

Cloud, conteneurs, répartition de charge et ingénierie des releases

Guide de la leçon · Guide du lab / de l'examen

● LearnWisej

Guide de la leçon

IIS, Kestrel, reverse proxies, conteneurs, sticky sessions, health checks, CI/CD, runbooks. Ce que cela signifie pour une équipe Wisej.NET avancée — et comment aborder ce module.



Enjeu de production

IIS, Kestrel, reverse proxies, conteneurs, sticky sessions, health checks, CI/CD, runbooks.

1. Le problème en production

Les applications Wisej.NET avancées doivent dépasser les postes des développeurs pour aller vers des environnements reproductibles, avec une discipline de configuration, des health checks, de la répartition de charge, un retour arrière et une responsabilité opérationnelle claire.

2. Concepts clés

Concept	Pourquoi c'est important
IIS	Le serveur web Windows qui héberge les applications Wisej.NET dans de nombreuses entreprises, avec des pools d'applications, des règles de recyclage et une prise en charge des websockets à configurer.
Kestrel	Le serveur web multiplateforme d'ASP.NET Core qui exécute directement une application Wisej.NET, généralement derrière un reverse proxy en production.
reverse proxy NGINX/Apache	Serveur frontal qui termine le TLS, transmet le trafic websocket et achemine les requêtes vers une ou plusieurs instances de l'application.
Azure/AWS	Plateformes cloud qui hébergent l'application sous forme d'app services, de conteneurs ou de machines virtuelles, chacune avec son propre modèle de montée en charge et de configuration.
Docker	Empaquette l'application et son runtime dans une seule image, pour que le même build s'exécute à l'identique sur un poste de développeur, un serveur de test et dans le cloud.
HealthCheck.json	La configuration de health check de Wisej.NET qui permet aux répartiteurs de charge et aux orchestrateurs de vérifier si une instance est prête à servir des sessions.
sticky sessions	Affinité du répartiteur de charge qui achemine chaque requête d'un utilisateur vers la même instance de serveur, indispensable parce que l'état de session vit dans la mémoire du serveur.
CI/CD	Pipelines automatisés de build, de test et de déploiement qui transforment chaque merge en une release reproductible plutôt qu'en une procédure manuelle sujette aux erreurs.

3. Déroulé de la leçon

Le déploiement fait partie de l'architecture

L'hébergement modifie le comportement de l'identité, des WebSockets, de la journalisation, de la configuration, du stockage de fichiers, des health checks et de la montée en charge. Les développeurs avancés conçoivent pour l'environnement cible suffisamment tôt pour éviter les surprises.

Une montée en charge consciente des sessions

Les applications Wisej.NET maintiennent un état riche côté serveur. Dans les environnements à charge répartie, l'infrastructure doit acheminer une session de manière cohérente ou utiliser un modèle de déploiement qui préserve le comportement des sessions. Les health checks permettent d'écarter les nœuds défaillants.

Choisir une cible de déploiement

Une application Wisej.NET s'exécute partout où ASP.NET Core s'exécute ; le choix est donc guidé par l'organisation plutôt que par le framework. IIS sous Windows convient aux équipes disposant déjà d'une exploitation Windows, de la gestion des pools d'applications et de l'authentification Windows. Kestrel

derrière un reverse proxy NGINX ou Apache convient aux hôtes Linux et confie au proxy la terminaison TLS et la transmission des WebSockets. Azure App Service et ses équivalents AWS suppriment la gestion des serveurs, mais ajoutent une configuration propre à la plateforme, des règles de montée en charge et une intégration de la journalisation.

Les conteneurs changent l'unité de déploiement plutôt que l'hôte. Une image Docker regroupe l'application et son runtime pour que le même artefact s'exécute sur un poste de développeur, un cluster de test et en production. L'image doit tout de même être hébergée quelque part, et cet hôte a toujours besoin des réglages de WebSocket, de health check et d'affinité de session dont dépend l'application.

Quelle que soit la cible, vérifiez trois points avant de vous engager : les WebSockets traversent chaque couche entre le navigateur et l'application, l'endpoint de health check est accessible par l'infrastructure, et l'application voit l'adresse et le schéma réels du client via les en-têtes du proxy.

Répartir la charge d'une application liée aux sessions

Chaque session Wisej.NET vit dans la mémoire d'un seul processus serveur. Un répartiteur de charge qui distribuerait les requêtes entre les nœuds sans affinité enverrait la deuxième requête d'un utilisateur à un nœud qui n'a jamais entendu parler de lui, et l'application démarrerait une nouvelle session. Les sticky sessions, où le répartiteur achemine tout le trafic d'un client vers le même nœud, sont donc une exigence et non une optimisation. Les notes de répartition de charge du lab l'énoncent explicitement à l'intention de l'équipe d'exploitation.

Les fonctionnalités temps réel ajoutent une seconde contrainte. Le push serveur repose sur un WebSocket qui reste ouvert entre le navigateur et le nœud qui détient la session ; le répartiteur et tout proxy doivent donc maintenir en vie les connexions de longue durée sans les couper pour inactivité. Monter en charge signifie donc ajouter des nœuds qui détiennent chacun une part des sessions, et réduire la capacité signifie vider un nœud en douceur pour que ses sessions puissent se terminer au lieu d'être coupées.

