RETI DI CALCOLATORI

Lab 17/10/2023

Telnet

telnet è uno strumento che permette di stabilire una connessione a un server remoto utilizzando il protocollo telnet (porta 23).

Era ampiamente utilizzato anni fa per le connessioni remote prima dell'avvento di protocolli più sicuri come SSH.

Oltre a questo uso principale, telnet può anche essere utilizzato per testare la connettività di servizi su altre porte.

telnet - creiamo un container

```
FROM ubuntu:latest

# Aggiorna la lista dei pacchetti e installa telnetd, curl, wget e dnsutils (per dig e nslookup)

RUN apt-get update && apt-get install -y telnetd curl wget dnsutils && apt-get clean && rm -rf /var/lib/apt/lists/*

# Crea un nuovo utente e impostala password

RUN useradd demo && echo "demo:demo123" | chpasswd

# Avvia il server telnet al boot del container

CMD ["/usr/sbin/in.telnetd", "-debug", "23"]
```

telnet - build del container

1. In questo Dockerfile:
    1. Aggiungiamo il pacchetto telnetd, curl, wget e dnsutils.
    2. Creiamo un nuovo utente chiamato `demo`.
    3. Impostiamo la password dell'utente `demo` come `demo123`.
    4. Alla fine, avviamo il server telnet.
2. Per costruire e avviare il container:
    1. Costruisci l'immagine:

```
docker build -t telnet_server .
```

3. Avvia il container:

```
docker run -d -p 23:23 telnet_server
```

il nome utente `demo` e la password `demo123`:

```
telnet localhost 23
```

DIG

"dig" è un comando utilizzato in ambiente Unix e Linux per interrogare server DNS (Domain Name System). Il nome "dig" è un acronimo che sta per "Domain Information Groper". Questo strumento è ampiamente usato dai professionisti IT per risolvere problemi relativi ai DNS e per ottenere informazioni dettagliate sul dominio.

Quando un utente esegue il comando "dig" seguito da un nome di dominio, "dig" invia una richiesta al server DNS e restituisce una risposta con le informazioni sul dominio specificato. La risposta può includere informazioni come l'indirizzo IP associato al nome di dominio, informazioni sul server mail, i nameserver associati e molte altre informazioni utili.

Ecco un esempio semplice di utilizzo:

```
dig example.com
```

Questo comando restituirà informazioni sul dominio "example.com", inclusi gli indirizzi IP associati.

"dig" offre anche una serie di opzioni che permettono all'utente di specificare il tipo di query (ad esempio, A, MX, NS), il server DNS da interrogare e molte altre configurazioni avanzate.

> dig example.com

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23870
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;example.com.                   IN    A
;; ANSWER SECTION:
example.com.        47317    IN    A    93.184.216.34
;; Query time: 8 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Sun Oct 15 20:51:43 UTC 2023
;; MSG SIZE  rcvd: 56
```

dig - output 1-4

1. **Intestazione**:

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com
```

Questo è semplicemente l'intestazione che mostra la versione di `dig` che stai usando e il dominio che stai interrogando (`example.com`).

2. **Opzioni Globali**:

```
;; global options: +cmd
```

Questo mostra le opzioni globali utilizzate nella tua query. In questo caso, `+cmd` indica che la linea del comando di input viene visualizzata nell'output.

3. **Risposta Ricevuta**:

```
;; Got answer:
```

Indica che `dig` ha ricevuto una risposta alla sua query.

dig - output 2-4

4. **Sezione HEADER**:

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23870
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
```

Questa sezione fornisce informazioni sull'intestazione della risposta DNS:

- `opcode`: Tipo di query. Di solito è `QUERY`.
- `status`: Stato della risposta. `NOERROR` significa che non ci sono stati errori.
- `id`: Un identificativo univoco per la query.
- `flags`: Bandiere della risposta.
  - `qr` significa "query response".
  - `rd` significa "recursion desired".
  - `ra` significa "recursion available".
- Le altre voci contano il numero di records nelle diverse sezioni della risposta: `QUERY`, `ANSWER`, `AUTHORITY`, e `ADDITIONAL`.

dig - output 3-4

5. **Sezione OPT PSEUDOSECTION**:

```
; EDNS: version: 0, flags:; udp: 4096
```

Questo si riferisce a Extension Mechanisms for DNS (EDNS) che estende le capacità di DNS. In questo caso, indica la versione di EDNS e la dimensione massima del pacchetto UDP supportato.

6. **Sezione QUESTION**:

```
;; QUESTION SECTION:
;example.com.                IN    A
```

Questo mostra la domanda che hai fatto. In questo caso, stai chiedendo l'indirizzo IP (record `A`) per `example.com`.

7. **Sezione ANSWER**:

```
;; ANSWER SECTION:
example.com.          47317 IN    A     93.184.216.34
```

Questa è la risposta alla tua domanda. Ti dice che l'indirizzo IP per `example.com` è `93.184.216.34`.

dig - output 4-4

8. **Dettagli Aggiuntivi**:

```
;; Query time: 8 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Sun Oct 15 20:51:43 UTC 2023
;; MSG SIZE  rcvd: 56
```

Queste sono informazioni aggiuntive sulla tua query:

- `Query time`: Quanto tempo ha impiegato la query.
- `SERVER`: Il server DNS a cui hai fatto la query.
- `WHEN`: Quando è stata fatta la query.
- `MSG SIZE`: La dimensione del messaggio ricevuto.

> nslookup example.com

Server: 192.168.0.1

Address: 192.168.0.1#53

Non-authoritative answer:

Name: example.com

Address: 93.184.216.34

Name: example.com

Address: 2606:2800:220:1:248:1893:25c8:1946

nslookup output 1-2

`nslookup` è un altro strumento utilizzato per interrogare i server DNS.
Ecco una spiegazione dell'output:

1. **Server e Indirizzo**:

