



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"

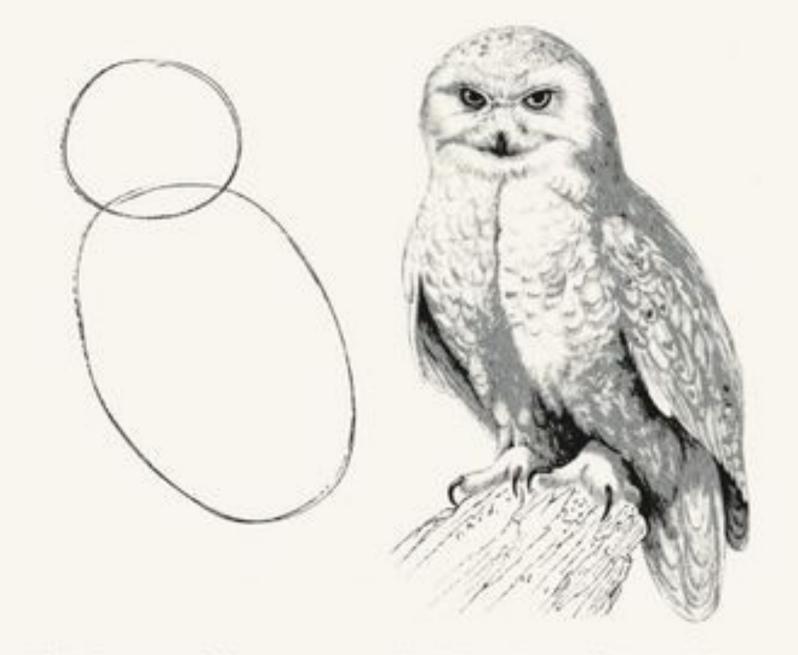


Fig 1. Draw two circles

Fig 2. Draw the rest of the damn Owl



Don't take notes



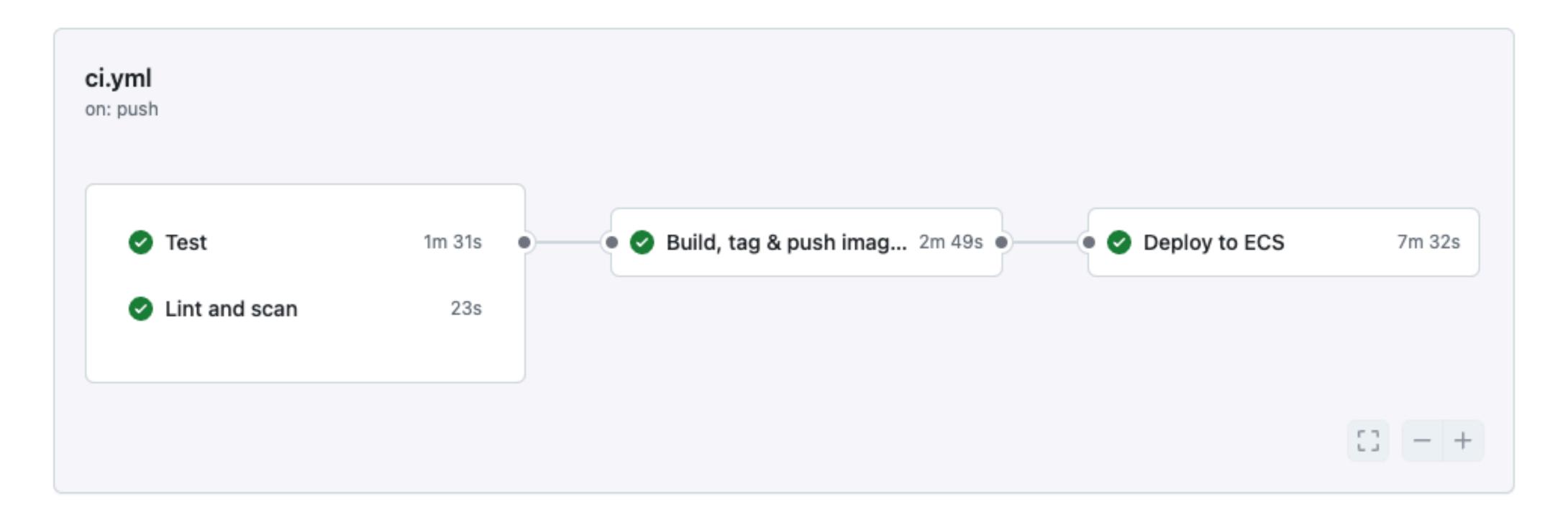
- Focus and connect the dots
- Slides and code available online





