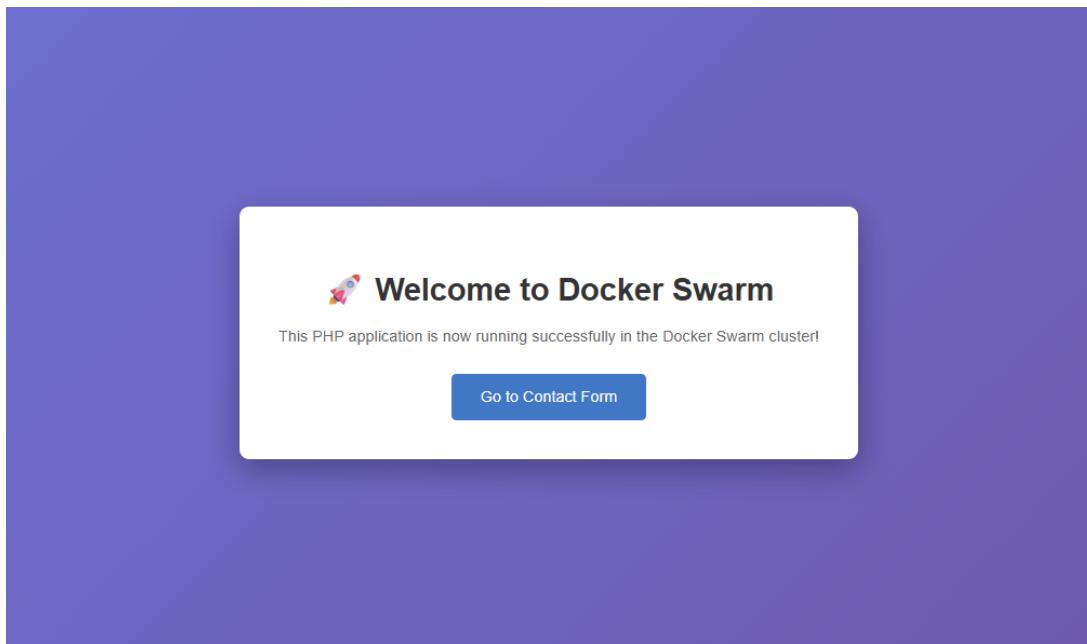


# Docker Swarm

---



**GitHub Repo:** <https://github.com/91maxore-hub/docker-swarm-app>

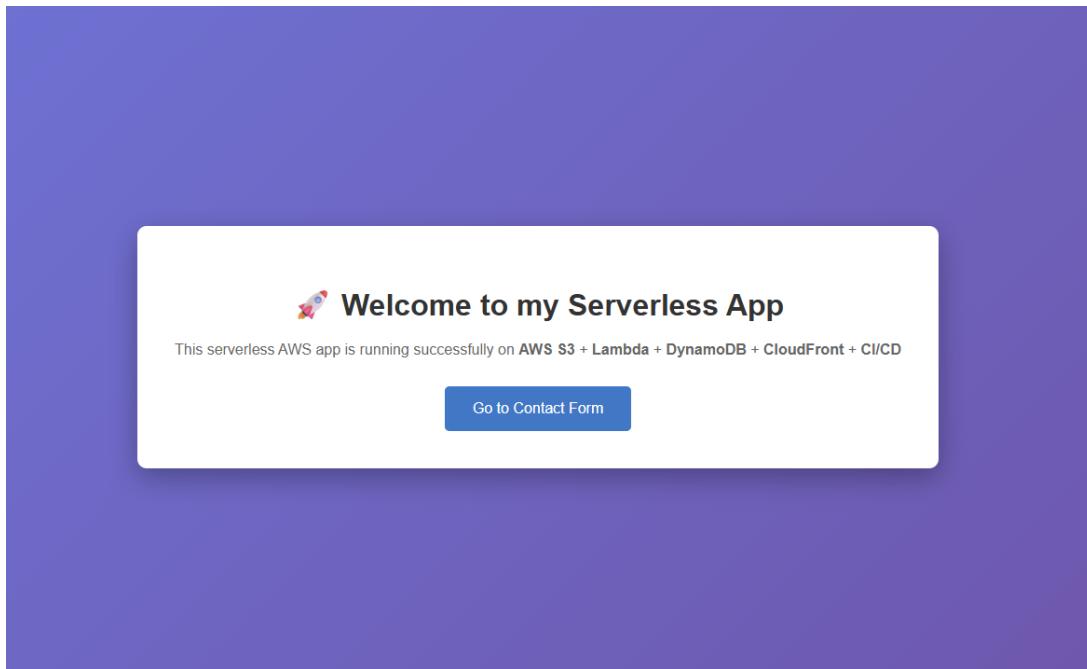
**<https://wavy.se>**

**Serverless App**

---

**GitHub Repo:** <https://github.com/91maxore-hub/serverless-app>

**<https://d3vjy5bvefx3w.cloudfront.net>**



# Docker Swarm

---

I detta projekt har jag byggt en skalbar och robust miljö för en webbapplikation med **Docker Swarm** på AWS. Miljön består av tre virtuella EC2-servrar, där en fungerar som manager och två som worker-noder. Applikationen, som är utvecklad med HTML, PHP och CSS, körs i tre separata containrar – en på varje server – vilket ger hög tillgänglighet och enkel skalning.

För att hantera inkommande trafik och säkerställa säkra anslutningar har jag implementerat **Traefik** som reverse proxy med stöd för HTTPS. För övervakning och visualisering av klustret används **Docker Visualizer**, vilket ger en tydlig överblick över vilka containrar som körs på vilka noder. Dessutom har jag kopplat CI/CD via **GitHub Actions**, vilket gör att uppdateringar av applikationen automatiskt byggs och distribueras till klustret.

Denna lösning visar hur containerteknologi och molninfrastruktur kan kombineras för att skapa en flexibel, skalbar och lättunderhållens webbmiljö, samtidigt som den säkerställer säkerhet, pålitlighet och tydlig översikt över klustrets status.

Noterbart är att i detta projekt har jag utnyttjat följande molntjänster från AWS:

- **EC2 (Elastic Compute Cloud):** Tre virtuella servrar används för att köra Docker Swarm – en som manager och två som worker-noder.
- **GitHub Actions:** För CI/CD, vilket möjliggör automatiska bygg och deployment av webbapplikationen.

Tillsammans skapar dessa tjänster en skalbar, flexibel och säker miljö för webbapplikationen.

# Komponentöversikt och Syfte

---

Komponent	Beskrivning	Användningsområde	Kommentar
<b>EC2-servrar</b>	Virtuella servrar i AWS som utgör infrastrukturen för Docker Swarm-klustret	Körning av applikationens containrar och Swarm-noder	En server som manager, två som workers för skalbarhet och redundans
<b>Docker Swarm</b>	Containerorkesteringssystem som hanterar distribution av containrar	Säkerställer att applikationen körs i flera noder	Gör applikationen skalbar och tillgänglig även vid noder som går ner
<b>Applikationscontainrar</b>	Containeriserad webbapplikation (HTML, PHP, CSS)	Kör själva webbapplikationen på Swarm-servrarna	En container per server för hög tillgänglighet
<b>Traefik</b>	Reverse proxy och load balancer med HTTPS-stöd	Hantering av inkommande trafik och säkra anslutningar	Automatiserar certifikat via HTTPS och styr trafiken till rätt container
<b>GitHub CI/CD</b>	Automatiserat bygg- och deployflöde	Uppdateringar och deployment av applikationen	Säkerställer att nya versioner distribueras snabbt och pålitligt
<b>Docker Visualizer</b>	Grafiskt verktyg som visar status och fördelning av containrar i Swarm	Övervakning och visualisering av Swarm-klustret	Hjälper till att se vilka containrar som körs på vilka noder i realtid

# Regler för säkerhetsgruppen docker-swarm-sg

---

Tillämpning / Resurser	Tillåtna portar	Protokoll	Syfte
EC2-servrar i Swarm-klustret	22	TCP	SSH-åtkomst för administration
EC2-servrar i Swarm-klustret	80	TCP	HTTP-trafik till webbapplikationen
EC2-servrar i Swarm-klustret	443	TCP	HTTPS-trafik till webbapplikationen
EC2-servrar i Swarm-klustret	4789	UDP	Overlay Network för Swarm-tjänster
EC2-servrar i Swarm-klustret	7946	TCP	Swarm intern kommunikation
EC2-servrar i Swarm-klustret	7946	UDP	Swarm intern kommunikation
EC2-servrar i Swarm-klustret	2377	TCP	Swarm management trafik mellan manager och workers
EC2-servrar i Swarm-klustret	8080	TCP	Traefik – reverse proxy med dashboard
EC2-servrar i Swarm-klustret	8081	TCP	Docker Visualizer dashboard

## Mappstruktur

---

Katalog / Fil	Typ	Beskrivning
<b>docker-swarm-app/</b>	Mapp	Rotmappen för hela projektet.
── <b>Dockerfile</b>	Fil	Bygger Docker-imagen och definierar miljö/beroenden.
── <b>index.html</b>	Fil	Huvudsidan för webbapplikationen.
── <b>contact_form.html</b>	Fil	Sida med kontaktformulär.
── <b>process_contact_form.php</b>	Fil	PHP-script som hanterar formulärdata.
── <b>style.css</b>	Fil	CSS-stilmall för webbplatsen.
── <b>.github/workflows/</b>	Mapp	Mapp för GitHub Actions workflows.
── <b>deploy.yml</b>	Fil	Workflow som hanterar CI/CD och deployment.

# Provisionera Amazon EC2-server

Denna guide beskriver hur man provisionerar Amazon EC2-instanser som ska ingå i ett Docker Swarm-kluster. Målet är att skapa en stabil och skalbar miljö med en **Swarm Manager** och två **Swarm Workers**. EC2-instanserna kommer att konfigureras med nödvändig nätverksåtkomst, säkerhetsgrupper och grundläggande systemkrav för att stödja containerorkestrering med Docker Swarm.

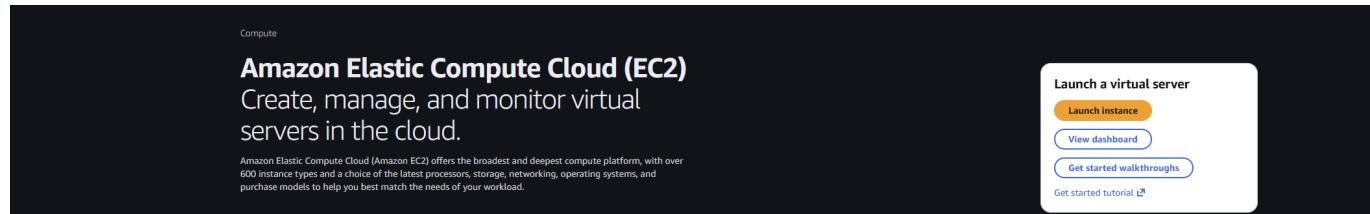
## Steg 1: Bege dig till aws.amazon.com

The screenshot shows the AWS Console Home page. At the top, there's a navigation bar with the AWS logo, a search bar, and account information. Below the navigation bar, the 'Console Home' section is visible. It includes a sidebar with 'Recently visited' services like EC2, CloudFront, S3, Lambda, etc. The main area has several cards: 'Applications' (0), 'Welcome to AWS' (with sections for Getting started with AWS, Training and certification, and AWS Builder Center), 'AWS Health' (with Open issues, Scheduled changes, and Other notifications), and 'Cost and usage' (showing current month cost of \$0.34, forecasted month end data unavailable, and savings opportunities). A sidebar on the left also lists Services, Features, Resources, Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials.

## Steg 2: Ange EC2 i sökrutan och välj "EC2 - Virtual Servers in the Cloud"

The screenshot shows the AWS EC2 service page. On the left, there's a sidebar with 'Services', 'Features', 'Resources', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area features a 'Services' section with a large card for 'EC2 Virtual Servers in the Cloud' (with a star icon) and smaller cards for 'EC2 Image Builder' (A managed service to automate build, customize and deploy OS images) and 'Recycle Bin' (Protect resources from accidental deletion). Below this is a 'Features' section with cards for 'EC2 Instances' (CloudWatch feature), 'EC2 Resource Health' (CloudWatch feature), and 'Dashboard' (EC2 feature). A 'Show more' link is located at the top right of the features section.

### Steg 3: Välj "Launch Instance"



### Steg 4: Ange ett namn för din server, operativsystem (AMI), instanstyp, samt skapa SSH-nyckel för säker åtkomst.

**Launch an instance** [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**Name and tags** [Info](#)

Name  
swarm-manager [Add additional tags](#)

**Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

**Recents** [Quick Start](#)

<a href="#">Amazon Linux</a> 	<a href="#">macOS</a> 	<a href="#">Ubuntu</a> 	<a href="#">Windows</a> 	<a href="#">Red Hat</a> 	<a href="#">SUSE Linux</a> 	<a href="#">Debian</a> 
----------------------------------	---------------------------	----------------------------	-----------------------------	-----------------------------	--------------------------------	----------------------------

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Amazon Linux 2023 kernel-6.1 AMI  
ami-0870af38096a5355b (64-bit (x86), uefi-preferred) / ami-041f2b319c071bcb8 (64-bit (Arm), uefi)  
Virtualization: hvm ENA enabled: true Root device type: ebs

**Description**

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.9.20251110.1 x86\_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username	<a href="#">Verified provider</a>
64-bit (x86)	uefi-preferred	ami-0870af38096a5355b	2025-11-08	ec2-user	<a href="#">Verified provider</a>

**Instance type** [Info](#) | [Get advice](#)

**Instance type**

t3.small  
Family: t3 2 vCPU 2 GiB Memory Current generation: true On-Demand Windows base pricing: 0.0412 USD per Hour  
On-Demand Linux base pricing: 0.0228 USD per Hour On-Demand Ubuntu Pro base pricing: 0.0263 USD per Hour  
On-Demand SUSE base pricing: 0.0538 USD per Hour On-Demand RHEL base pricing: 0.0516 USD per Hour

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

**Summary**  

Number of instances | [Info](#)  
1

**Software Image (AMI)**  
Amazon Linux 2023 AMI 2023.9.2... [read more](#)  
ami-0870af38096a5355b

**Virtual server type (instance type)**  
t3.small

**Firewall (security group)**  
New security group

**Storage (volumes)**  
1 volume(s) - 8 GiB

[Launch instance](#) [Preview code](#)

**Steg 5: Välj sedan säkerhetsgruppen (docker-swarm-sg) som ansvarar för vilka portar som ska användas för vårt Docker Swarm-kluster. Resten kan lämnas som det är.**

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - **required**

swarm-manager-key [Create new key pair](#)

▼ Network settings [Info](#) [Edit](#)

Network | [Info](#)  
vpc-04a930fdf061e6a90

Subnet | [Info](#)  
No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)  
Enable

Firewall (security groups) | [Info](#)  
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group  Select existing security group [Compare security group rules](#)

Common security groups | [Info](#)

Select security groups

docker-swarm-sg sg-05f60262028697eb7 [X](#)  
VPC: vpc-04a930fdf061e6a90

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Configure storage [Info](#) [Advanced](#)

1x  GiB [gp3](#) Root volume, 3000 IOPS, Not encrypted

[Add new volume](#)

[Click refresh to view backup information](#) [Edit](#)  
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems [Edit](#)

► Advanced details [Info](#)

**Steg 6: Gå sedan längst ner till Advanced details -> User data och klistra in följande:**

```
#!/bin/bash
dnf update -y
dnf install -y docker
systemctl enable --now docker
usermod -aG docker ec2-user
```

**User data - optional** | [Info](#)

Upload a file with your user data or enter it in the field.

