UNIVERSITÀ DEGLI STUDI DI FIRENZE
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica

Tesi di Laurea

IL PROGETTO INDICO KT

MIGLIORARE L'IMPATTO A LIVELLO MONDIALE DI INDICO

TOMMASO PAPINI

Relatore: *Pierluigi Crescenzi*

Anno Accademico 2014-2015

ABSTRACT

Nel 2014 il CERN (Conseil Européen pour la Recherche Nucléaire) ha celebrato i suoi *60 anni* di attività.

Dalla sua fondazione, nel lontano Settembre 1954, il CERN è stato protagonista di una serie di scoperte e innovazioni in svariati settori scientifici, informatica e fisica in primis, e si è affermato come uno tra i più importanti centri di ricerca in Europa e nel mondo. Il CERN è infatti il più grande centro di ricerca di fisica nucleare in Europa, possessore, al momento, del più grande acceleratore di particelle al mondo: l'LHC (Large Hadron Collider).

Grazie al CERN sono state fatte molte scoperte in ambito della fisica delle particelle, come la recente conferma sperimentale dell'effettiva esistenza del tanto cercato *bosone di Higgs*: il 4 Luglio 2012 i team degli esperimenti ATLAS (A Toroidal LHC ApparatuS) e CMS (Compact Muon Solenoid) confermarono la scoperta del bosone di Higgs, scoperta che portò poi, il 10 Dicembre 2013, al conferimento del premio Nobel per la fisica a Peter Higgs e François Englert, principali ricercatori che teorizzarono l'esistenza di questa particella subatomica.

Ma il CERN è stato protagonista anche di molte innovazioni in ambito informatico: pensiamo al servizio internet Web, o WWW (World Wide Web), che fu ideato proprio al CERN nel 1989 da Tim Berners-Lee, adesso fondatore e presidente del W3C (World Wide Web Consortium).

È in questo contesto che si inquadra il progetto *Improving the Worldwide Impact of Indico* che il sottoscritto, Tommaso Papini, ha seguito nel corso dei 14 mesi passati al CERN con il programma di *Technical Student* (dal 1 Ottobre 2013 al 30 Novembre 2014), sotto la supervisione di Pedro Ferreira (dipendente del CERN ed attuale Project Manager di Indico) al CERN e del Prof. Pierluigi Crescenzi (professore presso l'Università degli Studi di Firenze) da Firenze.

Quest'elaborato, volto ad essere un resoconto di questi 14 mesi, sarà suddiviso in tre parti ed esporrà, in modo comprensivo ed esaustivo, il progetto seguito in Indico. All'interno della prima parte verrà esposta una panoramica dell'ambiente CERN e del progetto Indico. Nella seconda parte, invece, saranno esposte le varie parti e le varie fasi che hanno composto il progetto stesso nell'arco di questi 14 mesi, assieme ad una panoramica sugli strumenti ed i linguaggi utilizzati. Infine, nella parte conclusiva, si parlerà di altri progetti minori seguiti al CERN e si esporranno delle note conclusive del progetto seguito.

INDICE

ELENCO DELLE FIGURE

LISTATO DEL CODICE

ACRONIMI

ACM Association for Computing Machinery

AD	Antiproton Decelerator
ALICE	A Large Ion Collider Experiment
API	Application Programming Interface
ATLAS	A Toroidal LHC ApparatuS
AVC	AudioVisual and Collaborative Services
BE	Beams
CERN	Conseil Européen pour la Recherche Nucléaire
CIS	Collaboration & Information Services
CMS	Compact Muon Solenoid
COMPASS	Common Muon and Proton Apparatus for Structure and Spectroscopy
CP	Charge Parity
DB	Database
DG	Director General
EGEE	Enabling Grids for E-sciencE
EN	Engineering
FNAL	Fermi National Accelerator Laboratory
FP	Finance, Procurement and Knowledge Transfer
GS	General Infrastructure Services
HEP	High Energy Physics
HR	Human Resources
IC	Indefinite Contract
IHEP	Institute of High Energy Physics
INFN	Istituto Nazionale di Fisica Nucleare
ISOLDE	On-Line Isotope Mass Separator
IT	Information Technology
KT	Knowledge Transfer

KVM	Kernel-based Virtual Machine
LEAR	Low Energy Antiproton Ring
LEIR	Low Energy Ion Ring
LEP	Large Electron–Positron Collider
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LHCf	Large Hadron Collider forward
Linac	Linear Accelerator
MIME	Multipurpose Internet Mail Extensions
MoEDAL	Monopole and Exotics Detector at the Large Hadron Collider
OERN	Organisation Européenne pour la Recherche Nucléaire
ORDBMS	Object-Relational Database Management System
PH	Physics
PS	Proton Synchrotron
PSB	Proton Synchrotron Booster
QEMU	Quick EMUlator
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
SL6	Scientific Linux 6
SLC6	Scientific Linux CERN 6
SMTP	Simple Mail Transfer Protocol
SPS	Super Proton Synchrotron
SSH	Secure Shell
SSL	Secure Sockets Layer
TE	Technology
TOTEM	TOTal Elastic and diffractive cross section Measurement
TTY	Teletypewriter

UI	User Interface
VM	Virtual Machine
W3C	World Wide Web Consortium
WSGI	Web Server Gateway Interface
WYSIWYG	What You See Is What You Get
WWW	World Wide Web
ZODB	Zope Object Database

Parte I

CERN E INDICO

1 CERN

Il CERN, anche noto come Organizzazione Europea per la Ricerca sul Nucleare, è l'organizzazione che opera il più grande laboratorio di fisica delle particelle al mondo.

Venne istituito il 29 Settembre 1954 da una serie di stati fondatori nei pressi di Ginevra, a cavallo del confine franco-svizzero. Al momento, il CERN vanta un totale di 21 stati membri, dei quali soltanto Israele non europeo.

Figura 1: Logo del CERN.

Col termine CERN si indica spesso il laboratorio stesso che, al 2013, contava 2'513 membri staff e circa 12'313 tra fellow, collaboratori associati e studenti, rappresentando un totale di 608 università e 113 nazionalità.

L'obiettivo principale del CERN è quello di fornire acceleratori di particelle ed altri strumenti e infrastrutture necessarie per le ricerche di fisica delle alte energie.

Al CERN è stata anche proposta e sviluppata una prima implementazione del World Wide Web. All'interno del sito di Meyrin del CERN è presente il Data Center principale, che vanta una serie di infrastrutture per l'elaborazione di dati, con lo scopo principale di analizzare dati sperimentali provenienti dal complesso degli acceleratori. Per questo motivo, e per il fatto che questi dati dovevano essere disponibili ed analizzabili da molti altri centri di ricerca, spesso molto distanti, il Data Center di Meyrin divenne un grandissimo centro di smistamento dati, formando una vasta rete tra i vari centri di ricerca.

1.1 BREVE STORIA

La convenzione che istituì il CERN venne ratificata il 29 Settembre 1954 da 12 stati dell'Europa occidentale.

L'acronimo CERN, in origine, indicava la sigla francese *Conseil Européen pour la Recherche Nucléaire* (Consiglio Europeo per la Ricerca sul Nucleare), che rappresentava il consiglio provvisorio istituito nel 1952 dai 12 stati fondatori per la costruzione di un laboratorio di ricerca di fisica nucleare europeo. Quando il consiglio venne sciolto nel 1954 il nome cambiò all'attuale Organizzazione Europea per la Ricerca sul Nucleare (in francese, *Organisation Européenne pour la Recherche Nucléaire*). In quest'occasione si decise di mantenere la vecchia sigla CERN al posto della, più brutta e meno d'impatto, sigla OERN. Il fisico Heisenberg stesso si pronunciò sul fatto che l'acronimo sarebbe dovuto rimanere CERN anche se il nome non lo era più.

Il CERN viene considerato un laboratorio di pace in quanto grazie ad esso persone provenienti da tutto il mondo hanno la possibilità di incontrarsi, discutere e collaborare su diversi progetti. Grazie al CERN riescono a lavorare assieme anche persone provenienti da stati in guerra tra loro, come ad esempio Israele e Palestina.

Inoltre, il CERN venne istituito circa 10 anni dopo la costruzione della bomba atomica come conseguenza principale della II guerra mondiale.

1.1.1 *Principali scoperte scientifiche*

In origine, lo scopo principale del CERN era quello di studiare i nuclei atomici, ma ben presto l'attenzione si volse verso lo studio della fisica delle alte energie, in particolare all'interazione tra le particelle subatomiche.

1973 Nel 1973 venne scoperta, grazie alla camera a bolle Gargamelle, una particolare interazione tra particelle subatomiche, detta *corrente debole neutra* (Figura 2) che, nel 1979, permise l'attribuzione del Premio Nobel per la Fisica ad Abdus Salam, Sheldon Glashow e Steven Weinberg, che teorizzarono quest'interazione in passato.

1983 Nel 1983 venne fatta un'altra importantissima scoperta di fisica delle particelle, ovvero la scoperta dei bosoni W e Z. La teorizzazione di questi bosoni fu una conseguenza quasi immediata dell'osservazione della corrente debole neutra, ma prima di poter osservare direttamente queste particelle subatomiche si dovette aspettare di avere un acceleratore di particelle abbastanza potente da produrle. Il primo acceleratore in grado di fare ciò installato al CERN fu il Super Proton Synchrotron (SPS), che permise, nel Gennaio del 1983 di osservare chiare tracce del bosone W. Gli esperimenti principali furono due, condotti in parallelo, e vennero chiamati UA1 (condotto da Carlo Rubbia) e UA2 (condotto da Pierre Darriulat). I due esperimenti osservarono anche, nel Maggio 1983, i bosoni Z,

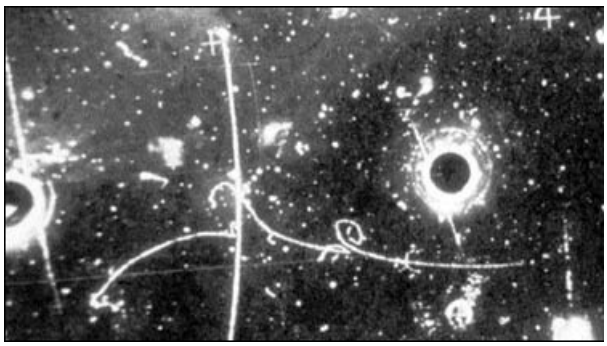


Figura 2: L'evento di corrente debole neutra osservato con Gargamelle.

grazie all'introduzione della tecnica del raffreddamento stocastico introdotta da Simon van der Meer. Nel 1984, la fondazione Nobel, attribuì a Rubbia e Van der Meer il Premio Nobel per la Fisica per l'insostituibile sforzo messo in queste ricerche.

1989

Nel 1989 venne confermato sperimentalmente, utilizzando il LEP (Large Electron–Positron Collider), che il numero di famiglie di particelle con neutrini leggeri è 3, numero consistente con il Modello Standard.

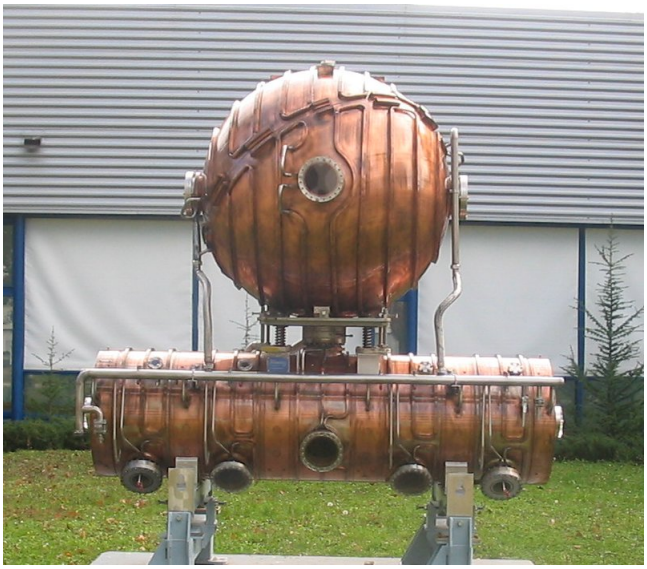


Figura 3: Una cavità RF del LEP esposta al Microcosm del CERN.

