



Advanced AWS Meetup

CI/CD: GitHub Actions to ECS

Peter Sankauskas | @pas256 | Sept 2024
he/him

CI/CD: GitHub Actions to ECS

I wanted to show you

- A few easy steps
- Simple to follow
- Get it running yourself

How to draw an Owl.

"A fun and creative guide for beginners"



Fig 1. Draw two circles



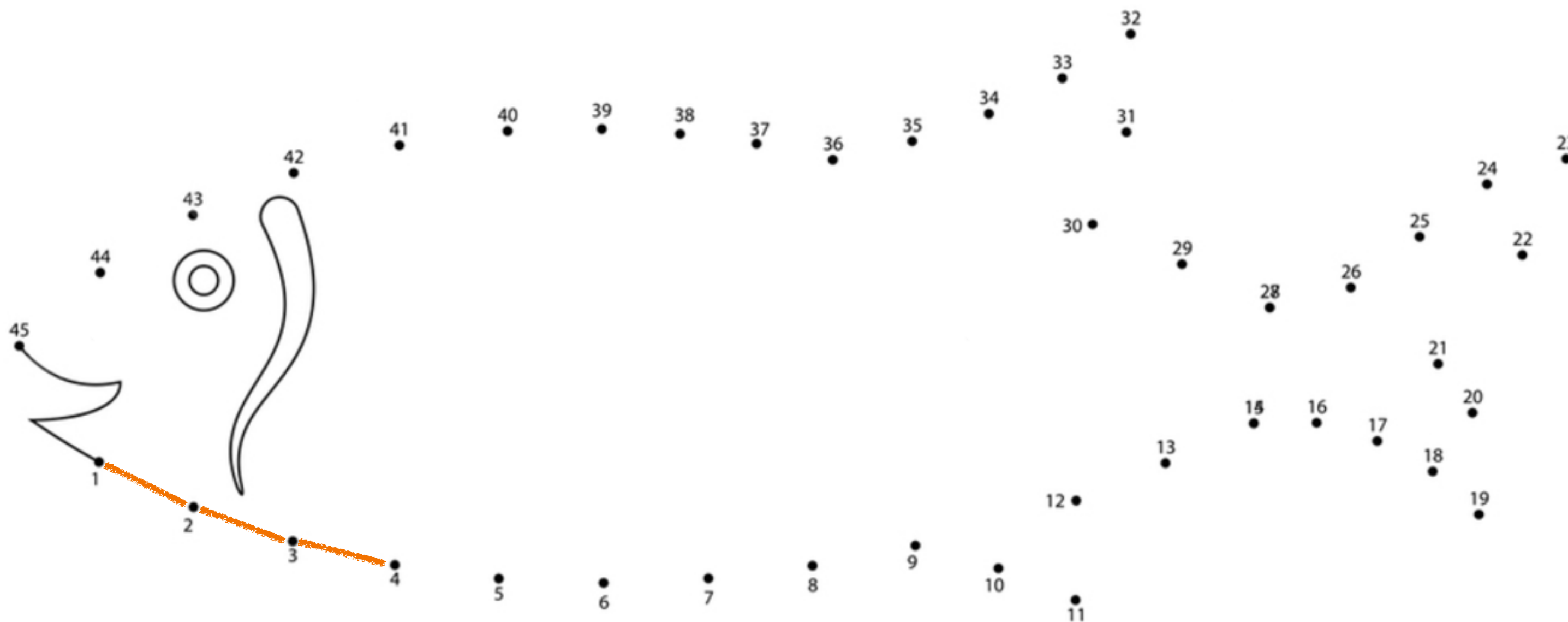
Fig 2. Draw the rest of the damn Owl

CI/CD: GitHub Actions to ECS

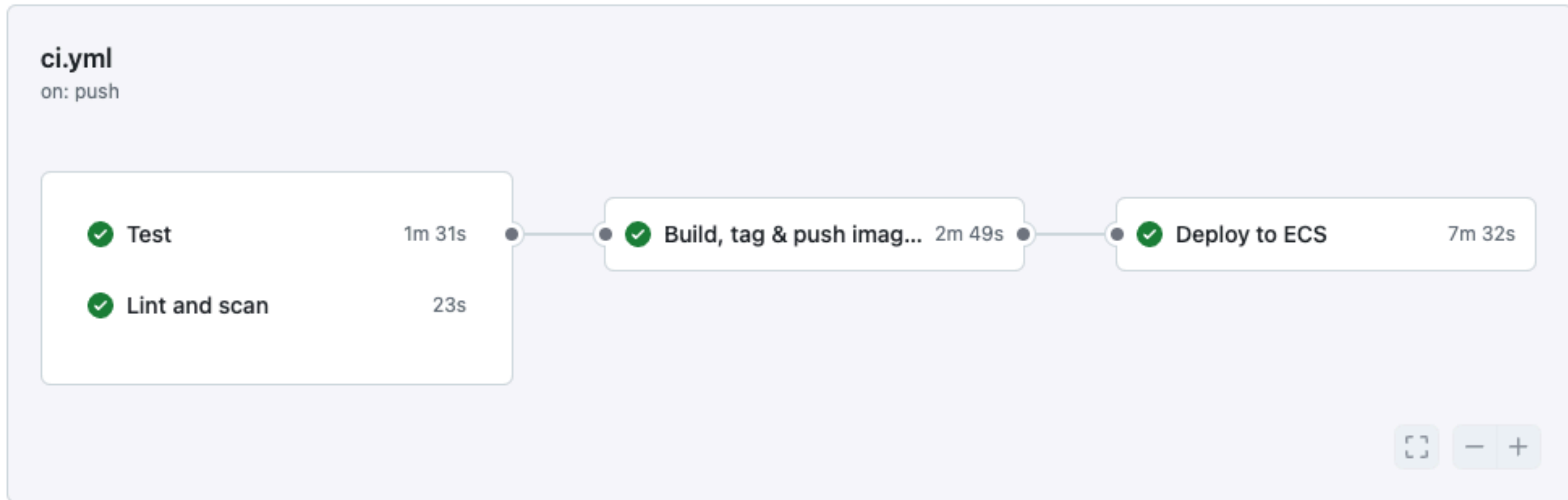


Don't take notes

- Focus and connect the dots
- Slides and code available online

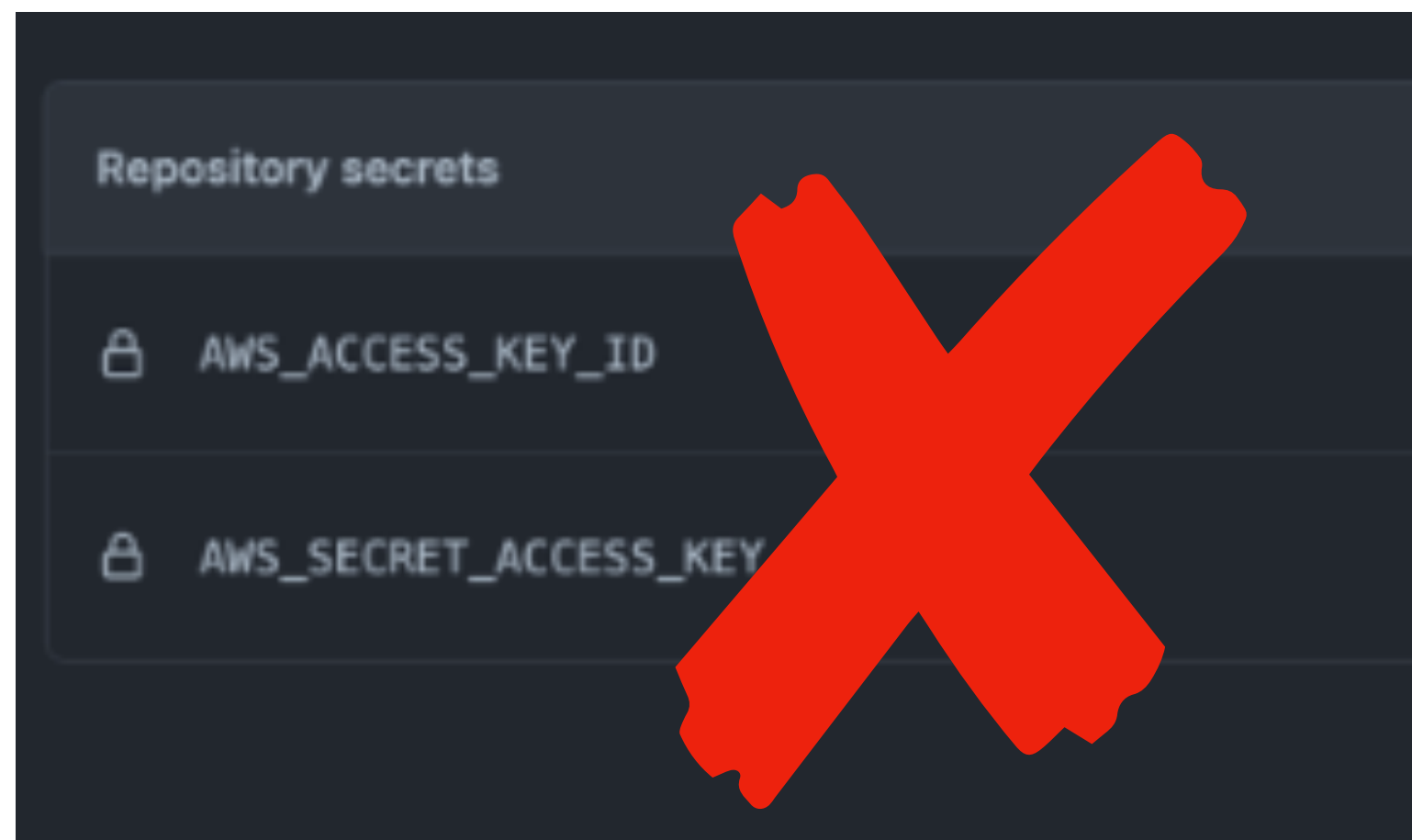


Overview



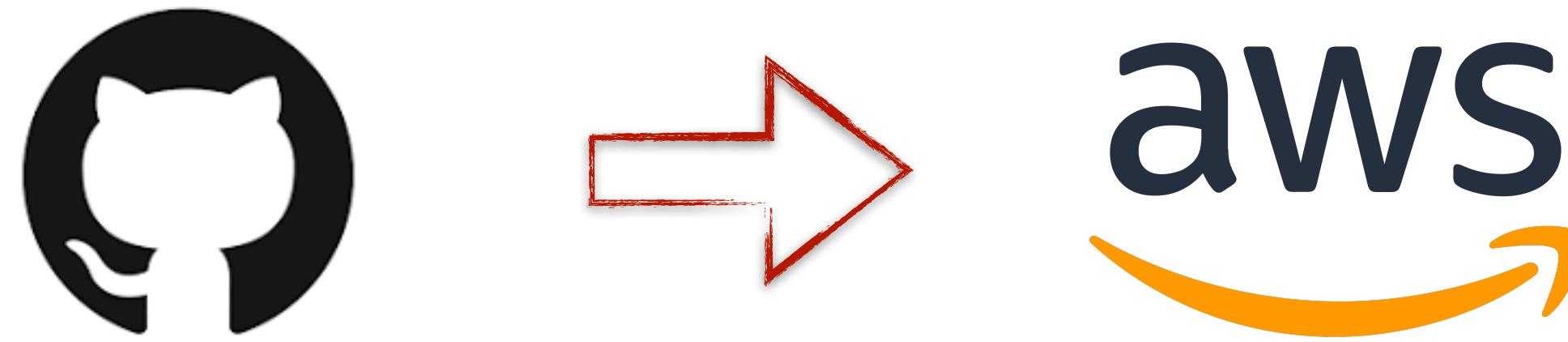
Take advantage of OIDC

- OpenID Connect (OIDC) is an identity layer built on top of the OAuth 2.0 framework
- Means GitHub has access to make AWS API
- ... *without long live credentials* ...



**IS NOT JUST AN
AFTERTHOUGHT**

Required AWS API Permissions



- 1) Push image to ECR repository
- 2) Read current ECS Task Definition the ECS Service is using
- 3) Create and register a new ECS Task Definition using the new image
- 4) Update ECS Service to use new Task Definition

Identity Provider

[IAM](#) > [Identity providers](#) > token.actions.githubusercontent.com

token.actions.githubusercontent.com [Info](#)

Assign role

Delete

Summary

Provider token.actions.githubusercontent.com	Provider Type OpenID Connect	Creation Time May 14, 2024, 14:00 (UTC-07:00)	ARN  arn:aws:iam::730335542499:oidc-provider/token.actions.githubusercontent.com
---	---------------------------------	--	--

[Audiences \(1\)](#)

Endpoint verification

Tags

Audiences (1)

Also known as client ID, audience is a value that identifies the application that is registered with an OpenID Connect provider.

Actions ▼

< 1 >

Audience

☐ sts.amazonaws.com

Official provider name
for GitHub

IAM Role - Trust relationship

[IAM](#) > [Roles](#) > github-oidc-provider-aws