 [Choose file](#)

```
#!/bin/bash
dnf update -y
dnf install -y docker
systemctl enable --now docker
usermod -aG docker ec2-user
```

 User data has already been base64 encoded

**Steg 6: Du får sedan en kort översikt över EC2-servern längst upp till höger. Välj "Launch instance".**

**▼ Summary****Number of instances** | [Info](#)**Software Image (AMI)**Amazon Linux 2023 AMI 2023.9.2...[read more](#)

ami-0870af38096a5355b

**Virtual server type (instance type)**

t3.small

**Firewall (security group)**

docker-swarm-sg

**Storage (volumes)**

1 volume(s) - 8 GiB

[Cancel](#)**Launch instance** [Preview code](#)

- Repetera nu likadant för **swarm-worker-1** och **swarm-worker-2**

## Steg 7: Du bör nu se en översikt som nedan för samtliga EC2-servrar som kommer används i vårt Docker Swarm-kluster.

Instances (3) <a href="#">Info</a>											
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	
<input type="checkbox"/>	swarm-worker-1	i-0122663740f818169	<span>Running</span>	t3.small	<span>3/3 checks passed</span>		eu-west-1a	ec2-54-170-222-173.eu...	54.170.222.173	-	
<input type="checkbox"/>	swarm-worker-2	i-064a3b5ef76da8f27	<span>Running</span>	t3.small	<span>3/3 checks passed</span>		eu-west-1a	ec2-52-214-90-15.eu...	52.214.90.15	-	
<input type="checkbox"/>	swarm-manager	i-0e76978203457f787	<span>Running</span>	t3.small	<span>3/3 checks passed</span>		eu-west-1a	ec2-34-246-185-128.eu...	34.246.185.128	-	

## Skapandet av ett Docker Hub-repository

Innan vi initierar Docker Swarm-klustret är det viktigt att vi förbereder den applikations-image som klustret ska använda. Eftersom min app är PHP-baserad behöver projektet först paketeras i en Docker-image och publiceras på Docker Hub, så att Swarm-noderna kan dra ned samma version av imagen oavsett vilken nod som kör tjänsten.

När initieringen av Docker Swarm-klustret senare är klar kommer vi att kunna deploya stacken direkt från denna image. Men för att detta ska fungera måste vi först skapa ett repository på Docker Hub som ska lagra och distribuera Docker-imagen. I mitt fall döpte jag detta repository till **docker-swarm-app** (se bilden nedan).

Därefter behöver vi skapa en **Dockerfile** som använder en PHP-image med inbyggd webbserver, vi kommer att använda **php:8.2-apache**, och som kopierar in alla projektets filer. Denna image byggs sedan lokalt och pushas upp till Docker Hub så att den kan användas av hela Swarm-klustret vid deployment.

The screenshot shows the Docker Hub interface for the user '91maxore'. On the left, there's a sidebar with options like 'Repositories', 'Hardened Images', 'Collaborations', 'Settings', 'Default privacy', and 'Notifications'. The main area is titled 'Repositories' and shows a list of repositories within the '91maxore' namespace. One repository, '91maxore/docker-swarm-app', is listed. To its right, there are filters for 'Search by repository name', 'All content', and a 'Create a repository' button. Below the search bar, there are columns for 'Name', 'Last Pushed', 'Contains', 'Visibility', and 'Scout'. The repository details show it was last pushed 26 minutes ago, contains an image, is public, and is inactive.

# Följ stegen nedan för att skapa ett **Docker Hub-repository**

## Steg 1: Logga in på Docker Hub:

Gå till <https://hub.docker.com/repositories/ditt-användarnamn>

## Steg 2: Navigera till dina repositories:

Du kommer direkt till listan över repositories under ditt konto.

The screenshot shows the Docker Hub user profile interface for the user '91maxore'. At the top, there's a blue profile icon, the username '91maxore', and the text 'Docker Personal'. Below this is a navigation bar with several options: 'Repositories' (which is highlighted in grey), 'Hardened Images', 'Collaborations', 'Settings', 'Default privacy', and 'Notifications'. The main area below the navigation bar lists the user's repositories.

## Steg 3: Skapa ett nytt repository genom att klicka på "Create a Repository" längst bort till höger.

The screenshot shows the 'Repositories' list for the '91maxore' namespace. It includes a search bar, a dropdown menu for 'All content', and a prominent blue 'Create a repository' button. A single repository entry is listed: '91maxore/docker-swarm-app', which was last pushed 26 minutes ago, contains an image, is public, and is inactive.

Name	Last Pushed	Contains	Visibility	Scout
91maxore/docker-swarm-app	26 minutes ago	IMAGE	Public	Inactive

## Steg 4: Fyll i repository-information:

- **Repository Name:** Ange ett namn för ditt repo, t.ex. `docker-swarm-app` kommer att bli `ditt-användarnamn/docker-swarm-app` senare när du ska bygga och pusha Docker-image
- **Visibility:** Välj om ditt repo ska vara **Public** eller **Private**
- **Description:** Lägg till en kort beskrivning om vad repot innehåller
- Klicka på "**Create**"

[Repositories](#) / [Create](#)

### Create repository

Repository Name \*

 Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

**Visibility**

Using 0 of 1 private repositories. [Get more](#)

**Public**  Appears in Docker Hub search results

**Private**  Only visible to you

[Cancel](#) [Create](#)

## Skapandet av Dockerfile

Jag skapade därefter en Dockerfile som använder PHP 8.2 med Apache och kopierar in mina applikationsfiler från projektmappen.

**Kortfattat:** en Dockerfile är en fil som beskriver hur ens Docker-image ska byggas.

**Dockerfile** (docker-swarm-app/Dockerfile) gör följande:

1. Använder officiell PHP 8.2 med Apache som grundimage.
2. Aktiverar Apache-modulen `mod_rewrite` för att möjliggöra URL-omskrivningar.
3. Kopierar alla applikationsfiler från projektmappen till Apache:s webbroot (`/var/www/html/`).
4. Exponerar port 80 så att webbservern kan ta emot HTTP-trafik.

# Byggandet av Docker Image och uppladdning till Docker Hub

Nu är det dags att gå igenom stegen för att paketera projektet i en Docker-image och publicera den på Docker Hub

## Steg 1: Byggandet av Docker Image

Jag använde terminalen i **Visual Studio Code** och angav följande kommando utifrån min projektmapp (där appens samtliga filer finns) för att bygga mina applikations-filer till en Docker-image och ge den en tagg.

**91maxore** = användarnamn

**docker-swarm-app** = repo på Docker Hub

```
docker build -t 91maxore/docker-swarm-app:latest .
```

## Steg 2: Logga in på Docker Hub

Logga in på Docker Hub via terminalen:

```
docker login
```

- Angav mitt användarnamn och lösenord som jag använder till Docker Hub.

## Steg 3: Pusha Docker-image till Docker Hub

När imagen är byggd och du är inloggad, pusha imagen till Docker Hub med:

```
docker push 91maxore/docker-swarm-app:latest
```

Detta pushar min nyskapade Docker-image till Docker Hub och är redo för användning.

Nu ligger den på Docker Hub:

🔗 <https://hub.docker.com/repository/docker/91maxore/docker-swarm-app>

- När man skapar eller uppdaterar en Docker Swarm-service skickar manager-noden instruktionen till alla workers. Om ens worker inte har den image-version som behövs, hämtar den automatiskt (pull) imagen från Docker Hub eller den angivna registry. Man behöver alltså inte göra pull manuellt på varje worker.

# Initiering av Docker Swarm

För att initiera ett Docker Swarm-kluster, anslut till din swarm-manager via SSH och kör kommandot nedan som startar klustret och utser noden till manager.

## Steg 1: Börja med att SSHa in på vår nyskapade EC2 genom att ange:

```
ssh -i ~/Downloads/swarm-manager-key.pem ec2-user@34.246.185.128
```

**Notera att du får ändra sökvägen till din SSH-nyckel, samt den publika IP-adress till din EC2-instans**

```
Max Oredson@DESKTOP-SN4B6V8 MINGW64 ~/Downloads
$ ssh -i swarm-manager-key.pem ec2-user@34.246.185.128
,
  _#
 ~\_ ##### Amazon Linux 2023
 ~~ \#####\_
 ~~ \|##|
 ~~ \#/ __ https://aws.amazon.com/linux/amazon-linux-2023
 ~~ V~' '-->
 ~~ / \
 ~~ /_/
 ~~ /_/
Last login: Sun Nov 16 13:23:30 2025 from 80.217.195.172
[ec2-user@ip-172-31-23-10 ~]$
```

## Steg 2: Initiera Docker Swarm-kluster på manager genom att köra följande:

```
docker swarm init --advertise-addr 34.246.185.128
```

**Notera att du får byta ut IP-adressen mot den publika IP som din swarm-manager har**

- Kopiera nu kommandot med dess token som skrivs ut för att ansluta våra övriga worker-noder till Docker Swarm-klustret, du bör få något som ser ut så här:

```
docker swarm join --token SWMTKN-1-
1qb2x87bw5wx75p5opwk8qqqoy513l2piskjrcze19acy8da3c-ec79bgjfs3q8doy3cpw3306js
172.31.23.10:2377
```

# Anslutning och hantering av Swarm-workernoder

## Steg 1: Kör nu följande för att ansluta swarm-worker-1 och swarm-worker-2 till Docker Swarm-klustret:

```
docker swarm join --token SWMTKN-1-  
1qb2x87bw5wx75p5opwk8qqoy513l2piskjrcze19acy8da3c-ec79bgjfs3q8doy3cpw3306js  
172.31.23.10:2377
```

- Notera att du får byta ut token.

## Steg 2: Verifera sedan på swarm-manager att worker-noderna har lagts till i klustret genom att ange:

```
docker node ls
```

- Du bör då se något liknande:

```
[ec2-user@ip-172-31-23-10 ~]$ docker node ls  
ID           HOSTNAME   STATUS  AVAILABILITY  MANAGER STATUS  ENGINE VERSION  
i8xm23liainh482l06otg9ts  ip-172-31-16-170.eu-west-1.compute.internal  Ready  Active  
so3r924ftotfuias57fz17q8s9 * ip-172-31-23-10.eu-west-1.compute.internal  Ready  Active  Leader  25.0.13  
z2io6pjcurxxdn4iq896zun8e  ip-172-31-30-160.eu-west-1.compute.internal  Ready  Active  
[ec2-user@ip-172-31-23-10 ~]$ |
```

**Detta bekräftar att vårt Docker Swarm-kluster är nu skapad med 1 manager och 2 workers.**

### Steg 3: Skapa Docker Stack-fil

- En **docker-stack.yml** behövs för att definiera hela applikationens tjänster, nätverk och inställningar på ett och samma ställe, så att Docker Swarm kan deployna och hantera allt som en komplett stack.
- På swarm-manager, skapa stackfilen **docker-stack.yml** och klistra in följande:

```
version: "3.8"
services:
  web:
    image: 91maxore/docker-swarm-app:latest
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
      update_config:
        parallelism: 1
        delay: 5s
    ports:
      - "80:80"
    networks: [webnet]

networks:
  webnet:
    driver: overlay
```

#### Beskrivning:

- Kör **91maxore/docker-swarm-app** som en Swarm-tjänst med 3 repliker.
- Startar om repliker automatiskt vid fel.
- Exponerar tjänsten på port 80.
- Använder overlay-nätverk (**webnet**) så att tjänsten kan kommunicera med andra tjänster i klustret.

### Steg 4: Distribuera Docker Swarm-stacken genom att ange följande:

```
sudo docker stack deploy -c docker-stack.yml docker-swarm-app
```

- **docker-swarm-app** blir namnet på stacken eftersom vår stack kommer i slutändan innehålla flera tjänster: **web**, **viz** och **traefik**
- Samtliga tjänster kommer befinnas sig på följande benämningar: **docker-swarm-app\_web**, **docker-swarm-app\_viz** och **docker-swarm-app\_traefik**

**Steg 5: Vi kan nu kontrollera statusen för varje instans av webbapplikationen, se på vilken nod de körs och verifiera att alla tre repliker fungerar som de ska. Detta görs med följande kommando:**

```
sudo docker service ps docker-swarm-app_web
```

- Webbapplikationen kör nu stabilt och som förväntat på alla tre noder i Swarm-klustret, vilket bekräftar att deploymenten fungerar korrekt.
- Kort sagt: bilden visar var och hur min web-app körs inom Swarm-klustret
- Det fungerar på samma sätt genom att senare ange **docker-swarm-app\_viz** för **Docker Vizualizer** och **docker-swarm-app\_traefik** för **Traefik**.

