Monitorare l'uptime di un sito web con Bash, cron e systemd

Assicurarsi che un sito web rimanga online e accessibile è cruciale per qualsiasi azienda o individuo che dipenda dalla presenza online. Il monitoraggio automatico può aiutare a identificare rapidamente i tempi di inattività e i problemi di prestazioni, consentendo di prendere provvedimenti correttivi tempestivi.

Questo script Bash verifica lo stato di un sito web a intervalli regolari e registra se il sito è attivo o inattivo, insieme al codice di stato HTTP e al tempo impiegato per ogni controllo.

```
#!/bin/bash

# URL di base del sito da monitorare
site_url=$1

# Intervallo di tempo tra le richieste in secondi
delay_between_requests=60

# Funzione per controllare lo stato del sito
check_site() {
    # Definire l'user agent per la richiesta
    local user_agent="Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/58.0.3029.110 Safari/537.36"

    # Registrare l'ora di inizio della richiesta
    local start_time=$(date +%s)

    # Ottenere il codice di stato HTTP dalla richiesta
    local status_code=$(curl -H "$user_agent" -s -o /dev/null
-w "%{http_code}" $site_url)
```

```
    # Ottenere la data e l'ora attuali per la registrazione
    local date_time_log=$(date +"%Y-%m-%d %H:%M:%S")

    # Registrare l'ora di fine della richiesta
    local end_time=$(date +%s)

    # Calcolare il tempo trascorso per la richiesta
    local elapsed_time=$((end_time - start_time))

    # Registrare lo stato in base al codice di stato HTTP
    if [ $status_code -ne 200 ]; then
        echo "[$date_time_log]: Il sito è offline. Codice di
stato: $status_code - Tempo trascorso: $elapsed_time secondi"
    else
        echo "[$date_time_log]: Il sito è online. Codice di
stato: $status_code - Tempo trascorso: $elapsed_time secondi"
    fi
}

# Loop infinito per controllare ripetutamente il sito
while true; do
    check_site
    sleep $delay_between_requests
done
```

Come funziona lo script:

1. **Input dell'URL**: Lo script prende l'URL del sito web come argomento quando viene eseguito.

2. **Intervallo di richieste**: La variabile `delay_between_requests` specifica il ritardo in secondi tra ogni controllo, impostato a 60 secondi per impostazione predefinita.

3. **User Agent**: La variabile `user_agent` è impostata per imitare una richiesta da un comune browser web, assicurando che la richiesta sia gestita in modo simile a una visita reale.

4. **Richiesta HTTP e cronometraggio**: La funzione `check_site`:
    - Registra l'ora di inizio.
    - Invia una richiesta HTTP GET utilizzando `curl`, catturando il codice di stato HTTP.
    - Registra l'ora di fine.
    - Calcola il tempo trascorso per la richiesta.
5. **Registrazione**: Lo script registra la data e l'ora attuali, il codice di stato HTTP e il tempo trascorso per ogni controllo. Se il codice di stato non è 200, indica che il sito è offline.
6. **Loop infinito**: Lo script funziona indefinitamente, controllando lo stato del sito ogni 60 secondi (o l'intervallo specificato).

Utilizzo di cron

Per eseguire lo script tramite `cron`, è necessario rimuovere il loop infinito poiché `cron` gestirà l'esecuzione periodica dello script. Ecco come è possibile modificare lo script e configurarlo con `cron`.

Rimuovere il loop infinito e il comando `sleep` dallo script:

```
#!/bin/bash

# URL di base del sito da monitorare
site_url=$1

# Funzione per controllare lo stato del sito
check_site() {
    # Definire l'user agent per la richiesta
    local user_agent="Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/58.0.3029.110 Safari/537.36"

    # Registrare l'ora di inizio della richiesta
    local start_time=$(date +%s)
```

```
    # Ottenere il codice di stato HTTP dalla richiesta
    local status_code=$(curl -H "$user_agent" -s -o /dev/null
-w "%{http_code}" $site_url)

    # Ottenere la data e l'ora attuali per la registrazione
    local date_time_log=$(date +"%Y-%m-%d %H:%M:%S")

    # Registrare l'ora di fine della richiesta
    local end_time=$(date +%s)

    # Calcolare il tempo trascorso per la richiesta
    local elapsed_time=$((end_time - start_time))

    # Registrare lo stato in base al codice di stato HTTP
    if [ $status_code -ne 200 ]; then
        echo "[$date_time_log]: Il sito è offline. Codice di
stato: $status_code - Tempo trascorso: $elapsed_time secondi"
    else
        echo "[$date_time_log]: Il sito è online. Codice di
stato: $status_code - Tempo trascorso: $elapsed_time secondi"
    fi
}

# Chiamare la funzione check_site
check_site
```

Salvare lo script modificato in un file, ad esempio `check_site.sh` e assicurarsi che il file abbia i permessi di esecuzione.

```
chmod a+x check_site.sh
```

