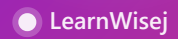


Cloud, container, bilanciamento del carico e ingegneria dei rilasci

Guida della lezione · Guida al lab / esame



Guida della lezione

IIS, Kestrel, reverse proxy, container, sticky session, health check, CI/CD, runbook. Che cosa significa per un team Wisej.NET avanzato — e come affrontare questo modulo.



Focus di produzione

IIS, Kestrel, reverse proxy, container, sticky session, health check, CI/CD, runbook.

1. Il problema in produzione

Le applicazioni Wisej.NET avanzate devono andare oltre le macchine degli sviluppatori, verso ambienti ripetibili con disciplina nella configurazione, health check, bilanciamento del carico, rollback e responsabilità operativa.

2. Concetti chiave

Concetto	Perché è importante
IIS	Il web server di Windows che in molte aziende ospita le applicazioni Wisej.NET, con application pool, regole di riciclo e supporto WebSocket da configurare.
Kestrel	Il web server multiplatforma di ASP.NET Core che esegue direttamente un'applicazione Wisej.NET, di solito dietro un reverse proxy in produzione.
reverse proxy NGINX/Apache	Un server di front-end che termina il TLS, inoltra il traffico WebSocket e instrada le richieste verso una o più istanze dell'applicazione.
Azure/AWS	Piattaforme cloud che ospitano l'applicazione come app service, container o macchine virtuali, ciascuna con il proprio modello di scalabilità e di configurazione.
Docker	Impacchetta l'applicazione e il suo runtime in un'unica immagine, così la stessa build gira in modo identico sulla macchina di uno sviluppatore, su un server di test e nel cloud.
HealthCheck.json	La configurazione dell'health check di Wisej.NET che permette a load balancer e orchestratori di verificare se un'istanza è pronta a servire sessioni.
sticky session	L'affinità del load balancer che instrada ogni richiesta di un utente verso la stessa istanza del server, necessaria perché lo stato di sessione vive nella memoria del server.
CI/CD	Pipeline automatiche di build, test e deployment che trasformano ogni merge in un rilascio ripetibile invece che in una procedura manuale soggetta a errori.

3. Il filo della lezione

Il deployment fa parte dell'architettura

L'hosting cambia il comportamento di identità, WebSocket, logging, configurazione, archiviazione dei file, health check e scalabilità. Gli sviluppatori avanzati progettano per l'ambiente di destinazione abbastanza presto da evitare sorprese.

Scalare tenendo conto della sessione

Le applicazioni Wisej.NET mantengono un ricco stato lato server. Negli ambienti con bilanciamento del carico, l'infrastruttura deve instradare una sessione in modo coerente oppure usare un modello di deployment che preservi il comportamento della sessione. Gli health check aiutano a instradare il traffico lontano dai nodi non in salute.

Scegliere un target di deployment

Un'applicazione Wisej.NET gira ovunque giri ASP.NET Core, il che significa che la scelta dipende dall'organizzazione più che dal framework. IIS su Windows si adatta ai team con operations Windows esistenti, gestione degli application pool e autenticazione Windows. Kestrel dietro un reverse proxy NGINX o Apache si adatta agli host Linux e affida al proxy la terminazione TLS e l'inoltro dei WebSocket. Azure

App Service e gli equivalenti AWS eliminano la gestione dei server ma aggiungono configurazione specifica della piattaforma, regole di scalabilità e integrazione del logging.

I container cambiano l'unità di deployment, non l'host. Un'immagine Docker racchiude l'applicazione e il suo runtime, così lo stesso artefatto gira sulla macchina di uno sviluppatore, su un cluster di test e in produzione. L'immagine deve comunque essere ospitata da qualche parte, e quell'host ha comunque bisogno delle impostazioni per WebSocket, health check e affinità di sessione su cui l'applicazione si basa.

Qualunque sia il target, verifica tre cose prima di impegnarti: i WebSocket attraversano ogni livello tra il browser e l'applicazione, l'endpoint dell'health check è raggiungibile dall'infrastruttura, e l'applicazione vede l'indirizzo e lo schema reali del client attraverso gli header del proxy.

Bilanciare il carico di un'applicazione che dipende dalla sessione

Ogni sessione Wisej.NET vive nella memoria di un solo processo server. Un load balancer che distribuisse le richieste tra i nodi senza affinità invierebbe la seconda richiesta di un utente a un nodo che non ne ha mai sentito parlare, e l'applicazione avvierebbe una nuova sessione. Le sticky session, in cui il bilanciatore instrada tutto il traffico di un client verso lo stesso nodo, sono quindi un requisito e non un'ottimizzazione. Le note sul bilanciamento del carico del lab lo dichiarano esplicitamente per il team operations.

Le funzionalità in tempo reale aggiungono un secondo vincolo. Il server push si basa su un WebSocket che resta aperto tra il browser e il nodo che detiene la sessione, quindi il bilanciatore e qualsiasi proxy devono mantenere attive le connessioni di lunga durata e non devono chiuderle per inattività. Scalare orizzontalmente significa quindi aggiungere nodi che detengono ciascuno una parte delle sessioni, e ridurre significa svuotare un nodo con garbo, così le sue sessioni possono terminare invece di essere interrotte.

