DevOps Exam

1. **Q- Explain how the implementation of “Infrastructure as code” is processed or executed in terms of AWS and other VMware based private clouds also, what is the differences between the different tools**

IaC is a DevOps concept where we manage and treat our infrastructures in a descriptive model, just as we would do with code, it actually means to manage your IT infrastructure using configuration files. The benefits your organization will reap by adopting an IaC solution are **speed**, **consistency**, **accountability**.

Using [Amazon Web Services](https://aws.amazon.com/) as our cloud provider, let’s see how we can take advantage of IaC and explore the tools that we can use.

By making a simple API request or running a command via [AWS CLI](https://aws.amazon.com/cli/), we are able to create new servers based on an image, add them to a subnet, and specify a key pair to connect to them, and so on. All ways can be added to scripts. These scripts, as we mentioned, can be version controlled. The problem is that managing changes to our infrastructure is very difficult when it gets more complex.

Let’s start by listing the things that we would need for an infrastructure to work:

* [Amazon EC2](https://aws.amazon.com/ec2/) (Elastic Compute Cloud) with Auto Scaling to maintain the application availability and automatically add or remove EC2 instances as needed.
* [AWS RDS](https://aws.amazon.com/rds/) (Relational Database Service) to launch and manage Relational Databases.
* [Amazon ElastiCache](https://aws.amazon.com/elasticache/) to deploy, operate and scale the in-memory data store/cache in their cloud environment.
* [AWS Elastic Beanstalk](https://aws.amazon.com/elasticbeanstalk/) to quickly deploy and manage applications.
* [AWS CloudWatch](https://aws.amazon.com/cloudwatch/) to monitor the overall health of the infrastructure in production.
* [Elastic Load Balancing (ELB)](https://aws.amazon.com/elasticloadbalancing/) to distribute incoming traffic across multiple targets in multiple Availability Zones.
* [Amazon Virtual Private Cloud (VPC)](https://aws.amazon.com/vpc/) to launch AWS resources into a virtual interconnected network.
* [Amazon Elastic Container Service (ECS)](https://aws.amazon.com/ecs/) for container management.
* [Amazon S3 (Simple Storage Service)](https://aws.amazon.com/s3/) to store all the static assets.
* [Amazon CloudFront](https://aws.amazon.com/cloudfront/) as the Content Delivery Network (CDN).
* [AWS Lambda](https://aws.amazon.com/lambda/) to run code without provisioning or managing servers.

All of the parts of this full infrastructure have their own APIs. This means that we can create the commands needed to have everything scripted. Starting with the **networking** (VPC and subnets), and going up to the **servers** (EC2 instances), **cache** (ElastiCache), **load balancers** (ELBs), **name resolution** (Route 53), **monitoring and alerting** (CloudWatch), **queues** (SQS), and **databases** (RDS).

This is a simple approach to IaC; this is just the tip of the iceberg.

Launching servers using commands is the easiest part. What about installing our application? Or configuration of the different services? What about performance tweaks to our servers? If our application runs on a web server like Apache, Nginx, Tomcat, or others, how does it get installed? Security? Governance? Scale? Launching an EC2 instance is not enough; we need to install software there. However, we shouldn’t do all of it manually.

1. **Q- List the essential tools used in Devops (in 2020) and its use**

Git

Distributed source code management (SCM) tool that allows developers to track the progress of your development work by maintaining different versions of the source code.

Docker

Creates isolated environments for container apps to eliminate the conflicts between the apps. Isolating applications into separate containers makes applications portable and more secure as well.

Artifactory

Enterprise-ready repository manager. It provides an end-to-end, automated solution for tracking artifacts from development to production.

Jenkins

It enables CI/CD for any combination of coding language and source code repositories using a pipeline system. Its Pipeline-as-code functionality makes the CI/CD pipelines a complete code and ensures integrating the entire DevOps chain.

Kubernetes

Kubernetes facilitates the deployment of containerized apps to a cluster of computers instead of a single machine by automating the distribution and scheduling of containers across the cluster. and much more…

Ansible

fairly simple agent-less, easy to deploy configuration management tool that offers the continuous delivery feature enabling faster deployments. It automates application deployment, cloud provisioning, intra-service orchestration, and many other repetitive tasks.

