HowTo för Proxmox VE-kluster

Jack-Benny Persson

2014-07-30

Versionshistorik

Utgåva	Datum	Vem	Ändring
1.3.1	2014-07-30	Jack-Benny Persson	Ändrat teckensnitt för bättre läslighet
1.3	2014-06-01	Jack-Benny Persson	Lagt till versionshistorik
1.2	2014-05-30	Jack-Benny Persson	Anpassad för publicering som HowTo
1.1	2014-04-10	Jack-Benny Persson	Första utgåvan som HowTo
1.0	2014-04-09	Jack-Benny Persson	Första utgåvan som labb

Innehåll

1 Syfte

Att sätta upp ett kluster med Proxmox VE. Först kommer ett vanligt kluster att sättas upp, d.v.s. ett kluster där man enkelt kan migrera alla sin VM's mellan noderna i klustret. Senare kommer även ett HA-kluster att sättas upp.

2 Genomförande

Tre noder kommer att sätta upp. Varje nod kommer att ha två NIC, ett för kommunikation mellan noderna och för åtkomst av web-GUI:t samt ett för kommunikation utåt nätet (WAN/LAN etc). Dessa tre noder kommer sedan att klustras.

2.1 Kort introduktion till Proxmox

Proxmox är en Linux-distribution som innehåller alla de delar som behövs för att köra OpenVZ och KVM på servrar. Dessutom har Proxmox ett lättanvänt webbinterface för att kontrollera och övervaka servern. Här i denna text kommer container att menas med OpenVZ container och VM med en KVM virtuell maskin. OpenVZ är en form av virtualisering som kör samma kärna som host-systemet. Det är just därför man kallar det för container, då det är mer en container än en virtualisering. Proxmox har även stöd för att skapa kluster, båda HA-kluster och "vanligt" kluster för att kunna flytta data och VM's mellan sig. Klusterdelen av Proxmox använder sig av Corosync/CMan och RGManager.

2.2 Installation av noderna

Börja med att hämta hem den senaste versionen av Proxmox (i skrivande stund 3.2) från http://www.proxmox.com/downloads. Boota upp noden med ISO:n och följ den guidade installationen. Jag använde själv DHCP vid nätverkskonfigurationen då detta nät är mitt interna nät, d.v.s. det nät som ska användas för klustret och åtkomst till webbinterfacet m.m. Under installationen kommer du att få välja root-lösenord och ett hostnamn och domännamn. Välj som hostnamn för de tre maskinerna, node1, node2 och node3 för enkelhetens skull. När installationen är klar klicka på Reboot. Gör detta för alla tre noderna. När alla noderna är installerade logga in på node1 via webbinterfacet. URL:en är https://IP-adress:8006/ där IP-adress är den IP-adress som syns i konsolen för noden. Notera att det under "Datacenter" i menyn till vänster just nu bara finns en enda nod, noden du precis loggat in till.

2.3 Uppsättning av klustret

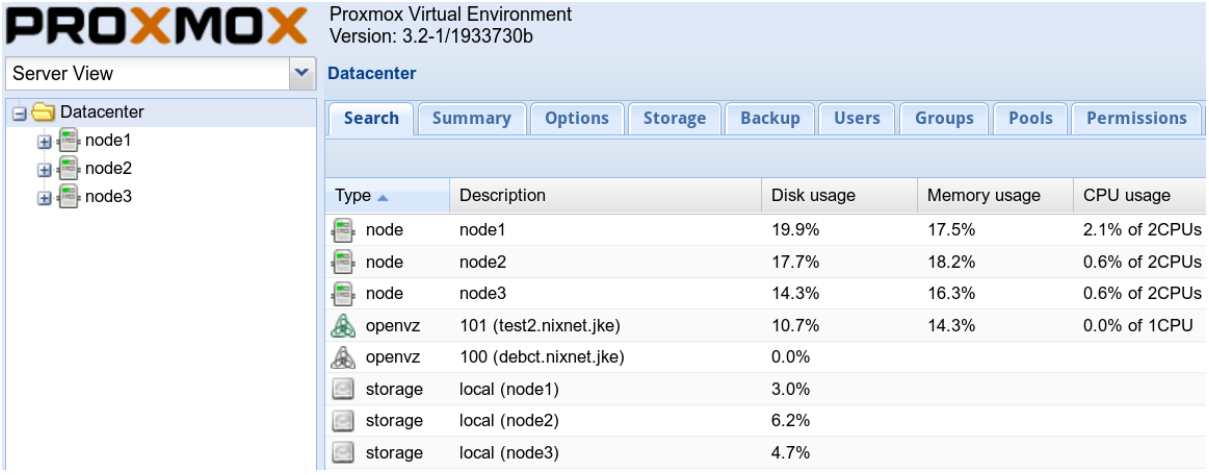
På samma nod som du precis loggade in på via webbinterfacet ska du nu logga in på via SSH som root. När du är inloggad ska du skapa själva klustret. Detta gör du genom att ange kommandot **pvecm create KLUSTERNAMN**. Observera att du inte kan ändra klusternamnet i efterhand så välj ett bra namn redan vid start. När denna processen är klar är det dags att logga in på de två andra noderna (node2 och node3) för att lägga till dessa till klustret. Ange följande kommando på både node2 och node3. **pvecm add NODE1-IP**. NODE1-IP ska vara IP-adressen till node1, d.v.s. den nod som du precis skapade klustret på i förra steget. Här blir du tillfrågad efter root-lösenordet till node1, skriv in det! Det som sker nu är att det skapas SSH-nycklar mellan noderna så att de kan skicka data mellan varandra (t.ex. VM's) och att noden görs till en medlem av klustret. Proxmox använder CMAN och CoroSync för klustret men allt detta sker helt transparent för användaren, så detta är inget man behöver konfigurera själv.

