
AWS
re:Invent

CON303

Amazon EKS on AWS Fargate Deep Dive

Akshay Ram

Sr. Product Manager – Technical, AWS Fargate

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

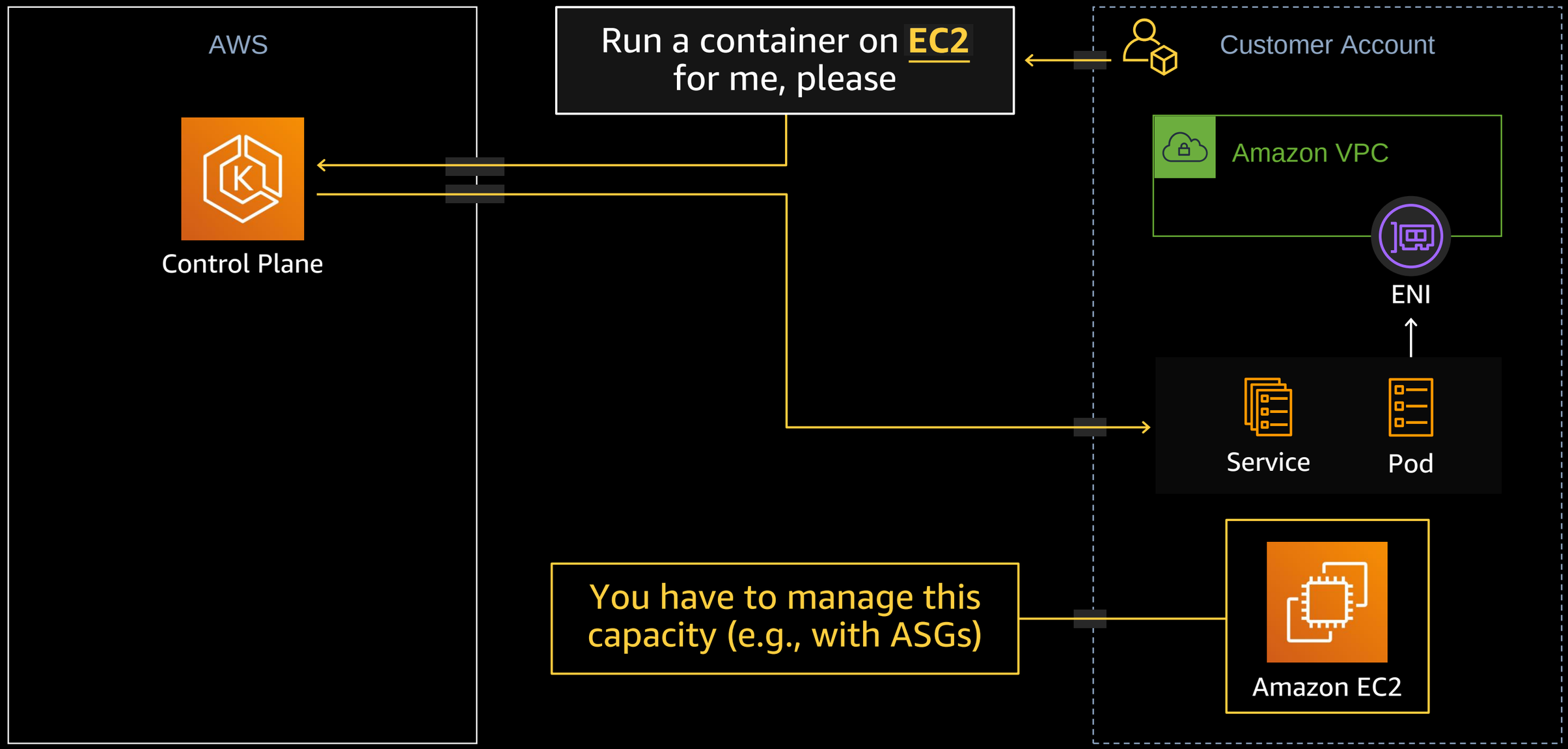
Agenda

- EKS with Fargate – Kubernetes with serverless compute
- What we shipped – Working backwards
- Roadmap – Looking Ahead