AWS, Azure and GCP

DevOps-minded teams don’t need to choose just one cloud provider, or even go with a public cloud option. However, public cloud options allow for greater flexibility and agility which is often highly-beneficial for successful businesses. Whether you make use of a single cloud provider or leverage a multi-cloud or [hybrid cloud](https://victorops.com/blog/incident-response-for-hybrid-clouds) strategy is highly dependent on the types of applications or services you’re working with. Amazon Web Services ([AWS](https://aws.amazon.com/)), Microsoft [Azure](https://azure.microsoft.com/en-us/), and Google Cloud Platform ([GCP](https://cloud.google.com/)) each offer highly-effective cloud computing, storage and hosting functionality.

Consul

Widely used for discovering and configuring services in any infrastructure. It is a perfect tool for modern, elastic infrastructures.

Terraform

Generates a graph of all resources and creates and modifies any independent resources in parallel and maintains different versions.

Prometheus

Generate precise alerts and visualizations that really lead to business insights and engineering outcomes.

Splunk

With the ability to ingest data not only from services and devices themselves but also from other monitoring tools, Splunk can serve as a single source of truth for system health and performance.

Selenium

The full suite offered by Selenium allows for full-scale test automation across all aspects of a web application and its connected parts.

Gremlin

Design the types of experiments you’d like to conduct or simply re-enact past issues you’ve had and run them through your applications and services to see how they hold up.

ServiceNow

Manage IT workflows and track tickets over time.

1. **Q - What are the core operations of DevOps in terms of development and Infrastructure?**

Core operations of DevOps include:

* Development
* Version Control
* Testing - Code coverage
* Integration
* Deployment With infrastructure
* Delivery
* Configuration
* Orchestration
* Deployment
* Monitoring
* Feedback

1. **Q - Describe code branching strategies you have used.**

So far, i professionally have been acquainted with:

### Feature Branching Strategy

Branch-per-issue workflow allows developers to work separately. Master is with PR only,

### Release Branching Strategy

Involves creating a branch for a potential release that includes all applicable stories. When a team starts working on a new release, the branch is created.

### Trunk Based Branching Strategy

Intention is to continuously use Master, or Release Branch as SSOT (Single Source of Truth).

### Heater/Freezer Branching Strategy

Where Freezer means a release-tag, and Heater stands for a release-environment.

### Named Branching Strategy

Each branch is named after one project member, and the master has the name of PO.

1. **Q - How to enable a Linux service at a particular run level ?**

We know that when a System V-compliant service is installed, it creates a shell script under the /etc/init.d directory.

In RHEL-based distributions like CentOS, a command called chkconfig can be used to enable or disable a service in System V. It can also list installed services and their runlevels.

No such utility ships with Debian natively (update-rc.d installs or removes services from runlevels only). We can, however, install a custom tool called sysv-rc-conf to help us manage services. Run the following command to install sysv-rc-conf: sudo apt-get install sysv-rc-conf -y Once the tool has been installed, simply execute this command to see the runlevel behavior for various services: sudo sysv-rc-conf

The output will be a pretty graphical window as shown below. From here, we can clearly see what services are enabled for what runlevels (marked by X).

1. **Q- How to check which ports are listening on my Linux Server ?**

Open a terminal application i.e. shell prompt. Run any one of the following command on Linux to see open ports:  
sudo lsof -i -P -n | grep LISTEN  
sudo netstat -tulpn | grep LISTEN  
sudo lsof -i:22 ## see a specific port such as 22 ##  
sudo nmap -sTU -O IP-address-Here For the latest version of Linux use the ss command. For example, ss -tulw Anyone can also create a portScan in python..

1. **Q- Explain how you change the NTP settings on a 500 production server which some of them are running Linux and some windows?**

* **via Ansible**

[**https://mdschreier.com/2016/05/13/setting-up-ntp-via-ansible-in-my-private-lab/**](https://mdschreier.com/2016/05/13/setting-up-ntp-via-ansible-in-my-private-lab/)

* **With Scripts manually from linux to windows**

Configure from command line ( can execute with winexe ) <https://github.com/skalkoto/winexe>

Command

w32tm.exe /config /manualpeerlist:”your.ntp.server.fqdn” /syncfromflags:manual /reliable:YES /update

To force a manual update from your newly configured time source, to check it's working and make the initial changeover:

w32tm.exe /config /update

Restart the time service for the change to take effect:

