

Kurs: Skalbara molnapplikationer

Inlämning: 2 Del 1

Namn: Maria Schillström

Datum: 2025-11-01

Git: <https://github.com/MariaSchillstrom/Docker-Swarm-.NET-MVC-application-.git>

Innehållsförteckning

- Innehållsförteckning
- Översikt av lösningen
 - Arkitektur
 - Hur det fungerar
- AWS-tjänster som används
- Komponenternas uppgift och syfte
 - Infrastrukturkomponenter
 - Applikationskomponenter
- Säkerhetshantering
- Infrastructure as Code och Automation
 - CloudFormation Templates
 - Automation
- Webbapplikationen
 - Test-applikation: .NET MVC
- Implementation - Steg för steg
 - 1. Skapa infrastruktur med CloudFormation
 - 2. Initiera Docker Swarm
 - 2.1 Hämta IP-adresser
 - 2.2 Anslut till Manager
 - 2.3 Initiera Swarm
 - 2.4 Joina Workers
 - 2.5 Verifiera kluster
 - 3. Deploya test-stack (nginx)
 - 3.1 Skapa deployment-skript
 - 3.2 Kör deployment
 - 3.3 Resultat
 - 4. Testa och skala
 - 4.1 Testa i webbläsare
 - 4.2 Skala services
 - 5. Skapa och containerisera MVC-app
 - 5.1 Skapa MVC-projekt
 - 5.2 Skapa Dockerfile
 - 5.3 Skapa ECR repository
 - 5.4 Logga in till ECR
 - 6. Bygg och deploya MVC-appen
 - 6.1 Bygg och pusha image

- 6.2 Uppdatera deployment-skript
- 6.3 Åtgärda ECR-access
- 6.4 Deploya MVC-appen
- Sammanfattning
 - Vad som skapats
 - Lärdomar
- Bilaga: Att använda IaC Generator
 - Steg 1: Skapa Security Group manuellt
 - 1.1 Grundinställningar
 - 1.2 Self-reference för Swarm-kommunikation
 - Steg 2: Skapa CloudFormation template med IaC Generator
 - 2.1 Navigera till IaC Generator
 - 2.2 Starta scan
 - 2.3 Välj Security Group
 - 2.4 Skapa template
 - 2.5 Namnge template
 - 2.6 Välj rätt resource
 - 2.7 Slutför
 - Steg 3: Parametrisera och spara
 - Sammanfattning av IaC Generator-processen

Översikt av lösningen

Arkitektur

Note: Denna tutorial följer inte ett production-ready arbetsflöde där applikationen normalt utvecklas och testas färdigt innan infrastrukturen sätts upp. I produktion skulle antingen (1) en färdig MVC-applikation finnas innan Swarm-konfigurationen, eller (2) frontend och backend separeras där frontend hostas på t.ex. AWS S3 för att möjliggöra snabba uppdateringar utan container-rebuilds. Det sistnämnda alternativet behandlas i en separat rapport.

För denna labb antar vi att en färdig applikation finns tillgänglig.

Implementerad lösning: Jag har skapat en skalbar containerbaserad värdmiljö för en .NET MVC-webbapplikation med följande komponenter:

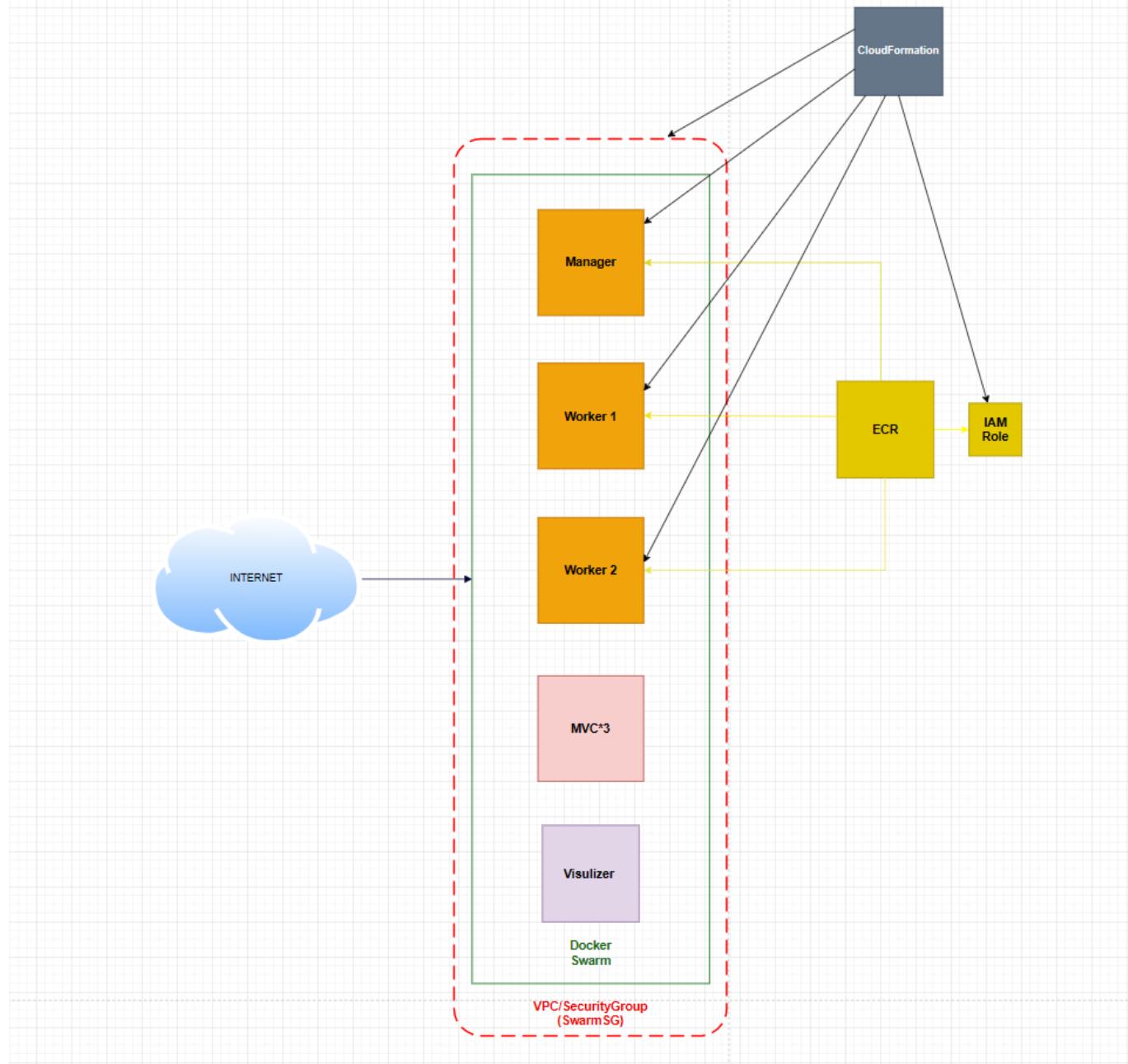
Infrastruktur:

