Zürcher Hochschule
für Angewandte Wissenschaften

School of
Management and Law

AACSB ACCREDITED
Seit 2015

# DevOps
# Cloud Deployment

Building Competence. Crossing Borders.

**Adrian Moser**
mosa@zhaw.ch, FS2024

# Deployment

# Deployment

Beim Deployment (Deutsch: Softwareverteilung ☺) wird eine Software oder mehrere Softwarekomponenten auf einem oder mehreren Rechner(n) installiert.

- Das Deployment läuft in der Regel voll- oder halbautomatisch ab.
- Bei lokal installierten Programmen muss die Installation auf jedem einzelnen Rechner durchgeführt werden.
- Bei Webapplikationen muss die Software nur auf den Servern erfolgen.

# Continuous Delivery

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

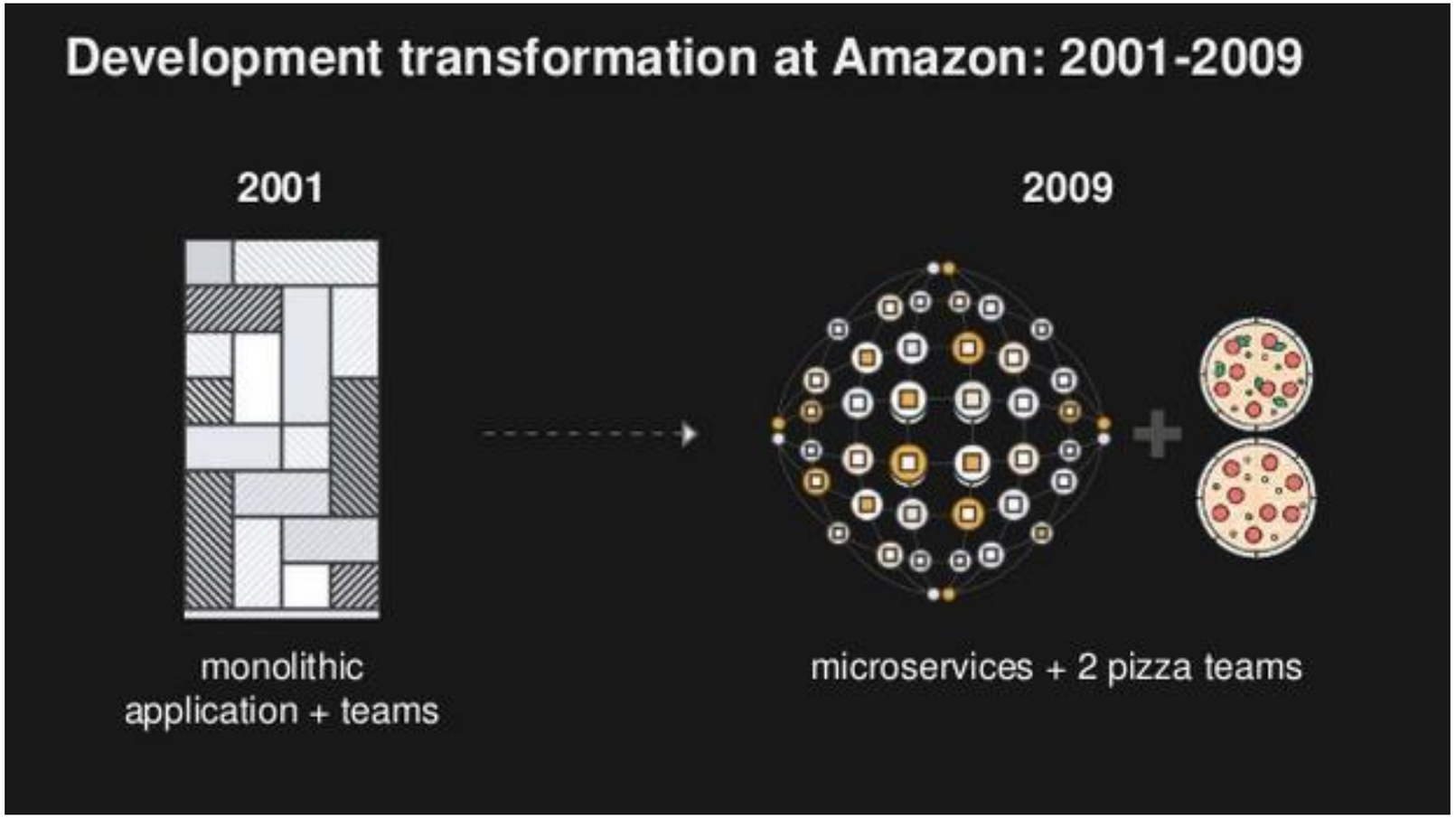
pexels.com

https://en.wikipedia.org/wiki/Continuous_delivery

# Warum ist Continuous Delivery sinnvoll?

Amazon Beispiel

# Beispiel: Entwicklung CI/CD bei Amazon




https://de.slideshare.net/AmazonWebServices/session-7-continuous-integration-amp-continuous-delivery-on-aws-raghuraman-balachandran

# DevOps: Culture + Practices + Tools

Each 2-pizza team “owns” their product:

