

# Break a Monolith Application into Microservices

## Overview

In this tutorial, you will deploy a monolithic node.js application to a Docker container, then decouple the application into microservices without any downtime. The node.js application hosts a simple message board with threads and messages between users.

## Module 1 Containerize the Monolith

### Step 1 Get Setup

1. Have AWS account
2. Install Docker
3. Install the AWS CLI

after using `pip install awscli --upgrade --user`, use `aws --version` to verify. Shows aws command not found. Try another:

```
Successfully installed awscli-1.16.43
[YimatoMacBook-Pro:api yiqian$ aws
-bash: aws: command not found
[YimatoMacBook-Pro:api yiqian$ ~/.local/bin/aws --version
aws-cli/1.16.43 Python/3.6.3 Darwin/18.0.0 botocore/1.12.33
YimatoMacBook-Pro:api yiqian$
```

4. Have a Text Editor

### Step 2 Download & Open the Project

Download from GitHub

### Step 3 Provision a Repository

The  
address

repository

请确保您已安装 AWS CLI 和 Docker 的最新版本。了解更多信息，请参阅 [ECR 文档](#)。

1) 检索登录命令，对 Docker 客户端进行身份验证，以允许其访问您的注册表：

对于 macOS 或 Linux 系统，请使用 AWS CLI：

```
$(aws ecr get-login --no-include-email --region us-east-1)
```

对于 Windows 系统，请使用适用于 PowerShell 的 AWS 工具：

```
Invoke-Expression -Command (Get-ECRLoginCommand -Region us-east-1).Command
```

注意：如果您在使用 AWS CLI 时收到 "Unknown options: --no-include-email" 错误，请确保您已安装了最新版本。了解更多信息

2) 如果您使用的是 AWS CLI，请从步骤 1 的输出运行登录命令。

3) 使用以下命令生成 Docker 映像。有关从头开始生成 Docker 文件的信息，请参阅 [此处](#) 的说明。如果您已生成映像，则可跳过此步骤：

```
docker build -t api .
```

4) 生成完成后，标记您的映像，以便将映像推送到此存储库：

```
docker tag api:latest 747142172942.dkr.ecr.us-east-1.amazonaws.com/api:latest
```

5) 运行以下命令将此映像推送到您新创建的 AWS 存储库：

```
docker push 747142172942.dkr.ecr.us-east-1.amazonaws.com/api:latest
```

## Step 4 Build & Push the Docker Image

### 1. Configure your credentials

```
YimatoMacBook-Pro:api yiqian$ ~/.local/bin/aws configure
AWS Access Key ID [None]: AKIAJZITGLEDT7MLXEBQ
AWS Secret Access Key [None]: DEciI1lb+bA38MEB2W92WeChDc0WdiLbJVHqlQIm
Default region name [None]: us-east-2
Default output format [None]: json
YimatoMacBook-Pro:api yiqian$
```

```
Default output format [json]:
[YimatoMacBook-Pro:api yiqian$ ~/.local/bin/aws ecr get-login --no-include-email
--region us-west-2

An error occurred (UnrecognizedClientException) when calling the GetAuthorizationToken operation: The security token included in the request is invalid.
YimatoMacBook-Pro:api yiqian$ ~/.local/bin/aws ecr get-login --no-include-email
--region us-west-2
```

