Università degli Studi dell'Aquila
Facoltà di Ingegneria

Tesi di Laurea Specialistica in Ingegneria Informatica e Automatica

Realizzazione di un prototipo della versione Cloud SaaS della suite IBM BigFix: Automazione del deployment e del testing

Relatore interno:
Prof. Serafino Cicerone

Laureando:
Beniamino Negrini

Correlatore:
Prof. Dott. Marco Secchi

Relatore esterno:
Dott. Bernardo Pastorelli

Anno Accademico 2016/2017

Dedica a piè pagina

Indice

Capitolo 1

Introduzione

E' sempre più evidente che il cloud computing è il futuro del software. La rivoluzione consta nella distribuzione dei servizi di calcolo e alla virtualizzazione delle risorse. Viene data così all'utente la sensazione di un utilizzo centralizzato. Tutto ciò si è reso realizzabile dal momento in cui l'accesso alla rete è divenuto possibile da sempre più dispositivi con velocità di connesione sempre maggiore.

la tematica del cloud computing è stata centrale nel mio lavoro di tesi presso l'azienda IBM (International Business Machines Corporation), nella sua sede di Roma. Ho partecipato attivamente alla realizzazione di un prototipo software, ossia la versione SaaS della suite IBM BigFix. BigFix è una suite di prodotti dedicati alle aziende che risolvono problematiche di Endpoint Security e di compliance di dispositivi interni a determinate politiche aziendali. Tramite questi prodotti si ottiene pieno controllo su tutti i dispositivi aziendali, si rilevano eventuali attacchi, si distribuiscono aggiornamenti e patch, fino anche alla gestione energetica.

La sfida da me raccolta è quindi proprio quella di portare tutto questo arsenale di strumenti nella leggerezza del cloud. Rendendolo disponibile, nel giro di pochi minuti, anche a chi è sempre stato intimorito dalla difficoltà di installazione di uno strumento così potente, ma allo stesso tempo complesso.

Capitolo 2

IBM BigFix

2.1 BigFix

I prodotti della suite IBM BigFix consentono di monitorare e gestire in tempo reale un elevato numero di dispositivi fisici e virtuali connessi (fino a 250.000). Questi possono essere sia fisici che virtuali, come ad esempio server, desktop, notebook, dispositivi mobili, tablet, POS, ATM, chioschi self-service. Gli utenti principali di questi prodotti sono gli amministratori di sistema. Tramite le applicazioni BigFix possono avere il pieno controllo sugli endpoint, come distribuire software, applicare delle patch, effettuare il deploy di sistemi operativi, proteggere da attacchi di rete e molto altro.

2.1.1 Architettura

Un'architettura di BigFix è, per sua natura, molto articolata, poichè la necessità e quella di gestire un numero elevato ed eterogeneo di dispositivi. Essa si basa sul consolidato pattern stilistico Client/Server, ma con una struttura leggermente variata, prevedendo l'inserimento di un ulteriore layer frapposto tra client e server, i relay, i quali sono fondamentali per bilanciare il carico.

Ma partiamo subito con un esempio per avere un ponto di riferimento. Come possiamo notare, l'elemento fondamentale è il server, il quale ha lo scopo di racco-

