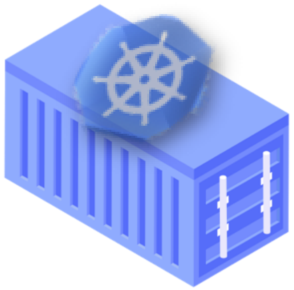
Mastering Amazon EKS Hands-On

Copyright © Thinknyx Technologies LLP

Mastering Amazon EKS Hands-On

EKS is a fully managed Kubernetes offering from AWS that simplifies container orchestration at scale



Cluster Creation

Node Groups

Load Balancing

Storage

Scaling

Copyright © Thinknyx Technologies LLP

The Ultimate Linux Bootcamp for DevOps SRE & Cloud Engineers

Practical Kubernetes – Beyond CKA and CKAD

Automation with Ansible - Hands-on DevOps

Kubernetes and Cloud Native Associate

Argo CD for the Absolute Beginners - Hands-On

Mastering Docker Essential - Hands-on

Kulbhushan Mayer

Yogesh Raheja

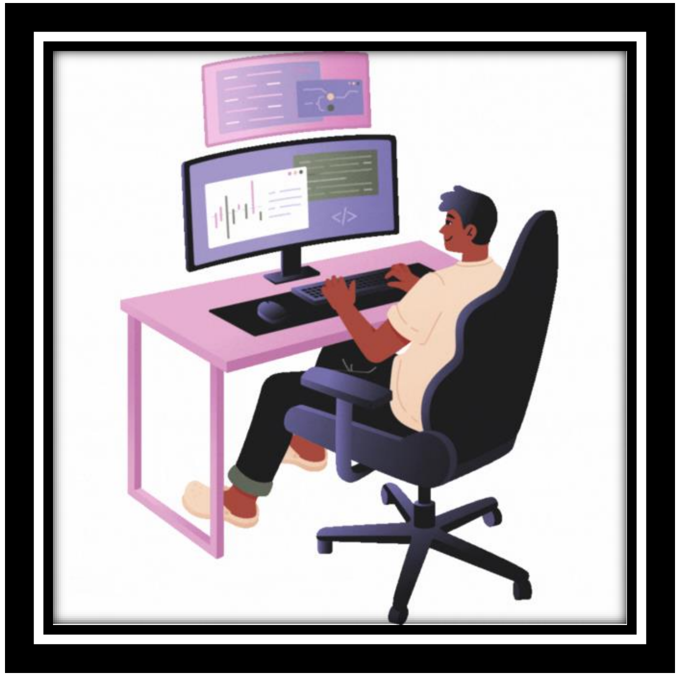
Copyright © Thinknyx Technologies LLP

How does this course work?

Copyright © Thinknyx Technologies LLP

How does this course work?

Core Concepts



Hands-On Demonstrations

Assessments

Copyright © Thinknyx Technologies LLP

Container Orchestration

Amazon EKS

Setting up IAM roles

Installing CLI tools

Launching first EKS Cluster

Upgrade EKS cluster

Explore worker node options

Managed Node Groups
AWS Fargate

Load Balancing

Copyright © Thinknyx Technologies LLP

Storage options

EBS and EFS

AWS Load Balancer Controller

Cluster Autoscaler

Amazon ECR

Real-World Demonstrations

Architectural Insights

Practical Use Cases

Copyright © Thinknyx Technologies LLP

Explore Our In-depth Courses

Build and Scale with AWS Cloud - A Hands-On Beginners Guide

Practical Kubernetes – Beyond CKA and CKAD | Hands-On

Copyright © Thinknyx Technologies LLP

Copyright © Thinknyx Technologies LLP

Introduction to Amazon EKS

Copyright © Thinknyx Technologies LLP

Introduction to Amazon EKS

Section Overview

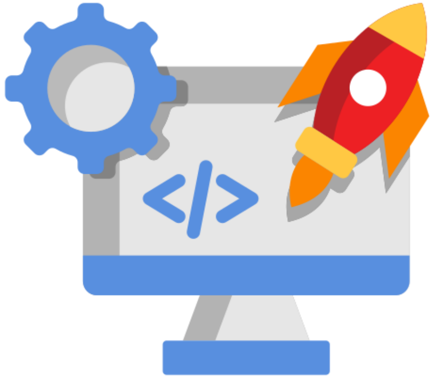
- → *Fundamentals of Container Orchestration*
- → *Challenges of Managing Kubernetes at Scale*
- → *Introduction to Amazon EKS (Elastic Kubernetes Service)*

Copyright © Thinknyx Technologies LLP

Demystifying Container Orchestration

Copyright © Thinknyx Technologies LLP

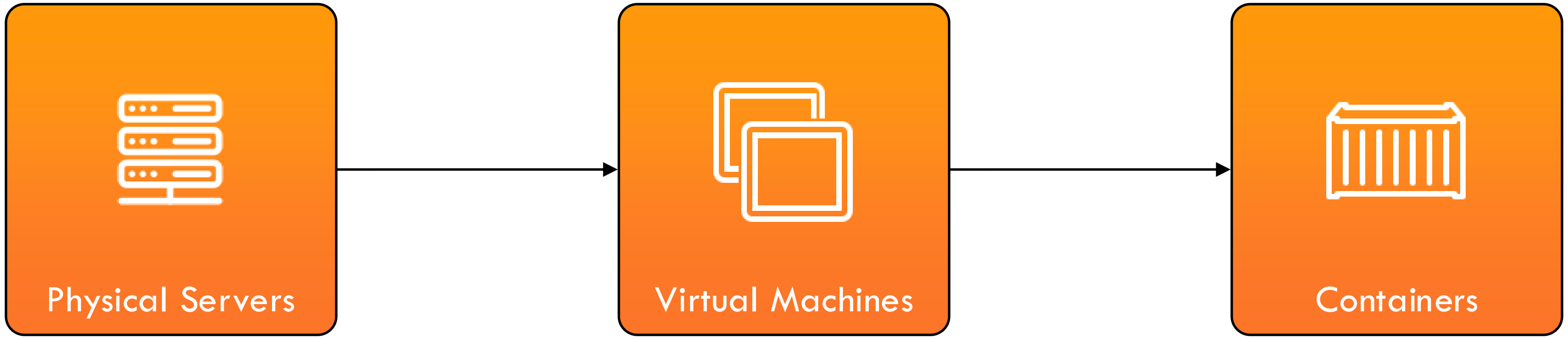
Demystifying Container Orchestration



Why container orchestration is essential ?

Copyright © Thinknyx Technologies LLP

Demystifying Container Orchestration



Copyright © Thinknyx Technologies LLP

Containers

- Faster development and deployment
- Consistent application behavior across environments
- Portability: build anywhere, run anywhere (laptop, VM, cloud, bare metal)
- Smooth transitions between testing, staging, and production

Copyright © Thinknyx Technologies LLP

Advantages of Containers

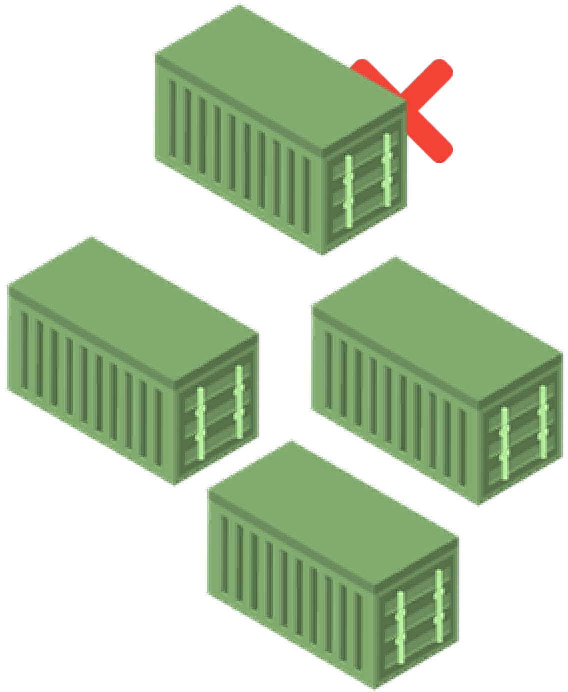
- Faster startup times, often in just a few seconds
- Lower resource usage, since containers don't need a full guest operating system
- Greater portability, making them easy to move across platforms without modification

Containers are a great way to package and run applications

Copyright © Thinknyx Technologies LLP

The Production-Scale Problem

- Real-world deployments just run one or two containers
- Organizations often run hundreds to thousands of containers
- Containers are distributed across multiple servers and regions
- Microservices architecture adds further complexity
- Applications are broken into small, independent services
- Services communicate with each other across the network
- Manual container management becomes impractical
- High risk of human error and operational inefficiency



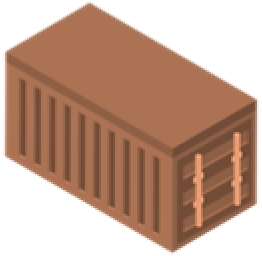
Copyright © Thinknyx Technologies LLP

Key Challenges in Managing Containers at Scale

- Automatic scaling
- Self-healing
- Load balancing
- Rolling updates and rollbacks
- High availability
- Data integrity
- Service discovery
- Persistent storage
- Security

Copyright © Thinknyx Technologies LLP

Container Orchestration



- Solved the packaging and portability problems
- Didn't solve the coordination problem

Container Orchestration

Copyright © Thinknyx Technologies LLP

Container Orchestration

Container Orchestration

- Deploying containers across multiple nodes
- Automatically scaling workloads based on traffic
- Replacing failed containers to ensure high availability
- Distributing traffic efficiently across services
- Managing networking, updates, and rollbacks - often with zero downtime

Copyright © Thinknyx Technologies LLP

Why Kubernetes?

- A powerful container orchestration platform
- Used to deploy, scale, and manage containerized applications
- Creates self-contained environments for applications
- Integrates compute, networking, storage, and configuration

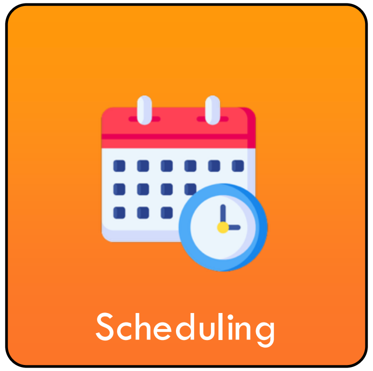
Copyright © Thinknyx Technologies LLP

Why Kubernetes?

- Open source and cloud-agnostic
- Capable of running in any environment - on-premises, cloud, or hybrid
- Based on declarative configuration, making it easy to define your app’s desired state
- Integrated with modern DevOps pipelines and CI/CD workflows
- Supported by a large and active open-source community

Copyright © Thinknyx Technologies LLP

Kubernetes Features at a Glance



- Core Function of Kubernetes
- Resource-Based Decisions
- Policy & Constraint Driven
- Efficient Cluster Utilization
- Workload Balancing

Copyright © Thinknyx Technologies LLP

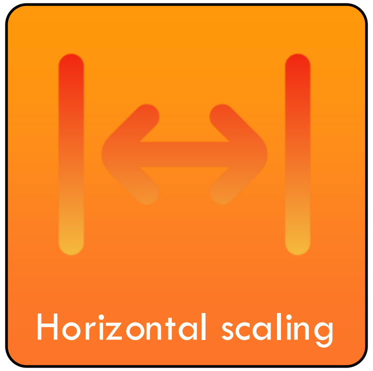
Kubernetes Features at a Glance

Self-healing & high availability

- Automatic Restart
- Pod Rescheduling
- Minimized Downtime
- Built-in Resilience

Copyright © Thinknyx Technologies LLP

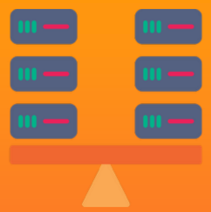
Kubernetes Features at a Glance



- Auto-scaling Triggers
- Efficient Resource Utilization

Copyright © Thinknyx Technologies LLP

Kubernetes Features at a Glance



Service discovery & load balancing

- Built-in DNS service for internal communication
- Containers communicate using simple service names
- Automatic traffic distribution across healthy pods
- Eliminates need for manual IP address management

Copyright © Thinknyx Technologies LLP

Kubernetes Features at a Glance

Rolling updates & rollbacks

- Gradual deployment of new application versions
- Minimizes service disruption during updates
- No Downtime Deployment
- Automatic Rollbacks

Copyright © Thinknyx Technologies LLP

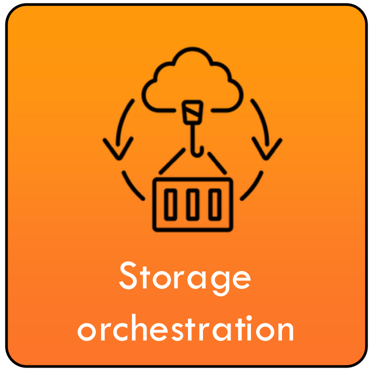
Kubernetes Features at a Glance

Secrets & configuration management

- Inject environment-specific data at runtime
- Manage sensitive data like passwords and API keys securely

Copyright © Thinknyx Technologies LLP

Kubernetes Features at a Glance



- Automatically Mount Storage Volumes
- Supports Local Storage, Public Cloud Providers, and NFS
- Simplifies Persistent Data Management

Copyright © Thinknyx Technologies LLP

Kubernetes Features at a Glance

- Custom controllers in Kubernetes
- Extend Kubernetes functionality
- Ideal for complex, stateful applications (e.g., databases)
- Automate routine operational tasks
  - Provisioning
  - Scaling
  - Replication
  - Failover
- Work based on declared user intent

Copyright © Thinknyx Technologies LLP

Move faster, scale confidently, and run applications anywhere

Copyright © Thinknyx Technologies LLP

Limitations of Native Kubernetes

Copyright © Thinknyx Technologies LLP

Limitations of Native Kubernetes

Container Orchestration

Copyright © Thinknyx Technologies LLP

Key Limitations

Steep Learning Curve

- Kubernetes setup and management require advanced skills in networking, RBAC, storage, and security - challenging for beginners

Manual Cluster Management

- Tasks like installation, scaling, and patching are often manual, leading to errors and downtime risks

Security Requires Extra Effort

- Enterprise-grade security needs custom setups and external tools beyond Kubernetes' built-in capabilities

Fragmented Tooling

- No built-in solutions for monitoring, logging, or CI/CD - teams must integrate and maintain separate tools

Copyright © Thinknyx Technologies LLP

Enterprise Kubernetes Platforms

Built on top of native Kubernetes but include enhanced tooling, security features, automation, and vendor support, making it easier and safer to use in large companies

Red Hat OpenShift

vmware Tanzu

RANCHER BY SUSE

Copyright © Thinknyx Technologies LLP

Managed Kubernetes Service

Managed Kubernetes services reduce the complexity of deploying, managing, and scaling Kubernetes at production scale

Amazon Elastic Kubernetes Service

Google Kubernetes Engine

Azure Kubernetes Service

Handles infrastructure and Kubernetes management, so you can focus on your applications

Copyright © Thinknyx Technologies LLP

Some Leading Managed Kubernetes Services

Amazon Elastic Kubernetes Service

Azure Kubernetes Service

Google Kubernetes Engine

IBM Cloud Kubernetes Service

Oracle Container Engine for Kubernetes (OKE)

Alibaba Cloud ACK (Container Service for Kubernetes)

Platform9 Managed Kubernetes

Red Hat OpenShift

Red Hat | aws

Red Hat | Microsoft Azure

Red Hat OpenShift Dedicated
Available on Google Cloud

Red Hat | IBM.

Copyright © Thinknyx Technologies LLP

What is Amazon EKS?

Copyright © Thinknyx Technologies LLP

- Limitations of Native Kubernetes
- Managed Kubernetes services

Amazon EKS

Copyright © Thinknyx Technologies LLP

What is Amazon EKS?

- Fully managed, certified Kubernetes-conformant service
- Compatible with existing Kubernetes tools and manifests
- Enables easy workload migration across environments
- Eliminates the need to re-architect applications
- Simplifies Kubernetes setup and operations at scale
- Provides production-grade clusters with minimal effort
- AWS manages the control plane, availability, and security

Amazon EKS

Users can focus on deploying and running containerized apps

Copyright © Thinknyx Technologies LLP

Why Choose Amazon EKS?

- Install
- Operate
- Upgrade

Kubernetes control plane

Multiple Availability Zones for high availability and resilience

- For Compute layer (Data plane)
  - Use self-managed EC2 nodes for total control
  - Choose EKS Auto Mode managed nodes for simplified operations
  - Go serverless with AWS Fargate, without managing any infrastructure

Copyright © Thinknyx Technologies LLP

Features of Amazon EKS

Built-in High Availability

Multiple AZs, automatically scaling and replacing unhealthy nodes, Stay highly available and resilient - even during failures

Integrated Security and Access Control

Integrates with AWS IAM and Kubernetes RBAC, Works with Secrets Manager and AWS Key Management Service

Autoscaling and Flexibility

Kubernetes-native tools like Cluster Autoscaler, or modern solutions like Karpenter, EKS automatically provisions and scales infrastructure

Persistent Storage Made Easy

EKS supports native Persistent Volumes (PVs and PVCs) using Amazon EBS, EFS, and even S3 via CSI drivers

Observability and Monitoring

CloudWatch Container Insights, and native Prometheus, AWS Distro for OpenTelemetry (ADOT)

Easy Management Interfaces

AWS Console, eksctl CLI, AWS CLI, CloudFormation, AWS CDK, Terraform

Copyright © Thinknyx Technologies LLP

Features of Amazon EKS

Hybrid and Edge Support

Run EKS clusters in your on-prem data center using the same components as AWS-hosted clusters. And with EKS on Outposts, you can extend Kubernetes to edge locations

Seamless Integration with Amazon ECR

Integrates directly with Amazon ECR, a fully managed container registry that provides secure, scalable, and high-performance image storage

Copyright © Thinknyx Technologies LLP

Introduction to Amazon EKS

Section Summary

→ *Importance of container orchestration*
→ *Limitations of native Kubernetes*
→ *Amazon EKS (Elastic Kubernetes Service)*
→ *Key features of Amazon EKS:*
- *High availability*
- *Autoscaling*
- *Enhanced security*
- *Observability*
- *Integration with AWS tools (e.g., ECR)*

Copyright © Thinknyx Technologies LLP

Getting Started with Amazon EKS

Copyright © Thinknyx Technologies LLP

Getting Started with Amazon EKS

Section Overview

- → *Introduction to Amazon EKS setup*
- → *Brief overview of EKS architecture*
- → *Create required IAM roles for EKS*
- → *Install kubectl and AWS CLI*

Copyright © Thinknyx Technologies LLP

Understanding EKS Architecture

Copyright © Thinknyx Technologies LLP

What is a Kubernetes Cluster?

- A **node** is a physical or virtual machine (e.g., an EC2 instance) that runs your workloads in the form of pods
- A **Kubernetes cluster** is a group of nodes working together to manage and run containerized applications