- 3 EC2-instanser på AWS (1 Manager + 2 Workers)
- Docker Swarm orchestrar containers över alla noder
- Security Group kontrollerar nätverkstrafik
- IAM Role ger säker åtkomst till container registry

Applikation:

- .NET MVC-app containeriserad med Docker
- Lagrad i privat ECR (Elastic Container Registry)
- Deployad med 3 replicas för redundans
- Visualizer för grafisk överblick av klustret (körs endast på Manager)

Hur det fungerar



Figur 1: Systemarkitektur för Docker Swarm på AWS. CloudFormation skapar infrastrukturen, IAM ger ECR-access, och Swarm orchestrerar containers över tre noder.

Flöde vid deployment:

1. Developer bygger Docker image lokalt
2. Image pushas till ECR
3. Deploy-skript körs på Manager
4. Manager hämtar image från ECR
5. Swarm distribuerar 3 replicas över noderna
6. Manager bestämmer placement baserat på constraints
7. Services exponeras via port 80

Fördelar med denna setup:

- **Resiliens:** Om en nod går ner fortsätter de andra
- **Skalbarhet:** `docker service scale myapp_web=5` ökar till 5 replicas

- **Loadbalancing:** Inbyggt i Swarm, ingen extern loadbalancer behövs
- **Orchestrering:** Manager koordinerar var containers ska köras
- **Rolling updates:** Uppdatera app utan downtime
- **Enkel hantering:** Ett deploy-kommando uppdaterar hela klustret

AWS-tjänster som används

EC2 (Elastic Compute Cloud)

- 3 t3.micro instanser för Swarm-klustret
- Kör Docker Engine och applikations-containers

ECR (Elastic Container Registry)

- Privat registry för Docker images
- Lagrar MVC-applikationens container image

VPC & Security Groups

- Nätverksisolering och brandväggsregler
- Kontrollerar in-/utgående trafik

IAM (Identity and Access Management)

- Hanterar behörigheter för EC2 att hämta images från ECR

CloudFormation

- Infrastructure as Code för automatiserad resurs-skapande

Komponenternas uppgift och syfte

Infrastrukturkomponenter

Security Group (SwarmSG)

- **Syfte:** Brandvägg som kontrollerar nätverkstrafik
- **Uppgift:** Tillåter SSH, HTTP, Swarm-kommunikation mellan noder

EC2 Manager-nod

- **Syfte:** Koordinerar Swarm-klustret
- **Uppgift:** Schemalägger containers, hanterar state, tar emot deploy-kommandon, bestämmer vilken worker som kör vilken container

EC2 Worker-noder (2st)

- **Syfte:** Kör applikations-containers
- **Uppgift:** Exekverar containers enligt Manager's instruktioner

ECR Repository

- **Syfte:** Lagrar Docker images privat
- **Uppgift:** Distribuerar MVC-app image till alla Swarm-noder

IAM Role (EC2-ECR-Access)

- **Syfte:** Säker åtkomst till ECR utan credentials
- **Uppgift:** Ger EC2-instanser read-only access till ECR

Applikationskomponenter

MVC Web Service (3 replicas)

- **Syfte:** Webbapplikation tillgänglig via HTTP
- **Uppgift:** Svarar på HTTP-förfrågningar, loadbalansas automatiskt av Swarm

Visualizer Service (1 replica)

- **Syfte:** Grafisk överblick av Swarm-klustret
- **Uppgift:** Visar noder, services och container-distribution
- **Placement:** Körs endast på Manager-noden (placement constraint)

Säkerhetshantering

Nätverkssäkerhet:

- Security Group begränsar SSH till min IP-adress
- Swarm-kommunikation (portar 2377, 7946, 4789) endast mellan noder
- HTTP öppen för publik access (port 80, 8080)

Åtkomstkontroll:

- IAM Role för EC2 istället för hårdkodade credentials
- Principle of least privilege - endast read-only ECR-access
- SSH-nycklar för säker server-access

Container-säkerhet:

- Privat ECR repository (inte publik Docker Hub)
- Multi-stage Dockerfile minimerar image-storlek
- .NET base images från Microsoft (verifierade)

Data-säkerhet:

- Ingen känslig data i containers eller images
- Secrets kan hanteras via Docker Secrets (ej implementerat i denna demo)

Infrastructure as Code och Automation

CloudFormation Templates

sg.yaml - Parametriserad Security Group

- Definierar alla nätverksregler
- Self-reference för Swarm-kommunikation
- Parametriserad för återanvändbarhet

ec2.yaml - EC2-instanser med IAM Role

- Skapar 3 instanser (1 Manager + 2 Workers)
- IAM Role för ECR-access inkluderad
- UserData installerar Docker automatiskt vid start

Fördelar med IaC:

- Reproducerbar infrastruktur
- Versionskontroll av infrastruktur-kod
- Enkel att skapa/radera hela miljön
- Dokumentation genom kod

Automation

Bash-skript (docker-stack.sh):

- Automatiserar deployment av Docker Stack
- Skapar docker-stack.yml dynamiskt
- Deployer och verifierar services automatiskt

EC2 UserData:

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

Installerar och konfigurerar Docker automatiskt vid instance-start.

Dockerfile: Multi-stage build för optimerad image-skapande och minimal runtime-image.

Webbapplikationen

Test-applikation: .NET MVC

Beskrivning: En minimal ASP.NET Core MVC-applikation med standard template.