Les health checks alimentent le répartiteur. Dans le chemin d'échec du lab, le nœud B échoue à son health check, le répartiteur écarte de ce nœud les nouvelles sessions, et les utilisateurs du nœud A ne remarquent rien. Documentez ce qu'il advient des sessions qui se trouvaient déjà sur le nœud B, car c'est la première question que posera l'exploitation.

Pipelines, smoke tests, retour arrière et runbooks

Un pipeline de release transforme chaque merge en une même séquence : build, exécution des tests, production d'un artefact versionné, déploiement dans un environnement de préproduction, exécution des smoke tests et promotion. La valeur réside dans la répétition. Quand chaque release suit le même chemin, un échec est une différence à analyser plutôt qu'un mystère. L'artefact est versionné, si bien que le tableau de bord des releases peut montrer que la 2.4.2 est en cours de déploiement et que la 2.4.1 est celle qui tournait auparavant.

Les smoke tests sont les quelques vérifications qui prouvent que l'application déployée est vivante : la page de connexion s'affiche, une session démarre, le health check renvoie un état sain, une requête connue renvoie des lignes et un WebSocket se connecte. Ils s'exécutent en quelques minutes et contre l'environnement réel. Le plan de retour arrière décrit ce qui se passe quand ils échouent. Il nomme l'artefact précédent, la commande qui le redéploie, la personne autorisée à la lancer et la façon de confirmer que cela a fonctionné.

Le runbook rassemble tout cela dans le document que lit un ingénieur d'astreinte à deux heures du matin. Le runbook du lab se termine par le retour arrière, car une procédure de release qu'on ne peut pas inverser n'est pas terminée.

La configuration comme contrat

Le même build doit s'exécuter en développement, en test et en production ; rien de spécifique à un environnement ne peut donc vivre dans le code. Les chaînes de connexion, les URL du fournisseur d'identité, les comptes de stockage, les feature flags, les niveaux de log et les hôtes autorisés relèvent de la configuration, fournie par l'environnement et lue au démarrage. Le tableau de configuration des environnements du lab liste chacun de ces paramètres, sa source dans chaque environnement et la personne responsable de sa modification.

Wisej.NET lit ses propres paramètres dans `Default.json`, et le comportement du health check dans `HealthCheck.json`. Les paramètres de l'application suivent le modèle de configuration d'ASP.NET Core ; une variable d'environnement ou un fichier de secrets monté peut donc remplacer une valeur d'un fichier sans toucher à l'image. Les secrets n'apparaissent jamais dans les fichiers de configuration versionnés. Ils sont injectés au moment du déploiement depuis un coffre-fort ou le magasin de secrets de la plateforme.

- Le même artefact peut-il être déployé dans chaque environnement en ne changeant que la configuration ?
- Chaque paramètre du tableau a-t-il un responsable et une source pour chaque environnement ?
- Une fuite du dépôt exposerait-elle un secret ?



Réflexe de production avancé

Ne vous arrêtez pas quand l'écran paraît correct. Expliquez à qui appartient l'état, les implications de sécurité, les chemins d'échec et le comportement en production avant de considérer le module comme terminé.



Objectifs d'apprentissage

À la fin de ce module, vous saurez :

- ✓ Choisir les cibles de déploiement parmi IIS, Kestrel/reverse proxy, Azure, AWS et les conteneurs.
- ✓ Préparer des notes de répartition de charge pour les applications temps réel liées à la session.
- ✓ Créer des pipelines de release, des smoke tests, des plans de retour arrière et des runbooks.
- ✓ Séparer la configuration d'environnement du code de l'application.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, l'approche suggérée et les livrables attendus.



Lab pratique

Créez un package de déploiement pour EnterpriseOps avec une configuration propre à chaque environnement, HealthCheck.json, une checklist de smoke tests, des notes d'exemple sur les conteneurs, des notes sur le reverse proxy, des recommandations sur les sticky sessions et le répartiteur de charge, et un runbook de retour arrière.

Approche suggérée

- ✓ Lisez le problème en production et identifiez le risque d'architecture avant de toucher à l'UI.
- ✓ Créez ou mettez à jour d'abord l'encapsulation par un service, de commande, de modèle ou de composant concernée.
- ✓ Construisez l'écran ou le composant Wisej.NET en suivant les standards de projet établis.
- ✓ Ajoutez une démonstration du chemin d'échec, pas seulement du chemin nominal.
- ✓ Consignez ce qui a changé dans le journal d'apprentissage et mettez à jour la checklist concernée.

Livrables attendus

- ✓ Diagramme de l'architecture de déploiement
- ✓ Tableau de configuration des environnements
- ✓ HealthCheck.json
- ✓ Runbook de release
- ✓ Checklist de retour arrière et de smoke tests

Questions de révision

- ✓ Qu'est-ce qui nécessite des sticky sessions ?
- ✓ Où les secrets sont-ils stockés ?
- ✓ Comment l'exploitation sait-elle qu'un nœud est défaillant ?



Réflexe de production avancé

Ne vous arrêtez pas quand l'écran paraît correct. Expliquez à qui appartient l'état, les implications de sécurité, les chemins d'échec et le comportement en production avant de considérer le module comme terminé.