Ejer:
Godkender:
Revisionsdato:

Version:
Godkendelsesdato:

Linux Containers

OS2-fællesskabet linux docker linux container

Version 1.0.0

Dato: 21-05-2019

1. Indledning

Mange OS2 produkter leveres som Docker images, der kan afvikles på en Linux server. Dette dokument er tænkt som en introduktion og quick-start guide for Windows administratorer, så man nemt og smertefrit kan komme i gang med at bruge både Docker og Linux servere.

1.1 Hvad er Docker?

Docker har 3 begreber som er vigtige at beskrive inden Docker beskrives i yderligere detaljer.

- **Docker Host**. En egentlig server (typisk en Linux server, men kan også være en Windows Server) der afvikler Docker
- **Docker Image**. En samlet pakke indeholdende software, afhængigheder, konfiguration m.m.
- **Docker Container**. Når et Image startes, skabes en kørende instans, som kan ændre tilstand (fx kan data skabes, konfigurationen kan ændres m.m.). Denne kørende instans kaldes for en Container.

En helt forkert (men ganske god) måde at beskrive Docker, er som en letvægts hypervisor til virtuelle maskiner. Docker softwaren installeres på en server (Windows eller Linux server) som så kan afvikle såkaldte Docker Images/Containers (som kan sammenlignes med virtuelle maskiner).

Docker er dog ikke en hypervisor, og Docker Images er ikke virtuelle maskiner, men fra et anvenderperspektiv er forskellen lille.

Den største forskel på Docker og et klassisk hypervisor/VM setup er at Docker Containere ”arver” operativsystemet fra den Docker Host som afvikler Docker Imaget.

I praksis betyder det at man ikke skal boote et operativsystem for at tænde for en Docker Container (modsat en virtuel maskine, hvor man skal boote operativsystemet først), hvilket gør det muligt at tænde for et Docker Image på millisekunder frem for sekunder/minutter.

Men det betyder også at Docker Images er ”begrænset” til at køre det operativsystem som Docker Host maskinen kører. Det er dog en sandhed med modifikationer, da der findes forskellige tricks til at afvikle Linux-baserede Docker Containers på Windows servere.

1.1.1 Hvorfor leveres OS2-produkterne på Docker?

Det er altid et mål med OS2 produkterne at gøre dem så lette at implementere som muligt, hvilket Docker hjælper med til at understøtte.

Leverandøren af OS2 produktet kan indkapsle alle de nødvendige afhængigheder og konfigurationer i et Docker Image, så en kommune blot kan tage imaget og deploye det i deres lokale Docker Host miljø (igen på samme måde som man kunne deploye en egentlig virtuel maskine).

Den kørende Docker Container vil så være helt afkoblet fra andre Docker Containers, og man kan på den måde nemt have software der kræver forskellige (konfliktende) udgaver af Java, .NET, mærkelige DLL filer, m.m. installeret på samme server. Man er også sikker på at softwaren virker uden at der skal installeres andet, da Docker Imaget kommer med alt hvad der skal bruges.

Docker kommer med en lang række værktøjer der gør det nemt at pakketere, opdatere og distribuere Docker Images. Bl.a. kan man finde rigtigt mange standard sofware pakker som Docker Images på Docker Hub.

1.2 Docker på Windows

Docker kan som nævnt anvendes på både Linux og Windows servere, men da Docker Containers arver operativsystemet fra den server hvor Docker Host kører, er man som udgangspunkt begrænset til at køre Windows Containers på en Windows Server, og Linux Containers på en Linux Server.

Langt de fleste Docker Images (fx næsten alle dem der findes på Docker Hub), er baseret på Linux, og skal afvikles via Docker Host på en Linux Server.

Det har dog siden Windows Server 2019 været muligt at køre Linux-baserede containers (og egentligt også på Windows Server 2016, omend det anbefales ikke til produktion).

Dette gøres via virtualiseringsteknologi, hvor Windows laver en virtuel Linux, som anvendes til at afvikle Linux-baserede containers. Man mister noget af fordelen ved Docker på den måde, da man så har et virtualiseret Linux operativsystem der skal bootes, samt endnu et lag i ens driftmodel.

Docker følger med Windows Server, og kan enables via Server Management konsollen i Windows. Man skal enable den feature der hedder ”Containers”, hvorefter Docker er tilgængelig på Windows Serveren.

Man skal dog være opmærksom på at Docker enten skal køre i Windows-mode (default) eller i Linux-mode (ikke default, og kræver installation af yderligere software).

Denne vejledning dækker ikke anvendelsen af Docker på Windows, men afsnittet omkring omkring brugen af Docker vil også være gældende for Docker på Windows, der vil blot være en anden opsætning og konfiguration der gør sig gældende på Windows platformen.

1.3 Docker på Linux

Docker anvendes i stort omfang på Linux platformen, og leverandører af software distribuerer ofte deres software som Docker images, fx kan man på Docker Hub finde produkter fra leverandører som

- Citrix
- Microsoft
- Oracle

Leverandører af infrastruktur appliances vil typisk være at finde på Docker Hub, samt selvfølgelig OS2, der også anvender Docker Hub til distribution af OS2 produkterne.

Den nemmeste måde at komme igang med Docker, er at etablere en Linux server og installere Docker på den. Herefter er den klar til at afvikle software fra en lang række leverandører, på en nem og smertefri måde.

De efterfølgende afsnit beskriver hvordan man nemt kan komme igang med Docker på Linux, og efterfølgende hvordan man så etablerer et solidt produktions-setup omkring anvendelsen af Docker.

2. Opsætning af Linux servere

2.1 Lokal Linux server

Hvis man ønsker at starte Linux servere op lokalt (enten på egen maskine, eller i eget driftcenter), skal man starte med at vælge den Linux distribution som passer til ens behov.

Da Docker Images kommer med alle de afhængigheder som en software applikation måtte have, er der ikke behov for at installere andet på Linux Serveren end Docker. Dette understøtter en brug-og-smid-væk strategi når det kommer til Linux, hvilket gør vedligehold af Linux servere til en forholdsvis simpel ting.

I stedet for at holde en Linux server opdateret med nyeste patches og sikkehedsopdateringer, så kan man vælge en alternativt strategi, hvor man bare slukker for Linux serveren (smide den væk), og så tænder for en frisk Linux server, som allerede er patchet med nyeste sikkerhedsopdateringer. Her installers så Docker, og man er kørende igen.

