

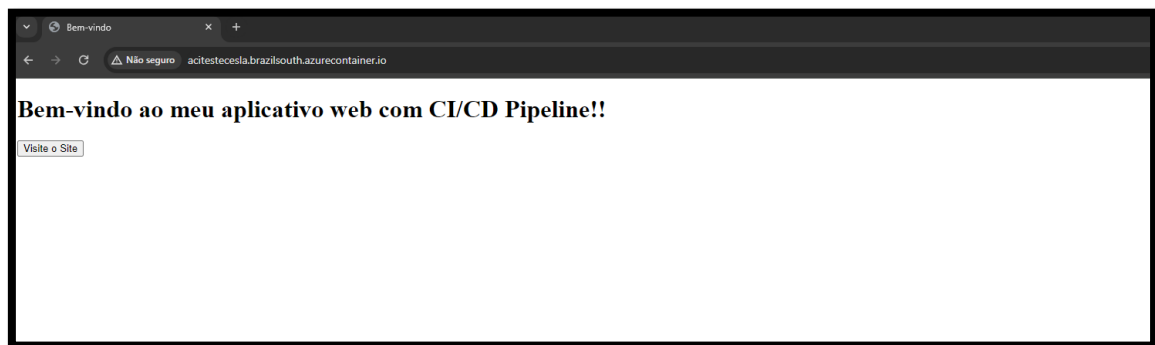
Teste Prático – Cesla

Neste projeto, o código a ser implantado consiste em uma página web simples que exibe uma mensagem de boas-vindas e possui um botão para redirecionamento à URL (<https://cesla.ind.br>), configurada previamente nas variáveis de ambiente. O projeto utiliza integração contínua e entrega contínua por meio das pipelines configuradas no Azure DevOps, e o ambiente de execução está no Azure Container Instance.

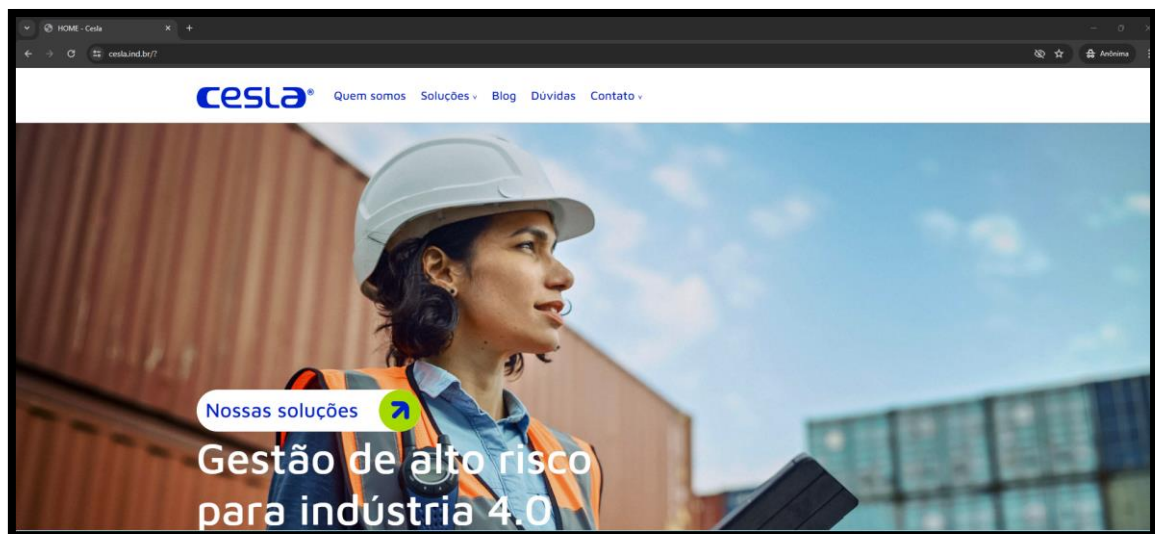
DNS publicado através do Azure Container Instance:

<http://acitestecesla.brazilsouth.azurecontainer.io>

Sendo o seguinte resultado:

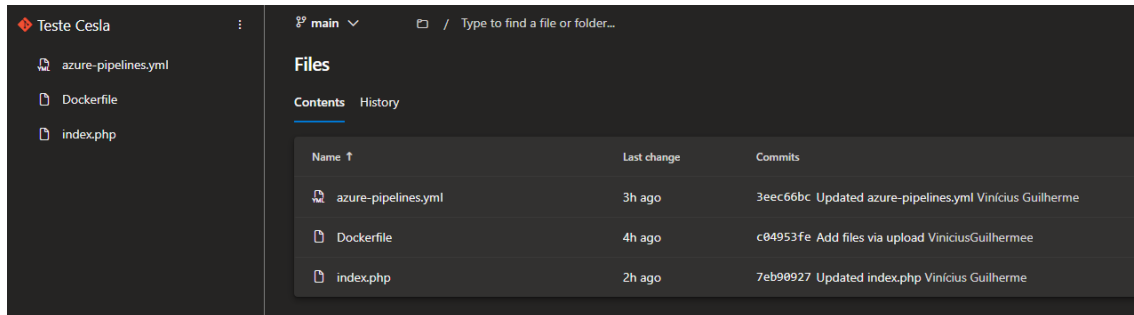


Resultado do redirecionamento do botão:



1. Repositório no Azure DevOps – Código fonte (PHP) + Dockerfile + azure-pipelines.yml

Importado código fonte (index.php) e Dockerfile no Azure Repos através do repositório do GitHub.



