

# Container mit AWS Fargate betreiben

Quelle: <https://bbwin.gitlab.io/m169-aws-fargate>

## Arbeitsumgebung

Für diesen Auftrag benötigen Sie:

- AWS Academy Learner Lab
- Windows mit WSL2 / Ubuntu / Mac
- Docker Desktop mit aktivierter WSL2 Integration

## Arbeitsverzeichnis

Starten Sie ein Terminal (WSL2 mit Ubuntu unter Windows) und erstellen Sie sich in Ihrem Homeverzeichnis ein neues Verzeichnis.

```
mkdir m169-aws-fargate
cd m169-aws-fargate
```

## Docker

Wir arbeiten mit dem alpine-slim Nginx-Image, welches wir mit einem Dockerfile personalisieren. Die Grundlagen haben Sie im Auftrag 10.6\_Docker Images erstellen erarbeitet.

Erstellen Sie die Datei Dockerfile (Gross-/Kleinschreibung beachten, keine Dateiendung):

```
FROM nginx:alpine-slim

COPY index.html /usr/share/nginx/html/index.html
```

Zusätzlich erstellen Sie sich die Datei index.html:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>AWS Fargate Tutorial</title>
</head>

<body bgcolor="PaleGreen">
  <p style="font-family: Helvetica">AWS Fargate Tutorial</p>
</body>

</html>
```

```
manuel@manuel-gaming:~/m169-aws-fargate$ cat Dockerfile
FROM nginx:alpine-slim

COPY index.html /usr/share/nginx/html/index.html
manuel@manuel-gaming:~/m169-aws-fargate$ cat index.html
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>AWS Fargate Tutorial</title>
</head>

<body bgcolor="PaleGreen">
  <p style="font-family: Helvetica">AWS Fargate Tutorial</p>
</body>

</html>
```

Als erster Arbeitsschritt prüfen wir docker build und starten daraus einen Container

```
docker build -t nginx-custom:latest .
```

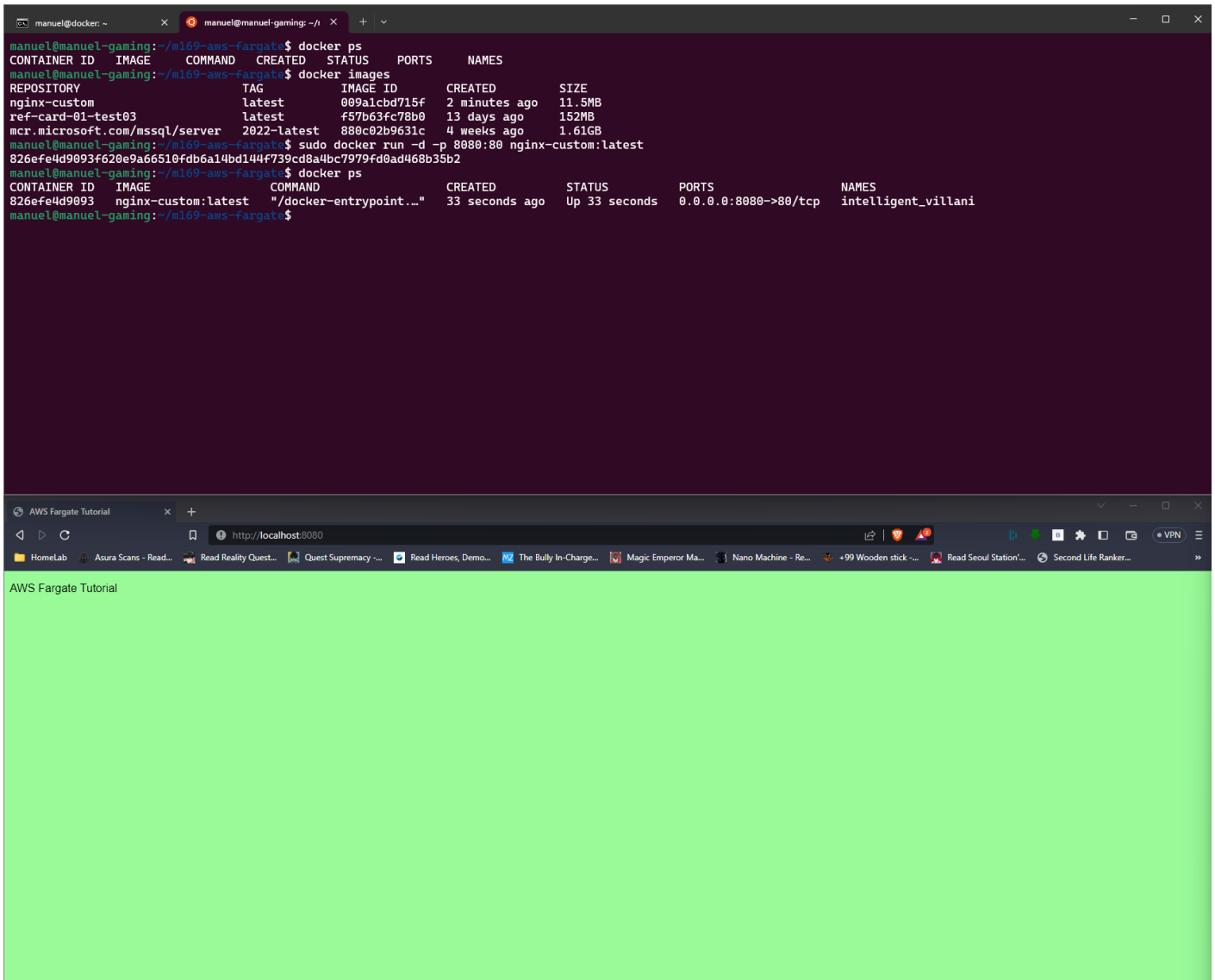
```

