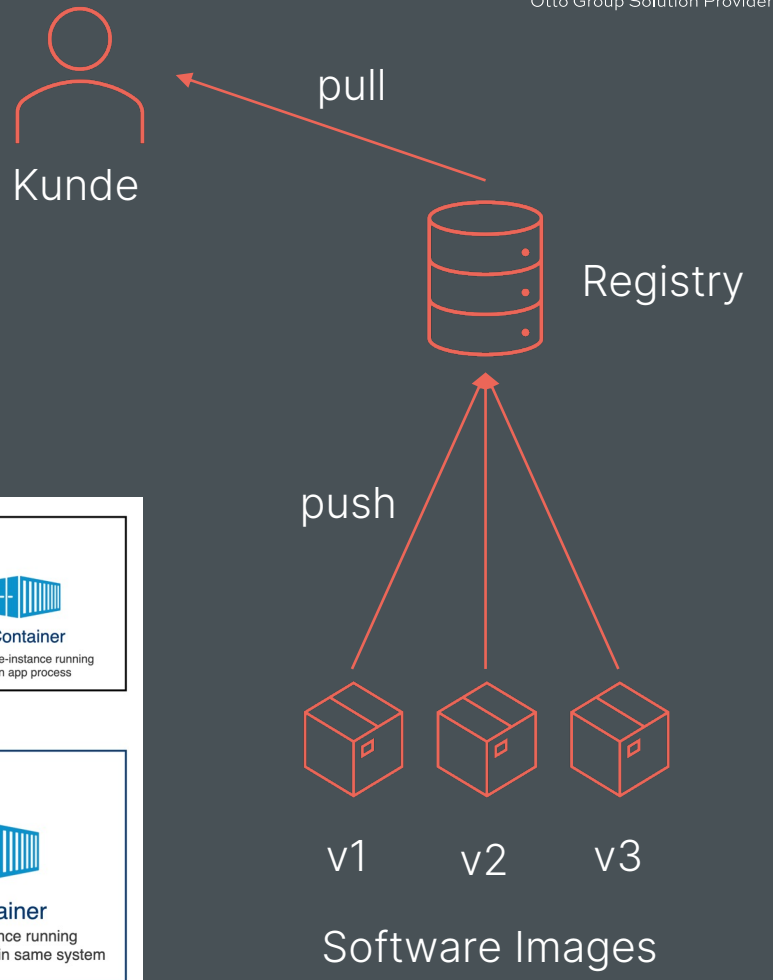
- Wir verpacken die API als Docker Image
- Ausführen aller notwendigen Befehle zum Einrichten

Continuous Delivery

Registry

- Das Image wird in eine Registry hochgeladen
- Registry ist eine Server-Anwendung welche Images speichert
- Dort können Images von Clients abgerufen werden
- Beispiele sind
  - Docker Hub
  - Amazon ECR
  - GitHub Packages (unser Tool)





Continuous Delivery

Image Abrufen

OSP
Otto Group Solution Provider

- Unser Image wurde erfolgreich durch die CI/CD-Pipeline bereitgestellt
- Der Kunde kann unser Image dann nach Belieben aus der Registry abrufen
- Zusätzlich kann gezielt eine Version des Images abgerufen werden

```
ruben@OSP3737:~/dev$ docker login ghcr.io -u USERNAME --password-stdin
```

```
ruben@OSP3737:~/dev$ docker pull ghcr.io/edlaser/cicd-example:main
```

Befehl zum Abrufen (Vorher muss ein Login erfolgen)

Continuous Delivery

Gelöste Probleme / Gewinn

OSP|
Otto Group Solution Provider

- ✓ Wir denken in kleinen Releases
- ✓ Wir bauen unsere Images automatisiert
- ✓ Unser Kunde kann neue Funktionalitäten schnell abrufen
- ✓ Wir haben keinen händischen Aufwand neue Releases auszuliefern

Unser Stand

GitHub Repository, GitHub Actions eingeführt

OSP|
Otto Group Solution Provider

Das können wir

- ✓ Kollaborativ erfolgreich Arbeiten
- ✓ Software durch Tests Qualität halten
- ✓ Fehler direkt erkennen
- ✓ Dokumentation generieren
- ✓ Software bauen
- ✓ Software „bereit halten“

Wir wollen zusätzlich

- × Software ausliefern

OSP|
Otto Group Solution Provider

Fragen?

OSP

Otto Group Solution Provider

Continuous Deployment

OSP
Otto Group Solution Provider

- Wie kriegt man Zugriff auf unsere Software?
- Wie kommt unsere Software zum Kunden?

Continuous Deployment

Was möchten wir erreichen?

