**Question 13**

**Describe a time you improved the performance of an infrastructure system. What challenges did you face?**

**Answer;**

### **The Situation**

### While managing the infrastructure for a high-traffic e-commerce platform hosted on Amazon EKS (Elastic Kubernetes Service), we noticed a significant performance degradation during peak traffic hours. Key issues included:

* **Slow API responses**: Response times exceeding Service Level Agreements (SLAs).
* **Resource contention**: Pods frequently evicted due to insufficient resources.
* **Unstable scaling**: Delays in adding new nodes to handle increased load.

The system needed to handle traffic spikes during promotional campaigns while maintaining high availability and performance.

### **My Role**

As a Cloud DevOps Engineer, I was responsible for diagnosing the performance issues, identifying bottlenecks, and implementing optimizations to meet SLA requirements.

### **The Actions Taken**

#### **1. Diagnose the Problem**

* **Logs and Metrics**: Used Prometheus and Grafana to analyze CPU, memory, and disk usage. Identified pods competing for resources and some underutilized nodes.
* **Scaling Delays**: Reviewed Cluster Autoscaler logs and discovered it was slow in scaling nodes due to under-configured thresholds.
* **API Bottlenecks**: Used AWS X-Ray for tracing requests and pinpointing slow services, which were database-heavy operations.

#### **2. Address Immediate Resource Contention**

* **Pod Resource Requests and Limits**:
  + Audited resource requests/limits and optimized them to match actual usage patterns.
  + Removed over-provisioning that caused wasted resources.
* **Node Taints and Tolerations**:
  + Configured taints to prioritize critical workloads and prevent eviction during contention.

#### **3. Optimize Scaling**

* **Karpenter Integration**:
  + Replaced Cluster Autoscaler with **Karpenter**, an open-source cluster autoscaler optimized for EKS.
  + Tuned scaling thresholds to respond faster to traffic spikes and reduced scale-down delays.
* **Spot Instances**:
  + Incorporated Spot Instances for cost efficiency while ensuring reliability with a mix of On-Demand and Spot Instances.

#### **4. Optimize Database and Application Performance**

* **Query Optimization**:
  + Worked with the development team to optimize slow SQL queries identified through AWS RDS Performance Insights.
* **Caching**:
  + Introduced Redis caching for frequent queries, reducing database load by 40%.
* **Horizontal Pod Scaling**:
  + Configured Horizontal Pod Autoscaler (HPA) to scale pods based on CPU and custom metrics (e.g., request rate).

#### **5. Stress Testing and Monitoring**

* Performed load testing with k6 to validate the improvements.
* Set up detailed monitoring dashboards in Grafana to track scaling events, pod utilization, and API latency in real time.

### **The Challenges**

1. **Lack of Immediate Visibility**: Initial metrics and logs were insufficient, requiring significant effort to instrument better monitoring.
2. **Balancing Cost and Performance**: Ensuring the platform scaled effectively without over-provisioning resources.
3. **Coordinating Changes Across Teams**: Collaborated closely with developers, database admins, and DevOps engineers to implement holistic improvements.
4. **Production Risks**: Making changes in a live environment while ensuring minimal disruption.

### **The Results**

1. **Improved Latency**: Reduced API response times from 1.5 seconds to under 500ms during peak traffic.
2. **Scalability**: Achieved seamless scaling during promotional events, handling 3x traffic spikes without downtime.
3. **Cost Optimization**: Reduced overall infrastructure costs by 20% through better utilization and Spot Instance adoption.
4. **Better Observability**: Introduced real-time dashboards and alerts, enabling proactive issue detection.

### **Key Takeaways**

This experience reinforced the importance of:

* Comprehensive observability for diagnosing and monitoring systems.
* Leveraging modern tools like Karpenter to simplify scaling in dynamic environments.
* Cross-functional collaboration for system-wide performance optimization.

**Question 14**

**How do you prioritize tasks when multiple urgent issues arise?**

**Answer;**

**Assess Impact:** Evaluate the impact and urgency of each issue.

**Prioritize:** Address the highest impact issues first.

**Delegate:** Delegate tasks if possible to ensure quick resolution.

**Communicate:** Keep stakeholders informed about progress and any changes in priorities.