1. Analyse der Verfügbarkeit

1.1. Nutzung verschiedener Zonen und Accounts

Die Lambda-Funktion (`lambda-meatweb-01`) wird von AWS in einer serverlosen Umgebung ausgeführt, die automatisch über mehrere Availability Zones innerhalb einer Region (`us-east-1`) repliziert wird. Dies sorgt für eine hohe Verfügbarkeit und reduziert das Risiko eines Zonenausfalls. Im Gegensatz zur EC2-Implementierung ist keine direkte Zonen- oder Subnetzzuordnung notwendig, da die Lambda-Ausführung von AWS verwaltet wird. Die Funktion nutzt jedoch nur ein AWS-Konto, wodurch keine zusätzliche Isolation oder Redundanz zwischen verschiedenen Konten besteht.

1.2. Skalierbarkeit und Zugriff

Die Lambda-Funktion ist vollständig serverless und skaliert automatisch basierend auf der Anzahl der eingehenden Anfragen. Das bedeutet, dass die Funktion theoretisch unendlich viele parallele Ausführungen unterstützen kann, solange die Limits von AWS nicht überschritten werden.

Der Zugriff auf die Lambda-Funktion erfolgt über:

- Eine Security Group (`SG-Lambda-MeatWeb-01`), die eingehenden und ausgehenden Traffic vollständig erlaubt (`0.0.0.0/0` für alle Ports und Protokolle).
- Eine direkte Bereitstellung über Function URLs oder indirekt über andere Dienste (z. B. API Gateway oder Event-Trigger).

In der aktuellen Konfiguration gibt es keine Einschränkungen im eingehenden Traffic, was ein potenzielles Sicherheitsrisiko darstellt.

2. Analyse der Datensicherheit

2.1. Disaster Recovery

Die Disaster-Recovery-Strategie für die Lambda-Funktion basiert vollständig auf der Nutzung von Infrastructure as Code (IaC). Der bereitgestellte Terraform-Code ermöglicht es, die Lambda-Funktion inklusive ihrer Umgebung (Security Groups, Umgebungsvariablen, IAM-Rollen) im Falle eines Ausfalls schnell und konsistent wiederherzustellen. Der bereitgestellte Quellcode für die Funktion wird in einem Zip-Archiv (`lambda_source_code.zip`) verwaltet, was die Konsistenz bei der Wiederherstellung sicherstellt.

Allerdings wird keine zusätzliche Validierung wie Code-Hashing implementiert, um sicherzustellen, dass der bereitgestellte Code nicht manipuliert wurde.

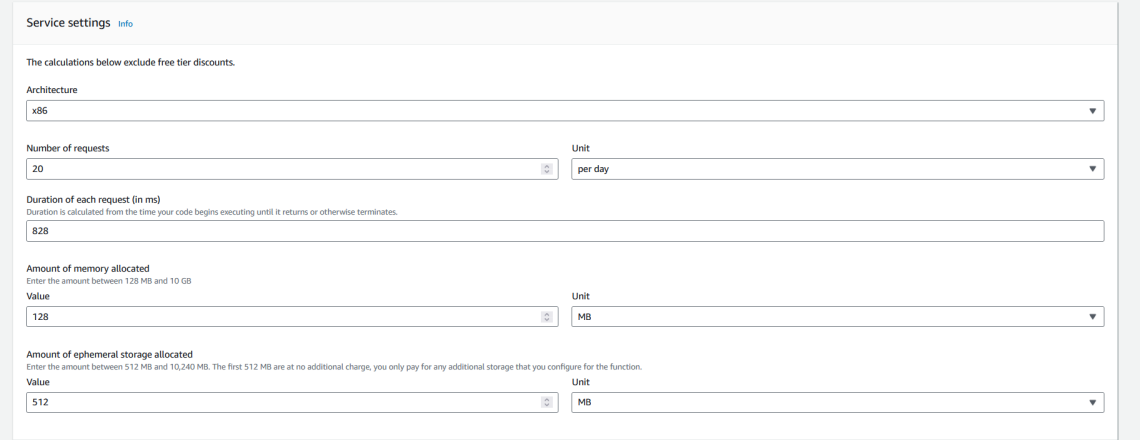
2.2. Backup-Strategien

Für die Lambda-Funktion gibt es keine expliziten Backup-Mechanismen. Da die Funktion jedoch vollständig über den Terraform-Code definiert ist, kann sie problemlos erneut bereitgestellt werden. Die Umgebungsvariablen für die Funktion, die sensible Informationen wie Datenbank-Host, -Port, -Benutzername und -Passwort enthalten, sind im Terraform-Code definiert, wodurch keine separaten Backups dieser Konfiguration erforderlich sind.

