

Configurando um *cluster* de instâncias EC2 que irão disponibilizar a API a partir de uma imagem Docker.

ECR (Amazon Elastic Container Registry) - A plataforma da AWS disponibiliza um serviço de repositório próprio para armazenamento de imagens, onde é possível hospedar a imagem Docker de uma aplicação.

- O primeiro passo é **criar o repositório**, escolhendo entre as opções privado ou público, e então nomeá-lo:

Amazon ECR > Repositórios > Criar repositório

Criar repositório

Configurações gerais

Configurações de visibilidade | [Informações](#)

Escolha a configuração de visibilidade para o repositório.

☒ **Privado**
O acesso é gerenciado pelas permissões da política do IAM e do repositório.

☐ **Público**
Visível publicamente e acessível para extrações de imagens.

Nome do repositório
Forneça um nome conciso. Um desenvolvedor deve ser capaz de identificar o conteúdo do repositório pelo nome.

036531232259.dkr.ecr.us-east-1.amazonaws.com/

7 de no máximo 256 caracteres (mínimo de 2). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Imutabilidade de etiquetas | [Informações](#)

Ative a imutabilidade de etiquetas para evitar que as etiquetas de imagem sejam substituídas por envios de imagens por push subsequentes usando a mesma etiqueta. Desative o recurso para permitir que as etiquetas de imagem sejam substituídas.

☐ **Desabilitado**

Depois que um repositório é criado, a configuração de visibilidade do repositório não pode ser alterada.

Configurações de verificação de imagem

Aviso de obsolescência
A configuração do ScanOnPush no nível do repositório tornou-se obsoleta em favor dos filtros de verificação no nível do registro.

Verificar por push
Habilite a verificação por push para que cada imagem seja verificada automaticamente após ser enviada para um repositório. Se estiver desabilitada, cada verificação de imagem deve ser iniciada manualmente para obter os resultados da verificação.

☐ **Desabilitado**

Configurações de criptografia

Criptografia do KMS
Você pode usar o AWS Key Management Service (KMS) para criptografar imagens armazenadas nesse repositório, em vez de usar as configurações de criptografia padrão.

☐ **Desabilitado**

Não é possível alterar ou desabilitar as configurações de criptografia do KMS após a criação do repositório.

[Cancelar](#) [Criar repositório](#)

- A seguir, nas definições de permissão, selecione “anexar políticas diretamente”, busque e marque a opção “AmazonEC2ContainerRegistryFullAccess”. Isto irá garantir que você tenha as permissões necessárias para a interação com o ECR:

Adicione usuário a um grupo existente ou crie um novo. Usar grupos é uma prática recomendada para gerenciar as permissões do usuário por funções de trabalho. [Saiba mais](#)

Para configurar o acesso pelo terminal de sua máquina, proceda como mostrado neste [video](#). **As chaves de acesso garantem a segurança de sua conta!** Tenha cuidado ao compartilhá-la, pois quem tiver suas chaves terá livre acesso aos serviços AWS da sua conta.

Observação: Caso já tenho um usuário IAM previamente registrado, certifique-se de que ele possui habilitada a política de acesso “AmazonEC2ContainerRegistryFullAccess”.

- Voltando ao ECR, para **hospedar a imagem no repositório**, é necessário executar comandos *push* no terminal local. Após criado repositório, ao clicar em “comandos push para” são mostrados os comandos que devem ser executados localmente para que a imagem seja enviada para o repositório no ECR:

Comandos push para sprint3-compass

macOS / Linux

Windows

Verifique se você tem a versão mais recente do AWS CLI e do Docker instalada. Para obter mais informações, consulte [Primeiros passos com o Amazon ECR](#).

Use as etapas a seguir para autenticar e enviar uma Imagem para o repositório. Para obter outros métodos de autenticação do registro, incluindo o auxílio de credenciais do Amazon ECR, consulte [Autenticação do registro](#).

1. Recupere um token de autenticação e autentique seu cliente Docker em seu registro.
Use o AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 036531232259.dkr.ecr.us-east-1.amazonaws.com
```

Observação: se você receber um erro ao usar o AWS CLI, verifique se você tem a versão mais recente do AWS CLI e do Docker instalada.
2. Crie sua imagem do Docker usando o comando a seguir. Para obter informações sobre como criar um arquivo Docker do zero, consulte as instruções [aqui](#). Você pode pular esta etapa se sua imagem já foi criada.

