

Chapter 12: Rsync, Crontab en Scripting


Doel	Basis van shell scripting leren
Benodigheden	Een Linux omgeving
Tijdsduur	Gemiddeld 6 lesuren

Theorie: Rsync

Rsync (Remote Synchronisation) is een krachtig programma om bestanden en mappen te kopiëren. De kracht blijkt uit twee belangrijke mogelijkheden: rsync kopieert niet alleen van de ene naar de andere locatie, maar checkt ook eerst of bestanden gewijzigd zijn. Stel, je hebt een map Foto's van 10 GB die je wilt back-uppen naar een externe harde schijf. Je kopieert de map met rsync en hebt een back-up. Wanneer je een paar nieuwe foto's in de map zet, en deze ook naar de back-up wilt schrijven, kopieer je eenvoudig de hele map nog een keer. Rsync ziet de wijzigingen en kopieert alleen de nieuwe foto's, dit noemen we incremental backup. Dat scheelt dus tijd en netwerkbelasting! Bovendien kijkt rsync of er gewijzigde bestanden zijn (grootte, datum) en kopieert rsync deze bestanden ook. Verder kan rsync naar verschillende locaties, dus ook naar andere machines, kopiëren over het netwerk of over internet.

Hoewel er enorm veel opties zijn, gebruiken wij in de opdracht de belangrijkste optie:

```
sudo rsync -av /map-die-je-wilt-backuppen/ /waar-je-de-backup-opslaat  
rsync -av /bronmap/ /doelmap
```

 **Let op:** de slash (/) na bronmap/: deze zorgt ervoor dat de inhoud van de map bronmap gekopieerd wordt. Zonder deze slash wordt de hele map gekopieerd.

De optie **-a** zorgt ervoor dat alle attributen van de bestanden meegenomen worden (rechten, tijd), de optie **-v** zorgt ervoor dat je na het kopiëren feedback krijgt op wat gekopieerd is.

Opdracht 01: Gebruik maken van rsync

1. Ga naar je home map (door middel van **cd**) en maak hierin een map genaamd **afbeeldingen**.
2. Verplaats je naar de map afbeeldingen (**cd afbeeldingen**) en maak hierin 2 bestanden aan:

```
sudo touch foto01 foto02
```
3. Ga naar de map **/tmp** en maak daar een nieuwe map aan genaamd **backup**.
4. Kopieer nu de 2 bestanden naar de map **foto-back**:

```
sudo rsync -av /home/<jouw-gebruikersnaam>/afbeeldingen/ /tmp/backup
```
5. Controleer met de commando **ls -l** of de bestanden verplaatst zijn naar **/tmp/backup**. Maak een **screenshot voor in je logboek** nadat je de commando **ls -l** hebt gebruikt.
6. Zorg ervoor dat je weer in de map **afbeeldingen** staat. Maak in de map **afbeeldingen** een nieuw bestand genaamd **foto03**. Open **foto01** met het commando de nano commando en typ het volgende in het bestand: "**Dit is een test om de backup te controleren**". Voer het commando uit stap 4 nog een keer uit.
7. Controleer de bestanden met de commando **ls -l** in **/tmp/backup**. Maak een **screenshot voor in je logboek** met de uitkomst wanneer je het commando **ls -l** hebt gebruikt in **/tmp/backup**. Zie je enig verschil?
8. Verwijder de bestanden in de map afbeeldingen. Voer het commando uit stap 4 nog een keer uit. Je kopieert nu een lege map naar de map backup. Merk op dat de map backup **niet** leeggemaakt wordt.
9. Om de foto's terug te zetten voer je het commando omgekeerd uit:

```
sudo rsync -av /tmp/backup/ /home/<jouw-gebruikersnaam>/afbeeldingen
```
10. Maak een **screenshot voor in je logboek** met de uitkomst wanneer je het commando **ls -l** hebt gebruikt in **/home/<jouw-gebruikersnaam>/afbeeldingen**. Zijn de afbeeldingen teruggezet?



Het kan natuurlijk zijn dat je de verwijderde bestanden ook in de back-up wilt verwijderen. Daarvoor moet je de optie **--delete** gebruiken. Wees hier erg voorzichtig mee. Voor de zekerheid kun je met de optie **-n** oefenen: rsync doet dan alsof de handeling verricht wordt, jij kunt de feedback controleren en als je het ermee eens bent daadwerkelijk uitvoeren door de **-n** optie weg te halen.

