I server di produzione

Indice generale

I server di produzione

Installa le applicazioni con un clic. Controlla i server nel cloud. Configura server My Cloud o Self Managed.

Introduzione

Dopo aver concluso la fase di sviluppo delle proprie applicazioni, è necessario pubblicarle per renderle disponibili agli utenti.

L'operazione di pubblicazione varia in funzione del tipo di applicazione: nel caso di applicazioni mobile essa richiede un launcher, negli altri casi è necessario un server di produzione. Alcune applicazioni mobile richiedono sia il launcher che l'installazione di un sistema di backend su un server di produzione.

In questo libro vengono descritti i tipi di server disponibili e viene indicato come effettuare l'installazione e il controllo degli stessi in funzione del loro tipo.

Tipi di server di produzione

I server utilizzabili per la pubblicazione delle applicazioni sviluppate con Instant Developer Cloud sono di tre tipi:

1) Server Instant Developer Cloud (denominati anche Server App IDC).
2) Server My Cloud.
3) Server Self Managed.

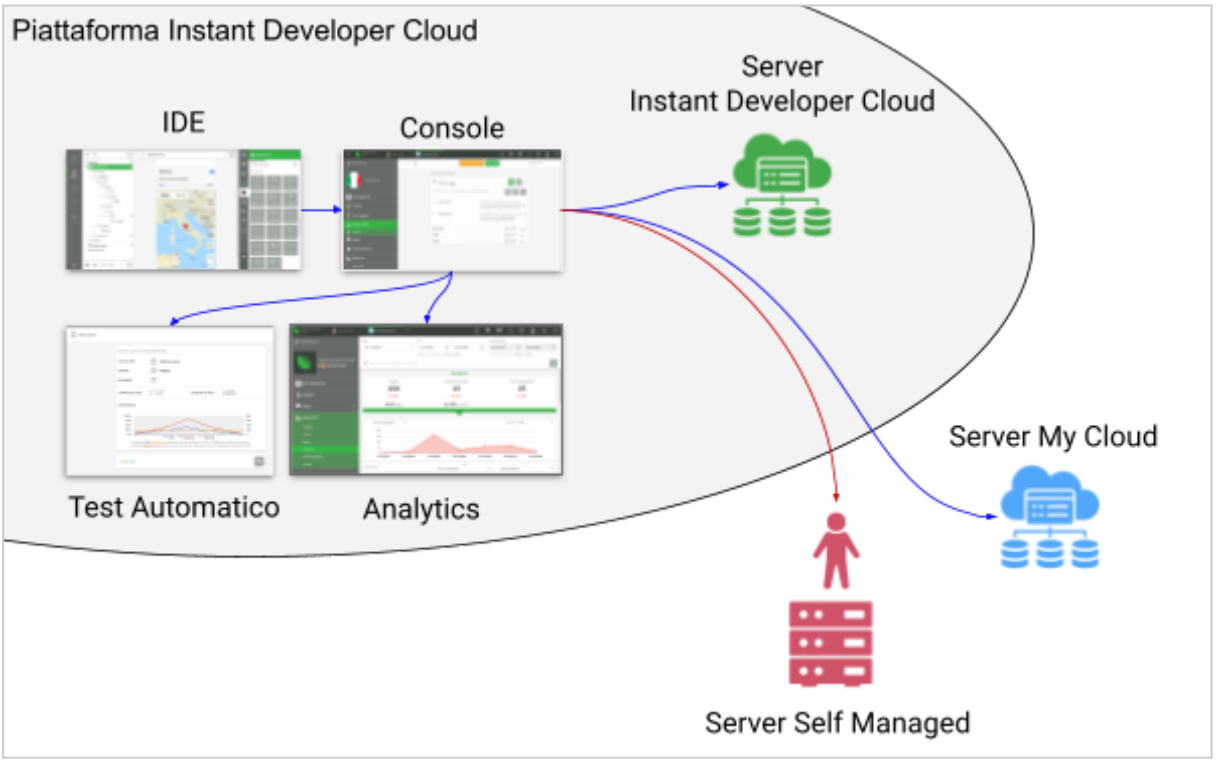
I server Instant Developer Cloud sono parte integrante della piattaforma e si trovano in datacenter di Google Cloud Platform situati nell'Unione Europea. Possono essere attivati tramite la console di Instant Developer Cloud, da cui vengono automaticamente resi disponibili nel giro di qualche secondo. L'intera gestione del server e delle applicazioni in esso installate è automatica e avviene sempre tramite la console.

I server My Cloud sono server cloud di proprietà dell'utilizzatore di Instant Developer Cloud che vengono collegati alla console. La preparazione e la gestione del server sono completamente manuali, mentre l'installazione e la gestione delle applicazioni vengono automatizzate tramite la console. Sono previsti costi per i servizi di installazione e di gestione delle applicazioni che variano a seconda della dimensione del server.

I server Self Managed sono server completamente autogestiti dall'utilizzatore di Instant Developer Cloud e di proprietà dell'utilizzatore stesso. Sia la preparazione che la gestione del server, delle applicazioni e dei database sono completamente manuali. In questo caso non sono previsti costi di gestione, in quanto il server non è collegato in alcun modo alla console di Instant Developer Cloud.

Per utilizzare i server My Cloud e Self Managed è necessaria la conoscenza dell'ambiente Node.js. A questi tipi di server si applicano le limitazioni elencate nella Tabella comparativa.

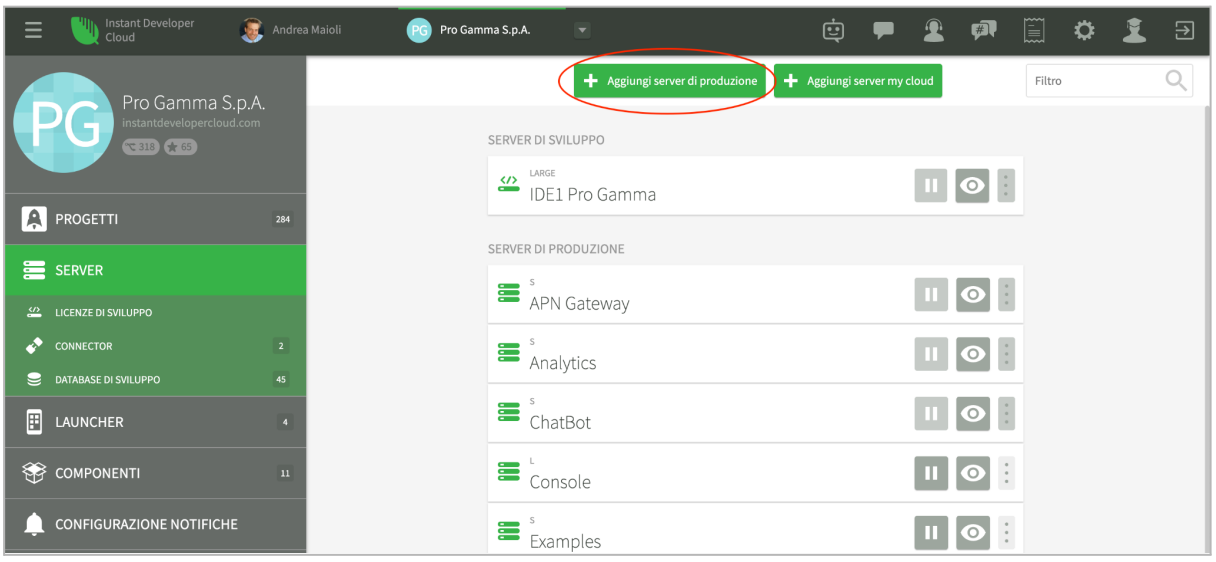
Il seguente schema mostra l'architettura del cloud nei vari tipi di configurazione:



I server di Instant Developer Cloud

I server di produzione di Instant Developer Cloud sono server integrati nella piattaforma e gestiti completamente tramite la console.

Per creare un nuovo server di produzione, è possibile accedere alla pagina server della propria organizzazione e poi utilizzare il pulsante *Aggiungi server di produzione*.



A questo punto si apre una schermata per la selezione della taglia del server, in cui è possibile selezionarne una tra le seguenti:

- Taglia **S**: 1 CPU condivisa, 1,7 GB di RAM, 10 GB disco SSD, 50 GB Egress mensile, fino a tre applicazioni installabili.
- Taglia **M**: 1 CPU dedicata, 3,75 GB di RAM, 20 GB disco SSD, 100 GB Egress mensile, fino a sei applicazioni installabili.
- Taglia **L**: 2 CPU dedicate, 7,5 GB di RAM, 40 GB disco SSD, 200 GB Egress mensile, fino a dieci applicazioni installabili.
- Taglia **XL**: 4 CPU dedicate, 15 GB di RAM, 80 GB disco SSD, 400 GB Egress mensile, fino a venti applicazioni installabili.
- Taglia **XXL**: 8 CPU dedicate, 30 GB di RAM, 160 GB disco SSD, 800 GB Egress mensile, fino a quaranta applicazioni installabili.
- Taglia **XXXL**: 16 CPU dedicate, 60 GB di RAM, 320 GB disco SSD, 1600 GB Egress mensile, fino a ottanta applicazioni installabili.

Per proseguire è necessario anche indicare la carta di credito su cui effettuare gli addebiti mensili. Se non è possibile selezionarla, occorre prima definire un profilo di fatturazione nella console.

Dopo aver selezionato la taglia è possibile cliccare il pulsante *Continua* in fondo alla pagina per accedere alla pagina dei servizi aggiuntivi. Sono presenti i seguenti servizi:

- **Sincronizzazione**: permette al server di funzionare come sistema di sincronizzazione per i database locali delle applicazioni mobile.
- **Cloud Connector**: permette al server di integrare dati e servizi provenienti da risorse on premise.
- **Analytics & Feedback**: permette di installare sul server i servizi di raccolta dati analitici e di feedback dei clienti.
- **Backup Giornaliero / Backup orario**: permette di mantenere automaticamente uno snapshot dell'intero server per ogni giorno o ogni ora degli ultimi 30 giorni.

Dopo aver selezionato gli eventuali servizi aggiuntivi, è possibile cliccare il pulsante *Verifica il tuo ordine* in fondo alla pagina per accedere al riepilogo e poi concludere l'acquisto.

Struttura di un server di produzione di Instant Developer Cloud

Un server di produzione è basato su un'istanza di macchina virtuale del servizio Compute Engine di Google Cloud Platform, in un datacenter situato nell'Unione Europea.

Il sistema operativo è di tipo linux gestito direttamente da Google e ottimizzato per l'utilizzo di container. In ogni istanza viene caricata un'immagine container che dipende dalla versione di Instant Developer Cloud in uso.

Nella versione 22.0, ad esempio, l'immagine contiene un sistema operativo Linux Alpine 3.13 in configurazione minima, Node.js 16.4, Postgres 13.3, Let's encrypt, ImageMagick, e Puppettier.

Le uniche porte aperte sono la 80 (http) e la 443 (https). La porta 80 serve solo per la redirezione http/https.

Tutti i server di Instant Developer Cloud sono dotati di due dischi. Il primo è il disco del sistema operativo, che contiene anche l'immagine container. Il secondo, di dimensione variabile in base alla taglia e di tipo SSD, contiene i dati, cioè:

- I database Postgres.
- Le applicazioni installate.
- Il file system di ogni applicazione installata.

