COMPUTAÇÃO EM NUVEM

AULA 6

Profª Ana Paula Costacurta

CONVERSA INICIAL

Na primeira parte desta aula, vamos falar sobre o conceito de Docker e de microsserviços. Apresentaremos a comparação de contêiner com outras arquiteturas. Citaremos os benefícios de utilização do Docker e as formas de execução.

Na segunda parte, aprenderemos sobre o serviço Amazon Elastic Container Registry (ECR), seus componentes e funcionamento na AWS. Aprenderemos mais em detalhes sobre registros e repositório.

Na terceira parte, conheceremos Amazon Elastic Container Service (ECS). Serão apresentados os benefícios da utilização do Amazon ECS e seus componentes. Iremos detalhar os componentes, conhecer os tipos de inicialização e como é realizada a atribuição de um nome de recursos (ARN) para tarefas, serviços e instâncias de contêineres da Amazon ECS.

Na quarta parte, vamos ver sobre Amazon Elastic Kubernetes Services (EKS), sua integração com demais serviços do AWS e seu funcionamento. Conheceremos os componentes do Kubernetes e como é o funcionamento do plano de controle e plano de dados. Serão detalhados os componentes do plano de trabalho e componentes do nó.

Na quinta parte, aprenderemos sobre Amazon Fargate, como é seu funcionamento e a utilização de instâncias EC2 com e sem utilização do Amazon Fargate. Conheceremos as principais diferenças de configurações do Amazon EC2 como tipo de inicialização.

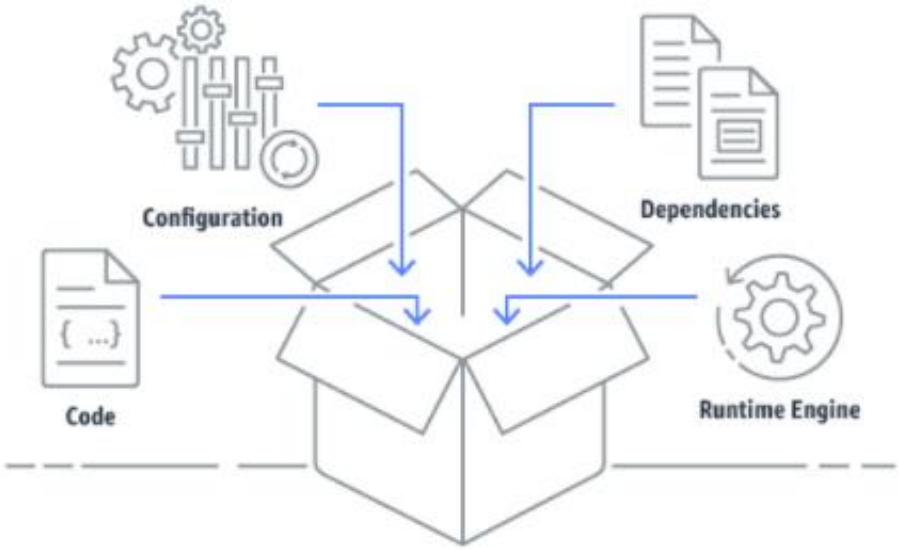
TEMA 1 – DOCKER E MICROSSERVIÇOS

Com Docker são possíveis criação, teste e implantação de *software* de forma mais rápida.

1.1 Docker

Com essa plataforma é possível criar pacotes de *softwares* em uma unidade padrão que chamamos de *contêineres*. Esses pacotes possuem tudo que o *software* necessita para realizar a execução: bibliotecas, ferramentas do sistema, código e *runtime*. Na Figura 1, podemos ver o conceito de contêiner de forma gráfica.

Figura 1 – Conceito de contêiner



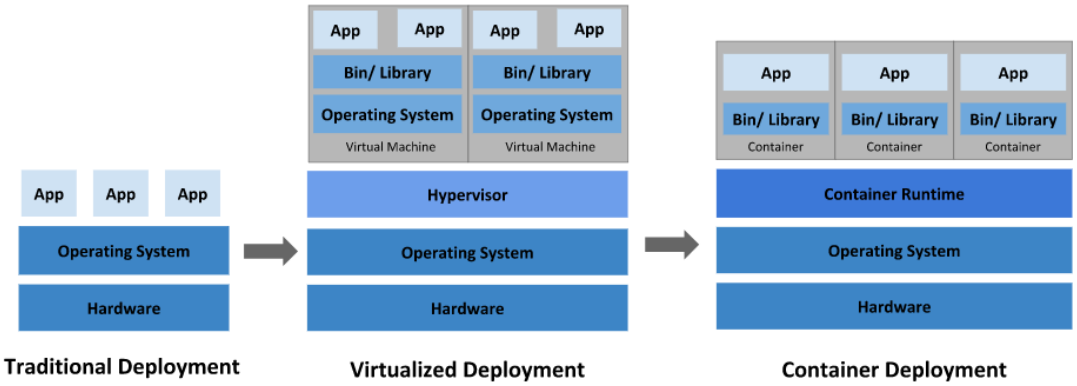
Fonte: AWS BR, 2020.

Existem dois modelos do Docker disponibilizados na AWS: o Docker Community Edition (CE) de código aberto e o Docker Enterprise Edition (EE) baseado em assinatura. O Docker CE é gratuito e pode ser instalado em uma máquina virtual na Amazon EC2 e o Docker EE é uma versão paga que adiciona recursos para gerenciamento e segurança de fluxo do trabalho.

A execução do código realizada pelo Docker é padronizada, sendo um sistema operacional para contêineres virtualizando o sistema operacional de um servidor.

O Docker é instalado em cada servidor e possibilita criar, iniciar ou interromper contêineres. Na Figura 2, podemos ver a arquitetura dos contêineres comparando a uma máquina virtual.

