TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ**

**MÔN KIỂM THỬ PHẦM MỀM**

**TÌM HIỂU LOADSTER VÀ THỰC HIỆN KIỂM THỬ WEBSITE TOÀN DIỆN VỚI LOADTER**

*Người hướng dẫn*: **Thầy DOÃN XUÂN THANH**

*Người thực hiện*: **TRẦN ANH KHOA – 51900706**

**NGUYỄN TRẦN BẢO VY - 51900289**

**NGUYỄN MỸ ANH - 51900005**

Khoá  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ**

**MÔN CHUYÊN ĐỀ CÔNG NGHỆ PHẦN MỀM**

**HỆ THỐNG QUẢN LÝ CỬA HÀNG BÁN HOA**

*Người hướng dẫn*: **Thầy DOÃN XUÂN THANH**

*Người thực hiện*: **TRẦN ANH KHOA – 51900706**

**NGUYỄN TRẦN BẢO VY - 51900289**

**NGUYỄN MỸ ANH - 51900005**

Khoá  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

# GIỚI THIỆU

Tài liệu này giải thích về Loadster để thực hiện Load Testing và cung cấp chi tiết về các bước để thực hiện Load Testing. Tài liệu này được chuẩn bị sau khi thành công thực hiện Load Testing trong một trong các đồ án của chúng tôi. Về cơ chúng ta, chung tôi cung cấp ý tưởng về cách thực hiện Load Testing được coi là một phương pháp để xác định cách một ứng dụng sẽ hoạt động dưới khi tải. "Tải", nói chung, đề cập đến tổng lưu lượng người dùng tại một thời điểm nhất định. Tài liệu này cũng giải thích cách sử dụng Loadster để có kiến ​​thức về một số các chỉ số khác nhau như stress, stability, spike, scalability, baseline.

Loadster là một môi trường tích hợp cho phép chúng ta thực hiện Load Testing. Load Testing có thể được thực hiện bằng cách sử dụng bất kỳ số lượng người dùng ảo nào. Nó cung cấp cho chúng tôi thông tin ghi lại và chỉnh sửa các tập lệnh cũng như kết thu được sau khi thực hiện kiểm tra. Loadster có in-built load qua đó chúng tôi có thể thực hiện Load Testing với những người dùng ảo đồng thời.

# MỤC LỤC

Contents

[GIỚI THIỆU 2](#_Toc105277321)

[MỤC LỤC 3](#_Toc105277322)

[1. Tổng quan về công cụ kiểm thử phần mềm tự động Loadster và Load Testing 4](#_Toc105277323)

[1.1 Load testing 4](#_Toc105277324)

[1.1.1 Khái niệm 4](#_Toc105277325)

[1.1.2 Các trường hợp nên sử dụng Load Testing: 4](#_Toc105277326)

[1.1.3 Các thông số cơ bản trong Load Testing: 4](#_Toc105277327)

[1.1.4 Tầm quan trọng của Load Testing 5](#_Toc105277328)

[1.1.5 Ưu và nhược điểm của Load Testing 7](#_Toc105277329)

[1.2 Loadster 8](#_Toc105277330)

[1.2.1 Giới thiệu chung 8](#_Toc105277331)

[1.2.2 Một số thuật ngữ trong Loadster 11](#_Toc105277332)

[1.2.3 Quy trình thiết kế kịch bản thực tế kiểm thử chịu tải 14](#_Toc105277333)

[1.2.4 Quy trình thực hiện Load Testing với Loadster 19](#_Toc105277334)

[1.2.5 Các công cụ của Loadster 21](#_Toc105277335)

[1.2.6 Đối tác của Loadster 23](#_Toc105277336)

[2. Cách sử dụng LoadStorm 23](#_Toc105277337)

[3. Tổng kết 25](#_Toc105277338)

# 1. Tổng quan về công cụ kiểm thử phần mềm tự động Loadster và Load Testing

## 1.1 Load testing

### 1.1.1 Khái niệm

Mô phỏng độ chịu tải của hệ thống. Thường là khi hệ thống hoạt động bình thường và khi có tải hiệu suất cao. Nhằm đánh giá sự ổn định khi sản phẩm chạy trong môi trường thực tế. Mục tiêu đánh giá khả năng đáp ứng hệ thống khi hoạt động bình thường và khi hoạt động tại giờ cao điểm .

Các thuộc tính được kiểm tra trong một bài Load Testing là hiệu suất hoạt động lúc cao điểm, số lượng máy chủ và thời gian phản hồi.Trong Load Test giới hạn tải là ngưỡng ngắt. Nó được thực hiện vào giai đoạn cuối của một dự án phát triển phần mềm.

### 1.1.2 Các trường hợp nên sử dụng Load Testing:

* Tải xuống hàng loạt file lớn từ internet.
* Chạy đồng thời nhiều ứng dụng trên máy tính hoặc máy chủ.
* Để nhiều công việc cho một máy in trong một hàng đợi.
* Đang chịu một máy chủ lưu lượng truy cập lớn.
* Viết và đọc dữ liệu đến và đi từ đĩa cứng liên tục.

### 1.1.3 Các thông số cơ bản trong Load Testing:

* **Average Response Time:** Thời gian phản hồi trung bình được tính từ khi bắt đầu mọi yêu cầu và mọi phản ứng đối với yêu cầu đó, cung cấp thông tin về một kỳ nghỉ tròn từ trình duyệt của chúng ta và thường là thời gian cần thiết để nhận được phản hồi. Thời gian trả lời trung bình đạt được bằng cách tính toán ngụ ý của hầu hết các yêu cầu / chu kỳ phản hồi.Thông thường, giá trị thu được cho thấy tốc độ từ chương trình web đang được kiểm tra, i. tại. trang web đang hoạt động tốt như thế nào thông qua quan điểm của người dùng.
* **Peak Response Time:** Thời gian phản hồi cao điểm đo thời gian của một chu kỳ yêu cầu / phản hồi. Đỉnh của nó cho biết khoảng thời gian mà chu kỳ là dài nhất. Thời gian phản hồi cao điểm cho biết một trong những tài nguyên nào có khả năng gây ra một số vấn đề và có thể phát sinh do các bất thường khác nhau trong ứng dụng web.
* **Error Rate**: Khi một ứng dụng đang tải, nó có thể cung cấp một số lỗi dự kiến ​​trong khi xử lý các yêu cầu. Tỷ lệ lỗi được tính theo tỷ lệ phần trăm các vấn đề được yêu cầu so với tất cả các yêu cầu. Những lỗi này thường gặp nhất khi ứng dụng đạt đến điểm mà sau đó ứng dụng web không thể xử lý các yêu cầu nữa. Tỷ lệ lỗi rất quan trọng trong việc đo lường hiệu quả của một ứng dụng, vì nó đo lường “lỗi hiệu suất” của một ứng dụng.
* **Concurrent Users**: Concurrent Users là một phương pháp điển hình để liệt kê tải đang được áp dụng trong suốt quá trình kiểm thử. Chỉ số này đo lường số lượng người dùng ảo đang hoạt động tại bất kỳ thời điểm nào. Tuy nhiên, nó không tương đương với RPS vì người dùng có thể tạo một số lượng yêu cầu cao, trong khi người dùng ảo sẽ không tạo ra các yêu cầu nhất quán.
* **Requests per Second**: RPS là phép đo liên quan đến số lượng yêu cầu được gửi đến máy chủ đích. Điều này bao gồm các yêu cầu về tệp dữ liệu Flash / đa phương tiện, biểu định kiểu Trang Web, hình ảnh, thư viện JavaScript, tài liệu XML, trang HTML và bất kỳ tài nguyên bắt buộc nào khác.
* **Throughput**: Thông lượng được đo bằng đơn vị kilobyte trên giây. Nó định lượng băng thông được tiêu thụ trong quá trình kiểm tra và đo lượng dữ liệu truyền qua lại giữa các máy chủ.

### 1.1.4 Tầm quan trọng của Load Testing

Thuê người kiểm tra để thực hiện Load Testing có thể phải chịu thêm chi phí nhưng chi phí sửa chữa một trang web bị sập trong thời gian thực thậm chí còn đắt hơn. Load Testing giúp tìm ra lỗi mà không thể tìm ra với bất kỳ phương pháp kiểm thử khác. Chẳng hạn như rò rỉ bộ nhớ, lỗi quản lý bộ nhớ, tràn bộ đệm, v.v ....Vì vậy nên thực hiện Load Testing để tiết kiệm chi phí và gián tiếp tạo doanh thu.

Ví dụ

* Target.com đã mất 780.000 đô la doanh số chỉ trong 3 giờ khi trang web ngừng hoạt động trong một chương trình khuyến mãi năm 2015
* Khi máy chủ Amazon.com gặp sự cố vào năm 2013 trong 30 phút, Amazon đã mất 66.240 đô la mỗi phút
* Vào tháng 9 năm 2010, các hãng hàng không Virgin Blue đã trải qua một loạt sự cố ngừng hoạt động trong 11 ngày dẫn đến việc đăng ký, đặt vé trực tuyến, đặt chỗ và hệ thống lên máy bay bị ngừng hoạt động. Điều này dẫn đến việc mất 20 triệu đô la ngoài ra hàng ngàn khách hàng thất vọng. Công ty Navitaire, công ty quản lý đặt chỗ cho Virgin Blue đã phải bồi thường cho Virgin Blue với giá 20 triệu đô la.
* Theo một khảo sát, 75% người dùng nói rằng nếu một trang web bị sập hoặc nếu nó chậm, họ sẽ rời khỏi trang web
* 50% người dùng nói rằng họ sẽ mua sắm ở nơi khác nếu trang web hoặc ứng dụng không tải trong 3 giây
* Như chúng ta có thể thấy, nhiều người có xu hướng mua sản phẩm trong một sự kiện quảng cáo như Black Friday, Cyber Monday, Diwali hoặc khi có khuyến mại / giảm giá, v.v.
* Nhiều người có xu hướng đặt vé máy bay trong các ngày lễ hoặc vào những ngày mà một hãng hàng không có lời đề nghị
* Nếu trang web của chúng ta gặp sự cố trong một sự kiện như vậy, khách truy cập có thể rời khỏi trang web của chúng ta và truy cập trang web của đối thủ cạnh tranh. Điều này có thể dẫn đến mất doanh thu và thị phần.
* Những tình huống như vậy có thể được ngăn chặn bằng cách tải kiểm thử hệ thống và khắc phục mọi sự cố được phát hiện

Bằng cách kiểm tra lưu lượng truy cập lớn trước thời hạn, chúng ta có thể tìm hiểu xem trang web của mình sẽ phản ứng như thế nào khi lượng người dùng thực truy cập. Tốt hơn nữa, chúng ta có thể sử dụng mô phỏng thực tế của mình để tối ưu hóa và cải thiện trang web của mình, để nó xử lý lưu lượng truy cập thực một cách hoàn hảo.

