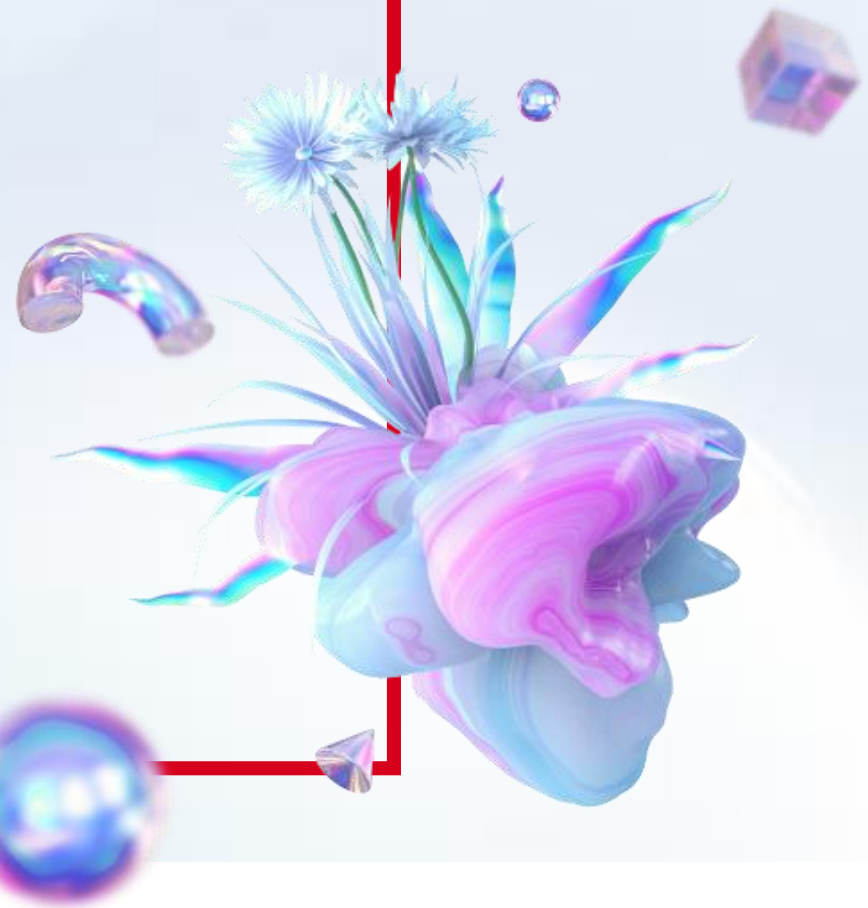
Elastic Kubernetes Service (EKS) and Elastic Container Registry (ECR)



Cao Thi Hoang Le

Jul-2023

Nash Tech.

Agenda

Elastic Kubernetes Service (EKS)

- What is AWS EKS?
- EKS architecture
- Why do we use EKS
- Limitations Of AWS EKS
- EKS Workflow
- Approach setup the worker nodes
- Setup EKS cluster

Elastic Container Registry (ECR)

- What is ECR?
- Components used in ECR

Demo

- Create EKS using GUI
- Create EKS Fargate using eksctl
- Deploy microservices to EKS

Elastic Kubernetes Service (EKS)

What is AWS EKS?

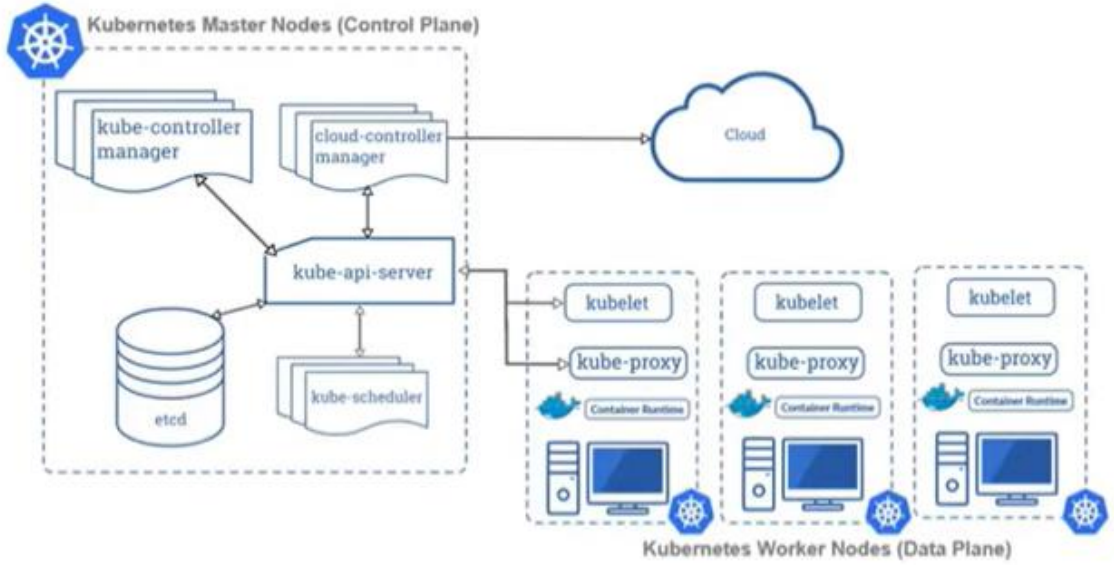
AWS EKS stands for Amazon Web Services Elastic Kubernetes Service and EKS is managed by Kubernetes service at Amazon Web Services.

