GO SERVERLESS! PARTE 2: L'INFRASTRUTTURA PER L'APPLICAZIONE DI FILE SHARING

AWS Lambda | CI/CD | Continuous Delivery | Serverless

beSharp | 25 Settembre 2018

Leggi la prima parte | Leggi la terza parte

Con la sempre maggiore diffusione dello sviluppo Serverless, è di fondamentale importanza comprendere come sfruttarne tutti i vantaggi per poter realizzare applicazioni **scalabili** e **resilienti**.

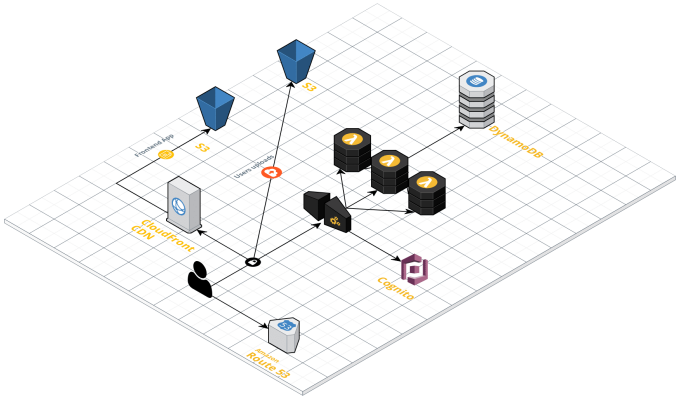
Questa è la seconda parte della serie di 3 articoli (qui trovate la prima parte) su come costruire una **piattaforma di file sharing completamente serverless**. In questo articolo andremo a realizzare l'infrastruttura, approfondendo gli aspetti fondamentali per una corretta ed efficace implementazione.

Come anticipato nell'articolo precedente, oltre all'infrastruttura dell'applicazione andremo anche a realizzare le **pipeline di Continuous Integration e Continuous Delivery**.

Cominciamo con il primo dei due punti.

Non tratteremo qui del codice applicativo e degli argomenti correlati allo sviluppo, che saranno descritti in modo approfondito nel prossimo articolo. Per testare l'infrastruttura impiegheremo delle semplici pagine statiche e una funzione lambda di test.

Iniziamo con l'analisi della soluzione presentata nella prima parte della serie.



L'applicazione è composta da un **front-end** e un **back-end**

Il front-end è una single page application di cui effettueremo il deploy su S3 e che sarà servita mediante **CloudFront**. Il back-end invece è una API REST realizzata con **API Gateway** e **Lambda**.

Per realizzare l'infrastruttura occorre effettuare il provisioning e la configurazione delle seguenti risorse AWS:

front-end

1. Un bucket S3 privato per il front-end
2. Una distribuzione di CloudFront con il permesso di accedere al bucket S3

back-end

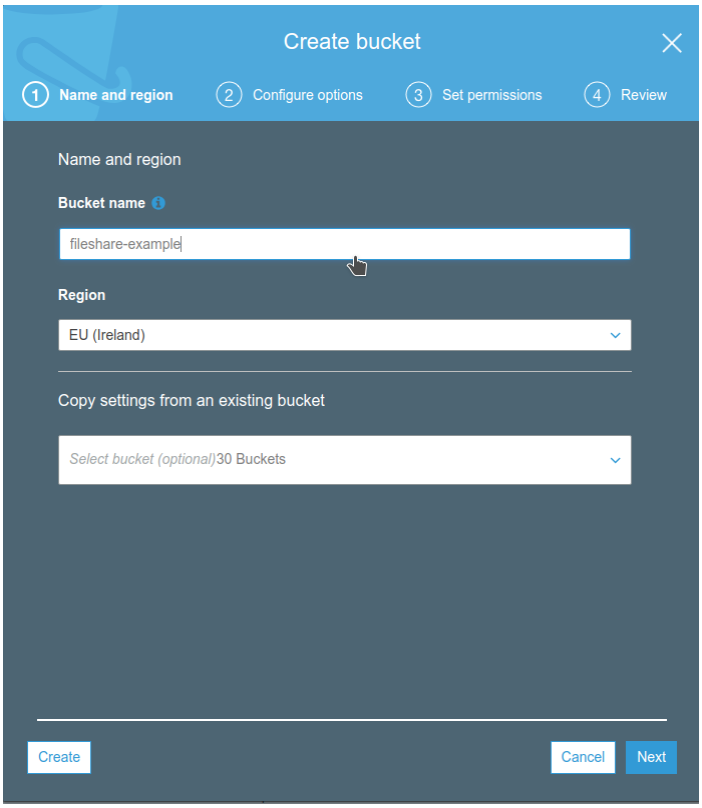
1. Una funzione lambda per testare l'integrazione
2. Una Cognito User Pool
3. Un Bucket S3 privato per gli upload degli utenti
4. Una tabella DynamoDB
5. Una API di API Gateway

