

corewire

Helm

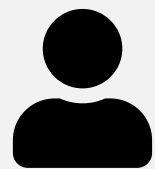
CI/CD und GitOps

# Folien-Hinweis

- Space, Page down: Nächste Folie
- Page up: Vorherige Folie
- ESC, o: Übersicht

[Zur Kapitelübersicht](#)

# CI/CD

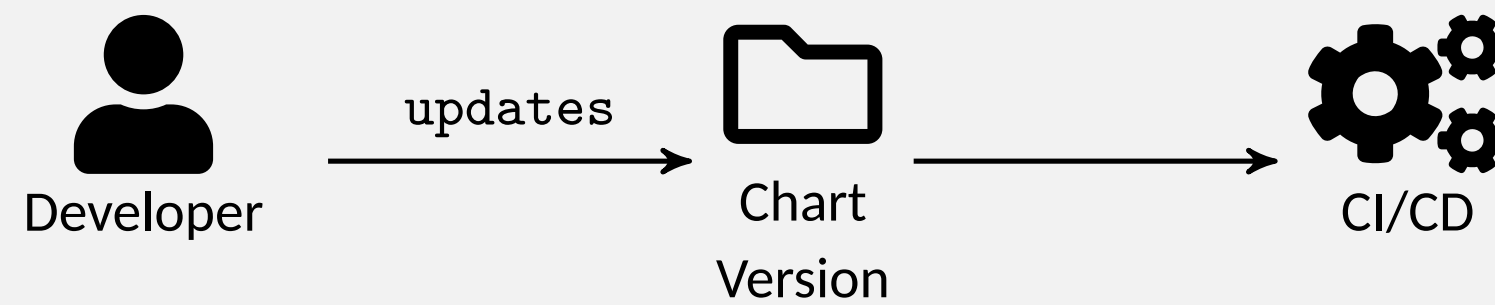


Developer

# CI/CD



# CI/CD



# CI/CD



# CI/CD Commands für Helm

```
helm dependency build ./my-chart
```

Installiert die in `Chart.lock` definierten Abhängigkeiten  
(Subcharts)

# CI/CD Commands für Helm

```
helm template ./my-chart | kubectl apply --dry-run=server -f -
```

Rendert die Kubernetes-Manifeste lokal und validiert sie gegen das Cluster-Schema



