Iaas/Saas computing, storage

Diego Casella, Fabio Marcuzzi
February 24, 2015

Università
degli Studi
di Padova

Contenuti

Introduzione

Quest'oggi parleremo dei *Iaas/SaaS*, e di altri paradigmi *"as a Service"* che sono sorti negli ultimi anni in ambito Web e Cloud.

Paradigma "*as a Service*"

Negli ultimi anni sono emersi diversi tipi di paradigmi "*as a Service*", differenziatisi per contradistinguere il servizio offerto all'utente fruitore di esso. I paradigmi "*as a Service*" piú conosciuti sono:

- *Software as a Service*;
- *Platform as a Service*;
- *Infrastructure as a Service*;

Paradigma "*as a Service*" - SaaS

Nel *Software as a Service* l'utente ottiene un SO perfettamente configurato e settato per poter funzionare col software con il quale intende lavorare. Eventuali update, sia del software che dell'hardware, settaggi e configurazioni vengono realizzati dal fruitore del servizio.

Esempio

Wordpress fornisce un hosting gratuito per poter realizzare il proprio blog: gli utenti utilizzano l'istanza di Wordpress, senza doversi preoccupare di effettuare updates dei software correlati (PHP o MYSQL), di migrare i contenuti in webserver piú capienti ecc...

Paradigma "*as a Service*" - PaaS

Il *Platform as a Service* fornisce ai clienti un un ambiente predefinto, la "Platform", nella quale essi possono sviluppare e rilasciare il proprio software in maniera efficiente e veloce.

Esempio

Google App Engine.

Paradigma "*as a Service*" - IaaS

Nell' *Infrastructure as a Service* gli utenti mettono mano alle macchine stesse - server, storage, networks - fornite sotto forma di virtual environment, nei quali possono settare gli OS desiderati e internconnetterle come desiderano.

Esempio

Servizio Amazon Web Services.

Integrazione di strumenti di terze parti

Facendo riferimento allo IaaS, vediamo alcuni strumenti fondamentali per corredare il SaaS di servizi fondamentali e rendere la user experience quanto piú completa ed efficace possibile, in particolare:

- accounting dell'utilizzo del servizio;
- gestione dei dati utente (WinSCP va bene, ma si puó fare di meglio);

Accounting

Il primo strumento presentato, sebbene sia quello meno visibile all'utente, é quello che gestisce l'accounting ed in generale i dati sensibili di un determinato utente come le credenziali di login, i dati sulla contabilizzazione e la cronologia dell'utilizzo del software. Tali informazioni devono essere mantenute confidenziali, e tale segretezza va quanto piú possibile mantenuta anche in caso di falle del sistema da parte di agenti esterni.

Accounting - Windows Azure

A tal proposito, pensare di salvare tale informazioni direttamente nel profilo utente rappresenta una scelta davvero poco oculata, per molteplici ragioni:

- per implementare un sistema di storage *in-house* sicuro e robusto occorre impegnare risorse, intese sia come sviluppatori, che come tempo;
- esistono giá servizi online che offrono questo tipo di funzionalitá e, essendo specializzate in questo ambito, sono molto piú sicure ed affidabili di quanto si possa pensare di ottenere con una soluzione *in-house*;

Per queste serie di motivazioni, é ragionevole utilizzare ad esempio il framework *Windows Azure*, che consente di tenere sicure e riservate le informazioni utente in uno *storage cloud* separato dal server dove vengono eseguite le applicazioni.

Storage

Lo storage rappresenta un altro punto focale nell'uso del *software as a service*: l'utente deve essere in grado di inviare e ricevere dal proprio computer al server cloud, in maniera semplice ed immediata, vari tipi di files:

- documenti di testo (es. /LaTeX),
- intere cartelle (progetti firmware, testbench, ecc...)
- script Python, sorgenti C, ...

Storage - WinSCP

Strumenti come WinSCP sono utili per trasferire files da un computer locale ad un server remoto ma, essendo tool generici, mancano di alcune caratteristiche fondamentali per il SaaS, come:

- la sincronizzazione automatica dei files;
- la possibilitá di gestirne il versioning.

Inoltre, un progetto non viene mai sviluppato da una singola persona ma da un gruppo di lavoro, e dunque é necessario che gli elementi costitutivi vengano raccolti in un repository comune, al quale tutti abbiano accesso, da qualsiasi postazione essi accedano.