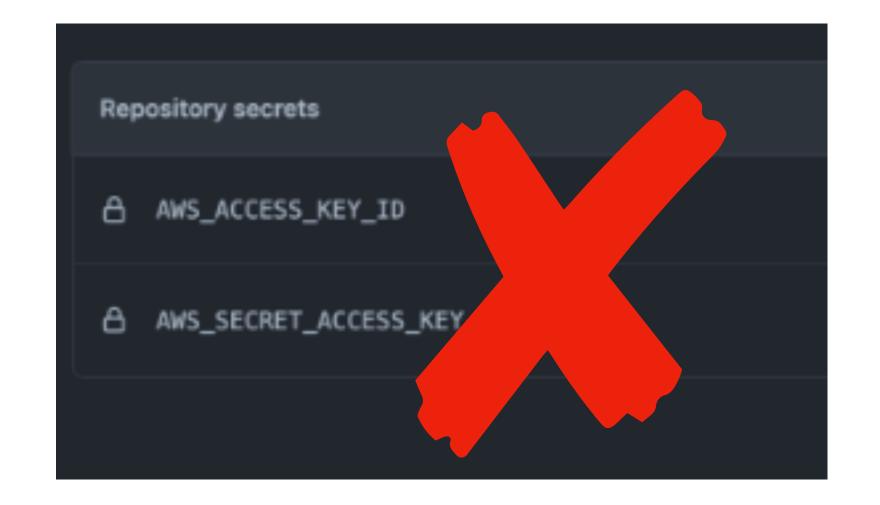
Overview

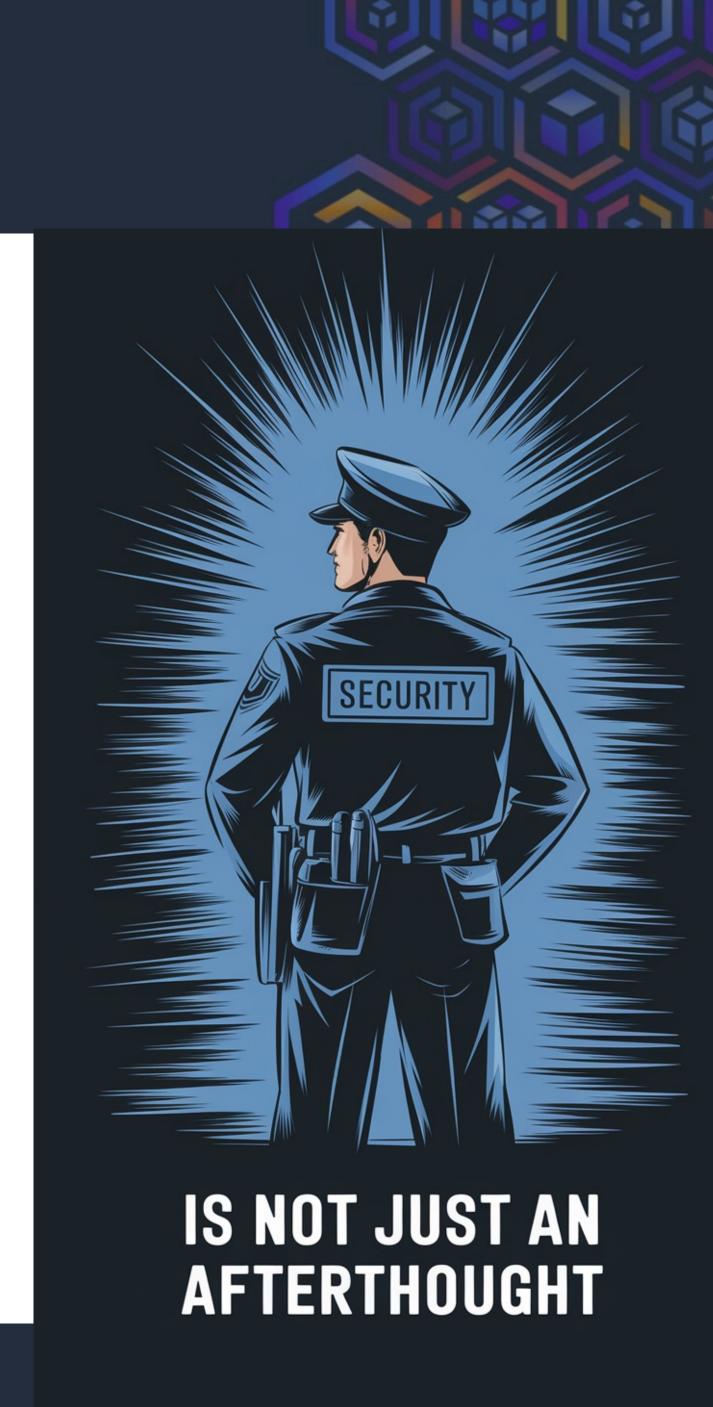




Take advantage of OIDC

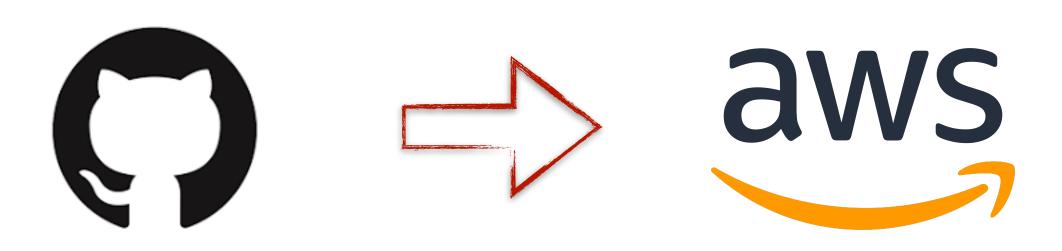
- OpenID Connect (OIDC) is an identity layer built on top of the OAuth 2.0 framework
- Grant GitHub access to make AWS API calls
 - ... without long live credentials ...





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



Identity providers > token.actions.githubusercontent.com token.actions.githubusercontent.com Info Assign role Delete Summary Provider Type Provider Creation Time ARN arn:aws:iam::730335542499:oidctoken.actions.githubusercontent.com May 14, 2024, 14:00 (UTC-07:00) OpenID Connect provider/token.actions.githubuser content.com **Endpoint verification** Audiences (1) Tags Audiences (1) Actions ▼ Also known as client ID, audience is a value that identifies the application that is registered with an OpenID Connect provider. Audience sts.amazonaws.com

Official provider name for GitHub

IAM Role - Trust relationship



```
IAM > Roles > github-oidc-provider-aws
```

Name of your Role

github-oidc-provider-aws Info

Role assumed by the GitHub OIDC provider.

16

Permissions Trust relationships Access Advisor Revoke sessions **Tags** (1) Trusted entities Edit trust policy Entities that can assume this role under specified conditions. 1 - { "Version": "2012-10-17", "Statement": [4 -"Effect": "Allow", "Principal": { "Federated": "arn:aws:iam::730335542499:oidc-provider/token.actions.githubusercontent.com" "Action": "sts:AssumeRoleWithWebIdentity", "Condition": { "StringLike": { 11 -Restrict to your 12 "token.actions.githubusercontent.com:sub" "repo:pas256/ensorcell:*" GitHub repository 13 14 15

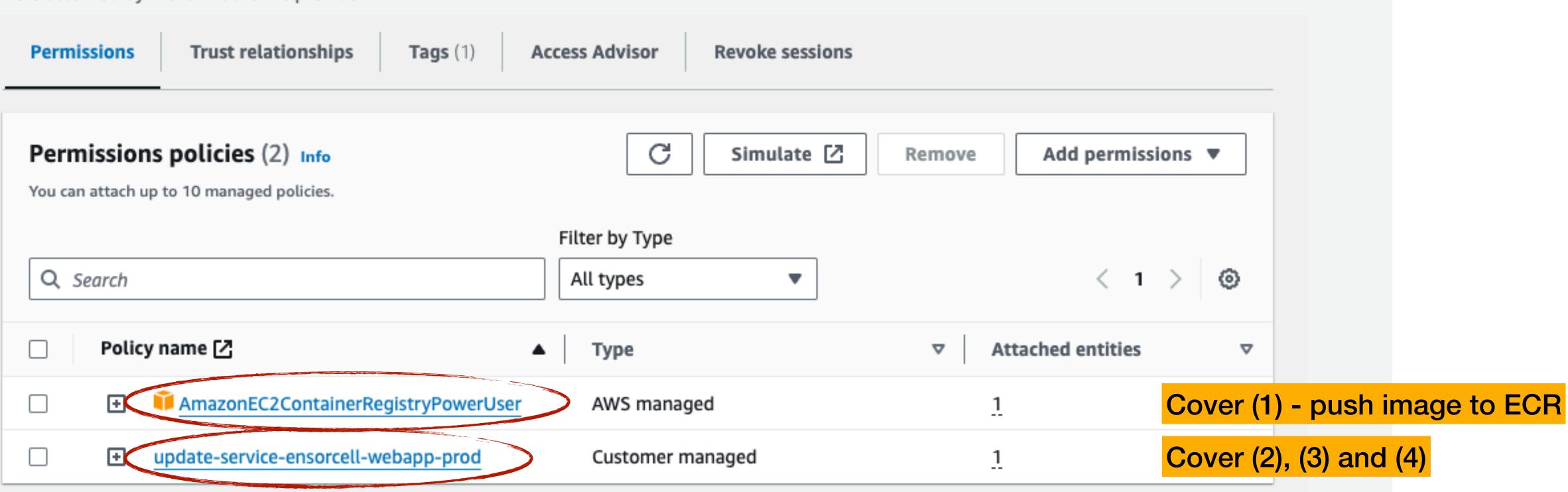
IAM Role - Permissions



IAM > Roles > github-oidc-provider-aws

github-oidc-provider-aws Info

Role assumed by the GitHub OIDC provider.



