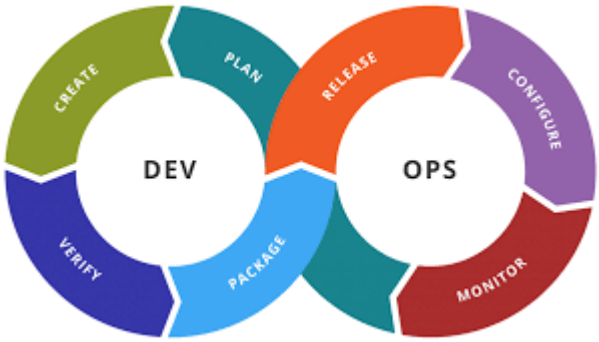

Nya tillämpningar och Buzz Words finns det gott om. Några är begrepp som DevOps, Container, Microservices etc.

Jag ska försöka bena ut vad detta är för mig själv och kanske ge er någon idè **MEN** framför allt hur jag tror det kan användas, går det att tillämpa i alla miljöer, vilka faktorer är styrande etc.

I och med en framtid mot mjukvarustyrda datacenter så finns det behov eller rättare sagt möjligheter att förändra vårt leveranssätt. Om vi riktar in oss på först DevOps, finns det ett behov av att förändra våra traditionella testmetoder? Och vad är egentligen DevOps? Vem använder det och varför ska vi använda det?

Traditionella testmetoder är inriktade mot en sekventiell leverans gärna mot olika scenarios där funktioner förväntas vara uppe och fungerande. Man får förlita sig på att det faktiskt finns en databas som kan innehålla vårt innehåll. Det finns tillräckligt med compute kraft för att driva vårt test etc.

Detta leder till en betoning på ”diskreta” uppsättningar av färdigheter, vi får en minimal effekt av varandra dvs att hårdvaran och mjukvaran har väldigt begränsat utbyte av varandra, de delger minimalt med information till var och en och har en silobaserad ”syn” på hur de ska accessas varandras cli, gui, Api.

Det finns även en begränsning hos utvecklare eftersom det finns hinder mellan utveckling, test och spridning/installationen av koden/tillämpningen.

Identifierade nackdelar med den sekventiella leveransmetoden är längre väntetid och högre risk. För att uppfylla målen i verksamheten - för att minska risken så måste vi svara snabbt.

Det finns några modeller, sajter som Facebook, Etsy, promotions, Groupon och Flickr som har kommit väldigt långt. Det här är alla siter som där snabba förändringar och förbättrad svarstid är helt avgörande för dessa företags framtid.

Netflix är ett exempel på detta som till och med går mot ett NoOps tillvägagångssätt.

Hur?

När det kommer till att testa, förändra vår IT till en DevOps hantera miljö så måste det ske gradvis och olika kompetenser som blir inblandade kommer att behöva skaffa sig nya färdigheter och anpassa sig till nya ansvarsområden. Testare behöver få kunskap i utvecklingen som lyckligtvis är lite lättare med användarvänliga verktyg men de måste också lära distributionsprocesser och verktyg.

Utvecklare och systemadministratörer, å andra sidan, måste lära sig om testprocesser, designtekniker och relaterade verktyg. Infrastrukturen är en helt avgörande faktor som **MÅSTE** fungera innan någon form av DevOps eller Continues Delivery kan tas i bruk.
Infrastrukturen måste kunna accessas via kod som i en Software Defined lösning. Vill man ha en ändå enklare miljö att hantera så bör hela stacken vara en Hyper Converged Infrastructure miljö.

Många organisationer börjar med att anställa utvecklare för att få in DevOps i sitt arbetssätt, i värsta fall så handlar det ofta om att personer helt enkelt sitter och genererar skript för att få scenariot att fungera hyfsat. Det är nästan ingen vinst i det alls. I det scenariot så byter vi installatörer mot scriptare…
Allt går ut på att automatisera och för att göra det så måste vi ha något att automatisera mot, en mottaglig infrastruktur. Vinsten är ju att leverera mer till mindre tid.
Om vi drar detta tankesätt lite längre så fungerar det då endast om vi kan använda hårdvara som kod, dvs att hårdvara = kod?