github-oidc-provider-aws [Info](#)

Role assumed by the GitHub OIDC provider.

[Permissions](#)

[Trust relationships](#)

[Tags \(1\)](#)

[Access Advisor](#)

[Revoke sessions](#)

Trusted entities

[Edit trust policy](#)

Entities that can assume this role under specified conditions.

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "Federated": "arn:aws:iam::730335542499:oidc-provider/token.actions.githubusercontent.com"  
8       },  
9       "Action": "sts:AssumeRoleWithWebIdentity",  
10      "Condition": {  
11        "StringLike": {  
12          "token.actions.githubusercontent.com:sub": "repo:pas256/ensorcell:*"  
13        }  
14      }  
15    }  
16  ]  
}
```

Restrict to your
GitHub repository

IAM Role - Permissions

[IAM](#) > [Roles](#) > github-oidc-provider-aws

github-oidc-provider-aws [Info](#)

Role assumed by the GitHub OIDC provider.

[Permissions](#)

[Trust relationships](#)

[Tags \(1\)](#)

[Access Advisor](#)

[Revoke sessions](#)

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.



[Simulate](#)

[Remove](#)

[Add permissions](#)

Filter by Type

All types

< 1 >



☐	Policy name ↗	Type	Attached entities
☐	AmazonEC2ContainerRegistryPowerUser	AWS managed	1
☐	update-service-ensorcell-webapp-prod	Customer managed	1

Cover (1) - push image to ECR

Cover (2), (3) and (4)

Terraform GitHub OIDC module

```
# GitHub OIDC permissions to push images to ECR
module "github-oidc" {
  source    = "terraform-module/github-oidc-provider/aws"
  version   = "~> 2.2"

  create_oidc_provider    = true
  create_oidc_role        = true
  repositories             = ["pas256/ensorcell"]
  # repositories           = ["pas256/ensorcell:ref:refs/heads/main"]
  oidc_role_attach_policies = [
    "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPowerUser",
    aws_iam_policy.update_service_policy.arn,
  ]
}

