Come sfruttare i servizi gestiti di Aws e i container Docker per semplificare la gestione di un sito WordPress su AWS

20 Agosto 2021 - 7 min. read

Containers | Docker

WordPress è la piattaforma di creazione contenuti più facile ed immediata da utilizzare. La flessibilità che mette a disposizione è ciò che la rende la piattaforma più diffusa in assoluto tra gli autori: utilizzando alcuni plugin è possibile creare e mantenere in modo semplice e con pochi clic qualsiasi tipo di contenuto, da semplici gallerie di immagini, fino ai più complessi siti di e-commerce.

Se per gli autori di contenuti WordPress risulta essere quindi una vera e propria manna dal cielo, dal punto di vista di chi lavora nell'IT è spesso considerato un software infernale. Garantirne il buon funzionamento è un compito che tiene svegli sistemisti e DevOps di tutto il mondo: la scalabilità non è immediata, l'installazione non è scriptata ed uno stack LAMP non è facile da mantenere aggiornato.

In questo articolo vi forniremo alcuni suggerimenti tecnici per rendere meno tragica la gestione di WordPress ospitati in un ambienti Cloud. In particolare,sfrutteremo i servizi offerti dal provider Amazon Web Services utilizzando il maggior numero possibile di servizi gestiti. In questo modo, elimineremo tutte le attività ripetitive, e quindi potenzialmente più pericolose, semplificando notevolmente gestione e manutenzione.

Database

Per prima cosa, vogliamo che il nostro database possa essere altamente affidabile e che riesca a scalare sostenendo senza fatica l'andamento del traffico. La miglior scelta per soddisfare questi requisiti risiede nell'utilizzo di Amazon Aurora MySQL.

Utilizzando Amazon Aurora, infatti, non dovremo preoccuparci dello spazio utilizzato perché lo storage può scalare automaticamente quando necessario. Inoltre, il point-in-time recovery rende possibile il recupero dei dati con la granularità di secondi.

Sono possibili anche configurazioni "Multi-AZ" con repliche in lettura (read replica) che possono essere promosse in caso di fallimento di una availability zone: quando si crea un cluster Aurora, AWS assegna automaticamente un endpoint (a cui è associato un record DNS) per la lettura e uno per la scrittura.

In caso di fallimento (o di operazioni di manutenzione) la read replica viene automaticamente promossa per diventare l'istanza di scrittura e l'endpoint viene modificato di conseguenza.

Basta semplicemente impostare il file wp-config.php per utilizzare l'endpoint di scrittura ed AWS si occuperà di fare tutto il resto. Con Amazon Aurora è possibile aggiungere fino a 15 read replica.

Se il tipo di traffico che il sito dovrà sopportare sarà prevalentemente di lettura è possibile sfruttare questa possibilità, ma occorre fare in modo che WordPress riesca ad utilizzare tutte le repliche. Il plugin hyperdb (https://wordpress.org/plugins/hyperdb/) permette proprio di fare questo, definendo un database master ed un numero di repliche a piacere.
Il file di configurazione è ben documentato (https://plugins.trac.wordpress.org/browser/hyperdb/trunk/db-config.php) e questa è una configurazione di esempio:

```
wpdb->add_database(array(
 'host' => mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com,
 'user' => DB_USER,
 'password'     => DB_PASSWORD,
  'name'        => DB_NAME,
));
```

```
$wpdb->add_database(array(
 'host'         => mydbcluster.cluster-ro-123456789012.us-east-1.rds.
amazonaws.com,
 'user' => DB_USER,
 'password'     => DB_PASSWORD,
 'name' => DB_NAME,
 'write'        => 0,
 'read' => 1,
));
```

In caso di basso traffico al sito o di traffico incerto in cui non è noto quando e se sia necessario scalare, invece, la miglior scelta possibile ricade su Aurora Serverless. Questo database ha la possibilità di scalare la capacità computazionale in modo automatico e di rimanere “in pausa” quando il sito web non è utilizzato, permettendo di ottimizzare al massimo i costi.

Compute

Vogliamo poter utilizzare l’elasticità che i servizi cloud mettono a disposizione, per cui gli Autoscaling Group di EC2 potrebbero sembrare la soluzione più naturale. Ci spingeremo oltre e utilizzeremo container Docker su un cluster ECS in modalità Fargate, anche se sarà necessario un po’ di refactoring applicativo.

