DevOps Interview Questions

codespaghetti.com/devops-interview-questions/

DevOps

DevOps Interview Questions, From beginner to expert level DevOps Professional. These questions covers a wide range of topics any DevOps professional needed to know to nail an interview.

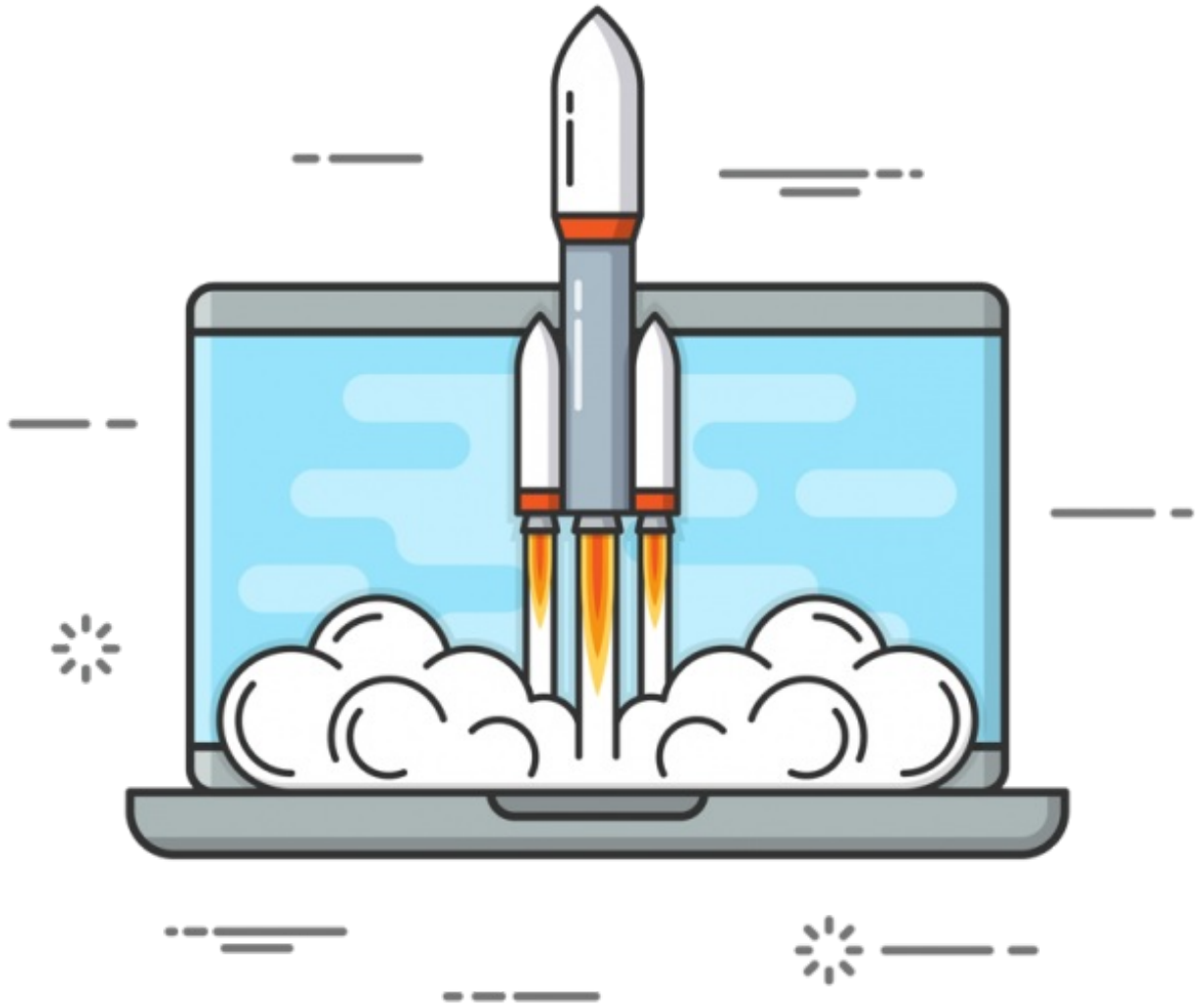

Table of Contents:

Question: **What Are Benefits Of DevOps?**

DevOps is gaining more popularity day by day. Here are some benefits of implementing DevOps Practice.

Release Velocity: DevOps enable organizations to achieve a great release velocity. We can release code to production more often and without any hectic problems.

Development Cycle: DevOps shortens the development cycle from initial design to production.

Full Automation: DevOps helps to achieve full automation from testing, to build, release and deployment.

Deployment Rollback: In DevOps, we plan for any failure in deployment rollback due to a bug in code or issue in production. This gives confidence in releasing feature without worrying about downtime for rollback.

Defect Detection: With DevOps approach, we can catch defects much earlier than releasing to production. It improves the quality of the software.

Collaboration: With DevOps, collaboration between development and operations professionals increases.

Performance-oriented: With DevOps, organization follows performance-oriented culture in

which teams become more productive and more innovative.

Question: What Is The Typical DevOps workflow?

The typical DevOps workflow is as follows:

- Atlassian Jira for writing requirements and tracking tasks.
- Based on the Jira tasks, developers checking code into GIT version control system.
- The code checked into GIT is built by using Apache Maven.
- The build process is automated with Jenkins.
- During the build process, automated tests run to validate the code checked in by a developer.
- Code built on Jenkins is sent to organization's Artifactory.
- Jenkins automatically picks the libraries from Artifactory and deploys it to Production.
- During Production deployment, Docker images are used to deploy same code on multiple hosts.
- Once a code is deployed to Production, we use monitoring tools like ngios are used to check the health of production servers.
- Splunk based alerts inform the admins of any issues or exceptions in production.

Question: DevOps Vs Agile?

Agile is a set of values and principles about how to develop software in a systematic way.

Where as DevOPs is a way to quickly, easily and repeatably move that software into production infrastructure, in a safe and simple way.

In oder to achieve that we use a set of DevOps tools and techniques.

Question: **What Is The Most Important Thing DevOps Helps Us To Achieve?**

Most important aspect of DevOps is to get the changes into production as quickly as possible while minimizing risks in software quality assurance and compliance. This is the primary objective of DevOps.

Question: What Are Some DevOps tools.

Here is a list of some most important DevOps tools

- Git
- Jenkins, Bamboo
- Selenium

- Puppet, BitBucket
- Chef
- Ansible, Artifactory
- Nagios
- Docker
- Monit
- ELK –Elasticsearch, Logstash, Kibana
- Collectd/Collect

Question: **How To Deploy Software?**

Code is deployed by adopting continuous delivery best practices. Which means that checked in code is built automatically and then artifacts are published to repository servers.

On the application severs there are deployment triggers usually timed by using cron jobs. All the artifacts are then downloaded and deployed automatically.

Gradle DevOps Interview Questions

Question: **What is Gradle?**

Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven. Gradle has a proper programming language instead of XML configuration file and the language is called ‘Groovy’.

Gradle uses a directed acyclic graph ("DAG") to determine the order in which tasks can be run.

Gradle was designed for multi-project builds, which can grow to be quite large. It supports incremental builds by intelligently determining which parts of the build tree are up to date, any task dependent only on those parts does not need to be re-executed.

Question: What Are Advantages of Gradle?

Gradle provides many advantages and here is a list

- **Declarative Builds:** Probably one of the biggest advantage of Gradle is Groovy language. Gradle provides declarative language elements. Which providea build-by-convention support for Java, Groovy, Web and Scala.
- **Structured Build:** Gradle allows developers to apply common design principles to their build. It provides a perfect structure for build, so that well-structured and easily maintained, comprehensible build structures can be built.
- **Deep API:** Using this API, developers can monitor and customize its configuration and execution behaviors.
- **Scalability:** Gradle can easily increase productivity, from simple and single project builds to huge enterprise multi-project builds.
- **Multi-project builds:** Gradle supports multi-project builds and also partial builds.
- **Build management:** Gradle supports different strategies to manage project dependencies.
- **First build integration tool** − Gradle completely supports ANT tasks, Maven and Ivy repository infrastructure for publishing and retrieving dependencies. It also provides a converter for turning a Maven pom.xml to Gradle script.
- **Ease of migration:** Gradle can easily adapt to any project structure.
- **Gradle Wrapper:** Gradle Wrapper allows developers to execute Gradle builds on machines where Gradle is not installed. This is useful for continuous integration of servers.
- **Free open source** − Gradle is an open source project, and licensed under the Apache Software License (ASL).
- **Groovy:** Gradle's build scripts are written in Groovy, not XML. But unlike other approaches this is not for simply exposing the raw scripting power of a dynamic language. The whole design of Gradle is oriented towards being used as a language, not as a rigid framework.

Question: Why Gradle Is Preferred Over Maven or Ant?

There isn't a great support for multi-project builds in Ant and Maven. Developers end up doing a lot of coding to support multi-project builds.

Also having some build-by-convention is nice and makes build scripts more concise. With Maven, it takes build by convention too far, and customizing your build process becomes a hack.

Maven also promotes every project publishing an artifact. Maven does not support subprojects to be built and versioned together.

But with Gradle developers can have the flexibility of Ant and build by convention of Maven.

Groovy is easier and clean to code than XML. In Gradle, developers can define dependencies between projects on the local file system without the need to publish artifacts to repository.

Question: Gradle Vs Maven

The following is a summary of the major differences between Gradle and Apache Maven:

Flexibility: Google chose Gradle as the official build tool for Android; not because build scripts are code, but because Gradle is modeled in a way that is extensible in the most fundamental ways.

Both Gradle and Maven provide convention over configuration. However, Maven provides a very rigid model that makes customization tedious and sometimes impossible.

While this can make it easier to understand any given Maven build, it also makes it unsuitable for many automation problems. Gradle, on the other hand, is built with an empowered and responsible user in mind.

Performance

Both Gradle and Maven employ some form of parallel project building and parallel dependency resolution. The biggest differences are Gradle's mechanisms for work avoidance and incrementally. Following features make Gradle much faster than Maven:

- **Incrementally:** Gradle avoids work by tracking input and output of tasks and only running what is necessary.
- **Build Cache:** Reuses the build outputs of any other Gradle build with the same inputs.
- **Gradle Daemon:** A long-lived process that keeps build information "hot" in memory.

User Experience

Maven's has a very good support for various IDE's. Gradle's IDE support continues to improve quickly but is not great as of Maven.

Although IDEs are important, a large number of users prefer to execute build operations through a command-line interface. Gradle provides a modern CLI that has discoverability features like `gradle tasks`, as well as improved logging and command-line completion.

Dependency Management

Both build systems provide built-in capability to resolve dependencies from configurable repositories. Both are able to cache dependencies locally and download them in parallel.

As a library consumer, Maven allows one to override a dependency, but only by version. Gradle provides customizable dependency selection and substitution rules that can be declared once and handle unwanted dependencies project-wide. This substitution mechanism enables Gradle to build multiple source projects together to create composite builds.

Maven has few, built-in dependency scopes, which forces awkward module architectures in common scenarios like using test fixtures or code generation. There is no separation between unit and integration tests, for example. Gradle allows custom dependency scopes, which provides better-modeled and faster builds.

Question: What are Gradle Build Scripts?

Gradle builds a script file for handling projects and tasks. Every Gradle build represents one or more projects.

A project represents a library JAR or a web application.

Question: What is Gradle Wrapper?

The wrapper is a batch script on Windows, and a shell script for other operating systems. Gradle Wrapper is the preferred way of starting a Gradle build.

When a Gradle build is started via the wrapper, Gradle will automatically download and run the build.

