



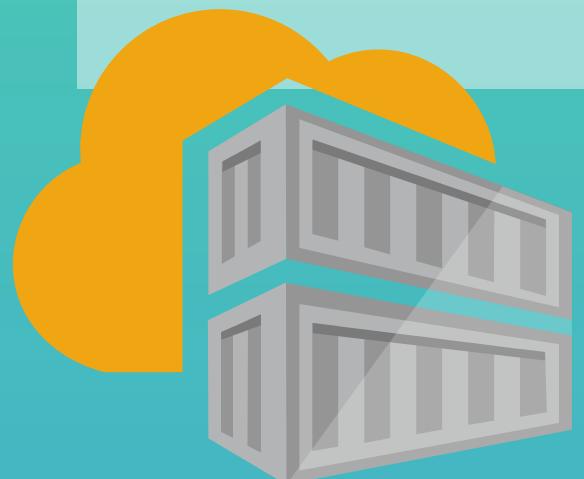
Key Takeaways

Kubernetes on
AWS

Container Services on AWS

Elastic Container Registry (ECR)

- Private Docker Repository



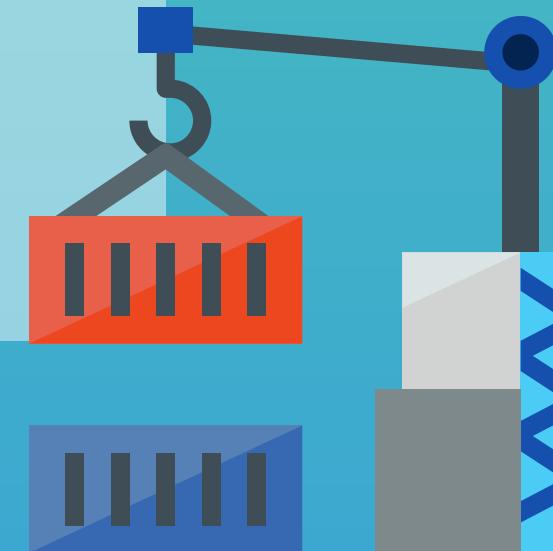
Elastic Kubernetes Service (EKS)

- Container Orchestration Service
- Managed Kubernetes Service



Elastic Container Service (ECS)

- Container Orchestration Service



Elastic Container Service

Different Container Orchestration Tools



Amazon's Elastic Container Service is one of several container orchestration tools

How does ECS work? - 1

- Control Plane:
managed by AWS
- Compute Fleet:
Hosted on EC2 instances, connected
and managed by ECS



EC2 instance needs to have:

1. Container Runtime
2. ECS Agent - for Control Plane communication

Using EC2 vs AWS Fargate

ECS hosted on EC2 Instances

- ✓ Containers managed by AWS
- ✓ Full access and **control** of your infrastructure
- ✗ You **still need to manage** and maintain the **virtual machine**
- ✗ Pay for **whole server**

ECS hosted on AWS Fargate

- **Serverless** way to launch containers
- ✓ Containers managed by AWS
- ✓ Infrastructure managed by AWS
- ✓ Provisions servers **on demand**
- ✓ Pay only for **what you use**
- ✓ Easily **scales up/down without fixed resources** defined beforehand



Elastic Kubernetes Service (EKS).

Amazon Elastic Kubernetes Service



Managed Container service to run and scale K8s applications

Difference between ECS and EKS



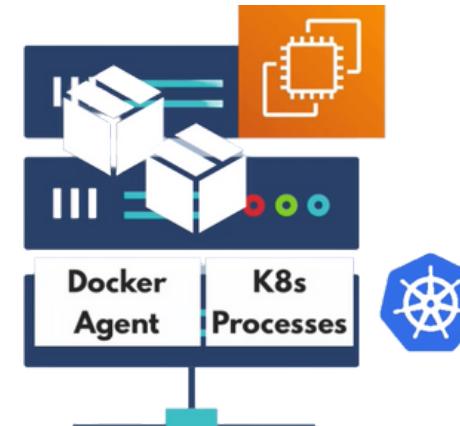
- Specific to AWS
- Migration difficult

- ✓ Less complex applications
- ✓ ECS control plane is free

ECS = via ECS Agent



EKS = via K8s Worker Processes



- Both: Managing the Control Plane
- Communication between Control Plane and Compute Fleet:



