Betclic
Pariez sur le meilleur

Betclic web app

Performance Test Plan

Version 1.0

28/06/2022

Michael De Madet - Nils Millot - Océane Renoux - Sid Ahmed Mekhiche

Table of Contents

Préambule 2

Architecture de l’application 3

Technos 3

Diagram d’architecture 3

Exigences du test 4

Environnement de tests 5

Planification des tests 6

Étapes des tests - scénarios 7

Rechercher un match 7

Faire un nouveau paris 7

Voir le résultats des matchs de foot 7

Jeu de données 8

Execution des tests 9

Préambule

Nous allons établir un plan de test pour l'application betclic.fr.

Il s'agit d'un site de paris sportif en ligne.

Sachant qu'il s'agit d'un site de paris sportif, on peut s'attendre à des pics d'utilisateurs durant les événements sportifs importants (par exemple les matchs de foot le soir, ou le week-end)

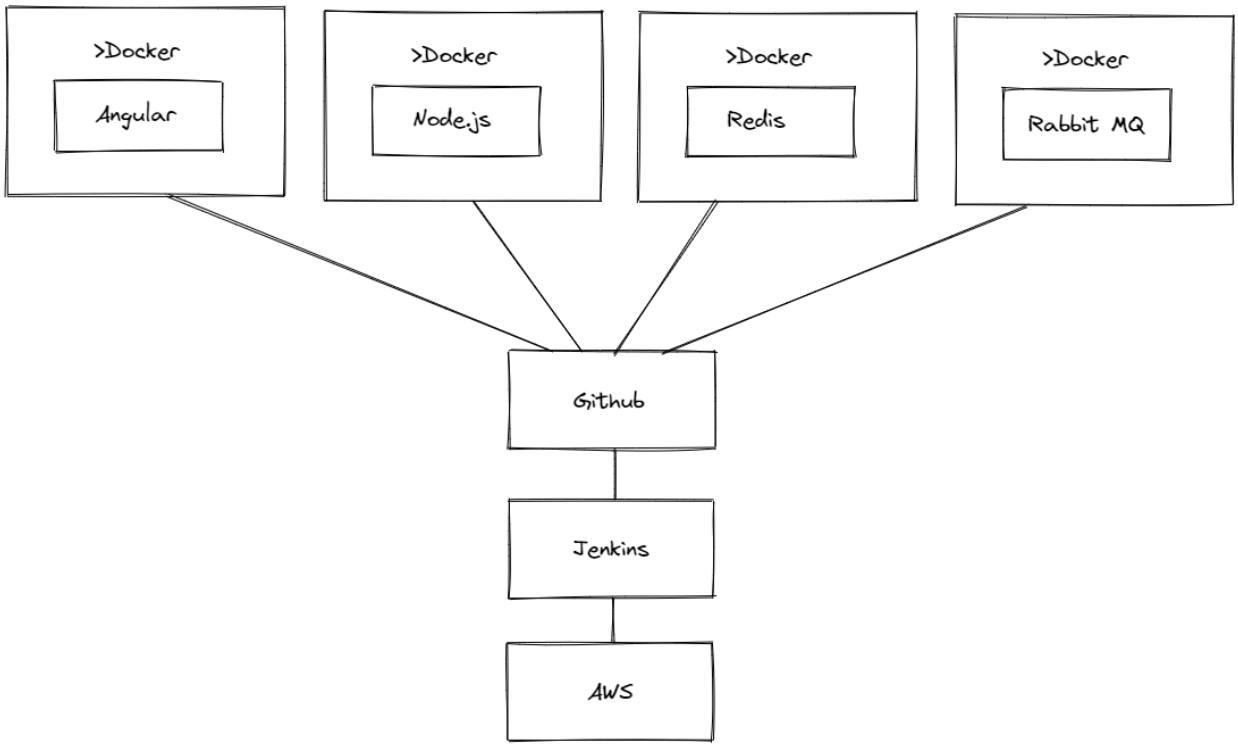
Architecture de l’application

Technos

Les technologies utilisés sur la web app betclic.fr sont :

- Angular
- Node.js
- Redis
- RabbitMQ
- AWS
- Docker
- Github
- Jenkins

Diagram d’architecture



Exigences du test

Sur cette application de paris sportif, on s'attend à avoir des arrivées massives d'utilisateurs au moment des gros événements sportifs, par exemple dans un match de foot à la fin du match.