Ora aprire l'editor di crontab per l'utente corrente e aggiungere una nuova riga per programmare lo script. Ad esempio, per eseguire lo script ogni 5 minuti:

```
*/5 * * * * /path/to/check_site.sh https://example.com >>
/path/to/logfile.log 2>&1
```

Questa riga esegue le seguenti operazioni:

- `*/5 * * * *`: Esegue lo script ogni 5 minuti.
- `/path/to/check_site.sh https://example.com`: Specifica il percorso dello script e l'URL da controllare.
- `>> /path/to/logfile.log 2>&1`: Aggiunge l'output a `logfile.log` e reindirizza gli errori allo stesso file.

Configurando lo script con `cron`, si delega al demone cron la responsabilità dell'esecuzione periodica, assicurando che lo script venga eseguito agli intervalli specificati senza la necessità di un loop infinito all'interno dello script stesso.

Utilizzo di systemd

Per eseguire lo script tramite `systemd`, è necessario creare un file di unità di servizio systemd e un file di unità timer per programmare l'esecuzione dello script.

Rimuovere il loop infinito e il comando `sleep` dallo script:

```
#!/bin/bash

# URL di base del sito da monitorare
site_url=$1

# Funzione per controllare lo stato del sito
check_site() {
    # Definire l'user agent per la richiesta
    local user_agent="Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
Chrome/58.0.3029.110 Safari/537.36"

    # Registrare l'ora di inizio della richiesta
    local start_time=$(date +%s)

    # Ottenere il codice di stato HTTP dalla richiesta
    local status_code=$(curl -H "$user_agent" -s -o /dev/null
-w "%{http_code}" $site_url)

    # Ottenere la data e l'ora attuali per la registrazione
    local date_time_log=$(date +"%Y-%m-%d %H:%M:%S")

    # Registrare l'ora di fine della richiesta
    local end_time=$(date +%s)

    # Calcolare il tempo trascorso per la richiesta
    local elapsed_time=$((end_time - start_time))

    # Registrare lo stato in base al codice di stato HTTP
    if [ $status_code -ne 200 ]; then
        echo "[$date_time_log]: Il sito è offline. Codice di
stato: $status_code - Tempo trascorso: $elapsed_time secondi"
    else
        echo "[$date_time_log]: Il sito è online. Codice di
stato: $status_code - Tempo trascorso: $elapsed_time secondi"
    fi
}

# Chiamare la funzione check_site
check_site
```

Salvare lo script modificato in un file e assicurarsi che il file abbia i permessi di esecuzione. Quindi creare un nuovo file in `/etc/systemd/system/check_site.service` con il seguente contenuto:

```
[Unit]
Description=Controlla lo stato del sito

[Service]
```

```
Type=oneshot
ExecStart=/usr/local/bin/check_site.sh https://example.com
```

Ora creare il file di unità timer systemd aggiungendo un nuovo file in `/etc/systemd/system/check_site.timer` con il seguente contenuto:

```
[Unit]
Description=Esegui lo script Check Site ogni 5 minuti

[Timer]
OnBootSec=5min
OnUnitActiveSec=5min
Unit=check_site.service

[Install]
WantedBy=timers.target
```

Ricaricare la configurazione del gestore systemd per riconoscere i nuovi file di unità.

```
sudo systemctl daemon-reload
```

Abilitare il timer per avviarlo all'avvio del sistema.

```
sudo systemctl enable check_site.timer
```

Avviare il timer immediatamente.

```
sudo systemctl start check_site.timer
```

Infine, verificare che il timer sia attivo.

```
sudo systemctl status check_site.timer
```

Per impostazione predefinita, l'output dello script eseguito da `systemd` verrà catturato dai log `journalctl`. È possibile visualizzare i log utilizzando:

```
journalctl -u check_site.service
```

Se si desidera registrare l'output in un file specifico, è possibile modificare il file di servizio:

```
[Service]
Type=oneshot
ExecStart=/usr/local/bin/check_site.sh https://example.com
StandardOutput=append:/var/log/check_site.log
StandardError=append:/var/log/check_site.log
```

Configurando lo script con `systemd`, si delega la responsabilità dell'esecuzione periodica alle unità timer di `systemd`. Questo approccio è robusto e si integra bene con il processo di init del sistema, fornendo migliori capacità di gestione e registrazione rispetto a `cron`.

Conclusione

Questo semplice script Bash fornisce uno strumento fondamentale per il monitoraggio dell'uptime di un sito web. Regolando i suoi parametri e ampliando le sue funzionalità, può essere adattato per soddisfare esigenze di monitoraggio più complesse. Automatizzare tali compiti assicura che gli amministratori del sito possano rimanere informati sullo stato del proprio sito e rispondere rapidamente a eventuali problemi che si presentano.