

PRAXISPROJEKT

ÜBERSETZUNGS-APP

**MIT AZURE COGNITIVE SERVICES
TRANSLATOR**

+ . AGENDA



- Anforderungsanalyse
- Beschreibung der Entwicklungsumgebung und Umsetzung
- Translator API und Datenbank
- Entwicklung der Web-Anwendung
- Bereitstellung von Server-Infrastruktur in Azure
- Ausrollen der Web-Anwendung auf Server und Demo



• ANFORDERUNGSANALYSE



Anforderungsanalyse

Funktionale Anforderungen:

- Entwicklung eine Web-Applikation
 - SaaS: Azure Cognitive Services Translator
 - PaaS: beliebige Datenbank
 - **→ Keine unnötigen / doppelten Anfragen an Translate-API → Entnahme aus DB**
- Deployment der Web-App in Azure Infrastruktur
 - IaaS: Infrastruktur mit Terraform
 - Deployment über Ansible



Nicht-Funktionale Anforderungen:

- Infrastruktur as Code Ansatz
- GitHub Repository / (Docker-)Container-Image
- Keine Secrets im Code / Auslagern von Credentials

• BESCHREIBUNG DER ENTWICKLUNGSUMGEBUNG UND UMSETZUNG

Beschreibung der Entwicklungsumgebung und Umsetzung

Entwicklung Web-Anwendung

- NodeJS als Docker-Container

Automatisierte Infrastruktur-Steuerung:

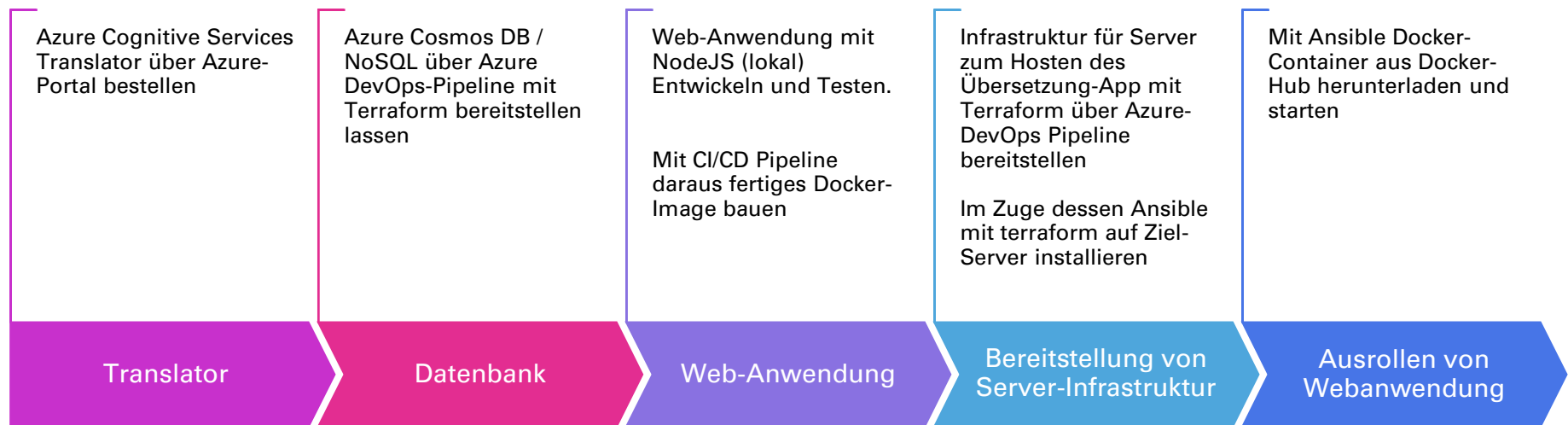
- Azure DevOps Server
- Azure Repo
- Bereitstellung von Infrastruktur mit Terraform Pipelines
 - Azure Cosmos DB
 - Server für Hosting der Webanwendung
- CI/CD Pipeline
 - zur Bildung der Anwendung als Docker-Container
 - Stellt Docker-Container in privates Docker-Hub zur Verfügung

Manuelle Prozesse:

- Erstellen von Storage-Account (terraform-Statefile)
- Azure Cognitive Services Translator besorgen
- Starten der Docker-Container über Ansible auf Server
- API-Keys/DB-Zugangsdaten als Secrets in Azure speichern