Terraform GitHub OIDC module



```
# GitHub OIDC permissions to push images to ECR
module "github-oidc" {
           = "terraform-module/github-oidc-provider/aws"
  version = "~> 2.2"
  create_oidc_provider
                             = true
  create_oidc_role
                             = true
                                                                            Your GitHub repository
  repositories
                             = ["pas256/ensorcell"]
                                                                            or restrict to main branch only (no PRs)
  # repositories
                             = ["pas256/ensorcell:ref:refs/heads/main"]
  oidc_role_attach_policies = |
                                                                      Use AWS's policy to cover (1)
    "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPowerUser",
    aws_iam_policy.update_service_policy.arn,
                                                                      Use our policy to cover the rest
resource "aws_iam_policy" "update_service_policy" {
               = "update-service-ensorcell-webapp-prod"
  name
  description = "Allows updating the ECS task definition and service"
```

@pas256

= jsonencode({ ... })

???

policy

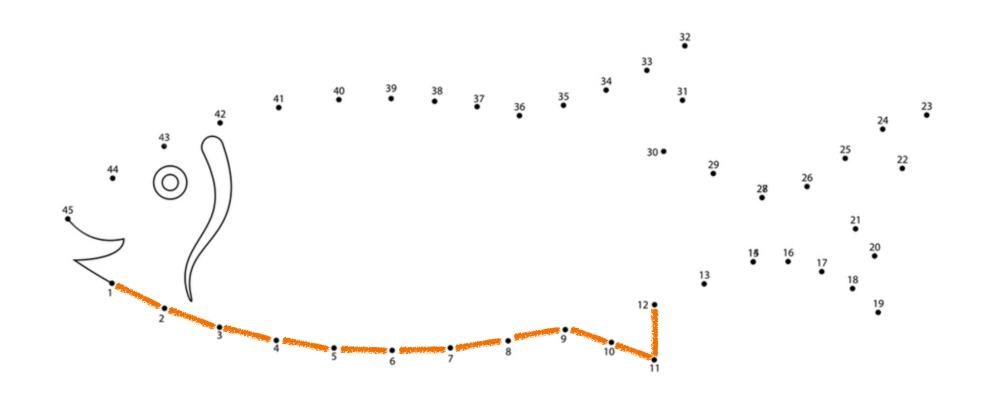
AWS API permissions required



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": [
                                        Covers (2)
         "ecs:DescribeTaskDefinition",
         "ecs:RegisterTaskDefinition"
                                         Covers (3)
      "Resource": "*"
   { ... },
   { ... }
```

Update Policy - Statement 2/3



```
"Version": "2012-10-17",
"Statement": [
   { ... },
      "Sid": "DeployService",
      "Effect": "Allow",
      "Action": [
         "ecs:DescribeServices",
                                   Covers (4) .... or does it?
         "ecs:UpdateService"
      "Resource": [
         "arn:aws:ecs:us-west-2:730335542499:service/my-ecs-cluster/my-prod-webapp",
         # Add more services here
                                                                           Service
                                                          Cluster
```

Task Definition for Service



```
"family": "my-prod-webapp",
                                                                                       The GitHub OIDC IAM Role
"taskRoleArn": "arn:aws:iam::730335542499:role/my-task-role",
                                                                                       needs to pass these roles
"executionRoleArn": "arn:aws:iam::730335542499:role/my-ecs-task-execution-role",
                                                                                       to the updated Service
"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": {
                                     e.g. Permission to log
       "logDriver": "awslogs",
                                         to CloudWatch Logs
        "options": {
            "awslogs-group": "/prod/ecs/ensorcell-webapp",
            "awslogs-region": "us-west-2",
            "awslogs-stream-prefix": "ecs"
```

Update Policy - Statement 3/3