Vi bör alltså satsa på teknik som ett Software Defined Data Center tillhandahåller. Det innebär att vi kan hantera infrastruktur som mjukvara. Det underlättar hela detta arbetssätt som DevOps och Continues Delivery går ut på drastiskt.

TRANSFORMERA TILL DEVOPS

En övergång till DevOps gör också kontinuerlig integration obligatoriskt. Vi i en testdriven utveckling räknar med att misslyckas först, men dessa fel informerar nästa cykel av utveckling och testning. Detta kräver att vi måste ha en stark disciplin runt en enda källa, vår kod. Byggprocessen kan automatiseras men håll i koden!

För att alla våra element för att fungera effektivt så har vi ett behov av transparens, alla i gruppen måste veta vad som händer. Gruppen i det här fallet är alltså Development och Operations (DevOps). Processen påverkas också av det tunga beroendet av innovativ automation.

AGILE OCH DEVOPS

Agile utveckling erbjuder en väg till kontinuerlig leverans eftersom det ger utvecklare och testare tillsammans fokus på kontinuerlig utveckling, Men den sista pusselbiten existerar fortfarande. Den mur som skiljer till driftsättning

Från utveckling och testsidan är det lång tid mellan testning och driftsättning, och dessutom behöver testare inte nödvändigtvis en systemadministratörs kompetens.

DevOps erbjuder en potentiell lösning på denna integrationsutmaning vilket är att tillsammans kan utveckling, testning och infrastruktur leverera inom sina kompetensområden alla i samma lag, ett lag som kan spela alla de olika rollerna.

Detta DevOps lag behöver användarvänliga verktyg som alla gruppmedlemmar- utvecklare, testare och system administratörer kan lära och använda. Dessutom ska de ha fokus på automation och återanvändbarhet ska intensifieras.

ANVÄNDA DEVOPS METODER FÖR PRE-RELEASE OCH PRODUKTION

Hur många helpdesk (HD) biljetter har ditt IT-team fått efter programvara är släppt i produktion? Vissa organisationer "efter ny release är HD-biljetternas volym 50% av den årliga totala”. Alltså 50% av våra ”pains” är faktiskt självförvållade? Värt en tanke…

Alltför många organisationer idag strävar efter att skapa programvara med kvalitet i utveckling men kan sedan inte kan bevara kvaliteten under driftsättning och i produktionen. Detta problem är vanligt och en grundläggande orsak är att utveckling och driftteam inte fungerar tillsammans innan en programversion släpps. Man är helt enkelt inte i samma lag (vårt berömda DevOps Lag).

DEVOPS WORK

Vad händer när Dev och Ops möts? Samtalar man ens om tex övervakningsstrategier innan de släpps så Ops vet vad pågår med produktions appar.
"Det innebär övervakning av nyckelfunktioner, vissa icke-funktionella krav och andra operativa aspekter ". Det är väldigt viktig för kvalitet och att vara konkurrenskraftiga att dessa punkter hanteras.

Om utvecklingsteamet levererar information om vad som ska testas inom infrastrukturen så att verksamhet kan få en felfri driftsättning så kommer det automatiskt att bli ekonomi i det hela. Ops kan testa innan produktionen. Ops bör informeras om utvecklingsprovning, förväntat resultat etc. Ops ska tillhanda hålla en specifikation om vad den väntade målplattformen ger för utrymme till Dev.

Tydliga ansvarsområden bör ställas in. Till exempel bör utvecklingsteam inte försöka ändra scheman för produktionssättning eftersom det kan vara är en uppgift för databasadministratörer eller andra roller att tillhandahålla etc.

När ett program/funktion är i produktion så vill de flesta organisationer flytta ansvaret för programmet från utvecklingsteam till verksamheten. Då måste Dev-Ops ha gjort sin läxa.
daglig drift ska kunna hanteras av en driftpartner, lokal som extern.

Automatiserad övervakning av produktion och kvalitetskontroll ger förmåner som att kunna bli av med manuella ingrepp vilket ger snabbar ingripande och minskad arbets- och stilleståndskostnader.
Automatisering av nya system och förbättringar av befintliga system kan komma i produktion med en mindre investering i infrastruktur.
Automatisering är en framgångsfaktor!

