🏁 De 0 a 100: Performance Testing y k6 a Máxima Velocidad

[**🏎️ Bienvenida e Introducción 3**](#_c3dtx4qnfwho)

[¿Por qué es Importante el Performance Testing? 3](#_wgvjfrgkdfuv)

[¿Qué Puedes Esperar Aprender? 3](#_x4pz2htrz6gd)

[📑 Sobre Este Paddock Digital 4](#_an7l31t3c28r)

[¿Por qué necesitas este documento? 4](#_u8hzwrh98gob)

[¿Cómo Navegar en Este Paddock Digital? 4](#_obxjg0k0u6i1)

[💬 Participa en la Carrera: Comentarios y Preguntas 5](#_5prygcwx4erq)

[**🧰 Requisitos de Boxes: Preparativos antes de la Carrera 6**](#_3bkmgyqum8yw)

[🛠️ Configuraciones Previas 6](#_bl4t0sc38yok)

[💻 Entorno de Desarrollo 6](#_v9i50wxadbtq)

[📦 Herramientas Específicas 6](#_6fsu7jjsyfjv)@Robert Ariechi CGTS

[🗄️ Repositorio y Código 6](#_uznlfnwubr28)

[🌐 Acceso a Internet 6](#_pkstyacvygj0)

[🔐 Credenciales y Permisos 6](#_qg0lk3eacmnj)

[📚 Materiales y Documentación 6](#_gs707abdp480)

[📖 Documentación Oficial 6](#_qigpujp4o6xj)

[📚 Libros y Artículos 6](#_265pj888brub)

[📹 Videos y Webinars 7](#_stm37vnhar3f)

[🛠️ Repositorios y Código Ejemplo 7](#_q4nzwvgfs3e9)

[📋 Plantillas y Scripts 7](#_tvvot58j3bmp)

[📊 Dashboards Predefinidos 7](#_onan6ujcu3pm)

[🧠 Conocimientos Necesarios 7](#_f82iufykadgb)

[🖥️ Fundamentos de Programación 7](#_lm01isw6sp08)

[🌐 Conceptos Básicos de Red 8](#_t7u7vyy6ikbz)

[🛠️ Herramientas de Desarrollo 8](#_b6m62mjurha3)

[⚙️ Introducción a DevOps 8](#_jlbpv68l2edj)

[📊 Interés en el Performance Testing 8](#_gyr4edz6kfnq)

[**EN LA PISTA 9**](#_cpoipx95y0cx)

[🚦 ¡Enciende los Motores!: Introducción al Performance Testing y k6 9](#_otca68ll27k0)

[● 🤔 ¿Por qué nos Importa el Performance Testing? 9](#_fx00ghuoiev)

[● 🧪 Tipos de Pruebas: Carga, Estrés, Resistencia y Más 9](#_ksnskbco17ez)

[● 🛠️ Introducción y Razones para Usar k6 9](#_w1rd7731r1rh)

[● 🚀 Primer Arranque y Configuración Básica con k6 9](#_yggbe8oyuafu)

[🛠️ Manos al Volante: Escritura y Análisis de Scripts con k6 10](#_32fbj1p3hat4)

[● 📝 Escenarios Básicos y Avanzados 10](#_ev72p8o5xf5s)

[● ✅ Uso de Check, Umbrales y Asserts 10](#_pgxw3shkivm0)

[● 📊 Lanzamiento de Pruebas y Análisis de Resultados 10](#_wzlv9cngngz)

[🚧 Pit-Stop Técnico: Integración de k6 en CI/CD 11](#_y4sbt775k67c)

[● 🔧 Integración con Jenkins, GitLab CI y Otros 11](#_mdfwvm3d548o)

[● 🚨 Automatización de Pruebas: Umbral, Alertas y Mucho Más 11](#_pgcofohq8sj5)

[● 🛠️ Práctica: Integración con una Herramienta de Pipeline 11](#_4cjiwda6tuob)

[🌐 Adelantamientos Estratégicos: k6 Browser y Pruebas Híbridas 12](#_39ar7fm8jm7l)

[● 🌐 ¿Qué es k6 Browser? Ventajas y Configuración 12](#_jv29wwhjcpnz)

[● 🤝 Creación y Gestión de Pruebas Híbridas 12](#_lxm6s5mwgw4e)

[🎯 Zonas de DRS: Performance Left Testing 13](#_d6i5yqdy7cr5)

[● 📚 Teoría y Beneficios 13](#_enrq36411vwo)

[● 🛠️ Implementación Práctica con k6 13](#_4g7uumwbj9t3)

[📊 ¡Monitor de Carrera! Dashboards en Grafana 14](#_a0ktj2jbkev3)

[● 🎨 Introducción y Configuración de Grafana con k6 14](#_avrzq6u4jw7n)

