Redis Object Cache instellen voor WordPress

Om de laadtijden van zware, dynamische WordPress-websites (zoals webshops) drastisch te verlagen, bieden wij een veilige en razendsnelle **Redis Object Cache** aan. Omdat veiligheid voorop staat, maken wij geen gebruik van standaard netwerkpoorten, maar krijgt elke website een eigen, afgeschermde UNIX-socket.

In deze handleiding leggen we uit wat Redis doet en hoe je dit in drie simpele stappen configureert voor jouw projecten.

Wat is Redis en waarom heb je het nodig?

Redis is een geavanceerd *in-memory* opslagsysteem. Voor WordPress werkt het als een **Object Cache**. Dit betekent dat het de resultaten van zware database-berekeningen (MySQL) opslaat in het razendsnelle werkgeheugen (RAM) van de server.

- **Waar het voor dient:**
  - **Snelheid:** Complexe pagina's die niet gecachet kunnen worden door een standaard Page Cache (zoals winkelmandjes, afrekenpagina's of ingelogde gebruikers), laden veel sneller.
  - **Database ontlasten:** Omdat WordPress veelvoorkomende vragen direct aan Redis stelt in plaats van aan de MySQL-database, blijft de server stabieler tijdens piekdrukte.
- **Waar het NIET voor dient:**
  - Redis is **geen Page Cache**. Het vervangt plugins zoals WP Rocket of W3 Total Cache niet. Ze werken juist perfect samen: de Page Cache bedient de anonieme gasten, Redis bedient de dynamische acties en de database.
  - Het is geen permanente opslag voor mediabestanden of bestellingen; het is een vluchtig geheugen puur bedoeld voor versnelling.

Stap 1: Jouw unieke Plesk-gebruikersnaam vinden

Omdat jouw Redis-instantie privé is afgeschermd, moet WordPress exact weten via welk pad het met Redis mag communiceren. Dit pad is gebaseerd op de systeemgebruikersnaam van het hostingpakket.

1. Log in op **Plesk**.
2. Ga naar het domein en navigeer naar **Hosting & DNS** > **Hosting**.
3. Kijk onder het kopje **System user's credentials**.
4. Kopieer hier de exacte waarde die achter **Username** staat (bijv. `mijndomein_abc123`).

Stap 2: WordPress configureren (wp-config.php)

Nu we de gebruikersnaam hebben, moeten we WordPress vertellen hoe het verbinding moet maken. Open het `wp-config.php` bestand van de website via bestandsbeheer of SSH en voeg helemaal onderaan de volgende twee regels toe, vlak boven de regel `require_once ABSPATH . 'wp-settings.php';`

```
// Activeer veilige verbinding via een UNIX-socket
define( 'WP_REDIS_SCHEME', 'unix' );

// Pad naar jouw prive-socket (vervang 'plesk_username' twee keer door jouw echte gebruikersnaam!)
define( 'WP_REDIS_PATH', '/run/redis-instances/plesk_username/plesk_username.sock' );
```

> **Belangrijk:** Vergeet niet in de tweede regel `plesk_username` daadwerkelijk te vervangen door de naam die je in stap 1 hebt gekopieerd. Zorg dat het pad klopt, anders kan WordPress de cache niet vinden!

Stap 3: De plugin installeren en activeren

Om WordPress te laten praten met onze Redis-server, raden wij de officiële **Redis Object Cache** plugin aan.

1. Ga in het WordPress-dashboard naar **Plugins > Nieuwe plugin toevoegen**.
2. Zoek naar **Redis Object Cache** (de plugin met het rode kubus-logo, gemaakt door Till Krüss). Bekijk de plugin hier.

3. Installeer en activeer de plugin.
4. Navigeer in het WordPress-dashboard naar **Instellingen > Redis**.
5. Klik op de knop **Enable Object Cache**.

Als alles goed is gegaan, springt de status na enkele seconden op *Connected*. Is de site vernieuwd of wil je testen of nieuwe producten direct zichtbaar zijn? Dan vind je op deze pagina ook altijd de knop **Flush Cache** om het geheugen handmatig te wissen.

