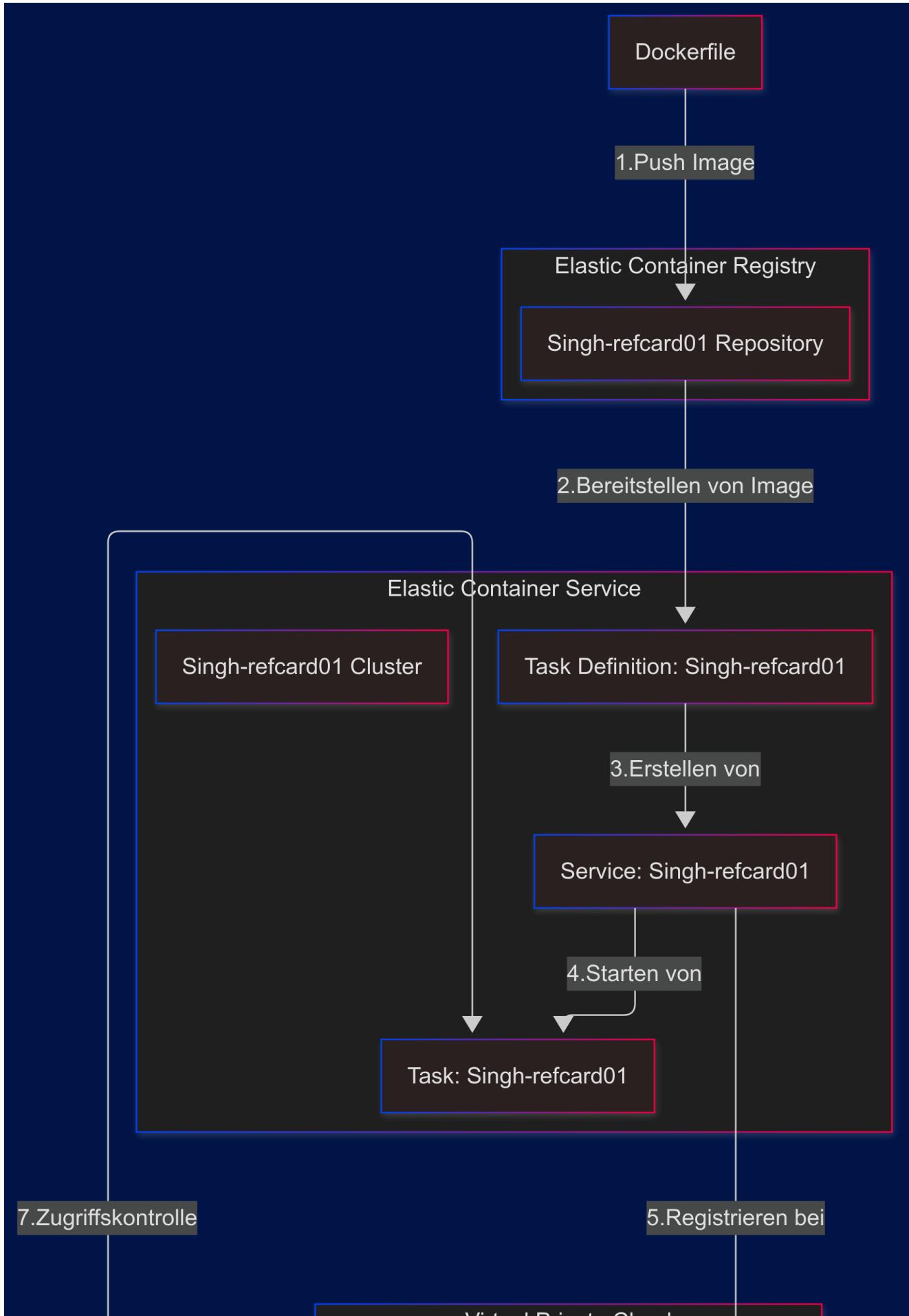
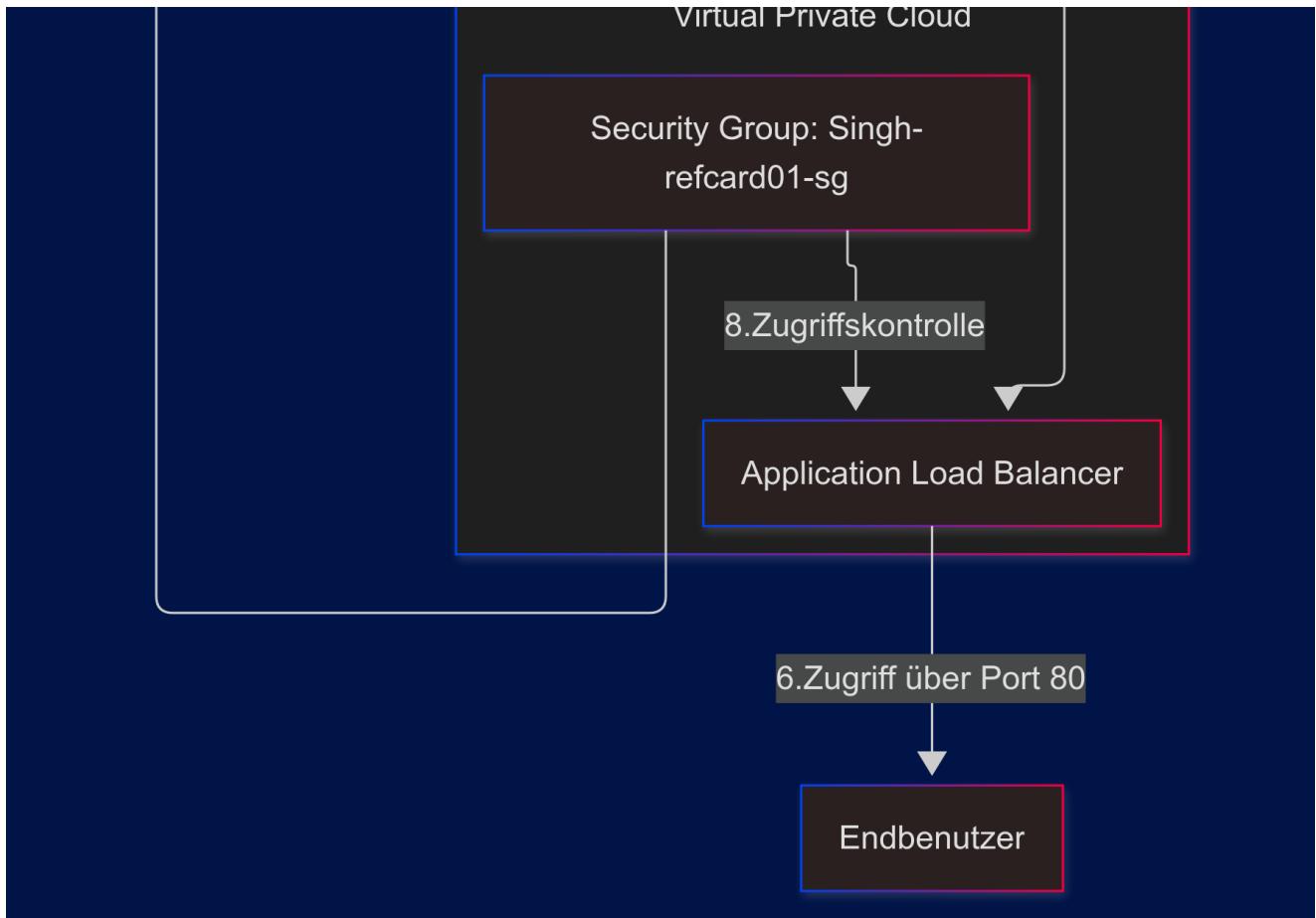


