Capitolo 1

Installazione

In questo capitolo

- **Il vostro server web**
- **Per iniziare**
- **Iniziare a usare Docker**
- **Ora è tutto pronto**

In questo libro, vi guiderò nei vostri primi passi oltre lo statico mondo della creazione di pagine web utilizzando unicamente tecnologie lato-client come HTML, CSS e JavaScript. Insieme, esploreremo il mondo della creazione di siti web e scopriremo la grande quantità di possibilità, strumenti e concetti dinamici disponibili. Qualsiasi cosa abbiate in mente, puntate in alto.

Ok… non così in alto. In fin dei conti il libro è qui in basso. Ma… avete capito che cosa intendevo dire.

Prima di creare il vostro primo sito web dinamico, dovrete approntare tutti gli strumenti necessari per svolgere il compito. Come nella preparazione di un dolce, dovrete preparare gli ingredienti prima di iniziare a seguire la ricetta. In questo capitolo, vi mostrerò come scaricare e configurare i pacchetti software richiesti.

Se siete soliti creare siti web con HTML, CSS e magari usare anche un po' di JavaScript, probabilmente sapete che cosa significhi eseguire l'upload dei file che costituiscono il sito in una determinata posizione. Può trattarsi di un servizio di hosting web a pagamento, di uno spazio web predisposto dal vostro provider Internet o magari di un server web predisposto dal dipartimento IT dell'azienda per la quale lavorate. In ogni caso, dopo aver copiato i file in una di queste destinazioni, un software chiamato *server web* potrà trovarli e *servire* copie di questi file a chiunque li abbia richiesti tramite un browser web come Microsoft Edge, Internet Explorer, Google Chrome, Apple Safari o Mozilla Firefox. Fra i più

noti server web di cui potreste aver sentito parlare vi sono Apache HTTP Server (Apache), NGINX e *Internet Information Services*.
PHP è un linguaggio script lato-server ed è completamente gratuito, al download e nell'uso. Potete immaginarlo come un plugin per il vostro server web, che gli consente di fare qualcosa di più che inviare semplicemente delle copie dei file richiesti dai browser web. Con PHP installato, il vostro server web sarà in grado di eseguire piccoli programmi (chiamati script PHP) che possono svolgere compiti come prelevare informazioni aggiornate da un database e usarle per generare al volo una pagina web, prima di inviarla al browser che l'ha richiesta. Molto di questo libro riguarderà la realizzazione di script PHP per fare esattamente questo.
Perché i vostri script PHP possano trarre informazioni da un database, dovete naturalmente partire da un database. Ecco dove interviene MySQL. MySQL è un RDBMS (*Relational DataBase Management System*), un sistema per la gestione di database relazionali. Vedremo più avanti il ruolo che gioca e come funziona, ma in poche parole, si tratta di un software in grado di organizzare e gestire in modo efficiente più tipi di informazioni, tenendo conto delle relazioni esistenti fra di esse. MySQL semplifica anche l'accesso a tali informazioni tramite linguaggi script lato-server come PHP. E, sempre come PHP, è completamente gratuito per la maggior parte degli utilizzi.
L'obiettivo di questo primo capitolo è quello di aiutarvi a configurare un server web dotato di PHP e MySQL. Vi fornirò istruzioni dettagliate utilizzabili con le versioni recenti di Windows, macOS e Linux, qualunque sia il computer che state usando, troverete istruzioni appropriate.

Il vostro server web

Molto probabilmente, il server web del vostro attuale web host ha già installato PHP e MySQL, uno dei motivi per cui PHP e MySQL sono così popolari. Se il vostro web host è già predisposto, la buona notizia è che potrete pubblicare il vostro primo sito web senza dover cercare un web host che supporti tali tecnologie.
Nello sviluppo di siti web statici, vi basta caricare i vostri file HTML direttamente dal vostro disco rigido al browser, per vedere il risultato prodotto. Non serve alcun server web per svolgere questa operazione, il che va bene, perché i browser web sono in grado di leggere e interpretare il codice HTML senza alcun aiuto.
Al contrario, quando si tratta di siti web dinamici realizzati con PHP e MySQL, il browser web ha bisogno di un aiuto. I browser web non sono in grado di comprendere gli script PHP, i quali contengono istruzioni che un server web adeguatamente predisposto può eseguire per *generare* il codice HTML che i browser saranno poi in grado di comprendere.
Anche se avete un web host che supporta PHP, avete comunque la necessità di eseguire script PHP senza dover contare su un server esterno. Per farlo, dovrete configurare un vostro server web. La parola "server" può indurvi a immaginare una grande sala-computer, con aria condizionata e zeppa di grandi "armadi" di computer in rack. Ma non preoccupatevi, non dovrete acquistare alcun nuovo hardware. Il vostro attuale computer laptop o desktop andrà più che bene.
Per eseguire script PHP sul web host, dovrete scriverli in un editor, aprire il client FTP o SSH e farne l'upload sul server. Solo allora potrete vedere i risultati nel browser,