# CI/CD Commands für Helm

```
helm upgrade --install my-release ./my-chart --version 3.1.0 -f values.yaml
```

# CI/CD



# CI/CD



Code  
Change

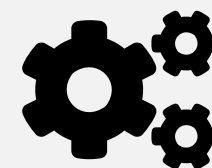


Developer

updates



Chart  
Version



CI/CD

installs

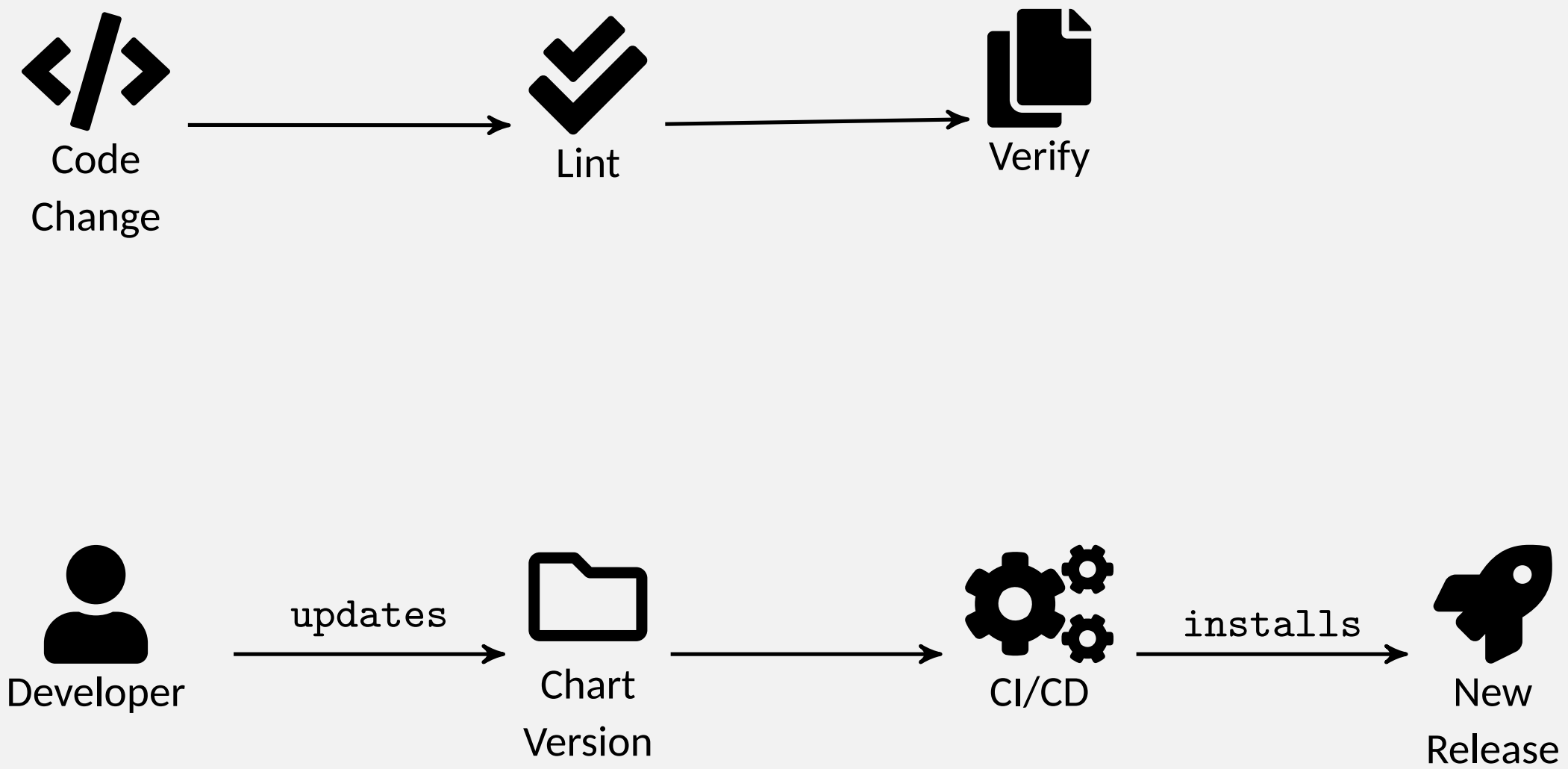


New  
Release

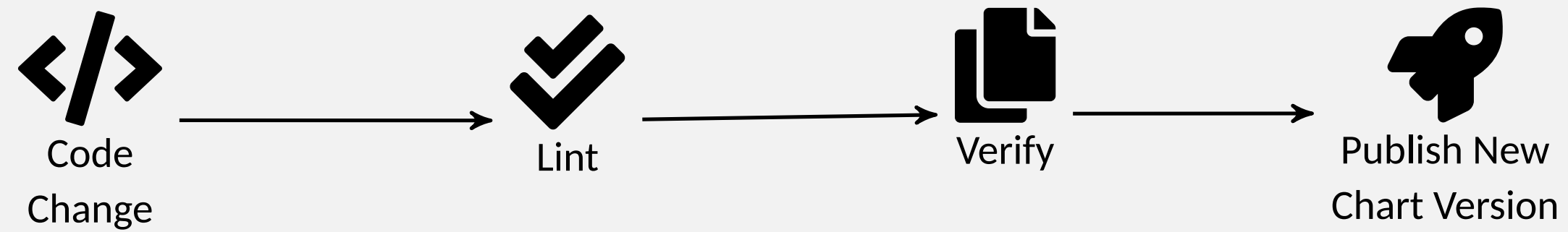
# CI/CD



# CI/CD



# CI/CD



# Standard-Flow für Helm Chart CI

1. Chart linten