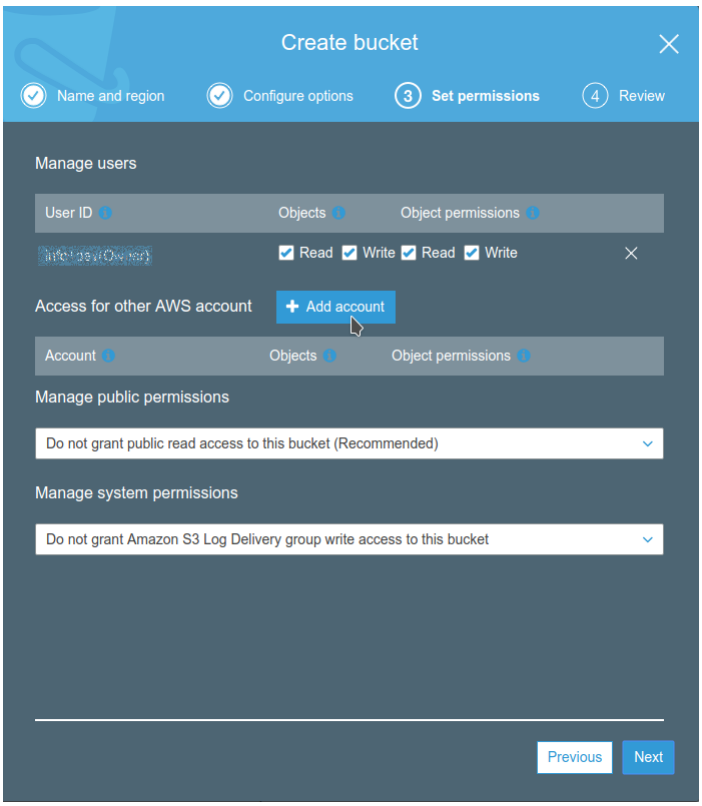
PREPARAZIONE DEL FRONT-END

Per prima cosa andiamo a creare un bucket S3 dove caricare l'applicazione javascript e html per il front-end.

Lo stesso bucket sarà la sorgente per la distribuzione di CloudFront.

Il bucket può essere privato, configureremo la distribuzione in modo da consentire a CloudFront l'accesso al bucket S3.