manuel@manuel-gaming:~/m169-aws-fargate$ docker build -t nginx-custom:latest .
[+] Building 3.8s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 110B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine-slim              1.9s
=> [auth] library/nginx:pull token for registry-1.docker.io                     0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 249B                                                0.0s
=> [1/2] FROM docker.io/library/nginx:alpine-slim@sha256:84111fd4584e9282a7331d73e031bd6160cff8cb3a43b31c10a42b7 1.6s
=> => resolve docker.io/library/nginx:alpine-slim@sha256:84111fd4584e9282a7331d73e031bd6160cff8cb3a43b31c10a42b7 0.0s
=> => sha256:2ce963c369bc5690378d31c51dc575c7035f6adfcc1e286051b5a5d9a7b0cc5c 1.80MB / 1.80MB 1.0s
=> => sha256:dd0d41fe48dbba258c57acd1febefff44711ad87772c6de8e4546f0a50328ffa0 8.24kB / 8.24kB 0.0s
=> => sha256:f56be85fc22e46face30e2c3de3f7fe7c15f8fd7c4e5add29d7f64b87abdaa09 3.37MB / 3.37MB 0.8s
=> => sha256:59b9d2200e632e457f800814693b3a01adf09a244c38ebe8d3beef5c476c4c55 626B / 626B 0.8s
=> => sha256:84111fd4584e9282a7331d73e031bd6160cff8cb3a43b31c10a42b7b09bb1bec 1.65kB / 1.65kB 0.0s
=> => sha256:11bf3beaa91524d60236685b2c05063c248d22b60247d63b9c35712cf761f7ed 1.57kB / 1.57kB 0.0s
=> => extracting sha256:f56be85fc22e46face30e2c3de3f7fe7c15f8fd7c4e5add29d7f64b87abdaa09 0.2s
=> => sha256:3e1e579c95fece6bbe0cb9c8c2949512a3f8caaf9dbe6219dc6495abb9902040 956B / 956B 1.0s
=> => sha256:547a97583f72a32903ca1357d48fa302e91e8f83ffa18e0c40fd87adb5c06025 773B / 773B 1.0s
=> => sha256:1f21f983520d9a440d410ea62eb0bda61a2b50dd79878071181b56b82efa9ef3 1.40kB / 1.40kB 1.1s
=> => extracting sha256:2ce963c369bc5690378d31c51dc575c7035f6adfcc1e286051b5a5d9a7b0cc5c 0.1s
=> => extracting sha256:59b9d2200e632e457f800814693b3a01adf09a244c38ebe8d3beef5c476c4c55 0.0s
=> => extracting sha256:3e1e579c95fece6bbe0cb9c8c2949512a3f8caaf9dbe6219dc6495abb9902040 0.0s
=> => extracting sha256:547a97583f72a32903ca1357d48fa302e91e8f83ffa18e0c40fd87adb5c06025 0.0s
=> => extracting sha256:1f21f983520d9a440d410ea62eb0bda61a2b50dd79878071181b56b82efa9ef3 0.0s
=> [2/2] COPY index.html /usr/share/nginx/html/index.html                      0.1s
=> exporting to image                                                            0.1s
=> => exporting layers                                                            0.0s
=> => writing image sha256:009a1cbdb715fd5963b6fd756730305391a56eb049ba5c55b3492e8a5f593428c 0.0s
=> => naming to docker.io/library/nginx-custom:latest                          0.0s
manuel@manuel-gaming:~/m169-aws-fargate$

```

Überprüfen Sie Ihr erstelltes Image mit den bereits bekannten Befehlen `images` oder `inspect`, lassen Sie anschliessend Ihr Image lokal mit `docker run` laufen und stellen Sie eine fehlerfreie Ausführung sicher.

```
sudo docker run -d -p 8080:80 nginx-custom:latest
```



# Amazon Elastic Container Registry

Im nächsten Schritt laden wir unser lokal erstelltes Image in die Amazon Elastic Container Registry, kurz ECR.

Dazu benötigen Sie Zugang zu Ihrem AWS Academy Learner Lab. Starten Sie Ihr Lab und öffnen Sie die AWS Management Console.

1. In der AWS Management Console, suchen Sie den Service Elastic Container Registry



2. Wählen Sie Get Started um ein neues Repository zu erstellen

## Ein Repository erstellen

### Erste Schritte

3. Wählen Sie als Visibility settings Private und als Repository name nginx-custom. Alle anderen Einstellungen können Sie wie vorausgewählt belassen.

### Allgemeine Einstellungen

Sichtbarkeitseinstellungen

Info

Wählen Sie die Sichtbarkeitseinstellung für das Repository.

☒ Privat  
Der Zugriff wird durch IAM und Repository-Richtlinienberechtigungen verwaltet.

☐ Öffentlich  
Öffentlich sichtbar und für Image-Pulls zugänglich.

Repository-Name  
Geben Sie einen präzisen Namen an. Ein Entwickler sollte in der Lage sein, die Repository-Inhalte anhand des Namens zu identifizieren.

910977011815.dkr.ecr.us-east-1.amazonaws.com/

12 von maximal 256 Zeichen (mindestens 2). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Unveränderlichkeit von Tags

Info

Aktivieren Sie die Unveränderlichkeit von Tags, um zu verhindern, dass Image-Tags durch nachfolgende Image-Pushs mit demselben Tag überschrieben werden. Deaktivieren Sie die Unveränderlichkeit von Tags, damit Image-Tags überschrieben werden können.

☒ Deaktiviert

Sobald ein Repository erstellt wurde, kann die Sichtbarkeitseinstellung des Repositorys nicht mehr geändert werden.

4. Wählen Sie Ihr soeben erstelltes Repository nginx-custom aus

Private Repositorys (1)

Push-Befehle anzeigen

Löschen

Aktionen ▾

Repository erstelle

Q

Repositorys suchen

<

1

>

<div><input type="checkbox"/></div>	<div>Repository- Name</div>	<div>URI</div>	<div>Erstellt um</div>	<div>Unveränderlichkeit von Tags</div>	<div>Scanfrequenz</div>	<div>Art der Verschlüsselung</div>	<div>Pull-Thro Cache</div>
<div><input type="checkbox"/></div>	<div>nginx-custom</div>	<div><div></div>910977011815.dkr.ecr.us-east-1.amazonaws.com/nginx-custom</div>	<div>03. April 2023, 20:54:31 (UTC+02)</div>	<div>Deaktiviert</div>	<div>Manuell</div>	<div>AES-256</div>	<div>Inaktiv</div>

5. Rechts finden Sie View push commands. Es enthält personalisierte Befehle damit Sie sich authentifizieren und Ihr Image in Ihre Repository pushen zu können.

nginx-custom

Push-Befehle anzeigen

Images (0)

Images suchen

Image-Typ

Scanstatus

Schwach

macOS / Linux

Windows

Stellen Sie sicher, dass Sie die neueste Version von AWS CLI und Docker installiert haben. Weitere Informationen finden Sie unter [Erste Schritte mit Amazon ECR](#).

Gehen Sie wie folgt vor, um sich zu authentifizieren und ein Image in Ihr Repository zu übertragen. Weitere Authentifizierungsmethoden für die Registrierung, einschließlich des Hilfsprogramms für Amazon ECR-Anmeldeinformationen, finden Sie unter [Registry-Authentifizierung](#).

1. Rufen Sie ein Authentifizierungstoken ab und authentifizieren Sie Ihren Docker-Client für Ihr Registry. Benutzen Sie AWS CLI:

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 910977011815.dkr.ecr.us-east-1.amazonaws.com

Hinweis: Wenn bei der Verwendung von AWS CLI ein Fehler auftritt, stellen Sie sicher, dass Sie die neueste Version von AWS CLI und Docker installiert haben.

2. Erstellen Sie Ihr Docker-Image mit dem folgenden Befehl. Informationen zum komplett neuen Erstellen einer Docker-Datei finden Sie in den Anweisungen [hier](#). Sie können diesen Schritt überspringen, wenn Ihr Image bereits erstellt ist:

docker build -t nginx-custom .

3. Nachdem die Erstellung abgeschlossen ist, markieren Sie Ihr Image, damit Sie das Image in dieses Repository übertragen können:

docker tag nginx-custom:latest 910977011815.dkr.ecr.us-east-1.amazonaws.com/nginx-custom:latest

4. Führen Sie den folgenden Befehl aus, um dieses Image in Ihr neu erstelltes AWS Repository zu übertragen:

docker push 910977011815.dkr.ecr.us-east-1.amazonaws.com/nginx-custom:latest

Schließen

[ALLv1-40472](#) > [Modules](#) > [Learner Lab](#) > [Learner Lab](#)

Home

Modules

Discussions

AWS

Used \$0 of \$100

03:52