Copyright © Thinknyx Technologies LLP

Kubernetes Cluster

- Types of Nodes

Control Plane

Acts as the brain of the cluster

Worker Nodes

Runs the applications

Scheduling

Scaling

Health Checks

Copyright © Thinknyx Technologies LLP

Control Plane Components

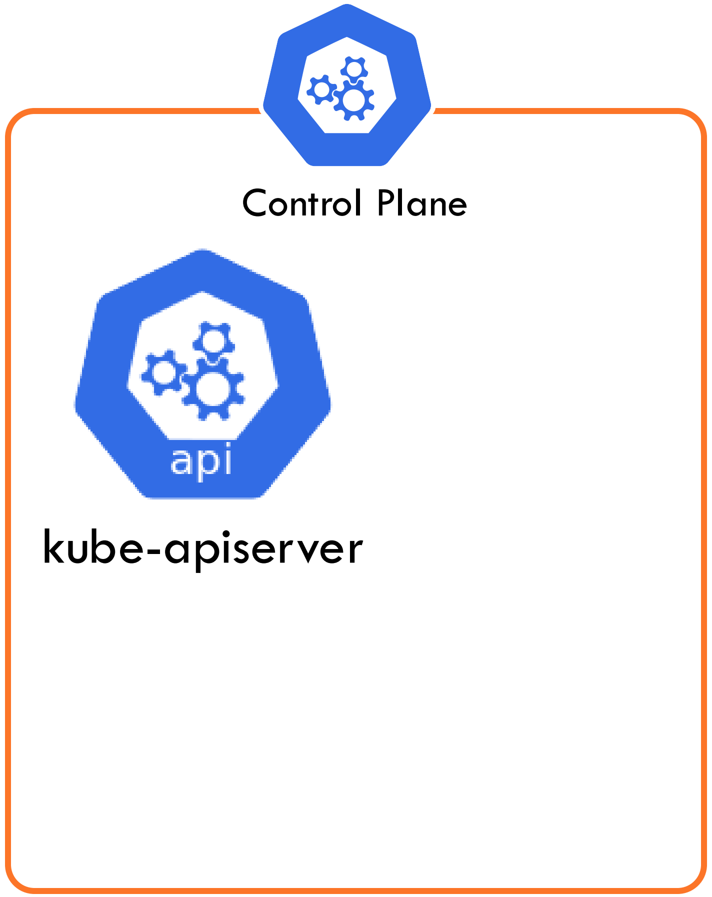
Control Plane

kube-apiserver

- ✓ Main entry point for all commands and communication

Copyright © Thinknyx Technologies LLP

Control Plane Components

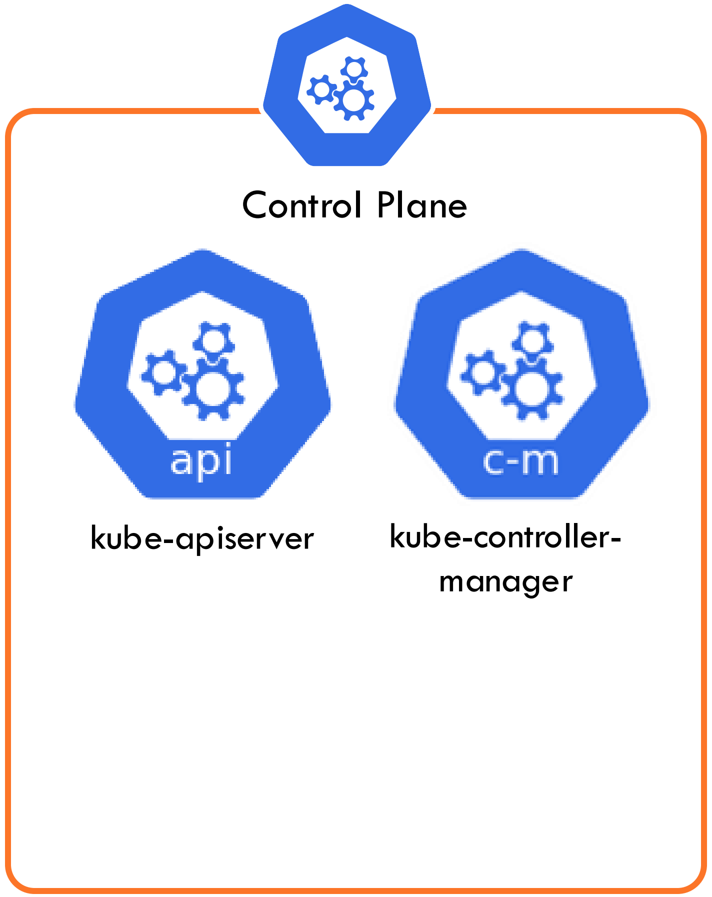


kube-controller-manager

- ✓ Monitors current cluster state & compares it to desired state

Copyright © Thinknyx Technologies LLP

Control Plane Components



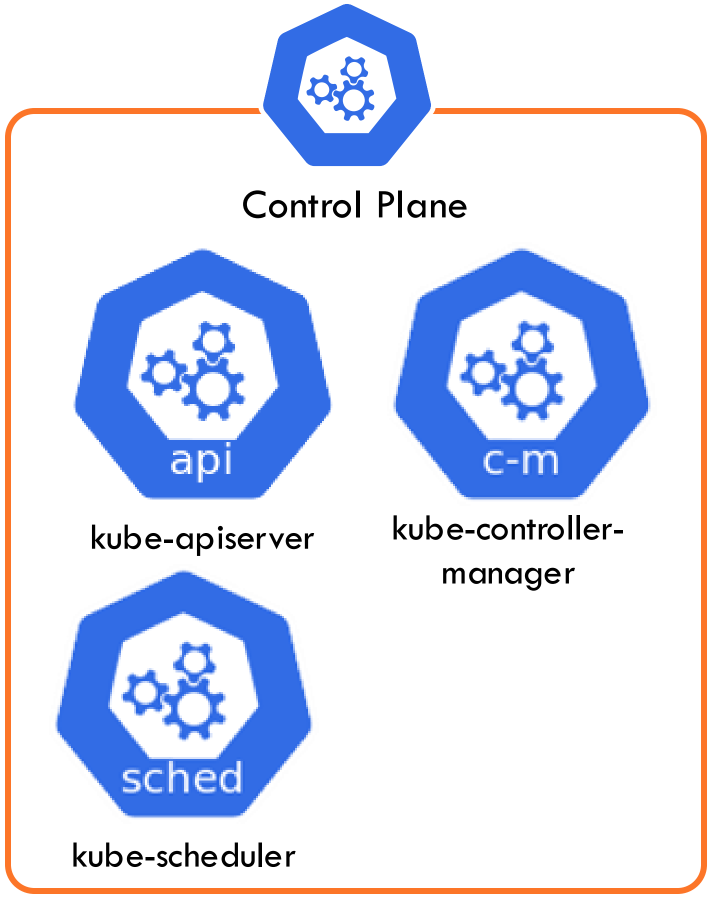
sched

kube-scheduler

✓ Acts as the placement officer

Copyright © Thinknyx Technologies LLP

Control Plane Components

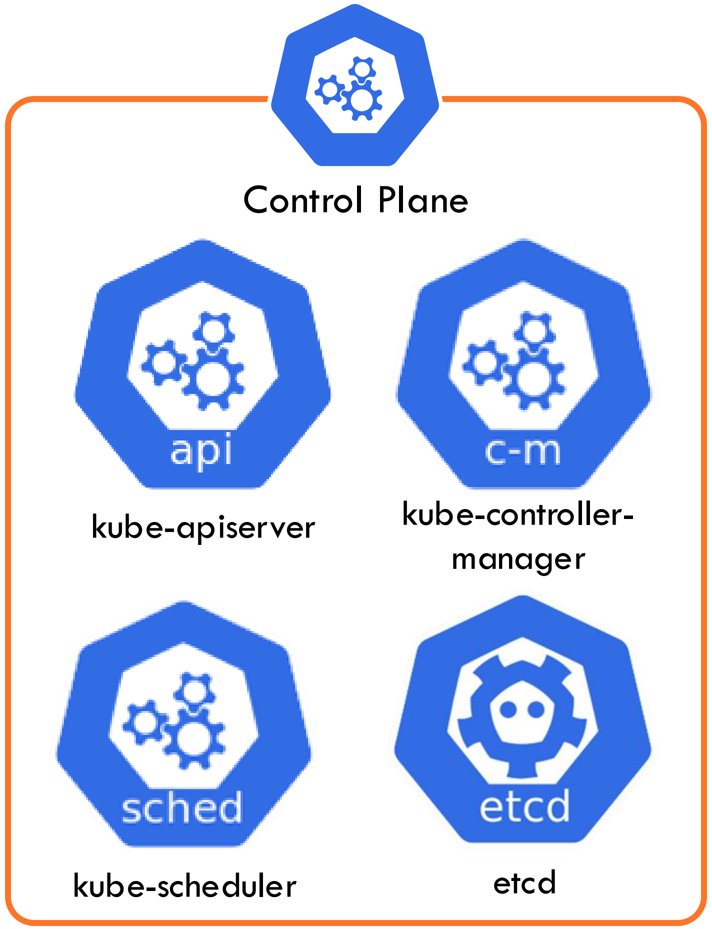


etcd

✓ Key-value store that keeps track of the entire cluster state

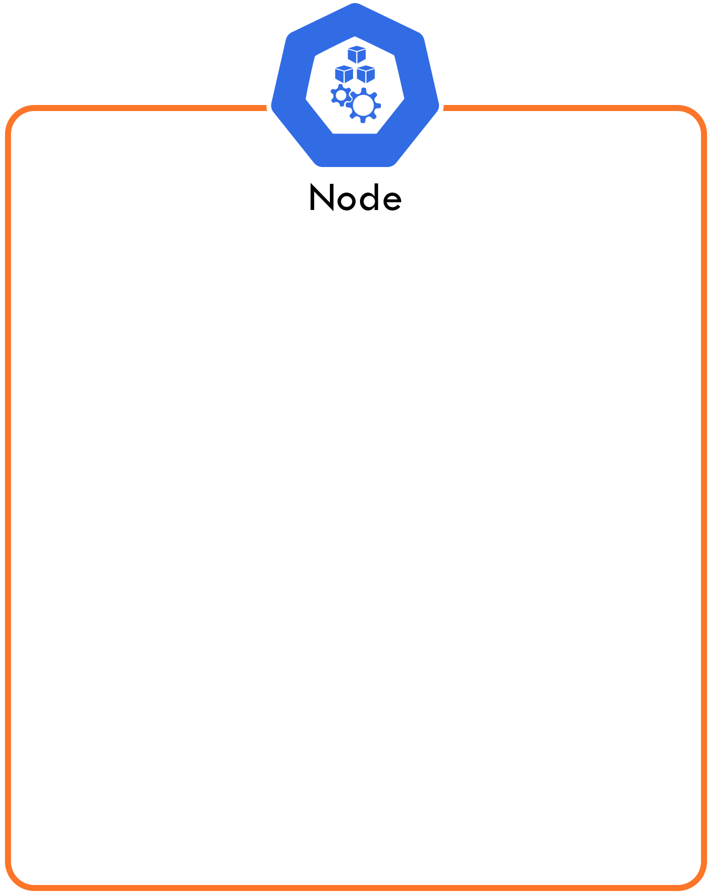
Copyright © Thinknyx Technologies LLP

Control Plane Components



Copyright © Thinknyx Technologies LLP

Node Components

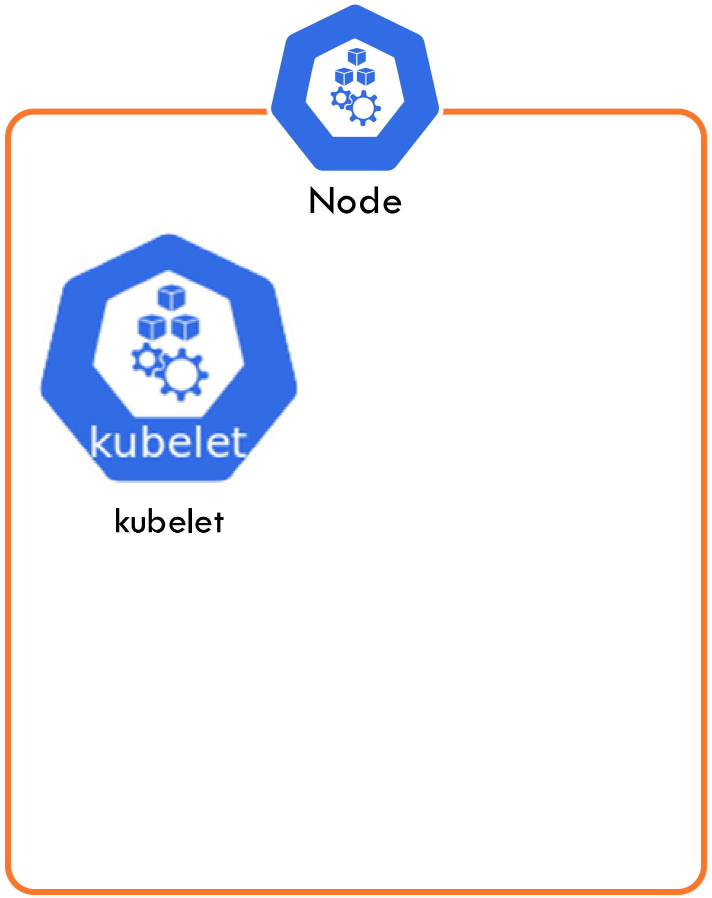


kubelet

- ✓ Communicates with API server, manages containers, and reports back

Copyright © Thinknyx Technologies LLP

Node Components

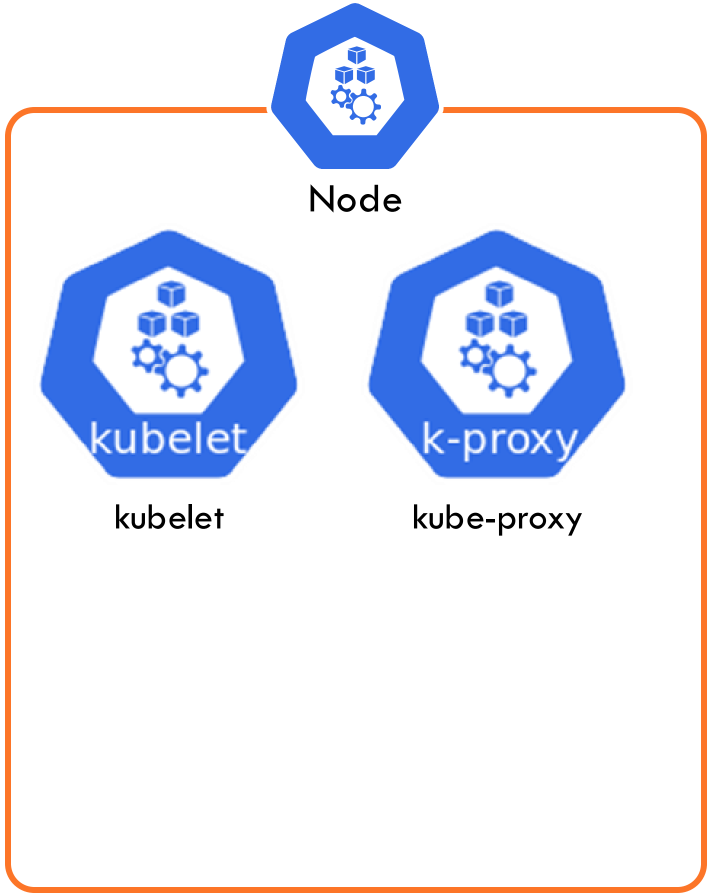


kube-proxy

- ✓ Acts as the network traffic director
- ✓ Ensures pods can communicate with each other seamlessly

Copyright © Thinknyx Technologies LLP

Node Components



Node

kubelet

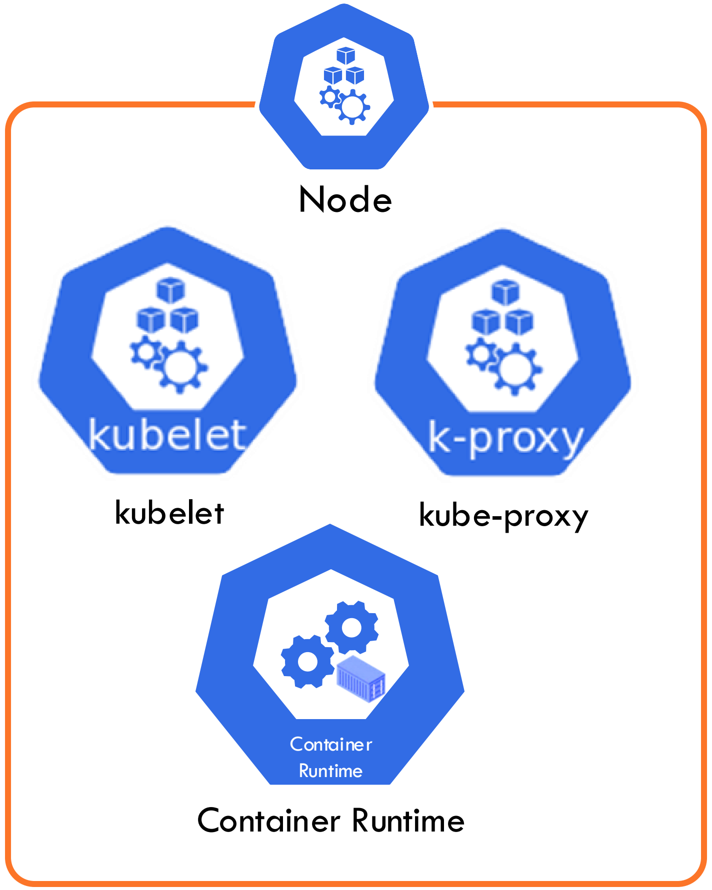
kube-proxy

Container Runtime

- ✓ Act as the engine that powers the containers

Copyright © Thinknyx Technologies LLP

Node Components



Copyright © Thinknyx Technologies LLP

Amazon EKS: What's Different?

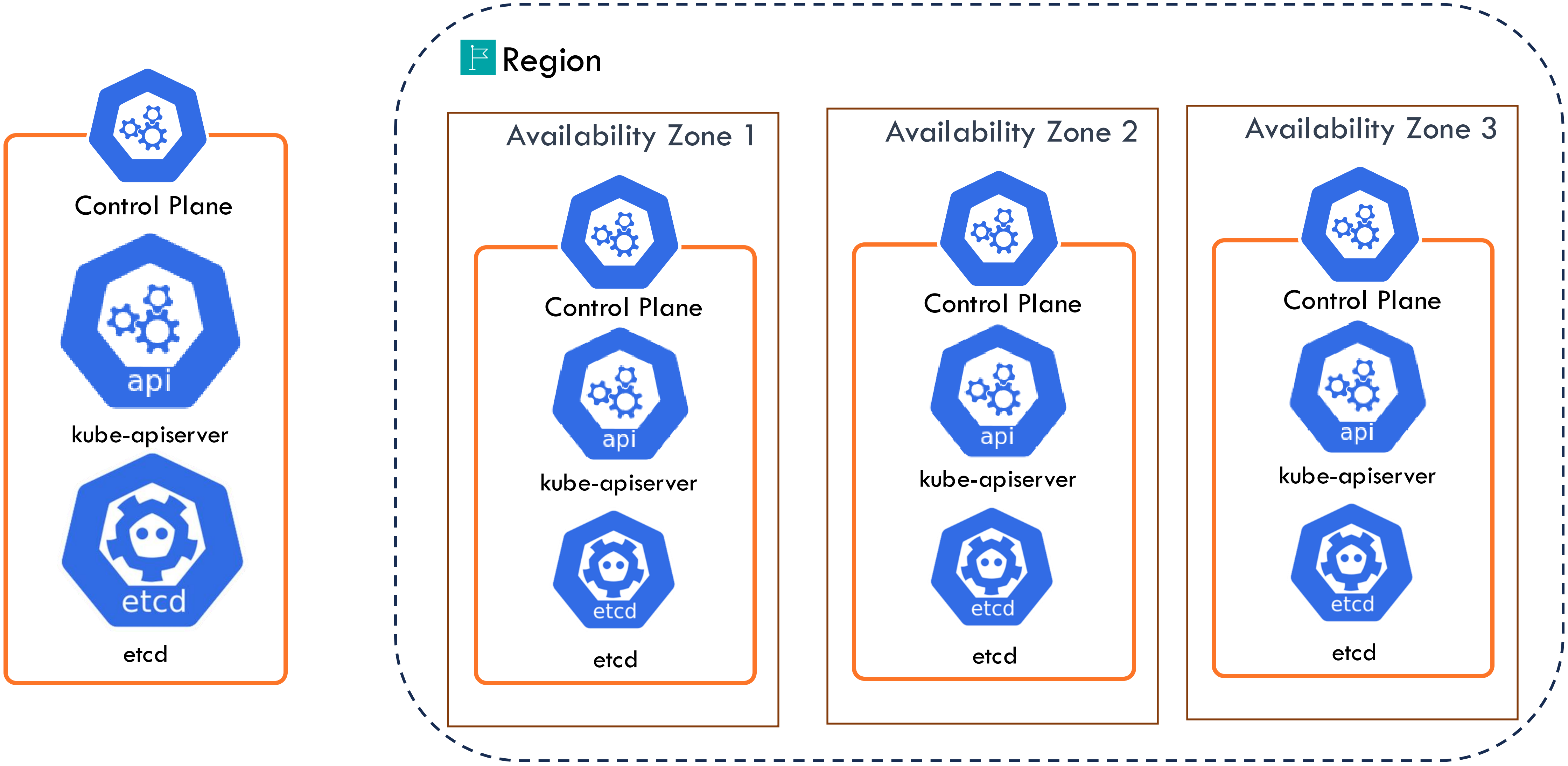
Amazon EKS

✓ AWS fully manages the **Control Plane**

- ✗ Install
- ✗ Upgrade
- ✗ Scale

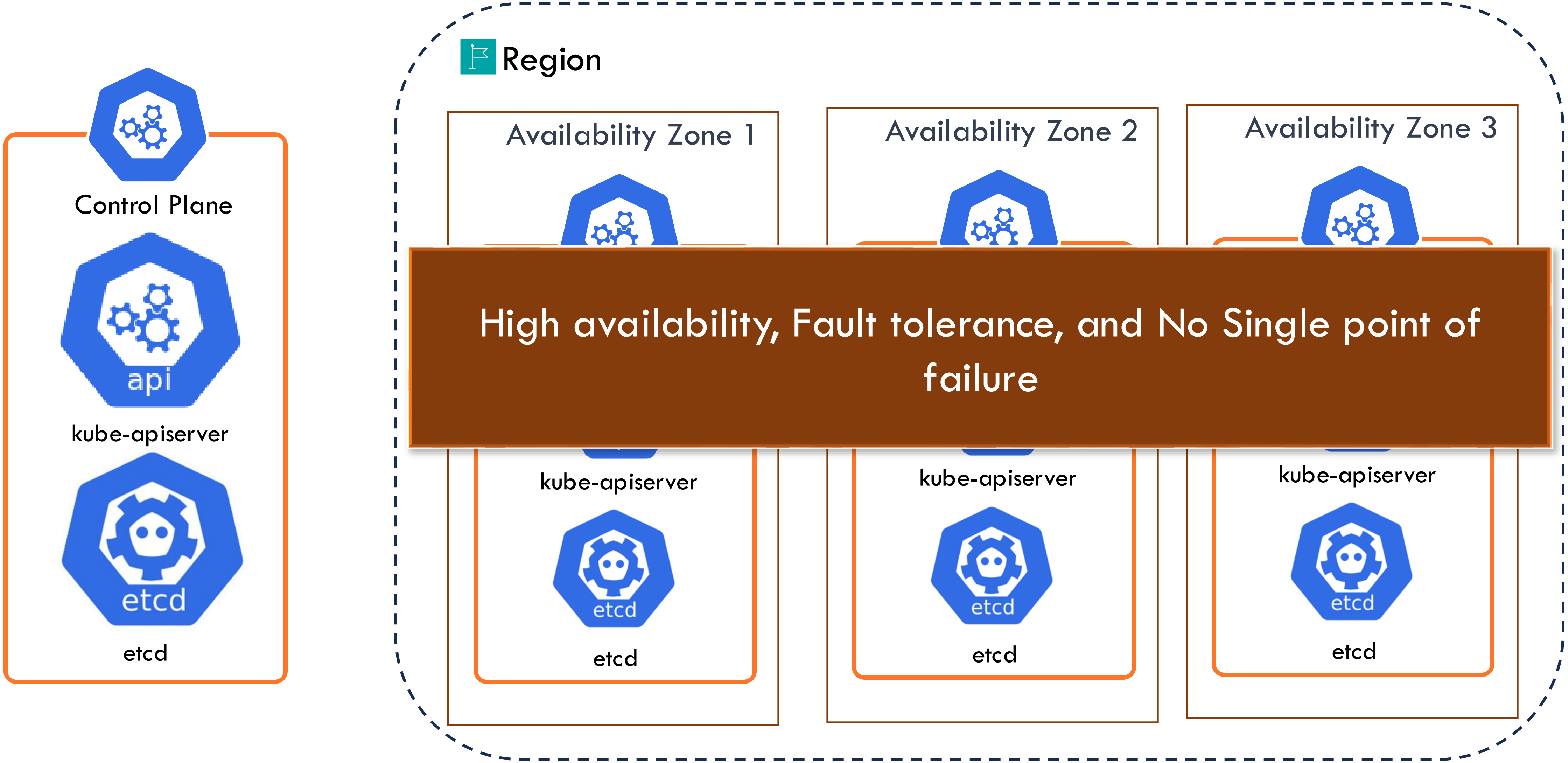
Copyright © Thinknyx Technologies LLP

Amazon EKS: What's Different?



Copyright © Thinknyx Technologies LLP

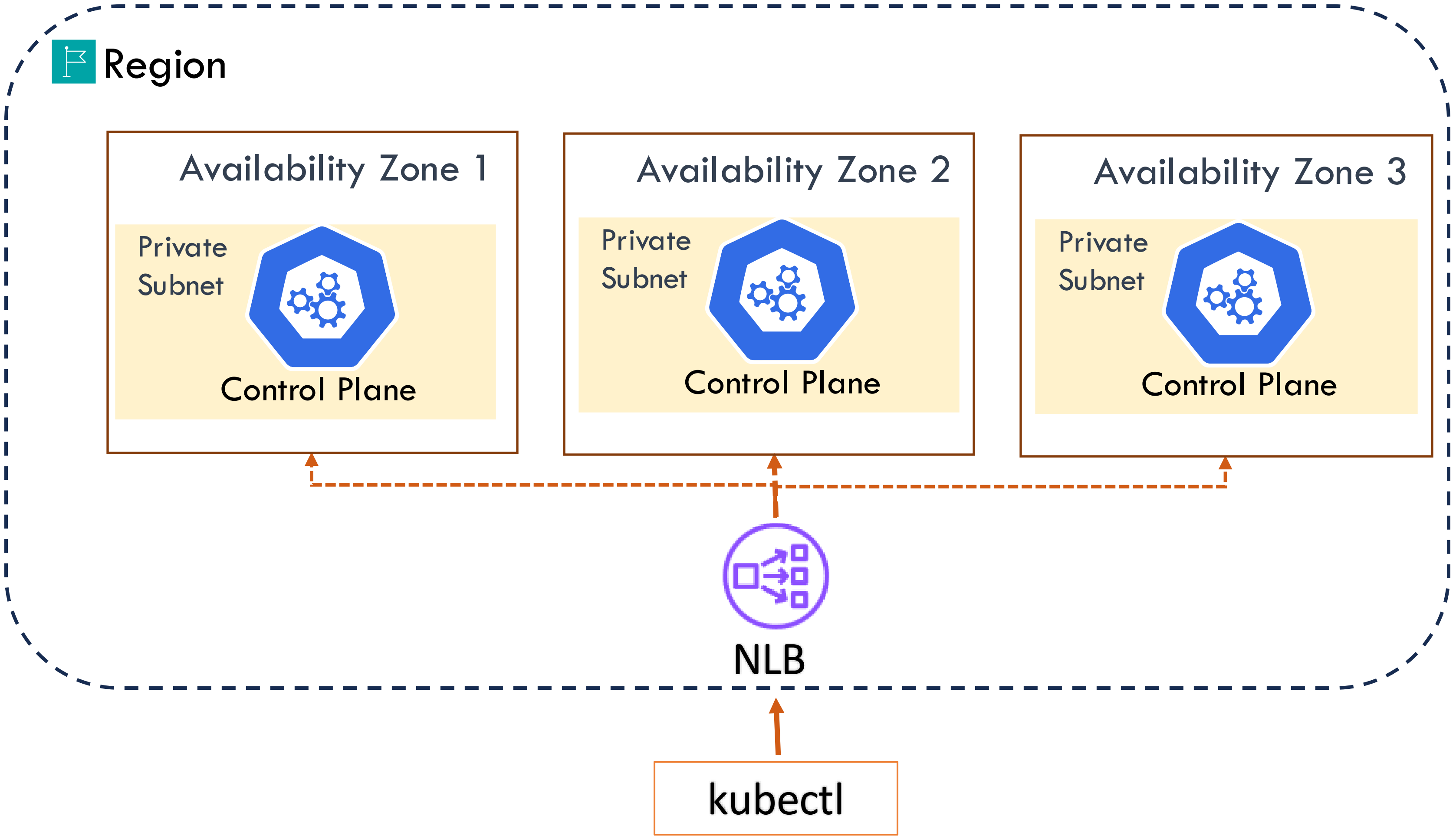
Amazon EKS: What's Different?



High availability, Fault tolerance, and No Single point of failure

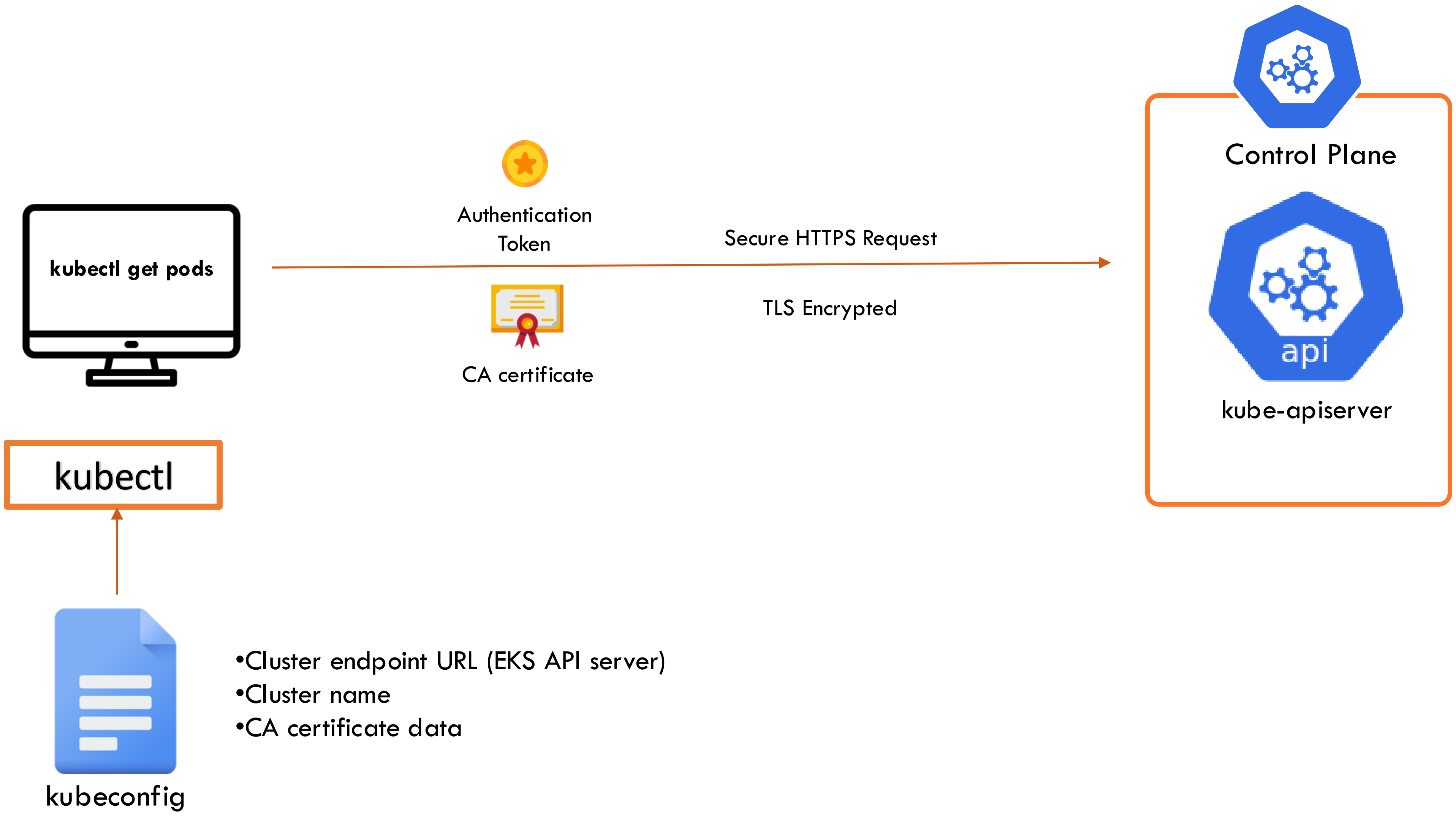
Copyright © Thinknyx Technologies LLP

Amazon EKS: What's Different?



Copyright © Thinknyx Technologies LLP

Connecting to EKS



- Cluster endpoint URL (EKS API server)
- Cluster name
- CA certificate data

Copyright © Thinknyx Technologies LLP

Managing Worker Nodes in EKS

Standard Mode

- You manage the worker nodes, while AWS manages the control plane
  - Maintain nodes yourself with **self-managed nodes**
  - Use **managed node groups** for AWS-assisted scaling and updates
- Nodes use **EKS-optimized AMIs** with preinstalled kubelet and container runtime

Copyright © Thinknyx Technologies LLP

Managing Worker Nodes in EKS

Standard Mode

- Nodes connect to the control plane after launch, given proper IAM and network setup
- Connection time may vary based on instance startup speed, network configurations, and cluster permissions
- Once connected, nodes are ready to run workloads

Copyright © Thinknyx Technologies LLP

Managing Worker Nodes in EKS

Auto Mode

- AWS manages both control plane and worker nodes
- Handles EC2 provisioning, scaling, patching, and security
- Ideal for teams looking to reduce operational overhead

Copyright © Thinknyx Technologies LLP

Managing Worker Nodes in EKS

Fargate Mode

- AWS's serverless option for running pods
- No need to manage any EC2 instances
- AWS provisions compute resources per pod
- Only pay for what you use
- Best suited for small, bursty, or event-driven workloads

❌ Limitations: No support for daemonsets, privileged containers, or AWS EBS as PV

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

Amazon EKS

How different components work together?