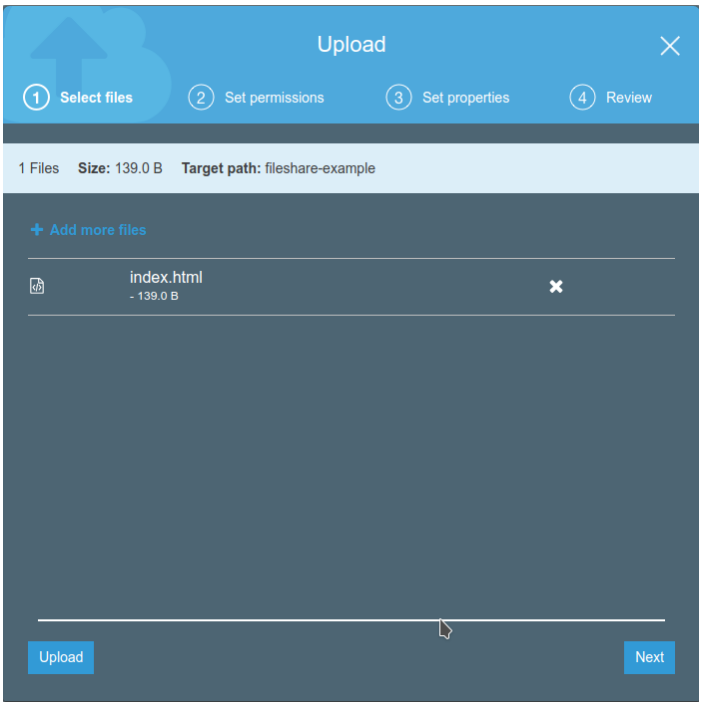
Dopo è possibile caricare un semplice file di prova, che useremo per verificare il corretto funzionamento del front-end.

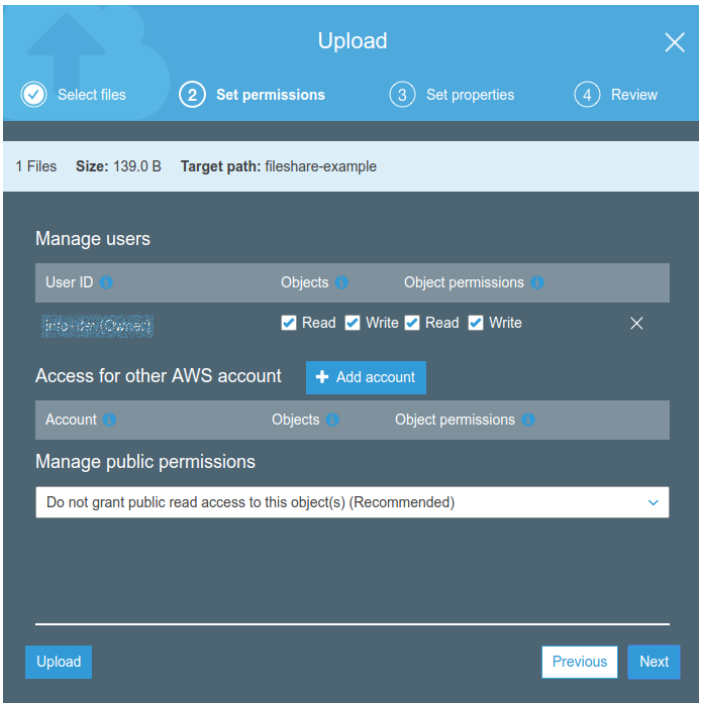
```
<html>
	<head>
		<title>Esempio</title>
	</head>

	<body>
```

```
            <h1>It Works</h1>
            <hr/>
            <p>This is a simple test page</p>
      </body>
</html>
```

Salviamo questo file come “**index.html**” e carichiamolo nel bucket appena creato.