Ablaufplan



API-Key und Azure Cosmos DB
Zugangsdaten als Azure Secret speichern

- - + • **TRANSLATOR API UND** - **DATENBANK**

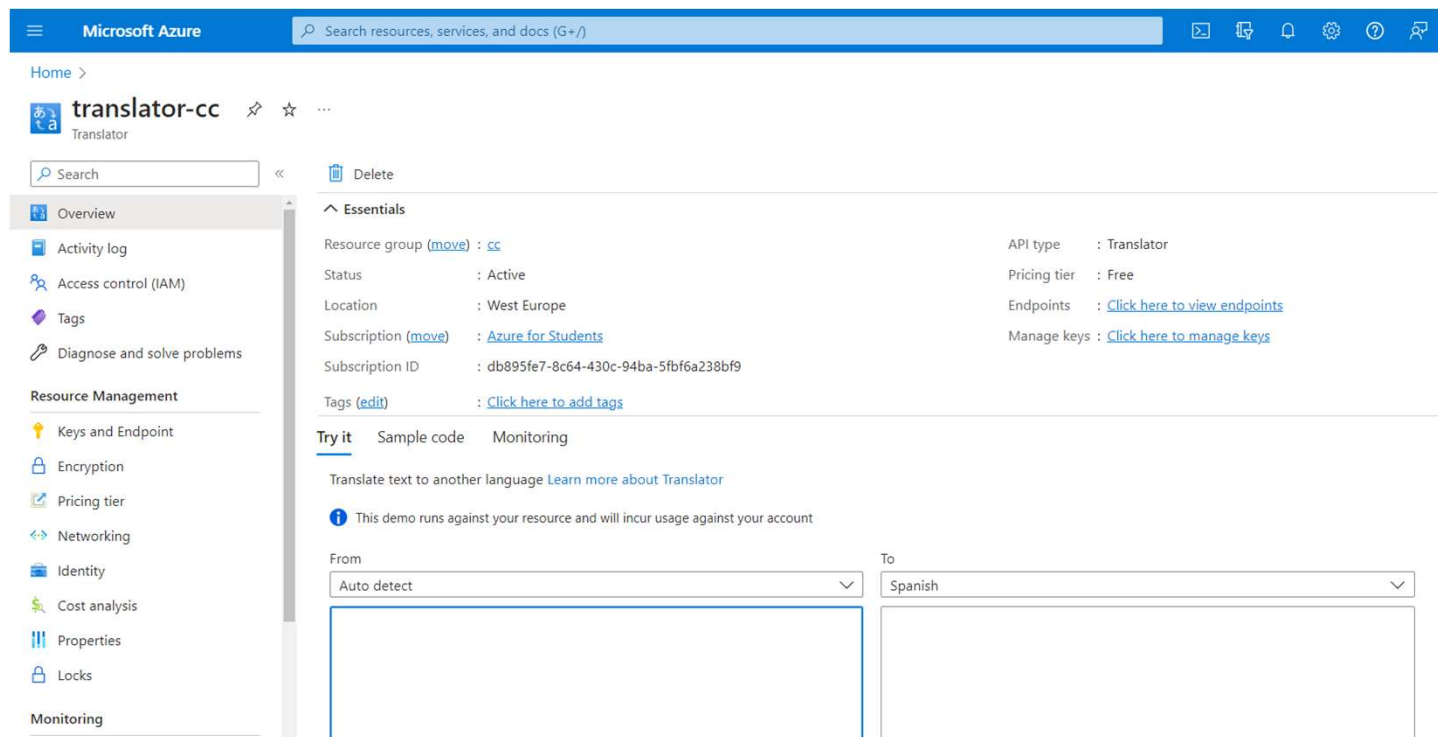
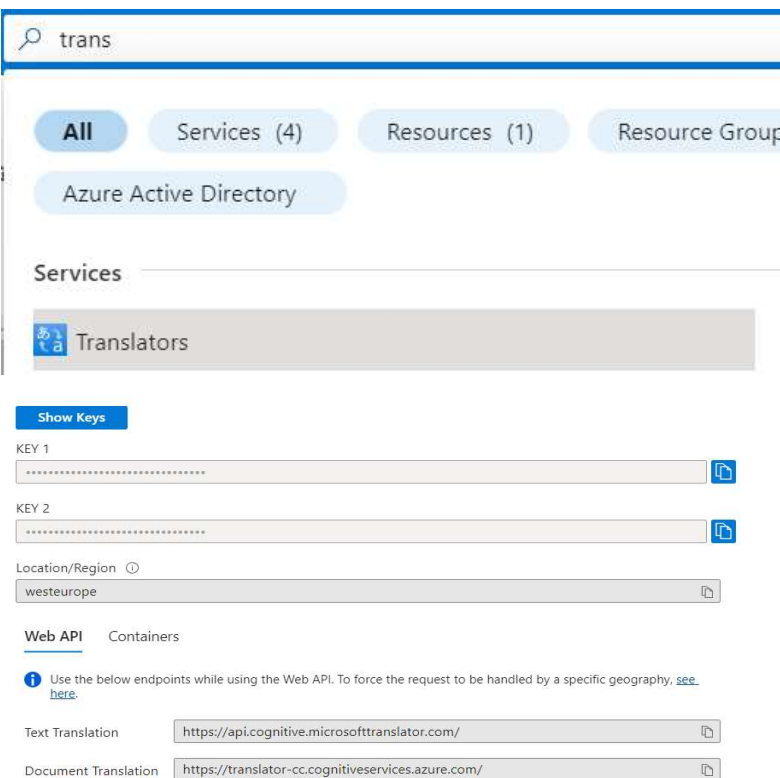


