

# DevSecOps, day 2



# 2. Lab: Tasks completed

# You've done some work

- Yesterday, you already completed a few stories!
  - You installed your Dev Workstations.
  - You cloned the Git repositories.
  - You got the local JuiceShop to run.

# Assignment

- Make a task, per team member, for these stories:
  - Setup Lab VM
  - Clone Git repo
  - Run local test.
- Set the tasks AND the stories to "Done".

# For example

- From:

The screenshot shows a Taskboard view for the 'ITVitae team 1 Team' in 'Sprint 1'. The board has three columns: 'To Do', 'Doing', and 'Done'. Each column contains two work items, each with a checkmark icon, a name, a person icon, and a state indicator.

To Do	Doing	Done
<p>131 Setup lab VM Unassigned State To Do</p>	<p>137 Ruben student1 State To Do</p> <p>138 Tabitha student2 State To Do</p>	<p>136 Tess Tess Sluijter State Doing</p>
<p>132 Clone Git repo Unassigned State To Do</p>	<p>140 Ruben student1 State To Do</p> <p>141 Tabitha student2 State To Do</p>	<p>139 Tess Tess Sluijter State Doing</p>

# For example

- To:

The screenshot shows a Taskboard view for the 'ITVitae team 1 Team' in 'Sprint 1'. The board has three columns: 'To Do', 'Doing', and 'Done'. In the 'To Do' column, there are two items: '131 Setup lab VM' (Unassigned, State: Done) and '132 Clone Git repo' (Unassigned, State: Done). A green '+' button is located at the bottom of this column. In the 'Doing' column, there are no items. In the 'Done' column, there are four items: '136 Tess' (Tess Sluijter, State: Done), '137 Ruben' (student1, State: Done), '138 Tabitha' (student2, State: Done), and '139 Tess' (Tess Sluijter, State: Done). A blue '+' button is located at the bottom of this column.

To Do	Doing	Done
<p>131 Setup lab VM Unassigned State: Done</p> <p>132 Clone Git repo Unassigned State: Done</p> <p>+</p>		<p>136 Tess Tess Sluijter State: Done</p> <p>137 Ruben student1 State: Done</p> <p>138 Tabitha student2 State: Done</p> <p>139 Tess Tess Sluijter State: Done</p> <p>140 Ruben student1 State: Done</p> <p>+</p>

# 3. Cooperative software delivery



# Preventing case mixups

## All Repositories

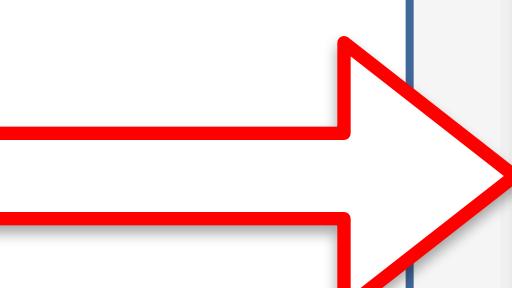
Repositories   Settings   Policies   Security

### Repository Policies

Off **Commit author email validation**  
Block pushes with a commit author email that does not match the following patterns.

Off **File path validation**  
Block pushes from introducing file paths that match the following patterns.

On **Case enforcement**  
Avoid case-sensitivity conflicts by blocking pushes that change name casing on files, folders, branches, and tags. [Learn more](#)



# Branching

- Go to your team's Git repository.
  - Go to "branches" in the side menu.
  - Make three new branches, from "main".
  - Make: *dev, test, prod.*

# Set "dev" as default

The screenshot shows a GitHub 'Branches' page. At the top, there are buttons for 'New branch' and a search bar. Below that, filters for 'Mine', 'All', and 'Stale' are shown. A notification says 'You updated `dev` 8m ago'. On the right, a 'Create a pull request' button is visible. The main table lists four branches:

Branch	C...	Author	Auth...	Behind   Ahead	S...	P...
<code>dev</code>	<a href="#">8e6d</a>	Tess Sl...	8m ...	0 2	<a href="#">?</a>	<a href="#"></a>
<code>master</code>	<a href="#">3d48</a>	Tess Sl...	11m ...	<a href="#">+</a>	New branch	
<code>prod</code>	<a href="#">2c9f</a>	Bjoern ...	Wed...	1 0	<a href="#">New pull request</a>	
<code>test</code>	<a href="#">2c9f</a>	Bjoern ...	Wed...	1 0	<a href="#">Delete branch</a>	

A red box highlights the 'dev' branch. A red arrow points from the 'master' row's context menu to the 'Set as default branch' option, which is also highlighted with a red box. The context menu options are:

- + New branch
- New pull request
- Delete branch
- View files
- View history
- Compare branches
- Set as compare branch
- Set as default branch**

# Branching

- A quick demo:

```
$ cd ~/Documents/Team1JS
$ git fetch
$ git branch -r
$ git switch main
$ git switch dev
```

# Let's cause problems

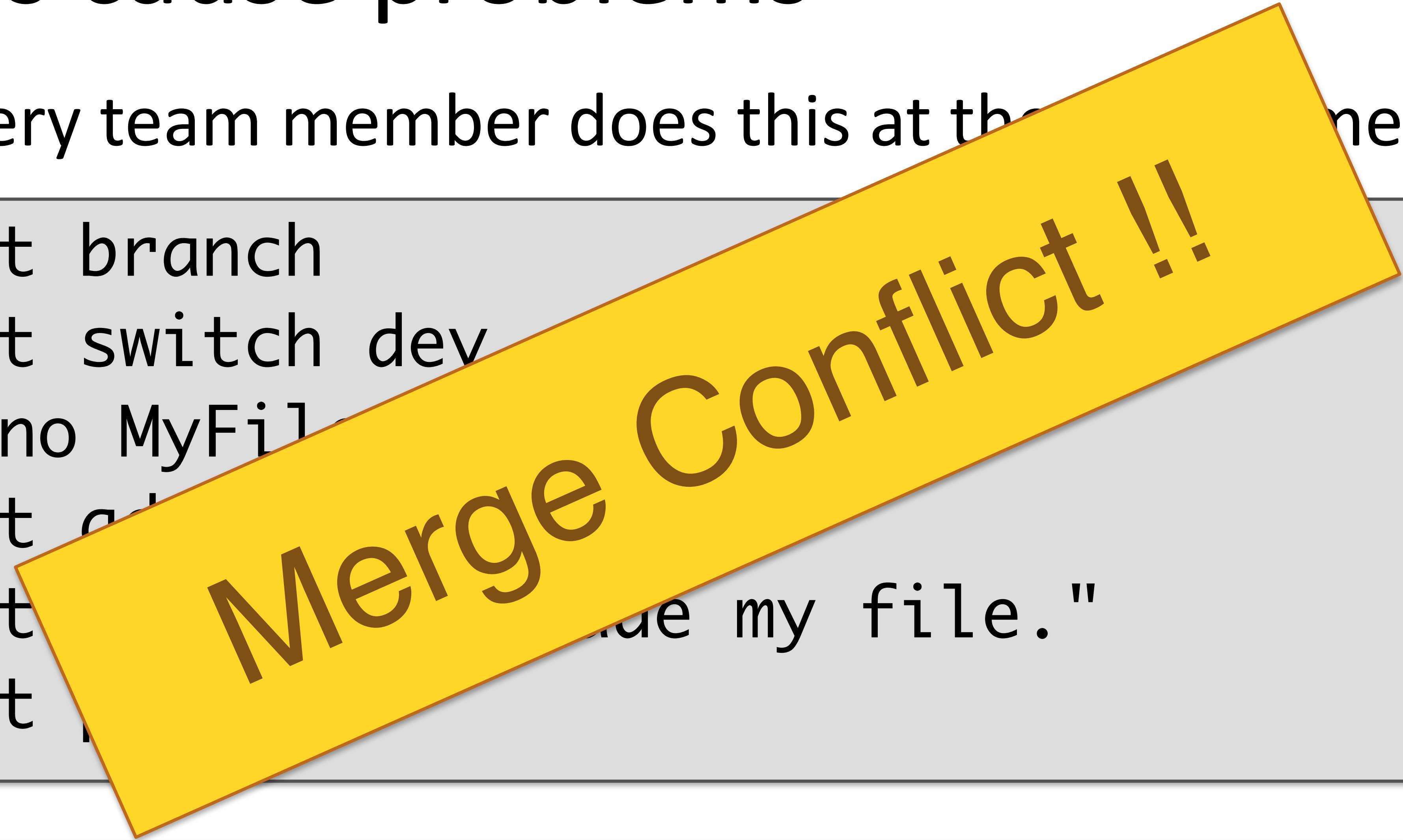
- Every team member does this at the same time.

```
$ git branch  
$ git switch dev  
$ nano MyFile.txt  
$ git add MyFile.txt  
$ git commit -m "Made my file."  
$ git push
```

