Merkblatt – Nutzung von GitHub Copilot Business/Enterprise

(IDE-Plugin in VS Code/JetBrains IDEs & GitHub Copilot CLI)

Allgemeine Grundsätze

- Der **KI-Leitfaden** [1] der Fachstelle Informationssicherheit ist zwingend **einzuhalten**.
- GitHub Copilot ist für Daten der Klassifizierungsstufen [2] **öffentlich** und **intern** freigegeben. Für Daten der Stufen vertraulich und geheim darf er nicht genutzt werden. Das Freigeben des Zugriffs für GitHub Copilot auf Workspaces mit vertraulichen oder geheimen Daten ist nicht erlaubt. Prompts dürfen keine vertraulichen oder geheimen Daten enthalten.
- GitHub Copilot ist ein KI-Pair-Programmierer, der **Codevorschläge** generiert, aber keine Garantie auf Korrektheit oder Sicherheit gibt. Die Vorschläge müssen **wie fremder Code behandelt** werden: Reviewen Sie, testen Sie, führen Sie Security-Checks und Code-Style-Prüfungen durch.
- **Sie bleiben** fachlich und rechtlich **immer verantwortlich** für den übernommenen Code (z. B. Security, Lizenzen, Datenschutz) und die Einhaltung von Coding Best Practices im Security Bereich, wie z.B. OWASP Top 10
- **Konfigurationsdateien** (z. B. «`.env`»- oder «`application.properties`»-Dateien), die vertrauliche oder geheime Daten wie z. B. Passwörter, Tokens oder andere Secrets enthalten, dürfen **nicht in Workspaces** abgelegt werden, die für den GitHub Copilot freigegeben sind.
- Beachten Sie, dass **Entwicklungsaufgaben** stets im Kontext und mit den Berechtigungen des aufrufenden Benutzers laufen, also **mit Ihren Berechtigungen.**

Dos und Don’ts

Bereich	Dos	Don’ts
Allgemeine Nutzung	Verwenden Sie Coding-Agents als unterstützendes Werkzeug, um Entwicklungsprozesse effizienter zu gestalten. Die technische und inhaltliche Verantwortung für den Code verbleibt stets bei der Entwicklerin bzw. dem Entwickler.	Setzen Sie automatisch generierten Code nicht ohne eigene Bewertung oder Qualitätssicherung in produktiven Umgebungen ein.
Codequalität	Nutzen Sie den Agenten zur Generierung von Codevorschlägen, Tests, Dokumentation oder Refactorings und übernehmen Sie nur nachvollziehbare und fachlich geeignete Ergebnisse. Verifizieren Sie generierte Ergebnisse durch Tests und Reviews und stellen Sie sicher, dass bewährte Sicherheitsprinzipien (z. B. Input-Validierung, Authentifizierung, Secrets) eingehalten werden.	Integrieren Sie keine umfangreichen oder sicherheitsrelevanten Codeabschnitte, deren Funktion oder Wirkung nicht eindeutig verstanden wurde. Gehen Sie nicht davon aus, dass generierter Code automatisch korrekt oder sicher ist, und hinterlegen Sie keine Passwörter, API-Schlüssel oder ähnliche sensiblen Daten im Prompt oder Code.
CLI & Automatisierung	Überprüfen Sie generierte Befehle sorgfältig, kontrollieren Sie Änderungen über «`git diff`» und führen Sie automatisierte Aktionen ausschliesslich in Entwicklungs- oder Testumgebungen aus.	Führen Sie keine generierten Befehle mit Schreib-, Lösch- oder Deployment-Auswirkungen aus, ohne ihre Konsequenzen zu verifizieren.
Datenschutz & Informationssicherheit	Stellen Sie sicher, dass keine vertraulichen oder geheimen Daten im Prompt verwendet werden. Beachten Sie dabei stets die internen Datenschutz- und Sicherheitsrichtlinien.	Geben Sie keine personenbezogenen Daten, Kundendaten oder vertrauliche Informationen in Prompts ein und speichern Sie keine Secrets im Sourcecode. Verwenden Sie Copilot nicht bei Sourcecode mit vertraulichen oder geheimen Daten; insbesondere mit Gesundheitsdaten oder Daten mit potenziell militärischer Bedeutung.
Kontext & Prompting	Formulieren Sie Prompts präzise, benennen Sie technische Rahmenbedingungen, Zielsetzungen und relevante Beispiele, um qualitativ hochwertige Ergebnisse zu erhalten.	Verfassen Sie keine unklaren oder kontextlosen Prompts wie «Schreibe den Code dafür».
Tests	Fordern Sie den Agenten gezielt zur Erstellung von Unit- und Integrationstests auf und führen Sie diese konsequent aus, um die Funktionalität zu bestätigen.	Verzichten Sie nicht auf Tests oder verlassen Sie sich ausschliesslich auf oberflächliche «Happy-Path»-Prüfungen.
Architektur & Design	Nutzen Sie den Agenten als Diskussions- und Inspirationsquelle für Architektur- und Designentscheidungen, treffen Sie finale Entscheidungen jedoch auf Basis fachlicher Beurteilung.	Übernehmen Sie keine komplexen Architekturentscheidungen oder Strukturen ohne eigene technische Validierung.

