**Amazon Services**

**Media content delivery**

* **AWS Services**: The course covers various AWS services such as ElastiCache, CloudSearch, AWS Glue, QuickSight, Athena, AWS Solutions, AWS Transit Gateway, AWS Backup, and AWS Cost Explorer. These services help in optimizing cost and performance.
* **Operational Excellence**: The course discusses the operational excellence process, which includes best practices for maintaining and improving operational efficiency on AWS.
* **Design Scenarios**: The course includes various design scenarios to illustrate resilient, performant, secure, and cost-optimized solutions. These scenarios help in understanding how to apply AWS services effectively.

**Desktop and app streaming**

* **Workspaces**: This AWS service provides virtual desktops in the cloud. You can run operating systems like Linux and Windows (including versions with Office pre-installed) as virtual desktops. Persistent storage is available as a virtual D: drive, which is regularly backed up by AWS.
* **AppStream 2.0**: This service allows you to run individual applications in the cloud, making them appear as if they are running locally on your machine. It is commonly used for custom-developed apps or commercial apps that need to be accessed by multiple users.
* **Choosing Between Workspaces and AppStream**: The decision depends on whether users need a full desktop environment with multiple applications (Workspaces) or just one or two applications (AppStream).

These services enable you to deliver client interfaces from the cloud, either as full desktops or individual applications, providing flexibility and scalability for different use cases.

Top of Form**ElastiCache**

* **ElastiCache Overview**: ElastiCache is an AWS service that provides in-memory caching for databases to improve performance. It helps in retrieving information quickly by storing data in memory rather than on disk.
* **Types of ElastiCache**:
  + **Memcached**: This is the simpler option for caching. It is suitable when you don't have specific compliance or security requirements. It offers high performance due to fewer processing requirements.
  + **Redis**: This option includes additional compliance capabilities, making it suitable for use cases requiring HIPAA or PCI-DSS compliance. Although it has slightly more processing overhead than Memcached, it ensures regulatory compliance.
* **Implementation**: When setting up ElastiCache, you create a cluster of caching servers. You need to choose the appropriate instance type with enough memory to cache the required data. The choice between Memcached and Redis depends on whether compliance is needed.
* **Usage**: ElastiCache is used to cache database queries, which reduces the time taken to retrieve data by keeping it in memory. This is particularly useful for applications requiring quick data access.

**Security services lab**

* **IAM (Identity and Access Management)**: This service allows you to manage users, groups, roles, and policies. It includes creating and managing encryption keys through the Key Management Services (KMS).
* **Key Management Services (KMS)**: This service is used to create, manage, and rotate encryption keys. It helps in securing your data by providing tools for key creation, management, and deletion.
* **CloudHSM (Hardware Security Module)**: This is a virtual hardware security module that performs encryption operations. It offloads encryption tasks from your applications, improving performance and security.
* **Directory Services**: AWS provides options to implement directory services in the cloud, such as AWS Managed Microsoft AD (Active Directory) and Simple AD. These services help manage network resources like users, groups, and devices.
* **Security AMIs (Amazon Machine Images)**: The AWS marketplace offers pre-configured security solutions that can be launched within your VPCs. These include various security tools and operating systems like Kali Linux.

These concepts cover the essential security services and tools discussed in the video, helping you understand how to manage and secure your AWS environment effectively.

Top of Form

Bottom of Form

Top of Form**Analytics engines**

* **CloudSearch**: This service allows you to create a search domain where you can upload data (like S3 buckets) and perform searches. It's useful for making offline data searchable within AWS.
* **Elasticsearch**: Similar to CloudSearch but more scalable. It can grow and shrink based on your needs, making it suitable for large-scale search operations.
* **Data Pipeline**: This service helps you manage data workflows. You can define data nodes (like S3 buckets or databases), schedule compute activities, and monitor the data flow to analyze it at different stages.
* **AWS Glue**: An ETL (Extract, Transform, Load) tool that lets you pull data from a source, transform it, and load it into a destination. It's great for manipulating and combining data from different sources.
* **QuickSight**: A business analytics service that provides advanced analytics out of the box. Note that it requires a separate subscription.
* **Athena**: Allows you to query data stored in S3 buckets using SQL. This is particularly useful if you are familiar with SQL and want to leverage that skill to analyze data in S3.

