„Azure DevOps Pipelines as Code”, czyli budowanie pipeline w oparciu o YAML

Jakub Podoba
Senior Microsoft Azure DevOps

softserve

WHOAMI

Obecnie:

- Senior Microsoft Azure DevOps @Soft**Serve**
- CEO @hotbee.eu
- Freelancer

https://www.linkedin.com/in/jakubpodoba/

jpodoba@hotbee.eu

@jpodoba1

https://hotbee.eu/blog

https://github.com/jpodoba

Po pracy:

Mąż i ojciec,

Pasjonat lotnictwa,

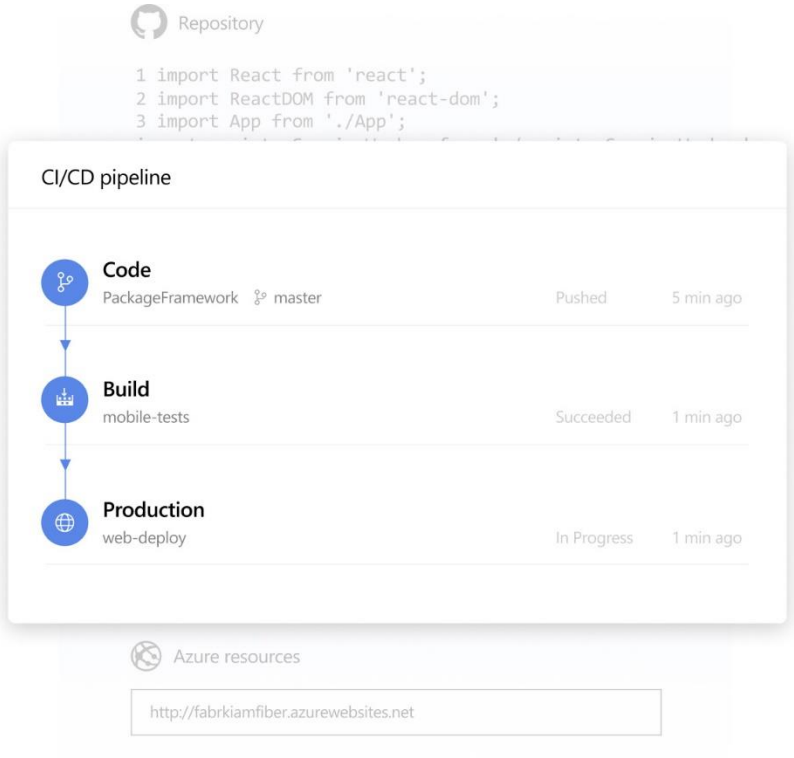
Zapalony Azurowiec, 🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥

Agenda

1. Czym są Azure pipelines
2. Klasyczny edytor vs YAML
3. Zalety YAML
4. Koncepcja Multi-Stage Pipelines
5. YAML structure
6. Dema
7. Q & A


Microsoft Azure
User Group Poland
Wrocław

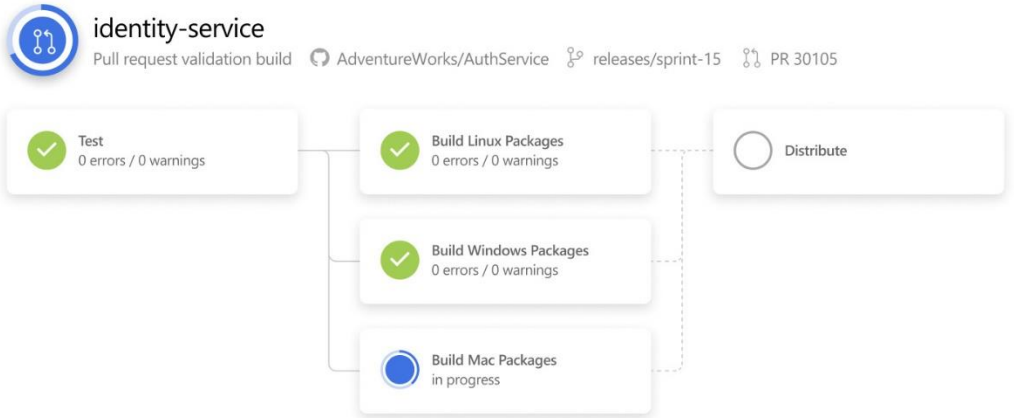
Azure pipelines



Azure Pipelines

Azure pipelines, dlaczego warto:

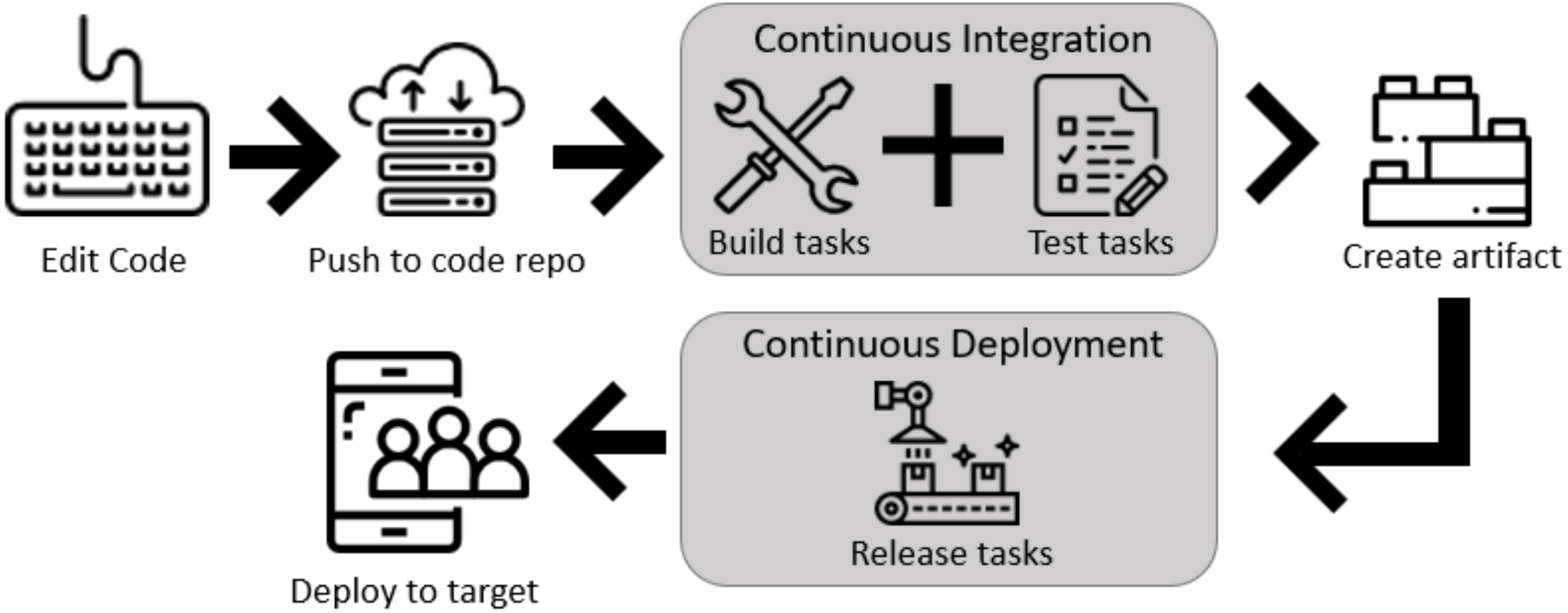
Czy warto używać?