Modsat Windows kommer Linux i mange forskellige varianter, kaldet distributioner. En distribution vedligeholdes af et firma eller et fællesskab, der sikrer at distributionen holdes opdateret med sikkerhedspatches, samt nye versioner af frameworks, management-værktøjer m.m.

Gode valg kune være (men er ikke begrænset til)

- Ubuntu LTS releases (LTS = Long Term Support)
- Debian
- Red Hat Enterprise Linux

Fælles for disse er at de typisk opdaterer det ISO-image man installerer Linux fra hver 1-2. måned, så man er sikker på at man installerer en version der er opdateret med nyeste sikkerhedspatches m.m.

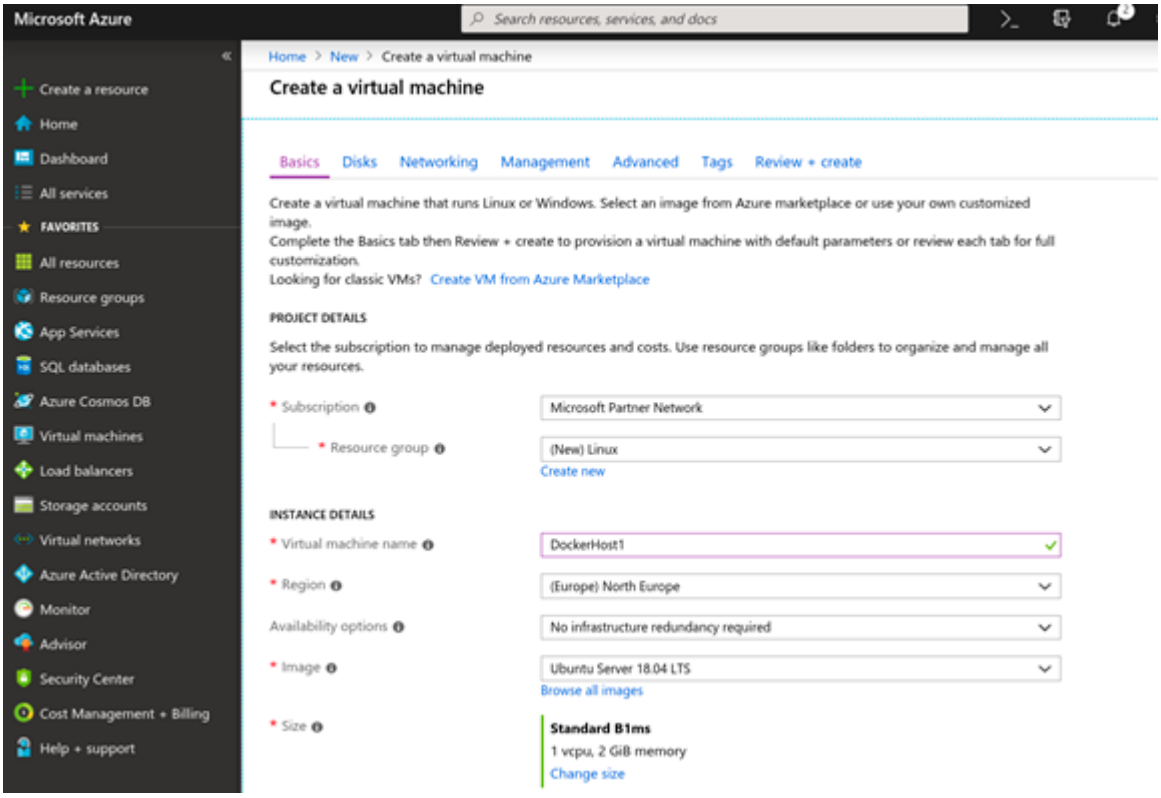
Dette understøtter brug-og-smid-væk strategien, hvor man så starter nye Linux server op når der kommer nye patch-releases til den udgave af Linux man kører, og smider de gamle Linux servere væk.

Man kan med fordel lave et installations-script, der hiver nyeste Linux release, installerer, og så kører Docker på, og starter containerer op derpå, og så sætte det til at afvikle efter behov, for på den måde at minimere egentlig server-administration.

2.2 Linux server i Azure

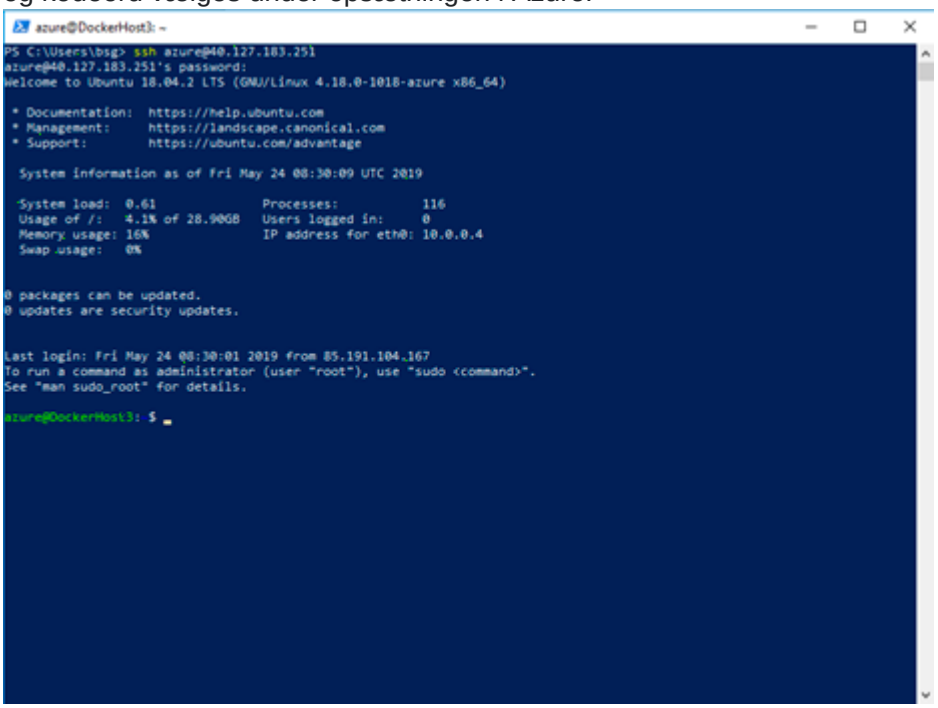
Den samme strategi som er beskrevet ovenfor, kan anvendes i Azure. Her får man dog noget af arbejdet foræret, da Azure altid tilbyder at man kan starte nye virtuelle maskiner op, med nyeste/patchede udgave af Linux, så man slipper for at holde øje med hvornår der er opdateringer til Linux, og redeployment er ”bare” at starte en ny Linux server i Azure.