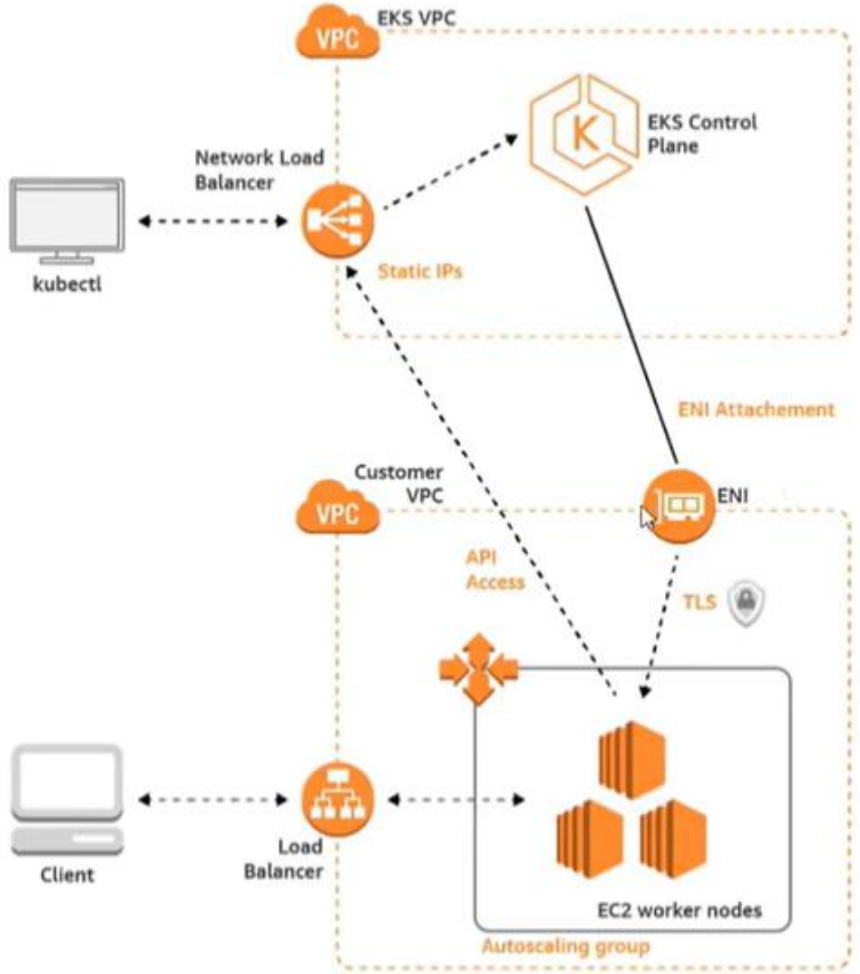
It provides the flexibility of Kubernetes with the security and resiliency of being an AWS-managed service.

Since AWS is now responsible for your control plane you no longer have to worry about that you don't have to touch it. The only thing that you have to worry about this worker nodes

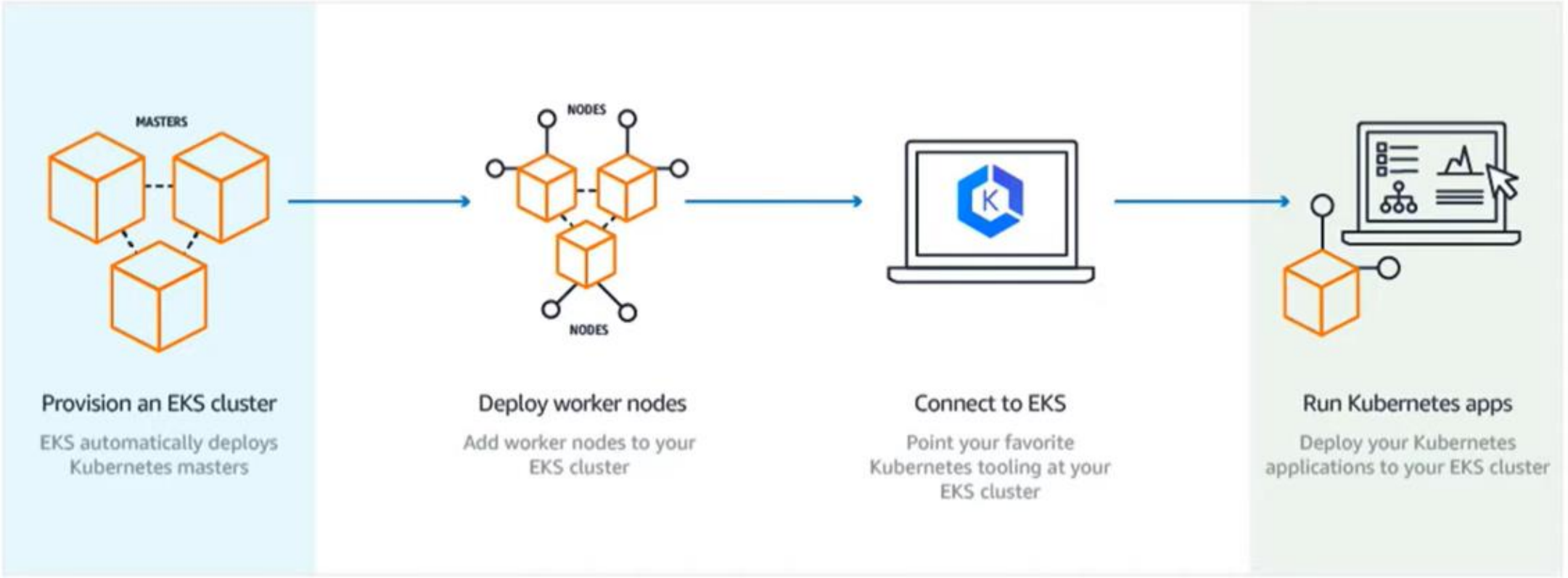
- AWS manages the control plane of you Kubernetes cluster
- Provisioning/maintaining master nodes
- Install Control plane processes
  - ✓ API Server
  - ✓ Scheduler
  - ✓ Controller Manager
  - ✓ Etcd

EKS Architecture





EKS Workflow



Why do we use EKS?

- The reasons why you should choose AWS EKS to manage Kubernetes for your operations team are the following

Running & Scaling	Running and scaling Kubernetes can be difficult and can require significant investment. Hence, you should choose AWS EKS for the job.
Securing Kubernetes	Securing Kubernetes increases the operational overhead of running applications.
Integration with Other AWS services	An application needs a native way to integrate with other AWS services securely and reliably, which AWS EKS provides for you

