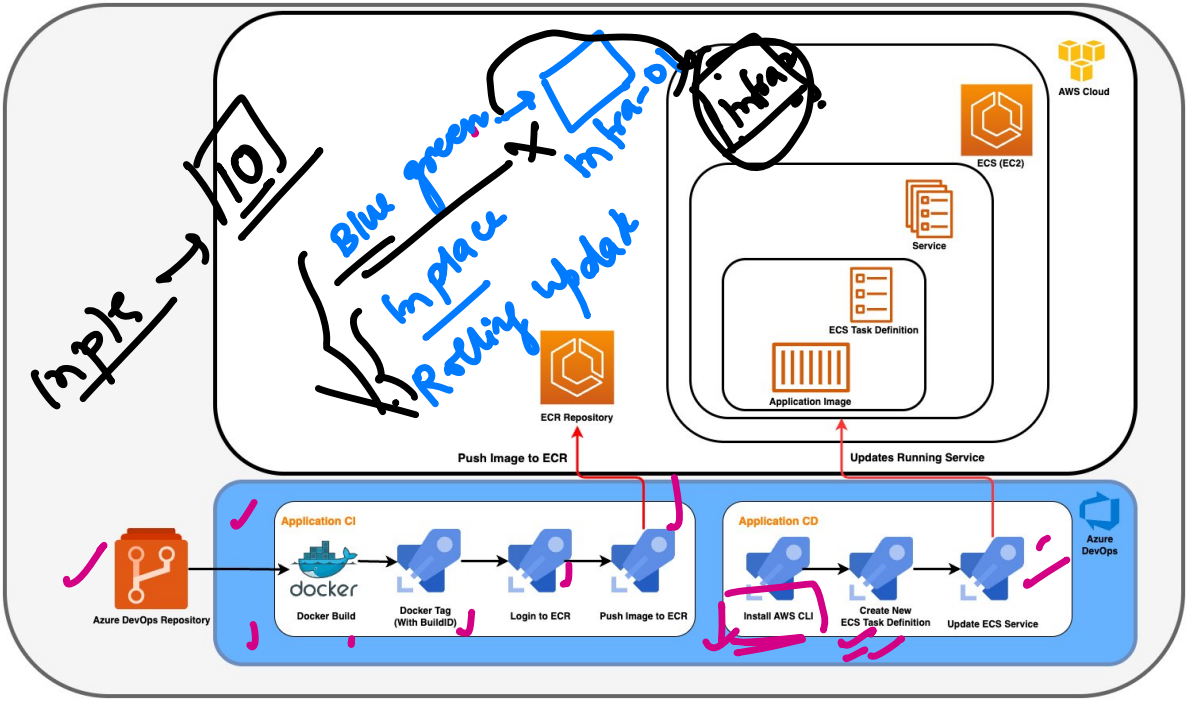
ECS Limitations that EKS Solves

ECS Limitations that EKS Solves

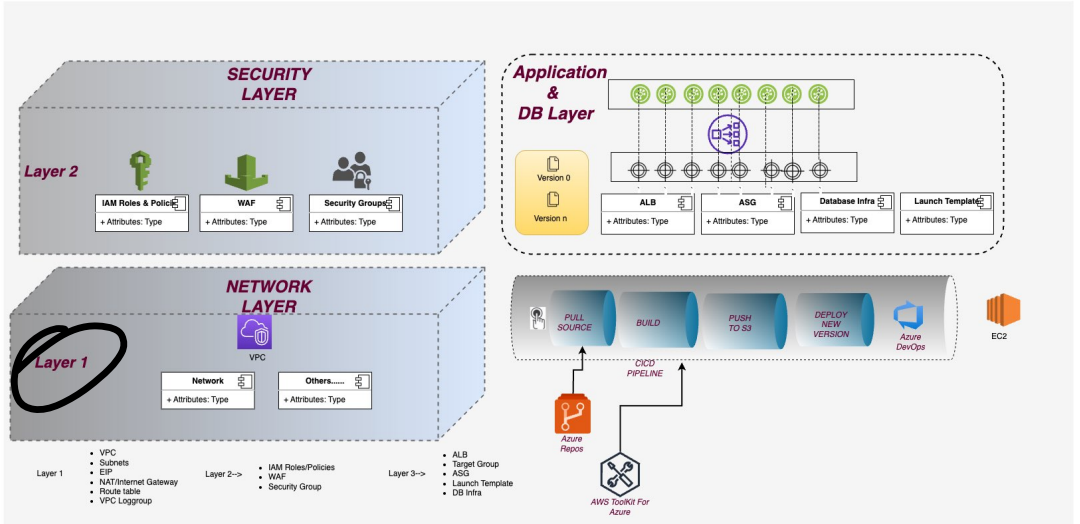
ECS Limitation	How EKS Solves It
AWS-specific orchestration	Standard Kubernetes API for multi-cloud compatibility
Limited ecosystem of tools	Rich ecosystem of Kubernetes tools and extensions
Less flexible networking	Advanced networking with CNI plugins
Simpler but less customizable	Highly customizable for complex workloads
Limited deployment strategies	Advanced deployment patterns (Blue/Green, Canary)

Agenda

- Ecs - Revision
- Limitations of Ecs
- Ecs - Pipeline - Sample explanation
- Eks - Def^n
- Components of Eks
- Control plane / Data plane
- Modes
- Eks
- Difference.