OSP
Otto Group Solution Provider

- Wir wollen unsere Software für den Kunden bereitstellen
- Die Software soll direkt nutzbar sein, ohne Images abrufen zu müssen
- Bereitstellen mit möglichst wenig Aufwand

Continuous Deployment

CD with extra Steps

OSP
Otto Group Solution Provider

- Continuous Deployment
- Wir deployen unsere Software
- Von Validierung (CI) über Lieferung (CD) bis zum produktiven Einsatz
- Sie ist in einer fertigen Version direkt nutzbar
- Updates direkt im Betrieb eingespielt

CD

Continuous Deployment

Zyklus

Continuous Deployment

Gelöste Probleme / Gewinn

OSP
Otto Group Solution Provider

- ✓ Wir haben ohne Aufwand eine neue Version produktiv gesetzt
- ✓ Unser getesteter Code kann direkt verwendet werden
- ✓ Features sind direkt „sichtbar“
- ✓ Mehrere Releases am Tag möglich

Unser Stand

GitHub Repository, GitHub Actions eingeführt

OSP
Otto Group Solution Provider

Das können wir

- ✓ Kollaborativ erfolgreich Arbeiten
- ✓ Software durch Tests Qualität halten
- ✓ Fehler direkt erkennen
- ✓ Dokumentation generieren
- ✓ Software bauen
- ✓ Software „bereit halten“
- ✓ Software ausliefern

Wir wollen zusätzlich

- ✓ Nichts, wir sind happy!

OSP|
Otto Group Solution Provider

- Haben wir dadurch eventuell Nachteile?

OSP|
Otto Group Solution Provider

Fragen?

OSP

Otto Group Solution Provider

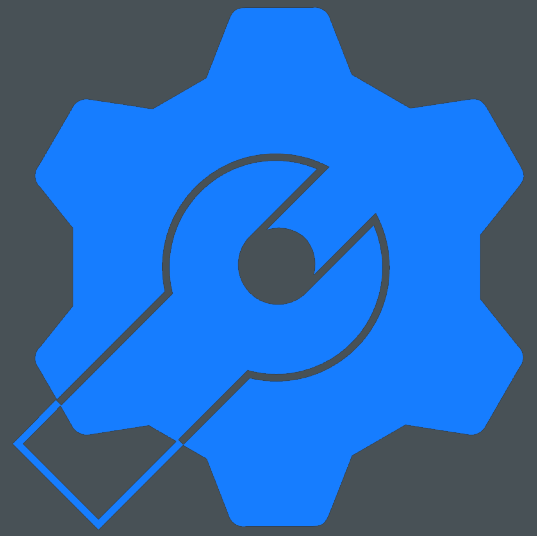
Best Practices

CI/CD

So wollen wir arbeiten

OSP
Otto Group Solution Provider

- Häufig Commiten
- Fleißig mergen
- Fehler direkt beheben
- Nicht zu viel und nicht zu wenig testen
- Alle nehmen Teil



OSP

Otto Group Solution Provider

Fazit

Learnings

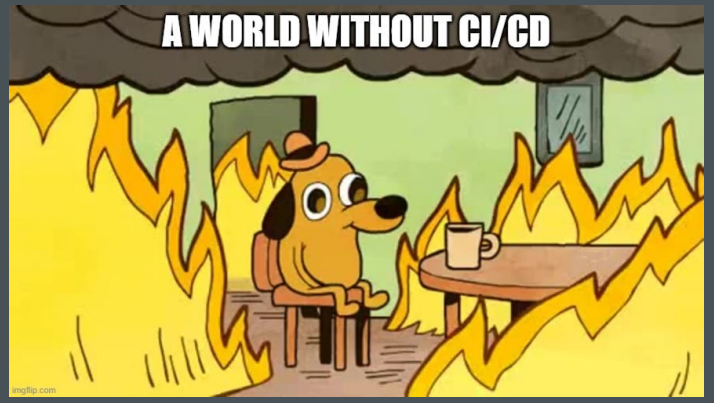
OSP
Otto Group Solution Provider

- Nutzen eines Repositorys zur Verwaltung von Code und Dokumenten -> Continuous Integration
  - Vereinfach uns kollaborative Arbeit
- Bereitstellen unserer Software als Images -> Continuous Delivery
  - Software kann schnell bei neuen Versionen abgerufen werden
- Bereitstellen unserer Umgebung in Produktion (Live) -> Continuous Deployment
  - Neue Features/Fehlerbehebungen sind direkt nutzbar
- CI/CD automatisiert gängige Aufgaben in der Entwicklung und Betreiben von Software
  - Weniger händische Arbeit
  - Fusion von Entwicklung und Betreiben

Was haben wir erreicht?

OSP
Otto Group Solution Provider

- Wir:
  - können unsere Dokumentation automatisiert bei Änderungen generieren
  - testen unsere Software bei Änderungen ohne händischen Aufwand
  - bauen unsere Software und stellen sie in einer Registry bereit
  - haben nervige händische Aufgaben automatisiert
  - erhalten direkt Feedback über unsere Codequalität



OSP

Otto Group Solution Provider

Wir haben es geschafft! Noch Fragen?