Container och Microservices

Det snabbast utvecklade konceptet i all Cloud Computing (min tolkning är mjukvarustyrd Computing ☺) idag är container teknik.

Det verkar som bara positiva saker man kan säga om molnteknik har sagts om Container. Många företag undrar varför något annat ens övervägs. Det är sant att Container tekniken troligen är underutnyttjade i olika företags plattformar idag. Men det finns också en stor risk att hoppa in i container tänket som kan leda dina planer till en mörk plats.

I traditionella virtuell maskin (VM) moln (on Prem – off Prem), delar en hypervisor en server/host i virtuella maskiner som i nästan alltid fungerar precis som separata fysiska servrar skulle. Varje VM driver sina egna applikationer, middleware och även operativsystemet. Och hypervisorn förmedlar delat minne, I / O, nätverksanslutningar och andra fysiska systemelement.

Detta tillvägagångssätt drar fördel av det faktum att moderna servrar har kraftfulla flerkärniga processorer som vanligtvis är underutnyttjade i traditionella applikationer. Dela hårdvara genom virtualisering och ”cloud computing” sänker hårdvarans kapitalkostnader. Eftersom det finns bara en fysisk server oavsett hur

många virtuella maskiner är på det, är hårdvaruledet även billigare. Varje VM är i huvudsak en fristående server så VM-teknik används ofta i publika moln.

Problemet med en VM strategi är att företag som bygger privata moln behöver inte flera hyresgästerstöd/tenants (multi tendens) - de är den enda hyresgästen. Dubblering av ett operativsystem för varje applikation innebär slöseri med minne och mer komplicerade att bygga och underhålla maskinbilder. Containrar är en typ av operativsystemnivå för virtualisering, där en enda OS körs i flera containrar som delar OS tjänster och hårdvara.

Fördelar med att använda container teknik

Någon utvärdering av containrar kontra VM för molnmiljöer bör börja med den första grundläggande fördelen: Mjukvarucontiner är i sig mer minnes effektiv. Det innebär att de kommer att vara mest användbar där det finns en hel del applikationer att köra och en hel del potentiella dubblering av ett OS finns om VM används.

Utöver frågan om förmåner vid minneskapaciteten är nästa fråga att göra ett beslut av containerbehovet vid ansökan/etablering av ny kund/hyresgäst.
Vissa företag har applikationer som kräver höga nivåer av åtkomstkontroll och datasäkerhet till den punkt där de skapar en egen virtuell hyresgäst.

Container teknik som delar ett OS och bygger färre hårda skiljeväggar mellan applikationer, är svårare att använda i dessa höga säkerhetstillämpningar.

Det är bäst att kolla med din interna revision eller organisation för att se till att du kan möta de krav som finns med en ”container” leverans.

Networking

Nästa sak att titta på är komplexiteten av nätverksanslutning som program behöver.
De flesta användarna är medvetna om det faktum att alla moln och virtualiseringsteknik inklusive den populära VM Openstack modell har endast mycket grundläggande nätverksfunktioner. Container verktyg med målet att förenkla distribution är oftast ännu mer grundläggande. Docker, den mest populära containerplattformen/ramverk har betydande begränsningar i sin nätverksmodell.

Komplexa nätverk i molnet (Off Prem – On Prem) innebär vanligtvis komplicerade modeller, failover strategier, storskalig användning av dynamiska

applikationskomponenter som Micro services eller en blandad grupp av användare var och en med sin egen uppsättning av värd applikation/er och verktyg.

Alla dessa frågor är en signal för att leta efter ett mjukvarudefinierat nätverks (SDN, Software Defined Network) alternativ som passar dina behov och sedan införliva antingen virtuella eller container i det alternativ du har valt.
SDN är ett kapitel för sig men det finns dag NVGRE och VXlan som mig veterligen är de mest utbredda inom Datacenter hantering. NVGRE var Microsofts variant i och med server 2012 och WaP installationer. I och med Azure Stack så har Microsoft skiftat till VXlan som är en standard som bland annat Cisco och VMware tagit fram. Lenovo är väl förbered på den tekniken och har certifierad hårdvara färdig för implementation.

Den sista frågan är korsningen av dina ”moln” mål och din personals kompetens. Programvarucontainer är lättare att använda än virtuella, särskilt om du har tid eller får professionell hjälp för att inrätta nätverket i din hosting miljö och kan tillämpa DevOps verktyg och tänk.