## Theoretischer Abschnitt





## Kurzzusammenfassung des Workflows

1. **Build**: Dockerfile → Image
  2. **Push**: Image → ECR
  3. **Referenzieren**: ECR-Image in Task Definition
  4. **Ausführen**: ECS Service startet Task aus der Task Definition
  5. **Zugreifen**: User stellt Anfrage über den Load Balancer
- 

## Praktischer Abschnitt

### Schritt 1: AWS ECR Repository erstellen

1. Gehen Sie auf die AWS ECR-Konsole:  
[AWS ECR Konsole](#)

Containers

# Amazon Elastic Container Registry

Share and deploy container software, publicly or privately

Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share and deploy your container images and artifacts anywhere.

Create a repository

Create

## How it works



## Pricing (US)

You only pay for the amount of data you store in your public or private repositories and data transferred to the Internet.

[ECR pricing](#)

Enable private connection to ECR private repositories using VPC interface endpoints and reduce costs.

## 2. Erstellen Sie ein neues Repository:

aws Search [Alt+5]

Amazon ECR > Private registry > Repositories > Create a private repository

Create a private repository

General settings

Repository name: 231912049494.dkr.ecr.eu-central-1.amazonaws.com/singh-refcard01

Image tag mutability: Mutable (selected)

Success: Successfully created singh-refcard01

## 3. Hinweis: Es gab Probleme mit der Region. Der Wechsel zu einem US-Server hat das Problem gelöst.

Error creating repository. User: arn:aws:sts::231912049494:assumed-role/voclabs/user3627574=bhupinder.singh@lernende.bbw.ch is not authorized to perform: ecr:CreateRepository on resource: arn:aws:ecr:eu-central-1:231912049494:repository/singh-refcard01 because no identity-based policy allows the ecr:CreateRepository action

Successfully created singh-refcard01

Private repositories (1)

Repository name	URI	Created at	Tag immutability	Encryption type
singh-refcard01	231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01	01 May 2025, 18:54:56 (UTC+02)	Mutable	AES-256

**Repository erfolgreich erstellt. ☑**

## Schritt 2: Docker-Image hochladen

### 1. Image erstellen und hochladen:

Beim Hochladen des Images gab es Probleme mit der Anmeldung (Session Token).

```

An error occurred (UnrecognizedClientException) when calling the GetAuthorizationToken operation: The security token included in the request is invalid.
Error: Cannot perform an interactive login from a non TTY device
PS C:\Users\singh> aws configure
AWS Access Key ID [*****JJ30]: ASIATL7YJBNLNK5LJJ30
AWS Secret Access Key [*****q3p5]: uq7NwAEj4rqK9+xqw2KMLsrza126X8FGg+lVq3p5
Default region name [us-east-1]:
Default output format [table]:
PS C:\Users\singh> aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 231912049494.dkr.ecr.us-east-1.amazonaws.com

An error occurred (UnrecognizedClientException) when calling the GetAuthorizationToken operation: The security token included in the request is invalid.
Error: Cannot perform an interactive login from a non TTY device
PS C:\Users\singh> aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 231912049494.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
PS C:\Users\singh> 