```
Server:         192.168.0.1

Address:   192.168.0.1#53
```

Queste righe indicano il server DNS che `nslookup` sta utilizzando per la sua query. In questo caso, stai utilizzando il server DNS situato all'indirizzo IP `192.168.0.1` sulla porta `53`, che è la porta standard per il DNS.

2. **Non-authoritative answer**:

```
Non-authoritative answer:
```

Questa è un'etichetta che ti indica che la risposta fornita non proviene da un server che ha autorità diretta sul dominio richiesto (cioè non proviene da uno dei server DNS ufficiali per `example.com`). Le risposte non autorevoli solitamente provengono da server DNS cache, che hanno memorizzato le informazioni da richieste precedenti.

nslookup output 2-2

3. **Sezione Answer**:

```
Name: example.com

Address: 93.184.216.34

Name: example.com

Address: 2606:2800:220:1:248:1893:25c8:194
```

Questa sezione fornisce le risposte alla tua query. Ti mostra gli indirizzi IP associati al dominio `example.com`:

- Il primo indirizzo (`93.184.216.34`) è un indirizzo IPv4.
- Il secondo indirizzo (`2606:2800:220:1:248:1893:25c8:1946`) è un indirizzo IPv6.

Questo significa che `example.com` ha sia un record `A` (per IPv4) sia un record `AAAA` (per IPv6) nel DNS, e entrambi gli indirizzi ti sono stati mostrati come risultato.

> dig example.com @8.8.8.8

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45988
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;example.com.                IN    A
;; ANSWER SECTION:
example.com.      2759    IN    A    93.184.216.34
;; Query time: 48 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Sun Oct 15 21:05:19 UTC 2023
;; MSG SIZE  rcvd: 56
```

> dig example.com @8.8.8.8 (1-2)

1. `; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com @8.8.8.8`
   - L'utente sta utilizzando `DiG` versione `9.18.12` su un sistema Ubuntu `22.04.3`.
   - L'utente sta interrogando il dominio `example.com` utilizzando il server DNS `8.8.8.8`, che è un server DNS pubblico fornito da Google.
2. `;; global options: +cmd`
   - L'utente ha impostato alcune opzioni globali per la sua query. In particolare, `+cmd` mostra il comando utilizzato all'inizio dell'output.
3. `;; Got answer:`
   - Questa linea indica che il server DNS ha restituito una risposta.
4. `;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45988`
   - L'utente ha inviato una operazione di tipo `QUERY` al server DNS.
   - Lo stato della risposta è `NOERROR`, il che indica che la query è stata eseguita con successo.
   - L'ID associato a questa query è `45988`.
5. `;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1`
   - Ci sono diverse bandiere (flags) nella risposta: `qr` (query response), `rd` (recursion desired), `ra` (recursion available), e `ad` (authenticated data).
   - La sezione QUESTION contiene 1 voce.
   - La sezione ANSWER contiene 1 voce.
   - Non ci sono voci nella sezione AUTHORITY.
   - C'è 1 voce nella sezione ADDITIONAL.

> dig example.com @8.8.8.8 (2-2)

6. `;; QUESTION SECTION:`
   - `;example.com. IN A`
     - L'utente ha chiesto un record di tipo `A` per il dominio `example.com`.
7. `;; ANSWER SECTION:`
   - `example.com. 2759 IN A 93.184.216.34`
     - Il server DNS ha restituito un record di tipo `A` per `example.com`, che punta all'indirizzo IP `93.184.216.34`. Il TTL (Time To Live) di questo record è di 2759 secondi.
8. `;; Query time: 48 msec`
   - La query ha impiegato 48 millisecondi per ricevere una risposta.
9. `;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)`
   - Il server DNS che ha risposto alla query è `8.8.8.8`, sulla porta standard `#53`, e ha utilizzato il protocollo UDP.
10. `;; WHEN: Sun Oct 15 21:05:19 UTC 2023`

- L'orario in cui è stata effettuata la query è stato `Sun Oct 15 21:05:19 UTC 2023`.

11. `;; MSG SIZE rcvd: 56`

- La dimensione del messaggio ricevuto è di 56 byte.

> dig example.com @1.1.1.1

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com @1.1.1.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13924
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;example.com.                   IN    A