Question: What is Gradle Build Script File Name?

This type of name is written in the format that is build.gradle. It generally configures the Gradle scripting language.

Question: How To Add Dependencies In Gradle?

In order to make sure that dependency for your project is added, you need to mention the

configuration dependency like compiling the block dependencies of the build.gradle file.

Question: What Is Dependency Configuration?

Dependency configuration comprises of the external dependency, which you need to install well and make sure the downloading is done from the web. There are some key features of this configuration which are:

1. **Compilation:** The project which you would be starting and working on the first needs to be well compiled and ensure that it is maintained in the good condition.
2. **Runtime:** It is the desired time which is required to get the work dependency in the form of collection.
3. **Test Compile:** The dependencies check source requires the collection to be made for running the project.
4. **Test runtime:** This is the final process which needs the checking to be done for running the test that is in a default manner considered to be the mode of runtime

Question: What Is Gradle Daemon?

A daemon is a computer program that runs as a background process, rather than being under the direct control of an interactive user.

Gradle runs on the Java Virtual Machine (JVM) and uses several supporting libraries that require a non-trivial initialization time.

As a result, it can sometimes seem a little slow to start. The solution to this problem is the Gradle *Daemon*: a long-lived background process that executes your builds much more quickly than would otherwise be the case.

We accomplish this by avoiding the expensive bootstrapping process as well as leveraging caching, by keeping data about your project in memory. Running Gradle builds with the Daemon is no different than without

Question: What Is Dependency Management in Gradle?

Software projects rarely work in isolation. In most cases, a project relies on reusable functionality in the form of libraries or is broken up into individual components to compose a modularized system.

Dependency management is a technique for declaring, resolving and using dependencies required by the project in an automated fashion.

Gradle has built-in support for dependency management and lives up the task of fulfilling typical scenarios encountered in modern software projects.

Question: What Are Benefits Of Daemon in Gradle 3.0

Here are some of the benefits of Gradle daemon

1. It has good UX
2. It is very powerful
3. It is aware of the resource
4. It is well integrated with the Gradle Build scans
5. It has been default enabled

Question: What Is Gradle Multi-Project Build?

Multi-project builds helps with modularization. It allows a person to concentrate on one area of work in a larger project, while Gradle takes care of dependencies from other parts of the project

A multi-project build in Gradle consists of one root project, and one or more subprojects that may also have subprojects.

While each subproject could configure itself in complete isolation of the other subprojects, it is common that subprojects share common traits.

It is then usually preferable to share configurations among projects, so the same configuration affects several subprojects.

Question: What Is Gradle Build Task?

Gradle Build Tasks is made up of one or more projects and a project represents what is been done with Gradle.

Some key of features of Gradle Build Tasks are:

1. Task has life cycled methods [do first, do last]
2. Build Scripts are code
3. Default tasks like run, clean etc
4. Task dependencies can be defined using properties like dependsOn

Question: What is Gradle Build Life Cycle?

Gradle Build life cycle consists of following three steps

-Initialization phase: In this phase the project layer or objects are organized

-Configuration phase: In this phase all the tasks are available for the current build and a dependency graph is created

-Execution phase: In this phase tasks are executed.

Question: What is Gradle Java Plugin?

The Java plugin adds Java compilation along with testing and bundling capabilities to the project. It is introduced in the way of a SourceSet which act as a group of source files complied and executed together.

Question: What is Dependency Configuration?

A set of dependencies is termed as dependency configuration, which contains some external dependencies for download and installation.

Here are some key features of dependency configuration are:

Compile:

The project must be able to compile together

Runtime:

It is the required time needed to get the dependency work in the collection.

Test Compile:

The check source of the dependencies is to be collected in order to run the project.

Test Runtime:

The final procedure is to check and run the test which is by default act as a runtime mode.

Groovy DevOps Interview Questions

Question: What is Groovy?

Apache Groovy is a object-oriented programming language for the Java platform.

It is both a static and dynamic language with features similar to those of Python, Ruby, Perl, and Smalltalk.

It can be used as both a programming language and a scripting language for the Java Platform, is compiled to Java virtual machine (JVM) bytecode, and interoperates seamlessly with other Java code and libraries.

Groovy uses a curly-bracket syntax similar to Java. Groovy supports closures, multiline strings, and expressions embedded in strings.

And much of Groovy's power lies in its ASTtransformations, triggered through annotations.

Question: Why Groovy Is Gaining Popularity?

Here are few reasons for popularity of Groovy

- Familiar OOP language syntax.
- Extensive stock of various Java libraries

- Increased expressivity (type less to do more)
- Dynamic typing (lets you code more quickly, at least initially)
- Closures
- Native associative array/key-value mapping support (you can create an associative array literal)
- String interpolation (cleaner creation of strings displaying values)
- Regex's being first class citizens

Question: What Is Meant By Thin Documentation In Groovy

Groovy is documented very badly. In fact the core documentation of Groovy is limitedand there is no information regarding the complex and run-time errors that happen.

Developers are largely on there own and they normally have to figure out the explanations about internal workings by themselves.

Question: How To Run Shell Commands in Groovy?

Groovy adds the `execute` method to `String` to make executing shells fairly easy

```
println "ls".execute().text
```

Question: In How Many Platforms you can use Groovy?

These are the infrastructure components where we can use groovy:

-Application Servers

-Servlet Containers

-Databases with JDBC drivers

-All other Java-based platforms

Question: Can Groovy Integrate With Non Java Based Languages?

It is possible but in this case the features are limited. Groovy cannot be made to handle all the tasks in a manner it has to.

Question: What are Pre-Requirements For Groovy?

Installing and using Groovy is easy. Groovy does not have complex system requirements. It is OS independent.

Groovy can perform optimally in every situation.There are many Java based components in Groovy,which make it even more easier to work with Java applications.

Questions: What Is Closure In Groovy?

A closure in Groovy is an open, anonymous, block of code that can take arguments, return a value and be assigned to a variable. A closure may reference variables declared in its surrounding scope. In opposition to the formal definition of a closure, `Closure` in the Groovy language can also contain free variables which are defined outside of its surrounding scope.

A closure definition follows this syntax:

```
{ [closureParameters -> ] statements }
```

Where `[closureParameters->]` is an optional comma-delimited list of parameters, and statements are 0 or more Groovy statements. The parameters look similar to a method parameter list, and these parameters may be typed or untyped.

When a parameter list is specified, the `->` character is required and serves to separate the arguments from the closure body. The *statements* portion consists of 0, 1, or many Groovy statements.

Question: What is ExpandoMeta Class In Groovy?

Through this class programmers can add properties, constructors, methods and operations in the task. It is a powerful option available in the Groovy.

By default this class cannot be inherited and users need to call explicitly. The command for this is “ExpandoMetaClass.enableGlobally()”.

Question: What Are Limitations Of Groovy?

Groovy has some limitations. They are described below

- It can be slower than the other object-oriented programming languages.
- It might need memory more than that required by other languages.
- The start-up time of groovy requires improvement. It is not that frequent.
- For using groovy, you need to have enough knowledge of Java. Knowledge of Java is important because half of groovy is based on Java.

- It might take you some time to get used to the usual syntax and default typing.
- It consists of thin documentation.

Question: How To Write HelloWorld Program In Groovy

The following is a basic Hello World program written in Groovy:

```
class Test {

static void main(String[] args) {

println('Hello World');

}

}
```

Question: How To Declare String In Groovy?

In Groovy, the following steps are needed to declare a string.

- The string is closed with single and double qotes.
- It contains Groovy Expressions noted in ${}
- Square bracket syntax may be applied like charAt(i)

Question: Differences Between Java And Groovy?

Groovy tries to be as natural as possible for Java developers. Here are all the major differences between Java and Groovy.

-Default imports

In Groovy all these packages and classes are imported by default, i.e. Developers do not have to use an explicit `import` statement to use them:

- java.io.*
- java.lang.*
- java.math.BigDecimal
- java.math.BigInteger
- java.net.*
- java.util.*
- groovy.lang.*
- groovy.util.*

-Multi-methods

In Groovy, the methods which will be invoked are chosen at runtime. This is called runtime dispatch or multi-methods. It means that the method will be chosen based on the types of the arguments at runtime. In Java, this is the opposite: methods are chosen at compile time, based on the declared types.

-Array initializers

In Groovy, the `{ … }` block is reserved for closures. That means that you cannot create array literals with this syntax:

```
int[] arraySyntex = { 6, 3, 1}
```

You actually have to use:

```
int[] arraySyntex = [1,2,3]
```

-ARM blocks

ARM (Automatic Resource Management) block from Java 7 are not supported in Groovy. Instead, Groovy provides various methods relying on closures, which have the same effect while being more idiomatic.

-GStrings

As double-quoted string literals are interpreted as `GString` values, Groovy may fail with compile error or produce subtly different code if a class with `String` literal containing a dollar character is compiled with Groovy and Java compiler.
While typically, Groovy will auto-cast between `GString` and `String` if an API declares the type of a parameter, beware of Java APIs that accept an `Object` parameter and then check the actual type.

-String and Character literals

Singly-quoted literals in Groovy are used for `String`, and double-quoted result in `String` or `GString`, depending whether there is interpolation in the literal.

```
assert 'c'.getClass()==String
assert "c".getClass()==String
assert "c${1}".getClass() in GString
```

Groovy will automatically cast a single-character `String` to `char` only when assigning to a variable of type `char`. When calling methods with arguments of type `char` we need to either cast explicitly or make sure the value has been cast in advance.

```
char a='a'
assert Character.digit(a, 16)==10 : 'But Groovy does boxing'
assert Character.digit((char) 'a', 16)==10

try {
  assert Character.digit('a', 16)==10
  assert false: 'Need explicit cast'
```

```
} catch(MissingMethodException e) {
}
```

Groovy supports two styles of casting and in the case of casting to `char` there are subtle differences when casting a multi-char strings. The Groovy style cast is more lenient and will take the first character, while the C-style cast will fail with exception.

```
// for single char strings, both are the same
assert ((char) "c").class==Character
assert ("c" as char).class==Character

// for multi char strings they are not
try {
  ((char) 'cx') == 'c'
  assert false: 'will fail - not castable'
} catch(GroovyCastException e) {
}
assert ('cx' as char) == 'c'
assert 'cx'.asType(char) == 'c'
```

-Behaviour of `==`

In Java `==` means equality of primitive types or identity for objects. In Groovy `==` translates to `a.compareTo(b)==0`, if they are `Comparable`, and `a.equals(b)` otherwise. To check for identity, there is `is`. E.g. `a.is(b)`.

Question: How To Test Groovy Application?

The Groovy programming language comes with great support for writing tests. In addition to the language features and test integration with state-of-the-art testing libraries and frameworks.

The Groovy ecosystem has born a rich set of testing libraries and frameworks.

Groovy Provides following testing capabilities

Junit Integrations

Spock for specifications

Geb for Functional Test

Groovy also has excellent built-in support for a range of mocking and stubbing alternatives. When using Java, dynamic mocking frameworks are very popular.

A key reason for this is that it is hard work creating custom hand-crafted mocks using Java. Such frameworks can be used easily with Groovy.

Question: What Are Power Assertions In Groovy?