C'est pour cela que l'on va réaliser des spike testing, pour simuler une arrivée massive d'utilisateurs sur une période courte, et voir si l'application tient le coup.

Business Transactions	**User Load**	**SLA/response times**	**Transactions per hour**
Rechercher un match	80,000	2 seconds	40,000
Faire un nouveau paris	100,000	3 seconds	50,000
Voir le résultat des match de foot	50,000	1 seconds	25,000

Environnement de tests

L’environnement de prod contient :

- RAM : 400Go
- CPU : 40
- OS : Linux (debian)

L’environnement de test contiendra :

- RAM : 100Go
- CPU : 10
- OS : Linux (debian)

Ce qui nous donne un coefficient proportionnel de 0.25, 100,000 utilisateurs en prod reviendra à 25,000 en environnement de test.

Planification des tests

Nous avons besoin de pouvoir tester des pics d'utilisateurs sur l'application, pour cela nous allons réaliser des spike test.

Le nombre maximum d'utilisateurs durant ces périodes de pic est estimé à 100,000 utilisateurs.

En ayant un coefficient de proportionnalité de 0.25, on planifiera nos tests pour 25,000 utilisateurs.

Les métriques que l'on souhaite surveiller seront le temps de réponse moyen, l'utilisation de CPU et de mémoire.

Le temps de réponse moyen ne devra pas dépasser 3s, l'utilisation de CPU et de mémoire ne devra pas dépasser 85%

Étapes des tests - scénarios

Rechercher un match

Step #	**Application Name: Betclic.fr** **Business Process Name:** Rechercher un match
1	Page d'accueil
2	Rechercher le match dans la barre de recherche
3	La page affiche les infos du match

Faire un nouveau paris

Step #	**Application Name: Betclic.fr** **Business Process Name:** Nouveau paris
1	Page d'accueil
2	Se connecter
3	Rechercher le match dans la barre de recherche
4	La page affiche les infos du match
5	Choisir l'équipe sur lequel parier
6	Paiement
7	Redirection page d'accueil

Voir le résultats des matchs de foot

Step #	**Application Name: Betclic.fr** **Business Process Name:** Voir ses paris en cours
1	Page d'accueil
2	Page "Live"
3	Cliquer sur le bouton "foot" pour filtrer le résultat
4	Page "Live" mis jour avec les matchs de foot uniquement

Jeu de données

- 25 000 utilisateurs (email + mot de passe)

Pour créer les données de tests, on utilisera la librairie faker.js.

On utilisera une base de données de tests, dans laquelle on insérera nos 25 000 faux utilisateurs.

Pour la partie paiement, on fera un mock de l'API stripe, pour simuler un paiement.

Execution des tests

Test Run	Date	Test Scenario Summary
Cycle 1 - Run 1	22h	Spike test - 1mn test with peak load
Cycle 1 - Run 2	22h05	Spike test - 2mn test with peak load
Cycle 1 - Run 3	22h10	Spike test - 3mn test with peak load
Cycle 2 - Run 1	23h	Spike test - 2mn test with peak load
Cycle 2 - Run 2	23h10	Spike test - 4mn test with peak load
Cycle 2 - Run 3	23h20	Spike test - 6mn test with peak load

	Test Details
Purpose	Le but du test est d'augmenter brutalement la charge sur des périodes courtes, jusqu'au pic de 25 000 utilisateurs, pour voir si l'application tient. Ces tests vérifient les métriques d'utilisation CPU/Mémoire et le temps de réponse moyen, en vérifiant que ces valeurs ne dépassent pas les limites imposées.
No. of Tests	6 (3 tests per cycle)
Duration	Ramp-up: 4000 VUser / 10 secondes Steady State: 40 000 VUser - 10mn Ramp-down: 8000 VUser / 10 secondes
Scripts	1. Rechercher un match 2. Faire un nouveau paris 3. Voir les résultats des matchs
Scenario Name	Spike Test Scenario
User Load / Volume	25 000 Vusers (Threads) Load
Entry Criteria	1. The code should be stable and functionally verified 2. Test Environment should be stable and ready to use 3. Test Data should be available 4. Test scripts should be ready to use
Validation Criteria	1. Le temps de réponse des pages doit êtres inférieur à 3 secondes 2. L'utilisation de CPU/mémoire doit être inférieur à 90%