



INSTALLAZIONE GIT SU INFRASTRUTTURA SPLUNK

Summary

1	Introduzione	1
1.1	Configurazione Docker Compose e Dockerfile.....	1
1.1.1	Dockerfile	1
1.2	Creazione e Configurazione Server Git	1
1.3	Collegamento di Splunk con il Server Git	1
1.4	Dettagli Tecnici	1
1.5	Conclusione	2
2	Istruzioni Dettagliate per Avvio e Configurazione	2
2.1	Avvio e Gestione Docker	2
2.2	Creazione e Configurazione del Server Git.....	3
2.3	Collegamento Client Git al Server	3
2.4	Abilitazione Script per il Push automatico	3
2.5	Errori Comuni	4

1 Introduzione

In questa documentazione viene descritto in dettaglio il progetto realizzato che comprende la creazione di un ambiente Docker con più container, la configurazione di un server Git privato, e il collegamento di Splunk con questo server Git per la gestione dei dati e degli indici. Verranno spiegati passo dopo passo i file utilizzati, la configurazione e le modalità di comunicazione tra i servizi.

1.1 Configurazione Docker Compose e Dockerfile

Per facilitare la gestione e l'orchestrazione dei container Docker è stato utilizzato Docker Compose. Questo strumento permette di definire e gestire più container come un unico sistema.

Il file principale è il `docker-compose.yml` che contiene la definizione dei servizi (container) che compongono il sistema: ad esempio il Search Head (SH), l'Indexer (IDX) e l'Heavy Forwarder (HF).

1.1.1 Dockerfile

Per ogni servizio viene creato un Dockerfile dedicato, che specifica l'immagine base, le variabili d'ambiente, i comandi di setup e i volumi da montare per la persistenza dei dati e configurazioni.

Il Dockerfile non contengono script di inizializzazione personalizzati ma utilizzano esclusivamente variabili d'ambiente e file di configurazione montati tramite volumi per configurare Splunk.

1.2 Creazione e Configurazione Server Git

Per gestire il codice e le configurazioni in modo centralizzato è stato creato un server Git privato. Il server Git è stato configurato su una macchina virtuale o container dedicato, con accesso SSH e permessi corretti.

La configurazione prevede:

- Creazione utente git
- Setup repository bare
- Configurazione SSH per accesso sicuro
- Permessi per push e pull

Inoltre, è stato implementato uno script di auto-push che automatizza il push delle modifiche verso il server Git.

1.3 Collegamento di Splunk con il Server Git

Splunk è stato configurato per collegarsi al server Git per sincronizzare le configurazioni e i dati. Attraverso i file di configurazione montati nel container e l'accesso SSH, Splunk può effettuare pull e push delle configurazioni.

Questo permette di avere un ambiente di lavoro replicabile e sincronizzato tra più nodi Splunk, fondamentale in ambienti distribuiti.

1.4 Dettagli Tecnici

Sono state usate le seguenti tecnologie e strumenti:

- Docker e Docker Compose per la containerizzazione
- Splunk Enterprise Docker image
- Git server configurato su SSH
- Script bash per automazione dei push git

- File di configurazione Splunk (outputs.conf, distsearch.conf) montati tramite volumi
- Network Docker personalizzato per la comunicazione tra container

Sono stati inoltre risolti problemi come:

- Configurazione delle chiavi SSH per accesso senza password
- Gestione corretta dei permessi e utenti
- Evitare script di inizializzazione complessi, favorendo configurazione via file e variabili d'ambiente
- Risoluzione problemi di comunicazione tra container

1.5 Conclusione

Questo progetto ha permesso di creare un ambiente Splunk multi-container orchestrato con Docker Compose, integrato con un server Git privato per la gestione delle configurazioni e del codice. La soluzione implementata è scalabile, modulare e facilmente gestibile, permettendo una configurazione centralizzata e sicura.

2 Istruzioni Dettagliate per Avvio e Configurazione

2.1 Avvio e Gestione Docker

Per avviare l'ambiente Docker con i container configurati è necessario utilizzare Docker Compose. Ecco i passaggi principali:

1. Assicurarsi di avere Docker e Docker Compose installati sul sistema.
2. Posizionarsi nella cartella contenente il file docker-compose.yml e Dockerfile.
3. Eseguire il comando:

```
'''
docker-compose up -d --build
'''
```

Questo comando avvierà tutti i container definiti nel file in modalità detached (in background).

