DevOps Shack

Real-Time Corporate K8 Configurations

[Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps]

Azure Kubernetes Service Config

Configuration	PROD	DR	QA	DEV
Cluster Name	XYZ-P	XYZ-DR	XYZ-QA	XYZ-DEV
Region	West Europe	West Europe	West Europe	West Europe
Kubernetes Version	1.27.3	1.27.3	1.27.3	1.27.3
Network Plugin	Azure CNI	Azure CNI	Azure CNI	Azure CNI
Number of node pools	1	1	1	1
Node pool configuration	Min: 3 / Max: 5	Min: 3 / Max: 5	Min: 3 / Max: 5	Min: 3 / Max: 8
Node Size	Standard E4s v5 (4 vCPUs, 32 GiB)	Standard E4s v5 (4 vCPUs, 32 GiB)	Standard E4s v5 (4 vCPUs, 32 GiB)	Standard_D4s_v4 (4 vCPUs, 16 GiB)
Max Pods per Node	30	30	30	30
Authentication	Local accounts with Kubernetes RBAC	Local accounts with Kubernetes RBAC	Local accounts with Kubernetes RBAC	Local accounts with Kubernetes RBAC

The provided table outlines various configuration parameters for Azure Kubernetes Service (AKS) across different environments: Production (PROD), Disaster Recovery (DR), Quality Assurance (QA), and Development (DEV). Each environment is tailored to suit specific needs in the lifecycle of software development and deployment. Here's a detailed explanation of each configuration parameter:

1. Cluster Name

- **XYZ-P, XYZ-DR, XYZ-QA, XYZ-DEV**: These are the names given to the Kubernetes clusters in different environments. Each name likely reflects its purpose:
  - **XYZ-P**: Production environment where live applications are run.
  - **XYZ-DR**: Disaster Recovery environment, used to ensure availability and continuity.
  - **XYZ-QA**: Quality Assurance for testing and quality checks before production.
  - **XYZ-DEV**: Development environment for building and testing applications.

2. Region

- **West Europe**: All environments are located in the "West Europe" region. This ensures that all environments are geographically aligned, which can help with network latency and regional compliance, but may be a risk if a regional outage occurs.

3. Kubernetes Version

- **1.27.3**: All clusters are running Kubernetes version 1.27.3. Using the same version across all environments ensures consistency and reduces issues that might arise from version discrepancies.

4. Network Plugin

- **Azure CNI**: All environments use Azure CNI (Container Networking Interface). Azure CNI allows Kubernetes pods to have an IP address in the Azure VNET, providing enhanced network capabilities like better security with VNET features and the ability to connect easily to other services within Azure.

5. Number of Node Pools

- **1**: Each environment has one node pool. Node pools contain groups of nodes, which are the VMs that host your applications.

6. Node Pool Configuration

- **Min: 3 / Max: 5 (and Max: 8 for DEV)**: This setting controls the scaling of nodes within the node pool. All environments are configured to scale between 3 to 5 nodes automatically based on load, except for the DEV environment, which can scale up to 8 nodes. This flexibility in the DEV environment allows accommodating varying loads during development without manually resizing the pool.

7. Node Size

- **Standard E4s v5 for PROD, DR, QA** and **Standard_D4s_v4 for DEV**:
  - **Standard E4s v5**: These nodes have 4 vCPUs and 32 GiB of RAM, indicating robust performance suitable for production, disaster recovery, and quality assurance.
  - **Standard_D4s_v4**: Used in the DEV environment, these nodes also have 4 vCPUs but only 16 GiB of RAM, which may be a cost-effective choice for development purposes where the maximum capacity of the production environment is not required.

8. Max Pods per Node

- **30**: This configuration limits the maximum number of pods that can run on a single node to 30. This setting helps in managing the Kubernetes scheduler's decisions on pod placement based on available resources and helps ensure that no single node is overwhelmed by too many pods.

9. Authentication

- **Local accounts with Kubernetes RBAC**: All environments use local Kubernetes accounts integrated with Role-Based Access Control (RBAC) for authentication and authorization. RBAC allows fine-grained control over what actions different users and services can perform, enhancing the security posture of the clusters.

  This setup exemplifies a well-thought-out AKS configuration that aligns different environments for effective development, testing, and production operations with an emphasis on consistency, scalability, and security.

