Dokumentation: Discord Bot einrichten

# Einführung

In dieser Dokumentation wird der Prozess der Einrichtung eines Ubuntu EC2 t2.micro Servers über Amazon AWS beschrieben. Ziel ist es, einen stabilen und sicheren Server für die Entwicklung und das Hosting eines Python-basierten Discord-Bots bereitzustellen.

# Vorbereitung

## Warum Amazon AWS und Ubuntu?

Die Wahl fiel auf **Amazon AWS** und speziell auf eine **EC2 t2.micro-Instanz**, da AWS mit seiner umfangreichen Infrastruktur und Skalierbarkeit eine der führenden Cloud-Plattformen darstellt. Die Entscheidung für das Free Tier-Angebot von AWS, welches die t2.micro-Instanz für ein Jahr kostenfrei bereitstellt, ermöglicht es, Ressourcen effizient zu nutzen und Kosten zu sparen. Dieser Aspekt ist besonders für Entwicklungs- und Testumgebungen von unschätzbarem Wert.

Die Entscheidung für **Ubuntu** als Betriebssystem wurde aufgrund seiner Stabilität, Sicherheit und Benutzerfreundlichkeit getroffen. Ubuntu ist bekannt für seine umfangreiche Dokumentation und eine aktive Community, was die Problembehandlung und die Suche nach Lösungen erheblich erleichtert. Darüber hinaus bietet Ubuntu regelmäßige Updates und eine breite Unterstützung für diverse Softwarepakete, was es zu einer idealen Wahl für die Servereinrichtung macht.

# Schritt-für-Schritt-Einrichtung

## Erstellung eines Servers bei Amazon AWS

Der erste Schritt bestand darin, ein **Schlüsselpaar** für die SSH-Verbindung zu generieren, welches eine sichere Art der Kommunikation mit dem Server ermöglicht. Vor dem Hintergrund, dass Amazon den Inhalt der Instanz standardmäßig bei deren Beendigung automatisch entfernt, war die Aktivierung des **Beendigungsschutzes** ein wesentlicher Schritt, um den Server vor versehentlicher Löschung zu schützen. Zudem wurde eine **Firewall** konfiguriert, um den eingehenden Datenverkehr auf Port 22 (SSH) zu beschränken und den ausgehenden Datenverkehr zu erlauben. Dies stellt sicher, dass der Server vor unautorisiertem Zugriff geschützt ist, während eine reibungslose Kommunikation für Entwicklungszwecke ermöglicht wird.

## Installation und Konfiguration der Software

Nach der Grundkonfiguration des Servers ist es entscheidend, das System und den **Kernel** zu aktualisieren. Dies ist ein wichtiger Schritt, um sicherzustellen, dass alle Softwarepakete auf dem neusten Stand sind und bekannte Sicherheitslücken geschlossen werden. Durch das Aktualisieren des Kernels stellen wir sicher, dass der Server die neustens Sicherheits- und Leistungsoptimierungen erhält, die für einen reibungslosen Betrieb und Schutz vor bekannten Angriffsvektoren unerlässlich sind. Nach der Sicherstellung der Server-Grundkonfiguration und der Aktualisierung des Kernels erfolgte die Installation der notwendigen Software. Die Entscheidung, **Node.js** und **NPM** zu installieren, beruhte auf der Notwendigkeit, JavaScript-Code serverseitig auszuführen. **pm2**, ein Prozessmanager für Node.js-Anwendungen, wurde aufgrund seiner Fähigkeit, Anwendungen im Hintergrund zu betreiben und bei Bedarf automatisch neu zu starten, installiert. Dies ist besonders nützlich, um die Verfügbarkeit des Discord-Bots zu gewährleisten.

Für die Entwicklung und den Betrieb des Discord-Bots wurde eine **Python-Entwicklungsumgebung** eingerichtet. Dies umfasste die Installation von python3-pip und python3-venv, um eine isolierte Umgebung für den Bot zu schaffen. Die Isolierung verhindert Konflikte zwischen verschiedenen Projekten oder Abhängigkeiten und erleichtert die Verwaltung von Paketversionen.

## Einrichtung des Bots und Webhooks

Die Einrichtung des Discord-Bots und des Webhooks erforderte besondere Aufmerksamkeit, vor allem da der Bot-Code in einem **privaten GitHub-Repository** liegt. Um den Server mit diesem privaten Repository zu verbinden, muss ein **SSH-Schlüssel** speziell für GitHub erstellt werden. Dieser Schritt ist von entscheidender Bedeutung, da er eine sichere Methode bietet, den Server ohne die Notwendigkeit eines Passworts mit GitHub zu verknüpfen. Nachdem der öffentliche Schlüssel auf GitHub hinterlegt wurde, kann der Server sicher auf das Repository zugreifen, was für die Einrichtung und Aktualisierung des Bots notwendig ist.

Nach der erfolgreichen Verbindung mit dem privaten GitHub-Repository, stand als nächster wichtiger Schritt das sichere Hochladen der **.env Datei** auf dem Server an. Diese Datei enthält essenzielle Konfigurationsvariablen und Geheimnisse, wie z.B. Bot-Token, die für den Betrieb des Discord-Bots unerlässlich sind. Die Übertragung dieser Datei erfolgte sicher entweder direkt auf dem Server mittels eines Texteditors nach der Herstellung einer **SSH-Verbindung** oder durch Hochladen mittels **SFTP**, um die sensiblen Informationen vor unberechtigten Zugriff zu schützen.