### Add json permission in User

策略摘要    {} JSON    编辑策略

```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "ecr:GetAuthorizationToken",
8                 "ecr:InitiateLayerUpload",
9                 "ecr:UploadLayerPart",
10                "ecr:CompleteLayerUpload",
11                "ecr:BatchCheckLayerAvailability",
12                "ecr:PutImage"
13            ],
14            "Resource": "*"
15        }
16    ]
17 }
```

```

Default region name [us-east-1]:
Default output format [json]:
YimatoMacBook-Pro:api yiqian$ ~/.local/bin/aws ecr get-login --no-include-email ]
--region us-east-1
docker login -u AWS -p eyJwYXlsb2FkIjoibEEZWhhbm8zczgyOFNpcFMwT3p1anJRQnBwWHpCN
WM4aHY3K3I2NGVSRNQTGF5U01yeTZWRRnZaaJjWVVPsB3RqMEJ20WM3QnkyVUZxWesySys1ZUwVHkva
XVQWlRPa09YQzJsRmtLWDRRRXNtQXI0b1FqelhuNmFMWnkrZ0RyME9xSEpTYUVFUmdpdErtMVNCFBIN
WR1eTA3ei9NV2hvRcTlUk1qb1MwRnE2UHEwRU5XN2pHQkZPKzM4VWt1QkFwM1BrBwc2cz1lZXNaRg9yT
mRRKzF3cnhxQVBaeVZUUUvK121ENDhGdFIxZEZYAUtVWkpsY1Z0VGNRl3hQN1BEeUdrU043aktwdDdNW
GFLbENZbwZVME9tdUpSqZJ0dHNBbGNPbkxiRG1UbXhNMDJ2ZkthSmpwWnZ6V1RHRjVJdFgzWGR3c0JJYe
EFKeEx2UG54R2dwbzUxSW54L0RcT0hpcFZGTlPFSnRQqjh0WUR5Y1hsK0VHNVA1NUVvOWwJcz1KWWEvM
3NiL29zaWfyWHk2RkZnNkF1VHg4ckpzOUphN01nZGxCUUF2dHREeGN0ck1JYVFadEZtW1dqRmZWmZyR
HdIZEZpbmxVv0pM0StSUDVnYzdWeGZBOW9ZSTVUY3RaQVhRNG9TSTRvVXR0bUVHRms4OFBZZG8wl1Q5V
GJnYXJYbUNHQTNUZDRUQ1VuaKpGVm1YdkoyWlh3SWRbubjFRdUNPKzIxbmwxbz16aHEwK0dxSXZIT2IrV
3U0V1hCelIdiNKNQ0y0yVZKZG0rb2EwRVhQWwKbCsyS1VLWCTjB3R0TFMR0HVBDUznZEVNQVpWMMWtVM
0doOUrtZk1wcm91by9Tjw1rRktXSEFUT3M2RjBXZHJCRRHpd0hQTDZzTmVzS1J1aEFKaTNwbVdCQy9Ca
U52QUVvG53T2JEN3hKMUhmVXkweH1HaDdYtjZMek1SZ1IzVzJrd0h2THZVdzJBRmtjUzF2dUk4Mm4yY
ThKamo0MzI2eGJZUeUyWUJ4cFAwVTRDS0Nqb3NuemZMOUJ6Znp0YWJQeVd6K3A3NG1ZR1g2M1NPRWpJm
mc1Uzgz0U0NWN0J3a0R0SkFbD0JPRGpSYVYVq3cUyaUt2MnZROE9IOEJtNi9GdmNqb1c1Y0xNWVJOK3I4R
WdXZkdSZ3FTbWNBbzJ6TytVv1d0OHVGKzBvc1JXM3VUd2JtbW1vQndnNEVMcTc0Rm9VK0lqUjZyY3QvK
1BmTFRDMXpBVStwNKRNSHNH1NFdON2p1U1N1aTBDM0tzTUV4dDdmcWJ6eEkyL21MSmM0ZDVCVjVBN3M3T
HFsek1NOD12SXdkcDZ0WFVMNEVWOGEEya0JQb1luWmwrVD1QSUZSK3RuZw5kM0hYeU5hM3RKbnFYsXBY
EJvQmdrcWhraUc5dzBCQndFd0hWwUpZSVpJQVdVREJBRXVnQkVFREZnQnNhRE15cFRnM1F1V1NBsUJFS
UE3TmozUXdleUZTQTVEMEJWS2RibyswTDgrWnNIEk1RR0N4Rkh5VFmVc3ZLYXZSemE5dHFik310b3dBd
nRuA2t1QVpzVjBmRldtQTNTLN3paz0iLCJZ2XJzaW9uIjoiMiIsInR5cGU0iOiJlEQVRBX0tFW5IsImV4c
GlyYXRpb24iOiE1NDA5M0c5MjV9 https://747142172942.dkr.ecr.us-east-1.amazonaws.com
YimatoMacBook-Pro:api yiqian$