```

ddd_v1_w_NSan_912034@runweb76970:~$ aws ecr get-login-password --region us-east-1
eyJwYX1kb2FkIjo1dS9kd2FuaHU4dTF5wU05m16cmgwc19TOEF2OC8wak90UH1PN2tZZHjcG1qM2xjV1JYaU1CRS9yaG1jR2thk1NGNFdsS15XSkZkdERFSXBjvXIS5bkxOR1pVbU50akh2YnR5NTVmdmUvRDFxVj1aEU2Yg9WOTZTd2Z2sk0RRK2JJSVWm2Cs0QU1uSFV1TXM2UXJ0ZD1Ka1NKYUFAQnFobEY3UndMbXRnZnhJTkp2RzgrNS9Lc1j1UUEQxVmFOODN2Uux3MeJ2aFJuK3R8VKZ2bktN2XY3VFZP2FpNT0N1i1hTbUJGaEtvCUN3ZnB0cXNYTkE1emNkOEExnK0F1ZDgxZFhwUVA3THU4aTUXOFk4OEJGTG5GaVZBUHdZUFNzYm5yb3Vqbjk3and0OEI2dittY1Z6SGVtNDH4dG8zdGFPRk5xaEd2TE1GV1c4QmVNDT14dHo0bzFLcFpVRVFSN5bW1Rwd11ramF1TkE1T1wJmb29FdvGzlvXNlHpcQkGFFSFFFPdHF1eGVmN0UzVDQ3a0x0RTJnNmFvcXZ2MGSIRWRTY0VFS3BfZjBvd1VwME8rOFB3JRjd3VG45SGNEbH1kHdYQvVFS25yMUccenZ1RXNDMlgvWincxZjUzdzJ5ckRhL3pZS19wQ3EwMTNTb1Q5M2p4w1J5SmocDvmU2d6YHduYkZw1J1LbWRDZX11OGtkZU8NjJnSEIyTTMvUdmUkg1e1VKEv1ueh10cGhDaDNsbnpOMy9sYmtbNS3R5K2RjcXh3dnY0NH2cG1ZUST1RXVURVcxM1FiQUcrMEd1djg4bXRVZlF3S1BhOGxibW9mQkZ1NTJjTCTwcituQU9wRDJ3cm5XbXhmikFTaUVRUvNANaYnA5c0UxvUFQUM1V0ZyVWV6Q1pRKYtMUEQRb2hEiXhsTVRLbHdLRk5laEc5cT1rc21YdDNZdFZTK0VTN3NpdvIraJb0YmNzIzNFmVJenQ1dVWFOERqbD3UMNKS2RmN3ZOERCbFZNeTBtNmVhZGFKb1FOTm1ZbWZ2OHY4eStabGR2VixCQvByZD8LV3pMcTFZOE1rMhoRjg3ehJdTvVxQitndUshdGVBenNBmFovazErYU5JQ89QOLUNISGhOS2w4dU8mTgptb:F6Y2v5zhTEx1bV14RVv1dkc3MdhZmpMUDFKY3FTcGN2dG1qMihabG8xU1BEQnd6Rzc3MUUuUz:F6eJ11e9VmtTF5eV10akpIMGNrhHFaRfH4YmVJS1BXZCt4a5s3eGpDRHY5mzRfUHJrZG1yT0F4bVQyYytPZVh4b3NOSFRS12dvQit1L3h6T310TVmVnG1TQ22YnKE2WmpQCcRMAEYvV1pw3QvdXB4UXVntjJdLenNQ2o2c0xrVhtVvk1QTUdFOEc4d24yYmhzUeUcCIVZFV0t6ag9PWk0wTgxSkg0Z13j1k5Na3ozit10eVdjQikvRU1rQVwT0NaQEVLUj1pdmInlhp5QXNYRHBldE1C40EYymVETTJIZ2R1V1Yzdw1rviUNXdk5wQnkoRDJz5JjvzbZM5dTI5VGg4WfH1VmZkbVfM101y1h3VG9CK0d1aHRQOC91aJgNkiEduInVSTOZHOE9DmnpCK0tMcFNIVTKNlUuUeXfnN0vUOFB1VkrKb1ZpHRSYnJLbvdsRTJNqJv2HEFFTk1RbJRTwInJRGdsS0ZoUk90UvJzeXJTI10MjJtdkpxly93Mv1ZiEkyTkM1LCJkYXRhaZV51jo1QVFFQkFIaHdtMFIhSVNkZVJ05m81bjFHNmVxZiVrVrHVvWfHqZTVVrmN1OVJx0C8xIHdEQUFINHdmQV1KS29aSwH2Y05BUiNHb0c4SUJBREJvQmdrcihrUc5dzBCQndF0hnlUz5VpJQvdvREJBRXVnQkvFREJwSjHhP0UwZH2JchIF0ZwJnSUJFSUE3aFBDZ3RLCTZIR3JOUE15R3dLRudxeG9GL21hc1VDCeZmD16K2045D3FVkrRiUthQakc2a0hTcm9tVEINRkJCdndQm5Bdnh1bHmaz01LCJ2ZXJzaW9uIjo1IiIsInR5cGU1OjEJQVRYR0x0Fm5IsImV4cGlyYXRPb241OjE2ODAA1OTE0IjF9
ddd_v1_w_NSan_912034@runweb76970:~$

```

6. In Ihrem lokalen Terminal geben Sie nun den zweiten Teil des Befehls nach dem | ein.