```

## 2. Nach der Behebung des Problems funktionierte der Upload:

```

PS C:\Users\singh\Repo\m347-ref-card-01> aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 231912049494.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
PS C:\Users\singh\Repo\m347-ref-card-01> docker build -t singh-refcard01 .
>
[+] Building 1.0s (13/13) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 601B                                         0.0s
=> [internal] load metadata for docker.io/library/openjdk:17-jdk-slim        0.9s
=> [internal] load metadata for docker.io/library/maven:3.8.5-openjdk-17-slim 0.9s
=> [internal] load .dockerignore                                           0.0s
=> => transferring context: 2B                                         0.0s
=> [build 1/4] FROM docker.io/library/maven:3.8.5-openjdk-17-slim@sha256:502e781d39f0b40fb02eb23f5b7663618b76ba52034da218c64e92f6c5647be 0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 11.91kB                                       0.0s
=> [stage-1 1/3] FROM docker.io/library/openjdk:17-jdk-slim@sha256:aaa3b3cb27e3e520b8f116863d0580c438ed55ecfa0bc126b41f68c3f62f9774 0.0s
=> CACHED [stage-1 2/3] WORKDIR /app                                     0.0s
=> CACHED [build 2/4] WORKDIR /app                                      0.0s
=> CACHED [build 3/4] COPY .                                           0.0s
=> CACHED [build 4/4] RUN mvn clean package -DskipTests                0.0s
=> CACHED [stage-1 3/3] COPY --from=build /app/target/*.jar app.jar    0.0s
=> exporting to image                                                 0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:b907060c872cb02a71d9b0380a06768625d920acddb55cdfb8ddd6bfcbaa1c8a 0.0s
=> => naming to docker.io/library/singh-refcard01                      0.0s

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\singh\Repo\m347-ref-card-01> docker tag singh-refcard01:latest 231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01:latest
PS C:\Users\singh\Repo\m347-ref-card-01> 

```

## Push commands for singh-refcard01

X

macOS / Linux

Windows

Make sure that you have the latest version of the AWS TOOLS for PowerShell and Docker installed. For more information, see [Getting started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS TOOLS for PowerShell:

```
 (Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin 231912049494.dkr.ecr.us-east-1.amazonaws.com
```

Note: if you receive an error using the AWS TOOLS for PowerShell, make sure that you have the latest version of the AWS TOOLS for PowerShell and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch, see the instructions [here](#). You can skip this step if your image has already been built:

```
 docker build -t singh-refcard01 .
```

3. After the build is completed, tag your image so you can push the image to this repository:

```
 docker tag singh-refcard01:latest 231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
 docker push 231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01:latest
```

[Close](#)

### Image erfolgreich hochgeladen.

```
PS C:\Users\singh\Repo\m347-ref-card-01> docker push 231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01:latest
The push refers to repository [231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01]
5b46d6d70699: Pushed
4fcfd69ae98ea: Pushed
6be690267e47: Pushing [=====] 241.1MB/326be690267e47: Pushing 245M6be690267e47: Pushing 246.7M
6be690267e47: Pushing 247.8M6be690267e47: Pushing 250.6M6be690267e47: Pushing 251.7M6be690267e47: Pushing 253.4M6be690267e47: Pushing 254.5M
6be690267e47: Pushing 256.1M6be690267e47: Pushing 257.8M6be690267e47: Pushing 259.5M6be690267e47: Pushing 261.2M6be690267e47: Pushing 262.8M
6be690267e47: Pushing 264.5M6be690267e47: Pushing 266.7M6be690267e47: Pushing 269M6be690267e47: Pushing 271.2M6be690267e47: Pushing 271.7M
6be690267e47: Pushed
B/322.5MB

latest: digest: sha256:9377ae905be4a7bb4b71ff8ec714e918bcd4737674a023e4e04b022f0e9d6b9e size: 1371
```

## Dockerfile

Das folgende Dockerfile wurde für den Multi-Stage-Build verwendet:

```

# 📦 STAGE 1: Baue das Projekt mit Maven
FROM maven:3.8.5-openjdk-17-slim AS build
WORKDIR /app

# Kopiere alle Dateien ins Image
COPY . .

# Baue das Projekt mit Maven
# -DskipTests (ohne Tests)
RUN mvn clean package -DskipTests

# 💡 STAGE 2: Erstelle ein kleines, sauberes Laufzeit-Image
FROM openjdk:17-jdk-slim
WORKDIR /app

# Kopiere das gebaute JAR aus dem vorherigen Schritt
COPY --from=build /app/target/*.jar app.jar

# Öffne Port 8080 für den Container
EXPOSE 8080

# Starte die App
ENTRYPOINT ["java", "-jar", "app.jar"]

```

## Schritt 3: ECS-Cluster erstellen

### 1. Erstellen Sie einen ECS-Cluster:



#### Create cluster Info

An Amazon ECS cluster groups together tasks and services, and allows for shared capacity and common configurations. All of your tasks, services and capacity must belong to a cluster.

**Cluster configuration**

**Cluster name**

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

► Service Connect defaults – optional

Cluster erstellt.

### Hinweis:

Achten Sie darauf, den Cluster anders korrekt zu benennen, um Probleme mit der Repository-Verknüpfung zu vermeiden.