1. Verwijder foto2 uit de map fotos.
2. Oefen nu droog met het commando samen met de delete-optie:

```
sudo rsync -n -av --delete /home/<jouw-gebruikersnaam>/afbeeldingen /tmp/backup/
```
3. Je ziet aan het einde (DRY RUN) staan om aan te geven dat niets echt gewijzigd is. Uiteraard ga je dit nog even controleren.
4. Voer hetzelfde commando nog een keer uit, maar zonder de optie **-n**.
5. Controleer of foto02 in de back-up ook verdwenen is.

Een back-up is een mooi voorbeeld van een proces om te automatiseren. Wanneer je elke dag rsync laat checken om te zien of iets gewijzigd is, houd je de back-up up-to-date. Dat kan met **Crontab**.

Theorie: Crontab

Crontab is een mogelijkheid om op vastgestelde tijden een commando automatisch uit te laten voeren. Wil je bijvoorbeeld elke zondag om 18.12 uur een back-up maken, dan kun je dit in Crontab invoeren. In Crontab kun je vijf velden gebruiken om bijvoorbeeld de dag, maand of tijd in te voeren in een vaste volgorde:

Minuten	0 – 59
Uur van de dag	0 – 24
Dag van de maand	1 – 31
Maand van het jaar	1 – 12
Dag van de week	0 – 6 (0 = zondag)

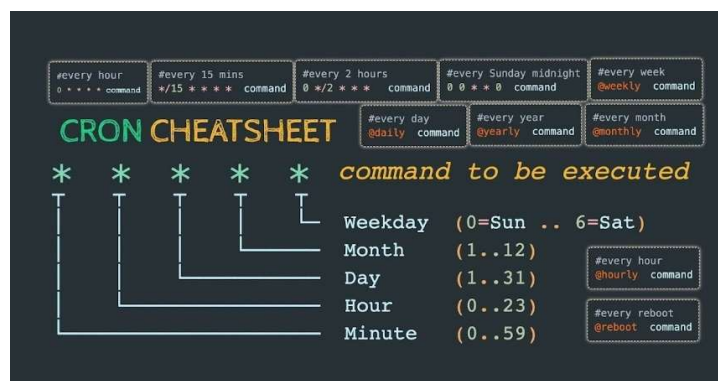
Een veld dat je niet gebruikt vul je met een *****.

Bijvoorbeeld: **12 18 * * 0 echo "Hallo" > /dev/tty2**

Op deze manier begroet je elke zondagavond om 12 minuten over 6 in de avond met Hallo in scherm 2.

 **Tip:** Kom je hier niet uit? Gebruik dan Google om te zoeken naar een crontab generator.

Bijvoorbeeld: <https://crontab-generator.org/>



Crontab instellen

Er zijn twee mogelijkheden om Crontab in te stellen: rechtstreeks invoeren in het bestand **/etc/crontab** en vervolgens de crondaemon starten (crontab bestandsnaam). Wij kiezen voor de optie om met het commando **crontab -e** automatisch het crontab-bestand aan te passen (**-e** staat voor edit).

Om het vorige voorbeeld in te voeren voer je het volgende commando in:

1. `sudo crontab -e`
2. Je gaat nu naar de editor **nano**, typ onderaan in het bestand, het volgende in:
3. **12 18 * * 0 echo "Hallo" > /dev/tty2**
4. Sla het bestand op.

Met het commando **sudo crontab -l** (list) kun je een lijst opvragen van alle crontabs waar jij rechten op hebt. Voer dit commando in om te controleren of het bovenstaande gelukt is. **Maak een screenshot voor in je logboek** met daarin de uitkomst van alle crontabs waar jij rechten op hebt. Met het commando **crontab -r** (remove) verwijder je de opdracht.

Door een komma te gebruiken kun je meerdere tijden in het tijdveld invullen (**15,30,45,0** in het eerste veld activeert elk 15 minuten het commando). Door ***/1** in het eerste veld in te voeren voer je het commando elke minuut uit.

Theorie: Wat is Bash Scripting?

BASH is een afkorting van **Bourne Again Shell**, een shell programma geschreven door Brian Fox als verbeterde versie van het Bourne Shell programma 'sh'. Het werd uitgebracht in 1989 als een van de meest populaire shell distributies van GNU/Linux besturingssystemen.



Wat is een shell?

Een shell-programma is een uitvoerbaar binair programma dat commando's verwerkt (die de gebruiker invoert) en deze vertaalt naar het aanroepen van bepaalde functionaliteiten van het besturingssysteem.

