Ora del GOLEM - i dotfiles

Una gestione ragionata dei files di configurazione

29/06/2021

Indice

1 Riscopriamo l'acqua calda: cosa sono i "dotfiles"?

Tutti lo sapete già: i dotfiles sono quei files (o directory) che iniziano con il carattere punto (".") e che proprio per questa caratteristica risultano normalmente non visibili con gli strumenti che elencano il contenuto del filesystem.

Sono prevalentemente dominio del mondo Unix-like dove vengono tradizionalmente utilizzati per memorizzare, relativamente allo user space, i dati utili al funzionamento dei programmi, ma si possono tranquillamente trovare anche in sistemi con Windows.

Rob Pike[1] racconta che i dotfiles sono nati come effetto collaterale di una scorciatoia nel codice di Unix presa, non si sa se da Ken (Thompson)[2] o da Dennis (Ritchie)[3] [!], fatta per non mostrare le entries "." e ".." quando il filesystem di Unix divenne gerarchico[4].

Da sempre i dotfiles, nell'accezione di files di configurazione, sono posizionati nella `$HOME` e questo porta, con l'avanzare del tempo dalla prima installazione, ad un certo affollamento fisiologicamente dovuto ai nuovi programmi che vengono di volta in volta installati e che hanno bisogno di memorizzare i propri dati.

Questa condizione di affollamento ha iniziato a migliorare con la progressiva adozione dello standard "XDG Base Directory Specification"; standard che si prefigge lo scopo di mettere ordine nel marasma dei files creati dai vari programmi nella home directory.

2 Note sullo standard "XDG Base Directory Specification"

Si tratta di un insieme di specifiche, redatte dal X Desktop Group, che puntano standardizzare il posizionamento dei dotfiles, definendone delle categorie e indicando le directory all'interno della home destinate ad essere usate per ciascuna categoria.

Molte delle applicazioni preesistenti sono state adeguate o sono in fase di adeguamento.

Ciascuna applicazione segue però le proprie scelte per la compatibilità con la gestione legacy delle impostazioni. Mentre alcune danno precedenza alla ricerca nelle directory indicate dalle specifiche e solo se la ricerca non ha esito proseguono con il metodo legacy, altre agiscono in modo diametralmente opposto. Allo stesso modo la posizione dove i files vengono creati automaticamente varia da un'applicazione all'altra. Occorre quindi consultare la documentazione dell'applicazione per i dettagli.

Da notare infine che il files e le directory poste all'interno della struttura definita dalle specifiche normalmente non sono nascosti.

Il wiki di Arch contiene un elenco di applicazioni che supportano le specifiche con indicazioni relative anche ai percorsi legacy.

Un paio di link esplicativi:

- XDG Base Directory Specification (X Desktop Group)[5]
- XDG Base Directory (Arch Wiki)[6]

Le categorie e le directory associate sono tipicamente indicate con delle variabili d'ambiente che, se non impostate, prevedono dei percorsi predefiniti.

[1] `https://it.wikipedia.org/wiki/Rob_Pike`

[2] `https://it.wikipedia.org/wiki/Ken_Thompson`

[3] `https://it.wikipedia.org/wiki/Dennis_Ritchie`

[4] `https://web.archive.org/web/20190202170417/https://plus.google.com/+RobPikeTheHuman/posts/R58WgWwN9jp`

[5] `https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html`

[6] `https://wiki.archlinux.org/title/XDG_Base_Directory`

2.1 $XDG_CONFIG_HOME

Destinata a contenere i files e le directory di configurazione specifici dell'utente. E' analoga a `/etc/`. Se la variabile non è impostata assume solitamente il valore predefinito di `$HOME/.config`.

2.2 $XDG_CACHE_HOME

Destinata a contenere dati non essenziali (cache) dell'utente. E' analoga a `/var/cache`. Se la variabile non è impostata assume solitamente il valore predefinito di `$HOME/.cache`.

2.3 $XDG_DATA_HOME

Destinata a contenere i files di dati dell'utente. E' analoga a `/usr/share`. Se la variabile non è impostata assume solitamente il valore predefinito di `$HOME/.local/share`.

2.4 $XDG_STATE_HOME

Destinata a contenere i files di stato delle applicazioni. E' analoga a `/var/lib`. Se la variabile non è impostata assume solitamente il valore predefinito di `$HOME/.local/state`.