Writing tests means formulating assumptions by using assertions. In Java this can be done by using the `assert` keyword. But Groovy comes with a *powerful variant* of `assert` also known as *power assertion statement*.
Groovy's power `assert` differs from the Java version in its output given the boolean expression validates to `false`:

```
def x = 1
assert x == 2

// Output:
//
// Assertion failed:
// assert x == 2
//        | |
//        1 false
```

This section shows the std-err output

The `java.lang.AssertionError` that is thrown whenever the assertion can not be validated successfully, contains an extended version of the original exception message. The power assertion output shows evaluation results from the outer to the inner expression. The power assertion statements true power unleashes in complex Boolean statements, or statements with collections or other `toString`-enabled classes:

```
def x = [1,2,3,4,5]
assert (x << 6) == [6,7,8,9,10]

// Output:
//
// Assertion failed:
// assert (x << 6) == [6,7,8,9,10]
//         | |     |
//         | |     false
//         | [1, 2, 3, 4, 5, 6]
//         [1, 2, 3, 4, 5, 6]
```

Question: Can We Use Design Patterns In Groovy?

Design patterns can also be used with Groovy. Here are important points

- Some patterns carry over directly (and can make use of normal Groovy syntax improvements for greater readability)
- Some patterns are no longer required because they are built right into the language or because Groovy supports a better way of achieving the intent of the pattern
- some patterns that have to be expressed at the design level in other languages can be implemented directly in Groovy (due to the way Groovy can blur the distinction between design and implementation)

Question: How To Parse And Produce JSON Object In Groovy?

Groovy comes with integrated support for converting between Groovy objects and JSON. The classes dedicated to JSON serialisation and parsing are found in the `groovy.json` package.
`JsonSlurper` is a class that parses JSON text or reader content into Groovy data structures (objects) such as maps, lists and primitive types like `Integer`, `Double`, `Boolean` and `String`.

The class comes with a bunch of overloaded `parse` methods plus some special methods such as `parseText`, `parseFile` and others

---

Question: What Is Difference Between XmlParser And XmlSluper?

XmlParser and XmlSluper are used for parsing XML with Groovy. Both have the same approach to parse an xml.

Both come with a bunch of overloaded parse methods plus some special methods such as `parseText`, parseFile and others.

XmlSlurper

```
def text = '''
    <list>
        <technology>
            <name>Groovy</name>
        </technology>
    </list>
'''

def list = new XmlSlurper().parseText(text)

assert list instanceof groovy.util.slurpersupport.GPathResult
assert list.technology.name == 'Groovy'
```

Parsing the XML an returning the root node as a GPathResult
Checking we're using a GPathResult
Traversing the tree in a GPath style

XmlParser

```
def text = '''
    <list>
        <technology>
            <name>Groovy</name>
        </technology>
    </list>
'''

def list = new XmlParser().parseText(text)

assert list instanceof groovy.util.Node
assert list.technology.name.text() == 'Groovy'
```

Parsing the XML an returning the root node as a Node
Checking we're using a Node
Traversing the tree in a GPath style

Let's see the similarities between `XMLParser` and `XMLSlurper` first:

- Both are based on `SAX` so they both are low memory footprint
- Both can update/transform the XML

But they have key differences:

- `XmlSlurper` evaluates the structure lazily. So if you update the xml you'll have to evaluate the whole tree again.
- `XmlSlurper` returns `GPathResult` instances when parsing XML
- `XmlParser` returns `Node` objects when parsing XML

When to use one or the another?

- If you want to transform an existing document to another then `XmlSlurper` will be the choice
- If you want to update and read at the same time then `XmlParser` is the choice.

Maven DevOps Interview Questions

Question: What is Maven?

Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software:

First: It describes how software is built

Second: It describes its dependencies.

Unlike earlier tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down.

An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins.

It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging.

Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache.

This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

Question: What Are Benefits Of Maven?

- One of the biggest benefit of Maven is that its design regards all projects as having a certain structure and a set of supported task work-flows.
- Maven has quick project setup, no complicated build.xml files, just a POM and go
- All developers in a project use the same jar dependencies due to centralized POM.
- In Maven getting a number of reports and metrics for a project "for free"
- It reduces the size of source distributions, because jars can be pulled from a central location
- Maven lets developers get your package dependencies easily
- With Maven there is no need to add jar files manually to the class path

Question: What Are Build Life cycles In Maven?

Build lifecycle is a list of named phases that can be used to give order to goal execution. One of Maven's standard life cycles is the *default lifecycle*, which includes the following phases, in this order

```
validate
generate-sources
process-sources
generate-resources
process-resources
compile
process-test-sources
process-test-resources
test-compile
test
package
install
deploy
```

Question: What Is Meant By Build Tool?

Build tools are programs that automate the creation of executable applications from source code. Building incorporates compiling, linking and packaging the code into a usable or executable form.

In small projects, developers will often manually invoke the build process. This is not practical for larger projects.

Where it is very hard to keep track of what needs to be built, in what sequence and what dependencies there are in the building process. Using an automation tool like Maven, Gradle or ANT allows the build process to be more consistent.

Question: What Is Dependency Management Mechanism In Gradle?

Maven's dependency-handling mechanism is organized around a coordinate system identifying individual artifacts such as software libraries or modules.

For example if a project needs Hibernate library. It has to simply declare Hibernate's project coordinates in its POM.

Maven will automatically download the dependency and the dependencies that Hibernate itself needs and store them in the user's local repository.

Maven 2 Central Repository is used by default to search for libraries, but developers can configure the custom repositories to be used (e.g., company-private repositories) within the POM.

Question: What Is Central Repository Search Engine?

The Central Repository Search Engine, can be used to find out coordinates for different open-source libraries and frameworks.

Question: What are Plugins In Maven?

Most of Maven's functionality is in plugins. A plugin provides a set of goals that can be executed using the following syntax:

```
mvn [plugin-name]:[goal-name]
```

For example, a Java project can be compiled with the compiler-plugin's compile-goal by running `mvn compiler:compile`. There are Maven plugins for building, testing, source control management, running a web server, generating Eclipse project files, and much more. Plugins are introduced and configured in a <plugins>-section of a `pom.xml` file. Some basic plugins are included in every project by default, and they have sensible default settings.

Questions: What Is Difference Between Maven And ANT?

Ant	Maven
Ant is a tool box.	Maven is a framework.
There is no life cycle.	There is life cycle.

Ant doesn't have formal conventions.	Maven has a convention to place source code, compiled code etc.
Ant is procedural.	Maven is declarative.
The ant scripts are not reusable.	The maven plugins are reusable.

Question: What is POM In Maven?

A Project Object Model (POM) provides all the configuration for a single project. General configuration covers the project's name, its owner and its dependencies on other projects.

One can also configure individual phases of the build process, which are implemented as plugins.

For example, one can configure the compiler-plugin to use Java version 1.5 for compilation, or specify packaging the project even if some unit tests fail.

Larger projects should be divided into several modules, or sub-projects, each with its own POM. One can then write a root POM through which one can compile all the modules with a single command. POMs can also inherit configuration from other POMs. All POMs inherit from the Super POM by default. The Super POM provides default configuration, such as default source directories, default plugins, and so on.

Question: What Is Maven Archetype?

Archetype is a Maven project templating toolkit. An archetype is defined as an original pattern or model from which all other things of the same kind are made.

Question: What Is Maven Artifact?

In Maven artifact is simply a file or JAR that is deployed to a Maven repository. An artifact has

-Group ID

-Artifact ID

-Version string. The three together uniquely identify the artifact. All the project dependencies are specified as artifacts.

Question: What Is Goal In Maven?

In Maven a goal represents a specific task which contributes to the building and managing

of a project.

It may be bound to 1 or many build phases. A goal not bound to any build phase could be executed outside of the build lifecycle by its direct invocation.

Question: What Is Build Profile?

In Maven a build profile is a set of configurations. This set is used to define or override default behaviour of Maven build.

Build profile helps the developers to customize the build process for different environments. For example you can set profiles for Test, UAT, Pre-prod and Prod environments each with its own configurations etc.

Question: What Are Build Phases In Maven?

There are 6 build phases. -Validate -Compile -Test -Package -Install -Deploy

Question: What Is Target, Source & Test Folders In Mavn?

Target: folder holds the compiled unit of code as part of the build process.
Source: folder usually holds java source codes. **Test:** directory contains all the unit testing codes.

Question: What Is Difference Between Compile & Install?

Compile: is used to compile the source code of the project **Install:** installs the package into the local repository, for use as a dependency in other projects locally.Design patterns can also be used with Groovy. Here are important points

Question: How To Activate Maven Build Profile?

A Maven Build Profile can be activated in following ways

- Using command line console input.
- By using Maven settings.
- Based on environment variables (User/System variables).

Linux DevOps Interview Questions

Question: What is Linux?

Linux is the best-known and most-used open source operating system. As an operating system, Linux is a software that sits underneath all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware.

In many ways, Linux is similar to other operating systems such as Windows, OS X, or iOS

But Linux also is different from other operating systems in many important ways.

First, and perhaps most importantly, Linux is open source software. The code used to create Linux is free and available to the public to view, edit, and—for users with the appropriate skills—to contribute to.

Linux operating system is consist of 3 components which are as below:

- **Kernel:** Linux is a monolithic kernel that is free and open source software that is responsible for managing hardware resources for the users.
- **System Library:** System Library plays a vital role because application programs access Kernels feature using system library.
- **System Utility:** System Utility performs specific and individual level tasks.

Question: What Is Difference Between Linux & Unix?

Unix and Linux are similar in many ways, and in fact, Linux was originally created to be similar to Unix.

Both have similar tools for interfacing with the systems, programming tools, filesystem layouts, and other key components.

However, Unix is not free. Over the years, a number of different operating systems have been created that attempted to be “unix-like” or “unix-compatible,” but Linux has been the most successful, far surpassing its predecessors in popularity.

Question: What Is BASH?

BASH stands for **Bourne Again Shell**. BASH is the UNIX shell for the GNU operating system. So, BASH is the command language interpreter that helps you to enter your input, and thus you can retrieve information.

In a straightforward language, BASH is a program that will understand the data entered by the user and execute the command and gives output.

Question: What Is CronTab?

The crontab (short for "cron table") is a list of commands that are scheduled to run at regular time intervals on computer system. The **crontab** command opens the crontab for editing, and lets you add, remove, or modify scheduled tasks.

The daemon which reads the crontab and executes the commands at the right time is called cron. It's named after Kronos, the Greek god of time.

Command syntax

crontab [-u *user*] *file*

crontab [-u *user*] [-l | -r | -e] [-i] [-s]

Question: What Is Daemon In Linux?

A **daemon** is a type of program on Linux operating systems that runs unobtrusively in the background, rather than under the direct control of a user, waiting to be activated by the occurrence of a specific event or condition

Unix-like systems typically run numerous daemons, mainly to accommodate requests for services from other computers on a network, but also to respond to other programs and to hardware activity.

Examples of actions or conditions that can trigger daemons into activity are a specific time or date, passage of a specified time interval, a file landing in a particular directory, receipt of an e-mail or a Web request made through a particular communication line.

It is not necessary that the perpetrator of the action or condition be aware that a daemon is *listening*, although programs frequently will perform an action only because they are aware that they will implicitly arouse a daemon.

Question: What Is Process In Linux?

Daemons are usually instantiated as *processes*. A process is an *executing* (i.e., running) instance of a program.

Processes are managed by the *kernel* (i.e., the core of the operating system), which assigns each a unique *process identification number* (PID).

There are three basic types of processes in Linux:

-Interactive: Interactive processes are run interactively by a user at the *command line*

-Batch: Batch processes are submitted from a queue of processes and are not associated with the command line; they are well suited for performing recurring tasks when system usage is otherwise low.

-Daemon: Daemons are recognized by the system as any processes whose *parent process* has a PID of one

---

Question: What Is CLI In Linux?