resource "aws_iam_policy" "update_service_policy" {
  name          = "update-service-ensorcell-webapp-prod"
  description   = "Allows updating the ECS task definition and service"
  policy        = jsonencode({ ... })
}
```

Your GitHub repository

or restrict to main branch only (no PRs)

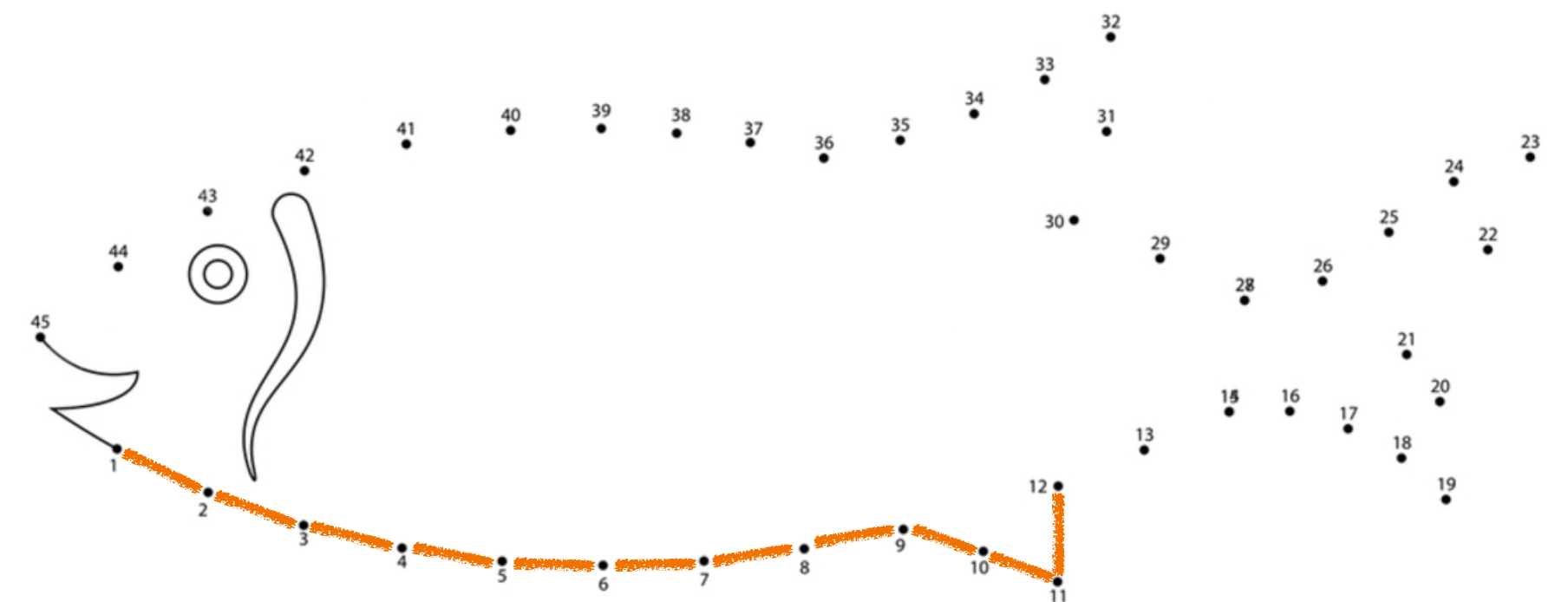
Use AWS's policy to cover (1)

Use our policy to cover the rest

???

AWS API permissions required

- 1) Push image to ECR repository ✓
- 2) Read current ECS Task Definition the ECS Service is using
- 3) Create and register a new ECS Task Definition using the new image
- 4) Update ECS Service to use new Task Definition



Update Policy - Statement 1/3

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ManageTaskDefinitions",
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeTaskDefinition",
        "ecs:RegisterTaskDefinition"
      ],
      "Resource": "*"
    },
    { ... },
    { ... }
  ]
}
```

Covers (2)

Covers (3)

Update Policy - Statement 2/3

```
{
  "Version": "2012-10-17",
  "Statement": [
    { ... },
    {
      "Sid": "DeployService",
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeServices",
        "ecs:UpdateService"
      ],
      "Resource": [
        "arn:aws:ecs:us-west-2:730335542499:service/my-ecs-cluster/my-prod-webapp",
        # Add more services here
      ]
    },
    { ... }
  ]
}
```

Covers (4) or does it?

Task Definition for Service

```
{
  "family": "my-prod-webapp",
  "taskRoleArn": "arn:aws:iam::730335542499:role/my-task-role",
  "executionRoleArn": "arn:aws:iam::730335542499:role/my-ecs-task-execution-role",
  "networkMode": "awsvpc",
  "requiresCompatibilities": ["FARGATE"],
  "containerDefinitions": [ {
    "name": "ensorcell-webapp-prod-webapp",
    "image": "730335542499.dkr.ecr.us-west-2.amazonaws.com/ensorcell/app:main.20240823-200431.719f87f.ci",
    "cpu": 1024,
    "memory": 2048,
    "essential": true,
    "portMappings": [ ... ],
    "environment": [ ... ],
    "secrets": [ ... ],
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "/prod/ecs/ensorcell-webapp",
        "awslogs-region": "us-west-2",
        "awslogs-stream-prefix": "ecs"
      }
    }
  }
]
```

The GitHub OIDC IAM Role needs to pass these roles to the updated Service

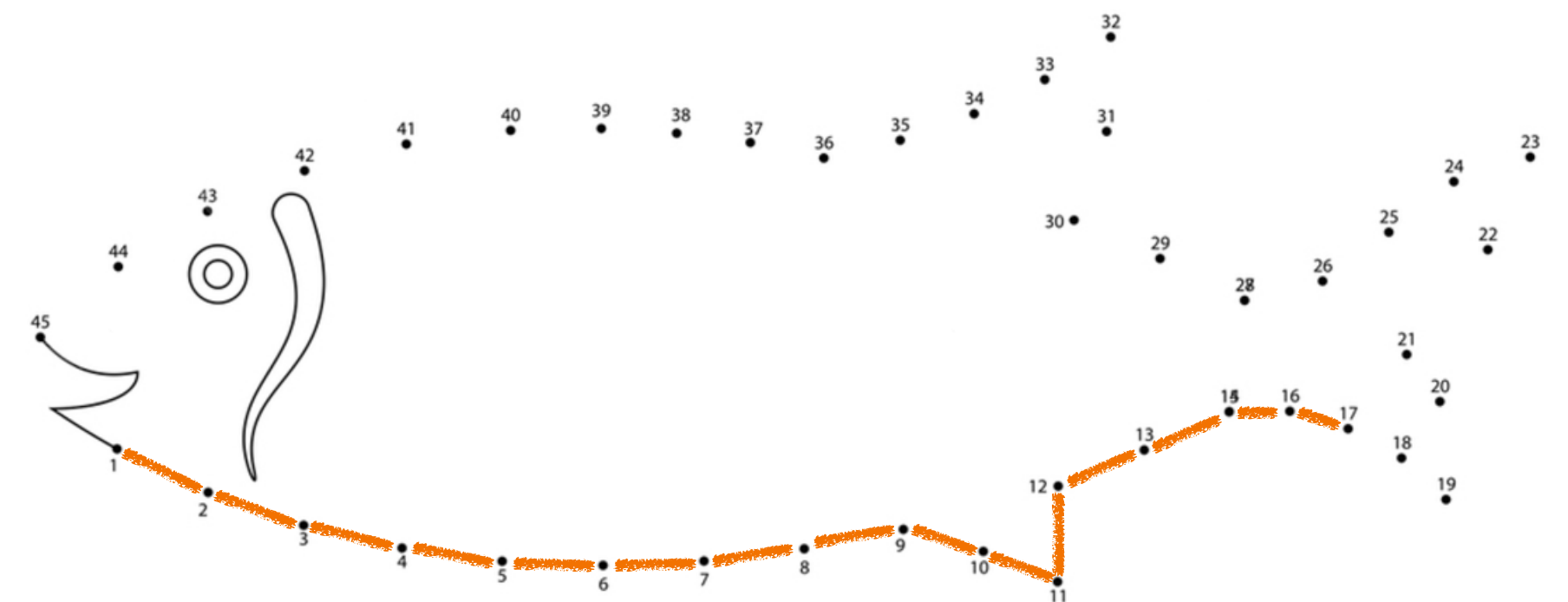
Update Policy - Statement 3/3

```
{
  "Version": "2012-10-17",
  "Statement": [
    { ... },
    { ... },
    {
      "Sid": "PassRolesInTaskDefinition",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::730335542499:role/my-ecs-task-execution-role",
        "arn:aws:iam::730335542499:role/my-task-role"
      ]
    }
  ]
}
```

Covers the rest of (4)

AWS API permissions required

- 1) Push image to ECR repository ✓
- 2) Read current ECS Task Definition the ECS Service is using ✓
- 3) Create and register a new ECS Task Definition using the new image ✓
- 4) Update ECS Service to use new Task Definition ✓



ECR Repository Permissions

[Amazon ECR](#) > [Private registry](#) > [Repositories](#) > [ensorcell/app](#) > Permissions

Permissions

Edit policy JSON

Edit

Statements

No statements

You don't have any permission statements for this repository.

