# **AWS LAMBDA CHUẨN HÓA VIỆC THANH TOÁN CHO GIAI ĐOẠN INIT**

bởi Shubham Gupta và Jeff Gebhart vào29 THÁNG 4 NĂM 2025 trong [Thông báo](https://aws.amazon.com/blogs/compute/category/post-types/announcements/) , [AWS Lambda](https://aws.amazon.com/blogs/compute/category/compute/aws-lambda/) , [Trung cấp (200)](https://aws.amazon.com/blogs/compute/category/learning-levels/intermediate-200/) , [Không có](https://aws.amazon.com/blogs/compute/category/serverless/) [máy chủ Liên kết cố định](https://aws.amazon.com/blogs/compute/aws-lambda-standardizes-billing-for-init-phase/) [Chia sẻ](https://aws.amazon.com/blogs/compute/aws-lambda-standardizes-billing-for-init-phase/)

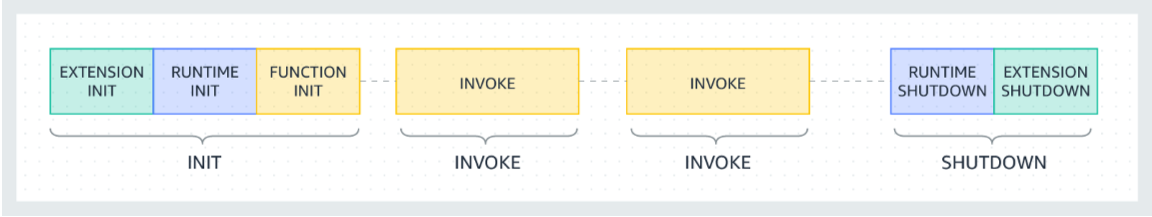
Có hiệu lực từ ngày 1 tháng 8 năm 2025, AWS sẽ chuẩn hóa việc thanh toán cho giai đoạn khởi tạo (INIT) trên tất cả các cấu hình hàm [AWS Lambda](https://aws.amazon.com/lambda/) . Thay đổi này đặc biệt ảnh hưởng đến các lệnh gọi hàm Lambda theo yêu cầu được đóng gói dưới dạng tệp ZIP sử dụng thời gian chạy được quản lý, trong đó thời lượng giai đoạn INIT trước đó không được thanh toán. Bản cập nhật này chuẩn hóa việc thanh toán cho giai đoạn INIT trên tất cả các loại thời gian chạy, gói triển khai và chế độ gọi. Hầu hết người dùng sẽ thấy tác động tối thiểu đến tổng hóa đơn Lambda của họ từ thay đổi này, vì giai đoạn INIT thường xảy ra đối với một phần rất nhỏ các lệnh gọi hàm. Trong bài đăng này, chúng tôi thảo luận về Vòng đời hàm Lambda và những thay đổi sắp tới đối với việc thanh toán giai đoạn INIT. Bạn sẽ tìm hiểu những gì xảy ra trong giai đoạn INIT và khi nào nó xảy ra, cách theo dõi thời lượng giai đoạn INIT của bạn và các chiến lược để tối ưu hóa giai đoạn này và giảm thiểu chi phí.

## **Hiểu về vòng đời thực thi hàm Lambda**

Vòng [đời thực thi hàm Lambda](https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtime-environment.html" \l "runtimes-lifecycle) bao gồm ba giai đoạn riêng biệt: INIT, INVOKE và SHUTDOWN. Giai đoạn INIT được kích hoạt trong quá trình “khởi động nguội” khi Lambda tạo môi trường thực thi mới cho một hàm để phản hồi lệnh gọi. Tiếp theo là giai đoạn INVOKE, trong đó yêu cầu được xử lý và cuối cùng là giai đoạn SHUTDOWN, trong đó môi trường thực thi bị chấm dứt. Để biết tóm tắt về vòng đời thực thi, hãy xem [Vòng đời môi trường thực thi AWS Lambda](https://www.youtube.com/watch?v=E20B8Izr5fI) .