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
11nfs1kb6e55	docker-swarm-app_web.1	91maxore/docker-swarm-app:latest	ip-172-31-16-170.eu-west-1.compute.internal	Running	Running 3 hours ago
uzauums3ylql	└ docker-swarm-app_web.1	91maxore/docker-swarm-app:latest	ip-172-31-16-170.eu-west-1.compute.internal	Shutdown	Shutdown 3 hours ago
8dr111ygxdo	└ docker-swarm-app_web.1	91maxore/docker-swarm-app:latest	ip-172-31-16-170.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
lkppzu3ch5qb	└ docker-swarm-app_web.1	91maxore/docker-swarm-app:latest	ip-172-31-16-170.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
6ae01knmtwlv	└ docker-swarm-app_web.1	91maxore/docker-swarm-app:latest	ip-172-31-16-170.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
72ache0sr6ue	docker-swarm-app_web.2	91maxore/docker-swarm-app:latest	ip-172-31-30-160.eu-west-1.compute.internal	Running	Running 3 hours ago
urgbnh8wkzs0	└ docker-swarm-app_web.2	91maxore/docker-swarm-app:latest	ip-172-31-30-160.eu-west-1.compute.internal	Shutdown	Shutdown 3 hours ago
vo7u6g2ordcv	└ docker-swarm-app_web.2	91maxore/docker-swarm-app:latest	ip-172-31-30-160.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
u863pxssfrfz	└ docker-swarm-app_web.2	91maxore/docker-swarm-app:latest	ip-172-31-30-160.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
x7nit75ka45x	└ docker-swarm-app_web.2	91maxore/docker-swarm-app:latest	ip-172-31-30-160.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
zdiwjgj1ly33q	docker-swarm-app_web.3	91maxore/docker-swarm-app:latest	ip-172-31-23-10.eu-west-1.compute.internal	Running	Running 3 hours ago
zgh5gjmzrn38	└ docker-swarm-app_web.3	91maxore/docker-swarm-app:latest	ip-172-31-23-10.eu-west-1.compute.internal	Shutdown	Shutdown 3 hours ago
ry3t9j0z13w	└ docker-swarm-app_web.3	91maxore/docker-swarm-app:latest	ip-172-31-23-10.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
urkp6upvb8qs	└ docker-swarm-app_web.3	91maxore/docker-swarm-app:latest	ip-172-31-23-10.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago
pr1omrlugnvq	└ docker-swarm-app_web.3	91maxore/docker-swarm-app:latest	ip-172-31-23-10.eu-west-1.compute.internal	Shutdown	Shutdown 4 hours ago

**Steg 7: Vi kan även verifiera att swarm-worker 1 och swarm-worker 2 tagit del av samma docker-image och att webbapplikationen är replikerad utöver alla 3 noder med följande kommando:**

```
docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
qpwqobw27qe2	docker-swarm-app_traefik	replicated	1/1	traefik:v2.11.3	*:80->80/tcp, *:443->443/tcp, *:8080->8080/tcp
4ko4udt8rfm	docker-swarm-app_viz	replicated	1/1	dockersamples/visualizer:stable	*:8081->8080/tcp
v9h3cbe9drc0	docker-swarm-app_web	replicated	3/3	91maxore/docker-swarm-app:latest	

- Som du kan se kör mitt Docker Swarm-kluster även **Traefik** för reverse proxy och HTTPS-hantering, och detta kommer jag att gå igenom mer detaljerat senare i guiden.
- Dessutom kör mitt Docker Swarm-kluster även **Docker Visualizer** för att enkelt kunna se noder, tjänster och containrar i realtid, och detta kommer jag att gå igenom mer detaljerat i nästa steg.

# Docker Vizualiser

**Docker Visualizer** är ett verktyg som ger en grafisk översikt över ditt Docker Swarm-kluster. Det visar alla noder, både manager och worker, samt vilka containrar som körs på respektive nod i realtid. Vizualizer är ett utmärkt sätt att snabbt förstå klustrets struktur, övervaka distributionen av tjänster och kontrollera att skalning och repliker fungerar som förväntat.

## **Steg 1: Börja med att addera följande till docker-stack.yml som vi skapade tidigare för att lägga till Vizualiser som tjänst till vår stack:**

- Eftersom **Docker Vizualiser** är en tjänst listar vi även den under **services** som nedan.

```
services:  
  viz:  
    image: dockersamples/visualizer:stable  
    deploy:  
      placement:  
        constraints:  
          - node.role == manager  
    ports:  
      - "8081:8080"                      # Visualizer-webbgränssnitt  
    volumes:  
      - /var/run/docker.sock:/var/run/docker.sock  
    networks:  
      - webnet  
  
networks:  
  webnet:  
    driver: overlay
```

## **Beskrivning (Docker Vizualizer)**

- Kör Visualizer som en Swarm-tjänst på manager-noden.
- Mountar Docker-socket för att kunna läsa klustrets noder och containrar.
- Exponerar Visualizer på port 8081
- Använder overlay-nätverk så den kan kommunicera med andra tjänster om det behövs.

## **Steg 2: Deploya återigen stacken genom att köra detta på manager-noden:**

```
docker stack deploy -c docker-stack.yml docker-swarm-app
```

### Steg 3: Kontrollera att tjänsten körs:

```
docker service ps docker-swarm-app_viz
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR	PORTS
ruei9nai1gi	docker-swarm-app_viz.1	dockersamples/visualizer:stable	ip-172-31-23-10.eu-west-1.compute.internal	Running	Running	2 days ago	

### Steg 4: Öppna Visualizer

- Surfa in till managers publika IP följt av port 8081, alltså i mitt fall: <http://34.246.185.128:8081>
- Du ser nu alla noder och containrar i ditt Swarm-kluster visuellt.



## Sammanfattningsvis:

- **Docker Visualizer** körs som en separat service på manager, exponerar ett webbläsargränssnitt och visar i realtid alla noder och containrar i Swarm-klustret.

# Traefik

---

**Traefik** är en dynamisk reverse proxy och lastbalanserare designad för **Docker Swarm**.

I min miljö körs **Traefik** på managern, där den automatiskt upptäcker alla tjänster och repliker som körs ute på klustrets noder. Detta gör att min applikation, oavsett om dess containrar körs på manager-noden eller på mina två workers, alltid nås via en central och smart styrd ingångspunkt.

Ett av huvudskälen att använda Traefik i min kluster är dess automatiserade hantering av HTTPS via Let's Encrypt. Med ACME-integration bygger Traefik själv ut, förnyar och lagrar certifikat utan att du behöver göra något manuellt — vilket ger en trygg och självgenererande säkerhetslösning på både port 80 och 443.

Utöver detta fungerar Traefik som en dynamisk reverse proxy, där routning uppdateras i realtid när tjänster skalas upp eller ned. All trafik lastbalanseras automatiskt över mina tre repliker av [web](#)-tjänsten och fördelar jämnt oavsett vilken nod de körs på.

Med Traefiks dashboard, som jag exponerar på port 8080, får jag dessutom en tydlig visuell överblick över routers, tjänster, certifikat och trafikflöden i realtid — perfekt för att verifiera att lastbalansering, HTTPS och routning fungerar som tänkt.

- Traefik är därför en komplett och självgående lösning för att hantera reverse proxy, trafikstyrning och automatiska HTTPS-certifikat i mitt Docker Swarm-kluster.

**Steg 1: Börja med att återigen addera följande till docker-stack.yml som vi skapade tidigare för att lägga till Traefik som tjänst till vår stack:**

- Eftersom Traefik är en tjänst listar vi även den under **services** som nedan.

```
services:  
  traefik:  
    image: traefik:v2.11.3  
    command:  
      - "--providers.docker=true"  
      - "--providers.docker.swarmMode=true"  
      - "--providers.docker.exposedbydefault=false"  
      - "--entrypoints.web.address=:80"  
      - "--entrypoints.websecure.address=:443"  
      - "--entrypoints.web.http.redirects.entrypoint.to=websecure"  
      - "--entrypoints.web.http.redirects.entrypoint.scheme=https"  
      - "--certificatesresolvers.myresolver.acme.httpchallenge=true"  
      - "--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web"  
      - "--certificatesresolvers.myresolver.acme.email=91maxore@afe.molndal.se"  
      - "--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json"  
      - "--api.insecure=true"  
      - "--log.level=DEBUG"  
    ports:  
      - "80:80"  
      - "443:443"  
      - "8080:8080"  
    volumes:  
      - /var/run/docker.sock:/var/run/docker.sock:ro  
      - traefik_letsencrypt:/letsencrypt  
    deploy:  
      placement:  
        constraints:  
          - node.role == manager  
      restart_policy:  
        condition: on-failure  
    networks:  
      - webnet  
  
networks:  
  webnet:  
    driver: overlay  
  
volumes:  
  traefik_letsencrypt:
```

## Beskrivning (Traefik)

- Kör Traefik som en Swarm-tjänst placerad på manager-noden.
- Använder Docker-socket för att automatiskt upptäcka tjänster och repliker i Swarm-klustret.
- Fungerar som en reverse proxy och lastbalanserare för alla tjänster som har Traefik-labels.
- Hanterar HTTPS automatiskt med Let's Encrypt via ACME.
- Exponerar HTTP (80), HTTPS (443) och Traefik Dashboard (8080) på manager-noden.
- Dirigerar all trafik från HTTP → HTTPS med automatisk omdirigering.
- Lagrar certifikat i en persistent volym för att undvika att certifikat återskapas vid omstart.
- Körs i overlay-nätverk ([webnet](#)) för att kunna nå alla tjänster i Swarm-klustret.

### Steg 2: Deploya återigen stacken genom att köra detta på manager-noden:

```
docker stack deploy -c docker-stack.yml docker-swarm-app
```

### Nu när vi konfigurerat alla tre tjänster inom stacken så kommer stacken starta:

- Traefik på managern
- Min app med 3 repliker fördelade över noderna inom Docker Swarm-klustret
- Docker Visualizer på managern

### Steg 3: Kontrollera att tjänsten körs

```
docker service ps docker-swarm-app_traefik
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
gv35hk62z5rf	docker-swarm-app_traefik.1	traefik:v2.11.3	ip-172-31-23-10.eu-west-1.compute.internal	Running	Running 2 days ago
nxoiv7zbeozi	\_ docker-swarm-app_traefik.1	traefik:v2.11.3	ip-172-31-23-10.eu-west-1.compute.internal	Shutdown	Shutdown 2 days ago

### Steg 4: Kontrollera att samtliga tjänster körs

```
docker service ls
```

### Detta borde visa att samtliga tjänster inom stacken vi konfigurerat körs och är replikerade.

- **Traefik:** Reverse proxy med HTTPS via Let's Encrypt, dashboard på port 8080.
- **Web:** Applikation med flera repliker, lastbalanseras av Traefik.
- **Docker Visualizer:** Visar klustrets noder och containrar i realtid på port 8081.

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
qpwqobw27qe2	docker-swarm-app_traefik	replicated	1/1	traefik:v2.11.3	*:80->80/tcp, *:443->443/tcp, *:8080->8080/tcp
4ko4udt8wrfm	docker-swarm-app_viz	replicated	1/1	dockersamples/visualizer:stable	*:8081->8080/tcp
v9h3cbe9drco	docker-swarm-app_web	replicated	3/3	91maxore/docker-swarm-app:latest	

## Steg 5: Öppna Traefiks Dashboard

- Surfa in till managers publika IP följt av port 8080, alltså i mitt fall: <http://34.246.185.128:8080>
- Du ser nu alla routers, tjänster och trafikflöden i ditt Swarm-kluster visuellt via Traefiks dashboard.
- Notera att jag konfigurerat **wavvy.se** domänen via Loopia så den konfigurationen är inte inkluderad i denna guide.

The screenshot shows the Traefik 2.11.3 dashboard with the title "træfik 2.11.3". The top navigation bar includes links for "Dashboard", "HTTP", "TCP", "UDP", and "Plugins". Below the navigation, the "Entrypoints" section is visible, showing three main entry points: "TRAEFIK" (port :8080), "WEB" (port :80), and "WEBSECURE" (port :443).

**Steg 6: För att HTTPS ska fungera korrekt behöver vi konfigurera Traefik-labels på web-tjänsten så att den kan rauta trafiken till min domän. Detta gör vi genom att ersätta nuvarande konfiguration för web i docker-stack.yml med följande:**

```
web:
  image: 91maxore/docker-swarm-app:latest
  deploy:
    replicas: 3
    restart_policy:
      condition: on-failure
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.web.rule=Host(`wavvy.se`)"
      - "traefik.http.routers.web.entrypoints=websecure"
      - "traefik.http.routers.web.tls=true"
      - "traefik.http.routers.web.tls.certresolver=myresolver"
      - "traefik.http.services.web.loadbalancer.server.port=80"
  networks:
    - webnet
```

- Ersätt därmed denna med den tidigare web-konfiguration i stack-filen vi använde oss av.
- Dessa labels gör att Traefik vet vilken domän trafiken ska routas till, vilka entrypoints som ska användas, att TLS ska aktiveras, och vilken certifikatlösare som ska hantera Let's Encrypt-certifikaten.
- Traefik-labels konfigurerar web-tjänsten så att HTTPS fungerar och all HTTP-trafik automatiskt dirigeras till HTTPS.

## Steg 7: Verifiera HTTPS

- Surfa nu in till **https://wavvy.se**
- Vi kan därmed granska att appen fungerar som den ska med HTTPS/SSL. Du kan även se på bilden att anslutningen är säker och att certifikatet är giltigt.



## Traefik

- Tar emot trafiken
- Skapar certifikat automatiskt via Let's Encrypt
- Lastbalanserar över mina 3 web-repliker
- Dirigerar all HTTP → HTTPS

### Sammanfattningsvis:

- Traefik körs som en separat service på manager, exponerar ett webbläsargränssnitt och visar i realtid alla routers, tjänster och trafikflöden i Swarm-klustret.

## Beskrivning av de tre tjänsterna i min stack:

- **docker-swarm-app\_web (Applikationen)** Webbapplikationen hanterar allt innehåll, som HTML och PHP, och körs som flera repliker som fördelas mellan manager och worker-noder i Swarm-klustret. Det gör att applikationen kan skalias och distribueras över flera noder, vilket ger hög tillgänglighet och jämn belastning utan att påverka användarupplevelsen.
- **docker-swarm\_viz (Docker Visualizer)** Visualizer är ett grafiskt verktyg som visar Swarm-klustret i realtid, inklusive manager- och worker-noder samt alla containrar. Det gör det enkelt att övervaka hur tjänster och repliker distribueras över klustret, vilket ger snabb insikt i klustrets status och hjälper till att upptäcka problem med belastning eller distribution.
- **docker-swarm\_traefik (Traefik)** Traefik är en reverse proxy och lastbalanserare som körs i Swarm på manager-noden och som hanterar inkommande trafik. Den hanterar automatiskt routing av trafik till mina tjänster, distribuerar trafiken till min web-applikation, skapar och förnyar HTTPS-certifikat via Let's Encrypt, och ger en visuell översikt över routers, tjänster och trafikflöden via dashboarden.

# Automatiserad deployment med GitHub Actions (CI/CD)

## Steg 1: Skapa ett GitHub-repo

- Bege dig över till ditt GitHub-konto
- Skapa ett nytt repo på GitHub genom att klicka på "**New repository**"
- Jag döpte min till **docker-swarm-app2** enbart för att demonstrera
- Välj **Public** eller **Private** beroende på behov.
- Klicka på "**Create repository**"

**Create a new repository**

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).  
Required fields are marked with an asterisk (\*).

**1 General**

Owner \*      Repository name \*

91maxore-hub      docker-swarm-app2

docker-swarm-app2 is available.

Great repository names are short and memorable. How about [psychic-rotary-phone](#)?

Description

0 / 350 characters

**2 Configuration**

Choose visibility \*      Public

Choose who can see and commit to this repository

Add README      Off

READMEds can be used as longer descriptions. [About READMEs](#)

Add .gitignore      No .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

Add license      No license

Licenses explain how others can use your code. [About licenses](#)

Connect GitHub Apps      1 app selected

Connect this repository to apps 91maxore-hub is subscribed to

**Create repository**

Efter att du skapat ditt repo kommer du bli hänvisad till följande instruktioner som du kan se nedan på bilden.  
Kopiera **Quick setup**-länken och följ vidare instruktionerna på mitt nästa steg.

The screenshot shows a GitHub repository page for 'docker-swarm-app2'. At the top, there are buttons for Pin, Watch (0), Fork (0), and Star (0). Below the header, there are sections for 'Set up GitHub Copilot' and 'Add collaborators to this repository'. The main section is titled 'Quick setup — if you've done this kind of thing before'. It provides instructions for cloning the repository using 'Set up in Desktop' (with options for HTTPS or SSH) or by copying the URL (<https://github.com/91maxore-hub/docker-swarm-app2.git>). It also suggests creating a new file or uploading an existing one, and recommends including a README, LICENSE, and .gitignore. Below this, there are sections for '...or create a new repository on the command line' and '...or push an existing repository from the command line', each containing a code block with terminal commands.

## Steg 2: Bege dig till projektmappen

- Öppna terminalen och bege dig till projektmappen där appens filer ligger på din lokala dator ex.