L’utilizzo di container, infatti, riduce le attività di manutenzione necessarie per l’aggiornamento dei sistemi operativi e rende più semplice implementare la scalabilità. Un ulteriore vantaggio ottenibile con poco sforzo deriva dalla possibilità di automatizzare l'intero flusso di deployment attraverso l'utilizzo di Pipeline. Più avanti nell’articolo vedremo come fare.

WordPress mette a disposizione una immagine docker già pronta con una installazione di default: https://hub.docker.com/_/wordpress

Possiamo utilizzare questa immagine ed adattarla alle nostre esigenze, automatizzando l’installazione di plugin mediante l’uso di wp-cli. In questo esempio installeremo il plugin wordpress-seo (assumendo che abbiate già a disposizione il file wp-config.php).

La configurazione e le soluzioni proposte in questo articolo sono solo di esempio. Per utilizzarla dovrete prima adattarla alle vostre esigenze. Vi raccomandiamo inoltre di

indagare sempre sul software in esecuzione nei vostri ambienti, specialmente in soluzioni basate su container.

Ecco un esempio di Dockerfile:

```
FROM wordpress

COPY wp-config.php /usr/src/wordpress/

RUN  curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar && chmod +x wp-cli.phar && mv wp-cli.phar /usr/local/bin/wp

WORKDIR /usr/src/wordpress/

RUN wp --allow-root core update
RUN wp --allow-root plugin install wordpress-seo

WORKDIR /var/www/html
```

In questo modo abbiamo automatizzato la nostra installazione, rendendola manutenibile e pronta per test in ambienti differenti.

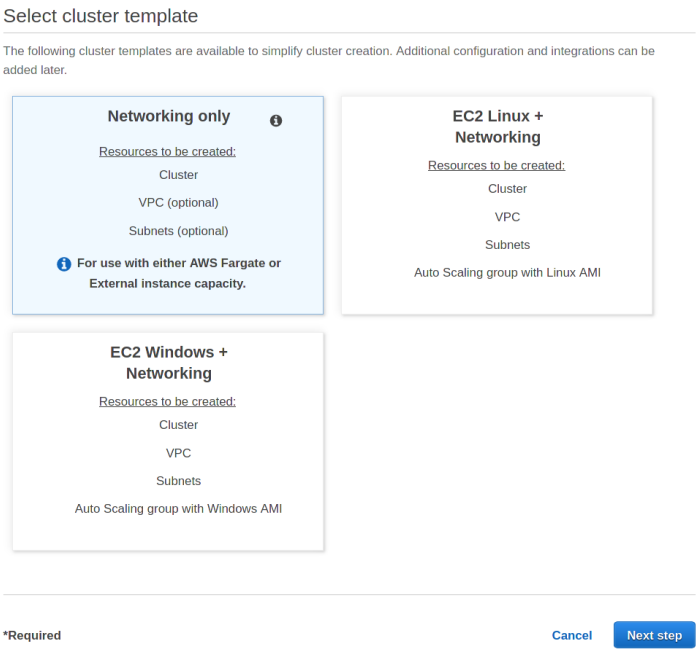
È possibile utilizzare docker-compose per effettuare il test o, in alternativa, eseguire

```
docker build . -t myawesomewordpresscontainer
```

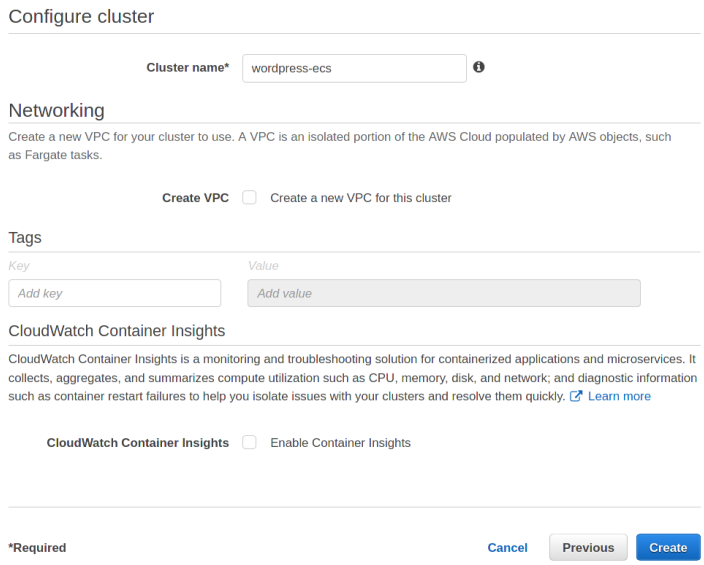
per avere un container pronto di cui fare deploy.