Trong giai đoạn INIT, Lambda thực hiện một loạt các bước chuẩn bị trong thời gian tối đa là 10 giây. Dịch vụ này lấy mã chức năng từ một thùng chứa [Amazon S3](https://aws.amazon.com/s3/) nội bộ hoặc từ [Amazon Elastic Container Registry](https://aws.amazon.com/ecr/) (Amazon ECR) cho các chức năng sử dụng đóng gói container. Sau đó, nó cấu hình một môi trường với bộ nhớ, thời gian chạy và các thiết lập khác đã chỉ định. Khi môi trường thực thi được chuẩn bị, Lambda thực hiện bốn tác vụ chính theo trình tự:

1. Khởi tạo bất kỳ tiện ích mở rộng nào được cấu hình (Extension INIT)
2. Khởi động thời gian chạy (Runtime INIT)
3. Thực thi mã tĩnh của hàm (Hàm INIT)
4. Chạy bất kỳ hook thời gian chạy trước điểm kiểm tra nào (chỉ áp dụng cho Lambda SnapStart)



## **Hiểu về những thay đổi trong thanh toán**

Phí Lambda dựa trên số lượng yêu cầu và thời gian cần thiết để chạy mã. Thời gian được tính từ thời điểm [mã hàm bắt đầu chạy cho đến khi hoàn thành hoặc kết thúc](https://aws.amazon.com/blogs/compute/operating-lambda-performance-optimization-part-1/) , làm tròn đến mili giây gần nhất. Chi phí thời gian phụ thuộc vào lượng bộ nhớ mà bạn phân bổ cho hàm của mình.

Trước đây, thời lượng giai đoạn INIT không được bao gồm trong Thời lượng tính phí cho các chức năng sử dụng thời gian chạy được quản lý với gói lưu trữ ZIP, như được chứng minh trong nhật ký [Amazon CloudWatch](https://aws.amazon.com/cloudwatch/) :

REPORT RequestId: xxxxx   Duration: 250.06 ms  Billed Duration: 251 ms  Memory Size: 1024 MB

Max Memory Used: 350 MB   Init Duration: 100.77 ms

Tuy nhiên, các hàm được cấu hình với [thời gian chạy tùy chỉnh](https://docs.aws.amazon.com/lambda/latest/dg/runtimes-custom.html) , [Đồng thời được cung cấp (PC)](https://docs.aws.amazon.com/lambda/latest/dg/provisioned-concurrency.html) hoặc [đóng gói OCI](https://docs.aws.amazon.com/lambda/latest/dg/images-create.html) đã bao gồm thời lượng giai đoạn INIT trong Thời lượng được tính phí của chúng. Có hiệu lực từ ngày 1 tháng 8 năm 2025, giai đoạn INIT sẽ được tính phí trên tất cả các loại cấu hình và thời lượng giai đoạn INIT sẽ được bao gồm trong Thời lượng được tính phí cho các lệnh gọi hàm theo yêu cầu sử dụng thời gian chạy được quản lý với cả đóng gói lưu trữ ZIP. Sau thay đổi này, dòng nhật ký ID yêu cầu BÁO CÁO sẽ hiển thị như sau:

REPORT RequestId: xxxxx   Duration: 250.06 ms  Billed Duration: 351 ms  Memory Size: 1024 MB

Max Memory Used: 350 MB   Init Duration: 100.77 ms

Các khoản phí thời lượng pha INIT tiếp theo sẽ tuân theo giá thời lượng theo yêu cầu tiêu chuẩn dành riêng cho từng [Vùng AWS](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/) , có thể tìm thấy trên [trang giá Lambda](https://aws.amazon.com/lambda/pricing/) . Đối với các chức năng [AWS Lambda@Edge](https://aws.amazon.com/lambda/edge/) , thời lượng pha INIT sẽ được tính theo giá thời lượng Lambda@Edge.