Linux servere startes på samme måde som andre virtuelle servere, man vælger blot den udgave af Linux som man ønsker at anvende. I nedenstående screenshot er der valgt Ubuntu 18.04 LTS.



For at logge på en Linux server, skal man anvende SSH, hvilket fx. kan gøres direkte fra Powershell, eller fra andre SSH værktøjer som Putty eller lignende.

Nedenstående screenshot viser forbindelsen til en Linux server i Azure via SSH kommandoen i powershell. Brugernavn og kodeord vælges under opsætningen i Azure.

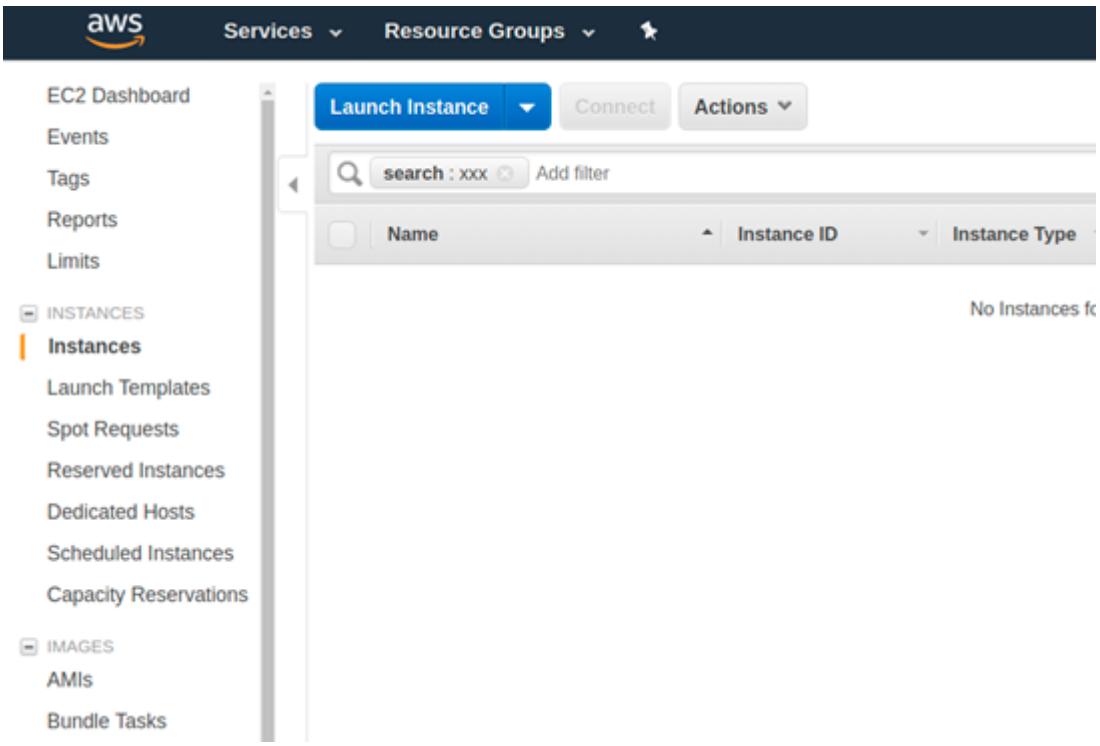


2.3 Linux server i AWS

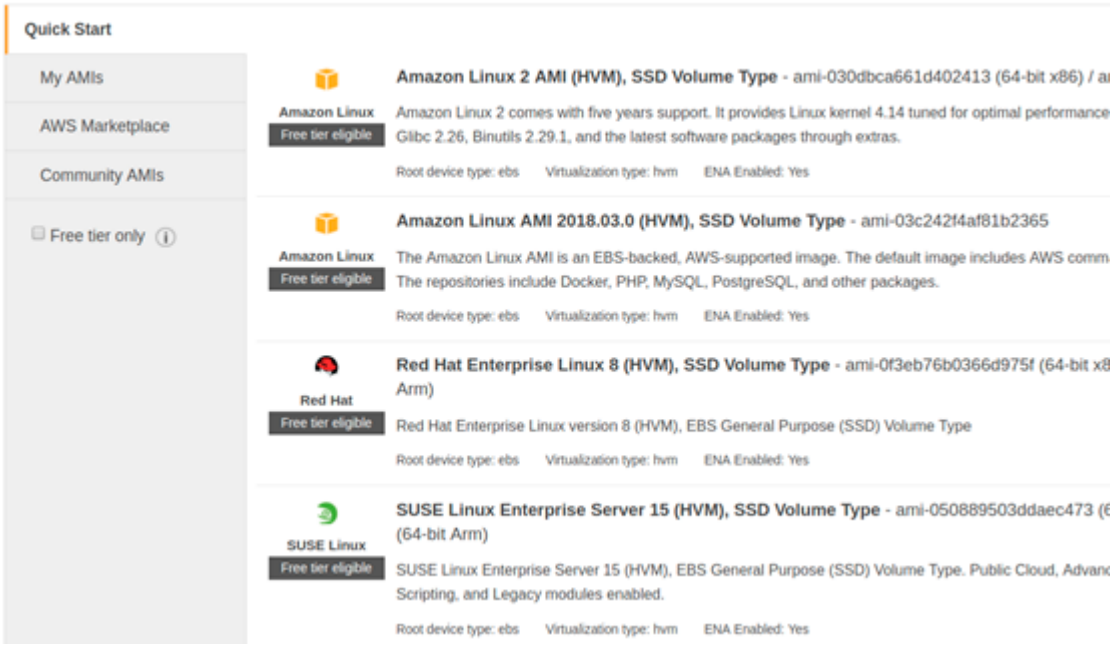
Amazon AWS tilbyder samme funktionalitet som Azure, og Amazon vedligeholder deres egen distribution af Linux (kaldet Amazon Linux), som Amazon er ansvarlige for at holde opdateret med sikkerhedsopdateringer m.m.

Processen minder meget om den i Azure, dog kan man ikke vælge at forbinde til servere med brugernavn/kodeord, men skal altid forbinde med en såkaldt SSH nøgle. Denne kan dannes (og downloades) inde fra AWS portalen.