[● 📉 Diseño y Personalización de Dashboards 14](#_2qh64ee2a8r7)

[● 📈 Visualización de Métricas en Tiempo Real 14](#_tzigwku6y5g9)

[🚦 Sobrepasando Bajo Bandera Amarilla: Pruebas en Producción y Despliegues Canarios/Blue-Green 15](#_7yu59vjtwr87)

[● ⚠️ Riesgos y Beneficios de las Pruebas en Producción 15](#_3cs3qgy4egxj)

[● 🔄 k6 en Despliegues Canarios y Blue-Green 15](#_1sy34qn9kgkv)

[🌍 Vueltas Rápidas: Casos de Uso Avanzados 16](#_4aozgak1vzq6)

[● 🌍 Pruebas Distribuidas y Geolocalización 16](#_woobjulrcau2)

[● 🧩 Extensiones, Plugins y Más 16](#_6vgut8kp3lk7)

[🏁 Cruzando la Línea de Meta: Puesta en Práctica Integral y Q&A 17](#_8erfwvnhcnl8)

[● 🛠️ Creación de un Proyecto Completo Desde Cero 17](#_xk5qkq9dh2jc)

[● ❓ Resolución de Dudas y Discusión Abierta 17](#_46m47wpqiw5w)

# 

## 🏎️ Bienvenida e Introducción

Bienvenidos al workshop "**De 0 a 100: Performance Testing y k6 a Máxima Velocidad**", un completo viaje a través del apasionante mundo del rendimiento en el desarrollo de software. Este documento tiene como objetivo servir de hoja de ruta y recurso de consulta a lo largo de las 18 horas del workshop. **Y DE FORMA GRATUITA, las 18 horas quedarán grabadas y disponibles para que la consultes cuando quieras.**

#### ¿Por qué es Importante el Performance Testing?

En el entorno competitivo actual, donde cada milisegundo cuenta, asegurar que tus aplicaciones sean robustas y eficientes es más crucial que nunca. Fallar en el rendimiento podría costar no solo dinero sino también la confianza de tus usuarios.

#### ¿Qué Puedes Esperar Aprender?

Este workshop está diseñado para llevarte desde los conceptos básicos hasta las técnicas más avanzadas en Performance Testing utilizando k6. No solo te familiarizarás con la teoría sino que también pondrás manos a la obra, aplicando lo aprendido en escenarios prácticos.

### @Robert Ariechi CGTS

### **📑 Sobre Este Paddock Digital**

¡Pilotos y copilotos, bienvenidos! ¿Qué sería de una carrera sin un mapa del circuito y las estadísticas clave del coche? Ahí es donde entra este documento de Google Drive.

#### ¿Por qué necesitas este documento?

Este no es un simple PDF para descargar y olvidar en alguna carpeta de tu ordenador. Es tu paddock digital, donde encontrarás el desglose de cada vuelta (módulo), desde las estrategias de calentamiento hasta el plan para cruzar la línea de meta en primer lugar.

#### ¿Cómo Navegar en Este Paddock Digital?

* **Paneles de Control**: Cada sección está diseñada como un panel de control que te guía a través de cada tema del workshop.
* **Indicadores y Alertas**: Además de la teoría, encontrarás enlaces a recursos, ejemplos de código, y "alertas" con tips y best practices.
* **Cuaderno de Bitácora**: Hazlo tuyo. Utiliza los espacios para tomar notas o añadir preguntas que se disparen durante el workshop.

### 

### **💬 Participa en la Carrera: Comentarios y Preguntas**

Este paddock digital no es una vía de una sola dirección; es un circuito lleno de oportunidades para el diálogo y la clarificación. **¿Tienes dudas o insights que compartir?** Utiliza la función de comentarios de Google Drive para dejármelos. Para hacerlo, simplemente selecciona el texto al que te quieres referir y presiona **Ctrl + Alt + M**. No eres solo un espectador, eres parte del equipo de carreras. Tus comentarios y preguntas serán contestados por mí, para asegurarnos de que todos lleguemos a la línea de meta con el máximo rendimiento posible.

## **🧰 Requisitos de Boxes: Preparativos antes de la Carrera**

### 🛠️ Configuraciones Previas

Antes de empezar con el workshop, asegúrate de tener listo el siguiente "setup" para que todo fluya como aceite en motor:

#### 💻 Entorno de Desarrollo

* Instala [Node.js](https://nodejs.org/) y [npm](https://www.npmjs.com/).
  + [Instalación de Node.js en Windows | paso a paso](https://www.youtube.com/watch?v=czFj5zoI5uc)
  + [Instalar NodeJS y NPM en MacOS](https://www.youtube.com/watch?v=PJojM__8bBI)
* Configura un editor de código de tu elección, como [VSCode](https://code.visualstudio.com/).
  + [🔴 ¿Cómo DESCARGAR e INSTALAR VISUAL STUDIO CODE? ➡️ En Windows | ✅ TUTORIAL PASO A PASO 2022](https://www.youtube.com/watch?v=Hs49jA6FizM)

#### 📦 Herramientas Específicas

* [Instala k6](https://k6.io/docs/getting-started/installation/): El protagonista de nuestro workshop.
  + [Installing k6 in 1 minute](https://www.youtube.com/watch?v=PrqxyxSEib8)

#### 🗄️ Repositorio y Código

* Clona el repositorio del workshop, que incluirá ejemplos de código y scripts.
* Asegúrate de tener acceso al documento compartido de Drive donde estarán las diapositivas y otros recursos adicionales.