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

Cluster

Node Groups

Fargate Profiles

Load Balancers

Persistent Storage

Advanced Networking

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

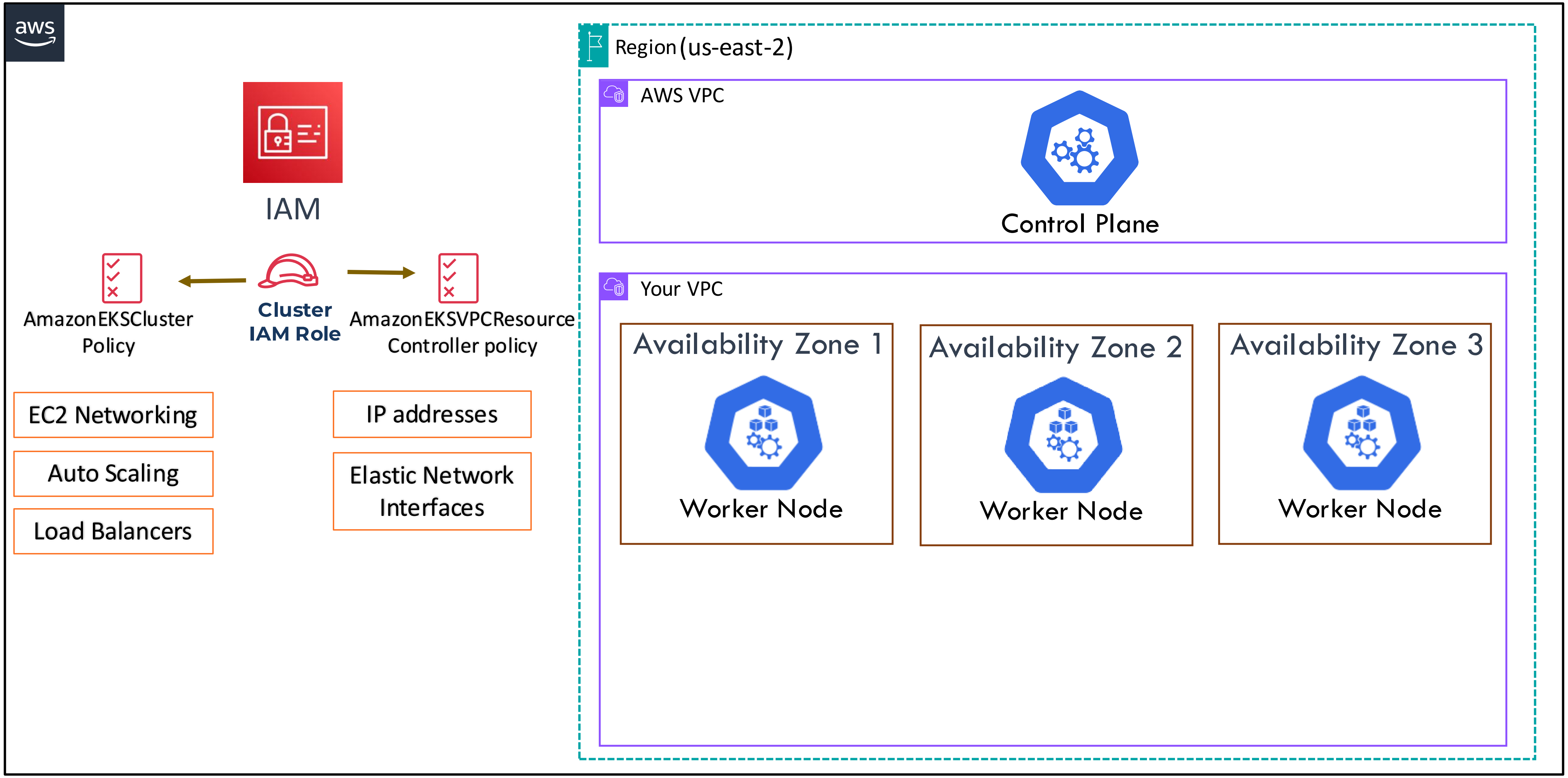
aws

Region (us-east-2)

The choice determines the physical location of the control plane and related networking resources

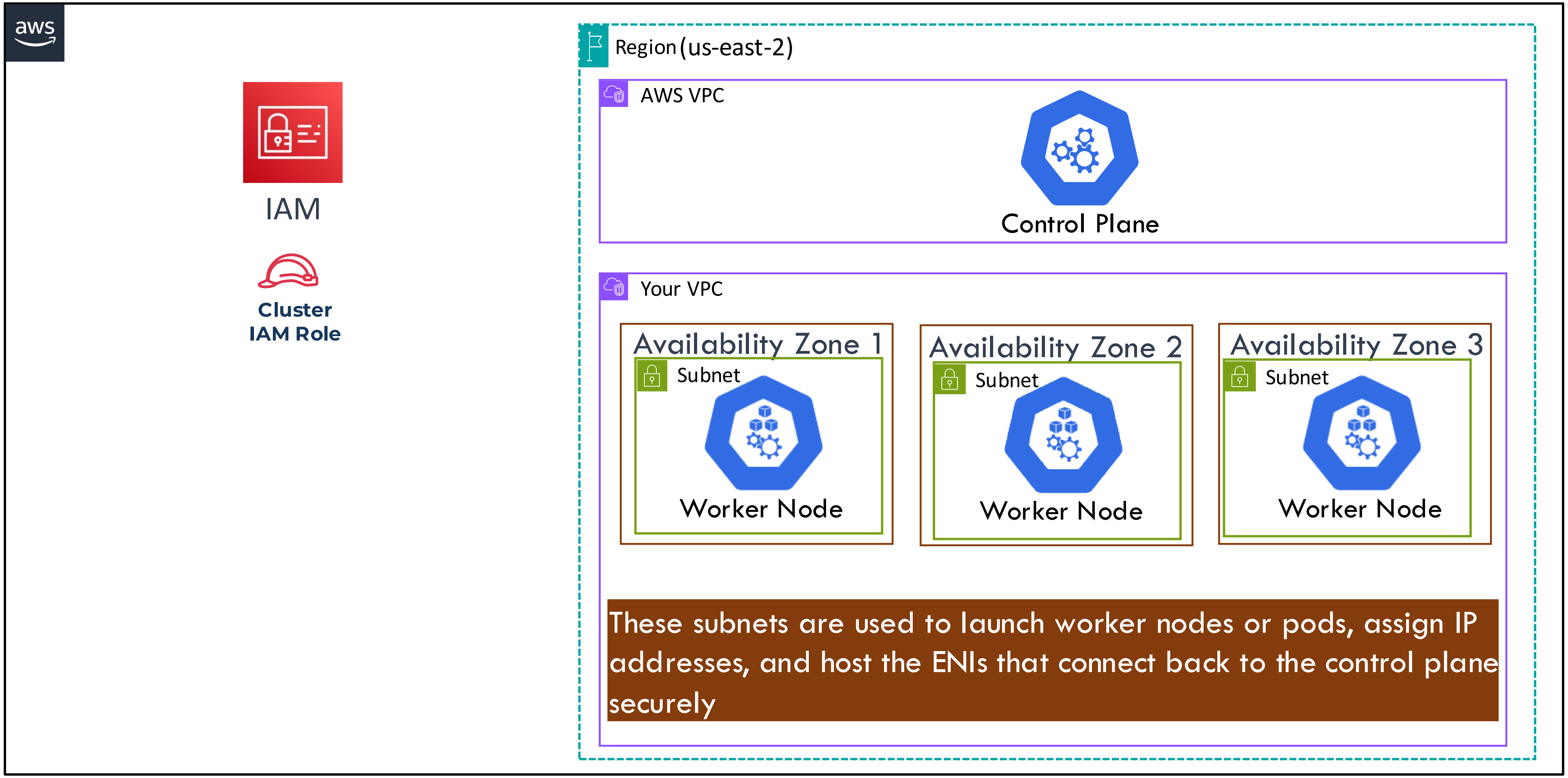
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



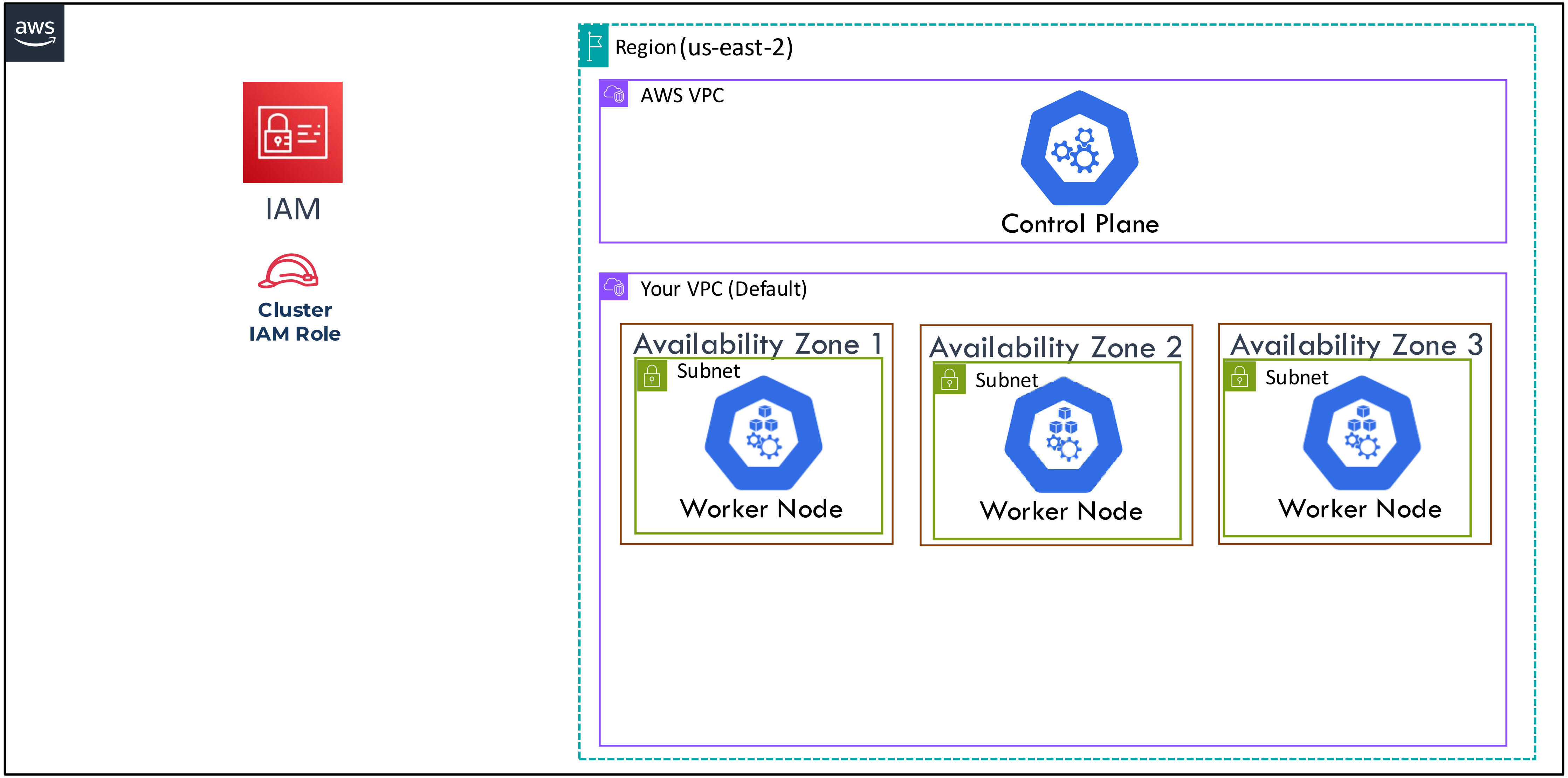
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



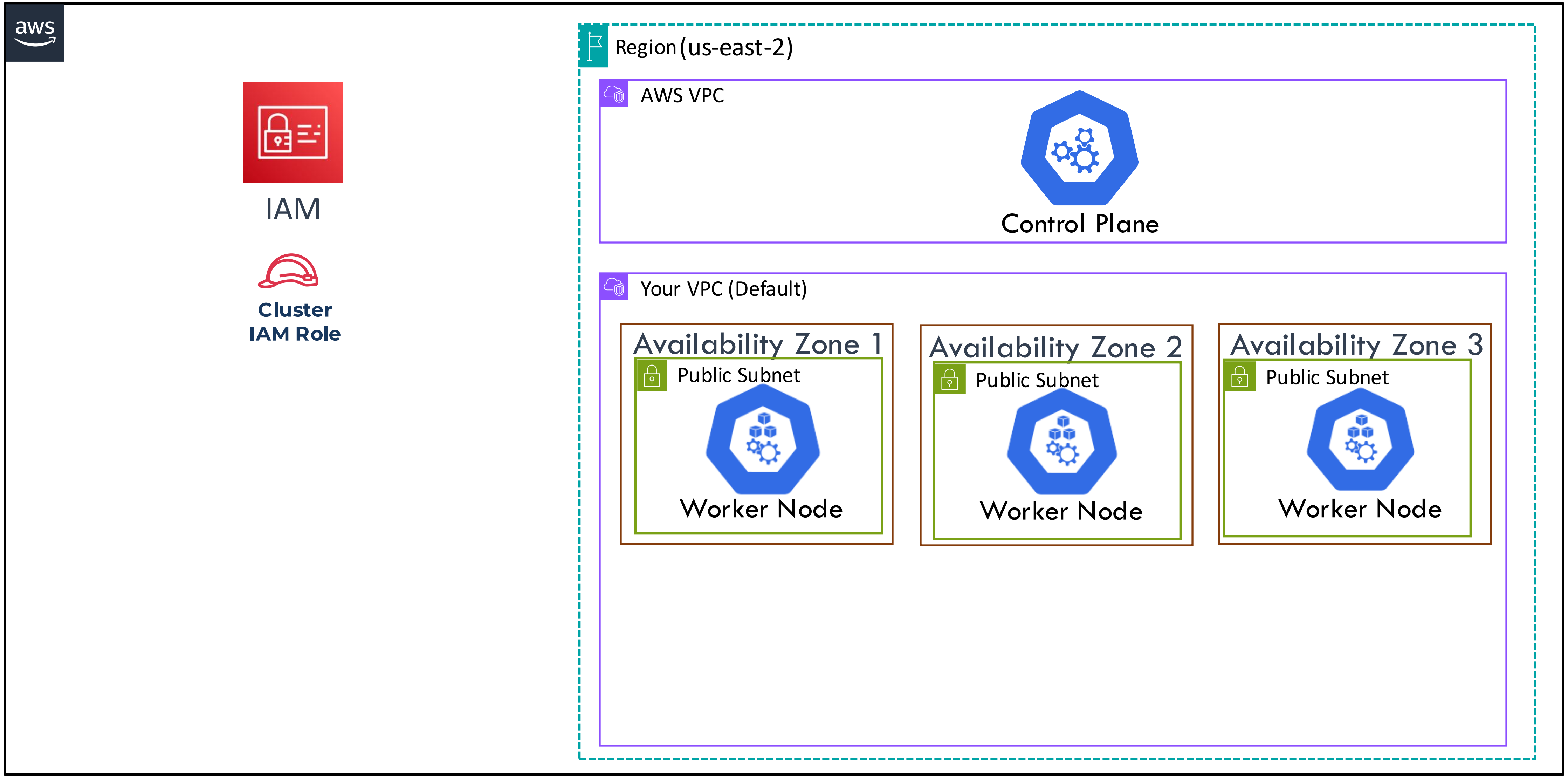
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



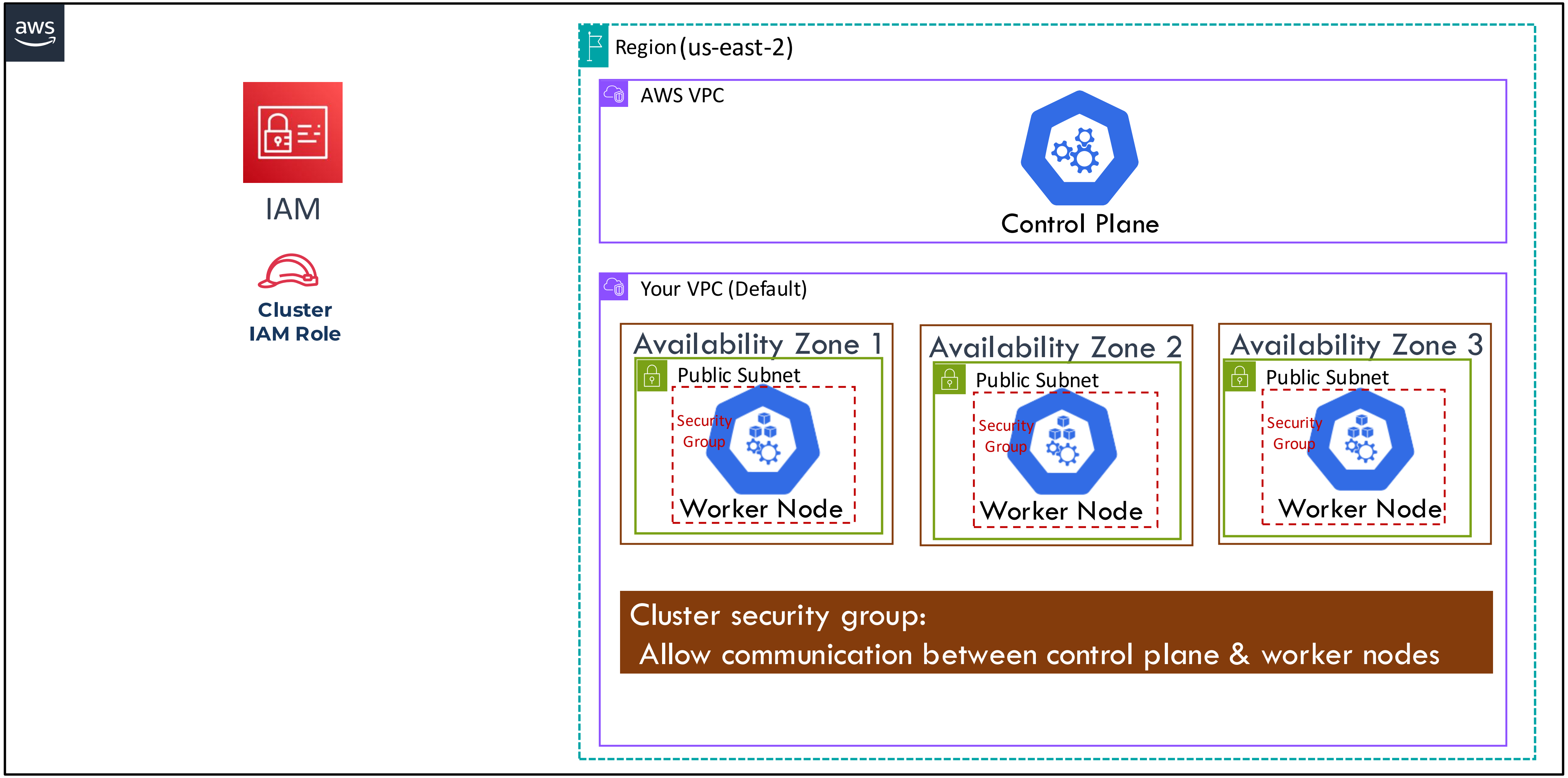
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



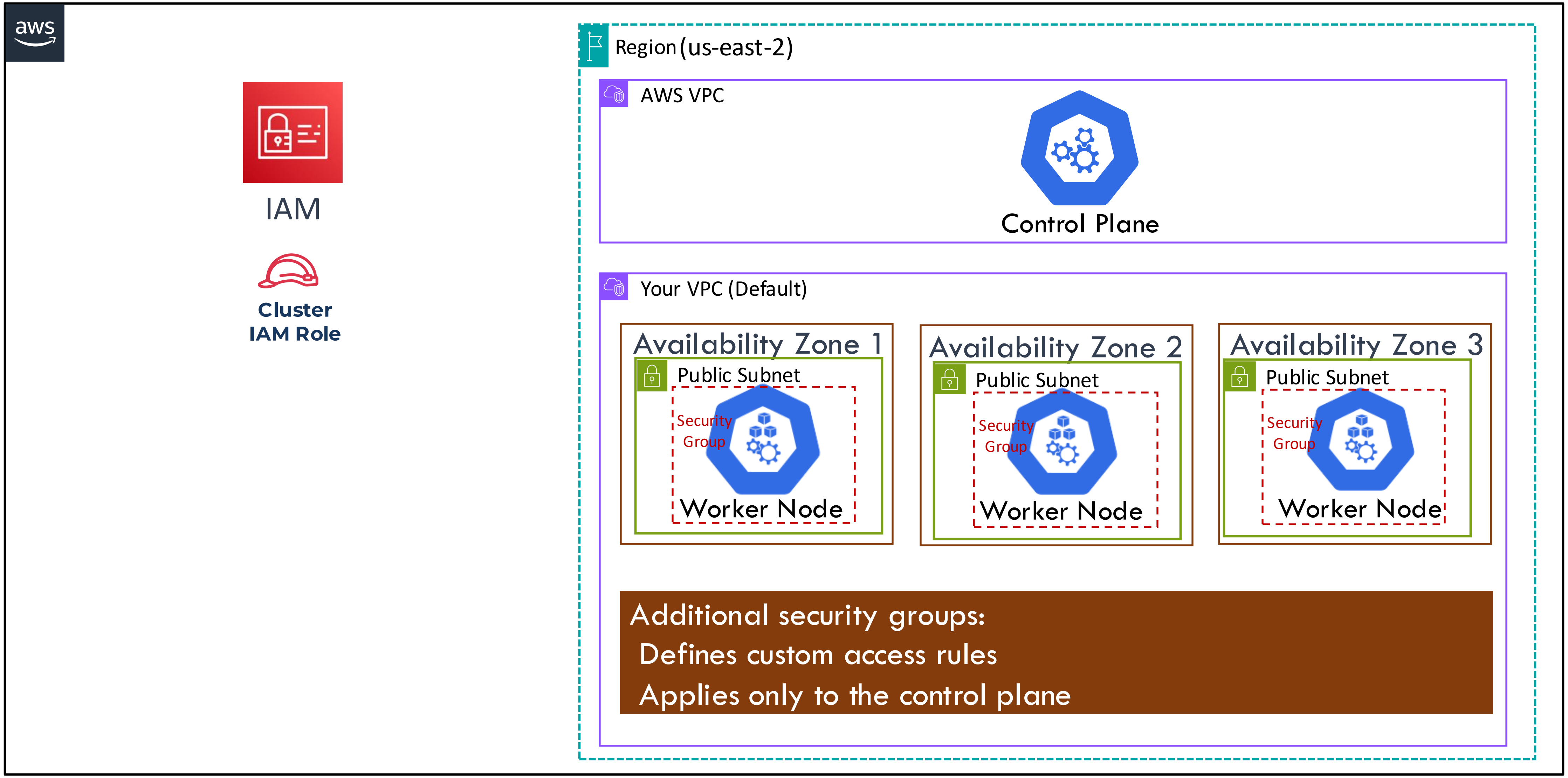
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



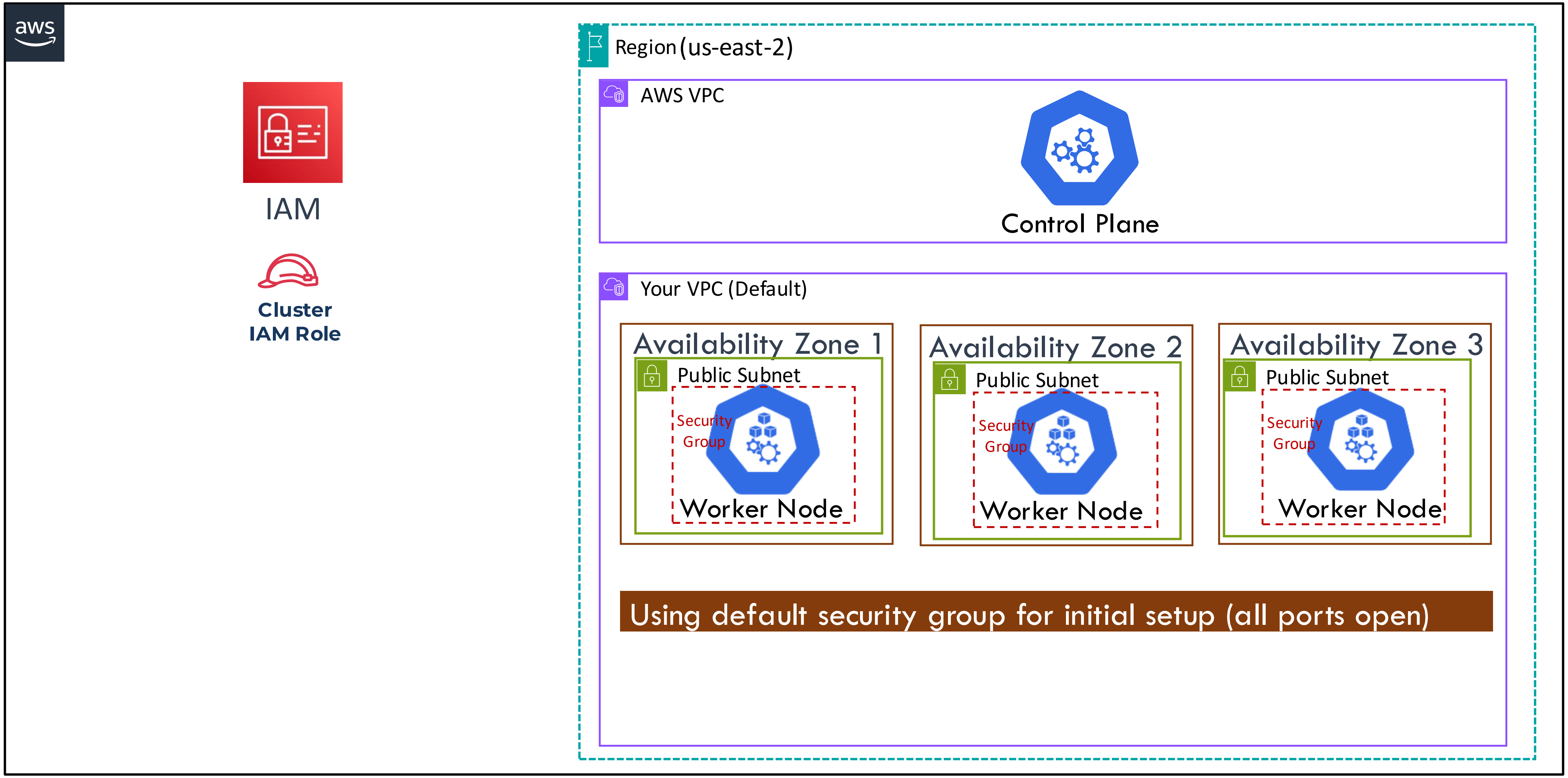
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



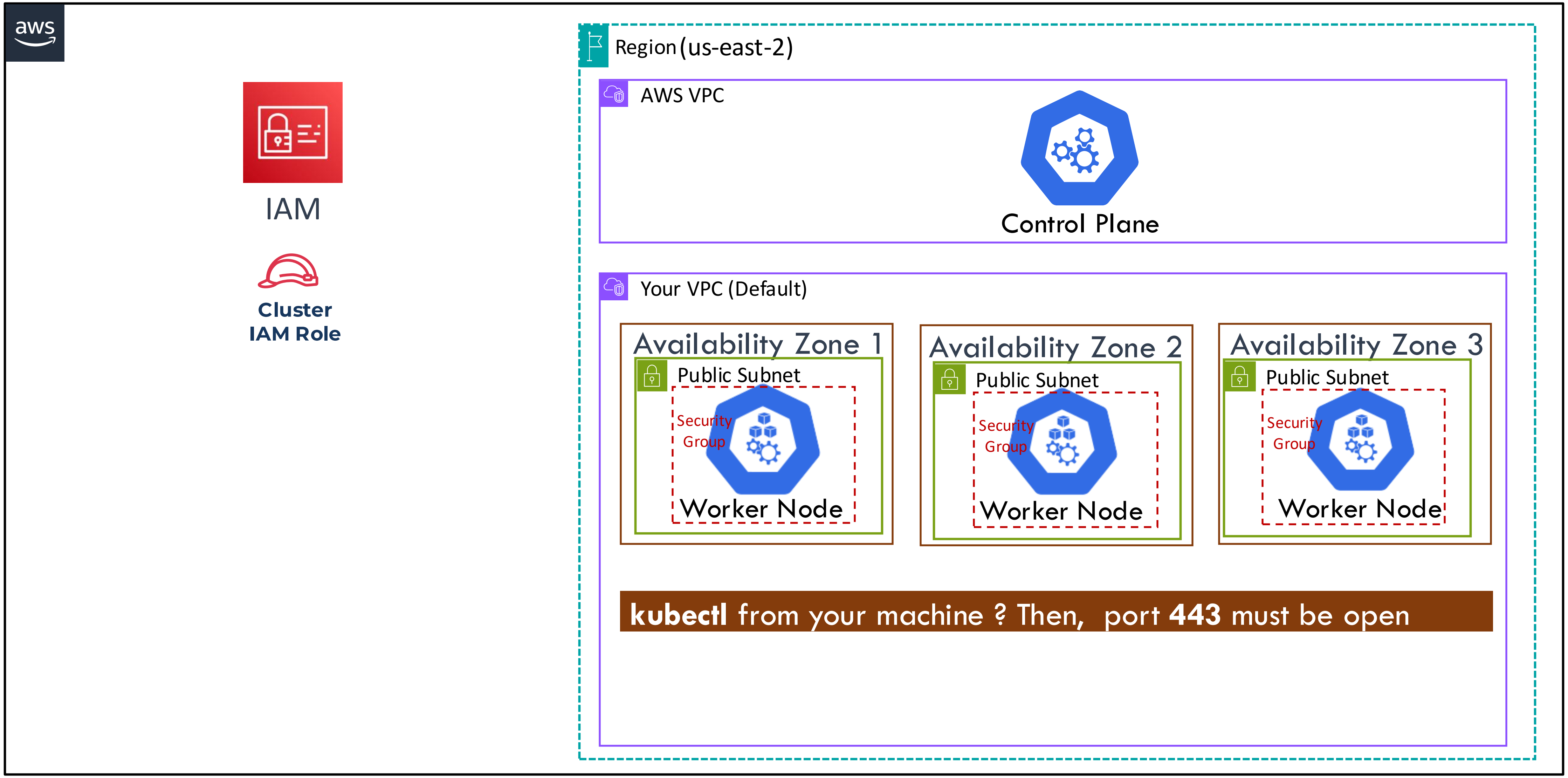
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- Cluster Endpoint Access
  - Public access
    - Public access allows the API server to be reachable over the internet
    - Enables remote access from anywhere
    - Less secure due to exposure to the public internet
    - Worker node traffic may leave the VPC

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- Cluster Endpoint Access
  - Private access
    - Private access restricts API server access to internal VPC only
    - Enhances cluster security by blocking public endpoints
    - External access requires VPN, bastion host, or private link

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- Cluster Endpoint Access
  - Public and private access
    - Combines public and private access for flexibility and security
    - API server is publicly accessible for admin tasks
    - Worker nodes communicate privately with the control plane
    - Communication stays within the VPC
    - Balances accessibility with security

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- EKS Add-ons

VPC CNI Plugin

CoreDNS

kube-proxy

Metrics Server

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

EKS Add-ons

Pod Identity Agent

- Enables secure IAM role assignment to individual pods
- Allows fine-grained access control for AWS services
- Must be installed externally (not included by default in EKS)
- Used in the EFS demo to show pod-level access without node credentials

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- EKS Add-ons

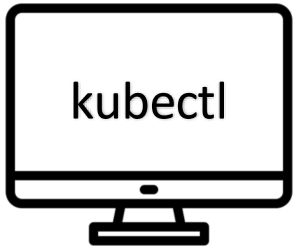
External DNS

- Helps manage DNS records automatically
- Node monitoring agent tracks node health and performance

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

Command Line Tools



- Standard Kubernetes CLI tool
- Use kubectl to run commands for managing Kubernetes resources
- Check pod status, deploy applications, and more using CLI
- kubectl communicates with the Kubernetes API server to perform actions

