Thema 10: Continuous Integration und Jenkins

Daniel Wolfschmidt

02.09.2018

Inhaltsverzeichnis

”The most powerful tool we have as developers is automation“

Scott Hanselman
Programmer, teacher and speaker
Microsoft Web Platform Team

Einleitung

Heutige Probleme:

- Immer schnellere Entwicklungszyklen
- Punkt2
- Punkt3
- Punkt4

Einleitung

Heutige Probleme:

- Immer schnellere Entwicklungszyklen
- Punkt2
- Punkt3
- Punkt4

Deshalb CI einführen!

Continuous Integration

Ablauf dieses Teiles

- Begriffsklärung und Abgrenzung zu ähnlichen Begriffen
- Ablauf von Continuous Integration
- Gründe für den Einsatz von Continuous Integration
- Mögliche Verbesserungen

CI nach Martin Fowler

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible"

Martin Fowler
Geistiger Vater von CI
Thoughtworks

CI nach Martin Fowler

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible"

Martin Fowler
Geistiger Vater von CI
Thoughtworks

- kollaborativ
- häufiges Integrieren
- Nachweis des Erfolgs
- Automatisierung von Schritten

CI nach R.Owen Rogers

"The practice of continuous integration represents a fundamental shift in the process of building software. It takes integration, commonly an infrequent and painful exercise, and makes it a simple, core part of a developer's daily activities. Integrating continuously makes integration a part of the natural rhythm of coding, an integral part of the test-code-refactor cycle. Continuous integration is about progressing steadily forward by taking small steps."

R. Owen Rogers
Entwickler und Konferenzsprecher
Thoughtworks

CI nach R.Owen Rogers

"The practice of continuous integration represents a fundamental shift in the process of building software. It takes integration, commonly an infrequent and painful exercise, and makes it a simple, core part of a developer's daily activities. Integrating continuously makes integration a part of the natural rhythm of coding, an integral part of the test-code-refactor cycle. Continuous integration is about progressing steadily forward by taking small steps."

R. Owen Rogers
Entwickler und Konferenzsprecher
Thoughtworks

- Auswirkungen im Fokus
- „Test-Code-Refactor"
- Teamaspekt eher implizit
- Sichtweisen decken sich

Zusammenfassend

Continuous Integration ist eine Praxis in der Softwareentwicklung, bei der folgende Punkte von zentraler Bedeutung sind:

Zusammenfassend

Continuous Integration ist eine Praxis in der Softwareentwicklung, bei der folgende Punkte von zentraler Bedeutung sind:

- Regelmäßige Integration in eine gemeinsame Codebasis

Zusammenfassend

Continuous Integration ist eine Praxis in der Softwareentwicklung, bei der folgende Punkte von zentraler Bedeutung sind:

- Regelmäßige Integration in eine gemeinsame Codebasis
- **Automatisierter** Nachweis des Integrationserfolgs

Zusammenfassend

Continuous Integration ist eine Praxis in der Softwareentwicklung, bei der folgende Punkte von zentraler Bedeutung sind:

- Regelmäßige Integration in eine gemeinsame Codebasis
- **Automatisierter** Nachweis des Integrationserfolgs
- **Automatisierte** Tests

Zusammenfassend

Continuous Integration ist eine Praxis in der Softwareentwicklung, bei der folgende Punkte von zentraler Bedeutung sind:

- Regelmäßige Integration in eine gemeinsame Codebasis
- **Automatisierter** Nachweis des Integrationserfolgs
- **Automatisierte** Tests
- Schnelle Feedback Zyklen

Continuous Delivery

"You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems. Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production. To do this you use a DeploymentPipeline."

Martin Fowler
Geistiger Vater von CI
Thoughtworks

Continuous Delivery

"You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems. Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production. To do this you use a DeploymentPipeline."

Martin Fowler
Geistiger Vater von CI
Thoughtworks

- Erweiterung von CI
- Schritte bis zum Kunden
- Auch umfangreichere Tests
- Ziel ist rechtzeitig zu liefern

Continuous Deployment

"The concept of continuous deployment, i.e. the ability to deliver software functionality frequently to customers [...]"

Helena Holmström Olsson
Dozentin für Computerwissenschaften
Universität Malmö

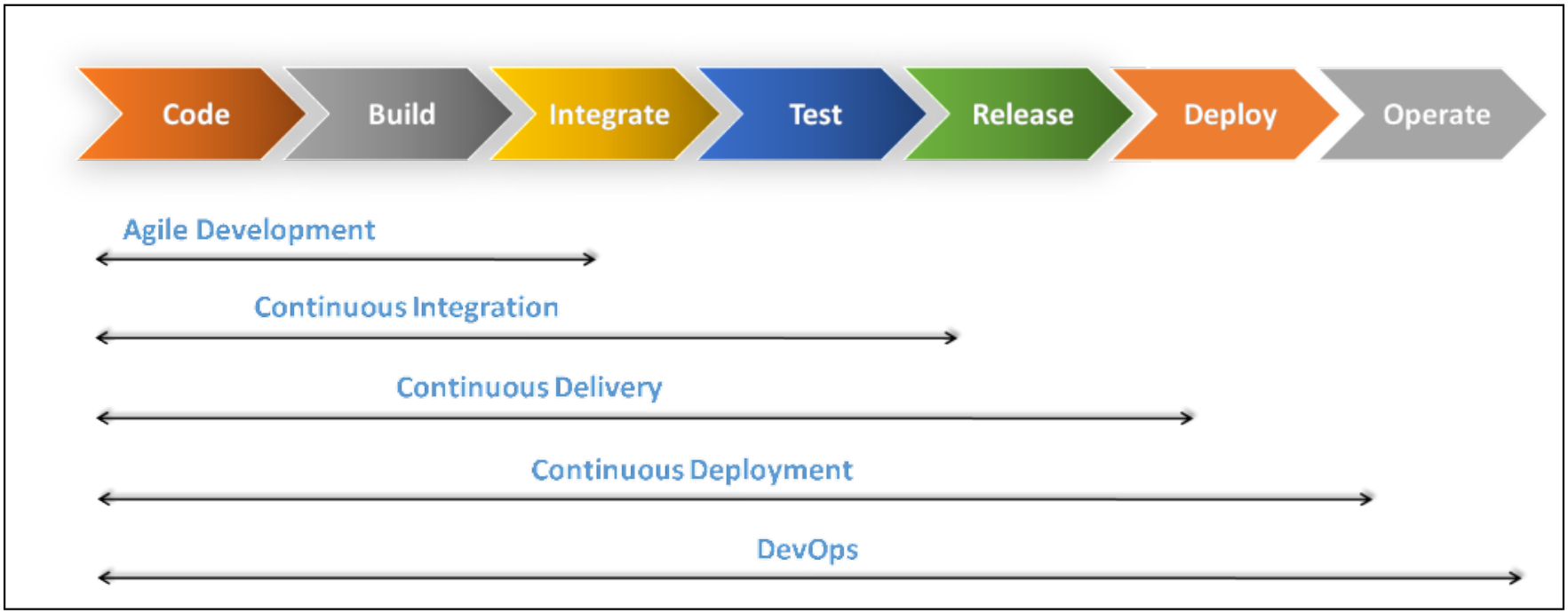
Continuous Deployment

"The concept of continuous deployment, i.e. the ability to deliver software functionality frequently to customers [...]"