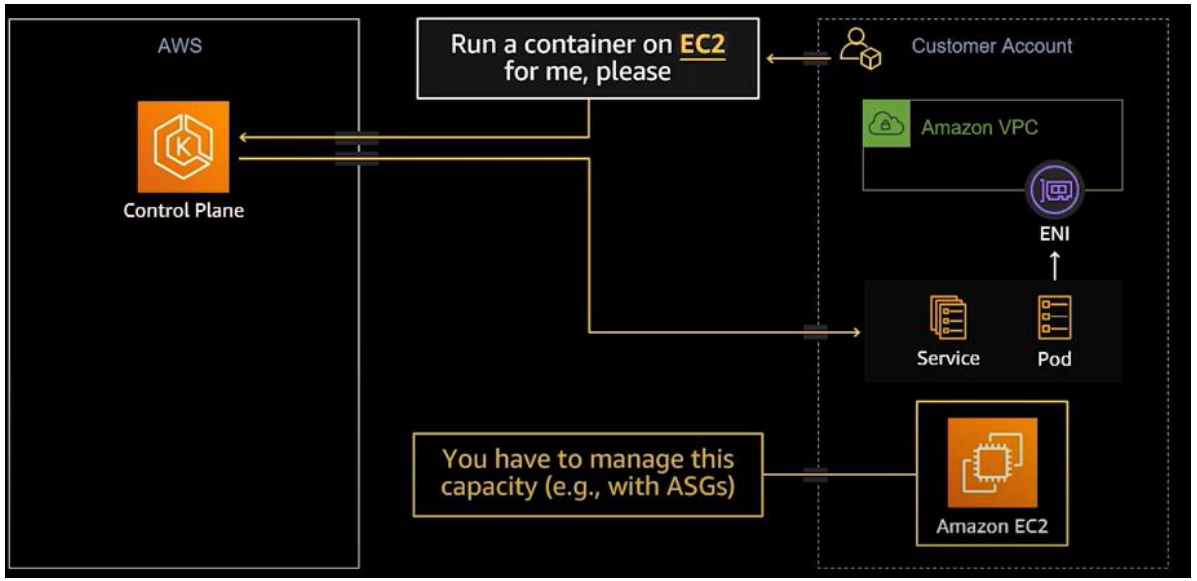
Limitations Of AWS EKS?

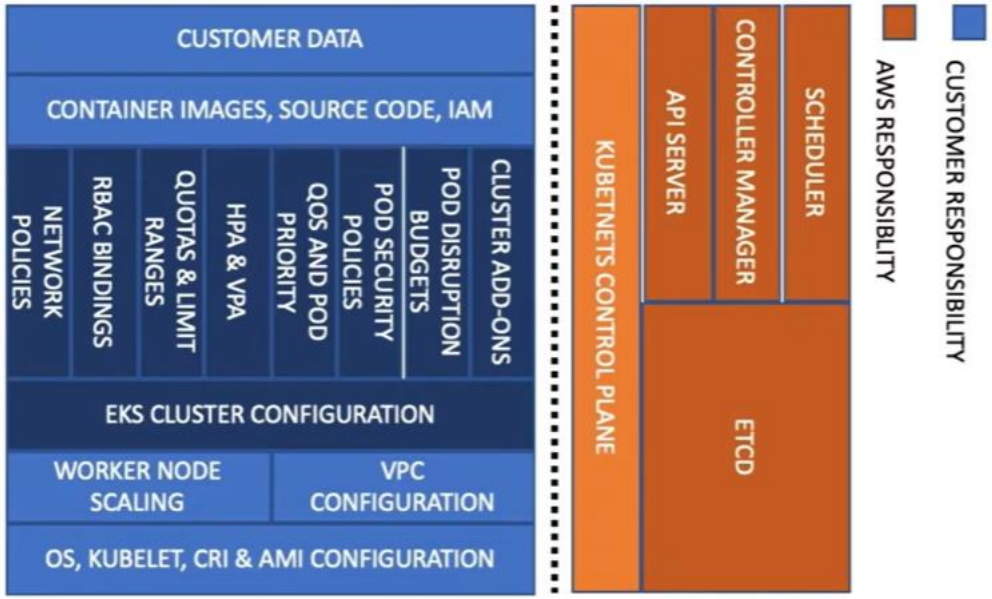
- Even though AWK EKS is a magnificent service, we will confess it isn't perfect, to be completely honest with you. Hence, the following are the biggest limitation AWK EKS possess

Expensive	AWS Kubernetes provide you with incredible service on all fronts. It is remarkable to use, but this comes with a hefty price tag as compared to other Kubernetes services
Manual Service Updates	Every service update in AWS EKS needs to be performed manually to use new features they introduce, as this service does not have an auto-update feature.
AWS Internal Integration To Be Performed By You	The final drawback of AWS Kubernetes is that although you can integrate it with other AWS services and use it to your advantage, you have to do all of this yourself. There is no automation setup, and you must perform every function you wish to achieve manually.

Approach setup the worker nodes (1/3)