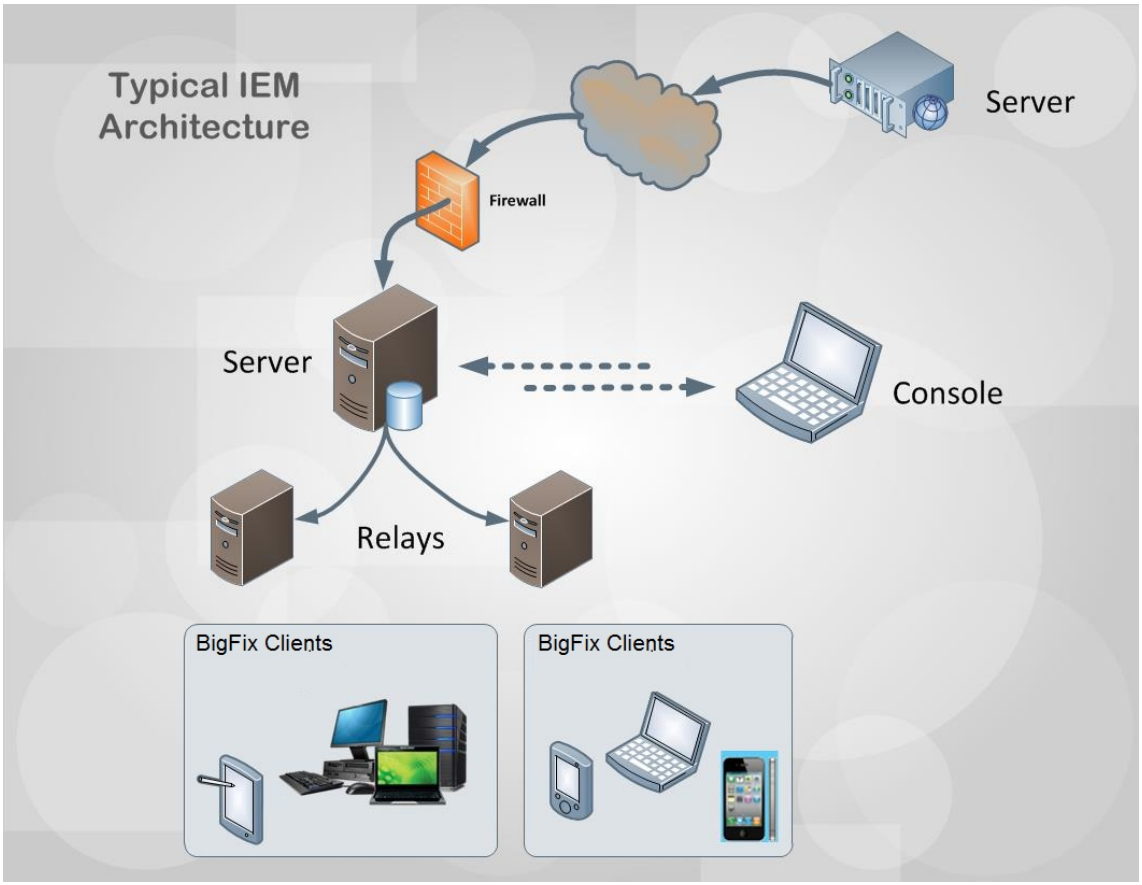

Figura 2.1: Un'architettura BigFix di esempio

gliere dei particolari messaggi, le Fixlet. Questi messaggi posso essere visualizzati dall'operatore che lavora sulla console e inoltrati quindi, a questo punto ai relay. E' competenza dei relay poi interagire con i singoli client e assicurarsi l'esecuzione delle Fixlet. Le Fixlet, infatti, altro non sono che delle azioni che devono essere necessariamente compiute dai client. Andiamo ad analizzare le singole componenti dell'archiettura.

Servers Il server coordina tutto il flusso di informazioni e si preoccupa di salvare le informazioni sul database. Al tempo stesso però, lascia agli Agent il compito di effettuare analisi ed eseguire azioni specifiche. Ciò consente di liberare il Server da un pesantissima computazione. Per questo motivo il Serve stesso può gestire al altissimo numero di client.

Relays I Relay si comportano come una cache tra i Client e il Server e sono di numero variabile in base al numero di Client. Aiutano il Server a gestire i dispositivi anche se funzionalmente non sono altro che Client che sono stati promossi a Relay, aggiungendo a loro dei servizi. A questo punto i Client non si interfacceranno mai con il Server, alleggerendone così notevolmente il workload. Possono, ad esempio, più Client richiedere un download al Relay, il quale effettuerà un'unica richiesta al Server

Agents Un Agent è installato su ogni Client facente parte dell'architettura di BigFix. essi hanno il compito di raccogliere le Fixlet, tramite le quali sono in grado, ad esempio, di individuare e correggere le exposure di sicurezza, configurazioni scorrette e altre vulnerabilità. Fa dei continui check per confrontare lo stato del dispositivo con le policy stabilite. Appena scopre che il dispositivo è fuori dalla compliance, viene informato il Server ed agisce subito per porre rimedio e, al termine, informa nuovamente sull'esito dell'operazione.

Web Reports I Web Reports sono il componente che consentono ad utenti autorizzati di monitorare tutti i dispositivi di BigFix. SI piò in questo modo tenere traccia di vulnerabilità, azioni richieste e molto altro.