Translator API

Translator

Azure Cognitive Services
Translator über Azure-
Portal bestellen

- Einfache Bereitstellung über Azure Portal Azure Cognitive Services Translator
- API und passender API-Key wird erstellt

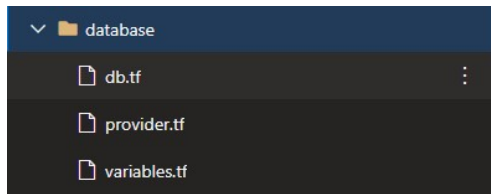


Azure Cosmos DB

Translator

Datenbank

- Terraform Konfiguration



variables.tf

Contents History Compare Blame

```
1 # Define variables
2 variable "resource_group_name" {
3   type = string
4   description = "The name of the backend storage account resource group"
5   default = "cc-tf-db"
6 }
7
8
9
10 variable "name" {
11   type = string
12   default = "inf20068db"
13 }
14
15 variable "location" {
16   type = string
17   default = "westeurope"
18 }
19
20 variable "location_name" {
21   type = string
22   default = "West Europe"
23 }
24
25 variable "default_experience" {
26   type = string
27   default = "Azure Cosmos DB for MongoDB API"
28 }
29
30 variable "is_zone_redundant" {
31   type = bool
32   default = false
33 }
```

db.tf

Contents History Compare Blame

```
1 resource "azurerm_resource_group" "example" {
2   # Name der Resource Group
3   name = var.resource_group_name
4   # Standort der Resource Group
5   location = var.location
6 }
7
8 # Define resource block for Cosmos DB account
9 resource "azurerm_cosmosdb_account" "example" {
10   name = var.name
11   resource_group_name = var.resource_group_name
12   location = var.location
13   offer_type = "Standard"
14   kind = "MongoDB"
15   is_virtual_network_filter_enabled = false
16   enable_free_tier = true
17   capacity {
18     total_throughput_limit = 1000
19   }
20
21   consistency_policy {
22     consistency_level = "Session"
23   }
24
25   geo_location {
26     location = "${var.location_name}"
27     failover_priority = 0
28   }
29
30   capabilities {
31     name = "EnableMongo"
32   }
33
34   capabilities {
35     name = "DisableRateLimitingResponses"
36   }
37
38   capabilities {
39     name = "EnableServerless"
40   }
41
42   tags = {
43     defaultExperience = var.default_experience
44     "hidden-cosmos-mmspecial" = ""
45   }
46
47   enable_automatic_failover = var.is_zone_redundant
48 }
49
50
```

Azure Cosmos DB /
NoSQL über Azure
DevOps-Pipeline mit
Terraform bereitstellen
lassen



Azure Cosmos DB

Translator

Datenbank

Azure Cosmos DB /
NoSQL über Azure
DevOps-Pipeline mit
Terraform bereitstellen
lassen

- Azure Cosmos DB für MongoDB über Terraform Pipeline erstellt

- Pipeline mit 2 Jobs

- 1. Job Validierung der Terraform-Skripte

```
← Create Cosmos-DB with terraform

main cloud-computing-2 / pipelines/database-terraform-pipelines.yml

6 trigger: none
7
8 pool:
9   name: default
10
11 variables:
12   resource_group_name: 'cc'
13   storage_account_name: 'inf20068ccstorage'
14   container_name: 'tfstate-db'
15   file_key: 'terraform.tfstate'
16
17 jobs:
18   - job: Terraform_validate_Infrastructure
19     displayName: Validate Infrastructure
20     steps:
21       - task: TerraformInstaller@1
22         displayName: Terraform install
23         inputs:
24           terraformVersion: 'latest'
25       - task: TerraformTaskV4@4
26         displayName: 'init'
27         inputs:
28           provider: 'azurerm'
29           command: 'init'
30           backendServiceArm: 'Azure-cc-Connection'
31           backendAzureRmResourceGroupName: '${resource_group_name}'
32           backendAzureRmStorageAccountName: '${storage_account_name}'
33           backendAzureRmContainerName: '${container_name}'
34           backendAzureRmKey: '${file_key}'
35           workingDirectory: '${System.DefaultWorkingDirectory}/database'
36
37       - task: TerraformTaskV4@4
38         displayName: 'validate'
39         inputs:
40           provider: 'azurerm'
41           command: 'validate'
42           workingDirectory: '${System.DefaultWorkingDirectory}/database'
43           continueOnError: false
```

