Visual Studio LIVE! | San Diego

EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

W24 Creating Reusable CI/CD with Azure DevOps Pipeline Templating

Josh Johanning

Senior DevOps Architect

GitHub

Level: Advanced

Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for “Converge360 Events” in your app store
- Find this session on the Agenda tab
- Click “Session Evaluation”
- Thank you!

Josh Johanning

@joshjohanning

Senior DevOps Architect, GitHub

Email: joshjohanning@github.com

Website: josh-ops.com

From: Madison, WI

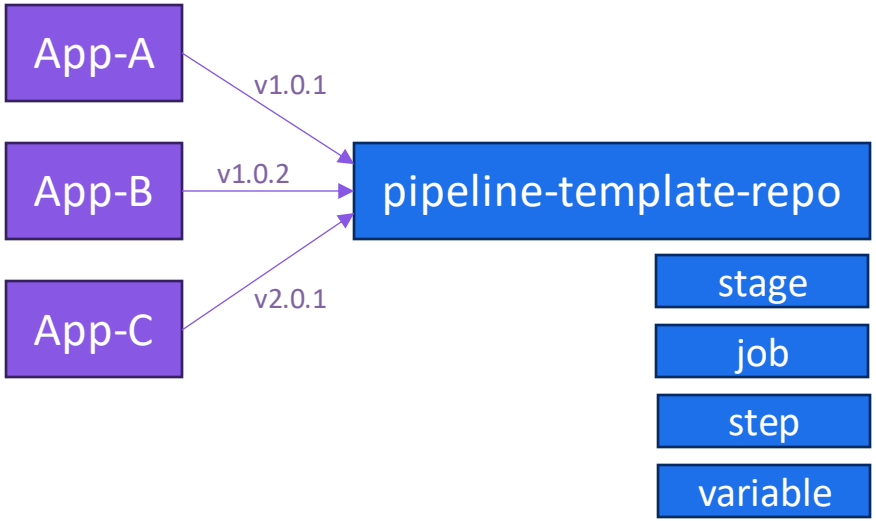
Session Agenda

- What is a Pipeline Template?
- Writing and Using Templates
- Managing Updates
- Demo and Samples

What are Pipeline Templates?

What is a Template?

- Templates let you define reusable content, logic, and parameters in YAML pipelines
- Speed up pipeline development for similar applications
- Encourage reusability and consistency for pipelines
- Versioned to preserve idempotence



Types of Templates

- Stage templates
  - Job templates
    - Step templates
      - Variable templates

Stage templates

- Templates out the entire stage
- Keeps YML simple
- Minimal flexibility with adding additional jobs in stages

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - stage templates
trigger:
- main

pool:
  vmImage: 'ubuntu-latest'

stages:
- template: templates/build.yml
  parameters:
    directory: my-web-app
- template: templates/deploy.yml
  parameters:
    environment: dev
- template: templates/deploy.yml
  parameters:
    environment: prod
```

Job templates

- Templates out a job
- YML is slightly more complex than stage template
- More flexibility on what jobs happen in each stage, variables, etc.

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - job templates
trigger:
- main

stages:
- stage: 'Build'
  variables:
    buildConfiguration: 'Release'
  jobs:
  - template: dotnet-core-web/web-app.yml@templates
    parameters:
      buildConfiguration: ${{ variables.buildConfiguration }}

- stage: deployDev
  displayName: Deploy to Dev
  condition: and(succeeded(), ne(variables['Build.Reason'], 'PullRequest'))
  variables:
    - name: azureSubscription
      value: demo-mslearn-tailspin-azure
    - group: tailspin.DEV
  jobs:
  - template: dotnet-core-web/deploy-web-app.yml@templates
    parameters:
      environment:
        name: 'dev'
```

Step templates

- Templates out a step
- Easier to create than an extension for a custom task

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - step templates
trigger:
- main

stages:
- stage: 'Build'
  variables:
    buildConfiguration: 'Release'
  jobs:
  - job: Build
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - script: sell paper # do stuff
    - template: templates/build-steps.yml
      parameters:
        net-version: 7.0
        buildConfiguration: ${{ variables.buildConfiguration }}
    - script: thwart dwight # do more stuff
```

Variable templates

- Templates out a set of variables
- Can also use parameters

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - step templates
variables:
  - template: vars.yml
    parameters:
      DIRECTORY: "dunder-mifflin/party-planning-committee"

pool:
  vmImage: 'ubuntu-latest'

stages:
- stage: Release_Stage
  displayName: Release Version
  variables:
  - template: vars.yml
    parameters:
      DIRECTORY: "dunder-mifflin/warehouse-crew"
  jobs:
  - job: A
    steps:
    - bash: $(RELEASE_COMMAND) #output release command var
```