2. Templates rendern
3. Chart paketieren
4. Chart in Registry oder Repo veröffentlichen

# CI/CD Commands für Helm

```
helm lint ./my-chart
```

Überprüft die Syntax und Struktur eines Charts



# CI/CD Commands für Helm

```
helm template ./my-chart
```

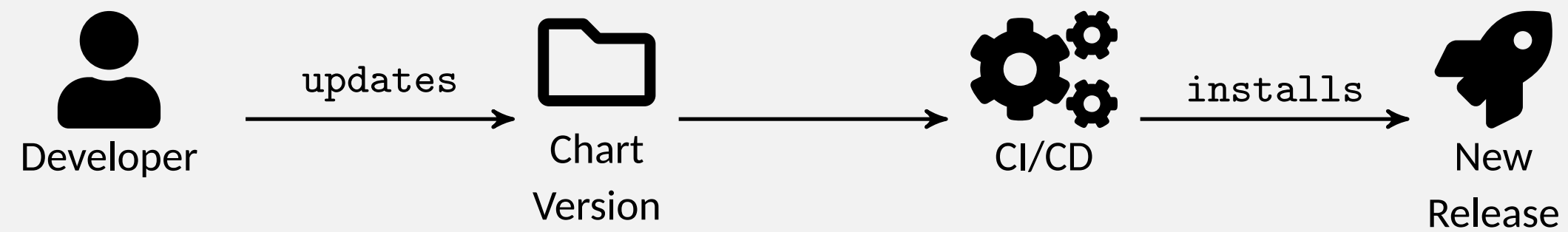
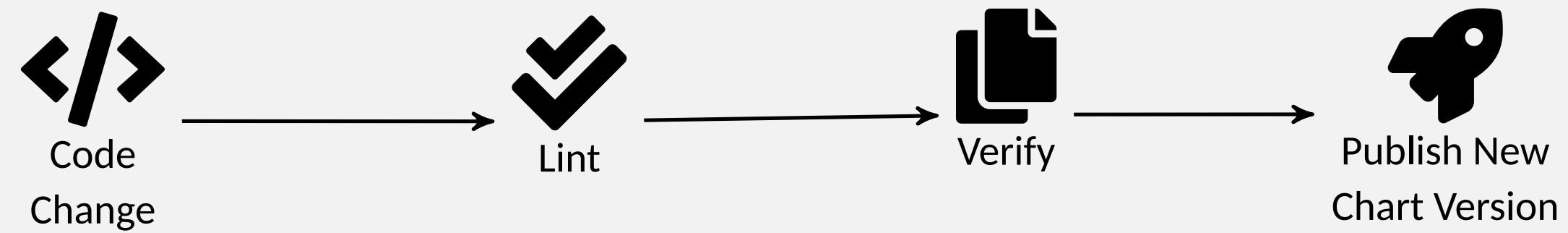
Rendert die Kubernetes-Manifeste lokal, ohne zu installieren

