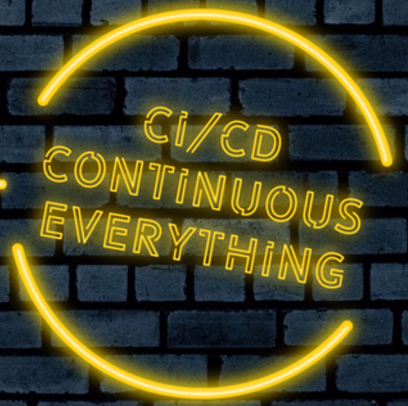
ALL DAY DEVOPS

caffeinated by sonatype

NOVEMBER 10, 2022

Azure DevOps:- Templating Your Pipelines and Keeping Them Dry



Thomas Thornton, Kainos

About Me – Thomas Thornton

- Azure Technical Specialist @ Kainos
- Microsoft MVP
- Azure Certified
- Azure Blogger – https://thomasthornton.cloud
- Twitter - https://twitter.com/tamstar1234
- LinkedIn - https://www.linkedin.com/in/thomas-thornton-21a86b75/

Initial Questions

Copy & pasting the same stage between different pipelines?

Creating the same task several times?

Copy and pasting the same job or stage?

Noticing reptition over time?

What will be cover?

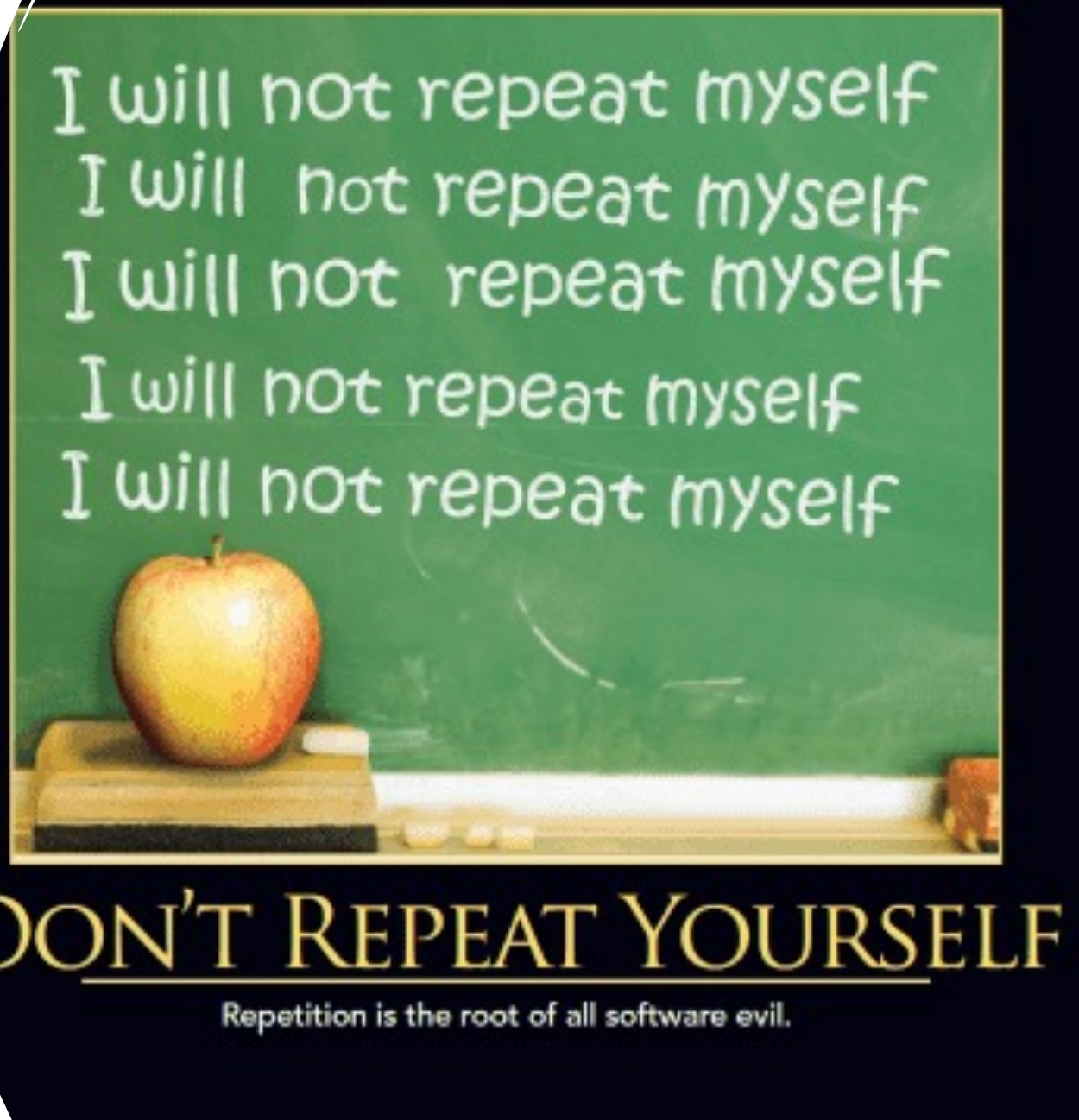
- DRY – what does it mean?
- Why DRY?
- Keeping Azure DevOps Pipelines DRY
- Azure DevOps – Templating
- Example Scenario
- Questions

DRY – What does it mean?