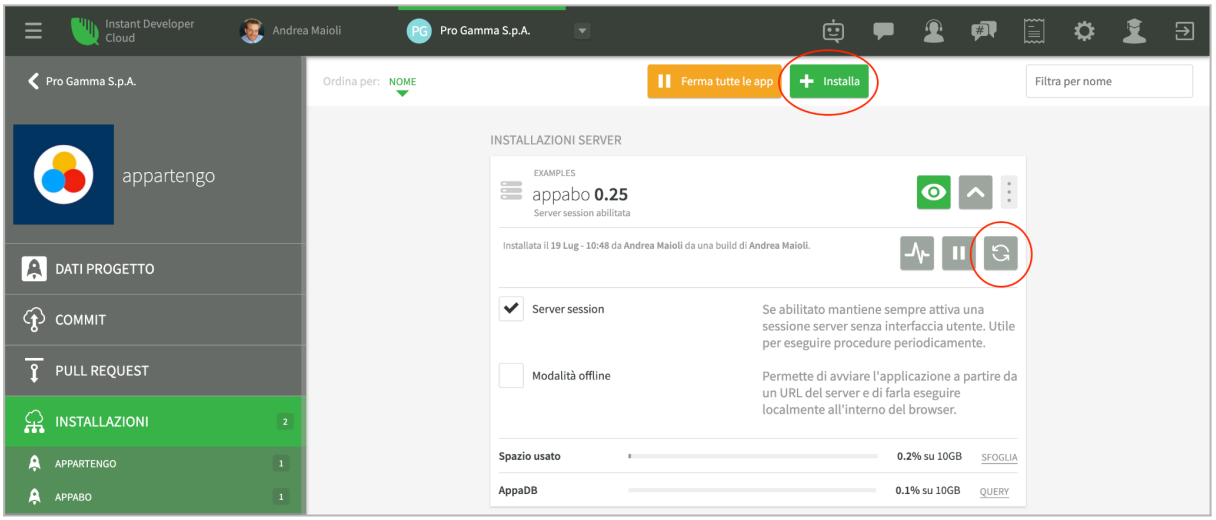
I dischi presenti sui server sono nativamente crittografati tramite Google Cloud Platform.

Il database Postgres presente nel container accetta solo connessioni locali e l'elenco degli account è gestito automaticamente dal sistema. Non è quindi possibile conoscere le password degli utenti.

Installazione di un'applicazione

Vediamo ora come funziona il processo di installazione delle applicazioni su un server di produzione di Instant Developer Cloud.

Per installare una nuova applicazione su un server di produzione è necessario cliccare il pulsante *+Installa* nella pagina *Installazioni* del menu di progetto.



Dopo aver iniziato il processo di installazione apparirà una serie di tre popup. Nel primo occorre selezionare l'applicazione da installare; in un progetto, infatti, possono essere presenti più applicazioni. Prima di continuare si potrà scegliere se pubblicare in un server o in un launcher. In questo caso occorre selezionare l'opzione *Server*.

Nel popup successivo sarà possibile dare un nome alla nuova build che verrà compilata al volo in base allo stato attuale dell'applicazione nel progetto, oppure selezionare una build esistente.

Nel caso di nuova build, il popup presenta anche le seguenti opzioni:

- *Compila la build senza installarla*: la build verrà solo compilata ma non verrà installata.
- *Abilita il debug a runtime*: la build verrà compilata aggiungendo il supporto al debug a runtime dell'applicazione nel server.
- *Salta il controllo delle traduzioni*: in caso di build di test, non verrà effettuato il controllo della traduzione completa dell'applicazione nelle lingue dichiarate. Questa opzione è disabilitata se il server non è stato marcato come server di test.

Infine nell'ultimo popup sarà possibile dare un nome all'applicazione installata e selezionare uno o più server in cui installarla. Cliccando infine il pulsante *Installa* nel terzo popup verrà iniziato il processo di installazione vero e proprio.

Aggiornamento di un'installazione

Se si desidera allineare un'installazione esistente allo stato attuale dell'applicazione nel progetto, è possibile cliccare il pulsante *refresh* evidenziato nell'immagine precedente che appare quando si visualizzano i dettagli di un'installazione della lista. In questo modo apparirà solo un popup di conferma in cui è possibile solamente cambiare il nome della build e modificare le opzioni di compilazione viste in precedenza.

Si noti, tuttavia, che in questo caso è presente un'ulteriore opzione attiva per default: *Sovrascrivi build esistente*. L'aggiornamento dell'installazione, infatti, per default sovrascrive la build precedente, che in questo modo viene persa. Se si desidera mantenere anche la build precedente, è possibile deselezionare l'opzione.

Processo di installazione

Il processo di installazione inizia con la compilazione dell'applicazione e la creazione di una nuova build. La nuova build viene poi caricata in un bucket di Google Cloud così da poterla mantenere come backup e passarla al server per l'installazione.

Quando la build è stata creata senza errori e caricata nel bucket, inizia l'operazione di installazione vera e propria che è fatta dai seguenti passaggi:

1) L'applicazione corrente viene messa in modalità manutenzione, cioè non si accettano più nuove sessioni e in quelle attuali viene visualizzato un messaggio che comunica all'utente di concludere il lavoro prima possibile.
2) Nel frattempo viene scaricata dal bucket la build da installare.
3) Dopo un tempo pari a 15 secondi le sessioni attuali vengono interrotte e l'applicazione viene messa in pausa.
4) Prima di iniziare le operazioni di modifica del server, viene effettuato uno snapshot dell'intero server in modo da poter ripristinare la situazione precedente all'installazione stessa.
5) Il sistema di installazione estrae i file della nuova build in una cartella temporanea. La build viene validata come applicazione Node.js.
6) Se richiesto, vengono aggiornati gli schemi dei database utilizzati dall'applicazione.

7) In caso di aggiornamento, il file system dell'applicazione attuale viene spostato nella nuova directory. La vecchia copia viene rinominata e alla nuova viene dato il nome giusto.
8) Se tutto ha funzionato come previsto, le modifiche agli schemi dei database vengono confermate e tutti i file temporanei vengono cancellati.
9) Se ci sono stati problemi in anche solo uno dei passaggi precedenti, tutte le modifiche al file system e ai database del server vengono annullate, ovvero tutto torna come prima.
10) L'applicazione viene riavviata e gli utenti ancora collegati possono rientrare automaticamente.

A parte i tempi di compilazione e di gestione delle sessioni esistenti, tutto il processo di installazione richiede solamente qualche secondo o al massimo qualche decina di secondi. In questo modo gli utenti dell'applicazione non avranno la percezione di un servizio interrotto, ma solo di una pausa momentanea.

Aggiornamento del database

Quando una build viene compilata, essa contiene anche le informazioni per la creazione e l'aggiornamento degli schemi dei database Postgres da essa utilizzati. Nel momento in cui la build è in fase di installazione, la console verifica se i database presenti nel server devono essere aggiornati o meno. Nel caso in cui sia richiesto un aggiornamento, nel popup di selezione del server sarà presente anche l'opzione *Aggiorna database*, come mostrato nell'immagine seguente:

L'opzione può apparire pre-selezionata se su quel server il database è effettivamente da aggiornare e se l'applicazione che ha aggiornato il database l'ultima volta è la stessa. Un database, infatti, può essere utilizzato da più di un'applicazione, ma normalmente solo una di esse ne mantiene aggiornato lo schema.

Se si seleziona l'opzione, l'aggiornamento dello schema dei database è parte del processo di installazione. Per ogni database le istruzioni di modifica vengono eseguite tutte in una stessa transazione in modo da poter essere completamente annullate se anche solo una di esse non ha successo.

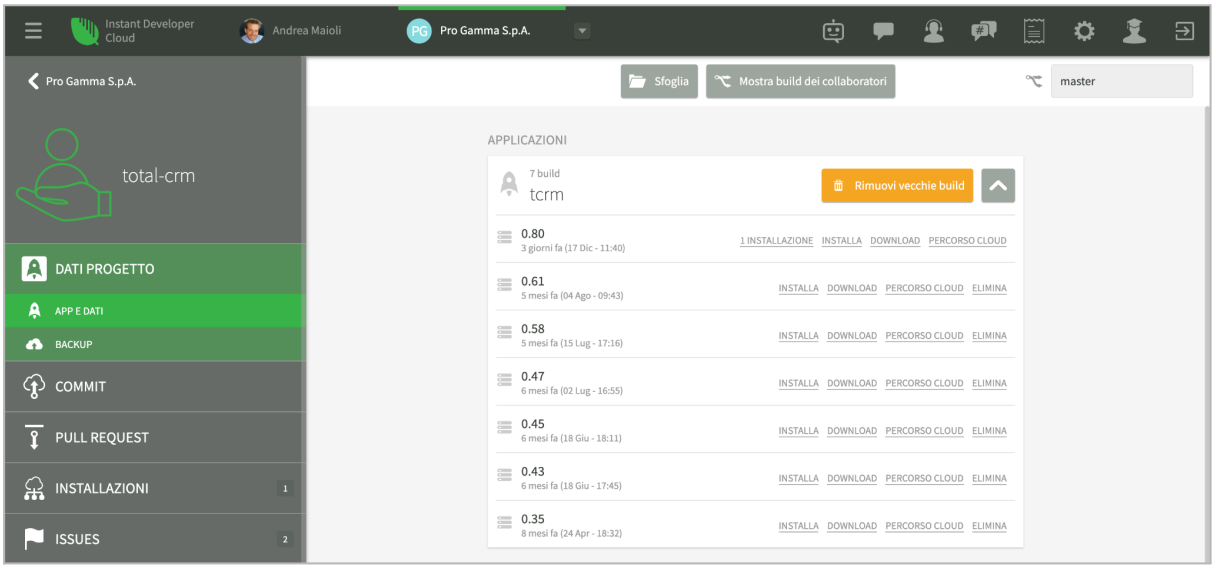
Se l'aggiornamento del database del server non ha successo, si consiglia di effettuare un backup e un restore del medesimo nell'ambiente IDE e poi testare l'aggiornamento in tale ambiente.

Gestione delle build

L'elenco delle build di una determinata applicazione è disponibile dal sotto-menu di progetto *App e dati,* come si vede nell'immagine seguente.