Creare un ambiente serverless AWS ECS è molto semplice:

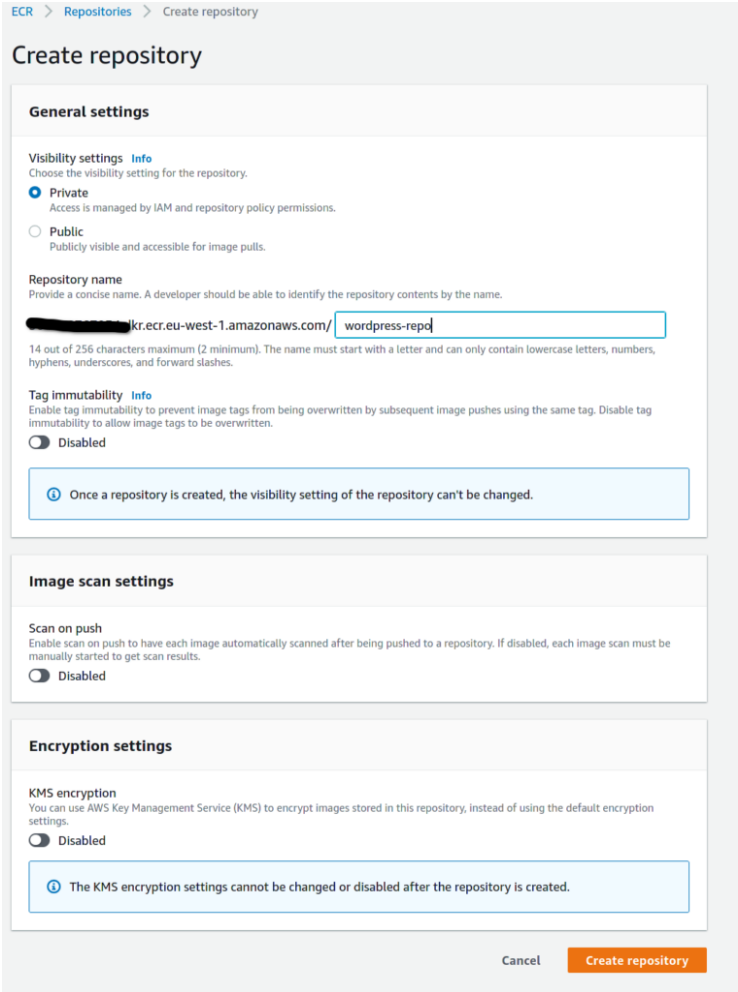
Sulla AWS Console fare click su ECS -> Create Cluster, selezionare “Networking Only” per creare un cluster Fargate:



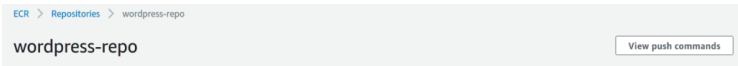
Diamogli un nome:



A questo punto è necessario creare un repository Docker per l’immagine. Basta fare click su “Amazon ECR” - Repositories.

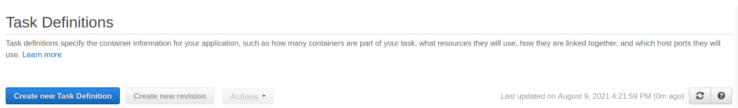


Facendo click sul repository appena creato è possibile avere le istruzioni pronte all'uso per fare la build e l'upload del container facendo click sul pulsante "View push commands".