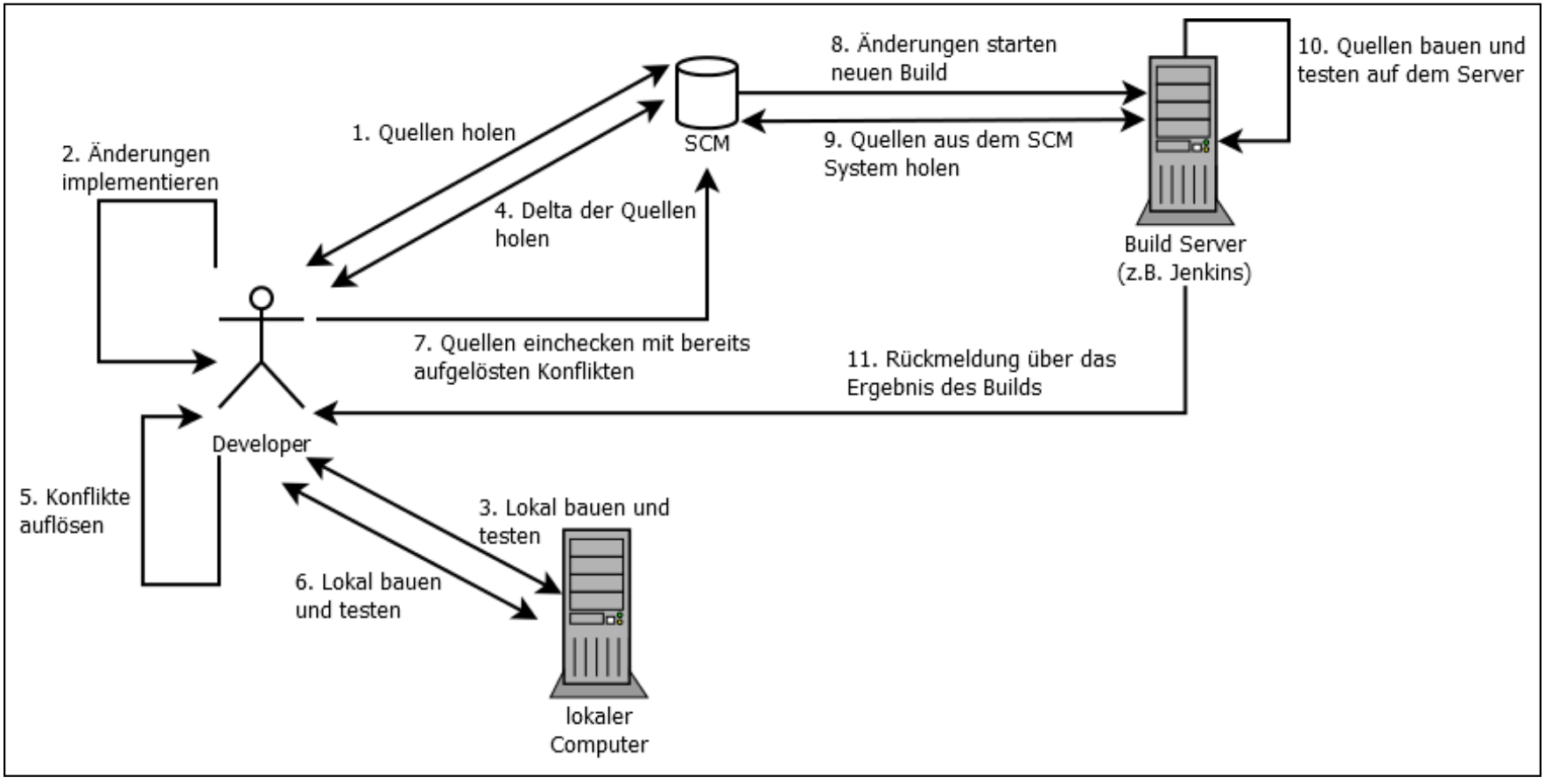
Helena Holmström Olsson
Dozentin für Computerwissenschaften
Universität Malmö

- Einen Schritt weiter als CD
- CD: Fester Lieferzeitpunkt
- Hier: Potentiell immer Lieferbar
- Am weitesten Automatisiert