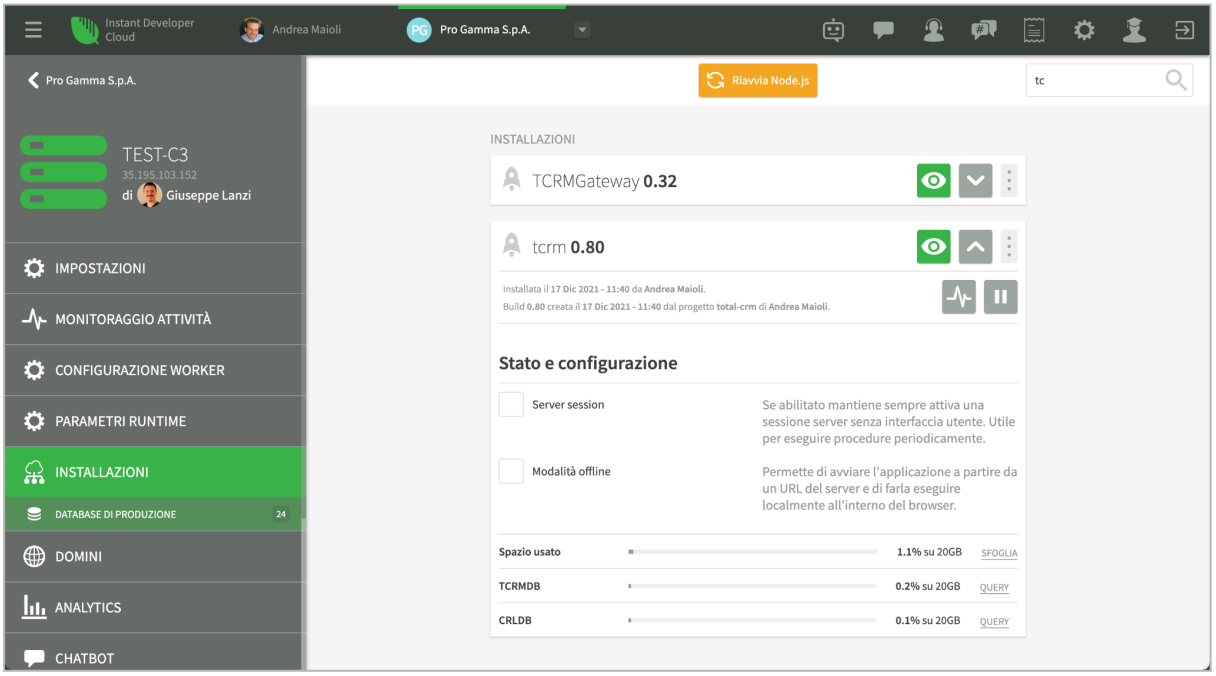
Per ogni build viene riportato su quali server è installata. Inoltre è possibile installarla su altri server, eliminarla oppure scaricarla per l'installazione su un proprio server in modalità *self managed*.

Il pulsante *Rimuovi vecchie build* è in grado di cancellare automaticamente tutte le build più vecchie di 30 giorni che non sono installate da nessuna parte. Si consiglia di utilizzarlo di tanto in tanto per eliminare lo spazio occupato dalle build, sia nel server IDE che nei bucket di backup della propria organizzazione.



Configurazione dell'applicazione

Dopo aver installato un'applicazione, essa diventa disponibile per la configurazione dal menu *Installazioni* nella sezione relativa al server in cui è stata installata.



Il menu relativo alla singola installazione presenta le seguenti voci:

- *Parametri runtime*: apre la videata per la configurazione dei parametri di runtime dell'applicazione, descritta in seguito.
- *Configurazione worker*: apre la videata per la configurazione dei processi worker dell'applicazione, descritta in seguito.
- *Naviga tra i file*: apre la videata per la gestione del file system dell'applicazione. È equivalente al pulsante *Sfoglia* nella riga *Spazio usato* sottostante.
- *Condividi*: permette di condividere un'installazione disponibile su un launcher con altri utenti.
- *Installa su un launcher*: rende l'applicazione disponibile all'interno di un launcher anche se essa è di tipo online, cioè installata sul server.
- *Allinea database*: forza l'allineamento dei database del server usati dalla build. Attenzione: prima di usare questo comando si consiglia di eseguire uno snapshot del server per poter annullare il comando in caso di bisogno.
- *Disinstalla*: rimuove l'applicazione installata.

Espandendo la riga relativa ad un'installazione è possibile accedere a diverse opzioni di configurazione.

Tramite il pulsante con l'icona ad impulso è possibile accedere alla pagina del *log strutturato* che verrà descritta in seguito. Il pulsante con l'icona pausa, infine, mette in pausa l'applicazione e impedisce l'accesso da parte di browser, client API, o dispositivi mobile. Le sessioni esistenti non vengono interrotte.

Attivando il flag *Server session*, viene lanciata una ulteriore sessione dell'applicazione senza interfaccia utente. Questa sessione può eseguire processi batch, rimanere in ascolto di eventi o di modifiche al database, o altri task di backend. Attivando il flag viene garantito che ci sarà sempre una sessione batch in esecuzione. Il codice dell'applicazione può distinguere se l'applicazione è in esecuzione in una sessione browser o in una sessione server (batch) utilizzando la funzione *app.isServerSession()* che restituisce *true* in questo caso.

Il flag *Modalità offline* permette di eseguire l'applicazione come PWA. Per una trattazione più estesa di questo argomento, è possibile leggere il relativo libro nella documentazione.

Si noti infine il pulsante *Riavvia Node.js* in alto nella lista delle installazioni, che permette di riavviare l'intero sistema applicativo nel server selezionato. Tutte le sessioni di tutte le applicazioni verranno interrotte.

Parametri di runtime

La videata dei parametri di runtime permette di definire delle coppie chiave-valore che l'applicazione è in grado di leggere o scrivere tramite i metodi *app.getParameter* e *app.setParameter*.

Il vantaggio di poter inserire tramite la console dei parametri che l'applicazione può leggere consiste nel poter comunicare chiavi segrete senza doverle inserire nel codice del progetto, o dati che possono variare senza dover aggiornare l'applicazione.

I parametri di runtime sono relativi alla singola installazione, tuttavia è possibile definire parametri di runtime anche a livello di server, tramite il menu *Parametri runtime* della pagina server. Se un parametro viene definito a livello di applicazione esso prevale su quelli definiti a livello di server, altrimenti viene usato quello di server come default per tutte le installazioni.

Quando l'applicazione scrive il valore di un parametro, esso sarà sempre definito a livello di applicazione. I parametri di runtime definiti a livello di server, quindi, sono scrivibili solamente tramite la console.

Si noti che a livello di applicazione è possibile gestire l'evento *app.onParameterChange* che comunica la variazione dei parametri anche alle sessioni già avviate.

Esistono alcuni parametri che vengono gestiti direttamente dal framework:

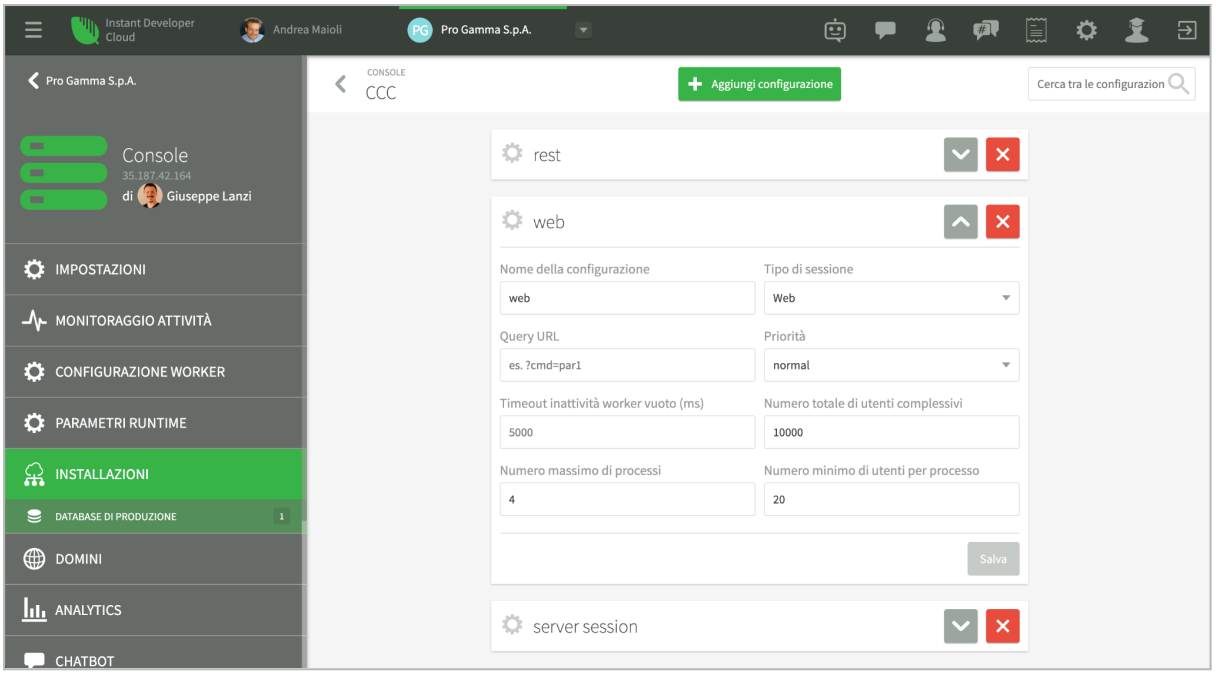
- *trackCodeLine*: valore booleano (*true*/*false*), con default *false*. Abilita un tracciamento più preciso delle righe di codice che generano eccezione; viene utilizzato in Analytics per aprire il progetto direttamente al punto che ha generato il problema; in alternativa viene riportato solo il numero di riga del file JavaScript. Impostandolo a *true* ci sarà una piccola penalizzazione nelle performance di esecuzione del codice applicazione.
- *allowOffline*: valore booleano (*true*/*false*), con default *false*. Deve essere impostato a *true* per consentire ad un'applicazione di essere eseguita anche in modalità PWA.
- *sendGridKey*: valore della chiave da utilizzare per inviare mail tramite l'oggetto *App.Mailer* del framework di Instant Developer Cloud. Esso utilizza a sua volta il servizio SendGrid a cui ci si deve registrare per ottenere le chiavi di invio mail.
- *stripeKey*: valore della chiave da utilizzare per la libreria per la gestione dei pagamenti elettronici *App.Stripe* del framework di Instant Developer Cloud.
- *gcmKey*: valore del parametro *gcmKey* per l'invio delle notifiche a dispositivi Android.
- *apnKey*: contenuto del file del certificato per l'invio delle notifica a dispositivi Apple.
- *apnKeyId*, *apnTeamId*: valori dei parametri per l'invio delle notifica a dispositivi Apple.
- *gmapKey*: valore della chiave per l'utilizzo del componente Google Maps. Non deve cominciare per "key=" e deve contenere solo il valore della chiave così come configurato nella console di Google Cloud.

Configurazione dei processi worker

Ogni applicazione installata su un server utilizza uno o più processi worker per gestire le sessioni. La pagina di configurazione dei worker serve per definire le politiche di gestione dei processi dell'applicazione. La configurazione standard è definita a livello di server ed è accessibile dal menu *Configurazione worker* della pagina del server.

La pagina *Configurazione worker* di un'installazione è accessibile dal menu corrispondente nella lista delle installazioni dell'applicazione da configurare.

Per aggiungere una nuova configurazione, cliccare il pulsante *+Aggiungi configurazione* nella testata della videata.