```

## Build the Image

```

--> Running in f9fb44e5fdf2
Removing intermediate container f9fb44e5fdf2
--> c93442bccf77
Step 3/6 : ADD .
--> 4241cc41853
Step 4/6 : RUN npm install
--> Running in 18a402b2a3e7
/srv
+-- koa@1.6.2
+-- accepts@1.3.5
+-- mime-types@2.1.21 deduped
+-- negotiator@0.6.1
+-- co@4.6.0
+-- composition@2.3.0
+-- any-promise@1.3.0
+-- co@4.6.0 deduped
+-- content-disposition@0.5.2
+-- content-type@1.0.4
+-- cookies@0.7.2
+-- depd@1.1.2
+-- keygrip@1.0.3
+-- debug@2.6.9
+-- ms@2.0.0
+-- delegates@1.0.0
+-- destroy@1.0.4
+-- error-inject@1.0.0
+-- escape-html@1.0.3
+-- fresh@0.5.2
+-- http-assert@1.4.0
+-- deep-equal@1.0.1
+-- http-errors@1.7.1 deduped
+-- http-errors@1.7.1
+-- depd@1.1.2 deduped
+-- inherits@2.0.3
+-- setprototypeof@1.1.0
+-- statuses@1.5.0 deduped
+-- toidentifier@1.0.0
+-- koa-compose@2.5.1
+-- koa-is-json@1.0.0
+-- mime-types@2.1.21
+-- mime-db@1.37.0
+-- on-finished@2.3.0
+-- ee-first@1.1.1
+-- only@0.0.2
+-- parseurl@1.3.2
+-- statuses@1.5.0
+-- type-is@1.6.16
+-- media-typer@0.3.0
+-- mime-types@2.1.21 deduped
+-- vary@1.1.2
+-- koa-router@5.4.2
+-- co@4.6.0 deduped
+-- debug@2.6.9 deduped
+-- http-errors@1.7.1 deduped
+-- methods@1.1.2
+-- path-to-regexp@1.7.0
+-- isarray@0.0.1

npm WARN srvc No description
npm WARN srvc No repository field.
npm WARN srvc No license field.
Removing intermediate container 18a402b2a3e7
--> 7d7a57ed0667
Step 6/6 : EXPOSE 3000
--> Running in 709d014fae5f
Removing intermediate container 709d014fae5f
--> 4fa1d03be549
Step 6/6 : CMD ["node", "server.js"]
--> Running in ea5d83cb004a
Removing intermediate container ea5d83cb004a
--> c50a8e5642fc
Successfully built c50a8e5642fc
Successfully tagged apilatest

```

Push the image to ECR:

```
The push refers to repository [747142172942.dkr.ecr.us-east-1.amazonaws.com/api]
002b83738171: Layer already exists
d1c9a2ba6d79: Layer already exists
3e893534526a: Layer already exists
040fd7841192: Layer already exists
latest: digest: sha256:6e3fdd2228a35326b24f148f697739d72b61675743bc85de79c28526918795b5 size: 1158
YimatoMacBook-Pro:api yiqian$
```

## 存储库

创建存储库

删除存储库

上次更新于 2018年10月29日 下午10:37:11

< 1 - 1 of 1 > Page

Filter by repository name

<input type="checkbox"/>	存储库名称 ▲	存储库 URI ▼	创建时间 ▼
<input type="checkbox"/>	api	747142172942.dkr.ecr.us-east-1.amazonaws.co...	2018-10-29 20:24:13 -0400

## Module 2 Deploy the Monolith

Step 1 Launch an ECS Cluster using AWS CloudFormation

Create stack

<input type="checkbox"/>	BreakTheMonolith-Demo	2018-10-29 22:41:30 UTC-0400	CREATE_IN_PROGRESS
--------------------------	-----------------------	------------------------------	--------------------

Step 2 Check your Cluster is Running

Cluster

### Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you can run task requests. Each account receives a default cluster the first time you use the Amazon ECS service. Clusters may contain more than one Amazon EC2 Instance type.

For more information, see the [ECS documentation](#).