#### 🌐 Acceso a Internet

* Una conexión estable para descargar paquetes y seguir el workshop en tiempo real.

#### 🔐 Credenciales y Permisos

* Si se van a hacer pruebas en un entorno específico, asegúrate de tener los permisos y credenciales necesarios.

Con todo esto preparado, estarás listo para salir a la pista y apretar el acelerador a fondo.

### 📚 Materiales y Documentación

Para seguir este workshop de manera efectiva, es recomendable tener acceso a los siguientes recursos:

#### 📖 Documentación Oficial

* [k6](https://k6.io/docs/): Guía definitiva para todo lo relacionado con k6.
* [Grafana](https://grafana.com/docs/): Documentación para personalizar y entender los dashboards.

#### 📚 Libros y Artículos

* “[Introducción a las Pruebas de Sistemas de Información](https://drive.google.com/file/d/1Y3TeqFqq1Od8h08L8nqi-rjGrkL3vVF9/view?usp=sharing)” de Federico Toledo.
* [Artículos de Medium](https://medium.com/tag/performance-testing) sobre Performance Testing.
* [Artículos de Medium](https://medium.com/tag/k6) sobre k6.

#### 📹 Videos y Webinars

* [Canal de YouTube de k6](https://www.youtube.com/channel/UC05yEjg-PLs96DgpYBdFQVw): Tutoriales, webinars y otros recursos en video.

#### 🛠️ Repositorios y Código Ejemplo

* [Ejemplos de k6 en GitHub](https://github.com/grafana/k6/tree/master/examples): Repositorio con ejemplos para arrancar tus pruebas.
* [Repositorio del workshop](https://github.com/jwcastillo/0_to_100_k6/tree/main)

#### 📋 Plantillas y Scripts

* [Plantillas de pruebas de carga en k6](https://github.com/jwcastillo/0_to_100_k6/tree/main): Scripts predefinidos que te ayudarán a arrancar rápidamente.

#### 📊 Dashboards Predefinidos

* [Grafana Dashboards para k6](https://grafana.com/grafana/dashboards?search=k6): Dashboards listos para importar y empezar a visualizar tus pruebas.

Con estos recursos en tu pit box, estarás más que listo para enfrentar cualquier desafío de rendimiento que se te presente. ¡Vamos a darle gas! 🏎️💨

### 🧠 Conocimientos Necesarios

Para poder sacar el máximo rendimiento de este workshop, es recomendable tener cierta experiencia o familiaridad con los siguientes aspectos:

#### 🖥️ Fundamentos de Programación

Conocimientos básicos de programación, preferiblemente en JavaScript.   
Pero **no te preocupes** si no tienes experiencia en programación. **¡Este curso está diseñado para que aprendas sobre la marcha!**Es por ello que te dejo unos enlaces interesantes que puedes comenzar a ver para prepararte.   
  
**Enlaces interesantes:**

* [¿Por qué cuesta tanto aprender a programar?](https://tzuzulcode.com/por-qu%C3%A9-cuesta-tanto-aprender-a-programar-634cc29a0b15)
* <https://platzi.com/cursos/programacion-basica>
* [JAVASCRIPT Desde Cero: Primeros Pasos en una hora](https://www.youtube.com/watch?v=6cnFl9aHD5Y)[[MoureDev by Brais Moure](https://www.youtube.com/@mouredev)]
* [Aprende JavaScript en 60 minutos](https://www.youtube.com/watch?v=bmGPv687toc&t=2904s)[[deivchoi](https://www.youtube.com/@deivchoi)]
* [👩🏻‍💻 ¿Qué es JAVASCRIPT? | 🚀 ARMÁ TU PRIMER PROGRAMA | ⭐ Curso JAVASCRIPT DESDE CERO #1](https://www.youtube.com/watch?v=x5YUu0eUc8s&list=PLQxX2eiEaqbwnzKnmqHDl0rkRvp_T7Q_W)
* [Curso gratuito de JavaScript 2023](https://tzuzulcode.com/curso-gratuito-de-javascript-2023-1868a23ae18e)
* [entendiendo-javascript](https://medium.com/entendiendo-javascript)

**Disclaimer:** No tengo ninguna relación con los creadores de los enlaces y recursos recomendados, más allá de ser un consumidor más de sus conocimientos. Los comparto porque creo que son útiles para tu aprendizaje.