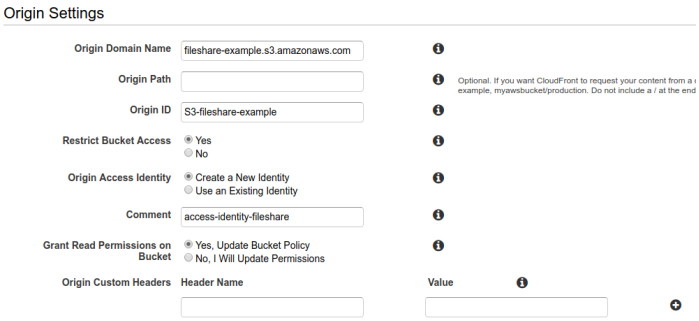




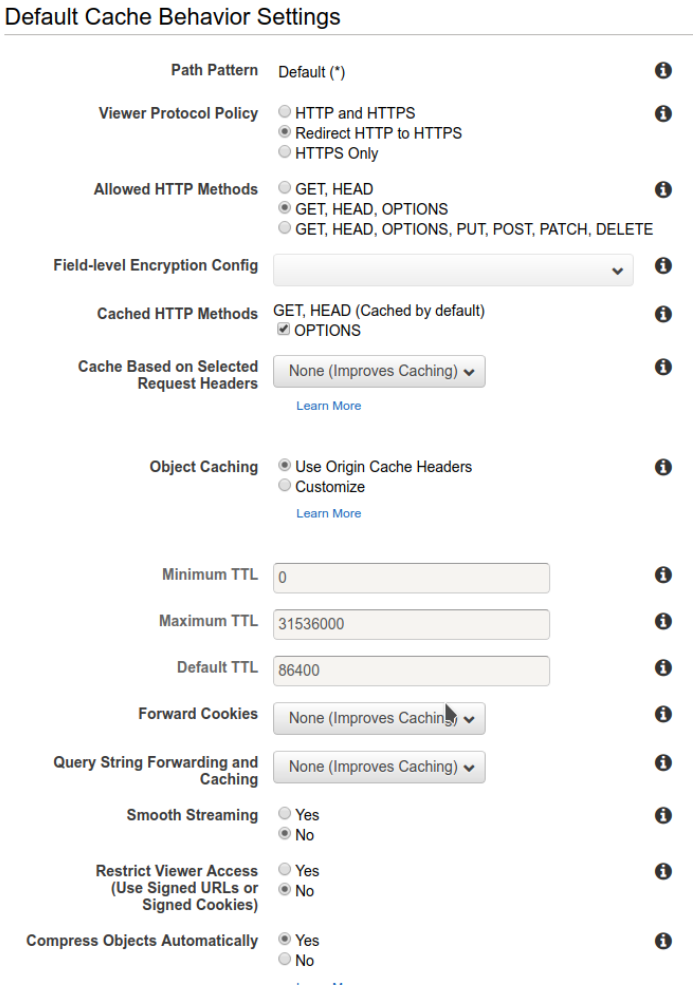
Una volta ultimate le operazioni per la creazione del bucket S3 occorre creare una **distribuzione CloudFront** che lo utilizzi come origine. Questo per distribuire il file che abbiamo caricato, che non è al momento accessibile.

Nella configurazione della distribuzione è importante abilitare l'opzione "**Restrict Bucket Access**" per mantenere il bucket S3 privato.

Al fine di consentire al servizio CloudFront di accedere al bucket in lettura bisogna provvedere ad attivare e configurare l'opzione "**Origin Access Identity**"; è possibile delegare al wizard la creazione di una Access Identity e l'aggiornamento della Bucket Policy.



Per quanto riguarda le impostazioni del comportamento di **caching** attiviamo il redirect di http su https e abilitiamo la compressione automatica dei contenuti serviti per migliorare le performance.

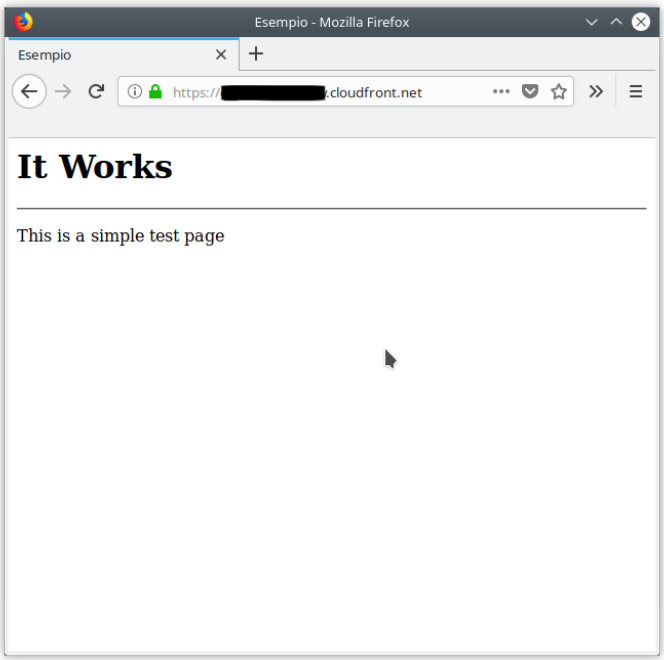


Non bisogna dimenticare di indicare quale file servire di default se non indicato, ed impostarlo ad index.html

