

DevOps Cloud Deployment



Building Competence. Crossing Borders.

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.

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

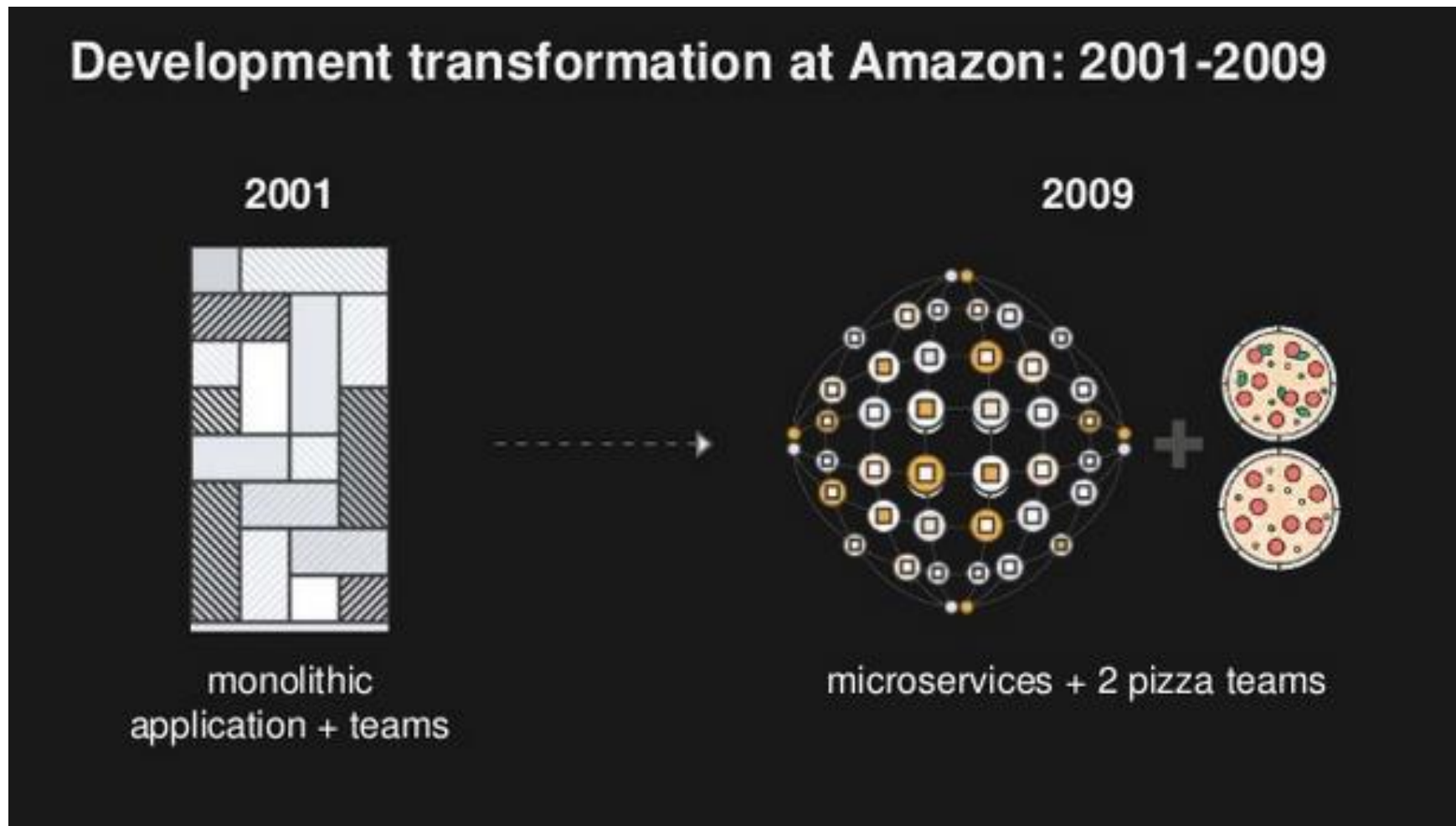


pexels.com

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



