







Capture The Flag vorbereitung



Dockerfile

Als ersten Schritt forken wir das Repository, das uns als Aufgabe gestellt wurde:


**Architecture Ref. Card 03** 
Project ID: 45214716  [Request Access](#)


  Star 2  Fork 103

3 Commits 2 Branches 0 Tags 6.3 MiB Project Storage

 **remove Dockerfile in this stage**
Dominic Rüttimann authored 2 months ago b7e23dfe 

main ref-card-03 / + Find file Edit Download Clone

 Forked from [Rinaldo / M346 Ref Card 03](#)
3 commits behind, 3 commits ahead of the upstream repository.

 README

Name	Last commit	Last update
src/main	Initial commit	2 months ago
README.md	add basic README.md	2 months ago
pom.xml	Initial commit	2 months ago

Anschließend erstellen wir ein Dockerfile, welches einen ausführbaren Docker-Container generiert (Testen in einer VM oder im WSL, wenn auf Sicherheit geachtet werden soll). Nachfolgend sehen Sie ein beispielhaften Code:



Update Dockerfile

Manuel Regli authored 1 month ago

ce251037



main

ref-card-03 / Dockerfile

Find file

Blame

History

Permalink



Forked from [bbwin / Architecture Ref. Card 03](#)

26 commits ahead of the upstream repository.

Create merge request



Dockerfile



166 B

Edit

Replace

Delete



```
1 FROM maven:3-openjdk-17-slim
2
3 COPY src /src
4 COPY pom.xml /
5
6 RUN mvn -f /pom.xml clean package
7 RUN mv /target/*.jar /app.jar
8
9
10 ENTRYPOINT ["java", "-jar", "/app.jar"]
11
```

```
FROM maven:3-openjdk-11-slim as builder
COPY src /src
COPY pom.xml /
RUN mvn -f pom.xml clean package
FROM adoptopenjdk/openjdk11:alpine-jre
COPY --from=builder /target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

```
FROM maven:3.6.1-jdk-13-alpine as builder
COPY src /src
COPY pom.xml /
RUN mvn -f pom.xml clean package
FROM adoptopenjdk/openjdk13:alpine-jre
COPY --from=builder /target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

```
FROM maven:3.8.5-openjdk-17-slim as builder
COPY src /src
COPY pom.xml /
RUN mvn -f pom.xml clean package
FROM eclipse-temurin:17-jre-alpine
COPY --from=builder /target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Repository erstellen

Nachdem das Dockerfile erstellt wurde, starten wir das Learner-Lab, navigieren zum Service ECR und erstellen ein neues privates Repository.

Amazon ECR > Repositories > Create repository

Create repository

General settings

Visibility settings | [Info](#)
Choose the visibility setting for the repository.

☒ **Private**
Access is managed by IAM and repository policy permissions.

☐ **Public**
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.

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

10 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability | [Info](#)
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☐ **Disabled**

i Once a repository is created, the visibility setting of the repository can't be changed.

Private repositories (1)

View push commands

Delete

Actions

Create repository

Find repositories

<

1

>

<input type="checkbox"/>	Repository name ▲	URI	Created at ▼	Tag immutability	Scan frequency	Encryption type	Pull through cache
<input type="checkbox"/>	refcard-03	910977011815.dkr.ecr.us-east-1.amazonaws.com/refcard-03	June 27, 2023, 21:30:48 (UTC+02)	Disabled	Manual	AES-256	Inactive

GitLab-Variablen

Um geheime Informationen für CI/CD nicht für jeden zugänglich zu machen, müssen wir Secret Variables erstellen. Die ersten drei finden wir im Learner Lab, wenn wir auf AWS-Details und dann auf AWS CLI klicken. Die Variablen heißen: "AWS_ACCESS_KEY_ID", "AWS_SECRET_ACCESS_KEY" und "AWS_SESSION_TOKEN".

03:37 ▶ Start Lab ■ End Lab ⓘ AWS Details ⓘ Readme ↺ Reset ✕

Cloud Access

AWS CLI:
Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=ASTA5T6FM88773YD24CP
aws_secret_access_key=ACVYGRXpR0LJaCF-0701-5cedhd211YAB56U
MCV
aws_session_token=FwoGZXIvYXdzEA0aDCS+AFqGJLPiPnbVlyLNAeaNa
x0aZ+cIKSpvo9Mqmk1CmrgV43j-3hKukcJCMG2NXht/qmn9+xMZQ5foKar
K0iX6Dp44Y-0701-16Ui3q36a6RWXhkIiym
3ogr7Sd-0701-44F40oqwT5j
t41h78+e/swJ5-0701-44F40oqwT5j
YVH6Rf49VEL-0701-44F40oqwT5j
DYpxKLfhyZcUKA10vZQbpxPudhc920tn+IldvkBZ2/4yBQ==
```

