

FIDO:/

43473603694952614438241656794348856  
72603588345260193819018008540358315  
44899145022615238810141872721276351  
75920943173795005669125196543856658  
5077271364354107096654083076



## Commit

### Create aws.yml each.t

[Browse files](#)

```
# This workflow will build and push a new container image to Amazon ECR,
# and then will deploy a new task definition to Amazon ECS, when there is a push to the "main"
# branch.
#
# To use this workflow, you will need to complete the following set-up steps:
#
# 1. Create an ECR repository to store your images.
#    For example: `aws ecr create-repository --repository-name my-ecr-repo --region us-east-2`.
#    Replace the value of the `ECR_REPOSITORY` environment variable in the workflow below with your
#    repository's name.
#    Replace the value of the `AWS_REGION` environment variable in the workflow below with your
#    repository's region.
#
# 2. Create an ECS task definition, an ECS cluster, and an ECS service.
#    For example, follow the Getting Started guide on the ECS console:
#    https://us-east-2.console.aws.amazon.com/ecs/home?region=us-east-2#/firstRun
#    Replace the value of the `ECS_SERVICE` environment variable in the workflow below with the name
#    you set for the Amazon ECS service.
#    Replace the value of the `ECS_CLUSTER` environment variable in the workflow below with the name
#    you set for the cluster.
#
# 3. Store your ECS task definition as a JSON file in your repository.
#    The format should follow the output of `aws ecs register-task-definition --generate-cli-
#    skeleton`.
#    Replace the value of the `ECS_TASK_DEFINITION` environment variable in the workflow below with
#    the path to the JSON file.
#    Replace the value of the `CONTAINER_NAME` environment variable in the workflow below with the
#    name of the container
#    in the `containerDefinitions` section of the task definition.
#
# 4. Store an IAM user access key in GitHub Actions secrets named `AWS_ACCESS_KEY_ID` and
#    `AWS_SECRET_ACCESS_KEY`.
#    See the documentation for each action used below for the recommended IAM policies for this IAM
#    user,
#    and best practices on handling the access key credentials.

name: Deploy to Amazon ECS

on:
  push:
    branches: [ "main" ]

env:
  AWS_REGION: MY_AWS_REGION          # set this to your preferred AWS region, e.g. us-west-
1
  ECR_REPOSITORY: MY_ECR_REPOSITORY  # set this to your Amazon ECR repository name
```

```

ECS_SERVICE: MY_ECS_SERVICE           # set this to your Amazon ECS service name
ECS_CLUSTER: MY_ECS_CLUSTER           # set this to your Amazon ECS cluster name
ECS_TASK_DEFINITION: MY_ECS_TASK_DEFINITION # set this to the path to your Amazon ECS task
definition                             # file, e.g. .aws/task-definition.json

CONTAINER_NAME: MY_CONTAINER_NAME      # set this to the name of the container in the
definition                             # containerDefinitions section of your task

permissions:
  contents: read

jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    environment: production

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
          aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
          aws-region: ${ env.AWS_REGION }

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v1

      - name: Build, tag, and push image to Amazon ECR
        id: build-image
        env:
          ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
          IMAGE_TAG: ${ github.sha }
        run: |
          # Build a docker container and
          # push it to ECR so that it can
          # be deployed to ECS.
          docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
          echo "image=$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG" >> $GITHUB_OUTPUT


      - name: Fill in the new image ID in the Amazon ECS task definition
        id: task-def
        uses: aws-actions/amazon-ecs-render-task-definition@v1
        with:
          task-definition: ${ env.ECS_TASK_DEFINITION }
          container-name: ${ env.CONTAINER_NAME }
          image: ${ steps.build-image.outputs.image }

      - name: Deploy Amazon ECS task definition
        uses: aws-actions/amazon-ecs-deploy-task-definition@v1
        with:
          task-definition: ${ steps.task-def.outputs.task-definition }

```

```
service: ${ env.ECS_SERVICE }}
cluster: ${ env.ECS_CLUSTER }}
wait-for-service-stability: true
```

 CumTrust-patch-1

 CumTrust committed 1 minute ago

1 parent [9eb7bb9](#) commit [d57d6a0](#)

Showing 1 changed file with 94 additions and 0 deletions.