<https://secure.flickr.com/photos/fox9408028555>

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

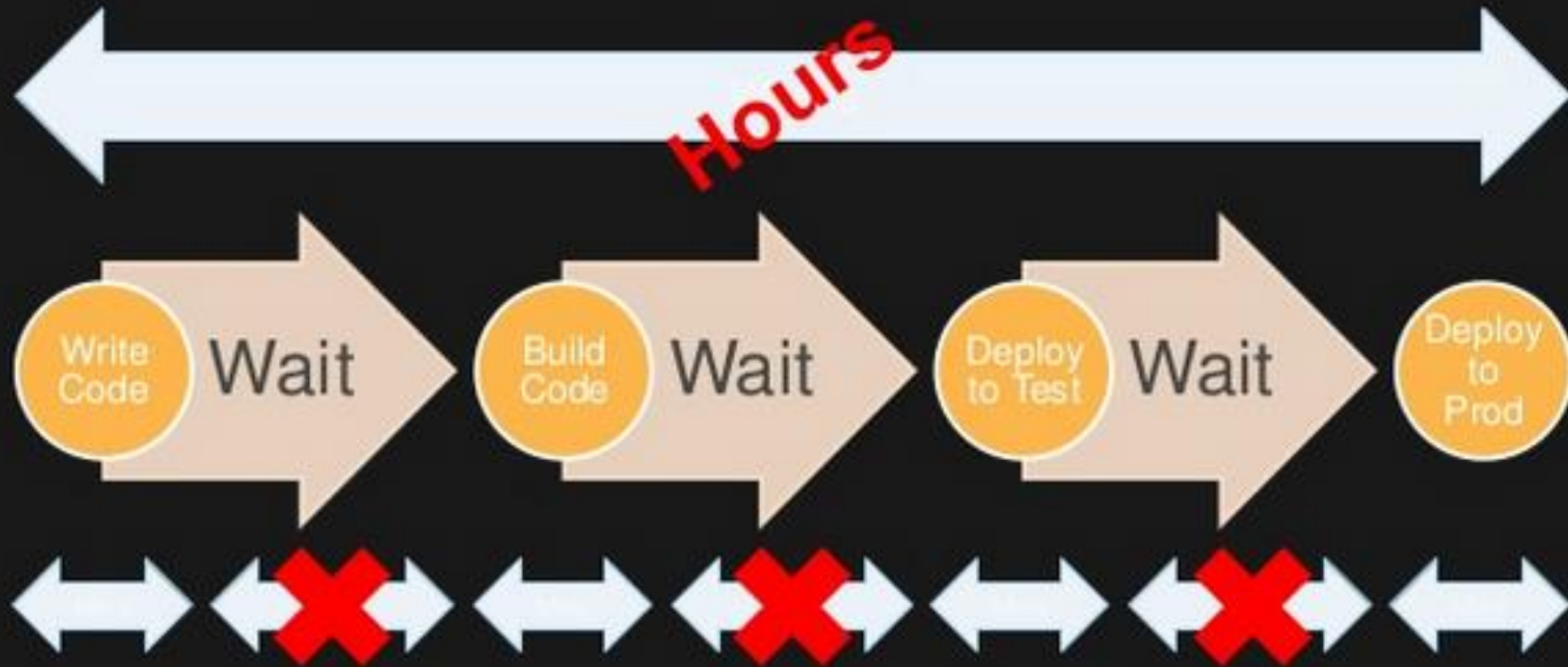


<https://secure.flickr.com/photos/fox9408028555>



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

We were just waiting.