raggiungendo l'URI del file che avete creato. Se avete commesso un errore, dovrete modificare il codice, riaprire il client FTP, caricare nuovamente il file sul server e poi ricaricare la pagina. Questo è quanto meno noioso, ma è anche uno spreco di tempo prezioso, che piuttosto sarebbe meglio dedicare alla programmazione. Utilizzando un server sul vostro PC, potrete salvare un file nel vostro editor e osservare le modifiche nel browser semplicemente aggiornando la pagina senza alcun upload di file. Questo rappresenta un grande vantaggio in termini di tempo e uno dei più grandi (non certo l'unico) del fatto di utilizzare un server sul proprio PC, anche se avete a disposizione un web host perfettamente funzionante.
Come predisporre sul PC un server web perfettamente operativo? Vi sono quattro modi per farlo, ognuno dei quali presenta vantaggi e svantaggi.

Configurazione del server 1: installazione manuale di tutti i componenti software

Apache è un server web e come la maggior parte dei software è dotato di un installer che consente di configurarlo con facilità sul PC. Senza troppa fatica, potete configurarlo in modo che *serva* pagine web. Tuttavia, vi sono centinaia di opzioni di configurazione e se non sapete esattamente che cosa fare, può essere davvero problematico metterlo in condizioni di produrre siti web con PHP.
Dato che il nostro scopo è quello di eseguire script PHP, un server web, da solo, non è sufficiente. In un'installazione manuale, dovrete anche installare PHP – il quale non ha un installer – e configurarlo. Come per Apache, vi sono molte opzioni e i valori di default sono predisposti in modo da farlo funzionare come un sito web attivo. In fase di sviluppo del codice, tale configurazione è inadatta, in quanto non mostra gli errori. Se avete commesso un errore, otterrete una pagina vuota, senza alcuna indicazione del problema. Basta un carattere fuori posto – magari una parentesi o un punto e virgola – e otterrete una pagina vuota. Punto e basta. Per risolvere la situazione, dovete configurare manualmente l'installazione di PHP e intervenire sulle impostazioni, in modo che il server mostri i messaggi d'errore, e attivare altri strumenti che agevolano le attività di sviluppo. Dovrete anche configurare Apache per comunicare con PHP, in modo che quando qualcuno si connette al server e richiede un file con l'estensione `.php`, il file venga prima inviato a PHP per l'elaborazione.
Per seguire questo libro avrete bisogno anche di MySQL, che richiede, anch'esso, un'installazione e una configurazione manuali.
Apache, MySQL e PHP hanno, ognuno, decine di opzioni di configurazione e, a meno che sappiate esattamente che cosa state facendo, il compito può essere davvero arduo. Se siete davvero grandi esperti, avrete bisogno di almeno un'ora per mettere tutto in funzione!
L'installazione manuale richiede molte competenze e non rientra negli scopi di questo libro. La capacità di configurare un server è certamente una competenza utile, ma non insegna a programmare in PHP, che credo sia quello che intendete fare, visto che avete in mano questo libro.
Dunque si tratta di un'opzione inadatta ai pavidi e perfino i professionisti possono benissimo dimenticarsi qualche impostazione importante. Fortunatamente, possiamo anche evitare di preoccuparci troppo della configurazione dei singoli software.

Configurazione del server 2: installazioni preconfezionate

I problemi insiti nelle installazioni manuali sono stati riconosciuti nel corso degli anni da parte di vari gruppi di sviluppatori e per risolverli hanno realizzato delle installazioni preconfezionate – un unico installer che installa PHP, Apache, MySQL e altri software necessari, tutti già preconfigurati con impostazioni appropriate per gli sviluppatori, come siete voi. L'esempio più comune di questo tipo di pacchetto è XAMPP: X (ogni sistema operativo), Apache, MySQL (o, per la precisione, MariaDB, una "fork" di MySQL con una migliore licenza d'uso), PHP e Perl. Fra le alternative vi sono WAMP (Windows, Apache, MySQL, PHP), LAMP (Linux, Apache, MySQL, PHP) e MAMP (macOS, Apache, MySQL, PHP).

Questa è ovviamente un'installazione molto più semplice di quella manuale di ogni singolo software e soprattutto non obbliga a imparare le tecniche di configurazione di un server. È rapida e semplice, e quindi migliore rispetto a un'installazione manuale, sebbene rimangano ancora alcuni problemi, che possono emergere utilizzando il seguente metodo.