Gli health check alimentano il bilanciatore. Nel percorso di errore del lab, il nodo B fallisce il controllo, il bilanciatore instrada le nuove sessioni lontano da esso e gli utenti sul nodo A non se ne accorgono. Documenta che cosa succede alle sessioni che erano già sul nodo B, perché è la prima domanda che faranno le operations.

Pipeline, smoke test, rollback e runbook

Una pipeline di rilascio trasforma ogni merge nella stessa sequenza: build, esecuzione dei test, produzione di un artefatto con versione, deployment in un ambiente di staging, esecuzione degli smoke test e promozione. Il valore sta nella ripetizione. Quando ogni rilascio segue lo stesso percorso, un fallimento è una differenza da indagare e non un mistero. L'artefatto ha una versione, così la dashboard dei rilasci può mostrare che la 2.4.2 è in fase di deployment e che prima girava la 2.4.1.

Gli smoke test sono le poche verifiche che dimostrano che l'applicazione rilasciata è viva: la pagina di login viene visualizzata, una sessione si avvia, l'health check risponde in salute, una query nota restituisce righe e un WebSocket si connette. Durano pochi minuti e vengono eseguiti sull'ambiente reale. Il piano di rollback è ciò che succede quando falliscono. Indica l'artefatto precedente, il comando che lo rilascia di nuovo, la persona che può avviarlo e come confermare che ha funzionato.

Il runbook raccoglie tutto questo nel documento che un tecnico reperibile legge alle due di notte. Il runbook del lab termina con il rollback, perché una procedura di rilascio che non si può annullare non è finita.

La configurazione come contratto

La stessa build deve girare in sviluppo, test e produzione, quindi nulla di specifico di un ambiente può vivere nel codice. Stringhe di connessione, URL dell'identity provider, storage account, feature flag, livelli di log e host consentiti sono configurazione, fornita dall'ambiente e letta all'avvio. La tabella della configurazione degli ambienti del lab elenca ogni impostazione di questo tipo, la sua origine in ciascun ambiente e chi è responsabile di modificarla.

Wisej.NET legge le proprie impostazioni da `Default.json` e il comportamento dell'health check da `HealthCheck.json`. Le impostazioni dell'applicazione seguono il modello di configurazione di ASP.NET Core, così una variabile d'ambiente o un file di segreti montato possono sovrascrivere un valore di un file senza toccare l'immagine. I segreti non compaiono mai nei file di configurazione di cui si fa commit. Vengono iniettati al momento del deployment da un vault o dall'archivio dei segreti della piattaforma.

- Lo stesso artefatto si può rilasciare in ogni ambiente cambiando solo la configurazione?
- Ogni impostazione della tabella ha un responsabile e un'origine per ciascun ambiente?
- Un repository trapiantato esporrebbe qualche segreto?



La regola di produzione avanzata

Non fermarti quando la schermata sembra corretta. Prima di considerare completo il modulo, spiega a chi appartiene lo stato, le implicazioni di sicurezza, i percorsi di errore e il comportamento in produzione.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Scegliere i target di deployment per IIS, Kestrel/reverse proxy, Azure, AWS e container.
- ✓ Preparare note sul bilanciamento del carico per applicazioni in tempo reale che dipendono dalla sessione.
- ✓ Creare pipeline di rilascio, smoke test, piani di rollback e runbook.
- ✓ Separare la configurazione dell'ambiente dal codice dell'applicazione.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.



Lab pratico

Crea un pacchetto di deployment per EnterpriseOps con configurazione specifica per ambiente, HealthCheck.json, checklist di smoke test, note sui container di esempio, note sul reverse proxy, indicazioni su sticky session e load balancer e un runbook di rollback.

Approccio suggerito

- ✓ Leggi il problema in produzione e individua il rischio architetturale prima di toccare la UI.
- ✓ Crea o aggiorna per primo l'incapsulamento in un servizio, comando, modello o componente coinvolto.
- ✓ Costruisci la schermata o il componente Wisej.NET seguendo gli standard di progetto stabiliti.
- ✓ Aggiungi una dimostrazione del percorso di errore, non solo del percorso felice.
- ✓ Annota che cosa è cambiato nel registro di apprendimento e aggiorna la checklist relativa.

Deliverable richiesti

- ✓ Diagramma dell'architettura di deployment
- ✓ Tabella della configurazione degli ambienti
- ✓ HealthCheck.json
- ✓ Runbook di rilascio
- ✓ Checklist di rollback e smoke test

Domande di revisione per lo studente

- ✓ Che cosa richiede le sticky session?
- ✓ Dove vengono memorizzati i segreti?
- ✓ Come fanno le operations a sapere che un nodo non è in salute?



La regola di produzione avanzata

Non fermarti quando la schermata sembra corretta. Prima di considerare completo il modulo, spiega a chi appartiene lo stato, le implicazioni di sicurezza, i percorsi di errore e il comportamento in produzione.