**1. could you provide an example of a situation where you had to identify and migrate risk security issue or bottlenecks in an aws architecture**

we had our application running smoothly on AWS for a while, but as our user base grew, we started noticing some sluggishness in certain parts of the system. After some digging, it became clear that our tight security measures were actually causing some performance bottlenecks.

So, what did we do? Well, first off, we rolled up our sleeves and dove into our AWS setup. We looked at our network configurations, pored over performance metrics, and sifted through logs to figure out where the slowdowns were happening and why.

After some analysis, we realized that our network ACLs were being a bit too strict, choking off the flow of outbound traffic that our application needed to talk to external services and APIs.

We knew we had to loosen things up without compromising security, so we came up with a plan. We identified the specific outbound traffic patterns our app needed and adjusted our network ACL rules accordingly. We also set up some nifty AWS VPC Flow Logs to keep an eye on things and catch any fishy activity.

Then came the fun part: actually making the changes. We coordinated with the dev team to make sure everything would play nice together, and then we migrated over to the new setup.

After that, it was all about monitoring and fine-tuning. We kept a close eye on performance metrics and security logs to make sure our changes were doing the trick. And you know what? They did. Our app started running smoother than ever, and we all breathed a sigh of relief.

Before change all outbound traffic was denied by default. While this level of restriction may seem secure, it was causing performance bottlenecks because our application needed to communicate with external APIs and services.

* After updation: Rule 100 allows outbound traffic on commonly used ports 80 (HTTP) and 443 (HTTPS), enabling communication with external web services.
* Rule 101 permits outbound traffic on higher ports (1024-65535) for dynamic connections and responses from external servers.

**2. what monitoring tools and practices do you prefer to use when ensuring system health and optimising efficiency on AWS**

1. **Amazon CloudWatch**: CloudWatch is a fundamental tool for monitoring AWS resources and applications in real-time. I utilize CloudWatch metrics to track performance indicators such as CPU utilization, memory usage, network traffic, and disk I/O. Additionally, CloudWatch Alarms enable proactive monitoring by triggering notifications or automated actions when predefined thresholds are breached.
2. **AWS CloudTrail**: CloudTrail provides a comprehensive record of AWS API calls and activities, enabling auditing, compliance, and troubleshooting. By analyzing CloudTrail logs, I can gain insights into resource changes, identify security threats, and track user activity within the AWS environment.
3. **AWS Config**: AWS Config continuously monitors the configuration of AWS resources and evaluates them against predefined rules. I leverage AWS Config to ensure compliance with organizational policies, detect configuration drift, and assess the impact of changes on system health and security.
4. **AWS Trusted Advisor**: Trusted Advisor offers personalized recommendations for optimizing AWS infrastructure across various dimensions, including cost optimization, performance, security, and fault tolerance. I regularly review Trusted Advisor checks to identify opportunities for improving system efficiency and reducing operational costs.
5. **Custom Metrics and Logging**: In addition to native AWS monitoring tools, I implement custom metrics and logging solutions to gain deeper insights into application performance and behavior. This may involve integrating third-party monitoring solutions or developing custom scripts to collect and analyze application-specific metrics.
6. **Infrastructure as Code (IaC)**: Adopting Infrastructure as Code principles allows for consistent and reproducible deployment of AWS resources. I use tools like AWS CloudFormation or Terraform to define infrastructure configurations in code, enabling automated provisioning, version control, and documentation. This ensures that infrastructure changes are managed systematically and can be rolled back if necessary.
7. **Continuous Integration/Continuous Deployment (CI/CD)**: Implementing CI/CD pipelines streamlines the deployment process and facilitates rapid iteration and testing of changes. I utilize services like AWS CodePipeline and AWS CodeDeploy to automate build, test, and deployment workflows, enabling frequent releases while maintaining system stability and reliability.