Cloud Labs
Remaining session time: 03:36:32(217 minutes)
Session started at: 2023-06-27T12:26:16-0700
Session to end at: 2023-06-27T16:26:16-0700

Accumulated lab time: 1 day 17:17:00 (2477 minutes)

No running instance

SSH key Show Download PEM Download PPK
AWS SSO Download URL

AWSAccountid	910977011815
Region	us-east-1

Die weiteren drei Variablen finden wir im neu erstellten Repository. Die Variablen und ihre Werte sind: "AWS_DEFAULT_REGION" = "us-east-1", "CI_AWS_ECR_REGISTRY" =

"910977011815.dkr.ecr.us-east-1.amazonaws.com" und "CI_AWS_ECR_REPOSITORY_NAME" = "refcard-03".

Push commands for refcard-03



macOS / Linux

Windows

Make sure that you have the latest version of the AWS CLI 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 CLI:

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

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI 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 is already built:

```
docker build -t refcard-03 .
```

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

```
docker tag refcard-03:latest 910977011815.dkr.ecr.us-east-1.amazonaws.com/refcard-03:latest
```


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

```
docker push 910977011815.dkr.ecr.us-east-1.amazonaws.com/refcard-03:latest
```

Close

.gitlab-ci.yml erstellen


Im Folgenden wird ein .gitlab-ci.yml vorgestellt, das dazu dient, ein Docker-Image zu generieren und es in ein AWS-Repository hochzuladen.



Update `.gitlab-ci.yml`
Manuel Regli authored 2 weeks ago

✓

ce6e2d3b



main ▾


ref-card-03 / .gitlab-ci.yml

Find file



Blame

History

Permalink

 Forked from [bbwin / Architecture Ref. Card 03](#)
26 commits ahead of the upstream repository.


Create merge request


 `.gitlab-ci.yml`  581 B


Edit ▾

Replace

Delete







```
1 image: docker:23.0.4
2
3 variables:
4   DOCKER_HOST: tcp://docker:2375
5   DOCKER_TLS_CERTDIR: ""
6
7 services:
8   - docker:23.0.4-dind
9
10 package:
11   stage: build
12   before_script:
13     - apk add --no-cache py3-pip
14     - pip install awscli
15     - aws --version
16
17     - aws ecr get-login-password | docker login --username AWS --password-stdin $CI_AWS_ECR_REGISTRY
18
19 script:
20   - docker build --cache-from $CI_AWS_ECR_REGISTRY/$CI_AWS_ECR_REPOSITORY_NAME:latest -t $CI_AWS_ECR_REGISTRY/$CI_AWS_ECR_REPOSITORY_NAME:latest
21   - docker push $CI_AWS_ECR_REGISTRY/$CI_AWS_ECR_REPOSITORY_NAME:latest
22
```

image: docker:23.0.4

variables:

DOCKER_HOST: tcp://docker:2375

DOCKER_TLS_CERTDIR: ""

services:

- docker:23.0.4-dind

package:

stage: build

before_script:

- apk add --no-cache py3-pip

- pip install awscli

- aws --version

- aws ecr get-login-password | docker login --username AWS --password-stdin \$CI_AWS_ECR_REGISTRY

script:

- docker build --cache-from \$CI_AWS_ECR_REGISTRY/\$CI_AWS_ECR_REPOSITORY_NAME:latest -t \$CI_AWS_ECR_REGISTRY/\$CI_AWS_ECR_REPOSITORY_NAME:latest .

- docker push \$CI_AWS_ECR_REGISTRY/\$CI_AWS_ECR_REPOSITORY_NAME:latest

Sobald diese Datei erstellt wurde, sollte automatisch ein Runner gestartet werden, vorausgesetzt ein solcher ist konfiguriert. Dies kann man unter Settings - CI/CD - Runners überprüfen.

Amazon ECR > Repositories > refcard-03

refcard-03

[View push commands](#) [Edit](#)

Images (1) [Refresh](#) [Delete](#) [Details](#) [Scan](#)

< 1 > ⚙

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
<input type="checkbox"/>	latest	Image	June 27, 2023, 21:56:57 (UTC+02)	371.34	Copy URI	sha256:1498f47c9640930b2734fa82aedd82...	-	-

Erstellen eines ECS Clusters

Anschließend beginnen wir mit der Erstellung eines ECS Clusters. Der Name spielt hierbei keine Rolle, wichtig ist nur, den Namespace zu entfernen, damit das Cluster im Learner Lab erstellt werden kann.