Create Cluster

Get Started

View

list

card

view all

< 1 - 1 of 1 >

BreakTheMonolith-Demo-ECSCluster-1VMWX0XCN4XN1 >

FARGATE

0

Services

0

Running tasks

0

Pending tasks

EC2

0

Services

0

Running tasks

0

Pending tasks

0.00%

CPUUtilization

0.00%

MemoryUtilization

2

Container instances

## 2 EC2 instances

Status: <b>ALL</b> ACTIVE DRAINING							
< 1-2 > Page size 50							
Filter by attributes (click or press down arrow to view filter options)							
<input type="checkbox"/>	Container Instance	EC2 Instance	Availability Zo...	Agent Connec...	Status	Running tasks...	CPU available
<input type="checkbox"/>	84be05c4-e1d8-4cc7-bc5...	i-0d01ee0fc314...	us-east-1b	true	ACTIVE	0	1024
<input type="checkbox"/>	aaaf736a-d1b2-41e5-b569...	i-05723a383f5d...	us-east-1a	true	ACTIVE	0	1024

## Step 3 Write a Task Definition

### Create task definition

<input type="checkbox"/>	Task Definition	Latest revision status
<input type="checkbox"/>	api	ACTIVE

## Step 4 Configure the Application Load Balancer: Target Group

### Configure the ALB Target Group

Create target groupActions ▾

Filter by tags and attributes or search by keyword

<input checked="" type="checkbox"/>	Name	Port	Protocol	Target type	Load Balanc	VPC ID
<input checked="" type="checkbox"/>	api	80	HTTP	instance		vpc-7bcb2403

## Step 5 Configure the Application Load Balancer: Listener

### Add listener

Description

Listeners

Monitoring

Tags

A listener checks for connection requests using its configured protocol and port, and the load balancer uses remove, or update listeners and listener rules.

Add listener

Edit

Delete

<input type="checkbox"/>	Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/>	HTTP : 80 arn...206d89255bfc090c ▾	N/A	N/A	Default: forwarding to <a href="#">api2</a> <a href="#">View/edit rules</a>

Step 6 Deploy the Monolith as a Service

Create Service

Service : api

Cluster

BreakTheMonolith-Demo-ECSCluster-1VMWX0XCN4XN1

Status

ACTIVE

Task definition

api:1

Service type

REPLICA

Launch type

FARGATE

Platform version

LATEST(1.2.0)

Service role

aws-service-role/ecs.amazonaws.com/AWSServiceRoleForECS

Desired count

1

Pending count

0

Running count

0

Details

Tasks

Events

Auto Scaling

Deployments

Metrics

Logs

Step 7 Test your Monolith

Filter by tags and attributes or search by keyword							1 to 1 of 1	
Name	DNS name	State	VPC ID	Availability Zones	Type			
demo	demo-1765946018.us-east-1...	active	vpc-041dfbbe30a4bdc2	us-east-1a, us-east-1b	application			

Part 3 Break the Monolith

Step 1 Provision the ECR Repositories

Create repository

Filter by repository name			
<input type="checkbox"/>	Repository name	Repository URI	Created at
<input type="checkbox"/>	api	747142172942.dkr.ecr.us-east-1.amazonaws.co...	2018-10-29 20:24:13 -0400
<input type="checkbox"/>	posts	747142172942.dkr.ecr.us-east-1.amazonaws.co...	2018-10-29 23:21:02 -0400
<input type="checkbox"/>	threads	747142172942.dkr.ecr.us-east-1.amazonaws.co...	2018-10-29 23:20:49 -0400
<input type="checkbox"/>	users	747142172942.dkr.ecr.us-east-1.amazonaws.co...	2018-10-29 23:20:04 -0400



## Step 3 Build and Push Images for Each Service

Build images tag then push

### < All repositories : posts

Repository ARN arn:aws:ecr:us-east-1:747142172942:repository/posts

Repository URI 747142172942.dkr.ecr.us-east-1.amazonaws.com/posts

View Push Commands

Images

Permissions

Dry run of lifecycle rules

Lifecycle policy

Amazon ECR [limits](#) the number of images to 1,000 per repository. [Request a limit increase](#).

Image sizes may appear compressed. [Learn more](#)

Delete

Last updated on October 29, 2018 11:28:35 PM (0m ago)



Filter in this page

< 1-1 >

Page size

100

<input type="checkbox"/>	Image tags	Digest	Size (MiB)	Pushed at
<input type="checkbox"/>	v1	<a href="#">view all</a> sha256:46e515a9ecfedc25db6f7453990abf2075...	19.18	2018-10-29 23:28:24 -0400

### < All repositories : threads

Repository ARN arn:aws:ecr:us-east-1:747142172942:repository/threads

Repository URI 747142172942.dkr.ecr.us-east-1.amazonaws.com/threads

View Push Commands

Images

Permissions

Dry run of lifecycle rules

Lifecycle policy

Amazon ECR [limits](#) the number of images to 1,000 per repository. [Request a limit increase](#).