;; ANSWER SECTION:
example.com.        79432    IN    A    93.184.216.34
;; Query time: 48 msec
;; SERVER: 1.1.1.1#53(1.1.1.1) (UDP)
;; WHEN: Sun Oct 15 21:05:42 UTC 2023
;; MSG SIZE  rcvd: 56
```

> dig example.com @1.1.1.1 (1-2)

1. `; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com @1.1.1.1`
   - L'utente sta utilizzando `DiG` versione `9.18.12` su un sistema Ubuntu `22.04.3`.
   - L'utente ha interrogato il dominio `example.com` utilizzando il server DNS `1.1.1.1` (questo è un server DNS pubblico fornito da Cloudflare).
2. `;; global options: +cmd`
   - L'utente ha impostato alcune opzioni globali per la sua query. `+cmd` mostra il comando utilizzato all'inizio dell'output.
3. `;; Got answer:`
   - Questa linea indica che il server DNS ha restituito una risposta.
4. `;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13924`
   - L'utente ha inviato una operazione di tipo `QUERY` al server DNS.
   - Lo stato della risposta è `NOERROR`, il che indica che la query è stata eseguita con successo.
   - L'ID associato a questa query è `13924`.
5. `;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1`
   - Sono presenti diverse bandiere nella risposta: `qr` (query response), `rd` (recursion desired), `ra` (recursion available), e `ad` (authenticated data).
   - La sezione QUESTION contiene 1 voce.
   - La sezione ANSWER contiene 1 voce.
   - Non ci sono voci nella sezione AUTHORITY.
   - C'è 1 voce nella sezione ADDITIONAL.

> dig example.com @1.1.1.1 (2-2)

6. `;; QUESTION SECTION:`
   - `;example.com. IN A`
     - L'utente ha chiesto un record di tipo `A` per il dominio `example.com`.
7. `;; ANSWER SECTION:`
   - `example.com. 79432 IN A 93.184.216.34`
     - Il server DNS ha restituito un record di tipo `A` per `example.com`, che punta all'indirizzo IP `93.184.216.34`. Il TTL (Time To Live) di questo record è di 79432 secondi.
8. `;; Query time: 48 msec`
   - La query ha impiegato 48 millisecondi per ricevere una risposta.
9. `;; SERVER: 1.1.1.1#53(1.1.1.1) (UDP)`
   - Il server DNS che ha risposto alla query dell'utente è `1.1.1.1`, sulla porta standard `#53`, e ha utilizzato il protocollo UDP.
10. `;; WHEN: Sun Oct 15 21:05:42 UTC 2023`

- L'orario in cui l'utente ha effettuato la query è stato `Sun Oct 15 21:05:42 UTC 2023`.

11. `;; MSG SIZE rcvd: 56`

- La dimensione del messaggio ricevuto dall'utente è di 56 byte.

> dig example.com MX

richiesta di informazioni sui record MX (Mail Exchange) per il dominio example.com

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13662
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
; COOKIE: 28b12f6011c0597106438178652c54390eaef742020ac67d (good)
;; QUESTION SECTION:
;example.com.                    IN    MX

;; ANSWER SECTION:
example.com.            86400   IN    MX    0 .

;; Query time: 160 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Sun Oct 15 21:06:02 UTC 2023
;; MSG SIZE  rcvd: 83
```

> dig example.com MX

1. `<<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com MX`
   - L'utente sta usando `DiG` versione `9.18.12` su un sistema Ubuntu `22.04.3`.
   - L'utente ha interrogato il dominio `example.com` per i record MX (Mail Exchange).
2. `;; global options: +cmd`
   - Questo mostra le opzioni globali impostate dall'utente. `+cmd` mostra il comando utilizzato all'inizio dell'output.
3. `;; Got answer:`
   - Questa linea indica che il server DNS ha restituito una risposta.
4. `;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13662`
   - L'utente ha inviato una operazione di tipo `QUERY` al server DNS.
   - Lo stato della risposta è `NOERROR`, indicando che la query è stata eseguita con successo.
   - L'ID associato a questa query è `13662`.
5. `;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1`
   - Ci sono diverse bandiere nella risposta: `qr` (query response), `rd` (recursion desired) e `ra` (recursion available).
   - La sezione QUESTION contiene 1 voce.
   - La sezione ANSWER contiene 1 voce.
   - Non ci sono voci nella sezione AUTHORITY.
   - C'è 1 voce nella sezione ADDITIONAL.

> dig example.com MX

6. `;; QUESTION SECTION:`
    - `;example.com. IN MX`
        - L'utente ha chiesto un record di tipo `MX` per il dominio `example.com`.
7. `;; ANSWER SECTION:`
    - `example.com. 86400 IN MX 0 .`
        - Il server DNS ha restituito un record MX per `example.com`. Il TTL (Time To Live) di questo record è di 86400 secondi. L'entry ha una priorità di `0`, ma curiosamente punta a un valore di `"."` che non è tipico per un record MX. Questo suggerisce che il dominio non ha un server di posta configurato o che c'è una configurazione speciale.
8. `;; Query time: 160 msec`
    - La query ha impiegato 160 millisecondi per ricevere una risposta.
9. `;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)`
    - Il server DNS che ha risposto alla query dell'utente è `192.168.0.1`, sulla porta standard `#53`, e ha utilizzato il protocollo UDP.
10. `;; WHEN: Sun Oct 15 21:06:02 UTC 2023`

- L'orario in cui l'utente ha effettuato la query è stato `Sun Oct 15 21:06:02 UTC 2023`.

11. `;; MSG SIZE rcvd: 83`

- La dimensione del messaggio ricevuto dall'utente è di 83 byte.

> dig example.com TXT

richiesta di informazioni sui record TXT (Text) per il dominio `example.com`

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com TXT
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43208
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
; COOKIE: 4b8a687ab8441f32c9768a4c652c544d675a04c358f1bd1c (good)
;; QUESTION SECTION:
;example.com.                   IN    TXT

;; ANSWER SECTION:
example.com.      86400   IN    TXT   "v=spf1 -all"
example.com.      86400   IN    TXT   "wgyf8z8cgvm2qmxpnbnldrcltvk4xqfn"