These tools help you analyze and manage data efficiently within AWS, each serving different purposes based on your specific needs.

**Development operations (DevOps)**

* **CodeCommit**: This is a source code management tool similar to git repositories. It allows you to store, manage, and version your source code.
* **CodeBuild**: This tool is used to compile your source code, run tests, and produce software packages that are ready to deploy. It supports various programming languages like Ruby, Node.js, Java, and .NET.
* **CodeDeploy**: This tool automates the deployment of your application to various AWS services like EC2 instances. It ensures that your application is deployed consistently across your development, test, and production environments.
* **CodePipeline**: This is a continuous integration and continuous delivery (CI/CD) service for fast and reliable application and infrastructure updates. It helps you automate the build, test, and deploy phases of your release process.
* **Cloud9**: An Integrated Development Environment (IDE) in the cloud, which allows you to write, run, and debug your code with just a browser. It provides a terminal with a Linux shell interface and supports various programming languages.
* **X-Ray**: This tool helps you analyze and debug your applications, providing insights into performance issues and errors.

These tools collectively help manage the development lifecycle from coding to deployment, ensuring efficiency and consistency in your DevOps processes.

Top of Form

**AWS Solutions**

* **AWS Solutions**: These are pre-built solutions created by professional architects and engineers to help you deploy various architectures in your AWS account easily. They save you time and effort by providing well-architected templates.
* **Types of Solutions**: You can find solutions for different needs such as data lakes, live streaming, centralized logging, real-time IoT device monitoring, and media analysis. These solutions are categorized by technology (e.g., analytics, blockchain) and industry (e.g., healthcare, digital marketing).
* **Filtering Options**: You can filter solutions by technology category or industry to find the most relevant ones for your needs. Additionally, you can choose solutions that use only AWS services to manage costs effectively.
* **AI-Driven Social Media Dashboard Example**: This solution involves several AWS services:
  + **Amazon EC2**: Receives raw tweets.
  + **Amazon Kinesis Data Firehose**: Delivers raw tweets to an S3 bucket.
  + **AWS Lambda**: Reads from the S3 bucket and uses Amazon Translate and Amazon Comprehend for analysis.
  + **Amazon Athena and QuickSight**: Used for reporting and data visualization.
* **Deployment Guide**: Each solution comes with a detailed deployment guide that includes an architectural overview, component descriptions, design considerations, and a CloudFormation template to automate deployment.
* **CloudFormation Templates**: These templates allow you to deploy the entire solution automatically, saving you the effort of manual setup. If needed, you can also modify or remove the solution easily.

These AWS Solutions are designed to be time-savers and provide well-engineered starting points for various architectural needs in AWS.

**AWS Transit Gateway**

* **AWS Transit Gateway**: This service simplifies the management of multiple Virtual Private Clouds (VPCs) and remote networks by acting as a central hub. Instead of creating individual VPC peering connections, you can connect all VPCs and remote networks through a single Transit Gateway.
* **Hub-and-Spoke Model**: The Transit Gateway operates like a hub in a hub-and-spoke network. Each VPC and remote network acts as a spoke connected to the central hub (Transit Gateway). This centralizes and simplifies network management.
* **Centralized Management**: With the Transit Gateway, you only need to manage one connection from the gateway to each VPC or remote network. This reduces the complexity and workload involved in managing multiple connections.
* **Scalability**: The Transit Gateway is particularly useful for large-scale deployments with many VPCs and remote networks, making it easier to scale and manage your network infrastructure.

**AWS Backup**

* **Centralized Backup Solution**: AWS Backup provides a centralized service to manage backups for various AWS services and on-premises data. Before AWS Backup, backups were managed individually for each service, but now it can all be managed in one place.
* **Fully Managed Service**: AWS Backup is fully managed, meaning you don't need to set up or maintain any infrastructure. You can start using it directly through the AWS Management Console.
* **Integration with AWS Services**: AWS Backup supports various AWS services, including RDS (databases), EC2 (instances), and Amazon Elastic File System (EFS). It also integrates with AWS CloudFormation for automated backups.
* **On-Premises Backup**: AWS Backup can also back up on-premises data using the AWS Storage Gateway, allowing you to have a unified backup strategy for both cloud and on-premises environments.
* **Cost Efficiency**: AWS Backup can use S3 Glacier and Deep Archive storage options, which are cost-effective for storing large amounts of data. For example, storing 100 terabytes can cost around $100 per month.