Una configurazione è definita dai seguenti parametri:

- *Nome*: identifica la configurazione.
- *Tipo di sessione*: indica per quali tipi di sessione è valida questa configurazione. I possibili valori sono: *web, sync, rest, server, all*.
- *Query string*: indica se questa configurazione entra in gioco solamente se sono presenti alcuni parametri sulla *query string* del browser che attiva la sessione.
- *Priorità*: priorità di esecuzione del processo worker. Anche se è possibile indicare qualunque valore ammesso (*max, high, normal, low, min*) si consiglia di utilizzare solo *normal* e *low*. Un processo con priorità *min* funzionerà solo quando tutti gli altri non sono impegnati; un processo con priorità *high* può interferire con il comportamento del server; un processo con priorità *max* può addirittura bloccare il server se esso non si comporta correttamente.
- *Timeout inattività*: indica dopo quanto tempo un worker che non ospita più alcuna sessione deve essere eliminato.
- *Numero totale utenti complessivi*: è il numero massimo di sessioni contemporanee ammesse per il tipo selezionato. Se, ad esempio, viene impostato a 100 ed il tipo è *rest*, significa che il server può gestire al massimo 100 richieste API contemporanee. Alle ulteriori richieste verrà restituito un codice di errore.
- *Numero massimo di processi*: è il numero massimo di processi worker che verranno generati per gestire le sessioni. Il sistema bilancia il numero di sessioni in modo che tutti i processi ne contengano un numero simile.
- *Numero minimo di utenti per processo*: è il numero minimo di sessioni che un processo worker deve contenere prima di aggiungere un nuovo processo worker. Se, ad esempio, lo si imposta a 20 ed il tipo è web, le prime 20 sessioni browser contemporanee verranno gestite da un solo processo worker. Se viene attivata un'ulteriore sessione, essa verrà gestita da un nuovo worker, se non si è ancora arrivati al numero massimo di processi worker.

Si consiglia di aggiungere una sola configurazione per tipo di sessione / query string.

Per decidere quanti processi worker dedicare ad ogni tipo di sessione, occorre tenere presente i seguenti parametri:

- Definire una politica per un determinato tipo di sessione determina la generazione di almeno un processo worker dedicato a quel tipo di sessione. Se, ad esempio, si aggiunge una configurazione per le sessioni *sync*, tutte le sessioni di sincronizzazione verranno gestite con processi diversi da quelli delle sessioni *web*.
- Nel caso di applicazioni che necessitano di una *server session*, mantenerla su un worker separato può avere il vantaggio di tenere i processi batch indipendenti dal carico di lavoro delle funzioni interattive.
- A parte il caso di applicazioni con calcoli intensivi, il collo di bottiglia delle applicazioni è sempre l'accesso al database. Non è quindi utile aumentare il numero di processi worker a più del numero di processori del server. La miglior strategia per gestire la scalabilità è quella di ottimizzare il numero e la qualità delle query.
- Ogni worker può indirizzare al massimo 1,7 GB di memoria. All'aumentare del numero di sessioni contemporanee può essere utile suddividere il carico su più worker anche per poter utilizzare più memoria.
- Occorre tenere presente la natura di Node.js come processo *single threaded* dotato di *event loop*. Se un thread di codice JavaScript non utilizza funzioni asincrone ma, ad esempio, esegue calcoli intensivi di durata maggiore ai 100 ms, il processo worker che li esegue potrebbe non riuscire a gestire tutte le altre sessioni in maniera puntuale. In questi casi si consiglia di spezzare i calcoli complessi introducendo funzioni asincrone, oppure di isolarli in processi worker dedicati.

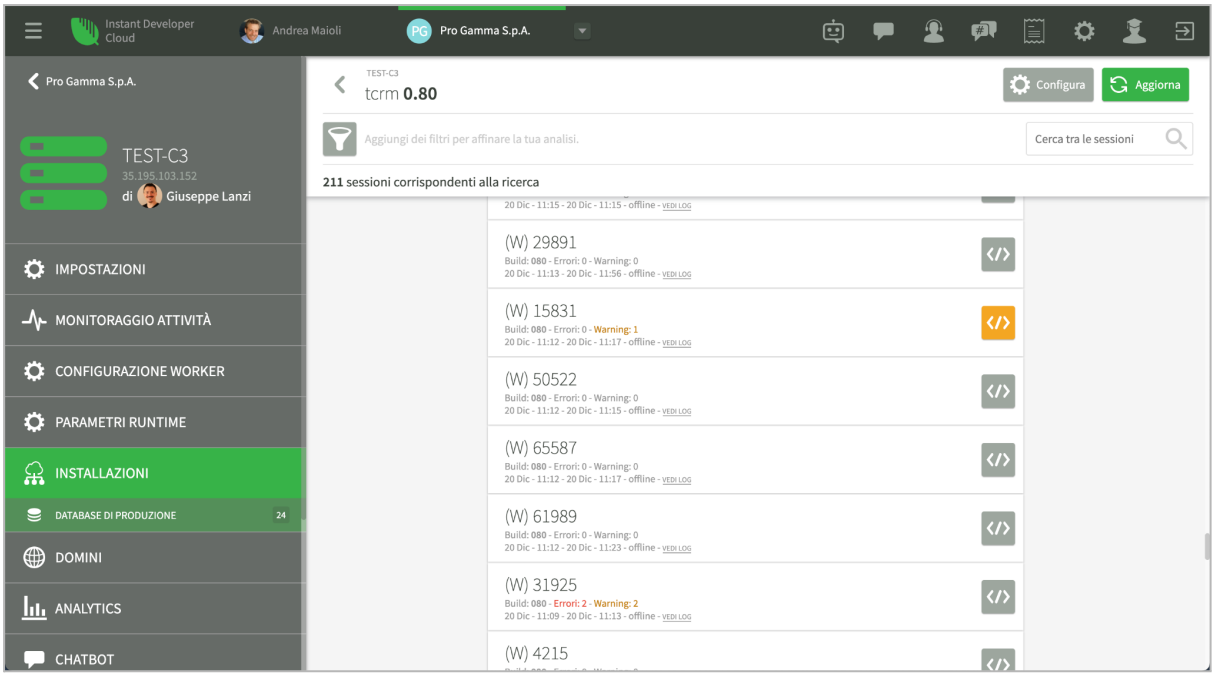
Log strutturato

Il log strutturato è un potente sistema di debug e controllo delle applicazioni in produzione. Attivando il log strutturato per un'installazione, tutte le sessioni applicative verranno loggate in maniera separata e sarà possibile identificare errori, warning o log di console specifici.

A richiesta ogni sessione può eseguire il log di tutte le query effettuate sul database e, se l'applicazione è stata compilata con il flag di debug a runtime attivo, per ogni sessione online sarà possibile attivare la modalità di debug interattivo.

Per utilizzare il log strutturato, è necessario far sì che il codice dell'applicazione imposti la proprietà *app.sessionName* per ogni sessione di cui interessa effettuare il log. Il nome della sessione può contenere qualunque stringa. Nella pratica è comodo inserire il tipo di sessione, ad esempio (W) per le sessioni web, (S) per quelle sync e (R) per le sessioni rest. Nel rispetto delle regole sulla privacy è anche possibile indicare il nome dell'utente collegato alla sessione, in modo da avere immediatamente un legame fra gli eventuali feedback riportati e l'analisi "dietro le quinte" di quanto è successo. Se una sessione non imposta *app.sessionName*, non verrà loggata nel log strutturato.

L'attivazione del log strutturato avviene dalla pagina che si apre cliccando il pulsante con l'icona ad impulso presente nella sezione relativa ad un'installazione. Cliccando il pulsante *Configura* nella testata della videata, sarà possibile attivare il log strutturato, attivare anche i log del database, cancellare le sessioni di log salvate ed infine selezionare un filtro per salvare solo le sessioni che hanno un determinato nome o parte di esso.



Cliccando il pulsante *Aggiorna,* sempre nella parte alta della videata, è possibile reperire immediatamente le ultime informazioni sulle sessioni, che, altrimenti, vengono estratte solo ogni 15-30 minuti.

Per ogni sessione in cui è presente un log, esso può essere visualizzato cliccando sulla riga della sessione stessa. Il pulsante sulla destra, invece, apre il progetto. Se l'applicazione è stata compilata con il debug a runtime attivo, il progetto sarà proprio quello che ha dato origine alla build, che in questo caso viene mantenuto in un backup apposito. Se la sessione è online, sarà possibile interagire con la sessione utilizzando il debugger dell'IDE.

Per ogni sessione viene loggata ogni chiamata alle funzioni *console.log*, *console.warn* e *console.error*, sia da parte del codice dell'applicazione che a livello di framework. I warning e gli errori vengono anche comunicati alla console, così da poterne visualizzare il numero nella lista prima ancora di aprire il log della sessione. Si tenga presente, infatti, che la console conosce i dati della sessione, ma il log rimane presente nel file system dell'applicazione sul server di produzione.

Tramite il pulsante con l'icona filtro in alto nella pagina ed il relativo campo di ricerca dal lato opposto sarà possibile filtrare la lista per visualizzare subito le sessioni che hanno generato errori o warning, oppure modificare il periodo di visualizzazione che, per default, riguarda le ultime 24 ore.

Gestione dello spazio relativo al log strutturato

Il sistema di log strutturato raccoglie l'output di tutti i comandi di logging che sono stati inseriti nell'applicazione e, se configurato, i testi di tutte le query eseguite. Lo spazio richiesto per memorizzare queste informazioni può crescere rapidamente se ci sono molte sessioni oppure se sono stati lasciati attivi log di oggetti molto corposi.

Dalla versione 22, il sistema possiede una protezione che disabilita il log strutturato se nel server rimangono meno di 500 MB di spazio libero. Si consiglia tuttavia di mantenere il log delle query normalmente disattivato e di evitare il log di oggetti complessi per non sovraccaricare inutilmente questo sistema.

Si segnala infine che i log strutturati delle sessioni vengono cancellati definitivamente dopo trenta giorni per motivi di privacy.

Gestione dei database di produzione

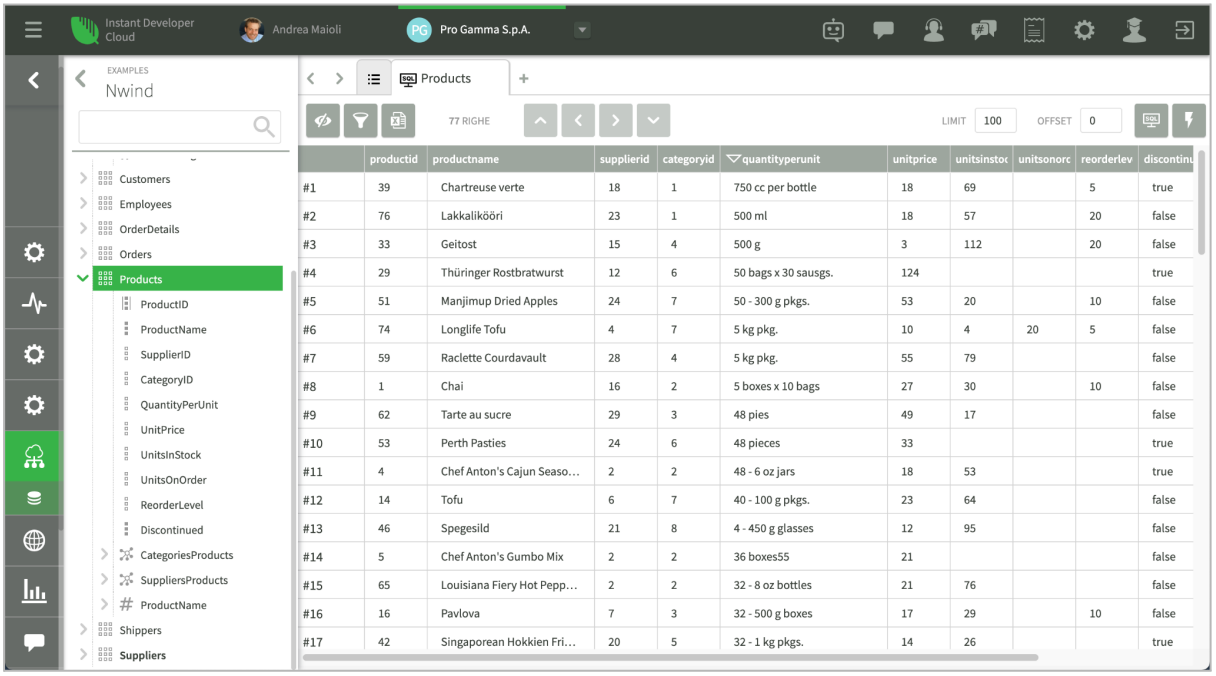
I database di produzione sono collocati nello stesso server delle applicazioni che li devono usare. È possibile accedere alle pagine di gestione dei database del server in due modi:

- Tramite il sottomenu *App e dati* del menu *Dati di progetto*, oppure
- Tramite il sottomenu *Database di produzione* del menu *Installazioni* del server.