## mehr probleme lol

⌚ There was an error creating cluster singh-refcard01-cluster.  
Resource handler returned message: "Invalid request provided: CreateCluster Invalid Request: Unable to assume the service linked role. Please verify that the ECS service linked role exists. (Service: AmazonECS; Status Code: 400; Error Code: InvalidParameterException; Request ID: ba8a9a31-48b7-437e-b688-a92550ba4302; Proxy: null)" (RequestToken: 4454079d-a447-29b6-052f-06fa02aa68aa, HandlerErrorCode: InvalidRequest)

[View in CloudFormation](#) [Diagnose with Amazon Q](#)

### Infra-ECS-Cluster-singh-refcard01-cluster-347b686a



[Delete](#) [Update stack](#) [Stack actions](#) [Create stack](#)

Stack info [Events](#) Resources Outputs Parameters Template Changesets Git sync

[Table view](#) | [Timeline view](#)

⌚ The resource ECSCluster is in a CREATE\_FAILED state

#### Events (7)

[View root cause](#)



Search events

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-05-01 20:32:49 UTC+0200	<a href="#">Infra-ECS-Cluster-singh-refcard01-cluster-347b686a</a>	<span>✓ DELETE_COMPLETE</span>	-	-
2025-05-01 20:32:48 UTC+0200	ECSCluster	<span>✓ DELETE_COMPLETE</span>	-	-
2025-05-01 20:32:46 UTC+0200	<a href="#">Infra-ECS-Cluster-singh-refcard01-cluster-347b686a</a>	<span>ⓘ DELETE_IN_PROGRESS</span>	-	User Initiated
2025-05-01 20:30:14 UTC+0200	<a href="#">Infra-ECS-Cluster-singh-refcard01-cluster-347b686a</a>	<span>⌚ CREATE_FAILED</span>	-	The following resource(s) failed to create: [ECSCluster].
2025-05-01 20:30:13 UTC+0200	ECSCluster	<span>⌚ CREATE_FAILED</span> <span>Likely root cause</span>	-	Resource handler returned message: "Invalid request provided: CreateCluster Invalid Request: Unable to assume the service linked role. Please verify that the ECS service linked role exists. (Service: AmazonECS; Status Code: 400; Error Code: InvalidParameterException; Request ID: ba8a9a31-48b7-437e-b688-a92550ba4302; Proxy: null)" (RequestToken: 4454079d-a447-29b6-052f-06fa02aa68aa, HandlerErrorCode: InvalidRequest)

### 3 Cluster erfolgreich erstellt:

⌚ Cluster singh-refcard01-cluster1 has been created successfully.

#### Clusters (1) [Info](#)

Search clusters

Cluster	Services	Tasks	Container instances	CloudWatch mo
<a href="#">singh-refcard01-cluster1</a>	0	No tasks running	0 EC2	<span>⌚ Default</span>

Cluster erstellt

⌚ <https://gitlab.com/bbwrl/m347...>

what minimum Requirements. are enough for this app in AWS task. definition CPU RAM. It is. [@https://gitlab.com/bbwrl/m347-ref-card-01](https://gitlab.com/bbwrl/m347-ref-card-01)

⌚ Rinaldo / m347-ref-card-01 · G...

⌚ Thought for 10 seconds



For a Java application like the one in your project `m347-ref-card-01` (from <https://gitlab.com/bbwrl/m347-ref-card-01>), determining the absolute *minimum* CPU and RAM without performance testing can be a bit of an estimate. Java applications, especially those built with frameworks like Spring Boot (which is common for projects with a `pom.xml`), have a certain overhead due to the Java Virtual Machine (JVM) and the framework itself.

However, for a "small Java-App" as you're aiming to deploy on AWS Fargate, a very common and often sufficient starting point is:

- **CPU: 0.25 vCPU** (which is 256 CPU units in Fargate terminology)
- **RAM: 512 MiB (0.5 GB)**

This is generally the smallest memory configuration available for 0.25 vCPU tasks on Fargate and is frequently used as a baseline for simple Java web applications or microservices. Your own initial plan (`CPU: z.B. 256, RAM: 512`) aligns perfectly with this common starting point.

#### Key Considerations:

1. **JVM Heap Size:** A significant portion of the 512 MiB RAM will be allocated to the JVM heap. If your application does a lot of in-memory processing or handles many concurrent requests, it might need more.
2. **Application Complexity:** A very simple "hello world" app will use less than a more complex one with many dependencies, database connections, etc.
3. **Startup Time:** Less RAM can sometimes lead to longer startup times as the JVM and application initialize.
4. **Monitoring is Crucial:** Once deployed, you should monitor your application's CPU and memory utilization in Amazon CloudWatch.
  - If you see high memory utilization consistently (e.g., >80-90%) or OutOfMemoryErrors, you'll need to increase the RAM.
  - If CPU utilization is consistently high, you might need more CPU.

#### Recommendation:

Start with **0.25 vCPU and 512 MiB RAM**. This is a cost-effective initial setting. After deployment, observe its performance under some load. If it runs well, great! If it struggles (e.g., slow responses, errors, high resource usage), you can easily update your Task Definition with more CPU or RAM. Fargate offers various combinations, like 0.25 vCPU / 1GB RAM, 0.5 vCPU / 1GB RAM, etc.

Now that we've discussed the Task Definition resources, are you ready to proceed with the `docker build`, `tag`, and `push` commands? And could you please confirm the name of your ECR repository (is it `singh-refcard01` or `refcard01`)?

