Practical-5 Platform as a service using AWS

Anshika Sharma

86062300034

A061

Platform as a Service (PaaS)

Platform as a Service (PaaS) is a cloud computing model that provides a fully managed platform for developing, running, and managing applications. PaaS solutions handle the underlying infrastructure, operating system, and middleware, allowing developers to focus solely on application development without worrying about server management, networking, or storage.

PaaS offers several advantages:

1. **Speed and Efficiency**: PaaS accelerates application development by providing pre-built frameworks, tools, and libraries, reducing the need for setup and configuration.
2. **Scalability**: PaaS platforms automatically handle scaling based on application demands, providing flexibility and cost-effectiveness.
3. **Support for Collaboration**: Developers can collaborate seamlessly as PaaS environments are accessible from anywhere with the right credentials, enabling remote and cross-functional teamwork.

Examples of popular PaaS solutions include AWS Elastic Beanstalk, Google App Engine, and Microsoft Azure App Service.

AWS Elastic Beanstalk

AWS Elastic Beanstalk is a PaaS offering from Amazon Web Services that allows developers to quickly deploy, manage, and scale applications without needing to manage the underlying infrastructure. Elastic Beanstalk supports a wide range of programming languages and frameworks, such as Java, Python, PHP, Node.js, Ruby, and Docker, making it flexible for various types of applications.

With Elastic Beanstalk, developers can simply upload their application code, and Beanstalk automatically handles deployment, load balancing, scaling, and monitoring. Elastic Beanstalk provides a fully managed environment that includes all the necessary resources for running applications.

Components of Elastic Beanstalk

Elastic Beanstalk consists of several components that work together to provide a fully managed application environment:

1. **Environment**: This is the core of Elastic Beanstalk, consisting of all the AWS resources needed to run an application. An environment includes an EC2 instance (or multiple instances if scaled), a load balancer, and an auto-scaling group.
2. **Application**: An application in Elastic Beanstalk is a collection of environments, versions, and configurations. Each application serves as a container for managing different deployment versions and configurations of the same application.
3. **Application Version**: This is a specific, deployable version of an application. Elastic Beanstalk allows developers to create, test, and deploy multiple versions, which can then be managed and used across different environments.
4. **Environment Configuration**: This is a set of parameters and settings defining how an environment and its resources behave. For example, you can configure environment variables, scaling policies, and load balancer settings to adjust the performance and cost of the environment.
5. **Environment Tier**: Elastic Beanstalk offers two environment tiers:
    - **Web Server Environment Tier**: Used for running web applications, Elastic Beanstalk automatically provisions an EC2 instance, load balancer, and autoscaling group.
    - **Worker Environment Tier**: Designed for applications that process background tasks, this tier provisions EC2 instances that handle jobs from an Amazon SQS (Simple Queue Service) queue.

These components allow Elastic Beanstalk to provide a flexible and comprehensive platform for managing application deployment and lifecycle.

IAM (Identity and Access Management) in Elastic Beanstalk

Identity and Access Management (IAM) is a critical part of managing security within Elastic Beanstalk. IAM enables the control of permissions and access to resources, helping to ensure that only authorized users and applications can interact with the Elastic Beanstalk environment.

Role of IAM in Elastic Beanstalk:

1. **User Access and Permissions**:
    - IAM lets administrators define which users and groups have access to Elastic Beanstalk resources, ensuring security by limiting access based on roles. For example, developers may have permissions to deploy applications, while administrators manage configuration settings.
2. **Roles for Elastic Beanstalk Services**:
    - Elastic Beanstalk requires specific IAM roles to interact with other AWS services. For instance, Elastic Beanstalk often requires permissions to access EC2 instances, S3 storage for deployment artifacts, and CloudWatch for monitoring.

- **Service Role**: Elastic Beanstalk's service role grants the platform permissions to access AWS resources on behalf of the application.
- **Instance Profile Role**: Elastic Beanstalk environments use this role to grant permissions to EC2 instances within the environment, enabling access to other resources like S3 or DynamoDB as needed by the application.

3. **Policy Management**:
    - IAM policies define and manage permissions within Elastic Beanstalk, ensuring resources are only accessible by those who need it. Policies are attached to users, groups, or roles to specify allowed or denied actions.
4. **Security and Compliance**:
    - IAM provides audit capabilities that help monitor and log actions taken in Elastic Beanstalk, making it easier to maintain compliance and security standards. IAM integrates with AWS CloudTrail to record API calls, ensuring visibility into user actions and supporting compliance requirements.