## **Tìm thời lượng giai đoạn INIT và tác động đến thanh toán Lambda**

Bạn đã có thể theo dõi thời gian dành cho giai đoạn INIT của các lệnh gọi hàm của mình bằng cách sử dụng số liệu CloudWatch “init\_duration”. Số liệu này cũng được báo cáo là “Init Duration” trong dòng nhật ký “REPORT RequestId” trong CloudWatch Logs. Các công cụ này cung cấp thông tin chi tiết có giá trị về thời gian INIT của các hàm Lambda, giờ đây sẽ được đưa vào tính toán hóa đơn.

Để có phân tích toàn diện hơn, bạn có thể sử dụng truy vấn CloudWatch Log Insights sau đây để tạo báo cáo chi tiết ước tính thời lượng chưa thanh toán trước đó của giai đoạn INIT. Truy vấn giúp bạn hiểu tỷ lệ thời gian của giai đoạn INIT chưa thanh toán so với mức sử dụng Lambda tổng thể của bạn, cho phép dự báo chi phí chính xác hơn sau khi thay đổi thanh toán này.

filter @type = "REPORT" and @billedDuration < (@duration + @initDuration)

| stats sum((@memorySize/1000000/1024) \* (@billedDuration/1000)) as BilledGBs,

sum((@memorySize/1000000/1024) \* ((ceil(@duration + @initDuration) - @billedDuration)/1000)) as UnbilledInitGBs,

(UnbilledInitGBs/ (UnbilledInitGBs+BilledGBs)) as Ratio

Truy vấn CloudWatch Log Insights cung cấp ba số liệu cần thiết:

1. **BilledGBs** : Biểu thị tổng số GB-s (gigabyte-giây) hiện đang được tính phí cho các nhóm nhật ký đã chọn.
2. **UnbilledInitGBs** : Hiển thị tổng số GB được sử dụng trong giai đoạn INIT mà trước đó không được bao gồm trong quá trình thanh toán.
3. **Ratio** : Chỉ ra phần trăm tổng số GB được phân bổ cho thời lượng giai đoạn INIT chưa được lập hóa đơn trước đó.



Việc sử dụng các khả năng giám sát hiện có này cho phép bạn chủ động đánh giá và tối ưu hóa thời gian INIT của chức năng Lambda, từ đó giảm thiểu tác động của cấu trúc thanh toán mới đến tổng chi phí của bạn.

## **Hiểu và tối ưu hóa giai đoạn INIT của Lambda**

Giai đoạn INIT của Lambda được kích hoạt trong hai tình huống cụ thể: trong quá trình tạo môi trường thực thi mới và khi một hàm mở rộng quy mô để đáp ứng nhu cầu. Mã INIT này chỉ chạy trong những lần "khởi động nguội" này và bị bỏ qua trong các lần gọi tiếp theo sử dụng môi trường ấm hiện có. Sau giai đoạn INIT, Lambda chạy mã trình xử lý hàm để xử lý lệnh gọi.

Sau khi thực thi trình xử lý, Lambda đóng băng môi trường thực thi. Để cải thiện hiệu suất và quản lý tài nguyên, dịch vụ Lambda giữ lại môi trường thực thi trong một khoảng thời gian không xác định. Trong thời gian này, nếu có yêu cầu khác đến cho cùng một chức năng, thì dịch vụ có thể sử dụng lại môi trường. Yêu cầu thứ hai này thường kết thúc nhanh hơn, vì môi trường thực thi đã tồn tại và không cần phải tải xuống mã và chạy mã INIT. Đây được gọi là "khởi động ấm".

