Teddy Wyatt

CS 470: Full Stack Development II

Southern New Hampshire University

Dec 10, 2023

Project 2 Script and Youtube link

Yoututbe Listing of Video: https://youtu.be/31lcT-1hcQ8

Hello everyone, my name is Teddy Wyatt, and I'm excited to take you on a journey from full-stack development to the world of AWS serverless computing.
In this presentation, we'll delve into the intricacies of cloud development, offering insights and inspiration for startup company developers eager to learn from our experiences.

Our cloud migration journey begins with containerization, a critical step in modernizing our application infrastructure
Containerization ensures consistent and reproducible environments for applications. It simplifies the development, testing, and deployment processes, making it easier to maintain consistency across different environments, including development, testing, and production. Containers share the host operating system's kernel, resulting in lower overhead compared to traditional virtual machines. This resource efficiency makes containerization well-suited for scenarios where maximizing resource utilization and achieving scalability are essential, such as microservices architectures and dynamic scaling of applications.
Note: While containerization offers flexibility and resource efficiency, it may not provide the same level of security as traditional virtualization or bare-metal installations. In cases where security is paramount, especially in classified government environments, traditional methods are preferred for their strong isolation and robust security to ensure the protection of sensitive data.
By using containers we were able to seamlessly package and transfer our application components to the cloud using lightweight containers, that use less processing then standard instances.

Docker serves as a robust platform for deploying and managing containerized applications in a DevOps environment. Containerization with Docker enables developers and operations teams to work together seamlessly throughout the application lifecycle.
Key Steps in DevOps Deployments with Docker:

Teddy Wyatt

CS 470: Full Stack Development II

Southern New Hampshire University

Dec 10, 2023

Project 2 Script and Youtube link

- **Creating Docker Images**: The first step involves defining application dependencies and configurations in a Dockerfile and using the Docker build command to create a Docker image.
- **Image Registry**: Docker images need to be stored in a registry, such as Docker Hub, Amazon ECR, Google Container Registry, or Azure Container Registry, for easy access and deployment.
- **Container Orchestration**: Container orchestration tools like Kubernetes, Docker Swarm, and Amazon ECS are used to manage and deploy containers at scale.

Now, let's explore the serverless cloud. Serverless computing means leaving server management to the cloud provider, like AWS. We took advantage of AWS services like S3 storage, which provides scalable and cost-effective storage. No more worrying about physical drives – it's all in the cloud.
Lambda is a serverless computing model in which developers write code in the form of functions, and the cloud provider (e.g., AWS Lambda) handles all aspects of the underlying infrastructure, including provisioning, scaling, and maintenance.
Well-suited for event-driven applications and microservices architecture, Lambda allows automatic scaling based on workload without developers needing to handle server provisioning. It abstracts away infrastructure concerns, making it efficient for specific use cases.
Compared to Traditional Container-Based Computing
Traditional container-based computing, exemplified by Docker and Kubernetes, provides developers with more control over the application environment. Containers encapsulate applications and dependencies, ensuring consistency across various environments.
Containers offer portability and are suitable for complex applications with multiple components and dependencies. However, managing container orchestration, scaling, and infrastructure can be more complex compared to the Lambda model.

- **Cloud-Based, Scalable Storage**: S3 is a cloud-based service by Amazon Web Services (AWS) that offers scalable, durable, and globally accessible storage suitable for various use cases like data backup, archiving, and content distribution.
- **Advanced Features**: S3 provides features like versioning, access control, and replication across multiple data centers for high durability and data availability.

Teddy Wyatt

CS 470: Full Stack Development II

Southern New Hampshire University

Dec 10, 2023

Project 2 Script and Youtube link

- **Global Accessibility**: S3's global accessibility and content distribution capabilities make it invaluable for serving content to a worldwide audience, ensuring high availability, and supporting disaster recovery across multiple regions. However, it may have associated cost considerations and relies on internet connectivity for data access.

Local Disk Storage:

- **Physical Storage**: Local disk storage involves physical hard drives or SSDs connected to a local server or computer, providing low-latency data access.
- **Limited Scalability and Durability**: While suitable for applications requiring fast local data processing, local disk storage lacks the scalability and durability of cloud-based services like S3.
- **Local Accessibility**: It offers local accessibility but doesn't provide the global accessibility and redundancy found in cloud-based solutions.

Our serverless API was powered by AWS Lambda, allowing us to run code without managing servers. Lambda's event-driven model enabled us to develop API logic effortlessly. We connected Lambda functions to our API methods and ensured smooth communication between our frontend and backend.

Scalability and High Availability

- Amazon API Gateway offers scalability and high availability capabilities.
- It can handle traffic spikes and distribute requests across multiple endpoints automatically.
- This ensures that APIs remain responsive and available even during periods of high demand, making it valuable for applications that need to adapt to changing user loads without manual intervention.

- Robust Traffic Management
  - Amazon API Gateway provides robust traffic management features, including caching, request throttling, and request/response transformation.
  - These features allow developers to fine-tune how their APIs are accessed, improving performance, reducing latency, and optimizing resource utilization.
  - Fine-grained traffic control enhances the efficiency and responsiveness of APIs.
- Customization and Serverless Integration
  - Amazon API Gateway supports integration with AWS Lambda functions, enabling serverless execution of code in response to API requests.