Copyright © Thinknyx Technologies LLP

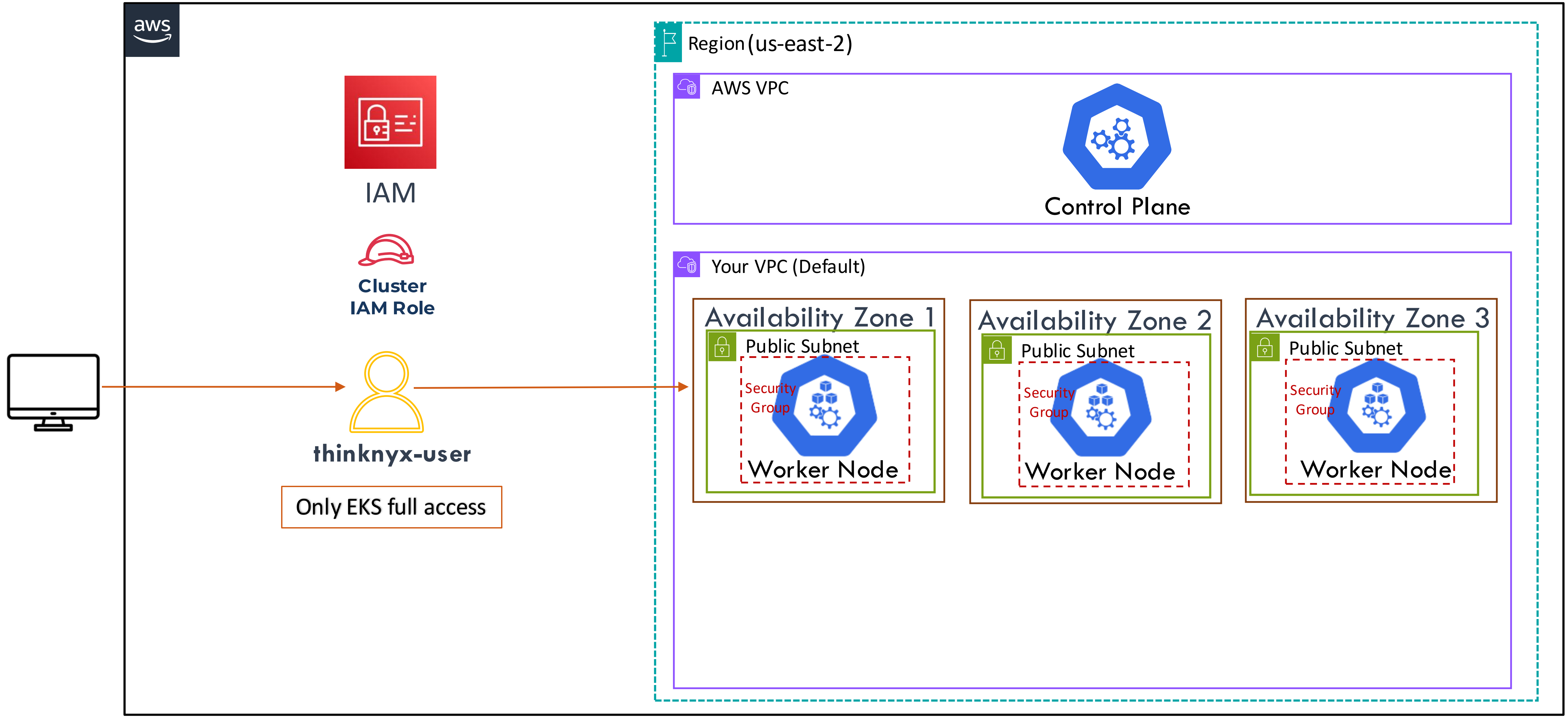
EKS Architecture – Cluster Creation Flow

Command Line Tools

- Used to generate the kubeconfig file
- **Kubeconfig includes:**
  - API server endpoint
  - Authentication details
  - Certificates for secure access

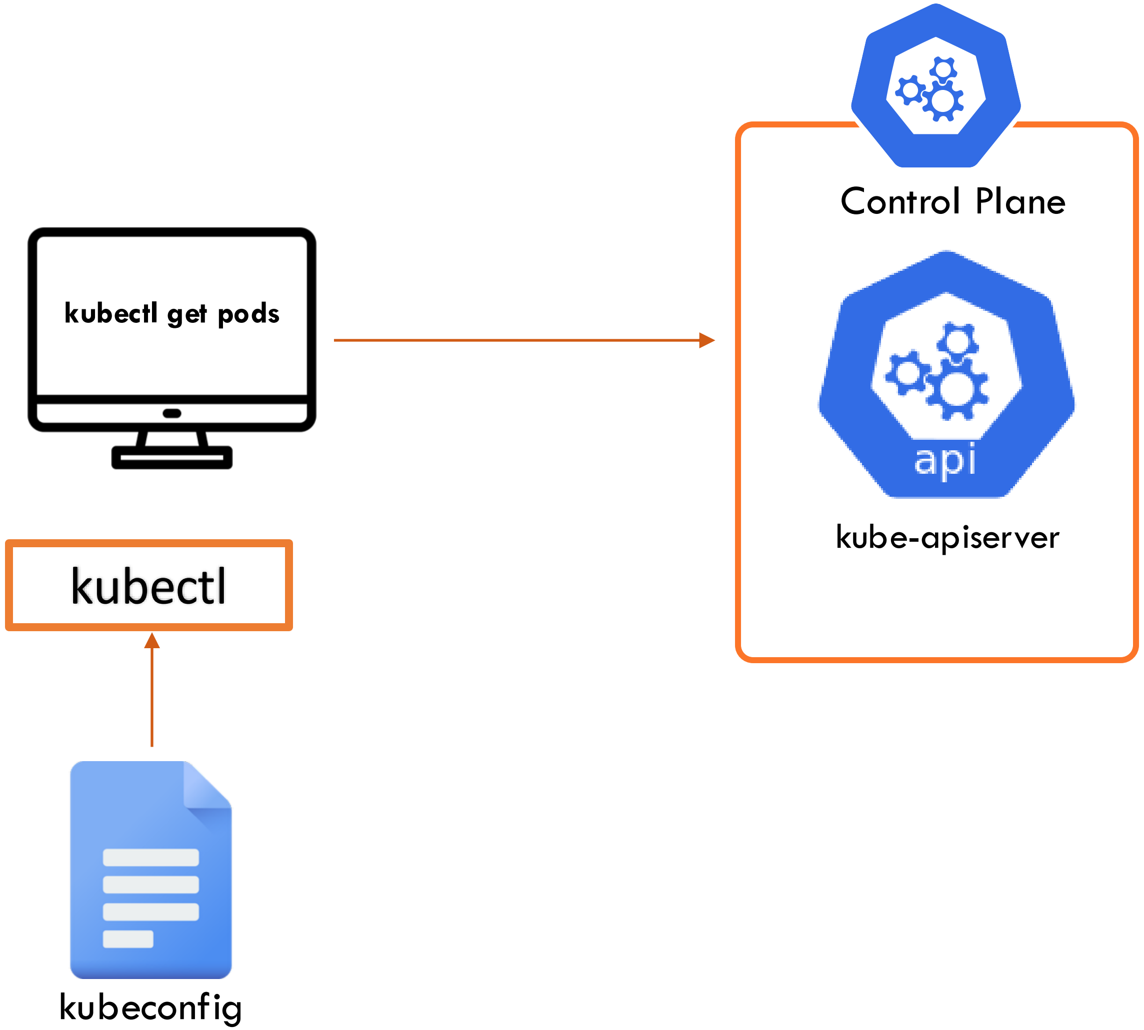
Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow



Copyright © Thinknyx Technologies LLP

Connecting to EKS



Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

Only EKS full access

Kubeconfig file is required for kubectl to communicate with EKS

Contains cluster **endpoint** and **user credentials**

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

Configure observability

- EKS allows configuration of observability during cluster setup
- Metrics collection can be enabled
- Control plane logging is available
- Helps monitor cluster performance
- Useful for troubleshooting issues

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

Authentication

- Once the EKS cluster is running, access control is crucial
- Authentication is managed using **AWS IAM**
- EKS uses the **IAM identity** to verify user access
- We are using the **thinkyx-user** IAM user for cluster access
- An **IAM access entry** is created for **thinkyx-user**
- This entry authorizes the user to connect using tools like **kubectl**

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- API Server Endpoint and OIDC Provider URL

Kubernetes API server endpoint	OIDC provider URL

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

API Server Endpoint and OIDC Provider URL

Kubernetes API server endpoint

- The API server endpoint is the access point for your EKS cluster
- kubectl uses this endpoint to communicate with the cluster
- All authenticated requests pass through this endpoint
- Examples include listing pods or deploying applications
- It acts as the gateway to the control plane

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

API Server Endpoint and OIDC Provider URL

OIDC provider URL

- OIDC provider URL is used for pod authentication
- Enables Kubernetes service accounts to assume IAM roles
- Allows secure access to AWS services from pods
- Eliminates the need to store credentials inside containers

Copyright © Thinknyx Technologies LLP

EKS Architecture – Cluster Creation Flow

- API Server Endpoint and OIDC Provider URL

Kubernetes API server endpoint	OIDC provider URL

Created automatically by Amazon EKS

Manage secure access for users and workloads

Copyright © Thinknyx Technologies LLP

Demo

Creating IAM Cluster Role

Copyright © Thinknyx Technologies LLP

Demo

Creating IAM NodeGroup Role

Copyright © Thinknyx Technologies LLP

Demo

Installing CLI Tools and Setting Up EKS Access

Copyright © Thinknyx Technologies LLP

Getting Started with Amazon EKS

Section Summary

- → *Understood the foundation of Amazon EKS Architecture*
- → *Created IAM role for secure EKS control plane access to AWS resources*
- → *Created IAM role for node groups to enable EC2 access and workload execution*
- → *Installed CLI tools (kubectl & AWS CLI) for EKS cluster interaction*

Copyright © Thinknyx Technologies LLP

Creating an EKS Cluster

Copyright © Thinknyx Technologies LLP

Creating an EKS Cluster

Section Overview

- → *Creating a cluster using the AWS Console*
- → *Exploring the EKS control plane*
- → *Navigating the EKS user interface*
- → *Upgrading the cluster version*
- → *Accessing the cluster using kubectl and AWS CLI*

Copyright © Thinknyx Technologies LLP

Demo

Create EKS Cluster via AWS Console & Explore Control Plane

Copyright © Thinknyx Technologies LLP

Demo

EKS UI Walkthrough

Copyright © Thinknyx Technologies LLP

Demo

Upgrading the EKS Cluster Version

Copyright © Thinknyx Technologies LLP

Demo

Access EKS Using kubectl and AWS CLI

Copyright © Thinknyx Technologies LLP

Creating an EKS Cluster

Section Summary

- → Created an EKS cluster from the AWS console
- → Explored control plane components
- → Toured the EKS console UI
- → Upgraded the cluster version
- → Connected to the cluster using kubectl and AWS CLI

Copyright © Thinknyx Technologies LLP

Worker Nodes and Compute Options in EKS

Copyright © Thinknyx Technologies LLP

Worker Nodes and Compute Options in EKS

Section Overview

- Understand how EKS runs workloads using Managed Node Groups and Fargate Profiles
- Quick comparison of Managed Node Groups and Fargate Profiles
- Hands-on Demonstrations:
  - Creating a Node Group
  - Setting up Fargate
  - Deploy pods on both Node Groups and Fargate
  - Enable logging for Fargate pods using the aws-logging ConfigMap

Copyright © Thinknyx Technologies LLP

Comparing Managed Node Groups and Fargate Profiles

Copyright © Thinknyx Technologies LLP

Comparing Managed Node Groups and Fargate Profiles

- Run apps by creating worker nodes
- EKS offers different ways to manage worker nodes
  - Standard Mode
  - Auto Mode
  - Fargate Mode
- We'll focus on:
  - Standard Mode with Node Groups
  - Fargate Mode
- Node Groups vs. Fargate Mode

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

➢ Compute & Scaling

Aspect	Node Groups	Fargate
Compute Type	Launches EC2 instances called Node Groups	No EC2 instances managed directly
Pod Placement	Multiple pods can run on the same instance	Each pod gets a new lightweight VM (a node)
Scaling Behaviour	Nodes are added when needed, based on scaling rules you define	AWS creates a node just for the pod when scheduled

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

➢ Infrastructure & Responsibility

Aspect	Node Groups	Fargate
Control	Full control over instance type, updates, patching, and security	AWS manages everything below the pod
Flexibility vs Simplicity	More flexibility, but more to manage	Focus purely on application needs like CPU and memory
Use Case Suitability	Good for teams comfortable managing infrastructure	Great for teams focusing only on applications

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

- Workload Compatibility

Aspect	Node Groups	Fargate
Suitability	Suited for complex or demanding workloads	Better for stateless apps, microservices, or batch jobs
Feature Support	DaemonSets, custom AMIs, EBS volumes, GPU nodes	EFS supported; no EBS or GPU; no DaemonSets

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

➢ Security & IAM Roles

Aspect	Node Groups	Fargate
IAM Responsibility	You manage EC2 security and IAM roles	AWS handles host security
IAM Role Type	Each node uses a **Node Instance Role** for cluster connection and AWS access	Uses a **Pod Execution Role** for defining pod-level permissions

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

➢ Cost Model

Aspect	Node Groups	Fargate
Pricing Model	Charged for entire EC2 instance, regardless of utilization	Pay-per-pod based on CPU and memory
Cost Efficiency	Cost-effective for long-running, high-density workloads	Ideal for short-lived, bursty workloads
EKS Charges	No extra cost for managed node groups	Charges are strictly based on resources defined in the pod spec
Billing Unit	Based on EC2 instance costs + related services	Billed per second (1-minute minimum)

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

➢ Networking & Subnets

Aspect	Node Groups	Fargate
Network Inheritance	Pods inherit EC2 instance network settings	Only supports private subnets
Public IPs	Pods can get public IPs if nodes are in public subnets	Pods do not receive public IPs
Outbound Internet Access	In private subnets, outbound traffic must go through NAT Gateway	All outbound traffic must go through NAT Gateway (no direct Internet Gateway routing allowed)

Copyright © Thinknyx Technologies LLP

Node Groups vs Fargate Profiles

So, When Should You Use What?

Node Groups

- Need more control and customization
- Run custom workloads, GPUs, DaemonSets, or use EBS volumes
- Want flexibility despite managing infrastructure

For heavy/specialized workloads

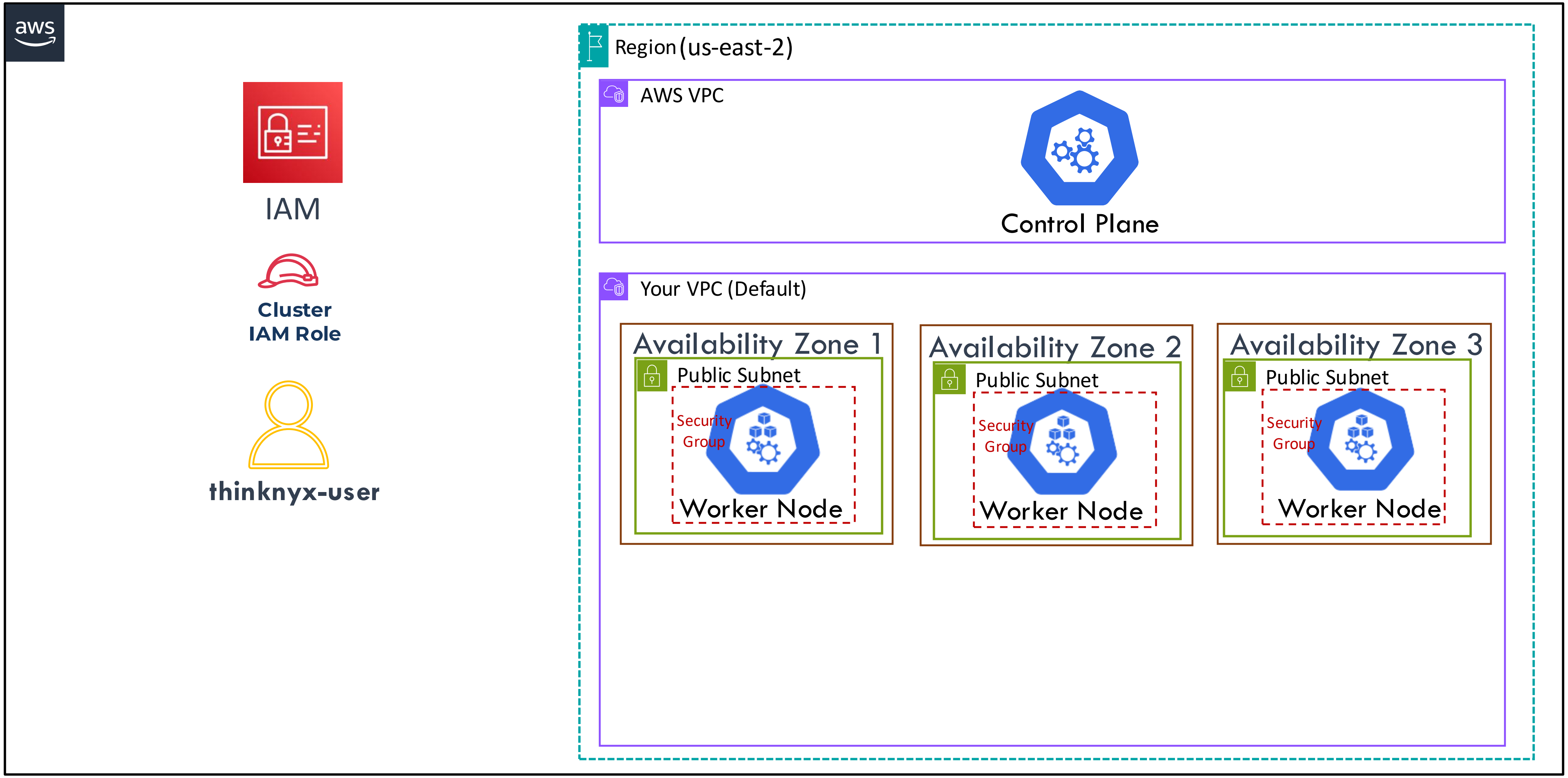
Fargate

- Prioritize simplicity and speed
- Don’t want to manage EC2 instances
- Run stateless or short-lived apps

On-demand scaling and lightweight services

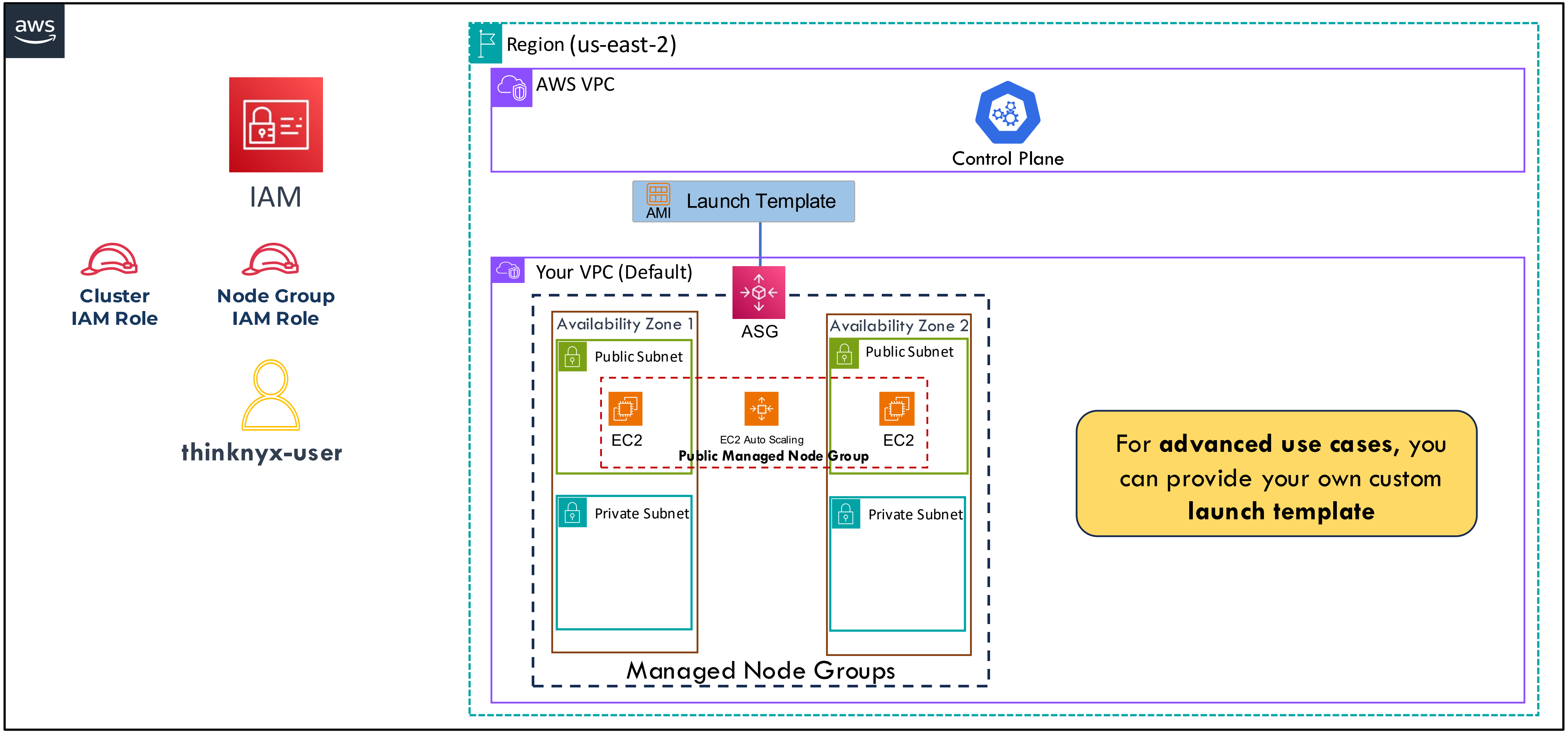
Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next



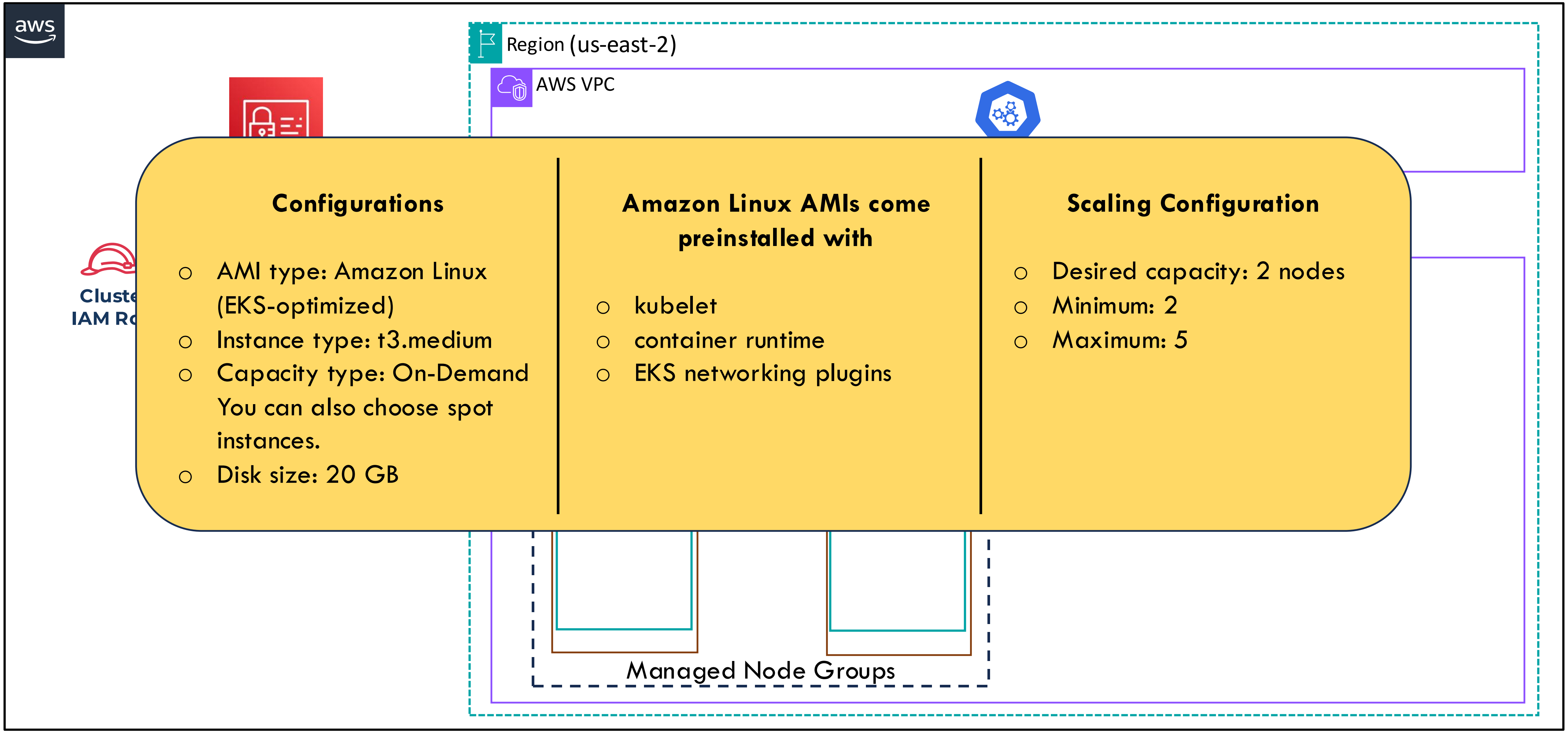
Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next



Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next



Configurations

- AMI type: Amazon Linux (EKS-optimized)
- Instance type: t3.medium
- Capacity type: On-Demand You can also choose spot instances.
- Disk size: 20 GB

Amazon Linux AMIs come preinstalled with

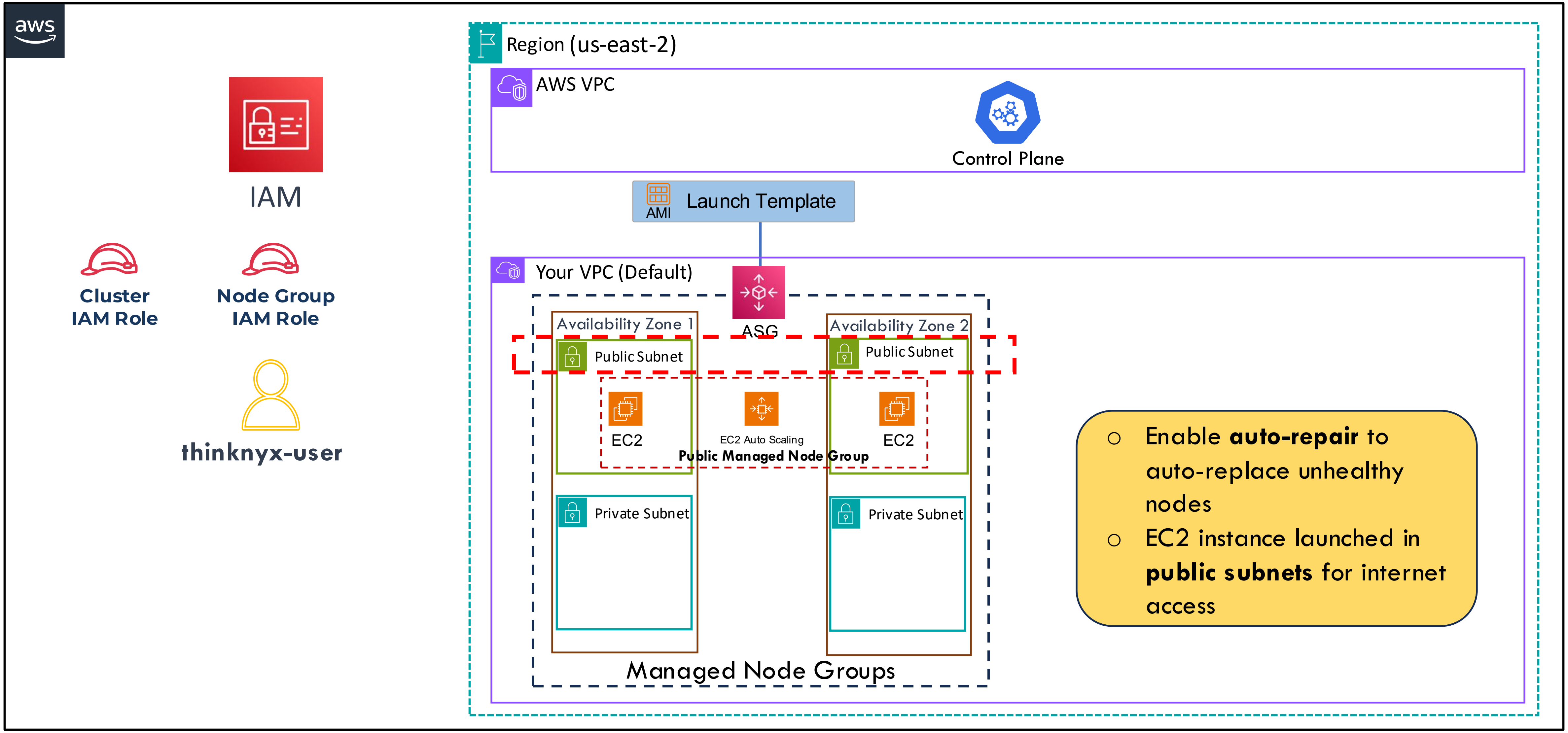
- kubelet
- container runtime
- EKS networking plugins

Scaling Configuration

- Desired capacity: 2 nodes
- Minimum: 2
- Maximum: 5

Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next



Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next

AWS Fargate Configuration

- Fargate removes the need to manage EC2 infrastructure by auto-provisioning compute for scheduled pods
- To use Fargate, a **Fargate profile** must be defined