# CI/CD Commands für Helm

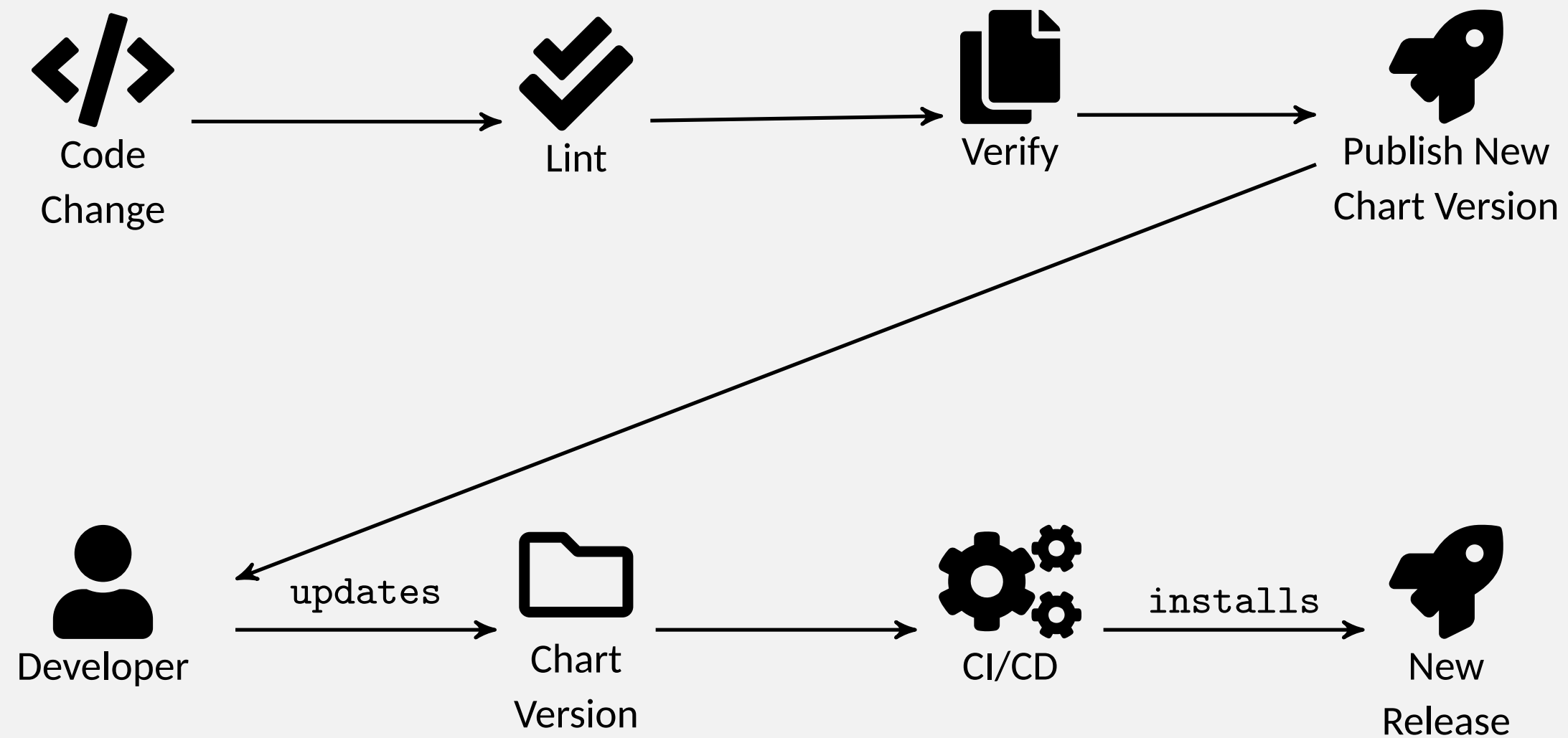
```
helm package ./my-chart
```

Packt ein Chart in eine `.tgz`-Datei. Diese kann dann in einer Registry oder einem Chart-Repo veröffentlicht werden.