Statistieken begrijpen (Metrics)

Op dezelfde pagina (**Instellingen > Redis**) zie je onder het tabblad 'Metrics' handige grafieken over de prestaties van je cache. Hiermee kun je direct zien of Redis efficiënt zijn werk doet:

Metric	Wat betekent het?	Wat is een goede score?
Time	De tijd die het kost om data naar Redis te schrijven of te lezen.	Hoe lager, hoe beter. Idealiter slechts een fractie van een milliseconde.
Bytes	Hoeveel RAM-geheugen de huidige cache in beslag neemt.	Dit schommelt afhankelijk van de grootte van de website. Blijft het heel laag, dan is de cache nog leeg of wordt er te weinig opgeslagen.
Ratio	De *Hit Ratio*. Dit is het percentage van de vragen dat Redis succesvol en direct kon beantwoorden, zonder dat de MySQL-database moest worden aangesproken.	**80% of hoger is uitstekend.** Zakt dit structureel onder de 50%? Dan gooit de cache te snel data weg of bevat de site erg veel niet-cachebare elementen.
Calls	Het totaal aantal aanvragen (commando's) dat WordPress naar Redis heeft gestuurd gedurende de gemeten periode.	Afhankelijk van het verkeer. Een hoog getal in combinatie met een hoge Ratio betekent dat de site enorm wordt versneld!

Het verschil tussen Page Caching en Object Caching (Redis)

Om een website razendsnel te maken, werken we op twee verschillende niveaus. Je kunt het vergelijken met een modern restaurant:

1. Page Caching (De "Kant-en-klare Maaltijd")

Plugins: WP Rocket, W3 Total Cache, LiteSpeed Cache.

Page Caching maakt een statische "foto" (HTML-bestand) van een volledige pagina.

- **Hoe het werkt:** Als de eerste bezoeker komt, moet de server hard werken. De Page Cache slaat het resultaat op. De volgende 1.000 bezoekers krijgen simpelweg dat kant-en-klare bestand voorgeschoteld zonder dat de server hoeft te "koken".
- **De beperking:** Zodra een pagina gepersonaliseerd is (bijv. een unieke winkelwagen, een 'Welkom Jan' tekst, of een dashboard voor ingelogde leden), kan Page Caching niet gebruikt worden. Je kunt immers niet de maaltijd van Jan aan Piet geven.

2. Object Caching met Redis (De "Mise-en-place" van de Chef")

Systeem: Redis.

Redis cachet geen hele pagina's, maar de **onderdelen** (objecten) die nodig zijn om een pagina te bouwen.

- **Hoe het werkt:** In plaats van dat de "chef-kok" (PHP) telkens naar de verre "voorraadkast" (de MySQL-database) moet lopen om elk ingrediënt te zoeken, zet Redis alle veelgebruikte ingrediënten alvast gewassen en gesneden klaar op het aanrecht (in het RAM-geheugen).
- **Waarom dit essentieel is:** Op dynamische momenten waarop Page Caching uitstaat—zoals tijdens het afrekenen, het zoeken in een webshop moet de server de pagina live opbouwen. Redis zorgt ervoor dat dit opbouwproces (het ophalen van prijzen, gebruikersrechten, productopties) in milliseconden gebeurt in plaats van seconden.

Waarom ze perfect samenwerken

Je vervangt de één niet door de ander; ze versterken elkaar:

1. **Page Caching** vangt de bulk van je verkeer op (meestal gasten/bezoekers op de homepage en blogs). Dit bespaart de server enorm veel rekenkracht.
2. **Redis** springt bij op de cruciale momenten waar Page Caching stopt:
   - **Interactie:** Iemand stelt een filter in op een productenpagina.
   - **Conversie:** Iemand voegt een product toe en gaat naar de checkout.

Kortom: Page Caching zorgt voor **schaalbaarheid** (veel bezoekers tegelijk aan kunnen), Redis zorgt voor **snelheid en responsiviteit** op de momenten dat de website voor de bezoeker aan het werk moet.

Kenmerk	Page Cache (WP Rocket / W3TC)	Object Cache (Redis)
Wat wordt bewaard?	Volledige HTML-pagina's.	Resultaten van database-queries en PHP-berekeningen.
Snelheidswinst?	Extreem (serveert statische bestanden).	Zeer hoog (vermijdt trage database-acties).
Ingelogde gebruikers?	Meestal niet (pagina is uniek per user).	**Ja, altijd actief.**
Winkelmandje/Checkout?	Nee (moet live gegenereerd worden).	**Ja, essentieel voor snelheid hier.**

Revisie #1
Gemaakt: 2026-05-04 12:22:26 UTC door Kris Lowet
Bijgewerkt: 2026-05-04 13:24:58 UTC door Kris Lowet