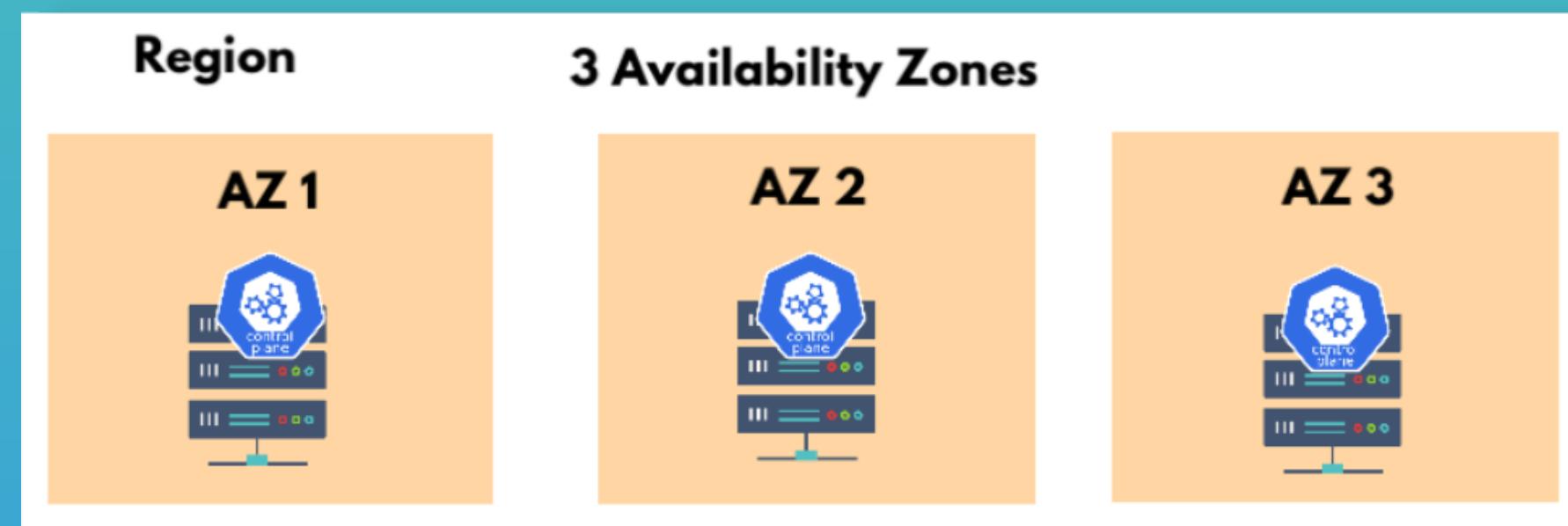