Continúa con el curso; cada etapa que superes te permitirá adquirir nuevas habilidades y seguir avanzando. 🚀

#### 🌐 Conceptos Básicos de Red

* Entender cómo funciona una petición HTTP/HTTPS.
* Conocimiento general de APIs y servicios web.

Me gustaron mucho estos videos:

[🔴 ARQUITECTURA CLIENTE SERVIDOR | 🌎 ¿Cómo FUNCIONA LA WEB? | ✅ PROGRAMACIÓN WEB + HTML + CSS 2022](https://www.youtube.com/watch?v=lC6JOQLIgp0)  
[⭐PROTOCOLO HTTP 🖥️ Requests y Responses con: GET, POST, PUT, PATCH y DELETE | DESARROLLO WEB 🌎](https://www.youtube.com/watch?v=l2MihYAj0Iw)[⭐¿Qué son las APIS? ¿Para qué sirven? 🖥️EJEMPLO PRÁCTICO con la POKEAPI | DESARROLLO WEB 🌎](https://www.youtube.com/watch?v=IwnIxk8DdHs)

[What is HTTP?](https://www.youtube.com/watch?v=SzSXHv8RKdM)

#### 🛠️ Herramientas de Desarrollo

* Familiaridad con la línea de comandos.
  + [CURSO rápido de TERMINAL y línea de comandos - Descubre todo lo que puede hacer](https://www.youtube.com/watch?v=iZtojEmq--c)
  + [Comandos Básicos e Intermedios CMD](https://www.youtube.com/watch?v=erKosEQaaFc)
* Experiencia mínima con sistemas de control de versiones como Git.
  + [Curso de GIT y GITHUB desde CERO para PRINCIPIANTES](https://www.youtube.com/watch?v=3GymExBkKjE)

#### ⚙️ Introducción a DevOps

* Conceptos generales sobre integración y entrega continua (CI/CD).

Me gustaron mucho estos videos:

[¿QUE ES CI/CD?](https://www.youtube.com/watch?v=6eRkCnFhHRg)

[¿Qué es DevOps y CI/CD?](https://www.youtube.com/watch?v=xm1psJEFFIY)

[🧪CI/CD para TODOS los Testers! (AL GRANO) - Continuous Integration y Deployment🚀| UPEX GALAXY](https://www.youtube.com/watch?v=lE6NEiF8E6Y&list=PLLYWsphuMYKs2qyOvQK8MGYthV3iX-0xE)

#### 📊 Interés en el Performance Testing

* Mientras no es un requisito técnico, la curiosidad y el deseo de mejorar el rendimiento de aplicaciones web será tu combustible durante todo el workshop.

# EN LA PISTA

Carrera del viernes 08-09

## 🚦 ¡Enciende los Motores!: Introducción al Performance Testing y k6

### 🤔 ¿Por qué nos Importa el Performance Testing?

### 🧪 Tipos de Pruebas: Carga, Estrés, Resistencia y Más

### 🛠️ Introducción y Razones para Usar k6

### 🚀 Primer Arranque y Configuración Básica con k6

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Imagen de la pantalla de un videojuego

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Latencia:** Tiempo que pasa desde que se solicita una acción hasta que se realiza.

**Escalabilidad/Elasticidad:** Aguante de la carga que se solicita y pueda ser soportada aun mas de la carga esperada, preparase para la demanda inesperada.

**Reliability/Availability:** Incrementar la confianza en la aplicación en cuanto a disponibilidad y confiabilidad de las aplicaciones en su rendimiento con servicio ininterrumpido.

**Resilience:** Evaluar y optimizar la infraestructura.

**Monitoreo y observabilidad:** Controlar los valores de las mediciones en tiempo real y anticiparnos a los problemas y optimizar además de prevenir futuras incidencias.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

La nube es infinita: Se puede continuar escalando o funcionando a menos que haya un fallo todo va a ir cayendo.

Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Las pruebas de performance como tal buscan validar comportamientos, sin embargo, consisten en evaluar como se comporta un sistema bajo ciertas condiciones de tráfico esperado o inesperado.

Aspectos claves a considerar en las pruebas de carga:

1. **Los usuarios** : Cantidad de peticiones de usuarios que nuestro aplicativo puede soportar.
2. **El rendimiento:** Cuanta potencia o capacidad tiene nuestro aplicativo potencia se refiere a que si responde 300 usuarios por segundo esto es aceleración en cuanto a velocidad.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Aceptar errores en pruebas de carga:** 9´s estándar para medir la (4-9´s) ejemplo: 99.62% este valor tiene cuatro nueves, estos 9´s se refieren a un estándar en el mundo de la tecnología, para medir la disponibilidad y la fiabilidad de un sistema o aplicativo por ejemplo en una carrera se representa como la cantidad de vueltas sin fallos ni accidentes.