# Let's cause problems

- Every team member does this at the same time.

```
$ git branch  
$ git switch dev  
$ nano MyFile  
$ git add .  
$ git commit -m "Value my file."  
$ git
```



# Resolving a merge conflict

- Git will tell you which file(s) to edit.
  - The file will clearly show the conflict:

```
<<<<<< HEAD:MyFile.txt
```

```
This was your text.
```

```
=====
```

```
This is their text.
```

```
>>>>>
```

# 6. Lab: Local Docker container

# Dockerfile

- The JuiceShop repository contains a config file.
  - "*Dockerfile*" is used to define an image build.
  - Each line is an instruction to Docker.
- Read the "*Dockerfile*".
  - See if you recognize what we did yesterday.

# Assignment: build container

- Let's makes a container image with the full web app.
  - Like before, "*npm install*" takes a long time.
  - It will take 3 to 15 minutes.

```
$ docker build -t team1:dev .
```

# Assignment: run container

- You can use the container to run your app!

```
$ docker run --rm -p 3000:3000 team1:dev
```

- Access it on <http://localhost:3000>

# Assignment: run container

- Either use your host OS' browser.
  - Or test with *curl* on the DEV VM.



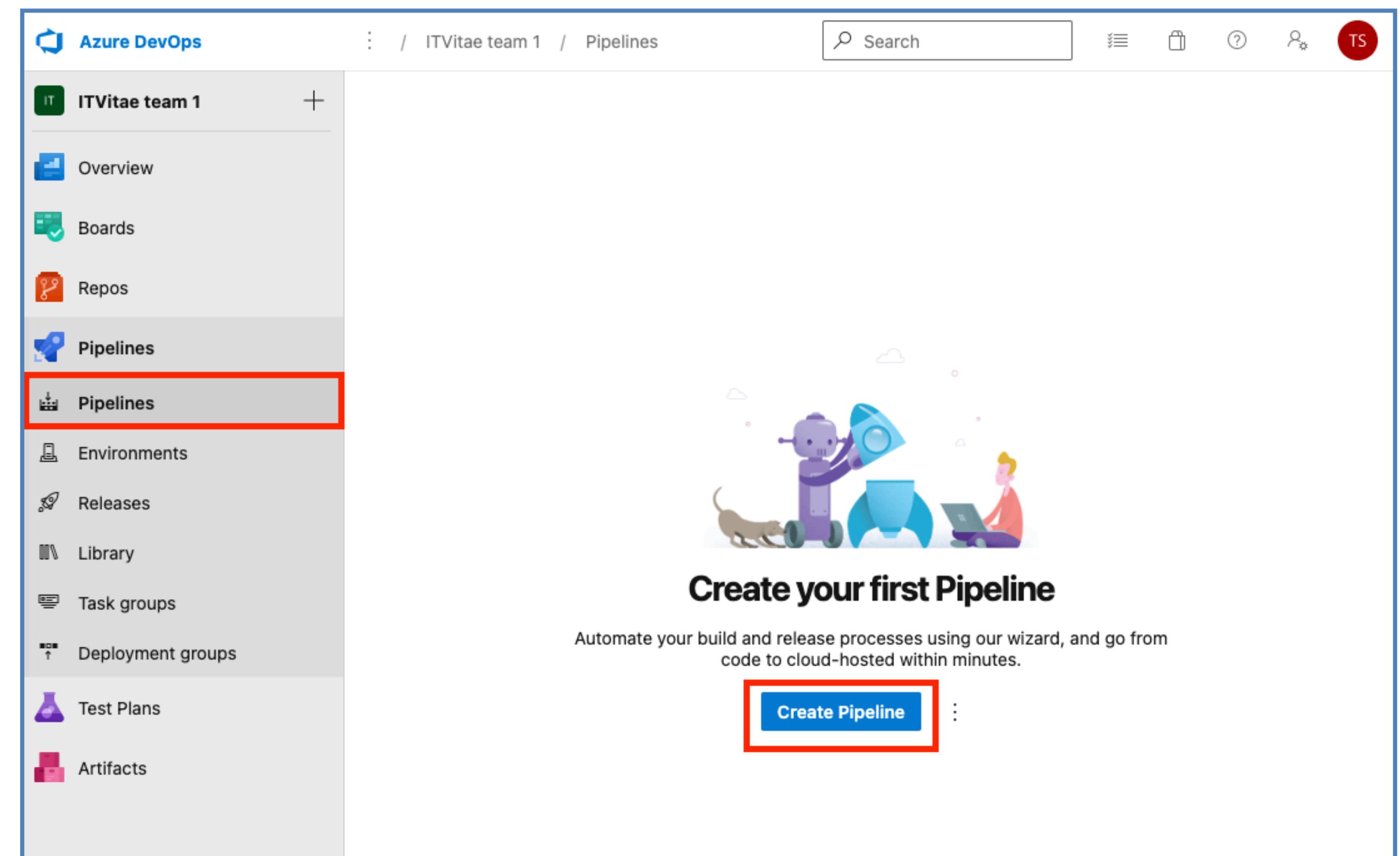
# 9. Lab: CI/CD pipelines

# What we will make

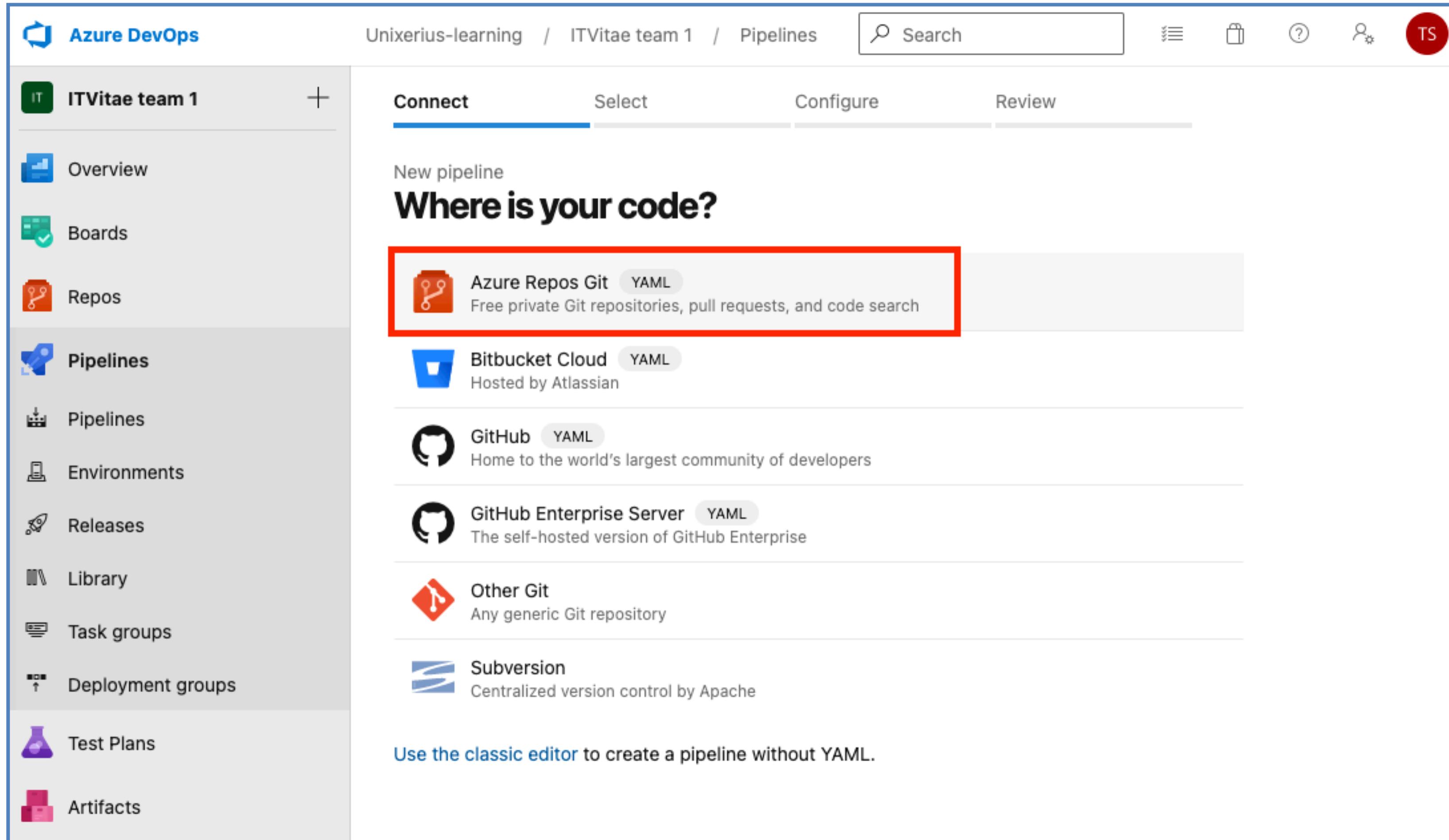
- A CICD pipeline which:
  - Logs in to Azure Container Registry.
  - Builds and pushes the JuiceShop image.
  - Logs out from ACR.
  - Orders Azure WebApp to deploy the image.

# Start with a "dummy"

- Go to Pipelines.
- Create a new pipeline.



# Start with a "dummy"



The screenshot shows the Azure DevOps interface for creating a new pipeline. The left sidebar is for the 'ITVitae team 1' project, with 'Pipelines' selected. The main area is titled 'Connect' and shows a list of providers:

- Azure Repos Git** (YAML) - Free private Git repositories, pull requests, and code search (highlighted with a red box)
- Bitbucket Cloud (YAML) - Hosted by Atlassian
- GitHub (YAML) - Home to the world's largest community of developers
- GitHub Enterprise Server (YAML) - The self-hosted version of GitHub Enterprise
- Other Git - Any generic Git repository
- Subversion - Centralized version control by Apache

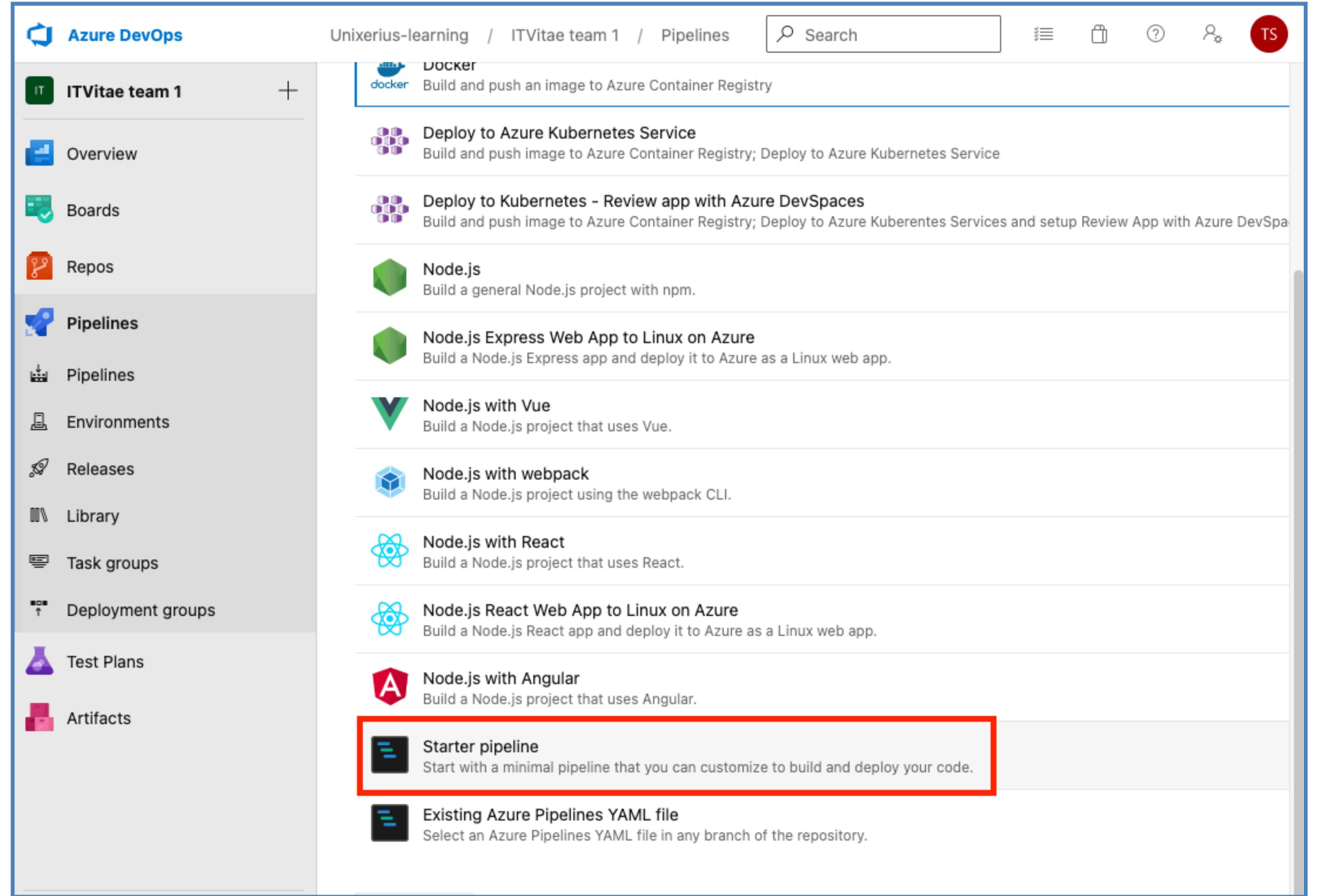
At the bottom, there is a note: [Use the classic editor](#) to create a pipeline without YAML.