# CI/CD



# CI/CD



# Nutzung in anderem Projekt: Dependency

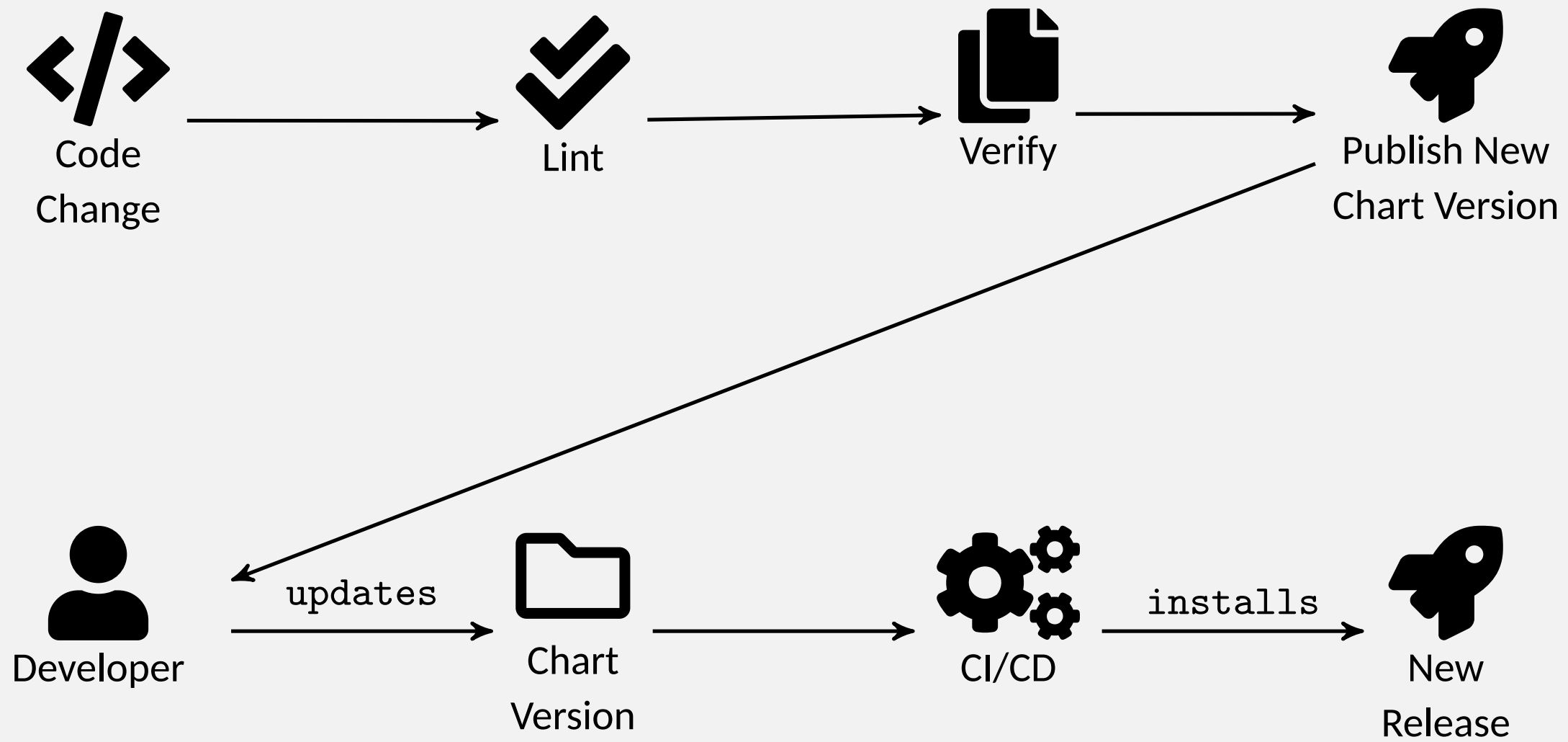
```
dependencies:  
  - name: webshop  
    version: 1.4.2  
    repository: oci://registry.example.com/helm
```

