**1. EKS & ECR - Introduction**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/10-ECR-Elastic-Container-Registry-and-EKS>

--- we are going to understand about an elastic container registry in combination with EKS.

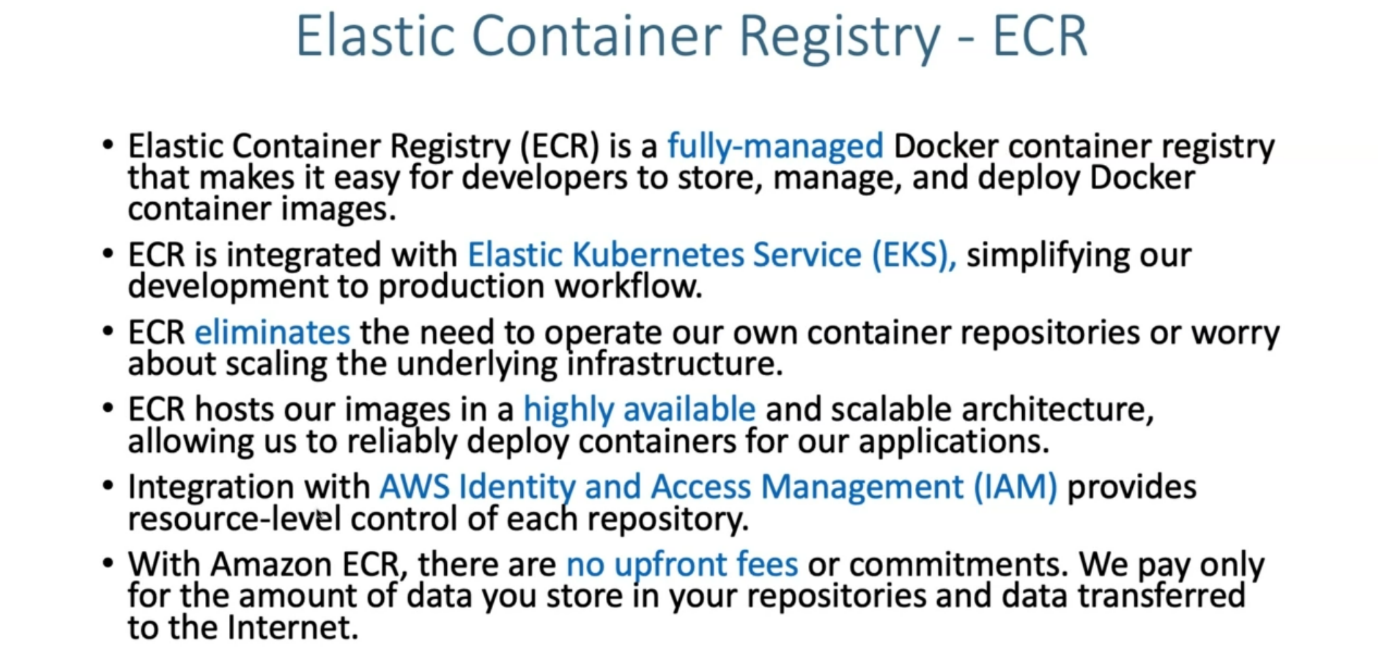
--- So far, Whatever the doctor images we are pulling and then testing in this course are from our Docker hub repository.

--- now, how we are going to use a elastic container registry, which is a aws registry for our docker images.

--- we are going to build a Docker image locally and then push it to container registry and pull that

from this ECR to EKS deployments, whatever we are going to deploy. the entire use case. We are going to test in this respective section.

