What is AdvDevOps ?

Advanced DevOps, also known as DevOps 2.0 or DevOps at Scale, is an evolution of the DevOps methodology. DevOps is a set of practices that combines development (Dev) and IT operations (Ops) to automate and streamline the software development and delivery process. Advanced DevOps takes these principles to a more mature and sophisticated level, typically to address the needs of larger and more complex organizations or software projects.

Features of AdvDevOps ?

1) **Scalable** – handle challenges that come with large and complex environment
2) **Continuous Integration and Continuous Delivery** – CI/CD pipelines are used in advdevops to combine operations such as advanced testing , security check and deployment strategies .This ensure code is continuously and readily delivered to production
3) **Microservices** – software is broken into more simple and smaller process , which can reduce time for processing
4) **Containerization and Orchestration:** Technologies like Docker and Kubernetes are commonly used in advanced DevOps for containerization and orchestration
5) **Infrastructure as Code (IaC):** Advanced DevOps places a strong emphasis on IaC, where infrastructure is defined and managed through code. Tools like Terraform or Ansible are used to automate infrastructure provisioning and configuration.
6) **Collaboration:** Collaboration and communication among development, operations, and other teams (often referred to as DevSecOps when security is involved) is a central theme.

Pipeline in AdvDevOps ?

a pipeline refers to a series of automated processes that software goes through from development to deployment. These pipelines are crucial for ensuring a smooth, efficient, and consistent delivery of software. They typically include various stages, such as building, testing, and deploying the software.

Components of pipeline :

1) **SCM (source control management) -** The pipeline usually begins with the source code residing in a version control system such as Git.
2) **Build Automation:** The pipeline then initiates the build process
3) **Continuous Integration (CI):** Automated tests, including unit tests, integration tests, and other types of tests, are run to verify that the code changes have not introduced any issues.
4) **Continuous Delivery (CD):** The CD stage involves deploying the software to different environments, such as development, testing, staging, and eventually production.
5) **Testing and Quality Assurance (QA)**
6) **IaC**
7) **Feedback Loops**

DevOps vs AdvDevOps

Aspect	DevOps	Advanced DevOps (DevOps 2.0)
Scope	Small to medium-sized projects and teams	Large, complex projects and enterprise-level teams
Automation	Basic automation for CI/CD	Extensive automation for all aspects of development and operations
Scaling	Limited scaling capabilities	Advanced scaling for complex and large environments
Tooling	Relies on standard DevOps tools	Utilizes advanced tools for microservices, containerization, and orchestration

Aspect	DevOps	Advanced DevOps (DevOps 2.0)
Testing	Basic testing and QA integration	Comprehensive testing and quality assurance processes, including automated testing and security testing
Collaboration	Collaboration between Dev and Ops teams	Cross-functional collaboration involving Dev, Ops, and other teams, often including security (DevSecOps)
Monitoring	Basic monitoring and observability	Comprehensive monitoring with advanced analytics and observability tools
Security	Security integrated to an extent	Strong emphasis on security throughout the pipeline (DevSecOps)
Infrastructure	Basic infrastructure management	Advanced Infrastructure as Code (IaC) with tools like Terraform and Ansible
Release Management	Simple release management strategies	Advanced release strategies like blue-green deployments and canary releases
Pipeline Automation	Basic pipeline automation	Highly automated pipeline with advanced integration and delivery processes

Aspect	DevOps	Advanced DevOps (DevOps 2.0)
Cloud Integration	Basic cloud integration	Advanced cloud-native practices and integration
Disaster Recovery	Limited disaster recovery strategies	Robust disaster recovery and high availability strategies
Objectives	Faster delivery and collaboration	Scaling, automation, and integration for large and complex environments

What is AWS ?

AWS, or Amazon Web Services, is a comprehensive and widely adopted cloud computing platform provided by Amazon. It offers a broad set of global cloud-based services, including computing power, storage, databases, analytics, networking, machine learning, and many other functionalities, allowing businesses and individuals to build and deploy applications and services with flexibility, scalability, and reliability.