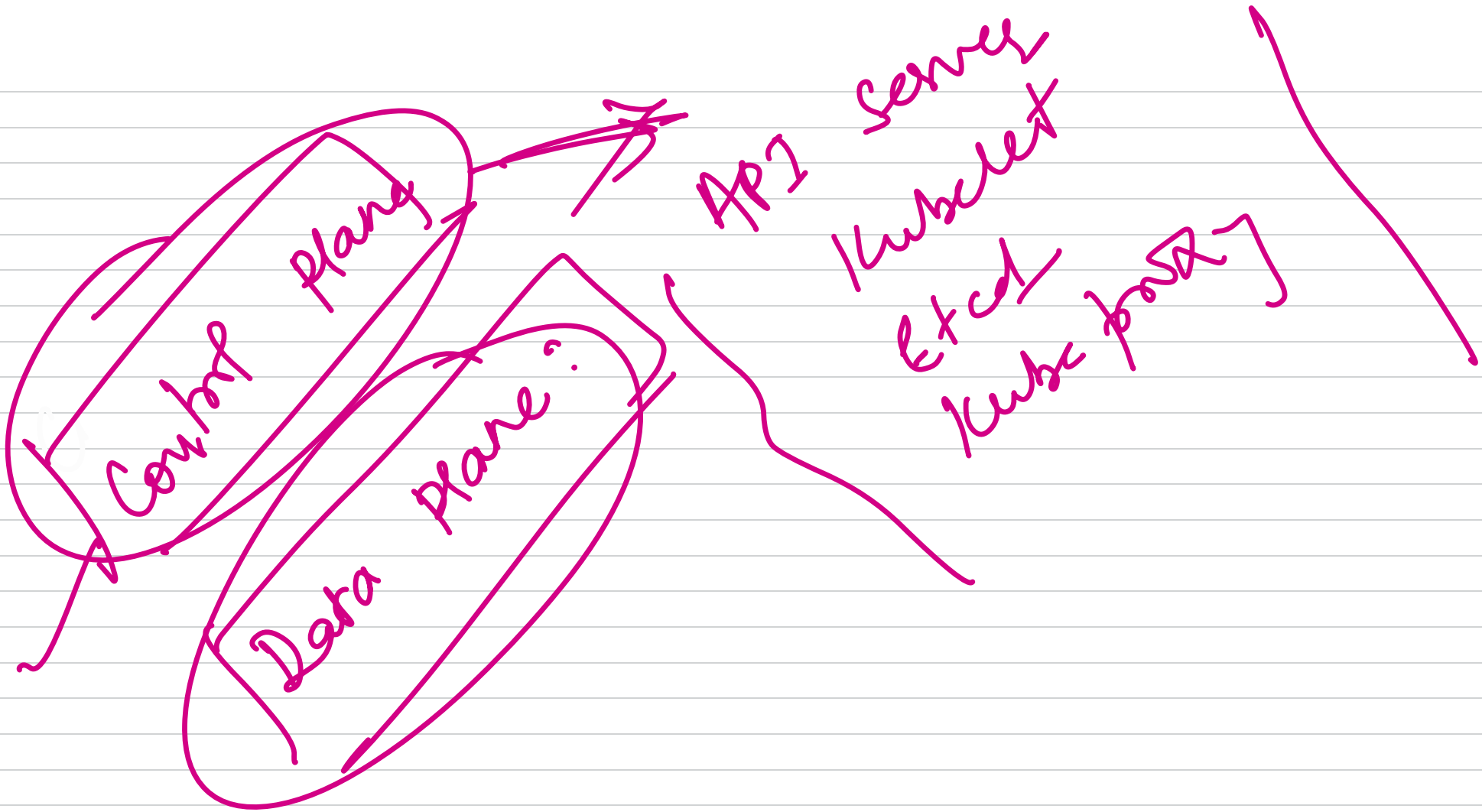

AWS Cloud
ECS (EC2)
Service
ECS Task Definition
Application Image
ECR Repository
Push Image to ECR
Updates Running Service
Application CI
Application CD
Azure DevOps
Azure DevOps Repository
docker
Docker Build
Docker Tag
(With BuildID)
Login to ECR
Push Image to ECR
Install AWS CLI
Create New
ECS Task Definition
Update ECS Service

Low Level Design



- Amazon EKS runs vanilla Kubernetes; Amazon EKS is upstream and a certified conformant version of Kubernetes (with backported security fixes)
- Amazon EKS supports at least 6 versions of Kubernetes, giving you time to test and roll out upgrades
- Amazon EKS provides a managed Kubernetes experience for performant, reliable, and secure Kubernetes
- Amazon EKS makes Kubernetes operations, administration, and management simple