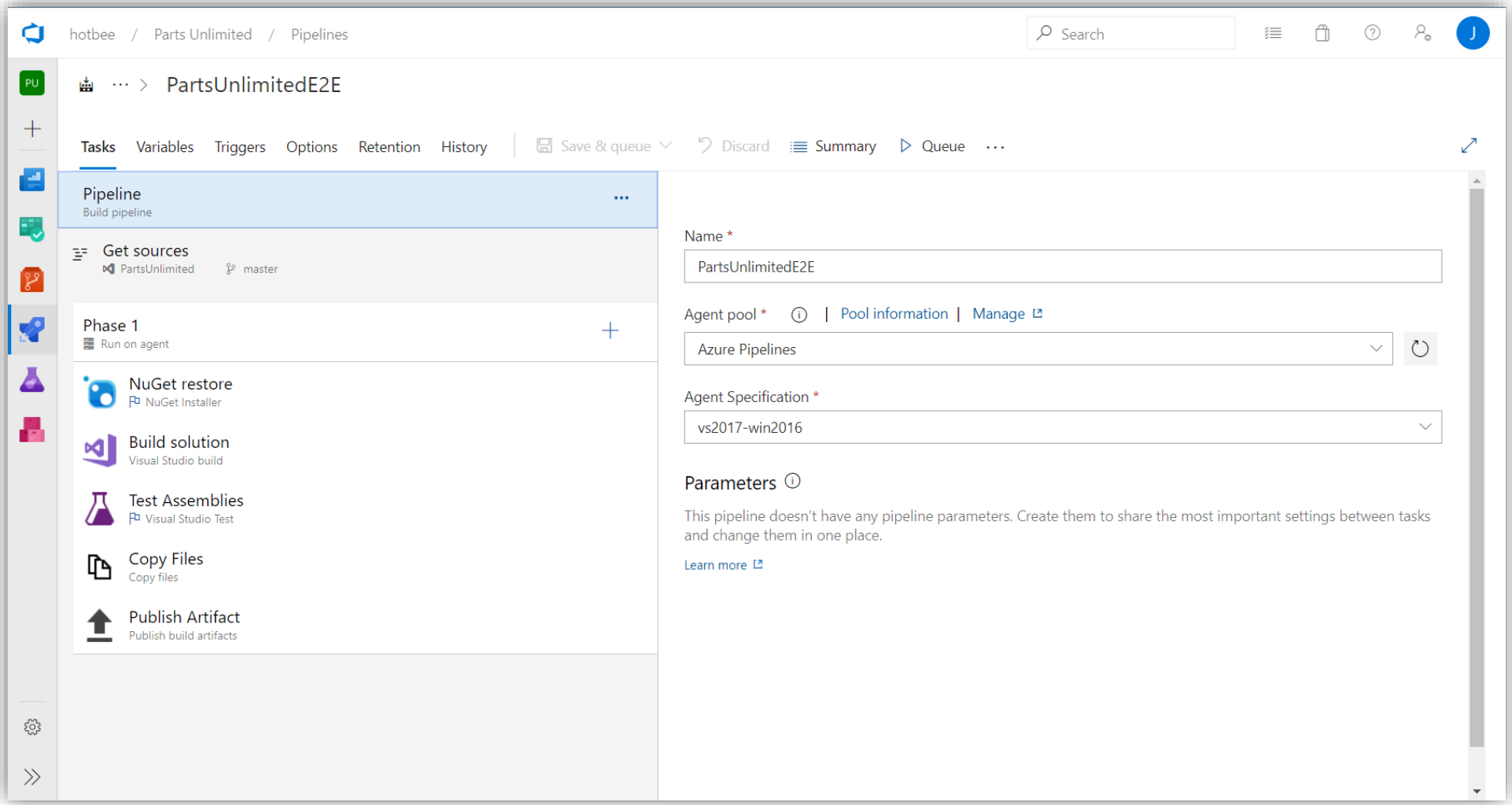
- Darmowe***
- Dowolny język, dowolna platforma,
- Kontenery i platforma Kubernetes,
- Wdrażanie w dowolnej chmurze,
- Możecie robić deploy do różnych systemów jednocześnie,
- Agenci dla systemów Linux, macOS i Windows hostowani przez firmę Microsoft
- Integracja z GitHub