- Probabilmente il vostro web host è basato su Linux, ma magari il vostro PC no. Sebbene Apache, MySQL e PHP operino in Windows, Linux o macOS, vi sono alcune grandi differenze nel suo funzionamento fra i vari sistemi operativi. In Windows, i nomi di file non sono *case-sensitive*, non distinguono fra maiuscole e minuscole, e quindi il nome `FILE.PHP` equivale a `file.php` o `fIlE.pHp`. Sul vostro web host, le cose non stanno affatto così! Questo causa frequenti problemi nel caso in cui uno script perfettamente funzionante sul server di sviluppo Windows non funziona più dopo l'upload, semplicemente perché i file sono scritti diversamente.
- Apache e MySQL sono *server* e funzionano in background. Anche quando non state sviluppando software, loro saranno lì, in esecuzione sul computer, usando la loro quota di RAM e potenza di elaborazione.
- I software preconfezionati sono sempre leggermente datati. Sebbene le correzioni relative alla sicurezza non siano una priorità per un computer usato solo per lo sviluppo (non consentite mai ad altri di accedervi via Web), è sempre utile per gli sviluppatori impiegare le versioni più recenti di ogni software, per risolvere i problemi che verranno affrontati quando il software del web host verrà aggiornato. Se il vostro web host usa una versione di PHP più recente di quella del server di sviluppo, questo può provocare problemi a causa di funzionalità modificate o rimosse. Infine, gli sviluppatori amano provare le nuove funzionalità nel momento in cui vengono rilasciate. E se non impiegherete proprio le ultime versioni questo non potrete farlo.

Sebbene le installazioni preconfezionate siano molto più agevoli di quelle manuali, questi problemi le rendono una soluzione non ideale. Fortunatamente, esiste un approccio ancora migliore.

Configurazione del server 3: server virtuali

Il terzo metodo per approntare un server consiste nell'impiegare un server virtuale. Un *server virtuale* si comporta come un server web su un altro computer. Questo computer può impiegare qualsiasi sistema operativo e potete connettervi ad esso tramite il vostro PC come se fosse all'altro capo del mondo.

L'uso del software di virtualizzazione, come VirtualBox e gli strumenti offerti da VMWare, è molto diffuso. Come sviluppatori web, dovreste conoscere strumenti come modern.ie (`http://modern.ie`), un utile servizio fornito da Microsoft che consente di scaricare delle *macchine virtuali* sulle quali sono in esecuzione diverse versioni di Windows, Microsoft Edge e Internet Explorer. Se volete vedere l'aspetto del vostro sito web in Internet Explorer 8 in Windows 7, potete scaricare la relativa macchina virtuale e lanciarla in una finestra sul vostro desktop Windows 10/macOS/Linux senza dovervi installare ed eseguire Windows 7 con Internet Explorer 8.

Figura 1.1 Windows 10 in Arch Linux.

Un software come VirtualBox consente di eseguire un sistema operativo all'interno di un altro sistema operativo. Per provare Internet Explorer 8, potete eseguire Windows 7 in una macchina virtuale. Tuttavia, per quanto riguarda l'esecuzione di script PHP, questo ci consente di fare qualcosa di ancora più interessante: possiamo eseguire un server web Linux con PHP, Apache e MySQL, anche lavorando su un PC Windows o macOS. Questo meccanismo può essere utilizzato per consentirvi di eseguire esattamente le stesse versioni di PHP, MySQL e Apache impiegate dal vostro web host, con esattamente lo stesso sistema operativo, aggirando così tutti i problemi che possono sorgere a causa di differenze di versione o di sistema operativo impiegato.

Uno dei più grandi vantaggi è che potete scaricare delle macchine virtuali preconfigurate, come la macchina virtuale Windows XP con Internet Explorer 8 fornita da Microsoft o una macchina virtuale dotata di PHP, Apache e MySQL già installati e configurati in modo da operare insieme. È un po' come avere un pacchetto preconfigurato, ma opera in Linux come se fosse il vero server web della rete.

Il difetto di questo sistema è che dovrete scaricare un intero sistema operativo per poter provare il codice. Si tratta di un bel download. Ma significa anche che i package saranno "bloccati" a quanto viene fornito dalla macchina virtuale che avete scaricato. Il passaggio da PHP7 a PHP8 vi obbligherà al download di una nuova copia, intera, del sistema operativo.

Configurazione del server 4: Docker

Docker capovolge l'idea della virtualizzazione. Ogni programma (o "servizio", nella terminologia di Docker) viene eseguito nel suo ambiente isolato, chiamato *container*.
Docker consente allo sviluppatore di software di produrre un file di configurazione che descrive tutti i programmi necessari per eseguire un'applicazione, per esempio PHP, MySQL, Apache e così via.
Questo file di configurazione viene poi trattato come "un'applicazione": quando eseguite l'applicazione, Docker scarica e configura automaticamente tutto il software elencato nel file di configurazione.
Il sovraccarico dovuto a questo meccanismo è molto più contenuto di quanto si possa pensare, ma con numerosi vantaggi.