```
cd ~/docker-swarm-app
```

## Steg 3: Initiera ett nytt Git-repo

```
git init
```

## Steg 4: Lägg till README.md

```
git add README.md
```

## Steg 5: Commit:a ändringarna:

```
git commit -m "CI/CD Pipeline"
```

**Steg 6: Anslut lokalt repo till GitHub:**

```
git remote add origin git@github.com:91maxore-hub/docker-swarm-app.git
```

- Ersätt med quick-setup länken vi kopierade tidigare.

**Steg 7: Pusha till GitHub**

```
git push -u origin main
```

**Steg 8: Sen varje gång du gör ändringar i en eller flera filer kan du enkelt ange följande kommando för att pusha samtliga ändringar i filer till GitHub:**

```
git add . && git commit -m "CI/CD Pipeline" && git push origin main
```

- Detta kommer endast pusha ändrade filer till GitHub och därifrån utgöra en CI/CD-automatiserings deployment så att Docker-imagen alltid håller sig uppdaterad, och därav samma med container-hosten som hostar appen.

**Jag har nu initierat GitHub-repot och det är redo att användas för CI/CD-deployments.**

**Steg 9. Skapa GitHub Actions workflow**

- Nästa steg är att skapa en **deploy.yml** för upprätthålla en CI/CD.
- Så skapa mappen och workflow-filen enligt strukturen som nedan:

```
mkdir -p .github/workflows
```

**Workflow-filen** (.github/workflows/deploy.yml) gör följande:

- Checkoutar koden från GitHub-repot.
- Sätter upp Docker Buildx för multi-platform builds.
- Loggar in på Docker Hub med GitHub Secrets.
- Bygger Docker-imagen för applikationen.
- Pushar imagen till Docker Hub.
- Ansluter till Swarm-manager via SSH med GitHub Secrets.
- Deployar stacken på Docker Swarm med **docker stack deploy -c docker-stack.yml**, uppdaterar tjänster och rullar ut den nya imagen automatiskt.

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2

      - name: Log in to DockerHub
        uses: docker/login-action@v2
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Build and push Docker image
        run: |
          docker build -t 91maxore/docker-swarm-app:latest .
          docker push 91maxore/docker-swarm-app:latest

      - name: Deploy to Swarm
        uses: appleboy/ssh-action@v0.1.7
        with:
          host: ${{ secrets.SSH_HOST }}
          username: ${{ secrets.SSH_USER }}
          key: ${{ secrets.SSH_PRIVATE_KEY }}
          script: |
            docker stack deploy -c /home/ec2-user/docker-stack.yml docker-swarm-app
```

- Innan vi kan gå vidare med att deploya deploy.yml-filen behöver vi sätta upp lite GitHub Secrets.

# GitHub Secrets-konfigurationer

Repository secrets		New repository secret
Name	Last updated	
DOCKER_PASSWORD	3 days ago	 
DOCKER_USERNAME	3 days ago	 
SSH_HOST	3 days ago	 
SSH_PRIVATE_KEY	3 days ago	 
SSH_USER	3 days ago	 

## GitHub Secrets-tabell

Secret	Beskrivning
DOCKER_USERNAME	Mitt användarnamn på Docker Hub, används för att logga in och pusha images - 91maxore
DOCKER_PASSWORD	Mitt lösenord för Docker Hub
SSH_HOST	IP-adress till Swarm-manager där stacken deployas - 34.246.185.128
SSH_USER	Användarnamnet som används för SSH-anslutningen till manager-noden - ec2-user
SSH_PRIVATE_KEY	Privat SSH-nyckel som matchar en publik nyckel på Swarm-manager för autentisering

# Skapandet av en GitHub Secret

- Öppna ditt repo på GitHub (ex. <https://github.com/91maxore-hub/docker-swarm-app>)
- Navigera till fliken **Settings**
- Navigera till **Secrets and variables** → **Actions**
- Klicka på "**New repository secret**"
- Fyll i:
  - **Name** – t.ex. `SSH_HOST`
  - **Secret** – `34.246.185.128`
- Spara med "**Add secret**"

Enligt bästa praxis ska inga känsliga värden, såsom IP-adresser, domännamn, SSH-nycklar eller e-postadresser etc. hårdkodas i koden. Istället lagras dessa uppgifter säkert som **GitHub Secrets** i repot för att skydda dem från obehörig åtkomst och för att underlätta säker hantering.

## Steg 1: Lägg till workflow och pusha

För att kontrollera att workflow-filen och CI/CD-deployment fungerar korrekt, pusha ändringarna i ett steg:

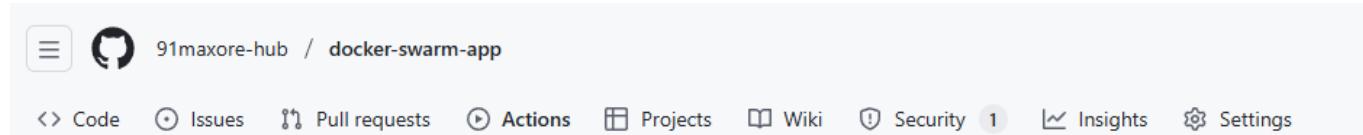
```
git add .github/workflows/deploy.yml && git commit -m "Lägg till GitHub Actions workflow för CI/CD" && git push origin main
```

## Steg 2: Verifiering av CI/CD funktionalitet

Gå till ditt GitHub-repo, exempelvis:

<https://github.com/91maxore-hub/docker-swarm-app>

Navigera sedan till fliken **Actions**



## Om CI/CD är korrekt konfigurerat bör du se att de senaste körningarna är markerade med en grön bok som nedan:

All workflows

Showing runs from all workflows

Filter workflow runs

Help us improve GitHub Actions  
Tell us how to make GitHub Actions work better for you with three quick questions.

Give feedback X

19 workflow runs

	Event	Status	Branch	Actor	...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #19: Commit <a href="#">8314f6e</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 11:45 PM GMT+1 44s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #18: Commit <a href="#">c0c9eec</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 11:45 PM GMT+1 57s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #17: Commit <a href="#">f368100</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 7:11 PM GMT+1 40s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #16: Commit <a href="#">3031ef7</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 3:57 PM GMT+1 41s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #15: Commit <a href="#">df71732</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 2:39 PM GMT+1 47s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #14: Commit <a href="#">2036145</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 2:39 PM GMT+1 44s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #13: Commit <a href="#">555d152</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 2:14 PM GMT+1 34s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #12: Commit <a href="#">2134b79</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 2:11 PM GMT+1 47s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #11: Commit <a href="#">9237c05</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 2:07 PM GMT+1 45s			...
<span>✓</span> CI/CD Pipeline CI/CD Pipeline #10: Commit <a href="#">ba34ec6</a> pushed by <a href="#">91maxore-hub</a>	main	<span>Success</span> Nov 18, 2:05 PM GMT+1 42s			...

Dessutom en **status** som visar **Success**. Exempel på ett lyckat arbetsflöde:

### build-and-push — Success

← CI/CD Pipeline

✓ CI/CD Pipeline #19

Summary

Triggered via push 11 hours ago  
91maxore-hub pushed → [8314f6e](#) main Status Success Total duration 44s Artifacts —

Jobs

✓ build-and-deploy

Run details

Usage

Workflow file

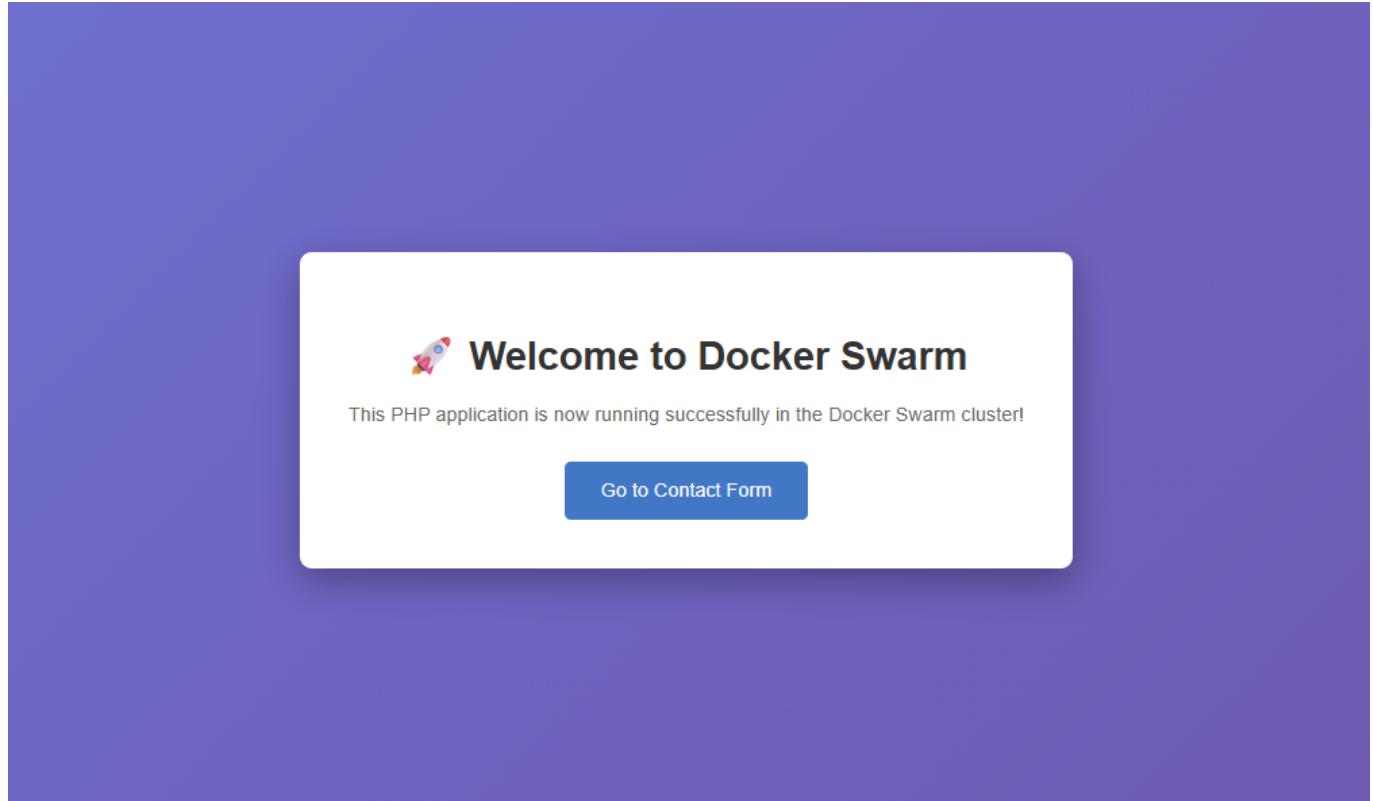
deploy.yml  
on: push

✓ build-and-deploy 39s

## Resultat

Efter att allt var uppsatt och CI/CD-deployment gick igenom kunde jag gå till: [🔗 https://wavvy.se](https://wavvy.se)

Min PHP-app laddas med giltigt SSL-certifikat, automatisk HTTPS och reverse proxy som hanterar trafiken smidigt genom **Traefik**. Allt detta sker helt automatiskt – både deployment och certifikatförflyelse.



# Serverless App

---

I detta projekt har jag byggt en skalbar och kostnadseffektiv serverless-miljö för en webbapplikation på AWS. Applikationen använder **AWS S3** för hosting av statiska filer, **AWS Lambda** för backend-logik, **API Gateway** för att hantera HTTP-förfrågningar och **DynamoDB** för lagring av formulärsvar. För att säkerställa snabb och säker åtkomst till applikationen används **CloudFront** som reverse proxy med stöd för HTTPS.

Applikationen är helt serverlös, vilket innebär att infrastrukturen automatiskt skalar baserat på belastning, utan behov av att hantera servrar eller operativsystem. Detta gör lösningen både flexibel och lättunderhållensamtidigt som den erbjuder hög tillgänglighet och prestanda.

För att underlätta utveckling och deployment har jag implementerat **CI/CD med AWS CodePipeline kopplat till GitHub**, vilket automatiskt bygger och distribuerar nya versioner av applikationen till S3 och Lambda. Detta säkerställer snabb iteration och pålitlig uppdatering av applikationen utan manuella steg.

Denna lösning visar hur serverless-teknologi kan kombineras med molntjänster för att skapa en modern webbmiljö som är skalbar, säker och kostnadseffektiv.

Noterbart är att i detta projekt har jag utnyttjat följande molntjänster från AWS:

- **S3 (Simple Storage Service):** Hosting av statiska filer som HTML, CSS och JavaScript.
- **Lambda:** Serverlös körning av backend-logik för formulärhantering och affärslogik.
- **API Gateway:** Hantering av HTTP-förfrågningar och routing till Lambda-funktioner.
- **DynamoDB:** Lagring av formulärsvar och annan applikationsdata.
- **CloudFront:** Content delivery och reverse proxy med HTTPS för säker och snabb åtkomst.
- **CodePipeline + GitHub:** CI/CD som möjliggör automatiska bygg och deployment av applikationen.