Die Verwendung von **Smee** als Webhook-Transportmechanismus ist eine strategische Entscheidung, um den Autoupdater des Bots zu betreiben, ohne den Server direkt dem Internet aussetzen zu müssen. Smee leitet die von GitHub kommenden Webhook-Events sicher an den lokalen Server weiter, wo ein **Flask-Webserver** läuft, der speziell dafür konfiguriert ist, diese Events abzufangen. Dieser Ansatz minimiert das Risiko von Sicherheitslücken, da keine offenen Ports auf dem Server für externe Anfragen benötigt werden. Der Flask Server verarbeitet die eingehenden Anfragen von Smee und initiiert bei Bedarf ein Skript, um den Bot automatisch zu aktualisieren, wenn neue **Pull Requests** auf GitHub gemerged werden. Diese Methode stellt sicher, dass der Bot immer auf dem neusten Stand ist, ohne die Sicherheit des Servers zu kompromittieren.

## Abschluss und Betrieb

Nach Abschluss aller Einrichtungsschritte wurde der Discord-Bot gestartet und durch pm2 überwacht. Diese Konfiguration gewährleistet, dass der Bot stets verfügbar ist und bei Bedarf automatisch neu gestartet wird. Die erfolgreiche Einrichtung und Konfiguration des Servers markiert den Beginn einer effizienten Entwicklungs- und Betriebsphase des Discord-Bots.

# Fazit

Die detaillierte Betrachtung der Einrichtungsprozesse zeigt die Komplexität und die notwendigen Überlegungen auf, die in die Entwicklung und den Betrieb eines modernen, interaktiven Discord-Bots fließen. Von der Wahl der Plattform und des Betriebssystems über die Sicherheitsaspekte bis hin zur Automatisierung der Softwareaktualisierungen – jede Entscheidung ist von dem Bestreben geleitet, Effizienz, Sicherheit und Benutzerfreundlichkeit zu maximieren. Die Implementierung eines Flask-Servers zur Handbung von GitHub-Webhooks mittels Smee stellt eine innovative Lösung dar, die die Aktualisierung des Bots vereinfacht und gleichzeitig die Sicherheit des Servers gewährleistet.

# Glossar

* **Amazon AWS:** Eine umfangreiche Cloud-Plattform von Amazon, die Rechenleistung, Datenbank-Speicherung und weitere Funktionalitäten bietet, um Unternehmen beim Wachstum zu unterstützen.
* **EC2 (Elsatic Compute Cloud):** Ein Teil von Amazon Web Services, der skalierbare Rechenkapazität in der Cloud anbietet.
* **t2.micro-Instanz:** Eine kostengünstige Instanz-Konfiguration innerhalb des Amazon EC2-Dienstes, die unter das AWS Free Tier fällt und für Testzwecke und kleine Anwendungen geeignet ist.
* **Ubuntu:** Eine kostenlose und open-source Linux-Distribution, basierend auf Debian. Bekannt für seine Benutzerfreundlichkeit und umfangreiche Unterstützung durch Community und professionelle Dienstleistungen.
* **Schlüsselpaar:** Ein Setz auf privatem und öffentlichem Schlüssel, das in der digitalen Kommunikation verwendet wird, um die sichere Authentifizierung und Verschlüsselung zu gewährleisten.
* **Beendigungsschutz:** Eine Funktion in AWS, die verhindert, dass eine EC2-Instanz versehentlich gelöscht oder beendet wird.
* **Firewall:** Ein Netzwerksicherheitssystem, das den ein- und ausgehenden Netzwerkverkehr basierend auf vordefinierten Sicherheitsregeln überwacht und steuert.
* **Kernel:** Der zentrale Bestandteil eines Betriebssystems, der die Kommunikation zwischen Software und Hardware steuert.
* **Node.js:** Eine plattformübergreifende, Open-Source JavaScript-Laufzeitumgebung, die es ermöglicht, JavaScript-Code serverseitig aufzuführen.
* **NPM (Node Package Manager):** NPM ist der Standardpaketmanager für die Node.js JavaScript-Laufzeitumgebung.
* **Python-Entwicklungsumgebung:** Eine Umgebung, die alle notwendigen Werkzeuge für die Python-Entwicklung enthält, einschließlich Bibliotheken und Interpreter.
* **GitHub-Repository:** Ein Speicherort auf der Plattform GitHub, der zu Versionierung und Speicherung von Code-Projekten verwendet wird.
* **SSH-Schlüssel:** Ein Zugangsschlüssel, der für die sichere Verbindung und Authentifizierung über das SSH-Protokoll verwendet wird.
* **.env Datei:** Eine Datei zur Speicherung von Umgebungsvariablen, die sensible Informationen wie API-Schlüssel und Passwörter enthält.
* **SSH (Secure Shell):** Ein Netzwerkprotokoll, das für eine sichere Kommunikation über ungesicherte Netzwerke verwendet wird.
* **SFTP (Secure File Transfer Protocol):** Ein Netzwerkprotokoll, das für den sicheren Transfer von Dateien über ungesicherte Netzwerke verwendet wird.
* **Smee:** Ein Tool, das Webhook-Events an einen lokalen Entwicklungs-Server weiterleitet, um die Entwicklung von Anwendungen, die auf Webhooks reagieren, zu vereinfachen.
* **Flask-Webserver:** Ein mikro Web-Framework für Python, das einfach zu verwenden ist und sich für kleine bis mittelgroße Webanwendungen eignet.
* **Pull Request:** Ein Vorgang auf GitHub, bei dem Entwickler Änderungen an einem Code vorschlagen und zur Überprüfung einreichen, bevor diese in die Hauptbranch des Projekts integriert werden.