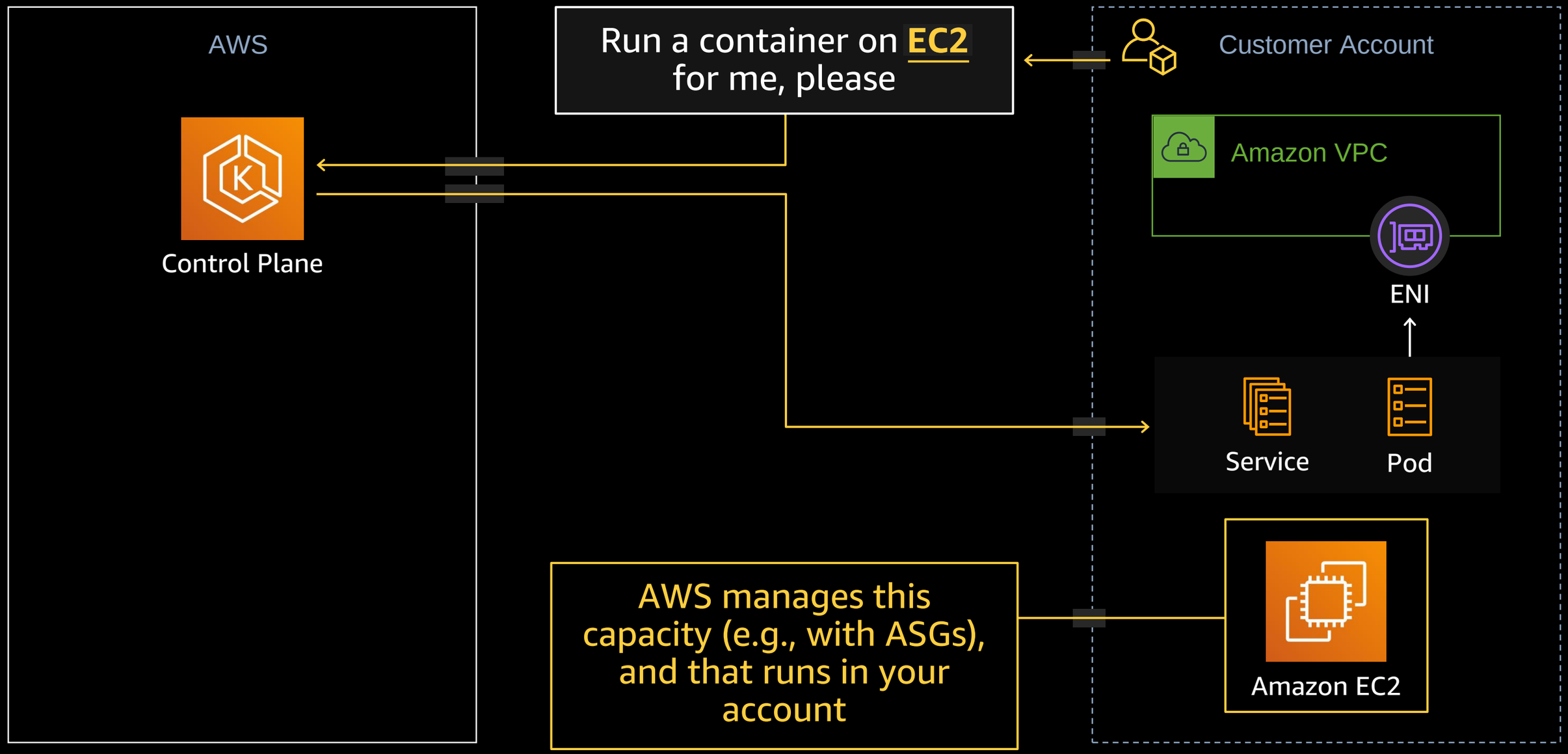


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

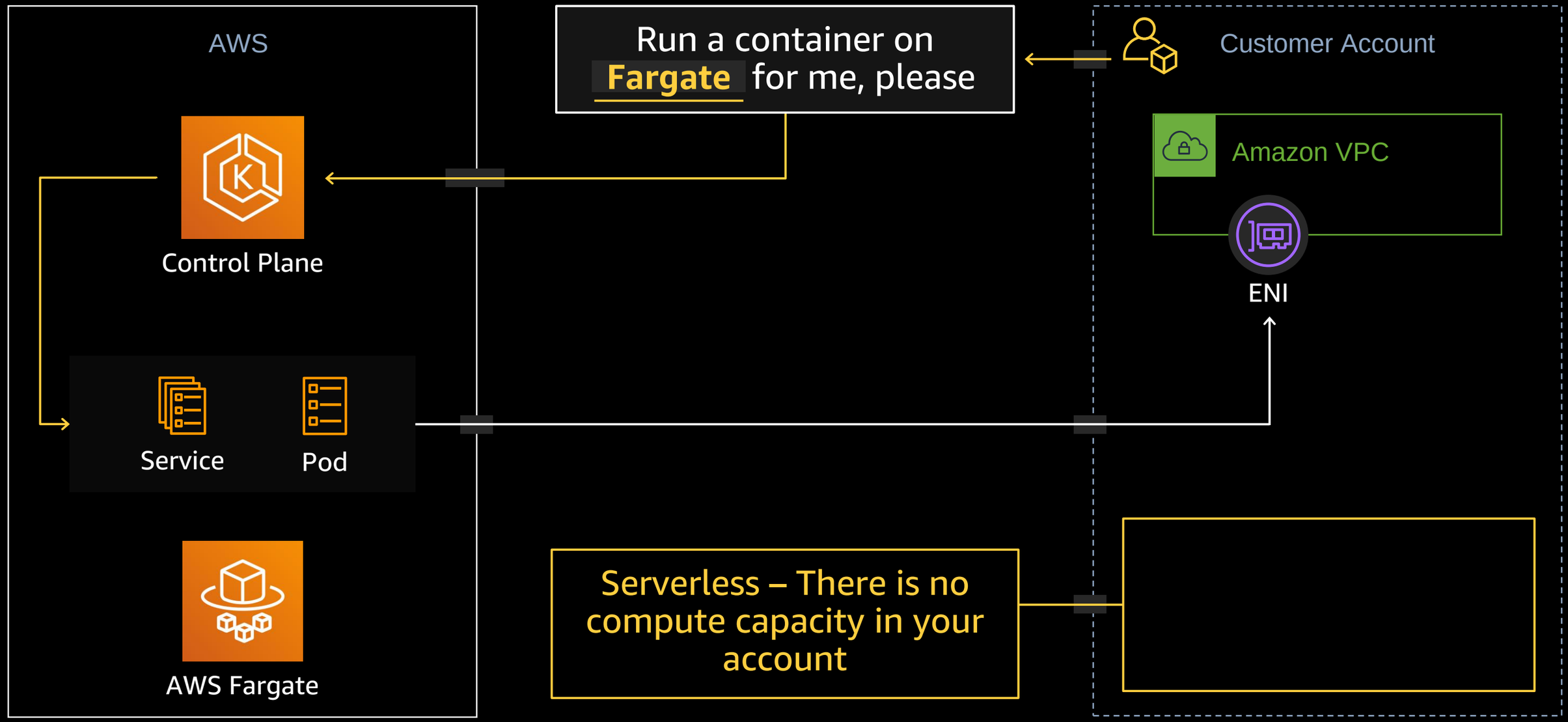
EKS with EC2


AWS
Run a container on EC2 for me, please
Customer Account
Amazon VPC
Control Plane
ENI
Service
Pod
You have to manage this capacity (e.g., with ASGs)
Amazon EC2

EKS with managed node groups



EKS with Fargate



Using Fargate with EKS

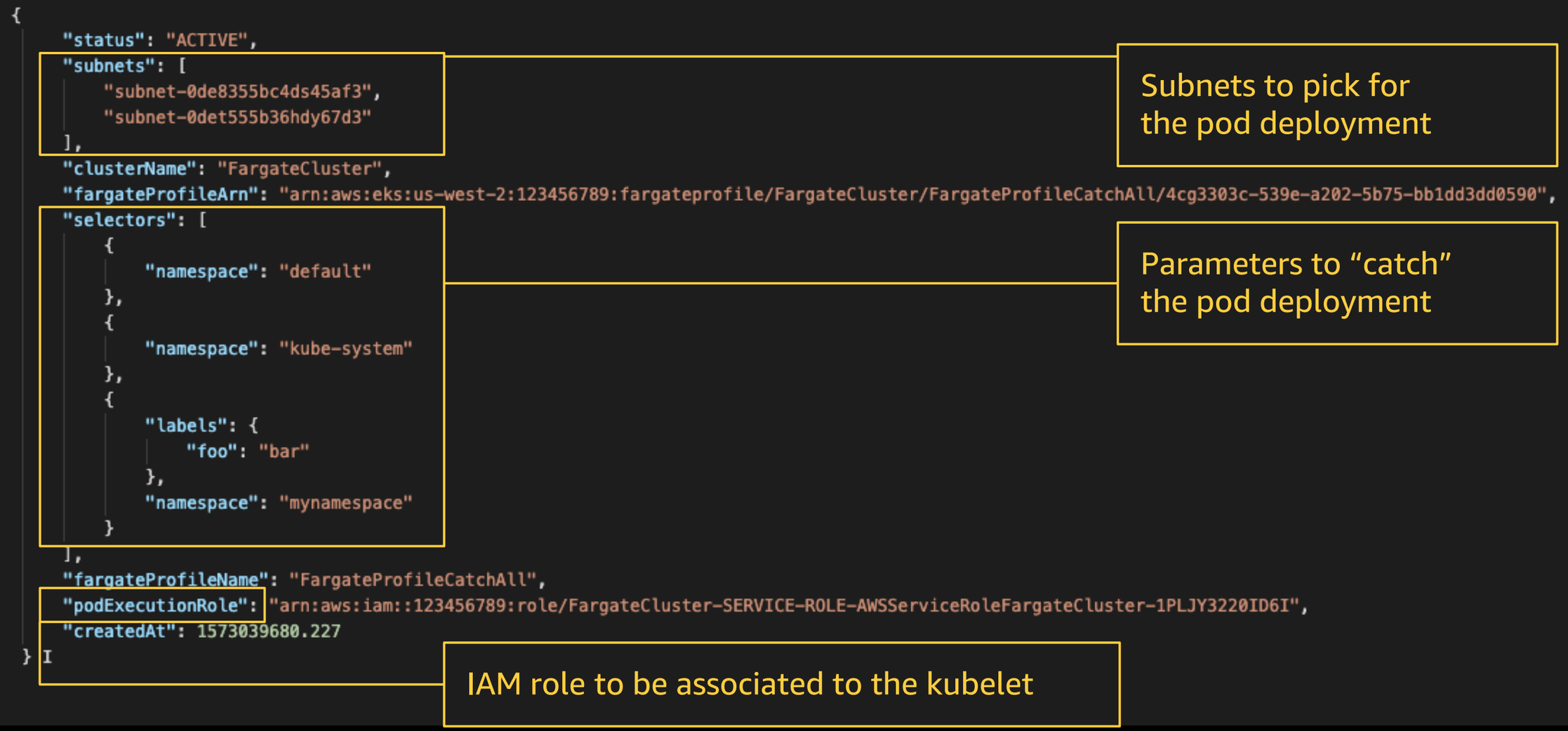
Fargate profile is your interface to configure Fargate on an EKS cluster