Komponent	Beskrivning	Användningsområde	Kommentar
<b>S3</b>	Lagrar och serverar statiska filer som HTML och CSS	Hosting av frontend	Ger hög tillgänglighet och enkel skalning utan serverhantering
<b>Lambda</b>	Serverlösa funktioner som kör backend	Hantering av formulärdata	Skalar automatiskt baserat på belastning, inga servrar att hantera
<b>API Gateway</b>	Hanterar HTTP-förfrågningar och routear dem till Lambda	Exponering av backendfunktioner som API	Säkerställer HTTP API-kommunikation mellan frontend och Lambda
<b>DynamoDB</b>	Databas för lagring av formulärsvar	Databas för applikationen	Fullt hanterad, serverlös, mycket låg latency
<b>CloudFront</b>	Reverse Proxy med HTTPS	Snabb och säker åtkomst till applikationen	Minskar latens globalt och ger HTTPS-stöd

Komponent	Beskrivning	Användningsområde	Kommentar
<b>CodePipeline + GitHub</b>	CI/CD-pipeline som bygger och deployar applikationen automatiskt	Automatiserad deployment	Säkerställer snabb iteration och pålitlig uppdatering

## Mappstruktur

Katalog / Fil	Typ	Beskrivning
<b>index.html</b>	HTML-fil	Huvudsida för webbapplikationen
<b>contact_form.html</b>	HTML-fil	Sida med kontaktformulär för användare
<b>thankyou.html</b>	HTML-fil	Sida som visas efter att formuläret skickats
<b>style.css</b>	CSS-fil	Stilark som styr utseende och layout för webbapplikationen
<b>contactFormHandler.js</b>	JavaScript-fil	Backend-funktion (AWS Lambda) som hanterar formulärinlämning och sparar data i DynamoDB

## Konfiguration av Amazon S3-bucket

Denna guide beskriver hur man skapar och konfigurerar en Amazon S3-bucket för att hosta statiska webbapplikationsfiler. Målet är att tillhandahålla en högpresterande och skalbar hostingmiljö för HTML-, CSS- och övriga statiska resurser. Bucketen kommer att konfigureras med offentlig läsbehörighet för hosting, samt integreras med CloudFront för snabb distribution och HTTPS-stöd.

### Steg 1: Bege dig till aws.amazon.com

The screenshot shows the AWS Console Home page. At the top, there's a search bar and a navigation bar with links like 'AWS', 'Search', 'Console Home', and 'Account ID: 6694-2073-9197'. Below the navigation is a sidebar with 'Recently visited' services: EC2, CloudFront, S3, Lambda, DynamoDB, API Gateway, CloudWatch, and IAM. To the right of the sidebar are several dashboard widgets:

- Applications (0)**: Shows a table for creating applications in the 'eu-west-1' region.
- Welcome to AWS**: Includes sections for 'Getting started with AWS', 'Training and certification', and 'AWS Builder Center'.
- AWS Health**: Displays 'Open issues 0', 'Scheduled changes 0', and 'Other notifications 0'.
- Cost and usage**: Shows current month costs (\$0.34), forecasted month end costs ('Data unavailable'), and savings opportunities ('Not enabled'). It includes a bar chart of costs over time from June 25 to Nov 25, categorized by service: EC2 - Compute, Virtual Private Cloud, Relational Database Service, Elastic File System, Backup, and Others.

### Steg 2: Ange S3 i sökrutan och välj "S3 - Scalable Storage in the Cloud"

The screenshot shows the AWS Services page with a search bar at the top. On the left, there's a sidebar with links to Services, Features, Resources, Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area has a heading 'Services' and three items listed: 'S3 Scalable Storage in the Cloud', 'S3 Glacier Archive Storage in the Cloud', and 'AWS Snow Family Large Scale Data Transport'. Each item has a star icon to its right.

### Steg 3: Välj Create bucket

The screenshot shows the Amazon S3 storage landing page. It features a large 'Amazon S3' logo and the tagline 'Store and retrieve any amount of data from anywhere'. Below the tagline, it says 'Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.' To the right, there's a 'Create a bucket' button inside a callout box. The text in the box explains that every object in S3 is stored in a bucket and that you need to create a bucket where the objects will be stored.

### Steg 4: Ange ett namn för vår S3-bucket, jag kommer namnge den serverless-bucket-2025

The screenshot shows the 'Create bucket' configuration page. At the top, there's a 'General configuration' section with 'AWS Region' set to 'Europe (Ireland) eu-west-1' and 'Bucket type' set to 'General purpose'. A note says 'Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.' Below this is a 'Bucket name' field containing 'serverless-bucket-2025'. A note says 'Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)'.

Under 'Copy settings from existing bucket - optional', there's a 'Choose bucket' button and a note about prefix format: 'Format: s3://bucket/prefix'.

At the bottom, there's an 'Object Ownership' section with two options: 'ACLs disabled (recommended)' (selected) and 'ACLs enabled'. A note for 'ACLs disabled' says 'All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.' A note for 'ACLs enabled' says 'Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.'

## Steg 5: Gå sedan lite längre ner och bocka ur "Block Public Access settings for this bucket" eftersom vi vill ju komma åt våra filer genom appen

- Bucket Versioning kan aktiveras för att automatiskt behålla tidigare versioner av filer, vilket underlättar återställning vid oavsiktliga ändringar eller borttagningar. Men i detta fall skippar vi det.
- Resten kan lämnas som det är.

**Block Public Access settings for this bucket**

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more ↗](#)

**Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLS)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLS)**  
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

**⚠️ Turning off block all public access might result in this bucket and the objects within becoming public**  
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

**Bucket Versioning**

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more ↗](#)

**Bucket Versioning**

Disable  
 Enable

## Steg 6: Välj slutligen Create bucket

## Steg 7: Du gör nu få en översikt över din nyskapade S3-bucket

**General purpose buckets (2) [Info](#)**

Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
<a href="#">serverless-bucket-2025</a>	Europe (Ireland) eu-west-1	November 16, 2025, 18:07:22 (UTC+01:00)

[Create bucket](#)

## Steg 8: Klicka på "Upload" längst bort till höger

**serverless-bucket-2025 [Info](#)**

[Objects](#) | [Metadata](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

**Objects (4)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory ↗](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more ↗](#)

[Find objects by prefix](#)

[Actions](#) | [Copy S3 URI](#) | [Copy URL](#) | [Download](#) | [Open ↗](#) | [Delete](#) | [Create folder](#) | [Upload](#)

## Steg 9: Välj därefter att ladda upp mappar eller filer. I vårt ändamål kommer vi att ladda upp endast appens filer. Så vi väljer "Add files" följt av "Upload" längst ner

**Files and folders (0)**  
All files and folders in this table will be uploaded.

Name	Type	Size
No files or folders You have not chosen any files or folders to upload.		

**Destination** [Info](#)  
Destination: <S3://serverless-bucket-2025>

▶ **Destination details**  
Bucket settings that impact new objects stored in the specified destination.

▶ **Permissions**  
Grant public access and access to other AWS accounts.

▶ **Properties**  
Specify storage class, encryption settings, tags, and more.

[Cancel](#) [Upload](#)

## Steg 10: Slutligen bör du se en översikt över filerna vi precis laddade upp.

**serverless-bucket-2025** [Info](#)

[Objects](#) [Metadata](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

**Objects (4)**  
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
<a href="#">contact_form.html</a>	html	November 18, 2025, 13:56:57 (UTC+01:00)	2.0 KB	Standard
<a href="#">index.html</a>	html	November 18, 2025, 13:56:57 (UTC+01:00)	821.0 B	Standard
<a href="#">style.css</a>	css	November 18, 2025, 13:56:57 (UTC+01:00)	2.0 KB	Standard
<a href="#">thankyou.html</a>	html	November 18, 2025, 13:56:57 (UTC+01:00)	1.2 KB	Standard

# Konfiguration av Amazon DynamoDB för lagring av formulärsvär

Denna guide beskriver hur man skapar och konfigurerar en Amazon DynamoDB-tabell för att lagra data från webbapplikationens kontaktformulär. Målet är att tillhandahålla en högpresterande, serverlös och skalbar databaslösning som kan hantera varierande trafik utan att behöva hantera servrar.

## Steg 1: Bege dig till <aws.amazon.com>

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' services like EC2, CloudFront, S3, Lambda, DynamoDB, API Gateway, CloudWatch, and IAM. Below that is a 'Welcome to AWS' section with links for 'Getting started with AWS', 'Training and certification', and 'AWS Builder Center'. To the right, there are three main cards: 'AWS Health' (with 0 open issues), 'Cost and usage' (showing current month cost of \$0.34 and a bar chart for forecasted month end), and 'Applications' (empty). At the bottom, there are links to 'Go to myApplications', 'Go to AWS Health', and 'Go to Billing and Cost Management'.

## Steg 2: Ange DynamoDB i sökrutan och välj "DynamoDB - Managed NoSQL Database"

The screenshot shows the AWS Services search results for 'DynamoDB'. The search bar at the top has 'DynamoDB' typed in. On the left, there's a sidebar with links to 'Services', 'Features', 'Resources', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area shows three services: 'DynamoDB' (Managed NoSQL Database), 'Amazon DocumentDB' (Fully-managed MongoDB-compatible database service), and 'Athena' (Serverless interactive analytics service). Each service has a star icon to its right. Below this, there's a 'Features' section with 'Settings' (containing a 'DynamoDB feature'), 'Clusters' (containing a 'DynamoDB feature'), and 'Subnet Groups' (containing a 'DynamoDB feature'). A 'Show more' link is visible on the right side of the features section.

## Steg 3: Välj Create table

The screenshot shows the Amazon DynamoDB landing page. The main heading is 'Amazon DynamoDB' with the subtext 'A fast and flexible NoSQL database service for any scale'. Below this, a small note says 'DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.' On the right, there's a 'Get started' section with the text 'Create a new table to start exploring DynamoDB.' and a 'Create table' button.

## Steg 4: Ange ett namn för vår DynamoDB-databas, jag kommer namnge den ContactFormMessages

- Ange även **id** som Partition key och värdet ska vara **String**

### Create table

**Table details** [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

**Table name**  
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.)

**Partition key**  
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.  
 String

1 to 255 characters and case sensitive.

**Sort key - optional**  
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.  
 String

1 to 255 characters and case sensitive.

**Table settings**

**Default settings**  
The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

**Customize settings**  
Use these advanced features to make DynamoDB work better for your needs.

## Steg 5: Slutligen bör du se en översikt över databasen vi precis skapade

The screenshot shows the AWS DynamoDB 'Tables' page. At the top, there are filters for 'Find tables', 'Filter by tag', and 'Filter by tag value'. The table list shows one item: 'ContactFormMessages' (Active, id (\$), Off, On-demand, On-demand, 5.6 kilobytes, Standard). The table has 1 row and 0 columns. The last update was November 19, 2025, at 11:47 (UTC+1:00).

## Uppsättning av AWS Lambda

Denna guide beskriver hur man skapar och konfigurerar AWS Lambda-funktioner för att hantera backend-logik i webbapplikationen. Målet är att tillhandahålla en skalbar, serverlös miljö där funktioner automatiskt kan exekveras som svar på HTTP-förfrågningar via API Gateway. Lambda-funktionerna kommer att hantera inlämning av formulärdata, validering av inkommande data och lagring i DynamoDB, utan att kräva några underhållskrav på servrar.

## Steg 1: Bege dig till [aws.amazon.com](https://aws.amazon.com)

The screenshot shows the AWS Console Home page. It includes sections for 'Recently visited' services (EC2, CloudFront, S3, Lambda, DynamoDB, API Gateway, CloudWatch, IAM), 'Welcome to AWS' (Getting started with AWS, Training and certification, AWS Builder Center), 'AWS Health' (Open issues, Scheduled changes, Other notifications), and 'Cost and usage' (Current month cost \$0.34, Forecasted month end, Savings opportunities, Cost (\$) chart for Jun 25 to Nov 25).

## Steg 2: Ange Lambda i sökrutan och välj "Lambda - Run code without thinking about servers"

The screenshot shows the AWS Lambda service page. At the top, there's a search bar with the word "Lambda". Below the search bar, there's a sidebar with links to "Services", "Features", "Resources", "Documentation", "Knowledge articles", "Marketplace", "Blog posts", "Events", and "Tutorials". The main content area is titled "Services" and lists three items: "Lambda" (Run code without thinking about servers), "CodeBuild" (Build and Test Code), and "AWS Signer" (Ensuring trust and integrity of your code). There's also a "Show more" link. Below the services section is another titled "Features" which lists "Lambda Insights" (CloudWatch feature), "Object Lambda Access Points" (S3 feature), and "Local processing" (IoT Core feature). There's also a "Show more" link here.

### Steg 3: Navigera till Functions

The screenshot shows the AWS Lambda navigation bar. It has a blue circular icon with three horizontal lines, followed by the word "Lambda" and a right-pointing arrow, and then the word "Functions". The "Functions" tab is highlighted in blue, while the other tabs ("Dashboard" and "Applications") are in grey.

### Lambda



Dashboard

Applications

**Functions**

### Steg 4: Välj Create function längst till höger

The screenshot shows the AWS Lambda functions list. At the top left, it says "Last fetched 50 seconds ago". To the right is a blue circular icon with a white "C" and a downward arrow. Next to it are the words "Actions" and a downward arrow. To the right of that is a large orange button with the text "Create function" in white. At the bottom right of the list area, there are navigation arrows (left, right) and a gear icon.

### Steg 5: Ange ett namn för vår Lambda-function, jag kommer namnge den contactFormHandler

- Resten kan lämnas som det är

**Create function** Info

Choose one of the following options to create your function.

