

Name: Mrunali Katta

ID: 017516785

DATA 235 – Homework 01

I. HTML

1. Create an HTML page with the title "HW1-{Your Name}"

Here created a file with name '**index_hw1.html**' with title "**HW1-Mrunali Katta**"

```
<!-- html page -->
<!DOCTYPE html>
<!--root element-->
<html lang="en">
    <!--head tag start-->
    <head>
        <!-- meta tags -->
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- HTML page with the title-->
        <title>HW1-Mrunali Katta</title>
    </head>
```



Blog Title

Blog Title*

Enter the title of your blog

Author Name*

Enter your name

Author Email Address*

Enter your email

Content*

Write your content here...

Category*

Technology

I agree to the terms and conditions

Publish Blog

2. Added a heading tag to name the title of the as “**Blog Topic**” using header tag which renders the biggest font size `<h1>`

```
<body>
    <!--title of the blog topic-->
    <!--biggest font size-->
    <h1>Blog Title</h1>
```

The screenshot shows a web browser window titled "HW1-Mrunali Katta" with the URL "localhost:8080". The page displays a form for creating a blog post. The title "Blog Title" is highlighted with a yellow background. The form fields are as follows:

- Blog Title*:
- Author Name*:
- Author Email Address*:
- Content*:
- Category*:
- I agree to the terms and conditions
-

3. Created a Form:

- a) Displaying a text input for the blog title, placeholder text “Enter the title of your blog”. This field should be required and the cursor should automatically focus on this field when the page loads.

```
<!--form tag to collect inputs from user-->
<form id="myBlogForm">
    <label for="blogTitle">Blog Title*</label><br>
    <!-- using autofocus the cursor will automatically be placed in the
textfield-->
    <input type="text" id="myBlogTitle" placeholder="Enter the title of
your blog" required autofocus><br><br>
```

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*
Enter the title of your blog

Author Name*
Enter your name

Author Email Address*
Enter your email

Content*
Write your content here...

Category*
Technology ▾

I agree to the terms and conditions

Publish Blog

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*
Enter the title of your blog

Please fill out this field.

Author Email Address*
Enter your email

Content*
Write your content here...

Category*
Technology ▾

I agree to the terms and conditions

Publish Blog

Elements Console

Running Ginger Widget v2.11.
Running Ginger Widget v2.11.

Console What's new

What's new in DevTo
See all new features

Highlights of

Search

Windows Start button

File Explorer

Google Chrome

Microsoft Edge

OneDrive

Spotify

Microsoft Word

Microsoft Excel

Microsoft Powerpoint

Microsoft Teams

Microsoft Word

Microsoft Excel

Microsoft Powerpoint

Microsoft Teams

Microsoft Word

Microsoft Excel

Microsoft Powerpoint

Microsoft Teams

b) A text input for the author name, placeholder "Enter your name", and required.

```
<label for="blogName">Author Name*</label><br>
    <input type="text" id="userName" placeholder="Enter your name"
required><br><br>
```

The screenshot shows a browser window with a developer tools console open. The page displays a form for creating a blog post. The fields include:

- Blog Title***: Input field containing "Homework01".
- Author Name***: Input field with placeholder "Enter your name". A tooltip error message "Please fill out this field." is displayed below it.
- Content***: Text area with placeholder "Write your content here...".
- Category***: Select dropdown menu showing "Technology".
- I agree to the terms and conditions**: An unchecked checkbox.
- Publish Blog**: A button at the bottom of the form.

The developer tools console shows two entries: "Running Ginger Widget v2.11.3" and "Running Ginger Widget v2.11.3". The browser taskbar at the bottom includes icons for File Explorer, Search, Task View, and other system utilities.

c) An email input for the email address, placeholder "Enter your email", and required.

```
<label for="blogEmailAddress">Author Email Address*</label><br>
    <input type="email" id="userEmailAddress" placeholder="Enter your
email" required><br><br>
```

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*
Homework01

Author Name*
Mrunali

Author Email Address*
Enter your email
here...  Please fill out this field.

Category*
Technology

I agree to the terms and conditions

Publish Blog

Console Elements Sources Network > Default levels 1 Issue: 1 content.min.js:6526 writer.min.js:6526

Running Ginger Widget v2.11.318
Running Ginger Widget v2.11.318

What's new X AI assistance Issues

What's new in DevTools 133

See all new features

Highlights of updates from Chrome 130-132

12:01 AM 2/6/2025

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*
Homework01

Author Name*
Mrunali

Author Email Address*
mrunali  Please include an '@' in the email address.
'mrunali' is missing an '@'.

Category*
Technology

I agree to the terms and conditions

Publish Blog

Console Elements Sources Network > Default levels 1 Issue: 1 content.min.js:6526 writer.min.js:6526

Running Ginger Widget v2.11.318
Running Ginger Widget v2.11.318

What's new X AI assistance Issues

What's new in DevTools 133

See all new features

Highlights of updates from Chrome 130-132

12:02 AM 2/6/2025

Blog Title

Blog Title*
Homework01

Author Name*
Mrunali

Author Email Address*
mrnali@

! Please enter a part following '@'. 'mrnali@' is incomplete.

Category*
Technology

I agree to the terms and conditions

Publish Blog

Console What's new X AI assistance Issues

Running Ginger Widget v2.11.318
Running Ginger Widget v2.11.318

content.min.js:6526
writer.min.js:6526

d) A text area for the blog content, placeholder “Write your content here...”, and required.

```
<label for="blogContent">Content*</label><br>
<textarea id="myBlogContent" placeholder="Write your content here...">
<required></textarea><br><br>
```

Blog Title

Blog Title*
Homework01

Author Name*
Mrunali

Author Email Address*
mrnali@hw1

Content*
Write your content here...

! Please fill out this field.