Azure Cosmos DB

Translator

Datenbank

Azure Cosmos DB /
NoSQL über Azure
DevOps-Pipeline mit
Terraform bereitstellen
lassen

- Azure Cosmos DB für MongoDB über Terraform Pipeline erstellt

- Pipeline mit 2 Jobs

- 2. Job Terraform-Skripte ausführen

```
← Create Cosmos-DB with terraform

main cloud-computing-2 / pipelines/database-terraform-pipelines.yml

47 -- job: Terraform_apply_Infrastructure
48 -- displayName: Apply Infrastructure
49 -- dependsOn: Terraform_validate_Infrastructure
50 -- steps:
51 --   - task: DownloadSecureFile@1
52 --     inputs:
53 --       secureFile: 'temp'
54 --     continueOnError: false
55 --   - task: TerraformInstaller@1
56 --     displayName: Terraform install
57 --     inputs:
58 --       terraformVersion: 'latest'
59 --   - task: TerraformTaskV4@4
60 --     displayName: 'init'
61 --     inputs:
62 --       provider: 'azurerm'
63 --       command: 'init'
64 --       backendServiceArm: 'Azure-cc-Connection'
65 --       backendAzureRmResourceGroupName: '${(resource_group_name)}'
66 --       backendAzureRmStorageAccountName: '${(storage_account_name)}'
67 --       backendAzureRmContainerName: '${(container_name)}'
68 --       backendAzureRmKey: '${(file_key)}'
69 --       workingDirectory: '${(System.DefaultWorkingDirectory)}/database'
70 --   - task: TerraformTaskV4@4
71 --     displayName: plan
72 --     inputs:
73 --       provider: 'azurerm'
74 --       command: 'plan'
75 --       environmentServiceNameAzureRM: 'Azure-cc-Connection'
76 --       workingDirectory: '${(System.DefaultWorkingDirectory)}/database'
77 --       continueOnError: false
78 --   - task: TerraformTaskV4@4
79 --     displayName: apply
80 --     inputs:
81 --       provider: 'azurerm'
82 --       command: 'apply'
83 --       environmentServiceNameAzureRM: 'Azure-cc-Connection'
84 --       workingDirectory: '${(System.DefaultWorkingDirectory)}/database'
85 --   - task: TerraformTaskV4@4
86 --     displayName: destroy
87 --     inputs:
88 --       provider: 'azurerm'
89 --       command: 'destroy'
90 --       environmentServiceNameAzureRM: 'Azure-cc-Connection'
91 --       workingDirectory: '${(System.DefaultWorkingDirectory)}/database'
```

• ⁺ ENTWICKLUNG DER WEB-ANWENDUNG



Web-Anwendung

Translator

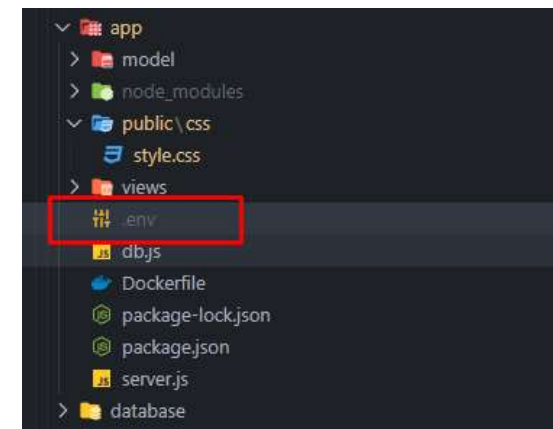
Datenbank

Web-Anwendung

Web-Anwendung mit NodeJS (Lokal Entwickeln und Testen.

Mit CI/CD Pipeline daraus fertiges Docker-Image bauen

- Web-Anwendung für die Übersetzung mit Azure Cognitive Services Translator mit NodeJS lokal entwickeln.
- Secrets wie API-Key und DB-Zugangsdaten in .env auslagern.
- Diese Datei nicht in GitHub/Azure Repos pushen



```
const translatorKey = process.env.TRANSLATOR_KEY;
// Translate Function
async function translateText(text, endpoint) {
  const response = await axios.post(endpoint, [{ 'Text': text }], {
    headers: {
      'Content-Type': 'application/json',
      'Ocp-Apim-Subscription-Key': translatorKey,
      'Ocp-Apim-Subscription-Region': 'westeurope'
    }
  });
  return response.data[0].translations[0].text;
}
```

```
const crypto = require('crypto');
const db = require('./model/text.js');
const mongooseur = 'mongodb://' + process.env.MONGO_USER + ':' + process.env.MONGO_KEY + '@' + process.env.MONGO_URL;

mongoose
  .connect(mongooseur, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  })
  .then(() => console.log("Database connected! \n "))
  .catch(err => console.log(err));
```

Web-Anwendung

Translator

Datenbank

Web-Anwendung

Web-Anwendung mit NodeJS (Lokal Entwickeln und Testen.