\*Cuantos mas 9´s consigamos mas ciclos habremos completado sin fallas ni incidencias en nuestros aplicativos lo que significa más fiabilidad y disponibilidad.

**Como se desglosan los 4 nueves:**

-Tenemos un 99% de fiabilidad.

-podría decirse que se tendrán 999 vueltas de 100 es traducido en tiempo si esto se representa en días del año se podría decir que hay indisponibilidad de 3.95% en el año.

-Y los 4 nueves una sola de 100 tiene error al año o indisponibilidad.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente con confianza media

**-Las pruebas de carga**: como tal no son para estresar un aplicativo, es para entender los limites de un sistema para que cuando se enfrente a un caso real lo haga de manera eficaz y confiable.

Texto

Descripción generada automáticamente

Imagen que contiene coche, camioneta, camión, hombre

Descripción generada automáticamente

**-Las pruebas de performance:** son las pruebas mas importantes mientras que las pruebas de carga son parte de las pruebas de performance.

Se pueden realizar pruebas de tiempos de respuesta, estabilidad, etc.

**-Las pruebas de carga:** son para poder ver como se comporta nuestro aplicativo al someterse a una carga de usuarios o transacciones simultaneas ahí evaluamos factores como tiempos de respuesta, la estabilidad, etc. .

Abarca varios ámbitos, las pruebas de carga o Load es validar que tantos usuarios o transacciones soporte el aplicativo de forma simultánea su objetivo es identificar cuellos de botella antes que se degrade su performance.

Su ámbito es mucho mas especifico carga y performance.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Se pueden realizar pruebas de comportamiento, esto con K6, para validar el performance.

Realizándose pruebas unitarias de carga, los microservicios se pueden realizar pruebas con cada uno de los endpoints de forma individual para conocer sus diferentes valores y límites.

En producción un servicio con 5 endpoint diferente se debe de probar una prueba unitaria para validar mover el performance para las etapas tempranas en desarrollo.

Con cada cambio se puede ir modificando para ver si mejoro o empeoro el performance.

Microservicios es como implementamos y API extracción de cómo se expondrá la comunicación al usuario.

Las API RESTful se utilizarán para realizar dichas pruebas, aunque pueda que existe otras formas de comunicación.

Se pueden realizar pruebas de integración en performance y estas consisten en que vamos recorriendo todo el flujo de la información del aplicativo similar a las pruebas E2E desde punto A al punto B.

Se puede realizar por ejemplo pruebas de Login, ingreso a home, etc. con pruebas de integración igualmente.

Realizar pruebas de performance y de funcionalidad son diferentes y esto radica en que el Tester automatizado o manual valida funcionalidades que el cliente ha solicitado, el de performance ve esta parte funcional, pero a nivel de arquitectura y su comunicación y disponibilidad.

Se limita a probar caminos felices(Happy Path) en casos muy específicos prueba que se puedan dar errores.

Captura de pantalla con letras blancas

Descripción generada automáticamente

-Tanto el tiempo del frontend como el tiempo del backend juegan un papel muy importante para algo llamado experiencia de usuario UX.

-**Tiempo de frontend:** se refiere al tiempo en que una pagina web renderiza(Dibujarse o mostrarse) completamente al usuario, puede ser también una aplicación incluye la carga de los recursos visuales, los scripts, los estilos si estos recursos tardan mucho en cargar el usuario puede percibir que la aplicación es lenta afectando negativamente su experiencia.

**-Tiempo de backend:** esto es el tiempo que tardara una solicitud en llegar a nuestro servidor local o en la nube y este en enviar la respuesta, cuando están demasiado cargados estos pueden generar tiempos prolongados de respuesta afectando al tiempo total de carga de una página o ventana del aplicativo o sistema.

A medida que aumentan los usuarios concurrentes o que están entrando al sitio web, el tiempo de respuesta aumenta.

Porque todo se degrada o tiene un limite o punto de fallo o quiebre pues hace un cuello de botella hasta que colapsa, la idea detrás de las pruebas de carga o de performance es mover lo mas que se pueda a la derecha del grafico anteriormente visto.

Que se aguanten más usuarios concurrentes con un bajo response time.

Interfaz de usuario gráfica

Descripción generada automáticamente

Son muchas piezas que hay que esperar que cumplan y que lo hagan con calidad como hay varias piezas de software involucradas se tiene un riesgo mayor que alguna de estas puede fallar.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Los **SLO´ s** (Acuerdos de servicio objetivos) : son estándares que indican los estatus de como se va con el objetivo proactivo.