När du utfört ovanstående steg på de båda noderna (2 & 3) ska de dyka upp i webbinterfacet under "Datacenter". Klicka runt bland de olika noderna i klustret och undersök CPU, minne och hårddiskar bland noderna. Redan nu är klustret färdigt att börja användas! Enkelt inte sant? Du kan även se lite mer

detaljerad information om klustret med kommandot **pvecm status**. Här ser man lite mer information som bör se ut enligt nedan. Här ser man bland annat att mitt kluster heter JB och att består av 3 noder.

```
Version: 6.2.0
Config Version: 3
Cluster Name: JB
Cluster Id: 214
Cluster Member: Yes
Cluster Generation: 168
Membership state: Cluster-Member
Nodes: 3
Expected votes: 3
Total votes: 3
Node votes: 1
Quorum: 2
Active subsystems: 1
Flags:
Ports Bound: 0
Node name: node1
Node ID: 1
Multicast addresses: 239.192.0.214
Node addresses: 192.168.122.12
```

Nedan visas också ett screenshot från Proxmox webbinterface med 3 st noder. I screenshotet ser man också en överblick över hela klustret med både noder, lagringsutrymmet, CPU-belastning för noderna samt eventuella virtuella maskiner (här 2 st OpenVZ containers).



2.4 Skapa VM's & Containers

Innan vi skapar några VM's eller OpenVZ containers måste vi sätta upp en ny brygga (såvida vi inte vill ha vår VM på samma nät som Proxmox). Det finns några viktiga saker vi måste tänka på här och ta hänsyn till som gås igenom nedan.

2.4.1 Nätverk och bryggor

Det sätt jag själv föredrar och som passar mitt nätverk bäst är att skapa en brygga mot det externa nätet och låta bryggan vara utan IP-inställningar. På så sätt "sparar" jag ett IP-nummer och får dessutom säkerhet på köpet. Eftersom bryggan inte har några IP-inställningar går det inte att på något sätt att

nå själva Proxmox-noden från utsidan, bara de virtuella maskinerna inuti Proxmox. Nackdelen med detta är ifall man vill använda *venet* i OpenVZ. Venet är ett sätt där man sätter en fast IP-adress på en OpenVZ container redan under skapandet av containern. Denna IP-adress blir då fast i containern och går inte att ändra inuti containern. Detta är en vanligt sätt att arbeta för VPS-hostingföretag. Eftersom man bara sätter själva IP-adressen och resten av inställningarna hämtas från bryggan så fungerar inte venet utan IP-inställningar på bryggan. Venet är således ett sätt att routa trafik genom bryggan. Det andra sättet, det som jag själv använder mig av, kallat *veth* är istället ett sätt för att brygga mot bryggan. På detta sätt kan de virtuella maskinerna använda DHCP för att få en IP-adress från det externa nätet t.ex. Observera att ifall du testar Proxmox inuti en KVM (nestad virtualisering) så fungerar det inte med veth mot en brygga som i KVM-hosten är bryggat mot ett fysiskt NIC. Jag slet mig själv i håret under många timmar innan jag själv förstod detta... Däremot fungerar det så klart utmärkt på en fysisk maskin (eller ett isolerat nät inuti en KVM-host)!