```
echo "<TOKEN>" | docker login --username AWS --password-stdin 1234EXAMPLE.dkr.ecr.us-east-1.amazonaws.com
```

```

manuel@manuel-gaming:~/m169-aws-fargate$ echo '9kd2FuAHU4dTFSWLU0Sm16cmgwcI9TOEF20C8wak90UHLPN2tZz
G1qM2xjVLJYaU1CRS9yaGLjR2thK1NGNFdsSW5XSkUzdERFSXBjWxI5bKx0RlpVbU50akhZUnJMYnR5NTVwdmUvRDFxVFJlaEU2MG9WOTZTd2ZsK0RRh
VVMZCs0QUlUSFViTXM2UXJ0ZDlKa1NkYUfAQnF0bEY3UndMbXRnZnhJTKp2RzgrNS9LcjlWUEQxVmF00DN2UUX3WEJZaFJuK3R0WFZQVvKZ2bktnZXY3V
FpNT0NiWlHtBtUJGaEtvCUN3ZnB0cXNYTKE1emNk0ExnK0FLDgXZFhwUVA3THU4aTUxOFk40EJGTG5GaVZBUHdZUFNzYm5yb3Vqbjk3and0OEI2dittV
GVtMDN4R7.1VhC87dCF0B15vaEd2TE1GV1c4QmVDNTL4dHo0bzFLcFpVRVFHsSnBwYLRwdLlramF1TKE1TWJmb29FdVgzWXVncHpCWGFFSFFFPdHF1eGVNV
DQ3a0x.1VWME8rOFBJRjd3VG45SGNEbHlkWHDyQWVFS25yMUczenZLRXNDNwgyWncxZjUzdzJ5ckRhL
i9wQ3EwMTNTb1Q5M2p4WLJ5Smo1cUVMUz00nnu4rKZWWLJLbWRDZXlLOGtkZU8vRjVwNjJnSEIyTMMvMudmUkg1eLVkaW1neWtAcGhDaDnsbnpOMy9sY
3R5K2RjcXh3dnY0NHh2cG1ZUST1RXVURVcxM1FiQUcrMed1dia4hXPv7lF3S1BhOGMxbW9mQkZlNTJjTctwcituQU!aUVRU
nA5c0UxVUFqQUM1V0ZyVvN6QlPrKytnUEQrb2hEWXh./dDNZdFZTK0VTN3NpdWlrajb0MmNzMzNFMwVJenQ1dVnFOERqb
kRSUWNKS2RWN3ZLOERCbFZNeTBtNmVhZGFKblFOTmLZbWZ20HY4eStabGR2VWxCQVByZDBLY3pMcTFZOELrMWhoRjg3eWJaTVVxQitndU5hdGVBenNB
zErYU5jQ09Q0OUNISgh0S2w4dURmaWJwTGptbzF6Y2wvSzhTExibVL4RVV1dkc3NmdhZmpNUDFKY3FTcGNZdG1qMWhabG8xU1BEQnd6Rzc3WUUrUzF6e
E9VWTF5eVI0akpIMGNrWHFaRfH4YmVJS1BXZCt4aSs3eGpDRHY5MzRPSEZWUHHrZG1yT0F4bVQyYytPZVh4b3N0SFRSM2dvQitIL3h6T3l0TVNmVGNTQ
kE2WmpQcERNaEYvV1pwb3QvdXB4UXVnMjdLenNQQ2o2c0xrYVhtVk1QTUdFOEc4d24yMmhzeUtCMVZFV0t0NlRMaG9PNk0wTGxsMkg0Z1JjWk5na3ozV
VdjQWkvRU1rQVNWt0Nq0EVLUilpdmMnM4n50XhYRHBWdE1CM0EyYmVETTJIZ2RlV1YzdW1rWUNXdk5wQnkxRDJzSjJvbzM5dTI5VGg4WFh1VmZ0NUFvb
01uY1h3VG9CCK0tMcFNIVTNKwUHuExFn0VUoFb1VkrKb1ZpMHRsYnJLbVdsRTJNOiV2MEFFTk1RbjRtV
GdsS0ZuOk90UJVJzaEyNythT1I0MjJtdkpxVy93MVLZWEkyTkMiLCJKYXRha2V5IjoiQVFFQk.ZWvrWHVvWFhQZ
mNLOVJxOC8xNHdBQUFiNHdmQVLKS29aSWH2Y05BUWNHb0c4d2JRSUJBREJvQmdrcWhraUc5dzBCQndFd0hnnWUpZSVpJQVdVREJBRXVnQkVFREJwSjhHM
HZJcWf0ZWJnSUJFSUE3aFBDZ3RLcTZIR3JOUeI5R3dLRUdXeg9GL21hclVDCeZmTD16K204SDJENFA4VkrWUThQakc2a0hTcm9tVENVRkJCdndQNm5Bc
Hlmaz0iLCJ2ZXJzaW9uIjoIiMiIsInR5cGUiOiJlEQVRBX0tFWSiImV4cGlyYXRpb24iOjE2ODA1OTE0MjF9" | docker login --username AWS -
sword-stdin 910977011815.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded

```

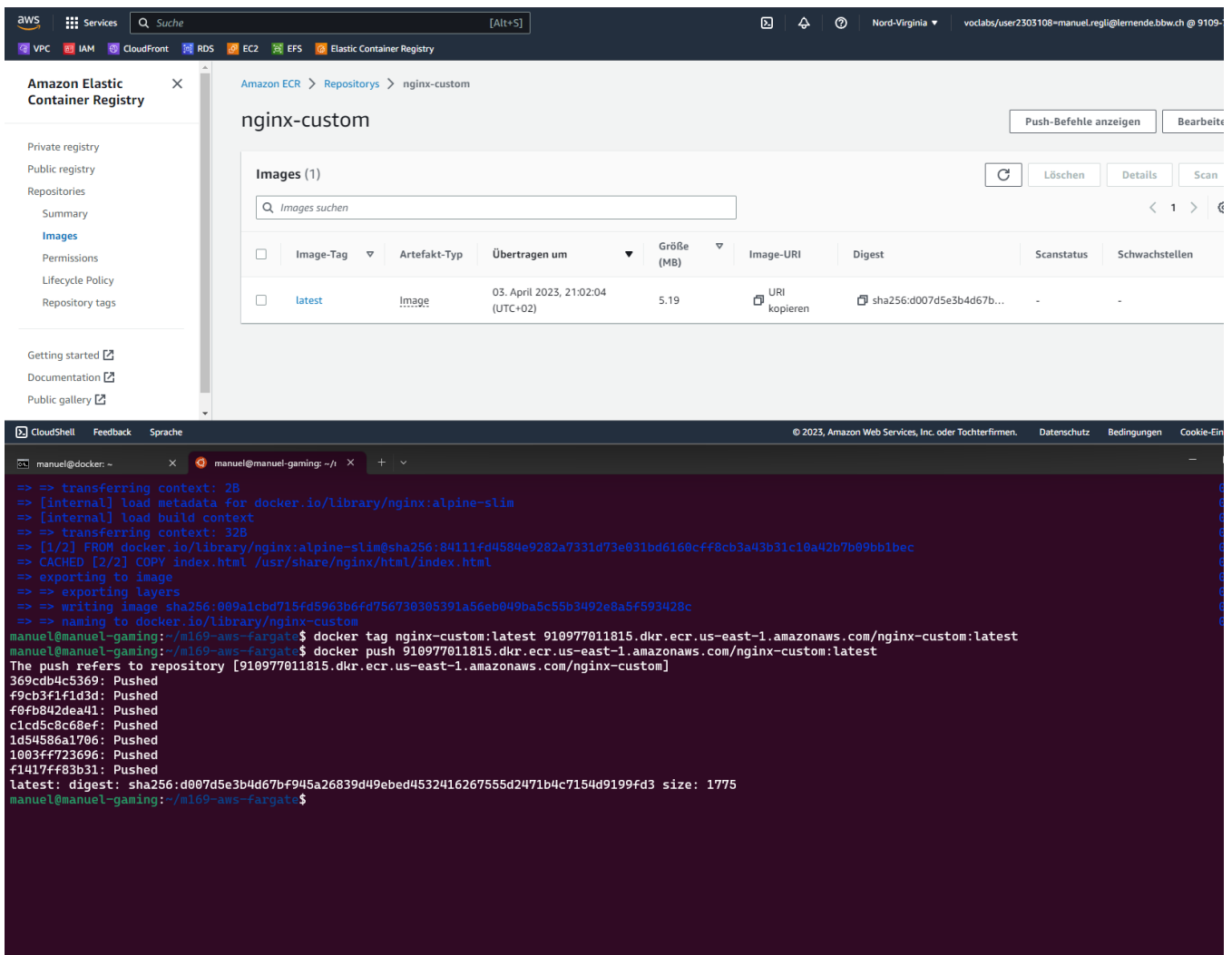
- Führen Sie nun die Schritte 2, 3 und 4 aus der Push commands for nginx-custom Anweisung in Ihrer Elastic Container Registry lokal im Terminal aus.