Copyright © Thinknyx Technologies LLP

EKS Architecture: What We’ll Build Next

How a Fargate Profile Works

- A Fargate profile tells EKS which pods should be scheduled on Fargate
- **Selectors** which include:
  - Namespace (required)
  - Labels (optional)
- We willl create a Fargate profile targeting:
  - Namespace: demo-ns
  - Label: type=fargate
- Only pods in demo-ns with label type=fargate run on Fargate

Copyright © Thinknyx Technologies LLP

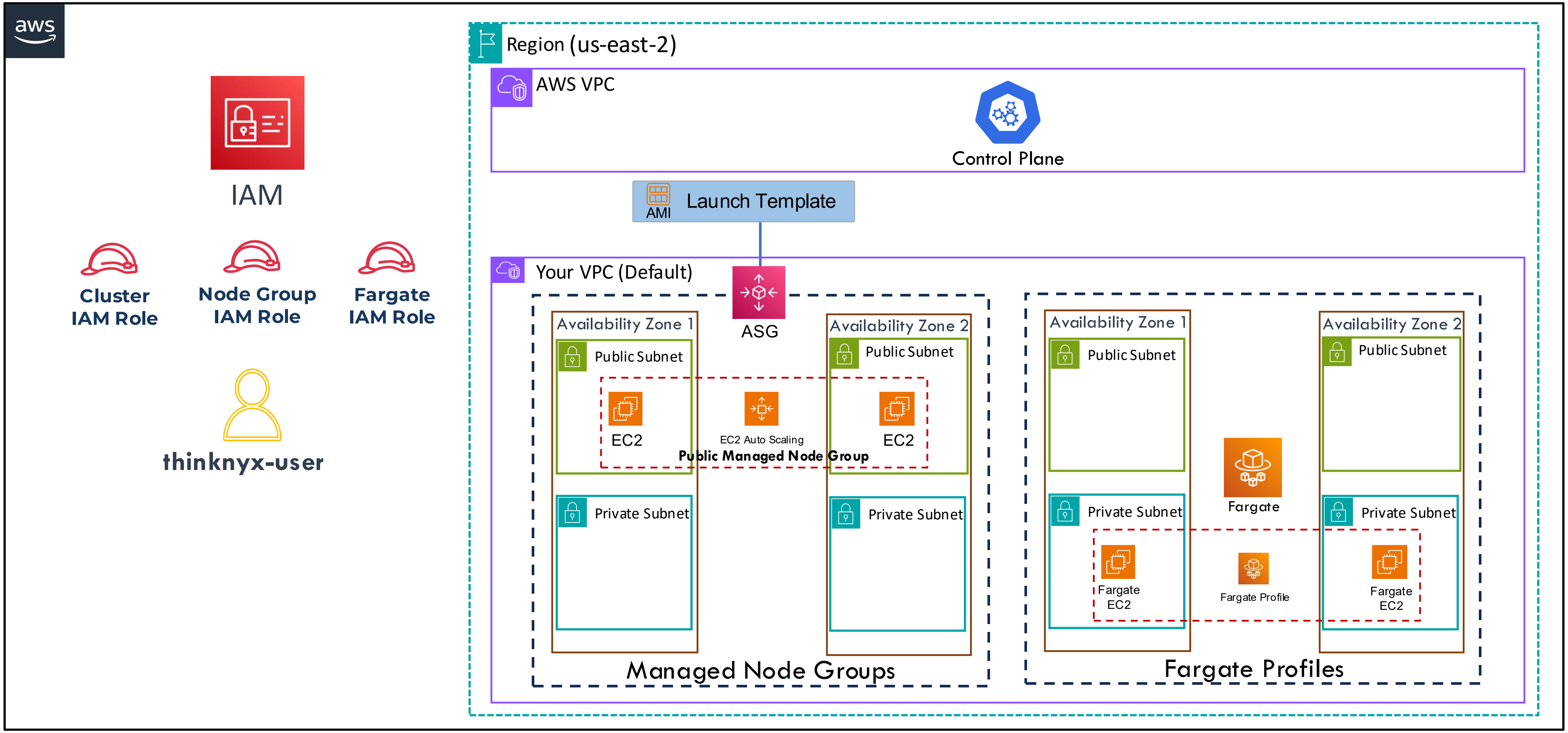
EKS Architecture: What We'll Build Next

➢ How a Fargate Profile Works

- Each Fargate profile can have up to 5 selectors
- If only a namespace is set, all its pods run on Fargate
- **Note:** Fargate profiles **cannot** be edited after creation

Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next



Copyright © Thinknyx Technologies LLP

EKS Architecture: What We'll Build Next

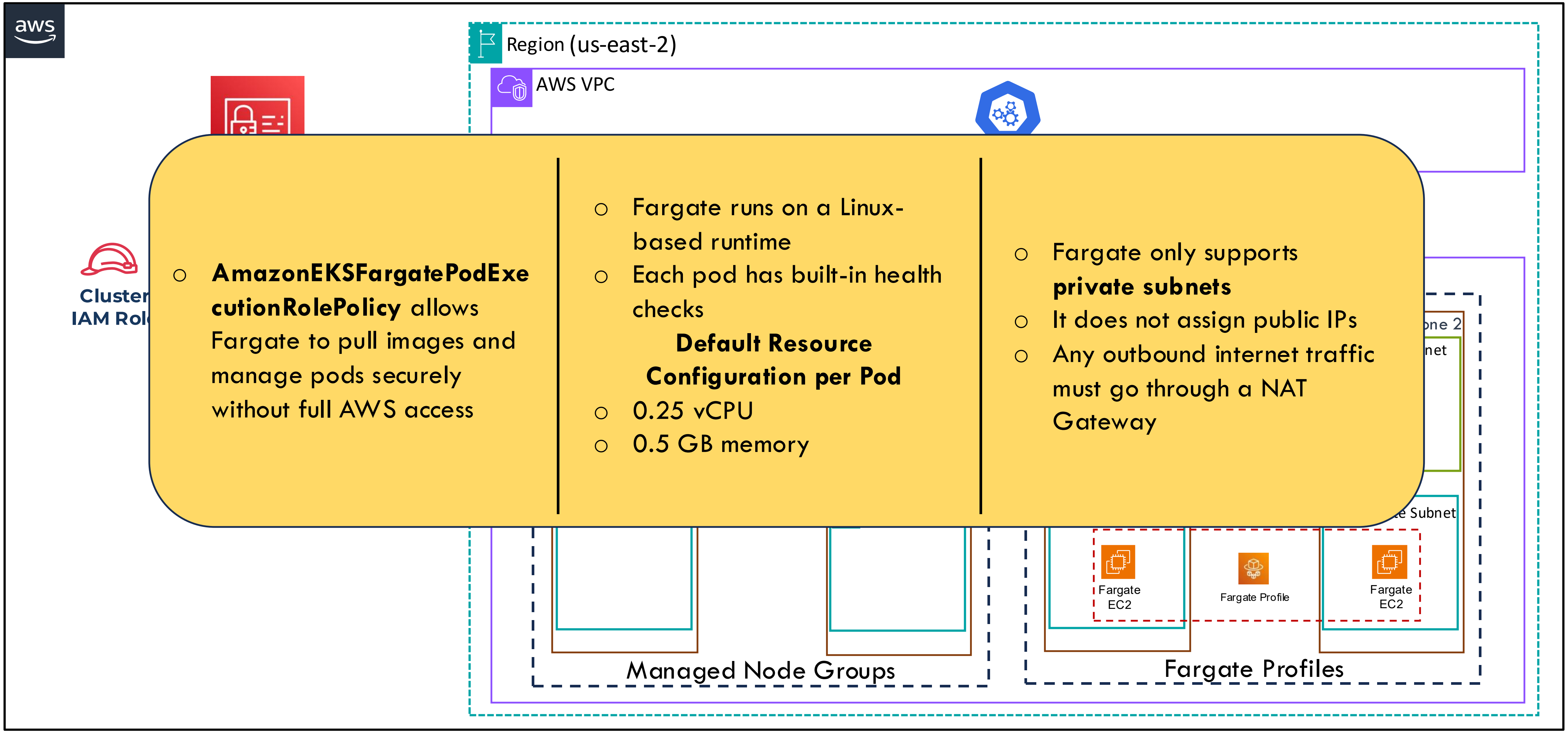
- **AmazonEKSFargatePodExecutionRolePolicy** allows Fargate to pull images and manage pods securely without full AWS access

- Fargate runs on a Linux-based runtime
- Each pod has built-in health checks

Default Resource Configuration per Pod

- 0.25 vCPU
- 0.5 GB memory

- Fargate only supports **private subnets**
- It does not assign public IPs
- Any outbound internet traffic must go through a NAT Gateway



Copyright © Thinknyx Technologies LLP

Demo

Creating Managed Node Groups

Copyright © Thinknyx Technologies LLP

Demo

Creating IAM Role for Fargate Profile

Copyright © Thinknyx Technologies LLP

Demo

Adding a Fargate Profile to EKS

Copyright © Thinknyx Technologies LLP

Demo

Deploying a Pod to Fargate

Demo

Adding aws-logging configmap

Copyright © Thinknyx Technologies LLP

Getting Started with Amazon EKS

Section Summary

- *Explored Managed Node Groups and Fargate*
- *Learnt how to:*
    - *Create a Node Group for EC2 worker nodes*
    - *Set up a Fargate Profile for serverless pod execution*
    - *Deploy workloads on EC2 and Fargate*
    - *Enable Fargate pod logging via aws-logging ConfigMap*

Copyright © Thinknyx Technologies LLP

Exposing Applications with Load Balancers in EKS

Copyright © Thinknyx Technologies LLP

Exposing Applications with Load Balancers in EKS

Section Overview

- → *Exposing applications in EKS using LoadBalancer service type*
- → *Deploying application with default LoadBalancer (Classic Load Balancer)*
- → *Using annotations to create a Network Load Balancer (NLB)*
- → *Cleaning up the created resources*

Copyright © Thinknyx Technologies LLP

Understanding Load Balancing in EKS

Copyright © Thinknyx Technologies LLP

Understanding Load Balancing in EKS

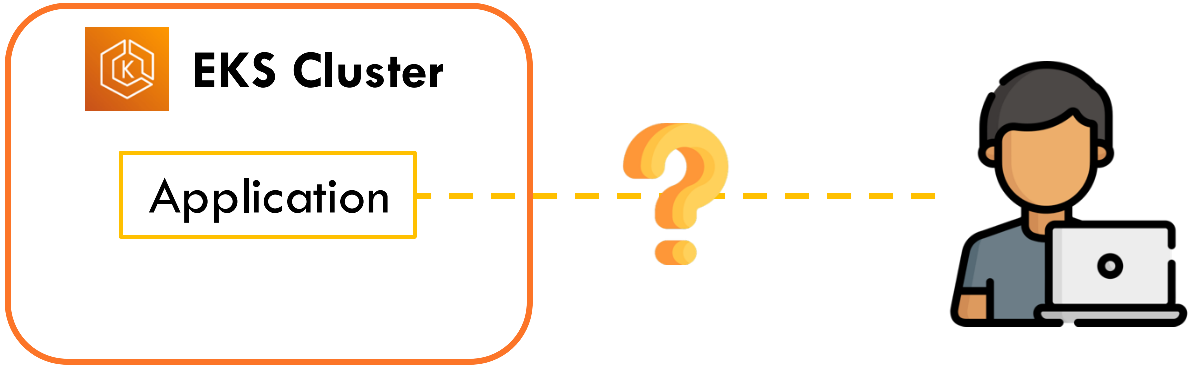
EKS Cluster

Managed Node Groups

Fargate

Copyright © Thinknyx Technologies LLP

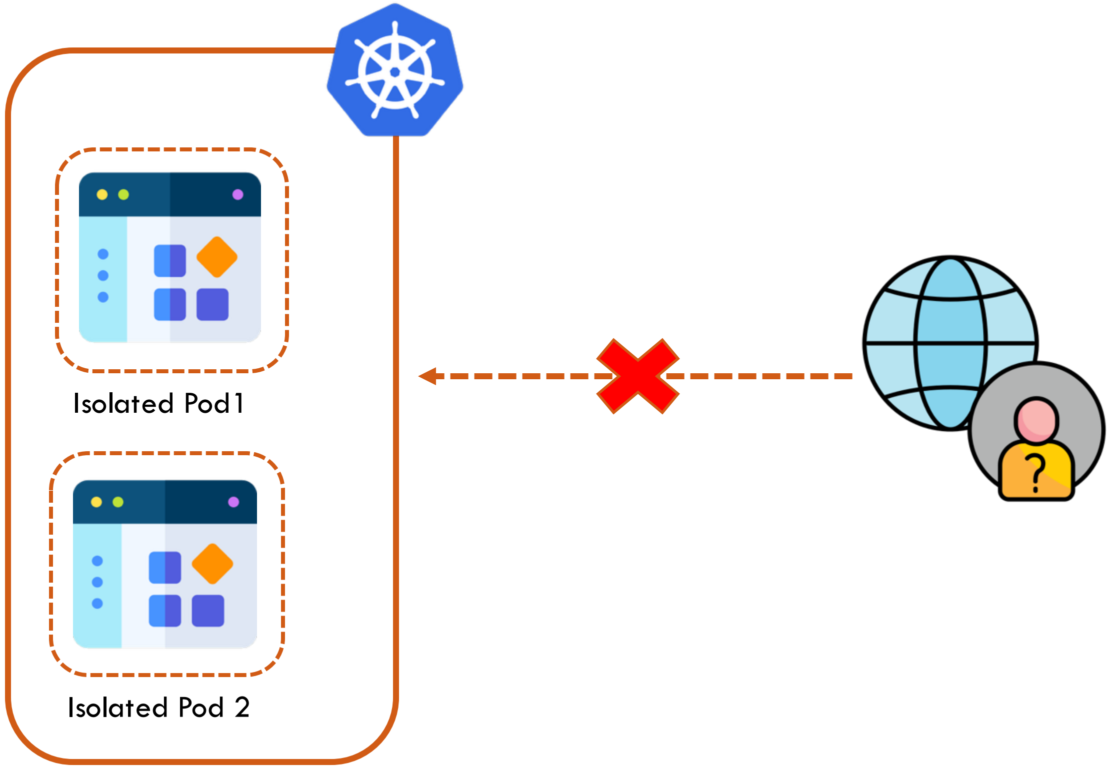
Understanding Load Balancing in EKS



LOAD BALANCER

Copyright © Thinknyx Technologies LLP

Why Load Balancers Are Needed



Copyright © Thinknyx Technologies LLP

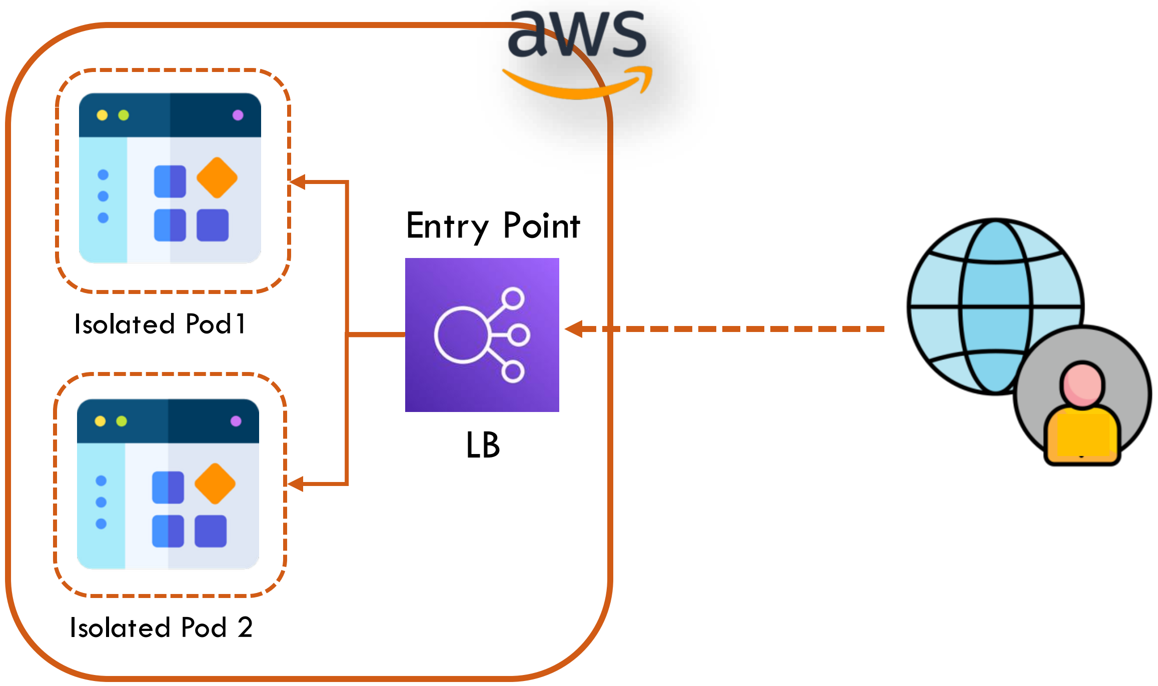
Why Load Balancers Are Needed

To expose them, we use **Kubernetes Services**

- **LoadBalancer** enables external access
- **Kubernetes** requests the creation of an external Load Balancer

Copyright © Thinknyx Technologies LLP

Why Load Balancers Are Needed



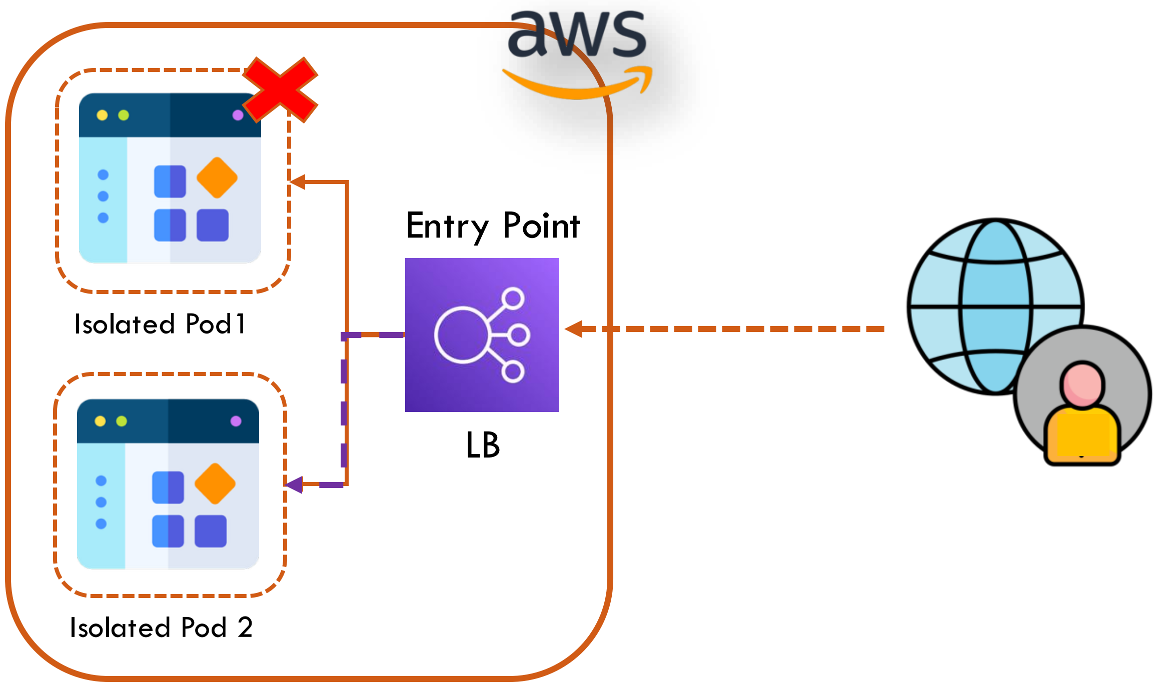
Allows Internet Access

Improves Availability

Improves Resilience

Copyright © Thinknyx Technologies LLP

Why Load Balancers Are Needed



Allows Internet Access

Improves Availability

Improves Resilience

Copyright © Thinknyx Technologies LLP

What Happens Behind the Scenes

Classic Load Balancer

Maintenance Mode

Receiving only critical bug fixes

What's the better alternative?

NLB

ALB

Copyright © Thinknyx Technologies LLP

What Happens Behind the Scenes

NLB

- Creating a Service of type **LoadBalancer** provisions a Network Load Balancer (NLB)
- NLB operates at **Layer 4** (Transport Layer)
- Suitable for **TCP or UDP**-based applications

Copyright © Thinknyx Technologies LLP

What Happens Behind the Scenes

ALB

- Defining an **Ingress** provisions an Application Load Balancer (ALB)
- ALB operates at **Layer 7** (Application Layer)
- Suitable for **HTTP/HTTPS** web applications
- Supports **path-based** and **host-based** routing

Copyright © Thinknyx Technologies LLP

How to Specify the Type of Load Balancer

- Annotations
  - Use annotations in Kubernetes YAML manifests
  - Annotations instruct AWS on infrastructure provisioning

Copyright © Thinknyx Technologies LLP

Subnet Tagging for Load Balancer Provisioning

- AWS requires subnet information to create a Load Balancer
- Subnets must be tagged appropriately
- For a public Load Balancer, use the tag:

  kubernetes.io/cluster/<cluster-name> = owned or shared

- This tag signals that the subnet supports internet-facing traffic
- It also indicates subnet association with the EKS cluster
- Load Balancer provisioning will fail without these required tags

Copyright © Thinknyx Technologies LLP

Cleanup and Health Checks

Cleanup

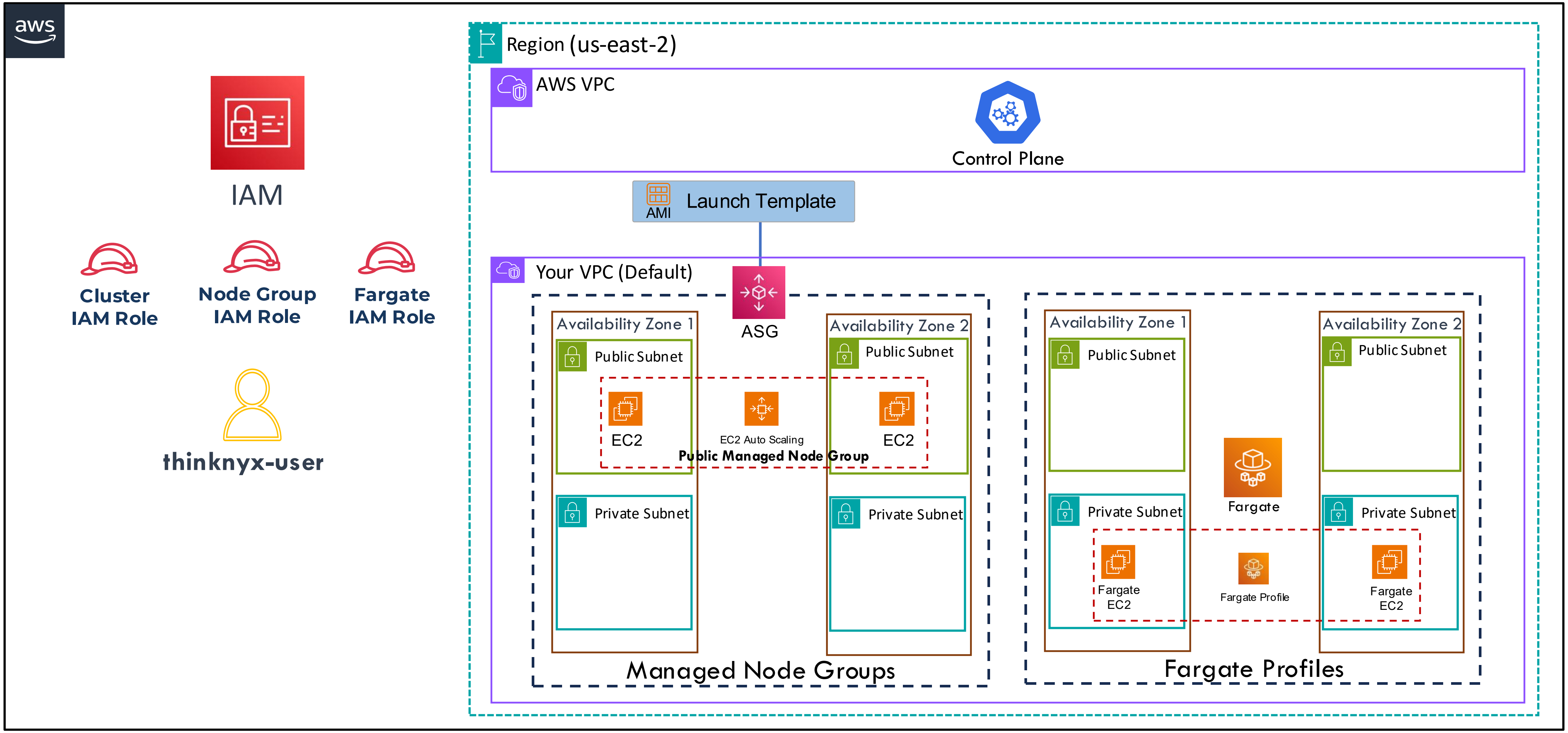
- AWS automatically deletes the associated Load Balancer
- No manual cleanup needed; AWS handles the removal of associated infrastructure

Health Checks

- AWS ELB health checks provide an additional layer beyond Kubernetes health checks
- Ensures traffic is only sent to healthy pods
- Useful when Kubernetes hasn't yet detected a pod issue
- ELB health checks are configurable

