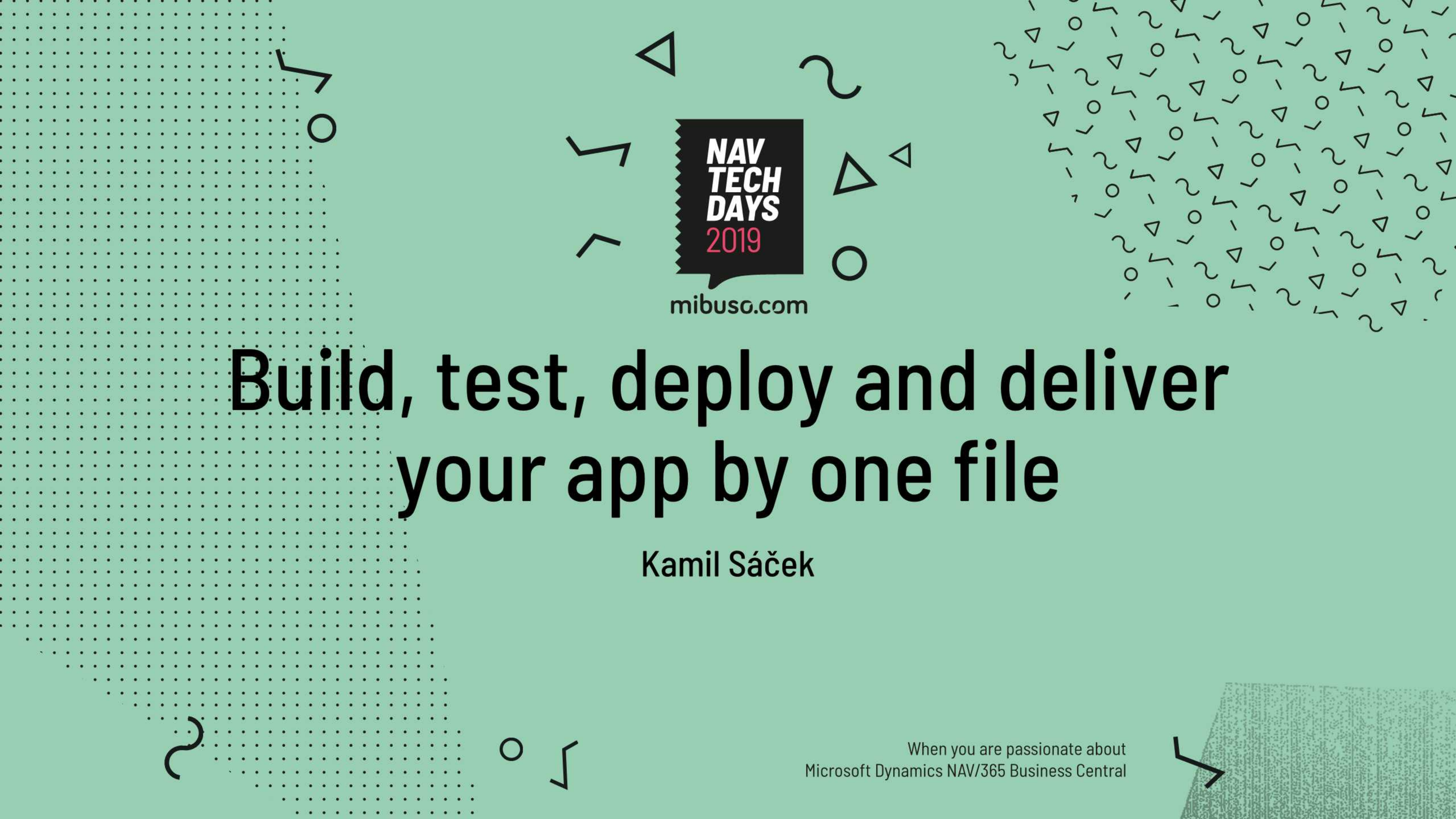


Build, test, deploy and deliver your app by one file

Kamil Sáček

When you are passionate about
Microsoft Dynamics NAV/365 Business Central

Recapitulation

NAV TECH DAYS 2019
mibuso.com

Agenda 2018:

- What is CI/CD
- Create repository
- Create build pipeline
- Branch policy
- Delivery pipeline
- Deployment pipeline

Continuous Irritation & Continuous Distraction

Current state – please, vote NOW!

