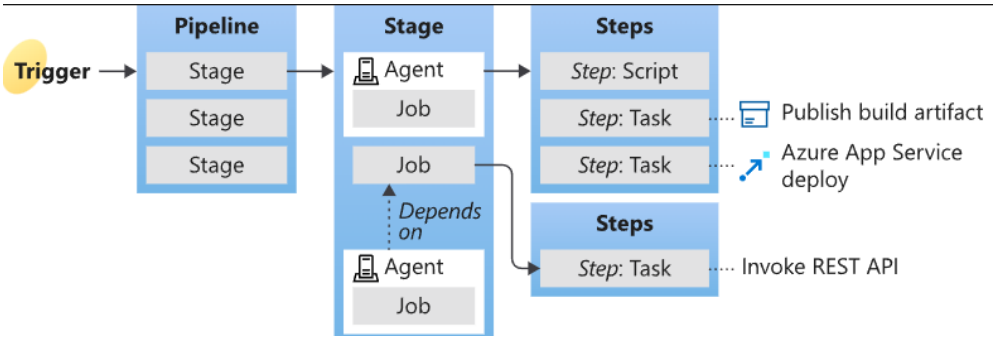
Azure Pipelines



Agent: führt Jobs aus (die aus mehreren Steps bestehen können)
Stage: Organisiert Jobs in einer Pipeline. Stage kann aus mehreren Jobs bestehen
Jobs: läuft auf Agents
Artifact: Collection von Files/Packages die im Zuge eines *runs* gepublisht werden
Deployment group: Set von target Machines die *Agents* installiert haben (Agent Pool)

> häufig existieren mehrere Pipelines, z.B. docu, build, ...

Triggers

> können durch Pull Request getriggert werden
> bei best. Kommentaren
> **Scheduled triggers**: z.B. *nightly build*, event-based triggers -> z.B.: Pull request made od. Commits auf branch pushen

```
  schedules:
- cron: '0 0 * * *'
  displayName: string # friendly name given to a specific schedule
  branches:
    include: [ string ] # which branches the schedule applies to
        - main
        - release
    exclude: [ string ] # which branches to exclude from the schedule
  always: boolean # whether to always run the pipeline or only if there have been source code changes since the last success
  batch: boolean # Whether to run the pipeline if the previously scheduled run is in-progress; the default is false.
  # batch is available in Azure DevOps Server 2022.1 and higher
```

> mit `always: false` läuft pipeline nur wenn Code sich seit dem letzten Run geändert hat, mit `always: true` immer zur angegeben Zeit

Resource triggers

Run upon completion of another pipeline
Können zusätzlich durch `tag filter` oder `stage filter` spezifiziert werden.
Z.B.: läfut sobald `Farbrikam-CI` die beiden Stages durchlaufen hat.

> **Important**
>
> Filter werden immer durch logisches **AND** verknüpft

```
resources:
  pipelines:
  - pipeline: MyCIAlias
    source: Farbrikam-CI
    trigger:
      stages:         # This stage filter is used when evaluating conditions for
      - PreProduction # triggering your pipeline. On successful completion of all the stages
      - Production    # provided, your pipeline will be triggered.
```

läuft nur, wenn die beiden angeführten Filter präsent sind

```
resources:
  pipelines:
```

```
  - pipeline: MyCIAlias
    source: Farbrikam-CI
    trigger:
      tags:        # This filter is used for triggering the pipeline run
      - Production # Tags are AND'ed
      - Signed
```

Templates

Dienen dazu bereits existierenden Content, Logik und Parameter in YAML pipelines wiederzuverwenden.
Jobs, Steps, Stages, ... können allesamt mittels Templates reused werden.
Variable/dynamische Werte können als Parameter übergeben werden (müssen im Template mit `- name`, `type: dataType` und `default` Value definiert werden).

Include Templates: Content wird insertiert
Extend Templates: definiert Logik der andere Files folgen müssen

```
# File: templates/include-npm-steps.yml

steps:
- script: npm install
- script: yarn install
- script: npm run compile

# File: azure-pipelines.yml

jobs:
- job: Linux
  pool:
    vmImage: 'ubuntu-latest'
  steps:
  - template: templates/include-npm-steps.yml  # Template reference
- job: Windows
  pool:
    vmImage: 'windows-latest'
  steps:
  - template: templates/include-npm-steps.yml  # Template reference
```

Template parameters

```
# File: simple-param.yml
parameters:
- name: yesNo # name of the parameter; required
  type: boolean # data type of the parameter; required
  default: false

steps:
- script: echo ${{ parameters.yesNo }}


# File: azure-pipelines.yml
trigger:
- main

extends:
  template: simple-param.yml
  parameters:
      yesNo: false # set to a non-boolean value to have the build fail
```

Template expressions

```
# job.yml
parameters:
- name: 'jobs'
  type: jobList
```

```
  default: []

jobs:
- ${{ each job in parameters.jobs }}: # Each job
  - ${{ each pair in job }}:          # Insert all properties other than "steps"
      ${{ if ne(pair.key, 'steps') }}:
        ${{ pair.key }}: ${{ pair.value }}
    steps:                            # Wrap the steps
    - task: SetupMyBuildTools@1       # Pre steps
    - ${{ job.steps }}                # Users steps
    - task: PublishMyTelemetry@1      # Post steps
      condition: always()

# azure-pipelines.yml
jobs:
- template: job.yml
  parameters:
    jobs:
    - job: A
      steps:
      - script: echo This will get sandwiched between SetupMyBuildTools and PublishMyTelemetry.
    - job: B
      steps:
      - script: echo So will this!
```

Tasks

> "Building blocks for defining automation in a pipeline"
> Führt eine Action in einer Pipeline aus und ist als script oder procedure verpackt. Kann Inputs entgegennehmen.
> Funktion von Tasks kann entweder mittels `task` oder `script` realisiert werden, nur Syntax ist anders

Step target: kann verwendet werden um execution context (agent host/container) in dem Task ausgeführt wird zu wechseln.