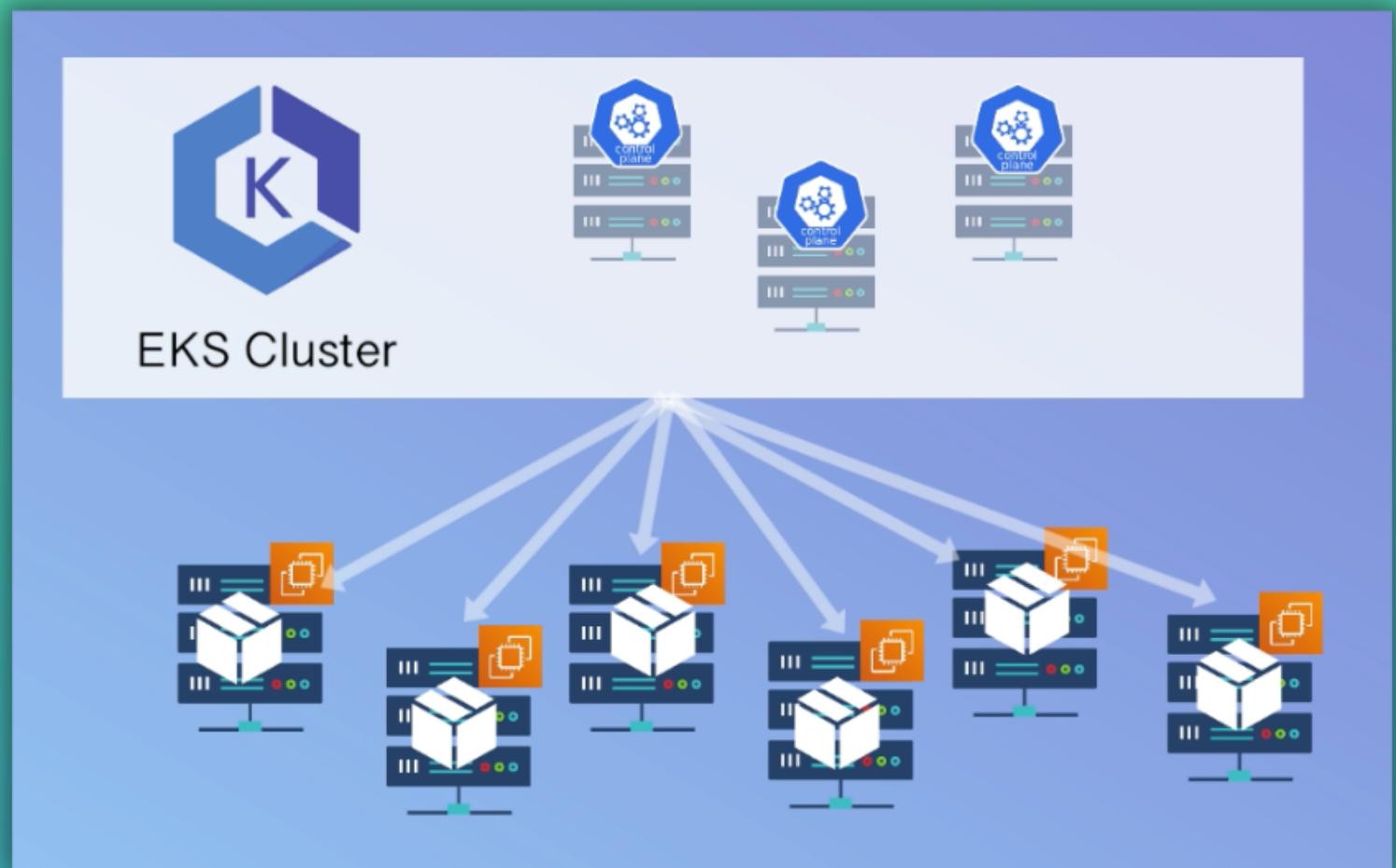
Amazon EKS