Figura 2 – Comparação entre máquina física, máquina virtual e contêineres



Fonte: Kube, 2020.

Podemos citar alguns benefícios da utilização do Docker:

1. Disponibilização de *softwares* com maior rapidez e de forma isolada;
2. Padronização de operações;
3. Transferência de máquinas com maior agilidade;
4. Economia de dinheiro, pois facilita a execução e melhora a utilização do servidor.

Os contêineres na AWS podem ser executados de várias maneiras:

1. Amazon Elastic Container Service (ECS): gerenciamento de contêineres;
2. AWS Fargate: execução de contêineres sem servidor;
3. Amazon Elastic Container Service for Kubernetes (EKS): execução do Kubernetes na AWS;
4. Amazon Elastic Container Registry (ECR): repositório privado de contêineres.

1.2 Microsserviços

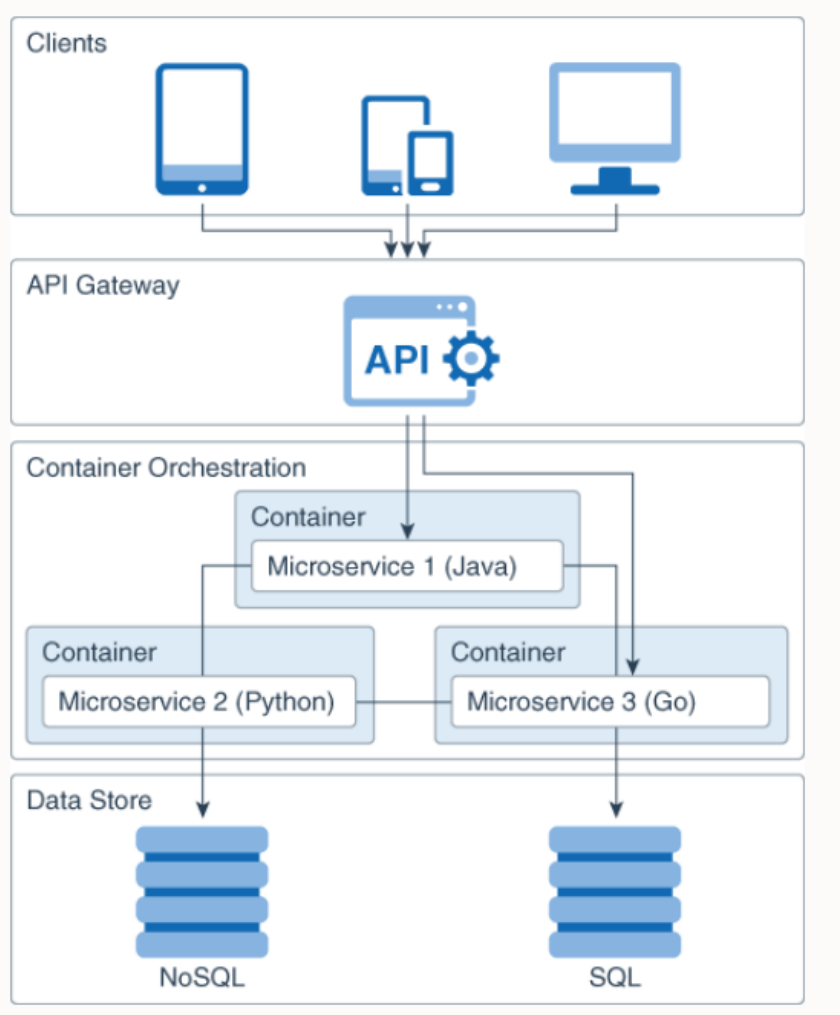
Microsserviços consistem em pequenos serviços, independentes entre si, que se comunicam utilizando APIs. Essa arquitetura facilita a escalabilidade e agilidade para desenvolvimento de aplicativos.

Uma solicitação de serviço é realizada por meio de uma API REST, que representa o ponto de entrada da funcionalidade do aplicativo, podendo realizar

a comunicação com o microsserviço diretamente ou por meio de um *gateway* de API. Estudamos esse assunto em aulas anteriores, no Amazon API Gateway.

Na Figura 3, podemos ver a arquitetura de um aplicativo que possui vários microsserviços.

Figura 3 – Arquitetura microsserviço



Fonte: Oracle BR, 2020.

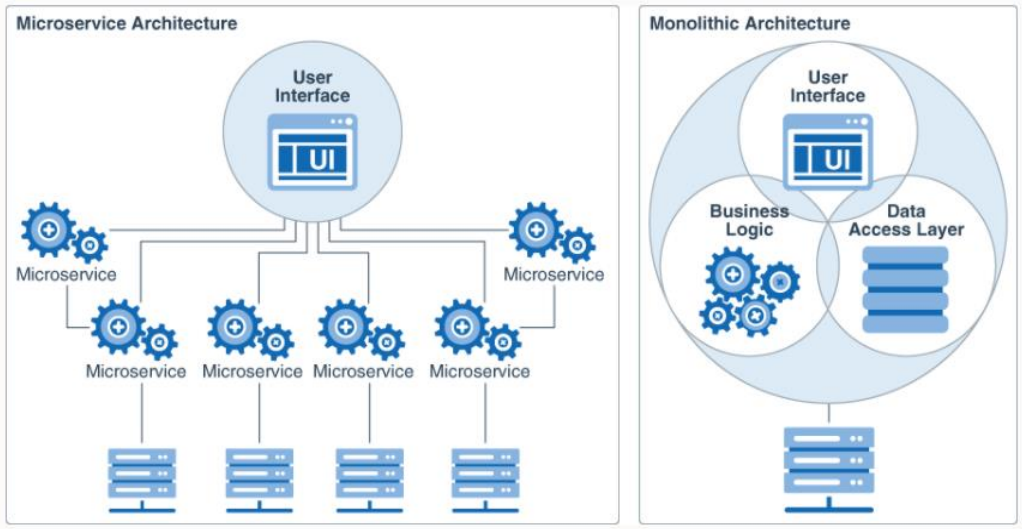
Com arquiteturas tradicionais, as arquiteturas monolíticas, os processos são acoplados fortemente e executam como um serviço único. As desvantagens dessa arquitetura são a complexidade para adição ou aprimoramento de recursos de aplicativos e também o crescimento do código, tornando muito complexo e limitando novas ideias.