Kiểm tra chịu tải có thể giúp chúng ta trả lời những câu hỏi như sau…

* Trang web của tôi có hoạt động tốt vào ngày bận rộn nhất trong năm của không?
* Tôi có đủ cơ sở hạ tầng đám mây hoặc phần cứng để chạy ứng dụng này trên quy mô lớn không?
* Tính năng autoscaling có hoạt động bình thường không?
* Ứng dụng của tôi có mang lại thời gian phản hồi nhanh và trải nghiệm người dùng tốt ngay cả khi peak load không?
* Khi bị đẩy đến breaking point, ứng dụng của tôi có phục hồi một cách nhanh chóng hay bị lỗi nặng và mất dữ liệu?
* Có các sự cố đồng thời hoặc "heisenbugs" trong ứng dụng của tôi chỉ xuất hiện khi tải nặng không?
* Tôi có bị rò rỉ bộ nhớ hoặc cạn kiệt tài nguyên xuất hiện sau khi sử dụng kéo dài không?
* Các hệ thống dự phòng và chuyển đổi dự phòng của chúng ta có hoạt động bình thường không?
* Trả lời những câu hỏi này với dữ liệu cụ thể là rất quan trọng, nếu chúng ta muốn cung cấp trải nghiệm người dùng nhanh chóng và hiệu quả cho khách hàng của mình trong các sự kiện có lưu lượng truy cập cao.

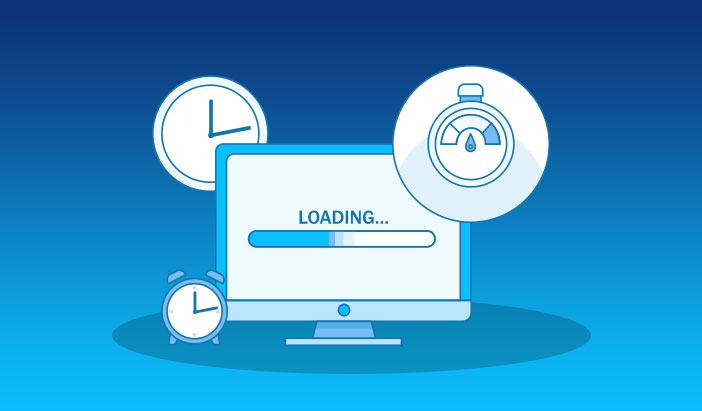
### 1.1.5 Ưu và nhược điểm của Load Testing

Ưu điểm của Load Testing:

* Các vấn đề liên quan đến hiệu suất và tắc nghẽn có thể được xác định trước khi release
* Khả năng mở rộng của hệ thống được cải thiện về cơ sở dữ liệu, phần mềm, mạng, v.v.
* Giảm thiểu rủi ro liên quan đến thời gian ngừng hoạt động của ứng dụng / hệ thống.
* Chi phí thất bại giảm và sự hài lòng của khách hàng tăng lên.

Nhược điểm của Load Testing:

* Người kiểm thử cần có kiến thức về các công cụ và trong một số trường hợp phức tạp, các ngôn ngữ lập trình để thực hiện kiểm tra load test
* Sẽ mất chi phí đáng kể liên quan đến kiểm tra load test vì các công cụ có thể tốn kém, có kiến thức đặc biệt về các công cụ kiểm tra có thể được yêu cầu.



## 1.2 Loadster

### 1.2.1 Giới thiệu chung

Loadster là một nền tảng dùng để thực hiện Load Test cho các ứng dụng web, dịch vụ web và các trang web tĩnh. Là một giải pháp kết hợp giữa đám mây, nó có thể tạo ra tải từ hàng chục vùng đám mây hoặc từ các công cụ tự lưu trữ của riêng chúng ta. Chúng ta có thể kiểm tra các trang web chạy trên đám mây công cộng hoặc sau tường lửa trên mạng riêng của mình. Đây là một công cụ dùng để giảm nguy cơ bị lỗi trang web do lưu lượng truy cập cao.

Loadster có một kho lưu trữ được gọi là kho dự án cho từng dự án, tập lệnh, v.v. Sau khi hoàn thành bài kiểm tra chịu tải, thì kết quả cũng được lưu trữ trong kho này.

Sử dụng Loadster mang lại những lợi ích sau:

* **Tải thử nghiệm ở lớp protocol, hoặc tự động hóa các real headless browsers**
* Headless Browser là một trình duyệt web không có giao diện đồ họa người dùng. Các headless browser cung cấp tương tác tự động một trang web trong một môi trường giống như các trình duyệt web phổ biến khác, nhưng nó được thực hiện thông qua giao diện dòng lệnh hoặc qua một mạng truyền thông.
* Protocol Bots thực thi các yêu cầu HTTP / S theo tập lệnh đối với trang web hoặc API . Với HTTP clients đa luồng mô phỏng trình real browsers hoặc programmatic callers, Protocol Bots là cách hiệu quả nhất về chi phí để thực hiện kiểm thử chịu tải các APIs và static sites.
* Browsers Bots kiểm soát các trình duyệt Chrome không có đầu thực, để kiểm tra tải thực tế nhất có thể. Kiểm soát hàng nghìn trình duyệt thực là cách dễ nhất để kiểm tra các ứng dụng web động và các trang web phức tạp.
* **Giúp hệ thống thực hiện các bài kiểm thử để sẵn sàng cho các đợt tấn traffic spikes và floods**
* **API:**
* Sử dụng bộ dữ liệu, trình xác thực và trình thu thập của Loadster để tải kiểm tra API với các yêu cầu động.
* Kiểm tra bất kỳ loại API HTTP nào, bao gồm REST, GraphQL, JSON-RPC và XML-RPC.
* Nếu sử dụng OpenAPI / Swagger, giao diện tập lệnh của Loadster có thể tạo các bước trực tiếp từ đặc tả API.
* **Bulletproof Web Apps:**
* Loadster xử lý cookie tự động và có thể gửi dữ liệu động với mỗi yêu cầu bằng cách liên kết với tập dữ liệu CSV.
* Thêm các quy tắc xác thực nâng cao để phát hiện lỗi hoặc sự bất thường và nắm bắt các giá trị từ phản hồi để sử dụng lại sau này.
* Đối với logic lặp lại và có điều kiện, hãy viết các khối mã JavaScript của riêng .
* **Indestructible Static Sites**
* Ghi lại các tập lệnh với tiện ích mở rộng trình duyệt Loadster Recorder miễn phí.
* Phát lại chúng với hàng nghìn người dùng đồng thời từ bất kỳ 24 khu vực đám mây nào trên toàn cầu.
* Với nền tảng thử nghiệm đám mây của Loadster, sẽ tiết kiệm thời gian và tiền bạc bằng cách tạo ra các phiên bản được quản lý hoàn toàn mà không cần thiết lập.
* **Load testing cho web apps thực hiện nhanh chóng và dễ dàng với:**
* Ghi kịch bản nhanh chóng với tiện ích mở rộng Chrome hoặc Firefox miễn phí. Hoặc thêm các bước theo cách thủ công.
* Tất cả chỉnh sửa được thực hiện ngay trên trình duyệt.
* Phản hồi kịch bản nhanh chóng.
* Chẩn đoán chi tiết
* Quản lý hoàn toàn bằng đám mây
* Kết quả kiểm tra luồng được thể hiện trực tiếp ngay lúc chạy.
* Báo cáo kiểm tra được tự động tạo thành sau khi thực hiện test.
* Cả nhóm có thể cùng nhau chỉnh sửa

Những công ty nổi tiếng sử dụng Loadster:

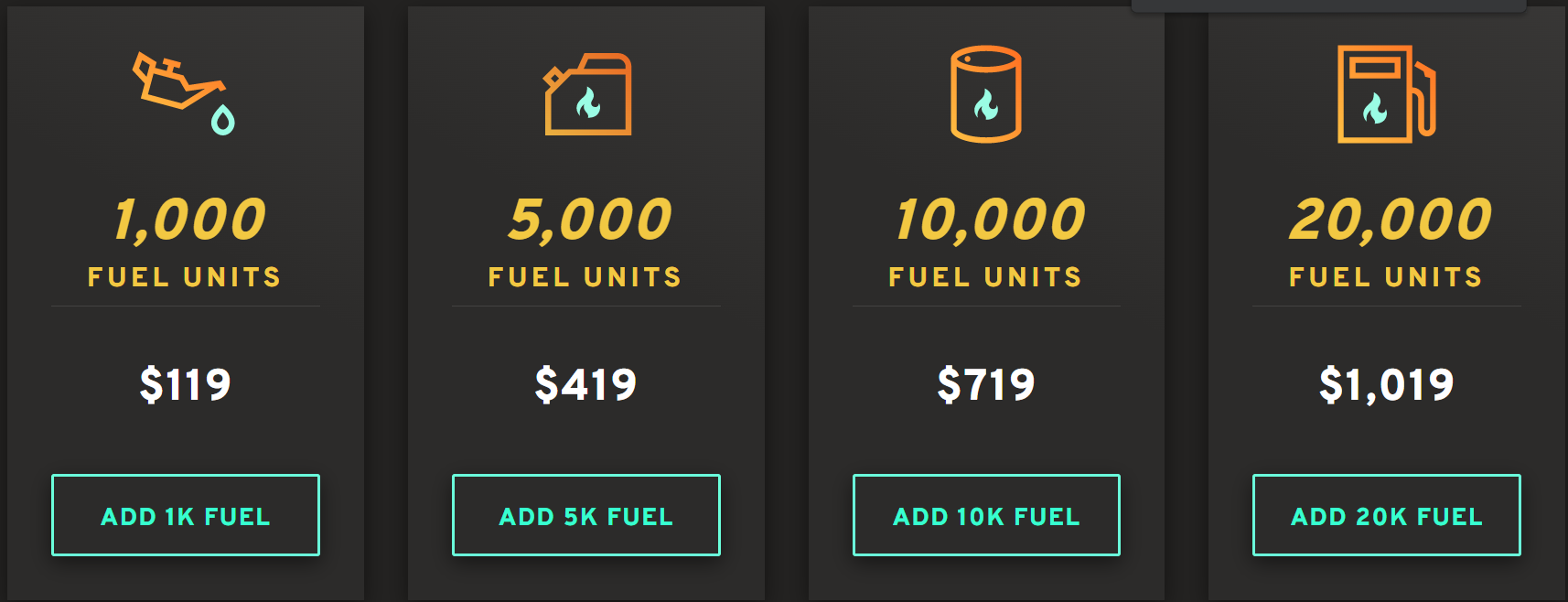


### 1.2.2 Một số thuật ngữ trong Loadster

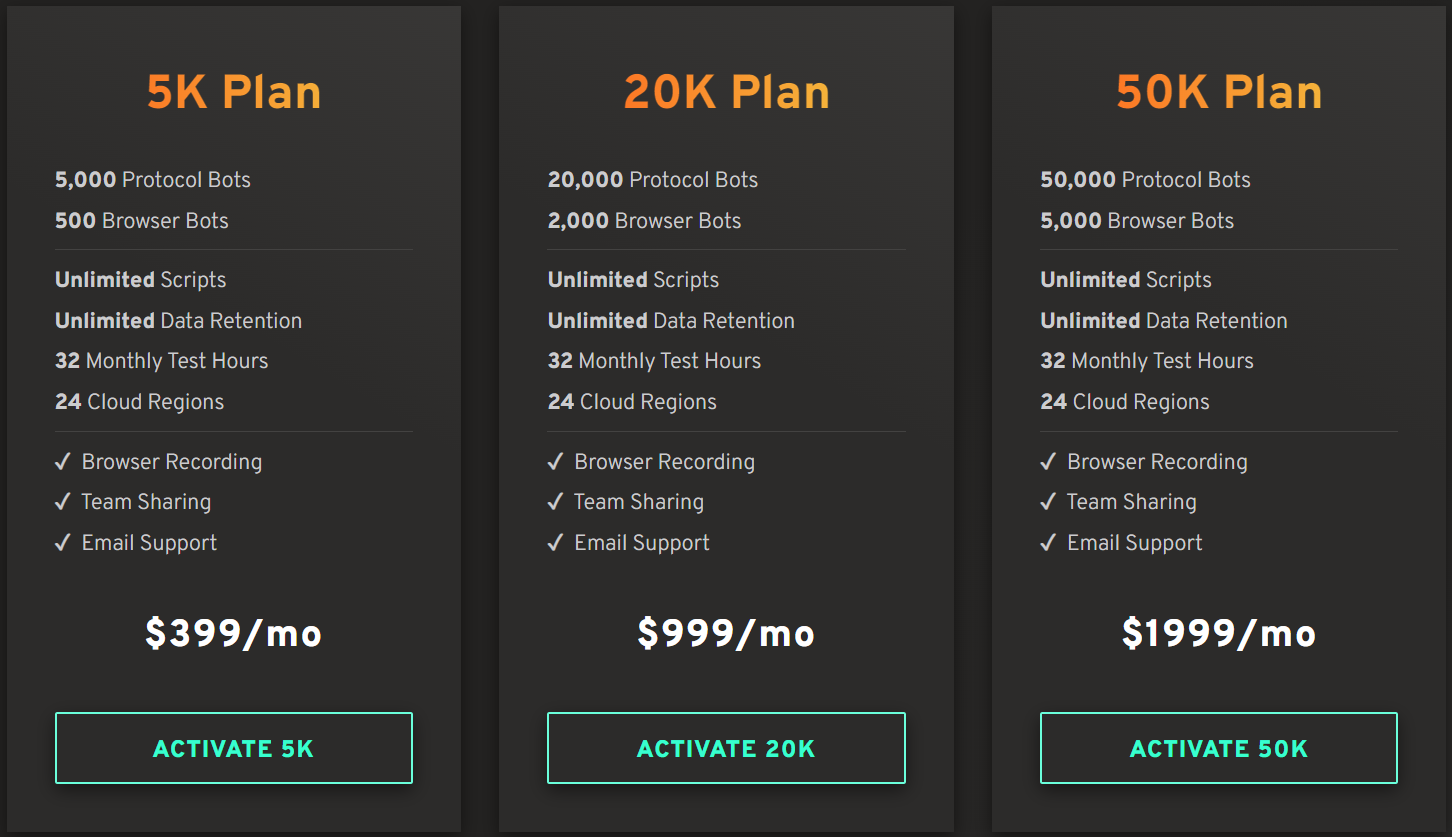
* **Browser scripts:** tự động hóa các trình duyệt web Chrome không đầu. Chúng được thực thi bởi Browser Bots.
* **Protocol Scripts**: một chuỗi các yêu cầu HTTP / HTTPS. Chúng được thực thi bởi Protocol Bots .
* **Site**: thuật ngữ “site” được dùng để mô tả một trang web, một ứng dụng web hoặc một API.
* **Bot (or Virtual User or V-User)**: Một người dùng mô phỏng. Hãy nghĩ về một con người với trình duyệt web. Bây giờ hãy tưởng tượng hàng nghìn người trong số họ, tất cả đều sử dụng trang web của chúng ta cùng một lúc. Mỗi bot duy trì trạng thái phiên độc lập, kho lưu trữ cookie, v.v., giống như một người dùng thực. Load Testing được thực hiện là khi nhiều bot tấn công trang web của chúng ta cùng một lúc. Loadster có hai loại bot: Giao thức Bots, với các máy khách HTTP đa luồng và Bots Trình duyệt, với các trình duyệt web không có đầu thực.
* **Bot Group**: Một nhóm bot (người dùng ảo), tất cả đều chạy cùng một tập lệnh với cùng một cấu hình. Chúng ta có thể có nhiều nhóm trong một bài thực hiện Load Testing nếu chúng ta cần mô phỏng nhiều hành vi hoặc cấu hình Concurrent Users.
* **Script:** Một chuỗi các bước tự động để bắt chước hành vi người dùng thực. Loadster hỗ trợ cả Tập lệnh giao thức và Tập lệnh trình duyệt . Hầu hết các bước trong tập lệnh thực hiện một hành động duy nhất, chẳng hạn như tải trang hoặc nhấp vào liên kết hoặc gửi yêu cầu HTTP. Các bước khác thể hiện thời gian chờ giữa các lần yêu cầu. Tập lệnh cũng có thể bao gồm các Khối mã để chúng ta có thể triển khai các điều kiện, vòng lặp và phân tích cú pháp tùy chỉnh với JavaScript.
* **Scenario**: Một kịch cho một bài kiểm tra chịu tải, được tạo thành từ một hoặc nhiều nhóm bot. Mỗi nhóm trong kịch chúng ta có thể chạy một tập lệnh khác nhau và có các cài đặt và hành vi riêng, chẳng hạn như băng thông mạng mô phỏng, thời gian chờ ngẫu nhiên, v.v.
* **Dataset**: Dataset chứa các giá trị duy nhất cho mỗi bot sử dụng. Khi nhiều bot đang gửi biểu mẫu web hoặc dữ liệu giao dịch khác, chúng ta có thể lấy dữ liệu này từ tập dữ liệu để mỗi lần gửi là duy nhất.
* **Test**: Kiểm tra chịu tải nhiều người dùng. Trong khi thực hiện Load Testing, chúng ta có thể xem kết quả tổng hợp và phân tích lỗi trong thời gian thực. Sau khi quá trình kiểm tra kết thúc, Loadster tạo một báo cáo để chúng ta có thể chia sẻ kết quả với nhóm của mình.
* **Response Times**: là con số gần đúng về thời gian mà người dùng thực của chúng ta sẽ phải đợi một trang hoặc điểm cuối phản hồi, trong các điều kiện tải tương tự như những gì thử nghiệm của chúng ta đang mô phỏng.
* **Avg**: là thời gian phản hồi trung bình có trọng số đơn giản của tất cả các bước trên tất cả các nhóm bot trong kiểm thử.
* **p80 and p90**: giá trị xấp xỉ của thời gian phản hồi phần trăm thứ 80 và 90. Loadster tính toán chúng trước tiên ở cấp độ nhóm (trả về phân vị cho mỗi nhóm) và sau đó tính trung bình phân vị trên mỗi nhóm. Mặc dù không phải là phần trăm thực sự của tất cả các giá trị riêng lẻ, nhưng cách tiếp cận này đáp ứng nhu cầu tương tự trong khi ít chuyên sâu hơn về mặt tính toán.
* **Network Throughput**: cho biết có bao nhiêu byte / bit đang được gửi và nhận trên mạng. Giá trị được tính dựa trên tổng kích thước của mỗi yêu cầu được gửi (tải lên) và mỗi phản hồi nhận được (tải xuống). Tính toán dựa trên cửa sổ 6 giây quay vòng.
* **Transaction Throughput**: cho biết số trang và số lần truy cập trung bình mỗi giây đang được tạo trong một khoảng thời gian luân phiên.
* **Pages**: đề cập đến bất kỳ request hoặc điều hướng chính nào được thực hiện bởi một bước trong tập lệnh của chúng ta, cho dù trang đó có tài nguyên bổ sung được tập hợp bên dưới hay không.
* **Hits**: bất kỳ yêu cầu nào được thực hiện bởi một bước, cộng với bất kỳ tài nguyên bổ sung nào có thể đã được tải cùng với bước đó. Về cơ bản, nó có nghĩa là tổng số yêu cầu HTTP đến tất cả các điểm đến.
* **Transactions**: hiển thị tổng số tích lũy của các loại transactions chính.
* **Iterations**: đề cập đến số lần lặp lại hoàn chỉnh đã được thực thi đối với bất kỳ tập lệnh nào của chúng ta. Mỗi khi bot hoàn thành một lần lặp lại tập lệnh của chúng ta, nó sẽ bắt đầu lại, miễn là vẫn còn thời gian trong giai đoạn thử nghiệm. Số lần lặp lại là một dấu hiệu tốt về số lượng phiên người dùng hoặc hành trình của người dùng đã được mô phỏng.
* **Errors**: lỗi giao thức nào, lỗi xác thực hoặc lỗi không mong muốn khác. Nếu các tập lệnh được thiết kế đúng cách và trang web chúng ta đang kiểm tra đang xử lý tải một cách duyên dáng, chúng ta sẽ không thấy lỗi. Nếu chúng ta thấy lỗi, họ có thể yêu cầu điều tra thêm.
* **Traces**: cung cấp thêm chi tiết về các transactions nhất định.