# Nutzung in anderem Projekt: Deployment

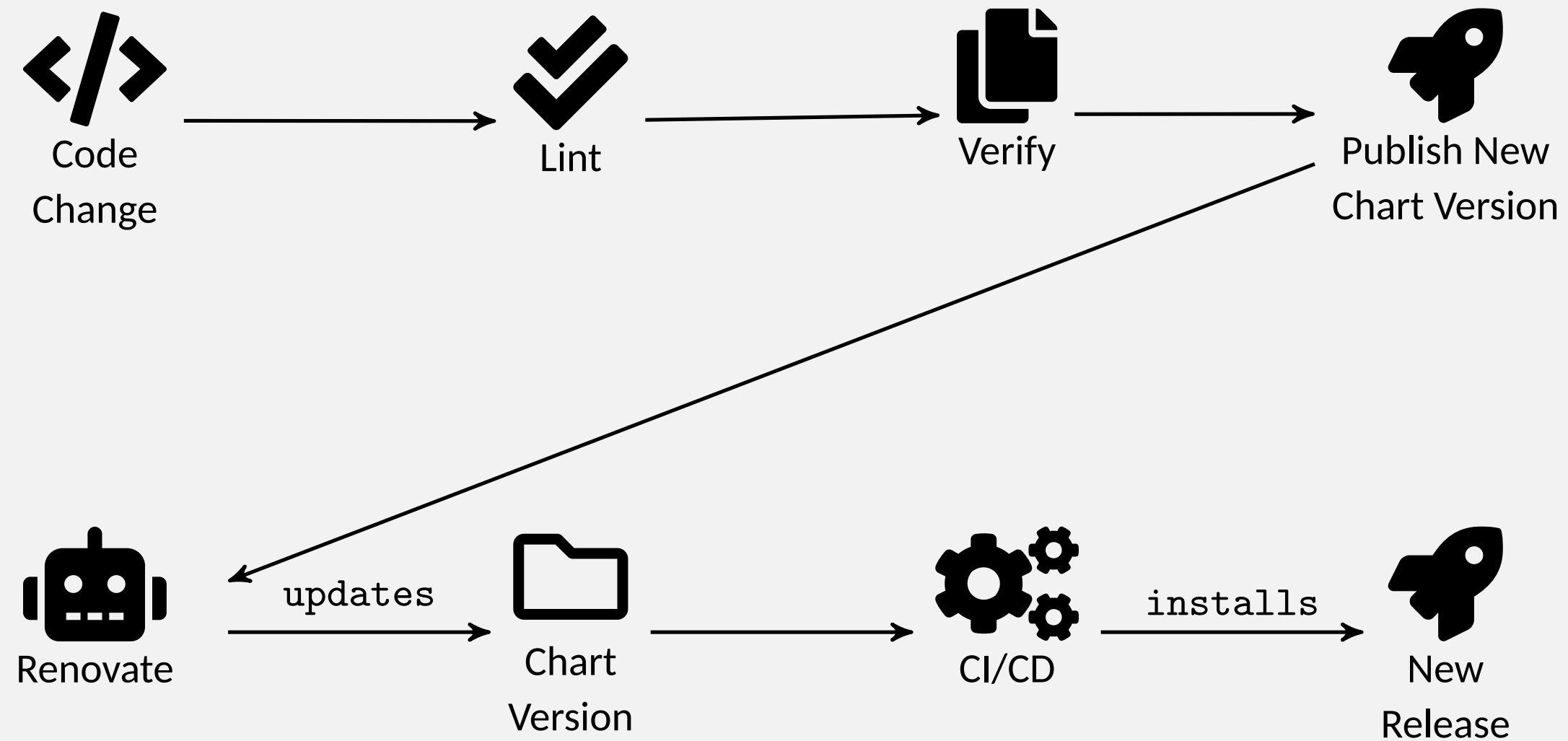
```
helm upgrade --install webshop-platform ./platform-chart \
-f values/prod.yaml
```

- Die Pipeline des Consumer-Projekts nutzt nur noch Version + Values
- Chart-Build und App-Deployment bleiben logisch getrennt