Syfte: Verifiera att:

- Container-bygge fungerar
- ECR push/pull fungerar
- Swarm loadbalancing fungerar
- Services kan nås från internet

Funktionalitet:

- Standard MVC hem-sida
- Visar att applikationen körs
- Minimal för att fokusera på infrastruktur

Varför .NET MVC:

- Representerar en verlig webbapplikation
- Visar att Swarm kan hantera stateful applikationer
- Containeriseras enkelt med Dockerfile

Implementation - Steg för steg

1. Skapa infrastruktur med CloudFormation

Jag började med att skapa resurser manuellt för att förstå strukturen, och använde sedan IaC Generator för att generera CloudFormation templates. Dessa parametriseras för återanvändbarhet.

Skapade resurser:

- Security Group (se bilaga)
- 3 EC2-instanser med IAM Role (IAM lades till senare)

Kör skripten:

```
# Security Group
aws cloudformation create-stack \
--stack-name swarm-sg \
--template-body file://templates/sg.yaml

# EC2-instanser
aws cloudformation create-stack \
--stack-name swarm-ec2 \
--template-body file://templates/ec2.yaml \
--capabilities CAPABILITY_NAMED_IAM \
--parameters ParameterKey=SubnetId,ParameterValue=subnet-058684dd8c601d55d
```

Verifiera:

```
aws cloudformation describe-stacks --stack-name swarm-sg --query
'Stacks[0].StackStatus'
aws cloudformation describe-stacks --stack-name swarm-ec2 --query
'Stacks[0].StackStatus'
```

2. Initiera Docker Swarm

2.1 Hämta IP-adresser

```
aws cloudformation describe-stacks --stack-name swarm-ec2 --query
'Stacks[0].Outputs[*].[OutputKey,OutputValue]' --output table
```

Spara Manager och Workers publika/privata IPs.

2.2 Anslut till Manager

```
ssh -i Keyswarm1029.pem ec2-user@<manager-public-ip>
```

```
ssh -i Keyswarm1029.pem ec2-user@54.154.62.190
```

A newer release of "Amazon Linux" is available.

Version 2023.9.20251027:

Run "/usr/bin/dnf check-release-update" for full release and version update info

```
,      #
~\_ ####_      Amazon Linux 2023
~~ \#####\
~~   \###|
~~     \#/ _-- https://aws.amazon.com/linux/amazon-linux-2023
~~     \~'-'>
~~     /
~~_. /_
~/ /_
/m/'
```

```
[ec2-user@ip-172-31-1-34 ~]$ ]
```

2.3 Initiera Swarm

```
sudo docker swarm init --advertise-addr <manager-private-ip>
```

Kopiera join-token som genereras.

```
[ec2-user@ip-172-31-1-34 ~]$ sudo docker swarm init --advertise-addr 172.31.1.34
Swarm initialized: current node (q521kpu631yy1qlmfbnrw11d4) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-303kpn7av16ntcn1r23e7s6nd9b4m99r5cbk08cpjxz60r16jq-6fjqnyfux22mdtjlzd7pnxft5 172.31.1.34:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

2.4 Joina Workers

```
# Worker 1
ssh -i Keyswarm1029.pem ec2-user@<worker1-public-ip>
sudo docker swarm join --token <TOKEN> <manager-private-ip>:2377
exit

# Worker 2
ssh -i Keyswarm1029.pem ec2-user@<worker2-public-ip>
sudo docker swarm join --token <TOKEN> <manager-private-ip>:2377
exit
```

```

> ssh -i keyswarm1029.pem ec2-user@34.245.181.76
The authenticity of host '34.245.181.76 (34.245.181.76)' can't be established.
ED25519 key fingerprint is SHA256:wqx+gnhyoBMmifYaadNZo9m6lafHbXpj0a57qtGeq9o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.245.181.76' (ED25519) to the list of known hosts.

A newer release of "Amazon Linux" is available.
  Version 2023.9.20251027:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #
  ~\_ ####_      Amazon Linux 2023
  ~~ \#####\
  ~~ \|##|
  ~~ \#/ ____ https://aws.amazon.com/linux/amazon-linux-2023
  ~~   \v~' '-->
  ~~~   /
  ~~~_./_
  ~~~_/_/
  _/m/ . ~

[ec2-user@ip-172-31-7-46 ~]$ SwMTKN-1-303kpn7av16ntcn1r23e7s6nd9b4m99r5cbk08cpjxz60r16jq-6fjqnyfux22mdtjlzd7pxnfts 172.31.1.34:2377
-bash: SwMTKN-1-303kpn7av16ntcn1r23e7s6nd9b4m99r5cbk08cpjxz60r16jq-6fjqnyfux22mdtjlzd7pxnfts: command not found
[ec2-user@ip-172-31-7-46 ~]$ sudo docker swarm join --token SwMTKN-1-303kpn7av16ntcn1r23e7s6nd9b4m99r5cbk08cpjxz60r16jq-6fjqnyfux22mdtjlzd7pxnfts 172.31.1.34:2377
This node joined a swarm as a worker.
[ec2-user@ip-172-31-7-46 ~]$ █

```

2.5 Verifiera kluster

```
sudo docker node ls
```

```
[ec2-user@ip-172-31-1-34 ~]$ sudo docker node ls
ID           HOSTNAME        STATUS  AVAILABILITY  MANAGER STATUS   ENGINE VERSION
q521kpu631yy1qlmfbnrw11d4 *  ip-172-31-1-34.eu-west-1.compute.internal  Ready  Active        Leader        25.0.13
vbn9u2t338mbgsmnuy250irws   ip-172-31-7-46.eu-west-1.compute.internal  Ready  Active        25.0.13
d56aulvpogqq01jt4xma214p5   ip-172-31-15-37.eu-west-1.compute.internal Ready  Active        25.0.13
[ec2-user@ip-172-31-1-34 ~]$ █
```

3. Deploya test-stack (nginx)

3.1 Skapa deployment-skript

Skapade bash-skript som automatiserar deployment. Skriptet skapar docker-stack.yml och deployer automatiskt.