```
docker build -t nginx-custom .
```

```
docker tag nginx-custom:latest 12345EXAMPLE.dkr.ecr.us-east-1.amazonaws.com/nginx-
custom:latest
```

```
docker push 12345EXAMPLE.dkr.ecr.us-east-1.amazonaws.com/nginx-custom:latest
```





# Amazon Elastic Container Service

## Konzept

Amazon Elastic Container Service (ECS) ist vergleichbar mit Kubernetes, Docker Swarm, and Azure Container Service.

ECS unterstützt zwei unterschiedliche Betriebsmodelle:

- Fargate: Diese Option ist serverless - Sie können Container betreiben, ohne dass Sie Ihre Infrastruktur verwalten müssen. Geeignet für kleine Applikationen, Applikationen mit kurzzeitig hohen Lasten oder schnell wechselnde Lasten
- EC2: Sie konfigurieren EC2-Instanzen (Virtuelle Maschinen) in Ihrem Cluster und stellen Sie sie bereit, um Ihre Container auszuführen. Geeignet für Applikationen mit konstant hoher CPU und Speichernutzung, Preisoptimierung oder Applikationen mit hohem Speicherbedarf.



# Task Definition

Die Task Definition ist eine Vorlage, die beschreibt, welche Docker-Container ausgeführt werden sollen und Ihre Anwendung darstellt. In unserem Beispiel wären es ein nginx-custom Container. Es beschreibt die zu verwendenden Images, die nötige CPU Rechenleistung, Speicher, die Umgebungsvariablen, die freizugebenden Ports und die Interaktion der Container.

## Task

Aus einer Task Definition können Instanzen gestartet werden, welche die Container gemäss dieser Konfiguration beinhaltet. Aus einer Task Definition können beliebig viele identische Tasks erstellt werden.

## Service

Definiert die minimale und maximale Anzahl von Tasks einer Task Definition, die zu einem bestimmten Zeitpunkt ausgeführt werden, sowie die automatische Skalierung und das Loadbalancing.

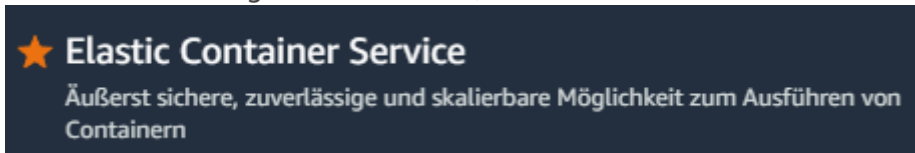
Falls die CPU durch den einzigen laufenden Task ausgelastet ist, kann der Service automatisch zusätzliche Tasks hinzufügen. Es erlaubt zudem die maximale Anzahl der Tasks zu begrenzen, die ausgeführt werden können. Dies hilft, die Kosten von AWS unter Kontrolle zu halten

## Cluster

Der Service muss seine Tasks nun irgendwo ausführen können, damit sie zugänglich sind. Er muss einem Cluster zugeordnet werden und der Containerverwaltungsdienst sorgt selbstständig dafür, dass er genügend ECS-Instanzen für Ihren Cluster bereitstellt.

# Umsetzung

1. In der AWS Management Console, suchen Sie den Service Elastic Container Service



2. Erstellen Sie unter Cluster mit Create cluster einen neuen Cluster. Verwenden Sie als Cluster name fargate-cluster

## Cluster-Konfiguration

Clustername

fargate-cluster

Es darf maximal 255 Zeichen lang sein. Die gültigen Zeichen sind Buchstaben (Groß- und Kleinbuchstaben), Zahlen, Bindestriche und Unterstriche.

3. Wählen Sie links im Menü Task definitions und erstellen Sie eine neue Task Definition mittels Create new task definition. Verwenden Sie nginx-custom als Cluster name und als Image URI Ihre persönliche URL zum privaten Repository aus dem Abschnitt Elastic Container Registry (Beispiel: 12345EXAMPLE.dkr.ecr.us-east-1.amazonaws.com/nginx-custom:latest)

Amazon Elastic Container Service > Neue Aufgabendefinition erstellen

Schritt 1

**Aufgabendefinition und Container konfigurieren**

Schritt 2

Konfigurieren von Umgebung, Speicher, Überwachung und Tags

Schritt 3

Überprüfen und erstellen

## Aufgabendefinition und Container konfigurieren

### Aufgabendefinitionskonfiguration

Aufgabendefinitionsfamilie [Info](#)

Geben Sie einen eindeutigen Namen für die Aufgabendefinition an.

nginx-custom

Bis zu 255 Buchstaben (Groß- und Kleinbuchstaben), Ziffern, Bindestriche und Unterstrichungen sind zulässig.

### Container – 1 [Info](#)

Essenzieller Container

Entfernen

#### Containerdetails

Geben Sie einen Namen, ein Container-Image an und legen Sie fest, ob der Container als wesentlich gekennzeichnet werden soll. Jede Aufgabendefinition muss mindestens einen wesentlichen Container haben.

Name

Image-URI

Essenzieller Container

nginx-custom



910977011815.dkr.ecr.us-east-1.amazonaws.com/nginx