1992

Nel 1992 venne conferito il Premio Nobel per la Fisica a Georges Charpak per "per l'invenzione e lo sviluppo dei rivelatori di particelle, in particolare della camera proporzionale a multifili".

1995

Nel 1995 vennero, per la prima volta nella storia, creati atomi di anti-idrogeno

grazie all'esperimento PS210 realizzato al LEAR (Low Energy Antiproton Ring). Questi atomi di anti-idrogeno erano molto instabili e si distrussero subito dopo la creazione, ma non senza produrre un segnale elettrico unico, indice che erano effettivamente stati creati atomi di anti-idrogeno.

1999 L'esperimento NA48 dimostrò, nel 1999, la violazione della simmetria CP (Charge Parity). L'esperimento fu lanciato nel 1990 come successore dell'esperimento NA31 con lo specifico scopo di dimostrare la violazione della simmetria CP, la quale afferma che le leggi della fisica dovrebbero rimanere le stesse quando una particella viene sostituita con la sua anti-particella (simmetria di coniugazione di carica, C) e le coordinate spaziali invertite (simmetria di parità, P).

2010 Nel 2010 un gruppo di ricercatori guidati dal fisico Jeffrey Hangst riuscì ad isolare 38 atomi di anti-idrogeno per circa un decimo di secondo, mentre nel 2011 si riuscì a mantenere degli atomi di anti-idrogeno per più di 15 minuti, in modo da poter studiare più a fondo l'antimateria e le sue proprietà.

2012 Infine, nel 2012, venne finalmente osservato un bosone di massa circa 125 GeV/c^2 consistente con il tanto cercato bosone di Higgs. Questo bosone venne teorizzato più di 40 anni fa dai ricercatori Peter Higgs e François Englert ed ha una grande importanza per la fisica delle particelle in quanto la sua esistenza può spiegare come mai alcune particelle hanno massa. L'osservazione definitiva del bosone di Higgs avvenne il 4 Luglio 2012, sfruttando le collisioni prodotte dall'LHC, e venne osservato in contemporanea dai due esperimenti CMS e ATLAS. Questa importantissima scoperta portò al conferimento del Premio Nobel per la Fisica, il 10 Dicembre 2013, ai ricercatori Higgs e Englert.

1.1.2 *Il CERN e l'informatica*

Il primo computer arrivò al CERN nel 1959 e da allora i fisici iniziarono ad utilizzare sempre più gli strumenti informatici per le loro analisi. Un elemento importante in questo contesto fu l'italiano Paolo Zanella, capo del dipartimento IT (Information Technology) per 13 anni, dal 1976 al 1989. Gli esperimenti condotti al CERN producevano una mole di dati sempre più grande, tale da rendere impossibile la sola analisi "a mano".

Nel tempo, venne anche sperimentato il collegamento fra più calcolatori portando all'utilizzo delle prime reti di calcolatori. Presto, al CERN si formò uno dei più grandi centri di calcolo in Europa. Più recentemente si sono iniziate a sviluppare al CERN tecnologie per il grid computing, con progetti come Enabling Grids for E-sciencE (EGEE) o LHC Computing Grid.

Il servizio Internet World Wide Web prese vita al CERN dal progetto ENQUIRE, condotto da Tim Berners-Lee nel 1989 e Robert Cailliau nel 1990. Nel 1995 Berners-Lee e Cailliau vennero premiati dalla ACM (Association for Computing Machinery) per l'indispensabile contributo allo sviluppo del World Wide Web. Il

Figura 4: L'interno della Server Room del Data Center nel sito di Meyrin.

progetto era basato sul concetto di ipertesto, ovvero testo riprodotto su display e collegato ad altri testi tramite iperlink. L'obiettivo originale era quello di facilitare lo scambio di informazioni tra i ricercatori per condurre analisi ed esperimenti.

Il primo sito web fu attivo nel 1991 ed il 30 Aprile 1993 il CERN annunciò pubblicamente che il Web sarebbe stato libero per tutti.

1.2 IL COMPLESSO DEGLI ACCELERATORI

Al CERN viene utilizzata una serie di acceleratori, disposti in sequenza in modo da aumentare l'energia delle particelle prima di essere ridirette agli eventuali esperimenti o all'acceleratore successivo. Il logo stesso del CERN (Figura 1) è ispirato al complesso degli acceleratori presente.

I macchinari attualmente attivi al CERN sono i seguenti:

LINAC2 E LINAC3 Questi due acceleratori lineari (Linac sta per Linear Accelerator) generano particelle a bassa energia. Linac2 accelera protoni fino a 50 MeV per poi iniettarli nel Proton Synchrotron Booster (PSB), mentre Linac3 genera ioni pesanti a 4.2 $^{MeV}/_{u}$ per iniettarli nel Low Energy Ion Ring (LEIR).

PROTON SYNCHROTRON BOOSTER Aumenta l'energia dei protoni prodotti dal Linac2 prima di trasferirli ad altri acceleratori.

LOW ENERGY ION RING Accelera gli ioni pesanti prodotti dal Linac3 prima di trasferirli al Proton Synchrotron (PS). Questo acceleratore fu commissionato

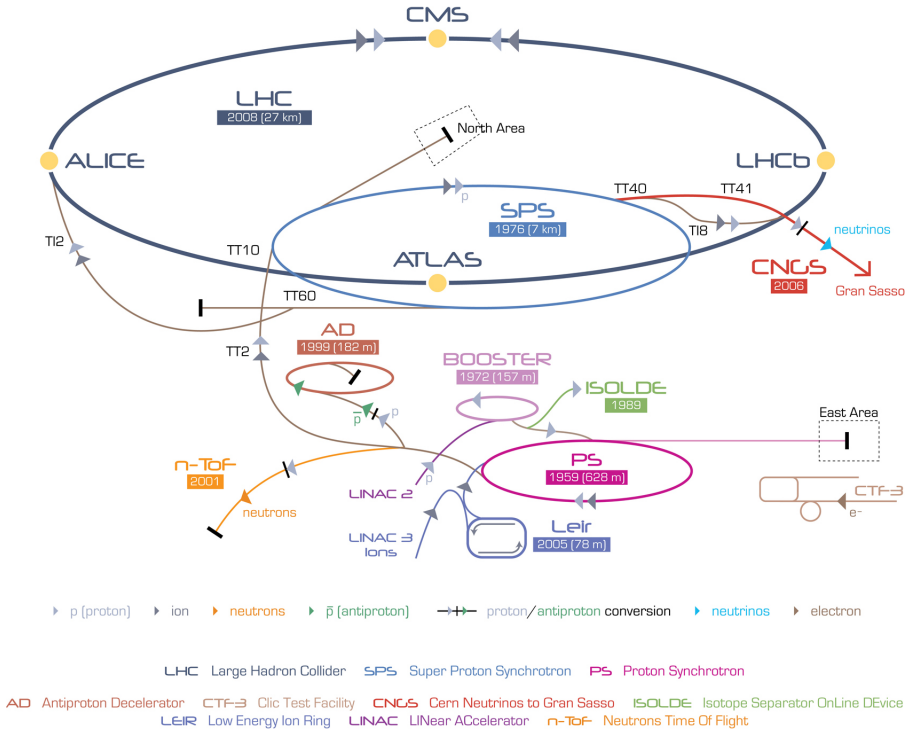


Figura 5: Uno schema del complesso degli acceleratori presenti al CERN con legenda.

nel 2005 come riconfigurazione del precedente acceleratore, il LEAR.

PROTON SYNCHROTRON Accelera le particelle fino a 28 GeV. Fu costruito tra il 1954 ed il 1959 ed è ancora attivo come alimentatore del più potente SPS.

SUPER PROTON SYNCHROTRON Acceleratore circolare installato in un tunnel di 2 Km di diametro ed attivo dal 1976. Fu costruito inizialmente per portare le particelle fino a 300 GeV, per poi essere gradualmente aggiornato fino ai 450 GeV. Questo acceleratore ha i propri esperimenti, al momento COMPASS (Common Muon and Proton Apparatus for Structure and Spectroscopy) ed NA62, ma veniva anche utilizzato come collisore protoni-antiprotoni e per accelerare elettroni e positroni prima di iniettarli nel LEP. Dal 2008 viene invece utilizzato per iniettare protoni e ioni pesanti nell'LHC.

ON-LINE ISOTOPE MASS SEPARATOR Anche indicato con la sigla ISOLDE, questa struttura viene utilizzata per studiare i nuclei instabili. Ioni radioattivi provenienti dal PSB vengono fatti scontrare con un energia tra 1.0 e 1.4 GeV.

ANTIPROTON DECELERATOR Anche noto come AD, serve a ridurre la velocità degli antiprotoni di circa un 10% della velocità della luce per poter studiare

l'antimateria.

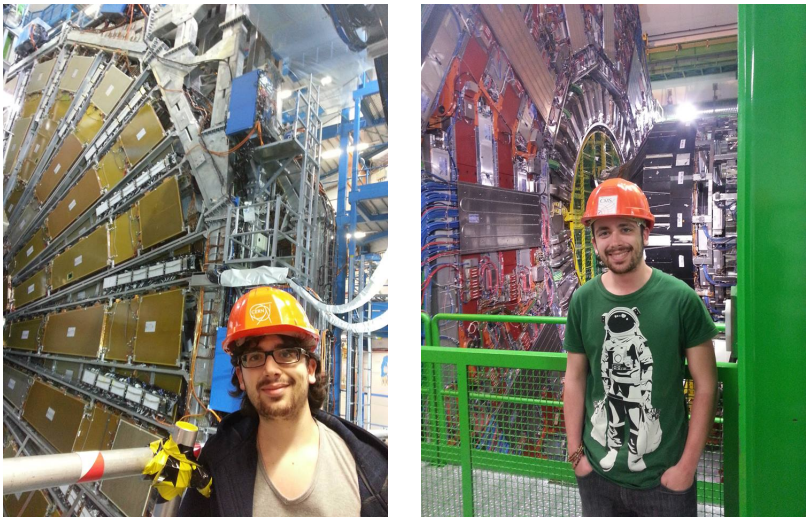
COMPACT LINEAR COLLIDER Studia la fattibilità di possibili acceleratori lineari futuri.

1.2.1 *Large Hadron Collider*

L'LHC è attualmente il più grande acceleratore di particelle installato al CERN, ovvero l'acceleratore principale su cui vengono eseguiti la maggior parte degli esperimenti.

Il tunnel dell'LHC è situato 100 metri sotto terra, nella zona tra l'Aeroporto Internazionale di Ginevra e i monti del Giura. Il tunnel ha una circonferenza di circa 27 Km ed era precedentemente occupato dal vecchio acceleratore LEP, che venne disattivato definitivamente nel 2000. Gli acceleratori PS ed SPS vengono utilizzati per pre-accelerare i protoni iniettati nell'LHC.