Don’t Repeat Yourself (DRY)

"**Don't repeat yourself**" (**DRY**) is a principle of software development aimed at reducing repetition of software patterns,[1] replacing it with abstractions or using data normalization to avoid redundancy
© Steven Foote

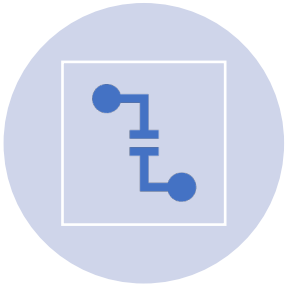
https://en.wikipedia.org/wiki/Don%27t_repeat_yourself#cite_note-1



To me, what does DRY mean?

- Avoid duplication if possible
- Don’t copy & paste
- Structure, structure and more structure!
- Create logic within your pipelines
- Think reusability

Questions to consider when creating an Azure DevOps Pipeline



Repetition

How many times has the configured been repeated?

Less code

Can I write this pipeline with less code?

Duplication

Is duplication needed?

Reusability

Should I refactor?

How can I keep my Azure DevOps Pipeline Dry?

Parameters & Variables

Utilise the use of parameters and variables

Avoid duplication

Don’t copy & paste

Expressions

Create expressions and conditions within your pipeline

Templating

More on that soon ☺

Recommended Blog Post:
https://thomasthornton.cloud/2021/01/20/azure-devops-pipelines-keeping-your-pipelines-dry/

Azure DevOps Pipeline Templates

Azure DevOps Pipeline Templates - What are they?

Templates let you split your pipelines into several files (templates)

Templates let you split your pipeline into several files (Templates)

You be aware of the Separation of Concerns principle

Azure DevOps – Creating Pipelines

Multi-Stage	• Always a situation where we need to reuse the same task, stage etc in several pipelines
DRY	• We want to avoid copy and pasting, adhering to DRY
YAML	• Can bloat over time, what can we do?
Template Pipelines	• This is where pipeline templating becomes super useful

Azure DevOps Pipeline Templates

Pipeline templates allow us to define usable code blocks within your YAML

Making maintenance a lot easier and less prone to errors!

We can define steps, jobs, stages etc

Azure DevOps Pipeline Structure

Now to spend sometime on pipeline structure, we can define the below in our templates

- Parameters
- Variables
- Steps
- Jobs
- Stages
- Conditions

Pipeline Structure – Parameters

```yaml
parameters:
- name: appName
  default: alldaydevops2022
  values:
  - alldaydevops2022
  - alldaydevops20222
  - alldaydevops20223
```

- Parameters help to pass values from main pipeline into our templates
- Parameters must be defined with a name and a type, a default value is optional

Pipeline Structure – Variables

```yaml
variables:
  - group: ${{ parameters.appName }}-app
  - name: RESOURCE_GROUP
    value: "${{ parameters.appName }}-rg"
  - name: LOCATION
    value: westeurope
  - name: LOG_ANALYTICS_WORKSPACE
    value: "${{ parameters.appName }}containersla"
  - name: ACR
    value: "${{ parameters.appName }}containers"
  - name: azureSubscription
    value: "thomasthorntoncloud"
  - name: REPOSITORY
    value: "sampleapp"
```

- Variables in a pipeline template are used similarly as we would as a Variable Group
- Variables can be used throughout your pipeline and templating
- Parameters can be referenced in variables

Pipeline Structure – Stages, Jobs & Templates

```yaml
- stage: Build
  dependsOn: [create_base_resources]
  displayName: Build sample app
  jobs:
  - job: "Build"
    steps:
      - template: pipeline-templates/build-app.yaml
        parameters:
          REPOSITORY: ${{ variables.REPOSITORY }}
          DOCKERFILE: ${{ variables.DOCKERFILE }}
          CONTAINERREGISTRY: ${{ variables.CONTAINERREGISTRY }}
          TAGS: ${{ variables.TAGS }}
```

- This is where the fun begins!
- Creating a stage, we can begin to reference templates pre-made!
- Notice the job reference “build”? It is deploying a templated step

Pipeline Structure – Steps

```yaml
steps:
- task: Docker@2
  displayName: Build and push sample app to container registry
  inputs:
    command: buildAndPush
    repository: ${{ parameters.REPOSITORY }}
    dockerfile: ${{ parameters.DOCKERFILE }}
    containerRegistry: ${{ parameters.CONTAINERREGISTRY }}
    tags: ${{ parameters.TAGS }}
```

- When creating a template to reuse, we usually want to create one or more steps in a job of a pipeline.
- Simply define the steps in the template, as you would in a YAML pipeline.

Pipeline Structure – Steps Template

```yaml
steps:
- task: Docker@2
  displayName: Build and push sample app to container registry
  inputs:
    command: buildAndPush
    repository: ${{ parameters.REPOSITORY }}
    dockerfile: ${{ parameters.DOCKERFILE }}
    containerRegistry: ${{ parameters.CONTAINERREGISTRY }}
    tags: ${{ parameters.TAGS }}
```

- Following on, looking at the steps task within a template
- Notice the reference of parameters?
- Steps task Docker@2 uses these parameters
- No copy/pasting

Pipeline Structure – Conditions

```yaml
- stage: deploy_container_app
  dependsOn: [create_container_environment]
  condition: succeeded('create_container_environment')
  jobs:
    - job: "deploy_app"
      steps:
        - template: pipeline-templates/create-containerapp.yaml
          parameters:
```

- Conditions are written as expressions in YAML pipelines
- You can specify the conditions under each stage, job or step runs

Example Scenario

Some final thoughts

- Less code, means less code to maintain
- Try to avoid duplication
- DRY!
- Make your pipelines reusable
- Make it easy to re-use
- Enjoy the journey ☺