- **Self-managed nodes – EKS with EC2**

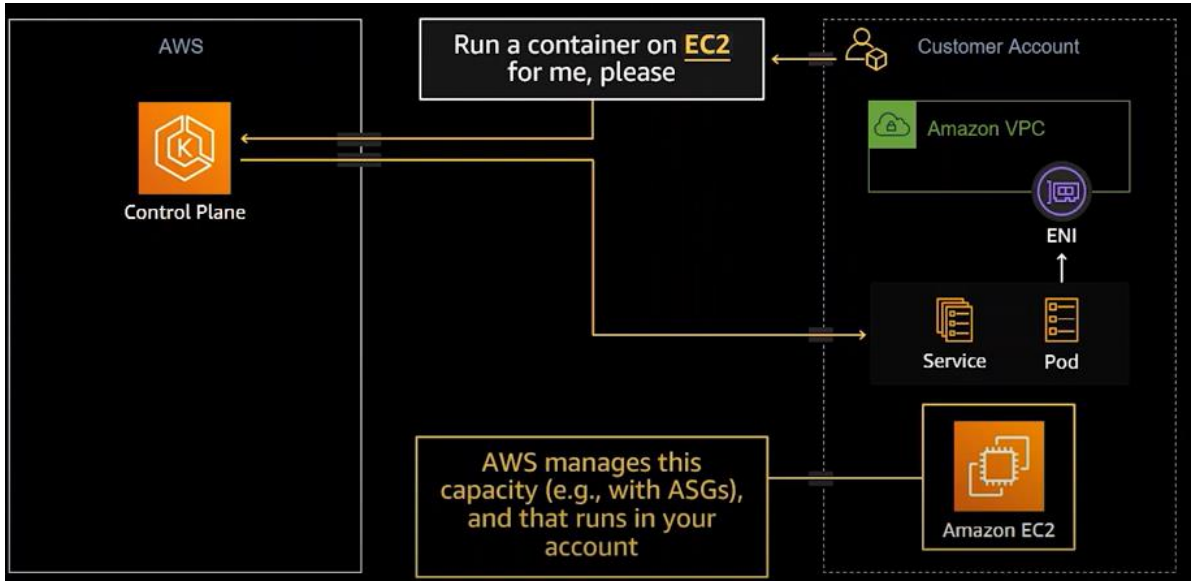


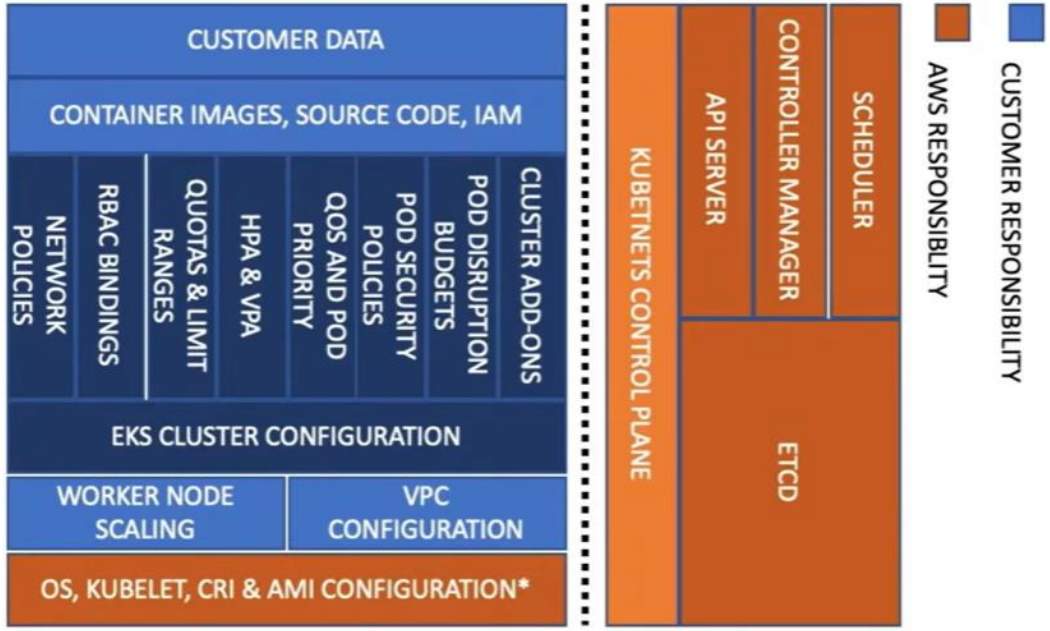


You have full control on both the applications and underlying infrastructure in the form of virtual machines.

Approach setup the worker nodes (2/3)

