Modul 09

Fortgeschrittene Themen

Lernziele

Nach diesem Modul kannst du:

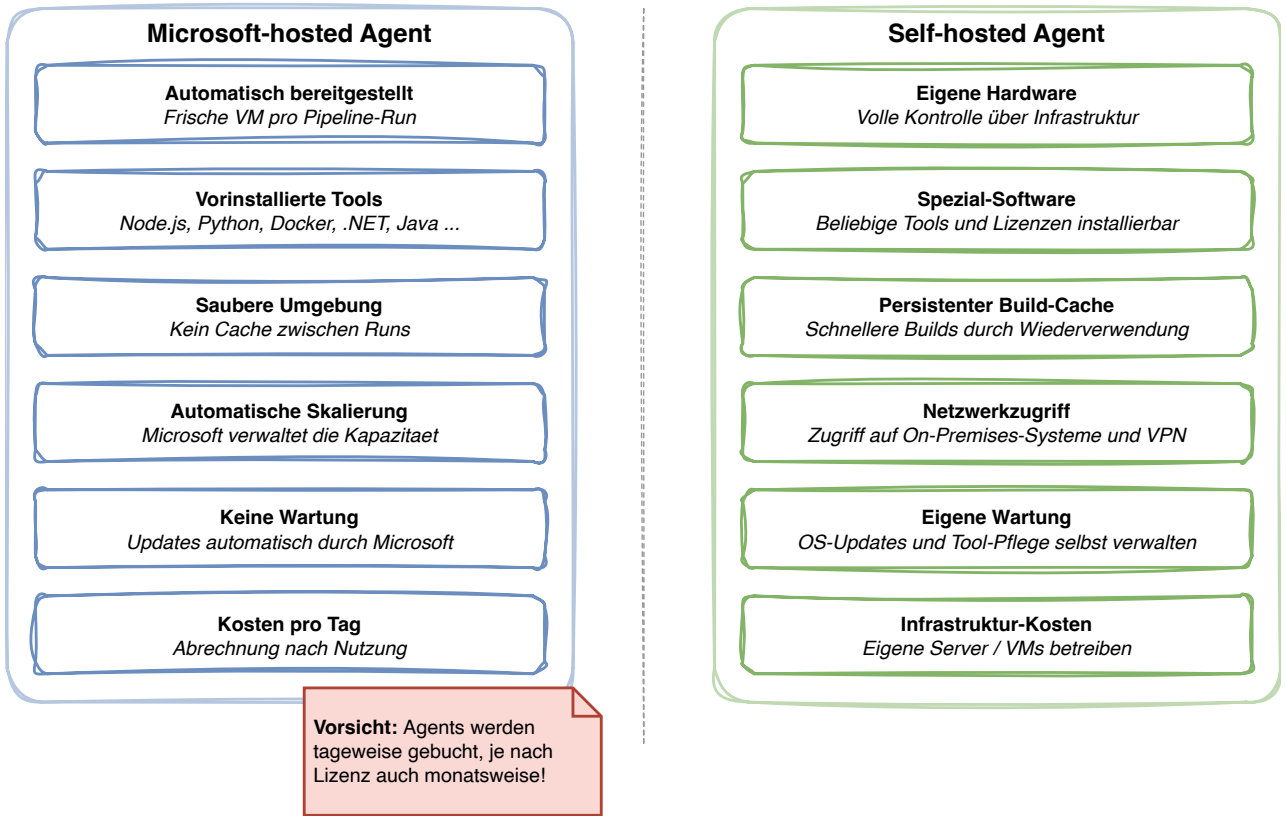
- **Self-hosted Agents einrichten und verwalten** - Einen eigenen Build-Agent auf Linux installieren und als systemd-Service betreiben
- **Pipeline-Templates erstellen und wiederverwenden** - Step-, Job-, Stage- und Variable-Templates organisationsweit einsetzen

1. Self-hosted Agents: Warum und Wann?

Microsoft-hosted Agents sind bequem, aber manchmal reichen sie nicht:

- **Spezielle Software oder Hardware** - Lizenzierte Tools, GPUs, besondere Compiler
- **Netzwerkzugriff** - Zugriff auf On-Premises-Systeme, Datenbanken hinter Firewalls, VPN-Verbindungen
- **Build-Performance** - Leistungsstärkere Maschinen, persistenter Build-Cache für schnellere Builds
- **Compliance** - Anforderungen an den Build-Ort (z.B. Daten dürfen die eigene Infrastruktur nicht verlassen)

Agent-Vergleich



1.1 Agent-Installation auf Linux

```
# Verzeichnis erstellen
mkdir -p ~/azagent && cd ~/azagent

# Agent herunterladen
curl -fkSL -o vsts-agent-linux-x64.tar.gz \
  https://vstsagentpackage.azureedge.net/agent/3.248.0/\
vsts-agent-linux-x64-3.248.0.tar.gz

# Entpacken
tar zxvf vsts-agent-linux-x64.tar.gz

# Konfiguration starten
./config.sh
# Server URL:  https://dev.azure.com/<org>
# Auth:        PAT
# Agent Pool:  linux-training-pool
# Agent Name:  training-agent-01
```