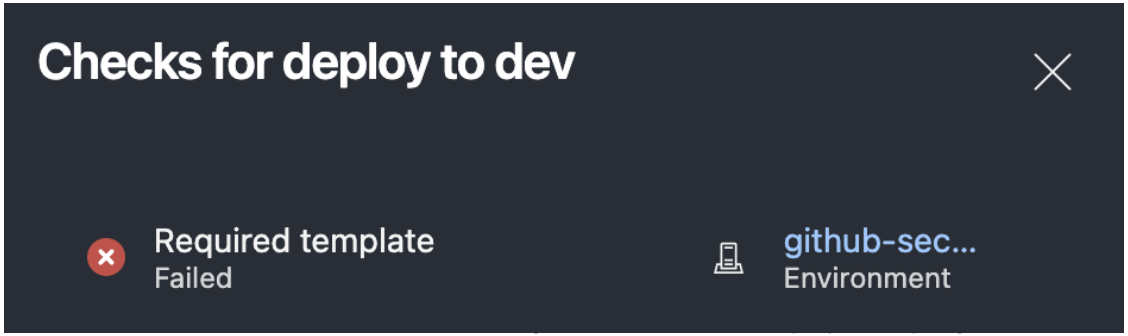
These are “Includes” Templates

… because they are additive

But there is another type of template…

Extends Template

- An Extends template is another type of template
- Security/compliance benefits:
  - Used to control what is allowed in a pipeline
  - You can enforce that a pipeline extends from a particular template
  - An environment or resource can have the required template check added
- More complex – not exactly recommended
  - If the required check is set, each pipeline job will have to refer to this template
  - If using, recommend to use job templates in combination



Extends template

- You HAVE to refer to the main extends template to run against protected environments

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - extends templates
trigger:
- main

extends:
  template: extends.yml@templates
  parameters:
    buildJobs:
    # job template
    - template: my-build-job.yml@templates
    deployStages:
    - stage: dev
      displayName: deploy to dev
      jobs:
      # job template
      - template: sample-deployment-job.yml@templates
        parameters:
          environment: dev
    - stage: prod
      displayName: deploy to prod
      jobs:
      - template: sample-deployment-job.yml@templates
        parameters:
          environment: prod
```

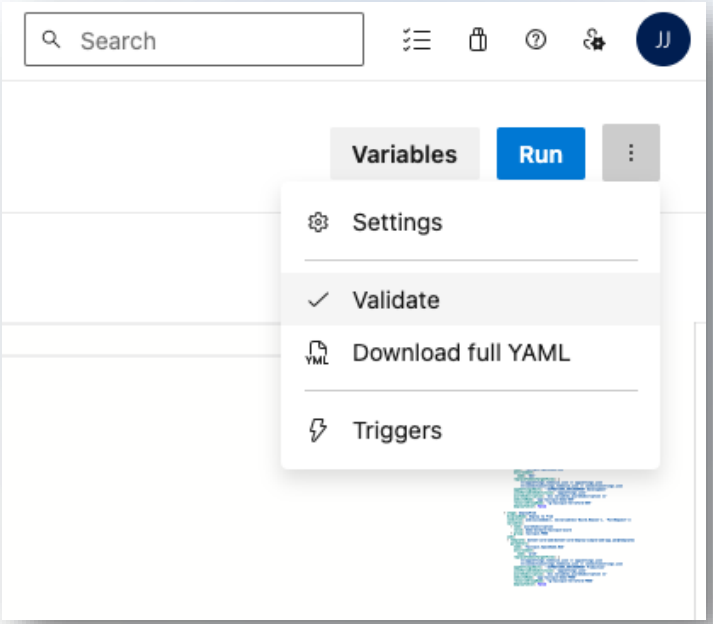
Some other pro-tips for writing

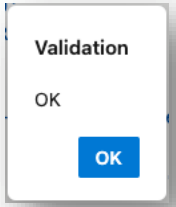
Parameter Types

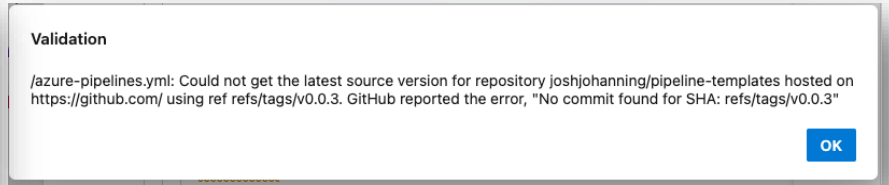
Data Type	Notes
string	generic string
number	may be restricted to values:, otherwise any number-like string is accepted
boolean	true or false
object	any yml structure
step	a single step
stepList	sequence of steps
job	a single job
jobList	sequence of jobs
deployment	a single deployment job
deploymentList	sequence of deployment jobs
stage	a single stage
stageList	sequence of stages

Validating YML in Editor

- Two options available:
  1. Validate
     - **Will return OK or return the error**
     - **Reduces failed pipeline runs**
  2. Download full YAML
     - **Downloads rendered YML**







Limits

- Maximum 100 separate YML files may be referenced
  - directly or indirectly
- Maximum 20 levels of template nesting
  - templates referencing other templates
- No more than 10 MB of memory consumed while parsing the YML
  - in practice, this is typically between 600 KB - 2 MB of on-disk YML
  - depends on the specific features used
- Other unsupported YML features:
  - Anchors
  - Complex keys and sets

And YML expressions!

If

- Conditionally run a job / step
- Different than a Condition since it doesn't show as skipped

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - ifs
stages:
- stage: build
  jobs:
  - template: dotnet/build.yml@templates
    parameters:
      buildConfiguration: 'Release'
      runBuild: 'false'
      buildDirectory: Tailspin.SpaceGame.Web
      runDotNetCoreTests: 'false'
```