A arquitetura de microsserviços possibilita criar componentes independentes que executam cada processo de um aplicativo na forma de um

serviço. As vantagens dessa arquitetura é a facilidade de implantação, atualização e escalabilidade para atender às demandas individuais de cada microsserviço.

Na Figura 4, podemos ver a comparação entre a arquitetura monolítica e microsserviços.

Figura 4 – Comparação entre arquitetura monolítica e microsserviços



Fonte: Oracle BR, 2020.

TEMA 2 – AMAZON ELASTIC CONTAINER REGISTRY (ECR)

O Amazon ECR é um serviço de registro de contêineres do Docker totalmente gerenciável. É possível armazenar, gerenciar e implantar facilmente imagens de contêineres do Docker. Existe uma integração entre o Amazon ECR é o Amazon Elastic Container Service (ECS), que abordaremos no próximo tema.

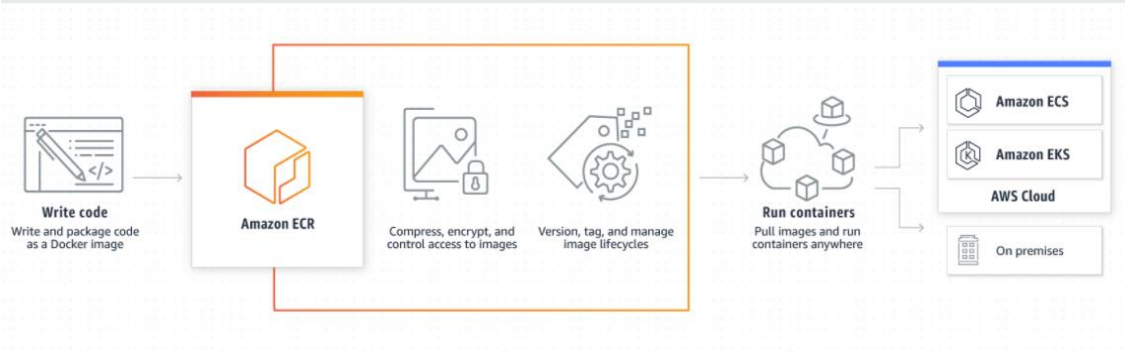
Esse serviço da AWS elimina a necessidade de realizar a operação de repositórios local de contêineres ou a preocupação com a escalabilidade da infraestrutura. O Amazon ECR realiza a hospedagem das imagens, em que há uma alta disponibilidade e com escalabilidade, sendo realizado o controle no nível de recurso de cada repositório pelo AWS IAM. O Amazon ECR utiliza o Amazon S3 para armazenar, o que torna suas imagens de contêiner. Na Figura 5, podemos ver o funcionamento do Amazon ECR.

2.1 Componentes

O Amazon ECR possui os seguintes componentes:

1. **Registro:** é fornecido um registro a cada conta AWS, assim, é possível criar repositórios de imagens no registro e armazenar imagens;
2. **Token de autorização:** é necessário estar autenticado nos registros da Amazon ECR com um usuário da AWS; só assim é possível enviar e extrair imagens;
3. **Repositório:** contém as imagens do Docker;
4. **Política de repositórios:** controle de acesso aos repositórios e as imagens;
5. **Imagem:** enviar ou extrair imagens de contêineres dos repositórios.

Figura 5 – Funcionamento Amazon ECR



Fonte: AWS BR, 2020.

2.2 Registros

Os registros do Amazon ECR hospedam as imagens do contêiner. Por padrão, toda conta da AWS possui um único registro do Amazon ECR e o usuário do IAM necessita de permissões adicionais para fazer chamada de APIs do Amazon ECR e para enviar e extrair imagens dos repositórios, pois por padrão tem acesso de leitura e gravação apenas.

O URL do registro padrão é:

https://**aws_account_id**.dkr.ecr.**region**.amazonaws.com

A utilização de Token de autorização é para acesso a qualquer registro do Amazon ECR, com validade por 12 horas.

2.3 Repositório

O Amazon ECR fornece formas de criar, monitorar e excluir imagem e definição de políticas de permissões no repositório para possibilitar o controle de quem pode acessá-los. Podem ser realizadas essas ações pelo console do ECR ou por operações de API, além de ter integração a CLI do Docker, permitindo enviar e extrair imagens de ambientes de desenvolvimento para os repositórios.

TEMA 3 – AMAZON ELASTIC CONTAINER SERVICE (ECS)

O Amazon ECS é um serviço de orquestração de contêineres da Docker na AWS. Possui duas possibilidades de lançamento: utilizando instâncias EC2 ou AWS Fargate. Ambos são similares em relação a orquestração, porém o AWS Fargate é o mais atual, tornando-se uma boa opção, pois pode ser executado sem necessidade de servidor e por ser seguro, confiável e com disponibilidade alta. Também é possível integrar de forma nativa com outros serviços da AWS.