Übersicht und Einordnung

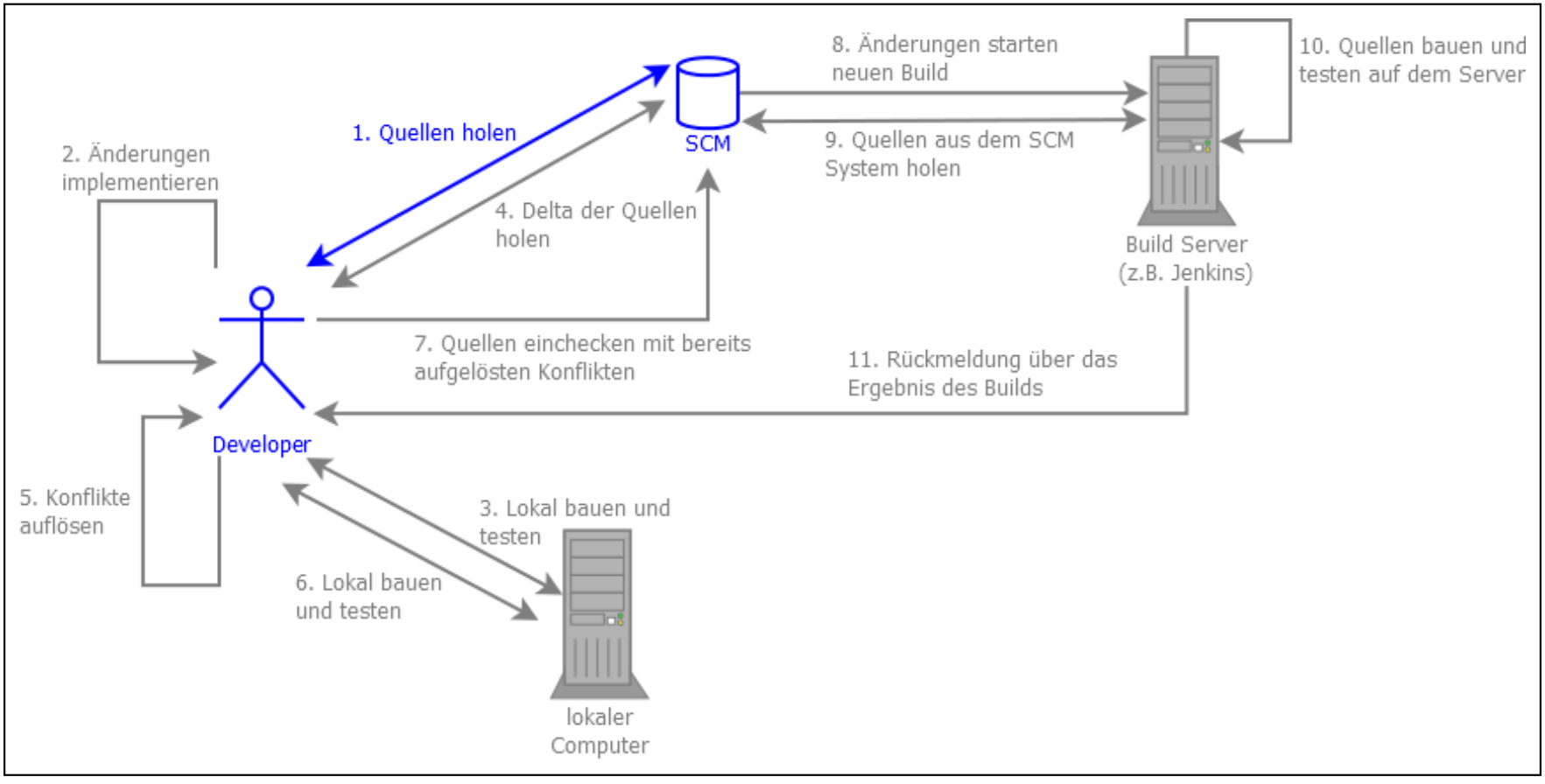


Ablauf nach Martin Fowler



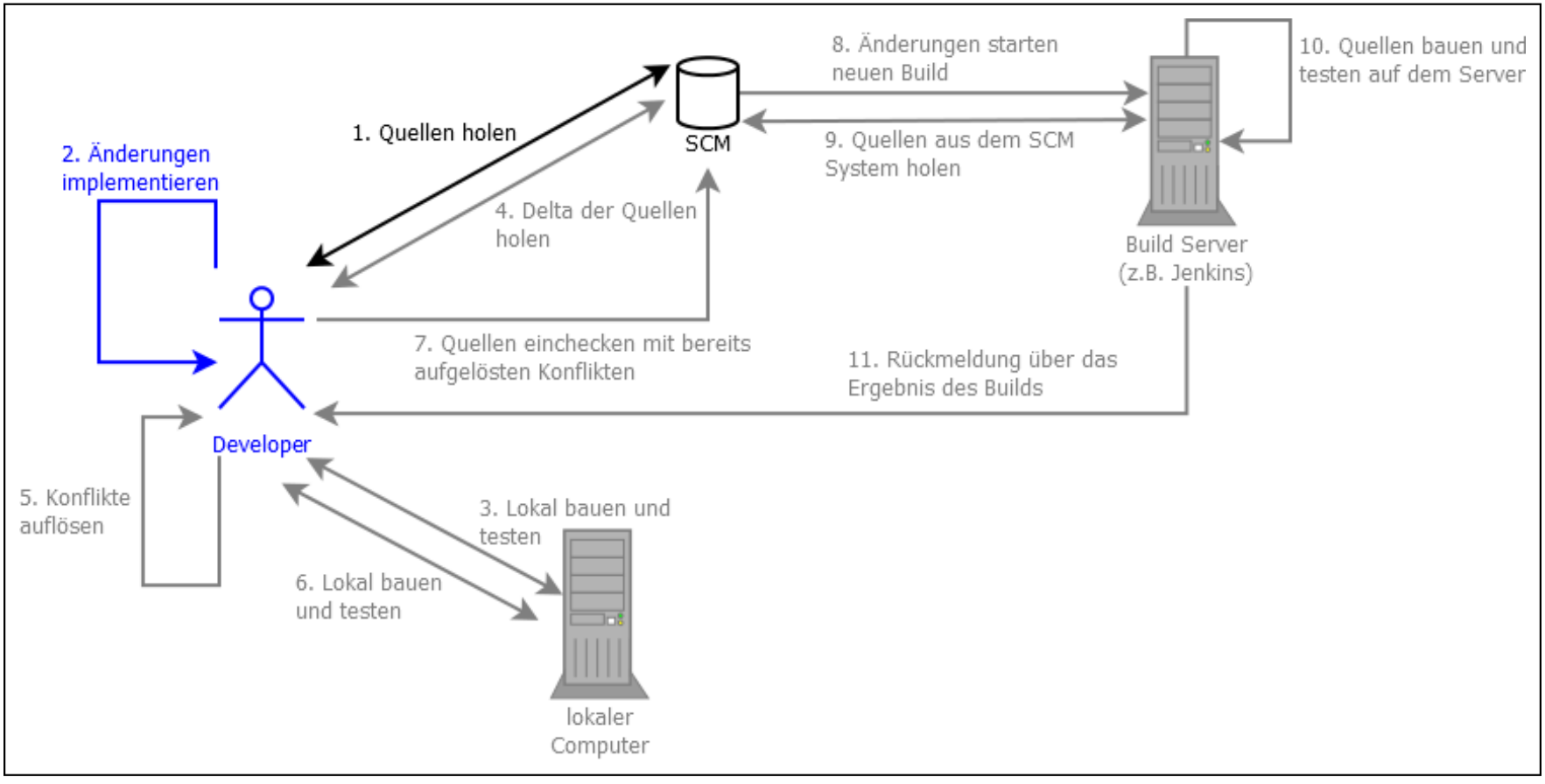
Schematischer Ablauf von CI, wie er von Martin Fowler beschrieben wird