dotnet/build.yml

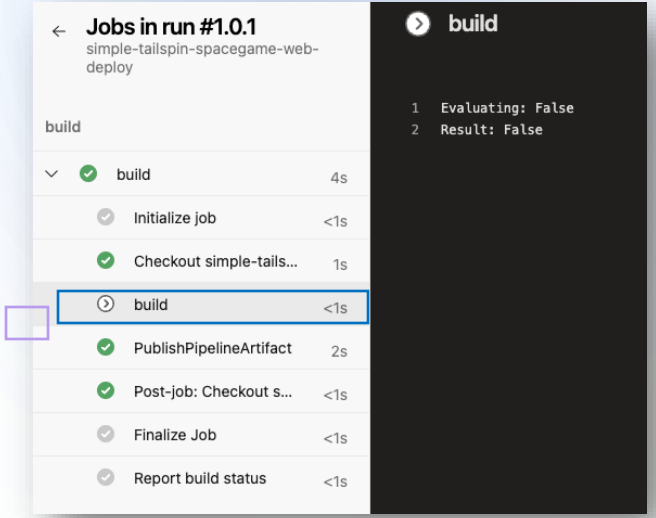
```yaml
# File: dotnet/build.yml - ifs
parameters:
  buildConfiguration: 'Release'
  runBuild: 'true'
  runDotNetCoreTests: 'true'

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

    steps:
    - checkout: 'self'

    - script: |
        dotnet publish \
          -c ${{ parameters.buildConfiguration}} \
          -o $(Build.ArtifactStagingDirectory) \
      condition: ${{ eq(parameters.runBuild, true) }}
      displayName: build

    - ${{ if eq(parameters.runDotNetCoreTests, 'true') }}:
      - task: DotNetCoreCLI@2
        displayName: 'dotnet test'
        inputs:
          command: test
          projects: '**/*[Tt]ests/*.csproj'
          publishTestResults: true

    - publish: '$(Build.ArtifactStagingDirectory)'
      artifact: 'drop'
```

If

- Compare:
  If to a conditional

dotnet/build.yml

```yaml
- script: |
    dotnet publish \
      -c ${{ parameters.buildConfiguration}} \
      -o $(Build.ArtifactStagingDirectory) \
  condition: ${{ eq(parameters.runBuild, true) }}
  displayName: build

- ${{ if eq(parameters.runDotNetCoreTests, 'true') }}:
  - task: DotNetCoreCLI@2
    displayName: 'dotnet test'
    inputs:
      command: test
      projects: '**/*[Tt]ests/*.csproj'
      publishTestResults: true
```