- EKS API that lets you define what pods should run on Fargate
- Other configurations such as subnets and IAM roles

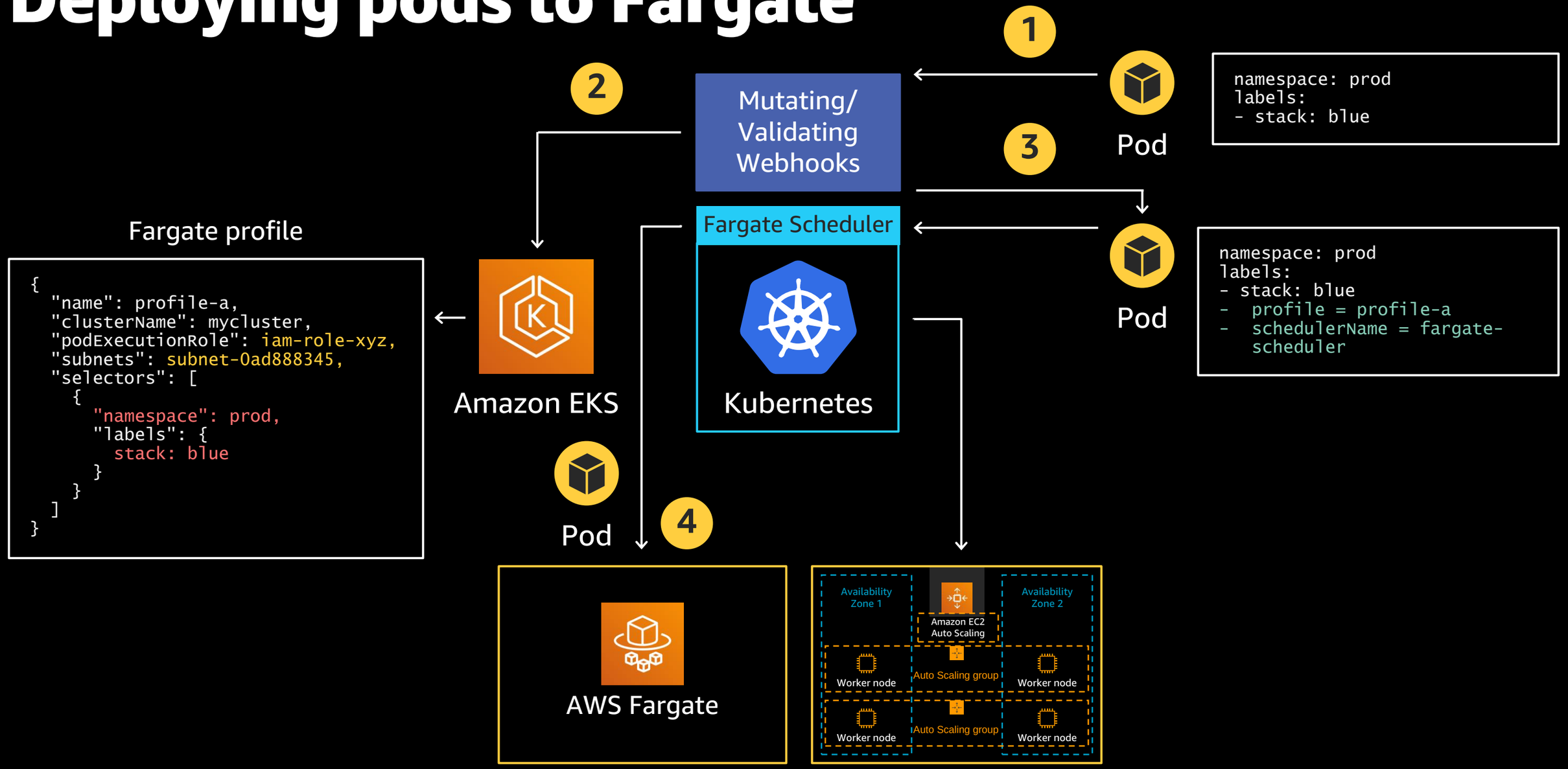
Fargate scheduler

- Under the hood implementation that runs alongside the standard Kubernetes scheduler

Fargate profile templates



Deploying pods to Fargate



Value proposition

- Applications not infrastructure
  - Use the pod spec and don’t worry about nodes
- No cluster scaling
  - We scale the clusters, you scale applications
- No host and container runtime patching
  - We patch the hosts, you focus on containers
- Pod isolation
  - Secure by default. Containers are isolated in their own VM
- Streamlined updates
  - Update cluster and simply relaunch pods