- La configurazione del server diventa parte dell'applicazione. Quando manderete online il sito web, potete caricarvi tutti i file di configurazione. Nel modello tradizionale, dovreste impostare manualmente il server web e configurarlo.
- Sul computer di sviluppo possono essere eseguiti contemporaneamente più siti web, con configurazioni differenti e perfino software per server differenti (per esempio, un sito web che utilizza Apache, un altro che utilizza NGINX). Senza Docker, generalmente viene intallata un'unica versione PHP con un'unica configurazione per ogni sito web in esecuzione sulla macchina.
- Potete facilmente cambiare un pezzo di software. Se desiderate modificare il sito web da PHP 7 a PHP 8, è una modifica che riguarda un solo file e potete applicarla su un unico sito web, anziché forzare un aggiornamento di tutti i vostri siti web, contemporaneamente.

Docker è attualmente l'opzione migliore per configurare un ambiente di sviluppo PHP. Per saperne di più sulla configurazione di un ambiente di sviluppo utilizzando Docker, consultate il mio articolo *Setting Up a Modern PHP Development Environment with Docker* (`https://www.sitepoint.com/docker-php-development-environment`).

Per iniziare

Prima di iniziare a scrivere del codice PHP e a sviluppare il sito web, configureremo l'ambiente di sviluppo utilizzando Docker. Presento tutta la configurazione, ma prima di iniziare dovrete installare Docker.

Installazione in Windows

Innanzitutto, scaricate e installate l'ultima versione di Docker per Windows dal sito di Docker (`https://hub.docker.com/editions/community/docker-ce-desktop-windows`).
Dopo aver installato Docker (e riavviato il computer, se necessario), create una cartella in cui intendete archiviare il sito web. Può essere ovunque: la cartella Documenti, il desktop, un disco rigido esterno e così via, ma assicuratevi di ricordarvi dove si trova, poiché dovrete tornarvi spesso.
Dovrete anche sapere come aprire un terminale. Windows 10 facilita molto le cose. Con la cartella aperta in *Esplora file*, scegliete il menu *File* in alto a sinistra nella finestra, quindi fate clic su *Apri Windows PowerShell*. Assicuratevi di farlo dal menu *File* e non dal

menu *Start*, poiché in tal modo il prompt dei comandi si aprirà sulla cartella corrente. Dopo aver aperto PowerShell, verificate che il percorso visualizzato sia proprio quello della cartella scelta, per esempio `C:\Users\Tom\Documents\My Website`. Quindi procedete la lettura al paragrafo "Introduzione a Docker", di seguito.
Al momento della scrittura di questa pagina, nella build di Windows 11, PowerShell è stato chiamato Terminale e vi si accede in un altro modo.

1. Attivate l'opzione aprendo manualmente l'applicazione Terminale dal menu di avvio. Potete semplicemente aprire e chiudere il programma. Inspiegabilmente, Microsoft ha progettato Windows 11 in modo tale che l'apertura di un programma attivi un'impostazione nascosta in un altro, e saltando questo passaggio, non trovereste la voce di menu al Passo 3.
2. Usate Esplora file per raggiungere la cartella desiderata.
3. Fate clic destro nel pannello principale (dovrebbe dire La cartella è vuota") e selezionate *Apri nel Terminale*.

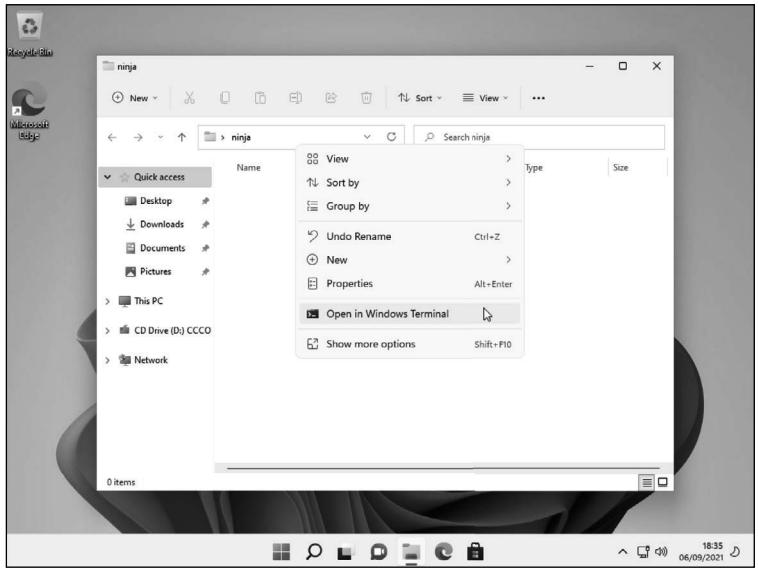

Figura 1.2 L'opzione Apri nel Terminale in Windows 11.