Mit CI/CD Pipeline daraus fertiges Docker-Image bauen

- Web-Anwendung mit CI/CD Pipeline als Docker-Container Bereitstellen
- Die Secrets aus der .env Datei müssen in Azure importiert werden

+

•

Home > secrets

secretres | Secrets ☆ ...

Key vault

Search

Generate/Import Refresh Restore Backup View sample code Manage deleted secrets

Name	Type	Status
MONGOKEY		✓ Enabled
MONGOURL		✓ Enabled
MONGOUSER		✓ Enabled
TRANSLATORKEY		✓ Enabled

Overview

Activity log

Access control (IAM)

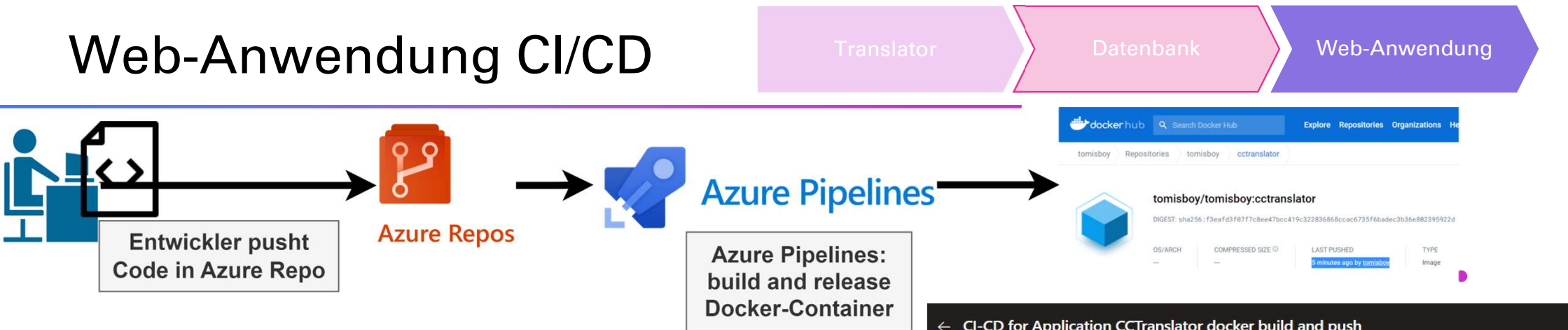
Tags

Diagnose and solve problems

Access policies

Events

Web-Anwendung CI/CD



- Bei Erstellung des Containers werden die Secrets aus Azure in die .env Datei geschrieben
- Anschließend Pushen in privates Docker-Hub repo

```
← CI-CD for Application CCTranslator docker build and push
main cloud-computing-2 / pipelines/build-app-pipeline.yml
15 pool:
16   name: default
17
18 jobs:
19   - job: BuildAndConfigure
20     displayName: Build and Configure translator APP
21     steps:
22
23       Settings
24       - task: AzureKeyVault@2
25         inputs:
26           azureSubscription: 'Azure-cc-Connection'
27           KeyVaultName: 'secrets'
28           SecretsFilter: '**'
29           RunAsPreJob: true
30
31       Settings
32       - task: CmdLine@2
33         displayName: Write Translator API-Key in .env
34         inputs:
35           script: 'touch app/.env && echo $(TRANSLATORKEY) > app/.env'
36
37       Settings
38       - task: CmdLine@2
39         displayName: Write Mongo-URL in .env
40         inputs:
41           script: 'echo $(MONGOURL) >> app/.env && echo $(MONGOUSER) >> app/.env && echo $(MONGOKEY) >> app/.env'
42
43       Settings
44       - task: Docker@2
45         inputs:
46           containerRegistry: 'docker'
47           command: 'login'
48
49       Settings
50       - task: Docker@2
51         inputs:
52           containerRegistry: 'docker'
53           repository: 'tomisboy/tomisboy'
54           command: 'buildAndPush'
55           Dockerfile: '**/Dockerfile'
56           tags: 'cctranslator'
```