# CI/CD



# CI/CD



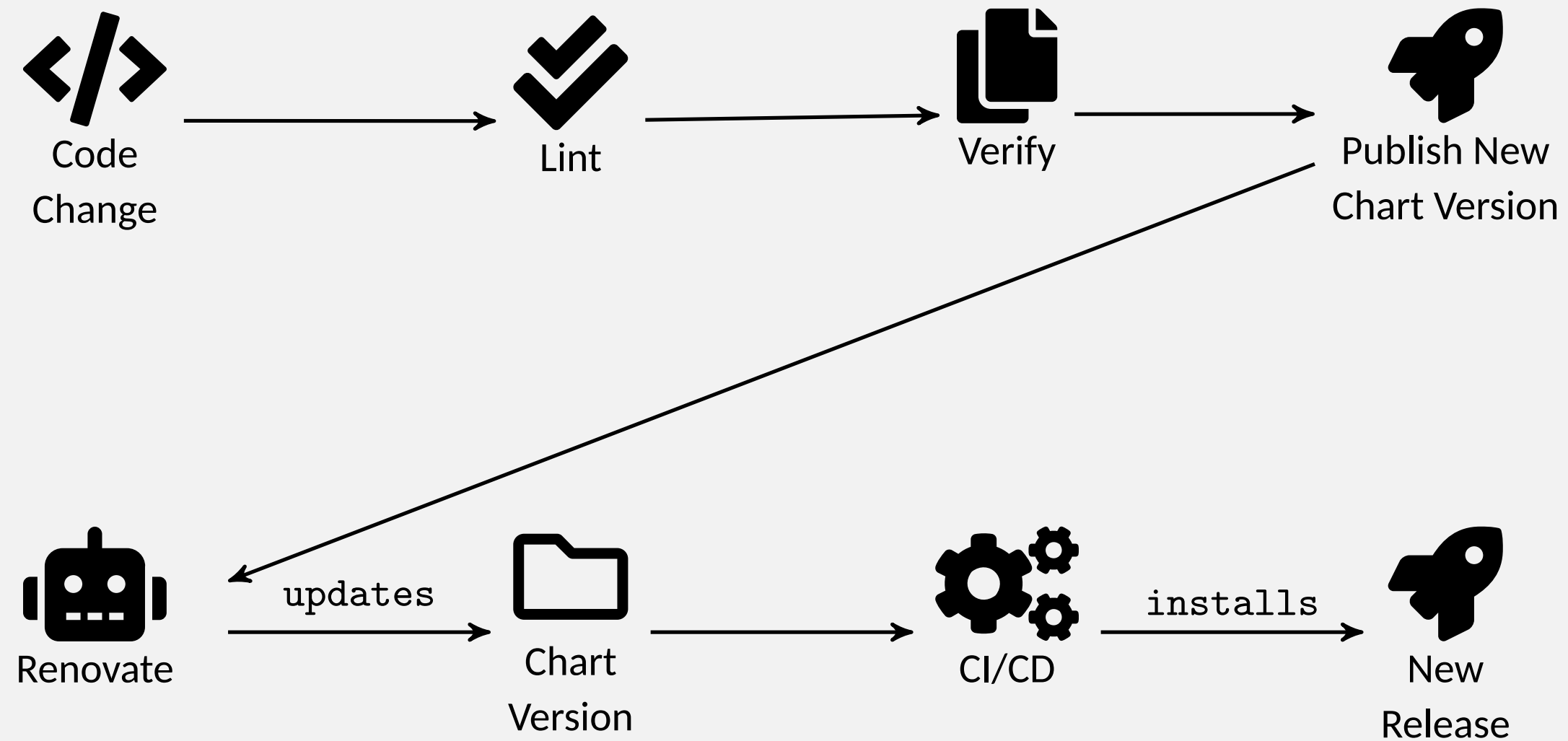


# Automatisierte Updates

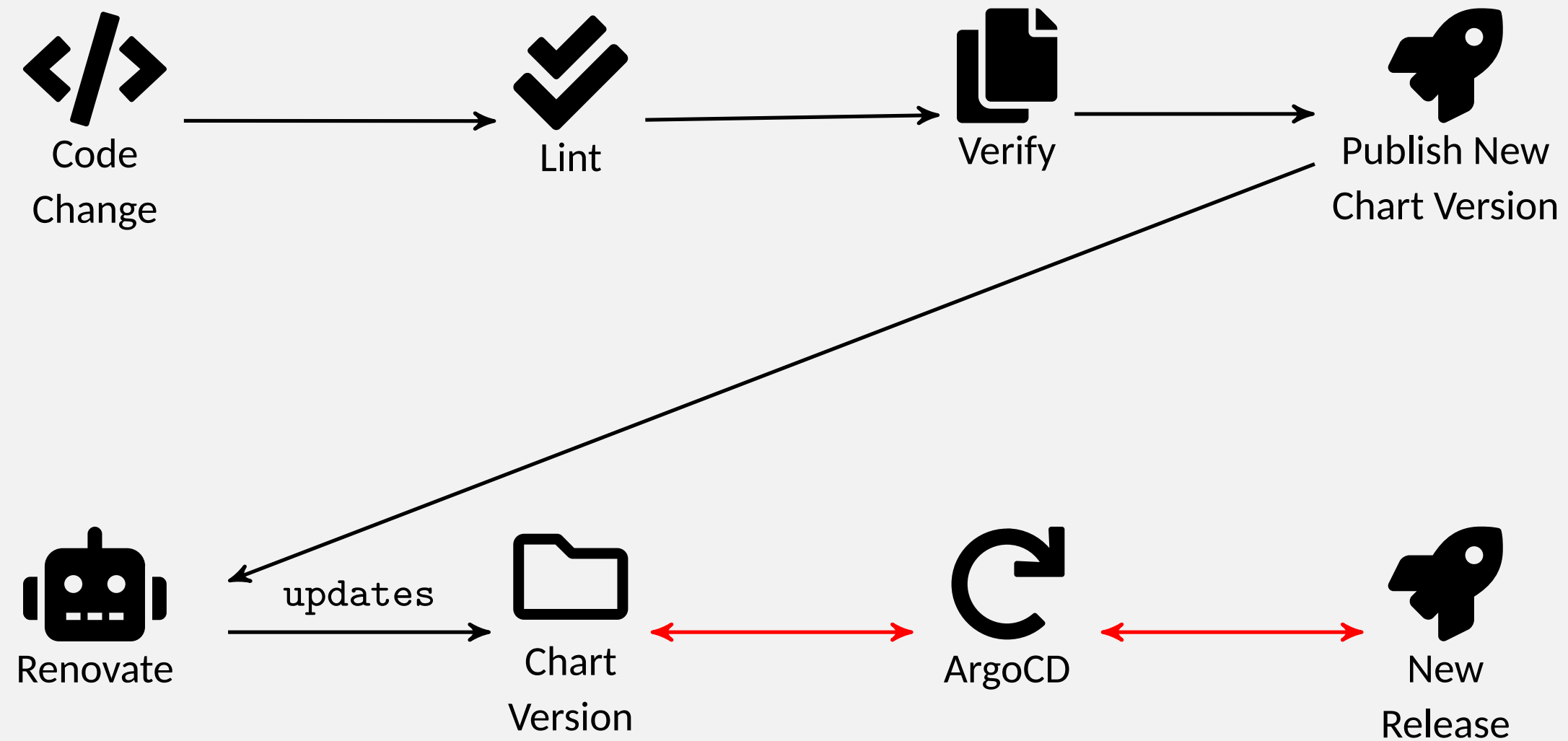
Automatisierte Updates mit Renovate oder Dependabot:

- Kann Versionen in `Chart.yaml` automatisch aktualisieren
- Kann Versionen in `helm upgrade` Befehlen in CI-Skripten aktualisieren

# CI/CD



# CI/CD



# Klassische Pipeline vs. GitOps

## Klassisch:

- CI/CD pusht aktiv in den Cluster (`helm upgrade`)
- Schneller Start, aber Credentials und Drift-Risiko in der Pipeline

## GitOps:

- CI erzeugt Artefakte und ändert nur Git (Desired State)
- Argo CD oder Flux synchronisiert Git -> Cluster

# GitOps-Flow mit Helm (vereinfacht)

1. Chart-Pipeline baut und veröffentlicht `webshop-1.4.2.tgz`
2. Deployment-Repo wird auf neue Chart-Version aktualisiert
3. GitOps-Controller erkennt Git-Änderung
4. Controller führt Helm-Release im Cluster aus

# Takeaways

- Helm Charts als versionierte Artefakte behandeln
- Projekte nur über feste Chart-Versionen deployen
- GitOps als Standard in Betracht ziehen

# Fragen? Anmerkungen? Kritik?

Kontakt: <https://corewire.de> | team@corewire.de | 0761 88 78 153 - 0