**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.

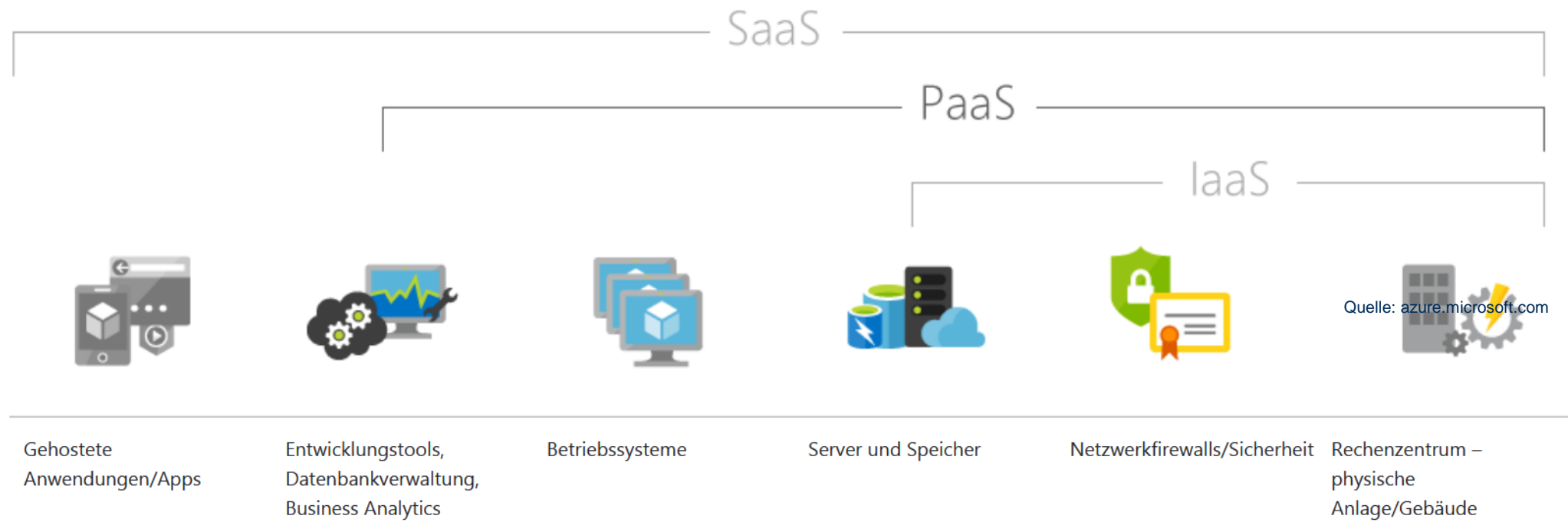
[https://en.wikipedia.org/wiki/Cloud_computing#Platform_as_a_service_\(PaaS\)](https://en.wikipedia.org/wiki/Cloud_computing#Platform_as_a_service_(PaaS))



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


```
'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
```

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}`);
```

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
```