Baseline holen



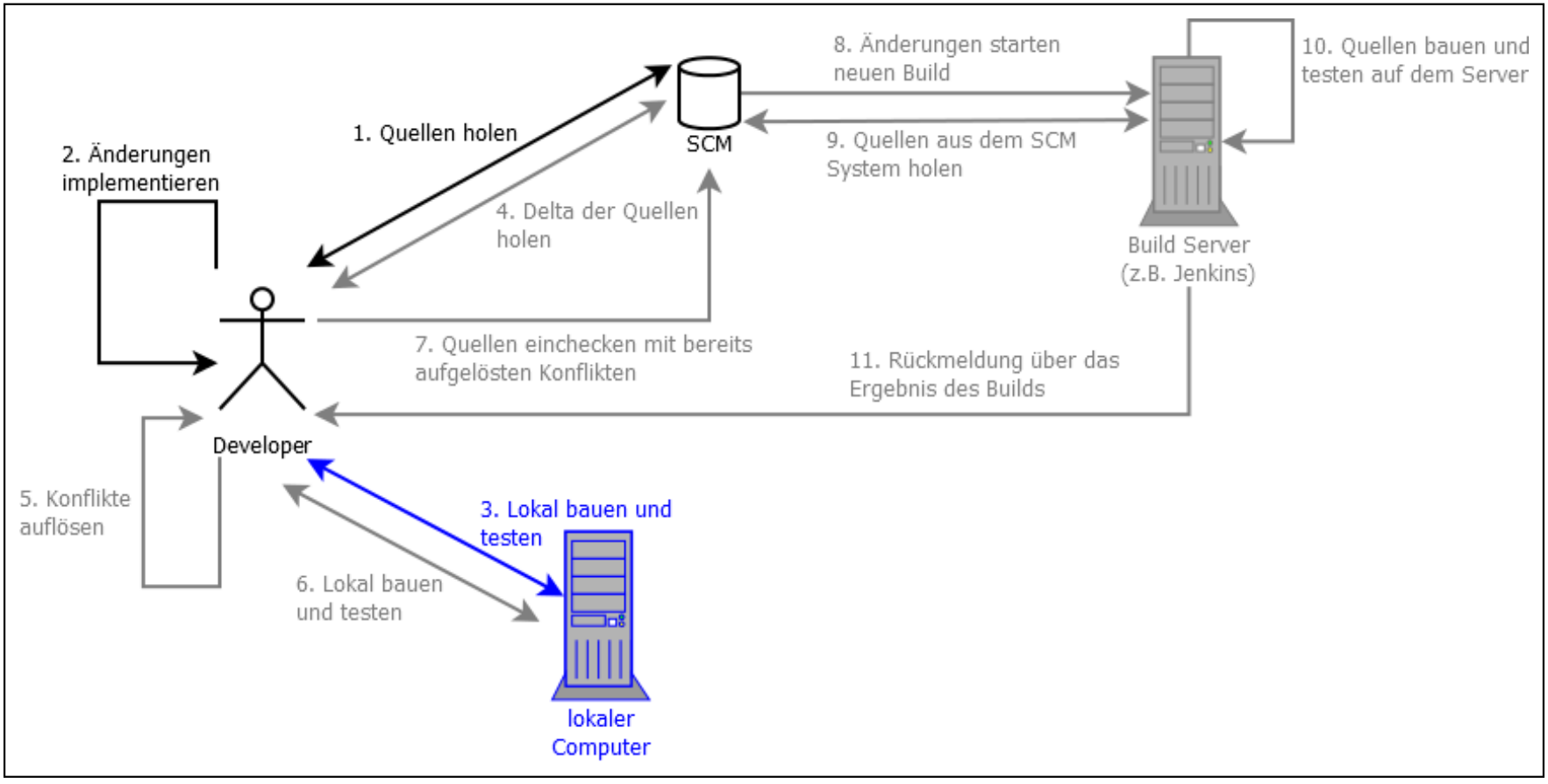
Entwickler holt sich Baseline

Implementierung durchführen



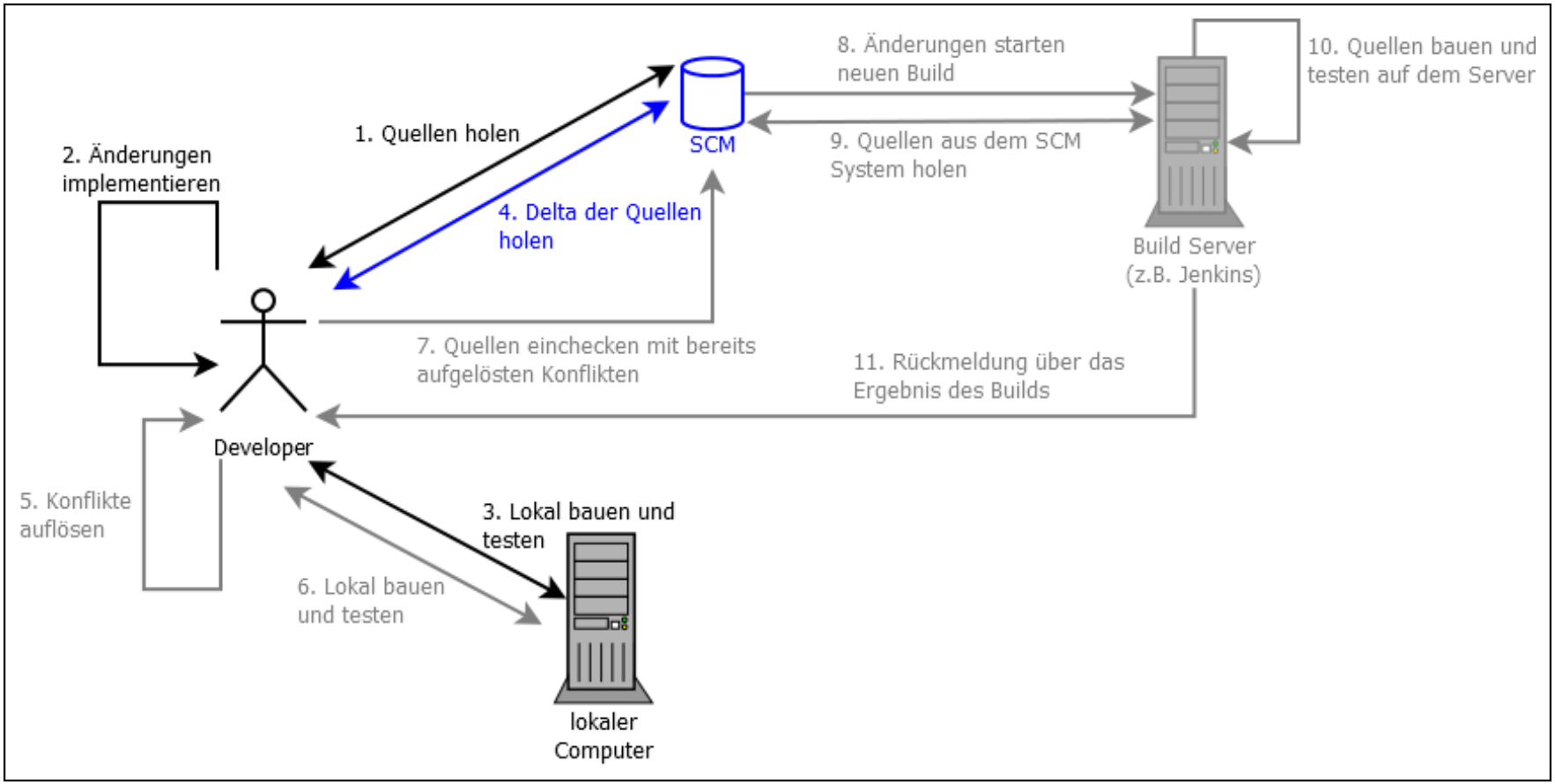
Die angestrebten Implementierungen werden durchgeführt