Lungo l'acceleratore sono presenti i suoi sette esperimenti attualmente attivi: CMS, ATLAS, LHCb (Large Hadron Collider beauty), MoEDAL (Monopole and Exotics Detector at the Large Hadron Collider), TOTEM (TOTal Elastic and diffractive cross section Measurement), LHCf (Large Hadron Collider forward) ed ALICE (A Large Ion Collider Experiment).



(a) ATLAS (b) CMS

Figura 6: Il sottoscritto in visita ad alcuni esperimenti dell'LHC.

L'LHC iniziò fin da subito a generare una mole enorme di dati da inviare a diversi laboratori sparsi per il mondo in modo da poter eseguire calcoli distribuiti utilizzando una struttura grid dedicata: la LHC Computing Grid.

Il primo fascio di particelle fu iniettato nell'LHC nell'Agosto del 2008 ma, a causa di problemi tecnici, non venne riattivato fino al Novembre del 2009. Il 30 Marzo 2010, invece, si verificò la prima collisione tra due fasci di protoni, ognuno alla velocità di 3.5 TeV, generando una collisione di 7 TeV. Dal Marzo 2012 si passò invece a collisioni di 8 TeV (ovvero 4 TeV in ogni direzione). Agli inizi del 2013 l'LHC venne disattivato per avviare un periodo di manutenzione e aggiornamento della durata di 2 anni. Dai primi mesi del 2015, l'LHC è stato finalmente rimesso in funzione per sperimentare le prime collisioni a 13 TeV. Nel Luglio 2012 venne annunciata l'osservazione sperimentale di una particella subatomica consistente con il tanto agognato bosone di Higgs. Nel Marzo 2013 venne confermato dal CERN che, a seguito di una serie di analisi condotte sui dati sperimentali, la particella trovata era effettivamente il bosone di Higgs.

1.3 LAVORO AL CERN

Il CERN è il risultato di una collaborazione tra stati, università, scienziati e studenti che sono motivati non da un margine di profitto, ma bensì da un impegno a creare e condividere la conoscenza.

Le persone del CERN, sia che siano impiegati fissi, che soltanto di passaggio, prendono parte a ricerche e collaborazioni interessanti con persone di tutto il mondo, cercando di risolvere i misteri della fisica o di creare qualcosa di nuovo.

1.3.1 *Cultura e valori*

Al CERN vige il Code of Conduct (Codice della Condotta), ovvero una guida su come trattare gli altri colleghi in accordo con i valori del CERN. [2]

I cinque valori di base su cui si basa il CERN sono:

INTEGRITÀ Ovvero comportarsi in modo etico, onesto e ritenersi responsabile delle proprie azioni.

IMPEGNO Dimostrare, durante il proprio lavoro, un alto livello di motivazione e dedizione verso l'organizzazione ed il proprio progetto.

PROFESSIONALITÀ Produrre risultati di alta qualità rispettando i limiti di tempo e le risorse previsti per il completamento del lavoro.

CREATIVITÀ Promuovere l'innovazione nel proprio settore di lavoro e permettere lo sviluppo e l'evoluzione dell'organizzazione.

DIVERSITÀ Ovvero saper apprezzare le differenze e promuovere l'uguaglianza e la collaborazione.

1.3.2 *Carriera al CERN*

Il CERN ammette un ampio ventaglio di possibilità per fare carriera ed acquisire esperienza lavorativa al suo interno. Ci sono svariati tipi di contratti per tutti i tipi di aspiranti dipendenti o collaboratori CERN: contatti di pochi mesi o contratti di anni, contratti per studenti, per laureati o per professionisti, ecc. . . [1]

Uno dei principi di assunzione e lavoro al CERN è la non rinnovabilità del contratto: ogni tipo di contratto può essere assegnato ad ogni individuo al più una sola volta. Questo serve a garantire la diversità ed il continuo ricambio di personale ed allo stesso tempo formare abitanti degli stati membri con un'esperienza di alto livello.

Studente

In qualità di studente universitario si hanno molte possibilità di lavoro al CERN con contratti di Summer, Admin, Technical e Doctoral Student. Il programma di Technical Student, vissuto dal sottoscritto, prevede una durata dai 6 ai 12 mesi, con la possibilità di estendere il contratto di 2 ulteriori mesi (risorse permettendo). Questi tipi di contratto permettono agli studenti di fare esperienza sul campo ed affrontare sfide intellettuali con alte possibilità di formazione.

Laureato

I contratti per laureati durano normalmente dai 2 ai 3 anni e permettono una formazione ad alto livello. I principali contratti di questo tipo sono la Fellowship, il Graduate Engineer Training ed il contratto Marie-Curie.

Membro dello Staff

I contratti di Staff durano 5 anni e sono uno dei più ambiti e prestigiosi tipi di contratto al CERN come dipendente effettivo e professionista.

E dopo...

Purtroppo anche i contratti di Staff sono non rinnovabili ed a tempo determinato (5 anni). Tuttavia, verso gli ultimi anni di contratto, il CERN da la possibilità al membro dello Staff di fare domanda per il cosiddetto Indefinite Contract (IC), ovvero un contratto di Staff a tempo indeterminato.

1.3.3 *La struttura generale*

Il consiglio del CERN è la più alta autorità dell'intera organizzazione ed è responsabile di tutte le decisioni più importanti. Si occupa di controllare tutte le attività scientifiche, tecniche ed amministrative del CERN e di approvare programmi, attività, budget e spese. Il consiglio è composto da 21 stati membri, ognuno dei quali fornisce due delegati come rappresentanti nel consiglio. Uno dei due delegati rappresenterà gli interessi amministrativi del proprio governo, mentre l'altro rappresenterà gli interessi scientifici nazionali. Nel consiglio, ogni stato membro ha a disposizione un solo voto e per la maggior parte delle decisioni si vota per maggioranza. Il consiglio è assistito, nelle sue decisioni, dalla Scientific Policy Commmittee e dalla Finance Committee.

La Scientific Policy Committee (Commissione della Politica Scientifica) valuta i pro e i contro delle attività proposte dai fisici e suggerisce strategie per il programma scientifico al CERN. I membri vengono eletti dalla commissione

stessa e suggeriti dal consiglio in base ai meriti scientifici, senza distinzioni di nazionalità. La Finance Committee (Commissione della Finanza) è invece composta da rappresentanti delle amministrazioni nazionali e si occupa di gestire i contributi degli stati membri e budget e spese dell'organizzazione.

Il Director General (DG), eletto dal consiglio ogni 5 anni, si occupa di gestire il laboratorio CERN, rispondendo direttamente al consiglio. Il DG è assistito da un direttorato e gestisce il laboratorio attraverso una struttura di dipartimenti. Al momento del periodo di Techical Student del sottoscritto il DG era il fisico tedesco Rolf-Dieter Heuer, in carica dal 2009. Dal Gennaio 2016, invece sarà Fabiola Giannotti, fisica italiana e prima DG donna della storia.

Sotto al DG si sviluppa la struttura del laboratorio CERN, ovvero una struttura piramidale organizzata in dipartimenti. Più in particolare, il CERN è suddiviso in otto dipartimenti: BE (Beams), EN (Engineering), FP (Finance, Procurement and Knowledge Transfer), GS (General Infrastructure Services), HR (Human Resources), IT, PH (Physics), TE (Technology). Ogni dipartimento (gestito da un Department Head) è suddiviso in gruppi, che si occupano di settori specifici all'interno dello stesso dipartimento. A loro volta, ogni gruppo (gestito da un Group Leader) si suddivide in sezioni, ognuna della quale si specializza ulteriormente a seconda del lavoro svolto. Infine, ogni sezione (gestita da un Section Leader) è composta da una serie di progetti. Ad ogni progetto è associato un team il quale, sotto la supervisione di un Project Manager, si occupa di portare a termine il proprio progetto nel modo migliore possibile. [3]

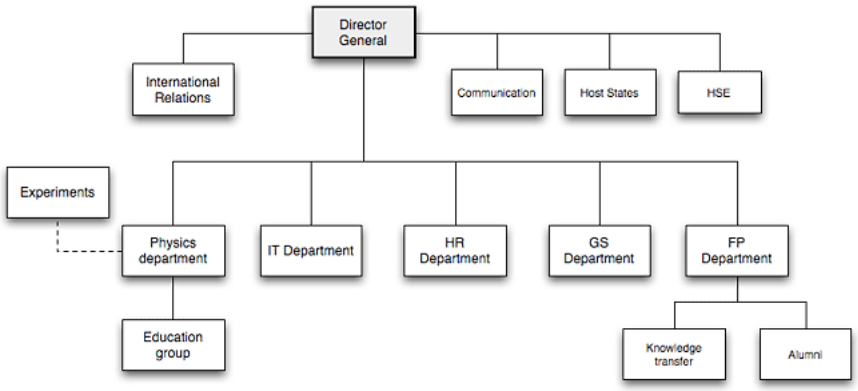


Figura 7: Organigramma ad alta granularità dell'organizzazione CERN.

Indico

Il progetto Indico, oggetto di questa tesi, è così collocato all'interno della struttura del CERN: IT - CIS - AVC - Indico. Indico fa quindi, innanzitutto, parte del dipartimento IT, ovvero il dipartimento dei servizi informatici al CERN (il cui Department Head è Frédéric Hemmer). Quindi, Indico fa parte del gruppo CIS (Collaboration & Information Services) che si occupa dei servizi di collaborazione come videoconferenze, pubblicazioni elettroniche, e servizi

di gestione dei documenti. Il Group Leader del gruppo CIS è l'inglese Tim Smith. Più nel dettaglio, Indico fa parte della sezione AVC (AudioVisual and Collaborative Services), la quale si occupa di servizi per le videoconferenze e la gestione di eventi, sotto la supervisione del Section Leader Thomas Baron. Come vedremo più in dettaglio nel Capitolo 2, Indico è proprio il progetto che si occupa della gestione dell'applicazione web (o web application) per la creazione e gestione di eventi (come conferenze, meeting, ecc...). Durante la permanenza del sottoscritto al CERN, il Project Manager di Indico è stato prima Jose Benito Gonzales e quindi Pedro Ferreira.

1.3.4 *Knowledge Transfer*

Il gruppo Knowledge Transfer (KT), parte del dipartimento FP, si occupa si organizzare, catturare e distribuire la conoscenza e la tecnologia all'interno e fuori dal CERN. Uno degli obiettivi è quello, ad esempio, di applicare i risultati scientifici ottenuti al CERN in altri campi, come ad esempio la medicina (si stanno studiando, ad esempio, possibili tecnologie per la rimozione di tumori utilizzando piccoli acceleratori di particelle). Spesso, invece, quello che il gruppo KT si prefigge di fare è rendere qualche prodotto o tecnologia CERN più accessibile al mondo esterno, come università o aziende.

Proprio questo era il fine dei fondi che il gruppo KT ha destinato al progetto Indico. L'obiettivo era quello di rendere Indico più accessibile alle istituzioni e alle aziende che lo volessero utilizzare al di fuori del CERN e di renderlo più facile da utilizzare. Ma di questo parleremo ampiamente nella Parte II di questa tesi.

2 INDICO

Indico è una web application che fornisce strumenti per l'organizzazione di eventi di qualsiasi dimensione, da semplici lezioni a grandi conferenze. Prese vita nel 2002 come progetto europeo, per poi venir adottato dal CERN nel 2004 come software per la gestione di eventi. Dal 2004 ad oggi, il CERN è stato il principale finanziatore del progetto Indico, dedicando un team al suo sviluppo e supporto.

Indico è uno strumento open source[2], il che vuol dire non soltanto che è completamente gratuito, ma anche che altre istituzioni o individui al di fuori del CERN possono ispezionarne il codice, contribuire allo sviluppo o modificarlo secondo le proprie preferenze.