## Create new task definition Info

### Task definition configuration

Task definition family Info

Specify a unique task definition family name.

`singh-refcard01-task`

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

## OS, Architecture, Network mode

Network mode is used for tasks and is dependent on the compute type selected.

Operating system/Architecture | [Info](#)

Linux/X86\_64

Network mode | [Info](#)

awsvpc

Task size | [Info](#)

Specify the amount of CPU and memory to reserve for your task.

CPU

.5 vCPU

Memory

.5 GB

## ▼ Task roles - conditional

Task role | [Info](#)

A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the [IAM console](#).

LabRole

Task execution role | [Info](#)

A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.

LabRole

## ▼ Container – 1 [Info](#)

Essential container

[Remove](#)

### Container details

Specify a name, container image and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name

refcard01

Image URI

231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-refcard01:latest

Essential container

Yes

### Private registry | [Info](#)

Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

### Port mappings | [Info](#)

Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port

8080

Protocol

TCP

Port name

refcard01-8080-protocol

App protocol

HTTP

[Remove](#)

[Add port mapping](#)

Task Definition erstellt.

Task definition successfully created

singh-refcard01-task:1 has been successfully created. You can use this task definition to deploy a service or run a task.

[View task definition](#)

X

**singh-refcard01-task:1**

[Deploy](#)

[Actions](#)

[Create new revision](#)

### Overview [Info](#)

ARN

arn:aws:ecs:us-east-1:231912049494:task-definition/singh-refcard01-task:1

Status

ACTIVE

Time created

01 May 2025 at 21:01 (UTC+2:00)

App environment

Fargate

Task role

LabRole

Task execution role

LabRole

Operating system/Architecture

Linux/X86\_64

Network mode

awsvpc

Fault injection

Turned off

## Cluster Service

Service vorbereiten um in der nächsten Aufgabe darauf zuzugreifen

**Services (0) [Info](#)**

[Create](#)

Filter services by value

Filter launch type

[Manage tags](#)

[Update](#)

[Delete service](#)

[Create](#)

Any launch type

Filter service type

Any service type

< 1 >

Service name

ARN

Status

▼

Service type

Created at

Deployments and tasks

▼

Last deployment

Task defi

No services

No services to display.

[Create](#)

## Service details

### Task definition family

Select an existing task definition. To create a new task definition, go to [Task definitions](#).



### Task definition revision Latest

Select the task definition revision from the 100 most recent entries, or enter a revision. Leave the field blank to use the latest revision.



### Service name

Assign a service name that is unique for this cluster.

Up to 255 letters (uppercase and lowercase), numbers, underscores and hyphens are allowed. Service names must be unique within a cluster.

## Environment

AWS Fargate

### Existing cluster

### ▼ Compute configuration (advanced)

#### Compute options | [Info](#)

To ensure task distribution across your compute types, use appropriate compute options.

Capacity provider strategy

Specify a launch strategy to distribute your tasks across one or more capacity providers.

Launch type

Launch tasks directly without the use of a capacity provider strategy.

### Launch type | [Info](#)

Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed). External instances are registered to your cluster using the ECS Anywhere capability.



### Platform version | [Info](#)

Specify the platform version on which to run your service.



## singh-refcard01-cluster1

Last updated 01 May 2025 at 22:13 (UTC+2:00)

[Update cluster](#)[Delete cluster](#)

### Cluster overview

#### ARN

#### Status

Active

#### CloudWatch monitoring

Default

#### Registered container instances

-

### Services

#### Draining

-

#### Status

Active

### Tasks

#### Pending

#### Running

1

[Services](#)[Tasks](#)[Infrastructure](#)[Metrics](#)[Scheduled tasks](#)[Configuration](#)[Tags](#)

### Services (1) [Info](#)

[Manage tags](#)[Update](#)[Delete service](#)[Create](#)[Filter launch type](#)[Filter service type](#) Service name[ARN](#)[Status](#)[Service type](#)[Created at](#)[Deployments and tasks](#)[Last deployment](#)[Task defini.](#) singh-refcard01-task-service-gqntdl5u

Active

REPLICA

5 minutes ago

1/1 tasks running

Completed

[View](#)[singh-refca](#)

## VPC Security groups

### sg-0c340e375348eb53 - default

[Actions](#)

#### Details

##### Security group name

default

##### Security group ID

##### Description

default VPC security group

##### VPC ID

##### Owner

##### Inbound rules count

3 Permission entries

##### Outbound rules count

1 Permission entry

[Inbound rules](#)[Outbound rules](#)[Sharing - new](#)[VPC associations - new](#)[Tags](#)

### Inbound rules (3)

[Manage tags](#)[Edit inbound rules](#)

< 1 >

[Name](#)[▼](#)[Security group rule ID](#)[▼](#)[IP version](#)[▼](#)[Type](#)[▼](#)[Protocol](#)[▼](#)[Port range](#)[▼](#)[Source](#)