Os benefícios da utilização do Amazon ECS são os seguintes:

1. **Sem necessidade de servidor:** compatível com Amazon Fargate, eliminando a necessidade de provisionar e gerenciar servidores;
2. **Provedor de capacidade:** ajuda a concentrar na criação e no gerenciamento de aplicativos em vez da infraestrutura, sendo as demandas do aplicativo que determinam a capacidade alocada;
3. **Performance em escala:** vários serviços podem ser iniciados rapidamente sem complexidade;
4. **Seguro:** executa os contêineres em suas próprias Amazon VPC, em que nenhum recurso computacional é compartilhado entre clientes da AWS;
5. **Confiável:** uma infraestrutura global com 69 zonas de disponibilidade em 22 regiões;
6. **Otimizando para custo:** pode misturar instâncias *spot*, sob demanda e reservadas para descontos.

3.1 Componentes

Antes de executar um contêiner Docker em uma conta AWS utilizando uma imagem, é necessário criar a infraestrutura onde será executado.

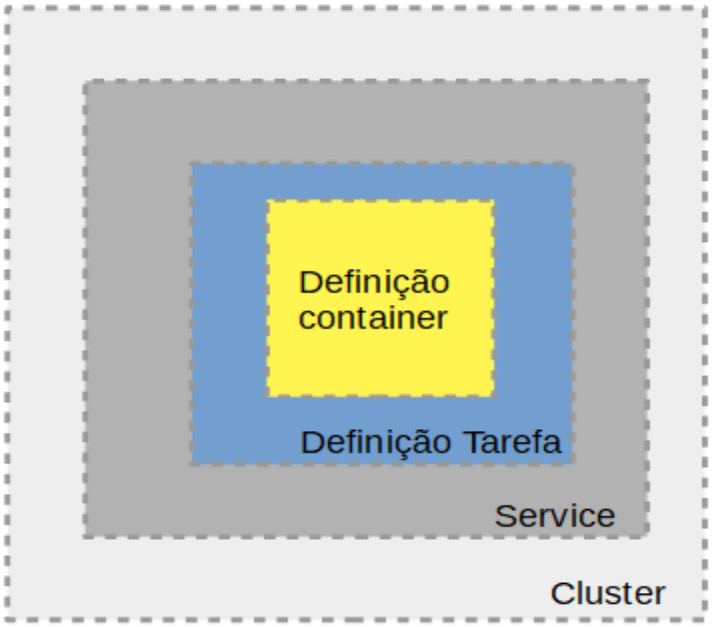
Na Figura 6, podemos ver os principais componentes da infraestrutura que precisam ser lembrados, sendo eles:

1. Definição de Tarefas (*Task definition*): especificação de uma tarefa do AWS ECS. A tarefa (*Task*) é a unidade miníma de trabalho do Amazon ECS. Deve estar obrigatoriamente associada a uma definição de tarefas e opcionalmente a um serviço;

2. Serviço (*Service*): gerencia o ciclo de vida das tarefas;

3. *Cluster*: é um agrupamento lógico de serviços e tarefas no AWS.

Figura 6 – Componentes principais da infraestrutura



3.3 Definição de tarefa

É necessária a definição das tarefas para execução de contêineres do Docker no Amazon ECS. Alguns dos parâmetros mais importantes que devem ser especificados são os listados a seguir.

1. Quantidade de CPU e memória;
2. Tipo de inicialização;
3. Modo de rede;

4. Volume de dados;
5. Função IAM.

Quando uma definição de tarefa está sendo executada no *cluster*, é chamado de *tarefa*, logo, um contêiner em execução é a mesma coisa que uma tarefa do ECS em execução.

3.4 Serviço

Com um serviço, é possível executar e manter, de forma simultânea, um número um ou mais instâncias de uma definição de tarefas em um *cluster* do Amazon ECS. Para manter o serviço, o programador de serviço do Amazon ECS iniciará uma outra instância da definição de tarefas para substituição da que falhou.

A definição de como executar o serviço é realizada especificando os parâmetros a serem seguidos na execução. O único parâmetro obrigatório é o nome do serviço. Caso algum outro parâmetro não seja informado, o Amazon ECS colocará o valor padrão, cujos principais parâmetros são os seguintes:

1. Tipo de inicialização;
2. Definição de tarefa;
3. *Cluster*;
4. Nome do serviço.

3.5 *Cluster*

É possível selecionar imagens e recursos dos contêineres AWS que serão necessários para a aplicação, podendo realizar ainda a criação de um *cluster*.

O *cluster* é um agrupamento lógico de serviços e tarefas no AWS. Se estiver falando de AWS EC2, é também um grupo de instâncias de contêineres ou se estiver falando de provedor de capacidade, é também um grupo de provedores de capacidade.

Quando é utilizado o Amazon ECS pela primeira vez, o AWS cria um *cluster* padrão, deixando livre a possibilidade de criação de novos *clusters* conforme necessidade.

3.7 Tipo de inicialização

O tipo de inicialização AWS ECS vai determinar o tipo de infraestrutura necessária para as tarefas e serviços, quando criada a arquitetura da aplicação define o tipo de inicialização: Amazon Fargate ou Amazon EC2.

A diferença será que, se você utilizar o AWS Fargate, a gestão será realizada automaticamente pelo AWS e no EC2 será necessário realizar algumas configurações, o que torna essa opção mais flexível e com um nível maior de controle.

Independente da opção que você escolher, Amazon EC2 ou Amazon Fargate, o Amazon ECS dimensiona seu aplicativo e gerencia seus contêineres AWS. A principal diferença será quando existir a necessidade de escalabilidade. No AWS Fargate, não há necessidade de se preocupar com nada, porém isso pode gerar um custo elevado. Já no AWS EC2, o acompanhamento tem que ser constante das instâncias e contêineres, e manualmente realizar a otimização caso seja necessário, assim o controle de custos é mais certo.

Quando utilizado o tipo de inicialização AWS Fargate, pode-se incluir vários contêineres na mesma tarefa ou implementar separadamente, sendo importante quando os contêineres compartilham recursos. Se utilizado o AWS EC2, isso pode agrupar tarefas com uma mesma finalidade, pois é necessário pensar na execução e na necessidade de escalabilidade de cada componente.