Vi går nu vidare till att faktiskt skapa vår brygga. Börja med att markera den nod du vill skapa din brygga på (i det vänstra fältet under "Datacenter"). Klicka sedan på fliken *Network* i högra fältet. Klicka sedan på *Create* och sedan på *Linux Bridge*. Välj sedan det nätverkskort som du vill att bryggan ska vara kopplat till (det externa nätet t.ex.). Här väljer du själv om du vill ange IP-inställningar eller inte (se ovan). När du sedan sparat dina inställningar för den nya bryggan måste du starta om Proxmox noden för att bryggan ska bli aktiv. Gör så! När du startat om noden och går tillbaks till nätverksinställningarna för noden bör de ut så ungefär som i bilden nedan.

Gör nu om samma procedur för alla noderna. Det är viktigt att alla noderna har samma inställningar! D.v.s. om du angett IP-inställningar för en nod måste du ange IP-inställningar för alla noderna (så klart med en egen unik IP-adress). Har du valt att inte ange några IP-inställningar för noden ska du inte göra det för de andra noderna heller. Detta för att man ska kunna flytta VM's och containrar fritt mellan noderna utan downtime.

2.4.2 Skapa din första OpenVZ container

Nu ska vi testa att skapa vår första OpenVZ container i vårt Proxmox kluster. Först och främst innan vi kan skapa någon container så måste vi tanka hem en template att skapa contianern utav. Klicka på plustecknet framför den nod där du vill skapa din OpenVZ container på i menyn till vänster direkt under Proxmox-loggan. Markera sedan *local*, alltså den disk som finns i noden. Klicka sedan på fliken *Content* och därefter på knappen *Templates*. Markera sedan den template du vill ladda ner till noden, t.ex. Debian-7.0-standard och klicka därefter på *Download*. Nu tankas automatiskt den valda templaten ner till noden. När detta är klart har det blivit dags att skapa din första container. Klicka nu på knappen *Create CT* längst upp i det högra hörnet. Välj den nod där du laddade hem templaten till i dialogrutan. Fyll sedan i hostname och lösenord för root-användaren och klicka på *Next*. I nästa steg ska du välja den template du precis laddade ner. Klicka på *Next* igen och välj hur mycket resurser du vill att din container ska ha och klicka återigen på *Next*. Här i nästa steg har det blivit dags att välja hur nätverket ska fungera på din container. Antingen kan du välja *Routed Mode; venet* och fylla i en IP-adress (detta kräver dock att man fyllt i IP-inställningarna för bryggan) eller så väljer du *Bridge Mode* som är en brygga mot

bryggan. Detta är det sätt jag själv föredrar, då ställer man istället in alla IP-inställningar inuti den virtuella maskinen. I nästa steg ställer du in dina DNS-inställningar för den virtuella maskinen. I det allra sista steget *Confirm* är det dags att godkänna alla inställningarna och genomföra dem. Om allt ser rätt ut klicka på *Finish*. Nu skapas din första container. Nu kan du starta upp den genom att högerklicka på den och välja *Start*. Högerklicka på den igen (i vänstermenyn) och välj *Console* så får du upp en konsol. Observera dock att för att detta ska fungera krävs att du har Java installerat på din dator. Väl inne i din maskin får du börja med att sätta upp ditt nätverk ifall du valt *Bridge Mode*. Testa även att SSH:a in till din virtuella maskin från "utsidan". Fungerar det så är allt frid och fröjd!