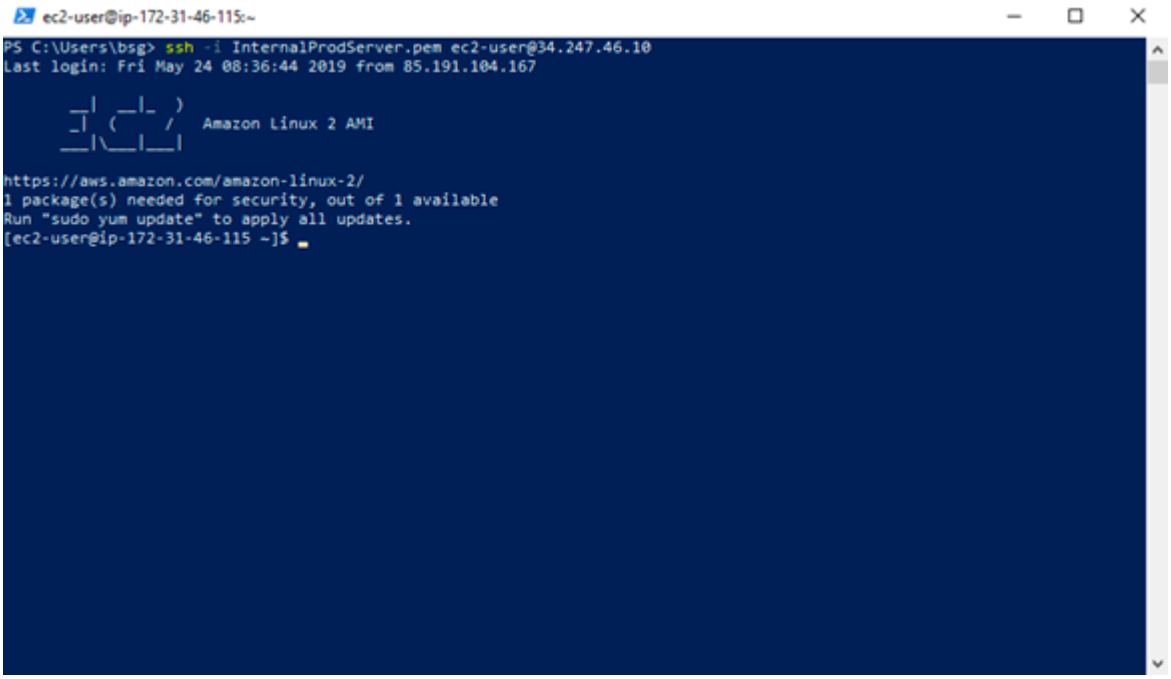
Når man starter en ny Linux server, gøres det inde fra EC2 dashboardet, hvor man vælger ”Launch Instance” som vist nedenfor



Her kan man vælge mellem de forskellige typer af Linux, hvor Amazon Linux vil være et godt valg i de fleste tilfælde.



Som med Linux servere i Azure, forbinder man til dem via SSH. I Azure har man valget mellem brugernavn/kodeord og SSH nøgler, og i Amazon AWS kan man kun vælge SSH nøgler. Nedenfor er vist et screenshot fra Powershell, hvor SSH nøglen anvendes (InternalProdServer.pem er nøglen, og kan dannes og downloades inde fra Amazon konsollen når man starter Linux serveren).



3. Installation af Docker

Ejer: Version:

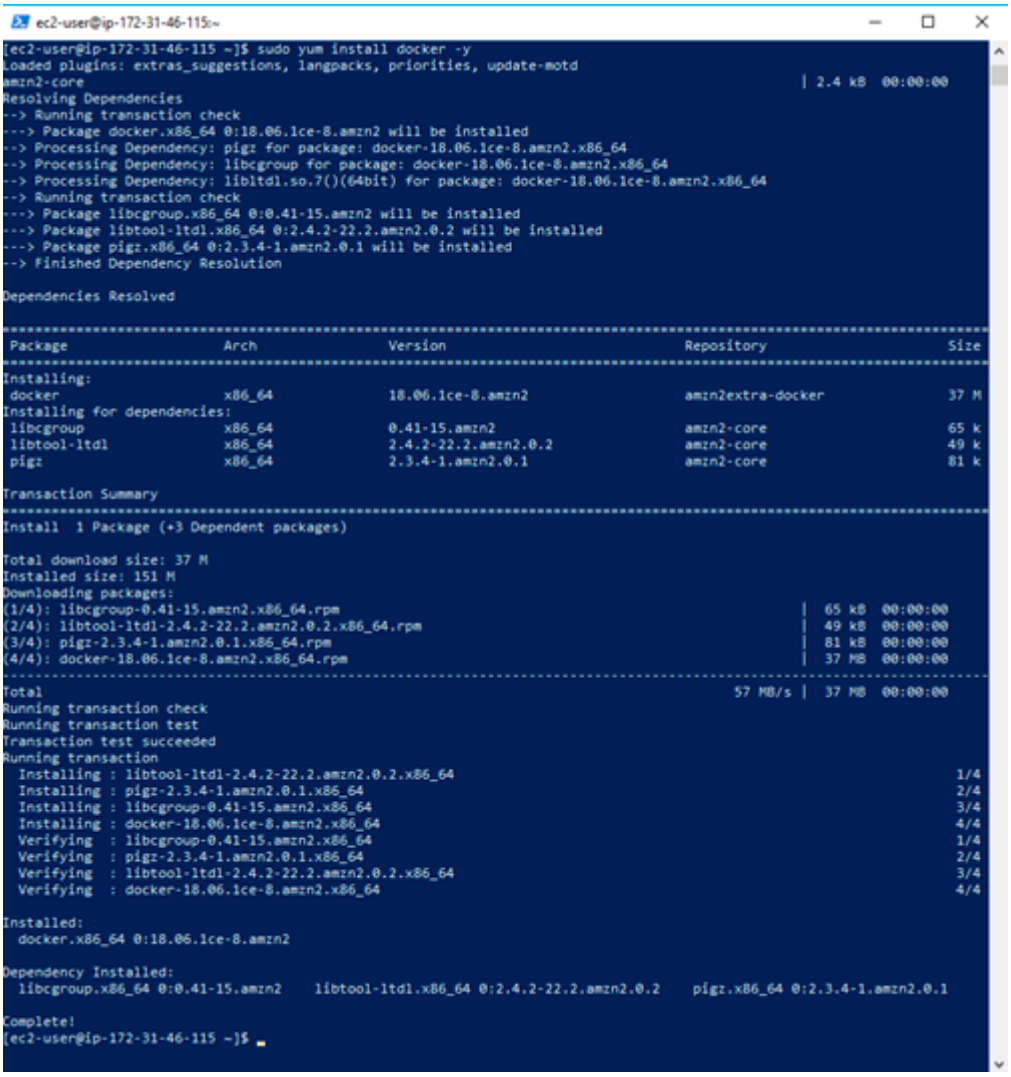
Godkender: Godkendelsesdato:

Revisionsdato:

Linux kommer ikke normalt med Docker installeret som default, så det første man skal gøre er at installere Docker. Dette klares via det værktøj som Linux bruger til at installere og opdatere software. Typisk er dette **apt-get** eller **yum**. Nedenfor er vist hvordan man installere Docker vha værktøjet yum (apt-get har en identisk syntax)

Først installeres Docker softwaren med kommandoen

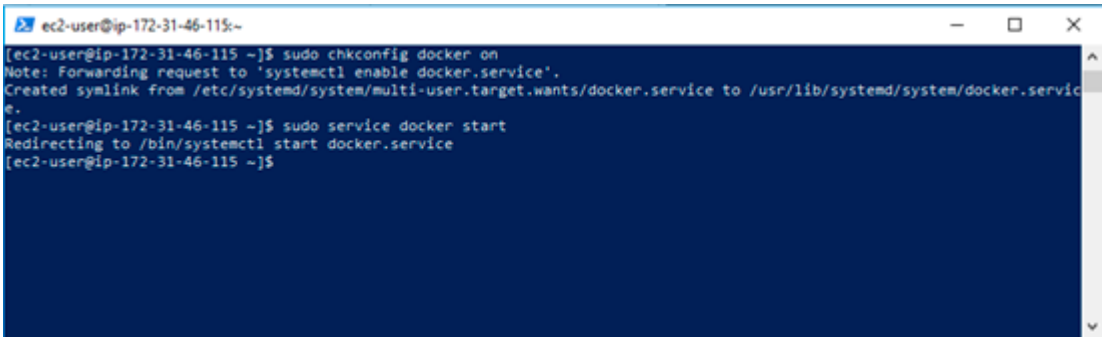
$ sudo yum install docker -y



Dernæst konfigureres Docker til at starte automatisk ved server genstart (chkconfig kommandoen), og så startes servicen (service kommandoen)

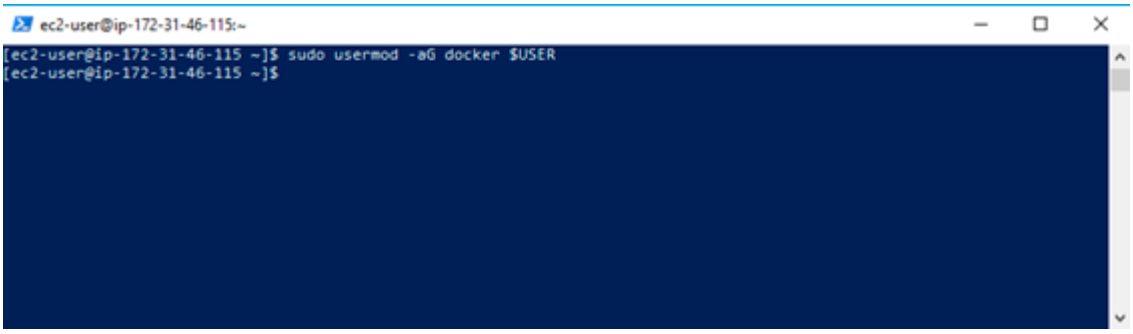
$ sudo chkconfig docker on

$ sudo service docker start



Endeligt gives den påloggede bruger adgang til at anvende Docker (så man slipper for at køre alle kommandoerne som administrator via sudo kommandoen)