”Du planerar att köra avancerade applikationer i ett privat eller hybridmoln och du behöver inte kompetens och personal för virtualisering och nätverksdrift. Kommer container stödet ger en enklare väg till framgång?”.

Containrar är för närvarande inte lika mångsidig som VM. Säg att din personal redan förstår virtualisering, särskilt om du har ett virtualiserat datacenter från en leverantör som stöder privat moln driftsättning. Då kan du vara i bättre händer att vistas med virtuella maskiner om du planerar omfattande användning av molnet, särskilt om du planerar dynamiska tjänster, microservices eller program som involverar några av funktionerna inom webbtjänst från Amazons eller Microsofts offentliga moln.

Ingenting står stilla i molnet, särskilt i genomförandet med container teknik. Docker dominerar container tänket i dag, men det finns andra nya alternativ, inklusive Virtuozzo / LXC, Lätt virtuella miljöer, RKT (uttalas raket) och Red Hats OCID.
Var och en av dessa alternativ ger en annan uppsättning funktioner som hanterar olika fördelar och begränsningar annorlunda från Docker.

Använda container och virtuella maskiner tillsammans

Även VM-användare bör hålla ett öga på dessa trender. VMs har utformats för en beräknings värld där användarna förlitat sig på flera operativsystem och där operativsystemet inte erbjuder fullständig säkerhet genom att dela applikationer och resurser. Vi är på väg mot en tid, kanske, när container mjukvara och virtuella

maskiner konvergerar och när en container liknande strategi kommer att vara den rätta vägen för alla. Ordet konvergens kommer inom de flesta nyheterna inom IT ☺

Det är nästan omöjligt att gå till någon teknik konferens och inte höra orden Docker container åtminstone en gång. Containrar var en gammal nisch teknik tills Docker dök upp med nya användningsområden och förändrat spelet.

Det hjälper till att inleda en ny era av DevOps tankesätt/arbetssätt genom att utvecklare kan snabbt paketera och distribuera applikationer mot infrastruktur med hjälp av Ops,
nyckelordet är samarbete.

Dockers projekt med öppen källkod är inte den enda container teknik, men det bidrar till att katalysera den totala container rörelsen som olika leverantörer och slutanvändarorganisationer vi nå. Man kan säga att Docker är ett ”framework”.

Containrar erbjuder många möjligheter för nätverkstekniker att verkligen bidra till att förenkla komplexa nätverksutmaningar.

Container nätverksfunktionalitet är mycket enklare än vad som finns i mer mogna virtuella miljöer. Även om det kan vara till hjälp i arbetet på container i större skala, innebär det att mer arbete måste göras för att lagret ska ha någon form av finess i miljön.

Går man ned till bare-metal och fysiska miljöer är containers nätverk liknande VM-nätverk - med så klart avvikelser. Container kommer och går snabbare än virtuella maskiner, så nätverket måste utformas kring denna verklighet. För det andra, människor kommer att köra många fler containrar än de skulle köra VMár, så mängden adressutrymme som en container miljö kan förbruka kommer sannolikt att vara större.

Med en VM är ett operativsystems typiska användningsområde att användas som en del av VMens programbild, medan container modellen bygger på värdsystemet för operativsystemfunktioner.

Genom att inte behöva sitt eget operativsystem som del av ett ”virtuell bild” kan en containers programbild vara mindre. Vi spar alltså storlek på ”paketet” vilket innebär att en snabbare transport kan ske, framför allt mellan On-prem och Off-prem siter.

Samtidigt kan motsvarande densiteten hos en containers utbyggnaden vara högre än en miljö baserad på virtuella maskiner.
Det är värt att notera att containrar och VM`s är inte ömsesidigt exklusiva tekniker

heller. I själva verket är det ofta en rekommenderad bästa praxis att containers körs inne i en VM.

I containers är ett IP-gränssnitt det enda objektet av intresse för utvecklare. Men för att lägga säkerhet eller fler hyresrätter/kunder till en container i synnerhet behöver container s mycket mer än den nativa nätverksfunktionaliteten. Det är ett område där mer arbete behövs, vilket har öppnat upp en möjlighet för tredje part att fylla tomrummet med open source-projekt för container nätverk, såsom Weaveworks 'Weave Net och CoreOS "flanell.