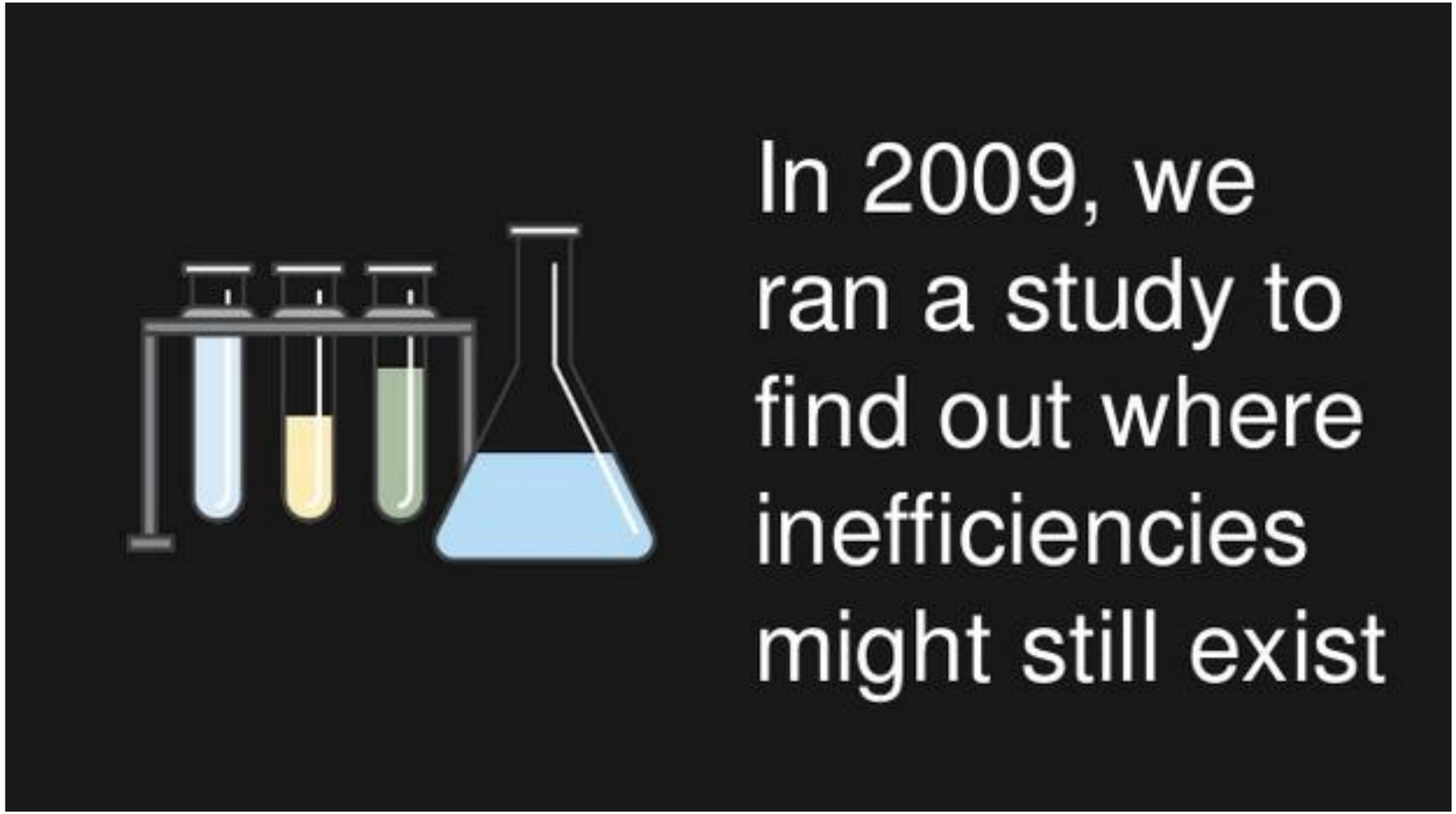
- Creates product (software typically)
- Handles Q/A of that product
- Responds to issues, is on-call
  - “you build it, you run it”
- Supports service & tracks/goals against business and technical metrics