- ✓ You already use K8s (**same K8s API**)
- ✓ K8s is open-source
- ✓ Easier to migrate to another platform
- ✓ Large community (Helm charts etc.)

How EKS works - 1

- **Control Plane:** Scheduling and Orchestration
- **Worker Nodes:** Compute Fleet

- ✓ EKS deploys and manages K8s Control Plane Nodes
- ✓ K8s Control Plane services are already installed
- ✓ **High Availability:** Control Plane Nodes replicated across Availability Zones:



How EKS works - 2

Different choices to host your Worker Nodes

EKS with EC2 Instances

self-managed

- You need to manage the infrastructure of Worker Nodes

EKS with Nodegroup

semi-managed

- Creates, deletes EC2 instances for you, but you need to configure it

EKS with Fargate

fully-managed

- Fully managed Worker Nodes

High-Level Steps to create an EKS cluster



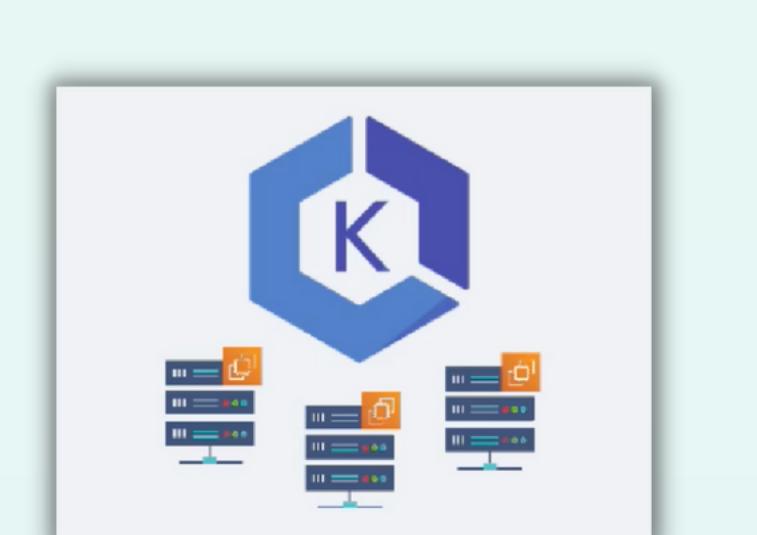
1)

Provision an EKS
cluster (**Control Plane
Nodes**)



2)

Create Nodegroup
of EC2 Instances
(**Worker Nodes**)



3)

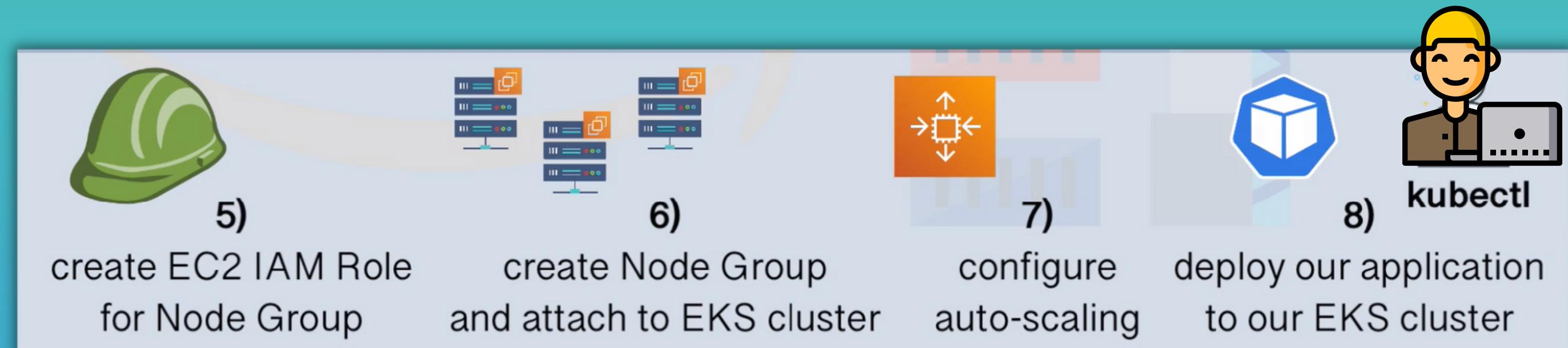
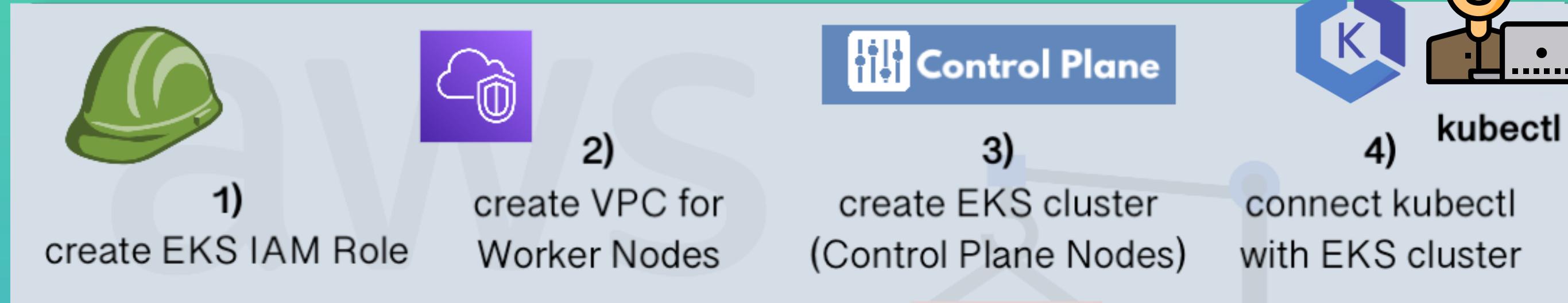
Connect
Nodegroup to
EKS cluster