2.5 Testa klustrets funktioner

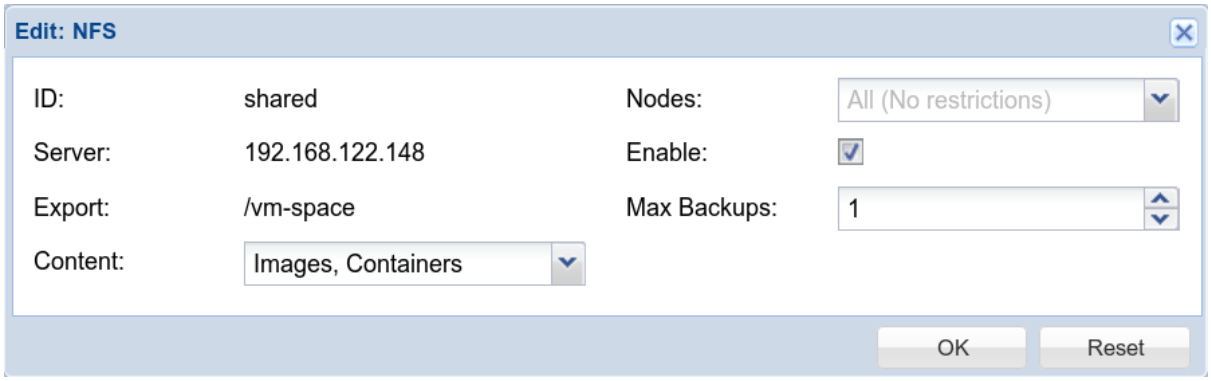
Nu när vi har en fungerande container kan vi testa lite av de funktioner som vårt Proxmox-kluster har att erbjuda. Från en dator på "utsidan" starta en konsol och pinga din virtuella maskin och låt den fortsätta pinga. Gå sedan till Proxmox's webbinterface och högerklicka på din container och välj *Migrate*. Välj en nod dit du vill flytta din container. Nu kommer containern att flyttas medans den är online. Om allt är OK så ska du endast få ett litet avbrott i ping sedan ska den fortsätta igen. Det kallas att migrera den virtuella miljön live. Avbrottet i ping bör vara maximalt ett par sekunder, sedan ska maskinen vara uppe igen på den andra noden med fullt fungerande nätverk! Hela containern har således nu flyttats mellan två noder med bibehållen funktionalitet, även under själva flytten. Det är detta som är den stora fördelen med klustret i Proxmox, när resurserna på en nod börjar tryta så kan man flytta över containerns och VM's till andra noder. Likaså om man behöver ta ner en nod för att t.ex. installera ny hårdvara så kan man under tiden flytta över alla VM och containrar till andra noder så att de fortfarande är uppe och fungerande.

2.6 Sätta upp VM's för HA

Att skapa ett HA-kluster för Proxmox är relativt enkelt. I själva verket har vi redan ett HA-kluster bortsett från att vi inte aktiverat resurshanteraren än och inte har satt några VM's under HA-kontroll ännu. Det enda vi behöver göra innan vi kan skapa virtuella maskiner för HA är att skapa en NFS-server som vi monterar i Proxmox-klustret. Detta för att alla noderna i klustret måste ju ha tillgång till den virtuella maskinen ifall den nod som för närvarande kör vår VM eller container skulle gå ner. Annars kan ju inte de andra noderna ta över den virtuella maskinen. Sätt nu upp en NFS-server och dela ut en katalog på den som går att montera från noderna. Var observant att du sätter NFS-servern i rätt nätverk, d.v.s. ditt interna nät. Ännu bättre hade ju självklart varit ett separat nät för NFS-servern. Ska Proxmox-klustret vara live rekommenderar jag att du sätter NFS-servern till noderna på ett helt eget nät och då gärna ett väldigt snabbt nät så att detta inte blir en flaskhals. Tänk på att noderna kommer att köra alla VM's som är HA-aktiverade från NFS-servern, även när den inte gör någon failover. Sätt upp din NFS-utdelning så att noderna kan skriva till den, så att de kan skapa nya VM's m.m. Min /etc/exports på min NFS-server ser ut enligt nedan.

```
/vm-space       *(rw,sync,no_root_squash)
```

När NFS-servern är upp och fungerande är det dags att montera utdelningen i alla våra Proxmox-noder. När allt är ifyllt ser min konfigurationsruta för NFS-utdelning ut enligt bilden här nedan.



Var observant så att du inte INTE väljer en nod här, då NFS-utdelningen ska vara tillgänglig för alla noderna. Nu ska NFS-utdelningen (i det här fallet kallad "shared") dyka upp under samtliga noder i klustret. Klicka på plustecknet framför alla noderna i menyn till vänster, där ser du nu din NFS-utdelning under alla noderna.

Nu kan vi gå vidare till att skapa en VM i "HA-läge". Skapa först en ny OpenVZ container precis som vanligt med undantaget att istället för att lägga den på *local* på en av noderna ska du lägga den på din NFS-utdelning. När allt är klart kan du testa att starta upp containern och ställa in IP-inställningar m.m. När du gjort alla inställningar i din container måste du stänga ner din container igen.