Key launches

- Amazon EFS support
- Built-in logging
- Network load balancing

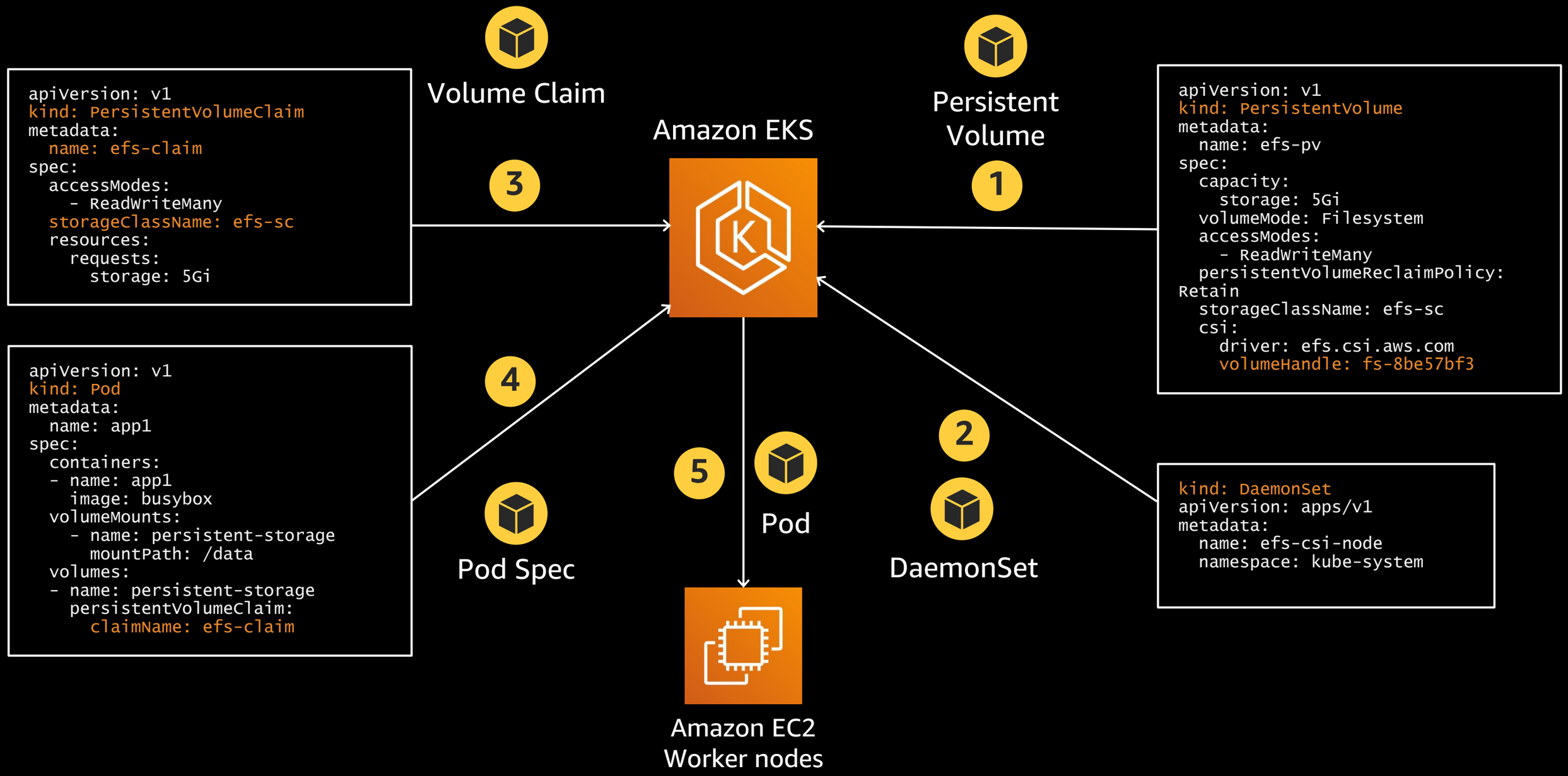
DaemonSets don’t work on Fargate

- Both Amazon EFS and logging usually work through a DaemonSet
- On Fargate, every pod runs in its own VM isolation boundary
- DaemonSets need at least two pods to run in the same VM

Design considerations

- **Consistency where it matters** by using native Kubernetes objects where applicable to give customers a consistent interface across compute types (EC2 and Fargate)
- **Simplify where possible** by managing more infrastructure for customers