# Start with a "dummy"

The screenshot shows the Azure DevOps interface for creating a new pipeline. The left sidebar is titled 'ITVitae team 1' and includes links for Overview, Boards, Repos, Pipelines, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The 'Pipelines' link is currently selected. The main area is titled 'Select a repository' and shows a list of repositories under 'ITVitae team 1'. One repository, 'ITVitae team 1 JuiceShop', is highlighted with a red border.

# Start with a "dummy"

- Choose type:
  - "Starter pipeline".



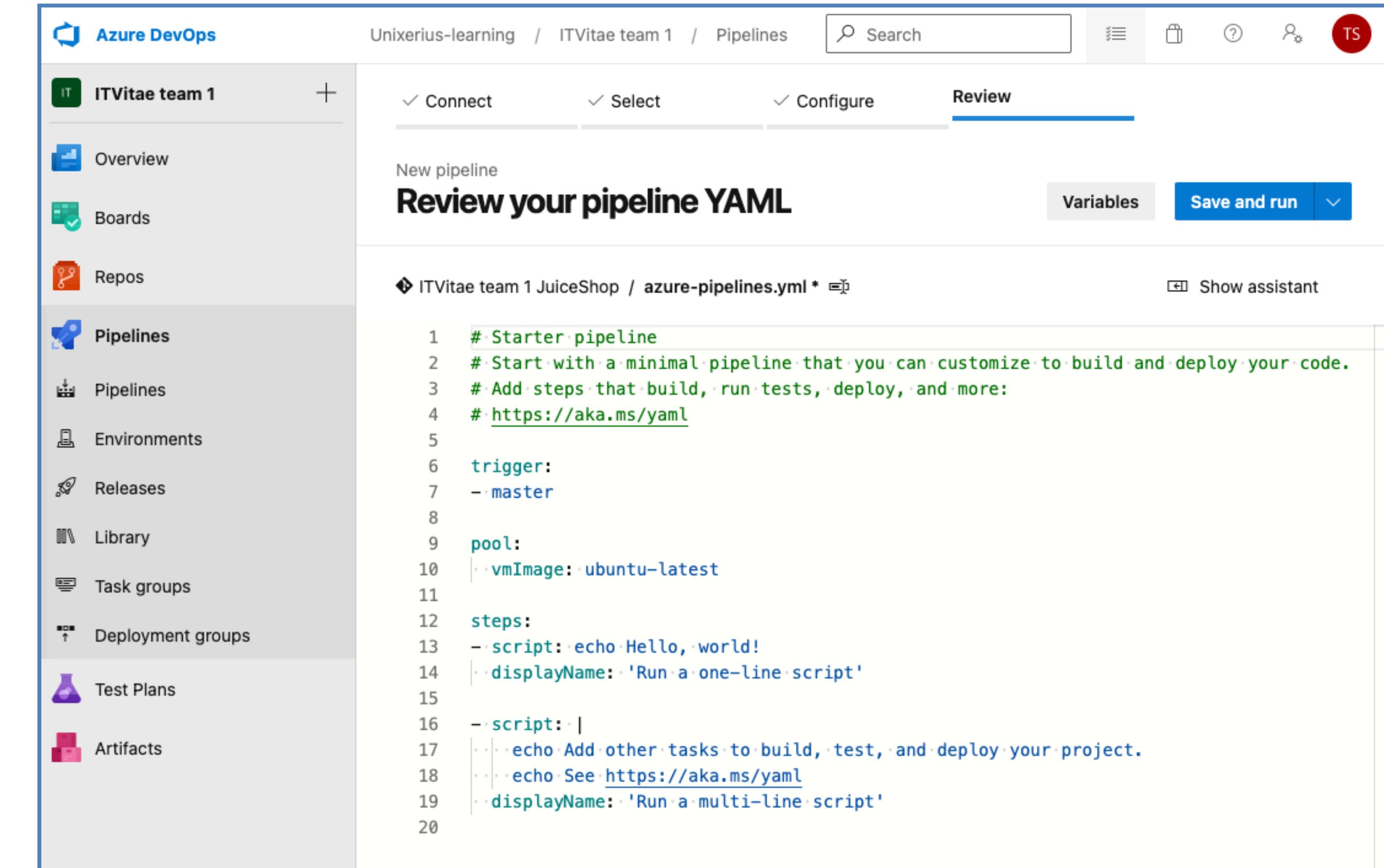
The screenshot shows the Azure DevOps Pipelines interface for the 'ITVitae team 1' project. The left sidebar lists navigation items: Overview, Boards, Repos, Pipelines (which is selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays a list of pipeline templates:

- Docker
- Deploy to Azure Kubernetes Service
- Deploy to Kubernetes - Review app with Azure DevSpaces
- Node.js
- Node.js Express Web App to Linux on Azure
- Node.js with Vue
- Node.js with webpack
- Node.js with React
- Node.js React Web App to Linux on Azure
- Node.js with Angular
- Starter pipeline** (highlighted with a red border)
- Existing Azure Pipelines YAML file

Each template entry includes a brief description of its purpose.

# Start with a "dummy"

- A dummy is made.
- Click "Save and run".



The screenshot shows the Azure DevOps Pipelines interface. The left sidebar is titled "ITVitae team 1" and includes links for Overview, Boards, Repos, Pipelines (which is selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area is titled "Review your pipeline YAML" and shows the following YAML code:

```
# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- master

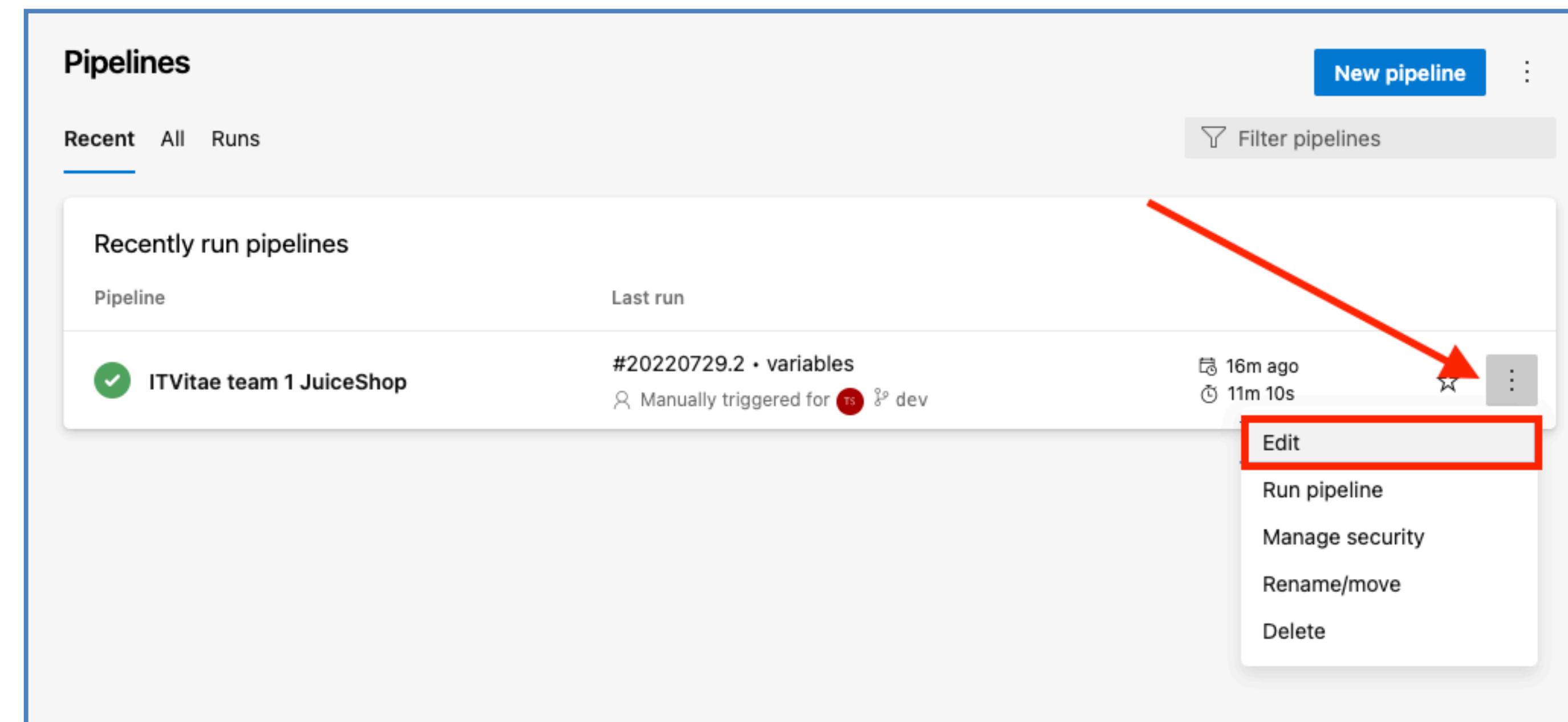
pool:
  vmImage: ubuntu-latest

steps:
- script: echo Hello, world!
  displayName: 'Run a one-line script'

- script: |
    echo Add other tasks to build, test, and deploy your project.
    echo See https://aka.ms/yaml
  displayName: 'Run a multi-line script'
```