Il processo di creazione della distribuzione può richiedere diversi minuti.
In alcuni casi, anche dopo l'attivazione della distribuzione potrebbe essere visualizzato un errore di accesso, tuttavia si tratta di una situazione transitoria dovuta ai tempi di propagazione DNS, e si risolve autonomamente entro qualche minuto. [Può richiedere fino ad un ora NDR]

Quando la distribuzione sarà pronta è possibile ottenere l'url pubblico e testare il deploy del front-end navigando a quell'indirizzo.

Dovrebbe essere visualizzata la pagina di esempio che abbiamo caricato nei passaggi precedenti.



Il front-end è quindi pronto. Il contenuto del bucket viene servito in modo efficiente e cost effective attraverso la CDN; per aggiornare l'applicazione basta caricarla su S3.

Attenzione! se si modificano file già presenti occorre invalidare la cache della distribuzione per vedere gli aggiornamenti.

PREPARAZIONE DEL BACK-END

Passiamo ora alla realizzazione dell'infrastruttura del back-end del nostro sistema di file sharing.

Come prima cosa possiamo **creare una funzione lambda** che utilizzeremo ai fini di test.

Come abbiamo già detto, non ci occuperemo adesso dell'applicazione vera e propria, ma solo di costruire tutte le parti dell'infrastruttura. Questa funzione non farà altro che restituire uno stato di successo ed un messaggio.

Procediamo quindi con la creazione della funzione impostando Python come Runtime e assegnando un ruolo con permessi di base.

Ecco un esempio per restituire sempre un messaggio:

```
def lambda_handler(event, context):

    response = {
        "statusCode": 200,
        "body": 'Hello from Lambda!'
    }

    return response
```

Testiamo la funzione, se tutto è stato impostato correttamente il test dovrebbe riuscire con un output simile a quello seguente:

{
“statusCode”: **200**,
“body”: “Hello from Lambda!”
}

Una volta pronta la funzione possiamo effettuare il **provisioning delle altre risorse**.

All’applicazione occorrerà un ulteriore bucket S3 per gli upload dei file degli utenti. Anche in questo caso si tratterà di un bucket privato. Provvediamo quindi alla sua creazione nello stesso modo utilizzato per il front-end, e prendiamo nota del nome del bucket, ci servirà per la configurazione dell’applicazione.

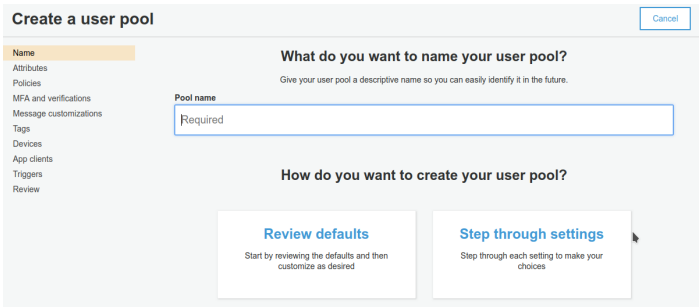
Servirà anche una **tabella di DynamoDB** per contenere i metadati e le informazioni di share dei file. Procediamo quindi alla creazione di una tabella lasciando i valori di default e specificando “shareID” come chiave di partizione primaria.

A questo punto dobbiamo occuparci di **Cognito** e di **API Gateway**. Il primo è necessario per la parte di autenticazione, ed il secondo permetterà al front-end di accedere alle funzionalità del back-end lambda mediante chiamate https.