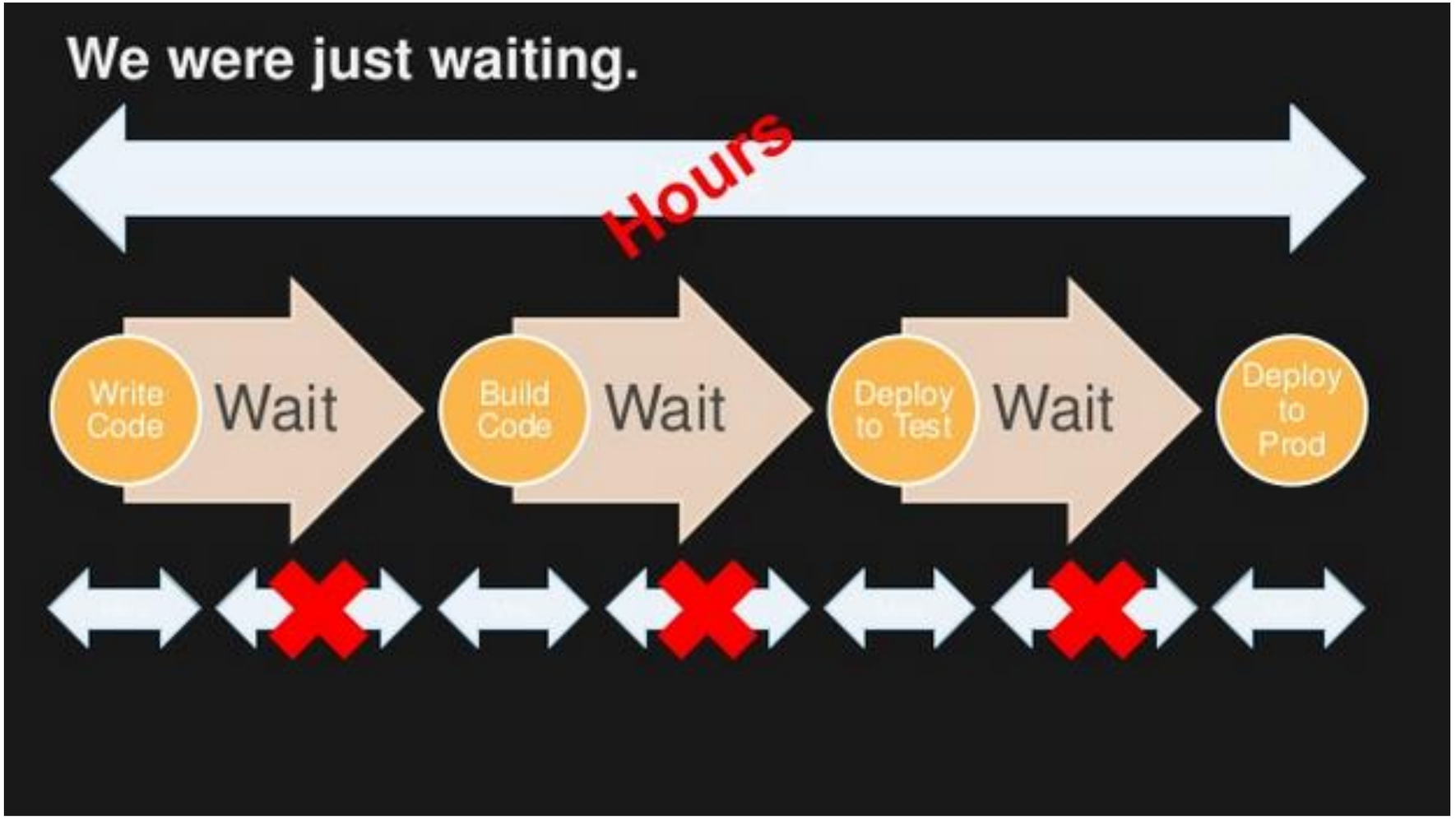
# DevOps: Culture + Practices + Tools

Each 2-pizza team's practices largely open so far as standards are met:

- Agile? Scrum? Daily standups? Weekly? None? Whatever you works for your team!
- No centralized change management board/team/approval, but tools that require a degree of sign-off/process review

https://secure.flickr.com/photos/ox/9408028555


In 2009, we ran a study to find out where inefficiencies might still exist


We were just waiting.
Hours
Write Code
Wait
Build Code
Wait
Deploy to Test
Wait
Deploy to Prod


We built tools to
automate our software
release process

**We move pretty fast at Amazon:**

In 2014:

- Thousands of service teams across Amazon
- + Building microservices
- + Practicing continuous delivery
- + Many environments (staging, beta, production, multiple regions)

**=50 million deploys**

# Azure

# Platform as a Service (PaaS)

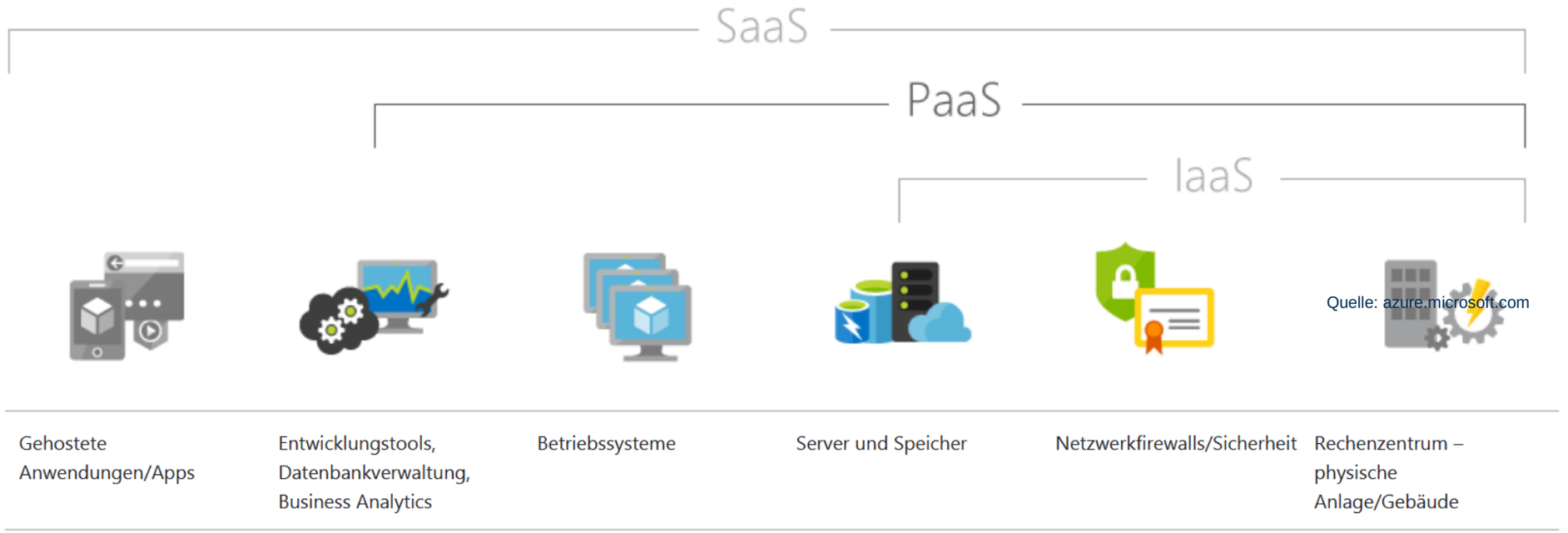
The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

https://en.wikipedia.org/wiki/Cloud_computing#Platform_as_a_service_(PaaS)

pexels.com

# SaaS vs. PaaS vs. IaaS

- Cloud-Computing-basierte Plattform von Microsoft
- Plattform als Service (PaaS)
- .NET, node.js, PHP, Python, Java und andere



# Azure Begriffe

| | |
|---|---|
| Portal | Web-Applikation zur Konfiguration von Azure Services |
| Ressource | Teil eines Azure Service, z.B. Datenbank oder VM |
| Ressourcegruppen | Ein Set von Ressourcen für eine App |
| Region | Ein oder mehrere Datacenter innerhalb eines Gebietes |
| Azure CLI | Azure Command Line Interface |

# Azure Deployment Methoden

Azure App Service Deployment Methoden

- ZIP Deployment (az webapp up)
- Local Git Deployment
- App Service WebHook GitHub
- GitHub Actions
- Docker Image oder Docker Compose
- …

# Azure «Hello World»

Azure «Hello World» über «Zip Deployment» deployen

# Demo App mit Node / Express


```
'use strict';

const express = require('express');

// Constants
const PORT = process.env.PORT || 8080;
const HOST = '0.0.0.0';

// App
const app = express();
app.get('/', (req, res) => {
  res.send('Hello DevOps Course :-))))');
});

app.listen(PORT, HOST);
console.log(`Running on
http://${HOST}:${PORT}`);
```

**Express Web App**

Express ist ein Microservice-Framework für Node, analog Spring Web für Java.