**AWS Cost Explorer**

* **Cost Management Tool**: AWS Cost Explorer is a tool that helps you manage and visualize your AWS costs. It provides detailed insights into your spending patterns.
* **Enabling Cost Explorer**: When you first enable Cost Explorer, it takes 24 hours to start displaying data. After this initial period, you can view historical data and track your costs in real-time.
* **Cost Overview**: The tool provides an overview of your costs, showing how much you have spent over a specific period. You can explore costs by service, account, or usage type.
* **Savings Plans**: Cost Explorer offers recommendations for savings plans based on your historical usage. These plans can help you save money by committing to a certain level of usage.
* **Budget Management**: You can create and manage budgets within Cost Explorer to help with financial planning and ensure you stay within your desired spending limits.
* **Recommendations**: The tool provides recommendations for cost optimization, such as switching to reserved instances to save money based on your usage patterns.

**Operational Excellence with AWSTop of Form**

**The operational excellence process**

* **Operational Excellence**: This involves preparing properly, operating effectively, and evolving as needed. It's about ensuring your AWS deployments are secure, reliable, performant, and cost-optimized.
* **AWS Well-Architected Framework**: This framework helps you design and operate reliable, secure, efficient, and cost-effective systems in the cloud. It includes five pillars: operational excellence, security, reliability, performance efficiency, and cost optimization.
* **Phases of Operational Excellence**:
  1. **Prepare**: Understand your workloads and expected behaviors. Identify your operational priorities and design systems to meet these priorities.
  2. **Operate**: Monitor your environment's health and respond to any issues. Use tools like CloudWatch to gain insights and improve performance.
  3. **Evolve**: Learn from your experiences and continuously improve your systems. Share insights within your team to enhance overall efficiency.

**Widget Makers scenario**

* **Understanding Existing Systems**: As an AWS architect, you need to gather detailed information about the current on-premises systems. This includes understanding the five most important systems: order processing, inventory management, payroll, user data storage, and the website.
* **Order Processing System**: This system is web-based with a SQL Server backend and 50-75 users. It's crucial to understand its data volume and usage patterns.
* **Inventory Management System**: This system uses MySQL and interacts with the order processing system to manage stock levels and perform reconciliations.
* **Payroll System**: This system includes a time clock function and interacts with the accounting side for payroll processing. It also stores basic HR information.
* **User Data Storage**: This involves storing user data on Windows Server shares, with around 700 megabytes per user and 250 active users.
* **Website**: The website is a WordPress system with custom plugins, handling around 3500 daily visits during the week and 600 on weekends.
* **Visualization Tools**: Tools like draw.io are used to create diagrams of the current systems and plan their migration to AWS.
* **AWS Well-Architected Framework**: The migration process focuses on the four pillars: resiliency, reliability, performance, and cost optimization. These principles guide the design and implementation of the cloud-based systems.

**Resilient design**

* **Resilient Design**: This refers to creating systems that are reliable and can automatically recover from failures without manual intervention. Automation is key for recovery, scaling, and backups.
* **Automation**: Essential for resilience, ensuring that recovery, scaling, and backups happen automatically. This reduces the need for manual intervention and enhances reliability.
* **Reliability Pillar**: AWS provides a reliability pillar document that outlines design principles for creating reliable systems. Key principles include testing recovery procedures, automatic failure recovery, horizontal scaling, and capacity planning.
* **Testing Recovery Procedures**: Regularly test your recovery procedures to ensure they work in disaster scenarios. This includes restoring from backups and running cloud formation launch templates.
* **Automatic Failure Recovery**: Use tools like CloudWatch to monitor the system and trigger automatic actions, such as launching more instances when needed.
* **Horizontal Scaling**: Decouple your application into different parts to increase system availability. This involves spreading the application across multiple instances rather than relying on a single server.
* **Capacity Planning**: Accurately determine your capacity needs based on actual usage data rather than guessing. This helps in planning the required resources in the cloud.
* **Managing Change with Automation**: Automate changes to the infrastructure as much as possible to ensure quick and reliable responses to needed changes.