La pagina di gestione mostra la lista di ogni database del server. Per ognuno di essi è possibile eseguire le seguenti operazioni tramite il menu di lista:

- Backup
- Restore
- Download di un backup
- Eliminazione

È possibile inoltre accedere ad un query editor, come mostrato nell'immagine seguente.



Le operazioni di backup e restore possono essere utili per spostare un database di produzione nell'ambiente di sviluppo o viceversa, ma anche per preparare i dati per i test di non regressione o di carico.

Il sistema di backup periodico dei dati dei database viene invece implementato a livello dell'intero server, come illustrato nei paragrafi seguenti.

Configurazione del server

La configurazione del server avviene attraverso le pagine a cui si accede dal menu del server. Vediamole una per una.

Impostazioni

La pagina delle impostazioni contiene le impostazioni generali del server, ed è divisa in varie card. Nella prima è possibile tenere sotto controllo i parametri più importanti (%CPU, RAM, storage e traffico), oltre a poter effettuare l'upgrade del server e la condivisione nel gruppo di lavoro.

Nella seconda card è possibile modificare il nome del server, così come è conosciuto nella console, mentre il codice univoco viene assegnato dalla console al momento della creazione e non è modificabile.

È inoltre possibile sapere quale versione della piattaforma è installata nel server ed eventualmente aggiornarla. Nei paragrafi seguenti illustreremo la procedura migliore per aggiornare la piattaforma dei propri server.

Nella zona seguente si possono attivare o meno alcuni servizi, fra i quali il test automatico, che permette di utilizzare il server come target di esecuzione di test automatici, e il servizio feedback e issue, che, appunto, permette di raccogliere feedback dagli utenti e gestire le issue sui server di sviluppo.

La gestione delle issue è attiva per default nei server di sviluppo, mentre quella dei feedback deve essere attivata in quanto il servizio di raccolta feedback è un servizio aggiuntivo contenuto nel pacchetto analytics.

Nell'ultima zona infine è possibile configurare un'applicazione come default per il server. In questo caso sarà sufficiente digitare il nome completo del server per entrare nell'applicazione.

È infine possibile aggiungere pacchetti *Node.js* personalizzati, inserendone il nome e la versione nel campo *Pacchetti npm personalizzati*. I pacchetti devono essere separati da virgola. Ad esempio, questa è una stringa valida per inserire il pacchetto *ftps* e *image-size*: *ftps@1.1.1,image-size@1.0.0*.

Dopo ogni modifica occorre cliccare il pulsante verde *Salva* in alto a destra per confermare l'operazione.

Monitoraggio delle attività

La pagina di monitoraggio delle attività permette di conoscere l'allocazione delle risorse del server in tempo reale. Oltre ai grafici relativi alla CPU, alla RAM e al numero di sessioni, è possibile conoscere lo stato di tutti i processi in esecuzione. Per ogni processo viene riportata la RAM allocata totale ed in percentuale, l'affinità con le CPU del server, la percentuale di CPU attualmente utilizzata e il codice di stato del processo.

I processi sono classificati nelle seguenti categorie:

1) Processi di sistema (System).
2) Processi del framework di Instant Developer Cloud (Framework).
3) Processi per le connessioni ai database (Connections).
4) Processi per la gestione del database (DB).

Infine ogni worker di ogni applicazione verrà riportato in un gruppo separato indicando il tipo di worker, il numero delle sessioni applicative nel worker e i restanti dati del processo.

Tutti i dati relativi ai processi vengono raccolti analizzando il risultato del comando *top* eseguito dal sistema operativo del server.

Se il periodo scelto nel filtro delle date contiene il momento attuale, il grafico delle risorse si aggiornerà ogni 30 secondi, mentre la lista dei processi si aggiornerà ogni 10 secondi. I dati vengono mantenuti solo per gli ultimi 60 giorni.

La pagina si chiude mostrando il grafico dell'utilizzo dello storage, cioè del disco dati e del traffico di rete.

Questa pagina fornisce informazioni molto importanti per comprendere se:

- La capacità del server è adeguata al carico di lavoro che varia nel tempo.
- Esistono processi bloccati o che assorbono una quantità eccessiva di risorse.
- Quali tipi di attività sono in corso.

Configurazione dei worker

Questa pagina permette di configurare i valori di default per la configurazione dei worker delle applicazioni installate.

In particolare, specificando il *Numero massimo di worker per app* è possibile definire il numero di processi worker che verranno creati dal server per gestire le sessioni di ogni applicazione. Il *Numero minimo di utenti per worker* indica qual è il numero di sessioni che ogni worker gestirà prima che venga avviato un nuovo processo che prenda in carico nuove sessioni client. Il *Numero massimo di utenti per app* indica il numero massimo di sessioni gestite contemporaneamente da un'applicazione prima che questa rifiuti l'avvio di una nuova sessione. In questo caso il valore tiene conto della somma di tutte le sessioni dei worker della stessa app.

Parametri di runtime

Questa pagina permette di aggiungere parametri di runtime che rappresentano valori di default per tutte le applicazioni installate nel server.

Installazioni

La pagina delle installazioni contiene la lista delle applicazioni installate e permette di gestirle. Si ricorda che la stessa build può essere installata più volte nello stesso server, per dare luogo a più installazioni a partire dalla medesima base di codice.

Domini

La pagina dei domini permette di configurare i nomi di dominio internet che devono essere associati al server. Per ogni nome di dominio configurato è possibile specificare l'eventuale certificato SSL da utilizzare e un'applicazione predefinita da avviare automaticamente.

Esistono due tipi di domini: gli alias e i domini personalizzati.

Alias

Gli alias sono i domini più semplici da configurare; per crearne uno è sufficiente cliccare il pulsante *+Alias*. Apparirà una nuova riga nella lista dei domini, all'interno della quale è possibile specificare un nome e l'app predefinita da associare al dominio.

Il valore inserito nel campo del nome verrà usato per calcolare un dominio con il pattern *[nome]-[codice della propria organizzazione].instantdevelopercloud.com*. Ad esempio, usando come nome *myapp* su un server dell'organizzazione *myorg*, il dominio personalizzato diventa *myapp-myorg.instantdevelopercloud.com*. Lo scopo è quello di fornire un modo semplice per personalizzare l'accesso alle proprie applicazioni da parte degli utenti finali, senza dover necessariamente acquistare un proprio dominio.

Il pulsante *Salva* permette di salvare i dati nel database della console, mentre il pulsante *Carica sul server* configura effettivamente il server di riferimento per l'uso del certificato.

Per quanto riguarda i certificati HTTPS, Il dominio di tipo *Alias* viene sempre gestito automaticamente da Instant Developer Cloud, attraverso il sistema *let's encrypt*.

Il pulsante *Ripristina l'ultima configurazione caricata* permette di riallineare i dati della console di uno specifico Alias all'ultima configurazione effettivamente caricata sul server.

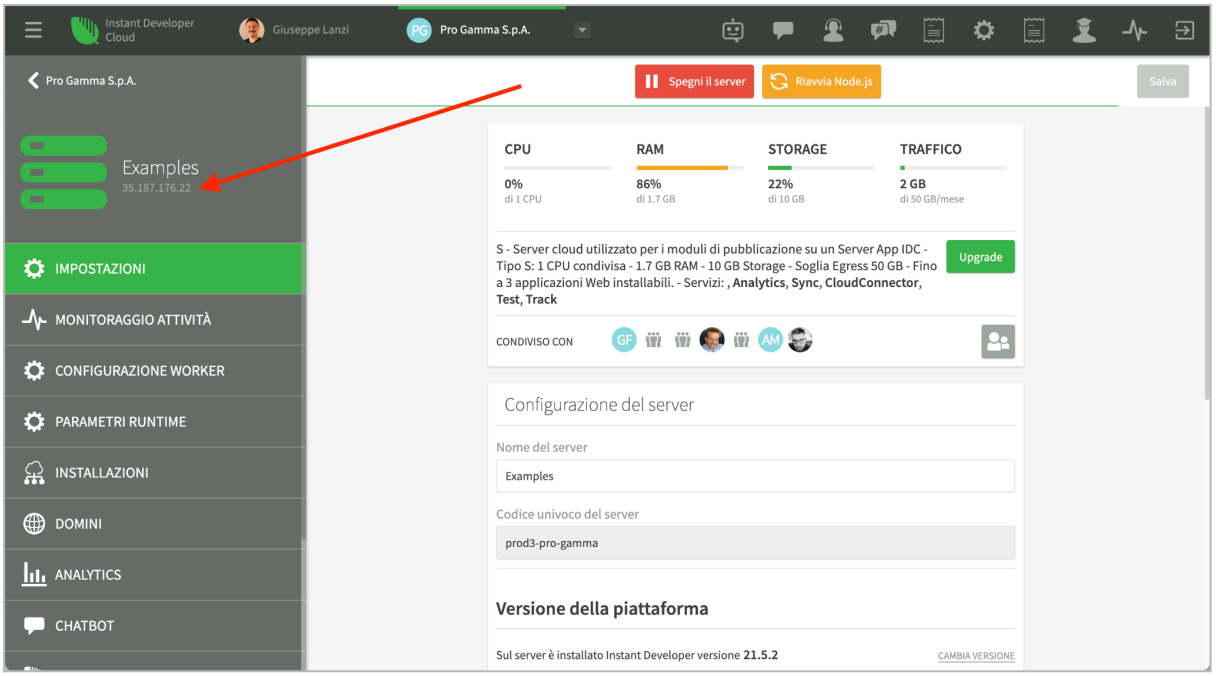
Siccome i certificati sono caricati all'avvio di Node.js, dopo aver modificato e salvato i dati di questa pagina occorre riavviarlo per rendere effettiva la modifica. A tal fine, è possibile utilizzare il comando *Riavvia Node.js* disponibile nella pagina *Impostazioni*.

Dominio personalizzato

I domini personalizzati permettono di utilizzare un qualsiasi dominio che sia puntato sull'IP del server. Questo dato è disponibile nel box in alto a sinistra nella pagina di gestione del server, sotto al nome, come mostrato nell'immagine alla pagina seguente.

Per creare un dominio personalizzato è sufficiente cliccare il pulsante *+Dominio*. Apparirà una nuova riga nella lista dei domini, all'interno della quale è possibile specificare un nome, i dati relativi al certificato SSL da utilizzare e l'app predefinita da associare al dominio.

