Azure pipeline documentation

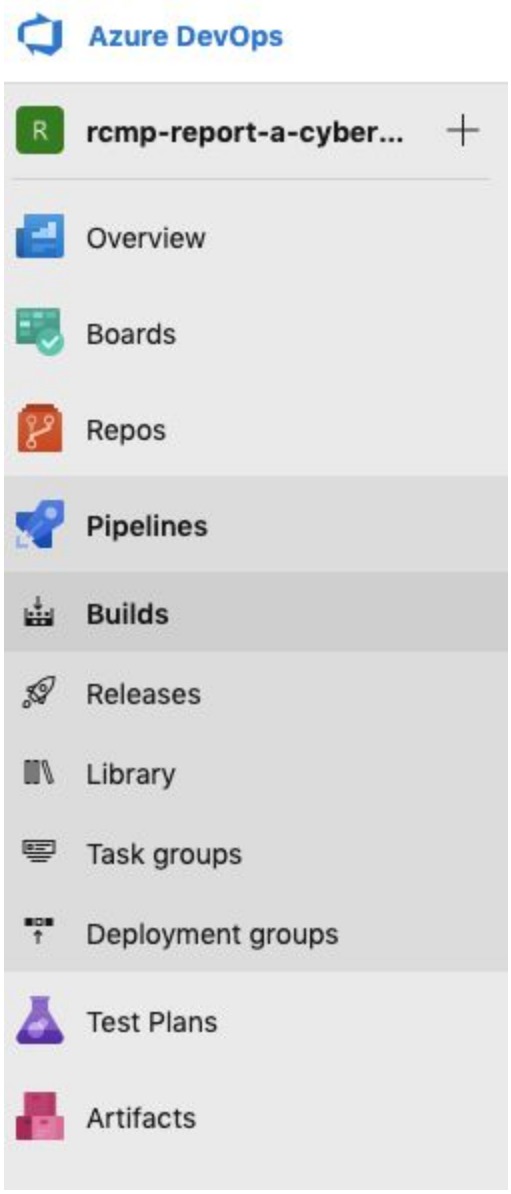
This document will attempt to outline work and knowledge the team has discovered while creating a CI pipeline within Azure for the RCMP-CDS ‘Report A Cybercrime’ product.

The first thing you need to do is to login into Azure DevOps. You need an RCMP Azure account to do this. Use the same credentials as you would to login to the Azure tenant.

RCMP Azure DevOps: https://dev.azure.com/rcmp-grc

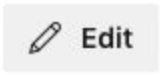
To get started, create a project. In our case, we’ve created `rcmp-report-a-cybercrime`

Within this project, there is a Pipelines section, which can be accessed along the menu of items on the left.



To create a pipeline, you will first need to authenticate with, and then connect your repo to the newly created pipeline.

To edit the pipeline, click the edit button in the top right hand corner.



Below is an example of a pipeline `yaml` file.

```yaml
#frontend pipeline

trigger:
  branches:
    include:
      - master
  paths:
    include:
      - frontend
    exclude:
      - api
      - frontend/manifests

pool:
  vmImage: 'ubuntu-16.04'

steps:
  Settings
  - task: Npm@1
    inputs:
      command: 'install'
      workingDir: 'frontend'

  - script: |
      cd frontend
      npm run compile
      npm run coverage
      npm run lint
      npm run check-translations
    displayName: 'Run npm scripts'
```

Settings

```yaml
- task: Docker@2
  inputs:
    containerRegistry: 'scACR'
    repository: 'frontend'
    command: 'build'
    Dockerfile: 'frontend/Dockerfile'
    tags: |
      $(Build.SourceBranchName)
      latest
    arguments: --build-arg RAZZLE_GOOGLE_ANALYTICS_ID=$(_RAZZLE_GOOGLE_ANALYTICS_ID)
```

Settings

```yaml
- task: Docker@2
  inputs:
    containerRegistry: 'scACR'
    repository: 'frontend'
    command: 'push'
    tags: |
      $(Build.SourceBranchName)
      latest
```

What does this file do?

The first thing is to define the `trigger`. The `trigger` indicates when the CI process will begin. In this case, we are triggering the pipeline on a commit to master (when a pull request is opened in our github repo). We have also indicated when not to trigger the pipeline (`exclude` statements).

`Pool` indicates the OS of the virtual machine that is running the pipeline.

`Steps` indicate the order of `tasks` and `scripts` we want to execute.