NAV TECH DAYS 2019
mibuso.com

How long it takes you from creating new AL App to have it installed on Production environment (PerTenantApp, online)?

What you need along the way?

Create app

Develop

Build/test

Deploy to QA

Internal Test

Deploy to Sandbox

Acceptance Test

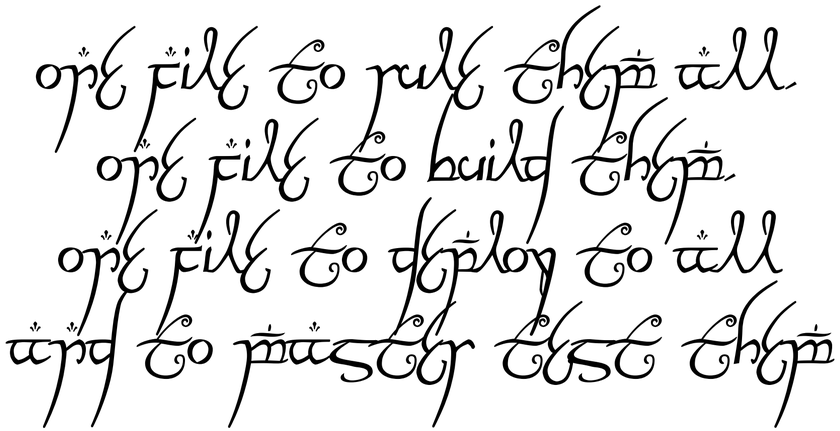
Deploy to Production

Maintain it next XYZ years

How complex is to create CI/CD pipelines for this?

Let’s do that by one file…

We'll do this by one file...


One File to rule them all
One File to build them
One File to deploy to all
And to master Dust them

NAV TECH DAYS 2019
mibuso.com

One file to rule them all,
one file to build them,
one file to deploy to all
and to master test them