Amazon EKS helps you build reliable, stable, and secure applications in virtually any environment


Control Plane
Data Plane
API server
kubelet
Etcd
kube proxy

EKS Components

Kubernetes Components

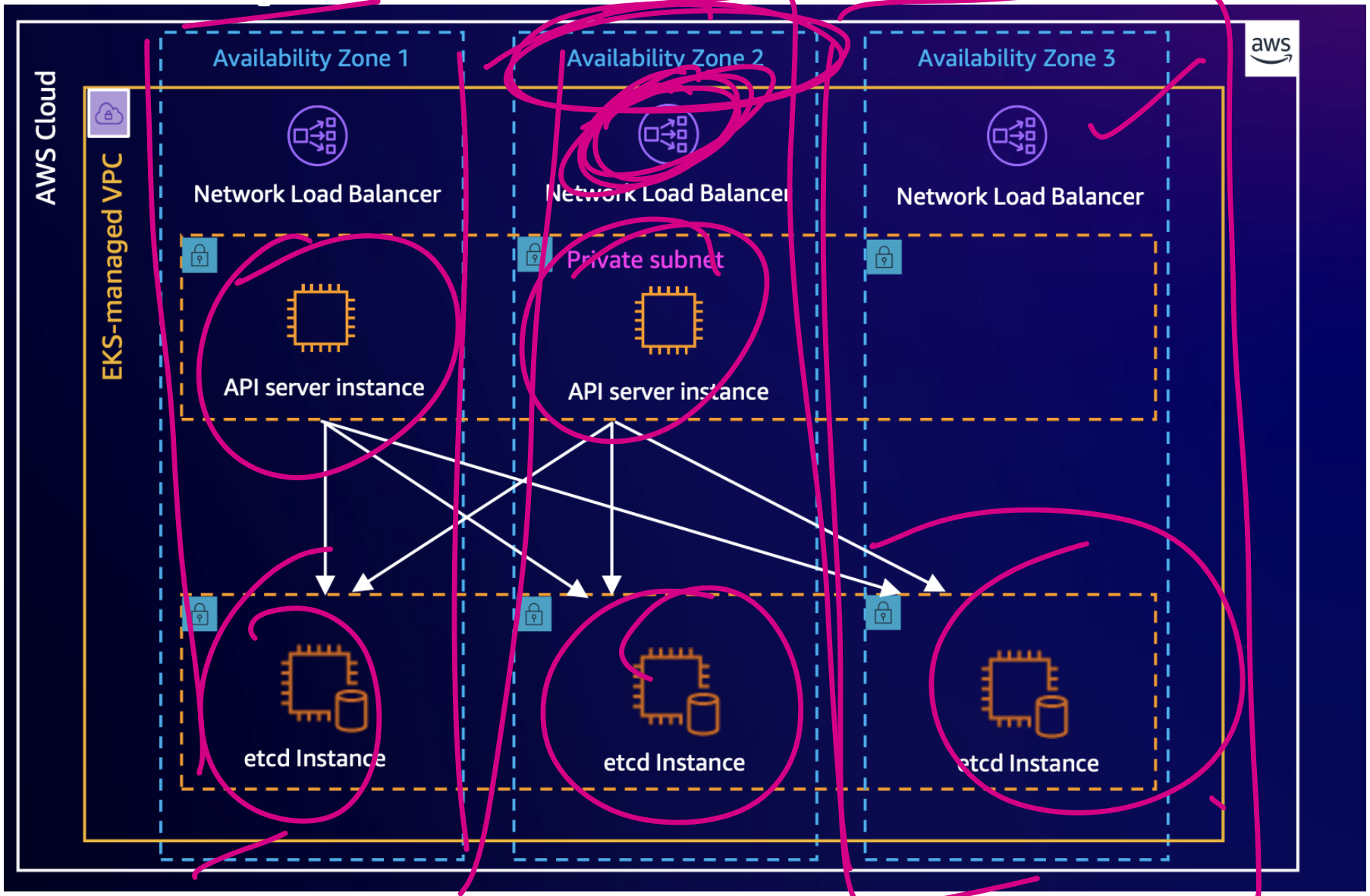
API Server
- Front end for Kubernetes Control plane
- Exposes the Kubernetes API
- Processes Restful requests
- Validates and configures data for API Objects

