UN SISTEMA DI BACKUP PER MOODLE SU SERVER LINUX

Edoardo Bontà[1]

[1] Università degli Studi di Urbino Carlo Bo
edoardo.bonta@uniurb.it

FULL PAPER

ARGOMENTO: Implementazioni e soluzioni tecniche

Abstract

Il sistema *Moodle Backup Scripts* è una collezione di comandi bash per Linux sviluppati per eseguire il salvataggio a freddo, o "*cold backup*", di un sito web Moodle, ovvero per salvare dati, file, codice e configurazioni del sito dopo averlo temporaneamente arrestato. La peculiarità del sistema è quella di realizzare con rapidità un backup consistente a livello di applicazione, permettendo sia la migrazione del sito in altre piattaforme, sia il recupero completo dell'applicazione web o di sue singole parti che risultassero eventualmente compromesse dopo interventi di aggiornamento o manutenzione. Il backup può essere avviato in modalità manuale o in modalità automatica, demandando le azioni più onerose in termini di tempo – ad esempio il trasferimento dei file già salvati/sincronizzati nel disco locale verso destinazioni remote – a script asincroni eseguiti dopo il riavvio del sito web e personalizzabili dall'amministratore.

Keywords – Moodle, Linux, backup, application consistent backup, cold backup, bash, downtime, performance, migrazione, manutenzione, aggiornamento.

1 INTRODUZIONE

La documentazione ufficiale di Moodle suggerisce sinteticamente le linee guida per il backup di un sito [10], utili per salvare tutto – e soltanto – quanto associato all'applicazione web Moodle. Questo tipo di salvataggio selettivo differisce da quello dei sistemi di *backup & recovery* impiegati usualmente nelle server farm e messi a disposizione dagli hosting provider, la cui strategia è di norma quella di creare una copia o "immagine" della intera macchina, reale o virtuale che sia, all'interno della quale è contenuto il sito. Tali sistemi, fondamentali per servizi in produzione in cui abbia senso preservare configurazione, dati, codice e tutte le altre informazioni presenti, sono concepiti per essere sufficientemente generali ed indipendenti dal contenuto della macchina di cui eseguire il backup, ed hanno l'obiettivo di consentire nei tempi più rapidi possibili il *disaster recovery* [18] ed eventualmente di mantenere la continuità del servizio [16,17] in caso di insorgenza di problemi hardware/software. Caratteristiche tipiche di questi sistemi sono la possibilità di creare *snapshot* – ovvero di salvare punti di ripristino incrementali – senza richiedere alcun arresto del servizio erogato agli utenti, nonché la garanzia del *crash consistent backup* [12], cioè la possibilità di preservare lo stato dell'intero sistema, ovvero di tutti i dati in un istante preciso anche quando essi appartengono a macchine diverse della stessa server farm. Alcuni fra i sistemi più sofisticati garantiscono l'*application consistent backup* [12,18], ovvero il salvataggio non solo di tutti i dati presenti su file, ma anche di tutti i dati in memoria e di tutte le transazioni in corso in un determinato momento. Tale caratteristica, come intuibile, è piuttosto complessa da garantire quando si abbia a che fare con applicazioni distribuite su più macchine e/o quando una applicazione e la relativa base di dati siano ospitate in server separati.

Analoghe considerazioni possono essere fatte riguardo le tecnologie cloud più recenti ed evolute, basate su *container*, come Docker [5], e relativi sistemi di gestione, ad esempio Kubernetes [7], per le quali esistono sistemi *general-purpose* di backup e ripristino molto avanzati [15] ma, purtroppo, comunque esposti a problemi di consistenza dei dati decentralizzati – più o meno volatili – appartenenti ad applicazioni e ad eventuali microservizi [13].

Ciò premesso, esistono varie situazioni in cui è opportuno ricorrere al backup selettivo del sito così come illustrato nella documentazione ufficiale di Moodle, anche se possibilmente in aggiunta – e non come alternativa – ai suddetti sistemi per il backup e ripristino dell'intero ambiente di hosting. Esigenze e situazioni tipiche in cui torna utile il backup selettivo della specifica applicazione web, piuttosto che dell'ambiente complessivo nel quale essa si trova, sono:

- *Migrazione*: si vuole spostare o clonare la sola applicazione web Moodle su un nuovo server, ignorando tutti gli altri elementi presenti nel sistema operativo originale.
- *Manutenzione*: si prevede che un intervento di manutenzione/aggiornamento su Moodle possa introdurre effetti collaterali, magari non immediatamente visibili, ai quali non sarebbe opportuno rimediare con un ripristino dell'intera macchina qualora fossero individuati tardivamente, perché ciò comporterebbe un regresso/involuzione di tutta la piattaforma a partire dalla data di salvataggio del punto di ripristino. Gli effetti collaterali potrebbero però essere affrontati in modo puntuale e mirato servendosi dei dati, delle parti di codice e delle informazioni salvate attraverso backup selettivi.
- *Ridondanza di sicurezza*: si desidera avere, per maggior sicurezza, un secondo sistema di backup rispetto al sistema principale, ovvero rispetto a quello messo a disposizione dalla server farm o dal provider del servizio di hosting.
- *Inconsistenza del sistema principale di backup*: il sistema principale, per motivi che potrebbero essere legati alla configurazione e/o alla distribuzione della applicazione e del relativo database su macchine distinte, non esegue – o non è in grado di garantire – un salvataggio consistente di codice, dati e file.
- *Assenza di altri sistemi di backup*: non si dispone di un sistema per il backup complessivo della macchina, e non è possibile – o non è semplice – implementarlo. Potrebbe essere il caso, ad esempio, di installazioni più o meno amatoriali *bare-metal* su PC, su *single-board computer* [21] o su schede specifiche dedicate [8].

