Bünting
INFORMATIONSTECHNOLOGIE

.NET Oldenburg
user group

Building Software

Christian Thevissen

Agenda

- Einleitung
- Continuous Integration / Contiuous Delivery
- Der Ideale Buildserver
  - Build
  - Validation
  - Integration in die Entwicklungsinfrastruktur
  - Das Ergebnis
  - Deployment
  - Security

Einleitung

- Was sollte ein Buildserver können
- Hinweise zur Umsetzung im Team Foundation Server

Continuous Integration / Continuous Delivery

- Continuous Integration ([FOWLER])
  - Technik aus der Softwareentwicklung
  - Code soll regelmäßig „Integriert“ werden
  - Code soll vorher validiert werden (Builds, Tests, Codeanalyse)
- Continuous Delivery ([CD_WIKI])
  - Technik aus der Softwareentwicklung
  - Integrations- und Auslieferungsprozess soll automatisiert werden
  - Deployment soll dadurch schnell, zuverlässig und wiederholbar werden

[FOWLER]: http://www.martinfowler.com/articles/continuousIntegration.html, Martin Fowler
[CD_WIKI]: https://de.wikipedia.org/wiki/Continuous_Delivery, Diverse Autoren

Der Ideale Buildserver - Build

- Build
  - Vielzahl von Projekten mit unterschiedlichen Typen pro Build möglich.
  - Unterschiedliche Buildartefakte als Ergebnis eines Builds
  - Abhängigkeiten müssen verwaltet werden (Maven, NuGet)
  - Flexibilität (Sprachen, Plattformen, Versionsverwaltung, etc.)
- Trigger
  - Manuell
  - CheckIn Trigger
    - Gated CheckIns
    - Rollin Builds
  - Scheduled Builds

Der Ideale Buildserver - Build

- Team Foundation Server
  - Nicht Microsoft Sprachen und Plattformen werden erst unter „Build VNext“ ab Team Foundation Server 2015 unterstützt.
  - Visual Studio auf Buildagent installieren (Xaml Build)

Der Ideale Buildserver - Validation

- Automatisierte Tests
  - Unit Tests (Gut geeignet für CI- Builds)
  - Akzeptanztests
  - Integrationstests
  - etc.
- Statische Codeanalyse
  - Vorsicht kann sehr lange Dauern
- Eigene Scripte

Der Ideale Buildserver - Validation

- Team Foundation Server
  - Einbinden eigener Scripte (PowerShell) seit Team Foundation Server 2013 ohne größen Aufwand möglich (Pre- und Post- Build, Pre- und Post Test)

Der Ideale Buildserver - Integration in die Entwicklungsinfrastruktur

- Verbindung mit
  - Versionsverwaltung
  - Ticketsystem
- Im Team Foundation Server ist das ein System
  - Items der einzelnen Subsystem werden miteinander verknüpft (z.B. ein Changeset mit einem WorkItem und einem Build).
  - Information können auch programmatisch abgefragt werden (TFS- API)

Der Ideale Buildserver - Das Ergebnis

- Buildartefakte
  - Deploybare Anwendungen, Datenbanken, etc
  - NuGet oder Maven packages
- Informationen
  - Symbolfiles
  - Build- und Testergebnisse
  - Custom Reports

Der Ideale Buildserver - Das Ergebnis

- Team Foundation Server (Beispiel für Custom Reports)
  - Voraussetzung: Angabe von WorkItems bei CheckIn verpflichtend
  - Für Release kann ermittelt werden, welche WorkItems und Changesets zu dieser Version geführt haben.

Der Ideale Buildserver - Deployment

- Automatisiertes deployment reduziert die Manuellen Aufwände beim deployment und bei der Dokumentation.
- Schnelleres Deployment macht mein Ergebnis früher Überprüfbar.

Der Ideale Buildserver - Deployment

- Automatisches Deployment auf ein Entwicklungs- /Testsystem (z.B. Nightly Build)
- Automatisches Deployment eines Release auf alle Stages (z.B. Microsoft Releasemanagement)

Security

- Es sollten nur die Personen Änderungen an dem System vornehmen, von denen sie auch wollen, dass sie dies tun.
- Anzeige von Informationen muss sich auch auf einen bestimmten Personenkreis einschränken lassen.
- Wählen sie Sinnvolle Einschränkungen

Vielen Dank


Bünting
INFORMATIONSTECHNOLOGIE


.NET
user group
Oldenburg