EFS on EKS with EC2

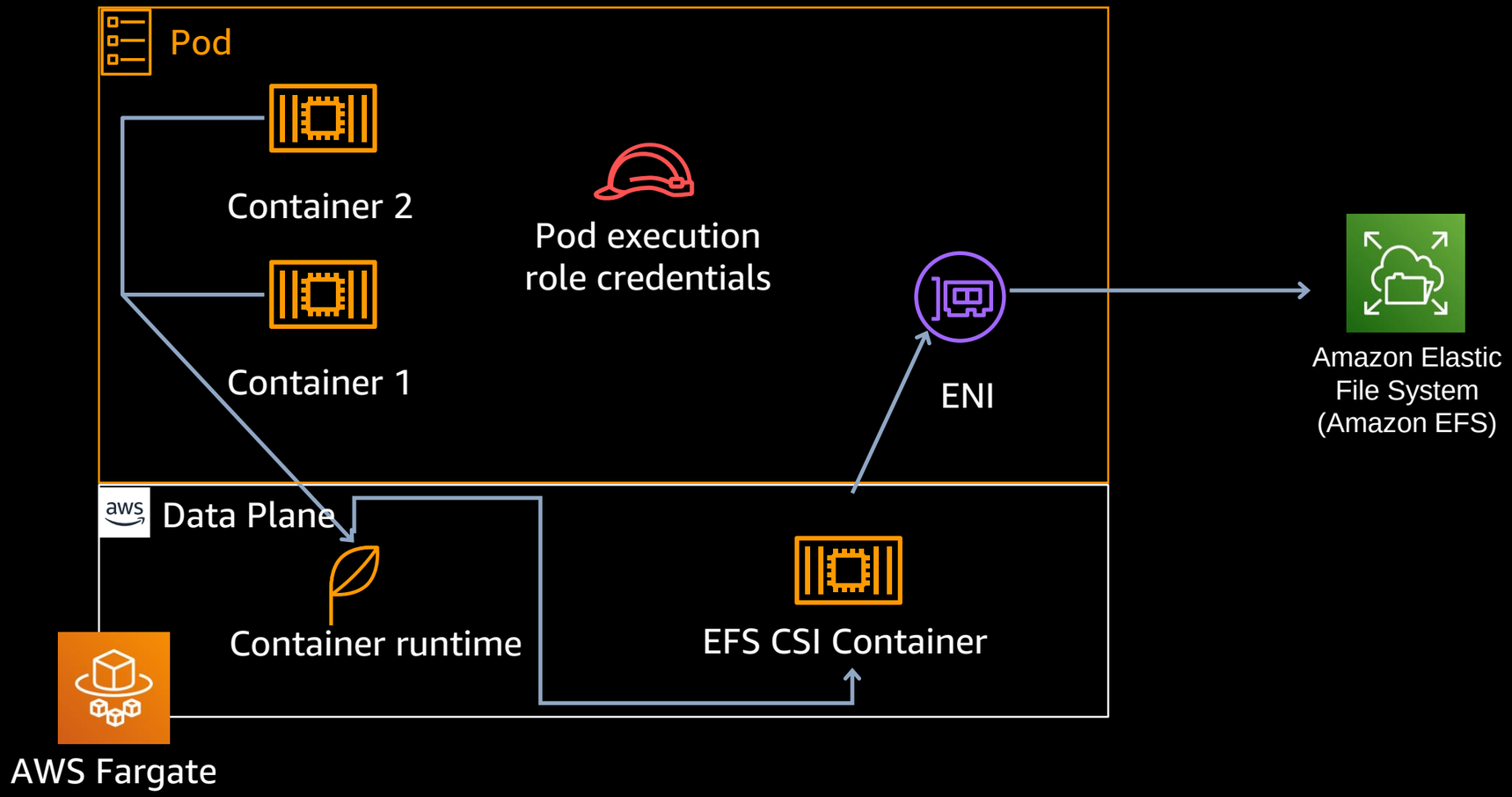


Shared responsibility

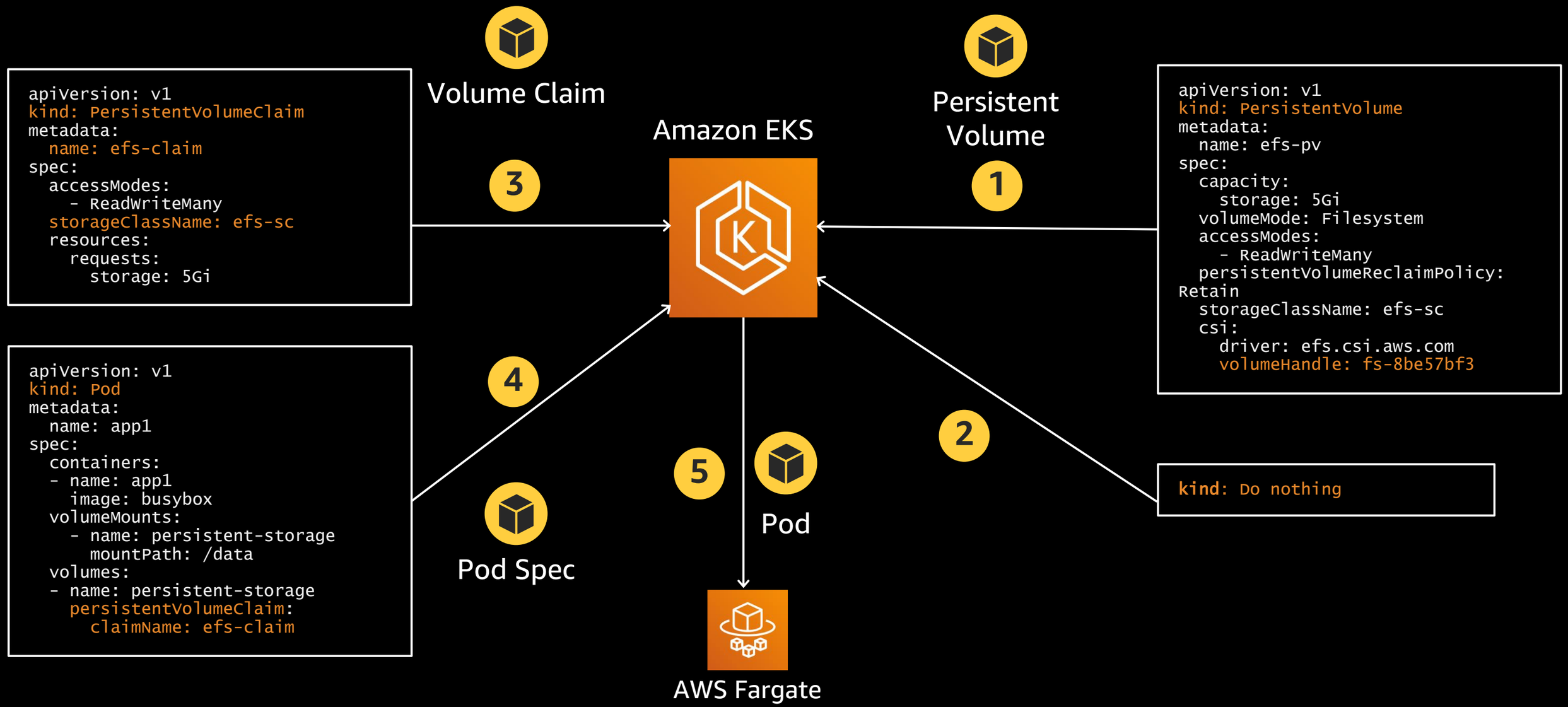
Customer ask	Kubernetes Administrator	Kubernetes User	Fargate
Ability to configure EFS	Persistent volumes (PV), StorageClass	PodSpec and PersistentVolumeClaims	Fully managed CSI driver

Deep dive: EFS support on EKS/Fargate

Managed CSI



EFS on EKS with Fargate



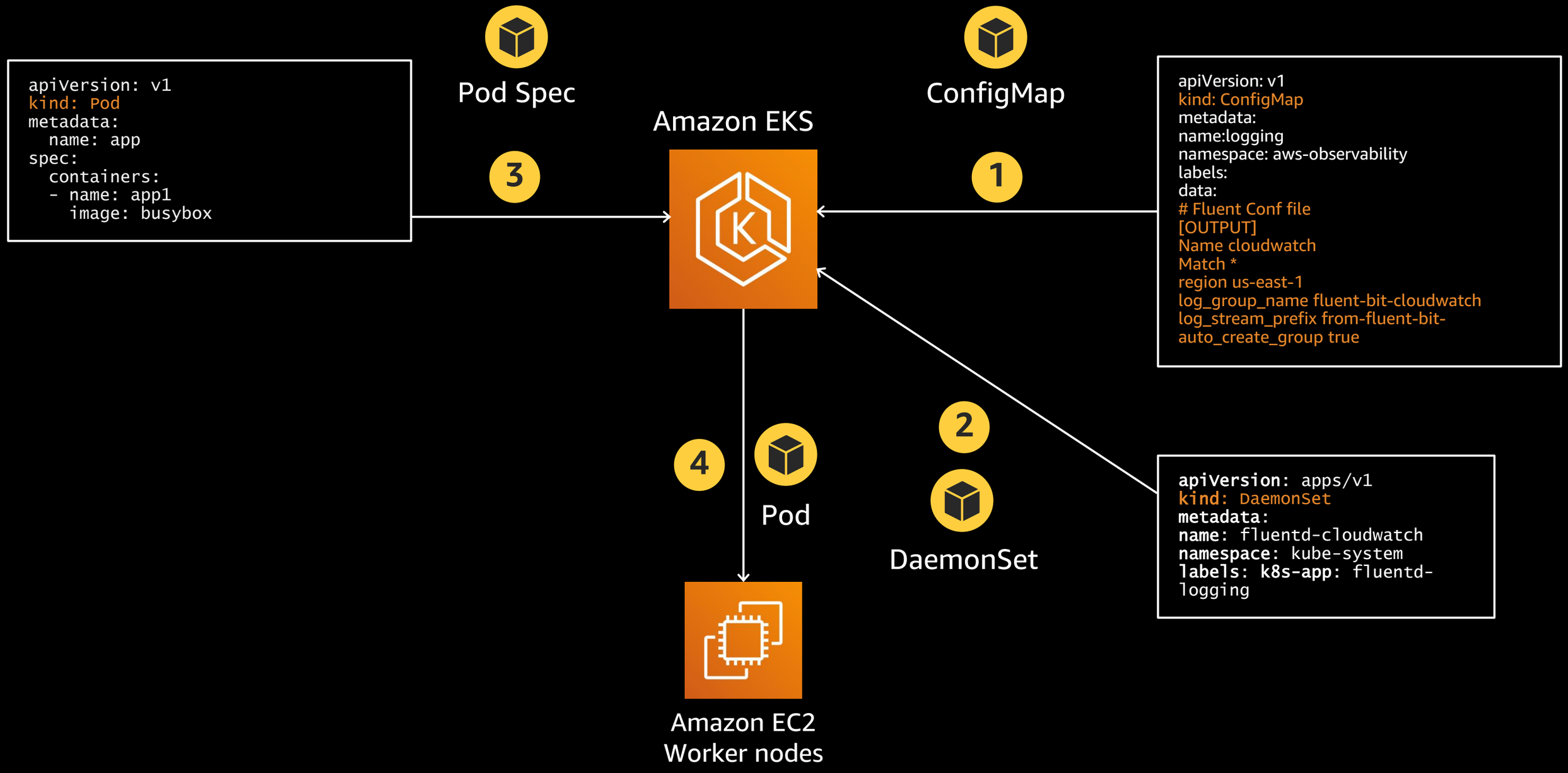
Experience across compute types on EKS

Design choices	EC2	Fargate
Simplify	Kubernetes administrator deploys CSI via a DaemonSet	CSI built in
Consistency	Kubernetes administrator creates PV and StorageClass	Kubernetes administrator creates PV and StorageClass
Consistency	Kubernetes user creates a pod spec with a claim	Kubernetes user creates a pod spec with a claim
Simplify	Kubernetes administrator manages the CSI driver lifecycle	Simplified day 2 ops