**Elastic container registry – ECR**

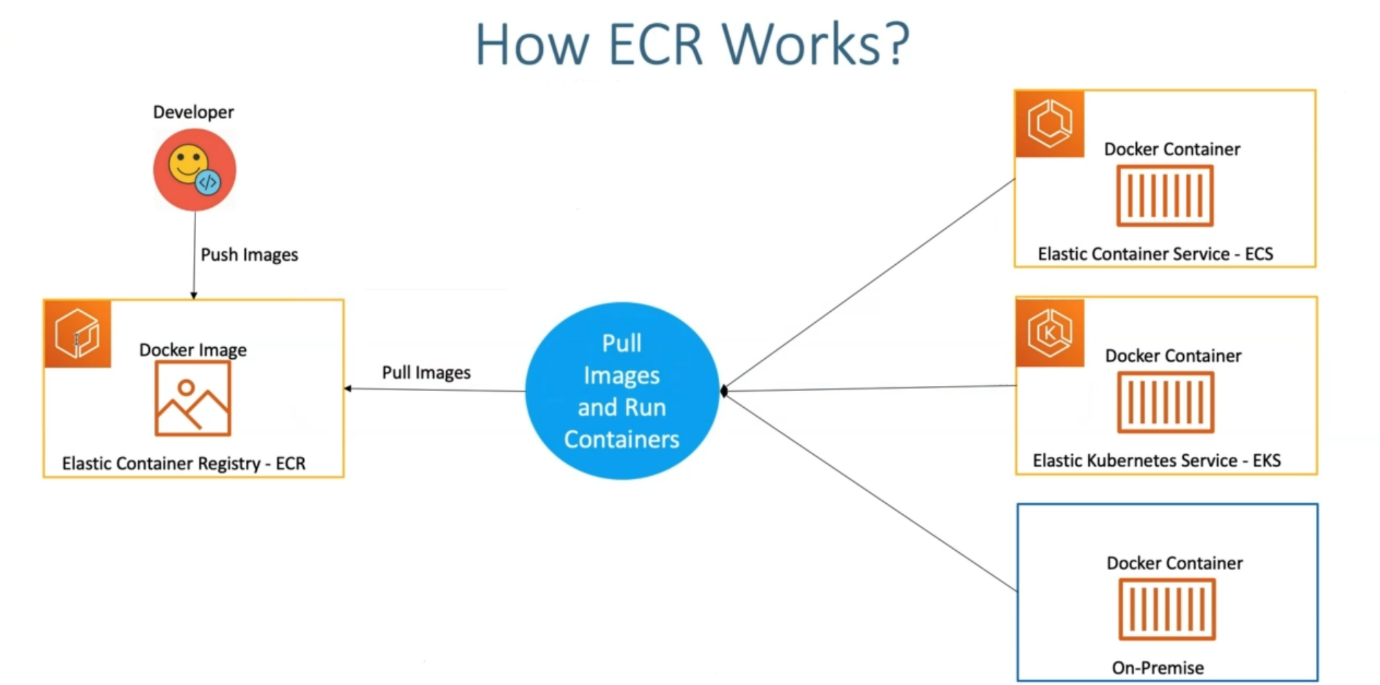


**ECR Benefits**

--- let's get a little bit background about what is this elastic container registry.



**How ECR works**



--- So, let's say I am a docker developer, I am going to push it to the Docker image related elastic container registry. we usually push your docker image to docker hub repository, which is open.

--- if you're dealing with aws and then if you want aws specific only and then we don't want our things to go outside the aws then we can push our docker image to elastic container registry in aws.

--- this ECR images, whatever we pushed, can be used in different places.

1. One is elastic container service, which is ECS.
2. Elastic kubernetes service which is this course about.
3. On-premise environment, if you want.

--- how this is going to happened…? whatever deployment's you perform in these respective services. those are going to pull the images from this elastic container registry using the repository url, whatever it provides.

--- this is about how ECR works.

**AWS ECR - Elastic Container Registry Integration & EKS**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/10-ECR-Elastic-Container-Registry-and-EKS>