Whitespace

Ignore whitespace

Split

Unified

✓ 94  .github/workflows/aws.yml 

... @@ -0,0 +1,94 @@

```
1 + # This workflow will build and push a new
  + # container image to Amazon ECR,
2 + # and then will deploy a new task definition
  + # to Amazon ECS, when there is a push to the
  + # "main" branch.
3 + #
4 + # To use this workflow, you will need to
  + # complete the following set-up steps:
5 + #
6 + # 1. Create an ECR repository to store your
  + # images.
7 + #   For example: `aws ecr create-repository
  + # --repository-name my-ecr-repo --region us-
  + # east-2`.
8 + #   Replace the value of the
  + # `ECR_REPOSITORY` environment variable in the
  + # workflow below with your repository's name.
9 + #   Replace the value of the `AWS_REGION`
  + # environment variable in the workflow below
  + # with your repository's region.
10 + #
11 + # 2. Create an ECS task definition, an ECS
  + # cluster, and an ECS service.
12 + #   For example, follow the Getting Started
  + # guide on the ECS console:
13 + #     https://us-east-
  + # 2.console.aws.amazon.com/ecs/home?region=us-
  + # east-2#/firstRun
14 + #   Replace the value of the `ECS_SERVICE`
  + # environment variable in the workflow below
  + # with the name you set for the Amazon ECS
  + # service.
15 + #   Replace the value of the `ECS_CLUSTER`
  + # environment variable in the workflow below
  + # with the name you set for the cluster.
16 + #
17 + # 3. Store your ECS task definition as a JSON
  + # file in your repository.
```

```
18 + # The format should follow the output of
    `aws ecs register-task-definition --generate-
    cli-skeleton`.
19 + # Replace the value of the
    `ECS_TASK_DEFINITION` environment variable in
    the workflow below with the path to the JSON
    file.
20 + # Replace the value of the
    `CONTAINER_NAME` environment variable in the
    workflow below with the name of the container
21 + # in the `containerDefinitions` section of
    the task definition.
22 + #
23 + # 4. Store an IAM user access key in GitHub
    Actions secrets named `AWS_ACCESS_KEY_ID` and
    `AWS_SECRET_ACCESS_KEY`.
24 + # See the documentation for each action
    used below for the recommended IAM policies
    for this IAM user,
25 + # and best practices on handling the
    access key credentials.
26 +
27 + name: Deploy to Amazon ECS
28 +
29 + on:
30 +   push:
31 +     branches: [ "main" ]
32 +
33 + env:
34 +   AWS_REGION: MY_AWS_REGION
    # set this to your preferred AWS region, e.g.
    us-west-1
35 +   ECR_REPOSITORY: MY_ECR_REPOSITORY
    # set this to your Amazon ECR repository name
36 +   ECS_SERVICE: MY_ECS_SERVICE
    # set this to your Amazon ECS service name
37 +   ECS_CLUSTER: MY_ECS_CLUSTER
    # set this to your Amazon ECS cluster name
38 +   ECS_TASK_DEFINITION: MY_ECS_TASK_DEFINITION
    # set this to the path to your Amazon ECS
    task definition
39 +
    # file, e.g. .aws/task-definition.json
40 +   CONTAINER_NAME: MY_CONTAINER_NAME
    # set this to the name of the container in
    the
41 +
    # containerDefinitions section of your task
    definition
42 +
43 + permissions:
44 +   contents: read
45 +
46 + jobs:
47 +   deploy:
```

```
48 +   name: Deploy
49 +   runs-on: ubuntu-latest
50 +   environment: production
51 +
52 +   steps:
53 +   - name: Checkout
54 +     uses: actions/checkout@v4
55 +
56 +   - name: Configure AWS credentials
57 +     uses: aws-actions/configure-aws-credentials@v1
58 +     with:
59 +       aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
60 +       aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
61 +       aws-region: ${ env.AWS_REGION }
62 +
63 +   - name: Login to Amazon ECR
64 +     id: login-ecr
65 +     uses: aws-actions/amazon-ecr-login@v1
66 +
67 +   - name: Build, tag, and push image to Amazon ECR
68 +     id: build-image
69 +     env:
70 +       ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
71 +       IMAGE_TAG: ${ github.sha }
72 +     run: |
73 +       # Build a docker container and
74 +       # push it to ECR so that it can
75 +       # be deployed to ECS.
76 +       docker build -t
77 +         $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
78 +       docker push
79 +         $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
80 +       echo
81 +         "image=$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG" >> $GITHUB_OUTPUT
82 +
83 +   - name: Fill in the new image ID in the Amazon ECS task definition
84 +     id: task-def
85 +     uses: aws-actions/amazon-ecs-render-task-definition@v1
86 +     with:
87 +       task-definition: ${ env.ECS_TASK_DEFINITION }
88 +       container-name: ${ env.CONTAINER_NAME }
89 +       image: ${ steps.build-image.outputs.image }
```

```
89 +     uses: aws-actions/amazon-ecs-deploy-
    task-definition@v1
90 +     with:
91 +       task-definition: ${{ steps.task-
    def.outputs.task-definition }}
92 +       service: ${{ env.ECS_SERVICE }}
93 +       cluster: ${{ env.ECS_CLUSTER }}
94 +       wait-for-service-stability: true
```


0 comments on commit `d57d6a0`


Write

Preview

H B I    |    |  @  

Leave a comment

 Markdown is supported

 Paste, drop, or click to add files

Comment on this commit

