GABRIELE ROMANATO

Come implementare un sistema di backup incrementale su un server remoto utilizzando Bash e i cronjob

Implementare un sistema di backup incrementale su un server remoto è fondamentale per garantire la sicurezza dei dati, ridurre al minimo l'utilizzo dello spazio di archiviazione e ottimizzare i tempi di esecuzione. Utilizzando Bash e i cronjob, è possibile automatizzare i processi di backup, assicurando che vengano eseguiti regolarmente senza intervento manuale. In questo articolo, esploreremo i passi fondamentali per configurare un sistema di backup incrementale.

Il **backup incrementale** è un tipo di backup che memorizza solo i file che sono stati modificati o aggiunti rispetto al backup precedente. Rispetto al backup completo, richiede meno spazio su disco e riduce il tempo di esecuzione.

Per questo scopo, utilizzeremo `rsync`, un potente strumento che sincronizza file e directory tra server locali e remoti in modo efficiente, supportando il trasferimento incrementale.

Iniziamo assicurandoci che `rsync` sia installato sia sul server locale che su quello remoto. È possibile installarlo con i seguenti comandi:

```
sudo apt update  
sudo apt install rsync
```

Per evitare di dover inserire la password ogni volta che eseguiamo il backup, configuriamo l'autenticazione SSH con chiave pubblica.

```
ssh-keygen -t rsa
```

Copiamo quindi la chiave pubblica sul server remoto:

```
ssh-copy-id user@remote_server
```

Creiamo uno script Bash che esegue il backup incrementale. Supponiamo di voler fare il backup della directory `/var/www` (contenente file web) su un server remoto.

```
#!/bin/bash

# Configurazione
SRC_DIR="/var/www/" # Directory di origine
DEST_DIR="/path/backup/" # Directory di destinazione sul
server remoto
REMOTE_USER="user" # Nome utente del server remoto
REMOTE_HOST="remote_server" # Host del server remoto
BACKUP_DATE=$(date +%Y-%m-%d) # Data corrente per il
backup
LOG_FILE="/var/log/backup.log" # File di log

# Backup incrementale usando rsync
rsync -avz --delete --link-dest=${DEST_DIR}/latest ${SRC_DIR}
${REMOTE_USER}@${REMOTE_HOST}:${DEST_DIR}/${BACKUP_DATE} >>
${LOG_FILE} 2>&1

# Aggiorna l'ultimo backup
ssh ${REMOTE_USER}@${REMOTE_HOST} "rm -rf ${DEST_DIR}/latest
&& ln -s ${DEST_DIR}/${BACKUP_DATE} ${DEST_DIR}/latest"
```

Spiegazione dello script:

- **SRC_DIR**: la directory locale che desideriamo salvare.
- **DEST_DIR**: la directory di backup sul server remoto.
- **BACKUP_DATE**: viene usata per creare una nuova cartella per ogni backup incrementale, utilizzando la data corrente.
- **rsync -avz**: sincronizza i file da SRC_DIR a DEST_DIR sul server remoto. Il flag `--link-dest` fa riferimento all'ultimo backup, quindi solo i file modificati o nuovi vengono copiati.
- **ssh**: viene utilizzato per aggiornare il collegamento simbolico `latest` sul server remoto, in modo che punti sempre all'ultimo backup eseguito.

Rendiamo quindi eseguibile lo script:

```
chmod a+x backup.sh
```

Il passo successivo è automatizzare l'esecuzione dello script di backup utilizzando cron, il demone che consente di programmare attività su Linux.

Modifica il file crontab per l'utente root o l'utente specifico (se vuoi effettuare backup su server remoto):

```
crontab -e
```

Aggiungi una nuova riga per eseguire ad esempio lo script ogni giorno alle 02:00:

```
0 2 * * * /path/backup.sh
```

Questo comando dice a cron di eseguire lo script di backup ogni giorno alle 02:00. Puoi modificare questa programmazione in base alle tue esigenze.

È buona norma verificare regolarmente lo stato dei backup e dei log. Puoi controllare il file di log creato dallo script:

```
cat /var/log/backup.log
```

Assicurati che non ci siano errori e che i file vengano copiati correttamente sul server remoto.

Ottimizzazione e Sicurezza

- **Compressione:** rsync supporta l'opzione `-z` per comprimere i file durante il trasferimento, riducendo l'uso della larghezza di banda.
- **Criptazione:** Utilizzare l'SSH per garantire che i dati siano criptati durante il trasferimento.
- **Backup off-site:** È buona pratica mantenere una copia dei backup in una posizione geografica diversa, nel caso di disastri naturali o problemi hardware.
- **Cancellazione automatica:** Puoi aggiungere una funzione per cancellare automaticamente i backup più vecchi di un certo numero di giorni, utilizzando un comando `find` nel tuo script.

Esempio:

```
ssh ${REMOTE_USER}@${REMOTE_HOST} "find ${DEST_DIR} -type d -mtime +30 -exec rm -rf {} \;"
```

Questo comando cancella i backup più vecchi di 30 giorni.

Conclusione

Con questa guida, hai configurato un sistema di backup incrementale utilizzando Bash, rsync e cronjob. Questa soluzione ti consente di risparmiare spazio su disco e larghezza di banda rispetto a un backup completo giornaliero e garantisce che i dati critici vengano salvati in modo sicuro su un server remoto. Personalizza lo script e la pianificazione cron in base alle tue esigenze aziendali per massimizzare l'efficienza del tuo sistema di backup.