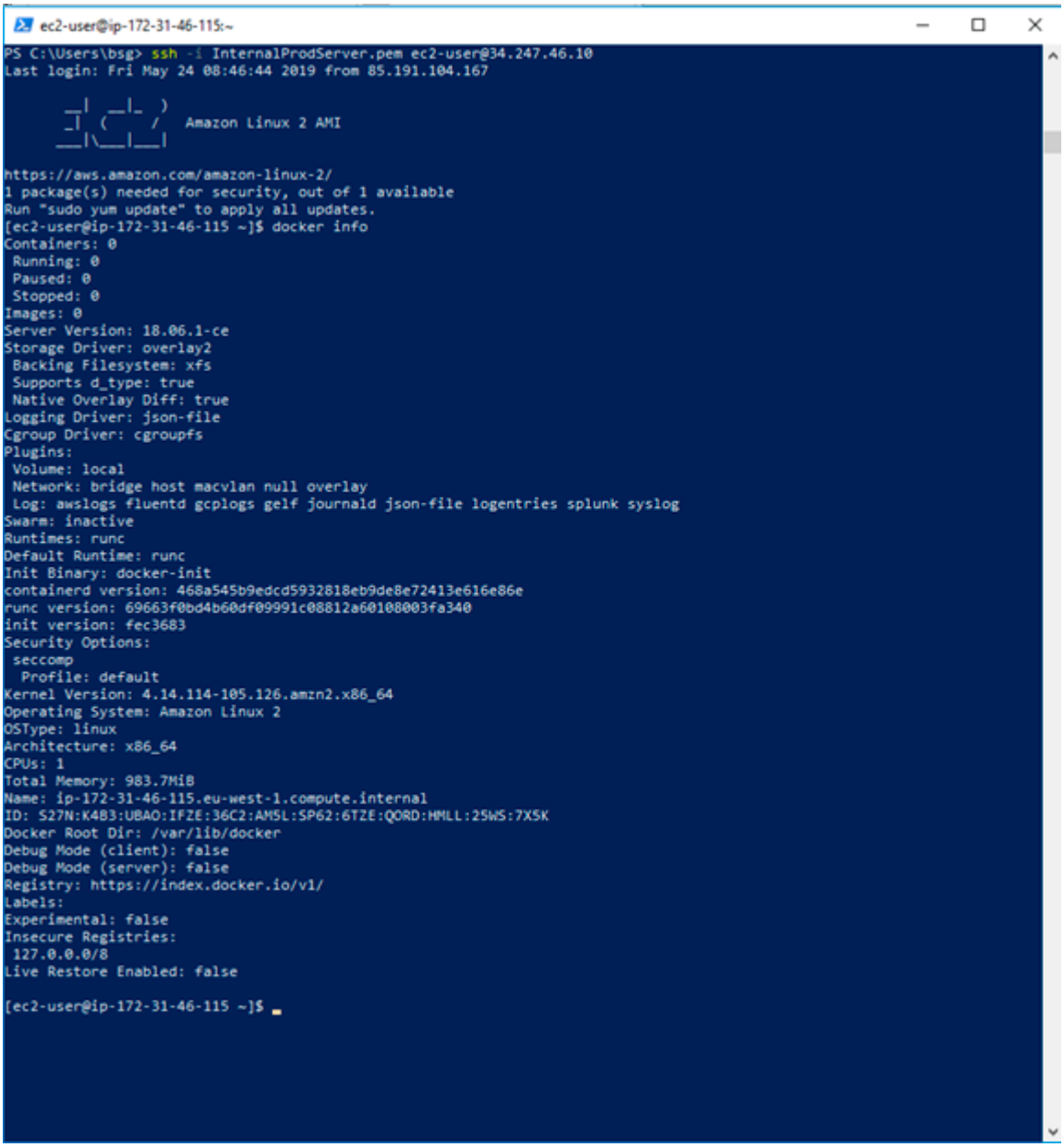
$ sudo usermod -aG docker $USER



Herefter skal man logge ud fra serveren (luk SSH vinduet, eller bare skriv ”exit”), og så logger man på igen. Først da har man fået tildelt de fulde adgange til Docker.

Herefter kan man kontrollere at Docker er kørende, og at man har adgang, ved at køre kommandoen

$ docker info



4. Værktøj til administration af Docker

En ren Docker installation giver adgant til kommando-linjeværktøjet ”docker”, der kan bruges til at starte, stoppe og få status på Docker Containere. Konfigurationsparametre, fildrev, m.m. som skal bruges sendes som kommandolinje-parametre, hvilket gør det en smule besværligt at bruge Docker i sin ”rå” form.

Heldigvis findes der en række forskellige værktøjer ovenpå Docker, som gør det betydeligt lettere at administrere sine Docker Containere.

4.1 Docker Compose

Docker Compose er et kommand-linje værktøj, som gør det lettere at administrere sine Docker Containere. I stedet for at sende konfigurationen til en Docker container via kommando-linjen, så arbejder Docker Compose med konfigurationsfiler, og så starter/stopper man alle Docker Containers som er listet i konfigurationsfilen på én gang.

Nedenfor er vist et eksempel på en Docker Compose konfigurationsfil, der starter både en MySQL database, samt en web-applikation der gør brug af databasen

---

version: "2.0"

services:

mysql:

image: mysql:5.7

environment:

MYSQL_PASSWORD: "pwd"

MYSQL_USER: "dbo"

MYSQL_DATABASE: "orgsync"

stsorgsync:

image: stsorgsync:1.3.0

environment:

ConnectionString: "server=mysql;user id=dbo;password=pwd;database=orgsync"

Environment: "TEST"

Konfigurationsfilen til Docker Compose er indelt i ”services”, hvor hver service er den samlede konfiguration for netop én Docker Container. I ovenstående konfigurationsfil, er der specificeret 2 Docker Containers, én der starter en MySQL

database og én der starter STSOrgSync.

Man kan læse mere om Docker Compose konfigurationsformatet her

https://docs.docker.com/compose/

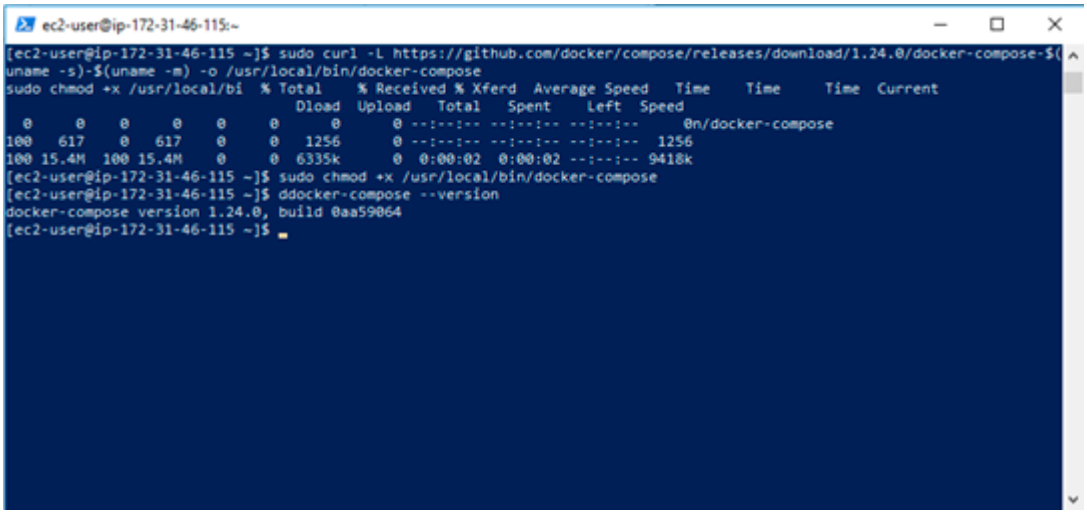
Typisk vil man modtage en Docker Compose konfigurationsfil fra sin leverandør, som indeholder en standardopsætning. Opsætningen kan man så tilpasse hvis man har nogle særlige behov.

4.1.1 Installation af Docker Compose

Docker Compose installeres med følgende 2 kommandoer på Linux serveren

$ sudo curl -L https://github.com/docker/compose/releases/download/1.24.0/docker-compo…-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose



Docker Compose værktøjet er nu klar til brug, og man kan verificere ved at skrive ”docker-compose --version” i terminalen.

4.1.2 Brug af Docker Compose

Med Docker Compose får man samlet hele sin konfiguration i én fil, og man kan starte og stoppe alt i konfigurationsfilen med følgende simple kommandoer

$ docker-compose up -d

$ docker-compose down

Parametren -d der sendes til ”up” kommandoen sørger for at alle Docker Containers startes i baggrunden. Hvis man undlader den, starter Containerne i den powershell/terminal man har åben, og stopper når man lukker for terminalen, hvilket typisk er uhensigtsmæssigt.