Indico viene quindi utilizzato da centinaia di istituzioni e organizzazioni in tutto il mondo, ma il principale utilizzatore rimane il CERN stesso, che lo utilizza ogni giorno per gestire più di 300.000 eventi di diversa complessità e circa 200 stanze per conferenze e riunioni. L'istanza di Indico installata al CERN prende il nome di Indico@CERN, raggiungibile all'indirizzo `http://indico.cern.ch/`. Inoltre, al CERN è installata anche una seconda istanza minore di Indico utilizzata per prenotare gli uffici Burotel.

2.1 PRINCIPALI CARATTERISTICHE

Con Indico si possono gestire eventi di qualsiasi complessità: lezioni, riunioni, seminari e conferenze. Indico fornisce all'utente, durante tutta la fase di creazione e modifica di un evento, tutta una serie di strumenti per agevolare queste operazioni: l'utente può decidere secondo quali criteri far registrare terzi al suo evento, oppure come gestire l'invio di pubblicazioni o di materiale relativo all'evento.

UI

Indico utilizza anche una potente e immediata interfaccia utente che si basa sul design WYSIWYG (What You See Is What You Get). Grazie a questa UI (User Interface) sarà facilissimo eseguire azioni altrimenti più complesse e tediose: la definizione di un form di registrazione per l'evento, la gestione delle timetables (Figura 8), che implementano un'intuitiva interfaccia drag-and-drop, oppure editor di testo che permettono l'utilizzo di rich text o formule matematiche.

Struttura

Indico, ed il suo sistema di protezione, è organizzato in una struttura ad albero per categorie. Indico è stato sviluppato principalmente per grandi aziende, per questo gli eventi, ed il materiale ad essi associato, sono organizzati secondo una

2 Il codice aggiornato di Indico, e di tutti i progetti ad esso correlati, può essere trovato all'indirizzo `https://github.com/indico`.

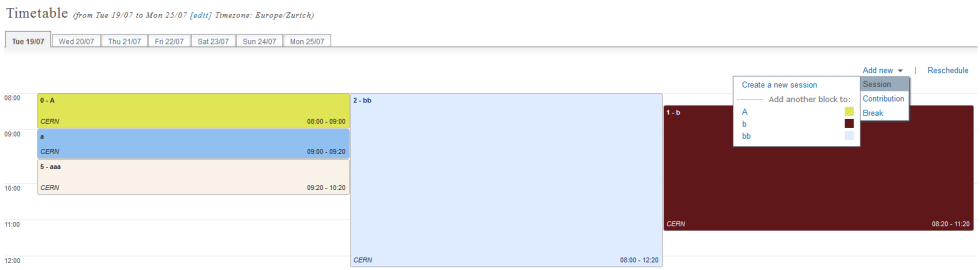


Figura 8: Un esempio di timetable di una conferenza in Indico.

struttura gerarchica di categorie. L'amministratore del sistema potrà decidere ed assegnare livelli di protezione alle varie categorie secondo diversi livelli di granularità.

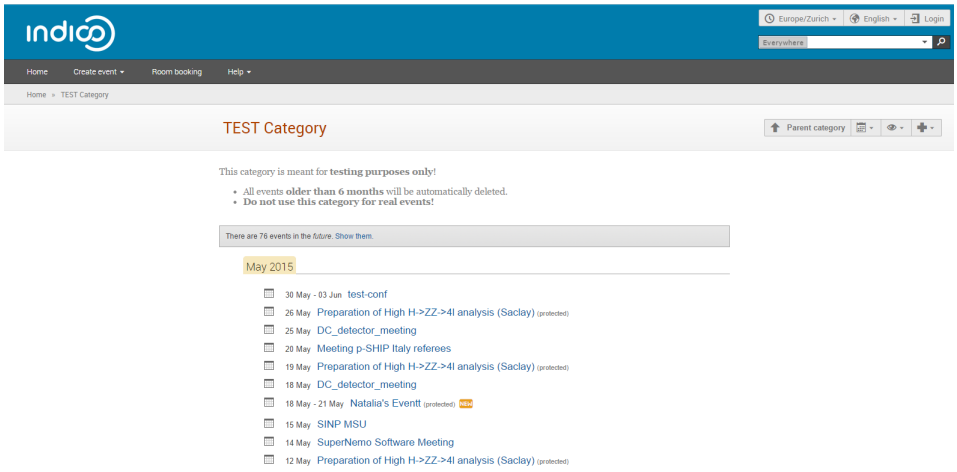


Figura 9: Un esempio di categoria con una serie di eventi al suo interno.

Ricerca

In Indico è incluso anche un potente sistema di ricerca, che permette di ricercare gli eventi desiderati tramite poche parole chiave o osservare gli eventi previsti per un determinato periodo di tempo. Inoltre, tramite la dashboard, è possibile accedere velocemente a tutti gli eventi a cui l'utente è iscritto o che sta osservando.

Room Booking

Le compagnie e le organizzazioni, specialmente le più grandi, hanno spesso bisogno di gestire, limitare e tener traccia dell'utilizzo delle varie stanze e siti al loro interno. Per questo Indico include un potente ed intuitivo modulo per la prenotazione delle stanza (in inglese, room booking) che permette all'utente di specificare le caratteristiche di una stanza, approvare o meno la prenotazione di certe stanze e gestire strumenti e materiale messi a disposizione di certe stanza, come dispositivi audiovisivi per le video conferenze.

Chat & Video

Per rendere la gestione online di riunioni e conferenze ancora più efficiente, Indico integra perfettamente Vidyo[1], uno strumento per le videoconferenze, e permette di associare delle chatrooms Jabber/XMPP agli eventi.

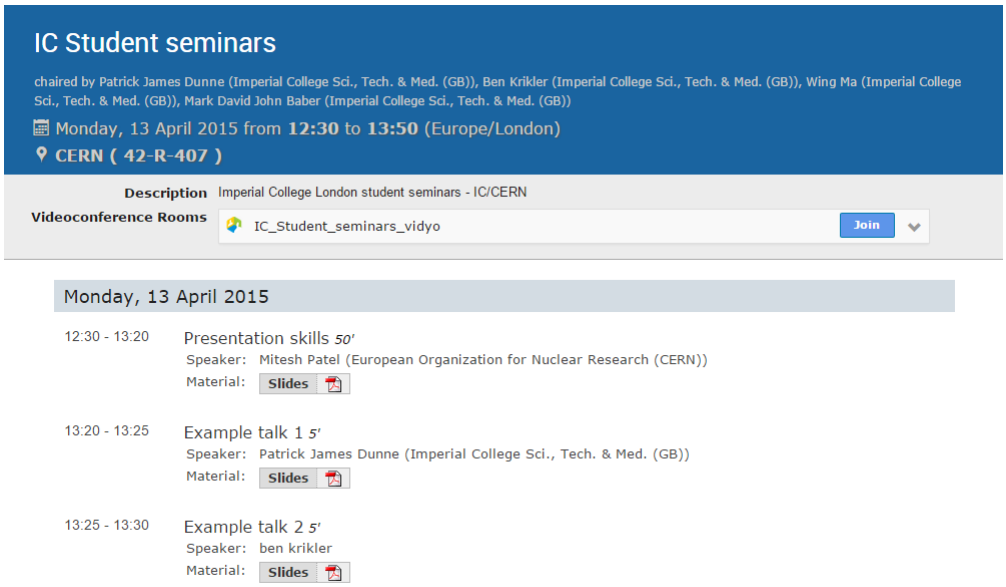


Figura 10: Un esempio di riunione in Indico con una videoconference room Vidyo associata.

API

Dal momento che Indico è stato creato per agevolare la collaborazione e lo scambio di dati, tutte le informazioni contenute in Indico non sono esclusive di Indico stesso, ma possono essere recuperate (una volta accertato di avere l'accesso a quelle informazioni) tramite una semplice API (Application Programming Interface). Quest'API è garantita essere RESTful (dove REST significa Representational State Transfer) ed è in costante aggiornamento.

2.2 DETTAGLI TECNICI

Indico è un'applicazione web scritta principalmente in Python e Javascript (si veda la Figura 11).

Figura 11: Una statistica dei linguaggi utilizzati nel codice di Indico (come mostrata su Github).

1 `http://it.vidyo.com/`.

Essendo un'applicazione web, Indico implementa un web server ed un Database (DB) per gestire le richieste dell'utente e recuperare i dati richiesti.

WSGI

Indico utilizza WSGI (Web Server Gateway Interface) per gestire il web server. WSGI è un'interfaccia standard tra web server e applicazioni Python, ovvero un'astrazione per rendere la comunicazione tra le due parti più semplice e modulare. In questo modo il server e l'applicazione sono completamente separati e seguono lo standard. Ad esempio per Indico@CERN si utilizza Apache come web server, ma un amministratore di un'altra istanza può scegliere un altro server se lo desidera, in quanto WSGI si configura senza problemi con quasi tutti gli web server in circolazione. [13]

DB

Prima dell'inizio del periodo di Techical Student, Indico utilizzava ZODB (Zope Object Database) come database. ZODB è un database object-oriented che immagazzina oggetti Python. Dai primi del 2014 è invece stata iniziata una fase di migrazione, in quanto ZODB presentava alcuni svantaggi, come la difficoltà di eseguire query, la non indicizzazione dei record e il fatto di poter essere utilizzabile soltanto con Python. Dopo alcuni mesi di ricerca su quale potesse essere il miglior candidato per sostituire ZODB, si è scelto PostgreSQL (spesso detto Postgres) come vincitore. Postgres è un ORDBMS (Object-Relational Database Management System), ovvero un database relazionale a oggetti, che garantirà prestazioni più elevate durante l'utilizzo di Indico. Ovviamente passare da un database ad un altro non è una cosa semplice, in quanto tutto il codice dove si comunica con il DB dev'essere adeguato al nuovo database. Migrare l'intero DB di Indico@CERN in sol colpo era quindi un'impresa del tutto on fattibile. Si è optato quindi per una migrazione modulare: il processo di migrazione durerà circa un anno, durante il quale Indico utilizzerà un sistema di gestione del DB ibrido ZODB+Postgres, migrando modulo per modulo. [11][5]

Flask

Flask è un micro framework per applicazioni web, scritto in Python e basato su Werkzeug e Jinja2. Uno dei principali vantaggi offerti da Flask è l'URL Routing, ovvero la possibilità di gestire gli URL in modo dinamico. Si possono associare diversi template per ogni tipo di richiesta ricevuta dal server, in modo da generare una pagina dinamica a seconda degli argomenti passati da Flask al template engine. [14]

Template engine(s)

Un template engine è uno strumento che permette di creare documenti personalizzati e dinamici a partire da un template (ovvero un modello) di base e dei dati di input. In questo caso, quando si parla di template engines, ci si riferisce a strumenti che generano pagine web secondo un certo modello inserendovi dei dati dinamici. Indico ha utilizzato per molti anni Mako come template engine ma in questi ultimi anni sta lentamente passando a Jinja2. [12]

A parte queste tecnologie citate, Indico ha iniziato a impiegare, nel corso degli ultimi anni, altri strumenti degni di nota, come ad esempio WTForms, per la gestione dei campi dei form (ad esempio form di registrazione), SQLAchemy, ovvero un toolkit open source SQL, ed Alembic, uno strumento per la migrazione di database da utilizzare con SQLAlchemy.

2.3 LA COMMUNITY

Come già accennato, Indico è un software open source. Il suo codice sorgente più aggiornato è infatti disponibile su Github, all'indirizzo `https://github.com/indico/indico`. Il fatto di essere open source ha portato centinaia di organizzazioni e istituzioni, soprattutto nel campo della fisica delle particelle (ad esempio INFN e IHEP), ad utilizzare Indico per la gestione dei propri eventi. Negli anni questo ha portato alla formazione di una vera e propria community di utilizzatori (user e admin) di Indico.