Storage - DropBox

Viste le considerazioni precedenti, un ottimo software che ben si presta allo scopo é DropBox. Esso infatti fornisce:

- servizio di file synchronization;
- servizio di file versioning;
- developer API disponibile in Java, utile per integrare le sue funzionalitá direttamente nell'applicazione offerta in modalitá SaaS.

Esempio - Autenticazione

```
private static final String APP_KEY = "INSERT_APP_KEY";
private static final String APP_SECRET = "INSERT_APP_SECRET";
DbxAppInfo appInfo = new DbxAppInfo(APP_KEY, APP_SECRET);
DbxRequestConfig config = new DbxRequestConfig(
"com.casella.testapp", "");
DbxWebAuthNoRedirect webAuth = new
DbxWebAuthNoRedirect(config, appInfo);
String authorizeUrl = webAuth.start();
// copy the url, get the auth code
DbxAuthFinish authFinish = webAuth.finish(code);
DbxClient client = new DbxClient(config, authFinish.accessToken);
```

Storage - DropBox API

DbxClient API

DbxEntry copy(String fromPath, String toPath);
DbxEntry.Folder createFolder(String path)
void delete(String path)
DbxEntry.File getFile(String path, String rev, OutputStream target)
DbxEntry.WithChildren getMetadataWithChildren(String path)
DbxClient.Downloader startGetFile(String path, String revision)
DbxClient.Uploader startUploadFile(String targetPath, DbxWriteMode writeMode, long numBytes)

Esempio - Visualizzazione File Remoti

```
DbxEntry.WithChildren listing = client.getMetadataWithChildren("/");
for (DbxEntry child : listing.children) {
    System.out.println( child.name + ": " + child.toString() );
}
```

Storage - EverNote

Un sottoinsieme particolare del servizio di storage é rappresentato dallo storage di appunti o note, che possono essere sincronizzate e condivise fra piú client. A volte infatti, si pensi per esempio a CfL, condividere file di testo é piú che sufficiente, e l'utilizzo di DropBox in tal caso risulta piú complicato del necessario. Per questa ragione si puó pensare di integrare dei servizi di notetaking, come ad esempio il famoso EverNote che offre molte funzionalitá interessanti fra le quali

- servizio sincronizzazione delle note;
- annotazione su PDF;
- riconoscimento scrittura a mano;
- developer API disponibile in Java, utile per integrare le sue funzionalitá direttamente nell'applicazione offerta in modalitá SaaS;

Storage - EverNote API

Esempio - Autenticazione

```
private static final String AUTH_TOKEN = "your developer token";
EvernoteAuth evernoteAuth = new
EvernoteAuth(EvernoteService.SANDBOX, AUTH_TOKEN);
ClientFactory factory = new ClientFactory(evernoteAuth);
NoteStoreClient noteStore = factory.createNoteStoreClient();
```

Esempio - Lettura Note Esistenti

```
List<Notebook> notebooks = noteStore.listNotebooks();
for (Notebook notebook : notebooks) {
    System.out.println("Notebook: " + notebook.getName());
    NoteFilter filter = new NoteFilter();
    // settaggio filter
    ...
    // cerca le prime 100 note corrispondenti al filtro indicato
    NoteList noteList = noteStore.findNotes(filter, 0, 100);
    for (Note note : notes) {
        System.out.println( note.getTitle() );
     }
}
```

Storage - Ricapitolazione

Abbiamo visto finora come sia possibile rendere semplice la sincronizzazione di file, documenti e progetti, che si trovano in piú locazioni differenti, all'interno dello spazio di lavoro presente nel server cloud. Quello che non abbiamo ancora visto é se sia possibile, ed in tal caso come attuare, una gestione *nel server* del progetto in maniera indipendente da dove esso sia arrivato, fornendo la possibiltá di creare degli *snapshot* dello stato del progetto ogni qual volta l'utente senza il bisogno di salvare lo stato del progetto in un determinato punto del suo sviluppo.

Versioning

Come conseguenza della considerazione fatta poco fa, si capisce che non é importante solo poter inviare, ricevere, sincronizzare files da piú computer remoti al server cloud; é anche necessario fornire un meccanismo di versioning, all'interno del server, che renda possibile memorizzare il progetto ed i suoi files costituenti, a discrezione dell'utente.
Per questa ragione si puó pensare di utilizzare Subversion, Git o altri sistemi di version control, e di integrarli nell'applicazione offerta in modalitá SaaS, per poter offire una interfaccia semplificata e funzionale all'utente.

Q&A