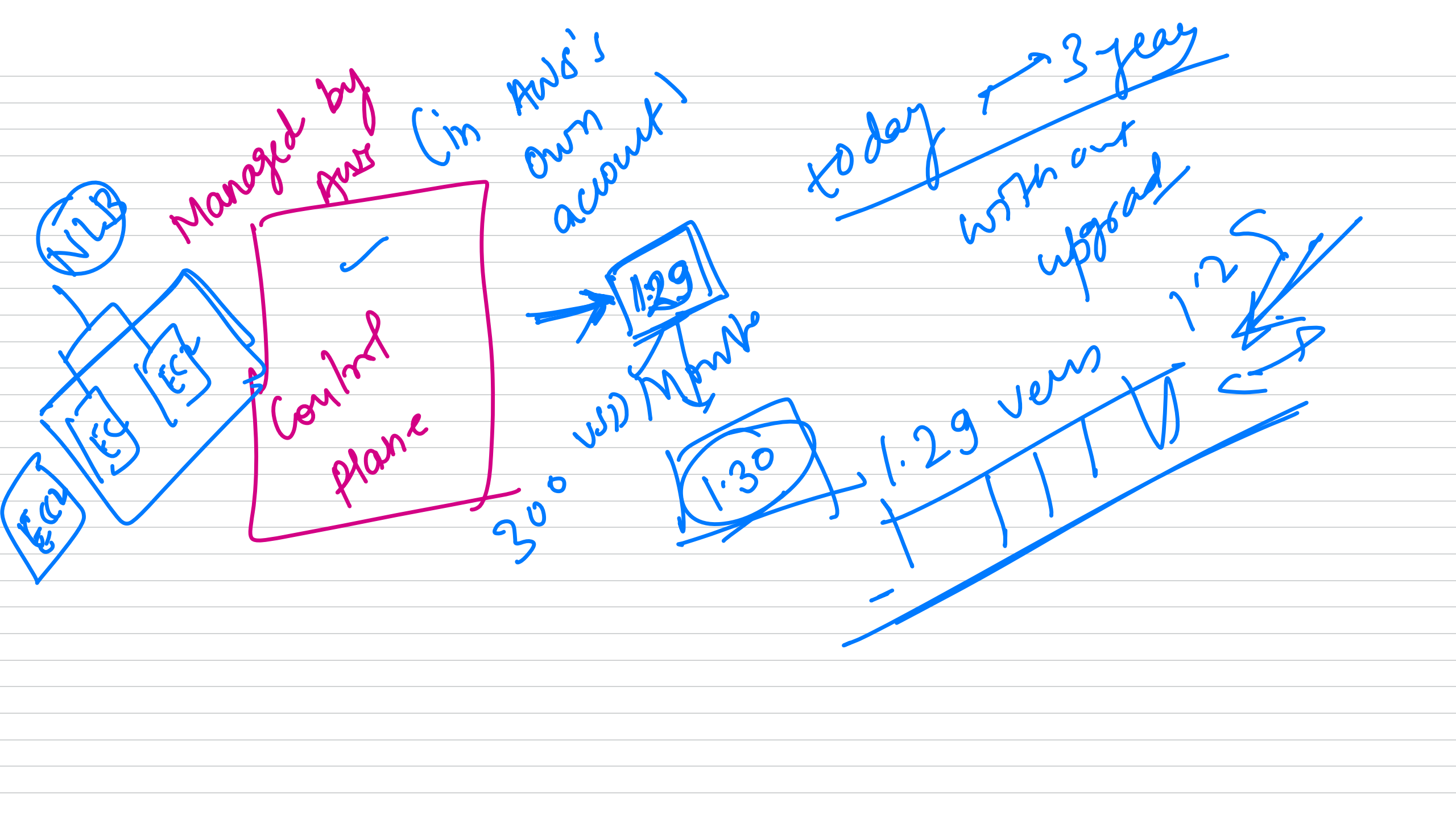
etcd
- Consistent and highly available key value store
- Stores all cluster data
- Source of truth for the cluster state
- Requires backup planning