Questa commuity non soltanto utilizza Indico per i propri eventi, ma contribuisce anche allo sviluppo stesso del software, inviando segnalazioni agli sviluppatori del team Indico al CERN sotto forma di ticket. Un ticket indica un problema da risolvere, o qualcosa da migliorare; risolvere il problema in questione o apportare la miglioria richiesta viene indicato con il termine di "chiusura del ticket".

Altre volte invece, se chi si imbatte in un errore è uno sviluppatore a sua volta, può capitare che provi a risolvere il problema da solo, lavorando sul codice di Indico. Una volta risolto, chi ha risolto il problema può inviare una cosiddetta *pull request* tramite Github (il repository ufficiale che ospita il codice di Indico) agli sviluppatori di Indico al CERN, in modo da indicare che quel problema è già stato risolto e possono includere (operazione di *merge*) le modifiche apportate al codice principale di Indico.

2.4 INDICOUSERGUIDE

Use IndicoUserGuide.pdf!!!

Parte II

INDICO KT PROJECT

3 KT PROJECT: UNA PANORAMICA

In questo capitolo introdurremo i principali obiettivi del Progetto KT per Indico in modo da dare una panoramica generale dei sottoprogetti di cui il progetto principale è composto. Inoltre verranno esposti i principali strumenti e linguaggi utilizzati durante lo sviluppo del progetto.

3.1 PROGETTI PRINCIPALI

Il progetto finanziato dal gruppo KT per Indico, oggetto principale del programma di Technical Student a cui ha partecipato il sottoscritto, è nato come co-progetto tra i gruppi IT e KT. L'obiettivo principale di questo progetto è quello di migliorare la visibilità e l'impatto di Indico in tutto il mondo, in particolare al di fuori della comunità HEP (High Energy Physics), all'interno della quale Indico si è già ampiamente affermato.

Come si è già accennato nel Capitolo 1, il concetto di KT si basa sull'idea della diffusione e condivisione della conoscenza e degli strumenti necessari per ottenerla. L'idea alla base del progetto era quindi quella di modificare e migliorare il software Indico in modo da permettere una miglior diffusione dello stesso nel mondo. Per fare questo, il KT Project si era prefissato tre obiettivi generali da raggiungere:

- rendere Indico più accessibile agli utenti
- rendere Indico più semplice da utilizzare e personalizzare
- rendere Indico, in generale, più moderno e visivamente "attraente"

Gli obiettivi prefissati col Progetto KT erano quindi molto ampi e generali e, come ci può immaginare, anche piuttosto complessi da mettere in pratica. Per questa ragione il progetto che è stato assegnato al sottoscritto non era che il primo di una serie di progetti, finanziati dal gruppo KT, al fine di migliorare l'impatto a livello mondiale del software Indico. Infatti, al durante il periodo di 14 mesi passati a Ginevra dal sottoscritto, erano stati approvati e finanziati già due progetti dal gruppo KT per Indico: il primo, assegnato al sottoscritto, ed un secondo da assegnare ad un futuro membro del team Indico, molto probabilmente un altro Techical Student.

Il primo Progetto KT per Indico è stato quindi pianificato e suddiviso in una serie di sotto-progetti, i quali dovevano essere terminati durante i 14 mesi del programma Technical Student. Quattro di questi sotto-progetti sono risultati

essere più importanti e complessi degli altri ed sono andati ad occupare gran parte del periodo di Technical Student. Di seguito ne parleremo brevemente per avere un'idea generale dei progetti principali del KT Project, mentre nei Capitoli successivi vedremo in dettaglio ognuno di essi.

3.1.1 *Cloud Deployment*

La prima fase del Progetto KT riguardava il Cloud Deployment di Indico, ovvero l'automatizzazione del processo di installazione e configurazione di Indico sia in ambiente cloud che in ambiente virtuale.

Il progetto è durato circe un mese e mezzo, dal 23 Ottobre 2013 al 9 Dicembre 2013, ed ha portato alla scrittura di due script fabric, uno per la creazione di immagini virtuali ed uno di gestione remota, ed una recipe cloud-init, per il deployment su struttura cloud.

3.1.2 *Distribuzione e Packaging*

3.1.3 *Instance Tracker*

3.1.4 *Conference Customization Prototype*

3.2 STRUMENTI E LINGUAGGI

Con questa ultima Sezione introduttiva, intendiamo fornire al lettore una serie di conoscenze e nozioni di base utili a capire il lavoro svolto con questo progetto. Indico infatti è composto da molti linguaggi diversi ed utilizza molti strumenti, sviluppati da terzi, senza i quali non potrebbe funzionare. È necessario quindi sapere quali sono e cosa fanno ognuno di questi strumenti, nonché essere a conoscenza dei linguaggi utilizzati.

3.2.1 *Python*

DA FARE: DESCRIZIONE PYTHON!!!

Un importante comando offerto da python è .**format**(): questa funzione sostituisce a degli speciali *placeholder* (o segnaposti, in italiano), presenti nell'oggetto stringa sul quale viene eseguito, i valori associati ad ogni placeholder tramite un particolare dizionario, passato come unico argomento. I placeholder sono parole chiave, che identificano un parametro in modo univoco, racchiuse tra parentesi graffe. Il comando .**format**() funziona quindi come segue:

```
data = {'first': 'Hodor', 'last': 'Hodor!'}
template = '{first} {last}'
result = template.format(**data)
```

Il risultato salvato in result sarà quindi la stringa 'Hodor Hodor!'.

3.2.2 *Cloud-init*

Cloud-init[1] è uno degli strumenti più utilizzati per l'inizializzazione e configurazione di server cloud. Tramite la compilazione di alcune semplici impostazioni, l'utente sarà in grado di avviare un nuovo server cloud specificando una serie di azioni da eseguire in automatico durante il primo avvio, come ad esempio eseguire determinati script, copiare alcuni file da remoto, installare pacchetti ed applicazioni necessarie, e così via. Cloud-init è installato di default su molte distribuzioni Linux, come Ubuntu, Fedora, Debian, CentOS, ecc. [8]

In poche parole, Cloud-init è un modulo che viene eseguito all'avvio di una macchina virtuale e permette di specificare delle azioni da eseguire tramite un file detto *user-data*. Un file di questo tipo, ovvero che permette di specificare una serie di azioni che verranno eseguite in automatico, viene detto *recipe* (ovvero "ricetta" in inglese). Infatti si parla di *cloud-init recipe* riferendosi ad una particolare configurazione da passare a cloud-init.

Per utilizzare una cloud-init recipe, è sufficiente specificare il file user−data generato quando si avvia il server sul cloud per la prima volta. Il comando da usare varia a seconda del Cloud Service Provider scelto. Per infrastrutture cloud basate su tecnologia OpenStack, ad esempio, è sufficiente eseguire il seguente comando da terminale:

```
$ nova boot −−image ubuntu−cloudimage −−flavor 1 −−user−data user−data
```

Tramite il file user−data è possibile passare al modulo cloud-init una serie di file in diversi formati supportati, tra i quali:

- file compresso in formato .gzip;
- file MIME (Multipurpose Internet Mail Extensions) multiparti;
- bash script;
- file cloud-config.

In particolare, i file gzip possono essere utili per ridurre le dimensioni del file user−data, essendo questo limitato a 16KB. I file MIME multiparti sono tipi di file composti che servono a raggruppare altri file, dei tipi sopra citati, in un

1 https://launchpad.net/cloud-init

unico file. I file di script servono ad eseguire una serie di comandi subito dopo il primo boot, mentre i file cloud-config sono particolari file utilizzati per copiare file in remoto sulla macchina sul cloud oppure per installare tutti i pacchetti aggiuntivi necessari.

Come vedremo nel Capitolo 4, le recipe cloud-init sono state molto utili per la fase di Cloud Deployment di Indico.

3.2.3 *Fabric*

Fabric è una "libreria Python e uno strumento da linea di comando per rendere più semplice l'uso di SSH (Secure Shell) per applicazioni di deployment o operazioni di amministrazione di sistema". [9]

Figura 12: Logo di Fabric.

Fabric fornisce una serie di comandi per l'esecuzione sia locale che remota di comandi shell (normali o tramite sudo), per l'upload o il download di file o per l'esecuzione di operazioni ausiliare, come richiedere un input all'utente o bloccare l'esecuzione di un'operazione in esecuzione.

Uno script fabric è quindi un semplice script python, dove però si possono utilizzare tutti i comandi messi a disposizione da Fabric. Due comandi fabric molto importanti sono local(cmd) e run(cmd). Entrambi i comandi eseguono il comando cmd passato come argomento come comando shell, l'unica differenza tra i due è che local(cmd) esegue cmd in locale, ovvero sulla macchina che sta eseguendo lo script fabric, mentre run(cmd) esegue cmd su una macchina remota, opportunamente specificata. Inoltre un altro comando molto utile è il comando put(**file**, dest) che permette di copiare il file **file**, che si trova sulla macchina locale, nella destinazione dest all'interno della macchina remota, quindi mimando il comportamento del comando scp di bash.

Per permettere una stesura più semplice e meno ridondante di uno script fabric, è possibile specificare alcune opzioni in un particolare dizionario, detto *ambiente* e denotato dalla variabile env, in modo da non doverle ripetere ogni volta che si invoca un comando fabric. Ad esempio specificando un valore per

il campo env.hosts possiamo definire una volta per tutte l'indirizzo di tutte le macchine remote su cui vogliamo eseguire i comandi passati a run(), senza dover stare a ripeterli inutilmente ogni volta.

Ogni script fabric deve specificare delle operazioni, dette *task*, che possono poi essere eseguite da linea di comando. Ogni task è una funzione python contrassegnata dal decoratore @task, che contrassegna le task (operazioni che possono essere eseguite dall'utente), da funzioni interne dello script.

Per poter essere eseguito, uno script fabric deve essere chiamato fabfile.py. Per eseguire quindi uno script fabric basta eseguire un comando della forma seguente:

```
$ fab task1 task2
```

Un comando come quello appena mostrato esegue prima la task task1 e quindi task2 definite nel file fabfile.py.

Fabric verrà utilizzato, all'interno del progetto di Cloud Deployment, per la stesura sia di uno script per la creazione automatica di immagini virtuali, che di uno script per la gestione remota di macchine cloud.

3.2.4 *QEMU e KVM*

QEMU (Quick EMUlator) è un emulatore e strumento di virtualizzazione open source. KVM (Kernel-based Virtual Machine) invece è anch'esso uno strumento di virtualizzazione e fornisce anche delle funzionalità di accelerazione hardware per sistemi Linux.

QEMU e KVM possono essere utilizzati in congiunzione, tant'è che KVM viene distribuito da anni assieme a QEMU. Si potrebbe pensare di utilizzare soltanto uno di questi due strumenti per lavorare con macchine virtuali, tuttavia, anche se QEMU fornisce un sistema di virtualizzazione completo e a se stante, per applicazioni pratiche è spesso necessario affiancargli KVM per migliorarne le performance. D'altro canto, KVM soltanto non fornisce tutte le funzionalità di un ambiente di virtualizzazione completo come QEMU.

QEMU e KVM sono stati utilizzati durante la fase di Cloud Deployment del progetto, ed in particolare nello script di creazione di immagini virtuali, come vedremo all'interno del Capitolo 4.

Per maggiori informazioni su QEMU e KVM si consultino [16] e [17].