Consoles La Console permette agli amministratori di interagire con tutti i Client dell'ambiente BigFix. Gli utenti possono così distribuire velocemente patch e configurazioni.

BigFix, da un punto di vista logico, si suddivide in due grandi macro-componenti, la Platform e le Applications. La prima svolge la funzione di layer sulla quale vengono sviluppate tutte le funzionalità dello strato di applications. Questa suddivisione consente una chiara suddivisione delle competenze da parte di progettisti, sviluppatori, tester e assistenti dei clienti. Il team della platform si concentra quindi nel fornire una solida infrastruttura al team delle applications, il quale svilupperà i singoli strumenti al servizio dell'utente.

2.1.2 BigFix Platform

La Platform è una tecnologia multi-layer scritta in linguaggio C++ che agisce come colonna portante di tutta l'infrastruttura di BigFix. La Platform svolge infatti funzioni fondamentali, spesso utilizzate anche da altre applicazioni dei layer superiori. La Platform divide le responsabilità.

Lifecycle Management Include il controllo remoto, la distribuzione dei softawre e il deploy di sistemi operativi.

Patch Management Consente l'applicazione e la gestione delle patch.

Core Protection E' il cuore di tutta la routine di Security. Troviamo quì delle funzionalità anti-malware, firewall e protezione da qualsiasi variante di virus.

Inventory Raccoglie informazioni peculiari a riguardo dei software installati su ogni dispositivo. E' in grado di effettuare analisi sugli utilizzi fornendo una base per la decisione sulle licenze da acquistare.

Server Automation Fornisce un Hypervisor per le Virtual Machines. Esso controlla infatti eventuali malfunzionamenti degli ambienti virtuali.

2.1.3 BigFix Applications

Tutti i prodotti applicativi che fanno parte di questo componente consentono di gestire in maniera semplice tutte le operazioni inerenti alla security. A differenza della Platform sono implementate in linguaggio Java ed hanno funzionalità atomiche tra di loro. Sono l'interfaccia principale con il quale interagisce l'amministratore aziendale.

2.1.4 Fixlets

Le Fixlet sono il metodo attraverso il quale si svolgono tutte le operazioni come distribuzione di software, installazioni di patch e configurazioni. Esse sono dei mes-

saggi inoltrati ai client di BigFix e utilizzano un linguaggio di query specifico, il Relevance.

Il linguaggio Relevance

Con una Fixlet si può anche ispezionare un desiderato aspetto di un client. Il linguaggio di Relevance consente di interrogare il client identificandone caratteristiche dell'hardware o del software tramite gli Inspectors. Una necessità può essere infatti quella di applicare una Fixlet solamente a dei client con determinate caratteristiche hardware/software oppure che si trovano in stati ben definiti. Si può in questo modo, facilmente identificare il corretto sottoinsieme di client ai quali è destinata una nuova Fixlet ed applicarla solo ad essi.

2.2 IBM

Il lavoro di tesi si è svolto nell'ambito di un progetto formativo stipulato tra l'Università dell'Aquila e IBM Italia Spa. Questo progetto ha previsto un tirocinio svolto nella seda di Roma con obiettivo: "Esplorazione e prototipazione di metodi per portare prodotti BigFix su cloud", per l'appunto la realizzazione del prototipo di BigFix Saas.

La storia dell'IBM ha inizio nei primi decenni del novecento, ma è dagli anni settanta che entra nel mercato dell'informatica, soprattutto nel settore hardware. Negli ultimi venti anni il business si è spostato sempre più sul software. In particolare soluzioni cognitive e piattaforme cloud.

IBM Security Presso la sede di Roma presente il più importante laboratorio IBM italiano, il Rome Softare Lab. Nella divisione italiana ci si concentra prevlentemente sullo sviluppo back-end. Una grossa fetta del laboratorio si occupa della divisione Security di IBM. Il portfolio di Security contiene prodotti che si occupano di diversi aspetti della security aziendale, tra questi BigFix è uno dei pù consolidati.

2.3 Saas Exploration Project

Lo scopo del progetto al quale ho partecipato con il mio lavoro è quello di esplorare le tecnologie esistenti e realizzare il prototipo della versione SaaS di BigFix. A questo scopo, oltre a me, sono state allocate altre tre persone full time al conseguimento del progetto, sotto la guida dell'architect Bernardo Pastorelli.