From the command line / batch script:

net stop w32time

net start w32time

Or if you've been using the powershell environment:

restart-service w32time

* **With Scripts manually from linux to linux**

Run an SSH wrapper script to execute this commands ( can use sshpass to automate )

1. **Q- What are the main Maven Build Lifecycle phases?**

Phases are:

* *validate:* check if all information necessary for the build is available
* *compile:* compile the source code
* *test-compile:* compile the test source code
* *test:* run unit tests
* *package:* package compiled source code into the distributable format (jar, war, …)
* *integration-test:* process and deploy the package if needed to run integration tests
* *install:* install the package to a local repository
* *deploy:* copy the package to the remote repository

Yet, I have to mention the fact most places set up unit-testing and integration-testing as the same test phase, mostly due to the simplicity of their test suite.

1. **Q- How Will You Convert A String To A Number In Python?**

If you have a decimal integer represented as a string and you want to convert the Python string to an int, then you just pass the string to int(), which returns a decimal integer: >>> int("10") 10 >>> type(int("10")) <class 'int'>

By default, int() assumes that the string argument represents a decimal integer. If, however, you pass a hexadecimal string to int(), then you’ll see a ValueError:

>>> int("0x12F")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: invalid literal for int() with base 10: '0x12F'

The error message says that the string is not a valid decimal integer. It’s important to recognize the difference between two types of failed results of passing a string to int():

1. Syntax Error: A ValueError will occur when int() doesn’t know how to parse the string using the provided base (10 by default).
2. Logical Error: int() does know how to parse the string, but not the way you expected.

When you pass a string to int(), you can specify the number system that you’re using to represent the integer. The way to specify the number system is to use base: >>> int("0x12F", base=16) 303 Now, int() understands you are passing a hexadecimal string and expecting a decimal integer. The argument that you pass to base is not limited to 2, 8, 10, and 16: >>> int("10", base=3) 3

1. **Q- Your manager asked you to get a MSSQL Server in AWS, you got two options, get an RDS or to launch an EC2 and install MSSQL on it. What are the pros and cons per each solution ?**

One advantage of developers using Amazon RDS instead of managing their own databases is that it reduces or eliminates their administrative responsibilities.

**Amazon RDS**

RDS is a web service which makes it easier to set-up, operate and scale your relational database in the cloud. It offers automatic installation process, disk provisioning, upgrades, security patches and backups of your SQL Server databases. Other than that it also allows you to set up a highly scalable environment, fully managed by AWS by Multi-AZ (Availability Zone) synchronous replication.

**Amazon EC2**

EC2 is a web service that will allow you to run application programs in the AWS cloud. It allows developers to spin up virtual machines, which provide compute capacity for IT projects that run with global AWS data centers. You will get full control, you are responsible for the administration of your database. Also, it allows you to provision and configure database instances and storage. With EC2, you can provision your databases across the world to provide low latency to your end-users worldwide.

**RDS vs EC2: Which one to choose?**

While comparing RDS vs EC2 you will come across many of their pros and cons for MySQL Server database. RDS is easy to set up, cost-effective and allows you to focus on more important tasks. Whereas, EC2 offers complete control and flexibility for your SQL Server database. Once you understand the requirements of your application, you would be able to make a better decision. Our overall experience and analysis make us lean towards RDS a bit due to the following reasons:

* It allows you to outsource tasks like provisioning of the database, updating versions, and security to Amazon.
* RDS allows you to focus on important tasks like performance tuning and schema optimization of your database.
* You wouldn’t have to manually set up database mirroring and failover clusters because you get highly optimized database solutions and synchronous Multi-AZ replication.
* At times of disaster, you don’t have to worry about managing your backups as RDS automates this process.

However, EC2 is also preferred by many people because:

* It gives you full control over your database, OS and software stack.
* EC2 allows you to hire your own database administrators. They will help you manage your database by looking after backups, replication, and clustering.
* You can use SQL Server features that are not currently supported by Amazon RDS.
* It allows you to exceed your maximum database size and performance needs.
* With EC2, you can set up a disaster recovery solution in SQL Server with AWS as the source.

Thus, while **RDS** makes sense as the *first choice*, **EC2** offers some real benefits, for most applications. What it really comes down to is the cost of your time required to set up and maintain a database.