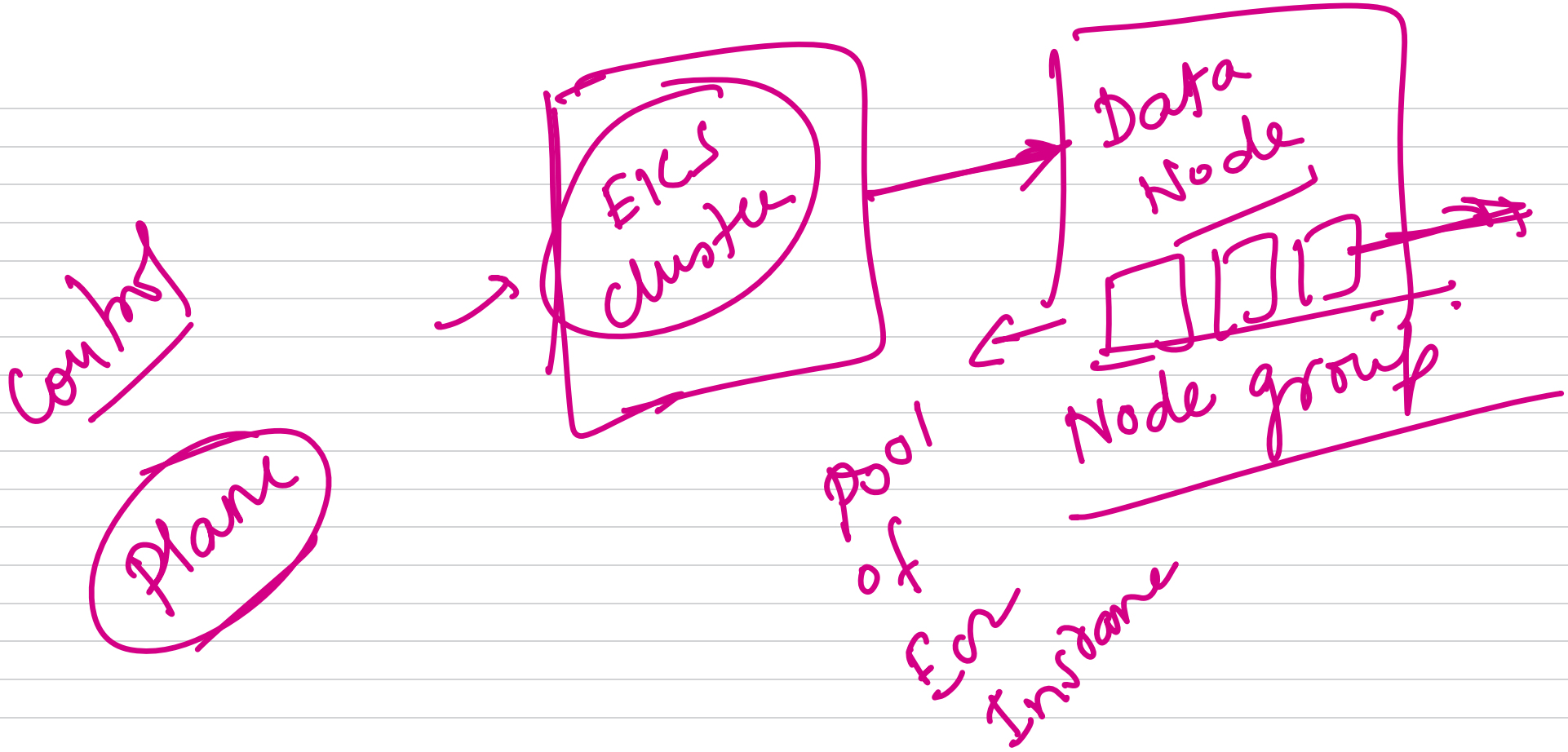
Scheduler
- Watches for newly created pods
- Assign pods to nodes based on constrains
- Considers resource requirement
- Implements scheduling policies

Controller Manager
- Runs controller process
- Monitor node health
- Replication controller - Maintains pod count
- Endpoint controller: Populates endpoint Objects
- Service account & Token controllers.