Si por ejemplo tenemos un sistema que tarda 5 segundo en responder se puede llegar a un acuerdo de servicio objetivo que se requerirá un tiempo de respuesta de 15 segundos.

Ese sería nuestro SLO.

Hay otro termino que se llama **SLA´s**: es el que el equipo revisa o acuerdo con la persona que desea o quiere llegar.

El otro sería el **SLI´s**: objetivo diariamente de revisión.

Son las tres métricas que permiten evaluar la confiabilidad es algo que no es del performance engineer sino del equipo.

Imagen que contiene Diagrama

Descripción generada automáticamente

En vez de ignorar las fallas nos preparamos antes que ocurran con diferentes estrategias y formular soluciones.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Estas empresas han puesto las expectativas altas, por las velocidades de búsquedas y demás, generan expectativas muy altas por tanto si se busca en otros sitios las búsquedas son tardadas para poder enfrentar los tiempos de estos gigantes tecnológicos.

Tik tok ha dañado nuestra capacidad de atención, pues se pasan cada 3 segundos de ver un video.

**PARAMETROS DE PRUEBAS.**

**Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente**

**-Usuarios virtuales (VU´s):** Son usuarios no reales o sintéticos también están los RUM´s Real user monitoring, se puede utilizar para pruebas de carga usuarios sintéticos usando una herramienta como K6 o JMeter o puede realizarse una prueba de carga con usuarios reales poniéndonos de acuerdo con varios colaboradores para que puedan acceder a una URL en un determinado espacio de tiempo.

Esta última técnica se realiza por parte de pentesting en seguridad para ir a la casa de bugs.

Esto mismo aplica en pruebas de performance y carga.

**-Iteraciones:** Son el flujo repetitivo que realiza un usuario virtual por ejemplo las compras en un carrito el cual el usuario va a la pagina y realiza el ciclo de seleccionar y hacer click para el carrito de compras.

**-Throughput:** Tiene que ver con la velocidad y rapidez de cuantos usuarios pasan por un punto especifico en una cantidad determinada de tiempo.

Diferencias entre iteraciones y request, una petición ejemplo un usuario debe hacer una compra en un sitio en línea.

1-Usuario se loguea.

2-Usuario entra en la página.

3-Usuario selecciona un articulo

4-Usuario paga el producto.

En este flujo el usuario ha realizado 4 iteraciones y unos 5 request.

**-Flujos de usuario:** Camino o estrategia que el usuario va a seguir, o diferentes combinaciones por ejemplo usuarios que entran con Login a la pagina o solo se loguean cuando van a pagar algún articulo seleccionado.

Se pueden probar uno a uno a todos si es posible dichos flujos.

**-Perfil de carga:** Son las interacciones de los usuarios primero llegaran 200, luego 1000 y así sucesivamente este comportamiento de ir y venir y llegar a un pico.

**-Duración:** Tiempo de duración de la prueba y las etapas que se consideraran.

**COMO SIMULAR CARGAS.**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**-Pruebas de carga basadas en el protocolo:** De estas podemos mencionar.

-Pruebas de API específicas.

-Pruebas de extremo a extremo.

**-Pruebas de carga basadas en navegadores** para imitar el comportamiento de un usuario con sus pausas y su interacción más real con el aplicativo.

**-Pruebas Hibridas** estas mezclan como esta funcionando la carga en el navegador + el backend.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Pequeña prueba tipo Smoke test para ver que todo este en óptimas condiciones de funcionamiento y del script de pruebas igualmente, si esta prueba fallara se detiene la batería de pruebas en esta prueba.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Es la prueba de carga promedio esta se hace conociendo mas o menos en promedio cuantos usuarios se tienen en periodos de tiempo en la aplicación la idea con estas pruebas es conocer los usuarios que utilizan el aplicativo o herramienta de software y como es su comportamiento si es de una sola vez o llega en olas no es estresar, etc.

-Ejemplo 10 VU´s X segundo.

Esto es diferente al throughput que se habló anteriormente, cuando involucra más de un request ahí se tendrá que hacer o un average load test o un throughput con lo cual el flujo puede cambiar.

PU de un componente que tiene un solo request.

Si se inyecta 100 usuarios por segundo el throughput puede hacer estos 100 por segundo, pero, dependiendo como se vea se tendrá iteraciones x segundo y request x segundo sesgado, no es lo mismo el home que el Login o seleccionar producto en cuanto a su tiempo de respuesta.

Cuando se mide el throughput que es el caudal de velocidad con el cual los usuarios están entrando y saliendo a ese flujo de velocidad efectiva se llama throughput.

La prueba de carga busca saber como se comporta nuestro sistema con una carga promedio y ver sus métricas para ver como esta en estado estable.

Interfaz de usuario gráfica

Descripción generada automáticamente

Es una prueba de carga en la que buscamos un punto en el cual el sistema pueda degradarse el degradamiento del sistema es lo que llamamos estrés.