Technology

I agree to the terms and conditions

Publish Blog

Console What's new X AI assistance Issues

Running Ginger Widget v2.11.318
Running Ginger Widget v2.11.318

content.min.js:6526
writer.min.js:6526

e) A dropdown for category selection, and options "Technology," "Lifestyle," "Travel," and "Education."

```
<label for="blogCategories">Category*</label><br>
  <select id="myBlogCategories">
    <option value="Technology">Technology</option>
    <option value="Lifestyle">Lifestyle</option>
    <option value="Travel">Travel</option>
    <option value="Education">Education</option>
  </select><br><br>
```

The screenshot shows a web browser window with a form for submitting a blog post. The form includes fields for Blog Title, Author Name, Author Email Address, Content, and Category. The Category field is a dropdown menu with options: Technology, Lifestyle, Travel, and Education. The 'Lifestyle' option is currently selected. The browser's developer tools are open, specifically the Console tab, which displays logs from the Ginger Widget and content.min.js. The system tray at the bottom of the screen shows various icons for system functions like battery level, signal strength, and language settings.

f) A checkbox and a label with the text "I agree to the terms and conditions."

```
<input type="checkbox" id="terms">
  <label for="terms">I agree to the terms and
  conditions</label><br><br>
```

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*

Homework01

Author Name*

Mrunali

Author Email Address*

mrunali@hw1

Content*

This is homework 1
submission

Category*

Lifestyle

I agree to the terms and conditions

Publish Blog

g) A submit button with the text “Publish Blog”.

```
<button type="submit">Publish Blog</button>
```

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*

Homework01

Author Name*

Mrunali

Author Email Address*

mrunali@hw1

Content*

This is homework 1
submission

Category*

Lifestyle

I agree to the terms and conditions

Publish Blog

4. Add a script tag to link your javascript code for part II at the end of your HTML file.

Here the “index_hw1.html” file is linked to .js file called “**blog_hw1.js**”

```
</form>
  <script src="blog_hw1.js"></script>
</body>
</html>
```

II. Javascript

1. Write a javascript function using an arrow function to validate:
 - a. Verify if the blog content is more than 25 characters. If the validation fails, display an alert with the message “Blog content should be more than 25 characters”.

```
//using "use strict" mode for error handling and to impose stricter parsing
in JS code
"use strict";

// form validation
const formValidation = () => {
    const blogTitle = document.getElementById("myBlogTitle").value.trim();
    const authorName = document.getElementById("userName").value.trim();
    const authorEmail =
document.getElementById("userEmailAddress").value.trim();
    const blogContent =
document.getElementById("myBlogContent").value.trim();
    const termsChecked = document.getElementById("terms").checked;

    // to see if all fields are filled
    if (!blogTitle || !authorName || !authorEmail || !blogContent) {
        alert("All fields are required!");
        return false;
    }

    // blog content > than 25 characters check condition
    if (blogContent.length <= 25) {
        alert("Blog content should be more than 25 characters");
        return false;
    }
}
```

localhost:8080 says
Blog content should be more than 25 characters

OK

Console Sources Network > Filter Default levels ▾ 1 Issue: 1 | 3 hidden

dget v2.11.318 content.min.js:6526
dget v2.11.318 writer.min.js:6526

Blog Title*
Homework01

Author Name*
Mrunali

Author Email Address*
mrunali@hw1

Content*
This is homework 1

Category*
Technology ▾

I agree to the terms and conditions

Console What's new X AI assistance Issues

What's new in DevTools 133 See all new features

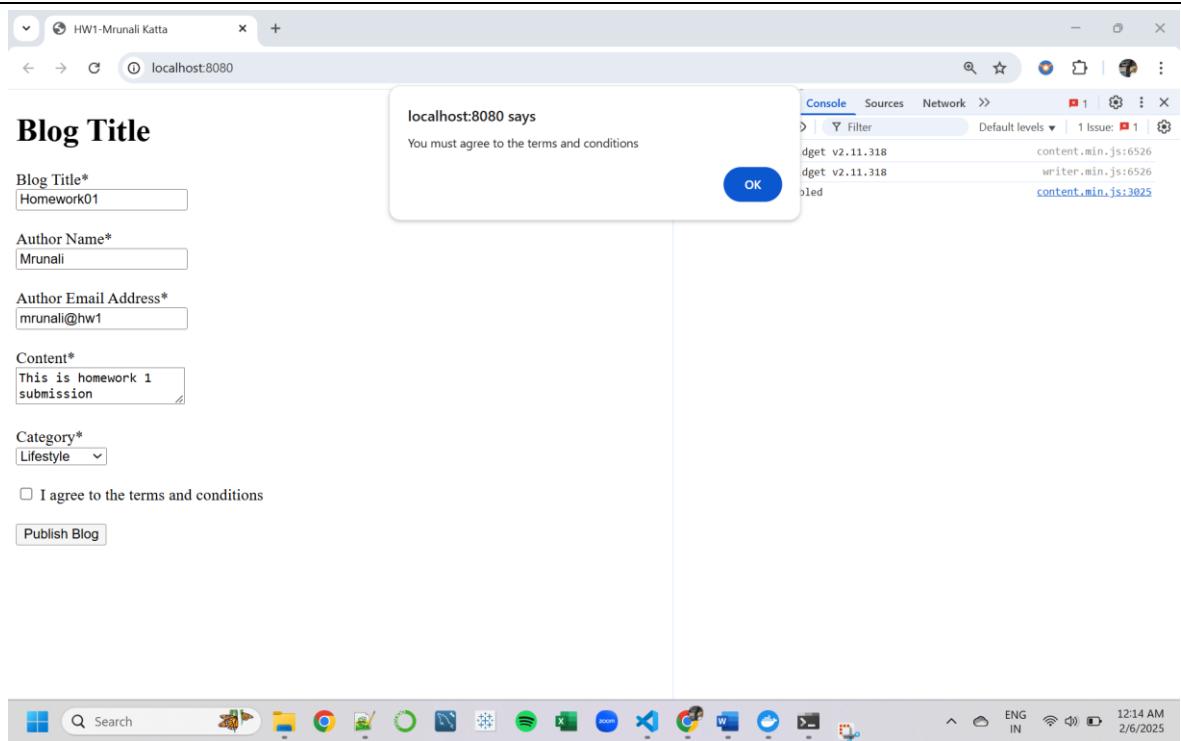
Highlights of updates from Chrome 130-132

Search ENG IN 12:04 AM 2/6/2025

b. Verify if the terms and conditions check box is checked. If the validation fails, display an alert with the message "You must agree to the terms and conditions".