Ja

4. Nachdem Sie mittels Next zum Teil Configure environment, storage, monitoring, and tags gelangt sind, stellen Sie sicher, dass Sie unter Environment AWS Fargate (serverless) verwenden, setzen Sie CPU auf .25 vCPU und Memory auf .5 GB.

Amazon Elastic Container Service > Neue Aufgabendefinition erstellen

Schritt 1  
Aufgabendefinition und Container konfigurieren

Schritt 2  
**Konfigurieren von Umgebung, Speicher, Überwachung und Tags**

Schritt 3  
Überprüfen und erstellen

## Konfigurieren von Umgebung, Speicher, Überwachung und Tags

▼ **Umgebung**  
Geben Sie die Infrastrukturanforderungen für die Aufgabendefinition an.

Umgebung [Info](#)  
Geben Sie die Infrastruktur für die Aufgabendefinition an.

Betriebssystem/Architektur [Info](#)

Aufgabengröße [Info](#)  
Geben Sie die CPU- und Arbeitsspeichermenge an, die für Ihre Aufgabe reserviert werden sollen.

CPU  Arbeitsspeicher

► Containergröße - optional [Info](#)

▼ **Aufgabenrollen, Netzwerkmodus- bedingt**

Aufgabenrolle [Info](#)  
Eine Aufgaben-IAM-Rolle ermöglicht Containern in der Aufgabe, API-Anforderungen an AWS-Services zu stellen. Sie können eine Aufgab IAM-Rolle über die [IAM-Konsole](#) erstellen.

Aufgabenausführungsrolle [Info](#)  
Eine IAM-Rolle für die Aufgabenausführung wird vom Container-Agenten verwendet, um AWS-API-Anforderungen in Ihrem Namen zu stellen. Wenn Sie noch keine IAM-Rolle für die Aufgabenausführung erstellt haben, können wir eine für Sie erstellen.

Netzwerkmodus [Info](#)  
The network mode that's used for your tasks. When the AWS Fargate (serverless) launch type is selected, you must use the awsvpc network mode. If you select the Amazon EC2 instance launch type, you can use different network modes in Linux or Windows. On Linux, you can choose between bridge, awsvpc, host, or none. On Windows, you can choose between default or awsvpc.

5. Wählen Sie links im Menü Cluster und erstellen Sie einen neuen Service.

Services | Aufgaben | Infrastruktur | Metriken | Geplante Aufgaben | Tags

Services (0) [Info](#)

	Servicename	Status	ARN	Service...	Bereitstellungen und Aufgaben	Letzte Berei...	Aufgab...
Keine Services Keine anzuzeigenden Services.							
<input type="button" value="Erstellen"/>							

6. Als Compute options wählen Sie Launch type und belassen die restlichen Voreinstellungen.

### ▼ Datenverarbeitungskonfiguration (erweitert)

#### Datenverarbeitungsoptionen [Info](#)

Zum Sicherstellen der Aufgabenverteilung auf Ihre Datenverarbeitungstypen verwenden Sie die entsprechenden Datenverarbeitungsoptionen.

☐ **Strategie des Kapazitätsanbieters**  
Geben Sie eine Startstrategie an, um Ihre Aufgaben auf einen oder mehrere Kapazitätsanbieter zu verteilen.

☒ **Starttyp**  
Starten Sie Aufgaben direkt ohne Verwendung einer Kapazitätsanbieterstrategie.

#### Starttyp [Info](#)

Wählen Sie entweder verwaltete Kapazität (Fargate) oder benutzerdefinierte Kapazität (EC2- oder benutzerverwaltete External-Instances). External-Instances werden mit der ECS-Anywhere-Funktion in Ihrem Cluster registriert.

FARGATE ▼

#### Plattformversion [Info](#)

Geben Sie die Plattformversion an, auf der Ihr Service ausgeführt werden soll.

LATEST ▼

7. Application type muss ein Service sein. Bei Task definition wählen Sie unter Family die zuvor erstellte Task definition nginx-custom mit Revision latest. Desired tasks ist in unserem Fall 1.

## Anwendungstyp [Info](#)

Geben Sie an, welchen Typ von Anwendung Sie ausführen möchten.

### ☒ Service

Starten Sie eine Gruppe von Aufgaben, die einen Datenverarbeitungsvorgang mit langer Ausführungszeit verarbeiten, der angehalten und neu gestartet werden kann, z. B. eine Webanwendung.

### ☐ Aufgabe

Starten Sie eine eigenständige Aufgabe, die ausgeführt und beendet wird, z. B. eine Batchaufgabe.

## Aufgabendefinition

Wählen Sie eine vorhandene Aufgabendefinition aus. Zum Erstellen einer neuen Aufgabendefinition rufen Sie [Aufgabendefinitionen](#)  an.

### ☐ Revision manuell angeben

Geben Sie die Revision manuell ein, anstatt sie aus den 100 neuesten Revisionen für die ausgewählte Aufgabendefinitionsfamilie auszuwählen.