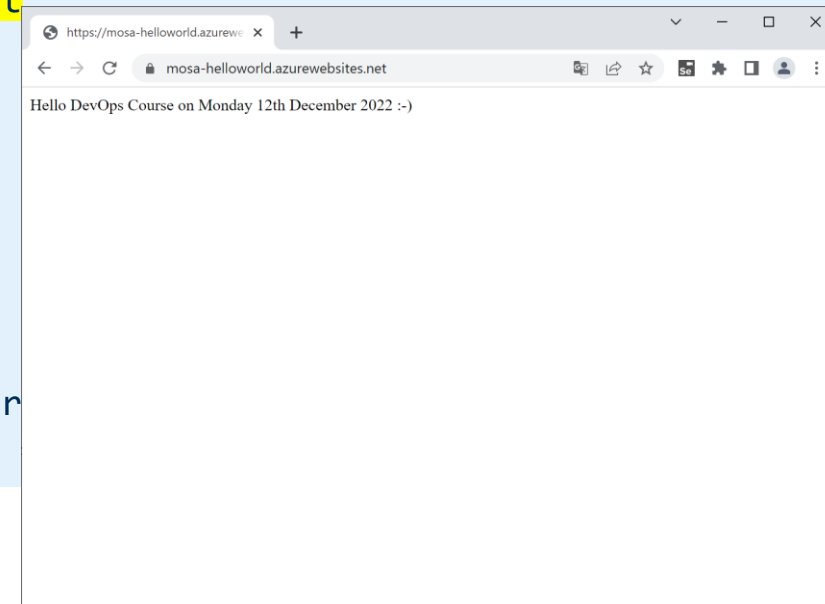
Eigenen Namen wählen

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

Log

```
C:\mosa\dev\edu\DevOps\12a DevOpsAzure>az webapp up --name mosa-helloworld --plan mosa-helloworld
--resource-group zhaw-mdm-westeuropa --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-westeuropa",
  "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

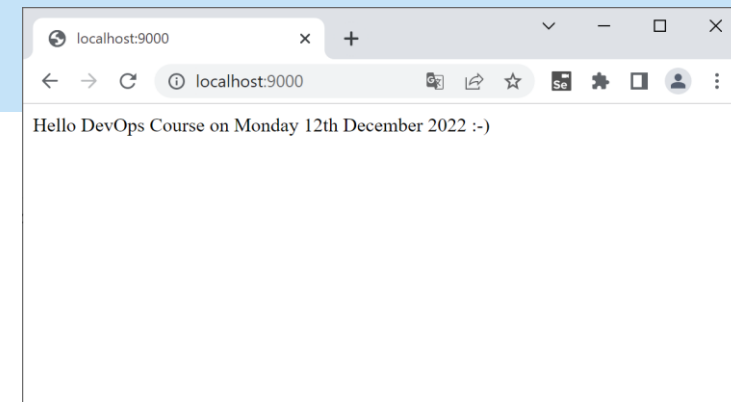
Docker: Build und Push auf Docker Hub

Docker Image bauen und pushen:

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

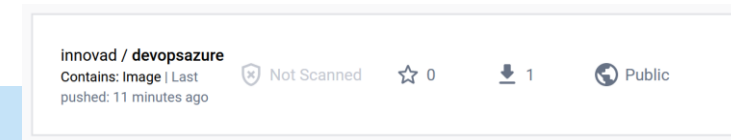
Docker lokaler Test:

```
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

The screenshot displays the Microsoft Azure portal interface for the 'mosa-helloworld' App Service. The 'Protokolle' (Logs) tab is active, showing a detailed log stream. The logs indicate a container startup failure with the following error message: 'Container mosa-helloworld_0_16b7c0a4 for site mosa-helloworld did not start within expected time limit. Elapsed time = 230.6157646 sec'. The logs also show the container being pulled from the registry and the site being initialized. A browser window in the foreground shows the 'Hello DevOps Course on Monday 12th December 2022 :-)' message.

```
2022-12-13T08:37:21.331Z INFO - waiting for response to warmup request for container mosa-helloworld_0_16b7c0a4. Elapsed time = 168.9342404 sec
2022-12-13T08:37:36.490Z INFO - waiting for response to warmup request for container mosa-helloworld_0_16b7c0a4. Elapsed time = 184.0936268 sec
2022-12-13T08:37:51.664Z INFO - waiting for response to warmup request for container mosa-helloworld_0_16b7c0a4. Elapsed time = 199.2675422 sec
2022-12-13T08:38:06.782Z INFO - waiting for response to warmup request for container mosa-helloworld_0_16b7c0a4. Elapsed time = 214.3853057 sec
2022-12-13T08:38:21.904Z INFO - waiting for response to warmup request for container mosa-helloworld_0_16b7c0a4. Elapsed time = 229.5070792 sec
2022-12-13T08:38:23.012Z ERROR - Container mosa-helloworld_0_16b7c0a4 for site mosa-helloworld did not start within expected time limit. Elapsed time = 230.6157646 sec
2022-12-13T08:38:23.054Z ERROR - Container mosa-helloworld_0_16b7c0a4 didn't respond to HTTP pings on port: 8080, failing site start. See container logs for debugging.
2022-12-13T08:38:23.256Z INFO - Stopping site mosa-helloworld because it failed during startup.
2022-12-13T08:38:31.953Z INFO - 0.0.1 Pulling from innovad/devopsazure
2022-12-13T08:38:31.955Z INFO - Digest: sha256:e851320a4125c1dff5182185f54f97188c81326bed25aebde348766e241b27c
2022-12-13T08:38:31.957Z INFO - Status: Image is up to date for innovad/devopsazure:0.0.1
2022-12-13T08:38:31.964Z INFO - Pull Image successful, Time taken: 0 Minutes and 0 Seconds
2022-12-13T08:38:32.012Z INFO - Starting container for site
2022-12-13T08:38:32.015Z INFO - docker run -d --expose=8080 --name mosa-helloworld_0_7445ce40 -e WEBSITES_ENABLE_APP_SERVICE_STORAGE=false -e WEBSITES_PORT=8080 -e WEBSITE_SITE_NAME=mosa-helloworld -e WEBSITE_AUTH_ENABLED=false -e WEBSITE_ROLE_INSTANCE_ID=0 -e WEBSITE_HOSTNAME=mosa-helloworld.azurewebsites.net -e WEBSITE_INSTANCE_ID=e1dca6eb4e14aac0886aa6d9a3403cde34565dddf57cb162459552cccd75bd55e -e WEBSITE_USE_DIAGNOSTIC_SERVER=false innovad/devopsazure:0.0.1
2022-12-13T08:38:32.018Z INFO - Logging is not enabled for this container. Please use https://aka.ms/linux-diagnostics to enable logging to see container logs here.
2022-12-13T08:38:35.378Z INFO - Initiating warmup request to container mosa-helloworld_0_7445ce40 for site mosa-helloworld
2022-12-13T08:38:50.885Z INFO - Container mosa-helloworld_0_7445ce40 for site mosa-helloworld initialized successfully and is ready to serve requests.
2022-12-13T08:40:07 No new trace in the past 1 min(s).
```