Edit

Nothing required here

GitHub Workflow

pas256 / ensorcell

Type to search

<> Code

Pull requests

Actions

Security 1

Insights

Settings

← CI/CD Pipeline

✓ Bump rails from 7.2.0 to 7.2.1 (#116) #314

Re-run all jobs

Summary

Jobs

Test

Lint and scan

Test assets:precompile

Build, tag & push image to ECR

Deploy to ECS

Run details

Usage

Workflow file

Triggered via push 7 hours ago

pas256 pushed a492e77 main

Status

Success

Total duration

17m 21s

Billable time

15m

Artifacts

1

ci.yml

on: push

Test 1m 31s

Lint and scan 23s

You know how to test your own app

Build, tag & push imag... 2m 49s

Deploy to ECS 7m 32s

Focus here

GitHub Workflow Overview

```
# This workflow will run the entire CI/CD pipeline
name: "CI/CD Pipeline"

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  test:
    name: Test
    ...

  lint:
    name: Lint and scan
    ...
```

```
make_image:
  name: Build, tag & push image to ECR
  ...
```

```
deploy:
  name: Deploy to ECS
  ...
```

.github/workflows/ci.yml

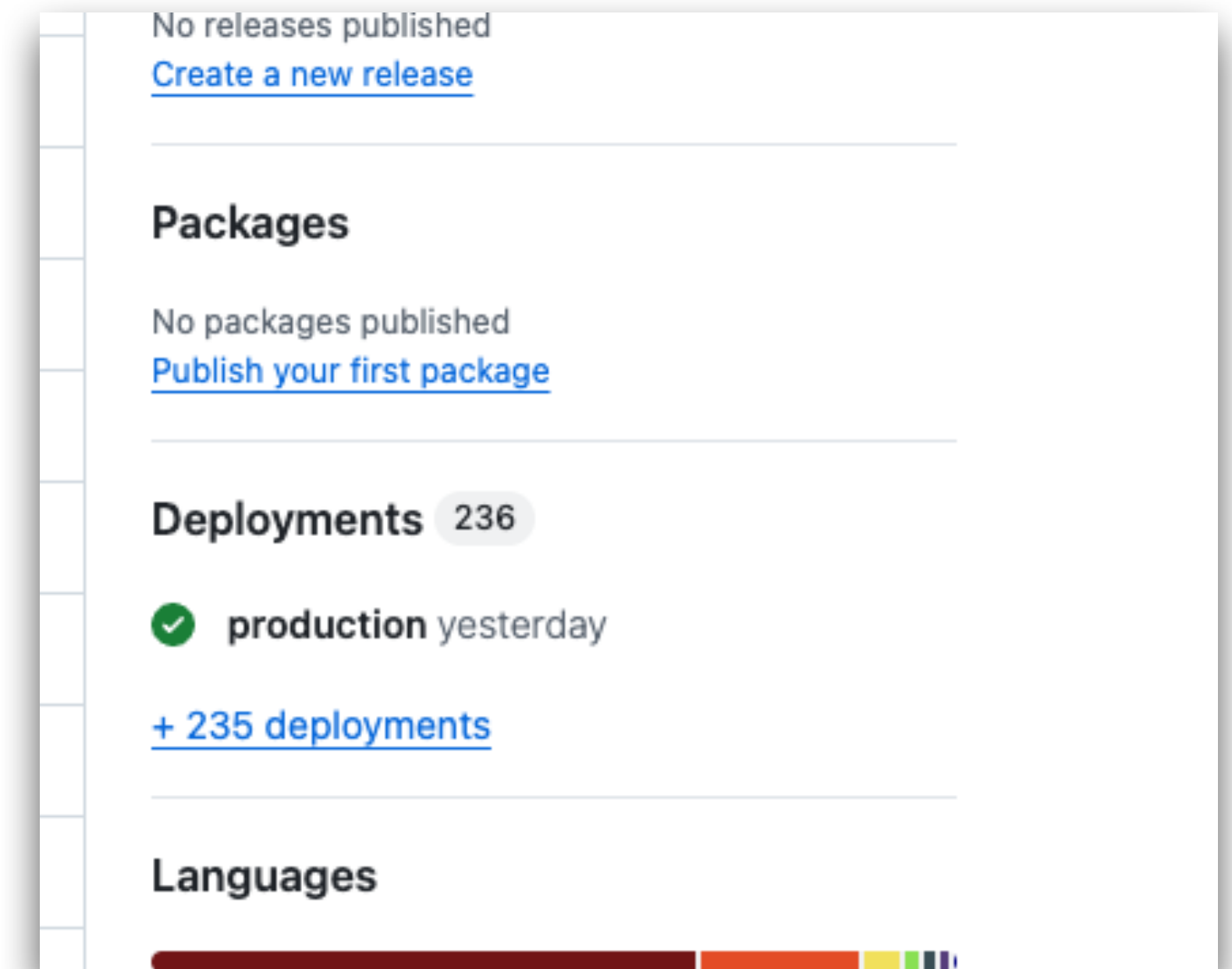
Job: Build, tag & push image to ECR

Build, tag & push image to ECR		
succeeded 10 hours ago in 2m 49s		
Search logs		
 		
>	✔ Set up job	3s
>	✔ Checkout code	0s
	⌚ Checkout code (PR version)	0s
>	✔ Configure AWS Credentials	0s
>	✔ Login to Amazon ECR	0s
>	✔ Set up Docker Buildx	1s
>	✔ Cache Docker layers	9s
>	✔ Calculate image tag	0s
>	✔ Clean up commit message	0s
>	✔ Create the full tag	0s
>	✔ Build, tag and push image to Amazon ECR	2m 14s
>	✔ Move cache to avoid growing forever	0s
>	✔ Post new image tag to Slack	0s
>	✔ Post Build, tag and push image to Amazon ECR	2s
>	✔ Post Cache Docker layers	11s
>	✔ Post Set up Docker Buildx	3s

sooooo many steps

Configure OIDC permissions

```
make_image:  
  name: Build, tag & push image to ECR  
  runs-on: ubuntu-latest  
  environment: production  
  
  # Only run this job if the CI job was successful  
  needs: [test, lint]  
  
  # Do not run on PRs  
  if: github.event_name != 'pull_request'  
  #if: github.actor != 'dependabot[bot]'  
  
  ...
```



.github/workflows/ci.yml

Configure OIDC permissions

```
make_image:
  name: Build, tag & push image to ECR
  runs-on: ubuntu-latest
  environment: production

  # Only run this job if the CI job was successful
  needs: [test, lint]

  # Do not run on PRs
  if: github.event_name != 'pull_request'
  #if: github.actor != 'dependabot[bot]'

  # Permissions needed for OIDC
  permissions:
    contents: read
    id-token: write

  # Outputs to be used by other jobs
  outputs:
    image_tag: ${ steps.image_tag.outputs.image_tag }
    image: ${ steps.full_tag.outputs.image }
```

Make these available for the next job

```
steps:
  - name: Configure AWS Credentials
    uses: aws-actions/configure-aws-credentials@v4
    with:
      audience: sts.amazonaws.com
      aws-region: us-west-2
      role-to-assume: |
arn:aws:iam::730335542499:role/github-oidc-provider-aws

    ...
```

.github/workflows/ci.yml

Configure OIDC permissions

```
make_image:
  name: Build, tag & push image to ECR
  runs-on: ubuntu-latest
  environment: production

  # Only run this job if the CI job was successful
  needs: [test, lint]

  # Do not run on PRs
  if: github.event_name != 'pull_request'
  #if: github.actor != 'dependabot[bot]'

  # Permissions needed for OIDC
  permissions:
    contents: read
    id-token: write

  # Outputs to be used by other jobs
  outputs:
    image_tag: ${ steps.image_tag.outputs.image_tag }
    image: ${ steps.full_tag.outputs.image }
```

OIDC AWS Magic

```
steps:
  - name: Configure AWS Credentials
    uses: aws-actions/configure-aws-credentials@v4
    with:
      audience: sts.amazonaws.com
      aws-region: us-west-2
      role-to-assume: |
arn:aws:iam::730335542499:role/github-oidc-provider-aws

...
```

.github/workflows/ci.yml

IAM Role - Permissions

[IAM](#) > [Roles](#) > github-oidc-provider-aws

github-oidc-provider-aws [Info](#)

Role assumed by the GitHub OIDC provider.

Permissions

Trust relationships

Tags (1)

Access Advisor

Revoke sessions

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.