La sussistenza di anche uno soltanto dei punti riportati sopra può essere un valido motivo per ricorrere al backup selettivo per Moodle. Ovviamente, sarà più incoraggiante farlo quanto più esso sarà semplice da eseguire, evitando qualora possibile le procedure manuali facilmente soggette ad errori e, quindi, automatizzando e cercando di rendere rapide le operazioni da svolgere per il backup, come descritto nelle prossime sezioni.

Questo articolo, nel dettaglio, è organizzato come segue. Nella sezione 2 vengono esposte le scelte e le considerazioni che hanno guidato la progettazione del sistema *Moodle Backup Scripts* [1], mentre nella sezione 3 vengono riportate le soluzioni specifiche e gli accorgimenti tecnici adottati per l'implementazione del sistema. La sezione 4 descrive la struttura e le modalità d'uso dei vari script, e la sezione 5 riporta i requisiti d'esecuzione e i risultati misurati. Infine, la sezione 6 conclude l'articolo riportando le peculiarità, i compromessi ed i possibili sviluppi futuri del sistema di backup.

2 ANALISI, STRATEGIE E SCELTE PROGETTUALI

Un sistema di backup è efficace quando ci si può permettere di dimenticarne l'esistenza. Il principio noto come *set-and-forget* [22] indica in generale quei processi che, una volta configurati, non richiedono ulteriori attenzioni da parte di chi li amministra. In un dispositivo questo significa automatizzare una serie di operazioni e, eventualmente, automatizzare la ripetizione stessa di quella serie di operazioni. Per qualsiasi sistema di backup l'automazione è fondamentale perché permette di eseguire attività che sarebbero tediose e ripetitive se svolte manualmente, ma che sono necessarie per garantire l'integrità e la sicurezza del sistema software sottoposto a backup. Senza automazione, spesso, tali attività non verrebbero neppure svolte, per via della mancanza di tempo, o per eccesso di fiducia, o per pura e semplice (e talvolta anche legittima) pigrizia, con conseguente aumento del rischio di perdita di dati importanti e, magari, di mesi o anni di lavoro.

Le linee guida per il backup di un sito fornite dalla documentazione ufficiale di Moodle [10] indicano precisamente quali sono le parti della applicazione web che devono essere salvate – in estrema sintesi: i dati memorizzati nel database, i file creati o caricati dagli utenti, il codice sorgente della applicazione web – ma sono anche sufficientemente generali perché possano essere applicate a qualsiasi sistema operativo e a qualsiasi database compatibile con l'applicazione. A causa di ciò, seppure esse siano corredate da esempi di istruzioni per il *dump* del database più utilizzato assieme a Moodle, cioè MySQL/MariaDB, queste rimangono pur sempre linee guida, e non codice eseguibile.

Un amministratore di sistema che voglia automatizzare il backup selettivo, a questo punto, si troverà di fronte alla seguente scelta: individuare un sistema di backup già esistente, personalizzabile e quindi in grado di eseguire tutti i passi specificati dalle linee guida sulle proprie installazioni Moodle, oppure tradurre direttamente le linee guida in software o in script automatizzati.

In questa sezione verranno descritte considerazioni e scelte messe in atto durante la progettazione del sistema Moodle Backup Scripts, attualmente operativo e collaudato nei server Moodle dell'Università di Urbino Carlo Bo, ma sufficientemente generale per poter essere adottato in molte altre implementazioni standard di Moodle su server Linux.

2.1 Backup selettivo: adottare un software esistente o crearne uno nuovo

Varie ricerche effettuate in rete non hanno permesso di individuare sistemi di backup specifici e consolidati per siti web Moodle. Si è pensato pertanto di orientarsi verso note utilità di backup *open source* [1], individuando fra queste due categorie principali: i sistemi specifici per la sincronizzazione locale o remota di file, come *rsync* [4] o *RClone* [14], e i sistemi di *meta-backup* o personalizzabili, come *backupninja* [3] o *Duplicity* [6]. La seconda categoria risultava senz'altro più indicata per eseguire operazioni eterogenee – come ad esempio il dump di un database, il riavvio di un servizio, il lancio di eventi – che andassero oltre la mera copia di file e directory e che potessero comunque incorporare anche l'esecuzione delle utilità della prima categoria.

L'impiego di sistemi personalizzabili predefiniti, tuttavia, da una parte imponeva alcuni vincoli sui formati dei file target e sul processo di sincronizzazione, mentre dall'altra lasciava troppi gradi di libertà nella implementazione delle operazioni per il backup. La creazione di un nuovo sistema di backup finalizzato a Moodle, invece, avrebbe implementato rigorosamente le operazioni essenziali e critiche per il salvataggio del sito web, e minimizzato l'insieme di opzioni di configurazione per l'amministratore, semplificandogli la vita. Al contempo l'amministratore avrebbe avuto la possibilità di introdurre anche operazioni personalizzate, utili ad esempio per gestire liberamente il trasferimento remoto del backup mediante i suddetti sistemi di sincronizzazione della prima categoria e/o attraverso altri strumenti avanzati di backup incrementale con eventuale compressione e crittografia dei dati

La scelta, dunque, è stata proprio quella di creare un nuovo software di backup, tenendo comunque conto della varietà delle configurazioni di Moodle e delle sue componenti accessorie – cioè il DBMS, il server web, i percorsi di sistema – per evitare soluzioni troppo specifiche e di scarsa applicabilità.