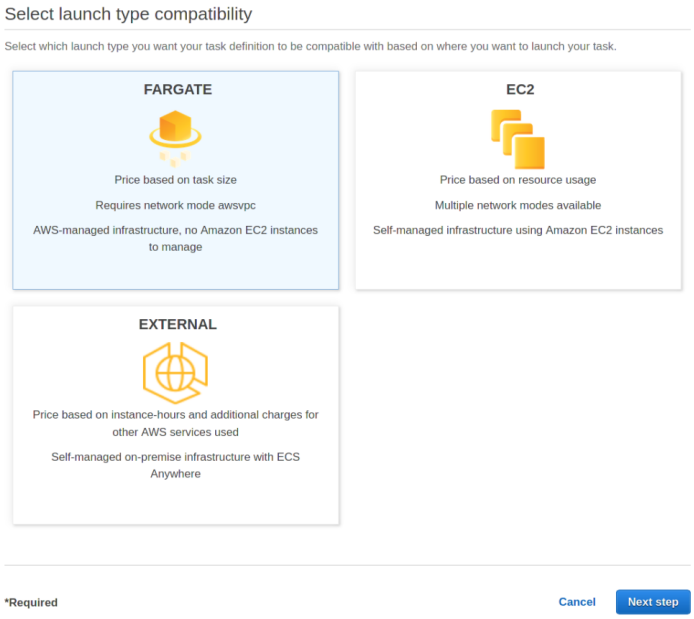


Terminato l'upload dell'immagine del container è possibile definire una task definition che si occuperà di definire come eseguire il nostro container nel cluster ECS.

Facendo click su "task definitions" sulla barra a lato e selezionando "Create new Task Definition" si avvierà la procedura guidata di creazione:

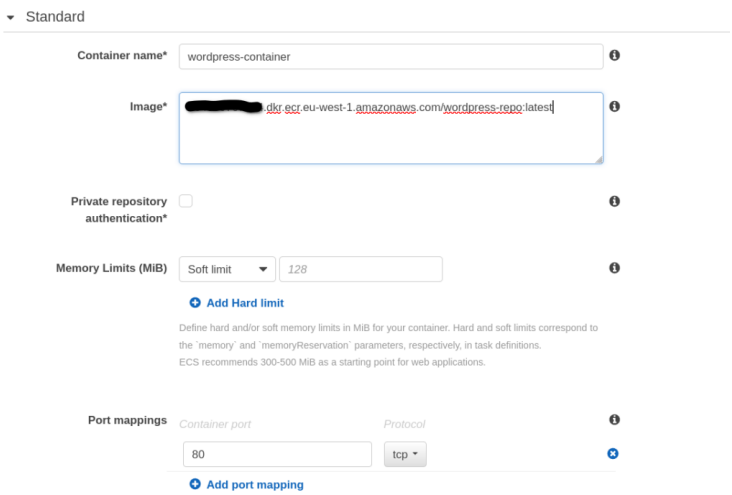


Selezionare Fargate

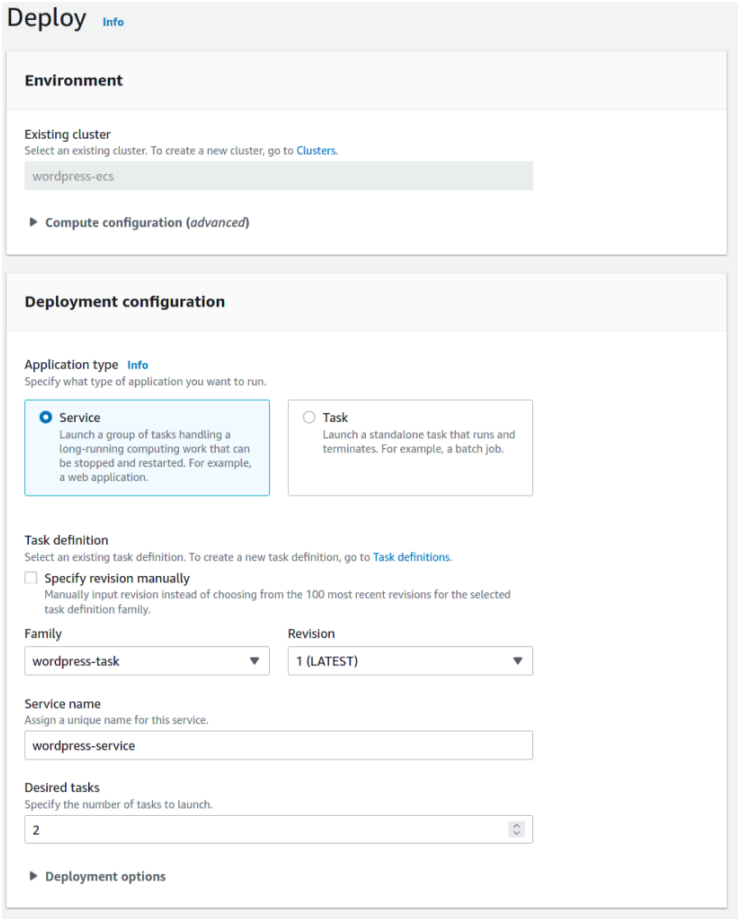


A questo punto occorre solo dare un nome alla task definition e definire le risorse assegnate. Per questa piccola demo selezioneremo 0.25 unità per la CPU e 0.5GB di memoria. È sempre buona cosa tenere presente che è meglio avere la possibilità di scalare orizzontalmente utilizzando più container piccoli.