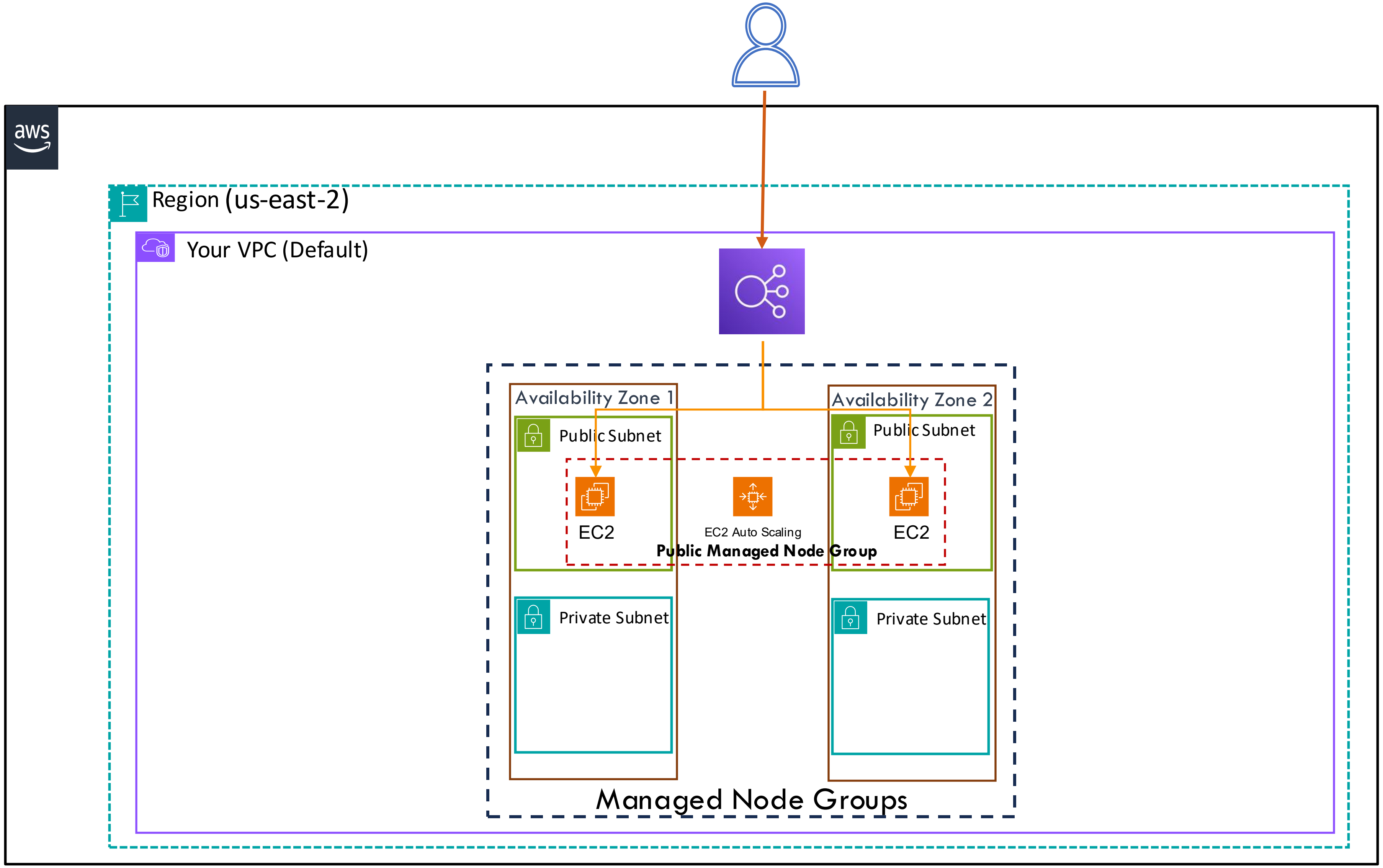
Copyright © Thinknyx Technologies LLP

What We're About to Do in the Demo



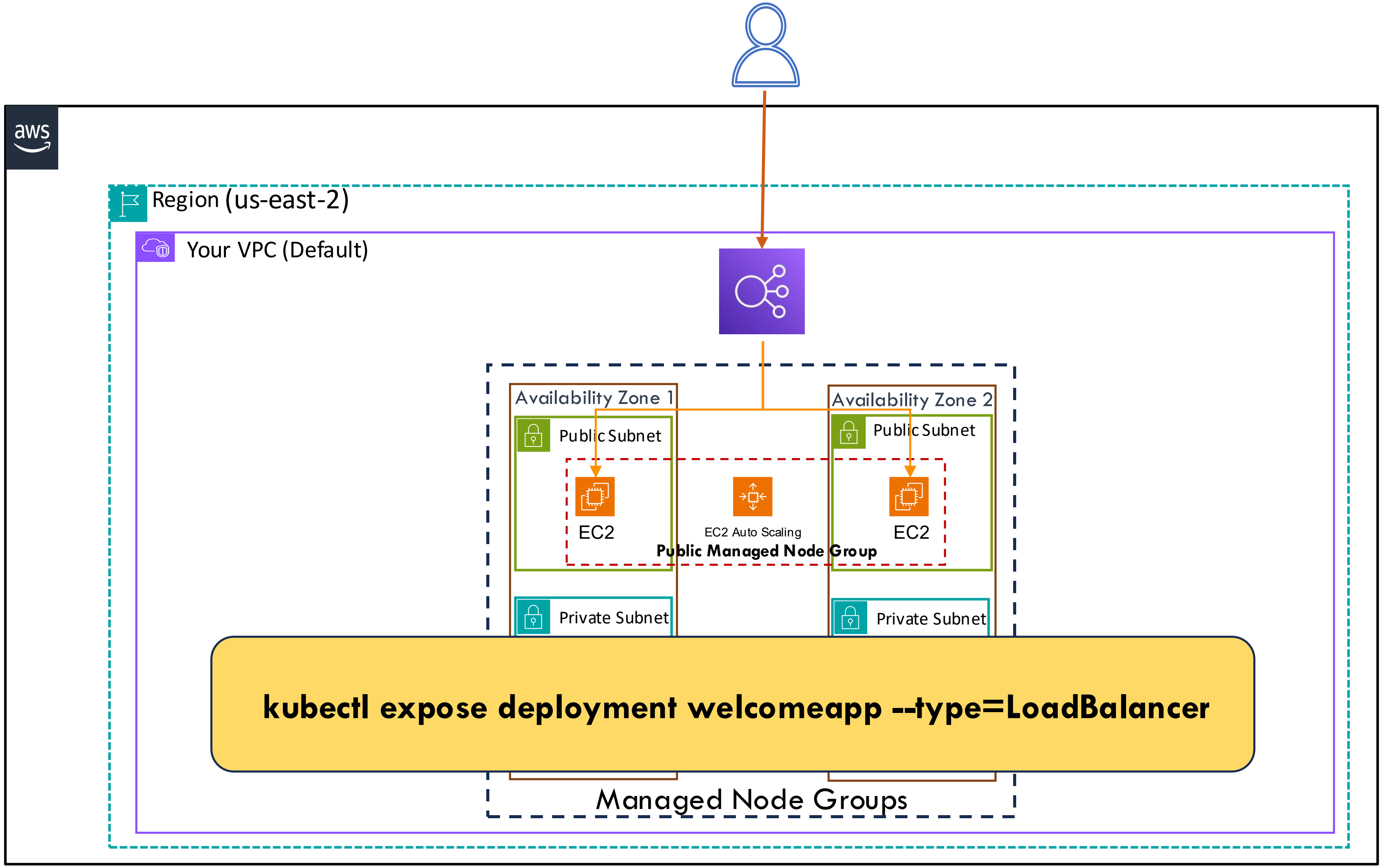
Copyright © Thinknyx Technologies LLP

What We're About to Do in the Demo



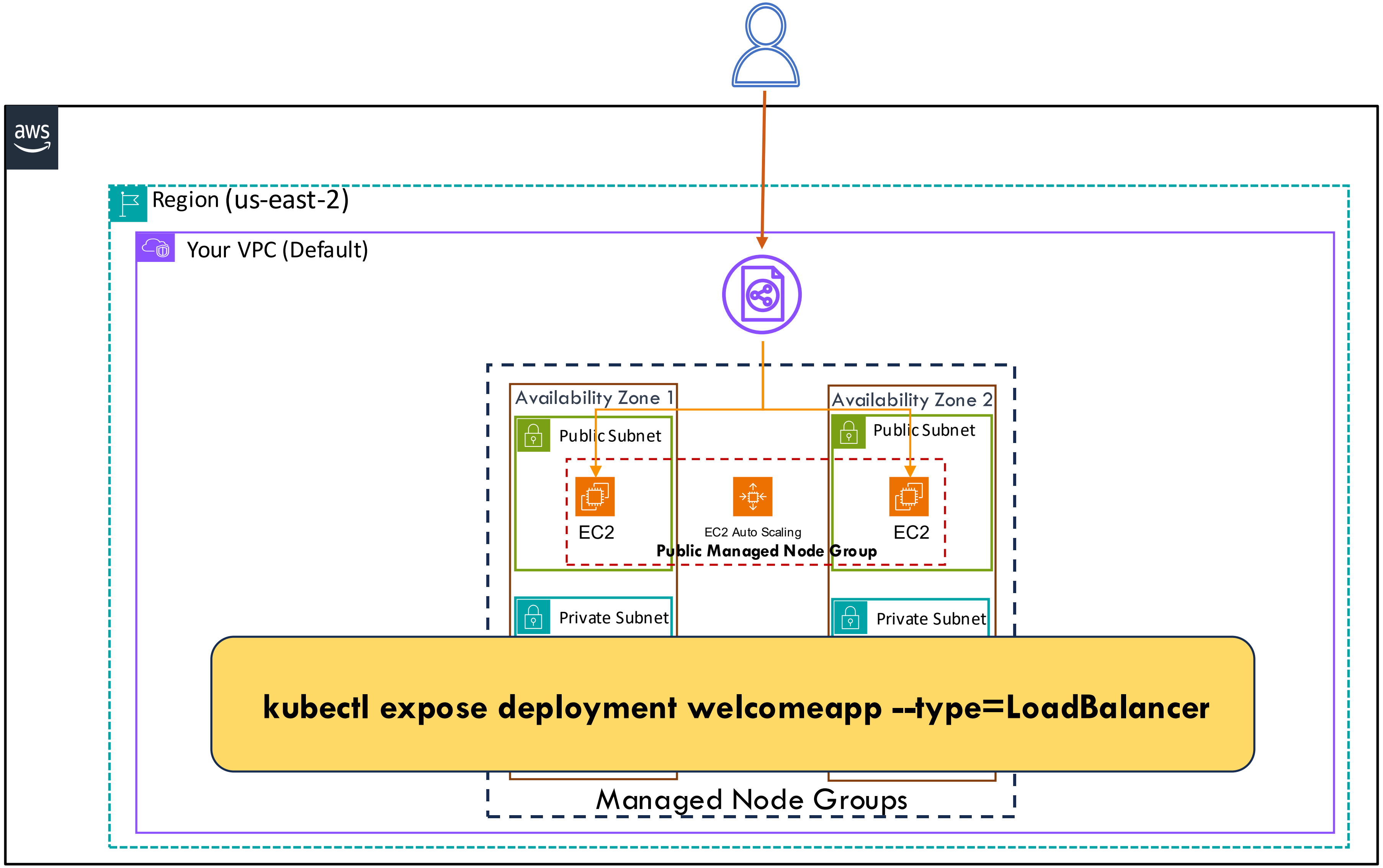
Copyright © Thinknyx Technologies LLP

What We're About to Do in the Demo



Copyright © Thinknyx Technologies LLP

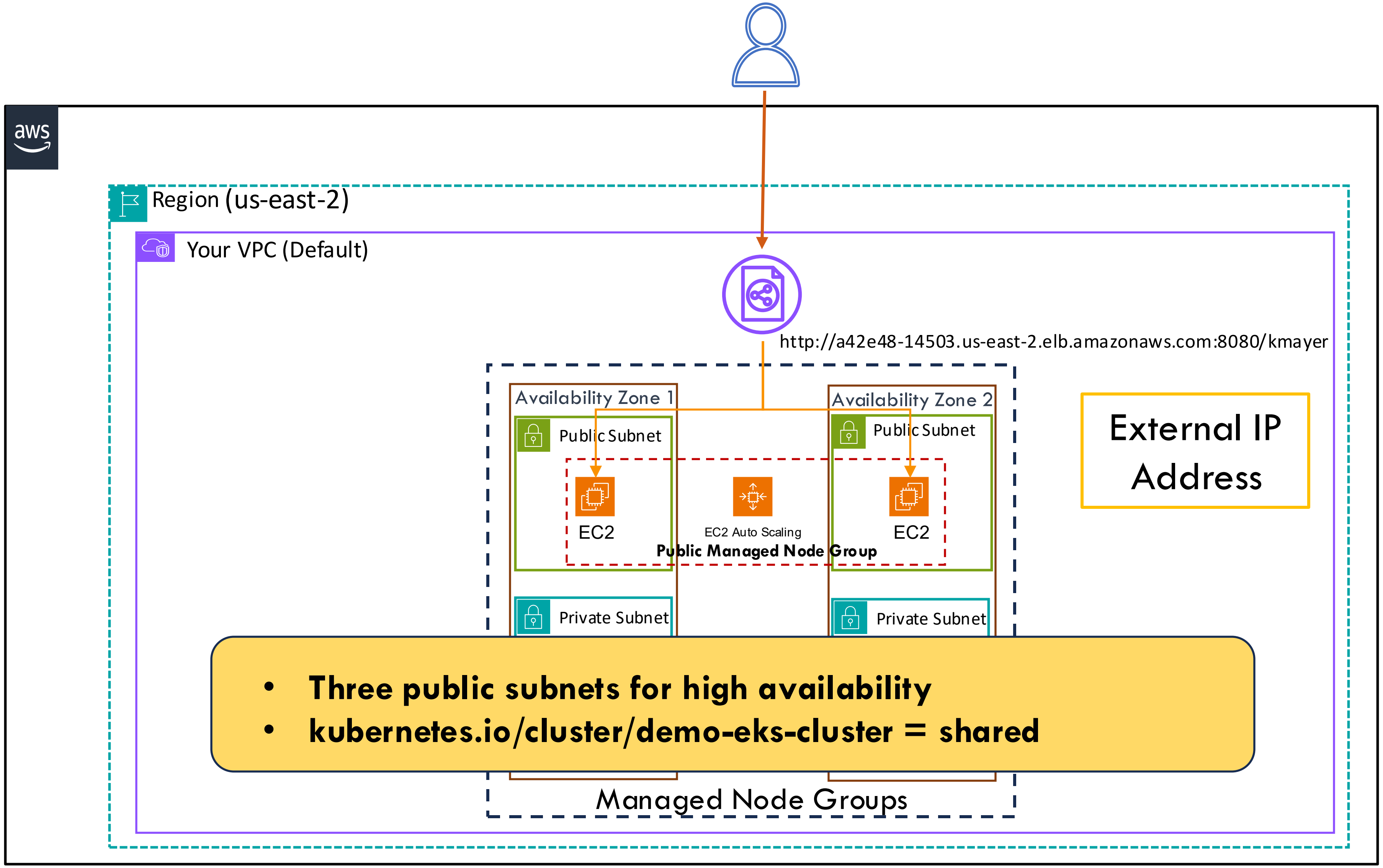
What We're About to Do in the Demo



kubectl expose deployment welcomeapp --type=LoadBalancer

Copyright © Thinknyx Technologies LLP

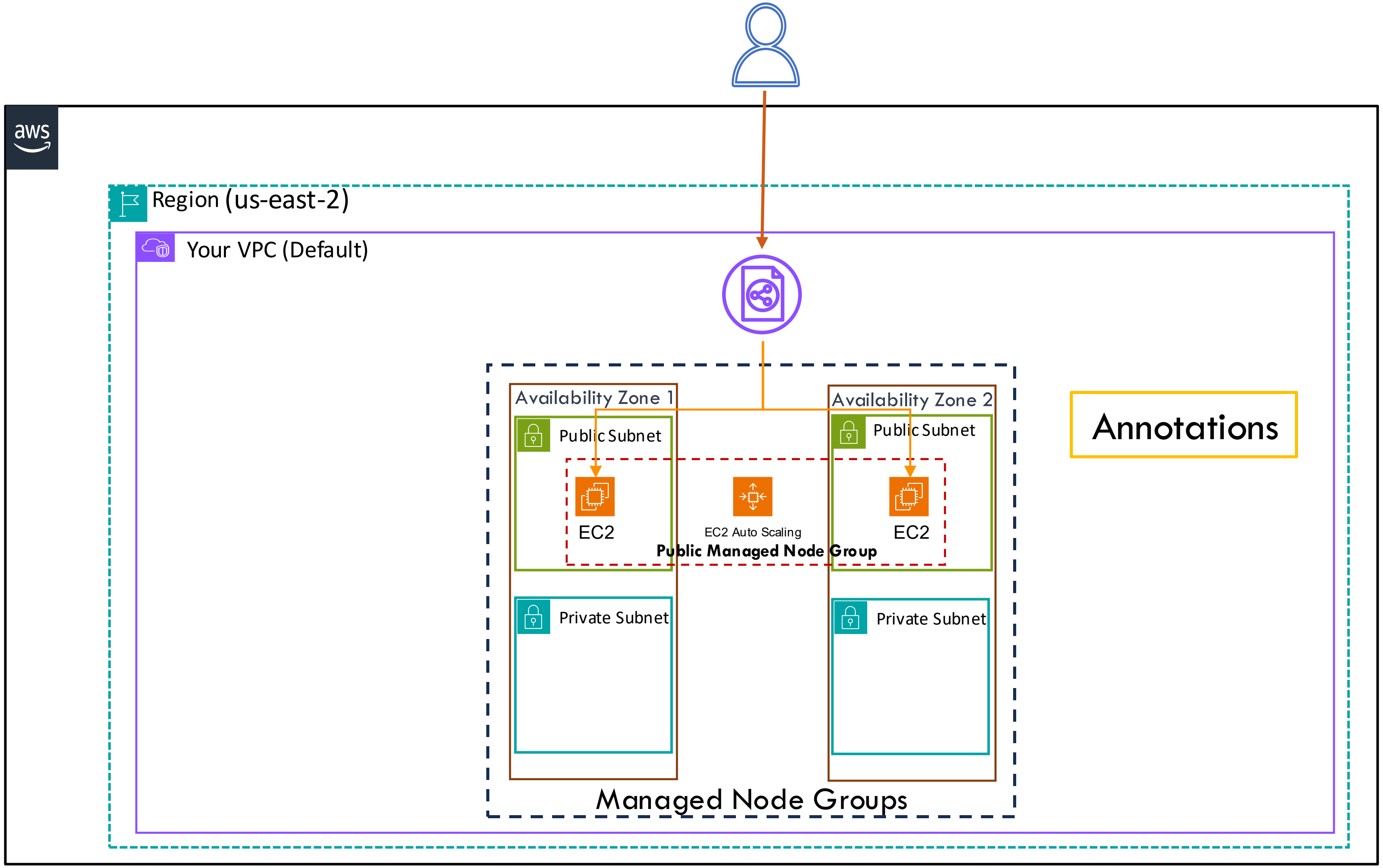
What We're About to Do in the Demo



- **Three public subnets for high availability**
- **kubernetes.io/cluster/demo-eks-cluster = shared**