Docker opsamler typisk alle logs der sendes fra de enkelte Containere, og det er muligt at tilgå disse logs via Docker værktøjet på følgende måde

Start med at finde “Container ID” på den Container som loggen skal trækkes ud fra. Dette gøres via ”docker ps” kommandoen, som lister alle kørende Docker Containere

$ docker ps

Herefter kan man trække logfilen fra den kørende Container ved at skrive ”docker logs [CONTAINER ID]”, fx hvis Container ID’et er ”a2ce5fddb64e” vil man skrive

$ docker logs a2ce5fddb64e

Man kan også tilføje -f parametren, så holdes logfilen åben, og alle nye rækker i logfilen vil automatisk blive vist

$ docker logs -f a2ce5fddb64e

Docker Compose er typisk en nem og hurtig måde at starte/stoppe Docker-baserede applikationer som bare skal køre på en enkelt server. Hvis man har brug for et mere komplekst setup, hvor man skal administrere mange Docker Containers på tværs af mange forskellige servere, så findes der nogle mere avancerede orkestreringsværktøjer, som er beskrevet i næste afsnit.

4.2 Orkerstreringsværktøjer

Docker Compose afvikles på den server hvor Docker Containerne skal køre, og er ikke meget mere end en struktureret form for bat-fil, der gør det nemmere at arbejde med Docker.

Der findes flere såkaldte orkestreringsværktøjer til Docker, hvor de to mest udbredte er

- Kubernetes (Googles orkestreringsværktøj til Docker)
- Docker Swarm (Dockers eget orkestreringsværktøj)

Fælles for dem begge, er at de virker på følgende måde

1. Der installeres en agent på de servere hvor Docker Containere skal afvikles
2. Der installeres en master på én server, som alle agenter konfigureres til at forbinde til

Via ens valgte orkestreringsværktøj, forbinder man så til masteren, og foretager konfigurationen derpå. Orkestreringsværktøjer sender så de relevante Docker kommandoer ud til alle de agenter som er tilsluttet masteren.

Hermed kan man styre mange servere på én gang, og kan opsætte mere komplekse regler, fx at man ønsker at der skal deployes 3 instanser af ens applikation, og at orkestreringsværktøjet bare skal finde en server med ledige resourcer at deploye på, og så sørger den for det.

Der findes også adminsitrative brugergrænseflader til disse orkestreringsværktøjer, som bruges i stedet for at anvende kommando-linje værktøjerne direkte. Der er både officielle brugergrænseflader fra levearndørene selv, samt et hav af 3.parts brugergrænseflader, som man kan vælge mellem.

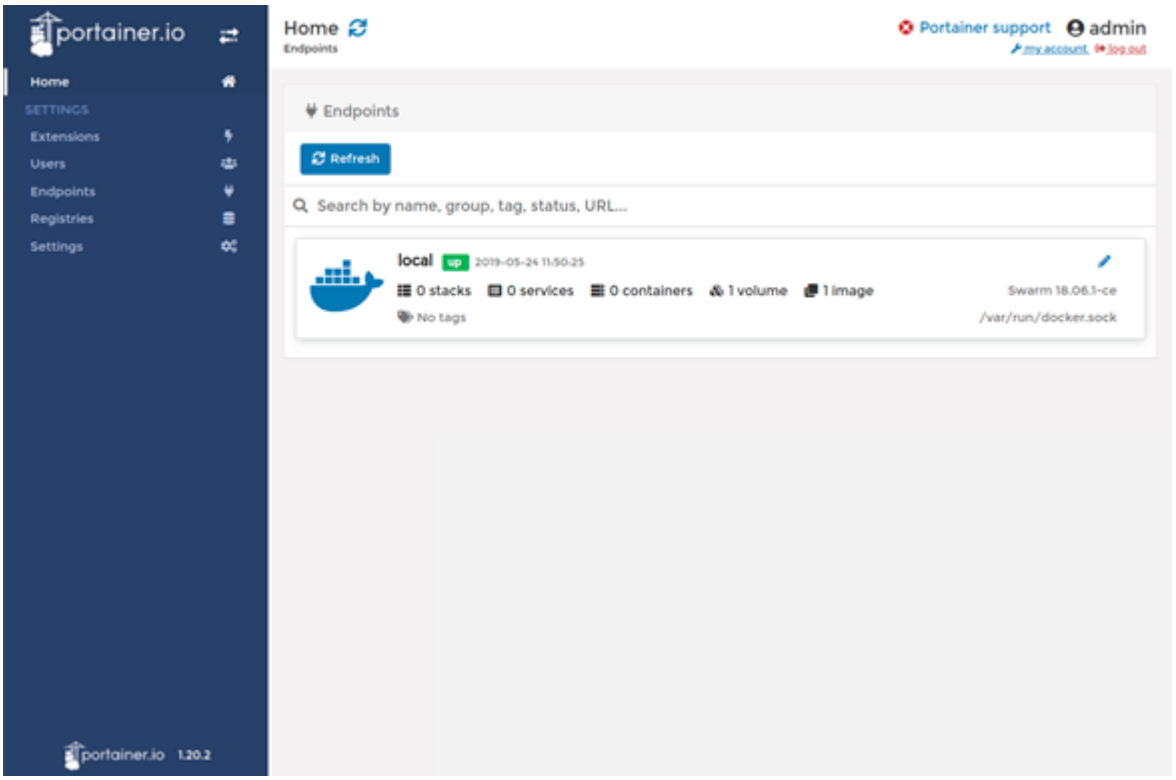
Det er lidt udenfor omfanget af denne guide at gå i dybden med orkestreringsværktøjerne, og det anbefales at starte med Docker Compose indtil ens behov vokser, da det er lettere at komme igang med, og typisk dækker ens behov i meget langt tid.