```
"Version": "2012-10-17",
"Statement": [
   { ... },
   { ... },
                                             Covers the rest of (4)
      "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"
```

AWS API permissions required



1) Push image to ECR repository



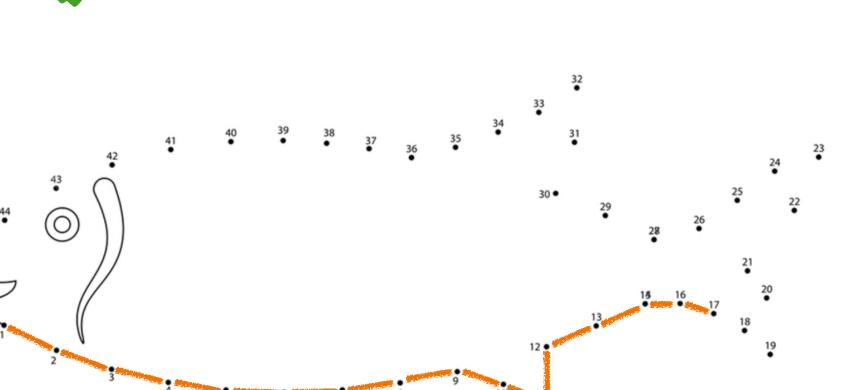
2) Read current ECS Task Definition the ECS Service is using



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



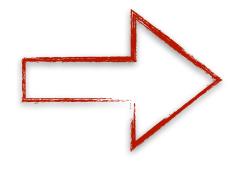
Amazon ECR > Private registry > Repositories > ensorcell/app > Permissions	
Permissions	Edit policy JSON Edit
Statements	
No statements You don't have any permission statements for this repository.	
Edit	

Nothing required here

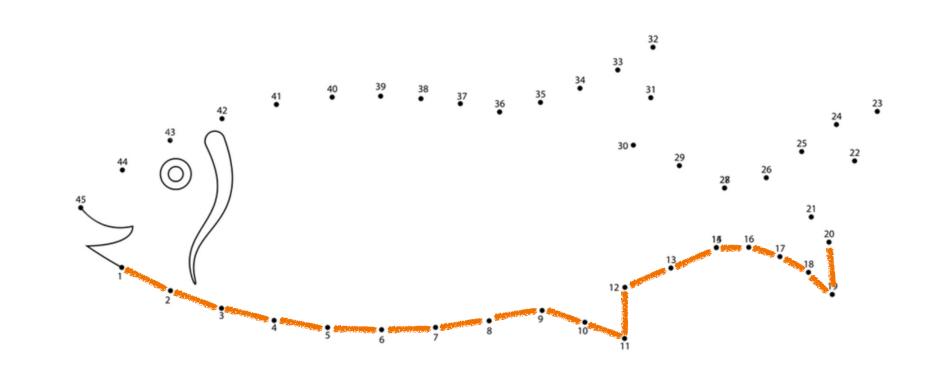
AWS side is done now



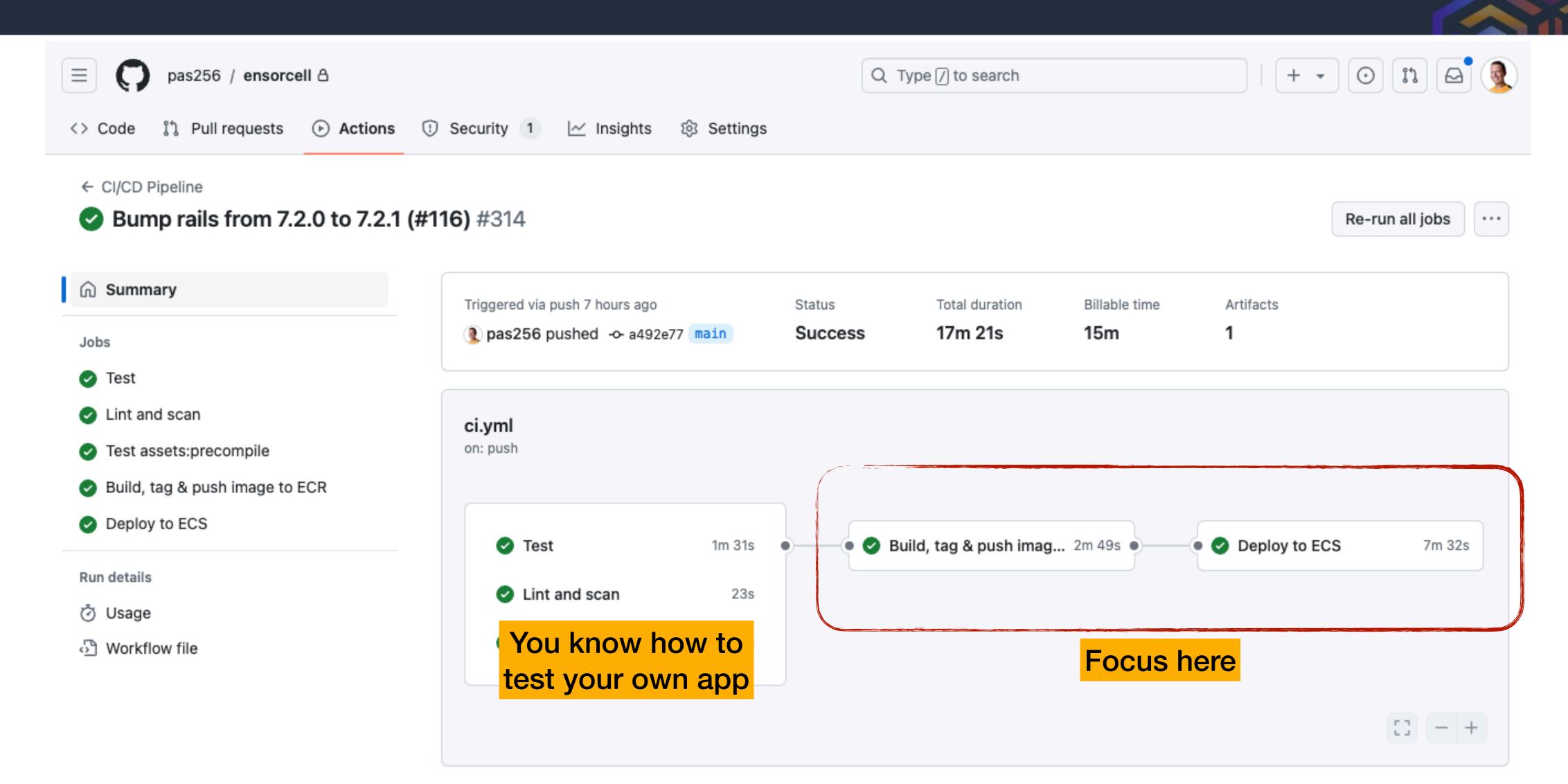








GitHub Workflow





GitHub Workflow Overview



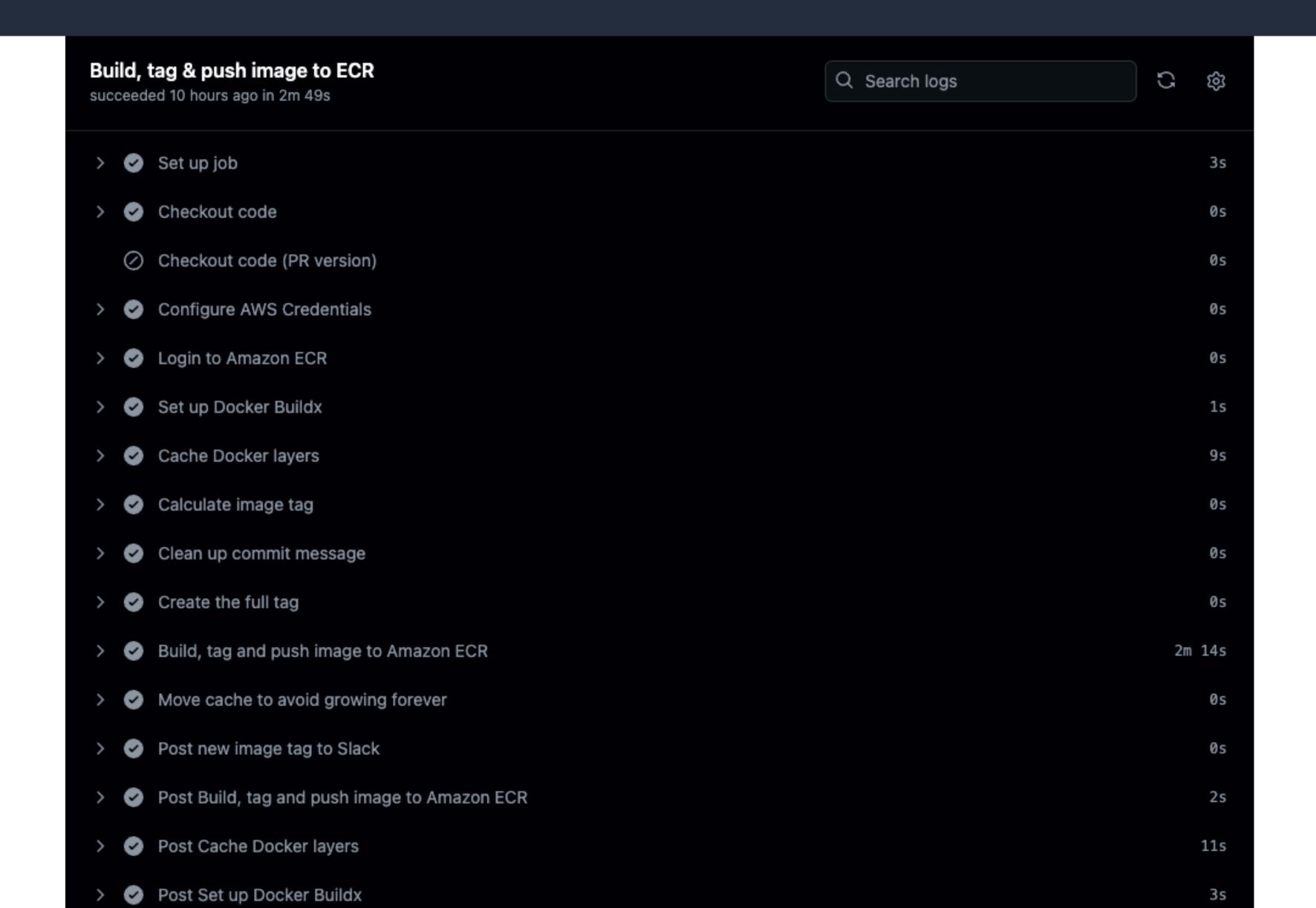
```
# This workflow will run the entire CI/CD pipeline
name: "CI/CD Pipeline"
on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]
jobs:
  test:
    name: Test
    . . .
  lint:
    name: Lint and scan
```

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

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

Job: Build, tag & push image to ECR





sooooo many steps

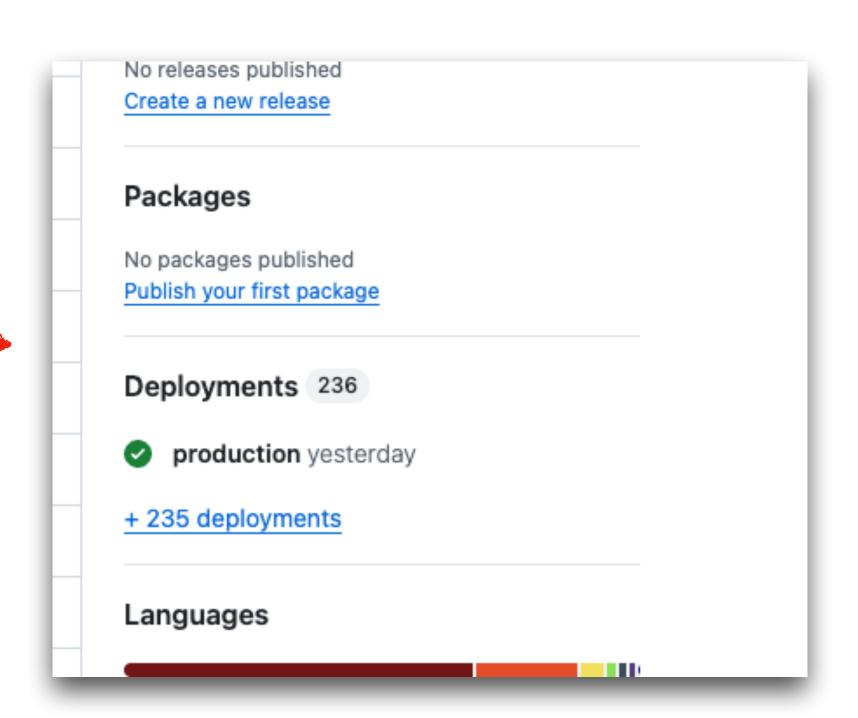