Each Loop

- Used for more complex objects
- Alternative to using ifs /conditions
- Can be used to have multiple resource types in same template

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - loops

- stage: deployDev
  jobs:
  - template: dotnet-core/dotnet-core-deploy.yml@templates
    parameters:
      environment: 'Dev'
      services:
      - name: 'my.Web.app'
        vmImage: 'windows-latest'
        iis:
        - websiteName: 'devtesting'
      - name: 'my.API.app'
        vmImage: 'ubuntu-latest'
        azure:
        - azureSubscription: ${{ variables.azureSubscription }}
          webApp:
          - websiteName: 'my-api-app'
            resourceGroupName: 'rg-api'
            slotName: ${{ variables.environment }}
      - name: 'my.Func.app'
        runDeploy: true
        azure:
        - azureSubscription: ${{ variables.azureSubscription }}
          functionApp:
          - appName: 'my-Func-app'
```

dotnet-core-deploy.yml

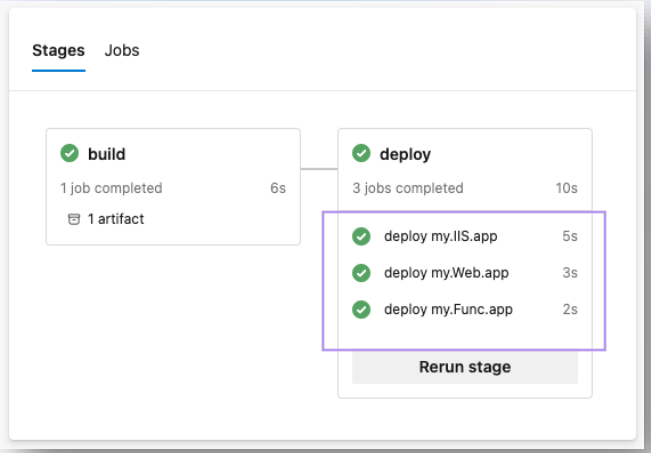
```yaml
# File: dotnet-core-deploy.yml - loops
parameters:
  environment: 'Dev'
  services: []

jobs:

- ${{ each s in parameters.services }}:

  - deployment:
    displayName: deploy ${{ s.name }}
    pool:
      vmImage: ${{ s.vmImage }}
    environment: ${{ parameters.environment }}
    strategy:
      runOnce:
        deploy:
          steps:
          - script: |
              echo "deploying ${{ s.name }} to ${{ parameters.environment }}"

          - ${{ each i in s.iis }}:
            - task: IISWebAppDeploymentOnMachineGroup@0
              inputs:
                WebSiteName: ${{ i.websiteName }}

          - ${{ each az in s.azure }}:
            - ${{ each azweb in az.webApp }}:
              - task: AzureRmWebAppDeployment@4
                inputs:
                  azureSubscription: ${{ az.azureSubscription }}
                  ResourceGroupName: ${{ azweb.resourceGroupName }}
                  WebAppName: ${{ azweb.websiteName }}
                  SlotName: ${{ azweb.slotName }}

            - ${{ each func in az.functionApp }}:
              - task: AzureFunctionApp@1
                inputs:
                  azureSubscription: ${{ az.azureSubscription }}
                  appName: ${{ func.appName }}
```

Each Loop

- Useful for steps or jobs

azure-pipelines.yml