Compute services – EC2 (elastic compute cloud) , ECS

Storage services – Amazon S3(simple storage service) , Amazon EBS(elastic block store) , Amazon Glacier (long term data retrieval)

Database services – RDS(relational database service) , Amazon Dynamo DB for NoSQL database and Amazon RedShift for data warehousing

Networking services – Amazon VPC(virtual private cloud) for virtual network , Amazon Direct connect to get dedicated network connection and Amazon Route 53 for scalable DNS

Analytics Services – Amazon kinesis for real time data streaming , Amazon EMR(elastic map reduce) for big data processing , Amazon QuickSight for Business Analytics

ML services – Amazon Sagemaker to build , train and deploy ML models and Amazon Rekognition for image and video analysis

Management Tools – AWS CloudFormation for IaC , AWS cloudtrial to moniter API usage , AWS config to audit , assess and evaluate resources

Security and Identity Services – IAM(Identity and Access Management) , AWS cognito , AWS WAF(web application firewall)

Developer tools – AWS codecommit , AWS codedeploy

EC2

Amazon EC2 (Elastic Compute Cloud) is a web service provided by Amazon Web Services (AWS) that offers resizable compute capacity in the cloud. It allows users to obtain and configure virtual servers (known as instances) and run various applications and workloads on them. With EC2, users have full control over their computing resources and can scale capacity up or down based on their requirements.

Use - Allows developer to develop a product with working on client machine configurations , so making it easy for deployment

OS images – ubuntu , Amazon Linux , Windows , RedHat , MacOS

If instance is kept on – resource wastage , billing costs

Cloud9

AWS Cloud9 is an integrated development environment (IDE) offered by Amazon Web Services (AWS). It provides a cloud-based platform for software development, allowing developers to write, run, and debug code from within a web browser. Here are some key features and information about AWS Cloud9:

1) Browser-Based IDE: AWS Cloud9 allows developers to access their development environment through a web browser. This means you can code from anywhere with an internet connection without needing to set up your development environment on a local machine.

2) Collaboration: Cloud9 supports collaborative coding by allowing multiple users to work on the same code in real-time. This can be particularly useful for team projects and pair programming.

3) Built-in Terminal: Cloud9 includes a built-in terminal that lets you execute commands and run scripts directly from the IDE. This eliminates the need to switch between different tools for coding and terminal access.

4) Preconfigured Development Environments: AWS Cloud9 supports various programming languages and runtimes. It provides preconfigured environments for popular languages like JavaScript, Python, Java, and more. You can also create custom environments tailored to your needs.

5) AWS Integration: Cloud9 is tightly integrated with other AWS services, making it easy to deploy and test applications on AWS infrastructure. You can access resources like Amazon EC2 instances, Lambda functions, and databases directly from within the IDE.

6) Code Debugging: Cloud9 offers debugging capabilities, allowing developers to set breakpoints, inspect variables, and step through code to identify and fix issues.

7) Version Control Integration: Cloud9 can integrate with version control systems like Git, making it easier to manage code repositories and collaborate with team members.

AWS S3 Bucket

Amazon S3 (Simple Storage Service) is a cloud storage service provided by Amazon Web Services (AWS) that offers scalable, secure, and durable object storage. It allows users to store and retrieve data from anywhere on the web, making it an essential component for various types of cloud-based applications and services. S3 is designed to provide developers and IT teams with easy-to-use, highly scalable, and reliable storage infrastructure for a wide range of use cases.

AWS codepipeline

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes

You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plug into it.

A pipeline defines your release process workflow, and describes how a new code change progresses through your release process.

A pipeline comprises a series of stages (e.g., build, test, and deploy), which act as logical divisions in your workflow. Each stage is made up of a sequence of actions, which are tasks such as building code or deploying to test environments.

Pipelines must have at least two stages. The first stage of a pipeline is required to be a source stage, and the pipeline is required to additionally have at least one other stage that is a build or deployment stage.

Define your pipeline structure through a declarative JSON document that specifies your release workflow and its stages and actions. These documents enable you to update existing pipelines as well as provide starting templates for creating new pipelines.

A revision is a change made to the source location defined for your pipeline. It can include source code, build output, configuration, or data. A pipeline can have multiple revisions flowing through it at the same time.

A stage is a group of one or more actions. A pipeline can have two or more stages.

An action is a task performed on a revision. Pipeline actions occur in a specified order, in serial or in parallel, as determined in the configuration of the stage.

You can add actions to your pipeline that are in an AWS Region different from your pipeline.

Min – 2 stages and Max – 10 stages in pipeline

Types of actions – source , build , test , deploy , invoke , approval

Kubernetes

Kubernetes is an open-source container management tool that automates container deployment, scaling & load balancing. It schedules, runs, and manages isolated containers that are running on virtual/physical/cloud machines. All top cloud providers support Kubernetes. One popular name for Kubernetes is K8s.