Caso seja especificado o tipo de inicialização, a estratégia do provedor de capacidade deve ser omitida e vice-versa. Se nenhum dos dois foi especificado, será utilizada a estratégia do provedor de capacidade padrão.

3.8 Amazon Recource Name (ARN)

Quando são criados os recursos do Amazon ECS, é realizada a atribuição a cada recurso um nome de recurso da Amazon (ARN) e um identificador de recurso (ID), que são exclusivos, necessários quando são usados em uma ferramenta de linha de comando.

Alguns recursos possuem nome amigável, porém, em outros casos, é necessário especificar um recurso utilizando o formato de ARN. O novo formato de ARN para tarefas, serviços e instâncias de contêineres do Amazon ECS, que está sendo utilizado a partir de abril de 2021, inclui o nome do *cluster*. O Quadro 1 mostra o formato do ARN para cada tipo de recurso.

Quadro 1 – Formato ARN

Tipo de recurso	ARN
Instância de contêiner	arn:aws:ecs:**region:aws_account_id**:container-instance/**cluster-name/container-instance-id**
Serviço da Amazon ECS	arn:aws:ecs:**region:aws_account_id**:service/**cluster-name/service-name**
Tarefa do Amazon ECS	arn:aws:ecs:**region:aws_account_id**:task/**cluster-name/task-id**

Fonte: AWS BR, 2020.

TEMA 4 – AMAZON ELASTIC KUBERNETES SERVICES (EKS)

O Amazon EKS é um serviço Kubernetes na AWS gerenciável. O Kubernetes é um *software* de código aberto que possibilita implementação e gerenciamento de aplicativos conteinerizados em grade escala.

A execução do Kubernetes na nuvem é facilitada pela AWS, com uma infraestrutura de máquinas virtuais escalável e disponível, integração com serviços e o Amazon EKS que provisiona e escala o plano de trabalho do Kubernetes. Na Figura 7, podemos ver como é a integração do Kubernetes com demais serviços.

Figura 7 – Integração do Kubernetes

4.1 Funcionamento

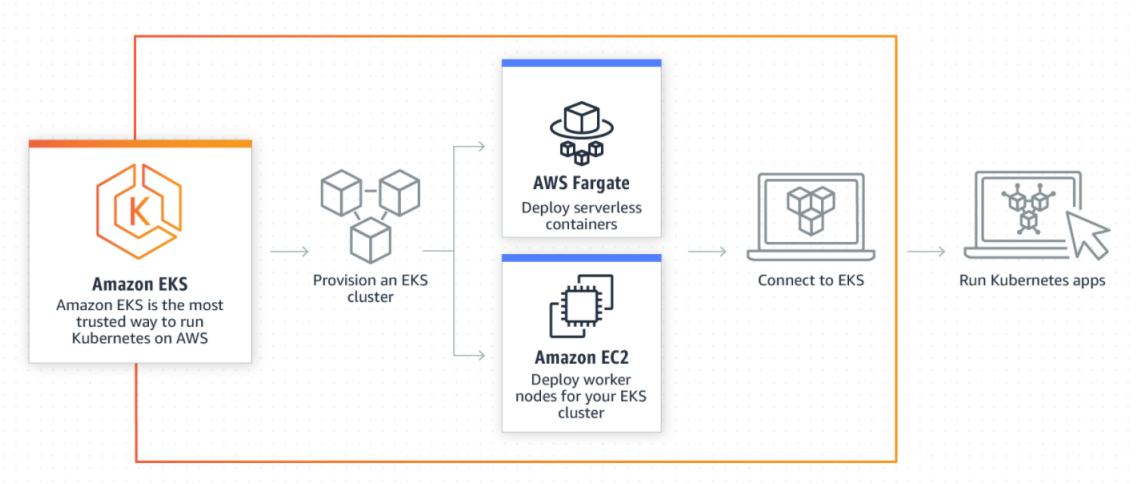
Com o Amazon EKS, é apenas necessário provisionar os nós de operador, e a AWS realiza as atividades de provisionamento, escalabilidade e gerenciamento do Plano de Controle do Kubernetes utilizando uma configuração segura e com alta disponibilidade.

Na Figura 8, podemos ver o resumo do funcionamento de aplicações Kubernetes no Amazon EKS.

4.2 Componentes do Kubernetes

Ao implantar o Kubernetes é obtido um *cluster*. O Kubernetes realiza o gerenciamento de um *cluster* de instâncias de computação e realiza a programação dos contêineres para execução no cluster conforme os recursos computacionais disponíveis e os requisitos de recursos de cada contêiner.

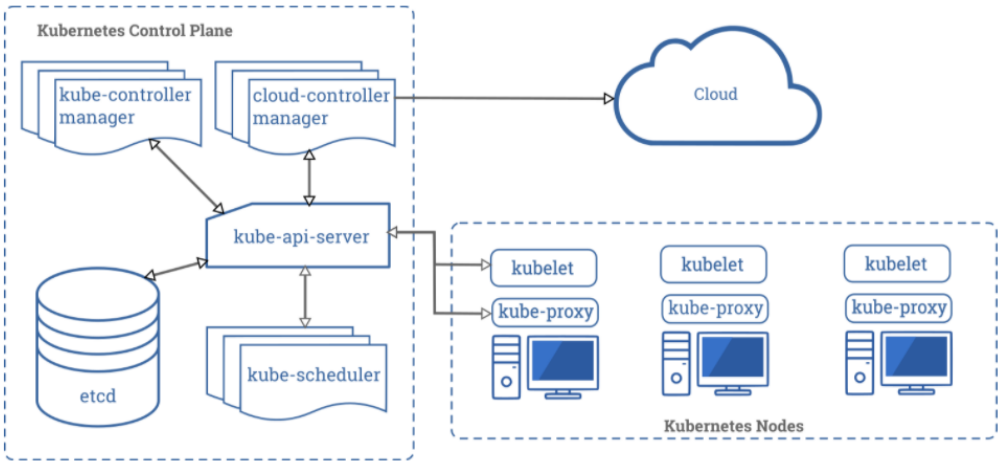
Figura 8 – Funcionamento Kubernetes no Amazon EKS



Fonte: AWS BR, 2020.