CLI (Command Line Interface) is a type of *human-computer interface* that relies solely on textual input and output.

That is, the entire display screen, or the currently active portion of it, shows only characters (and no images), and input is usually performed entirely with a keyboard.

Questions: What Is Linux Kernel?

A kernel is the lowest level of easily replaceable software that interfaces with the hardware in your computer.

It is responsible for interfacing all of your applications that are running in “user mode” down

to the physical hardware, and allowing processes, known as servers, to get information from each other using inter-process communication (IPC).

There are three types of Kernals

Microkernel:A microkernel takes the approach of only managing what it has to: CPU, memory, and IPC. Pretty much everything else in a computer can be seen as an accessory and can be handled in user mode.

Monolithic Kernel: Monolithic kernels are the opposite of microkernels because they encompass not only the CPU, memory, and IPC, but they also include things like device drivers, file system management, and system server calls

Hybrid Kernel: Hybrid kernels have the ability to pick and choose what they want to run in user mode and what they want to run in supervisor mode. Because the Linux kernel is monolithic, it has the largest footprint and the most complexity over the other types of kernels. This was a design feature which was under quite a bit of debate in the early days of Linux and still carries some of the same design flaws that monolithic kernels are inherent to have.

Question: What Is Partial Backup In Linux?

Partial backup refers to selecting only a portion of file hierarchy or a single partition to back up.

Question: What Is Root Account?

The root account a system administrator account. It provides you full access and control of the system.

Admin can create and maintain user accounts, assign different permission for each account etc

---

Question: What Is Difference Between Cron and **Anacron**?

One of the main difference between cron and anacron jobs is that cron works on the system that are running continuously.

While anacron is used for the systems that are not running continuously.

1. Other difference between the two is cron jobs can run every minute, but anacron jobs can be run only once a day.
2. Any normal user can do the scheduling of cron jobs, but the scheduling of anacron jobs can be done by the superuser only.

3. Cron should be used when you need to execute the job at a specific time as per the given time in cron, but anacron should be used in when there is no any restriction for the timing and can be executed at any time.
4. If we think about which one is ideal for servers or desktops, then cron should be used for servers while anacron should be used for desktops or laptops.

Question: What Is Linux Loader?

Linux Loader is a boot loader for Linux operating system. It loads Linux into into the main memory so that it can begin its operations.

Question: What Is Swap Space?

Swap space is the amount of physical memory that is allocated for use by Linux to hold some concurrent running programs temporarily.

This condition usually occurs when Ram does not have enough memory to support all concurrent running programs.

This memory management involves the swapping of memory to and from physical storage.

Question: What Are Linux Distributors?

There are around six hundred Linux distributors. Let us see some of the important ones

- ***UBuntu:*** It is a well known Linux Distribution with a lot of pre-installed apps and easy to use repositories libraries. It is very easy to use and works like MAC operating system.
- ***Linux Mint:*** It uses cinnamon and mate desktop. It works on windows and should be used by newcomers.
- ***Debian:*** It is the most stable, quicker and user-friendly Linux Distributors.
- ***Fedora:*** It is less stable but provides the latest version of the software. It has GNOME3 desktop environment by default.
- ***Red Hat Enterprise:*** It is to be used commercially and to be well tested before release. It usually provides the stable platform for a long time.
- ***Arch Linux:*** Every package is to be installed by you and is not suitable for the beginners.

Question: Why Do Developers Use MD5?

MD5 is an encryption method so it is used to encrypt the passwords before saving.

Question: What Are File Permissions In Linux?

There are 3 types of permissions in Linux

- **Read:** User can read the file and list the directory.
- **Write:** User can write new files in the directory .
- **Execute:** User can access and run the file in a directory.

Question: Memory Management In Linux?

It is always required to keep a check on the memory usage in order to find out whether the user is able to access the server or the resources are adequate. There are roughly 5 methods that determine the total memory used by the Linux.

This is explained as below

- ***Free command*:** This is the most simple and easy to use the command to check memory usage. For example: '$ free –m', the option 'm' displays all the data in MBs.
- **/proc/meminfo:** The next way to determine the memory usage is to read /proc/meminfo file. For example: '$ cat /proc/meminfo'
- ***Vmstat*:** This command basically lays out the memory usage statistics. For example: '$ vmstat –s'
- ***Top command*:** This command determines the total memory usage as well as also monitors the RAM usage.
- ***Htop*:** This command also displays the memory usage along with other details.

Question: Granting Permissions In Linux?

System administrator or the owner of the file can grant permissions using the 'chmod' command. Following symbols are used while writing permissions

chmod +x

Question: What Are Directory Commands In Linux?

Here are few important directory commands in Linux

- ***pwd:*** It is a built-in command which stands for **'print working directory'**. It displays the current working location, working path starting with / and directory of the user. Basically, it displays the full path to the directory you are currently in.
- ***ls:*** This command list out all the files in the directed folder.
- ***cd:*** This stands for 'change directory'. This command is used to change to the

directory you want to work from the present directory. We just need to type cd followed by the directory name to access that particular directory.

- ***mkdir:*** This command is used to create an entirely new directory.
- ***rmdir:*** This command is used to remove a directory from the system.

Question: What Is Shell Script In Linux?

In the simplest terms, a shell script is a file containing a series of commands.

The shell reads this file and carries out the commands as though they have been entered directly on the command line.

The shell is somewhat unique, in that it is both a powerful command line interface to the system and a scripting language interpreter.

As we will see, most of the things that can be done on the command line can be done in scripts, and most of the things that can be done in scripts can be done on the command line.

We have covered many shell features, but we have focused on those features most often used directly on the command line.

The shell also provides a set of features usually (but not always) used when writing programs.

Question: Which Tools Are Used For Reporting Statistics In Linux?

Some of the popular and frequently used system resource generating tools available on the Linux platform are

- vmstat
- netstat
- iostat
- **ifstat**
- mpstat.

These are used for reporting statistics from different system components such as virtual memory, network connections and interfaces, CPU, input/output devices and more.

Question: What Is Dstat In Linux?

dstat is a powerful, flexible and versatile tool for generating Linux system resource statistics, that is a replacement for all the tools mentioned in above question.

It comes with extra features, counters and it is highly extensible, users with Python knowledge can build their own plugins.

Features of dstat:

1. Joins information from vmstat, netstat, iostat, ifstat and mpstat tools
2. Displays statistics simultaneously
3. Orders counters and highly-extensible
4. Supports summarizing of grouped block/network devices
5. Displays interrupts per device
6. Works on accurate timeframes, no timeshifts when a system is stressed
7. Supports colored output, it indicates different units in different colors
8. Shows exact units and limits conversion mistakes as much as possible
9. Supports exporting of CSV output to Gnumeric and Excel documents

Question: Types Of Processes In Linux?

There are fundamentally two types of processes in Linux:

- **Foreground processes** (also referred to as interactive processes) – these are initialized and controlled through a terminal session. In other words, there has to be a user connected to the system to start such processes; they haven't started automatically as part of the system functions/services.
- **Background processes** (also referred to as non-interactive/automatic processes) – are processes not connected to a terminal; they don't expect any user input.

Question: Creatin Of Processes In Linux?

A new process is normally created when an existing process makes an exact copy of itself in memory.

The child process will have the same environment as its parent, but only the process ID number is different.

There are two conventional ways used for creating a new process in Linux:

- **Using The System() Function** – this method is relatively simple, however, it's inefficient and has significantly certain security risks.
- **Using fork() and exec() Function** – this technique is a little advanced but offers greater flexibility, speed, together with security.

Question: Creation Of Processes In Linux?

Because Linux is a multi-user system, meaning different users can be running various programs on the system, each running instance of a program must be identified uniquely by the kernel.

And a program is identified by its process ID (**PID**) as well as it's parent processes ID (**PPID**), therefore processes can further be categorized into:

- **Parent processes** – these are processes that create other processes during run-time.
- **Child processes** – these processes are created by other processes during run-time.

Question: What Is Init Process Linux?

I**nit** process is the mother (parent) of all processes on the system, it's the first program that is executed when the Linux system boots up; it manages all other processes on the system. It is started by the kernel itself, so in principle it does not have a parent process.

The init process always has process ID of **1**. It functions as an adoptive parent for all orphaned processes.

You can use the **pidof command** to find the ID of a process:

```
# pidof systemd
# pidof top
# pidof httpd

Find Linux Process ID
```

To find the process ID and parent process ID of the current shell, run:

```
$ echo $$
$ echo $PPID
```

Question: What Are Different States Of A Processes In Linux?

During execution, a process changes from one state to another depending on its environment/circumstances. In Linux, a process has the following possible states:

- **Running** – here it's either running (it is the current process in the system) or it's ready to run (it's waiting to be assigned to one of the CPUs).
- **Waiting** – in this state, a process is waiting for an event to occur or for a system resource. Additionally, the kernel also differentiates between two types of waiting processes; interruptible waiting processes – can be interrupted by signals and uninterruptible waiting processes – are waiting directly on hardware conditions and cannot be interrupted by any event/signal.
- **Stopped** – in this state, a process has been stopped, usually by receiving a signal. For instance, a process that is being debugged.

- **Zombie** – here, a process is dead, it has been halted but it's still has an entry in the process table.

Question: How To View Active Processes In Linux?

There are several Linux tools for viewing/listing running processes on the system, the two traditional and well known are ps and top commands:

1. ps Command

It displays information about a selection of the active processes on the system as shown below:

#ps

#ps -e] head

2. top – System Monitoring Tool

top is a powerful tool that offers you a dynamic real-time view of a running system as shown in the screenshot below:

#top

3. glances – System Monitoring Tool

glances is a relatively new system monitoring tool with advanced features:

#glances

Question: How To Control Process?

Linux also has some commands for controlling processes such as kill, pkill, pgrep and killall, below are a few basic examples of how to use them:

```
$ pgrep -u tecmint top
$ kill 2308
$ pgrep -u tecmint top
$ pgrep -u tecmint glances
$ pkill glances
$ pgrep -u tecmint glances
```

Question: Can We Send signals To Processes In Linux?

The fundamental way of controlling processes in Linux is by sending signals to them. There are multiple signals that you can send to a process, to view all the signals run:

```
$ kill -l
```

List All Linux Signals

To send a signal to a process, use the kill, pkill or pgrep commands we mentioned earlier on. But programs can only respond to signals if they are programmed to recognize those signals.

And most signals are for internal use by the system, or for programmers when they write code. The following are signals which are useful to a system user:

- **SIGHUP 1** – sent to a process when its controlling terminal is closed.
- **SIGINT 2** – sent to a process by its controlling terminal when a user interrupts the process by pressing `[Ctrl+C]`.
- **SIGQUIT 3** – sent to a process if the user sends a quit signal `[Ctrl+D]`.
- **SIGKILL 9** – this signal immediately terminates (kills) a process and the process will not perform any clean-up operations.
- **SIGTERM 15** – this a program termination signal (kill will send this by default).
- **SIGTSTP 20** – sent to a process by its controlling terminal to request it to stop (terminal stop); initiated by the user pressing `[Ctrl+Z]`.

Question: How To Change Priority Of A Processes In Linux?

On the Linux system, all active processes have a priority and certain nice value. Processes with higher priority will normally get more CPU time than lower priority processes.

However, a system user with root privileges can influence this with the **nice** and **renice** commands.

From the output of the top command, the NI shows the process nice value:

```
$ top
```

List Linux Running Processes

Use the **nice** command to set a nice value for a process. Keep in mind that normal users can attribute a nice value from zero to 20 to processes they own. Only the root user can use negative nice values.