Dokumentation & Wartbarkeit	Verwenden Sie den Agenten zur Unterstützung bei der Erstellung von Code-Kommentaren, Dokumentationen und Migrationshinweisen und achten Sie auf Nachvollziehbarkeit, Lesbarkeit und Einhaltung interner Konventionen.	Führen Sie keine Änderungen ohne angemessene Dokumentation oder ohne Berücksichtigung bestehender Standards durch.
Iteratives Arbeiten	Gehen Sie schrittweise vor, überprüfen Sie Zwischenergebnisse regelmässig und verfeinern Sie Ihre Prompts, um die Qualität der Ergebnisse kontinuierlich zu verbessern.	Versuchen Sie nicht, ein komplexes oder unpräzise definiertes Problem in einem einzigen Prompt vollständig lösen zu lassen.
Teamrichtlinien	Etablieren Sie gemeinsame Standards und Prozesse für den Einsatz von Coding-Agents und fördern Sie den regelmässigen Austausch von Erfahrungen und Best Practices.	Ermöglichen Sie keinen individuellen, unkoordinierten Einsatz von KI-Tools ohne definierte Leitlinien oder Abstimmung im Team.
Fehleranalyse & Debugging	Nutzen Sie den Agenten gezielt zur Interpretation von Logs, Stacktraces oder Fehlermeldungen und bewerten Sie vorgeschlagene Lösungsansätze systematisch.	Akzeptieren Sie nicht unkritisch den ersten vorgeschlagenen Korrekturvorschlag, ohne alternative Ursachen zu prüfen.
Lernfaktor	Verwenden Sie den Agenten als Lernunterstützung, um Konzepte besser zu verstehen und fachliche Zusammenhänge zu vertiefen.	Reduzieren Sie die Nutzung des Agenten nicht auf das blosse Kopieren und Einfügen von Code ohne inhaltliche Auseinandersetzung.
Projektstruktur & Versionskontrolle	Strukturieren Sie Aufgaben in überschaubare Einheiten, verwenden Sie nachvollziehbare Commits und stellen Sie sicher, dass alle Änderungen einem Review-Prozess unterliegen.	Erstellen Sie keine umfangreichen, ungetrennten Commits mit KI-generiertem Code und führen Sie diese nicht ohne Prüfung in den Hauptzweig zusammen.
Sicherheitsmassnahmen	Führen Sie sicherheitsrelevante Änderungen in isolierten Branches oder Testumgebungen durch und nutzen Sie Code-Scanning- und Security-Review-Werkzeuge.	Gewähren Sie Coding-Agents keine Schreib- oder Zugriffsbefugnisse auf produktive Systeme oder Datenbanken.

Nutzung als IDE-Plugin (VS Code / JetBrains IDEs)

Installation und Anmeldung

- Die Zentrale Informatik empfiehlt die Verwendung folgender IDE-Versionen: **VS Code** sowie die IDEs von JetBrains (wie **IntelliJ** oder **PyCharm**). Diese IDEs stehen im Unternehmensportal der verwalteten Geräte zur Verfügung. Installieren Sie für diese IDEs das offizielle GitHub-Copilot-Plugin aus dem jeweiligen Marketplace.
- Melden Sie sich ausschliesslich mit Ihrem UZH-GitHub-Account an, der dem Copilot-Business-Plan zugeordnet ist.
- Als Authentication Provider muss in JetBrains IDEs «**uzh-zi.ghe.com**» eingegeben werden. In VS Code ist der Authentication Provider «github-enterprise» und als GitHub-Enterprise Uri muss «**https://uzh-zi.ghe.com**» eingegeben werden.
- Aktivieren Sie Copilot nur in Projekten/Workspaces, die Daten der Stufe öffentlich oder intern enthalten. Der Copilot hat Zugriff auf dieses Verzeichnis und alle darunterliegenden Verzeichnisse.
- Man kann in beiden IDEs zwischen *Ask* und *Agent* Mode wählen: Im *Agent-Modus* geben Sie der KI klare Anweisungen oder eine strukturierte Aufgabe vor (z. B. «Schreiben Sie eine Funktion für X mit Y»)

Im *Ask-Modus* stellen Sie offene Fragen (z. B. «Wie löse ich Problem Z am besten?») – die KI schlägt Lösungen vor oder diskutiert Optionen. Im Gegensatz zum Agent Mode werden keine direkten Änderungen am Code vorgenommen. **Ask = Beratung, Agent = Umsetzung.** Wir empfehlen, zu Beginn und bei wenig Erfahrung mit solchen Tools zuerst den Ask Modus zu verwenden.