4 CLOUD DEPLOYMENT

I concetti di *cloud* e *virtualizzazione* stanno prendendo sempre più piede, nel corso degli ultimi anni, nel settore informatico ed in particolare nel campo di sviluppo software.

L'idea alla base del cloud si basa sul semplice fatto che molte volte un utente, o un amministratore di sistema, può voler essere in grado di eseguire una certa applicazione, senza però doversi preoccupare dell'hardware necessario. L'unica cosa che interessa è poter eseguire il software desiderato senza bisogno di doversi preoccupare dei dettagli tecnici quali installare un server, configurarlo, o anche pensare alle varie problematiche relative all'hardware come la capacità degli hard disk necessaria, di quanti processori ha bisogno, di quanta banda per la trasmissione dei dati, ecc. Per rispondere a questa esigenza, sono nati i *Cloud Service Provider*, ovvero aziende in possesso di un grandissimo numero di macchine, molto capienti, molto potenti e molto veloci. In parole povere, quello che offrono i Cloud Service Provider, è di affittare le loro macchine per un certo costo mensile (o annuale, dipende dal tipo di contratto). Questi provider fanno scegliere all'utente il tipo di profilo desiderato: ci saranno ad esempio i bassi profili, che forniscono risorse limitate in cambio di un pagamento minimo, oppure alti profili, che forniscono all'utente un'altissima potenza di calcolo in cambio, ovviamente, di un pagamento più alto. L'utente si limita quindi a scegliere il profilo che più gli è consono e a "lanciare" (in inglese, *deploy*) la propria applicazione sulla macchina appena affittata. Non sarà necessario quindi fare alcuna installazione fisica di macchine o hardware, né tanto meno mantenerle: il Cloud Service Provider scelto si occuperà di tutto questo, mentre l'utente potrà concentrarsi sulla gestione e sull'utilizzo della propria applicazione. Ovviamente, dato che facendo così l'utente si "astrae" dal concetto di macchina fisica, egli non saprà mai con certezza su che specifiche hardware viene effettivamente eseguita la sua applicazione. La sua applicazione potrebbe venire eseguita su un supercomputer molto potente, o magari su un una macchina più semplice dedicata solo a quell'applicazione. L'utente non lo sa ma, d'altro canto, nemmeno gli interessa, avendo scelto di lanciare la sua applicazione sul cloud. Addirittura spesso può anche capitare che, a seconda del carico di lavoro delle varie macchine del provider, l'applicazione venga prima eseguita su delle macchine e in un secondo momento su delle altre. Da questo il cloud prende il suo nome: un'applicazione lanciata sul cloud non risiede necessariamente in una macchina specifica, ma la possiamo immaginare all'interno di una sorta di "nuvola", ovvero in uno spazio indefinito all'interno del quale però l'applicazione ha accesso a tutte le risorse hardware richieste.

L'idea della virtualizzazione è molto simile a quella del cloud ma orientata più al sistema operativo di una macchina. Sappiamo che spesso un'applicazione è ottimizzata per un certo sistema operativo o, addirittura, funziona soltanto se eseguita su determinati sistemi operativi. Per gli sviluppatori, spesso, è una scelta obbligata quella di prediligere alcuni sistemi operativi rispetto ad altri: infatti alcuni sistemi operativi possono essere talmente diversi tra loro che garantire la compatibilità dell'applicazione su tutti i sistemi operativi comporterebbe dover riprogettare e riscrivere l'applicazione da capo, il che non è sempre possibile, a seconda delle risorse disponibili per lo sviluppo. Per ovviare a questi problemi sono stati sviluppati degli appositi tools di virtualizzazione, come il ben noto Virtualbox di Oracle[1]. I tool di virtualizzazione permettono di simulare un sistema operativo eseguendolo su una macchina fisica sulla quale è installato un altro sistema operativo. Quindi è come se il tool di virtualizzazione simulasse un hardware che in realtà non c'è per dar modo all'utente di utilizzare un sistema operativo a sua scelta senza bisogno di doverlo installare sul disco fisso della macchina fisica. Questo permette di installare in modo molto veloce molti sistemi operativi diversi, qualora se ne avesse il bisogno. Ogni istanza creata tramite un tool di virtualizzazione prende il nome di *macchina virtuale*, detta anche Virtual Machine (VM) in inglese, e vengono spesso archiviate in appositi file (la cui estensione varia a seconda dello strumento di virtualizzazione scelto) che prendono il nome di *immagini virtuali*. Un altro vantaggio delle macchine virtuali è quindi la loro portabilità: se un utente volesse, infatti, utilizzare una certa macchina virtuale su un'altra macchina fisica, non dovrà far altro che copiarvi l'immagine virtuale relativa ed eseguirla sulla nuova macchina fisica tramite lo strumento di virtualizzazione.

Le tecniche di cloud e virtualizzazione sono quindi molto utili in quei frangenti in cui l'utente vuole solo occuparsi di poter eseguire la propria applicazione senza dover stare a preoccuparsi della configurazione hardware o del sistema operativo.

Con queste idee in mente, il primo obiettivo dell'Indico KT Project è stato proprio quello di poter adattare il software di Indico alle tecnologie di cloud e virtualizzazione. Può capitare, infatti, che un utente voglia installare ed utilizzare Indico per un breve periodo, senza stare a configurare un intero web server: a volte possono capitare utenti che vogliono utilizzare Indico per un solo evento o anche associazioni che intendono utilizzare Indico in modo continuativo ma che sono troppo piccole per occuparsi di installare e mantenere un server per conto proprio. Tutto questo era possibile già prima, ma se un utente aveva la necessità di installare Indico sul cloud doveva occuparsi personalmente di tutta la parte tediosa di installazione e configurazione, sia della macchina che di Indico, prima di poterne usufruire. Inoltre ogni interazione con la macchina sul cloud richiedeva l'utente di effettuare il login sulla macchina remota ed interagire

1 https://www.virtualbox.org/

tramite terminale. Analogamente, se un utente avesse voluto installare Indico su una macchina virtuale, non avrebbe avuto altra scelta che farlo manualmente, dovendosi occupare personalmente dell'installazione e della configurazione sia della macchina virtuale che di Indico.

I principali obiettivi di questo progetto erano quindi di automatizzare il deployment su struttura cloud, da un lato, e la creazione di immagini virtuali, dall'altro. Successivamente è stato anche creato uno script in python fabric per la gestione remota (ad esempio sul cloud) di una macchina con installato Indico.

Il progetto, disponibile sulla pagina Github ufficiale di Indico[1], è strutturato come segue:

- all'interno della cartella usr/ sono presenti lo script per la generazione del file user—data e lo script fabric per la gestione remota dei server cloud;
- nella cartella dev/ è presente lo script fabric per la generazione di immagini virtuali;
- la cartella tpl/ raccoglie tutti i template necessari ai vari script;
- nella cartella conf/ sono presenti alcuni file di configurazione.

Dal momento che al CERN Indico è installato su una macchina con sistema operativo Scientific Linux CERN 6 (SLC6)[2], ovvero una distribuzione di Linux sviluppata al CERN basata su Scientific Linux 6 (SL6)[3], si è deciso, per motivi pratici, di basare gli script di cloud deployment proprio su questa distribuzione di Linux. In linea teorica, gli script dovrebbero funzionare senza problemi anche per altre distribuzioni basate, come SLC6 e SL6, su Red Hat Enterprise Linux (RHEL). Per altre distribuzioni Linux potrebbero esser necessarie delle modifiche agli script, come ad esempio quando vengono invocati i comandi per l'installazione di pacchetti[4].

4.1 DEPLOYMENT CON CLOUD-INIT

Il primo obiettivo del progetto di cloud deployment di Indico era appunto quello di automatizzare l'installazione e la configurazione di un server sul cloud e di Indico. Per fare ciò abbiamo sfruttato le potenzialità di uno strumento di cloud configuration molto diffuso in ambiente Linux: il modulo cloud-init. Abbiamo già introdotto cloud-init in Sezione 3.2, a pagina 25, quindi eviteremo di ripetere qui a cosa serve cloud-init e quali sono le sue funzionalità. Parleremo invece

1 https://github.com/indico/indico-cloud-images.

2 http://linux.web.cern.ch/linux/scientific6/.

3 Una distribuzione di Linux sviluppata dal Fermi National Accelerator Laboratory (FNAL), disponibile all'indirizzo: https://www.scientificlinux.org/.

4 Il comando per sistemi RHEL è yum mentre per sistemi, ad esempio, Ubuntu Linux è necessario utilizzare il comando apt—get.

di come è stato utilizzato cloud-init per il cloud deployment automatizzato di Indico.

Le idee alla base dell'automatizzazione del cloud deployment di Indico possono essere riassunte dalle seguente necessità che un utente, in procinto di installare Indico su un nuovo server cloud, si trova a voler soddisfare:

- la necessità di configurare da zero, in poco tempo, un nuovo server cloud con Indico già installato e pronto all'uso;
- la necessità di poter ripetere questo processo in modo automatico;
- la necessità di parametrizzare alcune parti del processo di configurazione (sia del server che dell'applicazione);
- la necessità di fare tutto questo in modo sicuro e affidabile.

Come abbiamo già accennato in Sezione 3.2, la risposta a queste necessità è cloud-init.

Abbiamo già detto che cloud-init funziona passando un apposito file, o ricetta, detto user−data, al comando che si occupa di avviare il server cloud in remoto. Ovviamente il comando specifico varia a seconda del Cloud Service Provider scelto, ma solitamente i comandi dei provider che supportano cloud-init presentano un'opzione tramite la quale è possibile specificare il file user−data contenente tutte le informazioni ed istruzioni necessarie a configurare la nuova macchina cloud.

Il lavoro necessario, quindi, per automatizzare il tutto, si è risolto con lo scrivere uno script per la generazione del file user−data. Andiamo ad analizzare come è composto questo script ed il file user−data risultante.

Il file user−data necessario ai nostri fini è un file MIME multiparti, ovvero un file in grado di raggruppare una serie di altri file. Il file MIME generato è così composto:

- uno script bash per eseguire le istruzioni richieste;
- un file cloud-config per copiare i file necessari.

Lo script python che genera il file user−data, denominato gen−user−data.py, è quindi suddiviso in quattro fasi principali: la fase di configurazione, le fasi di generazione dello script bash e del file cloud-config ed infine la fase di generazione del file user−data vero e proprio.

4.1.1 *Fase di configurazione*

La fase di configurazione consiste semplicemente in una serie di domande mostrate su linea di comando, ognuna delle quali serve a far scegliere all'utente

tutti quei parametri necessari per personalizzare la propria installazione di Indico sul nuovo server cloud. Tra i parametri che l'utente può scegliere ci sono ad esempio i percorsi delle varie directory di installazione di Indico, o le porte e gli indirizzi ai quali il web server di Indico sarà raggiungibile, o ancora i certificati SSL (Secure Sockets Layer) da utilizzare.

L'utente potrà scegliere, ad ogni domanda, di utilizzare il valore di default suggerito, oppure di specificare un nuovo valore per quel parametro. Inoltre, potrà anche scegliere di generare un file di configurazione, contenente tutti i valori scelti, in modo da poter rieffettuare, in futuro, lo stesso processo di deployment utilizzando gli stessi valori senza bisogno di doverli reinserire una seconda volta.