2.3.1 Il framework SCRUM

Il team adotta il framework agile SCRUM. Questo modo di operare è di sempre maggiore diffusione ed è basato su un approccio iterativo e incrementale nello sviluppo software. Il design e lo sviluppo sono divise in iterazioni, denominate "sprint", della durata fissa di due settimane. Queste due settimane terminano sempre con una versione funzionante del prodotto, il quale viene mostrato in una demo che ne evidenzia le nuove features implementate.

SCRUM utilizza un approccio empirico alla progettazione. La filosofia di fondo del framework quella che la conoscenza deriva dall'esperienza, e quindi tutte le scelte che si prendono nel corso della progettazione devono avvenire alla luce di una sempre maggiore esperienza, la quale si ottiene avendo a disposizione il prima possibile un sottoinsieme del prodotto testabile ed usabile a se stante. Dì quì l'approccio fortemente iterativo e incrementale massimizzando le opportunità di feedback.

All'inizio del progetto vengono definiti i requisiti del prodotto (item), i quali vengono da un'attenta analisi dei bisogni dell'utente. Ogni bisogno viene modellato con una Epica, che a sua volta viene prioritizzata e aggiunta al product backlog che le indicizza. Le Epiche vengono poi scomposte in User Stories, le quali si suddividono a loro volta in Task, ossia l'elemento atomico del progetto la cui implementazione viene presa in carico da un singolo componente del team.

L'inizio di uno sprint è sempre caratterizzato da un meeting in cui si pianificano gli obiettivi per il prossimo sprint. In questo contesto si fa sempre riferimento al backlog incentrandosi sulle user stories ancora non coperte. Si cerca quindi di suddividersi i task in modo tale da avere a fine sprint quelle nuove funzionalità

usabili e dimostrabili. Demo che viene svolta sempre con la presenza di tutto il team e anche dei colleghi americani.

2.3.2 Sistemi di controllo di versione

Da un'organizzazione di questo tipo ne scaturisce la necessità di tool di controllo di versione che permettano una fluida gestione del codice e della programmazione in parallelo tra i diversi componenti del team.

GitLab A tal scopo si è adottato ormai da tempo, da tutto il team di BigFix, il software di controllo di gestione distribuito Git e una repository aziendale che consiste in una versione enterprise ad hoc per IBM di GitLab, un hosting service molto simile a GitHub.

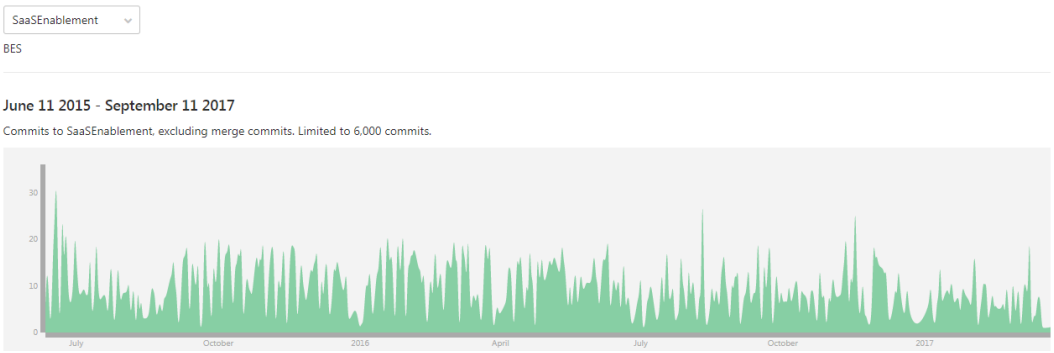

Figura 2.2: Panoramica degli ultimi contributi nel branch del progetto SaaS

Il flusso di lavoro è il seguente. Quando inizia il proprio task, il componente del team si pone su un proprio branch personale sul quale effettua i propri commit. Al termine del task viene fatta una merge request sul branch principale, sono una volta che si è testato il codice, per aggiungere i propri contributi al progetto. A questo punto, dopo una review effettuata da componenti del team accreditati, il nuovo branch verrà mergiato con il branch principale.