```yaml
# File: azure-pipelines.yml - loops
- stage: deploy
  jobs:
  - template: web/webapp.yml@templatesdeploy
    parameters:
      environment: 'Dev'
      services:
      - name: 'my.IIS.app'
        vmImage: 'windows-latest'
      - name: 'my.Web.app'
        vmImage: 'ubuntu-latest'
      - name: 'my.Func.app'
        vmImage: 'ubuntu-latest'
```



Other Template Expressions

- Format
  - Simple string token replacement
- Coalesce
  - Evaluates the first non-empty, non-null string argument
- Insertion
  - Insert an entire stepList into a job
  - Insertion can also be used to insert into a mapping
    - For example, bringing in an unknown number of variables under the variables keyword

- All template expressions are wrapped in ${{ <expression> }}, like:
  - ${{ format('{0} Build', parameters.os) }}

Writing and Using Templates

Referencing Templates

- The template can live in the same repo
  - Refer to the file path
- More likely, you will be referencing a pipeline template in another repo
  - Add the other repository as a reference
  - The repository can live in GitHub or Azure DevOps
  - Best practice to pin to a tag and not a branch ref

azure-pipelines.yml

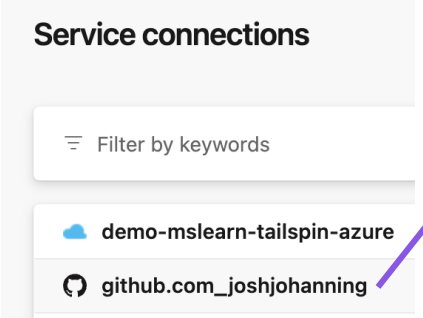
```yaml
# github repo reference
resources:
  repositories:
  - repository: templates
    type: github
    name: joshjohanning/pipeline-templates
    endpoint: github.com_joshjohanning
    ref: refs/tags/v0.0.2
```

azure-pipelines.yml

```yaml
# azure devops repo reference
resources:
  repositories:
  - repository: templates
    name: pipeline-templates
    ref: main
    type: git
```

Referencing Templates

- When referencing a template in another repo, use the “@<id>”
- Need to create a GitHub endpoint
  - For Auth, can use OAuth flow
  - Or PAT



azure-pipelines.yml

```yaml
# github repo reference
resources:
  repositories:
  - repository: templates
    type: github
    name: joshjohanning/pipeline-templates
    endpoint: github.com_joshjohanning
    ref: refs/tags/v0.0.2