;; Query time: 148 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Sun Oct 15 21:06:21 UTC 2023
;; MSG SIZE  rcvd: 137
```

> dig example.com TXT

1. `; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> example.com TXT`
   - L'utente sta usando `DiG` versione `9.18.12` su un sistema Ubuntu `22.04.3`.
   - L'utente ha interrogato il dominio `example.com` per i record TXT.
2. `;; global options: +cmd`
   - Questo mostra le opzioni globali impostate dall'utente. `+cmd` mostra il comando utilizzato all'inizio dell'output.
3. `;; Got answer:`
   - Questa linea indica che il server DNS ha restituito una risposta.
4. `;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43208`
   - L'utente ha inviato una operazione di tipo `QUERY` al server DNS.
   - Lo stato della risposta è `NOERROR`, indicando che la query è stata eseguita con successo.
   - L'ID associato a questa query è `43208`.
5. `;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1`
   - Ci sono diverse bandiere nella risposta: `qr` (query response), `rd` (recursion desired) e `ra` (recursion available).
   - La sezione QUESTION contiene 1 voce.
   - La sezione ANSWER contiene 2 voci.
   - Non ci sono voci nella sezione AUTHORITY.
   - C'è 1 voce nella sezione ADDITIONAL.

> dig example.com TXT

6. `;; QUESTION SECTION:`
   - `;example.com. IN TXT`
     - L'utente ha chiesto i record di tipo `TXT` per il dominio `example.com`.
7. `;; ANSWER SECTION:`
   - `example.com. 86400 IN TXT "v=spf1 -all"`
     - Il dominio `example.com` ha un record TXT con un TTL (Time To Live) di 86400 secondi. Questo record è un record SPF che specifica che nessun host è autorizzato a inviare posta per `example.com`.
   - `example.com. 86400 IN TXT "wgyf8z8cgvm2qmxpnbnldrcltvk4xqfn"`
     - C'è un altro record TXT associato a `example.com` con un TTL di 86400 secondi. Potrebbe trattarsi di un record associato a qualche tipo di verifica o servizio, come una chiave DKIM o un token di verifica.
8. `;; Query time: 148 msec`
   - La query ha impiegato 148 millisecondi per ricevere una risposta.
9. `;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)`
   - Il server DNS che ha risposto alla query dell'utente è `192.168.0.1`, sulla porta standard `#53`, e ha utilizzato il protocollo UDP.
10. `;; WHEN: Sun Oct 15 21:06:21 UTC 2023`

- L'orario in cui l'utente ha effettuato la query è stato `Sun Oct 15 21:06:21 UTC 2023`.

11. `;; MSG SIZE rcvd: 137`

- La dimensione del messaggio ricevuto dall'utente è di 137 byte.

> nslookup example.com

Server: 192.168.0.1
Address: 192.168.0.1#53

Non-authoritative answer:
Name: example.com
Address: 32.119.103.121
Name: example.com
Address: 11.118.61.115
Name: example.com
Address: 2606:2800:220:1:248:1893:25c8:1946

> nslookup example.com

1. **Server** e **Address**:
   - `Server: 192.168.0.1` e `Address: 192.168.0.1#53`
     - Questo indica che l'utente ha interrogato il server DNS all'indirizzo IP `192.168.0.1` sulla porta standard `#53` per ottenere informazioni sul dominio `example.com`.
2. **Non-authoritative answer**:
   - Questa dichiarazione significa che la risposta fornita dal server DNS non proviene da una fonte autorevole per il dominio specificato (cioè non proviene direttamente dai server DNS che ospitano le informazioni per `example.com`).
3. **Name e Address**:
   - Ci sono tre set di informazioni sotto questa sezione, che mostrano gli indirizzi IP associati al dominio `example.com`.
   - `Name: example.com` e `Address: 32.119.103.121`
     - Questo è uno degli indirizzi IP IPv4 associati al dominio `example.com`.
   - `Name: example.com` e `Address: 11.118.61.115`
     - Questo è un altro indirizzo IP IPv4 associato allo stesso dominio.
   - `Name: example.com` e `Address: 2606:2800:220:1:248:1893:25c8:1946`
     - Questo è un indirizzo IP IPv6 associato a `example.com`.

dig +trace example.com

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> +trace example.com
;; global options: +cmd
.                       478620  IN  NS  l.root-servers.net.
.                       478620  IN  NS  a.root-servers.net.
.                       478620  IN  NS  j.root-servers.net.
.                       478620  IN  NS  c.root-servers.net.
.                       478620  IN  NS  f.root-servers.net.
.                       478620  IN  NS  m.root-servers.net.
.                       478620  IN  NS  d.root-servers.net.
.                       478620  IN  NS  h.root-servers.net.
.                       478620  IN  NS  b.root-servers.net.
.                       478620  IN  NS  k.root-servers.net.
.                       478620  IN  NS  e.root-servers.net.
.                       478620  IN  NS  g.root-servers.net.
.                       478620  IN  NS  i.root-servers.net.
.                       478620  IN  RRSIG   NS 8 0 518400 20231028050000
20231015040000 46780 .
BFOBF8stwa+SKIsVehNL4QZ55/g+HPX+3m7u1/f4kjRAZ4+eXFOP4SM3Sv+SpA0ZV/
MYF44AyZIVTo9+aLht/xdmd1fsbnq8UYygN8qo8VyNgFUAt8cqfU2Eu0oOqBVr4g7clIS
aNn31L3XIWwIrMQPBfci/HaXC16Ww9PbmmCj0RpUCOAGa73HgnwfXShBiEIBLPNw
tmYjhmOWQyztFBLo7AjO0nej40u8d6gt+pPXeu8otBRoiczbBIADXAzwqbMgvQeaCbg
NN6j9B3X9zUA2tugjPE37ke6XBod/tY027neiPSEBdOHP2Jtky+bzs4Mv4uvXCHt669Rj
M HbtlUA==

;; Received 553 bytes from 192.168.0.1#53(192.168.0.1) in 35 ms
com.                    172800  IN  NS  e.gtld-servers.net.
com.                    172800  IN  NS  b.gtld-servers.net.
com.                    172800  IN  NS  m.gtld-servers.net.
com.                    172800  IN  NS  i.gtld-servers.net.
com.                    172800  IN  NS  f.gtld-servers.net.
com.                    172800  IN  NS  a.gtld-servers.net.
com.                    172800  IN  NS  f.gtld-servers.net.
com.                    172800  IN  NS  a.gtld-servers.net.
com.                    172800  IN  NS  g.gtld-servers.net.
com.                    172800  IN  NS  h.gtld-servers.net.
com.                    172800  IN  NS  l.gtld-servers.net.
com.                    172800  IN  NS  k.gtld-servers.net.
com.                    172800  IN  NS  c.gtld-servers.net.
com.                    172800  IN  NS  d.gtld-servers.net.
com.                    86400   IN  DS  30909 8 2
E2D3C916F6DEEAC73294E8268FB5885044A833FC5459588F4A9184CF C41A5766
com.                    86400   IN  RRSIG   DS 8 1 86400 20231028170000
20231015160000
46780 . WxNzxY4M439XQPmn+Vyq93kB2f7K8sqtVdC/mvlV/JUs3/jSOiZVnjkP
mDDiWPSAeCE53Bh0ZsZWu29vJDLqEpafZPwkAp56qTdi5l90j3u0y2ZK
MBqQVE6w43XO6OobnEPyQYJ123a9/Xq8wMynOPApLrHkYoUaIpHF/I/R
0v5P9/NGjS39nLoychRI2PDDbx3GefSID2DDn1oTy8fBFQFnZHi37p9S
X6hF3j15mX7UoU0yJ9fZefH9wbpbhgRkLspl/H6Vyg3GmfaDszAToHN5
7aL3pofGKpsWgPyy3fm9S9s0o8knKnz9gRJjmlWNifV7L5NQPG4rTqVc 8aXg2Q==

;; Received 1171 bytes from 198.41.0.4#53(a.root-servers.net) in 43 ms
example.com.            172800  IN  NS  a.iana-servers.net.
example.com.            172800  IN  NS  b.iana-servers.net.
example.com.            86400   IN  DS  370 13 2
BE74359954660069D5C63D200C39F5603827D7DD02B56F120EE9F3A8 6764247C
```