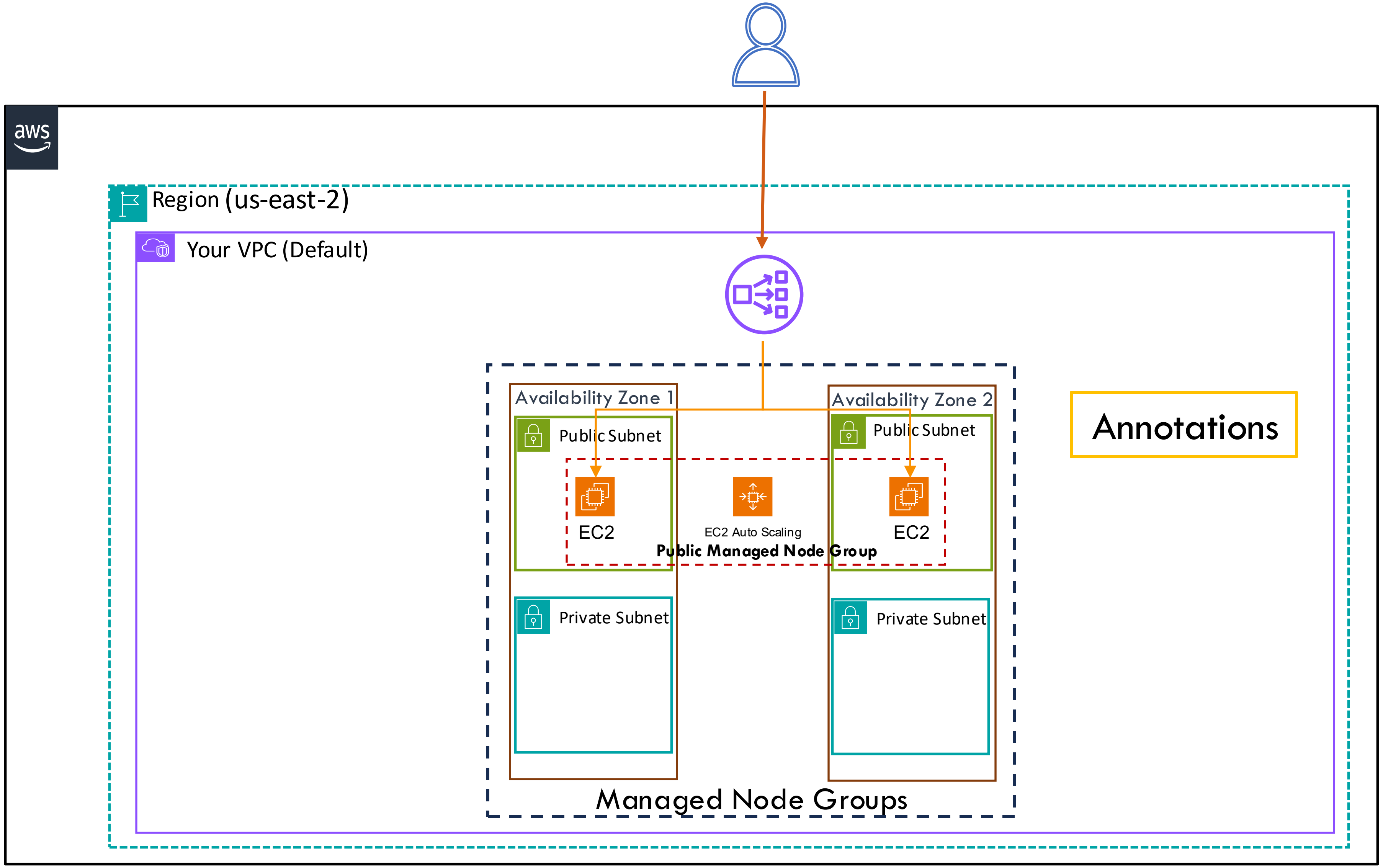
Copyright © Thinknyx Technologies LLP

What We're About to Do in the Demo



Copyright © Thinknyx Technologies LLP

What We're About to Do in the Demo

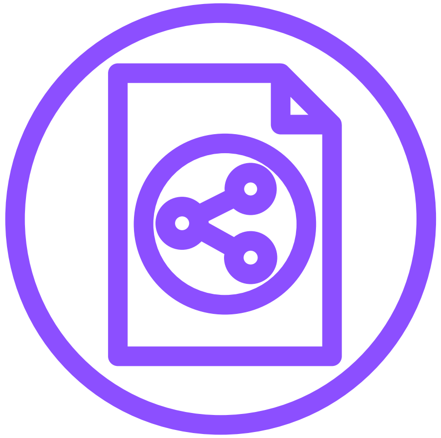


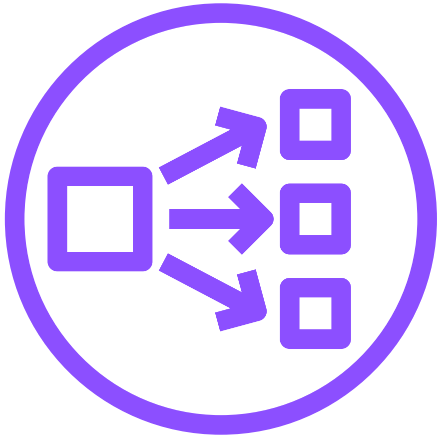
Copyright © Thinknyx Technologies LLP

What We're About to Do in the Demo

```yaml
metadata:
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
    service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Copyright © Thinknyx Technologies LLP





Copyright © Thinknyx Technologies LLP

Demo

Expose Application using ServiceType LoadBalancer

Copyright © Thinknyx Technologies LLP

Demo

Using Annotations to Create Network LB

Demo

Cleaning up Existing Resources

Copyright © Thinknyx Technologies LLP

Exposing Applications with Load Balancers in EKS

Section Summary

- → *Explored exposing applications in EKS using a LoadBalancer service type*
- → *Default setup created a Classic Load Balancer*
- → *Customized setup to use Network Load Balancer (NLB) via service annotations*
- → *Verified traffic routing to pods through the NLB*
- → *Confirmed configuration via the AWS Console*
- → *Cleaned up all resources created during the demo*

Copyright © Thinknyx Technologies LLP

Storage Options

Copyright © Thinknyx Technologies LLP

Storage Options

Section Overview

- → *Learn how to use Amazon EBS and EFS for persistent storage in EKS*
- → *Cover EBS for single-Pod storage using PVCs and StatefulSets*
- → *Explore EFS for shared storage across multiple Pods and Fargate*

Copyright © Thinknyx Technologies LLP

Overview of EBS Volumes & How it works

Copyright © Thinknyx Technologies LLP

Overview of EBS Volumes & How it works

Core Infrastructure for EKS Environment

Elastic Load Balancer

- Managed Node Groups
- Fargate Profiles

What happens when your application needs to store data?

How do you persist user information in a database, or retain uploaded files even after a Pod is terminated?

Storage

Copyright © Thinknyx Technologies LLP

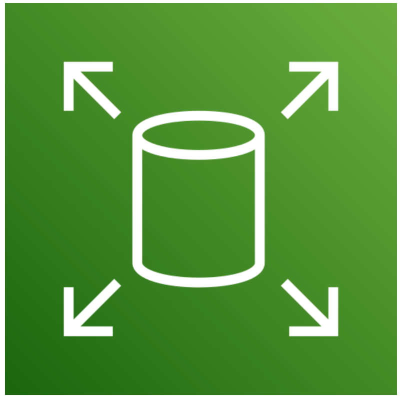
Understanding Kubernetes Storage

- Kubernetes lets you define volumes for Pods
- Volumes allow containers to access external storage
- It supports two storage types:
  - Ephemeral – data lost when Pod stops
  - Persistent – data stays even after Pod ends
- For persistent storage, Kubernetes uses Persistent Volumes (PVs)

Copyright © Thinknyx Technologies LLP

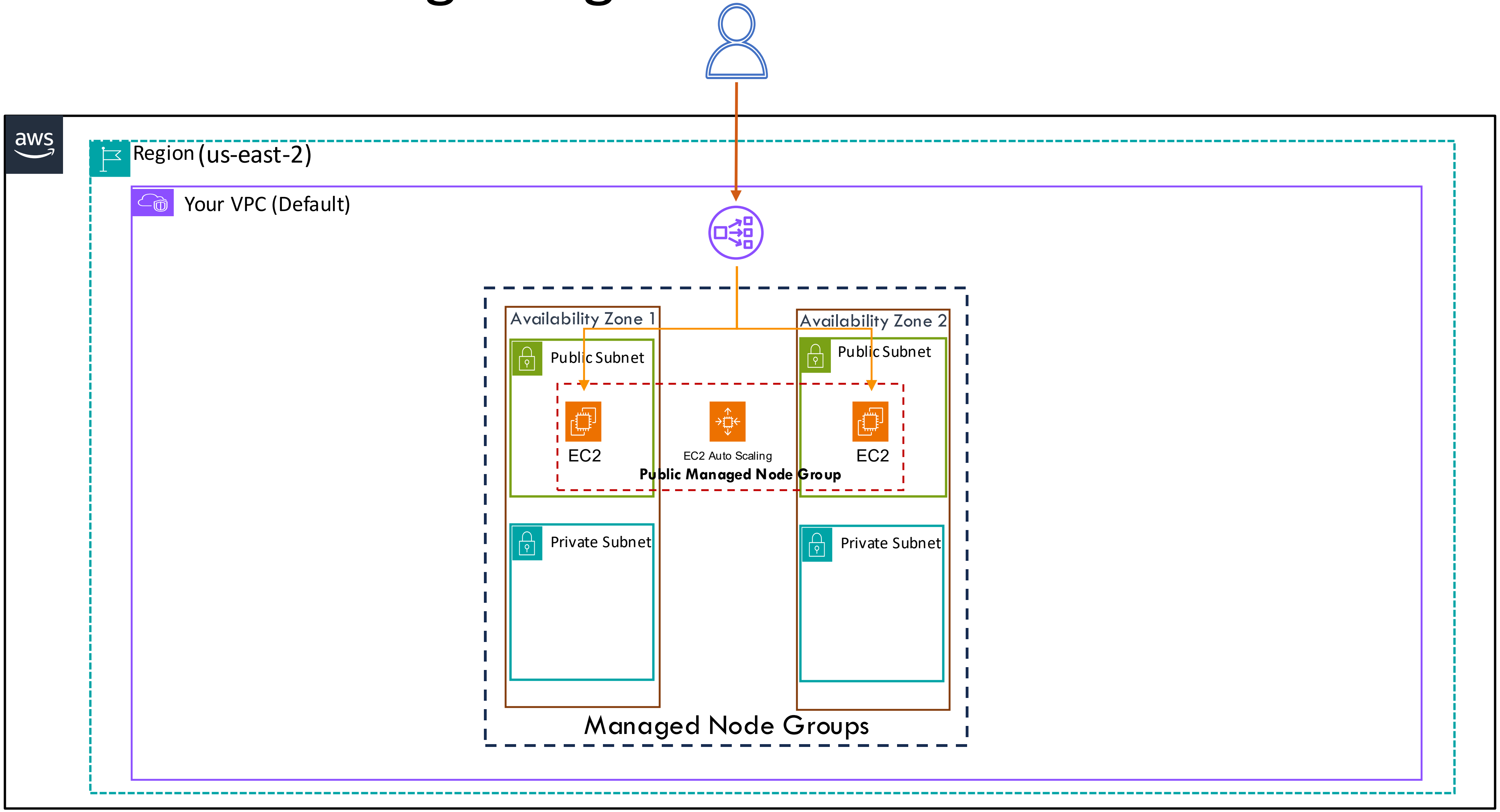
Why Amazon EBS?

- In Amazon EKS, one of the most widely used persistent storage solutions is Amazon Elastic Block Store, or EBS
- EBS provides block-level storage ideal for workloads such as:
  - Databases (MySQL & PostgreSQL)
  - High IOPS, high throughput apps
  - Random read & write operations



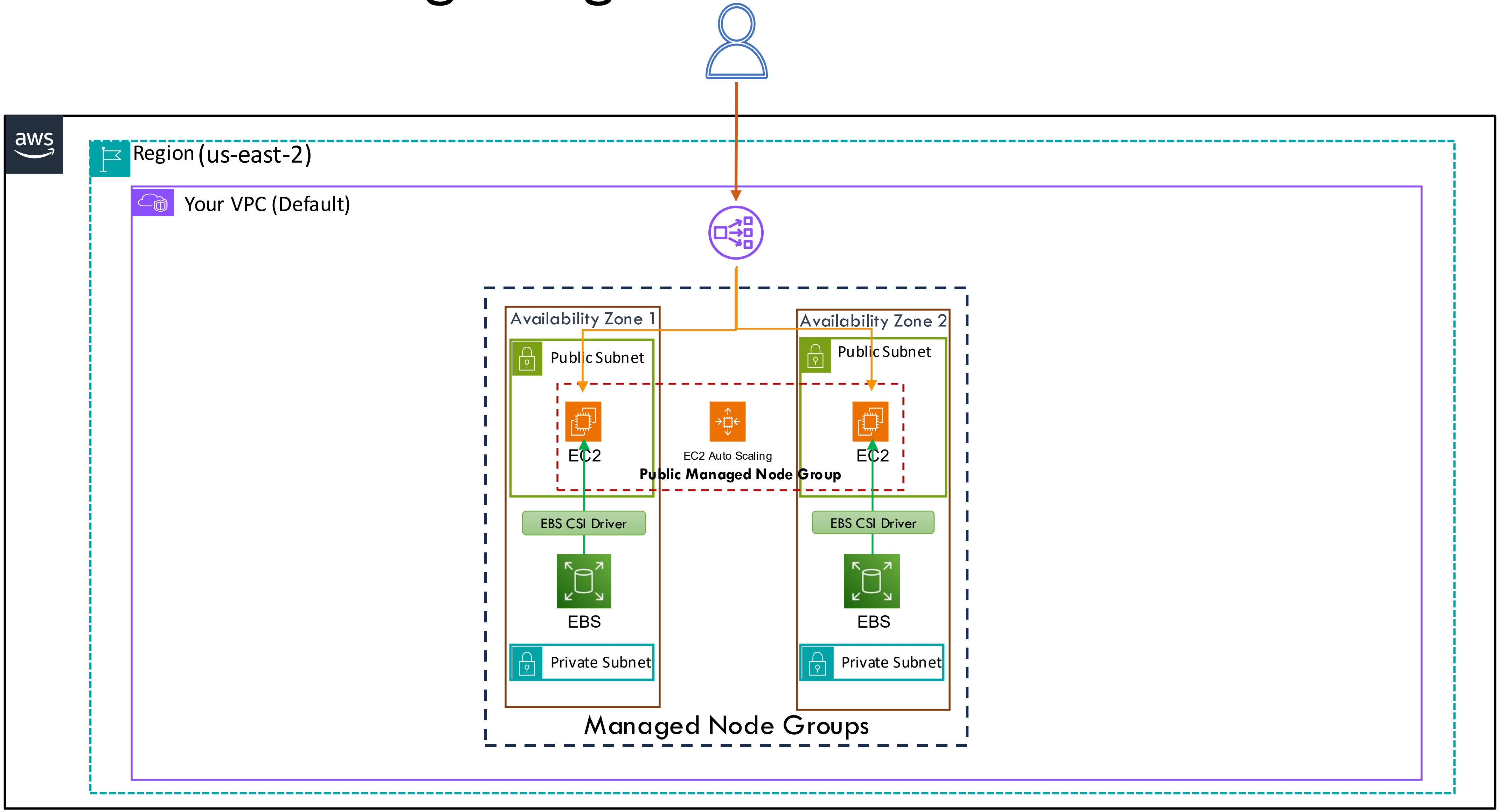
Copyright © Thinknyx Technologies LLP

Integrating EBS with Kubernetes



Copyright © Thinknyx Technologies LLP

Integrating EBS with Kubernetes



Copyright © Thinknyx Technologies LLP

Integrating EBS with Kubernetes

Copyright © Thinknyx Technologies LLP

Integrating EBS with Kubernetes

- The CSI driver acts as a bridge between Kubernetes and AWS EBS
- It allows to:
  - Dynamically provision & attach EBS volumes to Pods
  - Integrate with Kubernetes objects like PVs, PVCs, and StorageClasses
  - Automatically manages the lifecycle of storage volumes
- CSI driver is installed as an **add-on**

Copyright © Thinknyx Technologies LLP

Granting Access with IAM and IRSA

- CSI driver needs permissions to make API calls to AWS
- We create an IAM Role and attach the **AmazonEBSCSIDriverPolicy**
- For secure access, we use **IAM Roles for Service Accounts, or IRSA**
- Working:
  - Create an **IAM role** with the required **EBS permissions**
  - Define a **custom trust policy** to allow the role for a Kubernetes service account
  - Use the **OIDC identity provider**
  - **Annotate** the CSI driver's service account to link the IAM role
- This ensures **secure, least-privilege access** from Kubernetes to AWS services

Copyright © Thinknyx Technologies LLP

Stateful Workloads and StatefulSets

StatefulSets

Databases Messaging Queues Caching Systems

- Assign a stable, persistent identity to each Pod
- Create a **dedicated PersistentVolumeClaim** for each Pod using **volumeClaimTemplates**
- While Pods in a StatefulSet can fail, their **Persistent Pod Identifiers** help reattach existing volumes to replacement Pods
- Ideal when you need:
  - Stable network identity
  - Dedicated, persistent storage per Pod
  - Graceful and ordered scaling

Copyright © Thinknyx Technologies LLP

Key PVC Behavior

- Even if you delete the Pod, the associated volume and PVC remain intact.
- The default volumeBindingMode is WaitForFirstConsumer, which means the volume isn't provisioned until a Pod is scheduled.
- The accessModes for EBS-backed volumes are usually set to ReadWriteOnce, meaning the volume can be mounted by only one node at a time.

Copyright © Thinknyx Technologies LLP

What We’ll Build in the Demo

- **Demonstrations**
  - **IAM role** with the AmazonEBSCSIDriverPolicy
  - **Custom trust policy** for the CSI driver
  - **Amazon EBS CSI driver add-on** installation
  - Annotating the appropriate service account for **IRSA** configuration
  - Use a Deployment to attach a volume to a single Pod
  - Use StatefulSet to create multiple Pods, each with its own PVC using volumeClaimTemplate
  - Observe how volumes behave during Pod deletion

Copyright © Thinknyx Technologies LLP

Demo

IAM Configurations to use EBS as Storage

Copyright © Thinknyx Technologies LLP

Demo

Install & Configure EBS CSI Driver

Copyright © Thinknyx Technologies LLP

Demo

Persistent Storage with PVC EBS CSI Driver

Copyright © Thinknyx Technologies LLP

Demo

Persistent Storage with ClaimTemplates

Copyright © Thinknyx Technologies LLP

Overview of EFS for Shared Access Across Pods

Copyright © Thinknyx Technologies LLP

Overview of EFS for Shared Access Across Pods

How Amazon EBS allows us to add persistent block storage to our Kubernetes workloads

How StatefulSets help manage stateful applications by attaching a dedicated EBS volume to each Pod

Copyright © Thinknyx Technologies LLP

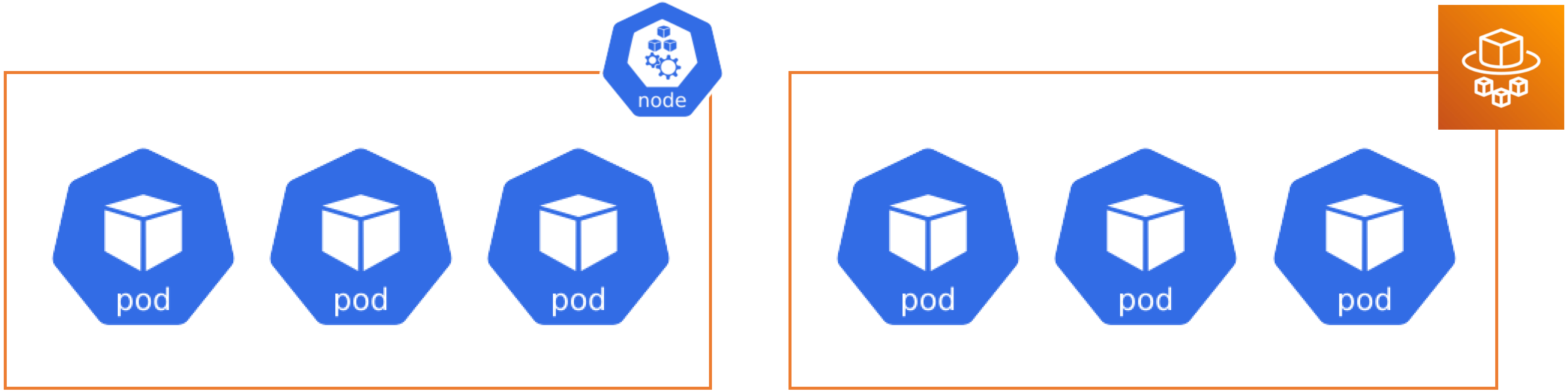
Overview of EFS for Shared Access Across Pods

EBS volumes support single-node attachment only

Cannot be shared across multiple Pods simultaneously

Copyright © Thinknyx Technologies LLP

Overview of EFS for Shared Access Across Pods



Need to access the same data at the same time

Copyright © Thinknyx Technologies LLP

Why Use Amazon EFS?

- Fully managed, serverless, and elastic file system
- Functions like a traditional local file system, but cloud-based
- Supports simultaneous access by multiple Pods across nodes
- Pay only for the storage used - no over-provisioning required

Copyright © Thinknyx Technologies LLP

Why Use Amazon EFS?

Content management systems such as WordPress or Drupal

Developer tools like JIRA and Git

Shared notebook systems like Jupyter

Even basic home directories that multiple users or Pods need to access

Copyright © Thinknyx Technologies LLP

How is EFS Different from EBS?

Feature	EBS (Elastic Block Store)	EFS (Elastic File System)
Type	Block storage	Shared file storage
Pod Attachment	Attaches to one Pod at a time	Supports multiple Pods on multiple Nodes
Use Case	Ideal for databases and single-instance access	Ideal for shared workloads needing simultaneous access
Access Mode	Typically ReadWriteOnce	Supports ReadWriteMany
Compatibility	Works only with Node Groups	Works with Node Groups and Fargate

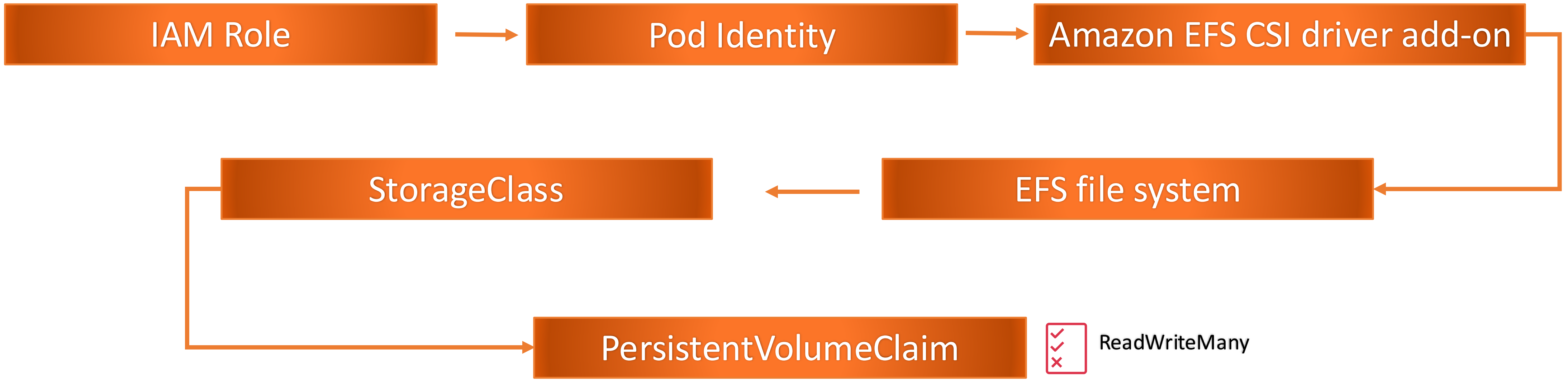
Copyright © Thinknyx Technologies LLP

How Do We Use EFS in Kubernetes?

EKS Pod Identity

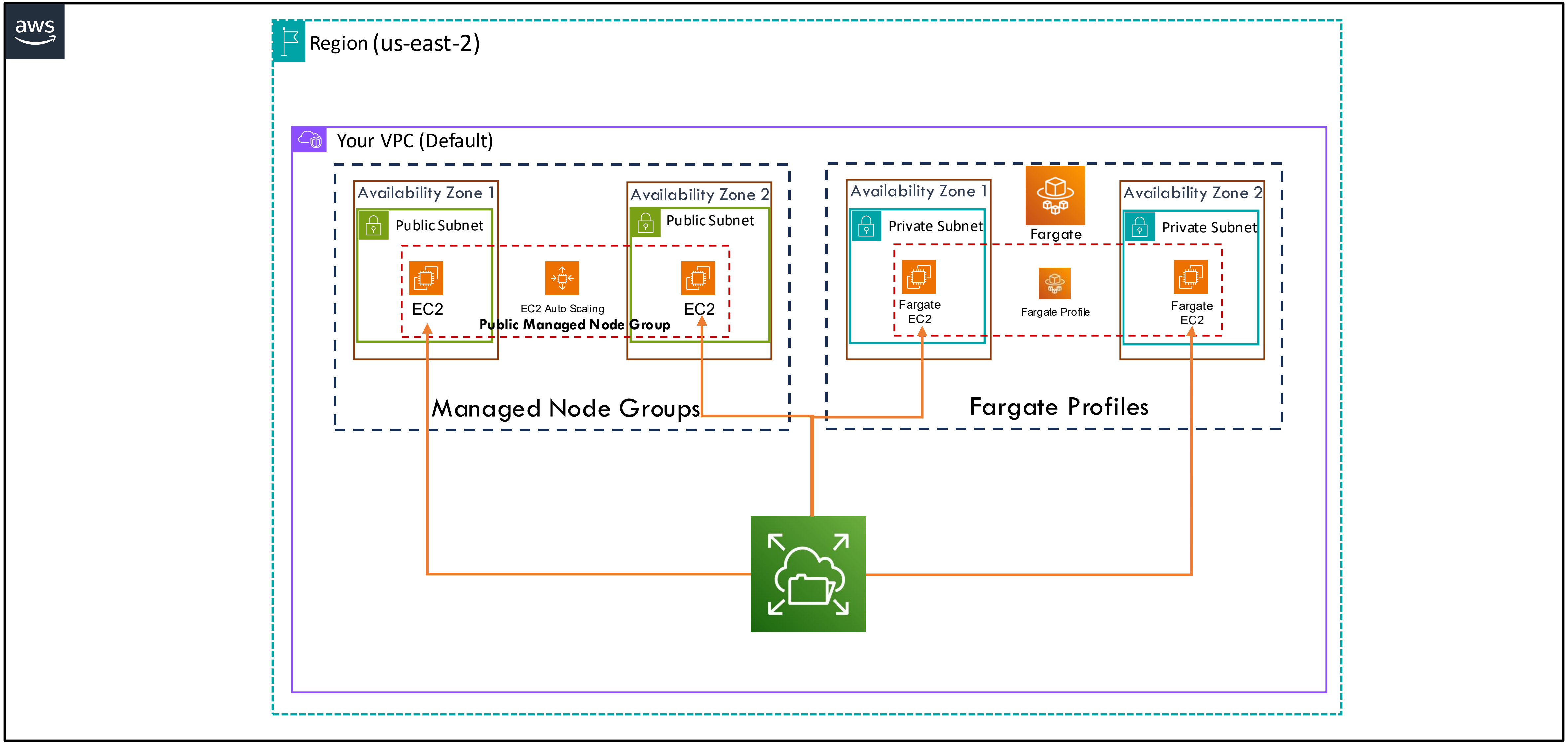
How Do We Use EFS in Kubernetes?

- Steps



Copyright © Thinknyx Technologies LLP

How Do We Use EFS in Kubernetes?



Copyright © Thinknyx Technologies LLP

EFS Behavior in Kubernetes

- EFS volume binding is immediate - no need to wait for Pod scheduling
- Unlike EBS, EFS supports simultaneous access by multiple Pods
- Pods across different nodes or Fargate can read/write to the same EFS volume
- Ideal for workloads needing shared, concurrent access

Copyright © Thinknyx Technologies LLP

What Amazon EFS is

How it benefits us over EBS in shared storage

Copyright © Thinknyx Technologies LLP

Demo

Configurations to use EFS as PersistentVolume

Copyright © Thinknyx Technologies LLP

Demo

Using EFS with for Multiple Pods

Copyright © Thinknyx Technologies LLP

Storage Options

Section Summary

- **→ *Amazon EBS***
  - ❑ *Block storage for individual Pods*
  - ❑ *Ideal for databases and stateful apps*
  - ❑ *Uses PVCs and StatefulSets*
- **→ *Amazon EFS***
  - ❑ *Shared storage across multiple Pods*
  - ❑ Supports EKS and Fargate

Copyright © Thinknyx Technologies LLP

Managing Networking and Ingress

Copyright © Thinknyx Technologies LLP

Managing Networking and Ingress

Section Overview

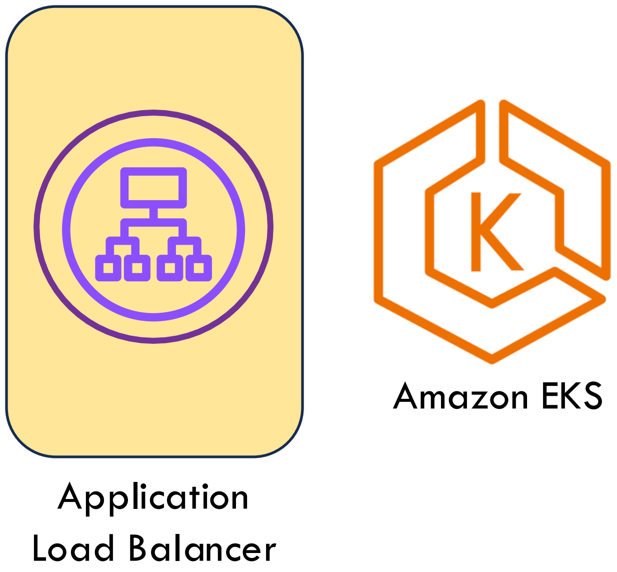
- → *Set up and understand AWS Load Balancer Controller in EKS*
- → *Create required IAM resources*
- → *Deploy the controller using Helm*
- → *Apply necessary Custom Resource Definitions (CRDs)*
- → *Understand how to route external traffic to services in the cluster*

Copyright © Thinknyx Technologies LLP

Understanding Ingress Controllers and ALB Setup

Copyright © Thinknyx Technologies LLP

Understanding Ingress Controllers and ALB Setup



- Classic Load Balancer
- Network Load Balancer

ALB operates at **Layer 7** (Application Layer)

Supports **path-based** and **host-based** routing

Copyright © Thinknyx Technologies LLP

What is the AWS Load Balancer Controller?

AWS Load Balancer Controller

Kubernetes controller developed by AWS

Automates provisioning and configuration of **AWS Elastic Load Balancers**

Application Load Balancers	Ingress
Network Load Balancers	Service

Copyright © Thinknyx Technologies LLP

What is the AWS Load Balancer Controller?

AWS ALB Ingress Controller

Ticketmaster

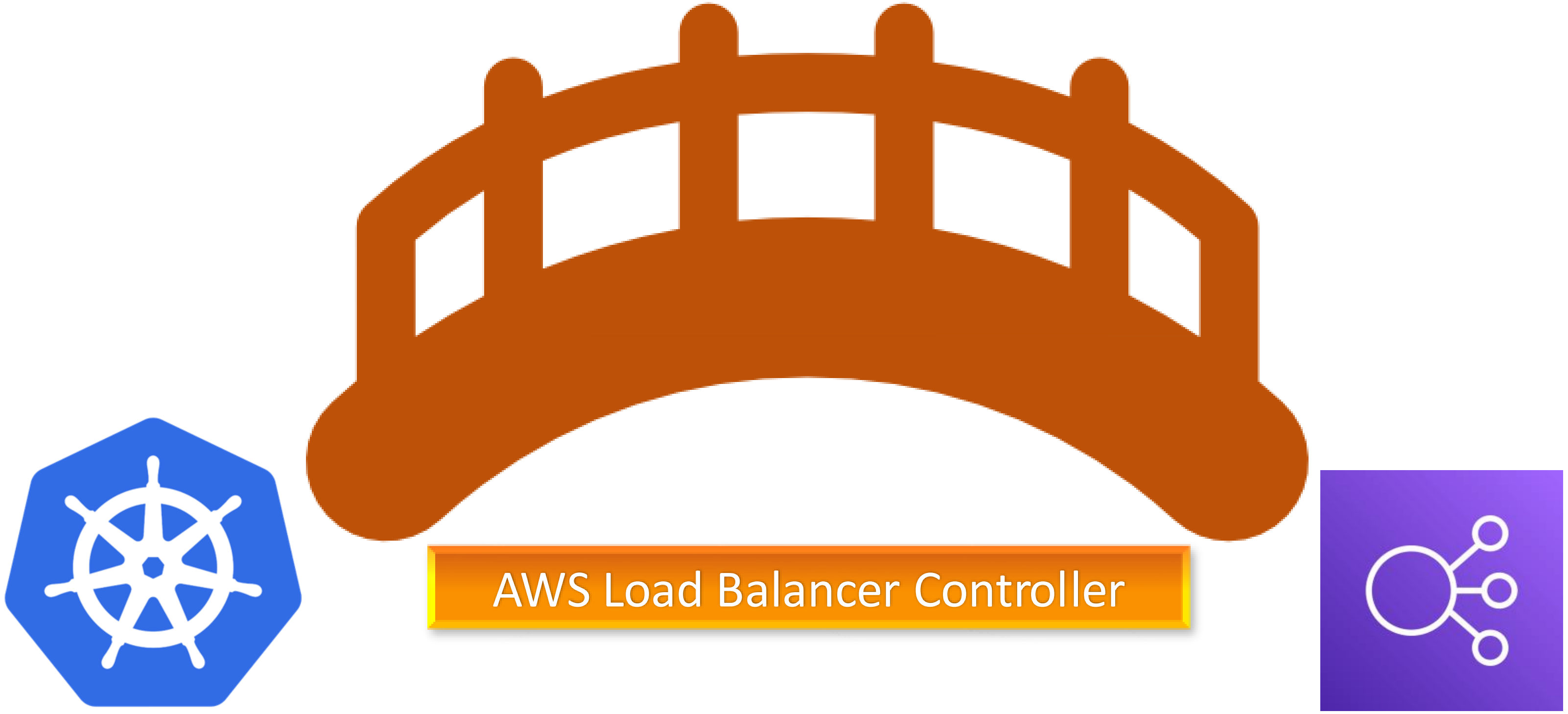
CoreOS

Donated to the **Kubernetes SIG-AWS** community to be maintained by contributors

from **AWS**, **CoreOS**, **Ticketmaster**, and others

Copyright © Thinknyx Technologies LLP

What is the AWS Load Balancer Controller?



Copyright © Thinknyx Technologies LLP

What Happens When You Create a Kubernetes Ingress?

Watches the Kubernetes API for the new Ingress resource

Provisions a new Application Load Balancer automatically

Configures listeners, typically on ports 80 and 443

Creates target groups for your backend services

Sets up routing rules based on hostnames or paths defined in the Ingress spec

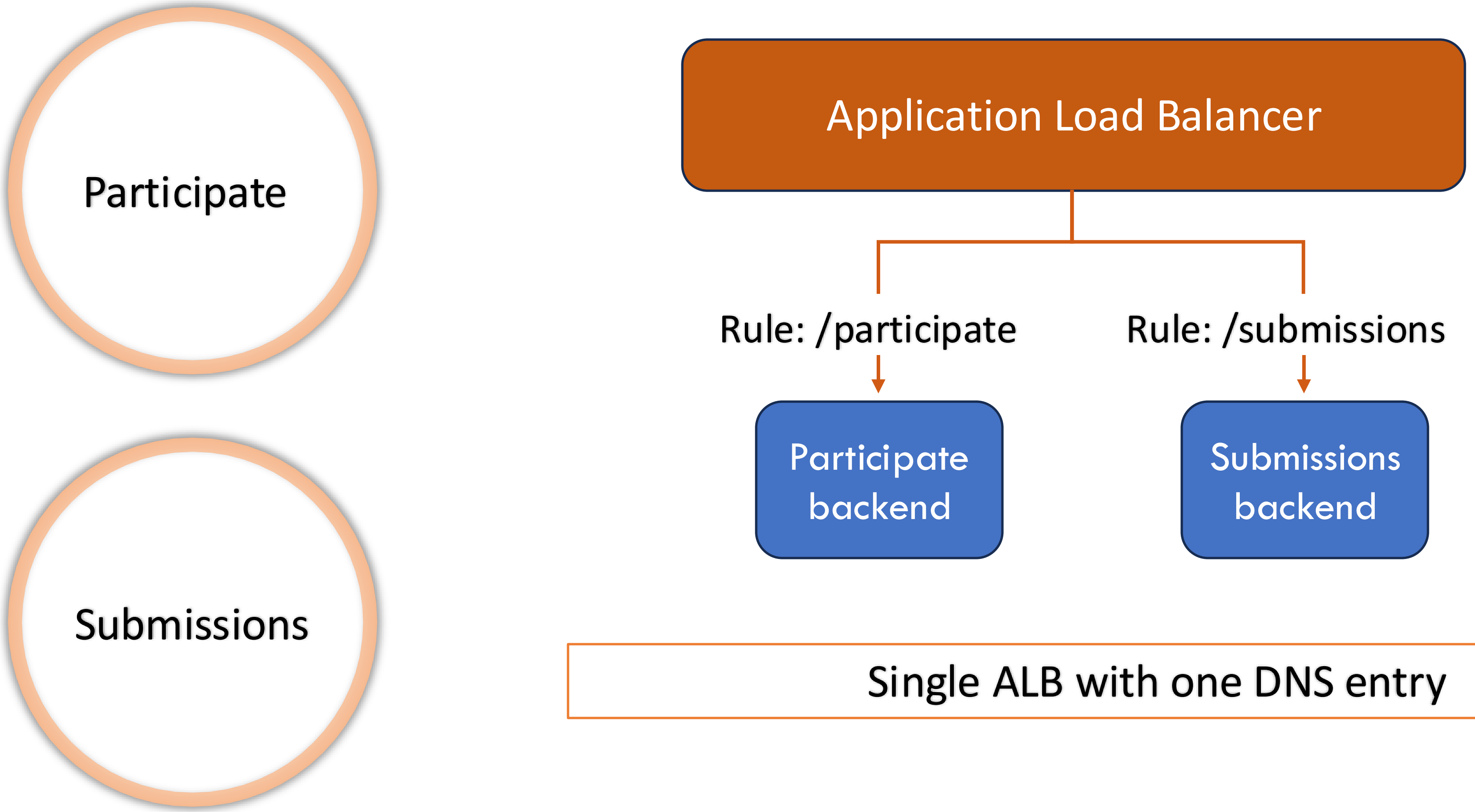
Copyright © Thinknyx Technologies LLP

What Happens When You Create a Kubernetes Ingress?

- ALB routes HTTP/HTTPS traffic to application Pods on EC2 or AWS Fargate
- ALB creation and management is fully automated
- Supports both public and private subnet deployments
- Choose subnet type based on whether the app is internet-facing or internal

Copyright © Thinknyx Technologies LLP

Our Use Case Example - Lucky Draw



Single ALB with one DNS entry

Copyright © Thinknyx Technologies LLP

Traffic Modes Supported by ALB

Instance Mode

- Traffic from ALB is forwarded to **NodePort** on EC2 worker nodes
- From NodePort, traffic is routed to the target Pods
- This is the default configuration mode
- Best suited for workloads running on EC2 worker nodes

Copyright © Thinknyx Technologies LLP

Traffic Modes Supported by ALB

IP Mode

- ALB routes traffic directly to Pod IPs
- NodePort is bypassed, improving efficiency
- Required for AWS Fargate deployments
- Essential for EKS on hybrid infrastructure

➢ We annotate the Ingress resource with:

alb.ingress.kubernetes.io/target-type: ip

IP mode

Copyright © Thinknyx Technologies LLP

What We'll Be Doing in the Demo

Create a custom IAM policy and IAM role

Set up the trust relationship

Create the Kubernetes service account and annotate it with the IAM role

Add the AWS EKS Helm chart repository

Install the AWS Load Balancer Controller

Apply the required CRDs

Deploy three services

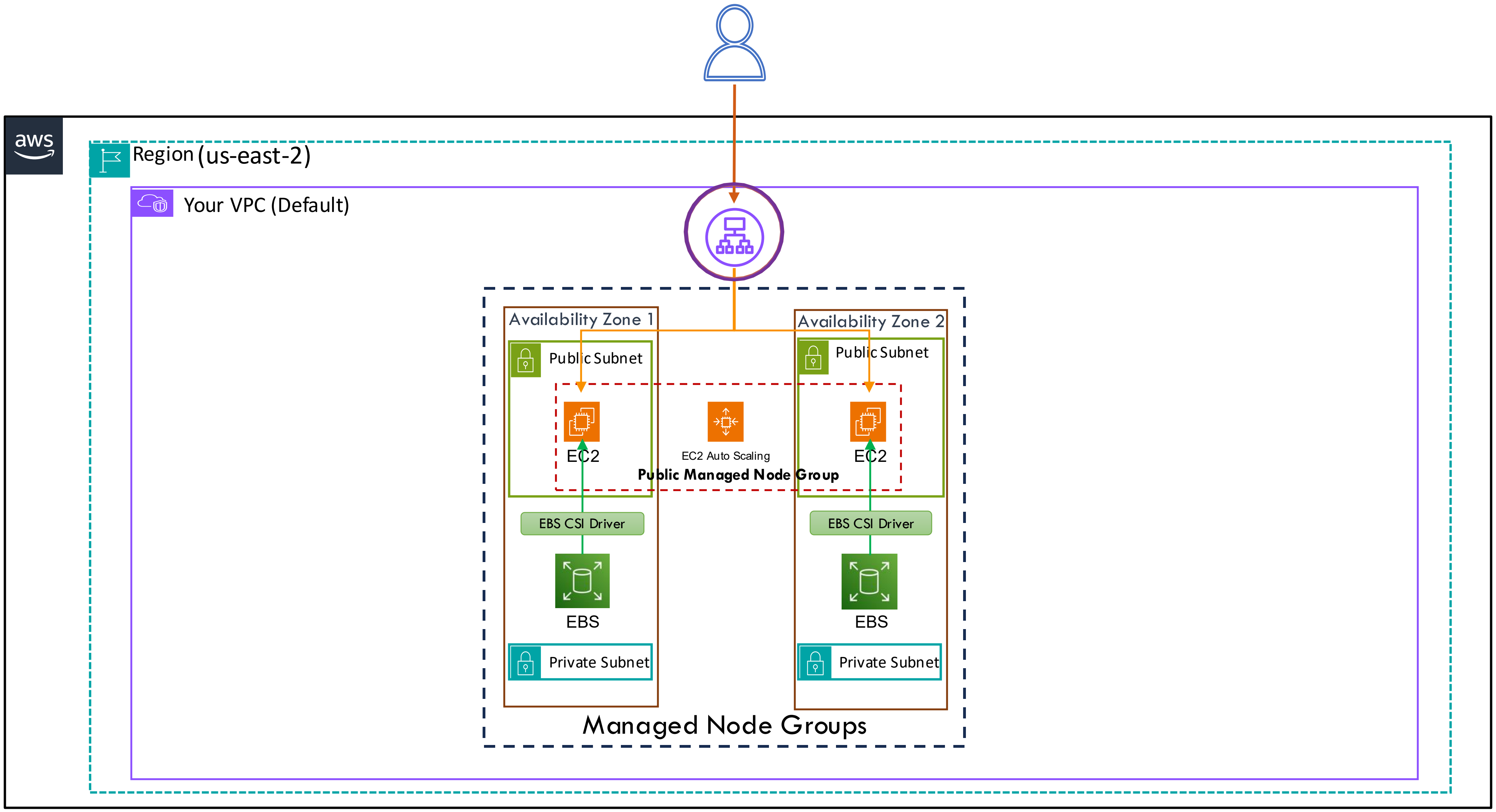
Copyright © Thinknyx Technologies LLP

What We'll Be Doing in the Demo

Define Ingress resource