Giá sử dụng dịch vụ của Loadster:

Loadster Fuel: Sử dụng đến khi nào hết nhiên liệu thì nạp thêm, không giới hạn thời gian



Monthly Plans: thích hợp cho những ai thường xuyên thực hiện kiểm thử trên Loadter



### 1.2.3 Quy trình thiết kế kịch bản thực tế kiểm thử chịu tải

Một kịch bản kiểm thử chịu tải là một công thức cho số lượng bot sẽ chạy các tập lệnh của chúng ta đồng thời trong một thử nghiệm tải. Mỗi bot đại diện cho một người dùng.

Một kịch bản có thể có nhiều nhóm bot, mỗi nhóm chạy một tập lệnh khác nhau. Ngoài ra, mỗi nhóm bot có lịch trình và cấu hình riêng để tăng tốc, tiếp tục ở tải cao nhất và giảm dần.

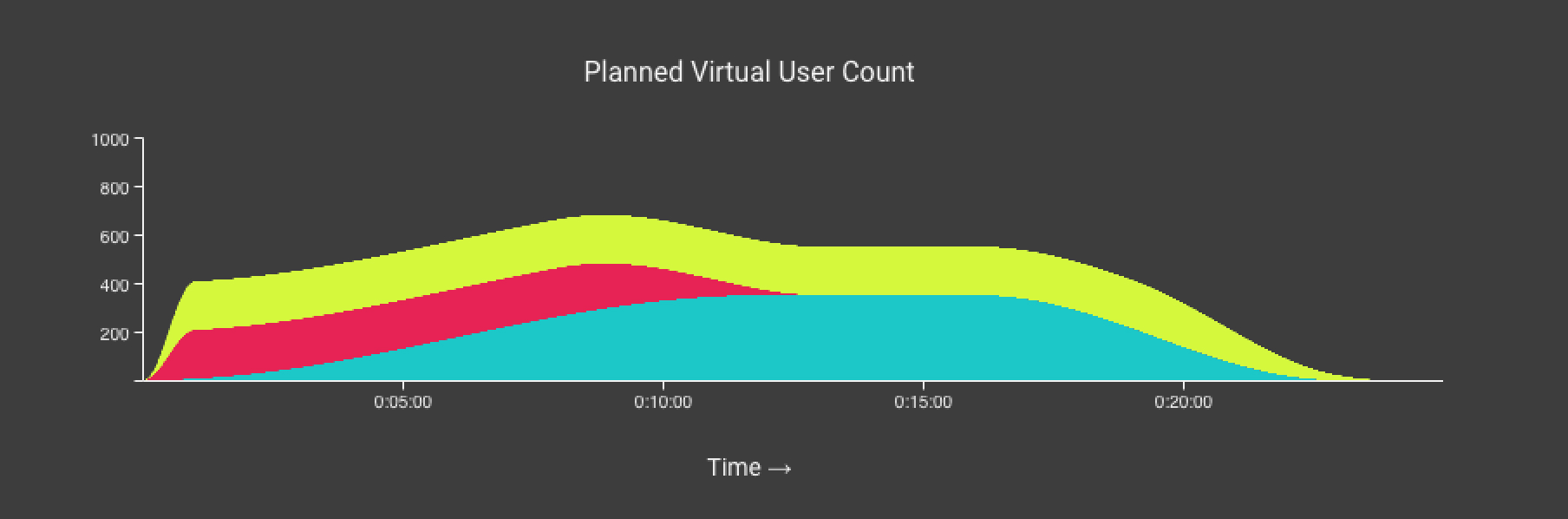
Trong giai đoạn tăng tốc, số lượng bot đang chạy bắt đầu từ 0 và thêm dần nhiều bot hơn cho đến khi đạt được số lượng mục tiêu. Trong giai đoạn cao điểm, bot chạy tập lệnh và lặp lại miễn là còn thời gian trong quá trình kiểm tra. Trong giai đoạn dốc xuống, các bot dần dần thoát ra cho đến khi tất cả kết thúc. Kịch bản chỉ định số lượng bot và thời gian của mỗi giai đoạn này.

**Để thiết kế một kịch bản kiểm thử cần làm những công việc sau**

**1.** Chọn số lượng bot tương tự với số lượng người dùng truy cập đồng thời của trang web sẽ kiểm thử Load Test. Concurrent Users của chúng ta có thể khác với tổng số người dùng truy cập trang web của chúng ta trong một khoảng thời gian dài. Ví dụ: chúng ta có thể có 50.000 người dùng duy nhất (tổng số người dùng), nhưng chỉ có 1.000 người hoạt động tại một thời điểm (Concurrent Users).

**2.** Xác định hành vi của người dùng thực tế và tạo cái tập lệnh phù hợp để Loadster sao chép hành vi của từng người dùng và xây dựng thành một kịch bản với các nhóm bot chạy tập lệnh đó. Ví dụ: tại bất kỳ thời điểm cụ thể nào, cửa hàng trực tuyến của chúng ta có thể có:

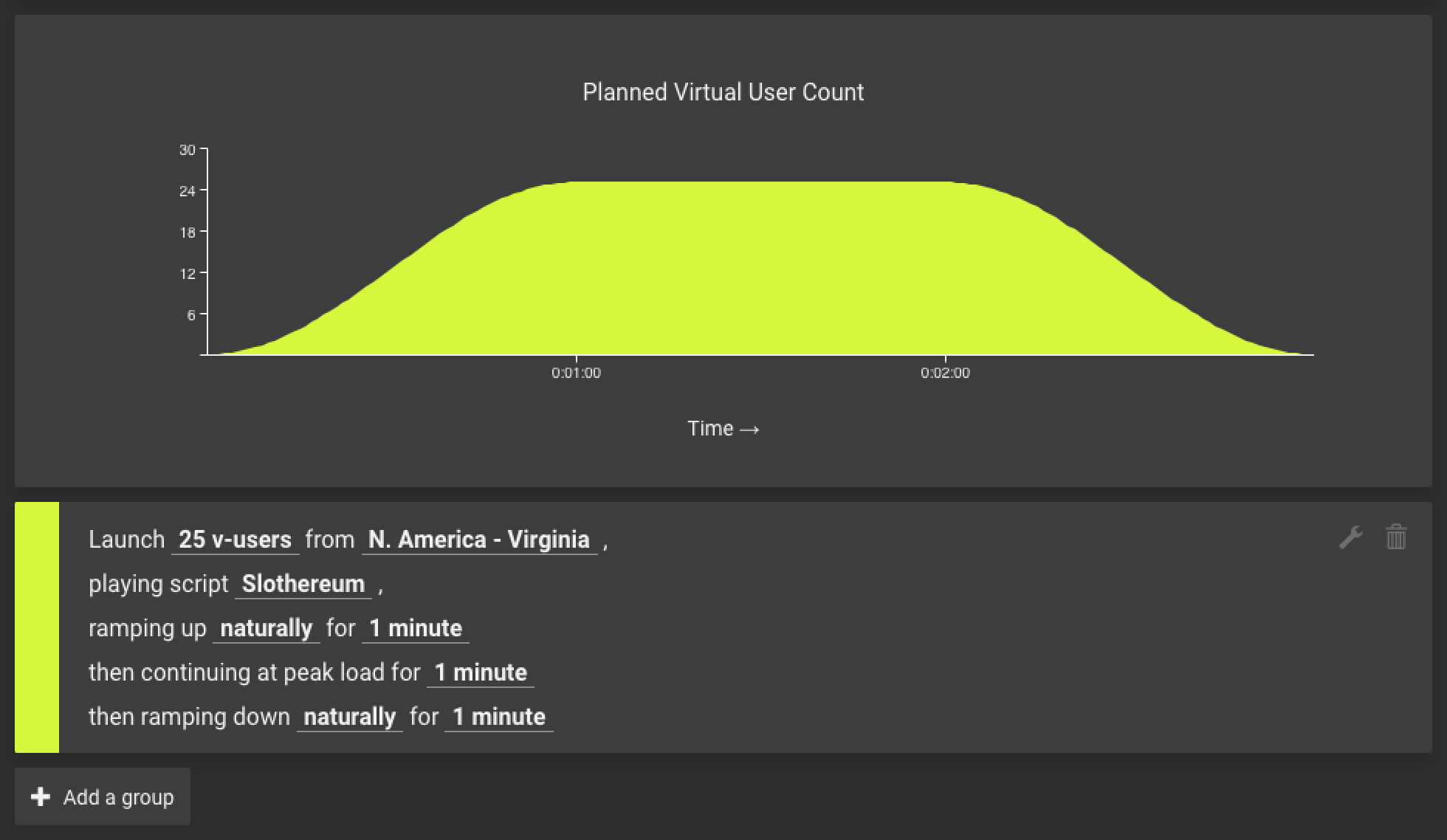
* 400 người dùng duyệt qua cửa hàng trực tuyến
* 150 người dùng thêm các mặt hàng vào giỏ hàng và thanh toán
* 150 người dùng đang xem tài khoản của họ và các đơn đặt hàng trước đây



Ba nhóm bot, đạt đỉnh điểm là 700 Concurrent Users, với thời lượng khác nhau

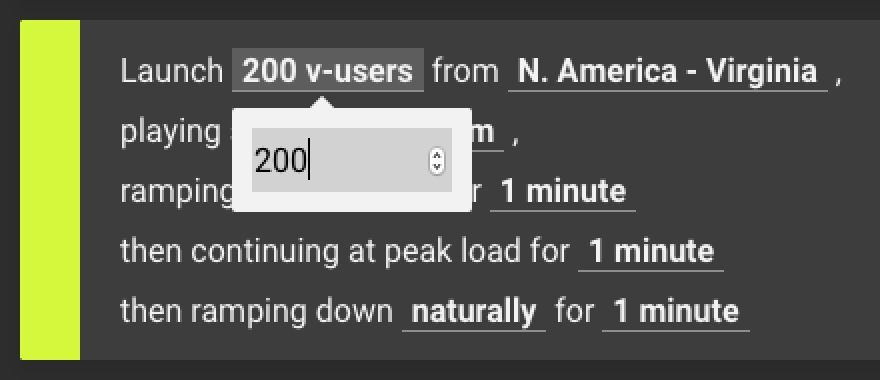
**3.** Chỉnh sửa kịch bản

* Thêm và xóa nhóm bot: Mọi tình huống cần ít nhất một nhóm bot để chạy. Để tạo một nhóm bot, hãy nhấp vào nút Thêm nhóm trong trình chỉnh sửa kịch bản. Mỗi nhóm có cấu hình và lịch trình riêng. Khi chúng ta chạy thử nghiệm tải, các nhóm này sẽ thực thi song song.