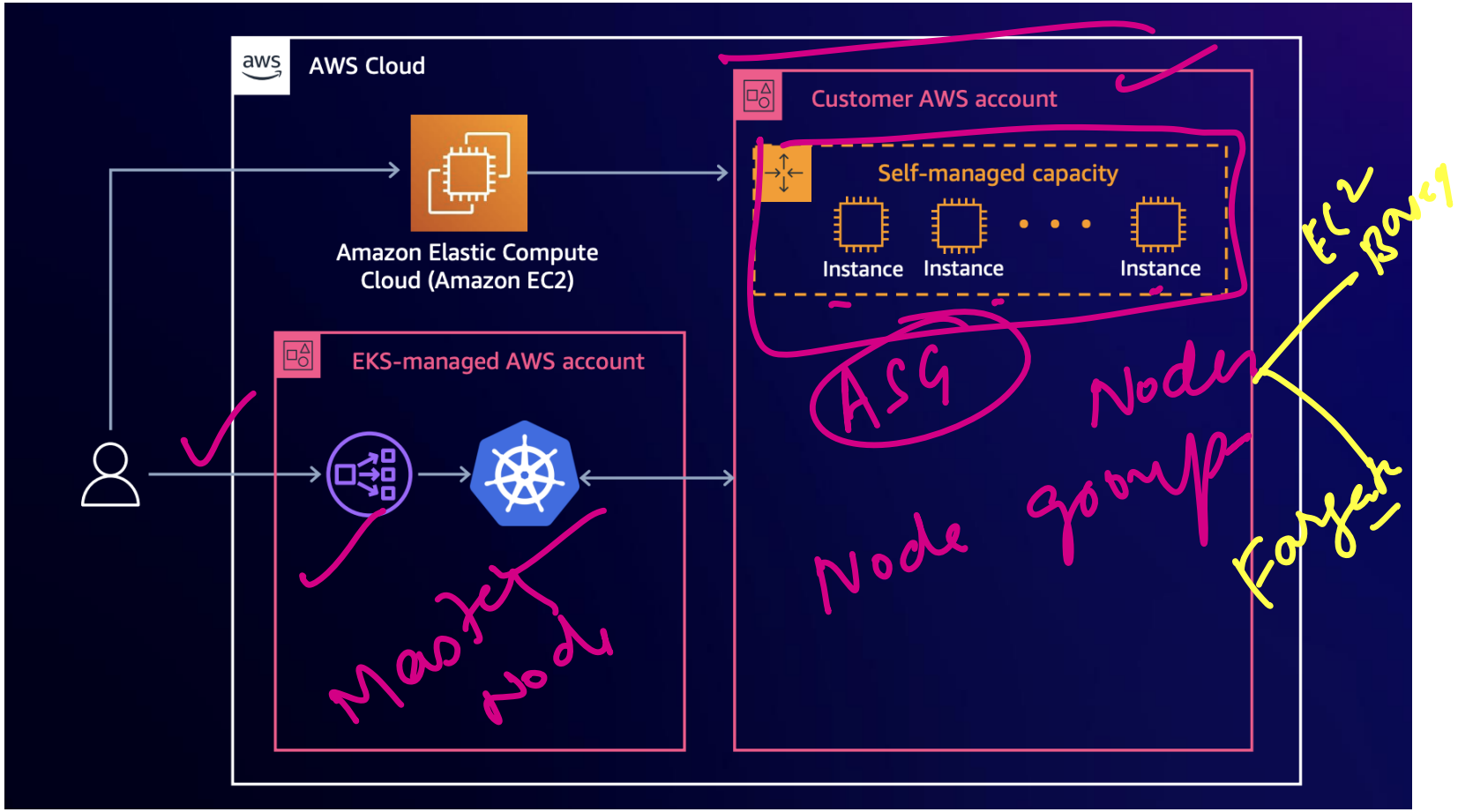
EKS Control Plane Architecture




NLB
EC2
EC2
EC2
Managed by AWS
(in AWS's own account)
Control plane
1.29
1.30
today → 3 year
without upgrade
1.29 version
1.2


Control
Plane
EKS cluster
Data Plane
Node group
Pool of EC2 Instance

EKS Data Plane Architecture



Control Plane Components

API Server

- Front-end for the Kubernetes control plane
- Exposes the Kubernetes API
- Processes RESTful requests
- Validates and configures data for API objects

etcd

- Consistent and highly-available key-value store
- Stores all cluster data
- Source of truth for the cluster state
- Requires backup planning

Scheduler

- Watches for newly created pods
- Assigns pods to nodes based on constraints
- Considers resource requirements
- Implements scheduling policies

Controller Manager

- Runs controller processes
- Node Controller: Monitors node health
- Replication Controller: Maintains pod count
- Endpoints Controller: Populates endpoint objects
- Service Account & Token Controllers: Create accounts and API tokens

Node Components

Kubelet

- Agent that runs on each node
- Ensures containers are running in a Pod
- Reports node and pod status to API server
- Executes container operations

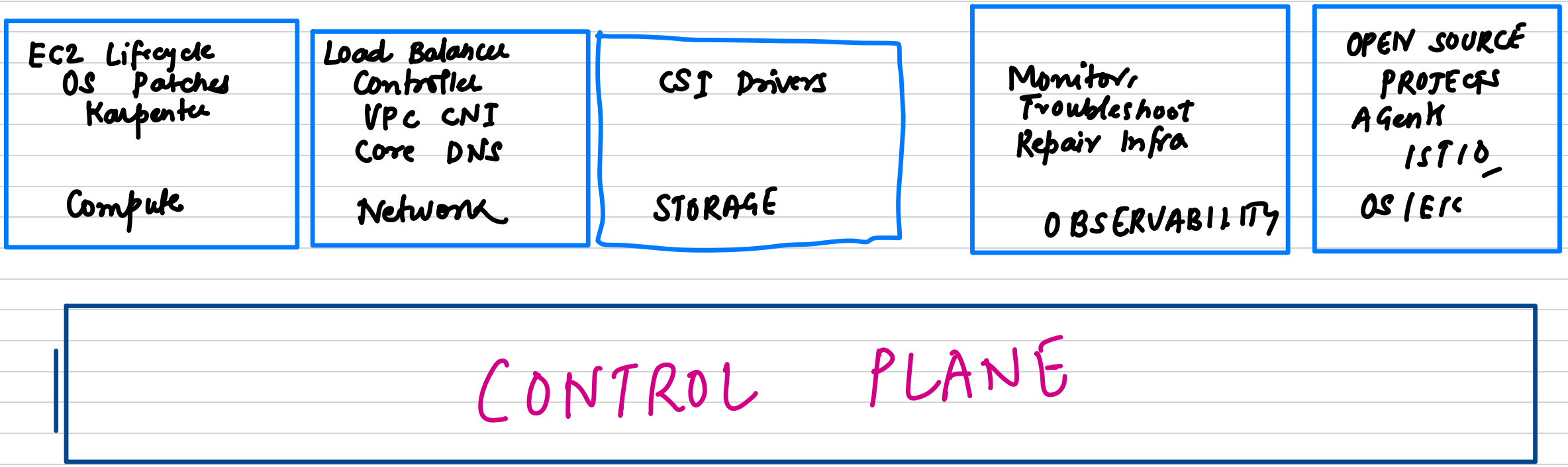
Kube-proxy

- Network proxy on each node
- Maintains network rules
- Enables pod network communication
- Implements part of Kubernetes Service concept