Código fonte (PHP):

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bem-vindo</title>
</head>
<body>
  <h1>Bem-vindo ao meu aplicativo web com CI/CD Pipeline!</h1>
  <?php
    // Obtendo a variável de ambiente
    $url = getenv('url');
  ?>
  <!--
  - Botão redireciona para a URL inserida na variavel de ambiente definida em Library -
  -->
  <form method="get" action="<?php echo $url; ?>">
    <button type="submit">Visite o Site</button>
  </form>
</body>
</html>
```

Dockerfile do código (PHP):

```
# Use a imagem base oficial do PHP com Apache
FROM php:7.4-apache

# Copie os arquivos do aplicativo para o diretório padrão do Apache
COPY . /var/www/html/

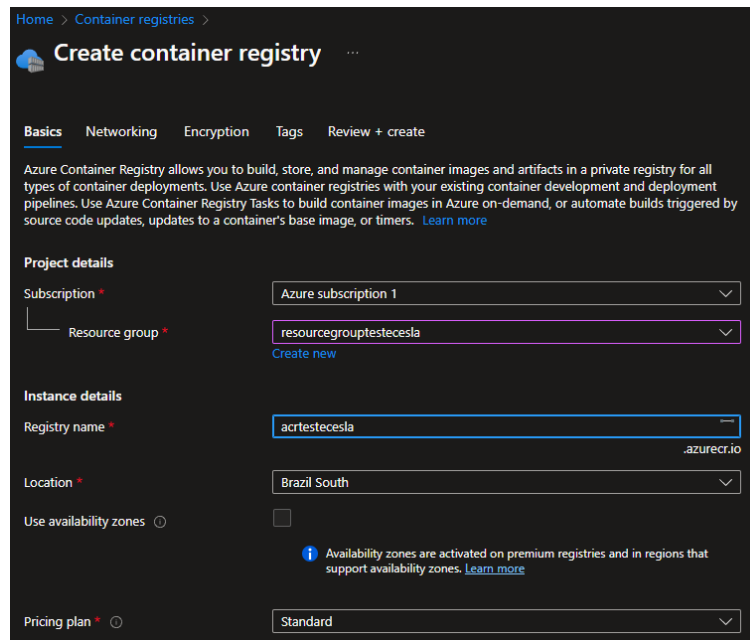
# Exponha a porta 80 para o tráfego HTTP
EXPOSE 80
```

Repositorio Git:

<https://github.com/ViniciusGuilhermee/testecesla.git>

2. Criação Azure Container Registry

Após a importação do repositório, realizado criação do ACR (***acrtestecsla***), atribuindo a um novo Resource Group (***resourcegrouptestecsla***).



Home > Container registries >

Create container registry

Basics Networking Encryption Tags Review + create

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. [Learn more](#)

Project details

Subscription * Azure subscription 1

Resource group * resourcegrouptestecsla [Create new](#)

Instance details

Registry name * acrtestecsla .azurecr.io

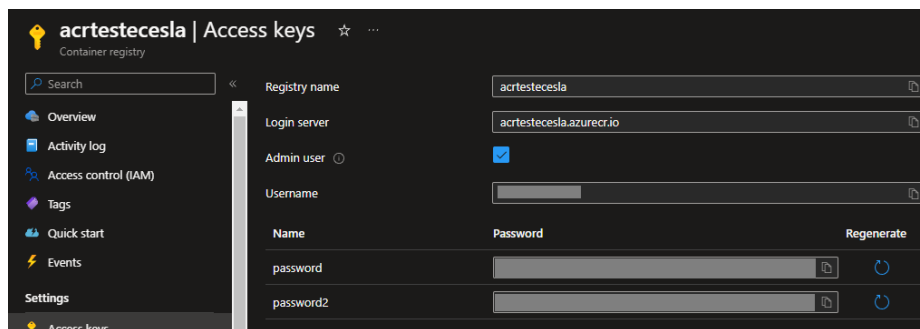
Location * Brazil South

Use availability zones ☐

Availability zones are activated on premium registries and in regions that support availability zones. [Learn more](#)

Pricing plan * Standard

Posteriormente habilitado login em “admin user”:



acrtestecsla | Access keys

Container registry

Search