Stellt unter dem Pfad «/» eine Antwort zur Verfügung

**Lokal starten**

Auf lokalem Rechner installieren, starten und mit Browser http://localhost:8080/ überprüfen

```
npm install
npm run start
```

# Azure Ports

Tutorial

**Ports**

- Es kann **nicht** ein beliebiger Port verwendet werden
- Der Port wird von Azure in der Variable **process.env.PORT** vorgegeben
- Danach wird die Applikation von Azure auf dem Standardport 80 zur Verfügung gestellt

Port-Variable verwenden

```
'use strict';

const express = require('express');

// Constants
const PORT = process.env.PORT || 8080;
const HOST = '0.0.0.0';

// App
const app = express();
app.get('/', (req, res) => {
  res.send('Hello DevOps Course :-)))');
});

app.listen(PORT, HOST);
console.log(`Running on
http://${HOST}:${PORT}`);
```

# Azure Ressourcengruppe

Tutorial

Eine Ressourcengruppe ist ein Container, der zugehörige Ressourcen für eine Azure-Lösung enthält.

```
az login

az group create --name dop-helloworld --location switzerlandnorth
```

Die Ressourcengruppe kann alle Ressourcen für die Lösung enthalten oder nur die Ressourcen, die Sie als Gruppe verwalten möchten. Sie entscheiden, wie Sie Ressourcen Ressourcengruppen zuweisen möchten, basierend darauf, was für Ihre Organisation am sinnvollsten ist.

# Azure «Hello World» mit ZIP-Deployment

Tutorial

## Azure Deployment

```
az webapp up --name devops-mosa-helloworld --plan devops-helloworld --resource-group
devops-helloworld --runtime NODE:20-lts --sku F1
```

Eigenen Namen wählen

## Hinweise

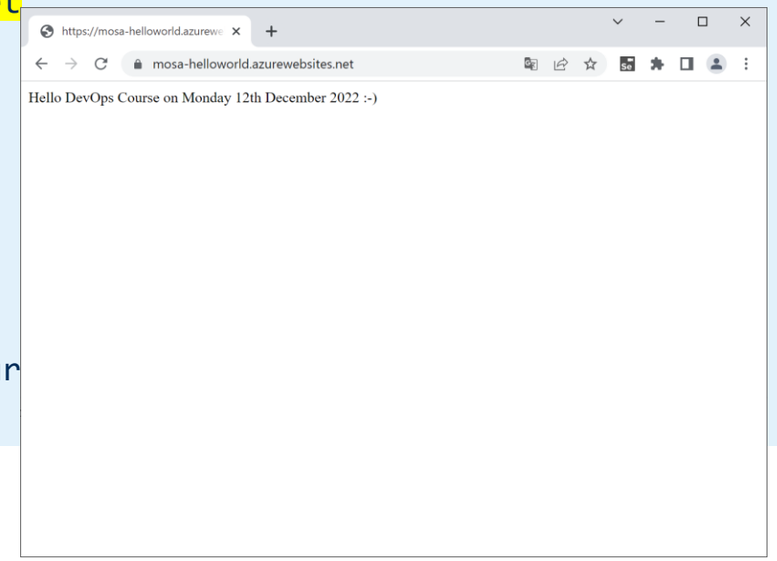
- Vorher bestehende F1 Apps **und Pläne** entfernen und Löschung abwarten!
- «node-modules»-Ordner vor Deployment löschen, damit er nicht übertragen werden muss
- Deployment kann wiederholt werden, App wird aktualisiert

# Azure «Hello World»

Tutorial

## Log

```
C:\mosa\dev\edu\DevOps\12a DevOpsAzure>az webapp up --name mosa-helloworld --plan mosa-helloworld
--resource-group zhaw-mdm-westeurope --runtime NODE:18-lts --sku F1
The webapp 'mosa-helloworld' doesn't exist
Creating AppServicePlan 'mosa-helloworld' ...
Creating webapp 'mosa-helloworld' ...
...
{
  "URL": "http://devops-mosa-helloworld.azurewebsites.net"
  "appserviceplan": "mosa-helloworld",
  "location": "centralus",
  "name": "mosa-helloworld",
  "os": "Linux",
  "resourcegroup": "zhaw-mdm-westeurope",
  "runtime_version": "NODE|18-lts",
  "runtime_version_detected": "-",
  "sku": "FREE",
  "src_path": "C:\\mosa\\dev\\edu\\DevOps\\12a DevOpsAzur
}
```



# Azure und Docker

Docker image auf Azure deployen

# Warum Docker mit Cloud Service kombinieren?

**Docker Deployment**

- Kann spezifische Applikationen umfassen (z.B. Datenbank)
- Standardisiert, verfügbar für mehrere Cloud-Anbieter
- Unabhängig von Anforderungen der Cloud-Anbieter
- Lokales Deployment entspricht 1:1 dem Cloud Deployment
- Für **Port** muss dennoch Variable verwendet werden