Nu kommer några viktiga moment för att få HA att fungera. På alla våra noder i klustret måste vi nu redigera en fil. Logga in till noderna via SSH och redigera filen /etc/default/redhat-cluster-pve. I denna fil måste vi avkommentera raden *FENCE_JOIN="yes"*. Spara filen och starta om cman med **service cman restart**. Därefter ska du köra kommandot **fence_tool join**. Efter detta är det dags att starta RGManager (Resource Group Manager) med **service rgmanager start**. Om *rgmanager* startar utan felmeddelanden är allt ok! Gör dessa steg på alla noderna så att alla noder kör RGManager. En liten förklaring kan vara på sin plats här så att vi vet vad allting gör. Det vi gjorde först med att avkommentera raden i /etc/default/redhat-cluster-pve var att vi gjorde noden en medlem i fencing-gruppen. Vi har dock inte satt upp någon fencing ännu utan bara satt upp så att noderna är medlem i fencingen. Detta krävs för att kunna starta RGManager. Sedan startar vi RGManager. RGmanager är det som tar hand om själva de virtuella miljöerna. De virtuella miljöerna är i själva verket resurserna i vårt kluster. Ett kluster består av en rad resurser som körs på varje nod, och dessa resurser i detta fall är våra virtuella maskiner. När du utfört dessa stegen på alla noderna så logga in på alla noderna och kontrollera så att tjänsten *rgmanager* körs. Detta ser du genom att markera noden och sedan klicka på fliken *services*. Kontrollera i listan så att RGManager står som *running*.

Nu kommer nästa viktiga steg, nämligen att lägga vår OpenVZ container eller VM under HA-kontroll. Gör detta genom att först markera *Datacenter* i menyn till vänster och sedan klicka på fliken *HA* i rutan till höger. Klicka därefter på *Add* och sedan på *HA managed VM/CT*. Skriv in ID-numret på din OpenVZ container eller VM här som du vill ha under HA-kontroll och klicka på *Create*. Nu ser du ändringarna som en diff i den nedre rutan. Ändringarna är alltså inte genomförda än. För att genomföra ändringarna måste du klicka på *Activate*. Innan du gör detta måste du se till att din container eller VM är avstängd. När du gjort detta är din container/VM nu i HA-läget (High Avalibility). Detta innebär att om din VM/Container körs på en nod som går ner, så kommer klustret automatiskt att föra över VM/containern till en annan nod så att den alltid är uppe. Och precis som när vi testade att migrera en container så kommer nertiden bli väldigt kort, maximalt ett par sekunder. På nedanstående bild kan man se hur min klusterkonfiguration ser ut efter att jag lagt in min container (med ID-nummer 103) till HA-kontroll.

Tag	Attributes
cluster	name="JB" config_version="4"
cman	keyfile="/var/lib/pve-cluster/corosync.authkey"
clusternodes	
clusternode	nodeid="1" name="node1" votes="1"
clusternode	nodeid="2" name="node2" votes="1"
clusternode	nodeid="3" name="node3" votes="1"
rm	
pvevm	autostart="1" vmid="103"

Man kan även dubbelkolla så att sin VM eller container hanteras av HA genom att markera containern i menyn till vänster och klicka på fliken *Summary*. Där ska det finnas en rad som säger *Managed by HA* och denna ska då vara satt till *Yes*, precis som i bilden här nedan.

Status	
Name	hadeb.nonet.jke
Status	running
CPU usage	0.6% of 1CPU
Memory usage	Total: 512MB Used: 18MB
VSwap usage	Total: 512MB Used: 0
Uptime	01:57:13
Managed by HA	Yes