*** - dla publicznych projektów, bądź 1800min (30h) za darmo na miesiac dla projektów prywatnych

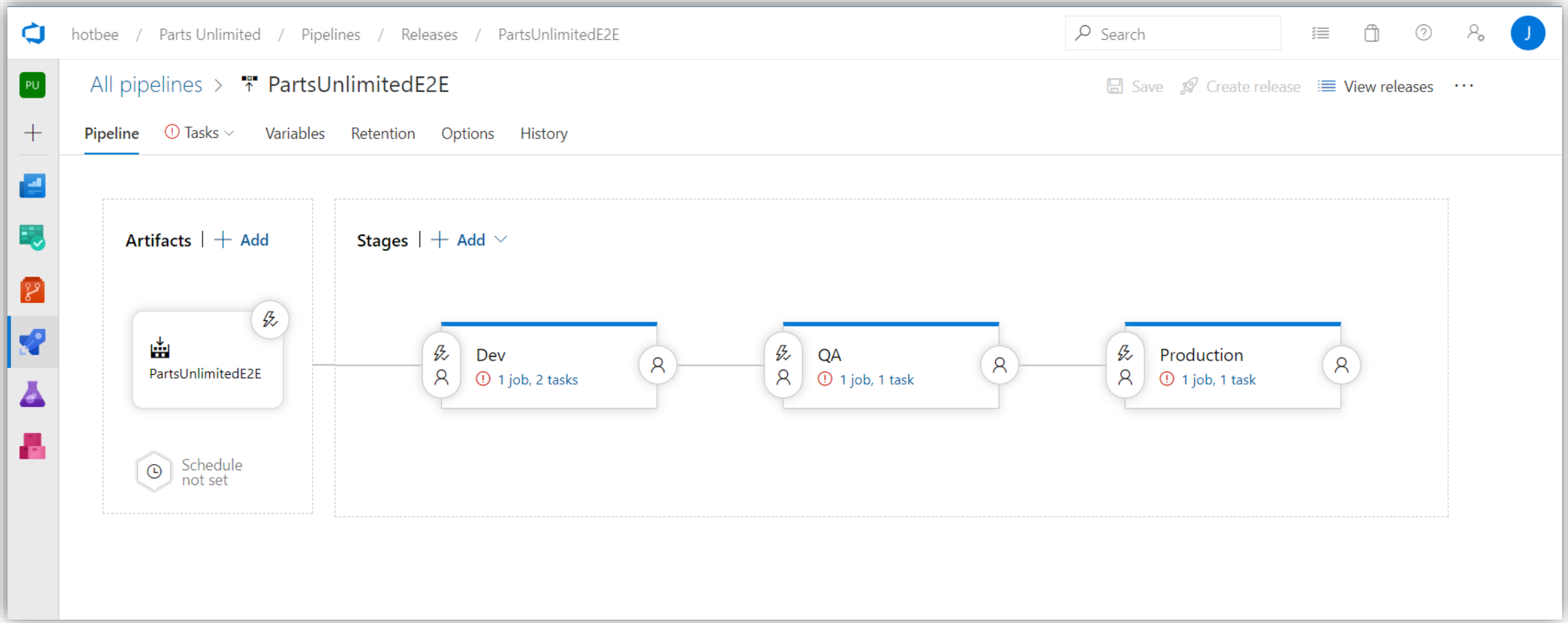
Klasyczny edytor



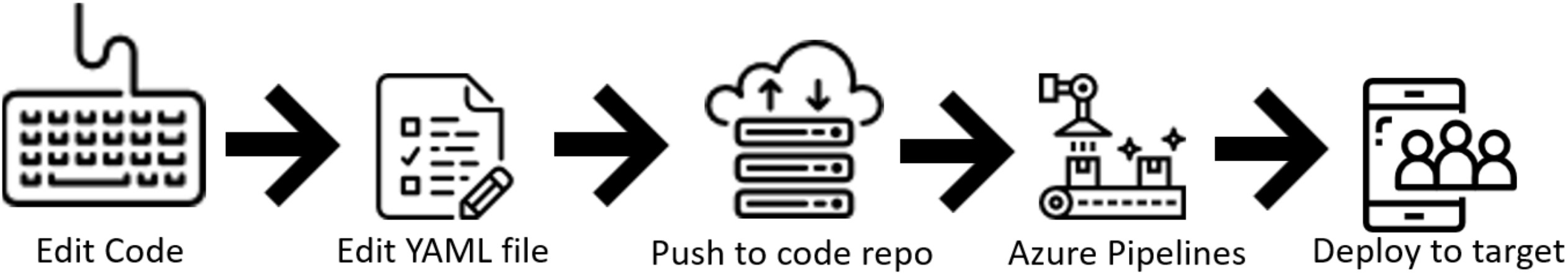
Klasyczny edytor - build



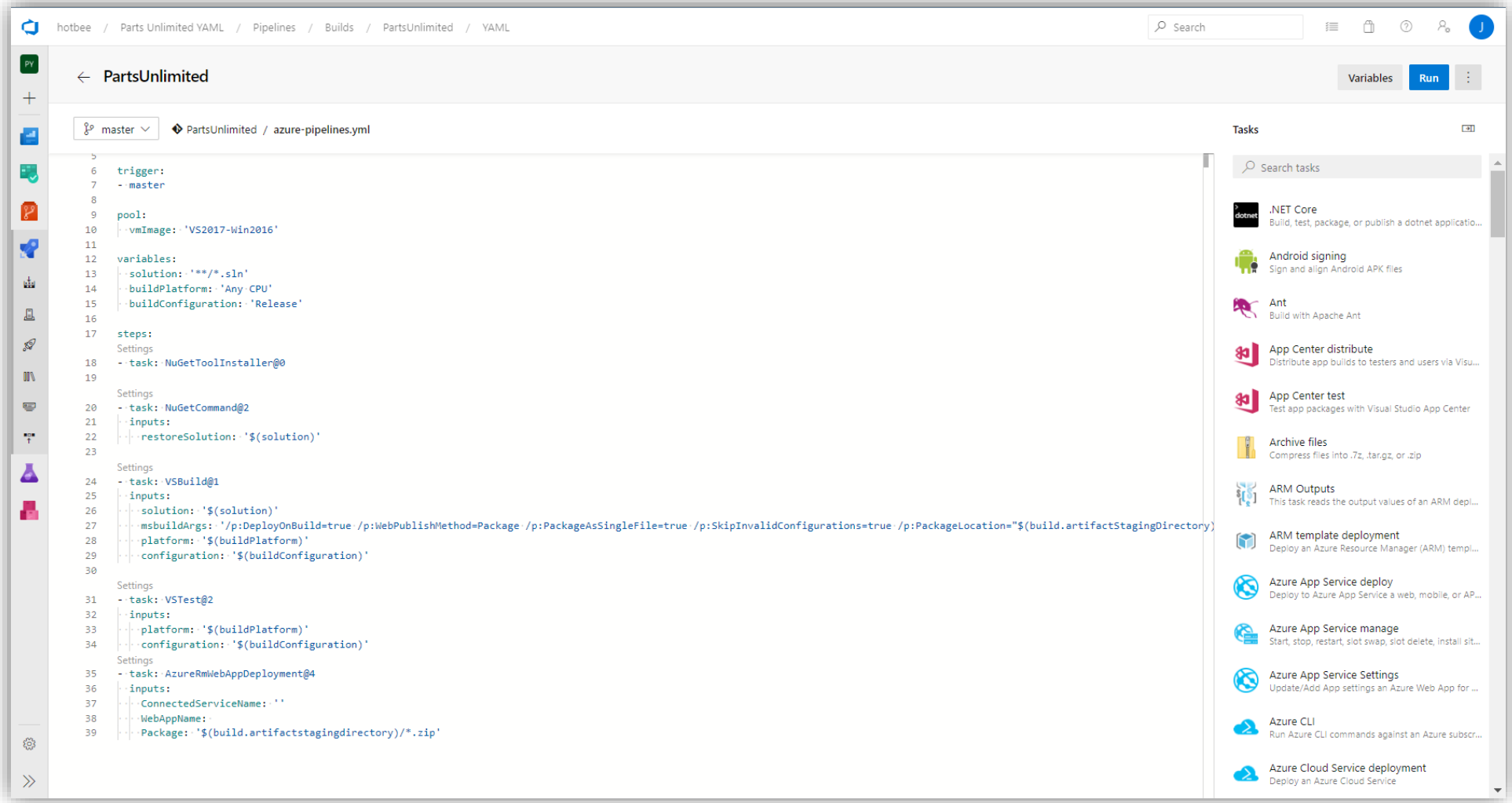
Klasyczny edytor - release



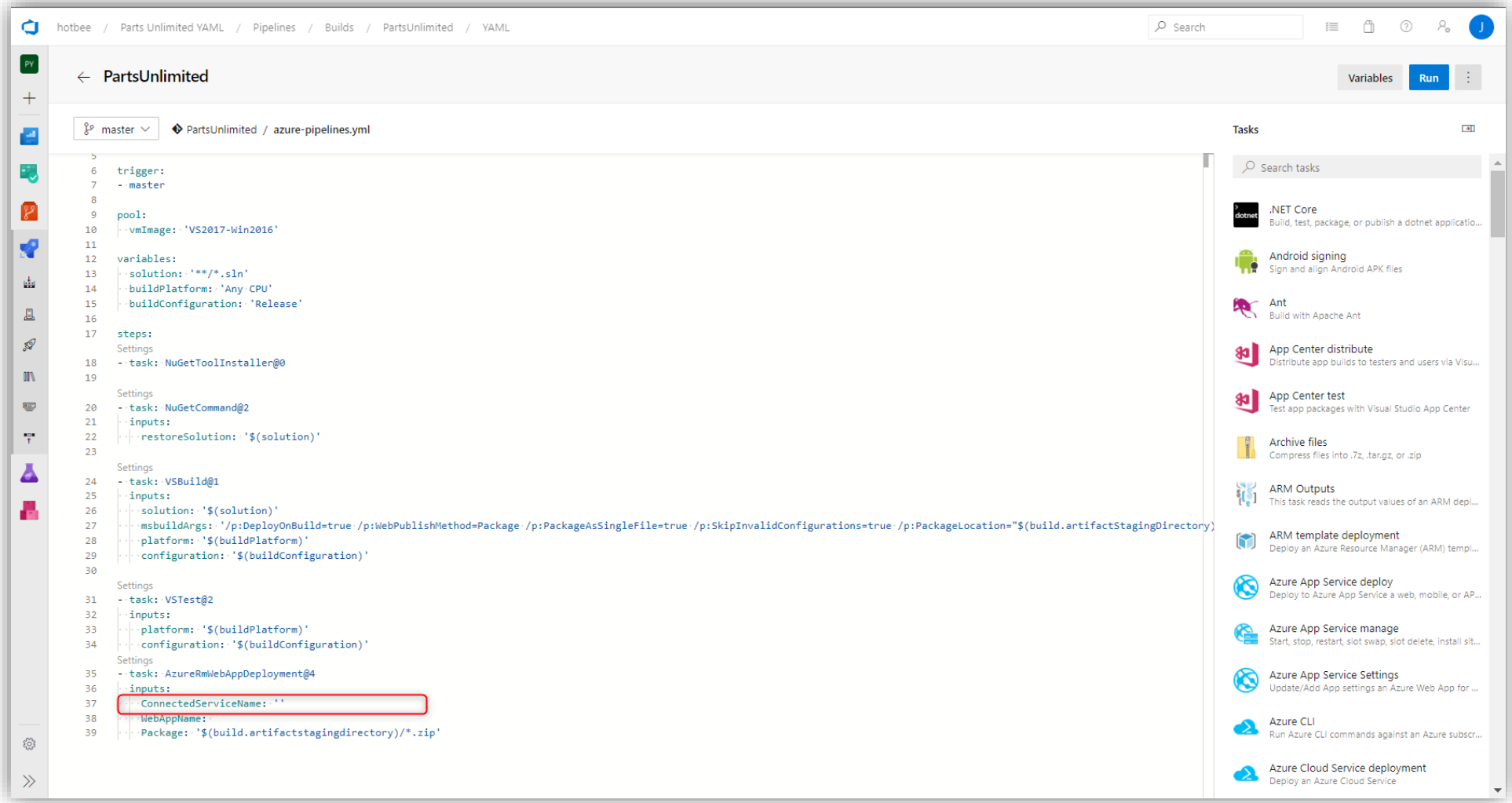
Azure Pipelines jako kod (YAML)



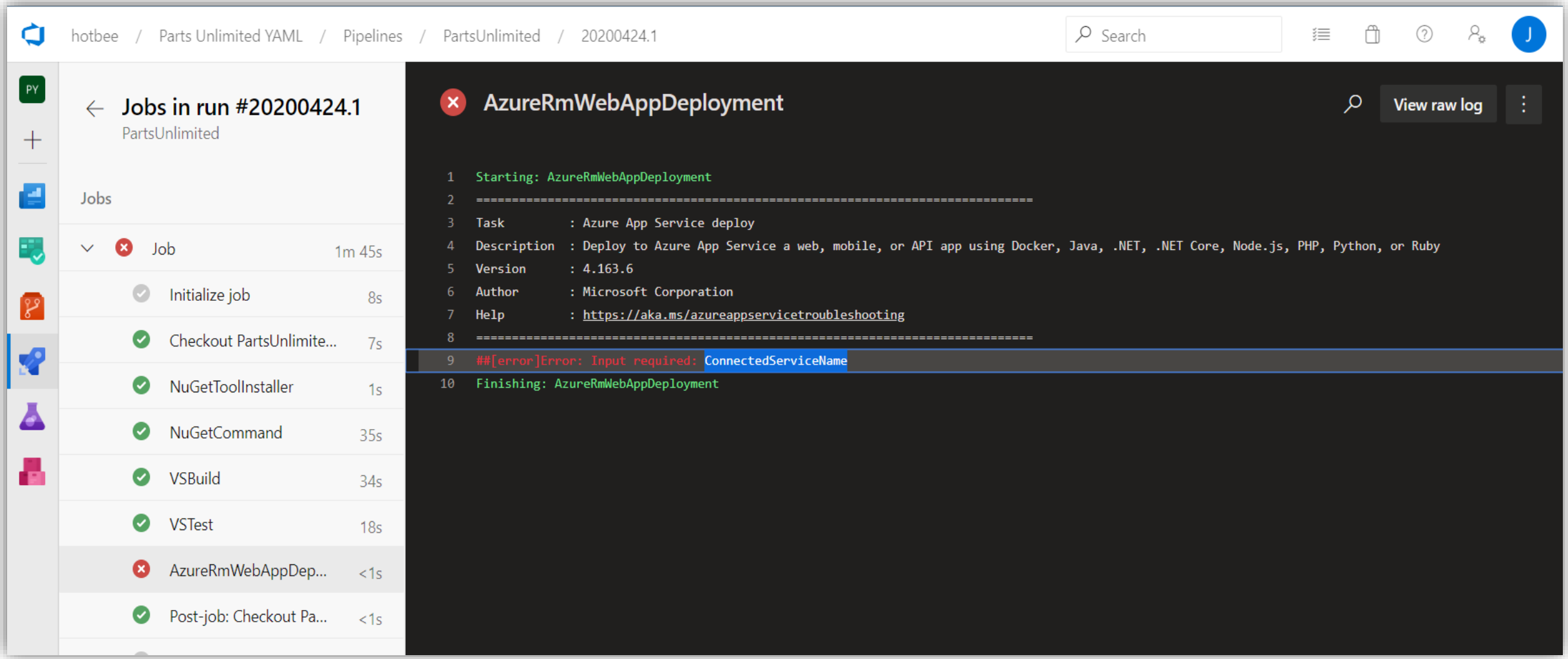
Azure Pipelines jako kod (YAML)



Azure Pipelines jako kod (YAML)



Azure Pipelines jako kod (YAML)



Demo 1 – pipeline github

Dlaczego YAML?