dig +trace example.com

example.com. 86400 IN RRSIG DS 8 2 86400 20231020062205
20231013051205 4459 com.
hqFVtjyFMdEoTDX+q0LZKNdSwvoz3O8ADSA40sMamegao8VgKUDQC7G2
QTBJaSs5zB+F2d4QSx/y3HbuLJf7wpBtqfNeY3pWJ8WyxVbZUK819SKW
cAe1QjQnZ6fgcnkNRyzppnAbVpAvldT6jaydAfe2s7HDGtXhmKm15P0h
YithuggnKLipz/krx4kE2Ep7YVTku0E42lpNLWy03ARqIw==

;; Received 331 bytes from 192.26.92.30#53(c.gtld-servers.net) in 55 ms

example.com. 86400 IN A 93.184.216.34
example.com. 86400 IN RRSIG A 13 2 86400 20231028144229
20231008002139 37939 example.com.
BfSk6wqpibs9a4AUa8gXHFzhBGiO9f0QuFLVaMYkPu0PgKrDUI94VEfW
lJ5pOMaIgL1RzFwngd51b3FFiLUp3g==
example.com. 86400 IN NS a.iana-servers.net.
example.com. 86400 IN NS b.iana-servers.net.
example.com. 86400 IN RRSIG NS 13 2 86400 20231028140639
20231007162139 37939 example.com.
uwaeQxMhNORkc3v1ymrSAec+d0RwrppXrSvTpkpVo8lnDAwZKfdzGJ+t
3I5dxju+KqTAFAF7Br2tFRRIRJLWIA==
;; Received 318 bytes from 199.43.133.53#53(b.iana-servers.net) in 143 ms

dig +trace example.com (1-2)

Il comando `dig +trace example.com` esegue una query DNS ricorsiva per ottenere informazioni sul dominio `example.com`. In pratica, con l'opzione `+trace`, si comincia dall'alto (root servers) e si scende fino al server DNS autorevole per il dominio specificato. Andiamo a vedere passo passo:

1. **Root Servers**: Il primo gruppo di record NS mostra i root servers. Questi sono i server DNS al top della gerarchia DNS e conoscono quali sono i server TLD (Top Level Domain, come .com, .net, .org) per ciascun dominio di primo livello.

```
.                   478620     IN    NS    l.root-servers.net.
…
.                   478620     IN    NS    i.root-servers.net.
```

La risposta proviene dal server DNS locale (192.168.0.1).
TLD Servers: Il dominio `example.com` ha un TLD di `.com`, quindi il prossimo passo è interrogare i server `.com`. La risposta mostra i server autorizzati per il dominio `.com`.

```
com.                172800     IN    NS    e.gtld-servers.net.
…
com.                172800     IN    NS    d.gtld-servers.net.
```

Questa risposta proviene dal root server `a.root-servers.net`.

dig +trace example.com (2-2)

Domain's Authoritative Name Servers: Una volta ottenuto l'elenco dei server TLD, viene interrogato uno di questi per ottenere i server DNS autorizzati per `example.com`.

```
example.com.            172800      IN    NS    a.iana-servers.net.
example.com.            172800      IN    NS    b.iana-servers.net.
```

La risposta proviene dal TLD server `c.gtld-servers.net`.

Domain's Records: Infine, uno dei server DNS autorizzati per `example.com` (in questo caso `b.iana-servers.net`) fornisce la risposta definitiva.

```
example.com.            86400 IN    A     93.184.216.34
```

1. Questo è l'indirizzo IP associato a `example.com`. Sono presenti anche altri record, come RRSIG (che è legato alla sicurezza DNS) e ulteriori record NS che confermano i server autorizzati per `example.com`.

In breve, con `dig +trace`, si esegue una serie di query che replicano il processo che un resolver DNS ricorsivo (come quelli di ISP o Google Public DNS) eseguirebbe per risolvere un dominio fino al suo indirizzo IP finale. Questo tipo di query può essere utile per il debugging o per comprendere come funziona la risoluzione dei nomi nell'infrastruttura DNS.