stages:
- stage: 'Build'
  variables:
    buildConfiguration: 'Release'
  jobs:
  - template: dotnet-core-web/build-simple-web-app.yml@templates
    parameters:
      buildConfiguration: ${{ variables.buildConfiguration }}
```

Same ID

Referencing

- The pipeline overview page shows what other resources you are referencing
- Such as the branch/tag of the pipeline templates repo(s)

Repositories 2
joshjohanning/simple-tailspin-spacegame-web-... +1
See Sources card for details

Time started and elapsed
Today at 6:01 PM
1m 21s

Related
0 work items
2 published; 1 consumed

Tests and c
Get star

Stages Jobs

Build
1 job completed 51s
100% tests passed
2 artifacts

Deploy to Dev
0/1 completed 18s
deploy Tailspin.SpaceGa... 1...
Cancel

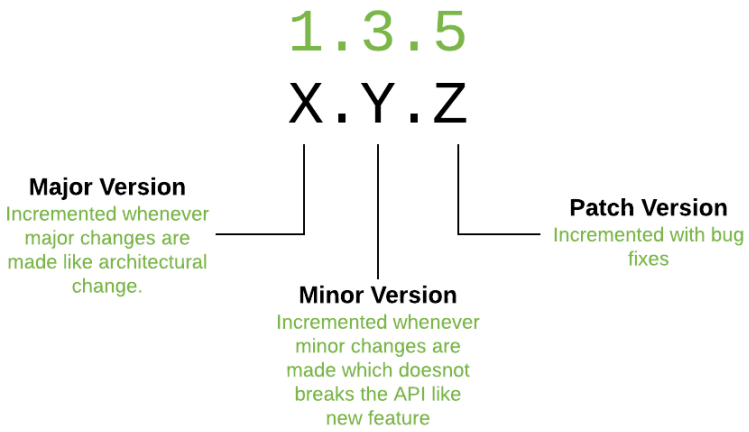
Deploy to Prod
Not started

Sources

Repository	Branch / tag	Version	Related
joshjohanning/simple-tailspin-spacegame-w... GitHub	main	4e2f9852	None
joshjohanning/pipeline-templates GitHub	v0.0.2	97cfd4d6	

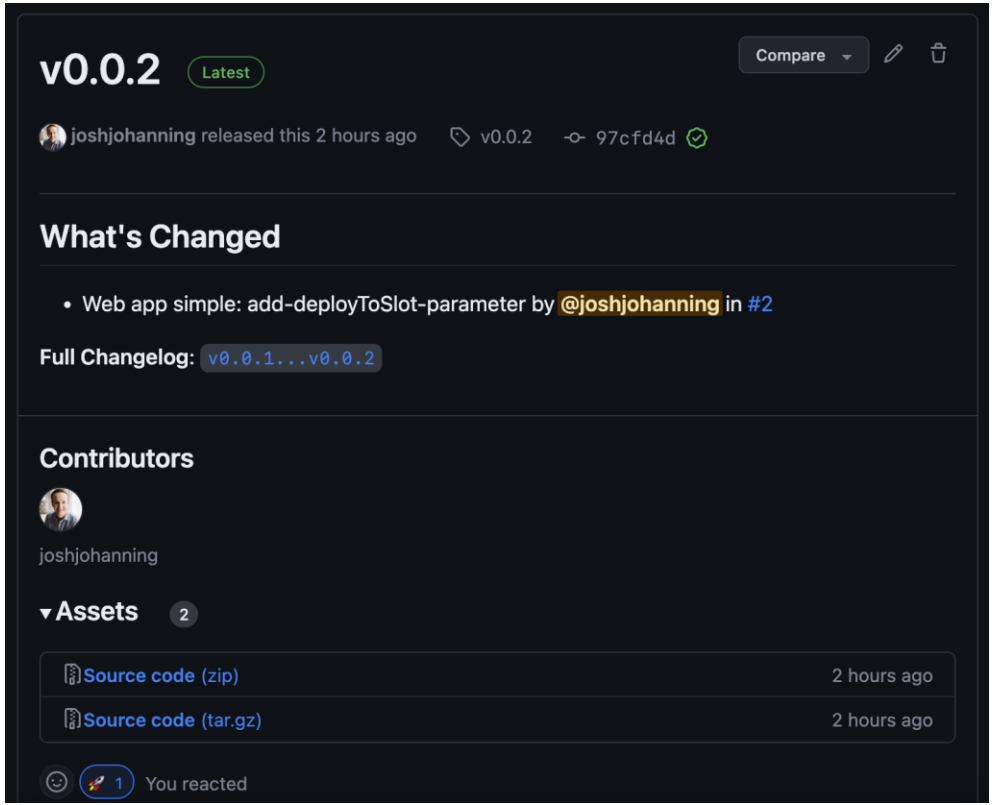
Managing Updates

- Referring to a branch (main) can be easy for rapid iteration, but…