Overview Activity log Access control (IAM) Tags Quick start Events Settings Access keys

Registry name acrtestecsla

Login server acrtestecsla.azurecr.io

Admin user ☒

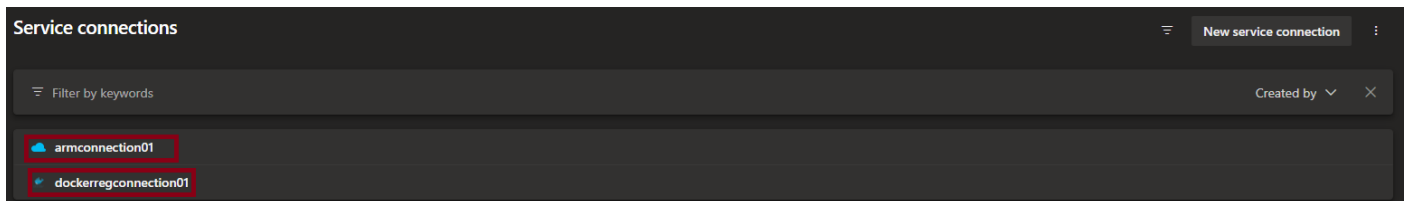
Username

Name	Password	Regenerate
password		
password2		

Username e password ocultos do anexo por questões segurança

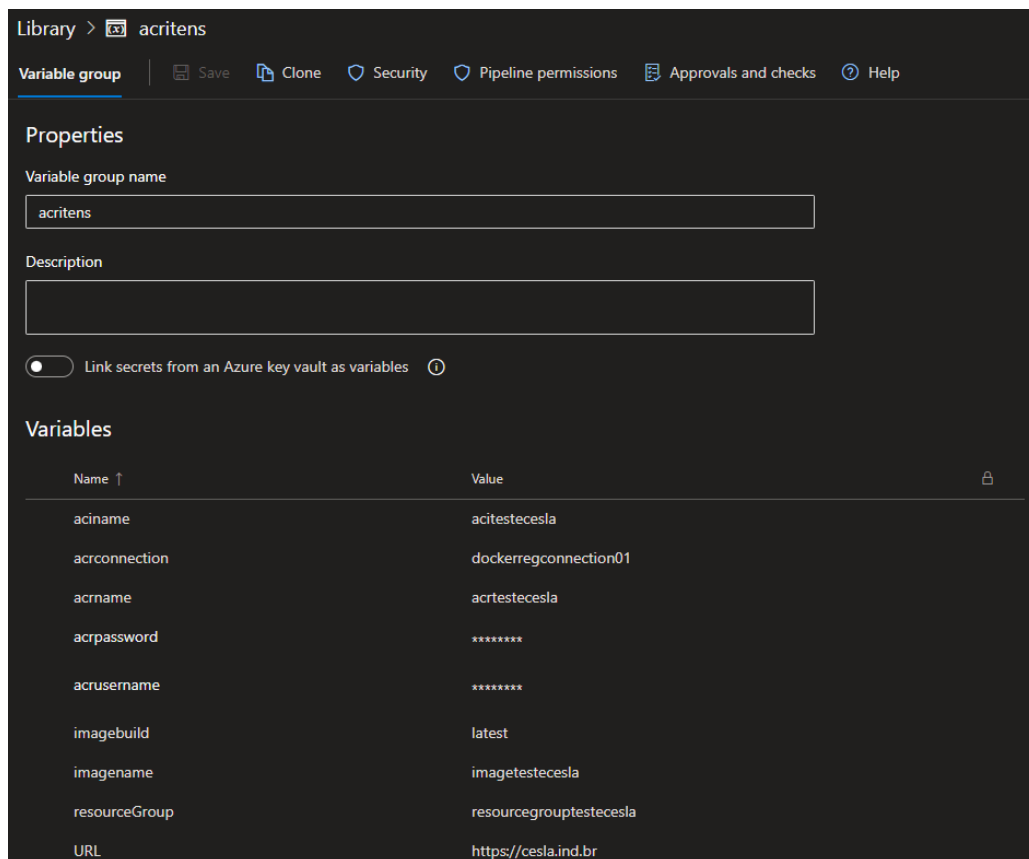
3. Service connections

Criado dois registros de conexão: autenticação aos recursos da azure (**armconnection**) e ao registro docker (**dockerregconnection01**) para ser possível acessar os recursos e realizar o envio ao container (ACR).



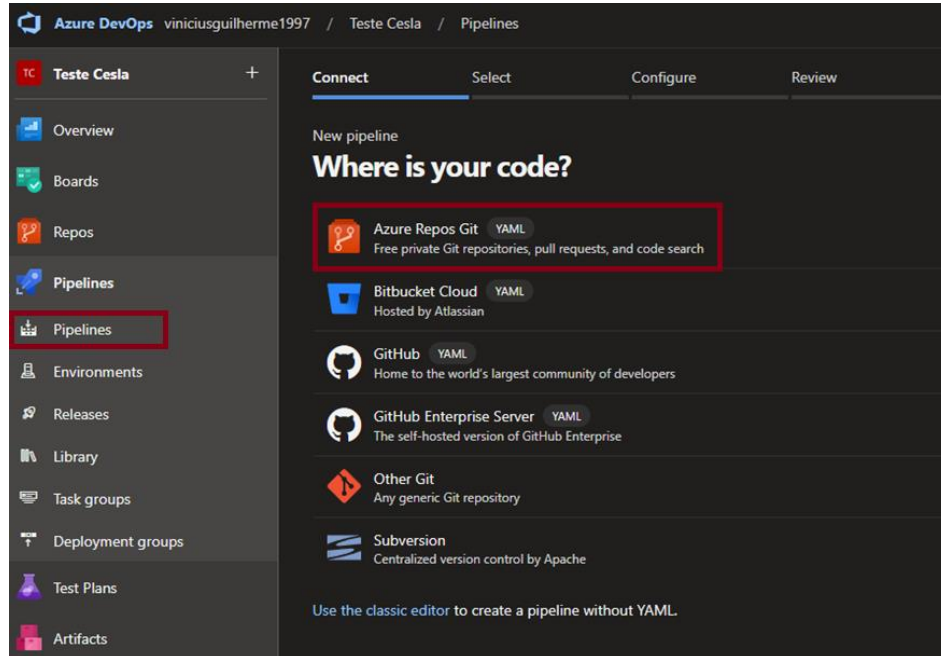
4. Library (Variable groups):

Criado um grupo de variáveis (**acritens**), sendo variáveis comumente utilizadas no BUILD, PUSH e RELEASE. Como o conteúdo das variáveis de login e password do ACR são sensíveis, o conteúdo das mesmas foram ocultadas por razões de segurança.

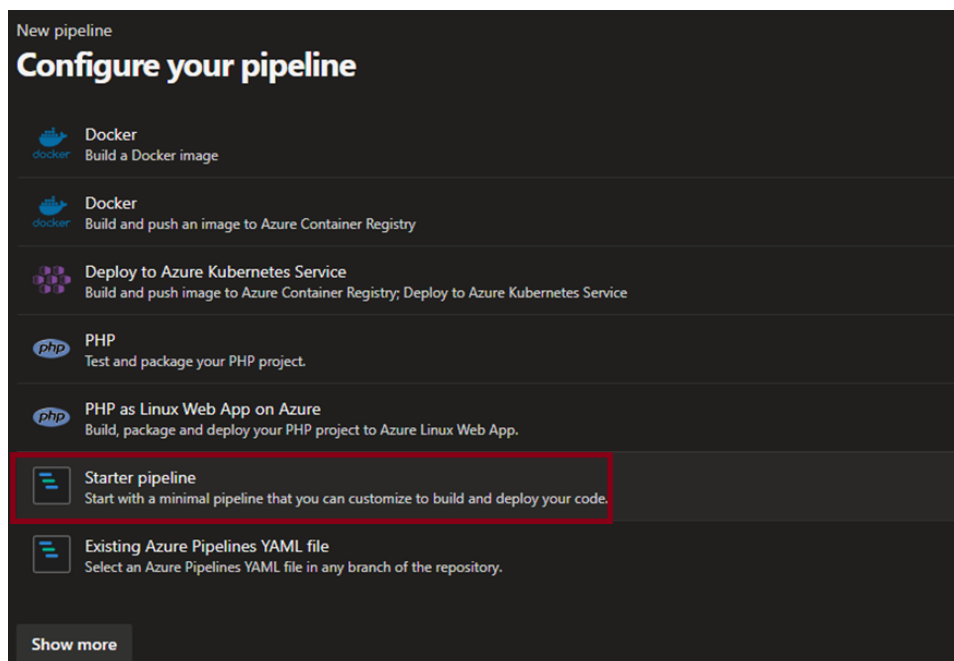


5. Pipeline (Checkout&Build&Push)

Após todas as etapas anteriores, criado a pipeline em YAML para realizar o checkout, build e push do código.



Selecionado “starter pipeline” para inserir o script YAML previamente definido:



Pipeline Yaml:

No YAML, definido trigger a partir do repositório central, toda e qualquer modificação no repositório, a pipeline é automaticamente iniciada (Integração Contínua – CI). O código é carregado em uma VM Ubuntu (última versão disponível) para compilação.

Os passos são: checkout do código-fonte, login no ACR (para possibilitar o push da imagem), build da imagem e, por fim, armazenamento da imagem no ACR. Todos os steps possuem um "display name" para que, durante a execução da pipeline, a visibilidade de cada um fique organizada e clara.