Lokaler Build und Test



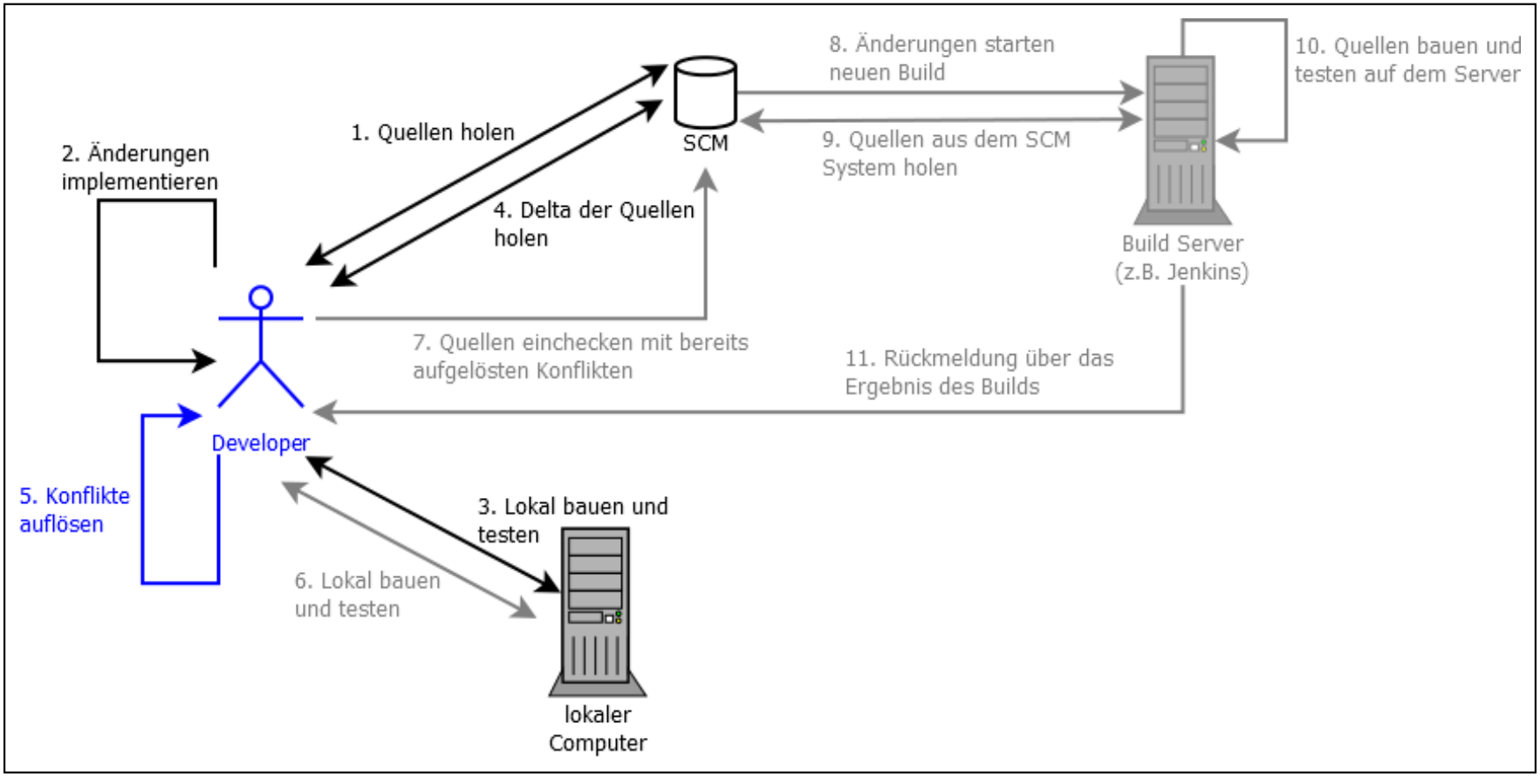
Lokaler Build mit Tests als erstes Quality Gate

Source Code Delta holen



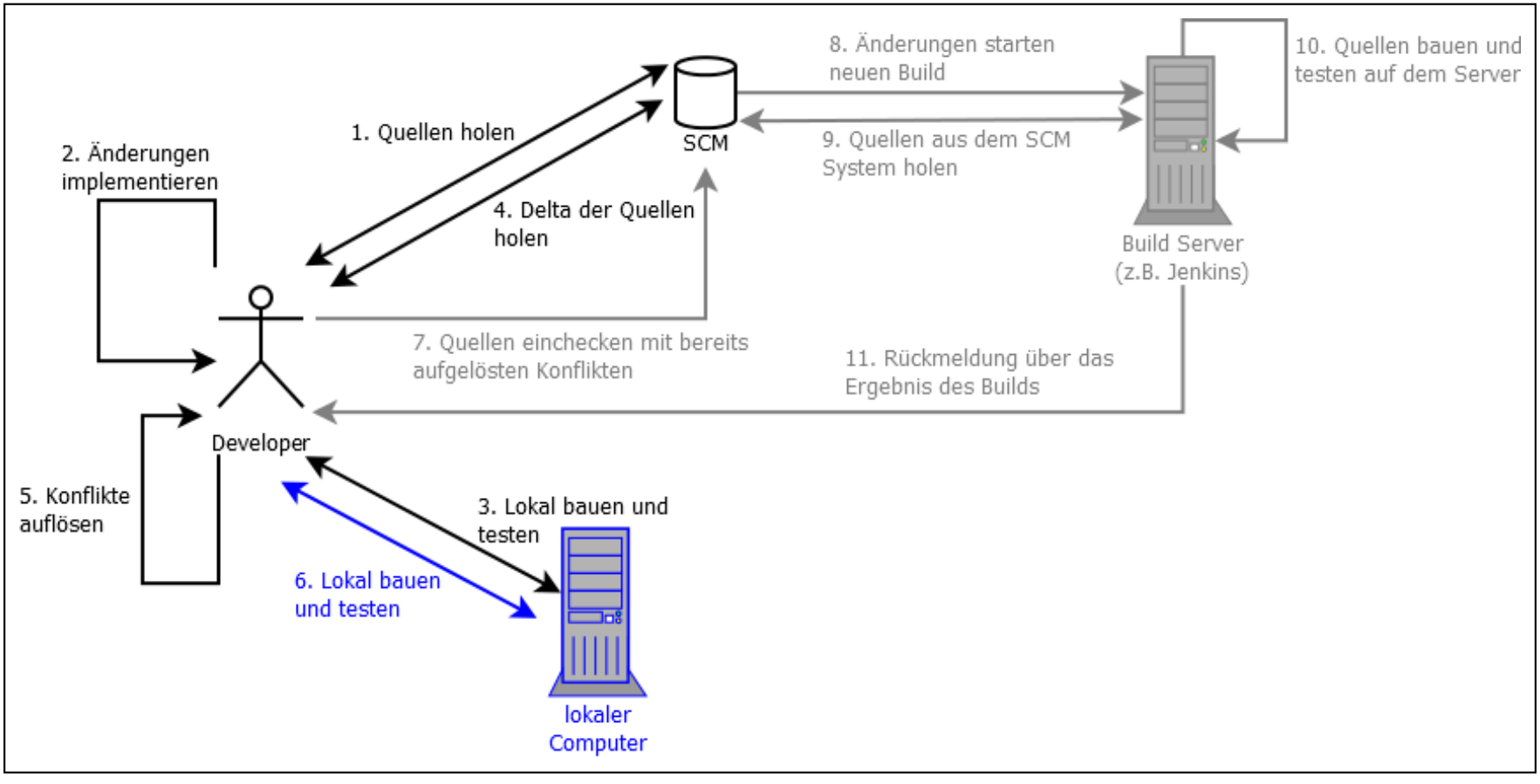
Zwischenzeitlich aufgelaufene Änderungen anderer Entwickler vom SCM holen