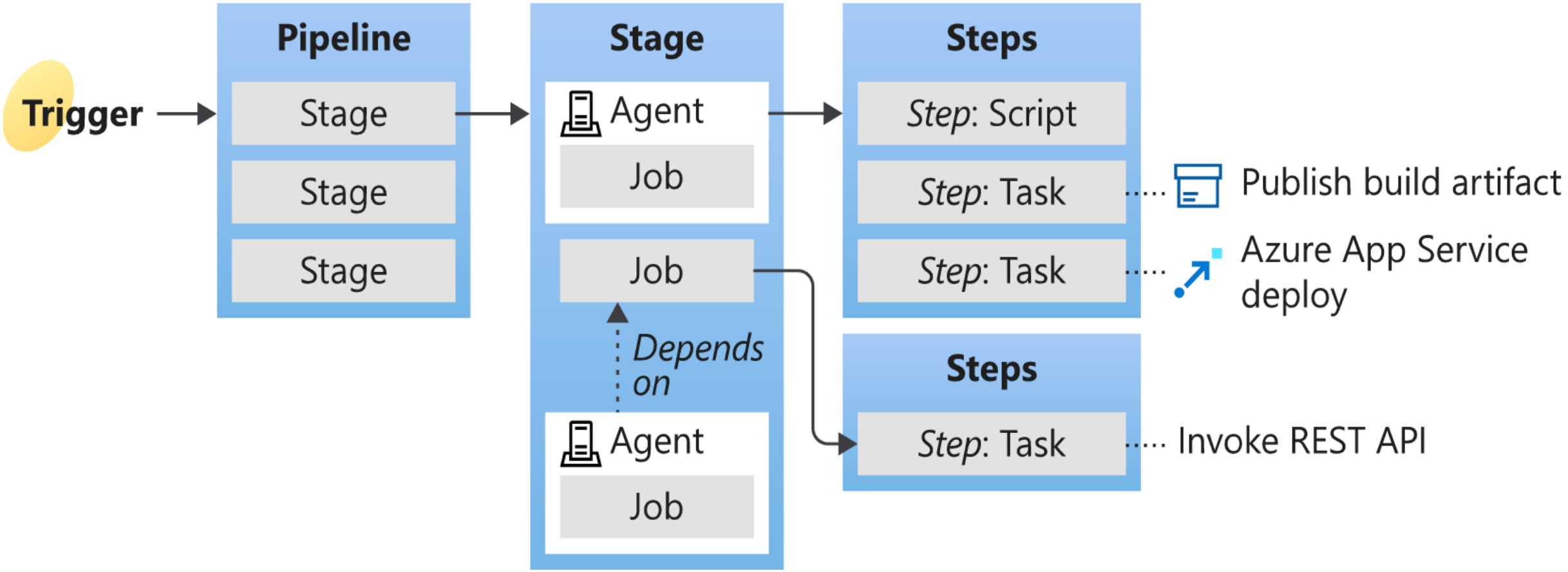
- Znany z poprzednich projektów, puppet, ansible, kubernetes
- Łatwiejszy w utrzymaniu niż edytor graficzny,
- Pipeline jest pisane dekralatywne,
- Wszystko w trzymane w GIT,
- YAML podąża za konkretnym branchem,
- Prosty do kopiowania i utrzymania,
- Dodatek w Visual Studio code,
- Prosty w konwersji z obecnego edytora,

Dostępne feature'y

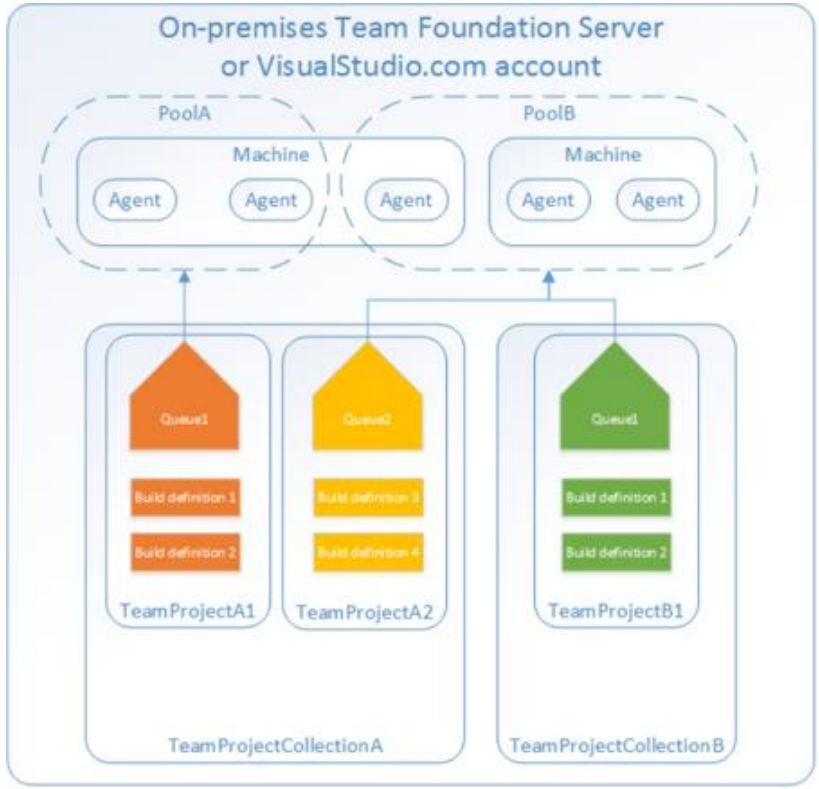
Feature	YAML	Classic Build	Classic Release	Notes
Agents	Yes	Yes	Yes	Specifies a required resource on which the pipeline runs.
Approvals	Yes	No	Yes	Defines a set of validations required prior to completing a deployment stage.
Artifacts	Yes	Yes	Yes	Supports publishing or consuming different package types.
Caching	Yes	Yes	No	Reduces build time by allowing outputs or downloaded dependencies from one run to be reused in later runs. In Preview, available with Azure Pipelines only.
Conditions	Yes	Yes	Yes	Specifies conditions to be met prior to running a job.
Container jobs	Yes	No	No	Specifies jobs to run in a container.
Demands	Yes	Yes	Yes	Ensures pipeline requirements are met before running a pipeline stage. Requires self-hosted agents.
Dependencies	Yes	Yes	Yes	Specifies a requirement that must be met in order to run the next job or stage.
Deployment groups	Yes	No	Yes	Defines a logical set of deployment target machines.

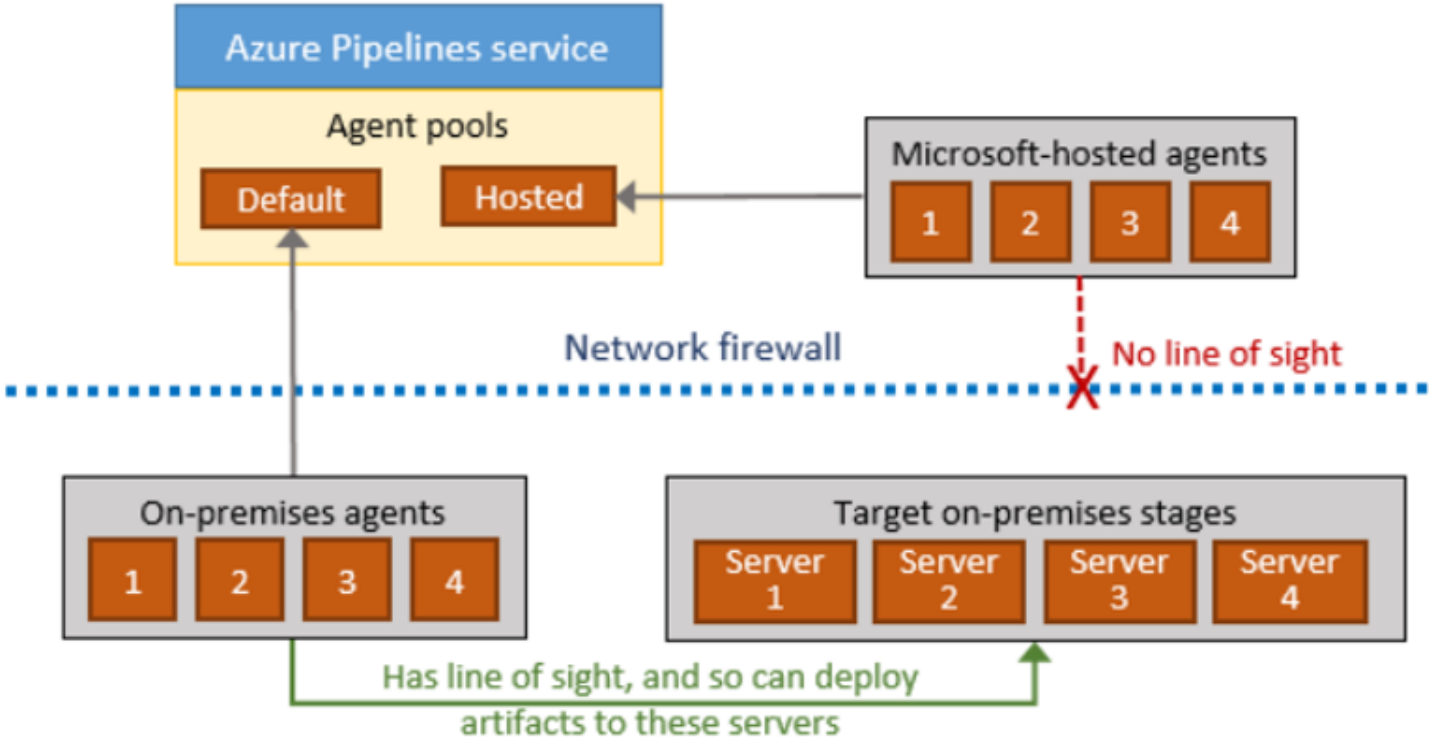
https://docs.microsoft.com/pl-pl/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops

Koncept Azure DevOps pipeline



Agenci





https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser

Agenci – hostowani przez Microsoft

Image	Classic Editor Agent Specification	YAML VM Image Label	Included Software
Windows Server 2019 with Visual Studio 2019	*windows-2019*	`windows-latest` OR `windows-2019`	Link
Windows Server 2016 with Visual Studio 2017	*vs2017-win2016*	`vs2017-win2016`	Link
Ubuntu 18.04	*ubuntu-18.04*	`ubuntu-latest` OR `ubuntu-18.04`	Link
Ubuntu 16.04	*ubuntu-16.04*	`ubuntu-16.04`	Link
macOS X Mojave 10.14	*macOS-10.14*	`macOS-10.14`	Link
macOS X Catalina 10.15	*macOS-10.15*	`macOS-latest` OR `macOS-10.15`	Link

https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops

Struktura YAML

- Pipeline
  - Stage A
    - Job 1
      - Step 1.1
      - Step 1.2
      - ...
    - Job 2
      - Step 2.1
      - Step 2.2
      - ...
  - Stage B
    - ...