Node services in Kubernetes

In Kubernetes, a Pod is the smallest deployable unit that can be created and managed. It is a group of one or more containers that are scheduled to run on the same host.

kubelet: It is the primary node agent that runs on each node in the cluster and ensures that the containers are running in a Pod.

kube-proxy: Kube-proxy is responsible for network proxying on the node and performs the necessary network address translation and load balancing to route the traffic to the appropriate Pod.

container runtime: Kubernetes supports various container runtimes such as Docker, containerd, and CRI-O. The container runtime is responsible for pulling the container images from the container registry, creating the container from those images, and managing the container lifecycle, including starting, stopping, and deleting containers.

kube-dns/coredns: These are the DNS services in Kubernetes that provide service discovery and enable communication between different services within the cluster.

kube-scheduler: The kube-scheduler is responsible for making decisions about which nodes should run specific Pods based on resource requirements, hardware constraints, and other policies. It assigns the Pods to nodes that can accommodate them and meet the specified criteria.

kube-controller-manager: This component includes several controllers that handle different aspects of the cluster, such as node and endpoint controllers, which perform tasks like node monitoring and managing endpoints

etcd: While not strictly a service running on the node, etcd is a consistent and highly available key-value store used by Kubernetes to store all of its data.

Pod Disruption Budget (PDB)?

A Pod Disruption Budget (PDB) is a Kubernetes feature that allows user to control the number of Pods that are down simultaneously in a given deployment or replica set. It ensures high availability by specifying the minimum number of Pods that must be available at any given time during voluntary disruptions, such as maintenance or scaling events.

pod : collection of container

What is the role of Load Balance in Kubernetes?

In Kubernetes, a load balancer serves the crucial role of distributing incoming network traffic across multiple backend services or Pods. Its primary functions include:

Traffic distribution: It evenly distributes incoming network traffic across multiple instances of an application or service running in the Kubernetes cluster. This ensures that no single Pod or node is overwhelmed by an excessive amount of traffic.

High availability: Load balancing helps ensure high availability and reliability of applications by preventing any single point of failure.

Scalability: Load balancing facilitates horizontal scaling by dynamically adding or removing Pods to handle varying levels of traffic.

Optimized resource utilization: By efficiently distributing traffic, a load balancer helps in optimizing resource utilization across the cluster.

What security measures can be taken while using kubernetes ?

There are a number of security measures that can be taken while using Kubernetes. Some of the most important include:

1)Use Role-Based Access Control (RBAC): RBAC allows you to define who has access to the Kubernetes API and what permissions they have.

2)Use third-party authentication for the API server: This allows you to integrate Kubernetes with an existing identity provider, such as GitHub or Okta.

3)Protect etcd with TLS and a firewall: etcd is a distributed key-value store that stores the state of your Kubernetes cluster. It is a critical component, so it is important to protect it from unauthorized access.

4)Isolate Kubernetes nodes: Kubernetes nodes are the machines that run your containerized applications.

5)Monitor network traffic to limit communications: Kubernetes workloads can communicate with each other over the network. It is important to monitor this traffic and limit it to only the necessary communication paths.

6)Keep Kubernetes up to date: The Kubernetes team regularly releases security updates.

What are the three security techniques used to protect data ?

The three most important security techniques used to protect data are:

1) Encryption

2) Access control

3) Backup and recovery

Encryption is the process of converting data into a format that cannot be read without a secret key.

Access control is the process of restricting access to data to authorized individuals. This can be done using a variety of methods, such as passwords, multi-factor authentication, and role-based access control (RBAC).

Backup and recovery is the process of creating copies of data and storing them in a secure location.

How do you expose a service using ingress in Kubernetes?

To expose a service using Ingress in Kubernetes, you need to create an Ingress resource. An Ingress resource specifies the rules for routing traffic to your services.

To create an Ingress resource, you can use the following command:

kubectl create ingress <ingress-name>

The Ingress resource must specify the following:

1) The rules for routing traffic to your services.

2) The hostname or IP address that traffic will be routed to.

3) The port that traffic will be routed to.

Which service protocol does Kubernetes ingress expose ?

Ingress is a Kubernetes resource that allows you to expose services running in a cluster to external traffic. Ingress can expose services through either **HTTP or HTTPS.**

Terraform

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

core Terraform workflow consists of three stages:

Write: You define resources, which may be across multiple cloud providers and services. For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.