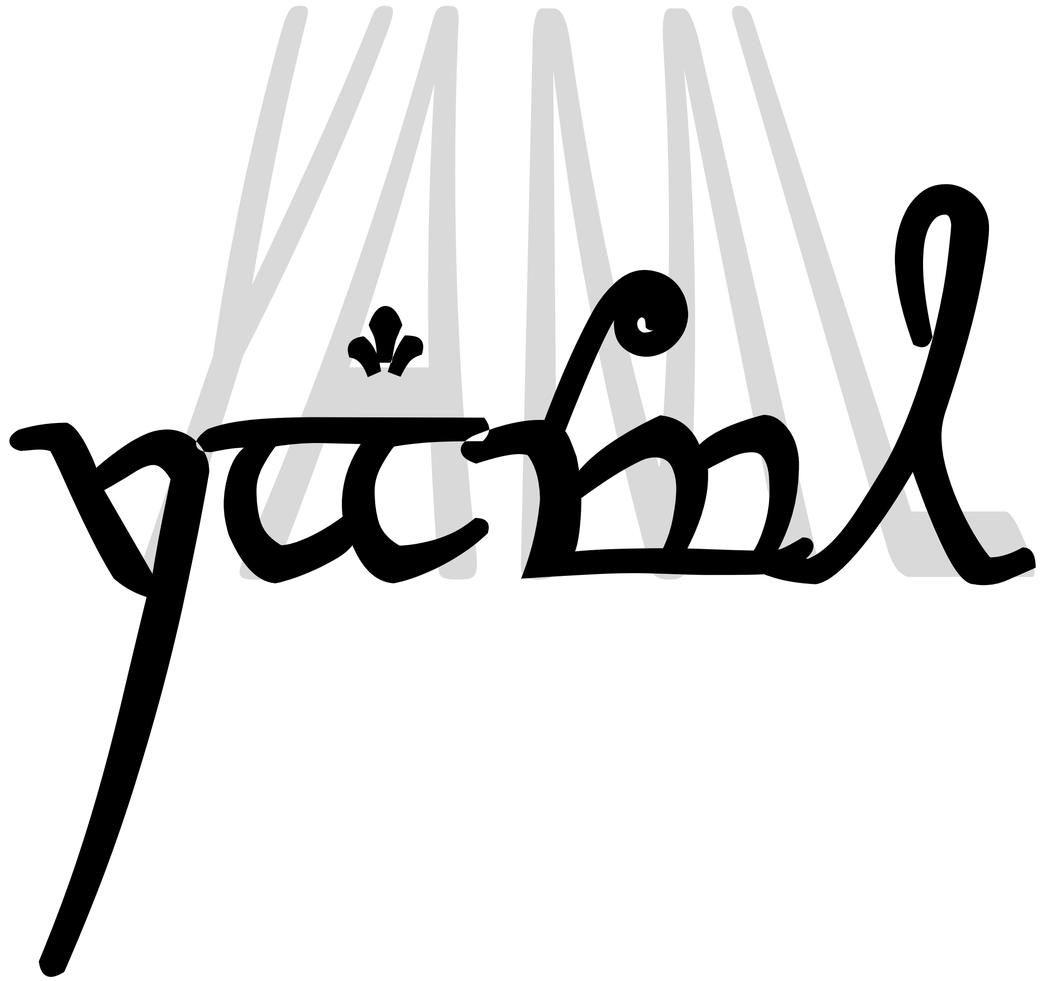
NAV
TECH
DAYS
2019
mibuso.com

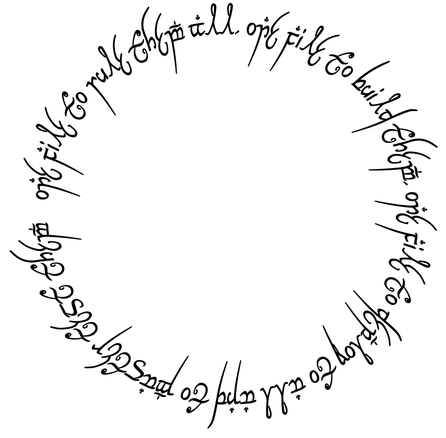
NAV TECH DAYS 2019

mibuso.com

NAV TECH DAYS 2019

mibuso.com

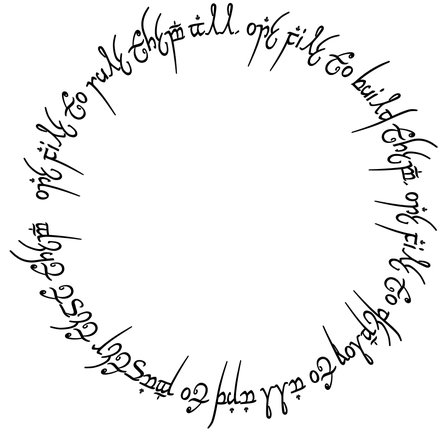




YAML pipeline

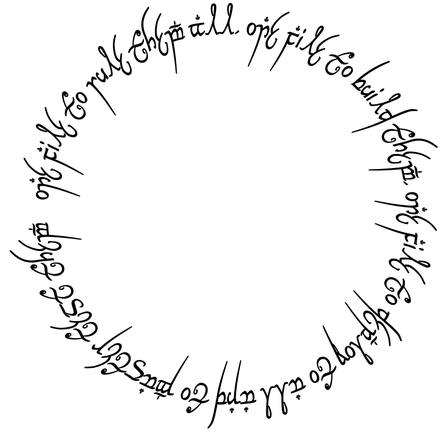
NAV TECH DAYS 2019
mibuso.com

- Today we will look at possibilities of YAML pipeline
- Will try to use it in a way to be as simple as possible (KISS principle)



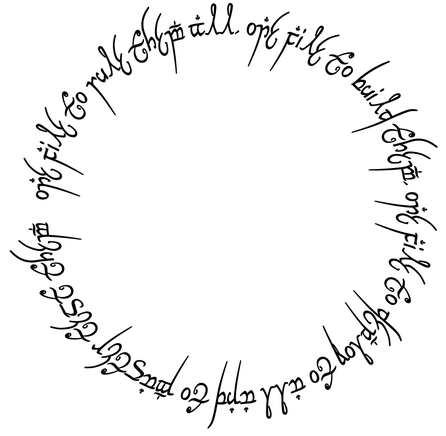
YAML pipeline part

- Variables
- Resources
- Triggers
- Stages
  - Jobs
    - Steps



YAML pipeline part

- Two types of Job
  - Build (CI)
    - Forge source code to application
      - Download source code by default
      - Create artifacts
  - Deployment (CD)
    - Takes application and deploy/deliver it
      - Download artifacts from previous build job/stage by default
      - Could create artifacts