To **renice** the priority of a process, use the **renice** command as follows:

```
$ renice +8  2687
$ renice +8  2103
```

GIT DevOps Interview Questions

Question: What is Git?

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.

It is primarily used for source code management in software development but it can be used to keep track of changes in any set of files.

As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvald. Git is an example of a Distributed Version Control System, In Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

Question: What Are Benefits Of GIT?

Here are some of the advantages of using Git

- Ease of use
- Data redundancy and replication
- High availability
- Superior disk utilization and network performance
- Only one .git directory per repository
- Collaboration friendly
- Any kind of projects from large to small scale can use GIT

Question: What Is Repository In GIT?

The purpose of Git is to manage a project, or a set of files, as they change over time. Git stores this information in a data structure called a repository. A git **repository** contains, among other things, the following:

- A set of **commit objects**.
- A set of references to commit objects, called **heads**.

The Git repository is stored in the same directory as the project itself, in a subdirectory called `.git`. Note differences from central-repository systems like CVS or Subversion:

- There is only one `.git` directory, in the root directory of the project.
- The repository is stored in files alongside the project. There is no central server repository.

Question: What Is Staging Area In GIT?

Staging is a step before the commit process in git. That is, a commit in git is performed in two steps:

-Staging and

-Actual commit

As long as a change set is in the staging area, git allows you to edit it as you like (replace staged files with other versions of staged files, remove changes from staging, etc.)

Question: What Is GIT STASH?

Often, when you've been working on part of your project, things are in a messy state and you want to switch branches for a bit to work on something else.

The problem is, you don't want to do a commit of half-done work just so you can get back to this point later. The answer to this issue is the `git stash` command. Stashing takes the dirty state of your working directory — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time.

Question: How To Revert Commit In GIT?

Given one or more existing commits, revert the changes that the related patches introduce, and record some new commits that record them. This requires your working tree to be clean (no modifications from the HEAD commit).

git-revert - Revert some existing commits

SYNOPSIS

```
git revert [--[no-]edit] [-n] [-m parent-number] [-s] [-S[<keyid>]] <commit>…
git revert --continue
git revert --quit
git revert --abort
```

Question: How To Delete Remote Repository In GIT?

Use the `git remote rm` command to remove a remote URL from your repository.
The `git remote rm` command takes one argument:

> A remote name, for example, `destination`

Questions: What Is GIT Stash Drop?

In case we do not need a specific stash, we use git stash drop command to remove it from the list of stashes.

By default, this command removes to latest added stash

To remove a specific stash we specify as argument in the git stash drop <stashname> command.

Question: What Is Difference Between GIT and Subversion?

Here is a summary of Differences between GIT and Subversion

- Git is a distributed VCS; SVN is a non-distributed VCS.
- Git has a centralized server and repository; SVN does not have a centralized server or repository.
- The content in Git is stored as metadata; SVN stores files of content.
- Git branches are easier to work with than SVN branches.
- Git does not have the global revision number feature like SVN has.
- Git has better content protection than SVN.
- Git was developed for Linux kernel by Linus Torvalds; SVN was developed by CollabNet, Inc.
- Git is distributed under GNU, and its maintenance overseen by Junio Hamano; Apache Subversion, or SVN, is distributed under the open source license.

Question: What Is Difference Between GIT Fetch & GIT Pull?

GIT fetch – It downloads only the new data from the remote repository and does not integrate any of the downloaded data into your working files. Providing a view of the data is all it does.

GIT pull – It downloads as well as merges the data from the remote repository into the local working files.

This may also lead to merging conflicts if the user's local changes are not yet committed. Using the "GIT stash" command hides the local changes.

Question: **What is Git fork? How to create tag?**

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

A fork is really a Github (not Git) construct to store a clone of the repo in your user account. As a clone, it will contain all the branches in the main repo at the time you made the fork.

Create Tag:

- Click the releases link on our repository page.
- Click on Create a new release or Draft a new release.
- Fill out the form fields, then click Publish release at the bottom.
- After you create your tag on GitHub, you might want to fetch it into your local repository too: git fetch.

Question: What is difference between fork and branch?

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
A fork is really a Github (not Git) construct to store a clone of the repo in your user account. As a clone, it will contain all the branches in the main repo at the time you made the fork.

Question: What Is Cherry Picking In GIT?

Cherry picking in git means to choose a commit from one branch and apply it onto another.

This is in contrast with other ways such as merge and rebase which normally applies many commits onto a another branch.

Make sure you are on the branch you want apply the commit to. git checkout master
Execute the following:

> git cherry-pick <commit-hash>

Question: What Language GIT Is Written In?

Much of Git is written in C, along with some BASH scripts for UI wrappers and other bits.

Question: How To Rebase Master In GIT?

Rebasing is the process of moving a branch to a new base commit.The golden rule of git rebase is to never use it on public branches.

The only way to synchronize the two master branches is to merge them back together, resulting in an extra merge commit and two sets of commits that contain the same changes.

Question: **What is ‘head’ in git and how many heads can be created in a repository?**

There can be any number of heads in a GIT repository. By default there is one head known as HEAD in each repository in GIT.

HEAD is a ref (reference) to the currently checked out commit. In normal states, it's actually a symbolic ref to the branch user has checked out.

if you look at the contents of **.git/HEAD** you'll see something like "ref: refs/**heads**/master". The branch itself is a reference to the commit at the tip of the branch

---

Question: Name some GIT commands and also explain their functions?

Here are some most important GIT commands

- **GIT diff** – It shows the changes between commits, commits and working tree.
- **GIT status** – It shows the difference between working directories and index.
- **GIT stash applies** – It is used to bring back the saved changes on the working directory.
- **GIT rm** – It removes the files from the staging area and also of the disk.
- **GIT log** – It is used to find the specific commit in the history.
- **GIT add** – It adds file changes in the existing directory to the index.
- **GIT reset** – It is used to reset the index and as well as the working directory to the state of the last commit.
- **GIT checkout** – It is used to update the directories of the working tree with those from another branch without merging.
- **GIT ls tree** – It represents a tree object including the mode and the name of each item.
- **GIT** instaweb – It automatically directs a web browser and runs the web server with an interface into your local repository.

Question: What is a “conflict” in GIT and how is it resolved?

When a commit that has to be merged has some changes in one place, which also has the changes of current commit, then the conflict arises.

The GIT will not be able to predict which change will take the precedence. In order to resolve the conflict in GIT: we have to edit the files to fix the conflicting changes and then add the resolved files by running the “GIT add” command; later on, to commit the

repaired merge run the “GIT commit” command. GIT identifies the position and sets the parents of the commit correctly.

Question: How To Migrate From Subversion To GIT?

SubGIT is a tool for smooth and stress-free subversion to GIT migration and also a solution for a company-wide subversion to GIT migration that is:

- It allows to make use of all GIT and subversion features.
- It provides genuine stress-free migration experience.
- It doesn’t require any change in the infrastructure that is already placed.
- It is considered to be much better than GIT-SVN

Question: What Is Index In GIT?

The index is a single, large, binary file in under .git folder, which lists all files in the current branch, their sha1 checksums, time stamps and the file name. Before completing the commits, it is formatted and reviewed in an intermediate area known as Index also known as the staging area.

Question: What is a bare Git repository?

A bare Git repository is a repository that is created without a Working Tree.

```
git init --bare
```

Question: WHow do you revert a commit that has already been pushed and made public??

One or more commits can be reverted through the use of *git revert*. This command, in essence, creates a new commit with patches that cancel out the changes introduced in specific commits.

In case the commit that needs to be reverted has already been published or changing the repository history is not an option, *git revert* can be used to revert commits. Running the following command will revert the last two commits:

```
git revert HEAD~2..HEAD
```

Alternatively, one can always checkout the state of a particular commit from the past, and commit it anew.

Question: How do you squash last N commits into a single commit?

Squashing multiple commits into a single commit will overwrite history, and should be done with caution. However, this is useful when working in feature branches.

To squash the last N commits of the current branch, run the following command (with {N} replaced with the number of commits that you want to squash):

```
git rebase -i HEAD~{N}
```

Upon running this command, an editor will open with a list of these N commit messages, one per line.

Each of these lines will begin with the word “pick”. Replacing “pick” with “squash” or “s” will tell Git to combine the commit with the commit before it.

To combine all N commits into one, set every commit in the list to be squash except the first one.

Upon exiting the editor, and if no conflict arises, *git rebase* will allow you to create a new commit message for the new combined commit.

Question: What is a conflict in git and how can it be resolved?

A conflict arises when more than one commit that has to be merged has some change in the same place or same line of code.

Git will not be able to predict which change should take precedence. This is a git conflict.

To resolve the conflict in git, edit the files to fix the conflicting changes and then add the resolved files by running `git add`.

After that, to commit the repaired merge, run `git commit`. Git remembers that you are in the middle of a merge, so it sets the parents of the commit correctly.

Question: How To Setup A Script To Run Every Time a Repository Receives New Commits Through Push?

To configure a script to run every time a repository receives new commits through push, one needs to define either a pre-receive, update, or a post-receive hook depending on when exactly the script needs to be triggered.

Pre-receive hook in the destination repository is invoked when commits are pushed to it. Any script bound to this hook will be executed before any references are updated.

This is a useful hook to run scripts that help enforce development policies.

Update hook works in a similar manner to pre-receive hook, and is also triggered before any updates are actually made.

However, the update hook is called once for every commit that has been pushed to the destination repository.

Finally, post-receive hook in the repository is invoked after the updates have been accepted into the destination repository.

This is an ideal place to configure simple deployment scripts, invoke some continuous integration systems, dispatch notification emails to repository maintainers, etc.

Hooks are local to every Git repository and are not versioned. Scripts can either be created within the hooks directory inside the “.git” directory, or they can be created elsewhere and links to those scripts can be placed within the directory.

Question: What Is Commit Hash?

In Git each commit is given a unique hash. These hashes can be used to identify the corresponding commits in various scenarios (such as while trying to checkout a particular state of the code using the *git checkout {hash}* command).

Additionally, Git also maintains a number of aliases to certain commits, known as refs.

Also, every tag that you create in the repository effectively becomes a ref (and that is exactly why you can use tags instead of commit hashes in various git commands).

Git also maintains a number of special aliases that change based on the state of the repository, such as HEAD, FETCH_HEAD, MERGE_HEAD, etc.

Git also allows commits to be referred as relative to one another. For example, HEAD~1 refers to the commit parent to HEAD, HEAD~2 refers to the grandparent of HEAD, and so on.

In case of merge commits, where the commit has two parents, ^ can be used to select one of the two parents, e.g. HEAD^2 can be used to follow the second parent.

And finally, refspecs. These are used to map local and remote branches together.

However, these can be used to refer to commits that reside on remote branches allowing one to control and manipulate them from a local Git environment.

Question: What Is Conflict In GIT?

A conflict arises when more than one commit that has to be merged has some change in the same place or same line of code.

Git will not be able to predict which change should take precedence. This is a git conflict.To resolve the conflict in git, edit the files to fix the conflicting changes and then add the resolved files by running `git add`. After that, to commit the repaired merge, run `git commit`. Git remembers that you are in the middle of a merge, so it sets the parents of the commit correctly.

Question: **What are git hooks?**?

Git hooks are scripts that can run automatically on the occurrence of an event in a Git repository. These are used for automation of workflow in GIT. Git hooks also help in customizing the internal behavior of GIT. These are generally used for enforcing a GIT commit policy.

Question: What Are Disadvantages Of GIT?