2.2 Scelta del linguaggio di scripting

La scelta del linguaggio per realizzare gli script di backup è stata dettata dal sistema operativo su cui si intendeva agire, Linux, e dalla necessità di usare comandi di sistema per svariate operazioni, quali l'assegnamento dei diritti e della proprietà dei file, la compressione di directory, la sincronizzazione di contenuti, ecc.

In tale contesto, è stato piuttosto naturale optare per il linguaggio Bash, ovvero il linguaggio della shell testuale di GNU/Linux. Una scelta alternativa avrebbe potuto essere PHP nella modalità a linea di comando (CLI), poiché si tratta dello stesso linguaggio su cui è basato Moodle – e quindi normalmente disponibile nel medesimo contesto della applicazione web. Fra le due opzioni, ha prevalso comunque l'esigenza di usare un linguaggio più immediato e sintetico per l'esecuzione diretta dei comandi del sistema operativo.

2.3 Garanzia di consistenza della applicazione

Banalmente, il modo più semplice e diretto per ottenere la consistenza della applicazione, ovvero una sincronia completa fra tutti i file, il database e gli eventuali dati in memoria, è quello di porre Moodle in "stato di manutenzione" [9], così da arrestare in maniera "soft" tutte le transazioni del database ed evitare la presenza di dati critici della applicazione nella memoria volatile, per poi fare il backup di file e database in una situazione assolutamente statica. In questo modo sarà possibile ottenere la consistenza più critica, ovvero quella tra i file caricati dagli utenti e lo stato del database, che in una situazione dinamica sarebbe ben più ardua da garantire.

Purtroppo, il prezzo da pagare per la semplicità di questa soluzione sarà l'interruzione del servizio web per gli utenti. Tale prezzo è assolutamente irrilevante in caso di intervento per manutenzione, ad esempio per aggiornare la versione di Moodle, in quanto il sito deve essere comunque arrestato per poi intervenire manualmente su esso. In tal caso, il salvataggio preliminare di tutti i dati del sito e

l'interruzione del servizio sono esattamente quanto desiderato per procedere con l'intervento. Nel caso di backup periodico ed automatico, invece, l'interruzione del servizio può risultare inopportuna per gli utenti. In tale situazione, gli effetti negativi possono essere attenuati nel seguente modo:

- programmando l'interruzione in orari di scarso traffico, ad esempio la notte;
- cercando di rendere l'operazione di backup più rapida possibile, per ristabilire poi immediatamente la funzionalità del servizio uscendo dallo stato di manutenzione di Moodle.

Più avanti verranno discusse le strategie e gli accorgimenti tecnici adottati per ridurre quanto più possibile i tempi del backup, o meglio, i tempi di inaccessibilità – o *downtime* – del servizio web.

2.4 Ricongiungimento dei dati delocalizzati

Il backup di un sito Moodle comprende tre parti fondamentali: i dati memorizzati nel database, i file caricati dagli utenti e il codice della applicazione web. Non è detto però che tutte e tre le parti del sito risiedano necessariamente nello stesso host. Una configurazione molto frequente, ad esempio, è quella in cui il database è localizzato su una macchina distinta rispetto a quella che contiene il codice dell'applicazione e i file degli utenti.

Fortunatamente, a differenza di quanto avviene per un backup generico della intera macchina o di un intero sistema di host interconnessi, il backup selettivo di Moodle prevede di eseguire il dump del database nello stesso host della applicazione web, seppure il database sia remoto. Infatti, se il database è raggiungibile dalla applicazione, sarà raggiungibile anche dalla stessa macchina che ospita l'applicazione. In maniera analoga, sarà possibile salvare localmente i file caricati dagli utenti, se questi sono remoti ma raggiungibili da un percorso montato nel filesystem dell'host in cui risiede l'applicazione web, ovvero nel quale risiede il codice della applicazione. In definitiva, tutti i dati e file eventualmente delocalizzati, verranno ricongiunti nell'host della applicazione, da cui potranno poi essere gestiti e trasferiti altrove.

2.5 Strategia di backup

In accordo con il meccanismo di ricongiungimento dei dati delocalizzati, la strategia di backup [19] più efficiente risulta quella di usare il/un disco locale dell'host della applicazione come deposito temporaneo ove salvare tutti i dati e i file, per poi copiare tutto quanto in una locazione remota. Una variante potrebbe essere quella di salvare tutto direttamente in un disco remoto ma ciò, solitamente, richiede tempi ben più lunghi rispetto a fare copie e/o sincronizzazioni locali. Avendo una copia locale, invece, sarà possibile ripristinare subito la piena funzionalità del servizio, ed eseguire poi il trasferimento remoto quando l'applicazione web, ovvero il sito Moodle, sarà tornata a funzionare, minimizzando così il downtime del sito.