YAML

```yaml
name: string  # build numbering format
resources:
  pipelines: [ pipelineResource ]
  containers: [ containerResource ]
  repositories: [ repositoryResource ]
variables: # several syntaxes, see specific section
trigger: trigger
pr: pr
stages: [ stage | templateReference ]
```

YAML

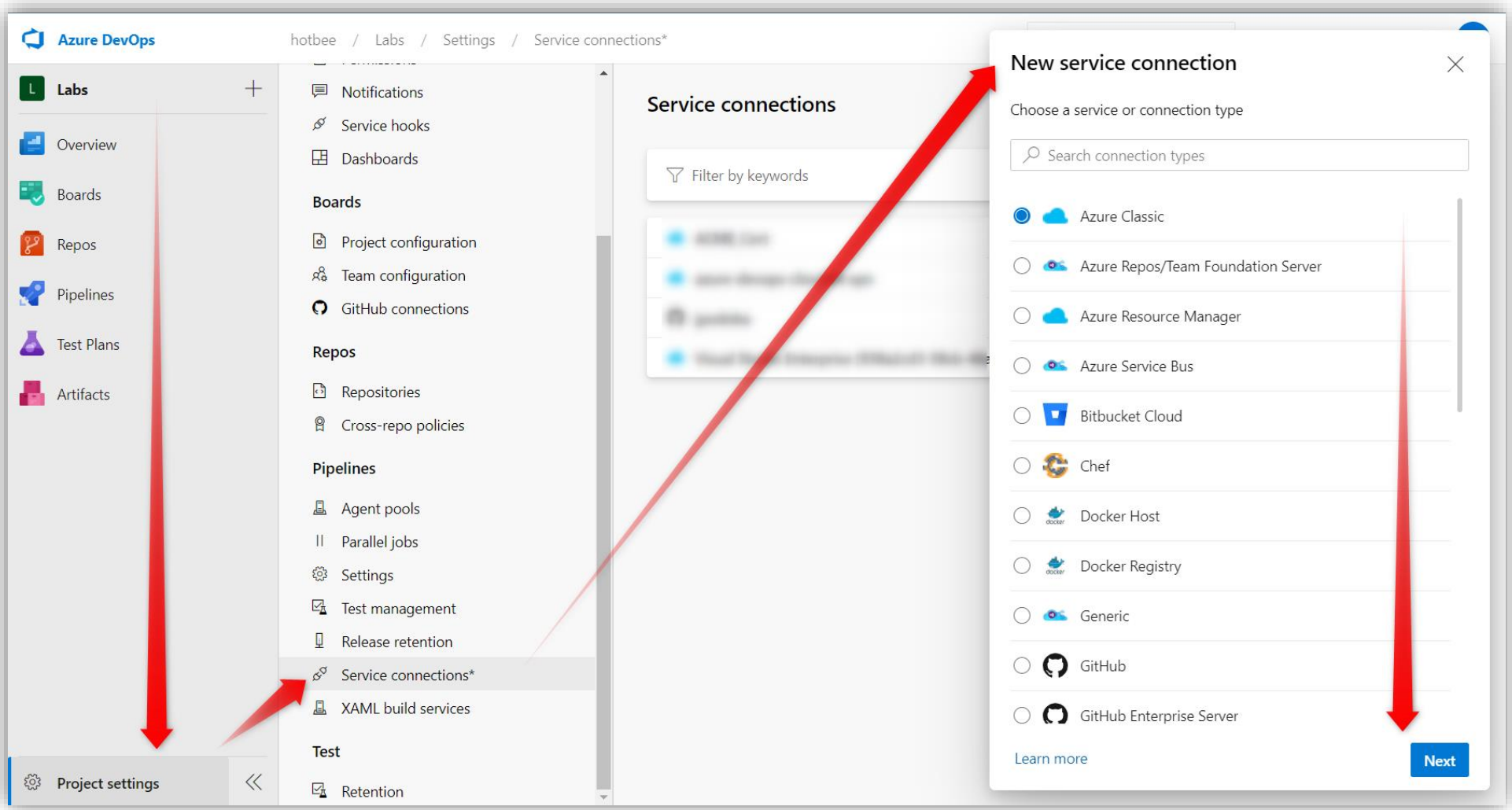
```yaml
# ... other pipeline-level keywords
jobs: [ job | templateReference ]
```

YAML

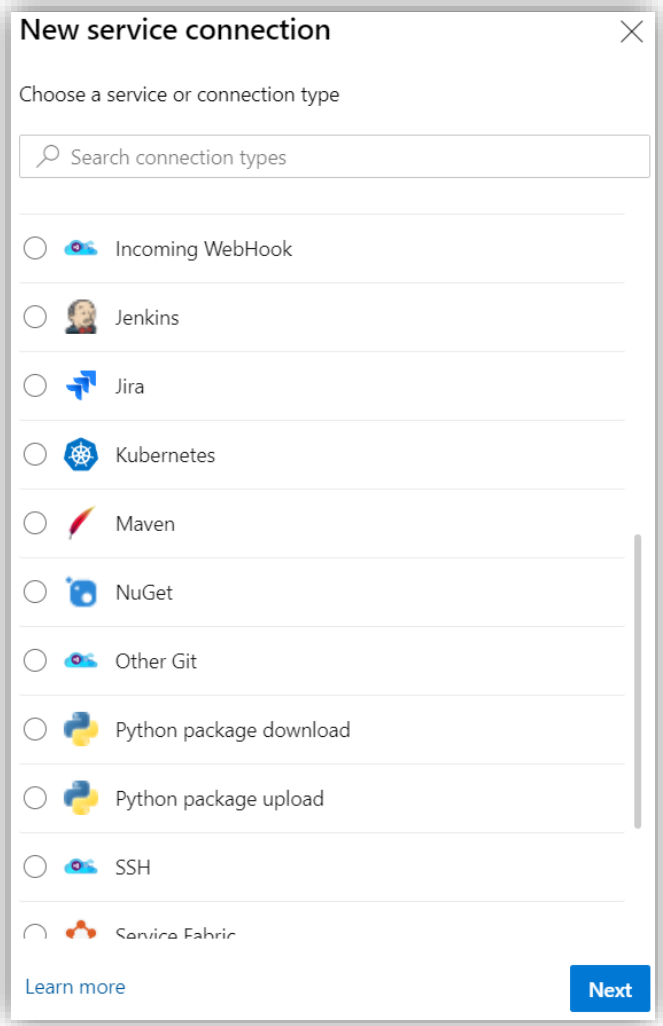
```yaml
# ... other pipeline-level keywords
steps: [ script | bash | pwsh | powershell | checkout | task | templateReference ]
```