Con i files di stato si intendono files che contengono informazioni di cui è necessaria la persistenza tra i riavvii delle applicazioni ma che comunque non sono fondamentali. Un esempio tipico è rappresentato da logs, history, recents ecc.

2.5 Altre variabili

Lo standard prevede anche altre variabili d'ambiente che consentono di specificare elenchi ordinati di percorsi di ricerca quando i files non sono presenti nelle directory `$XDG_???_HOME`. Si veda la documentazione dello standard.

2.6 Estensioni alla specifica

Molte distribuzioni Linux prevedono delle estensioni alla specifica che stanno diventando standard de-facto. Una estensione molto comune è l'uso della directory `$HOME/.local/bin` che normalmente non esiste ma che è sempre più spesso inclusa nel $PATH come impostazione predefinita.

3 E' importante gestire i dotfiles

I dotfiles contengono dati importanti, per questo è bene gestirli in modo adeguato.

Si potrebbe pensare che con le normali operazioni di backup si risolva definitivamente il problema. Soluzione semplice, senza troppi sbattimenti ed efficace.

In realtà i backup sono il minimo sindacale sotto il quale non si deve mai scendere, e la cosa deve riguardare tutto il sistema e non i soli dotfiles. Repetita iuvant.

Ma i dotfiles sono spesso più complessi di semplici elenchi di parametri composti da chiave/valore. Capita non di rado che contengano script di shell o scritti in altri linguaggi.

C'è di più: a differenza di Windows sono praticamente sempre files di testo (ma quella è un'altra storia) e quando si parla di files di testo la prima cosa che viene in mente a un programmatore è VCS, alias Git!

4 Perché usare Git per i dotfiles?

Perché no? L'uso Git e un minimo di organizzazione porta in dote tutti quei vantaggi tipici dei VCS:

- E' possibile vedere tutte le modifiche che sono state fatte nel tempo e, compatibilmente con la qualità della descrizione inserita nel commit, risalire alla motivazione della modifica.
- E' possibile ripristinare velocemente modifiche fatte per errore; molto più semplice e veloce che ripristinare un backup.
- E' relativamente semplice e veloce gestire le configurazioni di più macchine in un unico repository.
- Usando un repository in hosting è facile accedere ai propri files di configurazione in ogni parte del mondo.
- Per lo stesso motivo di cui sopra è possibile e semplice condividere le proprie impostazioni con altri.

Mettere i dotfiles in un repository è indubbiamente utile. Occorre però ricordarsi di prestare attenzione a cosa ci si mette: evitare assolutamente di mettere nel repository file che contengano informazioni sensibili. Qualcuno ha detto `$HOME/.ssh`? In realtà vedremo che certi strumenti consento la memorizzazione crittografata di questi files.

5 Stessi dotfiles su macchine diverse

Si diceva che è relativamente facile gestire le configurazioni su macchine diverse. Con un po' di impegno in più si può anche avere un certo automatismo anche su piattaforme diverse.

Questa funzionalità è raggiungibile sia tramite l'uso di script personalizzati che tramite l'uso di strumenti che prevedono l'uso multi-piattaforma.

Occorre però tenere presente che, mentre per il mondo Linux/Mac il numero di strumenti utilizzabili è elevato, lo stesso con accade per il mondo Windows dove il numero di strumenti decisamente ridotto.

6 Ok, da dove comincio?

Occorre usare un "qualcosa", software o metodo, che aiuti in questa gestione.

Parlando di dotfiles mantenuti in repository Git, le argomentazioni sul metodo che si trovano in rete portano essenzialmente a due scuole di pensiero:

Uso di un bare repository. all'interno della home. E' solitamente posizionato nella home, ma contrariamente a quanto si possa pensare in prima battuta, non si tratta far diventare la home un repository. Si tratta appunto di creare un bare repository che sarà contenuto in una subdirectory della home, subdirectory che può essere nascosta e spesso è chiamata `.dotfiles`. Le operazioni di interazione con il repository saranno definite con degli alias di shell. Questo approccio è veramente semplice ed efficace, ma soffre di qualche idiosincrasia. Mentre è immediatamente fattibile con Linux e similari, con Windows ci si può arrivare soltanto con l'uso della git-bash, di WSL o di powershell (questa sconosciuta). Questo metodo diventa infine sempre più complicato se si vogliono avere configurazioni leggermente diverse da macchina a macchina e mantenere un repository unico. Alcuni link sull'argomento:

- The best way to store your dotfiles: A bare Git repository (Atlassian)[7]
- The best way to store your dotfiles: A bare Git repository **EXPLAINED**[8]
- How to manage dotfiles with a Git bare repository[9]

[7] `https://www.atlassian.com/git/tutorials/dotfiles`

[8] `https://www.ackama.com/blog/posts/the-best-way-to-store-your-dotfiles-a-bare-git-repository-explained`

[9] `https://daniele.tech/2021/03/how-to-manage-dotfiles-with-a-git-bare-repository`

Uso di un repository normale. Si tratta di un normale repository Git con la sua working directory e contenuto all'interno della home o in una qualsiasi altra directory. Anche qui non si tratta di far diventare la home un repository. Si differenzia dall'altro approccio perché mentre nel primo i files sono effettivamente presenti nella home, in questo si andrà tipicamente ad utilizzare dei link simbolici. Con questo metodo si sfruttano le capacità del filesystem per operare direttamente sul file nel repository come se fosse nella home. La gestione del repository avviene normalmente e con l'uso di script personalizzati o strumenti appositi è possibile gestire facilmente i link simbolici ma anche eventuali differenze tra vari PC.

La differenza tra le due scuole di pensiero si evidenzia anche nel come si devono gestire le modifiche.

Con il metodo del bare repository tutto si limita ad usare degli alias che prevedano i parametri necessari ma continuando a ragionare come se si avesse a che fare con un normale repository.

Con il repository normale, invece, occorre avere un qualcosa che ne consenta la gestione.

Da questo punto di vista, scartando l'opzione di fare tutto a mano di volta in volta, si può affrontare la cosa in vari modi.

6.1 Mi faccio la gestione in casa

Con le giuste competenze e dedicandoci tempo si possono creare delle procedure che consentano la gestione dei dotfiles accontentando le nostre più particolari esigenze. E' la strada scelta da molti e, quasi certamente, è alla base dei tool che nel tempo sono stati presentati in rete. E' probabilmente l'unica strada per raggiungere il massimo grado di soddisfazione, ma è anche quella più impegnativa.

Questo approccio consente anche la gestione di elementi non necessariamente attinenti ai dotfiles. Facendo tutto su misura si può naturalmente andare a gestire anche impostazioni di sistema e non limitatamente al solo utente.

La prima cosa che viene in mente è usare uno script di shell, ma ovviamente non è un dogma. Girando in rete ho trovato questo video su Youtube dove viene mostrato un approccio basato su Make: How To Manage Your Dotfiles With Make[10]

6.2 Uso strumenti già pronti

E' probabilmente la soluzione più semplice. In rete sono presenti molti strumenti che coprono con varie modalità le esigenza più comuni.

Se poi lo strumento scelto non fosse perfetto per noi possiamo sempre sfruttare i vantaggi messi a disposizione dal mondo dell'open source: proporre delle patch o fare un fork.

L'elenco degli strumenti reperibili in rete è discretamente lungo e in continuo aggiornamento.

Tra questi quelli che hanno in qualche modo attirato la mia attenzione quelli che:

- Sono multi-piattaforma e oltre ai canonici Linux/Mac includono anche Windows.
- Sono per quanto possibili esenti da dipendenze esterne intendendo questa affermazione in modo "lasco", nel senso che se le dipendenze sono normalmente presenti nel sistema e non devo installare niente allora va bene, ma va bene anche usare un singolo eseguibile che magari può essere inserito nel repository stesso.
- Non soddisfano i criteri precedenti, ma appartengono comunque al gruppo dei più utilizzati in base ad una statistica veloce e non ortodossa risultante da qualche ricerca in rete.

6.2.1 GNU Stow

E' un software che nasce essenzialmente per gestire la creazione di link simbolici. E' nativo Linux e credo funzioni anche su Mac, ma non Windows.

[10] `https://www.youtube.com/watch?v=aP8eggU2CaU`

Gode di una certa fama in quanto semplice ed immediato, ma si limita a fare una ed una sola cosa, anche se la fa bene. Con Stow, infatti, potete soltanto gestire i link simbolici.

Il funzionamento è appunto molto semplice: è sufficiente creare una directory in cui andare a creare tante sottodirectory quanti sono i "moduli" che vogliamo gestire. Questa semplicità rende più complicata la gestione di configurazioni diverse su macchine diverse. L'ostacolo si può in qualche modo aggirare con degli script, ma non si possono gestire parti in comune.

Link alla pagina del software[11]

6.2.2 dotbot

Uno "Stow on steroid" per forza di cose un po' più complesso, ma non troppo.