Anche questa scelta ha un prezzo da pagare, ovvero il fatto che l'host della applicazione dovrà avere sufficiente spazio nella memoria di massa locale, cioè nel disco principale o in un disco secondario riservato espressamente al backup, per salvare la copia temporanea di Moodle prima di trasferirla. Un accorgimento per non dissipare ulteriori risorse quando si utilizza un disco secondario in questa maniera, è quello di far sì che il disco non sia soggetto ad altri meccanismi di backup – ad esempio il sistema di *backup & recovery* della server farm – essendo esso stesso già dedicato a tale scopo.

2.6 Gestione dei parametri specifici e delle istruzioni aggiuntive

Nella implementazione del sistema di backup, per evitare di disperdere energie in tante soluzioni simili e ripetitive, e per realizzare invece una singola soluzione flessibile gestita attraverso il sistema di *versioning* Git, si è pensato di separare il file di impostazione dei parametri specifici dalla base comune di codice. Ovvero, i file di script principali del sistema di backup hanno riferimenti verso un file di impostazioni esterno, chiamato *settings*, che deve però essere creato dall'utente – cioè dall'amministratore – assegnando i valori specifici alle variabili opportune. Per aiutare l'utente, nel repository Git viene fornito comunque un file di esempio chiamato *settings-dist*.

Analogamente al file di esempio delle impostazioni, è definito nel repository anche un file chiamato *postjobcustomscript-dist.sh*. Lo scopo è quello di fornire le istruzioni di esempio, ovvero la struttura minima, per creare il file di istruzioni *postjobcustomscript.sh* nel quale l'utente potrà specificare quali azioni compiere sui dati che saranno salvati localmente dal sistema di backup. In particolare, il file sarà utile per specificare le azioni di trasferimento del backup verso destinazioni remote, nonché eventuali operazioni di sincronizzazione / compressione / crittografia dei dati trasferiti.

3 SOLUZIONI ED ACCORGIMENTI ADOTTATI NEL SISTEMA DI BACKUP

Assieme alle considerazioni e alle scelte progettuali riportate nella sezione precedente, durante lo sviluppo del sistema di backup è emersa l'esigenza di realizzare alcune funzionalità che arricchissero le potenzialità e la flessibilità del sistema, permettendo quindi varie personalizzazioni da parte dell'utilizzatore, nonché alcuni accorgimenti implementativi che ottimizzassero la performance e prevenissero eventuali problemi e conflitti nel processo di backup.

3.1 Elementi aggiuntivi soggetti a backup

Come già affermato in precedenza, il backup di un sito Moodle comprende tre parti fondamentali: i dati memorizzati nel database, i file caricati dagli utenti e il codice della applicazione web. Tuttavia, l'amministratore potrebbe avere l'esigenza di salvare altri dati o file che in qualche modo hanno a che fare con Moodle. Si pensi ad esempio alla configurazione del server web attraverso cui viene attivato il sito. Oppure agli script per eseguire operazioni periodiche (*cron*), o alla configurazione del *Service Provider* di un eventuale sistema di autenticazione per l'accesso a Moodle, come *Shibboleth* [20]. Ancora, potrebbe essere utile salvare alcuni script e utilità definiti dallo stesso amministratore per interagire con Moodle o con il suo database. Oltre i file, potrebbe essere importante salvare anche (il dump di) eventuali database residenti nello stesso DBMS in cui viene conservato quello principale di Moodle e, in qualche modo, ad esso correlati.

Per venire incontro a questa esigenza, è stata inclusa nel sistema Moodle Backup Scripts – attraverso il file di impostazione *settings* – la possibilità di specificare elementi aggiuntivi che dovranno essere soggetti a backup. In particolare, sarà possibile specificare percorsi di directory che verranno salvate sotto forma di file archivio compressi, nonché nomi di database addizionali di cui verrà fatto il dump. Per questi ultimi, la condizione sarà che essi permettano il dump con le stesse credenziali con cui si accede al database principale di Moodle.

3.2 Estensione per personalizzazione istruzioni e procedure post-backup

Oltre gli elementi aggiuntivi soggetti a backup, il sistema permette di specificare anche azioni aggiuntive da eseguire al termine del backup locale. A tal fine, come già scritto in precedenza, l'amministratore può creare e personalizzare a piacimento il file *postjobcustomscript.sh*, che sarà invocato dal sistema di backup al momento opportuno.

3.3 Esecuzione manuale e automatica

Il sistema di backup, nella fase iniziale di sviluppo, è stato concepito per essere avviato soltanto manualmente, cioè in maniera interattiva da parte dell'amministratore. In particolare, sono state concepite due modalità manuali: una di esse, detta *full-restart*, arresta il sito ponendo Moodle in stato di manutenzione, poi esegue il backup locale e infine ripristina la piena funzionalità del sito, ovvero esce dallo stato di manutenzione. La seconda modalità, *half-restart*, al contrario, rimane in stato di manutenzione, consentendo così un effettivo intervento di manutenzione / aggiornamento da parte dell'amministratore immediatamente al termine del backup, senza riavviare inutilmente il sito.

La modalità automatica, invece, è stata sviluppata successivamente con l'idea di essere associata ad un timer periodico (*cron*) e facendo sì che lanciasse le procedure post-backup per il trasferimento remoto dei dati dopo avere eseguito il backup locale ed il ripristino della piena funzionalità del sito.