```
make_image:
   name: Build, tag & push image to ECR
   runs-on: ubuntu-latest
   environment: production

# 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



```
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
```

Make these available for the next job



```
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]'
                                           OIDC AWS Magic
 # 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
```

. . .

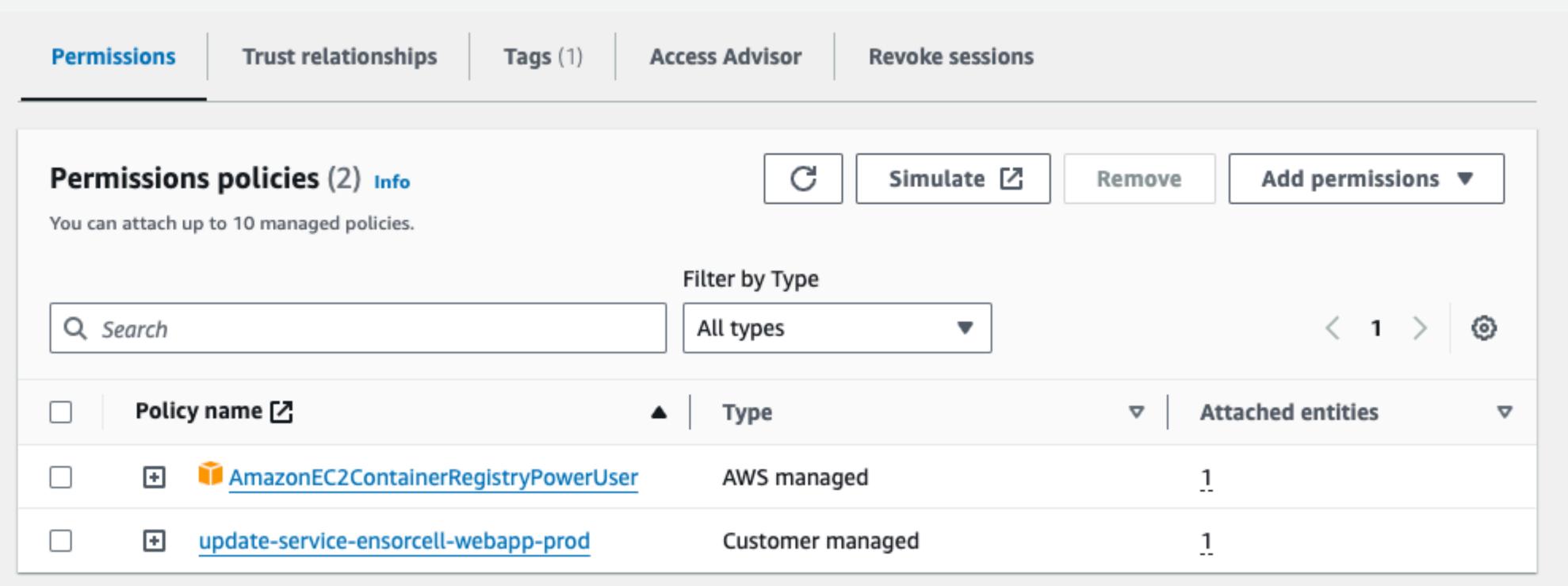
IAM Role - Permissions



IAM > Roles > github-oidc-provider-aws

github-oidc-provider-aws Info

Role assumed by the GitHub OIDC provider.





```
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]'
                                           OIDC AWS Magic
 # 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
```