# Building our pipeline

- Go back to the **Pipelines** tab.
- Edit the pipeline and choose the "dev" branch!



# Building our pipeline

← ITVitae team 1 JuiceShop

master ▾ ◆ ITVitae team 1 JuiceShop / azure-pipelines.yml

Filter branches

✓ master Default

Mine

dev ←

prod

test

All

develop

feat/ip  
pool:  
15   | vmImage: ubuntu-latest  
16  
17  
18

```
15 pool:  
16   vmImage: ubuntu-latest  
17  
18
```

# Cleanup ...

- Change the "trigger", to "*manual*".
- We can remove the dummy lines under "steps:"

```
1 trigger:  
2   - manual  
3  
4 pool:  
5   · vmImage: · ubuntu-latest  
6  
7 steps:  
8   |
```

# Docker login / logout

- Pull up the wizard, which has all plugins.



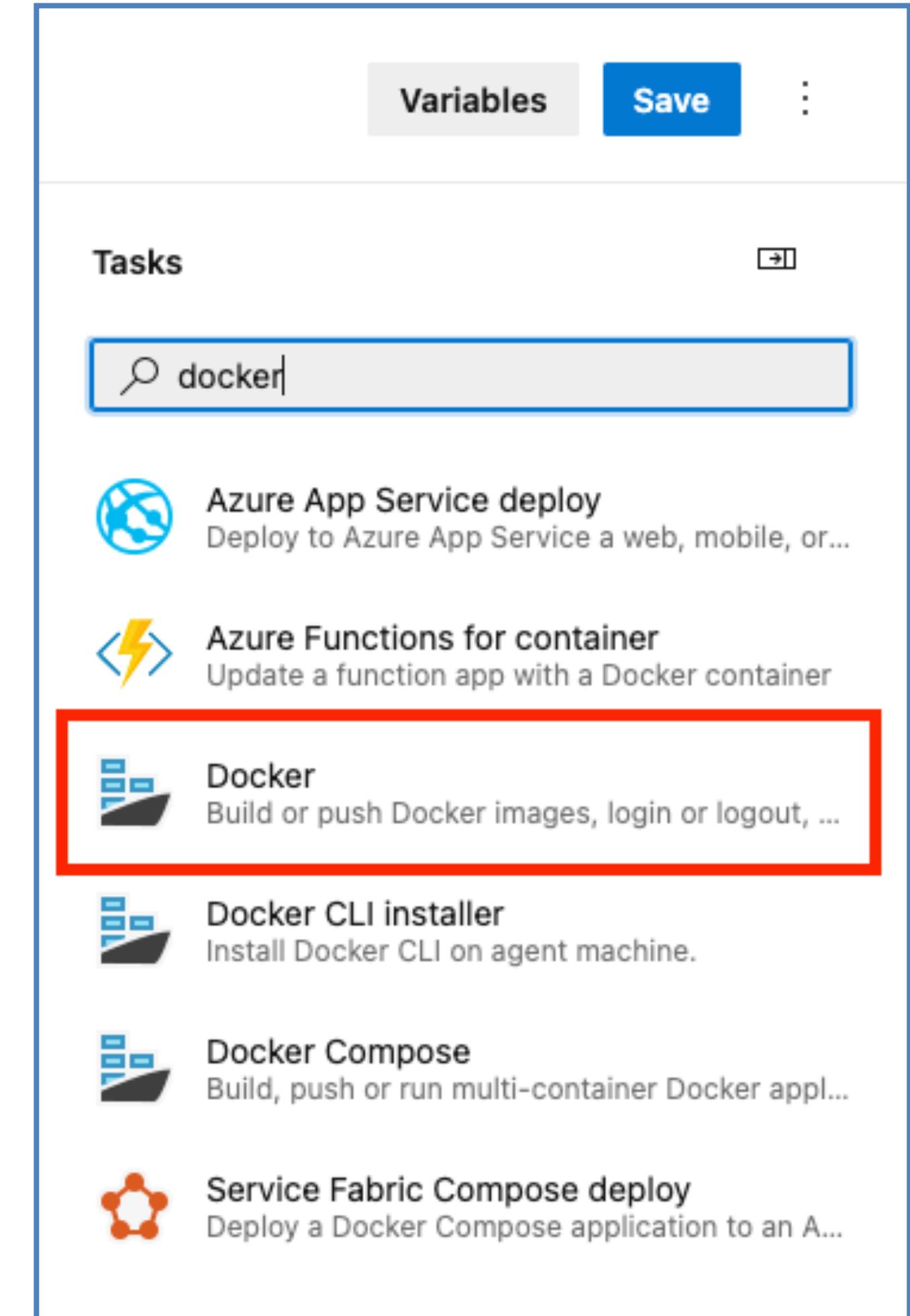
The screenshot shows the Azure Pipelines interface for a pipeline named "ITVitae team 1 JuiceShop (2)". The pipeline is currently set to the "dev" branch. The configuration file is "ITVitae team 1 JuiceShop / azure-pipelines-2.yml". The code editor displays the following YAML configuration:

```
1 trigger:
2 - manual
3
4 pool:
5   vmImage: ubuntu-latest
6
7 steps:
8   - [empty step]
```

In the top right corner, there are buttons for "Variables", "Save", and a more options menu. A red box highlights the "Show assistant" button, which is located next to the save button. The status bar at the bottom right shows "100% 0s ago".