Configurazione di un Server Web

1. Creazione del Dockerfile:

Prima di tutto, creiamo un `Dockerfile`:

```
# Utilizza l'immagine ufficiale di Apache come base
FROM httpd
# Copia il tuo file index.html nella directory del server web Apache nel container
COPY index.html /usr/local/apache2/htdocs/

# Opzionalmente, se vuoi avere curl o wget nel tuo container per scopi dimostrativi:
RUN apt update && apt install -y curl wget
```

Assicurati che il file `index.html` sia nella stessa directory del tuo `Dockerfile`.

Puoi creare un semplice `index.html` per scopi di test con il seguente contenuto:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <title>Benvenuto!</title>
</head>
<body>
    <h1>Ciao Mondo!</h1>
    <p>Questa è una semplice pagina di demo.</p>
</body>
</html>
```

Configurazione di un Server Web

1. Costruzione dell'immagine:

Nella directory che contiene il `Dockerfile` e `index.html`, esegui il seguente comando:

```
docker build -t my-custom-apache .
```

Questo comando costruirà un'immagine Docker chiamata `my-custom-apache` basata sul tuo `Dockerfile`.

2. Esecuzione del container:

Ora puoi eseguire il container basato sulla tua immagine personalizzata:

```
docker run -d --name my-custom-apache-container -p 8080:80 my-custom-apache
```

Il tuo server web Apache dovrebbe ora essere accessibile su `http://localhost:8080`.

3. Test con `curl` o `wget`:

Dalla tua macchina host:

```
curl http://localhost:8080
```

Oppure, per eseguire un test dall'interno del container:

```
docker exec my-custom-apache-container curl http://localhost
```

Creazione rete e nodi Client Server (SSH vs Telnet)

1. Creazione di una rete Docker:

Per isolare i nostri container, creeremo una rete Docker dedicata.

```
docker network create demo-network
```

2. Container Server:

Dockerfile per il server:

```
FROM ubuntu:latest
# Installazione di Telnet e SSH
RUN apt update && apt install -y xinetd telnetd openssh-server && apt clean
# Abilitazione del servizio SSH
RUN service ssh start
# Esegui un comando per mantenere il container in esecuzione
CMD ["tail", "-f", "/dev/null"]
```

Salva questo Dockerfile come `Dockerfile.server` e costruisci l'immagine:

```
docker build -t demo-server -f Dockerfile.server .
```

Avvia il container:

```
docker run -d --name demo-server-container --network demo-network demo-server
```

Creazione rete e nodi Client Server (SSH vs Telnet)

3. Container Client:

Dockerfile per il client:

```
FROM ubuntu:latest
# Installazione di SSH client, Telnet client e Wireshark
RUN apt update && apt install -y openssh-client telnet wireshark && apt clean
# Configura Wireshark
RUN echo "wireshark-common wireshark-common/install-setuid boolean true" |
debconf-set-selections \
    && DEBIAN_FRONTEND=noninteractive apt install -y wireshark
# Esegui un comando per mantenere il container in esecuzione
CMD ["tail", "-f", "/dev/null"]
```

Salva questo Dockerfile come `Dockerfile.client` e costruisci l'immagine:

```
docker build -t demo-client -f Dockerfile.client .
```

Avvia il container:

```
docker run -d --name demo-client-container --network demo-network demo-client
```

Creazione rete e nodi Client Server (SSH vs Telnet)

4. Cattura del traffico con `tcpdump`

Prima di tutto, avviamo `tcpdump` sui nostri container per catturare il traffico mentre ci connettiamo sia tramite telnet che SSH. La cattura con `tcpdump` ci fornirà i file pcap che possiamo esaminare successivamente.

All'interno del container "client":

```
docker exec -it demo-client-container bash
tcpdump -i any -w /tmp/telnet.pcap 'port 23'
```

In un altro terminale, sempre all'interno del container "client":

```
docker exec -it demo-client-container bash
tcpdump -i any -w /tmp/ssh.pcap 'port 22'
```

Lascia `tcpdump` in esecuzione in entrambi i terminali.

Creazione rete e nodi Client Server (SSH vs Telnet)

5. Connessione via Telnet e SSH

5.1. In un nuovo terminale, connettiti al server telnet:

```
docker exec -it demo-client-container telnet [SERVER_IP]
```

Effettua alcune operazioni, come ad esempio inserire una password.

5.2. In un altro nuovo terminale, connettiti via SSH:

```
docker exec -it demo-client-container ssh root@[SERVER_IP]
```

Inserisci la password e esegui qualche comando.

6. Analisi dei file pcap

6.1. Termina le sessioni `tcpdump` (puoi farlo con CTRL+C).

6.2. Ora, useremo `tcpdump` per leggere i file pcap direttamente dal terminale.

Per il traffico Telnet:

```
tcpdump -nn -r /tmp/telnet.pcap -A
```

Osserverai che le informazioni sono leggibili in chiaro, incluso qualsiasi testo o password che hai inserito.

Per il traffico SSH:

```
tcpdump -nn -r /tmp/ssh.pcap -A
```

Noterai che il traffico è cifrato e non puoi leggere chiaramente le informazioni, a differenza del traffico telnet.

Creazione rete e nodi Client Server (SSH vs Telnet)

Conclusione della Demo:

- **Telnet**: Trasmette dati in chiaro. Chiunque possa intercettare il traffico può vedere le informazioni sensibili (come le password).
- **SSH**: Cifra la tua sessione, rendendo molto difficile per gli attaccanti intercettare e leggere i dati.

Creazione e Configurazione di un Server DNS

1. **Guida alla Creazione e Configurazione di un Server DNS con Docker**
   Obiettivo: Questa demo illustra come configurare e avviare un server DNS personalizzato utilizzando Docker. Sarà un esercizio pratico per comprendere come funzionano i server DNS e come essi rispondono alle richieste.
   Preparazione:
   1. **Creazione di una Dockerfile per il server DNS**
      Nella directory di lavoro, creare un file chiamato `Dockerfile` con il seguente contenuto