Simulate [↗](#)

Remove

Add permissions ▼

Filter by Type

All types ▼

< 1 >



Policy name [↗](#)



Type ▼



Attached entities ▼



 [AmazonEC2ContainerRegistryPowerUser](#)

AWS managed

1



[update-service-ensorcell-webapp-prod](#)

Customer managed

1

Configure OIDC permissions

```
make_image:
  name: Build, tag & push image to ECR
  runs-on: ubuntu-latest
  environment: production

  # Only run this job if the CI job was successful
  needs: [test, lint]

  # Do not run on PRs
  if: github.event_name != 'pull_request'
  #if: github.actor != 'dependabot[bot]'

  # Permissions needed for OIDC
  permissions:
    contents: read
    id-token: write

  # Outputs to be used by other jobs
  outputs:
    image_tag: ${ steps.image_tag.outputs.image_tag }
    image: ${ steps.full_tag.outputs.image }
```

OIDC AWS Magic

```
steps:
  - name: Configure AWS Credentials
    uses: aws-actions/configure-aws-credentials@v4
    with:
      audience: sts.amazonaws.com
      aws-region: us-west-2
      role-to-assume: |
arn:aws:iam::730335542499:role/github-oidc-provider-aws

...
```

.github/workflows/ci.yml

Configure OIDC permissions

```
make_image:
  name: Build, tag & push image to ECR
  runs-on: ubuntu-latest
  environment: production

  # Only run this job if the CI job was successful
  needs: [test, lint]

  # Do not run on PRs
  if: ${ github.event_name != 'pull_request' }}
  #if: ${ github.actor != 'dependabot[bot]' }}

  # Permissions needed for OIDC
  permissions:
    contents: read
    id-token: write

  # Outputs to be used by other jobs
  outputs:
    image_tag: ${ steps.image_tag.outputs.image_tag }}
    image: ${ steps.full_tag.outputs.image }}
```

```
steps:
  - name: Configure AWS Credentials
    uses: aws-actions/configure-aws-credentials@v4
    with:
      audience: sts.amazonaws.com
      aws-region: us-west-2
      role-to-assume: |
arn:aws:iam::730335542499:role/github-oidc-provider-aws

  - name: Login to Amazon ECR
    id: login_ecr
    uses: aws-actions/amazon-ecr-login@v2

  ...
```

.github/workflows/ci.yml

Checkout code, set up Docker, tag

```
make_image:
  name: Build, tag & push image to ECR
  steps:
    ...
    - name: Checkout code
      if: ${ github.event_name != 'pull_request' }
      uses: actions/checkout@v4

    - name: Checkout code (PR version)
      if: ${ github.event_name == 'pull_request' }
      uses: actions/checkout@v4
      with:
        # PRs do a merge commit before running the
        # workflow, so we need to checkout the code
        # without that to get the correct SHA.
        # See: https://github.com/actions/checkout/issues/426
        ref: ${ github.event.pull_request.head.sha }

    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v3
```

```
- name: Cache Docker layers
  uses: actions/cache@v4
  with:
    path: /tmp/.buildx-cache
    key: ${ runner.os }-buildx-${ github.sha }
    restore-keys: |
      ${ runner.os }-buildx-
```

```
- name: Calculate image tag
  id: image_tag
  run: echo "image_tag=$(./bin/image-tag.sh)" >>
    $GITHUB_OUTPUT
```

Save output

```
- name: Clean up commit message
  id: commit
  run: echo "message=$(git log -1 --pretty=%B | head -1)"
    >> $GITHUB_OUTPUT
```

Save output

...

.github/workflows/ci.yml

Tag script

```
#!/bin/bash

set -e

NOW=$(date +%Y%m%d-%H%M%S)
SHORT_SHA=$(git rev-parse --short HEAD)
BRANCH=$(git rev-parse --abbrev-ref HEAD \
| sed -re 's/[^a-z0-9-]/-/g' \
| cut -c1-30 \
| sed -re 's/--+/-/g' \
| sed -re 's/-$//' )

# The image tag will be in the format:
# - GitHub PR:      pr-123.YYYMMDD-HHMMSS.SHA0123.ci
# - GitHub main:    BRANCH.YYYMMDD-HHMMSS.SHA0123.ci
# - Local main:     BRANCH.YYYMMDD-HHMMSS.SHA0123.local
# - Local branch:   BRANCH.YYYMMDD-HHMMSS.SHA0123.local
```

```
if [[ "$GITHUB_REF" == refs/pull* ]]; then
    IMAGE_TAG_PREFIX=pr-$(echo $GITHUB_REF | cut -d '/' -f3)
else
    IMAGE_TAG_PREFIX=$BRANCH
fi

if [[ "$CI" == true ]]; then
    IMAGE_TAG_SUFFIX=ci
else
    IMAGE_TAG_SUFFIX=local
fi

IMAGE_TAG="${IMAGE_TAG_PREFIX}.${NOW}.${SHORT_SHA}.${IMAGE_TAG_SUFFIX}"

echo $IMAGE_TAG
```

image-tag.sh

Task Definition for Service

```
{
  "family": "my-prod-webapp",
  "taskRoleArn": "arn:aws:iam::730335542499:role/my-task-role",
  "executionRoleArn": "arn:aws:iam::730335542499:role/my-ecs-task-execution-role",
  "networkMode": "awsvpc",
  "requiresCompatibilities": ["FARGATE"],
  "containerDefinitions": [ {
    "name": "ensorcell-webapp-prod-webapp",
    "image": "730335542499.dkr.ecr.us-west-2.amazonaws.com/ensorcell/app:main.20240823-200431.719f87f.ci",
    "cpu": 1024,
    "memory": 2048,
    "essential": true,
    "portMappings": [ ... ],
    "environment": [ ... ],
    "secrets": [ ... ],
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "/prod/ecs/ensorcell-webapp",
        "awslogs-region": "us-west-2",
        "awslogs-stream-prefix": "ecs"
      }
    }
  } ]
}
```

Checkout code, set up Docker, tag

```
make_image:
  name: Build, tag & push image to ECR
  steps:
    ...
    - name: Checkout code
      if: ${ github.event_name != 'pull_request' }
      uses: actions/checkout@v4

    - name: Checkout code (PR version)
      if: ${ github.event_name == 'pull_request' }
      uses: actions/checkout@v4
      with:
        # PRs do a merge commit before running the
        # workflow, so we need to checkout the code
        # without that to get the correct SHA.
        # See: https://github.com/actions/checkout/issues/426
        ref: ${ github.event.pull_request.head.sha }

    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v3
```

```
- name: Cache Docker layers
  uses: actions/cache@v4
  with:
    path: /tmp/.buildx-cache
    key: ${ runner.os }-buildx-${ github.sha }
    restore-keys: |
      ${ runner.os }-buildx-
```

```
- name: Calculate image tag
  id: image_tag
  run: echo "image_tag=$(./bin/image-tag.sh)" >>
    $GITHUB_OUTPUT
```

Save output

```
- name: Clean up commit message
  id: commit
  run: echo "message=$(git log -1 --pretty=%B | head -1)"
    >> $GITHUB_OUTPUT
```

Save output

...

.github/workflows/ci.yml

Tag & push to ECR

```
make_image:
  name: Build, tag & push image to ECR
  steps:
    ...
    - name: Create the full tag
      id: full_tag
      env:
        ECR_REGISTRY: ${ steps.login_ecr.outputs.registry }
        ECR_REPOSITORY: ensorcell/app
        IMAGE_TAG: ${ steps.image_tag.outputs.image_tag }
      run: |
        echo "image=$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG"
        >> $GITHUB_OUTPUT
      Save output
      pwd
      echo "$IMAGE_TAG" > ./RELEASE
      ls -la
      Include in image
```

```
- name: Build, tag and push image to Amazon ECR
  uses: docker/build-push-action@v6
  with:
    context: .
    push: true
    tags: ${ steps.full_tag.outputs.image }
    cache-from: type=local,src=/tmp/.buildx-cache
    cache-to: type=local,dest=/tmp/.buildx-cache-new,mode=max
  Use output
```

...