4. Per verificare lo stato dei container, eseguire:

```
'''
docker-compose ps
'''
```

5. Una volta avviato il docker bisogna collegare il search head all'indexer.

Dal Search Head:

1. Vai su Settings → Distributed Search → Search peers
2. Dovresti vedere il peer `idx:8089` con status UP
3. Se lo vedi come Down con errore: Authentication credentials rejected by peer, allora l'autenticazione è fallita.

Per risolvere clicca sul nome e inserisci manualmente:

- Username: admin
- Password: Splunk@00

Per fermare e rimuovere i container:

```
'''
docker-compose down
'''
```

2.2 Creazione e Configurazione del Server Git

La creazione di un server Git privato prevede i seguenti passaggi:

1. Assicurare di aver installato openssh e git, nel caso: `sudo apt update && sudo apt install git openssh-server`
2. Creare la cartella dove risiederanno i repository Git, ad esempio `/home/git/repositories``.
3. Inizializzare un repository bare per la condivisione:

```
...  
git init --bare nome_repository.git  
...
```

4. Assicurarsi che i permessi siano corretti per l'utente git e per l'accesso SSH.

2.3 Collegamento Client Git al Server

Fase da fare su macchina Splunk per collegare server Git, **è fortemente consigliato fare queste operazioni con l'utente Splunk:**

1. Configurare l'accesso SSH per l'accesso al server Git:

- Genera la chiave: `$ ssh-keygen -t rsa -b 4096`
- Copia la chiave sul server: `$ ssh-copy-id utente_server_git@ip_server_git`

2. Configurare git:

```
$ git config --global user.name "Tuo Nome"  
$ git config --global user.email "tuo@email.com"
```

3. Creare il repository con git init nella cartella che si desidera monitorare, nel nostro caso `/opt/splunk`

4. Collegare il repository locale con quello del server:

```
git remote add origin utente_server_git@ip_server_git:/home/git/nome_repository.git
```

5. Aggiungere il file `.gitignore`, il file contiene quali solo le directory e i file da ignorare

6. Fare il primo commit e il primo push:

```
$ git add .  
$ git commit -m "Primo commit"  
$ git push -u origin master
```

NOTA: Potrebbe esserci l'errore *path/ does not have a commit checked out fatal*, vedere nella sezione [Errori Comuni](#)

2.4 Abilitazione Script per il Push automatico

L'obiettivo è quello di eseguire automaticamente ``git add``, ``commit`` e ``push`` ogni volta avviene un cambiamento nella macchina, con o senza usare ``inotifywait``.

Nel caso in cui non si può installare inotifywait, usiamo uno script che ogni 30 secondi controlla se c'è stato un cambiamento e fa il commit. Invece con inotifywait ad ogni cambiamento fa il commit e push in automatico.

ATTENZIONE: Lo script non si avvia in automatico nel caso viene riavviata la macchina.

(In contesti reali, bisogna impostare un daemon per l'avvio al boot della macchina).

Installazione:

1. Crea il file:

```
sudo touch /usr/local/bin/auto-git-push.sh
```

2. Incolla il contenuto dello script auto-git-push.sh, nel caso dello script con inotifywait il file si chiama wlib-auto-git-push.sh

3. Rendi eseguibile:

```
chmod +x /usr/local/bin/auto-git-push.sh
```

4. Avvia in background:

```
nohup /usr/local/bin/auto-git-push.sh &
```

5. Verifica il funzionamento aprendo un altro terminale e avviare il processo tail -f nohup.out

2.5 Errori Comuni

Fatal detected dubious ownership:

non inizializzare Git nel root, usa una cartella come /opt/git.

remote origin already exists:

rimuovi con git remote remove origin

no identities found:

genera le chiavi SSH.

Connection refused:

assicurati che il servizio ssh sia attivo.

path/ does not have a commit checked out fatal, adding files failed:

In quel path esiste un file .git per risolvere bisogna eliminarlo.