```
# INTEGRAÇÃO CONTÍNUA
trigger:
- main

pool:
  vmImage: 'ubuntu-latest'

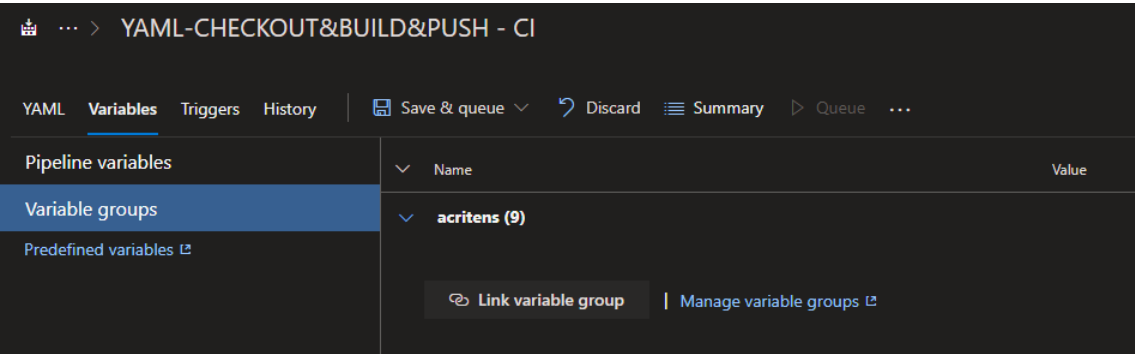
# CHECKOUT DO CÓDIGO
steps:
- checkout: self
  displayName: 'Checkout do código'