- **EKS with managed node groups**

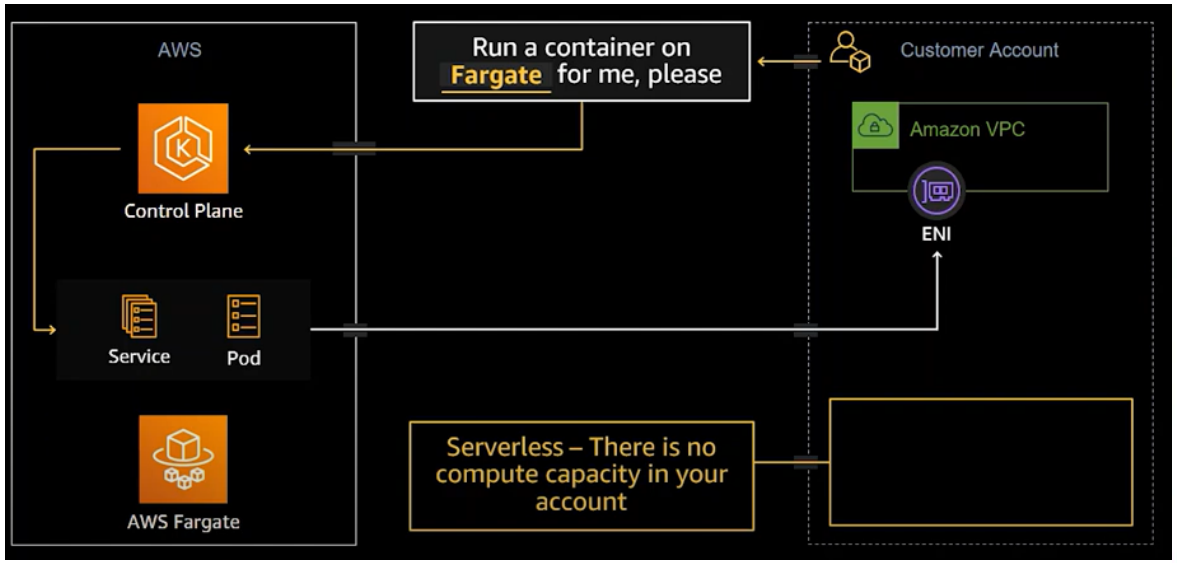


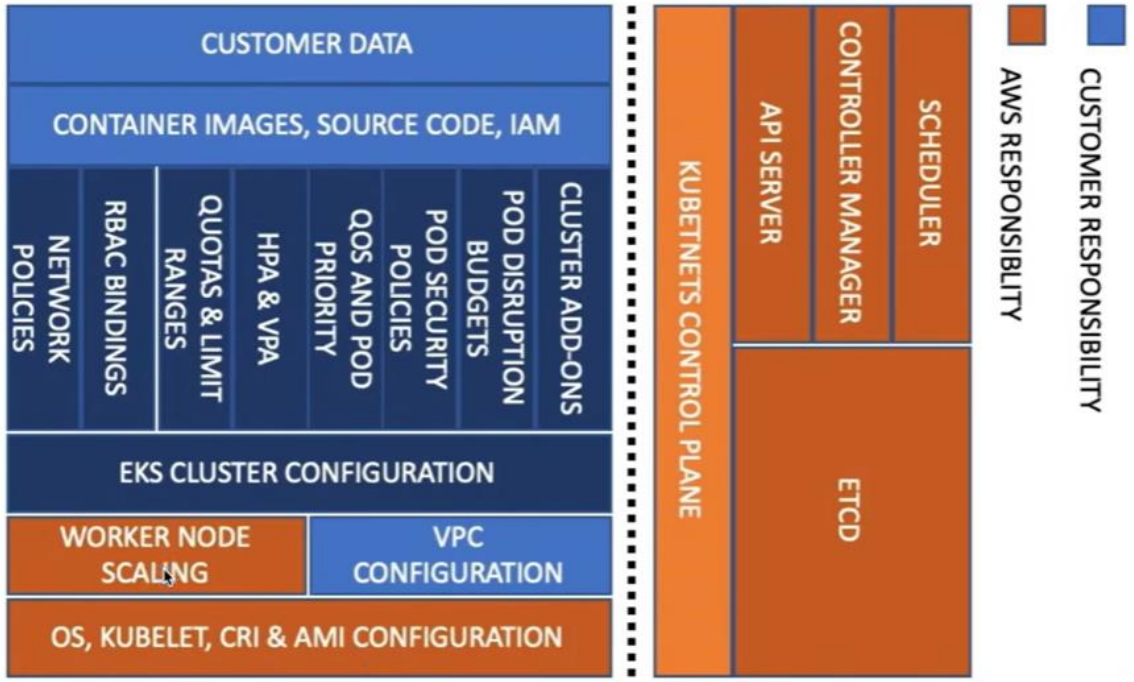


You have control on both the applications and underlying infrastructure and AWS helps you in some of the operational heavy lifting that comes with virtual machines

Approach setup the worker nodes (3/3)

- **EKS with Fargate**





You have control on the applications, and you DON'T have control over the infrastructure that is fully managed for you by AWS

Setup EKS cluster (1/2)

- To create an EKS cluster there are going to be a couple of things that you are going to have to configure

Setup EKS cluster (2/2)

Use AWS Console	• Create cluster • Create worker nodes
Use Eksctl	• Eksctl is CLI for Amazon EKS • Setup your cluster with just a single command `eksctl create cluster`
IaC – Terraform/Pulumi/AWS Cloudformation	• Define your infrastructure configuration in code and then you can just deploy it

Elastic Container Registry (ECR)

What is AWS ECR?

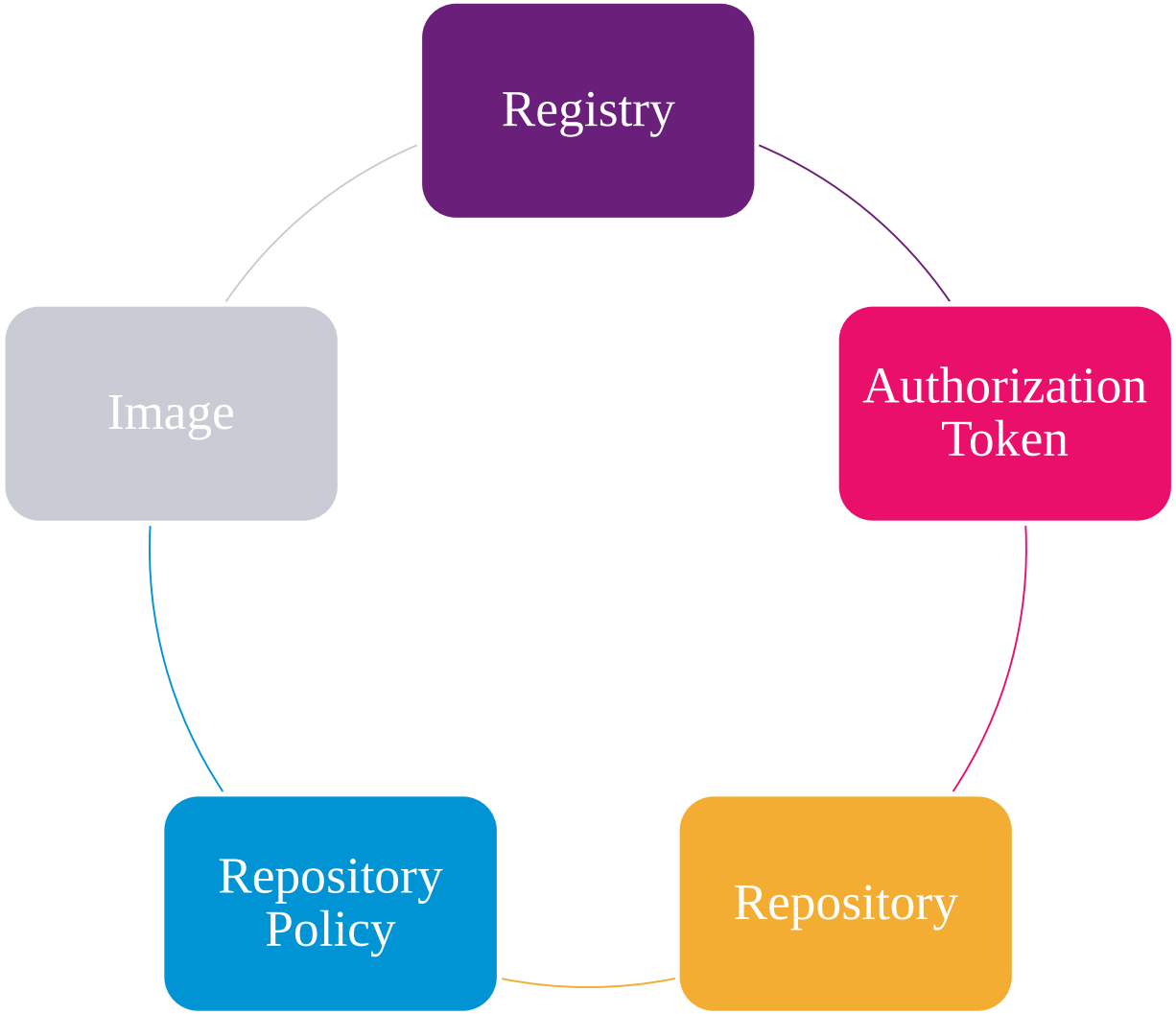
It’s managed Container image registry service. It like Docker hub but for AWS

It provides the flexibility of Kubernetes with the security and resiliency of being an AWS-managed service.