Konflikte lösen



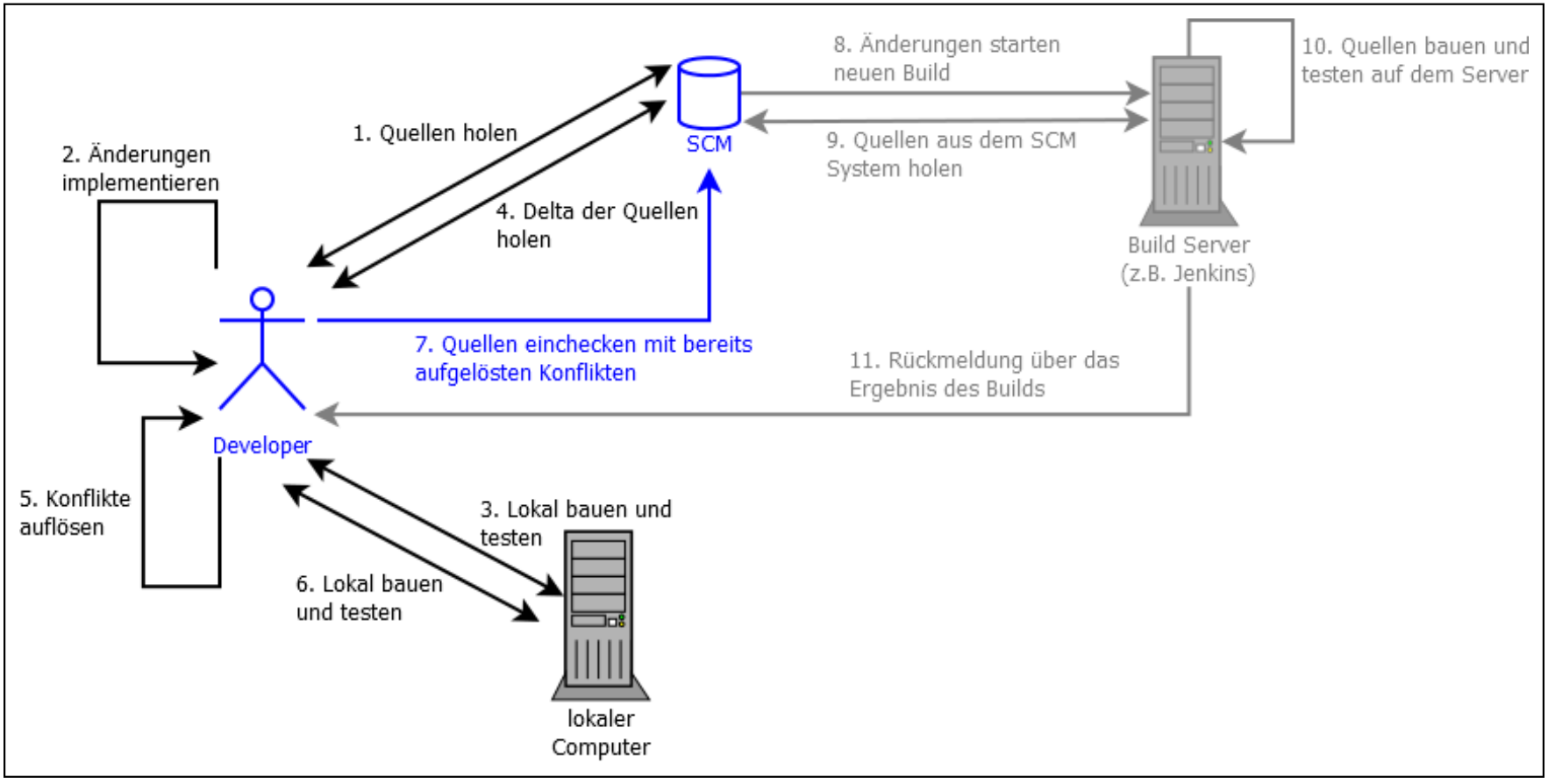
Neu entstandene Konflikte lokal auflösen

Lokaler Build II



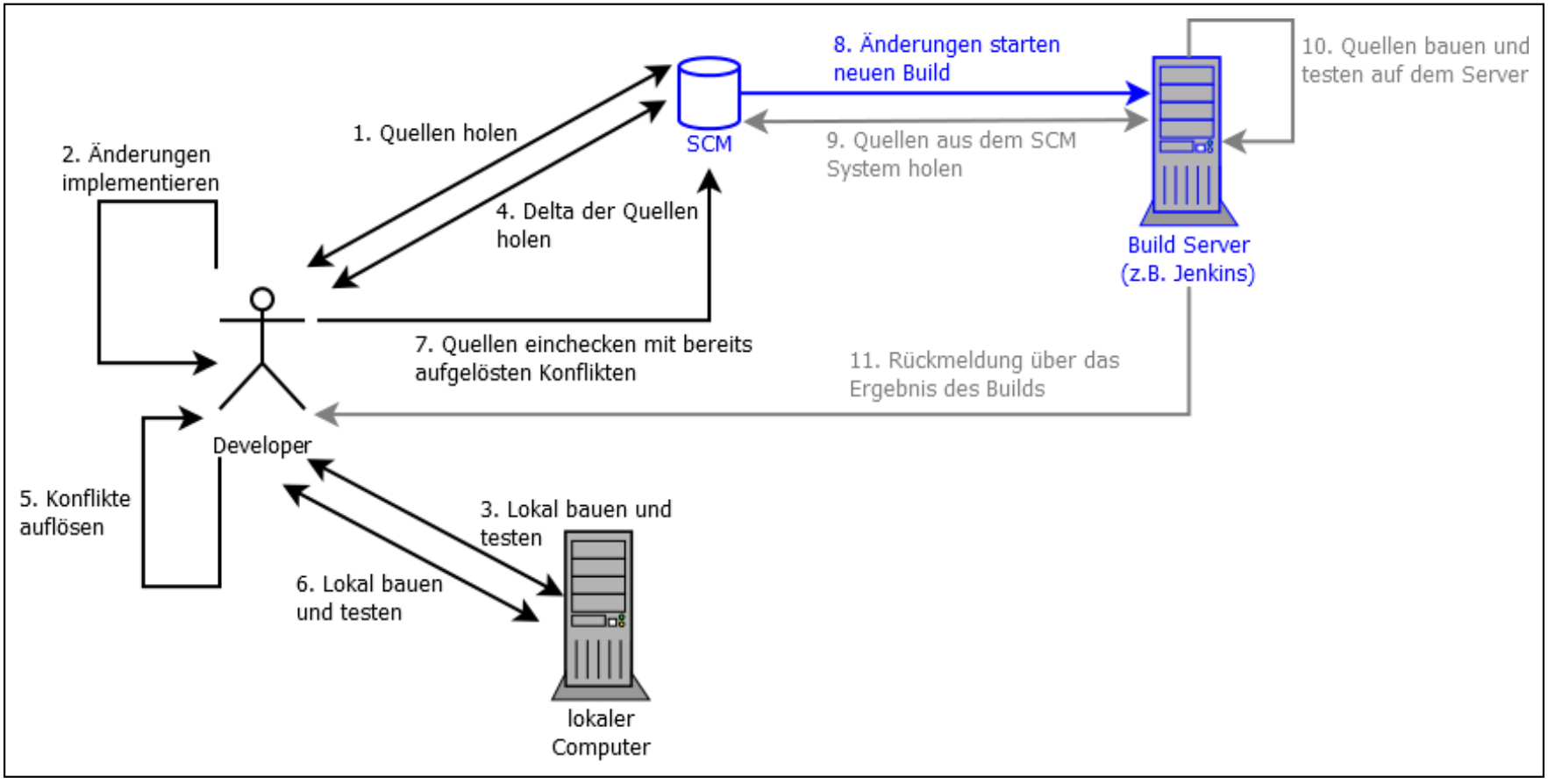
Lokaler Build mit Tests als letzes Quality Gate vor dem Checkin