4)

Deploy your
containerized
applications

Detailed Steps to create an EKS cluster - 1



Detailed Steps to create an EKS cluster - 2

1) Create IAM Role for EKS cluster

- 1.Create IAM Role in AWS account
- 2.Assign Role to EKS cluster managed by AWS

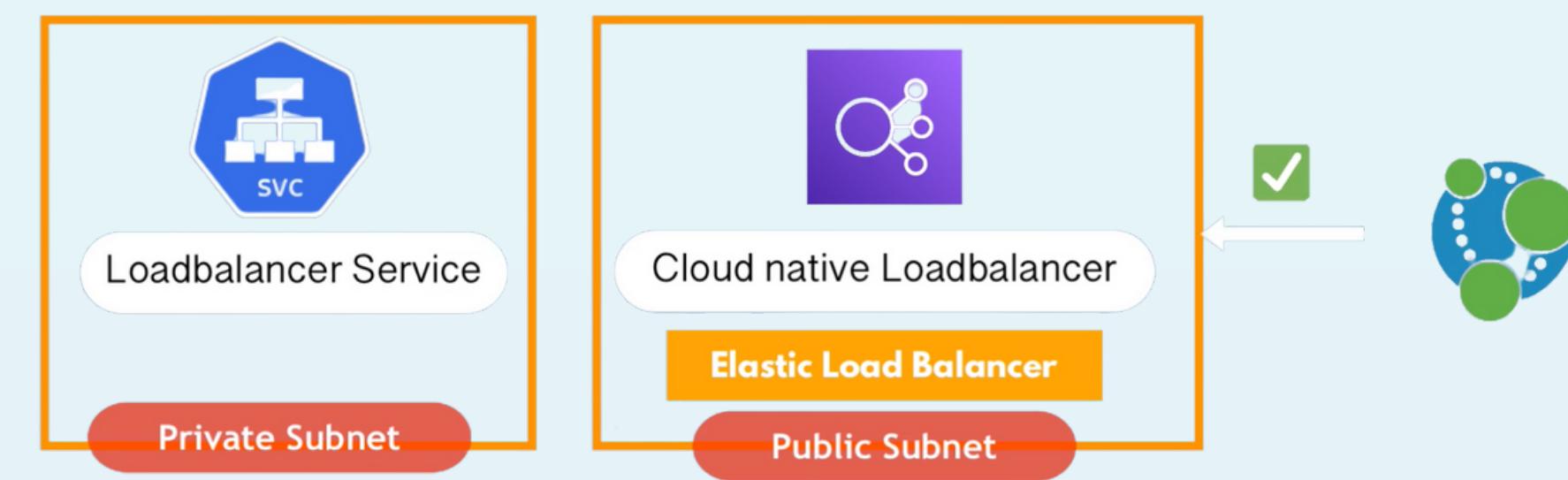
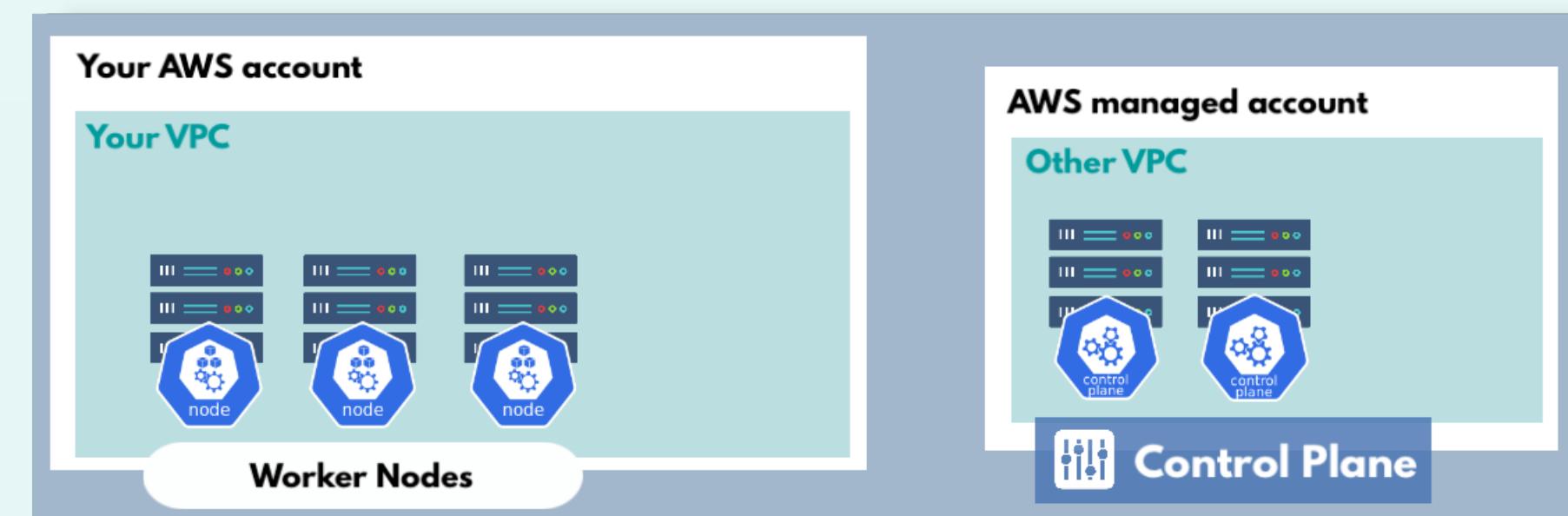
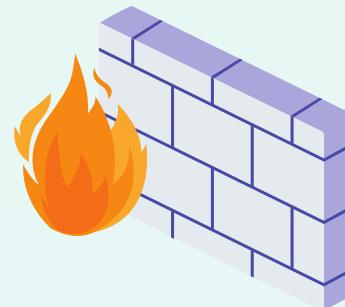
Why? To allow AWS to create and manage components on our behalf