. . .



```
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
        ...
```

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
```

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
                      BRANCH.YYYMMDD-HHMMSS.SHA0123.local
# - Local branch:
```

```
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
```

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
```

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 }}`"
```

Slack notification



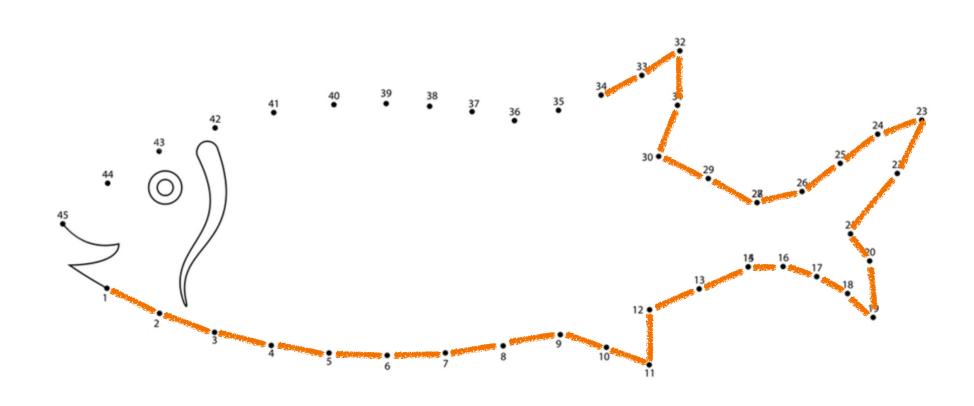


cell-builder APP 10:23 PM

GA Run: #278

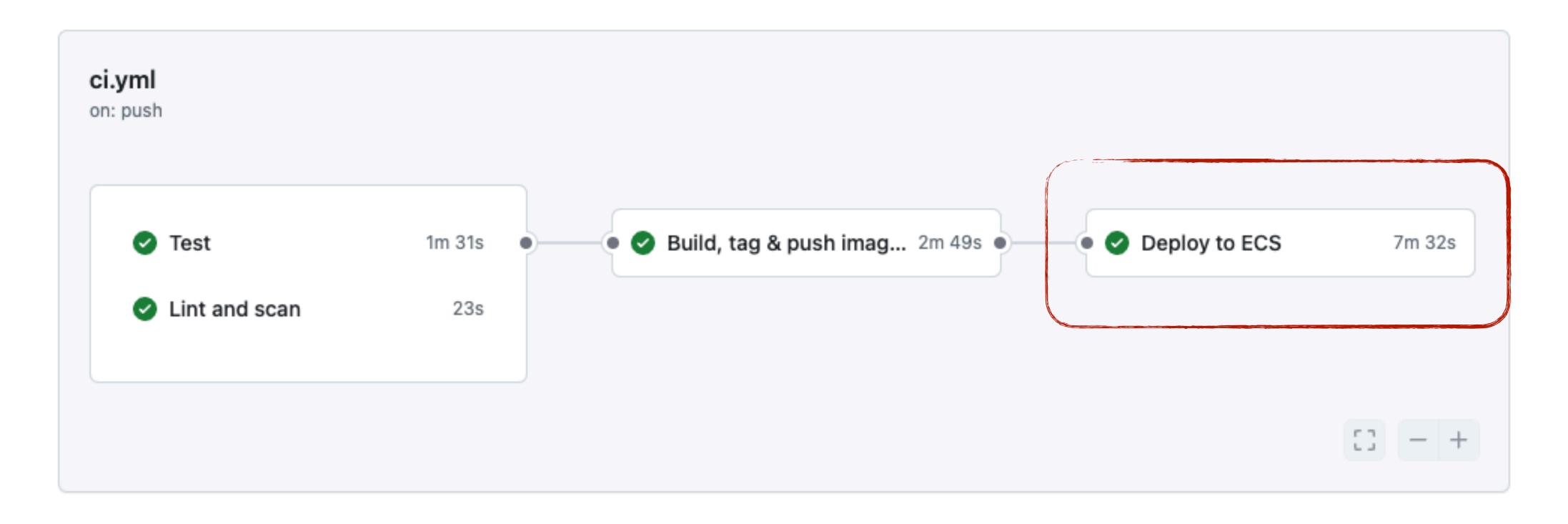
Bump sidekiq from 7.3.0 to 7.3.1 (#102)

New image: main.20240816-052053.d8069cc.ci



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
0	Update Slack with failure for workers		0s
0) Set flag for workers failure		0s
> @	Deploy Amazon ECS task definition webapp		3m 26s
> @	Format task definition for webapp		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.
- Deployment started. Watch this deployment's progress in the Amazon ECS console: https://us-west-2
 2.console.aws.amazon.com/ecs/v2/clusters/cell-prod-ecs-cluster/services/ensorcell-webapp-prod-workers/events?region=us-west-2">https://us-west-
 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



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: Configure AWS Credentials

steps:

- name: Checkout code

contents: read

id-token: write

uses: actions/checkout@v4



```
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

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:
  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

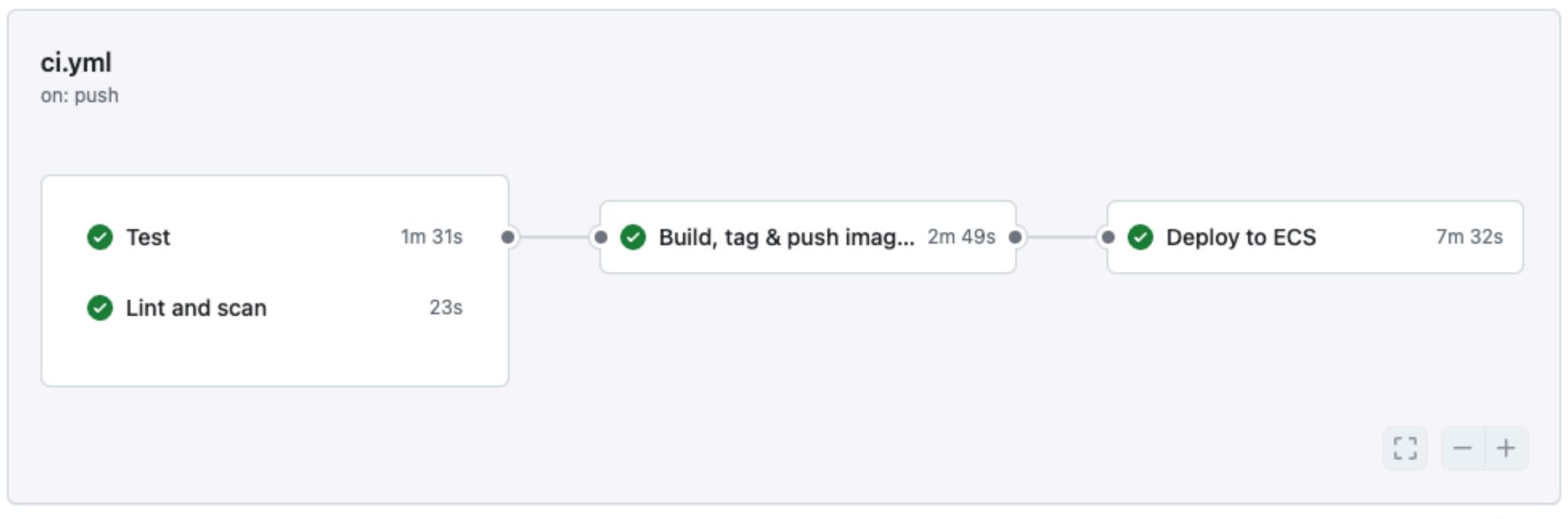
```
deploy:
  name: Deploy to ECS
  steps:
    . . .
    - name: Download task definition
      run:
        aws ecs describe-task-definition \
          --task-definition my-prod-webapp \
          --query taskDefinition \
          > /tmp/webapp-task-definition.json
    - name: Put new image ID in 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 }}
```

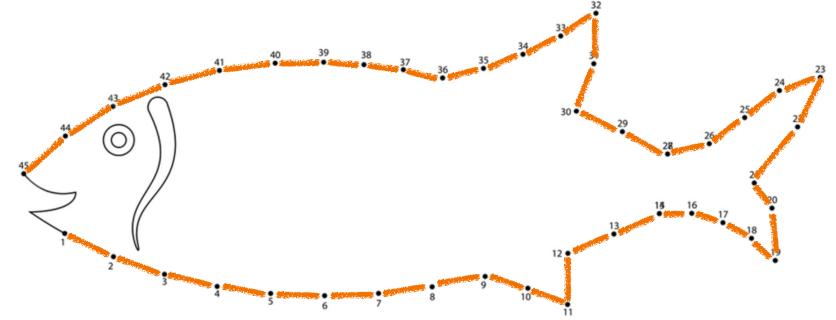
This is what causes those warnings