Per configurare il certificato SSL ci sono due possibilità: usare il proprio certificato SSL oppure farne creare uno al sistema tramite Let's Encrypt.



Per poter utilizzare il proprio certificato SSL è necessario caricare sul server tre file in formato PEM:

- *Certificato SSL*: è un file di testo che contiene la catena del certificato.
- *Chiave del certificato*: è un file di testo che contiene la chiave privata relativa al certificato.
- *Bundle*: è un file di testo che contiene gli eventuali certificati delle Autorità di Certificazione (AC) da utilizzare al posto della lista predefinita curata da Mozilla. Il file può essere vuoto, ma se il certificato è *autofirmato* è obbligatorio indicare la propria autorità di certificazione.

Per maggiori informazioni sul formato dei certificati è possibile far riferimento alla documentazione di node.js.

È possibile chiedere ad Instant Developer Cloud di generare i file del certificato in automatico, tramite il sistema Let's Encrypt. A tal fine è sufficiente abilitare la casella di controllo *Usa un certificato automatico*.

Il pulsante *Salva* permette di salvare i dati nel database della console, mentre il pulsante *Carica sul server* configura effettivamente il server di riferimento per l'uso del certificato.

Il pulsante *Ripristina l'ultima configurazione caricata* permette di riallineare i dati della console di uno specifico Dominio all'ultima configurazione effettivamente caricata sul server.

Siccome i certificati sono caricati all'avvio di Node.js, dopo aver modificato e salvato i dati di questa pagina occorre riavviarlo per rendere effettiva la modifica. A tal fine, è possibile utilizzare il comando *Riavvia Node.js* disponibile nella pagina *Impostazioni*.

Let’s Encrypt con proprio nome di dominio

Non appena si carica sul server un dominio basato su certificato Let’s Encrypt, il server effettua le operazioni di verifica e di creazione del dominio. Qualora si manifestassero errori nella registrazione del dominio, su Let’s Encrypt comparirà un messaggio di errore associato al dominio che lo ha generato.

La causa principale di errori di creazione del certificato risiede nella configurazione dei puntamenti del dominio al server. Infatti, il puntamento del dominio al server deve essere effettuato tramite IPv4, inoltre il dominio non deve avere un puntamento IPv6.

Molti registrar creano i domini con due puntamenti di default: IPv4 e IPv6. Modificando solo quello IPv4 per puntare all’IP del server e non cancellando il puntamento IPv6 si otterrà un errore nella fase di verifica effettuata da Let’s Encrypt.

Analytics

Questa pagina permette di installare il servizio Analytics sul server di produzione e di configurarlo. Per poter installare Analytics è richiesto l’acquisto del relativo pacchetto.

Una volta installato il servizio Analytics, il server potrà essere utilizzato come punto di raccolta delle informazioni provenienti dalle applicazioni.

Backup

La pagina dei backup contiene la lista degli snapshot del server effettuati dalla console tramite i servizi di Google Cloud Platform.

I backup del server vengono effettuati nelle seguenti condizioni:

- Subito prima dell’installazione o della disinstallazione di un’applicazione.
- Se si clicca il pulsante *Backup* in alto nella pagina.
- Se è attivo il servizio *Backup giornaliero* o *Backup orario* per il server.
- Subito prima di effettuare un ripristino del server.
-

Verranno mantenuti i backup degli ultimi 30 giorni indipendentemente dal motivo per cui sono stati creati.

Siccome il backup è uno snapshot coerente del disco dati dell’intero server, esso contiene tutti i database, tutto il file system e tutto il codice di tutte le applicazioni in esso installate.

Nel caso in cui un server risulti irraggiungibile o compromesso, la procedura di ripristino permette di ricreare completamente il server usando come disco dati il backup desiderato. La procedura di ripristino richiede qualche minuto di tempo e riporta lo stato dell’intero server al momento in cui il backup è stato effettuato. Il ripristino del disco di sistema è invece sempre totale. Il disco di sistema viene ricostruito a partire dalla versione originale relativa alla versione di piattaforma installata nel server.

L’uso degli snapshot coerenti è un ottimo sistema di disaster recovery in quanto permette di ripristinare lo stato del server ad un qualunque momento in cui è avvenuto il backup con una semplice operazione di console, richiedendo solo qualche minuto.

Log

La pagina dei log, infine, permette di accedere ai log dell’intero server. In particolare ai dati di *console.log* e *console.error* non catturati dai log strutturati e al log del server vero e proprio.

Si consiglia di utilizzare il pulsante di download per ottenere il contenuto del log come file di testo. Il contenuto dei file di log è spesso utile per ottenere informazioni sulle operazioni del server e su eventuali anomalie di funzionamento.

Aggiornamento della piattaforma

La piattaforma Instant Developer Cloud è in continua evoluzione: ogni anno sono previste due release maggiori, la prima a fine gennaio e la seconda a fine luglio. Fra queste due release maggiori possono essere rilasciate delle release minori, a seconda della necessità di correggere malfunzionamenti o regressioni.

L’aggiornamento dei propri server di sviluppo e di produzione alle release maggiori deve essere accuratamente pianificato.

L’aggiornamento alla release minori, invece, può essere tranquillamente demandato alla console attivando per il server il flag *Installa gli aggiornamenti automaticamente*, contenuto nella pagina delle impostazioni del server. In questo caso i server di sviluppo verranno automaticamente aggiornati. I server di produzione verranno invece aggiornati automaticamente al primo riavvio, oppure è possibile eseguire l’operazione manualmente.

L’aggiornamento di un server ad una nuova release avviene dalla pagina delle impostazioni del server, cliccando il pulsante *Cambia versione*. In questa fase verranno presentate anche le note di rilascio della nuova versione: si consiglia di leggerle accuratamente prima di continuare.

Aggiornamento alla release maggiore successiva

Il passaggio ad una release maggiore successiva dei propri server di sviluppo e di produzione è un’operazione normalmente non invertibile, fatta eccezione per i server di produzione, per i quali è possibile ritornare alla versione precedente ripristinando un backup del server eseguito tramite la console subito prima del passaggio.

La ragione consiste nel fatto che le nuove versioni possono contenere versioni successive dell’intero sistema operativo, a tutti i livelli. La versione 22.1, ad esempio, ha aggiornato il database server a Postgres 13.3 e questa operazione non è invertibile. Il codice stesso del framework di Instant Developer Cloud si adatta alle nuove versioni dei sistemi operativi e delle centinaia di pacchetti base che ogni volta vengono aggiornati per garantire la massima sicurezza.

Inoltre le migliaia di punti di modifica che ogni versione maggiore contiene, non permettono di garantire a priori al 100% che non ci saranno *breaking change* nei propri progetti. In questo caso la protezione dalle *breaking change* avviene a posteriori: ogni regressione segnalata al servizio di assistenza viene prontamente corretta, ripristinando esattamente il comportamento precedente o quello più vicino ad esso tecnicamente possibile.

Per queste ragioni è necessario pianificare la fase di passaggio alla release maggiore successiva come segue:

1) Evitare i periodi critici in cui sarà necessario effettuare rilasci delle proprie applicazioni. Si consiglia di mantenere una finestra temporale di due settimane.
2) Effettuare l'aggiornamento del solo server di sviluppo e del server dedicato ai test, se disponibile.
3) Testare prontamente i propri progetti per verificare che essi compilino ancora, possano essere installati su un server di test e che i test di non regressione abbiano successo.
4) Segnalare prontamente al servizio di assistenza eventuali regressioni. Le segnalazioni di regressione vengono sempre considerate urgenti, sebbene si tenga conto anche degli effetti della regressione per valutare la priorità.
5) Solo quando tutti i test avranno avuto successo sarà possibile aggiornare i server di produzione alla nuova release e di conseguenza aggiornare le applicazioni, se necessario.

Nel caso in cui non sia possibile pianificare una finestra temporale di almeno due settimane in cui eseguire i test di passaggio, si consiglia di operare come segue:

1) Nella console di Instant Developer Cloud, creare una seconda organizzazione dedicata ai test.
2) Acquistare una licenza di sviluppo per una sola mensilità.
3) Aggiornare il server di produzione alla nuova release.
4) Trasferire una copia dei progetti alla seconda organizzazione.
5) Effettuare il test dei progetti nella seconda organizzazione. Quando tutti i test hanno successo, sarà possibile aggiornare tutti i server dell'organizzazione principale.

I server My Cloud

Iniziamo ora a illustrare una seconda modalità di installazione delle applicazioni nel cloud. Invece che utilizzare server di produzione all'interno della piattaforma Instant Developer Cloud, utilizzeremo server in altri cloud, o anche in modalità on-premise, sempre che essi siano raggiungibili dalla console, cioè siano pubblici su internet.

Lo scopo dei server My Cloud è quello di permettere di riutilizzare una propria architettura Cloud per la distribuzione delle applicazioni sviluppate con Instant Developer Cloud. Questo può essere necessario se, ad esempio, si deve operare in una regione dove Google Cloud Platform non è disponibile, come la Cina, o se ci sono altri vincoli di progetto.

In mancanza di vincoli specifici, l'utilizzo di un server My Cloud non è mai conveniente rispetto ad un server Instant Developer Cloud in quanto le operazioni di gestione del server

dovranno essere eseguite manualmente. Nel caso di utilizzo di server My Cloud, tuttavia, le operazioni di controllo delle applicazioni potranno essere effettuate tramite la console di Instant Developer Cloud.

In particolare le seguenti operazioni di gestione del server dovranno essere eseguite manualmente:

- Gestione del sistema di disaster recovery.
- Gestione del database server.
- Controllo delle performance del server.
- Riavvio del server.
- Configurazione domini, alias e certificati.
- Aggiornamento del sistema operativo e delle patch di sicurezza.
- Messa in sicurezza del server. Controllo della sicurezza nel tempo.

Inoltre, Pro Gamma non fornirà supporto tecnico sulla particolare infrastruttura server utilizzata.

Per poter gestire con successo i server My Cloud è quindi necessaria una conoscenza sistemistica completa dell'architettura server utilizzata.

Tramite la console potranno essere eseguite tutte le operazioni di installazione, configurazione e verifica del funzionamento delle applicazioni. Pro Gamma potrà fornire supporto tecnico su quanto concerne la gestione delle applicazioni su server My Cloud.

Acquisto del servizio di interconnessione

Prima di poter utilizzare un proprio server in modalità My Cloud è necessario acquistare il servizio di interconnessione con la console, che permette poi di gestirne le applicazioni.

L'acquisto inizia dalla lista dei server della propria organizzazione, cliccando sul pulsante *Aggiungi server my cloud*.

Oltre alla scelta della taglia del server, in base al numero di processi worker utilizzabili, sarà possibile anche aggiungere i servizi aggiuntivi come Sincronizzazione, Cloud Connector e Analytics. Si noti che i servizi di backup giornaliero o orario non sono disponibili in questa configurazione. Tutti i prezzi indicati sono annuali.

A questo punto il nuovo server entrerà nella lista dei server disponibili e sarà possibile prepararlo per il collegamento vero e proprio.

Preparazione di un server My Cloud

Prima di poter collegare alla console di Instant Developer Cloud un proprio server, è necessario prepararlo installando il framework di base.