# LOGIN NO AZURE CONTAINER REGISTRY
- task: Docker@2
  inputs:
    containerRegistry: '$(acrconnection)'
    command: 'login'
    username: '$(acrusername)'
    password: '$(acrpassword)'
  displayName: 'Login Azure Container Registry'

# BUILD DA IMAGEM
- task: Docker@2
  inputs:
    command: build
    dockerfile: '$(Build.SourcesDirectory)/Dockerfile'
    containerRegistry: '$(acrconnection)'
    repository: '$(imagename)'
    tags: |
      $(imagebuild)
  displayName: 'Build'

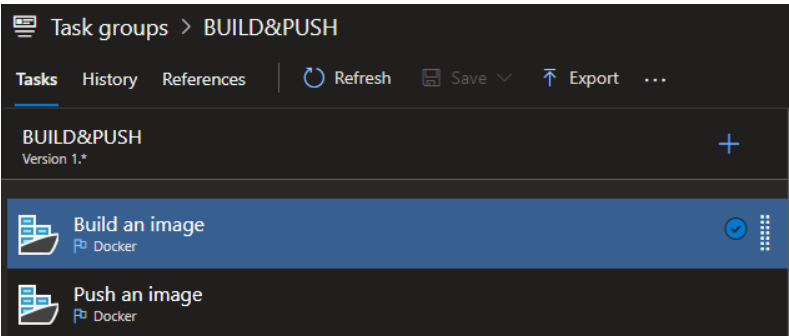
# PUSH DA IMAGEM NO AZURE CONTAINER REGISTRY
- task: Docker@2
  inputs:
    command: push
    containerRegistry: '$(acrconnection)'
    repository: '$(imagename)'
    tags: |
      $(imagebuild)
  displayName: 'Push Azure Container Registry'
```

Habilitado a utilização da library na pipeline, para referenciar corretamente as variaveis no YAML:

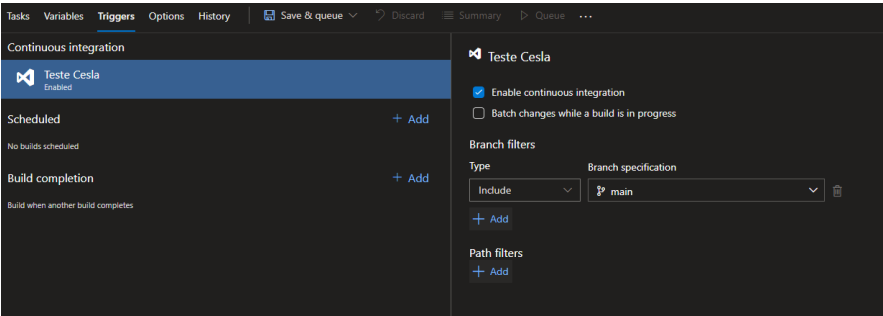


6. Task Group (BUILD&PUSH)

Neste projeto, não foram utilizados task groups, pois a pipeline de checkout&build&push estão configurados unicamente em YAML. No entanto, foi criada uma task group para demonstrar a possibilidade de sua utilização.