**Vorgehen**

- Azure CLI installieren und falls notwendig anmelden (az login)
- Container erstellen und lokal testen
- Container pushen
- Azure App mit Container deployen

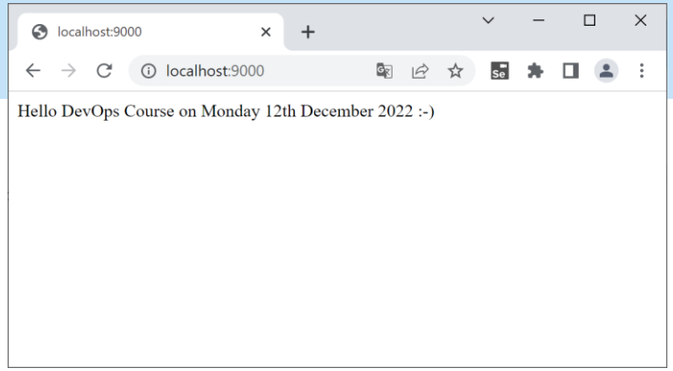
# Docker: Build und Push auf Docker Hub

Docker Image bauen und pushen:

```
docker build -t mosazhaw/devopsazure:latest .
```

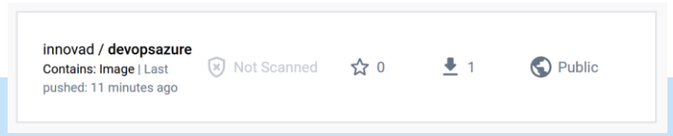
Docker lokaler Test:

```
docker run -p 9000:8080
-d mosazhaw/devopsazure:latest
```




Push auf Docker Hub

```
docker push mosazhaw/devopsazure:latest
```



# Azure Web App Container

App Service Plan erstellen (Vorbedingung: leere Resource-Group)

```
az appservice plan create --name devops-container --resource-group devops-container --sku F1 --is-linux
```

App erstellen

```
az webapp create --resource-group devops-container --plan devops-container --name devops-container --deployment-container-image-name mosazhaw/devopsazure:latest
```

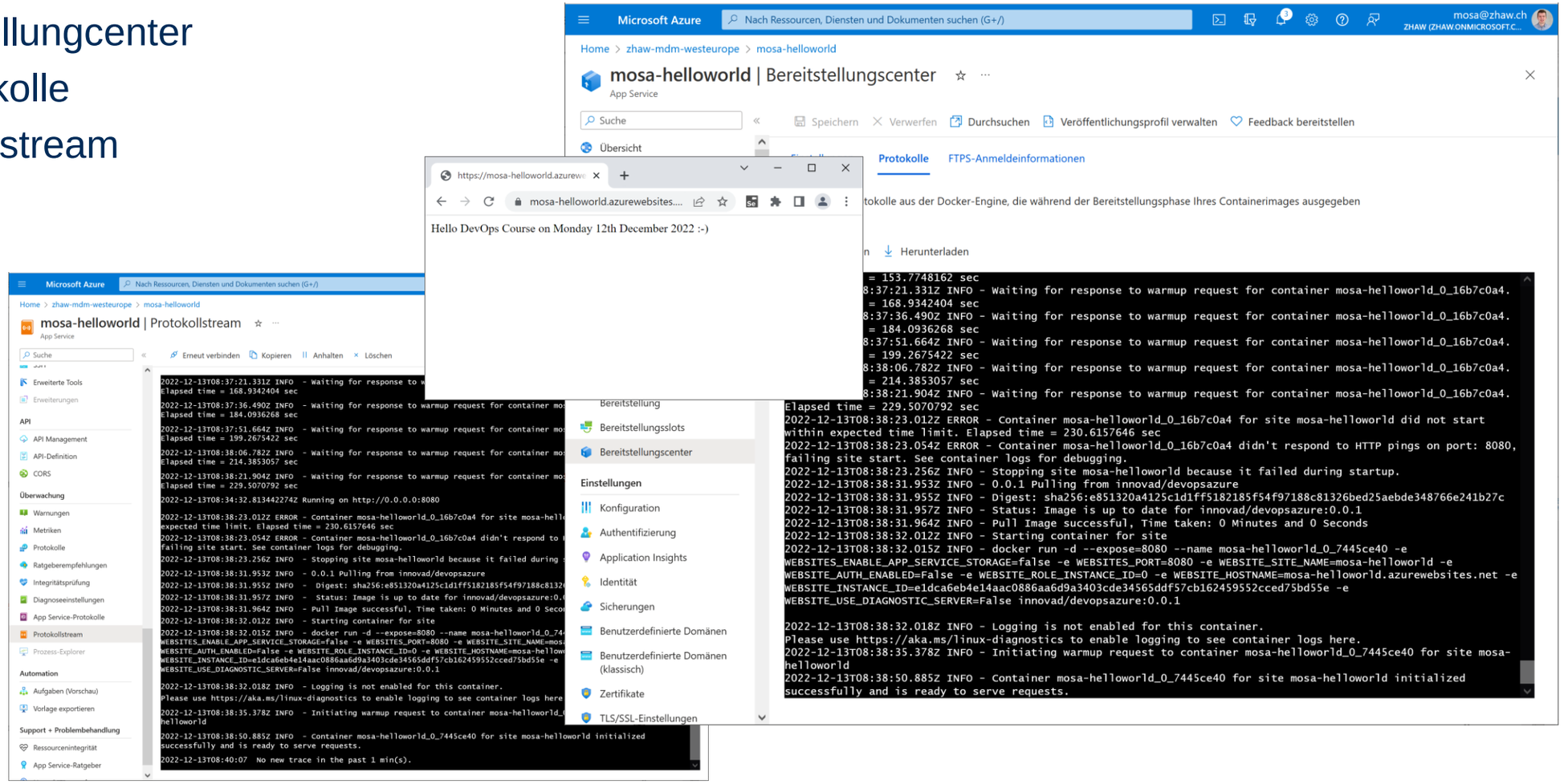
Eigenen Namen wählen

Port konfigurieren

```
az webapp config appsettings set --resource-group devops-container --name devops-container --settings WEBSITES_PORT=8080
```