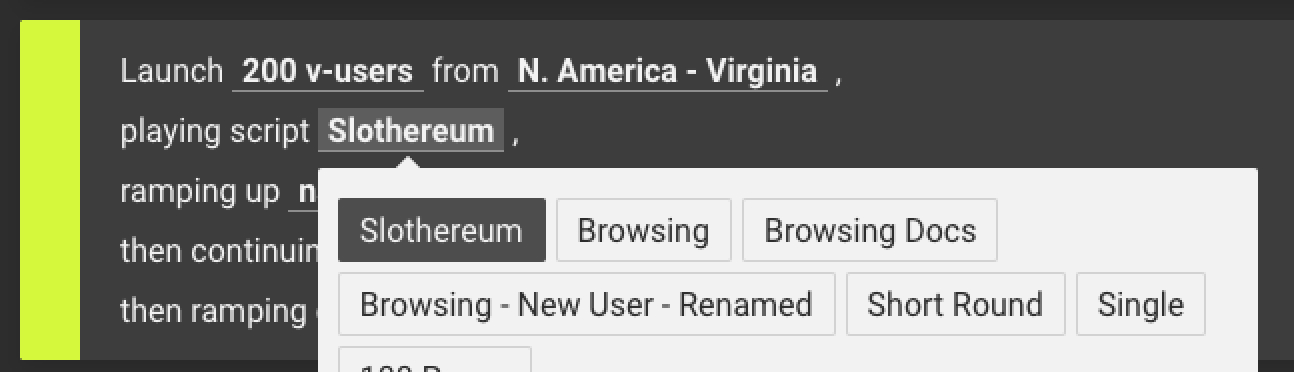


Một nhóm với 25 bot chạy tập lệnh 'Slothereum' trong 3 phút

* Đặt số lượng bot: Mỗi nhóm có thể có từ 1 đến 9999 bot.

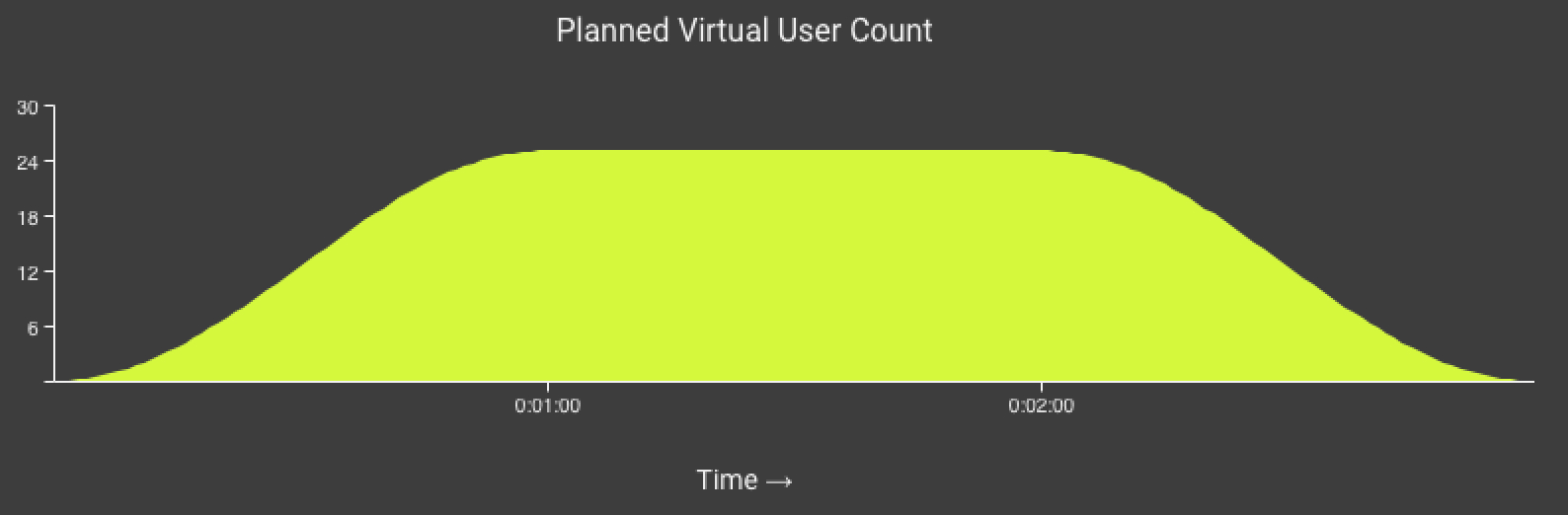


* Chọn tập lệnh: Mỗi nhóm bot được định cấu hình để chạy một tập lệnh. Để chọn một tập lệnh khác, hãy nhấp vào tên tập lệnh và chọn một tập lệnh khác.

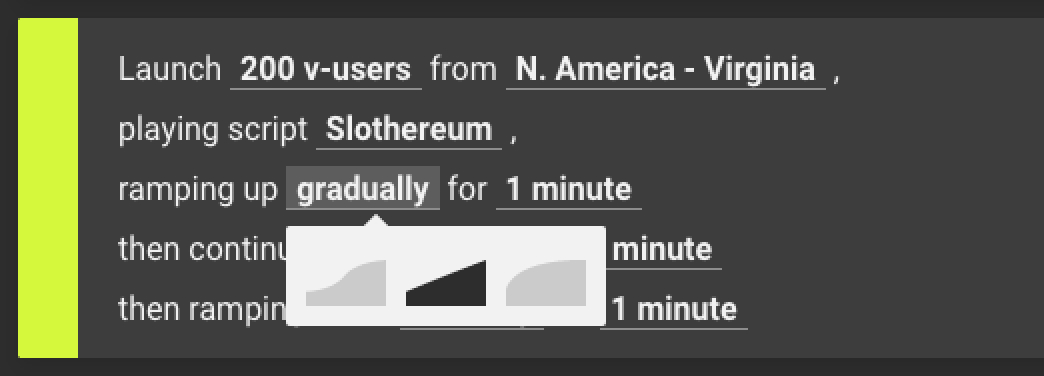


Chọn tập lệnh cho nhóm bot

* Tăng lên và xuống: Mỗi giai đoạn trong số này (giai đoạn tăng tốc, giai đoạn cao điểm và giai đoạn giảm tốc) có thời lượng riêng tính bằng phút. Chúng ta cũng có thể chọn công thức tăng hoặc giảm để xác định tốc độ tải nên tăng hoặc giảm.