.github/workflows/ci.yml

Clean up & Post to Slack

```
make_image:
  name: Build, tag & push image to ECR
  steps:
    ...
    - # Temp fix
      # https://github.com/docker/build-push-action/issues/252
      # https://github.com/moby/buildkit/issues/1896
      name: Move cache to avoid growing forever
      run: |
        rm -rf /tmp/.buildx-cache
        mv /tmp/.buildx-cache-new /tmp/.buildx-cache
```

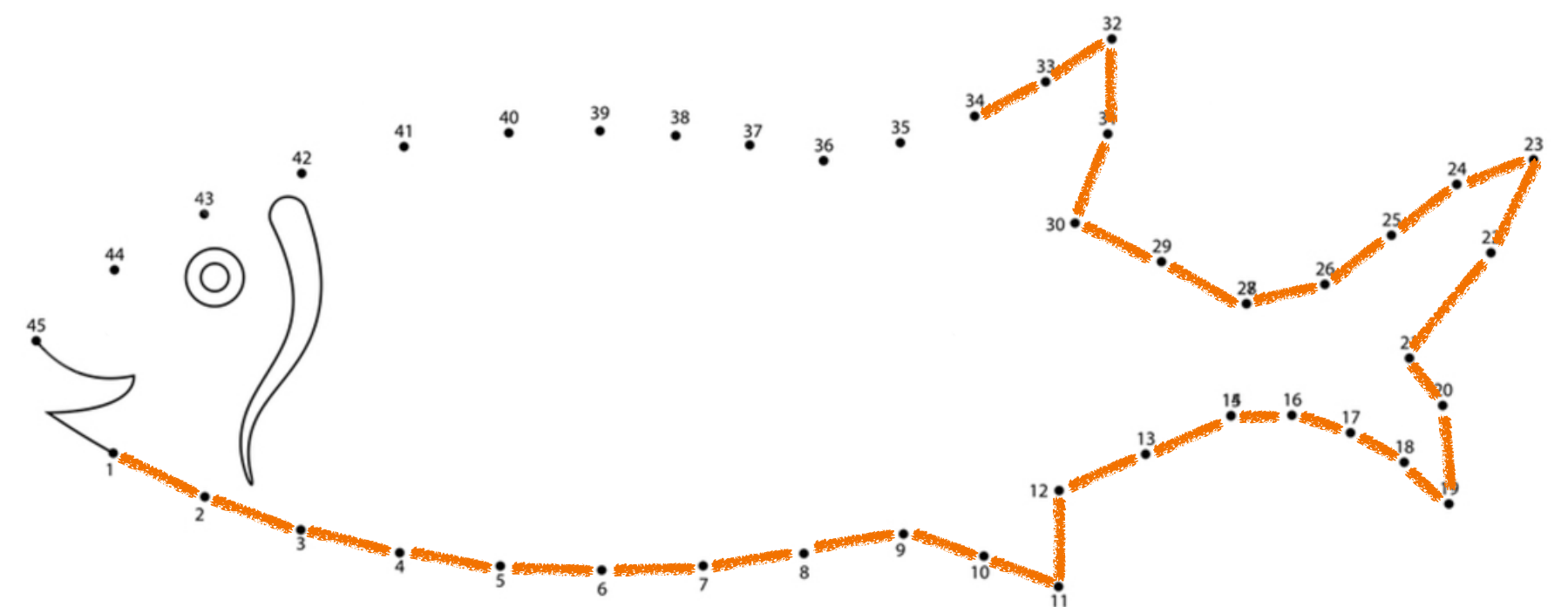
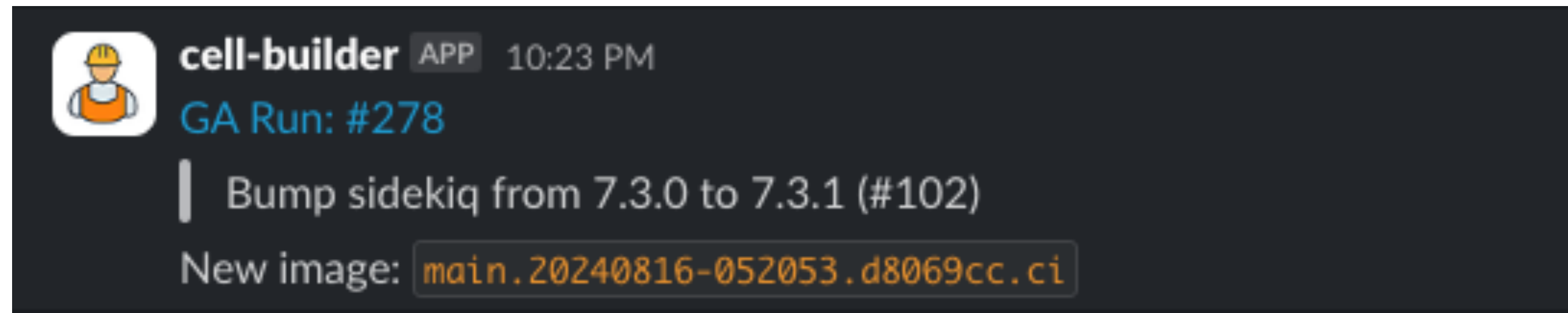
Optional

```
- name: Post new image tag to Slack
  uses: slackapi/slack-github-action@v1.26.0
  env:
    SLACK_BOT_TOKEN: ${ secrets.SLACK_BOT_TOKEN }
    IMAGE_TAG: ${ steps.image_tag.outputs.image_tag }
    COMMIT_MESSAGE: ${ steps.commit.outputs.message }
  with:
    channel-id: "C075ND1EKEH"
```

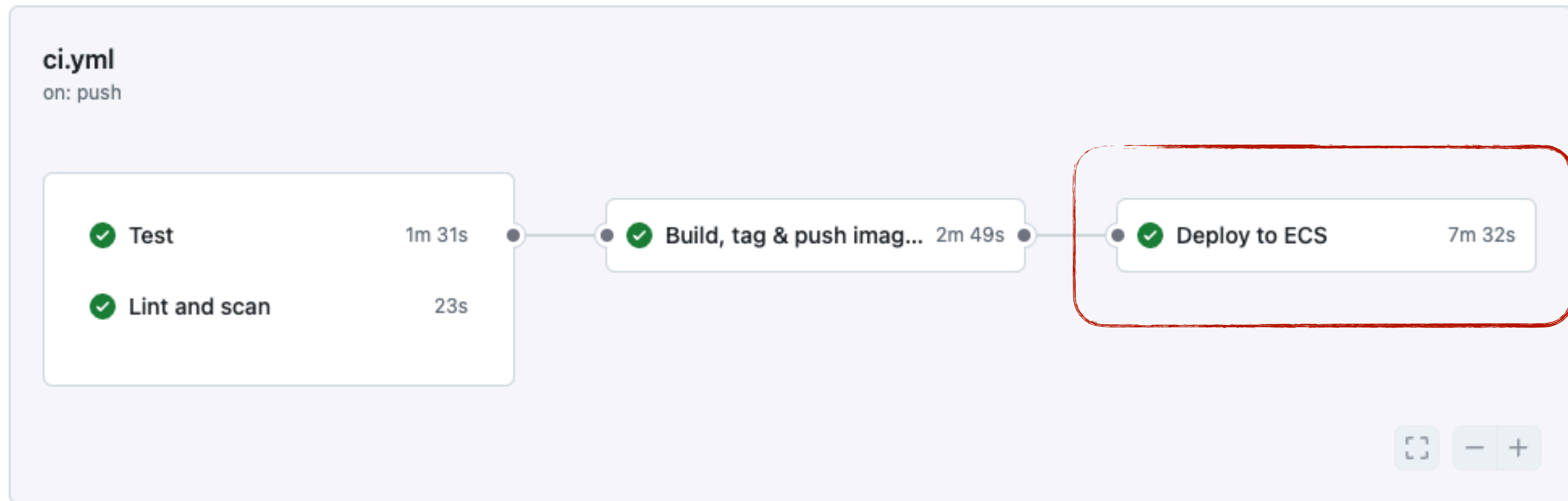
```
payload: |
{
  "blocks": [
    {
      "type": "section",
      "text": {
        "type": "mrkdwn",
        "text":
"<https://github.com/pas256/ensorcell/actions/runs/$
{{ github.run_id }}|GA Run: #${{ github.run_number }}>\n
> ${ env.COMMIT_MESSAGE }\n
New image: `${ env.IMAGE_TAG }`"
      }
    ]
  }
}
```

.github/workflows/ci.yml

Slack notification



Overview



Job: Deploy to ECS

Deploy to ECS		
succeeded 9 hours ago in 7m 32s		
Q Search logs ↺ ⚙		
>	✔ Set up job	2s
>	✔ Checkout code	1s
>	✔ Configure AWS Credentials	1s
>	✔ Download task definition for webapp	1s
>	✔ Download task definition for sidekiq workers	0s
>	✔ Fill in the new image ID in the Amazon ECS task definition for webapp	0s
>	✔ Fill in the new image ID in the Amazon ECS task definition for workers	0s
>	✔ Notify Slack deployment for workers has started	0s
>	✔ Start timer to monitor the duration of the deployment	0s
>	✔ Deploy Amazon ECS task definition workers	3m 56s
>	✔ Format task definition for workers	0s
>	✔ Update timer and calculate duration	0s
>	✔ Update Slack with success for workers	0s
	⊗ Update Slack with failure for workers	0s
	⊗ Set flag for workers failure	0s
>	✔ Deploy Amazon ECS task definition webapp	3m 26s
>	✔ Format task definition for webapp	0s
>	✔ Update timer and calculate duration	0s