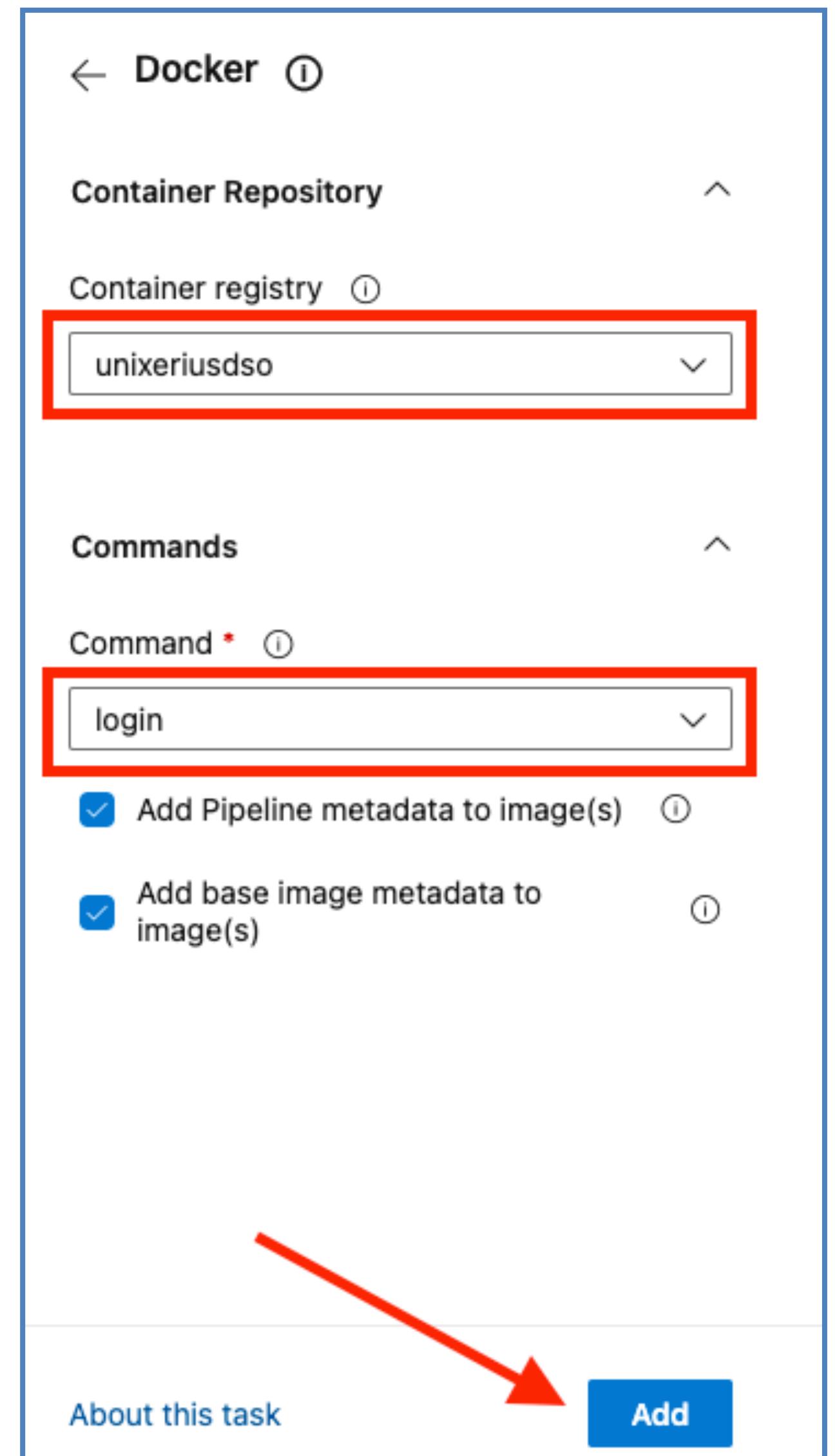
# Docker login / logout

- Search for Docker,
  - Grab the one for login/logout



# Docker login / logout

- Choose "*Login*" as command.
- Choose the "*unixeriusdso*" registry.
- Repeat for "*Logout*".
  - Example on next slide.



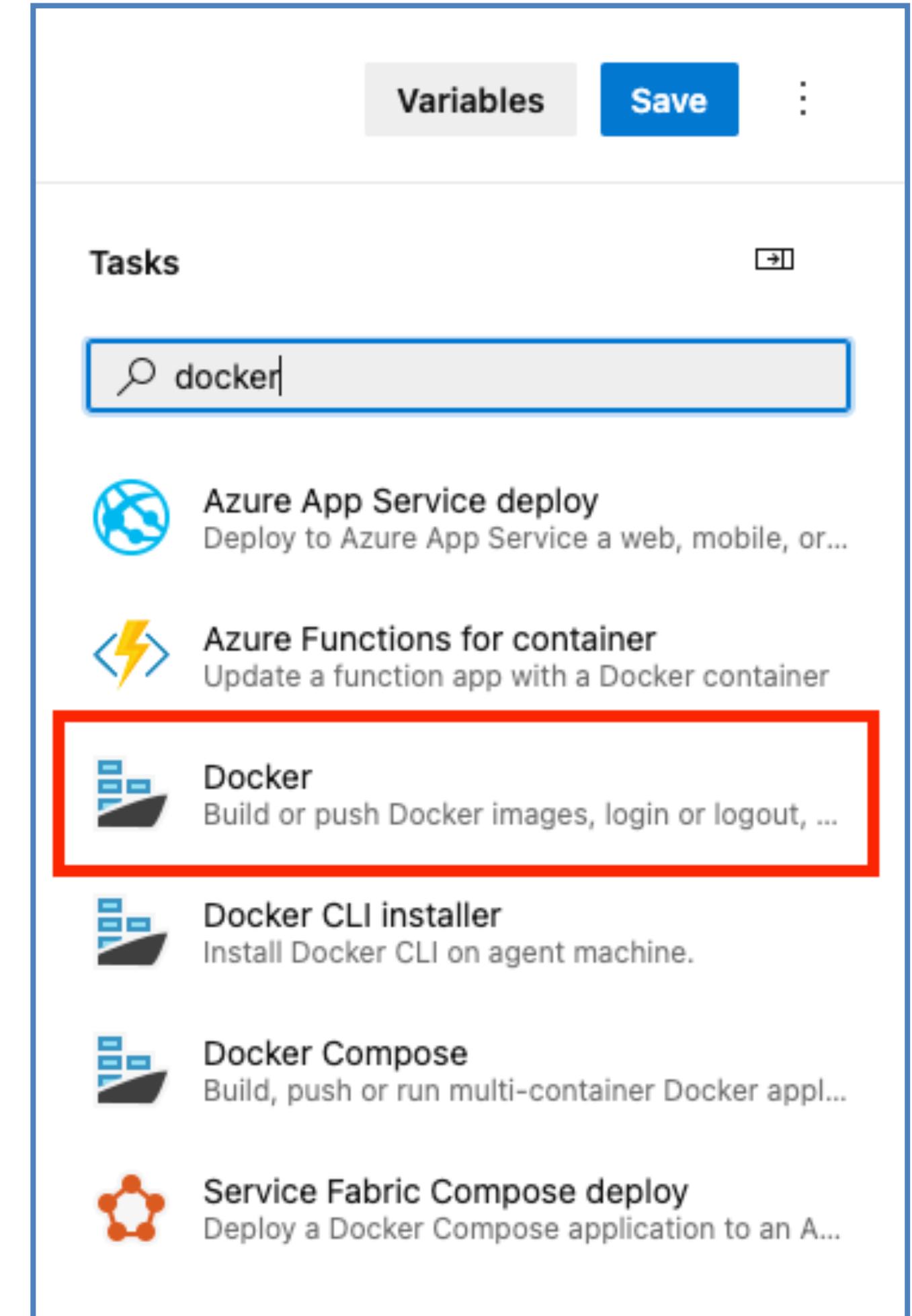
# Docker login / logout

- Your code should be ->
- Validate and save.
  - Then run on Dev branch.

```
1  trigger:  
2    - manual  
3  
4  pool:  
5    | - vmImage: ubuntu-latest  
6  
7  steps:  
8    Settings  
9    - task: Docker@2  
10   | - inputs:  
11   |   | - containerRegistry: 'unixeriusdso'  
12   |   | - command: 'login'  
13   Settings  
14   - task: Docker@2  
15   | - inputs:  
16   |   | - containerRegistry: 'unixeriusdso'  
17   |   | - command: 'logout'
```

# Docker build and push

- If the pipeline ran OK,
  - It's time to build the image.
- Go to the editor again,
  - Make sure you're on "dev"!
  - Open the wizard again.
  - Select the same Docker task.