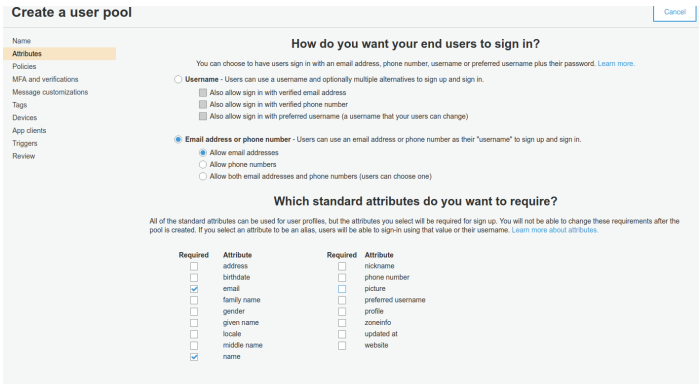
Per prima cosa creiamo e configuriamo una **Cognito User Pool**.

Durante il Wizard di creazione possiamo personalizzare alcuni comportamenti; per far funzionare l'applicazione bisogna creare un Pool che permetta gli utenti di registrarsi mediante email e di verificare l'indirizzo utilizzando un codice di sicurezza gestito da Cognito. Occorre anche creare un'app client e prendere nota del suo ID.

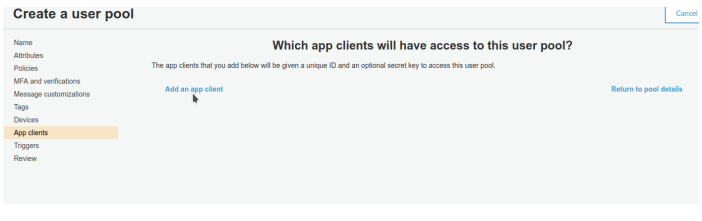
Scegliamo un nome per la User Pool e avviamo il wizard

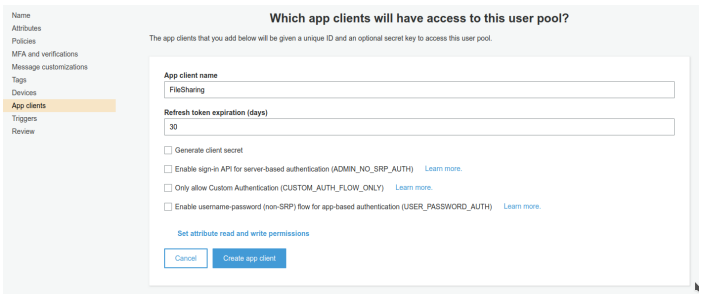


Configuriamo il meccanismo di autenticazione e i campi del profilo.



Infine bisogna creare un app client e prendere nota del suo ID. Non bisogna far generare alcun segreto, non servirà per l'utilizzo di Cognito mediante web app.



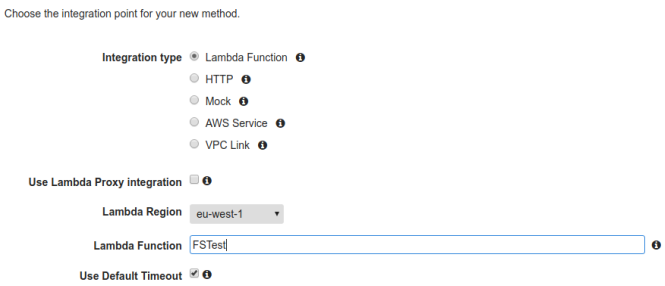


Con Cognito abbiamo finito per ora, queste risorse saranno utilizzate dall'applicazione e da API Gateway per il meccanismo di autenticazione.

Infine procediamo alla creazione di una API su API Gateway.

Questo componente fa da interfaccia tra le richieste dell'utente e la funzione lambda, inoltre si occuperà in modo automatico dell'autenticazione delle richieste mediante la user pool di Cognito.

Per scopo di test procediamo alla creazione di una API semplice con soltanto un metodo GET che chiama la funzione lambda.



Una volta configurata l'integrazione è possibile testarla

Nell'ultimo articolo della serie ci occuperemo della **parte applicativa** e della **pipeline di CD/CI**, e vedremo come **integrare Cognito in API Gateway** per beneficiare dell'autenticazione automatica.

Stay tuned! 🙂

Leggi la prima parte | Leggi la terza parte

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it