Habilitado integração continua (CI) em triggers:



Resultado pipeline TASK GROUP:

← Jobs in run #20240604.1 TASK GROUP - BUILD&PUSH			✓ Agent job 1
Jobs			1 Pool: Azure Pipelines
✓	Agent job 1	49s	2 Image: ubuntu-latest
✓	Initialize job	1s	3 Agent: Hosted Agent
✓	Checkout Teste Cesla@main to s	2s	4 Started: Just now
✓	Build an image	14s	5 Duration: 49s
✓	Push an image	29s	6
✓	Post-job: Checkout Teste Cesla@...	<1s	7 ▶ Job preparation parameters
✓	Finalize Job	<1s	37 ▶ <code>fx</code> 1 queue time variable used
✓	Report build status	<1s	40 Job live console data:
			41 Finishing: Agent job 1

Desabilitado pipeline (**TASK GROUP**) para não gerar duplicidade com a pipeline em YAML:

TASK GROUP - BUILD&PUSH

TasksVariablesTriggersOptionsHistorySave & queueDiscardSummaryQueue

Build properties
Define general build pipeline settings

Description

Build number format ⓘ
\$(date:yyyyMMdd)\$(rev:.r)

The new build request is processing

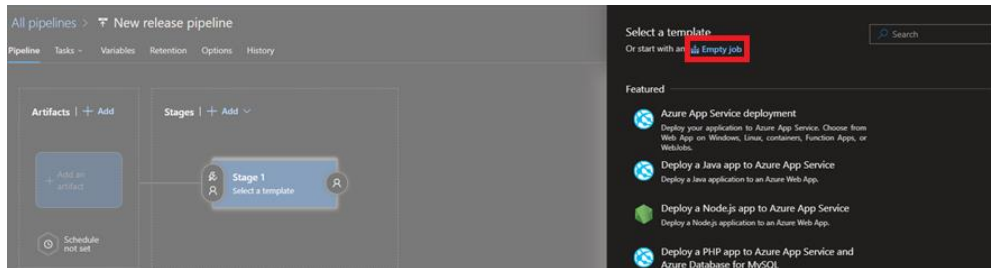
☐ Enabled - queue and start builds when eligible agent(s) available

☐ Paused - queue new builds but do not start

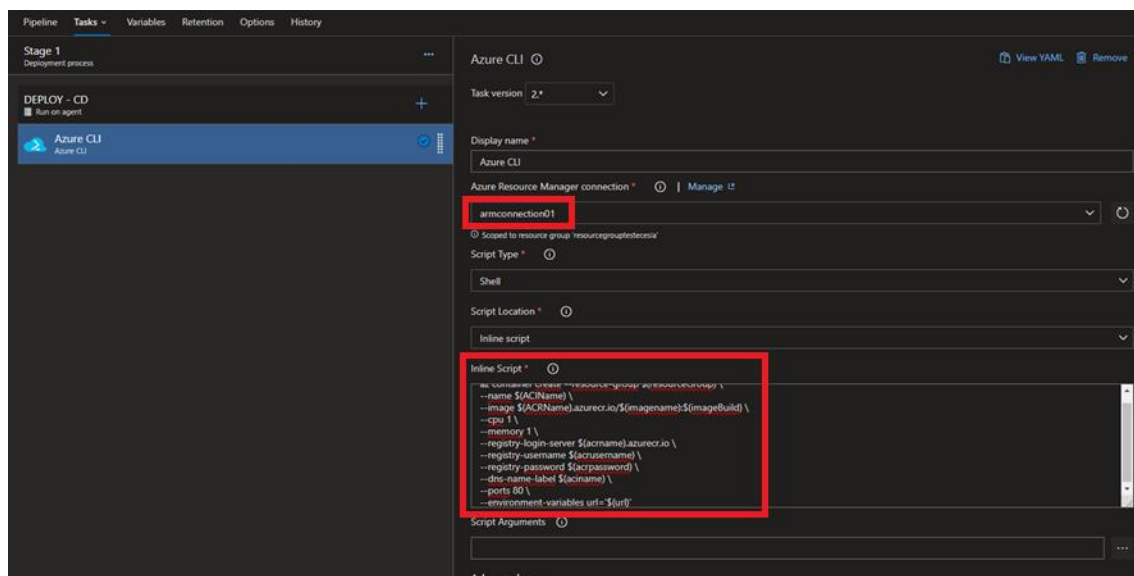
☒ Disabled - do not queue new builds

7. Release

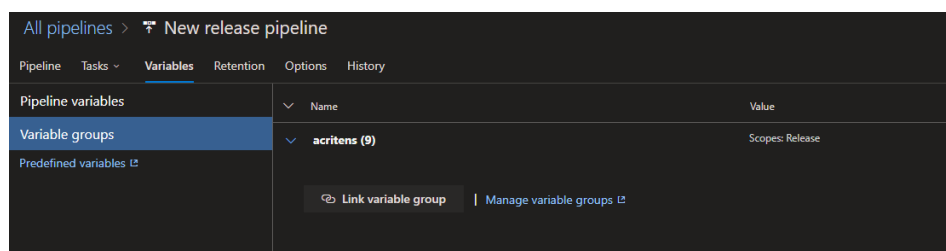
Por fim, para realizar o deploy da aplicação, é necessário configurar um job, que neste caso, foi utilizado “*Empty job*”.



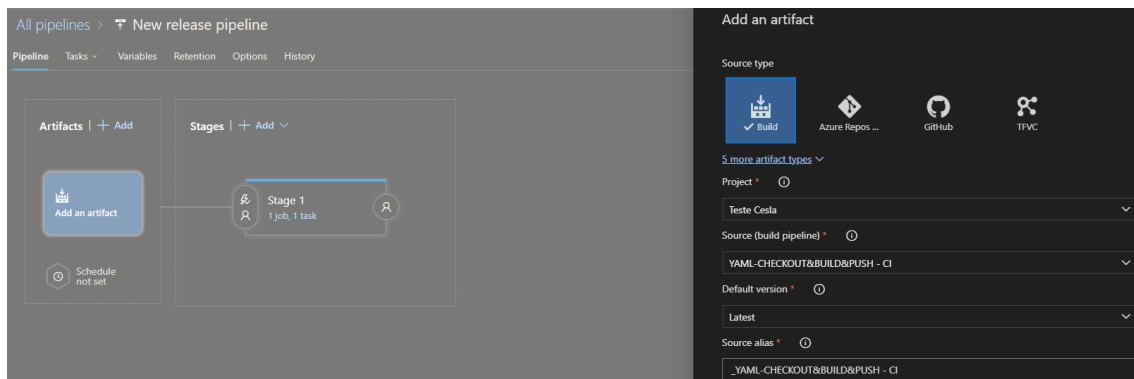
No agente, adicionado execução de script com “*AZURE CLI*”, selecionado a conexão de gerenciamento de recursos previamente configurado (***armconnection01***) e com o conteúdo, para criar a instancia e publica-la no Azure Container Instance:



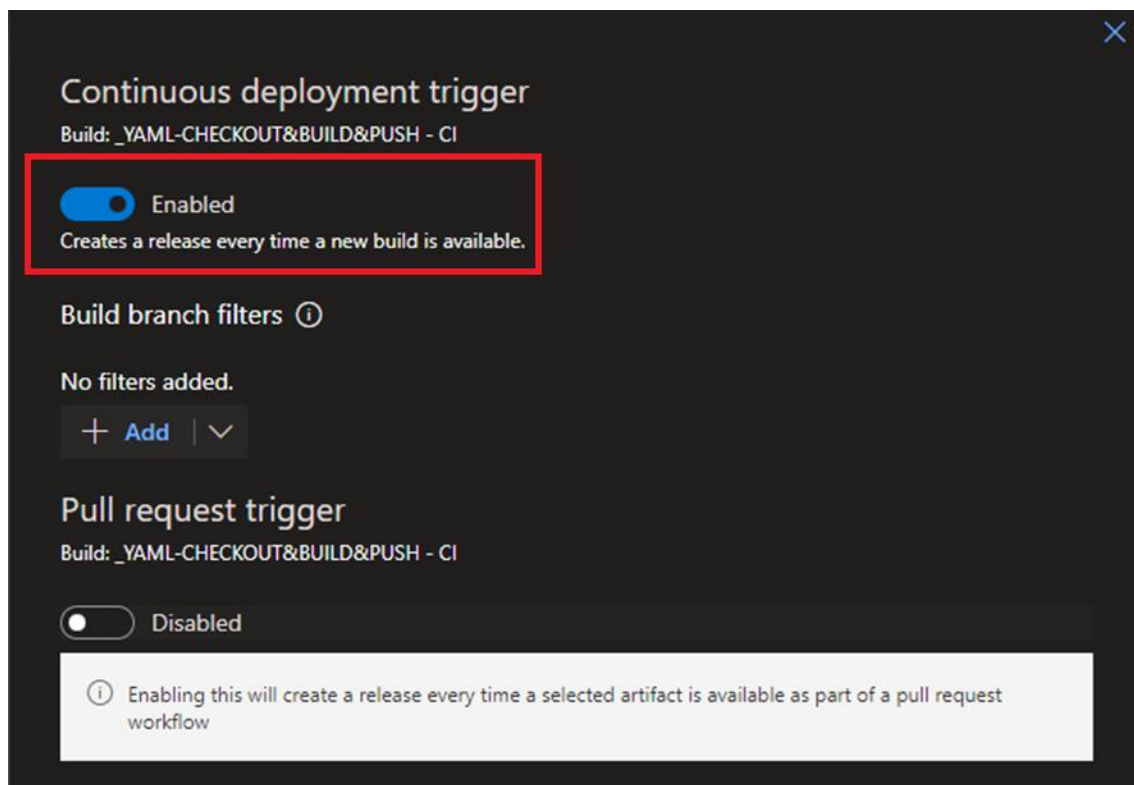
Como o script utiliza as mesmas variáveis das etapas anteriores da pipeline, foi habilitado a utilização da library ***acrutils***:



Configurado a captura do artefato gerado pelo build da pipeline anterior:

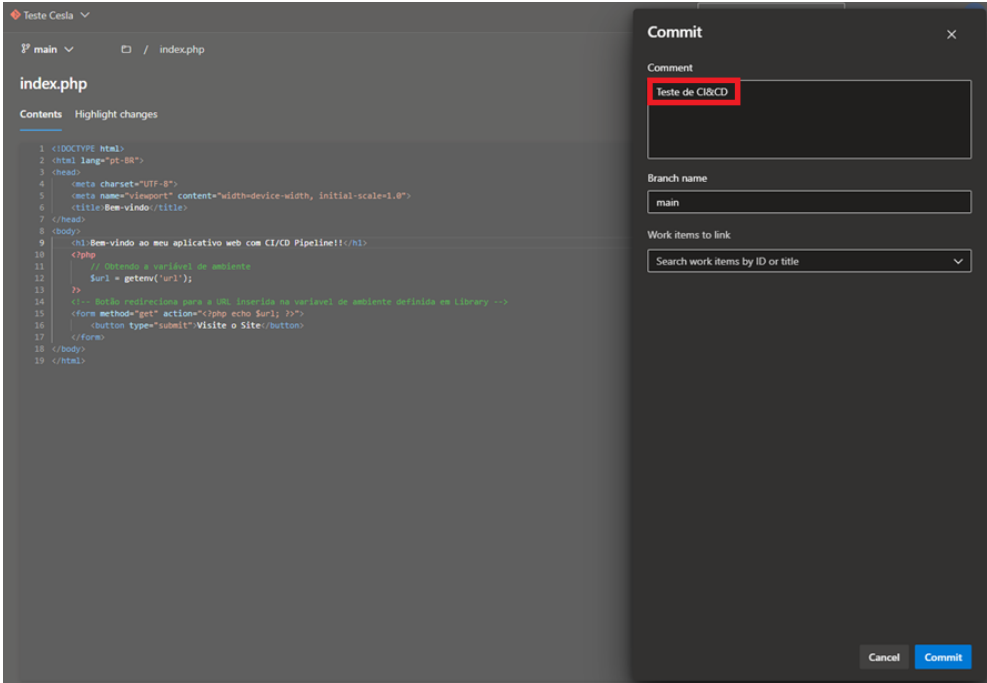


Habilitado entrega continua (CD) toda vez que um build é gerado:



8. Testes