Detailed Steps to create an EKS cluster - 3

2) Create VPC for EKS Worker Node

- EKS cluster needs specific networking configuration
- Worker Nodes need specific Firewall configurations for Control Plane-Worker communication
- Best Practice = Configure **public** and **private** subnet
- Through IAM Role you give K8s permission to change VPC configurations



Detailed Steps to create an EKS cluster - 4

3) Create EKS cluster

Manually



- Difficult to replicate
- Or do multiple times

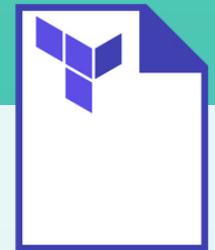
Options to create the cluster

eksctl



- **Command Line Tool** for working with EKS clusters that automates many manual tasks
- Execute just 1 command
- Necessary resources get created and configured in the background
- Cluster will be created with default parameters
- But you can customize your cluster with CLI options

IaC



- Use Infrastructure as Code tool like Terraform



You will learn this in Terraform Module

Detailed Steps to create an EKS cluster - 5

5) Create EC2 IAM Role for Node Group

- Kubelet is the **main worker process** - scheduling, managing Pods and **communicate with other AWS services**
- That's why Kubelet on Worker Node needs **permission**
- Create **Role for Node Group**



With Node Group all necessary worker processes (container runtime, kubelet, k-proxy) are installed

Elastic Container Registry (ECR) - 1

Repository for Container Images

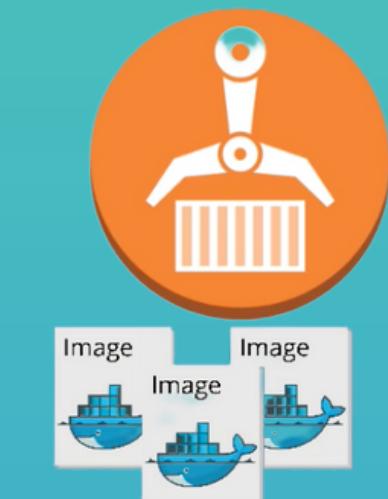
- Store, manage and deploy container images
- Amazon's alternative to:



- Integrates well with other AWS services



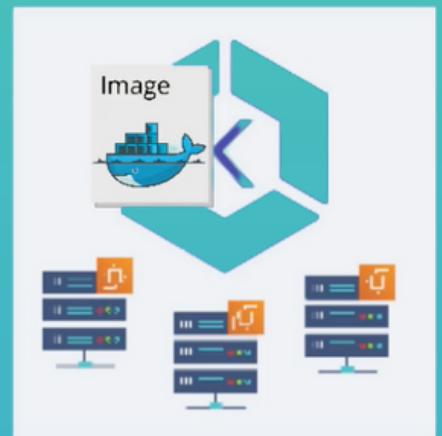
repository



your App Images
stored here

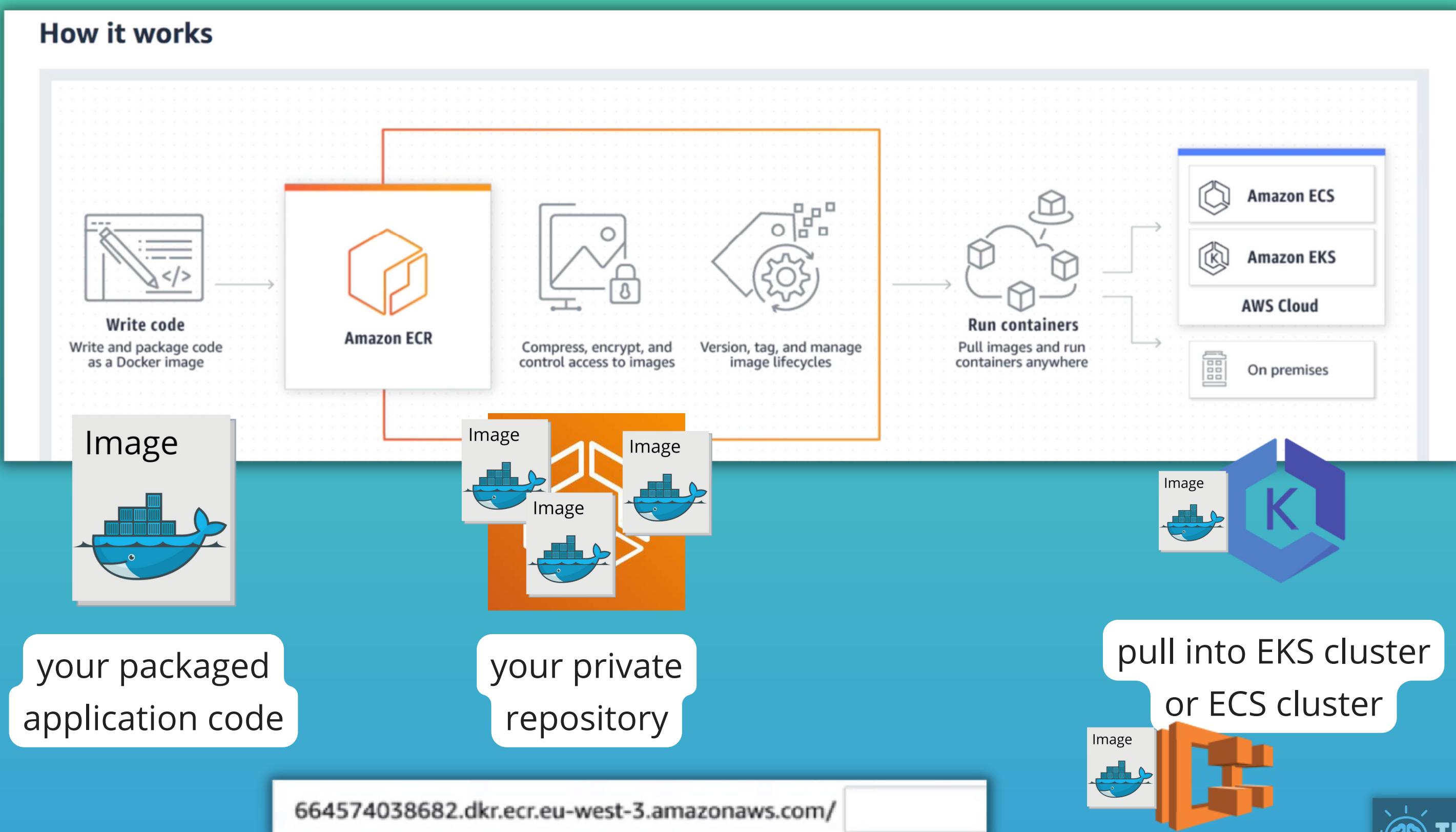
notify when new image
pull images

easy to connect and configure



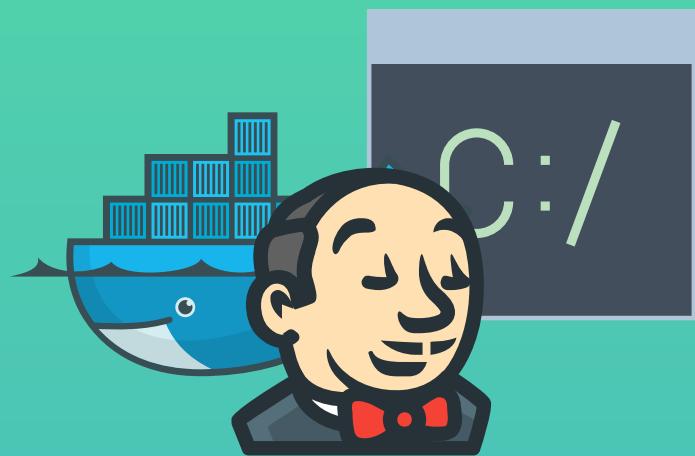
Elastic Container Registry (ECR) - 2

How it works:



CD - Jenkins & EKS

Deploy to EKS cluster from Jenkins Pipeline



1. **Install kubectl** CLI inside Jenkins container
2. **Install aws-iam-authenticator** tool inside Jenkins container
3. **Create kubeconfig** file to connect to EKS cluster
 - contains all the necessary information for **authentication**:

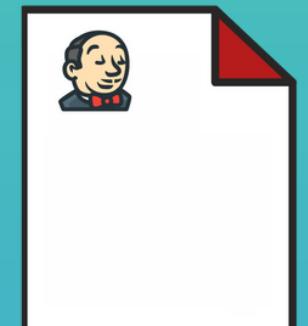
=> authenticate with AWS

=> authenticate with K8s cluster