In de basis is Bash eigenlijk een interpreter van de commandline, die gewoonlijk wordt uitgevoerd in een programmavenster waarin de gebruiker commando's kan geven om diverse acties uit te voeren. De combinatie van deze commando's in een serie binnen een bestand, staat bekend als een Shell Script. Bash kan de commando's van een Shell Script lezen en uitvoeren. Van alle beschikbare shells behoort Bash tot de meest populaire, de krachtigste en de meest gebruiksvriendelijke.

Zijn er andere beschikbare Shells?

Bash is natuurlijk niet de enige beschikbare Shell, andere Shells zijn onder andere:

- 🐧 Bourne Shell (sh)
- 🐧 Almquist Shell (ash)
- 🐧 Debian Almquist Shell (dash)
- 🐧 kornShell (ksh)
- 🐧 Z Shell (zsh)

Van sommige heb je wellicht gehoord of deze misschien weleens gebruikt. De andere genoemde shells hebben verschillende regels, conventies, logica en achtergronden, wat betekent dat zij op elkaar kunnen lijken, maar toch op enkele subtiele punten van elkaar kunnen verschillen.

Een script moet aan drie voorwaarden voldoen:

- In de eerste regel roep je de juiste shell aan door het script te beginnen met **#!/bin/bash** (dit wordt de she-bang genoemd).
- Je moet het executerecht (**x**) geven om het script uitvoerbaar te maken.
- Je moet tijdens het uitvoeren het hele pad aangeven of dit pad vastleggen met het commando **path**.



Running Bash

De meeste moderne Linux en Unix distributies bieden Bash als standaard Shell aan. Dit omdat Bash zeer geliefd en bekend is en bovendien een aantal handige functies bevat die andere Shells niet ondersteunen. Bepaalde besturingssystemen gebruiken echter een andere shell.

Om uit te vinden of je Bash gebruikt, kun je het echo commando gebruiken, samen met een speciale variabele die de naam van het huidige proces weergeeft:

```
sudo echo $0
```

Probeer de commando hierboven uit en **schrijf in je logboek** welke **shell** jij gebruikt.

Opdracht 02: Maken van een script

1. Om je kennis te laten maken met Shell Scripting maken we een simpel en kort script aan:

```
sudo nano hallo.sh
```



Tip: zoals je weet hoeft je in Linux geen extensies te gebruiken (.txt .exe .docx etc) zoals je ziet doen wij dit hier wel. Dit doen we om zo onderscheid voor de menselijk oog te maken en de beheerder weet wat een script is of een ander bestand. Het is dus makkelijker maar niet verplicht.

2. Plaats vervolgens de volgende regels in je **hallo.sh** bestand:

```
#!/bin/bash
#Print Hallo wereld bericht
echo "Hello World!"
```

- De eerste regel: **#!/bin/bash** is bekend als de shebang header.
 - De tweede regel: **#Print Hallo wereld bericht** is een opmerking wat weergeeft wat het script daaronder gaat uitvoeren. Dit is zeer belangrijk om mee te geven omdat dan iemand die de script voor het eerst ziet ook weet wat er gaat gebeuren.
 - De derde regel: **echo "Hello World!"** is de daadwerkelijke commando die de script gaat uitvoeren.
3. Sla het bestand op doormiddel van **<CTRL+O>** en daarna op **<ENTER>** aangezien we de script al een naam hebben gegeven. Sluit **nano** nu af doormiddel van de toets combinatie: **<CTRL+X>**.

4. Om de script te kunnen gebruiken moeten we de script nog rechten geven:

```
sudo chmod +x hallo.sh
```

5. Nu kunnen we de script uitvoeren dit doen we als volgt:

```
sudo bash hallo.sh
```

of

```
./hallo.sh
```

6. Maak een **screenshot voor in je logboek** met daarin de uitkomst van de script.

Je weet nu hoe je een script moet maken. Nog even samengevat:

- Begin met de she-bang (#!/bin/bash).
- Zorg dat je x-recht geeft.
- Maak gebruik van commando's.

Je hebt geleerd dat een Linux-script geen extensie hoeft te hebben. Een goed gebruik is om toch een script te laten eindigen op .sh om aan te geven dat het om een script gaat. Bovendien voorkom je hiermee dat je een naam voor een script gebruikt dat ook een commando is (zoals test). Ook een goed gebruik is om in het script een beschrijving te geven van wat het script doet. Om deze tekst te verbergen voor de uitvoer van het script begin je elke regel hiervan met #.



Theorie: Variabelen

Er zijn 'ingebouwde' variabelen die je gebruiken kunt. Door **\$USER** te gebruiken wordt de gebruikersnaam getoond. **\$HOME** toont het pad van de homedirectory.