La diferencia real es que si nuestro promedio es 50 VU´s x segundo colocamos mas hasta que los tiempos de servicio comienzan a cambiar y detectamos un posible cuello de botella dependiendo el servicio que se esta probando, hasta que cause estrés.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Esta se conoce como una prueba de resistencia involucra como tal ver como se comporta a lo largo del tiempo con carga promedio o menos de la carga es larga en el tiempo y aporta en cierto momento memory lead se necesitan ventanas de tiempo mas altas y se ve en monitoreo como línea hacia arriba como estar hasta el tope y reinicia el sistema.

Estas pueden ser muy largas por los tiempos que se pondrán para realizarse.

Gráfico

Descripción generada automáticamente con confianza media

Pruebas de aceleración inmediata de nuestro aplicativo, y como viaja esto en nuestro servicio por ejemplo un pico elevado y ver como se comporta con esto y saber si no se caerá debido a esta aceleración de peticiones el sistema debe escalar, aguantar y luego desescalar y volver a punto estable si en alguno de estos momentos nuestros sistema se cae o entra en un estrés hay que validar para que aguante el pico o spike.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Esta prueba se realiza para saber el punto en el cual se rompe el sistema, estrés es cuando se degrada el sistema o aplicativo, si continuo con este estrés llegara un punto en que se rompe o hay errores, ahí entra el criterio dependiendo que se cayo si estoy probando un sistema E2E y se cae el home no se necesita seguir probando mientras que, si se cae la BD, podemos seguir y ver hasta que punto puede afectar la caída.

Imagen que contiene interior, techo, cocina, tabla

Descripción generada automáticamente

**COMO SE HACIAN LAS PRUEBAS ANTERIORMENTE.**

Interfaz de usuario gráfica

Descripción generada automáticamente

**COMO SE HACEN PRUEBAS MODERNAMENTE.**

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Pruebas de todos los endpoints de forma progresiva por componentes de forma sencilla.

Al agregarse a un flujo CI/CD se pueden realizar pruebas regresivas de performance o un CT continuous testing, con esto se puede armar un plan de confianza o plan de disponibilidad.

Una captura de pantalla de una computadora

Descripción generada automáticamente

El la antigua forma las pruebas de performance eran realizadas por un especialista en pruebas de performance ahora todos pueden realizar pruebas de performance desde desarrolladores, QA´s, etc., es un esfuerzo de equipo.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

**K6** Es una herramienta de código abierto, y es una plataforma de paga también para poder realizar pruebas de performance a cargo de grafana.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En una sola maquina se pueden virtualizar hasta 40000 VU´s con lo cual es una herramienta muy sólida en el mercado, la otra herramienta es JMeter con java, aunque es sabido que utiliza mucha memoria.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Los script de K6 están hechos en JavaScript, pero mediante una rutina este los transforma internamente en código de GO pues k6 está desarrollado en Golang.

Tiene algunas limitaciones con librerías de NodeJS y algunas librerías de buscadores.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

### 

### \*Puede que si se puede caer el internet al hacer cargas altas de VU´s de ahí porque se debe validar la latencia preferiblemente desde el sitio de trabajo no desde la casa o en modo remoto pues esta tendrá sus variaciones.

### Imagen que contiene Texto Descripción generada automáticamente

### Por cada usuario se ejecutarán “X” cantidad de iteraciones, en uno de los casos se multiplica y en otro se divide dependiendo de las pruebas que se busquen cubrir.

### **\*La mejor forma de poder correr K6 es a través de un Docker embebido en la computadora que tengamos, es la mejor forma de ejecutarse en un enfoque de trabajo.**

Ramping escalera arriba, estamos hablando de throughput es constant arrival y ramping arrival.

Diagrama, Diagrama de Venn

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Esto aplica también a realizar test unitarios desde Dev, puesto que una prueba de un solo usuario se convierte en una prueba funcional probando E2E.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

El concepto de caos engineer lo promovió Netflix para poder inyectar errores o fallas.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**VISUALIZACION DE LOS RESULTADOS.**

Diagrama

Descripción generada automáticamente

visualización de resultados en terminal.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Visualización de reporte de resultados.

Interfaz de usuario gráfica

Descripción generada automáticamente

Reporte de prueba de carga o Load Test

Gráfico

Descripción generada automáticamente

\*En el reporte la línea gris son los VU´s que estamos inyectando a la prueba.

\*La línea amarilla es el performance que vemos que llega a un punto en el cual va bajando y es errático.

\*La línea azul es el tiempo el cual comienza a subir en cuanto a respuesta latencia.