ECR has public and private registry. Each registry you can have many repositories. Each repository can have many images. Images can have several tags and these tags need to be unique within your repository

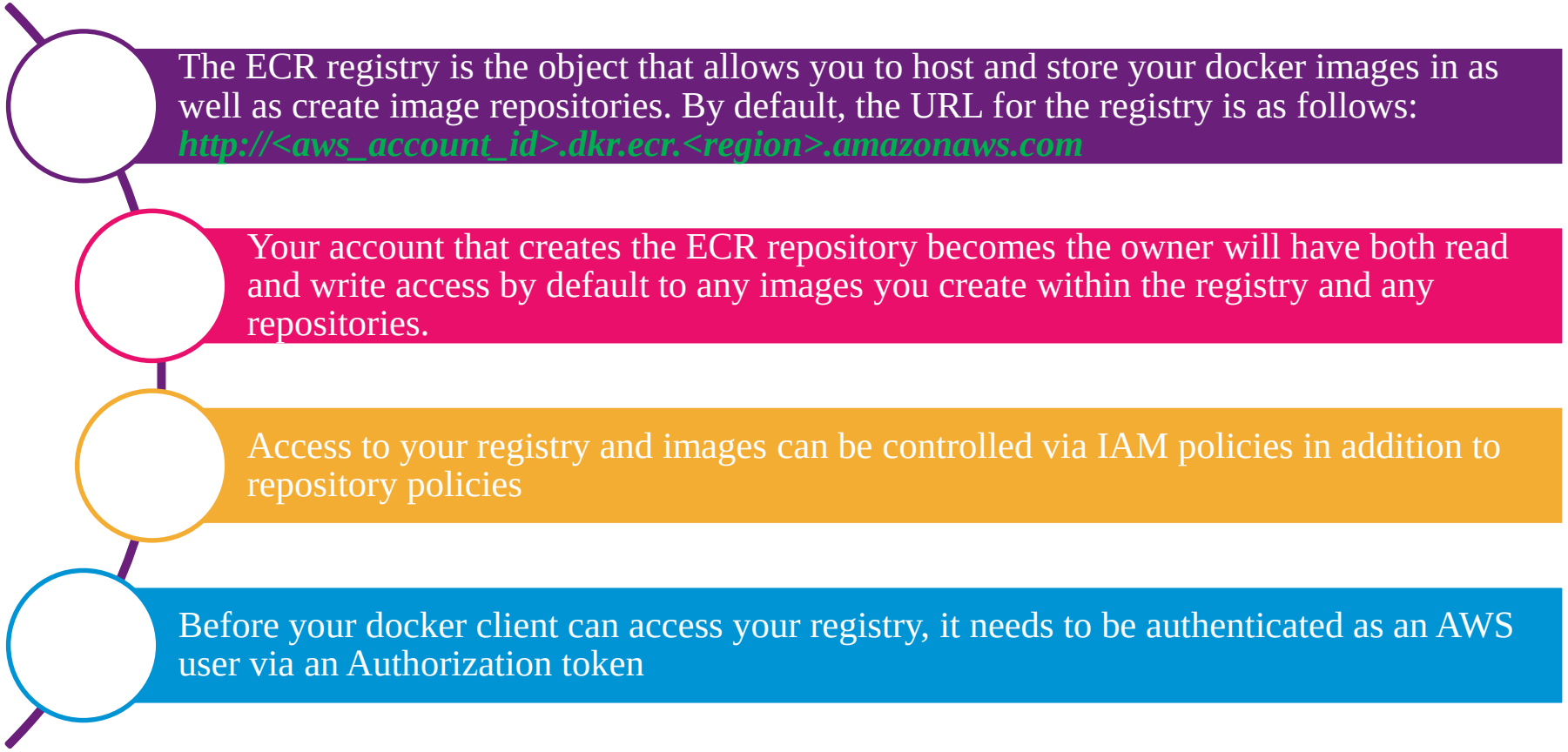
Public: Anyone can have read-only(pull) access to anything within that registry, but read-write (push) requires permission
Private: Permission are required for any read or any read-write operations

Components used in ECR (1/6)



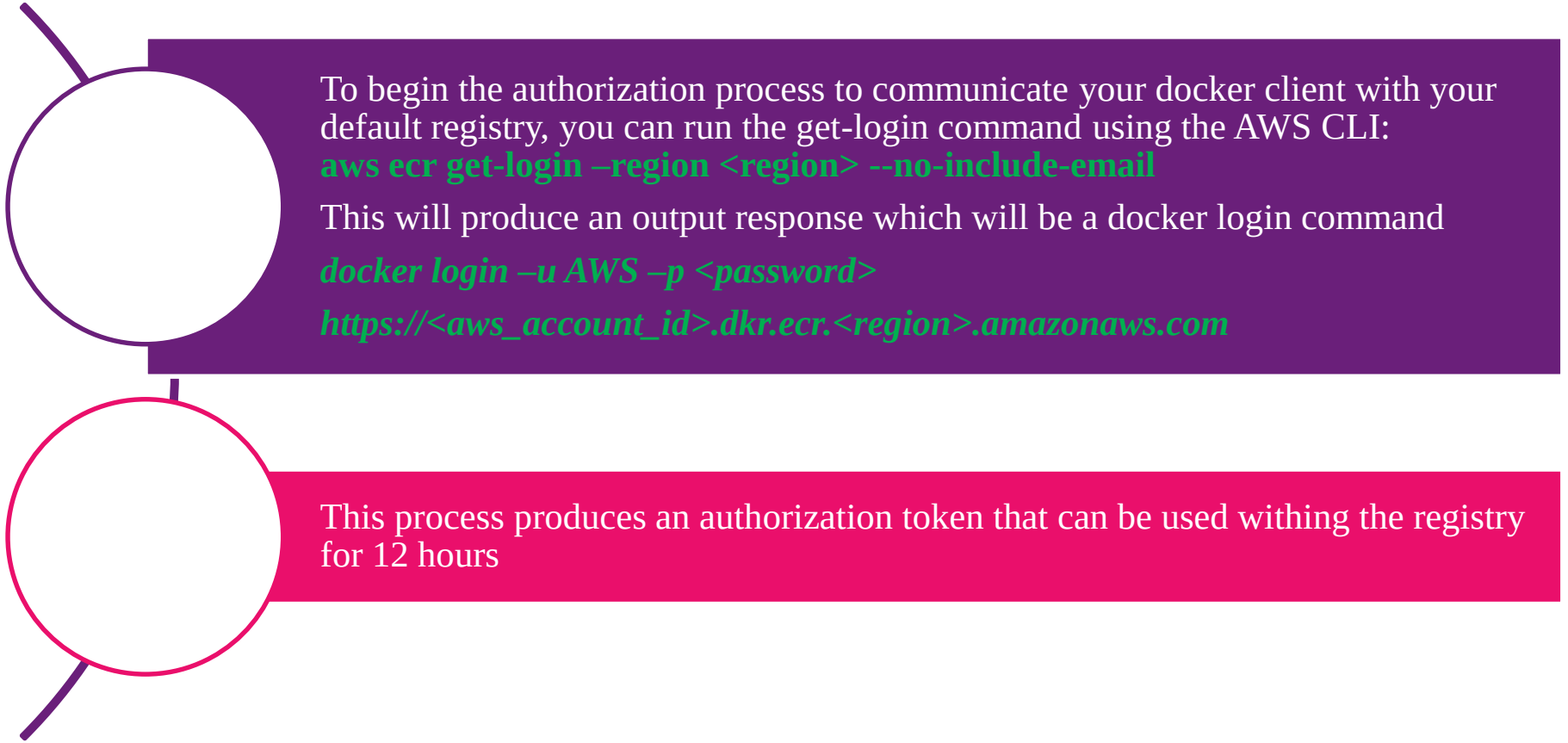
Components used in ECR (2/6)