Spezifische Punkte VS Code vs. IntelliJ

- VS Code:
  - Achten Sie auf projektbezogene Settings (z. B. aktiv in Workspace A, deaktiviert in Workspace B).
  - Nutzen Sie die Konfiguration, um Copilot nur für bestimmte Sprachen oder Ordner zu aktivieren.
  - In VS Code können Sie eine neue CLI-Session öffnen. Dadurch lässt sich die GitHub Copilot CLI direkt in VS Code nutzen – dabei gelten dieselben Regeln wie im nächsten Abschnitt («Nutzung der GitHub Copilot CLI») beschrieben.
- IntelliJ (und andere JetBrains-IDEs):
  - Plugin nur aus offizieller Quelle, Einstellungen über «Tools → GitHub Copilot»/ähnlich vornehmen.
  - Prüfen Sie IDE-weite vs. projektbezogene Einstellungen, um ungewollte Nutzung mit vertraulichen/geheimen Daten zu vermeiden.

Nutzung der GitHub Copilot CLI

Funktionsumfang und Risiken

Die Copilot CLI bietet eine Chat-ähnliche Schnittstelle im Terminal. Die Copilot CLI kann Dateien erstellen/ändern/löschen und Befehle auf Ihrem System ausführen. Sie kann u. a. Code generieren, Bugs beheben, Tests schreiben, Dokumentation anpassen und mit GitHub-Repos, Issues und Pull Requests interagieren. Dadurch besteht ein erhöhtes Risiko, unbeabsichtigt schädliche oder unerwünschte Befehle auszuführen (z. B. Daten löschen, produktive Dienste beeinflussen).

Sicherer Rahmen für CLI-Nutzung

- Nutzen Sie die Copilot CLI nur in:
  - Lokalen Entwicklungs- oder dedizierten Testumgebungen, nicht direkt auf Produktionssystemen.
  - Workspaces, die für KI-Unterstützung freigegeben sind.
- Starten Sie die CLI immer im richtigen Verzeichnis (Workspace-Root) und prüfen Sie, welche Dateien betroffen sein können.
- Pfadberechtigungen:
  - Standardmässig kann Copilot CLI auf das aktuelle Arbeitsverzeichnis, seine Unterverzeichnisse und das temporäre Systemverzeichnis zugreifen.
  - Die Eingrenzung von Berechtigungen erfolgt heuristisch, und GitHub garantiert nicht, dass alle Dateien ausserhalb vertrauenswürdiger Verzeichnisse geschützt werden.
- Da keine strikte «Allowlist»-Einstellung existiert, sollten Sie darauf achten, die Copilot CLI niemals mit dem Flag `--allow-all-paths` oder `--yolo` zu starten, da dies alle Pfadprüfungen deaktiviert.
- Lassen Sie keine automatischen Befehlsausführungen ohne Sichtprüfung zu:
  - Prüfen Sie vorgeschlagene Shell-Kommandos vor dem Ausführen.
  - Lehnen Sie Befehle ab, die Daten löschen, Systemkonfigurationen ändern oder in Produktionsumgebungen eingreifen könnten. Verwende Flags wie `--deny-tool`, um zu verhindern, dass beispielsweise Daten gelöscht oder Änderungen automatisch «gepushed» werden:

```
- copilot --deny-tool='shell(rm)'
- copilot --deny-tool='shell(git push)'
```

Konkrete Safe-Usage-Hinweise

- Verwenden Sie die CLI vor allem für:
  - Erklärungen zu Code und Befehlen
  - Vorschläge für Terminal-Kommandos
  - einfache Refactorings
  - Testgenerierung
  - Dokumentation.
- Bei Datenänderungen durch die CLI:
  - Nutzen Sie «git diff» zur Prüfung von Änderungen vor einem Commit.
  - Stellen Sie sicher, dass alle Änderungen Code-Reviews und CI/CD-Pipelines durchlaufen.
- Nutzen Sie den interaktiven Modus bewusst:
  - Reagieren Sie nur auf Vorschläge, die Sie verstanden haben.
  - Brechen Sie Sessions ab, wenn unklare oder riskante Aktionen vorgeschlagen werden.

Im Dokument erwähnte Verweise / Links

[1] Leitfaden zur Nutzung von KI-Systemen (PDF)
[2] Weisung zur Klassifizierung von Informationen (PDF)

Weiterführende Links

- Visual Studio Code - Open Source AI Code Editor
- Visual Studio Code - Sicherheit
- Nur falls andere IDEs erlaubt sind: Verwendung von GitHub Copilot mit einem Konto bei GHE-com
- GitHub Copilot CLI: Festlegen der zugelassenen bzw. abgelehnten Tools