Komplett skript: [templates/docker-stack.sh](#)

3.2 Kör deployment

```
chmod +x deploy-swarm.sh  
./deploy-swarm.sh
```

3.3 Resultat

Stacken deployades med:

- 3 nginx replicas (fördelade av Manager över noderna)
- 1 visualizer replica (placement constraint: endast Manager)

4. Testa och skala

4.1 Testa i webbläsare

Nginx: <http://54.154.62.190>

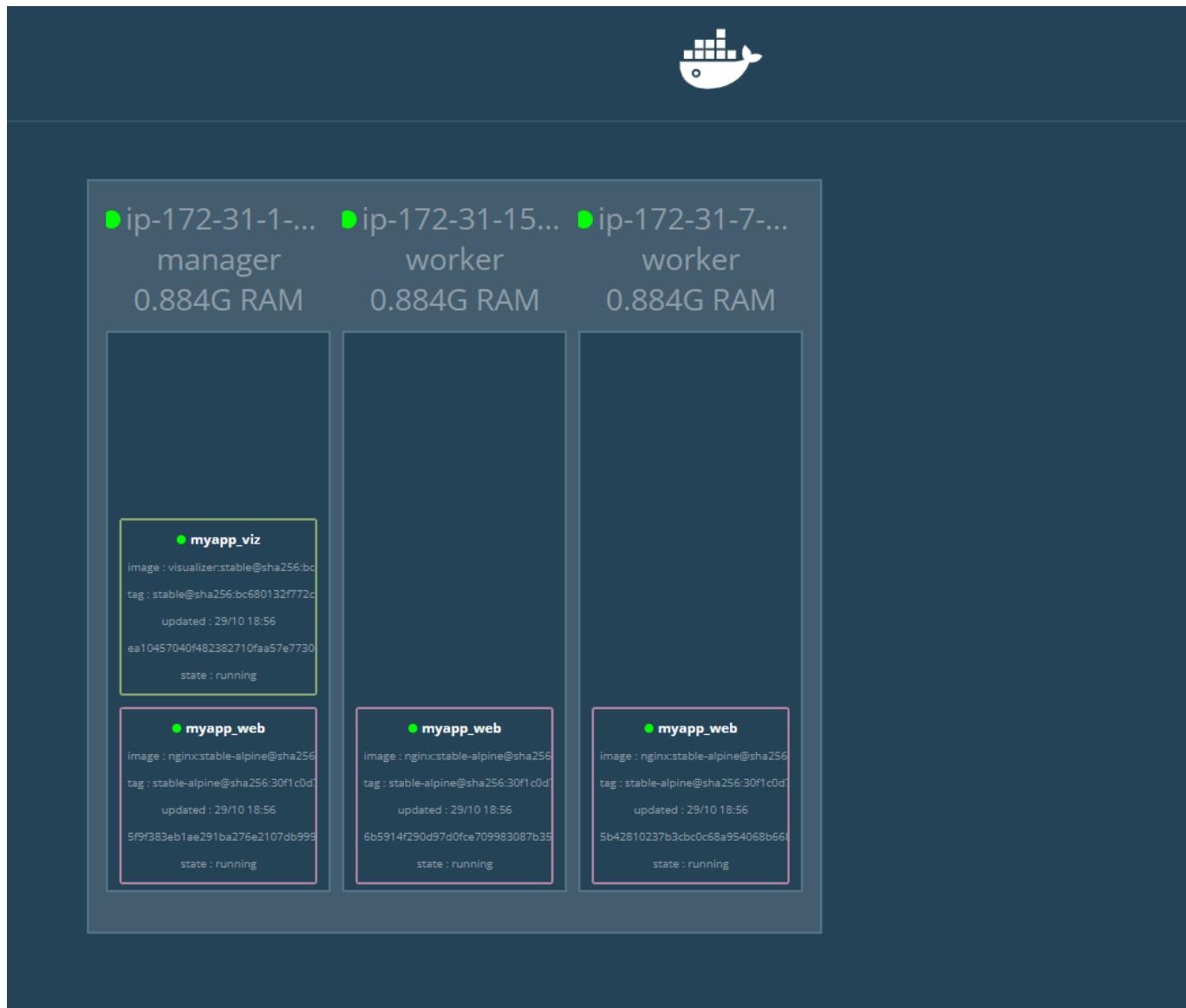
The screenshot shows a browser window with the following details:

- Address bar: <http://54.154.62.190>
- Page title: Welcome to nginx!
- Page content:

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.
- Browser tabs and icons: Classroom..., Mail - Maria Schillst..., My files - OneDrive, Meet, SQLZOO, SQL Tutorial W3, ChatGPT, Blank diagram: Luci...

Visualizer: <http://:8080/>



4.2 Skala services

```
# Skala upp
sudo docker service scale myapp_web=5

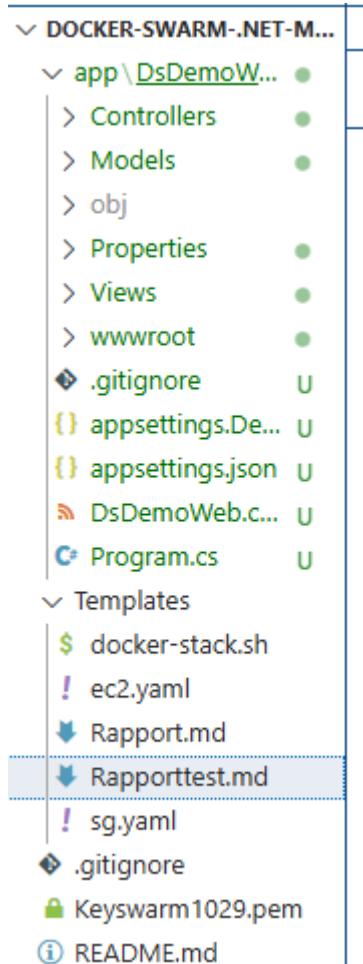
# Skala ner
sudo docker service scale myapp_web=3
```



5. Skapa och containerisera MVC-app

5.1 Skapa MVC-projekt

```
mkdir -p app && cd app  
dotnet new mvc -n DsDemoWeb -o DsDemoWeb  
cd DsDemoWeb  
dotnet new gitignore
```



Kommentera bort HTTPS-redirection i `Program.cs`.