```
kind: Ingress
metadata:
  name: demo-alb-ingress
  namespace: alb-ingres-controller-demo
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/subnets: subnet-0438cb26b1e4950e5, subnet-021f703c84ac65677, subnet-06cec28f61936c479
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/backend-protocol: HTTP
spec:
  ingressClassName: alb
```

Copyright © Thinknyx Technologies LLP



Copyright © Thinknyx Technologies LLP

Demo

Creating IAM Policy & Role

Copyright © Thinknyx Technologies LLP

Demo

Deploying ALB Ingress Controller Resources

Copyright © Thinknyx Technologies LLP

Demo

Deploying ALB Ingress Controller to Route External Traffic

Copyright © Thinknyx Technologies LLP

Managing Networking and Ingress

Section Summary

- → *Created IAM policy and role, linked to EKS cluster via OIDC provider*
- → *Installed controller using Helm*
- → *Manually applied required CRDs*
- → *Demonstrated automatic ALB provisioning from Kubernetes Ingress*
- → *Showed routing of external traffic to application Pods via ALB*

Copyright © Thinknyx Technologies LLP

Scaling Node Groups

Copyright © Thinknyx Technologies LLP

Scaling Node Groups

Section Overview

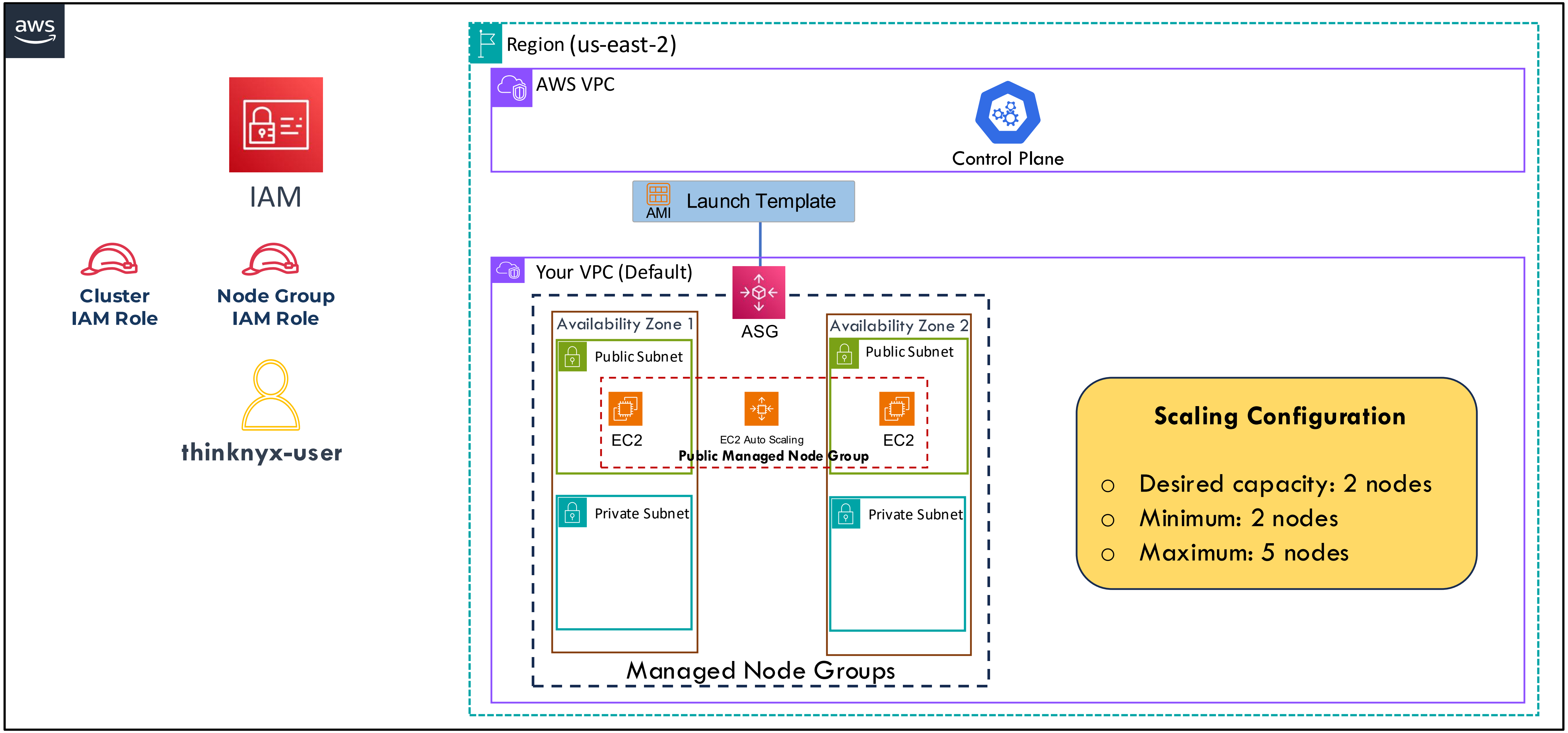
- → *Explore how Kubernetes scales worker nodes automatically using the* ***Cluster Autoscaler*** *in Amazon EKS*
- → *Learn:*
  - ❑ *Working*
  - ❑ *IAM role set up*
  - ❑ *Deployment using auto-discovery*
- → *Watch nodes scale based on demand*

Copyright © Thinknyx Technologies LLP

Understanding Cluster Autoscaler for Node Groups

Copyright © Thinknyx Technologies LLP

Understanding Cluster Autoscaler for Node Groups



Copyright © Thinknyx Technologies LLP

Understanding Cluster Autoscaler for Node Groups

- Just setting ASG min, max, and desired values doesn't enable autoscaling based on pod demand
- By default, ASG maintains the desired EC2 count
- It replaces unhealthy instances but doesn't scale for pod demand
- Scaling up/down needs the **Cluster Autoscaler**

Copyright © Thinknyx Technologies LLP

What is Cluster Autoscaler?

- Kubernetes **Cluster Autoscaler** is a powerful tool maintained by the SIG Autoscaling group
- Handles dynamic cluster scaling based on pod scheduling needs
- If pods can't be scheduled due to **low resources**, Cluster Autoscaler **increases** ASG nodes
- If nodes are **underused**, it **scales down** the ASG to reduce costs
- It simulates node changes and makes smart scaling decisions

Copyright © Thinknyx Technologies LLP

Integration with EKS

Each managed node group uses an EC2 Auto Scaling Group in your AWS account

EKS auto-tags resources for Cluster Autoscaler discovery

ASG covers all subnets chosen during node group setup

Managed node groups offer easy lifecycle management and integrate with autoscaling tools

Copyright © Thinknyx Technologies LLP

How Cluster Autoscaler Works on AWS

1

Runs as a **Deployment** in your cluster, usually in **kube-system**

2

Uses **leader election;** only one replica acts at a time (not horizontally scalable)

3

Follows ASG **min/max limits,** adjusting desired capacity within bounds

4

For Mixed Instance Policies, simulates nodes using the **first instance type** in the launch template

Copyright © Thinknyx Technologies LLP

Why Managed Node Groups?

- Managed Node Groups auto-tag resources for Cluster Autoscaler
- Support graceful node draining on scale-down
- Simplify management and enhance reliability

- In the upcoming demos:
  - Steps to securely configure Cluster Autoscaler
  - Demonstration of Cluster Autoscaler in our EKS cluster

Copyright © Thinknyx Technologies LLP

Step 1: Creating the IAM Role for Cluster Autoscaler

Customer Managed Policy

- Amazon EC2
- EC2 Auto Scaling
- Amazon EKS

- Create an IAM role using **web identity federation** for the Kubernetes service account via **EKS OIDC**
- Edit **trust relationship policy** to allow only the Cluster Autoscaler's service account to assume the role

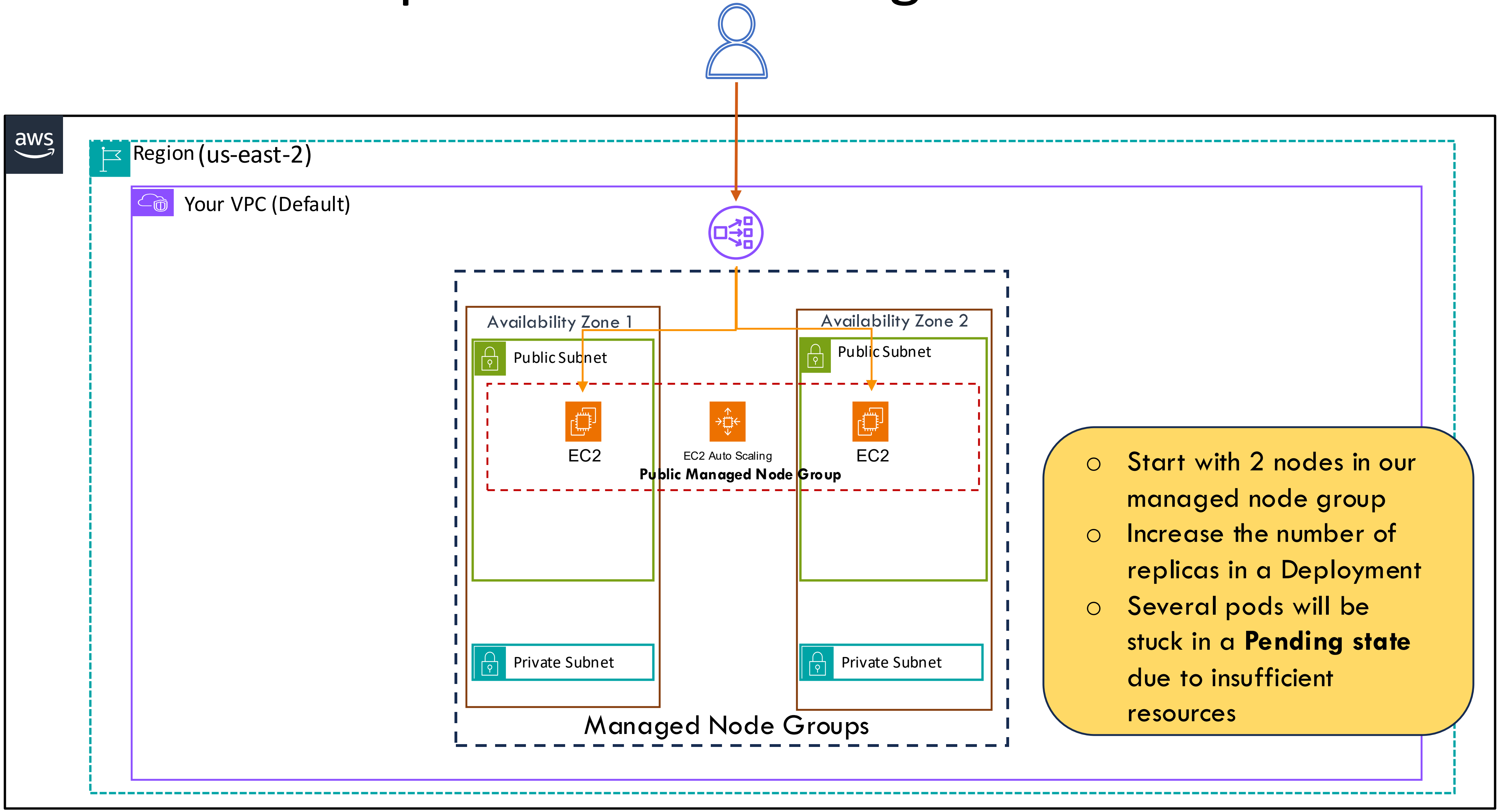
Copyright © Thinknyx Technologies LLP

Step 2: Enabling Auto-Discovery Mode

- Deploy the Cluster Autoscaler using the **Auto-Discovery method**
- Autoscaler automatically detects eligible ASGs based on **resource tags**
- EKS already applies the required tags to managed node groups

Copyright © Thinknyx Technologies LLP

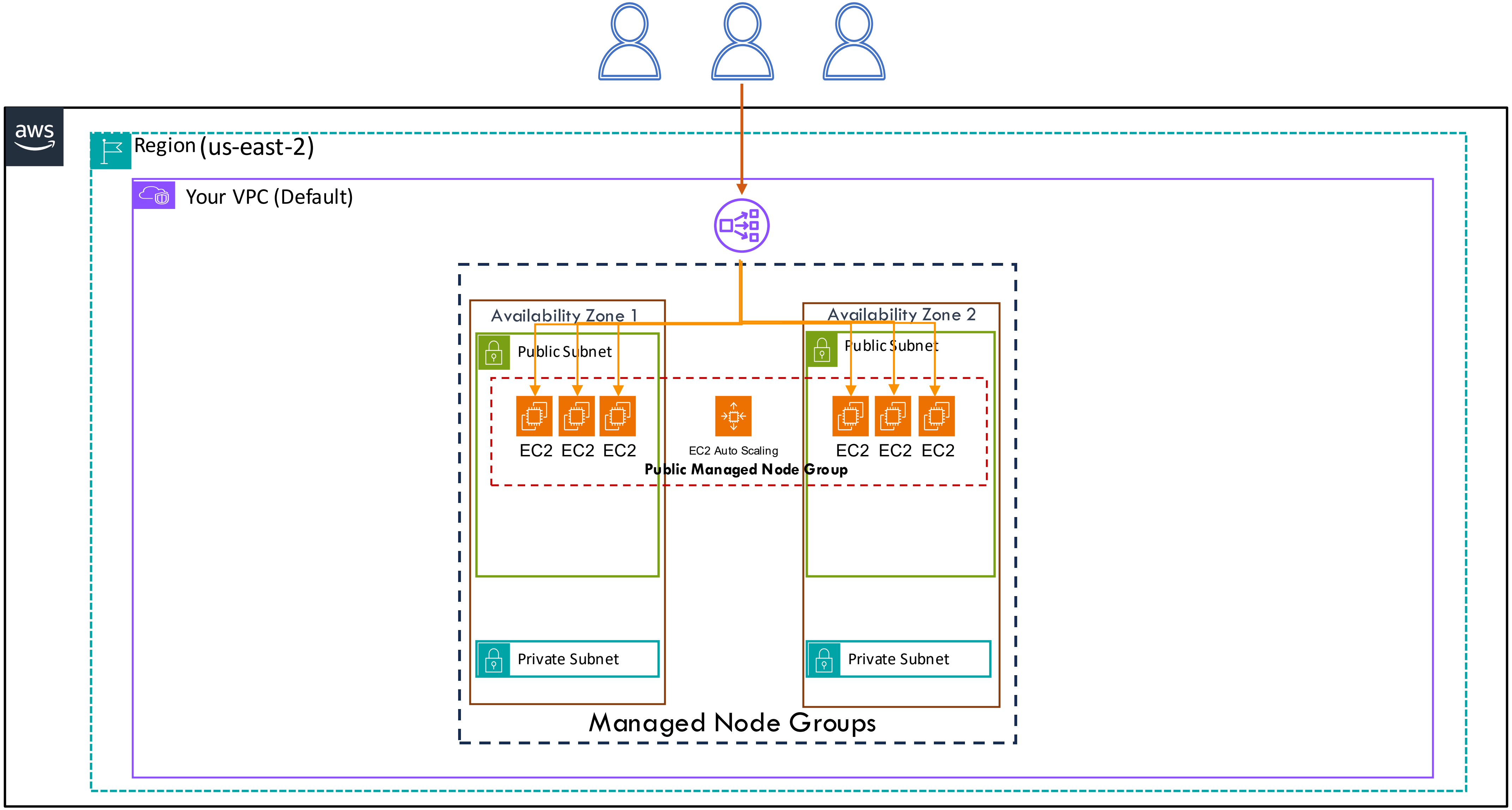
Step 3: Demonstrating Scale-Out



- Start with 2 nodes in our managed node group
- Increase the number of replicas in a Deployment
- Several pods will be stuck in a **Pending state** due to insufficient resources

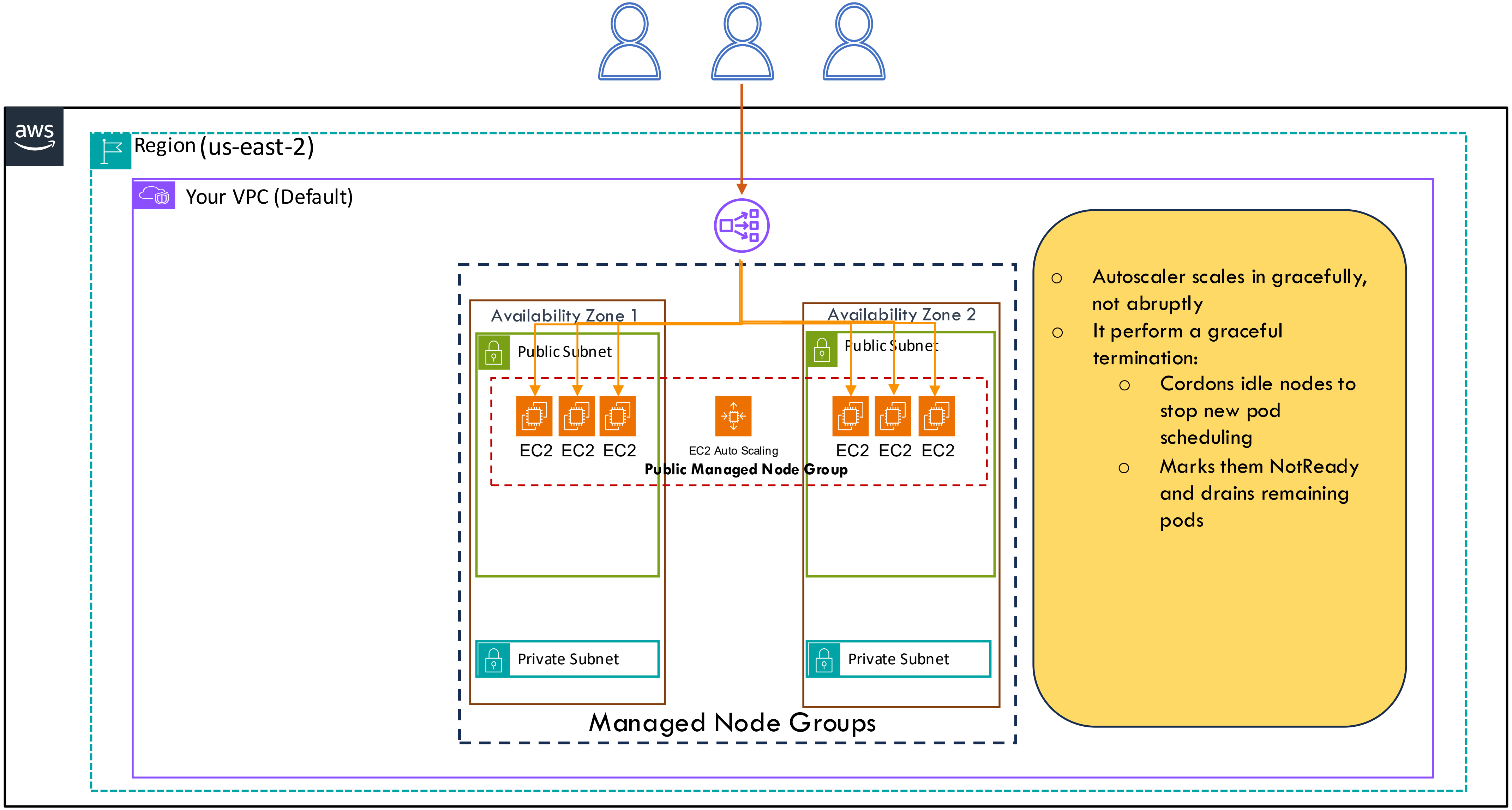
Copyright © Thinknyx Technologies LLP

Step 3: Demonstrating Scale-Out



Copyright © Thinknyx Technologies LLP

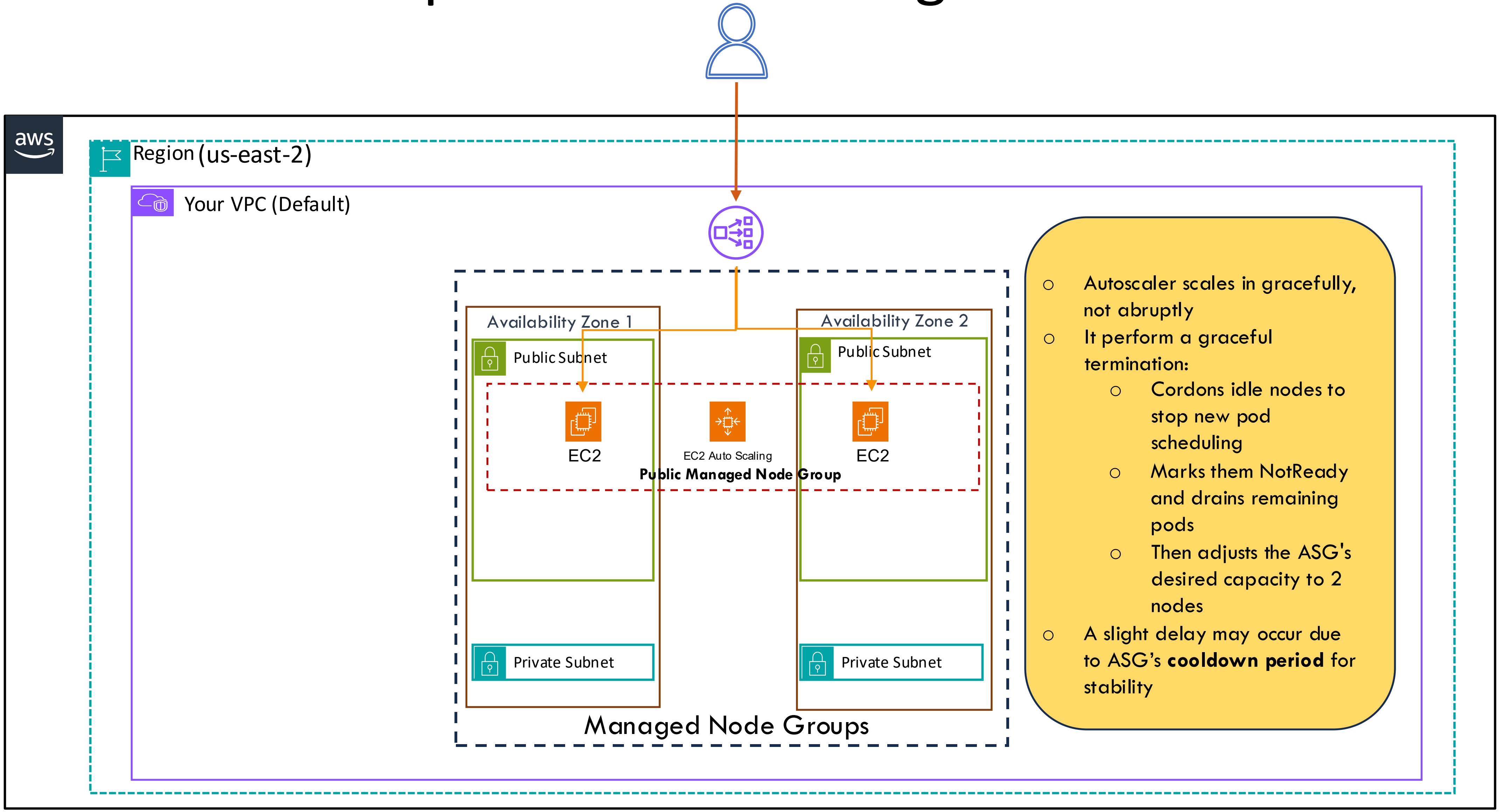
Step 4: Demonstrating Scale-In



- Autoscaler scales in gracefully, not abruptly
- It perform a graceful termination:
  - Cordons idle nodes to stop new pod scheduling
  - Marks them NotReady and drains remaining pods

Copyright © Thinknyx Technologies LLP

Step 4: Demonstrating Scale-In



- Autoscaler scales in gracefully, not abruptly
- It perform a graceful termination:
  - Cordons idle nodes to stop new pod scheduling
  - Marks them NotReady and drains remaining pods
  - Then adjusts the ASG's desired capacity to 2 nodes
- A slight delay may occur due to ASG's **cooldown period** for stability

Copyright © Thinknyx Technologies LLP

Demo

Created IAM Policy & Role for Cluster Autoscaler

Copyright © Thinknyx Technologies LLP

Demo

Enabling and Observing Cluster Autoscaler in Action

Copyright © Thinknyx Technologies LLP

Scaling Node Groups

Section Summary

- → *Integrated **Cluster Autoscaler** with Amazon EKS for automatic node scaling*
- → *Created a IAM role with OIDC trust*
- → *Deployed the autoscaler using **auto-discovery mode***
- → *Observed **scale-out** when pods were pending*
- → *Observed **scale-in** when resources were freed*

Copyright © Thinknyx Technologies LLP

ECR Integration

Copyright © Thinknyx Technologies LLP

ECR Integration

Section Overview

- → *Learn to create and manage Amazon ECR repositories*
- → *Authenticate to ECR from a client machine*
- → *Push Docker images to ECR*
- → *Use ECR-hosted images in EKS deployments*
- → *Understand how ECR integrates with Kubernetes workflows*

Copyright © Thinknyx Technologies LLP

Demo

Creating and Managing Amazon ECR Repositories

Copyright © Thinknyx Technologies LLP

Demo

Authenticating on Ubuntu Machine for ECR

Copyright © Thinknyx Technologies LLP

Demo

Pushing a Docker Image to ECR and Deploying it on EKS

Copyright © Thinknyx Technologies LLP

ECR Integration

Section Summary

- → *Created and configured Amazon ECR repositories*
- → *Authenticated and pushed Docker images to ECR from Ubuntu*
- → *Tagged, managed, and scanned container images in ECR*
- → *Deployed ECR-hosted images to EKS using IAM roles for secure image pulling*

Copyright © Thinknyx Technologies LLP

Thinknyx®

You Trust, We Deliver!

Mastering Amazon EKS Hands-On

Copyright © Thinknyx Technologies LLP

01 Essentials of Container Orchestration with Kubernetes on AWS Cloud

02 Deployed & Managed your own EKS cluster

03 Configured Managed Node Groups & Fargate profiles

04 Expose services securely using the AWS Load Balancer Controller

05 Provision persistent storage with EBS and EFS

06 Cluster efficiently scaled using the Cluster Autoscaler

07 Integration with Amazon ECR to store and deploy container images

Copyright © Thinknyx Technologies LLP

Explore Our In-depth Courses

Practical Kubernetes – Beyond CKA and CKAD | Hands-On

Build and Scale with AWS Cloud - A Hands-On Beginners Guide

Copyright © Thinknyx Technologies LLP

Follow us on:

@thinknyx

@thinknyx

@thinknyx-technologies

@thinknyx

@thinknyx-technologies

Copyright © Thinknyx Technologies LLP