COME REALIZZARE UN SISTEMA DI PAGAMENTO TOTALMENTE SERVERLESS CON STRIPE E AWS LAMBDA

AWS Lambda | Serverless | Stripe Payment

beSharp | 21 Febbraio 2020

Nell'epoca dello shopping online, **l'implementazione di sistemi di pagamento all'interno di applicazioni cloud native**, sta diventando una necessità sempre più presente sul mercato.

Come facilmente possiamo intuire, gestire pagamenti all'interno del proprio flusso applicativo richiede l'implementazione di un infrastruttura affidabile, che sia in grado di garantire **privacy** e **consistenza alle nostre transazioni.**

Inoltre, l'integrazione verso i sempre più numerosi circuiti di pagamento comporta un notevole impegno in termini di sviluppo e manutenzione di tali applicativi.

Ovviamente, da appassionati del cloud quali siamo, non potevamo che trovare la soluzione in un servizio del tutto managed: parliamo di Stripe, un **middleware di pagamento** che espone ai suoi utenti dashboard di controllo e interfacce REST.

Sebbene Stripe sia un servizio fully managed, ogni procedura di pagamento ha le proprie caratteristiche che vengono definite dai requisiti di business.

È altamente consigliato scrivere codice lato server per mantenere le procedure di pagamento riservate ed evitare così il diffondersi di dati sensibili a malintenzionati

Stripe è un servizio scalabile e completamente gestito. Per ottenere un'applicazione il più performante possibile, occorre fare in modo che il back-end sia in grado di sostenere l'agilità della nuova soluzione. Per fare ciò, le tecnologie serverless sono quelle che meglio vengono in nostro aiuto

In questo articolo andremo a sfruttare la potenza di AWS Lambda, il servizio di serverless computing offerto da Amazon Web Service.

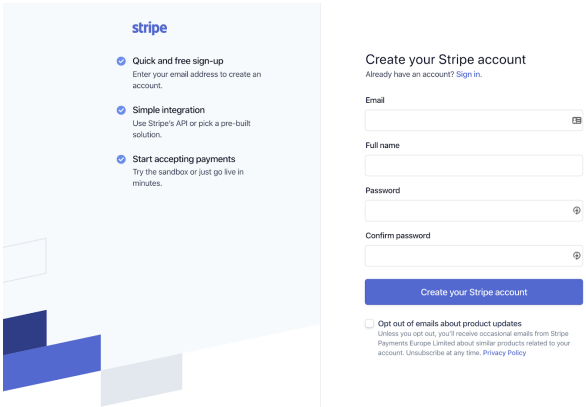
Per i neofiti del settore, si tratta di funzioni applicative stateless. Allo sviluppatore viene offerto un ambiente nel quale poter scrivere codice senza doversi preoccupare dell'hardware utilizzato e della scalabilità della soluzione. In questo caso il pagamento è a consumo.

Stripe

Vediamo ora nel dettaglio come utilizzare Stripe.

Per prima cosa, registriamoci al servizio da qui. tramite questo link:

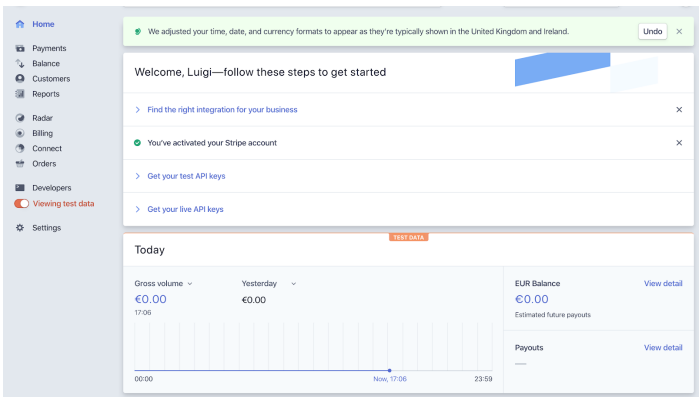
La registrazione è totalmente gratuita e il pagamento è a consumo. Trovate tutti i dettagli sul pricing in questa pagina.



In questo articolo non ci dilungheremo a raccontare le molteplici funzionalità di Stripe, che potrete trovare comodamente nella documentazione ufficiale.

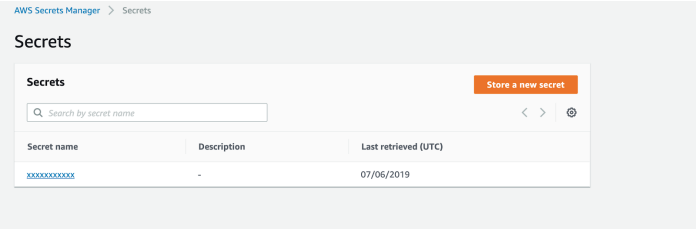
Ci focalizzeremo invece sul come integrare un applicativo serverless basato su Lambda alle API del nostro account Stripe appena creato.