Takeaways

- CSI built in
- Customers get compatibility with Kubernetes objects such as PersistentVolumes, StorageClass, PersistentVolumeClaims
- Run applications such as WordPress, Drupal, and data analytics, and leave the infrastructure management to AWS

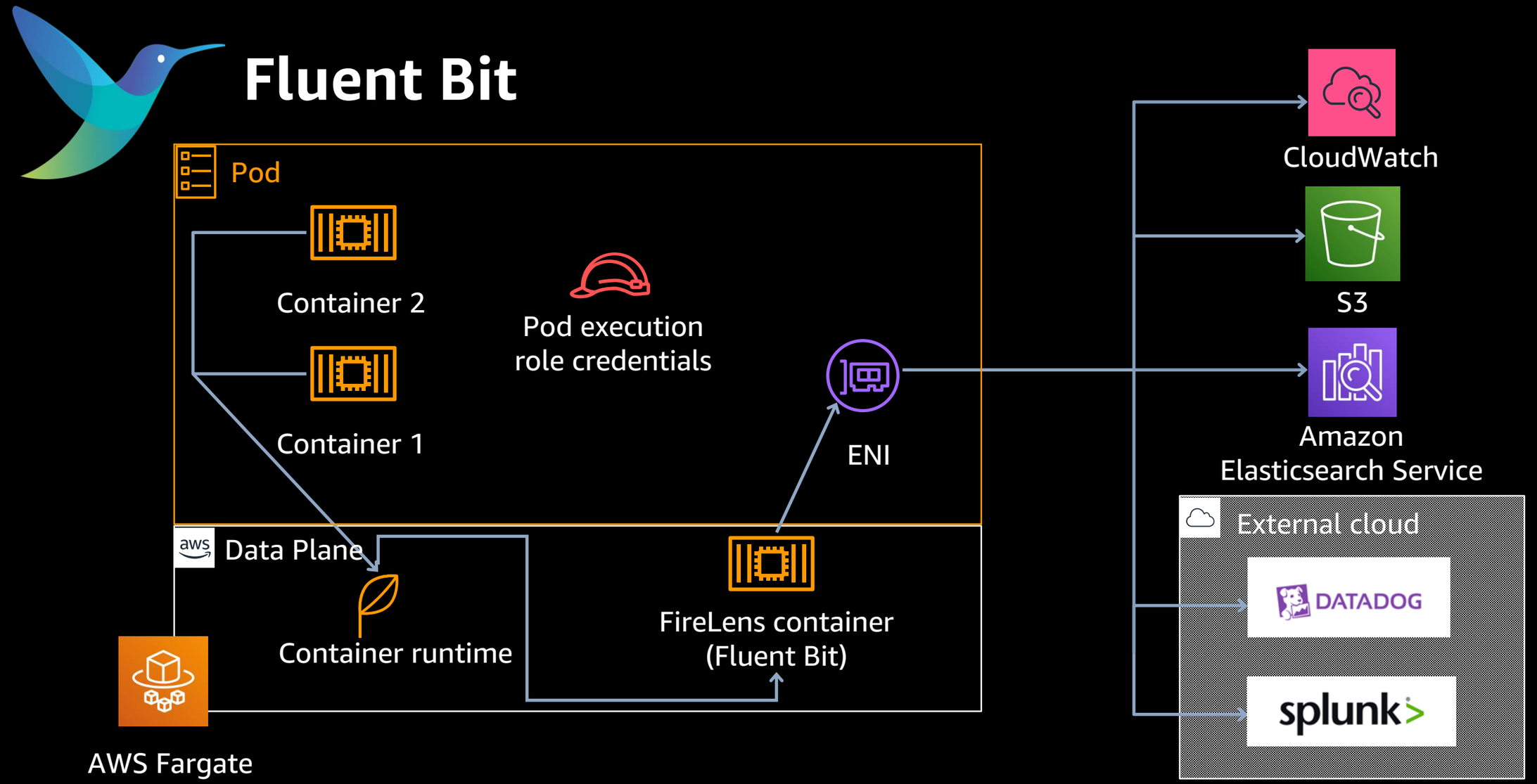
Logging on EKS with EC2



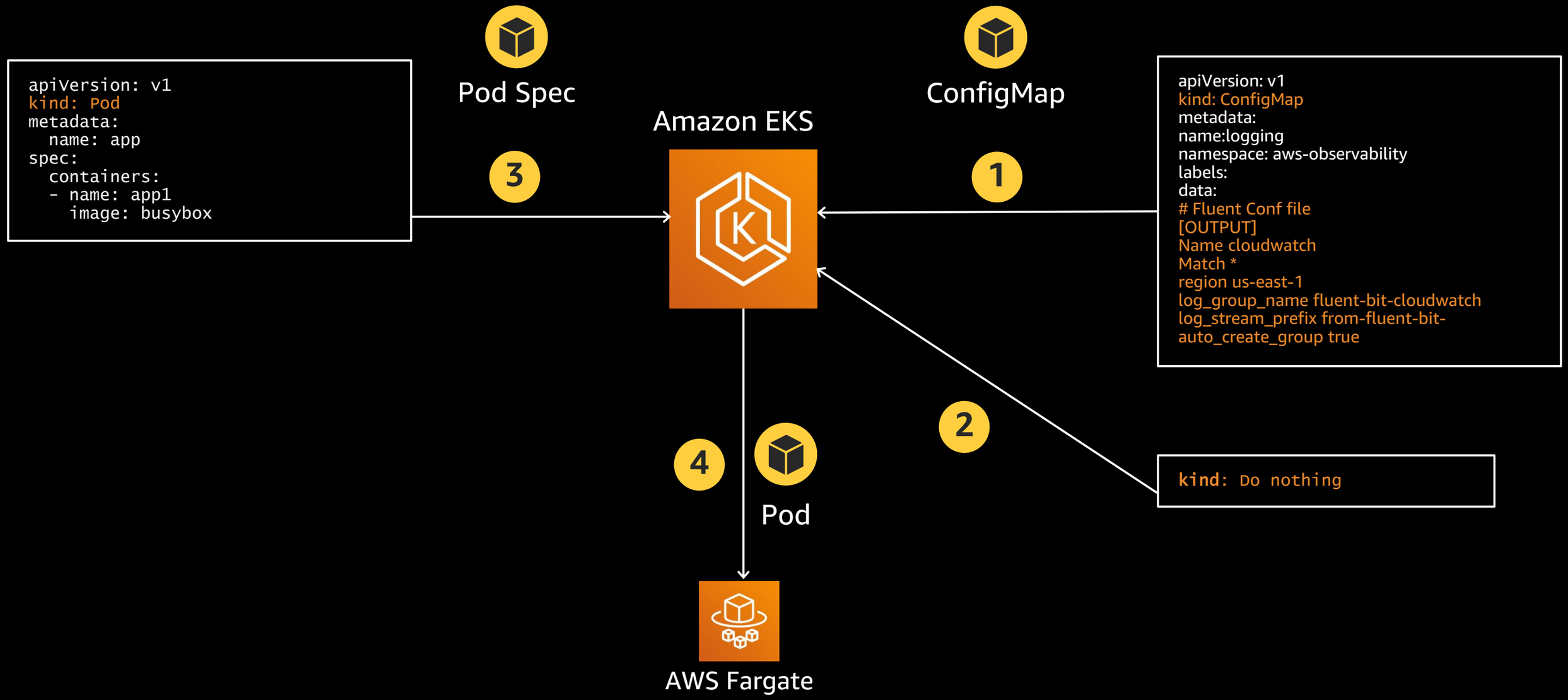
Shared responsibility

Customer ask	Kubernetes Administrator	Kubernetes User	Fargate
Ability to configure logging	ConfigMap, Fluent conf	PodSpec with no change in application container	Fully managed Fluent Bit

Deep dive: Logging on EKS/Fargate (New!)



Logging on EKS with Fargate



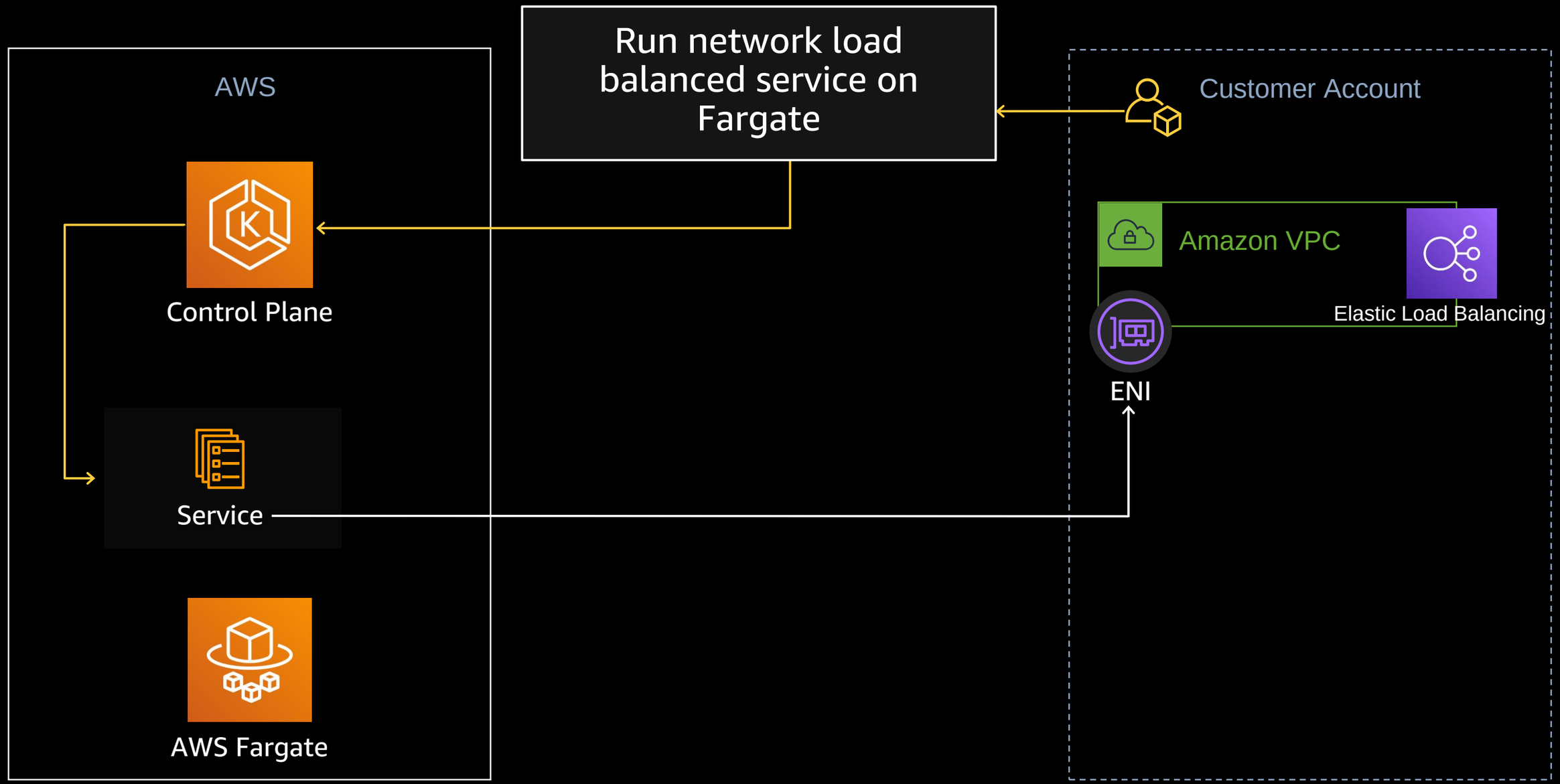
Experience across compute types on EKS

Design choices	EC2	Fargate
Simplify	Customer deploys Fluent Bit via a DaemonSet	Fluent Bit built-in
Consistency	Kubernetes administrator configures Fluent Conf via a ConfigMap	Kubernetes administrator configures Fluent Conf via a ConfigMap
Consistency	Kubernetes users just run pods	Kubernetes users just run pods
Simplify	Kubernetes administrator manages the Fluent Bit lifecycle management	Simplified day 2 ops

Takeaways

- Fluent Bit built in
- Customers get compatibility with Kubernetes objects through ConfigMap
- Built in support for CloudWatch, Amazon Kinesis Data Firehose, and Amazon Elasticsearch Service among others

NLB through AWS Load Balancer Controller



NLB through AWS Load Balancer Controller

- Just annotate services
- NLB provisioned on your behalf

```
kind: Service
apiVersion: v1
metadata:
  name: nlb-ip-svc
  annotations:
    # route traffic directly to pod
IPs
    service.beta.kubernetes.io/aws-
load-balancer-type: "nlb-ip"
```

Experience across compute types on EKS

Design choices	EC2	Fargate
Consistency	Kubernetes administrator applies Load Balancer Controller to the cluster	Kubernetes administrator applies Load Balancer Controller to the cluster
Consistency	Annotate service with service.beta.kubernetes.io/ aws-load-balancer-type: "nlb-ip”	Annotate service with service.beta.kubernetes.io/ aws-load-balancer-type: "nlb-ip”