<= is doing 2 ECS services

Job: Deploy to ECS

- > ✓ Download task definition for sidekiq workers 0s
- > ✓ Fill in the new image ID in the Amazon ECS task definition for webapp 0s
- > ✓ Fill in the new image ID in the Amazon ECS task definition for workers 0s
- > ✓ Notify Slack deployment for workers has started 0s
- > ✓ Start timer to monitor the duration of the deployment 0s

✓ Deploy Amazon ECS task definition workers 3m 56s

1 ▶ Run `aws-actions/amazon-ecs-deploy-task-definition@v2`

```
15 Warning: Ignoring property 'compatibilities' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
16 Warning: Ignoring property 'taskDefinitionArn' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
17 Warning: Ignoring property 'requiresAttributes' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
18 Warning: Ignoring property 'revision' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
19 Warning: Ignoring property 'status' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
20 Warning: Ignoring property 'registeredAt' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
21 Warning: Ignoring property 'registeredBy' in the task definition file. This property is returned by the Amazon ECS
DescribeTaskDefinition API and may be shown in the ECS console, but it is not a valid field when registering a new task
definition. This field can be safely removed from your task definition file.
22 Deployment started. Watch this deployment's progress in the Amazon ECS console: https://us-west-2.console.aws.amazon.com/ecs/v2/clusters/cell-prod-ecs-cluster/services/ensorcell-webapp-prod-workers/events?region=us-west-2
```

Main deploy step

Warnings are safe to ignore

They are the JSON fields left by duplicating the old Task Definition and will be ignored by the action

Deploy

deploy:

name: Deploy to ECS
runs-on: ubuntu-latest
needs: [make_image]
environment: production
concurrency: production

Run only if image built

Only run 1 at a time

Permissions needed for OIDC

permissions:
contents: read
id-token: write

<= Same as build job =>

steps:

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

- name: Configure AWS Credentials
uses: aws-actions/configure-aws-credentials@v4
with:
audience: sts.amazonaws.com
aws-region: us-west-2
role-to-assume: |
arn:aws:iam::730335542499:role/github-oidc-provider-aws

.github/workflows/ci.yml

Deploy

```
deploy:
  name: Deploy to ECS
  steps:
    ...
    - name: Download task definition
      run: |
        aws ecs describe-task-definition \
          --task-definition my-prod-webapp \
          --query taskDefinition \
          > /tmp/webapp-task-definition.json

    - name: Put new image ID in task definition
      id: task-def-webapp
      uses: aws-actions/amazon-ecs-render-task-definition@v1
      with:
        task-definition: /tmp/webapp-task-definition.json
        container-name: ensorcell-webapp-prod-webapp
        image: ${ needs.make_image.outputs.image }
```

This is what causes those warnings

Must match
name of container
in Task Definition

[.github/workflows/ci.yml](#)

Task Definition for Service

```
{
  "family": "my-prod-webapp",
  "taskRoleArn": "arn:aws:iam::730335542499:role/my-task-role",
  "executionRoleArn": "arn:aws:iam::730335542499:role/my-ecs-task-execution-role",
  "networkMode": "awsvpc",
  "requiresCompatibilities": ["FARGATE"],
  "containerDefinitions": [ {
    "name": "ensorcell-webapp-prod-webapp",
    "image": "730335542499.dkr.ecr.us-west-2.amazonaws.com/ensorcell/app:main.20240823-200431.719f87f.ci",
    "cpu": 1024,
    "memory": 2048,
    "essential": true,
    "portMappings": [ ... ],
    "environment": [ ... ],
    "secrets": [ ... ],
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "/prod/ecs/ensorcell-webapp",
        "awslogs-region": "us-west-2",
        "awslogs-stream-prefix": "ecs"
      }
    }
  }
]
```

Deploy

```
deploy:
  name: Deploy to ECS
  steps:
    ...
    - name: Download task definition
      run: |
        aws ecs describe-task-definition \
          --task-definition my-prod-webapp \
          --query taskDefinition \
          > /tmp/webapp-task-definition.json

    - name: Put new image ID in task definition
      id: task-def-webapp
      uses: aws-actions/amazon-ecs-render-task-definition@v1
      with:
        task-definition: /tmp/webapp-task-definition.json
        container-name: ensorcell-webapp-prod-webapp
        image: ${ needs.make_image.outputs.image }
```

This is what causes those warnings

Must match
name of container
in Task Definition

[.github/workflows/ci.yml](#)

Deploy

```
deploy:
  name: Deploy to ECS
  steps:
    ...
    - name: Download task definition
      run: |
        aws ecs describe-task-definition \
          --task-definition my-prod-webapp \
          --query taskDefinition \
          > /tmp/webapp-task-definition.json

    - name: Put new image ID in task definition
      id: task-def-webapp
      uses: aws-actions/amazon-ecs-render-task-definition@v1
      with:
        task-definition: /tmp/webapp-task-definition.json
        container-name: ensorcell-webapp-prod-webapp
        image: ${{ needs.make_image.outputs.image }}
```

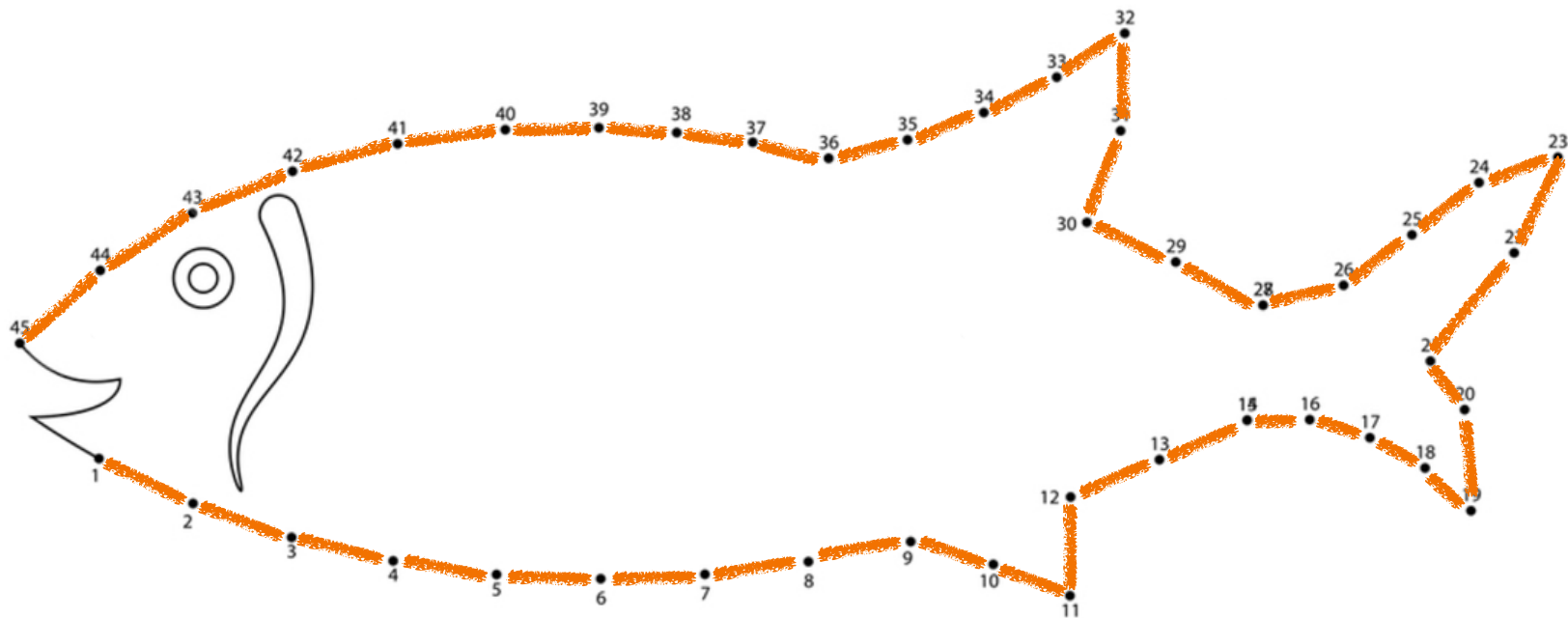
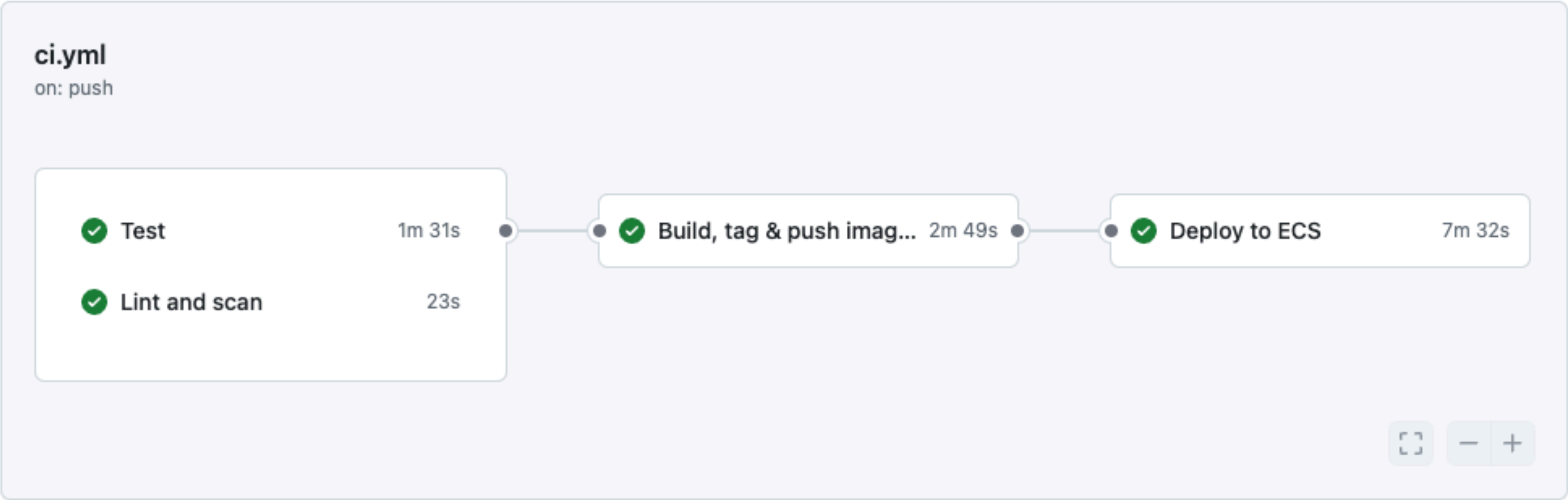
This is what causes those warnings

```
- name: Deploy Amazon ECS task definition workers
  id: deploy-workers
  uses: aws-actions/amazon-ecs-deploy-task-definition@v2
  with:
    task-definition: ${{steps.task-def-workers.outputs.task-definition}}
    cluster: my-ecs-cluster
    service: my-prod-webapp
    wait-for-service-stability: true
    wait-for-minutes: 10
```