1.2 Agent-Pool-Konfiguration

Ein **Agent Pool** gruppiert Agents für die Zuweisung an Pipelines:

1. **Organization Settings > Agent pools > Add pool**
2. Pool type: **Self-hosted**
3. Name: z.B. `linux-training-pool`
4. **Grant access permission to all pipelines** aktivieren

```
# Pipeline referenziert den Pool per Name
pool:
  name: 'linux-training-pool'
```

> **Tipp:** Verwende separate Pools für verschiedene Zwecke (Build, Test, Deploy) oder Umgebungen (Dev, Prod).

1.3 systemd Service für den Agent

Für den Produktionsbetrieb sollte der Agent als **systemd-Service** laufen:

```
# Service installieren und starten
sudo ./svc.sh install
sudo ./svc.sh start

# Status prüfen
sudo ./svc.sh status

# Service stoppen und deinstallieren
sudo ./svc.sh stop
sudo ./svc.sh uninstall
```

- Der Agent startet **automatisch beim Booten**
- Neustart bei Absturz wird durch systemd verwaltet
- Agent-Updates werden automatisch eingespielt

1.4 Capabilities und Demands

Self-hosted Agents melden ihre installierten Tools als **Capabilities**. Pipelines können **Demands** stellen:

```yaml
pool:
  name: 'linux-training-pool'
  demands:
    - npm
    - docker
    - Agent.OS -equals Linux
```

Typ	Beschreibung
System Capabilities	Automatisch erkannt (OS, Tools)
User Capabilities	Manuell im Portal hinzugefügt
Demands	Pipeline fordert bestimmte Capability

2. Pipeline-Templates: Überblick

Templates lösen das **Copy-Paste-Problem** bei ähnlichen Pipelines:

- **Wiederverwendbare** Pipeline-Bausteine
- **Parametrisierbar** für flexible Nutzung
- **Versionierbar** in eigenen Repositories
- **Organisationsweit** einsetzbar

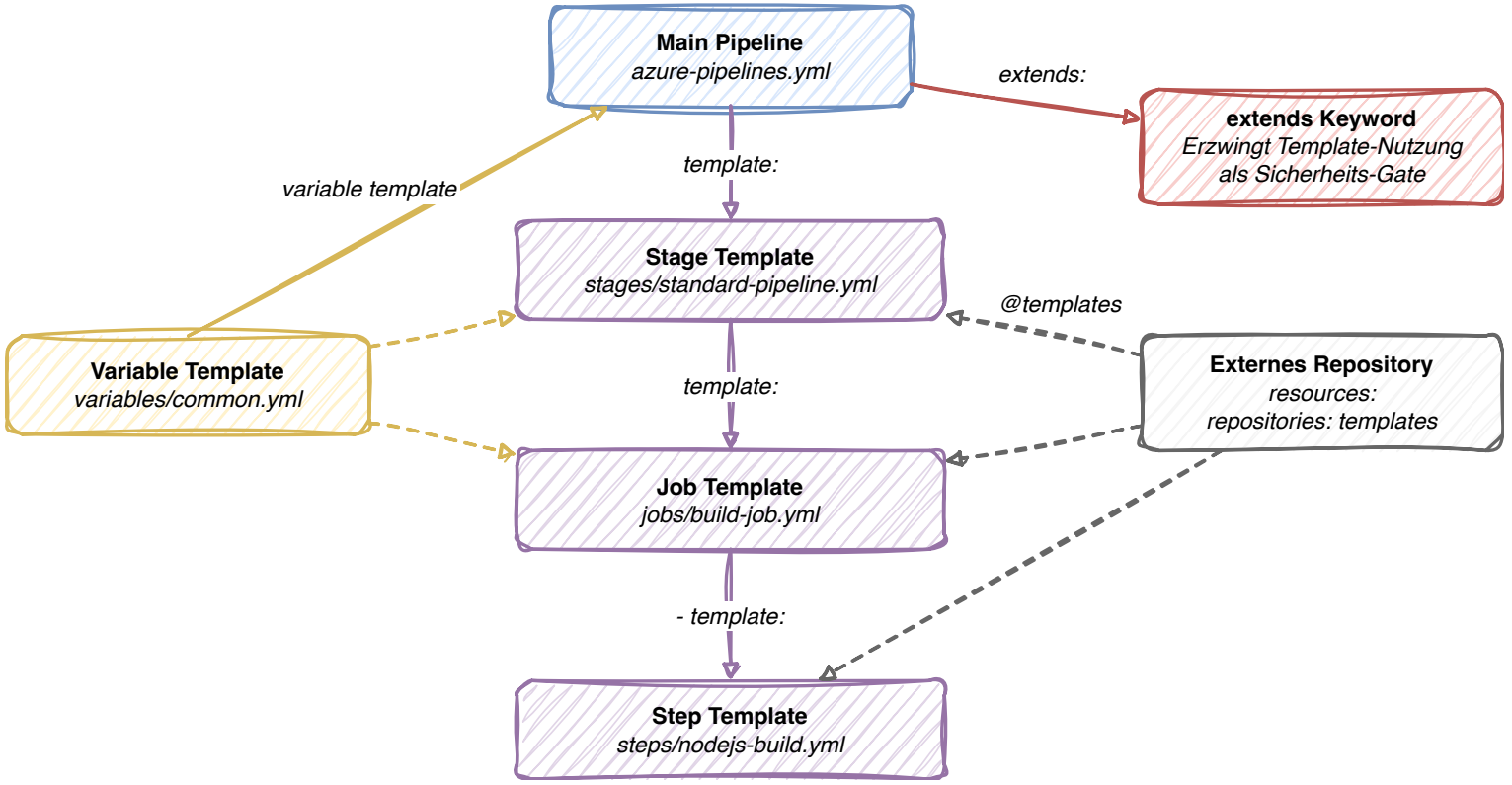
> Ohne Templates: Änderung an 50 Pipelines = 50 manuelle Edits.
>
> Mit Templates: Änderung am Template = alle Pipelines aktualisiert.