2.3.3 RTC

E' ovviamente necessario un tool che coordini anche la suddivisione dei task all'interno del team. A tal fine abbiamo utilizzato un software interno di IBM, ovvero Rational Team Concert (RTC). Esso offre comodi strumenti di agile planning e gestione di ciclo di vita del software. Ogni componente può così tracciare facilmente le aree di sua competenza. E' possibile inoltre usufruire di tool per il source control, controlo dei difetti e gestione delle build.

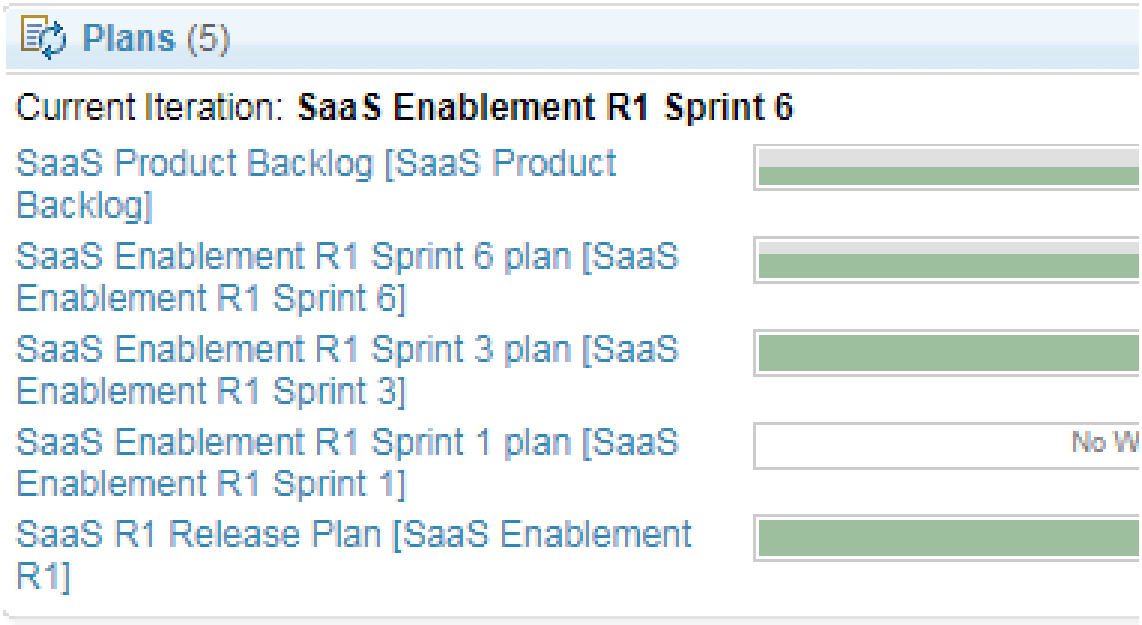

Figura 2.3: RTC. Un'esempio di come viene monitorato il completamneto dei diversi sprint

Capitolo 3

Software as a Service

3.1 Cloud Computing

La differenza tra il possedere e l'utilizzare. E' questo l'aspetto cruciale del cambiamento apportato dal cloud computing rispetto al software tradizionale. Le risorse, che siano esse stesse archiviazione, elaborazione o qualsivoglia risorsa informatica, non sono mai ad hoc per un singolo utente, ma vengono assegnate on demand ai singoli utenti e appartengono ad un insieme condiviso da tutti gli altri utenti del prodotto utenti. Attraverso internet ogni utente può accedere a queste risorse in qualsiasi momento. Tali risorse vengono opportunamente allocate all'utente in maniera dinamica e completamente automatizzata. L'utente può utilizzare così anche software non installati sul proprio computer o usufruire di una memoria di massa accessibile da parte sua da qualsiasi dispositivo.

