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

Översikt av lösningen

Arkitektur

Note: Tutorialen för denna uppgift 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.

För denna labb antar vi att du bygger klart hela MVC-applikationen innan steget för Deploy.

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

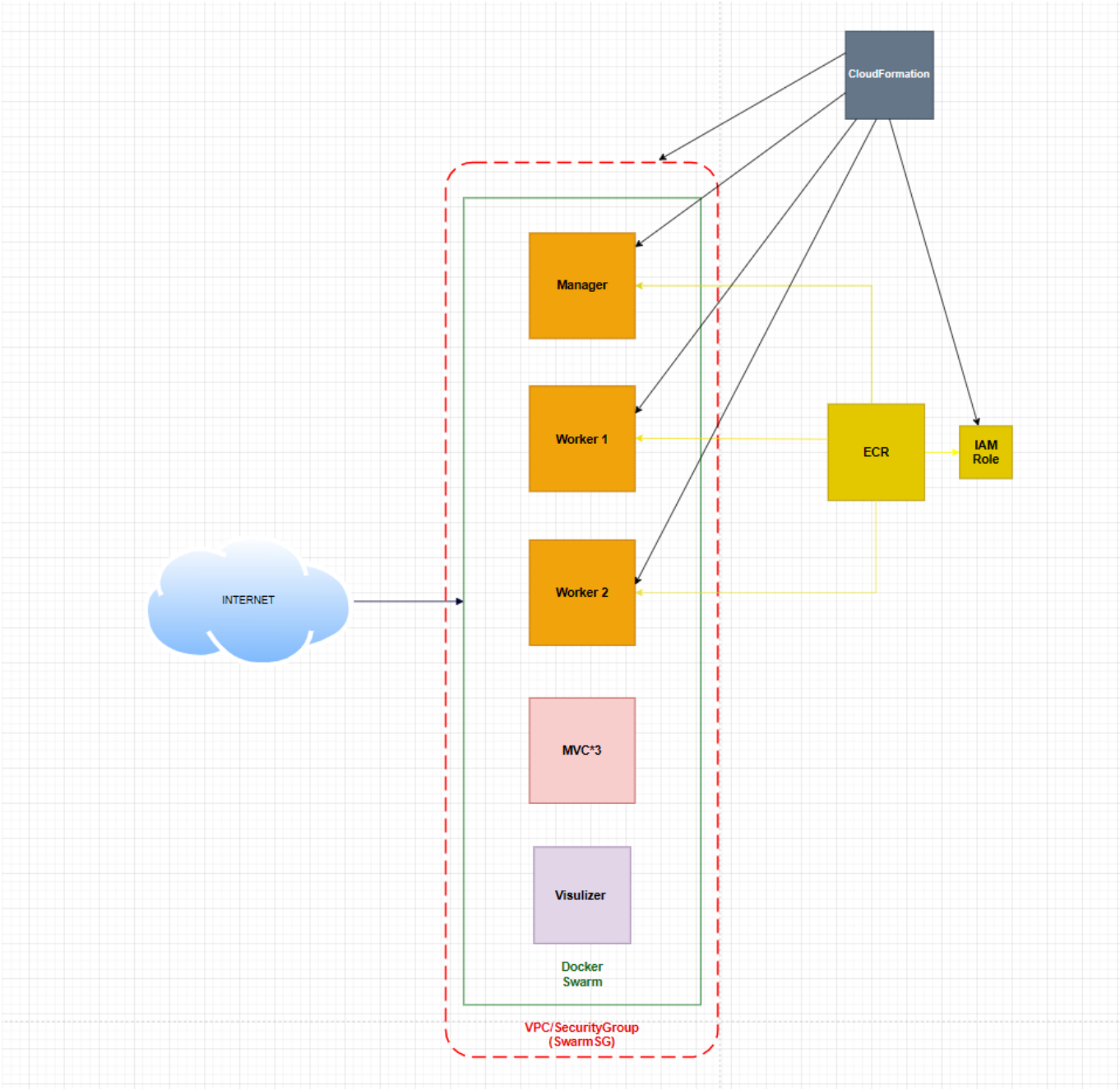
Infrastruktur:

- 3 EC2-instanser på AWS (1 Manager + 2 Workers)
- Docker Swarm orchestrerar 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 verklig webbapplikation
- Visar att Swarm kan hantera stateful applikationer
- Containeriseras enkelt med Dockerfile

Kompletta instruktioner samt templates finns på https://github.com/MariaSchillstrom/Docker-Swarm-.NET-MVC-application-.git