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.

```
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}
```

The screenshot shows the GitHub interface for the repository 'zhaw-iwi / DevOpsAzure'. The 'Settings' tab is selected, and the 'Actions secrets' section is active. The left sidebar shows the repository's navigation menu, with 'Secrets' highlighted. The main content area displays the 'Actions secrets' page, which includes a 'New repository secret' button and a list of existing secrets. The secrets listed are:

Secret Name	Updated	Actions
AZURE_WEBAPP_NAME	Updated 5 hours ago	Yes
AZURE_WEBAPP_PUBLISH_PROFILE	Updated 5 hours ago	Yes
DOCKERHUB_TOKEN	Updated 6 hours ago	Yes
DOCKERHUB_USERNAME	Updated 6 hours ago	Yes

Hinweise zur Umsetzung

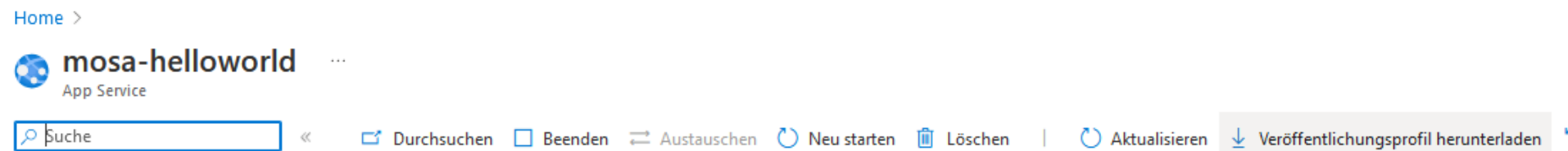
GitHub Action Secrets

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 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

The screenshot shows the Visual Studio Code interface with the 'gradlew' file selected in the Explorer. The file content is displayed in the editor, showing a shell script header. The terminal at the bottom shows the command prompt for PowerShell. A yellow callout points to the 'gradlew' file in the Explorer. Another yellow callout points to the terminal output, indicating that the line ending is LF, not CRLF.

```
backend > gradlew
1  #!/usr/bin/env sh
2
3  #
4  # Copyright 2015 the original author or authors.
5  #
6  # Licensed under the Apache License, Version 2.0 (the "License");
7  # you may not use this file except in compliance with the License.
8  # You may obtain a copy of the License at
9  #
10 # https://www.apache.org/licenses/LICENSE-2.0
```