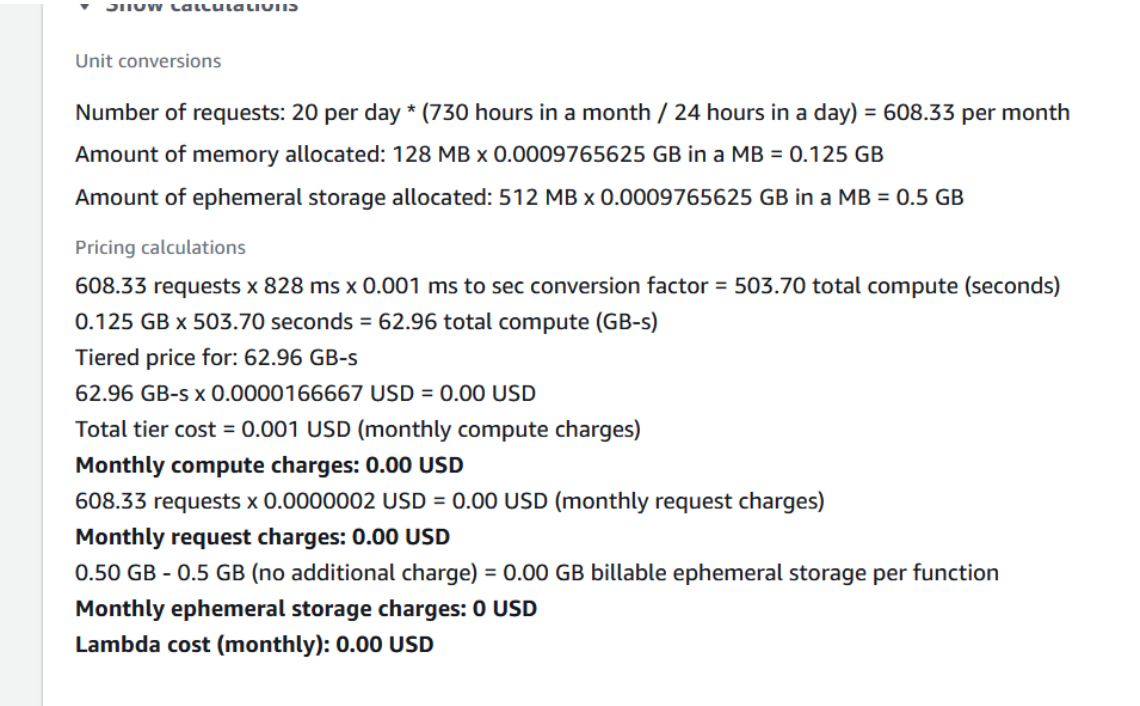
Ein kritisches Risiko ist jedoch, dass sensible Umgebungsvariablen (z. B. Datenbank-Passwort) nicht verschlüsselt verwaltet werden. Es wäre ratsam, AWS Secrets Manager oder Parameter Store zu verwenden, um diese Informationen sicher zu speichern und zu referenzieren.

Kostenanalyse

Ich habe eine Kostenanalyse mit dem AWS Pricing Calculator durchgeführt, um zu ermitteln, wie viel die Lambda-Funktion bei der aktuellen Nutzung kosten würde. Basierend auf dieser Konfiguration, die unserer Implementierung in AWS entspricht:

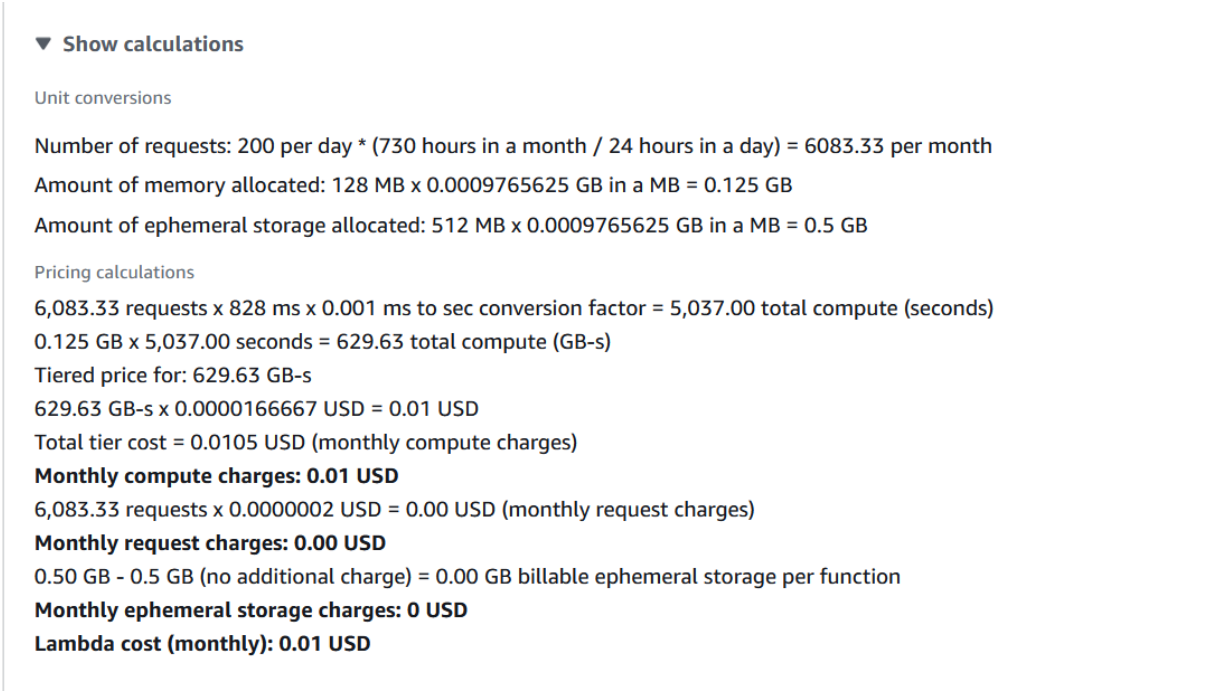


Ergeben sich folgende Ergebnisse:



Bei 20 Requests pro Tag, einer zugewiesenen Speichermenge von 128 MB und einer maximalen Ausführungszeit von 828 ms pro Request entstehen keine Kosten. Die Nutzung bleibt innerhalb des kostenfreien AWS-Kontingents, sodass die Funktion effektiv „kostenlos“ ist.

Wenn die Anzahl der täglichen Requests auf 200 (eine Verzehnfachung) erhöht wird und alle anderen Konfigurationen unverändert bleiben, ergeben sich die folgenden Werte:



▼ **Show calculations**

Unit conversions

Number of requests: 200 per day * (730 hours in a month / 24 hours in a day) = 6083.33 per month
Amount of memory allocated: 128 MB x 0.0009765625 GB in a MB = 0.125 GB
Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB

Pricing calculations

6,083.33 requests x 828 ms x 0.001 ms to sec conversion factor = 5,037.00 total compute (seconds)
0.125 GB x 5,037.00 seconds = 629.63 total compute (GB-s)
Tiered price for: 629.63 GB-s
629.63 GB-s x 0.0000166667 USD = 0.01 USD
Total tier cost = 0.0105 USD (monthly compute charges)
Monthly compute charges: 0.01 USD
6,083.33 requests x 0.0000002 USD = 0.00 USD (monthly request charges)
Monthly request charges: 0.00 USD
0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
Monthly ephemeral storage charges: 0 USD
Lambda cost (monthly): 0.01 USD

Die monatlichen Kosten betragen 1 Cent (0,01 USD).

Dies zeigt, dass das Pay-as-you-go-Modell von AWS perfekt zu unserem Anwendungsfall passt, da unser Service nur selten verwendet wird. Selbst bei einer signifikanten Erhöhung der Anfragen bleiben die Kosten minimal.