3.4 Minimizzazione del downtime

Una prima strategia per minimizzare il downtime del sito, ovvero il tempo in cui Moodle si trova in stato di manutenzione, è stata quella già menzionata di scegliere di salvare i dati localmente per poi trasferirli eventualmente in destinazioni remote dopo l'uscita dallo stato di manutenzione.

Poiché, però, la mole dei dati può essere piuttosto alta, in particolare per il numero e la dimensione dei file caricati dagli utenti in Moodle, si è pensato anche di accelerare la fase di salvataggio su disco locale ricorrendo alla utilità di sistema *rsync* [4]. Infatti, durante i test, è stato possibile verificare che la copia ex-novo di tutti i file verso l'area di backup era molto più lenta rispetto a copiare soltanto ciò che era cambiato dal backup precedente.

3.5 Esecuzione asincrona e blocco esecuzioni contemporanee

Per prevenire problemi di interruzione accidentale della connessione da shell remota, da cui l'amministratore potrebbe far partire il backup in modalità manuale, si è pensato di disaccoppiare dalla shell il processo di backup vero e proprio. In particolare, l'esecuzione manuale da shell provvede ad avviare un altro processo asincrono, cioè quello di backup, per poi visualizzare come output solo il file di log prodotto da questo secondo processo. In sostanza, se si chiude accidentalmente la shell o se si interrompe la connessione remota, il processo di backup continuerà ad essere eseguito.

In aggiunta a ciò, si è voluto impedire che per errore fossero avviati più processi di backup contemporaneamente, evitando così potenziali conflitti sulla commutazione dello stato di manutenzione di Moodle, nonché sulla destinazione dei dati di salvataggio. Questo è stato possibile attraverso la gestione di file *pid* e *lock* appositamente prodotti e controllati dallo stesso processo di backup.

4 STRUTTURA E MODALITÀ D'USO DEL SISTEMA DI BACKUP

Il diagramma riportato in Figura 1 mostra la struttura dei file del sistema *Moodle Backup Scripts* [1] e le relative interdipendenze.

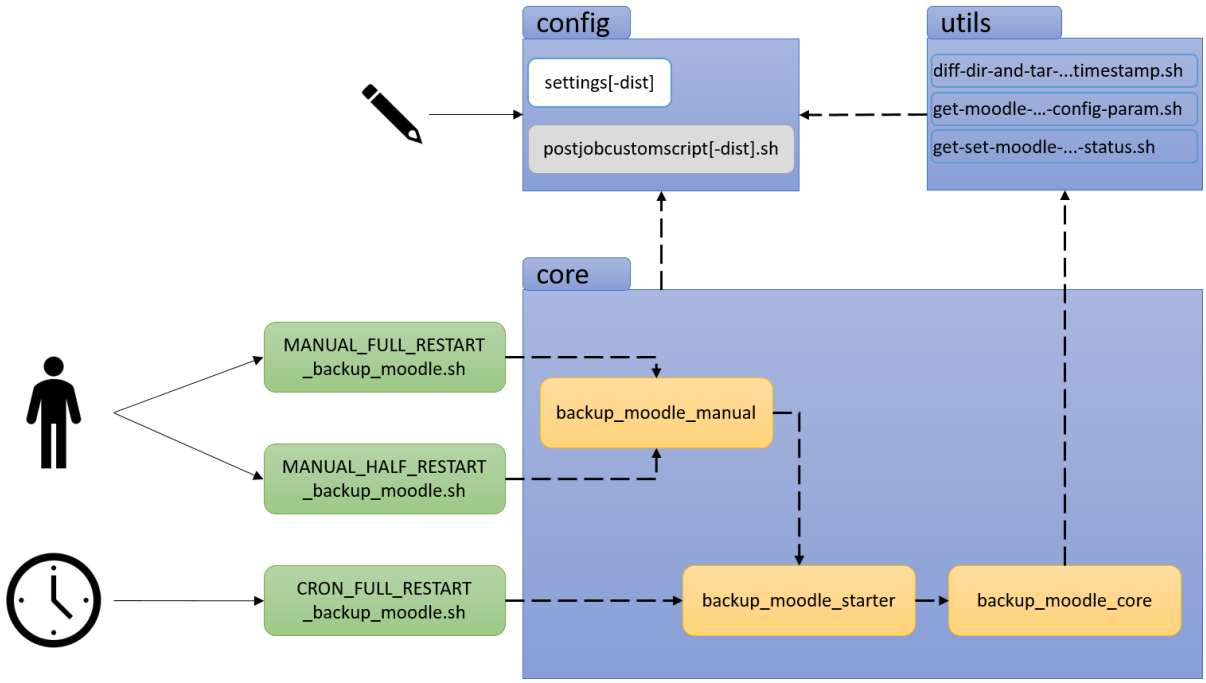

Figura 1 – Dipendenze tra i file bash e attivazione manuale o automatica

I paragrafi di questa sezione faranno riferimento agli elementi personalizzabili ed eseguibili presenti nel digramma. In particolare saranno descritti (1) il file *settings*, per l'impostazione dei parametri di configurazione, (2) il file opzionale *postjobcustomscript.sh*, per aggiungere istruzioni personalizzate da eseguire al termine del backup, (3) i file di backup manuale o automatico – via *cron* – con relativo meccanismo di riavvio del sito Moodle (con prefissi *MANUAL/CRON_FULL/HALF_RESTART* nel diagramma).