4. **Add AWS credentials** on Jenkins for AWS account authentication
5. **Adjust Jenkinsfile** to configure EKS cluster deployment

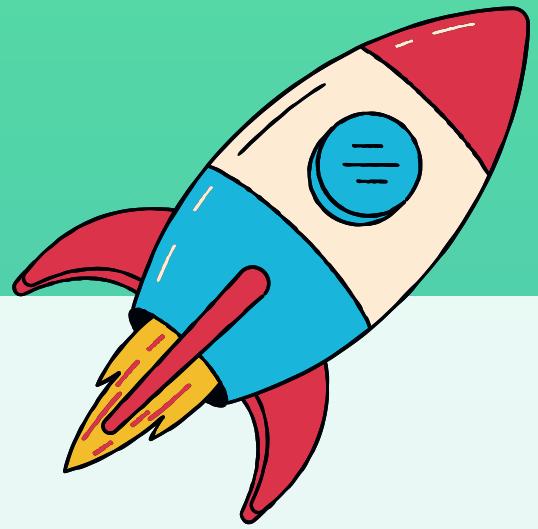


Best practice:

- 1) Create AWS IAM User for Jenkins
- 2) with limited permissions



Best Practices



- Create VPC with private and public subnet
- Security: Use AWS Key Management Service (KMS) keys to provide envelope encryption of Kubernetes secrets (<https://docs.aws.amazon.com/eks/latest/userguide/enable-kms.html>)

Official AWS best practices for EKS:

- <https://aws.github.io/aws-eks-best-practices/> - Each topic and recommendation is based on best practices implemented in production by AWS customers and validated by K8s specialists and the K8s engineering team at AWS.