```
FROM ubuntu:latest
RUN apt-get update && apt-get install -y bind9 bind9utils bind9-doc
EXPOSE 53/udp 53/tcp
ENTRYPOINT ["/usr/sbin/named", "-g", "-c", "/etc/bind/named.conf", "-u", "bind"]
```

2. Questa Dockerfile indica a Docker di:
   1. Utilizzare l'immagine `ubuntu:latest` come base.
   2. Installare il pacchetto `bind9` per il server DNS.
   3. Esporre le porte standard DNS (53) sia in TCP che in UDP.
   4. Avviare il server DNS all'avvio del container.

Creazione e Configurazione di un Server DNS

3. **Costruzione dell'immagine Docker**
   Dalla directory che contiene la Dockerfile, eseguire il seguente comando

```
docker build -t customdns .
```

4. **Avvio del container DNS**

```
docker run -d --name demo-dns-server –net=host customdns
```

5. **Interrogazione del server DNS personalizzato**
   Eseguire il seguente comando per interrogare il server DNS:
   bash

```
dig @127.0.0.1 example.com
```

Configurazione di una Zona e di un Record DNS

1. Preparazione dei file di configurazione sul tuo host

Nella stessa directory dove andremo a creare il `Dockerfile`, crea i seguenti file:

- `named.conf.local`:

```
zone "demo.local" {
    type master;
    file "/etc/bind/db.demo.local";
};
```

- `db.demo.local`:

```
;
; BIND data file for demo.local
;
$TTL    604800
@       IN      SOA     demo.local. root.demo.local. (
                        20231016         ; Serial
                             604800         ; Refresh
                              86400         ; Retry
                            2419200         ; Expire
                             604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.demo.local.
ns      IN      A       192.0.2.2
server  IN      A       192.0.2.3
```

Configurazione di una Zona e di un Record DNS

1. `$TTL 604800`: Questa è la Time-To-Live (TTL) di default per i record nel file. La TTL è il tempo (in secondi) che un resolver DNS dovrebbe mettere in cache la risposta. 604800 secondi equivalgono a una settimana.
2. `@ IN SOA demo.local. root.demo.local. ( ... )`: Questo è il record SOA (Start Of Authority). Esso specifica le impostazioni autoritative per questa zona.
    - `demo.local.`: è il nome del server che è l'autorità per questa zona.
    - `root.demo.local.`: è l'indirizzo e-mail dell'amministratore responsabile di questa zona (nota che il simbolo `@` dell'e-mail è sostituito da un punto).
    - `20231016`: è il numero seriale della zona. Di solito viene aumentato ogni volta che fai una modifica al file di zona.
    - `604800`: Refresh - il tempo che un server DNS secondario aspetterà prima di controllare se ci sono modifiche.
    - `86400`: Retry - il tempo che un server DNS secondario aspetterà per riprovare dopo un tentativo fallito di raggiungere il server DNS primario.
    - `2419200`: Expire - il tempo massimo che un server DNS secondario aspetterà prima di considerare i suoi dati come obsoleti se non riesce a contattare il server DNS primario.
    - `604800`: Negative Cache TTL - il tempo durante il quale una ricerca fallita dovrebbe essere memorizzata nella cache.
3. `@ IN NS ns.demo.local.`: Questo specifica che `ns.demo.local` è un server dei nomi (NS - Name Server) per il dominio `demo.local`.
4. `ns IN A 192.0.2.2`: Questo è un record A che risolve `ns.demo.local` all'indirizzo IP `192.0.2.2`.
5. `server IN A 192.0.2.3`: Questo è un record A che risolve `server.demo.local` all'indirizzo IP `192.0.2.3`.

Configurazione di una Zona e di un Record DNS

2. Dockerfile
Creiamo un `Dockerfile` per preparare l'immagine con BIND:

```
FROM ubuntu:20.04
RUN apt-get update && apt-get install -y bind9 bind9utils bind9-doc nano net-tools
dnsutils
RUN mkdir -p /run/named && chown bind:bind /run/named
RUN chown -R bind:bind /etc/bind && chmod 777 /etc/bind/rndc.key
COPY named.conf.local /etc/bind/named.conf.local
COPY db.demo.local /etc/bind/db.demo.local
COPY named.conf.options /etc/bind/named.conf.options
RUN chown -R bind:bind /etc/bind
CMD ["/usr/sbin/named", "-g", "-c", "/etc/bind/named.conf"]
```

3. Build e run del container
Con tutti i file preparati (Dockerfile, named.conf.local e db.demo.local), procedi con la build e la run:

```
docker build -t custom-bind-image .
docker run --name demo-dns-server –net==host custom-bind-image
```

Configurazione di una Zona e di un Record DNS

4. Test del server DNS

Dopo aver avviato il container, puoi testare la tua configurazione DNS con:

```
dig @127.0.0.1 server.demo.local
```

```
<<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> @127.0.0.1 -p 54 server.demo.local
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49320
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d8f6d7ca0570455401000000652cfc15087abb81a6f51476 (good)
;; QUESTION SECTION:
;server.demo.local.              IN   A