gradlew

LF, nicht CRLF

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`

The screenshot shows a terminal window on the left and a web browser on the right. The terminal window displays the output of the `docker build` and `docker run` commands. The browser window shows the DevOps application running on `localhost:9001`, displaying a welcome message and a button labeled "Studiengang".

```
C:\mosa\dev\vscode\projects\DevOps\04 DevOpsPath>docker build -t zhaw-mosa/devopsdemo .
[+] Building 276.8s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 550B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/azul/zulu-openjdk:14
=> [1/8] FROM docker.io/azul/zulu-openjdk:14@sha256:8e91762a09d62f8db9d0058d7bd232a3370c1b45fc2565dd654b76c7bf7a
=> [internal] load build context
=> => transferring context: 658.41MB
=> CACHED [2/8] RUN apt-get update && apt-get install -y curl && curl -sL https://deb.nodesource.com/setup_14.
=> CACHED [3/8] WORKDIR /usr/src/app
=> [4/8] COPY . .
=> [5/8] RUN cd frontend && npm install
=> [6/8] RUN cd frontend && npm run build
=> [7/8] RUN cd backend && chmod +x gradlew
=> [8/8] RUN cd backend && ./gradlew test
=> exporting to image
=> => exporting layers
=> => writing image sha256:e03bd602db9b06bcd8475f05d982358b653c8da48376ab88db4c9688791ef2b
=> => naming to docker.io/zhaw-mosa/devopsdemo

C:\mosa\dev\vscode\projects\DevOps\04 DevOpsPath>docker run -p 9001:4567 -d zhaw-mosa/devopsdemo
4e2dee04a9db97a0cb8fc6c6f4c4de3dde420805a2de7e503bda02424dd93181

C:\mosa\dev\vscode\projects\DevOps\04 DevOpsPath>
```

The browser window shows the DevOps application running on `localhost:9001`. The page displays a welcome message "Willkommen bei DevOps" and a button labeled "Studiengang". The browser address bar shows `localhost:9001` and the page is signed in as admin.

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

The screenshot shows the GitHub Actions workflow interface for the repository 'innovad / DevOpsDemo'. The workflow is titled 'Merge branch 'main' of https://github.com/innovad/DevOpsDemo #3'. The left sidebar shows the workflow status as 'Running' and lists the jobs: 'docker' (selected), 'Set up job', 'Set up QEMU', 'Set up Docker Buildx', 'Login to Docker Hub', and 'Build and push'. The 'docker' job is expanded, showing a list of steps with their durations: 'Set up job' (2s), 'Set up QEMU' (4s), 'Set up Docker Buildx' (5s), 'Login to Docker Hub' (0s), and 'Build and push' (1m 37s). The 'Build and push' step is expanded, showing a log entry with a warning message: 'WARN EBADENGINE Unsupported engine { package: 'devops@0.0.2', required: { node: '12.13.1', npm: '6.12.1' }, current: { node: 'v14.21.1', npm: '9.2.0' }'.

innovad / DevOpsDemo Public

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

← azure docker deployment

Merge branch 'main' of https://github.com/innovad/DevOpsDemo #3 Cancel workflow

Summary

Jobs

- docker**

Run details

- Usage
- Workflow file

docker Started 1m 47s ago

Search logs

- > ✓ Set up job 2s
- > ✓ Set up QEMU 4s
- > ✓ Set up Docker Buildx 5s
- > ✓ Login to Docker Hub 0s
- ▼ **Build and push** 1m 37s

```
486 #8 44.03 npm WARN EBADENGINE Unsupported engine {
487 #8 44.03 npm WARN EBADENGINE   package: 'devops@0.0.2',
488 #8 44.03 npm WARN EBADENGINE   required: { node: '12.13.1', npm: '6.12.1' },
489 #8 44.03 npm WARN EBADENGINE   current: { node: 'v14.21.1', npm: '9.2.0' }
490 #8 44.04 npm WARN EBADENGINE }
491 #8 44.04 npm WARN EBADENGINE Unsupported engine {
```

Lernjournal

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