Eseguendo lo script gen−user−data.py l'utente si troverà quindi a dover impostare i seguenti parametri (si noti che per ogni voce è presentato un valore di default, per il quale basta premere Enter):

```
$ python gen−user−data.py
Do you want to use a configuration file [y/N]?
Insert the Indico installation directory path [/opt/indico]:
Insert the Indico DB installation directory path [/opt/indico/db]:
Insert the Apache conf directory [/etc/httpd/conf]:
Insert the Apache confd directory [/etc/httpd/conf.d]:
Insert the SSL cert directory [/etc/ssl/certs]:
Insert the SSL private directory [/etc/ssl/private]:
Do you want to load a personal SSL certificate [y/N]?
Insert the http port [80]:
Insert the https port [443]:
Insert the hostname:
Insert the iptables path [/etc/sysconfig/iptables]:
Insert the Redis hostname [localhost]:
Insert the Redis port [6379]:
Insert the Redis password:
Do you want to use Postfix as mail server [Y/n]?
Insert the SMTP server port [25]:
Insert the SMTP login:
Insert the SMTP password:
Insert the YUM repositories directory [/etc/yum.repos.d]:
Insert the priority for the puias−unsupported repository [19]:
Do you want to generate a configuration file [Y/n]?
Specify the configuration file path [gen−user−data.conf]:
Choose a path for the MIME file [user−data]:
```

Al termine del processo, i valori vengono comunque salvati all'interno di un dizionario python, che sarà poi utilizzato, assieme ai vari template, per generare i file necessari.

4.1.2 *Generazione dello script bash*

Come già accennato, la generazione dello script bash da includere nel file user−data si basa su un template, chiamato user−data−script.sh, e sui valori dei parametri scelti dall'utente. Questo è necessario in quanto alcune parti dello script sono parametrizzate e devono essere compilate in base alle scelte dell'utente.

Per ottenere lo script finale, basterà allora eseguire il comando python .**format** () su ogni linea del template passando come unico argomento il dizionario, creato nella fase precedente, dei parametri. In particolare il template dello script è composto da tutte le istruzioni dello script finale ma in corrispondenza di ogni parametro vi sarà invece un particolare placeholder che sta a indicare dove il comando .**format**() dovrà andare a sostituire i valori effettivi dei parametri.

Le principali azioni intraprese dallo script, una volta avviata la macchina per la prima volta, sono:

1. scaricare e installare tutti i pacchetti necessari ad Indico;
2. installare e configurare Indico;
3. installare i certificati SSL;
4. aprire le porte scelte per il web server;
5. copiare i file di configurazione nelle cartelle corrispondenti.

Avendo basato lo script su SL6, l'installazione di pacchetti aggiuntivi avviene invocando il comando yum. L'installazione e la configurazione di Indico avvengono tramite i comandi easy_install indico e indico_initial_setup, rispettivamente. La copia dei certificati SSL e dei file di configurazione e l'apertura delle porte, invece, avvengono tramite semplici comandi per la manipolazione di file in ambiente Linux.

I problemi principali riscontrati durante la stesura dello script riguardavano tutti l'impossibilità (apparente) di rendere automatiche alcune azioni. Per alcune parti dello script sono stati infatti necessari alcuni accorgimenti per rendere il procedimento pienamente automatico.

Per quanto riguarda l'istallazione tramite comando yum, ad esempio, si è dovuta aggiungere l'opzione −y da linea di comando, per evitare che yum chiedesse conferma all'utente di voler effettivamente installare i pacchetti scelti, rimanendo ad aspettare all'infinito.

Un'altra problematica era legata al comando di configurazione di Indico, indico_initial_setup, che richiede all'utente di scegliere alcuni valori per configurare correttamente Indico. Nel nostro caso questi valori sono già stati scelti durante la fase di configurazione esposta prima, quindi devono essere passati in modo automatico al comando indico_initial_setup. La soluzione è far stampare questi valori al terminale tramite il comando **echo** e quindi concatenare **echo** con indico_initial_setup. Il risultato è il comando seguente:

```
$ echo -e "{indico_inst_dir}\nc\ny\n{db_inst_dir}" | indico_initial_setup
```

Si notino i due template {indico_inst_dir} e {db_inst_dir} che stanno a indicare, rispettivamente, il percorso in cui l'utente vuole installare Indico e in cui vuole installare il DB.

Infine è sorto il problema di dover far eseguire alcune istruzioni allo script come utente root, ovvero tramite il comando sudo. Il problema è che lo script non viene eseguito dall'utente, ma dal modulo cloud-init all'avvio della macchina sul cloud, senza possibilità di avere permessi da root. La soluzione è stata trovata utilizzando il comando visudo e andando a modificare il file etc/sudoers per permettere di eseguire sudo anche in assenza di TTY (Teletypewriter), ovvero una console. Il risultato[1] è il seguente frammento di codice che abilita il comando sudo anche all'interno di script cloud-init:

```
touch /etc/sudoers.tmp
cp /etc/sudoers /tmp/sudoers.new
find_replace /tmp/sudoers.new "Defaults requiretty" "Defaults !requiretty"
visudo -c -f /tmp/sudoers.new
if [ "$?" -eq "0" ]; then
    cp /tmp/sudoers.new /etc/sudoers
fi
rm /etc/sudoers.tmp
```

Ricapitolando, dopo aver generato lo script effettivo sostituendo i parametri ai rispettivi placeholder nel template, e con le dovute accortezze per rendere il tutto completamente automatico, il risultato è uno script che, durante il primo avvio della macchina sul cloud, si occuperà di effettuare tutte le azioni necessarie ad installare e configurare Indico, senza che l'utente debba fare niente. L'unica cosa di cui ha bisogno lo script è che i vari file di configurazione necessari ad Indico siano già stati copiati sulla macchina: a questo penserà il file cloud-config, generato nella prossima fase.

4.1.3 *Generazione del file cloud-config*

Dopo aver generato lo script bash da inserire nel file MIME multiparti finale, lo script gen-user-data.py si occupa di generare il file cloud-config, necessario a copiare file sulla macchina cloud che vogliamo avviare. Come per lo script generato nella fase precedente, anche la generazione del file cloud-config si basa su dei template e sull'uso del comando .**format**().

Il template del file cloud-config ha la seguente struttura:

1 Si vedano http://serverfault.com/questions/324415/running-sudo-commands-in-cloud-init-script e http://stackoverflow.com/questions/323957/how-do-i-edit-etc-sudoers-from-a-script.

```
#cloud-config

write_files:
- content: |
{puias_repo_content}
    path: /puias.repo
- content: |
{indico_httpd_conf_content}
    path: /indico_httpd.conf
- content: |
{indico_indico_conf_content}
    path: /indico_indico.conf
- content: |
{redis_conf_content}
    path: /redis.conf
- content: |
{ssl_conf_content}
    path: /ssl.conf
{ssl_files}
```

Si vede quindi che per ogni file che dev'essere copiato è presente, nel template, una riga `- content: |`, che serve a delimitare l'inizio di un nuovo file. A seguito è presente un placeholder, che verrà sostituito con il contenuto del file in questione, ed infine la voce `path:`, nella quale si indica in che percorso della macchina cloud vorremmo copiare il file. La struttura del file cloud-config è riconosciuta tramite l'indentazione: la stringa `- content: |` non dev'essere indentata, il contenuto del file dev'essere indentato di 8 spazi mentre il percorso finale del file dev'essere indentato di 4 spazi.

Per comodità, tutti i file vengono inizialmente copiati nella cartella root / del sistema: ci penserà poi lo script bash generato alla fase precedente a copiare i file nelle rispettive cartelle lavorando direttamente sulla macchina cloud.

Dal momento che alcuni dei parametri scelti durante la prima fase potrebbero essere necessari anche all'interno dei file da copiare, sono stati scritti dei template per ognuno di questi file. Prima di generare il file cloud-config vero e proprio è quindi necessario generare i vari file che si vogliono copiare, tramite i corrispettivi template ed il comando .**format**(), come visto prima.

I file da copiare sono file di configurazione, come il file di configurazione di Indico, quello per la configurazione di Apache, o i certificati SSL.

Una volta generati tutti i file necessari a partire dai template e dai parametri scelti nella prima fase, lo script leggerà il contenuto di ogni file e lo scriverà, con la dovuta indentazione, al posto del placeholder corrispondente nel template del file cloud-config. Il risultato finale sarà quindi il file cloud-config, riempito con il contenuto dei vari file che vogliamo copiare i quali, a loro volta, sono stati compilati con i parametri scelti dall'utente.

4.1.4 *Generazione del file user−data*

Quando finalmente sono stati generati sia lo script bash che il file cloud-init, lo script principale può procedere all'ultima fase, ovvero mettere i due file insieme scrivendoli in un file MIME multiparti, che rappresenterà il file user−data finale.

Per generare user−data, creando un nuovo file MIME multiparti a partire dai due file generati nelle fasi precedenti, si è usato uno script reso disponibile dagli sviluppatori di cloud-init[1], detto write−mime−multipart.

Questo comando, molto semplice, prende come input tutti i file che vogliamo includere nel file finale e richiede anche di specificare il nome e percorso del file che vogliamo venga generato. Quindi, nel nostro caso, il comando finale sarà il seguente:

```
$ ./write−mime−multipart −−output user−data user−data−script.sh cloud−config
```

Chiaramente, lo script principale per la generazione del file user−data si occuperà di invocare il comando precedente, assicurandosi di passargli i valori corretti per i vari input. Dopodiché lo script terminerà.

Una volta che lo script avrà terminato con successo, l'utente potrà utilizzare il file user−data generato per inizializzare e configurare una nuova macchina su cloud in modo completamente automatico, semplicemente specificando il file user−data come argomento del comando di boot. Come già accennato in Sezione 3.2, il comando di boot effettivo, ed il metodo in cui si passa il file user−data, cambia a seconda del Cloud Service Provider scelto.

4.2 CREAZIONE DI IMMAGINI VIRTUALI

Il secondo obiettivo del progetto di Cloud Deployment è stato scrivere uno script per la creazione automatica di immagini virtuali già configurate per poter eseguire Indico.

Lo script in questione è uno script python fabric che si occuperà di avviare una nuova macchina virtuale, usando due strumenti di virtualizzazione (QEMU e KVM), e di effettuare le necessarie operazioni all'interno di essa. Tutte i parametri che l'utente può scegliere, come nome e percorso dell'immagine di base o le porte da aprire sulla macchina virtuale, sono raccolti all'interno di un file esterno allo script, chiamato fabfile.conf, in modo che l'utente li possa cambiare in modo semplice e veloce.

Le task specificate da questo script fabric sono le seguenti:

- create_vm_image: crea un'immagine virtuale con Indico installato e configurato;

1 Reperibile all'indirizzo `https://github.com/lovelysystems/cloud-init/blob/master/tools/write-mime-multipart`

- run_vm_debug: avvia la macchina virtuale e Indico (in modalità debug);
- config_no_cloud: configura il file di configurazioni "fasullo" no-cloud;
- launch_vm: avvia la macchina virtuale;
- start: avvia Indico;
- deploy: esegue tutte le installazioni e configurazioni necessarie per far funzionare Indico sulla macchina virtuale;
- config: configura Indico e la macchina virtuale;
- vm_config: configura la macchina virtuale;
- indico_config: configura Indico;
- indico_inst: installazione e setup di indico;
- dependencies_inst: installazione delle dipendenze.

La task principale è ovviamente create_vm_image, che restituisce un'immagine virtuale con indico Installato e funzionante pronta ad essere avviata o caricata su una struttura cloud. Tutte le altre task sono sotto-task di create_vm_image che, all'occorrenza, possono essere invocate in modo autonomo.

create_vm_image è divisa in due fasi: nella prima fase si avvia una nuova macchina virtuale a partire da un'immagine virtuale di base; nella seconda si fanno tutte le installazioni e configurazioni necessarie per poter usare Indico sulla macchina virtuale.