Image sizes may appear compressed. [Learn more](#)

Delete

Last updated on October 29, 2018 11:29:37 PM (0m ago)



Filter in this page

< 1-1 >

Page size

100

<input type="checkbox"/>	Image tags	Digest	Size (MiB)	Pushed at
<input type="checkbox"/>	v1	<a href="#">view all</a> sha256:b899a2e34b94f2105cd84a1197d5ece2a...	19.18	2018-10-29 23:29:32 -0400

## < All repositories : users

Repository ARN    arn:aws:ecr:us-east-1:747142172942:repository/users

Repository URI    747142172942.dkr.ecr.us-east-1.amazonaws.com/users

[View Push Commands](#)

**Images**    Permissions    Dry run of lifecycle rules    Lifecycle policy

Amazon ECR [limits](#) the number of images to 1,000 per repository. [Request a limit increase](#).

Image sizes may appear compressed. [Learn more](#)

[Delete](#)

Last updated on October 29, 2018 11:30:25 PM (0m ago)



Filter in this page

< 1-1 > Page size 100

<input type="checkbox"/>	Image tags	Digest	Size (MiB)	Pushed at
<input type="checkbox"/>	v1	sha256:54b052f169a3975cbf8669734671f4cdd3...	19.18	2018-10-29 23:30:25 -0400

## Part 4 Deploy Microservices

### Step 1 Write Task Definition for your Services

<input type="checkbox"/>	api	ACTIVE
<input type="checkbox"/>	posts	ACTIVE
<input type="checkbox"/>	threads	ACTIVE
<input type="checkbox"/>	users	ACTIVE

### Step 2 Configure the Application Load Balancer: Target Groups

<input type="checkbox"/>	Name	Port	Protocol	Target type	Load Balanc	VPC ID
<input type="checkbox"/>	api	80	HTTP	instance		vpc-7bcb2403
<input type="checkbox"/>	api2	80	HTTP	instance	demo	vpc-041dfbbe30a4bdc2
<input type="checkbox"/>	posts	80	HTTP	instance		vpc-7bcb2403
<input type="checkbox"/>	threads	80	HTTP	instance		vpc-7bcb2403
<input type="checkbox"/>	users	80	HTTP	instance		vpc-7bcb2403

### Step 3 Configure Listener Rules



## Step 4 Create Services

Cluster : BreakTheMonolith-Demo-ECSCluster-1VMWX0XCN4XN1

Delete Cluster

Get a detailed view of the resources on your cluster.

Status

ACTIVE

Registered container instances

2

Pending tasks count

1 Fargate, 0 EC2

Running tasks count

2 Fargate, 0 EC2

Active service count

4 Fargate, 0 EC2

Draining service count

0 Fargate, 0 EC2

Services

Tasks

ECS Instances

Metrics

Scheduled Tasks

Create

Update

Delete

Last updated on October 29, 2018 11:43:09 PM (0m ago)

Filter in this page

Launch type

ALL

Service type

ALL

< 1-4 >

<input type="checkbox"/>	Service Name	Status	Service ...	Task De...	Desired ...	Running...	Launch ...	Platfor...
<input type="checkbox"/>	<a href="#">posts</a>	ACTIVE	REPLICA	<a href="#">posts:1</a>	1	1	FARGATE	LATEST(...
<input type="checkbox"/>	<a href="#">threads</a>	ACTIVE	REPLICA	<a href="#">threads:1</a>	1	0	FARGATE	LATEST(...
<input type="checkbox"/>	<a href="#">users</a>	ACTIVE	REPLICA	<a href="#">users:1</a>	1	0	FARGATE	LATEST(...
<input type="checkbox"/>	<a href="#">api</a>	ACTIVE	REPLICA	<a href="#">api:1</a>	1	1	FARGATE	LATEST(...

## Step 6 Validate your Deployment

```
{"id":3,"title":"In search of a new guitar","createdBy":1}
```