Top of Form

**Resilient design scenario**

* **Order Processing System**: Continue using Microsoft SQL Server but implement it as an RDS instance for AWS to manage the database. Ensure resiliency by using a Multi-AZ (Availability Zone) database deployment.
* **Inventory Management System**: Use MySQL as an RDS instance with Multi-AZ deployment to ensure resiliency without requiring clustering.
* **Payroll System**: Use SQL Server in the Cloud as an RDS instance with Multi-AZ deployment. Implement a read replica to handle intensive read operations during payroll processing, enhancing performance without affecting the main database.
* **User Data Storage**: Move user data to S3 buckets for inherent resiliency. Use third-party tools to map drive letters to S3 buckets, as AWS does not provide this functionality by default.
* **Website**: Host the WordPress website using an elastic load-balanced deployment with two servers. This ensures resiliency by maintaining website availability even if one server fails.
* **Key Resiliency Strategies**:
  + Multi-AZ deployments for databases to ensure availability across multiple zones.
  + S3 buckets for resilient file storage.
  + Elastic load balancing for maintaining website availability.

**Performant design**

* **Resiliency vs. Performance**: Resiliency ensures the system is always available, while performance focuses on response times and efficiency in serving users.
* **Performance Efficiency Plan**: AWS provides a document called AWS-Performance-Efficiency-Pillar.pdf that outlines design principles for performance:
  + **Democratize Advanced Technologies**: Use AWS managed services like RDS databases and machine learning services to leverage pre-optimized solutions.
  + **Go Global in Minutes**: Deploy systems in multiple regions to reduce latency and improve performance by getting servers closer to users.
  + **Use Serverless Architectures**: Utilize services like Lambda and API Gateway to decouple applications and enable automatic scaling.
  + **Experiment More Often**: Take advantage of AWS's flexibility to test different configurations quickly and cost-effectively.
  + **Mechanical Sympathy**: Choose the right technology approach based on the specific needs of your processes.
* **Auto Scaling**: Automatically scale EC2 instances to handle varying loads, ensuring performance is maintained during high demand periods.
* **Performance Storage**: Use the appropriate storage type based on performance needs:
  + **Block Storage**: Low latency, single instance attachment.
  + **File System (EFS)**: Low latency, multiple instance attachment.
  + **Object Storage (S3)**: Low latency, scalable for web applications.
  + **Archival Storage (Glacier)**: High latency, high throughput for long-term storage.

These principles and strategies help ensure a well-performing AWS solution.

Top of Form**Performant design scenario**

* **Order Processing**: Ensure the right instances with sufficient memory and processing capabilities are selected for deployment to achieve the required performance.
* **Inventory Management**: Automate inventory management using SNS (Simple Notification Service) for notifications and consider using AWS CloudWatch to monitor inventory levels and automate reordering.
* **Payroll Solution**: Use instances with sufficient memory and processing capabilities and perform payroll processing from the read replica to enhance performance.
* **User Data Storage**: Implement departmental S3 buckets to handle the workload efficiently and configure alarms to notify administrators when users exceed storage limits.
* **Website Performance**: Ensure instances have sufficient memory and processing capabilities, use EBS (Elastic Block Store) volumes for maintaining state and enhancing performance, and ensure the right storage media type is used for optimal performance.
* **Internet Connection**: Ensure the organization's internet connection is sufficient to handle the performance needs of the AWS architecture, considering the number of users accessing the services.  
  Top of Form