```
- 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
```

CI/CD Pipeline

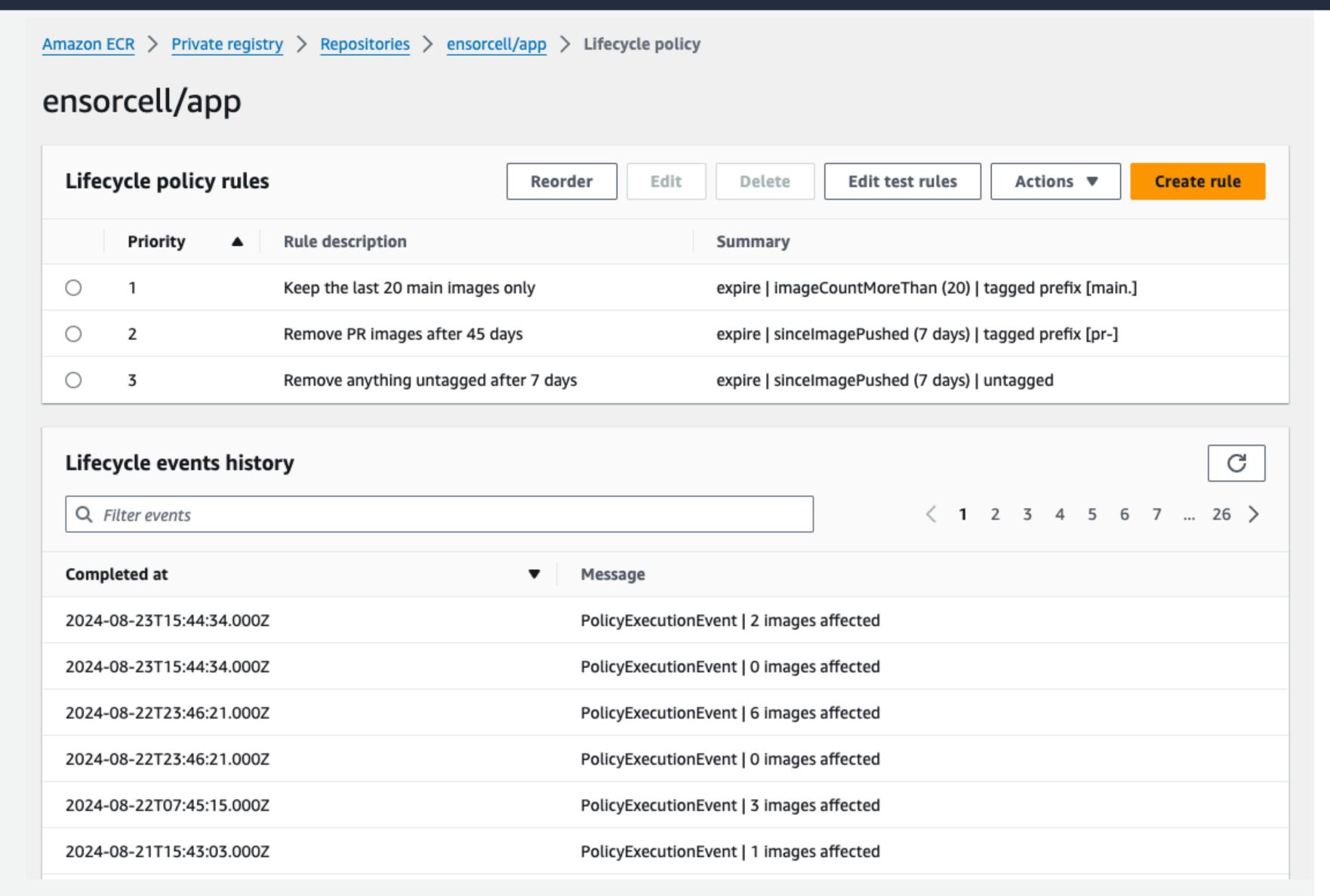






ECR Repository Lifecycle Rules





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

Resources

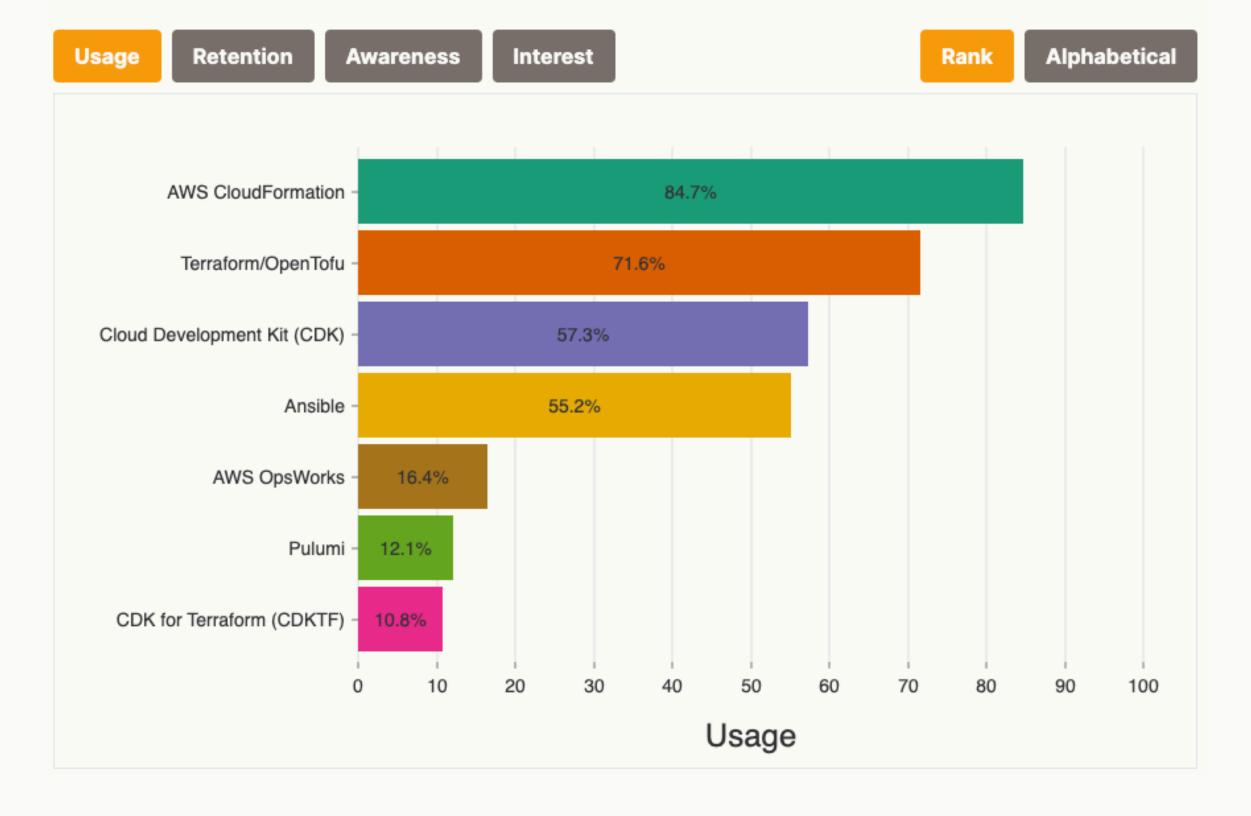




answersforaws.com

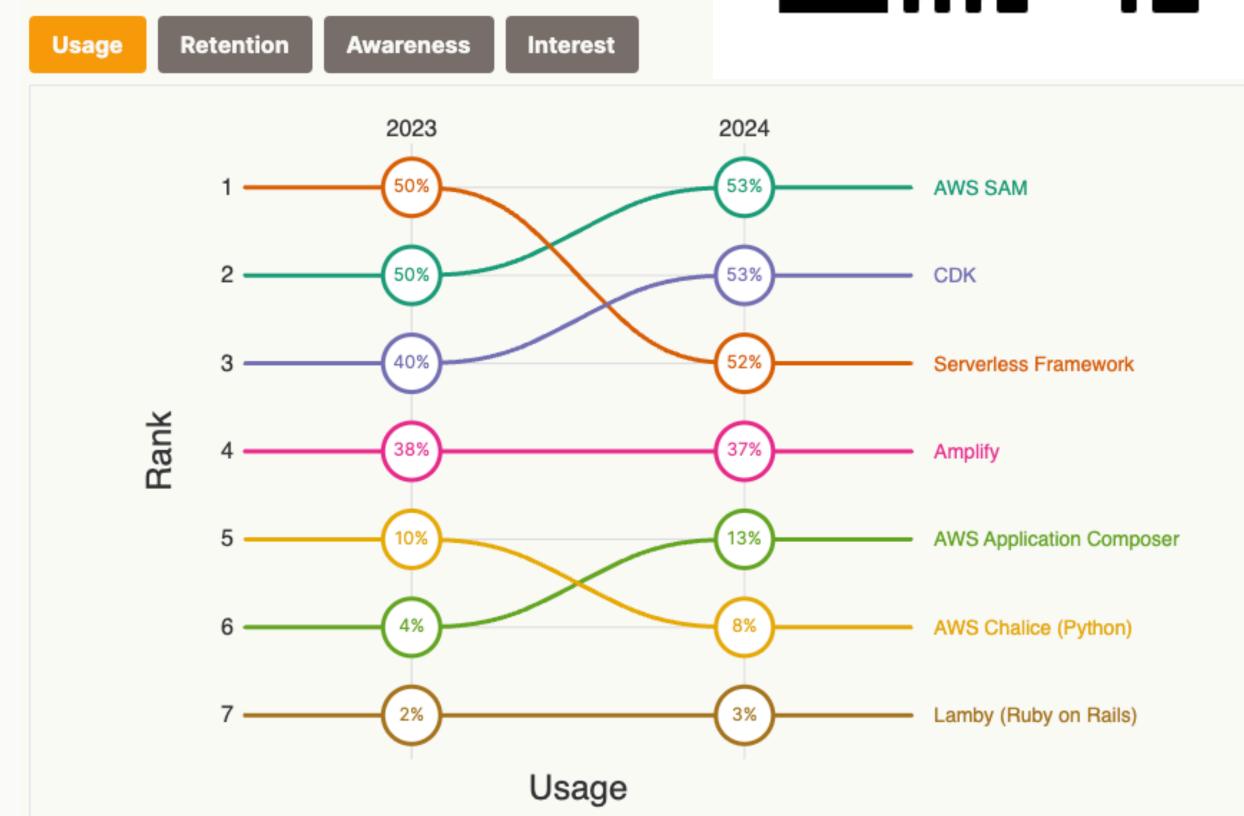
Infrastructure as Code

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



Serverless

Trend over the years





Meetup in SF

Advanced AWS Meetup

- Meets every month, usually in SF
- Next meetup: October 17



meetup.com/advancedaws





Thank you