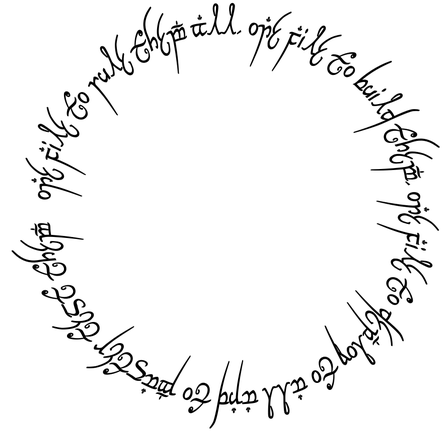


Using variables in YAML pipeline

NAV TECH DAYS 2019
mibuso.com

- There are multiple syntaxes to use variables in pipeline:

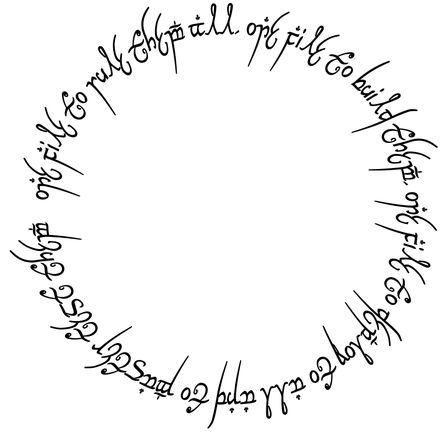
- Parse time expression:
  - ${{ expression }}
    - Where expression could be:
      - variables.VarName
      - variables['VarName']
      - or something else…
    - Access only to parameters and statically defined variables



Using variables in YAML pipeline

- There are multiple syntaxes to use variables in pipeline:

- Runtime expression
  - $[expression]
    - Access to more variables, but no parameters

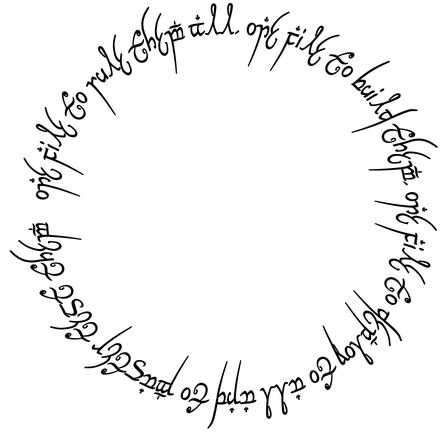


Using variables in YAML pipeline

NAV TECH DAYS 2019
mibuso.com

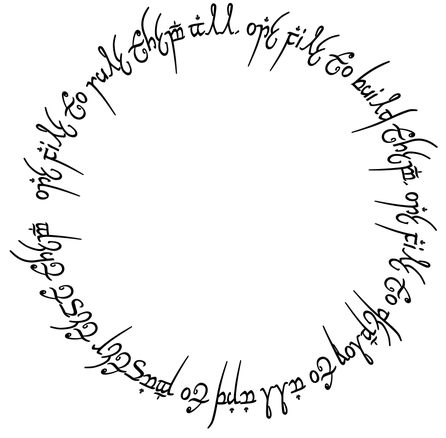
- There are multiple syntaxes to use variables in pipeline:

- Macro
  - $(VarName)
    - Expanded in runtime



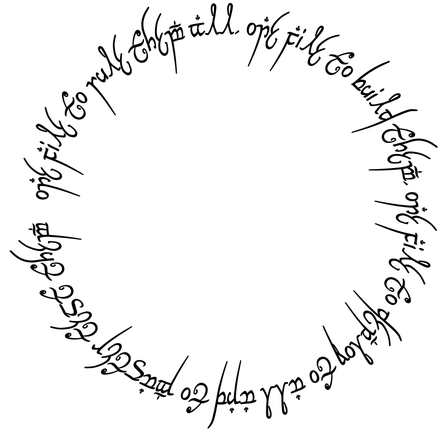
Time to look at real examples

Build



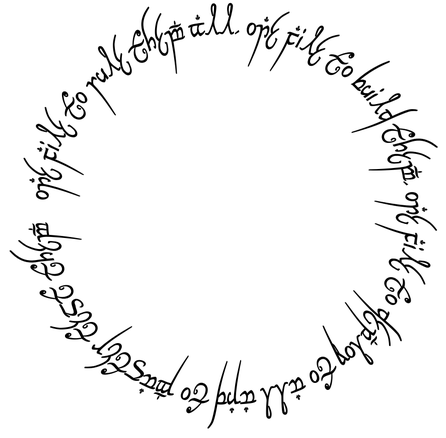
Time to look at real examples

Build + Deploy test



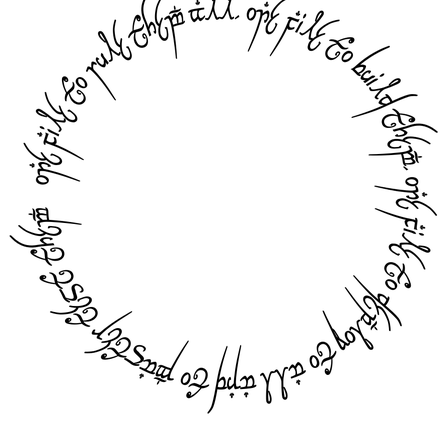
Time to look at real examples

Build + Deploy Test + Deploy to QA



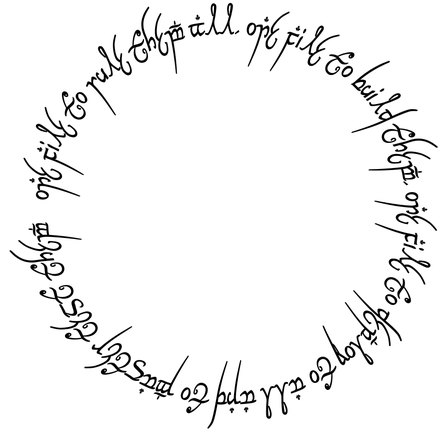
Time to look at real examples

Build + Deploy Test + Deploy to QA + Sign and Deliver



Time to look at real examples

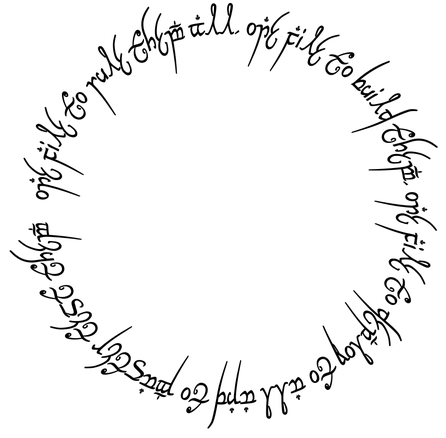
Build + Deploy Test + Deploy to QA + Sign and Deliver + Deploy to Sandbox




DEMO

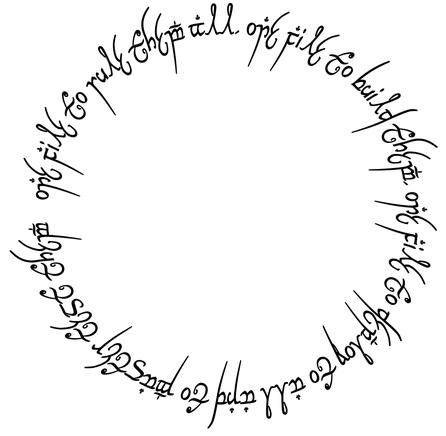
Additional requests:

Pull request => Build only



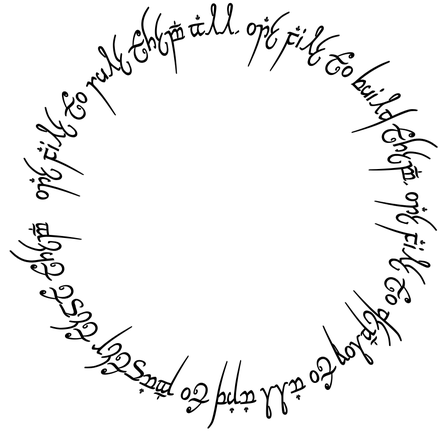
Additional requests:

Deploy only master or release branch



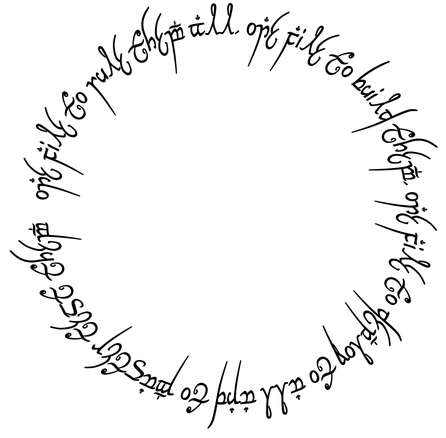
Additional requests:

Scheduled build with master image



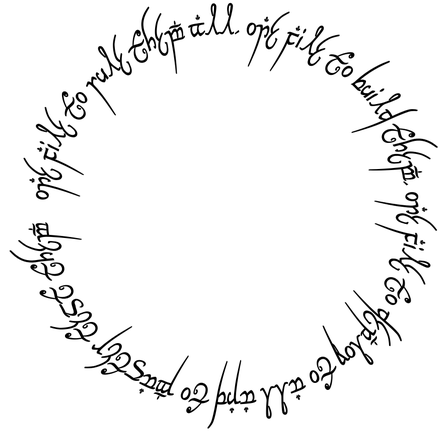

DEMO

Do you like the result?



Time to another vote...

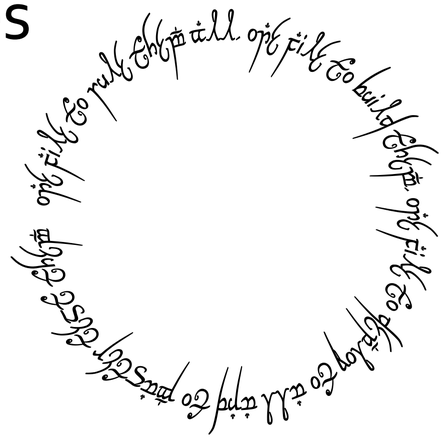
NAV TECH DAYS 2019
mibuso.com



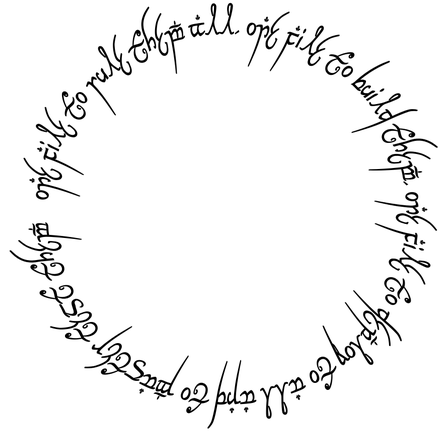
Me not...

Pains of “ONE file rules them all”

- Hard/Impossible to combine different triggers (cannot be dynamic)
- Having Deployment to Sandbox/Production as part of standard CI pipeline will fail pipeline if you cancel the deployment
- Release will take wrong last successful build
- Impossible to separate results by type of build
  - e.g. to have separate widgets on the Dashboard
- Too complex file to maintain
- Change of your process means manual update of all repositories
- Heavy copy/paste when e.g. multiple deployment targets

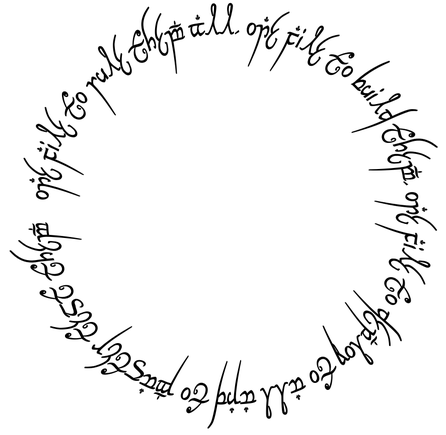


Let’s simplify it… ☺

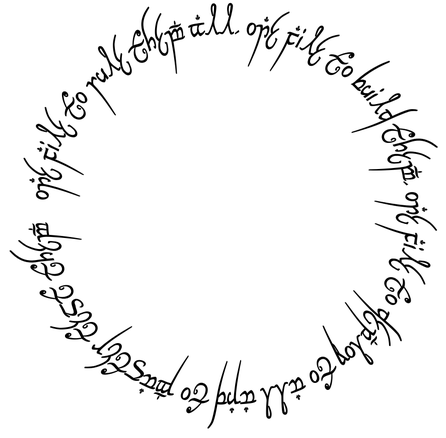


Templates

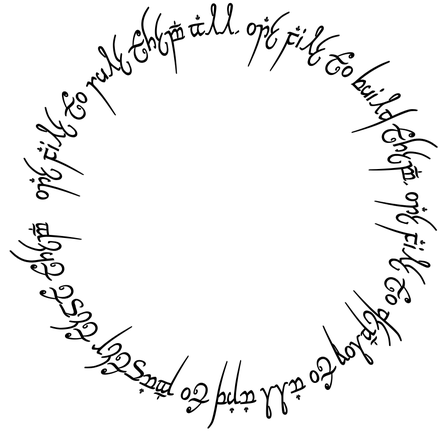
- Move what is shared to shared place
- Leave in App repository what is/could be App specific
- Separate by purpose (build, check, deploy)



Let’s go and break the file!