# Docker build and push

- Chose "*BuildAndPush*" as command.
- Chose the "*unixeriusdso*" registry.
- Set repository to "team1",
  - Adjust for your team!
- Set tags to "dev",
- Click "add".

The screenshot shows a configuration interface for a Docker build. The fields are as follows:

- Container Repository**:
  - Container registry**: UnixeriusDSO
  - Container repository**: team1
- Commands**:
  - Command \***: buildAndPush
- Dockerfile \***: \*\*/Dockerfile
- Build context**: \*\*
- Tags**: dev
- Add Pipeline metadata to image(s)

# Docker build + push

- Your code should be ->
- Save and run.
  - Building takes 8-10 mins.

```
1 trigger:  
2   - manual  
3  
4 pool:  
5   |-- vmImage: ubuntu-latest  
6  
7 steps:  
8   Settings  
9   |-- task: Docker@2  
10  |-- inputs:  
11  |   |-- containerRegistry: 'UnixeriusDS0'  
12  |   |-- command: 'login'  
13  |   Settings  
14  |-- task: Docker@2  
15  |-- inputs:  
16  |   |-- containerRegistry: 'UnixeriusDS0'  
17  |   |-- repository: 'team3'  
18  |   |-- command: 'buildAndPush'  
19  |   |-- Dockerfile: '**/Dockerfile'  
20  |   |-- tags: 'dev'  
21  |   Settings  
22  |-- task: Docker@2  
23  |-- inputs:  
24  |   |-- containerRegistry: 'UnixeriusDS0'  
25  |   |-- command: 'logout'
```

# Run on Azure WebApp

- Your team already has a WebApp!
  - Adjust for the right team name.

<http://unixeriusdso-team1.azurewebsites.net/>

# Run on Azure WebApp

- Go back to the pipeline editor.
  - Make sure you're on the "dev" branch!
- At the bottom, we will add a block of code.
  - This tells Azure WebApp to load your container.

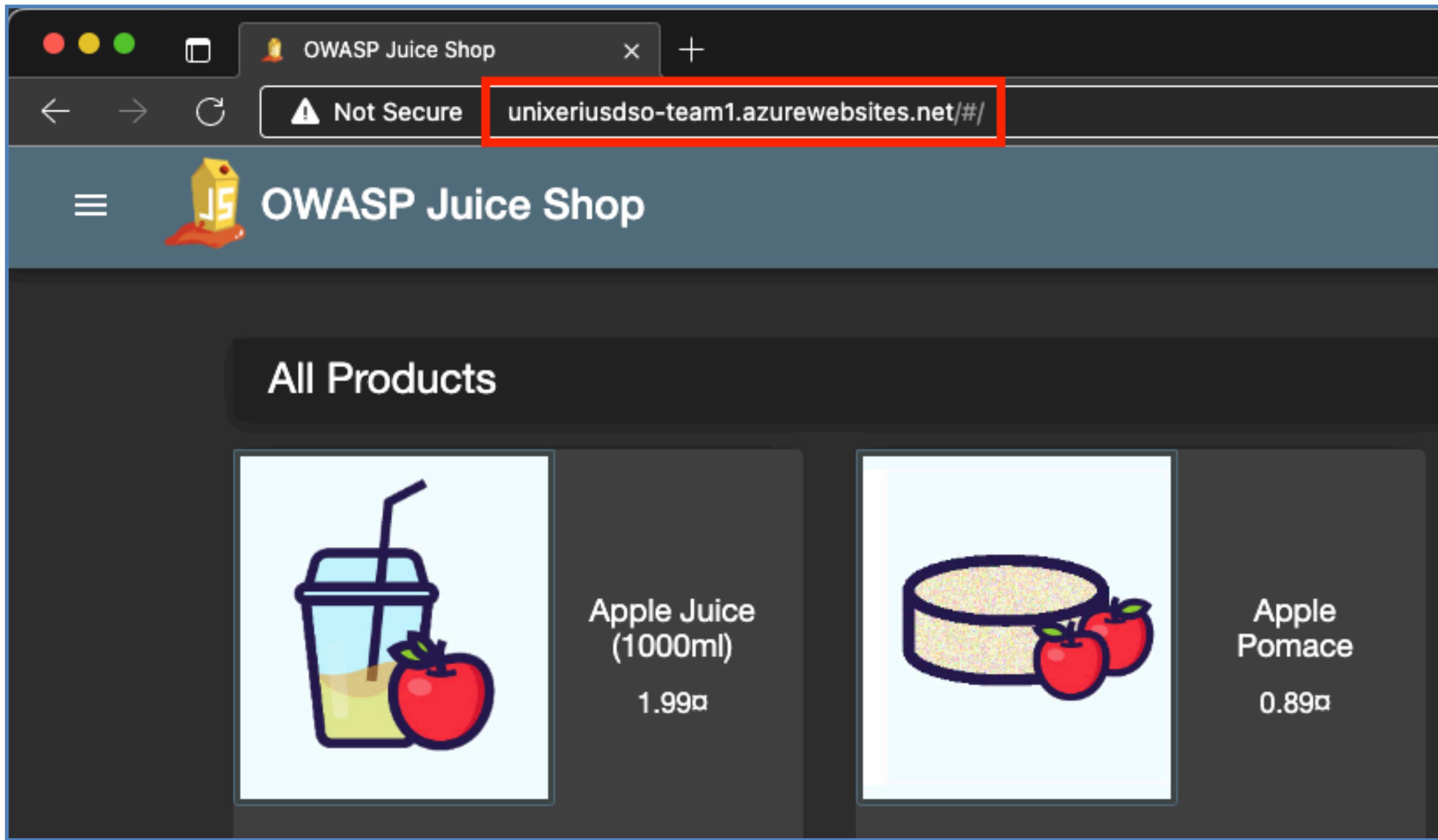
# The code

- Is also available as "[pipeline-step1-build-run.yml](#)".
  - The dashes shown below are a single dash!
  - ```
task: AzureWebAppContainer@1
inputs:
    azureSubscription: 'Azure Unixerius Learning'
    appName: 'unixeriusdso-team1'
    containers: 'unixeriusdso.azurecr.io/team1:dev'
    appSettings: '-Port 3000'
    configurationStrings: '-acrUseManagedIdentityCreds true'
```

# Deployment to Dev!

- If you run the pipeline again, it will:
  - Build and push your image to ACR.
  - Run your container as WebApp.
- After deployment, it will take a few minutes.
  - Our WebApp instances are a bit slow. 

# Deployment to Dev!



# Checkpoint!

- Does everyone have:
  - A pipeline on the "dev" branch.
  - Which builds and pushes to ACR?
  - Which runs the container?
- Have you tested this?