5.2 Skapa Dockerfile

```
# Build & publish  
FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build  
WORKDIR /src  
COPY *.csproj ./  
RUN dotnet restore  
COPY ..  
RUN dotnet publish -c Release -o /app/publish  
  
# Runtime
```

```
FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS production
WORKDIR /app
EXPOSE 80
ENV ASPNETCORE_URLS=http://+:80
COPY --from=build /app/publish .
ENTRYPOINT ["dotnet", "DsDemoWeb.dll"]
```

5.3 Skapa ECR repository

```
AWS_REGION=eu-west-1
REPO=ds-demo-web

aws ecr create-repository --repository-name $REPO --region $AWS_REGION

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REPO_URI=${ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/${REPO}
```

5.4 Logga in till ECR

```
aws ecr get-login-password --region eu-west-1 | docker login --username AWS --
password-stdin ${ACCOUNT_ID}.dkr.ecr.eu-west-1.amazonaws.com
```

```
maria@ADMINIS-S7QQ8L0 MINGW64 ~/source/repos/Docker-Swarm-.NET-MVC-application-/app/DsDemoWeb (Dev)
$ aws ecr get-login-password --region $AWS_REGION \
> AWS_REGION=eu-west-1
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
aws ecr get-login-password --region $AWS_REGION \
| docker login --username AWS --password-stdin ${ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com

Unknown options: AWS_REGION=eu-west-1
Login Succeeded
```

6. Bygg och depolya MVC-appen

6.1 Bygg och pusha image

Problem: Multi-arch build (amd64+arm64) misslyckades med .NET 9 och QEMU (exit code 134).

Lösning: Byggde endast för amd64 eftersom alla EC2-instanser använder denna arkitektur.

```
docker buildx build \
--platform linux/amd64 \
-t ${REPO_URI}:v1 \
-f Dockerfile \
--push \
.
```

=> => pushing manifest for 542478884453.dkr.ecr.eu-west-1.amazonaws.com/ds-demo-web:v1@sha256:4c568412b2e2118a7c9f9778f62920e678a148c251e0ff5fa94366efda349f6b6

Verifiera i ECR:

Private repositories (1)			View push commands	Delete	Actions ▾	Create repository
Repository name	URI		Created at	▼	Tag immutability	Encryption type
ds-demo-web	542478884453.dkr.ecr.eu-west-1.amazonaws.com/ds-demo-web		September 11, 2025, 11:55:19 (UTC+02)		Mutable	AES-256

6.2 Uppdatera deployment-skript

Bytte ut nginx-imagen mot MVC-app från ECR:

```
# Förut:
image: nginx:stable-alpine

# Nu:
image: 542478884453.dkr.ecr.eu-west-1.amazonaws.com/ds-demo-web:v1
```

6.3 Åtgärda ECR-access

Problem: EC2-instanserna saknade behörighet att hämta imagen från privat ECR.

Lösning:

1. Raderade gamla EC2-stacken:

```
aws cloudformation delete-stack --stack-name swarm-ec2
aws cloudformation wait stack-delete-complete --stack-name swarm-ec2
```

2. Uppdaterade ec2.yaml med IAM Role ([EC2-ECR-Access](#)) och Instance Profile

3. Skapade ny stack med IAM:

```
aws cloudformation create-stack \
--stack-name swarm-ec2 \
--template-body file://templates/ec2.yaml \
--capabilities CAPABILITY_NAMED_IAM \
--parameters ParameterKey=SubnetId,ParameterValue=subnet-058684dd8c601d55d
```

4. Återskapade Swarm-klustret (steg 2.2-2.5)

5. Autentiserade Docker på alla noder (IAM-rollen propagerade inte direkt):

```
# På alla 3 noder:
aws ecr get-login-password --region eu-west-1 | sudo docker login --username AWS --password-stdin 542478884453.dkr.ecr.eu-west-1.amazonaws.com
```

6.4 Deploya MVC-appen

```
# Kopiera uppdaterat skript
scp -i Keyswarm1029.pem templates/docker-stack.sh ec2-user@<manager-ip>:~/deploy-swarm.sh

# SSH och kör
ssh -i Keyswarm1029.pem ec2-user@<manager-ip>
chmod +x deploy-swarm.sh
./deploy-swarm.sh
```

Verifiera:

```
sudo docker service ps myapp_web
sudo docker service logs myapp_web --tail 20
```

Testa i webbläsare:

- <http://> (MVC-app)
- <http://:8080/> (Visualizer)

Resultat: MVC-appen körs med 3 replicas, koordinerade av Manager över alla noder! 🎉

Sammanfattning

Vad som skapats

En komplett, skalbar Docker Swarm-miljö på AWS med:

- Infrastructure as Code (CloudFormation)
- Automatiserad deployment (Bash-skript)
- Säker ECR-integration (IAM)
- Containeriserad .NET MVC-applikation
- 3 replicas för hög tillgänglighet
- Orchestrering där Manager koordinerar container-placering

Lärdomar

- **IaC är kraftfullt:** Hela miljön kan återskapas på minuter
- **IAM är kritiskt:** Behörigheter måste vara på plats för privata registries
- **Swarm är enkelt:** Mindre komplext än Kubernetes för små/medium setups
- **Orchestrering ger kontroll:** Manager's koordinering säkerställer optimal fördelning
- **Multi-stage builds:** Minskar image-storlek betydligt

Bilaga: Att använda IaC Generator

Som jag nämnde tidigare har jag använt CloudFormation för att skapa vissa resurser. Nedan följer ett exempel på hur man använder IaC Generator. Principen är densamma från det att resursen man vill använda till templetten är klar.

Steg 1: Skapa Security Group manuellt

1.1 Grundinställningar

Namnge Security Group och ange beskrivning.

Inbound rules:

You have successfully created the template. Next, you can migrate to CDK or import to a stack.