# Troubleshooting

- Bereitstellungcenter
  → Protokolle
- Protokollstream



# Azure Docker Deployment automatisieren

Automatisierung mit GitHub Actions

# GitHub Actions

**Konzept**

CI/CD-Vorgänge wie Build, Test und Deployment direkt aus dem GitHub-Repository heraus ausführen (sogenannte Workflows). Begrenzte Menge an monatlichen freien Minuten (https://github.com/settings/billing)

**Workflow**

Ein Workflow besteht aus einem oder mehreren Jobs. Workflows sind in **YAML-Files** (YAML Ain't Markup Language) definiert und werden im .github-Verzeichnis im Repository gespeichert.

**Ausführung**

Manuell über das GitHub-Webinterface oder bevorzugt beim Ausführen eines Push auf das GitHub-Repository.

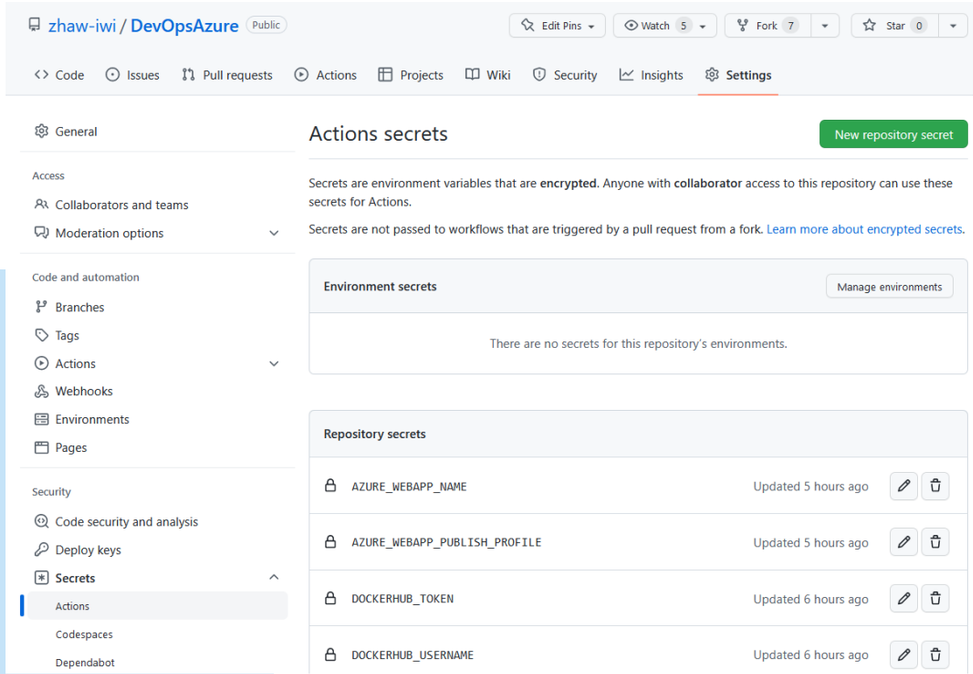
```yaml
name: docker hub push

on:
  push:
    branches:
      - 'main'
```

# GitHub Actions

## Secrets

GitHub Actions können auch Zugangsdaten wie Passwörter enthalten. Diese werden über Variablen referenziert.

```
jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up QEMU
        uses: docker/setup-qemu-action@v2
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2
      -
        name: Login to Docker Hub
        uses: docker/login-action@v2
        with:
          username: ${{secrets.DOCKERHUB_USERNAME}}
          password: ${{secrets.DOCKERHUB_TOKEN}}
```



# Hinweise zur Umsetzung

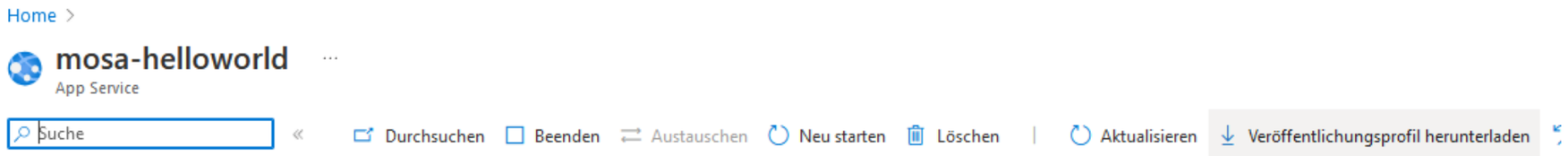
## GitHub Action Secrets

Folgende Secrets müssen unter https://github.com/XYZ/settings/secrets/actions erfasst werden:

- DOCKERHUB_USERNAME
- DOCKERHUB_TOKEN
- AZURE_WEBAPP_NAME
- AZURE_WEBAPP_PUBLISH_PROFILE

## Veröffentlichungsprofil

Das «Publish Profile» lässt sich im Azure Portal herunterladen und der Inhalt als Secret einfügen



# Dockerize DevOpsDemo

Nun wollen wir die Erkenntnisse der Beispiel-Applikation auf DevOpsDemo anwenden.