4.2.1 *Avvio della macchina virtuale*

La fase di avvio della macchina virtuale si divide, a sua volta, in due parti: configurazione di un'immagine .iso no-cloud e boot della macchina virtuale.

La fase no-cloud serve creare un'immagine .iso di configurazione per poter avviare la macchina virtuale anche al di fuori da una struttura cloud. Questa fase è necessaria in quanto il sistema operativo SLC6, che è stato utilizzato per sviluppare e testare lo script di creazione di immagini virtuali, è un sistema operativo pensato per operare in ambiente cloud e non su ambienti di virtualizzazione locali. Nel nostro caso, invece, l'idea è di avviare una nuova macchina virtuale in locale (usando uno strumento di virtualizzazione) e di effettuare tutte le operazioni necessarie tramite uno script fabric. Quindi è necessario, quando si avvia per la prima volta questa macchina virtuale, "ingannarla" in qualche modo, facendole ignorare il fatto che non viene eseguita in ambiente cloud. Per fare questo utilizziamo la funzionalità no-cloud offerta dal modulo cloud-init[1]

1 Si veda http://cloudinit.readthedocs.org/en/latest/topics/datasources.html#no-cloud.

che abbiamo descritto nella Sezione precedente. Senza scendere nel dettaglio, questa fase genera due file, user−data e meta−data, e li archivia all'interno di un'immagine in formato .iso. Quest'immagine, che chiameremo init.iso, verrà passata, nella fase successiva, al comando di boot della macchina virtuale per poterla avviare senza problemi anche in ambienti di virtualizzazione locali.

Una volta creato il file init.iso per attivare la funzionalità no-cloud, lo script potrà finalmente avviare la macchina virtuale. Il comando per l'avvio è il seguente:

```
$ kvm −m 256 −redir tcp:2222::22 −net nic −net user, −drive file=slc6_cern_x86_64.
    qcow2,if=virtio −drive file=init.iso,if=virtio −serial file:qemu−output.log &
```

Con l'opzione −drive file= si specificano le immagini virtuali da usare come input: in questo caso utilizziamo slc6_cern_x86_64.qcow2, che è l'immagine virtuale di base con installato SLC6, e init.iso, che permette di attivare il no-cloud. L'opzione −redir serve ad associare le porte specificate, in modo da permettere la comunicazione tra la macchina locale e la macchina virtuale tramite quelle porte (nello specifico quelle sono le porte relative al protocollo SSH). Infine, l'opzione −serial file: permette di specificare un file di log, dove verrà salvato l'output del terminale della macchina virtuale, mentre il simbolo & finale serve a lanciare il comando in background, in modo da lasciare la shell libera e non bloccare l'esecuzione dello script.

Il file di log passato al comando kvm è molto importante, in quanto consente allo script fabric di sapere quando la macchina virtuale ha terminato la fase di booting, in modo da poter proseguire ed effettuare le azioni sulla macchina necessarie. Lo script fabric, infatti, non può eseguire azioni come installare pacchetti aggiuntivi o installare Indico finché la macchina si sta ancora avviando. D'altro canto, la macchina virtuale non ha alcun modo diretto per comunicare allo script fabric quando la fase di booting è finita. L'idea è stata quindi quella di far loggare l'output del terminale della macchina virtuale sul file di log specificato e far leggere il file allo script fabric in continuazione finché non viene rilevata la fine del boot, nel qual caso lo script può continuare la sua esecuzione. Nello specifico, per capire quando la fase di boot è terminata o meno, è bastato fare un boot di prova e andare a vedere qual era l'ultima riga scritta sul file di log durante la fase di booting e quindi far controllare allo script fabric se tale stringa è presente o meno nel file di log: se la stringa compare nel log, allora lo script può procedere con l'esecuzione, altrimenti aspetta 5 secondi e poi ricontrolla. Il comando relativo a questa procedura è il seguente:

```
$ while ! grep −q "Starting atd:.*[.*OK.*]" "qemu−output.log"; do sleep 5; done
```

Prima di passare alla fase successiva, osserviamo che l'immagine di base, nell'esempio slc6_cern_x86_64.qcow2, deve essere ottenuta dall'utente in maniera

autonoma, ad esempio scaricandola tramite il sito ufficiale del sistema operativo scelto.

4.2.2 *Configurazione della macchina virtuale*

Durante la fase di configurazione della macchina virtuale lo script esegue tutte quelle istruzioni necessarie ad installare pacchetti aggiuntivi e ad installare e configurare Indico e la macchina virtuale affinché Indico sia pienamente utilizzabile al termine.

Le istruzioni eseguite in questa fase dallo script fabric sono quindi del tutto analoghe e speculari a quelle eseguite dallo script bash che avevamo descritto nella Sezione 4.1 per il deployment di Indico su cloud tramite cloud-init. Non staremo quindi a ripetere le azioni intraprese dallo script in questa fase, essendo del tutto uguali a quelle dello script bash già descritto.

Le uniche differenze riguardano il fatto che, in questo caso, le istruzioni sono eseguite tramite uno script fabric. Quindi tutte le istruzioni eseguite dallo script bash in modo diretto dovranno adesso essere eseguite dallo script fabric in remoto, ovvero tramite il comando `run()`. Inoltre, non sarà più necessario, come prima, aver bisogno di copiare i file sulla macchina virtuale tramite meccanismi come cloud-config (sempre visto nella Sezione precedente): in questo caso, grazie a Fabric, sarà sufficiente eseguite il comando `put()` che si occuperà si copiare i file richiesti al momento in cui lo script lo richiede.

Riguardo al comando `put()` è stata tuttavia necessaria una modifica in quanto esso non supporta, al momento, di passare un link simbolico come percorso di destinazione del file. Per aggiungere il supporto a link simbolici, abbiamo inserito la seguente variante del comando, da utilizzare al posto di `put()`:

```
def _putl(source_file, dest_dir):
    """
    To be used instead of put, since it doesn't support symbolic links
    """

    put(source_file, '/')
    run("mkdir -p {0}".format(dest_dir))
    run("mv -f /{0} {1}".format(os.path.basename(source_file), dest_dir))
```

Al termine dell'esecuzione dello script, l'immagine virtuale di base risulterà aggiornata con l'installazione e la configurazione di Indico (e di tutte le componenti necessarie ad Indico) e sarà pronta per essere utilizzata in locale (tramite strumenti di virtualizzazione) o su una struttura cloud.

4.3 SCRIPT DI GESTIONE REMOTA

L'ultima parte di questo progetto riguarda lo script di gestione remota di macchine cloud per l'esecuzione di Indico. Questo script non era previsto tra gli obiettivi iniziali del progetto, ma è stato scritto, per motivi pratici, durante la stesura degli altri due script e alla fine, data la sua utilità, si è deciso di includerlo nella versione finale del codice del progetto di Cloud Deployment.

Lo script di gestione remota è uno script fabric che definisce delle task per la gestione e la configurazione in remoto di Indico e di tutte le componenti ad esso connesse. Può capitare, ad esempio, che l'utente decida, in un certo momento, di voler cambiare l'indirizzo del web server su cui gira Indico, o le porte tramite le quali si accede ai vari servizi, o magari vuol semplicemente arrestare Indico e farlo ripartire in un secondo momento. Tutto questo sarebbe già possibile, ma sarebbe necessario che l'utente acceda alla macchina remota, ad esempio tramite SSH, e dal terminale esegua i vari comandi manualmente. Questa procedura può essere tediosa e complicata e potrebbe anche dar luogo a errori. Per questo motivo è stato scritto questo script di gestione remota che definisce una serie di task, che l'utente può eseguire, che rappresentano i comandi principali che l'utente può voler eseguire sulla macchina remota e su Indico.

Di seguito elenchiamo le task implementate, senza scendere ulteriormente in dettaglio:

- restart: permette di riavviare una o più componenti di Indico (valori accettati sono redis, db, httpd e postfix);
- start: avvia una o più componenti di Indico (accetta gli stessi valori di restart);
- config: configura la macchina virtuale con tutte le informazioni necessarie;
- update_smtp: aggiorna la configurazione di Indico per SMTP (Simple Mail Transfer Protocol);
- update_redis: aggiorna la configurazione di Indico riguardante Redis;
- update_server: aggiorna le configurazioni del web server (come hostname e porte);
- load_ssl: carica dei nuovi certificati SSL.

5

DISTRIBUZIONE E PACKAGING

6

INSTANCE TRACKER

7

CONFERENCE CUSTOMIZATION PROTOTYPE

Parte III

NOTE E CONCLUSIONI

8

ALTRI PROGETTI

9

CONCLUSIONI

BIBLIOGRAFIA

[1] CERN. Your carreer, 2015. URL http://jobs.web.cern.ch/content/your-career. (Citato a pagina 11.)

[2] CERN. Culture and values, 2015. URL http://jobs.web.cern.ch/content/culture-and-values. (Citato a pagina 10.)

[3] CERN. The structure of cern, 2015. URL http://home.web.cern.ch/about/structure-cern. (Citato a pagina 12.)

[4] IT Dept. Indico service description. URL http://information-technology.web.cern.ch/services/fe/indico.

[5] Pedro Ferreira. The road to indico 2.0. Aprile 2015. URL https://indico.cern.ch/event/304944/session/6/contribution/61/material/slides/0.pdf. (Citato a pagina 18.)

[6] Brandon Fuller. The Beauty of CloudInit, Maggio 2011. URL http://brandon.fuller.name/archives/2011/05/02/06.40.57/.

[7] Sriram S. KVM and QEMU – do you know the connection?, Marzo 2014. URL http://www.innervoice.in/blogs/2014/03/10/kvm-and-qemu/.

[8] CloudInit Team. Cloud-init Documentation, . URL https://cloudinit.readthedocs.org/en/latest/index.html. (Citato a pagina 25.)

[9] Fabric Team. Fabric Documentation, . URL http://docs.fabfile.org/en/1.10/. (Citato a pagina 26.)

[10] Indico Team. Indico's user guide, . URL https://indico.cern.ch/ihelp/html/UserGuide/index.html.

[11] Indico Team. ZODB, Settembre 2010. URL http://old.indico-software.org/wiki/Dev/Technical/DB. (Citato a pagina 18.)

[12] Indico Team. Template engines, Febbraio 2011. URL http://old.indico-software.org/wiki/Dev/Technical/TemplateEngines. (Citato a pagina 18.)

[13] Indico Team. WSGI, Febbraio 2011. URL http://old.indico-software.org/wiki/Dev/Technical/WSGI. (Citato a pagina 18.)

[14] Indico Team. Flask, Luglio 2013. URL http://old.indico-software.org/wiki/Dev/Technical/Flask/Usage. (Citato a pagina 18.)

[15] Indico Team. Indico features, 2014. URL http://indico.github.io/features/.

[16] KVM Team. KVM Wiki, . URL http://www.linux-kvm.org/page/Main_Page. (Citato a pagina 27.)

[17] QEMU Team. QEMU Wiki, . URL http://wiki.qemu.org/Main_Page. (Citato a pagina 27.)

[18] Wikipedia. CERN, Aprile 2015. URL http://en.wikipedia.org/wiki/CERN.

[19] Wikipedia. CP violation, Aprile 2015. URL http://en.wikipedia.org/wiki/CP_violation.

[20] Wikipedia. NA48 experiment, Marzo 2015. URL http://en.wikipedia.org/wiki/NA48_experiment.

[21] Wikipedia. W and Z bosons, Aprile 2015. URL http://en.wikipedia.org/wiki/W_and_Z_bosons.

[22] Wikipedia. Higgs boson, Maggio 2015. URL http://en.wikipedia.org/wiki/Higgs_boson.