Una delle componenti fondamentali di Stripe è senza dubbio la dashboard che offre agli utenti la possibilità di creare e gestire risorse come sottoscrizioni e prodotti.

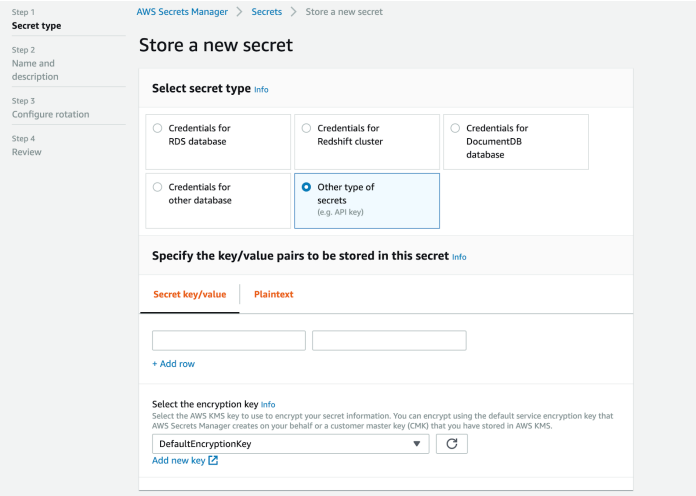


Come potete vedere potremo decidere se ottenere una **test API key pair** o una **live API key pair**: con la prima, saranno tutti dati di test che potremo vedere solo se la checkbox sulla sinistra *"Viewing test data"* è abilitata. Al contrario la live API key la si usa per transazioni reali: saremo quindi sull'ambiente di produzione.

Inutile dire che entrambe le chiavi dovranno essere conservate in un ambiente sicuro. Per fare ciò, un ottimo servizio offerto da Amazon Web Services è AWS Secret Manager, un database chiave valore pensato appositamente per lo storage di credenziali, chiavi di accesso e altre informazioni che si possano considerate altamente sensibili.



Per salvare una nuova informazione su AWS Secret Manager basterà cliccare su *"Store a new secret"* e selezionare la tipologia di segreto



In questo caso avremo bisogno solo di dati chiave valore senza alcun tipo di integrazione.

E adesso si inizia con le Lambda!

Finalmente è arrivato il momento di **creare la nostra Lambda** per i pagamenti online.

Nell'esempio useremo **Python3.6**

NOTA: Assicurati che attaccato alla Lambda ci sia un LambdaLayer contenete i pacchetti pip, stripe e boto3.

L'algoritmo implementato nel codice sottostante prevede l'abbonamento da parte di un utente ad una sottoscrizione precedentemente creata dalla dashboard di Stripe, quindi per usare questo codice avrete bisogno dell'ARN del secret manager creato in precedenza e del plan id.

```
import stripe
import boto3
import json

client = boto3.client('secretsmanager')
keys = json.loads(client.get_secret_value(
  SecretId = 'arn:aws:secretsmanager:eu-central-1:169954988972:secret:stripe-AiFSZg',
)['SecretString'])
public_key = keys['stripe-public']
secret_key = keys['stripe-secret']
stripe.api_key = secret_key

def signup(event, context):
  card_data = event.get('cardData')
  email = event.get('email')
  attributes = event.get('attributes')
  create_stripe_customer(email, attributes, card_data)
  return event

def create_stripe_customer(email, user_data, payment_info):
  customer_id = stripe.Customer.create(email = email, metadata = user_data)['id']
  payment_method_id = create_payment_method(payment_info)
  stripe.PaymentMethod.attach(
    payment_method_id,
    customer = customer_id
  )
  return {
    "customer_id": customer_id,
    "plan": create_stripe_plan(customer_id)
  }

def create_payment_method(payment_info):
  return stripe.PaymentMethod.create(
    type = "card",
    card = {
      "number": payment_info.get('cardNumber'),
      "exp_month": payment_info.get('expirationMonth'),
      "exp_year": payment_info.get('expirationYear'),
      "cvc": payment_info.get('ccv'),
    }).get('id')

def create_stripe_plan(customer_id):
  return stripe.Subscription.create(
    customer = customer_id,
    items = [{
      "plan": "plan_idxxxxxxx"
    }]
  ).get("id")
```

Come potete vedere questa Lambda prende i valori dal payload passato all'interno dell'event che cambierà a seconda del servizio integrato alla funzione.

Per usare la Lambda abbiamo diverse possibilità: è possibile ad esempio utilizzarla come trigger di una coda SQS, come resource a un API Gateway o fare invoke direttamente dai client.

Complimenti! Se avete seguito questi brevi passi avete costruito il vostro sistema di pagamento totalmente serverless! A questo punto siete pronti a gestire potenzialmente milioni di utenti con costi infrastrutturali direttamente proporzionale al traffico. Non vi resta che provarlo in produzione!

Se volete sapere di più su Stripe, su AWS Lambda o su altri possibili scenari di applicazione contattateci!

Arrivederci al prossimo articolo!

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