```
docker build -t sprint3-compass .
```
3. Depois que a compilação for concluída, marque sua imagem para que você possa enviá-la para este repositório:

```
docker tag sprint3-compass:latest 036531232259.dkr.ecr.us-east-1.amazonaws.com/sprint3-compass:latest
```
4. Execute o seguinte comando para enviar essa imagem para o repositório da AWS recém-criado:

```
docker push 036531232259.dkr.ecr.us-east-1.amazonaws.com/sprint3-compass:latest
```

Fechar

Observação: Para ações futuras é possível criar executáveis `.bat` ou `.sh`, para em caso de atualização dos códigos, a imagem no repositório atualiza automaticamente.

- Copiando os comando de push de seu repositório para um arquivo, é possível definir um script que irá facilitar o processo de envio e atualização da imagem no ECR:
 - No Windows, copie os comandos para o arquivo *script.bat*, na mesma pasta de seu *Dockefile*, e dê as permissões necessários para que seja executado.
 - No Linux, copie os comandos para um arquivo *script.sh* e execute: `chmod +x script.sh`

Executando o script, o resultado no terminal será:

```
b3rqb3r:~/Desktop/GIT/sprint-3-pb-aws-univesp$ chmod +x build.sh
b3rqb3r:~/Desktop/GIT/sprint-3-pb-aws-univesp$ ./build.sh
WARNING: Your password will be stored unencrypted in /home/b3r/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[*] Building 7.4s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 503B
=> [internal] load .dockerignore
=> => transferring context: 53B
=> [internal] load metadata for docker.io/library/node:14-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 252.13kB
=> [1/5] FROM docker.io/library/node:14-alpine@sha256:1d02d4073d7dfd58950a222a862f8b819afd774560b7b3e992b27ff4cd088285
=> => resolve docker.io/library/node:14-alpine@sha256:1d02d4073d7dfd58950a222a862f8b819afd774560b7b3e992b27ff4cd088285
=> => sha256:4c2996f1f84f2446434d1992c06e345a4ffa0dc16ede9b24ee18d9150ec967a4 1.16kB / 1.16kB
=> => sha256:4882e63f80a5867d9dc04815d5b27dc2f5a6b50fbcf22289a820f5a9698920ea 6.44kB / 6.44kB
=> => sha256:1d02d4073d7dfd58950a222a862f8b819afd774560b7b3e992b27ff4cd088285 1.43kB / 1.43kB
=> [2/5] WORKDIR /src
=> [3/5] COPY package*.json ./
=> [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:0ed90017ceb5a98a633b5249cbd9c49021de0044c47565e7051d985c702e30b6
=> => naming to docker.io/library/sprint3-compass
The push refers to repository [036531232259.dkr.ecr.us-east-1.amazonaws.com/sprint3-compass]
9c02a0434adb: Pushed
8a6a6aa201b6: Pushed
e8c598bb79a: Pushed
b678ff28d57d: Pushed
a7a37837ed5c: Pushed
e0f3c9fac6a6: Pushed
c9ced64ad27: Pushed
7cd52847ad77: Pushed
latest: digest: sha256:bb052952510e11bfea939ce6fcb9ac38e1d6383cd88b34f921d82df458649b size: 1993
b3rqb3r:~/Desktop/GIT/sprint-3-pb-aws-univesp$
```

- Com a imagem hospedada no ECR, agora podemos criar o cluster usando o serviço ECS. No ECS, clique em *cluster*, e depois em *adicionar cluster*. Dê um nome ao cluster, selecione a VPC, subnets. Em seguida, selecione o modelo (tipo) de cluster:

Criar cluster

Etapa 1: Selecionar modelo de cluster

Etapa 2: Configurar cluster

Selecionar modelo de cluster

Os modelos de cluster a seguir estão disponíveis para simplificar a criação do cluster. Outras configurações e integrações podem ser adicionadas posteriormente.

Somente redes ⓘ

Recursos a serem criados:

- Cluster
- VPC (opcional)
- Sub-redes (opcional)

ⓘ Para uso com capacidade de instância do AWS Fargate (Windows/Linux) ou externa.

EC2 Linux + redes

Recursos a serem criados:

- Cluster
- VPC
- Sub-redes

Grupo do Auto Scaling com AMI do Linux

EC2 Windows + redes

Recursos a serem criados:

- Cluster
- VPC
- Sub-redes

Grupo do Auto Scaling com AMI do Windows

*Obrigatório

Cancelar

Próxima etapa