4.3 GUI værktøjer på Docker Compose

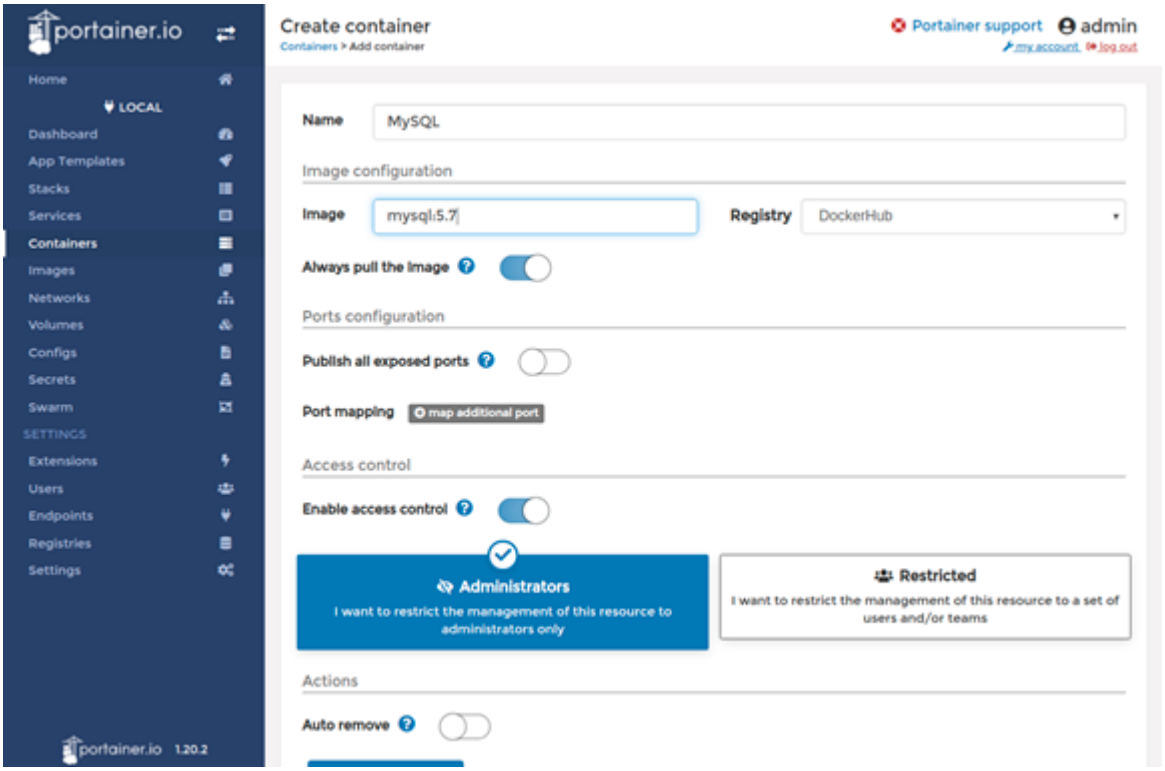
Hvis man ikke ønsker at anvende Docker Compose kommando-linje værktøjet, men gerne vil have en egentlig brugergrænseflade, findes der også en række sådanne til Docker Compose. Nedenfor er vist et sådan værktøj der hedder Portainer.

Man kan uploade en eksisterende Docker Compose fil, og så dukker de tilhørende Containers op inde i Portainer brugergrænsefladen, men man kan også oprette enkelte Containere som vist nedenfor.

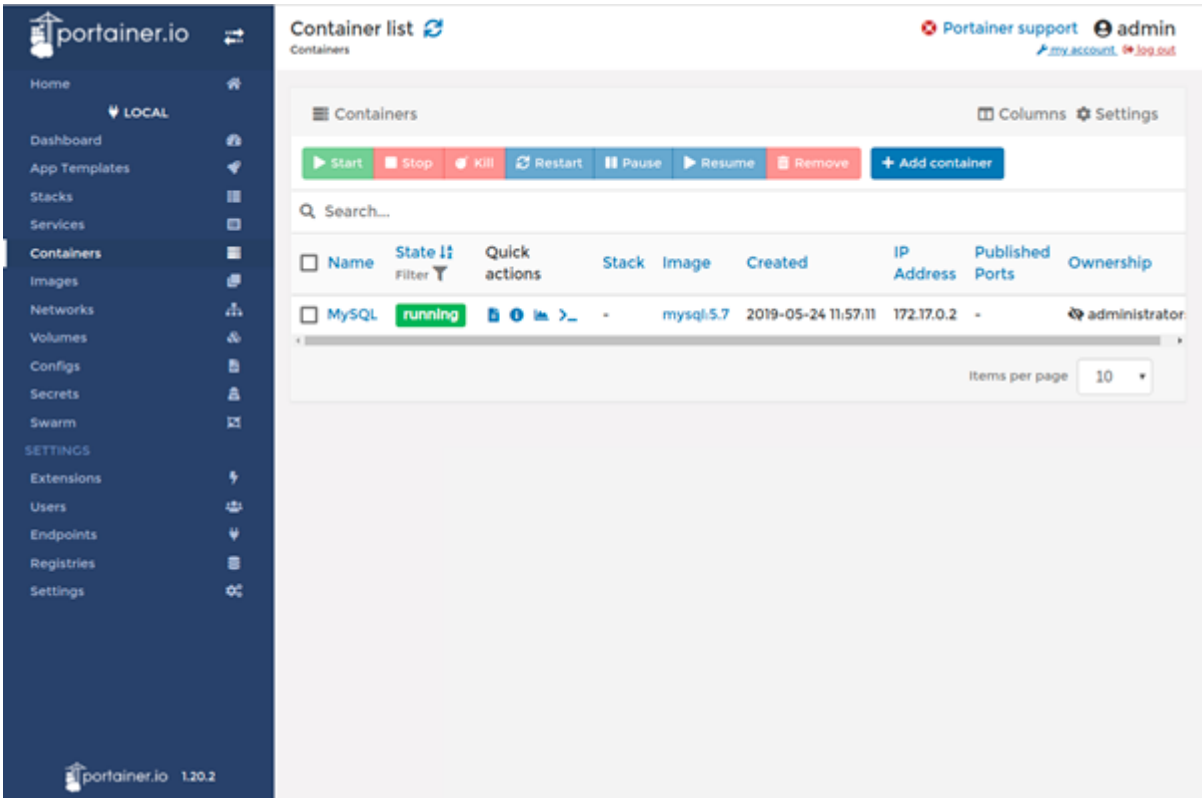
Når man tilgår Portainer via sin browser og logger på, så får man et overblik over alle de servere som man har Docker kørende på, som illustreret nedenfor (her er der kun én server)



Ved at klikke på den enkelte server, får man adgang til den enkelte servers opsætning, hvor man kan oprette nye Containers under ”Containers” menuen, som vist nedenfor.



Alle Containers der er oprettet på serveren, kan administreres inde fra Portainer dashboardet, og bl.a. startes/stoppes. Man kan også herfra tilgå logs m.m.



4.3.1 Installation af Portainer

Portainer er ikke det eneste sådanne værktøj, der kan bruges til administration af enkelte servere via Docker Compose, men den er ret feature-rig, og kan formodentligt dække langt de fleste behov.

Portainer er tilgængelig som et Docker Image, og kan startes på lige fod med andre Docker Containere. Da man ikke har Portainer kørende når man skal installere Portainer, er man nødt til at starte denne Container via kommando-linjen.

Der er en udførlig guide til dette på Portainer websitet

https://www.portainer.io/installation/