2.1 Template-Typen

Template-Typ	Inhalt	Einbindung
Step Template	Wiederverwendbare Steps	`steps: - template: steps/x.yml`
Job Template	Wiederverwendbare Jobs	`jobs: - template: jobs/x.yml`
Stage Template	Wiederverwendbare Stages	`stages: - template: stages/x.yml`
Variable Template	Gemeinsame Variablen	`variables: - template: vars/x.yml`

Verschachtelung möglich: Stage Template nutzt Job Template, das wiederum Step Templates einbindet.

Template-Hierarchie



2.2 Template-Parameter und Validierung

Templates akzeptieren **typisierte Parameter** mit Validierung:

```yaml
parameters:
  - name: nodeVersion
    type: string
    default: '20.x'
  - name: publishArtifact
    type: boolean
    default: true
  - name: environment
    type: string
    values:      # Erlaubte Werte
      - dev
      - staging
      - production
  - name: environments
    type: object
    default:
      - name: dev
      - name: staging
```

> Pflicht-Parameter ohne `default` müssen beim Aufruf angegeben werden.

2.3 extends Keyword

Das `extends` Keyword **erzwingt** die Nutzung eines Templates --
ideal als **Sicherheits-Gate**:

```yaml
# Pipeline MUSS dieses Template verwenden
extends:
  template: stages/secure-pipeline.yml@templates
  parameters:
    nodeVersion: '20.x'
```

- Die Pipeline kann **nur die vom Template erlaubten Steps** nutzen
- Administratoren können `extends` als
  Required Template vorschreiben
- Verhindert, dass Teams unsichere Pipeline-Konfigurationen verwenden

2.4 Externe Repos als Template-Quellen

Templates können aus **anderen Repositories** eingebunden werden:

```yaml
resources:
  repositories:
    - repository: templates
      type: git
      name: pipeline-labs/pipeline-templates
      ref: refs/heads/main    # oder refs/tags/v1.0

stages:
  - template: stages/standard-pipeline.yml@templates
    parameters:
      nodeVersion: '20.x'
```

> **Best Practice:** Verwende Tags (`refs/tags/v1.0`) statt Branches für stabile, versionierte Templates.

2.5 Code-Beispiel: Step Template

```yaml
# steps/nodejs-build.yml
parameters:
  - name: nodeVersion
    type: string
    default: '20.x'
  - name: workingDirectory
    type: string
    default: '$(System.DefaultWorkingDirectory)'

steps:
  - task: NodeTool@0
    inputs:
      versionSpec: ${{ parameters.nodeVersion }}
    displayName: 'Node.js installieren'

  - script: npm ci || npm install
    displayName: 'Abhängigkeiten installieren'
    workingDirectory: ${{ parameters.workingDirectory }}

  - script: npm run build --if-present
    displayName: 'Build ausführen'
    workingDirectory: ${{ parameters.workingDirectory }}
```

2.6 Code-Beispiel: Stage Template

```yaml
# stages/standard-pipeline.yml
parameters:
  - name: nodeVersion
    type: string
    default: '20.x'
  - name: runTests
    type: boolean
    default: true

stages:
  - stage: Build
    jobs:
      - template: ../jobs/build-job.yml
        parameters:
          nodeVersion: ${{ parameters.nodeVersion }}

  - ${{ if parameters.runTests }}:
    - stage: Test
      dependsOn: Build
      jobs:
        - job: RunTests
          pool:
            vmImage: 'ubuntu-latest'
          steps:
            - template: ../steps/nodejs-build.yml
            - template: ../steps/run-tests.yml
```

Labs

- **Lab 19:** Self-hosted Agent auf Linux einrichten
- **Lab 20:** Pipeline-Templates und Wiederverwendung