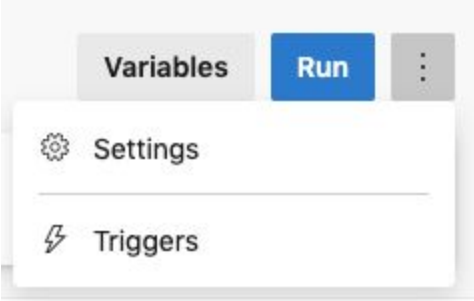
In our case, we have two pipelines - one for our `frontend` and one for our `api`. This document will use the `frontend` as an example. We are first running an npm install command to ensure all the dependencies that required by subsequent `tasks` and `scripts` will be installed on the VM. Next we ensure we are in the `frontend` folder and we run a series of npm commands that compile the code, check the translations, run tests and do linting on the code base.

After this we run `docker build` to containerize our `frontend` code base. We indicate the location of the dockerfile to do this and tag the container to fit our needs.

Finally, we run `docker push` to push the container to our `azure container registry`. It's important to note here that we need a `service connection` for the pipeline to authenticate with the container registry. For how to create a service connection, please see: https://docs.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml

Triggers

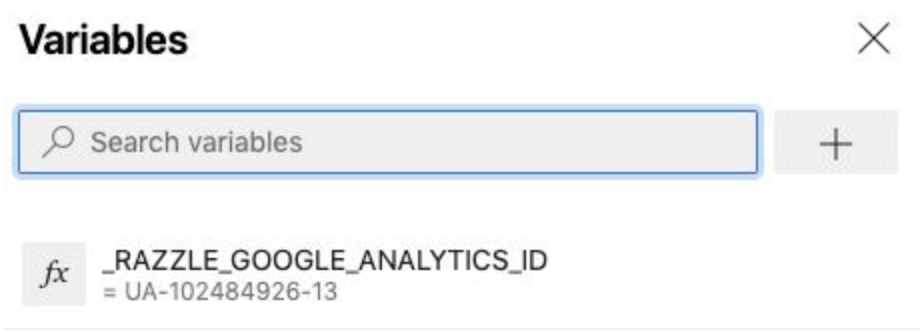
You can manually configure or override the triggers for the pipeline. More importantly, it's here where you can specify where the location of the `azure-pipelines.yaml` file should reside in the repo. You can access the triggers from the menu in the upper right hand side when in edit mode for a pipeline.



To specify the location of the yaml file, hit the YAML tab.

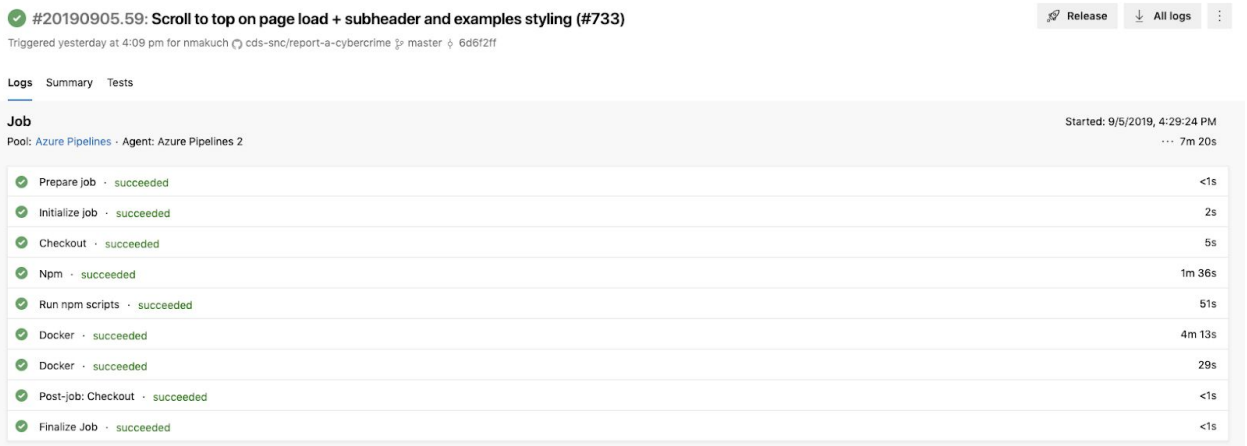
Variables

To access build variables for the pipeline, click the variables button in the upper right hand corner while in edit mode. In our example below, we have one variable - a Google Analytics variable. If you take a look at the yaml file displayed in the beginning of this doc, you will see how the build variable is used.



Builds

Builds are instances of the pipeline that have been run. Builds in progress and build history can be viewed from the Builds link the left hand menu. You can select a specific build and examine the results.



Testing a pipeline

When testing a pipeline, it's recommended that you create a branch in your repo and configure the pipeline trigger for that branch. Additionally, you can manually trigger the pipeline through the run function (accessible while in edit mode) rather than having to engage the trigger through the repo.


Variables
Run