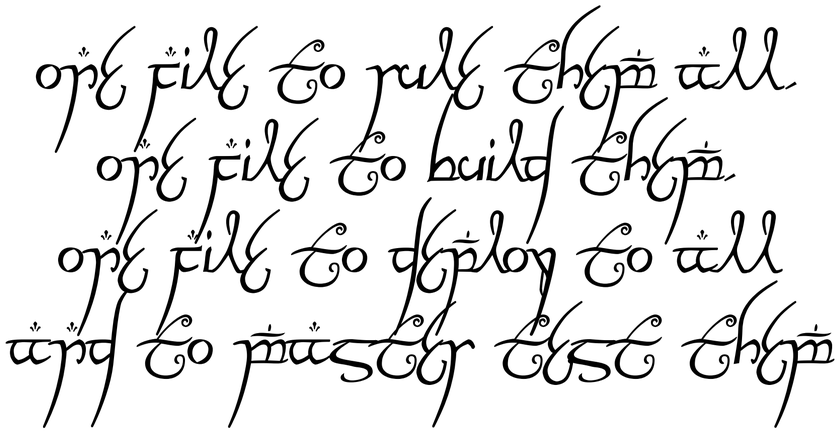
“Templated” YAML pipeline




DEMO

NAV TECH DAYS 2019

mibuso.com


One File to rule them all
One File to build them
One File to deploy to all
And to master Dust them

NAV TECH DAYS 2019
mibuso.com

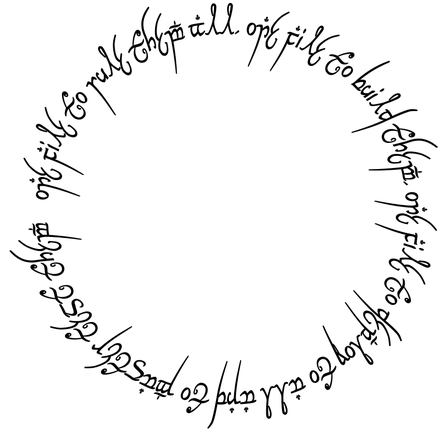
One file to rule them all,

one file to build them,

one file to deploy to all

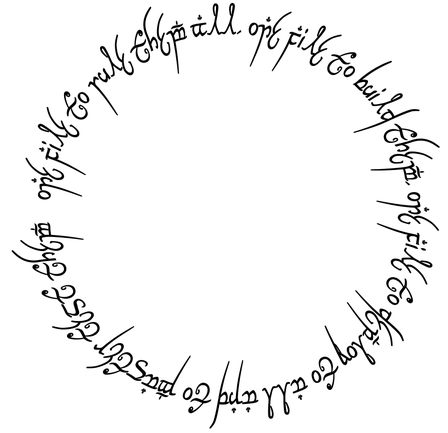
and to master test them

- Now we have
  - Multiple YAML files for multiple pipelines
  - Repeatable blocks
  - Configurable pipelines
- We need to create three pipelines for each app
  - Build and deploy to QA
  - Release to Sandbox/Production
  - Test with master image (vNext compatibility)
- We need to create the pipelines manually
- Or…



Time to another vote…

- How many apps you can create per hour with all three pipelines, distinct ID range, new GUIDs…



App creation script

How to deploy to OnPrem?

NAV TECH DAYS 2019
mibuso.com

1. Take the app, copy it to remote server, install through powershell

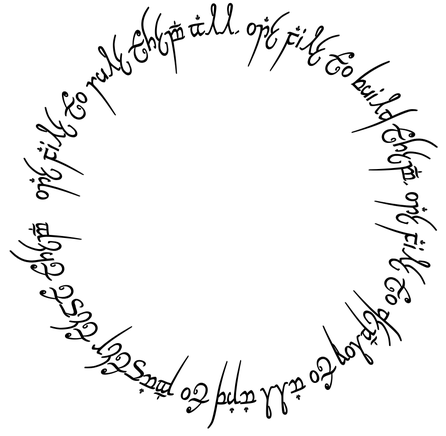
2. Build agent running on the server + powershell (security!!!)