• ⁺ • **BEREITSTELLUNG VON SERVER- INFRASTRUKTUR IN AZURE**



Server-Infrastruktur

Translator

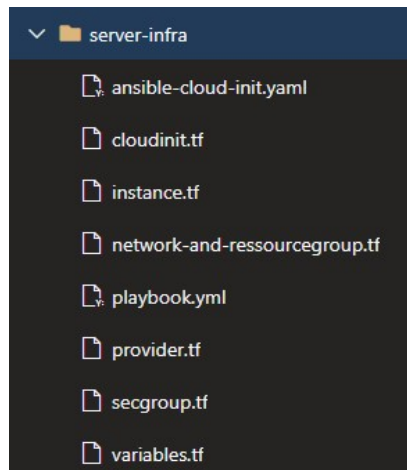
Datenbank

Web-Anwendung

Bereitstellung von
Server-Infrastruktur

Infrastruktur für Server
zum Hosten des
Übersetzung-App mit
terraform über Azure-
DevOps Pipeline
bereitstellen

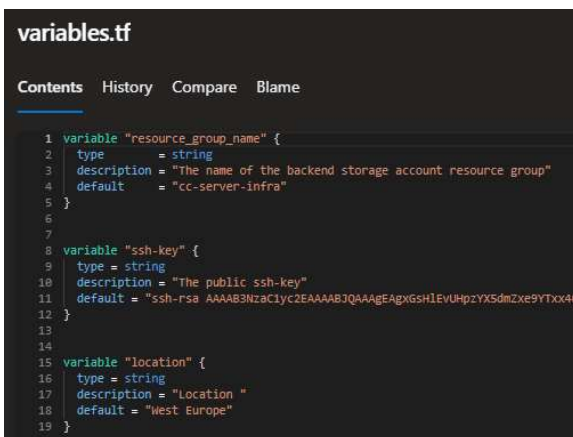
- Terraform
Konfiguration



- Aufteilung der Schritte in einzelne Terraform Skripte
- network-and-ressourcegroup.tf
 - Erstellt Netzwerk und Ressourcen-Gruppe für Infrastruktur
- Secgroup.tf
 - Erstellt Firewall / Portöffnung auf Port 5000 für NodeJS Anwendung
- Instance.tf
 - Erstellt die eigentliche (Linux)-Virtuelle Maschine
- cloudinit.tf und ansible-cloud-init.yaml
 - Führt Cloud-init Script aus welches Ansible und Docker auf Server installiert

+

•



Server-Infrastruktur

Translator

Datenbank

Web-Anwendung

Bereitstellung von
Server-Infrastruktur

- Pipeline mit 2 Jobs
- 1. Job Validierung der Terraform-Skripte

Infrastruktur für Server
zum Hosten des
Übersetzung-App mit
terraform über Azure-
DevOps Pipeline
bereitstellen

+

•

← Create Server-Infrastructure with terraform

main

cloud-computing-2 / pipelines/terraform-pipelines.yml

```
8 pool:
9   name: default
10
11 variables:
12   resource_group_name: 'cc'
13   storage_account_name: 'inf20068ccstorage'
14   container_name: 'tfstate'
15   file_key: 'terraform.tfstate'
16
17 jobs:
18   - job: Terraform_validate_Infrastructure
19     displayName: Validate Infrastructure
20     steps:
21       - task: DownloadSecureFile@1
22         inputs:
23           secureFile: 'temp'
24           continueOnError: false
25       - task: TerraformInstaller@1
26         displayName: Terraform install
27         inputs:
28           terraformVersion: 'latest'
29       - task: TerraformTaskV4@4
30         displayName: 'init'
31         inputs:
32           provider: 'azurerms'
33           command: 'init'
34           backendServiceArm: 'Azure-cc-Connection'
35           backendAzureRmResourceGroupName: '${(resource_group_name)}'
36           backendAzureRmStorageAccountName: '${(storage_account_name)}'
37           backendAzureRmContainerName: '${(container_name)}'
38           backendAzureRmKey: '${(file_key)}'
39           workingDirectory: '${(System.DefaultWorkingDirectory)}/server-infra'
40
41       - task: TerraformTaskV4@4
42         displayName: 'validate'
43         inputs:
44           provider: 'azurerms'
45           command: 'validate'
46           workingDirectory: '${(System.DefaultWorkingDirectory)}/server-infra'
47           continueOnError: false
```