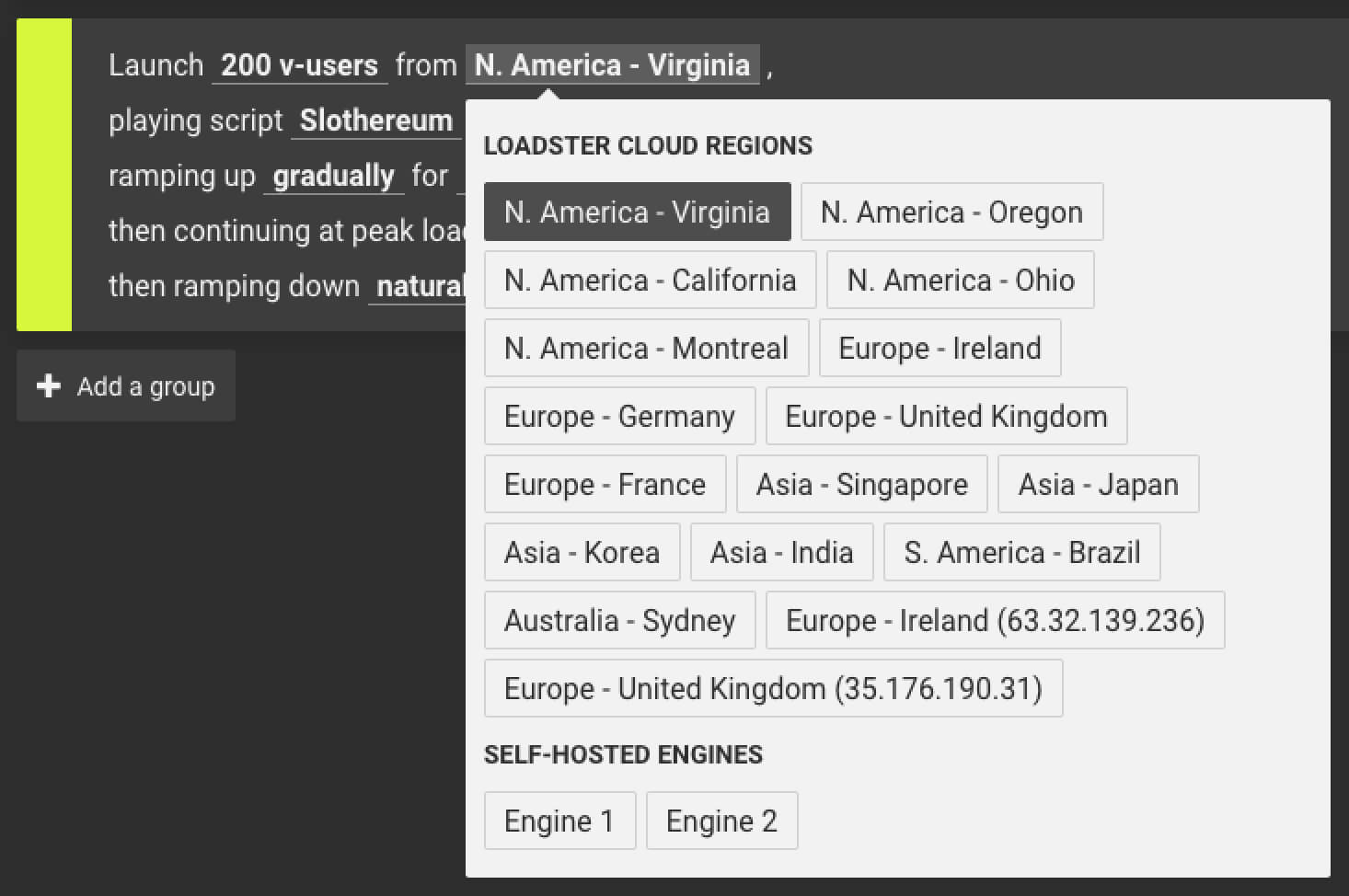


Hình ảnh trực quan về số lượng bot tăng và giảm

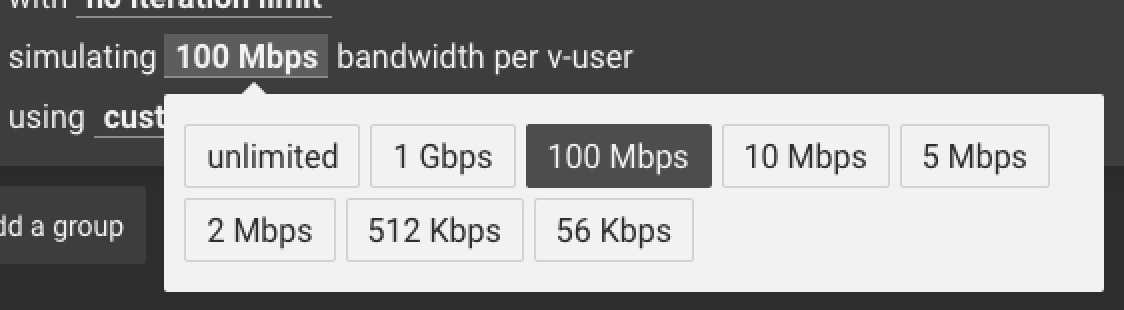


Hình ảnh trực quan về số lượng bot tăng và giảm

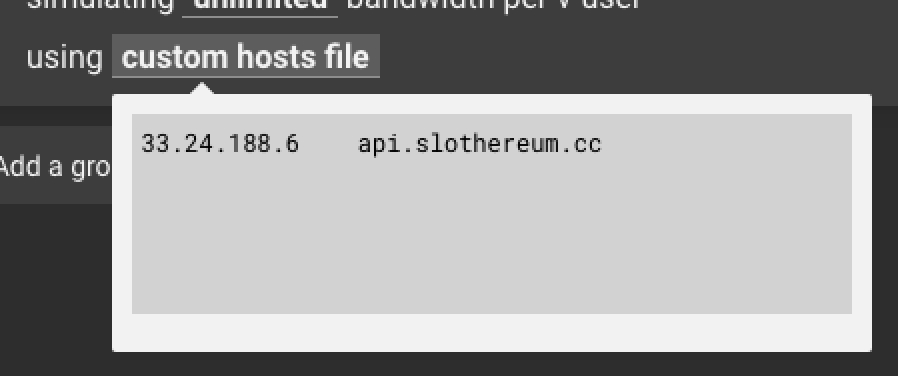
* Chọn Load Engine or Cloud Region: Mỗi nhóm bot chạy trên Loadster Engine tự lưu trữ hoặc một trong các vùng Loadster Cloud. Nếu trang web chúng ta đang kiểm thử có thể truy cập công khai, nên chọn Cloud Region. Nếu cần tạo tải từ mạng riêng, hãy đọc thêm về kiểm thử với Loadster Engine .



**4.** Mô phỏng băng thông mạng: thông thường, các bot sử dụng tất cả băng thông mạng có sẵn cho chúng.



**5.** Ghi đè DNS: Khi thử nghiệm trên nhiều môi trường của ứng dụng, chúng ta có thể muốn hướng lưu lượng truy cập đến các máy chủ nhất định. Loadster có thể thực hiện điều này bằng cách ghi đè độ phân giải tên DNS.



Ghi đè các mục nhập DNS

### 1.2.4 Quy trình thực hiện Load Testing với Loadster

**Bước 1**: Tạo một tập lệnh để mô phỏng đường dẫn của người dùng qua trang web chúng ta đang kiểm thử, càng thực tế càng tốt.

* Tập lệnh mô phỏng đường dẫn của một người dùng qua trang web của chúng ta. Chúng ta có thể tạo nhiều tập lệnh nếu chúng ta muốn kiểm tra nhiều đường dẫn người dùng trong cùng một kiểm thử chịu tải, đây thường là một ý tưởng hay.
* Các tập lệnh của chúng ta phải sao chép hành vi của người dùng thực càng gần càng tốt. Bộ tập lệnh kiểm thử càng thực tế, thì kết quả kiểm thử chịu tải càng hữu ích. Rốt cuộc, kết quả của một bài kiểm tra chịu tải chỉ có giá trị như các giả định đi vào nó.
* Các bot của Loadster có thể chạy các Tập lệnh Giao thức và Tập lệnh Trình duyệt.
* Thiết kế một kịch bản kiểm tra chịu tải mô tả có bao nhiêu bot sẽ chạy tập lệnh, từ Clound Region hoặc công cụ tự lưu trữ nào và trong bao lâu.
* Chạy kiểm thử chịu tải và quan sát trong thời gian thực cách trang web đang phản hồi và những lỗi Loadster đang gặp phải.
* Tập lệnh Giao thức là một chuỗi các yêu cầu HTTP / HTTPS để kiểm tra mức thấp. Vì chúng cho phép kích hoạt từng yêu cầu riêng lẻ, các tập lệnh giao thức rất lý tưởng để kiểm thử các API và các trang web tĩnh đơn giản.
* Tập lệnh trình duyệt kiểm soát các trình duyệt web không có đầu thực, tương tự như Selenium hoặc Puppeteer hoặc Playwright. Với các bước như “điều hướng đến trang” và “nhấp vào phần tử”, các tập lệnh trình duyệt là tốt nhất cho các ứng dụng web phức tạp.
* Kiểm tra chịu tải bằng tập lệnh trình duyệt tốn nhiều chi phí hơn so với tập lệnh giao thức, bởi vì các trình duyệt thực sử dụng nhiều tài nguyên hơn.
* Với Loadster, chúng ta có thể tự do chọn tập lệnh giao thức hoặc tập lệnh trình duyệt (hoặc cả hai!) Tùy thuộc vào ngân sách của chúng ta và điều gì có ý nghĩa đối với trang web chúng ta đang kiểm thử.
* Phân tích kết quả bài kiểm tra của chúng ta và tìm ra lỗi.

**Bước 2**: Xử lý dữ liệu động

* Nhiều trang web có một cách để người dùng gửi dữ liệu của họ thông qua các biểu mẫu web hoặc tải lên. Điều này đặc biệt đúng với các ứng dụng web và API. Ví dụ: một trang đăng nhập yêu cầu mỗi người dùng gửi tên người dùng và mật khẩu của riêng họ.
* Kiểm thử chịu tải trong đó mọi bot gửi dữ liệu chính xác không tuyệt vời cho các ứng dụng web động, vì ứng dụng hoặc cơ sở dữ liệu có thể lưu vào bộ nhớ cache nhiều thứ, dẫn đến kết quả tốt giả tạo.
* Để cung cấp dữ liệu duy nhất cho mỗi bot, hãy tham số hóa tập lệnh với các biến và tập dữ liệu.

**Bước 3**: Chạy kiểm thử chịu tải

* Để chạy kiểm chịu tra tải với nhiều bot cùng một lúc, chúng ta sẽ cần tạo một kịch bản kiểm chịu tra tải . Các kịch bản xác định một hoặc nhiều nhóm bot và mỗi nhóm bot bao gồm một số bot thực thi một tập lệnh. Một kịch bản chỉ định số lượng bot, hướng dẫn tăng và giảm tải, thời gian chạy kiểm thử, sử dụng băng thông đầy đủ hay mô phỏng kết nối chậm hơn, v.v.
* Khi chúng ta đã thiết kế kịch bản cho mẫu tải muốn kiểm tra, hãy khởi chạy kịch bản để chạy kiểm chịu tra tải .
* Đảm bảo theo dõi quá trình kiểm tra khi nó chạy, để xem ứng dụng đang phản hồi như thế nào khi tải. Luôn có thể dừng kiểm tra sớm, sửa đổi kịch bản hoặc kịch bản và khởi động lại kiểm tra nếu cần.

**Bước 4**: Phân tích kết quả kiểm tra. Sau khi Loadster kết thúc quá trình kiểm tra chịu tải và thu thập và xử lý dữ liệu kiểm thử, đã đến lúc phân tích và diễn giải kết quả để có thể đánh giá xem kiểm thử đạt hay không và lập kế hoạch cho các bước tiếp theo.

Các chỉ số hiệu suất chính ở đầu mỗi báo cáo Loadster là:

* Duration - Toàn bộ thời lượng của bài kiểm tra từ đầu đến cuối.
* Bots - Số lượng bot cao nhất chạy bất kỳ lúc nào trên tất cả các nhóm.
* Download - Tổng số byte được tải xuống (tiêu đề phản hồi HTTP + nội dung).
* Upload - Tổng số byte được tải lên (tiêu đề + nội dung yêu cầu HTTP).
* Total Pages - Tổng số “pages” cấp cao nhất được yêu cầu thành công.
* Total Hits - Tổng số “pages” cấp cao nhất cộng với mọi tài nguyên trang được bao gồm đã được yêu cầu thành công.
* Total Errors - Tổng số lỗi HTTP và lỗi xác thực.
* Total Iterations - Tổng số lần lặp lại tập lệnh đã hoàn thành giữa tất cả các bot từ tất cả các nhóm.