Các nhà phát triển có thể sử dụng giai đoạn INIT để tạo, khởi tạo và cấu hình các đối tượng dự kiến ​​sẽ được sử dụng lại qua nhiều lần gọi trong hàm INIT thay vì thực hiện trong trình xử lý. Khởi tạo các đối tượng phụ thuộc/chia sẻ trước sẽ giảm độ trễ của các lần gọi tiếp theo. Ví dụ:

* Tải xuống thêm thư viện hoặc phụ thuộc
* Thiết lập kết nối máy khách với các dịch vụ AWS khác như Amazon S3 hoặc [Amazon DynamoDB](https://aws.amazon.com/dynamodb/)
* Tạo kết nối cơ sở dữ liệu để chia sẻ qua các lệnh gọi
* Truy xuất các tham số hoặc bí mật ứng dụng từ [Amazon Systems Manager](https://aws.amazon.com/systems-manager/) Parameter Store hoặc [AWS Secrets Manager](https://aws.amazon.com/secrets-manager/)

Khi phát triển các hàm Lambda, điều quan trọng là phải quyết định chiến lược xem mã nào sẽ chạy trong giai đoạn INIT thay vì giai đoạn xử lý, vì điều này ảnh hưởng đến cả hiệu suất và chi phí.

### **Tối ưu hóa kích thước gói/thư viện**

Giai đoạn INIT bao gồm việc tạo môi trường thực thi, tải xuống mã chức năng và khởi tạo nó. Ba yếu tố chính ảnh hưởng đến hiệu suất của nó:

1. Kích thước của gói chức năng, xét về các thư viện và phụ thuộc được nhập, cũng như các lớp Lambda.
2. Lượng mã và công việc INIT.
3. Hiệu suất của thư viện và các dịch vụ khác trong việc thiết lập kết nối và các tài nguyên khác.

Các gói chức năng lớn hơn làm tăng thời gian tải xuống mã. Bạn có thể giảm thời lượng giai đoạn INIT bằng cách giảm kích thước gói, dẫn đến khởi động nguội nhanh hơn và chi phí INIT thấp hơn. Hơn nữa, việc tối ưu hóa việc tải các thư viện cũng có thể tác động đáng kể đến kích thước gói. Các công cụ như [esbuild](https://esbuild.github.io/) có thể tối ưu hóa hiệu suất hơn nữa bằng cách thu nhỏ và đóng gói các gói. Để biết chi tiết, hãy đọc [Giảm thời gian khởi động nguội Lambda: di chuyển sang AWS SDK cho JavaScript v3](https://aws.amazon.com/blogs/developer/reduce-lambda-cold-start-times-migrate-to-aws-sdk-for-javascript-v3/) .

### **Tối ưu hóa việc thực hiện giai đoạn INIT và hiệu quả chi phí**

Tần suất thực thi giai đoạn INIT (hoặc khởi động nguội) tác động trực tiếp đến cả hiệu suất và hiệu quả chi phí. Theo phân tích về khối lượng công việc Lambda sản xuất, INIT (khởi động nguội) thường xảy ra ở dưới 1% số lần gọi—có nghĩa là mã trong giai đoạn INIT có thể chỉ thực thi một lần trên một trăm lần gọi.

Bạn có thể sử dụng giai đoạn INIT để thực hiện các hoạt động một lần có lợi cho các lần gọi tiếp theo. Các mẫu tối ưu hóa phổ biến bao gồm tính toán trước các bảng tra cứu hoặc chuyển đổi các tập dữ liệu tĩnh. Ví dụ: tải xuống dữ liệu tĩnh từ Amazon S3 hoặc DynamoDB trong INIT, giúp dữ liệu này khả dụng cho tất cả các lần gọi hàm tiếp theo mà không cần tải xuống nhiều lần.