Amazon Elastic Container Service > Create 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

There can be a maximum of 255 characters. The valid characters are letters (uppercase and lowercase), numbers, hyphens, and underscores.

Datenbank erstellen (optional)

Für refcard03 war es notwendig, eine Datenbank zu verbinden, daher habe ich ein RDS erstellt. Für die Capture The Flag Aufgabe ist dies jedoch nicht erforderlich.

jokedb

ModifyActions

Summary

DB identifier jokedb	CPU -	Status ⌚ Backing-up	Class db.t3.micro
Role Instance	Current activity	Engine MariaDB	Region & AZ us-east-1a

Connectivity & security

Monitoring

Logs & events

Configuration

Maintenance & backups

Tags

Connectivity & security

<div>Endpoint & port</div> <div>Endpoint jokedb.ca574jaewyqv.us-east-1.rds.amazonaws.com</div> <div>Port 3306</div>	<div>Networking</div> <div>Availability Zone us-east-1a</div> <div>VPC vpc-05cdfb3208fad5c27</div> <div>Subnet group default-vpc-05cdfb3208fad5c27</div> <div>Subnets subnet-0ed51d4e0f72210a2 subnet-03b43b7ce82ce4ad4 subnet-09645762315c3ee00 subnet-005851b1264b729ea subnet-06c2afd2ff316b431</div>	<div>Security</div> <div>VPC security groups default (sg-08919adf51d796f7c) ✔ Active</div> <div>Publicly accessible No</div> <div>Certificate authority Info rds-ca-2019</div> <div>Certificate authority date August 22, 2024, 19:08 (UTC+02:00)</div> <div>DB instance certificate expiration date August 22, 2024, 19:08 (UTC+02:00)</div>
---	--	---

Erstellung einer Task Definition

Nun erstellen wir eine Taskdefinition mit der Image-URL des Repositories (am Ende sollte :latest stehen). Der Port ist vom Image abhängig und die Environment-Variablen werden nur mit der RDS-Datenbank benötigt.

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	Image URI	Essential container
<input type="text" value="spring"/>	<input type="text" value="910977011815.dkr.ecr.us-east-1.amazonaws.com/refca"/>	<input type="text" value="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. Any changes to port mappings configuration impacts the associated service connect settings.

Container port	Protocol	Port name	App protocol	
<input type="text" value="8080"/>	<input type="text" value="TCP"/>	<input type="text" value="spring-8080-tcp"/>	<input type="text" value="HTTP"/>	Remove

[Add more port mappings](#)

▼ Environment variables - optional [Info](#)

Add individually

Add a key-value pair to specify an environment variable.

Key	Value type	Value	
<input type="text" value="DB_URL"/>	<input type="text" value="Value"/>	<input type="text" value="onaws.com:3306/jokedb"/>	Remove
<input type="text" value="DB_USERNAME"/>	<input type="text" value="Value"/>	<input type="text" value="jokedbuser"/>	Remove
<input type="text" value="DB_PASSWORD"/>	<input type="text" value="Value"/>	<input type="text" value="12345678"/>	Remove

[Add environment variable](#)

Für Task-Rolle und Task-Ausführungsrolle sollten wir "LabRole" auswählen.

▼ Task roles, network mode - *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 ▼

Network mode [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.

awsvpc ▼

Änderung der Security Group

Aus Gründen der Schnelligkeit kann man in der Default Security Group festlegen, dass jeder eingehende Traffic zugelassen wird.

EC2 > Security Groups > sg-08919adf51d796f7c - default > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
Security group rule ID					
-	All traffic ▼	All	All	Anywhere-I... ▼	
				0.0.0.0/0 X	
Add rule					
Cancel Preview changes Save rules					

Erstellung eines Services im ECS Cluster

Folgen Sie den untenstehenden Konfigurationen (je nach Docker-Image können Änderungen erforderlich sein).

Create [Info](#)

Environment

AWS Fargate

Existing cluster

Select an existing cluster. To create a new cluster, go to [Clusters](#).

bbw

▼ 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). External instances are registered to your cluster using the ECS Anywhere capability.

FARGATE ▼

Platform version [Info](#)

Specify the platform version on which to run your service.

LATEST ▼

Deployment configuration

Application type [Info](#)

Specify what type of application you want to run.