**Secure design**

* **Implement a Strong Identity Foundation**: Set up proper roles, users, and groups in Identity and Access Management (IAM) and follow the principle of least privilege, giving users only the permissions they need.
* **Enable Traceability**: Use CloudTrail to monitor and log activities within AWS. Be selective about what you log to manage costs effectively.
* **Apply Security at All Layers**: Implement security measures at every level, including AWS account, VPC, subnet, instance, and within instances, following the defense-in-depth strategy.
* **Automate Security Best Practices**: Ensure secure AWS actions and code deployment by following best practices for AWS management and operating system security.
* **Protect Data in Transit and at Rest**: Use SSL and SSH for in-transit security and AWS encryption options for data at rest.
* **Keep People Away from Data**: Restrict direct access to data and use programmatic access through applications and web interfaces to minimize risks.
* **Prepare for Security Events**: Have a response plan and quick notification system in place using CloudTrail and alarms to handle security incidents promptly.
* **Shared Responsibility Model**: Understand that AWS is responsible for the security of the cloud infrastructure, while you are responsible for securing everything within the cloud, such as operating systems, S3 buckets, and databases.

**Secure design scenario**

* **Order Processing System**: Use IAM groups and policies to manage access to the SQL Server database. Ensure only authorized personnel can manage RDS instances and implement internal security features to control data access and modifications.
* **Inventory Management**: Similar to order processing, secure the database using IAM groups and policies. Implement internal security features to ensure only authorized changes to inventory levels.
* **Payroll System**: Secure database management through IAM groups and policies. Ensure only accounting employees can access the read replicas, and implement internal security features to control data processing and modifications.
* **User Data Storage**: Store user data in S3 buckets with appropriate security policies. Encrypt data at rest and use SSL for data transfers to ensure secure communication.
* **Website**: Run web server instances with roles that have limited access to required AWS resources. Configure security groups for network interfaces and the VPC to allow only necessary traffic, minimizing the risk of unauthorized access.

These points highlight the best practices for securing different systems within AWS, ensuring that only authorized users have access and that data is protected both at rest and in transit.

**Cost optimization**

* **Adopt a Consumption Model**: Pay only for the computing resources you consume. Scale down resources when not needed to save costs.
* **Measure Overall Efficiency**: Monitor the business output of your systems and the associated costs. Stop using systems that do not provide business value.
* **Stop Spending on Data Center Operations**: Use AWS to handle physical server management, reducing the need for costly data center operations.
* **Analyze and Attribute Expenditure**: Track where your AWS costs are coming from and ensure they align with business value. Reevaluate usage if costs are disproportionate to the value provided.
* **Use Managed Services**: Opt for managed services like RDS databases to reduce the cost of ownership and minimize the need for manual management.
* **Four Pillars of Cost Optimization**:
  + **Use Cost-Effective Resources**: Sometimes, more powerful instances that cost more per hour can be cheaper overall if they complete tasks faster.
  + **Match Supply with Demand**: Use auto-scaling to ensure you have the right number of servers based on demand.
  + **Expenditure Awareness**: Utilize AWS billing management features to track and get alerts on your spending.
  + **Optimize Over Time**: Continuously look for ways to improve performance, reliability, and security while minimizing costs.

Top of Form

**Cost optimization scenario**

* **Managed Databases**: Use managed databases for order processing, inventory management, and payroll to reduce operational hours and costs. For payroll, use read replicas only when needed to save on costs.
* **User Data Monitoring**: Monitor user data in S3 buckets to avoid unnecessary storage costs. Ensure only business-critical data is stored.
* **Website Optimization**: Choose the right instance class for your website to balance performance and cost. Monitor access to prevent misuse that could lead to high bandwidth fees.
* **Cost-Effective Design**: Implement a cost-optimized design by using Multi-AZ deployments, elastic load balancing, and appropriate technologies to ensure resiliency, performance, and security while minimizing costs.

These strategies help ensure a cost-efficient AWS deployment for the Widget Makers organization.

**General best practices**

* **Design for Failures**: Implement technologies like clustering and availability zones to ensure redundancy. Use backups and consider having an alternate AWS account as a cold or warm site for disaster recovery.
* **Implement Elasticity**: Take advantage of cloud computing's ability to scale resources up and down. Use auto-scaling, elastic load balancing, decoupled applications, and parallel task execution to optimize resource usage.
* **Continuous Learning**: Utilize AWS free tier accounts to practice building and tearing down solutions. Experiment with different configurations to learn cost optimization and improve your skills.

Top of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Bottom of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form