Plan: Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.

Apply: On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies. For example, if you update the properties of a VPC and change the number of virtual machines in that VPC, Terraform will recreate the VPC before scaling the virtual machines.

Terraform lifecycle

Every resource that is managed by Terraform has a lifecycle, this lifecycle contains three stages; Apply (Create), Update, and Destroy. The Apply stage is where the resource is actually created by Terraform, Update is when a change is made to a preexisting resource - this might be incremental or a full recreate - and finally Destroy where the resource is removed from the environment. Sometimes however we may want to control these stages of the resources lifecycle a little more for this Terraform gives us the lifecycle meta-argument.

the lifecycle further than the three above stages for our resources. Terraform gives us the following options that we can use in the lifecycle meta-argument:

create_before_destroy — when an in-place update has to occur Terraform will create the new instance prior to destroying the old

prevent_destroy — do not allow the destroy flow to actually destruct the resource

ignore_changes — ignore any changes on specified fields or an entire object

replace_triggered_by - replace_triggered_by is a very new addition to language, only coming out with Terraform v1.2, it is also a very powerful argument. This will replace a particular resource based on another resource.

precondition — check some thing before performing the action on the resource

postcondition — validate some thing after performing an action on the resource

SonarQube

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications. It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

SonarQube is a universal tool for static code analysis that has become more or less the industry standard. Keeping code clean, simple, and easy to read is also a lot easier with SonarQube.

Benefits of SonarQube

Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.

Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code

Quality code - Code quality control is an inseparable part of the process of software development.

Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.

Increase consistency - Determines where the code criteria are breached and enhances the quality

Business scaling - No restriction on the number of projects to be evaluated

Enhance developer skills - Regular feedback on quality problems helps developers to improve their coding skills

Why SonarQube?

Developers working with hard deadlines to deliver the required functionality to the customer. It is so important for developers that many times they compromise with the code quality, potential bugs, code duplications, and bad distribution of complexity. Additionally, they tend to leave unused variables, methods, etc. In this scenario, the code would work in the desired way. To avoid these issues in code, developers should always follow the good coding practice, but sometimes it is not possible to follow the rules and maintain the good quality as there may be many reasons.

In order to achieve continuous code integration and deployment, developers need a tool that not only works once to check and tell them the problems in the code but also to track and control the code to check continuous code quality. To satisfy all these requirements, here comes SonarQube in the picture

Nagios

Nagios is an open-source monitoring system that provides monitoring of services, applications, and network resources. It is designed to alert system administrators about potential issues before they become critical problems. Nagios allows you to monitor the entire IT infrastructure, including servers, switches, applications, and services. It provides a comprehensive monitoring solution for both small and large organizations. Some key features and capabilities of Nagios

include:

Monitoring Capabilities: Nagios can monitor a wide variety of network services including SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH, and many more.

Alerting and Notification: It provides alerting and notification functionalities to notify system administrators when something goes wrong. Nagios can send alerts via email, SMS, or other methods to ensure that the right people are notified in real-time.

Plugin Architecture: Nagios has a modular architecture that allows users to develop their plugins and addons to monitor specific devices and services that are not covered by default.

Customizable Dashboards and Reports: Nagios offers customizable dashboards and reporting capabilities that provide insights into the performance and health of the monitored resources.

Scalability and Flexibility: Nagios can scale to monitor complex, large-scale IT infrastructures. It is highly flexible and can be customized to meet specific monitoring and alerting requirements.

Extensibility: Nagios can be extended through various addons and plugins, allowing it to integrate with other tools and services, and enabling the monitoring of a wide range of devices and applications.

Historical Monitoring and Trend Analysis: Nagios can store historical data and provide trend analysis, allowing system administrators to identify patterns and plan for future infrastructure needs.

Community Support and Active Development: Being an open-source project, Nagios has a vibrant community that contributes to its development and support. This community-driven approach ensures that the software remains updated and robust.

Centralized Monitoring: Nagios provides a centralized view of the entire IT infrastructure, allowing administrators to have a comprehensive overview of the health and performance of all monitored resources from a single location.

Integration with Third-Party Tools: Nagios can integrate with various third-party tools and services, making it a versatile monitoring solution that can fit into different IT ecosystems and workflows.

AWS Lambda Functions

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.

The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of, so that you can focus on writing application code.