Other notable launches

- EKS/Fargate usage covered by compute savings plan
- 18 AWS regions
- Ephemeral storage encryption by default

Current limitations of Fargate

- No privileged pods
- No support for general purpose DaemonSets
- No GPUs
- No Amazon EBS support

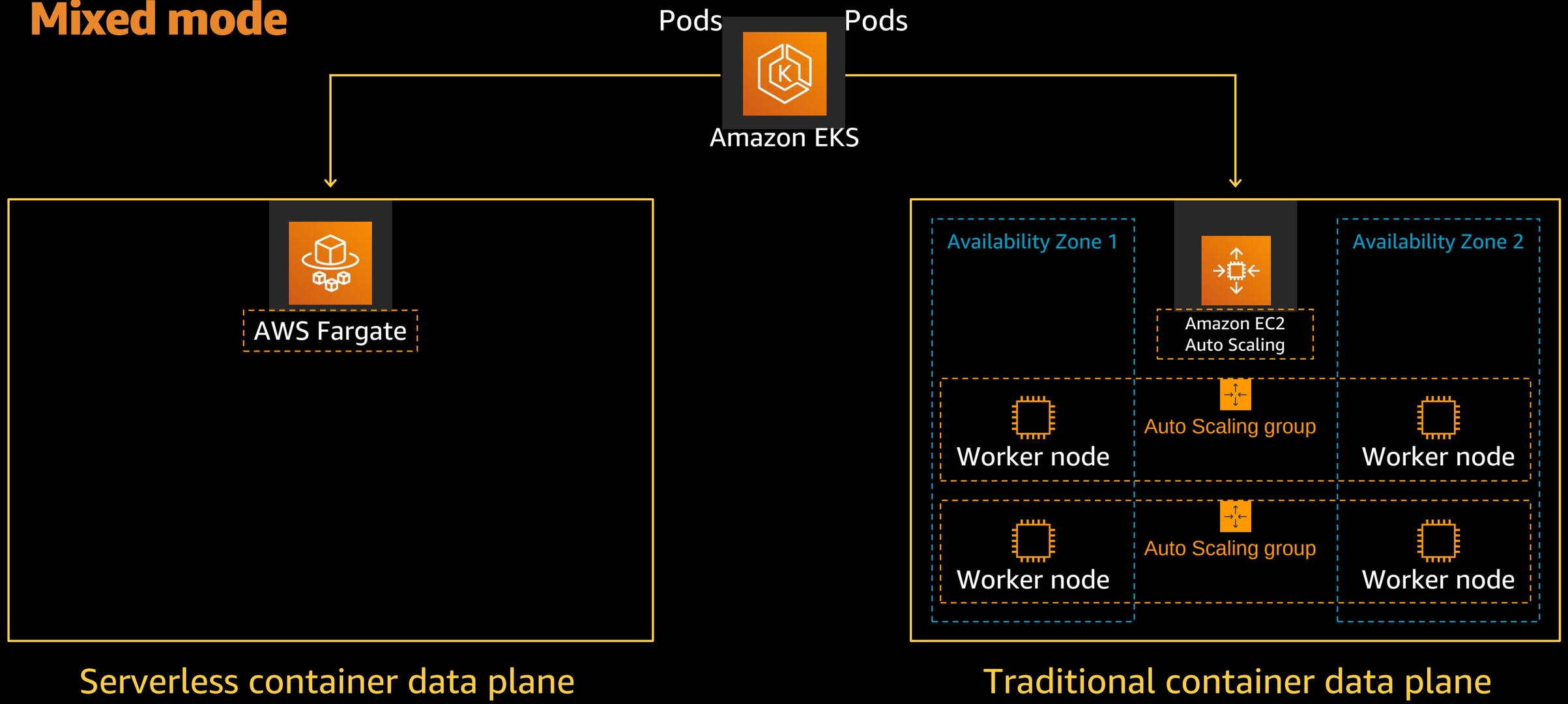
Match only supported applications


```
{
    "status": "ACTIVE",
    "subnets": [
        "subnet-0de8355bc4ds45af3",
        "subnet-0det555b36hdy67d3"
    ],
    "clusterName": "FargateCluster",
    "fargateProfileArn": "arn:aws:eks:us-west-2:123456789:fargateprofile/FargateCluster/FargateProfileCatchAll/4cg3303c-539e-a202-5b75-bb1dd3dd0590",
    "selectors": [
        {
            "namespace": "default"
        },
        {
            "namespace": "kube-system"
        },
        {
            "labels": {
                "foo": "bar"
            },
            "namespace": "mynamespace"
        }
    ],
    "fargateProfileName": "FargateProfileCatchAll",
    "podExecutionRole": "arn:aws:iam::123456789:role/FargateCluster-SERVICE-ROLE-AWSServiceRoleFargateCluster-1PLJY3220ID6I",
    "createdAt": 1573039680.227
} I
```


Parameters to “catch” the applications that are supported

EKS data plane options

Mixed mode



Serverless container data plane

Traditional container data plane

Roadmap

- Built in metrics and traces support through AWS Distro for OpenTelemetry
- Ability to expand ephemeral volumes
- Security groups per pod
- Partner support (security and observability)

Resources and call to action: Fast track

```
sh-4.2# eksctl create cluster --name eksfargate --fargate
[ℹ]  eksctl version 0.25.0-rc.0
[ℹ]  using region us-west-2
[ℹ]  setting availability zones to [us-west-2a us-west-2c us-west-2d]
[ℹ]  subnets for us-west-2a - public:192.168.0.0/19 private:192.168.96.0/19
[ℹ]  subnets for us-west-2c - public:192.168.32.0/19 private:192.168.128.0/19
[ℹ]  subnets for us-west-2d - public:192.168.64.0/19 private:192.168.160.0/19
[ℹ]  using Kubernetes version 1.17
[ℹ]  creating EKS cluster "eksfargate" in "us-west-2" region with Fargate profile
[ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-west-2 --cluster=eksfargate'
[ℹ]  CloudWatch logging will not be enabled for cluster "eksfargate" in "us-west-2"
[ℹ]  you can enable it with 'eksctl utils update-cluster-logging --region=us-west-2 --cluster=eksfargate'
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "eksfargate" in "us-west-2"
[ℹ]  2 sequential tasks: { create cluster control plane "eksfargate", create fargate profiles }
[ℹ]  building cluster stack "eksctl-eksfargate-cluster"
[ℹ]  deploying stack "eksctl-eksfargate-cluster"
[ℹ]  creating Fargate profile "fp-default" on EKS cluster "eksfargate"
[ℹ]  created Fargate profile "fp-default" on EKS cluster "eksfargate"
[ℹ]  "coredns" is now schedulable onto Fargate
[ℹ]  "coredns" is now scheduled onto Fargate
[ℹ]  "coredns" pods are now scheduled onto Fargate
[ℹ]  waiting for the control plane availability...
[✔]  saved kubeconfig as "/root/.kube/config"
[ℹ]  no tasks
[✔]  all EKS cluster resources for "eksfargate" have been created
[ℹ]  kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
[✔]  EKS cluster "eksfargate" in "us-west-2" region is ready
```

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.