GIT has very few disadvantages. These are the scenarios when GIT is difficult to use.

Some of these are:

Binary Files: If we have a lot binary files (non-text) in our project, then GIT becomes very slow. E.g. Projects with a lot of images or Word documents.

Steep Learning Curve: It takes some time for a newcomer to learn GIT. Some of the GIT commands are non-intuitive to a fresher.

Slow remote speed: Sometimes the use of remote repositories in slow due to network latency. Still GIT is better than other VCS in speed.

Question: What is stored inside a commit object in GIT?

GIT commit object contains following information:

SHA1 name: A 40 character string to identify a commit

Files: List of files that represent the state of a project at a specific point of time

Reference: Any reference to parent commit objects

Question: What Is GIT reset command?

Git reset command is used to reset current HEAD to a specific state. By default it reverses the action of git add command. So we use git reset command to undo the changes of git add command. Reference: Any reference to parent commit objects

Question: **How GIT protects the code in a repository**?

GIT is made very secure since it contains the source code of an organization. All the objects in a GIT repository are encrypted with a hashing algorithm called SHA1.

This algorithm is quite strong and fast. It protects source code and other contents of repository against the possible malicious attacks.

This algorithm also maintains the integrity of GIT repository by protecting the change history against accidental changes.

Continuos Integration Interview Questions

Question: What is Continuos Integration?

Continuous Integration is the process of continuously integrating the code and often multiple times per day. The purpose is to find problems quickly, s and deliver the fixes more rapidly.

CI is a best practice for software development. It is done to ensure that after every code change there is no issue in software.

Question: What Is Build Automation?

Build automation is the process of automating the creation of a software build and the associated processes.

Including compiling computer source code into binary code, packaging binary code, and running automated tests.

Question: What Is Automated Deployment?

Automated Deployment is the process of consistently pushing a product to various environments on a “trigger.”

It enables you to quickly learn what to expect every time you deploy an environment with much faster results.

This combined with Build Automation can save development teams a significant amount of hours.

Automated Deployment saves clients from being extensively offline during development and allows developers to build while “touching” fewer of a clients’ systems.

With an automated system, human error is prevented. In the event of human error, developers are able to catch it before live deployment – saving time and headache.

You can even automate the contingency plan and make the site rollback to a working or previous state as if nothing ever happened.

Clearly, this automated feature is super valuable in allowing applications and sites to continue during fixes.

Additionally, contingency plans can be version-controlled, improved and even self-tested.

Question: How Continuous Integration Implemented?

Different tools for supporting Continuous Integration are Hudson, Jenkins and Bamboo. Jenkins is the most popular one currently. They provide integration with various version control systems and build tools.

Question: How Continuous Integration process does work?

Whenever developer commits changes in version control system, then Continuous Integration server detects that changes are committed. And goes through following process

- Continuous Integration server retrieves latest copy of changes.
- It build code with new changes in build tools.
- If build fails notify to developer.
- After build pass run automated test cases if test cases fail notify to developer.
- Create package for deployment environment.

Question: What Are The Software Required For Continuous Integration process?

Here are the minimum tools you need to achieve CI

- Source code repository : To commit code and changes for example git.
- Server: It is Continuous Integration software for example Jenkin, Teamcity.

- Build tool: It builds application on particular way for example maven, gradle.
- Deployment environment : On which application will be deployed.

Question: **What Is Jenkins Software?**

Jenkins is self-contained, open source automation server used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Jenkins is one of the leading open source automation servers available. Jenkins has an extensible, plugin-based architecture, enabling developers to create 1,400+ plugins to adapt it to a multitude of build, test and deployment technology integrations.

Questions: What is a Jenkins Pipeline?

Jenkins Pipeline (or simply “Pipeline”) is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins..

Question: **What is the difference between Maven, Ant,Gradle and Jenkins** ?

Maven and Ant are Build Technologies whereas Jenkins is a continuous integration tool.

Question: **Why do we use Jenkins?**

Jenkins is an **open-source** continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real time.

The **Jenkins software** enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.

Question: **What are CI Tools?**?

Here is the list of the top 8 **Continuous Integration tools**:

- Jenkins
- TeamCity
- Travis CI
- Go CD
- Bamboo
- GitLab CI

- CircleCI
- Codeship

Question: **Which SCM tools Jenkins supports?**?

Jenkins supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and arbitrary shell scripts and Windows batch commands.

Question: **Why do we use Pipelines in Jenkins**?

Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive continuous delivery pipelines.

By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

- Code: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- Durable: Pipelines can survive both planned and unplanned restarts of the Jenkins master.
- Pausable: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- Versatile: Pipelines support complex real-world continuous delivery requirements, including the ability to fork/join, loop, and perform work in parallel.
- Extensible: The Pipeline plugin supports custom extensions to its DSL and multiple options for integration with other plugins.

Question: **How do you create Multibranch Pipeline in Jenkins**?

The Multi branch Pipeline project type enables you to implement different Jenkins files for different branches of the same project.

In a Multi branch Pipeline project, Jenkins automatically discovers, manages and executes Pipelines for branches which contain a Jenkins file in source control.

Question: **What are Jobs in Jenkins??**

Jenkins can be used to perform the typical build server work, such as doing continuous/official/nightly builds, run tests, or perform some repetitive batch tasks. This is called “**free-style software project**” in Jenkins.

Question: **How do you configuring automatic builds in Jenkins**?

Builds in Jenkins can be triggered periodically (on a schedule, specified in configuration), or when source changes in the project have been detected, or they can be automatically triggered by requesting the URL:

Question: **What is a Jenkins file?**

Jenkins file is a text file containing the definition of a Jenkins Pipeline and checks into source control.

Amazon AWS DevOps Interview Questions

Question: What is **Amazon Web Services**?

Amazon Web Services provides services that help you practice DevOps at your company and that are built first for use with AWS.

These tools automate manual tasks, help teams manage complex environments at scale, and keep engineers in control of the high velocity that is enabled by DevOps

Question: What Are Benefits Of AWS for DevOps?

There are many benefits of using AWS for devops

Get Started Fast: Each AWS service is ready to use if you have an AWS account. There is no setup required or software to install.

Fully Managed Services: These services can help you take advantage of AWS resources quicker. You can worry less about setting up, installing, and operating infrastructure on your own. This lets you focus on your core product.

Built For Scalability: You can manage a single instance or scale to thousands using AWS services. These services help you make the most of flexible compute resources by simplifying provisioning, configuration, and scaling.

Programmable: You have the option to use each service via the AWS Command Line Interface or through APIs and SDKs. You can also model and provision AWS resources and your entire AWS infrastructure using declarative AWS CloudFormation templates.

Automation: AWS helps you use automation so you can build faster and more efficiently. Using AWS services, you can automate manual tasks or processes such as deployments,

development & test workflows, container management, and configuration management.

Secure: Use AWS Identity and Access Management (IAM) to set user permissions and policies. This gives you granular control over who can access your resources and how they access those resources.

Question: **How To Handle Continuous Integration and Continuous Delivery in AWS** Devops?

The AWS Developer Tools help in securely store and version your application's source code and automatically build, test, and deploy your application to AWS.

Question: What Is The Importance Of Buffer In Amazon Web Services?

An Elastic Load Balancer ensures that the incoming traffic is distributed optimally across various AWS instances.

A buffer will synchronize different components and makes the arrangement additional elastic to a burst of load or traffic.

The components are prone to work in an unstable way of receiving and processing the requests.

The buffer creates the equilibrium linking various apparatus and crafts them effort at the identical rate to supply more rapid services.

Question: What Are The Components Involved In Amazon Web Services?

There are 4 components

Amazon S3 : with this, one can retrieve the key information which are occupied in creating cloud structural design and amount of produced information also can be stored in this component that is the consequence of the key specified.

Amazon EC2 instance : helpful to run a large distributed system on the Hadoop cluster. Automatic parallelization and job scheduling can be achieved by this component.

Amazon SQS : this component acts as a mediator between different controllers. Also worn for cushioning requirements those are obtained by the manager of Amazon.

Amazon SimpleDB : helps in storing the transitional position log and the errands executed by the consumers.

Question: How is a Spot instance different from an On-Demand instance or Reserved Instance?

Spot Instance, On-Demand instance and Reserved Instances are all models for pricing. Moving along, spot instances provide the ability for customers to purchase compute capacity with no upfront commitment, at hourly rates usually lower than the On-Demand rate in each region.

Spot instances are just like bidding, the bidding price is called Spot Price. The Spot Price fluctuates based on supply and demand for instances, but customers will never pay more than the maximum price they have specified.

If the Spot Price moves higher than a customer's maximum price, the customer's EC2 instance will be shut down automatically.

But the reverse is not true, if the Spot prices come down again, your EC2 instance will not be launched automatically, one has to do that manually.

In Spot and On demand instance, there is no commitment for the duration from the user side, however in reserved instances one has to stick to the time period that he has chosen.

Questions: **What are the best practices for Security in Amazon EC2?**

There are several best practices to secure Amazon EC2. A few of them are given below:

- Use AWS Identity and Access Management (IAM) to control access to your AWS resources.
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- Review the rules in your security groups regularly, and ensure that you apply the principle of least
- Privilege – only open up permissions that you require.
- Disable password-based logins for instances launched from your AMI. Passwords can be found or cracked, and are a security risk.

Question: **What is AWS CodeBuild in AWS** Devops**?**

AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy.

With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue.

Question: **What is Amazon Elastic Container Service in AWS** Devops**?**

Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.

Question: **What is AWS Lambda in AWS Devops?**

AWS Lambda lets you run code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.
Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.

Splunk DevOps Interview Questions

Question: What is Splunk?

The platform of Splunk allows you to get visibility into machine data generated from different networks, servers, devices, and hardware.

It can give insights into the application management, threat visibility, compliance, security, etc. so it is used to analyze machine data. The data is collected from the forwarder from the source and forwarded to the indexer. The data is stored locally on a host machine or cloud. Then on the data stored in the indexer the search head searches, visualizes, analyzes and performs various other functions.

Question: What Are The Components Of Splunk?

The main components of Splunk are Forwarders, Indexers and Search Heads.Deployment Server(or Management Console Host) will come into the picture in case of a larger environment.

Deployment servers act like an antivirus policy server for setting up Exceptions and Groups so that you can map and create adifferent set of data collection policies each for either window based server or a Linux based server or a Solaris based server. **plunk has four important components :**

- **Indexer –** It indexes the machine data
- **Forwarder –** Refers to Splunk instances that forward data to the remote indexers
- **Search Head –** Provides GUI for searching
- **Deployment Server –**Manages the Splunk components like indexer, forwarder, and

search head in computing environment.

Question: **What are alerts in Splunk?**

An alert is an action that a saved search triggers on regular intervals set over a time range, based on the results of the search.

When the alerts are triggered, various actions occur consequently.. For instance, sending an email when a search to the predefined list of people is triggered.

Three types of alerts:

1. **Pre-result alerts :** Most commonly used alert type and runs in real-time for an all-time span. These alerts are designed such that whenever a search returns a result, they are triggered.
2. **Scheduled alerts :** The second most common- scheduled results are set up to evaluate the results of a historical search result running over a set time range on a regular schedule. You can define a time range, schedule and the trigger condition to an alert.
3. **Rolling-window alerts:** These are the hybrid of pre-result and scheduled alerts. Similar to the former, these are based on real-time search but do not trigger each time the search returns a matching result . It examines all events in real-time mapping within the rolling window and triggers the time that specific condition by that event in the window is met, like the scheduled alert is triggered on a scheduled search.