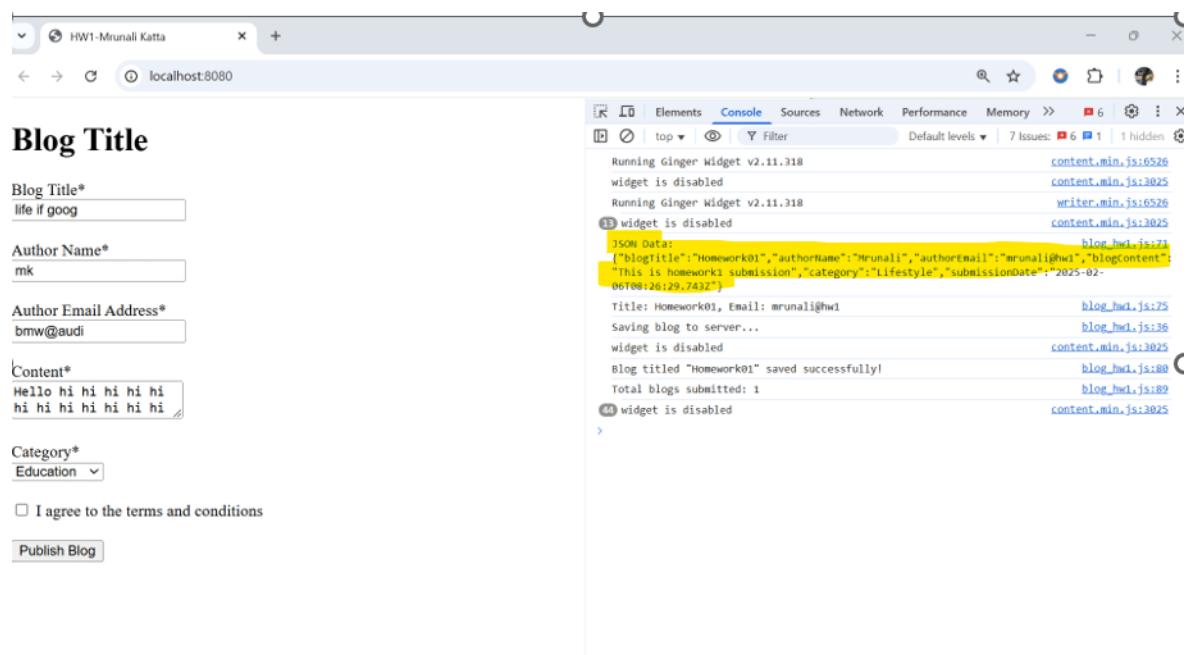
```
// terms and conditions - condition
if (!termsChecked) {
    alert("You must agree to the terms and conditions");
    return false;
}

return true;
}; // end of formValidation
```