Avvertenza di Windows 11

Queste istruzioni potrebbero essere cambiate nelle nuove versioni di Windows 11, successive alla pubblicazione di questo libro.

Installazione su macOS

Innanzitutto, scaricate e installate Docker per macOS dal sito di Docker (`https://hub.docker.com/editions/community/docker-ce-desktop-mac`).

Dopo aver installato Docker (e riavviato il computer, se richiesto), create una directory sul computer in cui desiderate porre il vostro sito web. Può essere ovunque: la directory Documenti, il desktop, un disco rigido esterno e così via, ma assicuratevi di ricordarvi dove si trova, poiché dovrete tornarvi spesso.
Se avete familiarità con la navigazione sul vostro computer dal terminale, potete saltare questo passaggio, ma per comodità vi consiglio di abilitare una funzione che consente di aprire un terminale nella directory corrente. Potete procedere utilizzando i seguenti passaggi.

1. Aprite *Preferenze di Sistema*.
2. Fate clic su *Tastiera*.
3. Selezionate la scheda *Abbreviazioni da tastiera*.
4. Fate clic, a sinistra, sul pulsante *Servizi*.
5. Selezionate, a destra, l'opzione *Nuovo terminale nella cartella*.

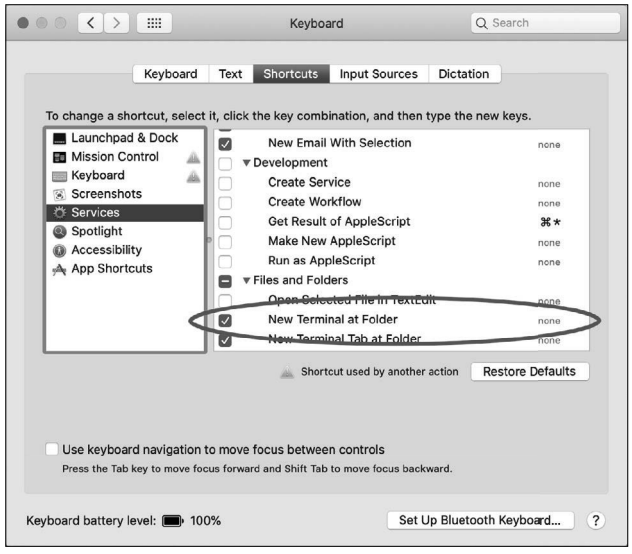

Figura 1.3 macOS: abilitate l'opzione Nuovo terminale nella cartella.

Dopo aver fatto questo.

1. Aprite l'app Docker dalla cartella Applicazioni e accettate i termini.
2. Tornate nel Finder.
3. Raggiungete la directory in cui desiderate collocare il sito web.
4. Fate clic destro del mouse (o Control-clic) sul nome della directory, scegliete l'opzione *Servizi* e selezionate *Nuovo terminale nella cartella*.

Ora che avete aperto il terminale, potete passare al paragrafo "Introduzione a Docker" di seguito.
Le prossime volte, dovrete solo ripetere l'ultima serie di passaggi ogni volta che volete avviare o arrestare il server.

Installazione su Linux

Linux generalmente rende l'installazione del software molto semplice, e Docker, fondamentalmente, è un programma Linux. Nella maggior parte delle distribuzioni, può essere installato tramite il gestore di pacchetti, ma potrebbe essere necessario configurare un repository di pacchetti personalizzato.
Per le distribuzioni basate su Debian (Debian, Ubuntu, Mint e KDE Neon) e le distribuzioni basate su RedHat (RedHat, CentOS, Fedora) c'è uno script di installazione che imposta un repository personalizzato e installa Docker:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Per istruzioni specifiche per la vostra distribuzione, consultate il manuale di Docker (`https://docs.docker.com/engine/install`).
Una volta installato Docker, dovrete abilitare e avviare il servizio utilizzando i seguenti comandi:

```
sudo systemctl enable docker
sudo systemctl start docker
```

Infine, potete aggiungervi al gruppo *docker* in modo da non dover più eseguire le applicazioni Docker tramite `sudo`:

```
sudo usermod -aG docker ${USER}
```

La modifica dei gruppi di utenti richiede di fare logout e poi login (ma ricordatevi di salvare tutto ciò che avete aperto!). Una volta rientrati, sarete pronti per aprire una finestra del terminale.
Dopo aver fatto nuovamente login, create sul computer una directory in cui desiderate collocare il sito web. Può essere ovunque: la directory Home, il desktop, un disco rigido esterno e così via, ma assicuratevi di ricordarvi dove si trova, poiché dovrete tornarvi spesso. Aprite la cartella nel file manager e scegliete un'opzione denominata "*Apri terminale*". Nel file manager Dolphin, potete fare clic destro e scegliere *Apri terminale*, ma questa opzione potrebbe trovarsi in una posizione differente e denominata in modo differente a seconda del file manager utilizzato.