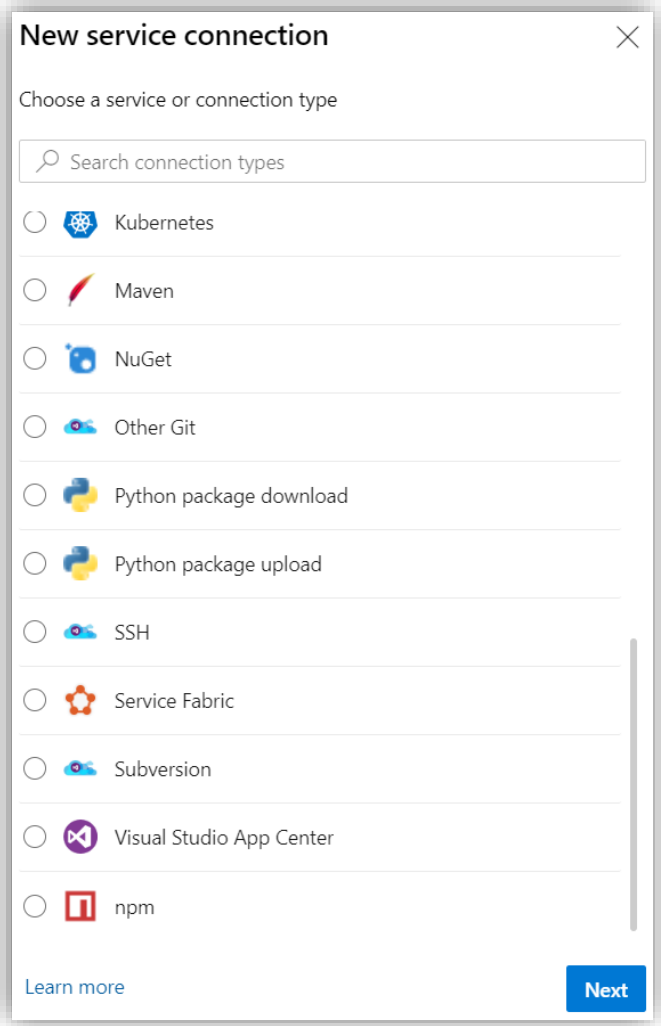
Demo 2 – prosty pipeline

Service connection



Service connection





Nested YAML

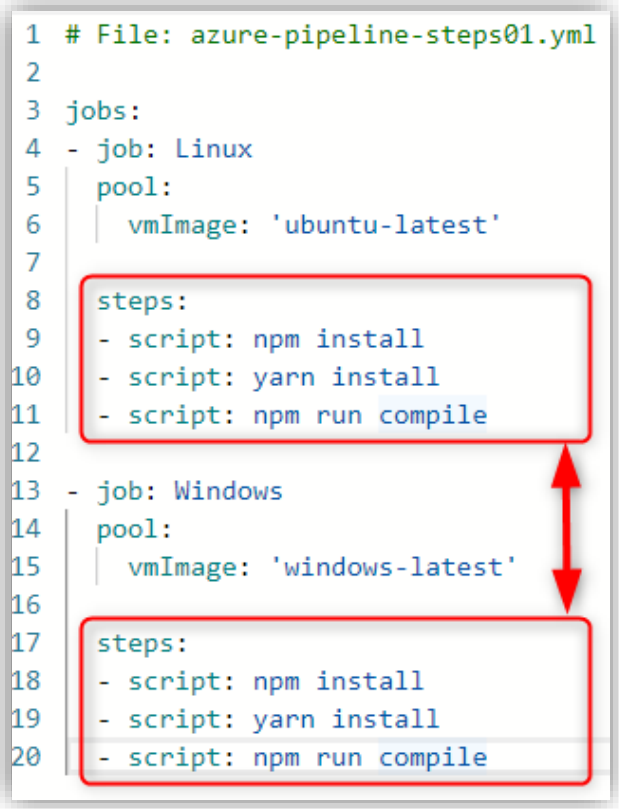
- Steps,
- Jobs,
- Stage,
- Variables,

```yaml
# File: azure-pipeline-steps01.yml

jobs:
- job: Linux
  pool:
    vmImage: 'ubuntu-latest'

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

- job: Windows
  pool:
    vmImage: 'windows-latest'

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

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/templates?view=azure-devops

Nested YAML

- Steps,
- Jobs,
- Stage,
- Variables,

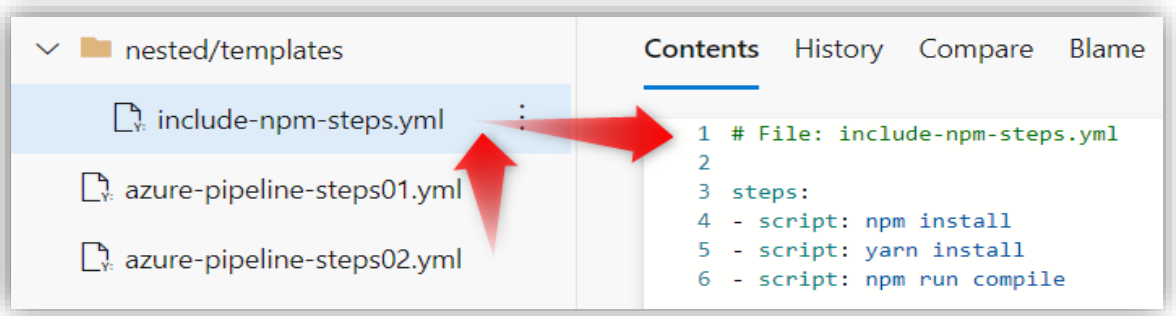


```yaml
# File: azure-pipeline-steps01.yml

jobs:
- job: Linux
  pool:
    vmImage: 'ubuntu-latest'

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

- job: Windows
  pool:
    vmImage: 'windows-latest'

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

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/templates?view=azure-devops

Nested - steps

```yaml
# File: azure-pipeline-steps02.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
```



https://docs.microsoft.com/en-us/azure/devops/pipelines/process/templates?view=azure-devops

Demo 3 – nested pipeline

Triggers

Classic build pipelines and YAML pipelines

Continuous integration (CI) triggers vary based on the type of repository you build in your pipeline.

- CI triggers in Azure Repos Git
- CI triggers in GitHub
- CI triggers in BitBucket Cloud
- CI triggers in TFVC

```yaml
# A pipeline with no CI trigger
trigger: none
```

Pull request validation (PR) triggers also vary based on the type of repository.

- PR triggers in Azure Repos Git
- PR triggers in GitHub
- PR triggers in BitBucket Cloud

```yaml
# specific path build
trigger:
  branches:
    include:
    - master
    - releases/*
  paths:
    include:
    - docs/*
    exclude:
    - docs/README.md
```

```yaml
# specific tag
trigger:
  tags:
    include:
    - v2.*
    exclude:
    - v2.0
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/build/triggers?view=azure-devops

Triggers - scheduled

```yaml
schedules:
- cron: "0 0 * * *"
  displayName: Daily midnight build
  branches:
    include:
    - master
    - releases/*
    exclude:
    - releases/ancient/*
- cron: "0 12 * * 0"
  displayName: Weekly Sunday build
  branches:
    include:
    - releases/*
  always: true
```

```yaml
# YAML file in the release branch
schedules:
- cron: "0 0 * * *"
  displayName: Daily midnight build
  branches:
    include:
    - master

# YAML file in the master branch with release added to the branches list
schedules:
- cron: "0 0 * * *"
  displayName: Daily midnight build
  branches:
    include:
    - master
    - release
```

```
mm HH DD MM DW
 \  \  \  \  \__ Days of week
  \  \  \  \____ Months
   \  \  \______ Days
    \  \________ Hours
     \__________ Minutes
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/scheduled-triggers?view=azure-devops

Triggers - pipeline

```yaml
# this is being defined in app-ci pipeline
resources:
  pipelines:
  - pipeline: securitylib   # Name of the pipeline resource
    source: security-lib-ci # Name of the triggering pipeline
    trigger:
      branches:
      - releases/*
      - master
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/pipeline-triggers?view=azure-devops

Variables

- Użytkownika,
- Systemowe, **(predefiniowane)**
- Środowiskowe, **(w zależności od OS)**

- Trzy poziomy definiowania:
- **Root level,**
- **Stage level,**
- **Job leve,**

```yaml
variables:
  global_variable: value    # this is available to all jobs

jobs:
- job: job1
  pool:
    vmImage: 'ubuntu-16.04'
  variables:
    job_variable1: value1    # this is only available in job1
  steps:
  - bash: echo $(global_variable)
  - bash: echo $(job_variable1)
  - bash: echo $JOB_VARIABLE1 # variables are available in the script environment too

- job: job2
  pool:
    vmImage: 'ubuntu-16.04'
  variables:
    job_variable2: value2    # this is only available in job2
  steps:
  - bash: echo $(global_variable)
  - bash: echo $(job_variable2)
  - bash: echo $GLOBAL_VARIABLE
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/build/variables?view=azure-devops&tabs=yaml

Conditions

- „by design” wszystko wyzwalane jest jednocześnie,
- „by design” nic od siebie nie zależy,
- Są po to aby ustalić zależności,
- I ograniczenia, (niezastąpione przy multi stage)

Run for the master branch, if succeeding

```
and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/master'))
```

Run if the branch is not master, if succeeding

```
and(succeeded(), ne(variables['Build.SourceBranch'], 'refs/heads/master'))
```