Dalla pagina di impostazioni del server My Cloud, cliccare il pulsante *Cambia versione*, e scegliere la versione del framework (solitamente l'ultima).

Di seguito l'elenco delle operazioni da eseguire per preparare il server My Cloud. Gli esempi dei comandi sono relativi ad un server Linux Ubuntu in cui si esegue l'installazione da parte di *root* a partire dalla cartella */idcloud* .

Installazione di Node.js
La versione di riferimento è Node.js 16.4. Per installare la versione si può far riferimento alla documentazione di Node.js, oppure seguire i seguenti passi:

curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
sudo apt-get install -y nodejs

Installazione di npm e pm2
Per installare pm2 è consigliabile aggiornare prima la versione di npm e successivamente installare pm2.

npm install npm@latest -g
npm install pm2 -g

Installazione di Postgresql 13
La versione di riferimento è Postgresql 13.3. Per installare la versione si può fare riferimento alla documentazione oppure eseguire i seguenti passi:

sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add
sudo apt-get update
sudo apt-get -y install postgresql-13

Installazione del pacchetto di versione
Eseguire le seguenti operazioni:

1) Copiare sul server il file di versione scaricato tramite il comando *Cambia versione* dalla pagina di impostazioni del server My Cloud e poi decomprimerlo in una cartella.
2) Creare la cartella *idcloud* in root:
   mkdir /idcloud
3) Rinominare e spostare la cartella ottenuta dalla scompattazione del pacchetto di versione in */idcloud/idserver* ad esempio tramite il seguente comando:
   *mv update_*** /idcloud/idserver*
4) Creare le seguenti cartelle con i seguenti comandi:
   mkdir /idcloud/config
   mkdir /idcloud/idserver/apps
   mkdir /idcloud/idserver/apps/apps
   mkdir /idcloud/idserver/apps/db
   mkdir /idcloud/idserver/apps/backups
   mkdir /idcloud/node_modules
   mkdir /idcloud/idserver/log

5) Scaricare il file config-example.json sul server nella cartella */idcloud/config/* e rinominarlo in *config.json,* ad esempio con il comando seguente:
   mv config-example.json /idcloud/config/config.json

6) Modificare il contenuto del file *config.json* impostando le seguenti proprietà:
   appDirectory: /idcloud/idserver/apps
   alias: lista di IP o domini del server separati da virgola (,)
   dbUser: postgres username
   dbPassword: postgres password
   consoleURL : https://console.instantdevelopercloud.com/CCC",

Aggiornamento dei moduli di Node.js

Alla prima installazione e per tutti i nuovi aggiornamenti è necessario aggiornare i pacchetti *node modules* per allinearsi alla definizione contenuta in *package.json*. Per ottenere questo risultato occorre spostarsi nella cartella */idcloud/idserver*

cd /idcloud/idserver

e lanciare il comando:

npm install

Aggiungere e configurare utente indert

Occorre ora aggiungere l'utente giusto per l'esecuzione del framework di Instant Developer Cloud.

1) Aggiungere l'utente e il gruppo *indert*: *sudo adduser indert*
2) Assegnare i permessi alla cartella *idcloud*
   sudo chown -R indert:indert /idcloud
   sudo chmod -R 755 /idcloud
3) Assegnare i permessi di scrittura alla cartella *db*
   chown -R postgres:postgres /idcloud/idserver/apps/db

Avviare server.js con PM2

Infine occorre configurare PM2 per avviare il server Node.js:

cd /idcloud/idserver/server
pm2 start server.js
pm2 save
pm2 startup ubuntu

Collegamento alla console

Una volta completata la procedura di installazione, è possibile completare la configurazione del server nella console, inserendo i seguenti dati nell'ultimo riquadro:

1. Indirizzo IP pubblico del server.
2. Protocollo di connessione (lasciare HTTPS).
3. Porta HTTP (lasciare il valore di default indicato).
4. Porta HTTPS (lasciare il valore di default indicato).
5. Porta Node.js (lasciare il valore di default indicato).

Per completare la procedura di collegamento, cliccare il pulsante *Connetti il server* in alto nella pagina. Al termine della procedura il server risulterà connesso e sarà possibile installare e gestire le applicazioni.

Installazione delle applicazioni

L'installazione delle applicazioni su un server My Cloud funziona nello stesso modo dei server della piattaforma Instant Developer Cloud. Per poter installare applicazioni, il server My Cloud deve risultare connesso alla piattaforma.

L'unica differenza riguarda la configurazione dei processi worker per l'applicazione. Siccome il server My Cloud ha un limite sul numero di processi worker utilizzabili in funzione della sua taglia, la configurazione dei processi tiene conto di questo limite.

Gestione delle applicazioni

La gestione delle applicazioni installate su un server My Cloud funziona nello stesso modo dei server della piattaforma Instant Developer Cloud. Per poter gestire le applicazioni, il server My Cloud deve risultare connesso alla piattaforma.

Gestione del server

La gestione di un server My Cloud tramite la console di Instant Developer Cloud ha funzionalità limitate rispetto ai server della piattaforma. In particolare non sono previste le seguenti pagine e le relative funzionalità:

1. Monitoraggio delle attività: il sistema di monitoraggio delle attività non è disponibile.
2. Configurazione dei worker: avviene solo a livello di applicazione.
3. Backup: il sistema di backup e ripristino del server non è disponibile.

Limitazioni

I server My Cloud non possono essere utilizzati per ospitare soluzioni basate su Instant Developer Chatbot, Instant Developer Support e per installare il configuratore di Instant Developer Creator.

I server My Cloud inoltre non possono essere usati come server per il test automatico delle applicazioni.

I server Self Managed

I server Self Managed sono server di produzione esterni alla piattaforma Instant Developer Cloud e quindi completamente sconosciuti alla console. Rappresentano quindi una modalità di installazione e gestione delle applicazioni completamente manuale e autogestita.

Si consiglia l'utilizzo di server Self Managed solo se si ha a disposizione un sistemista esperto nella gestione dei server, della sicurezza dell'infrastruttura e delle applicazioni, della gestione dei database e dell'ambiente operativo Node.js. In mancanza di una o più di queste conoscenze potrebbe non essere possibile portare a termine l'installazione, oppure si potrebbero verificare problemi di vario tipo per l'applicazione, ad esempio interruzioni di servizio o un basso livello di sicurezza.

Essendo questi server completamente esterni alla piattaforma Instant Developer Cloud, Pro Gamma non offre alcun servizio di assistenza su eventuali problematiche riscontrate, sia a livello di server, che di installazione e gestione delle applicazioni. Se si desidera sottoporre al servizio di assistenza una problematica riscontrata, sarà necessario prima riprodurla nell'ambiente di sviluppo o in un server di produzione interno alla piattaforma.

I server Self Managed, tuttavia, possono risultare una scelta obbligata quando l'applicazione deve essere utilizzata all'interno di una rete aziendale chiusa, senza accesso ad internet.

Preparazione di un server Self Managed

La preparazione di un server Self Managed è analoga a quella dei server My Cloud. Si consiglia di accedere al repository instant-developer-platform per scaricare i sorgenti del framework e quindi seguire la procedura di installazione contenuta nel file *readme*, sostanzialmente analoga a quella dei server My Cloud. Gli eventuali pacchetti aggiuntivi di Node.js dovranno essere gestiti manualmente.

Configurazione dei database

La configurazione dipende dal tipo di database utilizzato dall'applicazione. Se l'applicazione utilizza un database di tipo Postgres, è possibile indicare i parametri di connessione nel file *config.json* specificando le proprietà:

```
"dbPort": "db server port",
"dbAddress": "db server host",
"dbUser": "postgres user",
"dbPassword": "postgres password",
```

Altri tipi di database

Se invece l'applicazione utilizza un database MySQL, SQLServer oppure Oracle, è necessario impostare le opzioni di connessione nel codice dell'applicazione prima della connessione all'istanza.

A tal fine, è necessario reperire l'istanza di default del database tramite l'istruzione *App.<Nome DB>.getDefaultInstance()* e successivamente impostare su di essa i parametri di connessione con il metodo *setOptions()*. Vediamo un esempio:

```
let db = App.<DataModelName>.getDefaultInstance(app);
db.setOptions(connectionOptions);
```

Il contenuto di *connectionOptions* varia in relazione del tipo di database da utilizzare e, di conseguenza, al driver specifico.

MySQL

Per la connessione ai database di tipo MySQL viene utilizzato il driver Node.js mysql che viene installato insieme agli altri pacchetti del framework.

Il parametro *connectionOptions* in questo caso deve essere impostato come segue:

```
let connectionOptions = {
  host : "hostname",
  database : "database name",
  user : "username",
  password : "password",
  connectionLimit : 10
});
```

Per l'elenco completo dei parametri di connessione ad un database MySQL tramite il driver node, è possibile consultare la documentazione completa.

SQL Server

Per la connessione ai database di tipo SQL Server viene utilizzato il driver Node.js mssql che viene installato insieme agli altri pacchetti del framework.

Il parametro *connectionOptions* in questo caso deve essere impostato come segue:

```
let connectionOptions = {
  server : "hostname\\instance",
  database : "database",
  user : "username",
  password : "password",
  options : {
    useUTC : false
  }
});
```

Per l'elenco completo dei parametri di connessione ad un database SQL Server tramite il driver node, è possibile consultare la documentazione completa.

Oracle

Per la connessione ai database di tipo Oracle viene utilizzato il driver oracledb, non installato di default insieme agli altri pacchetti del framework.

Per eseguire l'installazione del driver e delle componenti necessarie che dipendono dal sistema operativo del server, è possibile consultare la documentazione ufficiale.

Il parametro *connectionOptions* in questo caso deve essere impostato come segue:

```
let connectionOptions ={
  connectString : "hostname/servicename",
  user : "username",
  password : "password"
});
```

Gestione dei certificati

Per attivare il protocollo https ed utilizzare i propri certificati SSL sul server, è sufficiente copiare i file sul server stesso e modificare opportunamente il file *config.json*.

Ipotizzando che il server risponda all'indirizzo *https://mysrv.mydomain.it* sulla porta default 443 e che i file dei certificati siano stati copiati nella directory */idcloud/config/cert* il file *config.json* deve essere modificato nel seguente modo.

```
"domain": "mydomain.com",
"alias" : "mysrv.mydomain.it",
"protocol": "https",
"portHttps": 443,
"SSLCert": "/idcloud/config/cert/mydomain_it_certificate.crt",
"SSLKey": "/idcloud/config/cert/mydomain_it_private.key",
"SSLCABundles": [
    "/idcloud/config/cert/mydomain_it_ca_bundle.crt"
],
```

È possibile aggiungere al file di configurazione la proprietà *customSSLCerts* per fare in modo di utilizzare uno specifico certificato a seconda della URL di destinazione del server.

Se ad esempio il server è configurato, tramite DNS, a rispondere anche all'indirizzo *https://mysrv.myotherdomain.it*, modificando la proprietà *alias* ed aggiungendo *customSSLCerts* è possibile utilizzare un diverso certificato se il server è stato raggiunto utilizzando questo nome. Ad esempio:

```
"alias" : "mysrv.mydomain.it, mysrv.myotherdomain.it",
…
"customSSLCerts": [{
  "SSLDomain": "myotherdomain.it",
  "SSLCert": "/idcloud/config/cert/myotherdomain_it_certificate.crt",
  "SSLKey": "/idcloud/config/cert/myotherdomain_it_private.key",
  "SSLCABundles": [
    "/idcloud/config/cert/myotherdomain_it_ca_bundle.crt"
  ]
}],
```

Pacchetti aggiuntivi

Per il corretto funzionamento del framework, qualora venga usato il package GraphicsMagick, è necessario eseguire il download e l'installazione del pacchetto relativo nel server Self Managed. Si consiglia di leggere la documentazione ufficiale.