Iniziare a usare Docker

Ora che avete installato tutto il software e il terminale è aperto nella posizione corretta, è il momento di eseguire alcuni comandi di Docker.
Sebbene possiate certamente impostare manualmente l'ambiente e configurare tutto da soli, tali operazioni richiedono un set di abilità che non riguardano la programmazione e non rientrano negli scopi di questo libro. Fornisco alcune configurazioni nell'articolo che ho citato in precedenza (`https://www.sitepoint.com/docker-php-development-environment`).
Innanzitutto, copiate la configurazione usando il seguente comando:

```
docker run --rm -v ${PWD}:/install vjedev/installer:latest
```

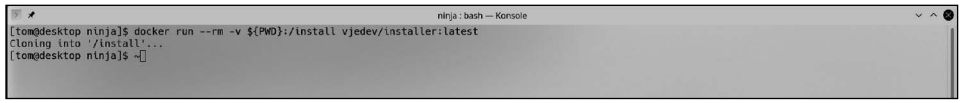

Figura 1.4 Esecuzione del programma di installazione.

Questo comando scarica alcuni file di configurazione da un repository GitHub e li copia nella cartella corrente. La directory in cui si esegue il comando deve essere vuota affinché questa operazione funzioni, quindi assicuratevi che non contenga file o subdirectory.

GitHub e i repository

GitHub è una piattaforma di hosting del codice sorgente utilizzata da molti progetti opensource. Un *repository* è il nome dato a un singolo progetto. In questo caso, il codice viene scaricato da `https://github.com/v-je/docker`.

Uso di Git

Se avete installato Git, a questo punto potete semplicemente eseguire il seguente comando:

```
git clone https://github.com/v-je/docker .
```

Tuttavia, invece di chiedere a tutti i lettori di questo libro di installare Git per un singolo comando, ho creato un apposito script di installazione.

Occorre farlo solo la prima volta e, al termine del comando, nella directory compariranno alcuni file di configurazione. Provate a dare un'occhiata ai file creati. C'è `nginx.conf`, che configura il server web; `PHP.Dockerfile`, che configura le estensioni PHP; e `dockercompose.yml`, che elenca tutti i programmi che verranno installati ed eseguiti all'avvio del server (NGINX, MySQL, PHP e così via).
Non preoccupatevi di cercare di capire il significato di questi file: servono solo per configurare il server.

Apache e NGINX

Prima di procedere, vale la pena di ricordare che l'ambiente che abbiamo appena impostato utilizza NGINX e non Apache. Se utilizzate XAMPP o un pacchetto simile, il server web che state utilizzando è Apache. Il server web è la parte del server che ascolta le richieste dei browser web e invia loro i file. Apache va bene e funziona, ed esiste da sempre. Quando è stato creato Apache, il Web era un luogo molto diverso. Apache è molto esteso e supporta ancora moltissime funzionalità che sono andate e sparite nel tempo. Il Web è cambiato molto da quando è stato creato Apache e, sebbene sia un server molto potente e funzioni bene, la maggior parte dei siti web oggi tende a utilizzare

NGINX. È più facile da configurare, più leggero e meglio sintonizzato con le attività utilizzate da molti siti web moderni (pensiamo allo streaming video). Per questo motivo, la sua quota di mercato sta crescendo rapidamente, a spese di Apache (`https://news.netcraft.com/archives/category/web-server-survey`).
Il mio consiglio generale è: se avete già un sito web che usa Apache, non c'è motivo di dover affrontare la seccatura di cambiarlo. Ma se state iniziando un nuovo progetto da zero, usate NGINX.

Ora che avete scaricato i file di configurazione del server, è ora che Docker scarichi ed esegua tutti i programmi richiesti per il server. Digitate il seguente comando:

```
docker compose up
```

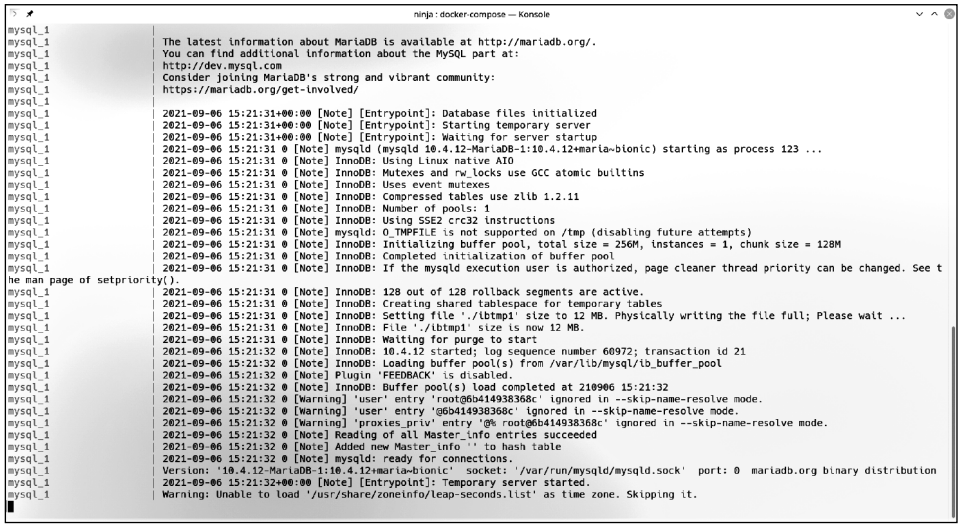

Figura 1.5 L'output del comando docker compose up.