Container Runtime

- Software responsible for running containers
- Examples: Docker, containerd, CRI-O
- Implements Container Runtime Interface (CRI)
- Manages container lifecycle

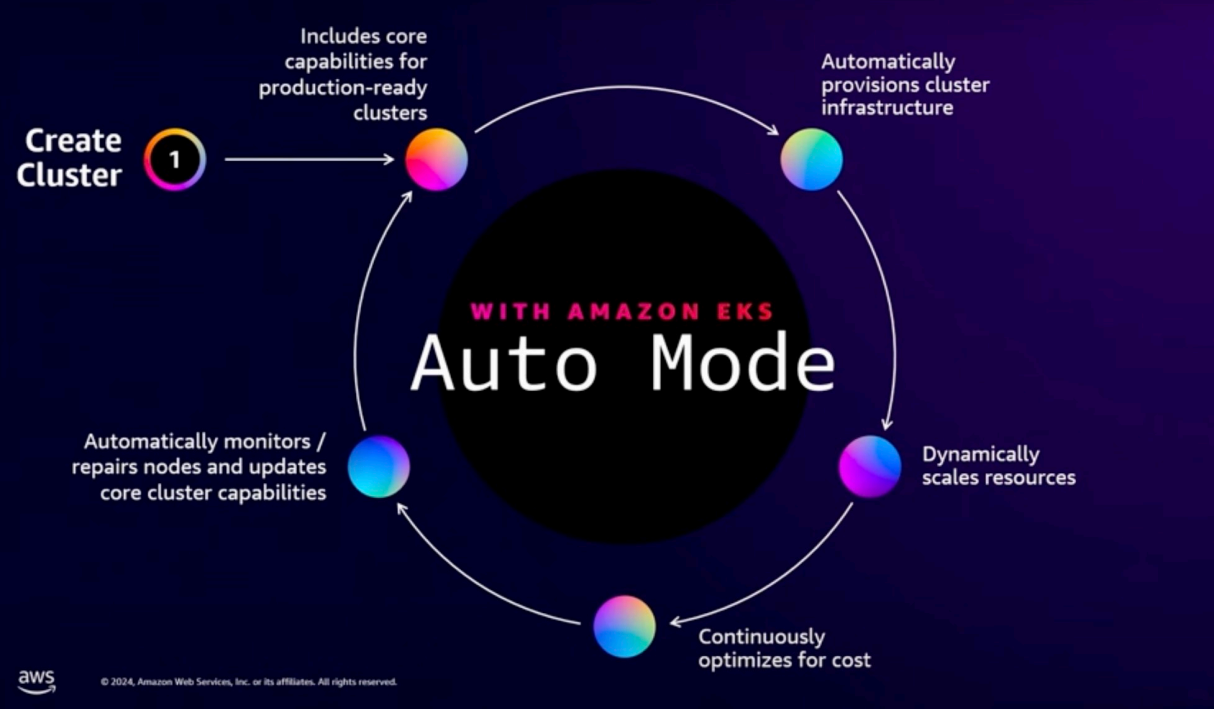
Feature	Amazon ECS	Amazon EKS
Orchestration Engine	AWS proprietary	Kubernetes (open-source)
Learning Curve	Lower	Steeper
AWS Integration	Deep, native integration	Good integration via controllers
Portability	AWS-specific	Multi-cloud compatible
Ecosystem	Limited to AWS tools	Rich ecosystem of tools and extensions
Deployment Options	EC2, Fargate	EC2, Fargate, self-managed
Pricing	No additional charge beyond resources	$0.10 per hour per cluster for control plane
Best For	AWS-focused teams, simpler workloads	Multi-cloud strategy, complex workloads

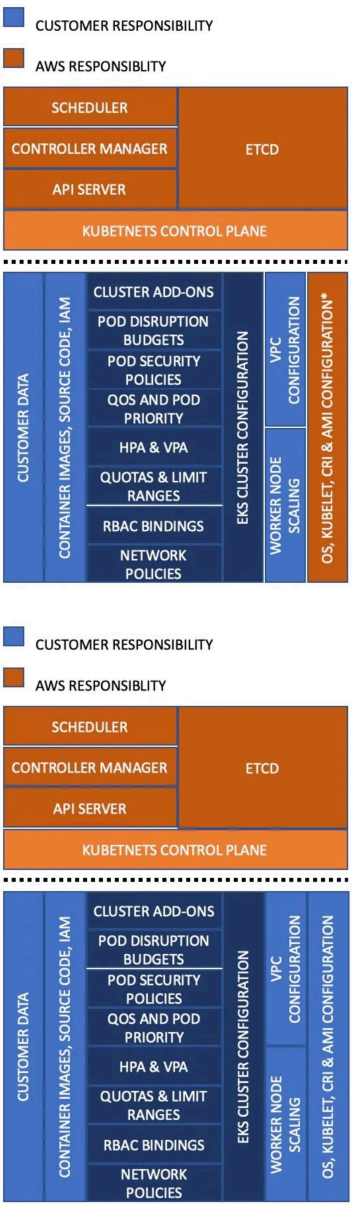

EC2 Lifecycle
OS Patches
Karpenter
Compute
Load Balancer Controller
VPC CNI
Core DNS
Network
CSI Drivers
STORAGE
Monitor
Troubleshoot
Repair Infra
OBSERVABILITY
OPEN SOURCE PROJECTS
AGENTS
ISTIO
OS / ETC
CONTROL PLANE