Installazione delle applicazioni

Per installare un'applicazione per la prima volta, eseguire le seguenti operazioni:

1. Configurare i database usati dall'applicazione nel file di configurazione.
2. Aggiornare manualmente lo schema dei database usati dall'applicazione o ripristinare un backup aggiornato di tali database.

3. Eseguire una build dell'applicazione tramite la console senza richiedere l'installazione su un server.
4. Dalla pagina di *App e dati* del menu di progetto, scaricare il pacchetto di installazione relativo alla build effettuata.
5. Decomprimere il pacchetto di installazione nella cartella: *idcloud/apps/apps/<app-name>*
6. Modificare il file *idcloud/config/config.json,* aggiungendo all'array *apps* un oggetto contenente le seguenti proprietà:
   apps = [{ "cl" : "Node.App", "name" : "app-name", "date" : "current-date in ISO string", "stopped" : false}]
7. Riavviare il server tramite: *pm2 restart all.*

L'aggiornamento di un'installazione avviene tramite i seguenti passi:

1. Eseguire la build e scaricare il pacchetto.
2. Arrestare il server tramite *pm2*.
3. Aggiornare lo schema dei database manualmente.
4. Spostare la sottocartella *files* dell'applicazione attuale in una posizione temporanea.
5. Eliminare la cartella relativa all'applicazione attuale.
6. Decomprimere il pacchetto di installazione nella cartella: *idcloud/apps/apps/<app-name>*
7. Ripristinare la sottocartella *files* dalla posizione temporanea a quella originale.
8. Riavviare il server tramite *pm2*.

Attivare la Server Session

Per attivare la Server Session di un'applicazione è necessario editare il file *idcloud/config/config.json* impostando a true la proprietà *startSS* dei parametri dell'applicazione.

```
"cl" : "Node.App",
"name" : "app-name",
"date" : "current-date in ISO string",
"stopped" : false,
"startSS": true
```

Impostare l'applicazione di default

È possibile impostare quale applicazione deve essere eseguita editando nel file *idcloud/config/config.json* la proprietà *alias* aggiungendo il nome dell'applicazione al nome del dominio o indirizzo ip preceduta dal carattere '|'.
Ad esempio:

```
"alias" : "mysrv.mydomain.it|app-name"
```

Configurazione dei processi worker per le applicazioni

Per configurare i valori di default per i worker delle applicazioni installate, è necessario modificare il file *config.json* aggiungendo i valori desiderati per le proprietà *maxAppUsers*, *minAppUsersPerWorker* e *maxAppWorkers*, come nel seguente esempio:

```
"maxAppUsers": 1000,
"minAppUsersPerWorker": 50,
"maxAppWorkers": 4,
```

La corrispondenza fra queste proprietà e quelle descritte nel capitolo Configurazione dei worker è la seguente:

- *maxAppUsers* = Numero massimo di utenti per app
- *minAppUsersPerWorker* = Numero minimo di utenti per worker
- *maxAppWorkers* = Numero massimo di worker per app

Per definire le politiche di gestione dei processi worker per una singola applicazione installata è necessario aggiungere, all'interno della configurazione dell'applicazione, contenuta nella sezione *apps* del file *config.json*, la proprietà *customWorkerConf*.
In questa proprietà di tipo array è possibile specificare le configurazioni dei parametri per ogni tipo di sessione, come mostrato nel seguente esempio:

```
"apps": [ {
  "cl": "Node.App",
  "name": "app-name",
  "date" : "2021-10-11T10:41:06.039Z"
  "stopped": false,
  "customWorkerConf": [{
      "type": "web",
      "query": null,
      "priority": 0,
      "idleTimeout": null,
      "maxAppUsers": 200,
      "minAppUsersPerWorker": 50,
      "maxAppWorkers": 2
    }, {
      "type": "rest",
      "query": null,
      "priority": 0,
      "idleTimeout": 10000,
      "maxAppUsers": 100,
      "minAppUsersPerWorker": 10,
      "maxAppWorkers": 1
  }]
}]
```

La corrispondenza fra queste proprietà e quelle descritte nel capitolo Configurazione dei processi worker è la seguente:

- *type* = Tipo di sessione
- *query* = Query string

- *priority* = Priorità
- *idleTimeout* = Timeout inattività
- *maxAppUsers* = Numero totale utenti complessivi
- *minAppUsersPerWorker* = Numero minimo di utenti per processo
- *maxAppWorkers* = Numero massimo di processi

Limitazioni

I server Self Managed sono soggetti alle seguenti limitazioni:

- Quelle già esistenti per i server My Cloud.
- Non possono ospitare il servizio Cloud Connector.
- Non possono ospitare il servizio Analytics e Feedback.
- Non possono ospitare il servizio di sincronizzazione dei database locali delle applicazioni. È invece possibile utilizzare l'accesso ai dati del server tramite Document Orientation Remota.

Tabella comparativa

Installazione delle applicazioni

Funzionalità	Server Instant Developer Cloud	Server My Cloud	Server Self Managed
Gestione sessioni attive al momento dell'installazione	Automatica: le sessioni esistenti vengono avvertite dell'imminente installazione, invitandole a salvare il lavoro in corso. Dopo un timeout configurabile, l'applicazione viene installata.	Automatica: le sessioni esistenti vengono avvertite dell'imminente installazione, invitandole a salvare il lavoro in corso. Dopo un timeout configurabile, l'applicazione viene installata.	A scelta fra: - Interruzione sessioni esistenti con perdita dei lavori in corso. - Blocco dell'applicazione fino al termine di tutte le sessioni esistenti. - Operazioni di installazione in orari notturni o giorni festivi.
Snapshot del server al momento dell'inizio dell'installazione.	Automatico, in tempo reale.	Deve essere eseguito manualmente, può richiedere decine di minuti o ore in funzione della dimensione del server	Deve essere eseguito manualmente, può richiedere decine di minuti o ore in funzione della dimensione del server
Backup del database prima dell'aggiornamento dello schema	Non necessario.	Deve essere eseguito manualmente, può richiedere decine di minuti o ore in funzione della dimensione del server	Deve essere eseguito manualmente, può richiedere decine di minuti o ore in funzione della dimensione del server
Aggiornamento dello schema del database in funzione delle modifiche all'applicazione	Automatica, con rollback automatico dell'intero processo di installazione in caso di eccezioni.	Automatica, con rollback automatico dell'intero processo di installazione in caso di eccezioni (solo database Postgres).	Le modifiche devono essere programmate e installate manualmente. Può richiedere diverse ore di lavoro ed è facile sbagliare e perdere dati.
Aggiornamento dei file e delle risorse dell'applicazione, rimozione dei file inutilizzati.	Automatica	Automatica	Le modifiche devono essere apportate manualmente con il rischio di errori o di lasciare file non aggiornati o contenenti dati personali
Riavvio e rientro delle sessioni esistenti	Automatica	Automatica	Manuale, la gestione delle sessioni esistenti non è possibile
Supporto tecnico su Installazione Applicazioni	Disponibile	Disponibile	Non disponibile

Gestione delle applicazioni

Funzionalità	Server Instant Developer Cloud	Server My Cloud	Server Self Managed
Controllo e storicizzazione del numero di utenti collegati all'applicazione	Tramite pannello nella console di Instant Developer Cloud	Tramite pannello nella console di Instant Developer Cloud	Tramite gli strumenti specifici dell'application server utilizzato. Richiede solitamente un accesso specifico al server.
Log strutturato di ogni sessione in real time	Log strutturato sessione per sessione disponibile nella console IDC	Log strutturato sessione per sessione disponibile nella console IDC	Disponibile solo il log del server, tutto insieme.
Debug in tempo reale delle sessioni server senza interruzione di servizio	Disponibile	Disponibile	Non disponibile
Impostazione parametri applicativi specifici del server	Disponibile	Disponibile	Deve essere gestito manualmente
Gestione del file system	Tramite pannello nella console di Instant Developer Cloud	Tramite pannello nella console di Instant Developer Cloud	Richiede un accesso al server, potenzialmente attaccabile.
Gestione del database	Tramite pannello nella console di Instant Developer Cloud	Tramite pannello nella console di Instant Developer Cloud	Richiede un accesso al server, potenzialmente attaccabile.
Supporto tecnico su Gestione delle applicazioni	Disponibile	Disponibile	Non disponibile

Gestione del server

Funzionalità	Server Instant Developer Cloud	Server My Cloud	Server Self Managed
Controllo in tempo reale dei principali parametri del server: CPU e memoria utilizzata	Tramite pannello nella console di Instant Developer Cloud	Tramite gli strumenti specifici dell'application server utilizzato. Richiede solitamente un accesso specifico al server.	Tramite gli strumenti specifici dell'application server utilizzato. Richiede solitamente un accesso specifico al server.
Operazioni di riavvio dell'application server e/o del server	Tramite pannello nella console di Instant Developer Cloud	Tramite gli strumenti specifici dell'application server utilizzato. Richiede solitamente un accesso specifico al server.	Tramite gli strumenti specifici dell'application server utilizzato. Richiede solitamente un accesso specifico al server.
Configurazione domini, alias e certificati	Automatica, certificati compresi	Operazioni manuali che richiedono l'acquisto di certificati da enti certificatori	Operazioni manuali che richiedono l'acquisto di certificati da enti certificatori
Aggiornamento del sistema operativo, installazione patch di sicurezza	Automatica e in tempo reale	Configurazione manuale. Può causare interruzioni di servizio.	Configurazione manuale. Può causare interruzioni di servizio.
Architettura a container (docker)	Automatica	Deve essere configurata manualmente	Deve essere configurata manualmente
Architettura sigillata e testata da specialisti in sicurezza informatica.	Disponibile	Non disponibile	Non disponibile
Sistemi di backup dei dati e del server	Disponibili snapshot giornalieri o orari dell'intero sistema	Devono essere configurati manualmente	Devono essere configurati manualmente
Sistema di disaster recovery in tempo reale	Ripristino snapshot in qualche minuto.	Non disponibile	Non disponibile
Supporto tecnico sulla gestione del server	Disponibile	Non disponibile	Non disponibile

Servizi aggiuntivi

Funzionalità	Server Instant Developer Cloud	Server My Cloud	Server Self Managed
Servizio Cloud Connector	Disponibile	Disponibile	Non disponibile
Sincronizzazione database locali applicazioni mobile	Disponibile	Disponibile	Non disponibile
Servizio Analytics e Feedback	Disponibile	Disponibile	Non disponibile
Test automatico di non regressione e di carico	Disponibile	Non disponibile	Non disponibile
Soluzioni basate su Instant Developer Chatbot	Disponibile	Non disponibile	Non disponibile
Soluzioni basate su Instant Developer Support	Disponibile	Non disponibile	Non disponibile
Configuratore Instant Developer Creator	Disponibile	Non disponibile	Non disponibile