Server-Infrastruktur

Translator

Datenbank

Web-Anwendung

Bereitstellung von
Server-Infrastruktur

- Pipeline mit 2 Jobs
- 2. Job Apply der Terraform-Skripte

```
← Create Server-Infrastructure with terraform

main cloud-computing-2 / pipelines/terraform-pipelines.yml

51 - job: Terraform_apply_Infrastructure
52   displayName: Apply_Infrastructure
53   dependsOn: Terraform_validate_Infrastructure
54   steps:
55     - task: DownloadSecureFile@1
56       inputs:
57         secureFile: 'temp'
58         continueOnError: false
59     - task: TerraformInstaller@1
60       displayName: Terraform install
61       inputs:
62         terraformVersion: 'latest'
63     - task: TerraformTaskV4@4
64       displayName: 'init'
65       inputs:
66         provider: 'azurerms'
67         command: 'init'
68         backendServiceArm: 'Azure-cc-Connection'
69         backendAzureRmResourceGroupName: '${(resource_group_name)}'
70         backendAzureRmStorageAccountName: '${(storage_account_name)}'
71         backendAzureRmContainerName: '${(container_name)}'
72         backendAzureRmKey: '${(file_key)}'
73         workingDirectory: '${(System.DefaultWorkingDirectory)}/server-infra'
74     - task: TerraformTaskV4@4
75       displayName: plan
76       inputs:
77         provider: 'azurerms'
78         command: 'plan'
79         environmentServiceNameAzureRM: 'Azure-cc-Connection'
80         workingDirectory: '${(System.DefaultWorkingDirectory)}/server-infra'
81         continueOnError: false
82     - task: TerraformTaskV4@4
83       displayName: apply
84       inputs:
85         provider: 'azurerms'
86         command: 'apply'
87         environmentServiceNameAzureRM: 'Azure-cc-Connection'
88         workingDirectory: '${(System.DefaultWorkingDirectory)}/server-infra'
```

Infrastruktur für Server
zum Hosten des
Übersetzung-App mit
terraform über Azure-
DevOps Pipeline
bereitstellen

+

•

Server-Infrastruktur



- Zusammenfassung

Infrastruktur für Server zum Hosten des Übersetzung-App mit terraform über Azure-DevOps Pipeline bereitstellen

Home > Resource groups >

Resource groups

Duale Hochschule Baden-Württemberg Stuttgart (...)

+ Create

Manage view

...

Filter for any field...

Name

↑↓

cc

...

cc-server-infra

...

cc-tf-db

...

NetworkWatcherRG

...

cc-server-infra

Resource group

...

Search

«

+ Create

Manage view

...

Delete resource group

Refresh

Export to CSV

Open query

Assign tags

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Deployments

Security

Policies

Properties

Locks

Cost Management

Cost analysis

Cost alerts (preview)

Budgets

Essentials

Subscription [\(move\)](#)

[Azure for Students](#)

Subscription ID

db895fe7-8c64-430c-94ba-5fbf6a238bf9

Tags [\(edit\)](#)

[Click here to add tags](#)

Deployments

[No deployments](#)

Location

West Europe

Resources

Recommendations (2)

Filter for any field...

Type equals all

Location equals all

Add filter

Showing 1 to 6 of 6 records.

Show hidden types

No grouping

List view

<input type="checkbox"/> Name	Type	Location
<input type="checkbox"/> my-nsg	Network security group	West Europe
<input type="checkbox"/> my-os-disk	Disk	West Europe
<input type="checkbox"/> my-public-ip	Public IP address	West Europe
<input type="checkbox"/> my-virtual-network	Virtual network	West Europe
<input type="checkbox"/> my-vm	Virtual machine	West Europe

- **AUSROLLEN DER WEB-ANWENDUNG AUF SERVER**

+

+

○

•

•

○

DEMO

Ausrollen und Ausführen der Anwendung



- Lokal entwickelte App auf Azure Repos pushen
- → CI/CD Pipeline stellt Anwendung als Docker-Container bereit
- Ansible: Zugangsdaten zu privatem Docker-Hub Repo als Secret in ansible-vault speichern

```
ubuntu@my-vm:~$ ansible-vault create var.yml
[DEPRECATION WARNING]: Ansible will require Python 3.
from ansible-core in version 2.12. Deprecation warnin
New Vault password:
Confirm New Vault password: █
```

The screenshot shows the Docker Hub interface for the repository 'tomisboy/tomisboy:cctranslator'. The repository details include the digest, OS/ARCH (linux/amd64), compressed size (346.85 MB), last pushed time (4 hours ago), and type (Image).

```
ubuntu@my-vm:~$ cat playbook.yml
- name: Start Docker Container with Port 5000
  connection: local
  hosts: localhost
  become: true
  vars_files:
    - var.yml

tasks:
  - name: Log in to Docker registry
    docker_login:
      username: "{{ docker_username }}"
      password: "{{ docker_password }}"

  - name: Delete old Docker Container
    docker_container:
      name: cctranslator
      image: tomisboy/tomisboy:cctranslator
      state: absent

  - name: Pull Docker Image
    docker_image:
      name: tomisboy/tomisboy
      tag: cctranslator
      source: pull
      force_source: yes

  - name: Start Docker Container
    docker_container:
      name: cctranslator
      image: tomisboy/tomisboy:cctranslator
      state: started
      restart_policy: always
      published_ports:
        - "5000:5000"
```


✓ #20230415.2 • go
🏠 CI-CD for Application CCTranslator docker build and push

🕒 This run is being retained as one of 3 recent runs by main (Branch).