MicroServices = Mikro tjänster

En term som ofta kommer upp när man diskuterar container är mikro tjänster. Den grundläggande idén bakom mikro tjänster är att istället för att ha en monolitisk ansökan stack så bryter man ut varje specifik tjänst i en ansökan inom hela leveranskedjan till enskilda delar.

Mikrotjänster, microservices, innebär som namnet antyder att applikationer och webbtjänster byggs upp av små delar.

Mikrotjänster körs var och en under sina egna processer och kommunicerar med lätta mekanismer, oftast via protokollet http med rest, och helst oberoende av varandra.

Varje mikrotjänst kan ha en egen unika arkitektur som är lämpad för ändamålet.

Eftersom man inte behöver ha samma kodbas för alla olika delar av en applikation blir det lättare att leverera ny funktionalitet snabbt och kontinuerligt. Man separerar/bryter ned leveranskedjan till olika delar.

Uppdateringar, till exempel för att rätta fel, blir enklare att hantera, eftersom det är mindre delar av en applikation som behöver uppdateras.

Skalbarhet blir enklare att ordna eftersom det går att skala de mikrotjänster som blir belastade i stället för hela applikationen.

Olika mikrotjänster kan vara av olika versioner, de kan alltså utvecklas oberoende av varandra och man slipper i hög grad synka de olika mikrotjänsterna mot varandra.

Sist men inte minst, kan den sammantagna arkitekturen för alla ingående mikrotjänster i en applikation bli mindre komplex än arkitekturen för en traditionell applikation i vilken alla delar hänger tätt ihop. Man slipper helt enkelt kod för att samordna funktionalitet i de enskilda mikrotjänsterna.

Alla de här fördelarna bottnar i ett enda antagande: ju mindre komplexitet, desto bättre på alla sätt.

Om man kortfattat sammanfattar hela denna artikel så handlar det hela om att hantera vår vardagliga scen med kod.

Hårdvaran måste bli accessbar via kod, all form av DevOps, automatisering, Continues Delivery bygger på kodhantering.

Vi kan använda ny teknik som Microservices för att underlätta, vi kan distribuera applikationer via container hantering för snabbhet och för att lätta på vår kommunikations hantering.

Vi vill få en modern leverans av verksamheten och det möjliggör vi genom att lyfta våra datacenter till att hanteras som ett Software Defined Data Center!

Hårdvara = Mjukvara ☺

Författare av dokumentet

MATTIAS KILTORP

Arbetar idag som Head Of Operations på DataCom. En Privat, publik och Hybrid leverantör av tjänster med en hög säkerhetsmognad och alltid med kunden i första rummet.
Han har mer än 20 års erfarenhet inom IT branchen och har djup teknisk och konceptuell kompetens inom hela IT leveransen och tjänsteutveckling. Han har erfarenhet av större uppdrag, personal-, utbildnings-, ledarskap- och förändringsarbete mycket genom tidigare uppdrag och genom att ha drivit IT konsultbolag. Konsulterfarenheter inom både tillverknings-, och tjänstesektorn i näringslivet såväl som statlig, kommunal verksamhet som privat. Han har haft uppdrag som driftchefs på MSP bolag, uppdragsansvar inom Telecom, drivit ett flertal egna konsultfirmor, hög kompetens inom Azure (Cloud-modell), Active Directory, Enterprise Mobility, Desktop virtualisering som Citrix, RDS. Stor erfarenhet inom Software Defined Data Center lösningar som Azure Stack, Nutanix, DataCore, Simplivity etc. Han brinner starkt för transformation till mjukvarubaserade lösningar genom hela kedjan av IT leverans från användaren till datacenter.

Mattias har skrivit flertalet artiklar inom ovanstående ämnen:

- **Multi-Cloud: It's all about choice`s**
- **Azure Cloud, DevOps, Services Introduction**
- **Cloud & Azure Reflections**

- **DevOps - Container – Cloud Application “Hardware=Software”**
- **Data Center + Evolution = Deliver more with less**
- **Azure Stack**
- **Converged & Hyper Converged Infrastructure**
- **NUTANIX**