AWS – Project

**Project 1 : Deploy a WordPress Website on AWS.**

Deploying a WordPress website on AWS involves several steps, including setting up an EC2 instance, configuring a database, installing WordPress, and configuring DNS settings. Here's a basic guide to get you started:

**1 Set up an EC2 Instance**:

Log in to your AWS Management Console.

Go to the EC2 dashboard and launch a new EC2 instance. Choose an Amazon Linux or Ubuntu Server instance.

Configure security groups to allow inbound traffic on ports 80 (HTTP) and 443 (HTTPS) for web traffic.

**2 Install LAMP Stack** (Linux, Apache, MySQL, PHP):

Connect to your EC2 instance using SSH.

Update the package index and install Apache, MySQL, PHP, and other necessary packages.

Configure Apache and PHP settings as required for WordPress.

Set up MySQL database and user for WordPress.

**3 Download and Install WordPress**:

Download the latest version of WordPress from the official website.

Extract the WordPress archive and move it to the appropriate directory on your EC2 instance.

Configure the WordPress configuration file (wp-config.php) with the database details you created earlier.

**4 Set Up DNS**:

Go to your domain registrar and point your domain's DNS records to the public IP address of your EC2 instance.

Optionally, you can set up a subdomain or use a service like Amazon Route 53 for DNS management.

**5 Configure Apache Virtual Host**:

Create a virtual host configuration file for your WordPress site in Apache.

Configure the virtual host to point to the directory where WordPress is installed.

**6 Secure your WordPress Installation**:

Install an SSL certificate to enable HTTPS. You can use a free certificate from Let's Encrypt.

Configure WordPress security settings and install security plugins to protect against common threats.

**7 Additional Steps** (Optional):

Set up backups for your WordPress site and database.

Configure caching and optimization plugins for better performance.

Set up monitoring and alerts for your AWS resources.

**8 Test your WordPress Site**:

Access your domain in a web browser to ensure that WordPress is installed and working correctly.

Log in to the WordPress admin dashboard to customize your site and add content.

Remember to regularly update WordPress, themes, and plugins to keep your site secure and up to date. Additionally, consider implementing regular backups and security measures to protect your site from potential threats.

**Project 2 : Server less Web Application on AWS**

Creating a serverless web application on AWS typically involves using services like AWS Lambda, Amazon API Gateway, Amazon S3, AWS DynamoDB or Amazon Aurora (for database needs), and AWS Amplify or AWS CloudFront for content delivery and hosting. Here's a basic outline of how you can create a serverless web application on AWS:

**1 Architecture Design**:

Plan the architecture of your serverless web application, including the services you'll use and how they'll interact with each other.

**2 Frontend Development**:

Develop the frontend of your web application using HTML, CSS, and JavaScript frameworks like React.js, Angular, or Vue.js.

You can use AWS Amplify to easily integrate AWS services into your frontend application and manage authentication, authorization, and data storage.

**3 Backend Development**:

Write backend logic using AWS Lambda, which allows you to run code without provisioning or managing servers.

Define RESTful APIs using Amazon API Gateway to expose your Lambda functions as HTTP endpoints.

**4 Data Storage**:

Choose a data storage solution based on your requirements. For example:

Use Amazon S3 for storing static assets like images, videos, and frontend files.

Use AWS DynamoDB for NoSQL database needs or Amazon Aurora Serverless for relational database needs.

**5 Authentication and Authorization**:

Implement user authentication and authorization using AWS Cognito or a custom solution integrated with AWS Lambda and other AWS services.

**6 Deployment**:

Deploy your frontend assets to Amazon S3, which can also act as a static website hosting service.

Deploy your Lambda functions and API Gateway configurations using AWS SAM (Serverless Application Model) or the AWS Management Console.

**7 Domain Configuration**:

Configure a custom domain for your serverless web application using Amazon Route 53.

Set up SSL/TLS certificates using AWS Certificate Manager to enable HTTPS.

**8 Monitoring and Logging**:

Set up monitoring and logging using Amazon CloudWatch to track the performance and health of your serverless application.

Configure alarms and notifications for critical metrics and events.

**9 Testing and Optimization**:

Test your serverless web application thoroughly to ensure it works as expected.

Optimize performance and cost by fine-tuning configurations, optimizing Lambda function code, and using AWS Cost Explorer to analyze costs.

**10 Maintenance and Updates**:

Regularly update your application to add new features, fix bugs, and improve security.

Monitor AWS service announcements for new features and updates that could benefit your application.

By following these steps, you can create a scalable, cost-effective, and low-maintenance serverless web application on AWS.

**Project 3 : Amazon Connect Call Center on AWS**

Setting up an Amazon Connect call center on AWS involves several steps, from creating the Amazon Connect instance to configuring call flows and integrating with other AWS services. Here's a basic guide to get started:

**1 Create an Amazon Connect Instance**:

Log in to your AWS Management Console and navigate to the Amazon Connect service.

Click on "Get started" and follow the wizard to create a new Amazon Connect instance.

Choose the appropriate settings for your instance, including the AWS region, instance alias, and administrator credentials.

**2 Configure Users and Permissions**:

Set up users and assign them appropriate permissions within Amazon Connect.

Define user roles to control access to features and functionalities within the call center.

**3 Design Contact Flows**:

Create contact flows to define the customer experience when they interact with your call center.

Design IVR (Interactive Voice Response) menus, queues, and routing profiles to handle incoming calls efficiently.

Customize contact flows using the Amazon Connect contact flow editor.

**4 Integrate with AWS Services**:

Integrate Amazon Connect with other AWS services to enhance functionality and data analytics.

For example, integrate with Amazon Lex for conversational bots, Amazon Polly for text-to-speech, Amazon S3 for call recording storage, and Amazon CloudWatch for monitoring.

**5 Configure Phone Numbers**:

Obtain phone numbers for your call center and configure them within Amazon Connect.

Assign phone numbers to specific contact flows or queues as needed.

**6 Set Up Queues and Routing**:

Create queues to organize and prioritize incoming calls based on criteria such as customer needs or agent skills.

Configure routing profiles to determine how calls are routed to agents based on queue priority, agent availability, or other conditions.

**7 Agent Management**:

Add agents to your call center and assign them to specific queues or routing profiles.

Configure agent availability hours and set up skills-based routing to match callers with the most appropriate agents.

**8 Testing and Monitoring**:

Test your call center configuration to ensure that contact flows, queues, and routing behave as expected.

Monitor call center performance using Amazon CloudWatch metrics and logs.

Use real-time metrics and historical reports to track key performance indicators (KPIs) and identify areas for improvement.

**9 Training and Support**:

Provide training to agents and supervisors on using the Amazon Connect interface and handling calls effectively.

Utilize AWS documentation, forums, and support resources for assistance with troubleshooting and optimization.

**10 Scale and Iterate**:

Monitor call center performance over time and make adjustments as needed to improve efficiency and customer satisfaction.

Scale your Amazon Connect instance as your call center grows, adding additional agents, phone numbers, and features as required.

By following these steps, you can set up and configure an Amazon Connect call center on AWS to provide efficient and reliable customer service through voice interactions.