L'esperienza utente che si vuole fornire però è quella di un utilizzo esclusivo della risorsa, come nei software tradizionali, quando in realtà la risorsa viene solo sapientemente distribuita tra gli utenti. Ciò fa si che, potenzialmente, un singolo utente possa acquisire risorse notevolmente maggiori nel caso medio

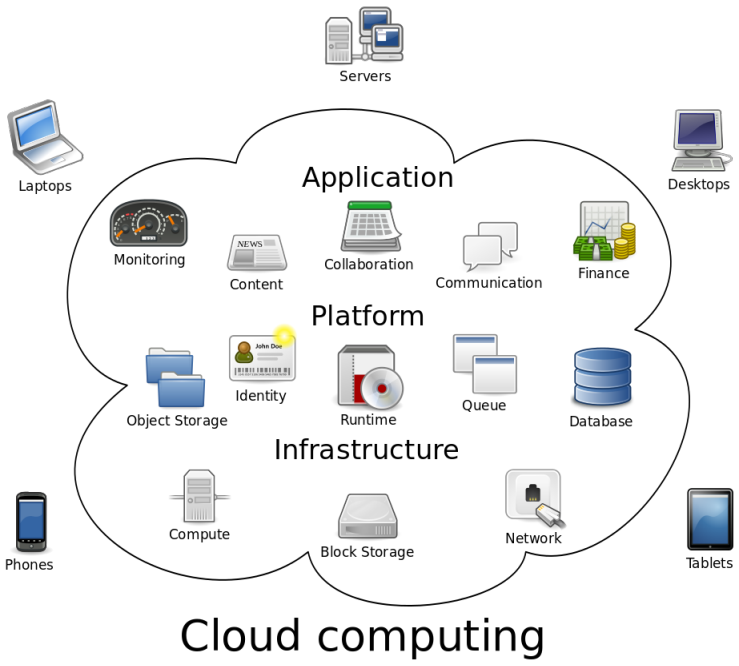

Figura 3.1: Diagramma logico di una rete Cloud Computing

3.1.1 Vantaggi del Cloud Computing

- Costo:
  Con l'avvento del cloud tutta la gestione dell'infrastruttura sottostante il software diviene a carico del provider. Vengono eliminate spese per la gestione dei data center locali. Facendo riferimento alla versione SaaS di BigFix ad esempio, il cliente viene sollevando dal pesante onere di utilizzare server locali e gestirne le relative connessioni. Il provider detiene tutto l'hardware di cui il cliente ha bisogno.

- Velocità:
  Anche quì ci risulta molto utile prendere ad esempio la suite di BigFix. quando un nuovo cliente acquista il prodotto nella sua versione on-premises, un'incaricato di IBM si reca presso il cliente e lì inizia un lungo processo di installazione della suite che può impiegare diverse ore. nella scenario SaaS il cambiamento è radicale. E' sufficente che il cliente compili una form online, dopo alcuni minuti poi riceve una mail con il link per accedere al servizio.

- Prestazioni
  Una delle motivazioni principali per la quale si sceglie di fare uso del clud computing sono proprio le prestazioni, soprattutto secondo il paradigma PaaS. Esternalizzando le risorse di calcolo, si può fare affidamento a dei provider che fanno dei server ad alte prestazioni il loro punto di forza. L'utente può in questo modo abbattere dei bottleneck che altrimenti risulterebbero di grande impedimento. Un esempio estremo in questo senso è la possiblità di utilizzo per la priama volta di un processore quantistico, proprio attraverso una piattaforma cloud.

- Affidabilità
  Operazioni di mirroring da parte dei provider dei servizi cloud fa sì che il backup dei dati sia continuo ed economico.

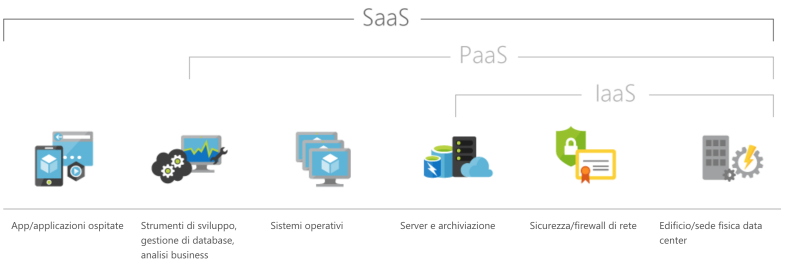

Figura 3.2: Panoramica delle principali tipologie Cloud

3.2 Tipologie di servizi Cloud

Il termine Cloud risulta in realtà molto generico. Esso comprende diverse tipologie di fornitura dei servizi, a seconda della risorsa che viene offerta dal provider. La maggior parte dei servizi di Cloud Computing rientrano in tre tipologie principali: Infrastruttura distribuita come Servizio (IaaS, Infrastructure as a Service), Piattaforma distribuita come Servizio (PaaS, Platform as a Service) e Software come un Servizio (SaaS, Software as a Service). Oltre a queste tipologie, annoveriamo anche soluzioni minori come il DaaS (Data a a Service) e l'HaaS (Hardware as a Service). Andiamo a vedere nel dettaglio come, a seconda della modalità di utilizzo del paradigma Cloud, queste tipologie si differenziano.