3. Automation API
   - Accessible from internet or some server with build agent
   - Hybrid network

4. Mark the builds, run script (semi-automatic)

5. Install through nuget and other ways

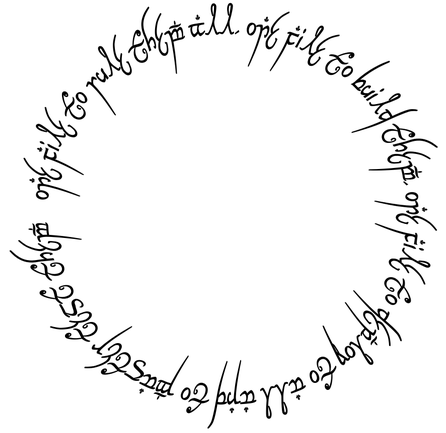
6. Do not forget to update license!



Install from build through script

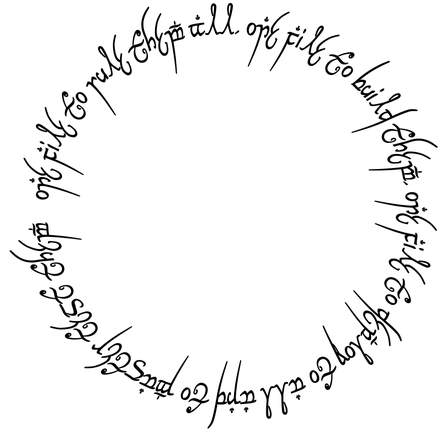
Summary

- Having all in one file is not nice and practical
- Multiple files are better for maintenance
- Automatize App and pipeline creation
- Automatize OnPrem Deployment (VARs)
- Allow download of the apps (ISVs)
- Translation is product of build



NAV TECH DAYS 2019
mibuso.com

Some tips around YAML



Variable expansion

NAV TECH DAYS 2019
mibuso.com

- Recursive expansion
  - On the agent, variables referenced using $( ) syntax will be recursively expanded. However, for service-side operations such as setting display names, variables are not expanded recursively. For example:

- variables:
- myInner: someValue
- myOuter: $(myInner)

- steps:
- - script: echo $(myOuter) # prints "someValue"
- displayName: Variable is $(myOuter) # display name is "Variable is $(myInner)"

Cryptic errors

- An error occurred while loading the YAML build pipeline. wrong number of segments

- Wrong version of task in yaml
- If you install Extension, only last version of each task is available!
- On old installed tenants, the old versions are still available
- If tasks were renamed in the Azure DevOps extension, on new installs only new names exists, on old both

Skipping CI if you really need it

NAV TECH DAYS 2019
mibuso.com

- You can also tell Azure Pipelines to skip running a pipeline that a commit would normally trigger. Just include [skip ci] in the commit message or description of the HEAD commit and Azure Pipelines will skip running CI. You can also use any of the variations below. This is supported for commits to Azure Repos Git, Bitbucket Cloud, GitHub, and GitHub Enterprise Server.

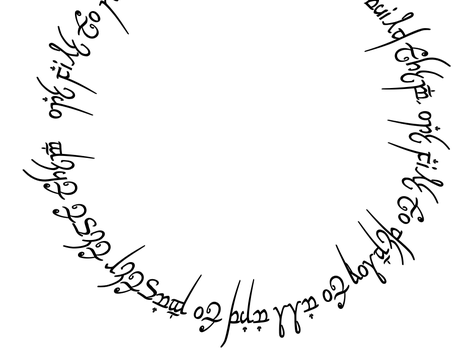
- [skip ci] or [ci skip]
- skip-checks: true
- [skip azurepipelines] or [azurepipelines skip]
- [skip azpipelines] or [azpipelines skip]
- [skip azp] or [azp skip]
- ***NO_CI***

NAV TECH DAYS 2019
mibuso.com

Q&A

Any Questions?

#BCALHelp



NAV TECH DAYS 2019
mibuso.com