Familie

nginx-custom ▼

Revision

2 (AKTUELL) ▼

## Servicename

Weisen Sie diesem Service einen eindeutigen Namen zu.

fargate-service

## Servicetyp [Info](#)

Specify the service type that the service scheduler will follow.

### ☒ Replica

Platzieren und verwalten Sie eine gewünschte Anzahl von Aufgaben in Ihrem Cluster.

### ☐ Daemon

Platzieren und verwalten Sie eine Kopie Ihrer Aufgabe auf jeder Container-Instance.

## Gewünschte Aufgaben

Geben Sie die Anzahl der zu startenden Aufgaben an.

1

- Erstellen Sie eine neue Security Group mit dem Namen fargate-service und einer Beschreibung. Erlauben Sie Type HTTP und Source Anywhere.

## Sicherheitsgruppe [Info](#)

Wählen Sie eine vorhandene Sicherheitsgruppe aus oder erstellen Sie eine neue Sicherheitsgruppe.

☐ Vorhandene Sicherheitsgruppe verwenden

☒ Neue Sicherheitsgruppe erstellen

### Sicherheitsgruppendetails

Geben Sie die Konfiguration an, die beim Erstellen der neuen Sicherheitsgruppe verwendet werden soll.

#### Name der Sicherheitsgruppe

fargate-service



Der Name der Sicherheitsgruppe kann bis zu 255 Zeichen lang sein. Gültige Zeichen: A–Z, a–z, 0–9, Leerzeichen und die Sonderzeichen.\_-:/()#,@[]+= &amp;; {}! \$\*.

#### Beschreibung der Sicherheitsgruppe

open port 80

Die Beschreibung der Sicherheitsgruppe kann bis zu 255 Zeichen lang sein. Gültige Zeichen: A–Z, a–z, 0–9, Leerzeichen und die Sonderzeichen.\_-:/()#,@[]+= &amp;; {}! \$\*.

### Regeln für eingehenden Datenverkehr für Sicherheitsgruppen

Fügen Sie Ihrer Sicherheitsgruppe eine oder mehrere Regeln für eingehenden Datenverkehr hinzu.

Typ

HTTP



Protokoll

TCP

Portbereich

80

Quelle

Anywhere



Werte

0.0.0.0/0, ::/0

Löschen

Add rule

## Öffentliche IP [Info](#)

Legen Sie fest, ob der Elastic Network-Schnittstelle (ENI) der Aufgabe automatisch eine öffentliche IP zugewiesen werden soll.

☒ Aktiviert

9. Loadbalancer.

## ▼ Lastenausgleich - optional

### Load Balancer-Typ [Info](#)

Konfigurieren Sie einen Load Balancer für die Verteilung des eingehenden Datenverkehrs auf die in Ihrem Service ausgeführten Aufgaben.

Application Load Balancer ▼

### Application Load Balancer

Geben Sie an, ob ein neuer Load Balancer erstellt oder ein vorhandener Load Balancer ausgewählt werden soll.

- ☒ Neuen Load Balancer erstellen
- ☐ Vorhandenen Load Balancer verwenden

### Load Balancer-Name

Weisen Sie dem Load Balancer einen eindeutigen Namen zu.

lb-fargate

### Container für Lastenausgleich auswählen

nginx-custom 80:80 ▼

### Listener [Info](#)

Geben Sie den Port mit Protokoll an, den Load Balancer auf Verbindungsanforderungen überwachen soll.

- ☒ Neuen Listener erstellen
- ☐ Vorhandenen Listener verwenden  
Sie müssen einen vorhandenen Load Balancer auswählen.

Port

80

Protokoll

HTTP ▼

### Zielgruppe [Info](#)

Geben Sie an, ob Sie eine neue Zielgruppe erstellen oder eine vorhandene auswählen möchten, die der Load Balancer verwenden wird, um Anforderungen an die Aufgaben in Ihrem Service weiterzuleiten.

- ☒ Neue Zielgruppe erstellen
- ☐ Vorhandene Zielgruppe verwenden  
Sie müssen einen vorhandenen Load Balancer auswählen.

Name der Zielgruppe

zg-fargate

Protokoll

HTTP ▼

Pfad der Zustandsprüfung [Info](#)

/

Zustandsprüfungsprotokoll

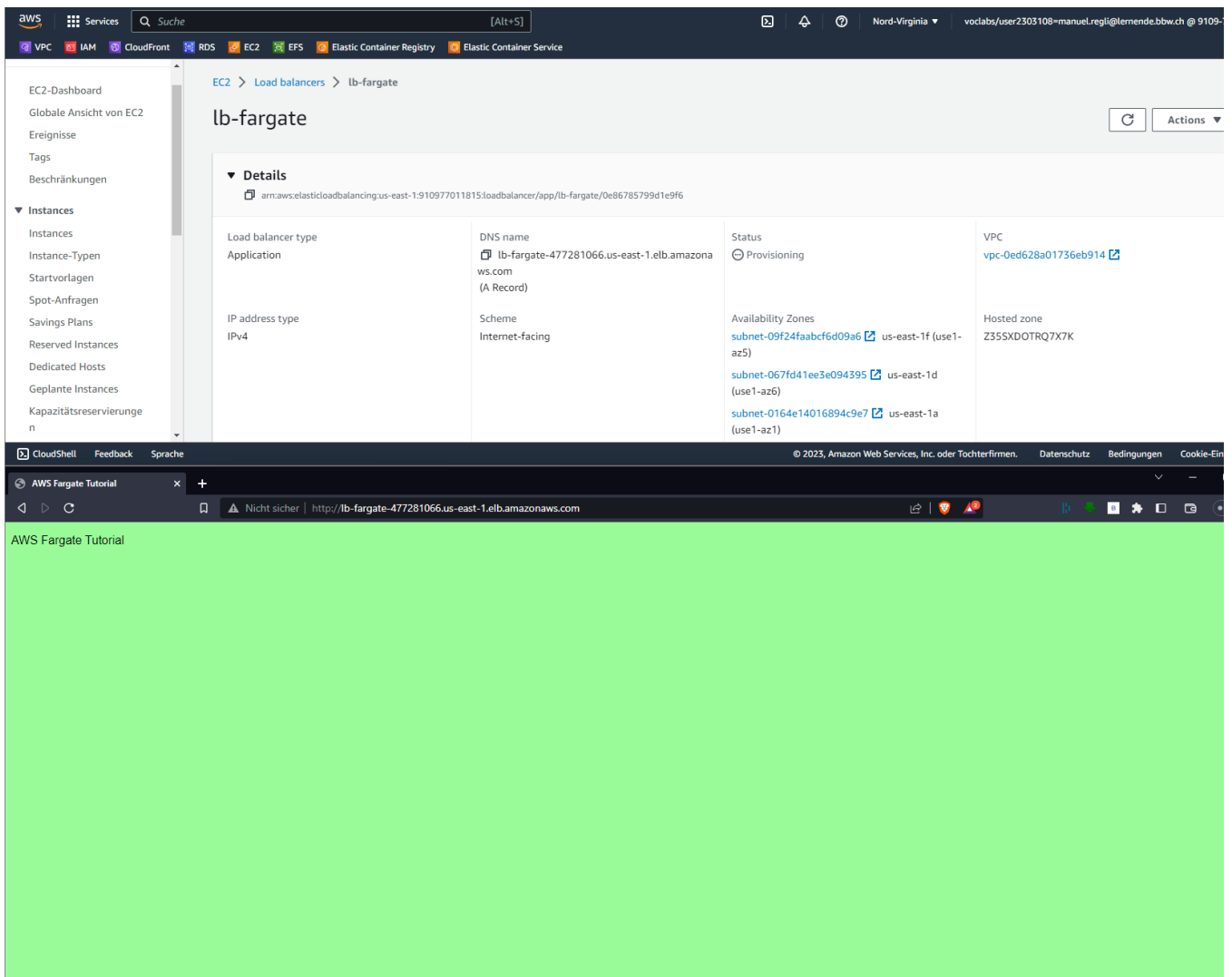
HTTP ▼

Übergangsfrist der Zustandsprüfung [Info](#)

10

Sekunden





# Aktualisiertes Docker Image

Ihr Container läuft nun mit einer öffentlichen IP in AWS. Sie stören sich aber an der Hintergrundfarbe der Webseite und möchten deshalb eine neue Version Ihres nginx-custom erstellen.

Öffnen Sie dazu die Datei index.html in einem Editor (nano index.html oder falls Sie die Visual Studio Code Remote Development Extension aktiviert haben indem Sie das Verzeichnis mit code . in VS Code öffnen). Wählen eine neue Hintergrundfarbe und ersetzen Sie den Wert mit der gewählten Farbe.

Wiederholen Sie die Push commands for nginx-custom aus dem Abschnitt Amazon Elastic Container Registry.