Teddy Wyatt

CS 470: Full Stack Development II

Southern New Hampshire University

Dec 10, 2023

Project 2 Script and Youtube link

- This serverless approach simplifies infrastructure management and reduces operational overhead, allowing developers to focus on code and features.
- Customization features through API stages facilitate version management and updates without disrupting existing users, ensuring a smooth and customizable development experience.
- Security Features
- Security is a paramount concern, and Amazon API Gateway offers various security features.
- It supports authentication and authorization mechanisms, including AWS IAM, OAuth 2.0, and custom authorizers, enabling control over API access.
- Additionally, it provides protection against common web exploits like SQL injection and XSS through request validation and output encoding, ensuring API security and compliance with best practices.

Database choices are crucial. We compared MongoDB and DynamoDB. While MongoDB offers versatility, DynamoDB integrates seamlessly with AWS services. We performed various queries and developed Lambda functions to interact with our chosen database, ensuring data retrieval and storage were efficient.

1. MongoDB:
   - **Document-Oriented Database**: MongoDB is a document-oriented database that offers flexible schema design.
   - **JSON-like Format (BSON)**: It stores data in a JSON-like format called BSON and supports complex queries.
   - **Use Cases**: MongoDB is typically used for applications with rapidly changing data structures, where flexibility and ease of development are important, such as web and mobile applications.
2. DynamoDB:
   - **Key-Value and Document Database**: DynamoDB, offered by AWS, is designed for high availability, scalability, and low-latency performance.
   - **Table-Based Data Model**: It uses a structured table-based data model, where data is organized into tables with primary keys and optional secondary indexes.
   - **Use Cases**: DynamoDB is suitable for applications that require predictable and consistent performance at scale, such as e-commerce, gaming, and IoT applications.
3. Single-Table Model in DynamoDB

Teddy Wyatt

CS 470: Full Stack Development II

Southern New Hampshire University

Dec 10, 2023

Project 2 Script and Youtube link

- **Efficiency and Cost-Effectiveness**: The single-table model in DynamoDB involves designing your data model within a single table, often using composite primary keys and secondary indexes.
- **Benefits**: It can be highly efficient and cost-effective for many use cases, minimizing the need for complex JOIN operations and reducing the number of tables.
- **Challenges**: However, it can be challenging to design and maintain, especially for applications with complex and evolving data structures. Its suitability depends on specific application requirements and team expertise.

4.Considerations for Single-Table Model:

- **Appropriateness**: The single-table model in DynamoDB can be a powerful tool when used appropriately.
- **Not Universally Applicable**: It may not be the best choice for all situations, and careful planning is essential to realize its benefits effectively.
- **Application and Expertise**: Whether it's a good fit depends on your specific application needs and your team's expertise, and it should be chosen accordingly.

Cloud-based development principles bring flexibility. AWS, with its pay-for-use model, ensures we only pay for the resources we consume. We no longer worry about wasted computing power. The cloud scales with our needs, making our application cost-effective and adaptable.

Security is paramount in the cloud. AWS Identity and Access Management (IAM) helps us control who accesses our resources. We created custom IAM policies, securing access to our database and API. We also ensured that data remained encrypted both in transit and at rest.

Access and IAM Roles/Policies Setup:

- **Identify Specific Needs**: Start by identifying the specific requirements of each AWS service used in your organization, such as EC2, S3, or Lambda, as each service requires different permissions.
- **Tailored IAM Roles**: Create IAM roles tailored to these services using the AWS IAM console.
- **Attach Appropriate Policies**: Attach the appropriate policies to these roles. AWS provides managed policies for common use cases, and you can also create custom policies for more specific requirements.

Teddy Wyatt

CS 470: Full Stack Development II

Southern New Hampshire University

Dec 10, 2023

Project 2 Script and Youtube link

- **Permission Definition**: Define permissions carefully, considering actions, resources, and conditions relevant to each service.

Ongoing Security and Relevance:

- **Regular Review and Audit**: Ensure ongoing security and relevance by regularly reviewing and auditing the roles and policies associated with AWS services.
- **AWS Tools for Security**: Utilize AWS tools like IAM Access Analyzer and AWS Trusted Advisor to identify security gaps and optimize configurations.
- **Balancing Security and Efficiency**: This practice helps maintain a secure and efficient environment, ensuring that each service has the necessary permissions without compromising security.

Cross Origin Resource Sharing (CORS) is **a W3C standard that allows a server to relax the same-origin policy**. Using CORS, a server can explicitly allow some cross-origin requests while rejecting others.

In conclusion, cloud-based development empowers us with flexibility, scalability, and cost-efficiency. Our journey from a full-stack application to the AWS serverless cloud has enabled us to achieve remarkable results. As you embark on your cloud development journey, remember that AWS simplifies collaboration, secures your assets, and optimizes resource usage.
Thank you for joining me today. If you have any questions or want to learn more about our migration process, feel free to ask.