-

sgr-06380728c5d1d340e

IPv6

Custom TCP

TCP

8080

::/0

-

sgr-0493c134f1822f794

IPv4

Custom TCP

TCP

8080

0.0.0.0/0

-

sgr-08b1adbcc9783cc25

-

All traffic

All

All

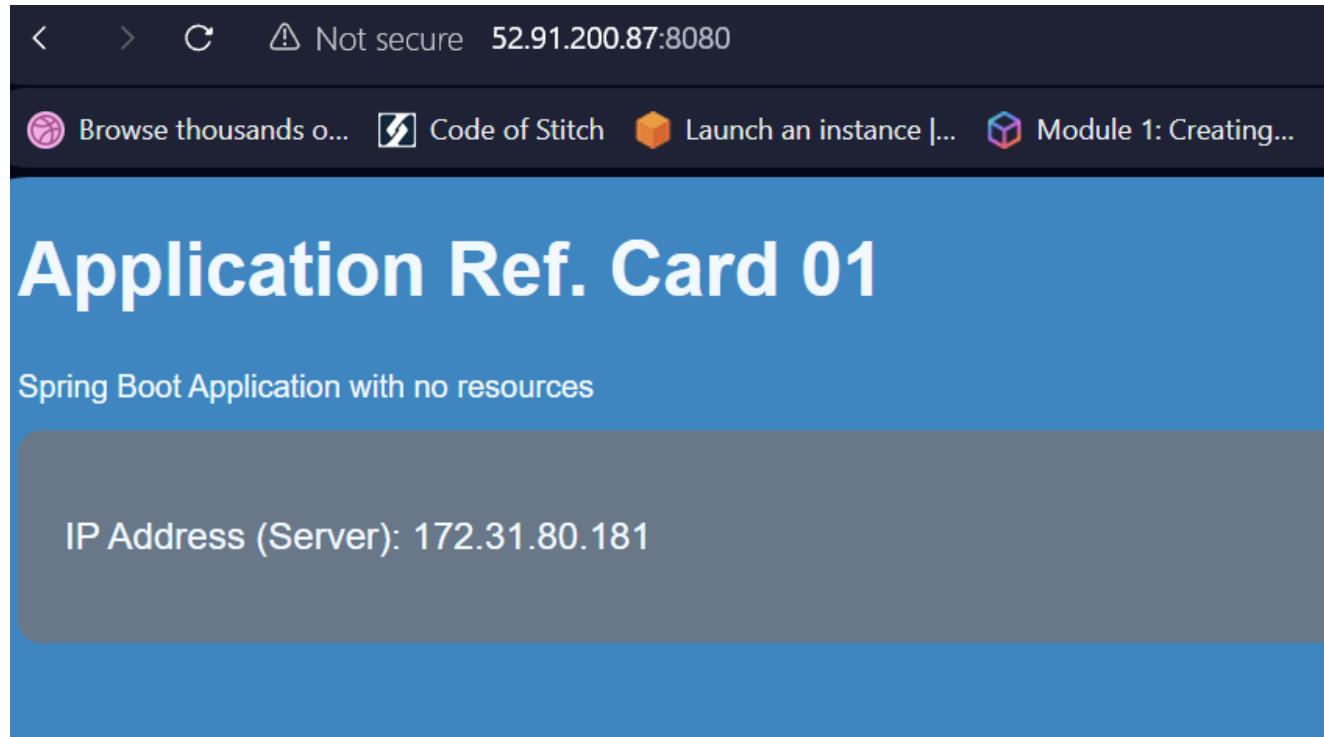
sg-060421

Service testen

## Public IP

 52.91.200.87 | [open address](#) 

Service auf IP öffnen



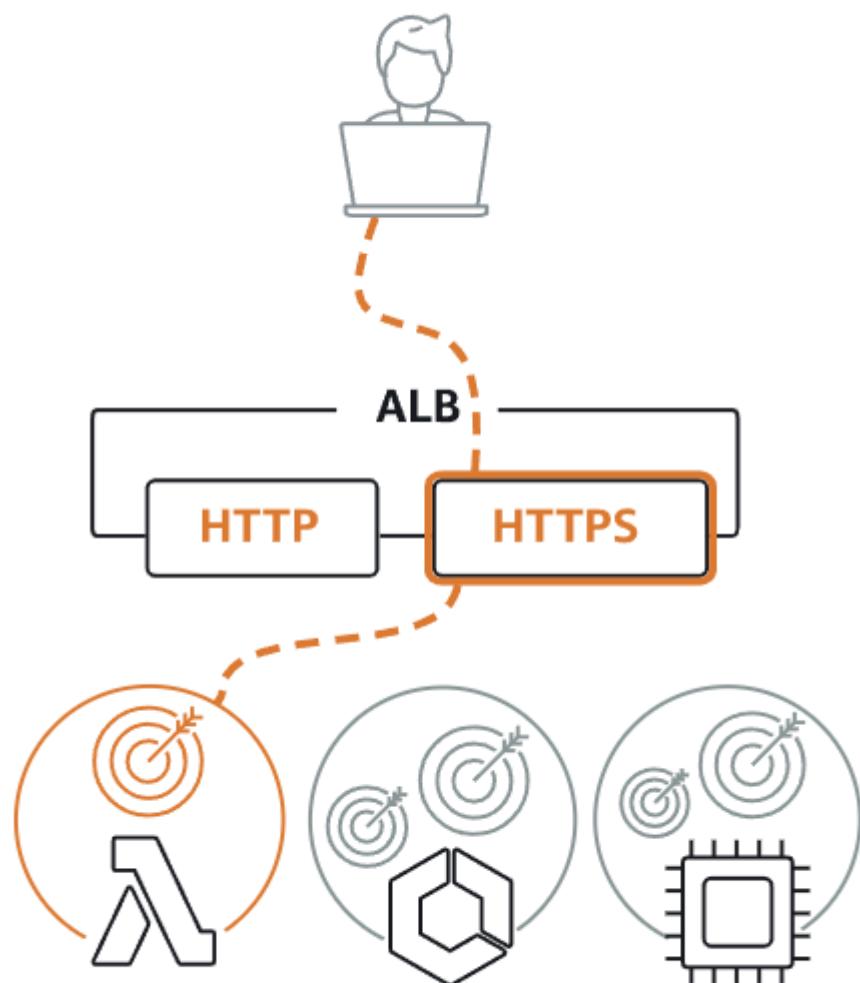
Load-Balancer erstellen

### No load balancers

You don't have any load balancers in us-east-1

[Create load balancer](#)