;; ANSWER SECTION:
server.demo.local.   604800   IN   A   192.0.2.3

;; Query time: 0 msec
;; SERVER: 127.0.0.1#54(127.0.0.1) (UDP)
;; WHEN: Mon Oct 16 11:02:13 CEST 2023
;; MSG SIZE  rcvd: 90
```

La tua query `dig` ha ricevuto una risposta positiva. La sezione `ANSWER` mostra che `server.demo.local` risolve all'indirizzo IP `192.0.2.3`.

Nota: il warning riguardo `.local` essendo riservato per Multicast DNS (mDNS) è normale quando si utilizza il dominio `.local`. mDNS è utilizzato in reti locali per la risoluzione di nomi senza necessità di un DNS server. In un ambiente di produzione, potresti voler evitare di utilizzare `.local` per evitare possibili conflitti.

Server SMTP con Docker e Postfix

1. Creazione del Dockerfile

Crea un file chiamato `Dockerfile` con il seguente contenuto:

```
FROM ubuntu:20.04
# Evita l'interazione durante l'installazione dei pacchetti
ENV DEBIAN_FRONTEND=noninteractive
# Installa Postfix e telnet (utile per i test)
RUN apt-get update && apt-get install -y \
    postfix \
    telnet \
    rsyslog

# Configurazione di Postfix per il relay senza restrizioni
RUN echo "smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated
defer_unauth_destination" >> /etc/postfix/main.cf && \
    echo "mynetworks = 0.0.0.0/0" >> /etc/postfix/main.cf
# Avvia i servizi
CMD service rsyslog start && service postfix start && tail -F /var/log/mail.log
```

Server SMTP con Docker e Postfix

2. Costruzione dell'immagine Docker

Nel terminale, naviga alla directory dove si trova il Dockerfile e costruisci l'immagine:

```
docker build -t my-postfix .
```

3. Avvio del container

Avvia un container utilizzando l'immagine appena creata:

```
docker run -d -p 25:25 my-postfix
```

4. Test del server SMTP

Utilizza `telnet` per testare il server SMTP:

```
telnet localhost 25
```

Dopo la connessione:

```
EHLO smtp.demo.local
MAIL FROM:<test@demo.local>
RCPT TO:<user@demo.local>
DATA
Subject: Test email
Questo è un test.
.
QUIT
```

Server SMTP con Docker e Postfix

telnet localhost 25

```
Trying 127.0.0.1…
Connected to localhost.
Escape character is '^]'.
220 localhost ESMTP Postfix (Ubuntu)
EHLO smtp.demo.local
250-localhost
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING
MAIL FROM:<test@demo.local>
250 2.1.0 Ok
RCPT TO:<user@demo.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Test email
Questo è un test.
.
250 2.0.0 Ok: queued as A80971444F98
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Server SMTP con Docker e Postfix

1. **telnet localhost 25**
   - Questo è il comando per connettersi al servizio SMTP sul localhost alla porta standard SMTP, che è la 25.
2. **Trying 127.0.0.1... Connected to localhost. Escape character is '^]'.**
   - Il client `telnet` sta cercando di connettersi all'indirizzo IP 127.0.0.1 (localhost). Una volta stabilita la connessione, ti informa che il carattere di escape (per uscire da telnet) è '^]'.
3. **220 localhost ESMTP Postfix (Ubuntu)**
   - Il server SMTP (in questo caso Postfix) risponde con un messaggio di saluto. `220` è un codice di risposta SMTP che indica che il servizio è pronto. ESMTP significa "Extended Simple Mail Transfer Protocol".
4. **EHLO smtp.demo.local**
   - Questo comando è inviato dal client al server per identificarsi. `EHLO` è una versione estesa del comando `HELO` e viene utilizzato per iniziare la conversazione con il server SMTP.
5. Dopo l'`EHLO`, il server risponde con vari codici `250`, che indicano una risposta positiva. Ogni riga fornisce una caratteristica o un comando che il server supporta:
   - **250-localhost**: Nome del server.
   - **250-PIPELINING**: Il server supporta l'invio di più comandi SMTP senza dover attendere per ciascuna risposta.
   - **250-SIZE 10240000**: Il server accetta messaggi fino a 10MB.
   - **250-VRFY**: Il server supporta il comando VRFY per verificare gli indirizzi email.
   - **250-ETRN**: Un comando utilizzato per la coda di messaggi in attesa.

Server SMTP con Docker e Postfix

- **250-STARTTLS**: Il server supporta connessioni criptate utilizzando TLS.
- **250-ENHANCEDSTATUSCODES**: Utilizza codici di stato estesi.
- **250-8BITMIME**: Supporta email con contenuti a 8 bit.
- **250-DSN**: Delivery Status Notifications supportate.
- **250-SMTPUTF8**: Supporta l'UTF-8 nell'SMTP.
- **250 CHUNKING**: Supporta il "chunking", una caratteristica dell'ESMTP.

2. **MAIL FROM:test@demo.local**
   - Questo comando indica al server l'indirizzo del mittente dell'email.
3. **RCPT TO:user@demo.local**
   - Questo comando indica al server l'indirizzo del destinatario dell'email.
4. **DATA**
   - Con questo comando, indichi che stai per iniziare a inviare il corpo dell'email.
5. **Subject: Test email**: Questo è un test.
   - Il corpo del messaggio. Si conclude con un punto su una nuova riga (indicato da `.`).
6. **QUIT**
   - Questo comando chiude la connessione SMTP.

Ogni comando inviato dal client viene risposto dal server con un codice di stato (come `250`, `354`, `221`, ecc.) che indica se il comando è stato eseguito correttamente o se ci sono stati problemi.
Nell'esempio,l'email è stata messa in coda per la consegna (come indicato dal codice `250 2.0.0 Ok: queued as A80971444F98`).