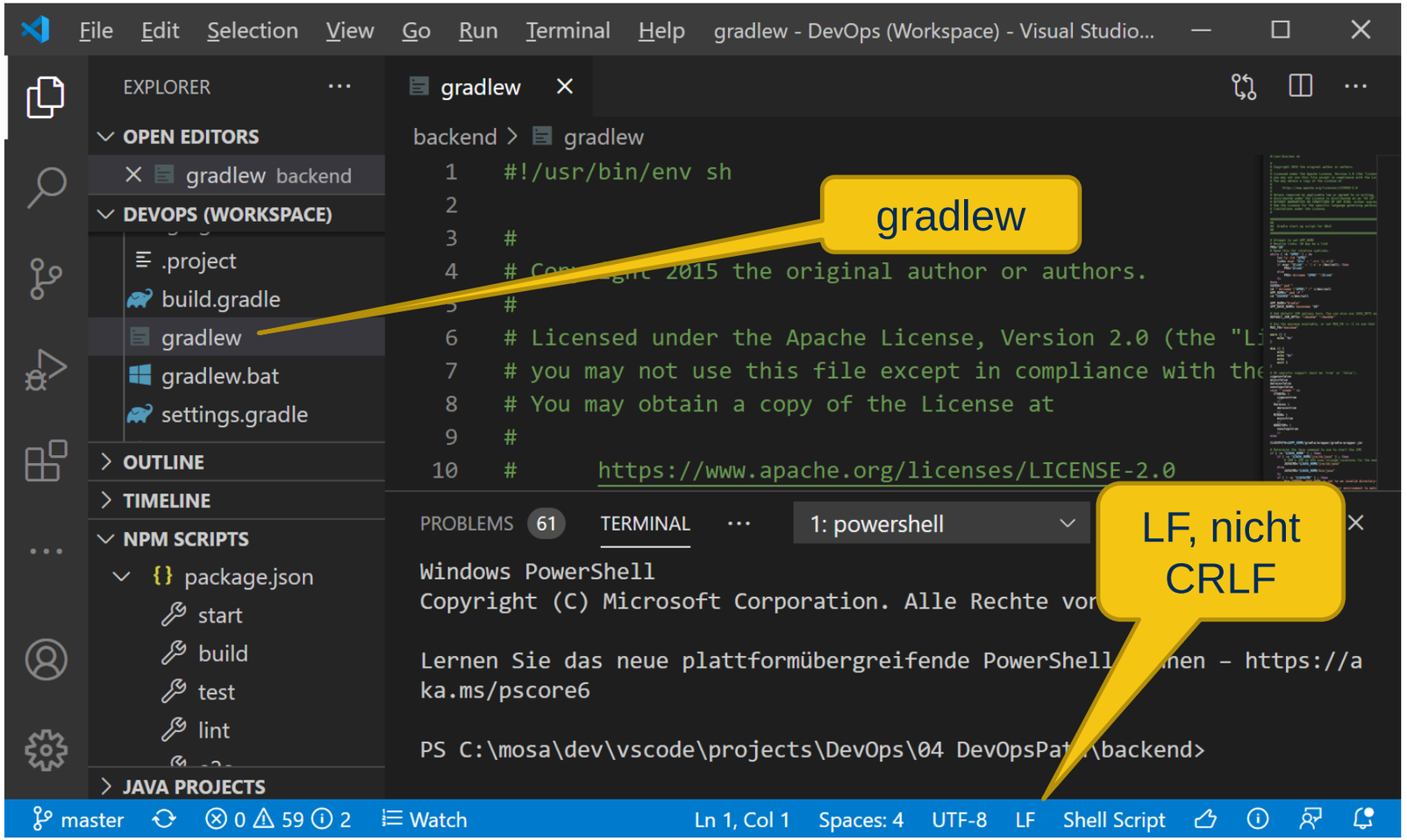
# Dockerfile


## Dockerfile mit **Java** und **Node**

```
FROM openjdk:17-jdk-slim
RUN apt-get update && apt-get install -y curl \
  && curl -sL https://deb.nodesource.com/setup_14.x | bash - \
  && apt-get install -y nodejs \
  && curl -L https://www.npmjs.com/install.sh | sh

WORKDIR /usr/src/app

COPY . .

RUN cd frontend && npm install
RUN cd frontend && npm run build
RUN cd backend && chmod +x gradlew
RUN cd backend && ./gradlew build

EXPOSE 4567
CMD ["java", "-jar", "/usr/src/app/backend/build/libs/demo-0.0.1-SNAPSHOT.jar"]
```

# Achtung: gradlew Zeilenumbruch (Windows)

Tutorial



# Lokaler Build und Deployment

Tutorial

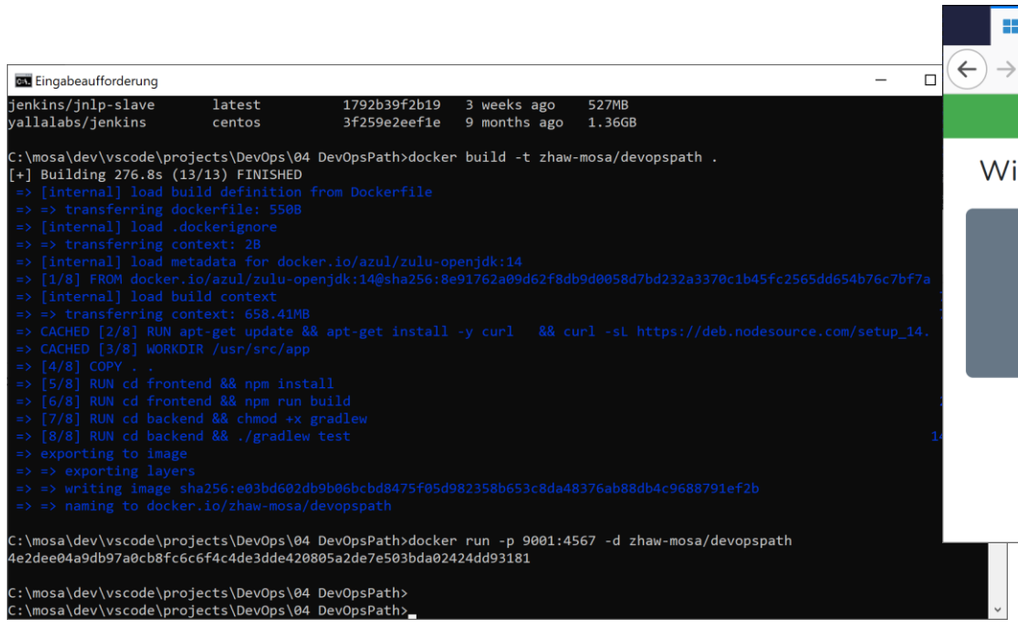
**Terminal / Konsole im Projektverzeichnis**

docker build -t zhaw-mosa/devopsdemo .