Elastic Kubernetes Service

Configuration	PROD	DR	QA	DEV
Cluster Name	XYZ-P	XYZ-DR	XYZ-QA	XYZ-DEV
Region	Europe (Frankfurt) eu-central-1	Europe (Frankfurt) eu-central-1	Europe (Frankfurt) eu-central-1	Europe (Frankfurt) eu-central-1
Kubernetes Version	1.27.3	1.27.3	1.27.3	1.27.3
Network Plugin	AWS VPC CNI	AWS VPC CNI	AWS VPC CNI	AWS VPC CNI
Number of Node Pools	1	1	1	1
Node Pool Configuration	Auto Scaling Group: Min 3, Max 5	Auto Scaling Group: Min 3, Max 5	Auto Scaling Group: Min 3, Max 5	Auto Scaling Group: Min 3, Max 8
Node Size	EC2 Instance Type: m5.xlarge (4 vCPUs, 16 GiB)	EC2 Instance Type: m5.xlarge (4 vCPUs, 16 GiB)	EC2 Instance Type: m5.xlarge (4 vCPUs, 16 GiB)	EC2 Instance Type: t3.xlarge (4 vCPUs, 16 GiB)
Max Pods per Node	30	30	30	30
Authentication	AWS IAM roles with Kubernetes RBAC	AWS IAM roles with Kubernetes RBAC	AWS IAM roles with Kubernetes RBAC	AWS IAM roles with Kubernetes RBAC

The detailed explanation below pertains to the configuration setup for Amazon Elastic Kubernetes Service (EKS) across various environments—PROD, DR, QA, and DEV—as presented in the uploaded image. This information helps in understanding the setup for each environment tailored for specific roles within the software deployment lifecycle:

General Configuration

1. Cluster Name:

- **XYZ-P, XYZ-DR, XYZ-QA, XYZ-DEV:** These names identify distinct EKS clusters dedicated to different environments. Each name helps in segregating and managing resources per their intended use:
    - **PROD**: Production, for live applications.
    - **DR**: Disaster Recovery, for backup and recovery.
    - **QA**: Quality Assurance, for testing.
    - **DEV**: Development, for development purposes.

2. Region:

- **Europe (Frankfurt) eu-central-1**: All clusters are located in the AWS Frankfurt region. This central location in Europe helps in managing data residency and compliance while providing robust connectivity and services.

3. Kubernetes Version:

- **1.27.3**: Ensures that all clusters run the same version of Kubernetes, promoting consistency across deployments, reducing compatibility issues, and simplifying management and updates.

Specific Configurations

4. Network Plugin:

- **AWS VPC CNI**: The Amazon VPC CNI plugin allows Kubernetes pods to have native VPC networking. This setup enhances network security and performance by assigning each pod an IP address from the VPC, thereby leveraging native AWS networking features.

5. Number of Node Pools:

- **1**: Indicates a single node pool per cluster, which simplifies node management. Each pool can automatically adjust the number of nodes based on demand within specified limits.

6. Node Pool Configuration:

- **Auto Scaling Group Settings:**
  - **Min 3, Max 5** (PROD, DR, QA): Supports standard operational load with the capability to scale up based on increased demand.
  - **Min 3, Max 8** (DEV): Allows more flexibility in development for testing different scales and loads.

7. Node Size:

- **EC2 Instance Types:**
  - **m5.xlarge (4 vCPUs, 16 GiB)** for PROD, DR, and QA: These instances provide a balance of compute, memory, and networking resources, suitable for most production and testing scenarios.
  - **t3.xlarge (4 vCPUs, 16 GiB)** for DEV: Typically more cost-efficient, suitable for development environments where cost savings are prioritized over the continuous uptime.

8. Max Pods per Node:

- **30**: This limit is set to manage the allocation of networking and compute resources efficiently, preventing any single node from being overwhelmed by too many pods, thus maintaining network and application performance.

9. Authentication:

- **AWS IAM roles with Kubernetes RBAC**: Integrates AWS Identity and Access Management (IAM) with Kubernetes Role-Based Access Control (RBAC). This setup allows fine-grained access control over resources in Kubernetes clusters, utilizing AWS's robust security framework to manage permissions and access.

Summary

This setup provides a robust, scalable, and secure framework for deploying and managing Kubernetes applications across various stages of development and production. By using AWS EKS with these configurations, organizations can leverage AWS's powerful infrastructure and services to maintain efficient and effective control over their deployments, ensuring reliability, scalability, and security.