4.1 Impostazione dei parametri nel file di configurazione

Il file di configurazione del sistema di backup è *settings*. Tale file non è presente nel repository Git del progetto, in quanto dovrà essere creato dall'amministratore per contenere valori specifici che variano da installazione a installazione, ma è presente un file di esempio, chiamato *settings-dist*. All'interno di esso, i parametri principali sono:

- *backuptargetparentpath* e *backuptargetdirname*: definiscono rispettivamente il percorso genitore e il prefisso del nome (a cui normalmente il sistema aggiungerà il suffisso *-last*) della destinazione del backup, ovvero della directory di salvataggio nel file system locale;

- *listofadditionaldbtodump* e *listofadditionaldbtosave*: definiscono rispettivamente una lista opzionale di nomi di database addizionali rispetto a quello di Moodle, e una lista opzionale di directory da salvare rispetto a quelle contenenti file e codice della applicazione web Moodle;
- *moodlesiteparentpath* e *moodlesitedirname*: definiscono rispettivamente il percorso genitore e il nome della directory della applicazione web Moodle. Se combinati, forniscono il percorso completo della applicazione. Tutti gli altri parametri utili ad individuare i dati di cui eseguire il backup (es. tipo di database, credenziali di accesso, IP locale o remoto), saranno ricavati dal file *config.php* di Moodle presente all'interno del percorso completo;
- *servermoodledaemon* e *serverservicename*: definiscono rispettivamente il nome utente associato al server web, normalmente *www-data*, e il nome del servizio del server, normalmente *apache2* o *nginx*.

4.2 Aggiunta di istruzioni personalizzate

Il file per personalizzare le operazioni da svolgere al termine del backup è *postjobcustomscript.sh*. Tale file non è presente nel repository Git del progetto, in quanto dovrà essere creato dall'amministratore, e la sua esecuzione è opzionale. In assenza del file, il backup verrà fatto soltanto localmente. È comunque presente un file di esempio, chiamato *postjobcustomscript-dist.sh*, contenente la struttura minima di esecuzione.

Personalizzando adeguatamente il file, è possibile fare copie rapide su altre macchine della rete. Oppure è possibile richiamare noti strumenti di backup come Duplicity [6], o RClone [14], per salvare e/o sincronizzare il backup dalla copia locale verso DropBox / Google Drive o altri servizi di archiviazione remoti, sfruttando eventualmente l'archiviazione incrementale e la crittografia di tali strumenti. Ancora, se il dump testuale del database è multilinea, si può pensare di usare un repository Git privato o locale come base per salvataggi incrementali del solo database (mediante *commit*), e via dicendo.

4.3 Avvio del backup

Il backup può essere avviato invocando opportuni script, sia in modalità manuale che in modalità automatica, tramite *cron*.

Avvio del backup in modalità manuale:

- *MANUAL_FULL_RESTART_backup_moodle.sh*: arresta il sito ponendo Moodle in stato di manutenzione, poi esegue il backup locale ed infine ripristina la piena funzionalità del sito, ovvero esce dallo stato di manutenzione;
- *MANUAL_HALF_RESTART_backup_moodle.sh*: arresta il sito ponendo Moodle in stato di manutenzione, poi esegue il backup locale e infine termina senza uscire dallo stato di manutenzione – suggerendo però all'utente, su shell, quali comandi usare per uscirne.

Avvio del backup in modalità automatica:

- *CRON_FULL_RESTART_backup_moodle.sh*: normalmente avviata tramite *cron*, arresta il sito ponendo Moodle in stato di manutenzione, poi esegue il backup locale e successivamente ripristina la piena funzionalità del sito, ovvero esce dallo stato di manutenzione. Infine, se definite, lancia le procedure post-backup per il trasferimento remoto dei dati.

5 REQUISITI E PERFORMANCE DEL SISTEMA DI BACKUP

5.1 Requisiti software

Il sistema Moodle Backup Scripts, a partire dalle prime fasi dei sua costruzione, è stato costantemente collaudato su macchine (virtuali) di test fino a giungere ad una versione sufficientemente matura da utilizzare, in produzione, in macchine con caratteristiche analoghe a quelle di collaudo.

Sotto sono riportate le caratteristiche delle macchine nelle quali è stato verificato un corretto supporto del sistema di backup, ovvero i requisiti software del sistema ospite:

- Distribuzione sistema operativo Linux: Debian (10, 11), Ubuntu (20.04, 22.04)

- Web server: Apache2 (2.4.x), Nginx (1.18)
- DBMS: MySQL/MariaDB (10.5), PostgreSQL (9.6)

È importante sottolineare che le versioni specifiche indicate nei diversi requisiti software sono solo quelle effettivamente verificate ad oggi, ma lo spettro delle compatibilità può essere molto più ampio. Anche per quanto riguarda le distribuzioni Linux, molte altre rispetto Debian/Ubuntu supportano esattamente i comandi di sistema usati negli script di backup ma, per motivi di tempo, non sono state ancora collaudate.