Facendo click su “add container” si specificano i parametri per l’ambiente di esecuzione Docker, facendo attenzione a definire un mapping per esporre la porta 80 del webserver.



Terminata la creazione della task definition occorre solamente passare alla creazione di un service:



Il service appena definito si occupa di fare il deploy e di mantenere attivi due task, in questo modo la nostra installazione di wordpress sarà in altamente affidabile e con bilanciamento del traffico.

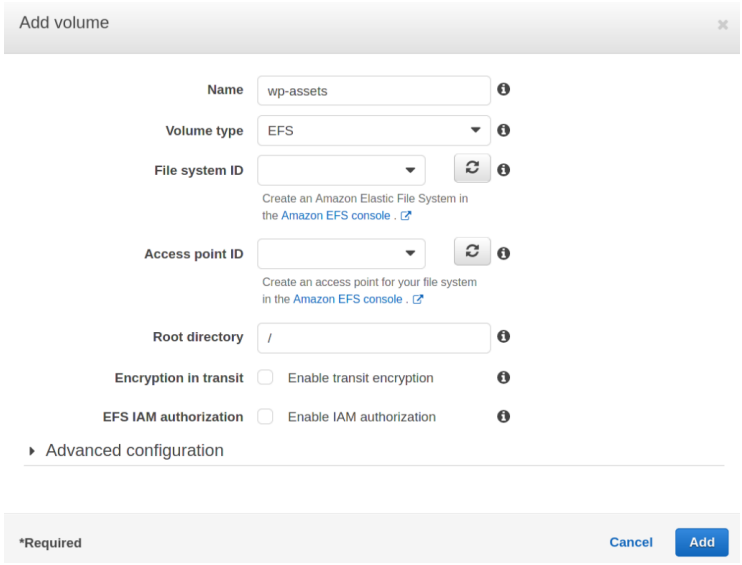
Essendo configurazioni ordinarie su AWS, non ci occuperemo in questo articolo della creazione del load balancer e dei security group associati. Se volete saperne di più, scriveteci!

Storage

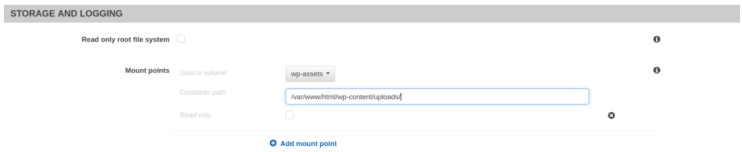
I container Docker sono stateless (per definizione), quindi rimane solo un ultimo tema da affrontare: la persistenza dei dati statici.

In questo caso il servizio di riferimento è Amazon EFS: un filesystem distribuito e condiviso.

Possiamo memorizzare le risorse statiche contenute nella directory wp-content/uploads aggiungendo semplicemente un file system EFS nella task definition:



Dopo aver assegnato il filesystem il container è in grado di utilizzarlo:



Non dimenticatevi di definire un job di AWS Backup per la share EFS !