Opdracht 03a: Werken met variabelen

In het volgende script wordt de hiervoor behandelde theorie duidelijk. Maak dit script na, noem het: **script1.sh**

⚠ **Let op** het gebruik van de dubbele aanhalingstekens (" "). Wanneer een zin ingesprongen is, betekent dit dat het dezelfde regel is.

1. Log in als **root**: `sudo su -`
2. Ga naar **/tmp**. Maak een bestand met de hiervoor genoemde naam: `nano script1.sh`
3. Typ de volgende tekst over in het bestand:
#!/bin/bash
#Dit script geeft de user en home dir weer
echo "Ik ben de gebruiker" "\$USER" "en mijn homedirectory is" "\$HOME"
4. Sla het script op en sluit nano af.
5. Geef alle rechten voor de gebruiker en de groep: `chmod 770 script1.sh`
6. Zorg ervoor dat de groep **Users** groepseligenaar is: `chgrp users script1.sh`
7. Voer het script als **root** uit (./script1.sh) en **maak een screenshot** van de resultaat **voor in je logboek**.
8. Log weer in als jezelf: `su <jouw gebruikersnaam>`
9. Ga naar /tmp, voer het script uit en **beschrijf de verschillen in je logboek**.

Opdracht 03b: Variabele declareren

Je kunt zelf een variabele declareren (vastzetten) in een script. Dit doe je door in hoofdletters de naam van de variabele te typen en na het =-teken de waarde, zonder spaties. **VOORNAAM=John** bijvoorbeeld. Het is een goed gebruik om variabelen met hoofdletters te typen. Dit verduidelijkt de code in het script. Je roept een variabele vervolgens aan door in de tekst een **\$** te gebruiken: **\$VOORNAAM**.

In het volgende script hebben we de variabelen gedeclareerd:

```
#!/bin/bash
#Dit script toont variabelen
VOORNAAM=Linus
ACHTERNAAM=Torvalds
echo "Mijn voornaam is" "$VOORNAAM" "en mijn achternaam is" "$ACHTERNAAM"."
```

Neem het script hierboven over maar geef daar je eigen naam en achternaam weer. Maak vervolgens een **screenshot voor in je logboek** met daarin de uitkomst van het script.

Uiteraard willen we dat een variabele, variabel ingevoerd kan worden. Dit leren we in [Chapter 16: Scripting Part 02](#) met de commando **read**. Voor nu gaan we eerst kijken wat er allemaal mogelijk is met gedeclareerde variabelen in een script.



Opdracht 03b: Parameters

Zodra je het commando **env** gebruikt krijg je een lijst met de standaardvariabelen, probeer maar eens uit. Met het **export** commando gaan we (voor deze sessie) een nieuwe variabele aanmaken. We willen dat de variabele **\$distro** als output '**Debian**' geeft. Eerst controleren we of deze variabele al bestaat:

- Typ: `echo $distro`
- Geen output
- Typ: `export distro='Debian'`

⚠ **Let op:** de opgegeven parameter moet tussen de '-'tekens staan.

- Typ: `echo $distro`
- Controleer eventueel of de variabele in de shell terug te vinden is: `env | grep distro`. Je ziet dat voor deze sessie **\$distro** te gebruiken is als variabele.
- Met het commando `unset distro` verwijder je de variabele weer. Maar wanneer je uitlogt, is deze ook weer verdwenen.

Je kunt een script tijdens het uitvoeren parameters meegeven en deze parameters gebruiken in jouw script. Stel, jouw script heet `script2.sh` en je voert dit uit met de parameters **rood spruitjes** en **gamen**. Dan kun je deze parameters als volgt gebruiken in jouw script. Je kan ook een variabele in de shellomgeving aanmaken.

Aan de slag met parameters:

1. Parameters werken op basis van de ingevoerde volgorde van links naar rechts. `$0` is de naam van het script, `$1` is de eerste parameter, enzovoort.
2. Maak een script aan genaamd **script3.sh**

```
#!/bin/bash
#Dit script gebruikt parameters als input voor variabelen.
echo "De naam van dit script is" "$0" ", mijn favoriete kleur is" "$1" ", ik eet graag" "$2" "en ik houd van" "$3" "."
```

⚠ **Let op:** onderste 2 regels, is 1 zin dit komt dus achterelkaar te staan.

⚠ **Let op de spaties:** twee dubbele quotes zijn altijd met een spatie van elkaar gescheiden.

3. We gaan nu het script uitvoeren op de volgende manier: `./script3.sh rood spruitjes gamen`
4. Maak een **screenshot voor in je logboek** met daarin de uitkomst van het script. Probeer ook eens met andere woorden.