From policy (4)

.github/workflows/ci.yml

CI/CD Pipeline



ECR Repository Lifecycle Rules

[Amazon ECR](#) > [Private registry](#) > [Repositories](#) > [ensorcell/app](#) > Lifecycle policy

ensorcell/app

Lifecycle policy rules				Reorder	Edit	Delete	Edit test rules	Actions ▼	Create rule
	Priority ▲	Rule description	Summary						
☐	1	Keep the last 20 main images only	expire imageCountMoreThan (20) tagged prefix [main.]						
☐	2	Remove PR images after 45 days	expire sinceImagePushed (7 days) tagged prefix [pr-]						
☐	3	Remove anything untagged after 7 days	expire sinceImagePushed (7 days) untagged						

Lifecycle events history		↺
Filter events		< 1 2 3 4 5 6 7 ... 26 >
Completed at ▼	Message	
2024-08-23T15:44:34.000Z	PolicyExecutionEvent 2 images affected	
2024-08-23T15:44:34.000Z	PolicyExecutionEvent 0 images affected	
2024-08-22T23:46:21.000Z	PolicyExecutionEvent 6 images affected	
2024-08-22T23:46:21.000Z	PolicyExecutionEvent 0 images affected	
2024-08-22T07:45:15.000Z	PolicyExecutionEvent 3 images affected	
2024-08-21T15:43:03.000Z	PolicyExecutionEvent 1 images affected	

Pro Tip:
Delete old images to save \$\$\$

Resources & thank you

Slides

<https://answersforaws.com/slides>

Code

<https://github.com/pas256/cicd-talk-2024>

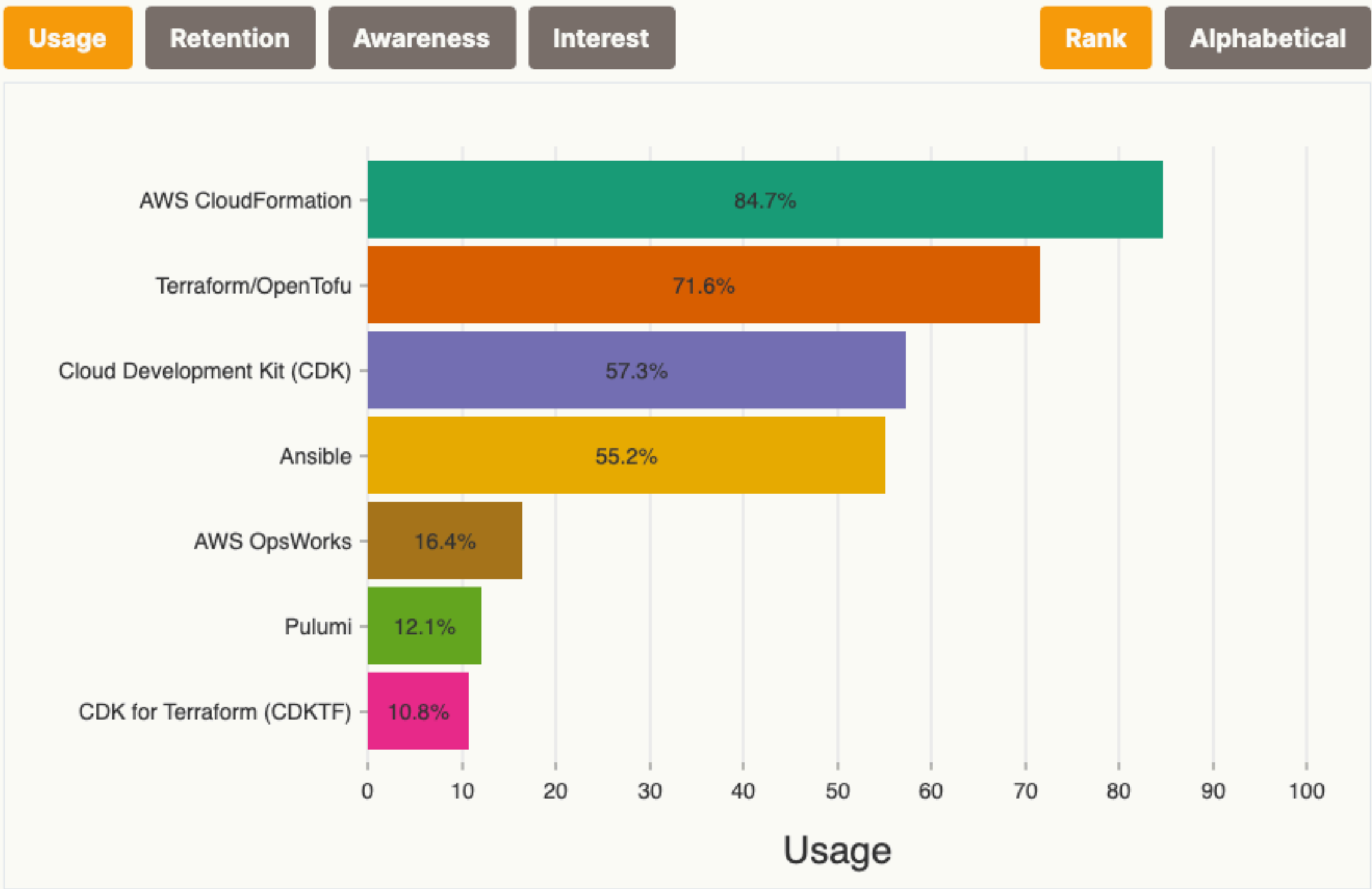


Check out: answersforaws.com



Infrastructure as Code

Ranking of Infrastructure as Code (IaC) tools and services used to manage AWS resources.



Serverless

Trend over the years