### 1.2.5 Các công cụ của Loadster

**Cloud Regions**: Loadster hỗ trợ kiểm tra tải theo yêu cầu từ 24 khu vực đám mây với Google Cloud Platform và Amazon Web Services là những nhà cung cấp dịch vụ đám mây cơ bản. Các phiên bản đám mây được khởi chạy khi chạy kiểm tra tải (thường trong khoảng 1-3 phút). Truy cập <https://loadster.app/manual/cloud-regions/> để biết thêm thông tin về 24 khu vực.

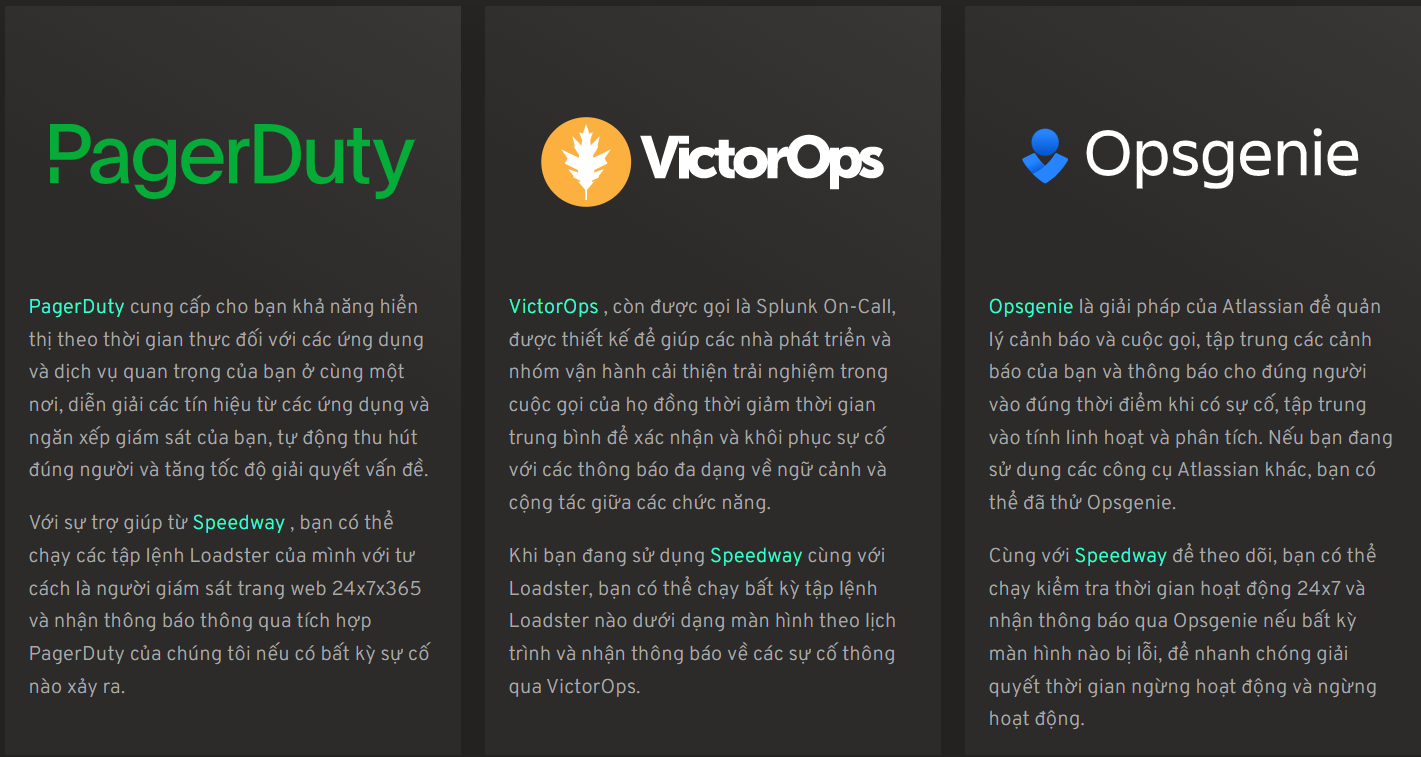


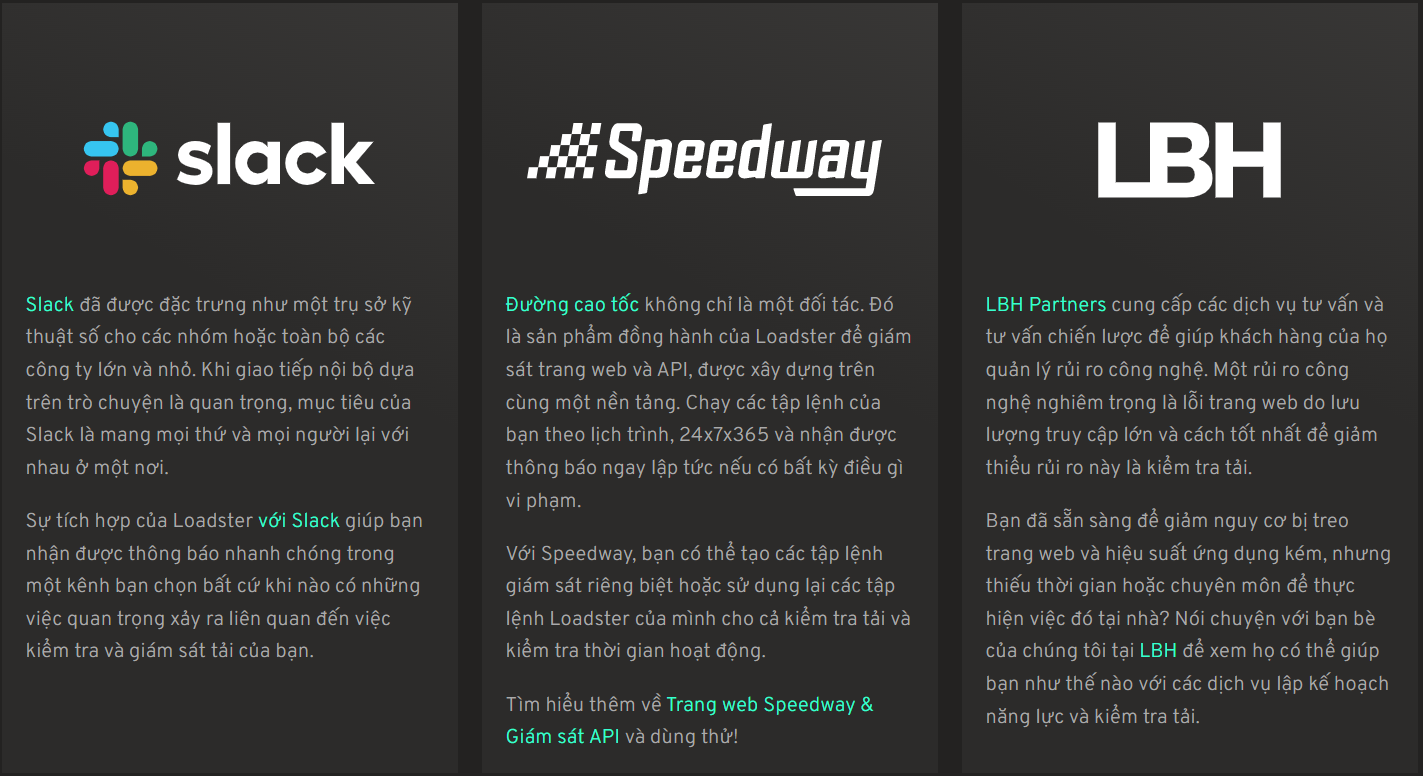
**Loadster Engine**: dành cho mạng riêng không thể truy cập từ các vùng đám mây của Loadster, cần phải lưu trữ các công cụ của riêng mình để thực sự tạo ra tải. Loadster Engine chạy trên bất kỳ nền tảng nào hỗ trợ Docker . Điều này bao gồm các phiên bản Windows, Mac và Linux gần đây. Chỉ cần có máy có ít nhất 2 lõi CPU và 2BG trống.

**Loadster CLI**: Tích hợp liên tục (CI) là một cách được sử dụng rộng rãi để tăng tốc quá trình phát triển phần mềm, bằng cách tích hợp code và kiểm thử thường xuyên thông qua tự động hóa. Nhiều nhóm chạy các bộ kiểm thử chức năng tự động như một phần của quá trình xây dựng và triển khai của họ. Loadster CLI là cách thích hợp để khởi chạy các bài kiểm tra Loadster từ đường ống CI, máy chủ xây dựng hoặc công việc cron của chúng ta. Nó có sẵn dưới dạng bản nhị phân riêng dành cho Windows, Mac và Linux, đồng thời giúp việc chạy và quan sát các bài kiểm tra tải trở nên dễ dàng.