Summary

Triggered by Thomas Alpert

Repository and version 📁 cloud-computing-2 📌 main 🔗 35c25999	Time started and elapsed 📅 Today at 18:41 🕒 1m 44s
---	--

Jobs

Name
✓ Build and Configure translator APP

```
ubuntu@my-vm:~$ ansible-playbook playbook.yml --ask-vault-pass
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the controller starting with
from ansible-core in version 2.12. Deprecation warnings can be disabled by setting deprecation_w
Vault password:
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit loc

PLAY [Start Docker Container with Port 5000] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Log in to Docker registry] *****
ok: [localhost]

TASK [Delete old Docker Container] *****
changed: [localhost]

TASK [Pull Docker Image] *****
ok: [localhost]

TASK [Start Docker Container] *****
changed: [localhost]

PLAY RECAP *****
localhost                : ok=5    changed=2    unreachable=0    failed=0    skipped=0    resc

ubuntu@my-vm:~$
```

Demo

- Ansible Playbook starten
- URL Aufrufen

← → ↻ ⚠ Nicht sicher | http://52.174.181.242:5000

Azure Translation und Azure Cosmos DB

Willkommen! Hier beliebigen Text zur Übersetzung eingeben

Zielsprache: Englisch (en) ▼

Übersetzen

Übersetzter Text:

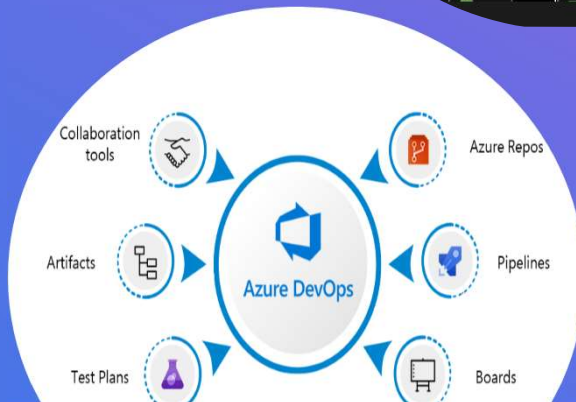
Übersetzter Text in Sprache:



VIELEN DANK!

```
ubuntu@my-vm:~$ . run-ansible.sh
[DEPRECATION WARNING]: Ansible will require Python 3.x to run
from ansible-core in version 2.12. Deprecation warning.
Vault password:
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available
PLAY [Start Docker Container with Port 5000] *****
TASK [Gathering Facts] *****
ok: [localhost]
TASK [Log in to Docker registry] *****
ok: [localhost]
TASK [Delete old Docker Container] *****
changed: [localhost]
TASK [Pull Docker Image] *****
changed: [localhost]
TASK [Start Docker Container] *****
changed: [localhost]
PLAY RECAP *****
localhost : ok=5  changed=3  unreachable=0  failed=0

Server läuft auf Port 5000
Database connected!
```



tomisboy/tomisboy:cctranslator

DIGEST: sha256:8561a41b71ae5a51396bb4b2ede4

OS/ARCH
linux/amd64

COMPRESSED SIZE
346.85 MB

Image Layers

Vulnerabilities

Azure Translation und Azure Cosmos DB

Es war einmal ein kleines Mädchen namens Lilly, das jede Nacht vor dem Schlafengehen ein Buch las. Eines Abends, als sie wieder in ihr Bett gekrochen war, fiel ihr ein Buch aus der Hand und öffnete sich auf einer Seite mit einem Bild von einem wunderschönen Schloss.

Lilly schloss die Augen und stellte sich vor, wie es wäre, in einem solchen Schloss zu leben. Plötzlich fand sie sich in einem wunderschönen Schlossgarten wieder, mit hohen Türmen und Mauern, die den Himmel berührten. Sie erkannte, dass sie tatsächlich im Schloss war!

Sie erkundete das Schloss und traf auf viele freundliche Menschen, die sie einluden, an einem großen Festmahl teilzunehmen. Es gab so viel köstliches Essen und Getränke, und alle waren so fröhlich und gesellig.

Lilly war so glücklich, dass sie beschloss, in diesem Schloss zu bleiben und

Zielsprache:

Übersetzen

Übersetzter Text:

Übersetzter Text in Sprache: en

Once upon a time, there was a little girl named Lilly who read a book every night before bed. One evening, when she had crawled back into her bed, a book fell out of her hand and opened on one page with a picture of a beautiful castle.

Lilly closed her eyes and imagined what it would be like to live in such a castle. Suddenly, she found herself in a beautiful castle garden, with high towers and walls touching the sky. She realized that she was indeed in the castle!

She explored the castle and met many friendly people who invited her to attend a big feast. There was so much delicious food and drinks, and everyone was so cheerful and sociable.

Lilly was so happy that she decided to stay in this castle and never go home