Spring Standard-Port

**DevOpsDemo local starten**

docker run -p 9001:8080 -d zhaw-mosa/devopsdemo



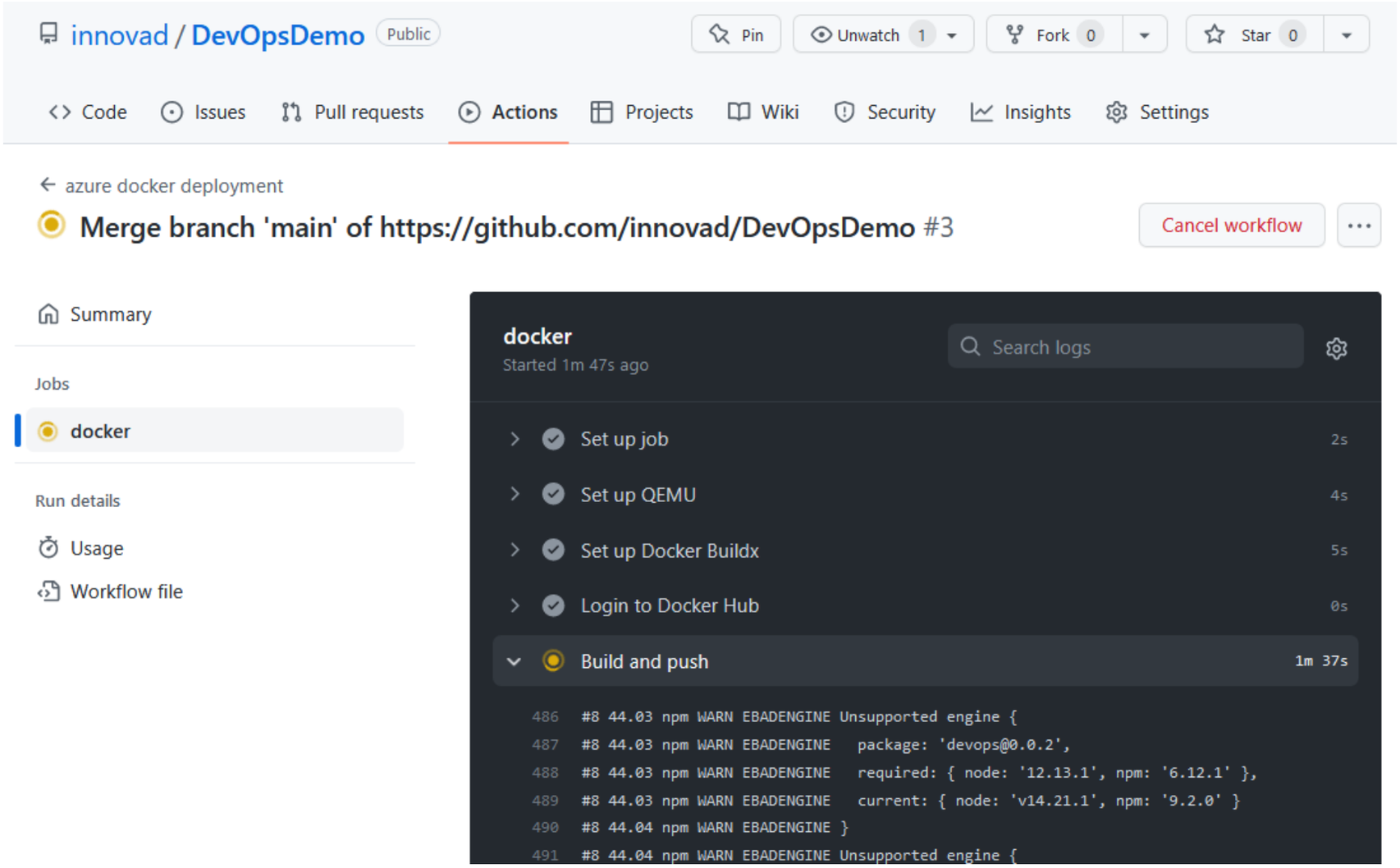
# Automatisierung GitHub Actions

## GitHub Action Secrets

Analog zu vorherigem Beispiel folgende Secrets setzen:

- DOCKERHUB_USERNAME
- DOCKERHUB_TOKEN
- AZURE_WEBAPP_NAME
- AZURE_WEBAPP_PUBLISH_PROFILE

# Troubleshooting: GitHub Web Interface



# Lernjournal

# Lernjournal «Cloud Deployment»


## Lernziele

- Azure als Beispiel für einen Cloud-Service kennenlernen
- Azure Zip-Deployment durchführen
- Azure Docker Deployment
- Automatisierung mit GitHub-Actions

## Checkliste

- ✓ Git-Repository «Node-Beispiel» mit automatisiertem Deployment auf Azure
- ✓ Docker Image «Node-Beispiel» auf Docker Hub vorhanden
- ✓ Mehrere Änderungen und Deployments sind dokumentiert
- ✓ Git-Repository «DevOpsDemo» mit Dockerfile
- ✓ Docker Image «DevOpsDemo» auf Docker Hub vorhanden
- ✓ Mehrere Änderungen (z.B. weitere Kacheln) und Deployments sind dokumentiert