SwarmSGTemplate

Template details

Template generation status: Complete

Template ID: arn:aws:cloudformation:eu-west-1:542478884453:generatedTemplate/424fb661-c241-429b-9080-ca4602c50e10

Configurations

Deletion policy: RETAIN

Update replace policy: RETAIN

Creation time: 2025-10-29 14:12:19 UTC+0100

Updated time: 2025-10-29 14:12:20 UTC+0100

Built off existing stack: -

Share your feedback on the generated template. Give feedback

Template definition | **Template resources** | **AWS CDK**

Template

Template | Canvas

```

1 ---  
2 Metadata:  
3   AWSToolsMetrics:  
4     IaC_Generator: "arn:aws:cloudformation:eu-west-1:542478884453:generatedTemplate/424fb661-c241-429b-9080-ca4602c50e10"  
5 Resources:  
6   EC2SecurityGroup:  
7     UpdateReplacePolicy: "Retain"  
8     Type: "AWS::EC2::SecurityGroup"  
9     DeletionPolicy: "Retain"  
10    Properties:  
11      GroupDescription: "Security group for Docker Swarm"  
12      GroupName: "SwarmSG"  
13      VpcId: "vpc-0b6849c800e4a6ff1"  
14      SecurityGroupIngress:

```

YAML | Download | Copy template | Import to stack | Copy

- **VPC:** Default
- **SSH:** Port 22 (rekommenderas att använda Your IP address)
- **HTTP:** Port 80, Source: 0.0.0.0/0
- **Custom TCP (Visualizer):** Port 8080, Source: 0.0.0.0/0

Jag glömde 8080 här, så fick göra det manuellt vid steg 4.1

Outbound rules:

- All traffic, Destination: 0.0.0.0/0

Lägg till tags om så önskas.

Create security group

The screenshot shows the 'Create security group' wizard in the AWS Management Console. The 'Basic details' section includes a security group name ('SwarmSG'), a description ('Security group for Docker Swarm'), and a VPC ('vpc-0b6849c800e4a6ff'). The 'Inbound rules' section contains three rules: one for SSH (TCP port 22) from 'My IP' to '91.128.144.160/32', one for HTTP (TCP port 80) from 'Cust...' to 'Q', and one for Custom TCP (TCP port 0) from 'Cust...' to 'Q'. The 'Outbound rules' section shows 'All traffic' to '0.0.0.0/0'. A note at the bottom warns against using 0.0.0.0/0 or ::/0 destinations. The 'Tags - optional' section adds a tag 'SwarmSG' with an optional value. The final step shows the 'Create security group' button.

Basic details

Security group name [Info](#)
SwarmSG
Name cannot be edited after creation.

Description [Info](#)
Security group for Docker Swarm

VPC [Info](#)
vpc-0b6849c800e4a6ff

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
SSH	TCP	22	My IP	91.128.144.160/32 X
HTTP	TCP	80	Cust... ▼	Q X
Custom TCP	TCP	0	Cust... ▼	Q X

[Add rule](#)

Outbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info
All traffic	All	All	Cust... ▼	0.0.0.0/0 X

[Add rule](#)

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
SwarmSG	X Enter value Remove

[Add new tag](#)
You can add up to 49 more tags

[Cancel](#) [Create security group](#)

Då ska det se ut så här:

⌚ Security group (sg-0f3e1b25482f1eb9d | SwarmSG) was created successfully X

▶ Details

sg-0f3e1b25482f1eb9d - SwarmSG

Actions ▾

Details			
Security group name SwarmSG	Security group ID sg-0f3e1b25482f1eb9d	Description Security group for Docker Swarm	VPC ID vpc-0b6849c800e4a6ff1 ⓘ
Owner 542478884453	Inbound rules count 3 Permission entries	Outbound rules count 1 Permission entry	

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

Inbound rules (3)

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0cf29e5fb0f48764e	IPv4	Custom TCP	TCP	0
<input type="checkbox"/>	-	sgr-0a3c66e2c10997282	IPv4	HTTP	TCP	80
<input type="checkbox"/>	-	sgr-0636f827e133609e6	IPv4	SSH	TCP	22

Manage tags
 Edit inbound rules
 < 1 > ⚙

1.2 Self-reference för Swarm-kommunikation

Gå in på **Edit inbound rules**. Nu ska vi referera SG till sig själv för intern Swarm-kommunikation.

Lägg till följande regler som alla refererar till samma Security Group:

- **Swarm Management:** Port 2377 (TCP)
- **Swarm Communication:** Port 7946 (TCP & UDP)
- **Overlay Network:** Port 4789 (UDP)

Spara

EC2 > Security Groups > sg-0f3e1b25482f1eb9d - SwarmSG > Edit inbound rules

Edit inbound rules Info

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

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0cf29e5fb0f48764e	Custom TCP	TCP	0	Cust... <small>Info</small>	<input type="text"/> 0.0.0.0/0 <small>X</small>
sgr-0a3c66e2c10997282	HTTP	TCP	80	Cust... <small>Info</small>	<input type="text"/> 0.0.0.0/0 <small>X</small>
sgr-0636f827e133609e6	SSH	TCP	22	Cust... <small>Info</small>	<input type="text"/> 91.128.144.160/32 <small>X</small>
-	Custom TCP	TCP	2377	Cust... <small>Info</small>	<input type="text"/> sg-0f3e1b25482f1eb9d <small>X</small> Management
-	Custom TCP	TCP	7946	Cust... <small>Info</small>	<input type="text"/> sg-0f3e1b25482f1eb9d <small>X</small> Communication
-	Custom UDP	UDP	7946	Cust... <small>Info</small>	<input type="text"/> sg-0f3e1b25482f1eb9d <small>X</small> Communication
-	Custom UDP	UDP	4789	Cust... <small>Info</small>	<input type="text"/> sg-0f3e1b25482f1eb9d <small>X</small>

[Add rule](#)

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

[Cancel](#) [Preview changes](#) [Save rules](#)

Tips: Skriv ner Resource identifier - den behövs till IaC Generator.

Steg 2: Skapa CloudFormation template med IaC Generator

2.1 Navigera till IaC Generator

Gå till **CloudFormation → IaC Generator**

The screenshot shows the AWS CloudFormation IaC Generator interface. On the left, there's a sidebar with navigation links for CloudFormation, Infrastructure Composer, Hooks, Registry, and Spotlight. The main area is titled 'Stacks (0)' and contains a search bar, filter status dropdown (set to 'Active'), and a 'Create stack' button. A message 'No stacks' indicates there are no stacks to display.

2.2 Starta scan

Starta en **new scan** och välj **Scan specific resource**

The screenshot shows the AWS CloudFormation IaC generator interface. On the left, there's a sidebar with navigation links like Stacks, StackSets, Exports, Infrastructure Composer, IaC generator (which is selected), Hooks overview, Invocation summary, and Hooks. Below these are sections for Registry (Public extensions, Activated extensions, Publisher), Spotlight, and Feedback.