- It is best practice to version the pipeline template with tags
  - That way, if the upstream pipeline template changes, your pipeline will still work
- Use SemVer!



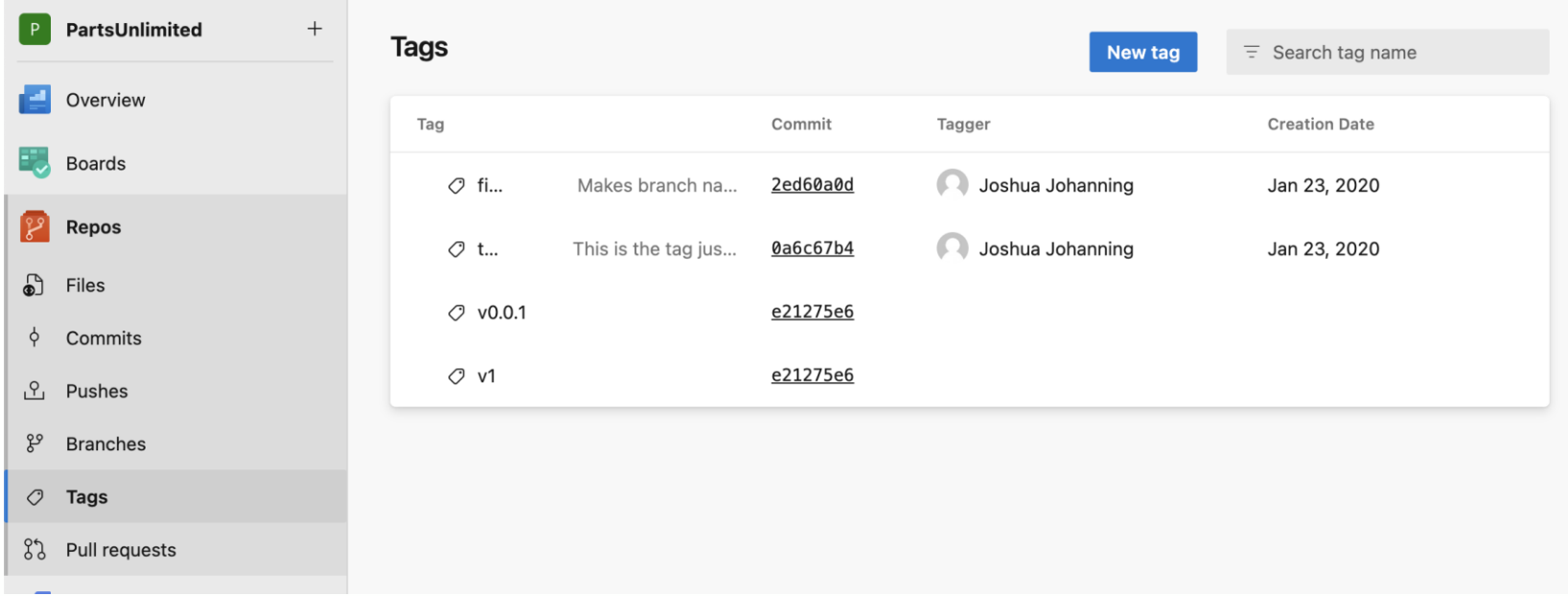
Managing Updates … in GitHub

- Use a release to create the tag
- In GitHub, a release consists of a git tag along with metadata
  - Release notes can be auto-generated from pull requests



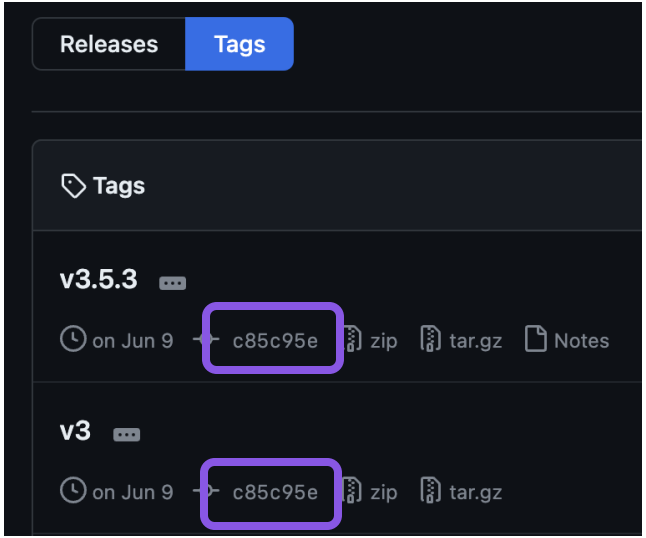
Managing Updates … in Azure DevOps

- Create tags in the UI (or via CLI)



Managing Updates … taking a page from GitHub Actions

- In GitHub Actions, Actions (tasks) often have SemVer version as well as a incrementing major version tag
- This allows users to reference the latest major version without having to increment for EACH minor SemVer tag

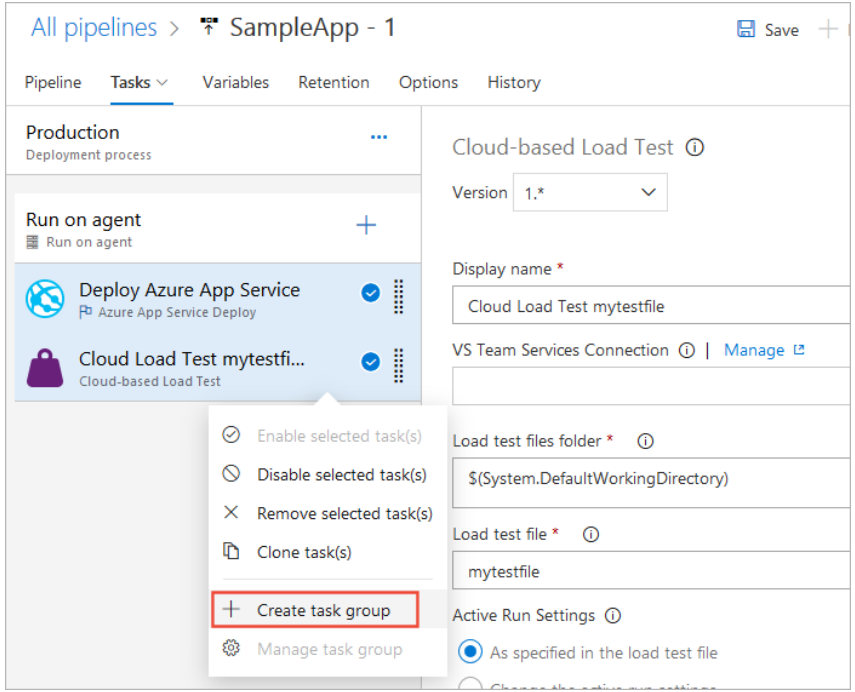


Managing Updates - Git CLI tips

Command	Description
git tag v0.0.1	Create tag locally with current commit hash
git tag v1 --force	Create tag and overwrite if already exists
git push origin v0.0.1	Push local tag to remote