Run for user topic branches, if succeeding

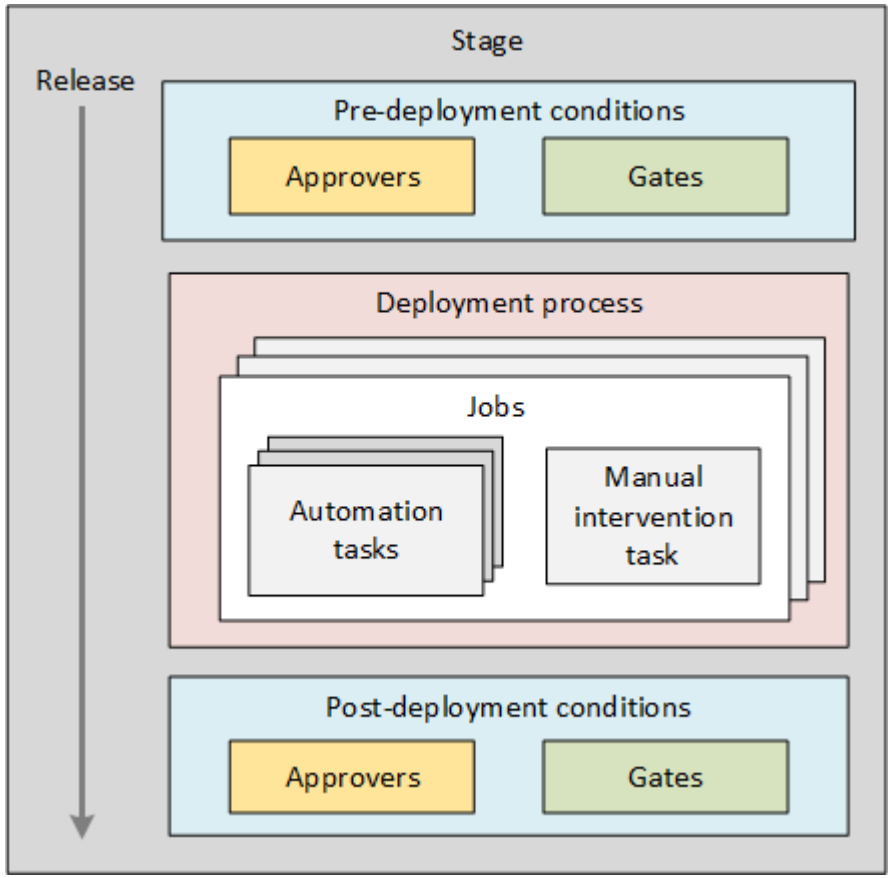
```
and(succeeded(), startsWith(variables['Build.SourceBranch'], 'refs/heads/users/'))
```

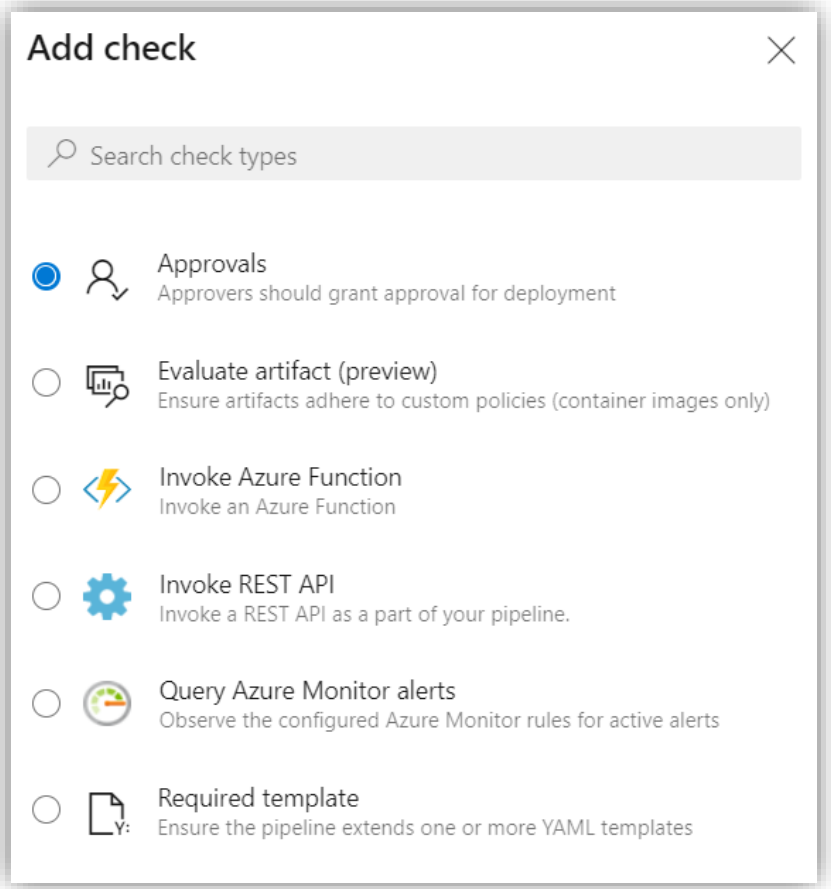
- zagadka

```
  condition: and(succeeded(), ne(variables['Build.Reason'], 'PullRequest'))
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/conditions?view=azure-devops&tabs=yaml

Approvals





https://docs.microsoft.com/en-us/azure/devops/pipelines/release/approvals/?view=azure-devops

Approvals - manual

https://docs.microsoft.com/en-us/azure/devops/pipelines/release/approvals/?view=azure-devops

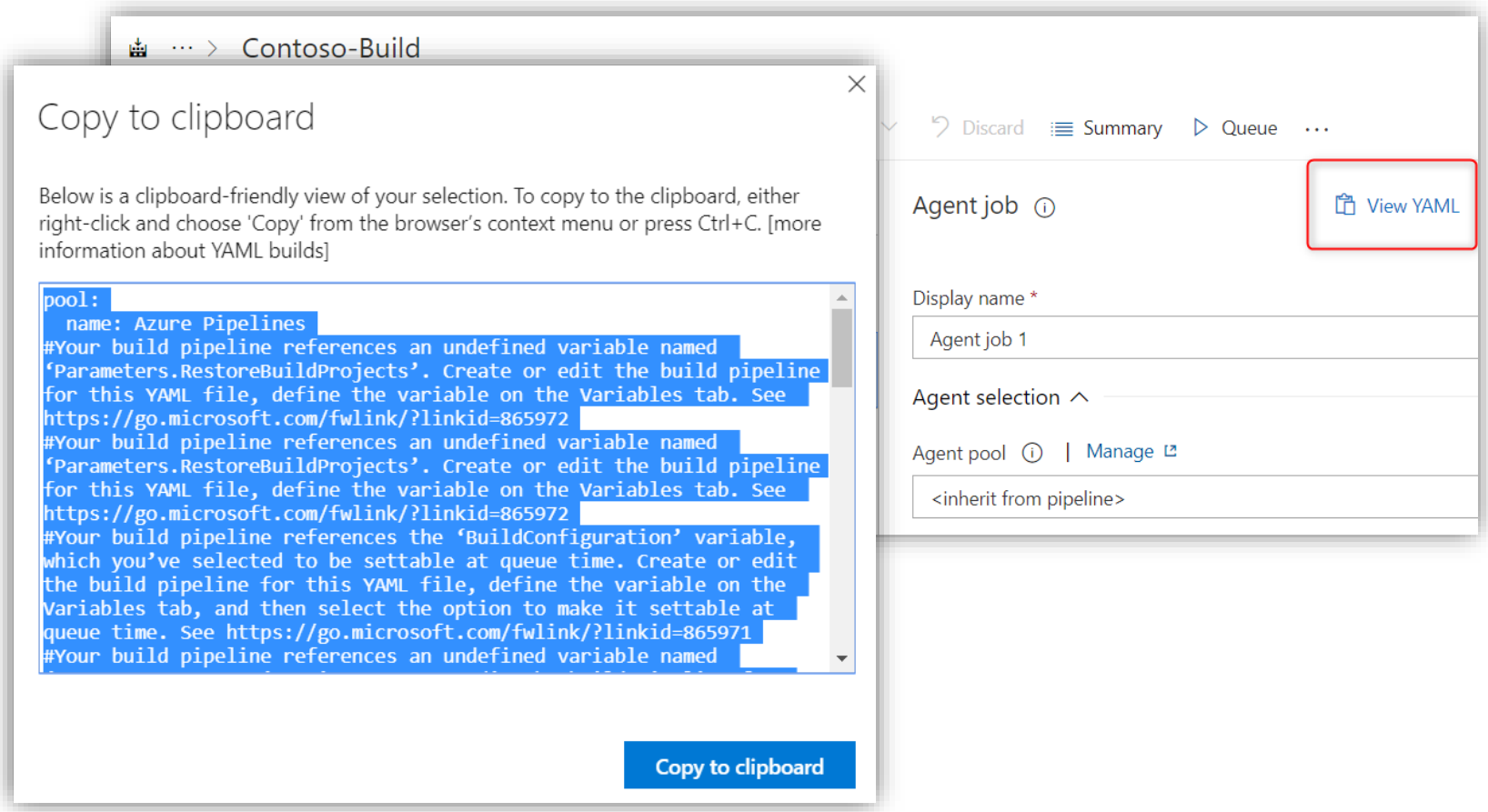
Approvals - manual

