

AWS Lift and Shift



Lift and Shift Summary Process.pdf
PDF

Extract only important information out of it



Lift and Shift Summary Process: Important Information Extracted

Project Overview

- **Objective:** Migrate the vProfile multi-tier web application from a local data center to AWS using a lift and shift strategy.
- **Goal:** Learn how to run application workloads on AWS cloud.

Background

- **Previous Setup:** Multi-tier Web application (vProfile) hosted using Vagrant.
- **New Setup:** Host and run the vProfile application on AWS for production.

Current Environment

- **Services:** Databases (Postgres, Oracle), Application services (Tomcat, LAMP Stack), DNS services, etc.
- **Challenges:** High complexity, multiple teams required, manual processes, high costs, difficult scaling.
- **Solution:** Cloud computing setup (AWS).

Benefits of Cloud Computing

- **Pay-as-you-go model**
- **Elasticity and automation**
- **Reduced management complexity and cost control**

AWS Services Used

- **Compute:** EC2 instances for Tomcat, RabbitMQ, Memcache, and MySQL servers.
- **Load Balancing:** Elastic Load Balancer (ELB) to replace Nginx service.
- **Scaling:** Auto Scaling service to manage EC2 instances based on load.
- **Storage:** S3 or EFS for storage.
- **DNS:** Route 53 for private DNS services.
- **Security:** IAM for access management, ACM for certificates, EBS for block storage.
- **Automation:** Infrastructure as Code (IaC).

Project Objectives

1. Create a flexible infrastructure.
2. Utilize a pay-as-you-go model.
3. Modernize application using AWS services.
4. Implement automation through Infrastructure as Code.

AWS Architectural Design

1. **User Access:** Users access the application via a URL pointing to an endpoint in GoDaddy DNS.
2. **Load Balancer:** Uses HTTPS encryption with certificates from ACM. Allows only HTTPS traffic.
3. **Application Servers:** Requests routed to Tomcat instances managed by an auto-scaling group. Instances in a security group allowing traffic on port 8080 from the load balancer only.
4. **Backend Services:** Tomcat accesses backend servers (MySQL, Memcache, RabbitMQ) using private DNS names in Route 53. Backend EC2 instances in separate security groups.

Execution Flow

1. **Login to AWS:** Access AWS account and create key pairs for EC2 instance access.
2. **Security Groups:** Create security groups for load balancer, Tomcat, and backend services.
3. **Launch Instances:** Deploy EC2 instances using user data (bash scripts).
4. **Route 53 Setup:** Update IP to name mapping in Route 53.
5. **Build Application:** Compile the application source code on a local machine.
6. **Artifact Storage:** Upload the artifact to an S3 bucket and download it to the Tomcat EC2 instance.
7. **Load Balancer Setup:** Configure the load balancer with HTTPS.
8. **DNS Configuration:** Map the ELB endpoint to the website in GoDaddy DNS.
9. **Verification:** Verify the setup and build an auto-scaling group for Tomcat instances.

Next Steps

- **Action:** Dive into AWS and start the execution process.

These notes provide a detailed summary of the project, highlighting the main objectives, architectural design, and execution steps for better understanding and revision.



You have not included everything out of the pdf



Lift and Shift Summary Process: Comprehensive Extracted Information

Project Overview

- **Objective:** Migrate the vProfile multi-tier web application from a local data center to AWS using a lift and shift strategy.
- **Goal:** Learn how to run application workloads on AWS cloud.

Background

- **Previous Setup:** Multi-tier Web application (vProfile) hosted using Vagrant.
- **New Setup:** Host and run the vProfile application on AWS for production.

Scenario

- **Current Environment:** Application services running on physical/virtual machines in a data center.
- **Services Include:** Databases (Postgres, Oracle), Application services (Tomcat, LAMP Stack), DNS services, etc.
- **Challenges:** High complexity, multiple teams required, manual processes, high costs, difficult scaling.
- **Solution:** Cloud computing setup (AWS).

Benefits of Cloud Computing

- **Pay-as-you-go model**
- **Elasticity and automation**
- **Reduced management complexity**
- **Cost control**

AWS Services Used

- **Compute:** EC2 instances for Tomcat, RabbitMQ, Memcache, and MySQL servers.
- **Load Balancing:** Elastic Load Balancer (ELB) to replace Nginx service.
- **Scaling:** Auto Scaling service to manage EC2 instances based on load.
- **Storage:** S3 or EFS for storage.
- **DNS:** Route 53 for private DNS services.
- **Security:** IAM for access management, ACM for certificates, EBS for block storage.
- **Automation:** Infrastructure as Code (IaC).

Project Objectives

1. Create a flexible infrastructure.
2. Utilize a pay-as-you-go model.
3. Modernize application using AWS services.
4. Implement automation through Infrastructure as Code.