3.2.1 IaaS, Infrastructure as a Service

E' la tipologia più basilare. Vengono messe a disposizione piattaforme di elaborazione. Utilizzando un IaaS si affittano le infrastrutture utili ai propri scopi, come add esempio server, macchine virtuali (VMs), risorse di archiviazione, reti e sistemi operativi. Può, inoltre, essere messo a isposizione anche hardware in remoto. Il provider di servizi cloud gestisce l'infrastruttura, mentre l'utente acquista, installa, configura e gestisce il software, tra cui sistemi operativi, middleware e applicazioni.

3.2.2 PaaS, Platform as a Service

Una piattaforma distribuita come servizio (PaaS, Platform as a Service) è un ambiente cloud di sviluppo completo. Una soluzione PaaS è progettata per consentire il ciclo completo dello sviluppo delle applicazioni: creazione, test, distribuzione,

gestione e aggiornamento. L'utente ha tutta la libertà di sviluppare gli applicativi a proprio piacimento, ma lavora con componenti software già pronti all'utilizzo (microservices). Questi componenti non sono localizzati presso l'utilizzatore, bensì presso il provider, il quale si occupa del loro mantenimento e aggiornamento.

IBM Bluemix

3.2.3 SaaS, Software as a Service

3.3 Confronto tra SaaS e On Premise

Vantaggi del Software On Premise

- Controllo esclusivo su sistemi e dati
- Gestione interna dei dati sensibili
- Alto investimento iniziale ammortizzato nel lungo periodo

Il paradigma di fornitura on premise risulta ancora essere la soluzione più adatta nel caso in cui la gestione diretta dei dati sia fondamentale per policy aziendali oppure sia necessaria una maggiore flessibilità di configurazione per l'integrazione con altre architetture software. Un'altro requisito è che l'architettura fisica del software sia geograficamente localizzata.

Appendice A

Tecnologie Utilizzate(template di prova - ANCORA DA SCRIVERE)

A.1 Linguaggi di programmazione

- PHP 5.4.7
  http://www.php.net/;
- Javascript
  http://www.w3.org/standards/webdesign/script;

A.2 Linguaggi di Markup e Stile

- HTML4/HTML5;
- CSS/CSS3;

A.3 Framework

- Smarty Template Engine
  http://www.smarty.net/;

- JQuery
  http://jquery.com/;
- JQueryUI
  http://jqueryui.com/;
- beContent
  http://www.becontent.org/;

A.4 Ambiente di Sviluppo

A.4.1 Eclipse

Per Eclipse sono state utilizzate due versioni differenti, la 4.2.2 in ambiente Windows e la 3.8.0 in ambiente Ubuntu/Linux
http://www.eclipse.org/
Inoltre è stato utilizzato il pacchetto

- PHP Development Tools 3.1.1
  http://projects.eclipse.org/projects/tools.pdt;

A.4.2 Piattaforma Web

XAMPP

http://www.apachefriends.org

- Apache Web Server ver. 2.4.3
  http://httpd.apache.org/;
- MySql Database Management System ver. 5.5.27
  http://dev.mysql.com/;

A.4.3 Browser Testing

Mozilla Firefox

- Firebug ver 1.11.2
  http://getfirebug.com/
  - Plug-In Validator ver. 0.0.6
    https://addons.mozilla.org/it/firefox/addon/validator/;
  - Plug-In Google Page Speed ver. 2.0.2.3
    https://developers.google.com/speed/pagespeed/?hl=it-IT;

Google Chrome

- Strumenti per gli sviluppatori integrati

Responsive Testing

- Viewport Resizer- Responsive Design Bookmarklet
  http://lab.maltewassermann.com/viewport-resizer/ ;

Dedica a fine pagina