**What are we going to learn?**

--- We are going build a Docker image

--- Push to ECR Repository

--- Update that ECR Image Repository URL in our Kubernetes Deployment manifest

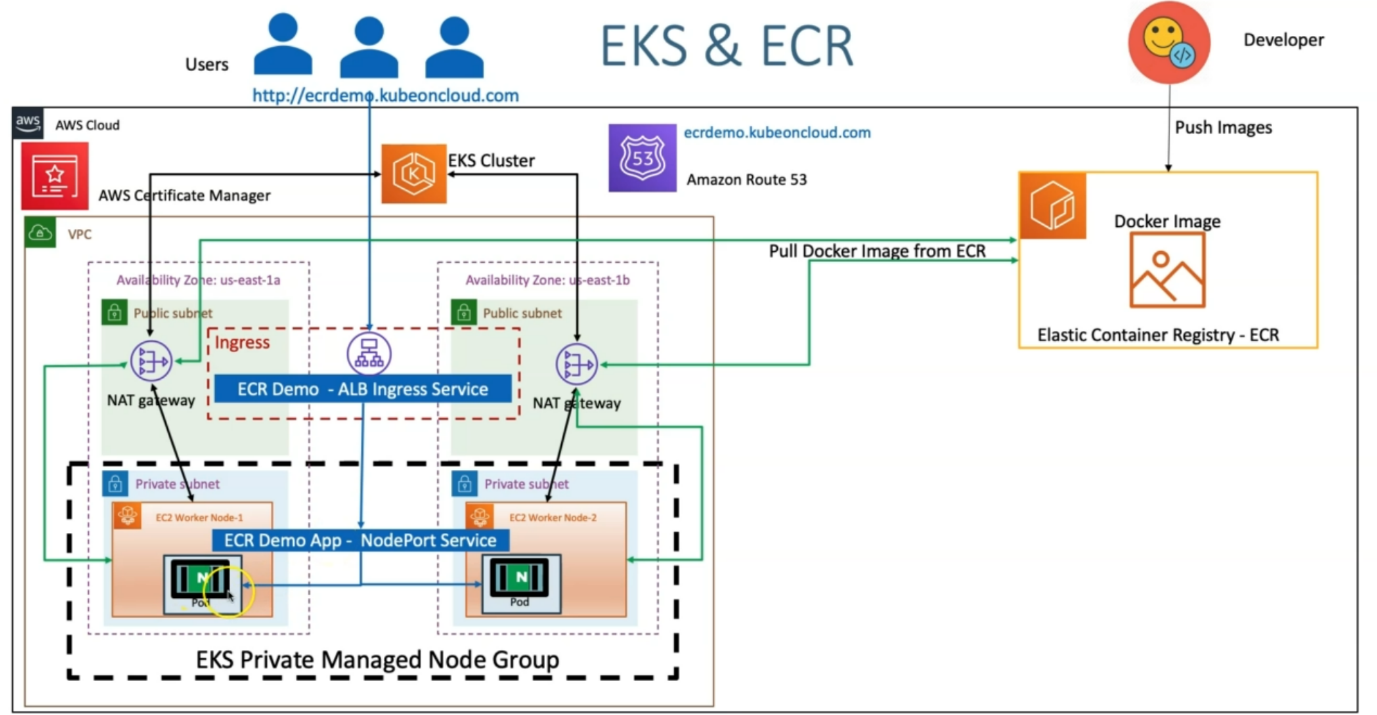
--- Deploy to EKS

--- Kubernetes Deployment, NodePort Service, Ingress Service and External-DNS will be used to depict a full-on deployment

--- We will access the ECR Demo Application using registered dns http://ecrdemo.kubeoncloud.com

--- **note** – lets see above description in visual manner.

**Architectural diagram**



--- in an aws cloud, I created Docker image and then push it to an elastic container registry.

--- from EKS cluster perspective, whenever we create EKS cluster. It creates its own VPC public subnets and then nat gateways in respect to availability zones. Even the Private subnets.

--- the managed node groups, which we are going to use is a private managed group. we are going to create in its private subnets.

--- as soon as we created that node group, it enables the communication with EKS cluster using the nat gateless deployed in public subnets.

--- these EC2 worker nodes. where you are going to deploy your workload on kubernetes.

--- the outbound connection to EKS cluster via nat gateway, in the same way if you need to pull the docker image.

--- So, you have done a deployment now and then to pull a Docker image, for that deployment. It needs to go via nat gateway and then go and then pull the Docker image from ECR.

--- In addition to that equilent ECR demo app nodeport service and ALB ingress service will

get created.

--- So as part of ALB ingress service, it also needs to reference the certificate from certificate manager and also it needs to register ecrdemo.kubeoncloud.com inside **ROUTE53** service.

--- as users, whenever you are planning to access ecrdemo.kubeoncloud.com, it will resolve to the ALB ingress service and from there it will come to the nodeport service. From there, the request will go to the respective ECR demo pod.