Event-Driven Computing: AWS Lambda allows you to execute your code in response to events such as changes to data in an Amazon S3 bucket, updates to a DynamoDB table, HTTP requests via Amazon API Gateway, or custom events from various AWS services.

Serverless Architecture: It enables you to build applications and services without the need to manage infrastructure.

Support for Multiple Languages: AWS Lambda supports multiple programming languages including Node.js, Python, Java, Go, Ruby, and .NET Core, allowing you to choose the language that best suits your application.

Automatic Scaling: Lambda automatically scales your application by running code in response to each trigger.

Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic. You can scale up or scale down your tables' throughput capacity without downtime or performance degradation. You can use the AWS Management Console to monitor resource utilization and performance metrics DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Code :

```
import json
import boto3
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')  # Corrected variable name
    table = dynamodb.Table('Student')     # Corrected variable name
    response = table.put_item(Item=event)  # Corrected variable name
    return response
```

How to deploy Lambda function on AWS?

To deploy a Lambda function on AWS, user can follow these general steps:

1) Create a Lambda Function
2) Write and Upload Code
3) Configure the function
4) Configure Trigger
5) Test the function
6) Deploy the function
7) Monitor the function using AWS CloudWatch

What are the deployment options for AWS Lambda?

AWS Management Console: Users can create, configure, and deploy Lambda functions directly through the AWS Management Console.

AWS Command Line Interface (CLI): The AWS CLI provides a command-line interface for managing AWS services, including Lambda. Users can use the AWS CLI to create, deploy, update, and manage Lambda functions, which is particularly useful for automating deployment processes and integrating Lambda into scripts and workflows.

AWS Software Development Kits (SDKs): AWS SDKs are available for multiple programming languages, allowing users to integrate Lambda into their applications directly.

AWS CloudFormation: AWS CloudFormation enables users to define and deploy infrastructure as code. Users can use CloudFormation templates to

provision and manage Lambda functions along with other AWS resources in a declarative manner, facilitating consistent and repeatable deployments.

AWS Serverless Application Model (SAM): SAM is an open-source framework that extends CloudFormation to simplify the deployment of serverless applications.

Integrated Development Environments (IDEs): Several IDEs, such as AWS Toolkit for Visual Studio, AWS Toolkit for PyCharm, and AWS Toolkit for IntelliJ, offer tools and features for developing, debugging, and deploying Lambda functions directly from the IDE environment.

What are the 3 full deployment modes that can be used for AWS?

EC2 Deployment:

Description: EC2 (Elastic Compute Cloud) is a web service that provides resizable compute capacity in the cloud. It allows users to launch virtual servers (EC2 instances) and provides full control over the operating system, network, and other aspects of the infrastructure.

Use Case: EC2 deployment is suitable for users who require complete control and customization over their computing resources. It is ideal for applications that demand a high level of configurability and flexibility.

2) Elastic Beanstalk Deployment:

Description: AWS Elastic Beanstalk is a platform as a service (PaaS) that simplifies the process of deploying and scaling web applications and services. It automatically handles the details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

Use Case: Elastic Beanstalk is well-suited for developers who want to deploy applications without managing the underlying infrastructure. It is an excellent choice for web developers who want to focus on writing code rather than managing the infrastructure.

3) Serverless Deployment with AWS Lambda:

Description: AWS Lambda is a serverless computing service that allows users to run code without provisioning or managing servers. It automatically scales applications by running code in response to specific events or triggers.

Use Case: Lambda is suitable for building serverless applications and for executing code in response to various events, such as changes to data in an

Amazon S3 bucket, DynamoDB table updates, or HTTP requests via Amazon API Gateway. It is particularly beneficial for event-driven and microservices architectures where the focus is on individual functions rather than entire applications.

What are the 3 components of AWS Lambda?

Lambda Function:

The Lambda function is the heart of the AWS Lambda service. It is the code that runs in response to events. Users can upload their code or write it directly in the AWS Management Console, using one of the supported programming languages such as Node.js, Python, Java, Go, and others.

Event Sources:

Event sources are the triggers that invoke the Lambda function. AWS Lambda can be triggered by various AWS services such as Amazon S3, Amazon DynamoDB, Amazon Kinesis, Amazon SQS, and others.

Lambda Execution Environment:

The Lambda execution environment consists of the infrastructure and resources necessary to run the Lambda function. AWS manages the execution environment and automatically provisions and scales the infrastructure based on the incoming request volume.