Here's a practical explanation of how AWS Config is used for drift detection:

1. **Setup**: First, you set up AWS Config in your AWS account. You configure it to monitor the resources you're interested in and specify the rules you want it to check against.
2. **Initial Snapshot**: AWS Config takes an initial snapshot of your resources' configurations and stores it securely.
3. **Continuous Monitoring**: From then on, AWS Config continuously monitors your resources and compares their current configurations to the stored snapshot.
4. **Detecting Drift**: If AWS Config detects any differences between the current configurations and the snapshot, it flags those differences as drift.
5. **Alerts and Notifications**: AWS Config can then alert you via Amazon SNS (Simple Notification Service) or other notification methods, letting you know about the detected drift.
6. **Remediation**: Once you're aware of the drift, you can take action to remediate it. This might involve reverting the changes, updating your configurations, or investigating further to ensure everything is as it should be.
7. **Continuous Compliance**: AWS Config continues to monitor for drift regularly, helping you maintain continuous compliance with your desired configurations and policies.

**3. difference b/w ebs and eks and which ill use when.**

Ebs is fully managed service by Aws. It takes care of the underlying infrastructure for the deployment of your application like servers, load balancer, autoscaling. you just have to upload your code and rest EBS will take care of.

Eks is container orchestration service managed by Aws, It is used for deploying and managing containerized application.

**which to choose when**

if the application is complex then go for eks. If you want more control on infrastructure then go for eks and for more scalability and potability go for eks.

If you want to minimized the overhead of managing the infrastructure then you can go for EBS. For simpler application you can choose EBS

**4. describe your approach to optimising cost in AWS how do you ensure that resources are used efficiently while maintaining system performance and reliability.**

1. spot underutilized ec2 to either remove them or resize them.

2.Look for the idle EBS volume with minimal activity, take the snapshot for backup and delete them.

3. use of s3 lifecycle to change the storage class of S3 based on the frequency of usage of data.

4. Review the autoscaling group policy so that it doesn’t lead to unnecessary instance launch

5. If the workload is fault tolerance then we can make use of spot instance to reduce the expense by 90%

6. commit to 1 year or # year reserved instance for discount

7. check for compute service like lambda, fargate and their saving plans

8. check for inactive ELBS and delete them.

**5. have you worked with aws IAM extensively how do you ensure proper security and access control for resources in your aws enviornment**

Absolutely. IAM, or Identity and Access Management, is a critical component of any AWS environment, and I've had significant experience working with it. One of the fundamental principles I adhere to is the principle of least privilege. This means granting users only the permissions they need to perform their tasks, minimizing the risk of unauthorized access.

I've implemented role-based access control (RBAC) extensively. By defining IAM roles with specific permissions and assigning users or groups to these roles based on their job responsibilities, I ensure that access is granted based on the principle of least privilege. Regularly reviewing and updating role assignments is also crucial to maintaining security posture as user roles and responsibilities evolve.

In addition, I advocate for the use of multi-factor authentication (MFA) to add an extra layer of security to IAM user accounts. Enforcing MFA requires users to provide a second authentication factor, such as a one-time password, in addition to their password when signing in, further safeguarding against unauthorized access.

IAM policies play a central role in defining permissions for users, groups, and roles. I've created and managed IAM policies to specify granular permissions for accessing AWS resources and actions. Regularly reviewing and refining these policies ensures alignment with security best practices and compliance requirements.

I also emphasize the importance of secure credential management. Avoiding the use of long-term access keys whenever possible and encouraging the use of temporary security credentials or IAM roles helps mitigate the risk of credential compromise.

Furthermore, enabling access logging and monitoring with AWS CloudTrail and Amazon CloudWatch Logs allows for real-time monitoring of IAM events and detection of suspicious activity, such as unauthorized access attempts.

Overall, my approach to AWS IAM revolves around implementing best practices, such as the principle of least privilege, RBAC, MFA, secure credential management, and continuous monitoring, to ensure proper security and access control for resources in the AWS environment."

**one of my basic principle that I adhere is the principle of least privilege which mean provide only those permission that are required to perform their task, minimizing the risk of unauthorized access**

**I have alos implement RBAC to enable what actions can be performed on which resource, Also enable MFa which act as an extra layer of security.**

**Make use of temporary password for the IAM roles minimisig the risk of credential compromised  
At the end making use of cloud watch and cloud trail to check the logs for any security breach such as unauthorized access attempts**