<input checked="" type="radio"/> Author from scratch Start with a simple Hello World example.	<input type="radio"/> Use a blueprint Build a Lambda application from sample code and configuration presets for common use cases.	<input type="radio"/> Container image Select a container image to deploy for your function.
<b>Basic information</b> <p><b>Function name</b> Enter a name that describes the purpose of your function. <input type="text" value="contactFormHandler"/></p> <p><b>Runtime</b> <small>Info</small> Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. <input type="text" value="Node.js 22.x"/> <span style="border: 1px solid #ccc; padding: 2px;">C</span></p> <p><b>Architecture</b> <small>Info</small> Choose the instruction set architecture you want for your function code. <input type="radio"/> arm64 <input checked="" type="radio"/> x86_64</p> <p><b>Permissions</b> <small>Info</small> By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.</p> <p><b>Change default execution role</b></p> <p><b>Execution role</b> Choose a role that defines the permissions of your function. To create a custom role, go to the <a href="#">IAM console</a>. <input checked="" type="radio"/> Create a new role with basic Lambda permissions <input type="radio"/> Use an existing role <input type="radio"/> Create a new role from AWS policy templates</p> <p><small>Role creation might take a few minutes. Do not delete the role or edit the trust or permissions policies in this role.</small></p> <p>Lambda will create an execution role named contactFormHandler-role-vripu5zb, with permission to upload logs to Amazon CloudWatch Logs.</p>		

**Steg 6: Slutligen bör du se en översikt över funktionen (contactFormHandler.js) vi precis skapade**

Functions (1)					
<input type="text"/> Search by attributes or search by keyword					
Function name	Description	Package type	Runtime	Last modified	Actions
<a href="#">contactFormHandler</a>		Zip	Node.js 22.x	23 hours ago	<span style="border: 1px solid #ccc; padding: 2px;">C</span>

**Steg 7: Gå in på den och klistra in följande kod:**

```

const { DynamoDBClient, PutItemCommand } = require("@aws-sdk/client-dynamodb");
const crypto = require("crypto");

const db = new DynamoDBClient({ region: "eu-west-1" });

exports.handler = async (event) => {
    console.log("Incoming event:", JSON.stringify(event));

    if (event.httpMethod === "OPTIONS") {
        return {
            statusCode: 204,
            headers: {
                "Access-Control-Allow-Origin": "https://d3vbjy5bvefx3w.cloudfront.net",
                "Access-Control-Allow-Methods": "POST,OPTIONS",
                "Access-Control-Allow-Headers": "Content-Type",
            },
            body: ""
        };
    }

    let data;
    try {
        data = JSON.parse(event.body || "{}");
    }

```

```
    } catch (e) {
      console.error("JSON parse error:", e);
      return {
        statusCode: 400,
        headers: { "Access-Control-Allow-Origin": "https://d3vbjy5bvefx3w.cloudfront.net" },
        body: JSON.stringify({ error: "Invalid JSON in request" })
      };
    }

    const id = crypto.randomUUID();

    const params = {
      TableName: "ContactFormMessages",
      Item: {
        id: { S: id },
        name: { S: data.name || "UNKNOWN" },
        email: { S: data.email || "UNKNOWN" },
        message: { S: data.message || "EMPTY" },
        createdAt: { S: new Date().toISOString() }
      }
    };

    try {
      console.log("Trying to write to Dynamo:", params);
      await db.send(new PutItemCommand(params));
      console.log("Write SUCCESS");

      return {
        statusCode: 200,
        headers: {
          "Access-Control-Allow-Origin": "https://d3vbjy5bvefx3w.cloudfront.net",
          "Access-Control-Allow-Headers": "Content-Type",
          "Access-Control-Allow-Methods": "POST,OPTIONS",
        },
        body: JSON.stringify({
          success: true,
          id,
          ...data
        })
      };
    }

    } catch (err) {
      console.error("DynamoDB ERROR:", err);

      return {
        statusCode: 500,
        headers: {
          "Access-Control-Allow-Origin": "https://d3vbjy5bvefx3w.cloudfront.net"
        },
        body: JSON.stringify({ error: "Could not save message", details: err.message
      })
    };
  
```

```
}
```

```
};
```

## Beskrivning (Lambda – contactFormHandler.js)

- Körs som en serverlös AWS Lambda-funktion som hanterar inkommande HTTP-förfrågningar via API Gateway.
- Tar emot formulärdata från frontend och validerar JSON-innehållet.
- Genererar ett unikt ID för varje formulärinlämning med `crypto.randomUUID()`.
- Sparar formulärdata (`name`, `email`, `message`, `createdAt`) i DynamoDB-tabellen `ContactFormMessages`.
- Hanterar CORS (Cross-Origin Resource Sharing) för att möjliggöra anrop från frontend-distributionen på CloudFront.
- Notera att jag har min CloudFront-URL för Access-Control-Allow-Origin, alltså den enda domänen som har tillstånd att använda min Lambda-fuction. Jag kommer gå igenom hur man sätter upp API Gateway och CloudFront i nästkommande steg.
- Du kan för tillfället ange S3-Bucket URL för att testa dess funktionalitet men det fungerar på samma sätt eftersom du talar endast om för Lambda-funktionen vilken domän som är tillåten att använda den.

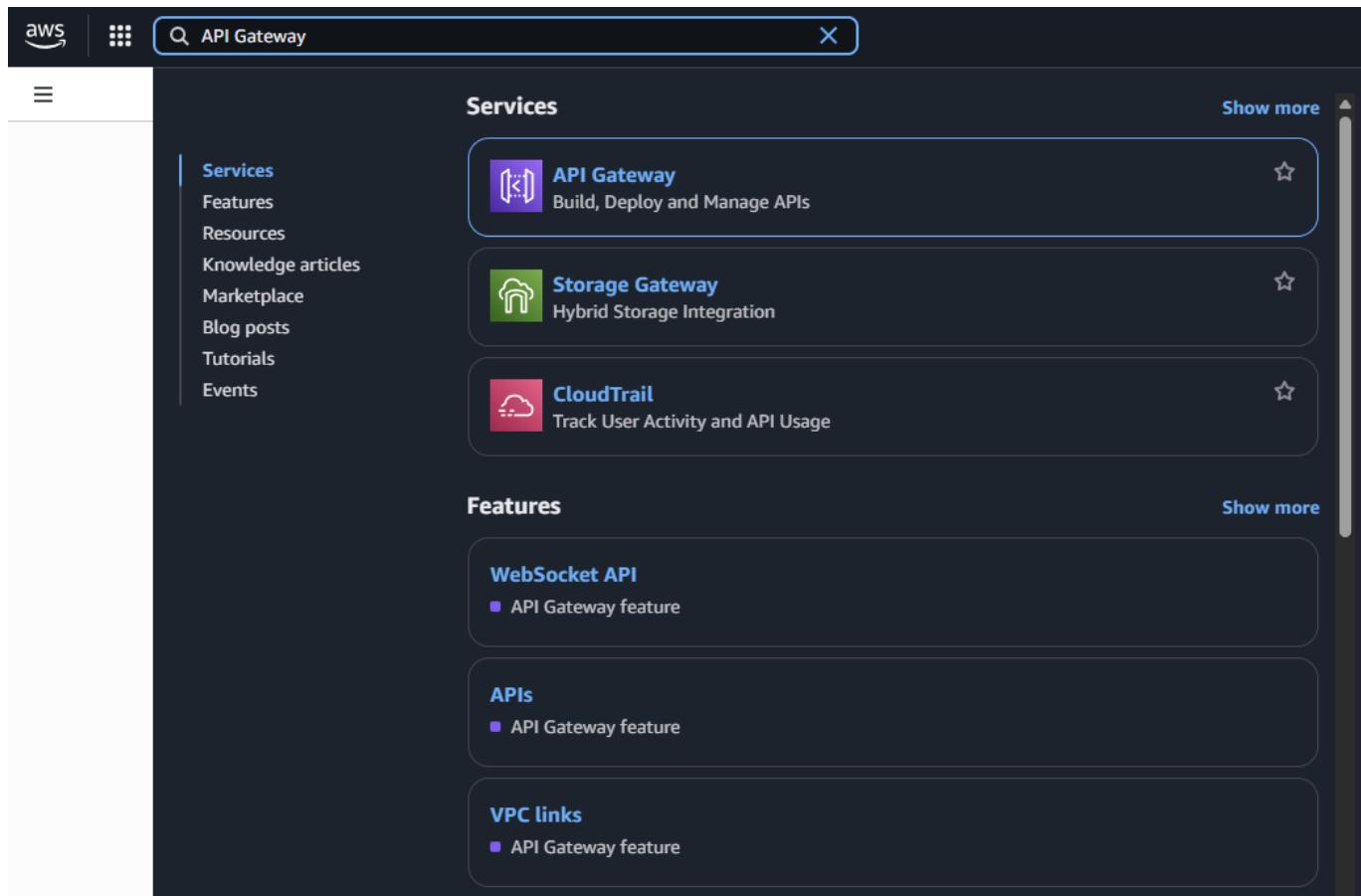
## Uppsättning av Amazon API Gateway för HTTP API

Denna guide beskriver hur man skapar och konfigurerar Amazon API Gateway som en HTTP API för att hantera kommunikationen mellan frontend och serverlösa Lambda-funktioner. Målet är att tillhandahålla en skalbar, säker och lättanvänd ingångspunkt för webbapplikationen, som möjliggör REST-likt interaktioner utan att behöva hantera servrar. API Gateway kommer att routa inkommande HTTP-förfrågningar till Lambda-funktionerna, hantera CORS och säkerställa att data från formulär kan skickas och tas emot på ett pålitligt sätt.

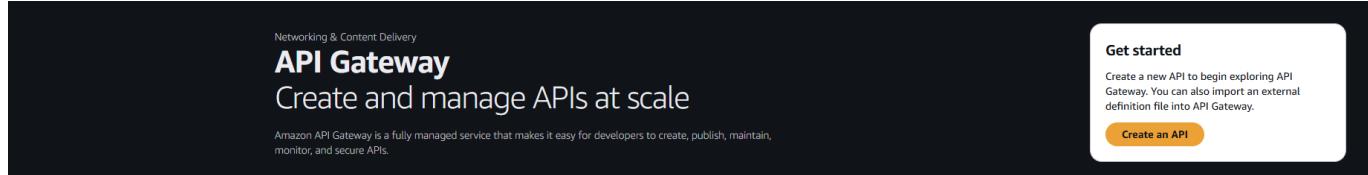
### Steg 1: Bege dig till aws.amazon.com

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' services: EC2, CloudFront, S3, Lambda, DynamoDB, API Gateway, CloudWatch, and IAM. Below this is a 'Welcome to AWS' section with links to 'Getting started with AWS', 'Training and certification', and 'AWS Builder Center'. The main area has four cards: 'AWS Health' (0 open issues, 0 scheduled changes, 0 other notifications), 'Cost and usage' (a bar chart showing costs from Jun 25 to Nov 25 across various AWS services), 'Applications' (empty), and 'Cost and usage' (another bar chart showing forecasted month-end costs). At the top right, it says 'Europe (Ireland)' and 'Account ID: 6684-2073-9197'.

### Steg 2: Ange API Gateway i sökrutan och välj "API Gateway - Build, Deploy and Manage APIs"



### Steg 3: Välj "Create an API" längst till höger



### Steg 4: Välj "HTTP API"

This screenshot shows the 'Choose an API type' step of the API creation process. It displays the 'HTTP API' option, which is described as 'Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.' Below this, it says 'Works with the following:' and lists 'Lambda, HTTP backends'. At the bottom right are two buttons: 'Import' and 'Build'.

### Steg 5: Ange ett namn för vår API, jag kommer namnge den contactHandlerFormAPI Välj även vår Lambda-function vi skapade under förgående steg under Integrations

## Configure API

**API details**

**API name**  
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.

**IP address type** | [Info](#)  
Select the type of IP addresses that can invoke the default endpoint for your API. You don't need to redeploy your API for the update to take effect.

**IPv4**  
Includes only IPv4 addresses.

**Dualstack**  
Includes IPv4 and IPv6 addresses.

**Integrations (1) [Info](#)**  
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Lambda	Remove	
AWS Region: eu-west-1	Lambda function: arn:aws:lambda:eu-west-1:669420739197:function:contactFormHandler	Version: 2.0

[Add integration](#)

[Cancel](#) [Review and create](#) [Next](#)

## Steg 6: Här behöver vi ange en route till vår API som ska nås via vår Lambda-fuction

Fyll i: - **Method** – t.ex. **POST** - **Resource path** – **/contact** - **Integration target** - **contactFormHandler** 6.  
Spara med "**Add route**"

### Configure routes - optional

**Configure routes [Info](#)**  
API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path	Integration target
POST	/contact	contactFormHandler

[Add route](#)

[Cancel](#) [Review and create](#) [Previous](#) [Next](#)

## Steg 7: Här kan du ange en stage för vår API, alltså ifall vi skapar en stage som heter prod så kan API nås via API-URL/prod/api

Men vi väljer att köra \$default för enkelhetens skull, detta resulterar med att vi kan nå vår API genom API-URL/api

### Define stages - optional

**Configure stages [Info](#)**  
Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to autodeploy. By default, all HTTP APIs created through the console have a default stage named \$default. All changes that you make to your API are autodeployed to that stage. You can add stages that represent environments such as development or production.

Stage name	Auto-deploy
\$default	<input checked="" type="checkbox"/>

[Add stage](#)

[Cancel](#) [Previous](#) [Next](#)

## Steg 8: Slutligen får vi en översikt över vår API med dess konfigurationer

Välj därefter "Create"

**Review and create**

**API name and integrations**

API name	contactHandlerFormAPI	IP address type	IPv4
Integrations	<ul style="list-style-type: none"> <li>• contactFormHandler (Lambda)</li> </ul>		

**Routes**

**Routes**

- POST /contact → contactFormHandler (Lambda)

**Stages**

**Stages**

- \$default (Auto-deploy: enabled)

Cancel Previous Create

## Steg 9: Du bör nu se vår nyskapade API som enligt bilden nedan:

The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar with 'API Gateway' and 'APIs'. The main area displays a table titled 'APIs (1/1)' with one entry: 'contactHandlerFormAPI'. The table columns include Name, ID, Protocol, API endpoint type, and Created. The 'Created' column shows the date '2025-11-16'.

## Steg 10: Gå sedan in API och granska att vår Routes och Integrations har skapats korrekt:

- **Routes:**

The screenshot shows the 'Routes' section for the 'contactHandlerFormAPI'. It lists a single route: 'POST /contact'. The 'Route details' pane shows the ARN (arn:aws:apigateway:eu-west-1::apis/dkt6vuri6i/routes/x8ooho8) and the integration configuration. The integration is set to 'ezc3neg' and has a 'Configure' button.

- **AWS Lambda Integration:**

The screenshot shows the AWS API Gateway 'Integrations' page. On the left, a sidebar navigation includes 'APIs', 'Develop', 'Deploy', 'Monitor', and 'Protect'. The main area displays 'Routes for contactHandlerFormAPI' with a single route: POST /contact. This route is integrated with a 'Lambda function' named 'contactFormHandler' from the 'eu-west-1' region. The 'Integration details for route' section shows the method as POST /contact (x8oooh08), the Lambda function name, and an 'Integration ID' of 'ezc3neg'. It also includes sections for 'Payload format version' (set to 2.0), 'Invoke permissions' (describing the Lambda resource policy), 'Example policy statement', 'Timeout' (set to 30000 milliseconds), and 'Request parameter mapping' and 'Response parameter mappings' (both set to 'Not configured').

## Steg 11: Slutligen behöver vi sätta upp CORS så att vi Lambda-funktionen kan nås genom vår app

Fyll i: - **Access-Control-Allow-Origin** – S3-bucket URL <http://serverless-bucket-2025.s3-website-eu-west-1.amazonaws.com/> - **Access-Control-Allow-Methods** – POST, OPTIONS - **Access-Control-Allow-Headers** - **content-type** 6. Spara med "Save"

- För tillfället anger vi vår S3-bucket URL, men vi kommer byta denna senare till cloudfront-url (såsom jag har det konfigurerat enligt bilden) när vi konfigurerat upp CloudFront.
- Detta gör att endast S3-domänen kan använda vår Lambda-funktion

The screenshot shows the AWS API Gateway 'CORS' configuration page. The sidebar navigation is identical to the previous screenshot. The main area is titled 'Cross-Origin Resource Sharing' and contains a 'Configure CORS' section. It includes fields for 'Access-Control-Allow-Origin' (with a value of 'https://d3vij5bvefx3w.cloudfront.net' and an 'Add' button), 'Access-Control-Allow-Headers' (with a value of 'content-type' and an 'Add' button), 'Access-Control-Allow-Methods' (with a dropdown menu showing 'POST' and 'OPTIONS'), 'Access-Control-Expose-Headers' (with a text input field and an 'Add' button), 'Access-Control-Max-Age' (with a value of '0'), and 'Access-Control-Allow-Credentials' (with a 'NO' radio button). At the bottom right are 'Cancel' and 'Save' buttons.