# Application Load Balancer [Info](#)



Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

[Create](#)

## Basic configuration

### Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

singh-refcard01-lb

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

### Scheme | Info

Scheme can't be changed after the load balancer is created.

#### Internet-facing

- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name resolves to public IPs.
- Requires a public subnet.

#### Internal

- Serves internal traffic.
- Has private IP addresses.
- DNS name resolves to private IPs.
- Compatible with the **IPv4** and **Dualstack** IP address types.

### Load balancer IP address type | Info

Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

#### IPv4

Includes only IPv4 addresses.

## Basic configuration

### Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

singh-refcard01-lb

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

### Scheme | Info

Scheme can't be changed after the load balancer is created.

#### Internet-facing

- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name resolves to public IPs.
- Requires a public subnet.

#### Internal

- Serves internal traffic.
- Has private IP addresses.
- DNS name resolves to private IPs.
- Compatible with the **IPv4** and **Dualstack** IP address types.

### Load balancer IP address type | Info

Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

#### IPv4

Includes only IPv4 addresses.

#### Dualstack

Includes IPv4 and IPv6 addresses.

#### Dualstack without public IPv4

Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with **internet-facing** load balancers only.

## Security groups

### Create security group | Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

#### Basic details

##### Security group name | Info

ingh-refcard01-lb-sg

Name cannot be edited after creation.

##### Description | Info

sg für lb

##### VPC | Info

vpc-0add8c131c0f234da

#### Inbound rules | Info

##### Type | Info

HTTP

##### Protocol | Info

TCP

##### Port range | Info

80

##### Source | Info

Anywhere...

##### Description - optional | Info

0.0.0.0/0

Delete

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

X

#### Outbound rules | Info

##### Type | Info

Custom TCP

##### Protocol | Info

TCP

##### Port range | Info

8080

##### Destination | Info

Anywhere...

##### Description - optional | Info

0.0.0.0/0

Delete

Neu erstellte SG für den Loadbalancer einsetzen

### Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

#### Security groups

Select up to 5 security groups



ingh-refcard01-lb-sg  
sg-07c45f5efac1d600c VPC: vpc-0add8c131c0f234da

Successfully created the target group: singh-refcard01-tg. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.

## singh-refcard01-tg

### Details

arn:aws:elasticloadbalancing:us-east-1:231912049494:targetgroup/singh-refcard01-tg/2ac56cd2f021cebf

#### Target type

IP

#### Protocol : Port

HTTP: 80

#### Protocol version

HTTP1

#### VPC

vpc-0add8c131c0f234da

IP address type  
IPv4

Load balancer  
[None associated](#)

Listener and rules noch einbauen mit den TG

### Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes request:

#### ▼ Listener HTTP:80

Protocol: [HTTP](#) Port: 80  
1-65535

#### Default action | [Info](#)

Forward to: singh-refcard01-tg  
Target type: IP, IPv4

HTTP



[Create target group](#)

Jetzt sollten sie zu der Target Group weiter geleitet werden

Successfully created load balancer: singh-refcard01-lb

It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.



[Application Load Balancers now support public IPv4 IP Address Management \(IPAM\)](#)  
You can get started with this feature by configuring IP pools in the Network mapping section.

[Edit IP pools](#)



## singh-refcard01-lb



[Actions](#)

### ▼ Details

Load balancer type:  
Application

Status:  
[Provisioning](#)

VPC:  
[vpc-0add8c131c0f234da](#)

Load balancer IP address type:  
IPv4

Scheme:  
Internet-facing

Hosted zone:  
Z35SXDOTRQ7X7K

Availability Zones:  
[subnet-0c96f09db67e16c04](#) us-east-1a  
(use1-az6)

Date created:  
May 2, 2025, 10:31 (UTC+01:00)

Load balancer ARN:  
arn:aws:elasticloadbalancing:us-east-1:231912049494:loadbalancer/app/singh-refcard01-lb/563914cc47ab7e76

DNS name [Info](#)

[singh-refcard01-lb-710669504.us-east-1.elb.amazonaws.com](#) (A Record)



singh-refcard01-lb-710669504.us-east-1.elb.amazonaws.com



Browse thousands o...



Code of Stitch



Launch an instance |...



Module 1: Creating...



# Application Ref. Card 01

Spring Boot Application with no resources

IP Address (Server): 172.31.80.181

# This site can't be reached

**52.91.200.87** took too long respond.

**Try:**

- ✓ Checking the connection
- ✓ Checking the proxy and the firewall
- ✓ [\*\*Running Windows Network Diagnostics\*\*](#)

ERR\_CONNECTION\_TIMED\_OUT

**Check your internet connection.**

Check any cables and reboot any routers, modems, or other network devices you may be using.