\*La línea roja son los errores, cuando estos empiezan a subir esta degradando el performance

**GRAFANA CLOUD**

Interfaz de usuario gráfica, Teams

Descripción generada automáticamente

### Captura de pantalla de computadora Descripción generada automáticamente

### Interfaz de usuario gráfica, Aplicación Descripción generada automáticamente

### Diagrama Descripción generada automáticamente

### Interfaz de usuario gráfica, Aplicación Descripción generada automáticamente

### Una caricatura de un hombre con lentes oscuros Descripción generada automáticamente con confianza baja

### Grafana es una herramienta de observabilidad en la cual se ve la traza para ver los puntos donde fallo.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Captura de pantalla de un celular

Descripción generada automáticamente

### Motor de un carro Descripción generada automáticamente con confianza media

Carrera del viernes 15-09

## 🛠️ Manos al Volante: Escritura y Análisis de Scripts con k6

### 📝 Escenarios Básicos y Avanzados

### ✅ Uso de Check, Umbrales y Asserts

### 📊 Lanzamiento de Pruebas y Análisis de Resultados

Imagen de la pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

-Como herramienta para uso diario se recomienda utilizar la herramienta k6 desde Docker para poder evitar cuestiones de configuración.

-Utilizar un gestor de paquetes para poder gestionar de mejor forma nuestros recursos el recomendado es:

https://asdf-vm.com/

Por ejemplo, para instalar k6 seria:

$ asdf install k6 latest

$ yay -S k6

Cuando se necesita hacer performance en ambientes productivos no es recomendable utilizar un ordenador por la latencia además que cada ordenador es diferente, con diferentes procesadores los cuales afectan como será el rendimiento de las herramientas, el ordenador es solo para validar el script y poder probarlo antes de poder correrlo en una batería de pruebas.

-Se deben hacer las pruebas preferentemente desde la oficina puesto que en la mayoría de los casos el internet residencial no es muy bueno y esto puede quitar mucha calidad a nuestras pruebas de performance.

https://github.com/grafana/test.api.k6.io

-Toda ejecución que se haga desde una máquina virtual o física es una **ejecución local.**

-Cuando utilizamos el servicio de K6 cloud entonces estamos en una **ejecución en modo.** **remoto.**

-Cuando queramos correr nuestros archivos locales que están en carpeta remota vamos a hacerlo así.

$k6 run path/file/name.js

-Cuando enviamos un get tenemos todo en la url que llamamos mientras que en Post debemos enviar el body para traer información.

-El método delete funciona como un get al hacer la llamada en http.

-Para generar un archivo de tipo json de nuestras ejecuciones debemos hacerlo de la siguiente manera.

$k6 --out json=my\_test\_result.json run http\_getV2.js

-La función check es menos pesada que los assertions aunque están presentes igualmente en K6

-Cuando agregamos funciones como sleep y la utilizamos para poder controlar los espacios entre ejecuciones(valores random) ya no tenemos pruebas unitarias sino una prueba con flujo.

- En las pruebas de performance hay algo que se llama think time o tiempo de pensamiento, tiempo que tarda el usuario en realizar una iteración.

Carrera del viernes 22-09

## 🚧 Pit-Stop Técnico: Integración de k6 en CI/CD

### 🔧 Integración con Jenkins, GitLab CI y Otros

### 🚨 Automatización de Pruebas: Umbral, Alertas y Mucho Más

### 🛠️ Práctica: Integración con una Herramienta de Pipeline

### 

Carrera del viernes 29-09

## 🌐 Adelantamientos Estratégicos: k6 Browser y Pruebas Híbridas

### 🌐 ¿Qué es k6 Browser? Ventajas y Configuración

### 🤝 Creación y Gestión de Pruebas Híbridas

### 

Carrera del viernes 06-10

## 🎯 Zonas de DRS: Performance Left Testing

### 📚 Teoría y Beneficios

### 🛠️ Implementación Práctica con k6

### 

Carrera del viernes 13-10

## 📊 ¡Monitor de Carrera! Dashboards en Grafana

### 🎨 Introducción y Configuración de Grafana con k6

### 📉 Diseño y Personalización de Dashboards

### 📈 Visualización de Métricas en Tiempo Real

### 

Carrera del viernes 20-10

## 🚦 Sobrepasando Bajo Bandera Amarilla: Pruebas en Producción y Despliegues Canarios/Blue-Green

### ⚠️ Riesgos y Beneficios de las Pruebas en Producción

### 🔄 k6 en Despliegues Canarios y Blue-Green

### 

Carrera del viernes 27-10

## 🌍 Vueltas Rápidas: Casos de Uso Avanzados

## 🌍 Pruebas Distribuidas y Geolocalización

## 🧩 Extensiones, Plugins y Más

### 

Rally desde viernes 27-10 -- 03-11

## 🏁 Cruzando la Línea de Meta: Puesta en Práctica Integral y Q&A

### 🛠️ Creación de un Proyecto Completo Desde Cero

### ❓ Resolución de Dudas y Discusión Abierta