git tag -d v0.0.1	Delete tag locally
git push --delete origin v0.0.1	Delete tag on remote

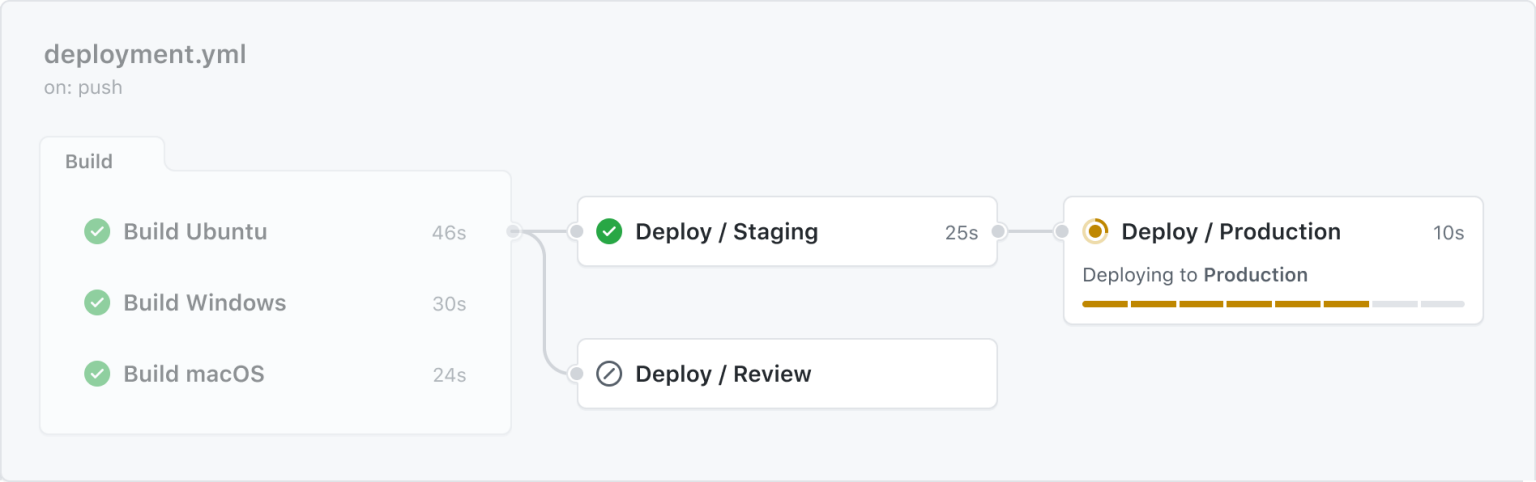
What about for Classic Releases?

- If within the same Team Project:
  - Use Task Groups
  - Can do versioning in task groups too!
- If spread across multiple team projects:
  - Create a custom extension and deploying using tfx npm package
  - Extension can have versions



What about … GitHub? 👀

- Reusable Workflows = Job templates
- Composite Actions = Step templates



Demos and Samples

Samples!

github.com/joshjohanning/pipeline-templates

dev.azure.com/jjohanning0798/pipeline-templates

Additional References

https://josh-ops.com/posts/extends-template/

https://josh-ops.com/posts/pipeline-templates/

https://learn.microsoft.com/en-us/azure/devops/pipelines/process/templates

https://learn.microsoft.com/en-us/azure/devops/pipelines/security/templates

Thank you!

Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for “Converge360 Events” in your app store
- Find this session on the Agenda tab
- Click “Session Evaluation”
- Thank you!