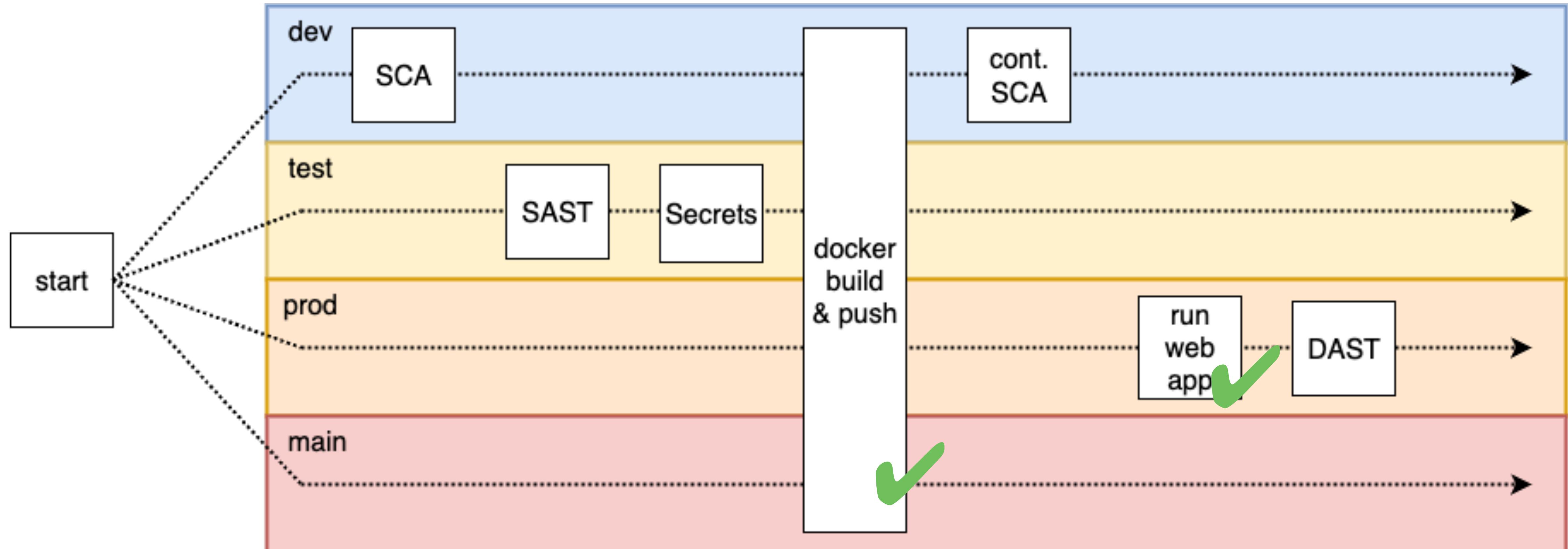
SmartSign.com • 800-952-1457 • K2-4293

# 10. Lab: Quick cleanup

# I have a spoiler file.

- Available as "[pipeline-step1-build-run.yml](#)".
  - It prepares you for "stages" and more.
- Use it for your "dev" pipeline.
  - Replace the "team1" with your team.

# Our final pipeline goal



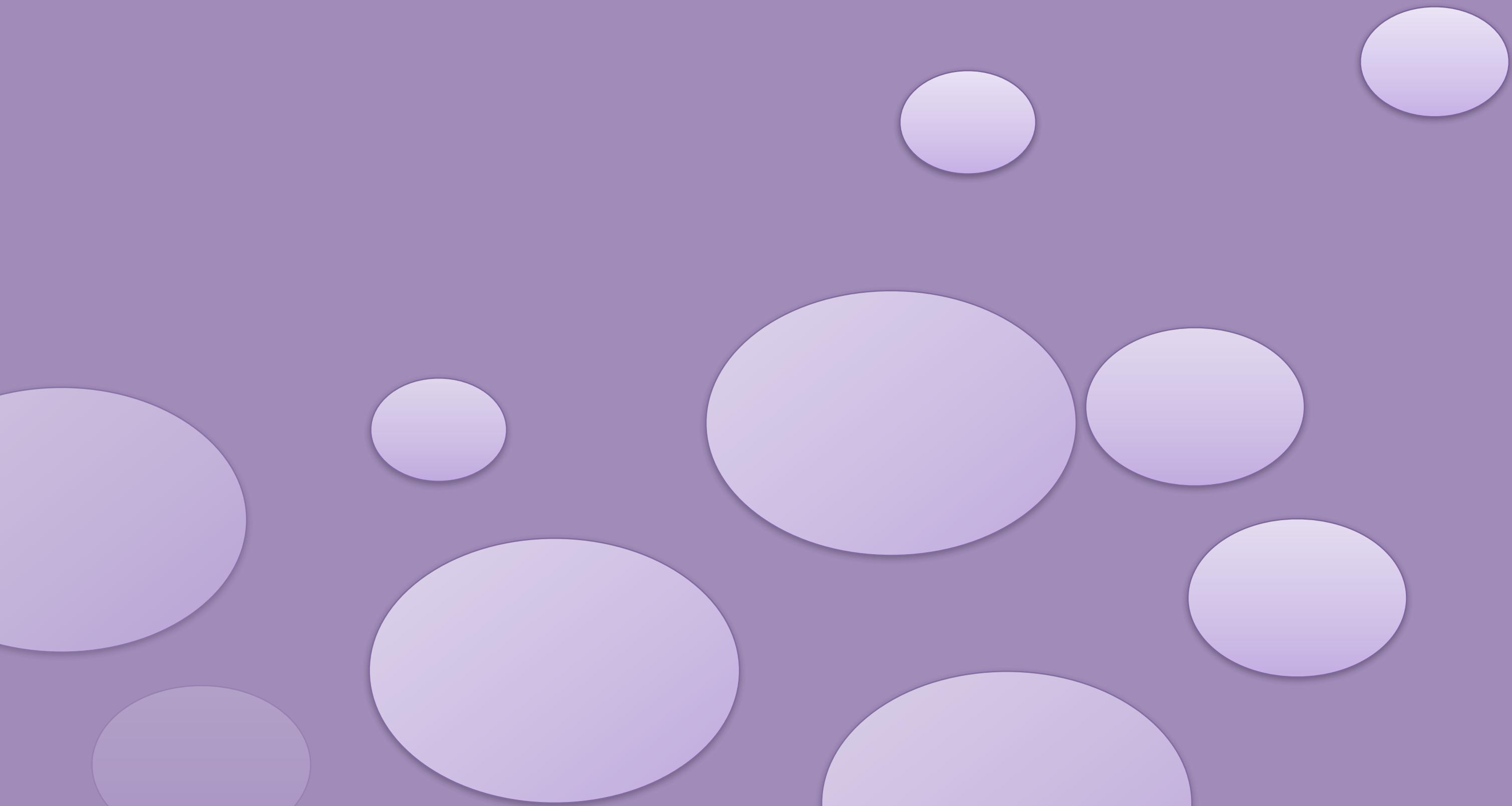
# Checkpoint!

- Does everyone have:
  - My spoilers as their pipeline?
  - Does it still build and deploy?
- Have you tested this?



SmartSign.com • 800-952-1457 • K2-4293

# Closing



# What have we achieved?

- We practiced with Git.
- We ran JuiceShop in Docker, locally.
- We made our first pipeline.
- We ran JuiceShop on Azure, via CICD.

# Tomorrow

- We will automate functional testing.
- We will look into software vulnerabilities,
  - DevSecOps fundamental concepts, and
  - Threat modelling.

# Relevant reading

| Topic                                 | Book  |
|---------------------------------------|-------|
| Vulnerability management              | Ch 6  |
| Threat assessments                    | Ch 8  |
| Agile software (and security) testing | Ch 11 |
|                                       |       |
|                                       |       |

# Reference materials

# Resources

- [Understanding the TCO of "serverless"](#)
- [ArgoCD for beginners](#)
- In-depth: [So you think you know Git?](#)
- In-depth: [So you think you know Git? Part 2](#)