**Loadster Recorder**: dùng để ghi lại lưu lượng truy cập trình duyệt và tạo tập lệnh Loadster, gồm Loadster Recorder miễn phí cho Chrome hoặc Loadster Recorder cho Firefox và xử lý ghi cho cả Loadster và Speedway . Mã nguồn mở và chúng ta có thể xem lại mã nguồn trên GitHub: https://github.com/loadster/loadster-browser-extension

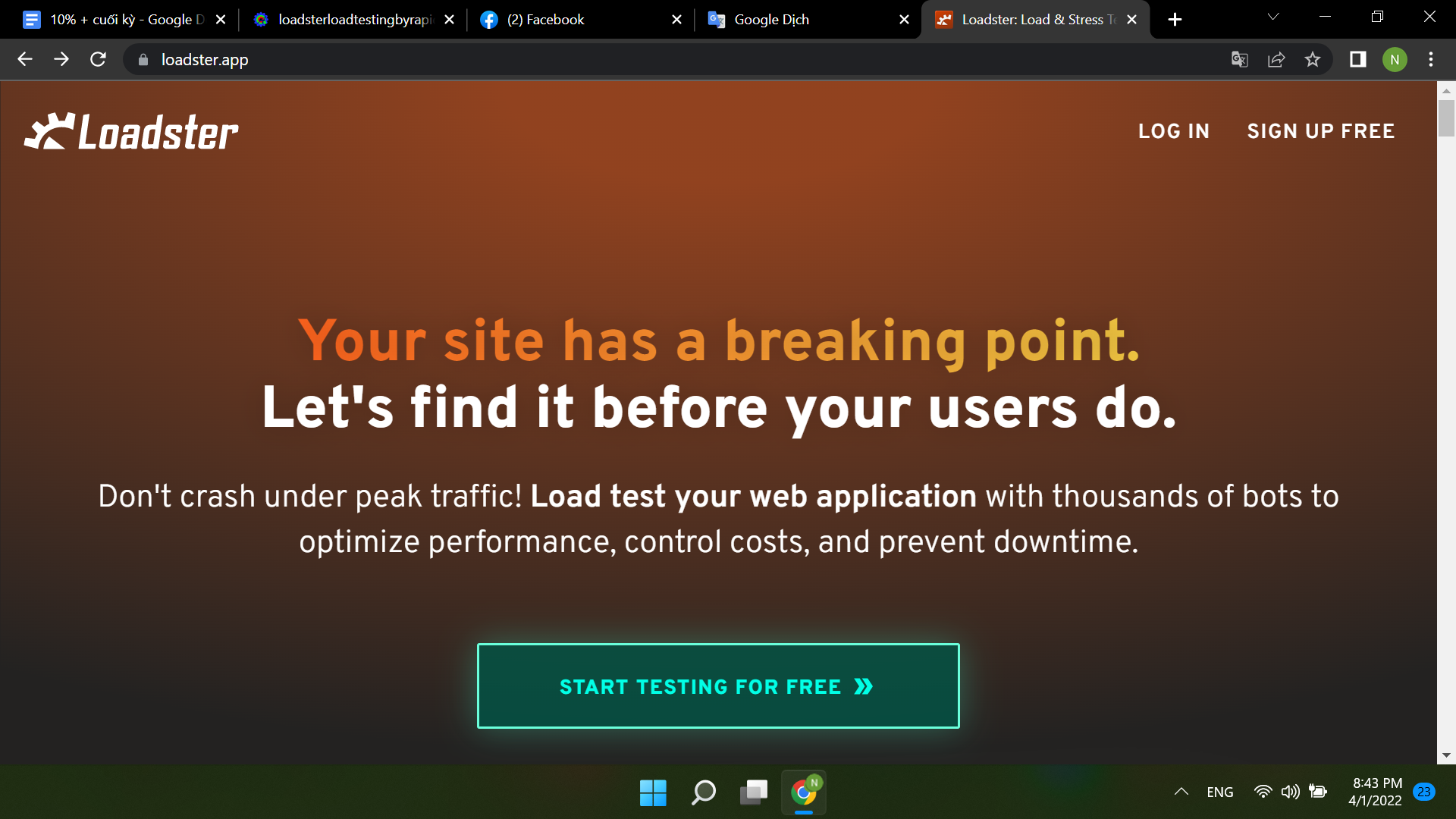
### 1.2.6 Đối tác của Loadster



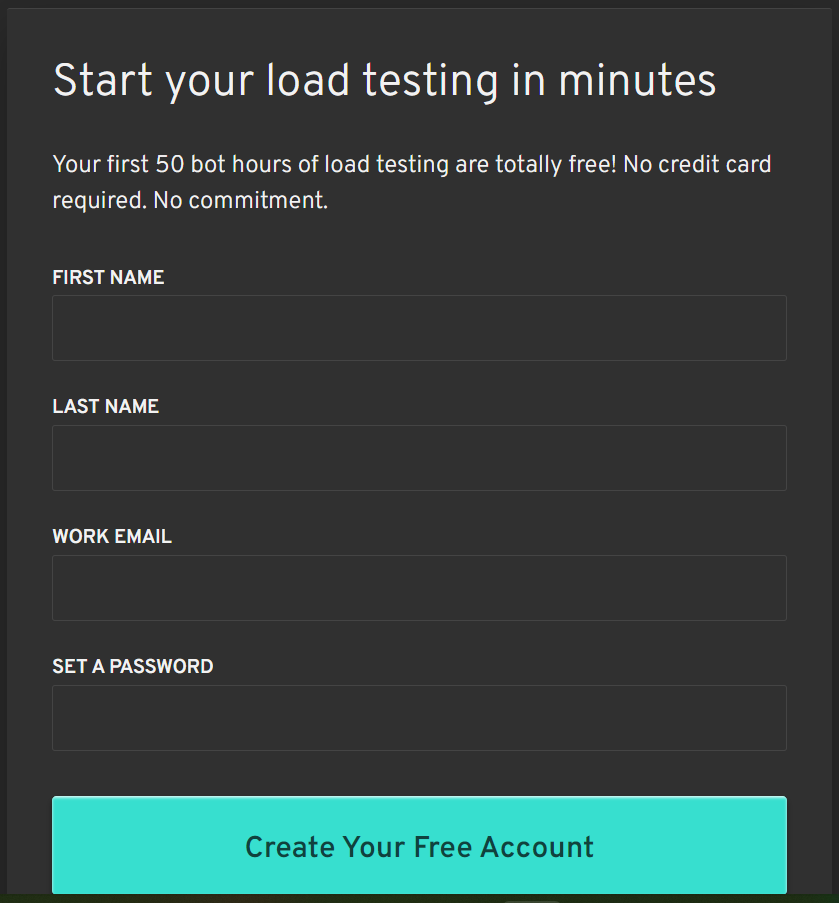


# 2. Cách sử dụng LoadStorm

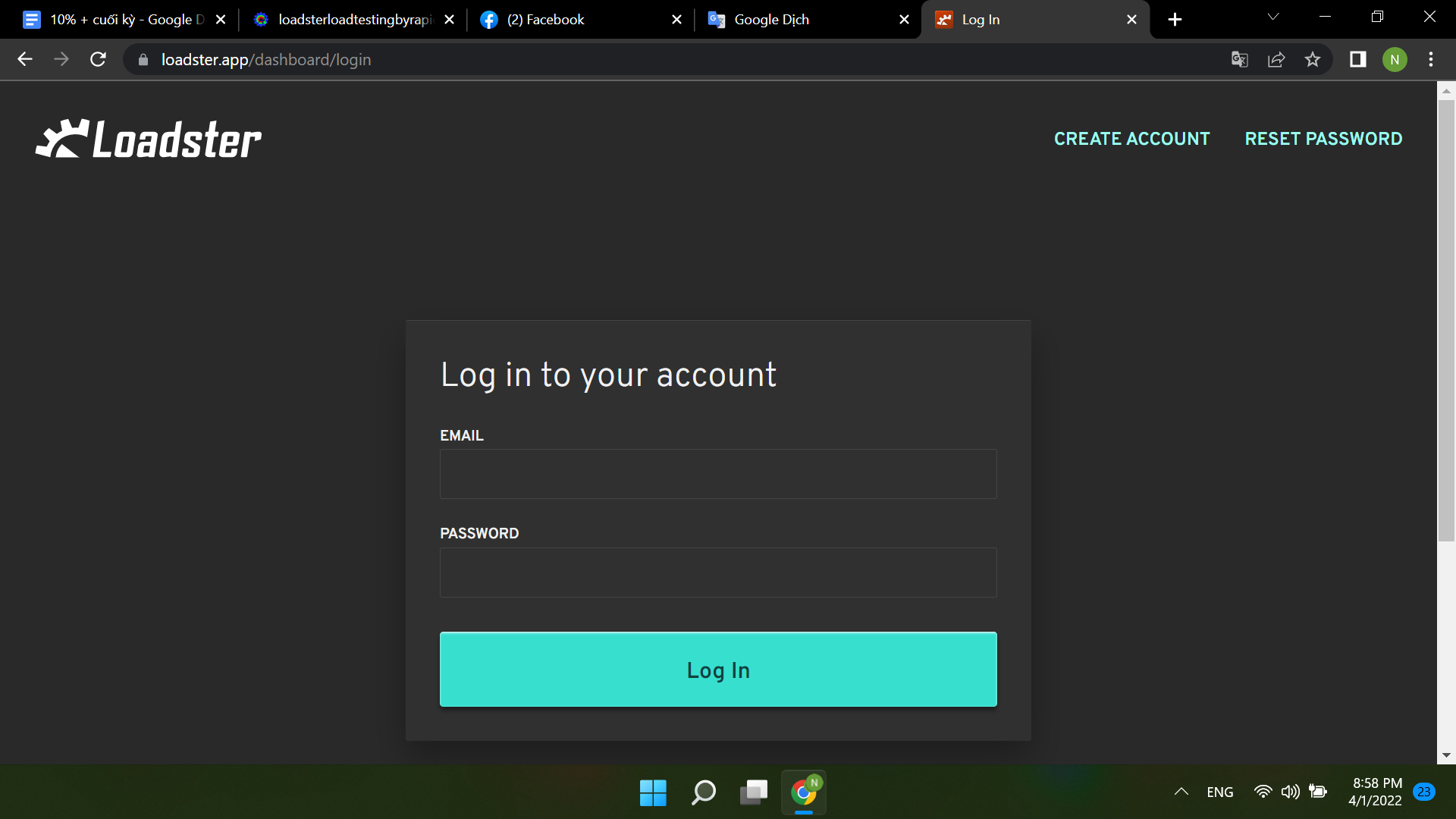
Bước 1: truy cập vào trang web <https://loadster.app/>, chọn “Start testing for free” để sử dụng thử miễn phí



Bước 3: Nhập thông tin để đăng ký tài khoản (nếu đã có tài khoản có thể bỏ qua bước này)



Bước 4: Nhập email và password để đăng nhập



# 3. Tổng kết

Loadster là một công cụ thực hiện Load Testing cho các trang web, ứng dụng web, dịch vụ web HTTP và API. Nó giúp chúng ta, dễ dàng, mô phỏng thực tế một lượng lớn người dùng để tìm các nút thắt cổ chai, cải thiện độ tin cậy và tối ưu hóa trải nghiệm người dùng. Loadster Performance là người tạo ra Loadster là một công cụ Load Test thương mại. Nó giúp mô phỏng và dự đoán cách các ứng dụng web sẽ hoạt động khi hiệu suất tải cao. Công cụ này tốt nhất để xác định tắc nghẽn hiệu suất trong ứng dụng của chúng ta.