Um agrupamento lógico de instâncias de computação do EC2 que executa contêineres é um *cluster* do Kubernetes, local que contém plano de controle e plano de dados. Na Figura 7, podemos ver o diagrama de um *cluster* Kubernetes com todos os componentes interligados.

Figura 9 – Componentes Kubernetes



Fonte: Kube, 2020.

O plano de controle é o local onde estão os componentes que controlam o *cluster* e os dados sobre o estado e configuração do *cluster*. Esse plano contém como as instâncias que controlam *como*, *quando* e *onde* os contêineres são executados. Os planos de dados são as instâncias em que os contêineres são executados.

Para realizar a execução dos contêineres ou serviços como Kubernetes, é necessário definir um *cluster*, e o plano de controle faz com que a configuração do *cluster* aconteça conforme o planejado. As execuções dos contêineres em agrupamentos lógicos são chamadas de *pods*, em que cada *pod* pode conter um contêiner ou pôr vários deles com acoplamento. O *pod* representa uma instância de uma aplicação.

O nó é uma instância de computação que faz parte de um *cluster* do Kubernetes. O plano de controle possui dois tipos possíveis de instâncias: *mestre* e *operador*. Os nós mestres são componentes essenciais, que garantem que os contêineres sejam executados na quantidade e recursos corretos. Os nós operadores são as instâncias de computação que executam os contêineres e processam dados. Os *pods* são programados e orquestrados para serem executados nos nós operadores.

4.3 Componentes do plano de controle

As decisões globais sobre o *cluster* são tomadas pelos componentes do plano de controle. A realização e a detecção também respondem a eventos do *cluster*, cujos componentes são os seguintes:

1. **Servidor API Kubernetes (*Kube-apiserver*):** valida e configura dados para os objetos de API, que incluem *pods*, serviços, controladores de replicação e outros;
2. **Gerenciador de controlador de nuvem (*cloud-controller-manager*):** é um *daemon*, um programa que realiza a execução como um processo em plano de fundo, realizando a incorporação dos loops de controle específicos da nuvem enviados com o Kubernetes;
3. **Gerenciador de controlador Kubernetes (*kube-controller-manager*):** é um *daemon*, um programa que executa como um processo em plano de fundo, que incorpora os *loops* de controle principais fornecidos com o Kubernetes;
4. **Armazenamento de valor chave (etcd):** consistente e altamente disponível, sendo utilizado como armazenamento de apoio do Kubernetes para todos os dados do *cluster*;
5. **Programador kubernetes (*kube-scheduler*):** observa os *Pods* que são criados sem um nó e realiza a seleção de um nó para executar.

4.4 Componentes do nó

Os componentes são executados em cada nó, mantendo os *pods* em execução e fornecendo o ambiente de execução do Kubernetes. Os componentes de nó são:

1. **Agente de nó (kubelet):** é executado em cada nó, podendo registrar o nó com o *apiserver* utilizando um dos seguintes: o nome do *host*; um sinalizador para substituir o nome do *host*; ou lógica específica para um provedor de nuvem;
2. **Proxy de Rede (Kube-proxy):** é executado em cada nó, refletindo os serviços definidos na API Kubernetes em cada nó e possibilitando realizar o encaminhamento de *stream* TCP, UDP e SCTP simples ou encaminhamento *round robin* de TCP, UDP e SCTP em um conjunto de *back-ends*;
3. **Tempo de execução do contêiner (*container runtime*):** é responsável por executar os contêineres.

TEMA 5 – AMAZON FARGATE

O AWS Fargate é um mecanismo de computação sem servidor para contêineres e o chamamos de *serveless do ECS*, o que facilita a concentração no desenvolvimento de aplicativos, eliminando a necessidade e provisionamento e gerenciamento de servidores. Funciona com Amazon Elastic Container Service (Amazon ECS) e Amazon Elastic Kubernetes Services (Amazon EKS).

Com a utilização do AWS Fargate, eliminamos a necessidade de escolha de instâncias e ajuste de escala da capacidade do *cluster*, alocando a quantidade certa de computação.