5.2 Misurazione della performance

I gruppi di informazioni riportati in seguito rappresentano le dimensioni dei dati salvati e il relativo downtime – ovvero il tempo di permanenza in stato di manutenzione – di tre differenti siti Moodle su cui è stato eseguito il backup in modalità automatica. I quattro valori riportati in ogni riga riguardano nell'ordine (1) il dump del database, (2) il codice della applicazione web, (3) i file caricati dagli utenti e (4) la somma dei tre precedenti valori.

A. Sito di test con circa 10000 corsi (di cui oltre il 99% vuoti) e 90 utenti registrati

- Dimensione = 14MB + 524MB + 641MB = 1.15GB
- Tempo = 5s + 14s + 1s = 20 secondi

B. Sito mooc in produzione con circa 20 corsi e 24000 utenti registrati

- Dimensione = 232MB + 518MB + 7623MB = 8.18GB
- Tempo = 54s + 16s + 4s = 1 minuto e 14 secondi

C. Sito blended learning in produzione con circa 15000 corsi e 34000 utenti registrati

- Dimensione = 624MB + 520MB + 383751MB = 375.87GB
- Tempo = 152s + 17s + 16s = 3 minuti e 5 secondi

Si noti che la maggior parte del tempo di downtime viene investita per il dump del database, che deve essere prodotto ex-novo ad ogni backup. Questo è ben visibile quando il dump ha una dimensione di qualche centinaio di MB e richiede tempi di creazione dell'ordine dei minuti, mentre l'insieme dei file caricati dagli utenti, pur avendo una dimensione di vari ordini di grandezza superiore, cioè di qualche centinaio di GB, richiede soltanto pochi secondi per il backup. Il motivo principale di questa differenza, sta nel fatto che per il backup dei file utente viene usata la sincronizzazione *rsync*, facendo sì che nella directory locale di destinazione venga modificato solo ciò che è cambiato rispetto al backup (della notte) precedente. Ovviamente, data la natura differenziale della sincronizzazione, i tempi saranno sicuramente più alti durante il primo backup, cioè quando non ci sono salvataggi precedenti a disposizione, oppure in seguito a consistenti variazioni del contenuto dell'area dei file utente.

Ulteriori ottimizzazioni si potrebbero fare sul codice della applicazione web, avente una dimensione pressoché costante e, solitamente, nessuna variazione rispetto al backup precedente – se non dopo un aggiornamento oppure dopo la modifica di impostazioni di configurazione nel solo file *config.php* di Moodle. Attualmente il codice d'origine viene comparato con il contenuto del backup precedente che, per risparmiare spazio, è compresso in un singolo file di archivio *tar.gz* di circa 520MB. Per la comparazione fra i file del codice originale (non compressi) e l'archivio compresso viene usata una operazione più lenta di *rsync*, ed in caso di variazione viene rimosso il vecchio archivio di backup, creandone poi uno nuovo al suo posto. Per ottimizzare i tempi, però, risparmiando così circa 10 secondi o poco più, si potrebbe rinunciare alla compressione del backup del codice e usare *rsync* anche per questa operazione. Oppure si potrebbe definire un parametro di configurazione per lasciare all'amministratore la scelta del giusto compromesso spazio/tempo, cioè per attuare o meno la compressione.

6 CONCLUSIONI

In questo articolo è stato presentato il sistema Moodle Backup Scripts illustrando le ragioni e le scelte progettuali che hanno portato a realizzarlo. Lo strumento è correntemente attivo presso l'Ateneo di Urbino, ma è stato concepito come sistema *general-purpose*, adattabile a varie distribuzioni Linux e a diversi database e web server.

In seguito vengono riassunte le peculiarità e i compromessi del sistema, nonché i potenziali sviluppi futuri di Moodle Backup Scripts.

6.1 Peculiarità

Oltre ad avere una portabilità piuttosto ampia, il sistema è dotato anche di una certa flessibilità, consentendo all'amministratore di selezionare dati e file extra da salvare rispetto a quelli strettamente appartenenti a Moodle, ma anche di personalizzare le azioni da compiere dopo il backup, potendo così utilizzare gli strumenti software più efficaci per il trasferimento dei dati in destinazioni remote.

Il sistema permette inoltre di eseguire backup in modalità manuale, facendo in modo che Moodle venga sospeso solo temporaneamente, oppure che rimanga sospeso anche dopo il salvataggio locale dei dati al fine di intervenire rapidamente per aggiornare l'applicazione web. In alternativa, il backup può essere eseguito in modalità automatica, avviando poi il trasferimento remoto dei dati una volta terminato il backup locale e riattivata la piena funzionalità del sito Moodle.

Rispetto ad altri sistemi di backup selettivo presi in considerazione, tra cui i già citati Duplicity e RClone, il sistema di script è orientato al backup specifico di Moodle e richiede all'amministratore una configurazione molto semplice, eseguendo implicitamente azioni specifiche quali il dump del database – attraverso credenziali e dati già specificati nel file di configurazione *config.php* di Moodle – e la sospensione ed il successivo riavvio di Moodle dallo stato di manutenzione. Inoltre, il backup locale del sistema qui presentato ottimizza quanto più possibile la performance rispetto ad altre applicazioni generiche che offrono, ad esempio, caratteristiche quali il backup incrementale e la crittografia, ma con tempi generalmente più lunghi. Tali applicazioni, come già accennato, possono però essere integrate ed usate proficuamente nello script personalizzabile di post-backup del sistema, per sfruttare le loro specificità per il backup/trasferimento remoto, ma dopo avere rispristinato la piena funzionalità del sito Moodle.