```yaml
- stage: PROD
  displayName: 'PROD(CD)'
  condition: and(succeeded('STG'), eq(variables['Build.SourceBranch'], 'refs/heads/master'))
  dependsOn:
   - BLD
   - STG
  variables:
    stage: 'prod'
  jobs:
  - deployment: Primary_NorthErope
    pool:
      vmImage: $(vmImage)
    environment: prod
    strategy:
      runOnce:
        deploy:
          steps:
          - template: 'pipelines/infrastructure/deploy.yml'
            parameters: {type: 'primary', spn: 'azure-devops-cloud4it-spn', location: 'northeurope'}
          - template: 'pipelines/application/deploy.yml'
            parameters: {type: 'primary', spn: 'azure-devops-cloud4it-spn'}
```

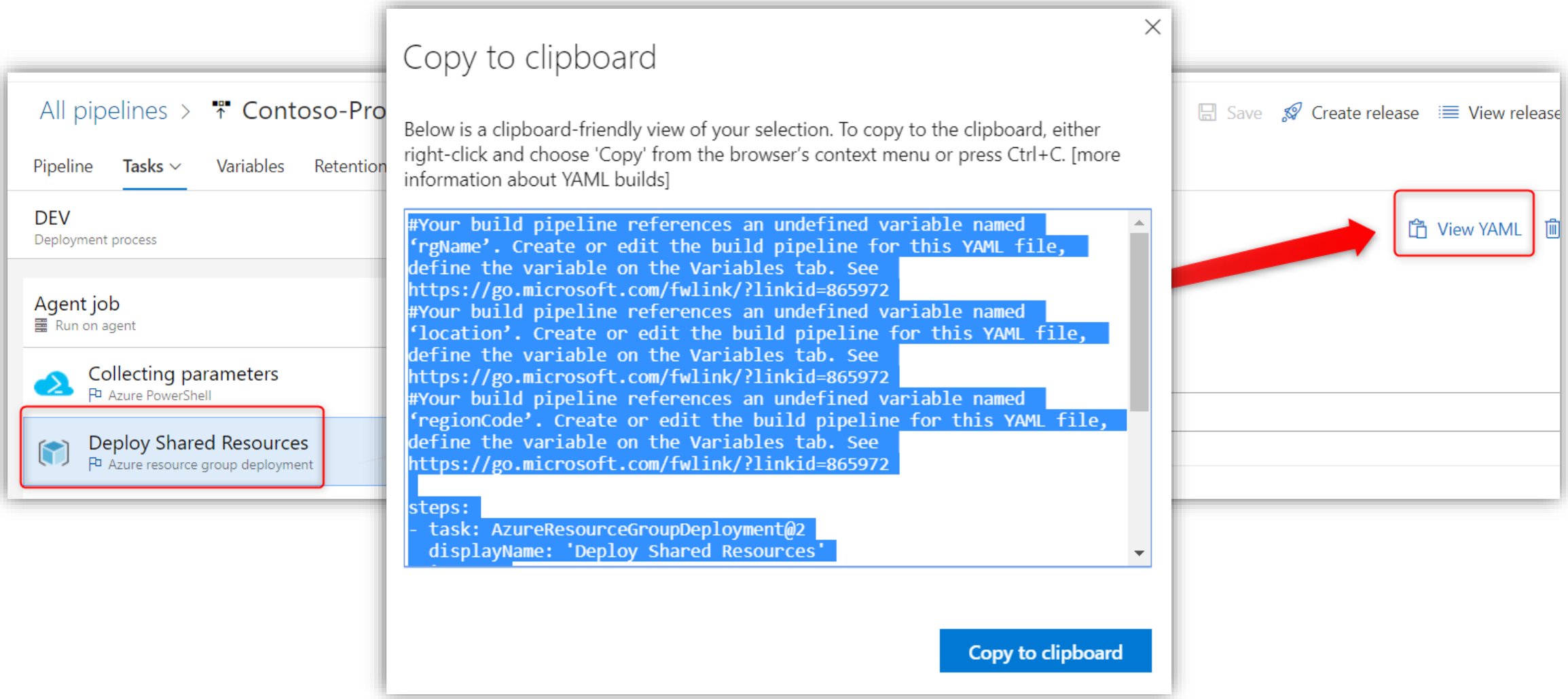
https://docs.microsoft.com/en-us/azure/devops/pipelines/release/approvals/?view=azure-devops

Migracja do YAML

- Prosta dla „build”,
- Prosta dla „release”
  - Ale nie aż tak

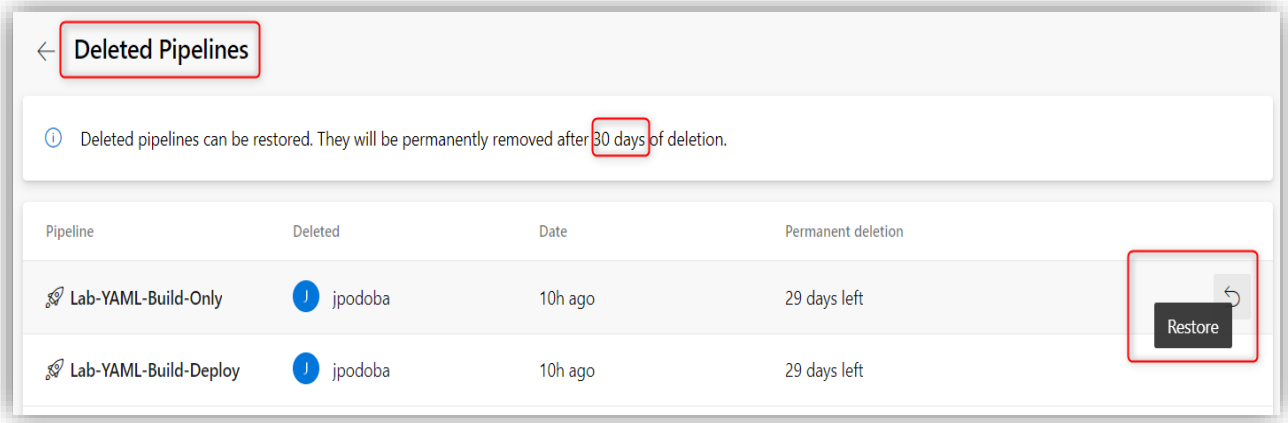


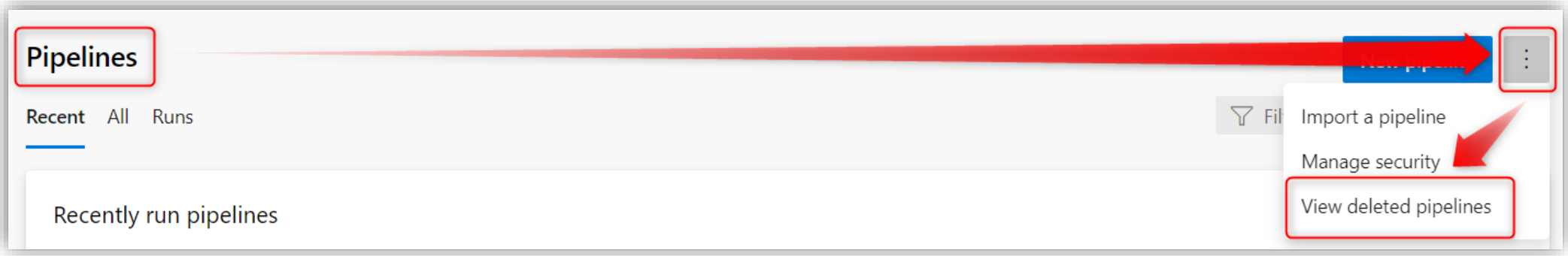
Migracja do YAML



Historia i oddzyskiwanie

- „pipeline as code” – jest w GIT,
- proste w śledzeniu zmian,
- proste w cofaniu zmian,
- proste w oddzyskiwaniu po usunięciu,
- Ale co z **pipeline id** ???





Multi stage pipeline

```
Build
| * Infrastructure
| * Application
```

1

```
Build
| * Infrastructure
| * Application
|
|-> develop
|       |
|       |-> Deploy DEV (Primary Region)
|       |     * Primary Region
|       |
|       |-> Deploy QA (Primary Region)
|       |     * Primary Region
```

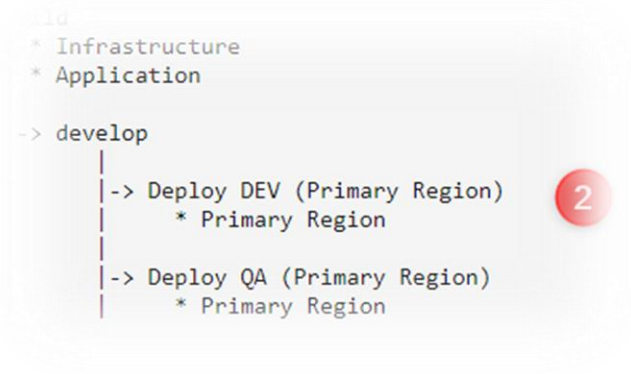
2

```
Build
| * Infrastructure
| * Application
|
|-> develop
|       |
|       |-> Deploy DEV (Primary Region)
|       |     * Primary Region
|       |
|       |-> Deploy QA (Primary Region)
|       |     * Primary Region
|
|- master
|       |
|       |-> Deploy UAT
|       |     * Primary Region
|       |
|       |-> Deploy STG
|       |     * Primary Region
|       |
|       |-> Deploy PROD
|       |     * Primary Region
|       |     * Secondary Region
```

3

Multi stage pipeline

```
Build
| * Infrastructure
| * Application
|
|-> develop
|       |
|       |-> Deploy DEV (Primary Region)
|       |      * Primary Region
|       |
|       |-> Deploy QA (Primary Region)
|       |      * Primary Region
|
|- master
|       |
|       |-> Deploy UAT
|       |      * Primary Region
|       |
|       |-> Deploy STG
|       |      * Primary Region
|       |
|       |-> Deploy PROD
|       |      * Primary Region
|       |      * Secondary Region
```



Demo 4 – Multi stage pipeline

Pytania ???

Przydate linki:

- https://docs.microsoft.com/en-us/azure/devops/pipelines/?view=azure-devops
- https://github.com/microsoft/azure-pipelines-yaml
- https://azuredevopslabs.com/labs/azuredevops/yaml/

- https://github.com/jpodoba/Presentations/tree/master/2020-04-27-Cloud4it-Group

Ankieta prelegenta:

https://tinyurl.com/yamlwroclaw

Dziękuje za uwagę

Media copyright of https://christianbuehlmann.com/CMS/CMS2/site-map/item/2694-thank-you-for-your-attention