Modificado código-fonte do repositório para certificar que todas as etapas anteriores foram configuradas com sucesso, e que o CI&CD foi implementado.



Validação da pipeline de Checkout, Build e Push em YAML com integração continua (CI):

← Jobs in run #20240604.8

YAML-CHECKOUT&BUILD&PUSH - CI

Jobs

Job29s

Initialize job3s

Checkout do código2s

Login Azure Container Registry<1s

Build14s

Push Azure Container Registry7s

Post-job: Checkout do código<1s

Finalize Job<1s

Report build status<1s

✓ Finalize Job

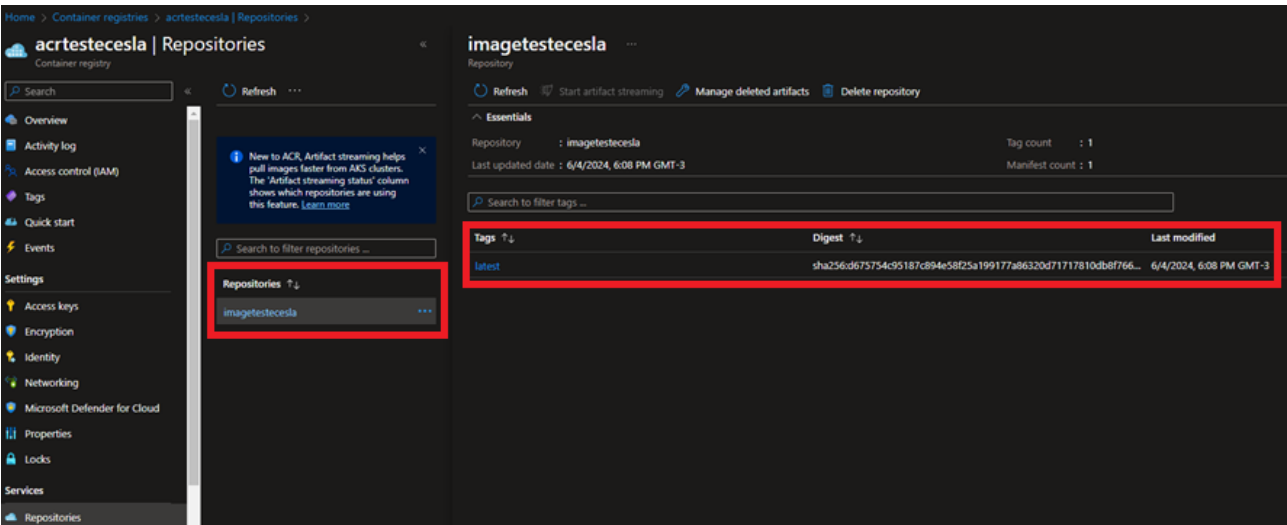
1 Starting: Finalize Job

2 Cleaning up task key

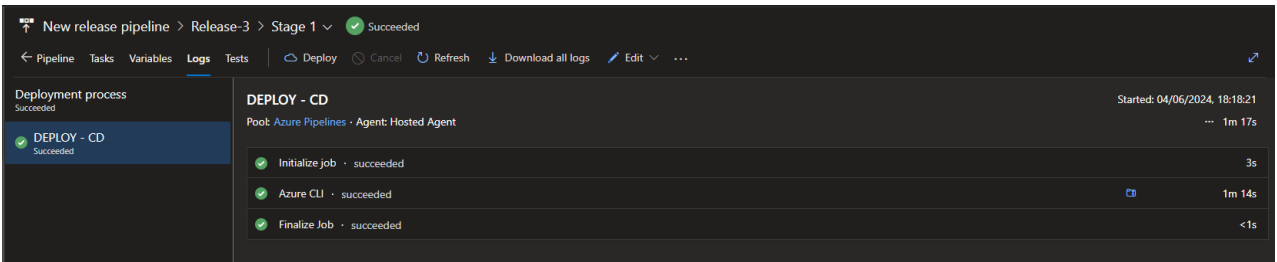
3 Start cleaning up orphan processes.

4 Finishing: Finalize Job

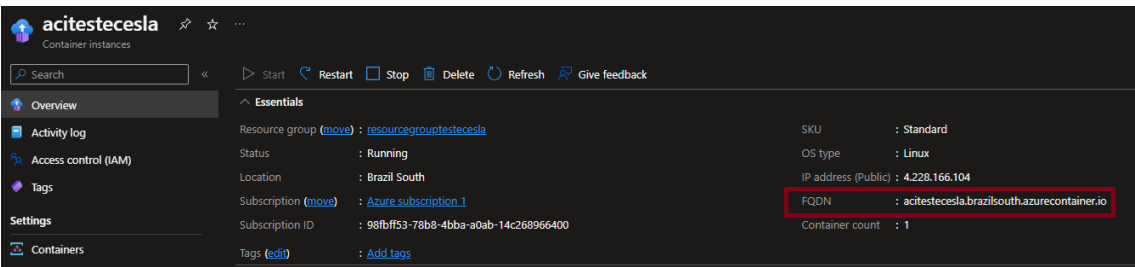
Validação do armazenamento da imagem no Azure Container Registry:



Validação da release com entrega continua (CD):



Validação da criação da instancia no Azure Container Instance:



Validação das variáveis de ambiente (URL) implementada no ACI:

Environment variables	
Key	Value
url	https://cesla.ind.br