☒ **Service**

Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

☐ **Task**

Launch a standalone task that runs and terminates. For example, a batch job.

Task definition

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

☐ **Specify the revision manually**

Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family

refcard-03 ▼

Revision

3 (LATEST) ▼

Service name

Assign a unique name for this service.

refcard-03



Service type [Info](#)

Specify the service type that the service scheduler will follow.

☒ **Replica**

Place and maintain a desired number of tasks across your cluster.

☐ **Daemon**

Place and maintain one copy of your task on each container instance.

Desired tasks

Specify the number of tasks to launch.

1

► **Deployment options**

► **Deployment failure detection** [Info](#)

▼ Load balancing - optional

Load balancer type | Info

Configure a load balancer to distribute incoming traffic across the tasks running in your service.

Application Load Balancer ▼

Application Load Balancer

Specify whether to create a new load balancer or choose an existing one.

- ☒ Create a new load balancer
☐ Use an existing load balancer

Load balancer name

Assign a unique name for the load balancer.

lb-refcard-03

Choose container to load balance

spring 8080:8080 ▼

Listener | Info

Specify the port and protocol that the load balancer will listen for connection requests on.

- ☒ Create new listener
☐ Use an existing listener

You need to select an existing load balancer.

Port

80

Protocol

HTTP ▼

Target group | Info

Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

- ☒ Create new target group
☐ Use an existing target group

You need to select an existing load balancer.

Target group name

tg-refcard-03

Protocol

HTTP ▼

Health check path | Info

/

Health check protocol

HTTP ▼

Health check grace period | Info

100

seconds

Testen des Loadbalancers

Um den Loadbalancer zu testen, können wir zu EC2 gehen und dort auf Loadbalancer klicken. Anschließend kopieren wir den DNS-Namen.

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Scheduled Instances

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Auto Scaling

EC2 > Load balancers > lb-refcard-03

lb-refcard-03

Details

Load balancer type	Status	VPC	IP address type
Application	Provisioning	vpc-05cdfb3208fad5c27	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z355XDOTRQ7X7K	subnet-0ed51d4e0f72210a2 us-east-1c (use1-az4)	June 27, 2023, 22:12 (UTC+02:00)
		subnet-03b43b7ce82ce4ad4 us-east-1e (use1-az3)	
		subnet-09645762315c3ee00 us-east-1b (use1-az2)	
		subnet-005851b1264b729ea us-east-1d (use1-az6)	
		subnet-06c2afd2ff316b431 us-east-1f (use1-az5)	
		subnet-0e3e4f455b118e408 us-east-1a (use1-az1)	
Load balancer ARN	DNS name		
arn:aws:elasticloadbalancing:us-east-1:910977011815:loadbalancer/app/lb-refcard-03/037af4e92840ec31	lb-refcard-03-1713030138.us-east-1.elb.amazonaws.com (A Record)		

Listeners (0) info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners by property or value

Diesen Loadbalancer-Namen geben wir dann in den Browser ein und nach einiger Zeit sollten wir die Webanwendung sehen.

Modulbaukasten

Learner Lab

EC2 Management Console

Load balancers | EC2 Management Console

Amazon ECS

CloudFormation - Stack ECS-Card 03

Architecture Ref. Card 03

Not secure | lb-refcard-03-1713030138.us-east-1.elb.amazonaws.com

Architecture Ref. Card 03

Spring Application with JPA/Database (MariaDB)

Witz 1

Kunde: "Ich möchte Ihren Chef sprechen!" Sekretärin: "Geht leider nicht, er ist nicht da!" Kunde: "Ich hab ihn doch durchs Fenster gesehen!" Sekretärin: "Er Sie auch!"

2014-01-08, Flachwitze

Rating: 5

Witz 2

Der Verwaltungsrat zum CEO: "Na, wie macht sich denn der neue Buchhalter?" CEO: "Toll, dieser Mann!" Verwaltungsrat: "Was kann er denn so besonderes?" CEO: "Er ist gelernter Friseur, er kann frisieren!"

2014-01-08, Flachwitze

Rating: 3

Witz 3

Chef: "Müller, Sie sind das beste Pferd in meinem Stall!" Müller: "Wirklich, Chef?" Chef: "Ja, Sie machen den meisten Mist!"

2014-01-08, Flachwitze

Rating: 5

Witz 6

Was steht auf dem Grabstein eines Mathematikers? "Damit hat er nicht gerechnet."

2021-04-06, Schwarzer Humor

Rating: 3