Deploy to Amazon ECS

How to deploy to ECS using Pipes (Recommended)

This option provides you with a simplified way of deploying to ECS. This approach requires less maintenance since the pipeline is maintained on your behalf. However, this option restricts the amount of control you have over the deployment.

Prerequisites

You'll need to have:

- An existing image registry such as Docker Hub or ECR. Alternatively, you can also use your own Docker registry.
- An existing AWS Elastic Container Service cluster running a service, which will be updated with the task definition in the repo.
- An AWS IAM user with programmatic access, with sufficient permissions to execute the *RegisterTaskDefinition* and *UpdateService* actions.

Steps

1. Clone AWS ECS deployment example repo

2. Configure your registry variables by going to **Pipelines** > **Settings** > **Repository variables**, and clicking **Add**.

Example:

This example uses Docker Hub variables:

- `DOCKERHUB_USERNAME`
- `DOCKERHUB_PASSWORD`

3. Add your AWS credentials by going to In Pipelines **Settings** > **Repository variables**, and clicking **Add**.

Example:

Amazon variables:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_DEFAULT_REGION`

Outcome: You can now reference these variables from within the bitbucket-pipelines.yml

4. Go to your bitbucket-pipelines.yml and edit the cluster and service name to match your existing ECS cluster and service.

Example:

```
- pipe: atlassian/aws-ecs-deploy:1.0.0
  variables:
    AWS_ACCESS_KEY_ID: $AWS_ACCESS_KEY_ID
    AWS_SECRET_ACCESS_KEY: $AWS_SECRET_ACCESS_KEY
    AWS_DEFAULT_REGION: $AWS_DEFAULT_REGION
    CLUSTER_NAME: 'aws-ecs-deploy-example'
    SERVICE_NAME: 'aws-ecs-deploy-example-service'
    TASK_DEFINITION: 'taskDefinition.json'
```

Outcome: Once the changes have been pushed, Pipelines builds the app, packages it to a Docker container, push to Docker Hub, and deploys the container to ECS.

Deploy to ECS using AWS CLI

This option is recommended for advanced scenarios where you need more control over the customization.

Quick start guide (recommended)

Prerequisites

You'll need to have:

- An existing image registry such as Docker Hub or ECR. Alternatively, you can also use your own Docker registry.
- An existing AWS Elastic Container Service cluster running a service, which will be updated with the task definition in the repo.
- An AWS IAM user with programmatic access, with sufficient permissions to execute the RegisterTaskDefinition and UpdateService actions.

Steps

1. Clone https://bitbucket.org/bitbucketpipelines/example-aws-ecs-deploy-no-pipe/src

2. Configure your registry variables by going to **Pipelines** > **Settings** > **Repository variables**, and clicking **add**.

3. Ensure you tick the *Secured* checkbox for the password.

Example:

This example uses Docker Hub variables:

- `DOCKERHUB_USERNAME`
- `DOCKERHUB_PASSWORD`

3. Add your AWS credentials by going to In Pipelines **Settings** > **Repository variables**, and clicking **add**.

Example:

Amazon variables:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_DEFAULT_REGION`

Outcome: You can now reference these variables from within the bitbucket-pipelines.yml

Step-by-step guide

Prerequisites

You'll need to have:

- An existing image registry such as Docker Hub or ECR. Alternatively, you can also use your own Docker registry.
- An existing AWS Elastic Container Service cluster running a service, which will be updated with the task definition in the repo.
- An AWS IAM user with programmatic access, with sufficient permissions to execute the *RegisterTaskDefinition* and *UpdateService* actions.
- Install the deploy tools. The deployment script requires the AWS CLI, and the jq library as part of your build. If you use a different Docker image or you've created your own, ensure these packages are installed.

Steps

1. Install the AWS CLI.

Example:

In the following example, you can see an image with the installed tools.

```
- apt-get update && apt-get install -y jq
- curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
- unzip awscli-bundle.zip
- ./awscli-bundle/install -b ~/bin/aws
- export PATH=~/bin:$PATH
```

2. Register an ECS task definition that references the newly pushed Docker image. When we register the task definition with our ECS cluster, we get back the version. We'll store this in an environment variable so we can reference it later when we update the ECS service.

Example:

```
# Replace the container name in the task definition with the new image.
- export IMAGE_NAME="${DOCKERHUB_USERNAME}/${BITBUCKET_REPO_SLUG}:${BITBUCKET_BUILD_NU
- envsubst < task-definition.json >  task-definition-envsubst.json
 # Update the task definition and capture the latest revision.
- >
export UPDATED_TASK_DEFINITION=$(aws ecs register-task-definition --cli-input-json fil
jq '.taskDefinition.taskDefinitionArn' --raw-output)
```

3. Update the ECS Service

Example:

```
- aws ecs update-service --service example-ecs-service --cluster example-ecs-cluster -
```

Was this helpful? Yes No Provide feedback about this article

Still need help?

The Atlassian Community is here for you.

Ask the Community

Access Pipelines deployment guides

Show more

Deploy to AWS EKS (Kubernetes)

Deploy a Lambda function update to AWS

Deploy to Amazon ECS

Deploy to Firebase

Deploy to Google Cloud

Show more

On this page

How to deploy to ECS using Pipes (Recommended)

Prerequisites

Steps

Deploy to ECS using AWS CLI

Quick start guide (recommended)

Prerequisites

Steps

Step-by-step guide

Steps

Community

Questions, discussions, and articles