1. Überprüfen der Änderungen

Bevor Sie Änderungen committen, können Sie sich einen Überblick verschaffen, was geändert wurde:

bash

Copy code
```
git status
```
Dieser Befehl zeigt Ihnen die Dateien, die geändert wurden oder neu sind und noch nicht zum Tracking hinzugefügt wurden.

2. Dateien zum Staging-Bereich hinzufügen

Um Dateien zum Commit vorzubereiten, fügen Sie sie zum Staging-Bereich hinzu:

bash

Copy code
```
git add .
```
oder

bash

Copy code
```
git add <Dateipfad>
```
Verwenden Sie `git add .` um alle geänderten Dateien hinzuzufügen, oder ersetzen Sie `.` durch spezifische Dateipfade, wenn Sie nur bestimmte Änderungen committen möchten.

3. Änderungen committen

Nachdem Sie die Dateien hinzugefügt haben, committen Sie die Änderungen mit einer aussagekräftigen Nachricht:

bash

Copy code
```
git commit -m "Eine klare Beschreibung der Änderungen"
```
Die Commit-Nachricht sollte kurz und präzise sein und erklären, was in diesem Commit geändert wurde.

4. Änderungen zum Remote-Repository pushen

Nachdem Sie Ihre Änderungen committet haben, pushen Sie sie zum Remote-Repository, um sie sicher zu speichern und für andere verfügbar zu machen:

bash

Copy code
```
git push origin main
```
Dieser Befehl sendet Ihre Commits zum main Branch Ihres Remote-Repositories auf GitHub.

Zusätzliche Tipps:

- Regelmäßiges Pullen: Um Konflikte zu vermeiden, ist es eine gute Praxis, regelmäßig den aktuellen Stand des Remote-Repositories zu pullen, besonders bevor Sie mit der Arbeit beginnen oder wenn Sie in einem Team arbeiten:bash

  Copy code
  ```
  git pull origin main
  ```
-

- Branches nutzen: Für größere Features oder Änderungen ist es ratsam, separate Branches zu verwenden, statt direkt im main Branch zu arbeiten. Dies hält Ihren main Branch stabil und die Arbeit organisiert:bash

  Copy code
  ```
  git checkout -b feature-branch-name
  ```
-

  Nachdem die Arbeit im Feature-Branch abgeschlossen ist, können Sie ihn zurück in den main Branch mergen.

Diese Befehle stellen die Grundlagen dar, um Ihre Arbeit mit Git effektiv zu verwalten und Ihr Repository auf dem neuesten Stand zu halten.