Composizione

Per noiosi motivi tecnici che non vale neanche la pena di approfondire, il comando `docker compose` un tempo era un programma separato chiamato `docker-compose`. Questa funzionalità è stata ora spostata in Docker. Tuttavia, se utilizzate una distribuzione Linux precedente o una versione precedente di Docker Desktop, potrebbe essere necessario sostituire `docker compose` con `docker-compose` in tutti i comandi che io scrivo come `docker compose`. Altri comandi, come `docker run`, non dovrebbero avere questo problema.
Potreste ottenere il seguente errore durante l'esecuzione di `docker compose up`:

```
docker: 'compose' is not a docker command.
```

In tal caso, dovrete sostituire il comando con `docker-compose up` e, su Linux, probabilmente dovrete anche installare il pacchetto `docker-compose` utilizzando il gestore di pacchetti.

Il comando `docker compose` carica `docker-compose.yml`, quindi scarica e installa tutto il software elencato come servizi. Le cose principali che ci interessano, per ora, sono PHP, NGINX e MariaDB.
La prima volta che eseguite questo comando, impiegherà un paio di minuti, poiché deve scaricare tutto il software richiesto. Non preoccupatevi: accade solo la prima volta. Una volta scaricato e installato il software, avviare il server, in futuro, sarà molto più semplice. A differenza dell'utilizzo di un'installazione manuale di NGINX/PHP/MySQL direttamente sul PC, il server viene avviato solo quando lo desiderate, eseguendo `docker compose up`. Potete fermare il server in qualsiasi momento tornando nel terminale e premendo Ctrl+C.
In alternativa, potete avviare il server in background usando questo comando:

```
docker compose -d
```

Potete arrestare il server eseguendo `docker compose down` dal terminale (ma ricordatevi di aprire il terminale nella directory corretta).
Ogni volta che volete lavorare sul sito web, potete avviare e arrestare il server utilizzando la stessa procedura.

1. Andate alla cartella che avete creato.
2. Aprite il terminale o Windows PowerShell come avete fatto in precedenza.
3. Eseguite il comando `docker compose up -d`.

Al termine, potete arrestare il server utilizzando lo stesso processo.

1. Andate alla cartella che avete creato.
2. Aprite il terminale o Windows PowerShell come avete fatto in precedenza.
3. Eseguite `docker compose down`.

Se avete lasciato il terminale aperto in background, potete semplicemente aprire la finestra ed eseguire il Passo 3.

Alti e bassi

Il comando `docker compose down` ferma i servizi in esecuzione e li rimuove, liberando spazio su disco. Li costringe anche a venire ricreati ogni volta.
Potete anche utilizzare `docker compose stop` e `docker compose start` per arrestare e avviare il server. Ma questi comandi contano sul container, sulla rete e su altre cose che Docker crea dietro le quinte tra le esecuzioni. I ripristini del sistema, lo spostamento di file tra computer o persino l'esecuzione di altri comandi Docker che state utilizzando per altre attività (attenzione con `docker prune`!) potrebbero rimuoverli.
L'utilizzo di `up` e `down` è un un approccio a "tabula rasa" e sarà sempre affidabile. Tuttavia, richiede la rimozione e la ricreazione. Ma non preoccupatevi: tutti i vostri file PHP, JavaScript e CSS contenuti nella directory che avete creato non verranno mai rimossi.

Connessione al server e creazione del primo file

Ora che avete il server in esecuzione, potete connettervi con esso utilizzando il browser web. Il server si comporta come un vero server web in esecuzione da qualche parte in Internet. Potete connettervi aprendo il browser, digitando `v.je` nella barra degli indirizzi e premendo Invio.
Dovresti ottenere una pagina di test, come quella rappresentata nella Figura 1.6.