- Selecione o tipo de imagem do S.O., o tipo de instância, e informe quantas máquinas o *cluster* deverá ter. Pela imagem, o cluster deverá ter no mínimo 1 e no máximo 2 instâncias, e o ECS irá gerenciar de forma a garantir que a quantidade desejada de instâncias estejam rodando:

☒ **Instâncias do Amazon EC2**
Configurações manuais. Use para grandes workloads com demandas de recursos consistentes.

Grupo do Auto Scaling (ASG) | [Informações](#)
Use grupos do Auto Scaling para escalar as instâncias do Amazon EC2 no cluster.

Sistema operacional/arquitetura
Escolha o sistema operacional Windows ou a arquitetura Linux para a instância.

Tipo de instância do EC2
Escolha de acordo com as workloads que você planeja executar neste cluster.

Capacidade desejada
Especifique o número de instâncias a serem executadas no cluster.

Mínimo **Máximo**

Par de chaves SSH
Crie um par de chaves no console do EC2 composto por uma chave privada e uma chave pública, que serão usadas para comprovar sua identidade ao se conectar a uma instância.

- Cluster ECS inicializado:

Status de inicialização

As instâncias de contêiner estão sendo iniciadas, e pode levar alguns minutos até que elas estejam no estado de execução e prontas para acesso. As horas de uso das novas instâncias de contêiner começam imediatamente e continuam a acumular até que você as interrompa ou encerre.

[Voltar](#) [Exibir cluster](#)

Status do ECS: 3 de 3 concluído **Cluster-Sprint3**

Cluster do ECS
Cluster do ECS criado Cluster-Sprint3 corretamente

Política do IAM de instâncias do ECS
A política do IAM para o perfil ecsInstanceRole foi anexada

Pilha do CloudFormation
A pilha do CloudFormation EC2ContainerService-Cluster-Sprint3 e seus recursos foram criados

Recursos do cluster

Tipo de instância	t2.micro
Número desejado de instâncias	1
Par de chaves	b3r_sprint2
ID da AMI do ECS	ami-083cd4eb32643c8a0
VPC	vpc-0377035793db42e9e
Sub-redes	subnet-05b44d02da9db82ed
Zonas de disponibilidade da VPC	us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, us-east-1f
Grupo de segurança	sg-0cbcd0bb29b6f77bd
Configuração de inicialização	EC2ContainerService-Cluster-Sprint3-EcsInstanceLc-DIA2EwV09tcl
Grupo do Auto Scaling	EC2ContainerService-Cluster-Sprint3-EcsInstanceAsg-1HVQV8G3J3Y4E

- Criado o cluster, agora podemos criar a definição da tarefa (*task definition*), que irá direcionar o ECS para qual *cluster* e imagem Docker utilizar. Na interface do ECS, clique em *Definição de tarefas*, depois em *Criar nova tarefa* e escolha o tipo de instância que seu *cluster* terá:

Criar nova definição da tarefa

Etapa 1: Selecionar compatibilidade com o tipo de inicialização

Etapa 2: Configurar definições de tarefa e contêiner

Selecionar compatibilidade com o tipo de inicialização

Selecione com qual tipo de inicialização você deseja que a definição da tarefa seja compatível, de acordo com o local em que deseja iniciar a tarefa.

FARGATE



Preço com base no tamanho da tarefa
Requer o modo de rede awsvpc
Infraestrutura gerenciada pela AWS, sem instâncias do Amazon EC2 para gerenciar

EC2



Preço com base no uso de recursos
Vários modos de rede disponíveis
Infraestrutura autogerenciada que utiliza instâncias do Amazon EC2

EXTERNO



Preço baseado em horas de instância e cobranças adicionais para outros produtos da AWS usados
Infraestrutura local autogerenciada com o ECS Anywhere

*Obrigatório

Cancelar

Próxima etapa

- Dê um nome e selecione a imagem Docker a ser utilizada na instância (copie o ID do repositório no ECR); indique também o mapeamento de portas (neste caso, o servido da API estará disponível na porta 9000, que foi associada à porta 3000 padrão de saída do Node JS):

Adicionar contêiner

▼ Padrão

Nome do contêiner* ⓘ

Imagem* ⓘ

Autenticação de repositório privado* ☐ ⓘ

Limites de memória (MiB)* Limite flex... ▼ ⓘ

➕ Adicionar limite rígido