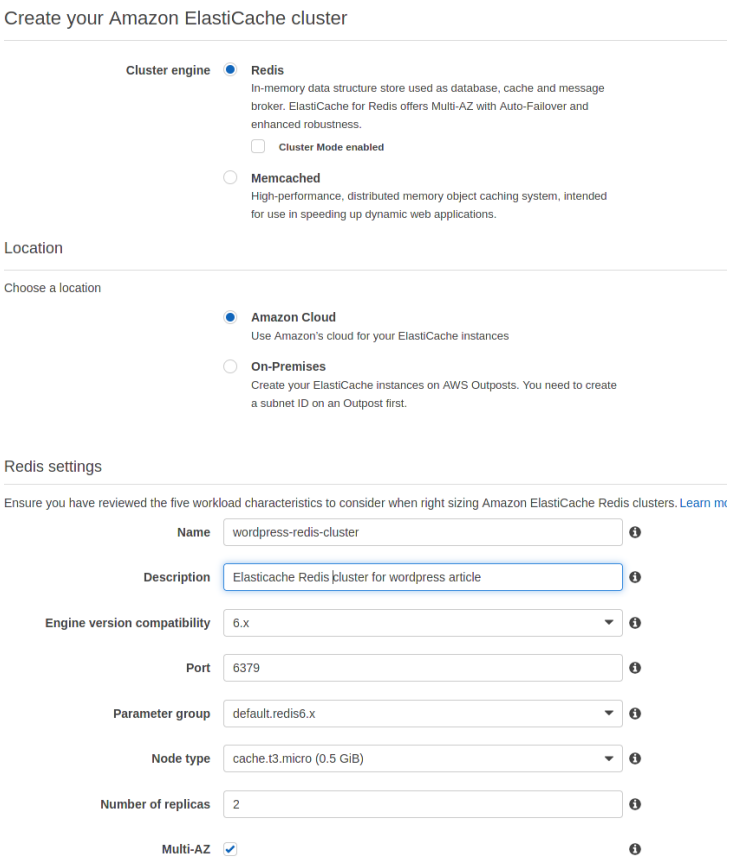
Esistono anche plugin WordPress per utilizzare S3 che non useremo in questo articolo per mantenere il nostro esempio semplice. È bene però sapere che esistono per poterli tenere in considerazione nel caso di un'installazione in ambiente di produzione.

Caching

Come regola generale, anche se non mostreremo la procedura passo passo, è possibile ridurre i costi e rendere il sito più veloce per gli utenti aggiungendo una distribuzione CloudFront in fronte al load balancer.

Per quanto riguarda la cache applicativa è possibile usare Amazon Elasticache for Redis ed il plugin WordPress redis-cache per ridurre il carico delle query al database e mantenere le sessioni utente.

Il deploy richiede solo alcuni minuti: basta cliccare sul pulsante “Create” sulla Console AWS alla voce “Elasticache”.



Per mantenere il cluster privato ed in alta affidabilità occorre creare un nuovo subnet group selezionando almeno due subnet in due Availability Zones differenti. Le impostazioni si trovano alla sezione “Advanced Redis settings”. Per fare in modo che i container WordPress riescano a raggiungere il servizio Redis basta selezionare o creare un security group adatto allo scopo.

Una volta fatto click sul pulsante “Create” il cluster sarà disponibile nel giro di alcuni minuti.

Terminata la creazione del cluster basterà aggiungere al Dockerfile la riga:

```
RUN --allow-root plugin install redis-cache
```

E successivamente configurare il plugin per fare cache delle sessioni utente utilizzando il primary endpoint visualizzabile nei dettagli del cluster.

WAF

Per mitigare attacchi e vulnerabilità applicative possiamo tenere in considerazione l’attivazione di AWS WAF, abilitando il set di regole “Managed AWS Rules for AWS Firewall”. Il servizio WAF è associabile all’Application Load Balancer o alla distribuzione CloudFront (se avete optato per utilizzarla).

Manutenzione ed aggiornamento (le Pipeline)

La nostra scelta per l'automatizzazione della manutenzione del nostro container WordPress è l'accoppiata CodeBuild/CodePipeline.

Per questi aspetti basta seguire l'ottimo articolo di Alessio (https://www.proud2becloud.com/aws-fargate-services-deployment-with-continuous-delivery-pipeline/) e la strategia di deploy Blue/Green descritta da Alessandro (https://www.proud2becloud.com/how-to-setup-a-continuous-deployment-pipeline-on-aws-for-ecs-blue-green-deployments/)

Con i giusti servizi managed è possibile mantenere l'attenzione ed il focus sulla creazione di contenuti: delegando le operazioni per mantenere alta affidabilità e riducendo i costi operativi si rende più facile l'implementazione ed il deploy di un ambiente WordPress a partire da zero.

Damiano Giorgi

Ex sistemista on-prem, pigro e incline all'automazione di task noiosi. Alla ricerca costante di novità tecnologiche e quindi passato al cloud per trovare nuovi stimoli.L'unico hardware a cui mi dedico ora è quello del mio basso; se non mi trovate in ufficio o in sala prove provate al pub o in qualche aeroporto!

Copyright © 2011-2021 by beSharp srl - P.IVA IT02415160189