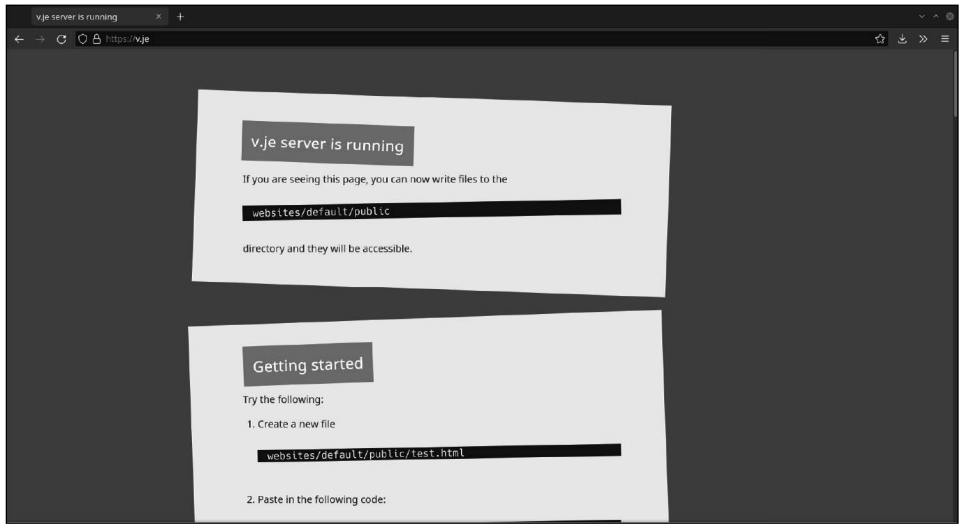

Figura 1.6 Uno screenshot della pagina di test del server.

Ora che il server è in esecuzione, è il momento di creare il primo file. Quando avete eseguito il comando `docker-compose up`, è stata creata automaticamente una cartella chiamata `websites` nella directory dalla quale è stato eseguito il comando. I file contenuti nella cartella `websites/default/public` sono proprio i file utilizzati per la pagina di test della Figura 1.6.
Usando un editor di testi, create un file chiamato `test.html` nella directory `websites/default/public`. Inserite nel file il seguente codice:

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

Ora potete visualizzare la pagina web sul server. Per default (vedremo più avanti che le cose non sempre sono così) un URL viene mappato direttamente su un file. Un file chiamato `test.html` collocato nella directory `public` può essere richiamato come `https://v.je/test.html`, un file `public/products.html` può essere richiamato come `https://v.je/products.html` e i file possono anche essere collocati in subdirectory. Per esempio, `public/images/logo.png` può essere richiamato con l'URL `https://v.je/images/logo.png`.

Livelli di cartelle

Potreste chiedervi perché ci sono tre livelli di cartelle (`websites`, `default` e `public`) invece di avere solo la directory `public` contenente i file.
L'ambiente è configurato per ospitare più siti web in cartelle differenti all'interno della directory generale `websites`. Per creare un nuovo sito web disponibile su `https://mysite.v.je/`, procedete come segue.

1. Create la directory `mysite` all'interno della directory dei siti web.
2. Create la directory `public` all'interno della directory `mysite`.
3. Collocate i file che devono essere accessibili dal Web all'interno della directory `websites/mysite/public`. Per esempio, il file `websites/mysite/public/phpinfo.php` sarà accessibile all'URL `https://mysite.v.je/phpinfo.php`.

Qualsiasi directory creata all'interno della directory `websites` viene trattata come un sottodominio di `v.je`. Se non specificate alcun sottodominio e visitate solo `v.je`, il sito web verrà caricato dalla directory `default`.
Come vedremo più avanti nel libro, è buona norma mantenere determinati file al di fuori della directory `public` per motivi di sicurezza; questo è il motivo per cui ogni sito web ha una sua directory `public`.

Editor di testo

Gli editor di testo forniti dal sistema operativo, come il Blocco note o TextEdit, non sono molto adatti per l'editing di script HTML e PHP. Tuttavia, ci sono diversi editor di testo molto validi e gratuiti, con un ricco supporto per l'editing di script PHP. Eccone alcuni che funzionano in Windows, macOS e Linux.

- Visual Studio Code (`https://code.visualstudio.com`): guida introduttiva in `https://visualstudio.microsoft.com/vs/getting-started`.
- Atom (`http://atom.io`): guida introduttiva in: `https://flight-manual.atom.io/getting-started`.
- Sublime Text (`http://www.sublimetext.com`): documentazione in: `http://www.sublimetext.com/docs`.

Sono tutti molto simili e, ai fini di questo libro, ognuno di essi è una buona scelta e vi semplificherà molto le cose rispetto al Blocco note o a TextEdit.

Ora è tutto pronto

In questo capitolo, avete imparato a configurare un server web con Docker e a collocare un file HTML sul server. Ho trattato solo le basi per farvi arrivare rapidamente alla "sostanza" di questo libro: programmare in PHP. Quello che ci interessa soprattutto è avere un buon flusso di lavoro come sviluppatori PHP. Per i nostri scopi, quello che conta è che ora il server è attivo e funzionante e siete pronto per scrivere il vostro primo script PHP.