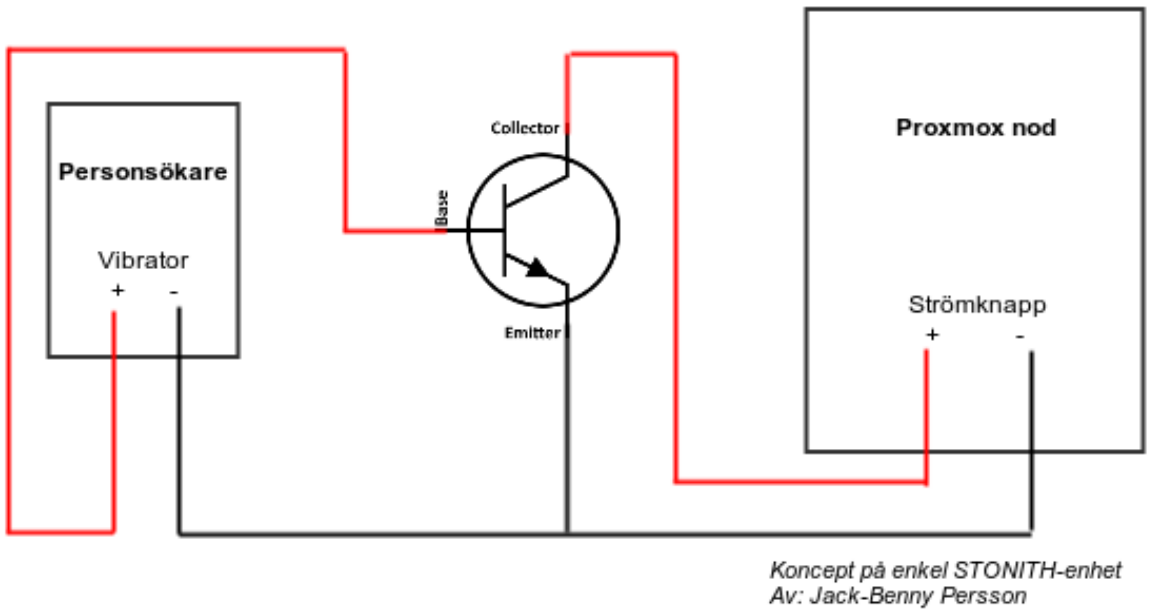
2.7 Testa HA

Nu kan vi testa vårt HA-kluster genom att stänga ner den nod som för tillfället kör vår container/VM. Om det är så att containern/VM för tillfället körs på den nod du är inloggad på via webbinterfacet så logga in via en annan nod. Det spelar ingen roll från vilken nod du loggar in till webbinterfacet, samma information visas överallt. Stäng nu ner noden som kör din container/VM och se hur den automatiskt hoppar över till en annan nod. Helt magiskt! Du har nu ett fullt fungerande Proxmox kluster med HA! Det enda enda vi inte har ännu som vi egentligen måste ha är någon form av fencing för att helt döda den nod som strular så att den inte börjar skriva till en container eller VM som precis flyttats över till en annan nod. Om så sker så kommer troligen containern eller den virtuella maskinen att förstöras.

2.8 Kort om fencing

Fencing används som sagt för att döda en nod som strular för att den inte ska börja skriva till en VM när denna redan har flyttats över till en annan nod. Det finns en mängd olika sätt att fenca en Proxmox-nod på. För mer information om hur fencing sätts upp i Proxmox se deras egna Wiki på URL

https://pve.proxmox.com/wiki/Fencing. För att få en lista på fencing-skript som finns implementerade i Proxmox så kör kommandot **find /usr/sbin -name fence_***. STONITH är ett annat namn på fencing och står för Shoot The Other Node In The Head. STONITH brukar menas med just att man dödar den andra noden genom att helt enkelt stänga av strömmen till den. En enkel fencing-enhet kan byggas med ganska enkla medel. Nedan visas en konceptuell hemmabyggd fencing-enhet som består av en personsökare och en NPN-transistor. Personsökarens vibrator brukar ge tillräckligt hög spänning för att kunna sluta transistorn. Transistorn kopplas sedan över strömbrytarpinnarna på noden. När strömbrytarknappen/pinnarna varit slutna i ca 10 sekunder stängs datorn av.



Koncept på enkel STONITH-enhet
Av: Jack-Benny Persson

3 Sammanfattning

Att köra ett kluster med Proxmox VE-maskiner har många fördelar. Framförallt för att ha möjligheten att ta ner noder för uppgradering och reparation och då under tiden migrera alla virtuella maskiner till en annan nod. På så sätt kan man bibehålla maximal uptime samtidigt som man har "råd" att ta ner noder för underhåll. HA-kluster har sina uppenbara fördelar. Skulle en nod gå ner så tar en annan vid automatiskt och kunderna märker inget av att noden gick ner. Den korta downtime som blir på ett par sekunder när den virtuella maskinen förs över till en annan nod kan nog de flesta leva med. Man har således mycket att vinna på att klustra Proxmox VE-maskiner. Dessutom är ju Proxmox VE gratis att använda.