Question: What Are The Categories Of SPL Commands?

SPL commands are divided into five categories:

1. **Sorting Results –** Ordering results and (optionally) limiting the number of results.
2. **Filtering Results –** It takes a set of events or results and filters them into a smaller set of results.
3. **Grouping Results –** Grouping events so you can see patterns.
4. **Filtering, Modifying and Adding Fields –** Taking search results and generating a summary for reporting.
5. **Reporting Results –** Filtering out some fields to focus on the ones you need, or modifying or adding fields to enrich your results or events.

Question: **What Happens If The License Master Is Unreachable?**

In case the license master is unreachable, then it is just not possible to search the data.

However, the data coming in to the Indexer will not be affected. The data will continue to flow into your Splunk deployment.

The Indexers will continue to index the data as usual however, you will get a warning message on top your Search head or web UI saying that you have exceeded the indexing volume.

And you either need to reduce the amount of data coming in or you need to buy a higher capacity of license. Basically, the candidate is expected to answer that the indexing does not stop; only searching is halted

---

Question: What are common port numbers used by Splunk?

Common port numbers on which default services run are:

Service	Port Number
Splunk Management Port	8089
Splunk Index Replication Port	8080
KV store	8191
Splunk Web Port	8000
Splunk Indexing Port	9997
Splunk network port	514

Question: What Are Splunk Buckets? Explain The Bucket Lifecycle?

A directory that contains indexed data is known as a Splunk bucket. It also contains events of a certain period. Bucket lifecycle includes following stages:

- **Hot –** It contains newly indexed data and is open for writing. For each index, there are one or more hot buckets available
- **Warm –** Data rolled from hot
- **Cold –** Data rolled from warm
- **Frozen –** Data rolled from cold. The indexer deletes frozen data by default but users can also archive it.
- **Thawed –** Data restored from an archive. If you archive frozen data , you can later return it to the index by thawing (defrosting) it.

Question: **Explain Data Models and Pivot?**

Data models are used for creating a structured hierarchical model of data. It can be used when you have a large amount of unstructured data, and when you want to make use of that information without using complex search queries.

A few use cases of Data models are:

- **Create Sales Reports:** If you have a sales report, then you can easily create the total number of successful purchases, below that you can create a child object containing the list of failed purchases and other views
- **Set Access Levels:** If you want a structured view of users and their various access levels, you can use a data model

On the other hand with pivots, you have the flexibility to create the front views of your results and then pick and choose the most appropriate filter for a better view of results.

Question: What Is **File Precedence In Splunk?**

File precedence is an important aspect of troubleshooting in Splunk for an administrator, developer, as well as an architect.

All of Splunk's configurations are written in .conf files. There can be multiple copies present for each of these files, and thus it is important to know the role these files play when a Splunk instance is running or restarted. To determine the priority among copies of a configuration file, Splunk software first determines the directory scheme. The directory schemes are either a) Global or b) App/user. When the context is global (that is, where there's no app/user context), directory priority descends in this order:

1. System local directory — *highest priority*
2. App local directories
3. App default directories
4. System default directory — *lowest priority*

When the context is app/user, directory priority descends from user to app to system:

1. User directories for current user — *highest priority*
2. App directories for currently running app (local, followed by default)
3. App directories for all other apps (local, followed by default) — for exported settings only
4. System directories (local, followed by default) — *lowest priority*

Question: D**ifference Between Search Time And Index Time Field Extractions?**

Search time field extraction refers to the fields extracted while performing searches.

Whereas, fields extracted when the data comes to the indexer are referred to as Index time field extraction.

You can set up the indexer time field extraction either at the forwarder level or at the indexer level.

Another difference is that Search time field extraction's extracted fields are not part of the metadata, so they do not consume disk space.

Whereas index time field extraction's extracted fields are a part of metadata and hence consume disk space.

Question: **What Is Source Type In Splunk?**

Source type is a default field which is used to identify the data structure of an incoming event. Source type determines how Splunk Enterprise formats the data during the indexing process.
Source type can be set at the forwarder level for indexer extraction to identify different data formats.

Question: What is SOS?

SOS stands for Splunk on Splunk. It is a Splunk app that provides graphical view of your Splunk environment performance and issues.

It has following purposes:

- Diagnostic tool to analyze and troubleshoot problems
- Examine Splunk environment performance
- Solve indexing performance issues
- Observe scheduler activities and issues
- See the details of scheduler and user driven search activity
- Search, view and compare configuration files of Splunk

Question: What Is Splunk Indexer And Explain Its Stages?

The indexer is a Splunk Enterprise component that creates and manages indexes. The main functions of an indexer are:

- Indexing incoming data
- Searching indexed data **Splunk indexer has following stages:**

Input : Splunk Enterprise acquires the raw data from various input sources and breaks it into 64K blocks and assign them some metadata keys. These keys include host, source and source type of the data. **Parsing :** Also known as event processing, during this stage, the Enterprise analyzes and transforms the data, breaks data into streams, identifies, parses and sets timestamps, performs metadata annotation and transformation of data. **Indexing :** In this phase, the parsed events are written on the disk index including both compressed data and the associated index files. **Searching :** The 'Search' function plays a

major role during this phase as it handles all searching aspects (interactive, scheduled searches, reports, dashboards, alerts) on the indexed data and stores saved searches, events, field extractions and views

Question: State The Difference Between Stats and Eventstats Commands?

Stats – This command produces summary statistics of all existing fields in your search results and store them as values in new fields. **Eventstats –** It is same as stats command except that aggregation results are added in order to every event and only if the aggregation is applicable to that event. It computes the requested statistics similar to stats but aggregates them to the original raw data.

log4J DevOps Interview Questions

Question: What is log4j?

log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License.

log4j has been ported to the C, C++, C#, Perl, Python, Ruby, and Eiffel languages.

log4j is highly configurable through external configuration files at runtime. It views the logging process in terms of levels of priorities and offers mechanisms to direct logging information to a great variety of destinations.

Question: **What Are The Features Of Log4j**

Log4j is widely used framework and here are features of log4j

- It is thread-safe.It is optimized for speed
- It is based on a named logger hierarchy.
- It supports multiple output appenders per logger.
- It supports internationalization.
- It is not restricted to a predefined set of facilities.
- Logging behavior can be set at runtime using a configuration file.
- It is designed to handle Java Exceptions from the start.
- It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL.
- The format of the log output can be easily changed by extending the Layout class.
- The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface.
- It is fail-stop. However, although it certainly strives to ensure delivery, log4j does not

guarantee that each log statement will be delivered to its destination.

Question: What are the components of log4j**?**

log4j has three main components

- loggers: Responsible for capturing logging information.
- appenders: Responsible for publishing logging information to various preferred destinations.
- layouts: Responsible for formatting logging information in different styles.

Question: How do you initialize and use Log4J ?

public class LoggerTest { static Logger log = Logger.getLogger (LoggerTest.class.getName()); public void my logerMethod() { if(log.isDebugEnabled()) log.debug("This is test message" + var2);) } }

Question: **What are Pros and Cons of Logging?**

Following are the Pros and Cons of Logging Logging is an important component of the software development. A well-written logging code offers quick debugging, easy maintenance, and structured storage of an application's runtime information. Logging does have its drawbacks also. It can slow down an application. If too verbose, it can cause scrolling blindness. To alleviate these concerns, log4j is designed to be reliable, fast and extensible. Since logging is rarely the main focus of an application, the log4j API strives to be simple to understand and to use.

Question:**What Is The Purpose Of Logger Object?**

Logger Object − The top-level layer of log4j architecture is the Logger which provides the Logger object.

The Logger object is responsible for capturing logging information and they are stored in a namespace hierarchy.

Question: What is the purpose of Layout object?

The layout layer of log4j architecture provides objects which are used to format logging information in different styles. It provides support to appender objects before publishing logging information.

Layout objects play an important role in publishing logging information in a way that is human-readable and reusable.

Questions: What is the purpose of Appender object?

The Appender object is responsible for publishing logging information to various preferred destinations such as a database, file, console, UNIX Syslog, etc.

Question: **What Is The Purpose Of ObjectRenderer Object?**

The ObjectRenderer object is specialized in providing a String representation of different objects passed to the logging framework.

This object is used by Layout objects to prepare the final logging information.

Question: **What Is LogManager object?**

The LogManager object manages the logging framework. It is responsible for reading the initial configuration parameters from a system-wide configuration file or a configuration class.

Question: **How Will You Define A File Appender Using Log4j.properties?**

Following syntax defines a file appender −
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

Question: W**hat Is The Purpose Of Threshold In Appender?**

Appender can have a threshold level associated with it independent of the logger level. The Appender ignores any logging messages that have a level lower than the threshold level.

Docker DevOps Interview Questions

Question: What is Docker?

Docker provides a container for managing software workloads on shared infrastructure, all while keeping them isolated from one another.

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.

Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

By doing so, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system. Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

Question: **What Are Linux Containers**?

Linux containers, in short, contain applications in a way that keep them isolated from the host system that they run on.

Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

And they are designed to make it easier to provide a consistent experience as developers and system administrators move code from development environments into production in a fast and replicable way.

Question: Who Is Docker For**?**

Docker is a tool that is designed to benefit both developers and system administrators, making it a part of many DevOps (developers + operations) toolchains.

For developers, it means that they can focus on writing code without worrying about the system that it will ultimately be running on.

It also allows them to get a head start by using one of thousands of programs already designed to run in a Docker container as a part of their application.

For operations staff, Docker gives flexibility and potentially reduces the number of systems needed because of its small footprint and lower overhead.

Question: **What Is Docker Container?**

Docker containers include the application and all of its dependencies, but share the kernel with other containers, running as isolated processes in user space on the host operating system.

Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.

Now explain how to create a Docker container, Docker containers can be created by either creating a Docker image and then running it or you can use Docker images that are present on the Dockerhub. Docker containers are basically runtime instances of Docker images.

Question: **What Is Docker Image?**

Docker image is the source of Docker container. In other words, Docker images are used to create containers.

Images are created with the build command, and they'll produce a container when started with run.

Images are stored in a Docker registry such as registry.hub.docker.com because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.

Question: **What Is Docker Hub?**

Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts.

It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

Question: What is Docker Swarm?

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host.

Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

I will also suggest you to include some supported tools:

- Dokku
- Docker Compose
- Docker Machine
- Jenkins

Questions: **What is Dockerfile used for**?

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

Using docker build users can create an automated build that executes several command-line instructions in succession.

Question: **How is Docker different from other container technologies?**

Docker containers are easy to deploy in a cloud. It can get more applications running on the same hardware than other technologies.

It makes it easy for developers to quickly create, ready-to-run containerized applications and it makes managing and deploying applications much easier. You can even share containers with your applications.

Question: **How to create Docker container?**

We can use Docker image to create Docker container by using the below command:

```
docker run -t -i command name
```

This command will create and start a container. You should also add, If you want to check the list of all running container with the status on a host use the below command:

```
docker ps -a
```

Question: **How to stop and restart the Docker container?**

In order to stop the Docker container you can use the below command:

```
docker stop container ID
```

Now to restart the Docker container you can use:

```
docker restart container ID
```

Question: What is the difference between docker run and docker create?

The primary difference is that using **‘docker create’** creates a container in a stopped state. **Bonus point:** You can use **‘docker create’** and store an outputed container ID for later use. The best way to do it is to use **‘docker run’** with **--cidfile FILE_NAME** as running it again won’t allow to overwrite the file.

Question: What four states a Docker container can be in?

- Running
- Paused
- Restarting
- Exited

Question:What Is Difference Between Repository and a Registry?