Checkin



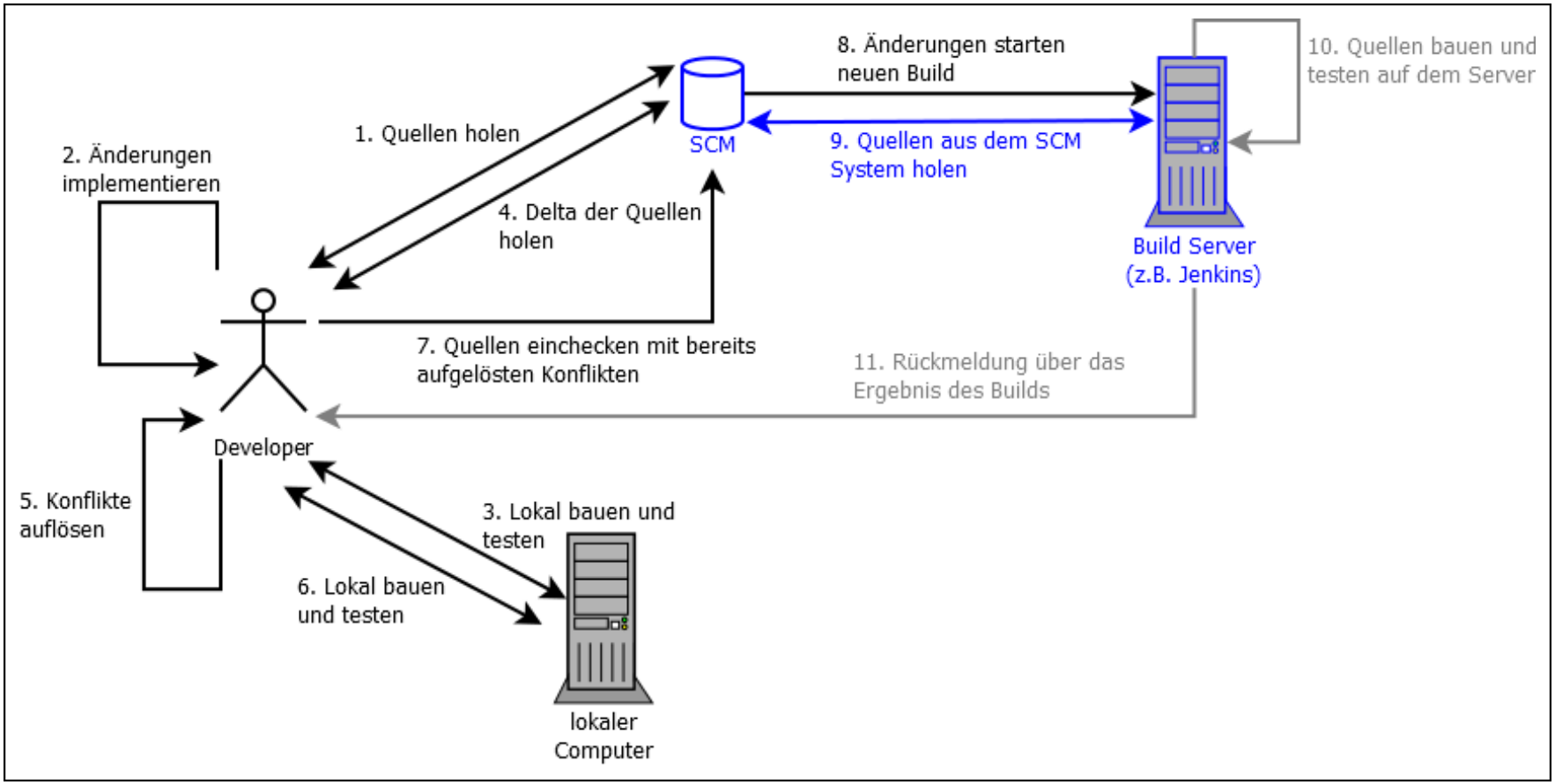
Nach erfolgreichem lokalen Build sind die Änderungen reif für den Checkin