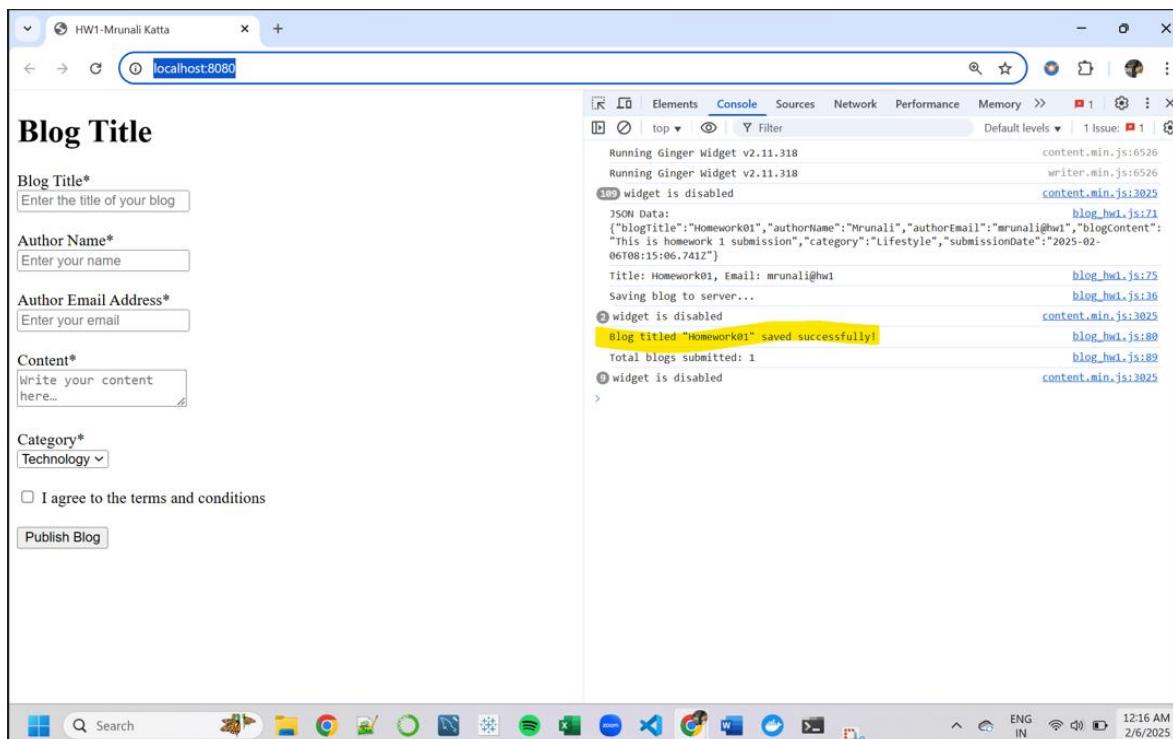


2. After the form submission is successful, convert the form data into a JSON string and log the output in the console.

```
// converting data to JSON ONLY after validation
const jsonData = JSON.stringify(updatedBlogData);
console.log("JSON Data:", jsonData);
```



```
// to simulate an asynchronous operation
const saveBlogToServer = (blogData) => {
    return new Promise((resolve, reject) => {
        console.log("Saving blog to server...");
        setTimeout(() => {
            const success = Math.random() > 0.2; // 80% success rate
            if (success) {
                resolve(`Blog titled "${blogData.blogTitle}" saved
successfully!`);
            } else {
                reject("Failed to save the blog. Please try again.");
            }
        }, 3000); // insert a 3-second delay
    });
};
```



3. Use object destructuring to extract the title and email fields from the parsed object and log their values in the console.

```
// extracting title and email
const { blogTitle, authorEmail } = updatedBlogData;
console.log(`Title: ${blogTitle}, Email: ${authorEmail}`);
```

The screenshot shows a browser window with a form titled "Blog Title". The form fields include "Blog Title*", "Author Name*", "Author Email Address*", "Content*", "Category*", and checkboxes for "I agree to the terms and conditions" and "Publish Blog". The developer tools' Console tab is open, displaying logs from "blog_hw1.js" and "content.min.js". The logs show the submission process, including JSON data being sent to the server and confirmation messages like "Blog titled 'Homework01' saved successfully!".

4. Use the spread operator to add a new field "submissionDate" with the current date and time to the parsed object. Log the updated parsed object in the console.

```
// spread operator to Add Submission Date
const updatedBlogData = {
  ...blogData,
  submissionDate: new Date().toISOString()
};
```

This screenshot is similar to the first one, showing the same form and developer console. However, the developer console now includes a log entry for the "updatedBlogData" object, which includes the newly added "submissionDate" field with the current ISO string value.

5. Create a closure to track how many times the form has been successfully submitted and log the submission count each time the form is submitted.

```
// a closure to track the number of blog submissions
const blogCounter = (() => {
  let count = 0;
  return () => ++count;
})();
```

HW1-Mrunali Katta

localhost:8080

Blog Title

Blog Title*
Homework01

Author Name*
Mrunali

Author Email Address*
mrunali@hw1

Content*
This is homework 1 submission

Category*
Lifestyle

I agree to the terms and conditions

localhost:8080 says
Blog saved successfully!

OK

Console Sources Network >

Default levels ▾ 1 Issue: 1 | 1 OK

dget v2.11.318 content.min.js:6526

dget v2.11.318 writer.min.js:6526

loaded content.min.js:3025

blog_hw1.js:71

{"blogTitle": "Homework01", "authorName": "Mrunali", "authorEmail": "mrunali@hw1", "blogContent": "This is homework 1 submission", "category": "Lifestyle", "submissionDate": "2025-02-06T08:15:06.741Z"}

Title: Homework01, Email: mrunali@hw1

Saving blog to server...

widget is disabled

Blog titled "Homework01" saved successfully!

blog_hw1.js:136 content_min.js:3025 blog_hw1.js:80

Count updated to 1

The screenshot shows a web browser window with the title "HW1-Mrunali Katta" and the URL "localhost:8080". The page displays a "Blog Title" form with fields for "Blog Title*", "Author Name*", "Author Email Address*", "Content*", and "Category*". Below the form is a checkbox for "I agree to the terms and conditions" and a "Publish Blog" button. The browser's developer tools are open, specifically the "Console" tab, which shows the following log entries:

```
Running Ginger Widget v2.11.318
widget is disabled
Running Ginger Widget v2.11.318
widget is disabled
JSON Data:
{
  "blogTitle": "Homework01",
  "authorName": "Mrunali",
  "authorEmail": "mrunali@hw1.com",
  "category": "Lifestyle",
  "submissionDate": "2025-02-06T08:26:29.743Z"
}
Title: Homework01, Email: mrunali@hw1.com
Saving blog to server...
widget is disabled
Blog titled "Homework01" saved successfully!
Total blogs submitted: 1
widget is disabled
```

Blog Title

Blog Title*
life if goog

Author Name*
mk

Author Email Address*
bmw@audi

Content*
Hello hi hi hi hi
hi hi hi hi hi

Category*
Education

I agree to the terms and conditions

Publish Blog

Count updated to 2

Running Ginger Widget v2.11.318
widget is disabled
Running Ginger Widget v2.11.318
widget is disabled
JSON Data:
{"blogTitle": "Homework01", "authorName": "Mrunali", "authorEmail": "mrunali@hw1", "blogContent": "This is homework1 submission", "category": "Lifestyle", "submissionDate": "2025-02-06T08:26:29.743Z"}
title: Homework01, Email: mrunali@hw1
Saving blog to server...
widget is disabled
Blog titled "Homework01" saved successfully!
Total blogs submitted: 1
widget is disabled

Blog Title

Blog Title*
Enter the title of your blog

Author Name*
Enter your name

Author Email Address*
Enter your email

Content*
Write your content
here...