Scritto in Python consente la gestione dei dotfiles come link simbolici ma anche molto altro come, ad esempio, configurazioni separate per macchina/profilo o l'esecuzione di comandi. Da notare la possibilità di usare dei plug-in oltre al fatto che, dando per scontato la presenza di Python su tutte le installazioni Linux, la replica di una configurazione si limita ad una riga di bash.

Purtroppo non funziona in modo nativo in Windows.

Pur mantenendo una relativa semplicità offre molte funzionalità in più rispetto a Stow.

Link alla pagina del software[12] I dotfiles dello sviluppatore[13]

6.2.3 Chezmoi

Scritto in GO - quindi senza dipendenze - funziona in ambiente Linux/Mac e Windows.

Usa un approccio diverso rispetto agli altri perché invece di usare dei link simbolici gestisce a livello applicativo i dotfiles.

Sulla carta offre capacità notevoli, tra cui il funzionamento in Windows. Ad una prima prova è però risultato poco confortevole, forse anche a causa di una documentazione che trovo poco chiara.

Occorre comunque tenere presente questo strumento che gode di molti giudizi positivi in rete.

Link alla pagina del software[14] I dotfiles dello sviluppatore[15]

6.2.4 Yadm

E' uno degli strumenti per la gestione dei dotfiles che si trovano più frequentemente nelle discussioni in rete.

Scritto in Python, funziona soltanto su Linux/Mac, ma offre molte funzionalità comunemente apprezzate quale la gestione sia di files specifici per sistema che l'uso di template. Apprezzabile la possibilità di crittografare dati sensibili.

Link alla pagina del software[16]

6.2.5 Varie ed eventuali

Il mondo degli strumenti di ausilio alla gestione dei dotfiles è i continuo fermento e il loro numero aumenta quasi giornalmente, per cui è praticamente impossibile approfondire tutta la produzione disponibile.

Alcuni tra quelli trovati nelle varie ricerche in rete sono comunque degni di nota e da considerare, se non altro, per vedere come evolvono.

1. Homeshick

   Completamente scritto in Bash - quindi senza dipendenze.

[11] `https://www.gnu.org/software/stow`
[12] `https://github.com/anishathalye/dotbot`
[13] `https://github.com/anishathalye/dotfiles`
[14] `https://github.com/twpayne/chezmoi`
[15] `https://github.com/twpayne/dotfiles`
[16] `https://yadm.io`

Un altro gestore di link simbolici.

Link alla pagina del software[17]

2. Dotprop

   Scritto in Python.

   Offre alcune caratteristiche interessanti quali la possibilità di gestire configurazioni separate per macchina di destinazione e l'utilizzo di template per avere impostazioni differenti all'interno dello stesso dotfile.

   Link alla pagina del software[18]

3. Toml-bombadil

   Scritto in Rust, multi-piattaforma

   Interessante progetto che prendendo come modello Chezmoi cerca di semplificarne un po' l'uso gestendo soltanto alcune funzionalità peculiari. A differenza di Chezmoi presenta una documentazione scritta discretamente e quindi facilmente comprensibile

   Link alla pagina del software[19] I dotfiles dello sviluppatore[20] L'esperienza d'uso di un utilizzatore[21]

4. Dotter

   Un altro "Stow on steroid" che però offre alcuni importanti vantaggi:

   - E scritto in Rust: non ha dipendenze e funziona su tutte le piattaforme
   - Consente l'uso di template anche se files originati da template non sono link simbolici ma files veri e propri: se si deve modificare qualcosa occorre modificare il template e poi ricreare il dotfile con dotter
   - Consente una qualche forma di gestione differenziata tra le macchine target, ma deve essere gestita manualmente.

   Link alla pagina del software[22] I dotfiles dello sviluppatore[23]

5. Pearl

   Scritto in Python è usabile soltanto in ambito Linux/Mac. Approccio interessante basato sulla modularità.

   Link alla pagina del software[24]

6. Comtrya

   Scritto in Rust, multi-piattaforma.

   Un progetto nato da poco che coniuga la gestione dei dotfiles con l'installazione del sistema da zero consentendo l'installazione anche del software. Nella documentazione del progetto viene definito una versione semplificata di Ansible o Salt.

   Il progetto e' ancora molto giovane, ma certamente interessante.