AWS Architectural Design

1. **User Access:** Users access the application via a URL pointing to an endpoint in GoDaddy DNS.
2. **Load Balancer:** Uses HTTPS encryption with certificates from ACM. Allows only HTTPS traffic.
3. **Application Servers:** Requests routed to Tomcat instances managed by an auto-scaling group. Instances in a security group allowing traffic on port 8080 from the load balancer only.
4. **Backend Services:** Tomcat accesses backend servers (MySQL, Memcache, RabbitMQ) using private DNS names in Route 53. Backend EC2 instances in separate security groups.

Execution Flow

1. **Login to AWS:** Access AWS account and create key pairs for EC2 instance access.
2. **Security Groups:** Create security groups for load balancer, Tomcat, and backend services.
3. **Launch Instances:** Deploy EC2 instances using user data (bash scripts).

4. **Route 53 Setup:** Update IP to name mapping in Route 53.
5. **Build Application:** Compile the application source code on a local machine.
6. **Artifact Storage:** Upload the artifact to an S3 bucket and download it to the Tomcat EC2 instance.
7. **Load Balancer Setup:** Configure the load balancer with HTTPS.
8. **DNS Configuration:** Map the ELB endpoint to the website in GoDaddy DNS.
9. **Verification:** Verify the setup and build an auto-scaling group for Tomcat instances.

Detailed Steps for AWS Setup

1. **AWS Account Login:**
 - Log in to AWS.
 - Ensure North Virginia region.
 - Create and validate an HTTPS certificate via GoDaddy.
2. **Security Groups Creation:**
 - Load Balancer Security Group: Allow port 80 (HTTP) and port 443 (HTTPS).
 - Tomcat Instance Security Group: Allow port 8080 from the load balancer.
 - Backend Services Security Group: Allow ports 3306 (MySQL), 11211 (Memcache), and 5672 (RabbitMQ) from the application security group.
3. **EC2 Instances Deployment:**
 - Launch EC2 instances with user data (bash scripts).
 - Use key pairs for SSH access.
4. **Route 53 Configuration:**
 - Update IP-to-name mapping for backend servers.
 - Use private DNS zones for backend services.
5. **Application Build and Deployment:**
 - Build the application source code on a local machine.
 - Upload artifacts to an S3 bucket.
 - Download artifacts to the Tomcat EC2 instance.
6. **Load Balancer and DNS Setup:**
 - Configure the load balancer with HTTPS.
 - Map the ELB endpoint to the website in GoDaddy DNS.
7. **Verification and Auto-Scaling:**
 - Verify the entire setup.
 - Create an auto-scaling group for Tomcat instances.

Conclusion

- **Action:** Dive into AWS and start the execution process.

These notes provide a detailed summary of the project, highlighting the main objectives, architectural design, and execution steps for better understanding and revision.



Explain how and why did we use this services



Explanation of AWS Services Used and Their Purpose

1. Compute: EC2 Instances

- **Why:** EC2 (Elastic Compute Cloud) provides scalable computing capacity in the cloud. It allows for easy deployment and management of applications.
- **How:** EC2 instances are used to run Tomcat, RabbitMQ, Memcache, and MySQL servers. These instances can be configured and scaled according to the application's needs.

2. Load Balancing: Elastic Load Balancer (ELB)

- **Why:** ELB automatically distributes incoming application traffic across multiple targets, such as EC2 instances. It improves fault tolerance and ensures high availability.
- **How:** ELB is configured to replace the existing Nginx service, handling incoming HTTPS traffic and distributing it to Tomcat instances. This ensures efficient load management and better user experience.

3. Scaling: Auto Scaling

- **Why:** Auto Scaling helps maintain application availability and allows for automatic scaling of EC2 instances based on predefined conditions.
- **How:** An Auto Scaling group is set up to manage the number of Tomcat instances. This group adjusts the number of running instances based on the load, ensuring that the application can handle varying traffic levels efficiently.

4. Storage: S3 (Simple Storage Service) or EFS (Elastic File System)

- **Why:** Both S3 and EFS provide scalable storage solutions with high durability, availability, and performance.
- **How:**
 - **S3:** Used for storing build artifacts, backups, and other static content. Artifacts are uploaded to an S3 bucket and downloaded to the Tomcat EC2 instances.
 - **EFS:** Could be used for shared storage needs across multiple instances, such as shared configuration files or logs.

5. DNS: Route 53

- **Why:** Route 53 is a scalable and highly available Domain Name System (DNS) web service. It provides reliable routing and health checks.
- **How:** Route 53 is used to manage private DNS services for backend servers and public DNS for the application. It ensures that requests are routed efficiently to the appropriate instances.