**Allow Opera to access the network in your firewall or antivirus settings.**

If it is already listed as a program allowed to access the network, try removing it from the list and adding it again.

man sieht hier das es nur noch über den Loadbalancer geht

Aktualisierungsprozess für Versionen (ECS)

Der Aktualisierungsprozess einer Anwendung, die auf ECS mit Fargate läuft, umfasst typischerweise die folgenden Schritte, um eine neue Version des Docker-Images bereitzustellen:

1. **ECR-Repository und Docker Image:**

- **Docker Image erneuern:** Das bestehende Docker Image wird lokal neu erstellt (Build-Prozess), um Code-Änderungen oder Abhängigkeitsupdates einzubeziehen.

```
# Beispiel: Docker-Image mit einem neuen Versions-Tag bauen (z.B.  
v1)  
docker build -t singh-refcard01:v1 .  
  
# Alternativ: Docker-Image mit dem :latest Tag bauen (überschreibt  
das vorherige :latest)  
docker build -t singh-refcard01:latest .
```

- **Image neu taggen:** Das aktualisierte Image erhält einen eindeutigen Tag (z.B. eine neue Versionsnummer oder :latest). Es ist wichtig, das Image korrekt für das ECR-Repository zu taggen.

```
# Beispiel: Das lokale Image 'singh-refcard01:latest' für das ECR-  
Repository taggen  
# Ersetzen Sie 231912049494 und singh-refcard01 mit Ihrer AWS  
Account ID und Ihrem ECR Repository Namen  
docker tag singh-refcard01:latest 231912049494.dkr.ecr.us-east-  
1.amazonaws.com/singh-refcard01:latest
```

- **Push in ECR:** Das neu getaggte Image wird in das Amazon ECR-Repository hochgeladen. Stellen Sie sicher, dass Ihre AWS CLI korrekt konfiguriert ist ([aws configure](#)) und Sie die notwendigen Berechtigungen zum Pushen in das Repository haben.

```
# AWS CLI konfigurieren, falls noch nicht geschehen (interaktive  
Eingabeaufforderung)  
# aws configure  
  
# Docker-Image in das ECR-Repository pushen  
docker push 231912049494.dkr.ecr.us-east-1.amazonaws.com/singh-  
refcard01:latest
```

- **Zeitschätzung:** Dieser Vorgang (Build und Push) dauert bei mir in der Regel nicht länger als eine Minute. Bei grösseren Images kann es bis zu maximal drei Minuten dauern.

## 2. Task Definition:

- **Task Definition erneut revisieren/aktualisieren:** Die bestehende Task Definition wird überarbeitet, um auf das neue Image-Tag (die URI des soeben gepushten Images in ECR) zu verweisen. Alternativ kann eine neue Revision der Task Definition erstellt werden. Dies ist der empfohlene Weg, da er ein Rollback erleichtert.
- **Zeitschätzung:** Das Überarbeiten oder Neuerstellen der Task Definition hat bei mir aufgrund anfänglicher Unklarheiten etwas Zeit in Anspruch genommen, insgesamt jedoch nicht mehr

als drei Minuten.

### 3. Service:

Service aktualisieren: Der ECS-Service wird aktualisiert, um die neue Revision der Task Definition zu verwenden. Dies löst ein neues Deployment aus.

Zeitschätzung für Service-Aktualisierung:

Die Initialisierung eines Deployments dauert pro Service etwa zwei Minuten.

Bis die neuen Tasks vollständig gestartet und betriebsbereit sind ("running"), vergehen üblicherweise rund fünf Minuten.

Der Load Balancer benötigt zusätzliche Zeit für Health Checks der neuen Tasks und das ordnungsgemäße Herunterfahren der alten Tasks ("Draining"), was weitere drei bis fünf Minuten dauern kann.

### 4. Rolling Updates (Hinweise zum Prozess):

Während eines Rolling Updates laufen alte und neue Tasks für etwa zwei bis drei Minuten parallel.

In dieser Übergangsphase ist es entscheidend, dass beide Task-Generationen die definierten Mindestanforderungen an Zustand und Verfügbarkeit (definiert durch "Minimum Healthy Percent" in der Service-Konfiguration) erfüllen. Dies stellt die kontinuierliche Erreichbarkeit des Services sicher.

Persönliches Fazit:

Leider konnte ich die zweite Zusatzaufgabe nicht bearbeiten. Der Grund dafür war eine sehr ernste persönliche Krise, die meine ganze Energie und Aufmerksamkeit beansprucht hat. Dadurch war es mir nicht möglich, die zusätzliche Arbeit zu leisten. Es tut mir sehr leid, dass ich diese Umstände nicht früher mitgeteilt habe und die Aufgabe daher unbearbeitet geblieben ist.

Trotz dieser schwierigen Situation bin ich froh, dass ich die übrigen Kernaufgaben erfolgreich abschliessen konnte. Anfangs hatte ich einige Schwierigkeiten mit AWS, die ich aber überwinden konnte.

Ich hoffe, dass diese Dokumentation einen besseren Eindruck macht als meine vorherige. Auch wenn sie noch nicht perfekt ist – was unter anderem daran liegt, dass ich aufgrund der erwähnten Umstände verspätet mit der Bearbeitung beginnen konnte – bin ich diesmal deutlich zuversichtlicher und stolz darauf, sie einzureichen. Im Gegensatz zur letzten Dokumentation konnte ich diesmal alle wichtigen Schritte erfolgreich umsetzen und dokumentieren.