   Link alla pagina del software[25]

[17] `https://github.com/andsens/homeshick`
[18] `https://github.com/deadc0de6/dotdrop`
[19] `https://github.com/oknozor/toml-bombadil`
[20] `https://github.com/oknozor/dotfiles`
[21] `https://rgoswami.me/posts/dotfiles-dotgit-bombadil`
[22] `https://github.com/SuperCuber/dotfiles`
[23] `https://github.com/SuperCuber/dotfiles`
[24] `https://github.com/pearl-core/pearl`
[25] `https://github.com/comtrya/comtrya`

6.3 Una via di mezzo

Quello che può capitare è che magari non si ha il tempo o le conoscenze per potersi mettere a sviluppare un proprio modo per gestire i dotfiles e che non si trovino strumenti che ci piacciono.

Mantenendo l'intenzione di gestire in qualche modo i dotfiles non rimane che cercare in rete quello che più si avvicina al nostro pensiero e poi provare a modificarlo. Si può magari fare un lavoro di collage prendendo spezzoni e idee provenienti da più progetti.

Anche se utile non è necessario essere profondi conoscitori del linguaggio usato dal creatore originale. Certo che prima di eseguire sulla propria macchina un pezzo di codice preso da uno "sconosciuto" è buona regola provare almeno a capire se presenta situazioni in cui si possono creare dei danni.

Quello che si otterrà è un franken-qualcosa che magari legato con il filo di ferro fa quello che volevamo. Più o meno.

Storicamente le mie conoscenze di Bash sono sempre state piuttosto misere. Quelle di Powershell sono probabilmente ancora minori. Con i vecchi script ".bat" o ".cmd" di Windows si possono fare diverse cose ma forse questa sarebbe stata un po' troppo complessa.

Qualche anno fa mi sono messo alla ricerca di qualcuno che avesse risolto il mio "problema". Dopo vari tentativi ho trovato un qualcosa che si avvicinava alle mie esigenze, ho spudoratamente copiato quello che aveva fatto lui e con qualche modifica fatta cercando di limitare al massimo i danni mi sono avvicinato ancora di più a quello che volevo.

Nella realtà le mie necessità erano e sono abbastanza semplici: poter gestire i files di configurazione dei programmi che uso sia sulle macchine Windows che su quelle Linux.

Sono passati degli anni e non ho più memoria di chi è la persona da cui ho spudoratamente copiato, la ringrazio comunque.

Ad oggi ho un paio di script che, seguendo delle convenzioni nel naming dei files, mi consentono di gestire i miei files di configurazione partendo dallo stesso repository sia su Windows che su Linux.

7 Conclusioni

Come dicevo la mia esigenza rispetto ai dotfiles è sempre stata piuttosto semplice: avere per quanto possibile un unico file di configurazione che possa andare bene sia su Windows che su Linux e mettere questi files di configurazione in un repository Git in modo da poterlo facilmente scaricare da più macchine sfruttando tutti i vantaggi di un VCS. A corollario di questo un qualche sistema che mi consenta un qualche automatismo nel bootstrap della configurazione.

Fino a quando non ho cominciato ad accarezzare l'idea di parlare dei dotfiles quello che avevo mi è sempre stato più che sufficiente. Cercando però documentazione per approfondire ho trovato che l'argomento è molto più trattato di quanto potessi pensare e non è considerato affatto banale come credevo.

Nell'ambito dei dotfiles c'è una certa effervescenza sia negli articoli di blog che spiegano i vari approcci, sia negli strumenti di gestione che presentano con una certa frequenza nuovi elementi.

8 Possibili approfondimenti

La classica ricerca su Google mostrerà la solita quantità infinita di link.

Ho però trovato due punti da cui partire e che, in linea di massima, concentrano quanto di più importante c'è da sapere. Sono due pagine che in realtà si mettono nei riferimenti l'una dell'altra, per questo potrei anche metterne una sola, ma per quanto alla fine contengano molte informazioni duplicate, si completano in quelle mancanti:

- Awesome dotfiles - A curated list of dotfiles resources[26]

[26] `https://github.com/webpro/awesome-dotfiles`

- Your unofficial guide to dotfiles on GitHub.[27]

Partendo da questi è facile perdersi in una miriade di articoli che trattano l'argomento. E' tuttavia vero che in mezzo a questo mare si trovano elementi anche molto interessanti che possono essere di spunto per migliorare la nostra impostazione nella gestione dei dotfiles, ma talvolta anche per approfondire o affrontare argomenti di più ampio respiro.

[27] `https://dotfiles.github.io`