Defina limites de memória rígida ou flexível em MiB para o contêiner. Os limites rígidos e flexíveis correspondem aos parâmetros "memória" e "memoryReservation", respectivamente, em definições da tarefa.
O ECS recomenda 300 a 500 MiB como ponto de partida para aplicações Web.

Mapeamentos de porta	Porta do host	Porta do contêiner	Protocolo
	<input type="text" value="9000"/>	<input type="text" value="3000"/>	tcp ▼

➕ Adicionar mapeamento de porta

- Resumo da *task*:

Definições de contêiner

	Nome do contêiner	Imagem	Unidades de CPU	GPU	Acelerador de inferência	Limites de memória fixos/flexíveis (MiB)	Essencial
▼	 My_API-Sprint3	036531232259.dkr.ecr.us-e...	0			--/400	true
Detalhes							
Mapeamentos de porta							
	Porta do host	Porta do contêiner	Protocolo				
	9000	3000	tcp				
Variáveis de ambiente							
	Chave	Valor/valor de					
	Nenhuma variável de ambiente						
Arquivos de ambiente							
	Origem	Local					
	Nenhum arquivo de ambiente						
Ordenação de contêineres							
	Nome do contêiner	Condição					
	Nenhuma ordenação de contêineres						
Tempos limite de contêiner							
Tempo limite de início:							
Tempo limite de interrupção:							
Rótulos do Docker							
	Chave	Valor					
	Nenhum rótulo do Docker						

Pontos de montagem

Caminho do contêiner	Volume de origem	Somente leitura
Nenhum ponto de montagem		

Volumes de

Contêiner de origem	Somente leitura
Nenhum volume de origem	

Ulimits

Nome	Limite flexível	Limite rígido
Nenhum ulimit		

Inferência elástica

Acelerador
None

Configuração de log

Driver de log:

Chave	Valor
Nenhuma configuração de log	

Agora falta configurar o Serviço (*Service*) que será o responsável por lançar a *Definição de tarefa* (*Task definition*) que, por sua vez, sabe qual cluster de instâncias ECS deve lançar e qual imagem Docker no ECR as instâncias irão rodar.

- Ainda na interface do ECS, selecione a *Definição de tarefa* que acabou de criar; em *ações*, selecione *criar serviço*:

Criar serviço

[Etapa 1: Configurar serviço](#)

[Etapa 2: Configurar rede](#)

[Etapa 3: Definir Auto Scaling \(opcional\)](#)

Etapa 4: Revisar

Revisar

Editar

Cluster Cluster-Sprint3

Tipo de inicialização EC2

Definição da tarefa Task_Def-Sprint3-1

Nome do serviço service-task-sprint3

Tipo de serviço REPLICA

Número de tarefas 1

Percentual de integridade mínima 1

Porcentagem máxima 100

Disjuntor de implantação Desabilitado

Configurar rede

Editar

não configurado

Definir Auto Scaling (opcional)

Editar

não configurado

Cancelar

Anterior

Criar serviço

- Resumo do *service*: cluster de 1 instância EC2 rodando a aplicação!

Contêineres

Última atualização em março 18, PM 4:27:25 PM (há 0 minutos)



Nome	ID do tempo de exec...	Status	Imagem	Resumo de imagens	Unidades de CPU	Limites de memó...	Essencial	ID do recurso
▼ Sprint3-Co...	80a978976822823ea...	RUNNING	036531232259.dkr.ecr.us-east-1.a...	sha256:bb052952510e11bfea939c...	--	--/400	true	51325849-ed03-4...

Detalhes

Associações de rede

Porta do host	Porta do contêiner	Protocolo	Link externo
9000	3000	tcp	44.202.222.131:9000

Variáveis de ambiente - não configurado

Arquivos de ambiente - não configurado

Rótulos do Docker - não configurado

Hosts adicionais - não configurado

Pontos de montagem - não configurado

Volumes de - não configurado

Ulimits - não configurado

Inferência elástica - não configurado

Configuração de log - não configurado

Importante: Adicionalmente, é possível configurar o **Load Balancer** (caso seu cluster tenha duas ou mais instâncias, redistribui o tráfego entre elas) e definir também configurações de **Auto Scaling** (a partir de determinada parâmetro de acesso, ou mesmo para data e hora definidas, o ECS pode aumentar a quantidade de instâncias fornecendo sua API – isto é muito útil para período de pico de acesso à sua aplicação instância).