IaC generator

How it works

IaC generator makes it easy to generate infrastructure as code (IaC) configuration files, such as CloudFormation templates, from resources provisioned in your account. [Learn more](#)

Step 1. Scan account resources

Scan your account to discover existing resources. You can scan all resource types and their related resources, or scan specific types.

[Start a new scan](#)

[Scan all resources](#)

[Scan specific resources](#)

Step 2. Create CloudFormation template

Search and select the existing resources you want to include in your template. The tool will suggest related resources that should also be included.

[Create template](#)

Step 3. Import to CloudFormation or CDK

Open a generated template to import resources into CloudFormation or convert the template into a CDK app.

[View templates](#)

Scans

Scan your provisioned resources (all or by specific resource types) before creating templates. A scan is valid for 30 days and cannot be used for template generation after this. It can take up to 10 minutes for 1,000 resources.

Last completed scan: 2025-10-28 13:53:10 UTC+0100

Last attempted scan: 2025-10-28 13:53:10 UTC+0100

Scan status: Complete

Days until expiry: 29

Resources scanned: 1

Selected scan ID (active): arn:aws:cloudformation:eu-west-1:54247884453:resourceScan/36367092-6e59-4c39-af5b-8352762ac10d

Scan type: Partial scan

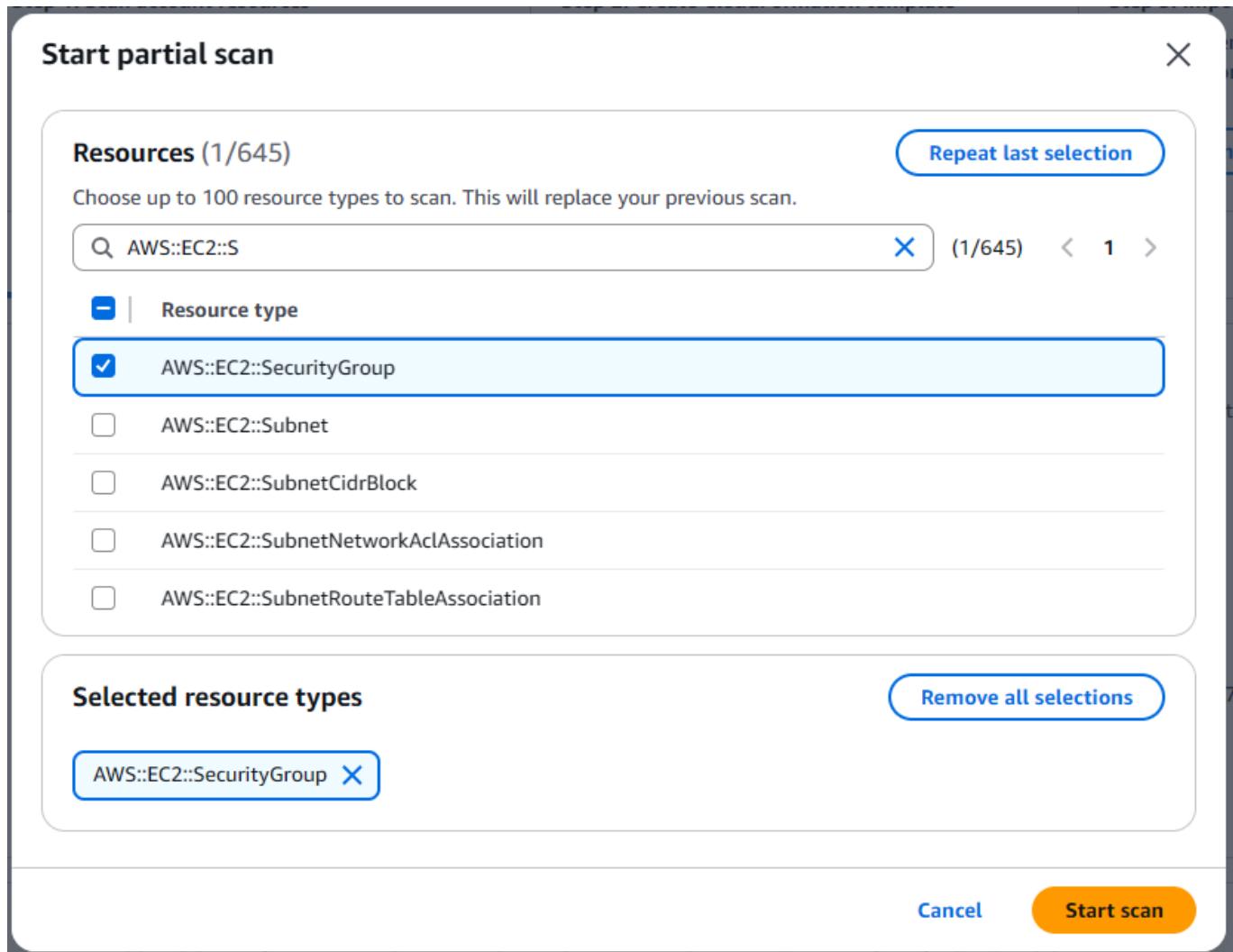
Scanned resources breakdown 1

View a visual breakdown of your scanned resources by product types. Select a product type for a further breakdown.

Filter displayed data: Filter by product type ▾

2.3 Välj Security Group

Sök efter SG, bocka i din Security Group och klicka **Start scan**



2.4 Skapa template

När scan är klar, klicka **Create template**

✓ IaC generator resource scan is complete.

[Create template](#) [X](#)

IaC generator

How it works

IaC generator makes it easy to generate infrastructure as code (IaC) configuration files, such as CloudFormation templates, from resources provisioned in your account. [Learn more ↗](#)

Step 1. Scan account resources
Scan your account to discover existing resources. You can scan all resource types and their related resources, or scan specific types.
[Start a new scan ▾](#)

Step 2. Create CloudFormation template
Search and select the existing resources you want to include in your template. The tool will suggest related resources that should also be included.
[Create template](#)

Step 3. Import to CloudFormation or CDK
Open a generated template to import resources into CloudFormation or convert the template into a CDK app.
[View templates](#)

[Scan summary](#) [Templates](#)

Scans

Scan your provisioned resources (all or by specific resource types) before creating templates. A scan is valid for 30 days and cannot be used for template generation after this. It can take up to 10 minutes for 1,000 resources.