- **Registry**



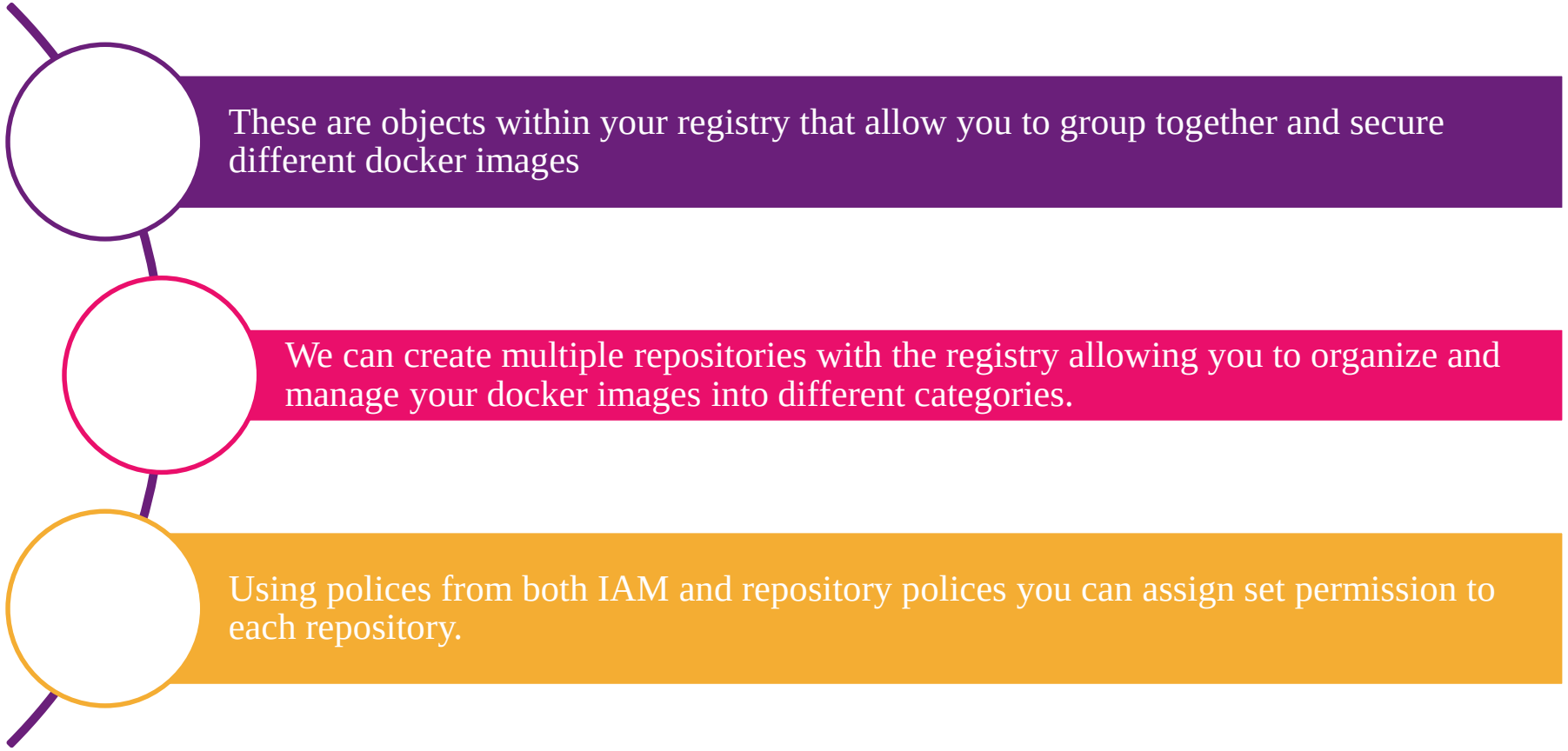
Components used in ECR (3/6)