Category*
Technology

I agree to the terms and conditions

Publish Blog

Running Ginger Widget v2.11.318
widget is disabled
Running Ginger Widget v2.11.318
widget is disabled
JSON Data:
{"blogTitle": "Homework01", "authorName": "Mrunali", "authorEmail": "mrunali@hw1", "blogContent": "This is homework1 submission", "category": "Lifestyle", "submissionDate": "2025-02-06T08:26:29.743Z"}
title: Homework01, Email: mrunali@hw1
Saving blog to server...
widget is disabled
Blog titled "Homework01" saved successfully!
Total blogs submitted: 1
widget is disabled
JSON Data: {"blogTitle": "life if goog", "authorName": "mk", "authorEmail": "bmw@audi", "blogContent": "Hello hi hi hi hi hi hi hi", "category": "Education", "submissionDate": "2025-02-06T08:29:34.279Z"}
title: life if goog, Email: bmw@audi
Saving blog to server...
widget is disabled
Blog titled "life if goog" saved successfully!
Total blogs submitted: 2
widget is disabled

III. DEPLOYMENT:

1) Dockerize the Application

1. Create a Dockerfile in the same directory as your application

DS_HW1					
This PC > New Volume (D:) > Distributed systems > hw1 > DS_HW1					
	Name	Date modified	Type	Size	
	blog_hw1	2/5/2025 11:03 PM	JavaScript Source File	4 KB	
	Dockerfile	2/5/2025 11:19 PM	File	1 KB	
	index_hw1	2/5/2025 10:28 PM	Chrome HTML Docu...	2 KB	
	Mrunali Katta DATA 236 - Homework 01	2/5/2025 10:07 PM	Microsoft Word Doc...	14 KB	

2. Build the docker image using docker build -t <name> .

- Docker Image Successfully Built using

docker build -t myblogapp_hw1 .

```
Command Prompt
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mruna>docker --version
Docker version 27.2.0, build 3ab4256

C:\Users\mruna>cd /d "D:\Distributed systems\hw1\DS_HW1"

D:\Distributed systems\hw1\DS_HW1>docker build -t myblogapp_hw1 .
[+] Building 9.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 522B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerrignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/nginx:latest@sha256:91734281c0ebfc6f1aea979cffeed5079cfef86228a71cc6f1f46a228cde
=> => resolve docker.io/library/nginx:latest@sha256:91734281c0ebfc6f1aea979cffeed5079cfef86228a71cc6f1f46a228cde
=> => sha256:84t8as195l5342dbcf9d9ce46691b61a4ca318164a44 1.40kB / 1.40kB
=> => sha256:6c78b0ba132d228d04680a4dc7bd67a21f2e30f6fc032b56480cae756efe14 1.21kB / 1.21kB
=> => sha256:976e8fb625dd4b8451fa2d71524f61103d32dc1af6b1ddb70b0ea848a3e373 403B / 403B
=> => sha256:c558df2179491b5490a7b895e422ffccddcfaec6c045d499670948d32a2ab5784 955B / 955B
=> => sha256:24ff42a0d907c9f6e59a6ce8ad8691f8b72ffad5e1b86cb632f5cfa 627B / 627B
=> => sha256:e19db8451abd4b9c6dc54aaec5c9fd8917d5267d249a977b95dab00ce960e4bf 43.95MB / 43.95MB
=> => sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7bcc76205dd6acf3260 28.21MB / 28.21MB
=> => extracting sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7bcc76205dd6acf3260
=> => extracting sha256:e19db8451abd4b9c6dc54aaec5c9fd8917d5267d249a977b95dab00ce960e4bf
=> => extracting sha256:24ff42a0d907c9f6e59a6ce8ad8691f8b72ffad5e1b86cb632f5cfa
=> => extracting sha256:c558df2179491b5490a7b895e422ffccddcfaec6c045d499670948d32a2ab5784
=> => extracting sha256:976e8fb625dd4b8451fa52d01524f61103d32dc1af6b1dd70b0ea848a3e373
=> => extracting sha256:6c78b0ba132d228d04680a4dc7bd67a21f2e30f6fc032b56480dcae756efe14
=> => extracting sha256:84cadef77a8319515342dbc0f9d9ce4bbddd2d95f93646091b61a4ca318164a44
=> [internal] load build context
=> => transferring context: 5.44kB
=> [2/5] WORKDIR /usr/share/nginx/html
=> => rm -rf ./*
=> [4/5] COPY index_hw1.html /usr/share/nginx/html/index.html
=> [5/5] COPY blog_hw1.js /usr/share/nginx/html/blog_hw1.js
=> => exporting to image
=> => exporting layers
=> => exporting manifest sha256:d61e76e6c3375ab61a5c138d2fd3a30ced4071e90f61cbbe99016ba3b8079b02
=> => exporting config sha256:0643f692935ce00a28cd40bbb2e641dcf4ed4d7a0403957b7b10bc6582a6e530
=> => exporting attestation manifest sha256:319302111a769a57f482dd491fde10dad9ca7915a16d861f6cb4556250cc25
=> => exporting manifest list sha256:89da25a61d9d7a73a86579266aece486dd57d5a89dbb181b2b30aa9aba150c1b
=> => naming to docker.io/library/myblogapp_hw1:latest
=> => unpacking to docker.io/library/myblogapp_hw1:latest

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview

D:\Distributed systems\hw1\DS_HW1>
```

3. Run the docker container locally docker run -d -p 8080:80 <name>

docker run -d -p 8080:80 myblogapp_hw1

```
D:\Distributed systems\hw1\DS_HW1>docker run -d -p 8080:80 myblogapp_hw1
c8adfba2e57c48221a9911f474ff4f1e5d9b368b1c8ff249785b6264445550dc
```

```
D:\Distributed systems\hw1\DS_HW1>
```

- the container is running

docker ps

```
D:\Distributed systems\hw1\DS_HW1>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c8adfba2e57c myblogapp_hw1 "/docker-entrypoint..." 5 minutes ago Up 5 minutes 0.0.0.0:8080->80/tcp suspicious_leavitt

D:\Distributed systems\hw1\DS_HW1>
```

Docker desktop application

The screenshot shows the Docker Desktop application interface. On the left, there's a sidebar with options: Containers (selected), Images, Volumes, Builds, Docker Scout, and Extensions. The main area is titled 'Containers' and shows a table with one item:

Name	Container ID	Image	Port(s)	CP	Actions
suspicious_leavitt	c8adfba2e57c	myblogapp_hw	8080:80	c7	[Edit, Stop, Delete]

Below the table, it says 'Showing 1 item'. At the bottom, there's a 'Walkthroughs' section with links to 'Multi-container applications', 'Containerize your application', and a 'Terminal' tab.