6. Security: IAM (Identity and Access Management), ACM (AWS Certificate Manager), EBS (Elastic Block Store)

- **Why:**
 - **IAM:** Manages user access and permissions securely.
 - **ACM:** Provides and manages SSL/TLS certificates for use with AWS services and internal connected resources.
 - **EBS:** Provides block storage volumes for use with EC2 instances, ensuring persistent data storage.
- **How:**
 - **IAM:** Used to create roles and policies for secure access management.

- **ACM:** Used to provision and manage HTTPS certificates for secure communication.
- **EBS:** Used for persistent storage needs, such as database storage for MySQL servers.

7. Automation: Infrastructure as Code (IaC)

- **Why:** IaC allows for the automated management and provisioning of technology infrastructure through code, improving efficiency and reducing manual errors.
- **How:** Using tools like AWS CloudFormation or Terraform, the entire infrastructure can be defined and managed as code. This includes the creation of EC2 instances, security groups, load balancers, and more.

Summary of Why Each Service is Used

- **EC2 Instances:** For scalable and flexible compute resources.
- **Elastic Load Balancer:** To ensure high availability and efficient traffic distribution.
- **Auto Scaling:** To automatically adjust the number of EC2 instances based on demand.
- **S3/EFS:** For reliable and scalable storage solutions.
- **Route 53:** For efficient DNS management and routing.
- **IAM, ACM, EBS:** For secure access, SSL/TLS management, and persistent storage.
- **IaC:** For automation and efficient infrastructure management.

These services collectively provide a robust, scalable, and secure environment for running the vProfile application on AWS, leveraging the benefits of cloud computing.



Create project flow diagram



Creating a project flow diagram for the Lift and Shift process involves detailing the steps involved in migrating the vProfile application to AWS. Here's a textual representation of the flow which you can visualize using a diagramming tool like Microsoft Visio, Lucidchart, or even hand-drawn sketches:

Project Flow Diagram

- 1. Project Initiation**
 - Define Objectives
 - Gather Requirements
- 2. AWS Account Setup**
 - Log in to AWS
 - Set up Billing and Access Controls
- 3. AWS Certificate Manager (ACM) Setup**
 - Request and Validate HTTPS Certificate via GoDaddy
- 4. Security Groups Creation**
 - Create Load Balancer Security Group (Allow HTTP/HTTPS)
 - Create Tomcat Instance Security Group (Allow traffic from Load Balancer on port 8080)
 - Create Backend Services Security Group (Allow specific ports for MySQL, Memcache, RabbitMQ)
- 5. Launch EC2 Instances**

- Create Key Pairs for SSH access
- Deploy EC2 instances with user data (bash scripts for configuration)

6. Configure Route 53

- Update IP-to-Name Mapping
- Configure Private DNS Zones for Backend Services

7. Build and Deploy Application

- Compile Application Source Code locally
- Upload Artifacts to S3
- Download Artifacts to Tomcat EC2 Instance

8. Elastic Load Balancer Setup

- Configure ELB with HTTPS
- Ensure Traffic Distribution among Tomcat Instances

9. DNS Configuration in GoDaddy

- Map ELB Endpoint to Website

10. Verification

- Check Setup
- Test Application Functionality

11. Setup Auto Scaling Group

- Configure Auto Scaling Policies
- Ensure Application Scalability

12. Final Validation

- Full System Testing
- User Acceptance Testing

13. Project Completion

- Documentation
- Training
- Go-Live

Diagram Visualization (Description)

- **Start Node:** Project Initiation
 - Connect to **AWS Account Setup**
 - Connect to **ACM Setup**
- **AWS Account Setup:**
 - Connects to **Security Groups Creation**
 - Connects to **Launch EC2 Instances**
- **ACM Setup:**
 - Connects to **Elastic Load Balancer Setup**
- **Security Groups Creation:**
 - Connects to **Launch EC2 Instances**
 - Connects to **Configure Route 53**
- **Launch EC2 Instances:**
 - Connects to **Build and Deploy Application**
 - Connects to **Elastic Load Balancer Setup**
- **Configure Route 53:**
 - Connects to **Build and Deploy Application**
- **Build and Deploy Application:**
 - Connects to **Elastic Load Balancer Setup**
- **Elastic Load Balancer Setup:**

- Connects to **DNS Configuration in GoDaddy**
- **DNS Configuration in GoDaddy:**
 - Connects to **Verification**
- **Verification:**
 - Connects to **Setup Auto Scaling Group**
- **Setup Auto Scaling Group:**
 - Connects to **Final Validation**
- **Final Validation:**
 - Connects to **Project Completion**
- **End Node:** Project Completion

This flow can be used to create a visual diagram using any diagramming tool, with each step represented as a node and arrows indicating the flow from one step to the next.

ChatGPT can make mistakes. Check important info.