## Steg 12: Slutligen kopiera APIs "Default endpoint" och ersätt följande API-URL på raden i contact\_form.html som innehåller:

**API details**

- API ID:** dkt6vuri6i
- Protocol:** HTTP
- Description:** No Description
- IP address type:** IPv4
- Created:** 2025-11-16
- Default endpoint:** Enabled
- ARN:** arn:aws:apigateway:eu-west-1::apis/dkt6vuri6i

**Stages for contactHandlerFormAPI (1)**

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
<u>Default</u>	<a href="https://dkt6vuri6i.execute-api.eu-west-1.amazonaws.com">https://dkt6vuri6i.execute-api.eu-west-1.amazonaws.com</a>	rs5ry	disabled	2025-11-18

```
const apiUrl = "https://dkt6vuri6i.execute-api.eu-west-1.amazonaws.com/contact";
```

## Uppsättning av Amazon CloudFront som reverse proxy med HTTPS

Denna guide beskriver hur man konfigurerar Amazon CloudFront för att distribuera frontend-filer från S3 och ge säker åtkomst via HTTPS. Målet är att skapa en snabb, säker och skalbar distribution av webbapplikationens statiska innehåll. CloudFront fungerar som en reverse proxy som hanterar HTTPS-anslutningar, och som säkerställer att användare alltid får en pålitlig och krypterad anslutning till webbapplikationen.

### Steg 1: Bege dig till aws.amazon.com

**Recently visited**

- EC2
- CloudFront
- S3
- Lambda
- DynamoDB
- API Gateway
- CloudWatch
- IAM

**Billing and Cost Management**

- Aurora and RDS
- Lightsail
- EFS
- Certificate Manager
- VPC

**Applications (0)**

No applications

**Welcome to AWS**

- Getting started with AWS
- Training and certification
- AWS Builder Center

**AWS Health**

- Open issues: 0
- Scheduled changes: 0
- Other notifications: 0

**Cost and usage**

Current month: \$0.34

Forecasted month end: Data unavailable

Savings opportunities: Not enabled

Cost (\$):

Month (Year)	Cost (\$)
Jun 25	0
Jul 25	0
Aug 25	0
Sep 25	0
Oct 25	0
Nov 25	0

### Steg 2: Ange Cloudfront i sökrutan och välj "CloudFront - Global Content Delivery Network"

The screenshot shows the AWS CloudFront documentation page. At the top, there's a search bar with the text "Cloudfront". On the left, a sidebar lists "Services", "Documentation", "Knowledge articles", "Marketplace", "Blog posts", "Events", and "Tutorials". The main content area features a large box for "CloudFront" with its description "Global Content Delivery Network" and a star icon. Below this are three sections: "Documentation" (with links to "Implementation Guide" and "User Guide"), and "Amazon CloudFront" (with a link to "AWS Whitepaper"). A "Show more" link is also present.

### Steg 3: Välj "Create a CloudFront distribution"

The screenshot shows the Amazon CloudFront landing page. It features the title "Amazon CloudFront" and the subtitle "Securely deliver content with low latency and high transfer speeds". Below this is a description of what CloudFront does. On the right, there's a "Get started with CloudFront" section with a button labeled "Create a CloudFront distribution".

### Steg 4: För betalningsplan är enklast att välja pay-as-you-go för vårt ändamål eftersom vi kommer ändå inte hantera större mängder trafik

## Choose a plan

<input type="radio"/> Free <b>\$0/month</b> <small>For hobbyists, learners, and developers getting started.</small>	<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Always-on DDoS protection</li> <li><input checked="" type="checkbox"/> Protect against common web threats with AWS WAF</li> <li><input checked="" type="checkbox"/> IP-based rate limiting</li> <li><input checked="" type="checkbox"/> Geographic traffic blocking</li> <li><input checked="" type="checkbox"/> Serverless edge compute</li> <li><input checked="" type="checkbox"/> Smart routing</li> <li><input checked="" type="checkbox"/> Global CDN</li> </ul>	<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> DNS</li> <li><input checked="" type="checkbox"/> Free TLS certificate</li> <li><input checked="" type="checkbox"/> Tiered caching</li> <li><input checked="" type="checkbox"/> Default caching rules</li> <li><input checked="" type="checkbox"/> Fast cache invalidations</li> <li><input checked="" type="checkbox"/> 5GB S3 storage included</li> </ul>
<input type="radio"/> Pro <b>\$15/month</b> <small>Launch and grow small websites, blogs, and applications.</small>	<p><b>Everything in Free, plus:</b></p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Protections for WordPress, PHP, and SQL databases</li> <li><input checked="" type="checkbox"/> Header-based threat filtering</li> <li><input checked="" type="checkbox"/> Edge key-value store</li> <li><input checked="" type="checkbox"/> Logging</li> <li><input checked="" type="checkbox"/> 50GB S3 storage included</li> </ul>	<p><b>Business</b>  <b>\$200/month</b>  <small>Protect and accelerate business applications.</small></p> <p><b>Everything in Pro, plus:</b></p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Advanced DDoS protection</li> <li><input checked="" type="checkbox"/> Bot management and analytics</li> <li><input checked="" type="checkbox"/> Regex-based threat filtering</li> <li><input checked="" type="checkbox"/> JavaScript challenge</li> <li><input checked="" type="checkbox"/> Custom caching rules</li> <li><input checked="" type="checkbox"/> Private origins within VPC</li> <li><input checked="" type="checkbox"/> Uptime SLA</li> <li><input checked="" type="checkbox"/> 1TB S3 storage included</li> </ul>
<input type="radio"/> Premium <b>\$1000/month</b> <small>Scale and protect business and mission-critical applications.</small>	<p><b>Premium</b>  <b>\$1000/month</b>  <small>Scale and protect business and mission-critical applications.</small></p> <p><b>Everything in Business, plus:</b></p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> High-speed origin routing</li> <li><input checked="" type="checkbox"/> Origin load reduction</li> <li><input checked="" type="checkbox"/> Automatic origin failover</li> <li><input checked="" type="checkbox"/> 5TB S3 storage included</li> </ul>	<p><b>Usage allowance</b>  <small>10M requests / 50TB per month</small></p> <p><b>Usage allowance</b>  <small>125M requests / 50TB per month</small></p> <p><b>Usage allowance</b>  <small>500M requests / 50TB per month</small></p>

Pay as you go  
Pricing varies with usage

Pay only for what you use based on traffic and enabled features related to this distribution in CloudFront, AWS WAF, Route 53, S3, and CloudWatch Logs. Choose this option if you serve more than 50TB or 500M requests per month, need control over feature selection (including [features not available in pricing plans](#)), or want to use your custom rates.

[Cancel](#)[Next](#)

## Steg 5: Ange ett namn för vår CloudFront distribution, jag kommer namnge den serverless-app-cloudfront

### Get started

Connect your websites, apps, files, video streams, and other content to CloudFront. We optimize the performance, reliability, and security for your web traffic.

<p><b>Distribution options</b> <a href="#">Info</a></p> <p><b>Distribution name</b>  Name will be stored as a tag on the resource. You can change the name, or more tags, later.  <input type="text" value="serverless-app-cloudfront"/></p> <p><b>Description - optional</b>  <input type="text"/></p> <p><b>Distribution type</b></p> <p><input checked="" type="radio"/> Single website or app  Choose if each website or application will have a unique configuration.</p> <p><input type="radio"/> Multi-tenant architecture - New  Choose when you have multiple domains that need to share configurations. This is a common architecture for SaaS providers.</p>
<p><b>Domain</b> <a href="#">Info</a></p> <p><b>Route 53 managed domain - optional</b>  Enter a domain that's already registered with Route 53 in your AWS account. CloudFront will provision a TLS certificate for you. If you have a domain from a different DNS provider, skip this step and configure your domain later.  <input type="text"/>  <a href="#">Check domain</a></p> <p><b>Tags - optional</b></p>

[Cancel](#)[Previous](#)[Next](#)

## Steg 6: Välj sedan Amazon S3 och bläddra fram vår S3-bucket. Resten kan du lämna som det är

## Specify origin

### Origin type

Your origin is where your content (such as a website or app) lives. CloudFront works with AWS-based origins and origins hosted on other cloud providers.

#### Origin type

##### Amazon S3

Deliver static assets like files and images, statically generated websites or single page applications (SPA).

##### Elastic Load Balancer

Deliver applications hosted behind ELB such as dynamic websites, web services, and APIs.

##### API Gateway

Deliver API endpoints for REST APIs hosted on API Gateway.

##### Elemental MediaPackage

Deliver end-to-end live events or video on demand (VOD).

##### VPC origin

Deliver applications and content hosted within private VPCs, such as EC2 instances and Application Load Balancers.

##### Other

Refer to any AWS or non-AWS origin through its publicly resolvable URL.

### Origin

#### S3 origin

Choose an AWS origin, or enter your origin's domain name. [Learn more ↗](#)

serverless-bucket-2025.s3.eu-west-1.amazonaws.com

[Browse S3](#)

⚠ This S3 bucket has static web hosting enabled. If you plan to use this distribution as a website, we recommend using the S3 website endpoint rather than the bucket endpoint.

[Use website endpoint](#)

#### Origin path - optional

The directory path within your origin where your content is stored. [Learn more ↗](#)

/path

## Steg 7: Vi kan hoppa över att sätta upp säkerheten med WAF eftersom vår CloudFront är ändå endast i testnings-syfte

### Enable security

#### Web Application Firewall (WAF) [Info](#)

##### Enable security protections

Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.

##### Do not enable security protections

Select this option if your application does not need security protections from AWS WAF.

[Cancel](#)

[Previous](#)

[Next](#)

## Steg 8: Du får nu översikt över vår CloudFront distribution. Gå vidare genom att välja "Create distribution"

## Review and create

**General configuration**

Distribution name serverless-app-cloudfront	Description -	Billing Pay-as-you-go (\$0/month)	Edit
--	------------------	--------------------------------------	------

**Origin**

<small>Because you granted CloudFront access to your origin, CloudFront can write and update S3 bucket policies that restrict access to your S3 origin to CloudFront.</small>			
S3 origin serverless-bucket-2025.s3.eu-west-1.amazonaws.com	Origin path -	Grant CloudFront access to origin Yes	Enable Origin Shield No
Connection attempts 3	Connection timeout 10		

**Cache settings**

CloudFront will apply default cache settings tailored to serving content from a S3 origin. You can customize settings after you create your distribution.			
---	--	--	--

**Security**

Security protections None	Use monitor mode No	Use existing WAF configuration No	Edit
------------------------------	------------------------	--------------------------------------	------

[Cancel](#) [Previous](#) [Create distribution](#)

## Steg 9: Slutligen bör du se en översikt över CloudFront distributionen vi precis skapade

CloudFront	Distributions (1) <small>Info</small>		Enable	Disable	Delete	Create distribution
Distributions						
Policies						
Functions						
Static IPs						
VPC origins						

Distributions (1) Info

ID	Status	Description	Type	Domain name (standard)	Alternate domain n...	Origins	Pr...
EA40LKJ18UZM7	Enabled	-	Standard	d3vij5bvefx3w.cloudfront.net	-	serverless-bucket-2025.s3.eu-west-1.amazon...	Pay

## Steg 10: Gå in på vår CloudFront Distribution och börja med att lägga till index.html för "Default root object" genom att navigera till "Edit" längst bort till höger

**Settings**

Name serverless-app-cloudfront	Alternate domain names -	Standard logging Off
Description -	Add domain	Cookie logging Off
Price class Use all edge locations (best performance)		Default root object index.html
Supported HTTP versions HTTP/2, HTTP/1.1, HTTP/1.0		

## Steg 11: Ange sedan "index.html" för "Default root object"

## Edit settings

**General**

**AWS WAF web ACL - optional**  
Choose the web ACL in AWS WAF to associate with this distribution.

**Alternate domain name (CNAME) - optional**  
Add the custom domain names that you use in URLs for the files served by this distribution.

**Add item**

① To add a list of items, use the [bulk editor](#).

**Custom SSL certificate - optional**  
Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

**Choose certificate**

**Request certificate**

**Supported HTTP versions**  
Add support for additional HTTP versions. HTTP/1.0 and HTTP/1.1 are supported by default

HTTP/2

HTTP/3

**Default root object - optional**  
The object (file name) to return when a viewer requests the root URL (/) instead of a specific object.

**index.html**

**Description - optional**

**Steg 12:** Nu behöver vi lägga till vår API som vi skapade tidigare som en origin. Gör detta genom att navigera till Origins -> "Create origin"

General | Security | **Origins** | Behaviors | Error pages | Invalidations | Logging | Tags

Origins (2)			
<input type="text"/> Filter origins by property or value			
Origin name	Origin domain	Origin path	Origin type
serverless-bucket-2025.s3-website-eu-west-1.amazonaws.com-mi342b31xjm	serverless-bucket-2025.s3.eu-west-1.amazonaws.com		S3
API Gateway	dkt6vuri6i.execute-api.eu-west-1.amazonaws.com		API Gateway

**Steg 13:** Välj vår API Gateway i dropdown-listan när du väljer "Origin domain". Den kommer automatiskt generera din API-url som bilden nedan visar. Resten kan du lämna som det är

CloudFront > Distributions > EA40LJKJ18UZM7 > Create origin

**Create origin**

**Settings**

**Origin domain**  
Choose an AWS origin, or enter your origin's domain name. [Learn more](#)

dkt6vuri6i.execute-api.eu-west-1.amazonaws.com

Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.