4. Verify by visiting <http://localhost:8080>

The screenshot shows a web browser window with a form for creating a blog post. The URL bar shows 'localhost:8080'. The form fields include:

- Blog Title***: Placeholder text 'Enter the title of your blog'.
- Author Name***: Placeholder text 'Enter your name'.
- Author Email Address***: Placeholder text 'Enter your email'.
- Content***: Placeholder text 'Write your content here...'.
- Category***: A dropdown menu currently set to 'Technology'.
- I agree to the terms and conditions
-

Pushing the docker image to ECR

- Creating an ECR Repository called 'myblogapp_hw1'

The screenshot shows the 'Create private repository' wizard. In the 'General settings' step, a repository name '137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1' is entered. Under 'Image tag mutability', 'Mutable' is selected. In the 'Encryption' step, it's noted that encryption settings can't be changed once created. AES-256 is selected as the encryption type. A note at the bottom indicates 'Image scanning settings - deprecated'. At the bottom right are 'Cancel' and 'Create' buttons.

The screenshot shows the ECR console with the 'Private registry' section selected. A green success message 'Successfully created myblogapp_hw1' is displayed. The 'Private repositories' list shows one item: 'myblogapp_hw1' (Repository URI: 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1), created on February 06, 2025, at 00:44:11 (UTC-08), with 'Mutable' tag immutability and 'AES-256' encryption type. The left sidebar includes links for Amazon Elastic Container Registry, Private registry (selected), Public registry, ECR public gallery, Amazon ECS, Amazon EKS, Getting started, and Documentation.

Authenticated Docker to ECR

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin  
137068252487.dkr.ecr.us-east-1.amazonaws.com
```

```
Command Prompt + X
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mruna>aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 137068252487.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
C:\Users\mruna>
```

Docker Image Tagged

```
docker tag myblogapp_hw1:latest 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1:latest
```

```
Command Prompt + X
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mruna>aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 137068252487.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
C:\Users\mruna>
C:\Users\mruna>docker tag myblogapp_hw1:latest 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1:latest
C:\Users\mruna>
```

- Push the Docker Image to AWS ECR

```
docker push 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1:latest
```