```
docker build -t nginx-custom .  
docker tag nginx-custom:latest 12345EXAMPLE.dkr.ecr.us-east-1.amazonaws.com/nginx-custom:latest  
docker push 12345EXAMPLE.dkr.ecr.us-east-1.amazonaws.com/nginx-custom:latest
```

In Ihrem private Repository in AWS ECR finden Sie nun zwei Images.

Images (2)									
<input type="text" value="Images suchen"/>									
<input type="checkbox"/>	Image-Tag ▾	Artefakt-Typ	Übertragen um ▾	Größe (MB) ▾	Image-URI	Digest	Scanstatus	Schwachstellen	
<input type="checkbox"/>	latest	Image	03. April 2023, 21:32:49 (UTC+02)	5.19	URI kopieren	sha256:9a30b01b8fa27e...	-	-	
<input type="checkbox"/>	<untagged>	Image	03. April 2023, 21:02:04 (UTC+02)	5.19	URI kopieren	sha256:d007d5e3b4d67b...	-	-	

Um den neuen Container im Service zu Deployen wähle den Service aus und gehe dann auf Aktualisieren.

Services (1/1) Info									
<input type="text" value="Services nach Wert filtern"/>									
Alle Starttypen ▾				Alle Servicetypen ▾					
<input checked="" type="checkbox"/>	Servicename ▾	Status ▾	ARN	Service... ▾	Bereitstellungen und Aufgaben ▾		Letzte Berei... ▾	Aufgab... ▾	
<input checked="" type="checkbox"/>	fargate-service	🟢 Aktiv	arn:aws:ecs...	REPLICA	1/1 ausgefü...		🟢 Abgeschlosse...	nginx-cus...	

Danach muss man den hacken "Neue Bereitstellung erzwingen" auswählen und dann auf Aktualisieren. Wenn man dann eine weile wartet sollte die Loadbalancer URL umschalten.

## Bereitstellungskonfiguration

☒ Neue Bereitstellung erzwingen

Tags | Manuels Anleitungen

Aktualisiertes Docker Image - M169 - | Learner Lab

Amazon ECS

M169-AWS-Fargate - kDrive by Inform

AWS Fargate Tutorial

Nicht sicher | http://lb-fargate-477281066.us-east-1.elb.amazonaws.com

AWS Fargate Tutorial

Revision #1

Created 15 December 2023 12:19:10 by Manuel Regli

Updated 15 December 2023 12:27:36 by Manuel Regli