**Protocol** | [Info](#)

HTTP only

HTTPS only

Match viewer

**HTTPS port**  
Enter your origin's HTTPS port. The default is port 443.  
443

**Minimum Origin SSL protocol**  
The minimum SSL protocol that CloudFront uses with the origin.

TLSv1.2

TLSv1.1

TLSv1

SSLv3

**Origin path - optional**  
Enter a URL path to append to the origin domain name for origin requests.  
 Enter the origin path

**Name**  
Enter a name for this origin.  
 dkt6vuri6i.execute-api.eu-west-1.amazonaws.com

**Steg 14:** Du bör nu ha två origins för din CloudFront distribution. En för din S3-bucket, och en för din API

Origins (2)			
<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Create origin"/> <span style="float: right;">◀ 1 ▶ ⚙</span>			
Origin name	Origin domain	Origin path	Origin type
<input type="radio"/> serverless-bucket-2025.s3-website-eu-west-1.amazonaws.com-mi342b31xjm	serverless-bucket-2025.s3.eu-west-1.amazonaws.com		S3
<input type="radio"/> API Gateway	dkt6vuri6i.execute-api.eu-west-1.amazonaws.com		API Gateway

**Steg 15: Slutligen behöver vi även lägga till två behaviors. Återigen, en för din S3-bucket, och en för din API. Gör detta genom att navigera till Behaviors -> "Create behavior"**

Fyll i följande för S3-bucket: - **Path pattern** – `/` - **Origin and origin groups** – Välj din S3-bucket - **Viewer protocol policy** - Redirect HTTP to HTTPS - **Allowed HTTP methods** - GET, HEAD - **Cache policy** - CachingOptimized Spara med "**Save changes**"

### Edit behavior

#### Settings

Path pattern | [Info](#)

Origin and origin groups

serverless-bucket-2025.s3-website-eu-west-1.amazonaws.com-mi342b31xjm

Compress objects automatically | [Info](#)  Yes

Viewer

Viewer protocol policy  Redirect HTTP to HTTPS

Allowed HTTP methods  GET, HEAD

Restrict viewer access  No

Cache key and origin requests  Cache policy and origin request policy (recommended)

Cache policy  Legacy cache settings

Choose an existing cache policy or create a new one.

CachingOptimized Policy with caching enabled. Supports Gzip and Brotli compression. Recommended for S3

[Create cache policy](#) [View policy](#)

Fyll i följande för API – **Path pattern** – `/api/*` - **Origin and origin groups** – Välj din API Gateway - **Viewer protocol policy** - HTTPS only - **Allowed HTTP methods** - GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE - **Cache policy** - CachingDisabled - **Origin request policy** - AllViewerExceptHostHeader Spara med "**Save changes**"

## Edit behavior

**Settings**

Path pattern | [Info](#)  
 X

Origin and origin groups  
 API Gateway ▼

Compress objects automatically | [Info](#)  
 No  
 Yes

**Viewer**

Viewer protocol policy  
 HTTP and HTTPS  
 Redirect HTTP to HTTPS  
 HTTPS only

Allowed HTTP methods  
 GET, HEAD  
 GET, HEAD, OPTIONS  
 GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Cache HTTP methods  
 GET and HEAD methods are cached by default.  
 OPTIONS

Allow gRPC requests over HTTP/2 | [Info](#)  
 Enable

Restrict viewer access  
 If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.  
 No  
 Yes

**Cache key and origin requests**  
 We recommend using a cache policy and origin request policy to control the cache key and origin requests.

Cache policy and origin request policy (recommended)  
 Legacy cache settings

**Cache policy**  
 Choose an existing cache policy or create a new one.  
 CachingDisabled  
 Policy with caching disabled Recommended for API Gateway ▼ C

[Create cache policy](#) View policy

**Origin request policy - optional**  
 Choose an existing origin request policy or create a new one.  
 AllViewerExceptHostHeader  
 Policy to forward all parameters in viewer requests except for the Host header Recommended for API Gateway ▼ C

[Create origin request policy](#) View policy

**Steg 16: Slutligen, ifall du redan ersatt API-URL på raden i contact\_form.html som innehåller följande med din API**

```
const apiUrl = "https://dkt6vuri6i.execute-api.eu-west-1.amazonaws.com/contact";
```

**Så är det bara slutligen ersätta cloudfont-urlen som finns i contactFormHandler.js med din cloudfont-url**

## Uppsättning av AWS CodePipeline för CI/CD

Denna guide beskriver hur man skapar en CI/CD-pipeline med AWS CodePipeline kopplad till ett GitHub-repository. Målet är att automatisera bygg och deployment av både frontend-filer till S3 och backend-funktioner till Lambda. Den säkerställer att ändringar i koden automatiskt testas, byggs och distribueras, vilket gör att nya funktioner snabbt och på ett pålitligt sätt blir tillgängliga i produktionsmiljön.

**Steg 1: Bege dig till [aws.amazon.com](https://aws.amazon.com)**

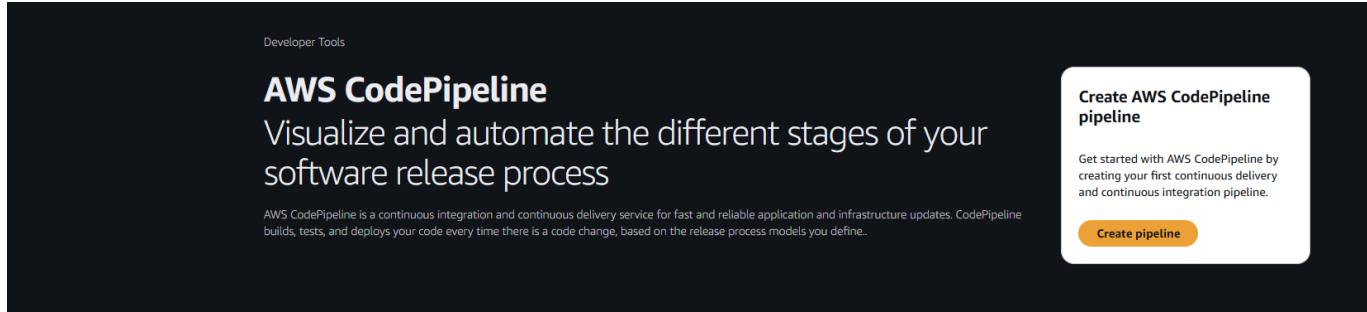
The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' services like EC2, CloudFront, S3, Lambda, DynamoDB, API Gateway, CloudWatch, and IAM. Below that is a 'Welcome to AWS' section with links for 'Getting started with AWS', 'Training and certification', and 'AWS Builder Center'. The main area has several cards: 'Applications (0)' which says 'No applications. Get started by creating an application.'; 'Cost and usage' showing current month cost of \$0.34 and a bar chart for forecasted month end; and 'AWS Health' showing 0 open issues and 0 scheduled changes. At the bottom, there are links to 'Go to myApplications', 'Go to AWS Health', and 'Go to Billing and Cost Management'.

## Steg 2: Ange Codepipeline i sökrutan och välj "CodePipeline - Release Software using Continuous Delivery"

The screenshot shows the AWS Marketplace search results for 'Codepipeline'. The search bar at the top contains 'Codepipeline'. On the left, a sidebar lists 'Services', 'Resources', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', and 'Tutorials'. The main results section shows a card for 'CodePipeline' with the subtext 'Release Software using Continuous Delivery'. Below it, a message says 'Looking for resources in other Regions? You can enable cross-Region search for resources across all Regions in your account by specifying an aggregator index.' with a 'Enable cross-Region search' button. Further down, three specific resources are listed: 'codepipeline-source-trail' (CloudTrail trail), 'AmazonS3Pipeline' (CodePipeline), and 'codepipeline-78334589-artifactsbucket-rule' (Amazon EventBridge rule). Each resource card includes its name, type, and location ('Europe (Ireland) eu-west-1').

- Notera för att ansluta GitHub ihop med CodePipeline på AWS behövs följande connector:  
<https://github.com/marketplace/aws-connector-for-github>

## Steg 3: Välj "Create pipeline"



## Steg 4: Välj därefter "Build custom pipeline" under Category

Step 1 Choose creation option [Info](#)

Step 2 Choose pipeline settings

Step 3 Add source stage

Step 4 Add build stage

Step 5 Add test stage

Step 6 Add deploy stage

Step 7 Review

**Category**

Deployment     Continuous Integration     Automation

**Build custom pipeline**

[Cancel](#) [Next](#)

## Steg 5: Ange ett namn för vår CI/CD Pipeline, jag kommer namnge den AmazonS3Pipeline

5. Välj/Fyll i även in följande:

- **Execution Mode** – **Queued**
- **New Service Role** – Låt AWS CodePipeline skapa en IAM-roll åt dig med korrekta rättigheter

6. Navigera sedan ner till **Advanced settings** och välj **Custom location**

- Du behöver nämligen ha en S3-bucket för att lagra dina artifacts.
- Skapa helt enkelt en S3-bucket som tidigare och ge den ett passande, jag döpte min till **artifacts-bucket-2025**
- Välj därefter din nyskapade S3-bucket för Custom location

**S3-artifacts i CI/CD** är helt enkelt filer som din bygg- och deployprocess sparar i ett tryggt förråd (Amazon S3) under arbetets gång.

Tänk dig att din CI/CD-pipeline bygger något — till exempel en app, en konfigurationsfil eller ett paket. Resultatet behöver sparas någonstans så att nästa steg i processen kan använda det.

Amazon S3 fungerar då som **en gemensam lagringsplats** där pipelinen kan lägga sina filer och hämta dem när de behövs.

**Kort sagt:** S3-artifacts är filer som CI/CD-systemet lagrar i S3 så att de kan användas och delas mellan olika steg i automatiseringskedjan.

- Step 1  
[Choose creation option](#)
- 
- Step 2  
**Choose pipeline settings**
- 
- Step 3  
[Add source stage](#)
- 
- Step 4  
[Add build stage](#)
- 
- Step 5  
[Add test stage](#)
- 
- Step 6  
[Add deploy stage](#)
- 
- Step 7  
[Review](#)

## Choose pipeline settings Info

Step 2 of 7

### Pipeline settings

#### Pipeline name

Enter the pipeline name. You cannot edit the pipeline name after it is created.

AmazonS3Pipeline

No more than 100 characters

#### Execution mode Info

Choose the execution mode for your pipeline. This determines how the pipeline is run.

- Superseded  
 Queued  
 Parallel

#### Service role

New service role

Create a service role in your account

Existing service role

Choose an existing service role from your account

#### Role name

AWSCodePipelineServiceRole-eu-west-1-AmazonS3Pipeline

Type your service role name

- Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

### ▼ Advanced settings

Configure artifact store location, encryption settings, and pipeline variables for your pipeline.

#### Artifact store

Default location

Create a default S3 bucket in your account.

Custom location

Choose an existing S3 location from your account in the same region and account as your pipeline

#### Bucket

artifacts-bucket-2025



#### Encryption key

Default AWS Managed Key

Use the AWS managed customer master key for CodePipeline in your account to encrypt the data in the artifact store.

Customer Managed Key

To encrypt the data in the artifact store under an AWS KMS customer managed key, specify the key ID, key ARN, or alias ARN.

**Steg 6: När vi kommer till "Add source stage" är det dags att koppla samman vår GitHub-repo och AWS CodePipeline**

- Step 1  
[Choose creation option](#)
- Step 2  
[Choose pipeline settings](#)
- Step 3  
**Add source stage**
- Step 4  
[Add build stage](#)
- Step 5  
[Add test stage](#)
- Step 6  
[Add deploy stage](#)
- Step 7  
[Review](#)

### Add source stage Info

Step 3 of 7

#### Source

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

**Connection**  
Choose an existing connection that you have already configured, or create a new one and then return to this task.

or

**Repository name**  
Choose a repository in your GitHub account.

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

**Default branch**  
Default branch will be used only when pipeline execution starts from a different source or manually started.

**Output artifact format**  
Choose the output artifact format.

**CodePipeline default**  
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

**Full clone**  
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions. [Learn more ↗](#)

Enable automatic retry on stage failure

**Steg 7: "Add test stage" och "Add build stage" kan vi skippa**

**Steg 8: När vi kommer till "Add deploy stage" behöver vi tala om för AWS CodePipeline vilken S3-bucket det är som ska ingå i CI/CD deploymentprocessen genom att ange vår S3-bucket under "Bucket". Resten kan du lämna som det är**

Step 1  
[Choose creation option](#)

Step 2  
[Choose pipeline settings](#)

Step 3  
[Add source stage](#)

Step 4  
[Add build stage](#)

Step 5  
[Add test stage](#)

Step 6  
[Add deploy stage](#)

Step 7  
Review

## Add deploy stage Info

Step 6 of 7

**ⓘ You cannot skip this stage**  
Pipelines must have at least two stages. Your second stage must be either a build, test or deployment stage.  
Choose a provider for either the build stage, test stage or deployment stage.

### Deploy

**Deploy provider**  
Choose how you want to deploy your application or content. Choose the provider, and then provide the configuration details for that provider.

Amazon S3

**Region**

Europe (Ireland)

**Input artifacts**  
Choose an input artifact for this action. [Learn more](#)

SourceArtifact X  
Defined by: Source

No more than 100 characters

**Bucket**

serverless-bucket-2025

**Deployment path - optional**

Extract file before deploy  
The deployed artifact will be unzipped before deployment.

**► Additional configuration**

Configure automatic rollback on stage failure

Enable automatic retry on stage failure

[Cancel](#) [Previous](#) [Next](#)

**Steg 9: Du får nu översikt över vår AWS CodePipeline. Gå vidare genom att välja "Create pipeline"**

- Step 1  
[Choose creation option](#)
- Step 2  
[Choose pipeline settings](#)
- Step 3  
[Add source stage](#)
- Step 4  
[Add build stage](#)
- Step 5  
[Add test stage](#)
- Step 6  
[Add deploy stage](#)
- Step 7  
[Review](#)

**Review** [Info](#)

Step 7 of 7

**Step 2: Choose pipeline settings****Pipeline settings****Pipeline name**  
AmazonS3Pipeline2**Pipeline type**  
V2**Execution mode**  
QUEUED**Artifact location**  
artifacts-bucket-2025**Service role name**  
AWSCodePipelineServiceRole-eu-west-1-AmazonS3Pipeline2**Step 3: Add source stage****Source action provider****Source action provider**  
GitHub (via GitHub App)**OutputArtifactFormat**  
CODE\_ZIP**DetectChanges**  
true**ConnectionArn**  
arn:aws:codeconnections:eu-west-1:669420739197:connection/6d033439-6143-4d4e-8d6a-e722a9b8a156**FullRepositoryId**  
91maxore-hub/serverless-app**Default branch**  
main**Enable automatic retry on stage failure**  
Enabled**Steg 10: Slutligen bör du se en översikt över AWS CodePipelinen vi precis skapade som kommer hantera CI/CD deployment**

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
AmazonS3Pipeline	Succeeded	Source - face7f9212: CI/CD Pipeline	1 day ago	View details



Efter att allt var uppsatt och CI/CD-deployment gick igenom kunde jag gå till: [🔗](https://d3vbjy5bvefx3w.cloudfront.net)  
<https://d3vbjy5bvefx3w.cloudfront.net>

Min PHP-app laddas med giltigt SSL-certifikat, automatisk HTTPS och reverse proxy som hanterar trafiken smidigt genom CloudFront. Allt detta sker helt automatiskt – både deployment och certifikatförflyelse.