5.1 Funcionamento

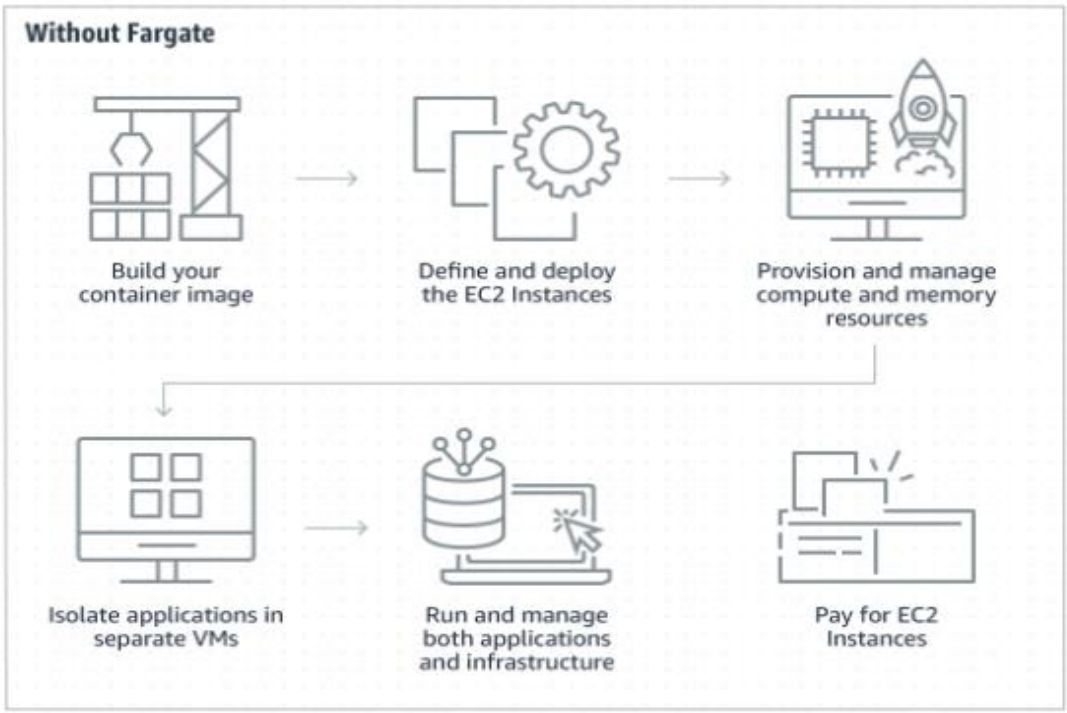
AWS Fargate é uma nova funcionalidade do Amazon ECS, que possibilita a AWS realizar o gerenciamento automático dos contêineres deixando apenas para o usuário a implementação do serviço. Caso exista a necessidade de um maior controle das instâncias do EC2, requisitos de conformidade e governança ou opções avançadas de personalização, é recomendada a utilização do Amazon ECS ou Amazon EKS sem o AWS Fargate.

A única coisa que você precisa fazer é fornecer ao AWS Fargate a imagem do contêiner e implantá-la como um serviço ou tarefa única (contêiner) no ECS.

Na Figura 9, podemos ver o funcionamento de instâncias EC2 sem utilização do AWS Fargate. Nesse caso, existe a necessidade de provisionar e escalar *cluster*.

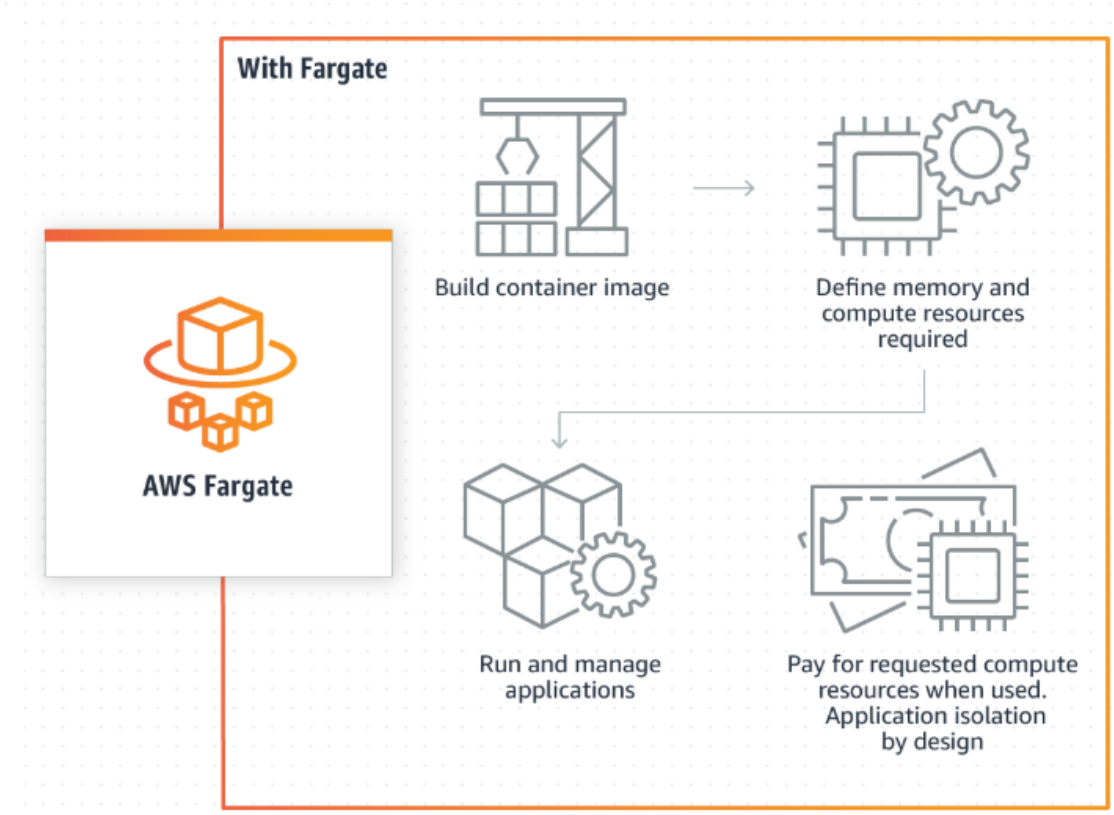
E na Figura 10, é possível observar o funcionamento de instâncias utilizando EC2, em que o usuário precisa apenas definir os requisitos do aplicativo, selecionar que executará o AWS Fargate e este se encarrega de toda a escalabilidade e gerenciamento da infraestrutura necessária para execução.

Figura 10 – Funcionamento de instâncias EC2 sem AWS Fargate



Fonte: AWS BR, 2020.

Figura 11 – Funcionamento instância EC2 com AWS Fargate



Fonte: AWS BR, 2020.