Last completed scan
2025-10-29 14:04:34 UTC+0100

Last attempted scan
2025-10-29 14:04:34 UTC+0100

Scan status
✓ Complete

Days until expiry
30

Resources scanned
3

Selected scan ID (active)
arn:aws:cloudformation:eu-west-1:542478884453:resourceScan/735b70bd-43a9-4523-a0ba-304ca640bec6

Scan type
Partial scan

Scanned resources breakdown 3

View a visual breakdown of your scanned resources by product types. Select a product type for a further breakdown.

Filter displayed data
[Filter by product type ▾](#)

2.5 Namnge template

Namnge templet och välj **Next**

The screenshot shows the 'Specify template details' step of the CloudFormation wizard. A sidebar on the left lists steps: Step 1 (selected), Step 2, Step 3, Step 4. The main area has a green header bar with the message 'IaC generator resource scan is complete.' and a 'Create template' button. Below is a 'Prerequisite - Prepare template' section with a note about updating existing stacks. It offers two options: 'Start from a new template' (selected) and 'Update the template for an existing stack'. The 'Provide template details' section includes fields for 'Template name' (SwarmSGTemplate), 'Deletion policy' (Retain), and 'Update replace policy' (Retain). At the bottom right are 'Cancel' and 'Next' buttons.

2.6 Välj rätt resource

Nu får du upp alla SG du har skapat. Se till att välja rätt **Resource identifier**.

The screenshot shows the AWS CloudFormation 'Create template' wizard at Step 2: 'Add scanned resources'. A green banner at the top says 'IaC generator resource scan is complete.' A 'Create template' button is in the top right.

Step 1: Specify template details
Step 2: Add scanned resources (selected)
Step 3: Add related resources
Step 4: Review and create

Add scanned resources

Scans

Last completed scan: 2025-10-29 14:04:34 UTC+0100 | Days until expiry: 30

Last attempted scan: 2025-10-29 14:04:34 UTC+0100 | Resources scanned: 3

Scan status: Complete | Selected scan ID (active): arn:aws:cloudformation:eu-west-1:54247884453:resourceScan/735b70bd-43a9-4523-a0ba-304ca640bec6

Scan type: Partial scan

Add scanned resources

We recommend organizing resources across templates by lifecycle and ownership. For example, consider having a dedicated template for resources that change more often than others or resources that persist data. For more information, see [AWS CloudFormation best practices](#).

Resource identifier	Resource type	Managed by stack
sg-06922e100c207807c	AWS::EC2::SecurityGroup	No
sg-0f3e1b25482f1eb9d	AWS::EC2::SecurityGroup	No
sg-055114f12ac511c2e	AWS::EC2::SecurityGroup	No

Selected resources 1

Resource identifier	Resource type
{"id": "sg-0f3e1b25482f1eb9d"}	AWS::EC2::SecurityGroup

Cancel Previous Next

Tryck **Next** på kommande två fönster.

2.7 Slutför

Klicka **Create Template**

Review and create

Step 1: Specify template details

[Edit](#)

Template name	Deletion policy
SwarmSGTemplate	RETAIN
Update replace policy	Base stack name
RETAIN	-
Base stack ID	
-	

Step 2: Add scanned resources

[Edit](#)

Scanned resources		
Resource identifier	Resource type	Managed by stack
sg-0f3e1b25482f1eb9d	AWS::EC2::SecurityGroup	No

Step 3: Related resources

[Edit](#)

Add related resources		
Resource identifier	Resource type	
No related resources		

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

Steg 3: Parametrisera och spara

Nu har templetten skapats med ett långt skript.

Tips: Parametrisera skriptet för återanvändbarhet. Det kan vara knepigt i början, så då kan man be LLM om hjälp.

Välj att spara ner skriptet - då kan du använda det flera gånger som det är eller med ändringar.

The screenshot shows the AWS CloudFormation IaC Generator interface. At the top, a green banner indicates: "You have successfully created the template. Next, you can migrate to CDK or import to a stack." Below this, the template name "SwarmSGTemplate" is displayed. The "Template details" section includes fields for "Template generation status" (Complete), "Template ID" (arn:aws:cloudformation:eu-west-1:542478884453:generatedTemplate/424fb661-c241-429b-9080-ca4602c50e10), "Creation time" (2025-10-29 14:12:19 UTC+0100), and "Updated time" (2025-10-29 14:12:20 UTC+0100). The "Configurations" section shows "Deletion policy" (RETAIN) and "Update replace policy" (RETAIN). On the right, it says "Built off existing stack" with a note "-". Action buttons include "Import to stack", "Edit", and "Delete". A feedback section at the bottom left encourages users to share their feedback, with a "Give feedback" button.

Template definition

```

1 ---  
2 Metadata:  
3   AWSToolsMetrics:  
4     IaC_Generator: "arn:aws:cloudformation:eu-west-1:542478884453:generatedTemplate/424fb661-c241-429b-9080-ca4602c50e10"  
5 Resources:  
6   EC2SecurityGroup:  
7     UpdateReplacePolicy: "Retain"  
8     Type: "AWS::EC2::SecurityGroup"  
9     DeletionPolicy: "Retain"  
10    Properties:  
11      GroupDescription: "Security group for Docker Swarm"  
12      GroupName: "SwarmSG"  
13      VpcId: "vpc-0b6849c800e4a6ff1"  
14      SecurityGroupTags:  


```

Template

YAML ▾ Download Copy template Import to stack

Copy

OBS! Se till att radera stacken under CloudFormation samt SG innan du kör skriptet, eller uppdatera templet med andra namn för att undvika konflikter.

Sammanfattning av IaC Generator-processen

- Skapa resurs manuellt i AWS Console
- Använd IaC Generator för att scanna resursen
- Generera CloudFormation template
- Parametrisera templetten
- Spara och återanvänd för framtida deployments

Fördel: Får korrekt CloudFormation-syntax direkt från befintlig resurs, vilket minskar risken för fel och sparar tid.

Tips, deploya till swarm i produktionsmiljö, inte för utveckling