When to Choose EKS

- When you need Kubernetes-specific features
- For multi-cloud or hybrid cloud strategies
- When you have existing Kubernetes expertise
- For complex microservices architectures
- When you need the extensive Kubernetes ecosystem
- For advanced deployment strategies (canary, blue/green)

> Both ECS and EKS are fully managed container orchestration services. Choose based on your team's expertise, application complexity, and cloud strategy.




CUSTOMER RESPONSIBILITY
AWS RESPONSIBLITY
SCHEDULER
CONTROLLER MANAGER
API SERVER
ETCD
KUBETNETS CONTROL PLANE
Managed Node Groups
CUSTOMER DATA
CONTAINER IMAGES, SOURCE CODE, IAM
CLUSTER ADD-ONS
POD DISRUPTION BUDGETS
POD SECURITY POLICIES
QOS AND POD PRIORITY
HPA & VPA
QUOTAS & LIMIT RANGES
RBAC BINDINGS
NETWORK POLICIES
EKS CLUSTER CONFIGURATION
WORKER NODE SCALING
VPC CONFIGURATION
OS, KUBELET, CRI & AMI CONFIGURATION*
CUSTOMER RESPONSIBILITY
AWS RESPONSIBLITY
SCHEDULER
CONTROLLER MANAGER
API SERVER
ETCD
KUBETNETS CONTROL PLANE
Self Managed Workers
CUSTOMER DATA
CONTAINER IMAGES, SOURCE CODE, IAM
CLUSTER ADD-ONS
POD DISRUPTION BUDGETS
POD SECURITY POLICIES
QOS AND POD PRIORITY
HPA & VPA
QUOTAS & LIMIT RANGES
RBAC BINDINGS
NETWORK POLICIES
EKS CLUSTER CONFIGURATION
WORKER NODE SCALING
VPC CONFIGURATION
OS, KUBELET, CRI & AMI CONFIGURATION

Different Modes of Operation

Standard Mode

- EKS-managed control plane
- Self-managed or managed EC2 worker nodes
- Full control over worker node configuration
- Best for customization and specific instance types

Fargate Mode

- EKS-managed control plane
- Serverless compute for pods
- No node management required
- Pay-per-pod execution model
- Best for simplicity and variable workloads

Hybrid Mode

- Combination of EC2 and Fargate
- Use EC2 for predictable workloads
- Use Fargate for variable workloads
- Best for mixed requirements

Ways to Deploy an EKS Cluster

AWS Management Console

- Visual interface for cluster creation
- Step-by-step wizard
- Good for learning and exploration
- Less suitable for automation

eksctl (Recommended)

- Official CLI tool for EKS
- Simple one-line cluster creation
- YAML-based configuration
- Handles IAM roles and VPC setup
- Good for both simple and advanced use cases

Infrastructure as Code

- AWS CloudFormation
- Terraform
- AWS CDK
- Best for production environments
- Enables version control and repeatability

Example: Creating a Cluster with eksctl

```
# Basic cluster creation eksctl create cluster --name my-cluster --region us-west-2 # Advanced configuration eksctl create cluster \ --name my-cluster \ --region us-west-2 \ --nodegroup-name standard-workers \ --node-type t3.medium \ --nodes 3 \ --nodes-min 1 \ --nodes-max 5 \ --with-oidc \ --ssh-access \ --ssh-public-key my-key
```

- **Workloads**
  - PodTemplates
  - Pods
  - ReplicaSets
  - Deployments
  - StatefulSets
  - DaemonSets
  - Jobs
  - CronJobs
  - PriorityClasses
  - HorizontalPodAutoscalers

- **Cluster**
  - Nodes
  - Namespaces
  - APIServices
  - Leases
  - RuntimeClasses
  - FlowSchemas
  - PriorityLevelConfigurations

PriorityLevelConfigurations

▼ **Service and networking**

Services

Endpoints

EndpointSlices

Ingresses

IngressClasses

- **Config and secrets**
  - ConfigMaps
  - Secrets
- **Storage**
  - PersistentVolumeClaims
  - PersistentVolumes
  - StorageClasses
  - VolumeAttachment
  - CSIDrivers
  - CSINodes
  - CSIStorageCapacities
- **Authentication**
  - ServiceAccounts
- **Authorization**
  - ClusterRoles
  - ClusterRoleBindings
  - Roles
  - RoleBindings

- **Policy**
  - LimitRanges
  - ResourceQuotas
  - NetworkPolicies
  - PodDisruptionBudgets
- **Extensions**
  - CustomResourceDefinitions
  - MutatingWebhookConfigurations
  - ValidatingWebhookConfigurations