Docker registry is a service for hosting and distributing images. Docker repository is a collection of related Docker images.

Question: How to link containers?

The simplest way is to use network port mapping. There’s also the **- -link** flag which is deprecated.

Question: What is the difference between Docker RUN, CMD and ENTRYPOINT?

A **CMD** does not execute anything at build time, but specifies the intended command for the image.
RUN actually runs a command and commits the result.
If you would like your container to run the same executable every time, then you should consider using **ENTRYPOINT** in combination with CMD.

Question: How many containers can run per host?

As far as the number of containers that can be run, this really depends on your

environment. The size of your applications as well as the amount of available resources will all affect the number of containers that can be run in your environment.
Containers unfortunately are not magical. They can't create new CPU from scratch. They do, however, provide a more efficient way of utilizing your resources.
The containers themselves are super lightweight (remember, shared OS vs individual OS per container) and only last as long as the process they are running. Immutable infrastructure if you will.

Question: **What is Docker hub?**

Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts.
It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

VmWare DevOps Interview Questions

Question: What is VmWare?

VMware was founded in 1998 by five different IT experts. The company officially launched its first product, VMware Workstation, in 1999, which was followed by the VMware GSX Server in 2001. The company has launched many additional products since that time. VMware's desktop software is compatible with all major OSs, including Linux, Microsoft Windows, and Mac OS X. VMware provides three different types of desktop software:

- VMware Workstation: This application is used to install and run multiple copies or instances of the same operating systems or different operating systems on a single physical computer machine.
- VMware Fusion: This product was designed for Mac users and provides extra compatibility with all other VMware products and applications.
- VMware Player: This product was launched as freeware by VMware for users who do not have licensed VMWare products. This product is intended only for personel use.

VMware's software hypervisors intended for servers are bare-metal embedded hypervisors that can run directly on the server hardware without the need of an extra primary OS. VMware's line of server software includes:

- VMware ESX Server: This is an enterprise-level solution, which is built to provide better functionality in comparison to the freeware VMware Server resulting from a lesser system overhead. VMware ESX is integrated with VMware vCenter that provides additional solutions to improve the manageability and consistency of the server implementation.
- VMware ESXi Server: This server is similar to the ESX Server except that the service

console is replaced with BusyBox installation and it requires very low disk space to operate.
- VMware Server: Freeware software that can be used over existing operating systems like Linux or Microsoft Windows.

Question: **What is Virtualization?**

The process of creating virtual versions of physical components i-e Servers, Storage Devices, Network Devices on a physical host is called virtualization.

Virtualization lets you run multiple virtual machines on a single physical machine which is called ESXi host.

Question: **What are different types of virtualization?**

There are 5 basic types of virtualization

- Server virtualization: consolidates the physical server and multiple OS can be run on a single server.
- Network Virtualization: Provides complete reproduction of physical network into a software defined network.
- Storage Virtualization: Provides an abstraction layer for physical storage resources to manage and optimize in virtual deployment.
- Application Virtualization: increased mobility of applications and allows migration of VMs from host on another with minimal downtime.
- Desktop Virtualization: virtualize desktop to reduce cost and increase service

---

Question: **What is Service Console?**

The service console is developed based up on Redhat Linux Operating system, it is used to manage the VMKernel

---

Question: **What is vCenter Agent?**

VC agent is an agent installed on ESX server which enables communication between VC and ESX server.

This Agent will be installed on ESX/ESXi will be done when you try to add the ESx host in Vcenter.

Question: **What is VMKernel?**

VMWare Kernel is a Proprietary kernel of vmware and is not based on any of the flavors of Linux operating systems.

VMkernel requires an operating system to boot and manage the kernel. A service console is being provided when VMWare kernel is booted.

Only service console is based up on Redhat Linux OS not VMkernel.

Question:**What is VMKernel and why it is important?**

VMkernel is a virtualization interface between a Virtual Machine and the ESXi host which stores VMs.

It is responsible to allocate all available resources of ESXi host to VMs such as memory, CPU, storage etc.

It’s also control special services such as vMotion, Fault tolerance, NFS, traffic management and iSCSI.

To access these services, VMkernel port can be configured on ESXi server using a standard or distributed vSwitch. Without VMkernel, hosted VMs cannot communicate with ESXi server.

Question: **What is hypervisor and its types?**

Hypervisor is a virtualization layer that enables multiple operating systems to share a single hardware host.

Each operating system or VM is allocated physical resources such as memory, CPU, storage etc by the host. There are two types of hypervisors

- Hosted hypervisor (works as application i-e VMware Workstation)
- Bare-metal (is virtualization software i-e VMvisor, hyper-V which is installed directly onto the hardware and controls all physical resources).

Questions: **What is virtual networking?**

A network of VMs running on a physical server that are connected logically with each other is called virtual networking.

Question: **What is vSS?**

---

vSS stands for Virtual Standard Switch is responsible for communication of VMs hosted on a single physical host.

it works like a physical switch automatically detects a VM which want to communicate with other VM on a same physical server.

Question: **What is VMKernal adapter and why it used?**

AVMKernel adapter provides network connectivity to the ESXi host to handle network traffic for vMotion, IP Storage, NAS, Fault Tolerance, and vSAN.

For each type of traffic such as vMotion, vSAN etc. separate VMKernal adapter should be created and configured.

Question:**What are three port groups are configured in ESXi networking?**

- Virtual Machine Port Group – Used for Virtual Machine Network
- Service Console Port Group – Used for Service Console Communications
- VMKernel Port Group – Used for VMotion, iSCSI, NFS Communications

Question: W**hat are main components of vCenter Server architecture?**

There are three main components of vCenter Server architecture.

- vSphere Client and Web Client: a user interface.
- vCenter Server database: SQL server or embedded PostgreSQL to store inventory, security roles, resource pools etc.
- SSO: a security domain in virtual environment

Question: **What is datastore?**

A Datastore is a storage location where virtual machine files are stored and accessed. Datastore is based on a file system which is called VMFS, NFS

Question: **How many disk types are in VMware?**

There are three disk types in vSphere.

1. Thick Provisioned Lazy Zeroes: every virtual disk is created by default in this disk format. Physical space is allocated to a VM when virtual disk is created. It can’t be converted to thin disk.
2. Thick Provision Eager Zeroes: this disk type is used in VMware Fault Tolerance. All required disk space is allocated to a VM at time of creation. It takes more time to create a virtual disk compare to other disk formats.

3. Thin provision: It provides on-demand allocation of disk space to a VM. When data size grows, the size of disk will grow. Storage capacity utilization can be up to 100% with thin provisioning.
4. What is Storage vMotion?

It is similar to traditional vMotion, in Storage vMotion, virtual disk of a VM is moved from datastore to another. During Storage vMotion, virtual disk types think provisioning disk can be transformed to thin provisioned disk.

Question: **What is the use of VMKernel Port ?**

Vmkernel port is used by ESX/ESXi for vmotion, ISCSI & NFS communications. ESXi uses Vmkernel as the management network since it don't have serviceconsole built with it.

Question: **What are different types of Partitions in ESX server?**

AC/-root Swap /var /Var/core /opt /home /tmp

Question: E**xplain What Is VMware DRS?**

VMware DRS stands for Distributed Resource Scheduler; it dynamically balances resources across various host under cluster or resource pool. It enables users to determine the rules and policies which decide how virtual machines deploy resources, and these resources should be prioritized to multiple virtual machines.

DevOps Testing Interview Questions

Question: **What is Continuous Testing?**

Continuous Testing is the process of executing automated tests to obtain immediate feedback on the business risks associated with in the latest build.

In this way, each build is tested continuously, allowing Development teams to get fast feedback so that they can prevent those problems from progressing to the next stage of Software delivery life-cycle.

Question: **What is Automation Testing**

Automation testing is a process of automating the manual testing process. Automation testing involves use of separate testing tools, which can be executed repeatedly and

doesn’t require any manual intervention.

Question: **What Are The Benefits of Automation Testing?**

Here are some of the benefits of using Continuous Testing;

- Supports execution of repeated test cases
- Aids in testing a large test matrix
- Enables parallel execution
- Encourages unattended execution
- Improves accuracy thereby reducing human generated errors
- Saves time and money

Question: **Why is Continuous Testing important for DevOps?**

Continuous Testing allows any change made in the code to be tested immediately.

This avoids the problems created by having “big-bang” testing left to the end of the development cycle such as release delays and quality issues.

In this way, Continuous Testing facilitates more frequent and good quality releases.”

Question: **What are the Testing types supported by Selenium?**

Selenium supports two types of testing:

Regression Testing: It is the act of retesting a product around an area where a bug was fixed.

Functional Testing: It refers to the testing of software features (functional points) individually.

Question: **What is the Difference Between Assert and Verify commands in Selenium?**

Assert command checks whether the given condition is true or false.

Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn’t halts i.e. any failure during verification would not stop the execution and all the test steps would be executed.

Summary

DevOps refers to a wide range of tools, process and practices used bycompanies to improve their build, deployment, testing and release life cycles.

In order to ace a DevOps interview you need to have a deep understanding of all of these tools and processes.

Most of the technologies and process used to implement DevOps are not isolated. Most probably you are already familiar with many of these. All you have to do is to prepare for these from DevOps perspective.

In this guide I have created the largest set of interview questions. Each section in this guide caters to a specific area of DevOps.

In order to increase your chances of success in DevOps interview you need to go through all of these questions.

Other Related Interview Questions:

- AngularJs Interview Questions

- Spring Interview Questions

- Java MultiThreading Interview Questions

- Interview Questions

- Phone Interview Questions

DevOps Interview Questions PDF

Spring-Interview-Questions

About The Author

References

- https://theagileadmin.com/what-is-devops/
- https://en.wikipedia.org/wiki/DevOps
- http://www.javainuse.com/misc/gradle-interview-questions
- https://mindmajix.com/gradle-interview-questions
- https://tekslate.com/groovy-interview-questions-and-answers/
- https://mindmajix.com/groovy-interview-questions
- https://www.wisdomjobs.com/e-university/groovy-programming-language-interview-questions.html
- https://www.quora.com/What-are-some-advantages-of-the-Groovy-programming-language

- https://www.quora.com/What-are-some-advantages-of-the-Groovy-programming-language
- http://groovy-lang.org/documentation.html
- https://maven.apache.org/guides/introduction/introduction-to-archetypes.html
- https://en.wikipedia.org/wiki/Apache_Maven
- https://www.tecmint.com/linux-process-management/
- https://www.tecmint.com/dstat-monitor-linux-server-performance-process-memory-network/
- https://www.careerride.com/Linux-Interview-Questions.aspx
- https://www.onlineinterviewquestions.com/git-interview-questions/#.WxcTP9WFMy4
- https://www.atlassian.com/git/tutorials/what-is-git
- https://www.toptal.com/git/interview-questions
- https://www.sbf5.com/~cduan/technical/git/git-1.shtml
- http://preparationforinterview.com/preparationforinterview/continuous-integration-interview-question
- https://codingcompiler.com/jenkins-interview-questions-answers/
- https://www.edureka.co/blog/interview-questions/top-splunk-interview-questions-and-answers/
- https://intellipaat.com/interview-question/splunk-interview-questions/
- https://www.edureka.co/blog/interview-questions/docker-interview-questions/
- http://www.vmwarearena.com/vmware-interview-questions-and-answers/
- https://www.myvirtualjourney.com/top-80-vmware-interview-questions-answers/
- https://www.edureka.co/blog/interview-questions/top-devops-interview-questions-2016/