5.2 Componentes

Como já mencionado, com AWS Fargate não são necessárias muitas preocupações para provisionamento, pois o AWS gerencia isso, bastando realizar a configuração de definição de tarefa e mandar executar, porém, algumas configurações de definição de tarefas e serviços são um pouco diferentes das instâncias EC2 sem AWS Fargate, como tipo de inicialização, sendo:

1. modo de rede apenas *awsvpc*;
2. gestão de volumes com opções de driver e volumes voláteis;
3. não está disponível em todas as regiões;
4. apenas contêineres Linux são suportados.

FINALIZANDO

Na primeira parte de nossa aula, falamos Docker, que possibilita o empacotamento de aplicações para executar independente da infraestrutura, em que criamos contêineres que possuem tudo que é necessário para a execução da aplicação e citamos as duas versões do Docker na AWS: a gratuita, o Docker CE e uma versão paga Docker EE.

Apresentamos as formas em que podemos executar os contêineres, utilizando: Amazon ECS, Amazon EKS e Amazon ECR. Falamos sobre microsserviços, que são pequenos serviços que possibilitam executar cada processo de um aplicativo independentemente e facilitando a implantação e manutenção de aplicações.

Na segunda parte, conhecemos os Amazon Elastic Container Registry (ECR), um serviço de registro de contêineres do Docker que é totalmente gerenciável. O Amazon ECR realiza as hospedagens das imagens e realiza o controle de cada repositório pelo AWS IAM.

Conhecemos os componentes do Amazon ECS, sendo eles: registro, token de autorização, repositório, política de repositório e Imagem. Os registros hospedam as imagens dos contêineres e utilizam o token de autorização para acessar qualquer registro do Amazon ECR. Os repositórios fornecem formas de criar, monitorar e excluir imagens e também realizar as definições de permissões do repositório criando políticas.

Na terceira parte, aprendemos sobre Amazon Elastic Container Service (ECS) um serviço de orquestração de contêineres da Docker, que possui duas possibilidades de lançamento: Instâncias EC2 ou AWS Fargate.

Citamos os benefícios de utilização: Sem necessidade de servidor, provedor de capacidade, performance em escala, seguro, confiável e otimizando para custo. Conhecemos os componentes da infraestrutura de um contêiner Docker: definição de tarefa, serviço e *cluster*.

A definição de tarefa é importante para a execução do contêiner. Citamos os principais parâmetros que devem ser especificados: quantidade de CPU e Memória, tipo de inicialização, modo de rede, volume de dados e função IAM. Quando a definição de tarefa é executada, nós a chamamos de *cluster*.

O serviço possibilita a execução e manter em um *cluster*, iniciando uma nova instância no caso de falha de uma outra. A definição de como executar um

serviço é realizada especificando alguns parâmetros: tipo de inicialização, definição de tarefa, *cluster* e nome do serviço.

O *cluster* é a seleção de recursos e imagem necessários do contêiner AWS, sendo um agrupamento lógico de serviços e tarefas no AWS. O *cluster* padrão é criado quando utilizado o Amazon ECS pela primeira vez.

Para realizar a inicialização do AWS ECS, é possível utilizar a Amazon Fargate ou a Amazon EC2, sendo que no AWS Fargate a gestão é realizada automaticamente pelo AWS e no EC2 é necessário realizar algumas configurações. Em função da necessidade de realizar a otimização das instâncias no EC2, será necessário realizar manualmente o que o Amazon Fargate fará automaticamente.

São realizados atribuição de nome de recursos da Amazon (ARN) para tarefas, serviços e instâncias de contêineres do Amazon ECS, o que inclui o nome do *cluster* no formato do ARN.

Na quarta parte, conhecemos o Amazon Elastic Kubernetes Services (EKS), um *software* de código aberto que possibilita a implementação e gerenciamento de aplicativos conteinerizados em uma grande escala. Com a utilização do Amazon EKS, somente é necessário provisionar o nó operador e o restante fica a cargo da AWS.

Os componentes do Kuberentes são: plano de trabalho e nós operadores. Os *pods* representam uma instância da aplicação, que é a execução dos contêineres em agrupamentos lógicos.

No plano de trabalho são tomadas as decisões globais e detecção e resposta a eventos do *cluster*. Os componentes do plano de trabalho são: servidor API Kubernetes, gerenciador de controlador de nuvem, gerenciador de controlador Kubernetes, armazenamento de valor chave e programador Kubernetes.

Os componentes do nó que mantêm os *pods* em execução são executados em cada nó. Os componentes do nó são: agente de nó, Proxy de rede e tempo de execução do contêiner.

Na quinta parte, falamos sobre Amazon Fargate, um mecanismo de computação sem servidor, eliminando a necessidade de provisionamento e gerenciamento de servidores. Para utilzá-lo, é necessário somente fornecer a imagem do contêiner e realizar a implantação como serviço ou tarefa única.

Os componentes que diferenciam na inicialização com Amazon Fargate são os seguintes: modo de rede, opção de *drive* e volume voláteis. Apenas algumas regiões têm disponibilidade e suportam somente contêiner Linux.

REFERÊNCIAS

AWS BR. Disponível em: <https://aws.amazon.com/pt>. Acesso em: 13 jan. 2021.

IFRAH, S. **Deploy containers on AWS**: with EC2, ECS, and EKS. New York: Apress, 2019.

KANE, S. P.; MATTHIAS, K. **Docker:** up & running – shipping reliable containers in production**. 2. ed.** Sebastopol, United States, 2018.

KUBE. Disponível em: <https://kubernetes.io/>. Acesso em: 13 jan. 2021.

LUKSA, M. Kubernetes in action. 2. ed. [S.l.]: Manning, 2020.

ORACLE BR. Disponível em: <https://oracle.com>. Acesso em: 13 jan. 2021.

VITALINO, J. F. N; CASTRO, M. A. N. **Descomplicando o Docker**. 2. ed. São Paulo: Brasport, 2018