6.2 Compromessi

Per la realizzazione del sistema, è stato necessario accettare dei compromessi. Il primo di questi è il fatto che Moodle deve essere sospeso, seppur temporaneamente. Ovvero deve essere posto in stato di manutenzione durante il backup locale. Questa sospensione può creare disagio agli utenti, ma permette di salvare le informazioni del database e i file caricati dall'utente evitando i problemi di inconsistenza sui dati.

Il secondo compromesso riguarda la necessità di disporre di un disco, o di una partizione di disco, da dedicare permanentemente al salvataggio e alla sincronizzazione dei file. La capacità richiesta dovrebbe essere, almeno, quanto la somma della dimensione del dump del database, più quelle dei file caricati in Moodle e del codice della applicazione, oltre alla dimensione degli eventuali file e dump di database extra. Più, ovviamente, un certo margine di crescita previsto per i vari contenuti.

6.3 Sviluppi futuri

Oltre le possibili ottimizzazioni della performance discusse nel dettaglio in una sezione precedente di questo articolo, sono previsti alcuni sviluppi futuri ed estensioni che contribuirebbero ad aumentare l'usabilità e la portabilità del sistema di backup.

Una delle estensioni previste riguarda la verifica preliminare della presenza dei comandi di sistema. Ovvero, prima di cominciare ad eseguire il backup, i vari script di avvio manuale o automatico dovrebbero verificare se tutti i comandi che verranno usati in seguito sono presenti e installati nel sistema operativo. In caso negativo, il processo di backup non dovrebbe avviarsi neppure ed il problema dovrebbe essere segnalato su console e su log di output. Ovviamente la verifica non sarà applicabile ai comandi definiti dall'amministratore all'interno del file *postjobcustomscript.sh*, non potendo questi essere noti a priori.

Si potrebbe anche definire un *installer*, ovvero un file di script da eseguire una-tantum, che avrebbe il duplice scopo di installare i comandi di sistema richiesti, qualora non presenti, nonché di creare il file *settings* raccogliendo i valori dei parametri mediante analisi dell'host e chiedendo conferma all'amministratore.

Al contempo, anche se ciò potrebbe comportare dei rischi rispetto al ripristino manuale dei dati di backup desiderati, eventualmente residenti in una locazione remota, si può pensare di definire uno script per automatizzare il ripristino [11] eseguendo il processo inverso del backup per l'ultimo

salvataggio eseguito, andando cioè a recuperare i file e i dati dall'area di backup locale e sostituendoli a quelli attualmente utilizzati da Moodle.

Un altro sviluppo possibile è quello di creare una libreria di stringhe multilingua, in modo da internazionalizzare i messaggi prodotti su shell e su file di log.

Infine, occorrerebbe estendere lo spettro delle distribuzioni Linux in grado di supportare il sistema di backup, verificandone di nuove rispetto a quelle già collaudate.

Riferimenti bibliografici

[1] E. Bontà, *Moodle Backup Scripts*, https://github.com/uniurbit/moodle-backup-scripts

[2] Debian, *Backup and Recovery*, https://wiki.debian.org/BackupAndRecovery

[3] Debian, *Details of package backupninja*, https://packages.debian.org/en/stable/backupninja

[4] Debian, *Details of package rsync*, https://packages.debian.org/en/stable/rsync

[5] Docker, https://www.docker.com/

[6] Duplicity, https://duplicity.us/

[7] Kubernetes, https://kubernetes.io/

[8] MoodleBox, https://moodlebox.net/

[9] MoodleDocs, *Maintenance Mode*, https://docs.moodle.org/en/Administration_via_command_line#Maintenance_mode

[10] MoodleDocs, *Site backup*, https://docs.moodle.org/en/Site_backup

[11] MoodleDocs, *Site restore*, https://docs.moodle.org/en/Site_restore

[12] Nakivo, *Crash-Consistent vs. Application-Consistent Backup*, https://www.nakivo.com/blog/crash-consistent-vs-application-consistent-backup/

[13] G. Pardon, C. Pautasso and O. Zimmermann, *Consistent Disaster Recovery for Microservices: the BAC Theorem*, in IEEE Cloud Computing, vol. 5, no. 1, pp. 49-59, Jan./Feb. 2018

[14] RClone, https://rclone.org/

[15] Veeam, *Kubernetes Backup*, https://www.veeam.com/kubernetes-native-backup-and-restore.html

[16] Veeam, *What is Disaster Recovery?*, https://www.veeam.com/blog/what-is-disaster-recovery.html

[17] Wikipedia, *Continuos Data Protection*, https://en.wikipedia.org/wiki/Continuous_Data_Protection

[18] Wikipedia, *Data Consistency*, https://en.wikipedia.org/wiki/Data_consistency

[19] Wikipedia, *Disaster Recovery*, https://en.wikipedia.org/wiki/Disaster_recovery

[20] Wikipedia, *Shibboleth sign-on architecture*, https://en.wikipedia.org/wiki/Shibboleth_Single_Sign-on_architecture

[21] Wikipedia, *Single-board computer*, https://en.wikipedia.org/wiki/Single-board_computer

[22] Wiktionary, *set-and-forget*, https://en.wiktionary.org/wiki/set-and-forget