- **Authorization Token**



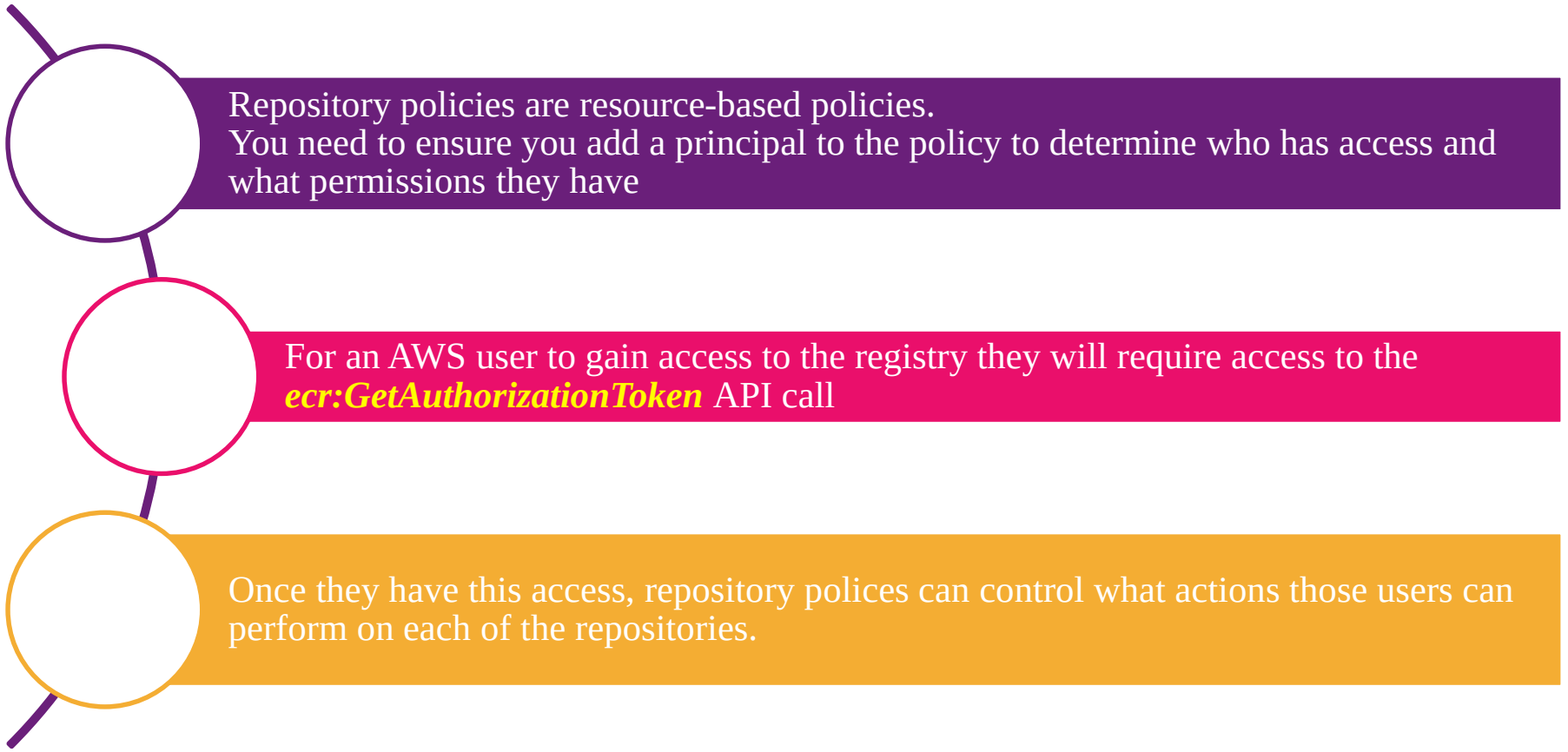
Components used in ECR (4/6)

- **Repository**



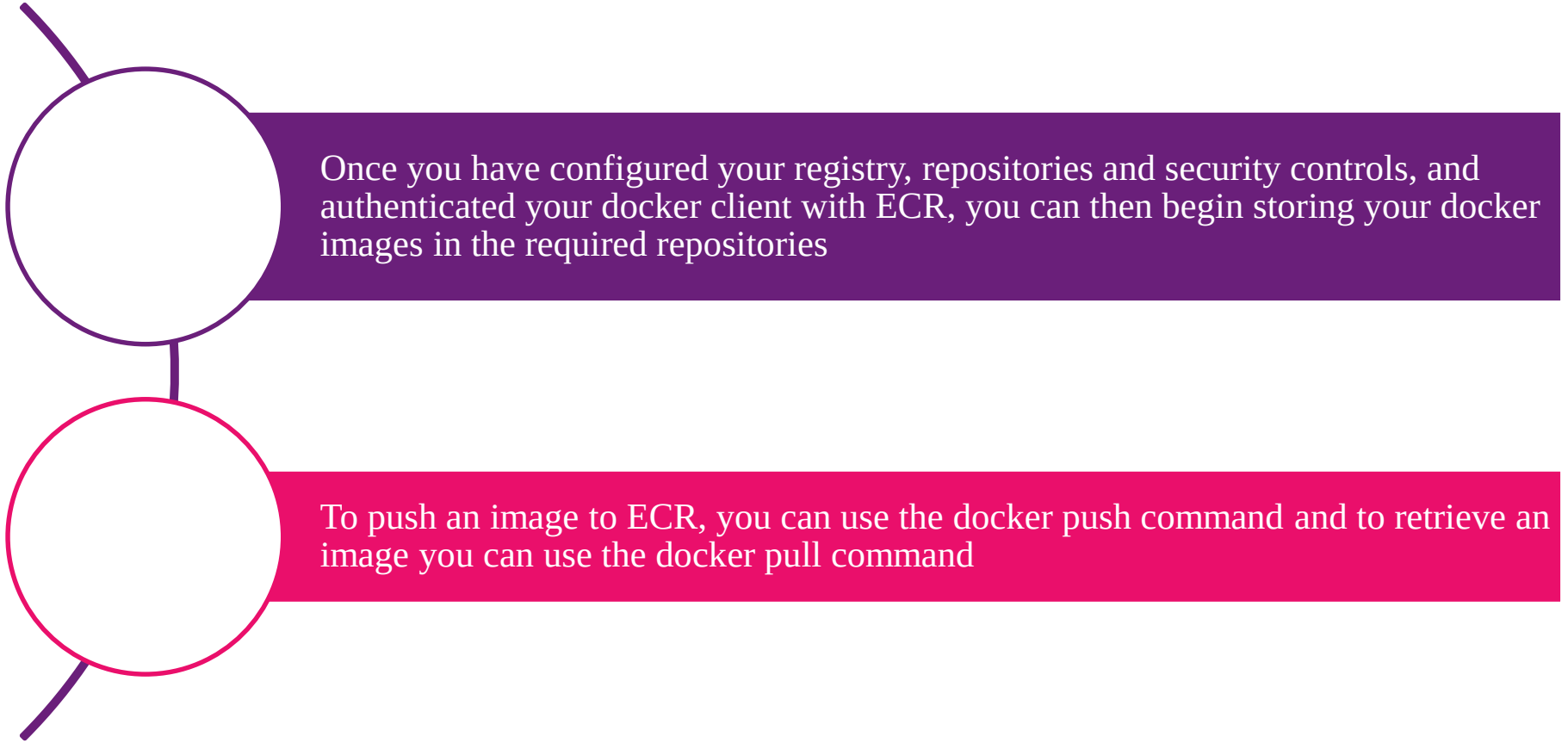
Components used in ECR (5/6)

- **Repository Policy**



Components used in ECR (6/6)

- **Image**



Demo

EKS

- Create EKS using GUI
- Create EKS Fargate using eksctl
- Create EKS Fargate using eksctl
  - Refer to EKS_HandsOn file
- Deploy Application on EKS
  - Refer to Application_deployed_EKS.pdf file

ECR

- Push Docker Images into Private AWS ECR
  - Refer to ECR_HandsOn file

Thank you

Nash Tech.