### **Lambda SnapStart**

[Lambda SnapStart](https://docs.aws.amazon.com/lambda/latest/dg/snapstart.html) cung cấp giải pháp hiệu quả để giảm độ trễ khởi động nguội và chi phí giai đoạn INIT. Khi được bật, SnapStart sẽ tạo một ảnh chụp nhanh trong hàm INIT đầu tiên và sử dụng lại cho các lần khởi động nguội tiếp theo, loại bỏ nhu cầu thực thi giai đoạn INIT nhiều lần. Phương pháp này đặc biệt có giá trị đối với các hàm có thời gian INIT dài hơn do tải các phụ thuộc/khung mô-đun, khởi tạo thời gian chạy hoặc thực thi mã INIT một lần. SnapStart được hỗ trợ cho thời gian chạy Java, .NET và Python. Bạn có thể triển khai SnapStart thông qua bảng điều khiển Lambda hoặc [Giao diện dòng lệnh AWS (AWS CLI),](https://aws.amazon.com/cli/) đảm bảo rằng mã của bạn tuân thủ [các nguyên tắc tuần tự hóa AWS](https://docs.aws.amazon.com/lambda/latest/dg/snapstart-uniqueness.html) để tương thích với khả năng khôi phục ảnh chụp nhanh. Sử dụng SnapStart cho phép bạn cải thiện đáng kể thời gian khởi động hàm và tối ưu hóa chi phí trên nhiều ngôn ngữ lập trình phổ biến.

### **Đồng thời được cung cấp**

Provisioned Concurrency là một tính năng Lambda khởi tạo trước môi trường thực thi trước khi bất kỳ lệnh gọi nào xảy ra. Cách tiếp cận chủ động này loại bỏ hiệu quả tác động hiệu suất của giai đoạn INIT đối với các lệnh gọi hàm riêng lẻ, vì INIT được hoàn thành trước.

Mặc dù tất cả các chức năng sử dụng Provisioned Concurrency đều được hưởng lợi từ thời gian khởi động ngắn hơn so với thực thi theo yêu cầu, nhưng tác động này đặc biệt rõ rệt đối với một số môi trường thời gian chạy nhất định. Ví dụ: các chức năng C# và Java—thường có INIT chậm hơn nhưng thời gian thực thi nhanh hơn so với Node.js hoặc Python—có thể đạt được mức tăng hiệu suất đáng kể thông qua tính năng này. Việc triển khai Provisioned Concurrency cho phép bạn quản lý hiệu quả cả các mẫu lưu lượng truy cập nhất quán và các mức sử dụng đột biến dự kiến, do đó giảm thiểu độ trễ khởi động nguội trên các ứng dụng không có máy chủ của bạn. Chiến lược tối ưu hóa này đặc biệt có giá trị đối với các chức năng có yêu cầu INIT phức tạp hoặc các chức năng phục vụ khối lượng công việc nhạy cảm với độ trễ. Theo quan điểm tối ưu hóa chi phí, Provisioned Concurrency phù hợp nhất với các khối lượng công việc có mẫu sử dụng liên tục trên 60% mức sử dụng, vì điều này thường mang lại hiệu quả chi phí tốt hơn so với thực thi theo yêu cầu.

## **Phần kết luận**

Có hiệu lực từ ngày 1 tháng 8 năm 2025, AWS sẽ chuẩn hóa việc thanh toán giai đoạn INIT cho AWS Lambda. AWS cung cấp nhiều cách để bạn tối ưu hóa cả hiệu suất và chi phí của các hàm Lambda. Cho dù bạn đang sử dụng SnapStart, triển khai Provisioned Concurrency hay tối ưu hóa mã INIT, chúng tôi khuyên bạn nên làm việc chặt chẽ với nhóm hỗ trợ AWS để xác định phương pháp tối ưu hóa phù hợp nhất cho các yêu cầu khối lượng công việc cụ thể của bạn.

Để được hỗ trợ và hướng dẫn thêm, hãy cân nhắc tham gia [hội thảo Tối ưu hóa chi phí AWS](https://workshops.aws/categories/Cost Optimization) hoặc tham khảo [tài liệu Lambda](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) .