Trigger Build



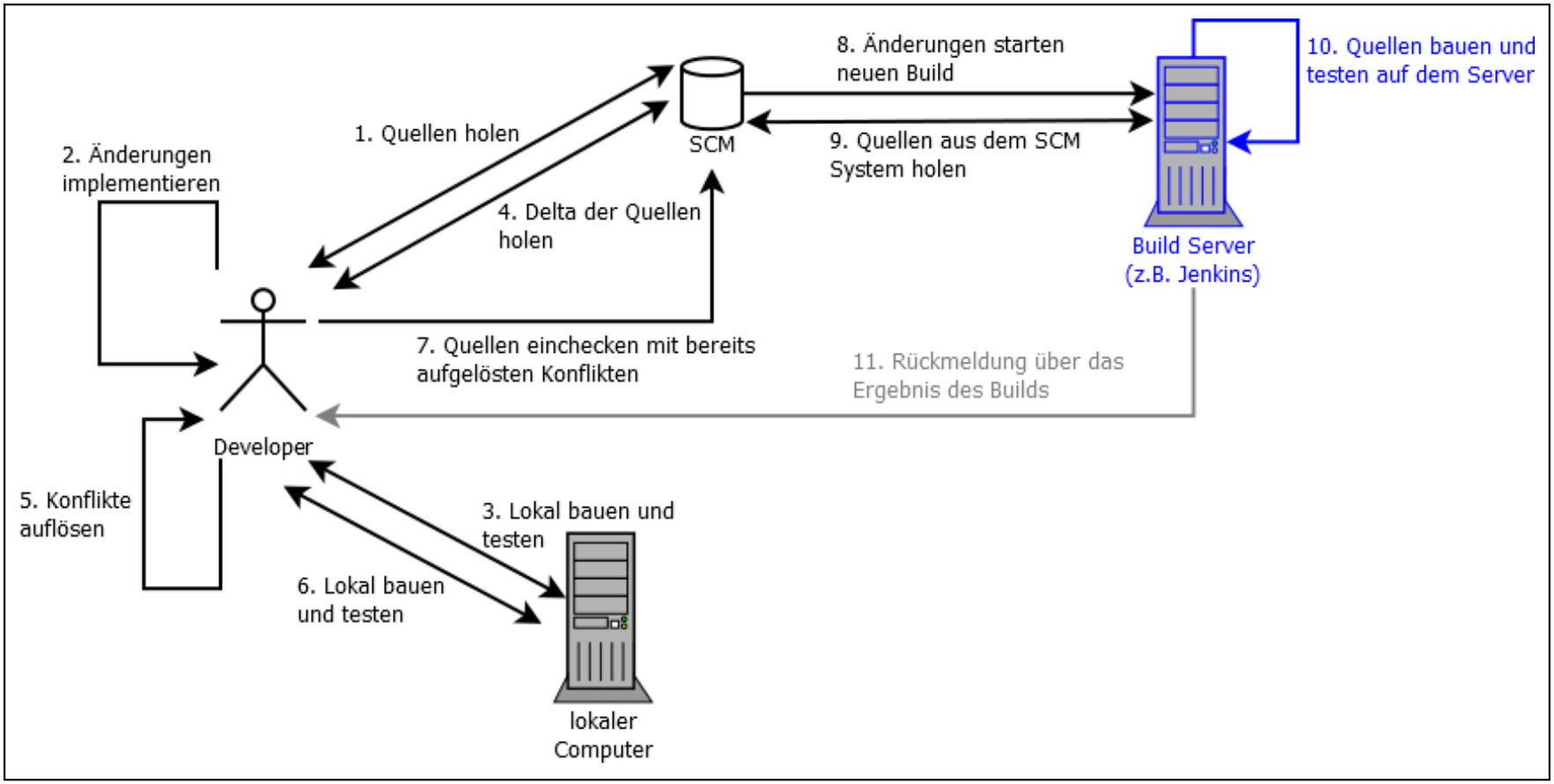
Ein automatischer Build startet, weil im SCM neue Sourcen vorhanden sind

Aktuellen Stand holen



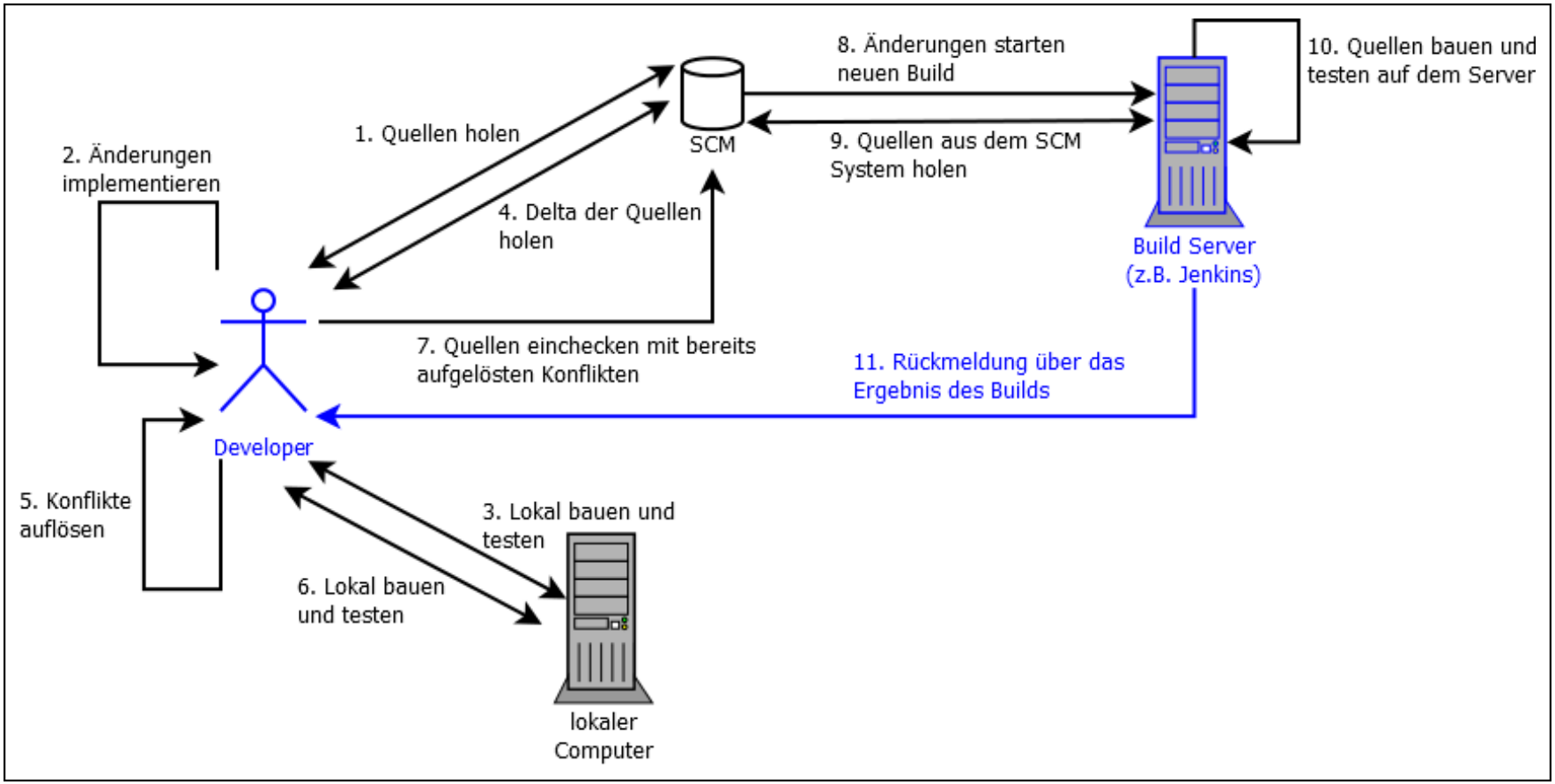
Der Build Server braucht die letzten Sourcen um sie zu bauen und zu testen

Bauen am Server



Der Server Build ist das ultimative Quality Gate im Prozess.
In diesem Schritt wird gebaut und getestet.

Feedback



Das Ergebnis des Builds wird veröffentlicht und ruft eventuell weitere Aktionen des Entwicklers hervor.

Vielen Dank

Vielen Dank für Ihre Aufmerksamkeit!