```
C:\Users\mruna>docker push 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1:latest
The push refers to repository [137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1]
1e03dd033147: Already exists
26c43257ce17: Layer already exists
6c78b0ba1a32: Layer already exists
e19db8451adb: Pushed
84cade77a831: Layer already exists
c29f5b76f736: Pushed
24ff42a0d907: Layer already exists
c558df217949: Layer already exists
4f4fb700ef54: Layer already exists
976e8f6b25dd: Layer already exists
b61765e9c6e2: Layer already exists
a8e654ce6927: Layer already exists
latest: digest: sha256:89da25a61d9d7a73a86579266aece486dd57d5a89dbb181b2b30aa9aba150c1b size: 856
C:\Users\mruna>
```

the image is present in AWS ECR

The screenshot shows the AWS ECR console interface. On the left, there's a navigation sidebar with sections for Amazon Elastic Container Registry, Private registry (Repositories, Features & Settings), Public registry (Repositories, Settings), ECR public gallery, Amazon ECS, and Amazon EKS. Below these are links for Getting started and Documentation. The main content area is titled "Private repositories (1)" and shows a table with one entry: "myblogapp_hw1". The table columns are Repository name, URI, Created at, Tag immutability, and Encryption type. The URI is listed as 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1. The Created at field shows February 06, 2025, 00:44:11 (UTC-08). The Tag immutability is set to Mutable, and the Encryption type is AES-256. At the top right, there are buttons for View push commands, Delete, Actions, and Create repository. A search bar at the top of the main content area allows searching by repository substring.

the image is present inside the repository

The screenshot shows the AWS ECR console interface, similar to the previous one but with a different URL. It displays the "Images (3)" section for the repository "myblogapp_hw1". A message at the top states: "Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag." The main table lists three images: "latest" (Image Index, pushed at February 06, 2025, 00:57:19 UTC-08, size 72.18 MB), and two unnamed images (both pushed at February 06, 2025, 00:57:18 UTC-08, sizes 72.18 MB and 0.00 MB respectively). For each image, there are "Copy URI" and "sha256" download links. The table has columns for Image tag, Artifact type, Pushed at, Size (MB), Image URI, and Digest. At the top right of the table, there are buttons for Delete, Details, Scan, and View push commands. A search bar is also present at the top of the main content area. The left sidebar is identical to the first screenshot, showing the same navigation options and links.

Deploy to AWS ECS

- Create an ECS Cluster called '**myblogappcluster_hw1**'

The screenshot shows the AWS Elastic Container Service (ECS) Clusters page. On the left, there's a sidebar with links like 'Clusters', 'Namespaces', 'Task definitions', and 'Account settings'. The main area has a green header bar stating 'Cluster myblogappcluster_hw1 has been created successfully.' Below it, a table lists the cluster details: Cluster 'myblogappcluster_hw1' (0 services, 0 tasks running), 0 EC2 container instances, CloudWatch monitoring (Default), and Capacity provider strategy (No default found). A 'View cluster' button is at the top right.

Creating a new task

The screenshot shows the 'Create new task definition' page. The sidebar on the left includes 'Task definitions' under 'Amazon Elastic Container Service'. The main form shows a green header bar saying 'Cluster myblogappcluster_hw1 has been created successfully.' The 'Task definition configuration' section has a 'Task definition family' input field containing 'myblogapptask_hw1'. The 'Infrastructure requirements' section is expanded, showing 'AWS Fargate' selected as the launch type. The 'OS, Architecture, Network mode' section shows 'Linux/X86_64' selected for both operating system and network mode. The 'Task size' section shows 'CPU' set to '.25 vCPU' and 'Memory' set to '5 GB'. At the bottom, a 'Task roles - conditional' section is partially visible.

us-east-1.console.aws.amazon.com/ecs/v2/create-task-definition?region=us-east-1

Amazon Elastic Container Service

- Clusters
- Namespaces
- Task definitions**
- Account settings

Install AWS Copilot

Amazon ECR

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

Container - 1 Info

Container details

Name: myblogappcontainer_hw1

Image URI: 137068252487.dkr.ecr.us-east-1.amazonaws.com/myblogapp_hw1:latest

Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

Essential container

Yes

Private registry

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

Private registry authentication

Port mappings

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	Protocol	Port name	App protocol
80	TCP	container-port-protocol	HTTP

[Add port mapping](#)

Read only root file system

When this parameter is turned on, the container is given read-only access to its root file system.

Read only

Resource allocation limits - conditional

Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
1 in vCPU	1	3 in GB	1 in GB

Environment variables - optional

Environment variables

CloudShell Feedback

Search

1:19 AM 2/6/2025

us-east-1.console.aws.amazon.com/ecs/v2/task-definitions/myblogapptask_hw1/1/containers

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

us-east-1.console.aws.amazon.com/ecs/v2/task-definitions/myblogapptask_hw1/1/containers

Amazon Elastic Container Service

- Clusters
- Namespaces
- Task definitions**
- Account settings

Install AWS Copilot

Amazon ECR

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

CloudShell Feedback

Search

1:19 AM 2/6/2025

Cluster myblogapptask_hw1 has been created successfully.

Task definition successfully created myblogapptask_hw1:1 has been successfully created. You can use this task definition to deploy a service or run a task.

myblogapptask_hw1:1

[View cluster](#)

[View task definition](#)

[Deploy](#)

[Actions](#)

[Create new revision](#)

Overview

ARN: arn:aws:ecs:us-east-1:137068252487:task-definition/myblogapptask_hw1:1	Status: ACTIVE	Time created: February 06, 2025 at 01:20 (UTC-8:00)	App environment: Fargate
Task role: -	Task execution role: ecsTaskExecutionRole	Operating system/Architecture: Linux/X86_64	Network mode: awsvpc
Fault injection: Turned off			

[Containers](#)

[JSON](#)

[Task placement](#)

[Volumes \(0\)](#)

[Requires attributes](#)

[Tags](#)

Task size

Task CPU: 256 units (0.25 vCPU)

Task CPU maximum allocation for containers: 256 units (0.25 vCPU)

Task memory: 512 MiB (0.5 GB)

Task memory maximum allocation for container memory reservation: 512 MiB (0.5 GB)

CloudShell Feedback

Search

1:20 AM 2/6/2025

- [Created a service ‘myblogappservice_hw1’ in AWS ECS using the Fargate launch type](#)

The screenshot shows two consecutive steps in the AWS Elastic Container Service 'Create service' wizard:

Compute Configuration Step:

- Environment:** Existing cluster: myblogappcluster_hw1.
- Compute configuration (advanced):**
 - Capacity provider strategy:** Selected 'Capacity provider strategy'.
 - Base: FARGATE, Weight: 0.
 - Add capacity provider button.
 - Launch type:** Selected 'Launch tasks directly without the use of a capacity provider strategy.'
- Capacity provider strategy info:** Select either your cluster default capacity provider strategy or select the custom option to configure a different strategy.
 - Use cluster default (No default capacity provider strategy configured for this cluster).
 - Use custom (Advanced) (Selected).
- Capacity provider:** FARGATE, Weight: 1.
- Platform version:** LATEST.

Deployment Configuration Step:

- Deployment configuration:**
 - Application type:** Service (Selected).
 - 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 (Selected). Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.
 - Family:** myblogapptask_hw1, Revision: 1 (LATEST).
 - Service name:** myblogappservice_hw1.
 - Service type:** Replica (Selected). Place and maintain a desired number of tasks across your cluster.
 - Desired tasks:** 1.
 - Availability Zone rebalancing:** Turn on Availability Zone rebalancing (Selected). Amazon ECS automatically detects Availability Zone imbalances in task distributions across an ECS service, and evenly redistributes ECS service tasks across Availability Zones.
 - Deployment options:**
 - Deployment failure detection:** Info.

The screenshot shows the AWS Elastic Container Service (ECS) console. On the left, there's a sidebar with various AWS services like Clusters, Namespaces, Task definitions, and Account settings. The main area displays a success message: "Task definition successfully created" and "myblogapptask_hw1 has been deployed successfully." Below this, the "myblogappcluster_hw1" cluster overview is shown with sections for Services, Tasks, and CloudWatch monitoring. A table shows one active task under the "Draining" service. The "Services" tab is selected, showing a list of services including "myblogapppservice_hw1". The status of this service is "Active" with 1 task running. The bottom of the page includes a toolbar with CloudShell, Feedback, and other AWS icons, along with system information like the date and time.

Was unable to access the application. Was getting error “This site can’t be reached” so had to add a new rule in order to access the application.

The screenshot shows the AWS EC2 Security Groups console. The left sidebar lists various EC2 services like Instances, AMIs, and Network & Security. The main area shows the details for the security group "sg-051ba9ed2ee178617 - default". Under the "Inbound rules" tab, a new rule is being added for port 80. The rule is defined as follows:

Name	Security group rule ID	Type	Protocol	Port range	Source	Description
sgr-051114caf7a4ed9a	-	All traffic	All	All	sg-051ba9ed2ee178617	-
-	sgr-05cfaf8cb1b912bc07	HTTP	TCP	80	0.0.0.0/0	-

Successfully accessed application for public IP address 52.207.242.243

HW1-Mrunali Katta Not secure 52.207.242.243

Blog Title

Blog Title*
great book

Author Name*
Percy

Author Email Address*
audi@123

Content*
great great great
great great

Category*
Lifestyle

I agree to the terms and conditions

Publish Blog

HW1-Mrunali Katta Not secure 52.207.242.243

Blog Title

Blog Title*
great book

Author Name*
Percy

Author Email Address*
audi@123

Content*
great great great
great great

Category*
Lifestyle

I agree to the terms and conditions

Publish Blog

52.207.242.243 says

Blog saved successfully!

OK

Blog Title

Blog Title*
Enter the title of your blog

Author Name*
Enter your name

Author Email Address*
Enter your email

Content*
Write your content here...

Category*
Technology

I agree to the terms and conditions

Publish Blog

The developer tools console shows the following log entries:

```

Running Ginger Widget v2.11.318
Running Ginger Widget v2.11.318
JSON Data: {"blogTitle": "great book", "authorName": "Percy", "authorEmail": "audi@123", "blogContent": "great great great great great great", "category": "Travel", "submissionDate": "2025-02-06T10:03:00.086Z"}
Title: great book, Email: audi@123
Saving blog to server...
Blog titled "great book" saved successfully!
Total blogs submitted: 1
widget is disabled

```

Delete ECS Service

The AWS ECS service 'myblogappservice_hw1' is selected for deletion. A confirmation dialog box is displayed:

Delete myblogappservice_hw1 service?

Delete myblogappservice_hw1 permanently? This action cannot be undone.
Any CloudFormation stacks associated with services will also be deleted.

The service must be scaled down to zero before it can be deleted.

Alternatively, you can use the force delete service option to have Amazon ECS scale the service down on your behalf before deleting it.

Force delete

To confirm deletion, enter **delete** in the text input field.

delete

Cancel **Delete**

us-east-1.console.aws.amazon.com/ecs/v2/clusters/myblogappcluster_hw1/services?region=us-east-1

Amazon Elastic Container Service > Clusters > myblogappcluster_hw1 > Services

Successfully deleted myblogappservice_hw1

myblogappcluster_hw1

Last updated: February 06, 2025 at 02:14 (UTC-8:00)

Update cluster Delete cluster

Cluster overview

ARN: arn:aws:ecs:us-east-1:137068252487:cluster/myblogappcluster_hw1 Status: Active

CloudWatch monitoring: Default

Registered container instances: -

Services

Draining: Active 1

Tasks

Pending: -

Running: 1

Services Tasks Infrastructure Metrics Scheduled tasks Configuration Tags

Services (0) info

Filter services by value: Any launch type: Any service type: No services

Create

Tell us what you think

This screenshot shows the AWS Elastic Container Service (ECS) Cluster Overview page for the cluster 'myblogappcluster_hw1'. A green success message at the top indicates that the service 'myblogappservice_hw1' has been successfully deleted. The cluster status is shown as 'Active'. The 'Services' section shows one active task. The 'Tasks' section shows one pending task. The 'Registered container instances' section is currently empty. The 'Services' tab is selected, and the 'Info' button is visible. Below the main cluster details, there is a table for filtering services by various parameters like launch type and service type. A large orange 'Create' button is prominently displayed.

Deleted ECR repository

us-east-1.console.aws.amazon.com/ecr/private-registry/repositories?region=us-east-1

Amazon ECR > Private registry > Repositories

Private repositories (1)

myblogapp_hw1

View push commands Delete Actions Create repository

Delete myblogapp_hw1

Are you sure you want to delete myblogapp_hw1? This will also delete any container images and artifacts in this repository.

Cancel Delete

This screenshot shows the AWS ECR Private Registry Repositories page. It displays a single repository named 'myblogapp_hw1'. The 'Actions' dropdown menu is open, and the 'Delete' option is selected, opening a confirmation dialog box. The dialog box asks if the user is sure they want to delete the repository, noting that it will also delete any container images and artifacts. The 'Cancel' and 'Delete' buttons are visible. The left sidebar shows navigation options for the ECR public gallery, Amazon ECS, and Amazon EKS, along with documentation links.

us-east-1.console.aws.amazon.com/ecr/private-registry/repositories?region=us-east-1

Amazon ECR > Private registry > Repositories

successfully deleted myblogapp_hw1

Private repositories

No repositories

View push commands Delete Actions Create repository

This screenshot shows the AWS ECR Private Registry Repositories page after the deletion of the repository 'myblogapp_hw1'. A green success message at the top indicates the deletion was successful. The 'Private repositories' section now shows 'No repositories', indicating that no repositories were found. The 'Actions' dropdown menu is still open, and the 'Delete' option is selected. The left sidebar shows navigation options for the ECR public gallery, Amazon ECS, and Amazon EKS, along with documentation links.

Deleted ECS Cluster

The screenshot shows the AWS Elastic Container Service (ECS) console. A modal dialog box titled "Delete myblogappcluster_hw1" is open, asking for confirmation to delete the cluster. It states: "Deleting the cluster also deletes the CloudFormation stack Infra-ECS-Cluster-myblogappclusterhw1-6858d0e9". Below this, it says "After deletion, the cluster transitions to the INACTIVE state. Clusters with an INACTIVE status might remain discoverable in your account for a period of time." A text input field contains the phrase "delete myblogappcluster_hw1". At the bottom of the dialog are "Cancel" and "Delete" buttons.

The screenshot shows the same ECS cluster page after the deletion process. The cluster status is now listed as "Inactive". The "Services" section shows 0 services. The "Tasks" section shows 0 pending tasks. The "CloudWatch monitoring" and "Registered container instances" sections are also empty.

The screenshot shows the ECS console with the "Clusters" section. The list is empty, displaying the message "No clusters" and "No clusters to display". The "Create cluster" button is visible at the top right of the list area.